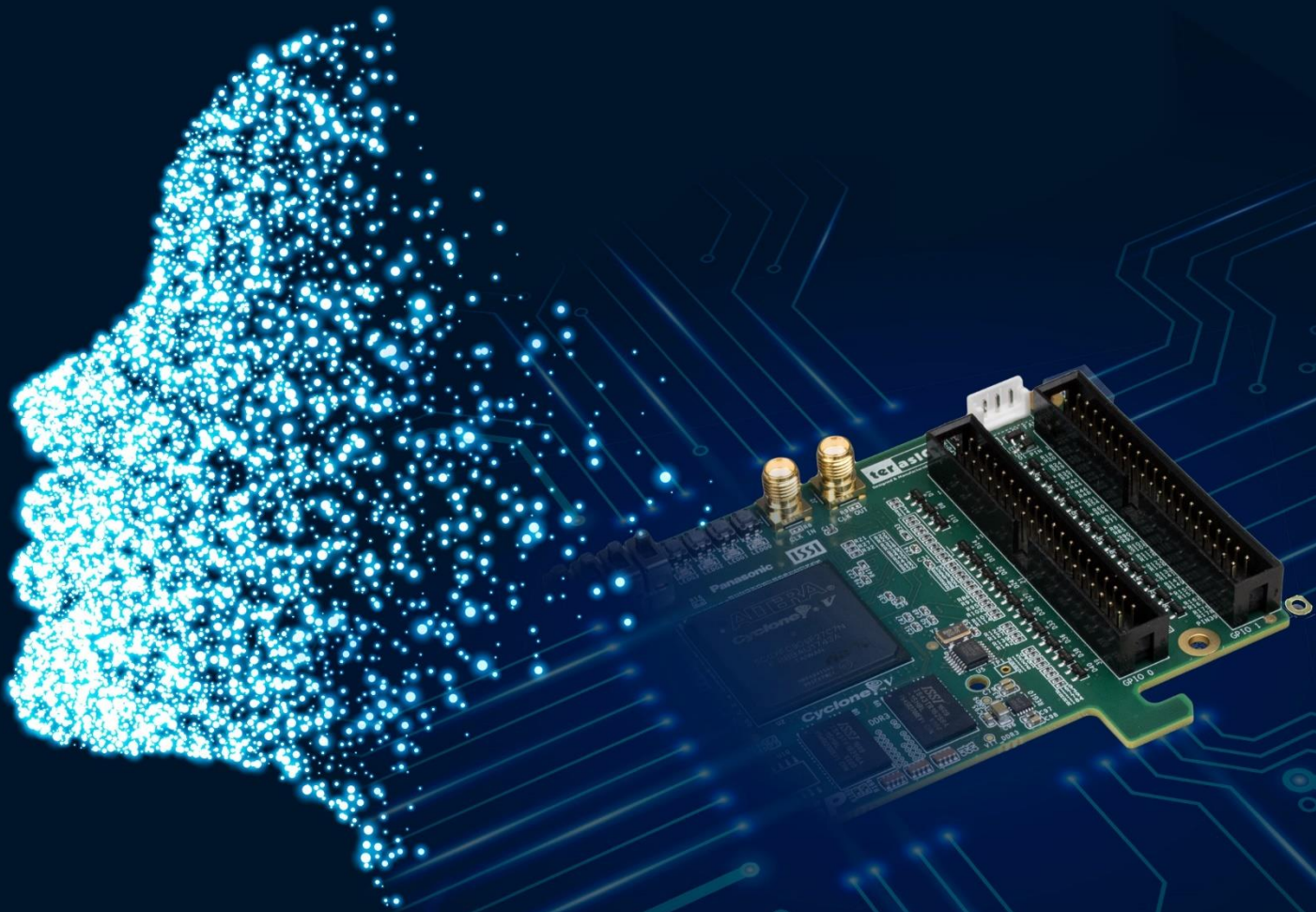


# *OpenVINO* Development Guide



# TABLE OF CONTENTS

<b>Chapter 1 OpenVINO Toolkit &amp; Process Introduction .....</b>	<b>3</b>
1.1 About the Guide .....	3
1.2 OpenVINO Toolkit Features .....	3
1.3 What's Inside OpenVINO Toolkit .....	3
1.4 OpenVINO Workflow .....	4
1.5 Model Optimizer .....	5
1.6 Inference Engine.....	6
<b>Chapter 2 Run the DEMO on the Starter Platform for OpenVINO™ Toolkit .....</b>	<b>8</b>
2.1 Introduction .....	8
2.2 Execute the Demo .....	9
<b>Chapter 3 Starter Platform for OpenVINO™ Toolkit Lab .....</b>	<b>21</b>
3.1 Verify the Environment of the Experiment.....	21
3.2 Lab 1. How to use the Model Optimizer to transform the Model? .....	25
3.3 Lab 2. How to compile an Inference Engine app? .....	31
3.4 Lab 3. Execute the created application file, use the Inference Engine for the classification predication. ....	35
3.5 Advanced Experiment .....	41

## 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 [Starter Platform for OpenVINO™ Toolkit](#), 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 on the Starter Platform for OpenVINO™ Toolkit
- Starter Platform for OpenVINO™ Toolkit Lab

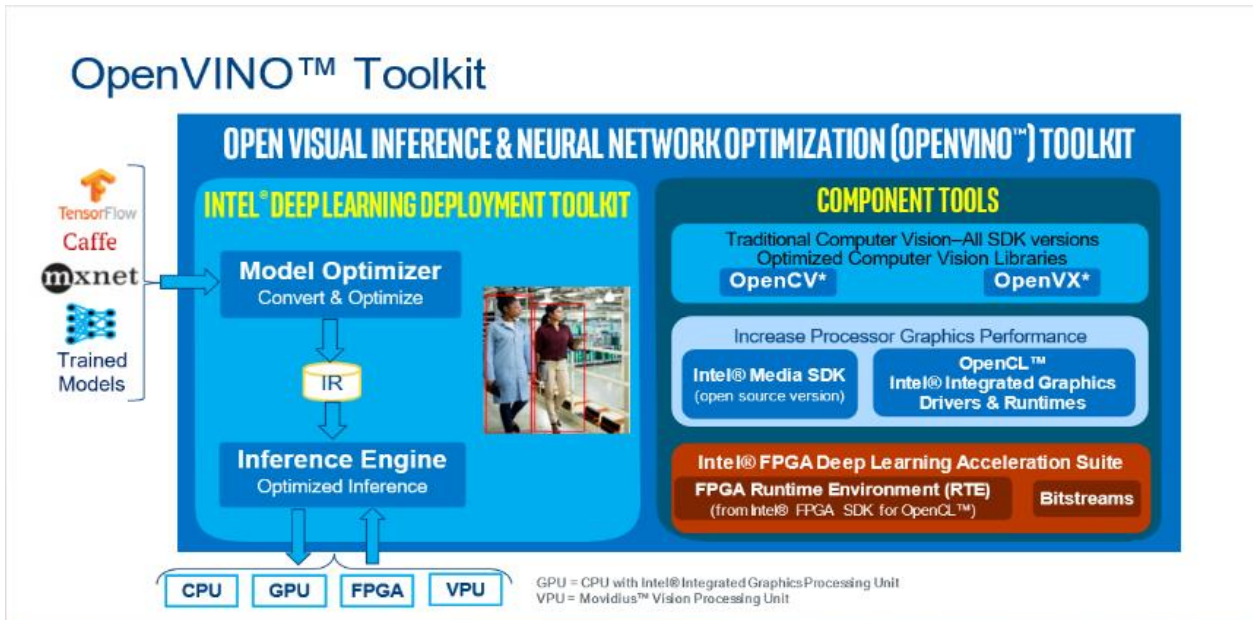
### 1.2 OpenVINO Toolkit Features

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.



The Toolkit includes:

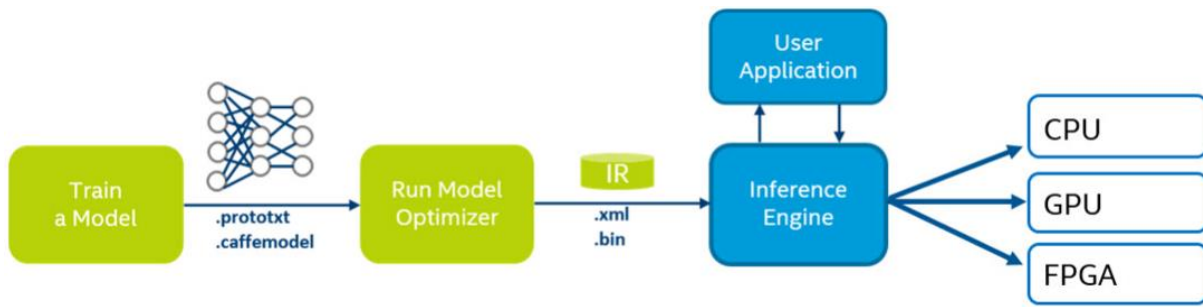
1. Deep Learning Deployment Toolkit, which comprises the following two components:
  - **Model Optimizer:** This Python\*-based command line tool imports trained models from popular deep learning frameworks such as Caffe, TensorFlow, and Apache MXNet\*. Input trained model, optimize topology, and convert it to an 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.



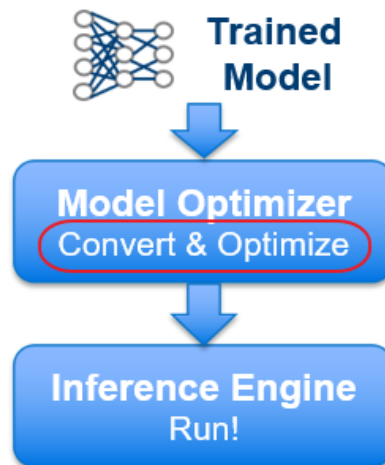


## 1.5 Model Optimizer

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.

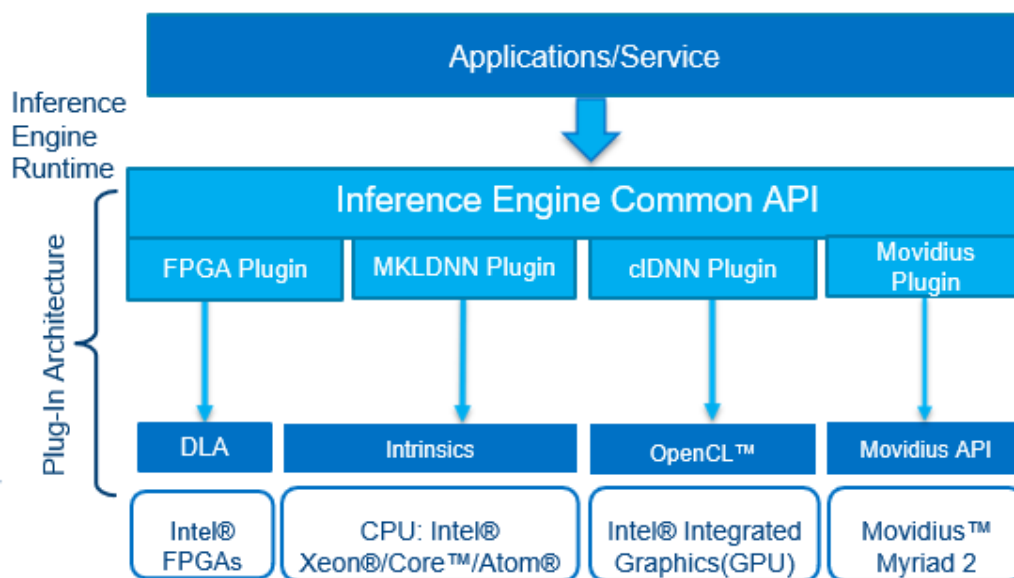
Many common layers exist across known frameworks and neural network topologies. Examples of these layers are Convolution, Pooling, and Activation. The Model Optimizer must be able to work with these layers to read the original model and produce the Intermediate Representation of a model, the layer list 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.

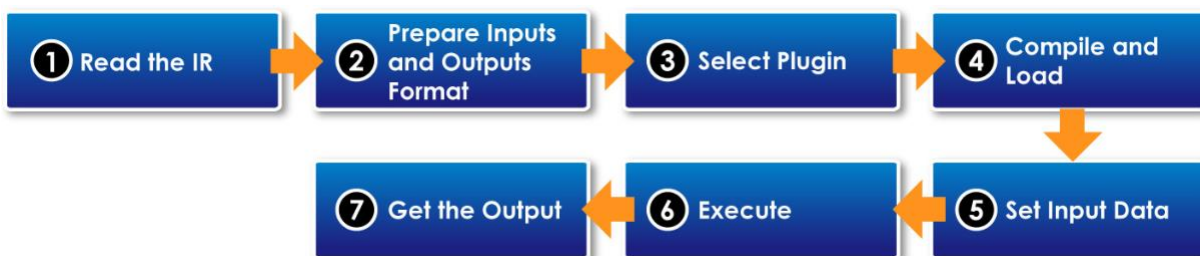


Each supported target device has a plugin and each plugin is a DLL/shared library. The Heterogeneous plugin lets you distribute a calculation workload across devices. One needs to make sure those libraries are specified in the path of host PC or in the place pointed to the plugin loader. Additionally, 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::CNNNetReader` class 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 output 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](#).

