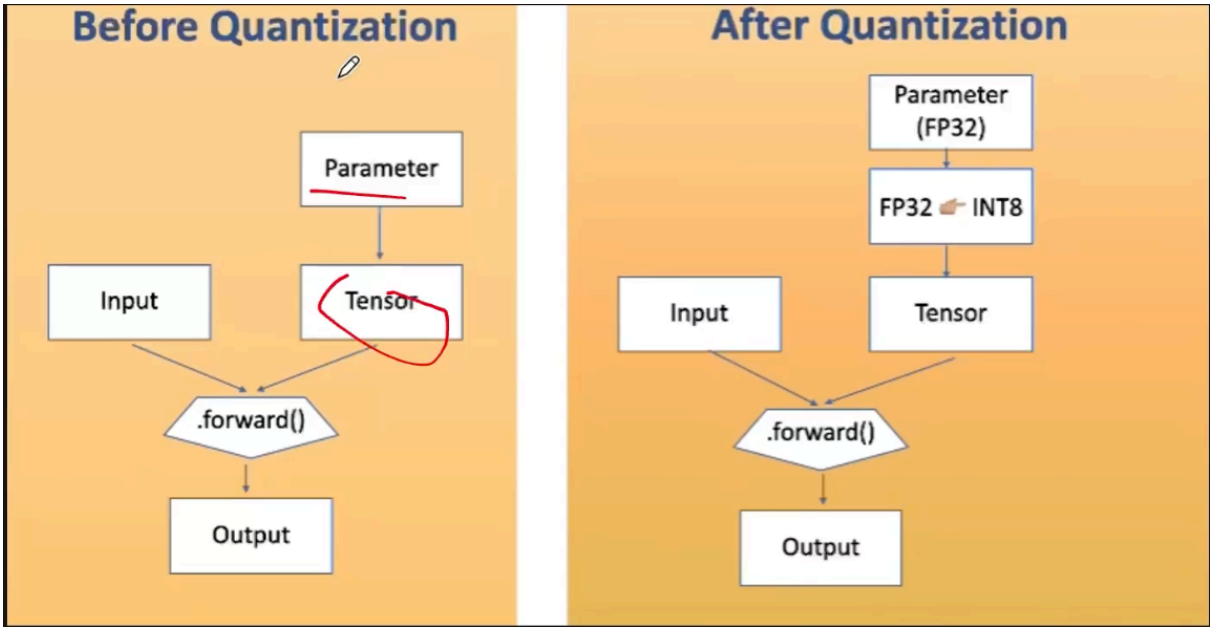
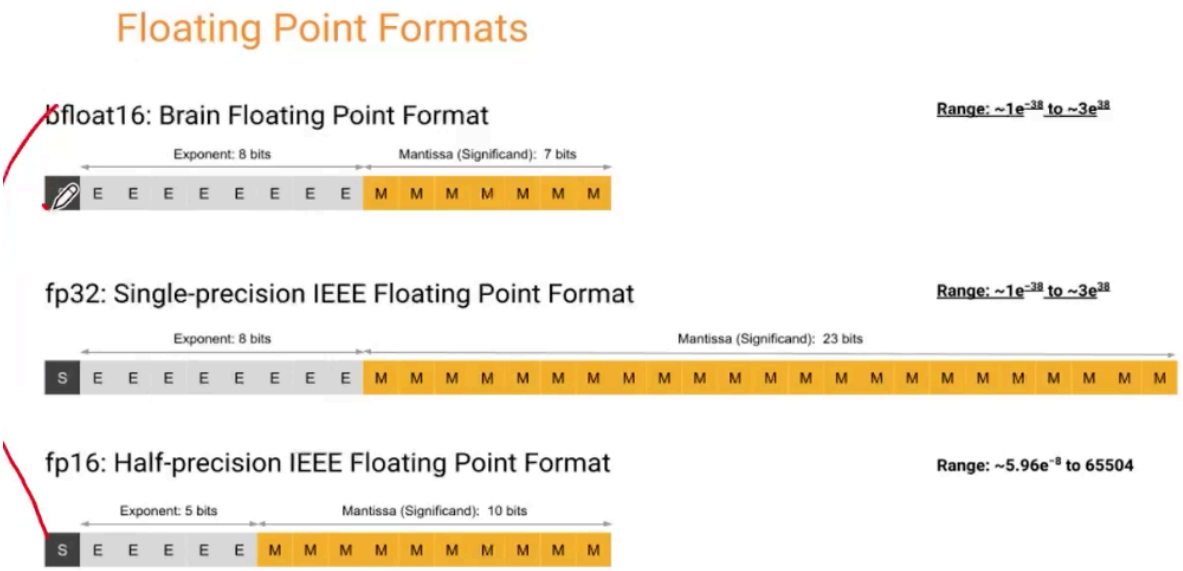
Quantization의 개념

