LoQT: Low-Rank Adapters for Quantized Pre-Training

Sebastian Loeschcke*, Mads Toftrup*, Michael J. Kastoryano,
Serge Belongie, and Vésteinn Snæbjarnarson

NeurIPS 2024

UNIVERSITY OF COPENHAGEN

AARHUS UNIVERSITY

PIONEER CENTRE FOR ARTIFICIAL INTELLIGENCE

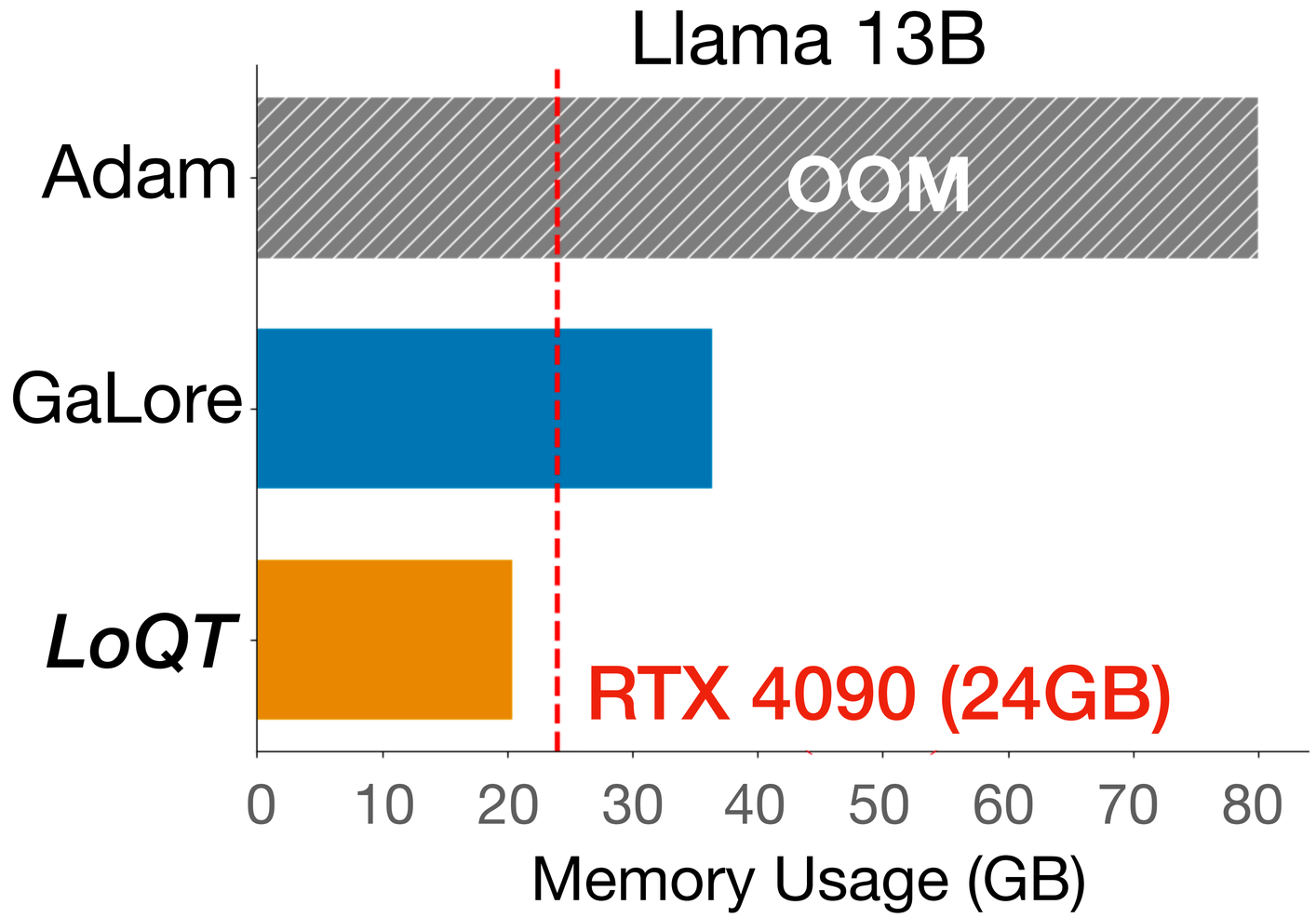
*Equal Contribution

Pre-training LLMs require big GPU clusters

Pre-training a **LLaMA 13B** model with Adam and a *batch size of one* requires over **100 GB** of memory.

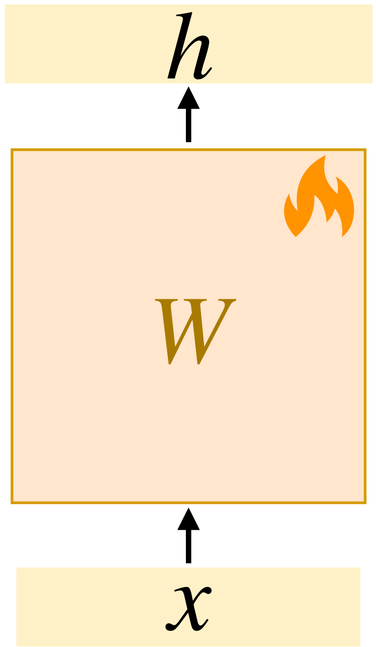
Pre-training LLMs require big GPU clusters

Pre-training a **LLaMA 13B** model with Adam and a *batch size of one* requires over **100 GB** of memory.



LoQT: Low-Rank Quantized Pre-Training

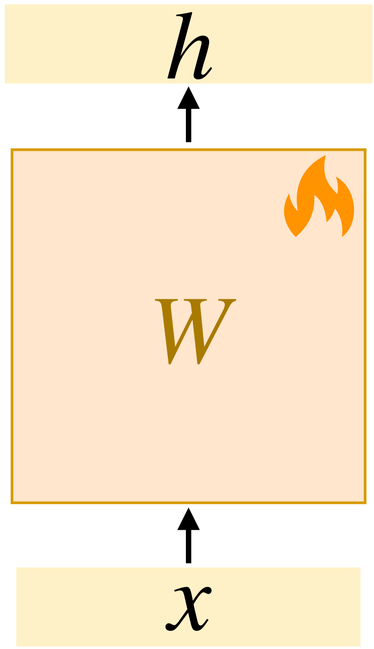
Standard Linear Layer



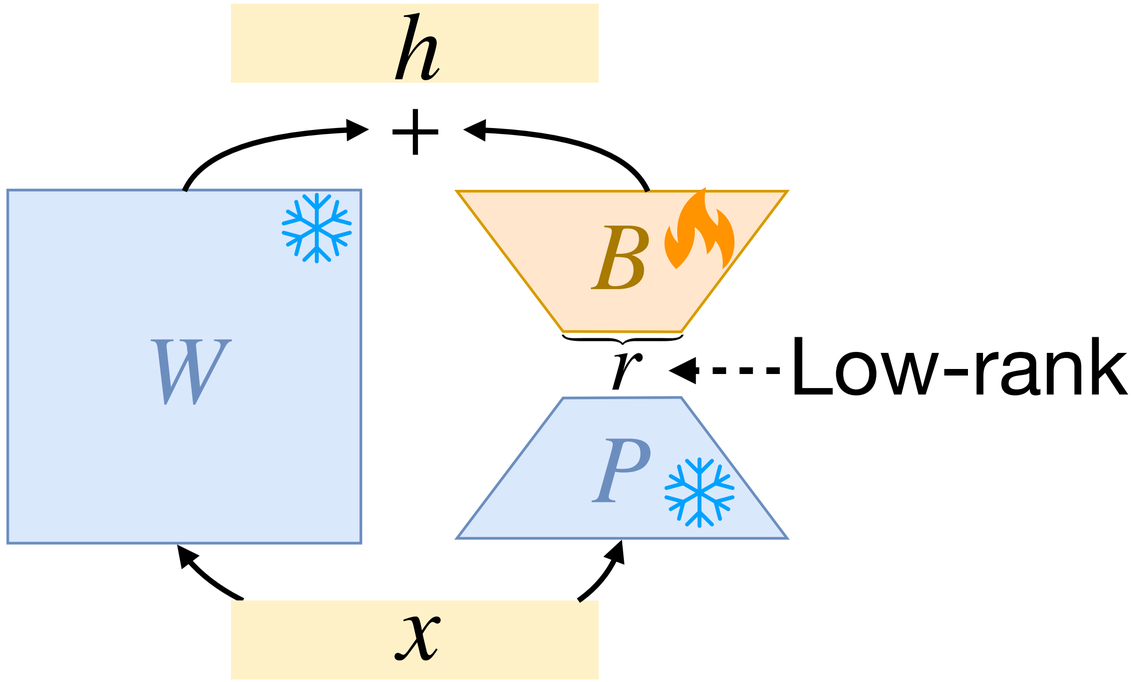
Optimize ❄ Freeze

LoQT: Low-Rank Quantized Pre-Training

Standard Linear Layer



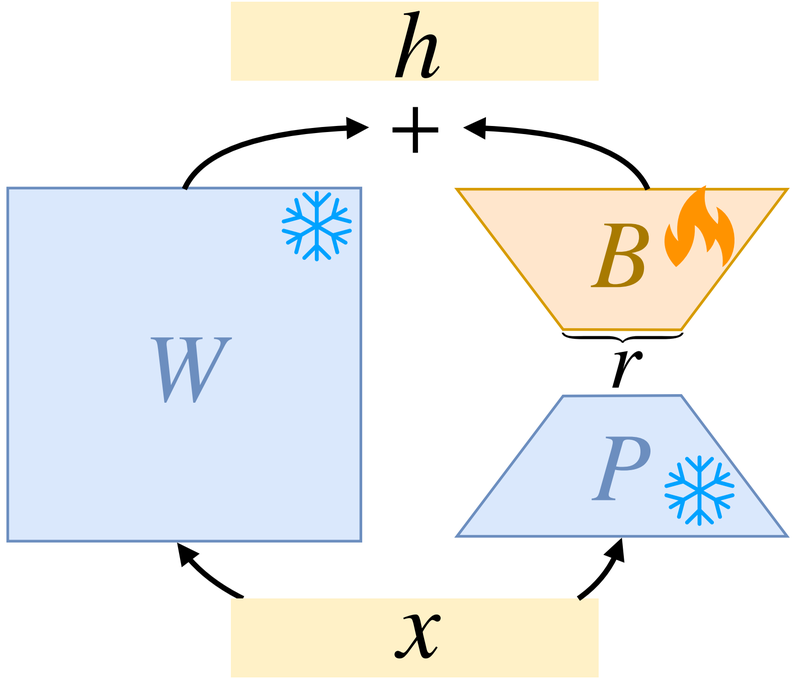
Replace



Optimize ❄ Freeze

LoQT: Low-Rank Quantized Pre-Training

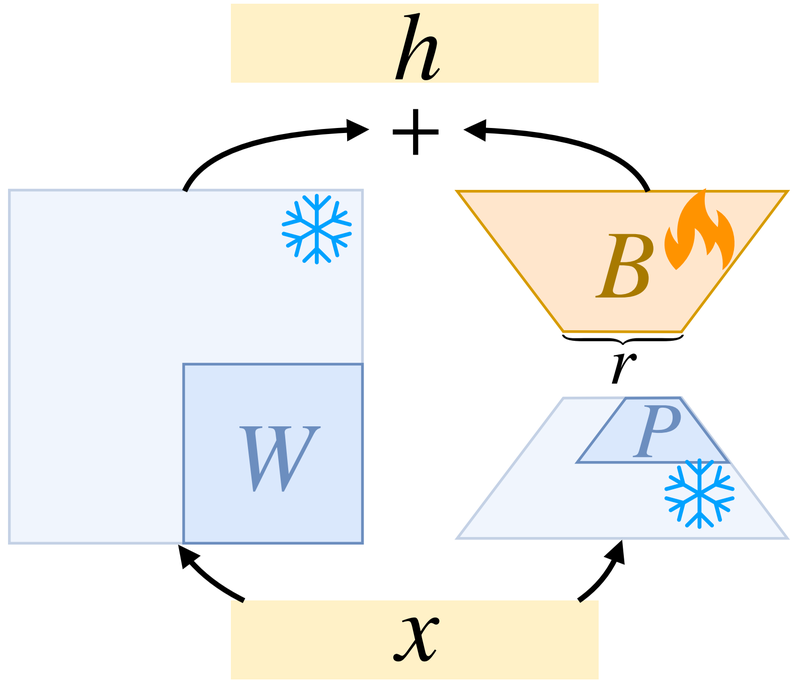
- **Pre-training from scratch** while only *optimizing a **single low-rank factor** per layer.*



Optimize Freeze

LoQT: Low-Rank Quantized Pre-Training

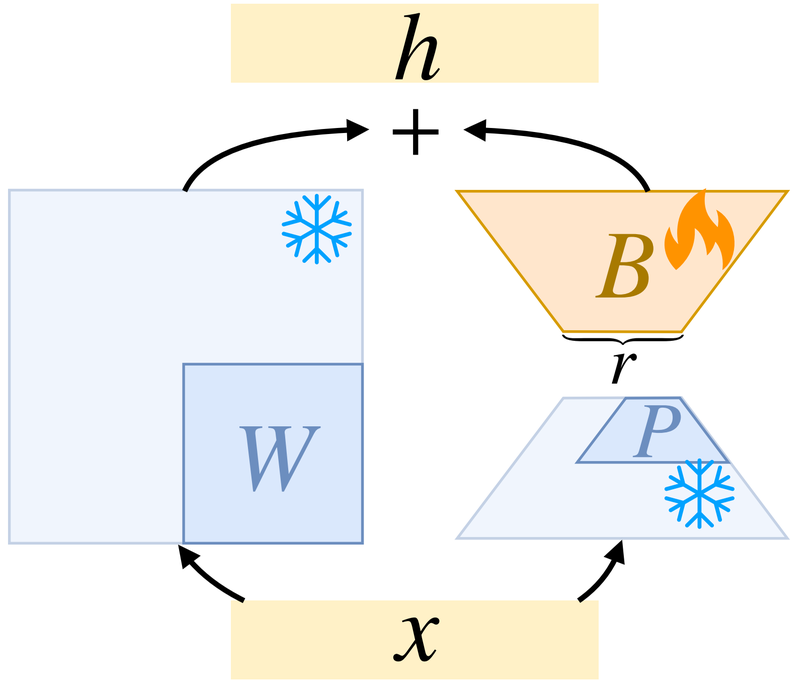
- **Pre-training from scratch** while only *optimizing a **single low-rank factor** per layer.*
- Keep most *weights in **4-bit precision.***



Optimize ❄ Freeze

LoQT: Low-Rank Quantized Pre-Training

- **Pre-training from scratch** while only *optimizing a **single low-rank factor** per layer.*
- Keep most *weights in **4-bit precision.***
- *Matches regular FP16 training*

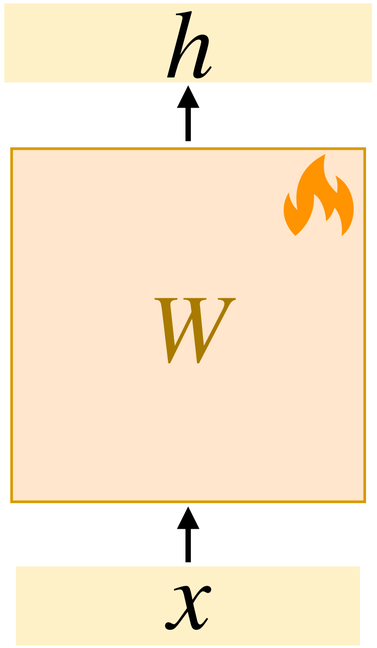


Optimize ❄ Freeze

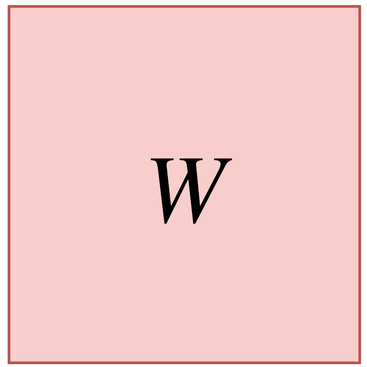
Training with the Adam Optimizer

Memory Requirements

Linear Layer

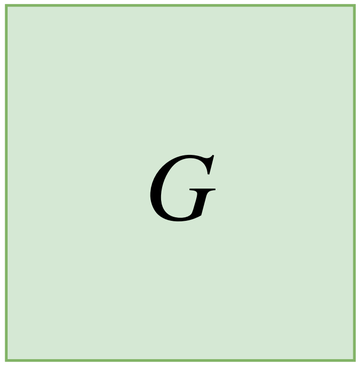


Weights



mn

Gradients



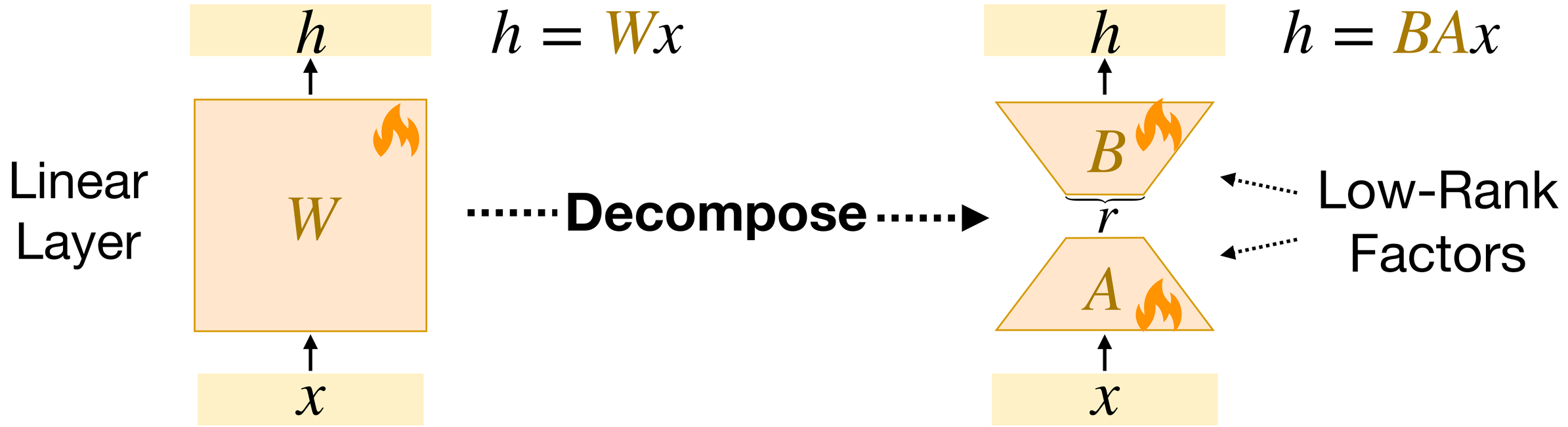
mn

Optimizer States

$2mn$

Total Memory: $4mn$

Low-Rank Training



Requires Large Rank r

Weights are not necessarily low-rank [1]

[1] Zhao et al., “GaLore”, ICML 2024.

