IEI Integration Corp.

MODEL:

Mustang-V100-MX4

Intel® Vision Accelerator Design with Intel® Movidius™ VPU

Powered by Open Visual Inference & Neural Network Optimization (OpenVINO™) toolkit

User Manual

Rev. 1.00 - July 22, 2019

RoHS Lead Free

Revision

Date	Version	Changes
July 22, 2019	1.00	Initial release

Copyright

COPYRIGHT NOTICE

The information in this document is subject to change without prior notice in order to improve reliability, design and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

TRADEMARKS

All registered trademarks and product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective owners.

Manual Conventions

WARNING

Warnings appear where overlooked details may cause damage to the equipment or result in personal injury. Warnings should be taken seriously.

CAUTION

Cautionary messages should be heeded to help reduce the chance of losing data or damaging the product.

NOTE

These messages inform the reader of essential but non-critical information. These messages should be read carefully as any directions or instructions contained therein can help avoid making mistakes.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

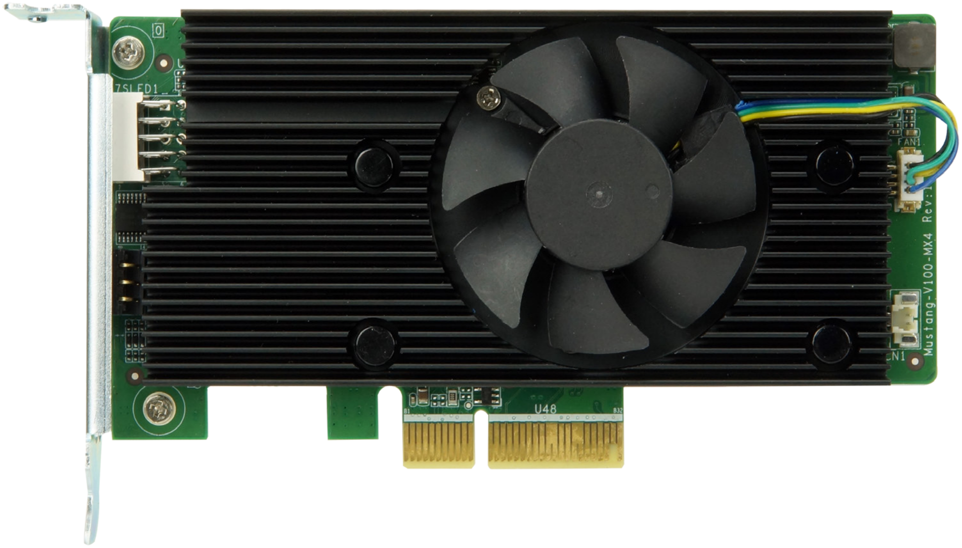


Figure 1-1: Mustang-V100-MX4

The Mustang-V100-MX4 is a deep learning convolutional neural network acceleration card for speeding up AI inference, in a flexible and scalable way. Equipped with Intel® Movidius™ Myriad™ X Vision Processing Unit (VPU), the Mustang-V100-MX4 PCIe card can be used with the existing system, enabling high-performance computing without costing a fortune. VPUs can run AI faster, and is well suited for low power consumption applications such as surveillance, retail and transportation. With the advantage of power efficiency and high performance to dedicate DNN topologies, it is perfect to be implemented in AI edge computing device to reduce total power usage, providing longer duty time for the rechargeable edge computing equipment.

"Open Visual Inference & Neural Network Optimization (OpenVINO™) toolkit" is based on convolutional neural networks (CNN), the toolkit extends workloads across Intel® hardware and maximizes performance. It can optimize pre-trained deep learning model such as Caffe, MXNET, Tensorflow into IR binary file then execute the inference engine across Intel®-hardware heterogeneously such as CPU, GPU, Intel® Movidius™ Myriad X VPU, and FPGA.

1.2 Features

Mustang-V100-MX4 features are listed below:

- Intel® Movidius™ Myriad™ X Vision Processing Unit
- Interface: PCIe 2.0 x2
- Dimensions: 113 mm x 56 mm x 23 mm
- Cooling: Active fan
- OpenVINO™ Toolkit, AI edge computing ready device
- Four Intel® Movidius™ Myriad™ X chips can execute four topologies simultaneously
- Low power consumption, approximate 15W
- Equipped with rotary switch and LED indicator for card number identification

1.3 Dimensions

The dimensions of the board are listed below:

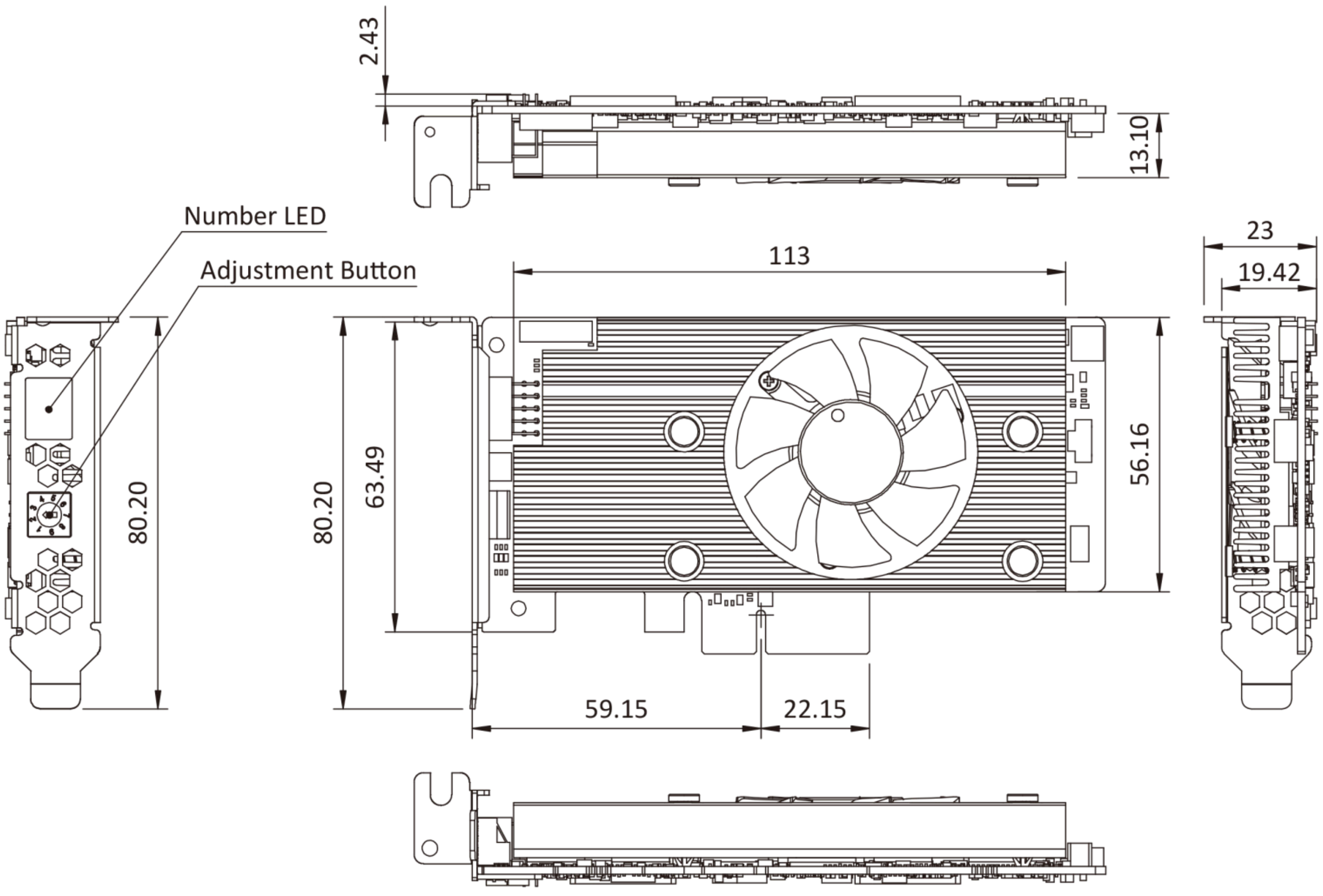


Figure 1-2: Dimensions (mm)

1.4 Technical Specifications

Mustang-V100-MX4 technical specifications are listed below.

Specification	Mustang-V100-MX4
Main Chip	Four Intel® Movidius™ Myriad™ X MA2485 VPU
Operating Systems	Ubuntu 16.04.3 (LTS), 64-bit, CentOS 64-bit Windows10 64-bit
Interface	PCIe 2.0 x2
Fan	Active fan
DIP Switch / LED Indicator	Yes (for card number identification)
Power Consumption	Approximate 15 W
Operating Temperature	0°C~55°C (in TANK AIoT Dev. Kit)
Operating Humidity	5% ~ 90%
Dimensions	113 mm x 56 mm x 23 mm
Supported Topologies	AlexNet, GoogleNetV1/V2, Mobile_, SSD, obileNetV1/V2, MTCNN, Squeezenet1.0/1.1, Tiny Yolo V1 & V2, Yolo V2, ResNet-18/50/101

Table 1-1: Technical Specifications

Chapter 2

Unpacking

2.1 Anti-static Precautions

WARNING!

Static electricity can destroy certain electronics. Make sure to follow the ESD precautions to prevent damage to the product, and injury to the user.

Make sure to adhere to the following guidelines:

- ***Wear an anti-static wristband***: Wearing an anti-static wristband can prevent electrostatic discharge.
- ***Self-grounding***: Touch a grounded conductor every few minutes to discharge any excess static buildup.
- ***Use an anti-static pad***: When configuring any circuit board, place it on an anti-static mat.
- ***Only handle the edges of the PCB***: Don't touch the surface of the motherboard. Hold the motherboard by the edges when handling.

2.2 Unpacking Precautions

When the Mustang-V100-MX4 is unpacked, please do the following:

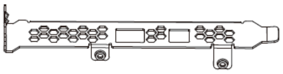
- Follow the antistatic guidelines above.
- Make sure the packing box is facing upwards when opening.
- Make sure all the packing list items are present.

2.3 Packing List

NOTE:

If any of the components listed in the checklist below are missing, do not proceed with the installation. Contact the IEI reseller or vendor the Mustang-V100-MX4 was purchased from or contact an IEI sales representative directly by sending an email to sales@ieiworld.com.

The Mustang-V100-MX4 is shipped with the following components:

Quantity	Item and Part Number	Image
1	Mustang-V100-MX4 AI acceleration card	
1	Full-height bracket	

Chapter

3

Hardware Installation

3.1 Anti-static Precautions

WARNING:

Failure to take ESD precautions during the installation of the Mustang-V100-MX4 may result in permanent damage to the Mustang-V100-MX4 and severe injury to the user.

Electrostatic discharge (ESD) can cause serious damage to electronic components, including the Mustang-V100-MX4. Dry climates are especially susceptible to ESD. It is therefore critical that whenever the Mustang-V100-MX4 or any other electrical component is handled, the following anti-static precautions are strictly adhered to.

- ***Wear an anti-static wristband***: Wearing a simple anti-static wristband can help to prevent ESD from damaging the board.
- ***Self-grounding*** Before handling the board, touch any grounded conducting material. During the time the board is handled, frequently touch any conducting materials that are connected to the ground.
- ***Use an anti-static pad***: When configuring the Mustang-V100-MX4, place it on an anti-static pad. This reduces the possibility of ESD damaging the Mustang-V100-MX4.
- ***Only handle the edges of the PCB***: When handling the PCB, hold the PCB by the edges.

3.2 Installation Considerations

NOTE:

The following installation notices and installation considerations should be read and understood before installation. All installation notices must be strictly adhered to. Failing to adhere to these precautions may lead to severe damage and injury to the person performing the installation.

WARNING:

The installation instructions described in this manual should be carefully followed in order to prevent damage to the Mustang-V100-MX4, its components and injury to the user.

Before and during the installation please **DO** the following:

- Read the user manual:
  The user manual provides a complete description of the Mustang-V100-MX4 installation instructions and configuration options.
- Wear an electrostatic discharge cuff (ESD):
  Electronic components are easily damaged by ESD. Wearing an ESD cuff removes ESD from the body and helps prevent ESD damage.
- Turn off system:
  When installing the Mustang-V100-MX4, make sure that the system to be connected is disconnected from all power supplies and that no electricity is being fed into the system.

Before and during the installation of the Mustang-V100-MX4 **DO NOT:**

- Remove any of the stickers on the PCB board. These stickers are required for warranty validation.
- Use the product before verifying all the cables and power connectors are properly connected.
- Allow screws to come in contact with the PCB circuit, connector pins, or its components.

3.3 Hardware Installation

WARNING:

DO NOT install the Mustang-V100-MX4 into the TANK AIoT Dev. Kit before shipment. It is recommended to ship them with their original boxes to prevent the Mustang-V100-MX4 from being damaged.

To install the Mustang-V100-MX4, please follow the steps below.

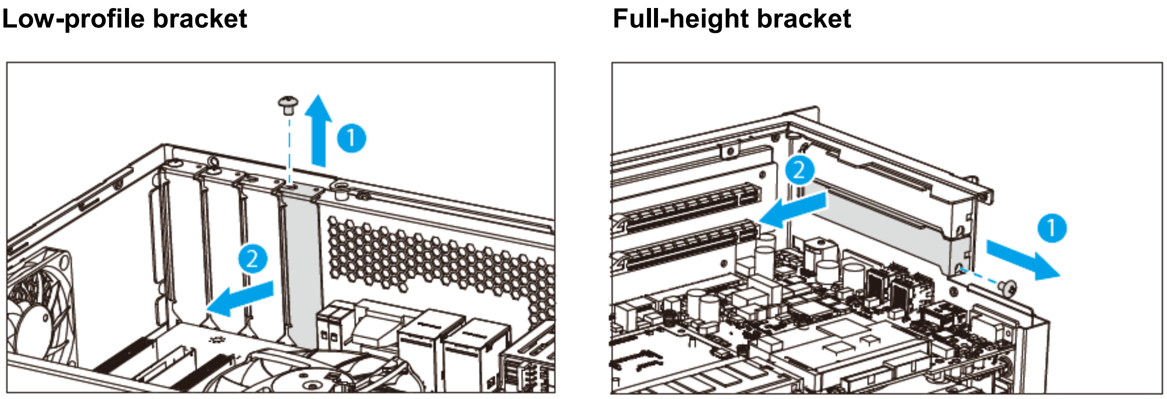
Step 1: **Prepare the computer.** Turn off the computer, and remove the power cord from the rear of the power supply.

WARNING:

Disconnect the computer from the power supply and from any networks to which you will install the Mustang-V100-MX4, or you risk damaging the system or experiencing electrical shock.

Step 2: **Remove the cover from the chassis.**

Step 3: **Locate available PCIe slots and remove the blank brackets.** The Mustang-V100-MX4 is compatible with PCIe x2 slots, and only needs one PCIe slot for installation. Remove a blank bracket panel on the back of the computer that aligns with the PCIe slot for installing the Mustang-V100-MX4. Save the bracket screw.



Low-profile bracket

Full-height bracket

Figure 3-1: Remove One Blank Bracket

Step 4: **[Only needed for full-height installation] Change the bracket on the Mustang-V100-MX4** from low-profile bracket to full-height bracket.

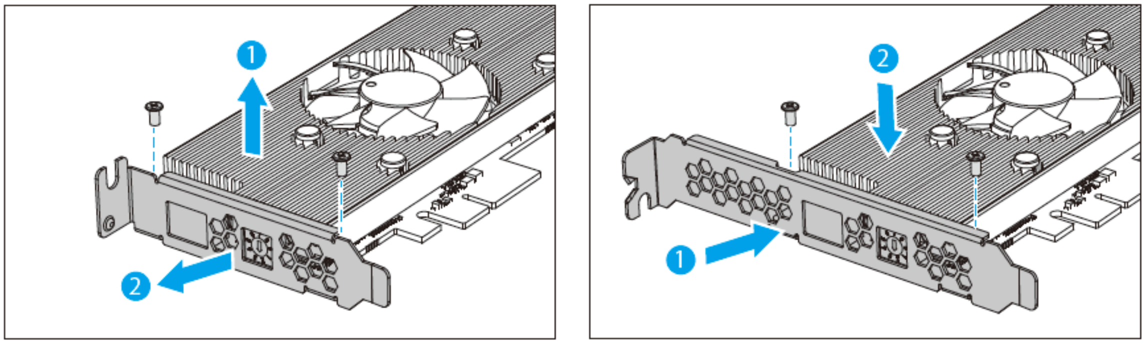


Figure 3-2: Change to Full-height Bracket

Step 5: **Install and secure the Mustang-V100-MX4 to the system.** Align the Mustang-V100-MX4 to the PCIe slot. Press down gently, but firmly, to seat the Mustang-V100-MX4 correctly in the slot. Install the bracket screw to secure the Mustang-V100-MX4 to the system's chassis.

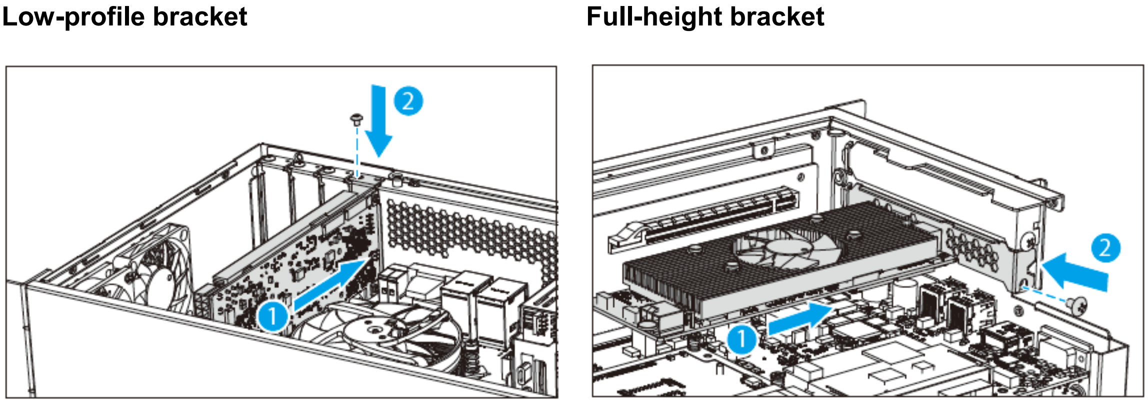


Figure 3-3: Change to Full-height Bracket

Step 6: **Assign a card ID to the Mustang-V100-MX4** by adjusting the rotary switch. The card ID number assigned here will be shown on the LED display of the card after power-up. **NOTE:** The LED digit only displays 0 through 7; if switched to 8, the LED will display 0; if switched to F, LED will display 7.

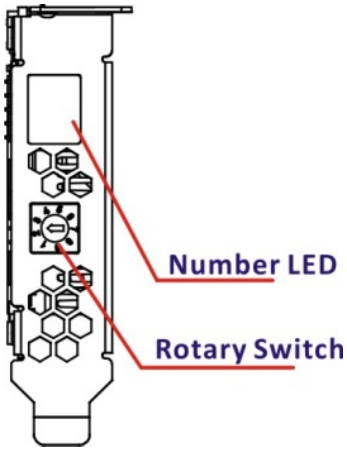


Figure 3-4: Assign a Card ID

Step 7: **Repeat Step 3 ~ Step 7 to install multiple Mustang-V100-MX4 into the system** if available.

Step 8: **Re-install the cover of the chassis**.

Step 9: **Reconnect any power cords and any network cables to the system.** Power up the system.

Chapter

4

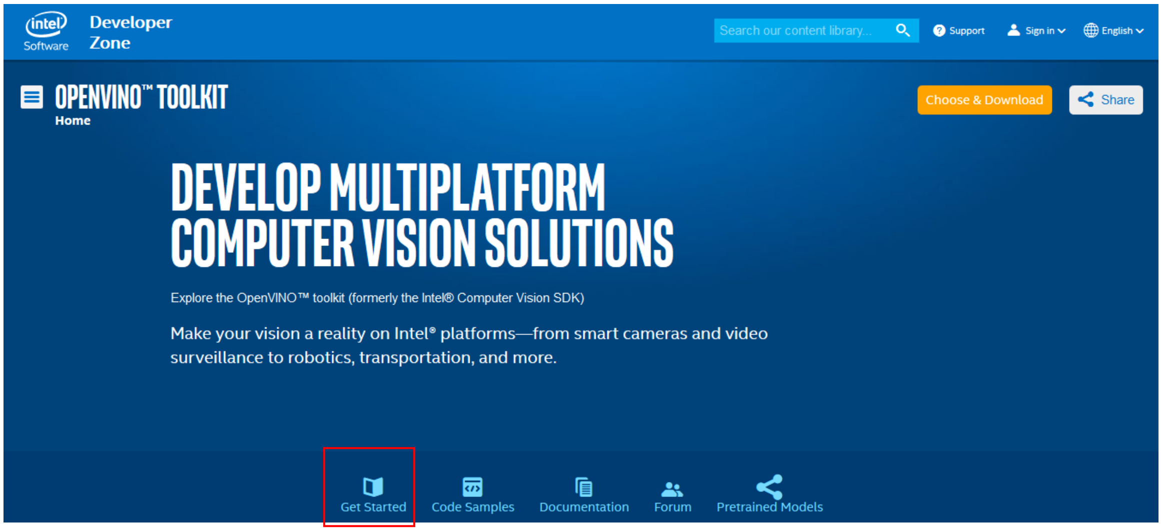
OpenVINO™ Toolkit Installation (Linux)

4.1 System Requirements

- Linux Ubuntu 16.04.3 LTS 64-bit

4.2 Online Installation Guide

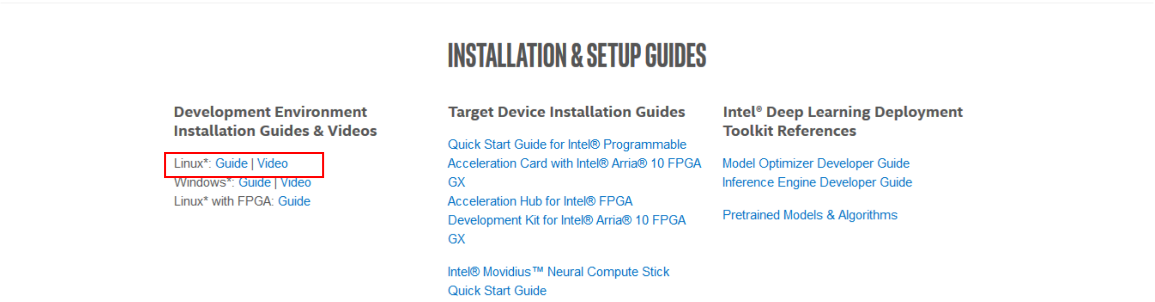
Go to https://software.intel.com/en-us/openvino-toolkit. Click “Get Started” then choose your configuration from “Development Environment Installation Guides & Videos”. Follow the instruction to complete the installation procedure.



DEVELOP MULTIPLATFORM COMPUTER VISION SOLUTIONS

Explore the OpenVINO™ toolkit (formerly the Intel® Computer Vision SDK)

Make your vision a reality on Intel® platforms—from smart cameras and video surveillance to robotics, transportation, and more.



INSTALLATION & SETUP GUIDES

Development Environment Installation Guides & Videos

Linux*: Guide | Video
Windows*: Guide | Video
Linux* with FPGA: Guide

Target Device Installation Guides

Quick Start Guide for Intel® Programmable Acceleration Card with Intel® Arria® 10 FPGA GX
Acceleration Hub for Intel® FPGA Development Kit for Intel® Arria® 10 FPGA GX

Intel® Movidius™ Neural Compute Stick Quick Start Guide

Intel® Deep Learning Deployment Toolkit References

Model Optimizer Developer Guide
Inference Engine Developer Guide

Pretrained Models & Algorithms

4.3 Installation - Step by Step (2019 R1)

This guide assumes you downloaded the Intel Distribution of OpenVINO toolkit for Linux* OS. If you do not have a copy of the toolkit package file, download the latest version and then return to this guide to proceed with the installation.

1. Open the **Terminal*** or your preferred console application and go to the **directory** in which you downloaded the Intel® Distribution of OpenVINO™ toolkit. This document assumes this is your ~/Downloads directory. If not, **replace** ~/Downloads with the directory where the file is located:
```
cd ~/Downloads/
```

2. By default, the package file is saved as **l_openvino_toolkit_p_<version>.tgz**

3. Unpack the **.tgz** file you downloaded:
```
tar -zxf l_openvino_toolkit_p_<version>.tgz
```

4. The files are unpacked to a directory named **l_openvino_toolkit_p_<version>**.

5. Go to the **l_openvino_toolkit_p_<version>** directory:
```
cd l_openvino_toolkit_p_<version>
```

6. Run a script to automatically download and install **external software dependencies**. These dependencies are required for the Intel-optimized version of OpenCV, the Deep Learning Inference Engine, and the Deep Learning Model Optimizer tools. Install these before the Intel Distribution of OpenVINO toolkit:
```
sudo -E ./install_openvino_dependencies.sh
```

The dependencies are installed. Continue to the next section to install the OpenVINO toolkit core components.

Note: The Model Optimizer has additional prerequisites that are addressed later in this document.

4.3.1 Install the Intel® Distribution of OpenVINO™ Toolkit Core Components

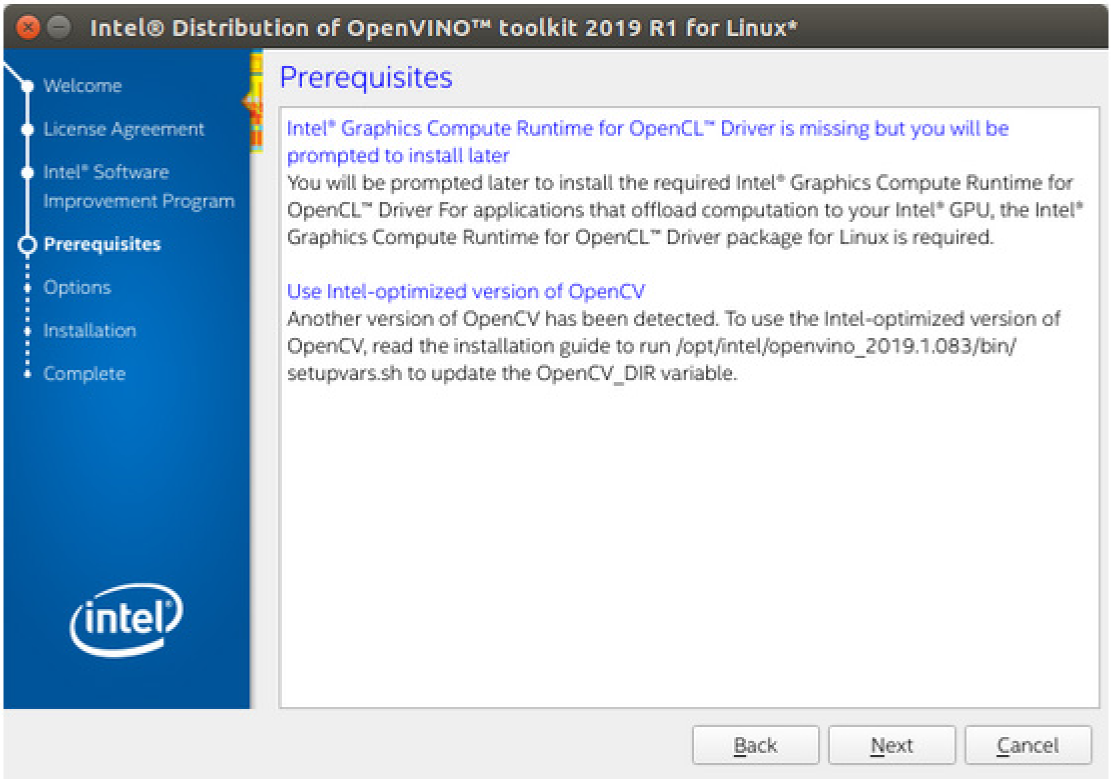
If you have a previous version of the toolkit installed, rename or delete **two directories**:

- /home/<user>/inference_engine_samples
- /home/<user>/openvino_models

To install the OpenVINO™ core components:

1. Choose one of the installation options below and run the related script with root or regular user privileges. The default installation directory path depends on the privileges you choose for the installation.
   You can use either a GUI installation wizard or command-line instructions. The only difference between the two options is that the command-line instructions are text-based. This means that instead of clicking options in a GUI, command-line prompts ask for input on a text screen.