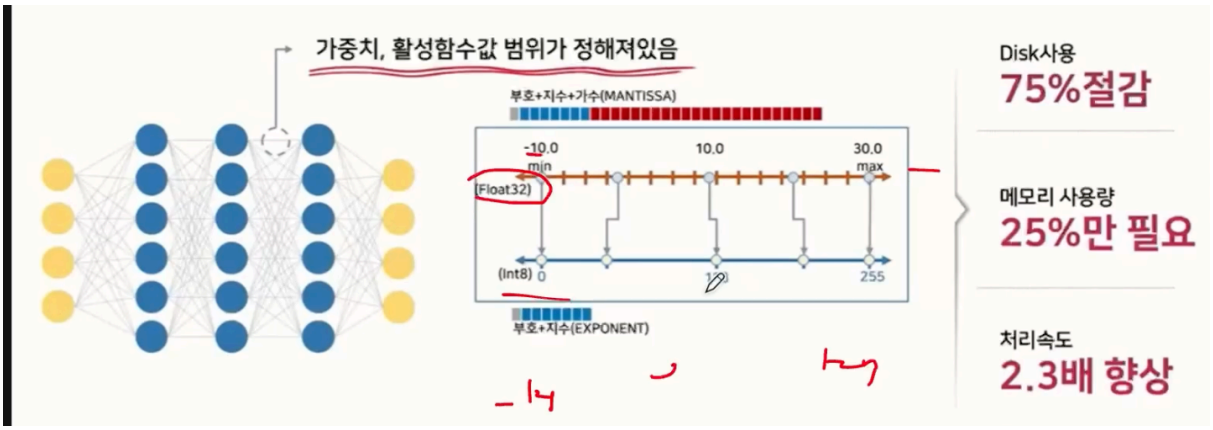


- 숫자를 더 정확히 표현할 수록 큰 모델 크기가 필요하다.
- 보통의 딥러닝 모델은 FP32(4byte) 사용
  - 이 표현값을 줄이고 싶어하는 경향성이 있음 (→ INT8)



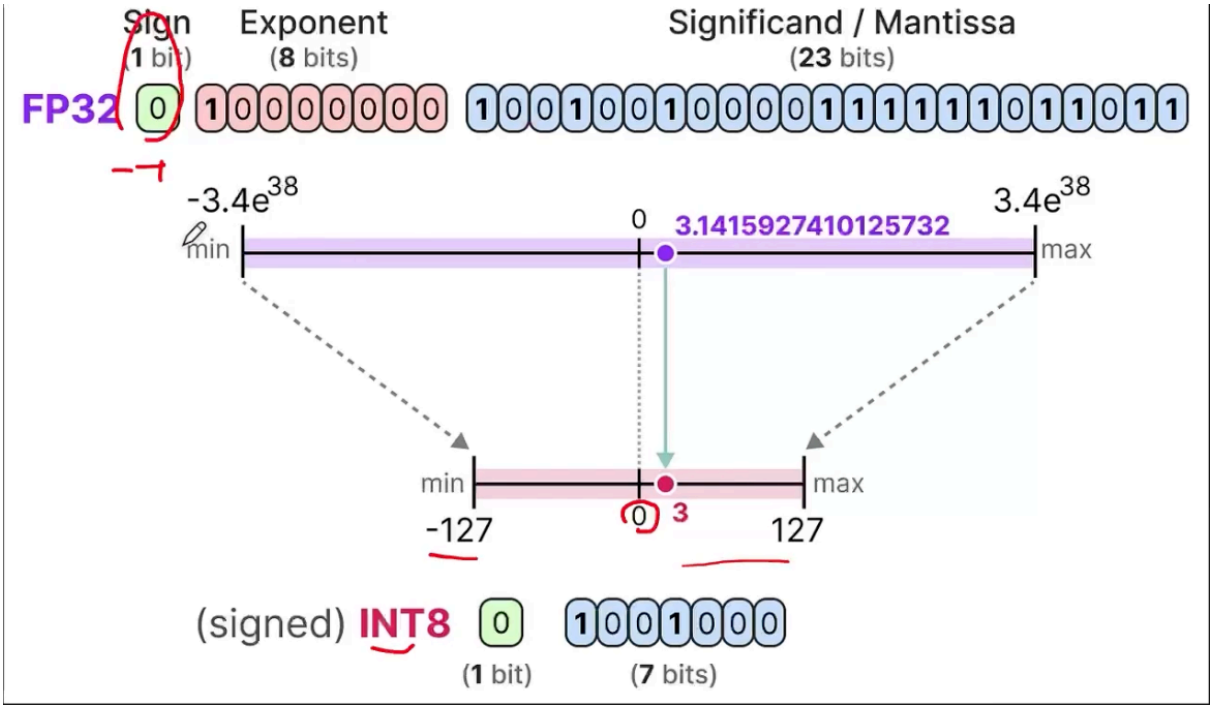
- bfloat은 fp를 16비트만으로 표현하고자 하는 방법론

- 최근 딥러닝에서 많이 사용되는 표현 방법
- 지수에 대한 표현력을 조금 더 높여줌
  - 가수부가 적기때문에 정밀도가 낮지만, 값에 대한 표현력을 증가시켰음



- 전체 메모리 사용량 감소 및 연산 속도를 향상시키는 효과 있음
  - 정수형이 하드웨어 친화적이기도 하다.
  - 처리속도의 경우는 NPU같은 하드웨어 의존적인 경향이 있다.

