http://inst.eecs.berkeley.edu/~ee290-2/sp21/

Hardware for Machine Learning

Lecture 4: Quantization

Sophia Shao

NVIDIA Ampere Architecture

Third-generation Tensor Cores:

- Acceleration for all data types including FP16, BF16, TF32, FP64, INT8, INT4, and Binary.
- New Tensor Core sparsity feature exploits fine-grained structured sparsity in deep learning networks, doubling the performance of standard Tensor Core operations.

https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

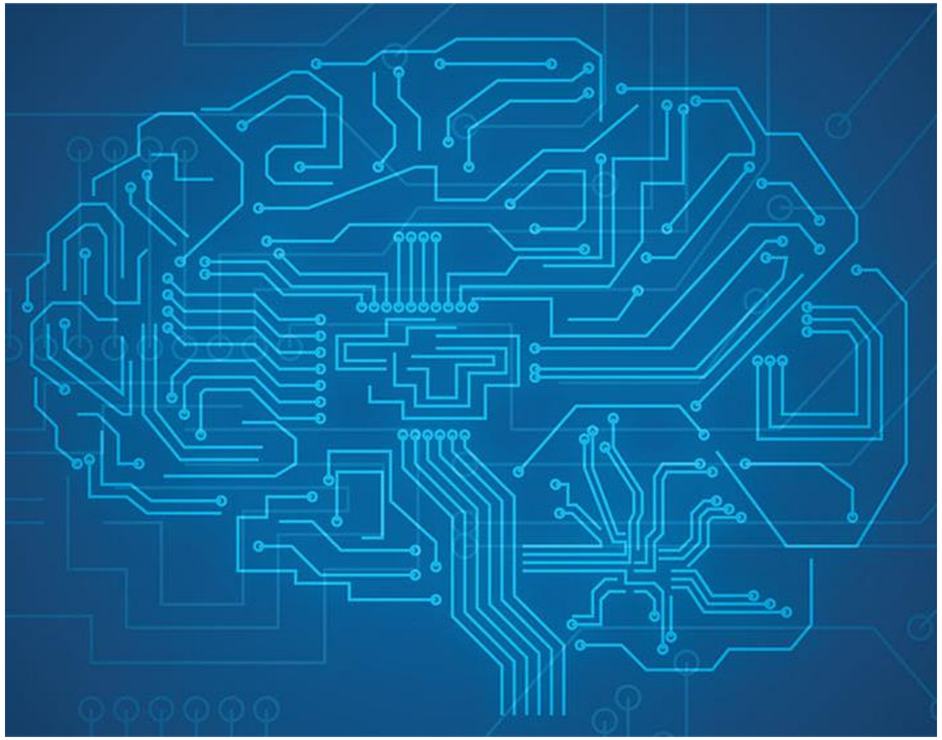
Table 2. A100 speedup over V100 (TC=Tensor Core, GPUs at respective clock speeds)

	V100	A100	A100 Sparsity[1]	A100 Speedup	A100 Speedup with Sparsity
A100 FP16 vs V100 FP16	31.4 TFLOPS	78 TFLOPS	NA	2.5x	NA
A100 FP16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 BF16 TC vs V100 FP16 TC	125 TFLOPS	312 TFLOPS	624 TFLOPS	2.5x	5x
A100 FP32 vs V100 FP32	15.7 TFLOPS	19.5 TFLOPS	NA	1.25x	NA
A100 TF32 TC vs V100 FP32	15.7 TFLOPS	156 TFLOPS	312 TFLOPS	10x	20x
A100 FP64 vs V100 FP64	7.8 TFLOPS	9.7 TFLOPS	NA	1.25x	NA
A100 FP64 TC vs V100 FP64	7.8 TFLOPS	19.5 TFLOPS	NA	2.5x	NA
A100 INT8 TC vs V100 INT8	62 TOPS	624 TOPS	1248 TOPS	10x	20x
A100 INT4 TC	NA	1248 TOPS	2496 TOPS	NA	NA
A100 Binary TC	NA	4992 TOPS	NA	NA	NA

1 - Effective TOPS / TFLOPS using the new Sparsity Feature

Review

- Artificial intelligence, machine learning, and deep learning
- Building a machine learning algorithm:
  - Dataset
  - Cost function
  - Optimization function
  - Model
- Deep learning to automatically extract hierarchical data
  - Better at handle high-dimensional data
  - Key differences are in the Model
    - Multiple layers, i.e., deep



Quantization

- Floating-Point Arithmetic
- Fixed-Point Arithmetic
- Hardware Implications
- DNN Quantization

Scientific Notation (in Decimal)

mantissa **exponent**

$$6.02_{10} \times 10^{23}$$

decimal point **radix (base)**

- Normalized form: no leadings 0s (exactly one digit to left of decimal point)
- Alternatives to representing 1/1,000,000,000
  - Normalized: 1.0×10^{-9}
  - Not normalized: 0.1×10^{-8}, 10.0×10^{-10}

Scientific Notation (in Binary)

mantissa **exponent**

$$1.01_{two} \times 2^{-1}$$

“binary point” **radix (base)**

- Computer arithmetic that supports it called floating point, because it represents numbers where the binary point is not fixed, as it is for integers

Floating-Point Representation

- Normal format: $+1.\text{xxx}\ldots\text{x}_{\text{two}} * 2^{\text{yyy}\ldots\text{y}_{\text{two}}}$

- **S represents Sign**
- **Exponent represents y's**
- **Significand represents x's**
- **Represent numbers as small as 2.0×10^{-38} to as large as 2.0×10^{38}**

Floating-Point Representation (fp32)

- IEEE 754 Floating Point Standard
  - Called **Biased Notation**, where bias is number subtracted to get real number
  - IEEE 754 uses bias of 127 for single prec.
  - Subtract 127 from Exponent field to get actual value for exponent
  - 1023 is bias for double precision

- Summary (single precision, or fp32):