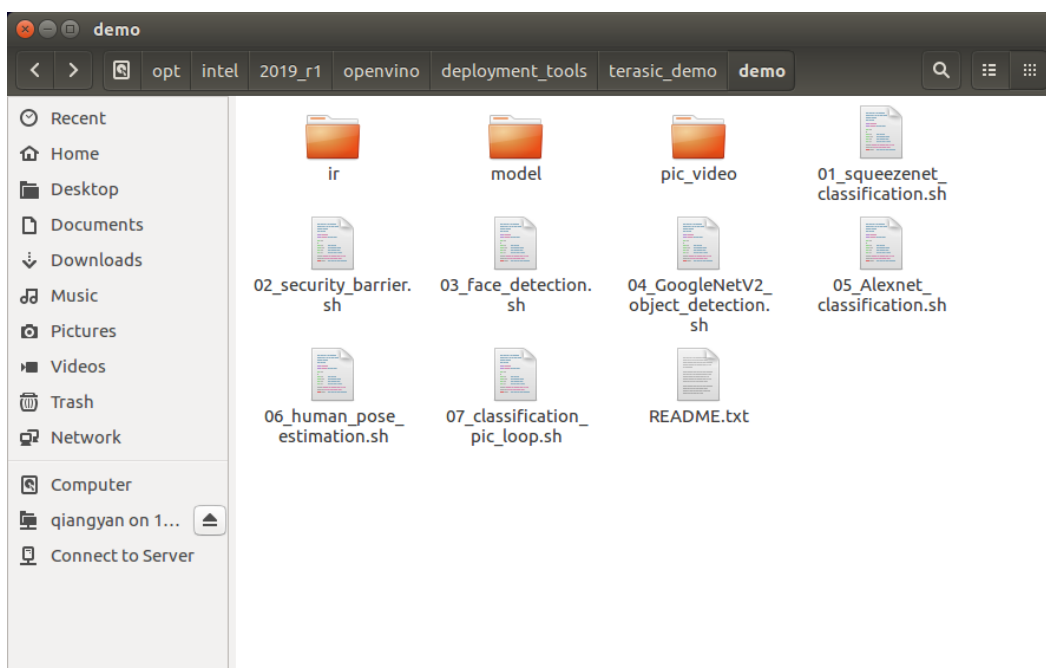
# Chapter 2

## Run the DEMO on the Starter Platform for OpenVINO™ Toolkit

This chapter describes how to run the demo on the Starter Platform for OpenVINO™ Toolkit 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.

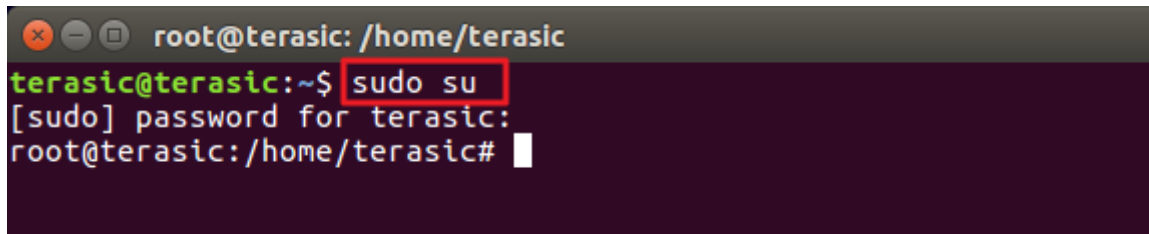
- The model generated under FP16 folder is used for FPGA
- The model generated under FP32 folder is used for CPU

## 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.

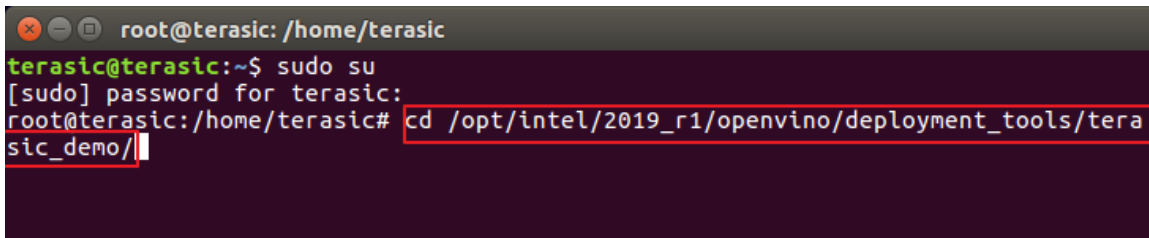
(login account: here, our username is terasic, password is terasic)

1. Right click on the desktop to open the Terminal, enter command "`sudo su`" to change user to root Super User, enter password "`terasic`"



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

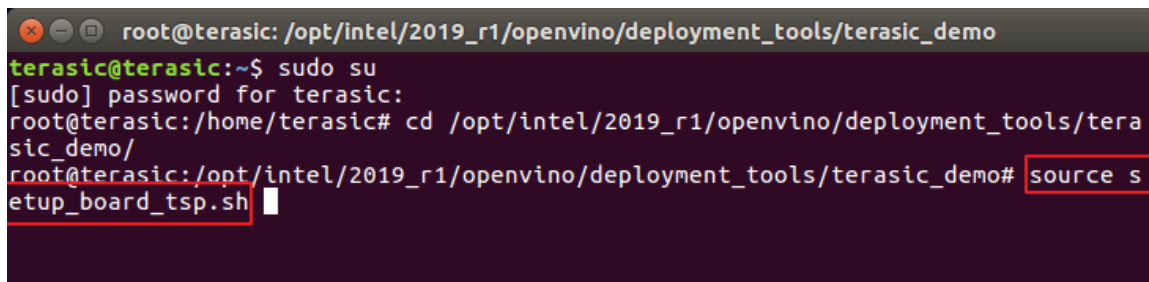
2. Enter "`cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo`" to switch to terasic\_demo path.



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

3. Source `setup_board_tsp.sh`

Note: User needs to setup the corresponding `.sh` file for the FPGA board, for example, source `setup_board_tsp.sh` for TSP GT Edition board or TSP GX Edition board, source `setup_board_de5a_net_ddr4.sh` for DE5a-Net-DDR4 board.



```
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
terasic@terasic:~$ sudo su
[sudo] password for terasic:
root@terasic: /home/terasic# cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# source setup_board_tsp.sh
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo#
```

- Type “y” to install.

```

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# source setup_board_tsp.sh
[setupvars.sh] OpenVINO environment initialized
INTELFPGAOCCLSDKROOT is set to /opt/altera/aocl-pro-rte/aclrte-linux64. Using that.

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/ask. 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_PATH
Adding /opt/altera/aocl-pro-rte/aclrte-linux64/board/tsp/linux64/lib to LD_LIBRARY_PATH
Do you want to install /opt/altera/aocl-pro-rte/aclrte-linux64/board/tsp? [y/n]
y

```

- Enter *aocl diagnose* to check the environment, the “DIAGNOSTIC\_PASSED” represents the environment setup is successful.

```

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
RY_PATH
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# aocl diagnose
-----
Warning:
No devices attached for package:
/opt/altera/aocl-pro-rte/aclrte-linux64/board/tsp
-----

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 machine ...

Device Name      Status      Information
-----
aocl0            Passed     Cyclone V HPC Reference Platform
                PCIe dev_id = D800, bus:slot.func = 01:00.00, at Gen 2 with 1 lanes

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

DIAGNOSTIC_PASSED
-----

```

- Use an USB cable to connect the TSP USB Blaster II connector and PC. Enter “*aocl program acl0 \$DLA\_AOCX\_GT*” (note: if user tests the OpenVINO examples for GX edition, please enter “*aocl program acl0 \$DLA\_AOCX\_GX*” command) to program an .aocx file (click [here](#) to know how to program an .aocx file) to the FPGA.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
S
Call "aocl diagnose all" to run diagnose for all devices
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo#
aocl program acl0 $DLA_AOCX_GT
aocl program: Running program from /opt/altera/aocl-pro-rte/aclrte-linux
64/board/tsp/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 Agree-
ent,
Info: the Intel FPGA IP License Agreement, or other applicable licen-
se
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. Pleas-
e
Info: refer to the applicable agreement for further details.
Info: Processing started: Tue Apr 21 16:37:13 2020
Info: Command: quartus_pgm -c 1 -m jtag -o P;reprogram_temp.sof@1
Info (213045): Using programming cable "C5P [1-3]"
Info (213011): Using programming file reprogram_temp.sof with checksum 0
x253CDA26 for device 5CGTFD9D5F27@1
Info (209060): Started Programmer operation at Tue Apr 21 16:37:28 2020
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 Tue Apr 21 16:37:33 2020
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 479 megabytes
Info: Processing ended: Tue Apr 21 16:37:33 2020
Info: Elapsed time: 00:00:20
Info: Total CPU time (on all processors): 00:00:11
Program succeed.
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo#

```

7. Switch to the demo path.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
ent,
Info: the Intel FPGA IP License Agreement, or other applicable licen
se
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. Pleas
e
Info: refer to the applicable agreement for further details.
Info: Processing started: Tue Apr 21 16:37:13 2020
Info: Command: quartus_pgm -c 1 -m jtag -o P;reprogram_temp.sof@1
Info (213045): Using programming cable "C5P [1-3]"
Info (213011): Using programming file reprogram_temp.sof with checksum 0
x253CDA26 for device 5CGTFD9D5F27@1
Info (209060): Started Programmer operation at Tue Apr 21 16:37:28 2020
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 Tue Apr 21 16:37:33 2020
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 479 megabytes
Info: Processing ended: Tue Apr 21 16:37:33 2020
Info: Elapsed time: 00:00:20
Info: Total CPU time (on all processors): 00:00:11
Program succeed.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo#
cd 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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
se
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. Pleas
e
Info: refer to the applicable agreement for further details.
Info: Processing started: Tue Apr 21 16:37:13 2020
Info: Command: quartus_pgm -c 1 -m jtag -o P;reprogram_temp.sof@1
Info (213045): Using programming cable "C5P [1-3]"
Info (213011): Using programming file reprogram_temp.sof with checksum 0
x253CDA26 for device 5CGTFD9D5F27@1
Info (209060): Started Programmer operation at Tue Apr 21 16:37:28 2020
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 Tue Apr 21 16:37:33 2020
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 479 megabytes
Info: Processing ended: Tue Apr 21 16:37:33 2020
Info: Elapsed time: 00:00:20
Info: Total CPU time (on all processors): 00:00:11
Program succeed.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo#
cd demo/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/d
emo# ./01_squeezenet_classification.sh fpga

```

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



```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
#####
Run Inference Engine classification sample
Run ./classification_sample -d HETERO:FPGA,CPU -i /opt/intel/2019_r1/ope
nvino/deployment_tools/terasic_demo/demo/pic_video/car.png -m /opt/intel
/2019_r1/opencvino/deployment_tools/terasic_demo/demo/ir/FP16/squeezenet1
.1/squeezenet1.1.xml[ INFO ] InferenceEngine:
    API version ..... 1.6
    Build ..... custom_releases/2019/R1_c9b66a26e4d65bb
986bb740e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
[ INFO ] Files were added: 1
[ INFO ]     /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/d
emo/pic_video/car.png
[ INFO ] Loading plugin

    API version ..... 1.6
    Build ..... heteroPlugin
    Description ..... heteroPlugin
[ INFO ] Loading network files:
    /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/i
r/FP16/squeezenet1.1/squeezenet1.1.xml
    /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/i
r/FP16/squeezenet1.1/squeezenet1.1.bin
[ INFO ] Preparing input blobs
[ WARNING ] Image is resized from (787, 259) to (227, 227)

```

3) It prints out the top 10 results.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo
[ INFO ] Preparing output blobs
[ INFO ] Loading model to the plugin
[ INFO ] Starting inference (1 iterations)
[ INFO ] Processing output blobs

Top 10 results:

