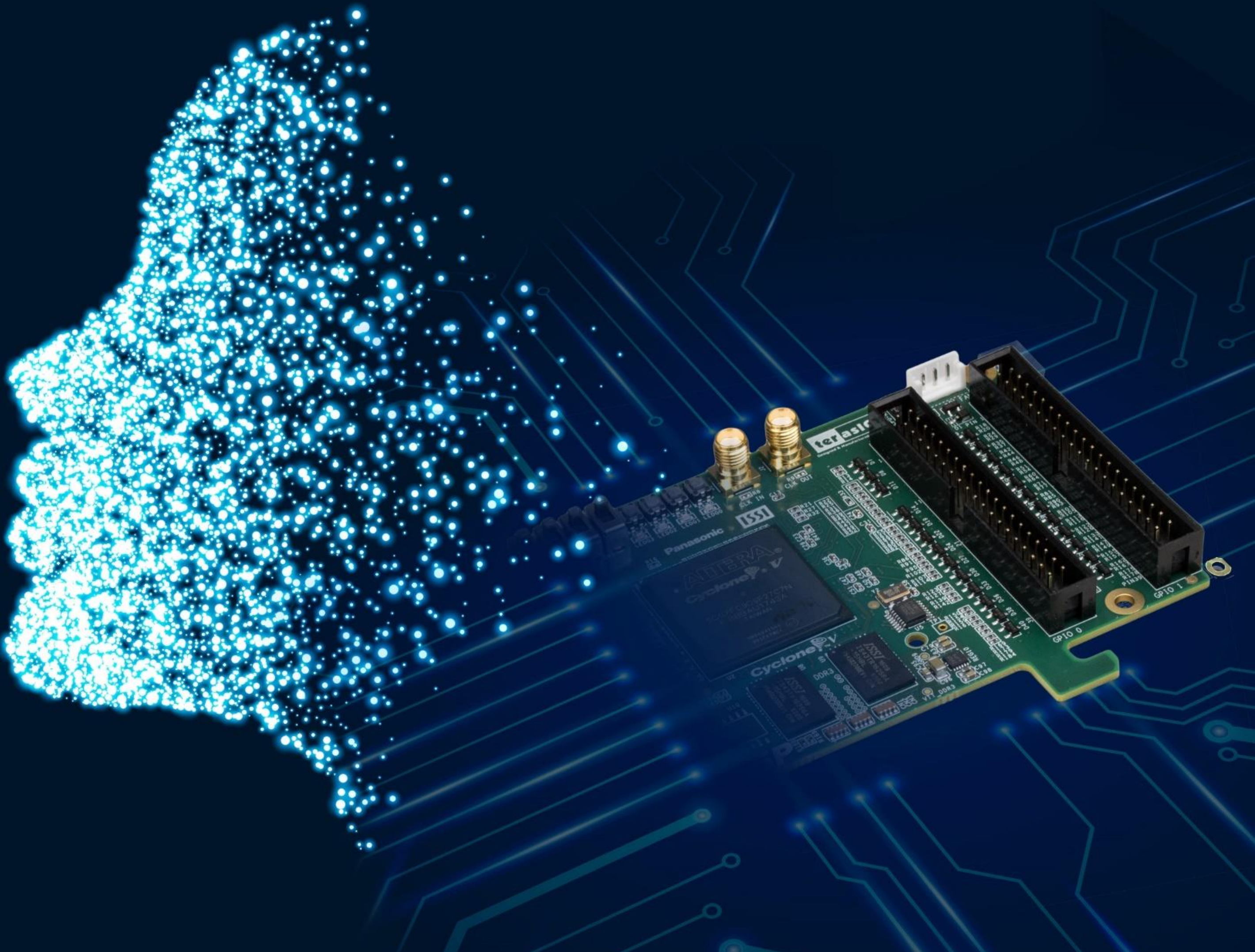
OpenVINO
Development Guide



www.terasic.com

TABLE OF CONTENTS

Chapter 1

OpenVINO Toolkit & Process Introduction

1.1 About the Guide

The OpenVINO (Open Visual Inference and Neural Network Optimization) development guide covers the workflow, Lab guide of OpenVINO Toolkit. The guide shows users how to deploy applications and solutions that emulate human vision with Intel OpenVINO Toolkit. It implements the CNN-based deep learning by using heterogeneous execution acceleration. With the easy-to-use functions library for computer vision, the OpenVINO Toolkit speeds up time-to-market for the products. This guide also shows users how to quickly setup the CNN-based deep learning applications running on FPGA.

This guide is created based on Terasic OpenVINO Starter Kit , user also can refer to this guide for DE5a-Net-DDR4 and DE5a-Net boards OpenVINO development, it includes the following contents:

- OpenVINO Toolkit Introduction
- OpenVINO Workflow
- Model Optimizer and Inference Engine
- Run demo in OpenVINO Starter Kit
- OpenVINO Starter Kit Lab

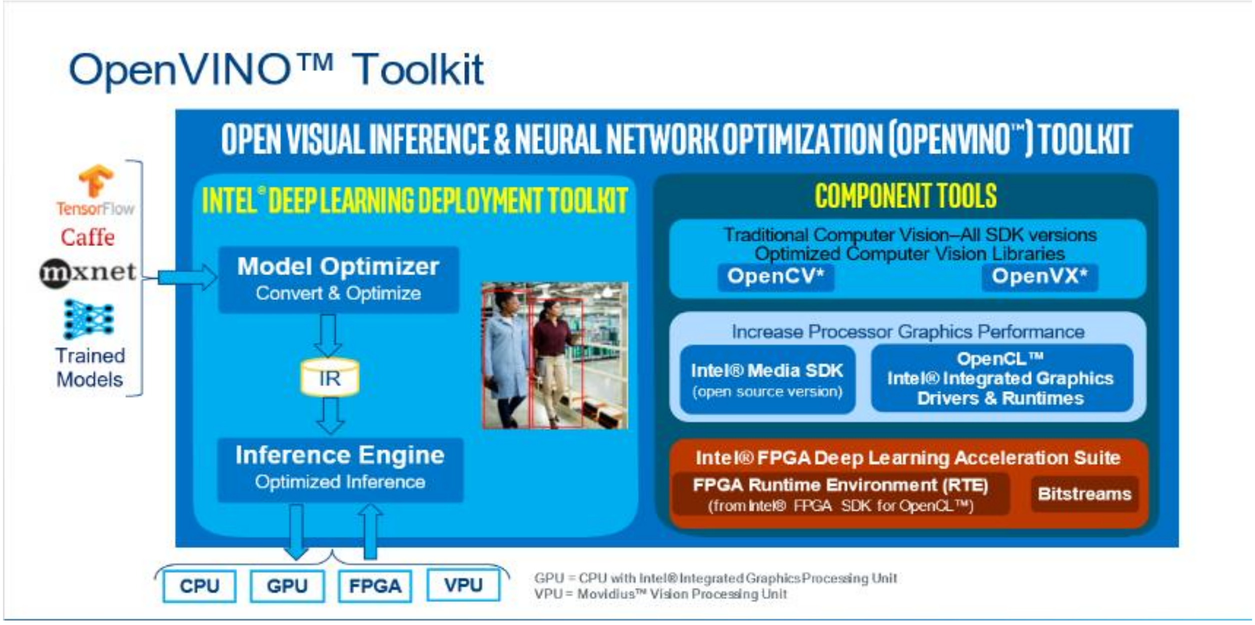
1.2 OpenVINO Toolkit Features

OpenVINO (Open Visual Inference and Neural Network Optimization) Toolkit can improve the performance of the Computer Vision, and shorten the time taken for product to market. It can help user to take the advantages of Terasic FPGA boards easily, including improving performance, reducing power consumption and significantly improving FPGA utilization. Users can achieve double efficiency with half effort and open new design possibilities. The main features are:

- Enable CNN-based deep learning inference on the edge
- Support heterogeneous execution across Intel's CV accelerators, using a common API for the CPU, Intel® Integrated Graphics, Intel® Movidius™ Neural Compute Stick, and FPGA
- Speed up time-to-market through an easy-to-use library of CV functions and pre-optimized kernels
- Include optimized calls for CV standards, including OpenCV*, OpenCL™, and OpenVX*

1.3 What's Inside OpenVINO Toolkit

OpenVINO Toolkit uses a common API, which is based on the general development standards such as OpenCL, OpenCV and OpenVX.


OpenVINO™ Toolkit
OPEN VISUAL INFERENCE & NEURAL NETWORK OPTIMIZATION (OPENVINO™) TOOLKIT
TensorFlow
Caffe
mxnet
Trained Models
INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT
Model Optimizer
Convert & Optimize
IR
Inference Engine
Optimized Inference
COMPONENT TOOLS
Traditional Computer Vision–All SDK versions
Optimized Computer Vision Libraries
OpenCV*
OpenVX*
Increase Processor Graphics Performance
Intel® Media SDK
(open source version)
OpenCL™
Intel® Integrated Graphics
Drivers & Runtimes
Intel® FPGA Deep Learning Acceleration Suite
FPGA Runtime Environment (RTE)
(from Intel® FPGA SDK for OpenCL™)
Bitstreams
CPU
GPU
FPGA
VPU
GPU = CPU with Intel® Integrated Graphics Processing Unit
VPU = Movidius™ Vision Processing Unit

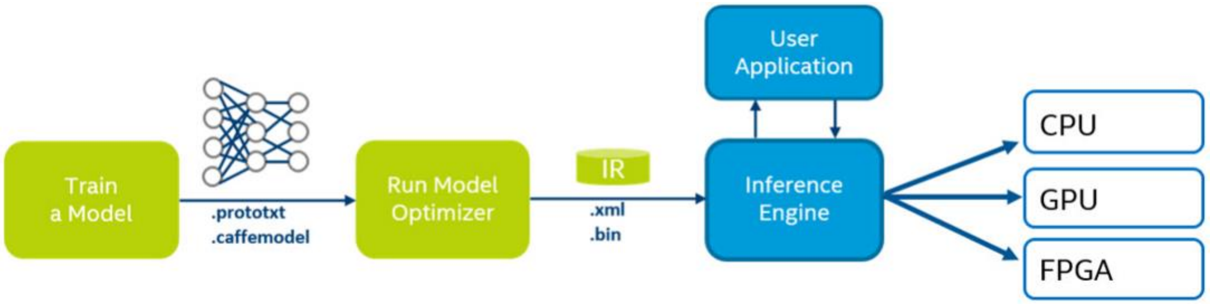
The Toolkit includes:

1. Deep Learning Deployment Toolkit, which comprises the following two components:
   - Model Optimizer: This Python*-based command line tool imports trained models from popular deep learning frameworks such as Caffe, TensorFlow, and Apache MXNet*. Input trained model, optimize topology, and convert it to a IR (IR, Intermediate Representation) file.
   - Inference Engine: This execution engine uses a common API to deliver inference solutions on the platform of your choice: CPU, GPU, VPU, or FPGA to work on heterogeneous processing and asynchronous execution to save the development time.
2. Optimized computer vision library for OpenCV, OpenVX, and image vision for CPU and GPU.
3. The improved performance of Intel processor graphics card components in Linux, including Intel Media SDK open source version, OpenCL graphics driver, and runtime environment.
4. The runtime environment (RTE) supports running OpenCL on FPGA and bitstreams for configuring FPGA.

1.4 OpenVINO Workflow

The steps for OpenVINO optimizing and deploying a trained model are:

1. Configure the Model Optimizer for your framework.
2. Convert a trained model to produce an optimized Intermediate Representation (IR) of the model based on the trained network topology, weights, and biases values.
3. Test the model in the Intermediate Representation format using the Inference Engine in the target environment by the Validation application or the sample applications.
4. Integrate the Inference Engine in your application to deploy the model in the target environment.

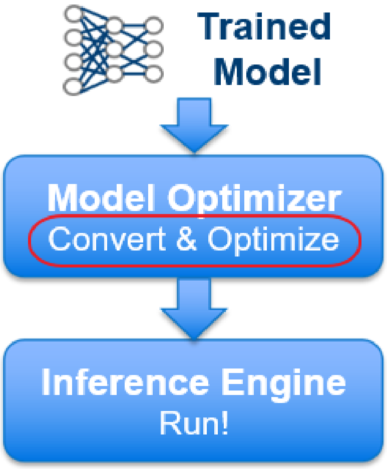


1.5 Model Optimizer

Model Optimizer is a cross-platform command-line tool that facilitates the transition between the training and deployment environment, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices.

Model Optimizer produces an OpenVINO supported framework as a trained model input and an Intermediate Representation (IR) of the network as output. Intermediate Representation is a pair of files that describe the whole model:

- .xml: Describes the network topology
- .bin: Contains the weights and biases binary data



- How the Model Optimizer Works

Model Optimizer loads a model into memory, followed by reading it and building the internal representation of the model. The Model Optimizer then optimizes it and produces the Intermediate Representation. The Intermediate Representation is the only format the Inference Engine accepts. Model Optimizer has two main purposes:

1) Produce a valid Intermediate Representation.

If this main conversion artifact is not valid, the Inference Engine cannot run. The primary responsibility of the Model Optimizer is to produce the two files that form the Intermediate Representation.

2) Produce an optimized Intermediate Representation.

Pre-trained models contain layers that are important for training such as the dropout layer. These layers are useless during inference and might increase the inference time.

In many cases, these layers can be automatically removed from the resulting Intermediate Representation. However, if a group of layers can be represented as one mathematical operation and thus a single layer, the Model Optimizer recognizes such patterns and replaces these layers with one layer. The result is an Intermediate Representation that has fewer layers than the original model. This reduces the inference time.

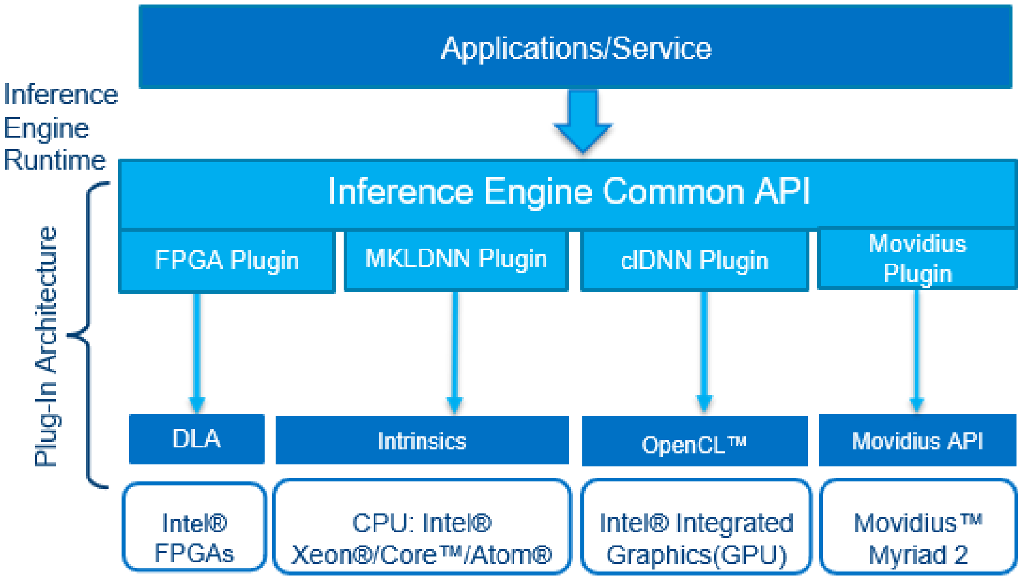
Many common layers exist across known frameworks and neural network topologies. Examples of these layers are Convolution, Pooling, and Activation. The Model Optimizer must be able to work with these layers to read the original model and produce the Intermediate Representation of a model, the layer list varies by framework. Please refer to the documentation of the Caffe*, TensorFlow* and MXNet* for the topologies supported by each of these frameworks. If your topology contains only layers from the list of layers, which is the case for the topologies used by most users, the Model Optimizer can easily create the Intermediate Representation. Users can proceed to work with the Inference Engine afterwards.

However, if you use a topology with layers that are not recognized by the Model Optimizer, please refer to Custom Layers in the Model Optimizer to learn how to work with custom layers.

1.6 Inference Engine

After an Intermediate Representation is created by the Model Optimizer, input data can be inferred by the Inference Engine.

The Inference Engine is a C++ library with a set of C++ classes to infer input data (images) and get a result. The C++ library provides an API to read the Intermediate Representation, set the input and output formats, and execute the model on devices.



Each supported target device has a plugin and each plugin is a DLL/shared library. The Heterogeneous plugin lets you distribute a calculation workload across devices. One needs to make sure those libraries are specified in the path of host PC or in the place pointed to the plugin loader. Additionally, the related library for each plugin must be included in the LD_LIBRARY_PATH. When the Inference Engine calls the FPGA-based DLA plug-in, the DLA runtime software layer is called to use the DLA API. These

APIs are converted to the corresponding modules executed on the FPGA device. This part would be executed in different levels of in-depth learning networks.

Common Workflow for Using the Inference Engine API:

1. Read the Intermediate Representation:
   Use the InferenceEngine::CNNNetReaderclass and read an Intermediate Representation file into a CNNNetwork class. This class represents the network in host memory.

2. Prepare inputs and outputs formats:
   After loading the network, specify input and output precision, and the layout on the network using CNNNetwork::getInputInfo() and CNNNetwork::getOutputInfo().

3. Select Plugin:
   Select the plugin to load your network. Create the plugin with the InferenceEngine:: PluginDispatcher load helper class. Pass per device loading configurations specific to this device and register extensions to this device.

4. Compile and Load:
   Use the plugin interface wrapper class InferenceEngine::InferencePlugin to call the LoadNetwork () API to compile and load the network on the device. Pass in the per-target load configuration for this compilation and load operation.

5. Set input data:
   There's an ExecutableNetwork object with the network loaded.

   Use this object to create an InferRequest in which you signal the input buffers to use for input and output. Specify a device-allocated memory and copy it into the device memory directly or tell the device to use your application memory to save a copy.

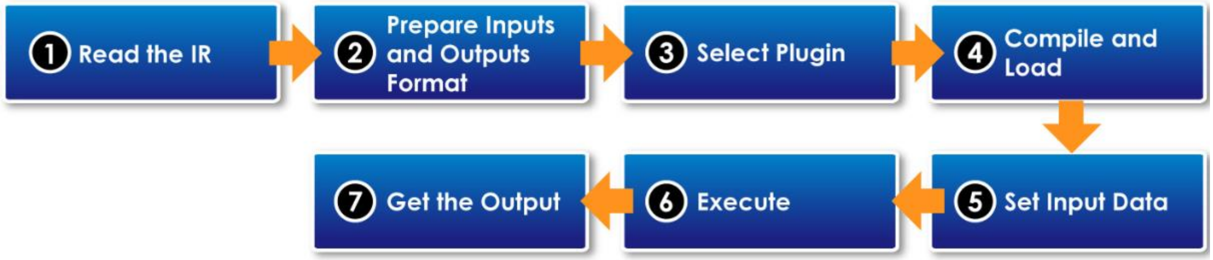
6. Execute:
   Choose the execution mode with the input and outpu memory defined:

   Synchronously - Infer() method. Block until inference finishes.

   Asynchronously - StartAsync() method. Check status with the wait() method (0 timeout), wait, or specify a completion callback.

7. Get the output:
   Get the output memory or read the memory provided earlier after inference is complete.

   This can be done with the InferRequest GetBlob API.



For more information about integrating the Inference Engine in your application, please refer to the Inference Engine Developer Guide.

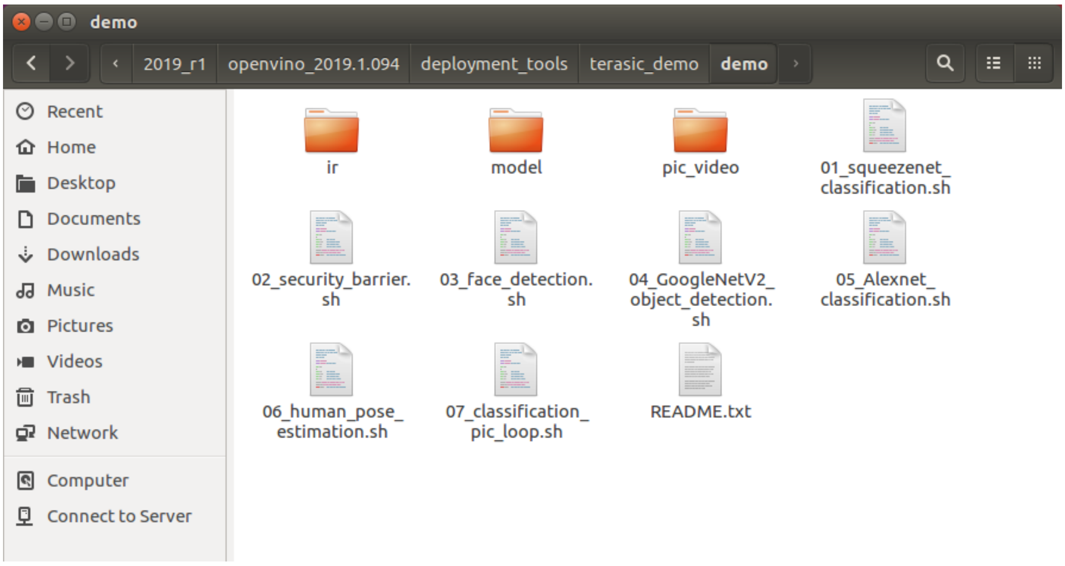
Chapter 2

Run the DEMO on the OpenVINO Starter Kit

This chapter describes how to run the demo on the OpenVINO Starter Kit and shows the execution result of the demo. Before running these demos, user needs to finish OpenVINO Development installation by referring to the manual “**OpenVINO_Installation_Guide**”.

2.1 Introduction

As shown in the figure below, there are some shell scripts in the terasic_demo/demo folder.



Below is the brief introduction of the demo folder.

1. How to use these Shell script files

   Users can run any one of the shell script files with default parameters, and users can also use cpu , vpu or fpga to specify the target device to run the demo. There are also other parameters for using, users can run shell script file with ‘-h’ for more details. And the default parameter is for cpu.

2. The images and video required by the demo are in the pic_video folder.

3. The Caffe model downloaded from internet is in the model folder.

   - alexnet
   - squeezenet1.1
   - GoogleNetV2
   - Users can add Caffe model by referring to the writing rule of the script. Please pay attention to the path and name.
   - Please refer to OpenVINO-Using-TensorFlow to transfer the Tensorflow model

4. IR folder

   While running the demo, the corresponding model IR file will be generated automatically if it’s needed.

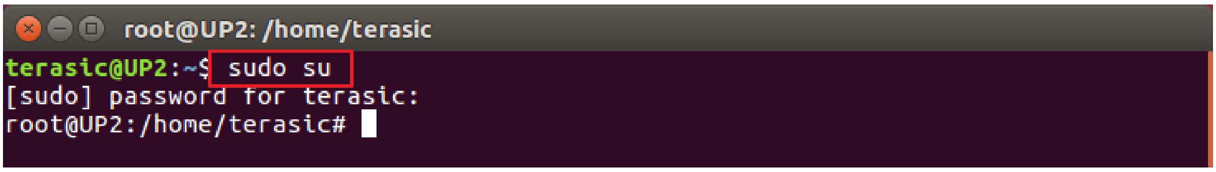
- The model generated under FP16 folder is used for FPGA
- The model generated under FP32 folder is used for CPU

2.2 Execute the Demo

There are seven demos included in the OpenVINO Toolkit. To run these demos, users need to open a terminal and type sudo su, source the setup script, and prepare the running environment.

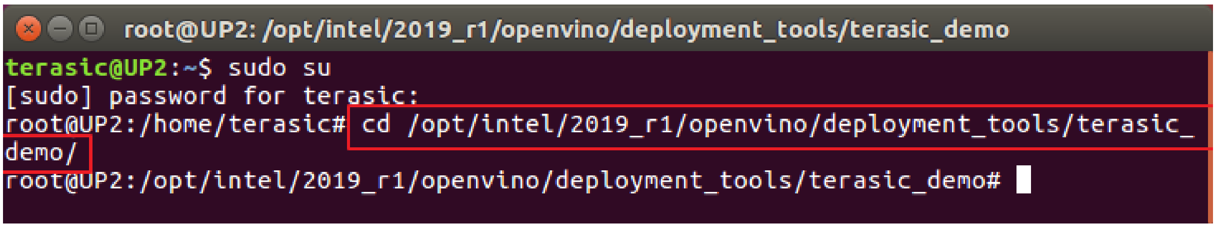
(login account: here, our username is terasic, password is terasic)

1. Click on the desktop to open the Terminal, enter command “*sudo su*” to change user to root Super User, enter password “*terasic*”



```
root@UP2: /home/terasic
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# 
```

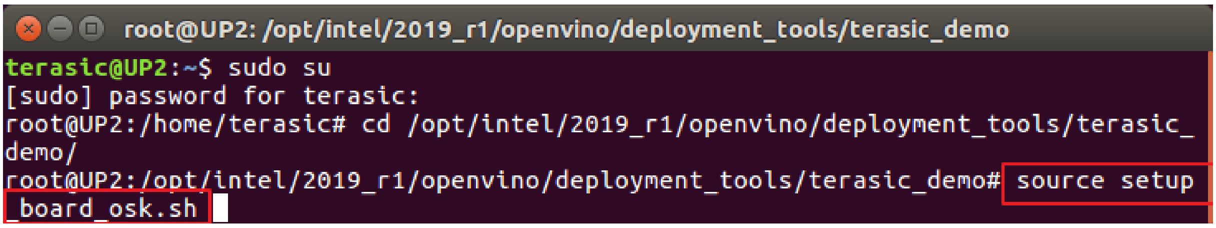
2. Switch to terasic_demo path



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# 
```

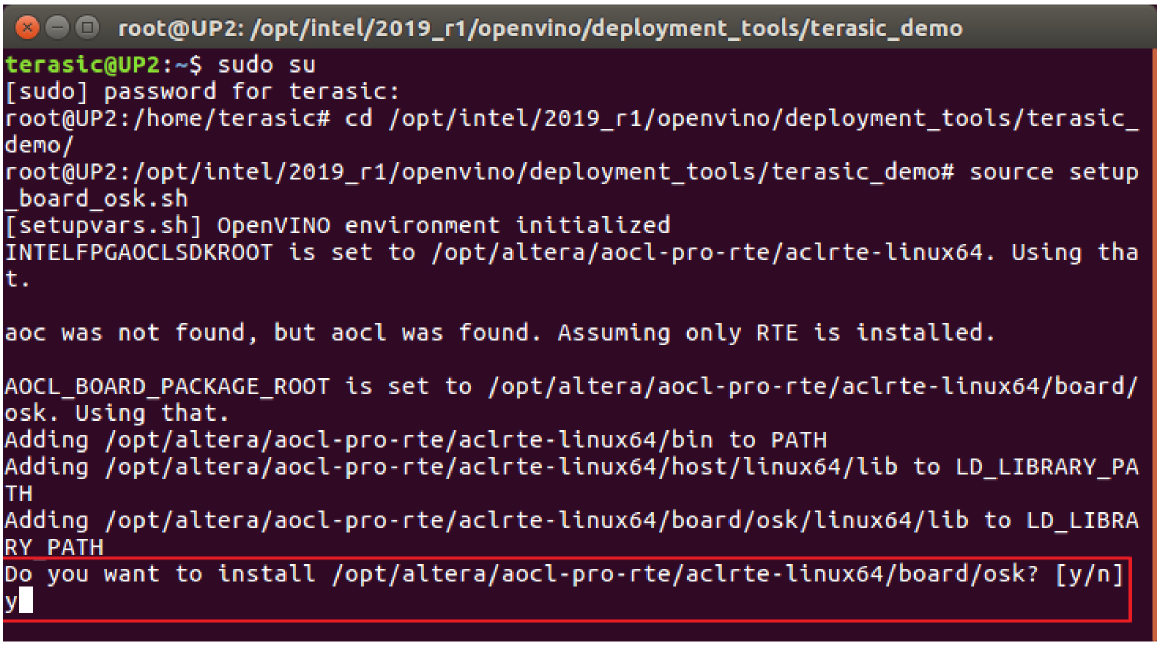
3. Source setup_board_osk.sh

   Note: User needs to setup the corresponding .sh file for the FPGA board, for example, source setup_board_de5a_net_ddr4.sh for DE5a-Net-DDR4 board.



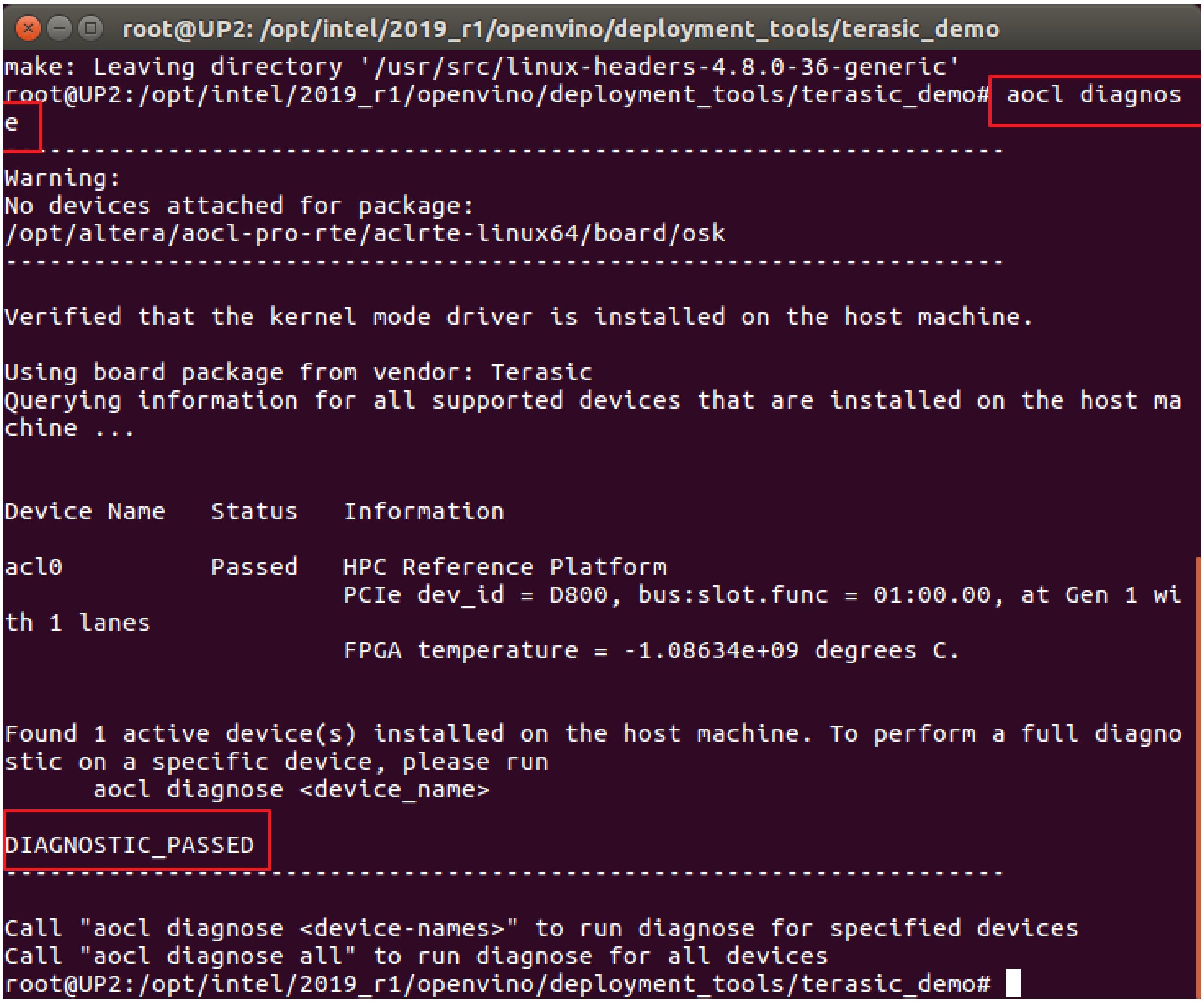
```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# source setup
_board_osk.sh 
```

4. Type “y” to install.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

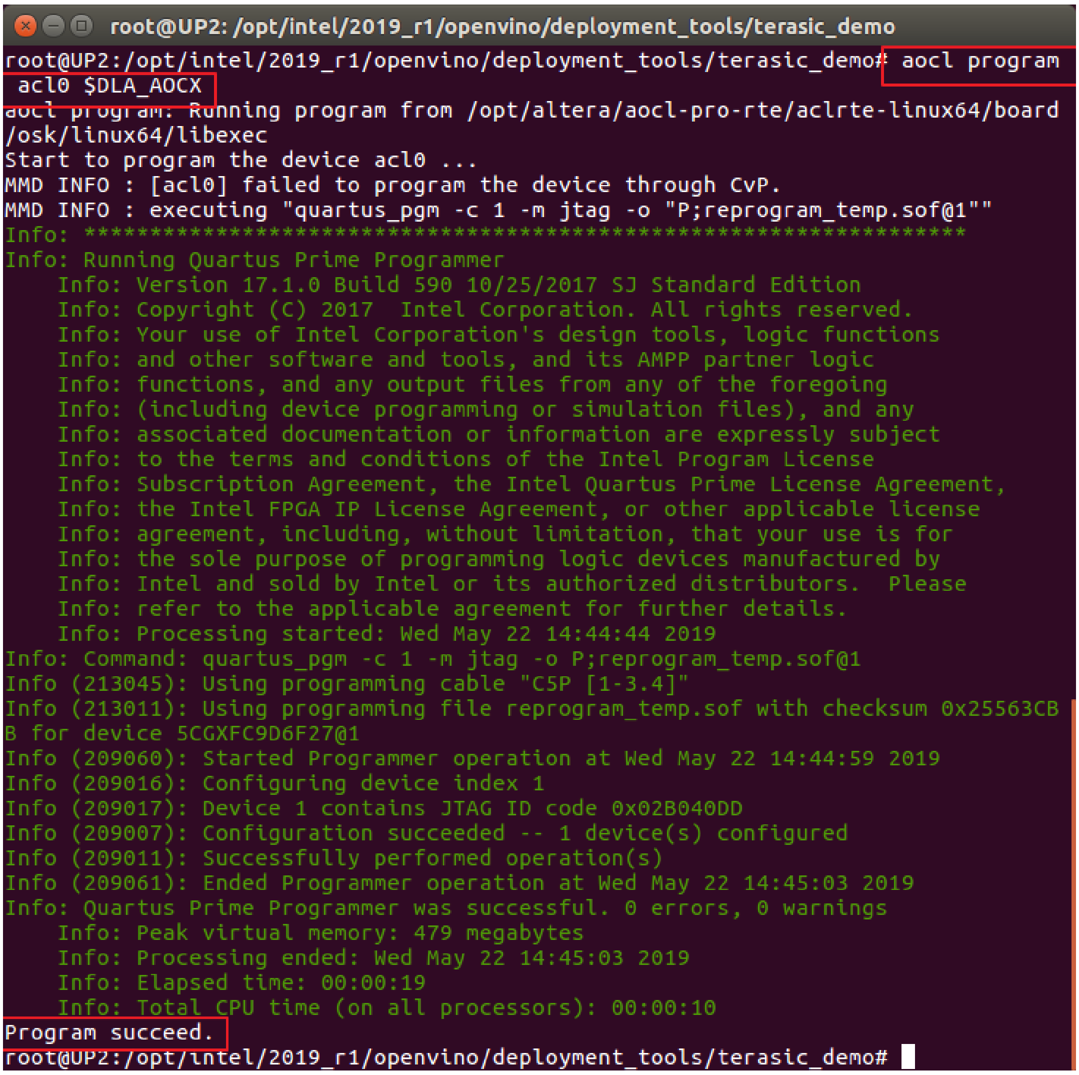
```
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# source setup
_board_osk.sh
[setupvars.sh] OpenVINO environment initialized
INTELFPGAOCLSDKROOT is set to /opt/altera/aocl-pro-rte/aclrte-linux64. Using tha
t.

aoc was not found, but aocl was found. Assuming only RTE is installed.

AOCL_BOARD_PACKAGE_ROOT is set to /opt/altera/aocl-pro-rte/aclrte-linux64/board/
osk. Using that.
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/bin to PATH
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/host/linux64/lib to LD_LIBRARY_PA
TH
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/board/osk/linux64/lib to LD_LIBRA
RY_PATH
Do you want to install /opt/altera/aocl-pro-rte/aclrte-linux64/board/osk? [y/n]
y
```

5. Enter *aocl diagnose* to check the environment, the “DIAGNOSTIC_PASSED” represents the environment setup is successful.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

```
make: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# aocl diagnos
e
--------------------------------------------------------------------
Warning:
No devices attached for package:
/opt/altera/aocl-pro-rte/aclrte-linux64/board/osk
--------------------------------------------------------------------

Verified that the kernel mode driver is installed on the host machine.

Using board package from vendor: Terasic
Querying information for all supported devices that are installed on the host ma
chine ...


Device Name   Status   Information

acl0          Passed   HPC Reference Platform
                       PCIe dev_id = D800, bus:slot.func = 01:00.00, at Gen 1 wi
th 1 lanes
                       FPGA temperature = -1.08634e+09 degrees C.


Found 1 active device(s) installed on the host machine. To perform a full diagno
stic on a specific device, please run
      aocl diagnose <device_name>

DIAGNOSTIC_PASSED
--------------------------------------------------------------------

Call "aocl diagnose <device-names>" to run diagnose for specified devices
Call "aocl diagnose all" to run diagnose for all devices
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# 
```

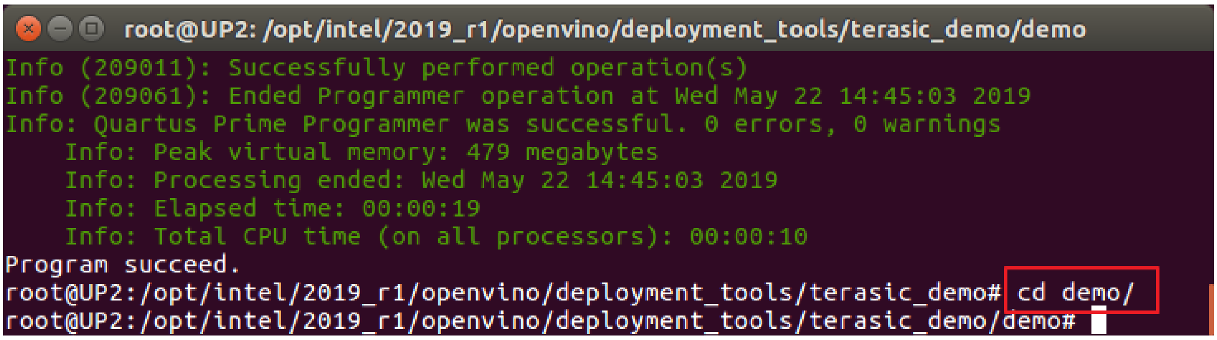
6. Use an USB cable to connect the OSK USB Blaster II connector and PC. Enter “*aocl program acl0 $DLA_AOCX*” to program an .aocx file (click here to know how to program an .aocx file) to the FPGA.



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

```
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# aocl program
 acl0 $DLA_AOCX
aocl program: Running program from /opt/altera/aocl-pro-rte/aclrte-linux64/board
/osk/linux64/libexec
Start to program the device acl0 ...
MMD INFO : [acl0] failed to program the device through CvP.
MMD INFO : executing "quartus_pgm -c 1 -m jtag -o "P;reprogram_temp.sof@1""
Info: *******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 17.1.0 Build 590 10/25/2017 SJ Standard Edition
    Info: Copyright (C) 2017  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Wed May 22 14:44:44 2019
Info: Command: quartus_pgm -c 1 -m jtag -o P;reprogram_temp.sof@1
Info (213045): Using programming cable "C5P [1-3.4]"
Info (213011): Using programming file reprogram_temp.sof with checksum 0x25563CB
B for device 5CGXFC9D6F27@1
Info (209060): Started Programmer operation at Wed May 22 14:44:59 2019
Info (209016): Configuring device index 1
Info (209017): Device 1 contains JTAG ID code 0x02B040DD
Info (209007): Configuration succeeded -- 1 device(s) configured
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Wed May 22 14:45:03 2019
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 479 megabytes
    Info: Processing ended: Wed May 22 14:45:03 2019
    Info: Elapsed time: 00:00:19
    Info: Total CPU time (on all processors): 00:00:10
Program succeed.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# 
```

7. Switch to the demo path



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

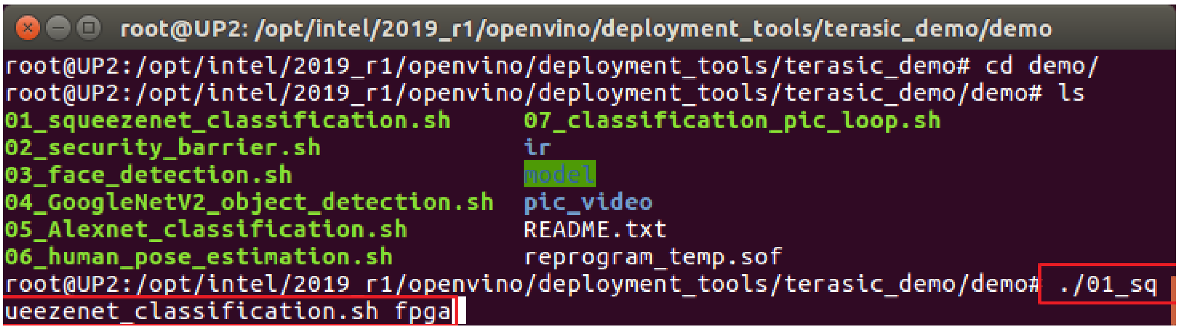
```
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Wed May 22 14:45:03 2019
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 479 megabytes
    Info: Processing ended: Wed May 22 14:45:03 2019
    Info: Elapsed time: 00:00:19
    Info: Total CPU time (on all processors): 00:00:10
Program succeed.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# cd demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# 
```

8. Execute the DEMOs:

- **01_squeezenet demo**

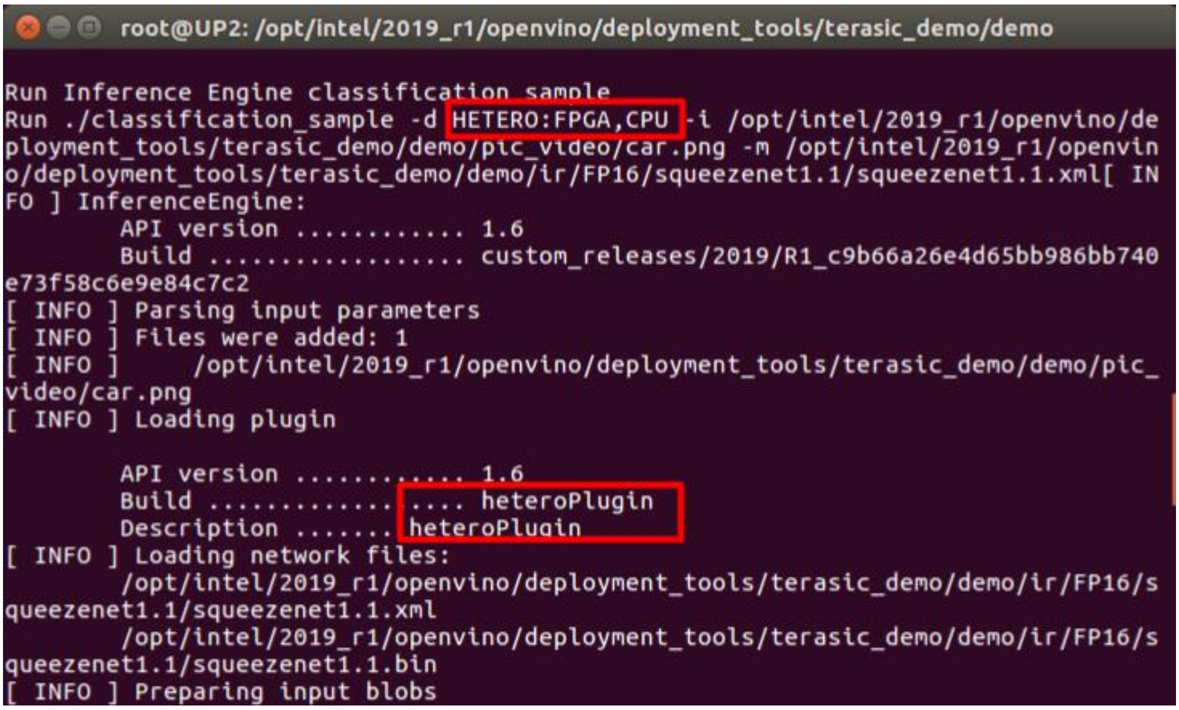
This demo can recognize the objects in the figure by using the squeezenet model

1) *./01_squeezenet_classification.sh fpga* (run demo with FPGA)


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

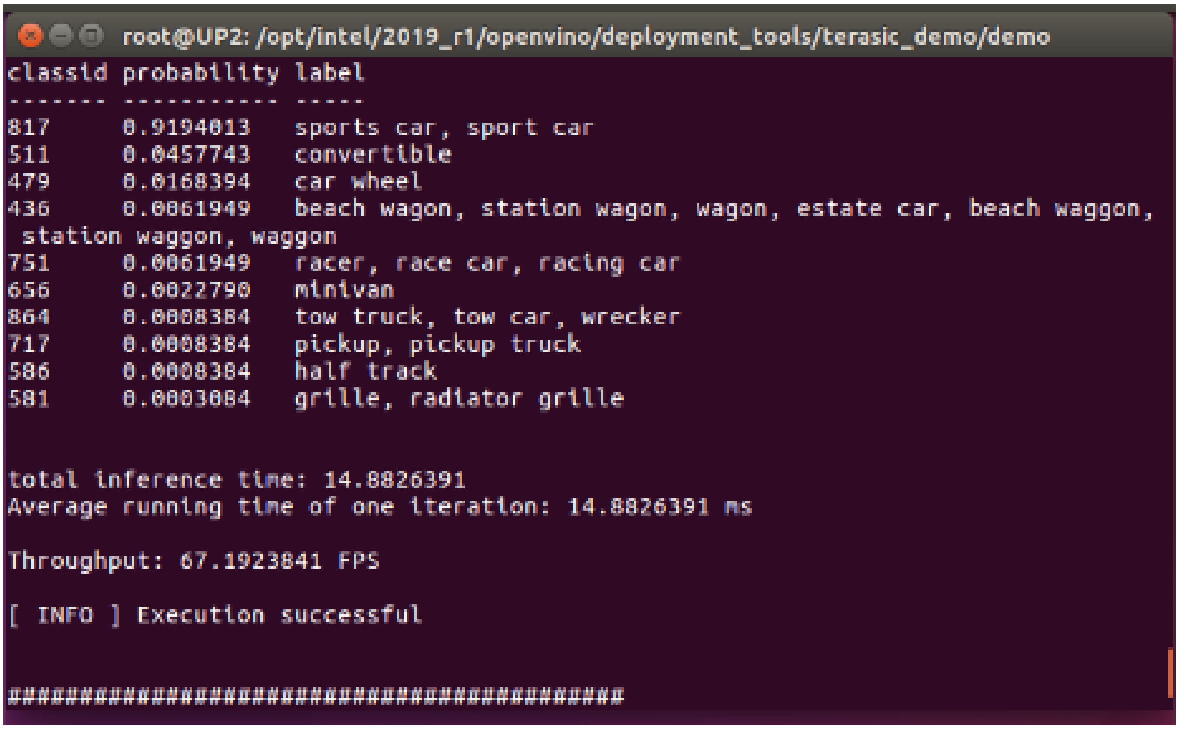
```
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# cd demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ls
01_squeezenet_classification.sh      07_classification_pic_loop.sh
02_security_barrier.sh               ir
03_face_detection.sh                 model
04_GoogleNetV2_object_detection.sh   pic_video
05_Alexnet_classification.sh         README.txt
06_human_pose_estimation.sh          reprogram_temp.sof
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./01_sq
ueezenet_classification.sh fpga
```

2) Users can see "HETERO:FPGA, CPU", which prompts the DEMO is running on FPGA and CPU.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```

Run Inference Engine classification sample
Run ./classification_sample -d HETERO:FPGA,CPU -i /opt/intel/2019_r1/openvino/de
ployment_tools/terasic_demo/demo/pic_video/car.png -m /opt/intel/2019_r1/openvin
o/deployment_tools/terasic_demo/demo/ir/FP16/squeezenet1.1/squeezenet1.1.xml[ IN
FO ] InferenceEngine:
        API version ............ 1.6
        Build .................. custom_releases/2019/R1_c9b66a26e4d65bb986bb740
e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
[ INFO ] Files were added: 1
[ INFO ]     /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/pic_
video/car.png
[ INFO ] Loading plugin

        API version ............ 1.6
        Build .................. heteroPlugin
        Description ....... heteroPlugin
[ INFO ] Loading network files:
        /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/ir/FP16/s
queezenet1.1/squeezenet1.1.xml
        /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/ir/FP16/s
queezenet1.1/squeezenet1.1.bin
[ INFO ] Preparing input blobs
```

3) It prints out the top 10 results.

