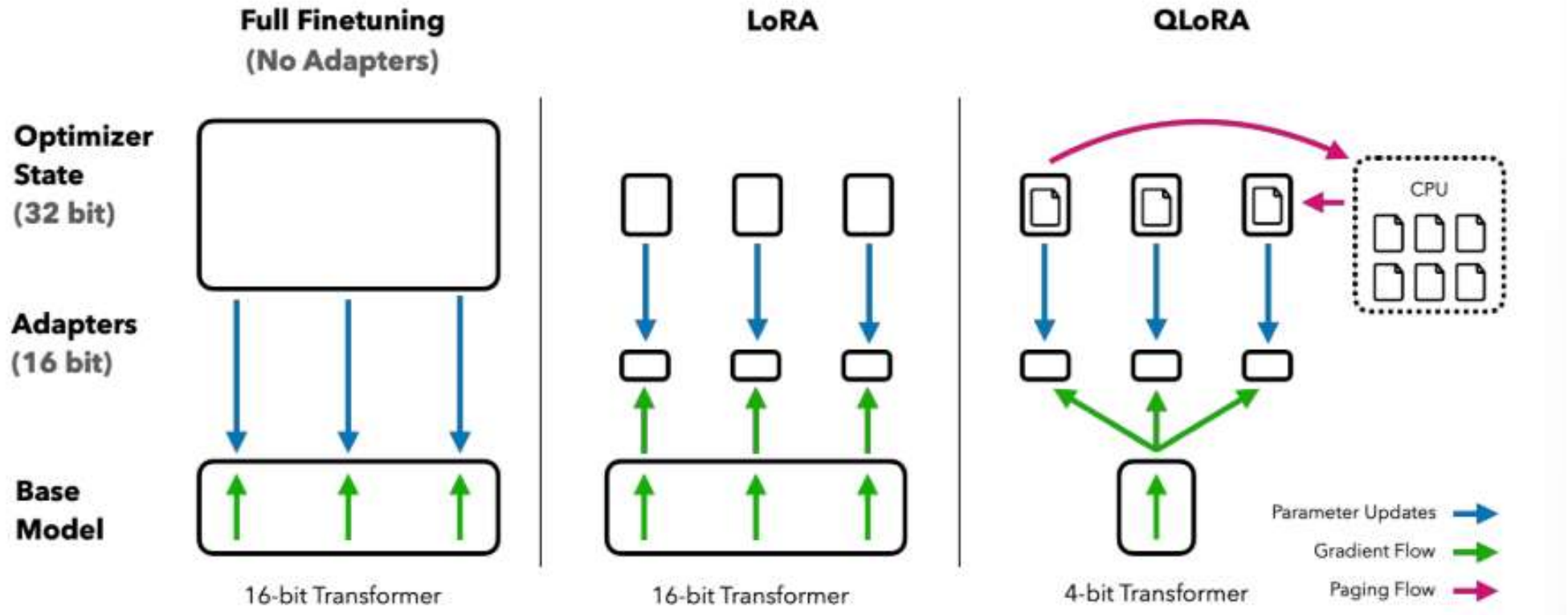
QLoRA : Efficient Finetuning of Quantized LLMs

Seongeun Baek

2024. 3. 26

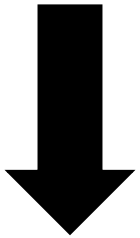
QLoRA란?



한정된 리소스로 **full fine-tuning**의 성능을 재현할 수 있는 효과적인 **tuning** 방법

Main points

- **4-bit NormalFloat (NF4)**
- **Double Quantization by quantizing the quantization constants**
- **Paged Optimizers**



