Group name: How Do You Turn This On

Group members present in lab today: Yi-Ting Yeh,Ting-Rui Chiang

1: Models

1. Which models and/or model variants will your group be studying in this lab? What is the original bit width of the models, and what precision will you be quantizing to? What parts of the model will be quantized (e.g. parameters, activations, ...)? Please be specific.

   We choose the same model sets Transformer and LSTM with varied CTC decoder weights as Lab2. Specifically, both Transformer and LSTM have 3 variants which are CTC weight = 0 (e.g. autoregressive decoder), CTC weight = 1 (e.g. only CTC decoder), and CTC weight = 0.3 which means the hybrid decoder.

   Originally, models are using 32 bits floating points, and we will quantize them to 8-bit integers. We only do dynamic quantization on linear layers since the implementation of the quantized LSTM cell causes errors on our Raspberry Pi.

2. Why did you choose these models?

   We choose the same models to Lab2 to observe the effect of quantization.

3. For each model, you will measure model size (in (mega,giga,...)bytes), and inference latency. You will also be varying a parameter such as input size or batch size. What are your hypotheses for how the quantized models will compare to non-quantized models according to these metrics? Do you think latency will track with model size? Explain.

   We did not expect that quantization will accelerate the model, because int-8 operations are not accelerated on the ARM Cortex-A72 CPU Raspberry Pi 4 has and quantization does not reduce the number of required operations.

2: Quantization in PyTorch

1. Try changing the model to mobilenet_v3_large and set quantize=False. (Note that quantize=True may fail due to unsupported operations.) What happens?

   Mobilenet_v3_large with quantize=False gives a prediction “bluetick” while mobilenet_v2 with quantize=True says “Great Dane”. We directly open the image and agree that the dog in that image should be bluetick. On the other hand, mobilenet_v3_large has a significantly larger model size 22134.609 KB while the size of mobile_v2_large is only 3625.937 KB. The results show that, although quantization can effectively compress the model size, it might hurt the performance.

2. Any difficulties you encountered here? Why or why not?

   To run the quantized models on our ARM device, we need to set `torch.backends.quantized.engine` to “qnnpack” manually. On the other hand, we should set it to “fbgemm” if we test the models using the cloud server.

3: Model size

1. Compute the size of each model.
   Transformer:
   - Encoder: 333240.511 KB, Quantized: 120201.487 KB
   - Decoder: 121456.191 KB, Quantized: 38323.147 KB
   - CTC decoder: 10261.031 KB, Quantized: 2581.727
   - LM: 343293.444 KB, Quanitzed: 312934.142

   LSTM:
   - Encoder: 27217.042 KB, Quantized: 24762.428 KB
   - Decoder: 5841.467 KB, Quantized: 5194.077 KB
   - CTC decoder: 35.687 KB, Quantized: 10.463 KB
   - LM: 27225.156 KB, Quantized: 27173.31 KB
2. Any difficulties you encountered here? Why or why not?
   a. As we mentioned before, we can’t run the quantized version of the LSTM cell on Raspberry Pi. Therefore we only quantized linear layers here.