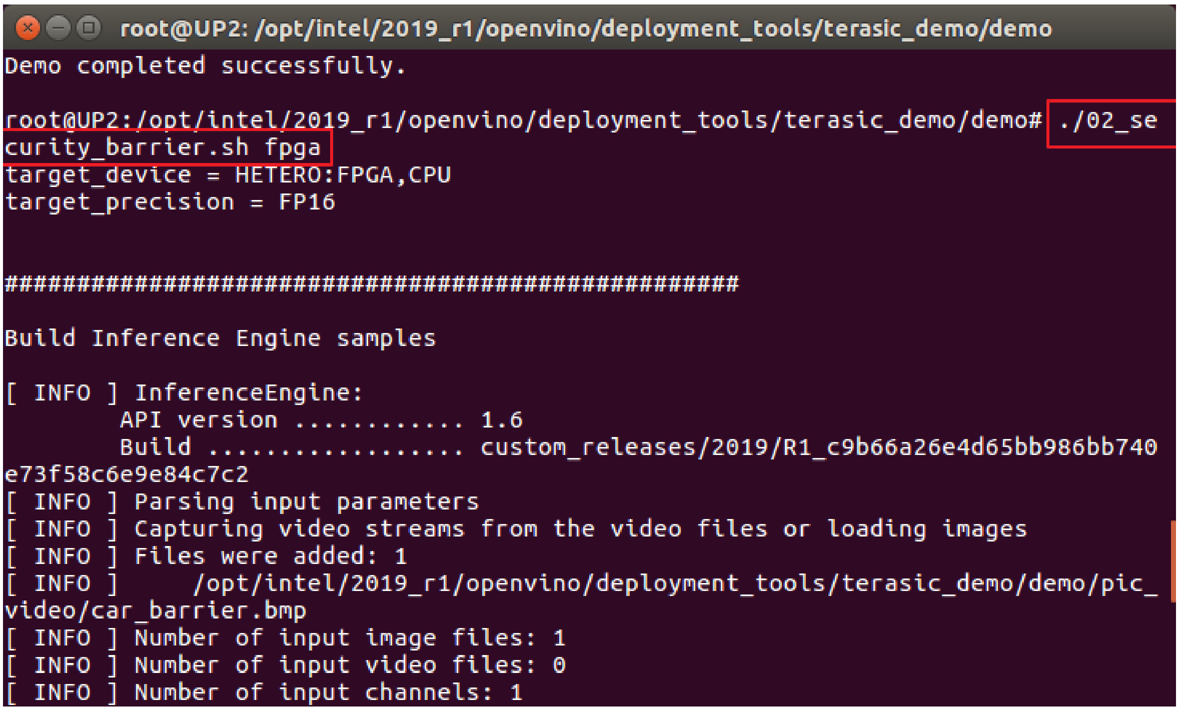

root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
classid probability label
------- ----------- -----
817     0.9194013   sports car, sport car
511     0.0457743   convertible
479     0.0168394   car wheel
436     0.0061949   beach wagon, station wagon, wagon, estate car, beach waggon,
 station waggon, waggon
751     0.0061949   racer, race car, racing car
656     0.0022790   minivan
864     0.0008384   tow truck, tow car, wrecker
717     0.0008384   pickup, pickup truck
586     0.0008384   half track
581     0.0003084   grille, radiator grille


total inference time: 14.8826391
Average running time of one iteration: 14.8826391 ms

Throughput: 67.1923841 FPS

[ INFO ] Execution successful


###########################################
```

02_security_barrier demo

This demo can recognize the car, car license number, and its location by using the three models.

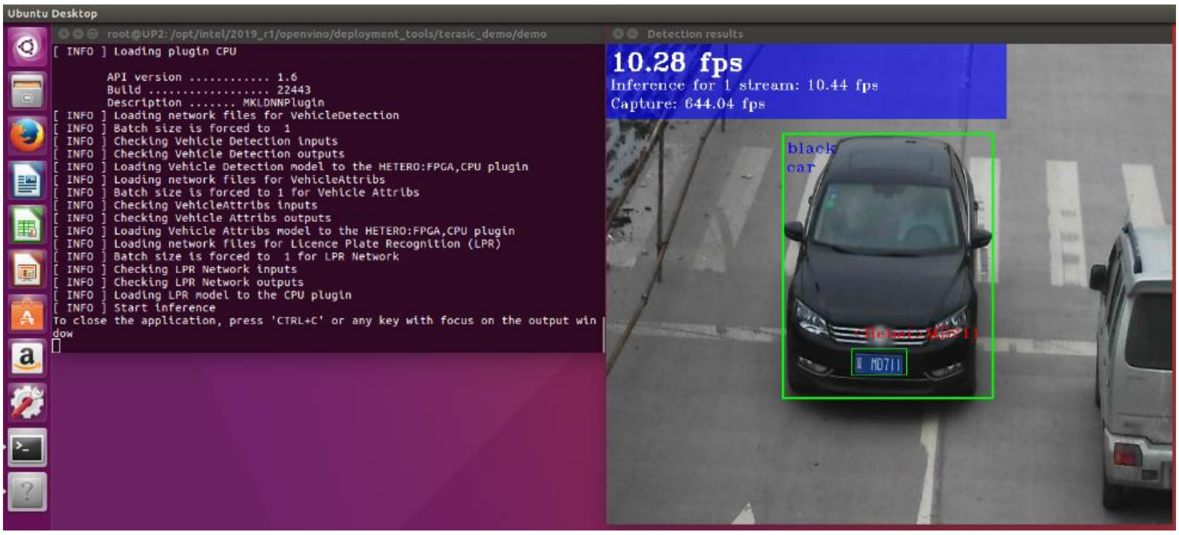
1) *./02_security_barrier.sh fpga* (run the demo with FPGA)



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
Demo completed successfully.

root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./02_se
curity_barrier.sh fpga
target_device = HETERO:FPGA,CPU
target_precision = FP16


###################################################

Build Inference Engine samples

[ INFO ] InferenceEngine:
        API version ............ 1.6
        Build .................. custom_releases/2019/R1_c9b66a26e4d65bb986bb740
e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
[ INFO ] Capturing video streams from the video files or loading images
[ INFO ] Files were added: 1
[ INFO ]     /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/pic_
video/car_barrier.bmp
[ INFO ] Number of input image files: 1
[ INFO ] Number of input video files: 0
[ INFO ] Number of input channels: 1
```

2) The result is shown in the figure below. Enter Ctrl+C to close the Application.



Ubuntu Desktop
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
Detection results
10.28 fps
Inference for 1 stream: 10.44 fps
Capture: 644.04 fps
black
car
M0711

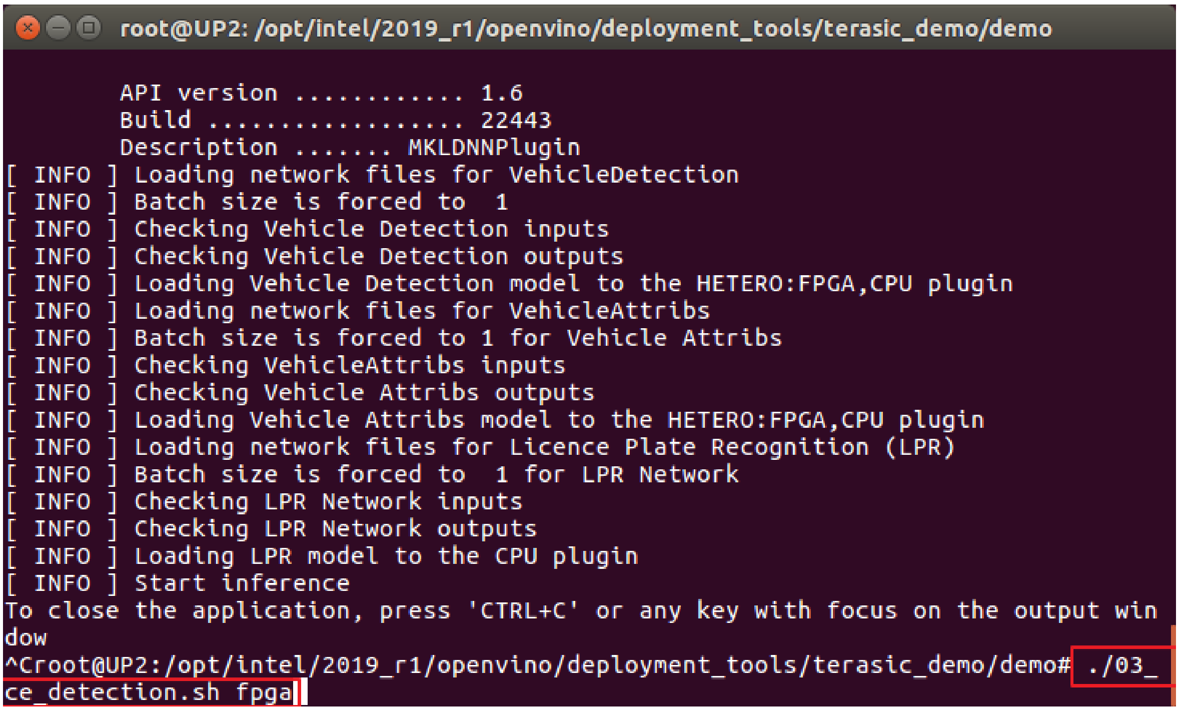
```
[ INFO ] Loading plugin CPU

        API version ............ 1.6
        Build .................. 22443
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files for VehicleDetection
[ INFO ] Batch size is forced to  1
[ INFO ] Checking Vehicle Detection inputs
[ INFO ] Checking Vehicle Detection outputs
[ INFO ] Loading Vehicle Detection model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for VehicleAttribs
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
```

03_face_detection

This demo uses four models and it can recognize human face position in the figure. It can also judge the human gender, age, expression, and head gesture according to the human face.

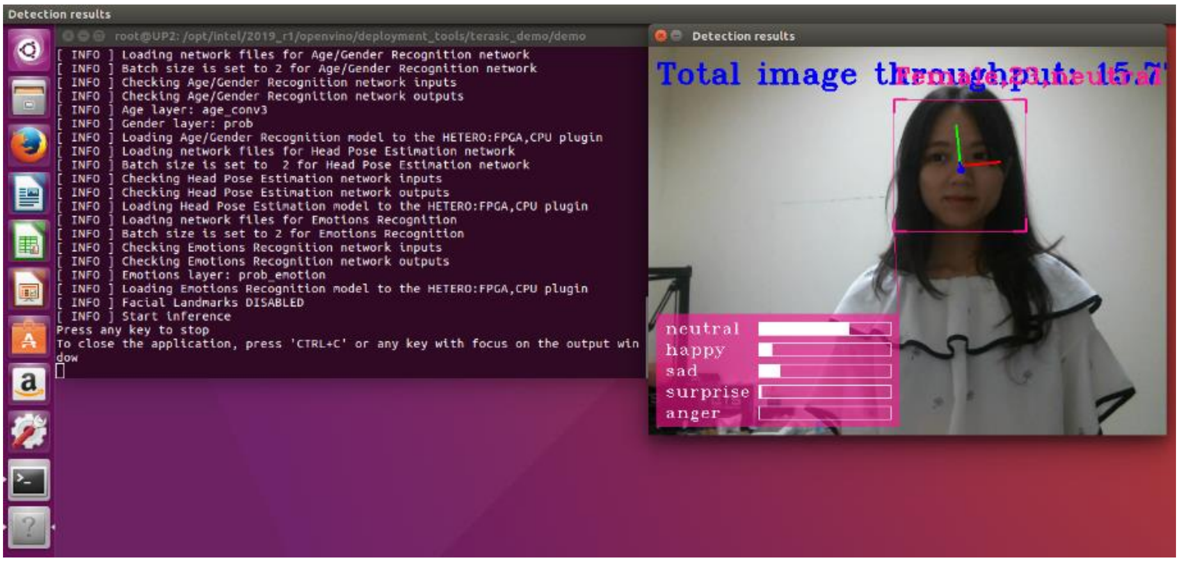
1) Plug a UVC USB camera to the host PC USB port.

2) Execute "*./03_face_detection.sh fpga*" to run the demo with FPGA.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
        API version ............ 1.6
        Build .................. 22443
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files for VehicleDetection
[ INFO ] Batch size is forced to  1
[ INFO ] Checking Vehicle Detection inputs
[ INFO ] Checking Vehicle Detection outputs
[ INFO ] Loading Vehicle Detection model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for VehicleAttribs
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
^Croot@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./03_
ce_detection.sh fpga
```

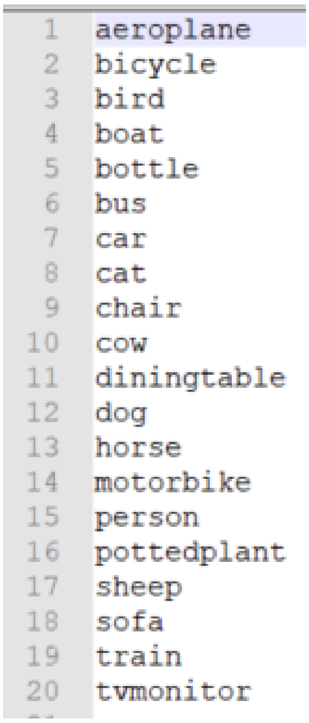
3) The result is shown in the figure below, enter Ctrl+c to close the Application.


Detection results
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
Detection results
Total image throughput: 15.7
neutral
happy
sad
surprise
anger

```
[ INFO ] Loading network files for Age/Gender Recognition network
[ INFO ] Batch size is set to 2 for Age/Gender Recognition network
[ INFO ] Checking Age/Gender Recognition network inputs
[ INFO ] Checking Age/Gender Recognition network outputs
[ INFO ] Age layer: age_conv3
[ INFO ] Gender layer: prob
[ INFO ] Loading Age/Gender Recognition model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for Head Pose Estimation network
[ INFO ] Batch size is set to  2 for Head Pose Estimation network
[ INFO ] Checking Head Pose Estimation network inputs
[ INFO ] Checking Head Pose Estimation network outputs
[ INFO ] Loading Head Pose Estimation model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for Emotions Recognition
[ INFO ] Batch size is set to 2 for Emotions Recognition
[ INFO ] Checking Emotions Recognition network inputs
[ INFO ] Checking Emotions Recognition network outputs
[ INFO ] Emotions layer: prob_emotion
[ INFO ] Loading Emotions Recognition model to the HETERO:FPGA,CPU plugin
[ INFO ] Facial Landmarks DISABLED
[ INFO ] Start inference
Press any key to stop
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
```

- **04_GoogleNetV2_object_detection**

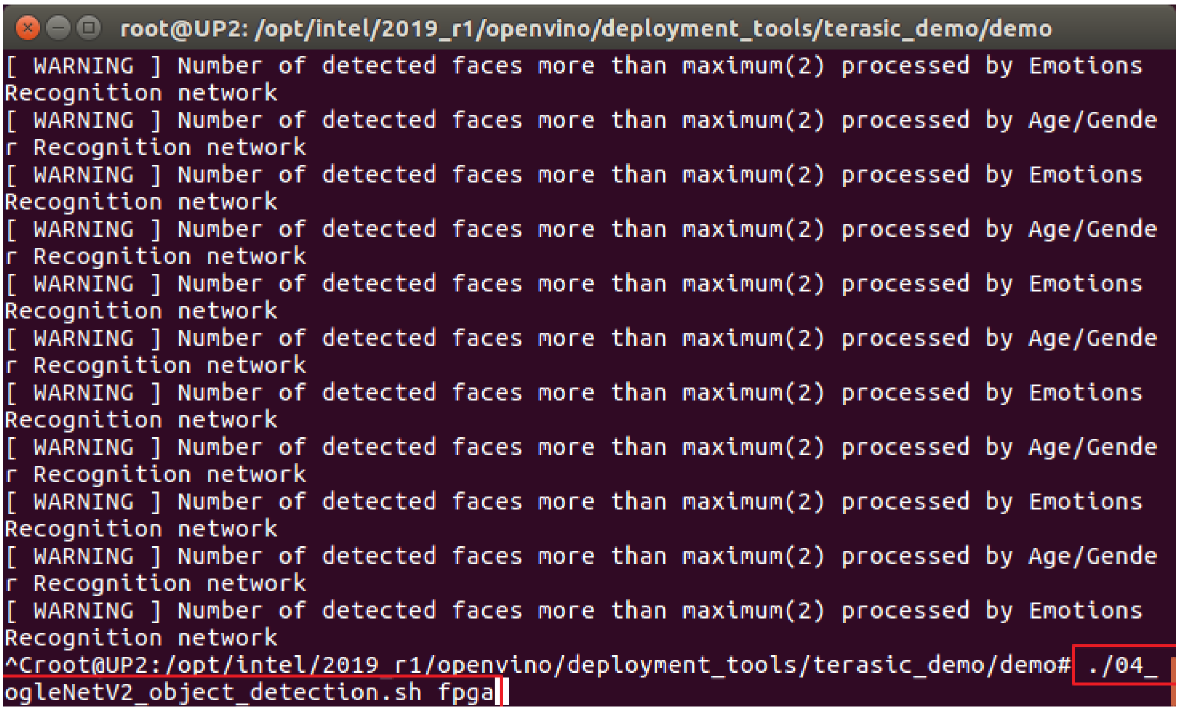
This demo can recognize the target object by using GoogleNetV2. The object tags are shown in the figure below:



```
aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor
```

1) Plug the UVC USB camera to the host PC USB port

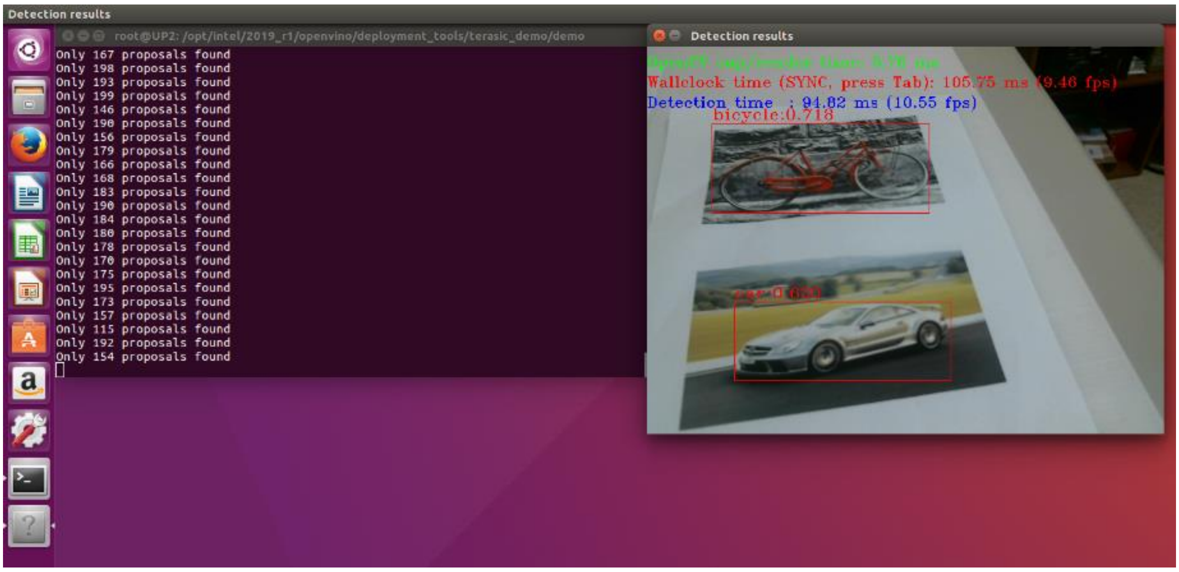
2) Execute “*./04_GoogleNetV2_object_detection.sh fpga*” to run the demo with FPGA



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Age/Gende
r Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Age/Gende
r Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Age/Gende
r Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Age/Gende
r Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Age/Gende
r Recognition network
[ WARNING ] Number of detected faces more than maximum(2) processed by Emotions
Recognition network
^Croot@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./04_
ogleNetV2_object_detection.sh fpga
```

3) Recognize the figures with the camera. The results are shown in the figure below. Enter Ctrl+c to close the Detection results window.


Detection results
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
Detection results
bicycle:0.718

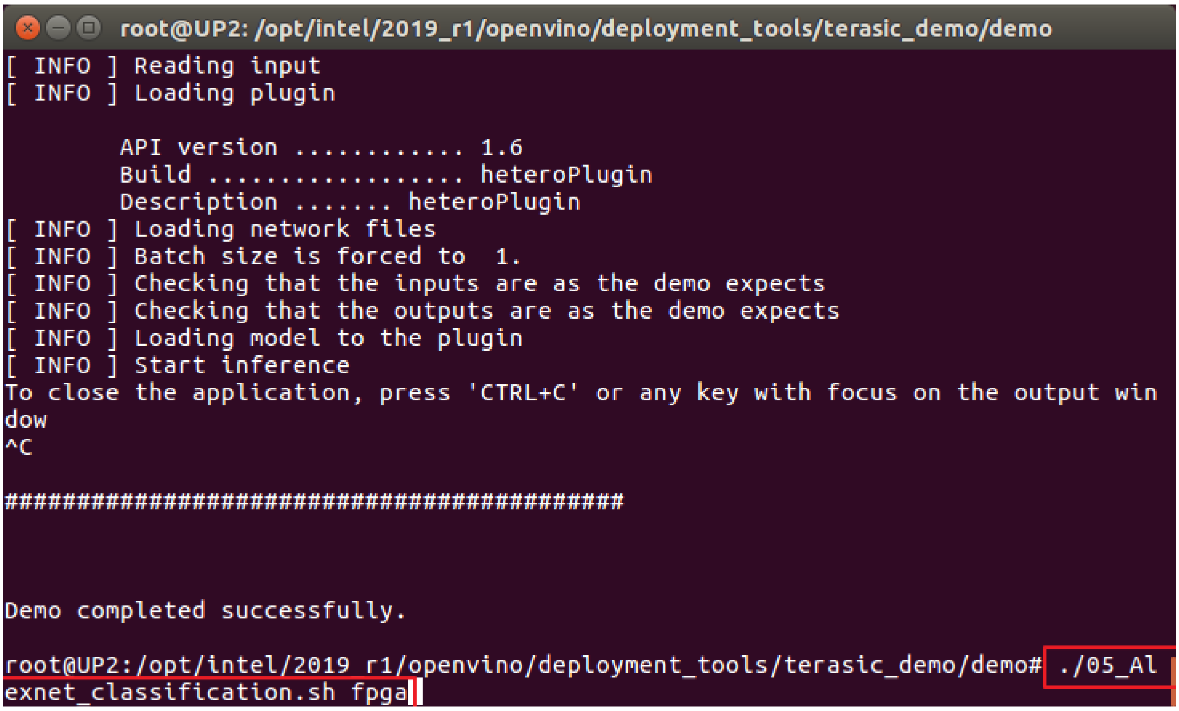
```
Only 167 proposals found
Only 198 proposals found
Only 193 proposals found
Only 199 proposals found
Only 146 proposals found
Only 190 proposals found
Only 156 proposals found
Only 179 proposals found
Only 166 proposals found
Only 168 proposals found
Only 183 proposals found
Only 190 proposals found
Only 184 proposals found
Only 180 proposals found
Only 178 proposals found
Only 170 proposals found
Only 175 proposals found
Only 195 proposals found
Only 173 proposals found
Only 157 proposals found
Only 115 proposals found
Only 192 proposals found
Only 154 proposals found
```

```
Wallclock time (SYNC, press Tab): 105.75 ms (9.46 fps)
Detection time  : 94.82 ms (10.55 fps)
```

- **05_Alexnet_classification**

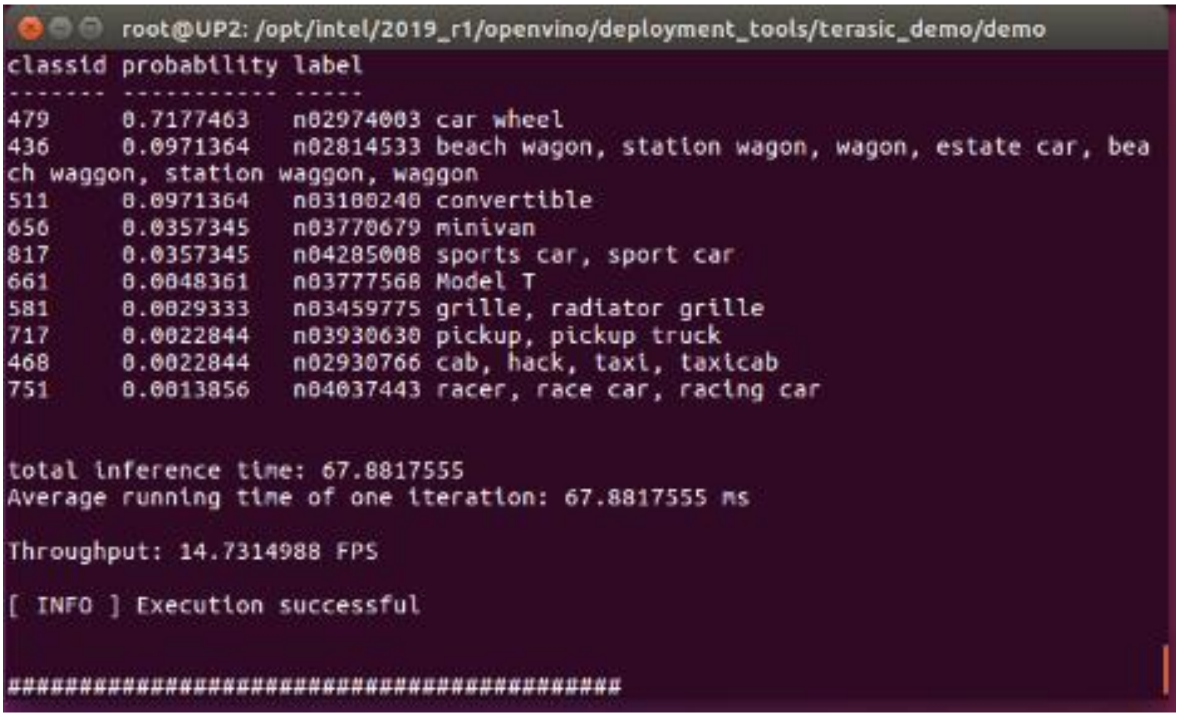
This demo can recognize the target objects by using Alexnet model and print out the top 10 information (the recognized result in top 10 probabilities).

1) Execute "*./05_Alexnet_classification.sh fpga*" to run the demo with FPGA.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
[ INFO ] Reading input
[ INFO ] Loading plugin

        API version ............ 1.6
        Build .................. heteroPlugin
        Description ....... heteroPlugin
[ INFO ] Loading network files
[ INFO ] Batch size is forced to  1.
[ INFO ] Checking that the inputs are as the demo expects
[ INFO ] Checking that the outputs are as the demo expects
[ INFO ] Loading model to the plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
^C

###########################################


Demo completed successfully.

root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./05_Al
exnet_classification.sh fpga
```

2) The results are shown in the figure below.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

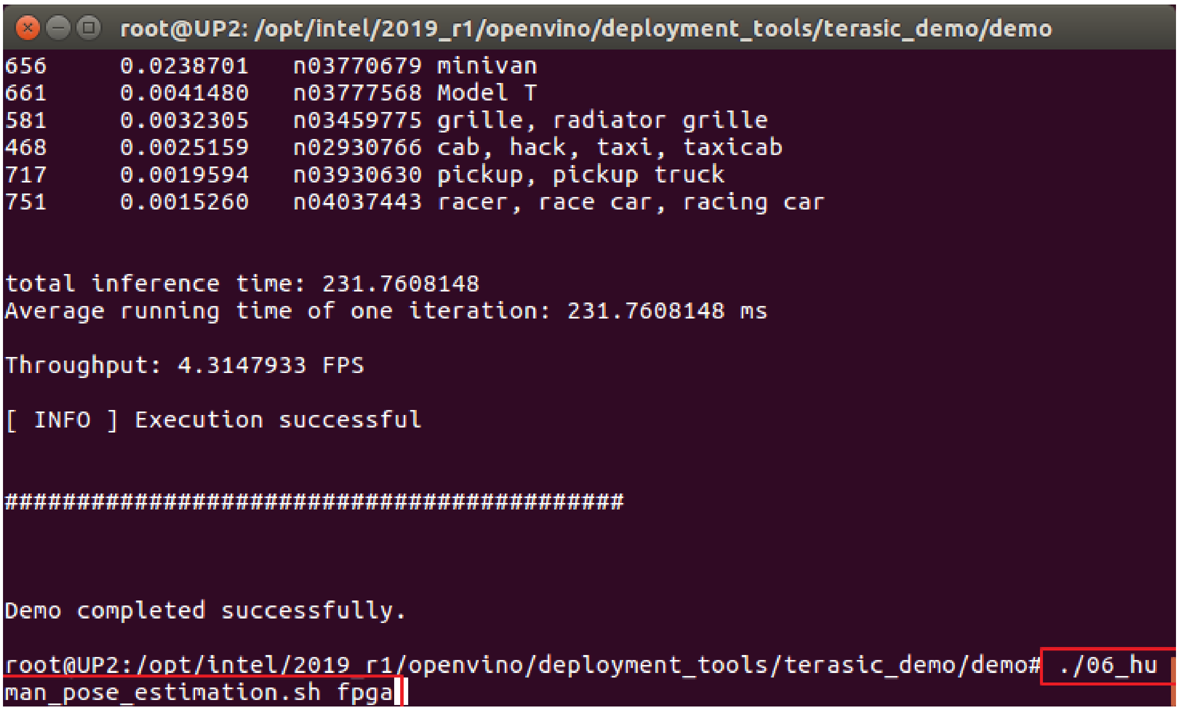
```
classid probability label
------- ----------- -----
479     0.7177463   n02974003 car wheel
436     0.0971364   n02814533 beach wagon, station wagon, wagon, estate car, bea
ch waggon, station waggon, waggon
511     0.0971364   n03100240 convertible
656     0.0357345   n03770679 minivan
817     0.0357345   n04285008 sports car, sport car
661     0.0048361   n03777568 Model T
581     0.0029333   n03459775 grille, radiator grille
717     0.0022844   n03930630 pickup, pickup truck
468     0.0022844   n02930766 cab, hack, taxi, taxicab
751     0.0013856   n04037443 racer, race car, racing car


total inference time: 67.8817555
Average running time of one iteration: 67.8817555 ms

Throughput: 14.7314988 FPS

[ INFO ] Execution successful


################################################
```

■ 06_human_pose_estimation

This demo can recognize human pose and display it.

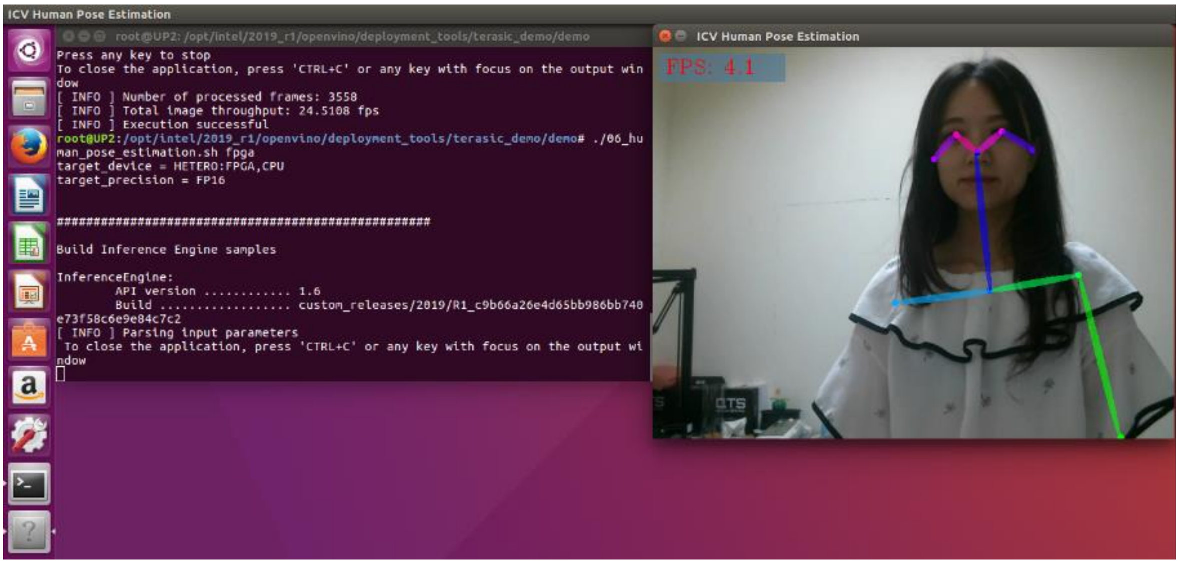
1) Plug a UVC USB camera to the host PC USB port.

2) Execute "./ 06_human_pose_estimation.sh fpga" to run the demo with FPGA.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
656     0.0238701   n03770679 minivan
661     0.0041480   n03777568 Model T
581     0.0032305   n03459775 grille, radiator grille
468     0.0025159   n02930766 cab, hack, taxi, taxicab
717     0.0019594   n03930630 pickup, pickup truck
751     0.0015260   n04037443 racer, race car, racing car


total inference time: 231.7608148
Average running time of one iteration: 231.7608148 ms

Throughput: 4.3147933 FPS

[ INFO ] Execution successful


################################################



Demo completed successfully.

root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./06_hu
man_pose_estimation.sh fpga
```

3) The result is shown in the figure below. Enter Ctrl+c to close the Application.



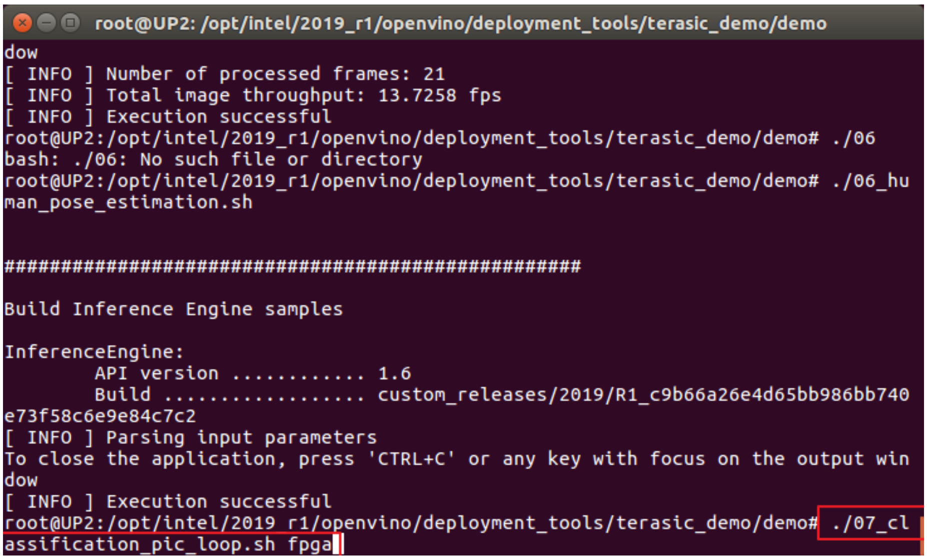
ICV Human Pose Estimation
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
ICV Human Pose Estimation
FPS: 4.1

```
Press any key to stop
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
[ INFO ] Number of processed frames: 3558
[ INFO ] Total image throughput: 24.5108 fps
[ INFO ] Execution successful
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./06_hu
man_pose_estimation.sh fpga
target_device = HETERO:FPGA,CPU
target_precision = FP16


###################################################

Build Inference Engine samples

InferenceEngine:
        API version ............ 1.6
        Build .................. custom_releases/2019/R1_c9b66a26e4d65bb986bb740
e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
 To close the application, press 'CTRL+C' or any key with focus on the output wi
ndow
```

07_classification_pic_loop

This demo bases demo 01_squeesnet_classification, and adds more pictures to run in a loop.

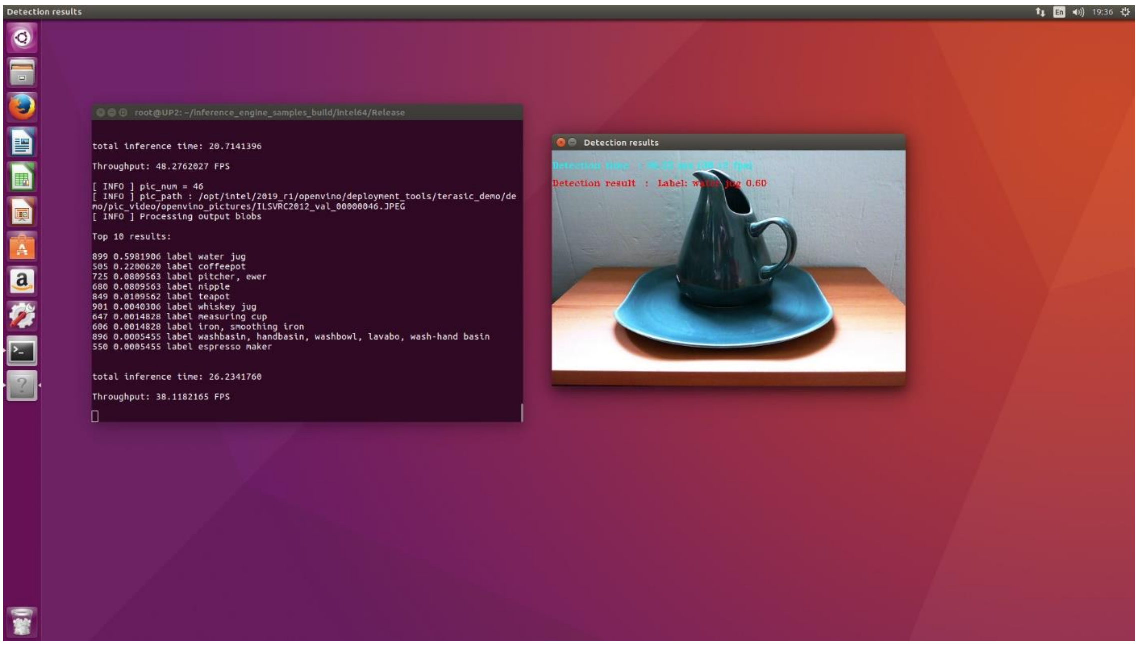
1) Execute "*./ 07_classification_pic_loop.sh fpga*" to run the demo with FPGA.



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
dow
[ INFO ] Number of processed frames: 21
[ INFO ] Total image throughput: 13.7258 fps
[ INFO ] Execution successful
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./06
bash: ./06: No such file or directory
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./06_hu
man_pose_estimation.sh


###################################################

Build Inference Engine samples

InferenceEngine:
        API version ............ 1.6
        Build .................. custom_releases/2019/R1_c9b66a26e4d65bb986bb740
e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
[ INFO ] Execution successful
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./07_cl
assification_pic_loop.sh fpga
```

2) The result is shown in the figure below.



```
total inference time: 20.7141396

Throughput: 48.2762027 FPS

[ INFO ] pic_num = 46
[ INFO ] pic_path : /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/de
mo/pic_video/openvino_pictures/ILSVRC2012_val_00000046.JPEG
[ INFO ] Processing output blobs

Top 10 results:

899 0.5981906 label water jug
505 0.2200620 label coffeepot
725 0.0809563 label pitcher, ewer
680 0.0809563 label nipple
849 0.0109562 label teapot
901 0.0040306 label whiskey jug
647 0.0014828 label measuring cup
606 0.0014828 label iron, smoothing iron
896 0.0005455 label washbasin, handbasin, washbowl, lavabo, wash-hand basin
550 0.0005455 label espresso maker


total inference time: 26.2341760

Throughput: 38.1182165 FPS
```

```
Detection result  :  Label: water jug 0.60
```

Note: Some demos execution result will show the frame rate, the result of these demos is depending on the performance of user's PC CPU and the FPGA device. The CPU grade is higher (I7 instead of I5), the result is better, the result is better if user execute the demo on the DE5a-Net-DDR4 instead of OSK.

Chapter 3

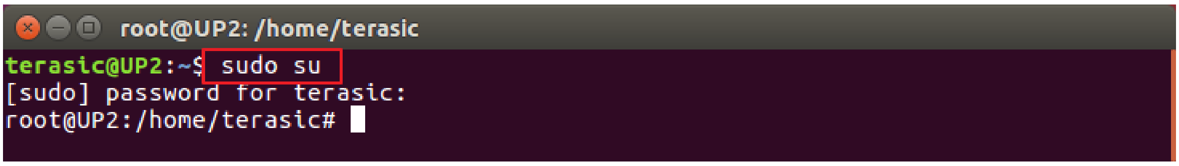
OpenVINO Starter Kit Lab

This chapter describes how to verify the experiment environment, implement the acceleration of users own AI demonstrations on the FPGA platform.

3.1 Verify the Environment of the Experiment

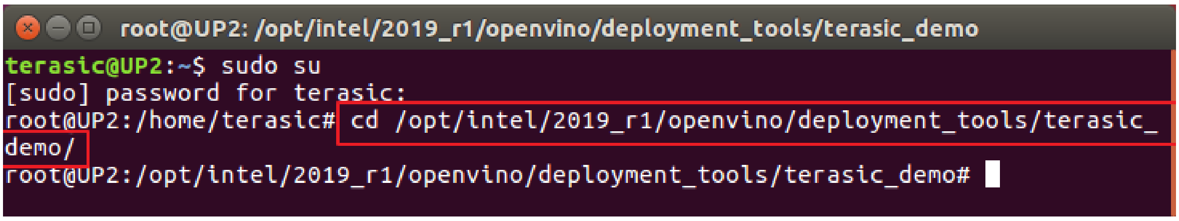
This section will show user how to verify the environment of the experiment by running the demo "02_security_barrier.sh" with CPU which is provided in terasic demo.

1. Open a terminal by right clicking on the Desktop.
2. Enter command "*sudo su*" to change the user to the root Super User, the password is terasic.



