Neural Network Quantization

Energy efficient machine learning and the need for quantization:

Deep neural networks are energy hungry and growing faster.Due to the digital world,there are lot of data available .Availability of data boosts complex model.In complex model,there are lot of moves between memory and compute which leads to higher memory consumption,power consumption,increased latency etc...

There are three ways to reduce memory consumption,powerconsumption and increased latency.

* compression
* Quantization
* Compilation

Compression :
Learn to prune model and maintaining same accuracy.

Quantization:
Reduce memory footprints of overparameterized model.In simple words,storing weights in low bits.

Compilation:
Learning to compile AI models for efficient hardware utilization.

Significance of quantization:

* Quantization reduces storage bits.
* Significant reduction in energy for both compute and memory access
* Due to lower bit representation,it decreases latency
* It occupies less silicon area

Quantization computation:

Quantization approximate an float tensor with an integer tensor multiplied by scale factor.

$w = s * w_{int8} = \hat{w}$

Ex :

Quantization:

x:tensor([
[0.0114, -0.0231, 0.0445],

[0.0000, 0.0000, 0.0000],

[-0.0409, -0.0435, -0.0400]
])

choose scale as 1/255:

Divide the tensor by scale:

= 255* ([
[0.0114, -0.0231, 0.0445],

[0.0000, 0.0000, 0.0000],

[-0.0409, -0.0435, -0.0400]
])

= ([[2.9070, -5.8905, 11.3475],

[0.0000, 0.0000, 0.0000],

[-10.4295, -11.0925, -10.2000]])

Round the tensor to nearest integer:

([[3., -6., 11.],

[0., 0., 0.],

[-10., -11., -10.]])

Dequantization:

Divide the tensor by scale :

=1/255*([[3., -6., 11.],

[0., 0., 0.],

[-10., -11., -10.]])

= ([[0.0118, -0.0235, 0.0431],

[0.0000, 0.0000, 0.0000],

[-0.0392, -0.0431, -0.0392]])

Quantization error : W-$W^{\wedge}$: (
[[0.0004, -0.0004, -0.0014],
[0.0000, 0.0000, 0.0000],
[0.0017, 0.0004, 0.0008]])

Types of quantization:

* symmetric signed quantization
* symmetric unsigned quantization
* asymmetric quantization

symmetric signed quantization:

$X=s*x_{int8}$

int8 range : -128 to 127

symmetric unsigned quantization:

$X=s*x_{unit8}$

uint8 range : 0 to 255

Asymmetric quantization:

$X=S*(X_{uint8}-Z)$

where Z is zero point.

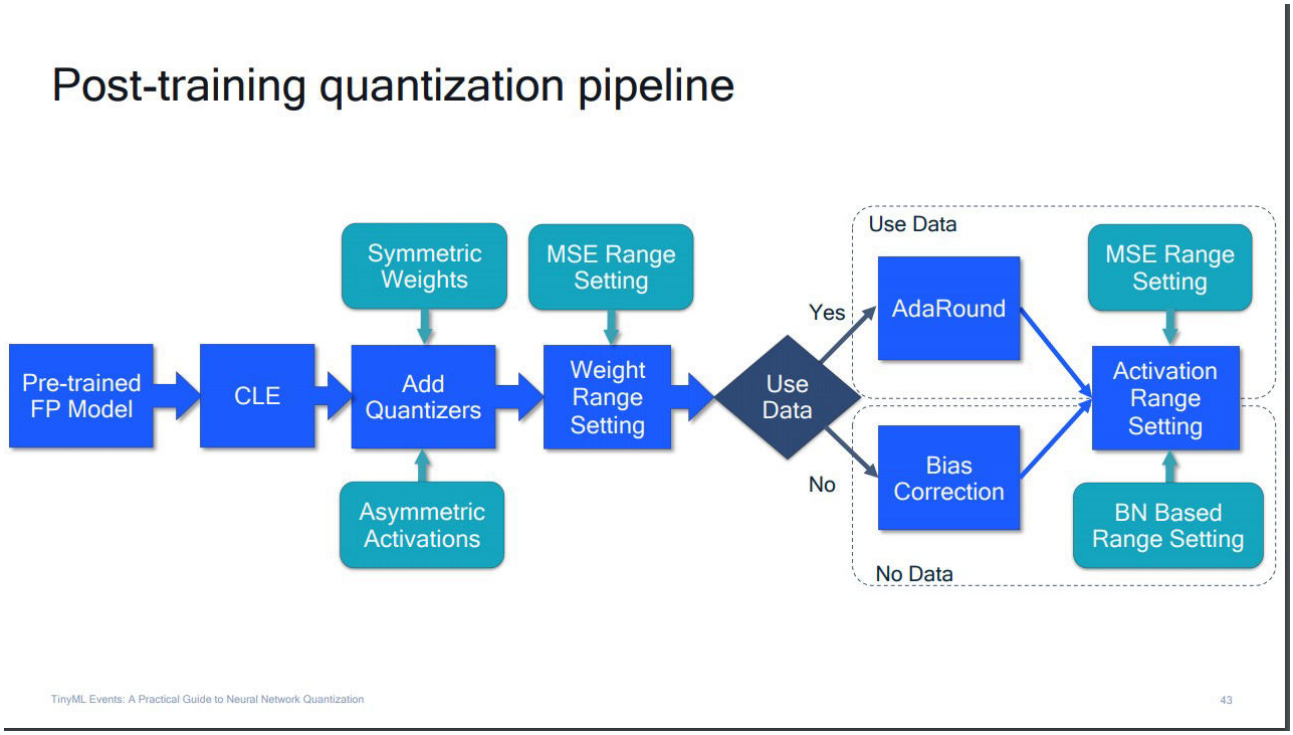
What type of quantizations should we use ?

Symmetric weights and asymmetric activations more hardware efficient

When to apply quantization:

Post training quantization (PTQ)	**Quantization Aware Training (QAT)**
* It applies quantization to pretrained network which is trained in fp32 and converts it into fixed point precision without accessing training pipeline.	* Requires access to training pipeline
* Data free or small calibration needed	* Takes longer training time
* Use through simple API calls	* Takes place during training process.requires hyper parameter tuning
* Lower accuracy at lower bit widths	* Achieves higher accuracy

Post training quantization pipeline :



Quantization followed in paper :

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Quantization:

Example weights :

tensor([
[0.0114, -0.0231, 0.0445],

[0.0000, 0.0000, 0.0000],

[-0.0409, -0.0435, -0.0400]
])

Compute absolute max along axis 1 :

0.0445, 0.00 , 0.0435

Inorder to avoid divide by zero error while computing scale :

replace 0 with 127.0 : (2^8-1)

0.0445, 127 , 0.0435

compute scaling vector:

scale=127/absmax

2.8539326e+03, 1.0000000e+00, 2.9195403e+03

convert float to int:

multiply weights and scale:

array([
[32.534832, -65.92584 , 127.],

[0. , 0. , 0.],

[-119.409195, -127. , -116.78161]], dtype=float32)

Round float to int8:

array([
[33, -66, 127],

[0, 0, 0],

[-119, -127, -117]], dtype=int8)

Store weights and scale :

weights :

array([
[33, -66, 127],

[0, 0, 0],

[-119, -127, -117]], dtype=int8)

scale:

2.8539326e+03, 1.0000000e+00, 2.9195403e+03

References:

* https://cms.tinyml.org/wp-content/uploads/industry-news/tinyML_Talks-_Marios_Fournarakis_210929.pdf

* Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation - https://arxiv.org/abs/1609.08144