Low-Rank Properties of Gradients

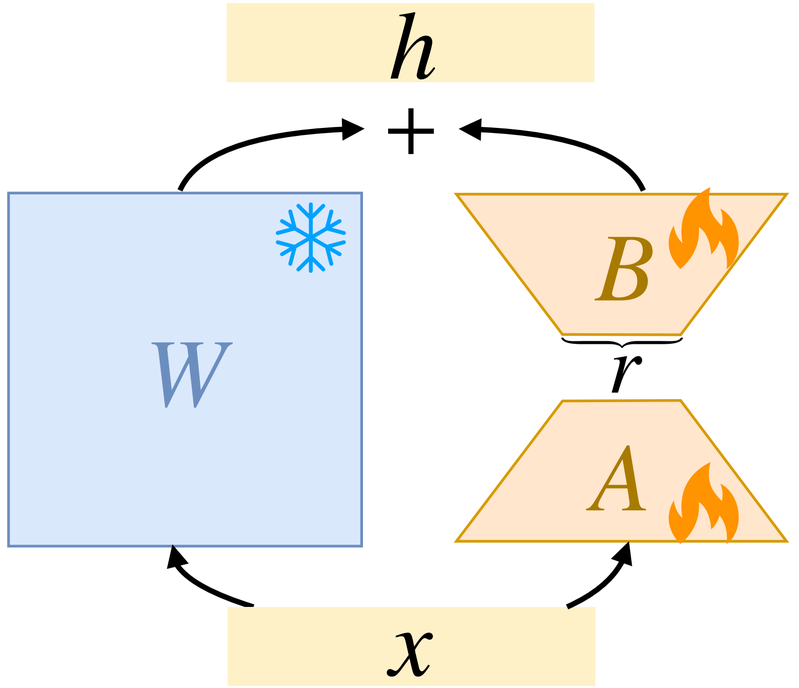
- **GaLore:** “Gradient becomes low-rank during training” [1]
- **InRank:** “Cumulative weight updates follow an incremental low-rank trajectory” [2]

[1] Zhao et al., “GaLore”, ICML 2024. [2] Zhao et al., “InRank: Incremental Low-Rank Learning”, ICLR 2023

LoRA - Low-Rank Adaption [3]

Low-Rank Cumulative Updates

Pre-trained



$h = Wx + BAx$

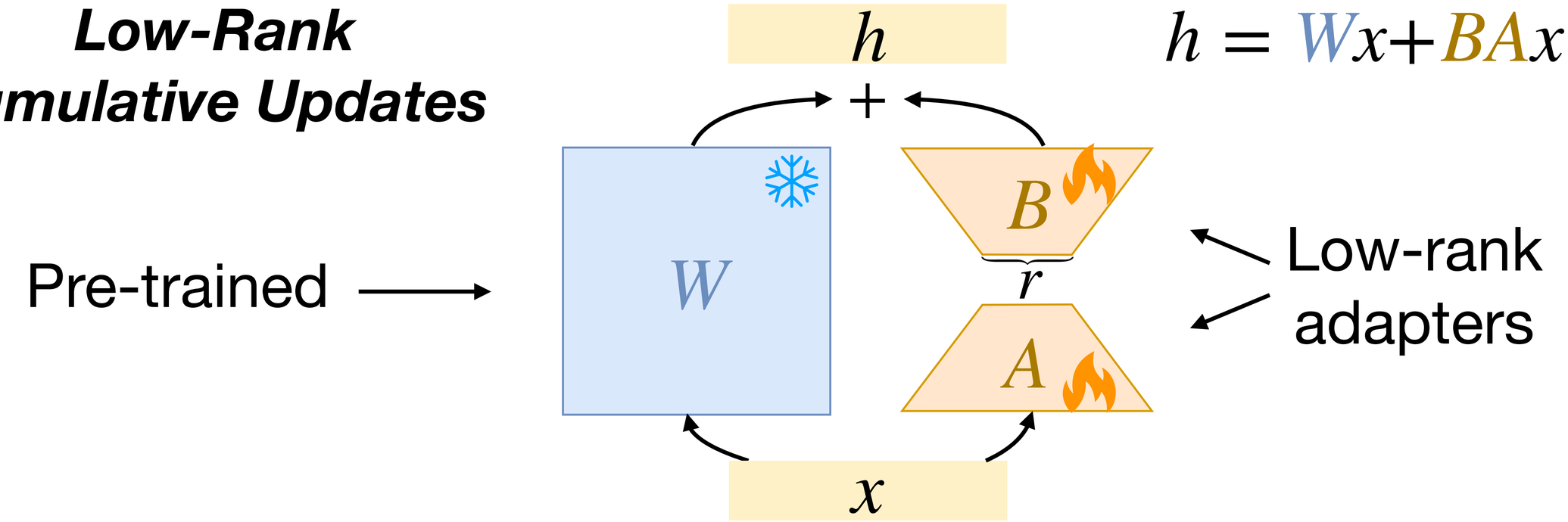
Low-rank adapters

[3] Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models”, ICLR 2022

LoRA - Low-Rank Adaption [3]

Low-Rank Cumulative Updates

$$h = Wx + BAx$$



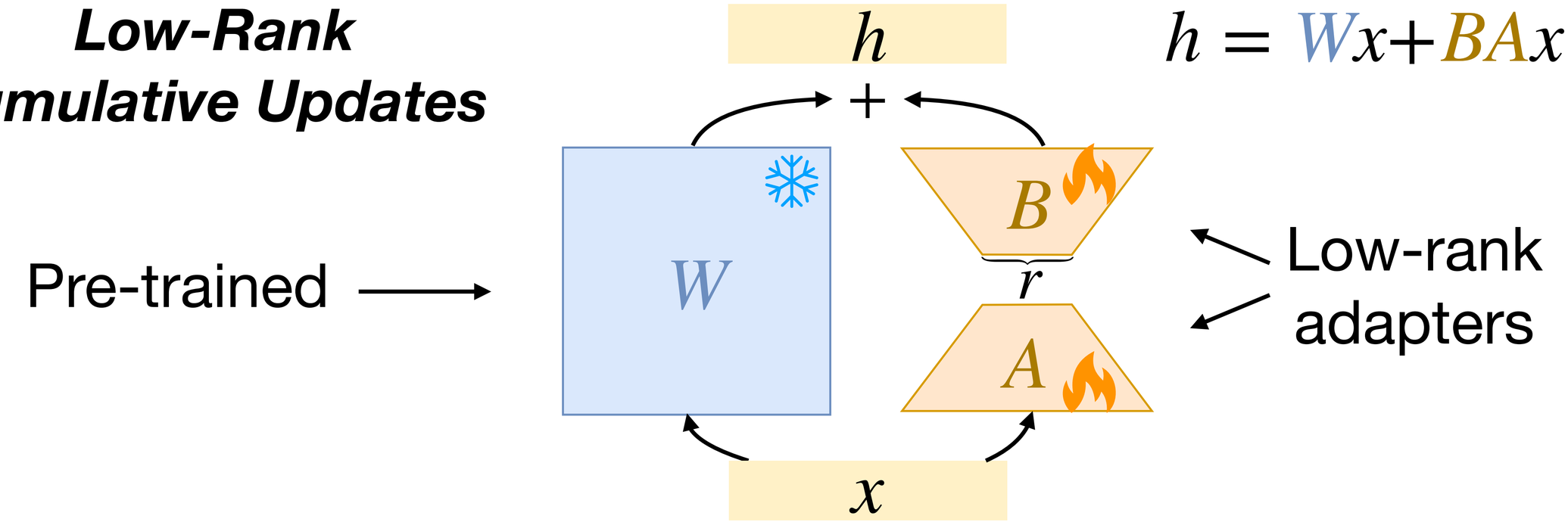
+ Reduces memory and is effective for fine-tuning

[3] Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models”, ICLR 2022

LoRA - Low-Rank Adaption [3]

Low-Rank Cumulative Updates

$$h = Wx + BAx$$



\+ Reduces memory and is effective for fine-tuning

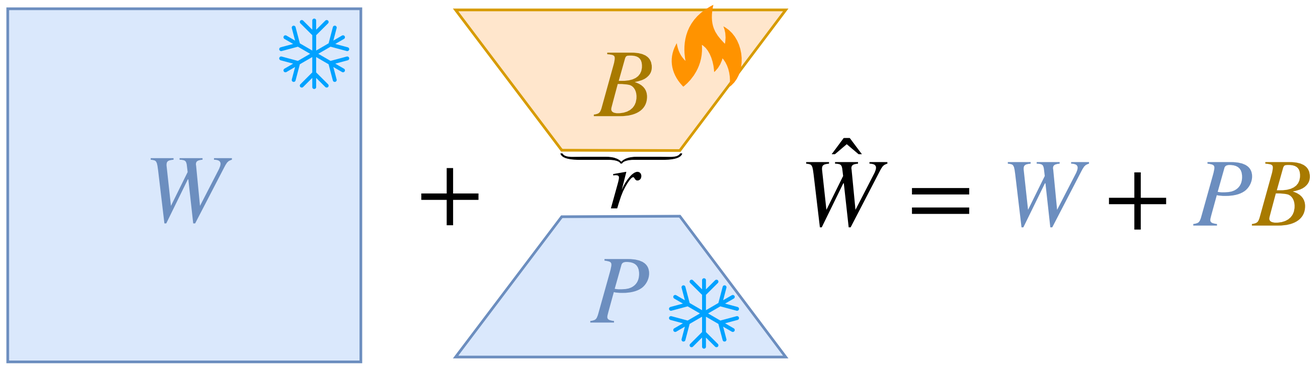
\- Does not work well for pre-training

[3] Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models”, ICLR 2022

LoQT: Low-Rank Quantized Pre-Training

Key Components

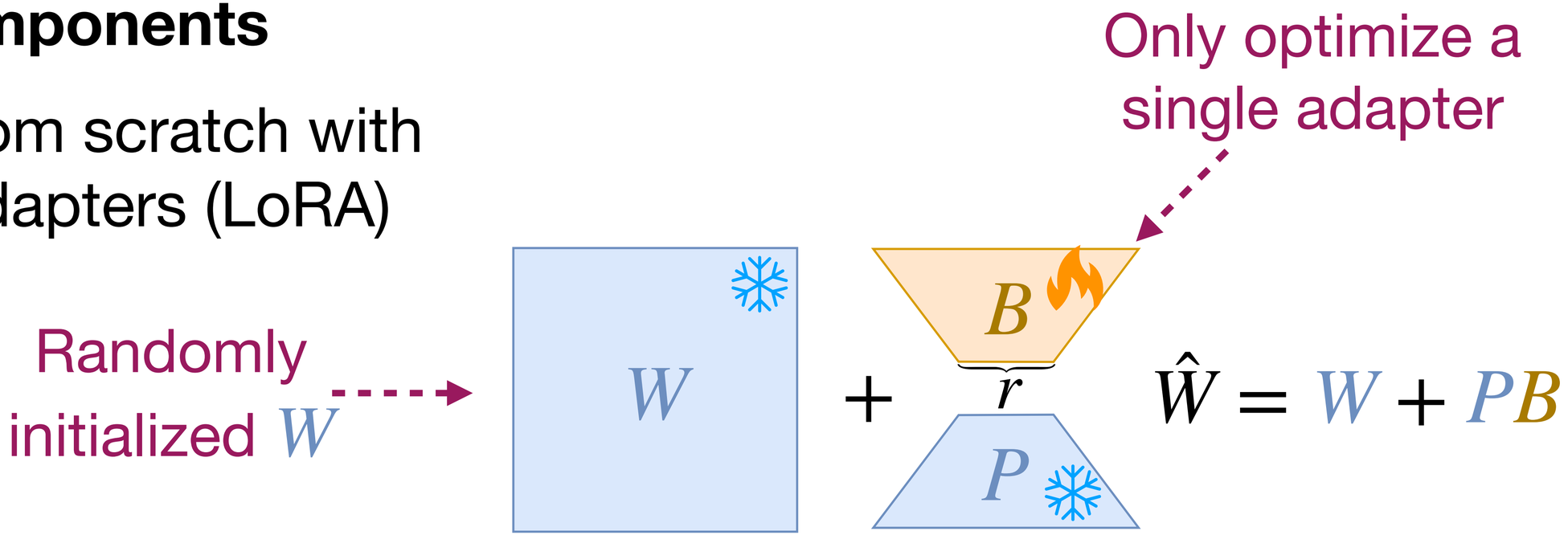
- Pre-train from scratch with low-rank adapters (LoRA)



LoQT: Low-Rank Quantized Pre-Training

Key Components

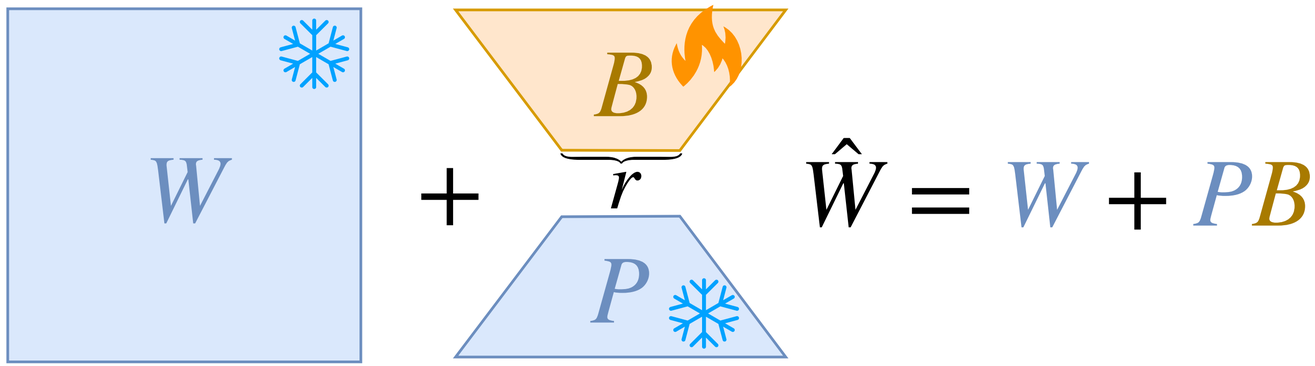
- Pre-train from scratch with low-rank adapters (LoRA)



