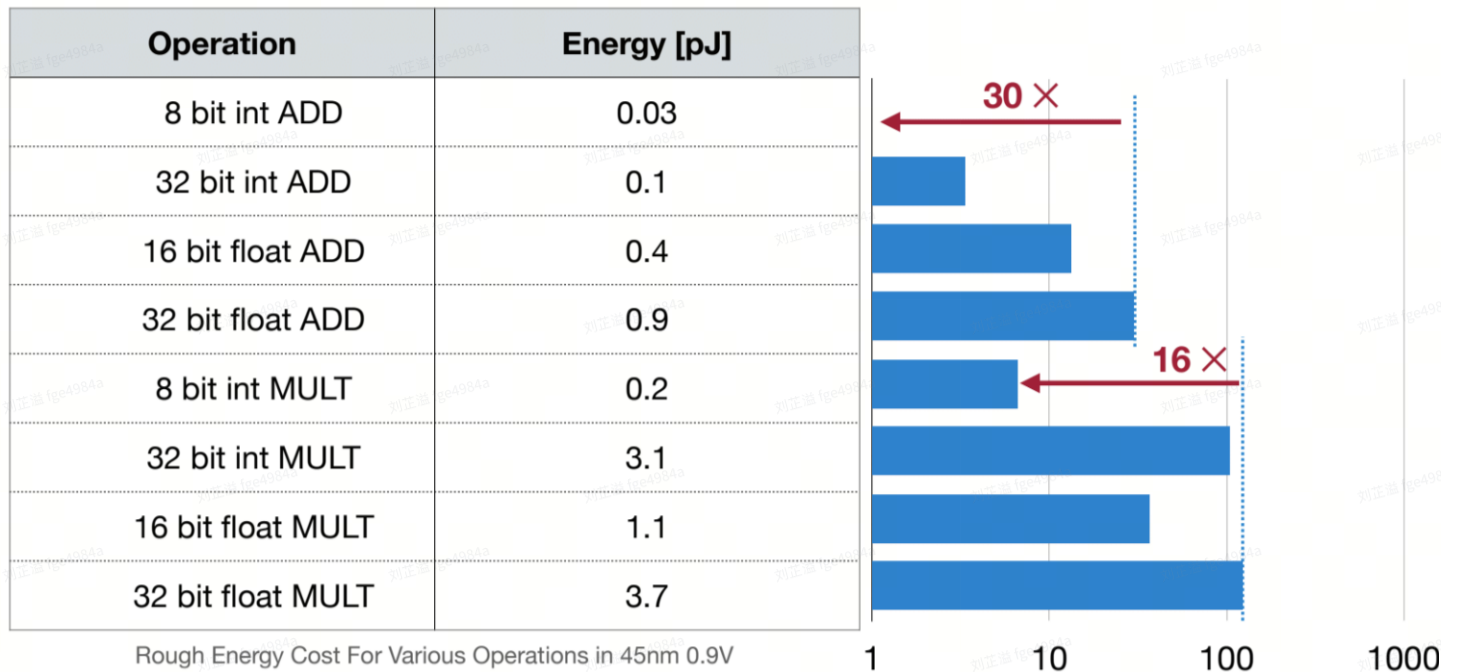


# Quantization

- 减小存放权重的位数
- 剪枝+量化可以极大地减小权重矩阵W的内存占用
- 更少的比特宽度有利于减小计算资源的消耗



## Numeric Data Types

- Integer
  - Unsigned Integer
    - $n$ -bit Range:  $[0, 2^n - 1]$
  - Signed Integer
    - Sign-Magnitude Representation
      - $n$ -bit Range:  $[-2^{n-1} - 1, 2^{n-1} - 1]$
      - Both 000...00 and 100...00 represent 0
    - Two's Complement Representation
      - $n$ -bit Range:  $[-2^{n-1}, 2^{n-1} - 1]$
      - 000...00 represents 0
      - 100...00 represents  $-2^{n-1}$
- Fixed-Point Number

