GE Global Research

Data Science Institute
COLUMBIA UNIVERSITY

Energy Efficient AI on Edge

DSI Poster Presentation | December 11, 2020

Mentors:
Tapan Shah, GE Research
Eleni Drinea, Columbia University

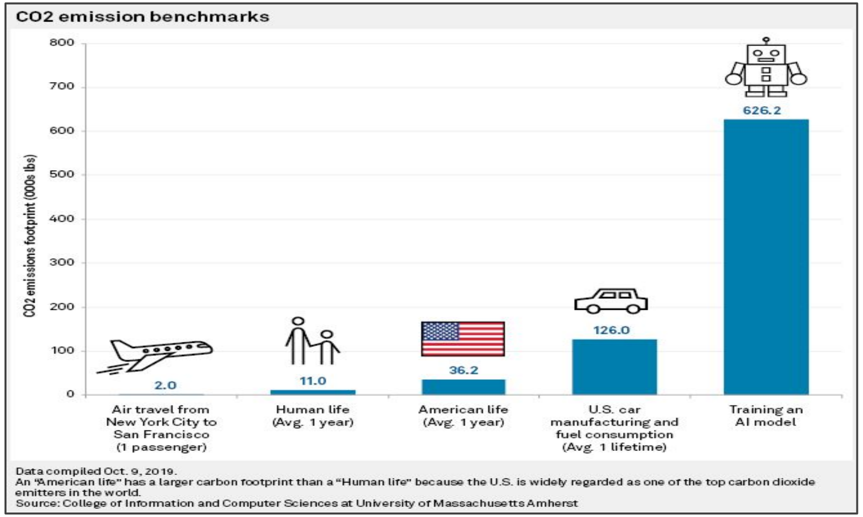
Presentors:
Kumari Nishu (kn2492)
Pritam Biswas (pb2796)
Neelam Patodia (np2723)
Mohit Chander Gulla (mcg2208)
Prasham Dhaneshbhai Sheth (pds2136)

Agenda

Problem Statement

Rising Carbon Emissions from AI needs our attention

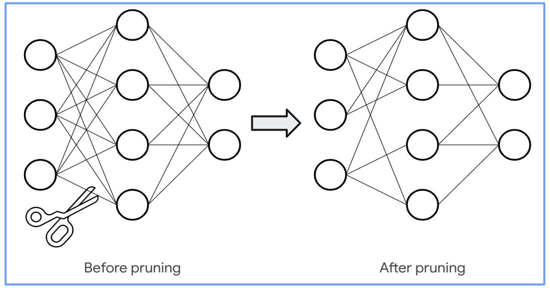
- Present day neural networks such as BERT, ELMo tend to be deep, with millions of weights and activations. These large models are **compute** and **memory intensive**
- 300,000x increase in computation required for deep learning research between 2013-2018*. Energy consumption is not limited to model training
- The carbon emissions from training a deep learning model is equivalent to **5 times** the lifetime emissions of an average car
- Hence, **GE** is **focussing** on **low-latency** and **lighter** NN models **without compromising on accuracy**, to be deployed on its **EDGE** devices



Data compiled Oct. 9, 2019.
An "American life" has a larger carbon footprint than a "Human life" because the U.S. is widely regarded as one of the top carbon dioxide emitters in the world.
Source: College of Information and Computer Sciences at University of Massachusetts Amherst

* Roy Schwartz, Allen Institute for Artificial Intelligence

Our Approach for Carbon Efficient AI



- **Pruning**
  - Removing less contributing weight from the network
  - Inducing sparsity in weights and activations, having sparsity allows us to make inference and model storage more efficient

- **Quantization**
  - Reducing the number of bits that represent a number
  - Performing computations and storing tensors at lower bit width than the predominant FP32 format
  - We have focused on quantizing only the weights using Post Training and Quantization Aware Training in 2-16 precision range

Quantization Techniques

Dynamic Quantization

The activations and weights are quantized on the fly. Only the computations (multiplication and convolutions) are efficient. Activations are read & written in full precision

Post-Training Static Quantization

Allows for both integer arithmetic operations and memory access. This is achieved by first obtaining distribution of weights and using that to quantize it

Quantization Aware Training

Leads to highest accuracy. During training, activations and weights are fake quantized. Thus adjustments are made with a priori knowledge they will be eventually quantized