LoQT: Low-Rank Quantized Pre-Training

Key Components

- Pre-train from scratch with low-rank adapters (LoRA)
- Initialize P using ∇W



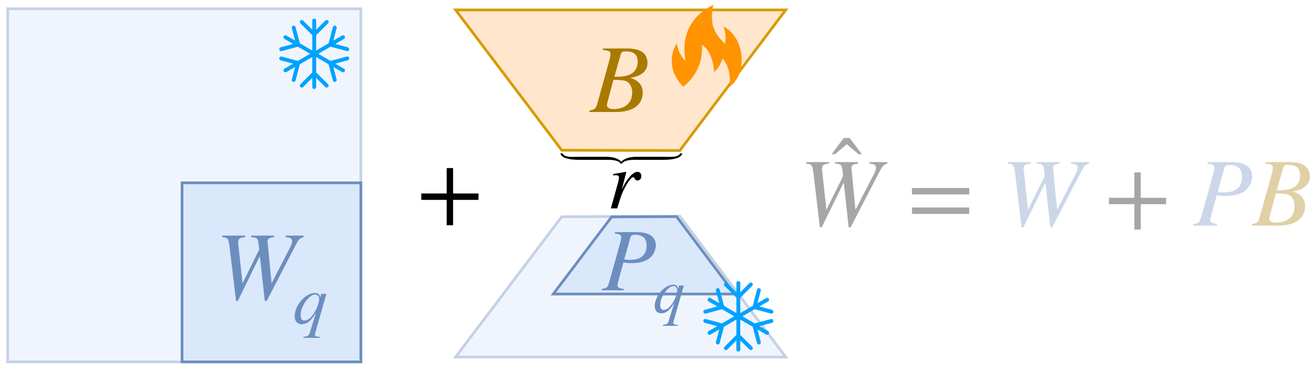
$$\text{SVD}(\nabla W) = U\Sigma V^\top$$

$$\mathbf{P} = U[:, r]$$

LoQT: Low-Rank Quantized Pre-Training

Key Components

- Pre-train from scratch with low-rank adapters (LoRA)
- Initialize P using ∇W
- Quantize P and W to 4-bit



$$\text{SVD}(\nabla W) = U\Sigma V^\top$$

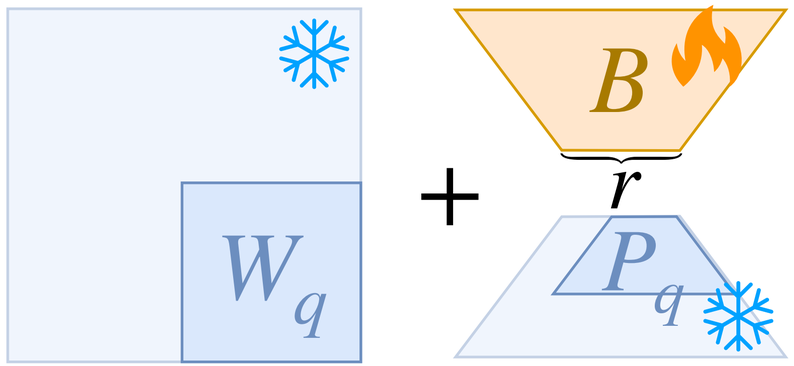
$$\mathbf{P} = U[:, r]$$

LoQT: Low-Rank Quantized Pre-Training

Key Components

- Pre-train from scratch with low-rank adapters (LoRA)
- Initialize P using ∇W
- Quantize P and W to 4-bit
- Initialize B using a *quantization error compensation* term.