```
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic#
```

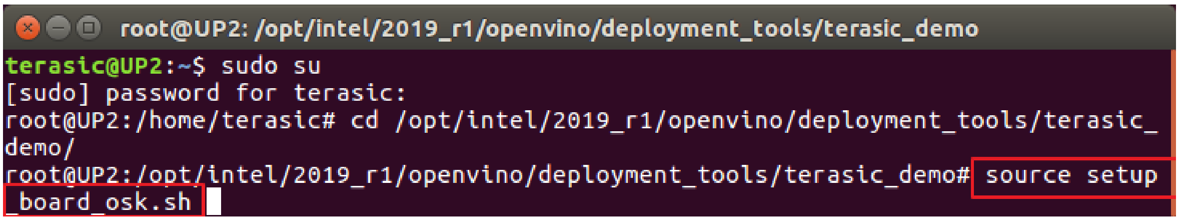
3. Enter "*cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo*" to switch the path.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

```
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo#
```

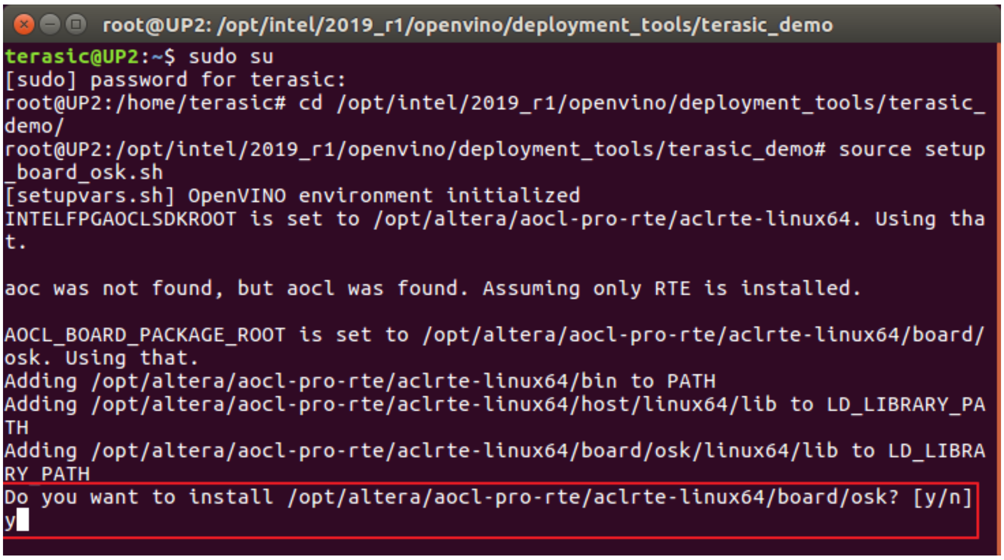
4. Enter "*source setup_board_osk.sh*".

   Note: User needs to setup the corresponding .sh file for the FPGA board, for example, source setup_board_de5a_net_ddr4.sh for DE5a-Net-DDR4 board.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

```
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# source setup
_board_osk.sh
```

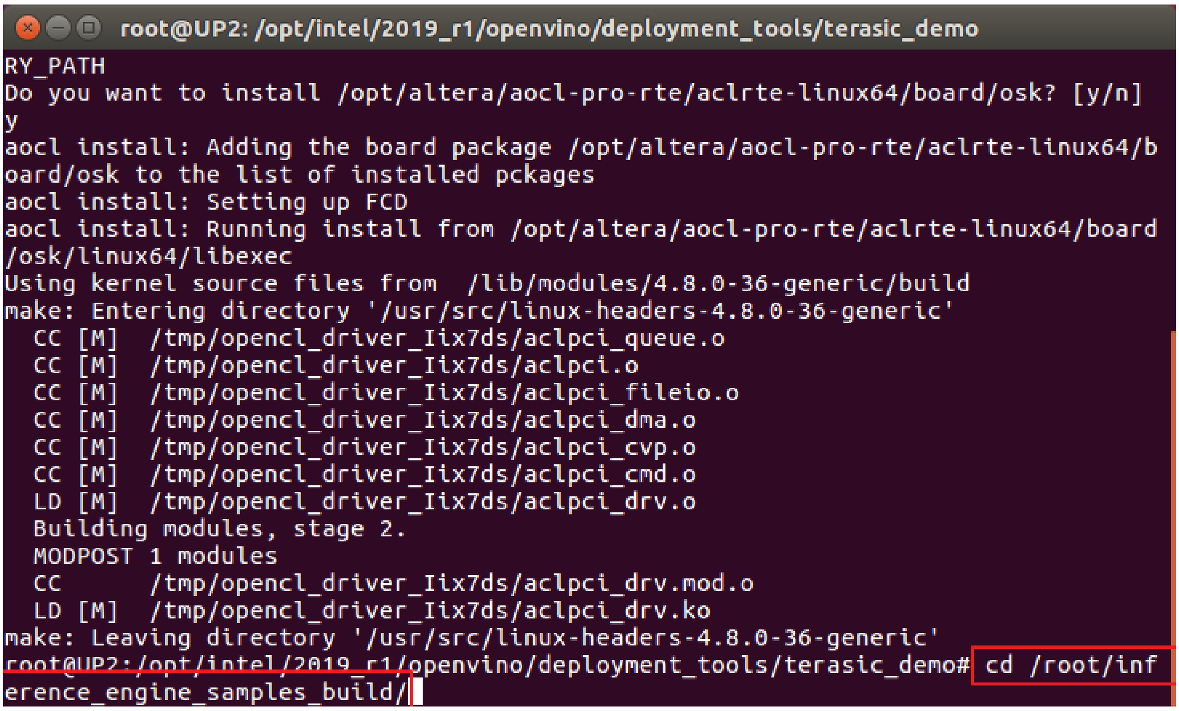
5. Enter "*y*" for the driver installation.



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

```
terasic@UP2:~$ sudo su
[sudo] password for terasic:
root@UP2:/home/terasic# cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# source setup
_board_osk.sh
[setupvars.sh] OpenVINO environment initialized
INTELFPGAOCLSDKROOT is set to /opt/altera/aocl-pro-rte/aclrte-linux64. Using tha
t.

aoc was not found, but aocl was found. Assuming only RTE is installed.

AOCL_BOARD_PACKAGE_ROOT is set to /opt/altera/aocl-pro-rte/aclrte-linux64/board/
osk. Using that.
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/bin to PATH
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/host/linux64/lib to LD_LIBRARY_PA
TH
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/board/osk/linux64/lib to LD_LIBRA
RY_PATH
Do you want to install /opt/altera/aocl-pro-rte/aclrte-linux64/board/osk? [y/n]
y
```

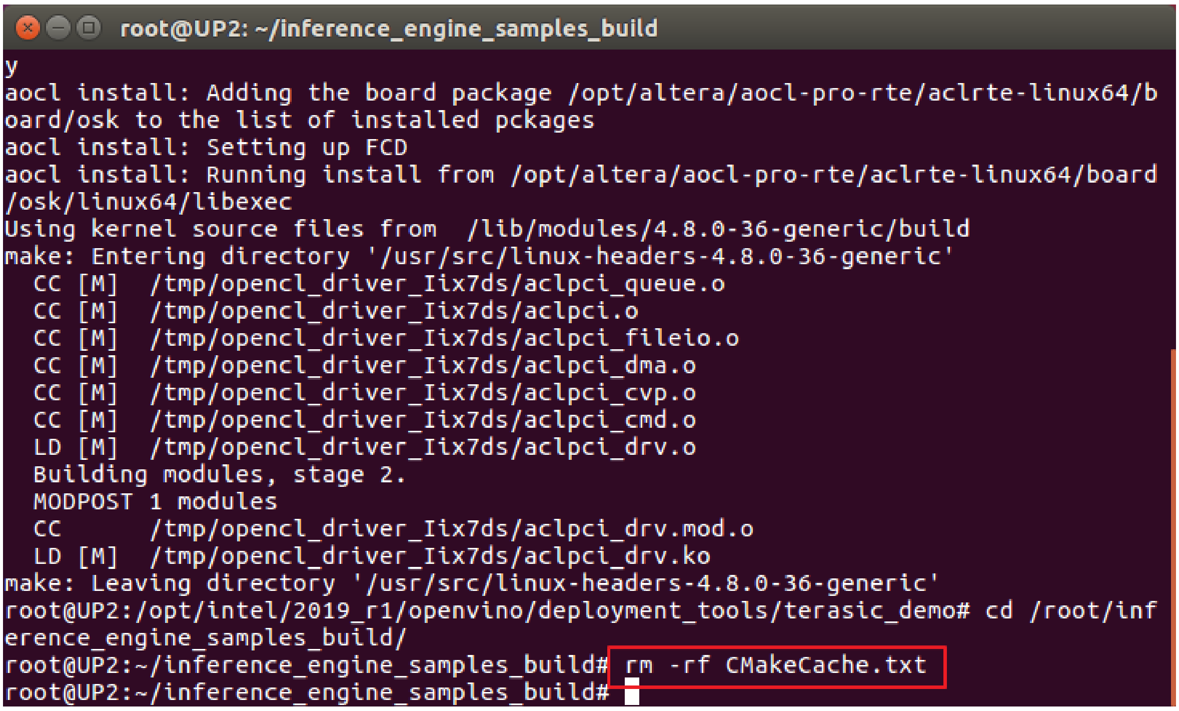
6. Enter *cd /root/inference_engine_samples_build/*



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo

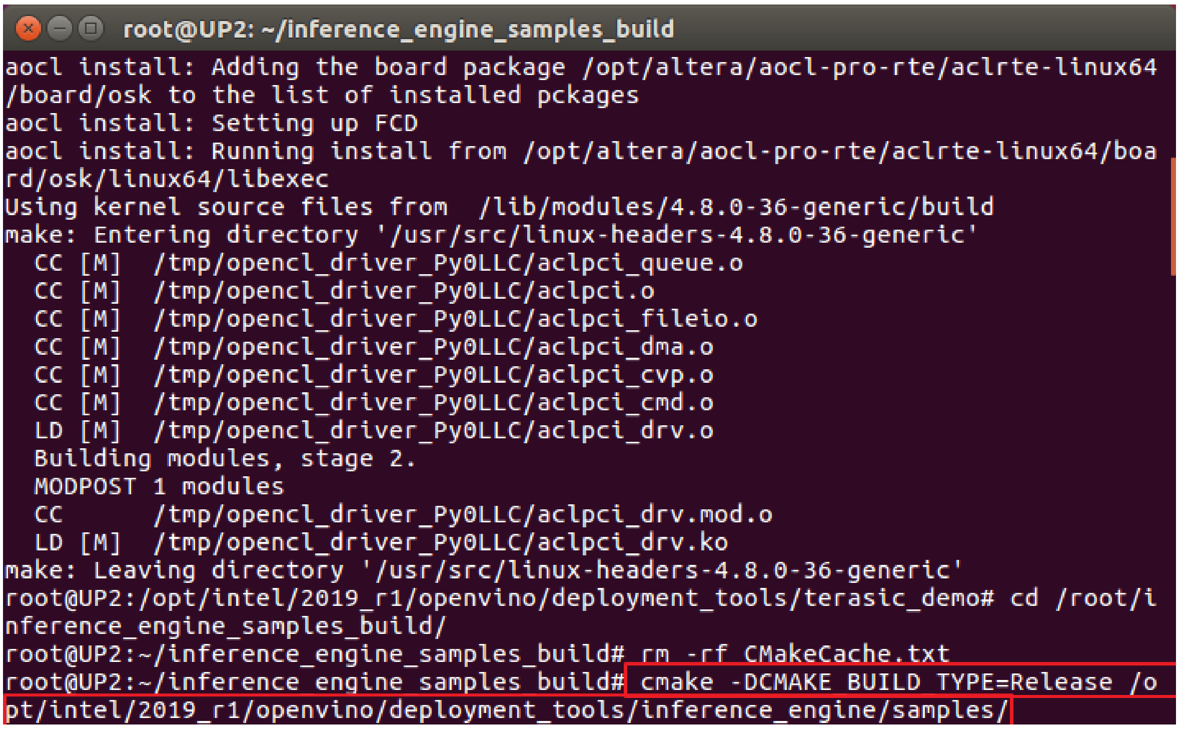
```
RY_PATH
Do you want to install /opt/altera/aocl-pro-rte/aclrte-linux64/board/osk? [y/n]
y
aocl install: Adding the board package /opt/altera/aocl-pro-rte/aclrte-linux64/b
oard/osk to the list of installed pckages
aocl install: Setting up FCD
aocl install: Running install from /opt/altera/aocl-pro-rte/aclrte-linux64/board
/osk/linux64/libexec
Using kernel source files from  /lib/modules/4.8.0-36-generic/build
make: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_queue.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_fileio.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_dma.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_cvp.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_cmd.o
  LD [M]  /tmp/opencl_driver_Iix7ds/aclpci_drv.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/opencl_driver_Iix7ds/aclpci_drv.mod.o
  LD [M]  /tmp/opencl_driver_Iix7ds/aclpci_drv.ko
make: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# cd /root/inf
erence_engine_samples_build/
```

7. Enter “*rm -rf CMakeCache.txt*”

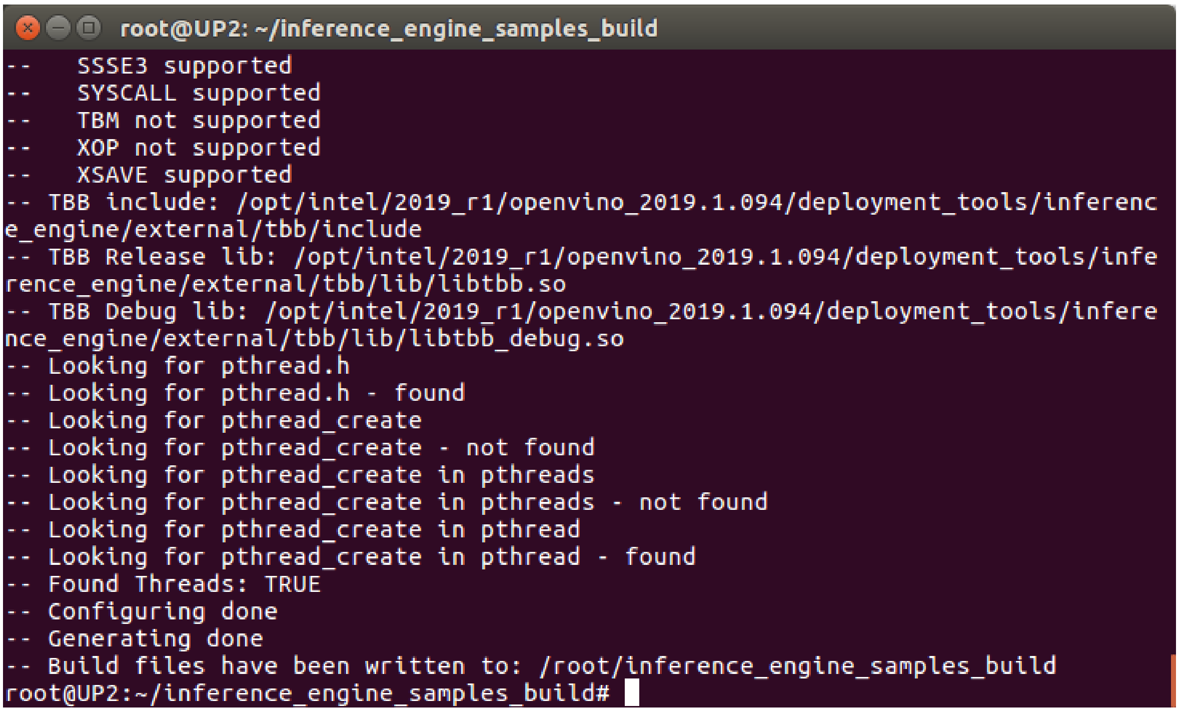


```
root@UP2: ~/inference_engine_samples_build
y
aocl install: Adding the board package /opt/altera/aocl-pro-rte/aclrte-linux64/b
oard/osk to the list of installed pckages
aocl install: Setting up FCD
aocl install: Running install from /opt/altera/aocl-pro-rte/aclrte-linux64/board
/osk/linux64/libexec
Using kernel source files from  /lib/modules/4.8.0-36-generic/build
make: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_queue.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_fileio.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_dma.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_cvp.o
  CC [M]  /tmp/opencl_driver_Iix7ds/aclpci_cmd.o
  LD [M]  /tmp/opencl_driver_Iix7ds/aclpci_drv.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/opencl_driver_Iix7ds/aclpci_drv.mod.o
  LD [M]  /tmp/opencl_driver_Iix7ds/aclpci_drv.ko
make: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# cd /root/inf
erence_engine_samples_build/
root@UP2:~/inference_engine_samples_build# rm -rf CMakeCache.txt
root@UP2:~/inference_engine_samples_build# 
```

8. Enter "*cmake -DCMAKE_BUILD_TYPE=Release \ /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/*"

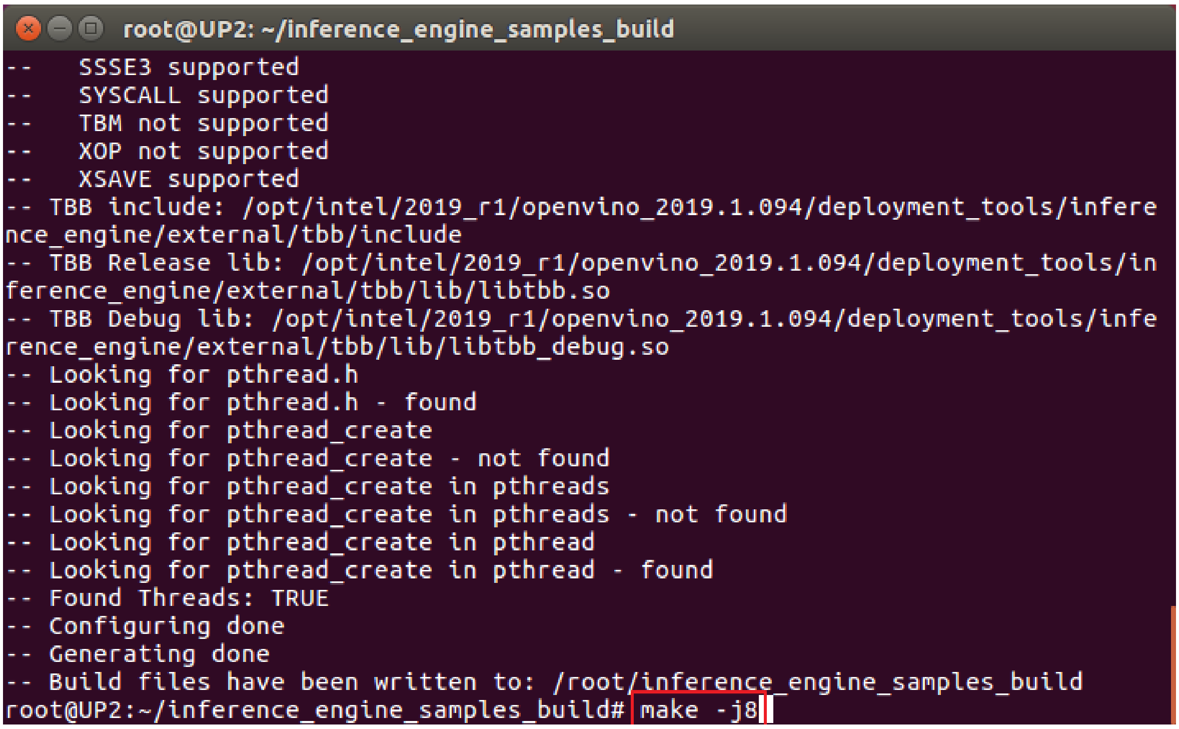


```
root@UP2: ~/inference_engine_samples_build
aocl install: Adding the board package /opt/altera/aocl-pro-rte/aclrte-linux64
/board/osk to the list of installed pckages
aocl install: Setting up FCD
aocl install: Running install from /opt/altera/aocl-pro-rte/aclrte-linux64/boa
rd/osk/linux64/libexec
Using kernel source files from  /lib/modules/4.8.0-36-generic/build
make: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci_queue.o
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci.o
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci_fileio.o
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci_dma.o
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci_cvp.o
  CC [M]  /tmp/opencl_driver_Py0LLC/aclpci_cmd.o
  LD [M]  /tmp/opencl_driver_Py0LLC/aclpci_drv.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/opencl_driver_Py0LLC/aclpci_drv.mod.o
  LD [M]  /tmp/opencl_driver_Py0LLC/aclpci_drv.ko
make: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo# cd /root/i
nference_engine_samples_build/
root@UP2:~/inference_engine_samples_build# rm -rf CMakeCache.txt
root@UP2:~/inference_engine_samples_build# cmake -DCMAKE_BUILD_TYPE=Release /o
pt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/
```



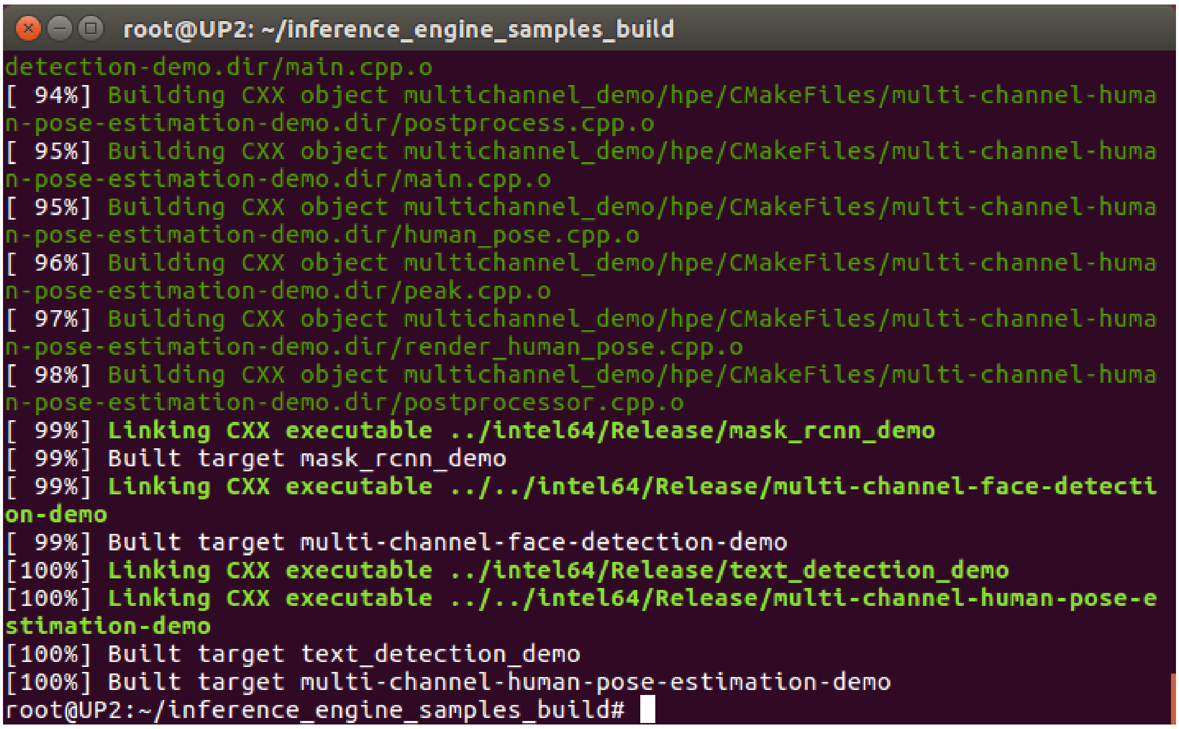
```
--   SSSE3 supported
--   SYSCALL supported
--   TBM not supported
--   XOP not supported
--   XSAVE supported
-- TBB include: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/inferenc
e_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/infe
rence_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/infere
nce_engine/external/tbb/lib/libtbb_debug.so
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /root/inference_engine_samples_build
root@UP2:~/inference_engine_samples_build#
```

9. Enter “*make -j8*”



```
--   SSSE3 supported
--   SYSCALL supported
--   TBM not supported
--   XOP not supported
--   XSAVE supported
-- TBB include: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/infere
nce_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/in
ference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/openvino_2019.1.094/deployment_tools/infe
rence_engine/external/tbb/lib/libtbb_debug.so
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /root/inference_engine_samples_build
root@UP2:~/inference_engine_samples_build# make -j8
```

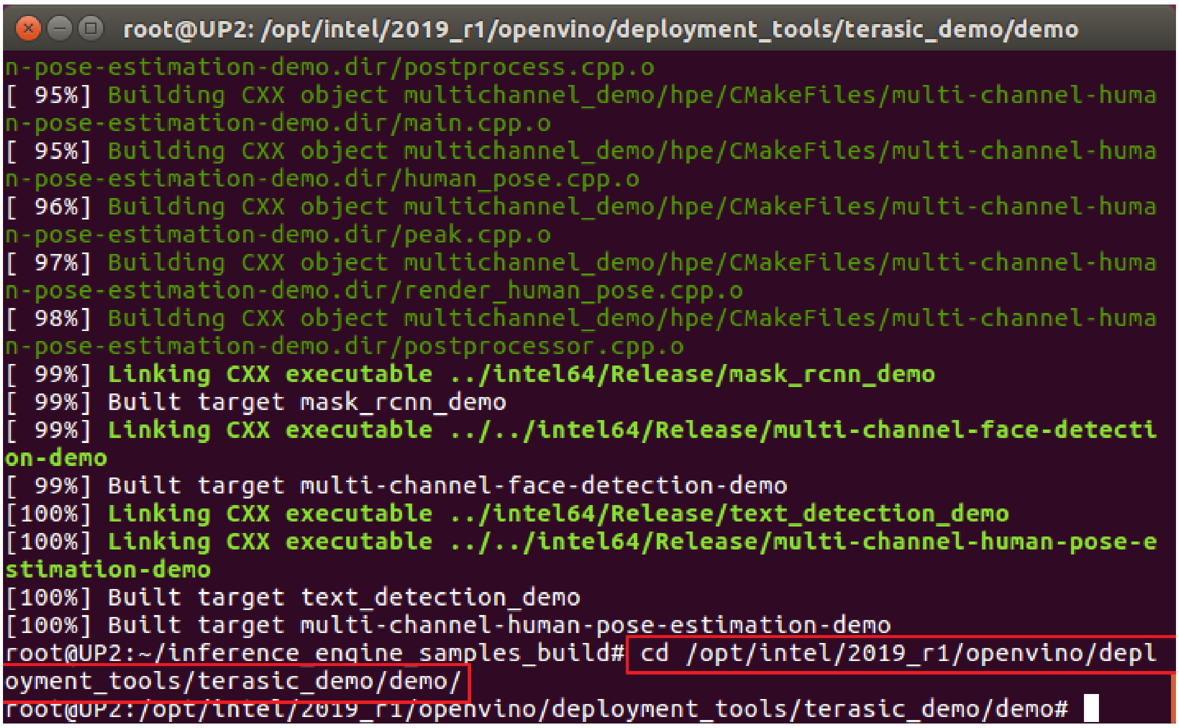
10. Wait until the build process is finished.



root@UP2: ~/inference_engine_samples_build

```
detection-demo.dir/main.cpp.o
[ 94%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocess.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/main.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/human_pose.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/peak.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/render_human_pose.cpp.o
[ 98%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocessor.cpp.o
[ 99%] Linking CXX executable ../intel64/Release/mask_rcnn_demo
[ 99%] Built target mask_rcnn_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-face-detecti
on-demo
[ 99%] Built target multi-channel-face-detection-demo
[100%] Linking CXX executable ../intel64/Release/text_detection_demo
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-e
stimation-demo
[100%] Built target text_detection_demo
[100%] Built target multi-channel-human-pose-estimation-demo
root@UP2:~/inference_engine_samples_build#
```

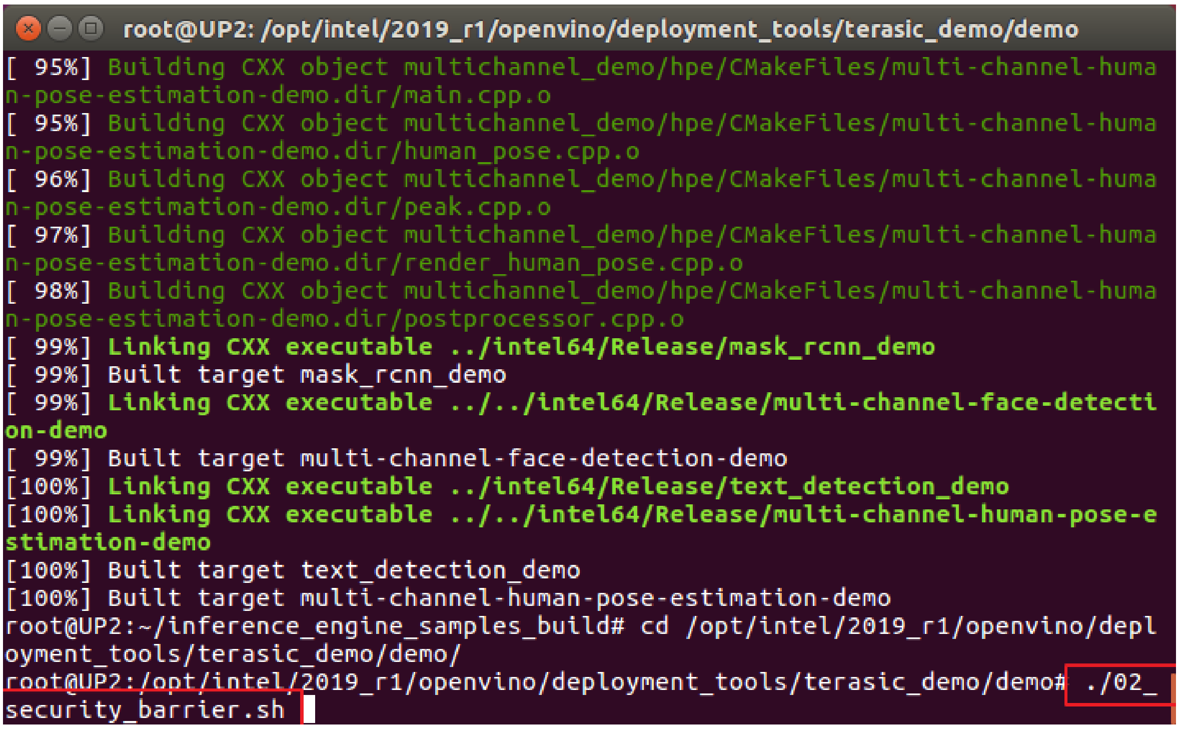
11. Enter "*cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo*" to switch the workspace.



root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

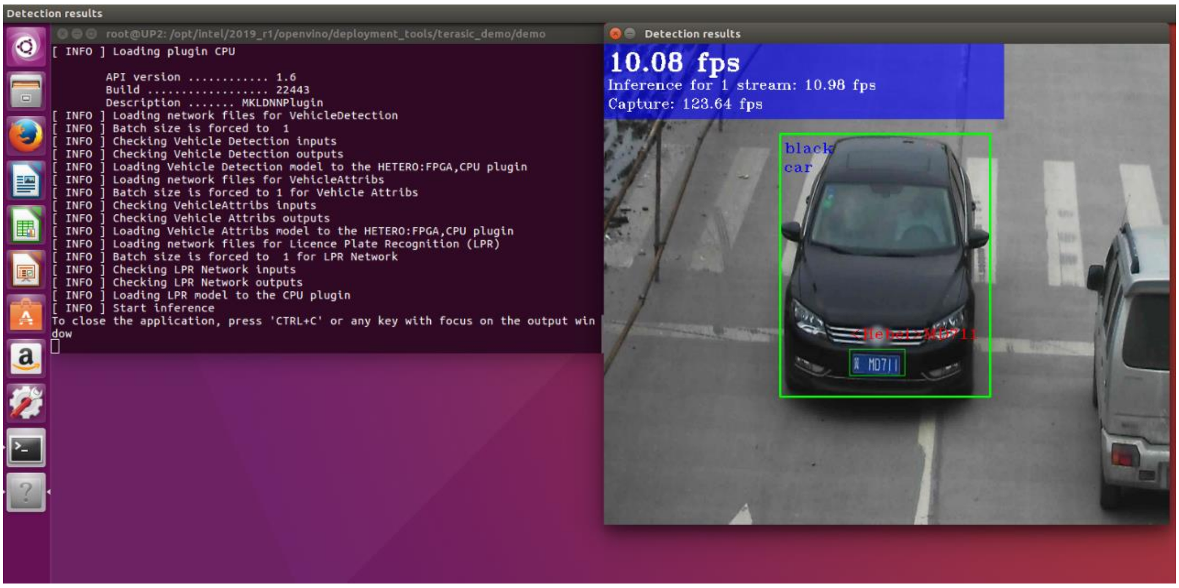
```
n-pose-estimation-demo.dir/postprocess.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/main.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/human_pose.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/peak.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/render_human_pose.cpp.o
[ 98%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocessor.cpp.o
[ 99%] Linking CXX executable ../intel64/Release/mask_rcnn_demo
[ 99%] Built target mask_rcnn_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-face-detecti
on-demo
[ 99%] Built target multi-channel-face-detection-demo
[100%] Linking CXX executable ../intel64/Release/text_detection_demo
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-e
stimation-demo
[100%] Built target text_detection_demo
[100%] Built target multi-channel-human-pose-estimation-demo
root@UP2:~/inference_engine_samples_build# cd /opt/intel/2019_r1/openvino/depl
oyment_tools/terasic_demo/demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo#
```

12. Enter "*./02_security_barrier.sh*".



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/main.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/human_pose.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/peak.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/render_human_pose.cpp.o
[ 98%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocessor.cpp.o
[ 99%] Linking CXX executable ../intel64/Release/mask_rcnn_demo
[ 99%] Built target mask_rcnn_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-face-detecti
on-demo
[ 99%] Built target multi-channel-face-detection-demo
[100%] Linking CXX executable ../intel64/Release/text_detection_demo
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-e
stimation-demo
[100%] Built target text_detection_demo
[100%] Built target multi-channel-human-pose-estimation-demo
root@UP2:~/inference_engine_samples_build# cd /opt/intel/2019_r1/openvino/depl
oyment_tools/terasic_demo/demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# ./02_
security_barrier.sh
```

13. The result of the Demo is as shown below.



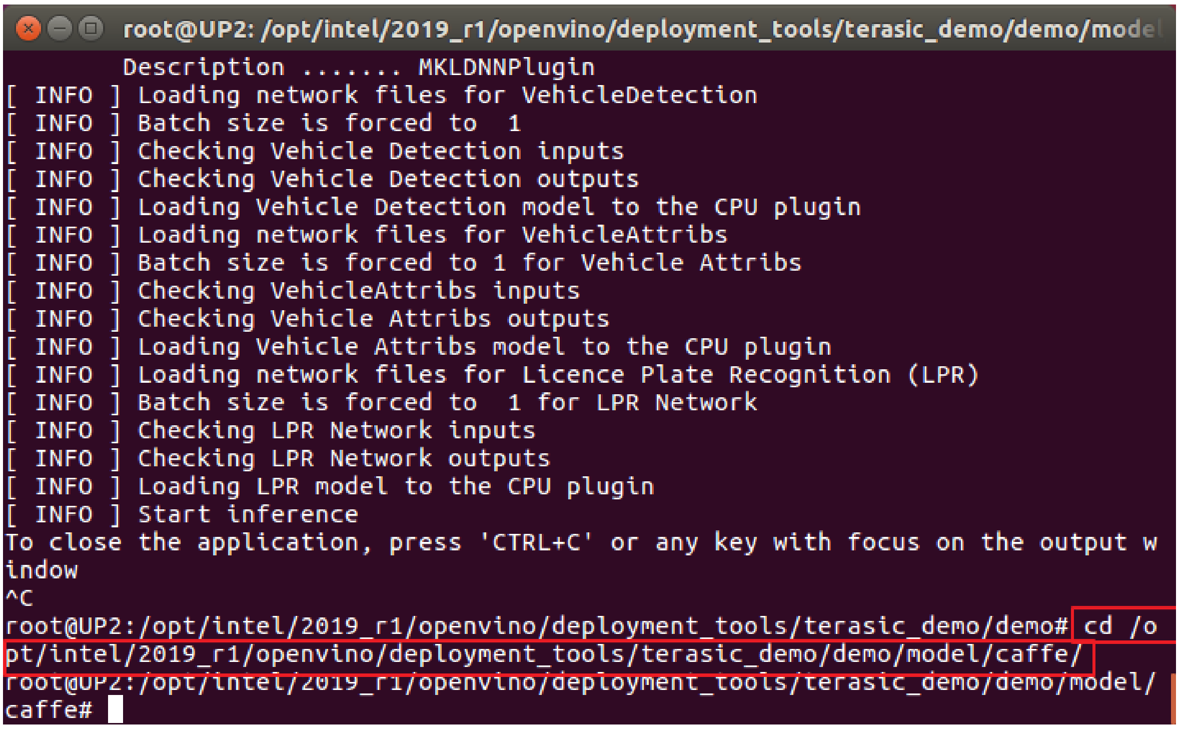
```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
[ INFO ] Loading plugin CPU

        API version ............ 1.6
        Build .................. 22443
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files for VehicleDetection
[ INFO ] Batch size is forced to  1
[ INFO ] Checking Vehicle Detection inputs
[ INFO ] Checking Vehicle Detection outputs
[ INFO ] Loading Vehicle Detection model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for VehicleAttribs
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the HETERO:FPGA,CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output win
dow
```

```
10.08 fps
Inference for 1 stream: 10.98 fps
Capture: 123.64 fps
```

3.2 Lab 1. How to use the Model Optimizer to transform the Model?

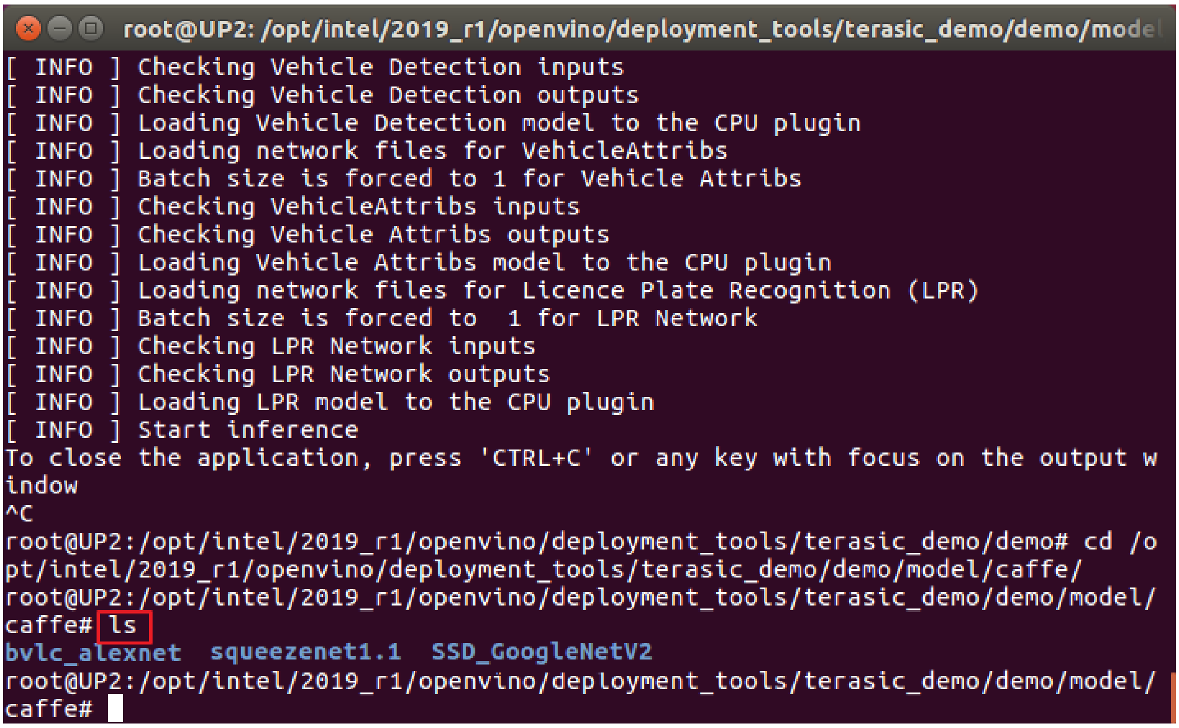
This section will show user how to use the Model Optimizer tool to get the IR parameters from pre-downloaded caffe model file "squeesenet1.1" which will be used by Inference Engine app.