31	30 … 23	22 … 0
S	Exponent	Significand
1 bit	8 bits	23 bits

- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

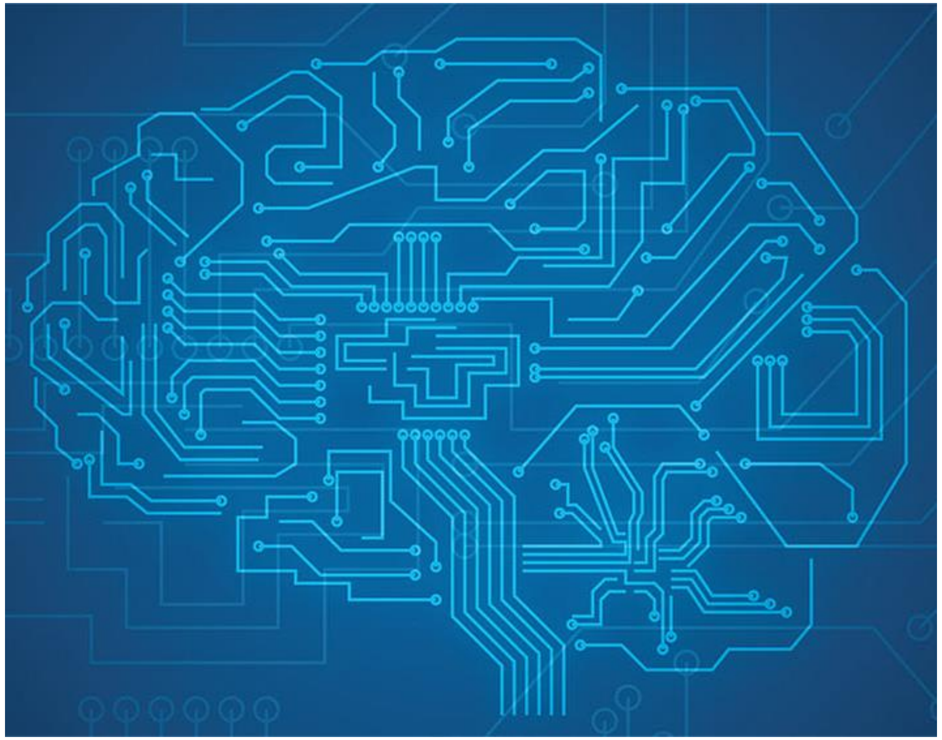
Floating-Point Representation (fp16)

- IEEE 754 Floating Point Standard
  - Called **Biased Notation**, where bias is number subtracted to get real number
  - IEEE 754 uses bias of 15 for half prec.
  - Subtract 15 from Exponent field to get actual value for exponent

- Summary (half precision, or fp15):

15	15 10	9 0
S	Exponent	Significand
1 bit	5 bits	10 bits

- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-15)}$



Quantization

- Floating-Point Arithmetic
- Fixed-Point Arithmetic
- Hardware Implications
- DNN Quantization

Fixed-Point Arithmetic

- Integers with a binary point and a bias
  - “slope and bias”: y = s*x + z
  - Qm.n: m (# of integer bits) n (# of fractional bits)

s = 1, z = 0

2^2	2^1	2^0	Val
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**

s = 1/4, z = 0

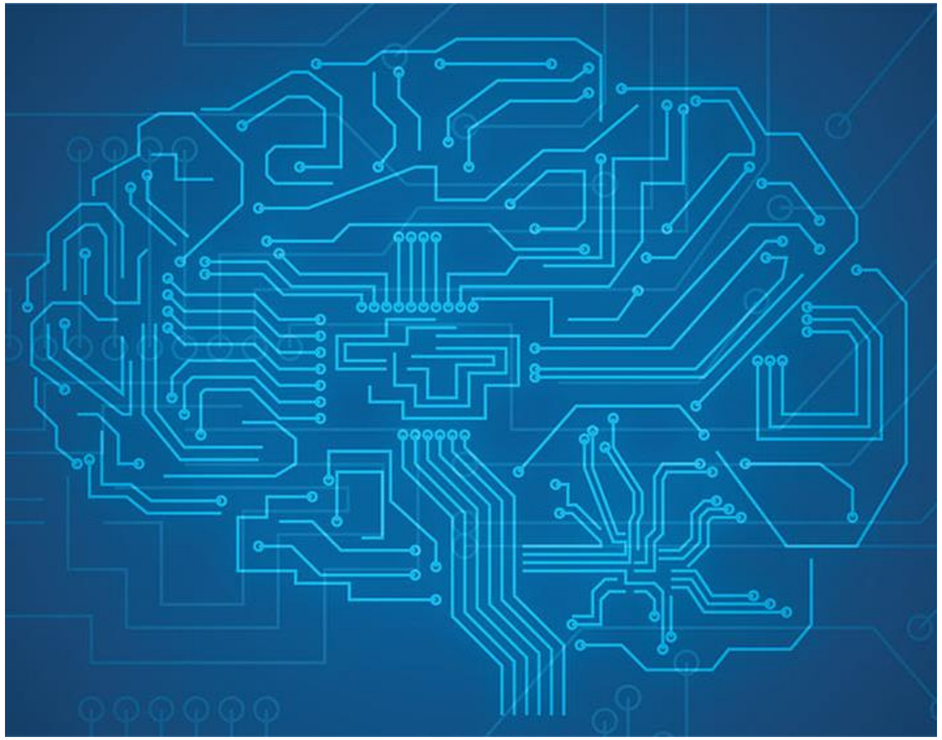
2^0	2^-1	2^-2	Val
0	0	0	**0**
0	0	1	**1/4**
0	1	0	**2/4**
0	1	1	**3/4**
1	0	0	**1**
1	0	1	**5/4**
1	1	0	**6/4**
1	1	1	**7/4**

s = 4, z = 0

2^4	2^3	2^2	Val
0	0	0	**0**
0	0	1	**4**
0	1	0	**8**
0	1	1	**12**
1	0	0	**16**
1	0	1	**20**
1	1	0	**24**
1	1	1	**28**