Image /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_video/car.png

classid probability label
-----
817 0.8363336 sports car, sport car
511 0.0946490 convertible
479 0.0419133 car wheel
751 0.0091072 racer, race car, racing car
436 0.0068162 beach wagon, station wagon, wagon, estate car, beach waggon, statio
n waggon, waggon
656 0.0037564 minivan
586 0.0025741 half track
717 0.0016069 pickup, pickup truck
864 0.0012027 tow truck, tow car, wrecker
581 0.0005882 grille, radiator grille

total inference time: 186.9174242
Average running time of one iteration: 186.9174242 ms

Throughput: 5.3499560 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@terasic: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/de
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: 85.2543712
Average running time of one iteration: 85.2543712 ms

Throughput: 11.7296038 FPS

[ INFO ] Execution successful

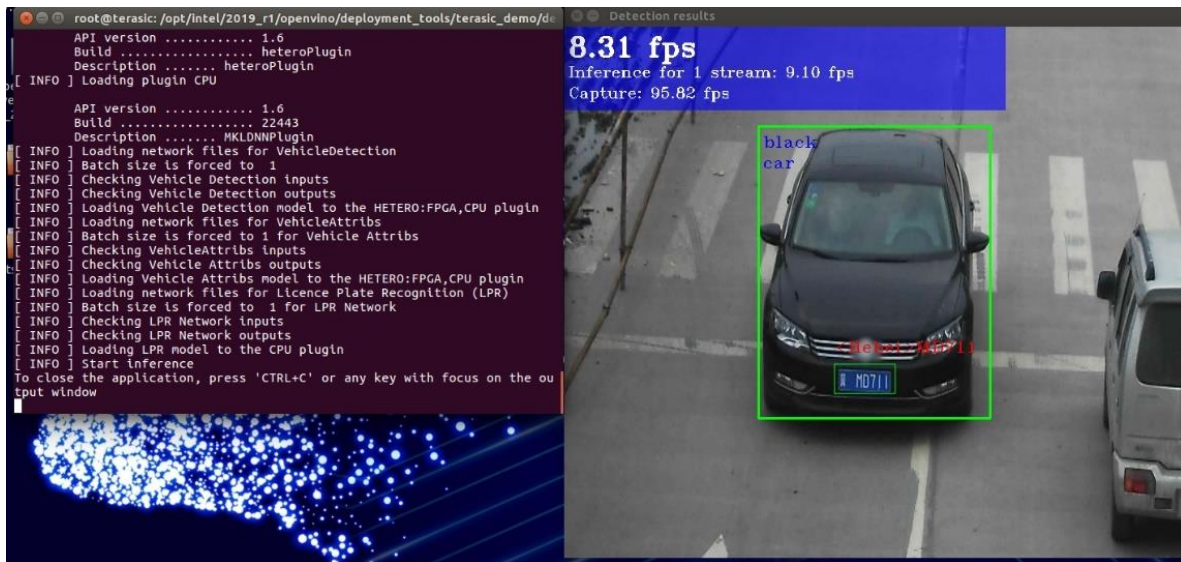
#####

Demo completed successfully.

root@terasic: /opt/intel/2019_r1/openvino/deployment_tools/terasic_demo/d
emo# ./02_security_barrier.sh fpga

```

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



### ■ 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.

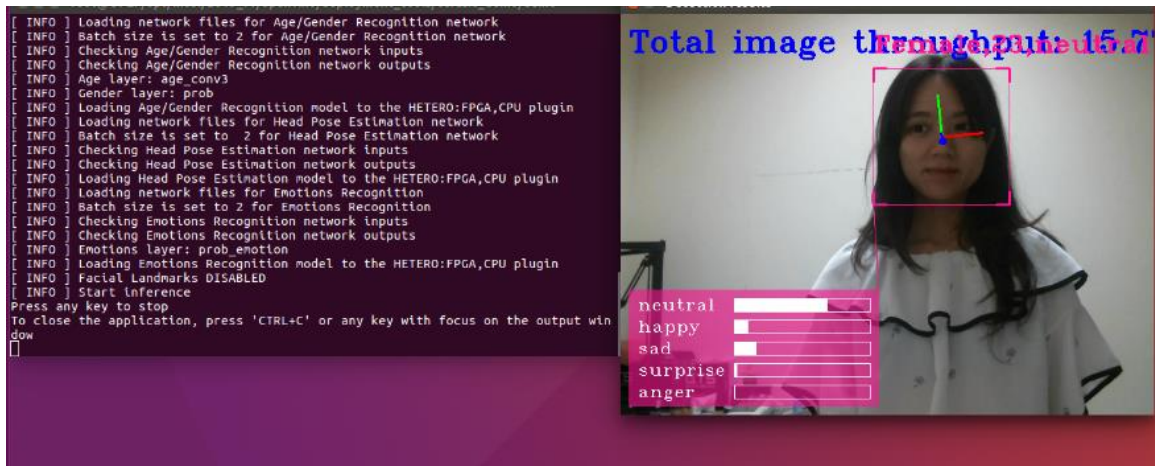
```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
Build ..... heteroPlugin
Description ..... heteroPlugin
[ 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 ou
tput window
^Croot@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
emo# ./03 face detection.sh fpga

```

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



#### ■ 04\_GoogleNetV2\_object\_detection

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

1	aeroplane
2	bicycle
3	bird
4	boat
5	bottle
6	bus
7	car
8	cat
9	chair
10	cow
11	diningtable
12	dog
13	horse
14	motorbike
15	person
16	pottedplant
17	sheep
18	sofa
19	train
20	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

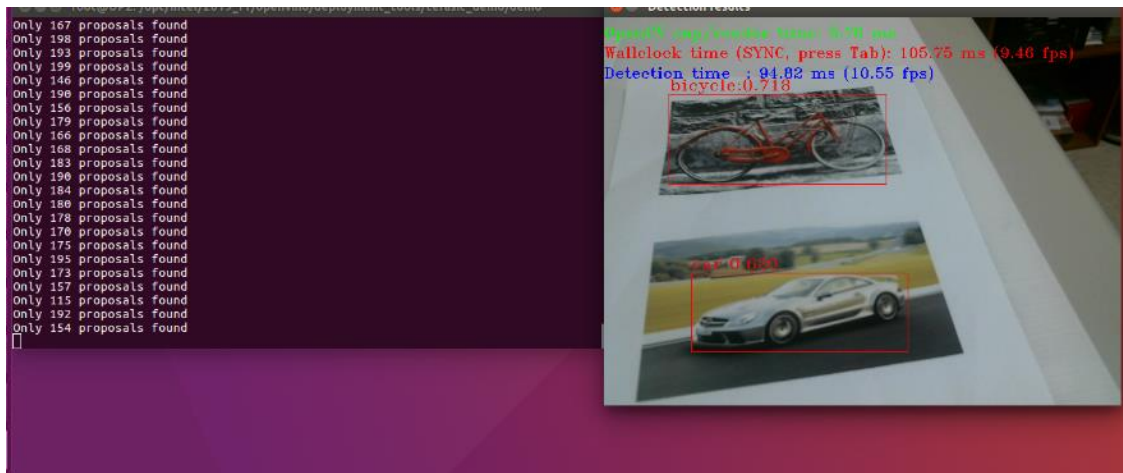
```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
[ 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 plu
gin
[ 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 plugi
n
[ 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 plugi
n
[ 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 ou
tput window
^Croot@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
emo# ./04_GoogleNetV2_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.





## ■ 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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
Build ..... custom_releases/2019/R1_c9b66a26e4d65bb
986bb740e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
[ 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 ou
tput window
^C

#####

Demo completed successfully.

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/d
emo# ./05 Alexnet_classification.sh fpga

```

- 2) The results are shown in the figure below.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
_video/car.png

classid probability label
-----
479 0.7904695 n02974003 car wheel
436 0.0648857 n02814533 beach wagon, station wagon, wagon, estate
car, beach waggon, station waggon, waggon
511 0.0648857 n03100240 convertible
817 0.0393552 n04285008 sports car, sport car
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: 222.4984914
Average running time of one iteration: 222.4984914 ms

Throughput: 4.4944125 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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/de
car, beach waggon, station waggon, waggon
511 0.0648857 n03100240 convertible
817 0.0393552 n04285008 sports car, sport car
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: 222.4984914
Average running time of one iteration: 222.4984914 ms

Throughput: 4.4944125 FPS

[ INFO ] Execution successful

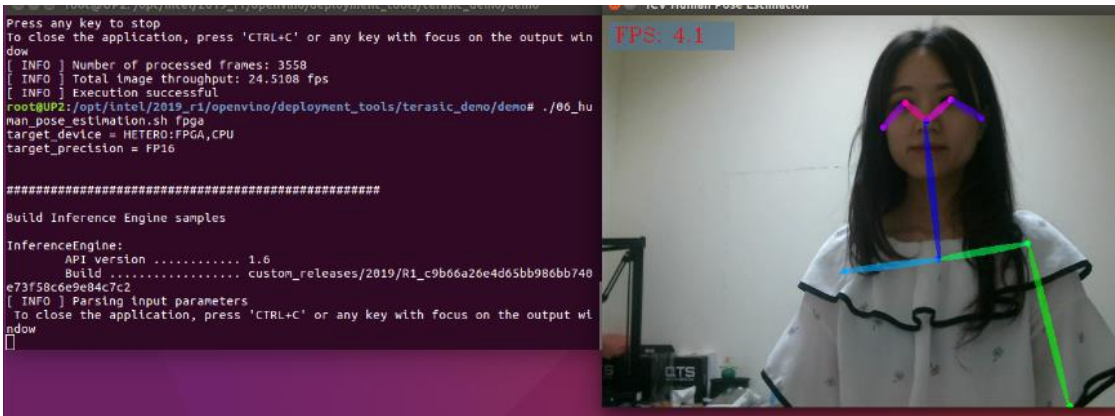
#####

Demo completed successfully.

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/d
emo# ./06 human pose estimation.sh fpga

```

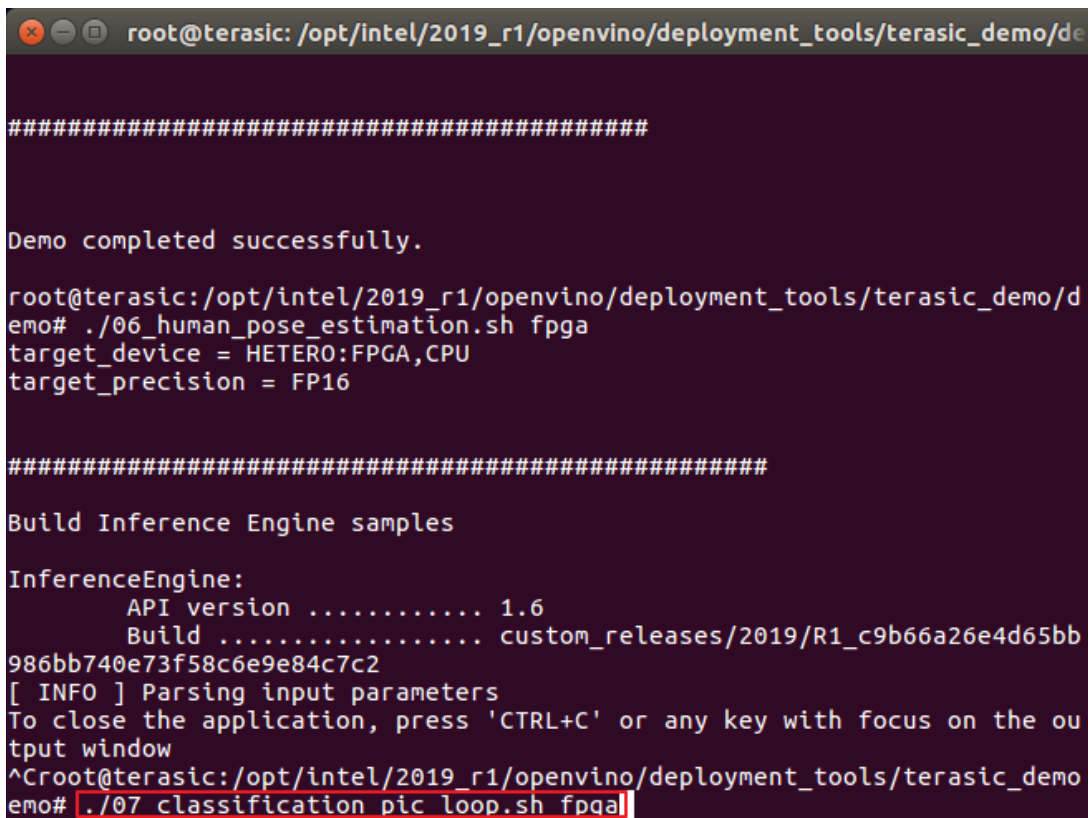
3) The result is shown in the figure below. Enter Ctrl+C to close the Application.