0	0	1	1	0	0	0	1
x	x	x	x	x	x	x	x

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 49$$

### Sign Bit

1	0	1	1	0	0	0	1
x	x	x	x	x	x	x	x

$$- 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -49$$

1	1	0	0	1	1	1	1
x	x	x	x	x	x	x	x

$$-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -49$$



**Integer . Fraction**

**“Decimal” Point**

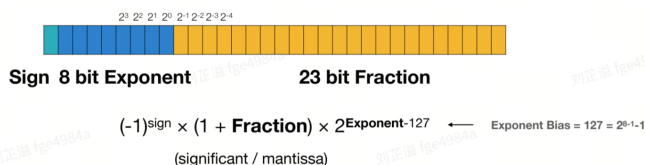


$$\begin{matrix} \times & \times & \times & \times & \times & \times & \times & \times \\ -2^3 & +2^2 & +2^1 & +2^0 & +2^{-1} & +2^{-2} & +2^{-3} & +2^{-4} \end{matrix} = 3.0625$$

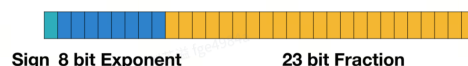
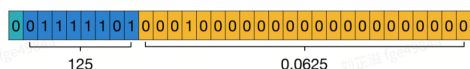


$$\begin{pmatrix} -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 \end{pmatrix} \times 2^{-4} = 49 \times 0.0625 = 3.0625$$

- Floating-Point Number: IEEE754 Standard



$$0.265625 = 1.0625 \times 2^{-2} = (1 + 0.0625) \times 2^{125-127}$$



Exponent	Fraction=0	Fraction≠0	Equation
00H = 0	±0	subnormal	$(-1)^{\text{sign}} \times \text{Fraction} \times 2^{1-127}$
01H ... FEH = 1 ... 254	normal		$(-1)^{\text{sign}} \times (1 + \text{Fraction}) \times 2^{\text{Exponent}-127}$
FFH = 255	±INF	NaN	



**Exponent Width → Range; Fraction Width → Precision**

**IEEE 754 Single Precision 32-bit Float (IEEE FP32)**



**IEEE 754 Half Precision 16-bit Float (IEEE FP16)**



**Google Brain Float (BF16)**



Exponent (bits)	Fraction (bits)	Total (bits)
8	23	32
5	10	16
8	7	16

- FP8 (E4M3)通常在forward使用；FP8 (E5M2)通常在backward使用，因为可表示的范围更大

## IEEE 754 Single Precision 32-bit Float (IEEE FP32)



Exponent (bits)	Fraction (bits)	Total (bits)
8	23	32

## IEEE 754 Half Precision 16-bit Float (IEEE FP16)



Exponent (bits)	Fraction (bits)	Total (bits)
5	10	16

## Nvidia FP8 (E4M3)



\* FP8 E4M3 does not have INF, and S.1111.111<sub>2</sub> is used for NaN.  
\* Largest FP8 E4M3 normal value is S.1111.110<sub>2</sub>=448.

Exponent (bits)	Fraction (bits)	Total (bits)
4	3	8

## Nvidia FP8 (E5M2) for gradient in the backward



\* FP8 E5M2 have INF (S.11111.00<sub>2</sub>) and NaN (S.11111.XX<sub>2</sub>).  
\* Largest FP8 E5M2 normal value is S.11110.11<sub>2</sub>=57344.

