

Final exercise

NXSOL-OJT-2024

Exported on 02/27/2024

Table of Contents

1	Set up	4
1.1	Download tensorflow and its dependencies	4
1.2	Root directory after cloning	4
1.3	Local.conf	4
1.4	Add the layers	5
2	Build image	6
2.1	Set the build environment	6
2.2	Start build	6
3	Verify	7
3.1	Run the built image with slrip + kvm + 5GB memory:	7
3.2	Verify the install	7
3.3	Run tutorial case	7
3.4	TensorFlow/TensorFlow Lite C++ Image Recognition Demo	8
4	Trouble-shooting	9
4.1	Problem	9
4.2	Solution	9

- This tensorflow image is based on the hello world image.
- Reference: <https://git.yoctoproject.org/meta-tensorflow/> (meta-tensorflow/BUILD.md)

1 Set up

1.1 Download tensorflow and its dependencies

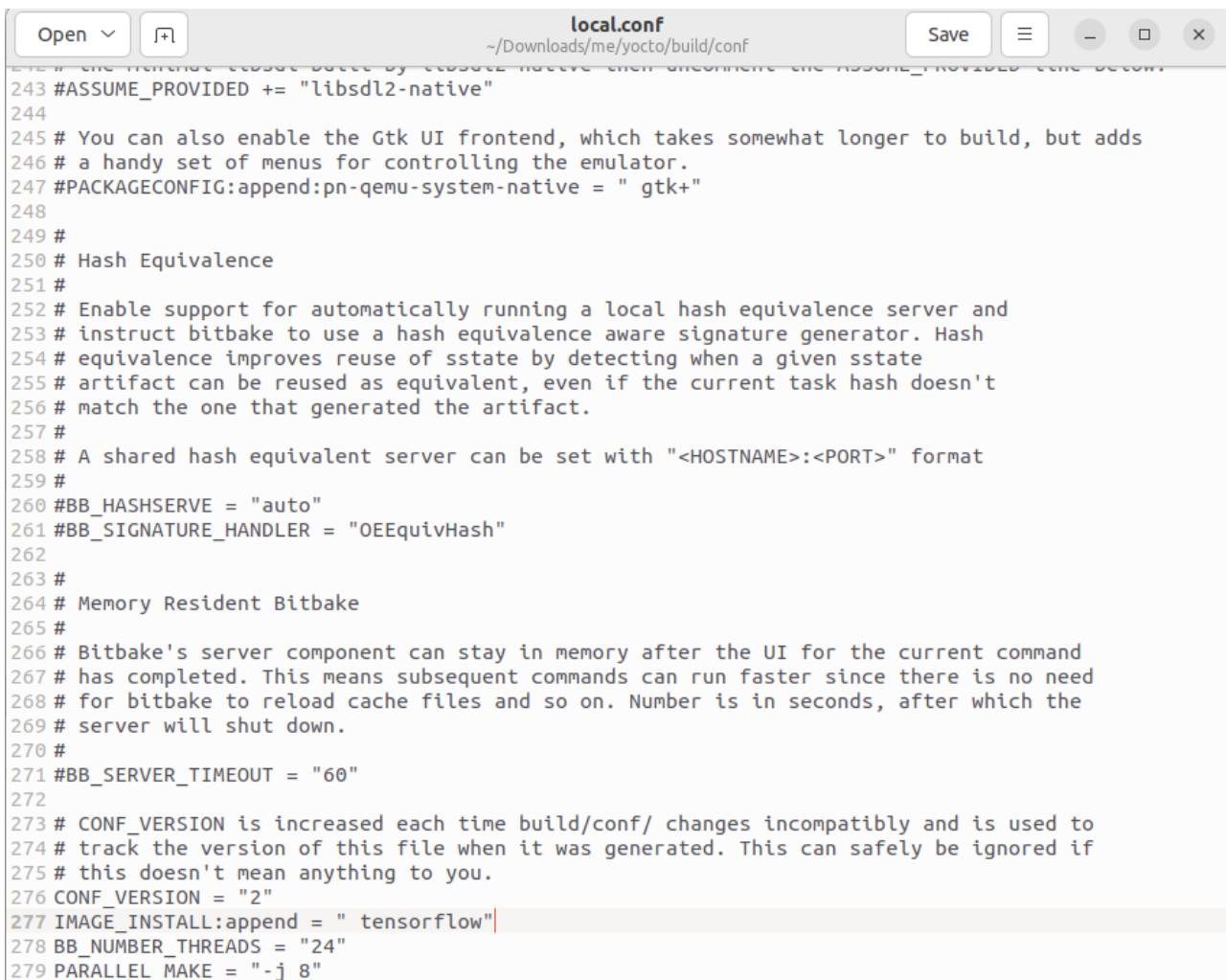
```
git clone -b kirkstone git://git.yoctoproject.org/meta-tensorflow
git clone -b kirkstone git://git.openembedded.org/meta-openembedded
```

