Quantization in ML

Why and What Quantization means ?

- Neural networks are very resource intensive algorithms. They not only incur significant computational costs, they also consume a lot of memory in addition.
- When we are running our models on the edge, network optimization becomes even more significant
- Since more and more models move from the servers to the edge, reducing size and computational complexity is essential
- **Quantization** - Fundamental idea behind quantization is that if we convert the weights and inputs into integer types, we consume less memory and on certain hardware, the calculations are faster.

What Quantization does ?

- There can be many ways on which quantization to apply depending upon the technique used
    - Binary , -1/1
    - Ternary - -1/0/1
    - Multi-Bit
- Affine & Scale Quantization -
- Statement - Convert a range of data [A1,A2] to B-bits Range

  Ie linearly transforming the range [A1,A2] to [-2^(B-1) ,2(B-1)-1]

  Affine - f(x) = mx + c

  Scaling - f(x) = mx + (c=0)

- Converting to INT8 is most common one

Where to Apply the Quantization ?

- In NN's, Quantization is applied on weights and activation functions

When to Apply the Quantization ?

- **Post-Training** - Train the model using *float32* weights and inputs, then quantize the weights. Its main advantage that it is simple to apply. Downside is, it can result in accuracy loss.
- **Quantization-aware training-** quantize the weights during training. Here, even the gradients are calculated for the quantized weights. When applying *int8* quantization, this has the best result

Methods and their performance

Source: TensorFlow Lite documentation

Technique	Data requirements	Size reduction	Accuracy	Supported hardware
Post-training float16 quantization	No data	Up to 50%	Insignificant accuracy loss	CPU, GPU
Post-training dynamic range quantization	No data	Up to 75%	Accuracy loss	CPU, GPU (Android)
Post-training integer quantization	Unlabelled representative sample	Up to 75%	Smaller accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP
Quantization-aware training	Labelled training data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP

On many CNN’s

Model	Top-1 Accuracy (Original)	Top-1 Accuracy (Post Training Quantized)	Top-1 Accuracy (Quantization Aware Training)	Latency (Original) (ms)	Latency (Post Training Quantized) (ms)	Latency (Quantization Aware Training) (ms)	Size (Original) (MB)	Size (Optimized) (MB)
Mobilenet-v1-1-224	0.709	0.657	0.70	124	112	64	16.9	4.3
Mobilenet-v2-1-224	0.719	0.637	0.709	89	98	54	14	3.6
Inception_v3	0.78	0.772	0.775	1130	845	543	95.7	23.9
Resnet_v2_101	0.770	0.768	N/A	3973	2868	N/A	178.3	44.9

Source: [TensorFlow Lite documentation]

Other ways of Quantization

- **Partial Quantization -** In Partial Quantization, the idea is to do Quantization only for a few layers in a specific Machine Learning model and leave out the other layers

- **Learning Quantization Parameters Itself -**
  - The idea in Learning Quantization Parameters is to find the values for Quantization parameters like scale value, zero point, range value and others during the training of the model when the weights are calculated.
  - This produces good accuracy with most models as the Quantization parameters are associated with the weights.

Impacts of Quantization

- As while quantizing the weights and activation functions, it is evident that there is some information loss and could result in loss of accuracy
- More compressed quantization may result to high loss in accuracy [more than 1%]
- We know that NN's consists of many parameters, weights so by changing the weights a bit, difference in loss can be enormous

Apart from Quantization

Apart from Quantization, there can be some other ways

- **Pruning** - Pruning involves removing connections between neurons or entire neurons, channels, or filters from a trained network
- **Knowledge Distillation -** involves a large teacher model and a small student model, in which the student model learns the representations of the teacher model.
- **Neural Architecture Search (NAS) -** it is a systematic, automized way of learning optimal model architectures. The idea is to remove human bias from the process to arrive at novel architectures that perform better than human-designed ones