Data and Model Selection

We experimented different quantization and pruning techniques over a variety of models across multiple datasets

Model Types

Model Category	Model Complexity	Dataset
ANN based Classification	Simple: 2 dense layers Complex: 5 dense layers	Churn Data Telescope Data
ANN based Regression	Simple: 2 dense layers Complex: 5 dense layers	California Housing Data MV Data
CNN based Classification	Vanilla CNN ResNet-9 ResNet-50 VGG16	Fashion MNIST CIFAR-100

Dataset Details

Dataset Name	Rows	Features	Classes
Churn	10K	14	2
Telescope	19K	11	2
California Housing	20.6K	8	-
MV Data	40.7K	10	-
Fashion MNIST	70K	-	10
CIFAR- 100	60K	-	100

1. **Post Training Quantization**
2. Pruning
3. Quantization Aware Training

Methods - Single-Point Quantization

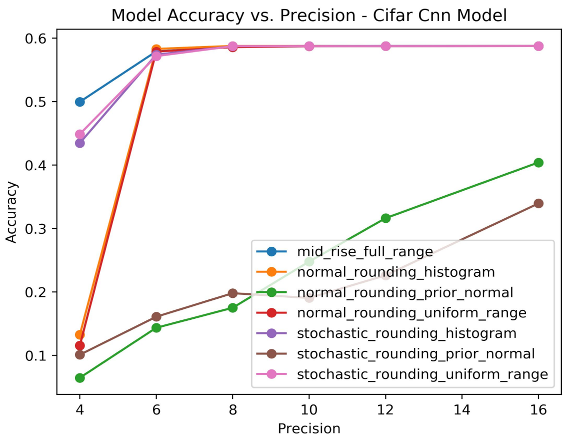
Single-point Quantization approximates a weight value using a single low precision number.

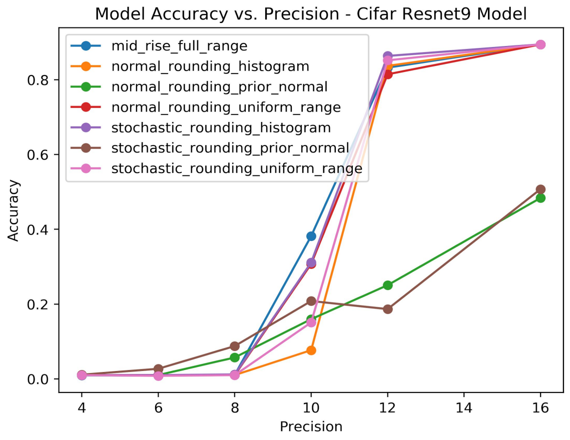
1. **Mid-Rise**
   a. Delta - controls granularity of data quantization, high delta implies high quantization and significant loss of information
   b. Uniform division of range of Weight values into 2^p bins for p precision
   c. w_quantized = Delta * (floor(w/Delta) + 0.5)

2. **Regular Rounding**
   a. Quantization Set - collect a set of landmark values using uniform bin, histogram, prior normal on weight values
   b. Map each weight value to the nearest landmark value from quantization set

3. **Stochastic Rounding**
   a. Quantization Set - collect a set of landmark values using uniform bis, histogram, prior normal on weight values
   b. Assign each weight value to either the closest smaller value or the closest larger value from quantization set probabilistically

Results - Single Point Quantization





CIFAR100

Accuracy of simple model = **0.58**,

Accuracy of complex model = **0.89**

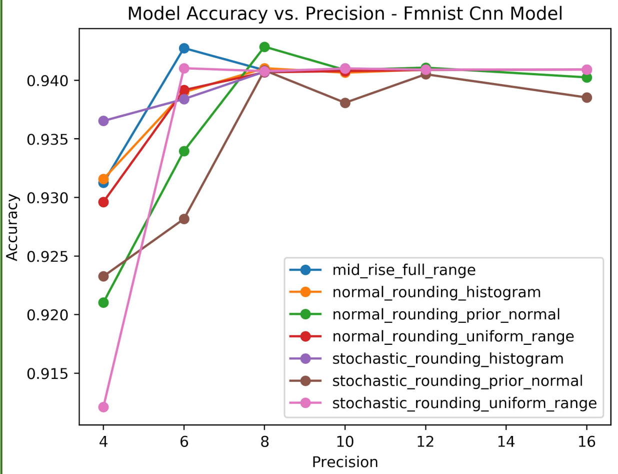
ResNet with huge parameters suffer more from quantization

