CSC 591
REAL TIME AND HIGH-PERFORMANCE ML
UNIT PROJECT 2
sthakur5
xhui

(1) What quantization methods you have applied to which DNN model.?

Model Used	VAE	
Quantization Method 1	QAT Quantizer	This affects the forward pass of the training and backpropagation is left unchanged
Quantization Method 2	DoReFa Quantizer	During backward pass, parameter gradients are stochastically quantized to low bitwidth numbers before being propagated to convolutional layers. As convolutions during forward/backward passes can now operate on low bitwidth weights and activations/gradients respectively, DoReFa-Net can use bit convolution kernels to accelerate both training and inference
Quantization Method 3	LSQ Quantizer	Estimate and scale the task loss gradient at each weight and activation layer's quantizer step size, such that it can be learned in conjunction with other network parameters.

(2) What configurations did you try on each quantization method, and what machine(s) you have used, including the CPU and GPU models of the machine(s).
(2) Configuration used for each quantizer:

Quantizatior	Configuration	
QAT Quantizer	C0	[{ 'quant_types': ['input', 'weight'], 'quant_bits': {'input': 8, 'weight': 8}, 'op_names': ['fc1', 'fc21','fc22','fc3'] }]
	C1	[{ 'quant_types': ['input', 'weight'], 'quant_bits': {'input': 4, 'weight': 4}, 'op_names': ['fc1', 'fc21','fc22','fc3'] }]

DoReFa Quantizer	C0	[{
		'quant_types': ['weight'],
		'quant_bits': {
		'weight': 8,
		}, 'op_types':['Linear']
		}]

	C1	[{
		'quant_types': ['weight'],
		'quant_bits': {
		'weight': 4}, 'op_types':['Linear']
		}]

LSQ Quantizer	C0	[{ 'quant_types': ['input', 'weight'], 'quant_bits': {'input': 8, 'weight': 8}, 'op_names': ['fc1', 'fc21','fc22','fc3'] }]
	C1	[{ 'quant_types': ['input', 'weight'],

		'quant_bits': {'input': 4, 'weight': 4}, 'op_names': ['fc1', 'fc21','fc22','fc3'] }]

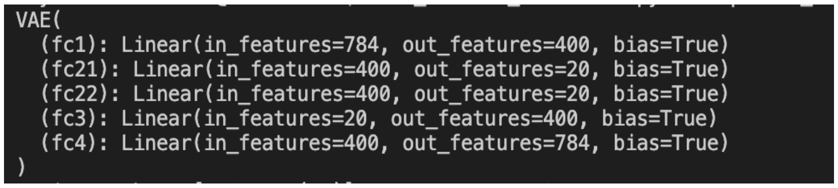
System Configuration:

Cluster	Node	CPU Detail	GPU Detail
NCSU ARC Cluster	c25	**Architecture**: x86_64 **Thread(s) per core:** 2 **Core(s):** 16 Model name: **AMD EPYC 7302P** 16-Core Processor **CPU MHz:** 1496.343 **CPU max MHz:** 3000.0000 **CPU min MHz:** 1500.0000	**GPU make**:NVIDIA GeForce RTX 2060 SUPER, 90.06.45.00.01 **Memory**: 8GB

(3) What results you have obtained, including the output of "print(model)" of your original model and pruned model, their running speeds and accuracies;

(3.) **Model size:**

Print model output is same for all configurations:



```
VAE(
  (fc1): Linear(in_features=784, out_features=400, bias=True)
  (fc21): Linear(in_features=400, out_features=20, bias=True)
  (fc22): Linear(in_features=400, out_features=20, bias=True)
  (fc3): Linear(in_features=20, out_features=400, bias=True)
  (fc4): Linear(in_features=400, out_features=784, bias=True)
)
```

Reason:
Quantization only changes the bits used for the weights/inputs. It does not affect the actual shape of the model.