$$B = P^{-1}(W_q - W)$$

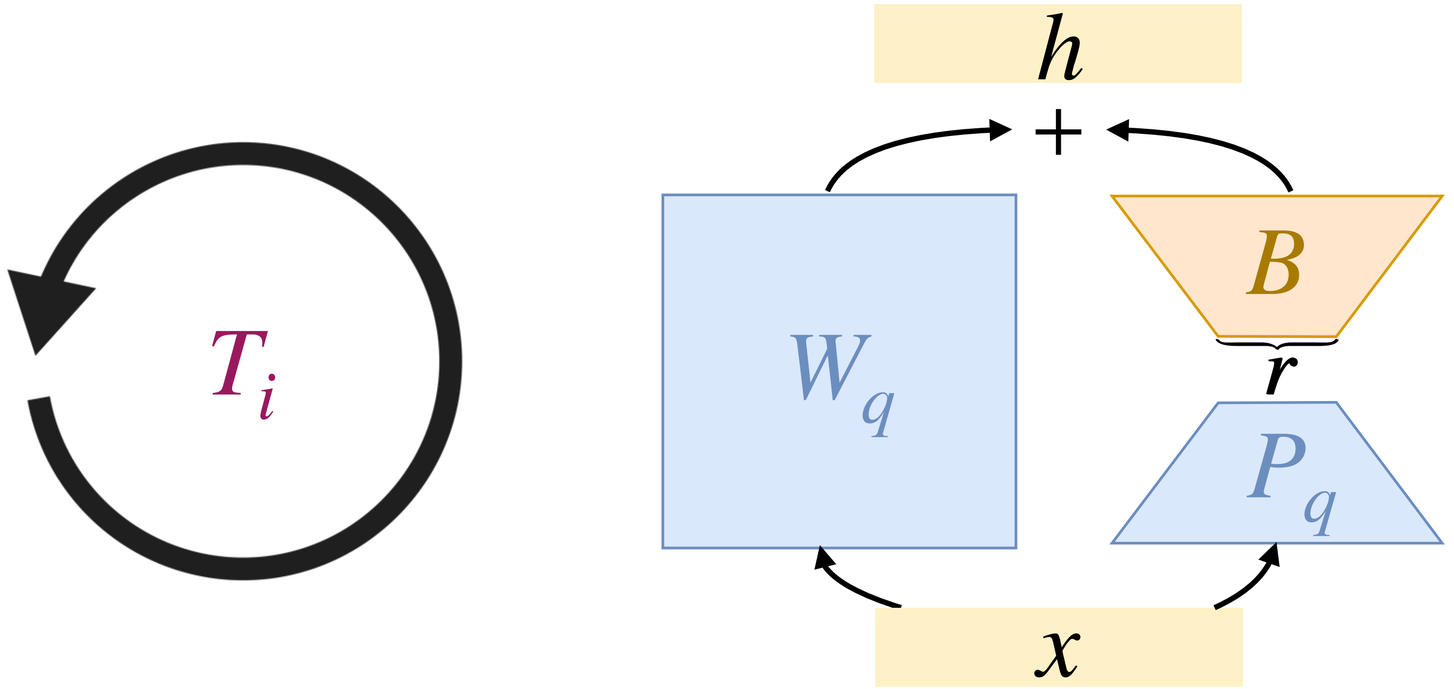


$$\hat{W} = W + PB$$

$$\text{SVD}(\nabla W) = U\Sigma V^\top$$

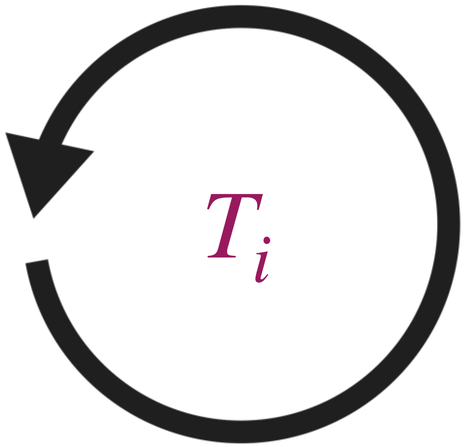
$$\mathbf{P} = U[:, r]$$

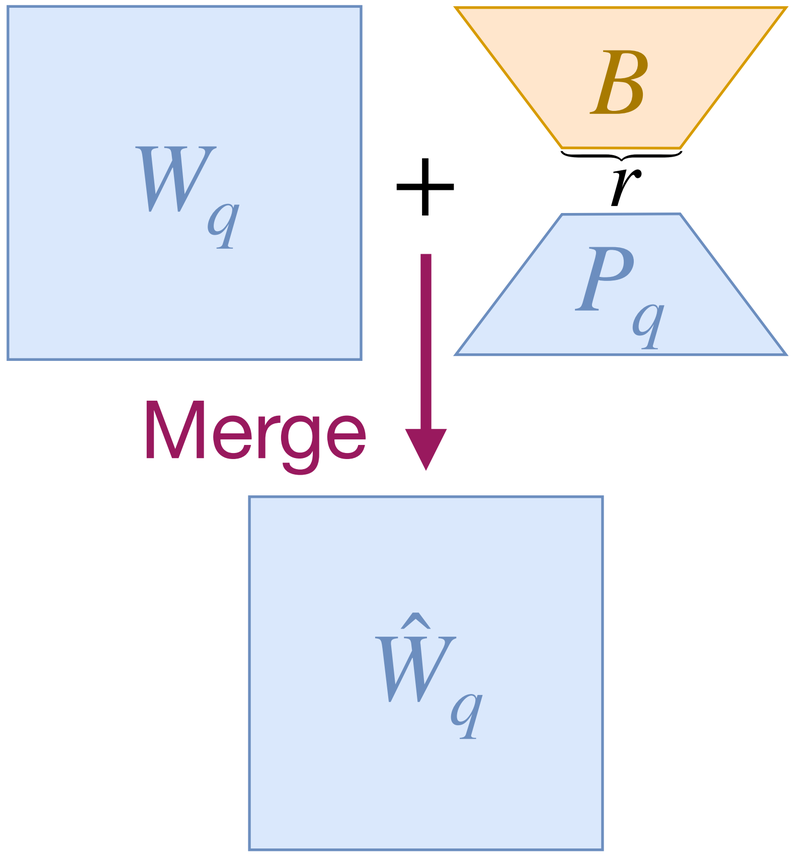
LoQT - Training



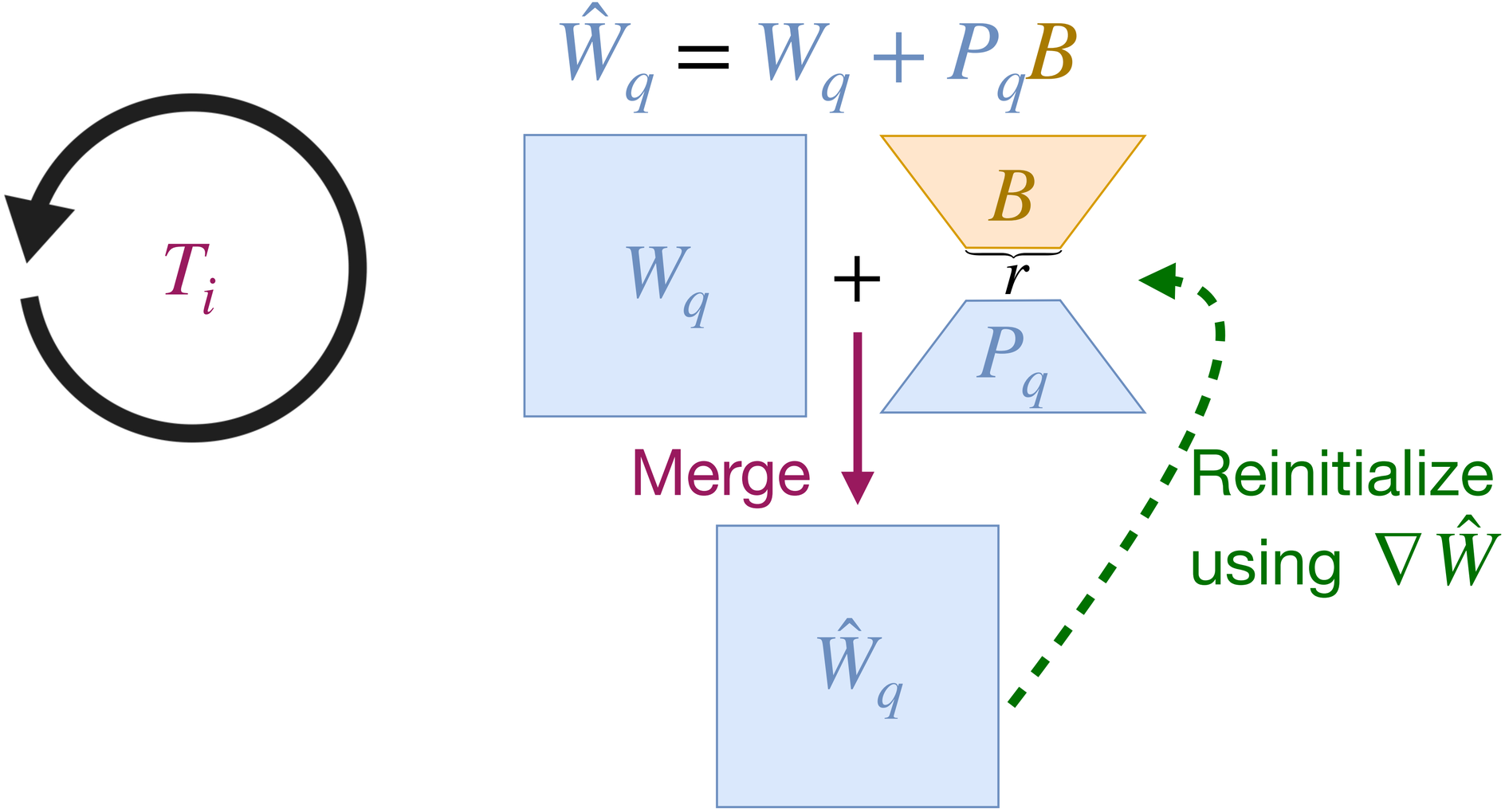
LoQT - Merge and Reinitialize

$$\hat{W}_q = W_q + P_q B$$

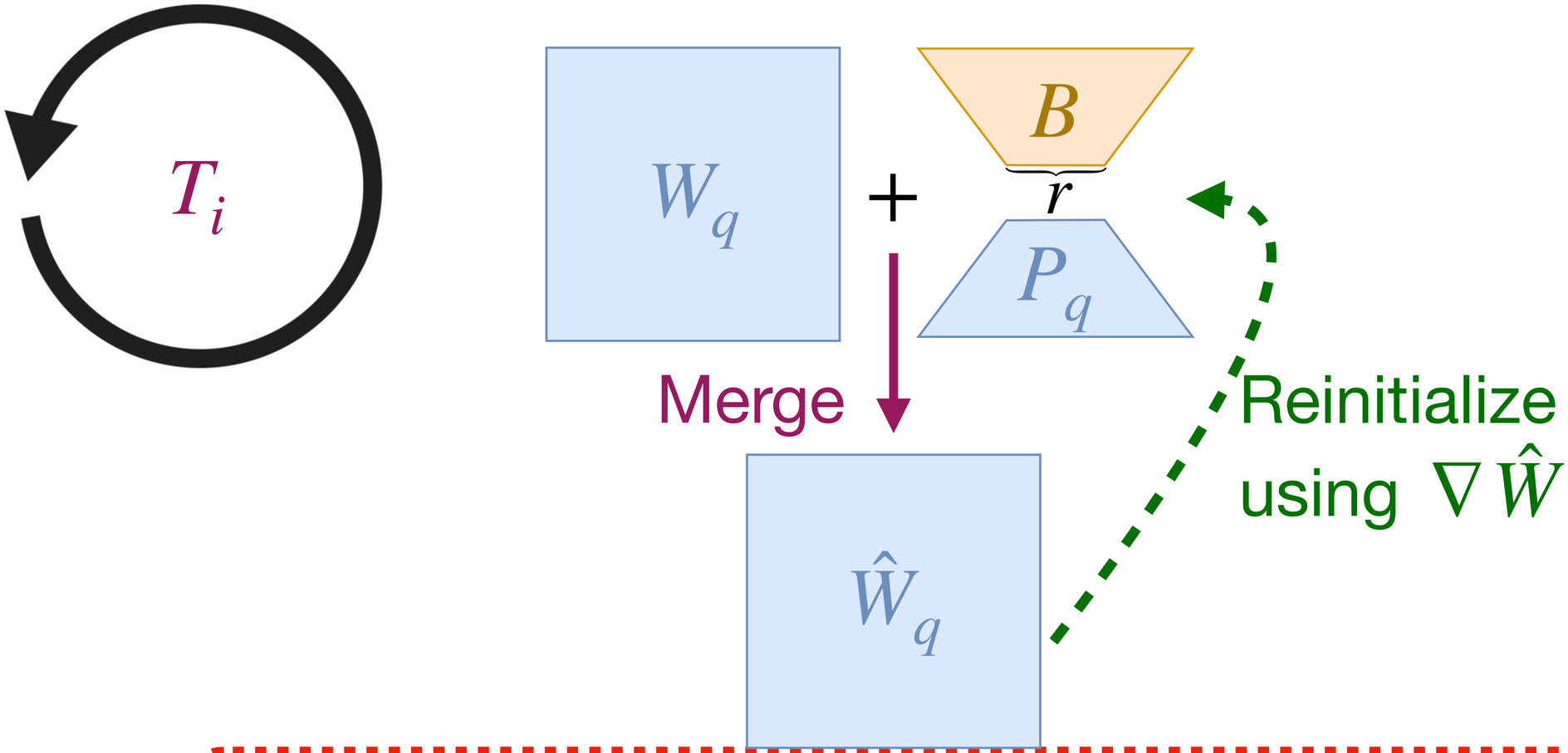




LoQT - Merge and Reinitialize

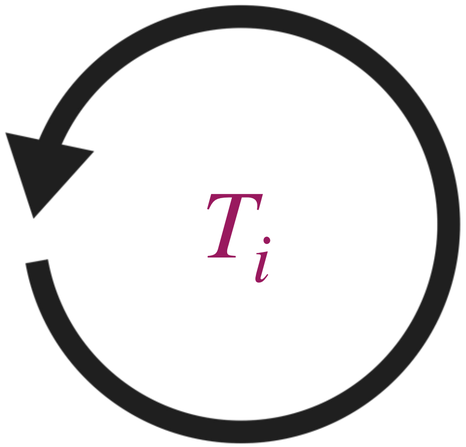


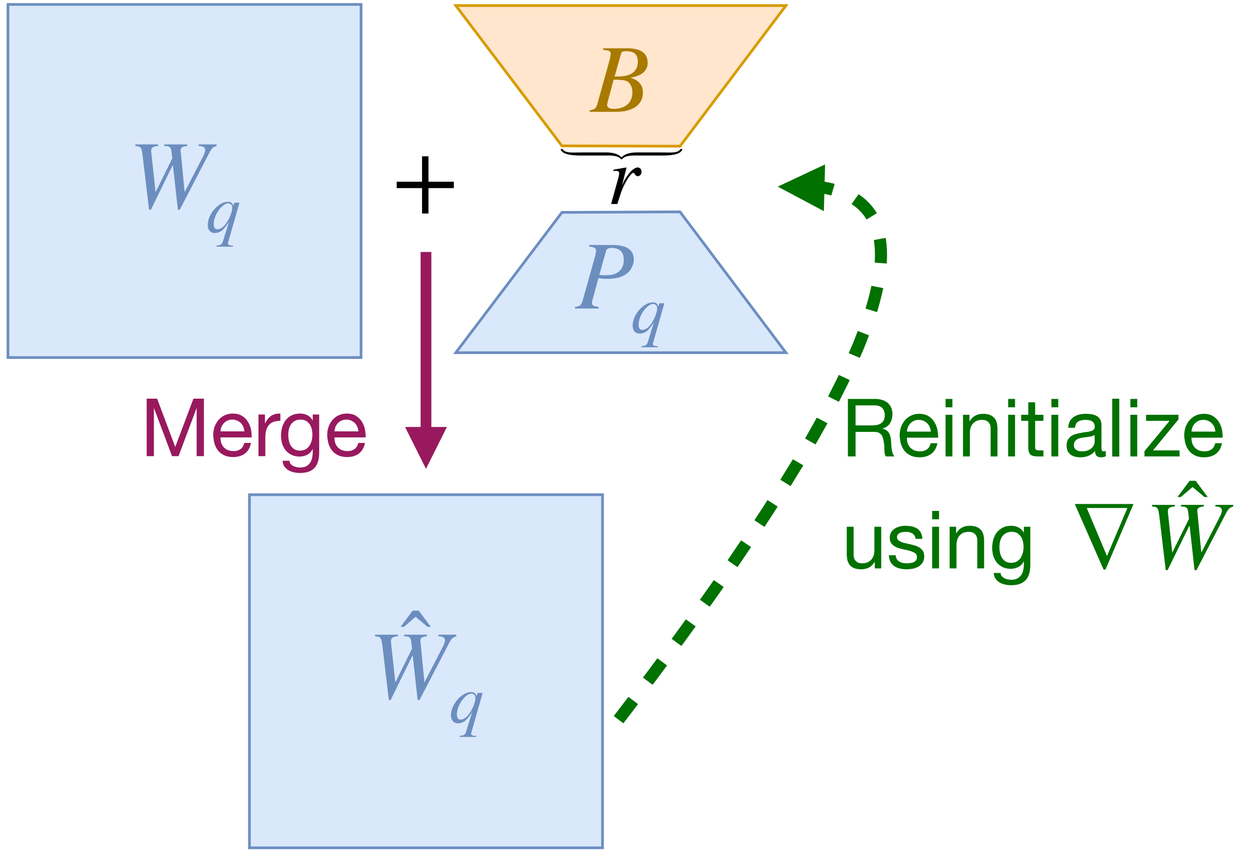
LoQT - Merge and Reinitialize



$$W_N = W_0 + P_0 B_0 + P_1 B_1 + \ldots + P_N B_N$$

LoQT - Merge and Reinitialize





$$T_i = \tau + \psi^i$$

$$W_N = W_0 + P_0 B_0 + P_1 B_1 + \ldots + P_N B_N$$

Memory Requirements with Adam

Regular Training

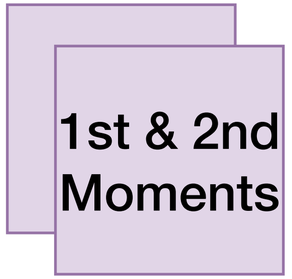
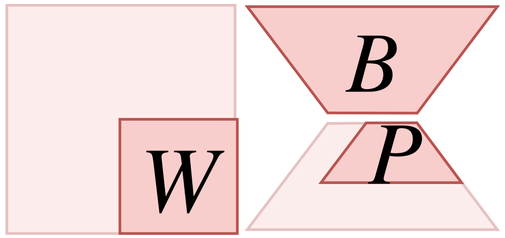
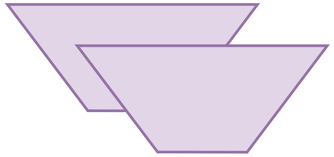
$$h = Wx$$

LoQT

$$h = Wx + PBx$$

Good

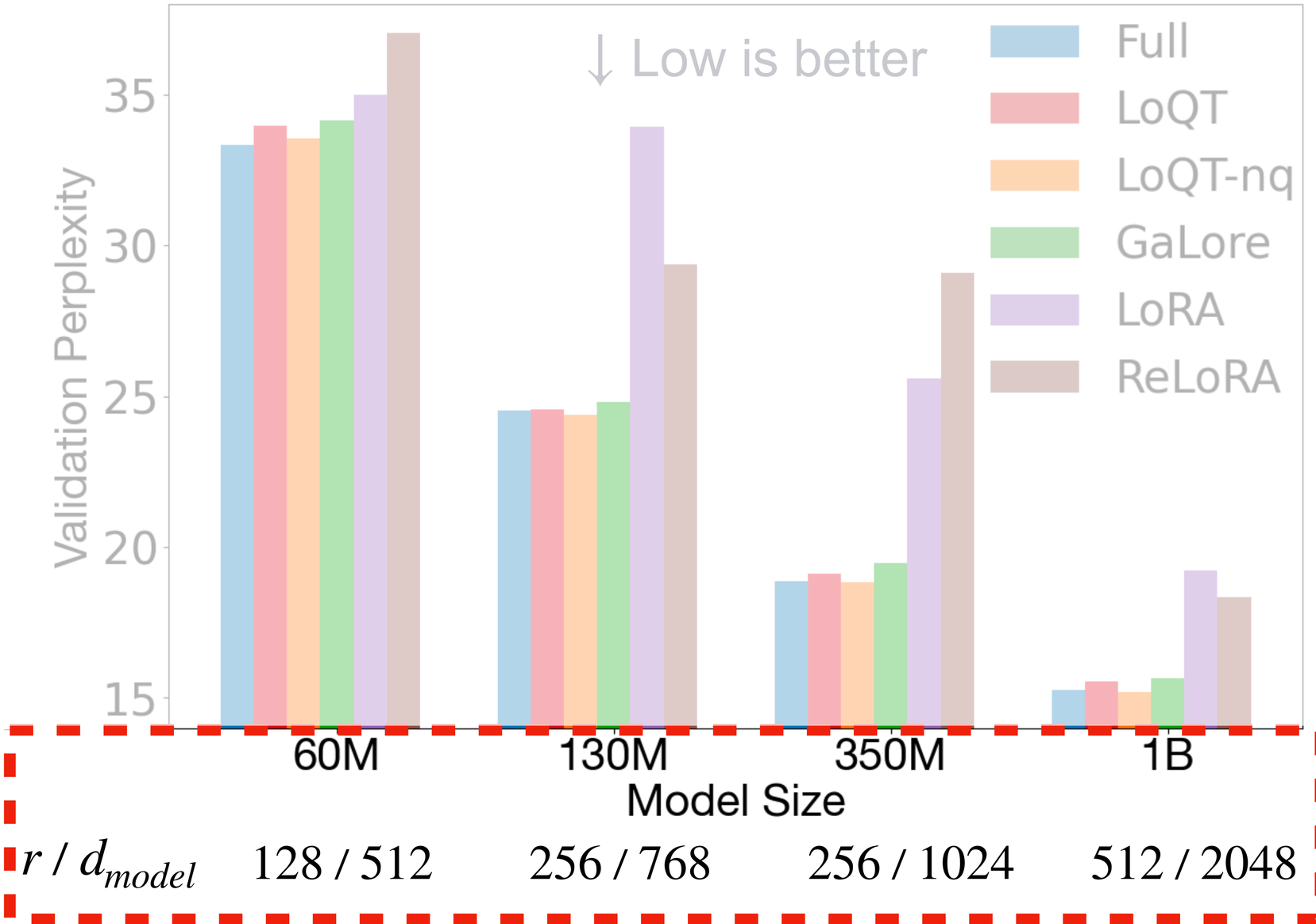
if $r \ll \min(m, n)$

	Weights	Gradients	Optimizer States
Regular Training			
	mn	mn	$2mn$
LoQT			
	$m_q n_q + nr + m_q r$	nr	$2nr$

$\frac{m}{r}$

Results

Pre-training LLaMA2 Models on the C4 Dataset



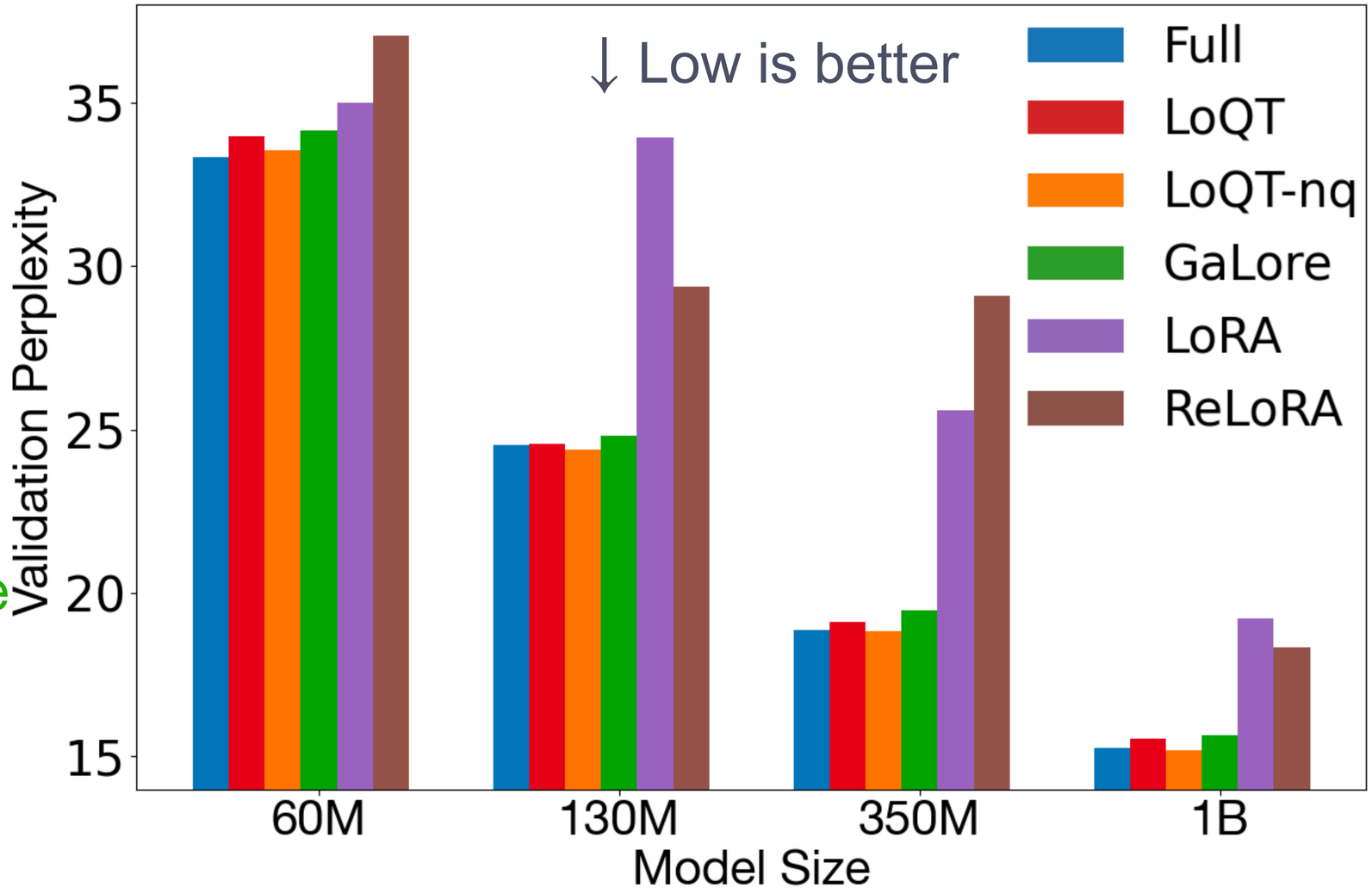
Pre-training LLaMA2 Models on the C4 Dataset

Takeaways

- LoQT is very comparable to Full training and GaLore

LoQT - 1B model

61% less memory than Full
41% less memory than GaLore

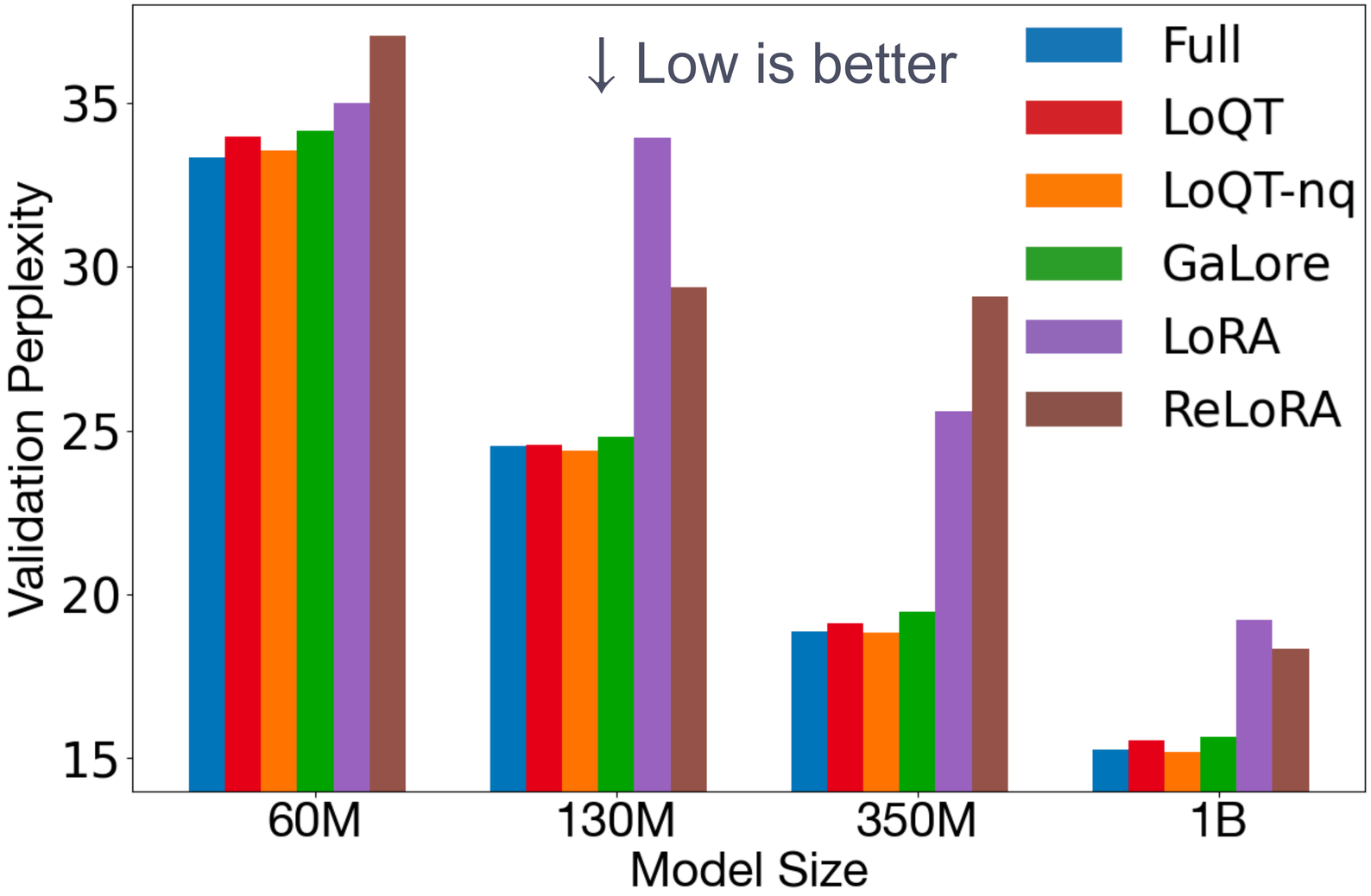


r / d_{model}	128 / 512	256 / 768	256 / 1024	512 / 2048

Pre-training LLaMA2 Models on the C4 Dataset

Takeaways

- LoQT is very comparable to Full training and GaLore
- No-quantization (LoQT-nq) is slightly better than LoQT