s = 1.5, z =10

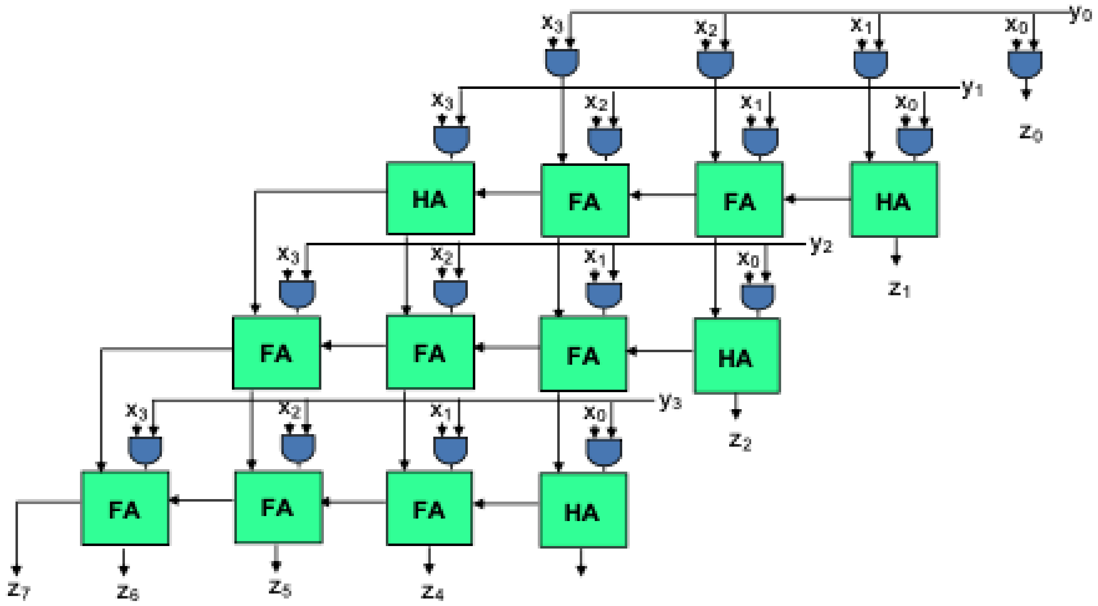
2^2	2^1	2^0	Val
0	0	0	**1.5*0 +10**
0	0	1	**1.5*1 +10**
0	1	0	**1.5*2 +10**
0	1	1	**1.5*3 +10**
1	0	0	**1.5*4 +10**
1	0	1	**1.5*5 +10**
1	1	0	**1.5*6 +10**
1	1	1	**1.5*7 +10**



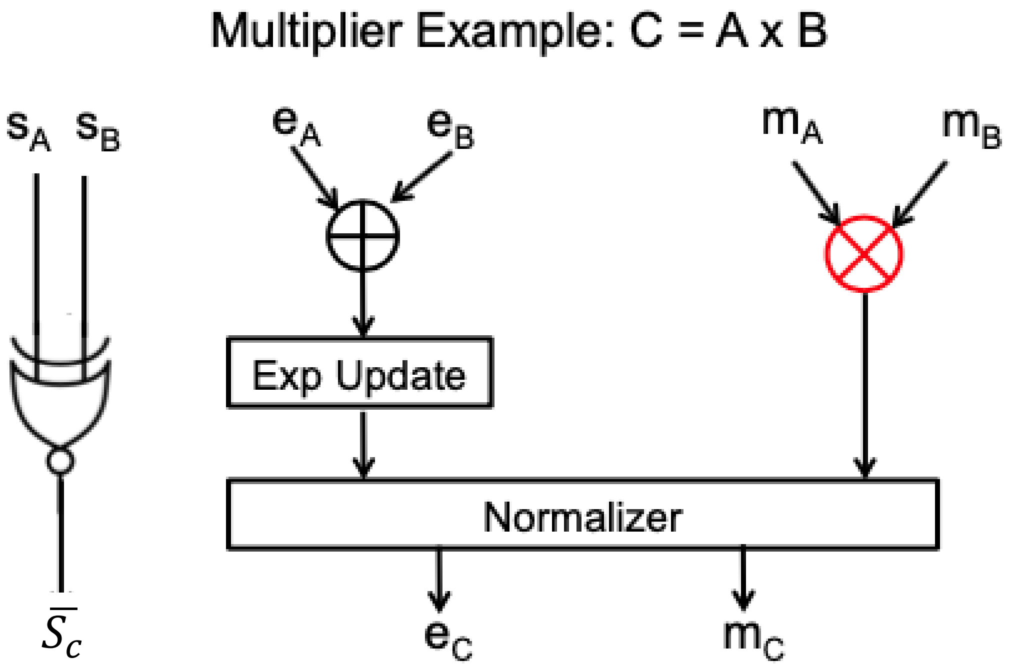
Quantization

- Floating-Point Arithmetic
- Fixed-Point Arithmetic
- Hardware Implications
- DNN Quantization

Hardware Implications



Fixed-Point Multiplier

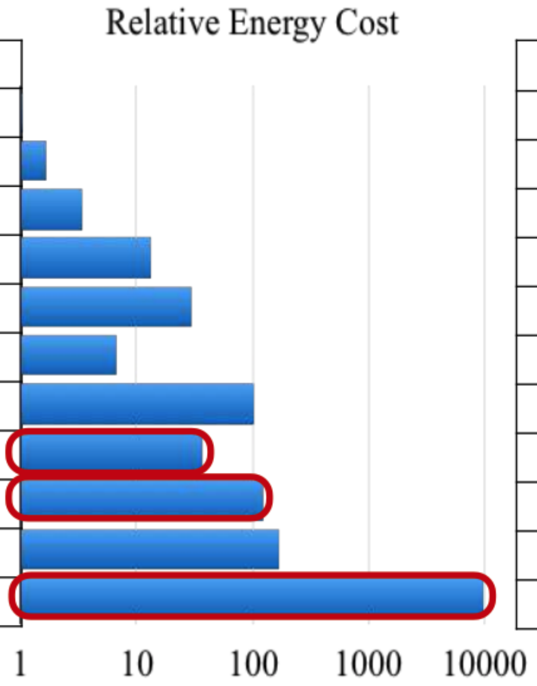


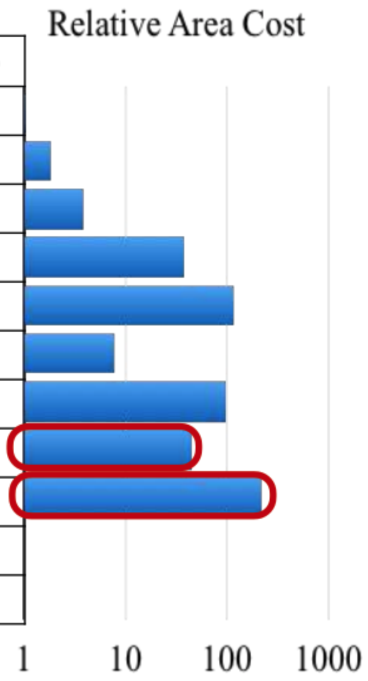
Floating-Point Multiplier

Hardware Implications

- "Rough Energy Numbers (45nm)" from "computing's Energy Problem, M. Horowitz, ISSCC, 2014

