# Embedded System Design Final

第一組 311552067 劉承熙 312551129 蔡政邦

## Cross Compile Tensorflow Lite and Opencv

### cross compile tensorflow lite

```
git clone https://github.com/tensorflow/tensorflow.git tensorflow_src
```

- change tensorflow lite version

  ```
  cd tensorflow_src
  git checkout tags/v2.5.0

  cd ..
  mkdir tensorflow_build
  cd tensorflow_build

  ARMCC_PREFIX=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linu
  x-gnueabihf/bin/arm-linux-gnueabihf-
  ARMCC_FLAGS="-DWITH_PARALLEL_PF=OFF -funsafe-math-optimizations"
  cmake -DCMAKE_C_COMPILER=${ARMCC_PREFIX}gcc \
    -DCMAKE_CXX_COMPILER=${ARMCC_PREFIX}g++ \
    -DCMAKE_C_FLAGS="${ARMCC_FLAGS}" \
    -DCMAKE_CXX_FLAGS="${ARMCC_FLAGS}" \
    -DCMAKE_VERBOSE_MAKEFILE:BOOL=ON \
    -DCMAKE_SYSTEM_NAME=Linux \
    -DCMAKE_SYSTEM_PROCESSOR=arm \
    -DTFLITE_ENABLE_XNNPACK=OFF \
    /home/user/tensorflow_src/tensorflow/lite
  ```

  CMake 需要升級

  ```
  cmake --build . -j
  ```

- 生成 static library .a 檔，並利用自己的方法將 .a 檔撈出

### Cross Compile Opencv

- git clone and change opencv version

  ```
  git clone https://github.com/opencv/opencv.git
  cd opencv
  git checkout 3.4.7

  cd ~/opencv/platforms/linux
  mkdir -p build_hardfp
  cd build_hardfp
  ```

- add code into opencv/CMakeLists.txt (search “ocv_include_directories”)

```
ocv_include_directories(./3rdparty/zlib)
```

- cross compile

```
ARMCC_PREFIX=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linu
x-gnueabihf/bin/arm-linux-gnueabihf-
ARMCC_FLAGS="-DWITH_PARALLEL_PF=OFF -funsafe-math-optimizations"
cmake -DCMAKE_C_COMPILER=${ARMCC_PREFIX}gcc \
  -DCMAKE_CXX_COMPILER=${ARMCC_PREFIX}g++ \
  -DCMAKE_C_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_CXX_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_VERBOSE_MAKEFILE:BOOL=ON \
  -DCMAKE_SYSTEM_NAME=Linux \
  -DCMAKE_SYSTEM_PROCESSOR=arm \
  -DBUILD_SHARED_LIBS=OFF \
  /home/user/opencv

make -j
sudo make install (including lib, include, you need)
```

- show linking options, including the paths to the required library files and the library names.

```
pkg-config --cflags --libs opencv.pc

-lopencv_dnn -lopencv_highgui -lopencv_ml -lopencv_objdetect -lop
encv_shape -lopencv_stitching \
-lopencv_superres -lopencv_videostab -lopencv_calib3d -lopencv_vi
deoio -lopencv_imgcodecs \
-lopencv_features2d -lopencv_video -lopencv_photo -lopencv_imgpro
c -lopencv_flann -lopencv_core \
```

## Build Yolov5

```
git clone https://github.com/muhammedakyuzlu/tensorflow_lite_libs_cpp.g
it
```

後續編譯時需要裡面 /include 底下的 .h 檔

```
git clone https://github.com/muhammedakyuzlu/yolov5-tflite-cpp.git
```

修改這個 sample code 編譯之後即完成，編譯時需 linked 以上 cross compile 出來的 .a 檔和所有的 .h 檔才會正確 (library and include)