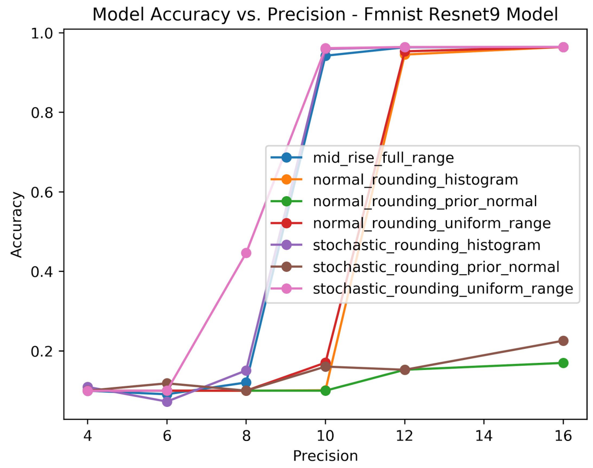
FMNIST

Accuracy of simple model = **0.92**,

Accuracy of complex model = **0.94**

ResNet with huge parameters suffer more from quantization





Methods - Multi-Point Quantization

Multi-point Quantization approximates a weight value using a linear combination of multiple values of low precision.

4. Multi-point - mixed precision method

a. Assign more bits to important layers, and fewer bits to unimportant layers to balance the accuracy and cost more efficiently
b. Achieves the same flexibility as mixed precision hardware but using only a single-precision level
c. The quantization set, Q, is constructed using a uniform grid on [-1, 1] with increment epsilon. K as scaling factor and B is center of Q

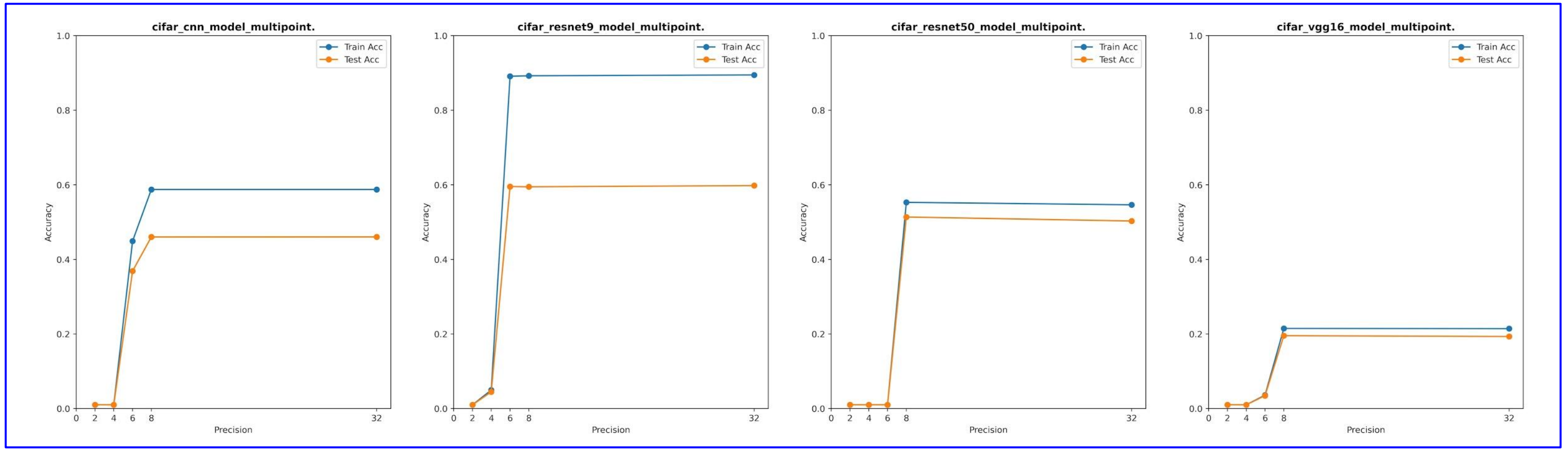
$$\mathcal{Q} = K \times [-1 : \epsilon_b : 1] \; + \; B, \quad \epsilon_b := \frac{1}{2^{b-1} - 1}$$

d. Each weight value w is approximated as follows. With a_i as a real number and wi~ from Qd for all i=1, ... , n

$$\tilde{\boldsymbol{w}} = \sum_{i=1}^{n} a_i \tilde{\boldsymbol{w}}_i$$

e. Apply above quantization method for weight matrix W of all layers one by one.