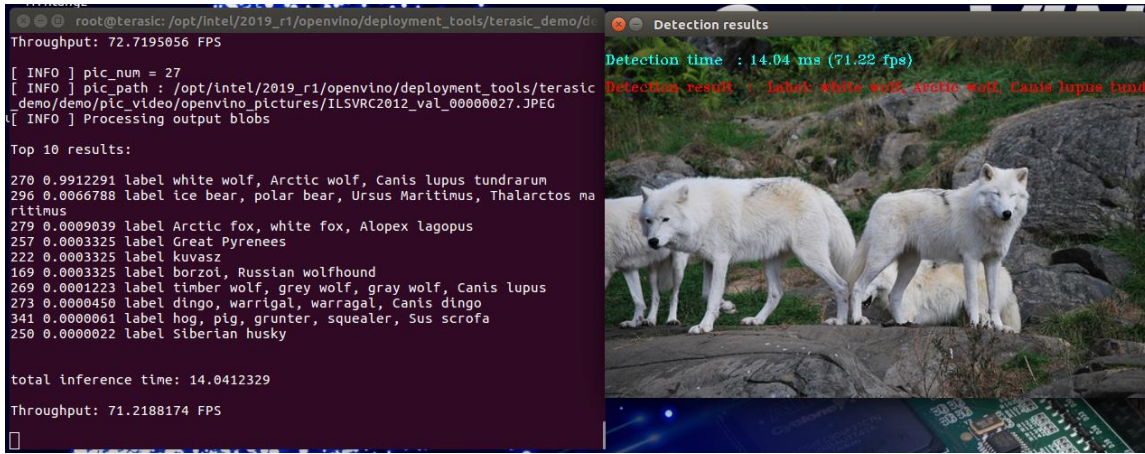
■ **07\_classification\_pic\_loop**

This demo bases demo 01\_squeezenet\_classification, and adds more pictures to run in a loop.

1) Execute “./07\_classification\_pic\_loop.sh fpga” to run the demo with FPGA.



2) The result is shown in the figure below.



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 if user executes the demo on the DE5a-Net-DDR4 instead of TSP.



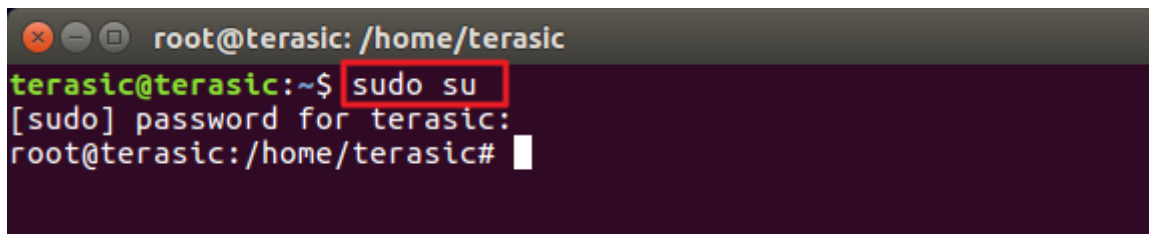
## Starter Platform for OpenVINO™ Toolkit Lab

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

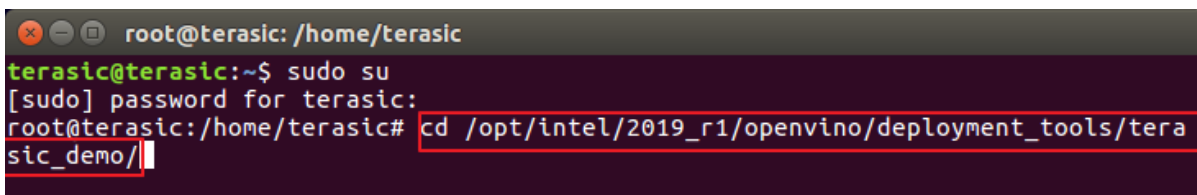
This section will show user how to verify the environment of the experiment by running the demo “02\_security\_barrier.sh” with CPU which is provided in terasic demo.

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.



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

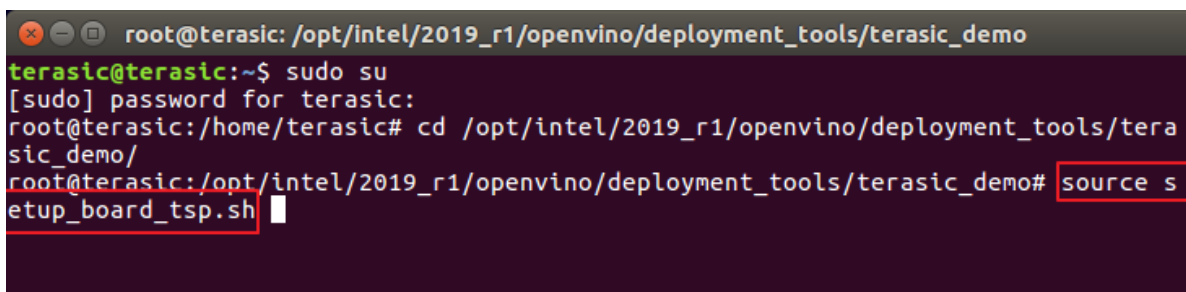
3. Enter “*cd /opt/intel/2019\_r1/openvino/deployment\_tools/terasic\_demo*” to switch the path.



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

4. Enter “*source setup\_board\_tsp.sh*”.

Note: User needs to setup the corresponding .sh file for the FPGA board, for example, source *setup\_board\_tsp.sh* for TSP GT Edition board or TSP GX Edition board, source *setup\_board\_de5a\_net\_ddr4.sh* for DE5a-Net-DDR4 board.



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

5. Enter “y” for the driver installation.

```

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# source s
etup_board_tsp.sh
[setupvars.sh] OpenVINO environment initialized
INTELFPGAOCSDKROOT 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/tsp/linux64/lib to LD_LIBRA
RY_PATH
Do you want to install /opt/altera/aocl-pro-rte/aclrte-linux64/board/tsp? [y/n]
y

```

6. Enter `cd /root/inference_engine_samples_build/`

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo
terasic@terasic:~$ sudo su
[sudo] password for terasic:
root@terasic:/home/terasic# cd /opt/intel/2019_r1/opencvino/deployment_tools/tera
sic_demo
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# source s
etup_board_tsp.sh
[setupvars.sh] OpenVINO environment initialized
INTELFPGAOCSDKROOT 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/
tsp. 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/tsp/linux64/lib to LD_LIBRA
RY_PATH
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# cd /root
/inference_engine_samples_build/

```

7. Enter `rm -rf CMakeCache.txt`

```

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# cd /root
/inference_engine_samples_build/
root@terasic:~/inference_engine_samples_build# rm -rf CMakeCache.txt

```

8. Enter `cmake -DCMAKE_BUILD_TYPE=Release \ /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/`.

```

root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo# cd /root
/inference_engine_samples_build/
root@terasic:~/inference_engine_samples_build# rm -rf CMakeCache.txt
root@terasic:~/inference_engine_samples_build# cmake -DCMAKE_BUILD_TYPE=Release
> /opt/intel/2019_r1/opencvino/deployment tools/inference engine/samples/

```

Then you can see the following messages.

```

root@osk: ~/inference_engine_samples_build
File Edit View Search Terminal Help
-- SSE supported
-- SSE2 supported
-- SSE3 supported
-- SSE4.1 supported
-- SSE4.2 supported
-- SSE4a not supported
-- SSSE3 supported
-- SYSCALL supported
-- TBM not supported
-- XOP not supported
-- XSAVE supported
-- TBB include: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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@osk:~/inference_engine_samples_build#

```

9. Enter “*make -j8*”

```

root@terasic: ~/inference_engine_samples_build
-- SSSE3 supported
-- SYSCALL supported
-- TBM not supported
-- XOP not supported
-- XSAVE supported
-- TBB include: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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@terasic:~/inference_engine_samples_build# make -j8

```

10. Wait until the build process is finished.

```

root@terasic: ~/inference_engine_samples_build
pose-estimation-demo.dir/postprocess.cpp.o
[ 94%] Linking CXX executable ../intel64/Release/security_barrier_camera_demo
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/peak.cpp.o
[ 95%] Built target security_barrier_camera_demo
[ 95%] Building CXX object smart_classroom_demo/CMakeFiles/smart_classroom_demo.
dir/main.cpp.o
[ 96%] Building CXX object pedestrian_tracker_demo/CMakeFiles/pedestrian_tracker
_demo.dir/main.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/postprocessor.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/main.cpp.o
[ 98%] Linking CXX executable ../intel64/Release/calibration_tool
[ 98%] Built target calibration_tool
[ 98%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 98%] Built target pedestrian_tracker_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-human-est
imation-demo
[ 99%] Built target multi-channel-human-pose-estimation-demo
[100%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[100%] Built target smart_classroom_demo

```

11. Enter “`cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo`” to switch the workspace.

```

root@terasic: ~/inference_engine_samples_build
pose-estimation-demo.dir/postprocess.cpp.o
[ 94%] Linking CXX executable ../intel64/Release/security_barrier_camera_demo
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/peak.cpp.o
[ 95%] Built target security_barrier_camera_demo
[ 95%] Building CXX object smart_classroom_demo/CMakeFiles/smart_classroom_demo.
dir/main.cpp.o
[ 96%] Building CXX object pedestrian_tracker_demo/CMakeFiles/pedestrian_tracker
_demo.dir/main.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/postprocessor.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-
pose-estimation-demo.dir/main.cpp.o
[ 98%] Linking CXX executable ../intel64/Release/calibration_tool
[ 98%] Built target calibration_tool
[ 98%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 98%] Built target pedestrian_tracker_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-human-est
imation-demo
[ 99%] Built target multi-channel-human-pose-estimation-demo
[100%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[100%] Built target smart_classroom_demo
root@terasic:~/inference_engine_samples_build# cd /opt/intel/2019_r1/opencvino/de
ployment_tools/terasic_demo/demo

```

12. Enter “`./02_security_barrier.sh fpga`”.

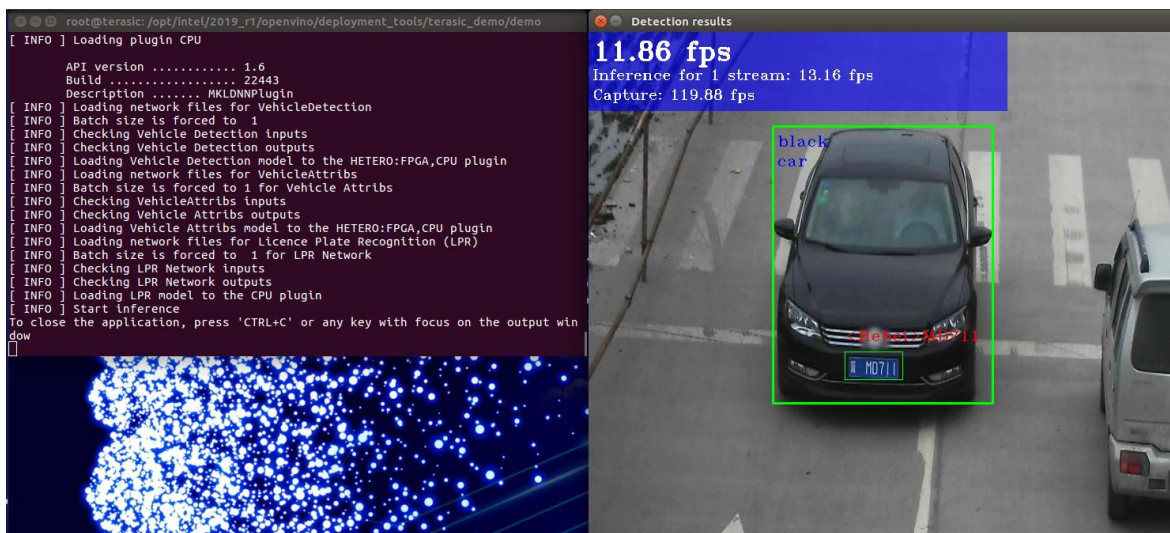


```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo
[ 95%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-pose-estimation-demo.dir/peak.cpp.o
[ 95%] Built target security_barrier_camera_demo
[ 95%] Building CXX object smart_classroom_demo/CMakeFiles/smart_classroom_demo.dir/main.cpp.o
[ 96%] Building CXX object pedestrian_tracker_demo/CMakeFiles/pedestrian_tracker_demo.dir/main.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-pose-estimation-demo.dir/postprocessor.cpp.o
[ 97%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-pose-estimation-demo.dir/main.cpp.o
[ 98%] Linking CXX executable ../intel64/Release/calibration_tool
[ 98%] Built target calibration_tool
[ 98%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 98%] Built target pedestrian_tracker_demo
[ 99%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-estimation-demo
[ 99%] Built target multi-channel-human-pose-estimation-demo
[100%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[100%] Built target smart_classroom_demo
root@terasic:~/inference_engine_samples_build# cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# ./02_security_barrier.sh fpga

```

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



## 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/opencvino/deployment_tools/terasic_demo/demo/model/caffe/`” to switch the path to the pre-downloaded Model folder.