ChatGPT performance의 97.8%까지 재현 가능

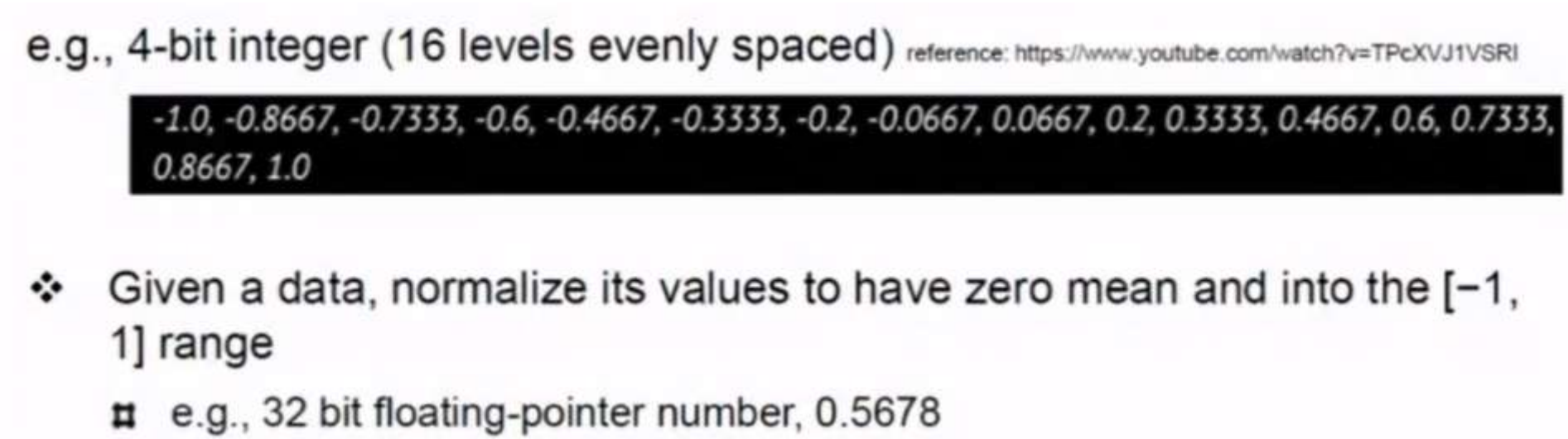
Background (Block-wise k-bit Quantization)

$$\mathbf{X}^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(\mathbf{X}^{\text{FP32}})}\mathbf{X}^{\text{FP32}}\right) = \text{round}(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}}),$$

$$\text{dequant}(c^{\text{FP32}}, \mathbf{X}^{\text{Int8}}) = \frac{\mathbf{X}^{\text{Int8}}}{c^{\text{FP32}}} = \mathbf{X}^{\text{FP32}}$$

- **Quantization :** 많은 정보를 가지고 있는 representation → 적은 정보를 가진 representation으로 변환하는 과정
- 32-bit float → 8-bit Integers

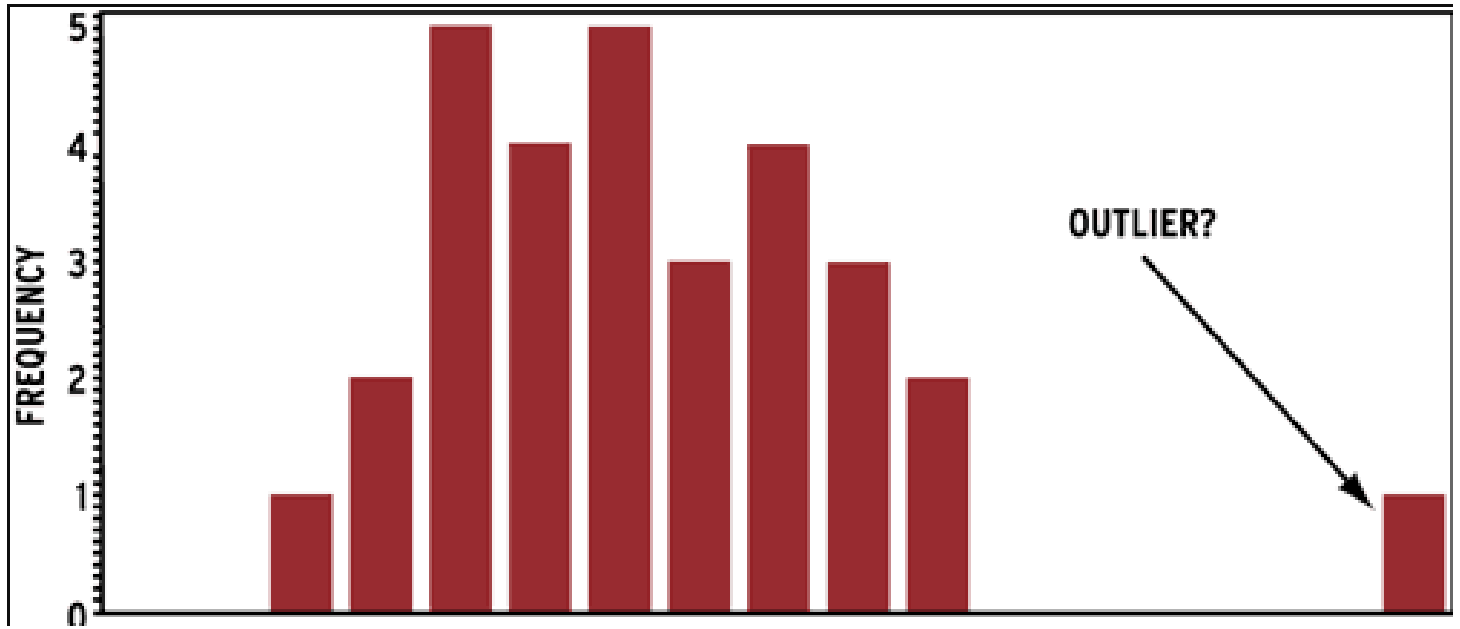
Background (Block-wise k-bit Quantization)