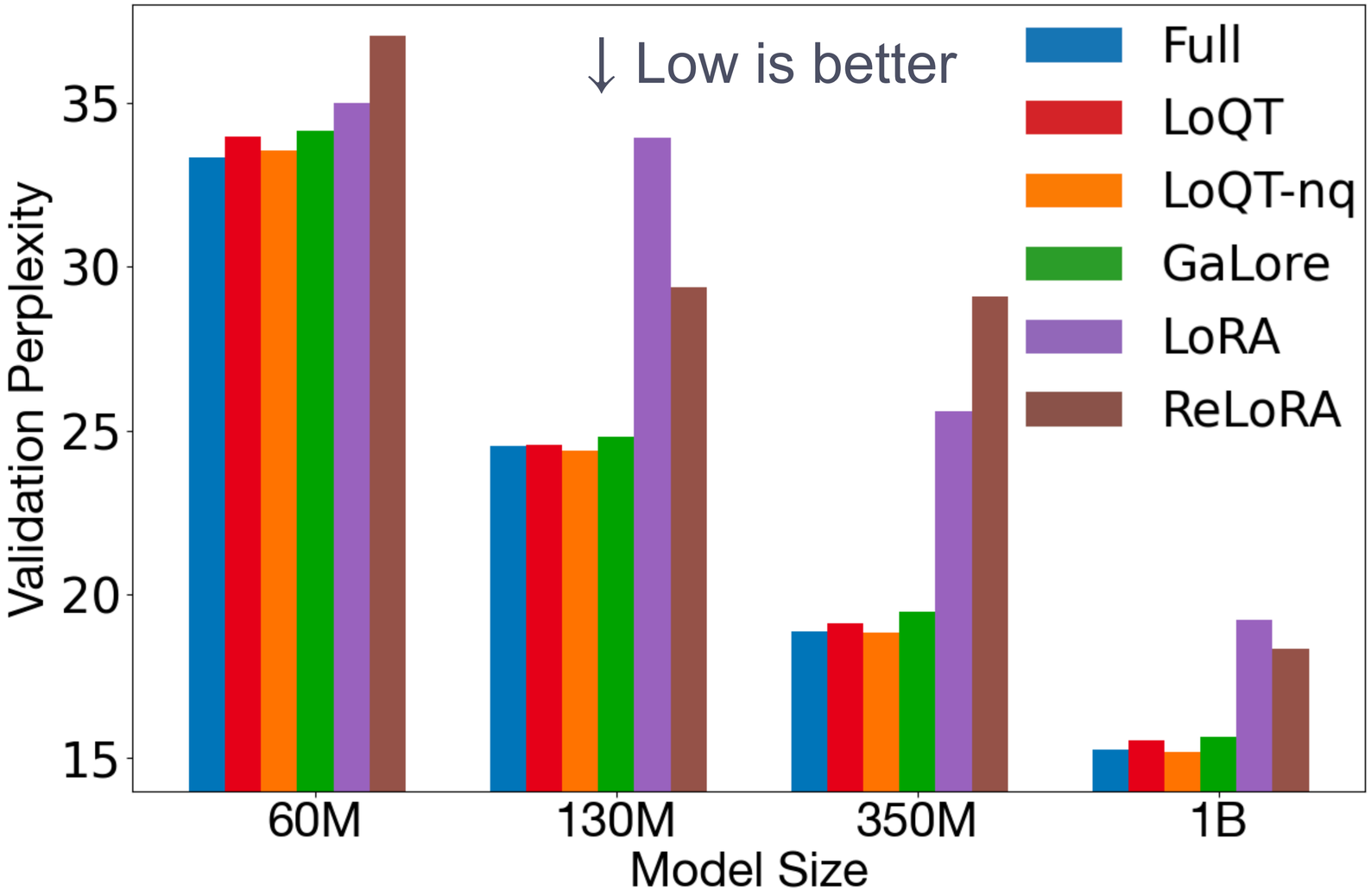


r / d_{model}	128 / 512	256 / 768	256 / 1024	512 / 2048

Pre-training LLaMA2 Models on the C4 Dataset

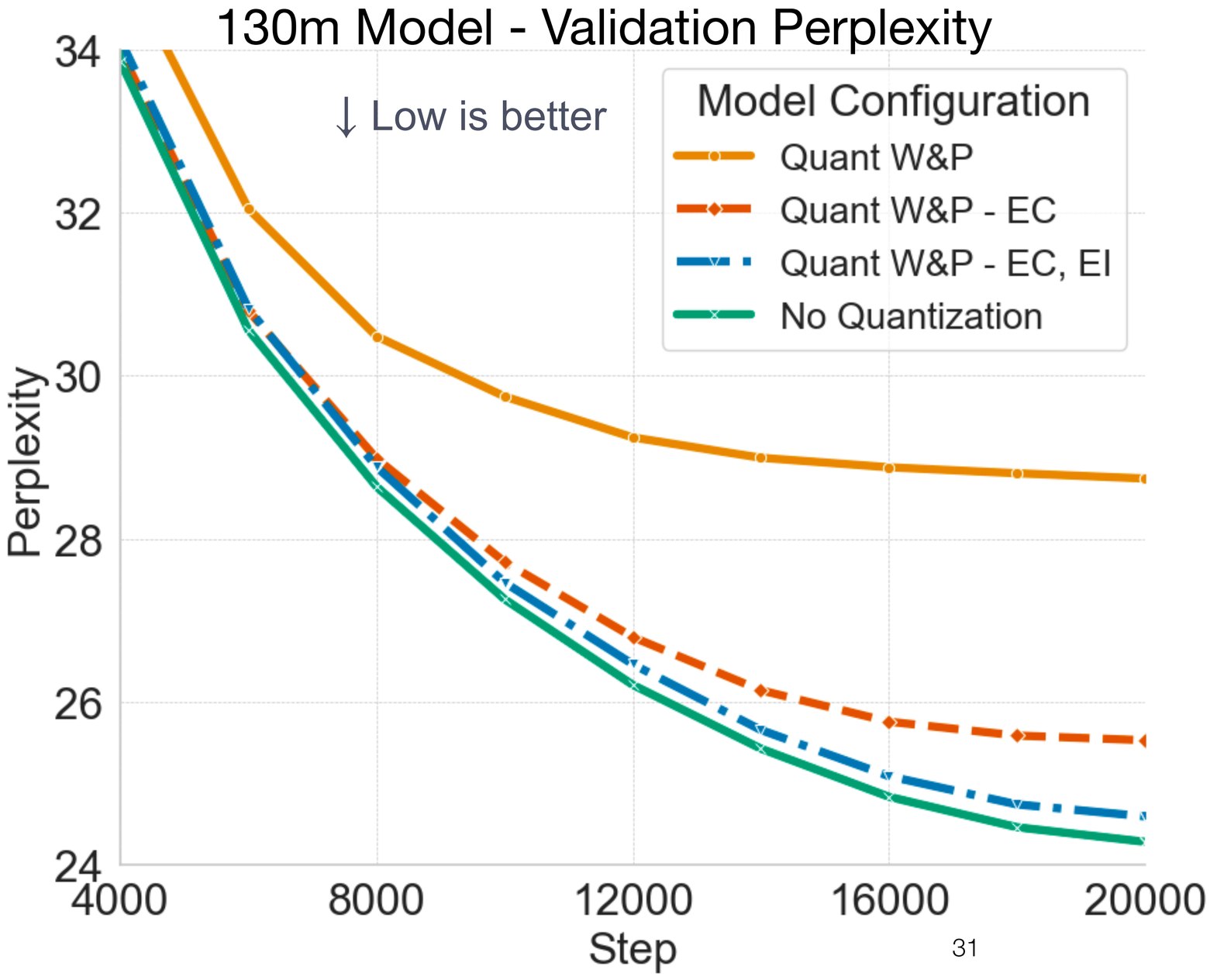
Takeaways

- LoQT is very comparable to Full training and GaLore
- No-quantization (LoQT-nq) is slightly better than LoQT
- LoQT is much better than LoRA and ReLoRA

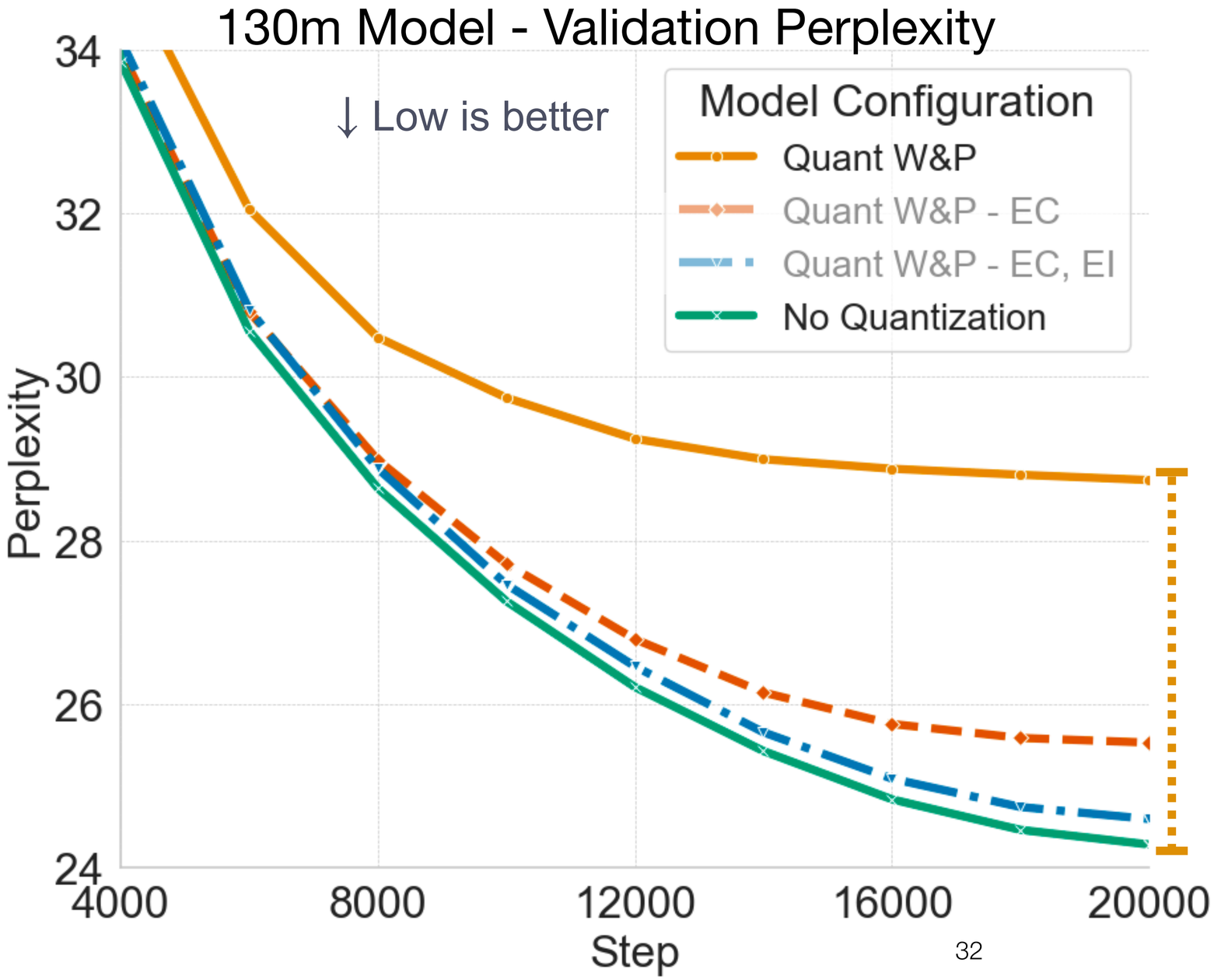


r / d_{model}	128 / 512	256 / 768	256 / 1024	512 / 2048

Component Ablations

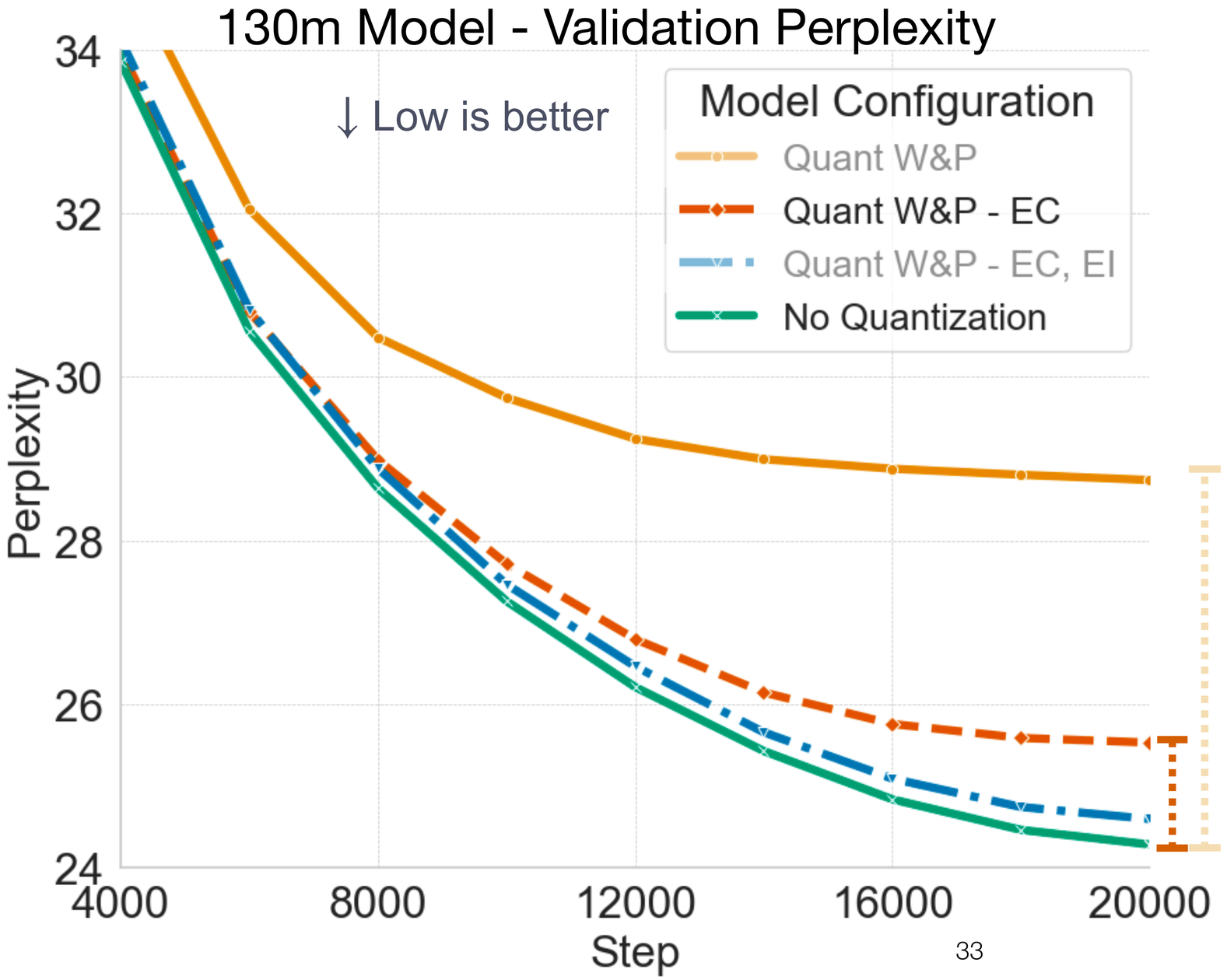


Component Ablations



- **Simple 4-bit Quantization of W and P:** Flattens early.