```

root@osk: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe
File Edit View Search Terminal Help
[ INFO ] InferenceEngine:
[ INFO ] API version ..... 1.6
[ INFO ] Build ..... custom_releases/2019/R1_c9b66a26e4d65bb986bb740e73f58c6e9e84c7c2
[ 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/opencvino/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
[ INFO ] Display resolution: 1920x1080
[ INFO ] Loading plugin CPU
[ INFO ]
[ INFO ] API version ..... 1.6
[ INFO ] Build ..... 22443
[ INFO ] 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 window
^C
root@osk: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# cd /opt/intel/2019_r1/opencvino
/deployment_tools/terasic_demo/demo/model/caffe/
root@osk: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe#

```

2. Enter “ls” to see the folder information, which includes bvlc\_alexnet, squeezenet1.1, and SSD\_GoogleNetV2 models.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
[ 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 window
^C
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe#

```

3. Enter “cd squeezenet1.1” to select the corresponding Model folder.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model
[ 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@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd 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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model
[ 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@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd squeezenet1.1
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1#

```

5. Enter “cd ../../..” to go back to the demo folder.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
[ 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@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd squeezenet1.1
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# cd ../../../../

```

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

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo
[ 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@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd squeezenet1.1
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# cd ../../../../
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# mkd
ir my_ir

```

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



```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer
[ 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@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd squeezenet1.1
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# cd ../../..
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# mkd
ir my_ir
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# cd
/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer#

```

8. Enter “python3.5 mo\_caffe.py \
  
--input\_model /opt/intel/2019\_r1/opencvino/deployment\_tools/terasic\_demo/demo/\
  
model/caffe/squeezenet1.1/squeezenet1.1.caffemodel \
  
--output\_dir /opt/intel/2019\_r1/opencvino/deployment\_tools/terasic\_demo/demo/my\_ir \
  
--data\_type FP16”.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer
dow
^Croot@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# c
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# ls
bvlc_alexnet  squeezenet1.1  SSD_GoogleNetV2
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe# cd squeezenet1.1
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# ls
squeezenet1.1.caffemodel  squeezenet1.1.labels  squeezenet1.1.prototxt
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/mode
l/caffe/squeezenet1.1# cd ../../..
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# mkd
ir my_ir
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# cd
/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# pytho
n3.5 mo_caffe.py \
> --input_model /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/\
> model/caffe/squeezenet1.1/squeezenet1.1.caffemodel \
> --output_dir /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/my
ir \
> --data_type FP16

```

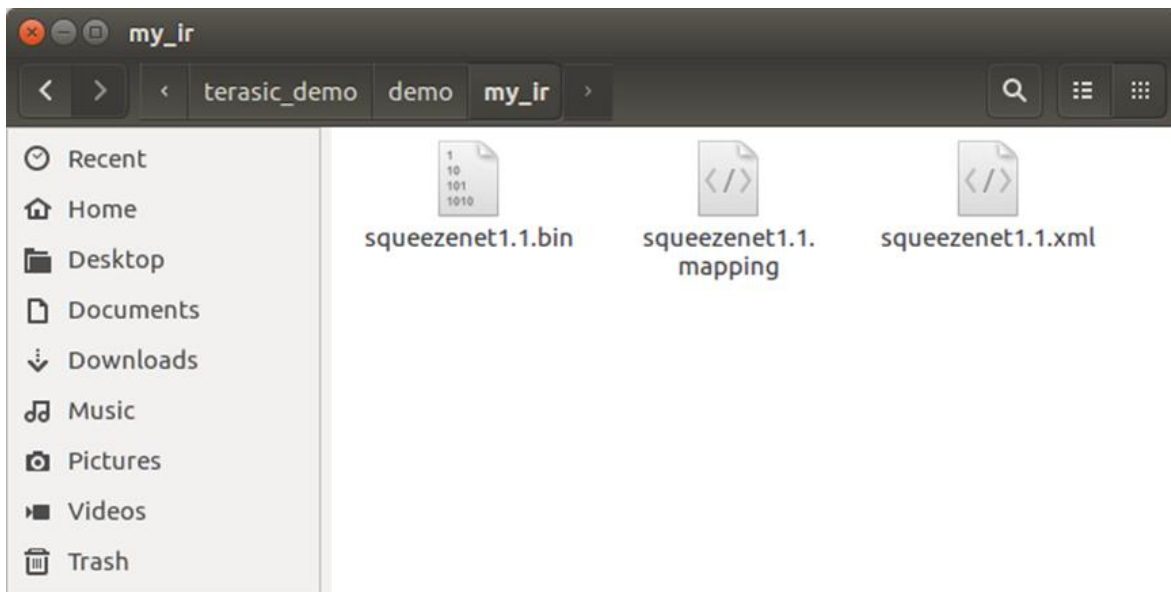
9. The corresponding IR files are generated in the my\_ir folder.

```

root@terasic: /opt/intel/2019_r1/opencvino/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/opencvino/deployment_t
ools/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/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.99 seconds.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer#

```



10. Copy the .label file from Model folder to the my\_ir folder by entering

```

“cp \
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo\
model/caffe/squeezenet1.1/squeezenet1.1.labels \
/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/my_ir/”.

```



```

root@terasic: /opt/intel/2019_r1/opencvino/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/opencvino/deployment_t
ools/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/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.99 seconds.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cp \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/\
> model/caffe/squeezenet1.1/squeezenet1.1.labels \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer
- 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/opencvino/deployment_t
ools/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/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.99 seconds.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cp \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/\
> model/caffe/squeezenet1.1/squeezenet1.1.labels \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/my_ir/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cd ..
/inference_engine/samples

```

3. Enter “`cp -r classification_sample my_classification_sample`”.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
- Reverse input channels:      False
Caffe specific parameters:
- Enable resnet optimization:   True
- Path to the Input prototxt:   /opt/intel/2019_r1/opencvino/deployment_t
ools/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/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.99 seconds.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cp \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/\
> model/caffe/squeezenet1.1/squeezenet1.1.labels \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/my_ir/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cd ..
/inference_engine/samples
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cp -r classification_sample my_classification_sample

```

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

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
- Enable resnet optimization:   True
- Path to the Input prototxt:   /opt/intel/2019_r1/opencvino/deployment_t
ools/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/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.xml
[ SUCCESS ] BIN file: /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/my_ir/squeezenet1.1.bin
[ SUCCESS ] Total execution time: 8.99 seconds.
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cp \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/\
> model/caffe/squeezenet1.1/squeezenet1.1.labels \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/my_ir/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/model_optimizer# cd ..
/inference_engine/samples
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cp -r classification_sample my_classification_sample
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# 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.

```

es# cp -r classification_sample my_classification_sample
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cd my_classification_sample
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es/my_classification_sample# gedit CMakeLists.txt

```

```

CMakeLists.txt
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_classification_sample

# 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()
    
```

CMake Tab Width: 8 Ln 7, Col 15 INS

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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
es# cd my_classification_sample
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_classification_sample# gedit CMakeLists.txt

(gedit:6687): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported

** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_classification_sample# cd ../
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples# mkdir my_build
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples# cd my_build
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build#
    
```

9. Enter “cmake -DCMAKE\_BUILD\_TYPE=Release \n /opt/intel/2019\_r1/opencvino/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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_classification_sample# gedit CMakeLists.txt

(gedit:6687): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported

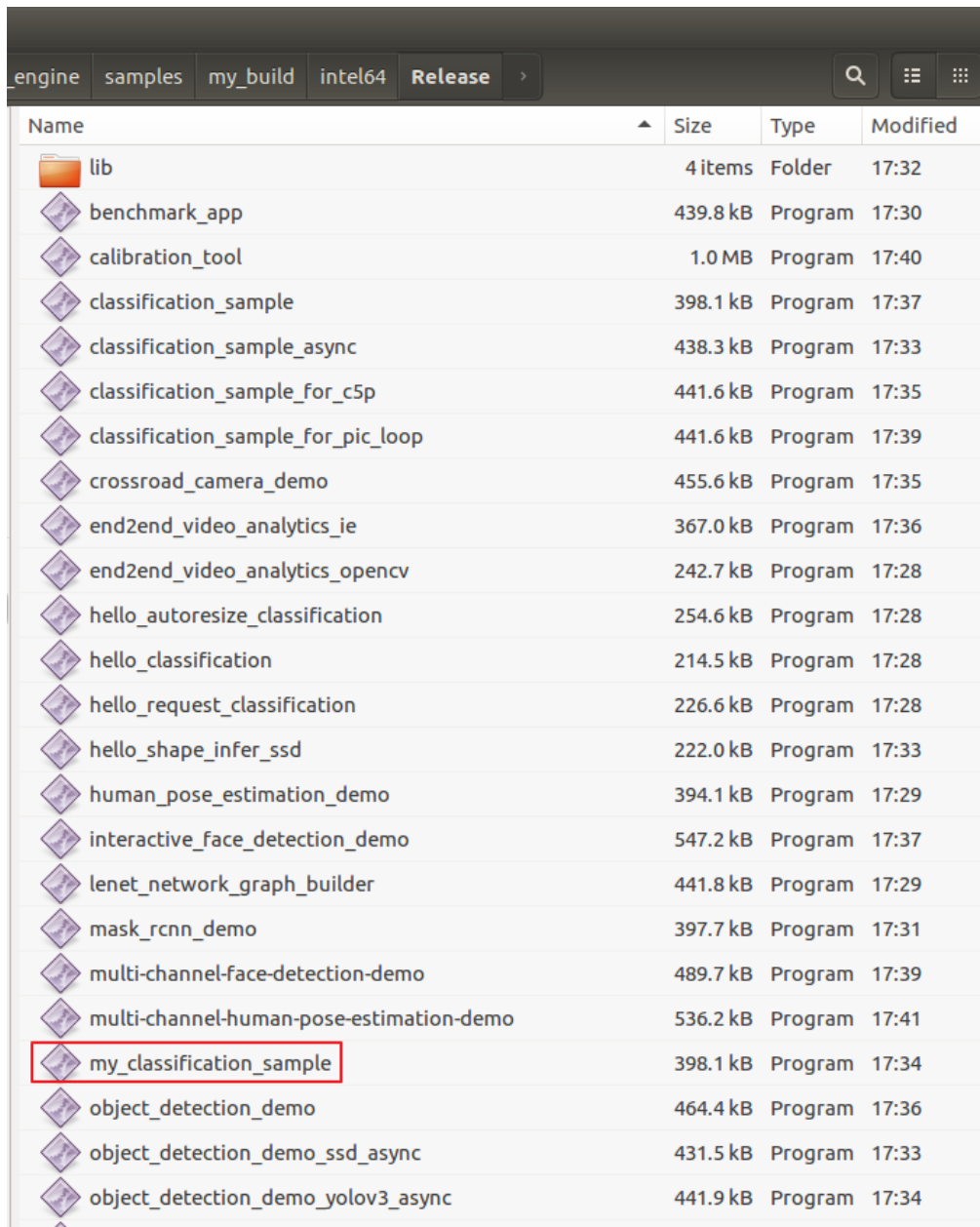
** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:6687): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_classification_sample# cd ../
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples# mkdir my_build
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples# cd my_build
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build# cmake -DCMAKE_BUILD_TYPE=Release \
> /opt/intel/2019_r1/opencvino/deployment_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	4 items	Folder	17:32
benchmark_app	439.8 kB	Program	17:30
calibration_tool	1.0 MB	Program	17:40
classification_sample	398.1 kB	Program	17:37
classification_sample_async	438.3 kB	Program	17:33
classification_sample_for_c5p	441.6 kB	Program	17:35
classification_sample_for_pic_loop	441.6 kB	Program	17:39
crossroad_camera_demo	455.6 kB	Program	17:35
end2end_video_analytics_ie	367.0 kB	Program	17:36
end2end_video_analytics_opencv	242.7 kB	Program	17:28
hello_autoresize_classification	254.6 kB	Program	17:28
hello_classification	214.5 kB	Program	17:28
hello_request_classification	226.6 kB	Program	17:28
hello_shape_infer_ssd	222.0 kB	Program	17:33
human_pose_estimation_demo	394.1 kB	Program	17:29
interactive_face_detection_demo	547.2 kB	Program	17:37
lenet_network_graph_builder	441.8 kB	Program	17:29
mask_rcnn_demo	397.7 kB	Program	17:31
multi-channel-face-detection-demo	489.7 kB	Program	17:39
multi-channel-human-pose-estimation-demo	536.2 kB	Program	17:41
my_classification_sample	398.1 kB	Program	17:34
object_detection_demo	464.4 kB	Program	17:36
object_detection_demo_ssd_async	431.5 kB	Program	17:33
object_detection_demo_yolov3_async	441.9 kB	Program	17:34