Results - Multi Point Quantization



CIFAR100

Avg Accuracy of Stochastic Rounding, Uniform in = 0.276

Avg Accuracy of Multipoint = 0.347

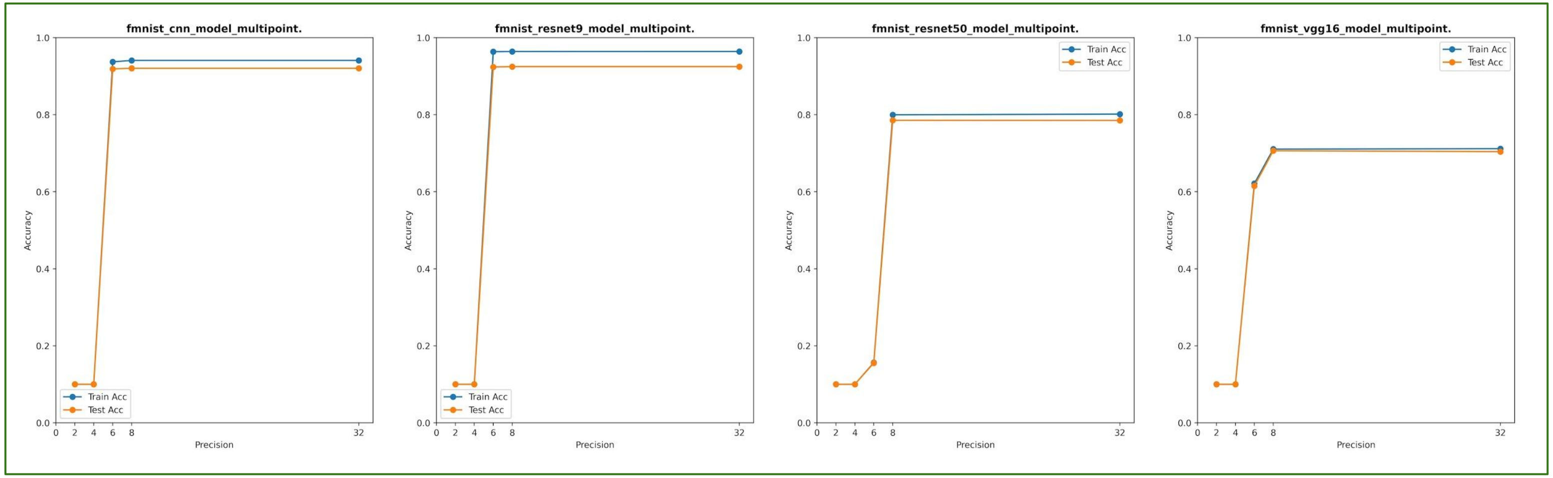
Improvement = 25.7%

FMNIST

Avg Accuracy of Stochastic Rounding, Uniform Bin = 0.568

Avg Accuracy of Multipoint = 0.647

Improvement = 14.1%



1. Post Training Quantization
2. **Pruning**
3. Quantization Aware Training

Pruning

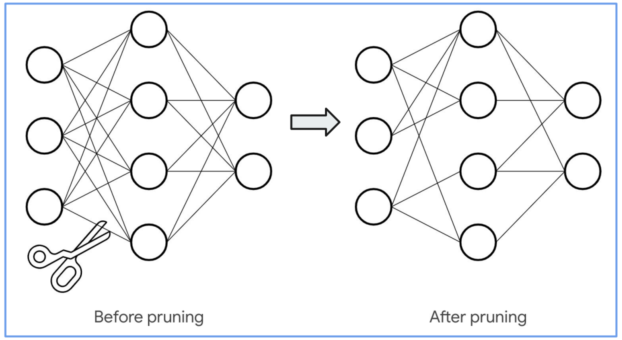
Pruning: Method of compression that involves removing less contributing weights from a trained model.