   ▪ GUI installation Wizard:
   ```
   sudo ./install_GUI.sh
   ```
2. Follow the instructions on your screen. Watch for informational messages such as the following in case you must complete additional steps:



Intel® Distribution of OpenVINO™ toolkit 2019 R1 for Linux*

Prerequisites

Intel® Graphics Compute Runtime for OpenCL™ Driver is missing but you will be prompted to install later

You will be prompted later to install the required Intel® Graphics Compute Runtime for OpenCL™ Driver For applications that offload computation to your Intel® GPU, the Intel® Graphics Compute Runtime for OpenCL™ Driver package for Linux is required.

Use Intel-optimized version of OpenCV

Another version of OpenCV has been detected. To use the Intel-optimized version of OpenCV, read the installation guide to run /opt/intel/openvino_2019.1.083/bin/setupvars.sh to update the OpenCV_DIR variable.

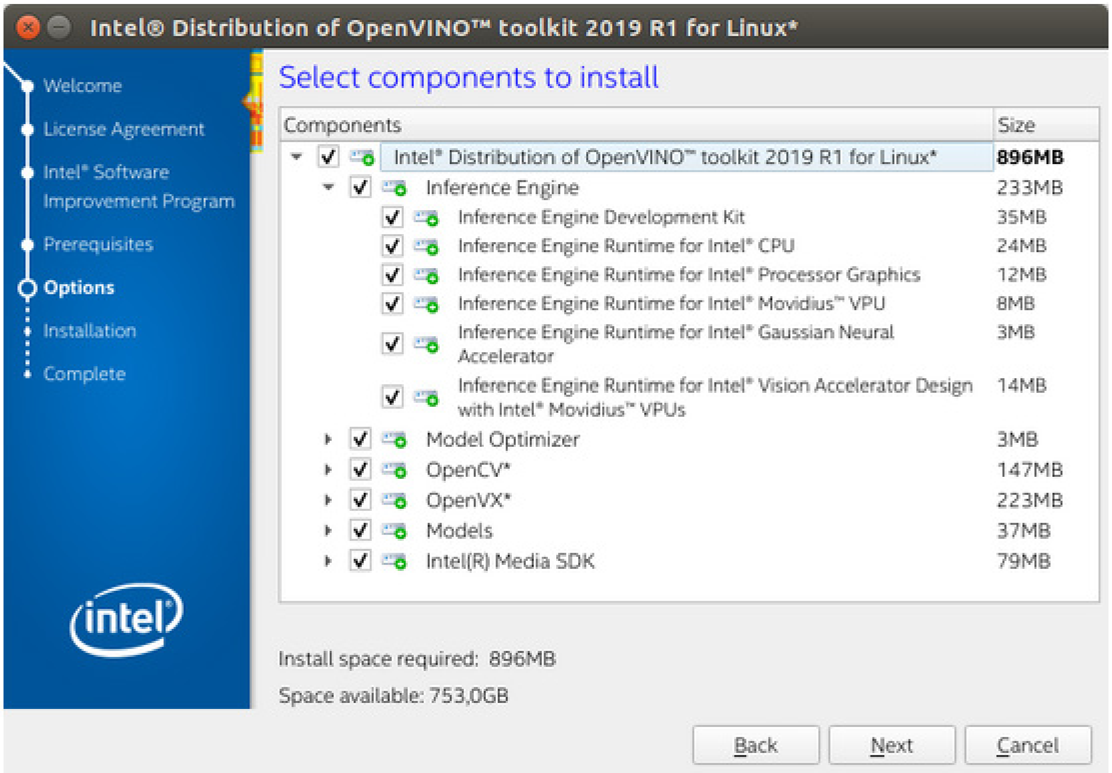
3. Click **Next**.

4. The **Installation summary** screen shows you the default component set to install:

o If you used **root privileges** to run the installer, it installs the Intel Distribution of OpenVINO toolkit in this directory: /opt/intel/computer_vision_sdk_<version>/
For simplicity, a symbolic link to the latest installation is also created: /opt/intel/computer_vision_sdk/

o If you used **regular user** privileges to run the installer, it installs the Intel Distribution of OpenVINO toolkit in this directory: /home/<user>/intel/computer_vision_sdk_<version>/
For simplicity, a symbolic link to the latest installation is also created: /home/<user>/intel/computer_vision_sdk/

5. If needed, click **Customize** to change the installation directory or the components you want to install:


Intel® Distribution of OpenVINO™ toolkit 2019 R1 for Linux*
Welcome
License Agreement
Intel® Software Improvement Program
Prerequisites
Options
Installation
Complete
intel
Back
Next
Cancel

Select components to install

Components	Size
☑ Intel® Distribution of OpenVINO™ toolkit 2019 R1 for Linux*	**896MB**
☑ Inference Engine	233MB
☑ Inference Engine Development Kit	35MB
☑ Inference Engine Runtime for Intel® CPU	24MB
☑ Inference Engine Runtime for Intel® Processor Graphics	12MB
☑ Inference Engine Runtime for Intel® Movidius™ VPU	8MB
☑ Inference Engine Runtime for Intel® Gaussian Neural Accelerator	3MB
☑ Inference Engine Runtime for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs	14MB
☑ Model Optimizer	3MB
☑ OpenCV*	147MB
☑ OpenVX*	223MB
☑ Models	37MB
☑ Intel(R) Media SDK	79MB

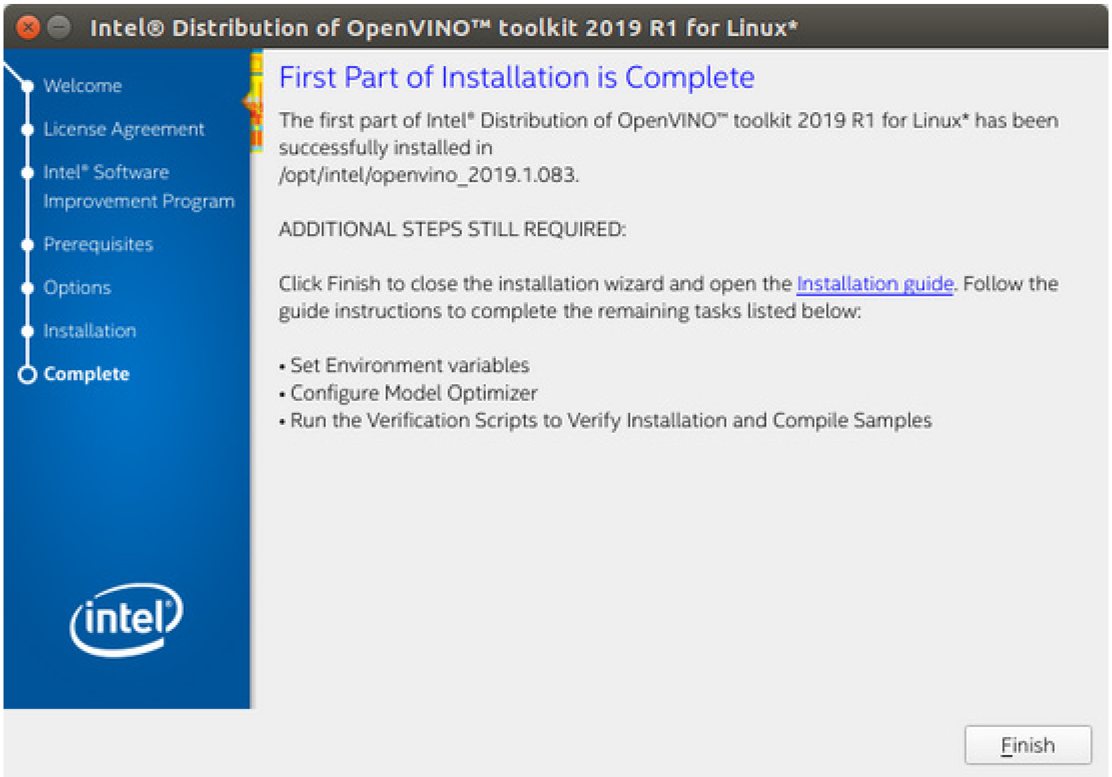
Install space required: 896MB

Space available: 753,0GB

Click **Next** to save the installation options and show the Installation summary screen.

6. On the **Installation summary** screen, press **Install** to begin the installation.

7. When the first part of installation is complete, the final screen informs you that the core components have been installed and additional steps still required:


Intel® Distribution of OpenVINO™ toolkit 2019 R1 for Linux*
Welcome
License Agreement
Intel® Software Improvement Program
Prerequisites
Options
Installation
Complete
intel
Finish

First Part of Installation is Complete

The first part of Intel® Distribution of OpenVINO™ toolkit 2019 R1 for Linux* has been successfully installed in /opt/intel/openvino_2019.1.083.

ADDITIONAL STEPS STILL REQUIRED:

Click Finish to close the installation wizard and open the Installation guide. Follow the guide instructions to complete the remaining tasks listed below:

- Set Environment variables
- Configure Model Optimizer
- Run the Verification Scripts to Verify Installation and Compile Samples

8. Click **Finish** to close the installation wizard. A new browser window opens to the next section of the installation guide to set the environment variables. You are in the same document. The new window opens in case you ran the installation without first opening this installation guide.

The core components are installed. Continue to the next section to set environment variables.

4.3.2 Set the Environment Variables

You must update several environment variables before you can compile and run OpenVINO™ applications. **Run the following script** to temporarily set your environment variables:

```
source /opt/intel/openvino_<version>/bin/setupvars.sh
```

(Optional) The Intel Distribution of OpenVINO toolkit environment variables are removed when you close the shell. As an option, you can permanently set the environment variables as follows:

1. Open the .bashrc file in <user_directory>:
   ```
   vi <user_directory>/.bashrc
   ```
2. **Add this line** to the end of the file:
   ```
   source /opt/intel/openvino_<version> /bin/setupvars.sh
   ```
3. **Save and close the file**: press the Esc key and type `:wq`.

4. To test your change, **open a new terminal**. You will see [setupvars.sh] OpenVINO environment initialized.

The environment variables are set. Continue to the next section to configure the Model Optimizer.

4.3.3 Configure the Model Optimizer

Important: This section is required. You must configure the Model Optimizer for at least one framework. The Model Optimizer will fail if you do not complete the steps in this section.

The Model Optimizer is a key component of the Intel Distribution of OpenVINO toolkit. You cannot do inference on your trained model without running the model through the Model Optimizer. When you run a pre-trained model through the Model Optimizer, your output is an Intermediate Representation (IR) of the network. The IR is a pair of files that describe the whole model:

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

The Inference Engine reads, loads, and infers the IR files, using a common API across the CPU, GPU, or VPU hardware.

The Model Optimizer is a Python*-based command line tool (mo.py), which is located in **/opt/intel/openvino_<version>/deployment_tools/model_optimizer**.

Use this tool on models trained with popular deep learning frameworks such as Caffe*, TensorFlow*, MXNet*, and ONNX* to convert them to an optimized IR format that the Inference Engine can use.

This section explains how to use scripts to configure the Model Optimizer either for all of the supported frameworks at the same time or for individual frameworks. If you want to manually configure the Model Optimizer instead of using scripts, see the using manual configuration process section in the Model Optimizer Developer Guide.

For more information about the Model Optimizer, see the Model Optimizer Developer Guide.

Model Optimizer configuration steps

You can either configure the Model Optimizer for all supported frameworks at once, or for one framework at a time. Choose the option that best suits your needs. If you see error messages, make sure you installed all dependencies.

Note: If you did not install the Intel Distribution of OpenVINO toolkit to the default installation directory, replace /intel/ with the directory where you installed the software to.

Option 1: Configure the Model Optimizer for all supported frameworks at the same time:

1. **Go to the Model Optimizer** prerequisites directory:

   ```
   cd
   /opt/intel/openvino_<version>/deployment_tools/model_optimizer
   /install_prerequisites
   ```

2. **Run the script** to configure the Model Optimizer for Caffe, TensorFlow, MXNet, Kaldi*, and ONNX:

   ```
   sudo ./install_prerequisites.sh
   ```

Option 2: Configure the Model Optimizer for each framework separately:

1. **Go to the Model Optimizer** prerequisites directory:

   ```
   cd
   /opt/intel/openvino_<version>//deployment_tools/model_optimize
   r/install_prerequisites
   ```

2. **Run the script** for your model framework. You can run more than one script:

o For Caffe:

```
sudo ./install_prerequisites_caffe.sh
```

o For TensorFlow:

```
sudo ./install_prerequisites_tf.sh
```

o For MXNet:

```
sudo ./install_prerequisites_mxnet.sh
```

o For ONNX:

```
sudo ./install_prerequisites_onnx.sh
```

o For Kaldi:

```
sudo ./install_prerequisites_kaldi.sh
```

The Model Optimizer is configured for one or more frameworks. You are ready to use two short demos to see the results of running the Intel Distribution of OpenVINO toolkit and to verify your installation was successful. The demo scripts are required since they perform additional configuration steps. Continue to the next section.

4.4 Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs Mustang-V100-MX4 Driver Installation

1. Install dependencies:

```
sudo apt install libusb-1.0-0 libboost-program-options1.58.0
libboost-thread1.58.0 libboost-filesystem1.58.0 libssl1.0.0
libudev1 libjson-c2 libelf-dev
```

2. Add the current Linux user to the users group:

```
sudo usermod -a -G users "$(whoami)"
```

3. Log out and log in for it to take effect.

4. If your Intel® Vision Accelerator Design with Intel® Movidius™ VPUs card requires SMBUS connection to PCIe slot (Raw video data card with HW version Fab-B and before), generate rules for SMBUS controller and enable the i2c_i801 driver (Intel® SMBUS controller driver):

```
cd
/opt/intel/openvino_<version>/deployment_tools/inference_engine/e
```

```
xternal/hddl

sudo chmod +x ./generate_udev_rules.sh
sudo ./generate_udev_rules.sh /etc/udev/rules.d/98-hddlbsl.rules
```

5. Check if the file /etc/modprobe.d/blacklist.conf contains the line "blacklist i2c_i801" and comment it out if so:

```
sudo sed -i "s/\(.*i2c_i801$\)/#\1/g"
/etc/modprobe.d/blacklist.conf

sudo modprobe i2c_i801
```

6. Before you start installing the drivers, make sure there is no inference application is running. Use the following command to make sure there is no HDDL service is running:

```
kill -9 $(pidof hddldaemon autoboot)
```

7. Install the drivers:

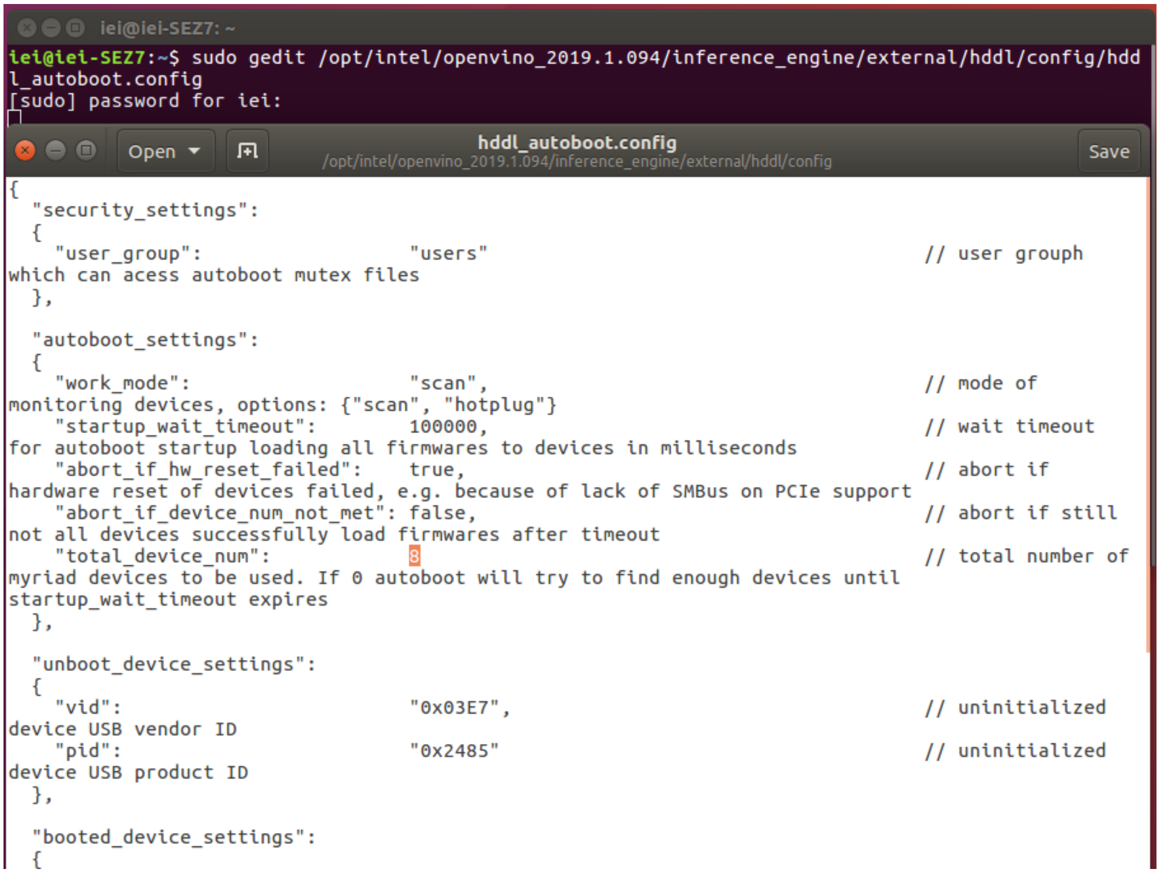
```
cd
/opt/intel/openvino_<version>/deployment_tools/inference_engine/e
xternal/hddl/drivers
sudo chmod +x ./setup.sh
sudo ./setup.sh install
sudo cp -av
/opt/intel/openvino_<version>/deployment_tools/inference_engine/e
xternal/hddl/}/../97-myriad-usbboot.rules /etc/udev/rules.d/
sudo cp -av
/opt/intel/openvino_<version>/deployment_tools/inference_engine/e
xternal/hddl/etc /
sudo udevadm control --reload-rules
sudo udevadm trigger
sudo ldconfig
```

Now, the Mustang-V100-MX4 drivers are installed.

8. Modify the device number on your host

   Change the total device numbers of “hddl_autoboot.config”.

   Each Mustang-V100-MX4 contains 4 devices, therefore, user has to change device number from default 8 to 4.



iei@iei-SEZ7: ~
Open
Save
hddl_autoboot.config
/opt/intel/openvino_2019.1.094/inference_engine/external/hddl/config

```
iei@iei-SEZ7:~$ sudo gedit /opt/intel/openvino_2019.1.094/inference_engine/external/hddl/config/hdd
l_autoboot.config
[sudo] password for iei:
```

```
{
  "security_settings":
  {
    "user_group":                   "users"                                      // user grouph
which can acess autoboot mutex files
  },

  "autoboot_settings":
  {
    "work_mode":                    "scan",                                      // mode of
monitoring devices, options: {"scan", "hotplug"}
    "startup_wait_timeout":         100000,                                      // wait timeout
for autoboot startup loading all firmwares to devices in milliseconds
    "abort_if_hw_reset_failed":     true,                                        // abort if
hardware reset of devices failed, e.g. because of lack of SMBus on PCIe support
    "abort_if_device_num_not_met": false,                                        // abort if still
not all devices successfully load firmwares after timeout
    "total_device_num":             8                                            // total number of
myriad devices to be used. If 0 autoboot will try to find enough devices until
startup_wait_timeout expires
  },

  "unboot_device_settings":
  {
    "vid":                          "0x03E7",                                    // uninitialized
device USB vendor ID
    "pid":                          "0x2485"                                     // uninitialized
device USB product ID
  },

  "booted_device_settings":
  {
```

4.5 Use the Demo Scripts to Verify Your Installation

Important: This section is required. In addition to confirming that your installation was successful, the demo scripts perform additional steps, such as setting up your computer to use the Model Optimizer samples.

To learn more about the demo applications, see README.txt in /opt/intel/oprnvino_<version>/deployment_tools/demo/.

For detailed description of the pre-trained object detection and object recognition models, go to /opt/intel/openvino_<version>/deployment_tools/intel_models/ and open index.html.

Note:

The paths in this section assume you used the default installation directory to install the Intel Distribution of OpenVINO toolkit. If you installed the software to a directory other than /opt/intel/, update the directory path with the location where you installed the toolkit to.

If you installed the product as a root user, you must switch to the root mode before you continue: sudo –i

4.5.1 Run the Image Classification Demo

1. Go to the Inference Engine demo directory:

```
cd /opt/intel/computer_vision_sdk/deployment_tools/demo
```

2. Set environment variables and run the **Image Classification** demo:

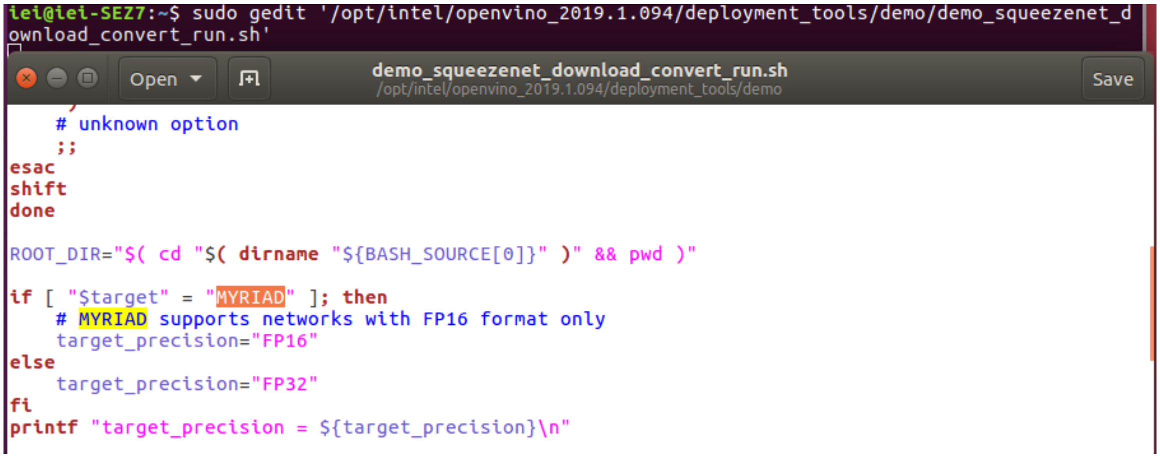
1) Set your environment variables:

```
source /opt/intel/computer_vision_sdk/bin/setupvars.sh
```

2) Modify .sh content:

```
sudo gedit demo_squeezenet_download_convert_run.sh
```

Search MYRIAD and modify to HDDL then save



```
iei@iei-SEZ7:~$ sudo gedit '/opt/intel/openvino_2019.1.094/deployment_tools/demo/demo_squeezenet_download_convert_run.sh'
```

demo_squeezenet_download_convert_run.sh
/opt/intel/openvino_2019.1.094/deployment_tools/demo

```
    # unknown option
    ;;
esac
shift
done

ROOT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

if [ "$target" = "MYRIAD" ]; then
    # MYRIAD supports networks with FP16 format only
    target_precision="FP16"
else
    target_precision="FP32"
fi
printf "target_precision = ${target_precision}\n"
```

3) Assign Mustang-V100-MX4 (-d HDDL) and run demo

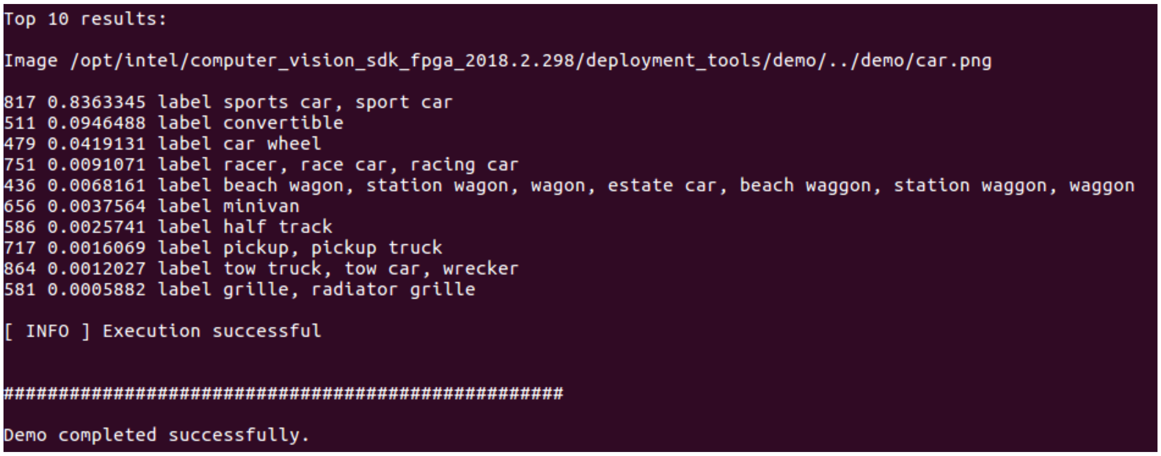
```
/demo_squeezenet_download_convert_run.sh -d HDDL
```

3. The Image Classification demo uses the Model Optimizer to convert a SqueezeNet model to .bin and .xml Intermediate Representation (IR) files. The Inference Engine component uses these files.

   For a brief description of the Intermediate Representation .bin and .xml files, see Configuring the Model Optimizer.

   This demo creates the directory /home/<user>/inference_engine_samples/.

   This demo uses car.png in the demo directory. When the demo completes, you will see the label and confidence for the top-10 categories:



```
Top 10 results:

Image /opt/intel/computer_vision_sdk_fpga_2018.2.298/deployment_tools/demo/../demo/car.png

817 0.8363345 label sports car, sport car
511 0.0946488 label convertible
479 0.0419131 label car wheel
751 0.0091071 label racer, race car, racing car
436 0.0068161 label beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
656 0.0037564 label minivan
586 0.0025741 label half track
717 0.0016069 label pickup, pickup truck
864 0.0012027 label tow truck, tow car, wrecker
581 0.0005882 label grille, radiator grille

[ INFO ] Execution successful


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

Demo completed successfully.
```

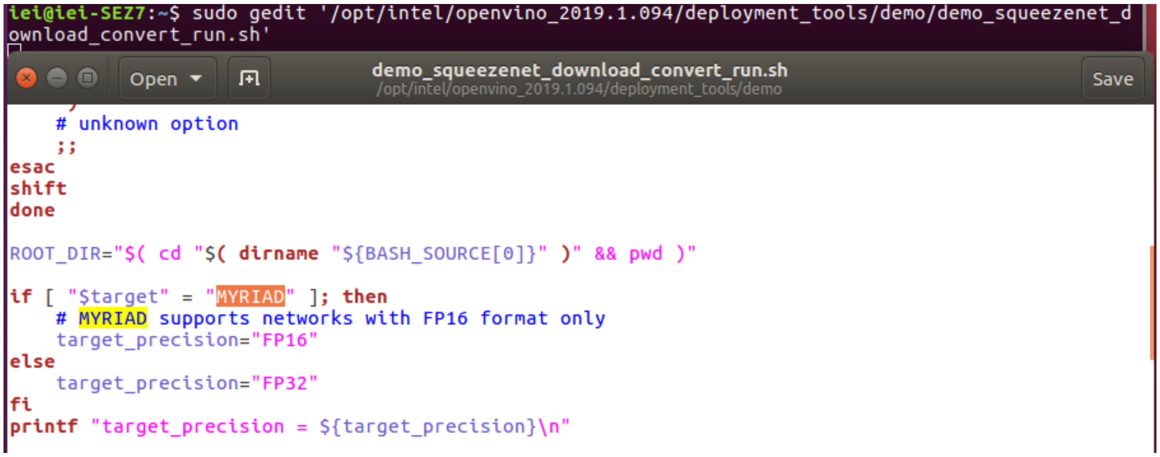
This demo is complete. Continue to the next section to run the Inference Pipeline demo.