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

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

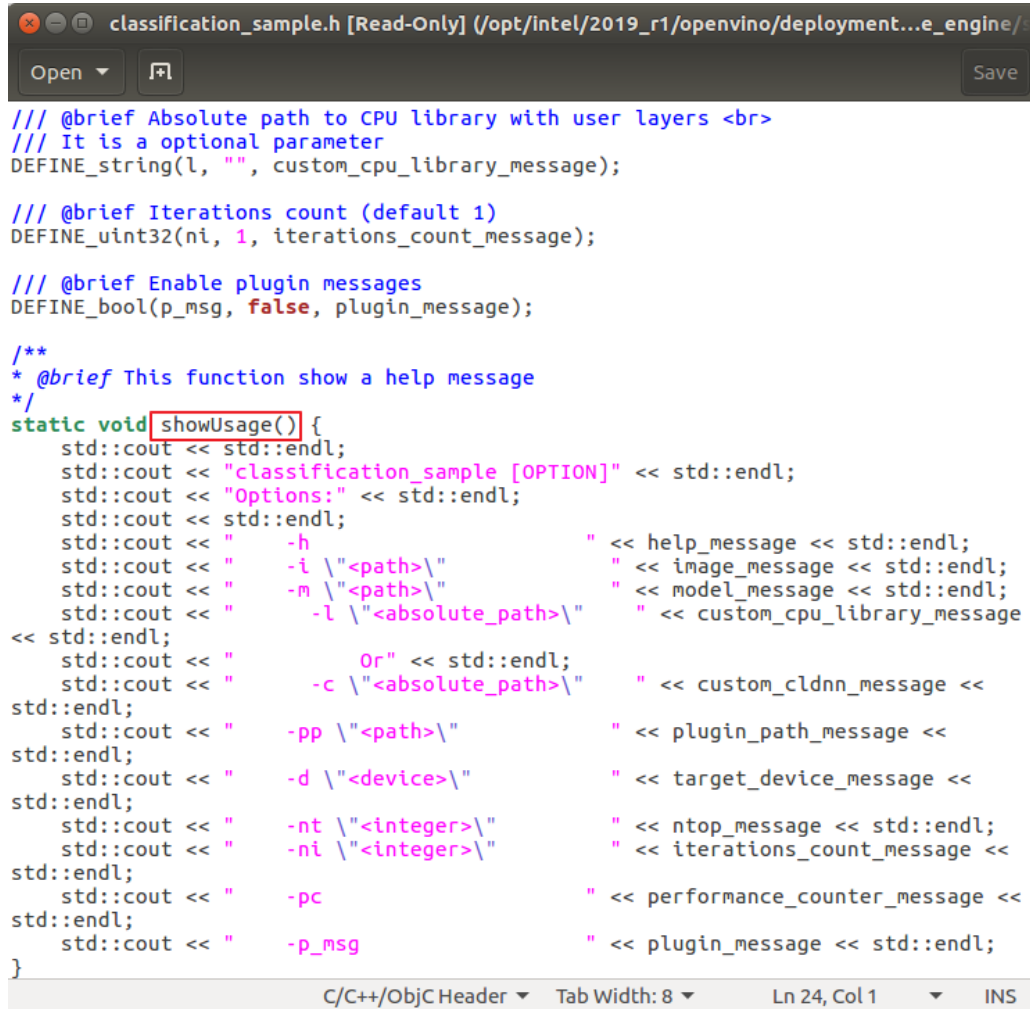
[https://docs.openvinotoolkit.org/latest/\\_docs\\_IE\\_DG\\_Integrate\\_with\\_customer\\_application\\_new\\_API.html](https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application_new_API.html)

### 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.



```

classification_sample.h [Read-Only] (/opt/intel/2019_r1/opencvino/deployment...e_engine/
Open Save

/// @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_uint32(ni, 1, iterations_count_message);

/// @brief Enable plugin messages
DEFINE_bool(p_msg, false, plugin_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;
    std::cout << "    -p_msg                 " << plugin_message << std::endl;
}
C/C++/ObjC Header Tab Width: 8 Ln 24, Col 1 INS

```

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 mkl_dnn, 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::UB);
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/opencvino/deployment_tools/inference_engine/samples\  
/my_build/intel64/Release`” to switch the directory to application folder.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
dir/src/image_grabber.cpp.o
[ 94%] Building CXX object smart_classroom_demo/CMakeFiles/smart_classroom_demo.dir/src/align_transform.cpp.o
[ 95%] Building CXX object smart_classroom_demo/CMakeFiles/smart_classroom_demo.dir/main.cpp.o
[ 96%] Building CXX object calibration_tool/CMakeFiles/calibration_tool.dir/__/validation_app/ObjectDetectionProcessor.cpp.o
[ 97%] Building CXX object calibration_tool/CMakeFiles/calibration_tool.dir/__/validation_app/Processor.cpp.o
[ 97%] Building CXX object calibration_tool/CMakeFiles/calibration_tool.dir/__/validation_app/VOCAnnotationParser.cpp.o
[ 98%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 99%] Linking CXX executable ../intel64/Release/calibration_tool
[ 99%] Built target pedestrian_tracker_demo
[ 99%] Built target calibration_tool
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-estimation-demo
[100%] Built target multi-channel-human-pose-estimation-demo
[100%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[100%] Built target smart_classroom_demo
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build# cd \
> /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples\
> /my_build/intel64/Release

```

6. Enter “`./my_classification_sample -i \`  
`/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_video/car.png \  
-m /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo\  
demo/my_ir/squeezenet1.1.xml -d "HETERO:FPGA,CPU"`” to execute the Inference Engine.

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
validation_app/ObjectDetectionProcessor.cpp.o
[ 97%] Building CXX object calibration_tool/CMakeFiles/calibration_tool.dir/__/validation_app/Processor.cpp.o
[ 97%] Building CXX object calibration_tool/CMakeFiles/calibration_tool.dir/__/validation_app/VOCAnnotationParser.cpp.o
[ 98%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 99%] Linking CXX executable ../intel64/Release/calibration_tool
[ 99%] Built target pedestrian_tracker_demo
[ 99%] Built target calibration_tool
[100%] Linking CXX executable ../../intel64/Release/multi-channel-human-pose-estimation-demo
[100%] Built target multi-channel-human-pose-estimation-demo
[100%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[100%] Built target smart_classroom_demo
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build# cd \
> /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples\
> /my_build/intel64/Release
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build/intel64/Release# ./my_classification_sample -i \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_video/car.png \
> -m /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/\
demo/my_ir/squeezenet1.1.xml -d "HETERO:FPGA,CPU"

```



```

root@terasic: /opt/intel/2019_r1/opencvino/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 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: 125.6192029
Average running time of one iteration: 125.6192029 ms

Throughput: 7.9605664 FPS

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

```

### 3.5 Advanced Experiment

1. In the previous steps, we already 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/opencvino/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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
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: 90.4634371
Average running time of one iteration: 90.4634371 ms

Throughput: 11.0541898 FPS

[ INFO ] Execution successful
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es/my_build/intel64/Release# cd /opt/intel/2019_r1/opencvino/deployment_tools/inf
erence_engine/samples
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cp -r my_classification_sample my_demo
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cd my_demo

```

6. Enter “*gedit CMakeLists.txt*” to open the file. Modify the file as follow:

```

root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
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: 90.4634371
Average running time of one iteration: 90.4634371 ms

Throughput: 11.0541898 FPS

[ INFO ] Execution successful
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es/my_build/intel64/Release# cd /opt/intel/2019_r1/opencvino/deployment_tools/inf
erence_engine/samples
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cp -r my_classification_sample my_demo
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es# cd my_demo
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/sampl
es/my_demo# gedit CMakeLists.txt

```

```

5 set (TARGET_NAME "my_demo")
6
7 # Find OpenCV components if exist
8 find_package(OpenCV COMPONENTS highgui QUIET)
9 if(NOT(OpenCV_FOUND))
10     message(WARNING "OPENCV is disabled or not found, " ${TARGET_NAME} " skipped")
11     return()
12 endif()
13
14 file (GLOB SRC
15     ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp
16 )
17 file (GLOB MAIN_HEADERS
18     ${CMAKE_CURRENT_SOURCE_DIR}/*.h
19 )
20
21 # Create named folders for the sources within the .vcproj
22 # Empty name lists them directly under the .vcproj
23 source_group("src" FILES ${SRC})
24
25 source_group("include" FILES ${MAIN_HEADERS})
26
27 link_directories(${LIB_FOLDER})
28
29 # Create library file from sources.
30 add_executable(${TARGET_NAME} ${SRC})
31
32 add_dependencies(${TARGET_NAME} gflags)
33
34 set_target_properties(${TARGET_NAME} PROPERTIES "CMAKE_CXX_FLAGS" "${CMAKE_CXX_FLAGS} -fPIE"
35 COMPILE_PDB_NAME ${TARGET_NAME})
36
37 #target_link_libraries(${TARGET_NAME} ${InferenceEngine_LIBRARIES} IE::ie_cpu_extension format_reader gflags)
38 target_link_libraries(${TARGET_NAME} IE::ie_cpu_extension ${InferenceEngine_LIBRARIES} gflags ${OpenCV_LIBRARIES})
39 if(UNIX)
40     target_link_libraries(${TARGET_NAME} ${LIB_DL} pthread)
41 endif()

```

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 MAIN_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 ${MAIN_SRC})

source_group("include" FILES ${MAIN_HEADERS})

```

```

link_directories(${LIB_FOLDER})

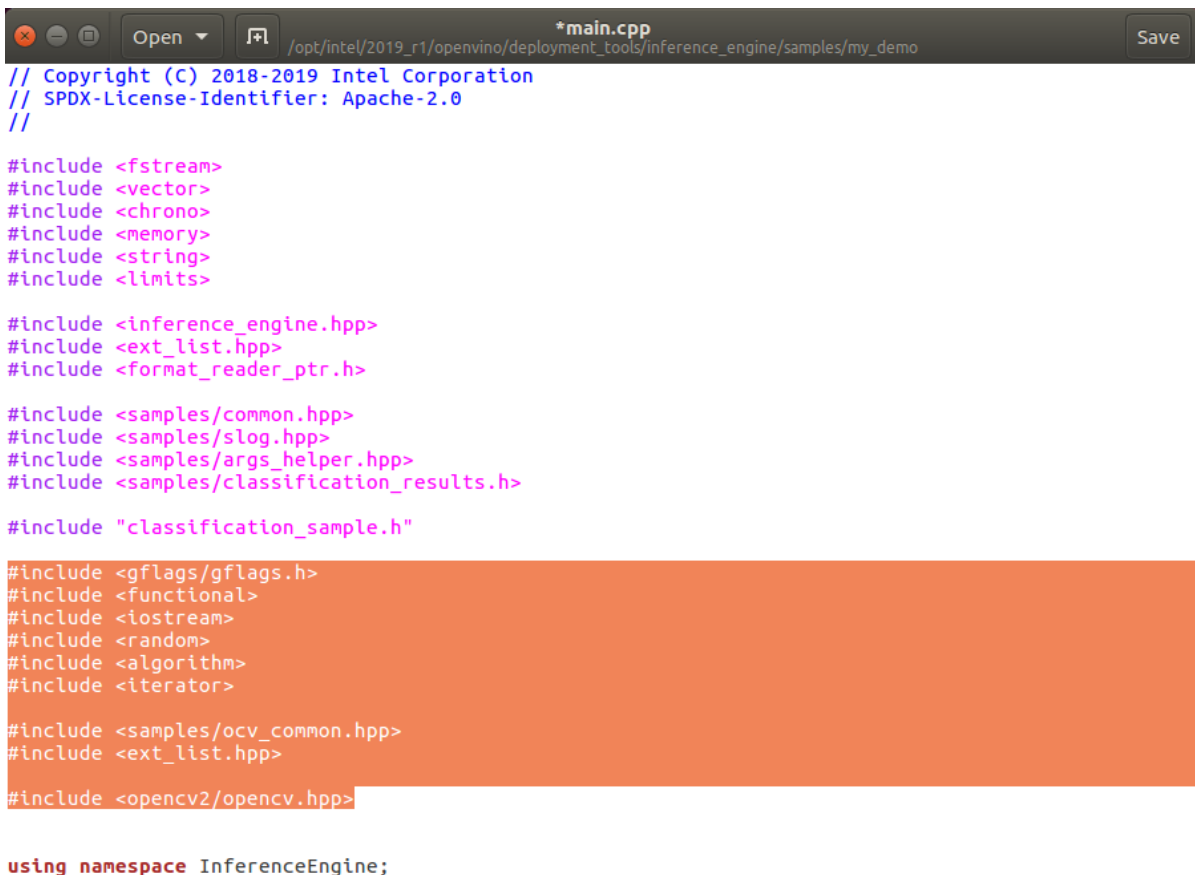
# Create library file from sources.
add_executable(${TARGET_NAME} ${MAIN_SRC} ${MAIN_HEADERS})
add_dependencies(${TARGET_NAME} gflags)
set_target_properties(${TARGET_NAME} PROPERTIES "CMAKE_CXX_FLAGS"
"${CMAKE_CXX_FLAGS} -fPIE"
COMPILE_PDB_NAME ${TARGET_NAME})

#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.