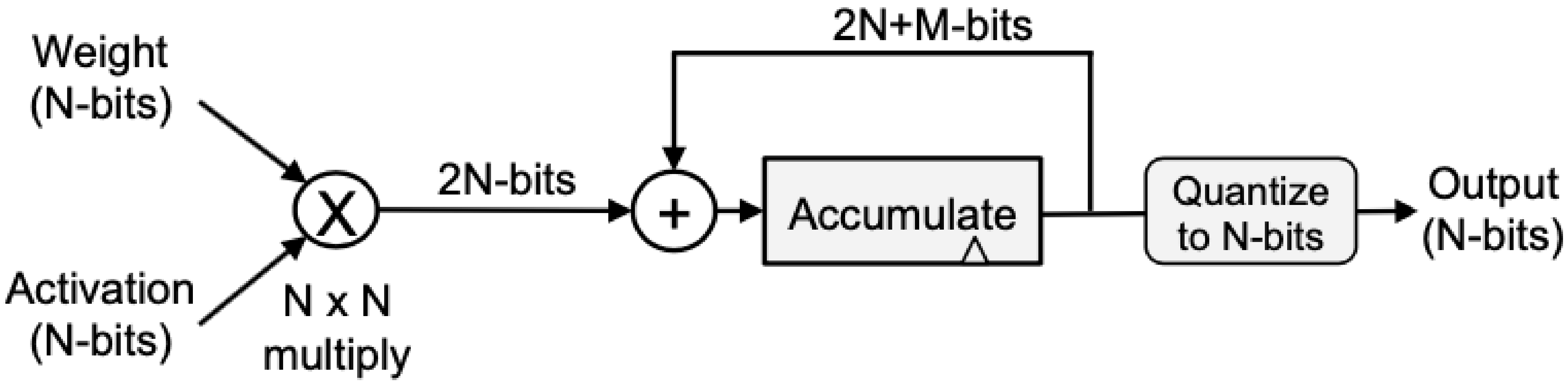
Operation:	Energy (pJ)	Area (µm²)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





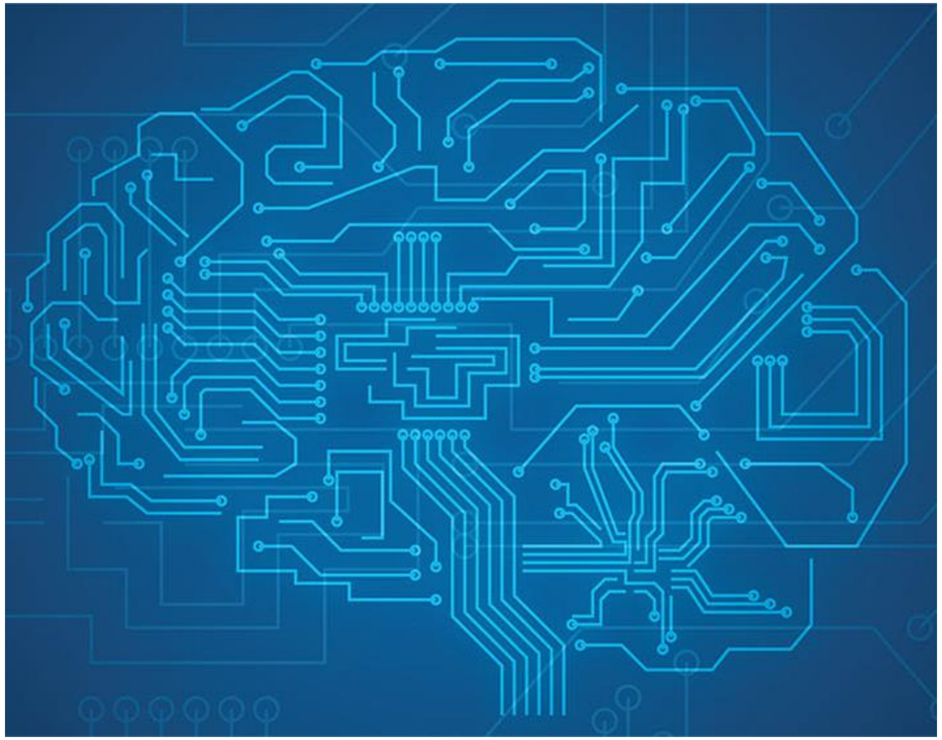
MAC Precision

- Accumulation requires higher precision than inputs.