Quantizer	Configuration	"calibration_config" output
QAT Quantizer	C0	{'fc1': {'weight_bits': 8, 'weight_scale': tensor([0.0039], device='cuda:0'), 'weight_zero_point': tensor([162.], device='cuda:0'), 'input_bits': 8, 'tracked_min_input': 0.0, 'tracked_max_input': 1.0}, 'fc21': {'weight_bits': 8, 'weight_scale': tensor([0.0024], device='cuda:0'), 'weight_zero_point': tensor([122.], device='cuda:0'), 'input_bits': 8, 'tracked_min_input': 0.0, 'tracked_max_input': 6.204421520233154}, 'fc22': {'weight_bits': 8, 'weight_scale': tensor([0.0028], device='cuda:0'), 'weight_zero_point': tensor([116.], device='cuda:0'), 'input_bits': 8, 'tracked_min_input': 0.0, 'tracked_max_input': 6.204421520233154}, 'fc3': {'weight_bits': 8, 'weight_scale': tensor([0.0112], device='cuda:0'), 'weight_zero_point': tensor([153.], device='cuda:0'), 'input_bits': 8, 'tracked_min_input': -3.3633360862731934, 'tracked_max_input': 3.3483726978302}}
QAT Quantizer	C1	{'fc1': {'weight_bits': 4, 'weight_scale': tensor([0.0619], device='cuda:0'), 'weight_zero_point': tensor([9.], device='cuda:0'), 'input_bits': 4, 'tracked_min_input': 0.0, 'tracked_max_input': 1.0}, 'fc21': {'weight_bits': 4, 'weight_scale': tensor([0.0323], device='cuda:0'), 'weight_zero_point': tensor([8.], device='cuda:0'), 'input_bits': 4, 'tracked_min_input': 0.0, 'tracked_max_input': 5.1463494300842285}, 'fc22': {'weight_bits': 4, 'weight_scale':

		tensor([0.0377], device='cuda:0'), 'weight_zero_point': tensor([6.], device='cuda:0'), 'input_bits': 4, 'tracked_min_input': 0.0, 'tracked_max_input': 5.1463494300842285}, 'fc3': {'weight_bits': 4, 'weight_scale': tensor([0.1587], device='cuda:0'), 'weight_zero_point': tensor([7.], device='cuda:0'), 'input_bits': 4, 'tracked_min_input': -3.7945075035095215, 'tracked_max_input': 3.6436119079589844}}
DoReFa Quantizer	C0	{'fc1': {'weight_bits': 8}, 'fc21': {'weight_bits': 8}, 'fc22': {'weight_bits': 8}, 'fc3': {'weight_bits': 8}, 'fc4': {'weight_bits': 8}}
DoReFa Quantizer	C1	{'fc1': {'weight_bits': 4}, 'fc21': {'weight_bits': 4}, 'fc22': {'weight_bits': 4}, 'fc3': {'weight_bits': 4}, 'fc4': {'weight_bits': 4}}
LSQ Quantizer	C0	{'fc1': {'weight_bits': 8, 'tracked_min_input': -2.161423683166504, 'tracked_max_input': 2.161423683166504, 'input_bits': 8}, 'fc21': {'weight_bits': 8, 'tracked_min_input': -12.510388374328613, 'tracked_max_input': 12.510388374328613, 'input_bits': 8}, 'fc22': {'weight_bits': 8, 'tracked_min_input': -6.47183895111084, 'tracked_max_input': 6.47183895111084, 'input_bits': 8}, 'fc3': {'weight_bits': 8, 'tracked_min_input': -4.632283687591553, 'tracked_max_input': 4.632283687591553, 'input_bits': 8}}

LSQ Quantizer	C1	`{'fc1': {'weight_bits': 4, 'tracked_min_input': -0.9929074048995972, 'tracked_max_input': 0.9929074048995972, 'input_bits': 4}, 'fc21': {'weight_bits': 4, 'tracked_min_input': -2.6289753913879395, 'tracked_max_input': 2.6289753913879395, 'input_bits': 4}, 'fc22': {'weight_bits': 4, 'tracked_min_input': -2.565124034881592, 'tracked_max_input': 2.565124034881592, 'input_bits': 4}, 'fc3': {'weight_bits': 4, 'tracked_min_input': -2.5324721336364746, 'tracked_max_input': 2.5324721336364746, 'input_bits': 4}}`

Running Speed:
Since VAE is a generative model we calculate the running speed by simply measuring the training and evaluation times (epochs=10).