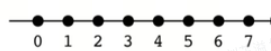
Exponent (bits)	Fraction (bits)	Total (bits)
5	2	8

- FP4 and INT4

### INT4

S			
0	0	0	1
0	1	1	1

-1, -2, -3, -4, -5, -6, -7, -8  
0, 1, 2, 3, 4, 5, 6, 7

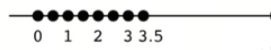


-1, -2, -3, -4, -5, -6, -7, -8  
0, 1, 2, 3, 4, 5, 6, 7

### FP4 (E1M2)

S	E	M	M
0	0	0	1
0	1	1	1

-0, -0.5, -1, -1.5, -2, -2.5, -3, -3.5  
0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5

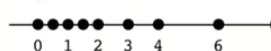


-0, -1, -2, -3, -4, -5, -6, -7  
0, 1, 2, 3, 4, 5, 6, 7 × 0.5

### FP4 (E2M1)

S	E	E	M
0	0	0	1
0	1	1	1

-0, -0.5, -1, -1.5, -2, -3, -4, -6  
0, 0.5, 1, 1.5, 2, 3, 4, 6



-0, -1, -2, -3, -4, -6, -8, -12  
0, 1, 2, 3, 4, 6, 8, 12 × 0.5

### FP4 (E3M0)

S	E	E	E
0	0	0	1
0	1	1	1

-0, -0.25, -0.5, -1, -2, -4, -8, -16  
0, 0.25, 0.5, 1, 2, 4, 8, 16



-0, -1, -2, -4, -8, -16, -32, -64  
0, 1, 2, 4, 8, 16, 32, 64 × 0.25

## 什么是量化

- 量化是将输入从连续或大型集的值限制成离散集的过程

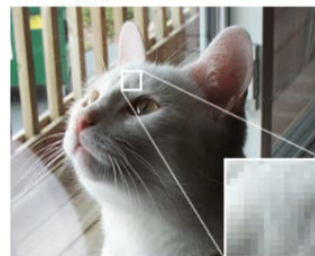
### Continuous Signal Quantized Signal



The difference between an input value and its quantized value is referred to as quantization error.

Quantization [Wikipedia]

### Original Image



### 16-Color Image



Images are in the public domain.

"Palettization"

## K-Means-based Weight Quantization

- Storage: Integer Weights; Floating-Point Codebook

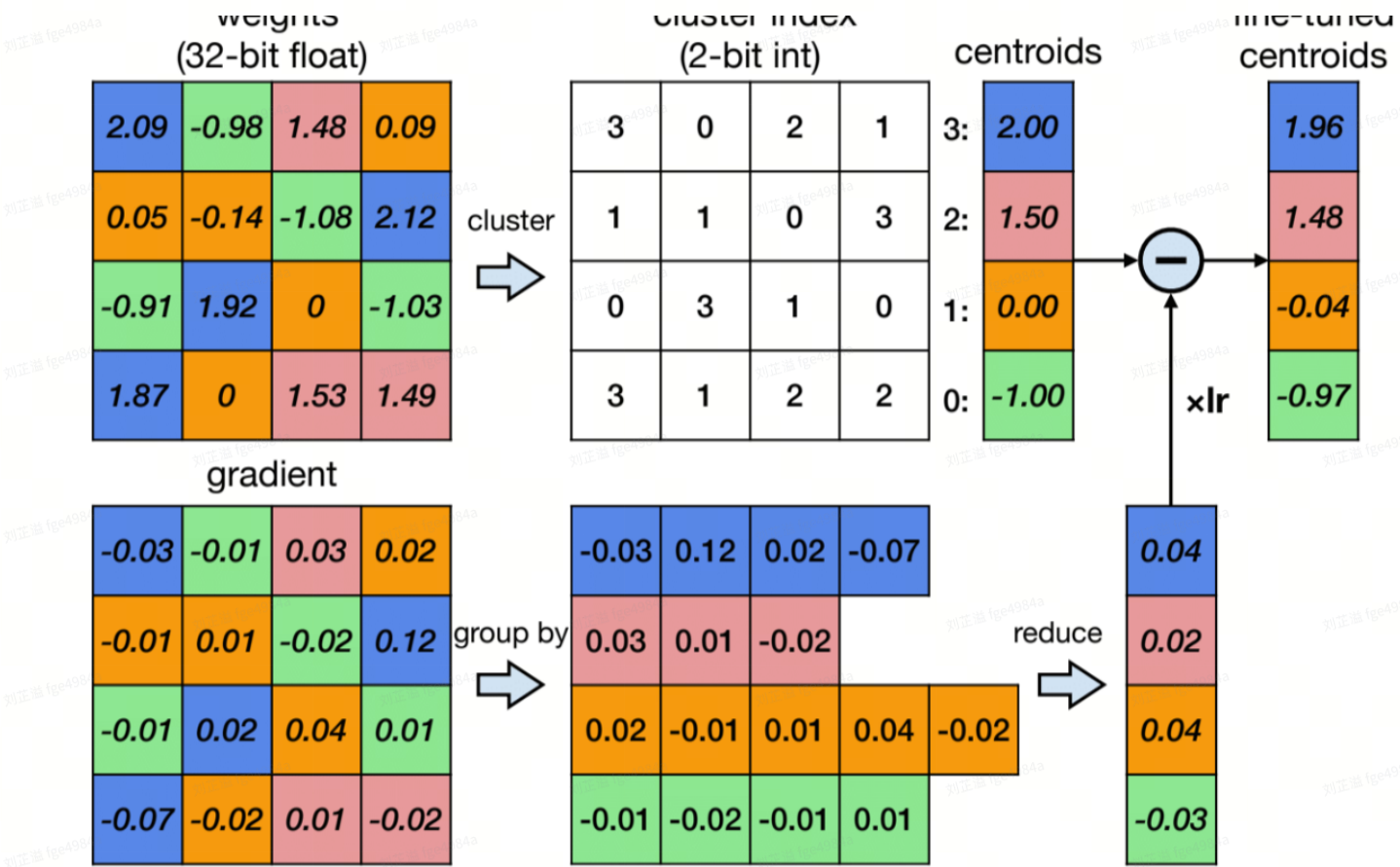
- Computation: Floating-Point Arithmetic

