**OpenCV-Python**

OpenCV-Python Tutorials

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html>

한글 <https://opencv-python.readthedocs.io/en/latest/doc/02.videoStart/videoStart.html>

<https://pypi.org/project/opencv-python/>

**Linux OpenCV-Python 환경 설정**

Notebook 환경 : Linux20.04, VSCode, Anaconda, JupyterLab

NUC 환경 : Linux18.04

1) 가상환경 만들기

- Anaconda

conda create -n opencv python=3.8

conda activate opencv

- Venv

python3 -m venv opencv/opencv\_env

cd opencv

source openvino\_env/bin/activate

2) opencv 설치

conda install -c conda-forge opencv

또는 (Qobject::moveToThread: error 조치 위해 아래 내용으로 설치)

pip install opencv-python==4.1.2.30

pip install opencv-contrib-python==4.1.2.30

3) 확인

import cv2

print(cv2.\_\_version\_\_)

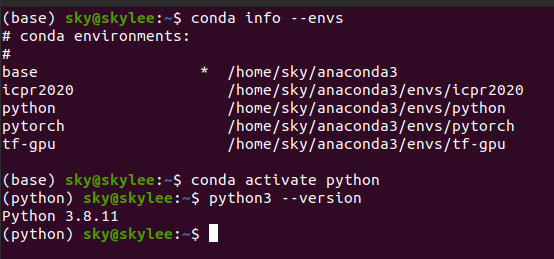
[Note] opencv 삭제

**1. Linux 상에 OpenCV-Python 설치** (Section 2.1.3)

<https://docs.opencv.org/master/d2/de6/tutorial_py_setup_in_ubuntu.html>

* Ubuntu18.04 (또는 20.04)에 설치되어 있는 python 버전 확인

$python3 –version



**1. Installing OpenCV-Python from Pre-built Binaries**

1) python3-opencv

|  |
| --- |
| $ sudo apt-get install python3-opencv |

2) pip install

pip install opencv-python==4.1.2.30

pip install opencv-contrib-python==4.1.2.30

3) conda install

conda install -c conda-forge opencv

Open Python IDLE (or IPython) and type following codes in Python terminal.

|  |
| --- |
| import cv2 as cv  print(cv.\_\_version\_\_) |

※ [Note] Error

* opencv-python 를 쓰는 과정에서 다음과 같은 오류

QObject::moveToThread: Current thread (0x561d8e139860) is not the object's thread (0x561d8e25a580). Cannot move to target thread (0x561d8e139860)

* 해결 : downgrade

pip uninstall opencv-python==4.1.2.30

pip install opencv-python==4.1.2.30

**2. Building OpenCV from source**

**1) 기본작업 수행**

* 기존 설치된 opencv가 있으면 삭제

|  |
| --- |
| sudo apt-get remove libopencv\*  sudo apt-get autoremove |

* 패키지 리스트 update하고, upgrade 진행

|  |
| --- |
| $sudo apt-get update  $sudo apt-get upgrade |

**2) 컴파일 전 필요한 패키지 설치 (빌드 도구, Image, Video, GUI 등)**

**OpenCV와 종속 라이브러리 설치**

* 홈 디렉토리로 이동하여 설치된 패키지 라이브러리를 업데이트와 업그레이드한 다음 (업그레이드 하는데 시간이 많이 걸릴 수 있다) 개발자 도구(build-essential, cmake, unzip, pkg-config)를 설치한다.
* **build-essential**은 빌드에 필요한 각종 기본 개발도구 및 라이브러리로 c/c++ 컴파일러와 관련한 라이브러리, make 같은 도구들이 포함된다.
* **cmake**는 MakeFile을 생성하는 도구 중의 하나이며,
* **pkg-config**은 컴파일할 목적으로 설치된 라이브러리를 조회하기 위해 설치된다.

|  |
| --- |
| $sudo apt-get install build-essential cmake unzip pkg-config *git dkms* |

- sudo apt-get upgrade에서 시간이 많이 소요

* 다음은 JPEG, PNG, TIFF 등과 같은 이미지 처리 관련 라이브러리(libjpeg, libpng, libtiff), 비디오 인코딩 디코딩와 처리 관련 라이브러리(libavcodec, libavformat, libswscale, libv4l)와 MPEG-4 및 H.264 비디오 코덱 라이브러리(libxvidcore, libx264)를 설치한다.

|  |
| --- |
| $sudo apt-get install libjpeg-dev libpng-dev libtiff-dev  $sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev  $sudo apt-get install libxvidcore-dev libx264-dev |

* 다음은 GUI(Graphical User Interface) 처리를 위한 GTK(GIMP Toolkit) 라이브러리(libgtk)도 설치한다. 그리고 다양한 OpenCV 함수를 최적화하기 위한 라이브러리(libatlas-base)도 함께 설치하기를 권장한다.

|  |
| --- |
| $sudo apt-get install libgtk-3-dev  $sudo apt-get install libatlas-base-dev gfortran |

* Python3 헤더와 라이브러리를 설치한다.

|  |
| --- |
| $sudo apt-get install python3-dev |

* 기타 참고 (https://cafepurple.tistory.com/40)

|  |
| --- |
| $ sudo apt-get install build-essential cmake git pkg-config dkms  $ sudo apt-get install freeglut3 freeglut3-dev libxi-dev libxmu-dev  $ sudo apt-get install ubuntu-restricted-extras  $ sudo apt-get install libjpeg-dev libtiff5 libjasper-dev libpng-dev  $ sudo apt-get install libavcodec-dev ffmpeg libavformat-dev libswscale-dev libv4l-dev v4l-utils  $ sudo apt-get install libxvidcore-dev libx264-dev libxine2-dev  $ sudo apt-get install libgtk-3-dev  $ sudo apt-get install libatlas-base-dev gfortran  $ sudo apt-get install mesa-utils libgl1-mesa-dri libgtkgl2.0-dev libgtkglext1-dev |

※ libjasper-dev 패키지를 설치할 때 오류가 발생할 경우, 아래와 같이 입력한다.

sudo add-apt-repository "deb http://security.ubuntu.com/ubuntu xenial-security main"

**3) opencv와 opencv\_contrib 다운로드(공식)**

wget(Web Get) 명령을 사용하여 OpenCV(version 4.5.0)를 다운로드한 다음, SIFT, SURF 등의 알고리즘이 들어간 opencv\_contrib 모듈도 함께 다운로드한다.

[Memo]

* OpenCV github-Release에 들어간다. OpenCV github(github.com/opencv/opencv/releases)에 접속하면, 현재 release 되고 있는 버전 정보를 확인할 수 있다. OpenCV는 3.x 버전과 4.x 버전이 각각 릴리즈 되고 있고, 3.x 버전이 4.x 버전보다 좀 더 최적화가 잘 되어있고 안정적이라는 장점이 있다. 4.x 버전은 C++11로 쓰여졌고, opencv\_contrib를 따라가려면 추천한다고 함. 연구용으로 쓸 것이면 4.x 가 나을 것 같고, 상업용으로 쓴다면 3.x가 낫지 않을까 개인적으로 생각해봅니다.
* opencv\_contrib는 opencv의 extra module이다. 아직 안정화가 덜 되었거나, 최신 기술이어서 좀 더 검증이 필요하다던가, 특허 이슈가 있다거나, non-free 등의 이유로 opencv\_contrib에 속해 있다. 버전업이 되면서 opencv\_contrib에 있다가 opencv쪽으로 넘어오는 API도 있다. opencv\_contrib는 상용 제품에 사용하기 전에 반드시 확인해야 한다.
* 임의의 디렉터리를 만들고, 여기에 opencv와 opencv-contrib를 다운로드 받고, unzip에 의하여 푼다.

|  |
| --- |
| $cd ~  $mkdir opencv  $cd opencv  $wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.0.zip  $wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/4.5.0.zip  $unzip opencv.zip  $unzip opencv\_contrib.zip |

* 다음으로 python3 패키지 관리 시스템을 설치한 다음, pip 패키지 관리 시스템을 이용하여 NumPy 패키지를 설치한다.

|  |
| --- |
| $sudo apt-get install python3-pip  $pip3 install numpy |

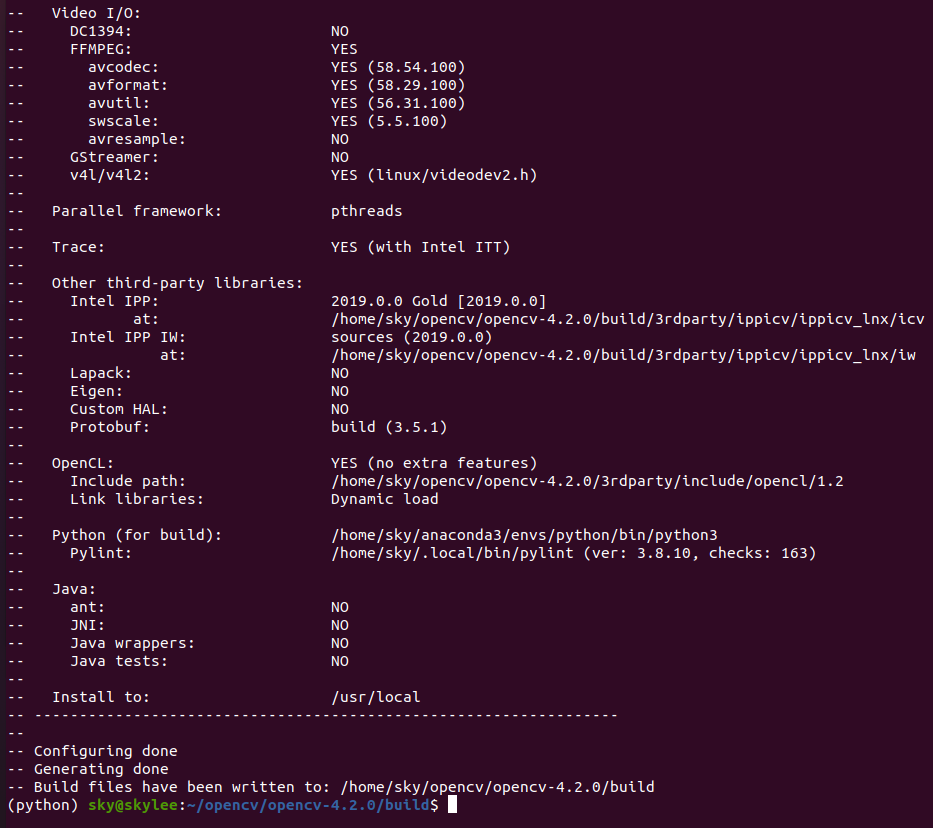
**4) OpenCV의 설정과 소스코드 컴파일**

이제 다음과 같이 OpenCV 빌드를 수행한다.

* 먼저 opencv-4.5.0 폴더로 이동해서 컴파일을 위한 임시폴더 build를 만든다. 이제 build 폴더로 이동하여, 컴파일을 위한 Configuring을 진행한다.

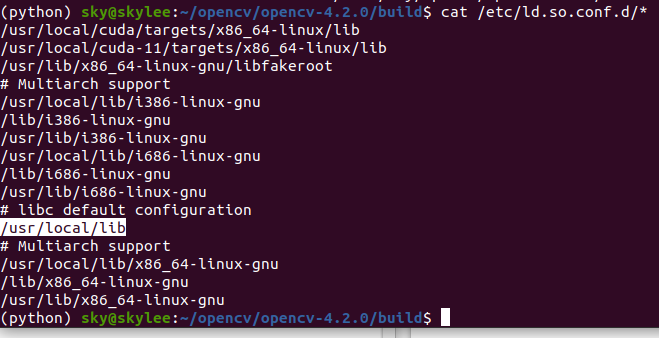
[교재]

|  |
| --- |
| $cd ~/opencv-4.5.0/  $mkdir build  $cd build  $cmake -D CMAKE\_BUILD\_TYPE=RELEASE \  -D CMAKE\_INSTALL\_PREFIX=/usr/local \  -D BUILD\_EXAMPLES=ON \  -D INSTALL\_C\_EXAMPLES=OFF \  -D INSTALL\_PYTHON\_EXAMPLES=ON \  -D OPENCV\_EXTRA\_MODULES\_PATH=../../opencv\_contrib-4.5.0/modules \  -D PYTHON\_EXECUTABLE=~/.virtualenvs/cv/bin/python \  -D BUILD\_NEW\_PYTHON\_SUPPORT=ON \  -D OPENCV\_GENERATE\_PKGCONFIG=ON \  ..\ |



* /etc/ld.so.conf.d/ 디렉토리에 /usr/local/lib 를 포함하는 설정파일이 있는지 확인

|  |
| --- |
| $cat /etc/ld.so.conf.d/\* |



[Note]

* cmake를 쓸 때 인터넷을 뒤져가며 옵션을 설정하는데, 매번 할 때마다 'Configuring incomplete, errors occurred!' 메시지를 안 만난 적이 없던 것 같음. 아래 사용한 옵션을 참고하셔서 진행하시기 바람.

|  |
| --- |
| $cd ~/opencv-4.5.0/  $mkdir build  $cd build  $cmake -D CMAKE\_BUILD\_TYPE=RELEASE \  -D CMAKE\_INSTALL\_PREFIX=/usr/local \  -D OPENCV\_GENERATE\_PKGCONFIG=ON \  -D OPENCV\_ENABLE\_NONFREE=ON \  -D OPENCV\_EXTRA\_MODULES\_PATH=~/opencv/opencv\_contrib-4.5.0/modules \  -D INSTALL\_C\_EXAMPLES=ON \  -D INSTALL\_PYTHON\_EXAMPLES=ON \  -D BUILD\_EXAMPLES=ON \  -D BUILD\_DOCS=OFF \  -D BUILD\_SHARED\_LIBS=ON \  -D BUILD\_opencv\_python2=OFF \  -D BUILD\_opencv\_python3=ON \  -D BUILD\_NEW\_PYTHON\_SUPPORT=ON \  -D WITH\_CUDA=ON \  -D WITH\_CUBLAS=ON \  -D WITH\_CUDNN=ON \  -D CUDA\_FAST\_MATH=1 \  -D CUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda-11.2 \  -D OPENCV\_DNN\_CUDA=ON \  -D CUDA\_ARCH\_BIN=7.5 \  -D CUDA\_ARCH\_PTX=7.5 \  -D CUDNN\_VERSION=8.0 \  -D CUDNN\_INCLUDE\_DIR=/usr/local/cuda-11.2/include \  -D CUDNN\_LIBRARY=/usr/local/cuda-11.2/lib64/libcudnn.so.8.0.4 \  -D WITH\_VTK=ON \  -D WITH\_OPENCL=ON \  -D OPENCV\_SKIP\_PYTHON\_LOADER=ON \  -D PYTHON\_EXECUTABLE=~/anaconda3/bin/python3 \  -D PYTHON3\_INCLUDE\_DIR=~/anaconda3/include/python3.8m \  -D PYTHON3\_NUMPY\_INCLUDE\_DIRS=~/anaconda3/lib/python3.8/site-packages/numpy/core/include \  -D PYTHON3\_PACKAGES\_PATH=~/anaconda3/lib/python3.8/site-packages \  -D PYTHON3\_LIBRARY=~/anaconda3/lib/libpython3.8.so \  -D PYTHON\_LIBRARIES=~/anaconda3/lib/python3.8 .. |

※ 마지막의 .. 은 컴파일할 원본 소스의 경로이므로 꼭 입력해야 된다.

