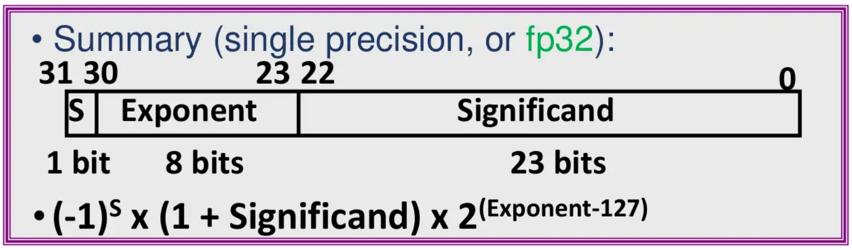
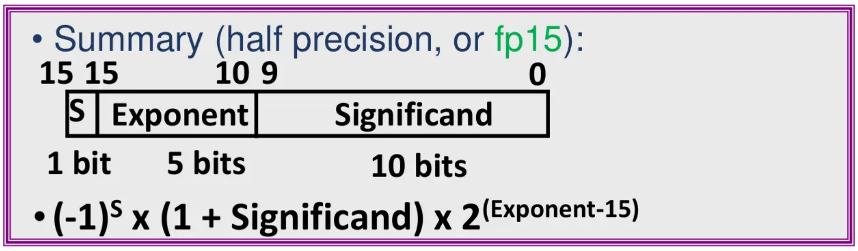
IEEE 754 Floating Point Standard





Fixed-Point

- Qm.n: m (# of integer bits) n (# of fractional bits)

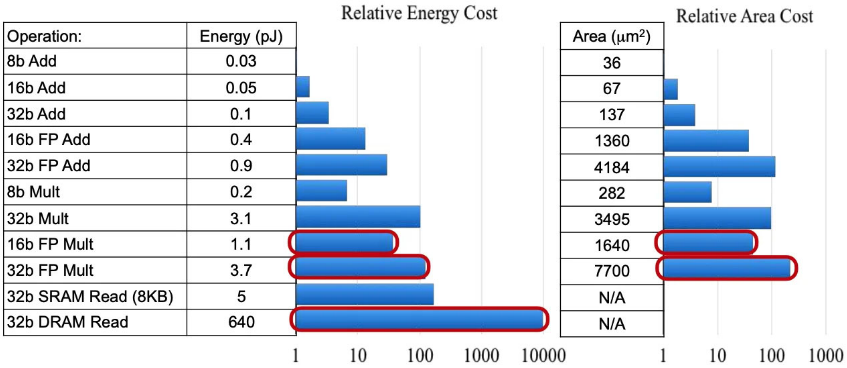
Table 2. A100 speedup over V100 (TC=Tensor Core, GPUs at respective clock speeds)

	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

Hardware Multiplier

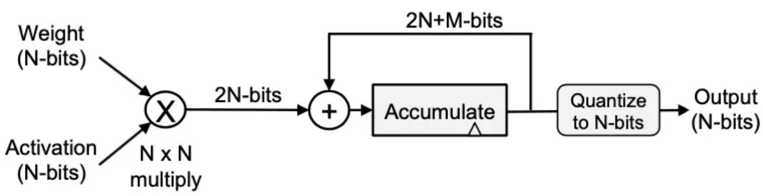
- Fixed-Point Multiplier
- Floating-Point Multiplier



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

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

- accumulator needs more bits



Quantization

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

- r : real value `float`
- S : scaling `float`
- q : quantized value `int8`
- Z : bias `int8`

matrix multiplication

- input `batch_size*c_in` $\times$ weight `c_in*c_out` $\rightarrow$ output `batch_size * c_out`