Administrivia

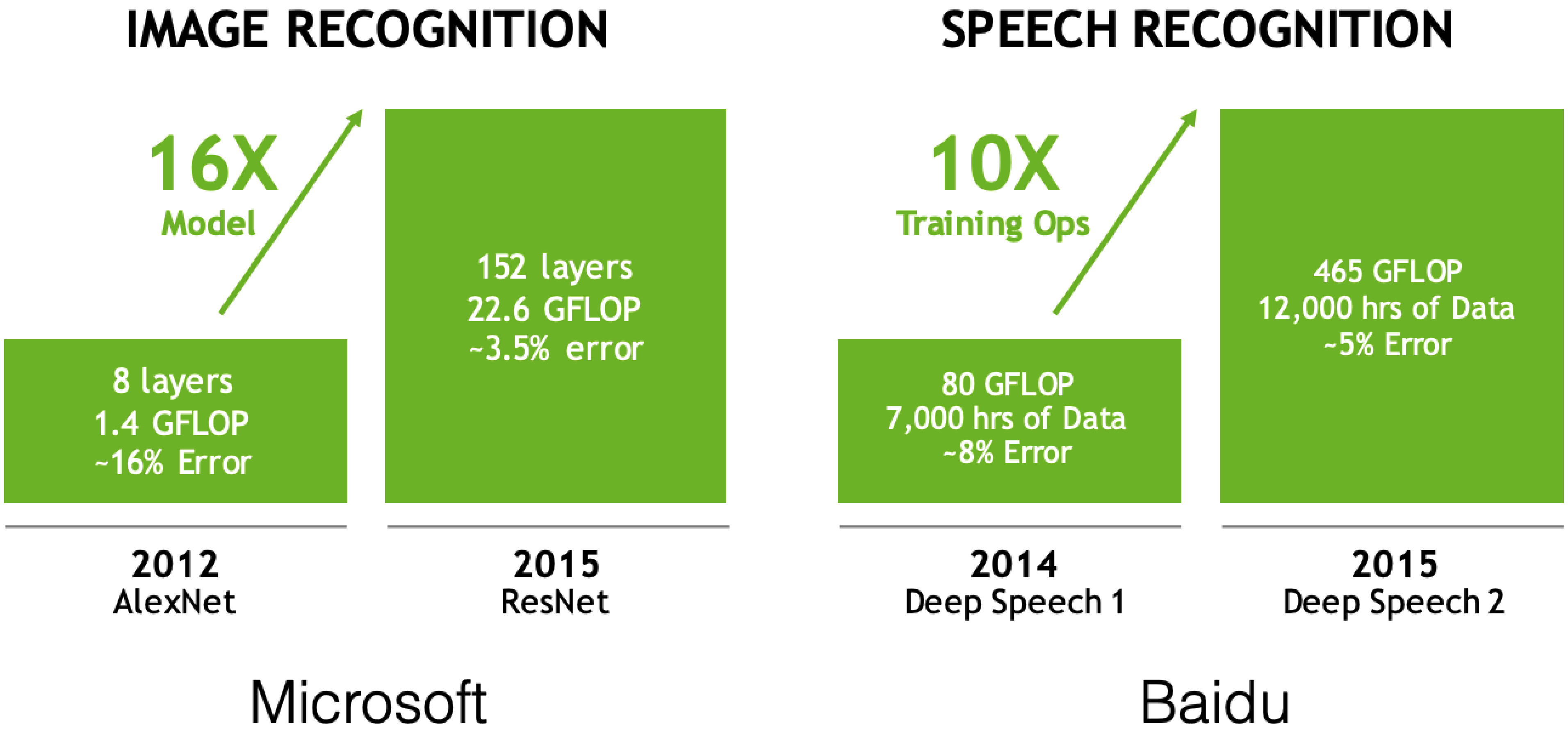
- Lab 1 posted!
  - Due 2/5
  - Start early.
- Paper readying posted!
  - Due Wednesdays.
  - Will post every Wednesday as well.
- Guest lectures:
  - 2/10 AWS Guest lecture on Inferentia from Randy Huang and Ron Diamant
  - 2/17 Guest lecture on Advanced Quantization from Amir Gholami



Quantization

- Floating-Point Arithmetic
- Fixed-Point Arithmetic
- Hardware Implications
- DNN Quantization

Models are getting larger…



Dally, NIPS'2016 workshop on Efficient Methods for Deep Neural Networks

Quantization

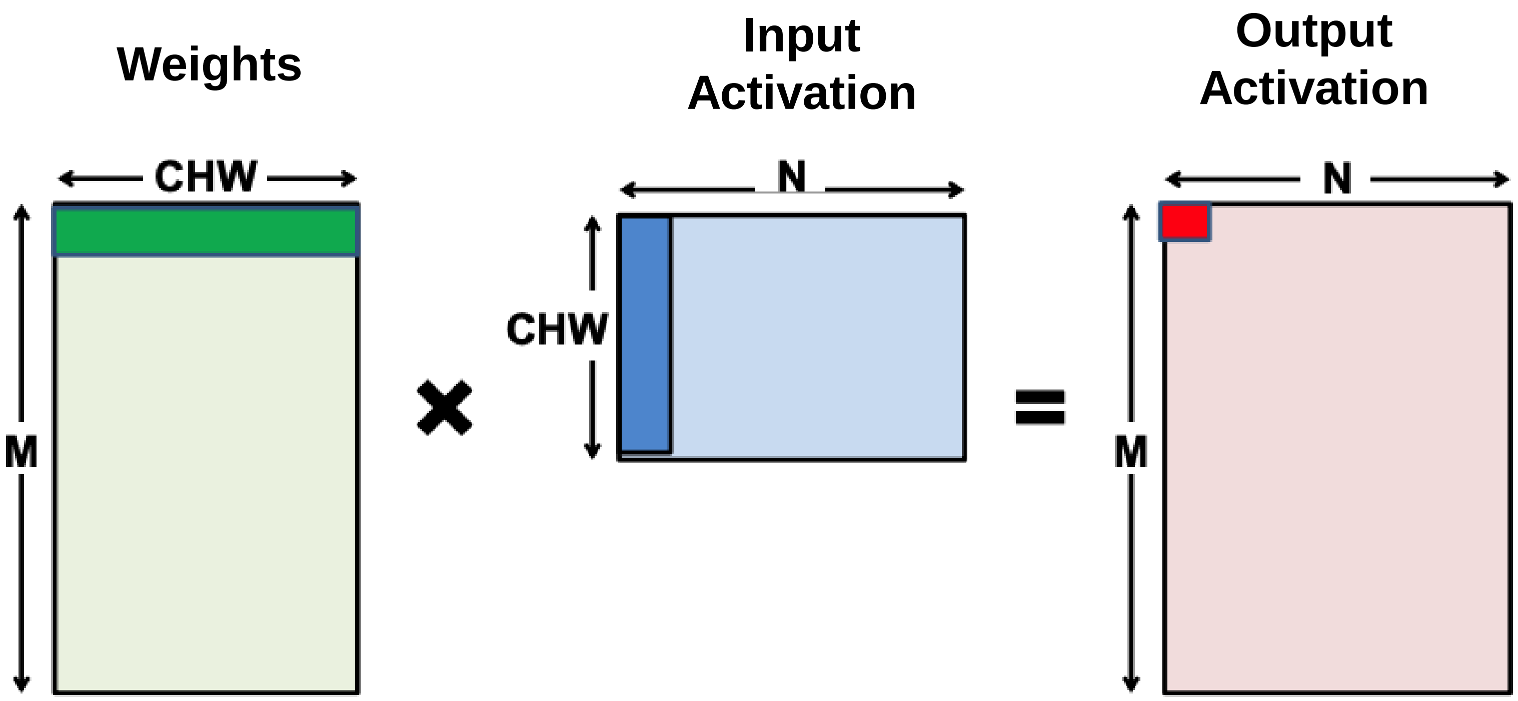
- Quantization scheme:
  - The correspondence between the fixed-point representation of values, i.e., **"q" for "quantized value"** and their floating-point value, i.e., **"r" for "real value".**
  - Recall "slope and bias" of fixed-point representation: y = s*x + z

$$r = S(q - Z)$$

```
template<typename QType>  // e.g. QType=uint8
struct QuantizedBuffer {
  vector<QType> q;        // the quantized values
  float S;                // the scale
  QType Z;                // the zero-point
};
```

r: real floating-point value
q: quantized fixed-point value
S: scaling factor
Z: zero point (bias)