※ 옵션의 세부 경로는 각 컴퓨터 환경에 맞추어 진행해야 한다. 제 PC는 anaconda가 설치되어 있었기 때문에 python3을 따로 설치하지 않고 경로만 잡아주었다.

※ CUDNN 관련 문제가 발생했는데, CUDNN\_VERSION=8.0을 넣어서 해결했습니다.

* ls /usr/include 와 같이 직접 해당 디렉토리에 가서 설치된 버전을 확인해도 된다.

|  |
| --- |
| cat /proc/cpuinfo | grep processor | wc -l |

cpu info 정보를 확인하거나 위 명령어를 이용하여 cpu 코어수를 확인하세요.

노트북의 경우 cpu 코어수 = 12

* OpenCV 빌드가 완료되면, 다음의 OpenCV 컴파일에 대한 명령을 수행한다. (j뒤의 숫자는 cpu 코어 수를 의미한다. 전체 코어 수의 20%만 사용해도 성능은 괜찮다고 함)

|  |
| --- |
| $make -j4  $sudo make install |

* Build 시간이 오래 걸림
* make의 -j 옵션은 한번에 수행할수 있는 명령(Job)을 지정하는 parallel 옵션이다. 예를 들어, CPU코어가 4개이라면, 한번에 수행할수 있는 명령을 -j 4으로 지정하여 프로세스가 4개가 생성되어 병렬으로 수행합니다. 명령수는 프로세스수로 정의된다.
* 컴파일에 성공하면 마지막으로 ldconfig 실행하여 공유 라이브러리를 다시 로드한 다음, OpenCV를 설치하고, 버전을 확인한다.

|  |
| --- |
| $sudo ldconfig  $pkg-config --modversion opencv4 |

* ldconfig 는 /etc/ld.so.conf 파일에서 명령 줄에 지정된 디렉토리에서 발견 된 가장 최근의 공유 라이브러리에 필요한 링크와 캐시 (런타임 링커에서 사용하기 위해 ld.so )를 만듭니다. ( / usr / lib 및 / lib ). ldconfig 는 링크가 업데이트되어야하는 버전을 결정할 때 발견 된 라이브러리의 헤더 및 파일 이름을 확인합니다. ldconfig 는 라이브러리를 검색 할 때 심볼 링크를 무시합니다.

---------------------------------------------------------------------------------------------------------------------------------------

$sudo ldconfig 오류

sudo ldconfig

/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_cnn\_train.so.8은(는) 심볼릭 링크가 아닙니다

/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_cnn\_infer.so.8은(는) 심볼릭 링크가 아닙니다

/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_ops\_infer.so.8은(는) 심볼릭 링크가 아닙니다

/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_ops\_train.so.8은(는) 심볼릭 링크가 아닙니다

/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_adv\_infer.so.8은(는) 심볼릭 링크가 아닙니다

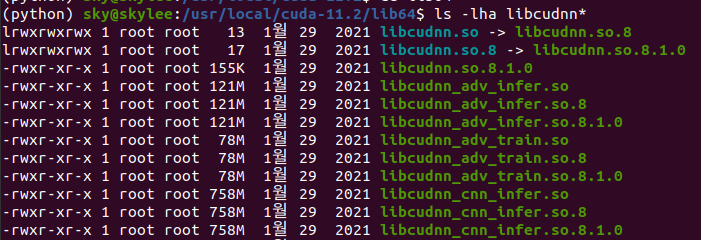
/sbin/ldconfig.real: /usr/local/cuda/targets/x86\_64-linux/lib/libcudnn\_adv\_train.so.8은(는) 심볼릭 링크가 아닙니다

자신의 CUDA 버전에 맞는 폴더로 이동한 다음, 아래와 같이 검색을 해주면 된다.

|  |
| --- |
| $ cd /usr/local/cuda-11.2/lib64  $ ls -lha libcudnn\* |

심볼릭 링크 ->

제일 상단에 libcudnn.so, so.8, so.8.1.0 파일들에 링크가 없는걸 볼 수 있음. 우리가 복사하면서 사라진 거임. 이걸 다시 만들어줘야함.

..

아래와 같이 명령어를 입력하고 다시 검색한다.

|  |
| --- |
| $ sudo ln -sf libcudnn.so.8.1.0 libcudnn.so.8  $ sudo ln -sf libcudnn.so.8 libcudnn.so  $ ls -al libcudnn.so\* |

심볼릭 링크가 형성됨

lrwxrwxrwx 1 root root 13 10월 19 01:34 libcudnn.so -> libcudnn.so.8

lrwxrwxrwx 1 root root 17 10월 19 01:33 libcudnn.so.8 -> libcudnn.so.8.1.0

-rwxr-xr-x 1 root root 155K 1월 29 2021 libcudnn.so.8.1.0

**마지막 설정**

설치된 Python은 다음 폴더에 위치할 것이다

|  |
| --- |
| $ls /usr/local/lib/python3.8/site−packages/cv2.cpython−36m−x86\_64−linux−gnu.so |

먼저, 라이브러리 이름을 변경해둔다.

|  |
| --- |
| $cd /usr/local/lib/python3.6/site−packages/  $sudo mv cv2.cpython−36m−x86\_64−linux−gnu.so cv2.so |

그리고 OpenCV가 제대로 설치되어는지를 확인하기 위하여 다음과 같이 테스트해 본다.

|  |
| --- |
| $python3  >>import cv2  >>cv2.\_\_version\_\_  >> exit() |

OpenCV가 제대로 설치되어 있으면, 설치한 OpenCV 버전을 반환한다(결과 3.4.3). 마지막으로 Python 인터프리터에서 로그 아웃한다.

===============================================================================

\* 참고 : https://wiserloner.tistory.com/1398

opencv4.2.0, python 3.8 기준

버전에 따라 설정을 약간씩 바꾸면 됨

- 우분투에 opencv를 설치하는 방법입니다.

opencv 4.2.0을 기준으로 하며,

python은 2.7과 3.8을 사용합니다.

전체적으로 [webnautes.tistory.com/1186](https://webnautes.tistory.com/1186) 이분의 자료를 참고했으며,

버전이 달라서 고생했으므로 개인적으로 적어둡니다.

버전에 따라 설정을 약간씩 바꾸시면 됩니다.

- 기존 설치 opencv 제거

1. pkg-config --modversion opencv

으로 기존 버전이 설치되어있는지를 확인 후 설치가 안 되어있으면 설치를 진행하면 됨

2. 제거

sudo apt-get purge libopencv\* python-opencv

sudo apt-get autoremove

sudo find /usr/local/ -name "\*opencv\*" -exec rm -i {} \;

- 설치

1. 준비

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install build-essential cmake

sudo apt-get install pkg-config

sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev

sudo apt-get install libv4l-dev v4l-utils

sudo apt-get install libstreamer1.0-dev libstreamer-plugins-base1.0-dev

sudo apt-get install libgtk2.0-dev

sudo apt-get install libgtk-3-dev

sudo apt-get install libqt4-dev

sudo apt-get install libqt5-dev

sudo apt-get install mesa-utils libgl1-mesa-dri libgtkgl2.0-dev libgtkglext1-dev

sudo apt-get install python2.7-dev python3-dev python-numpy python3-numpy

여기서 주의해야할 것이 각 버전들입니다.

특히나 파이썬 dev의 경우에는 설치되며 내부 파이썬 버전이 달라질수 있는데 이를 나중에 파악합니다.

2. opncv 빌드

먼저 임의의 디렉터리를 만듭니다.

mkdir opencv

cd opencv

다음으로 opencv와 opencv-contrib를 다운받습니다.

wget -O [opencv.zip](http://opencv.zip/) <https://github.com/opencv/opencv/archive/4.2.0.zip>

unzip opencv.zip

wget -O [opencv\_contrib.zip](http://opencv_contrib.zip/) <https://github.com/opencv/opencv_contrib/archive/4.2.0.zip>

unzip opencv\_contrib.zip

다음으로 opencv 디렉터리 안으로 이동하여 빌드를 진행할 폴더를 만듭니다.

cd opencv-4.2.0

mkdir build

cd build

다음으로는 cmd 창에 cmake 설정을 합니다.

cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D WITH\_TBB=OFF \

-D WITH\_IPP=OFF \

-D WITH\_1394=OFF \

-D BUILD\_WITH\_DEBUG\_INFO=OFF \

-D BUILD\_DOCS=OFF \

-D INSTALL\_C\_EXAMPLES=ON \

-D INSTALL\_PYTHON\_EXAMPLES=ON \

-D BUILD\_EXAMPLES=OFF \

-D BUILD\_TESTS=OFF \

-D BUILD\_PERF\_TESTS=OFF \

-D WITH\_QT=OFF \

-D WITH\_GTK=ON \

-D WITH\_OPENGL=ON \

-D OPENCV\_EXTRA\_MODULES\_PATH=../../opencv\_contrib-4.2.0/modules \

-D WITH\_V4L=ON \

-D WITH\_FFMPEG=ON \

-D WITH\_XINE=ON \

-D BUILD\_NEW\_PYTHON\_SUPPORT=ON \

-D OPENCV\_GENERATE\_PKGCONFIG=ON \

-D OPENCV\_ENABLE\_NONFREE=ON \

-D PYTHON2\_INCLUDE\_DIR=/usr/include/python2.7 \

-D PYTHON2\_NUMPY\_INCLUDE\_DIRS=/usr/lib/python2.7/dist-packages/numpy/core/include/ \

-D PYTHON2\_PACKAGES\_PATH=/usr/lib/python2.7/dist-packages \

-D PYTHON2\_LIBRARY=/usr/lib/x86\_64-linux-gnu/libpython2.7.so \

-D PYTHON3\_INCLUDE\_DIR=/usr/include/python3.8 \

-D PYTHON3\_NUMPY\_INCLUDE\_DIRS=/usr/lib/python3/dist-packages/numpy/core/include/ \

-D PYTHON3\_PACKAGES\_PATH=/usr/lib/python3/dist-packages \

-D PYTHON3\_LIBRARY=/usr/lib/x86\_64-linux-gnu/libpython3.8.so \

../

여기서 각각의 버전을 잘 확인하세요.

ls /usr/include 와 같이 직접 해당 디렉토리에 가서 설치된 버전을 확인해도 됩니다.

cat /proc/cpuinfo | grep processor | wc -l

cpu info 정보를 확인하거나 위 명령어를 이용하여 cpu 코어수를 확인하세요.

저의 경우는 40인데,

make -j40

이렇게 make 명령어를 넣을 때에 넣어주면 됩니다.

코어 수가 4개 정도로 적은 일반 pc의 경우는 시간이 꽤 걸리는 작업입니다.

다시 말씀드리지만 에러가 나면 설치된 버전들을 잘 확인해보세요. 저도 이것때문에 시간 엄청 잡아먹었습니다.

3 인스톨

sudo make install

위의 명령어로 인스톨을 하고,

아래 명령어로 확인해서 /usr/local/lib이 있으면 된 것입니다.

cat /etc/ld.so.conf.d/\*

해당 파일이 없다면,

sudo sh -c 'echo '/usr/local/lib' > /etc/ld.so.conf.d/opencv.conf'

을 하고,

모두 완료되었으면

sudo ldconfig

===============================================================================

**OpenCV**

**\* 기반 : opencv tutorial + 여러 응용 사례들**

Step 1 : 이미지 다루기

https://opencv-python.readthedocs.io/en/latest/doc/01.imageStart/imageStart.html

* 이미지 읽기 및 보기
* 이미지 저장하기
* Matplotlib 사용하기

https://bkshin.tistory.com/entry/OpenCV-3-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EC%9E%85%EC%B6%9C%EB%A0%A5?category=1148027

Step 2 : 비디오 다루기

https://opencv-python.readthedocs.io/en/latest/doc/02.videoStart/videoStart.html

* Camera로 부터 영상 재생
* File로 부터 영상 재생
* 영상 저장

VideoCapture 함수

• capture.get(속성) : VideoCapture의 속성을 반환합니다.

• capture.grab() : 프레임(frame)의 호출 성공 유/무를 반환합니다.

• capture.isOpened() : VideoCapture의 성공 유/무를 반환합니다.

• capture.open(카메라 장치 번호 또는 경로) : 카메라나 동영상 파일을 엽니다.

• capture.release() : VideoCapture의 장치를 닫고 메모리를 해제합니다.

• capture.retrieve() : VideoCapture의 프레임과 플래그를 반환합니다.

• capture.set(속성, 값) : VideoCapture의 속성의 값을 설정합니다.

- get(propId)



Sample Project

https://bkshin.tistory.com/entry/OpenCV-27-%ED%8A%B9%EC%A7%95-%EB%94%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%84%B0-%EA%B2%80%EC%B6%9C%EA%B8%B0-SIFT-SURF-ORB?category=1148027

[Warning or Error]

QObject::moveToThread: Current thread (0x56011c0524a0) is not the object's thread (0x56011c0d4cc0).

Cannot move to target thread (0x56011c0524a0)

→ pyqt

Step 3. Basic Operation : 내용 파악

https://opencv-python.readthedocs.io/en/latest/doc/06.operation/operation.html

* Pixel Value
* 이미지이 기본 속성
* 이미지 ROI

코드 예제

* Color space : BGR, HSV, YUV

[https://bkshin.tistory.com/entry/OpenCV-7-%E3%85%87%E3%85%87?category=1148027](https://bkshin.tistory.com/entry/OpenCV-7-ㅇㅇ?category=1148027)

* RoI : 마우스 드래그, cv2.selectROI 함수 사용

<https://bkshin.tistory.com/entry/OpenCV-6-dd?category=1148027>

Step 4. Image Arithmetic : 내용 파악

<https://opencv-python.readthedocs.io/en/latest/doc/07.imageArithmetic/imageArithmetic.html>

* 이미지 더하기 : cv2.add()
* 이미지 blending
* bit operation

코드 예제

[https://bkshin.tistory.com/entry/OpenCV-9-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EC%97%B0%EC%82%B0?category=1148027](https://bkshin.tistory.com/entry/OpenCV-9-이미지-연산?category=1148027)

* Image blending
* bitwise operation
* Image difference
* Image addition rgba and hsv masking
* Chromakey
* Seamleassclone

Step 5. Image Processing : 내용 파악

<https://opencv-python.readthedocs.io/en/latest/doc/08.imageProcessing/imageProcessing.html>

* Digital Image 유형
* Color-space 변환
* Object Tracking

코드 예제

* Thresholding : Global, Ostu, Adaptive Thresholding

[https://bkshin.tistory.com/entry/OpenCV-8-%EC%8A%A4%EB%A0%88%EC%8B%9C%ED%99%80%EB%94%A9Thresholding?category=1148027](https://bkshin.tistory.com/entry/OpenCV-8-스레시홀딩Thresholding?category=1148027)

* Histogram : Histogram (gray, color), Normalization & Equalization, CLAHE

<https://opencv-python.readthedocs.io/en/latest/doc/20.imageHistogramEqualization/imageHistogramEqualization.html>

Step6. Image Filtering : 내용 파악

* Image Smoothing : Image filtering, Smoothing

<https://opencv-python.readthedocs.io/en/latest/doc/11.imageSmoothing/imageSmoothing.html>

* Image Gradients

<https://opencv-python.readthedocs.io/en/latest/doc/13.imageGradient/imageGradient.html>

코드 예제

* Filter, Convolution operator, blur, Gaussian Blurring, Median, Bilateral Filtering

