Introduction to Edge Computing and Tensorflow Lite

Vaidheeswaran Archana

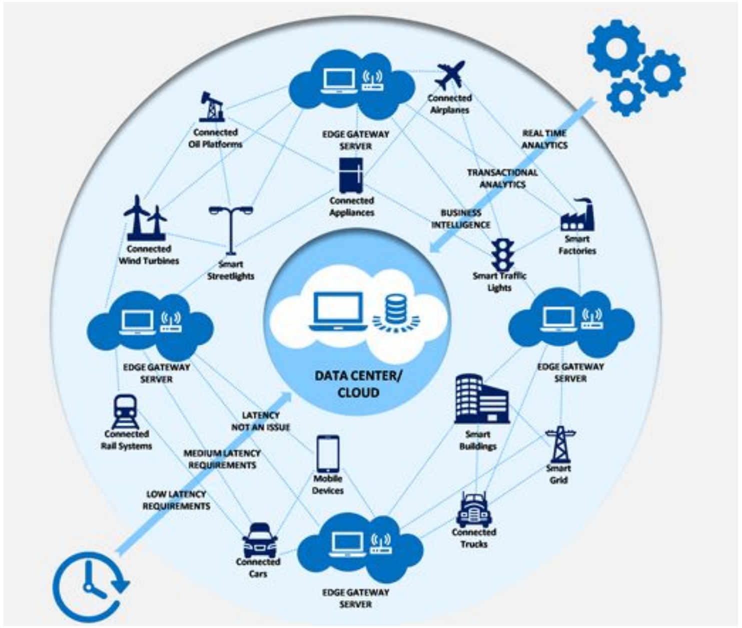
About Me

1. Graduate Student at NUS
2. Deep Learning and Smart grids Research
3. Previously, Research Engineer at Saama Technologies
4. Instructor at Udacity, Intel Edge AI for IoT Developers

What will you learn

1. Edge Computing
2. Advantages of Edge Computing
3. Need for Quantization
4. Definition of Quantization
5. How does INT8 conversion work?
6. Quantization in simple words
7. Tensorflow Lite Optimization Toolkit
8. Post Quantization Techniques
9. Quantization Aware Training

Edge Computing



1- Increase in IoT devices causes an increase in cloud dependency

2-Need edge devices which have their own data centres

3-The computing that is performed in those data centers is Edge computing

4- Application: Security Cameras, Self Driving Cars

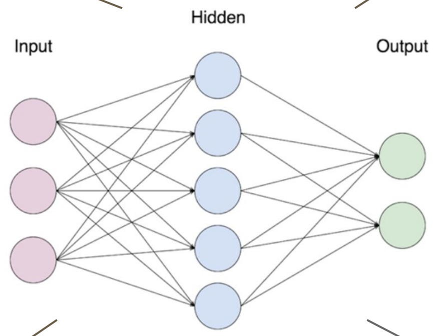
Advantages of Edge Computing

- Reduced Latency
- Reduced Internet Bandwidth
- Increased Security
- Reduction in Dependence on Cloud Services

Need for Quantization

Power constraints:
Traditionally neural networks require massive amount of computational power and energy while running on CPU and GPUs

Composed of Floating Point Values: Neural Networks were traditionally trained to preserve accuracy and speed

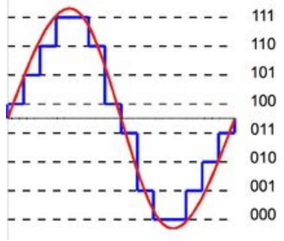


Memory Constraints:
Laptops and PC come with at least 4GB of RAM whereas Raspberry Pi has 1GB of RAM

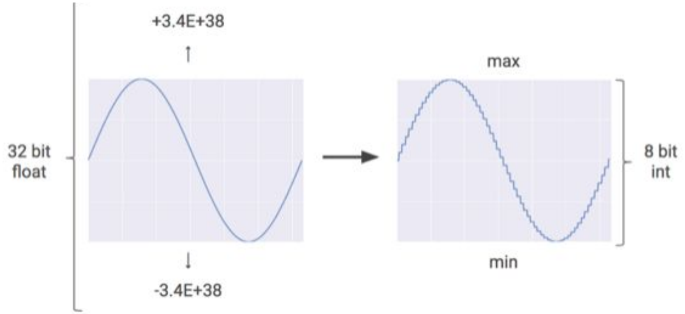
Inference Efficiency:
We need model that takes less time for inference

Definition of Quantization

Actual Definition: It is the process of mapping of input values to a large set of output values in a smaller set, with a finite number of elements; used in mathematics and signal processing.