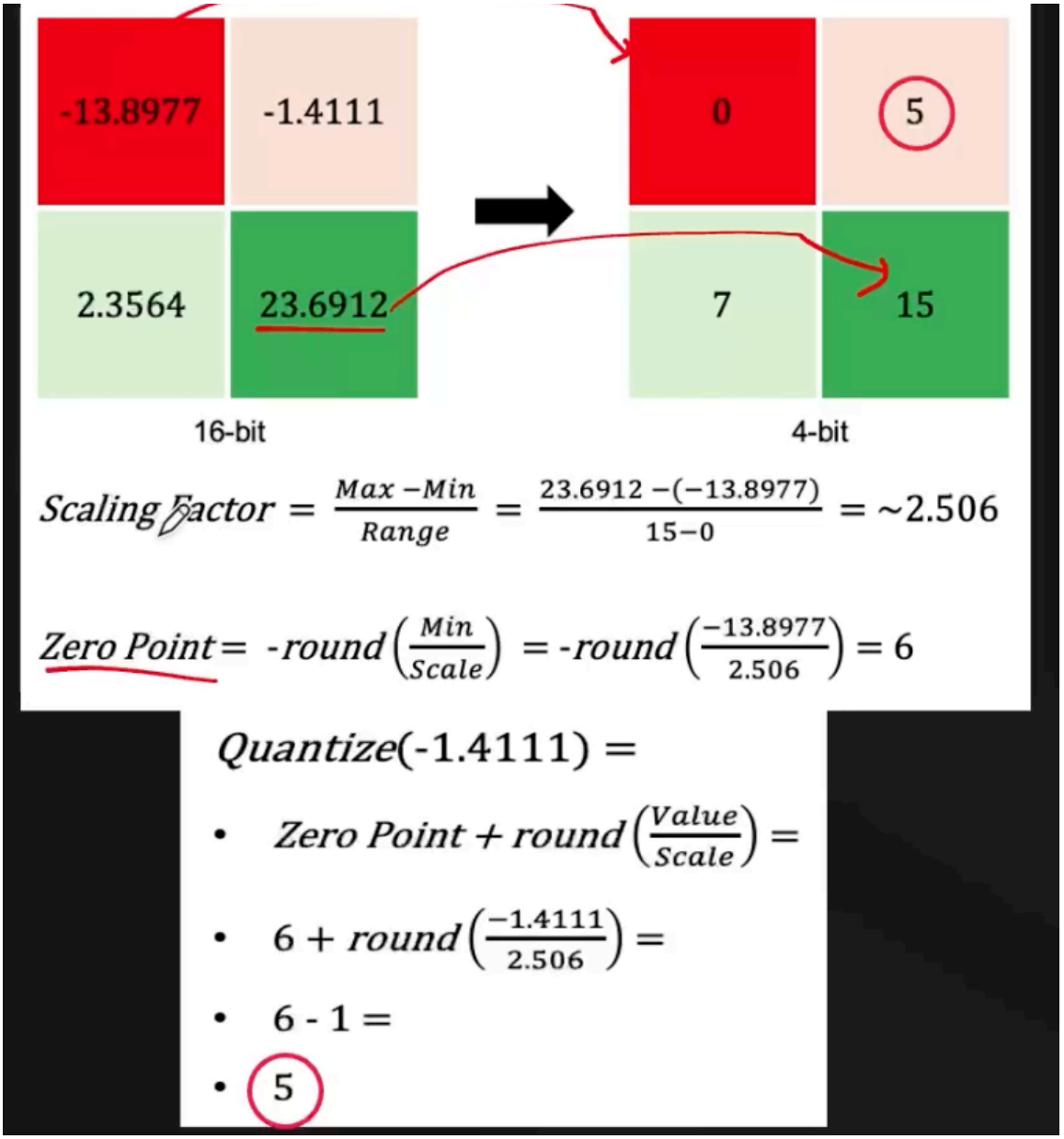
양자화의 데이터 타입

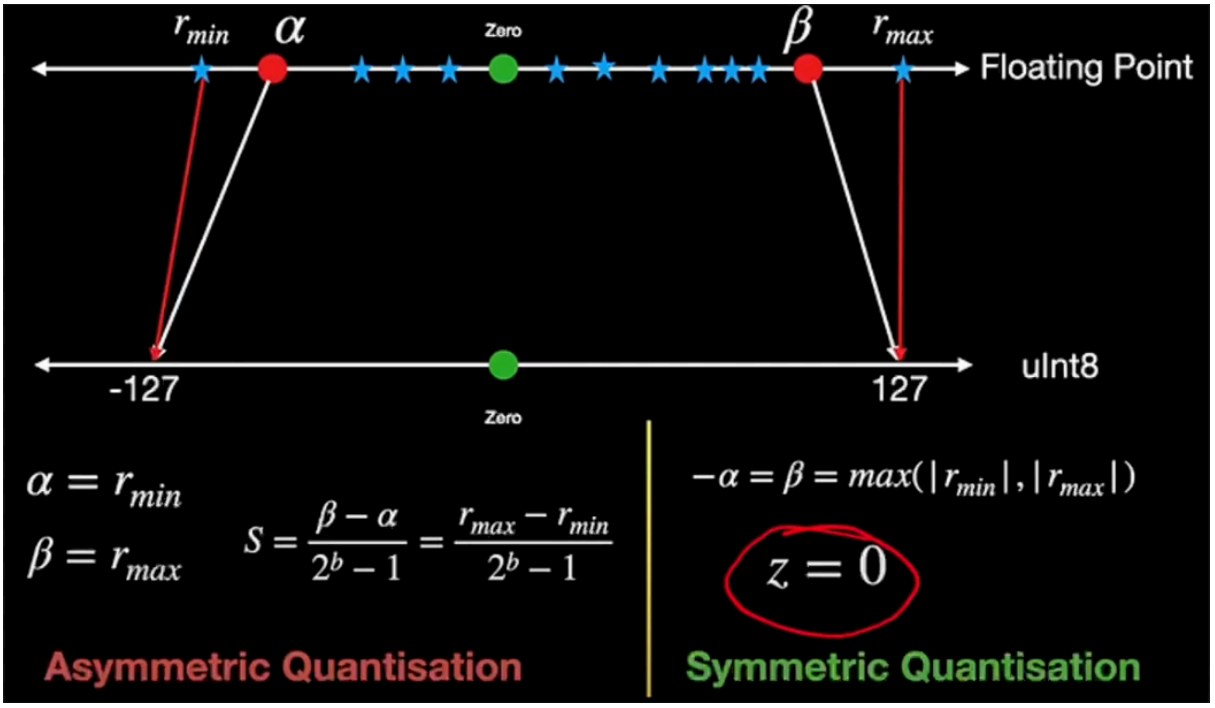


- MIN/MAX 값 매핑
- 3.14의 경우 round()해서 3으로매핑

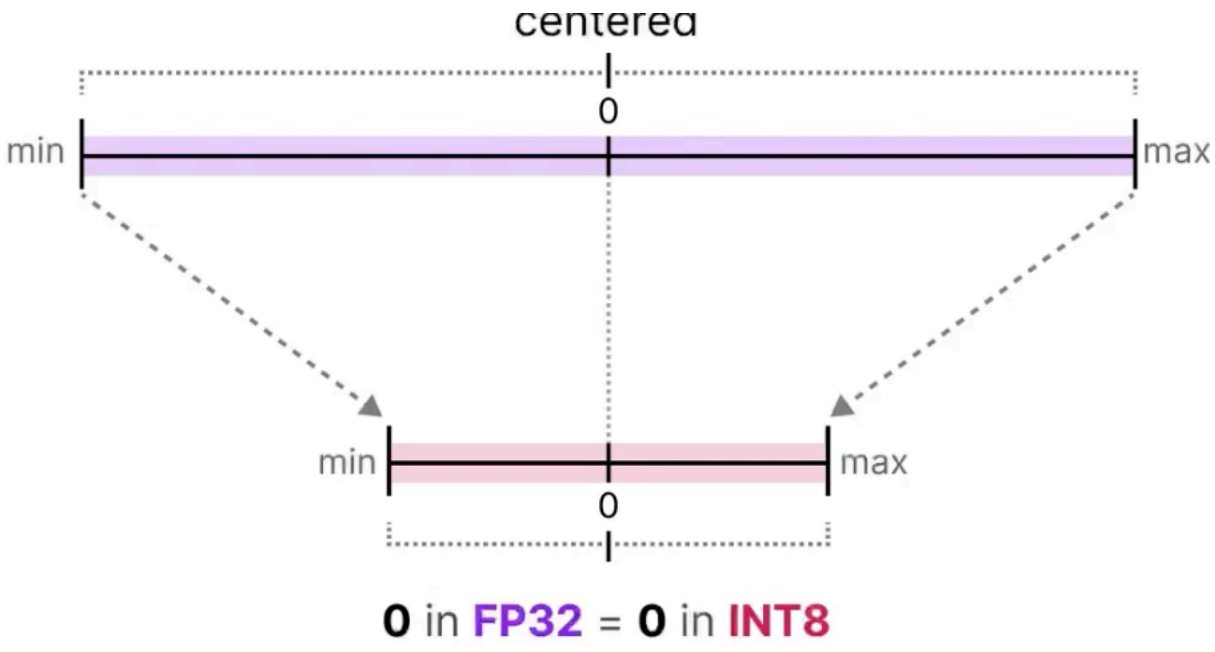
어떻게 값을 매핑하는가? →Quantization의 원리와 핵심

- Quantizaton의 핵심 원리는 모델의 파라미터와 activation의 d_type을 더 적은 용량으로 표현하기 위함에 있음
- 높은 정밀도는 수를 표현하는 데 매우 풍부한 표현을 가질 수 있지만
  그 만큼 메모리에 trade-off 함
- 다만 이를 Quantization을 통해 더 적은 메모리로 표현할 수 있다면,
  더 다양하고 큰 모델을 다루는 데 있어서 매우 핵심적이고 효율적임


-13.8977
-1.4111
2.3564
23.6912
0
5
7
15
16-bit
4-bit
$Scaling\ Factor = \frac{Max - Min}{Range} = \frac{23.6912 - (-13.8977)}{15-0} = \sim 2.506$
$Zero\ Point = -round\left(\frac{Min}{Scale}\right) = -round\left(\frac{-13.8977}{2.506}\right) = 6$
$Quantize(-1.4111) =$
$Zero\ Point + round\left(\frac{Value}{Scale}\right) =$
$6 + round\left(\frac{-1.4111}{2.506}\right) =$
6 - 1 =
5