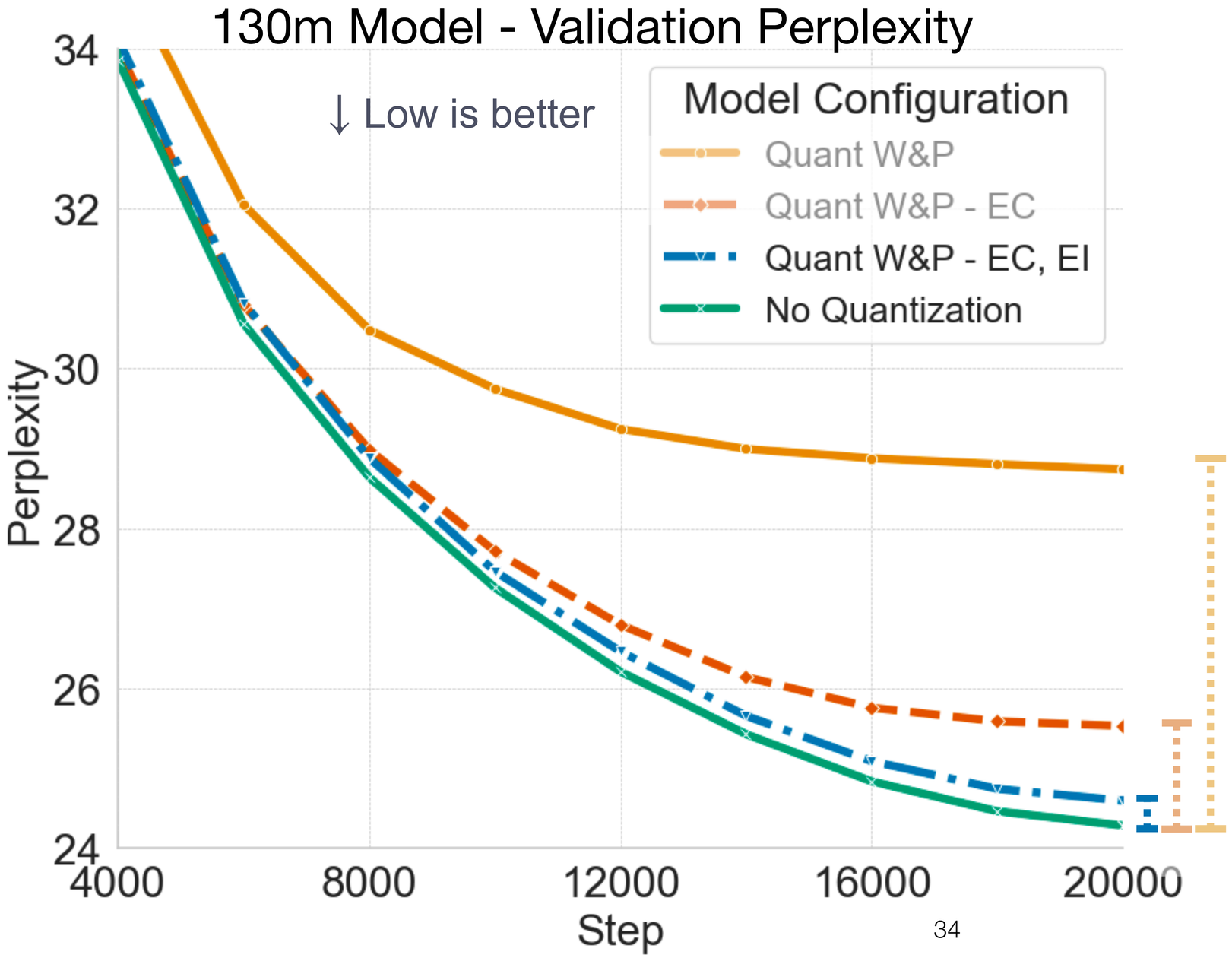
Component Ablations



$$B = P^{-1}(W_q - W)$$

- **Simple 4-bit Quantization of W and P:** Flattens early.
- **Error Compensation (EC):** This gap is reduced by 72%.

Component Ablations

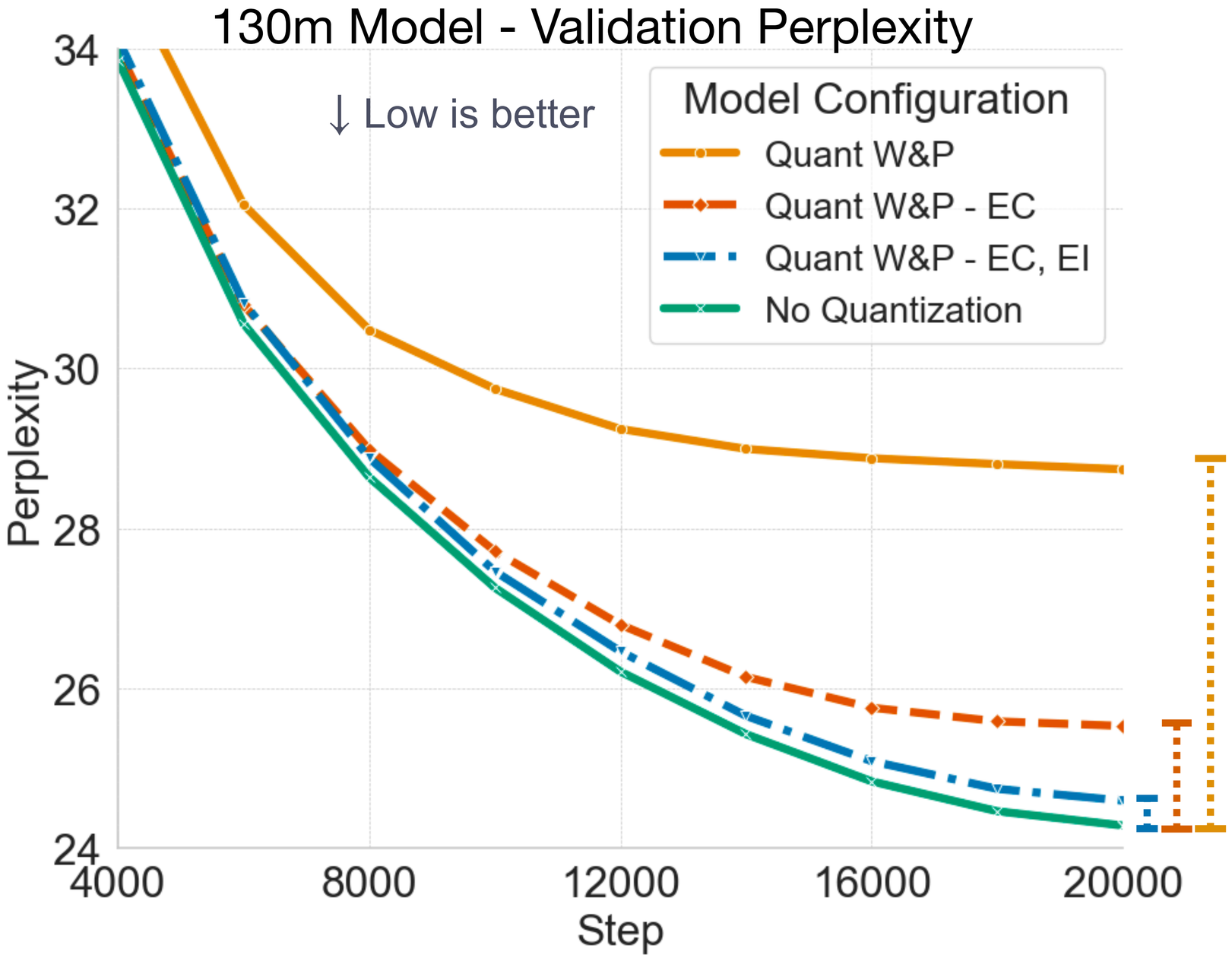


$$B = P^{-1}(W_q - W)$$

$$T_i = \tau + \psi^i$$

- **Simple 4-bit Quantization of W and P:** Flattens early.
- **Error Compensation (EC):** This gap is reduced by 72 %.
- **Exponentially increasing update intervals (EI) and EC**: the gap is reduced by 93 %.

Component Ablations

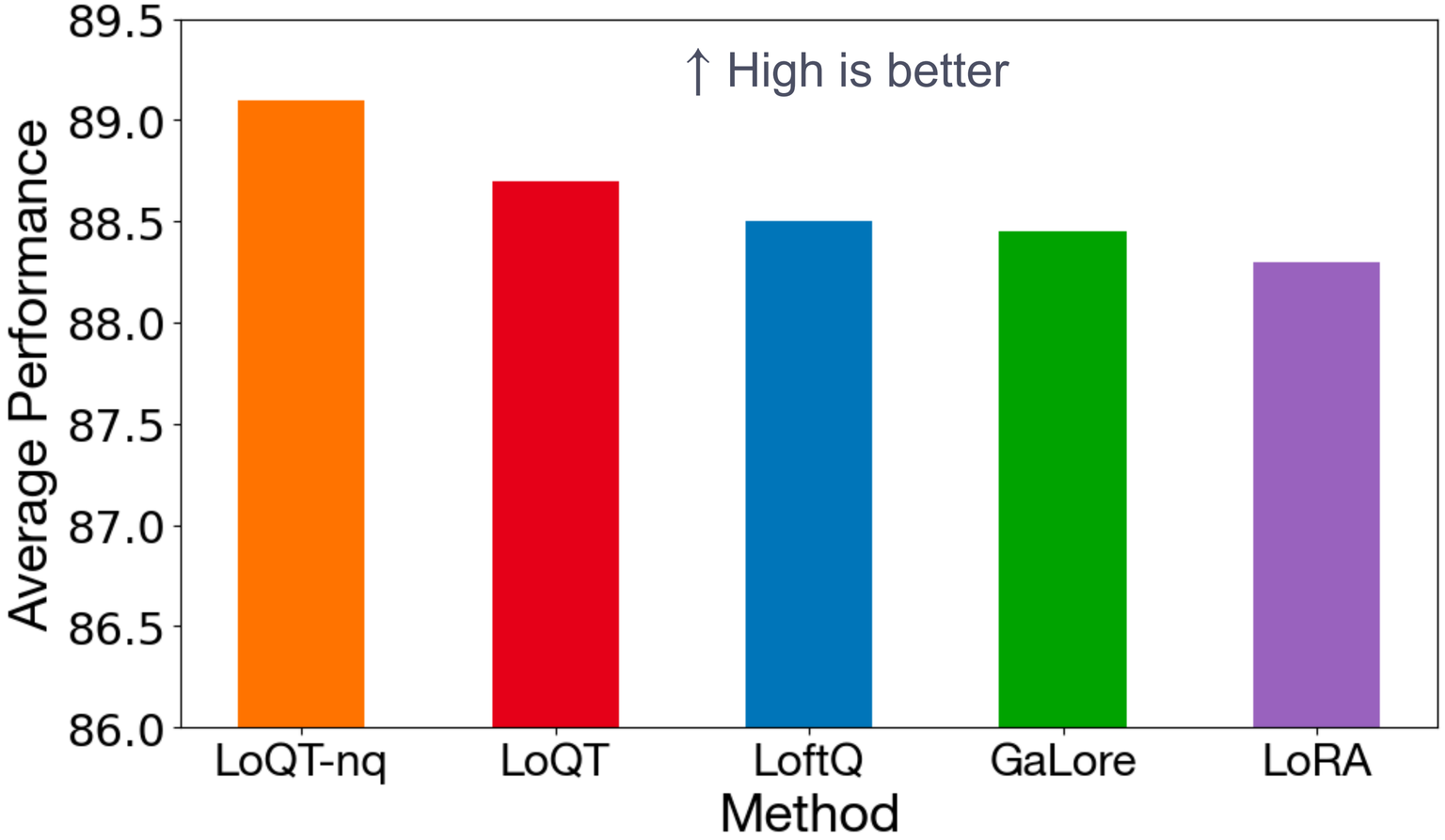


$$B = P^{-1}(W_q - W)$$

$$T_i = \tau + \psi^i$$

- **Simple 4-bit Quantization of W and P:** Flattens early.
- **Error Compensation (EC):** This gap is reduced by 72%.
- **Exponentially increasing update intervals (EI) and EC**: the gap is reduced by 93%.

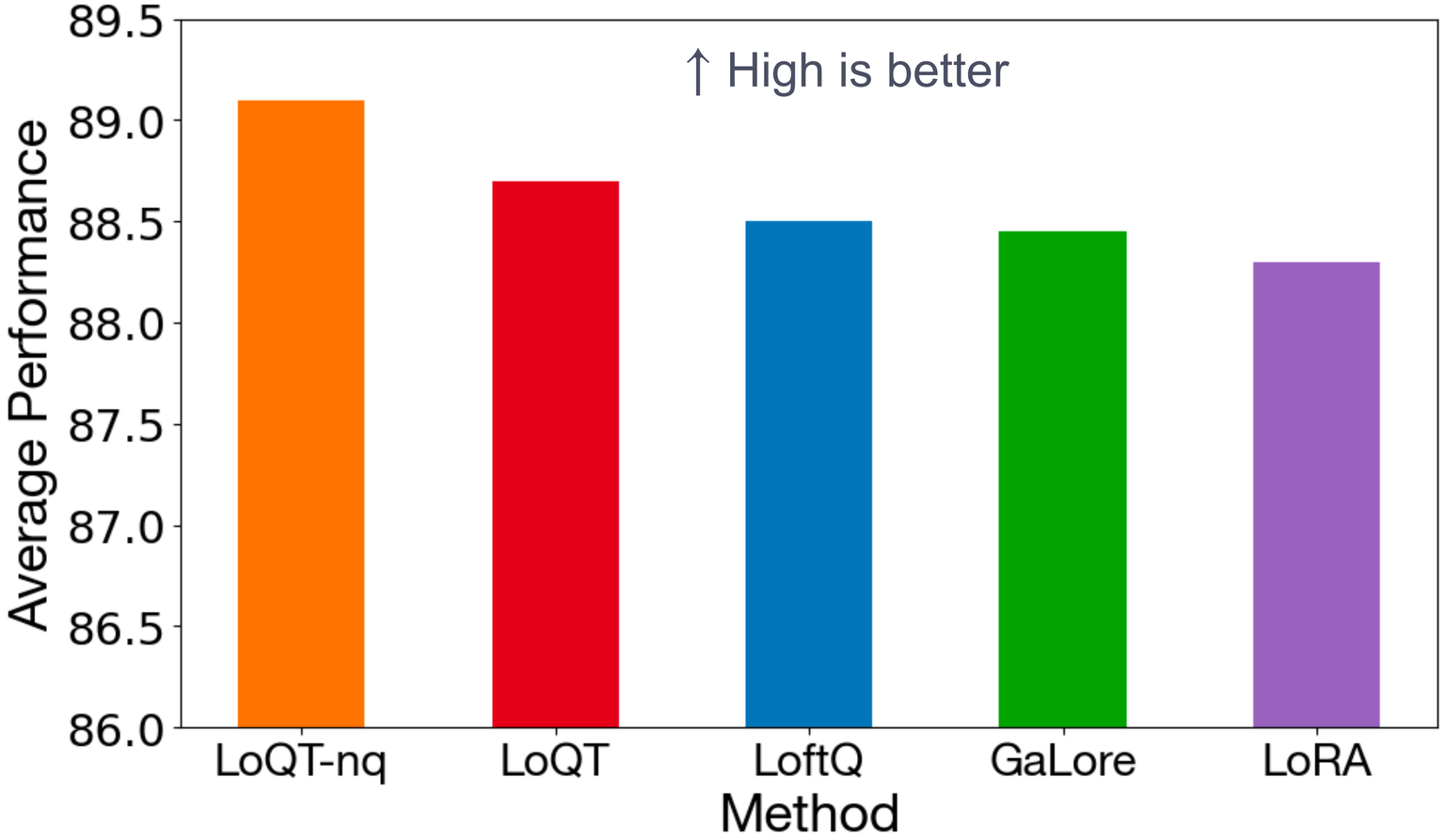
Fine-tuning DeBERTaV3-base Models on GLUE