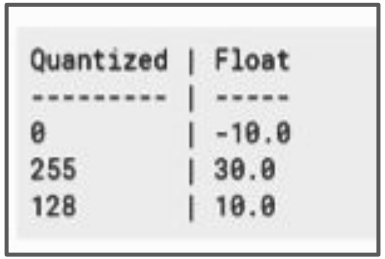
Definition used for Edge Computing: It is the process of reducing the FP-32-bit values to lower bit precision values while preserving the information in the neural networks



How does INT8 Quantization work?

For eg, your neural networks has weight values from -10.00 to. 30.00

- Define Max and Min values
- While doing INT8 quantization, the least value = 0
- The maximum value = 255. ; since 2^8=256
- All the values in between are scaled to the range of 0 to 255



```
Quantized | Float
--------- | -----
0         | -10.0
255       | 30.0
128       | 10.0
```

What happens when we want to represent value = 10.00?

oldRange = oldMax - oldMin

newRange = newMax - newMin

newValue = ((oldValue - oldMin) * newRange / oldRange) + newMin

oldRange = 30-(-10) = 30+10 = 40

newRange = 255-0 = 255

newValue = ((10-(-10))*255/40)+0

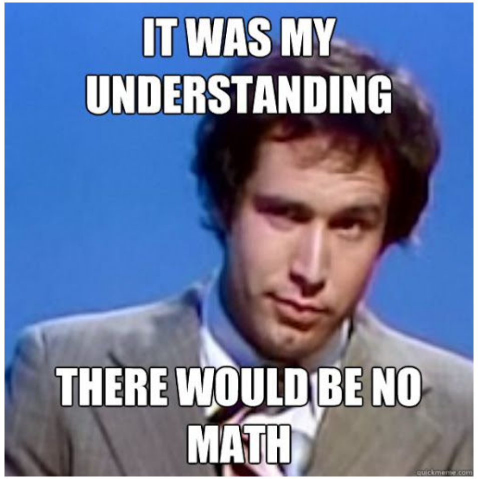
= 127.5~rounding off to 128

Need for Quantization in Simple Words

- If it’s raining outside, you probably don’t need to know exactly how many droplets of water are falling per second — you just wonder **whether it’s raining lightly or heavily.**

- Similarly, **neural network predictions** often **don't require** the **precision of** floating point calculations with **32-bit** or even **16-bit** precision.

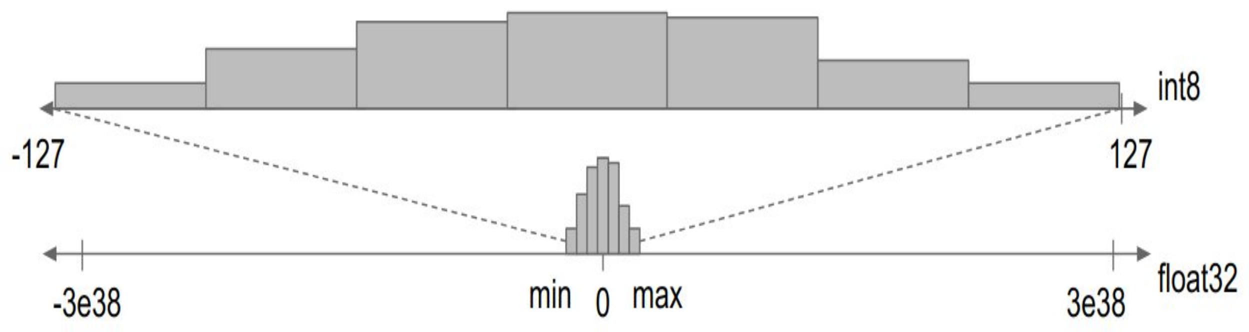
- With some effort, you may be able to **use 8-bit integers** to perform a neural network inference and **still maintain** the appropriate level of **accuracy.**