```

*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo Save

// 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.

```
#define PIC_NUM 199
```

```
*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo
Save

#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;
}

/**
 * @brief The entry point the Inference Engine sample application
 */
```

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

Before:

```

*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo

// ----- 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 mkl_dnn, 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);
}
C++ Tab Width: 8 Ln 83, Col 89 INS

```

After:

```

*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo

/**
 * @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 mkl_dnn, but they can be useful for inferring custom topologies.
             */
            plugin.AddExtension(std::make_shared<Extensions::Cpu::CpuExtensions>());
        }
    }
}
C++ Tab Width: 8 Ln 80, Col 8 INS

```

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

Before:

```

*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo
Save

/** 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) {
    FileReader::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;
C++ Tab Width: 8 Ln 168, Col 94 INS

```

After:

```

*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo
Save

/** 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);
}
C++ Tab Width: 8 Ln 153, Col 9 INS

```

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

Before:

```

*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo

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;

```

C++ Tab Width: 8 Ln 155, Col 49 INS

After:

```

*main.cpp
/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo

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;

```

C++ Tab Width: 8 Ln 155, Col 31 INS



➤ Step6, Delete the highlighted code as below

Before:

```

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

```

After:

```

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

```

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

Before:

```

*main.cpp
/opt/intel/2019_r1/opencv/deployment_tools/inference_engine/samples/my_demo
Save

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

C++ Tab Width: 8 Ln 228, Col 24 INS

```

After:

```

*main.cpp
/opt/intel/2019_r1/opencv/deployment_tools/inference_engine/samples/my_demo
Save

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);

C++ Tab Width: 8 Ln 205, Col 9 INS

```

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

```

*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save

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”

```

*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo
Save

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

C++ Tab Width: 8 Ln 214, Col 1 INS

```

```
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:

```

*main.cpp
/opt/intel/2019_r1/opencv/deployment_tools/inference_engine/samples/my_demo
Save

//
-----
// ----- 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;
C++ Tab Width: 8 Ln 259, Col 101 INS

```

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

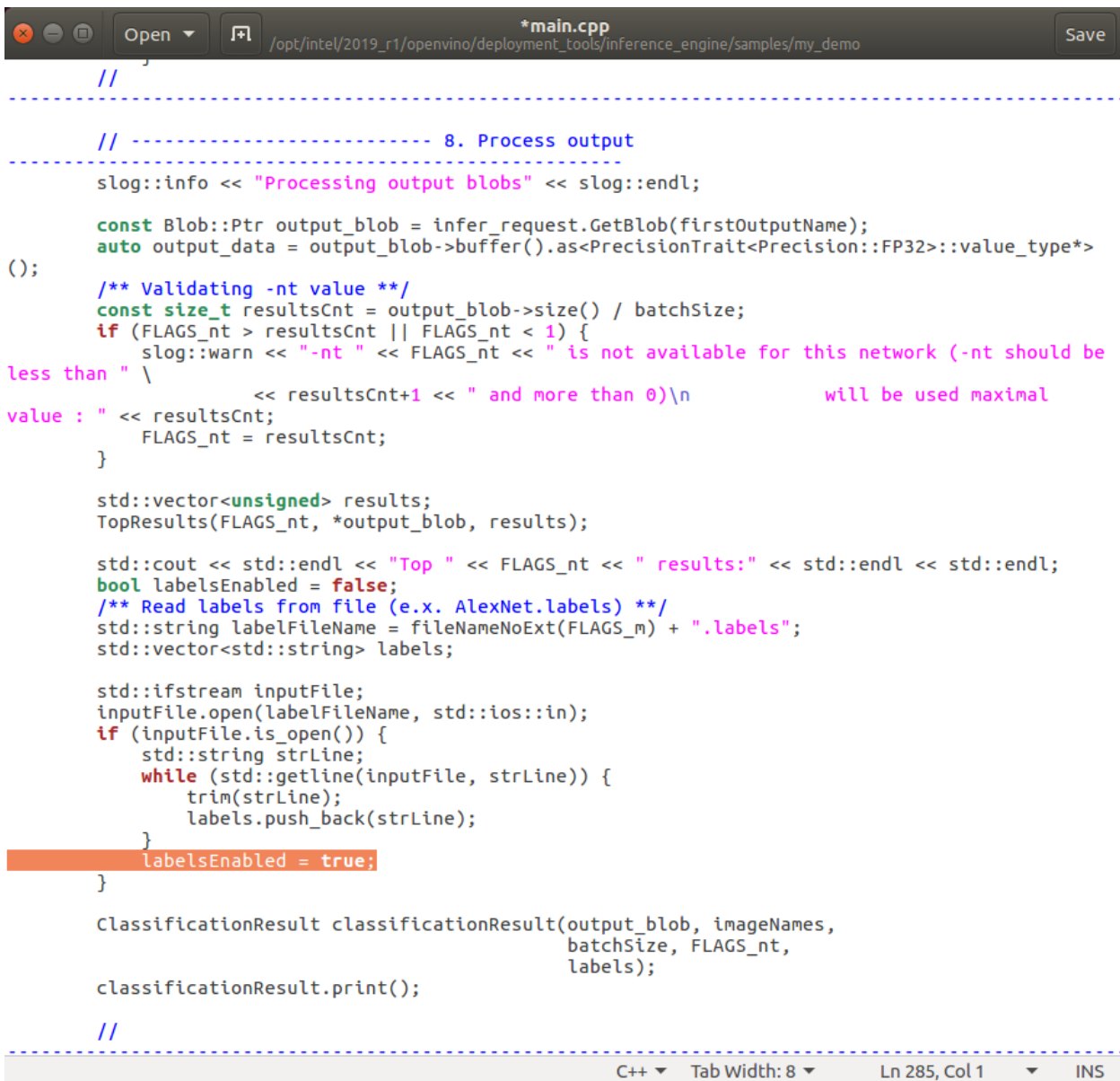
Add the code as below:

```
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:

```
labelsEnabled = true;
```



```
*main.cpp
/opt/intel/2019_r1/openvino/deployment_tools/inference_engine/samples/my_demo Save

//
// ----- 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();

//
C++ Tab Width: 8 Ln 285, Col 1 INS
```





```

*main.cpp
/opt/intel/2019_r1/opencv/deployment_tools/inference_engine/samples/my_demo
Save

std::string strLine;
while (std::getline(inputFile, strLine)) {
    trim(strLine);
    labels.push_back(strLine);
}
labelsEnabled = true;
}

/** 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));

```

```

/** 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@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples
adata::gedit-position not supported
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo# gedit main.cpp

(gedit:2720): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:2720): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported

** (gedit:2720): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:2720): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported

** (gedit:2720): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:2720): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@terasic: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo# cd /root/inference_engine_samples_build/

```

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

```

root@terasic: ~/inference_engine_samples_build
ce_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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@terasic:~/inference_engine_samples_build# cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# cd /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe# cd /root/inference_engine_samples_build/
root@terasic:~/inference_engine_samples_build# rm CMakeCache.txt

```

10. Enter “*cmake -DCMAKE\_BUILD\_TYPE=Release \ /opt/intel/2019\_r1/opencvino/deployment\_tools/inference\_engine/samples*” to copy the file to the samples build directory.

```

root@terasic: ~/inference_engine_samples_build
ce_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference
_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@terasic:~/inference_engine_samples_build# cd /opt/intel/2019_r1/opencvino/deplo
yment_tools/terasic_demo/demo/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo# cd /op
t/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/caffe/
root@terasic:/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/model/c
affe# cd /root/inference_engine_samples_build/
root@terasic:~/inference_engine_samples_build# rm CMakeCache.txt
root@terasic:~/inference_engine_samples_build# cmake -DCMAKE_BUILD_TYPE=Release \
> /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples

```

```

not supported
root@osk:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo# cd /root/inferen
ce_engine_samples_build/
root@osk:~/inference_engine_samples_build# rm CMakeCache.txt
root@osk:~/inference_engine_samples_build# cmake -DCMAKE_BUILD_TYPE=Release \
> /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples

```

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

```

root@osk: ~/inference_engine_samples_build
-- SHA supported
-- SSE supported
-- SSE2 supported
-- SSE3 supported
-- SSE4.1 supported
-- SSE4.2 supported
-- SSE4a not supported
-- SSSE3 supported
-- SYSCALL supported
-- TBM not supported
-- XOP not supported
-- XSAVE supported
-- TBB include: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/
external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_eng
ine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engin
e/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@osk:~/inference_engine_samples_build# make -j8 my_demo

```

```

root@terasic: ~/inference_engine_samples_build
-- SSSE3 supported
-- SYSCALL supported
-- TBM not supported
-- XOP not supported
-- XSAVE supported
-- TBB include: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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@terasic:~/inference_engine_samples_build# make -j8 my_demo

```

12. Enter “cd intel64/Release” to switch to app directory, then enter "ls", under this folder, the corresponding executable file for my\_demo is generated, a new application is completed

```

root@terasic: ~/inference_engine_samples_build
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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@terasic:~/inference_engine_samples_build# make -j8 my_demo
[ 7%] Built target gflags_nothreads_static
[ 92%] Built target ie_cpu_extension
Scanning dependencies of target my_demo
[ 96%] Building CXX object my_demo/CMakeFiles/my_demo.dir/main.cpp.o
[100%] Linking CXX executable ../intel64/Release/my_demo
[100%] Built target my_demo
root@terasic:~/inference_engine_samples_build# cd intel64/Release/

```