Fine-tuning DeBERTaV3-base Models on GLUE

Takeaways

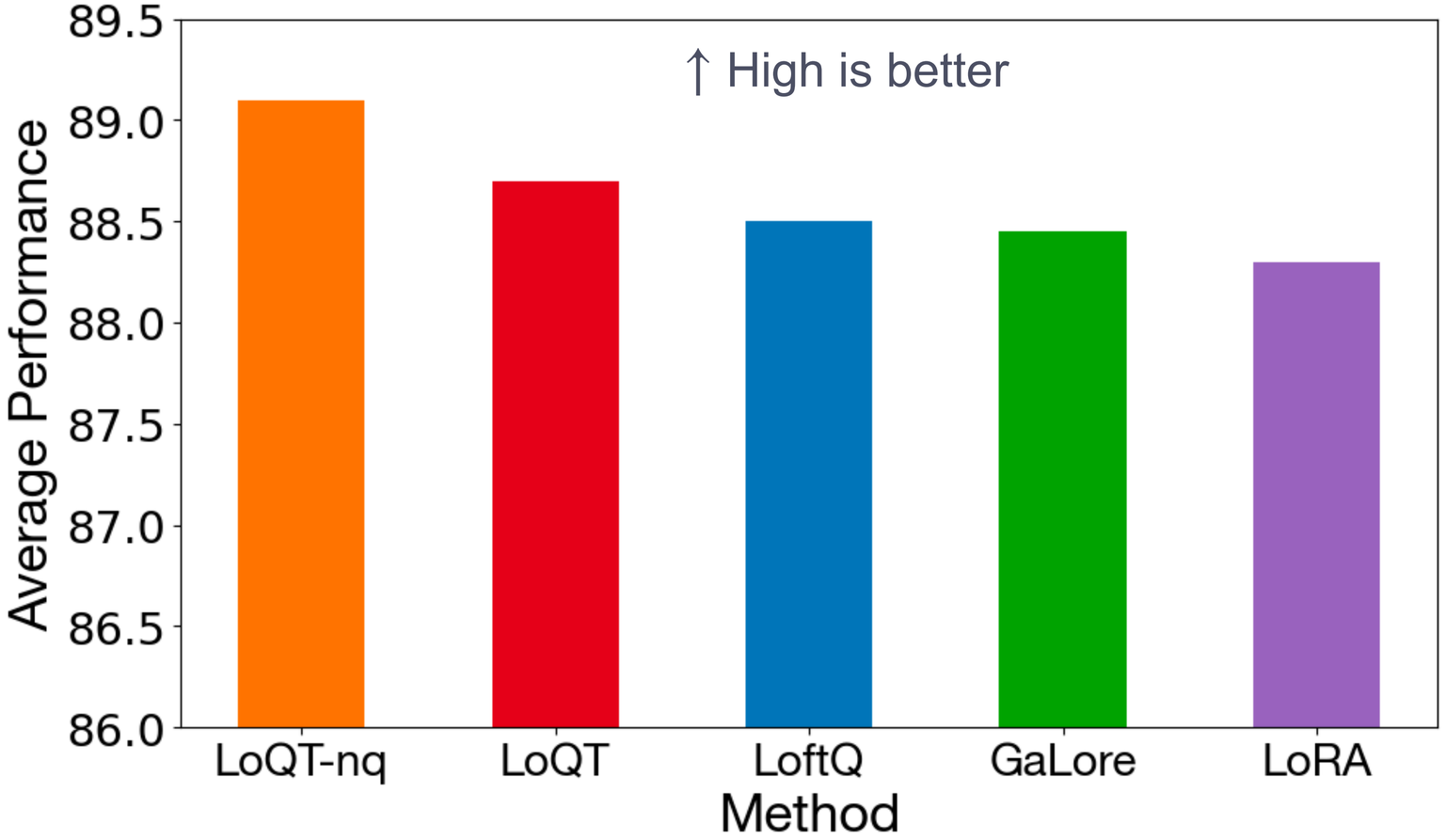
- LoQT and LoQT-*nq* show promising results.



Fine-tuning DeBERTaV3-base Models on GLUE

Takeaways

- LoQT and LoQT-*nq* show promising results.
- Gradient-initialized LoRA factors can benefit fine-tuning.

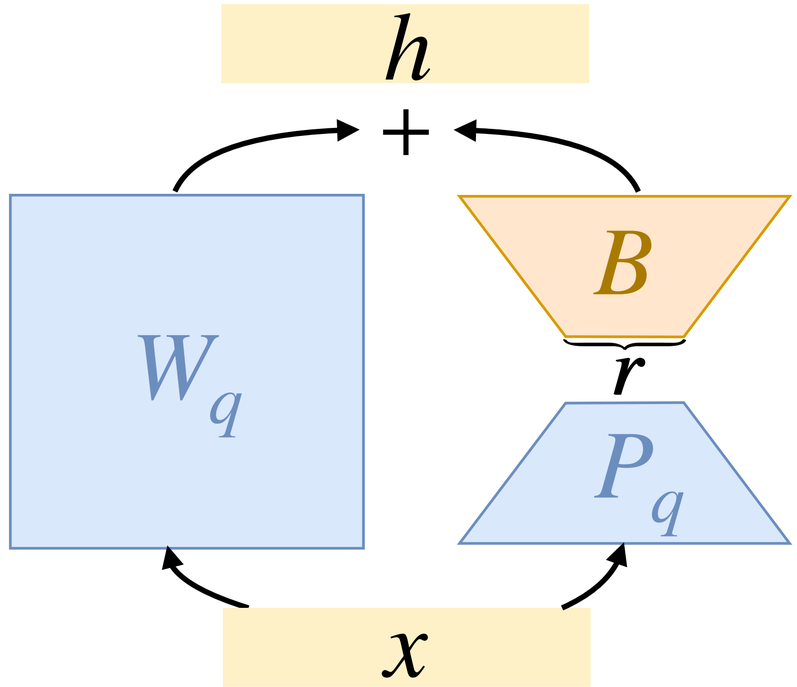


GSM8k

Continued Pretraining Results

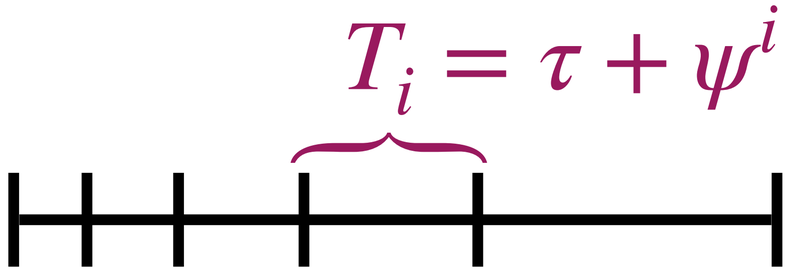
Limitations and Future Work

- **Speed**: LoQT is 16% slower than FP16 training.



Limitations and Future Work

- **Speed**: LoQT is 16% slower than FP16 training.
- **Dynamic Scheduling** for update intervals.



Limitations and Future Work

- **Speed**: LoQT is 16% slower than FP16 training.
- **Dynamic Scheduling** for update intervals.
- **Alternative Architectures and Quantization.**

Key Takeways

- **Pretraining** from scratch with **low-rank updates** and **4-bit quantization** can match FP16 training.

Key Takeways

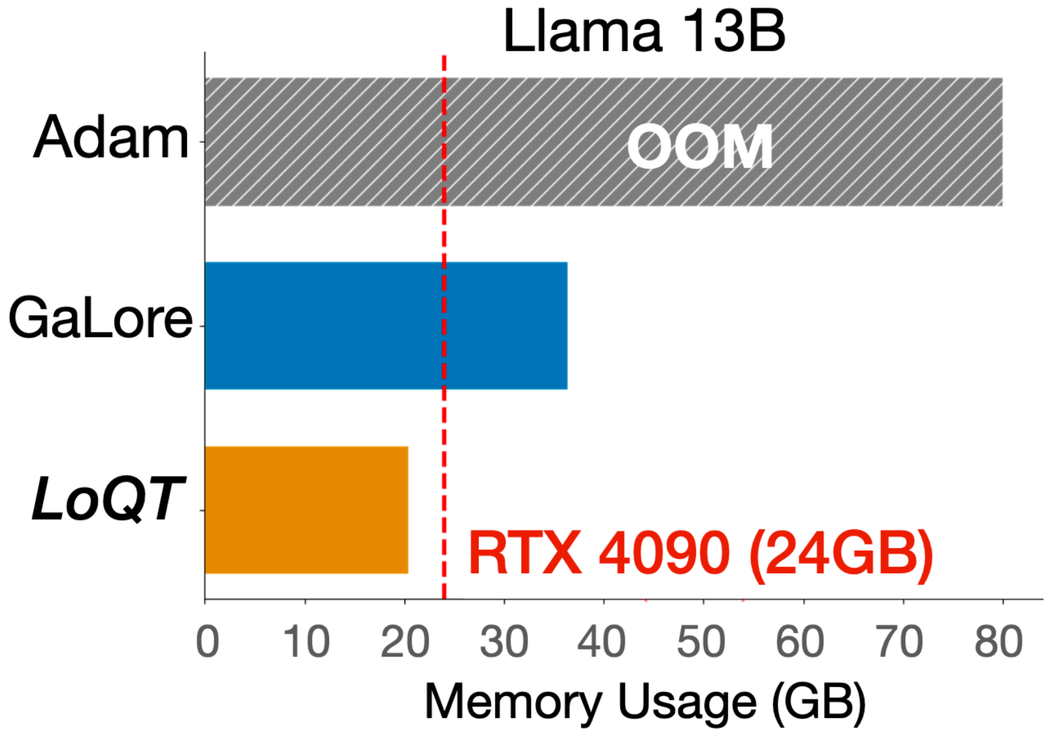
- **Pretraining** from scratch with **low-rank updates** and **4-bit quantization** can match FP16 training.

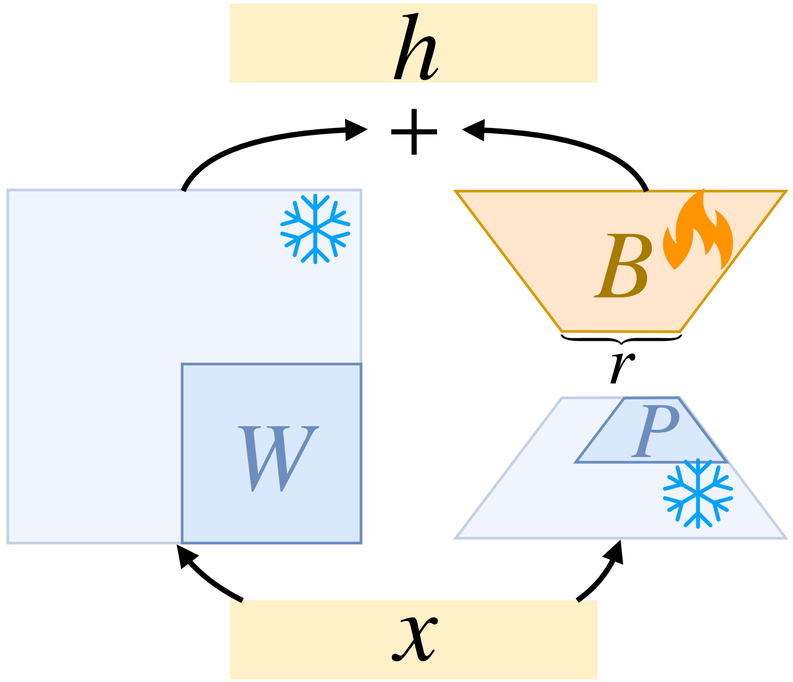
- **Low Memory:** LoQT enables training on consumer hardware.

Key Takeways

- **Pretraining** from scratch with **low-rank updates** and **4-bit quantization** can match FP16 training.
- **Low Memory:** LoQT enables training on consumer hardware.
- **Fine-tuning** may benefit from *gradient-based initialization.*

LoQT: Low Rank Adapters for Quantized Pre-Training





Sebastian Loeschcke*, Mads Toftrup*, Michael J. Kastoryano,
Serge Belongie, and Vésteinn Snæbjarnarson