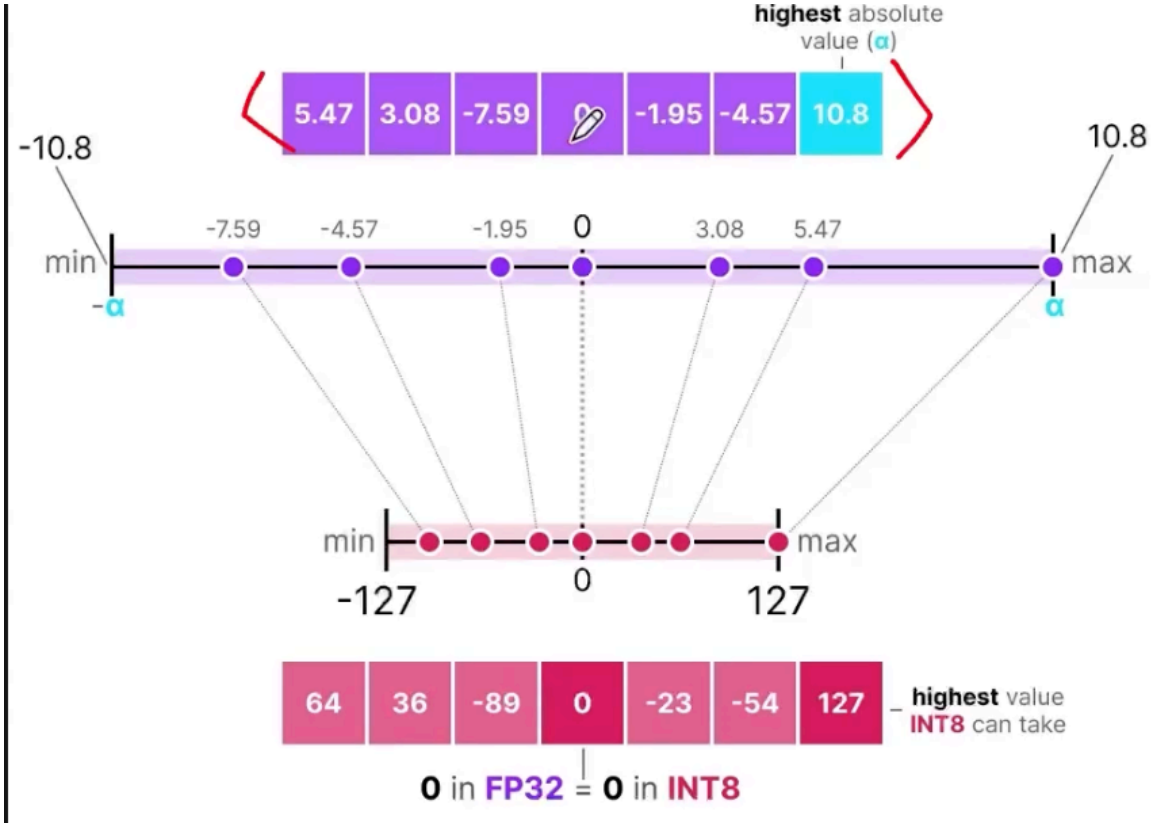


Symmetric Quantization

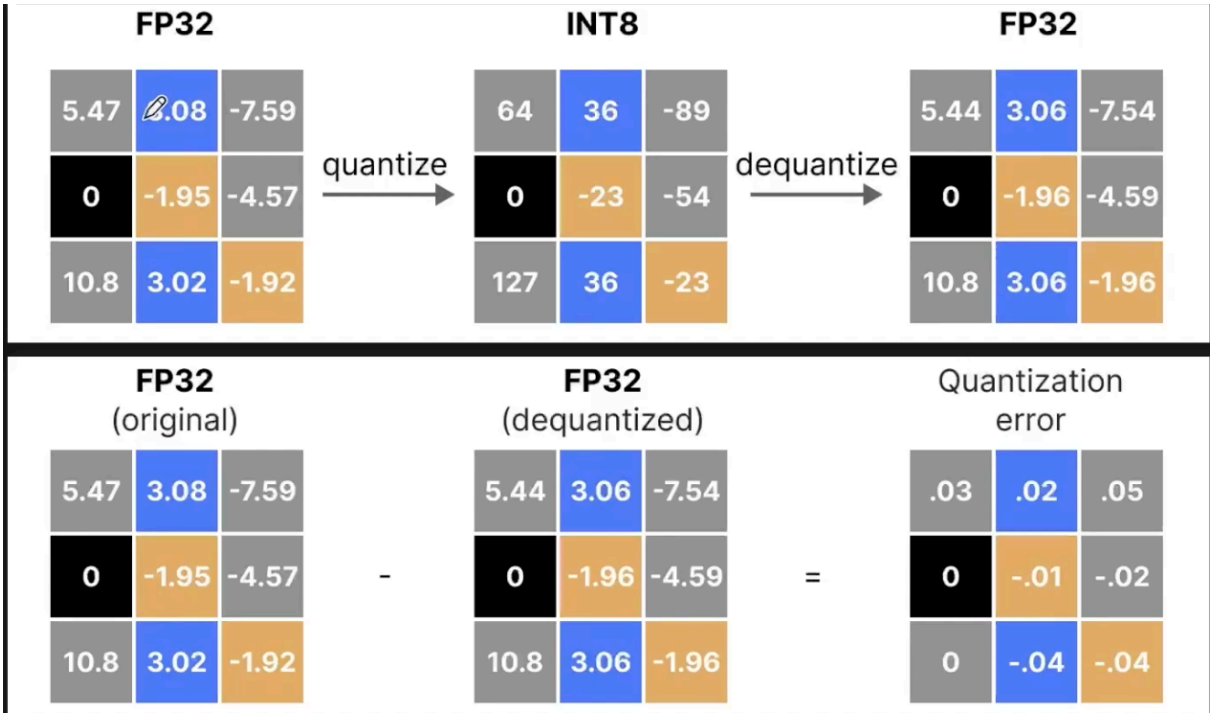


- 원래 모델 weight의 float 범위를 양자화된 공간에서 0을 중ㅅ미으로 대칭적으로 매핑
- 양자화 전후의 범위가 0을 중심으로 유지
- 대칭 양자화의 좋은 예는 (absolute maximum, absmax) 양자화임

- 값 목록을 주어진 상태에서 가장 큰 절대값(α)을 범위로 사용하여 선형 매핑을 수행
- 0을 중심으로 하기에 매우 간단한 공식으로 매핑할 수 있음
- scaling factor는 양자화 바이트수 (b; 8)와 가장 큰 절대값(a)을 활용하여 구함