4.5.2 Run the Inference Pipeline Demo

1. While still in /opt/intel/computer_vision_sdk/deployment_tools/demo/, **run the Inference Pipeline demo**.

2. Modify .sh content:
   ```
   sudo gedit demo_squeezenet_download_convert_run.sh
   ```

   Search MYRIAD and modify to HDDL then save



```
iei@iei-SEZ7:~$ sudo gedit '/opt/intel/openvino_2019.1.094/deployment_tools/demo/demo_squeezenet_d
ownload_convert_run.sh'
```

Open
Save
demo_squeezenet_download_convert_run.sh
/opt/intel/openvino_2019.1.094/deployment_tools/demo

```
    # unknown option
    ;;
esac
shift
done

ROOT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

if [ "$target" = "MYRIAD" ]; then
    # MYRIAD supports networks with FP16 format only
    target_precision="FP16"
else
    target_precision="FP32"
fi
printf "target_precision = ${target_precision}\n"
```

3. Assign Mustang-V100-MX4 (-d HDDL) and run demo

   ```
   /demo_security_barrier_camera.sh -d HDDL
   ```

4. This demo uses car.png in /opt/intel/computer_vision_sdk/deployment_tools/demo/ to show an inference pipeline. This demo uses three pre-trained models. The demo uses vehicle recognition in which vehicle attributes build on each other to narrow in on a specific attribute. The demo works as follows:

   1) An object is identified as a vehicle.

   2) This identification is used as input to the next model, which identifies specific vehicle attributes, including the license plate.

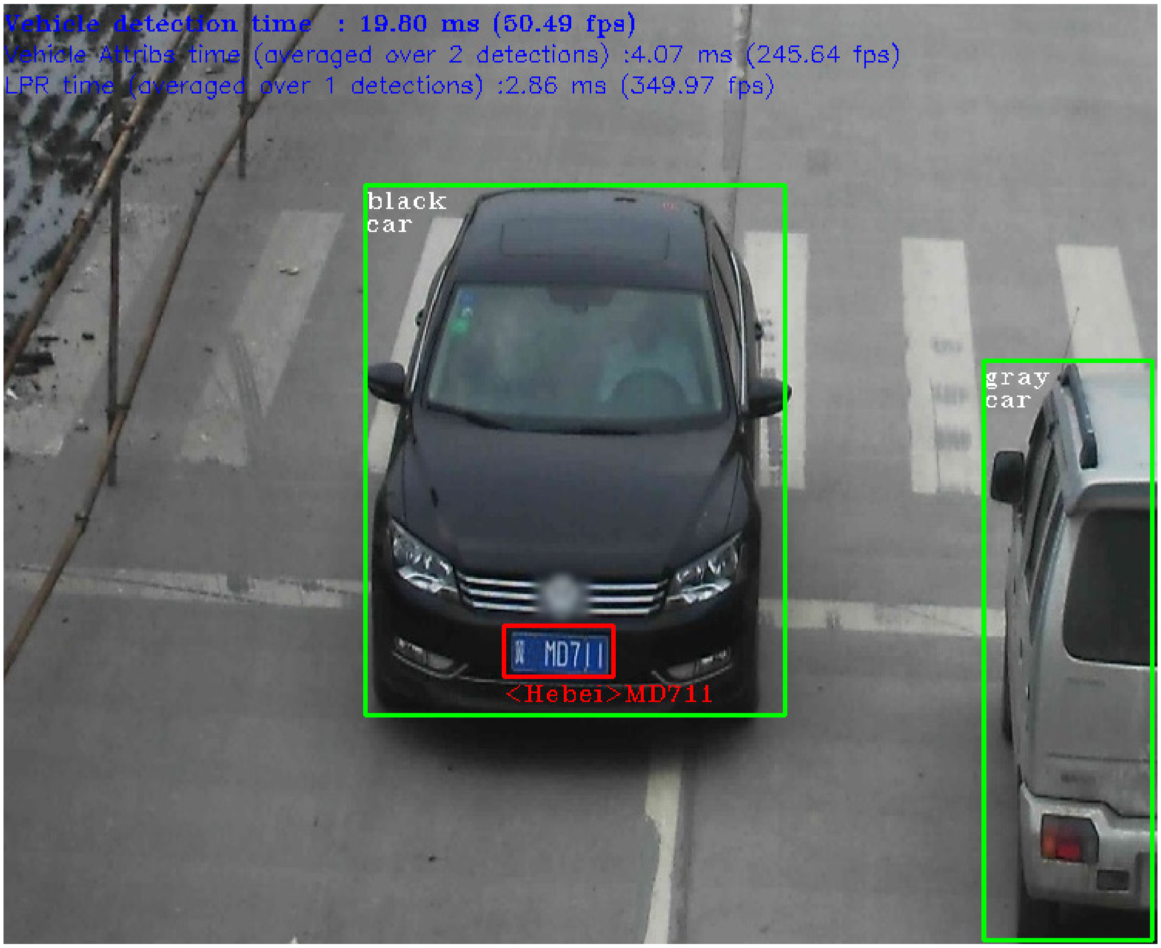
   3) The attributes identified as the license plate are used as input to the third model, which recognizes specific characters in the license plate.

   For more information about the demo, see the Security Camera Sample.

5. When the demo completes, two windows are open:

o A console window that displays information about the tasks performed by the demo

o An image viewer window that displays a picture similar to the following:



```
Vehicle detection time  : 19.80 ms (50.49 fps)
Vehicle Attribs time (averaged over 2 detections) :4.07 ms (245.64 fps)
LPR time (averaged over 1 detections) :2.86 ms (349.97 fps)
```

6. Close the image viewer screen to end the demo.

In this section, you saw a preview of the Intel Distribution of OpenVINO toolkit capabilities.

You have completed all the required installation, configuration, and build steps to work with your trained models using the Mustang-V100-MX4.

4.6 Installation - Step by Step (OpenVINO 2018 R5)

This guide assumes you downloaded the Intel Distribution of OpenVINO toolkit for Linux* OS. If you do not have a copy of the toolkit package file, download the latest version and then return to this guide to proceed with the installation.

NOTE: Before beginning the installation, make sure you have the correct Linux kernel version:

```
cat /proc/version
```

Make sure you are using a Linux kernel version 4.14 and later.

For example:

- 4.15.0-38-Generic

Install kernel of 4.15.0-38

- `apt-get install linux-image-4.15.0-38-generic`
- `apt-get install linux-headers-4.15.0-38-generic`
- `apt-get install linux-modules-extra-4.15.0-38-generic`
- `apt-get remove linux-modules-extra-4.15.0-38-generic`

4.6.1 Install External Software Dependencies

1. Open the **Terminal*** or your preferred console application and go to the **directory** in which you downloaded the Intel® Distribution of OpenVINO™ toolkit. This document assumes this is your ~/Downloads directory. If not, **replace** ~/Downloads with the directory where the file is located:
   ```
   cd ~/Downloads/
   ```

2. By default, the package file is saved as **l_openvino_toolkit_p_<version>.tgz**

3. Unpack the **.tgz** file you downloaded:
   ```
   tar -zxf l_openvino_toolkit_p_<version>.tgz
   ```

4. The files are unpacked to a directory named **l_openvino_toolkit_p_<version>**.

5. Go to the **l_openvino_toolkit_p_<version>** directory:

```
cd l_openvino_toolkit_p_<version>
```

6. Run a script to automatically download and install **external software dependencies**. These dependencies are required for the Intel-optimized version of OpenCV, the Deep Learning Inference Engine, and the Deep Learning Model Optimizer tools. Install these before the Intel Distribution of OpenVINO toolkit:

```
sudo -E ./install_cv_sdk_dependencies.sh
```

7. As an option, you can install all the dependencies manually instead of running install_cv_sdk_dependencies.sh. In this case, use the list of dependencies at System Requirements.

The dependencies are installed. Continue to the next section to install the OpenVINO toolkit core components.

Note: The Model Optimizer has additional prerequisites that are addressed later in this document.

4.6.2 Install the Intel® Distribution of OpenVINO™ Toolkit Core Components

If you have a previous version of the toolkit installed, rename or delete **two directories**:

- /home/<user>/inference_engine_samples
- /home/<user>/openvino_models

To install the OpenVINO™ core components:

1. Choose one of the installation options below and run the related script with root or regular user privileges. The default installation directory path depends on the privileges you choose for the installation.
You can use either a GUI installation wizard or command-line instructions. The only difference between the two options is that the command-line instructions are text-based. This means that instead of clicking options in a GUI, command-line prompts ask for input on a text screen.

   ▪ GUI installation Wizard:

```
sudo ./install_GUI.sh
```

2. Follow the instructions on your screen. Watch for informational messages such as the following in case you must complete additional steps:

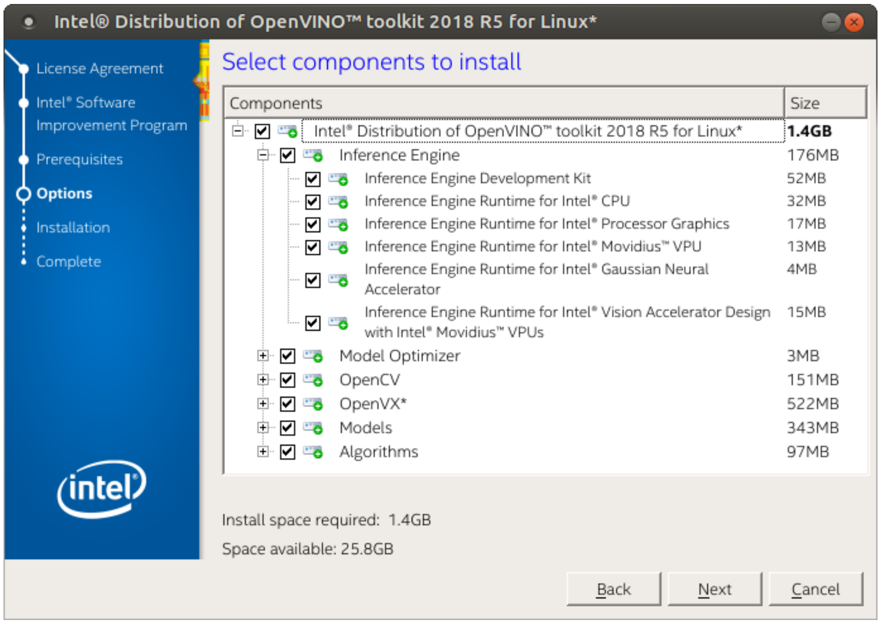
3. Click **Next**.

4. The **Installation summary** screen shows you the default component set to install:

o If you used **root privileges** to run the installer, it installs the Intel Distribution of OpenVINO toolkit in this directory: /opt/intel/computer_vision_sdk_<version>/
For simplicity, a symbolic link to the latest installation is also created: /opt/intel/computer_vision_sdk/

o If you used **regular user** privileges to run the installer, it installs the Intel Distribution of OpenVINO toolkit in this directory: /home/<user>/intel/computer_vision_sdk_<version>/
For simplicity, a symbolic link to the latest installation is also created: /home/<user>/intel/computer_vision_sdk/

5. If needed, click **Customize** to change the installation directory or the components you want to install:



Intel® Distribution of OpenVINO™ toolkit 2018 R5 for Linux*

Select components to install

Components	Size
☑ Intel® Distribution of OpenVINO™ toolkit 2018 R5 for Linux*	**1.4GB**
☑ Inference Engine	176MB
☑ Inference Engine Development Kit	52MB
☑ Inference Engine Runtime for Intel® CPU	32MB
☑ Inference Engine Runtime for Intel® Processor Graphics	17MB
☑ Inference Engine Runtime for Intel® Movidius™ VPU	13MB
☑ Inference Engine Runtime for Intel® Gaussian Neural Accelerator	4MB
☑ Inference Engine Runtime for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs	15MB
☑ Model Optimizer	3MB
☑ OpenCV	151MB
☑ OpenVX*	522MB
☑ Models	343MB
☑ Algorithms	97MB

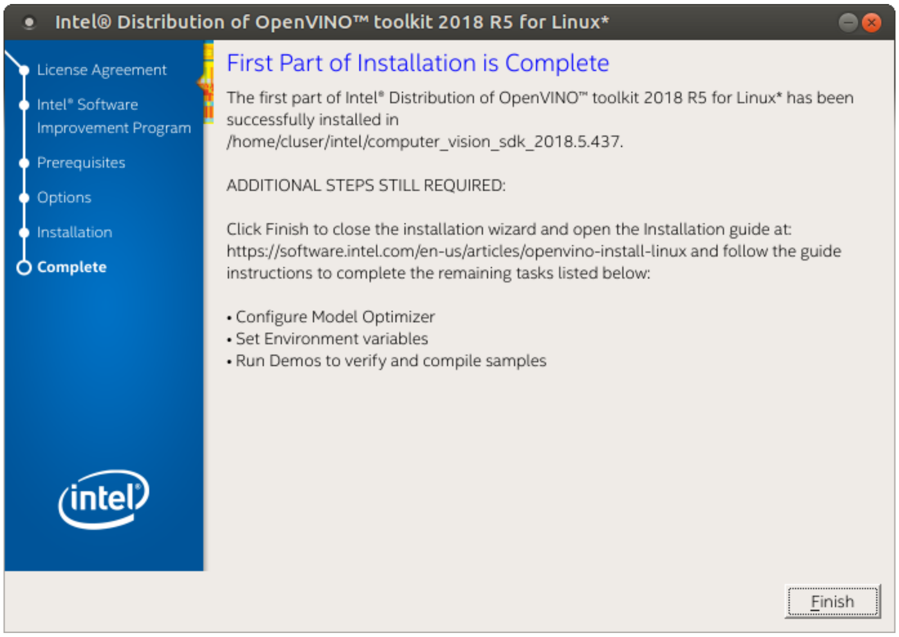
Install space required: 1.4GB

Space available: 25.8GB

Click **Next** to save the installation options and show the Installation summary screen.

6. On the **Installation summary** screen, press **Install** to begin the installation.

7. When the first part of installation is complete, the final screen informs you that the core components have been installed and additional steps still required:



Intel® Distribution of OpenVINO™ toolkit 2018 R5 for Linux*
License Agreement
Intel® Software Improvement Program
Prerequisites
Options
Installation
Complete
intel
Finish

First Part of Installation is Complete

The first part of Intel® Distribution of OpenVINO™ toolkit 2018 R5 for Linux* has been successfully installed in /home/cluser/intel/computer_vision_sdk_2018.5.437.

ADDITIONAL STEPS STILL REQUIRED:

Click Finish to close the installation wizard and open the Installation guide at: https://software.intel.com/en-us/articles/openvino-install-linux and follow the guide instructions to complete the remaining tasks listed below:

- Configure Model Optimizer
- Set Environment variables
- Run Demos to verify and compile samples

8. Click **Finish** to close the installation wizard. A new browser window opens to the next section of the installation guide to set the environment variables. You are in the same document. The new window opens in case you ran the installation without first opening this installation guide.

The core components are installed. Continue to the next section to set environment variables.

4.6.3 Set the Environment Variables

You must update several environment variables before you can compile and run OpenVINO™ applications. **Run the following script** to temporarily set your environment variables:

```
source /opt/intel/computer_vision_sdk/bin/setupvars.sh
```

(Optional) The Intel Distribution of OpenVINO toolkit environment variables are removed when you close the shell. As an option, you can permanently set the environment variables as follows:

1. Open the .bashrc file in <user_directory>:
   ```
   vi <user_directory>/.bashrc
   ```
2. **Add this line** to the end of the file:
   ```
   source /opt/intel/computer_vision_sdk/bin/setupvars.sh
   ```
3. **Save and close the file**: press the Esc key and type :wq.

4. To test your change, **open a new terminal**. You will see [setupvars.sh] OpenVINO environment initialized.

The environment variables are set. Continue to the next section to configure the Model Optimizer.

4.6.4 Configure the Model Optimizer

Important: This section is required. You must configure the Model Optimizer for at least one framework. The Model Optimizer will fail if you do not complete the steps in this section.

The Model Optimizer is a key component of the Intel Distribution of OpenVINO toolkit. You cannot do inference on your trained model without running the model through the Model Optimizer. When you run a pre-trained model through the Model Optimizer, your output is an Intermediate Representation (IR) of the network. The IR is a pair of files that describe the whole model:

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

The Inference Engine reads, loads, and infers the IR files, using a common API across the CPU, GPU, or VPU hardware.

The Model Optimizer is a Python*-based command line tool (mo.py), which is located in **/opt/intel/computer_vision_sdk/deployment_tools/model_optimizer**.

Use this tool on models trained with popular deep learning frameworks such as Caffe*, TensorFlow*, MXNet*, and ONNX* to convert them to an optimized IR format that the Inference Engine can use.

This section explains how to use scripts to configure the Model Optimizer either for all of the supported frameworks at the same time or for individual frameworks. If you want to manually configure the Model Optimizer instead of using scripts, see the using manual configuration process section in the Model Optimizer Developer Guide.

For more information about the Model Optimizer, see the Model Optimizer Developer Guide.

Model Optimizer configuration steps

You can either configure the Model Optimizer for all supported frameworks at once, or for one framework at a time. Choose the option that best suits your needs. If you see error messages, make sure you installed all dependencies.

Note: If you did not install the Intel Distribution of OpenVINO toolkit to the default installation directory, replace /intel/ with the directory where you installed the software to.

Option 1: Configure the Model Optimizer for all supported frameworks at the same time:

1. **Go to the Model Optimizer** prerequisites directory:

```
cd
/opt/intel/computer_vision_sdk/deployment_tools/model_optimize
r/install_prerequisites
```

2. **Run the script** to configure the Model Optimizer for Caffe, TensorFlow, MXNet, Kaldi*, and ONNX:

```
sudo ./install_prerequisites.sh
```

Option 2: Configure the Model Optimizer for each framework separately:

1. **Go to the Model Optimizer** prerequisites directory:

```
cd
/opt/intel/computer_vision_sdk/deployment_tools/model_optimize
r/install_prerequisites
```

2. **Run the script** for your model framework. You can run more than one script:

o For Caffe:

```
sudo ./install_prerequisites_caffe.sh
```

o For TensorFlow:

```
sudo ./install_prerequisites_tf.sh
```

o For MXNet:

```
sudo ./install_prerequisites_mxnet.sh
```

o For ONNX:

```
sudo ./install_prerequisites_onnx.sh
```

o For Kaldi:

```
sudo ./install_prerequisites_kaldi.sh
```

The Model Optimizer is configured for one or more frameworks. You are ready to use two short demos to see the results of running the Intel Distribution of OpenVINO toolkit and to verify your installation was successful. The demo scripts are required since they perform additional configuration steps. Continue to the next section.

4.7 Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs Mustang-V100-MX4 Driver Installation

1. Install dependencies:

```
sudo apt install libusb-1.0-0 libboost-program-options1.58.0
libboost-thread1.58.0 libboost-filesystem1.58.0 libssl1.0.0
libudev1 libjson-c2
```

2. Add the current Linux user to the users group:

```
sudo usermod -a -G users "$(whoami)"
```

3. Log out and log in for it to take effect.

4. If your Intel® Vision Accelerator Design with Intel® Movidius™ VPUs card requires SMBUS connection to PCIe slot (Raw video data card with HW version Fab-B and before), generate rules for SMBUS controller and enable the i2c_i801 driver (Intel® SMBUS controller driver):

```
cd ${HDDL_INSTALL_DIR}
sudo chmod +x ./generate_udev_rules.sh
sudo ./generate_udev_rules.sh /etc/udev/rules.d/98-hddlbsl.rules
```

5. Check if the file /etc/modprobe.d/blacklist.conf contains the line "blacklist i2c_i801" and comment it out if so:

```
sudo sed -i "s/\(.*i2c_i801$\)/#\1/g"
/etc/modprobe.d/blacklist.conf

sudo modprobe i2c_i801
```

6. Before you start installing the drivers, make sure there is no inference application is running. Use the following command to make sure there is no HDDL service is running:

```
kill -9 $(pidof hddldaemon autoboot)
```

7. Install the drivers:

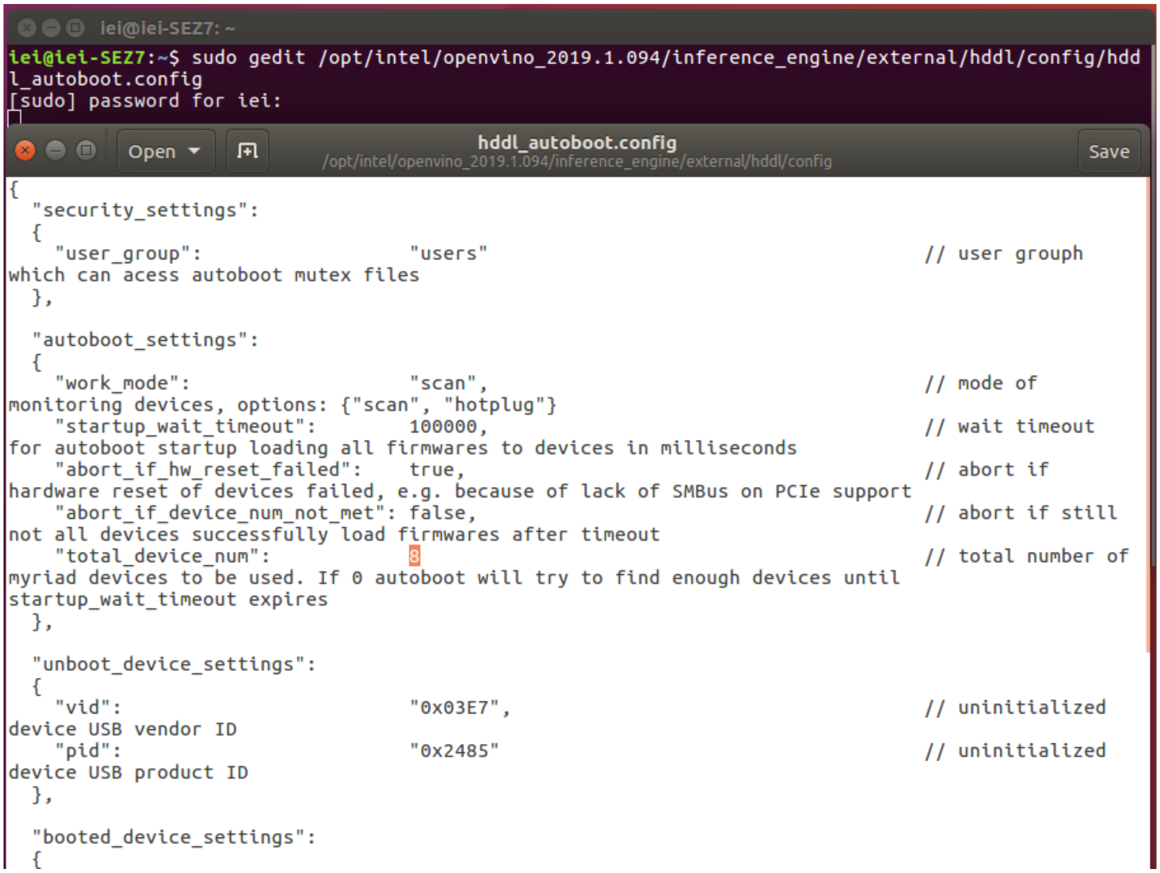
```
cd ${HDDL_INSTALL_DIR}/drivers
sudo chmod +x ./setup.sh
sudo ./setup.sh install
sudo cp -av ${HDDL_INSTALL_DIR}/../97-myriad-usbboot.rules
/etc/udev/rules.d/
sudo cp -av ${HDDL_INSTALL_DIR}/etc /
sudo udevadm control --reload-rules
sudo udevadm trigger
sudo ldconfig
```

Now, the Mustang-V100-MX4 drivers are installed.

8. Modify the device number on your host

   Change the total device numbers of “hddl_autoboot.config”.

   Each Mustang-V100-MX4 contains 4 devices, therefore, user has to change device number from default 8 to 4.


iei@iei-SEZ7: ~
Open
Save
hddl_autoboot.config
/opt/intel/openvino_2019.1.094/inference_engine/external/hddl/config

```
iei@iei-SEZ7:~$ sudo gedit /opt/intel/openvino_2019.1.094/inference_engine/external/hddl/config/hdd
l_autoboot.config
[sudo] password for iei:
```

```
{
  "security_settings":
  {
    "user_group":                 "users"                                   // user grouph
which can acess autoboot mutex files
  },

  "autoboot_settings":
  {
    "work_mode":                  "scan",                                   // mode of
monitoring devices, options: {"scan", "hotplug"}
    "startup_wait_timeout":       100000,                                   // wait timeout
for autoboot startup loading all firmwares to devices in milliseconds
    "abort_if_hw_reset_failed":   true,                                     // abort if
hardware reset of devices failed, e.g. because of lack of SMBus on PCIe support
    "abort_if_device_num_not_met": false,                                   // abort if still
not all devices successfully load firmwares after timeout
    "total_device_num":           8                                         // total number of
myriad devices to be used. If 0 autoboot will try to find enough devices until
startup_wait_timeout expires
  },

  "unboot_device_settings":
  {
    "vid":                        "0x03E7",                                 // uninitialized
device USB vendor ID
    "pid":                        "0x2485"                                  // uninitialized
device USB product ID
  },

  "booted_device_settings":
  {
```

4.8 Use the Demo Scripts to Verify Your Installation

Important: This section is required. In addition to confirming that your installation was successful, the demo scripts perform additional steps, such as setting up your computer to use the Model Optimizer samples.

To learn more about the demo applications, see README.txt in /opt/intel/oprnvino_<version>/deployment_tools/demo/.

For detailed description of the pre-trained object detection and object recognition models, go to /opt/intel/openvino_<version>/deployment_tools/intel_models/ and open index.html.

Note:

The paths in this section assume you used the default installation directory to install the Intel Distribution of OpenVINO toolkit. If you installed the software to a directory other than /opt/intel/, update the directory path with the location where you installed the toolkit to.

If you installed the product as a root user, you must switch to the root mode before you continue: sudo –i

4.8.1 Run the Image Classification Demo

1. Go to the Inference Engine demo directory:

```
cd /opt/intel/computer_vision_sdk/deployment_tools/demo
```

2. Set environment variables and run the **Image Classification** demo:

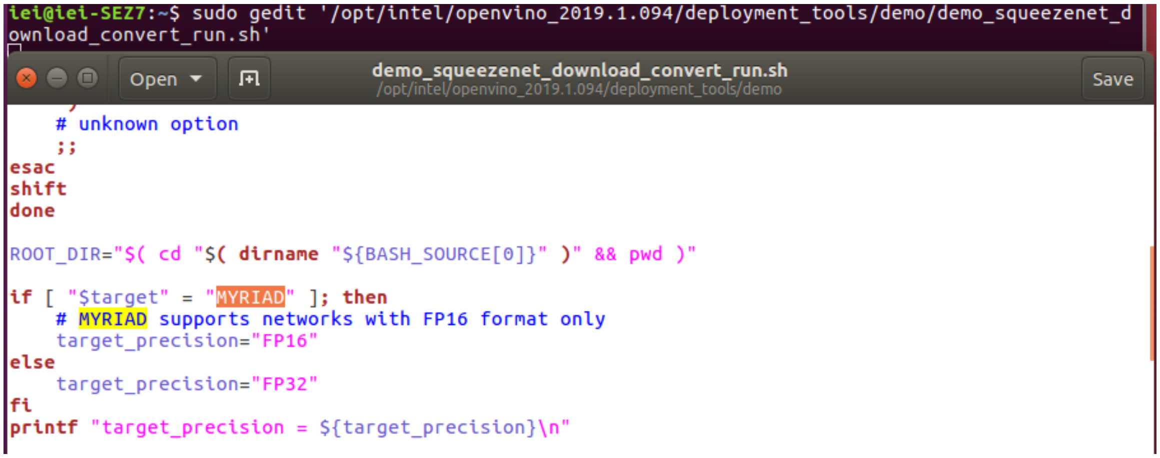
1) Set your environment variables:

```
source /opt/intel/computer_vision_sdk/bin/setupvars.sh
```

2) Modify .sh content:

```
sudo gedit demo_squeezenet_download_convert_run.sh
```

Search MYRIAD and modify to HDDL then save



```
iei@iei-SEZ7:~$ sudo gedit '/opt/intel/openvino_2019.1.094/deployment_tools/demo/demo_squeezenet_download_convert_run.sh'
```

Open
demo_squeezenet_download_convert_run.sh
/opt/intel/openvino_2019.1.094/deployment_tools/demo
Save

```
    # unknown option
    ;;
esac
shift
done

ROOT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

if [ "$target" = "MYRIAD" ]; then
    # MYRIAD supports networks with FP16 format only
    target_precision="FP16"
else
    target_precision="FP32"
fi
printf "target_precision = ${target_precision}\n"
```

3) Assign Mustang-V100-MX4 (-d HDDL) and run demo

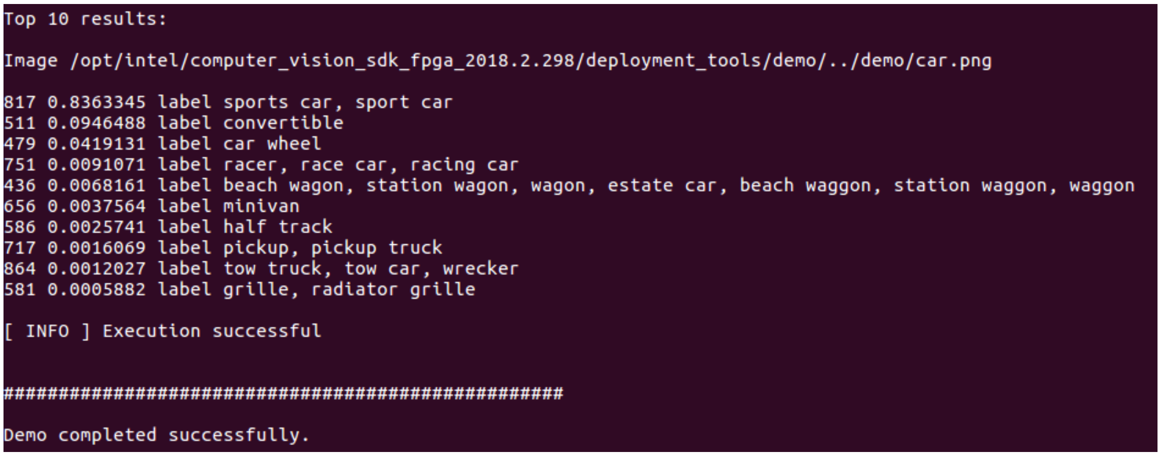
```
/demo_squeezenet_download_convert_run.sh -d HDDL
```

3. The Image Classification demo uses the Model Optimizer to convert a SqueezeNet model to .bin and .xml Intermediate Representation (IR) files. The Inference Engine component uses these files.

