Assignment-2

Sai Teja Kuchi (21317, kuchisai@iisc.ac.in)

Configuration Details:-

Packages :-

python	3.10.6
torch	1.13.1+cu117

Task-1

- The code for training the model on CPU for 10 epochs with the provided hyperparameters can be found in the task1.py file. The test accuracy obtained is 9903/10000 (99%). The weights are saved in the ‘mnist_cnn_cpu.pt’ file in the ‘Results’ folder.

Task-2

- The code for quantizing the above model can be found in the task2.py file. The test accuracy obtained is 9901/10000 (99%). The weights are saved in the ‘mnist_cnn_int8.pt’ file in the ‘Results’ folder.

Task-3

- The code for generating the measurements for normal model, quantized model can be found in the measureFloat.py and measureInt8.py files respectively. The test function in the utils.py file is modified to generate the below report data. The saved weights are loaded into the model to perform the inference on the test dataset.

Accuracy

	Before Quantization	After Quantization
Accuracy	9897/10000 (99%)	9896/10000 (99%)
Average Loss	0.0306	0.0307

- It can be clearly observed that there is a very minute increase in the Average loss of the quantized model with respect to the normal model which is leading to few incorrect predictions while still maintaining the same percentage of accuracy to that of the normal

model. So for the inference tasks its better to use quantized model because the weights are fixed, compared to the training phase where the weights needs to be updated during the back propagation phase which might lead to heavy accuracy differences.

Runtime

- When calculating the total runtime, the time required for loading the test data of input and output is not considered. Both the tasks are run thrice and the average runtime for the complete test data with batch size of 64 in milli seconds is reported.
- Memory tracking is commented out while running this step.

	Before Quantization	After Quantization
Average Runtime (ms)	787.7371311187744	549.9086380004883

- It can be seen that after quantization there is a slight improvement in the overall runtime. So incase the models are much deeper with large number of weights, a much better improvement can be observed in terms of the runtime after performing quantization.

Speedup = $\frac{Runtime\ before\ quantization}{Runtime\ after\ quantization}$ = $\frac{787.737131118774}{549.9086380004883}$ = 1.4324

Model Compression Ratio

- The model size is reported in Mega Bytes for both the tasks.

	Before Quantization	After Quantization
Model Size(MB)	1.727115	0.449627

- Its nothing much of a surprise that the model size has reduced drastically as we are going from float datatype to int datatype.

Compression Ratio = $\frac{Size\ of\ the\ model\ before\ quantization}{Size\ of\ the\ model\ after\ quantization}$ = $\frac{1.727115}{0.449627}$ = 3.9252613636363636

Peak memory usage

- When calculating the memory usage, the memory required for loading the test data of input and output is not considered as they will be same for both. Both the tasks are run thrice and the average runtime of those 3 for a single batch input is reported in Bytes.
- Timing calculation is commented out while running this step.

	Before Quantization	After Quantization
Peak Memory Usage (B)	8185	6584

- It can be observed that due to use of quantization there is a reduction in the amount of memory that is used when compared to that of the non-quantized model

So in conclusion, seeing all the above data we can infer that its good to perform quantization of an model for the inference phase as it doesn't require any weight update as they are fixed throughout the whole inference phase. One can also explore the possibility of using quantization during the training phase which has proven to provide reasonable results in the papers published.