3	0	2	1	3:	2.00
1	1	0	3	2:	1.50
0	3	1	0	1:	0.00
3	1	2	2	0:	-1.00

## K-Means-based Quantization

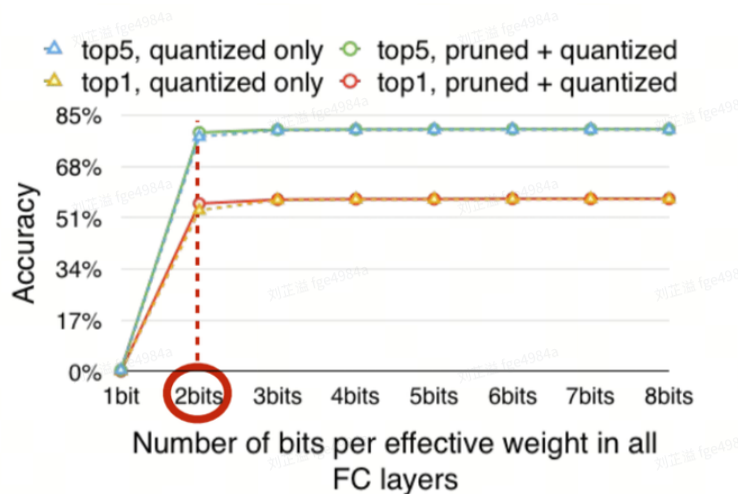
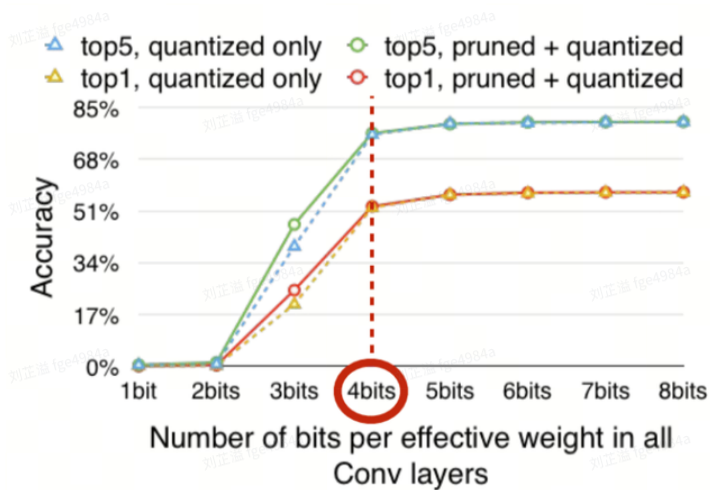
### Fine-tuning Quantized Weights