   For a brief description of the Intermediate Representation .bin and .xml files, see Configuring the Model Optimizer.

   This demo creates the directory /home/<user>/inference_engine_samples/.

   This demo uses car.png in the demo directory. When the demo completes, you will see the label and confidence for the top-10 categories:



```
Top 10 results:

Image /opt/intel/computer_vision_sdk_fpga_2018.2.298/deployment_tools/demo/../demo/car.png

817 0.8363345 label sports car, sport car
511 0.0946488 label convertible
479 0.0419131 label car wheel
751 0.0091071 label racer, race car, racing car
436 0.0068161 label beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
656 0.0037564 label minivan
586 0.0025741 label half track
717 0.0016069 label pickup, pickup truck
864 0.0012027 label tow truck, tow car, wrecker
581 0.0005882 label grille, radiator grille

[ INFO ] Execution successful


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

Demo completed successfully.
```

This demo is complete. Continue to the next section to run the Inference Pipeline demo.

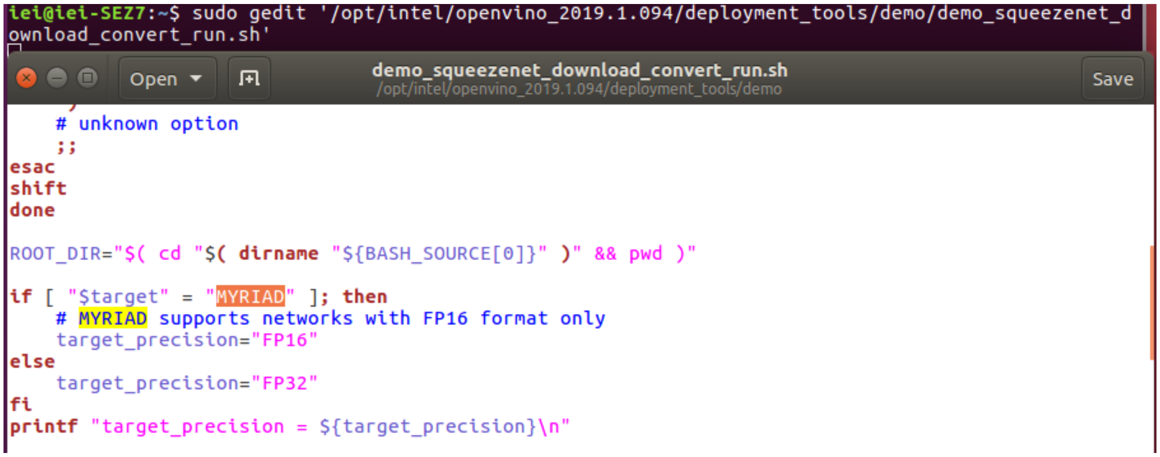
4.8.2 Run the Inference Pipeline Demo

1. While still in /opt/intel/computer_vision_sdk/deployment_tools/demo/, **run the Inference Pipeline demo**.

2. Modify .sh content:

```
sudo gedit demo_squeezenet_download_convert_run.sh
```

Search MYRIAD and modify to HDDL then save


Open
Save
demo_squeezenet_download_convert_run.sh
/opt/intel/openvino_2019.1.094/deployment_tools/demo

```
iei@iei-SEZ7:~$ sudo gedit '/opt/intel/openvino_2019.1.094/deployment_tools/demo/demo_squeezenet_d
ownload_convert_run.sh'
```

```
    # unknown option
    ;;
esac
shift
done

ROOT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

if [ "$target" = "MYRIAD" ]; then
    # MYRIAD supports networks with FP16 format only
    target_precision="FP16"
else
    target_precision="FP32"
fi
printf "target_precision = ${target_precision}\n"
```

3. Assign Mustang-V100-MX4 (-d HDDL) and run demo

```
   /demo_security_barrier_camera.sh –d HDDL
```

4. This demo uses car.png in /opt/intel/computer_vision_sdk/deployment_tools/demo/ to show an inference pipeline. This demo uses three pre-trained models. The demo uses vehicle recognition in which vehicle attributes build on each other to narrow in on a specific attribute. The demo works as follows:

   1) An object is identified as a vehicle.

   2) This identification is used as input to the next model, which identifies specific vehicle attributes, including the license plate.

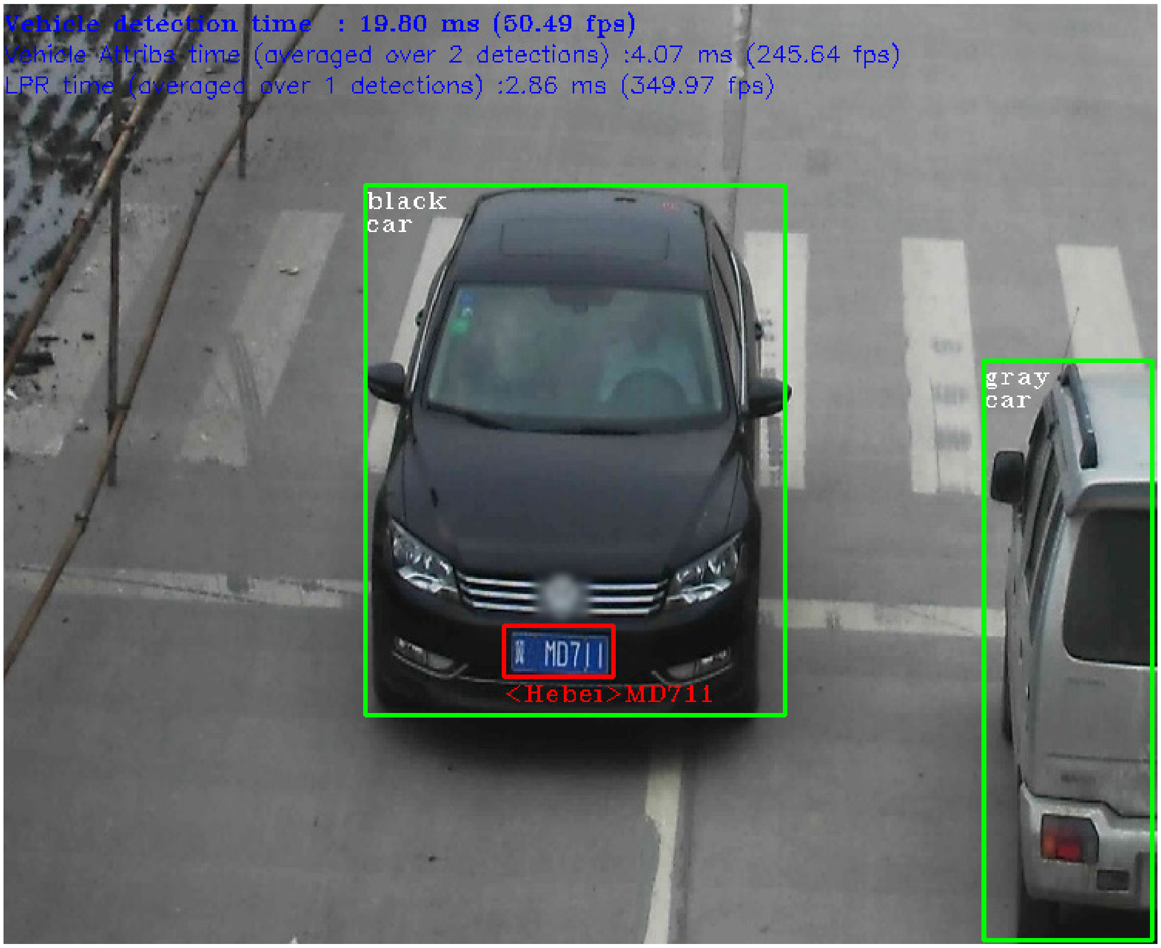
   3) The attributes identified as the license plate are used as input to the third model, which recognizes specific characters in the license plate.

   For more information about the demo, see the Security Camera Sample.

5. When the demo completes, two windows are open:

o A console window that displays information about the tasks performed by the demo

o An image viewer window that displays a picture similar to the following:



```
Vehicle detection time  : 19.80 ms (50.49 fps)
Vehicle Attribs time (averaged over 2 detections) :4.07 ms (245.64 fps)
LPR time (averaged over 1 detections) :2.86 ms (349.97 fps)
```

6. Close the image viewer screen to end the demo.

In this section, you saw a preview of the Intel Distribution of OpenVINO toolkit capabilities.

You have completed all the required installation, configuration, and build steps to work with your trained models using the Mustang-V100-MX4.

Chapter

5

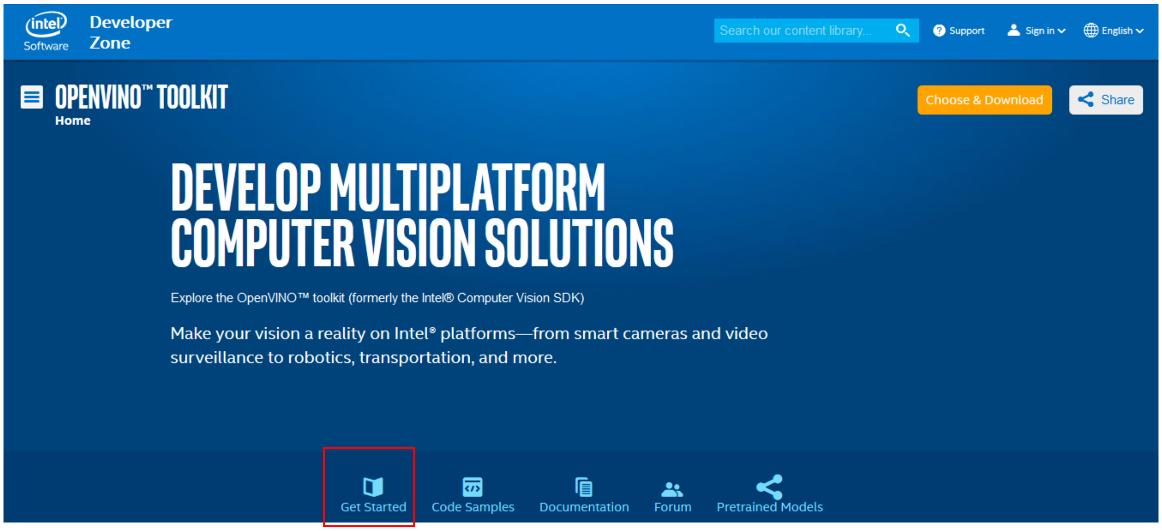
OpenVINO™ Toolkit Installation (Windows 10)

5.1 System Requirements

- Windows 10 64-bit

5.2 Online Installation Guide

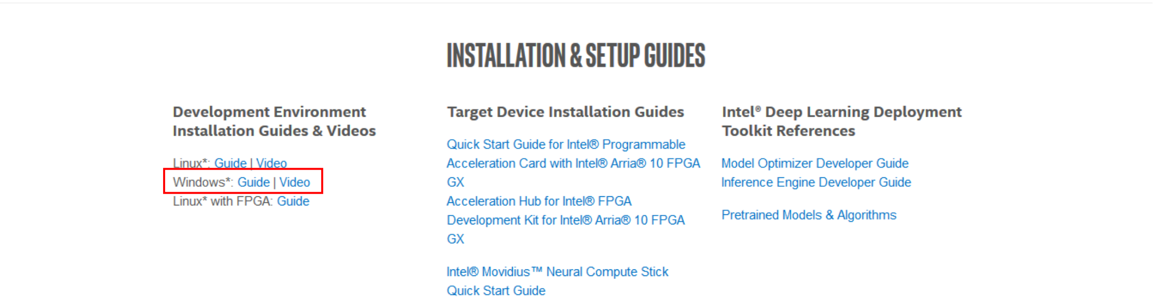
Go to https://software.intel.com/en-us/openvino-toolkit. Click "Get Started" then choose your configuration from "Development Environment Installation Guides & Videos". Follow the instruction to complete the installation procedure.



DEVELOP MULTIPLATFORM COMPUTER VISION SOLUTIONS

Explore the OpenVINO™ toolkit (formerly the Intel® Computer Vision SDK)

Make your vision a reality on Intel® platforms—from smart cameras and video surveillance to robotics, transportation, and more.



INSTALLATION & SETUP GUIDES

Development Environment Installation Guides & Videos

Linux*: Guide | Video
Windows*: Guide | Video
Linux* with FPGA: Guide

Target Device Installation Guides

Quick Start Guide for Intel® Programmable Acceleration Card with Intel® Arria® 10 FPGA GX

Acceleration Hub for Intel® FPGA Development Kit for Intel® Arria® 10 FPGA GX

Intel® Movidius™ Neural Compute Stick Quick Start Guide

Intel® Deep Learning Deployment Toolkit References

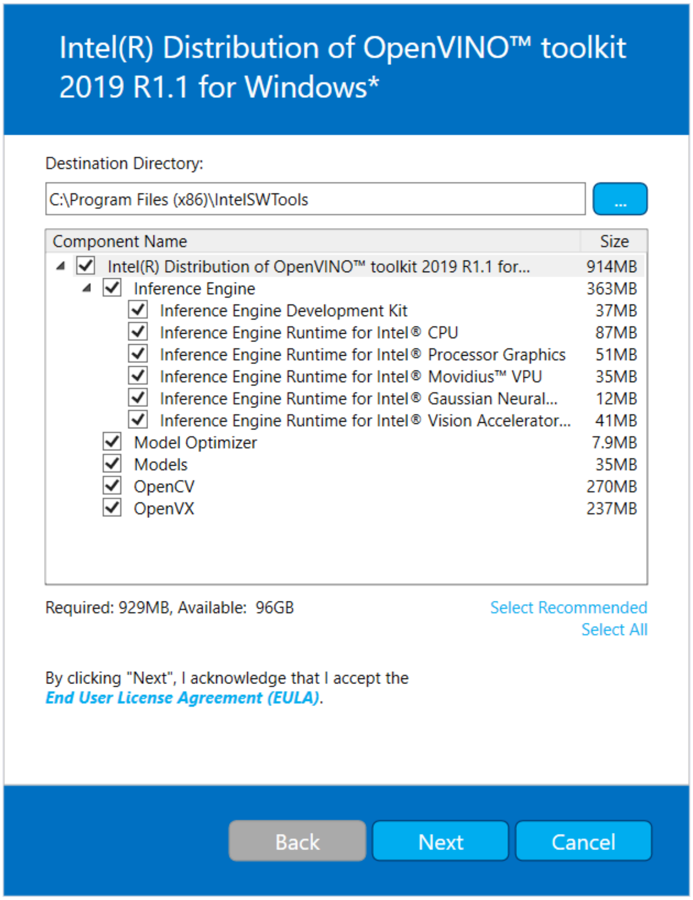
Model Optimizer Developer Guide
Inference Engine Developer Guide

Pretrained Models & Algorithms

5.3 Installation - Step by Step (2019 R1)

5.3.1 Install the Intel® Distribution of OpenVINO™ Toolkit Core Components

1. If you have not downloaded the Intel® Distribution of OpenVINO™ toolkit, download the latest version.. By default, the file is saved to the Downloads directory as w_openvino_toolkit_p_<version>.exe.
2. Go to the **Downloads** folder.
3. Double-click **w_openvino_toolkit_p_<version>.exe**. A window opens to let you choose your installation directory and components. The default installation directory is C:\Intel. If you choose a different installation directory, the installer will create the directory for you.



Intel(R) Distribution of OpenVINO™ toolkit 2019 R1.1 for Windows*

Destination Directory:

C:\Program Files (x86)\IntelSWTools

Component Name	Size
☑ Intel(R) Distribution of OpenVINO™ toolkit 2019 R1.1 for...	914MB
☑ Inference Engine	363MB
☑ Inference Engine Development Kit	37MB
☑ Inference Engine Runtime for Intel® CPU	87MB
☑ Inference Engine Runtime for Intel® Processor Graphics	51MB
☑ Inference Engine Runtime for Intel® Movidius™ VPU	35MB
☑ Inference Engine Runtime for Intel® Gaussian Neural...	12MB
☑ Inference Engine Runtime for Intel® Vision Accelerator...	41MB
☑ Model Optimizer	7.9MB
☑ Models	35MB
☑ OpenCV	270MB
☑ OpenVX	237MB

Required: 929MB, Available: 96GB

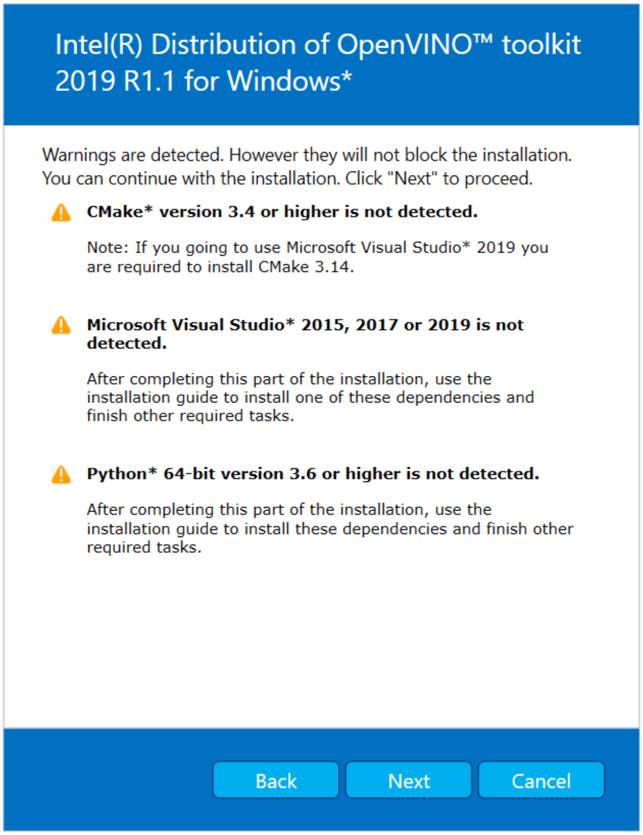
By clicking "Next", I acknowledge that I accept the *End User License Agreement (EULA)*.

4. Click **Next**.
5. You are asked if you want to provide consent to gather information. Choose the option

of your choice. Click **Next**.

6. If you are missing external dependencies, you will see a warning screen. Write down the dependencies you are missing. **You need to take no other action at this time**. After installing the Intel® Distribution of OpenVINO™ toolkit core components, you will be provided instructions to install the missing dependencies.

   The screen example below indicates you are missing two dependencies:



Intel(R) Distribution of OpenVINO™ toolkit 2019 R1.1 for Windows*

Warnings are detected. However they will not block the installation. You can continue with the installation. Click "Next" to proceed.

⚠ **CMake* version 3.4 or higher is not detected.**

Note: If you going to use Microsoft Visual Studio* 2019 you are required to install CMake 3.14.

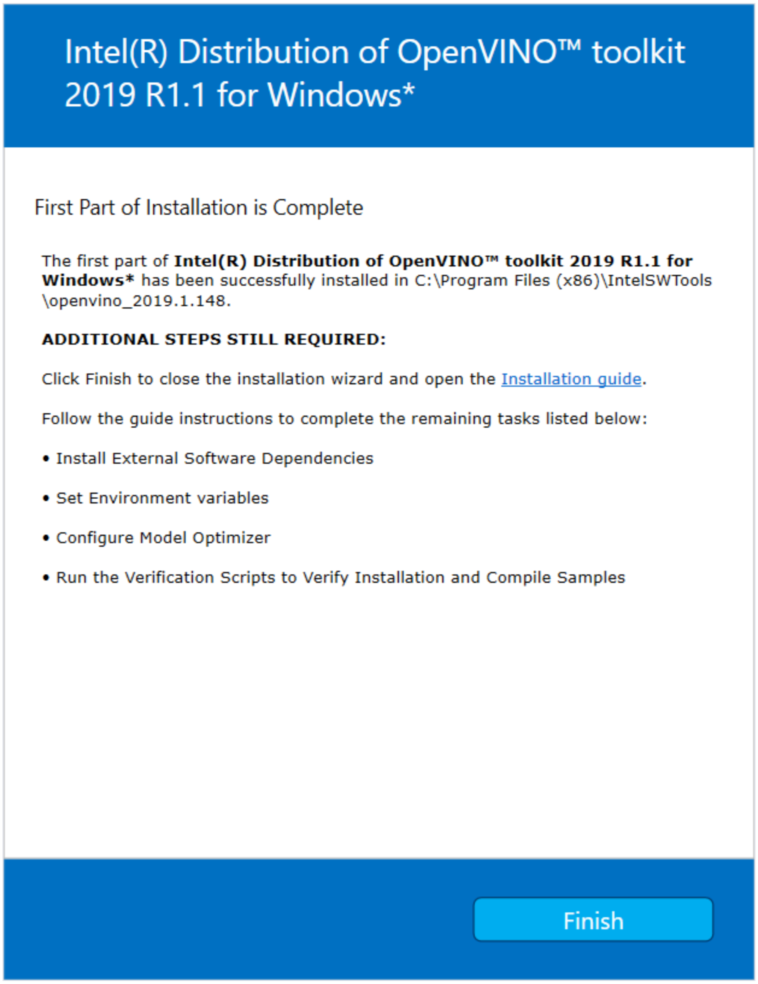
⚠ **Microsoft Visual Studio* 2015, 2017 or 2019 is not detected.**

After completing this part of the installation, use the installation guide to install one of these dependencies and finish other required tasks.

⚠ **Python* 64-bit version 3.6 or higher is not detected.**

After completing this part of the installation, use the installation guide to install these dependencies and finish other required tasks.

7. Click **Next**.
8. When the first part of installation is complete, the final screen informs you that the core components have been installed and additional steps still required:



Intel(R) Distribution of OpenVINO™ toolkit 2019 R1.1 for Windows*

First Part of Installation is Complete

The first part of **Intel(R) Distribution of OpenVINO™ toolkit 2019 R1.1 for Windows*** has been successfully installed in C:\Program Files (x86)\IntelSWTools \openvino_2019.1.148.

ADDITIONAL STEPS STILL REQUIRED:

Click Finish to close the installation wizard and open the Installation guide.

Follow the guide instructions to complete the remaining tasks listed below:

- Install External Software Dependencies
- Set Environment variables
- Configure Model Optimizer
- Run the Verification Scripts to Verify Installation and Compile Samples

9. Click **Finish** to close the installation wizard. A new browser window opens to the next section of the installation guide to install the dependencies. You are in the same document. The new window opens in case you ran the installation without first opening this installation guide. If the installation did not indicate you must install dependencies, you can skip ahead to set the environment variables..

5.3.2 Install the External Software Dependencies

If the installation process indicated if you are missing dependencies, you must install each missing dependency. Click the link for the first dependency you must install:

- Microsoft Visual Studio* with C++ 2019, 2017, or 2015 with MSBuild
- CMake 3.4 or higher
- Python 3.6.5

If you have no missing dependencies, skip ahead to Configure the Model Optimizer.

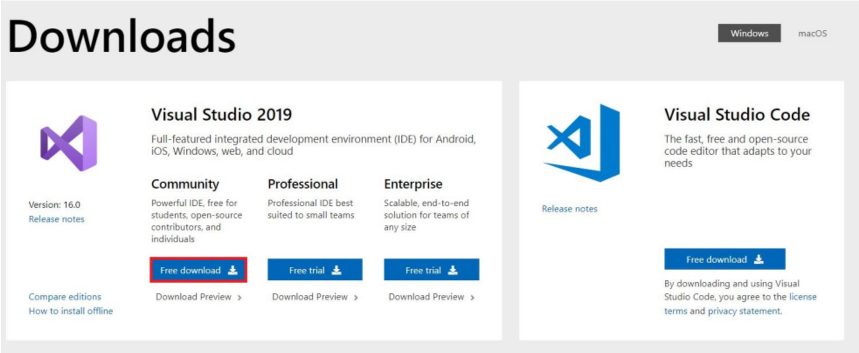
5.3.2.1 Microsoft Visual Studio* with C++ and MSBuild

Microsoft Visual Studio with Visual Studio C++ is required for building the Intel® Deep Learning Deployment Toolkit samples and demonstration applications. **You can install the free Community version of Microsoft Visual Studio**.

IMPORTANT: The Microsoft Visual Studio dependency is a two-part installation that consists of **Microsoft Visual Studio 2017 or 2015** and the **Microsoft Visual Studio Build Tools**. This guide includes steps for installing both parts of this dependency. These are separate installations. **MAKE SURE YOU INSTALL BOTH COMPONENTS**.

The steps below apply to Microsoft Visual Studio 2019. If you prefer to use Microsoft Visual Studio 2017 or 2015, see Installing Microsoft Visual Studio 2017 or 2015 for Intel® Distribution of OpenVINO™ toolkit.

1. Go to the Visual Studio downloads page.
2. Click **Free Download** in the **Visual Studio 2019** box, **Community** section:



Visual Studio 2019

Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

Community

Powerful IDE, free for students, open-source contributors, and individuals

Professional

Professional IDE best suited to small teams

Enterprise

Scalable, end-to-end solution for teams of any size

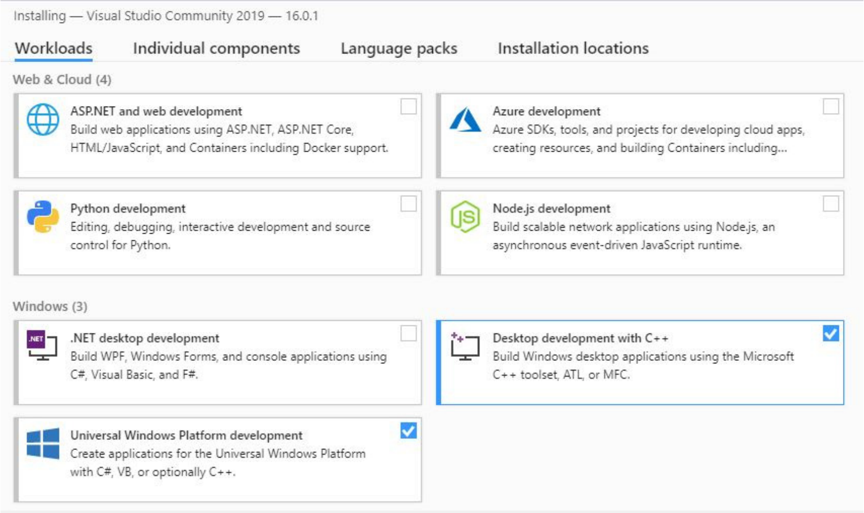
Visual Studio Code

The fast, free and open-source code editor that adapts to your needs

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

An executable file named **vs_community__313888930.1524151023.exe**, or similar, is saved in your **Downloads** folder.

3. Double-click the executable file to launch the **Visual Studio** installer.
4. On the opened window, click **Continue**. Visual Studio Installer configuration takes several minutes.
5. From the Workloads tab, use the check boxes to select **Universal Windows Platform development** and **Desktop development with C++**.