Pruning

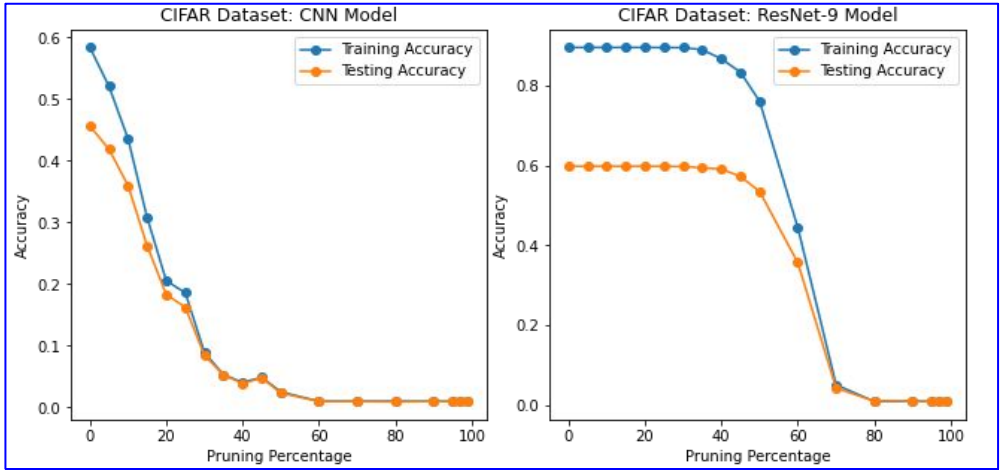
- We are practically setting the neural network parameters' values to zero to remove what we estimate are less contributing (unnecessary connections) between the layers of a neural network.
- We use the magnitude of weights to determine the importance of the weights towards the model's performance.

Combining Post Training Pruning and Post Training Quantization

- We further experimented to combine the two compression techniques: Pruning, Quantization
- We formulated them as 2 independent tasks and were applied in the following order:
    I. Post Training Quantization
    II. Pruning



Results - Pruning



CIFAR100

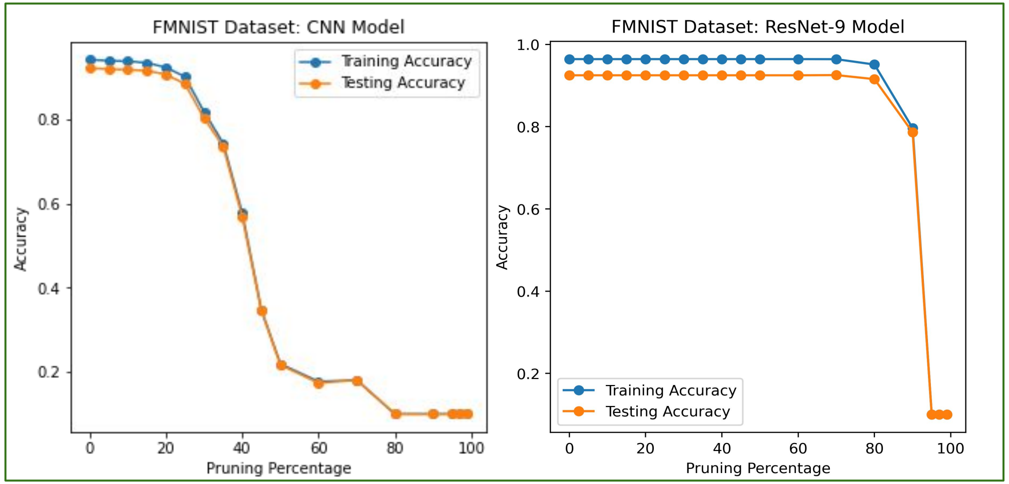
Accuracy of simple model (before pruning) = **0.58**
Accuracy of complex model (before pruning) = **0.89**

Performance of the simpler model with less number of parameters is affected a lot as compared to the more complex model