```

root@terasic:~/inference_engine_samples_build/intel64/Release# ls
benchmark_app
calibration_tool
classification_sample
classification_sample_async
classification_sample_for_c5p
classification_sample_for_pic_loop
crossroad_camera_demo
end2end_video_analytics_ie
end2end_video_analytics_opencv
hello_autoresize_classification
hello_classification
hello_request_classification
hello_shape_infer_ssd
human_pose_estimation_demo
interactive_face_detection_demo
lenet_network_graph_builder
lib
mask_rcnn_demo
multi-channel-face-detection-demo
multi-channel-human-pose-estimation-demo
my_demo
object_detection_demo
object_detection_demo_ssd_async
object_detection_demo_yolov3_async
object_detection_sample_ssd
pedestrian_tracker_demo
perfcheck
security_barrier_camera_demo
segmentation_demo
smart_classroom_demo
speech_sample
style_transfer_sample
super_resolution_demo
text_detection_demo
validation_app
root@terasic:~/inference_engine_samples_build/intel64/Release#

```



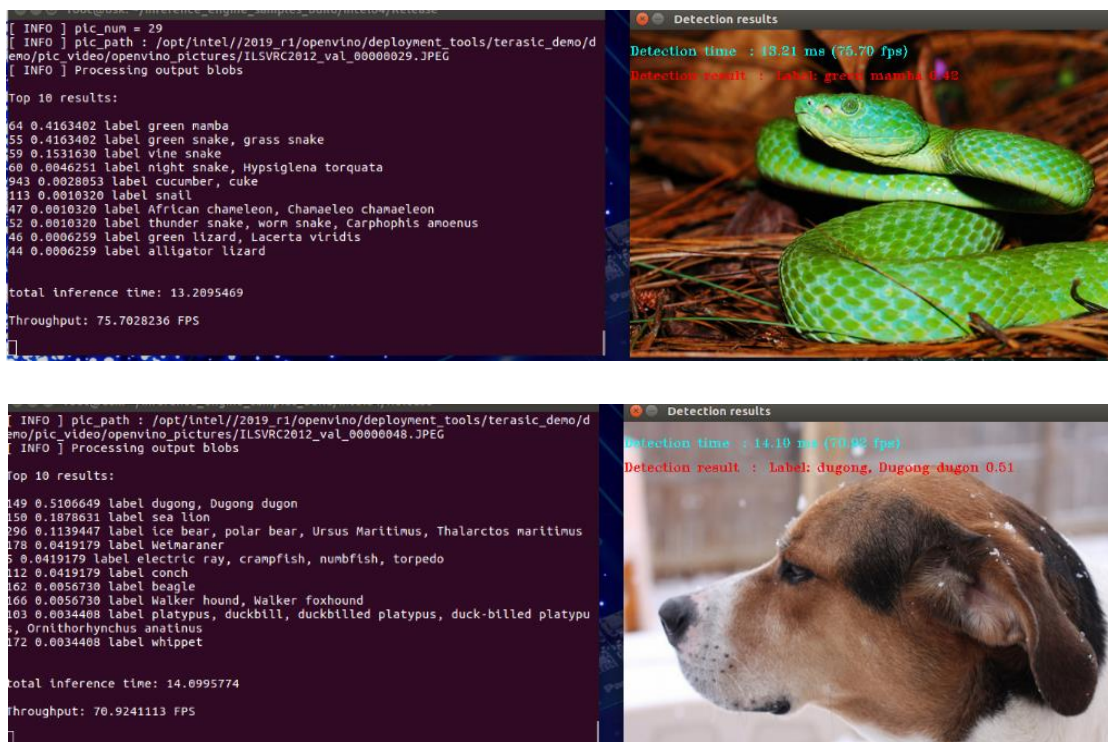
13. Enter `./my_demo -d "HETERO:FPGA,CPU" -i \`  
`/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/`  
`pic_video/opencvino_pictures -m \`  
`/opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/ir/`  
`FP16/squeezenet1.1/squeezenet1.1.xml`" to execute the host app.

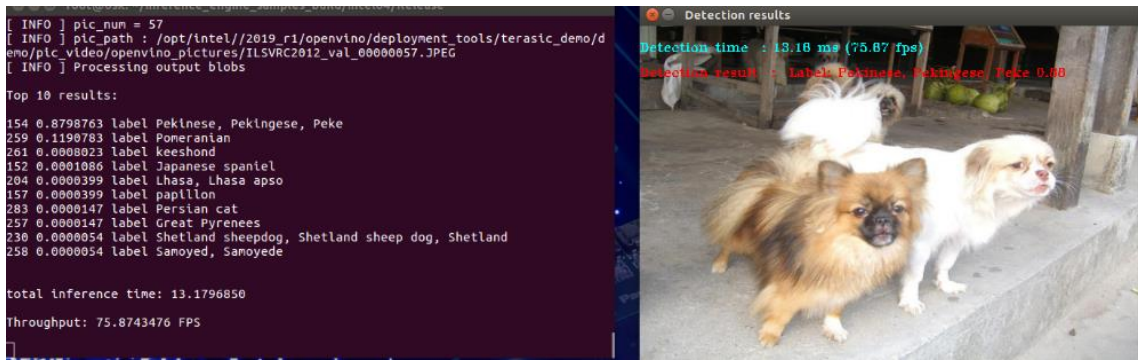
```

root@terasic:~/inference_engine_samples_build/intel64/Release# ls
benchmark_app                multi-channel-face-detection-demo
calibration_tool            multi-channel-human-pose-estimation-demo
classification_sample        my_demo
classification_sample_async  object_detection_demo
classification_sample_for_c5p object_detection_demo_ssd_async
classification_sample_for_pic_loop object_detection_demo_yolov3_async
crossroad_camera_demo       object_detection_sample_ssd
end2end_video_analytics_ie   pedestrian_tracker_demo
end2end_video_analytics_opencv perfcheck
hello_autoresize_classification security_barrier_camera_demo
hello_classification         segmentation_demo
hello_request_classification smart_classroom_demo
hello_shape_infer_ssd        speech_sample
human_pose_estimation_demo   style_transfer_sample
interactive_face_detection_demo super_resolution_demo
lenet_network_graph_builder  text_detection_demo
lib                            validation_app
mask_rcnn_demo
root@terasic:~/inference_engine_samples_build/intel64/Release# ./my_demo -d "HETERO
:FPGA,CPU" -i /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_v
ideo/opencvino_pictures -m /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/
demo/ir/FP16/squeezenet1.1/squeezenet1.1.xml

```

14. The results are as below:

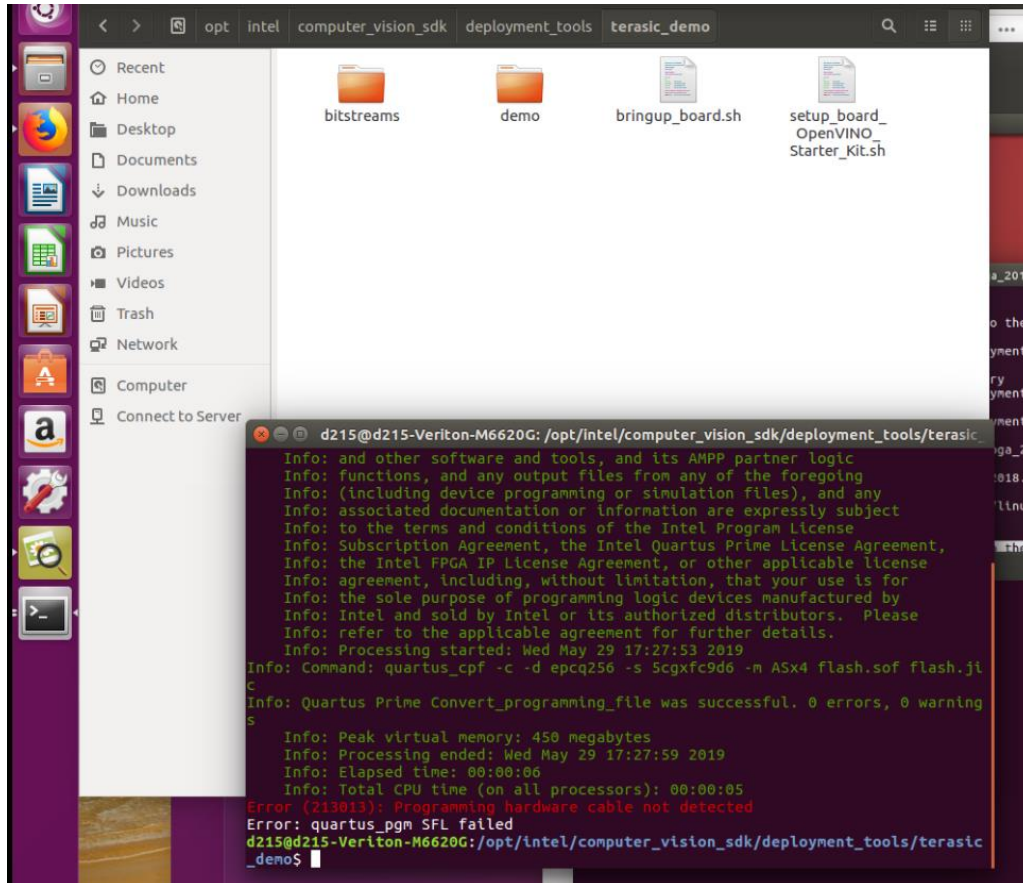




If you test all of these demos successfully, you will find the result is the same as demo 07\_classification\_pic\_loop.sh.

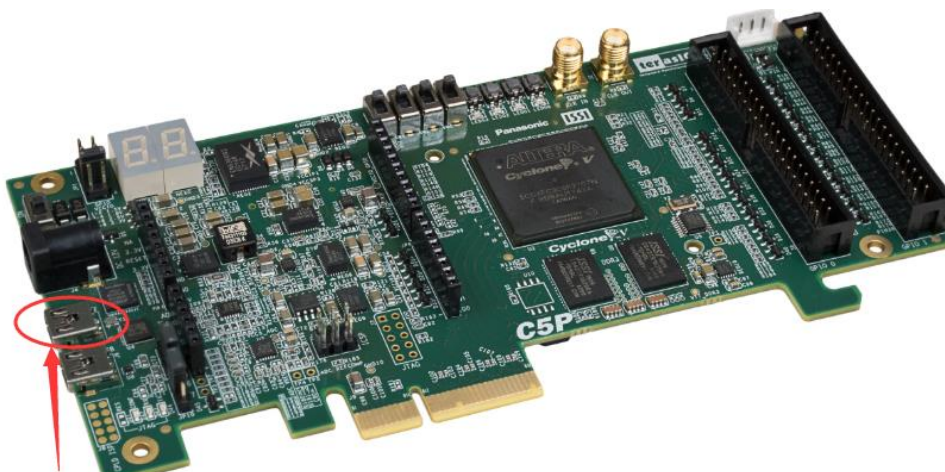
## FAQ

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:



**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.openvino toolkit.org/latest/docs\\_MO\\_DG\\_Deep\\_Learning\\_Model\\_Optimizer\\_DevGuide.html](https://docs.openvino toolkit.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
```

4. **Q:** After entering the command “./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, why does it report the error “Cannot find plugin for device: Default”?



```

root@osk: /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_demo
File Edit View Search Terminal Help
r/render_human_pose.cpp.o
[ 96%] Building CXX object multichannel_demo/hpe/CMakeFiles/multi-channel-human-pose-estimation-demo.dir/postprocess.cpp.o
[ 96%] Linking CXX executable ../intel64/Release/smart_classroom_demo
[ 96%] Linking CXX executable ../intel64/Release/object_detection_demo
[ 96%] Built target object_detection_demo
[ 96%] Built target smart_classroom_demo
[ 97%] Linking CXX executable ../intel64/Release/multi-channel-human-pose-estimation-demo
[ 97%] Built target multi-channel-human-pose-estimation-demo
[ 98%] Linking CXX executable ../intel64/Release/mask_rcnn_demo
[ 98%] Built target mask_rcnn_demo
[ 99%] Linking CXX executable ../intel64/Release/perfcheck
[ 99%] Built target perfcheck
[100%] Linking CXX executable ../intel64/Release/crossroad_camera_demo
[100%] Built target crossroad_camera_demo
root@osk:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build# cd \
> /opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples\
> /my_build/intel64/Release
root@osk:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build/intel64/Release
# ./my_classification_sample -i \
> /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_video/car.png \
-m /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/\
demo/my_ir/squeezenet1.1.xml -d "HETERO:FPGA,CPU
[ INFO ] InferenceEngine:
      API version ..... 1.6
      Build ..... custom_releases/2019/R1_c9b66a26e4d65bb986bb740e73f58c6e9e84c7c2
[ INFO ] Parsing input parameters
[ INFO ] Files were added: 1
[ INFO ]      /opt/intel/2019_r1/opencvino/deployment_tools/terasic_demo/demo/pic_video/car.png
[ INFO ] Loading plugin
[ ERROR ] Cannot find plugin for device: Default
root@osk:/opt/intel/2019_r1/opencvino/deployment_tools/inference_engine/samples/my_build/intel64/Release

```

A: Please make sure the syntax of quotation marks in the command is entered correctly

5. Q: After entering the command “cmake -DCMAKE\_BUILD\_TYPE=Release \ /opt/intel/2019\_r1/opencvino/deployment\_tools/inference\_engine/samples” to copy the file to the samples build directory, it reports the error as the following picture, why?

```

root@osk: ~/inference_engine_samples_build
File Edit View Search Terminal Help
-- TBM not supported
-- XOP not supported
-- XSAVE supported
-- TBB include: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/include
-- TBB Release lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_engine/external/tbb/lib/libtbb.so
-- TBB Debug lib: /opt/intel/2019_r1/opencvino_2019.1.094/deployment_tools/inference_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
CMake Warning (dev) at my_demo/CMakeLists.txt:29 (Add_executable):
  Policy CMP0037 is not set: Target names should not be reserved and should match a validity pattern. Run "cmake --help-policy CMP0037" for policy details. Use the cmake_policy command to set the policy and suppress this warning.

  The target name "'my_demo'" is reserved or not valid for certain CMake features, such as generator expressions, and may result in undefined behavior.
  This warning is for project developers. Use -Wno-dev to suppress it.

CMake Error at my_demo/CMakeLists.txt:33 (set_target_properties):
  set_target_properties called with incorrect number of arguments.

-- Configuring incomplete, errors occurred!
See also "/root/inference_engine_samples_build/CMakeFiles/CMakeOutput.log".
See also "/root/inference_engine_samples_build/CMakeFiles/CMakeError.log".

```