Installing — Visual Studio Community 2019 — 16.0.1

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

□ ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

□ Azure development
Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...

□ Python development
Editing, debugging, interactive development and source control for Python.

□ Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

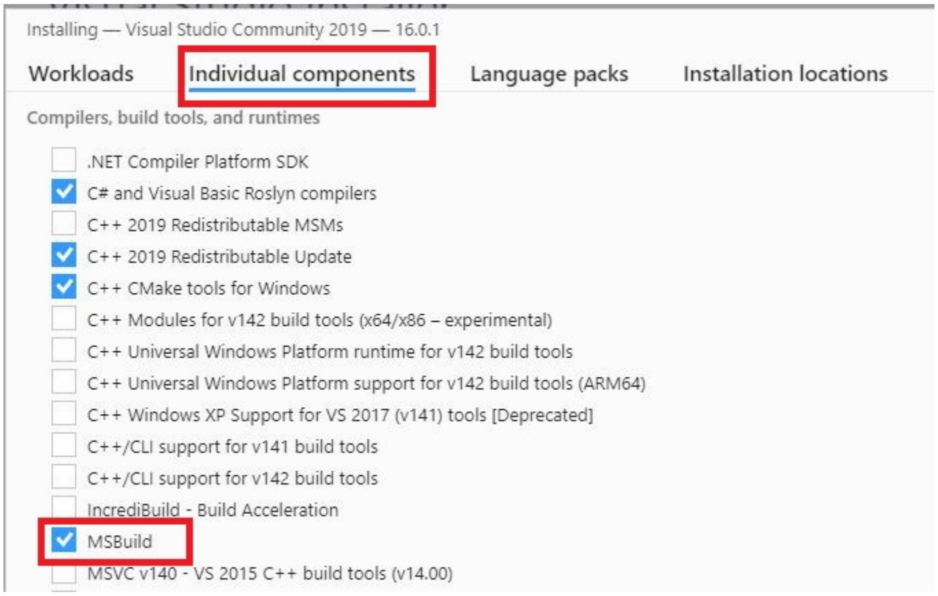
Windows (3)

□ .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

☑ Desktop development with C++
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

☑ Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

6. Under the **Individual components** tab, select **MSBuild**:



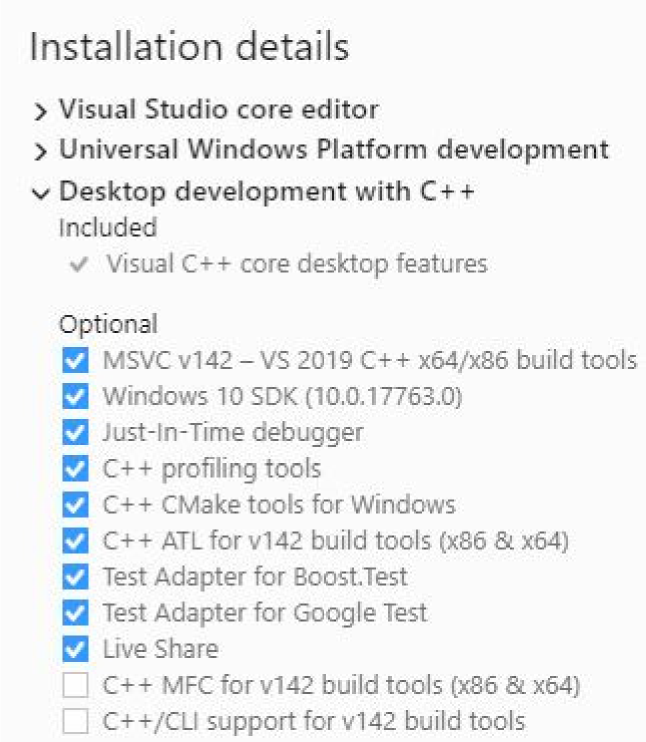
Installing — Visual Studio Community 2019 — 16.0.1

Workloads | Individual components | Language packs | Installation locations

Compilers, build tools, and runtimes

- □ .NET Compiler Platform SDK
- ☑ C# and Visual Basic Roslyn compilers
- □ C++ 2019 Redistributable MSMs
- ☑ C++ 2019 Redistributable Update
- ☑ C++ CMake tools for Windows
- □ C++ Modules for v142 build tools (x64/x86 – experimental)
- □ C++ Universal Windows Platform runtime for v142 build tools
- □ C++ Universal Windows Platform support for v142 build tools (ARM64)
- □ C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]
- □ C++/CLI support for v141 build tools
- □ C++/CLI support for v142 build tools
- □ IncrediBuild - Build Acceleration
- ☑ MSBuild
- □ MSVC v140 - VS 2015 C++ build tools (v14.00)

The **Installation details** at the right side of the screen displays your installation selections:



Installation details

> Visual Studio core editor

> Universal Windows Platform development

⌄ Desktop development with C++

Included

✓ Visual C++ core desktop features

Optional

☑ MSVC v142 – VS 2019 C++ x64/x86 build tools
☑ Windows 10 SDK (10.0.17763.0)
☑ Just-In-Time debugger
☑ C++ profiling tools
☑ C++ CMake tools for Windows
☑ C++ ATL for v142 build tools (x86 & x64)
☑ Test Adapter for Boost.Test
☑ Test Adapter for Google Test
☑ Live Share
□ C++ MFC for v142 build tools (x86 & x64)
□ C++/CLI support for v142 build tools

7. Make no other changes. Click **Install**. The installation begins and takes around 30 minutes to complete.
8. If you see a prompt to restart your computer after the installation completes, dismiss it.

You have completed the Microsoft Visual Studio 2019 installation.
Install your next dependency:

- CMake
- Python

If you have installed all the dependencies, you are ready to set the environment variables.

5.3.2.2 Install CMake* 3.4 or higher

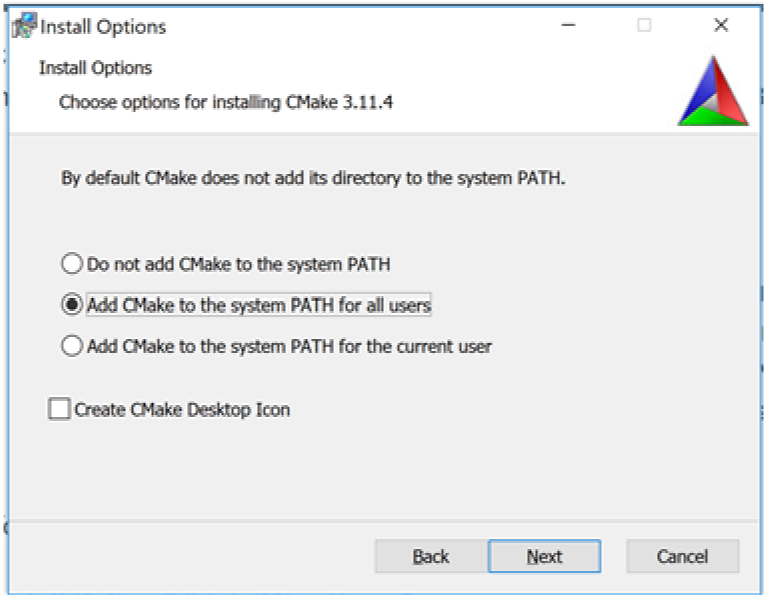
These steps guide you through installing CMake 3.4 or higher, which is required to build the Intel® Distribution of OpenVINO™ toolkit samples.

NOTE: If you want to use Microsoft Visual Studio 2019, you are required to install CMake 3.14.

1. Go to the CMake download site.
2. Under the heading **Get the Software**, click the link for **latest stable** in the first paragraph. Your screen displays **Latest Release** information.
3. Scroll down to the line **Windows win64-x64 Installer**.
4. Click the associated file name to download the installer. The file name will have the extension .msi. The file is saved to your Downloads folder.
5. Go to the Downloads folder.
6. Double-click the file to launch the installer.

NOTE: If you have a previous version of CMake installed, you are prompted to uninstall it. You must uninstall the previous version before installing the new version. Follow the instructions on the screen and then launch the installer again to install the new version.

7. In the installer, select the option to **Add CMake to the system PATH for all users**:



Install Options

Install Options

Choose options for installing CMake 3.11.4

By default CMake does not add its directory to the system PATH.

○ Do not add CMake to the system PATH

● Add CMake to the system PATH for all users

○ Add CMake to the system PATH for the current user

□ Create CMake Desktop Icon

8. Click **Next**.

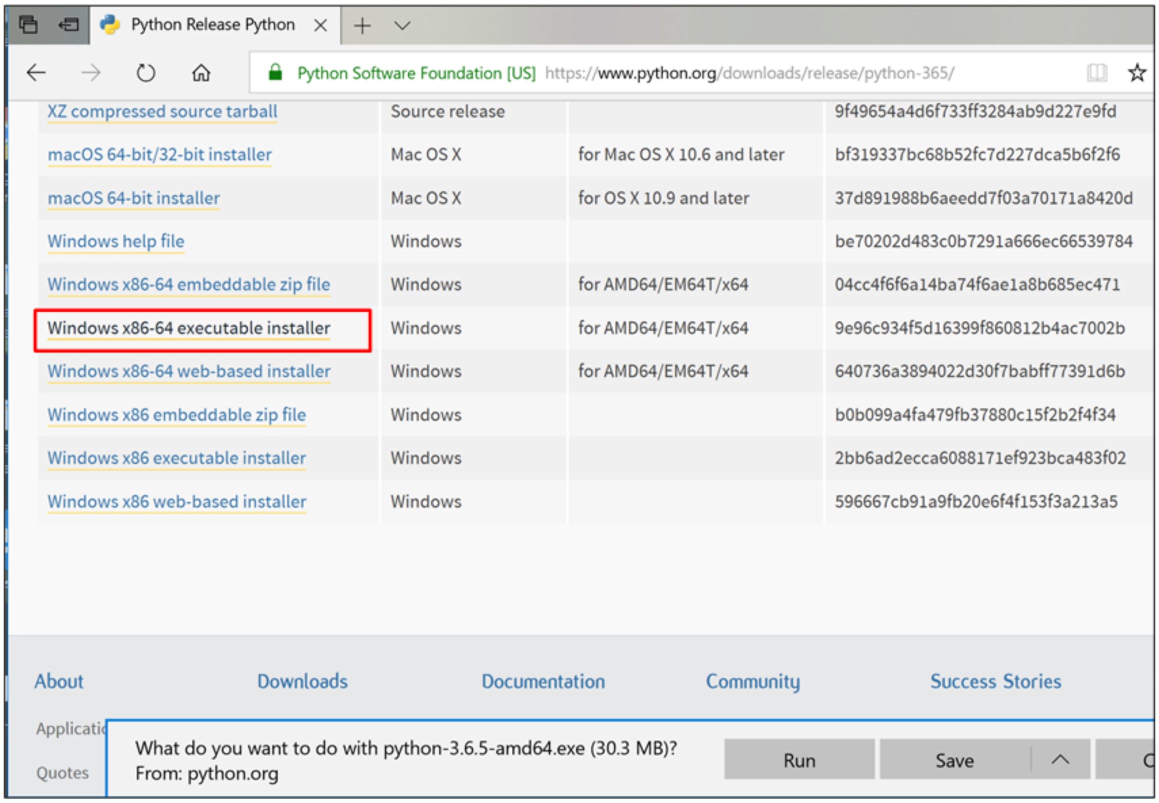
9. Click **Finish** when the installation completes.

You have completed the CMake installation. Next, install Python 3.6.5 if the Intel® Distribution of OpenVINO™ toolkit installation indicated you are missing the software.

5.3.2.3 Install Python* 3.6.5

Python 3.6.5 with pip is required to run the Model Optimizer. Use these steps to install the correct version of the Python software.

1. Go to the Python 3.6.5 download page and click **Windows x86-64 executable installer** to download the executable file. The file is saved as python-3.6.5-amd64.exe in your Downloads folder.



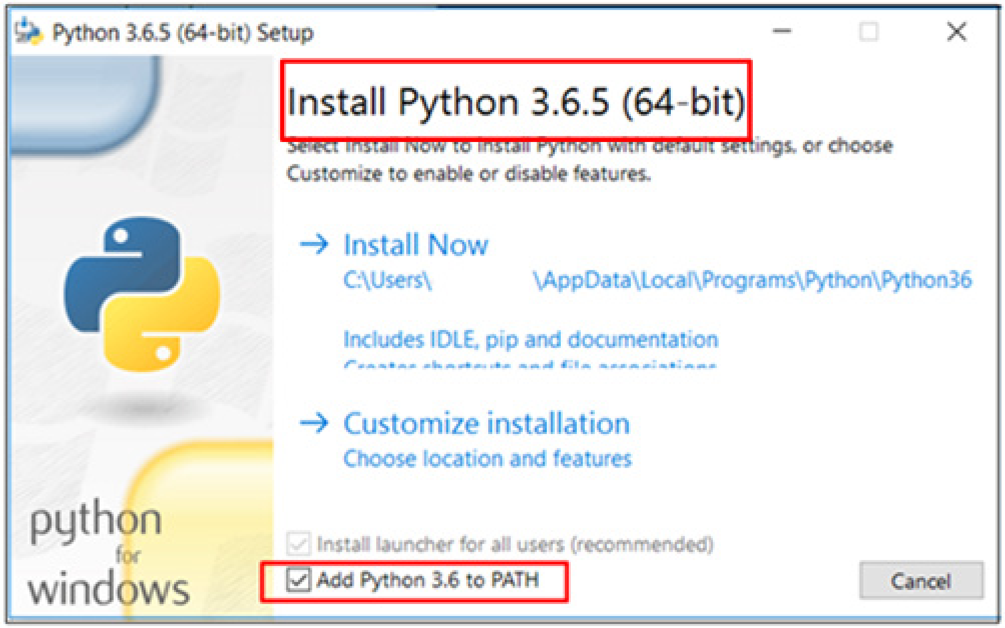
Python Release Python
Python Software Foundation [US] https://www.python.org/downloads/release/python-365/

XZ compressed source tarball	Source release		9f49654a4d6f733ff3284ab9d227e9fd
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	bf319337bc68b52fc7d227dca5b6f2f6
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	37d891988b6aeedd7f03a70171a8420d
Windows help file	Windows		be70202d483c0b7291a666ec66539784
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	04cc4f6f6a14ba74f6ae1a8b685ec471
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	9e96c934f5d16399f860812b4ac7002b
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	640736a3894022d30f7babff77391d6b
Windows x86 embeddable zip file	Windows		b0b099a4fa479fb37880c15f2b2f4f34
Windows x86 executable installer	Windows		2bb6ad2ecca6088171ef923bca483f02
Windows x86 web-based installer	Windows		596667cb91a9fb20e6f4f153f3a213a5

What do you want to do with python-3.6.5-amd64.exe (30.3 MB)?
From: python.org

2. Double-click the file to launch the installation.
3. Make sure the top of the screen shows **Python 3.6.5 (64-bit)**.

4. **IMPORTANT**: At the bottom of the install screen, select **Add Python 3.6 to PATH**.



Install Python 3.6.5 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\ \AppData\Local\Programs\Python\Python36

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

5. Click **Install Now** near the top of the install screen and let the installation complete.
6. When the installation finishes, click **Close**.

You have completed the Python installation and are ready to set environment variables. Continue to the next section.

5.3.3 Set the Environment Variables

You must update several environment variables before you can compile and run OpenVINO™ applications. Open the Command Prompt, and run the setupvars.bat batch file to temporarily set your environment variables:

```
cd C:\Program Files (x86)\IntelSWTools\openvino\bin\
setupvars.bat
```

(**Optional**): OpenVINO toolkit environment variables are removed when you close the Command Prompt window. As an option, you can permanently set the environment variables manually.

The environment variables are set. Continue to the next section to configure the Model Optimizer.

5.3.4 Configure the Model Optimizer

Important: These steps are required. You must configure the Model Optimizer for at least one framework. The Model Optimizer will fail if you do not complete the steps in this section.

If you see an error indicating Python is not installed when you know you installed it, your computer might not be able to find the program. For instructions to add Python to your system environment variables, see Update Your Windows Environment Variables.

The Model Optimizer is a key component of the Intel® Distribution of OpenVINO™ toolkit. You cannot do inference on your trained model without running the model through the Model Optimizer. When you run a pre-trained model through the Model Optimizer, your output is an Intermediate Representation (IR) of the network. The IR is a pair of files that describe the whole model:

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

The Inference Engine reads, loads, and infers the IR files, using a common API across the CPU, GPU, or VPU hardware.

The Model Optimizer is a Python*-based command line tool (mo.py), which is located in **C:\Intel\computer_vision_sdk_<version>\deployment_tools\model_optimizer**, where <version> is the version of the Intel® Distribution of OpenVINO™ toolkit that you installed. Use this tool on models trained with popular deep learning frameworks such as Caffe, TensorFlow, MXNet, and ONNX to convert them to an optimized IR format that the Inference Engine can use.

This section explains how to use scripts to configure the Model Optimizer either for all of the supported frameworks at the same time or for individual frameworks. If you want to manually configure the Model Optimizer instead of using scripts, see the Using Manual Configuration Process section in the Model Optimizer Developer Guide.

For more information about the Model Optimizer, see the Model Optimizer Developer Guide.

5.3.4.1 Model Optimizer Configuration Steps

You can configure the Model Optimizer either for all supported frameworks at once or for one framework at a time. Choose the option that best suits your needs. If you see error messages, make sure you installed all dependencies.

IMPORTANT: The Internet access is required to execute the following steps successfully. If you have access to the Internet through the proxy server only, please make sure that it is configured in your environment.

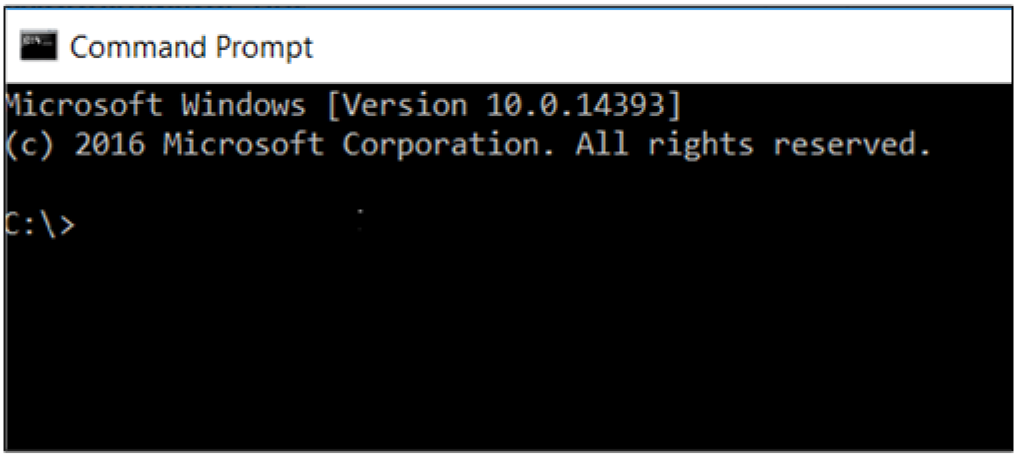
NOTE: In the steps below:

- If you you want to use the Model Optimizer from another installed versions of Intel® Distribution of OpenVINO™ toolkit installed, replace openvino with openvino_<version>.
- If you installed the Intel® Distribution of OpenVINO™ toolkit to the non-default installation directory, replace C:\Program Files (x86)\IntelSWTools with the directory where you installed the software.

These steps use a command prompt to make sure you see error messages.

Option 1: Configure the Model Optimizer for all supported frameworks at the same time:

1. Open a command prompt. To do so, type `cmd` in your **Search Windows** box and then press **Enter**. Type commands in the opened window:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\>
```

2. Go to the Model Optimizer prerequisites directory.

   ```
   cd C:\Program Files
   (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\i
   nstall_prerequisites
   ```

3. Run the following batch file to configure the Model Optimizer for Caffe*, TensorFlow*, MXNet*, Kaldi*, and ONNX*::

   ```
   install_prerequisites.bat
   ```

Option 2: Configure the Model Optimizer for each framework separately:

1. Go to the Model Optimizer prerequisites directory:

   ```
   cd C:\Program Files
   (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\i
   nstall_prerequisites
   ```

2. Run the batch file for the framework you will use with the Model Optimizer. You can use more than one:

   o For Caffe:

   ```
   install_prerequisites_caffe.bat
   ```

   o For TensorFlow:

   ```
   install_prerequisites_tf.bat
   ```

   o For MXNet:

   ```
   install_prerequisites_mxnet.bat
   ```

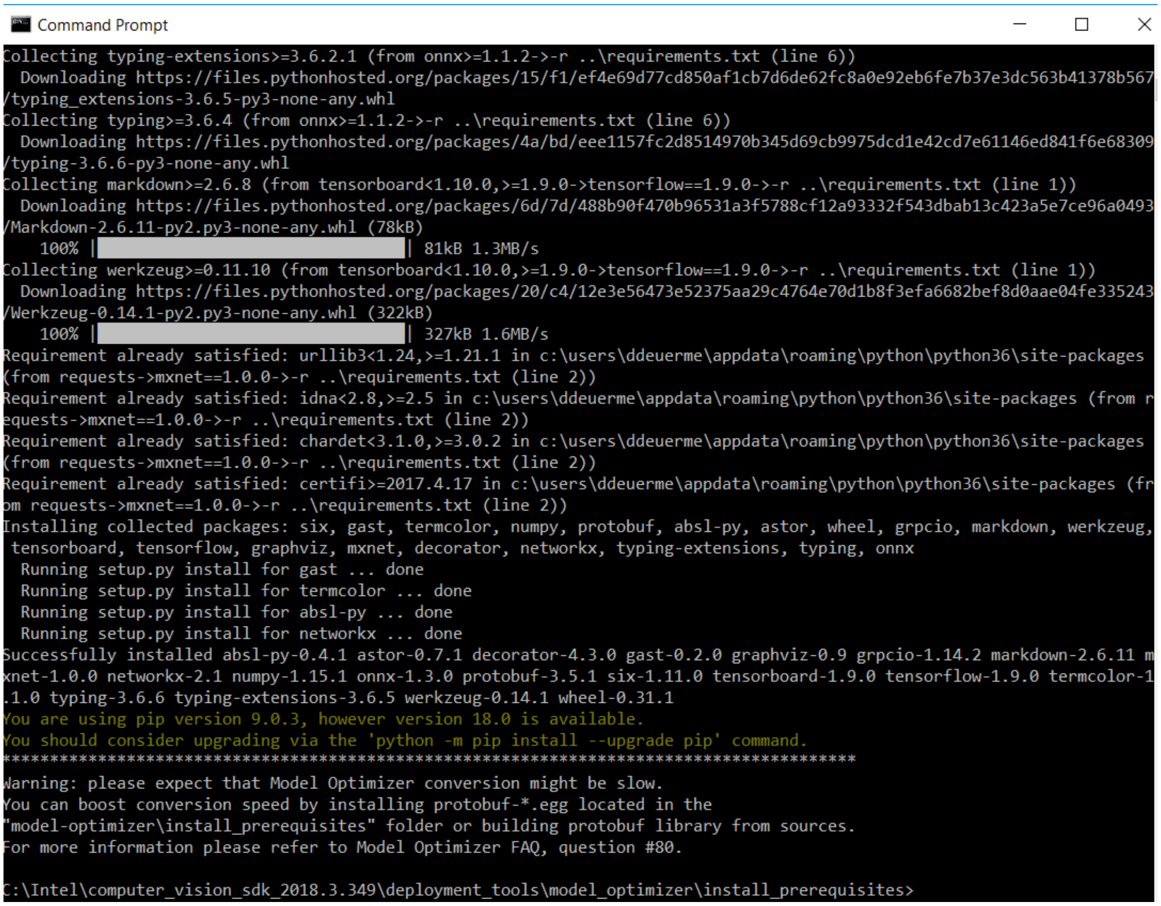
- For ONNX:

  ```
  install_prerequisites_onnx.bat
  ```

- For Kaldi:

  ```
  install_prerequisites_kaldi.bat
  ```

The Model Optimizer is configured for one or more frameworks. Success is indicated by a screen similar to this:



```
Collecting typing-extensions>=3.6.2.1 (from onnx>=1.1.2->-r ..\requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/15/f1/ef4e69d77cd850af1cb7d6de62fc8a0e92eb6fe7b37e3dc563b41378b567
/typing_extensions-3.6.5-py3-none-any.whl
Collecting typing>=3.6.4 (from onnx>=1.1.2->-r ..\requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/4a/bd/eee1157fc2d8514970b345d69cb9975dcd1e42cd7e61146ed841f6e68309
/typing-3.6.6-py3-none-any.whl
Collecting markdown>=2.6.8 (from tensorboard<1.10.0,>=1.9.0->tensorflow==1.9.0->-r ..\requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/6d/7d/488b90f470b96531a3f5788cf12a93332f543dbab13c423a5e7ce96a0493
/Markdown-2.6.11-py2.py3-none-any.whl (78kB)
    100% |████████████████████████████████| 81kB 1.3MB/s
Collecting werkzeug>=0.11.10 (from tensorboard<1.10.0,>=1.9.0->tensorflow==1.9.0->-r ..\requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243
/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
    100% |████████████████████████████████| 327kB 1.6MB/s
Requirement already satisfied: urllib3<1.24,>=1.21.1 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages
(from requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: idna<2.8,>=2.5 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages (from r
equests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages
(from requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages (fr
om requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Installing collected packages: six, gast, termcolor, numpy, protobuf, absl-py, astor, wheel, grpcio, markdown, werkzeug,
 tensorboard, tensorflow, graphviz, mxnet, decorator, networkx, typing-extensions, typing, onnx
  Running setup.py install for gast ... done
  Running setup.py install for termcolor ... done
  Running setup.py install for absl-py ... done
  Running setup.py install for networkx ... done
Successfully installed absl-py-0.4.1 astor-0.7.1 decorator-4.3.0 gast-0.2.0 graphviz-0.9 grpcio-1.14.2 markdown-2.6.11 m
xnet-1.0.0 networkx-2.1 numpy-1.15.1 onnx-1.3.0 protobuf-3.5.1 six-1.11.0 tensorboard-1.9.0 tensorflow-1.9.0 termcolor-1
.1.0 typing-3.6.6 typing-extensions-3.6.5 werkzeug-0.14.1 wheel-0.31.1
You are using pip version 9.0.3, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
******************************************************************************************
Warning: please expect that Model Optimizer conversion might be slow.
You can boost conversion speed by installing protobuf-*.egg located in the
"model-optimizer\install_prerequisites" folder or building protobuf library from sources.
For more information please refer to Model Optimizer FAQ, question #80.