1. Enter "*cd /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/*" to switch the path to the pre-downloaded Model folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/mode

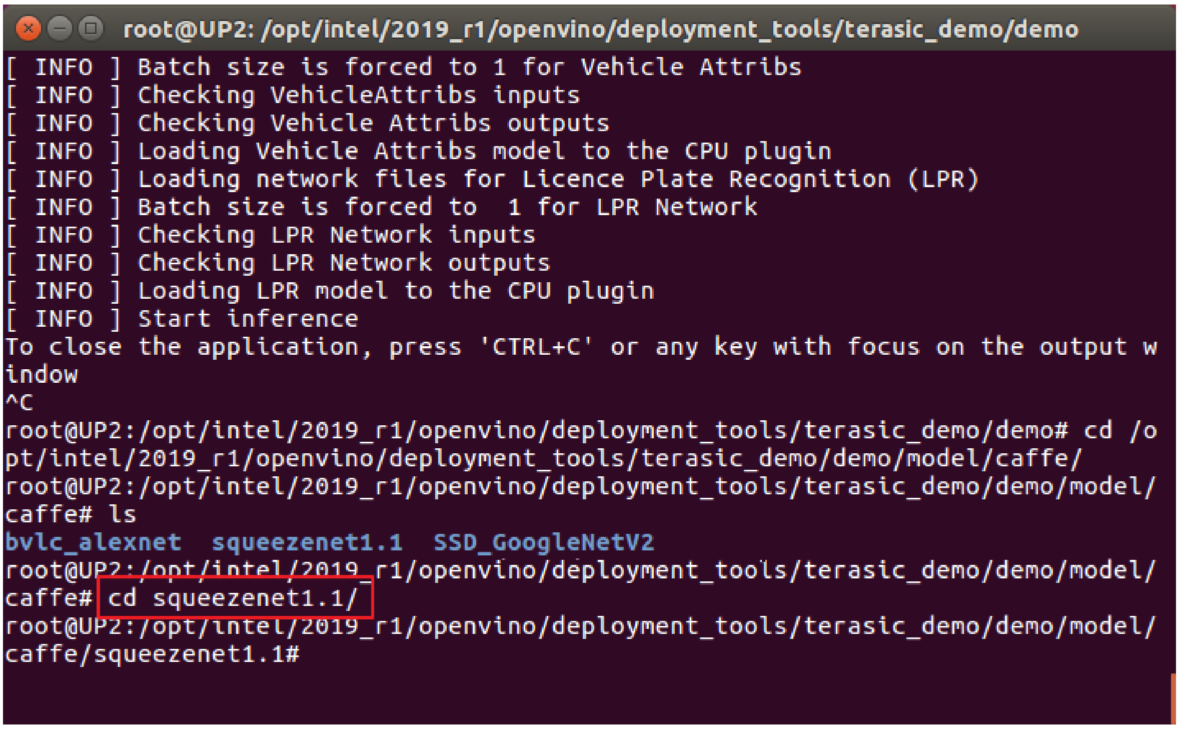
```
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files for VehicleDetection
[ INFO ] Batch size is forced to  1
[ INFO ] Checking Vehicle Detection inputs
[ INFO ] Checking Vehicle Detection outputs
[ INFO ] Loading Vehicle Detection model to the CPU plugin
[ INFO ] Loading network files for VehicleAttribs
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# 
```

2. Enter "*ls*" to see the folder information, which includes bvlc_alexnet, squeezent1.1, and SSD_GoogleNetV2 models.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/mode

```
[ INFO ] Checking Vehicle Detection inputs
[ INFO ] Checking Vehicle Detection outputs
[ INFO ] Loading Vehicle Detection model to the CPU plugin
[ INFO ] Loading network files for VehicleAttribs
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# 
```

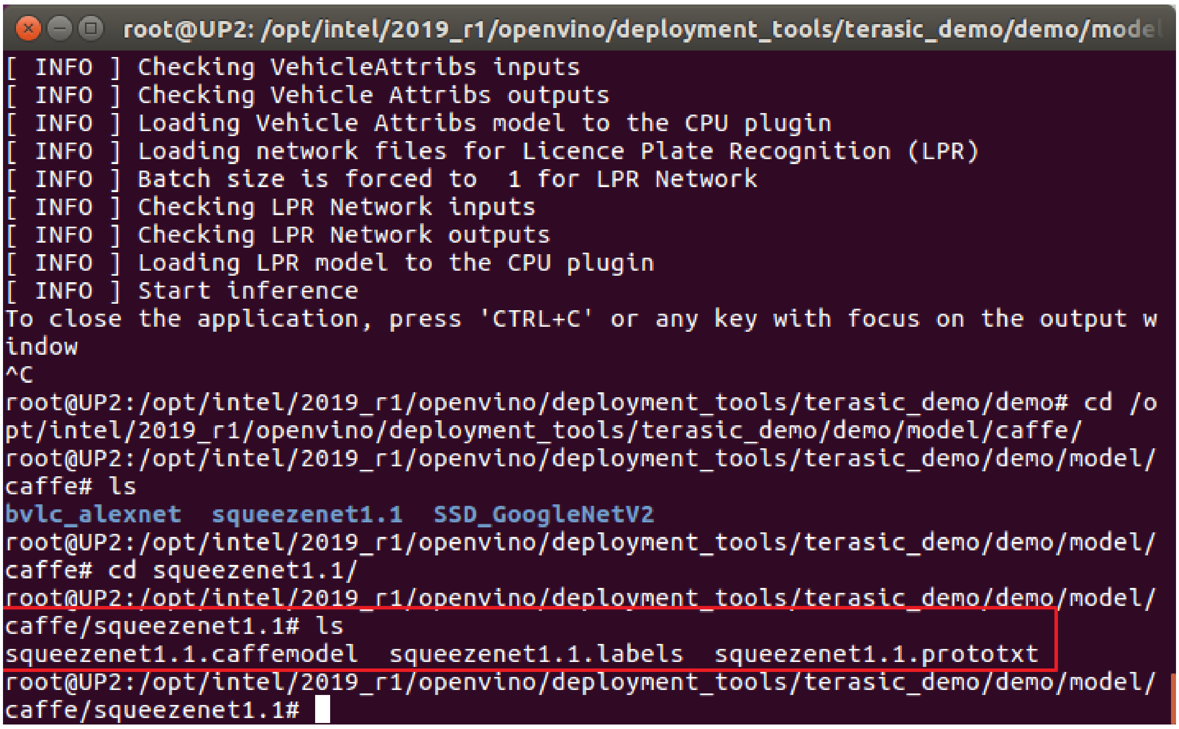
3. Enter "*cd squeezenet1.1*" to select the corresponding Model folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
[ INFO ] Batch size is forced to 1 for Vehicle Attribs
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# cd squeezenet1.1/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1#
```

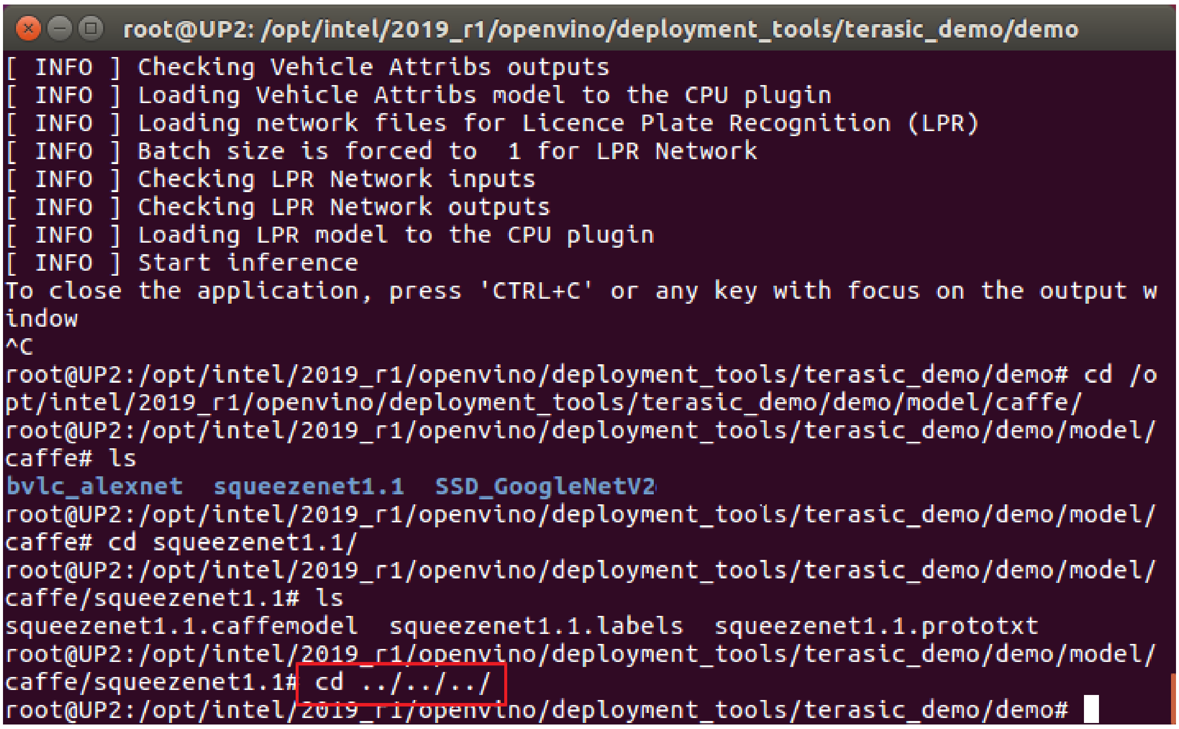
4. Enter "*ls*" to see the composition of the model, there are 3 files:

   - squeezenet1.1.caffemodel is the file to describe the adjusted weights and biases for the trained model.
   - squeezenet1.1.labels is the label file for classification model.
   - squeezenet1.1.prototxt is the description file for the model structure.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/mode

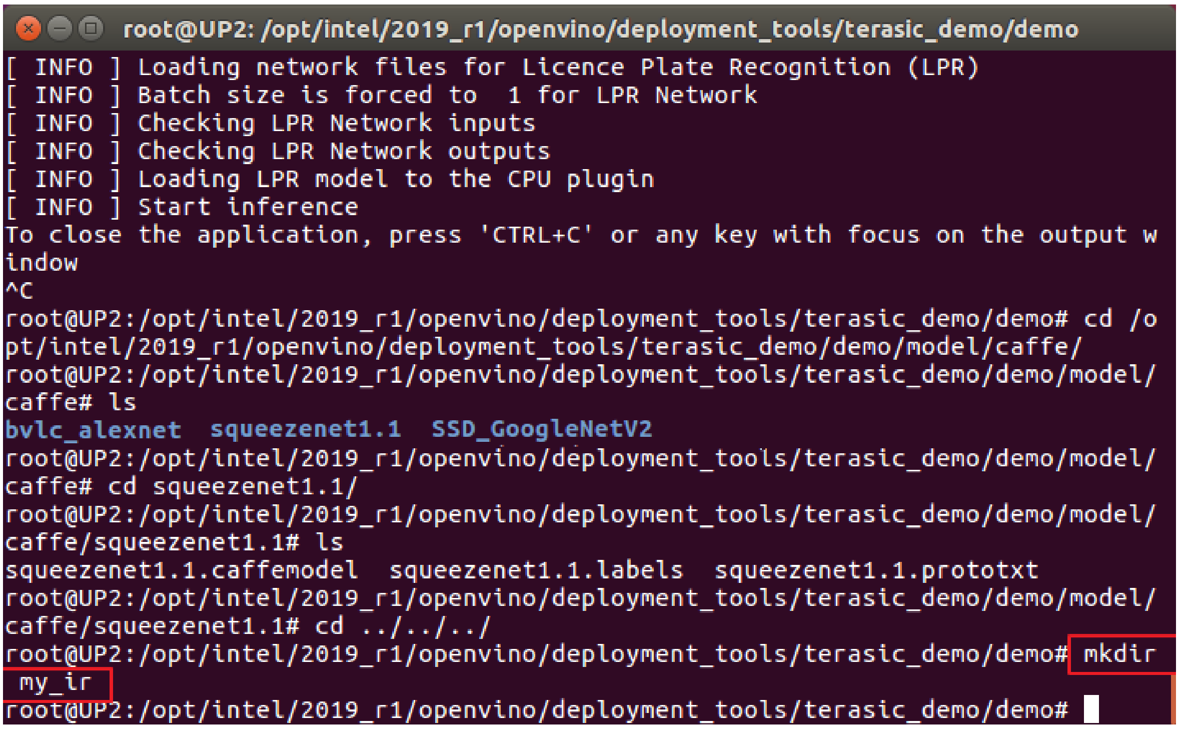
```
[ INFO ] Checking VehicleAttribs inputs
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# cd squeezenet1.1/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# 
```

5. Enter "*cd ../../../*" to go back to the demo folder.



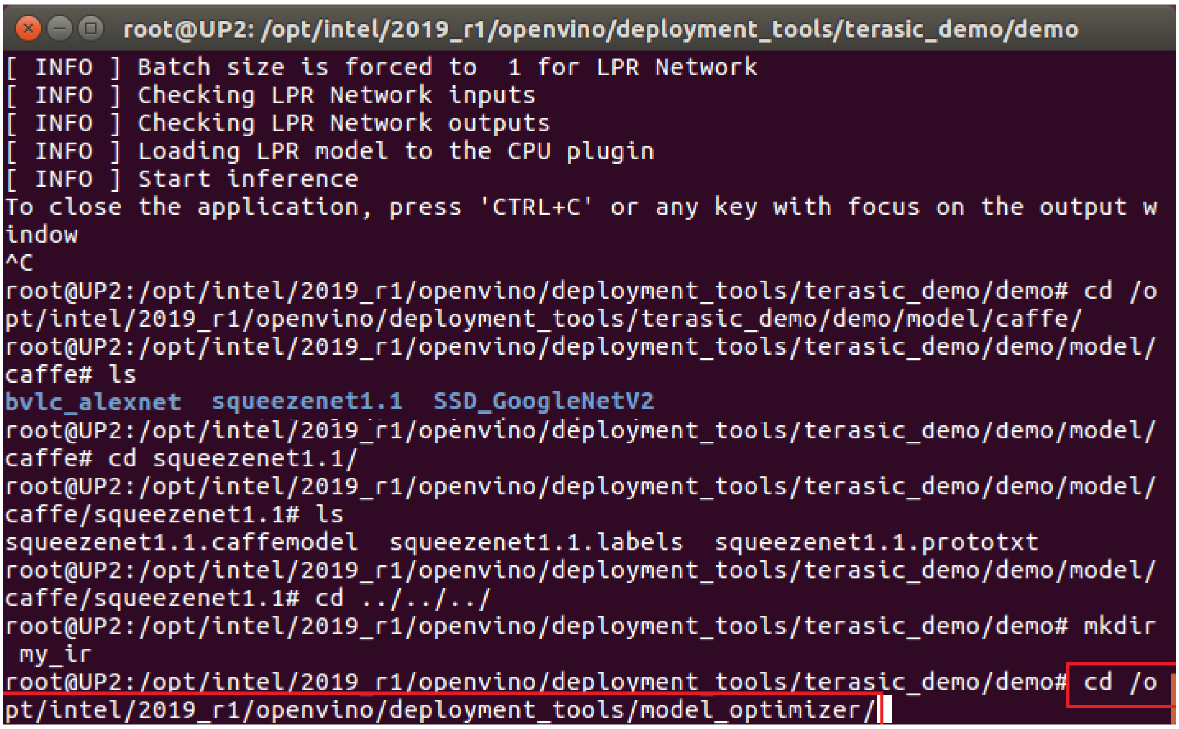
```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
[ INFO ] Checking Vehicle Attribs outputs
[ INFO ] Loading Vehicle Attribs model to the CPU plugin
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# cd squeezenet1.1/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# cd ../../../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# 
```

6. Enter “*mkdir my_ir*” to create a new folder for saving IR files.



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo
[ INFO ] Loading network files for Licence Plate Recognition (LPR)
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# cd squeezenet1.1/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# cd ../../../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# mkdir
 my_ir
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# 
```

7. Enter “*cd /opt/intel/2019_r1/openvino/deployment_tools/model_optimizer*” to switch the workspace to the Model Optimizer folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo

```
[ INFO ] Batch size is forced to  1 for LPR Network
[ INFO ] Checking LPR Network inputs
[ INFO ] Checking LPR Network outputs
[ INFO ] Loading LPR model to the CPU plugin
[ INFO ] Start inference
To close the application, press 'CTRL+C' or any key with focus on the output w
indow
^C
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe# cd squeezenet1.1/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# cd ../../../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# mkdir
 my_ir
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/model_optimizer/
```

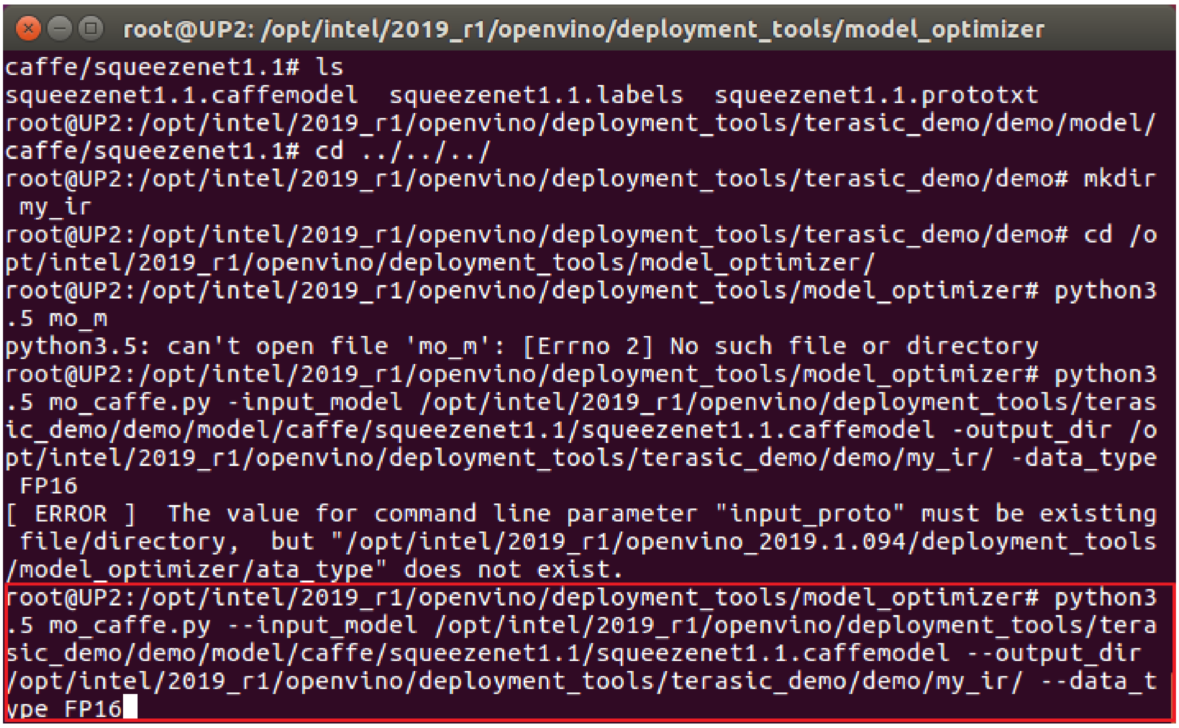
8. Enter "*python3.5 mo_caffe.py *

--input_model /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/

*model/caffe/squeezenet1.1/squeezenet1.1.caffemodel *

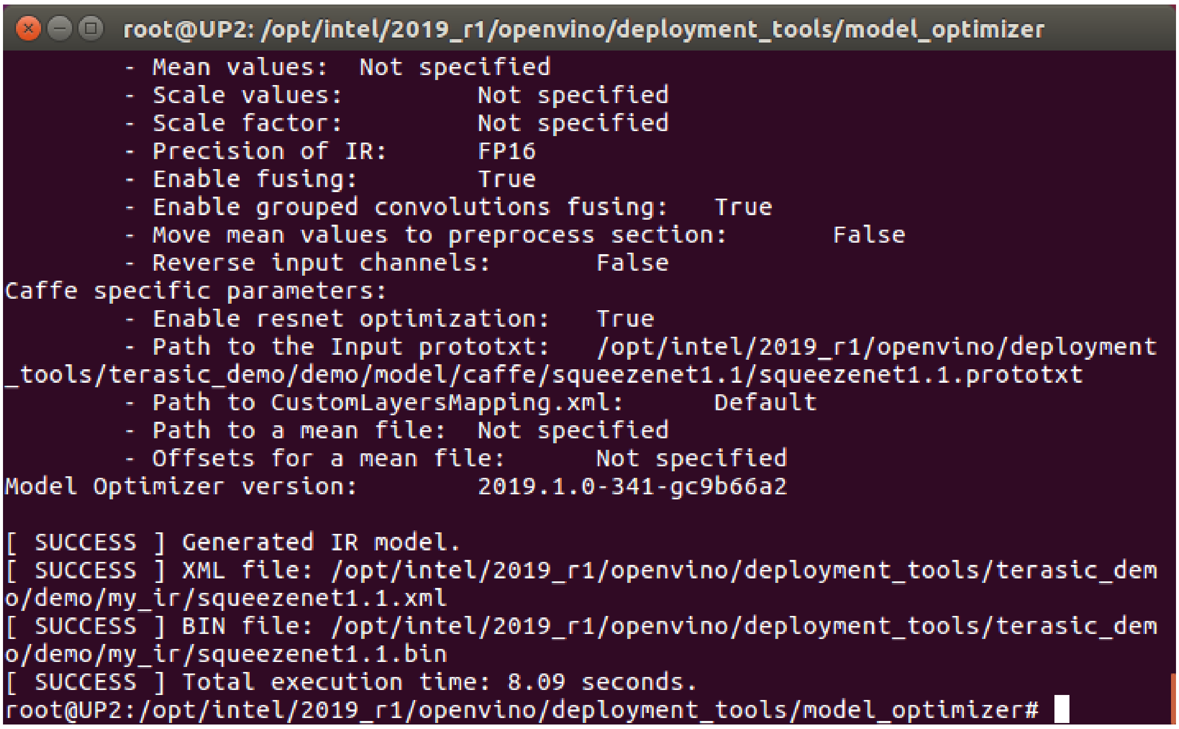
*--output_dir /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/my_ir *

--data_type FP16".

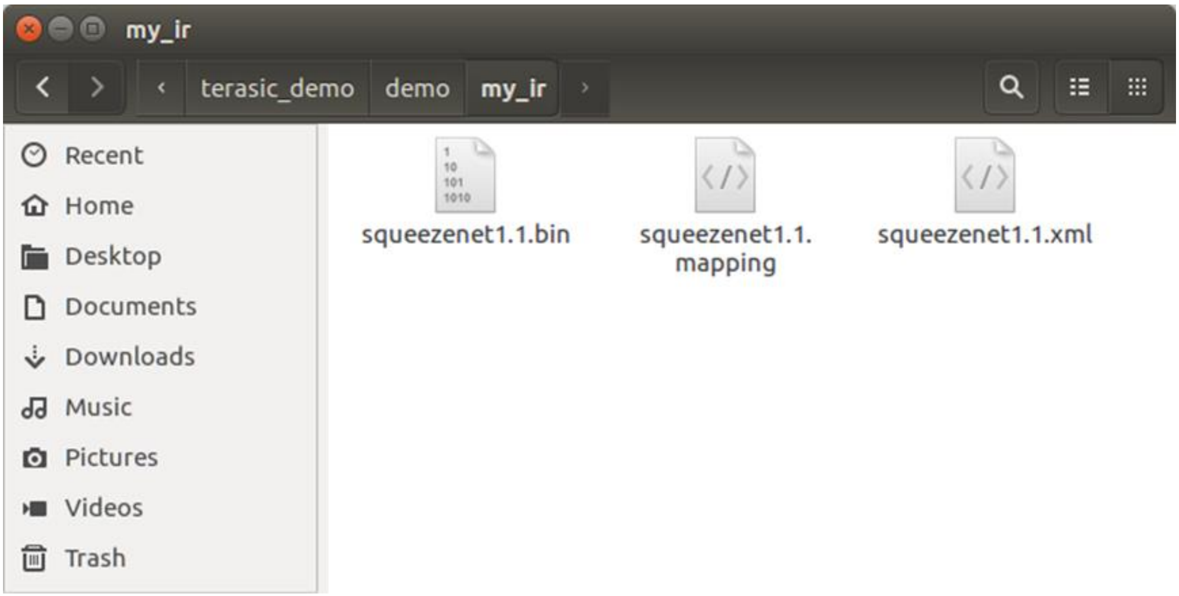

root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/model_optimizer

```
caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/
caffe/squeezenet1.1# cd ../../../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# mkdir
 my_ir
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo# cd /o
pt/intel/2019_r1/openvino/deployment_tools/model_optimizer/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# python3
.5 mo_m
python3.5: can't open file 'mo_m': [Errno 2] No such file or directory
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# python3
.5 mo_caffe.py -input_model /opt/intel/2019_r1/openvino/deployment_tools/teras
ic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.caffemodel -output_dir /o
pt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/my_ir/ -data_type
 FP16
[ ERROR ]  The value for command line parameter "input_proto" must be existing
 file/directory,  but "/opt/intel/2019_r1/openvino_2019.1.094/deployment_tools
/model_optimizer/ata_type" does not exist.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# python3
.5 mo_caffe.py --input_model /opt/intel/2019_r1/openvino/deployment_tools/tera
sic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.caffemodel --output_dir
/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/my_ir/ --data_t
ype FP16
```

9. The corresponding IR files are generated in the my_ir folder.



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/model_optimizer
        - Mean values:  Not specified
        - Scale values:         Not specified
        - Scale factor:         Not specified
        - Precision of IR:      FP16
        - Enable fusing:        True
        - Enable grouped convolutions fusing:   True
        - Move mean values to preprocess section:       False
        - Reverse input channels:       False
Caffe specific parameters:
        - Enable resnet optimization:   True
        - Path to the Input prototxt:   /opt/intel/2019_r1/openvino/deployment
_tools/terasic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.prototxt
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2019.1.0-341-gc9b66a2

[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.09 seconds.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# 
```

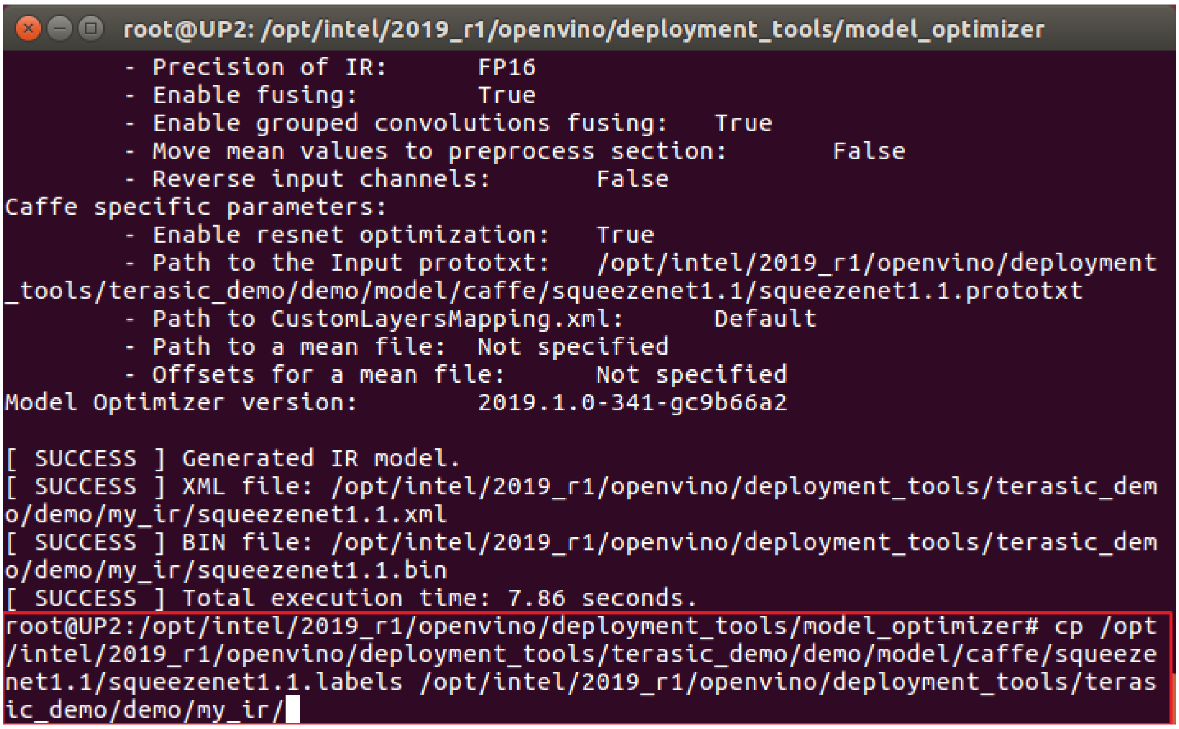


10. Copy the .label file from Model folder to the my_ir folder by entering

    “*cp *

    /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/

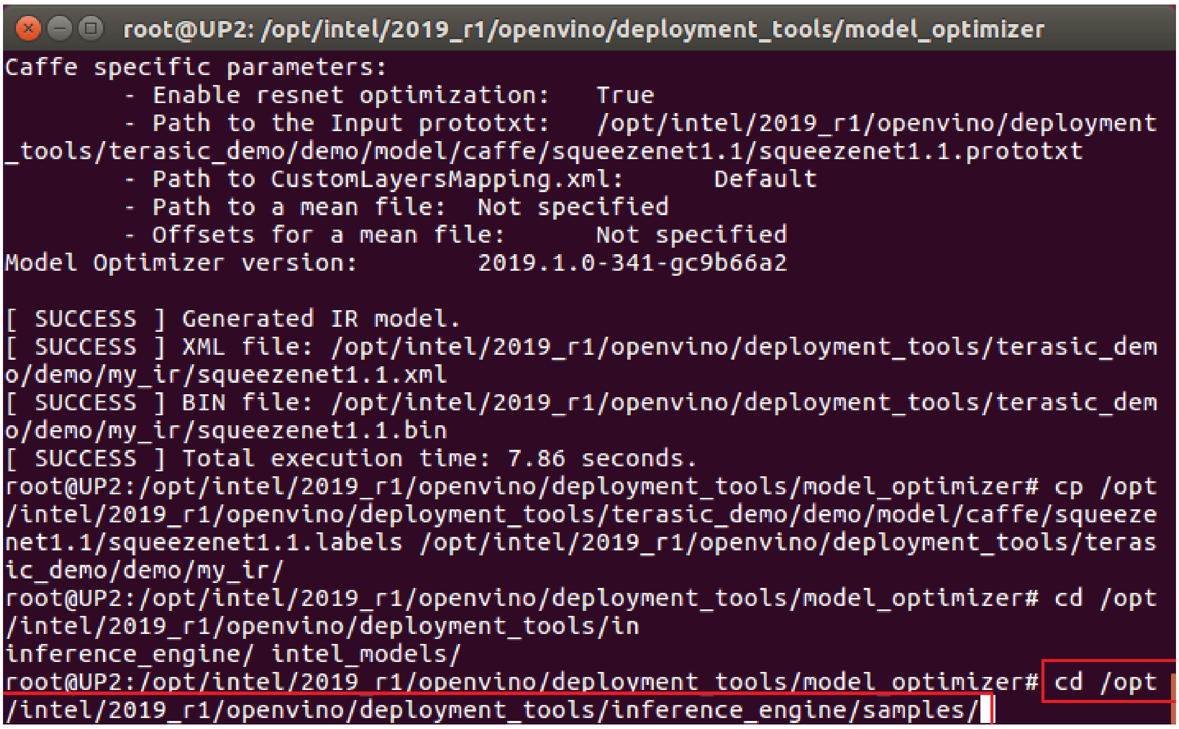
    *model/caffe/squeezenet1.1/squeezenet1.1.labels *
    /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/my_ir/ ”.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/model_optimizer

```
        - Precision of IR:      FP16
        - Enable fusing:        True
        - Enable grouped convolutions fusing:   True
        - Move mean values to preprocess section:       False
        - Reverse input channels:       False
Caffe specific parameters:
        - Enable resnet optimization:   True
        - Path to the Input prototxt:   /opt/intel/2019_r1/openvino/deployment
_tools/terasic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.prototxt
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2019.1.0-341-gc9b66a2

[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 7.86 seconds.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cp /opt
/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/squeeze
net1.1/squeezenet1.1.labels /opt/intel/2019_r1/openvino/deployment_tools/teras
ic_demo/demo/my_ir/
```

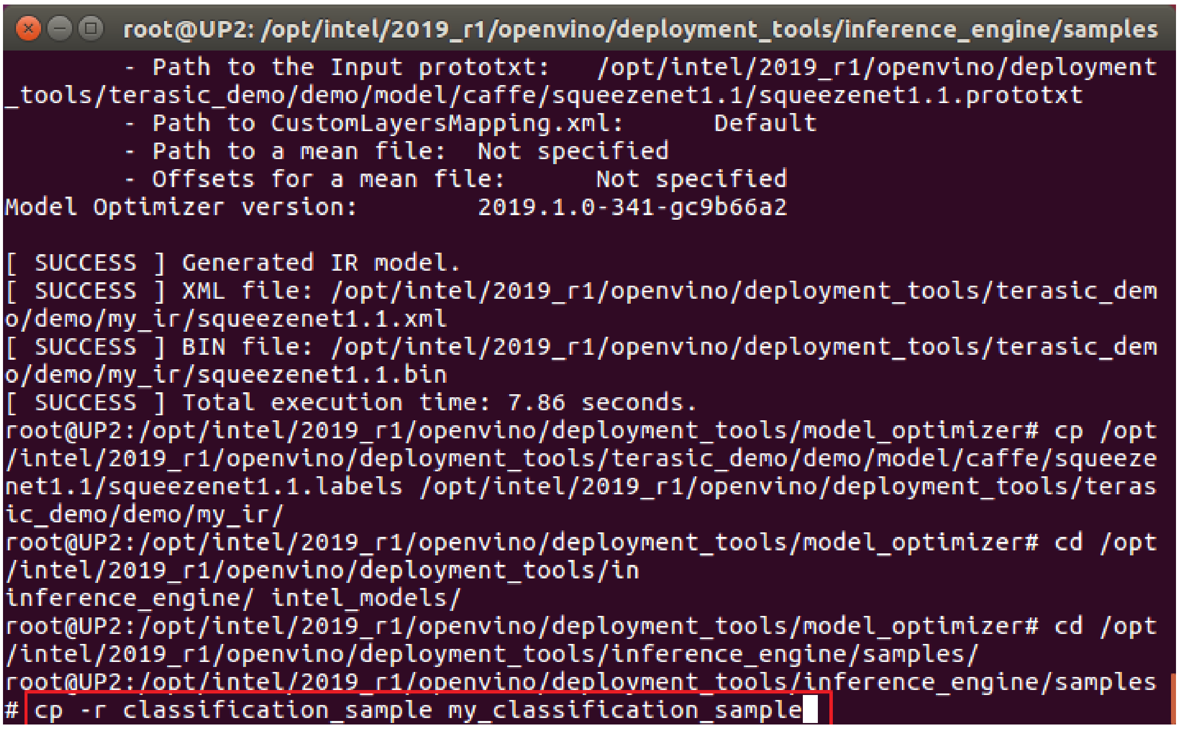
3.3 Lab 2. How to compile an Inference Engine app?

1. As the time is limited, we directly copy an existed application, rename it, and make a compilation.
2. Enter "*cd ../inference_engine/samples*" to switch the workspace to the Inference Engine samples folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/model_optimizer

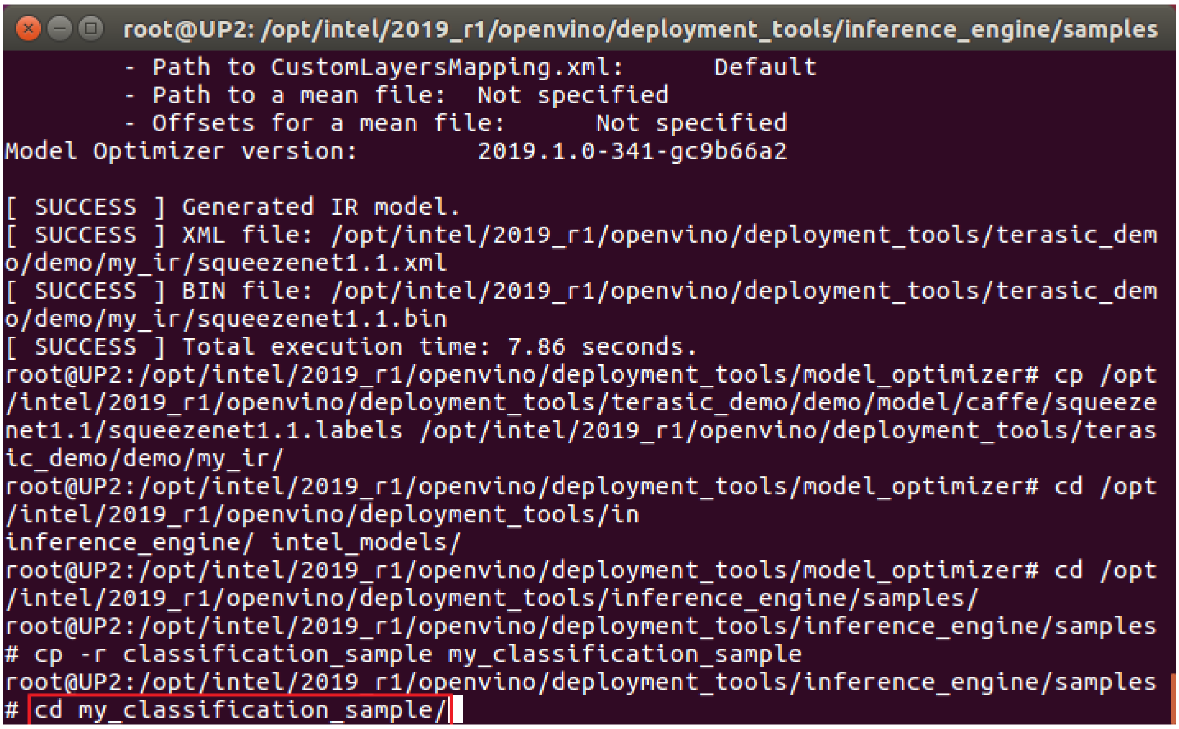
```
Caffe specific parameters:
        - Enable resnet optimization:   True
        - Path to the Input prototxt:   /opt/intel/2019_r1/openvino/deployment
_tools/terasic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.prototxt
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2019.1.0-341-gc9b66a2

[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 7.86 seconds.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cp /opt
/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/squeeze
net1.1/squeezenet1.1.labels /opt/intel/2019_r1/openvino/deployment_tools/teras
ic_demo/demo/my_ir/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/in
inference_engine/ intel_models/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/
```

3. Enter "*cp -r classification_sample my_classification_sample*".



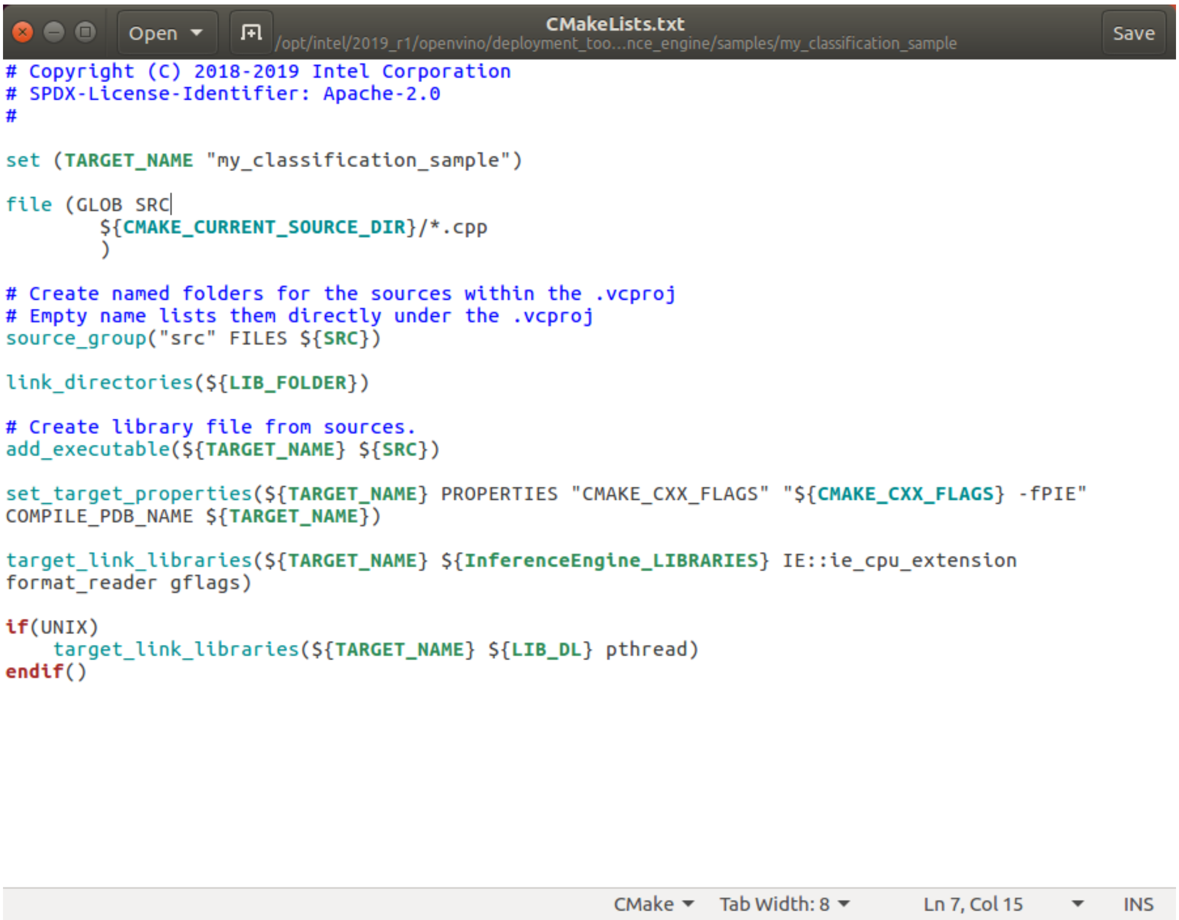
```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
        - Path to the Input prototxt:   /opt/intel/2019_r1/openvino/deployment
_tools/terasic_demo/demo/model/caffe/squeezenet1.1/squeezenet1.1.prototxt
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2019.1.0-341-gc9b66a2

[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 7.86 seconds.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cp /opt
/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/squeeze
net1.1/squeezenet1.1.labels /opt/intel/2019_r1/openvino/deployment_tools/teras
ic_demo/demo/my_ir/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/in
inference_engine/ intel_models/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cp -r classification_sample my_classification_sample
```

4. Enter “*cd my_classification_sample*” to switch the workspace to the new app folder.



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2019.1.0-341-gc9b66a2

[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/openvino/deployment_tools/terasic_dem
o/demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 7.86 seconds.
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cp /opt
/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/model/caffe/squeeze
net1.1/squeezenet1.1.labels /opt/intel/2019_r1/openvino/deployment_tools/teras
ic_demo/demo/my_ir/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/in
inference_engine/ intel_models/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/model_optimizer# cd /opt
/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cp -r classification_sample my_classification_sample
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cd my_classification_sample/
```

5. Enter “*gedit CmakeLists.txt*” to open the file, re-name the target_name to my_classification_sample, save and close the file.

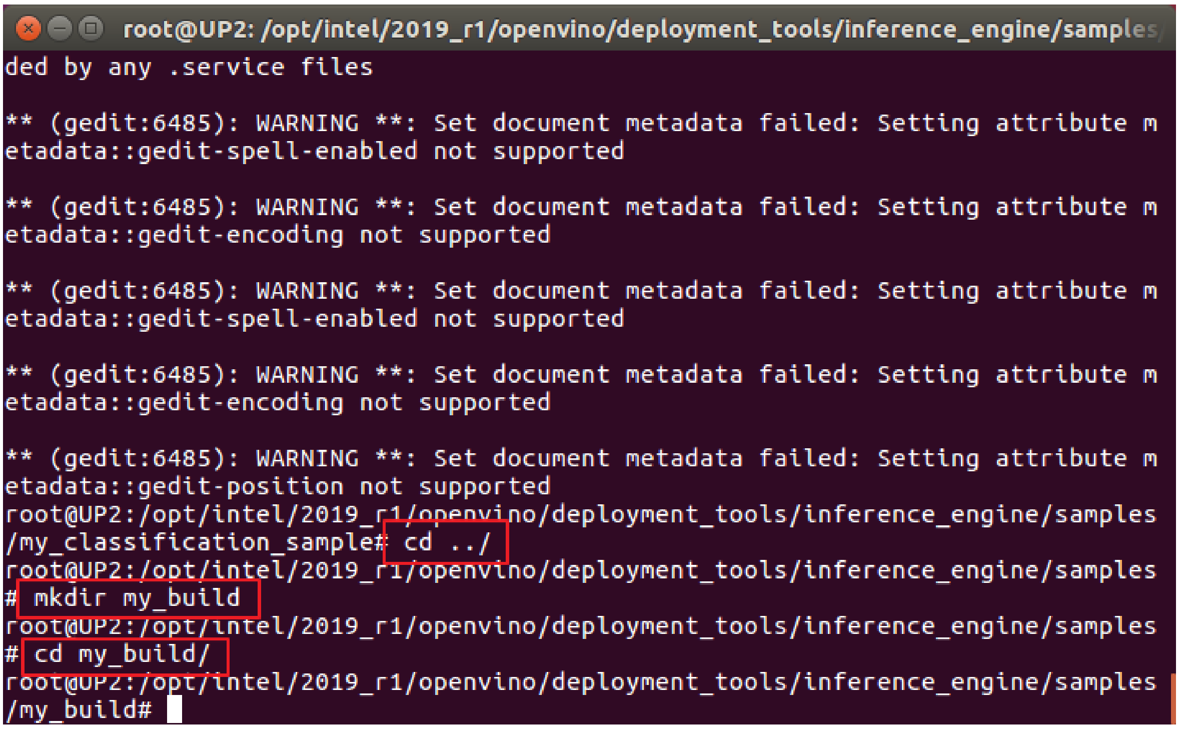

Open
CMakeLists.txt
/opt/intel/2019_r1/openvino/deployment_too...nce_engine/samples/my_classification_sample
Save
CMake
Tab Width: 8
Ln 7, Col 15
INS

```
# Copyright (C) 2018-2019 Intel Corporation
# SPDX-License-Identifier: Apache-2.0
#

set (TARGET_NAME "my_classification_sample")

file (GLOB SRC
        ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp
        )

# Create named folders for the sources within the .vcproj
# Empty name lists them directly under the .vcproj
source_group("src" FILES ${SRC})

link_directories(${LIB_FOLDER})

# Create library file from sources.
add_executable(${TARGET_NAME} ${SRC})

set_target_properties(${TARGET_NAME} PROPERTIES "CMAKE_CXX_FLAGS" "${CMAKE_CXX_FLAGS} -fPIE"
COMPILE_PDB_NAME ${TARGET_NAME})

target_link_libraries(${TARGET_NAME} ${InferenceEngine_LIBRARIES} IE::ie_cpu_extension
format_reader gflags)

if(UNIX)
    target_link_libraries(${TARGET_NAME} ${LIB_DL} pthread)
endif()
```

6. Enter “cd ../” to go back to the sample folder.

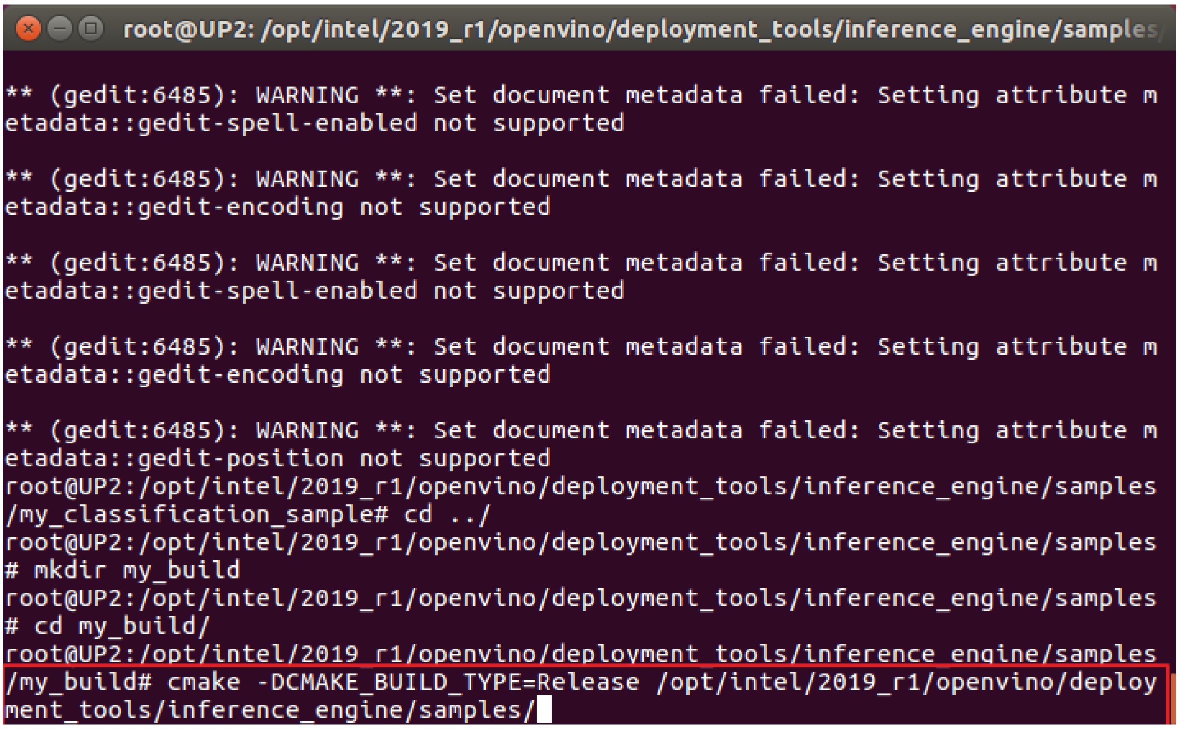
7. Enter “*mkdir my_build*” to create a new folder for saving the generated executable program.

