Intel's Deep Learning Inference Engine (DL IE) is a part of Intel® OpenVINO™ toolkit. You can use it as a computational backend for OpenCV deep learning module.

To use OpenCV with Inference Engine, choose one of the options: * Intel® OpenVINO™ toolkit - includes ready to use build of OpenCV * OpenCV+OpenVINO Windows package (community version) * Build from source * Linux * Microsoft Windows * Raspbian

---

Intel® OpenVINO™ toolkit

- Download and install Intel® OpenVINO™ toolkit.

  Important note: if you want to transfer the installed Inference Engine binaries to another machine w/o running OpenVINO installer there, you need the redistributable files of Intel C++ compiler (use the latest update, 64-bit version), otherwise the Inference Engine or some of its essential plugins will refuse to load and run, which may result in an app crash.

- To perform deep neural networks inference on ARM CPUs, build ARM CPU Plugin from openvino_contrib. ARM CPU Plugin isn't distributed in the package. This way requires building OpenVINO and OpenCV from source. ARM CPU Plugin build instructions.

  - OpenVINO release tag 2021.3
  - OpenVINO contrib release tag 2021.3

OpenCV+OpenVINO Windows package (community version)

This community package uses open source version of Inference Engine from Deep Learning Deployment Toolkit repository (distributed under Apache 2 license).

Hardware requirements: - CPU with support of AVX2 instruction set - *[optional]* Intel® Integrated Graphics - *[optional]* Intel® Neural Compute Stick 2

Software requirements: - Windows* 10, 64-bit - *[recommended]* Microsoft* Visual Studio* 2019 - or MSVS* 2019 redistibutables: vc_redist.x64.exe - Python 3.x (64-bit) to use OpenCV Python bindings - CMake 3.5.1+ (MSVS 2015+) or CMake 3.14+ (MSVS 2019+)

OpenCV 4.5.5

OpenVINO version	GitHub releases	SourceForge
2021.4.2 (release binaries)	.7z (164.5 Mb) / .zip (253.4 Mb)	.7z (164.5 Mb) / .zip (253.4 Mb)
2021.4.2 (debug binaries)	.7z (196.8 Mb)	.7z (196.8 Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- IE Hetero plugin
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

Archive

OpenCV 4.5.4

OpenVINO version	GitHub releases	SourceForge
2021.4.1 (release binaries)	.7z (~164.2 Mb) / .zip (~252.7Mb)	.7z (~164.2Mb) / .zip (~252.7Mb)
2021.4.1 (debug binaries)	.7z (~195.6Mb)	.7z (~195.6Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- IE Hetero plugin
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.5.3

OpenVINO version	GitHub releases	SourceForge
2021.4 (release binaries)	.7z (~166.9Mb) / .zip (~257.4Mb)	.7z (~166.9Mb) / .zip (~257.4Mb)
2021.4 (debug binaries)	.7z (~199.7Mb)	.7z (~199.7Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- IE Hetero plugin
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.5.2

DLDT version	GitHub releases	SourceForge
2021.3 (release binaries)	.7z (~161.1Mb) / .zip (~246Mb)	.7z (~161.1Mb) / .zip (~246Mb)
2021.3 (debug binaries)	.7z (~192.5Mb)	.7z (~192.5Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.5.1

DLDT version	GitHub releases	SourceForge
2021.2 (release binaries)	.7z (~152.9Mb) / .zip (~230Mb)	.7z (~152.9Mb) / .zip (~230Mb)
2021.2 (debug binaries)	.7z (~185.5Mb)	.7z (~185.5Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.5.0

DLDT version	GitHub releases	SourceForge
2021.1 (release binaries)	.7z (~154Mb) / .zip (~232Mb)	.7z (~154Mb) / .zip (~232Mb)
2021.1 (debug binaries)	.7z (~183Mb)	.7z (~183Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.4.0

DLDT version	GitHub releases	SourceForge
2020.4 (release binaries)	.7z (~155Mb) / .zip (~236Mb)	.7z (~155Mb) / .zip (~236Mb)
2020.4 (debug binaries)	.7z (~184Mb)	.7z (~184Mb)
2020.3 (release binaries)	.7z (~152Mb) / .zip (~231Mb)	.7z (~152Mb) / .zip (~231Mb)
2020.3 (debug binaries)	.7z (~179Mb)	.7z (~179Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- Intel® Threading Building Blocks (TBB)
- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