- cross compile cpp code 的 shell script
  - $1 : main.cpp
  - $2 : yolo.cpp

- $3 : output 執行檔

```
arm-linux-gnueabihf-g++ $1 $2 -o $3 \
-I /opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabih
f/include/ \
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-l
inux-gnueabihf/arm-linux-gnueabihf/libc/lib/ \
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-l
inux-gnueabihf/qt5.5/rootfs_imx6q_V3_qt5.5_env/lib/ \
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-l
inux-gnueabihf/qt5.5/rootfs_imx6q_V3_qt5.5_env/qt5.5_env/lib/ \
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-l
inux-gnueabihf/qt5.5/rootfs_imx6q_V3_qt5.5_env/usr/lib/ \
-lpthread \
-I /home/user/tensorflow_lite_libs_cpp/include \
/home/user/tensorflow_build/libtensorflow-lite.a \
-std=c++11 \
-ldl \
-I/usr/local/include/opencv \
-I/usr/local/include/  \
-L/home/user/opencv/platforms/linux/build_hardfp/lib \
-lopencv_dnn -lopencv_highgui -lopencv_ml -lopencv_objdetect -lop
encv_shape -lopencv_stitching -lopencv_superres -lopencv_videosta
b -lopencv_calib3d -lopencv_videoio -lopencv_imgcodecs -lopencv_f
eatures2d -lopencv_video -lopencv_photo -lopencv_imgproc -lopencv
_flann -lopencv_core \
/home/user/app/liblibtiff.a \
/home/user/app/libabsl_time_zone.a \
/home/user/app/libabsl_strings_internal.a \
/home/user/app/libabsl_throw_delegate.a \
/home/user/app/libabsl_flags_config.a \
/home/user/app/libabsl_bad_optional_access.a \
/home/user/app/libabsl_symbolize.a \
/home/user/app/libtegra_hal.a \
/home/user/app/libabsl_base.a \
/home/user/app/libabsl_debugging_internal.a \
/home/user/app/libabsl_raw_logging_internal.a \
/home/user/app/libittnotify.a \
/home/user/app/libabsl_flags_internal.a \
/home/user/app/libabsl_flags_registry.a \
/home/user/app/libabsl_stacktrace.a \
/home/user/app/libabsl_graphcycles_internal.a \
/home/user/app/libabsl_city.a \
/home/user/app/libabsl_civil_time.a \
/home/user/app/libzlib.a \
/home/user/app/libabsl_str_format_internal.a \
/home/user/app/libabsl_cord.a \
/home/user/app/libabsl_demangle_internal.a \
```

```
/home/user/app/libabsl_status.a \
/home/user/app/libabsl_hash.a \
/home/user/app/liblibwebp.a \
/home/user/app/libabsl_malloc_internal.a \
/home/user/app/liblibjasper.a \
/home/user/app/libabsl_spinlock_wait.a \
/home/user/app/libruy.a \
/home/user/app/liblibpng.a \
/home/user/app/libabsl_log_severity.a \
/home/user/app/libabsl_time.a \
/home/user/app/libfft2d_fftsg.a \
/home/user/app/libtensorflow-lite.a \
/home/user/app/libfarmhash.a \
/home/user/app/liblibjpeg-turbo.a \
/home/user/app/libabsl_synchronization.a \
/home/user/app/liblibprotobuf.a \
/home/user/app/libabsl_bad_variant_access.a \
/home/user/app/libabsl_dynamic_annotations.a \
/home/user/app/libabsl_flags_program_name.a \
/home/user/app/libabsl_strings.a \
/home/user/app/libflatbuffers.a \
/home/user/app/libquirc.a \
/home/user/app/libabsl_flags.a \
/home/user/app/libabsl_int128.a \
/home/user/app/libfft2d_fftsg2d.a \
/home/user/app/libabsl_flags_marshalling.a \
/home/user/opencv/platforms/linux/build_hardfp/lib/libopencv_ts.a
```

## Execute

帶入 train 好的 model 和 names file 即可完成

demo1

```
./test ./models/yolov5n-int8.tflite ./models/coco.names input.jpg out.j
pg
```

real time

```
./test ./models/weight/yolov5n-int8.tflite ./models/coco.names
```