[https://bkshin.tistory.com/entry/OpenCV-17-%ED%95%84%ED%84%B0Filter%EC%99%80-%EC%BB%A8%EB%B3%BC%EB%A3%A8%EC%85%98Convolution-%EC%97%B0%EC%82%B0-%ED%8F%89%EA%B7%A0-%EB%B8%94%EB%9F%AC%EB%A7%81-%EA%B0%80%EC%9A%B0%EC%8B%9C%EC%95%88-%EB%B8%94%EB%9F%AC%EB%A7%81-%EB%AF%B8%EB%94%94%EC%96%B8-%EB%B8%94%EB%9F%AC%EB%A7%81-%EB%B0%94%EC%9D%B4%EB%A0%88%ED%84%B0%EB%9F%B4-%ED%95%84%ED%84%B0?category=1148027](https://bkshin.tistory.com/entry/OpenCV-17-필터Filter와-컨볼루션Convolution-연산-평균-블러링-가우시안-블러링-미디언-블러링-바이레터럴-필터?category=1148027)

* Gradient (edge filtering) : differential filter, Prewitt Filter, Sobel

[https://bkshin.tistory.com/entry/OpenCV-17-%ED%95%84%ED%84%B0Filter%EC%99%80-%EC](https://bkshin.tistory.com/entry/OpenCV-17-필터Filter와-컨볼루션Convolution-연산-평균-블러링-가우시안-블러링-미디언-블러링-바이레터럴-필터?category=1148027)

* Canny Edge detection

<https://docs.opencv.org/4.5.3/da/d5c/tutorial_canny_detector.html>

* Blurring Mosaic & Image Sketching

[https://bkshin.tistory.com/entry/OpenCV-21-%EB%B8%94%EB%9F%AC%EB%A7%81%EC%9D%84-%ED%99%9C%EC%9A%A9%ED%95%9C-%EB%AA%A8%EC%9E%90%EC%9D%B4%ED%81%AC-%EC%B2%98%EB%A6%AC-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EC%8A%A4%EC%BC%80%EC%B9%98-%ED%9A%A8%EA%B3%BC-%EC%A0%81%EC%9A%A9%ED%95%98%EA%B8%B0?category=1148027](https://bkshin.tistory.com/entry/OpenCV-21-블러링을-활용한-모자이크-처리-이미지-스케치-효과-적용하기?category=1148027)

Step 6. Applications

* Object Tracking API

[https://bkshin.tistory.com/entry/OpenCV-32-%EA%B0%9D%EC%B2%B4-%EC%B6%94%EC%A0%81%EC%9D%84-%EC%9C%84%ED%95%9C-Tracking-API?category=1148027](https://bkshin.tistory.com/entry/OpenCV-32-객체-추적을-위한-Tracking-API?category=1148027)

* HOG Descriptor

[https://bkshin.tistory.com/entry/OpenCV-33-HOG-%EB%94%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%84%B0HOG-Descriptor?category=1148027](https://bkshin.tistory.com/entry/OpenCV-33-HOG-디스크립터HOG-Descriptor?category=1148027)

**OpenVINO**

Ref

[https://docs.openvino.ai/2021.1/index.html#index](https://docs.openvino.ai/2021.1/index.html" \l "index)

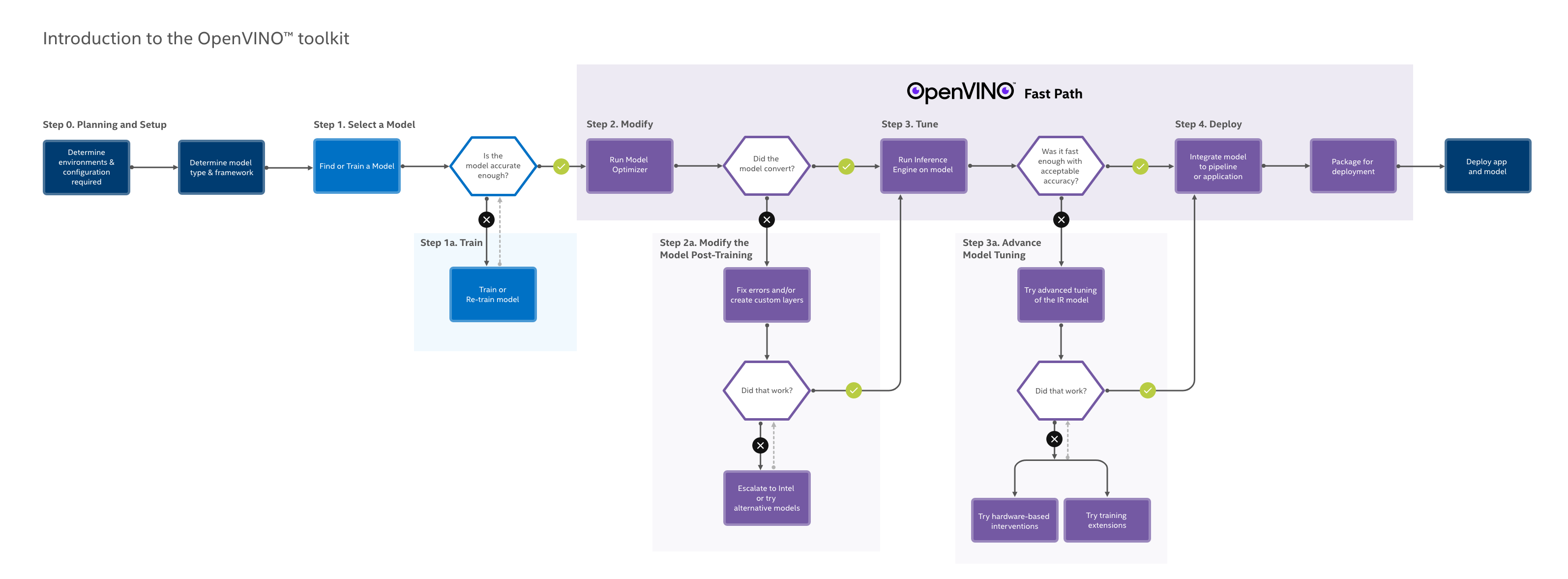
<https://github.com/openvinotoolkit/openvino>

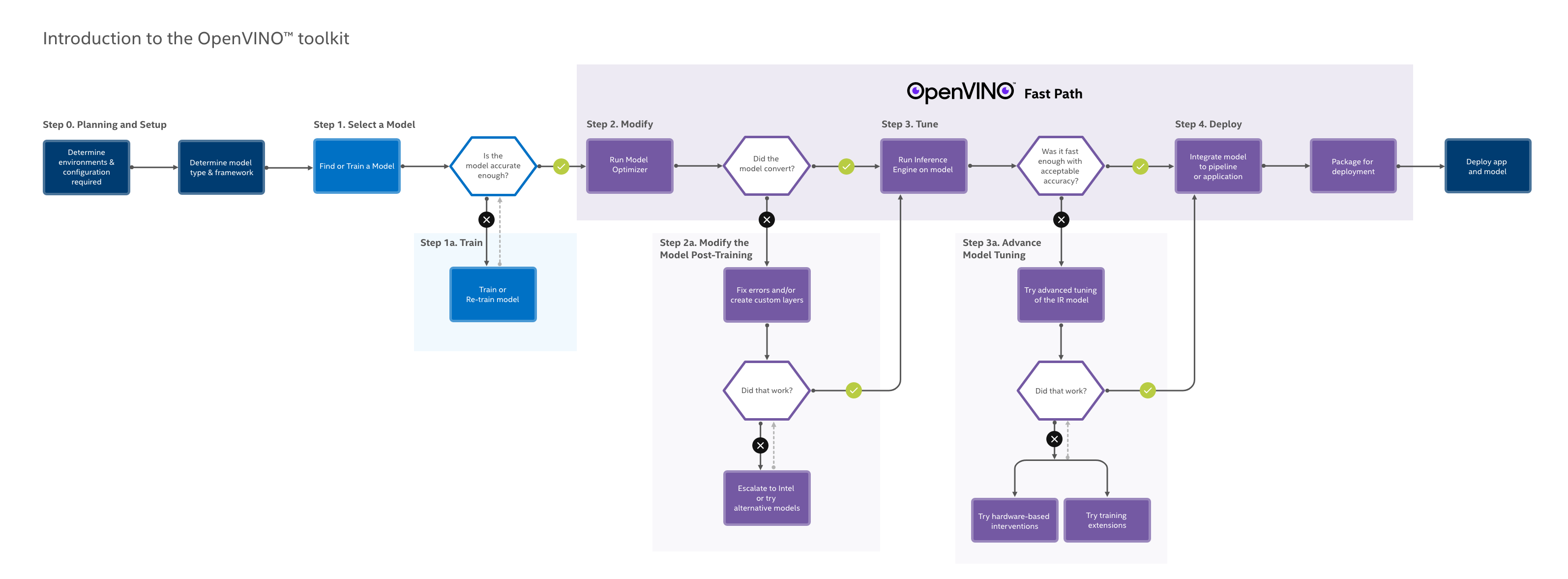
<https://github.com/openvinotoolkit/openvino_notebooks>

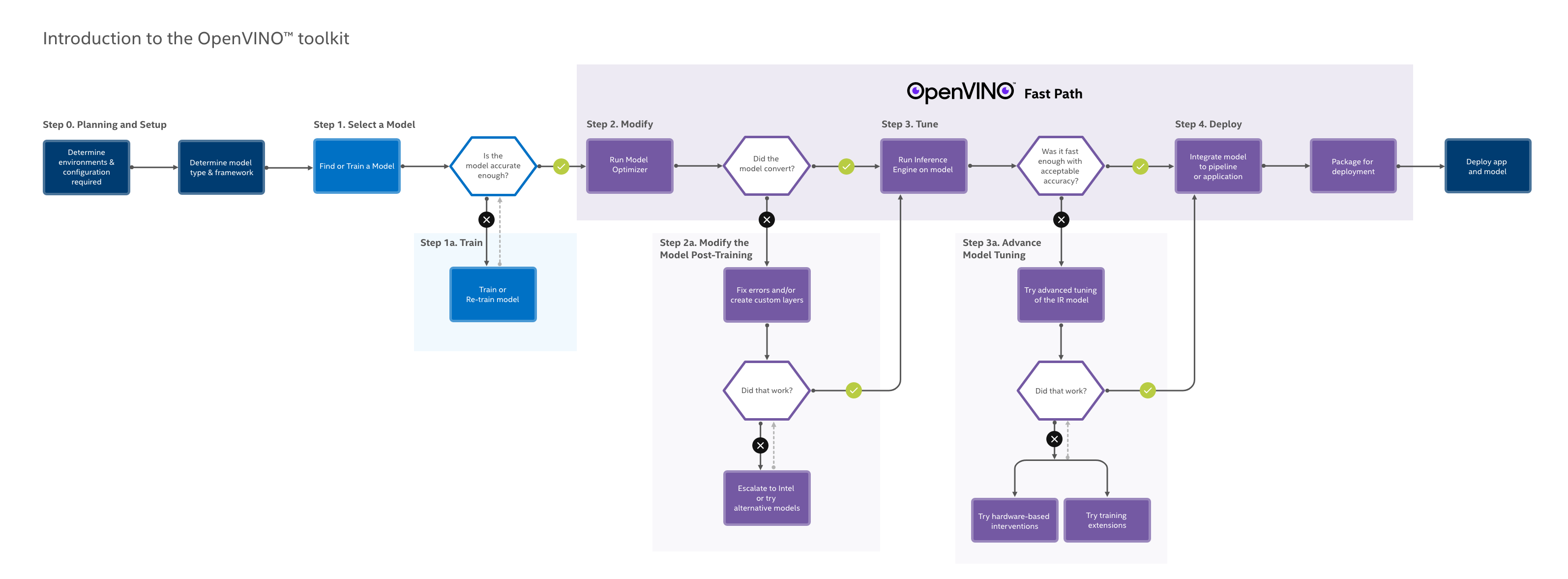
OpenVINO™ toolkit:

* Enables CNN-based deep learning inference on the edge
* Supports heterogeneous execution across an Intel® CPU, Intel® Integrated Graphics, Intel® Neural Compute Stick 2 and Intel® Vision Accelerator Design with Intel® Movidius™ VPUs
* Speeds time-to-market via an easy-to-use library of computer vision functions and pre-optimized kernels
* Includes optimized calls for computer vision standards, including OpenCV\* and OpenCL™

## OpenVINO™ Toolkit Workflow







### Model Preparation, Conversion and Optimization

You can use your framework of choice to prepare and train a Deep Learning model or just download a pretrained model from the Open Model Zoo.

* The Open Model Zoo includes Deep Learning solutions to a variety of vision problems, including object recognition, face recognition, pose estimation, text detection, and action recognition, at a range of measured complexities.
* Several of these pretrained models are used also in the [code samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) and [application demos](https://docs.openvino.ai/2021.1/omz_demos_README.html). To download models from the Open Model Zoo, the [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html) tool is used.

One of the core component of the OpenVINO™ toolkit is the [**Model Optimizer**](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html) a cross-platform command-line tool that converts a trained neural network from its source framework to an open-source, nGraph-compatible [Intermediate Representation (IR)](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_IR_and_opsets.html) for use in inference operations.

* The Model Optimizer imports models trained in popular frameworks such as **Caffe\*, TensorFlow\*, MXNet\*, Kaldi\*, and ONNX\*** and performs a few optimizations to remove excess layers and group operations when possible into simpler, faster graphs.

* If your neural network model contains layers that are not in the list of known layers for supported frameworks, you can adjust the conversion and optimization process through use of [Custom Layers](https://docs.openvino.ai/2021.1/openvino_docs_HOWTO_Custom_Layers_Guide.html).

Run the [**Accuracy Checker utility**](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html) either against source topologies or against the output representation to evaluate the accuracy of inference. The Accuracy Checker is also part of the [Deep Learning Workbench](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html), an integrated web-based performance analysis studio.

Useful documents for model optimization:

* [Model Optimizer Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html)
* [Intermediate Representation and Opsets](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_IR_and_opsets.html)
* [Custom Layers Guide](https://docs.openvino.ai/2021.1/openvino_docs_HOWTO_Custom_Layers_Guide.html)
* [Accuracy Checker utility](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html)
* [Deep Learning Workbench](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html)
* [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html) utility
* [Pretrained Models (Open Model Zoo)](https://docs.openvino.ai/2021.1/omz_models_public_index.html)

### Running and Tuning Inference

The other core component of OpenVINO™ is the [**Inference Engine**](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html),

* which manages the loading and compiling of the optimized neural network model, runs inference operations on input data, and outputs the results.
* Inference Engine can execute synchronously or asynchronously, and its plugin architecture manages the appropriate compilations for execution on multiple Intel® devices, including both workhorse CPUs and specialized graphics and video processing platforms (see below, Packaging and Deployment).

You can use OpenVINO™ **Tuning Utilities** with the Inference Engine to trial and test inference on your model. The Benchmark utility uses an input model to run iterative tests for throughput or latency measures, and the **Cross Check utility** compares performance of differently configured inferences. The [Post-Training Optimization Tool](https://docs.openvino.ai/2021.1/pot_README.html) integrates a suite of quantization- and calibration-based tools to further streamline performance.

For a full browser-based studio integrating these other key tuning utilities, try the [Deep Learning Workbench](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html).

OpenVINO™ toolkit includes a set of [inference code samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) and [application demos](https://docs.openvino.ai/2021.1/omz_demos_README.html) showing how inference is run and output processed for use in retail environments, classrooms, smart camera applications, and other solutions.