8. Enter “*cd my_build*” to switch to the working directory.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples

```
ded by any .service files

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-spell-enabled not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-encoding not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-spell-enabled not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-encoding not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-position not supported
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_classification_sample# cd ../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# mkdir my_build
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cd my_build/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build# 
```

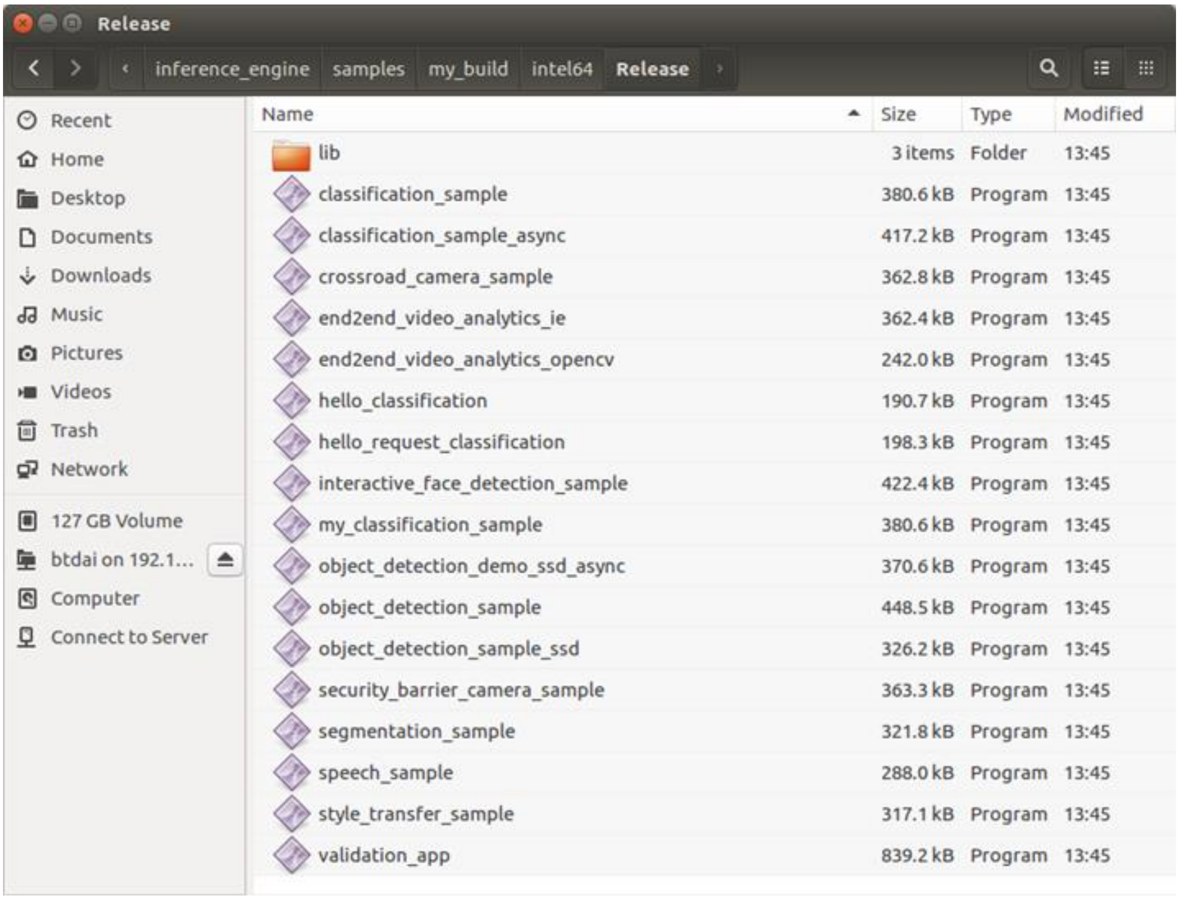
9. Enter “*cmake -DCMAKE_BUILD_TYPE=Release \ /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/*” to copy the file automatically into the directory-my_build, and it will generate a makefile for the code compilation.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples

```

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-spell-enabled not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-encoding not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-spell-enabled not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-encoding not supported

** (gedit:6485): WARNING **: Set document metadata failed: Setting attribute m
etadata::gedit-position not supported
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_classification_sample# cd ../
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# mkdir my_build
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cd my_build/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build# cmake -DCMAKE_BUILD_TYPE=Release /opt/intel/2019_r1/openvino/deploy
ment_tools/inference_engine/samples/
```

10. Enter “*make -j8*” to compile the application program, please be patient for the compilation process. Current settings of compilation will compile all the applications in samples folder to executable program.

11. Under the path: my_build/intel64/Release/, the corresponding my_classification_sample executable program is generated, and the application is created.



Name	Size	Type	Modified
lib	3 items	Folder	13:45
classification_sample	380.6 kB	Program	13:45
classification_sample_async	417.2 kB	Program	13:45
crossroad_camera_sample	362.8 kB	Program	13:45
end2end_video_analytics_ie	362.4 kB	Program	13:45
end2end_video_analytics_opencv	242.0 kB	Program	13:45
hello_classification	190.7 kB	Program	13:45
hello_request_classification	198.3 kB	Program	13:45
interactive_face_detection_sample	422.4 kB	Program	13:45
my_classification_sample	380.6 kB	Program	13:45
object_detection_demo_ssd_async	370.6 kB	Program	13:45
object_detection_sample	448.5 kB	Program	13:45
object_detection_sample_ssd	326.2 kB	Program	13:45
security_barrier_camera_sample	363.3 kB	Program	13:45
segmentation_sample	321.8 kB	Program	13:45
speech_sample	288.0 kB	Program	13:45
style_transfer_sample	317.1 kB	Program	13:45
validation_app	839.2 kB	Program	13:45

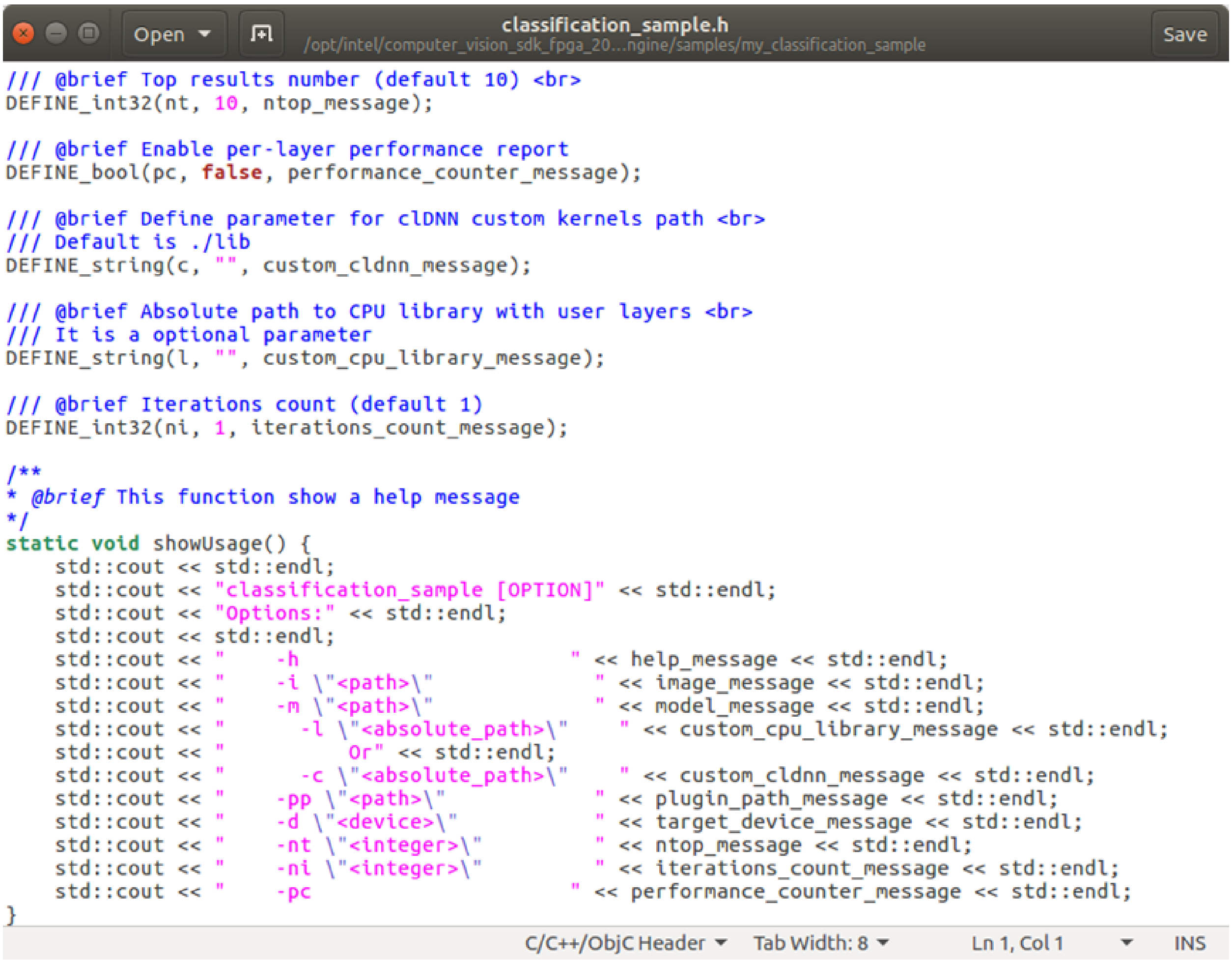
For more information of Inference Engine API, user can refer to following link:

https://docs.openvinotoolkit.org/latest/annotated.html

https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application_new_API.html

3.4 Lab 3. Execute the created application file, use the Inference Engine for the classification predication.

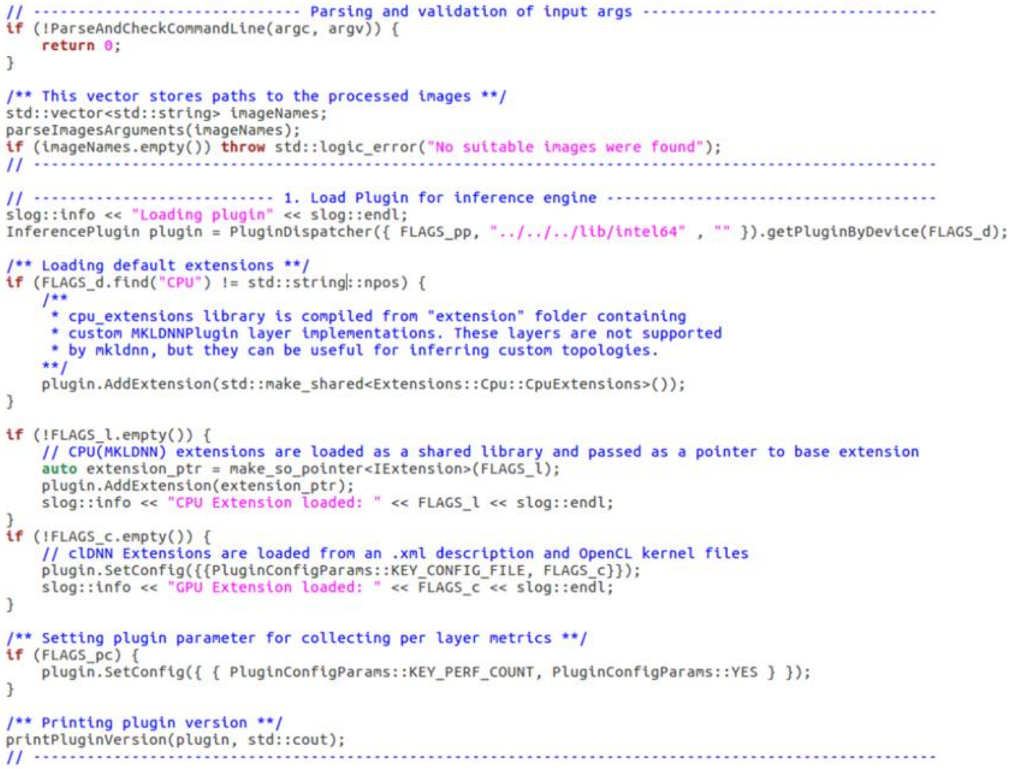
1. In the previous steps, we have transformed the model to the IR files which are used by the inference engine, and generated the corresponding executable file. Before executing the file, let’s have a general understanding of the operations in the application program.
2. Open the file: classification_sample.h, there is a parameter “showUsage” for executing the app, -h is for help, -i is for the path to a folder with images or the camera parameters, -m is for the path of a trained Model (IR Path), -d is for the target device.


Open
classification_sample.h
/opt/intel/computer_vision_sdk_fpga_20...ngine/samples/my_classification_sample
Save
C/C++/ObjC Header
Tab Width: 8
Ln 1, Col 1
INS

```
/// @brief Top results number (default 10) <br>
DEFINE_int32(nt, 10, ntop_message);

/// @brief Enable per-layer performance report
DEFINE_bool(pc, false, performance_counter_message);

/// @brief Define parameter for clDNN custom kernels path <br>
/// Default is ./lib
DEFINE_string(c, "", custom_cldnn_message);

/// @brief Absolute path to CPU library with user layers <br>
/// It is a optional parameter
DEFINE_string(l, "", custom_cpu_library_message);

/// @brief Iterations count (default 1)
DEFINE_int32(ni, 1, iterations_count_message);

/**
* @brief This function show a help message
*/
static void showUsage() {
    std::cout << std::endl;
    std::cout << "classification_sample [OPTION]" << std::endl;
    std::cout << "Options:" << std::endl;
    std::cout << std::endl;
    std::cout << "    -h                      " << help_message << std::endl;
    std::cout << "    -i \"<path>\"              " << image_message << std::endl;
    std::cout << "    -m \"<path>\"              " << model_message << std::endl;
    std::cout << "      -l \"<absolute_path>\"    " << custom_cpu_library_message << std::endl;
    std::cout << "          Or" << std::endl;
    std::cout << "      -c \"<absolute_path>\"    " << custom_cldnn_message << std::endl;
    std::cout << "    -pp \"<path>\"             " << plugin_path_message << std::endl;
    std::cout << "    -d \"<device>\"            " << target_device_message << std::endl;
    std::cout << "    -nt \"<integer>\"          " << ntop_message << std::endl;
    std::cout << "    -ni \"<integer>\"          " << iterations_count_message << std::endl;
    std::cout << "    -pc                     " << performance_counter_message << std::endl;
}
```

3. Open the main.cpp in the folder: inference_engine/samples/my_classification_sample, there are explanations:

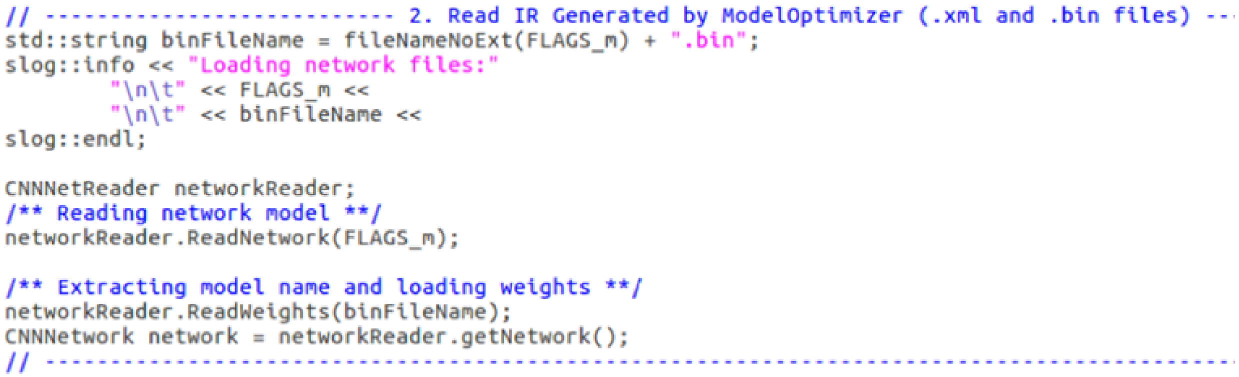
   ➢ Step1, Load Plugin for inference engine, for this lab, the plugin is hetero plugin for FPGA and CPU.



```
// ----------------------------- Parsing and validation of input args ---------------------------------
if (!ParseAndCheckCommandLine(argc, argv)) {
    return 0;
}

/** This vector stores paths to the processed images **/
std::vector<std::string> imageNames;
parseImagesArguments(imageNames);
if (imageNames.empty()) throw std::logic_error("No suitable images were found");
// -----------------------------------------------------------------------------------------------------

// --------------------------- 1. Load Plugin for inference engine -------------------------------------
slog::info << "Loading plugin" << slog::endl;
InferencePlugin plugin = PluginDispatcher({ FLAGS_pp, "../../../lib/intel64" , "" }).getPluginByDevice(FLAGS_d);

/** Loading default extensions **/
if (FLAGS_d.find("CPU") != std::string::npos) {
    /**
     * cpu_extensions library is compiled from "extension" folder containing
     * custom MKLDNNPlugin layer implementations. These layers are not supported
     * by mkldnn, but they can be useful for inferring custom topologies.
    **/
    plugin.AddExtension(std::make_shared<Extensions::Cpu::CpuExtensions>());
}

if (!FLAGS_l.empty()) {
    // CPU(MKLDNN) extensions are loaded as a shared library and passed as a pointer to base extension
    auto extension_ptr = make_so_pointer<IExtension>(FLAGS_l);
    plugin.AddExtension(extension_ptr);
    slog::info << "CPU Extension loaded: " << FLAGS_l << slog::endl;
}
if (!FLAGS_c.empty()) {
    // clDNN Extensions are loaded from an .xml description and OpenCL kernel files
    plugin.SetConfig({{PluginConfigParams::KEY_CONFIG_FILE, FLAGS_c}});
    slog::info << "GPU Extension loaded: " << FLAGS_c << slog::endl;
}

/** Setting plugin parameter for collecting per layer metrics **/
if (FLAGS_pc) {
    plugin.SetConfig({ { PluginConfigParams::KEY_PERF_COUNT, PluginConfigParams::YES } });
}

/** Printing plugin version **/
printPluginVersion(plugin, std::cout);
// -----------------------------------------------------------------------------------------------------
```

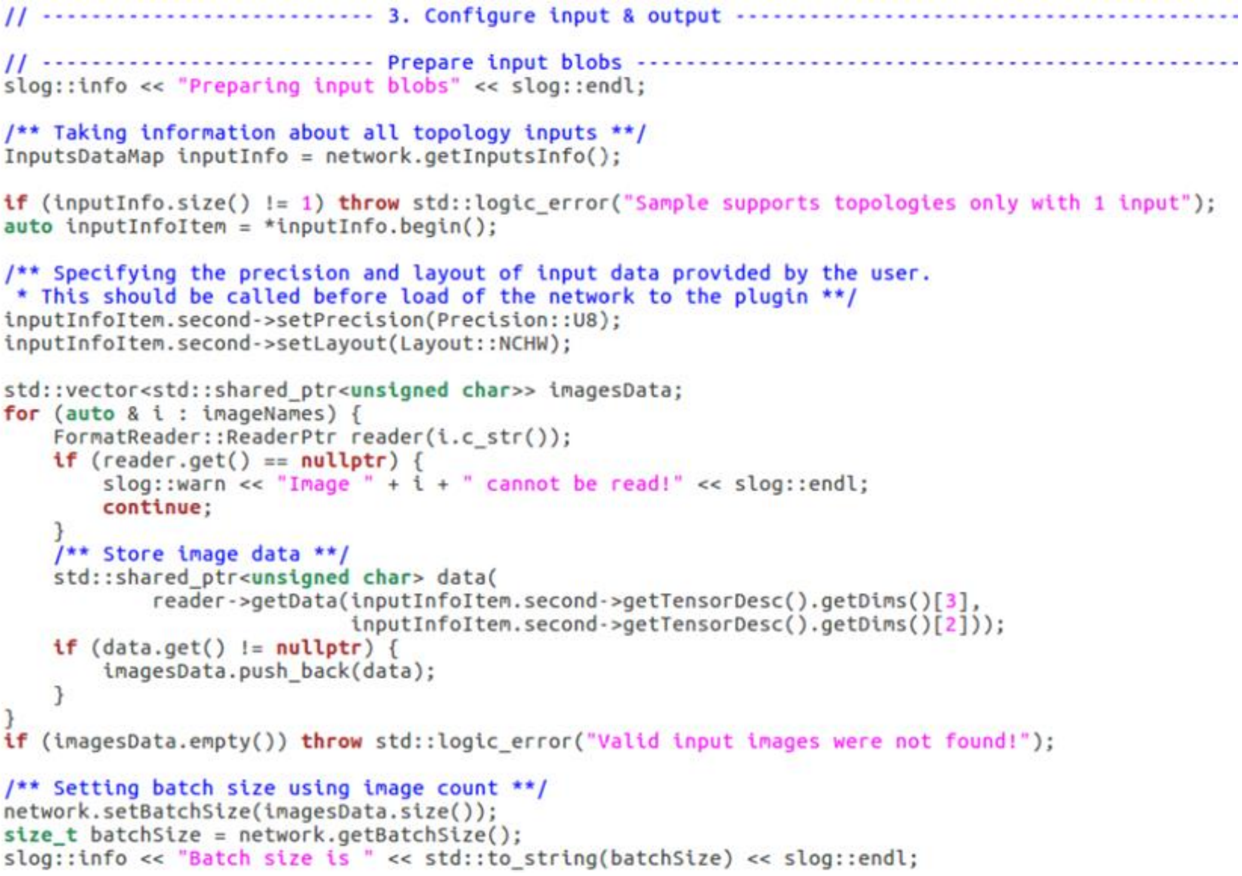
- Step2, Read the IR Generated by Model Optimizer (.xml and .bin files).

  For this lab, the xml file is squeezenet1.1.xml.



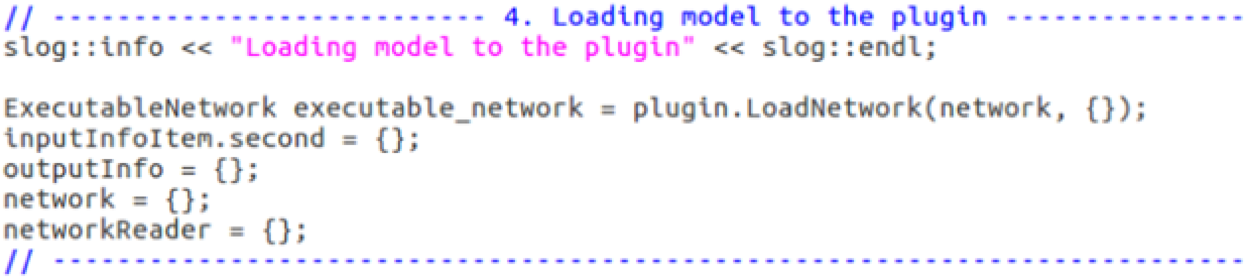
```
// --------------------------- 2. Read IR Generated by ModelOptimizer (.xml and .bin files) --
std::string binFileName = fileNameNoExt(FLAGS_m) + ".bin";
slog::info << "Loading network files:"
        "\n\t" << FLAGS_m <<
        "\n\t" << binFileName <<
slog::endl;

CNNNetReader networkReader;
/** Reading network model **/
networkReader.ReadNetwork(FLAGS_m);

/** Extracting model name and loading weights **/
networkReader.ReadWeights(binFileName);
CNNNetwork network = networkReader.getNetwork();
// -----------------------------------------------------------------------------------------
```

- Step3, configure the input & output, prepare the input blobs, read the input size information, read the images path, set batch size, prepare the output blobs.



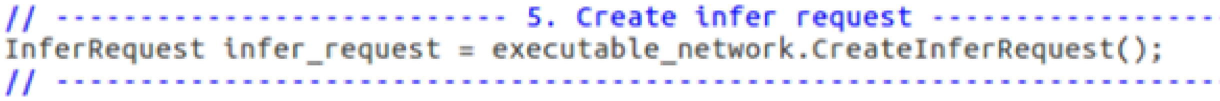
```
// --------------------------- 3. Configure input & output ---------------------------------------------

// --------------------------- Prepare input blobs -----------------------------------------------------
slog::info << "Preparing input blobs" << slog::endl;

/** Taking information about all topology inputs **/
InputsDataMap inputInfo = network.getInputsInfo();

if (inputInfo.size() != 1) throw std::logic_error("Sample supports topologies only with 1 input");
auto inputInfoItem = *inputInfo.begin();

/** Specifying the precision and layout of input data provided by the user.
 * This should be called before load of the network to the plugin **/
inputInfoItem.second->setPrecision(Precision::U8);
inputInfoItem.second->setLayout(Layout::NCHW);

std::vector<std::shared_ptr<unsigned char>> imagesData;
for (auto & i : imageNames) {
    FormatReader::ReaderPtr reader(i.c_str());
    if (reader.get() == nullptr) {
        slog::warn << "Image " + i + " cannot be read!" << slog::endl;
        continue;
    }
    /** Store image data **/
    std::shared_ptr<unsigned char> data(
            reader->getData(inputInfoItem.second->getTensorDesc().getDims()[3],
                            inputInfoItem.second->getTensorDesc().getDims()[2]));
    if (data.get() != nullptr) {
        imagesData.push_back(data);
    }
}
if (imagesData.empty()) throw std::logic_error("Valid input images were not found!");

/** Setting batch size using image count **/
network.setBatchSize(imagesData.size());
size_t batchSize = network.getBatchSize();
slog::info << "Batch size is " << std::to_string(batchSize) << slog::endl;
```

- ➢ Step4, Loading model to the plugin.



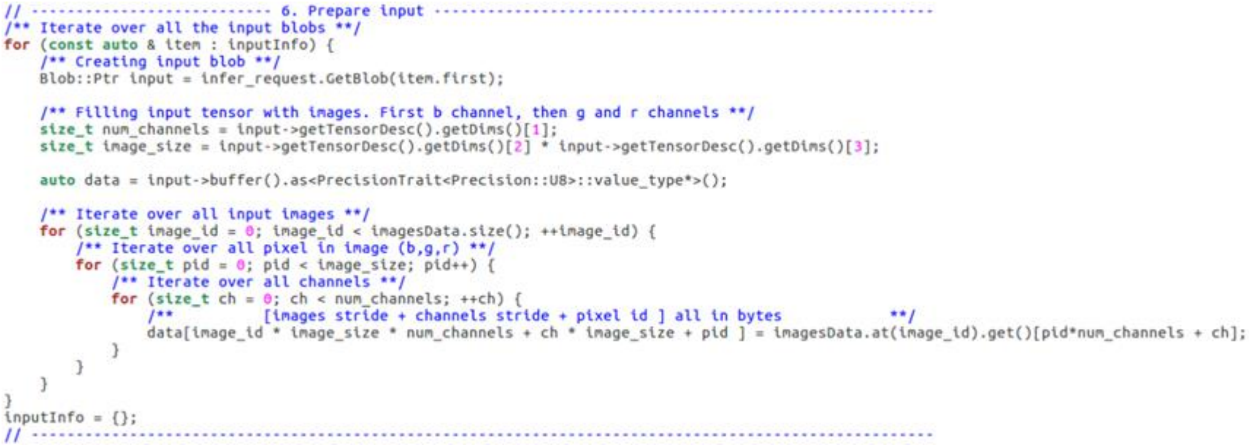
```
// --------------------------- 4. Loading model to the plugin ---------------
slog::info << "Loading model to the plugin" << slog::endl;

ExecutableNetwork executable_network = plugin.LoadNetwork(network, {});
inputInfoItem.second = {};
outputInfo = {};
network = {};
networkReader = {};
// -------------------------------------------------------------------------
```

- ➢ Step5, create infer request.



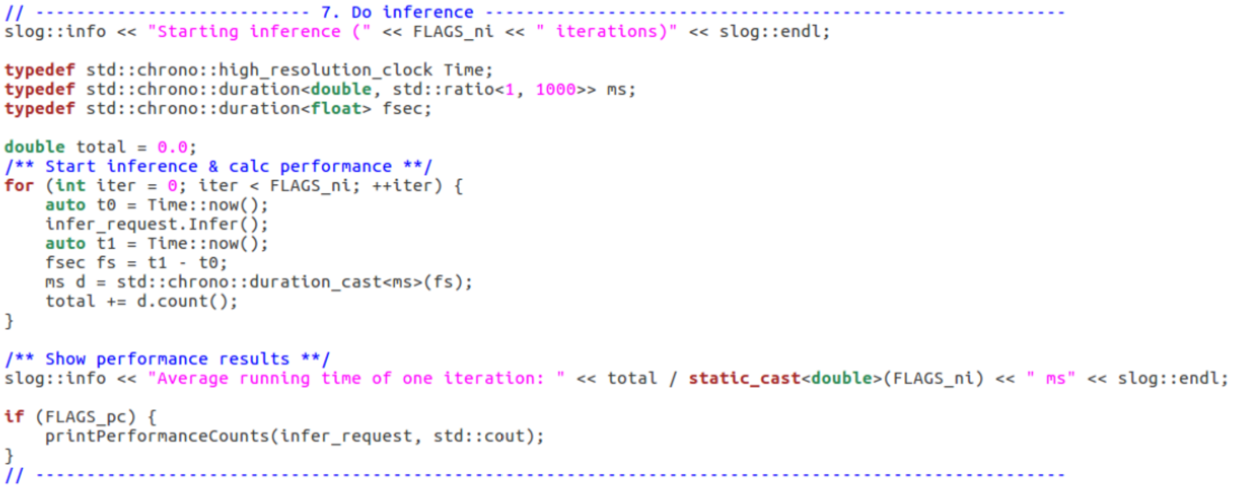
```
// --------------------------- 5. Create infer request -----------------
InferRequest infer_request = executable_network.CreateInferRequest();
// ---------------------------------------------------------------------
```

- ➢ Step6, Prepare input.



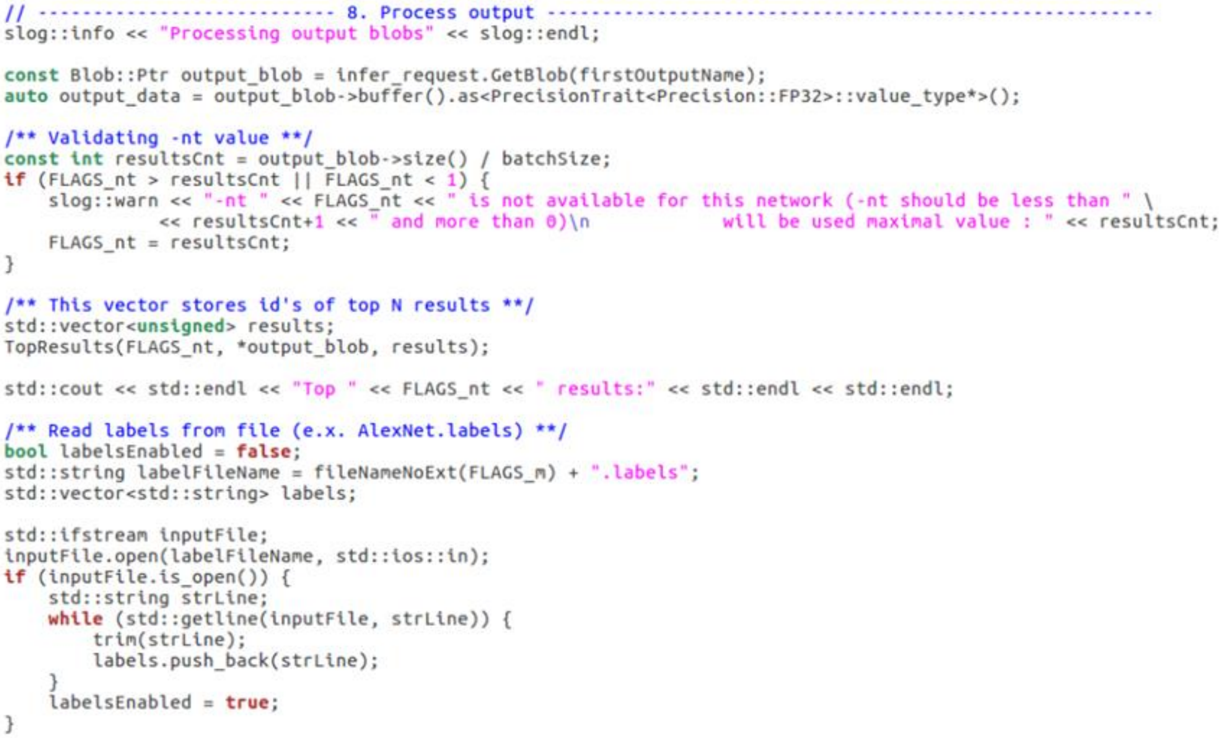
```
// --------------------------- 6. Prepare input --------------------------------------------------------
/** Iterate over all the input blobs **/
for (const auto & item : inputInfo) {
    /** Creating input blob **/
    Blob::Ptr input = infer_request.GetBlob(item.first);

    /** Filling input tensor with images. First b channel, then g and r channels **/
    size_t num_channels = input->getTensorDesc().getDims()[1];
    size_t image_size = input->getTensorDesc().getDims()[2] * input->getTensorDesc().getDims()[3];

    auto data = input->buffer().as<PrecisionTrait<Precision::U8>::value_type*>();

    /** Iterate over all input images **/
    for (size_t image_id = 0; image_id < imagesData.size(); ++image_id) {
        /** Iterate over all pixel in image (b,g,r) **/
        for (size_t pid = 0; pid < image_size; pid++) {
            /** Iterate over all channels **/
            for (size_t ch = 0; ch < num_channels; ++ch) {
                /**          [images stride + channels stride + pixel id ] all in bytes            **/
                data[image_id * image_size * num_channels + ch * image_size + pid ] = imagesData.at(image_id).get()[pid*num_channels + ch];
            }
        }
    }
}
inputInfo = {};
// -----------------------------------------------------------------------------------------------------
```

- Step7, Do the inference, send data to FPGA for processing, and send the result to CPU.



```
// --------------------------- 7. Do inference ---------------------------------------------------------
slog::info << "Starting inference (" << FLAGS_ni << " iterations)" << slog::endl;

typedef std::chrono::high_resolution_clock Time;
typedef std::chrono::duration<double, std::ratio<1, 1000>> ms;
typedef std::chrono::duration<float> fsec;

double total = 0.0;
/** Start inference & calc performance **/
for (int iter = 0; iter < FLAGS_ni; ++iter) {
    auto t0 = Time::now();
    infer_request.Infer();
    auto t1 = Time::now();
    fsec fs = t1 - t0;
    ms d = std::chrono::duration_cast<ms>(fs);
    total += d.count();
}

/** Show performance results **/
slog::info << "Average running time of one iteration: " << total / static_cast<double>(FLAGS_ni) << " ms" << slog::endl;

if (FLAGS_pc) {
    printPerformanceCounts(infer_request, std::cout);
}
// -----------------------------------------------------------------------------------------------------
```

- Step8, Process output, process the result and compare with the label file, the last printf the result and the average inference time.



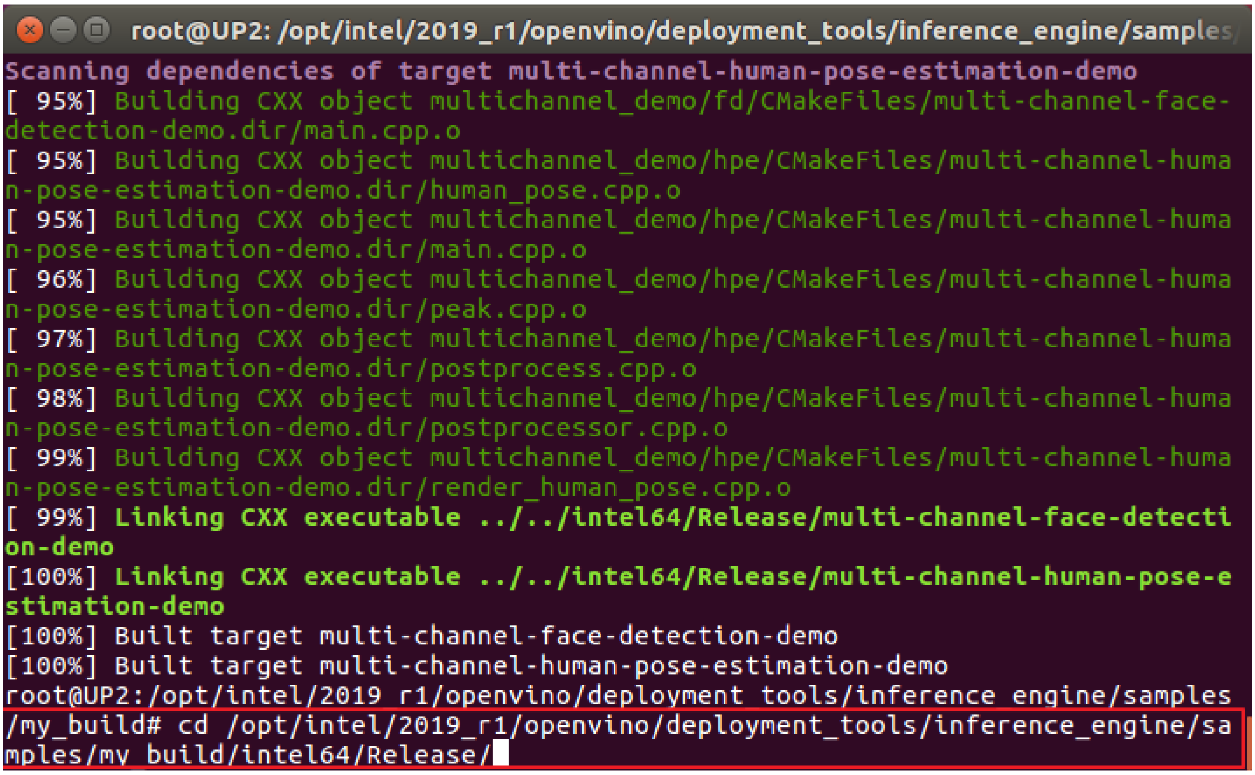
```
// --------------------------- 8. Process output -------------------------------------------------------
slog::info << "Processing output blobs" << slog::endl;

const Blob::Ptr output_blob = infer_request.GetBlob(firstOutputName);
auto output_data = output_blob->buffer().as<PrecisionTrait<Precision::FP32>::value_type*>();

/** Validating -nt value **/
const int resultsCnt = output_blob->size() / batchSize;
if (FLAGS_nt > resultsCnt || FLAGS_nt < 1) {
    slog::warn << "-nt " << FLAGS_nt << " is not available for this network (-nt should be less than " \
               << resultsCnt+1 << " and more than 0)\n            will be used maximal value : " << resultsCnt;
    FLAGS_nt = resultsCnt;
}

/** This vector stores id's of top N results **/
std::vector<unsigned> results;
TopResults(FLAGS_nt, *output_blob, results);

std::cout << std::endl << "Top " << FLAGS_nt << " results:" << std::endl << std::endl;

/** Read labels from file (e.x. AlexNet.labels) **/
bool labelsEnabled = false;
std::string labelFileName = fileNameNoExt(FLAGS_m) + ".labels";
std::vector<std::string> labels;

std::ifstream inputFile;
inputFile.open(labelFileName, std::ios::in);
if (inputFile.is_open()) {
    std::string strLine;
    while (std::getline(inputFile, strLine)) {
        trim(strLine);
        labels.push_back(strLine);
    }
    labelsEnabled = true;
}
```

4. Now, we are clear about the operations executed in the application, next, let us run the executable file we generated before.