MAC in DNNs



$$\boldsymbol{OA[i,k]} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{W[i,j]} * \boldsymbol{IA[j,k]})$$

$$OA[0,0] = \sum_{j=1}^{CHW} (W[0,j] * IA[j,0])$$

Eyeriss tutorial

Quantization Process

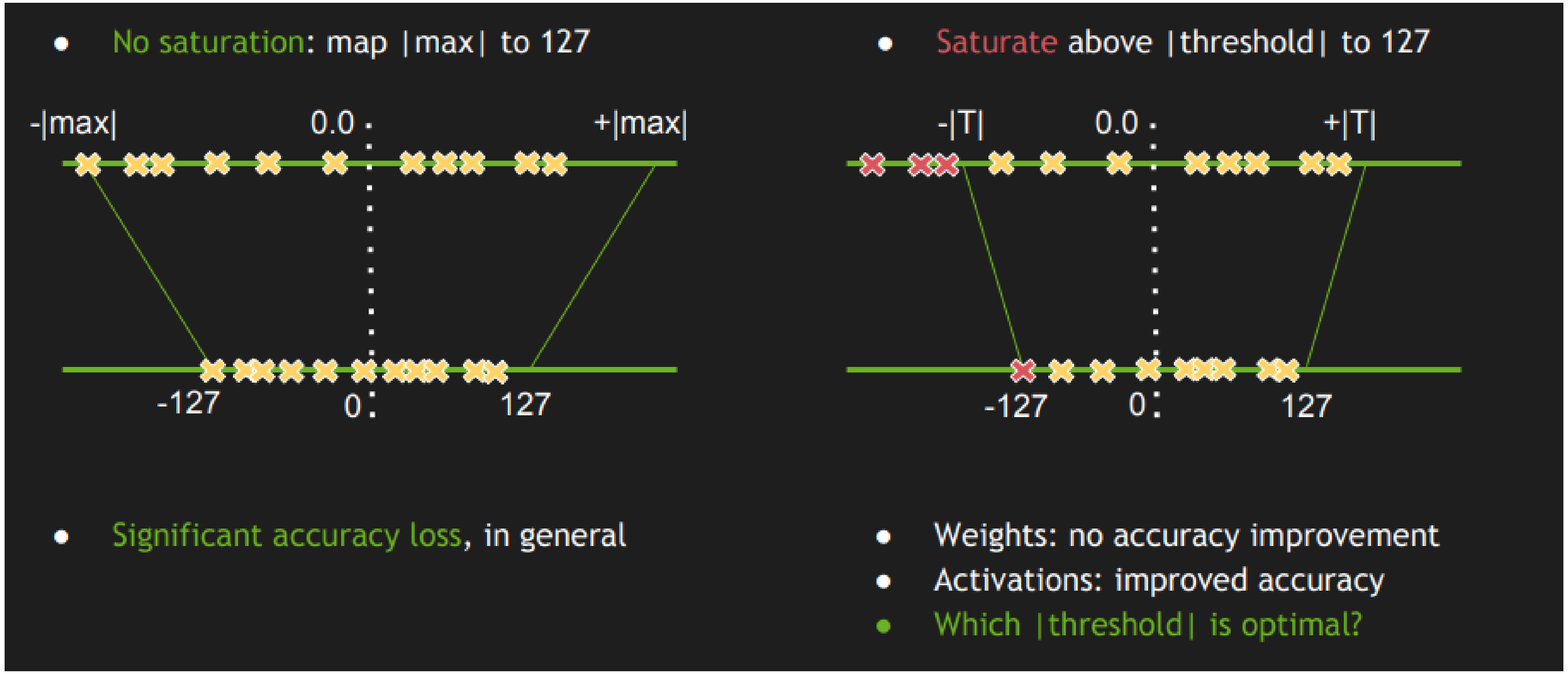
$$\boldsymbol{OA[i,k]} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{W[i,j]} * \boldsymbol{IA[j,k]}) \qquad r = S(q - Z)$$

$$S_{OA}\left(q_{OA}^{(i,k)} - Z_{OA}\right) = \sum_{j=1}^{N} (S_W\left(q_W^{(i,j)} - Z_W\right) * (S_{IA}(q_{IA}^{(j,k)} - Z_{IA})))$$

$$q_{OA}^{(i,k)} = Z_{OA} + \frac{S_W * S_{IA}}{S_{OA}} \sum_{j=1}^{N} \left((q_W^{(i,j)} - Z_W) * (q_{IA}^{(j,k)} - Z_{IA})\right)$$

$$q_{OA}^{(i,k)} = \frac{S_W * S_{IA}}{S_{OA}} \sum_{j=1}^{N} \left(q_W^{(i,j)} * q_{IA}^{(j,k)}\right)$$

