Compiling and running OpenCV with TIM-VX backend

We are thrilled to introduce you the TIM-VX backend integrated in OpenCV DNN, which allows OpenCV DNN runs quantized DL models in neural processing units (NPU) on edge devices, such as Khadas VIM3 etc. It achives up to 2X faster than ARM CPU backend for running face detection and recognition model from OpenCV Zoo. More details can be found in OpenCV Zoo Benchmarks.

TIM-VX is provided with x86_64 simulator. So you can try OpenCV with TIM-VX backend on your x86_64 machine following steps below, or if you happen to have a physical board equiped with the A311D chip (like the Khadas VIM3 mentioned above). In this guide, we provide two ways compiling OpenCV with TIM-VX backend: - (Recommanded) Compile OpenCV together with TIM-VX. - Compile OpenCV with TIM-VX library installed previously.

Requirements

Before compiling, make sure you have the following dependencies installed:

- git
- cmake (3.14 or above)
- (optional) python3 (3.6 or above), python3-numpy

Install packages on Ubuntu:

```
sudo apt install git cmake

# python3 is needed if you want to run examples from OpenCV Zoo
sudo apt install python3 python3-dev python3-numpy
```

Extra Requirements for Khadas VIM3

For Khadas VIM3 users, you will need to download VIVANTE SDK at https://github.com/VeriSilicon/TIM-VX/releases/download/v1.1.34.fix/aarch64_A311D_6.4.8.tgz, extract and replace `galcore.ko` in the system:

```
# Download and extract VIVANTE SDK
cd /opt
wget https://github.com/VeriSilicon/TIM-VX/releases/download/v1.1.34.fix/aarch64_A311D_6.4.8
echo 'da530e28f73fd8b143330b6d1b97a1d8 /opt/aarch64_A311D_6.4.8.tgz' | md5sum -c
# If get '/opt/aarch64_A311D_6.4.8.tgz: OK', then proceed; otherwise, re-download the packag
tar xvf aarch64_A311D_6.4.8.tgz

# Set environment variable VIVANTE_SDK_DIR for compiling and running examples.
export VIVANTE_SDK_DIR=/opt/aarch64_A311D_6.4.8

# Replace galcore.ko. Choose either one of the following option to proceed.
# 1. Replace everytime the system is rebooted
sudo rmmod galcore
sudo insmod /path/to/aarch64_A311D_6.4.8/galcore.ko
# 2. Replace once for all
sudo find /usr/lib -name galcore.ko
mv /path/to/aarch64_A311D_6.4.8/galcore.ko <found_galcore_path_in_usr_lib>
sudo reboot # reboot to take effect
```

Compile OpenCV together with TIM-VX

To compile OpenCV together with TIM-VX, all you need to do is adding a CMake option `-DWITH_TIMVX=ON`:

```
cd /opt

# Get source code
git clone https://github.com/opencv/opencv

# Configure
# Turn on OpenCV's Python interface: -D BUILD_opencv_python3=ON
cmake -B opencv-build \
      -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=opencv-install \
      -D WITH_TIMVX=ON opencv
# NOTE: Make sure you see `TIMVX: YES` in the CMake output.
#       Copy the value of `VIVANTE SDK path` which is needed below.

# Compile
cmake --build opencv-build --target install -j 8
```

`LD_LIBRARY_PATH` and `VIVANTE_SDK_DIR` is needed to run OpenCV with TIM-VX backend:

```
export LD_LIBRARY_PATH=/path/to/<VIVANTE_SDK_path>/lib:$LD_LIBRARY_PATH
export VIVANTE_SDK_DIR=/path/to/<VIVANTE_SDK_path>
# For x86_64 Linux, VIVANTE SDK prebuilt for x86_64 comes along with TIM-VX source code. A 
# For Khadas VIM3, a typical <VIVANTE_SDK_path> is /opt/aarch64_A311D_6.4.8 if followed sec
```

Compile OpenCV with TIM-VX library installed previously

You can try latest or a different version of TIM-VX with the following steps, but **note that latest TIM-VX may not work with OpenCV**.

Step 1: Compile and install TIM-VX

```
cd /opt

# Get source code
```

```
git clone https://github.com/VeriSilicon/TIM-VX.git

# Configure
# x86_64 Linux
cmake -B timvx-build \
      -D BUILD_SHARED_LIBS=OFF TIM-VX
# Khadas VIM3
cmake -B timvx-build \
      -D EXTERNAL_VIV_SDK=/opt/aarch64_A311D_6.4.8 \
      -D BUILD_SHARED_LIBS=OFF TIM-VX

# Compile
cmake --build timvx-build --target install -j 8
```

Step2: Compile OpenCV with TimVX

```
cd /opt

# Get source code
git clone https://github.com/opencv/opencv

# Configure
# For Khadas VIM3, turn off building with Eigen if compilation fails at Eigen: -D WITH_EIGE
# For Khadas VIM3, turn off building with OpenCL to pass unit tests: -D WITH_OPENCL=OFF
# Turn on OpenCV's Python interface: -D BUILD_opencv_python3=ON
cmake -B opencv-build \
      -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=./install \
      -D WITH_TIMVX=ON \
      -D TIMVX_INSTALL_DIR=/opt/timvx-build/install \
      -D VIVANTE_SDK_DIR=/opt/aarch64_A311D_6.4.8 opencv
# NOTE: Make sure you see `TIMVX: YES` in the CMake output.
#       Copy the value of `VIVANTE SDK path` which is needed below.

# Compile
cmake --build opencv-build --target install -j 8
```