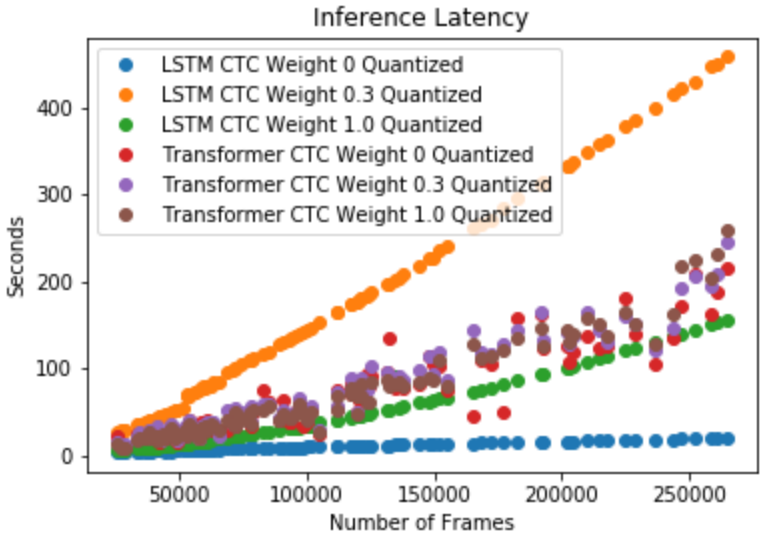
4: Latency

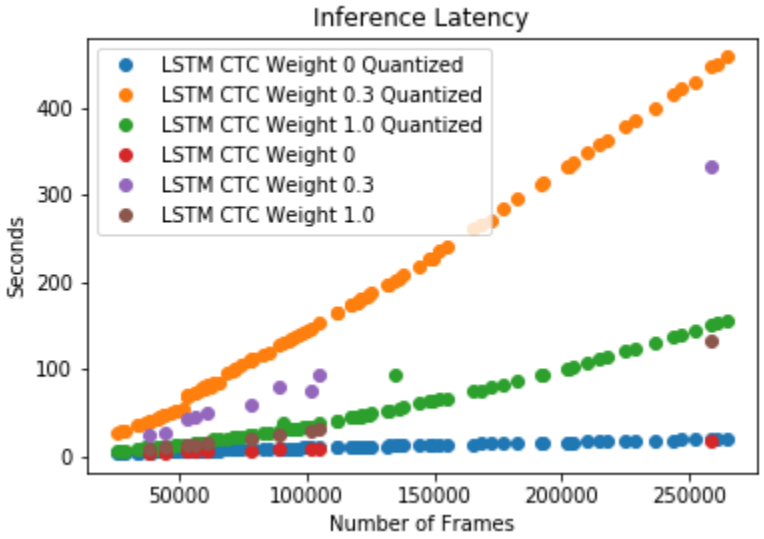
1. Compute the inference latency of each model:

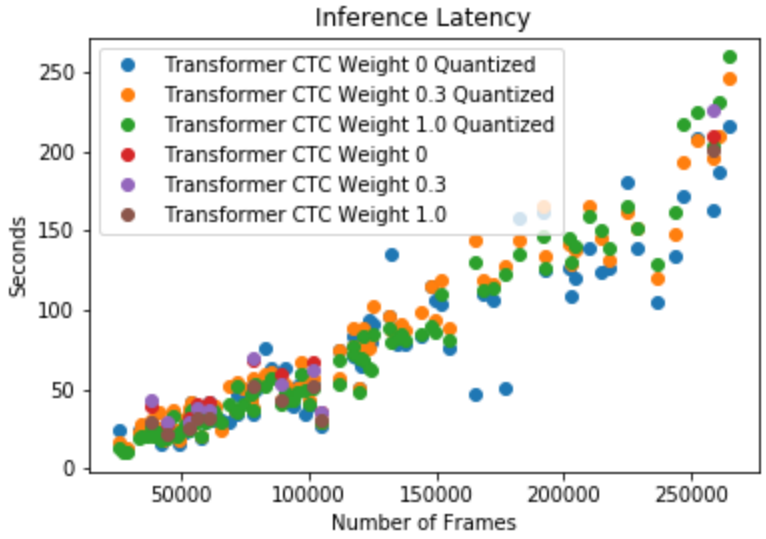
Latency / WER	Transformer	Quantized Transformer	LSTM	Quantized LSTM
CTC Weight: 0	62.1 secs / 47.5%	48.6 secs / 59.6%	7.8 secs / 100%	9.3 secs / 100%
CTC Weight: 0.3	62.2 secs / 3.3%	53.6 secs / 7.4%	82.3 secs / 100%	130.2 secs / 100%
CTC Weight: 1.0	51.8 secs / 2.9%	50.2 secs / 3.3%	30.2 secs / 100%	34.9 secs / 100%

2. Plot the results.

We choose to vary the number of input frames and compare results in Lab2.


Inference Latency
LSTM CTC Weight 0 Quantized
LSTM CTC Weight 0.3 Quantized
LSTM CTC Weight 1.0 Quantized
Transformer CTC Weight 0 Quantized
Transformer CTC Weight 0.3 Quantized
Transformer CTC Weight 1.0 Quantized
Seconds
0
100
200
300
400
50000
100000
150000
200000
250000
Number of Frames


Inference Latency
LSTM CTC Weight 0 Quantized
LSTM CTC Weight 0.3 Quantized
LSTM CTC Weight 1.0 Quantized
LSTM CTC Weight 0
LSTM CTC Weight 0.3
LSTM CTC Weight 1.0
Seconds
0
100
200
300
400
50000
100000
150000
200000
250000
Number of Frames



2. Any difficulties you encountered here? Why or why not?
   We lost connection to our device because of network issues (CMU-device).

5: Discussion

1. Analyze the results. Do they support your hypotheses? Why or why not? Did you notice any strange or unexpected behavior? What might be the underlying reasons for that behavior?
   We do not observe apparent speed up for the transformer model, which is as our expectation. However, the LSTM model becomes slower after quantization. We conjecture that it is because we do not quantize most parts of the LSTM model, and the overhead of converting float32 to int8 outweighs the small acceleration of replacing float32 operations with int8.

5: Extra

A few options:

1. Try to run static quantization, or quantization aware training (QAT). Benchmark and report your results. Here is a nice blog post on using static quantization on Torchvision models in PyTorch.
2. Compute FLOPs and/or energy use (if your device has the necessary power monitoring hardware) for your models.
3. Evaluate on different hardware (for example, you might run the same benchmarking on your laptop.) Compare the results to benchmarking on your device(s).

4. Use real evaluation data and compare accuracy vs. efficiency. Describe your experimental setup in detail (e.g. did you control for input size? Batch size? Are you reporting average efficiency across all examples in the dev set?) Which model(s) appear to have the best trade-off? Do these results differ from benchmarking with synthetic data? Why or why not?