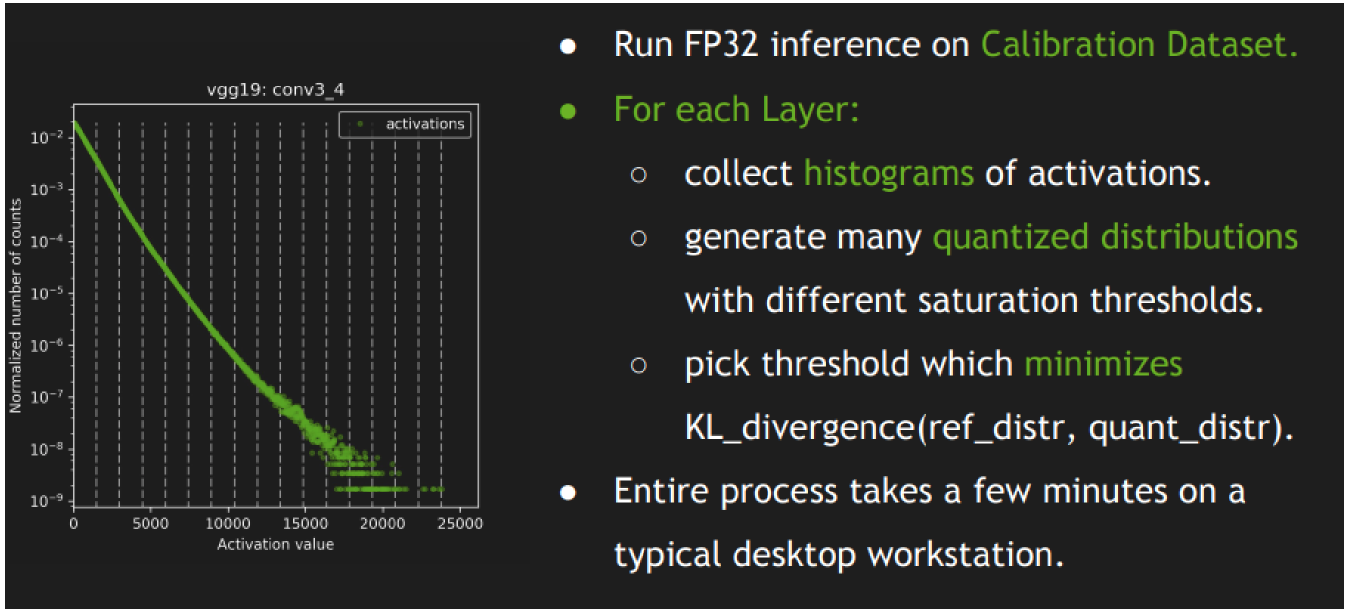
How to set scale factor?



8-bit Inference with TensorRT

Which threshold is optimal?

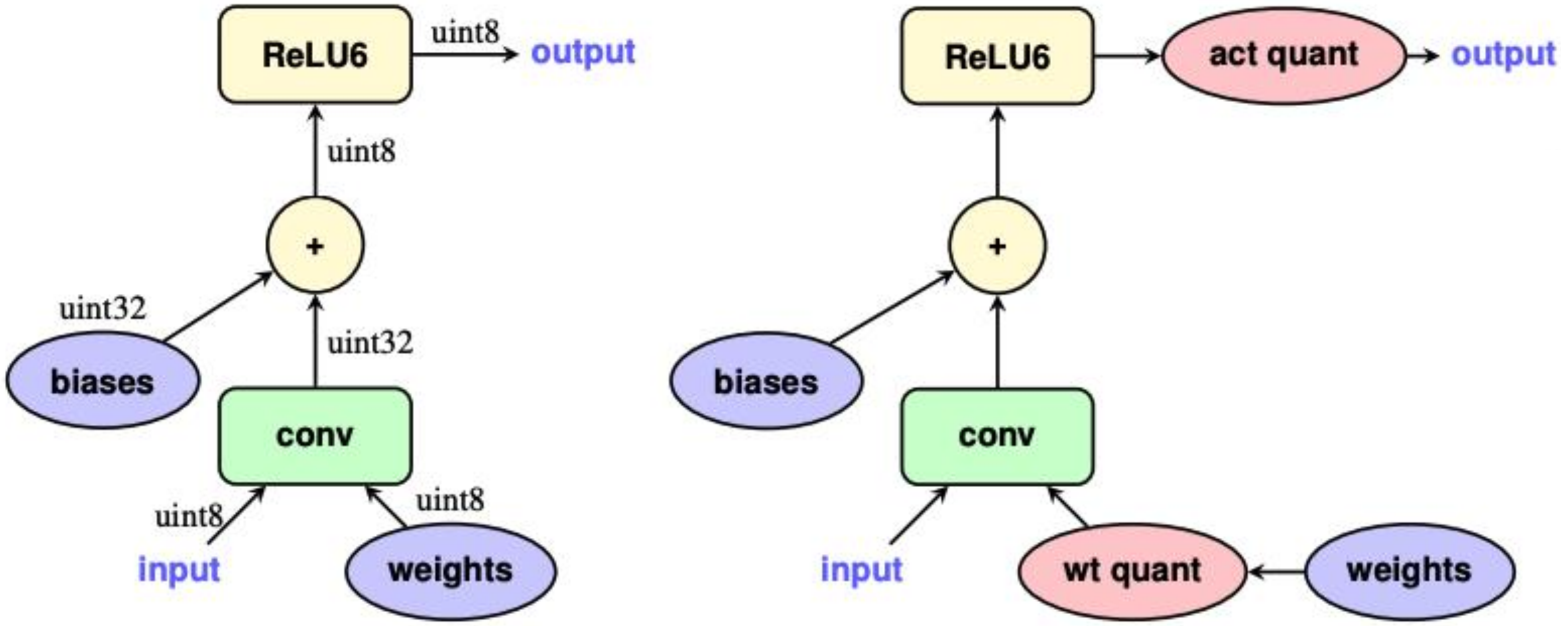
- Again, it depends.
  - A lot of research/engineering practices in this area, e.g.,
  - Averaging: using a (weighted) average of min/max values of samples in the batch instead of the global min/max
  - Mean +/- N *std: Take N standard deviation from the mean
  - Calibration (TensorRT)



8-bit Inference with TensorRT

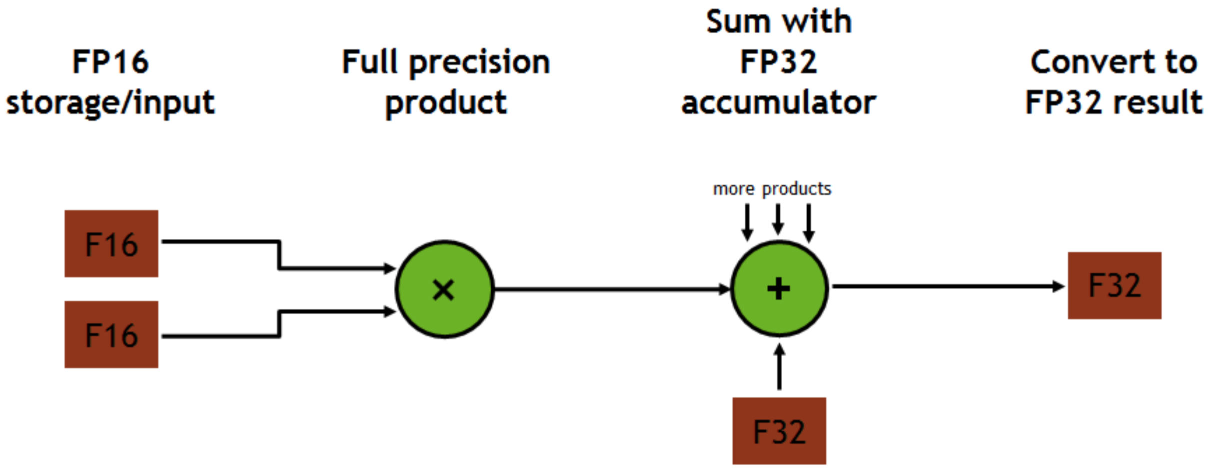
Quantization-Aware Training

- Typically performs better than post-training quantization
- “Simulate” quantization effects in the forward pass
- Weights and biases are updated in floating point during backpropagation so that they can be nudged by small amounts.