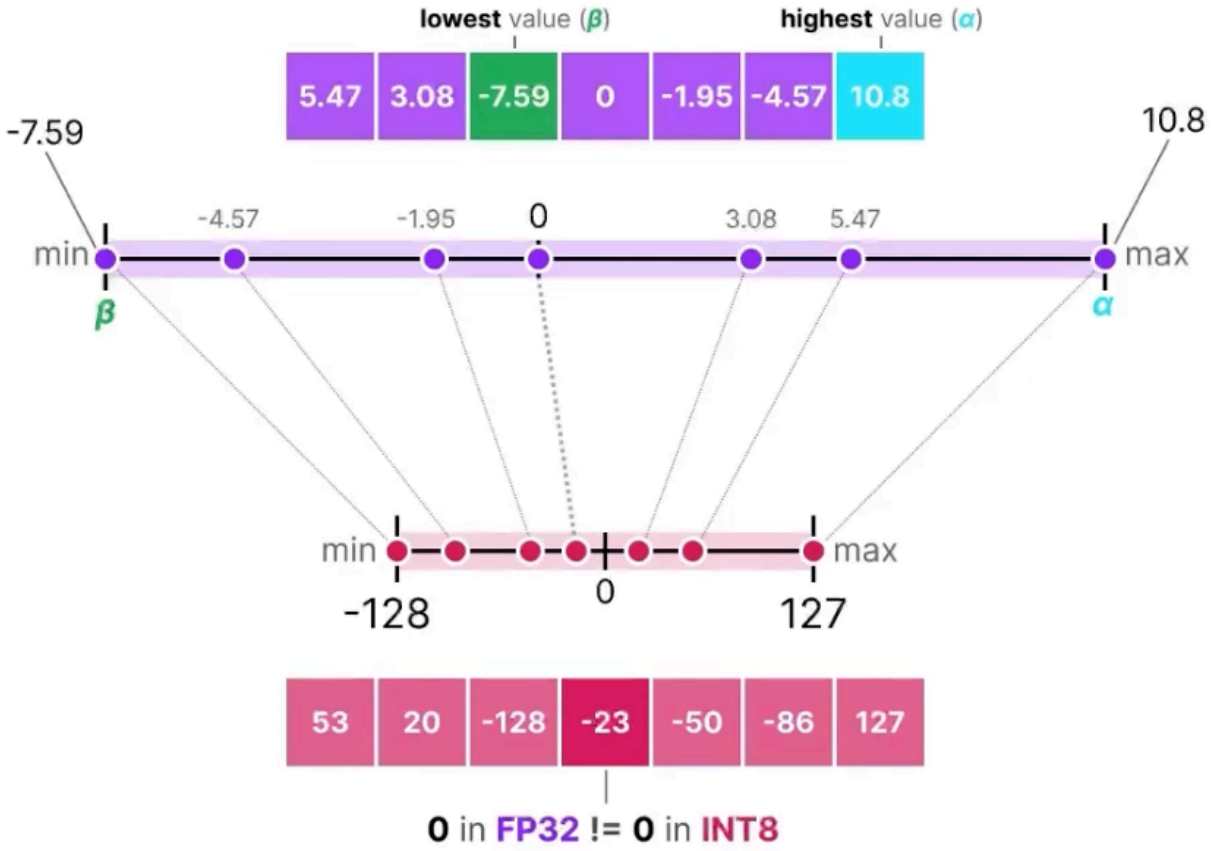


Quantization Error



Asymmetric Quantization

- 데이터의 분포를 그대로 잘 반영하겠다.



- 원래 모델의 weight의 float 범위를 양자화된 공간에서 0을 중심으로 매핑하지 않고, 부동소수점의 최소와 최대범위 값을 활용하여 매핑함
- 0의 위치가 기존과 다르게 중심이 아니라 최소값을 반영해서 매핑하겠다.
- 제로 포인트를 데이터의 실제 중심에 맞출 수 있어 양자화가 데이터 분포를 더 잘 반영
- 데이터 분포에 대한 유연성이 있어 정밀도를 높일 수 있지만, 구현의 복잡도와 연산 비용이 증가

$$s = \frac{128 - -127}{\alpha - \beta} \quad \text{(scale factor)}$$

$$z = \text{round}\left(-s \cdot \beta\right) - 2^{b-1} \quad \text{(zeropoint)}$$

$$x_{quantized} = \text{round}\left(s \cdot x + z\right) \quad \text{(quantization)}$$

$$s = \frac{255}{10.8 - -7.59} = 13.86 \quad \text{(scale factor)}$$

$$z = \text{round}\left(-13.86 \cdot -7.59\right) - 128 = -23 \quad \text{(zeropoint)}$$

$$x_{quantized} = \text{round}\left(13.86 \cdot \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare + -23\right) \quad \text{(quantization)}$$