32-bit floating pointer number → 4-bit로 바꾸는 예시

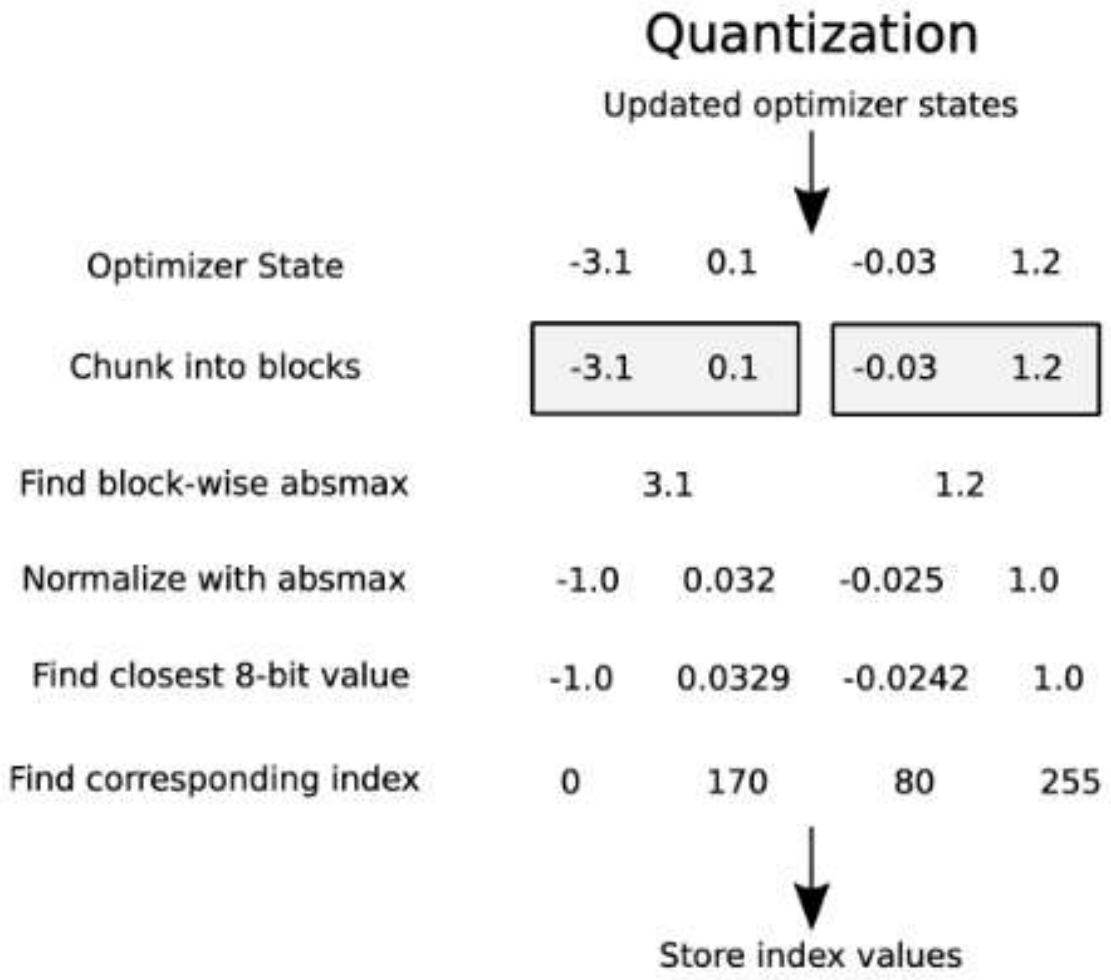
- [-1,1] 사이 간격을 16 level로 고르게 분배
- 32-bit floating pointer number를 normalizing 후, quantization
- ex. 0.5678 → 13

Background (Block-wise k-bit Quantization)



Outlier와 같은 문제점 → Quantized block 형태로 chunk

Background (Block-wise k-bit Quantization)



Background (Low-Rank Adapter)

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + s\mathbf{X}\mathbf{L}_1\mathbf{L}_2,$$

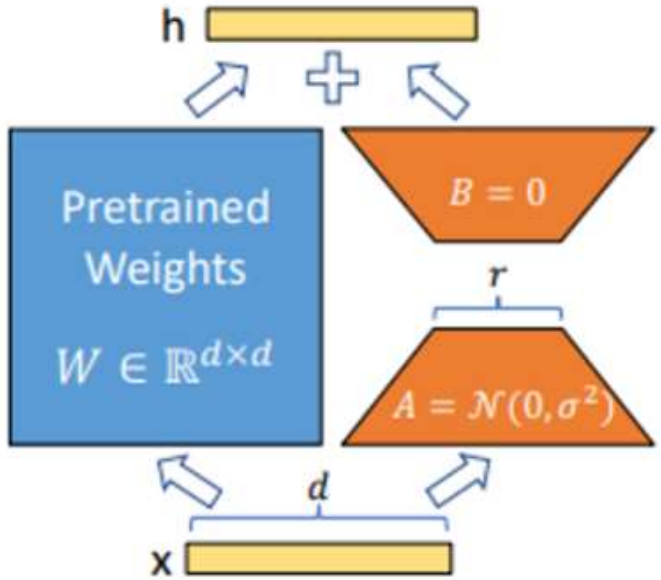


Figure 1: Our reparametrization. We only train A and B.