(a) Integer-arithmetic-only inference
(b) Training with simulated quantization

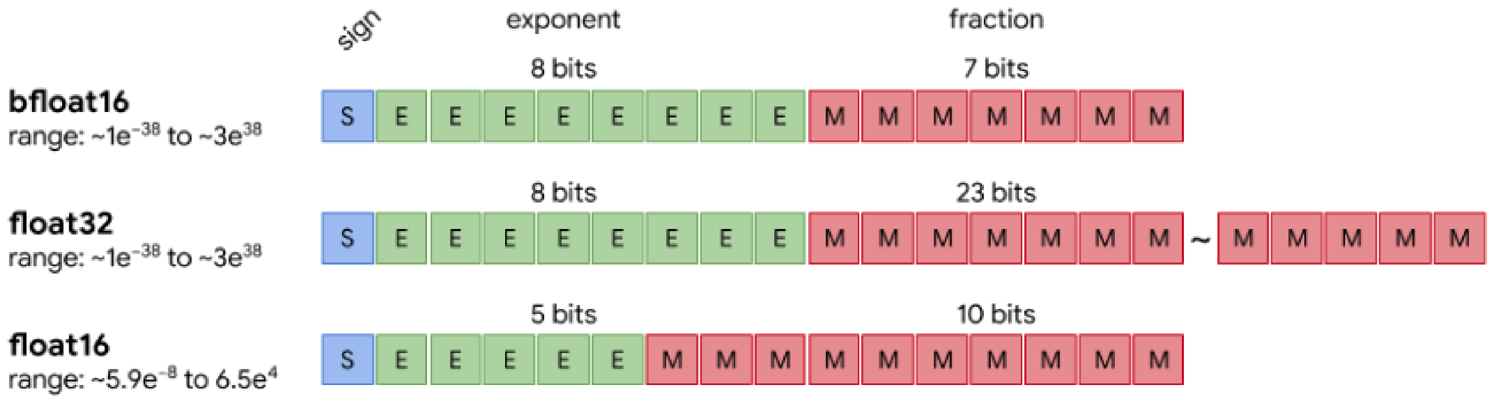
GPU Tensor Cores



- https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/
  - Used by cuDNN and CUBLAS libraries
  - Exposed in CUDA as WMMA
    - https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#wmma
- Accelerate convolutions and matrix multiplication
  - A single instruction multiply-accumulates matrices
  - Computes many dot-products in parallel

Bfloat16 for Google's Tensor Processing Unit

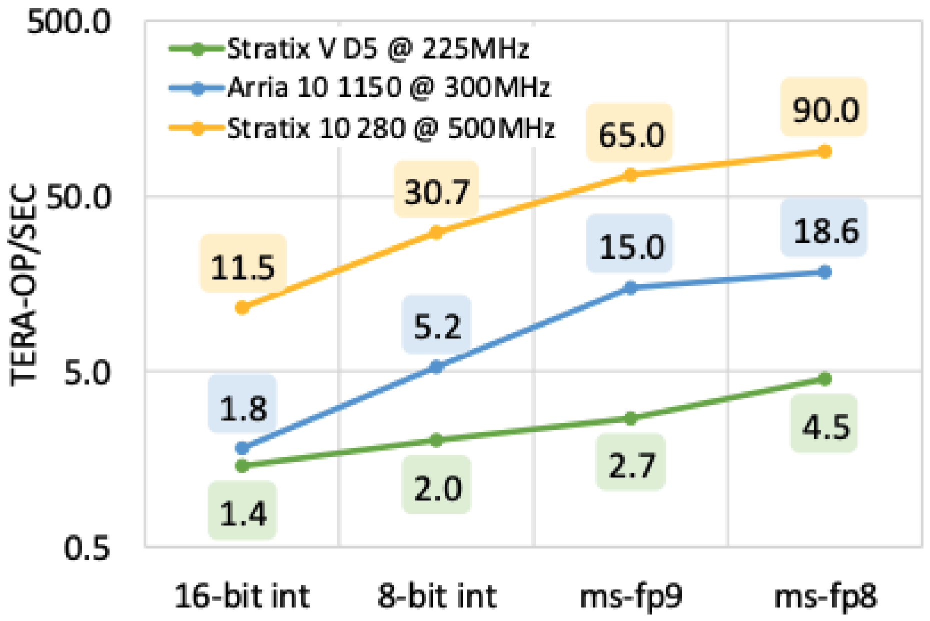
- fp32 - IEEE single-precision floating-point
- fp16 - IEEE half-precision floating point
- bfloat16 - 16-bit *brain floating point*



https://cloud.google.com/tpu/docs/bfloat16

MS-FP in Brainwave FPGA @ Microsoft

- “ ‘neural’-optimized data formats based on 8- and 9-bit floating point, where mantissas are trimmed to 2 or 3 bits. “
- “ These formats, referred to as ms-fp8 and ms-fp9, exploit efficient packing into reconfigurable resources and are comparable in FPGA area"



Serving DNNs in Real Time at Datacenter Scale with Project Brainwave

Review

- AlexNet's cost function and optimization function
- Floating-point and fixed-point representations
- Hardware implications:
  - Fewer # of bits -> Energy/storage efficiency
- DNN Quantization
  - Using the "slope and bias" of fixed-point representation: y = s*x + z
    - Scaling factor
      - How to scale? How to choose threshold value?
    - Zero point
  - Post-training quantization vs Quantization-aware training
  - State-of-the-art hardware support for low-precision DNNs