Pruner	configure	Training Time(seconds)	Evaluation Time(seconds)	Model Training Time After quantization (seconds)	Model Evaluation After quantization (seconds)
QAT	C0	7.526	1.344	7.57	1.338
QAT	C1	7.43	1.344	7.54	1.31
DoReFa	C0	7.45	1.34	7.51	1.36
DoReFa	C1	7.42	1.37	7.46	1.37
LSQ	C0	7.56	1.34	7.53	1.37
LSQ	C1	7.45	1.38	7.46	1.35

Observations:
There is no improvement in training time and evaluation time before and after quantization.

Reason:
We only apply the one quantization process without fine-tuning and speedup process. This process is a quantization simulation process. It does not make an effect on the real model. If we want to get real quantization, we need to apply fine-tuning and speedup process. After

this, the model is really quantized and we may get some speedup. Currently, we cannot get any speedup with quantization simulation.

<u>**Accuracy comparison:**</u>
Since VAE is an auto-generative model and we are not predicting anything we need to figure out the metrics necessary to compare the performance of original model and quantized model. These are:

1. Loss

Total Loss for all configurations:

Quantizer	Configuration	Loss	Loss After quantization
QAT	C0	105.47	103.92
QAT	C1	105.61	111.31
DoReFa	C0	105.59	105.73
DoReFa	C1	105.62	106.02
LSQ	C0	105.75	104.58
LSQ	C1	105.33	108.04

Observations:
We can get the most accuracy with QAT quantizer with 8 bits, the least accuracy with QAT quantizer with 4 bits.

Reason:
- We can see that there is not much loss difference before and after quantization for DoReFa Quantizer. Because the DoReFa quantizer is designed for low bitwidth convolutional neural networks, thus, we can get the same accuracy before and after quantization. Also, when we set the configuration with 4 bits (C1), we can still get the similar accuracy compared with 8 bits(C0).
- For QAT quantizer, we will get less accuracy with 4 bits configuration compared with 8 bits. Because QAT quantizer is designed for Inter-Arithmetic-Only Inference. When we set 8 bits, the type is int, all quantization types share the same length. While with the bit 4, it’s half an int. Thus, we will lose some accuracy with the QAT quantizer.
- For LSQ quantizer, it’s learned step size quantization. With 4 bits, we will get less accuracy compared with 8 bits. With less bits, we will loss the precision during training and thus get less accuracy.

(4) What lessons you have learned through the project?

1. Decreasing the number of floating point bits affects the overall performance of the model. As seen, in configuration 0(C0) we are getting better performance compared to configuration 1(C1) where we have more degree of quantization in C1.

2. Performing quantization requires complex software support for GPU which is hard to achieve. In this project, we tried using TensorRt but didn't succeed doing the same owing to the permission issues on the arc cluster. This shows how esoteric quantization is.

3. Quantizating more layers leads to worse performance than quantizing selected layers.

4. The quantizer only performs the simulated quantization. We need to perform the fine-tuning and speedup process to get a real quantized model.

5. For the general quantizer, we will lose accuracy. We need to choose an appropriate quantizer with a good configuration to get good performance while keep high accuracy.

(5) A link to your GitHub repository containing all the scripts used in this project and a README describing the repository's structure and how the folders in the repository corresponding to the items in the report. The report must be in PDF, with the five required parts clearly marked
(5) https://github.com/Sakshamrzt/CSC591-P2

References:
1.https://nni.readthedocs.io/en/stable/reference/compression/quantizer.html#qat-quantizer
2. https://arxiv.org/abs/1606.06160
3. https://arxiv.org/pdf/1902.08153.pdf