OpenCV 4.3.0

DLDT version	GitHub releases	SourceForge
2020.2 (release binaries)	.7z (~150Mb) / .zip (~220Mb)	.7z (~150Mb) / .zip (~220Mb)
2020.2 (debug binaries)	.7z (~175Mb)	.7z (~175Mb)
2020.1 (release binaries)	.7z (~150Mb) / .zip (~220Mb)	.7z (~150Mb) / .zip (~220Mb)
2020.1 (debug binaries)	.7z (~175Mb)	.7z (~175Mb)

DLDT components included in this release:

- IE MKLDNN plugin (CPU)
- IE clDNN plugin (GPU-OpenCL)
- IE Myriad plugin (VPU)
- Intel® Threading Building Blocks (TBB)

- nGraph

Both package versions include .PDB files for better debugging experience (generated by MSVS 2019). Package with debug binaries is intended for development of C++ applications only (missing Python support, etc).

First steps Steps below require package with release binaries. Steps for debug binaries might differ.

1. Extract package contents
   - Unpack .7z archive using 7-Zip: https://www.7-zip.org/download.html
   - .zip archive can be extracted using Windows 10 builtin tools (check content menu in explorer)
2. Run OpenCV Python samples (can be launched from explorer):

   ```
   <package_root>\src\opencv\samples\python\_run_winpack_demo.cmd
   ```

   Note: OpenCV DNN Python samples are located in `samples\dnn` directory
3. Open console terminal and configure OpenCV environment:

   ```
   CALL <package_root>\build\setup_vars_opencv4.cmd
   opencv_version
   ```

   For better experience consider running MSVS instance from the terminal with configured environment (type `start MyProject.sln`)
4. Run opencv_version:

   ```
   <package_root>\build\bin\opencv_version_win32.exe
   ```

   or execute in the terminal:

   ```
   <package_root>\build\setup_vars_opencv4.cmd opencv_version -v
   ```
5. Run Python interpreter (python.exe should be in `PATH`):

   ```
   <package_root>\build\setup_vars_opencv4.cmd python
   > import cv2 as cv
   > print(cv.getBuildInformation())
   ```

   Note: Python support is missing in the package with "Debug" binaries. Use package with "Release" binaries instead.
6. Build and run C++ sample:
   - open `src\opencv\samples` in explorer (unmaximize window)
   - open `src\opencv\samples\cpp` in explorer (unmaximize window)
   - drag&drop `drawing.cpp` file from `cpp` directory onto `_winpack_build_sample.cmd` script

- or execute in the terminal:

  ```
  <package_root>\src\opencv\samples\_winpack_build_sample.cmd <package_root>\src\ope
  ```

Build OpenCV from source

Build OpenCV with pre-built Inference Engine binaries from OpenVINO toolkit.

Linux Setup environment variables to detect Inference Engine:

```
source /opt/intel/openvino/bin/setupvars.sh
export ngraph_DIR=/opt/intel/openvino/deployment_tools/ngraph/cmake/
```

Build OpenCV with extra flags:

```
cmake \
  -DWITH_INF_ENGINE=ON \
  -DENABLE_CXX11=ON \
  ...
```

Microsoft Windows Setup environment variables to detect Inference Engine:

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

Build OpenCV with extra flags:

```
cmake ^
  -DWITH_INF_ENGINE=ON ^
  -DENABLE_CXX11=ON ^
  ...
```

Raspbian Buster Use Docker to cross-compile OpenCV for Raspberry Pi. Check that `uname -m` detects `armv7l` CPU architecture (starts from Raspberry Pi 2 model B).