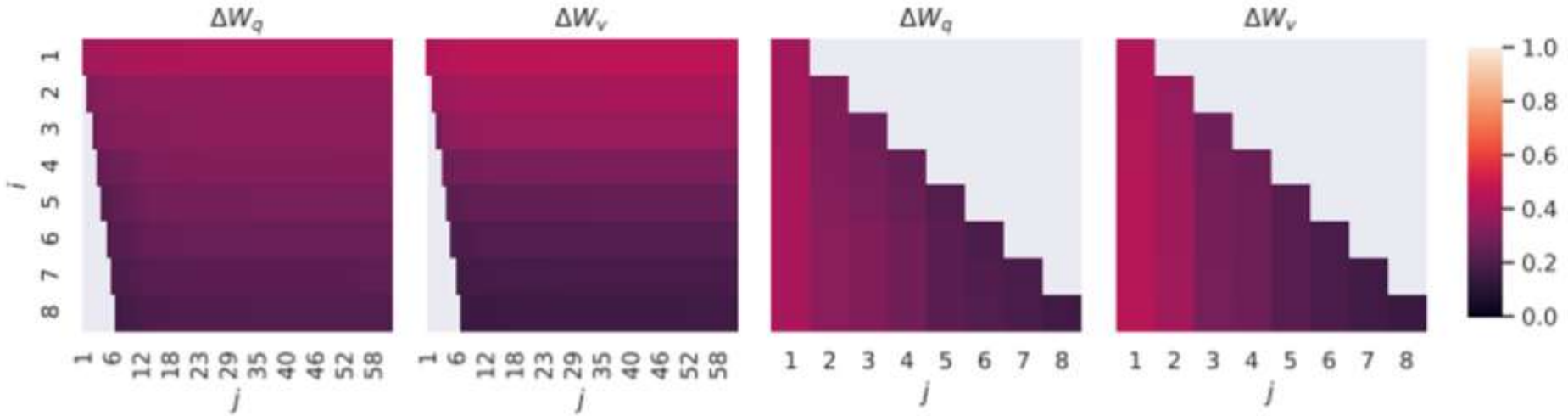
기존 모델의 **weight**는 **freeze + Adapter**를 통해 **fine-tuning ➔ LoRA**

Background (Low-Rank Adapter)

$$W_0 + \Delta W = W_0 + BA$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

$$W \in \mathbb{R}^{d \times k}, B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$



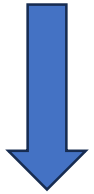
Background (Memory Requirement of PEFT)

- **LoRA adapter의 parameter 개수 : GPT-3 175B 기준 18M (=35MB)**
- **대부분의 memory requirement는 LoRA adapter의 parameter가 아니라 activation gradient 계산에 의해 발생하는 것**
- **따라서, adapter의 개수를 늘리는 것은 전체적인 memory footprint를 늘리지 않고도 성능을 높여줄 수 있다.**

QLoRA Fine-tuning (4-bit NormalFloat Quantization)