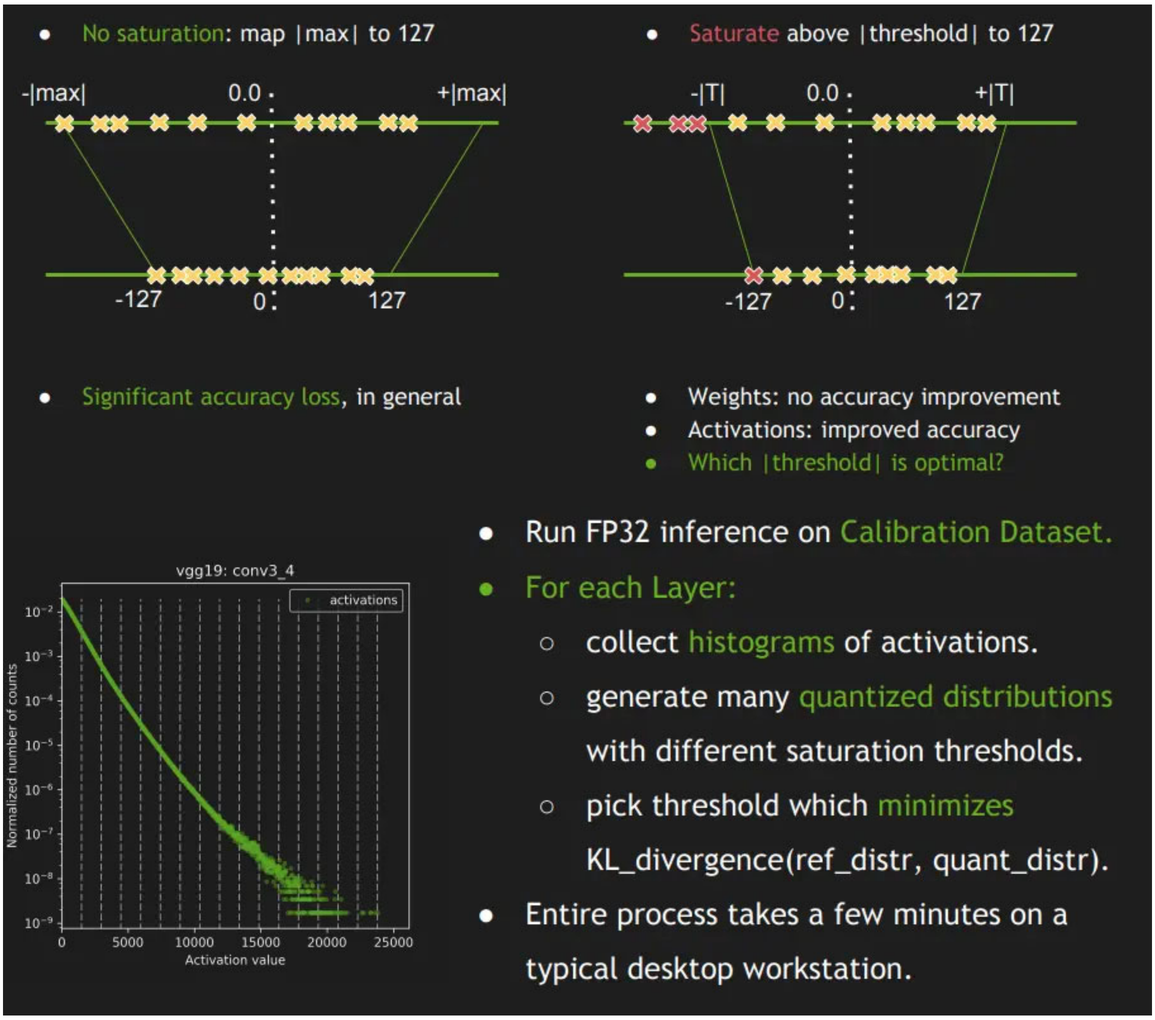
$$\text{output}[i,k] = \sum \text{input}[i,:] \times \text{weight}[:,k]$$

after quantization

$$\color{red}{Q_{output}} \color{black}{= Z_{output} + \frac{S_{input}S_{weight}}{S_{output}}(}\color{red}{Q_{input}}\color{black}{- Z_{input}) \times (}\color{red}{Q_{weight}}\color{black}{- Z_{weight})}$$

