RUNNING MNIST FROM CAMERA

Tensai Flow 1.0 Beta Release

Oct 26, 2021

ETA COMPUTE
340 N Westlake Blvd. Suite 115, Westlake Village, California 91362

Table of Contents

1. Generating c model files

Refer below commands to generate a c model of the tflite

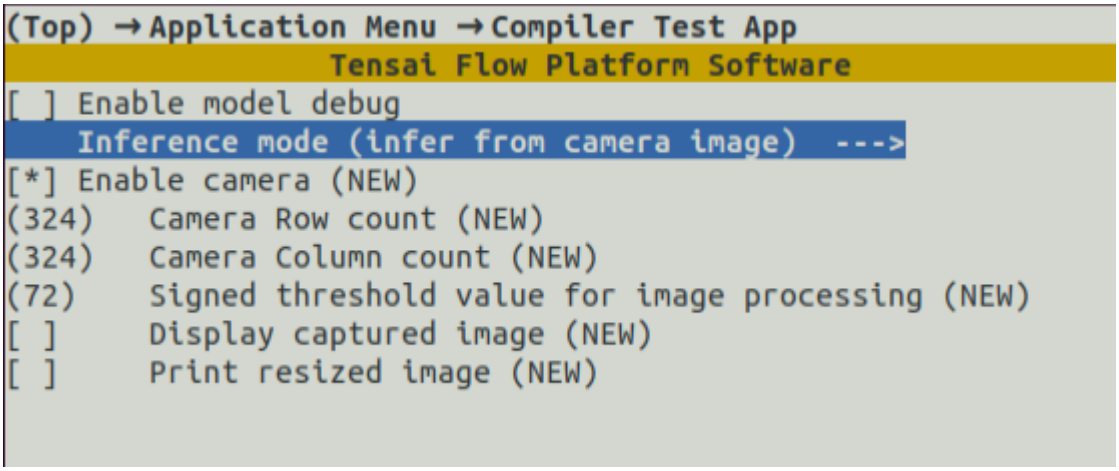
```
/* generate infer.c for MNIST model */
<Tools/Tensai_compiler/bin> ./tensaicc \
--model=../model_zoo/mnist_model/model.tflite \
--app_dir=../../../Applications/compiler-test-app/ \
--target_config=core_config.ini --nodata_io

<Tools/Tensai_compiler/bin> cp ../model_zoo/mnist_model/data_io.c
../../../Applications/compiler-test-app/src/

<Tools/Tensai_compiler/bin> cp ../model_zoo/mnist_model/input_data.h
../../../Applications/compiler-test-app/include/
```

2. Build and Run

Run below command to configure the application to use camera for inferencing

```
/* menuconfig */
<TENSAI_FLOW/Applications/compiler-test-app/build># make menuconfig
```

Application specific menuconfig
Select “Application Menu => Compiler Test App” to configure application specific configs



Enable below entries in the menuconfig

- Inference mode -> infer from camera image
- Enable camera

Control parameters

- Signed threshold value for image processing
  Room lighting may impact the accuracy of the model, the default value is kept to work in most of the cases, but in case if the accuracy is going wrong, adjusting this param will help improve the accuracy.

The thresholding value is to be based on the range of [-128, 127] as opposed to [0, 255] because the image is expected to be shifted by the TFLite model.

Optional configs

- Display captured image: Choose this option for displaying captured image using parser script provided in the SDK in below path

  Thirdparty/SDK/<Release SDK name>/fcp/platform/tahiti/tools

```
python parser.py -c <serial_device> -b 115200 -f <filename>

e.g python parser.py -c /dev/ttyUSB0 -b 115200 -f test
```

  Note: Enabling display of images adds significant time overhead due to the UART transfer of uncompressed images.
- Print resize image: Choose this option for printing resized and a thresholded image array over debug UART.
  Note: Inference will be done on this image array.