OpenVINO also makes use of open-Source and Intel™ tools for traditional graphics processing and performance management. Intel® Media SDK supports accelerated rich-media processing, including transcoding. OpenVINO™ optimizes calls to the rich OpenCV and OpenVX libraries for processing computer vision workloads. And the new DL Streamer integration further accelerates video pipelining and performance.

Useful documents for inference tuning:

* [Inference Engine Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html)
* [Inference Engine API References](https://docs.openvino.ai/2021.1/api_references.html)
* [Inference Code Samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html)
* [Application Demos](https://docs.openvino.ai/2021.1/omz_demos_README.html)
* [Post-Training Optimization Tool Guide](https://docs.openvino.ai/2021.1/pot_README.html)
* [Deep Learning Workbench Guide](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html)
* [Intel Media SDK](https://github.com/Intel-Media-SDK/MediaSDK)
* [DL Streamer Samples](https://docs.openvino.ai/2021.1/gst_samples_README.html)
* [OpenCV](https://docs.opencv.org/master/)
* [OpenVX](https://software.intel.com/en-us/openvino-ovx-guide)

### Packaging and Deployment

The Intel Distribution of OpenVINO™ toolkit outputs optimized inference runtimes for the following devices:

* Intel® CPUs
* Intel® Processor Graphics
* Intel® Neural Compute Stick 2
* Intel® Vision Accelerator Design with Intel® Movidius™ VPUs

The Inference Engine's plug-in architecture can be extended to meet other specialized needs. [Deployment Manager](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_deployment_manager_tool.html) is a Python\* command-line tool that assembles the tuned model, IR files, your application, and required dependencies into a runtime package for your target device. It outputs packages for CPU, GPU, and VPU on Linux\* and Windows\*, and Neural Compute Stick-optimized packages with Linux.

* [Inference Engine Integration Workflow](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Integrate_with_customer_application_new_API.html)
* [Inference Engine API References](https://docs.openvino.ai/2021.1/api_references.html)
* [Inference Engine Plug-in Developer Guide](https://docs.openvino.ai/2021.1/ie_plugin_api/index.html)
* [Deployment Manager Guide](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_deployment_manager_tool.html)

## OpenVINO™ Toolkit Components

Intel® Distribution of OpenVINO™ toolkit includes the following components:

* [Deep Learning Model Optimizer](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html) - A cross-platform command-line tool for importing models and preparing them for optimal execution with the Inference Engine. The Model Optimizer imports, converts, and optimizes models, which were trained in popular frameworks, such as Caffe\*, TensorFlow\*, MXNet\*, Kaldi\*, and ONNX\*.
* [Deep Learning Inference Engine](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_inference_engine_intro.html) - A unified API to allow high performance inference on many hardware types including Intel® CPU, Intel® Integrated Graphics, Intel® Neural Compute Stick 2, Intel® Vision Accelerator Design with Intel® Movidius™ vision processing unit (VPU)
* [Inference Engine Samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) - A set of simple console applications demonstrating how to use the Inference Engine in your applications
* Additional Tools - A set of tools to work with your models including [Accuracy Checker utility](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html), [Post-Training Optimization Tool Guide](https://docs.openvino.ai/2021.1/pot_README.html), [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html) and other
* [Open Model Zoo](https://docs.openvino.ai/2021.1/omz_models_intel_index.html)
  + [Demos](https://docs.openvino.ai/2021.1/omz_demos_README.html) - Console applications that demonstrate how you can use the Inference Engine in your applications to solve specific use cases
  + Additional Tools - A set of tools to work with your models including [Accuracy Checker utility](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html), [Post-Training Optimization Tool Guide](https://docs.openvino.ai/2021.1/pot_README.html), [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html) and other
  + [Documentation for Pretrained Models](https://docs.openvino.ai/2021.1/omz_models_intel_index.html) - Documentation for pretrained models that are available in the [Open Model Zoo repository](https://github.com/opencv/open_model_zoo)
* [Post-Training Optimization tool](https://docs.openvino.ai/2021.1/pot_README.html) - A tool to calibrate a model and then execute it in the INT8 precision
* [Deep Learning Workbench](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html) - A web-based graphical environment that allows you to easily use various sophisticated OpenVINO™ toolkit components
* Deep Learning Streamer (DL Streamer) – Streaming analytics framework, based on GStreamer, for constructing graphs of media analytics components. DL Streamer can be installed by the Intel® Distribution of OpenVINO™ toolkit installer. Its open source version is available on [GitHub](https://github.com/opencv/gst-video-analytics). For the DL Streamer documentation, see:
  + [DL Streamer Samples](https://docs.openvino.ai/2021.1/gst_samples_README.html)
  + [API Reference](https://openvinotoolkit.github.io/dlstreamer_gst/)
  + [Elements](https://github.com/opencv/gst-video-analytics/wiki/Elements)
  + [Tutorial](https://github.com/opencv/gst-video-analytics/wiki/DL Streamer Tutorial)
* [OpenCV](https://docs.opencv.org/master/) - OpenCV\* community version compiled for Intel® hardware
* [Intel® Media SDK](https://software.intel.com/en-us/media-sdk) (in Intel® Distribution of OpenVINO™ toolkit for Linux only)

<https://github.com/openvinotoolkit/openvino_notebooks>

**OpenVINO™ Notebooks**

## 1. Install Python, Git and GPU drivers (optional)

You may need to install some additional libraries on Ubuntu Linux. These steps work on a clean install of Ubuntu Desktop 20.04, and should also work on Ubuntu 18.04 and 20.10, and on Ubuntu Server.

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install python3-venv build-essential python3-dev git-all

If you have a CPU with an Intel Integrated Graphics Card, you can install the [Intel Graphics Compute Runtime](https://github.com/intel/compute-runtime) to enable inference on this device. The command for Ubuntu 20.04 is:

sudo apt-get install intel-opencl-icd

See the [documentation](https://github.com/intel/compute-runtime) for other installation methods and instructions for other versions.

## 2. Install the Notebooks

After installing Python 3 and Git, run each step below in a terminal. Note: If OpenVINO is installed globally, please do not run any of these commands in a terminal where setupvars.sh is sourced.

## 3. Create a Virtual Environment

Note: If you already installed openvino-dev and activated the openvino\_env environment, you can skip to [Step 4](https://github.com/openvinotoolkit/openvino_notebooks/wiki/Ubuntu" \l "4-clone-the-repository). If you use Anaconda, please see the [Conda guide](https://github.com/openvinotoolkit/openvino_notebooks/wiki/Conda).

python3 -m venv openvino\_env

python3 -m venv openvino/openvino\_env

cd openvino

## 4. Activate the Environment

source openvino\_env/bin/activate

## 4. Clone the Repository

git clone https://github.com/openvinotoolkit/openvino\_notebooks.git

cd openvino\_notebooks

## 5. Install the Packages

This step installs OpenVINO and dependencies like Jupyter Lab. First, upgrade pip to the latest version. Then, install the required dependencies.

python -m pip install --upgrade pip

pip install -r requirements.txt

[Note] requirements.txt

|  |
| --- |
| openvino-dev[onnx,tensorflow2]==2021.4.\*  matplotlib<3.4  gdown  pytube  openvino-extensions  yaspin  # ONNX notebook requirements  geffnet==0.9.8  fastseg  ipywidgets  torch>=1.5.1,<=1.7.1; sys\_platform == 'darwin'  torchvision>=0.6.1,<=0.8.2; sys\_platform == 'darwin'  --find-links https://download.pytorch.org/whl/torch\_stable.html  torch>=1.5.1+cpu,<=1.7.1+cpu; sys\_platform =='linux' or platform\_system == 'Windows'  torchvision>=0.6.1+cpu,<=0.8.2+cpu; sys\_platform =='linux' or platform\_system == 'Windows'  # PaddlePaddle notebook requirements  # For 103 PaddlePaddle MO conversion tutorial and 206 PaddleGAN/AnimeGAN demo  paddlepaddle==2.1.\*  paddlehub  paddle2onnx>=0.6  ppgan>=2.0.0  # BERT quantization notebook requirements  transformers  sklearn  # Jupyter requirements  jupyterlab  # Pin versions to prevent known dependency issues  ipython==7.10.\*  jedi==0.17.2  setuptools>=56.0.0  Pillow>=8.3.2  ipykernel==5.\*  pygments>=2.7.4 # not directly required, pinned by Snyk to avoid a vulnerability  nltk>=3.6.4 # not directly required, pinned by Snyk to avoid a vulnerability  rsa>=4.7 # not directly required, pinned by Snyk to avoid a vulnerability  scikit-learn>=0.24.2 # not directly required, pinned by Snyk to avoid a vulnerability  # NNCF notebook requirements  nncf[torch]  nncf[tf]  tensorflow\_datasets==4.2.0 |

## 6. Install the virtualenv Kernel in Jupyter

python -m ipykernel install --user --name openvino\_env

### 7. Launch the Notebooks!

To launch a single notebook, like the Monodepth notebook

jupyter notebook notebooks/201-vision-monodepth/201-vision-monodepth.ipynb

To launch all notebooks in Jupyter Lab

jupyter lab notebooks

In Jupyter Lab, select a notebook from the file browser using the left sidebar. Each notebook is located in a subdirectory within the notebooks directory.

## Troubleshooting

* On Ubuntu 18.04, python3-dev installs the required libraries for the system default version of Python. On Ubuntu 18 this is Python 3.6, on Ubuntu 20.04, Python 3.8. If you also installed other versions of Python, it is recommended to use the full path the to system default Python: /usr/bin/python3.6 -m venv openvino\_env on Ubuntu 18, /usr/bin/python3.8 -m venv openvino\_env on Ubuntu 20.
* If you use Anaconda or Miniconda, see the [Conda](https://github.com/openvinotoolkit/openvino_notebooks/wiki/Conda) wiki page.
* On Ubuntu, if you see the error "libpython3.7m.so.1.0: cannot open shared object file: No such object or directory" please install the required package using apt install libpython3.7-dev.

## Cleaning Up

### Shut Down Jupyter Kernel

To end your Jupyter session, press Ctrl-c. This will prompt you to Shutdown this Jupyter server (y/[n])? enter y and hit Enter.

### Deactivate Virtual Environment

To deactivate your virtualenv, simply run deactivate from the terminal window where you activated openvino\_env. This will deactivate your environment.

To reactivate your environment, run source openvino\_env/bin/activate on Linux or openvino\_env\Scripts\activate on Windows, then type jupyter lab or jupyter notebook to launch the notebooks again.

### Delete Virtual Environment (Optional)

To remove your virtual environment, simply delete the openvino\_env directory:

#### On Linux and macOS:

rm -rf openvino\_env

#### On Windows:

rmdir /s openvino\_env

### Remove openvino\_env Kernel from Jupyter

jupyter kernelspec remove openvino\_env

===============================================================================

**[Note]**

**VSCode / Pycharm → Python Interpreter 설정**

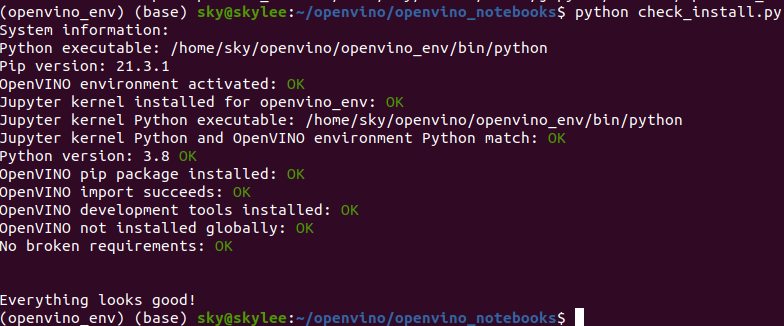
**/home/<user>/openvino/openvino\_env/bin/python3**

===============================================================================

## Troubleshooting

If these tips do not solve your problem, please open a [discussion topic](https://github.com/openvinotoolkit/openvino_notebooks/discussions) or create an [issue](https://github.com/openvinotoolkit/openvino_notebooks/issues)!

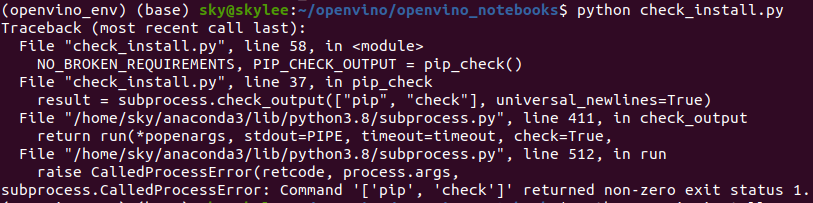
* To check some common installation problems, run python check\_install.py. This script is located in the openvino\_notebooks directory. Please run it after activating the openvino\_env virtual environment.
* If you get an ImportError, doublecheck that you installed the Jupyter kernel. If necessary, choose the openvinoenv kernel from the \_Kernel->Change Kernel menu) in Jupyter Lab or Jupyter Notebook
* If OpenVINO is installed globally, do not run installation commands in a terminal where setupvars.bat or setupvars.sh are sourced.
* For Windows installation, we recommend using Command Prompt (cmd.exe), not PowerShell.



[Note] subprocess.calledProceeError : pip, check

python -m pip install --upgrade pip

pip install -r requirements.txt



<https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html>

[https://docs.openvino.ai/2021.4/openvino\_docs\_install\_guides\_installing\_openvino\_linux.html#install-external-dependencies](https://docs.openvino.ai/2021.4/openvino_docs_install_guides_installing_openvino_linux.html" \l "install-external-dependencies)

**Install Intel® Distribution of OpenVINO™ toolkit for Linux\***

**Included with the Installation and installed by default:**

|  |  |
| --- | --- |
| Component | Description |
| [Model Optimizer](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html) | * This tool imports, converts, and optimizes models that were trained in popular frameworks to a format usable by Intel tools, especially the Inference Engine. * Popular frameworks include Caffe\*, TensorFlow\*, MXNet\*, and ONNX\*. |
| [Inference Engine](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_inference_engine_intro.html) | * This is the engine that runs the deep learning model. It includes a set of libraries for an easy inference integration into your applications. |
| Intel® Media SDK | * Offers access to hardware accelerated video codecs and frame processing |
| [OpenCV](https://docs.opencv.org/master/) | * OpenCV\* community version compiled for Intel® hardware |
| [Inference Engine Code Samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) | * A set of simple console applications demonstrating how to utilize specific OpenVINO capabilities in an application and how to perform specific tasks, such as loading a model, running inference, querying specific device capabilities, and more. |
| [Demo Applications](https://docs.openvino.ai/2021.1/omz_demos_README.html) | * A set of simple console applications that provide robust application templates to help you implement specific deep learning scenarios. |
| Additional Tools | * A set of tools to work with your models including [Accuracy Checker utility](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html), [Post-Training Optimization Tool Guide](https://docs.openvino.ai/2021.1/pot_README.html), [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html) and other |
| [Documentation for Pre-Trained Models](https://docs.openvino.ai/2021.1/omz_models_intel_index.html) | * Documentation for the pre-trained models available in the [Open Model Zoo repo](https://github.com/opencv/open_model_zoo). |
| Deep Learning Streamer (DL Streamer) | * Streaming analytics framework, based on GStreamer, for constructing graphs of media analytics components. For the DL Streamer documentation, see [DL Streamer Samples](https://docs.openvino.ai/2021.1/gst_samples_README.html), [API Reference](https://openvinotoolkit.github.io/dlstreamer_gst/), [Elements](https://github.com/opencv/gst-video-analytics/wiki/Elements), [Tutorial](https://github.com/opencv/gst-video-analytics/wiki/DL Streamer Tutorial). |

**Could Be Optionally Installed**

[Deep Learning Workbench](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Introduction.html) (DL Workbench) is a platform built upon OpenVINO™ and provides a web-based graphical environment that enables you to optimize, fine-tune, analyze, visualize, and compare performance of deep learning models on various Intel® architecture configurations. In the DL Workbench, you can use most of OpenVINO™ toolkit components:

* [Model Downloader](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html)
* [Intel® Open Model Zoo](https://docs.openvino.ai/2021.1/omz_models_intel_index.html)
* [Model Optimizer](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html)
* [Post-training Optimization Tool](https://docs.openvino.ai/2021.1/pot_README.html)
* [Accuracy Checker](https://docs.openvino.ai/2021.1/omz_tools_accuracy_checker_README.html)
* [Benchmark Tool](https://docs.openvino.ai/2021.1/openvino_inference_engine_samples_benchmark_app_README.html)

Proceed to an [easy installation from Docker](https://docs.openvino.ai/2021.1/workbench_docs_Workbench_DG_Install_from_Docker_Hub.html) to get started.