1. Create a folder named `debian_armhf` with a file `Dockerfile` with the following content:

```
FROM debian:buster

USER root

RUN dpkg --add-architecture armhf && \
    apt-get update && \
    apt-get install -y --no-install-recommends \
    crossbuild-essential-armhf \
    cmake \
    pkg-config \
```

```
    wget \
    xz-utils \
    libgtk2.0-dev:armhf \
    libpython-dev:armhf \
    libpython3-dev:armhf \
    python-numpy \
    python3-numpy \
    libgstreamer1.0-dev:armhf \
    libgstreamer-plugins-base1.0-dev:armhf

# Install Inference Engine
RUN wget --no-check-certificate https://download.01.org/opencv/2019/openvinotoolkit/R2/l_ope
    tar -xf l_openvino_toolkit_runtime_raspbian_p_2019.2.242.tgz
```

2. Build a Docker image

```
docker image build -t debian_armhf debian_armhf
```

3. Run Docker container mounting source code folder from host.

```
docker run -it -v /absolute/path/to/opencv:/opencv debian_armhf /bin/bash
```

4. Build

```
cd opencv && mkdir opencv_build && mkdir opencv_install && cd opencv_build
cmake -DCMAKE_BUILD_TYPE=Release \
      -DCMAKE_INSTALL_PREFIX="../opencv_install" \
      -DOPENCV_CONFIG_INSTALL_PATH="cmake" \
      -DCMAKE_TOOLCHAIN_FILE="../platforms/linux/arm-gnueabi.toolchain.cmake" \
      -DWITH_IPP=OFF \
      -DBUILD_TESTS=OFF \
      -DBUILD_PERF_TESTS=OFF \
      -DOPENCV_ENABLE_PKG_CONFIG=ON \
      -DPKG_CONFIG_EXECUTABLE="/usr/bin/arm-linux-gnueabihf-pkg-config" \
      -DPYTHON2_INCLUDE_PATH="/usr/include/python2.7" \
      -DPYTHON2_NUMPY_INCLUDE_DIRS="/usr/local/lib/python2.7/dist-packages/numpy/core/incluc
      -DPYTHON3_INCLUDE_PATH="/usr/include/python3.7" \
      -DPYTHON3_NUMPY_INCLUDE_DIRS="/usr/local/lib/python3.7/dist-packages/numpy/core/incluc
      -DPYTHON3_CVPY_SUFFIX=".cpython-37m-arm-linux-gnueabihf.so" \
      -DENABLE_NEON=ON \
      -DCPU_BASELINE="NEON" \
      -DWITH_INF_ENGINE=ON \
      -DINF_ENGINE_LIB_DIRS="/l_openvino_toolkit_runtime_raspbian_p_2019.2.242/inference_eng
      -DINF_ENGINE_INCLUDE_DIRS="/l_openvino_toolkit_runtime_raspbian_p_2019.2.242/inference
      -DCMAKE_FIND_ROOT_PATH="/l_openvino_toolkit_runtime_raspbian_p_2019.2.242" \
      -DENABLE_CXX11=ON ..
make -j4 && make install
```

5. Copy `opencv_install` to the board. Follow http://docs.openvinotoolkit.org/latest/_docs_install_guides_

to install OpenVINO distribution for Raspberry Pi. Then type the following commands to specify new location of OpenCV:

```
export PYTHONPATH=/path/to/opencv_install/lib/python2.7/dist-packages/:$PYTHONPATH
export PYTHONPATH=/path/to/opencv_install/lib/python3.7/dist-packages/:$PYTHONPATH
export LD_LIBRARY_PATH=/path/to/opencv_install/lib/:$LD_LIBRARY_PATH
```

Usage

- Enable Intel's Inference Engine backend right after `cv::dnn::readNet` invocation:

  ```
  net.setPreferableBackend(DNN_BACKEND_INFERENCE_ENGINE);
  ```

  > **NOTE**: Starts from OpenCV 3.4.2 (OpenVINO 2018.R2) this backend is used by default if OpenCV is built with the Inference Engine support. To switch to origin implementation, use `DNN_BACKEND_OPENCV`. Also, the Inference engine backend is the only available option when the loaded model is represented in OpenVINO™ Model Optimizer format (`.bin` and `.xml`).

- Then, optionally you can also set the device to use for the inference (by default it will use CPU):

  ```
  net.setPreferableTarget(DNN_TARGET_OPENCL);
         // the possible options are
         // DNN_TARGET_CPU,
         // DNN_TARGET_OPENCL,
         // DNN_TARGET_OPENCL_FP16
         //   (fall back to OPENCL if the hardware does not support FP16),
         // DNN_TARGET_MYRIAD
  ```

- You may also import pre-trained models from Open Model Zoo passing paths to `.bin` and `.xml` files to `cv::dnn::readNet` function.

- Other names and brands may be claimed as the property of others.