$Q_{output}, Q_{input}, Q_{weight}$: matrix

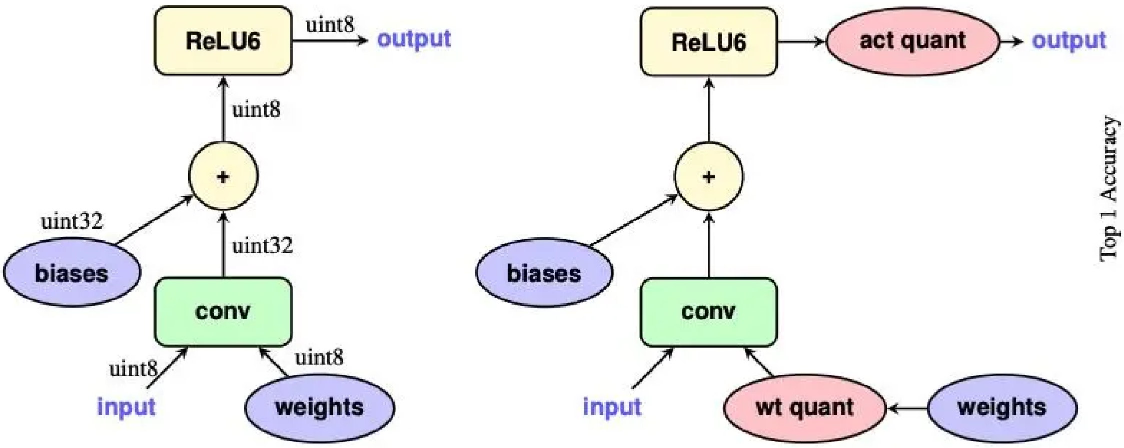
scaling factor



$$KL(p\|q) = -\int p(x)\log q(x)dx - \left(-\int p(x)\log p(x)dx\right) = \int p(x)\log\left[\frac{p(x)}{q(x)}\right]dx$$