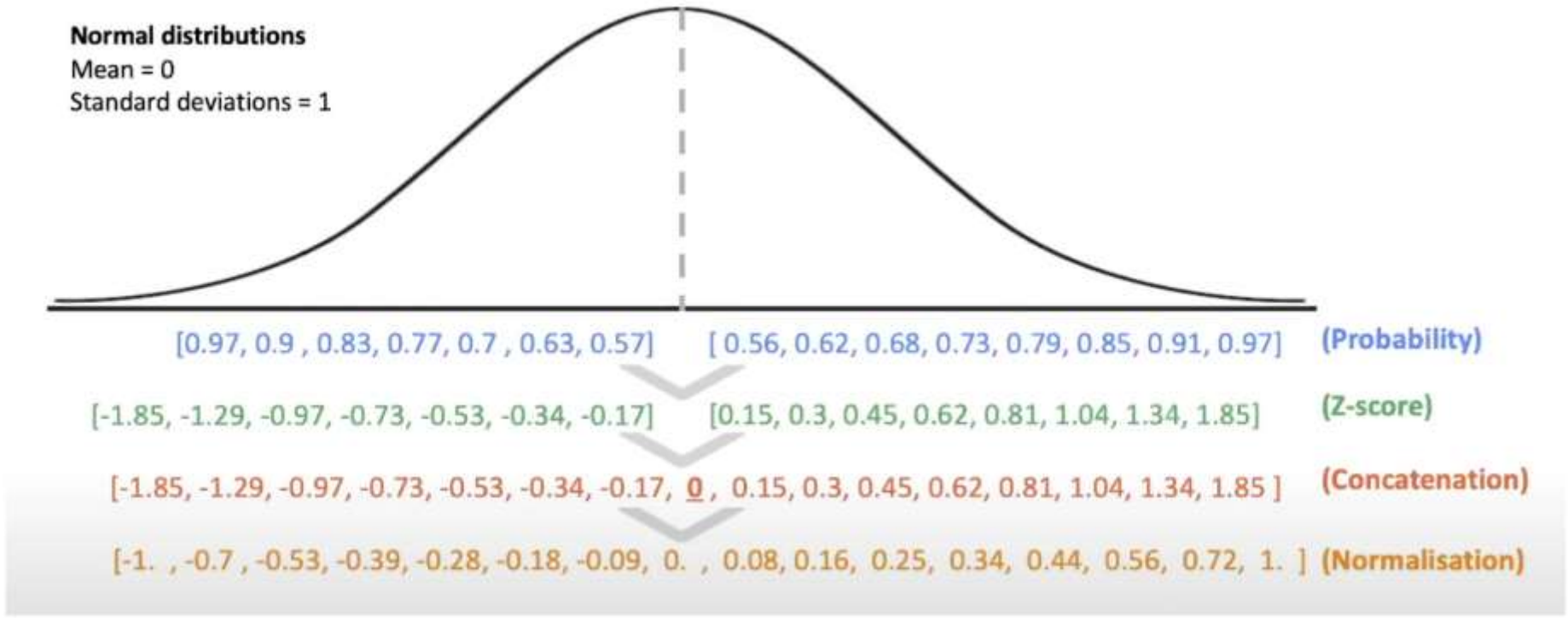
- 데이터 구간을 최솟값과 최댓값 사이로 균등하게 나누는 일반적인 **quantization**과 다르게, 데이터 분포를 고려하여 구간을 나누는 **quantile quantization**을 사용

$$q_i = \frac{1}{2}\left(Qx\left(\frac{i}{2^k+1}\right) + Qx\left(\frac{i+1}{2^k+1}\right)\right),$$



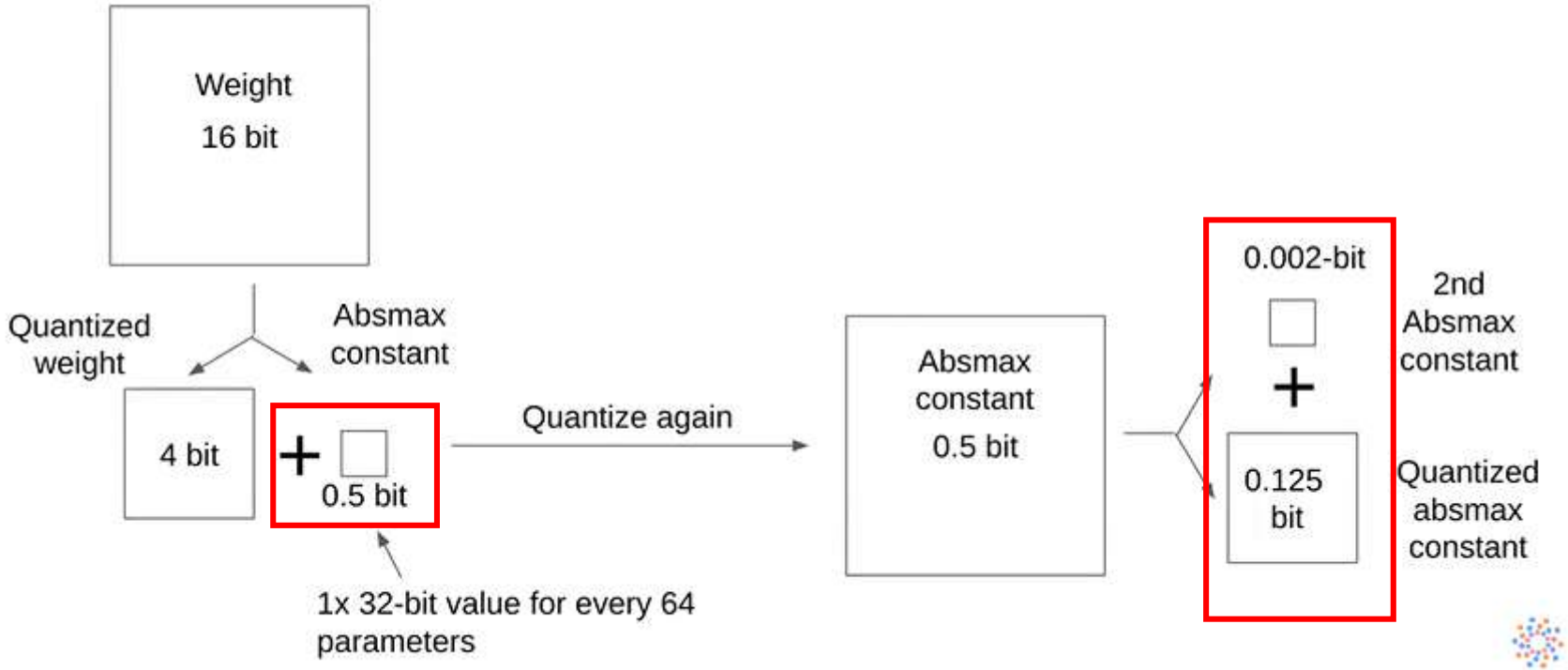
[-1.0, -0.6961928009986877, -0.5250730514526367,
-0.39491748809814453, -0.28444138169288635, -0.18477343022823334,
-0.09105003625154495, 0.0, 0.07958029955625534, 0.16093020141124725,
0.2461123019456863 4, 0.33791524171829224, 0.44070982933044434,
0.5626170039176941, 0.7229568362236023, 1.0]