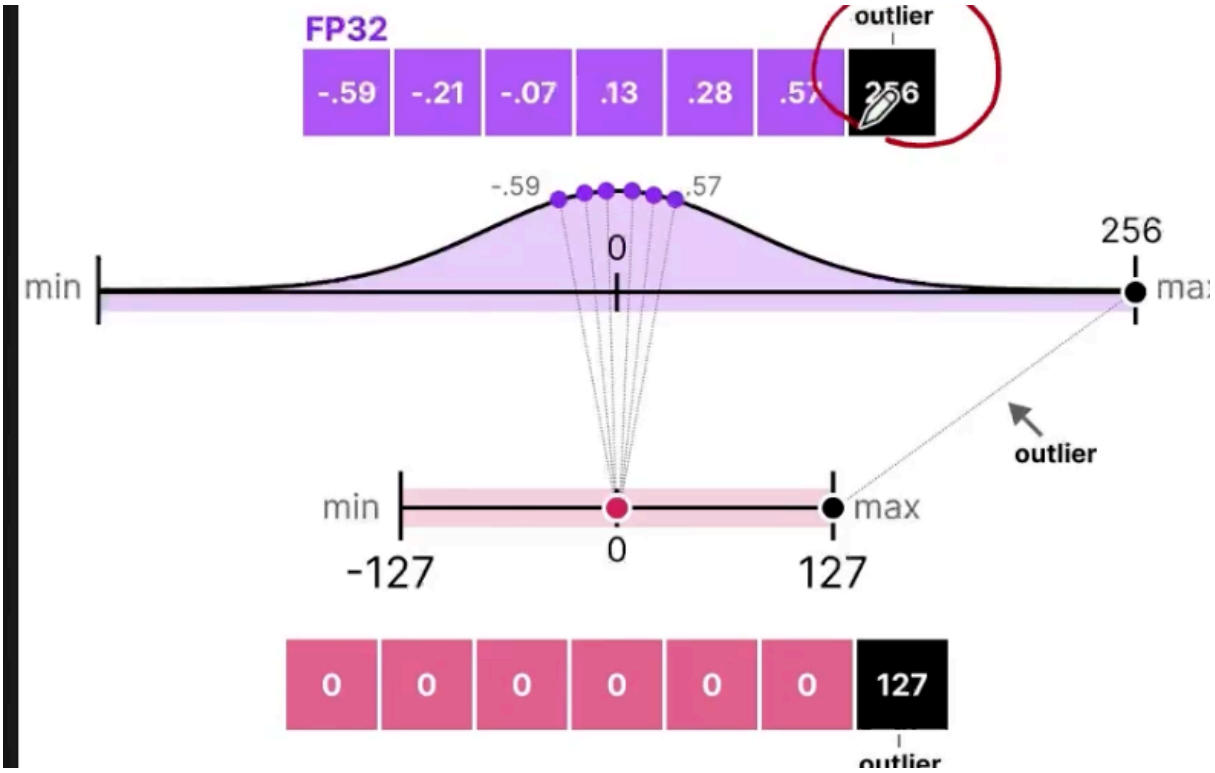
- INT8의 최대 최소 범위 차이 [-128, 127]를 사용하여 분자로 하고,
  최솟값과 최대값을 활용하여 scale factor를 구함
- 이후 zero point를 scale factor와 최소값을 활용하여 구한 이후
  이를 초기 시작점과 같이 활용하여 양자화를 진행함 (zero point quantization)
- dequantize도 이에 대한 역수를 활용하여 기존 데이터 값으로 변환 가능

$$x_{dequantized} = \frac{\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare - z}{s} \quad \text{(dequantize)}$$

두 양자화 기법의 차이

Symmetric Quantization	Asymmetric Quantization
단순한 구현	**구현의 복잡성**
일관된 스케일링	효과적인 표현범위
효율적인 하드웨어 사용	**메모리 사용 증가**
정밀도 손실 가능성	정밀도에서 적은 손실
데이터를 유동적으로 처리하지 않음	**유연한 표현**
	추가연산의 부담

Quantization의 뒷면

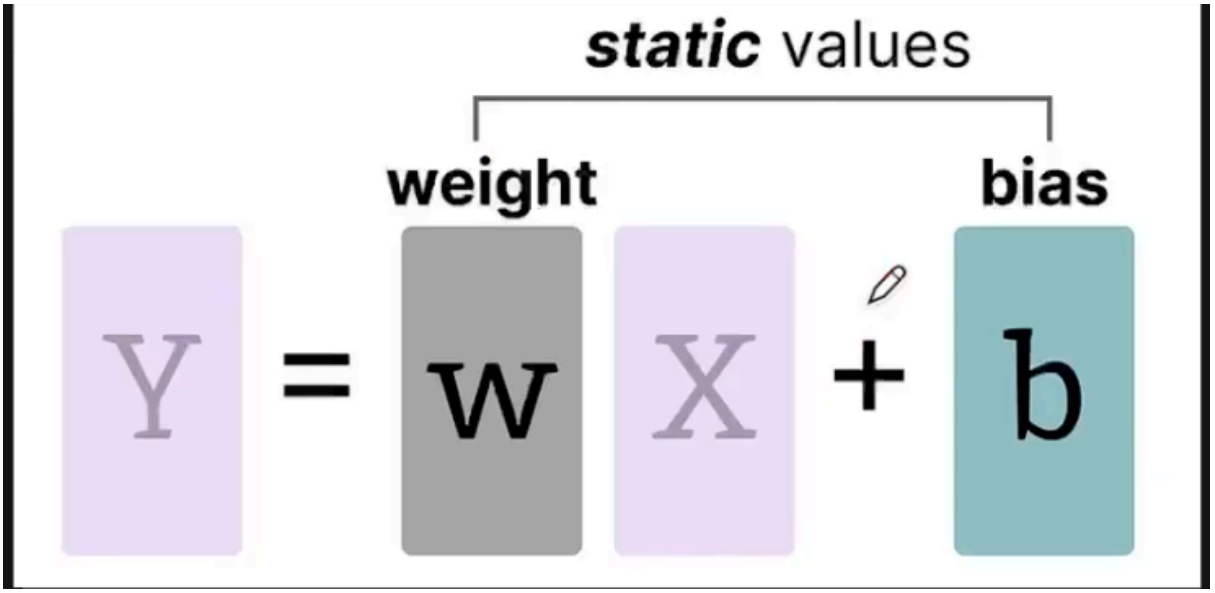


- 다음과 같이 256의 큰 Outlier가 있는 Vector
- 이를 기반으로 quantization을 진행하게 되면, outlier 값만 제외하고 모든 값들이 다 0이 되어버리는 문제가 발생함
- 즉, 특정 가중치 이외의 가중치는 의미없는 가중치가 되어버리고, 모델이 망가지는 문제가 발생
- 이런 경우 cliping을 통해 이상치를 적당한 값으로 조절하여 quantization 진행