Tensorflow Optimization Toolkit

1. It contains tools for
   a. Quantization Aware Training
   b. Model Pruning
   c. Post Training Quantization
2. Choose layers you want to quantize in the model
3. Pair the reduced-precision operation definitions in the resulting model with kernel implementations that use a **mix of fixed- and floating-point math.**
4. The most sensitive ones with higher precision, thus typically resulting in little to no final accuracy losses for the task, yet a significant speed-up over pure floating-point execution
5. **Case of Hybrid Kernels: Reconvert the parameters to the higher floating-point precision for execution**

Types of Quantization

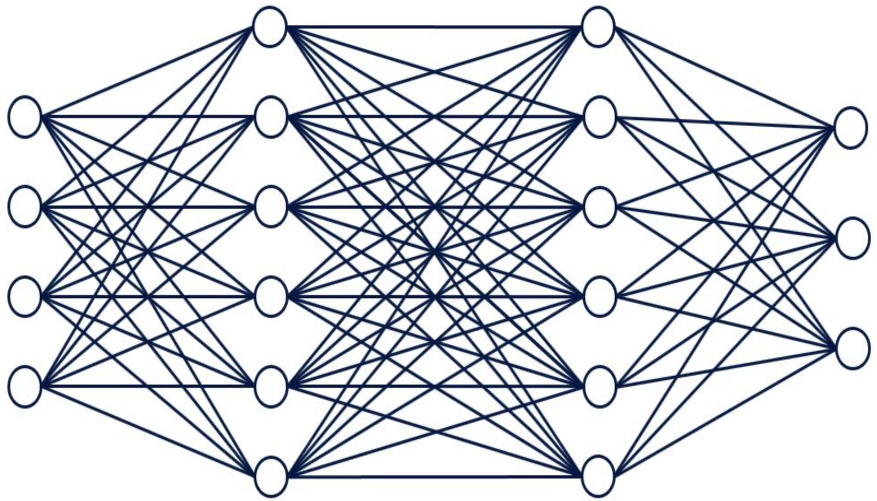


1. In float 32 you have values from -1 to +1, but in int 8 you are converting it to 256 numbers
2. Weight Quantization
3. Weight and Activation Quantization

FORWARD PASS

→

I/P



O/P

Loss

BACKWARD PASS

←

Post Training Quantization

Forward Pass

1. 32 bits

$$\begin{tikzcd} \text{32 bits} \arrow[d] \\ \text{8 bits} \arrow[d] & \text{Loss of Precision} \\ \text{32 bits} & \text{Inference} \end{tikzcd}$$

2. Example you have 0.2345678~ 0.235
3. During Conversion you have a loss in precision

In conclusion, it leads to loss of precision in weights which leads to a drop in accuracy

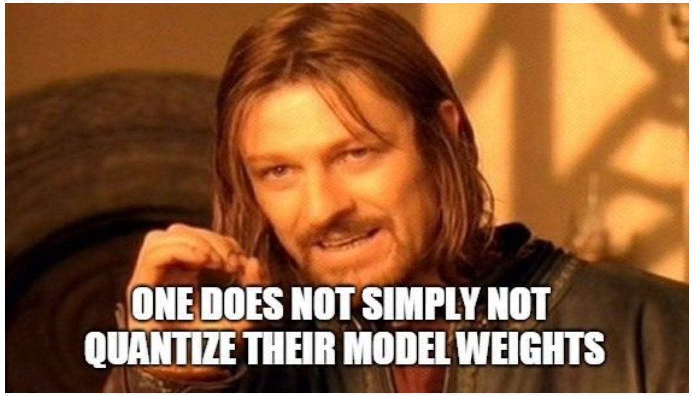
Post Training Integer Quantization

1. Integer quantization is a general technique that reduces the numerical precision of the **weights and activations of models** to reduce memory and improve latency
2. Differs from Hybrid operations, since they focused on weights and had certain parts which allowed computations to take part in floating points
3. Enables users to take an already-trained floating-point model and fully quantize it to only use 8-bit signed integers (i.e. `int8`).
4. TensorFlow Lite will execute the integer operations in the integer-only accelerator, falling back to CPU for the operations involving floating point. To execute entirely on specialized hardware that does not support floating point operations at all (for example, some machine learning accelerators, including the Edge TPU), you can specify a flag in order to output only integer operations