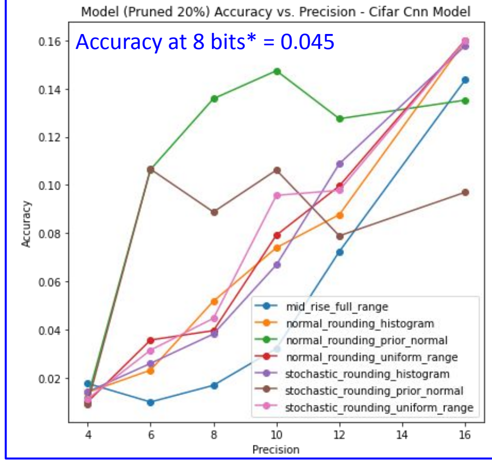
FMNIST

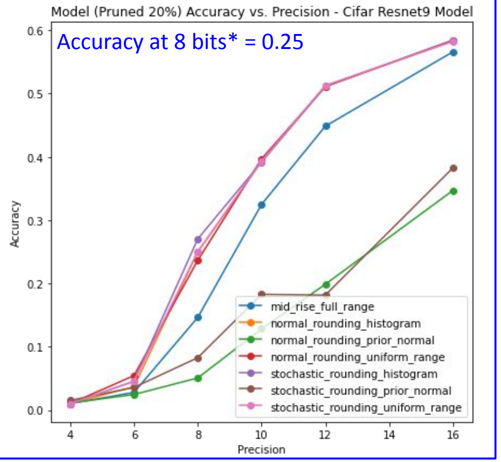
Accuracy of simple model (before pruning) = **0.92**
Accuracy of complex model (before pruning) = **0.94**

Performance of the simpler model with less number of parameters is affected a lot as compared to the more complex model



Results - Pruning + Quantization



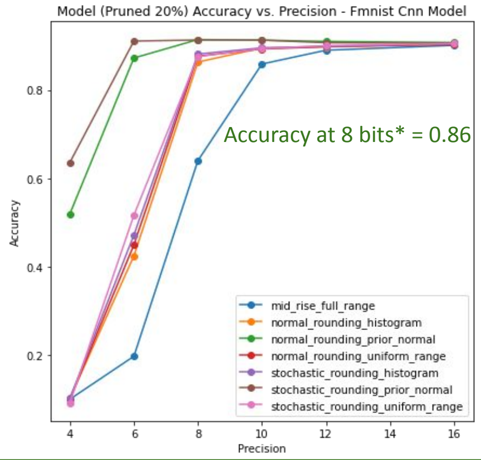


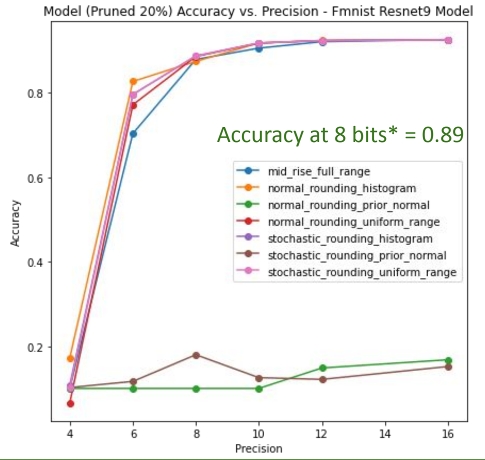
CIFAR100

Accuracy of unpruned simple model **(32 bit)** = **0.58**
Accuracy of unpruned complex model **(32 bit)** = **0.89**

Accuracy of unpruned simple model **(8 bit)*** = **0.46**
Accuracy of unpruned complex model **(8 bit)*** = **0.0097**

The performance of pruned model after quantization depends a lot on the pruning percentage as well as how the model behaves before pruning





FMNIST

Accuracy of unpruned simple model **(32 bit)** = **0.92**
Accuracy of unpruned complex model **(32 bit)** = **0.94**

Accuracy of unpruned simple model **(8 bit)*** = **0.92**
Accuracy of unpruned complex model **(8 bit)*** = **0.44**

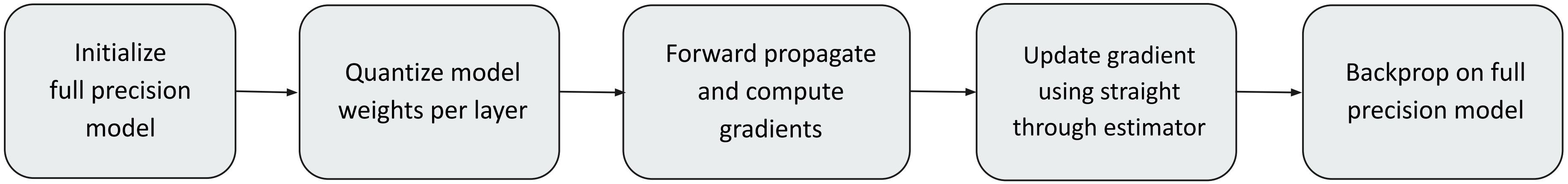
The performance of pruned model after quantization depends a lot on the pruning percentage as well as how the model behaves before pruning