C:\Intel\computer_vision_sdk_2018.3.349\deployment_tools\model_optimizer\install_prerequisites>
```

You are ready to use two short demos to see the results of running the Intel Distribution of OpenVINO toolkit and to verify your installation was successful. The demo scripts are required since they perform additional configuration steps. Continue to the next section.

5.4 Installation Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs Mustang-V100-MX4

To perform inference on Intel® Vision Accelerator Design with Intel® Movidius™ VPUs, the following additional installation steps are required:

1. Install the Movidius™ VSC driver:

a. Go to the <INSTALL_DIR>\deployment_tools\inference-engine\external\MovidiusDriver directory, where <INSTALL_DIR> is the directory in which the Intel Distribution of OpenVINO toolkit is installed.

b. Right click on the Movidius_VSC_Device.inf file and choose **Install** from the pop up menu.

2. If your Intel® Vision Accelerator Design with Intel® Movidius™ VPUs card requires SMBUS connection to PCIe slot (Raw video data card with HW version Fab-B and before), install the SMBUS driver:

a. Go to the <INSTALL_DIR>\deployment_tools\inference-engine\external\hddl\SMBusDriver directory, where <INSTALL_DIR> is the directory in which the Intel Distribution of OpenVINO toolkit is installed.

b. Right click on the hddlsmbus.inf file and choose **Install** from the pop up menu.

3. Download and install Visual C++ Redistributable for Visual Studio 2015

You are done installing your device driver and are ready to use Mustang-V100-MX4.

5.5 Use the Demo Scripts to Verify Your Installation

Important: This section is required. In addition to confirming your installation was successful, demo scripts perform other steps, such as setting up your computer to use the Model Optimizer samples.

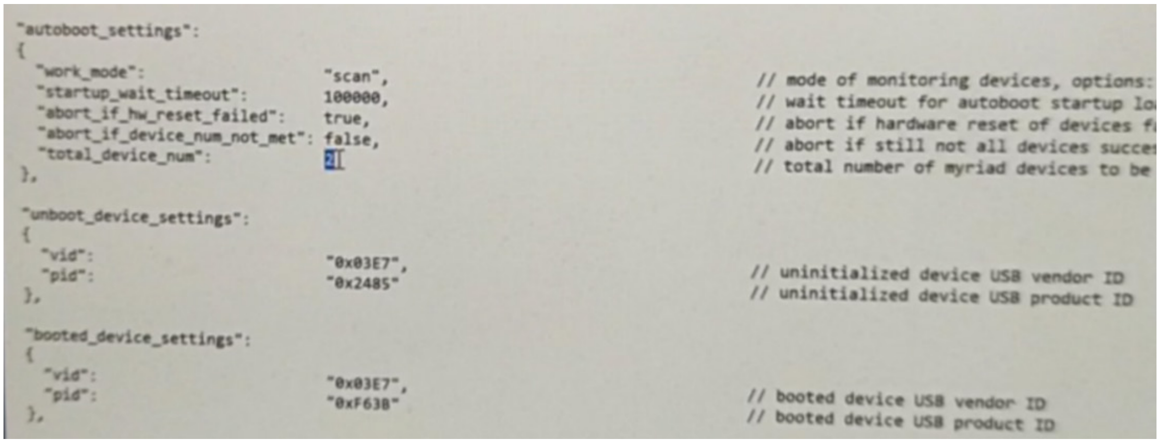
Note: To run the demo applications on Intel® Processor Graphics, Intel® Movidius™ Neural Compute Stick or Intel® Neural Compute Stick 2, make sure you completed the Additional Installation Steps first.

To learn more about the demo applications, see `README.txt` in `C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\`.

For detailed description of the pre-trained object detection and object recognition models, go to `C:\Intel\computer_vision_sdk_<version>\deployment_tools\intel_models\` and open `index.html`.

Notes:
- The paths in this section assume you used the default installation directory. If you used a directory other than C:\Intel, update the directory with the location where you installed the software.
- If you are migrating from the Intel® Computer Vision SDK 2017 R3 Beta version to the Intel® Distribution of OpenVINO™ toolkit, read this information about porting your applications.

1. Go to modify device number of VPU from 8 to 2, in C:\Program Files (x86)\IntelSWtools\openvino_<version>\`deployment_tools\inference_engine\external\hddl\config\hddl_autoboot.config`



```
"autoboot_settings":
{
  "work_mode":                    "scan",                    // mode of monitoring devices, options:
  "startup_wait_timeout":         100000,                    // wait timeout for autoboot startup lo
  "abort_if_hw_reset_failed":     true,                      // abort if hardware reset of devices f
  "abort_if_device_num_not_met":  false,                     // abort if still not all devices succe
  "total_device_num":             2                          // total number of myriad devices to be
},

"unboot_device_settings":
{
  "vid":                          "0x03E7",                  // uninitialized device USB vendor ID
  "pid":                          "0x2485"                   // uninitialized device USB product ID
},

"booted_device_settings":
{
  "vid":                          "0x03E7",                  // booted device USB vendor ID
  "pid":                          "0xF63B"                   // booted device USB product ID
},
```

1. Open a command prompt window.

2. Go to the Inference Engine demo directory:

```
cd
C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\
```

3. Run the demos by following the instructions in the next two sections.

5.5.1 Run the Image Classification Demo

5.5.2 Set Up a Neural Network Model

If you are running inference on hardware other than VPU-based devices, you already have the required FP32 neural network model converted to an optimized Intermediate Representation (IR). Follow the steps in the Run the Sample Application section to run the sample.

If you want to run inference on a VPU device (Intel® Movidius™ Neural Compute Stick, Intel® Neural Compute Stick 2 or Intel® Vision Accelerator Design with Intel® Movidius™ VPU), you'll need an FP16 version of the model, which you will set up in this paragraph.

To convert the FP32 model to a FP16 IR suitable for VPU-based hardware accelerators, follow the steps below:

1. Create a directory for the FP16 SqueezeNet Model, for example, `C:\Users\<username>\Documents\squeezenet1.1_FP16`
2. Open the Command Prompt and run the Model Optimizer to convert the FP32 Squeezenet Caffe* model delevered with the installation into an optimized FP16 Intermediate Representation (IR):

```
python3 "C:\Program Files
(x86)\IntelSWTools\openvino\deployment_tools\model_optimizer
\mo.py" --input_model
"C:\Users<username>\Documents\Intel\OpenVINO\openvino_models
```

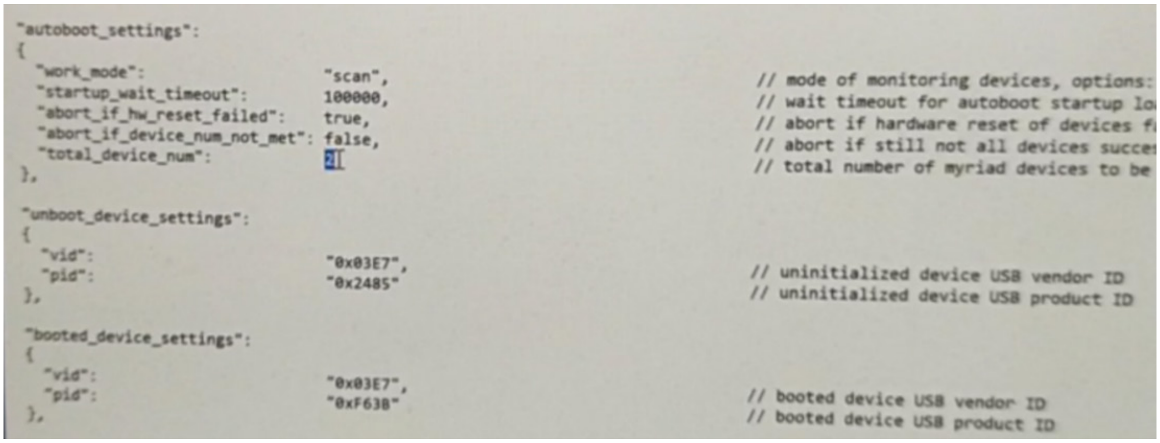
```
\models\FP32\classification\squeezenet\1.1\caffe\squeezenet1
.1.caffemodel" --data_type FP16 --output_dir
"C:\Users<username>\Documents\squeezenet1.1_FP16"
```

3. The `squeezenet1.1.labels` file contains the classes that ImageNet uses. This file is included so that the inference results show text instead of classification numbers. Copy `squeezenet1.1.labels` to your optimized model location:

   ```
   copy
   "C:\Users<username>\Documents\Intel\OpenVINO\openvino_models
   \ir\FP32\classification\squeezenet\1.1\caffe\squeezenet1.1.l
   abels" "C:\Users<username>\Documents\squeezenet1.1_FP16"
   ```

Now your neural network setup is complete and you're ready to run the sample application.

1. Go to modify device number of VPU from 8 to 2, in C:\Program Files (x86)\IntelSWtools\openvino_<version>\`deployment_tools\inference_engine\external\hddl\config\hddl_autoboot.config`



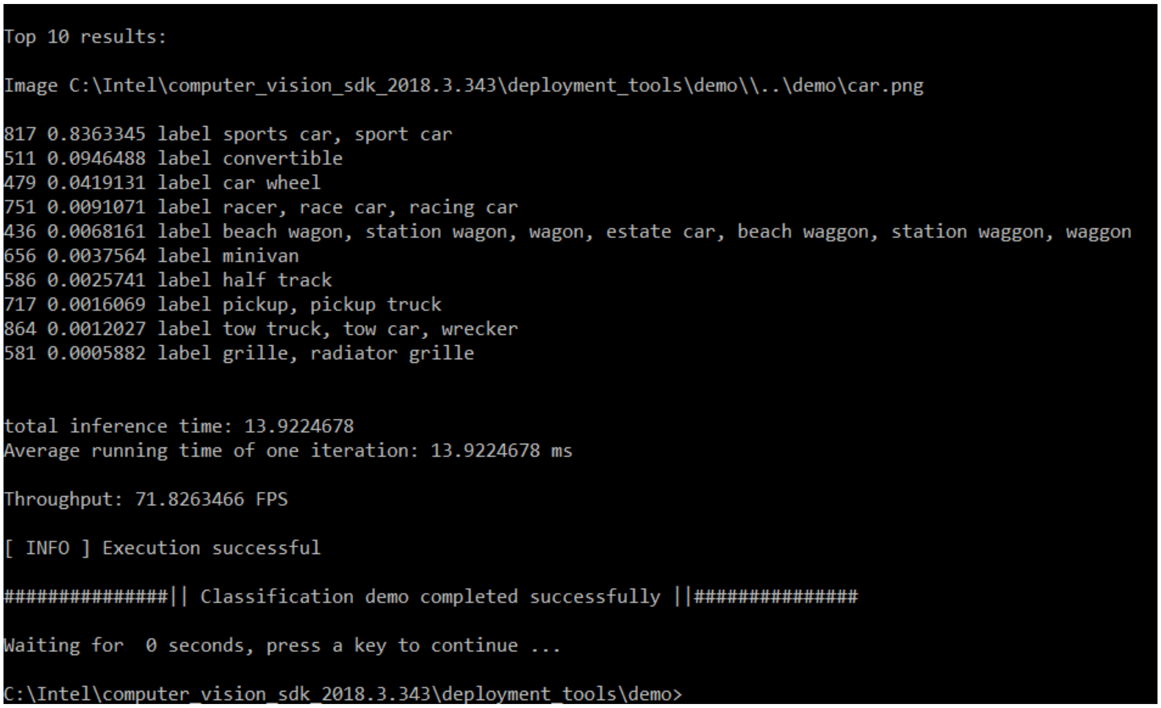
```
"autoboot_settings":
{
  "work_mode":                     "scan",                              // mode of monitoring devices, options:
  "startup_wait_timeout":          100000,                              // wait timeout for autoboot startup lo
  "abort_if_hw_reset_failed":      true,                                // abort if hardware reset of devices f
  "abort_if_device_num_not_met":   false,                               // abort if still not all devices succes
  "total_device_num":              2                                    // total number of myriad devices to be
},

"unboot_device_settings":
{
  "vid":                           "0x03E7",                            // uninitialized device USB vendor ID
  "pid":                           "0x2485"                             // uninitialized device USB product ID
},

"booted_device_settings":
{
  "vid":                           "0x03E7",                            // booted device USB vendor ID
  "pid":                           "0xF63B"                             // booted device USB product ID
},
```

2. Run the sample executable with specifying the car.png file from the demo directory as an input image, the IR of your FP16 model and a plugin for a hardware device to perform inference on.

```
classification_sample.exe -i "C:\Program Files
(x86)\IntelSWTools\openvino\deployment_tools\demo\car.png" -m
"C:\Users<username>\Documents\squeezenet1.1_FP16\squeezenet1.1
.xml" -d HDDL
```



```
Top 10 results:

Image C:\Intel\computer_vision_sdk_2018.3.343\deployment_tools\demo\\..\demo\car.png

817 0.8363345 label sports car, sport car
511 0.0946488 label convertible
479 0.0419131 label car wheel
751 0.0091071 label racer, race car, racing car
436 0.0068161 label beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
656 0.0037564 label minivan
586 0.0025741 label half track
717 0.0016069 label pickup, pickup truck
864 0.0012027 label tow truck, tow car, wrecker
581 0.0005882 label grille, radiator grille


total inference time: 13.9224678
Average running time of one iteration: 13.9224678 ms

Throughput: 71.8263466 FPS

[ INFO ] Execution successful

###############|| Classification demo completed successfully ||###############

Waiting for  0 seconds, press a key to continue ...