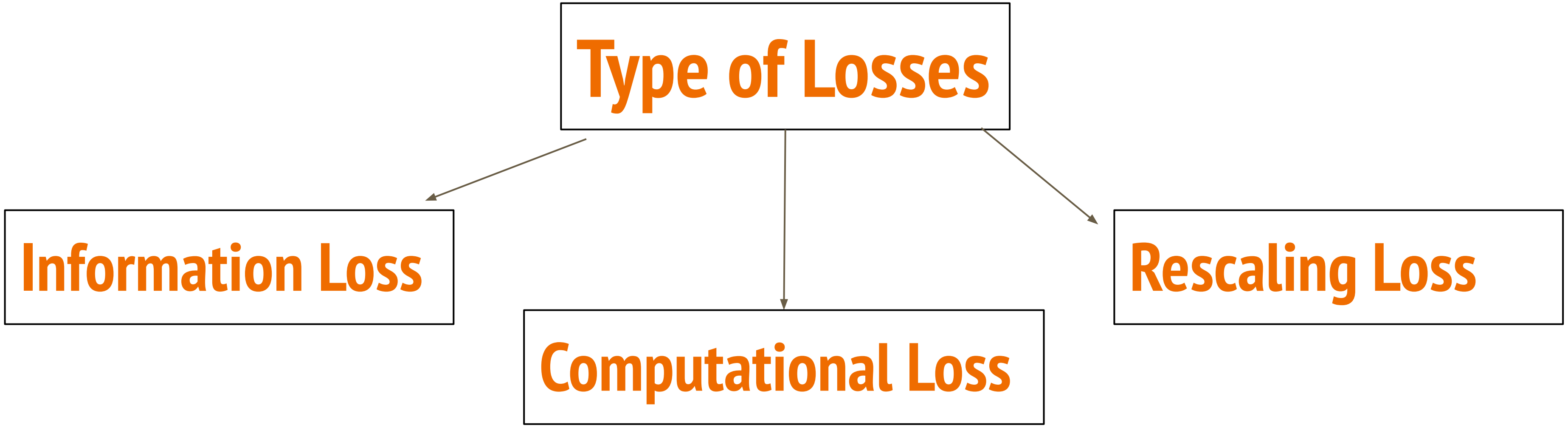
me watching my model's validation accuracy slowly drop over time

Accuracy Drop?

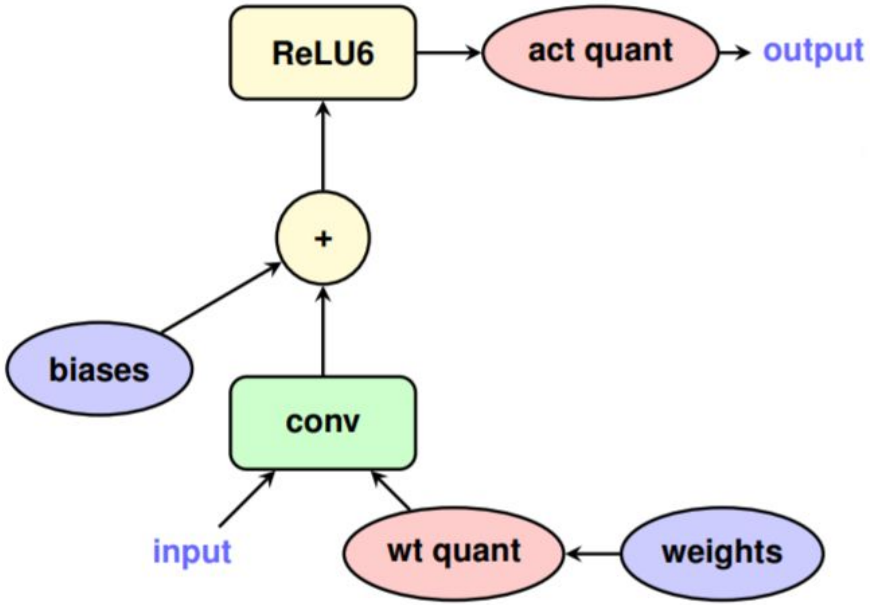
Quantization Aware Training~Mimicking Real World Quantization at Inference



1. Enables you to train and deploy models with the performance and size benefits of quantization, while retaining close to their original accuracy
2. Problems with quantization:
    a. process of going from higher to lower precision is lossy in nature.
    b. You are rounding off all values: For example, all values in range [2.0, 2.3] may now be represented in one single bucket. This is similar to rounding errors when fractional values are represented as integers.
    c. Accumulation of loss: Neural Networks consist of MAC operations, several multiply-add computations, these losses accumulate
    d. More computational error