5. Enter "*cd \*

   /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples\

   /my_build/intel64/Release" to switch the directory to application folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples

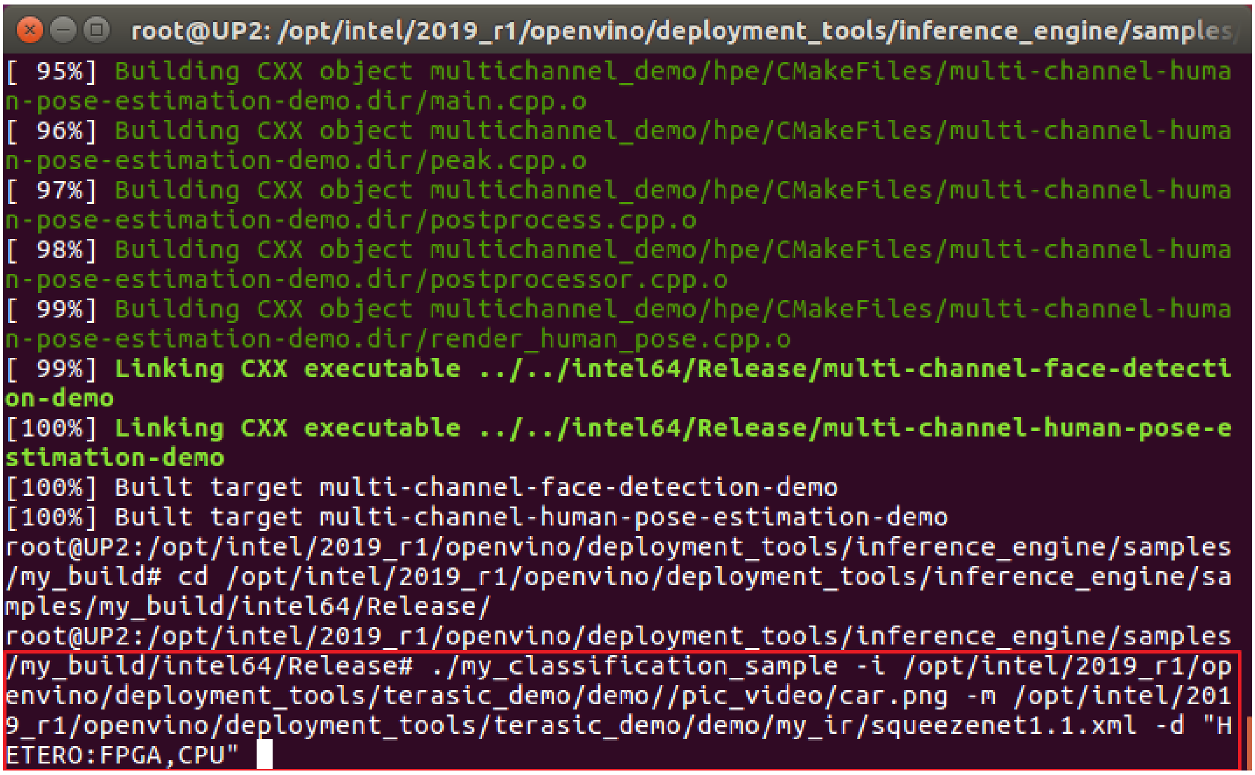
```
Scanning dependencies of target multi-channel-human-pose-estimation-demo
[ 95%] Building CXX object multichannel_demo/fd/CMakeFiles/multi-channel-face-
detection-demo.dir/main.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/human_pose.cpp.o
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/main.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/peak.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocess.cpp.o
[ 98%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocessor.cpp.o
[ 99%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/render_human_pose.cpp.o
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-face-detecti
on-demo
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-e
stimation-demo
[100%] Built target multi-channel-face-detection-demo
[100%] Built target multi-channel-human-pose-estimation-demo
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build# cd /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/sa
mples/my_build/intel64/Release/
```

6. Enter "*./my_classification_sample -i \*

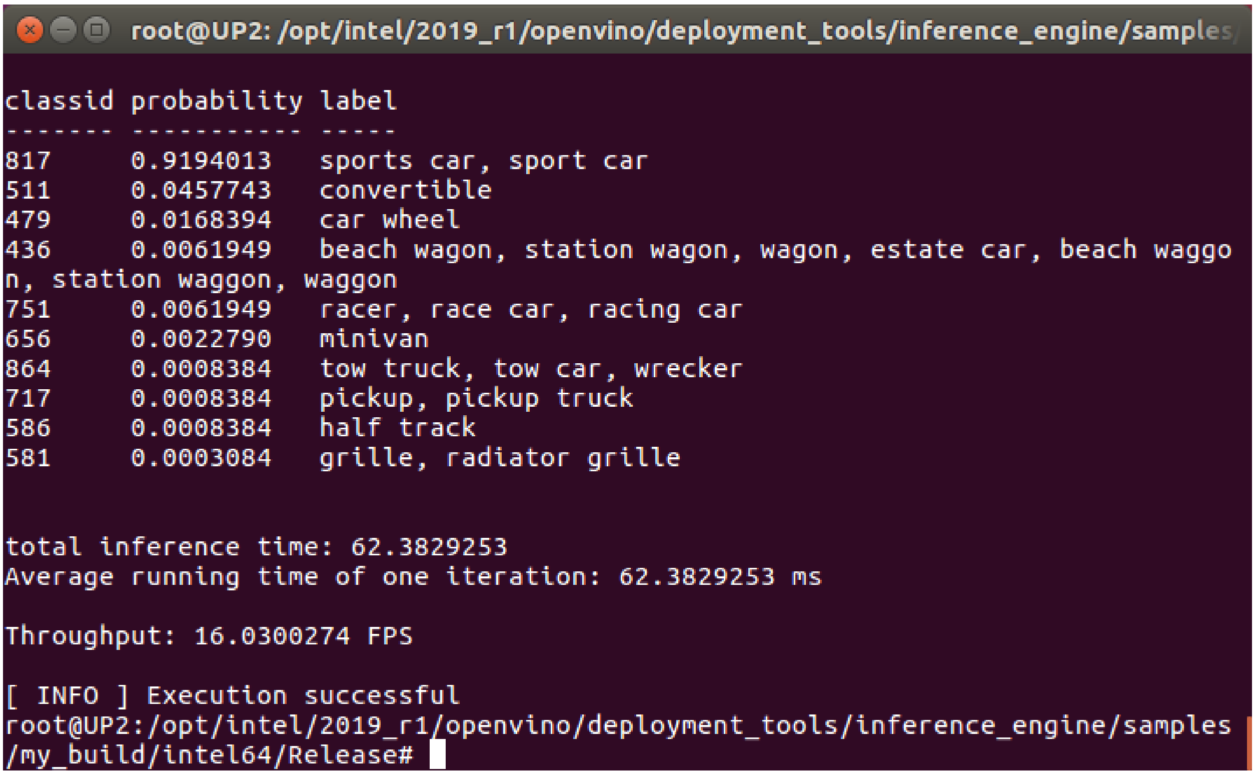
   /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/pic_video/car.png \

   -m /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/\

   demo/my_ir/squeezenet1.1.xml -d "HETERO:FPGA,CPU" " to execute the Inference Engine.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples

```
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/main.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/peak.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocess.cpp.o
[ 98%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/postprocessor.cpp.o
[ 99%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-huma
n-pose-estimation-demo.dir/render_human_pose.cpp.o
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-face-detecti
on-demo
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-e
stimation-demo
[100%] Built target multi-channel-face-detection-demo
[100%] Built target multi-channel-human-pose-estimation-demo
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build# cd /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/sa
mples/my_build/intel64/Release/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build/intel64/Release# ./my_classification_sample -i /opt/intel/2019_r1/op
envino/deployment_tools/terasic_demo/demo//pic_video/car.png -m /opt/intel/201
9_r1/openvino/deployment_tools/terasic_demo/demo/my_ir/squeezenet1.1.xml -d "H
ETERO:FPGA,CPU"
```


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples

```
classid probability label
------- ----------- -----
817     0.9194013   sports car, sport car
511     0.0457743   convertible
479     0.0168394   car wheel
436     0.0061949   beach wagon, station wagon, wagon, estate car, beach waggo
n, station waggon, waggon
751     0.0061949   racer, race car, racing car
656     0.0022790   minivan
864     0.0008384   tow truck, tow car, wrecker
717     0.0008384   pickup, pickup truck
586     0.0008384   half track
581     0.0003084   grille, radiator grille


total inference time: 62.3829253
Average running time of one iteration: 62.3829253 ms

Throughput: 16.0300274 FPS

[ INFO ] Execution successful
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build/intel64/Release#
```

3.5 Advanced Experiment

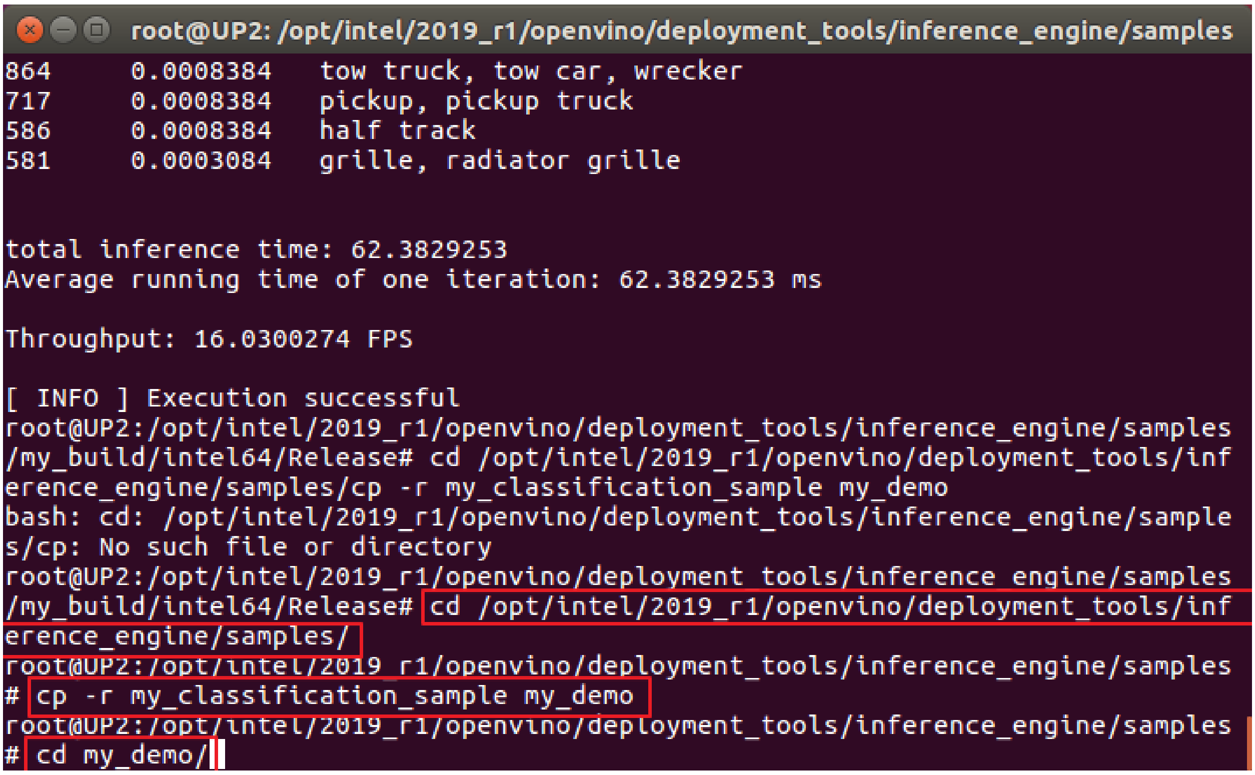
1. In the previous steps, we already to know how to convert the model to IR, and how to make executable files which can be used by the inference engine. Next, let’s make a new demo.

2. Since the IR files have been generated already, no need to regenerate it, we continue to use the previously generated squeezenet1.1.xml and the corresponding bin file.

3. Enter “*cd /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples*” to switch the workspace to Inference Engine samples folder.

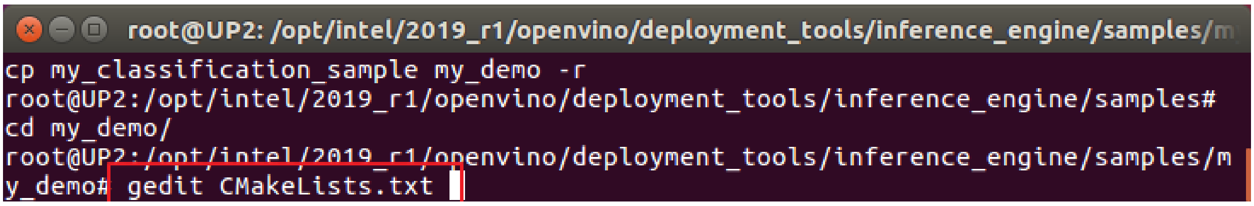
4. Enter "*cp -r my_classification_sample my_demo*" to copy the files generated in the previous experiment, and we will modify them to be used in my_demo.

5. Enter "*cd my_demo*" to switch to the new copied samples folder.

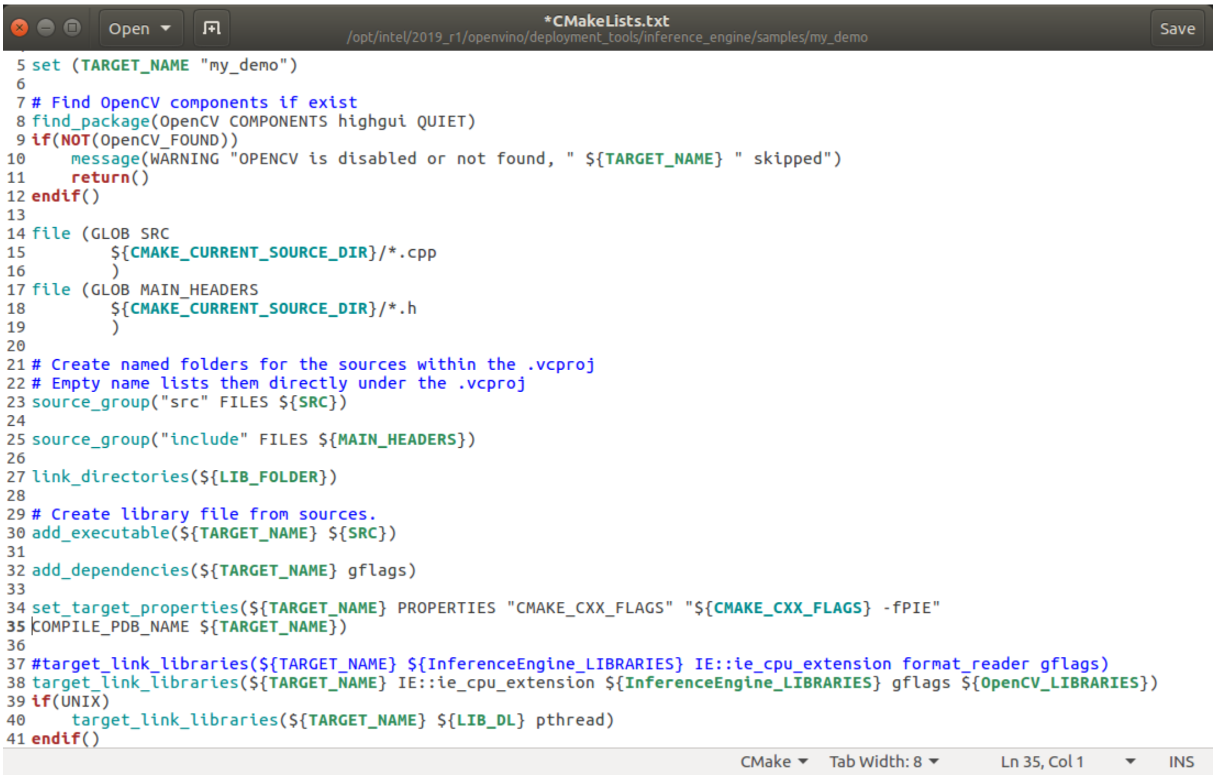


```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
864     0.0008384   tow truck, tow car, wrecker
717     0.0008384   pickup, pickup truck
586     0.0008384   half track
581     0.0003084   grille, radiator grille


total inference time: 62.3829253
Average running time of one iteration: 62.3829253 ms

Throughput: 16.0300274 FPS

[ INFO ] Execution successful
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build/intel64/Release# cd /opt/intel/2019_r1/openvino/deployment_tools/inf
erence_engine/samples/cp -r my_classification_sample my_demo
bash: cd: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/sample
s/cp: No such file or directory
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
/my_build/intel64/Release# cd /opt/intel/2019_r1/openvino/deployment_tools/inf
erence_engine/samples/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cp -r my_classification_sample my_demo
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
# cd my_demo/
```

6. Enter "*gedit CmakeLists.txt*" to open the file. Modify the file as follow:



```
root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m
cp my_classification_sample my_demo -r
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples#
cd my_demo/
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m
y_demo# gedit CMakeLists.txt
```


Open
*CMakeLists.txt
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
CMake
Tab Width: 8
Ln 35, Col 1
INS

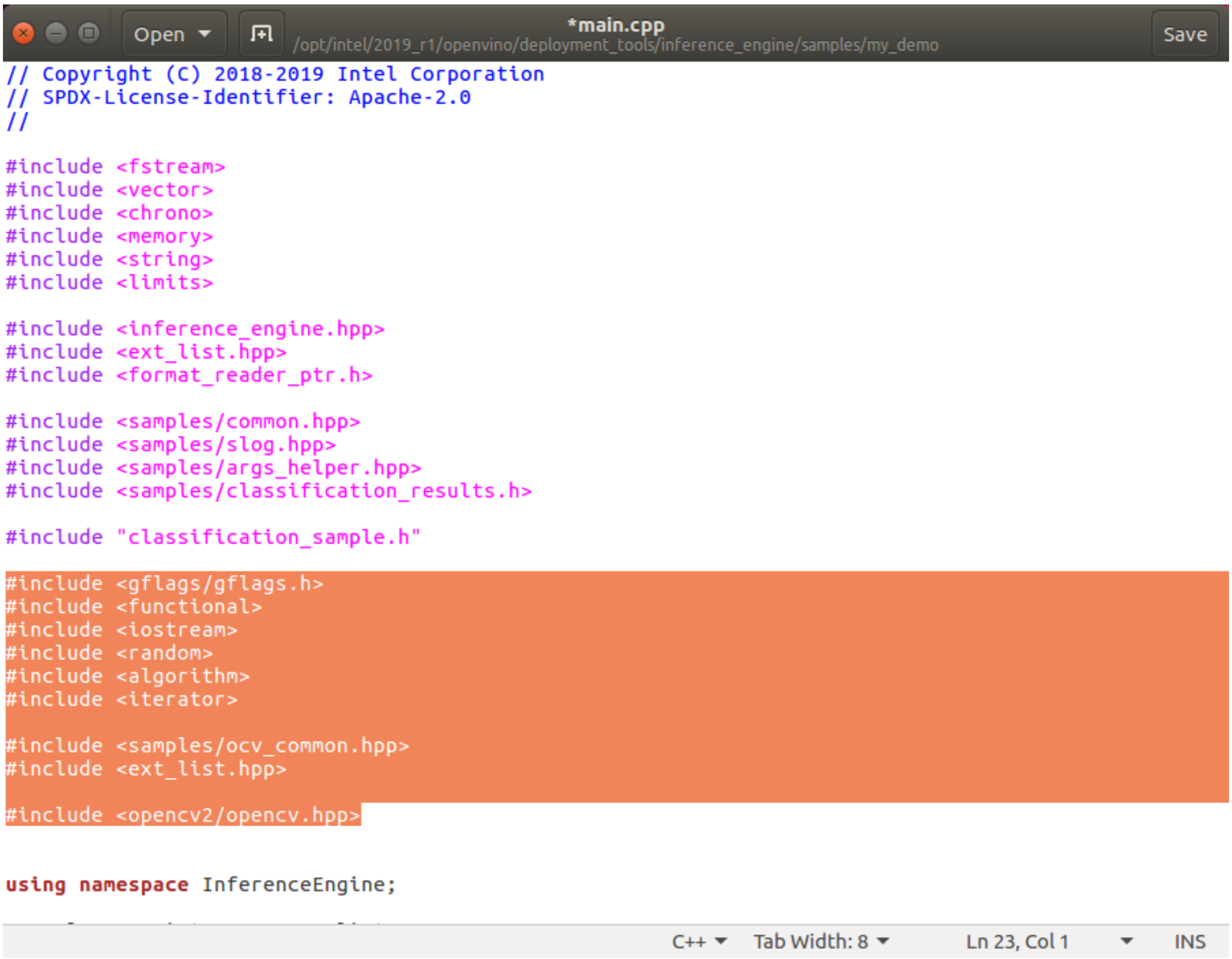
```
set (TARGET_NAME "my_demo")

# Find OpenCV components if exist
find_package(OpenCV COMPONENTS highgui QUIET)
if(NOT(OpenCV_FOUND))
    message(WARNING "OPENCV is disabled or not found, " ${TARGET_NAME} " skipped")
    return()
endif()

file (GLOB SRC
        ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp
        )
file (GLOB MAIN_HEADERS
        ${CMAKE_CURRENT_SOURCE_DIR}/*.h
        )

# Create named folders for the sources within the .vcproj
# Empty name lists them directly under the .vcproj
source_group("src" FILES ${SRC})

source_group("include" FILES ${MAIN_HEADERS})

link_directories(${LIB_FOLDER})

# Create library file from sources.
add_executable(${TARGET_NAME} ${SRC})

add_dependencies(${TARGET_NAME} gflags)

set_target_properties(${TARGET_NAME} PROPERTIES "CMAKE_CXX_FLAGS" "${CMAKE_CXX_FLAGS} -fPIE"
COMPILE_PDB_NAME ${TARGET_NAME})

#target_link_libraries(${TARGET_NAME} ${InferenceEngine_LIBRARIES} IE::ie_cpu_extension format_reader gflags)
target_link_libraries(${TARGET_NAME} IE::ie_cpu_extension ${InferenceEngine_LIBRARIES} gflags ${OpenCV_LIBRARIES})
if(UNIX)
    target_link_libraries(${TARGET_NAME} ${LIB_DL} pthread)
endif()
```

```
set (TARGET_NAME “my_demo”)


# Find OpenCV components if exist
find_package(OpenCV COMPONENTS highgui QUIET)
if(NOT(OpenCV_FOUND))
    message(WARNING “OPENCV is disabled or not found,
“ ${TARGET_NAME} “ skipped”)
    return()
endif()


file (GLOB SRC
          ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp
          )
file (GLOB MAIN_HEADERS
          ${CMAKE_CURRENT_SOURCE_DIR}/*.h
          )


# Create named folders for the sources within the .vcproj
# Empty name lists them directly under the .vcproj
source_group(“src” FILES ${SRC})


source_group(“include” FILES ${MAIN_HEADERS})
```

```
link_directories(${LIB_FOLDER})

# Create library file from sources.
Add_executable(${TARGET_NAME} ${SRC})

add_dependencies(${TARGET_NAME} gflags)

set_target_properties(${TARGET_NAME} PROPERTIES
"CMAKE_CXX_FLAGS" "${CMAKE_CXX_FLAGS} -fPIE"
COMPILE_PDB_NAME ${TARGET_NAME})

#target_link_libraries(${TARGET_NAME} ${InferenceEngine_LIBRARIES}
IE::ie_cpu_extension #format_reader gflags)
target_link_libraries(${TARGET_NAME} IE::ie_cpu_extension
${InferenceEngine_LIBRARIES} gflags ${OpenCV_LIBRARIES})
if(UNIX)
    target_link_libraries(${TARGET_NAME} ${LIB_DL} pthread)
endif()
```

7. Enter "*gedit main.cpp*" to open the file for application modification. The host application generated in the previous steps is for the classification of entering a single picture. Next, we will modify the application to support the classification display on multiply picture entering for loop.

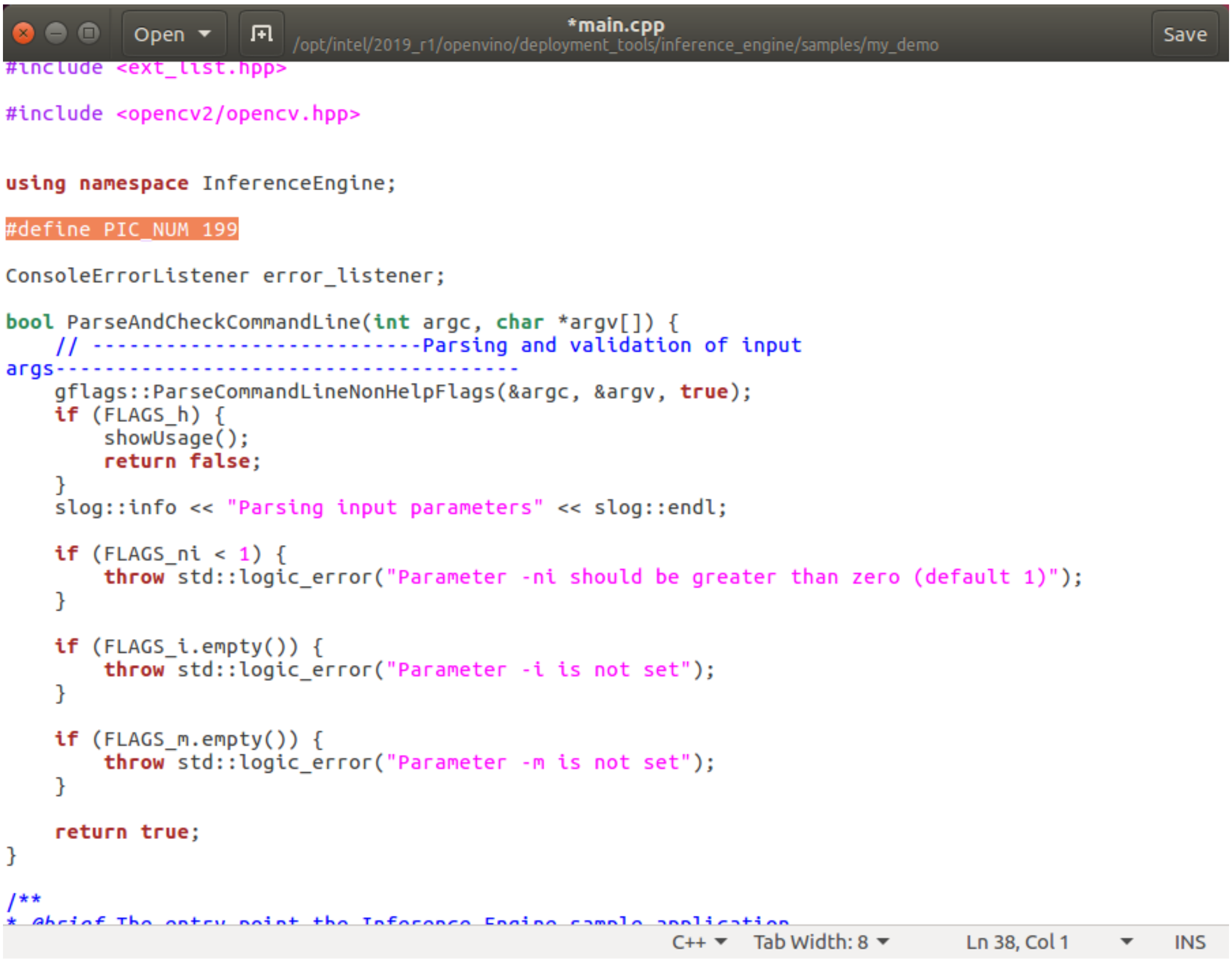
- Setp1, add a header file for opencv and video operation.


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 23, Col 1
INS

```
// Copyright (C) 2018-2019 Intel Corporation
// SPDX-License-Identifier: Apache-2.0
//

#include <fstream>
#include <vector>
#include <chrono>
#include <memory>
#include <string>
#include <limits>

#include <inference_engine.hpp>
#include <ext_list.hpp>
#include <format_reader_ptr.h>

#include <samples/common.hpp>
#include <samples/slog.hpp>
#include <samples/args_helper.hpp>
#include <samples/classification_results.h>

#include "classification_sample.h"

#include <gflags/gflags.h>
#include <functional>
#include <iostream>
#include <random>
#include <algorithm>
#include <iterator>

#include <samples/ocv_common.hpp>
#include <ext_list.hpp>

#include <opencv2/opencv.hpp>


using namespace InferenceEngine;
```

```
#include <gflags/gflags.h>
#include <functional>
#include <iostream>
#include <random>
#include <algorithm>
#include <iterator>


#include <samples/ocv_common.hpp>
#include <ext_list.hpp>


#include <opencv2/opencv.hpp>
```

- Step2, define a macro for picture number.

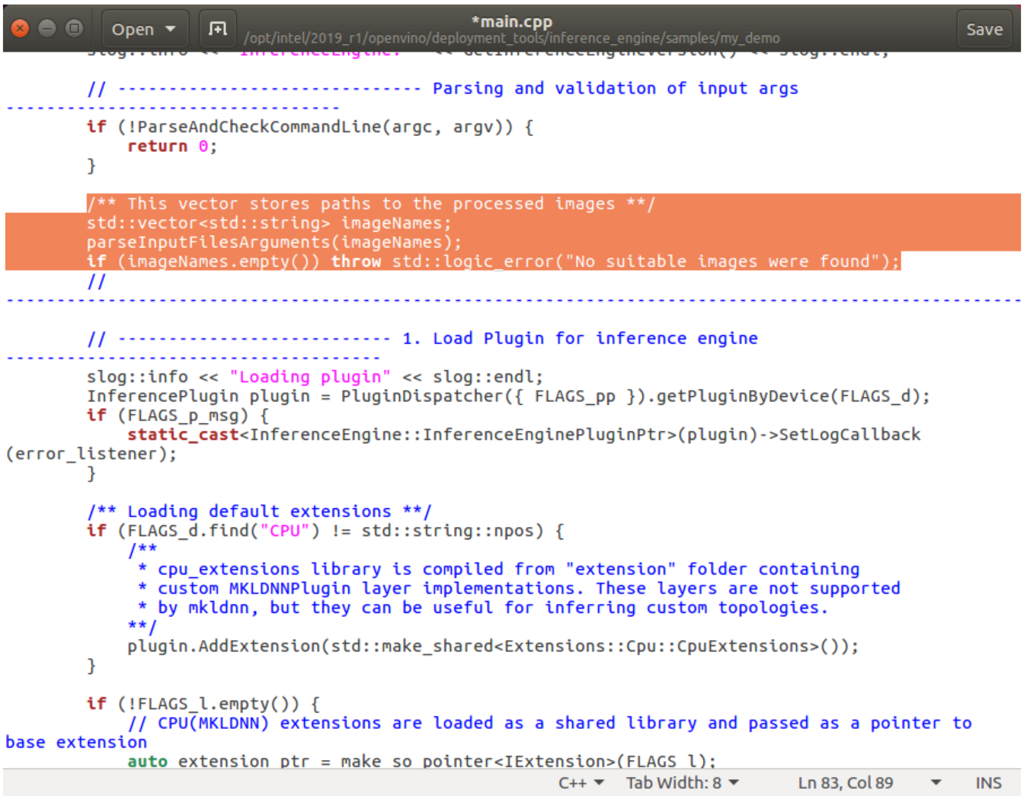


*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Open
Save
C++
Tab Width: 8
Ln 38, Col 1
INS

```
#include <ext_list.hpp>

#include <opencv2/opencv.hpp>


using namespace InferenceEngine;

#define PIC_NUM 199

ConsoleErrorListener error_listener;

bool ParseAndCheckCommandLine(int argc, char *argv[]) {
    // ---------------------------Parsing and validation of input
args--------------------------------------
    gflags::ParseCommandLineNonHelpFlags(&argc, &argv, true);
    if (FLAGS_h) {
        showUsage();
        return false;
    }
    slog::info << "Parsing input parameters" << slog::endl;

    if (FLAGS_ni < 1) {
        throw std::logic_error("Parameter -ni should be greater than zero (default 1)");
    }

    if (FLAGS_i.empty()) {
        throw std::logic_error("Parameter -i is not set");
    }

    if (FLAGS_m.empty()) {
        throw std::logic_error("Parameter -m is not set");
    }

    return true;
}

/**
```

- Step3, delete the imageNames, because we don't use it anymore

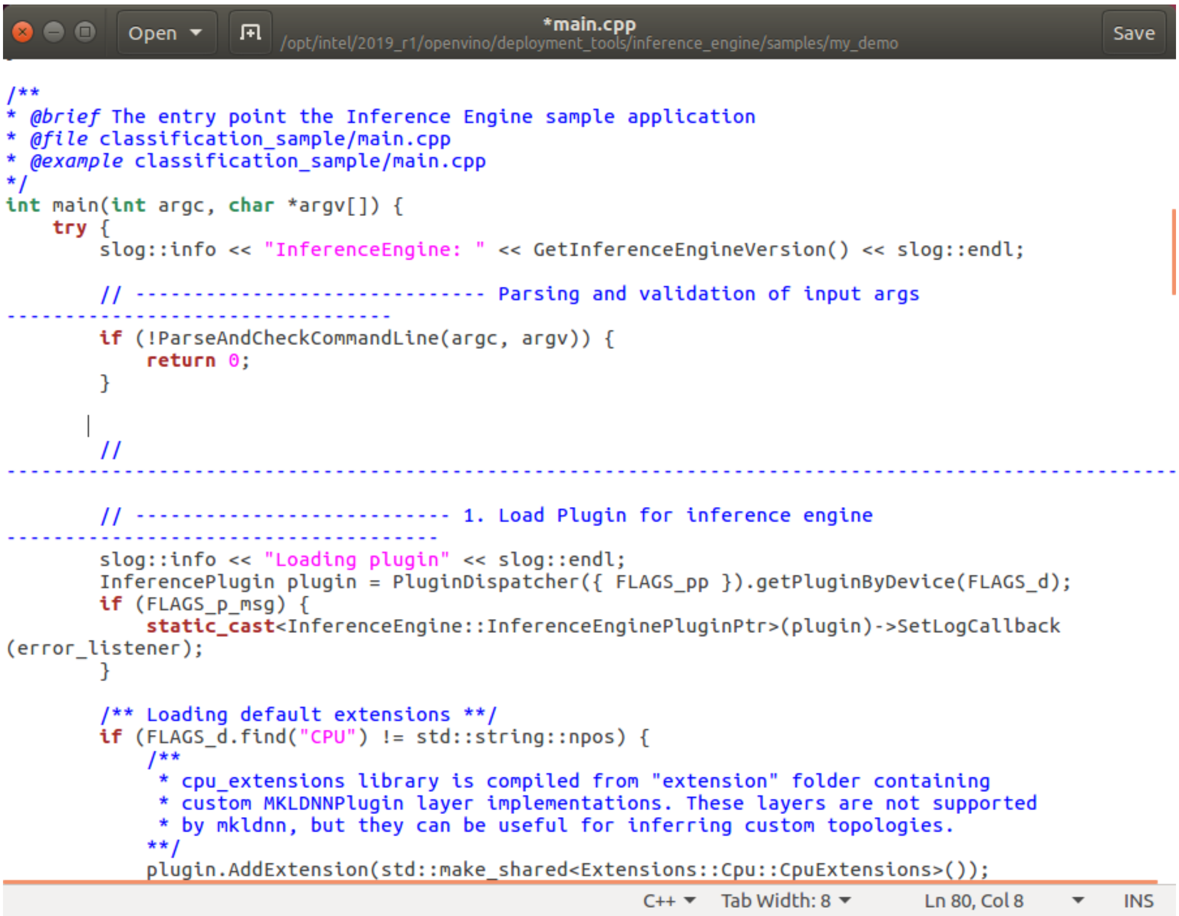
Before:



*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Open
Save
C++
Tab Width: 8
Ln 83, Col 89
INS

```
        // ------------------------------ Parsing and validation of input args
---------------------------------
        if (!ParseAndCheckCommandLine(argc, argv)) {
            return 0;
        }

        /** This vector stores paths to the processed images **/
        std::vector<std::string> imageNames;
        parseInputFilesArguments(imageNames);
        if (imageNames.empty()) throw std::logic_error("No suitable images were found");
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 1. Load Plugin for inference engine
-------------------------------------
        slog::info << "Loading plugin" << slog::endl;
        InferencePlugin plugin = PluginDispatcher({ FLAGS_pp }).getPluginByDevice(FLAGS_d);
        if (FLAGS_p_msg) {
            static_cast<InferenceEngine::InferenceEnginePluginPtr>(plugin)->SetLogCallback
(error_listener);
        }

        /** Loading default extensions **/
        if (FLAGS_d.find("CPU") != std::string::npos) {
            /**
             * cpu_extensions library is compiled from "extension" folder containing
             * custom MKLDNNPlugin layer implementations. These layers are not supported
             * by mkldnn, but they can be useful for inferring custom topologies.
            **/
            plugin.AddExtension(std::make_shared<Extensions::Cpu::CpuExtensions>());
        }

        if (!FLAGS_l.empty()) {
            // CPU(MKLDNN) extensions are loaded as a shared library and passed as a pointer to
base extension
            auto extension_ptr = make_so_pointer<IExtension>(FLAGS_l);
```

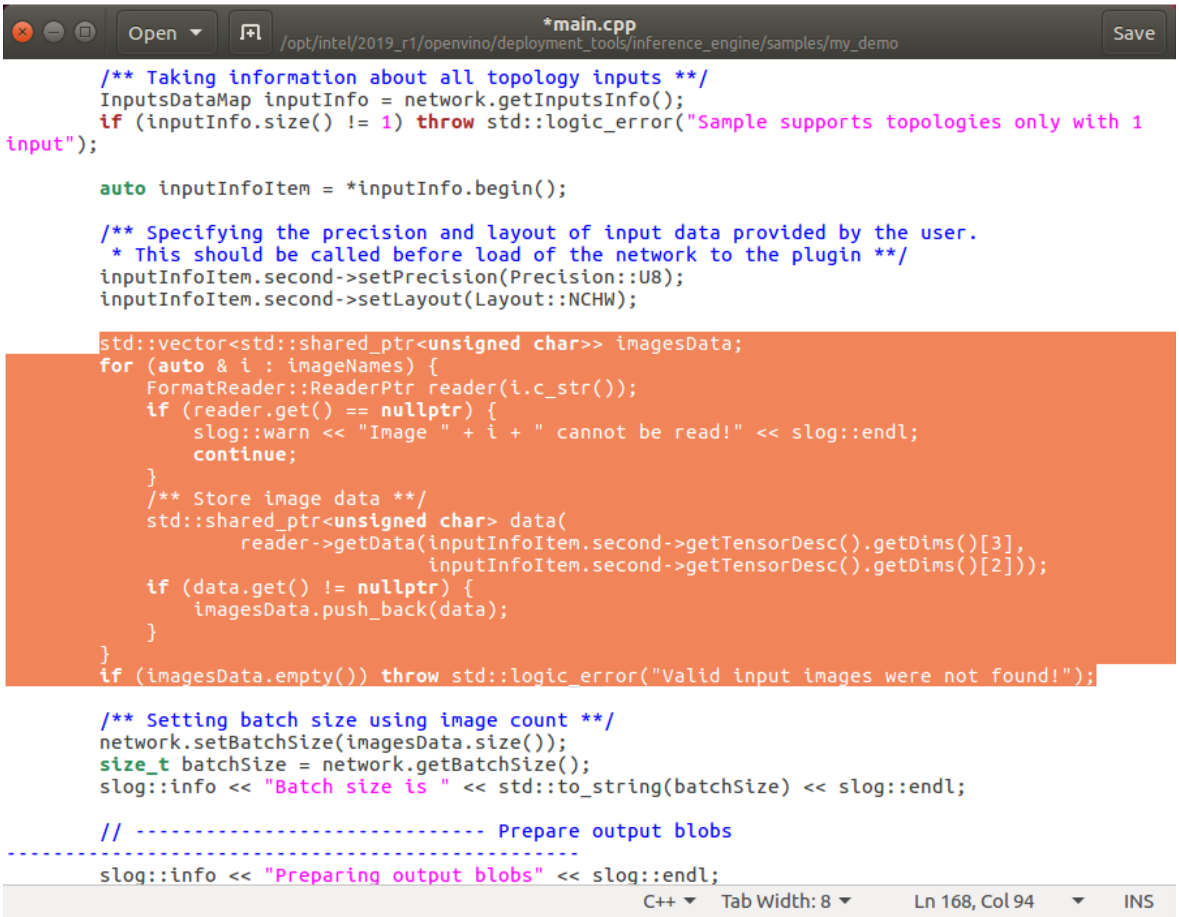
After:


*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Open
Save
C++
Tab Width: 8
Ln 80, Col 8
INS

```
/**
* @brief The entry point the Inference Engine sample application
* @file classification_sample/main.cpp
* @example classification_sample/main.cpp
*/
int main(int argc, char *argv[]) {
    try {
        slog::info << "InferenceEngine: " << GetInferenceEngineVersion() << slog::endl;

        // ------------------------------ Parsing and validation of input args
---------------------------------
        if (!ParseAndCheckCommandLine(argc, argv)) {
            return 0;
        }

       |
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 1. Load Plugin for inference engine
-------------------------------------
        slog::info << "Loading plugin" << slog::endl;
        InferencePlugin plugin = PluginDispatcher({ FLAGS_pp }).getPluginByDevice(FLAGS_d);
        if (FLAGS_p_msg) {
            static_cast<InferenceEngine::InferenceEnginePluginPtr>(plugin)->SetLogCallback
(error_listener);
        }

        /** Loading default extensions **/
        if (FLAGS_d.find("CPU") != std::string::npos) {
            /**
             * cpu_extensions library is compiled from "extension" folder containing
             * custom MKLDNNPlugin layer implementations. These layers are not supported
             * by mkldnn, but they can be useful for inferring custom topologies.
            **/
            plugin.AddExtension(std::make_shared<Extensions::Cpu::CpuExtensions>());
```

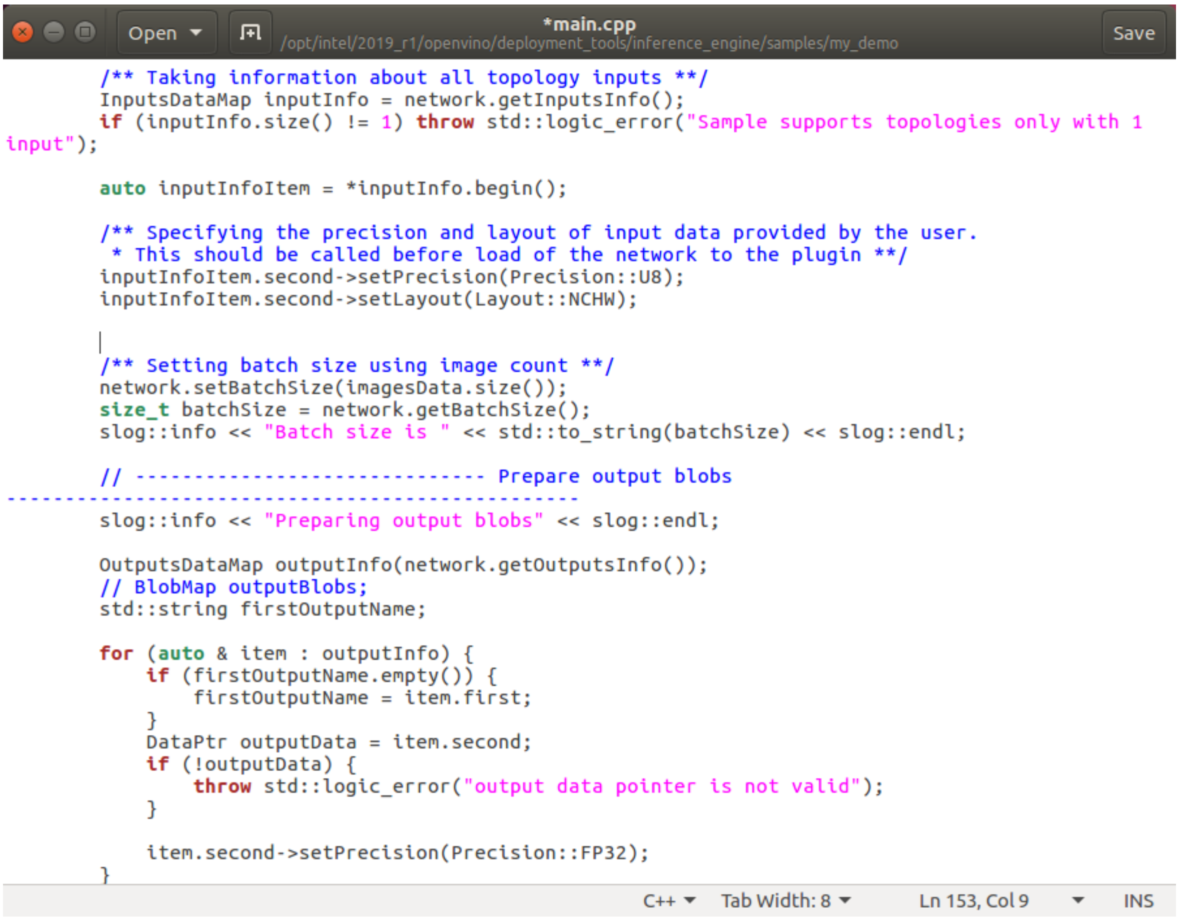
- Step4, delete the imagesData, because it’s used with imageNames.

Before:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 168, Col 94
INS

```
        /** Taking information about all topology inputs **/
        InputsDataMap inputInfo = network.getInputsInfo();
        if (inputInfo.size() != 1) throw std::logic_error("Sample supports topologies only with 1
input");

        auto inputInfoItem = *inputInfo.begin();

        /** Specifying the precision and layout of input data provided by the user.
         * This should be called before load of the network to the plugin **/
        inputInfoItem.second->setPrecision(Precision::U8);
        inputInfoItem.second->setLayout(Layout::NCHW);

        std::vector<std::shared_ptr<unsigned char>> imagesData;
        for (auto & i : imageNames) {
            FormatReader::ReaderPtr reader(i.c_str());
            if (reader.get() == nullptr) {
                slog::warn << "Image " + i + " cannot be read!" << slog::endl;
                continue;
            }
            /** Store image data **/
            std::shared_ptr<unsigned char> data(
                    reader->getData(inputInfoItem.second->getTensorDesc().getDims()[3],
                                    inputInfoItem.second->getTensorDesc().getDims()[2]));
            if (data.get() != nullptr) {
                imagesData.push_back(data);
            }
        }
        if (imagesData.empty()) throw std::logic_error("Valid input images were not found!");

        /** Setting batch size using image count **/
        network.setBatchSize(imagesData.size());
        size_t batchSize = network.getBatchSize();
        slog::info << "Batch size is " << std::to_string(batchSize) << slog::endl;

        // ----------------------------- Prepare output blobs
-------------------------------------------------
        slog::info << "Preparing output blobs" << slog::endl;
```

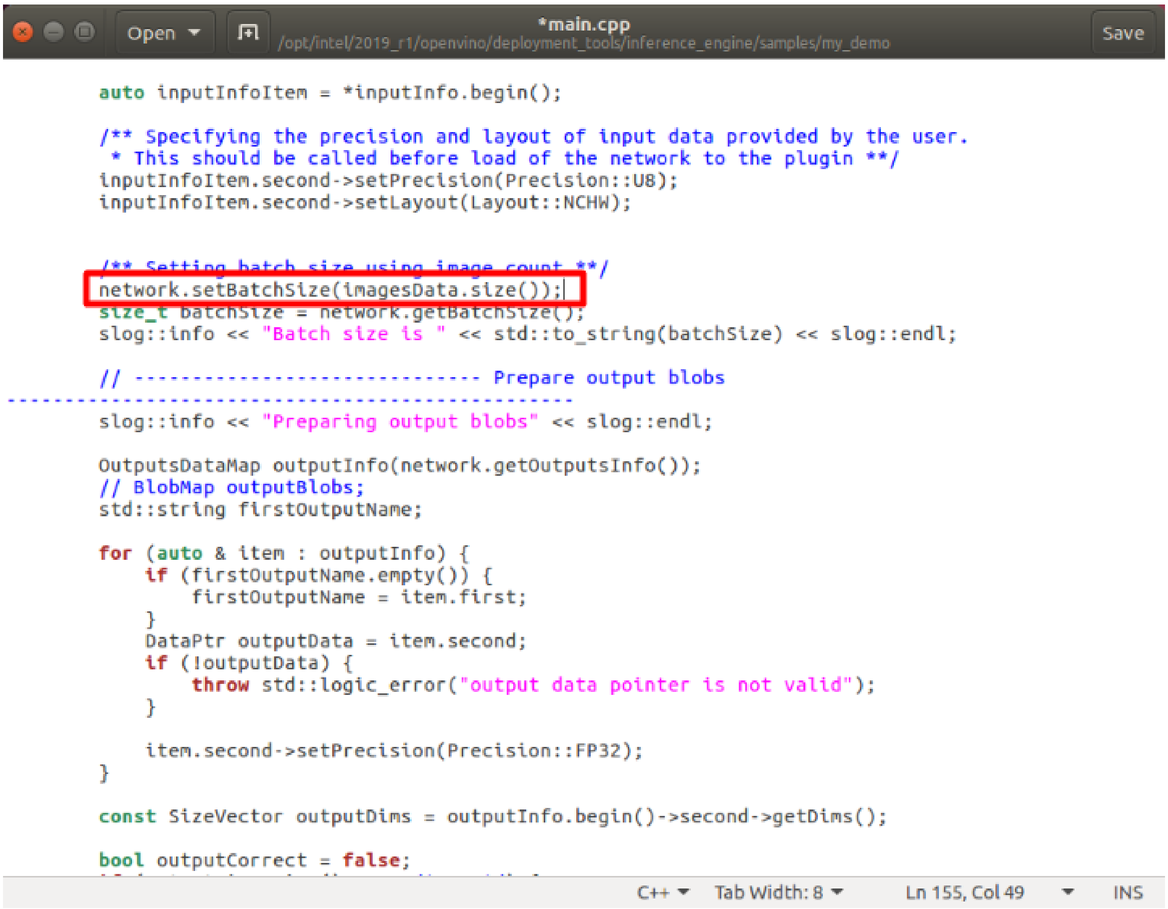
After:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 153, Col 9
INS

```
        /** Taking information about all topology inputs **/
        InputsDataMap inputInfo = network.getInputsInfo();
        if (inputInfo.size() != 1) throw std::logic_error("Sample supports topologies only with 1
input");

        auto inputInfoItem = *inputInfo.begin();

        /** Specifying the precision and layout of input data provided by the user.
         * This should be called before load of the network to the plugin **/
        inputInfoItem.second->setPrecision(Precision::U8);
        inputInfoItem.second->setLayout(Layout::NCHW);

        
        /** Setting batch size using image count **/
        network.setBatchSize(imagesData.size());
        size_t batchSize = network.getBatchSize();
        slog::info << "Batch size is " << std::to_string(batchSize) << slog::endl;

        // ----------------------------- Prepare output blobs
-------------------------------------------------
        slog::info << "Preparing output blobs" << slog::endl;

        OutputsDataMap outputInfo(network.getOutputsInfo());
        // BlobMap outputBlobs;
        std::string firstOutputName;

        for (auto & item : outputInfo) {
            if (firstOutputName.empty()) {
                firstOutputName = item.first;
            }
            DataPtr outputData = item.second;
            if (!outputData) {
                throw std::logic_error("output data pointer is not valid");
            }

            item.second->setPrecision(Precision::FP32);
        }
```

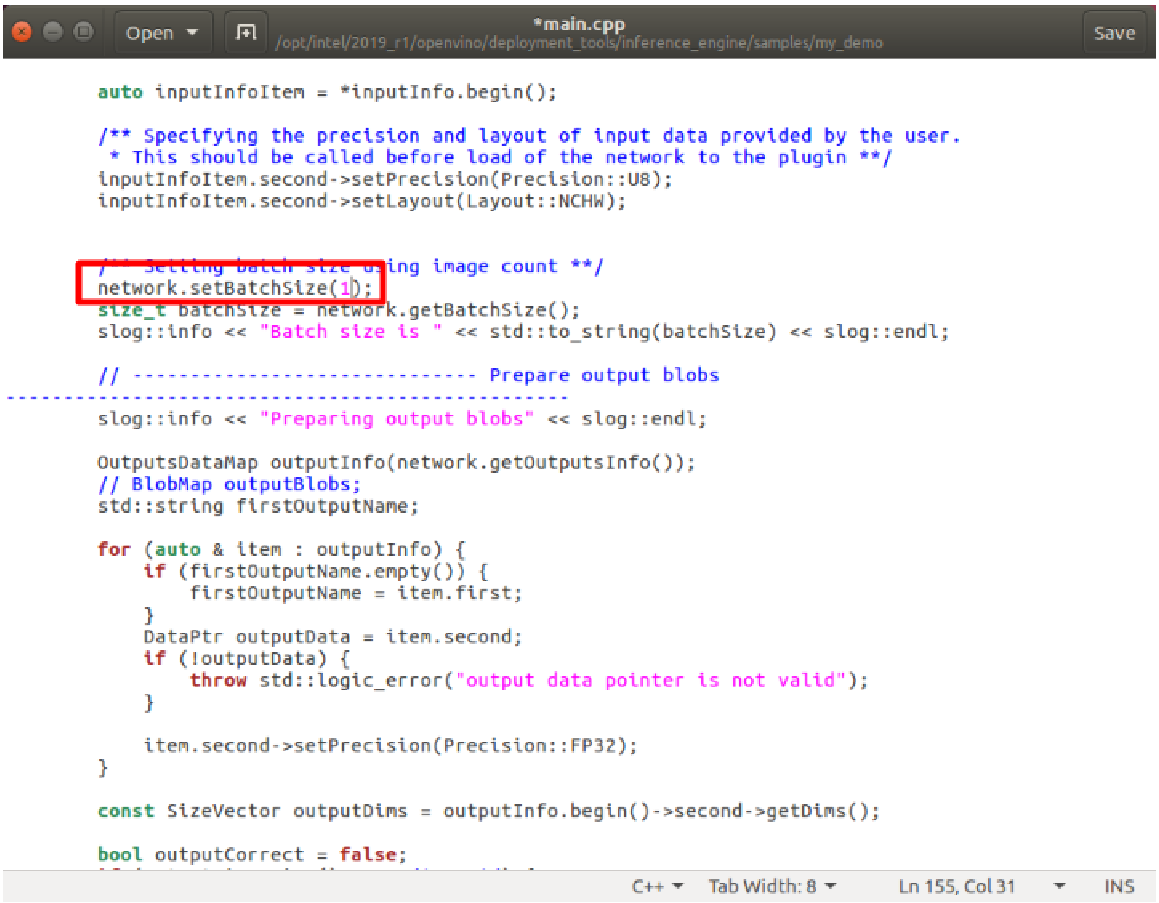
- Step5, Change network.setBatchSize(imagesData.size()) to network.setBatchSize(1).