1.2 Root directory after cloning



1.3 Local.conf

- Add the tensorflow package to the final image
- Speed up the build by setting multi-threads

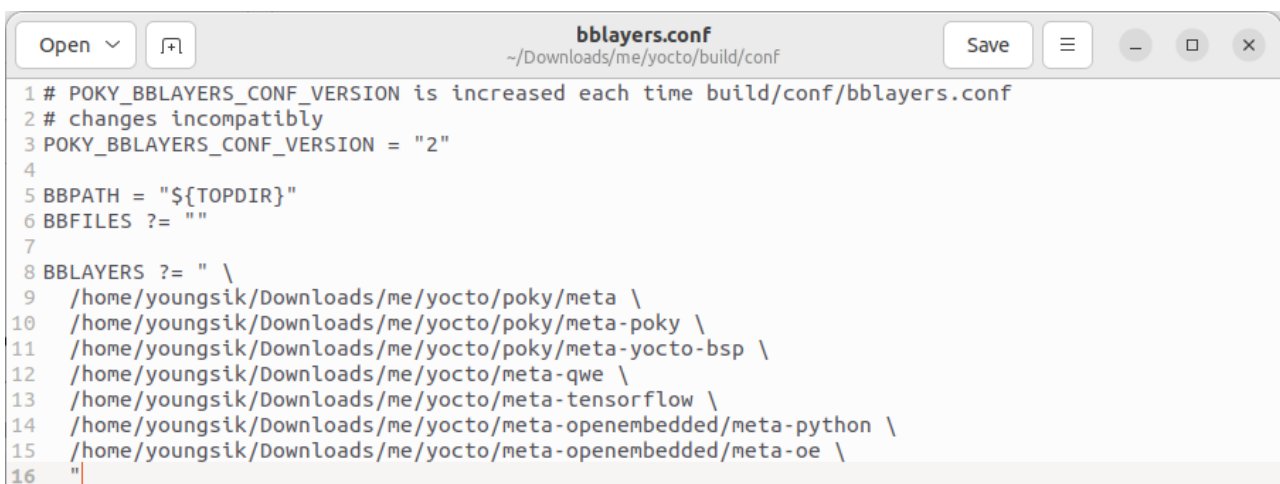


```

243 #ASSUME_PROVIDED += "libSDL2-native"
244
245 # You can also enable the Gtk UI frontend, which takes somewhat longer to build, but adds
246 # a handy set of menus for controlling the emulator.
247 #PACKAGECONFIG:append:pn-qemu-system-native = " gtk+"
248
249 #
250 # Hash Equivalence
251 #
252 # Enable support for automatically running a local hash equivalence server and
253 # instruct bitbake to use a hash equivalence aware signature generator. Hash
254 # equivalence improves reuse of sstate by detecting when a given sstate
255 # artifact can be reused as equivalent, even if the current task hash doesn't
256 # match the one that generated the artifact.
257 #
258 # A shared hash equivalent server can be set with "<HOSTNAME>:<PORT>" format
259 #
260 #BB_HASHSERVE = "auto"
261 #BB_SIGNATURE_HANDLER = "OEEquivHash"
262
263 #
264 # Memory Resident Bitbake
265 #
266 # Bitbake's server component can stay in memory after the UI for the current command
267 # has completed. This means subsequent commands can run faster since there is no need
268 # for bitbake to reload cache files and so on. Number is in seconds, after which the
269 # server will shut down.
270 #
271 #BB_SERVER_TIMEOUT = "60"
272
273 # CONF_VERSION is increased each time build/conf/ changes incompatibly and is used to
274 # track the version of this file when it was generated. This can safely be ignored if
275 # this doesn't mean anything to you.
276 CONF_VERSION = "2"
277 IMAGE_INSTALL:append = " tensorflow"
278 BB_NUMBER_THREADS = "24"
279 PARALLEL_MAKE = "-j 8"