- 用概率分布 q 拟合 p
- ≥ 0

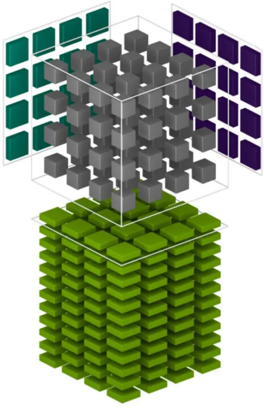
Quantization-Aware Training

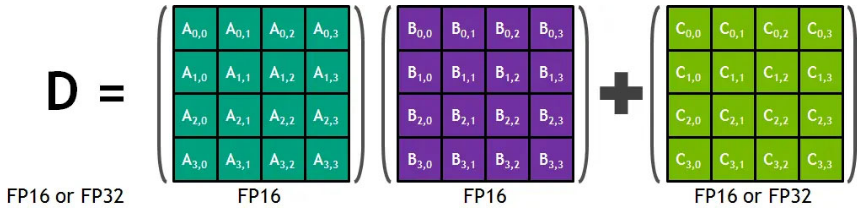


(a) Integer-arithmetic-only inference (b) Training with simulated quantization

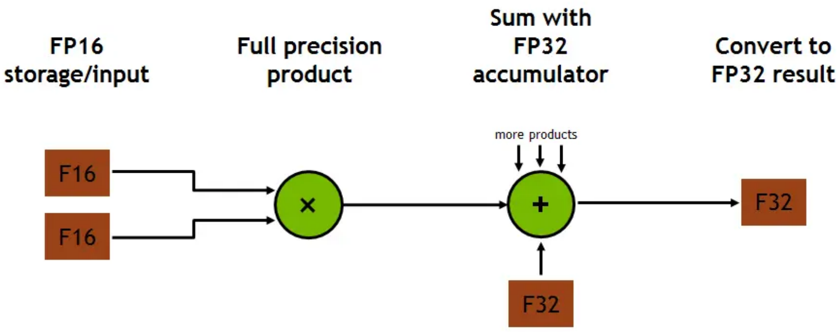
- 在训练时模拟量化
- 精度使用 float, 取值范围使用 int8

NVIDIA Tensor Core



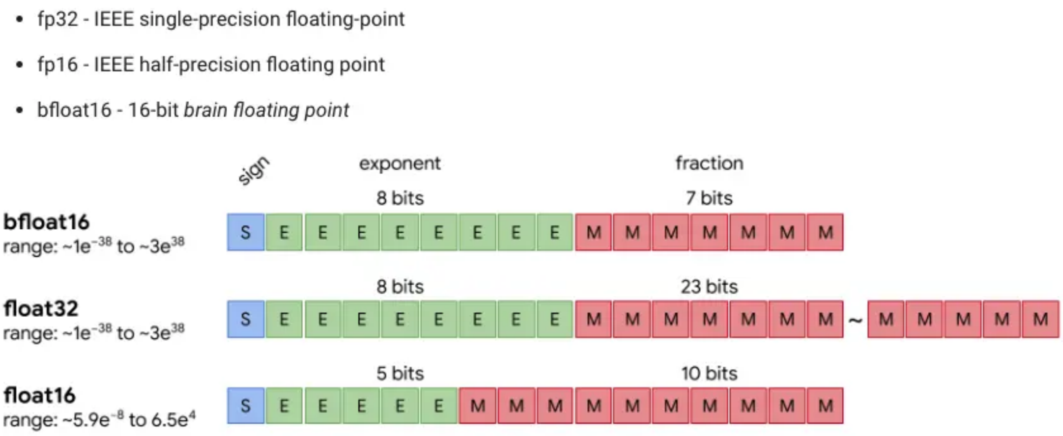


- Each Tensor Core performs 64 floating point FMA mixed-precision operations per clock



- mixed-precision GEMM

Bfloat16



DNN Kernels

convolution

- **input:** $N, C_{in}, H_{in}, W_{in}$
- **kernel:** $C_{out}, C_{in}, H_k, W_k$
- **output:** $N, C_{out}, H_{out}, W_{out}$

```
# padding=0, stride=1
for n in range(batch_size):
    for c_out in range(C_out):
        for h_out in range(H_out):
            for w_out in range(W_out):
                # per output element
                output[n,c_out,h_out,w_out] = 0
                for h_k in range(H_k):
                    for w_k in range(W_k):
                        for c_in in range(C_in):
                            output[n,c_out,h_out,w_out] += input[n,c_in,h_out+h_k,w_
                                                          kernel[c_out,c_in,h_k,w_k]
```

im2col

- **input:** $(\mathrm{N},\ \mathrm{H_{in}W_{in}},\ \mathrm{C_{in}H_kW_k})$
- **kernel:** $(\mathrm{C_{out}},\ \mathrm{C_{in}H_kW_k})$
- 大量元素重复

Winograd

**Fast Algorithms for Convolutional Neural Networks*

一维卷积

$[d_0\ d_1\ d_2\ d_3] \otimes [g_0\ g_1\ g_2]$

$$F(2,3) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

6次乘法，4次加法

$$F(2,3) = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

$$m_1 = (d_0 - d_2)g_0 \qquad m_2 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2}$$

$$m_4 = (d_1 - d_3)g_2 \qquad m_3 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2}$$

4次乘法，4次加法

- Input transform
- Filter transform
- Hadamar product
- Output transform

二维卷积

一维基础上再一次嵌套，降低一半以上乘法
https://www.cnblogs.com/shine-lee/p/10906535.html

$$Y = A^T \left[(GgG^T) \cdot (B^T dB) \right] A$$

$F(2 \times 2, 3 \times 3)$: output 2x2, kernel 3x3

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \quad B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

- 大feature map: 分割成小的
- 适用于小卷积核，且步长为1
- 需要额外的转换和存储

BatchNorm

Attention