## System Requirements

**Hardware**

* 6th to 11th generation Intel® Core™ processors and Intel® Xeon® processors
* Intel® Xeon® processor E family (formerly code named Sandy Bridge, Ivy Bridge, Haswell, and Broadwell)
* 3rd generation Intel® Xeon® Scalable processor (formerly code named Cooper Lake)
* Intel® Xeon® Scalable processor (formerly Skylake and Cascade Lake)
* Intel Atom® processor with support for Intel® Streaming SIMD Extensions 4.1 (Intel® SSE4.1)
* Intel Pentium® processor N4200/5, N3350/5, or N3450/5 with Intel® HD Graphics
* Intel® Neural Compute Stick 2
* Intel® Vision Accelerator Design with Intel® Movidius™ VPUs

**NOTE**: With OpenVINO™ 2020.4 release, Intel® Movidius™ Neural Compute Stick is no longer supported.

**Processor Notes:**

* Processor graphics are not included in all processors. See [Product Specifications](https://ark.intel.com/) for information about your processor.
* A chipset that supports processor graphics is required for Intel® Xeon® processors.

**Operating Systems**

* Ubuntu 18.04.x long-term support (LTS), 64-bit
* CentOS 7.6, 64-bit (for target only)
* Yocto Project v3.0, 64-bit (for target only and requires modifications)

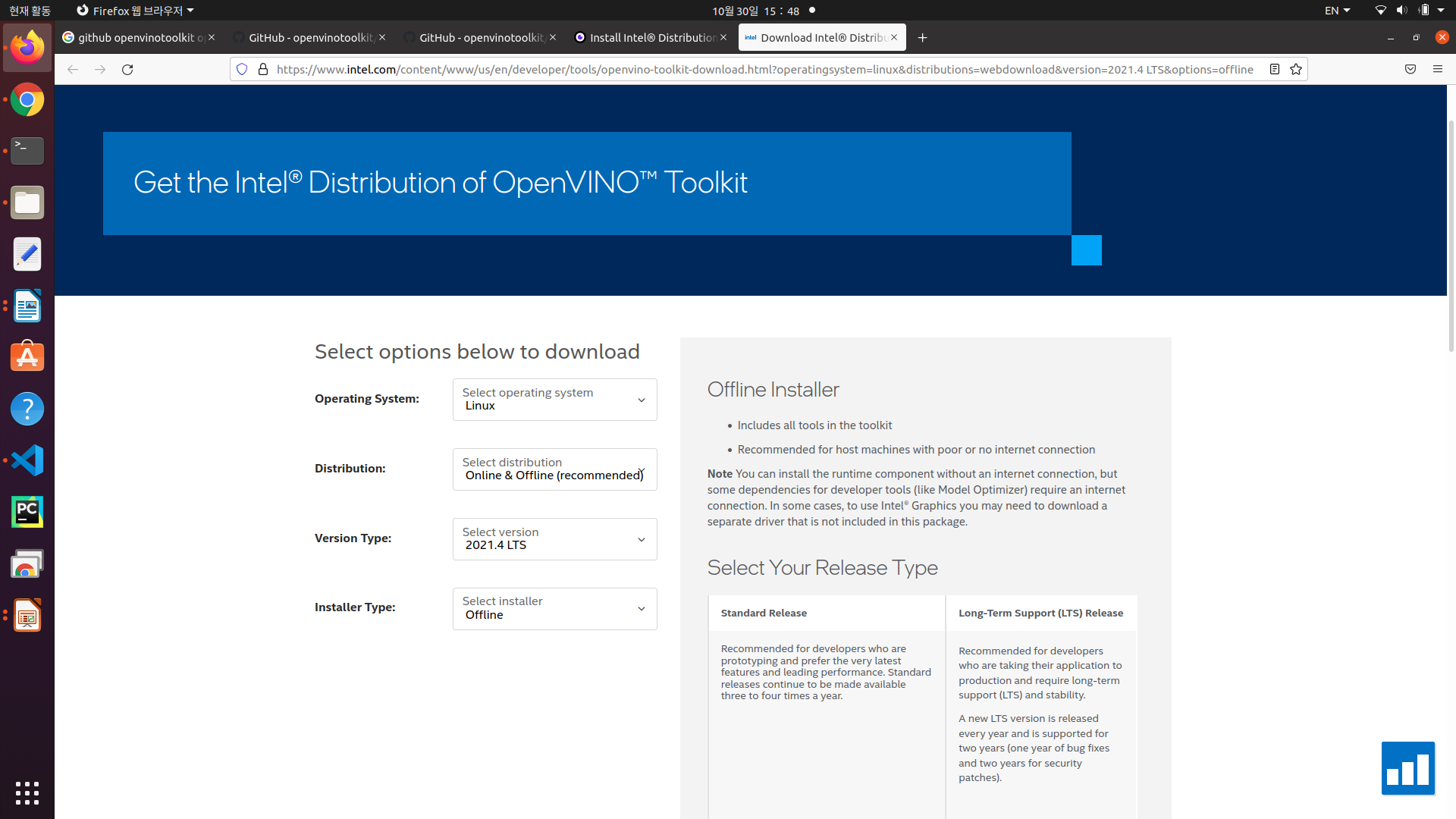
## Overview

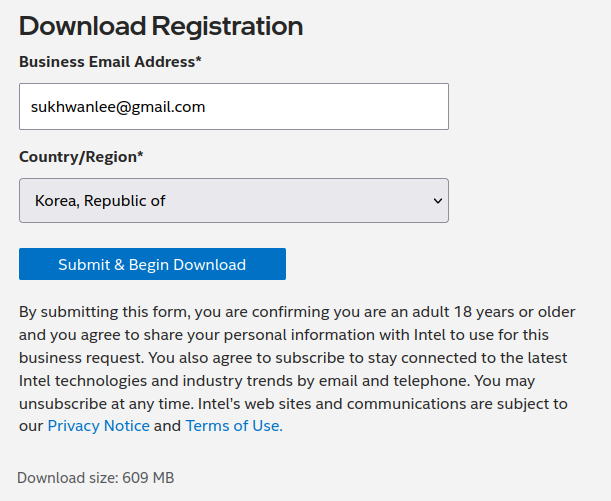
This guide provides step-by-step instructions on how to install the Intel® Distribution of OpenVINO™ toolkit. Links are provided for each type of compatible hardware including downloads, initialization and configuration steps. The following steps will be covered:

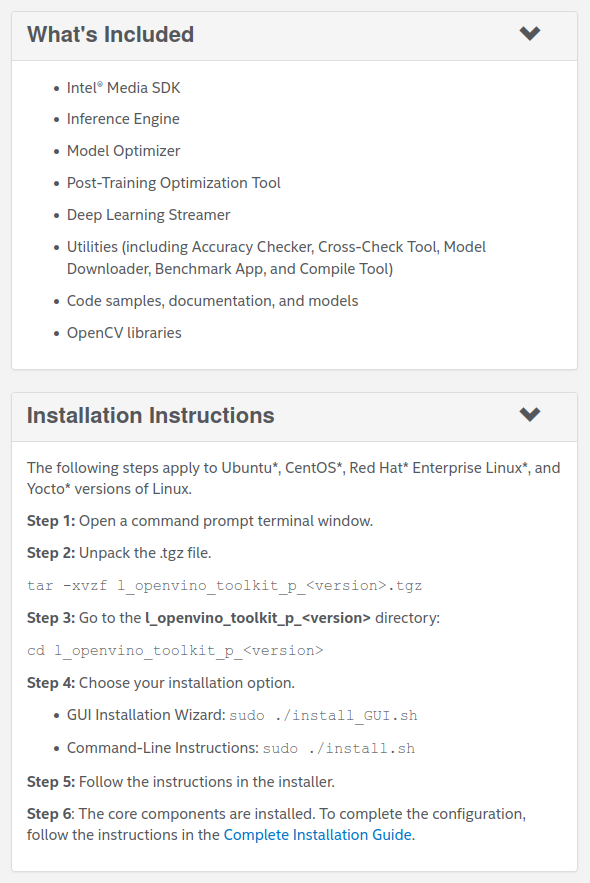
1. [Install the Intel® Distribution of OpenVINO™ Toolkit](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "install-openvino)
2. [Install External software dependencies](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "install-external-dependencies)
3. [Set the OpenVINO™ Environment Variables: Optional Update to .bashrc](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "set-the-environment-variables).
4. [Configure the Model Optimizer](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "configure-model-optimizer)
5. [Run the Verification Scripts to Verify Installation and Compile Samples](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "run-the-demos)
6. [Steps for Intel® Processor Graphics (GPU)](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "additional-GPU-steps)
7. [Steps for Intel® Neural Compute Stick 2](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "additional-NCS-steps)
8. [Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPU](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "install-VPU)  
   After installing your Intel® Movidius™ VPU, you will return to this guide to complete OpenVINO™ installation.
9. [Run a Sample Application](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "run-a-sample)
10. [Use the Face Detection Tutorial](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "Hello-World-Face-Detection-Tutorial)

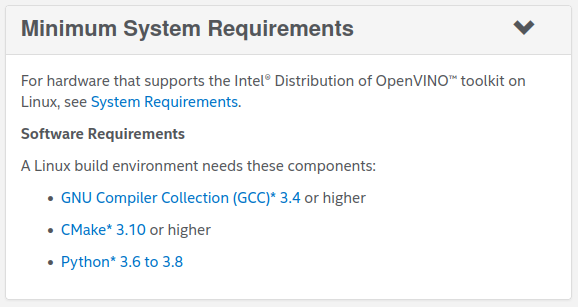
## Step 1. Install the Intel® Distribution of OpenVINO™ Toolkit Core Components

Download the Intel® Distribution of OpenVINO™ toolkit package file from [Intel® Distribution of OpenVINO™ toolkit for Linux\*](https://software.intel.com/en-us/openvino-toolkit/choose-download). Select the Intel® Distribution of OpenVINO™ toolkit for Linux package from the dropdown menu.









[Preparation before OpenVINO installation]

* GCC 설치 확인 : gcc –version
* Cmake 설치 확인 : gmake –version
* Python 3.8 설치 확인

**2021년10월 기준 설치 버전 : l\_openvino\_toolkit\_p\_2021.4.582**

1. Open a command prompt terminal window.
2. Change directories to where you downloaded the Intel Distribution of OpenVINO toolkit for Linux\* package file.  
   If you downloaded the package file to the current user's Downloads directory:

content\_copy

cd ~/Downloads/

By default, the file is saved as l\_openvino\_toolkit\_p\_<version>.tgz.

1. Unpack the .tgz file:

content\_copy

tar -xvzf l\_openvino\_toolkit\_p\_<version>.tgz

The files are unpacked to the l\_openvino\_toolkit\_p\_<version> directory.

1. Go to the l\_openvino\_toolkit\_p\_<version> directory:

content\_copy

cd l\_openvino\_toolkit\_p\_<version>

If you have a previous version of the Intel Distribution of OpenVINO toolkit installed, rename or delete these two directories:

* ~/inference\_engine\_samples\_build
* ~/openvino\_models

**Installation Notes:**

* + Choose an installation option and run the related script as root.
  + You can use either a GUI installation wizard or command-line instructions (CLI).
  + Screenshots are provided for the GUI, but not for CLI. The following information also applies to CLI and will be helpful to your installation where you will be presented with the same choices and tasks.

1. Choose your installation option:
   * **Option 1:** GUI Installation Wizard:

content\_copy

sudo ./install\_GUI.sh

* + **Option 2:** Command-Line Instructions:

content\_copy

sudo ./install.sh

* + **Option 3:** Command-Line Silent Instructions:

content\_copy

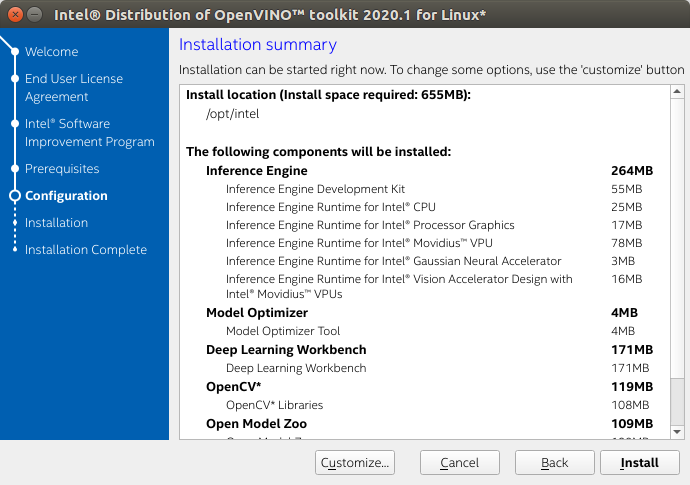
sudo sed -i 's/decline/accept/g' silent.cfg

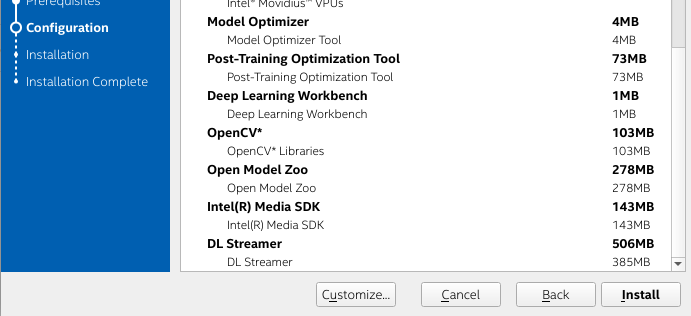
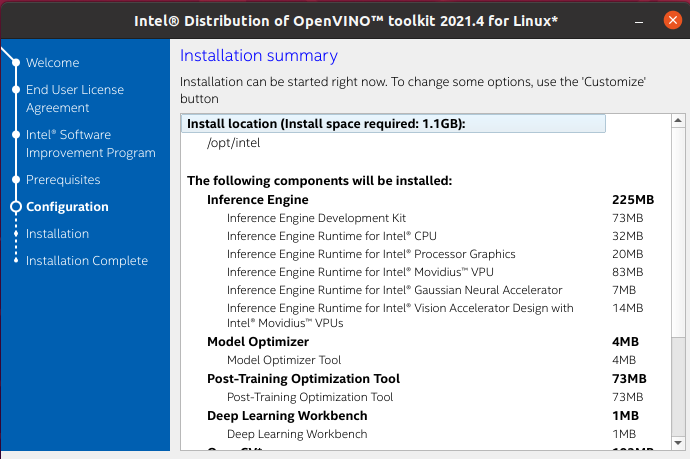
sudo ./install.sh -s silent.cfg

You can select which OpenVINO components will be installed by modifying the COMPONENTS parameter in the silent.cfg file. For example, to install only CPU runtime for the Inference Engine, set COMPONENTS=intel-openvino-ie-rt-cpu\_\_x86\_64 in silent.cfg. To get a full list of available components for installation, run the ./install.sh --list\_components command from the unpacked OpenVINO™ toolkit package.