QLoRA Fine-tuning (4-bit NormalFloat Quantization)



Quantiles 추정 → [-1,1] range로 normalization → input weight tensor를 abs max로 rescaling하여 [-1,1] range로 normalization → quantize

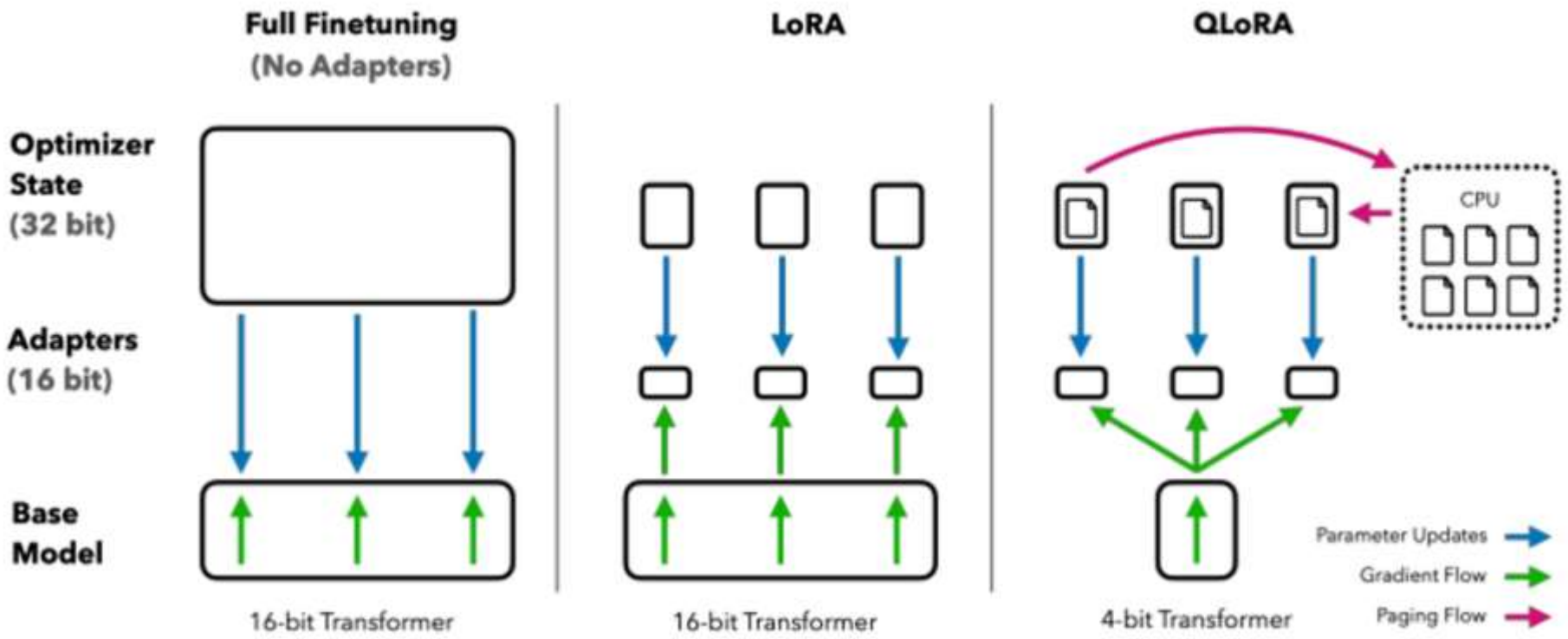
QLoRA Fine-tuning (Double Quantization)