* Quantized using Stochastic Rounding and Uniform Range as the unique value generator

1. Post Training Quantization
2. Pruning
3. **Quantization Aware Training**

Quantization Aware Training

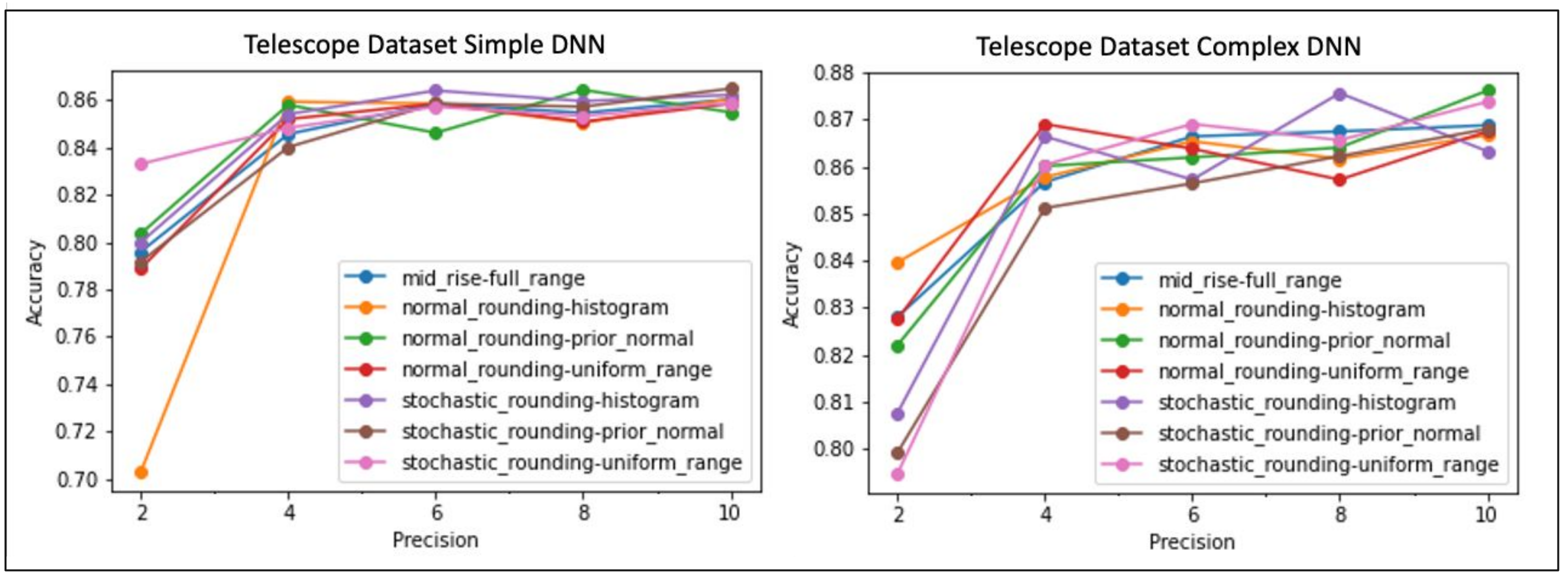
Quantization Aware Training is a process of training the model assuming that it will be quantized later during inference


Initialize full precision model
Quantize model weights per layer
Forward propagate and compute gradients
Update gradient using straight through estimator
Backprop on full precision model

Key observations

- The biases of the model weights are not quantized for any layer during the training
- Straight through estimator is used to calculate the last term of the gradient during backprop
- The model obtained after QAT is at full precision and must be quantized before storage or inference

Results - Quantization Aware Training



<u>**Telescope Dataset**</u>
Accuracy of simple model (before QAT) = **0.86**
Accuracy of complex model (before QAT) = **0.87**

The complex model at full precision was performing slightly better than the simple model, which may be due more complex features generated by complex model