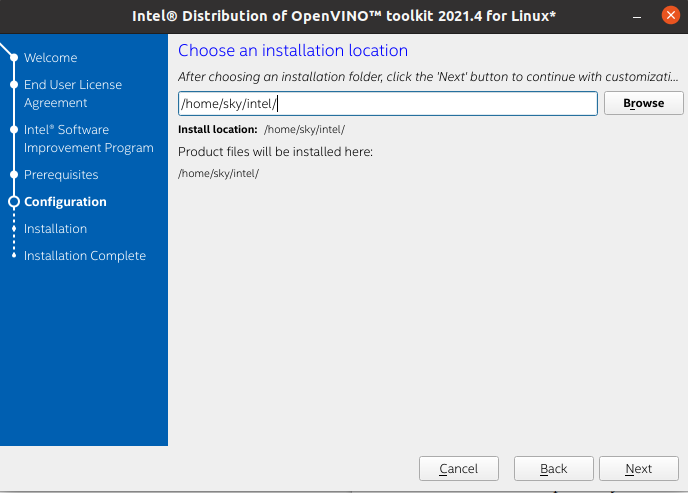
1. Follow the instructions on your screen. Watch for informational messages such as the following in case you must complete additional steps:

1. If you select the default options, the **Installation summary** GUI screen looks like this:





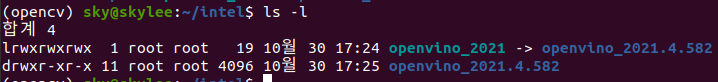
* + **Optional:** You can choose **Customize** to change the installation directory or the components you want to install: **/home /<user>/intel/**



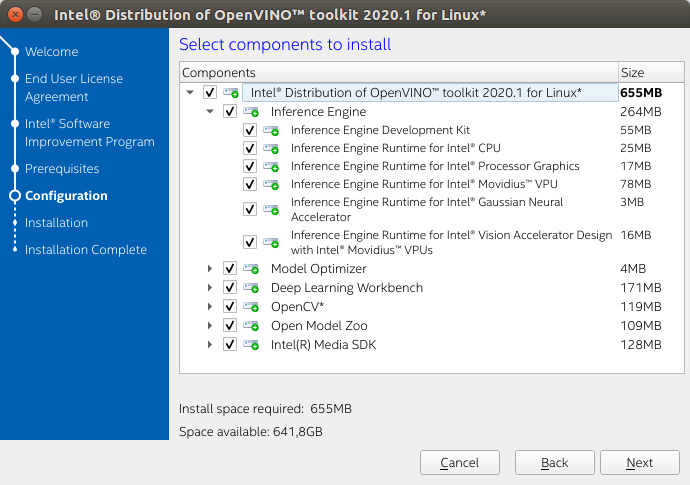
By default, the Intel® Distribution of OpenVINO™ is installed to the following directory:

* + For root or administrator: /opt/intel/openvino\_<version>/
  + For regular users: /home/<USER>/intel/openvino\_<version>/

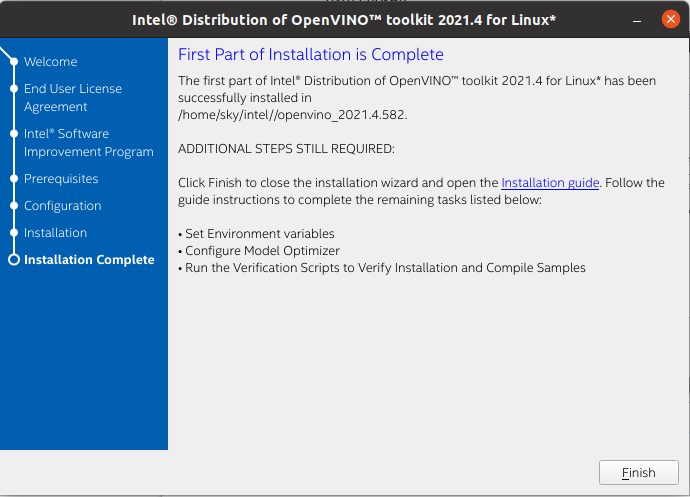
For simplicity, a symbolic link to the latest installation is also created: /opt/intel/openvino\_2021/ or /home/<USER>/intel/openvino\_2021/



**NOTE**: The Intel® Media SDK component is always installed in the /opt/intel/mediasdk directory regardless of the OpenVINO installation path chosen.



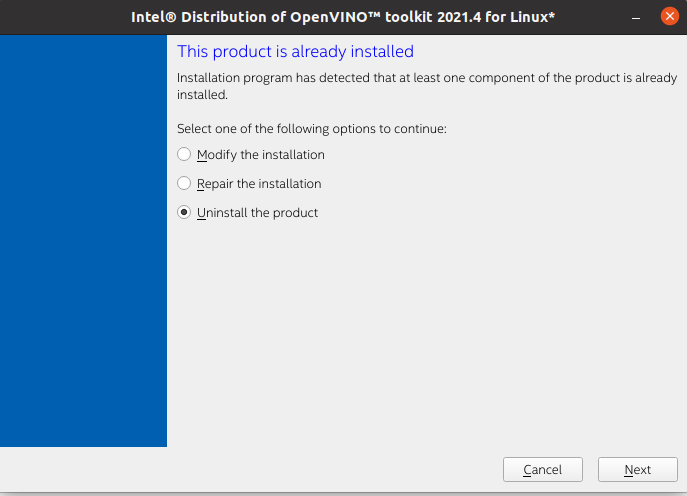
1. A Complete screen indicates that the core components have been installed:



The first core components are installed. Continue to the next section to install additional dependencies.

[Uninstall]

sudo ./install\_GUI.sh



## Step 2 : Install External Software Dependencies

**NOTE**: If you installed the Intel® Distribution of OpenVINO™ to the non-default install directory, replace /opt/intel with the directory in which you installed the software.

These dependencies are required for:

* Intel-optimized build of OpenCV library
* Deep Learning Inference Engine
* Deep Learning Model Optimizer tools

1. Change to the install\_dependencies directory:

cd /opt/intel/openvino\_2021/install\_dependencies

cd /home/<user>/intel/openvino\_2021/install\_dependencies

1. Run a script to download and install the external software dependencies:

sudo -E ./install\_openvino\_dependencies.sh

The dependencies are installed. Continue to the next section to set your environment variables.

## Step 3 : Configure the Environment Variables

You must update several environment variables before you can compile and run OpenVINO™ applications. Set persistent environment variables as follows, using vi (as below) or your preferred editor:

1. Open the .bashrc file in /home/<USER> :

vi **~/**.bashrc

gedit ~/.bashrc

1. Press the **I** key to switch to insert mode.
2. Add this line to the end of the file:

source **/**opt**/**intel**/**openvino\_2021**/**bin**/**setupvars.sh

1. source /home/<user>/intel**/**openvino\_2021**/**bin**/**setupvars.sh
2. Save and close the file: press the **Esc** key and type :wq.
3. To verify the change, open a new terminal. You will see [setupvars.sh] OpenVINO environment initialized.

**Optional:** If you don’t want to change your shell profile, you can run the following script to temporarily set your environment variables for each terminal instance when working with OpenVINO™:

source /home/<user>/intel**/**openvino\_2021**/**bin**/**setupvars.sh

The environment variables are set. Next, you will configure the Model Optimizer.

## Step 4 : Configure the Model Optimizer

The Model Optimizer is a Python\*-based command line tool for importing trained models from popular deep learning frameworks such as Caffe\*, TensorFlow\*, Apache MXNet\*, ONNX\* and Kaldi\*.

The Model Optimizer is a key component of the Intel Distribution of OpenVINO toolkit. You cannot perform 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 Intermediate Representation is a pair of files that describe the whole model:

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

For more information about the Model Optimizer, refer to the [Model Optimizer Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html).

**Configure all supported frameworks at the same time**

1. Go to the Model Optimizer prerequisites directory:

cd /home/<user>/intel/openvino\_2021/deployment\_tools/model\_optimizer/install\_prerequisites

cd /opt/intel/openvino\_2021/deployment\_tools/model\_optimizer/install\_prerequisites

1. Run the script to configure the Model Optimizer for Caffe, TensorFlow 1.x, MXNet, Kaldi\*, and ONNX:

content\_copy

sudo ./install\_prerequisites.sh

* **Note:** You can choose to install Model Optimizer support for only certain frameworks. In the same directory are individual scripts for Caffe, TensorFlow 1.x, TensorFlow 2.x, MXNet, Kaldi, and ONNX (install\_prerequisites\_caffe.sh, etc.).

The Model Optimizer is configured for one or more frameworks. You are ready to compile the samples by [running the verification scripts](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "run-the-demos).

## Run the Verification Scripts to Verify Installation (데모 프로그램 실행)

**IMPORTANT**: This section is required. In addition to confirming our installation was successful, demo scripts perform other steps, such as setting up your computer to use the Inference Engine samples.

To verify the installation and compile two samples, use the steps below to run the verification applications provided with the product on the CPU.

1. Go to the **Inference Engine demo** directory:

cd /home/<user>/intel/openvino\_2021/deployment\_tools/demo

1. Run the **Image Classification verification script**:

./demo\_squeezenet\_download\_convert\_run.sh

This verification script downloads a SqueezeNet model, uses the Model Optimizer to convert the model to the .bin and .xml Intermediate Representation (IR) files. The Inference Engine requires this model conversion so it can use the IR as input and achieve optimum performance on Intel hardware.  
This verification script builds the [Image Classification Sample Async](https://docs.openvino.ai/2021.1/openvino_inference_engine_samples_classification_sample_async_README.html) application and run it with the car.png image located in the demo directory. When the verification script completes, you will have the label and confidence for the top-10 categories:

|  |
| --- |
| pip3 동작하지 않을 때 (pip upgrade 후) 또는 pyyaml 등 오류 발생시  sudo python3 -m pip uninstall pip && sudo apt install python3-pip –reinstall  증상  sudo apt-get install python3-pip 로 pip3를 설치했으나 아래와 비슷한 에러가 발생할 때  Traceback (most recent call last):  File "/usr/bin/pip3", line 9, in <module>  from pip import main  ImportError: cannot import name 'main'  해결방법 : 스택오버플로우 참조  \*sudo pip install pip –upgrade 같은 명령으로 무심코 시스템 pip를 업그레이드 한 것이 원인입니다. pip 10.x은 내부적으로 위치해야 할 곳을 정하게 되어 있으며, pip3 명령은 pip에 의해 관리되는 파일이 아니라 패키지 관리자에 의해 제공되는 파일입니다. 더 자세한 정보는 여기를 참고하세요. 시스템 pip를 업그레이드 하지 않고 virtualenv를 사용하고 싶다면 아래의 명령으로 pip3를 복구해야 합니다.\*  sudo python3 -m pip uninstall pip && sudo apt install python3-pip –reinstall  python3-pip : python3용 pip - Python 모듈을 구축하는 데 필요한 모든 종속성도 설치  pip3 재설치 |

쉘 스크립트 : gedit ./demo\_squeezenet\_download\_convert\_run.sh

**Step 1) Download the Caffe model and the prototxt of the model**

* + 모델 최적화가 필요한 구성 요소 설치

|  |
| --- |
| sudo -E apt update  sudo -E apt -y install build-essential python3-pip virtualenv cmake libcairo2-dev libpango1.0-dev libglib2.0-dev libgtk2.0-dev libswscale-dev libavcodec-dev libavformat-dev libgstreamer1.0-0 gstreamer1.0-plugins-base  sudo -E apt-get install -y libpng-dev |

* + 모델 다운로더에 의해 squeezenet 1.1의 학습 완료 모델을 다운로드한다. 모델은 $Home/openvino\_modles/models/에 저장된다.

|  |
| --- |
| model\_name="squeezenet1.1"  models\_path="$HOME/openvino\_models/models"  models\_cache="$HOME/openvino\_models/cache"  python\_binary=python3  downloader\_path="${INTEL\_OPENVINO\_DIR}/deployment\_tools/open\_model\_zoo/tools/downloader/downloader.py"  print\_and\_run "$python\_binary" "$downloader\_path" --name "$model\_name" --output\_dir "${models\_path}" --cache\_dir "${models\_cache}" |

Run python3 /home/sky/intel/openvino\_2021.4.582/deployment\_tools/open\_model\_zoo/tools/downloader/downloader.py --name squeezenet1.1 --output\_dir /root/openvino\_models/models --cache\_dir /root/openvino\_models/cache

========== Downloading /root/openvino\_models/models/public/squeezenet1.1/squeezenet1.1.prototxt

... 100%, 9 KB, 22993 KB/s, 0 seconds passed

========== Downloading /root/openvino\_models/models/public/squeezenet1.1/squeezenet1.1.caffemodel

... 100%, 4834 KB, 9551 KB/s, 0 seconds passed

========== Replacing text in /root/openvino\_models/models/public/squeezenet1.1/squeezenet1.1.prototxt

**Step 2) Configure Model Optimizer**

* + 환경 변수의 설정과 모델 최적화 설정 (Install 단계에서 이미 실행 완료된 상태)

|  |
| --- |
| cd "${INTEL\_OPENVINO\_DIR}/deployment\_tools/model\_optimizer/install\_prerequisites"  . ./install\_prerequisites.sh caffe |

**Step 3) Convert a model with Model Optimizer**

* + 다운로드한 모델을 IR(중간 표현)로 변환한다. 입력으로서 모델을 지정하고 변환된 파일(IR)의 저장 위치를 지정한다
  + IR의 저장 위치 지정

|  |
| --- |
| target\_precision="FP16"  model\_dir=$("$python\_binary" "$downloader\_dir/info\_dumper.py" --name "$model\_name" |  "$python\_binary" -c 'import sys, json; print(json.load(sys.stdin)[0]["subdirectory"])')  irs\_path="$HOME/openvino\_models/ir"  ir\_dir="${irs\_path}/${model\_dir}/${target\_precision}" |

* + 변환시에는 FP16(16비트 부동소수점 연산) 또는 FP32(32비트 부동소수점 연산) 중 하나를 지정한다. 추론에 사용 정밀도를 지정한다.

|  |
| --- |
| print\_and\_run "$python\_binary" "$downloader\_dir/converter.py" --mo "$mo\_path" --name "$model\_name" -d "$models\_path" -o "$irs\_path" --precisions "$target\_precision" |

Run python3 /home/sky/intel/openvino\_2021.4.582/deployment\_tools/open\_model\_zoo/tools/downloader/converter.py --mo /home/sky/intel/openvino\_2021.4.582/deployment\_tools/model\_optimizer/mo.py --name squeezenet1.1 -d /root/openvino\_models/models -o /root/openvino\_models/ir --precisions FP16

* + 실행이 완료되면 모델 이름+"bin"과 모델 이름+"xml"의 두개의 파일이 생성된다.