`LD_LIBRARY_PATH` and `VIVANTE_SDK_DIR` is needed as well to run OpenCV with TIM-VX backend:

```
export LD_LIBRARY_PATH=/path/to/<VIVANTE_SDK_path>/lib:$LD_LIBRARY_PATH
export VIVANTE_SDK_DIR=/path/to/<VIVANTE_SDK_path>
```

Performance Test on OpenCV Zoo

We have done performance tests on OpenCV Zoo: https://github.com/opencv/opencv_zoo#models–benchmark-results. You can reproduce the results as follows:

```
# Install git-lfs from https://git-lfs.github.com/
git clone https://github.com/opencv/opencv_zoo && cd opencv_zoo
git lfs install
git lfs pull

cd benchmark

# edit the config of the model you want to run, switch backend to TIM-VX and target to npu
vim config/face_detection_yunet.yaml
# in vim, press i to edit:
backend: "default" -> backend: "timvx"
target: "cpu" -> target: "npu"
# in vim, press esc and type :wq to save & exit

# run benchmarking
PYTHONPATH=.. python3 benchmark.py --cfg ./config/face_detection_yunet.yaml
```

Ensure you have set `LD_LIBRARY_PATH` and `VIVANTE_SDK_DIR` properly as noted above.

Example

Python Example

We have prepared some quantized models and demo code at OpenCV Zoo: https://github.com/opencv/opencv_zoo. Please set the right backend and quantized model path for the demo argument.

C++ Example

The example model is Resnet Int8 Model. Some example image can be found at this repo, and the label information can be found at this repo.

```
#include <iostream>
#include <vector>
#include <opencv2/dnn.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include<algorithm>

using namespace std;
using namespace cv;
using namespace cv::dnn;

int main()
{
    // load input
    Mat image = imread("PATH_TO_image");
    Scalar meanValue(0.485, 0.456, 0.406);
    Scalar stdValue(0.229, 0.224, 0.225);

    Mat blob = blobFromImage(image, 1.0/255.0, Size(224, 224), meanValue, true);
    blob /= stdValue;
    Net net = readNetFromONNX("PATH_TO_MODEL/resnet50-v1-12-int8.onnx");

    // set TimVX backend
    net.setPreferableBackend(DNN_BACKEND_TIMVX);
    net.setPreferableTarget(DNN_TARGET_NPU);

    std::vector<Mat> out;
    net.setInput(blob);
    net.forward(out);

    double min=0, max=0;
    Point minLoc, maxLoc;
    minMaxLoc(out[0], &min, &max, &minLoc, &maxLoc);
    cout<<"class num = "<<maxLoc.x<<std::endl;
}
```

FAQ

Q1: I cannot get OpenCV's Python interface compiled.

Please make sure you have installed python3, python3-dev and numpy:

```
sudo apt install python3 python3-dev python3-numpy
```

Remove cmake cache:

```
# lets say $opencv_build is the build root for OpenCV
rm $opencv_build/CMakeCache.txt
```

Add the following options to your cmake command:

```
-D PYTHON3_INCLUDE_DIR=/path/to/python/include # A typical one is /usr/include/python3.8, wh
-D PYTHON3_EXECUTABLE=/path/to/python_binary   # A typical one is /usr/bin/python3.8, where
-D PYTHON3_LIBRARY=/path/to/libpython3.x.so    # A typical one is /usr/lib/aarch64-linux-gnu
```

Then run `cmake --build $opencv_build --target install -j 8` to rebuild OpenCV. The built Python interface `cv2.x.so` should be located under `$opencv_build/install/lib/python3.8/site-packages/cv2/python-3.8/`.

Q2: How do I install the built Python interface of OpenCV?

By default, you can locate the OpenCV's Python interface `cv2.x.so` under `$opencv_build/install/lib/python3.8/site-packages/cv2/python-3.8` once compilation is finished.

You may create a symbolic link to `cv2.x.so` under python3's package directory.
For example,

```
# '/usr/lib/python3/dist-packages/cv2.so': you will need to find your own location to replac
ln -sf ${opencv}/build/install/lib/python3.8/site-packages/cv2/python-3.8/cv2.cpython-38-aar
```

OpenCV should be ready for Python:

```
python3 -c "import cv2; print(cv2.__version__)"
```

Q3: Why do I have to set `VIVANTE_SDK_DIR` and `LD_LIBRARY_PATH` for TIM-VX backend?

`VIVANTE_SDK_DIR` is set for TIM-VX to find `$VIVANTE_SDK_DIR/include/CL/cl_viv_vx_ext.h`.
A workaround to avoid setting `VIVANTE_SDK_DIR` is put `cl_viv_vx_ext.h` along with executable binary files.

`LD_LIBRARY_PATH` is set for TIM-VX to find OpenVX and other dependencies. TIM-VX is built on top of OpenVX and other shared libraries. Integrating TIM-VX into OpenCV DNN also brings the dependencies for OpenCV DNN.