Type of Losses
Information Loss
Computational Loss
Rescaling Loss

Process of QAT

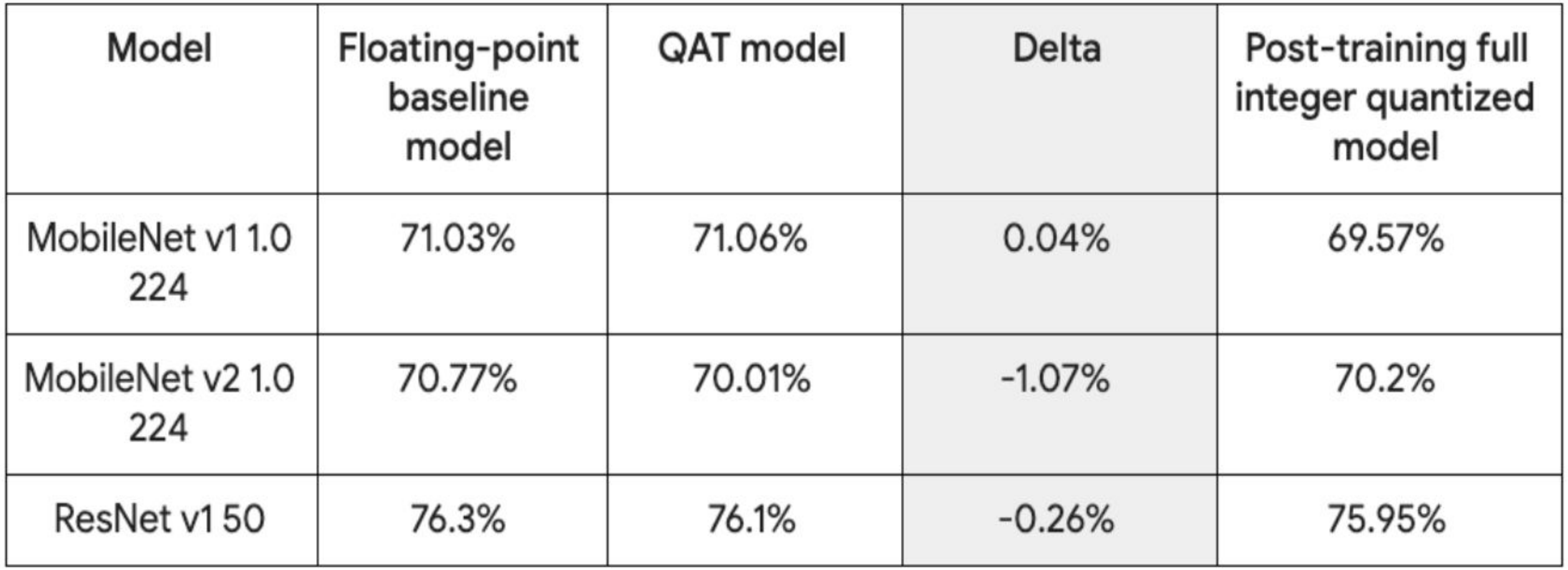


1. **Core idea** is that QAT simulates low-precision inference-time computation in the forward pass of the training process.
2. Introduction of quantisation noise: during the training and as part of the overall loss, which the optimization algorithm tries to minimize.Hence, the model learns parameters that are more robust to quantization.

Process Involved

a. Convert floating-point values to lower precision bits
b. Convert lower precision bits to higher bits again
c. Losses mimic low precision computations
d. Function Used: (tensorflow::ops::FakeQuantWithMinMaxVars)

Results



Model	Floating-point baseline model	QAT model	Delta	Post-training full integer quantized model
MobileNet v1 1.0 224	71.03%	71.06%	0.04%	69.57%
MobileNet v2 1.0 224	70.77%	70.01%	-1.07%	70.2%
ResNet v1 50	76.3%	76.1%	-0.26%	75.95%

Demo

Resources to Follow

1. Quantization: Pete Warden Blog
2. Talks: Pete Warden TinyML
3. Tensorflow Blogs on Optimisation Toolkit
4. Sayak Paul Blog
5. Tensorflow DevSummit Talk-Video

Contact Me

- **Feedback- QR Code**
- All would love to hear from you and your responses will be **Anonymous**!
- **Contact me:**
- Archana and Soham Chatterjee
  - varchanaiyer139@gmail.com