**Step 4) Build samples**

* + 데모 프로그램을 포함한 모든 샘플 프로그램이 빌드된다. R3는 사용자 디렉토리 오브젝트 파일과 실행 파일이 생성된다.
  + Build 디렉토리

|  |
| --- |
| samples\_path="${INTEL\_OPENVINO\_DIR}/deployment\_tools/inference\_engine/samples/cpp"  build\_dir="$HOME/inference\_engine\_samples\_build"  binaries\_dir="${build\_dir}/${OS\_PATH}/Release" |

Build files have been written to: /root/inference\_engine\_samples\_build

**Step 5) Run samples**

* + 다음 샘플 프로그램이 실행된다.

|  |
| --- |
| cd "$binaries\_dir"  cp -f "$ROOT\_DIR/${model\_name}.labels" "${ir\_dir}/"  print\_and\_run ./classification\_sample\_async -d "$target" -i "$target\_image\_path" -m "${ir\_dir}/${model\_name}.xml" ${sampleoptions} |

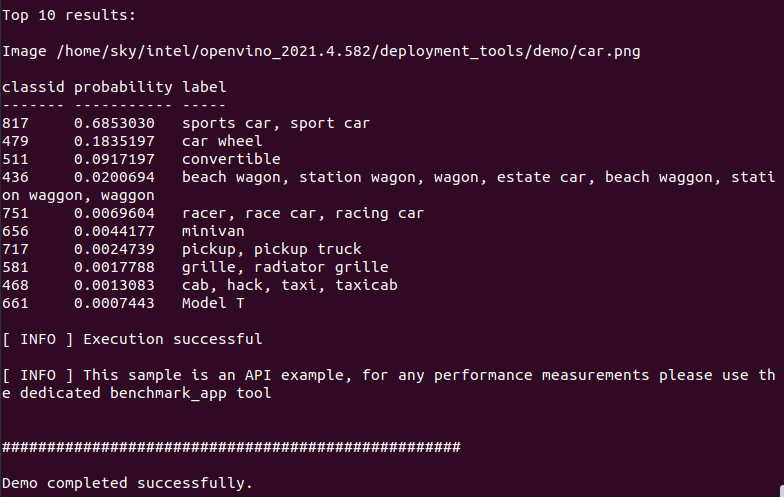
Run ./classification\_sample\_async

-d CPU

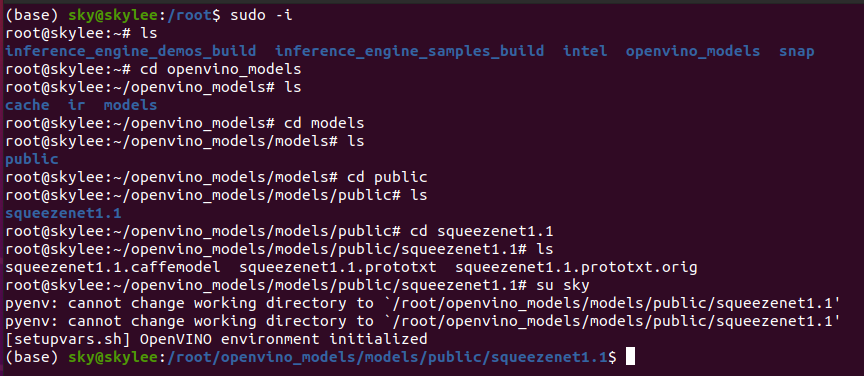
-i /home/sky/intel/openvino\_2021.4.582/deployment\_tools/demo/car.png

-m /root/openvino\_models/ir/public/squeezenet1.1/FP16/squeezenet1.1.xml

* + 실행시 지정하는 인수
    - -d : 추론을 수행하는 하드웨어
    - -i : 입력 이미지 파일
    - -m : 실행 IR의 xml 파일
  + 도중 squeezenet1.1.labels가 복사된다. SqueezeNet 1000 종류의 이미지 (ImageNET) 를 학습하고 있다. 추론 결과는 그림의 번호로 출력되지만 숫자에서는 판정 결과가 올바른지 직관적으로 알기 어렵기 때문에 번호를 구체적으로 인식한 물건이나 동 물의 호칭으로 변환한다. labels는 그 변환에 사용하기 위한 테이블이다. 실행하면 다음과 같이 추론 결과가 출력된다.



※ $HOME/inference\_engine\_samples\_build, HOME/inference\_engine\_sample\_build, openvino\_models 등 [root@user](mailto:root@user):~# 폴더에 설정되어 있음.



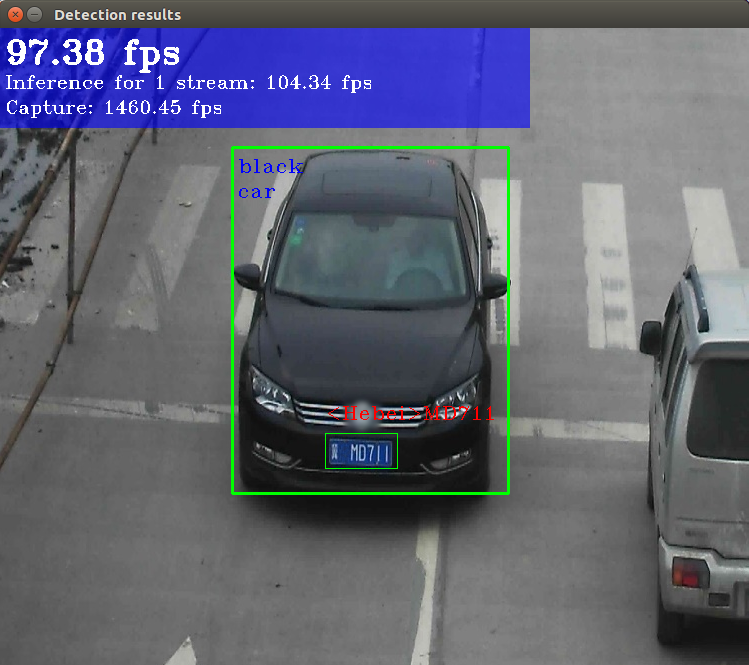
※ HOME 디렉토리 변경

1. Run the **Inference Pipeline verification script**:

content\_copy

./demo\_security\_barrier\_camera.sh

This script downloads three pre-trained model IRs, builds the [Security Barrier Camera Demo](https://docs.openvino.ai/2021.1/omz_demos_security_barrier_camera_demo_README.html) application, and runs it with the downloaded models and the car\_1.bmp image from the demo directory to show an inference pipeline. The verification script uses vehicle recognition in which vehicle attributes build on each other to narrow in on a specific attribute.  
First, an object is identified as a vehicle. This identification is used as input to the next model, which identifies specific vehicle attributes, including the license plate. Finally, the attributes identified as the license plate are used as input to the third model, which recognizes specific characters in the license plate.  
When the verification script completes, you will see an image that displays the resulting frame with detections rendered as bounding boxes, and text:



1. Close the image viewer window to complete the verification script.

To learn about the verification scripts, see the README.txt file in /home/<user>/intel/openvino\_2021/deployment\_tools/demo.

For a description of the Intel Distribution of OpenVINO™ pre-trained object detection and object recognition models, see [Overview of OpenVINO™ Toolkit Pre-Trained Models](https://docs.openvino.ai/2021.1/omz_models_intel_index.html).

You have completed all required installation, configuration and build steps in this guide to use your CPU to work with your trained models. To use other hardware, see;

* [Steps for Intel® Processor Graphics (GPU)](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "additional-GPU-steps)
* [Steps for Intel® Neural Compute Stick 2](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "additional-NCS-steps)
* [Steps for Intel® Vision Accelerator Design with Intel® Movidius™ VPUs](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "install-VPU)

## Steps for Intel® Processor Graphics (GPU)

The steps in this section are required only if you want to enable the toolkit components to use processor graphics (GPU) on your system.

1. Go to the install\_dependencies directory:

cd /opt/intel/openvino\_2021/install\_dependencies/

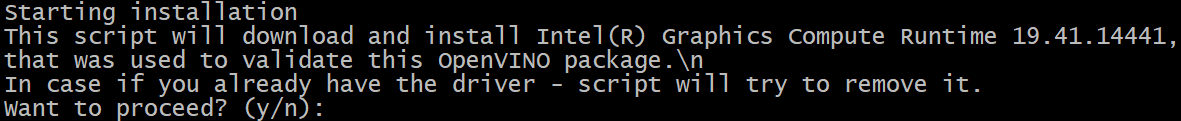
1. Enter the super user mode:

sudo -E su

1. Install the **Intel® Graphics Compute Runtime for OpenCL™** driver components required to use the GPU plugin and write custom layers for Intel® Integrated Graphics. Run the installation script:

./install\_NEO\_OCL\_driver.sh

The drivers are not included in the package and the script downloads them. Make sure you have the internet connection for this step.  
The script compares the driver version on the system to the current version. If the driver version on the system is higher or equal to the current version, the script does not install a new driver. If the version of the driver is lower than the current version, the script uninstalls the lower and installs the current version with your permission:



Higher hardware versions require a higher driver version, namely 20.35 instead of 19.41. If the script fails to uninstall the driver, uninstall it manually. During the script execution, you may see the following command line output:

Add OpenCL user to video group

Ignore this suggestion and continue.

1. **Optional** Install header files to allow compiling a new code. You can find the header files at [Khronos OpenCL™ API Headers](https://github.com/KhronosGroup/OpenCL-Headers.git).

## Run a Sample Application

**IMPORTANT**: This section requires that you have [Run the Verification Scripts to Verify Installation](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "run-the-demos). This script builds the Image Classification sample application and downloads and converts the required Caffe\* Squeezenet model to an IR.

In this section you will run the Image Classification sample application, with the Caffe\* Squeezenet1.1 model on three types of Intel® hardware: CPU, GPU and VPUs.

Image Classification sample application binary file was automatically built and the FP16 model IR files are created when you [Ran the Image Classification Verification Script](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "run-the-image-classification-verification-script).

The Image Classification sample application binary file located in the /home/<user>/inference\_engine\_samples\_build/intel64/Release directory. The Caffe\* Squeezenet model IR files (.bin and .xml) are located in the /home/<user>/openvino\_models/ir/public/squeezenet1.1/FP16/ directory.

=> /root/inference\_engine\_samples\_build/

=> /root/openvino\_models/ir/public/

**NOTE**: If you installed the Intel® Distribution of OpenVINO™ to the non-default install directory, replace /opt/intel with the directory in which you installed the software.

To run the sample application:

1. Set up environment variables:

source /opt/intel/openvino\_2021/bin/setupvars.sh

1. Go to the samples build directory:

cd ~/inference\_engine\_samples\_build/intel64/Release

1. 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.

**NOTE**: Running the sample application on hardware other than CPU requires performing [additional hardware configuration steps](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html" \l "optional-steps).

* + **For CPU**:

./classification\_sample\_async

-i /opt/intel/openvino\_2021/deployment\_tools/demo/car.png

-m ~/openvino\_models/ir/public/squeezenet1.1/FP16/squeezenet1.1.xml -d CPU

* + **For GPU**:

./classification\_sample\_async

-i /opt/intel/openvino\_2021/deployment\_tools/demo/car.png

-m ~/openvino\_models/ir/public/squeezenet1.1/FP16/squeezenet1.1.xml -d GPU

For information on Sample Applications, see the [Inference Engine Samples Overview](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html).

## Hello World Face Detection Tutorial

See the [OpenVINO™ Hello World Face Detection Exercise](https://github.com/intel-iot-devkit/inference-tutorials-generic).

## Troubleshooting

PRC developers might encounter pip installation related issues during OpenVINO™ installation. To resolve the issues, you may use one of the following options at your discretion:

* Add the download source with -i parameter in the pip command. For example:

pip install numpy.py -i https://mirrors.aliyun.com/pypi/simple/

Use the --trusted-host parameter if the URL above is http instead of https.

* Modify or create ~/.pip/pip.conf file to change the default download source with the content below:

[global]

index-url = http://mirrors.aliyun.com/pypi/simple/

[install]

trusted-host = mirrors.aliyun.com

## Additional Resources

* Intel® Distribution of OpenVINO™ toolkit home page: <https://software.intel.com/en-us/openvino-toolkit>
* OpenVINO™ toolkit online documentation: [https://docs.openvinotoolkit.org](https://docs.openvinotoolkit.org/)
* [Model Optimizer Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html).
* [Inference Engine Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html).
* For more information on Sample Applications, see the [Inference Engine Samples Overview](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html).
* For information on a set of pre-trained models, see the [Overview of OpenVINO™ Toolkit Pre-Trained Models](https://docs.openvino.ai/2021.1/omz_models_intel_index.html)
* For information on Inference Engine Tutorials, see the [Inference Tutorials](https://github.com/intel-iot-devkit/inference-tutorials-generic)
* For IoT Libraries and Code Samples see the [Intel® IoT Developer Kit](https://github.com/intel-iot-devkit).

To learn more about converting models, go to:

* [Convert Your Caffe\* Model](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_Caffe.html)
* [Convert Your TensorFlow\* Model](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_TensorFlow.html)
* [Convert Your MXNet\* Model](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_MxNet.html)
* [Convert Your ONNX\* Model](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_ONNX.html)

https://docs.openvino.ai/2021.1/openvino\_docs\_get\_started\_get\_started\_linux.html

**Get Started with OpenVINO™ Toolkit on Linux\***

The OpenVINO™ toolkit optimizes and runs Deep Learning Neural Network models on Intel® hardware. This guide helps you get started with the OpenVINO™ toolkit you installed on a Linux\* operating system.

In this guide, you will:

* Learn the OpenVINO™ inference workflow.
* Run demo scripts that perform the steps for you. These demo scripts illustrate the workflow.
* Run the workflow steps yourself, using detailed instructions with a code sample and demo application.

## OpenVINO™ toolkit Components

The toolkit consists of three primary components:

|  |
| --- |
| * **Inference Engine:** The software libraries that run inference against the Intermediate Representation (optimized model) to produce inference results. * **Model Optimizer:** Optimizes models for Intel® architecture, converting models into a format compatible with the Inference Engine. This format is called an Intermediate Representation (IR). * **Intermediate Representation (IR):** The Model Optimizer output. A model converted to a format that has been optimized for Intel® architecture and is usable by the Inference Engine. |

In addition, demo scripts, code samples and demo applications are provided to help you get up and running with the toolkit:

* **Demo Scripts** - Shell scripts that automatically perform the workflow steps to demonstrate running inference pipelines for different scenarios.
* [**Code Samples**](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) - Small console applications that show you how to:
  + Utilize specific OpenVINO capabilities in an application
  + Perform specific tasks, such as loading a model, running inference, querying specific device capabilities, and more.
* [**Demo Applications**](https://docs.openvino.ai/2021.1/omz_demos_README.html) - Console applications that provide robust application templates to help you implement specific deep learning scenarios. These applications involve increasingly complex processing pipelines that gather analysis data from several models that run inference simultaneously, such as detecting a person in a video stream along with detecting the person's physical attributes, such as age, gender, and emotional state.

## Intel® Distribution of OpenVINO™ toolkit Installation and Deployment Tools Directory Structure

This guide assumes you completed all Intel® Distribution of OpenVINO™ toolkit installation and configuration steps. If you have not yet installed and configured the toolkit, see [Install Intel® Distribution of OpenVINO™ toolkit for Linux\*](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html).