First quantization → C_2^{FP32} → Second Quantization → C_1^{FP32}

따라서 0.5bits/param → 0.127 bits/param의 메모리 절감

QLoRA Fine-tuning (Paged Optimizers)



GPU의 Out-Of-Memory (OOM) 문제로 오류가 생기는 상황을 방지하기 위해

Page-to-page transfer를 automatically하게 만드는 방법

QLoRA Fine-tuning (Double Quantization)

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}}\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}}, \tag{5}$$

where doubleDequant$(\cdot)$ is defined as:

$$\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{k-bit}}) = \text{dequant}(\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}), \mathbf{W}^{\text{4bit}}) = \mathbf{W}^{\text{BF16}}, \tag{6}$$

총 2가지의 data type을 사용

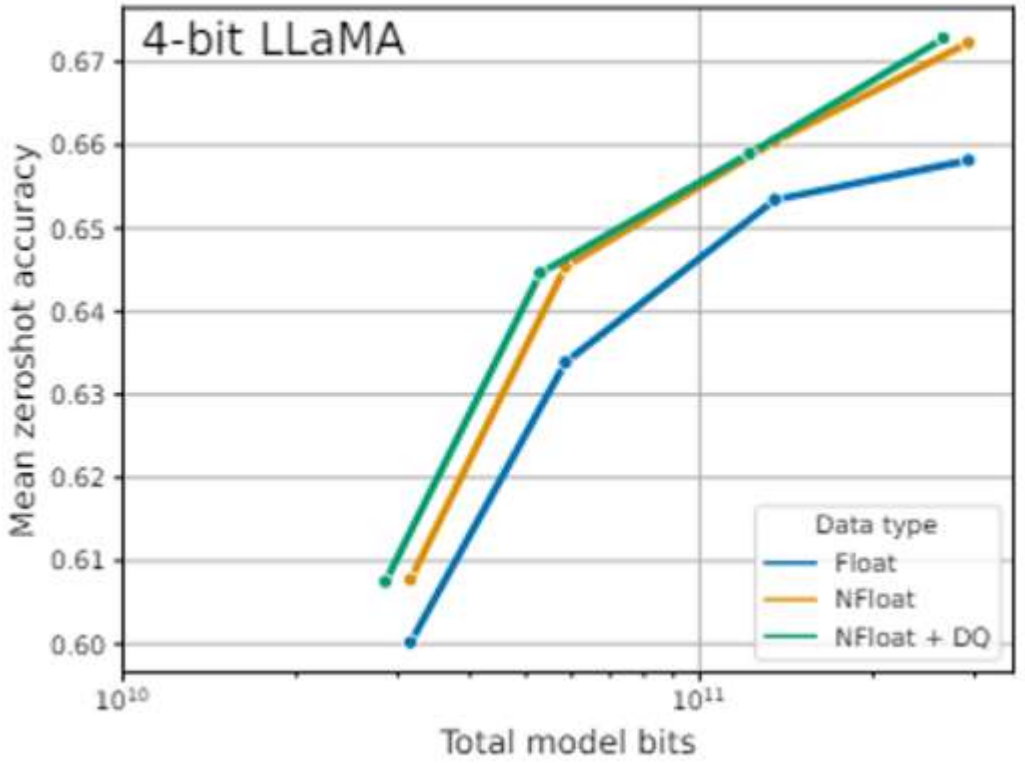
1. **4-bit NormalFloat : 데이터를 효율적으로 저장하는 역할**
2. **16-bit BrainFloat : 더 정확한 계산 효율을 위한 역할**