Before:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 155, Col 49
INS

```
auto inputInfoItem = *inputInfo.begin();

/** Specifying the precision and layout of input data provided by the user.
 * This should be called before load of the network to the plugin **/
inputInfoItem.second->setPrecision(Precision::U8);
inputInfoItem.second->setLayout(Layout::NCHW);


/** Setting batch size using image count **/
network.setBatchSize(imagesData.size());
size_t batchSize = network.getBatchSize();
slog::info << "Batch size is " << std::to_string(batchSize) << slog::endl;

// ----------------------------- Prepare output blobs
-------------------------------------------------
slog::info << "Preparing output blobs" << slog::endl;

OutputsDataMap outputInfo(network.getOutputsInfo());
// BlobMap outputBlobs;
std::string firstOutputName;

for (auto & item : outputInfo) {
    if (firstOutputName.empty()) {
        firstOutputName = item.first;
    }
    DataPtr outputData = item.second;
    if (!outputData) {
        throw std::logic_error("output data pointer is not valid");
    }

    item.second->setPrecision(Precision::FP32);
}

const SizeVector outputDims = outputInfo.begin()->second->getDims();

bool outputCorrect = false;
```

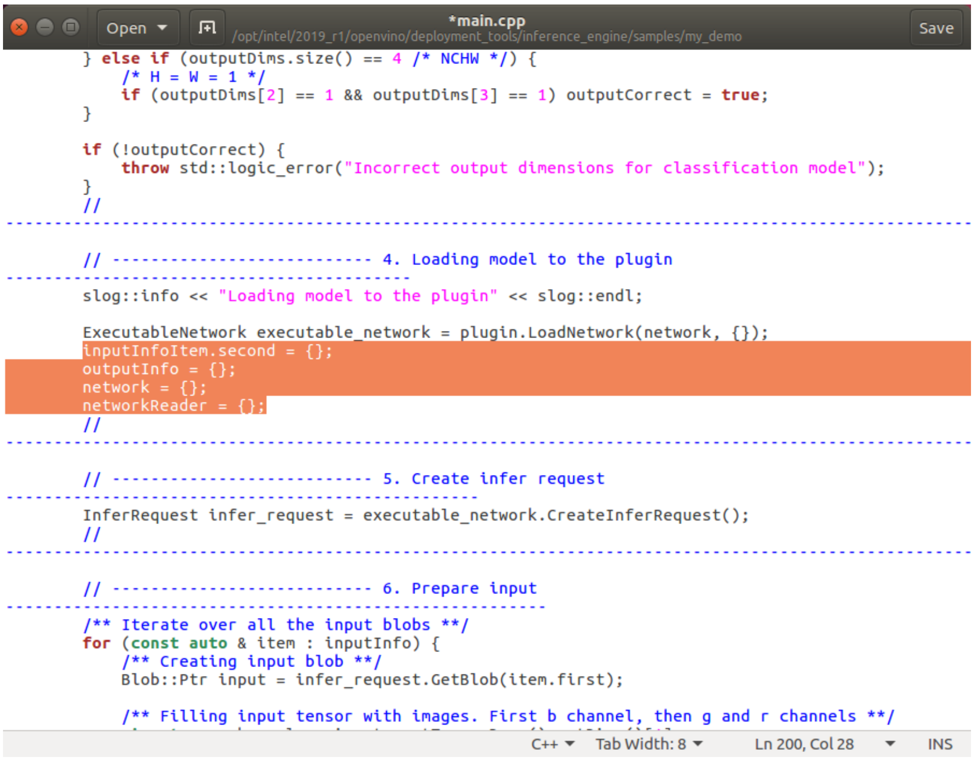
After:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 155, Col 31
INS

```
auto inputInfoItem = *inputInfo.begin();

/** Specifying the precision and layout of input data provided by the user.
 * This should be called before load of the network to the plugin **/
inputInfoItem.second->setPrecision(Precision::U8);
inputInfoItem.second->setLayout(Layout::NCHW);


/** Setting batch size using image count **/
network.setBatchSize(1);
size_t batchSize = network.getBatchSize();
slog::info << "Batch size is " << std::to_string(batchSize) << slog::endl;

// ----------------------------- Prepare output blobs
-------------------------------------------------
slog::info << "Preparing output blobs" << slog::endl;

OutputsDataMap outputInfo(network.getOutputsInfo());
// BlobMap outputBlobs;
std::string firstOutputName;

for (auto & item : outputInfo) {
    if (firstOutputName.empty()) {
        firstOutputName = item.first;
    }
    DataPtr outputData = item.second;
    if (!outputData) {
        throw std::logic_error("output data pointer is not valid");
    }

    item.second->setPrecision(Precision::FP32);
}

const SizeVector outputDims = outputInfo.begin()->second->getDims();

bool outputCorrect = false;
```

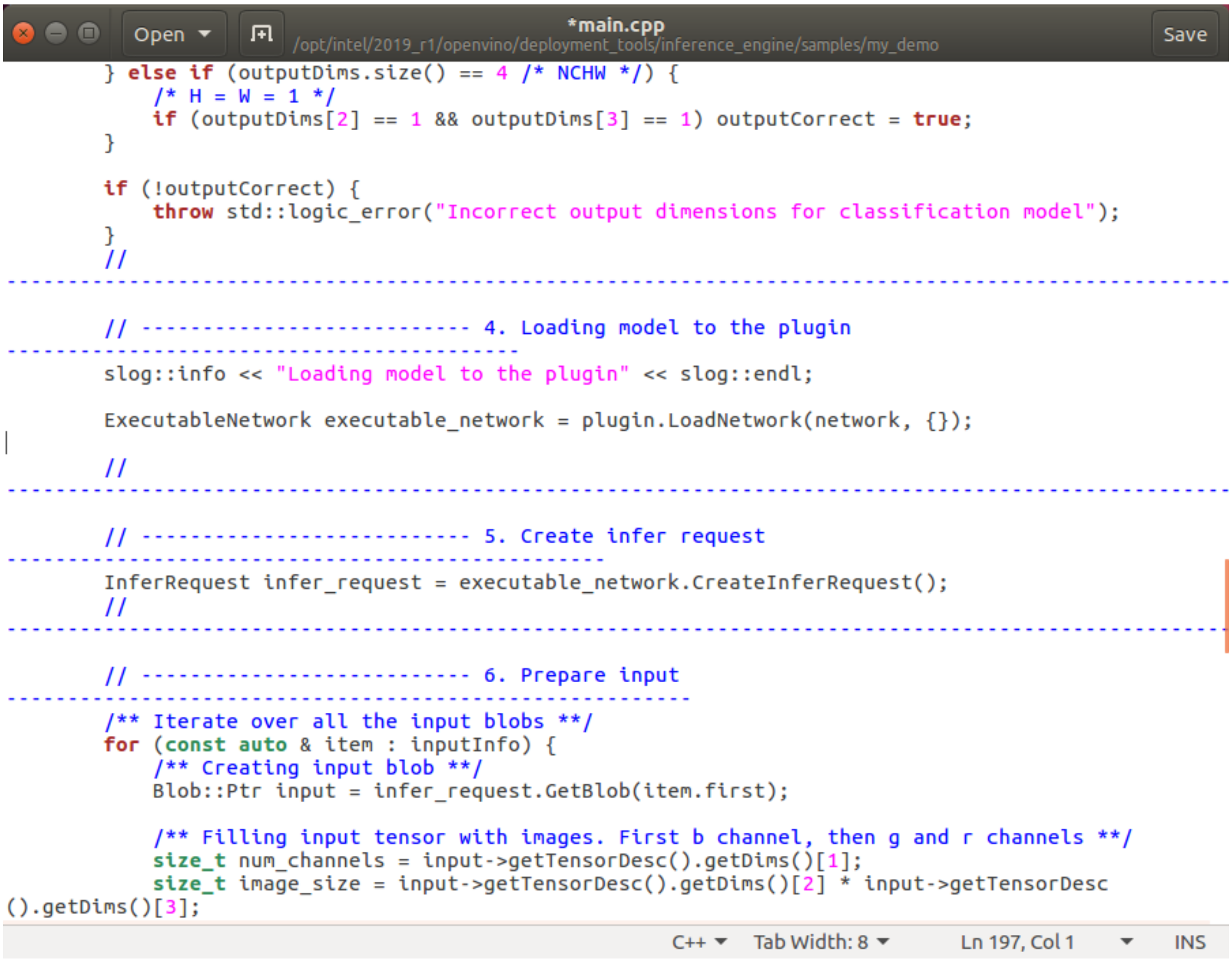
- Step6, Delete
  inputInfoItem.second = {};
  outputInfo = {};
  network = {};
  networkReader = {};

  Before:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 200, Col 28
INS

```
        } else if (outputDims.size() == 4 /* NCHW */) {
            /* H = W = 1 */
            if (outputDims[2] == 1 && outputDims[3] == 1) outputCorrect = true;
        }

        if (!outputCorrect) {
            throw std::logic_error("Incorrect output dimensions for classification model");
        }
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 4. Loading model to the plugin
------------------------------------------
        slog::info << "Loading model to the plugin" << slog::endl;

        ExecutableNetwork executable_network = plugin.LoadNetwork(network, {});
        inputInfoItem.second = {};
        outputInfo = {};
        network = {};
        networkReader = {};
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 5. Create infer request
-------------------------------------------------
        InferRequest infer_request = executable_network.CreateInferRequest();
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 6. Prepare input
--------------------------------------------------------
        /** Iterate over all the input blobs **/
        for (const auto & item : inputInfo) {
            /** Creating input blob **/
            Blob::Ptr input = infer_request.GetBlob(item.first);

            /** Filling input tensor with images. First b channel, then g and r channels **/
```

  After:

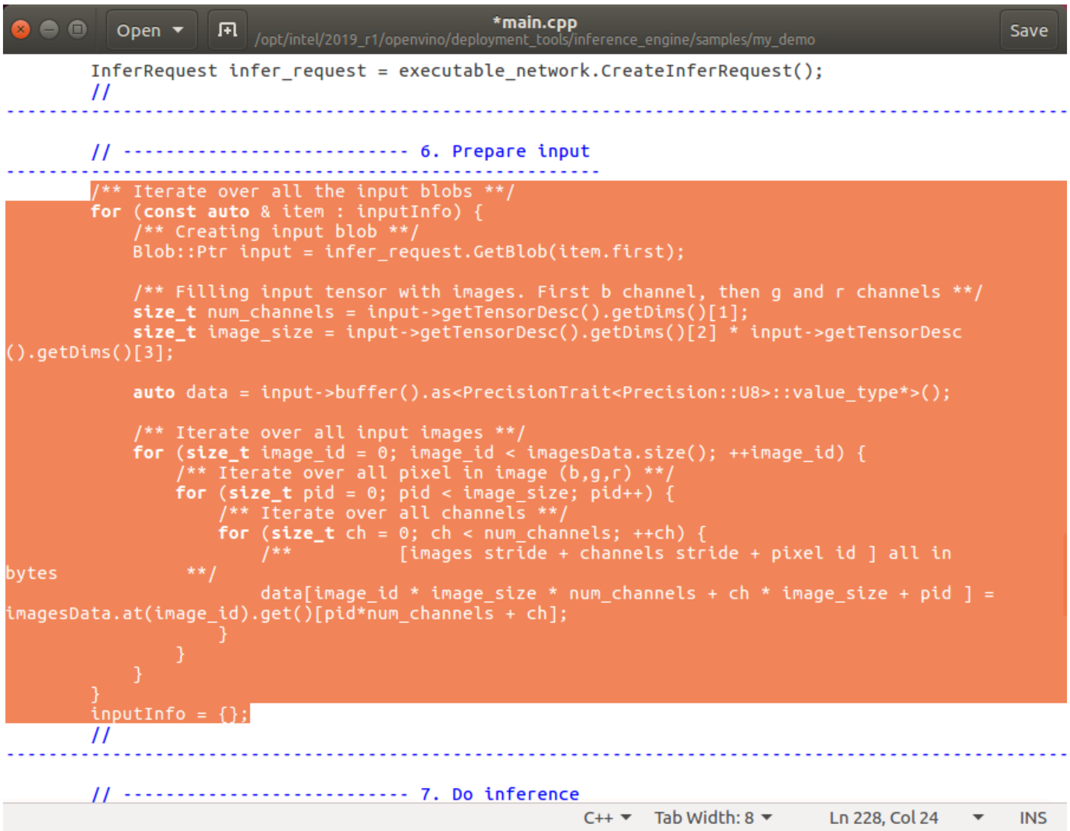


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 197, Col 1
INS

```
        } else if (outputDims.size() == 4 /* NCHW */) {
            /* H = W = 1 */
            if (outputDims[2] == 1 && outputDims[3] == 1) outputCorrect = true;
        }

        if (!outputCorrect) {
            throw std::logic_error("Incorrect output dimensions for classification model");
        }
        //
-------------------------------------------------------------------------------------------------

        // --------------------------- 4. Loading model to the plugin
------------------------------------------
        slog::info << "Loading model to the plugin" << slog::endl;

        ExecutableNetwork executable_network = plugin.LoadNetwork(network, {});

        //
-------------------------------------------------------------------------------------------------

        // --------------------------- 5. Create infer request
-------------------------------------------------
        InferRequest infer_request = executable_network.CreateInferRequest();
        //
-------------------------------------------------------------------------------------------------

        // --------------------------- 6. Prepare input
--------------------------------------------------------
        /** Iterate over all the input blobs **/
        for (const auto & item : inputInfo) {
            /** Creating input blob **/
            Blob::Ptr input = infer_request.GetBlob(item.first);

            /** Filling input tensor with images. First b channel, then g and r channels **/
            size_t num_channels = input->getTensorDesc().getDims()[1];
            size_t image_size = input->getTensorDesc().getDims()[2] * input->getTensorDesc
().getDims()[3];
```

- Step7, delete operation of "step 6, Prepare input"

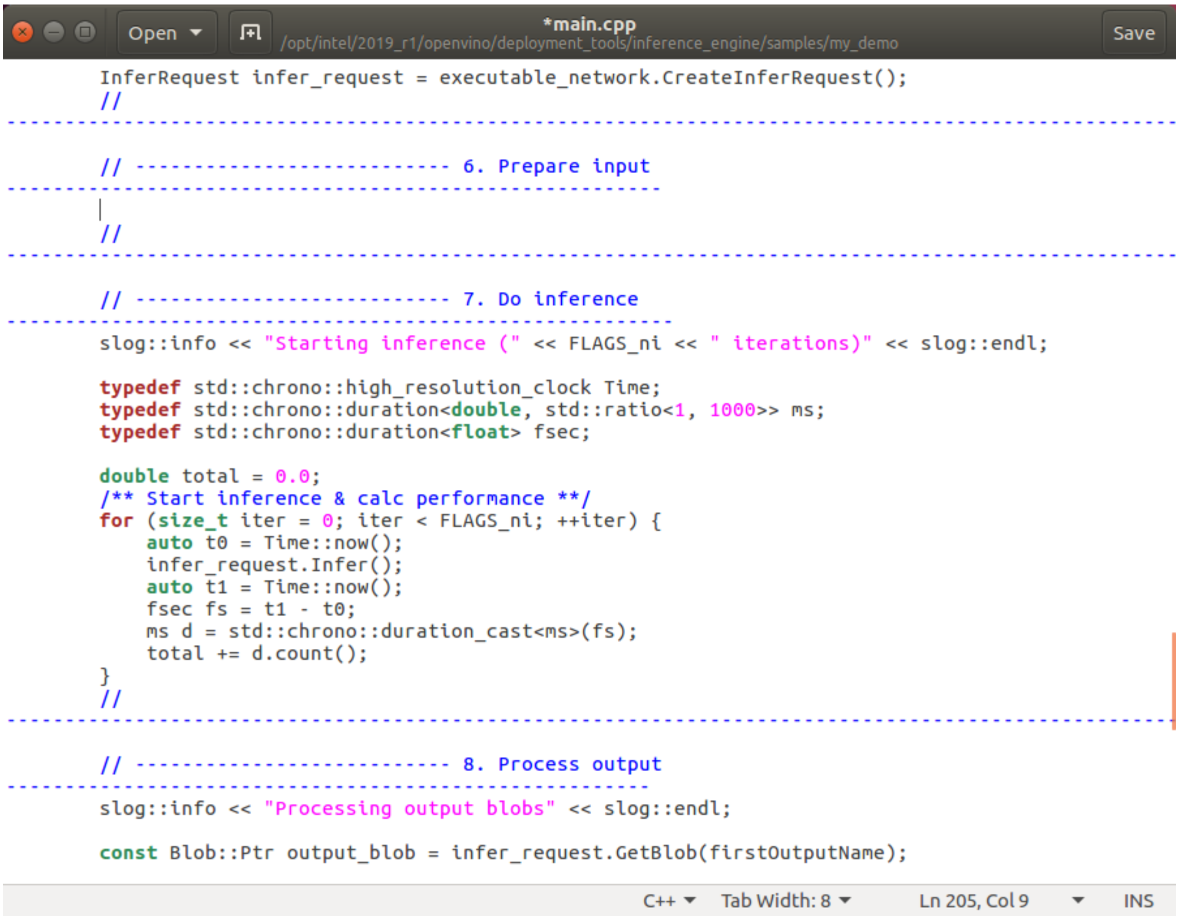
Before:



Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 228, Col 24
INS

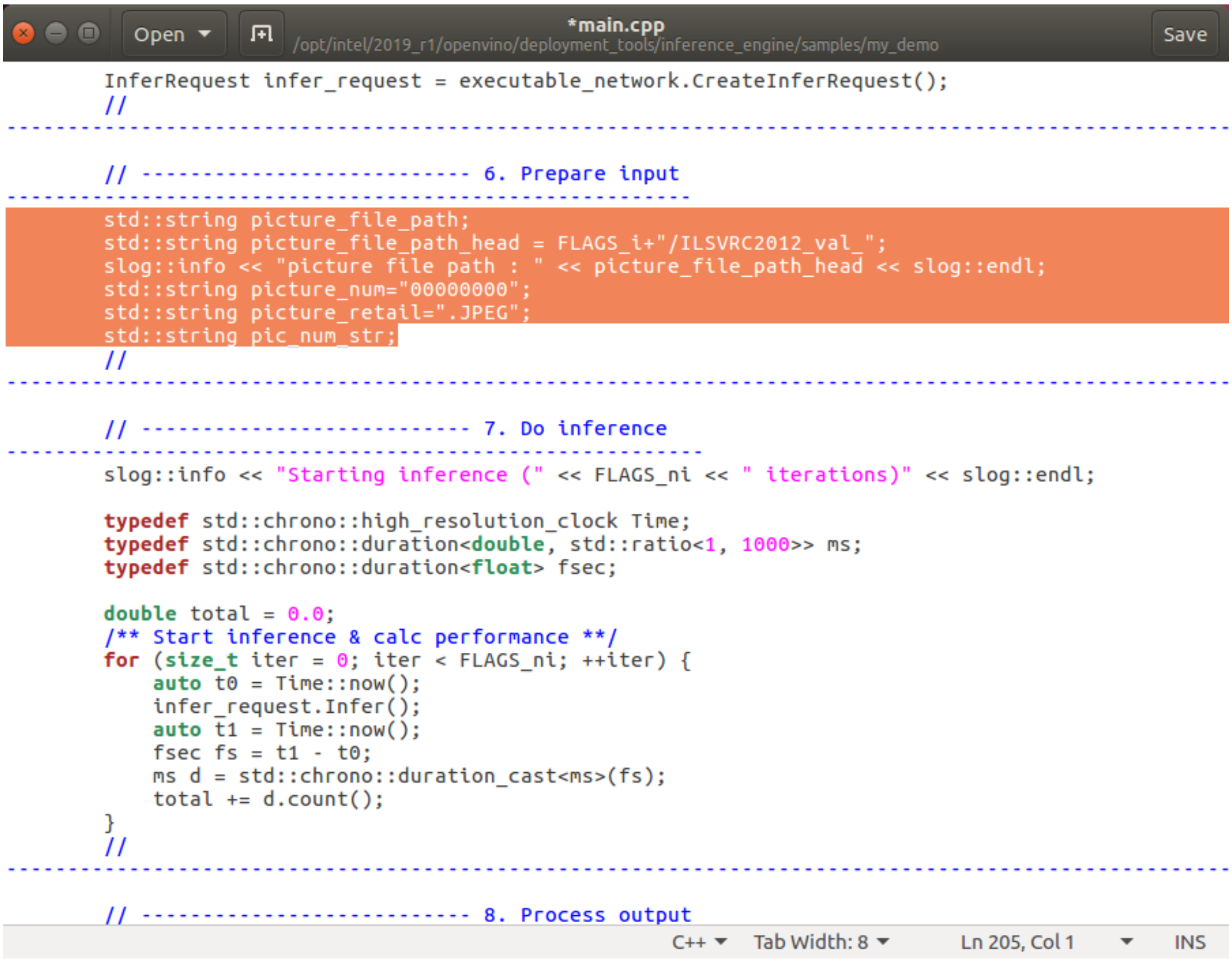
```
        InferRequest infer_request = executable_network.CreateInferRequest();
        //
-------------------------------------------------------------------------------------------------

        // --------------------------- 6. Prepare input
--------------------------------------------------------
        /** Iterate over all the input blobs **/
        for (const auto & item : inputInfo) {
            /** Creating input blob **/
            Blob::Ptr input = infer_request.GetBlob(item.first);

            /** Filling input tensor with images. First b channel, then g and r channels **/
            size_t num_channels = input->getTensorDesc().getDims()[1];
            size_t image_size = input->getTensorDesc().getDims()[2] * input->getTensorDesc
().getDims()[3];

            auto data = input->buffer().as<PrecisionTrait<Precision::U8>::value_type*>();

            /** Iterate over all input images **/
            for (size_t image_id = 0; image_id < imagesData.size(); ++image_id) {
                /** Iterate over all pixel in image (b,g,r) **/
                for (size_t pid = 0; pid < image_size; pid++) {
                    /** Iterate over all channels **/
                    for (size_t ch = 0; ch < num_channels; ++ch) {
                        /**          [images stride + channels stride + pixel id ] all in
bytes            **/
                        data[image_id * image_size * num_channels + ch * image_size + pid ] =
imagesData.at(image_id).get()[pid*num_channels + ch];
                    }
                }
            }
        }
        inputInfo = {};
        //
-------------------------------------------------------------------------------------------------

        // --------------------------- 7. Do inference
```

After:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 205, Col 9
INS

```
        InferRequest infer_request = executable_network.CreateInferRequest();
        //
-----------------------------------------------------------------------------------------------------

        // --------------------------- 6. Prepare input
--------------------------------------------------------

        //
-----------------------------------------------------------------------------------------------------

        // --------------------------- 7. Do inference
---------------------------------------------------------
        slog::info << "Starting inference (" << FLAGS_ni << " iterations)" << slog::endl;

        typedef std::chrono::high_resolution_clock Time;
        typedef std::chrono::duration<double, std::ratio<1, 1000>> ms;
        typedef std::chrono::duration<float> fsec;

        double total = 0.0;
        /** Start inference & calc performance **/
        for (size_t iter = 0; iter < FLAGS_ni; ++iter) {
            auto t0 = Time::now();
            infer_request.Infer();
            auto t1 = Time::now();
            fsec fs = t1 - t0;
            ms d = std::chrono::duration_cast<ms>(fs);
            total += d.count();
        }
        //
-----------------------------------------------------------------------------------------------------

        // --------------------------- 8. Process output
-------------------------------------------------------
        slog::info << "Processing output blobs" << slog::endl;

        const Blob::Ptr output_blob = infer_request.GetBlob(firstOutputName);
```

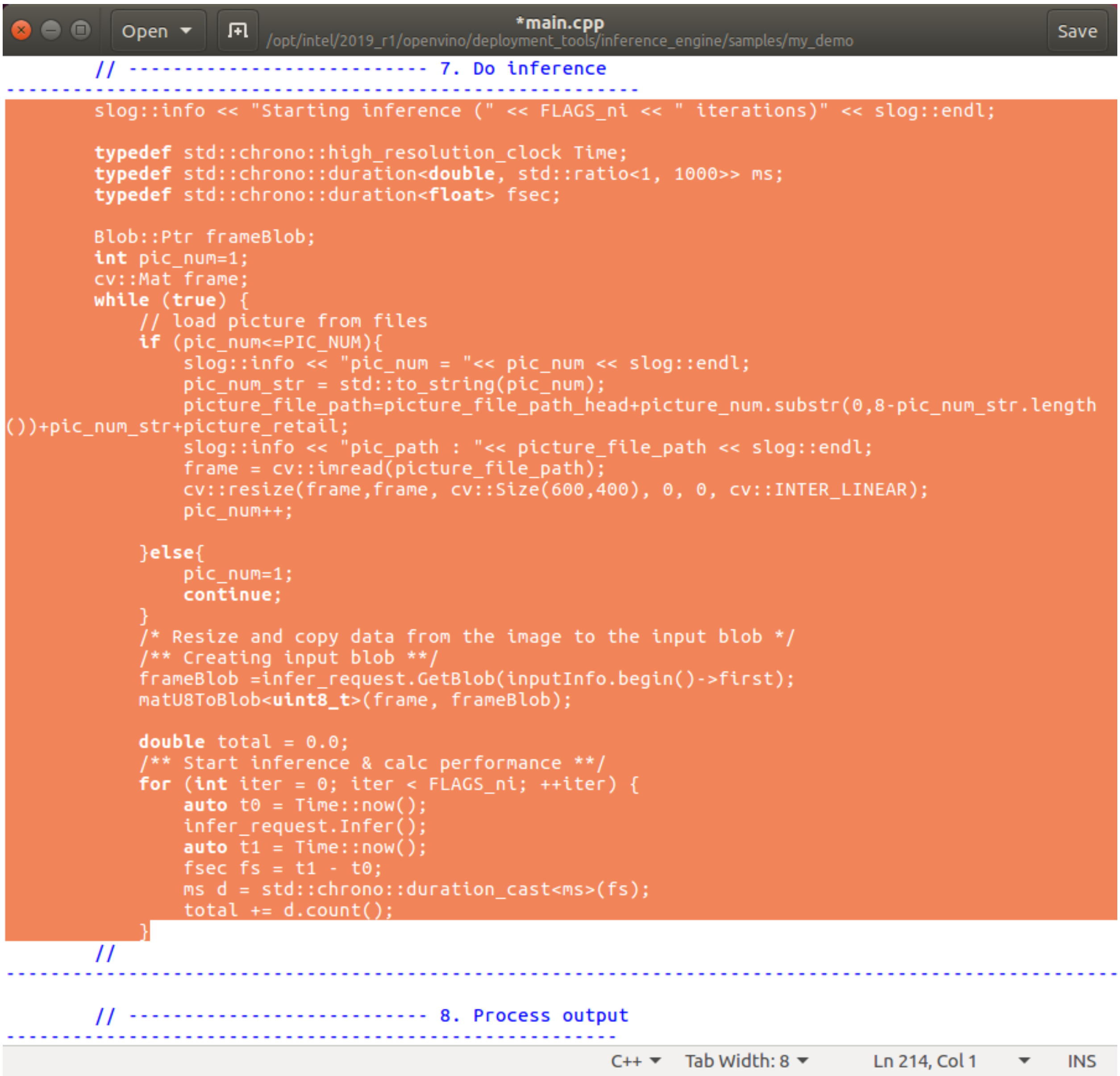
➢ Step8, add pictures relative information in "step 6, Prepare input"


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 205, Col 1
INS

```
        InferRequest infer_request = executable_network.CreateInferRequest();
        //
-----------------------------------------------------------------------------------------------

        // -------------------------- 6. Prepare input
-------------------------------------------------------
        std::string picture_file_path;
        std::string picture_file_path_head = FLAGS_i+"/ILSVRC2012_val_";
        slog::info << "picture file path : " << picture_file_path_head << slog::endl;
        std::string picture_num="00000000";
        std::string picture_retail=".JPEG";
        std::string pic_num_str;
        //
-----------------------------------------------------------------------------------------------

        // -------------------------- 7. Do inference
--------------------------------------------------------
        slog::info << "Starting inference (" << FLAGS_ni << " iterations)" << slog::endl;

        typedef std::chrono::high_resolution_clock Time;
        typedef std::chrono::duration<double, std::ratio<1, 1000>> ms;
        typedef std::chrono::duration<float> fsec;

        double total = 0.0;
        /** Start inference & calc performance **/
        for (size_t iter = 0; iter < FLAGS_ni; ++iter) {
            auto t0 = Time::now();
            infer_request.Infer();
            auto t1 = Time::now();
            fsec fs = t1 - t0;
            ms d = std::chrono::duration_cast<ms>(fs);
            total += d.count();
        }
        //
-----------------------------------------------------------------------------------------------

        // -------------------------- 8. Process output
```

```
        std::string picture_file_path;
            std::string picture_file_path_head = FLAGS_i+”/ILSVRC2012_val_”;
            slog::info << “picture file path : “ << picture_file_path_head << slog::endl;
            std::string picture_num=”00000000”;
            std::string picture_retail=”.JPEG”;
      std::string pic_num_str;
```

- Step9, modify “step 7, Do inference”

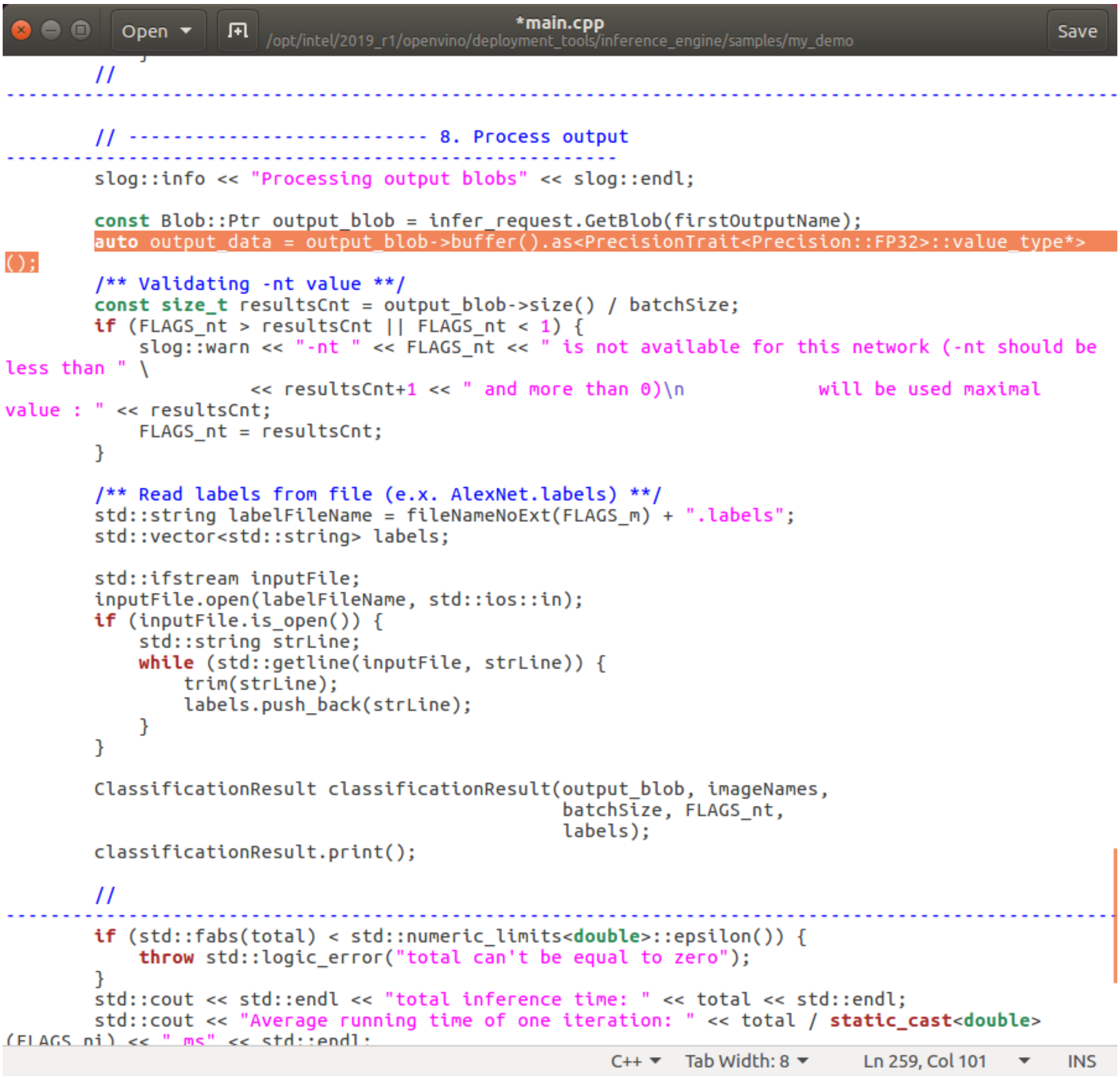

Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 214, Col 1
INS

```
        // --------------------------- 7. Do inference
--------------------------------------------------------
        slog::info << "Starting inference (" << FLAGS_ni << " iterations)" << slog::endl;

        typedef std::chrono::high_resolution_clock Time;
        typedef std::chrono::duration<double, std::ratio<1, 1000>> ms;
        typedef std::chrono::duration<float> fsec;

        Blob::Ptr frameBlob;
        int pic_num=1;
        cv::Mat frame;
        while (true) {
            // load picture from files
            if (pic_num<=PIC_NUM){
                slog::info << "pic_num = "<< pic_num << slog::endl;
                pic_num_str = std::to_string(pic_num);
                picture_file_path=picture_file_path_head+picture_num.substr(0,8-pic_num_str.length
())+pic_num_str+picture_retail;
                slog::info << "pic_path : "<< picture_file_path << slog::endl;
                frame = cv::imread(picture_file_path);
                cv::resize(frame,frame, cv::Size(600,400), 0, 0, cv::INTER_LINEAR);
                pic_num++;

            }else{
                pic_num=1;
                continue;
            }
            /* Resize and copy data from the image to the input blob */
            /** Creating input blob **/
            frameBlob =infer_request.GetBlob(inputInfo.begin()->first);
            matU8ToBlob<uint8_t>(frame, frameBlob);

            double total = 0.0;
            /** Start inference & calc performance **/
            for (int iter = 0; iter < FLAGS_ni; ++iter) {
                auto t0 = Time::now();
                infer_request.Infer();
                auto t1 = Time::now();
                fsec fs = t1 - t0;
                ms d = std::chrono::duration_cast<ms>(fs);
                total += d.count();
            }
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 8. Process output
-------------------------------------------------------
```

```
        slog::info << “Starting inference (“ << FLAGS_ni << “ iterations)” << slog::endl;


            typedef std::chrono::high_resolution_clock Time;
            typedef std::chrono::duration<double, std::ratio<1, 1000>> ms;
            typedef std::chrono::duration<float> fsec;


            Blob::Ptr frameBlob;
            int pic_num=1;
            cv::Mat frame;
            while (true) {
                // load picture from files
                if (pic_num<=PIC_NUM){
                slog::info << “pic_num = “<< pic_num << slog::endl;
                pic_num_str = std::to_string(pic_num);
                picture_file_path=picture_file_path_head+picture_num.substr(0,8-
        pic_num_str.length())+pic_num_str+picture_retail;
                slog::info << “pic_path : “<< picture_file_path << slog::endl;
                frame = cv::imread(picture_file_path);
                cv::resize(frame,frame, cv::Size(600,400), 0, 0, cv::INTER_LINEAR);
                pic_num++;

                }else{
                pic_num=1;
                continue;
                }
                /* Resize and copy data from the image to the input blob */
                    /** Creating input blob **/
                    frameBlob =infer_request.GetBlob(inputInfo.begin()->first);
                matU8ToBlob<uint8_t>(frame, frameBlob);

                double total = 0.0;
                /** Start inference & calc performance **/
                for (unsigned int iter = 0; iter < FLAGS_ni; ++iter) {
                auto t0 = Time::now();
                infer_request.Infer();
                auto t1 = Time::now();
                fsec fs = t1 – t0;
                ms d = std::chrono::duration_cast<ms>(fs);
                total += d.count();
                }
```

- Step10, modify “step 8, Process output”

Add the code as below:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 259, Col 101
INS

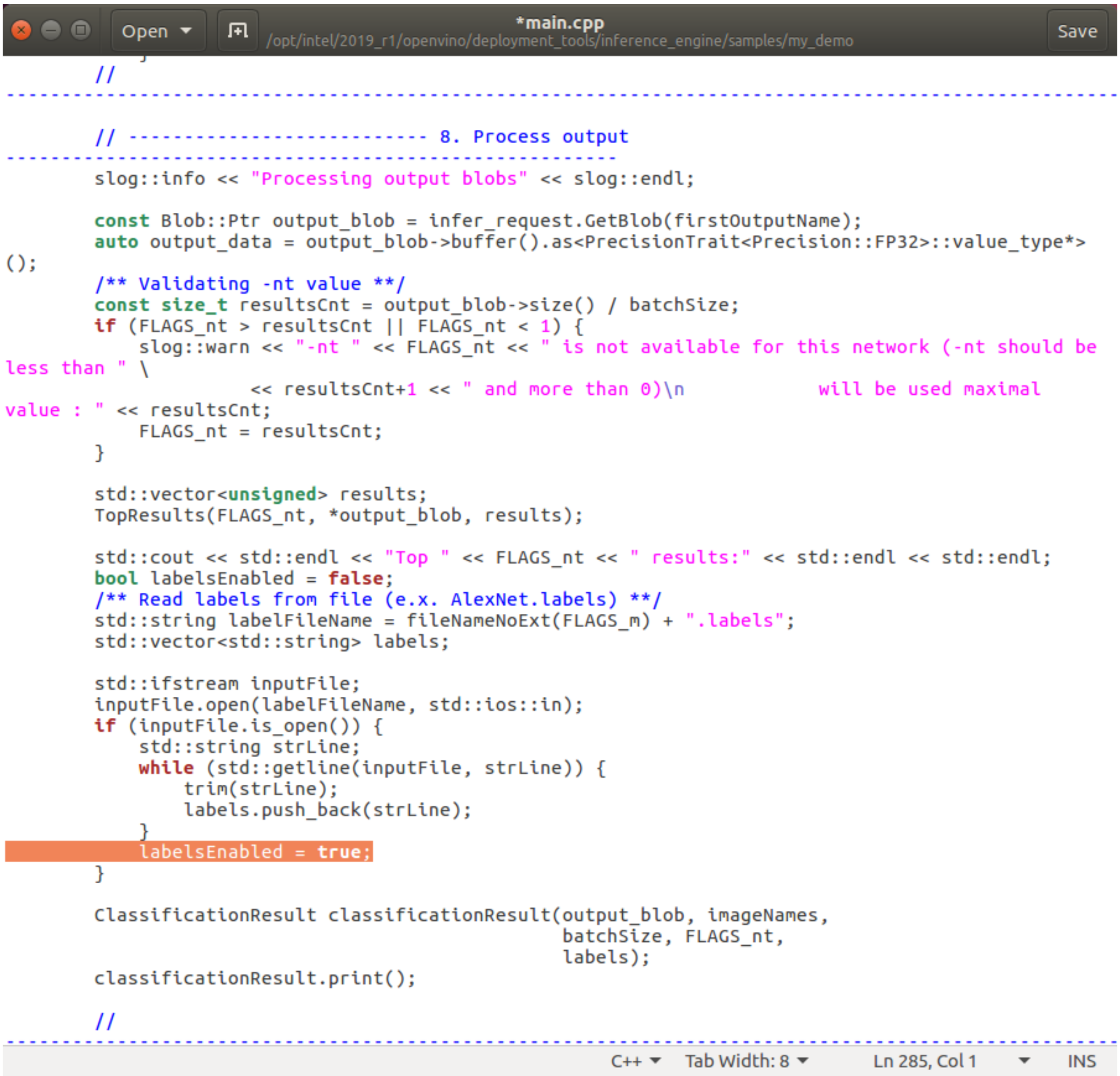
```
        //
-----------------------------------------------------------------------------------------------

        // --------------------------- 8. Process output
-------------------------------------------------------
        slog::info << "Processing output blobs" << slog::endl;

        const Blob::Ptr output_blob = infer_request.GetBlob(firstOutputName);
        auto output_data = output_blob->buffer().as<PrecisionTrait<Precision::FP32>::value_type*>
();

        /** Validating -nt value **/
        const size_t resultsCnt = output_blob->size() / batchSize;
        if (FLAGS_nt > resultsCnt || FLAGS_nt < 1) {
            slog::warn << "-nt " << FLAGS_nt << " is not available for this network (-nt should be
less than " \
                      << resultsCnt+1 << " and more than 0)\n            will be used maximal
value : " << resultsCnt;
            FLAGS_nt = resultsCnt;
        }

        /** Read labels from file (e.x. AlexNet.labels) **/
        std::string labelFileName = fileNameNoExt(FLAGS_m) + ".labels";
        std::vector<std::string> labels;

        std::ifstream inputFile;
        inputFile.open(labelFileName, std::ios::in);
        if (inputFile.is_open()) {
            std::string strLine;
            while (std::getline(inputFile, strLine)) {
                trim(strLine);
                labels.push_back(strLine);
            }
        }

        ClassificationResult classificationResult(output_blob, imageNames,
                                                  batchSize, FLAGS_nt,
                                                  labels);
        classificationResult.print();

        //
-----------------------------------------------------------------------------------------------
        if (std::fabs(total) < std::numeric_limits<double>::epsilon()) {
            throw std::logic_error("total can't be equal to zero");
        }
        std::cout << std::endl << "total inference time: " << total << std::endl;
        std::cout << "Average running time of one iteration: " << total / static_cast<double>
(FLAGS_ni) << " ms" << std::endl;
```

```
auto output_data = output_blob->buffer().as<PrecisionTrait<Precision::FP32>::value_type*>();
```