- 按照Codebook进行分组相加后得到Codebook的gradient

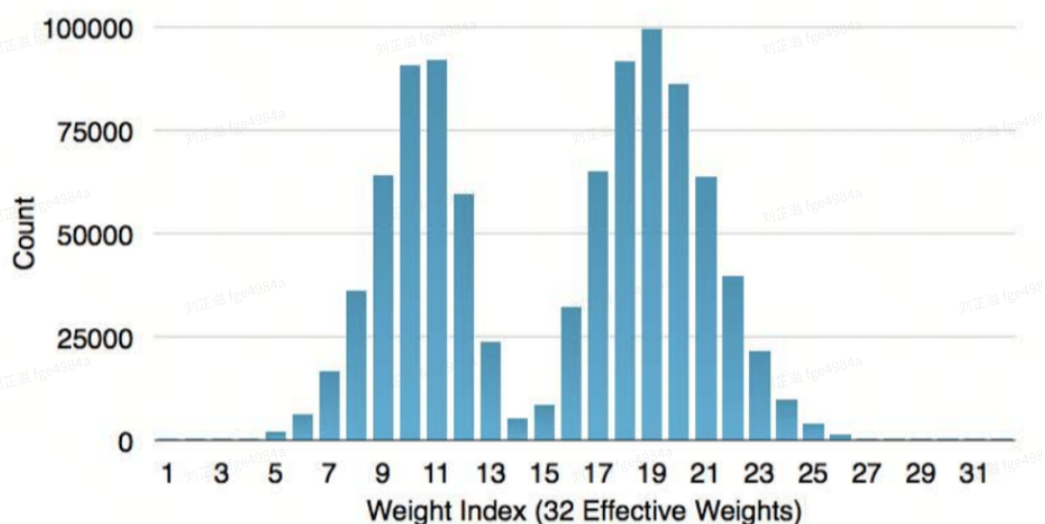
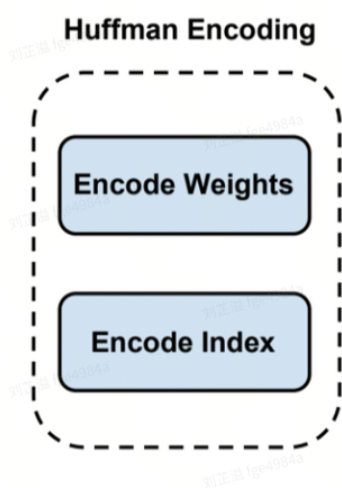