By default, the Intel® Distribution of OpenVINO™ is installed to the following directory, referred to as <INSTALL\_DIR>:

* For root or administrator: /opt/intel/openvino\_<version>/
* For regular users: /home/<USER>/intel/openvino\_<version>/

For simplicity, a symbolic link to the latest installation is also created: /home/<user>/intel/openvino\_2021/

If you installed the Intel® Distribution of OpenVINO™ toolkit to a directory other than the default, replace /opt/intel or /home/<USER>/ with the directory in which you installed the software.

The primary tools for deploying your models and applications are installed to the /opt/intel/openvino\_2021/deployment\_tools directory.

Click for the Intel® Distribution of OpenVINO™ toolkit directory structure

## OpenVINO™ Workflow Overview

The simplified OpenVINO™ workflow is:

1. **Get a trained model** for your inference task. Example inference tasks: pedestrian detection, face detection, vehicle detection, license plate recognition, head pose.
2. **Run the trained model through the Model Optimizer** to convert the model to an Intermediate Representation, which consists of a pair of .xml and .bin files that are used as the input for Inference Engine.
3. **Use the Inference Engine API in the application** to run inference against the Intermediate Representation (optimized model) and output inference results. The application can be an OpenVINO™ sample, demo, or your own application.

## Use the Demo Scripts to Learn the Workflow

The demo scripts in /opt/intel/openvino\_2021/deployment\_tools/demo give you a starting point to learn the OpenVINO workflow. These scripts automatically perform the workflow steps to demonstrate running inference pipelines for different scenarios. The demo steps let you see how to:

* Compile several samples from the source files delivered as part of the OpenVINO toolkit.
* Download trained models.
* Perform pipeline steps and see the output on the console.

**NOTE**: You must have Internet access to run the demo scripts. If your Internet access is through a proxy server, make sure the operating system environment proxy information is configured.

The demo scripts can run inference on any [supported target device](https://software.intel.com/en-us/openvino-toolkit/hardware). Although the default inference device is CPU, you can use the -d parameter to change the inference device. The general command to run the scripts looks as follows:

./<script\_name> -d [CPU, GPU, MYRIAD, HDDL]

Before running the demo applications on Intel® Processor Graphics or on an Intel® Neural Compute Stick 2 device, you must complete the additional configuration steps. For details, see:

* Steps for Intel® Processor Graphics (GPU) section in the [installation instructions](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html)
* Steps for Intel® Neural Compute Stick 2 section in the [installation instructions](https://docs.openvino.ai/2021.1/openvino_docs_install_guides_installing_openvino_linux.html).

The following paragraphs describe each demo script.

### Image Classification Demo Script

The demo\_squeezenet\_download\_convert\_run script illustrates the image classification pipeline.

The script:

1. Downloads a SqueezeNet model.
2. Runs the Model Optimizer to convert the model to the IR.
3. Builds the Image Classification Sample Async application.
4. Runs the compiled sample with the car.png image located in the demo directory.

Click for an example of running the Image Classification demo script

### Inference Pipeline Demo Script

The demo\_security\_barrier\_camera uses vehicle recognition in which vehicle attributes build on each other to narrow in on a specific attribute.

The script:

1. Downloads three pre-trained model IRs.
2. Builds the Security Barrier Camera Demo application.
3. Runs the application with the downloaded models and the car\_1.bmp image from the demo directory to show an inference pipeline.

This application:

1. Identifies an object identified as a vehicle.
2. Uses the vehicle identification as input to the second model, which identifies specific vehicle attributes, including the license plate.
3. Uses the the license plate as input to the third model, which recognizes specific characters in the license plate.

Click for an example of Running the Pipeline demo script

### Benchmark Demo Script

The demo\_benchmark\_app script illustrates how to use the Benchmark Application to estimate deep learning inference performance on supported devices.

The script:

1. Downloads a SqueezeNet model.
2. Runs the Model Optimizer to convert the model to the IR.
3. Builds the Inference Engine Benchmark tool.
4. Runs the tool with the car.png image located in the demo directory.

Click for an example of running the Benchmark demo script

## Use Code Samples and Demo Applications to Learn the Workflow

This section guides you through a simplified workflow for the Intel® Distribution of OpenVINO™ toolkit using code samples and demo applications.

You will perform the following steps:

1. [Use the Model Downloader to download suitable models.](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "download-models)
2. [Convert the models with the Model Optimizer.](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "convert-models-to-intermediate-representation)
3. [Download media files to run inference on.](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "download-media)
4. [Run inference on the Image Classification Code Sample and see the results](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "run-image-classification).
5. [Run inference on the Security Barrier Camera Demo application and see the results](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "run-security-barrier).

Each demo and code sample is a separate application, but they use the same behavior and components. The code samples and demo applications are:

* [Code Samples](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) - Small console applications that show how to utilize specific OpenVINO capabilities within an application and execute specific tasks such as loading a model, running inference, querying specific device capabilities, and more.
* [Demo Applications](https://docs.openvino.ai/2021.1/omz_demos_README.html) - Console applications that provide robust application templates to support developers in implementing specific deep learning scenarios. They may also involve more complex processing pipelines that gather analysis from several models that run inference simultaneously. For example concurrently detecting a person in a video stream and detecting attributes such as age, gender and/or emotions.

Inputs you'll need to specify:

* **A compiled OpenVINO™ code sample or demo application** that runs inferencing against a model that has been run through the Model Optimizer, resuiting in an IR, using the other inputs you provide.
* **One or more models** in the Intermediate Representation format. Each model is trained for a specific task. Examples include pedestrian detection, face detection, vehicle detection, license plate recognition, head pose, and others. Different models are used for different applications. Models can be chained together to provide multiple features; for example vehicle + make/model + license plate recognition.
* **One or more media files**. The media is typically a video file, but can be a still photo.
* **One or more target device** on which you run inference. The target device can be the CPU, GPU, or VPU accelerator.

### Build the Code Samples and Demo Applications

To perform sample inference, run the Image Classification code sample and Security Barrier Camera demo application that were automatically compiled when you ran the Image Classification and Inference Pipeline demo scripts. The binary files are in the ~/inference\_engine\_cpp\_samples\_build/intel64/Release and ~/inference\_engine\_demos\_build/intel64/Release directories, respectively.

To run other sample code or demo applications, build them from the source files delivered as part of the OpenVINO toolkit. To learn how to build these, see the [Inference Engine Code Samples Overview](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html) and [Demo Applications Overview](https://docs.openvino.ai/2021.1/omz_demos_README.html) sections.

### Step 1: Download the Models

You must have a model that is specific for you inference task. Example model types are:

* Classification (AlexNet, GoogleNet, SqueezeNet, others) - Detects one type of element in a frame.
* Object Detection (SSD, YOLO) - Draws bounding boxes around multiple types of objects.
* Custom (Often based on SSD)

Options to find a model suitable for the OpenVINO™ toolkit are:

* Download public and Intel's pre-trained models from the [Open Model Zoo](https://github.com/opencv/open_model_zoo) using [Model Downloader tool](https://docs.openvino.ai/2021.1/omz_tools_downloader_README.html).
* Download from GitHub\*, Caffe\* Zoo, TensorFlow\* Zoo, etc.
* Train your own model.

This guide uses the Model Downloader to get pre-trained models. You can use one of the following options to find a model:

* **List the models available in the downloader**:

content\_copy

cd /opt/intel/openvino\_2021/deployment\_tools/tools/model\_downloader/

content\_copy

python3 info\_dumper.py --print\_all

* **Use grep to list models that have a specific name pattern**:

content\_copy

python3 info\_dumper.py --print\_all | grep <model\_name>

Use the Model Downloader to download the models to a models directory. This guide uses <models\_dir> as the models directory and <models\_name> as the model name:

content\_copy

sudo python3 ./downloader.py --name <model\_name> --output\_dir <models\_dir>

**NOTE:** Always run the downloader with sudo.

Download the following models if you want to run the Image Classification Sample and Security Barrier Camera Demo application:

|  |  |
| --- | --- |
| Model Name | Code Sample or Demo App |
| squeezenet1.1 | Image Classification Sample |
| vehicle-license-plate-detection-barrier-0106 | Security Barrier Camera Demo application |
| vehicle-attributes-recognition-barrier-0039 | Security Barrier Camera Demo application |
| license-plate-recognition-barrier-0001 | Security Barrier Camera Demo application |

Click for an example of downloading the SqueezeNet Caffe\* model

Click for an example of downloading models for the Security Barrier Camera Demo application

### Step 2: Convert the Models to the Intermediate Representation

In this step, your trained models are ready to run through the Model Optimizer to convert them to the Intermediate Representation (IR) format. This is required before using the Inference Engine with the model.

Models in the Intermediate Representation format always include a pair of .xml and .bin files. Make sure you have these files for the Inference Engine to find them.

* **REQUIRED:** model\_name.xml
* **REQUIRED:** model\_name.bin

This guide uses the public SqueezeNet 1.1 Caffe\* model to run the Image Classification Sample. See the example to download a model in the [Download Models](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "download-models) section to learn how to download this model.

The squeezenet1.1 model is downloaded in the Caffe\* format. You must use the Model Optimizer to convert the model to the IR. The vehicle-license-plate-detection-barrier-0106, vehicle-attributes-recognition-barrier-0039, license-plate-recognition-barrier-0001 models are downloaded in the Intermediate Representation format. You don't need to use the Model Optimizer to convert these models.

1. Create an <ir\_dir> directory to contain the model's Intermediate Representation (IR).
2. The Inference Engine can perform inference on different precision formats, such as FP32, FP16, INT8. To prepare an IR with specific precision, run the Model Optimizer with the appropriate --data\_type option.
3. Run the Model Optimizer script:

content\_copy

cd /opt/intel/openvino\_2021/deployment\_tools/model\_optimizer

content\_copy

python3 ./mo.py --input\_model <model\_dir>/<model\_file> --data\_type <model\_precision> --output\_dir <ir\_dir>

The produced IR files are in the <ir\_dir> directory.

Click for an example of converting the SqueezeNet Caffe\* model

### Step 3: Download a Video or a Still Photo as Media

Many sources are available from which you can download video media to use the code samples and demo applications. Possibilities include:

* [https://videos.pexels.com](https://videos.pexels.com/)
* [https://images.google.com](https://images.google.com/)

As an alternative, the Intel® Distribution of OpenVINO™ toolkit includes two sample images that you can use for running code samples and demo applications:

* /opt/intel/openvino\_2021/deployment\_tools/demo/car.png
* /opt/intel/openvino\_2021/deployment\_tools/demo/car\_1.bmp

### Step 4: Run the Image Classification Code Sample

**NOTE**: The Image Classification code sample is automatically compiled when you ran the Image Classification demo script. If you want to compile it manually, see the Build the Sample Applications on Linux section in the [Inference Engine Code Samples Overview](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html).

To run the **Image Classification** code sample with an input image on the IR:

1. Set up the OpenVINO environment variables:

content\_copy

source /opt/intel/openvino\_2021/bin/setupvars.sh

1. Go to the code samples build directory:

content\_copy

cd ~/inference\_engine\_samples\_build/intel64/Release

1. Run the code sample executable, specifying the input media file, the IR of your model, and a target device on which you want to perform inference:

content\_copy

classification\_sample\_async -i <path\_to\_media> -m <path\_to\_model> -d <target\_device>

Click for examples of running the Image Classification code sample on different devices

### Step 5: Run the Security Barrier Camera Demo Application

**NOTE**: The Security Barrier Camera Demo Application is automatically compiled when you ran the Inference Pipeline demo scripts. If you want to build it manually, see the [Demo Applications Overview](https://docs.openvino.ai/2021.1/omz_demos_README.html) section.

To run the **Security Barrier Camera Demo Application** using an input image on the prepared IRs:

1. Set up the OpenVINO environment variables:

content\_copy

source /opt/intel/openvino\_2021/bin/setupvars.sh

1. Go to the demo application build directory:

content\_copy

cd ~/inference\_engine\_demos\_build/intel64/Release

1. Run the demo executable, specifying the input media file, list of model IRs, and a target device on which to perform inference:

content\_copy

./security\_barrier\_camera\_demo -i <path\_to\_media> -m <path\_to\_vehicle-license-plate-detection\_model\_xml> -m\_va <path\_to\_vehicle\_attributes\_model\_xml> -m\_lpr <path\_to\_license\_plate\_recognition\_model\_xml> -d <target\_device>

Click for examples of running the Security Barrier Camera demo application on different devices

## Basic Guidelines for Using Code Samples and Demo Applications

Following are some basic guidelines for executing the OpenVINO™ workflow using the code samples and demo applications:

1. Before using the OpenVINO™ samples, always set up the environment:

content\_copy

source /opt/intel/openvino\_2021/bin/setupvars.sh

1. Have the directory path for the following:

* Code Sample binaries located in ~/inference\_engine\_cpp\_samples\_build/intel64/Release
* Demo Application binaries located in ~/inference\_engine\_demos\_build/intel64/Release
* Media: Video or image. See [Download Media](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "download-media).
* Model: Neural Network topology converted with the Model Optimizer to the IR format (.bin and .xml files). See [Download Models](https://docs.openvino.ai/2021.1/openvino_docs_get_started_get_started_linux.html" \l "download-models) for more information.

## Typical Code Sample and Demo Application Syntax Examples

Template to call sample code or a demo application:

content\_copy

<path\_to\_app> -i <path\_to\_media> -m <path\_to\_model> -d <target\_device>

With the sample information specified, the command might look like this:

content\_copy

./object\_detection\_demo\_ssd\_async -i ~/Videos/catshow.mp4 \

-m ~/ir/fp32/mobilenet-ssd.xml -d CPU

## Advanced Demo Use

Some demo applications let you use multiple models for different purposes. In these cases, the output of the first model is usually used as the input for later models.

For example, an SSD will detect a variety of objects in a frame, then age, gender, head pose, emotion recognition and similar models target the objects classified by the SSD to perform their functions.

In these cases, the use pattern in the last part of the template above is usually:

-m\_<acronym> … -d\_<acronym> …

For head pose:

-m\_hp <headpose model> -d\_hp <headpose hardware target>

**Example of an Entire Command (object\_detection + head pose):**

content\_copy

./object\_detection\_demo\_ssd\_async -i ~/Videos/catshow.mp4 \

-m ~/ir/fp32/mobilenet-ssd.xml -d CPU -m\_hp headpose.xml \

-d\_hp CPU

**Example of an Entire Command (object\_detection + head pose + age-gender):**

content\_copy

./object\_detection\_demo\_ssd\_async -i ~/Videos/catshow.mp4 \

-m ~/r/fp32/mobilenet-ssd.xml -d CPU -m\_hp headpose.xml \

-d\_hp CPU -m\_ag age-gender.xml -d\_ag CPU

You can see all the sample application’s parameters by adding the -h or --help option at the command line.

## Additional Resources

Use these resources to learn more about the OpenVINO™ toolkit:

* [OpenVINO™ Release Notes](https://software.intel.com/en-us/articles/OpenVINO-RelNotes)
* [OpenVINO™ Toolkit Overview](https://docs.openvino.ai/2021.1/index.html" \l "index)
* [Inference Engine Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html)
* [Model Optimizer Developer Guide](https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html)
* [Inference Engine Samples Overview](https://docs.openvino.ai/2021.1/openvino_docs_IE_DG_Samples_Overview.html)
* [Overview of OpenVINO™ Toolkit Pre-Trained Models](https://software.intel.com/en-us/openvino-toolkit/documentation/pretrained-models)
* [OpenVINO™ Hello World Face Detection Exercise](https://github.com/intel-iot-devkit/inference-tutorials-generic)