Add the code as below:

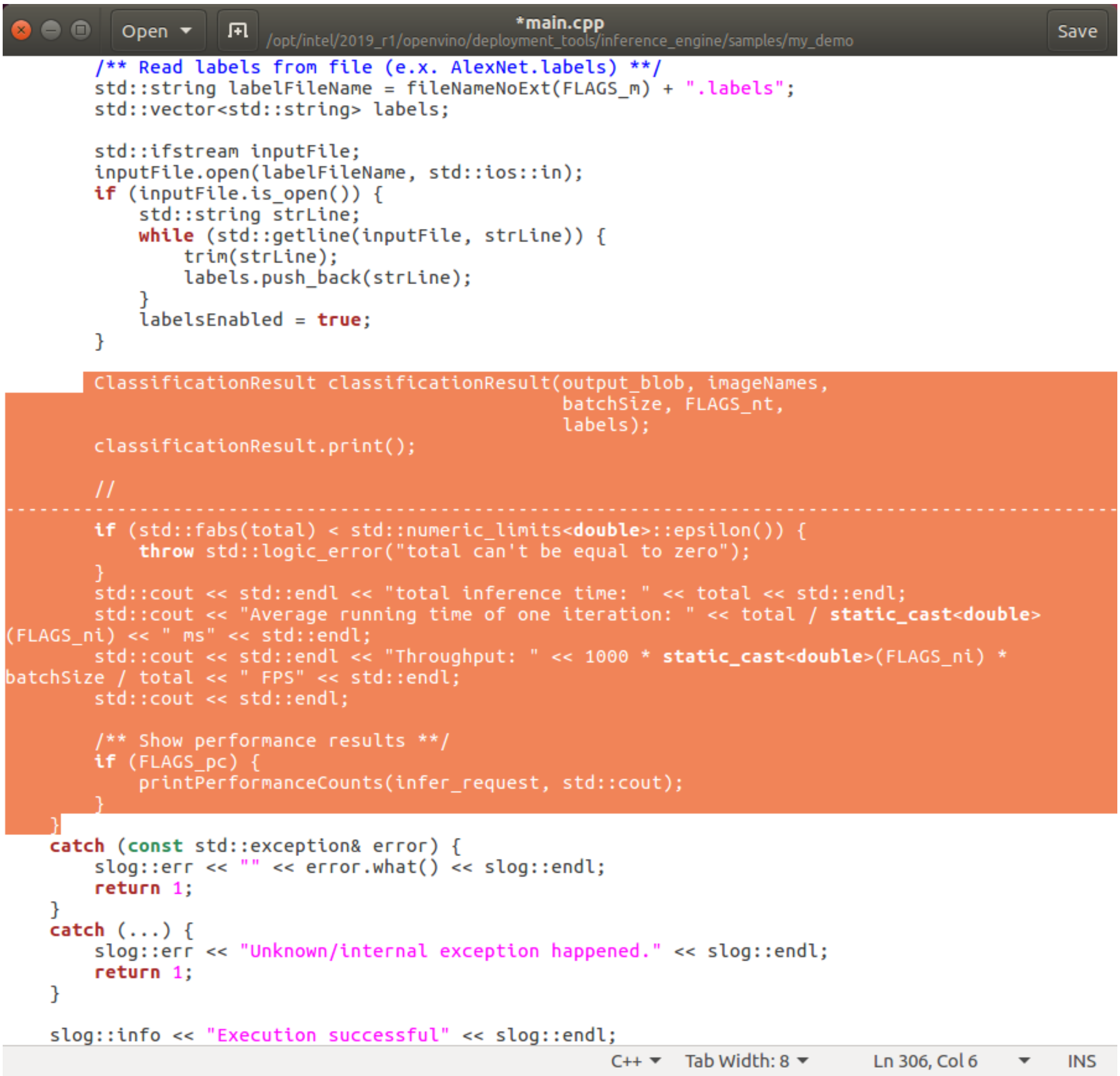
```
std::vector<unsigned> results;
TopResults(FLAGS_nt, *output_blob, results);
std::cout << std::endl << “Top “ << FLAGS_nt << “ results:” << std::endl << std::endl;
bool labelsEnabled = false;
```

Add the code as below:


*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Open
Save
C++
Tab Width: 8
Ln 285, Col 1
INS

```
        //
----------------------------------------------------------------------------------------------------

        // --------------------------- 8. Process output
-------------------------------------------------------
        slog::info << "Processing output blobs" << slog::endl;

        const Blob::Ptr output_blob = infer_request.GetBlob(firstOutputName);
        auto output_data = output_blob->buffer().as<PrecisionTrait<Precision::FP32>::value_type*>
();
        /** Validating -nt value **/
        const size_t resultsCnt = output_blob->size() / batchSize;
        if (FLAGS_nt > resultsCnt || FLAGS_nt < 1) {
            slog::warn << "-nt " << FLAGS_nt << " is not available for this network (-nt should be
less than " \
                      << resultsCnt+1 << " and more than 0)\n            will be used maximal
value : " << resultsCnt;
            FLAGS_nt = resultsCnt;
        }

        std::vector<unsigned> results;
        TopResults(FLAGS_nt, *output_blob, results);

        std::cout << std::endl << "Top " << FLAGS_nt << " results:" << std::endl << std::endl;
        bool labelsEnabled = false;
        /** Read labels from file (e.x. AlexNet.labels) **/
        std::string labelFileName = fileNameNoExt(FLAGS_m) + ".labels";
        std::vector<std::string> labels;

        std::ifstream inputFile;
        inputFile.open(labelFileName, std::ios::in);
        if (inputFile.is_open()) {
            std::string strLine;
            while (std::getline(inputFile, strLine)) {
                trim(strLine);
                labels.push_back(strLine);
            }
            labelsEnabled = true;
        }

        ClassificationResult classificationResult(output_blob, imageNames,
                                                  batchSize, FLAGS_nt,
                                                  labels);
        classificationResult.print();

        //
----------------------------------------------------------------------------------------------------
```

Delete the following code:


Open
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save
C++
Tab Width: 8
Ln 306, Col 6
INS

```
        /** Read labels from file (e.x. AlexNet.labels) **/
        std::string labelFileName = fileNameNoExt(FLAGS_m) + ".labels";
        std::vector<std::string> labels;

        std::ifstream inputFile;
        inputFile.open(labelFileName, std::ios::in);
        if (inputFile.is_open()) {
            std::string strLine;
            while (std::getline(inputFile, strLine)) {
                trim(strLine);
                labels.push_back(strLine);
            }
            labelsEnabled = true;
        }

        ClassificationResult classificationResult(output_blob, imageNames,
                                                  batchSize, FLAGS_nt,
                                                  labels);
        classificationResult.print();

        //
---------------------------------------------------------------------------------------------------
        if (std::fabs(total) < std::numeric_limits<double>::epsilon()) {
            throw std::logic_error("total can't be equal to zero");
        }
        std::cout << std::endl << "total inference time: " << total << std::endl;
        std::cout << "Average running time of one iteration: " << total / static_cast<double>
(FLAGS_ni) << " ms" << std::endl;
        std::cout << std::endl << "Throughput: " << 1000 * static_cast<double>(FLAGS_ni) *
batchSize / total << " FPS" << std::endl;
        std::cout << std::endl;

        /** Show performance results **/
        if (FLAGS_pc) {
            printPerformanceCounts(infer_request, std::cout);
        }
    }
    catch (const std::exception& error) {
        slog::err << "" << error.what() << slog::endl;
        return 1;
    }
    catch (...) {
        slog::err << "Unknown/internal exception happened." << slog::endl;
        return 1;
    }

    slog::info << "Execution successful" << slog::endl;
```

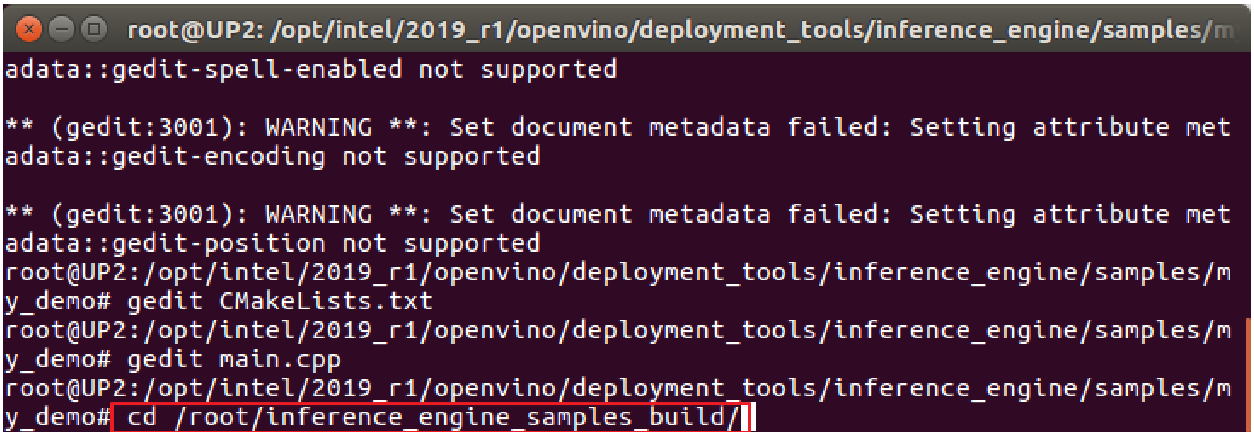
- Step11, modify the loop operation “ for” below /**print the result iterating over each batch**/ , and add picture display code.

```
    /** Print the result iterating over each batch **/
                for ( unsigned int id = 0, cnt = 0; cnt < FLAGS_nt; ++cnt, ++id) {
                    std::cout.precision(7);
                    /** Getting probability for resulting class **/
                    const auto result = output_data[results[id]];
                    std::cout << std::left << std::fixed << results[id] << “ “ << result;
                    if (labelsEnabled) {
                        std::cout << “ label “ << labels[results[id]] << std::endl;
                    } else {
                        std::cout << “ label #” << results[id] << std::endl;
                    }
                }
```

```
            std::cout << std::endl;

            std::cout << std::endl << “total inference time: “ << total << std::endl;
            std::cout   <<   std::endl   <<   “Throughput:   “   <<   1000   *
static_cast<double>(FLAGS_ni) * batchSize / total << “ FPS” << std::endl;
            std::cout << std::endl;

            /** Show performance results **/
            if (FLAGS_pc) {
                printPerformanceCounts(infer_request, std::cout);
            }

            //----------- paint picture -----------------
            std::ostringstream out;
            out  <<  “Detection  time    :  “  <<  std::fixed  <<  std::setprecision(2)
<<total
                << “ ms (“
                << 1000.f / total << “ fps)”;
            cv::putText(frame,          out.str(),          cv::Point2f(0,          30),
cv::FONT_HERSHEY_TRIPLEX, 0.5,cv::Scalar(255, 255, 0));
            out.str(“”);
            out << “Detection result    : “ << std::fixed << std::setprecision(2) <<
“ Label: “ << labels[results[0]] << “ “<< output_data[results[0]];
            cv::putText(frame,          out.str(),          cv::Point2f(0,          60),
cv::FONT_HERSHEY_TRIPLEX, 0.5,cv::Scalar(0, 0, 255));

            //---------------- picture display -------------------
            cv::imshow(“Detection results”, frame);
            const int key = cv::waitKey(1000);
            if (27 == key)    // Esc
                    break;
        }
    }
```

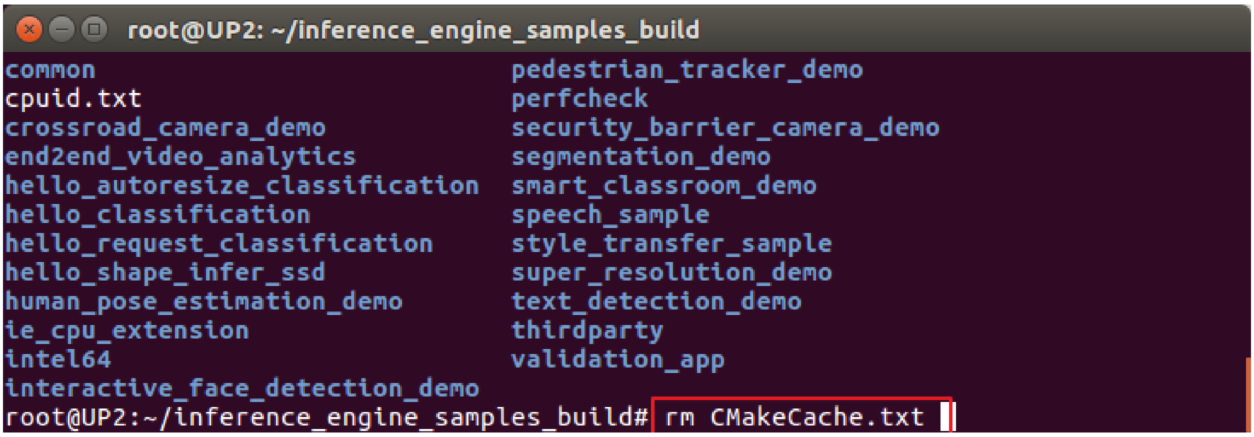
- ➢ Step12, After the modifying, save it as main.cpp.

8. Enter “*cd /root/inference_engine_samples_build/*” to go to the sample build folder.


root@UP2: /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m

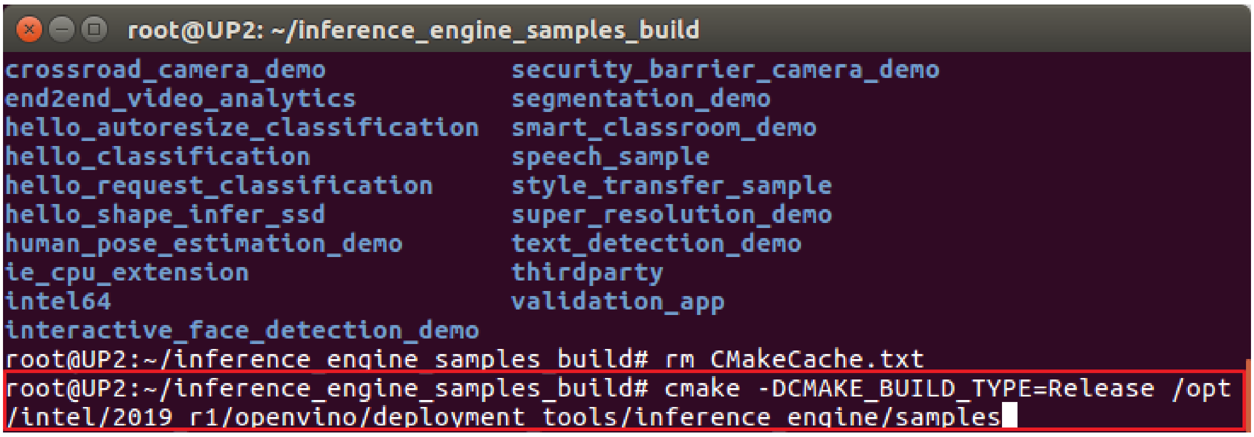
```
adata::gedit-spell-enabled not supported

** (gedit:3001): WARNING **: Set document metadata failed: Setting attribute met
adata::gedit-encoding not supported

** (gedit:3001): WARNING **: Set document metadata failed: Setting attribute met
adata::gedit-position not supported
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m
y_demo# gedit CMakeLists.txt
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m
y_demo# gedit main.cpp
root@UP2:/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/m
y_demo# cd /root/inference_engine_samples_build/
```

9. Enter “*rm CmakeCache.txt*” to clean the build cache.



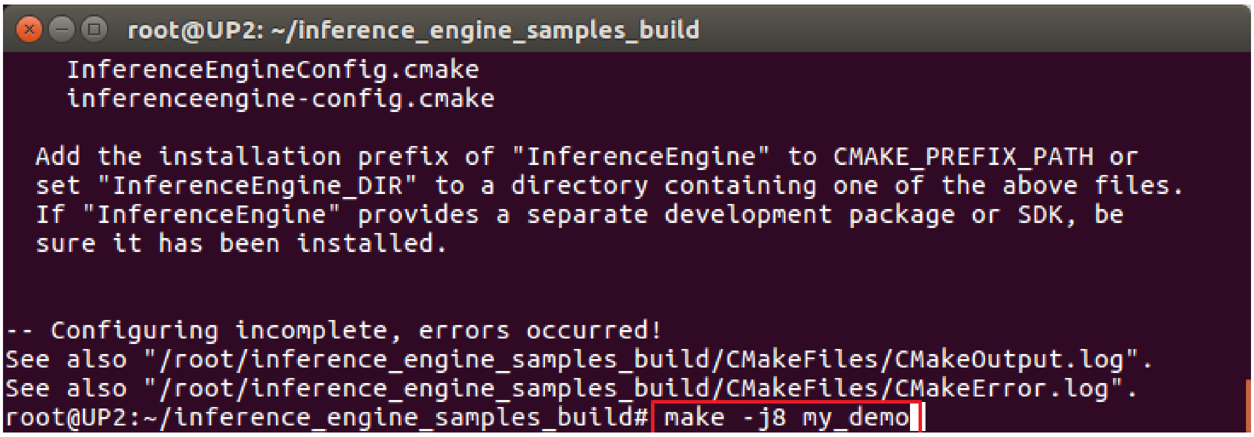
```
common                           pedestrian_tracker_demo
cpuid.txt                        perfcheck
crossroad_camera_demo            security_barrier_camera_demo
end2end_video_analytics          segmentation_demo
hello_autoresize_classification  smart_classroom_demo
hello_classification             speech_sample
hello_request_classification     style_transfer_sample
hello_shape_infer_ssd            super_resolution_demo
human_pose_estimation_demo       text_detection_demo
ie_cpu_extension                 thirdparty
intel64                          validation_app
interactive_face_detection_demo
root@UP2:~/inference_engine_samples_build# rm CMakeCache.txt 
```

10. Enter “*cmake -DCMAKE_BUILD_TYPE=Release \ /opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples*” to copy the file to the samples build directory.



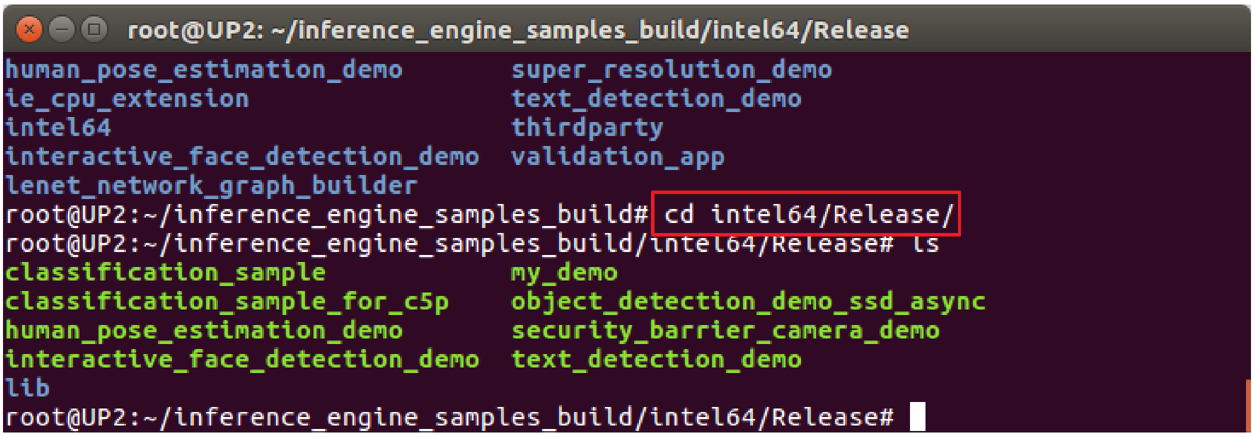
```
crossroad_camera_demo            security_barrier_camera_demo
end2end_video_analytics          segmentation_demo
hello_autoresize_classification  smart_classroom_demo
hello_classification             speech_sample
hello_request_classification     style_transfer_sample
hello_shape_infer_ssd            super_resolution_demo
human_pose_estimation_demo       text_detection_demo
ie_cpu_extension                 thirdparty
intel64                          validation_app
interactive_face_detection_demo
root@UP2:~/inference_engine_samples_build# rm CMakeCache.txt
root@UP2:~/inference_engine_samples_build# cmake -DCMAKE_BUILD_TYPE=Release /opt
/intel/2019_r1/openvino/deployment_tools/inference_engine/samples
```

11. Enter “*make -j8 my_demo*” to compile the application program, please be patient for the compilation process.



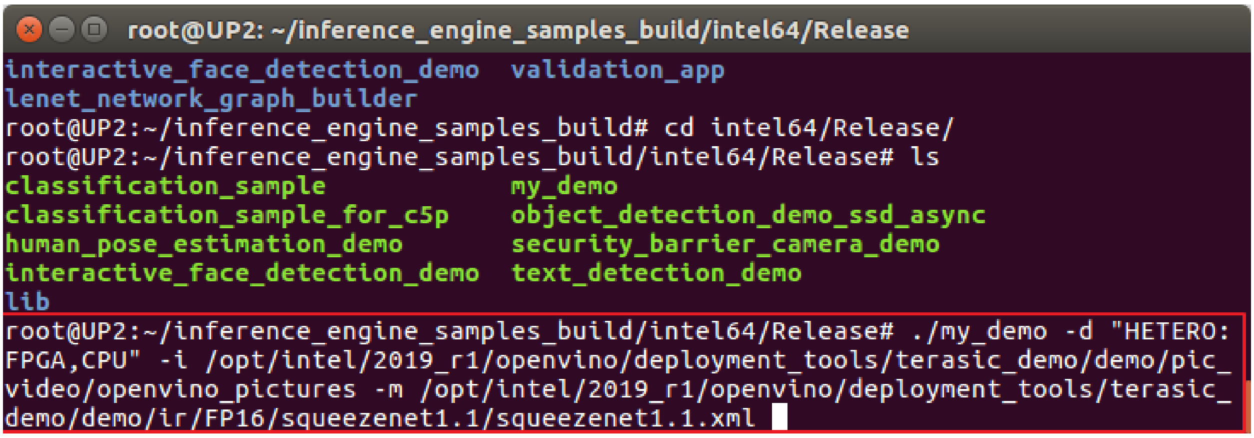
```
    InferenceEngineConfig.cmake
    inferenceengine-config.cmake

  Add the installation prefix of "InferenceEngine" to CMAKE_PREFIX_PATH or
  set "InferenceEngine_DIR" to a directory containing one of the above files.
  If "InferenceEngine" provides a separate development package or SDK, be
  sure it has been installed.


-- Configuring incomplete, errors occurred!
See also "/root/inference_engine_samples_build/CMakeFiles/CMakeOutput.log".
See also "/root/inference_engine_samples_build/CMakeFiles/CMakeError.log".
root@UP2:~/inference_engine_samples_build# make -j8 my_demo
```

12. Enter "*cd intel64/Release*" to switch to app directory. Under this folder, the corresponding executable file for my_demo is generated, a new application is completed



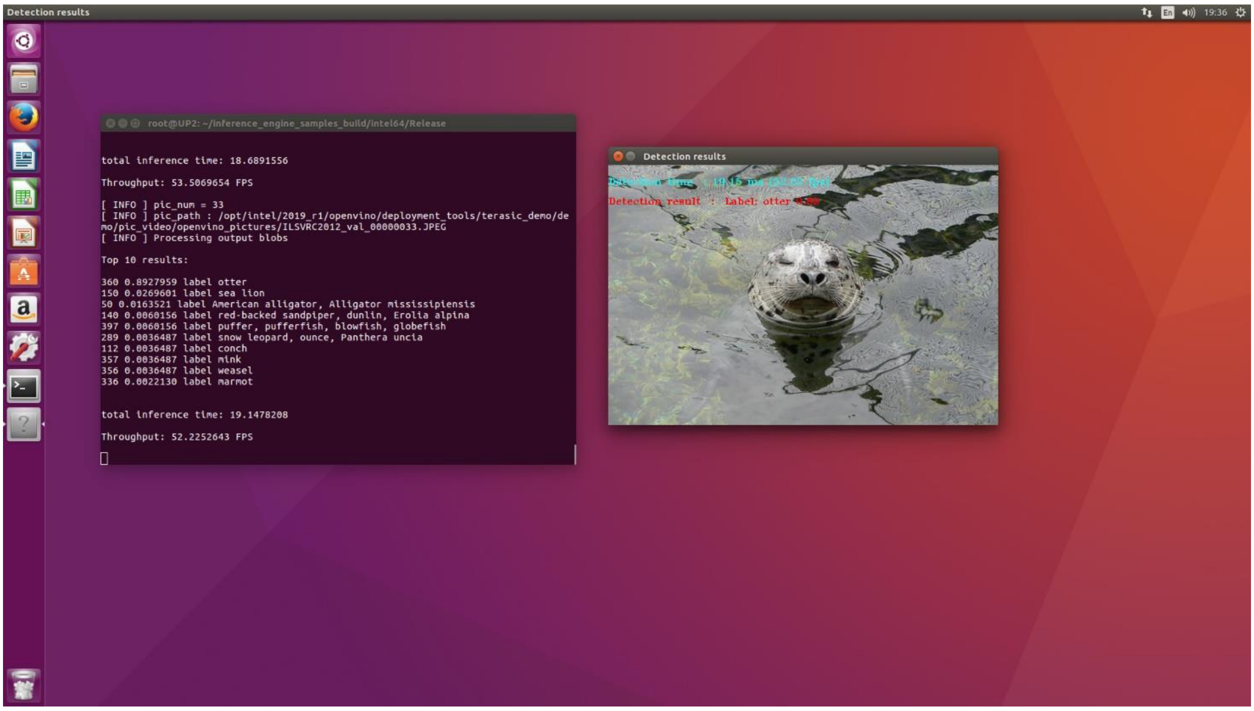
```
human_pose_estimation_demo       super_resolution_demo
ie_cpu_extension                 text_detection_demo
intel64                          thirdparty
interactive_face_detection_demo  validation_app
lenet_network_graph_builder
root@UP2:~/inference_engine_samples_build# cd intel64/Release/
root@UP2:~/inference_engine_samples_build/intel64/Release# ls
classification_sample            my_demo
classification_sample_for_c5p    object_detection_demo_ssd_async
human_pose_estimation_demo       security_barrier_camera_demo
interactive_face_detection_demo  text_detection_demo
lib
root@UP2:~/inference_engine_samples_build/intel64/Release# 
```

13. Enter "*./my_demo -d "HETERO:FPGA,CPU" -i *
/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/
*pic_video/openvino_pictures -m *
/opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/ir/
FP16/squeezenet1.1/squeezenet1.1.xml" to execute the host app.

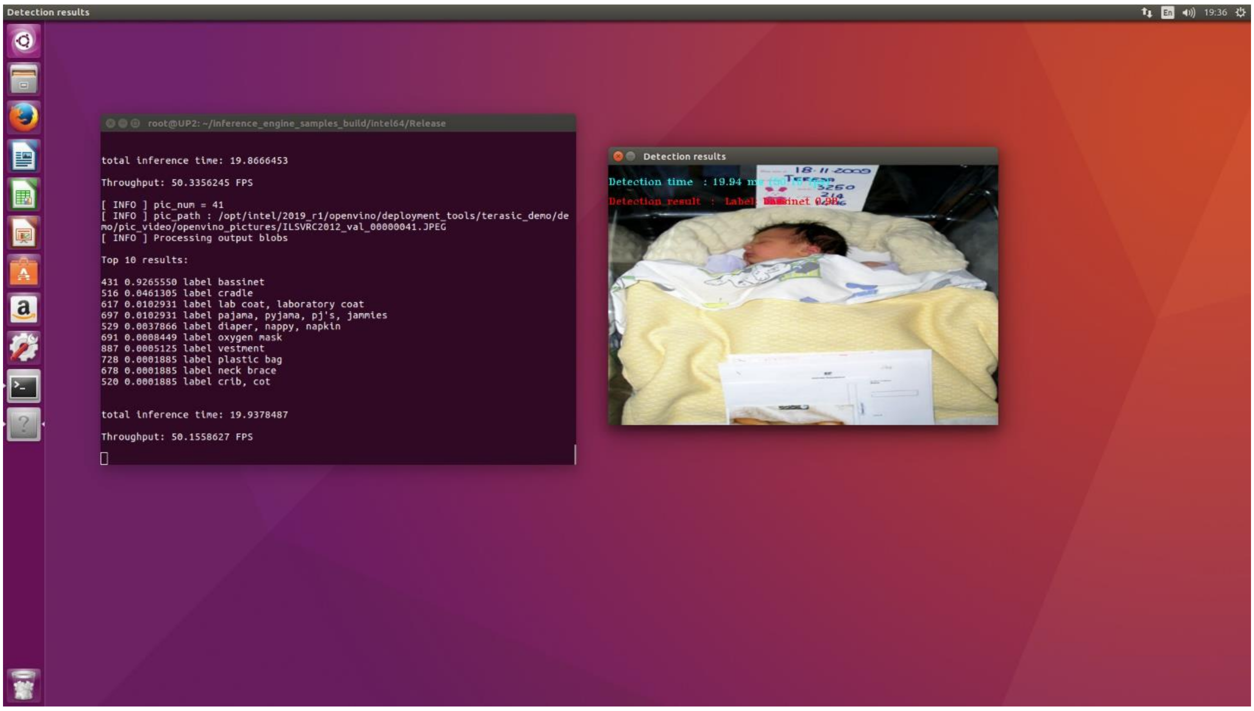


```
interactive_face_detection_demo  validation_app
lenet_network_graph_builder
root@UP2:~/inference_engine_samples_build# cd intel64/Release/
root@UP2:~/inference_engine_samples_build/intel64/Release# ls
classification_sample            my_demo
classification_sample_for_c5p    object_detection_demo_ssd_async
human_pose_estimation_demo       security_barrier_camera_demo
interactive_face_detection_demo  text_detection_demo
lib
root@UP2:~/inference_engine_samples_build/intel64/Release# ./my_demo -d "HETERO:
FPGA,CPU" -i /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/demo/pic_
video/openvino_pictures -m /opt/intel/2019_r1/openvino/deployment_tools/terasic_
demo/demo/ir/FP16/squeezenet1.1/squeezenet1.1.xml 
```

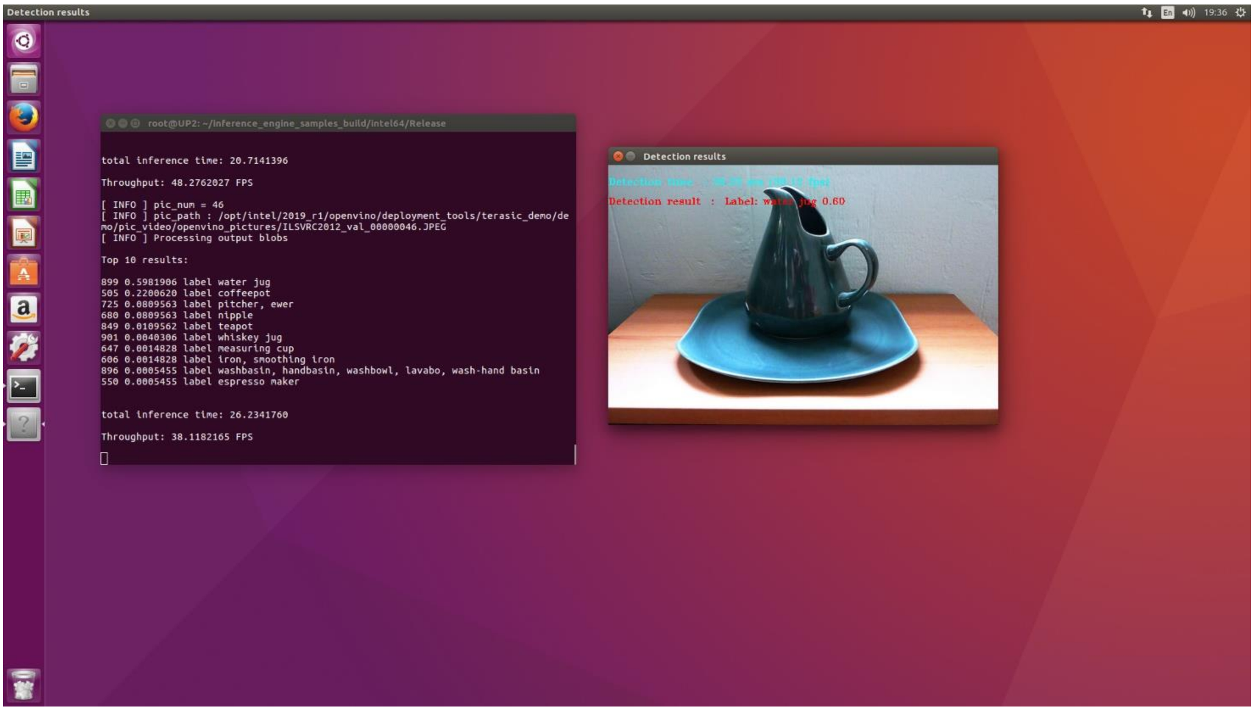
14. The results are as below:


Detection results
root@UP2: ~/inference_engine_samples_build/intel64/Release
Detection results
Detection result : Label: otter

```
total inference time: 18.6891556

Throughput: 53.5069654 FPS

[ INFO ] pic_num = 33
[ INFO ] pic_path : /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/de
mo/pic_video/openvino_pictures/ILSVRC2012_val_00000033.JPEG
[ INFO ] Processing output blobs

Top 10 results:

360 0.8927959 label otter
150 0.0269601 label sea lion
50 0.0163521 label American alligator, Alligator mississipiensis
140 0.0060156 label red-backed sandpiper, dunlin, Erolia alpina
397 0.0060156 label puffer, pufferfish, blowfish, globefish
289 0.0036487 label snow leopard, ounce, Panthera uncia
112 0.0036487 label conch
357 0.0036487 label mink
356 0.0036487 label weasel
336 0.0022130 label marmot


total inference time: 19.1478208

Throughput: 52.2252643 FPS

```


Detection results
root@UP2: ~/inference_engine_samples_build/intel64/Release
Detection results
Detection time : 19.94 ms
Detection result : Label: inet 0.98

```
total inference time: 19.8666453

Throughput: 50.3356245 FPS

[ INFO ] pic_num = 41
[ INFO ] pic_path : /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/de
mo/pic_video/openvino_pictures/ILSVRC2012_val_00000041.JPEG
[ INFO ] Processing output blobs

Top 10 results:

431 0.9265550 label bassinet
516 0.0461305 label cradle
617 0.0102931 label lab coat, laboratory coat
697 0.0102931 label pajama, pyjama, pj's, jammies
529 0.0037866 label diaper, nappy, napkin
691 0.0008449 label oxygen mask
887 0.0005125 label vestment
728 0.0001885 label plastic bag
678 0.0001885 label neck brace
520 0.0001885 label crib, cot


total inference time: 19.9378487

Throughput: 50.1558627 FPS

```



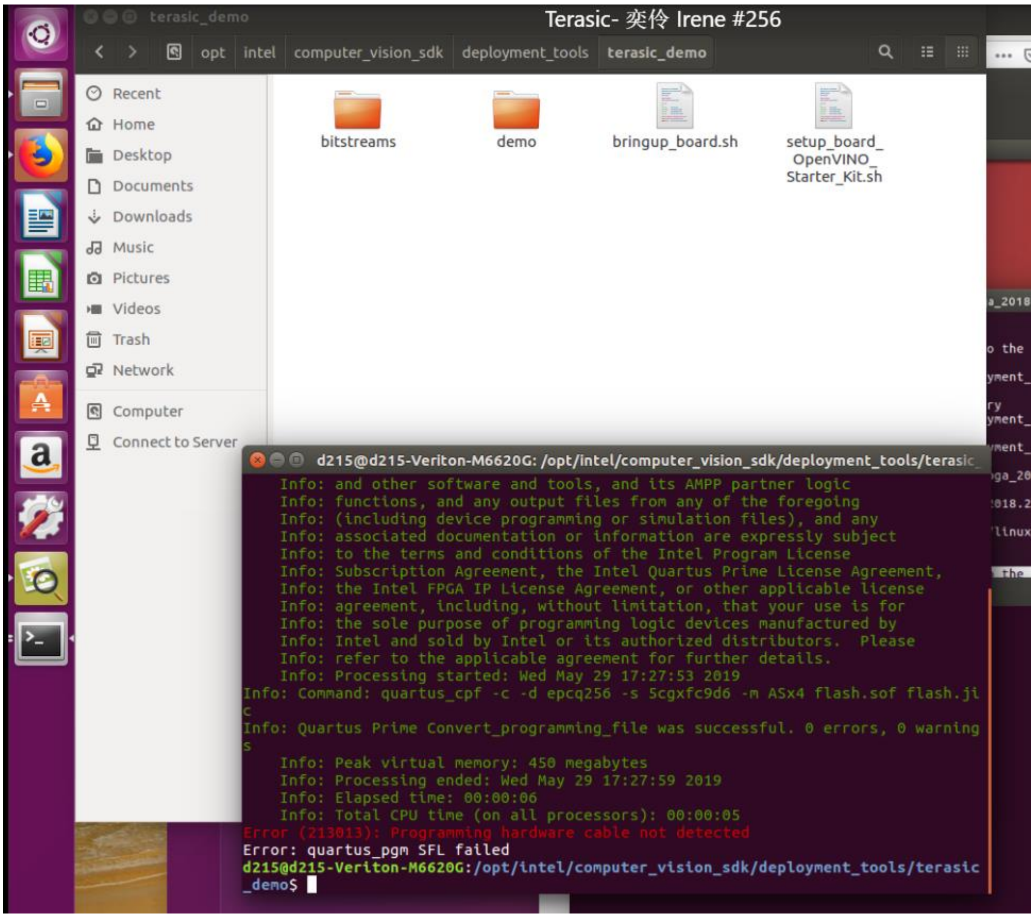
```
root@UP2: ~/inference_engine_samples_build/intel64/Release

total inference time: 20.7141396

Throughput: 48.2762027 FPS

[ INFO ] pic_num = 46
[ INFO ] pic_path : /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/de
mo/pic_video/openvino_pictures/ILSVRC2012_val_00000046.JPEG
[ INFO ] Processing output blobs

Top 10 results:

899 0.5981906 label water jug
505 0.2200620 label coffeepot
725 0.0809563 label pitcher, ewer
680 0.0809563 label nipple
849 0.0109562 label teapot
901 0.0040306 label whiskey jug
647 0.0014828 label measuring cup
606 0.0014828 label iron, smoothing iron
896 0.0005455 label washbasin, handbasin, washbowl, lavabo, wash-hand basin
550 0.0005455 label espresso maker


total inference time: 26.2341760

Throughput: 38.1182165 FPS

```

If you test all of these demos successfully, you will find the result is the same as demo 07_classification_pic_loop.sh.

Appendix

FAQ

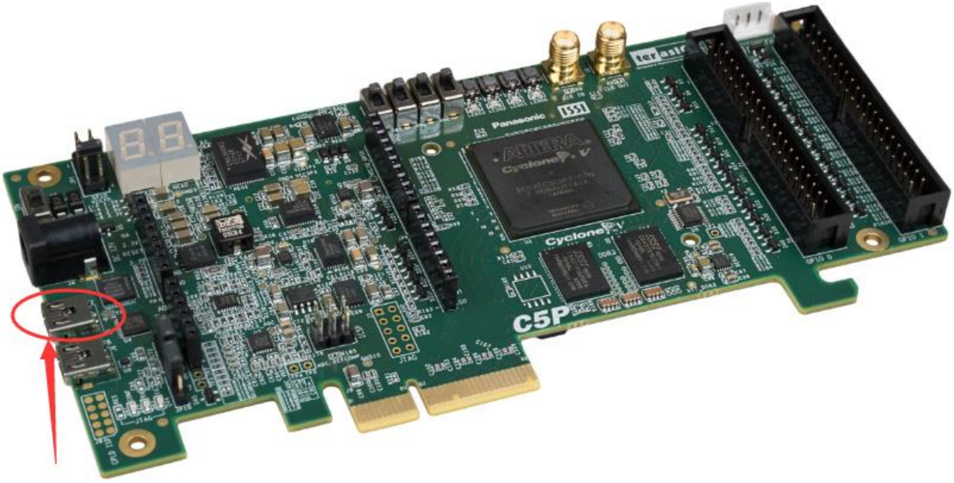
1. **Q**: When executing ./bringup_board.sh command, it reports “Error (213013): Programming hardware cable not detected Error: quartus_pgm SFL failed” as the picture below:



```
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Wed May 29 17:27:53 2019
Info: Command: quartus_cpf -c -d epcq256 -s 5cgxfc9d6 -m ASx4 flash.sof flash.ji
c
Info: Quartus Prime Convert_programming_file was successful. 0 errors, 0 warning
s
    Info: Peak virtual memory: 450 megabytes
    Info: Processing ended: Wed May 29 17:27:59 2019
    Info: Elapsed time: 00:00:06
    Info: Total CPU time (on all processors): 00:00:05
Error (213013): Programming hardware cable not detected
Error: quartus_pgm SFL failed
d215@d215-Veriton-M6620G:/opt/intel/computer_vision_sdk/deployment_tools/terasic
_demo$ 
```

A: Please make sure the USB cable is connected to the UB2 (USB Blaster II) port correctly (the UB2 port is shown in the following picture).

Using the command “lsusb” to double check if the PC detects the hardware or not.



2. **Q:** Which Nets can the OpenVINO Model Optimizer support?

   A: Please refer to Intel website:

   https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html for the more information.

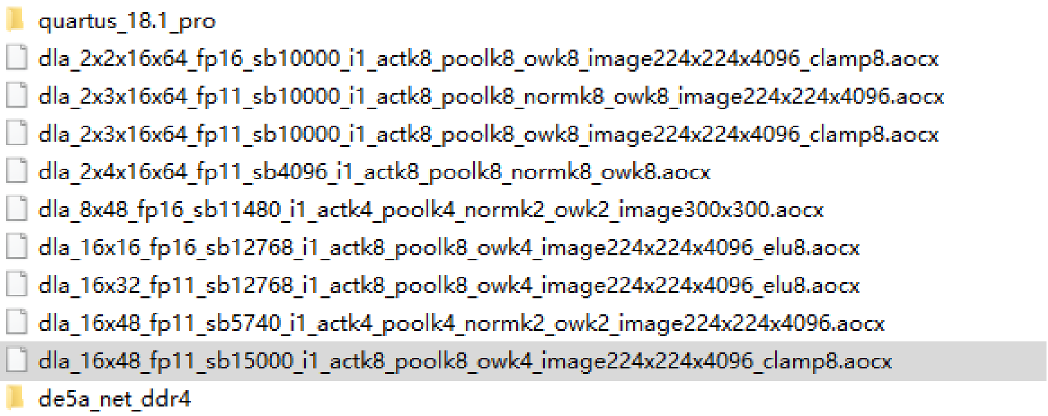
   So far, we have tested GoogleNet V2 and ResNet.

3. **Q:** How to program different aocx file for one FPGA board?

   A: Please refer to below command:

   ```
   aocl program acl0 <terasic_demo path>/bitstreams/<board name>/<aocx name>
   ```

   User needs to enter the right **terasic_demo path**, **board name** and **aocx name** to the command, for example, we use DE5a-Net-DDR4 board, we provide below aocx files for DE5a-Net-DDR4:



quartus_18.1_pro
dla_2x2x16x64_fp16_sb10000_i1_actk8_poolk8_owk8_image224x224x4096_clamp8.aocx
dla_2x3x16x64_fp11_sb10000_i1_actk8_poolk8_normk8_owk8_image224x224x4096.aocx
dla_2x3x16x64_fp11_sb10000_i1_actk8_poolk8_owk8_image224x224x4096_clamp8.aocx
dla_2x4x16x64_fp11_sb4096_i1_actk8_poolk8_normk8_owk8.aocx
dla_8x48_fp16_sb11480_i1_actk4_poolk4_normk2_owk2_image300x300.aocx
dla_16x16_fp16_sb12768_i1_actk8_poolk8_owk4_image224x224x4096_elu8.aocx
dla_16x32_fp11_sb12768_i1_actk8_poolk8_owk4_image224x224x4096_elu8.aocx
dla_16x48_fp11_sb5740_i1_actk4_poolk4_normk2_owk2_image224x224x4096.aocx
dla_16x48_fp11_sb15000_i1_actk8_poolk8_owk4_image224x224x4096_clamp8.aocx
de5a_net_ddr4

   Below is the command that we use to program one aocx file:

   ```
   aocl program acl0 /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/bitstreams/de5a_net_ddr4/dla_16x48_fp11_sb15000_i1_actk8_poolk8_owk4_image224x224x4096_clamp8.aocx
   ```

Contact Terasic

Users can refer to below contacts for Terasic technical support and products information:

Tel：+886-3-575-0880

Email: support@terasic.com / sales@terasic.com

Site：http://www.terasic.com

Address：9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan

Revision History

Version	Changes Log
V1.0	Initial Version
V2.0	Modified for 2019R1
V2.1	Update figures and operation describe for OSKS
V2.2	Following this guide to test DE5a-Net-DDR4 OpenVINO 2019R1

Copyright Statement