2.1 Building firmware binary image

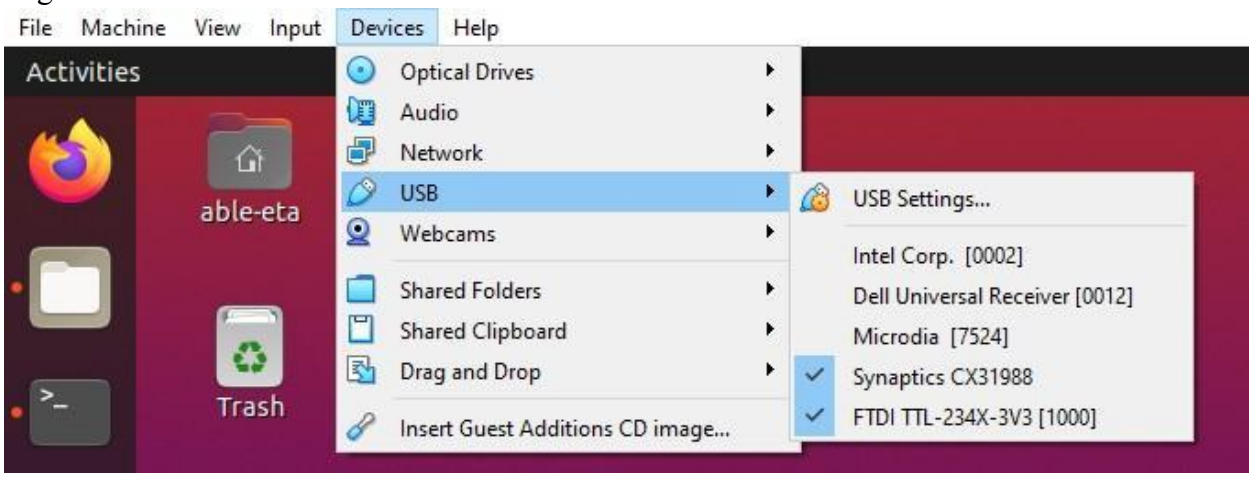
Applications are located inside TENSAI_FLOW/Applications/ directory, and the build needs to be triggered from the build directory inside the application. Below commands show how to build the “compiler-test-app” application.

```
/* Invoke CMake to generate Makefile and configure default
  configurations*/
<TENSAI_FLOW/Applications/compiler-test-app/build> cmake ..

<TENSAI_FLOW/Applications/compiler-test-app/build> make app
```

Above steps will build the target application with default configuration. In default configurations all supported features are not enabled; if the application depends on a feature other than default, that feature needs to be enabled using steps mentioned in section 2.6.

2.2 Running Application

If using a virtual machine, then make sure to select the Synaptics CX31988 and Debug UART as given below.



Follow the below command to load an application.

```
/* load program */
<TENSAI_FLOW/Applications/compiler-test-app/build># make flash
```

2.3 Showing MNIST image

To verify the pipeline, we have provided a pdf file:

`Tools/Tensai_compiler/model_zoo/mnist_model/test_images_for_camera.pdf`

Take a print out of the same and hold the image in front of the camera at a distance as shown in the picture below. Make sure to keep the image not too far away from the camera, use the credit card as a reference as shown in the image.

The model was trained on the MNIST dataset with white numerals with black backgrounds, hence the corresponding colored printouts.

3. Serial port debug

Open the debug UART using minicom or any other similar serial port viewer with 1152000 8N1 configuration.

```
minicom -b 1152000 -8 -D /dev/ttyUSB0
```

The print will show the inference output and the confidence level for each number, as shown below.

```
Image capture done
----Outputs ----
[0]:-127
[1]:-128
[2]:-128
[3]:-127
[4]:-128
[5]:-122
[6]:-128
[7]:-127
[8]:-128
[9]:118
Inference output for mnist : NINE
total infer time in 66 msec, in cycles 1618628
total time (capture + preprocess image + infer) in 587 msec, in cycles 14437155
```