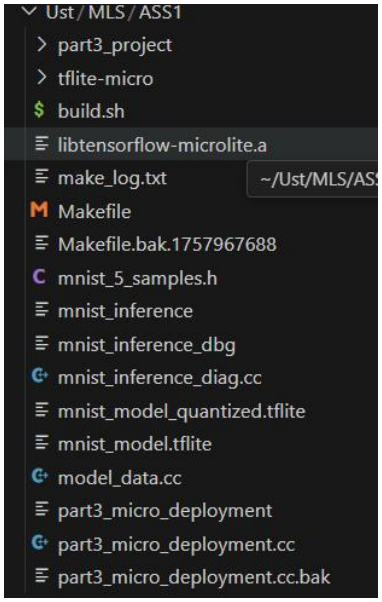
Linux Environment:

AWS Cloud



File tree

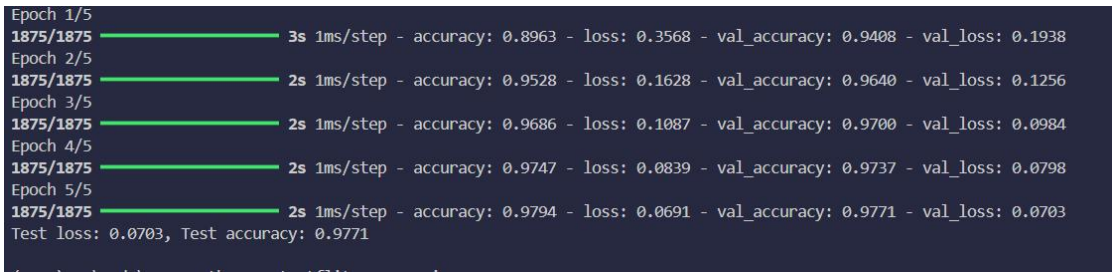
1 Model Size Comparison

Framework	Model Size	Compression Ratio	
Tensorflow	287KB	1	
Tensorflow lite	89KB	0.31	
TFlite Quantized	26KB	0.09	

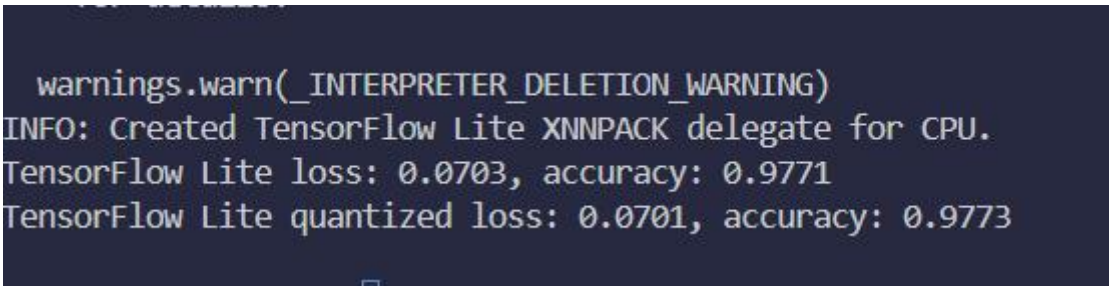
2 Accuracy Comparison

Framework	ACC	LOSS	
Tensorflow	0.9771	0.0703	

Tensorflow lite	**0.9771**	**0.0703**	
TFlite Quantized	**0.9773**	**0.0701**	



Pic1 Result of Part1

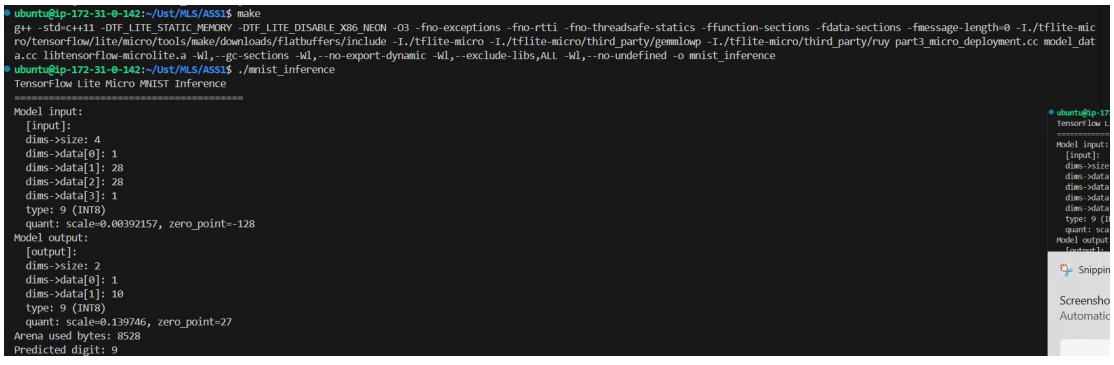


Pic2 Result of Part2

As we can see, in the Mnist training, for the reason that the model is small, and the job is not so hard, lite model performs exactly the same as original, and what's more surprising is that the quantized model even works better a little bit(I have tested 3times).

3 Memory Usage Analysis

Framework	RAM	FLASH	TOTAL
Tensorflow	**~60KB**	**~287KB**	**~350KB**
Tensorflow lite	**~15KB**	**~80KB**	**~100KB**
TFLM	**~1KB**	**~10KB**	**~11KB**



Pic3 Result of Part3

4Inference Performance

Since there is no actual real device but only in simulation, so performance is not available.

5. Development Complexity Analysis

Q1. Python in part1_tensorflow.py and part2_tflite_conversion.py, using TensorFlow/Keras high-level APIs, you can define, train, and convert an MNIST model with relatively few lines of code(2parts add up 200 rows .C++ in part3_micro_deployment.cc, performing inference using TensorFlow Lite for Microcontrollers (TFLM) requires more detailed and low-level code, which is about 250 rows alone.

Q2. Python is mainly depending on pip to install Tensorflow, and some regular dependencies like numpy and so on, while C++ code relies on the only outer dependency is the source cloned form github in Tensorflow lite micro.

Q3. Python dependencies don't need to build ,which is kind of convenient.

But, C++ code especially with this kind of dependency requires build, and making a available makefile is not an easy job.

Q4. Python is easy to debug using ide like pycharm, and is not so hard even using vscode, since I run it in my own computer. However, since the c++ code runs in a remote linux server, the only way to debug is gdb and is not so easy.

Use case:

Tensorflow keras models are used in cloud sever, in model research, large-scale cloud training, backend services requiring highest accuracy without power consumption concerns.

TF lite models are used in edge devices with mobile CPU or GPU， can be Smart Sensors, client-end smart small models

While TFLM models are used in cases like keyword wakeup, sensor date collection and etc, which are usually set on micro chips within 1mW power consumption.

About Quantized

As I mentions as below, quantized model size is reduced to 25% compared to lite models，which is helpful for smaller devices. As for performance, I don't see loss is accuracy in Mnist training, but I know absolutely there will be some loss in converting float to int.The Speed is quite fast for quantized models, but the deployment complexity is much more hard.

As for deployment considerations,

1.how to make ota? It is hard for these kinds of small devices to update, so when deploying, we should consider if this applicaton needs often updates

2. Build difficulty

It is not so easy to build the application within limited environment. If the project need cross compile? Or how can third party dependencies be applied?

3. Data security

Are the date used for model private? Can it be upload to cloud? Can others access to the data through some particular method?

As for future directions,

More generic frameworks and more special frameworks will appear according actual needs;

Maybe it is possible to do some small training on micro devices rather than just inferring.

And most importantly, more advanced hardwares which can lead to stronger performances with less energy cost.