```

1.4 Add the layers



```

1 # POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
2 # changes incompatibly
3 POKY_BBLAYERS_CONF_VERSION = "2"
4
5 BBPATH = "${TOPDIR}"
6 BBFILES ?= ""
7
8 BBLAYERS ?= " \
9   /home/youngsik/Downloads/me/yocto/poky/meta \
10  /home/youngsik/Downloads/me/yocto/poky/meta-poky \
11  /home/youngsik/Downloads/me/yocto/poky/meta-yocto-bsp \
12  /home/youngsik/Downloads/me/yocto/meta-qwe \
13  /home/youngsik/Downloads/me/yocto/meta-tensorflow \
14  /home/youngsik/Downloads/me/yocto/meta-openembedded/meta-python \
15  /home/youngsik/Downloads/me/yocto/meta-openembedded/meta-oe \
16 "

```

2 Build image

2.1 Set the build environment

```
source poky/oe-init-build-env ; cd ..
```

2.2 Start build

```
bitbake -k qwe-image
```

3 Verify

3.1 Run the built image with slirp + kvm + 5GB memory:

```
runqemu qemux86-64 qwe-image slirp kvm qemuparams="-m 5120" nographic
```

3.2 Verify the install

```
python3 -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

Result

```
root@qemux86-64:~# python3 -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
2024-01-25 08:29:04.042127: I tensorflow/core/platform/cpu_feature_guard.cc:186] This TensorFlow binary is optimized with onX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-01-25 08:29:04.831271: I tensorflow/core/platform/cpu_feature_guard.cc:186] This TensorFlow binary is optimized with onX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
tf.Tensor(-2393.77, shape=(), dtype=float32)
```

3.3 Run tutorial case

```
cat >code-v2.py <<ENDOF
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

predictions = model(x_train[:1]).numpy()
tf.nn.softmax(predictions).numpy()
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
loss_fn(y_train[:1], predictions).numpy()
```

```

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=2)

probability_model = tf.keras.Sequential([
    model,
    tf.keras.layers.Softmax()
])
probability_model(x_test[:5])

```

ENDOF

Result

```

root@qemux86-64:~# python3 ./code-v2.py
2024-01-25 08:31:45.542195: I tensorflow/core/platform/cpu_feature_guard.cc:186] This TensorFlow binary is optimized with onX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-01-25 08:31:46.622067: I tensorflow/core/platform/cpu_feature_guard.cc:186] This TensorFlow binary is optimized with onX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/5
1875/1875 [=====] - 1s 638us/step - loss: 0.2975 - accuracy: 0.9137
Epoch 2/5
1875/1875 [=====] - 1s 658us/step - loss: 0.1422 - accuracy: 0.9576
Epoch 3/5
1875/1875 [=====] - 1s 763us/step - loss: 0.1070 - accuracy: 0.9676
Epoch 4/5
1875/1875 [=====] - 1s 688us/step - loss: 0.0881 - accuracy: 0.9730
Epoch 5/5
1875/1875 [=====] - 1s 679us/step - loss: 0.0764 - accuracy: 0.9758
313/313 - 0s - loss: 0.0775 - accuracy: 0.9752 - 204ms/epoch - 653us/step

```

3.4 TensorFlow/TensorFlow Lite C++ Image Recognition Demo

```
time label_image_lite
```

Result

```

root@qemux86-64:~# time label_image_lite
INFO: Loaded model /usr/share/label_image/mobilenet_v1_1.0_224_quant.tflite
INFO: resolved reporter
INFO: invoked
INFO: average time: 23.515 ms
INFO: 0.780392: 653 military uniform
INFO: 0.105882: 907 Windsor tie
INFO: 0.0156863: 458 bow tie
INFO: 0.0117647: 466 bulletproof vest
INFO: 0.00784314: 835 suit
real    0m 0.08s
user    0m 0.25s
sys     0m 0.01s

```


4 Trouble-shooting

4.1 Problem

QEMU does not support access to the internet by default.

4.2 Solution

DNS configuration