ML application development on i.MX8M Plus

Overview

This is a guide about how to build an application with TensorFlow Lite C++ API on i.MX8MP, taking BSP 5.10.72 as an example.

To reduce size of rootfs on target, some dependencies of building the ML application are not included in the target rootfs. To generate the dependencies, you need to follow *i.MX Yocto Project User's Guide* to build the TensorFlow Lite libraries, alternatively build the full image. Then copy the dependencies to the target. Finnally, You can build the application on the target.

You can also build the C++ ML application with Yocto SDK. See details in *i.MX Machine Learning User's guide*.

Prepare Dependencies

1. Use pre-compiled NXP image for i.MX8M Plus EVK

1.1 Prepare iMX8MP EVK full image

Download the L5.10.72 pre-compiled image from NXP website: Embedded Linux for i.MX Applications Processors | NXP Semiconductors. Flash the image to eMMC/SD card and boot up the target.

1.2 Build TensorFlow Lite in Yocto project

In Yocto project, the TensorFlow Lite dependent source header files can be extracted together, and the dependent libraries with corresponding version can be downloaded automatically. I suggest to build the TensorFlow Lite in Yocto project.

1. On the Linux host machine, run the command below to install the packages that are needed to build the image:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo \
  gcc-multilib build-essential chrpath socat cpio python python3 python3-pip \
  python3-pexpect xz-utils debianutils iputils-ping python3-git \
  python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm rsync curl
```

2. If the repo tool is not installed, install it:

```
$ mkdir ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

3. If Git is not configured, configure Git:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --list
```

4. Create the Yocto build environment:

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b \
  imx-linux-hardknott -m imx-5.10.72-2.2.0.xml
$ repo sync
```

5. After repo finishes, create the build folder:

```
$ DISTRO=fsl-imx-xwayland MACHINE=imx8mpevk source \
  imx-setup-release.sh -b imx8mp-xwayland
```

6. Build the TensorFlow Lite for the i.MX8M Plus:

```
$ bitbake tensorflow-lite
```

Tip: *You can also build the full image by 'bitbake imx-image-full'. Then the TensorFlow Lite will be built automatically.*

Build the Application On Target

2. Copy the dependent files to the target

1. Copy source code of label image to the target:

```
$ cd ./ imx-yocto-bsp/ tmp/work/cortexa53-crypto-poky-linux/tensorflow-lite/2.6.0-r0/
$ scp -r  ./git/ tensorflow/lite/ examples/ label_image \
   ./git/tensorflow/lite/tools/evaluation/utils.cc  \
   ./git/tensorflow/lite/tools/delegates/delegate_provider.cc \
   ./git/tensorflow/lite/tools/delegates/external_delegate_provider.cc \
   root@<target IP address>:/home/root/
```

Note: *You need to change the including path of the headers in source code (bitmap_helpers.cc, bitmap_helpers.h, bitmap_helpers_impl.h and label_image.cc) following the attached patch.*

2. Copy TensorFlow Lite and flatbuffers header files to the target:

```
$ scp -r ./ image/usr/include/tensorflow/ \
   build/flatbuffers/include/flatbuffers/ \
   root@<target IP address>:/usr/include/
```

3. Since the absl header files are not extracted together. You need to create sub-directories of absl directory as-is:
   On target:

```
root@imx8mpevk:~# mkdir -p /usr/include/absl/memory/
root@imx8mpevk:~# mkdir -p /usr/include/absl/base/
root@imx8mpevk:~# mkdir -p /usr/include/absl/meta
```

   Copy the header files to the target:

   On host PC:

```
$ scp -r  ./build/abseil-cpp/absl/memory/*.h root@<target IP address>:/usr/include/absl/memory/
$ scp -r  ./build/abseil-cpp/absl/base/*.h root@<target IP address>:/usr/include/absl/base/
$ scp -r  ./build/abseil-cpp/absl/meta/*.h root@<target IP address>:/usr/include/absl/meta /
```

4. There is only realname of libtensorflow-lite on the target. You need to rename it to another linkname:
   On target:

```
root@imx8mpevk:~# cp /usr/lib/libtensorflow-lite.so.2.6.0 /usr/lib/libtensorflow-lite.so
```

3. Build the application on the target

Build the label image application on the target:

```
root@imx8mpevk:~# gcc -o label_image label_image.cc bitmap_helpers.cc utils.cc delegate_provider.cc
external_delegate_provider.cc -ltensorflow-lite -lstdc++ -lpthread -lm -ldl -lrt -I ./
```

Run the label image on the target:

```
root@imx8mpevk:~# ./label_image -m \
/usr/bin/tensorflow-lite-2.6.0/examples/mobilenet_v1_1.0_224_quant.tflite \
-l /usr/bin/tensorflow-lite-2.6.0/examples/labels.txt \
-i /usr/bin/tensorflow-lite-2.6.0/examples/grace_hopper.bmp \
```

```
--external_delegate_path=/usr/lib/libvx_delegate.so
```

Here is the log:

```
INFO: Loaded model /usr/bin/tensorflow-lite-2.6.0/examples/mobilenet_v1_1.0_224_quant.tflite
INFO: resolved reporter
Vx delegate: allowed_builtin_code set to 0.
Vx delegate: error_during_init set to 0.
Vx delegate: error_during_prepare set to 0.
Vx delegate: error_during_invoke set to 0.
EXTERNAL delegate created.
INFO: Applied EXTERNAL delegate.
W [HandleLayoutInfer:257]Op 18: default layout inference pass.
INFO: invoked
INFO: average time: 2.589 ms
INFO: 0.768627: 653 military uniform
INFO: 0.105882: 907 Windsor tie
INFO: 0.0196078: 458 bow tie
INFO: 0.0117647: 466 bulletproof vest
INFO: 0.00784314: 835 suit
```

> ***Note:*** *If you used OpenCV in your application, you also need to copy the OpenCV header files to the target and rename the OpenCV libraries to other linknames.*

Questions?

Feel free to reach out to us at https://community.nxp.com/.

Appendix

Revision History

Revision	Date	Description	Author
1	5/12/2022	Initial Release	Devin Jiao

Legal