Experiments

- **NF4를 사용하는 것이 일반적인 FP4와 Int4를 사용하는 것보다 더 좋은 성능을 보여준다는 것을 확인할 수 있다.**

Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

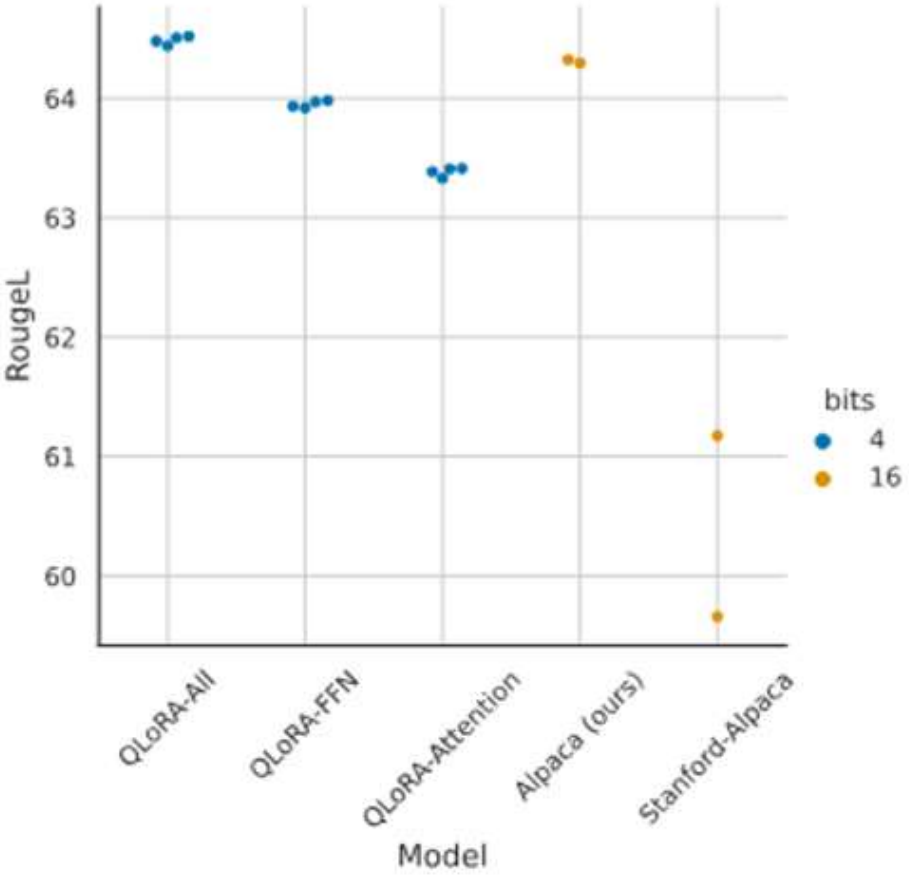
Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9



Experiments

Table 4: Mean 5-shot MMLU test accuracy for LLaMA 7-65B models finetuned with adapters on Alpaca and FLAN v2 for different data types. Overall, NF4 with double quantization (DQ) matches BFloat16 performance, while FP4 is consistently one percentage point behind both.

	Mean 5-shot MMLU Accuracy								
LLaMA Size	7B		13B		33B		65B		Mean
Dataset	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	Alpaca	FLAN v2	
BFloat16	38.4	45.6	47.2	50.6	57.7	60.5	61.8	62.5	53.0
Float4	37.2	44.0	47.3	50.0	55.9	58.5	61.3	63.3	52.2
NFloat4 + DQ	39.0	44.5	47.5	50.7	57.3	59.2	61.8	63.9	53.1



Adapter를 더 많이 사용하여 부정확한 quantization으로 인한 performance lost를 보완

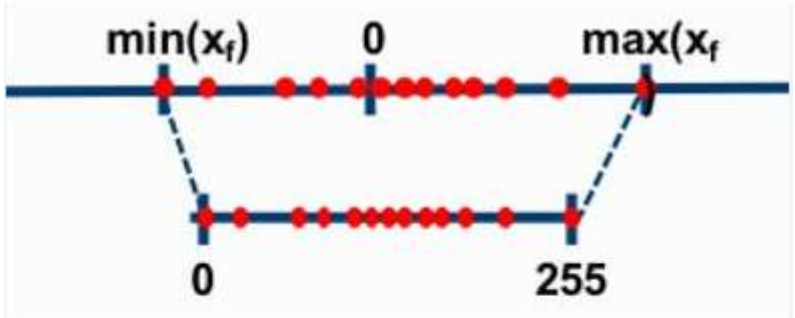
Conclusion

4-bit base model + LoRA를 통해 16-bit full fine-tuning performance를 재현

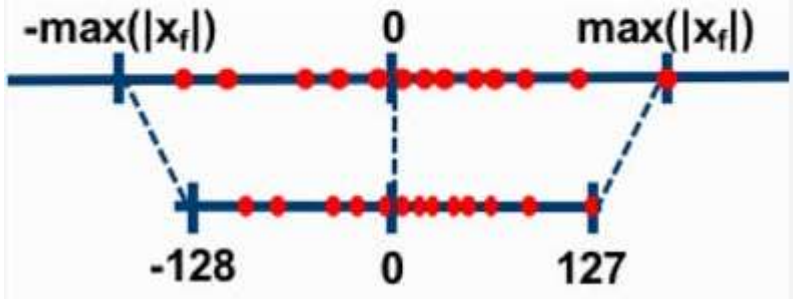
[Limitation]

- **33B & 65B scale에서는 full fine-tuning의 performance를 재현하기 어려웠음**
- **Fine-tuning model의 evaluation 방법에 대한 의문성 제기**
- **다른 bit-precision으로 평가해보지 않았다는 점**

Appendix



Asymmetric Mode



Symmetric Mode

Q&A

Thank you