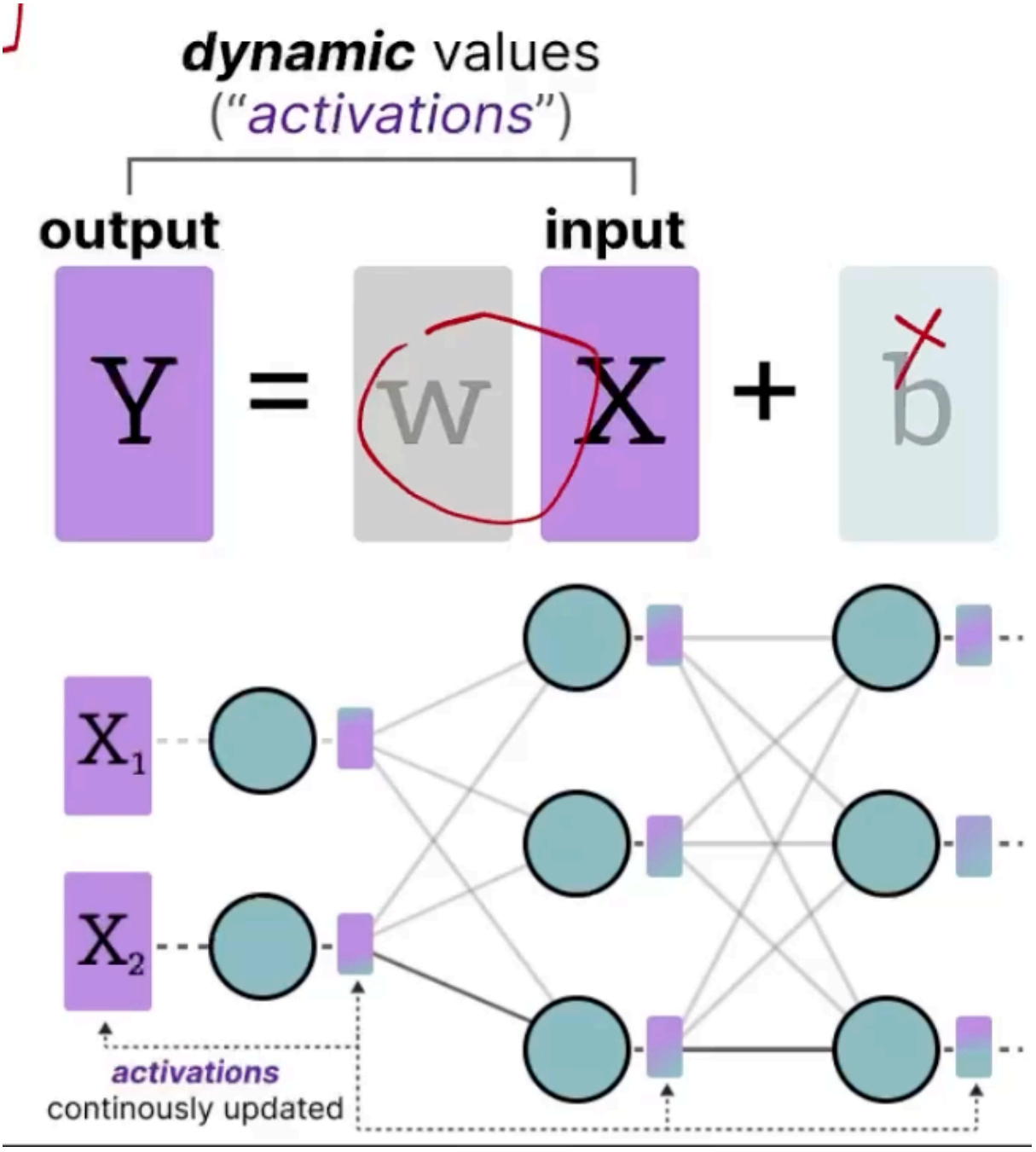
- cliping은 어떠한 가중치가 특정한 값의 범위를 넘어가게 되면,
  이를 INT 범위 내에 적절하게 매핑해준다는 장점
- 이를 통해 이상치에 대해서도 강건하게 값을 매핑해줌으로써 안정적으로 양자화를 진행
- 양자화의 핵심은 이상치와 error를 최소화 하는 것

Quantization: Calibration

- 그러면 이런 이상치의 값을 어떻게 판단하는 것이 맞을까?
  - 이 이상치라 판단하는 값의 범위를 찾아나가는 게 calibration으로 볼 수 있음
- LLM의 가중치는 Weight와 bias의 affine transformation으로 값을 고려하는 건 bias가 아닌 W에 집중하는 것이 가장 적합
  - wight의 값이 bias보다 훨씬 많기 때문
- 가중치의 Quantization calibration은 다음과 같이 수행
  - 수동으로 백분위를 선택
  - 양자화된 가중치와 기존 가중치 간의 MSE 최소화
    - outlier 판단 범위를 최소화 하는 방법(?)
  - 양자화된 가중치와 기존 가중치 간의 KL Divergence
    - KL Divergence: 두 분포 간의 차이점을 최소화 (분포 유사도를 최대한 일치하게)



- 그런데 과연 가중치만 신경써야 하는가?
- 모델을 foward 할 때 생기는 값을 일반적으로 activation이라고 함
- activation은 추론 중 입력 데이터가 모델을 통과할 때마다 달라지게 됨으로,
  정확한 양자화를 하기 매우 어려움
  (데이터 마다 값이 다르기 떄문)
- activation은 hidden layer를 거칠 때 마다 업데이트 되므로,
  모델이 실행될 때 입력 데이터가 모델을 통과할 때 그 값이 무엇인지 알 수 있다.
- 이러한 값들을 기반으로 가중치를 양자화하는 방법론 등이 있으며
  PTQ, QAT가 activation을 고려항며 양자화를 실행함


dynamic values
("activations")
output
input
Y = w X + b
X1
X2
activations
continously updated