C:\Intel\computer_vision_sdk_2018.3.343\deployment_tools\demo>
```

Note: performance here is only for reference, it should base on the benchmark app result.

This demo is complete. Leave the console open and continue to the next section to run the Inference Pipeline demo.

4. Close the image viewer window to end the demo.

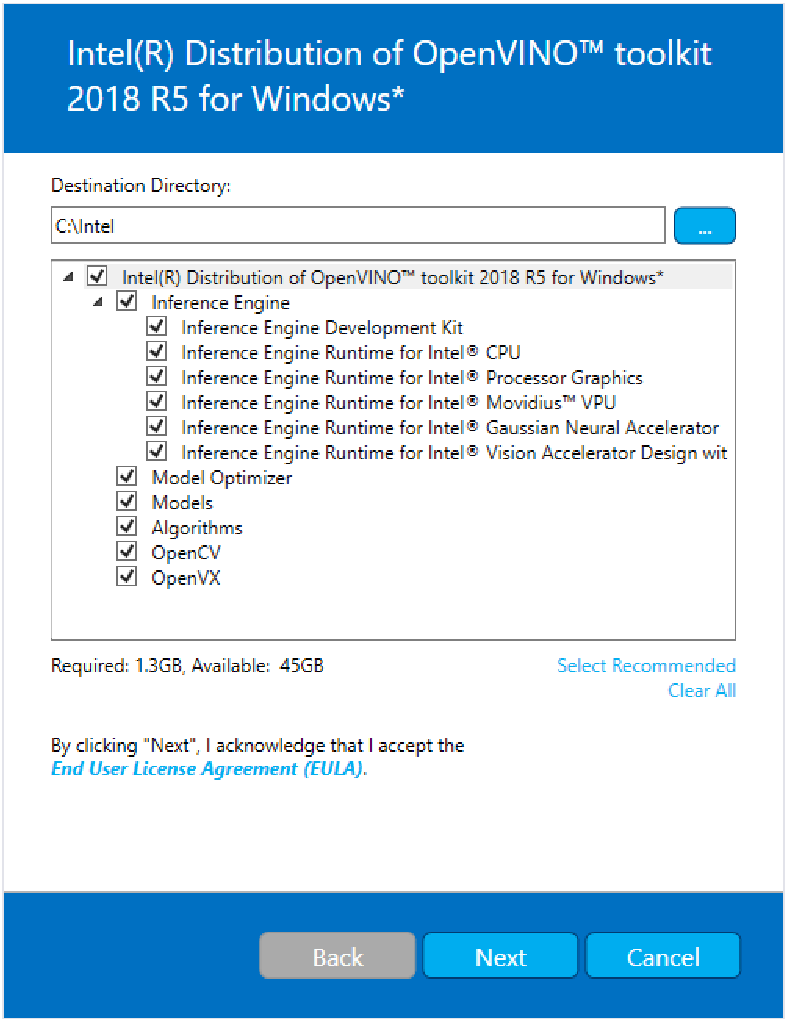
In this section, you saw a preview of the Intel® Distribution of OpenVINO™ toolkit capabilities.

You have completed all the required installation, configuration, and build steps to work with your trained models using the Mustang-V100-MX4.

5.6 Installation - Step by Step (2018 R5)

5.6.1 Install the Intel® Distribution of OpenVINO™ Toolkit Core Components

1. If you have not downloaded the Intel® Distribution of OpenVINO™ toolkit, download the latest version. By default, the file is saved to the Downloads directory as w_openvino_toolkit_p_<version>.exe.
2. Go to the **Downloads** folder.
3. Double-click **w_openvino_toolkit_p_<version>.exe**. A window opens to let you choose your installation directory and components. The default installation directory is C:\Intel. If you choose a different installation directory, the installer will create the directory for you.



Intel(R) Distribution of OpenVINO™ toolkit 2018 R5 for Windows*

Destination Directory:

C:\Intel

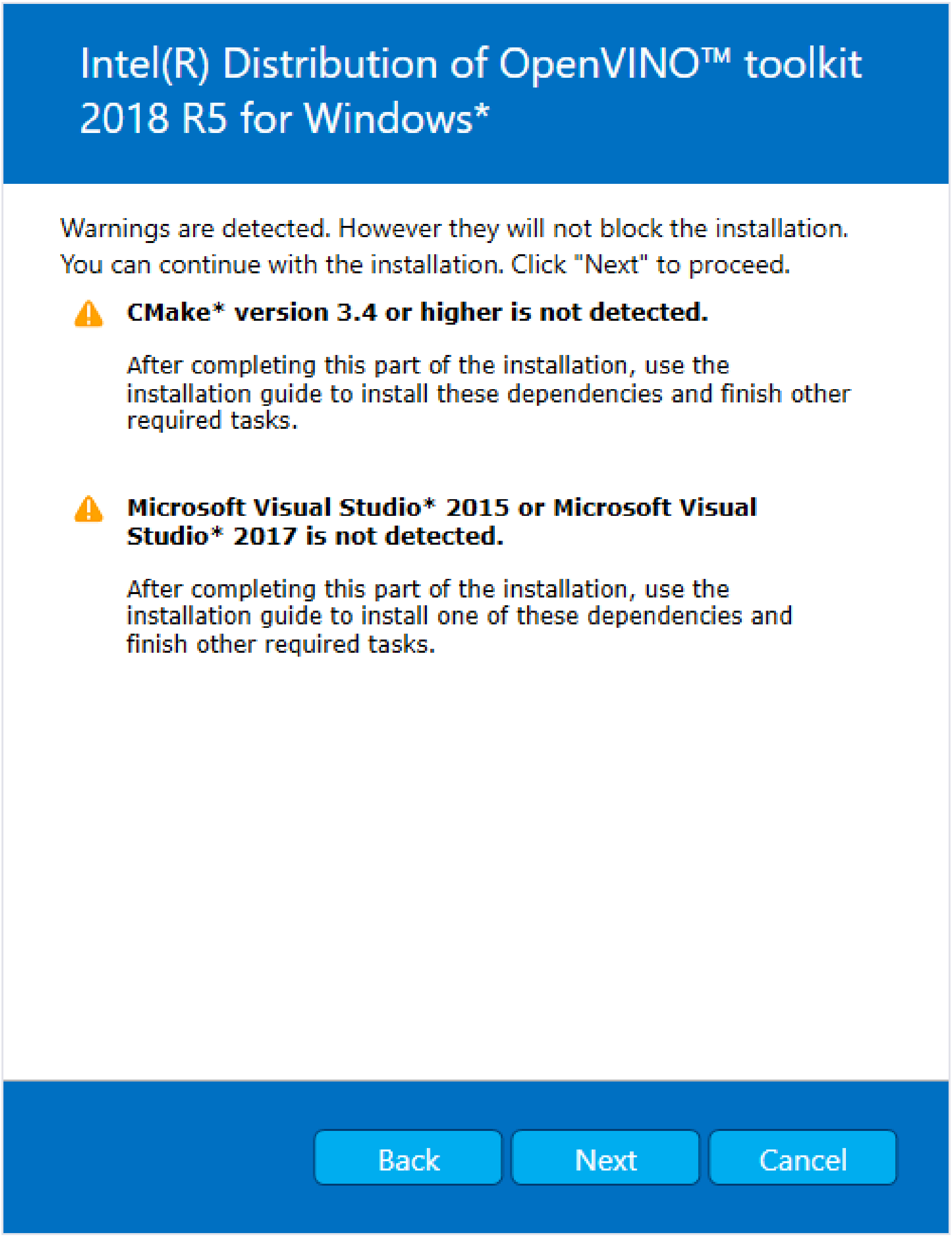
- ☑ Intel(R) Distribution of OpenVINO™ toolkit 2018 R5 for Windows*
  - ☑ Inference Engine
    - ☑ Inference Engine Development Kit
    - ☑ Inference Engine Runtime for Intel® CPU
    - ☑ Inference Engine Runtime for Intel® Processor Graphics
    - ☑ Inference Engine Runtime for Intel® Movidius™ VPU
    - ☑ Inference Engine Runtime for Intel® Gaussian Neural Accelerator
    - ☑ Inference Engine Runtime for Intel® Vision Accelerator Design wit
  - ☑ Model Optimizer
  - ☑ Models
  - ☑ Algorithms
  - ☑ OpenCV
  - ☑ OpenVX

Required: 1.3GB, Available: 45GB

By clicking "Next", I acknowledge that I accept the *End User License Agreement (EULA)*.

4. Click **Next**.
5. You are asked if you want to provide consent to gather information. Choose the option of your choice. Click **Next**.
6. If you are missing external dependencies, you will see a warning screen. Write down the dependencies you are missing. **You need to take no other action at this time**. After installing the Intel® Distribution of OpenVINO™ toolkit core components, you will be provided instructions to install the missing dependencies.

   The screen example below indicates you are missing two dependencies:



Intel(R) Distribution of OpenVINO™ toolkit 2018 R5 for Windows*

Warnings are detected. However they will not block the installation. You can continue with the installation. Click "Next" to proceed.

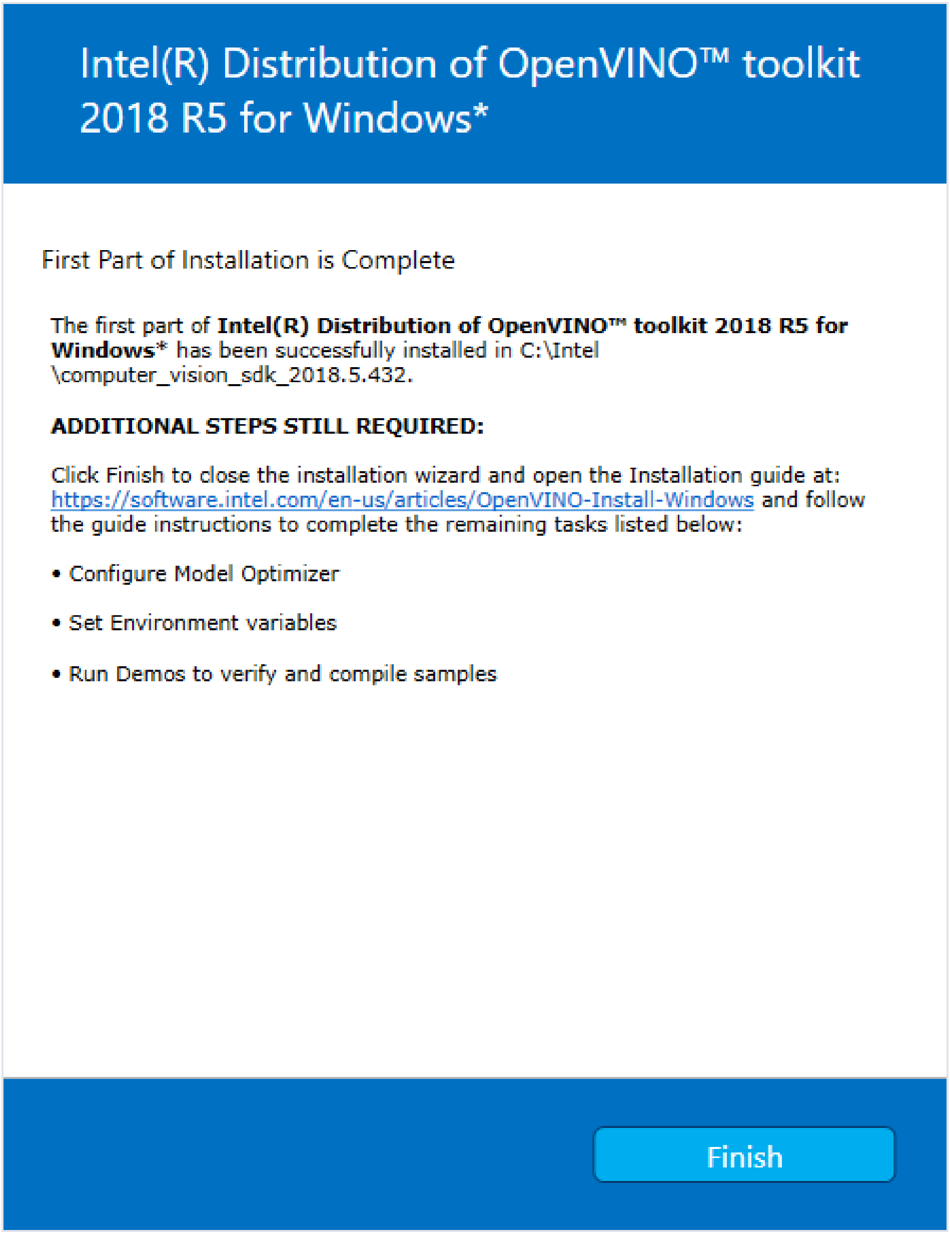
⚠ **CMake* version 3.4 or higher is not detected.**

After completing this part of the installation, use the installation guide to install these dependencies and finish other required tasks.

⚠ **Microsoft Visual Studio* 2015 or Microsoft Visual Studio* 2017 is not detected.**

After completing this part of the installation, use the installation guide to install one of these dependencies and finish other required tasks.

7. Click **Next**.
8. When the first part of installation is complete, the final screen informs you that the core components have been installed and additional steps still required:



Intel(R) Distribution of OpenVINO™ toolkit 2018 R5 for Windows*

First Part of Installation is Complete

The first part of **Intel(R) Distribution of OpenVINO™ toolkit 2018 R5 for Windows*** has been successfully installed in C:\Intel \computer_vision_sdk_2018.5.432.

ADDITIONAL STEPS STILL REQUIRED:

Click Finish to close the installation wizard and open the Installation guide at: https://software.intel.com/en-us/articles/OpenVINO-Install-Windows and follow the guide instructions to complete the remaining tasks listed below:

- Configure Model Optimizer
- Set Environment variables
- Run Demos to verify and compile samples

9. Click **Finish** to close the installation wizard. A new browser window opens to the next section of the installation guide to install the dependencies. You are in the same document. The new window opens in case you ran the installation without first opening this installation guide. If the installation did not indicate you must install dependencies, you can skip ahead to Configure the Model Optimizer.

5.6.2 Install the External Software Dependencies

If the installation process indicated if you are missing dependencies, you must install each missing dependency. Click the link for the first dependency you must install:

- Microsoft Visual Studio* with C++ 2017 or 2015 with MSBuild and the Visual Studio Build Tools. This is a two-part installation. You must install both components.
- CMake 3.4 or higher
- Python 3.6.5

If you have no missing dependencies, skip ahead to Configure the Model Optimizer.

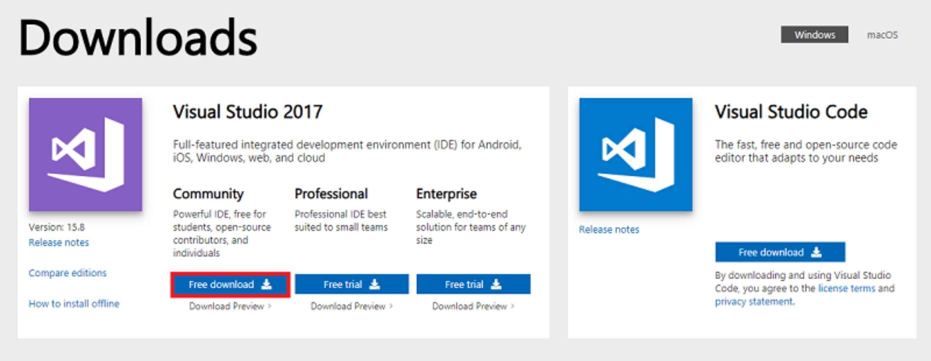
5.6.2.1 Microsoft Visual Studio* with C++ and MSBuild

Microsoft Visual Studio with Visual Studio C++ is required for building the Intel® Deep Learning Deployment Toolkit samples and demonstration applications. **You can install the free Community version of Microsoft Visual Studio**.

IMPORTANT: The Microsoft Visual Studio dependency is a two-part installation that consists of **Microsoft Visual Studio 2017 or 2015** and the **Microsoft Visual Studio Build Tools**. This guide includes steps for installing both parts of this dependency. These are separate installations. **MAKE SURE YOU INSTALL BOTH COMPONENTS**.

The steps below apply to Microsoft Visual Studio 2017. If you prefer to use Microsoft Visual Studio 2015, see Installing Microsoft Visual Studio* 2015 for Intel® Distribution of OpenVINO™ toolkit.

1. Go to the Visual Studio downloads page.
2. Click **Free Download** in the **Visual Studio 2017** box, **Community** section:



Downloads

Visual Studio 2017

Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

Community Powerful IDE, free for students, open-source contributors, and individuals

Professional Professional IDE best suited to small teams

Enterprise Scalable, end-to-end solution for teams of any size

Version: 15.8

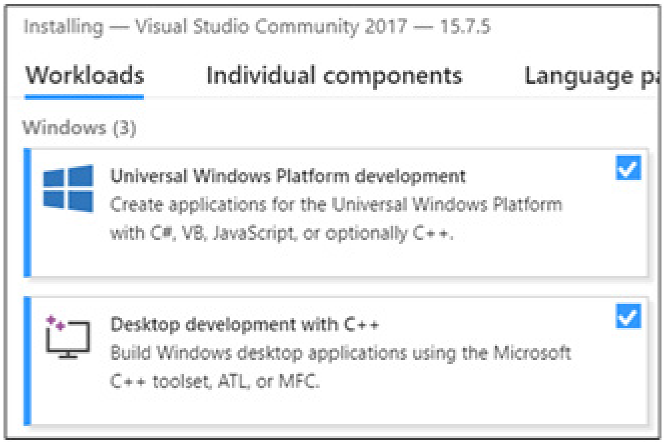
Visual Studio Code

The fast, free and open-source code editor that adapts to your needs

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

An executable file named **vs_community__313888930.1524151023.exe**, or similar, is saved in your **Downloads** folder.

3. Double-click the executable file to launch the **Visual Studio Community 2017** installer.
4. From the Workloads tab, use the check boxes to select **Universal Windows Platform development** and **Desktop development with C++**.

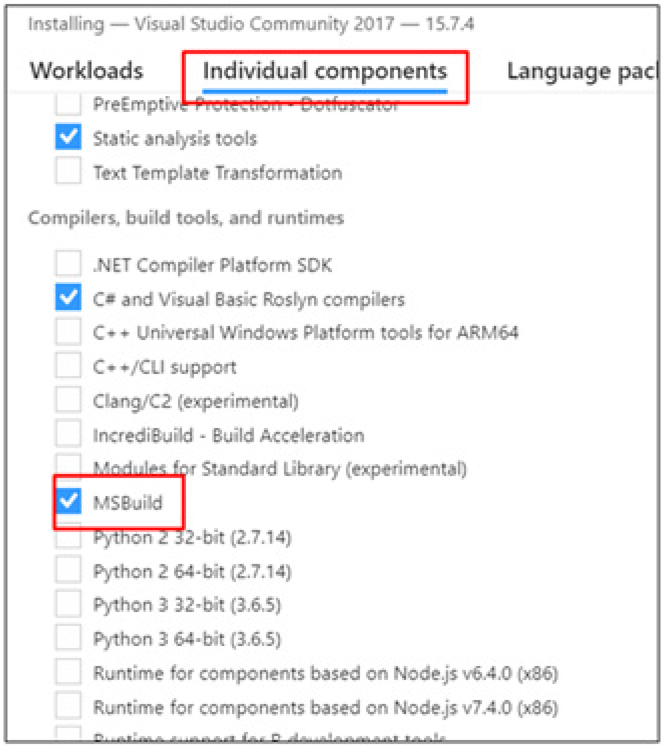


Installing — Visual Studio Community 2017 — 15.7.5

Universal Windows Platform development ☑
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

Desktop development with C++ ☑
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

5. Under the **Individual components** tab, select **MSBuild**:



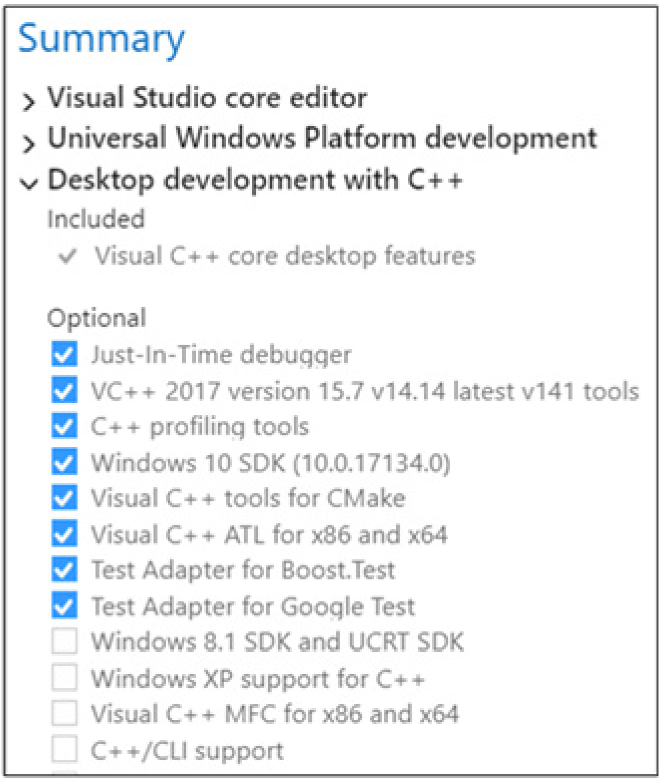
Installing — Visual Studio Community 2017 — 15.7.4

- □ PreEmptive Protection - Dotfuscator
- ☑ Static analysis tools
- □ Text Template Transformation

Compilers, build tools, and runtimes

- □ .NET Compiler Platform SDK
- ☑ C# and Visual Basic Roslyn compilers
- □ C++ Universal Windows Platform tools for ARM64
- □ C++/CLI support
- □ Clang/C2 (experimental)
- □ IncrediBuild - Build Acceleration
- □ Modules for Standard Library (experimental)
- ☑ MSBuild
- □ Python 2 32-bit (2.7.14)
- □ Python 2 64-bit (2.7.14)
- □ Python 3 32-bit (3.6.5)
- □ Python 3 64-bit (3.6.5)
- □ Runtime for components based on Node.js v6.4.0 (x86)
- □ Runtime for components based on Node.js v7.4.0 (x86)
- □ Runtime support for R development tools

The **Summary** at the right side of the screen displays your installation selections:



Summary

› Visual Studio core editor
› Universal Windows Platform development
⌄ Desktop development with C++

Included

✓ Visual C++ core desktop features

Optional

- ☑ Just-In-Time debugger
- ☑ VC++ 2017 version 15.7 v14.14 latest v141 tools
- ☑ C++ profiling tools
- ☑ Windows 10 SDK (10.0.17134.0)
- ☑ Visual C++ tools for CMake
- ☑ Visual C++ ATL for x86 and x64
- ☑ Test Adapter for Boost.Test
- ☑ Test Adapter for Google Test
- □ Windows 8.1 SDK and UCRT SDK
- □ Windows XP support for C++
- □ Visual C++ MFC for x86 and x64
- □ C++/CLI support

6. Make no other changes. Click **Next**. The installation begins, and takes around 30

minutes to complete.

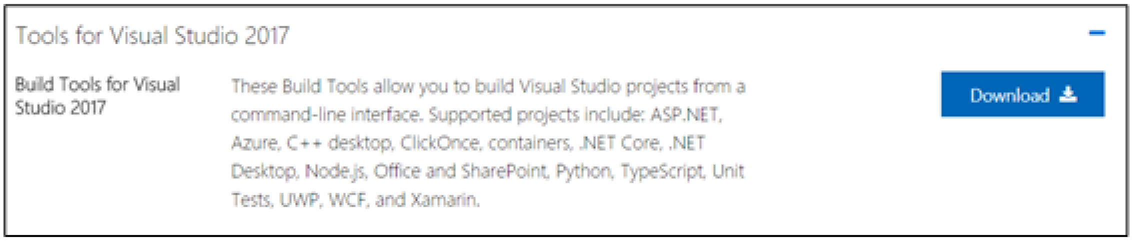
7. If you see a prompt to restart your computer after the installation completes, dismiss it.

Continue to the next section to install the Build Tools for Visual Studio 2017.

5.6.2.2 Install the Build Tools for Visual Studio 2017

The Build Tools for Visual Studio 2017 is the second part of the Microsoft Visual Studio dependency. You must complete this installation.

1. Go to the Tools for Visual Studio 2017 section of the Microsoft Visual Studio Downloads page.
2. Click the **Download** button next to **Build tools for Visual Studio 2017**:

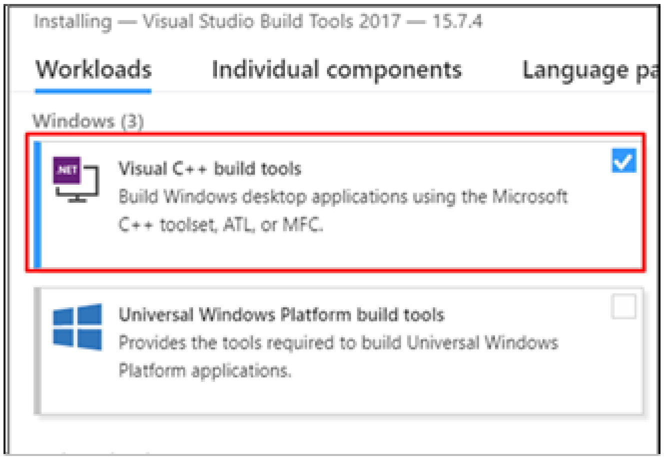


Tools for Visual Studio 2017

Build Tools for Visual Studio 2017

These Build Tools allow you to build Visual Studio projects from a command-line interface. Supported projects include: ASP.NET, Azure, C++ desktop, ClickOnce, containers, .NET Core, .NET Desktop, Node.js, Office and SharePoint, Python, TypeScript, Unit Tests, UWP, WCF, and Xamarin.

3. An executable file named **vs_buildtools.exe**, or similar, is saved in your **Downloads** folder.
4. Double-click the file to install **Build Tools for Visual Studio 2017**.
5. The installation opens to the **Workloads** tab. Select **Visual C++ build tools**:

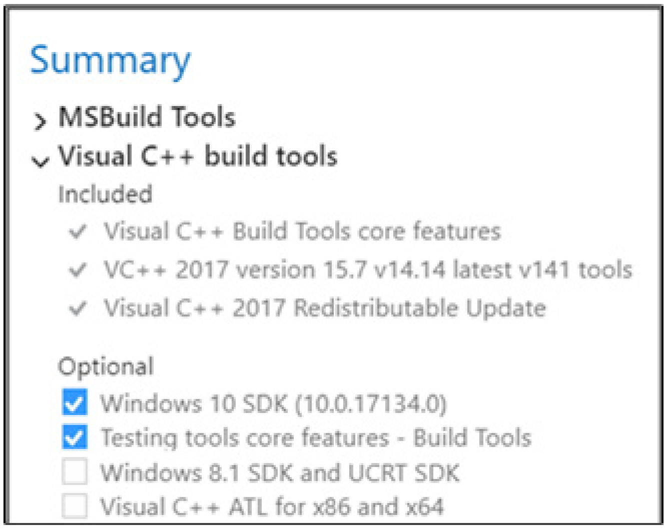


Installing — Visual Studio Build Tools 2017 — 15.7.4

☑ Visual C++ build tools
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

□ Universal Windows Platform build tools
Provides the tools required to build Universal Windows Platform applications.

The Summary on the right side shows the features you chose to install:



Summary

> MSBuild Tools

⌄ Visual C++ build tools

Included

- ✓ Visual C++ Build Tools core features
- ✓ VC++ 2017 version 15.7 v14.14 latest v141 tools
- ✓ Visual C++ 2017 Redistributable Update

Optional

- ☑ Windows 10 SDK (10.0.17134.0)
- ☑ Testing tools core features - Build Tools
- □ Windows 8.1 SDK and UCRT SDK
- □ Visual C++ ATL for x86 and x64

6. Click **Install**.
7. When the installation completes, restart your computer if prompted to do so.

You have completed the Visual Studio 2017 installation.

Install your next dependency:

- CMake 3.4 or higher
- Python 3.6.5

Or if you have installed all the dependencies, you are ready to configure the Model Optimizer.

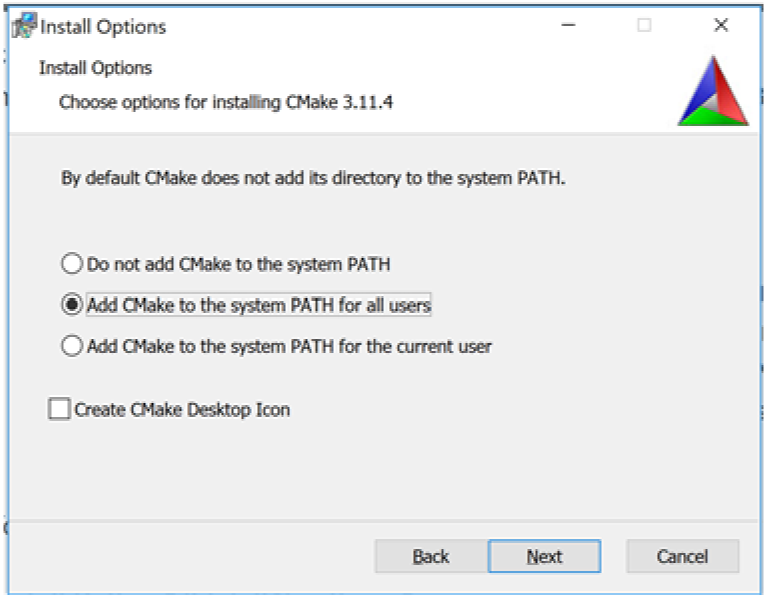
5.6.2.3 Install CMake* 3.4 or higher

These steps guide you through installing CMake 3.4 or higher, which is required to build the Intel® Distribution of OpenVINO™ toolkit samples.

1. Go to the CMake download site.
2. Under the heading **Get the Software**, click the link for **latest stable** in the first paragraph. Your screen displays **Latest Release** information.
3. Scroll down to the line **Windows win64-x64 Installer**.
4. Click the associated file name to download the installer. The file name will have the extension .msi. The file is saved to your Downloads folder.

5. Go to the Downloads folder.
6. Double-click the file to launch the installer.

   Note: If you have a previous version of CMake installed, you are prompted to uninstall it. You must uninstall the previous version before installing the new version. Follow the instructions on the screen and then launch the installer again to install the new version.
7. In the installer, select the option to **Add CMake to the system PATH for all users**:



Install Options

Install Options

Choose options for installing CMake 3.11.4

By default CMake does not add its directory to the system PATH.

○ Do not add CMake to the system PATH

● Add CMake to the system PATH for all users

○ Add CMake to the system PATH for the current user

□ Create CMake Desktop Icon

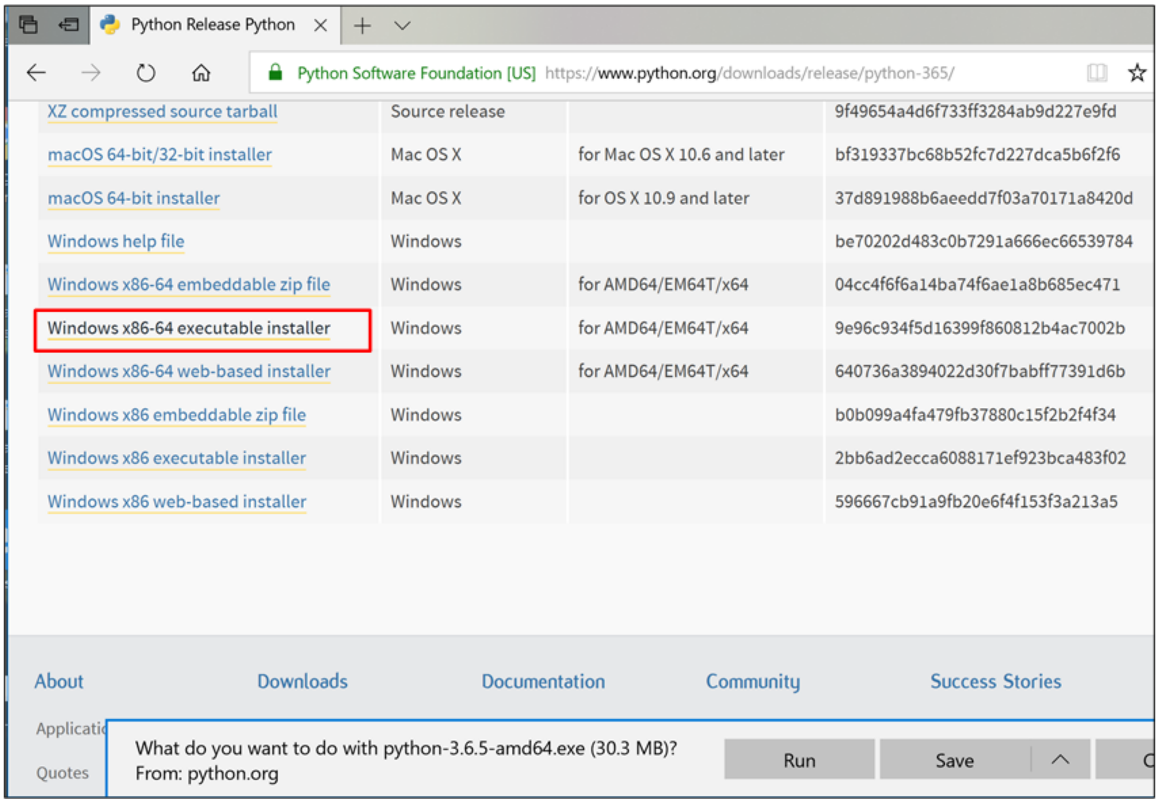
8. Click **Next**.
9. Click **Finish** when the installation completes.

You have completed the CMake installation. Next, install Python 3.6.5 if the Intel® Distribution of OpenVINO™ toolkit installation indicated you are missing the software.

5.6.2.4 Install Python* 3.6.5

Python 3.6.5 with pip is required to run the Model Optimizer. Use these steps to install the correct version of the Python software.

1. Go to the Python 3.6.5 download page and click **Windows x86-64 executable installer** to download the executable file. The file is saved as python-3.6.5-amd64.exe in your Downloads folder.

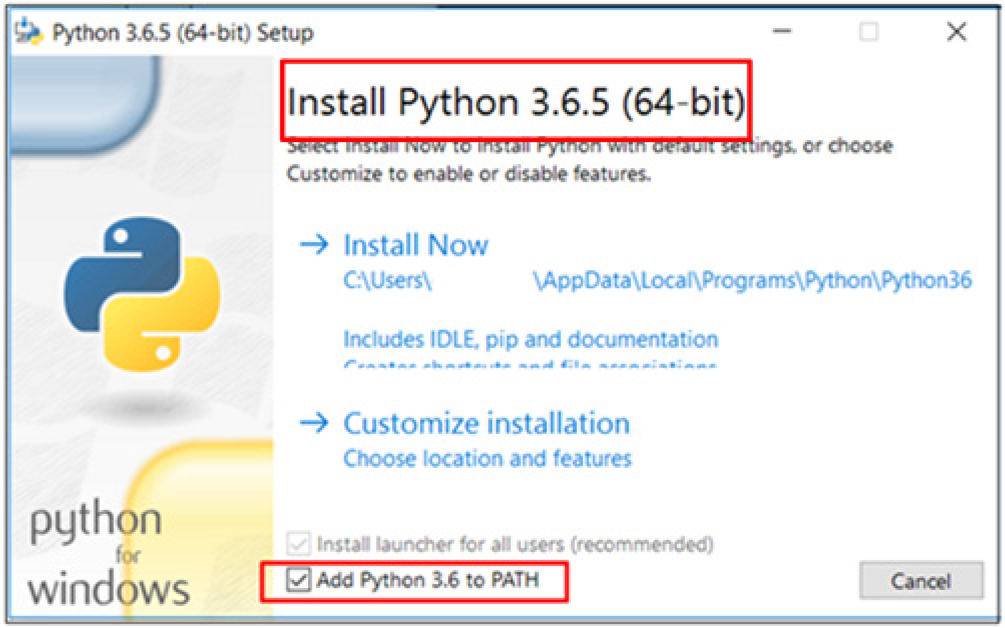


Python Release Python
Python Software Foundation [US] https://www.python.org/downloads/release/python-365/

XZ compressed source tarball	Source release		9f49654a4d6f733ff3284ab9d227e9fd
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	bf319337bc68b52fc7d227dca5b6f2f6
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	37d891988b6aeedd7f03a70171a8420d
Windows help file	Windows		be70202d483c0b7291a666ec66539784
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	04cc4f6f6a14ba74f6ae1a8b685ec471
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	9e96c934f5d16399f860812b4ac7002b
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	640736a3894022d30f7babff77391d6b
Windows x86 embeddable zip file	Windows		b0b099a4fa479fb37880c15f2b2f4f34
Windows x86 executable installer	Windows		2bb6ad2ecca6088171ef923bca483f02
Windows x86 web-based installer	Windows		596667cb91a9fb20e6f4f153f3a213a5

What do you want to do with python-3.6.5-amd64.exe (30.3 MB)?
From: python.org

2. Double-click the file to launch the installation.
3. Make sure the top of the screen shows **Python 3.6.5 (64-bit)**.
4. **IMPORTANT:** At the bottom of the install screen, select **Add Python 3.6 to PATH**.



Install Python 3.6.5 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\ \AppData\Local\Programs\Python\Python36

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

5. Click **Install Now** near the top of the install screen and let the installation complete.
6. When the installation finishes, click **Close**.

You have completed the Python installation and are ready to set environment variables. Continue to the next section.

5.6.3 Set the Environment Variables

You must update several environment variables before you can compile and run OpenVINO™ applications. Open the Command Prompt and run the following batch file to temporarily set your environment variables:

```
C:\Intel\computer_vision_sdk\bin\setupvars.bat
```

(Optional): OpenVINO toolkit environment variables are removed when you close the Command Prompt window. As an option, you can permanently set the environment variables manually.

The environment variables are set. Continue to the next section to configure the Model Optimizer.

5.6.4 Configure the Model Optimizer

Important: These steps are required. You must configure the Model Optimizer for at least one framework. The Model Optimizer will fail if you do not complete the steps in this section.

If you see an error indicating Python is not installed when you know you installed it, your computer might not be able to find the program. For instructions to add Python to your system environment variables, see Update Your Windows Environment Variables.

The Model Optimizer is a key component of the Intel® Distribution of OpenVINO™ toolkit. You cannot do inference on your trained model without running the model through the Model Optimizer. When you run a pre-trained model through the Model Optimizer, your output is an Intermediate Representation (IR) of the network. The IR is a pair of files that describe the whole model:

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

The Inference Engine reads, loads, and infers the IR files, using a common API across the CPU, GPU, or VPU hardware.

The Model Optimizer is a Python*-based command line tool (mo.py), which is located in **C:\Intel\computer_vision_sdk_<version>\deployment_tools\model_optimizer**, where <version> is the version of the Intel® Distribution of OpenVINO™ toolkit that you installed. Use this tool on models trained with popular deep learning frameworks such as Caffe, TensorFlow, MXNet, and ONNX to convert them to an optimized IR format that the Inference Engine can use.

This section explains how to use scripts to configure the Model Optimizer either for all of the supported frameworks at the same time or for individual frameworks. If you want to manually configure the Model Optimizer instead of using scripts, see the Using Manual Configuration Process section in the Model Optimizer Developer Guide.

For more information about the Model Optimizer, see the Model Optimizer Developer Guide.

5.6.4.1 Model Optimizer Configuration Steps

You can configure the Model Optimizer either for all supported frameworks at once or for one framework at a time. Choose the option that best suits your needs. If you see error messages, make sure you installed all dependencies.

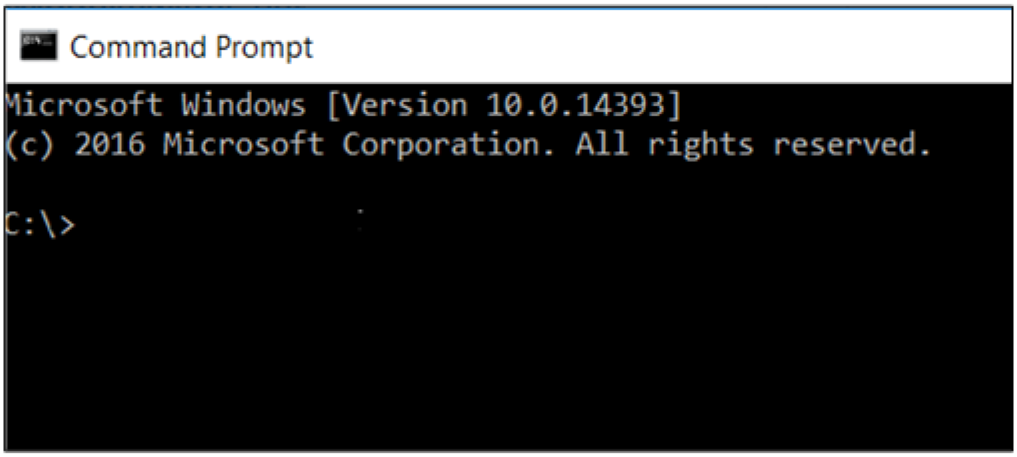
Note: These steps use a command prompt to make sure you see error messages.
In the steps below:
- Replace **<version>** with the version number of your Intel® Distribution of OpenVINO™ toolkit
- If you did not install Intel® Distribution of OpenVINO™ toolkit to the default installation directory, replace \Intel\ with the directory where you installed the software.

Option 1: Configure the Model Optimizer for all supported frameworks at the same time:

1. Open a command prompt. To do so, type the following in your **Search Windows** box and then press **Enter**:

   ```
   Cmd
   ```

2. Type commands in the opened window:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\>
```

3. Go to the Model Optimizer prerequisites directory. Remember to replace <version> with the version of the Intel® Distribution of OpenVINO™ toolkit that you installed:

   ```
   cd
   C:\Intel\computer_vision_sdk_<version>\deployment_tools\model_
   optimizer\install_prerequisites
   ```

4. Run the following batch file to configure Model Optimizer for Caffe*, TensorFlow*, MXNet*, Kaldi*, and ONNX*:

   ```
   install_prerequisites.bat
   ```

Option 2: Configure the Model Optimizer for each framework separately:

1. Go to the Model Optimizer prerequisites directory:

   ```
   cd
   C:\Intel\computer_vision_sdk_<version>\deployment_tools\model_
   optimizer\install_prerequisites
   ```

2. Run the batch file for the framework you will use with the Model Optimizer. You can use more than one:

- For Caffe:

  ```
  install_prerequisites_caffe.bat
  ```

- For TensorFlow:

  ```
  install_prerequisites_tf.bat
  ```

- For MXNet:

  ```
  install_prerequisites_mxnet.bat
  ```

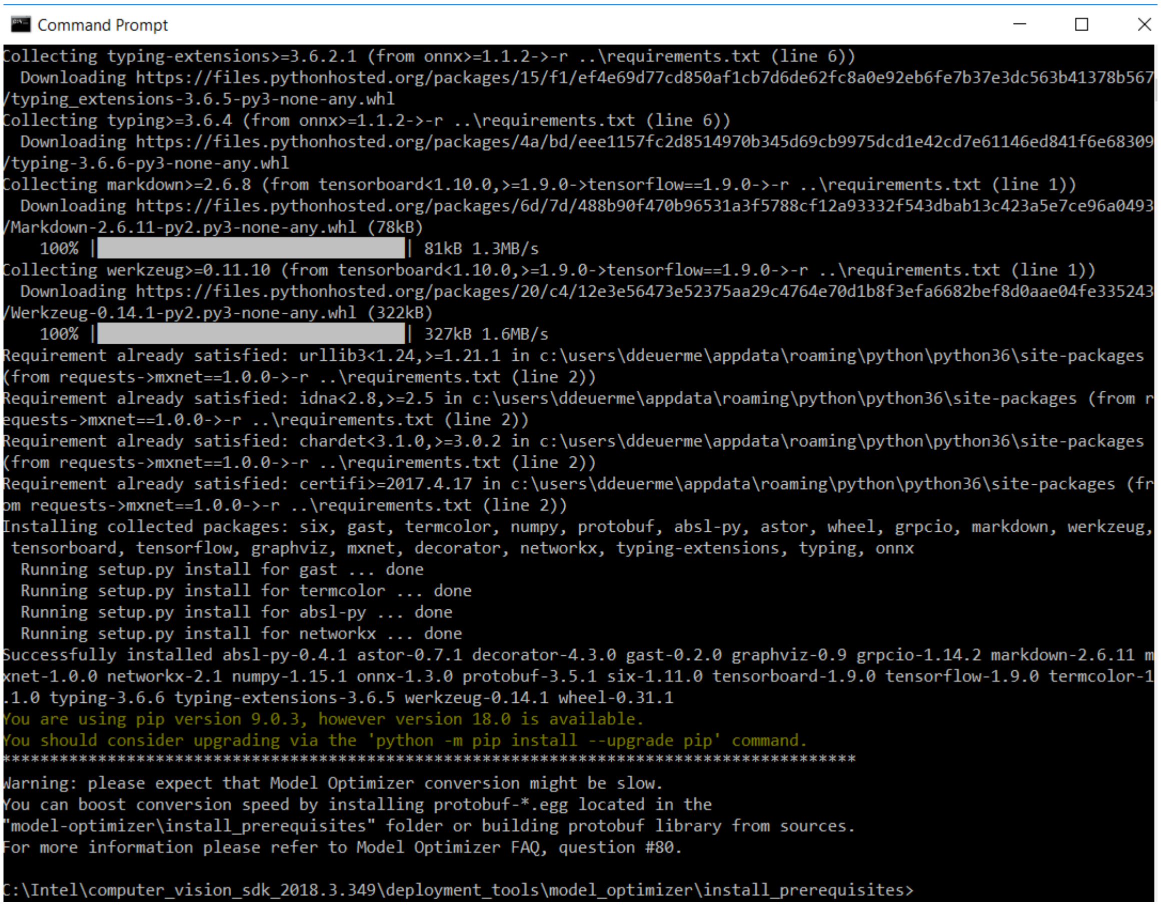
- For ONNX:

  ```
  install_prerequisites_onnx.bat
  ```

- For Kaldi:

  ```
  install_prerequisites_kaldi.bat
  ```

The Model Optimizer is configured for one or more frameworks. Success is indicated by a screen similar to this:



```
Collecting typing-extensions>=3.6.2.1 (from onnx>=1.1.2->-r ..\requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/15/f1/ef4e69d77cd850af1cb7d6de62fc8a0e92eb6fe7b37e3dc563b41378b567
/typing_extensions-3.6.5-py3-none-any.whl
Collecting typing>=3.6.4 (from onnx>=1.1.2->-r ..\requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/4a/bd/eee1157fc2d8514970b345d69cb9975dcd1e42cd7e61146ed841f6e68309
/typing-3.6.6-py3-none-any.whl
Collecting markdown>=2.6.8 (from tensorboard<1.10.0,>=1.9.0->tensorflow==1.9.0->-r ..\requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/6d/7d/488b90f470b96531a3f5788cf12a93332f543dbab13c423a5e7ce96a0493
/Markdown-2.6.11-py2.py3-none-any.whl (78kB)
    100% |████████████████████████████████| 81kB 1.3MB/s
Collecting werkzeug>=0.11.10 (from tensorboard<1.10.0,>=1.9.0->tensorflow==1.9.0->-r ..\requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243
/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
    100% |████████████████████████████████| 327kB 1.6MB/s
Requirement already satisfied: urllib3<1.24,>=1.21.1 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages
(from requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: idna<2.8,>=2.5 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages (from r
equests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages
(from requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ddeuerme\appdata\roaming\python\python36\site-packages (fr
om requests->mxnet==1.0.0->-r ..\requirements.txt (line 2))
Installing collected packages: six, gast, termcolor, numpy, protobuf, absl-py, astor, wheel, grpcio, markdown, werkzeug,
 tensorboard, tensorflow, graphviz, mxnet, decorator, networkx, typing-extensions, typing, onnx
  Running setup.py install for gast ... done
  Running setup.py install for termcolor ... done
  Running setup.py install for absl-py ... done
  Running setup.py install for networkx ... done
Successfully installed absl-py-0.4.1 astor-0.7.1 decorator-4.3.0 gast-0.2.0 graphviz-0.9 grpcio-1.14.2 markdown-2.6.11 m
xnet-1.0.0 networkx-2.1 numpy-1.15.1 onnx-1.3.0 protobuf-3.5.1 six-1.11.0 tensorboard-1.9.0 tensorflow-1.9.0 termcolor-1
.1.0 typing-3.6.6 typing-extensions-3.6.5 werkzeug-0.14.1 wheel-0.31.1
You are using pip version 9.0.3, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
*****************************************************************************************
Warning: please expect that Model Optimizer conversion might be slow.
You can boost conversion speed by installing protobuf-*.egg located in the
"model-optimizer\install_prerequisites" folder or building protobuf library from sources.
For more information please refer to Model Optimizer FAQ, question #80.

C:\Intel\computer_vision_sdk_2018.3.349\deployment_tools\model_optimizer\install_prerequisites>
```

You are ready to use two short demos to see the results of running the Intel Distribution of OpenVINO toolkit and to verify your installation was successful. The demo scripts are required since they perform additional configuration steps. Continue to the next section.