### 不同层需要的bit数

- 卷积层4bits；FC层2bits



## 霍夫曼编码

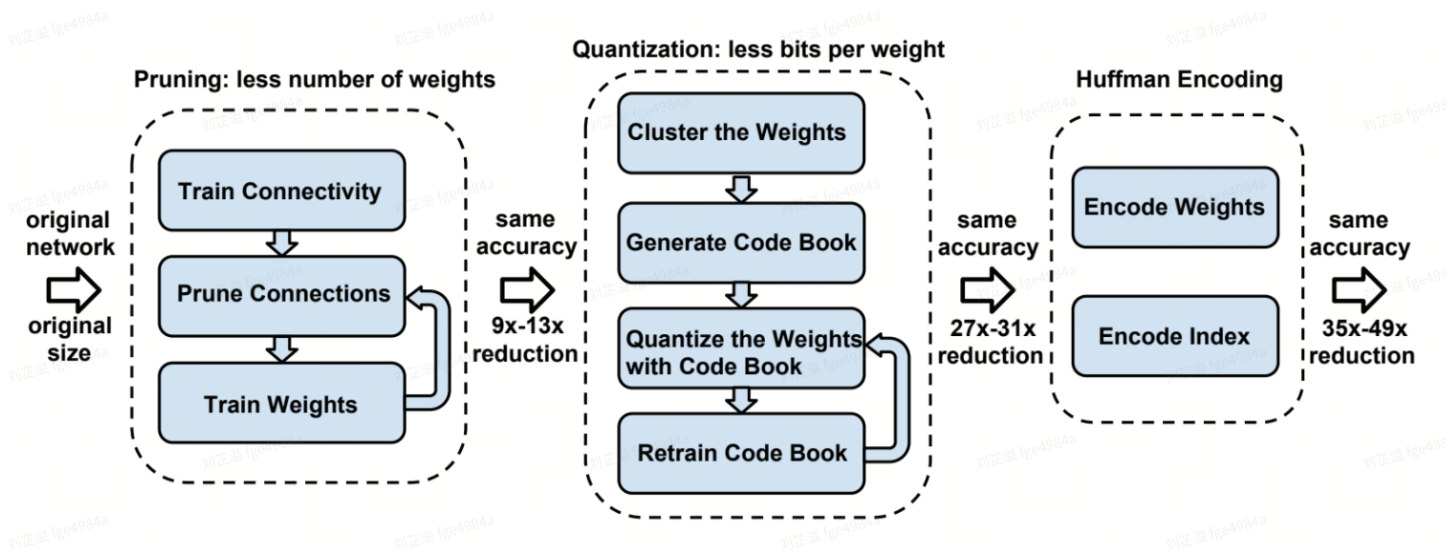


- In-frequent weights: use more bits to represent
- Frequent weights: use less bits to represent

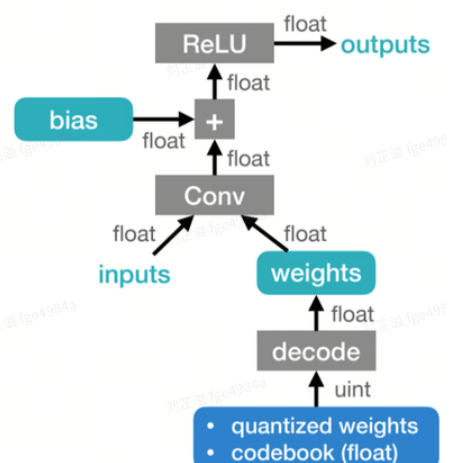
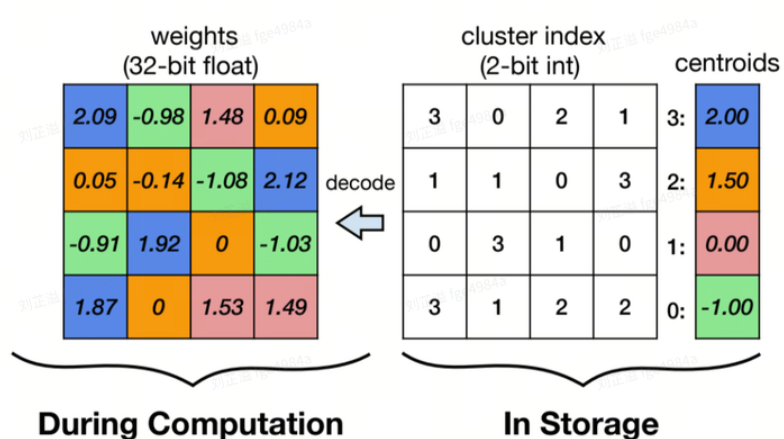
## Deep Compression

- 使用剪枝+量化+编码（不常使用）





- 只有存储是使用量化，而计算和内存访问仍然是floating-point



- The weights are decompressed using a lookup table (i.e., codebook) during runtime inference.
- K-Means-based Weight Quantization only saves storage cost of a neural network model.
  - All the computation and memory access are still floating-point.

## Linear Quantization