Key Takeaways

- There is **not much variability in accuracy across different quantization techniques** in quantization aware training as opposed to post-quantization schemes.
- Even stochastic rounding-prior normal and normal rounding-prior normal schemes perform well at all quantization levels, unlike during post-training scheme
- The similar performance of all QAT strategies **can be attributed to the closed loop feedback update** system
- We notice a marginal improvement in the accuracy metrics, given that we quantized within 10 bits only
- **Training times are considerably high** especially for deep convolutional networks, which is a major drawback

Conclusion & Future Work

Conclusion

Single-Point Quantization

- **Quantization Method:** Stochastic Rounding performs better than Regular Rounding and Mid-Rise in accuracy
- **Model Complexity:** Complex models are impacted heavily due to quantization. CNN accuracy at 8 bits ~ full precision accuracy and DNN accuracy at 6 bits ~ full precision accuracy
- **Data Complexity:** In classification problems with more number of classes, the accuracy deteriorates faster

Multi-Point Quantization

- It outperforms single point post-quantization methods for both simple and complex model architectures across both CIFAR-100 and FMINST datasets with an improvement of **19.8%** on accuracy

Pruning

- It is more effective in complex and deeper models as compared to simple model at full precision. Moreover, pruning a quantized model depends on pruning % and how well the unpruned model is trained.

Quantization-Aware Training

- It marginally improves accuracy by **5%** - more in case of simple than for complex models and does not depend much on the quantization technique used. It comes at a cost of longer training time.

Future Work

Model Size

- To get a complete picture of each method's effectiveness, we need to observe model size at different levels of precision. This relates to our objective of reducing the carbon footprint of deep learning models.

Quantize Activations

- Along with quantizing weights, we can explore quantization of activations as well.

Improve Training Algorithm

- Most of the carbon emissions are caused due to the intensive computations required during the training. (e.g.) BERT and GPT-3 require a lot a computation to learn the parameters. We can explore techniques to get smart weight updates and reduce computations required during the training.

Hardware Simulations

- Experiment on specialized low-precision hardware to accurately evaluate different quantization techniques.

Acknowledgement

Acknowledgement

We would like to acknowledge and thank our project mentor and advisor for their efforts and expert guidance:

Tapan Shah - Lead Machine Learning Scientist, GE Research

Eleni Drinea - Lecturer, Data Science Institute, Columbia University

For more detailed information about the project, please refer to:

1. GitHub Repository: https://github.com/mohitgulla/Edge
2. Capstone Progress Reports: Phase 1 and Phase 2

References

References

1. Xi. Liu, Mao Ye, Dengyong Zhou, & Qiang Liu. (2020). Post-training Quantization with Multiple Points: Mixed Precision without Mixed Precision.
2. Shaokai Ye and Tianyun Zhang and Kaiqi Zhang and Jiayu Li and Jiaming Xie and Yun Liang and Sijia Liu and Xue Lin and Yanzhi Wang (2018). A Unified Framework of DNN Weight Pruning and Weight Clustering/Quantization Using ADMM CoRR, abs/1811.01907.
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers Found. Trends Mach. Learn., 3(1), 1–122.
4. Raghuraman Krishnamoorthi (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper CoRR, abs/1806.08342.
5. Benoit Jacob and Skirmantas Kligys and Bo Chen and Menglong Zhu and Matthew Tang and Andrew G. Howard and Hartwig Adam and Dmitry Kalenichenko (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference CoRR, abs/1712.05877
6. Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin and Song Han (2018). HAQ: Hardware-Aware Automated Quantization CoRR, abs/1811.08886.
7. Song Han, Jeff Pool, John Tran, William Dally (2015). Learning Weights and Connections for Efficient Neural Networks CoRR, abs/1506.02626.
8. Suyog Gupta, Ankur Agrawal, Kailash G., Pritish Narayanan (2015). Deep Learning with Limited Numerical Precision CoRR, abs/1502.02551.
9. Aojun Zhou and Anbang Yao and Yiwen Guo and Lin Xu and Yurong Chen (2017). Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights CoRR, abs/1702.03044
10. https://pytorch.org/docs/stable/nn.html#utilities
11. https://www.tensorflow.org/model_optimization/guide/pruning
12. https://pytorch.org/blog/introduction-to-quantization-on-pytorch/