5.7 Installation Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs Mustang-V100-MX4

To perform inference on Mustang-V100-MX4, the following additional installation steps are required:

1. Install the Movidius™ VSC driver:

   1) Go to the `<INSTALL_DIR>\deployment_tools\inference-engine\external\` directory, where `<INSTALL_DIR>` is the directory in which the OpenVINO™ toolkit is installed.

   2) Right click on the `Movidius_VSC_Device.inf` file and choose **Install** from the pop up menu.

2. Install the SMBUS driver:

   1) Go to the `<INSTALL_DIR>\deployment_tools\inference-engine\external\hddl\SMBusDriver` directory, where `<INSTALL_DIR>` is the directory in which the OpenVINO™ toolkit is installed.

   2) Right click on the `hddlsmbus.inf` file and choose Install from the pop up menu.

3. Download and install Visual C++ Redistributable for Visual Studio 2015

You are done installing your device driver and are ready to use Mustang-V100-MX4.

5.8 Use the Demo Scripts to Verify Your Installation

Important: This section is required. In addition to confirming your installation was successful, demo scripts perform other steps, such as setting up your computer to use the Model Optimizer samples.

Note: To run the demo applications on Intel® Processor Graphics, Intel® Movidius™ Neural Compute Stick or Intel® Neural Compute Stick 2, make sure you completed the Additional Installation Steps first.

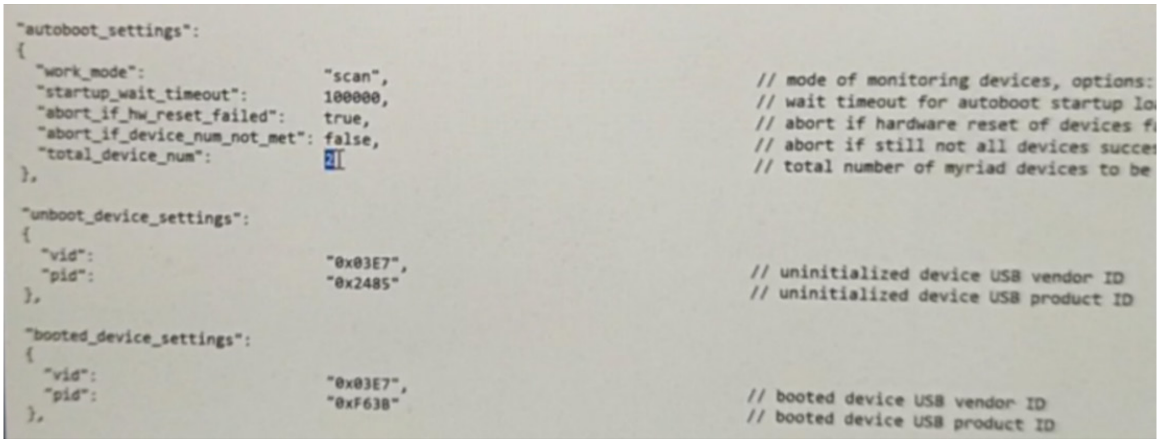
To learn more about the demo applications, see `README.txt` in `C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\`.

For detailed description of the pre-trained object detection and object recognition models, go to `C:\Intel\computer_vision_sdk_<version>\deployment_tools\intel_models\` and open `index.html`.

Notes:

- The paths in this section assume you used the default installation directory. If you used a directory other than C:\Intel, update the directory with the location where you installed the software.
- If you are migrating from the Intel® Computer Vision SDK 2017 R3 Beta version to the Intel® Distribution of OpenVINO™ toolkit, read this information about porting your applications.

3. Go to modify device number of VPU from 8 to 4, in <install_DIR>\openvino_<version>\`deployment_tools\inference_engine\external\hddl\config\hddl_autoboot.config`



```
"autoboot_settings":
{
  "work_mode":                     "scan",                // mode of monitoring devices, options:
  "startup_wait_timeout":          100000,                // wait timeout for autoboot startup lo
  "abort_if_hw_reset_failed":      true,                  // abort if hardware reset of devices fa
  "abort_if_device_num_not_met":   false,                 // abort if still not all devices succes
  "total_device_num":              2                      // total number of myriad devices to be
},

"unboot_device_settings":
{
  "vid":                           "0x03E7",              // uninitialized device USB vendor ID
  "pid":                           "0x2485"               // uninitialized device USB product ID
},

"booted_device_settings":
{
  "vid":                           "0x03E7",              // booted device USB vendor ID
  "pid":                           "0xF63B"               // booted device USB product ID
},
```

1. Open a command prompt window.

2. Go to the Inference Engine demo directory:
   ```
   cd
   C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\
   ```

3. Run the demos by following the instructions in the next two sections.

5.8.1 Run the Image Classification Demo

This demo serves two purposes:

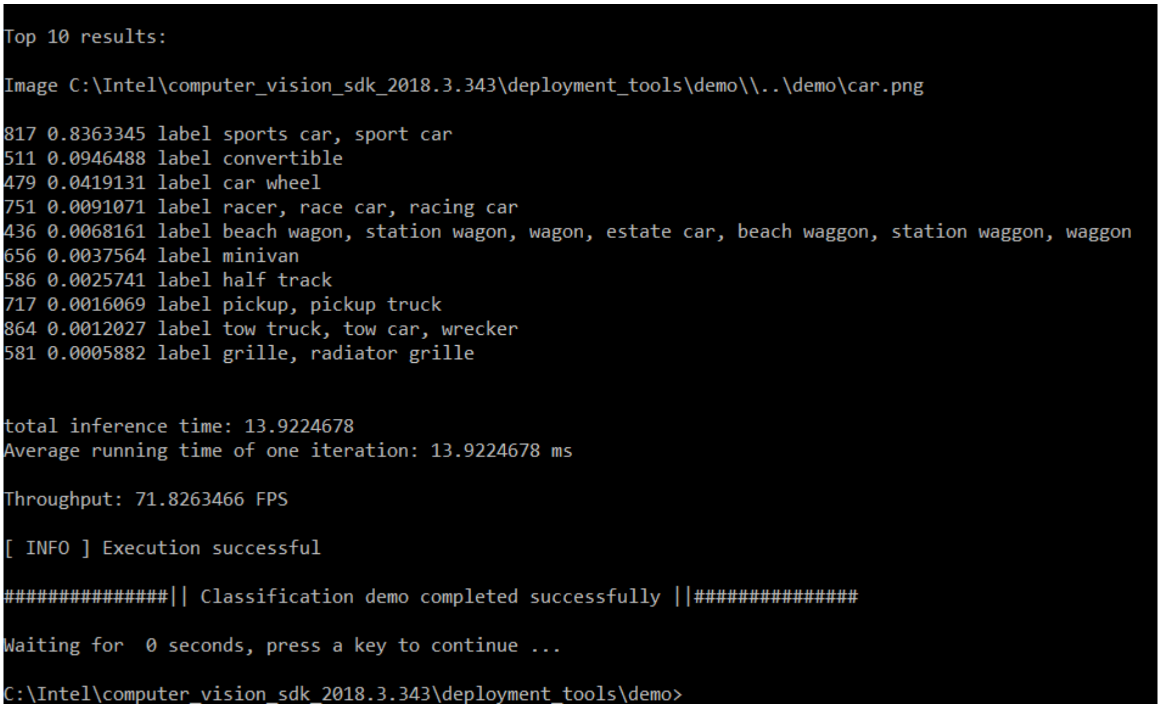
- It creates a directory named build_`<version>` in `C:\Intel\computer_vision_sdk_<version>\deployment_tools\inference_engine\samples`.
- It uses the Model Optimizer to convert a SqueezeNet model to `.bin` and `.xml` Intermediate Representation (IR) files that are used by the Inference Engine.

For a brief description of the Intermediate Representation .bin and .xml files, see Configuring the Model Optimizer.

For more information about the Inference Engine, see the Inference Engine Developer Guide.

1. Run the Image Classification demo:
   ```
       demo_squeezenet_download_convert_run.bat
   ```

2. This demo uses the car.png image located in the `C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo` directory. When the demo completes, the label and confidence for the top-10 categories are displayed on your screen:



```
Top 10 results:

Image C:\Intel\computer_vision_sdk_2018.3.343\deployment_tools\demo\\..\demo\car.png

817 0.8363345 label sports car, sport car
511 0.0946488 label convertible
479 0.0419131 label car wheel
751 0.0091071 label racer, race car, racing car
436 0.0068161 label beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
656 0.0037564 label minivan
586 0.0025741 label half track
717 0.0016069 label pickup, pickup truck
864 0.0012027 label tow truck, tow car, wrecker
581 0.0005882 label grille, radiator grille


total inference time: 13.9224678
Average running time of one iteration: 13.9224678 ms

Throughput: 71.8263466 FPS

[ INFO ] Execution successful

###############|| Classification demo completed successfully ||###############

Waiting for  0 seconds, press a key to continue ...

C:\Intel\computer_vision_sdk_2018.3.343\deployment_tools\demo>
```

Note: performance here is only for reference, it should base on the benchmark app result.

This demo is complete. Leave the console open and continue to the next section to run the Inference Pipeline demo.

5.8.2 Run the Inference Pipeline Demo

1. While still in the `C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\` directory, run the Inference Pipeline demo:

    ```
    demo_security_barrier_camera.bat
    ```

2. This demo uses the car.png image located in `C:\Intel\computer_vision_sdk_<version>\deployment_tools\demo\` to show an inference pipeline. This demo uses three pre-trained models. The demo uses vehicle recognition in which vehicle attributes build on each other to narrow in on a specific attribute. The demo works as follows:

    1) An object is identified as a vehicle.

    2) This identification is used as input to the next model, which identifies specific

vehicle attributes, including the license plate.

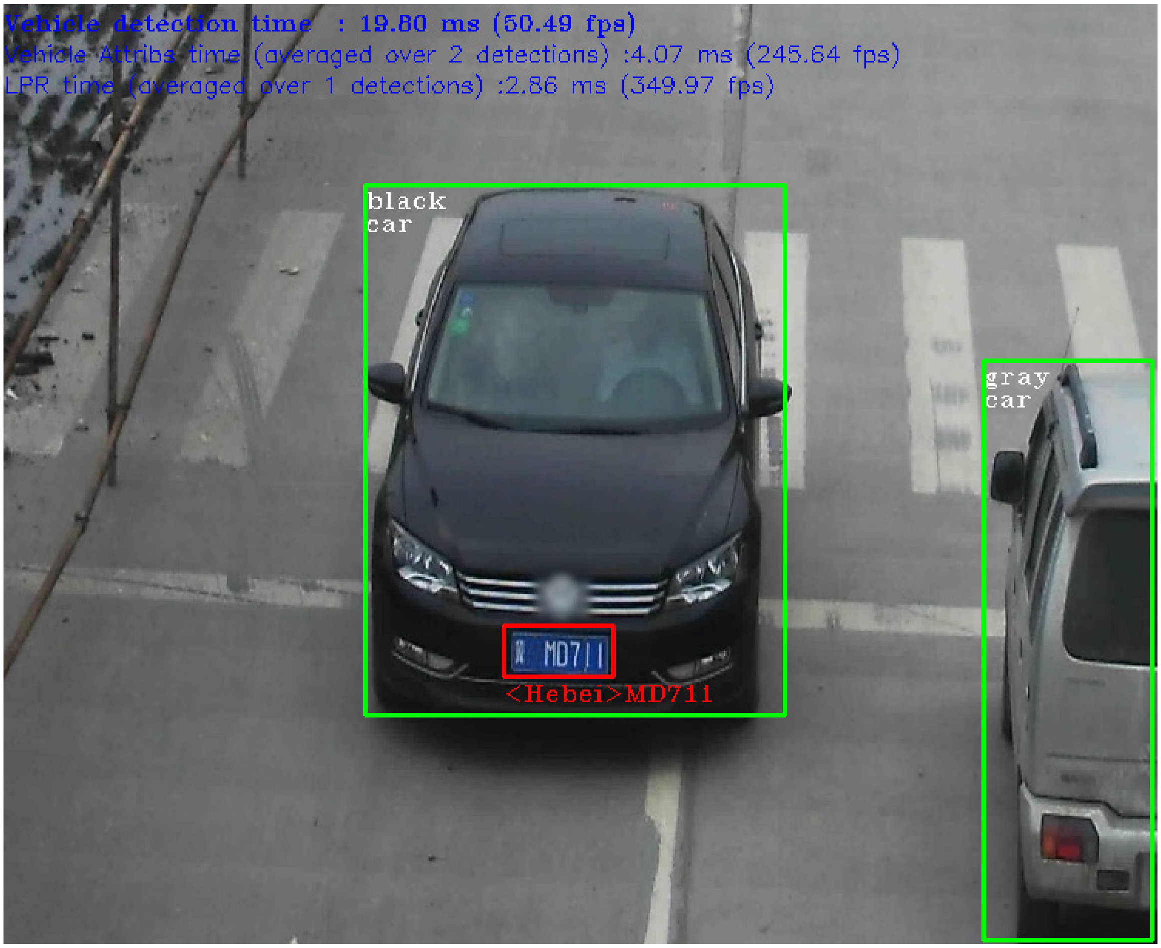
3) The attributes identified as the license plate are used as input to the third model, which recognizes specific characters in the license plate.

For more information, see the Security Camera Sample.

3. When the demo completes, you have two windows open:

o A console window that displays information about the tasks performed by the demo.

o An image viewer window that displays a picture similar to the following:



```
Vehicle detection time  : 19.80 ms (50.49 fps)
Vehicle Attribs time (averaged over 2 detections) :4.07 ms (245.64 fps)
LPR time (averaged over 1 detections) :2.86 ms (349.97 fps)
```

Note: performance here is only for reference, it should base on the benchmark app result.

4. Close the image viewer window to end the demo.

In this section, you saw a preview of the Intel® Distribution of OpenVINO™ toolkit capabilities.

You have completed all the required installation, configuration, and build steps to work with your trained models using the Mustang-V100-MX4.

Appendix

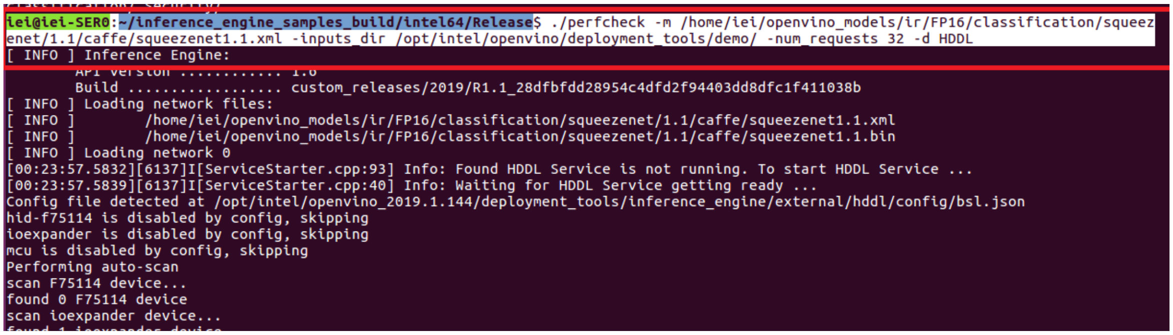
A

Performance Test

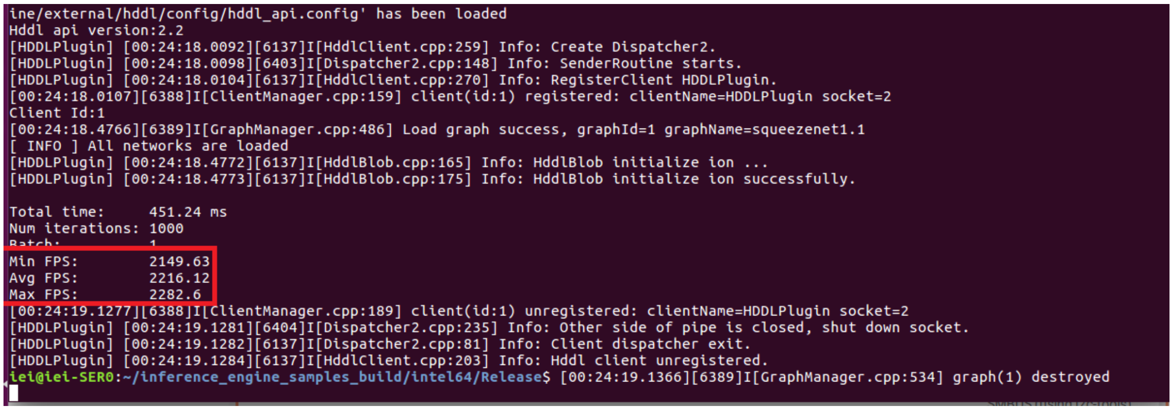
./perfcheck -m **<model_Dir>** -inputs_dir**<image_Dir>** -num_requests **8** -d **HDDL**

Note: suggest -num_requests # = 4xVPU qty. (Mustang-V100-MX4=> 8)

Ex. squeezenet1.1.xml



```
iei@iei-SER0:~/inference_engine_samples_build/intel64/Release$ ./perfcheck -m /home/iei/openvino_models/ir/FP16/classification/squeez
enet/1.1/caffe/squeezenet1.1.xml -inputs_dir /opt/intel/openvino/deployment_tools/demo/ -num_requests 32 -d HDDL
[ INFO ] Inference Engine:
        API version ............ 1.6
        Build .................. custom_releases/2019/R1.1_28dfbfdd28954c4dfd2f94403dd8dfc1f411038b
[ INFO ] Loading network files:
[ INFO ]        /home/iei/openvino_models/ir/FP16/classification/squeezenet/1.1/caffe/squeezenet1.1.xml
[ INFO ]        /home/iei/openvino_models/ir/FP16/classification/squeezenet/1.1/caffe/squeezenet1.1.bin
[ INFO ] Loading network 0
[00:23:57.5832][6137]I[ServiceStarter.cpp:93] Info: Found HDDL Service is not running. To start HDDL Service ...
[00:23:57.5839][6137]I[ServiceStarter.cpp:40] Info: Waiting for HDDL Service getting ready ...
Config file detected at /opt/intel/openvino_2019.1.144/deployment_tools/inference_engine/external/hddl/config/bsl.json
hid-f75114 is disabled by config, skipping
ioexpander is disabled by config, skipping
mcu is disabled by config, skipping
Performing auto-scan
scan F75114 device...
found 0 F75114 device
scan ioexpander device...
found 1 ioexpander device
```



```
ine/external/hddl/config/hddl_api.config' has been loaded
Hddl api version:2.2
[HDDLPlugin] [00:24:18.0092][6137]I[HddlClient.cpp:259] Info: Create Dispatcher2.
[HDDLPlugin] [00:24:18.0098][6403]I[Dispatcher2.cpp:148] Info: SenderRoutine starts.
[HDDLPlugin] [00:24:18.0104][6137]I[HddlClient.cpp:270] Info: RegisterClient HDDLPlugin.
[00:24:18.0107][6388]I[ClientManager.cpp:159] client(id:1) registered: clientName=HDDLPlugin socket=2
Client Id:1
[00:24:18.4766][6389]I[GraphManager.cpp:486] Load graph success, graphId=1 graphName=squeezenet1.1
[ INFO ] All networks are loaded
[HDDLPlugin] [00:24:18.4772][6137]I[HddlBlob.cpp:165] Info: HddlBlob initialize ion ...
[HDDLPlugin] [00:24:18.4773][6137]I[HddlBlob.cpp:175] Info: HddlBlob initialize ion successfully.

Total time:     451.24 ms
Num iterations: 1000
Batch:          1
Min FPS:        2149.63
Avg FPS:        2216.12
Max FPS:        2282.6
[00:24:19.1277][6388]I[ClientManager.cpp:189] client(id:1) unregistered: clientName=HDDLPlugin socket=2
[HDDLPlugin] [00:24:19.1281][6404]I[Dispatcher2.cpp:235] Info: Other side of pipe is closed, shut down socket.
[HDDLPlugin] [00:24:19.1282][6137]I[Dispatcher2.cpp:81] Info: Client dispatcher exit.
[HDDLPlugin] [00:24:19.1284][6137]I[HddlClient.cpp:203] Info: Hddl client unregistered.
iei@iei-SER0:~/inference_engine_samples_build/intel64/Release$ [00:24:19.1366][6389]I[GraphManager.cpp:534] graph(1) destroyed
```

Appendix

B

Regulatory Compliance

DECLARATION OF CONFORMITY

This equipment has been tested and found to comply with specifications for CE marking. If the user modifies and/or installs other devices in the equipment, the CE conformity declaration may no longer apply.

FCC WARNING

This equipment complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

- This device may not cause harmful interference, and
- This device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Appendix C

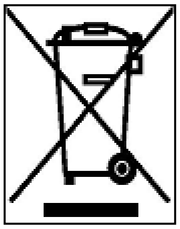
Product Disposal

CAUTION:

Risk of explosion if battery is replaced by an incorrect type. Only certified engineers should replace the on-board battery.

Dispose of used batteries according to instructions and local regulations.

- Outside the European Union – If you wish to dispose of used electrical and electronic products outside the European Union, please contact your local authority so as to comply with the correct disposal method.
- Within the European Union – The device that produces less waste and is easier to recycle is classified as electronic device in terms of the European Directive 2012/19/EU (WEEE), and must not be disposed of as domestic garbage.



EU-wide legislation, as implemented in each Member State, requires that waste electrical and electronic products carrying the mark (left) must be disposed of separately from normal household waste. This includes monitors and electrical accessories, such as signal cables or power cords. When you need to dispose of your device, please follow the guidance of your local authority, or ask the shop where you purchased the product. The mark on electrical and electronic products only applies to the current European Union Member States.

Please follow the national guidelines for electrical and electronic product disposal.

Appendix

D

Hazardous Materials Disclosure

The details provided in this appendix are to ensure that the product is compliant with the Peoples Republic of China (China) RoHS standards. The table below acknowledges the presences of small quantities of certain materials in the product, and is applicable to China RoHS only.

A label will be placed on each product to indicate the estimated "Environmentally Friendly Use Period" (EFUP). This is an estimate of the number of years that these substances would "not leak out or undergo abrupt change." This product may contain replaceable sub-assemblies/components which have a shorter EFUP such as batteries and lamps. These components will be separately marked.

Please refer to the following table.

Part Name	Toxic or Hazardous Substances and Elements					
	Lead (Pb)	Mercury (Hg)	Cadmium (Cd)	Hexavalent Chromium (CR(VI))	Polybrominated Biphenyls (PBB)	Polybrominated Diphenyl Ethers (PBDE)
Housing	O	O	O	O	O	O
Display	O	O	O	O	O	O
Printed Circuit Board	O	O	O	O	O	O
Metal Fasteners	O	O	O	O	O	O
Cable Assembly	O	O	O	O	O	O
Fan Assembly	O	O	O	O	O	O
Power Supply Assemblies	O	O	O	O	O	O
Battery	O	O	O	O	O	O

O: This toxic or hazardous substance is contained in all of the homogeneous materials for the part is below the limit requirement in SJ/T11363-2006 (now replaced by GB/T 26572-2011).

X: This toxic or hazardous substance is contained in at least one of the homogeneous materials for this part is above the limit requirement in SJ/T11363-2006 (now replaced by GB/T 26572-2011).

此附件旨在确保本产品符合中国 RoHS 标准。以下表格标示此产品中某有毒物质的含量符合中国 RoHS 标准规定的限量要求。

本产品上会附有”环境友好使用期限”的标签，此期限是估算这些物质”不会有泄漏或突变”的年限。本产品可能包含有较短的环境友好使用期限的可替换元件，像是电池或灯管，这些元件将会单独标示出来。

部件名称	有毒有害物质或元素					
	铅 **(Pb)**	汞 **(Hg)**	镉 **(Cd)**	六价铬 **(CR(VI))**	多溴联苯 **(PBB)**	多溴二苯醚 **(PBDE)**
壳体	O	O	O	O	O	O
显示	O	O	O	O	O	O
印刷电路板	O	O	O	O	O	O
金属螺帽	O	O	O	O	O	O
电缆组装	O	O	O	O	O	O
风扇组装	O	O	O	O	O	O
电力供应组装	O	O	O	O	O	O
电池	O	O	O	O	O	O

O: 表示该有毒有害物质在该部件所有物质材料中的含量均在 SJ/T 11363-2006 (现由 GB/T 26572-2011 取代) 标准规定的限量要求以下。

X: 表示该有毒有害物质至少在该部件的某一均质材料中的含量超出 SJ/T 11363-2006 (现由 GB/T 26572-2011 取代) 标准规定的限量要求。