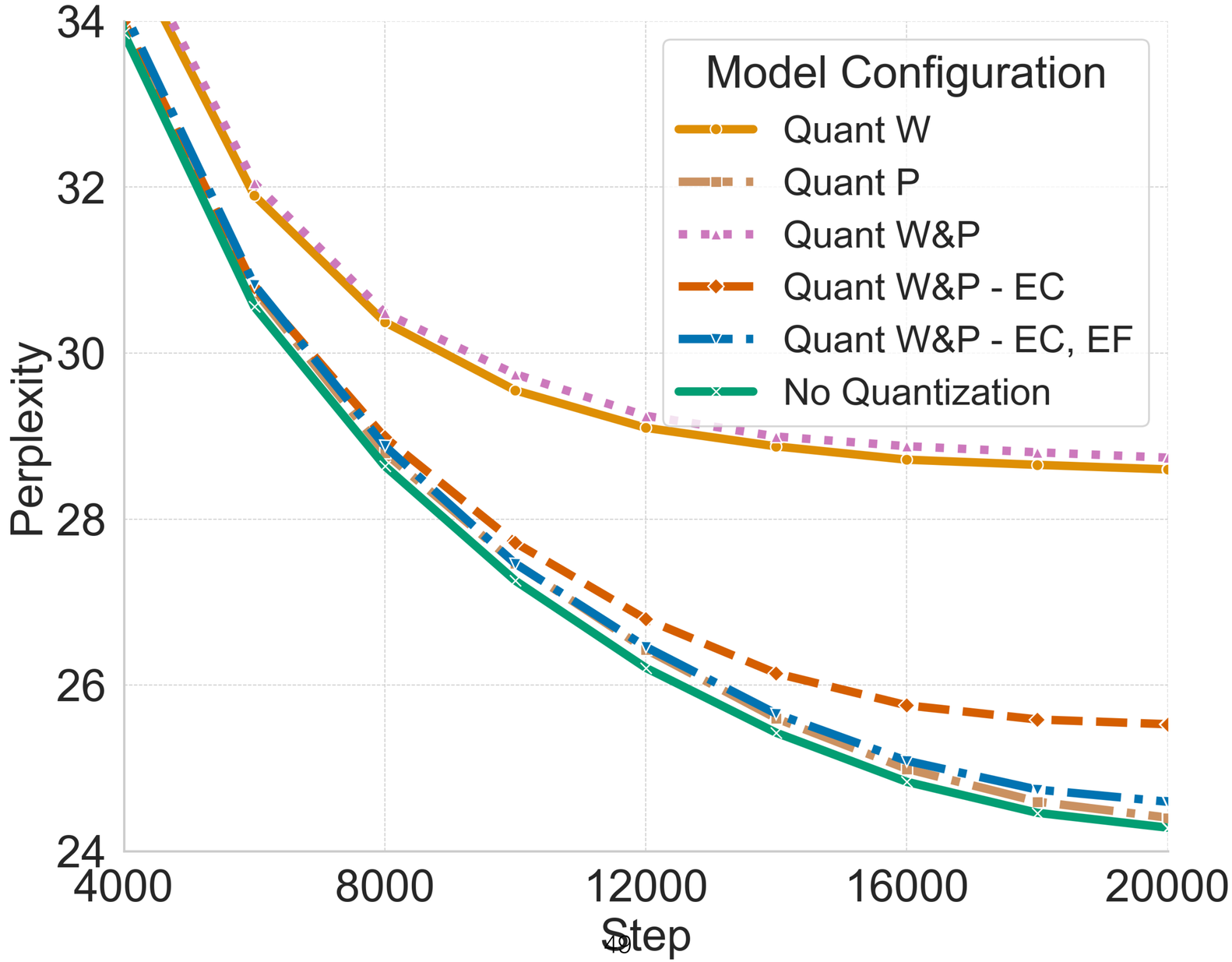
UNIVERSITY OF COPENHAGEN

AARHUS UNIVERSITY

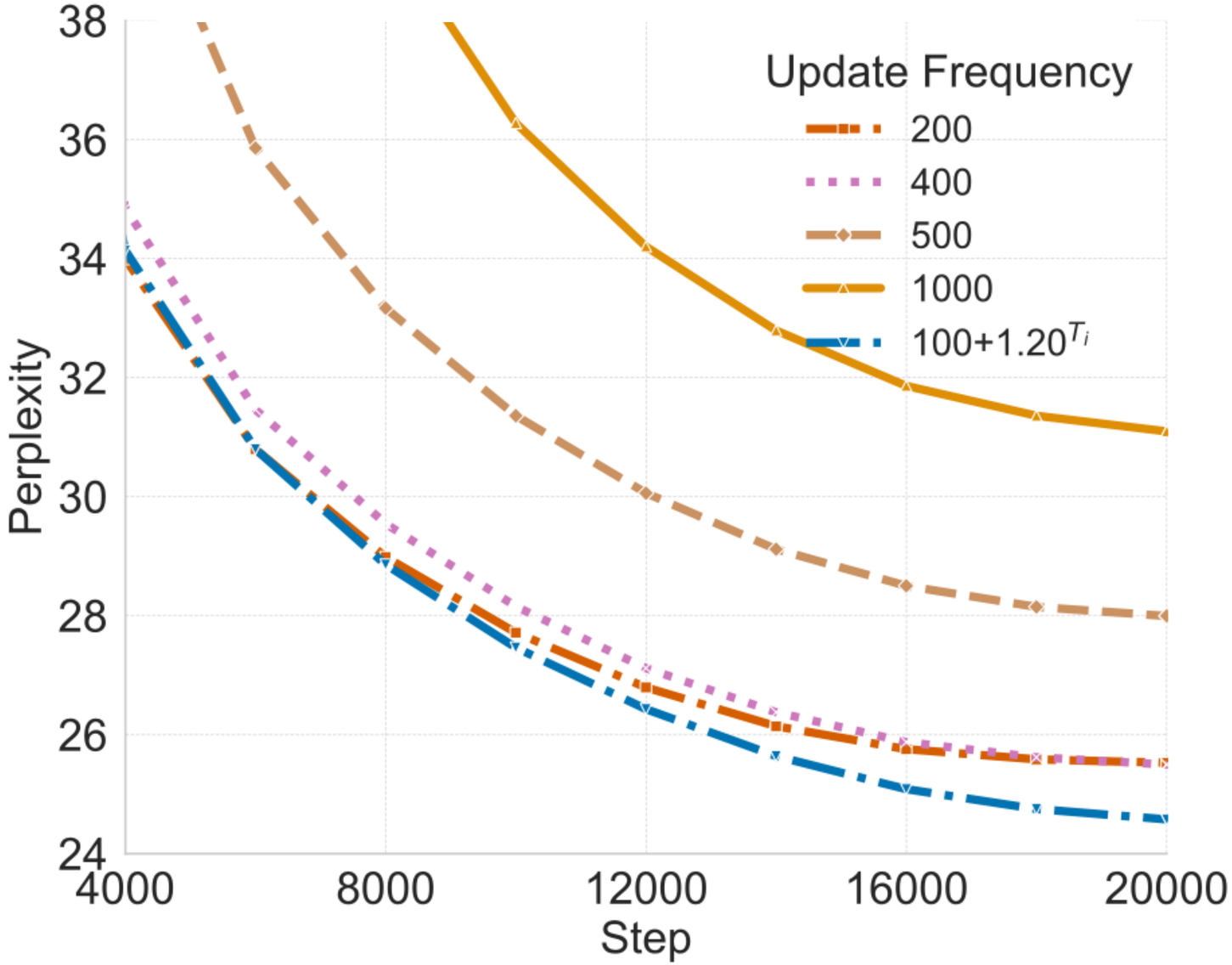
PIONEER CENTRE FOR ARTIFICIAL INTELLIGENCE

Ekstra

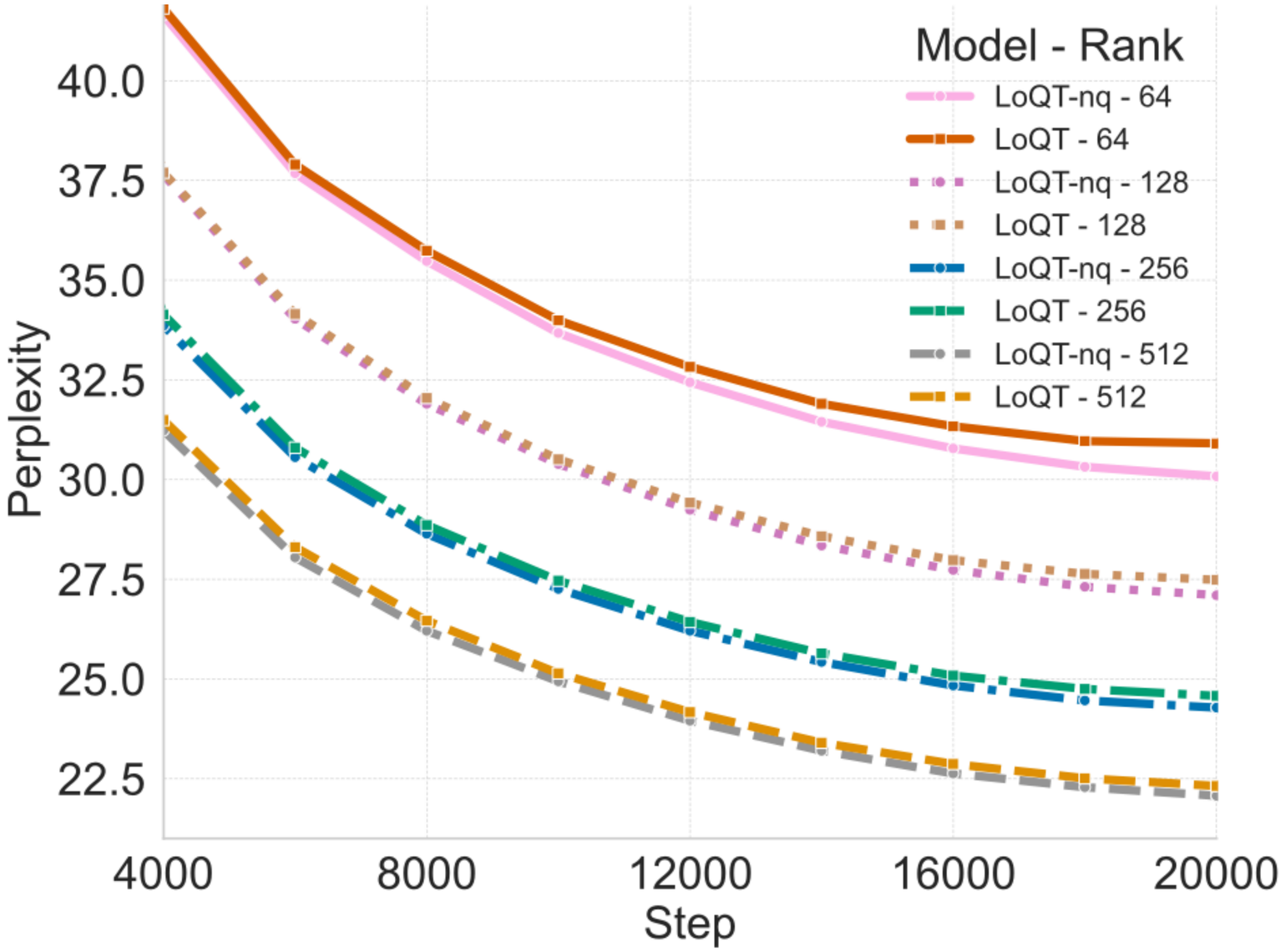
Component Ablations

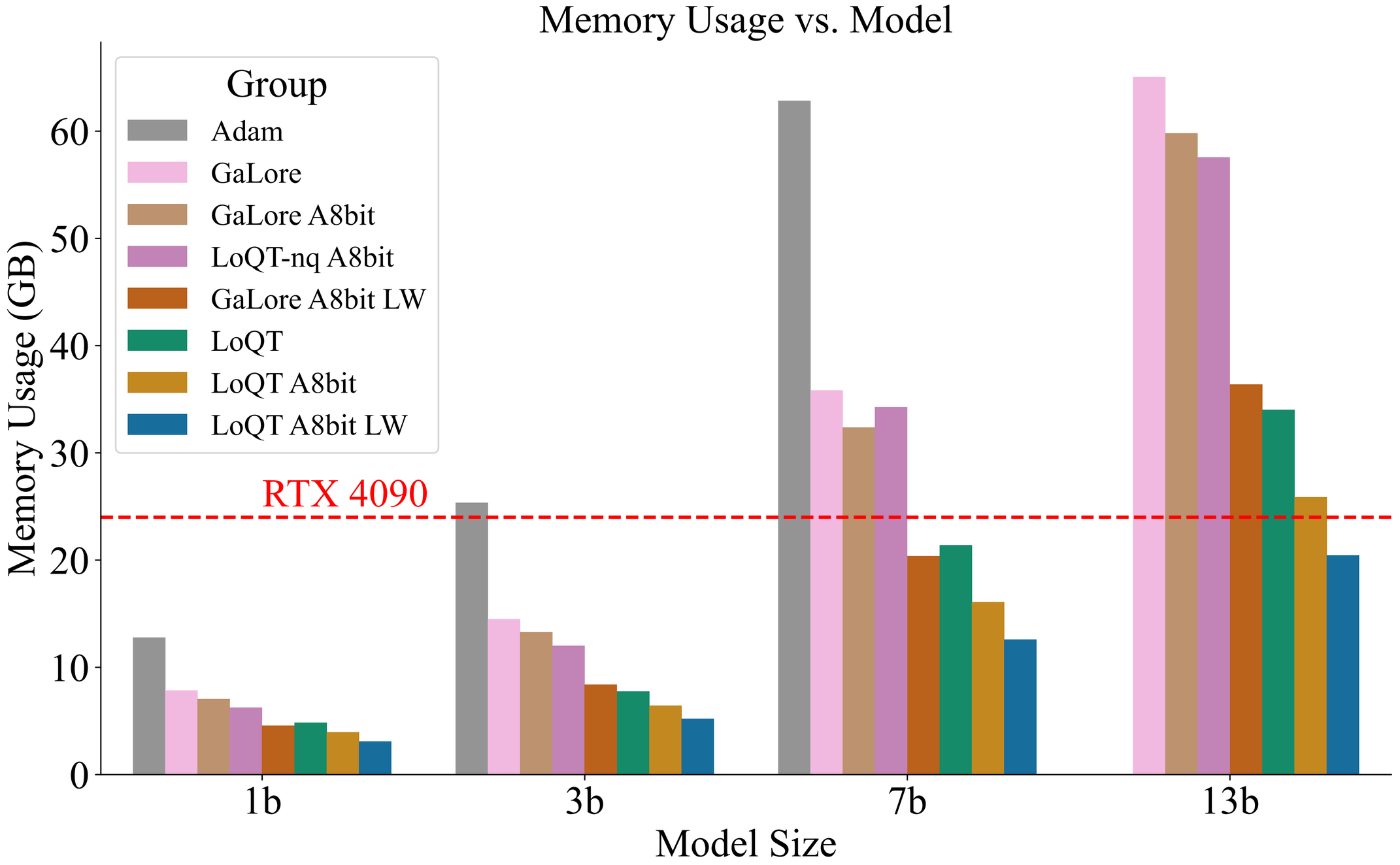


Update Frequency Ablations



Rank Ablations




Memory Usage vs. Model
Group
Adam
GaLore
GaLore A8bit
LoQT-nq A8bit
GaLore A8bit LW
LoQT
LoQT A8bit
LoQT A8bit LW
RTX 4090
Memory Usage (GB)
0
10
20
30
40
50
60
1b
3b
7b
13b
Model Size

Memory requirements by LLMs

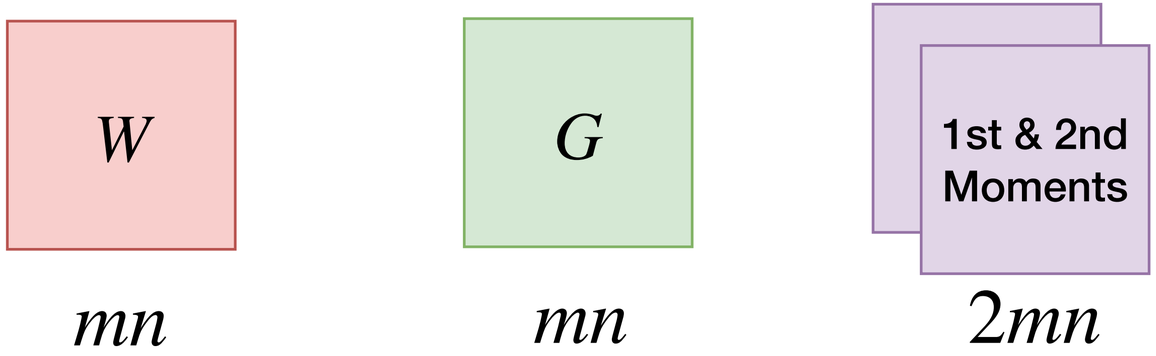
- **Gradient Descent**

Optimizer

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(G_t)$$

Learning Rate

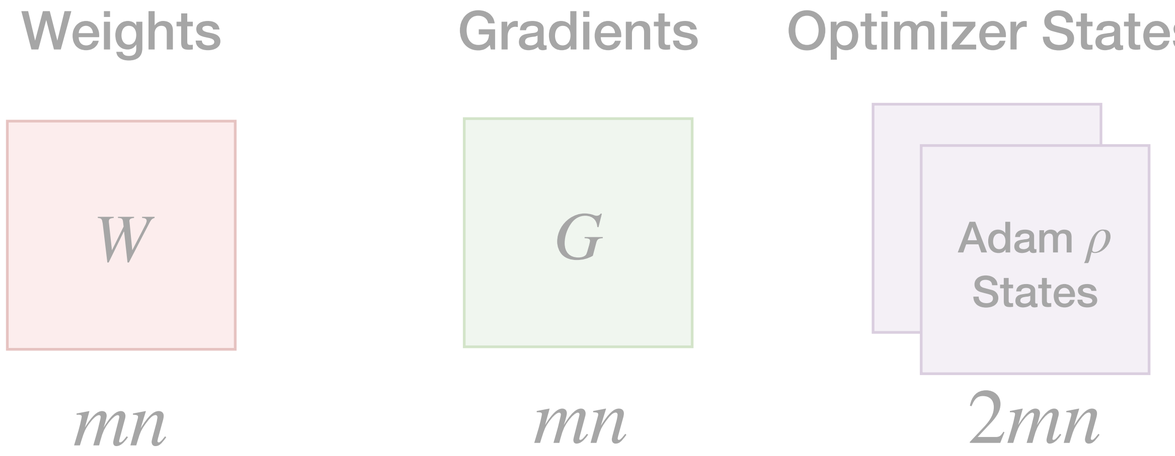
$$G_t = -\nabla_W \varphi_t(W_t) \in \mathbb{R}^{m \times n}$$



LoRA - Low-Rank Adaption

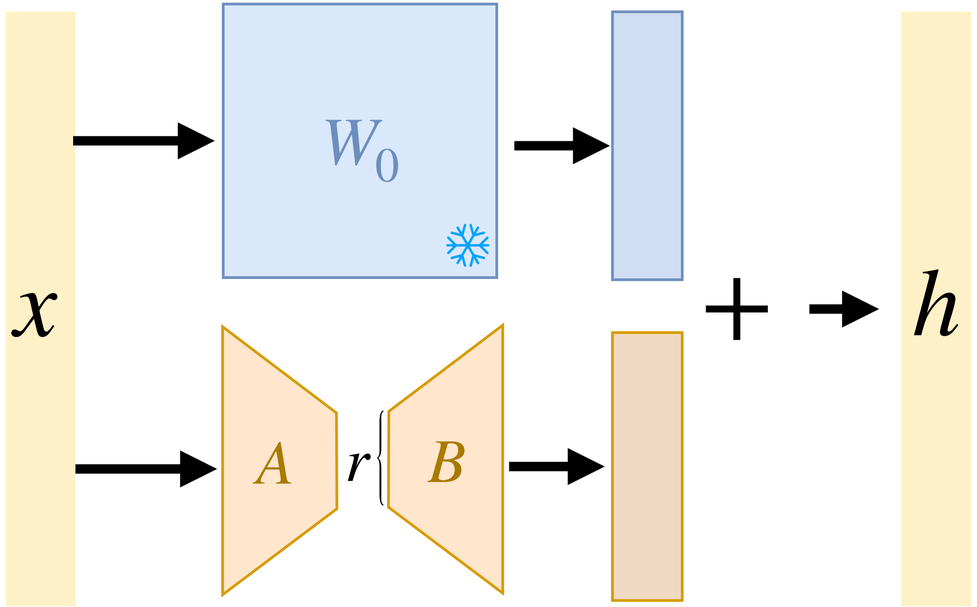
- **Gradient Descent**

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(G_t)$$



- **LoRA** (Hu et. al. 2021)

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(A)\rho(B)$$



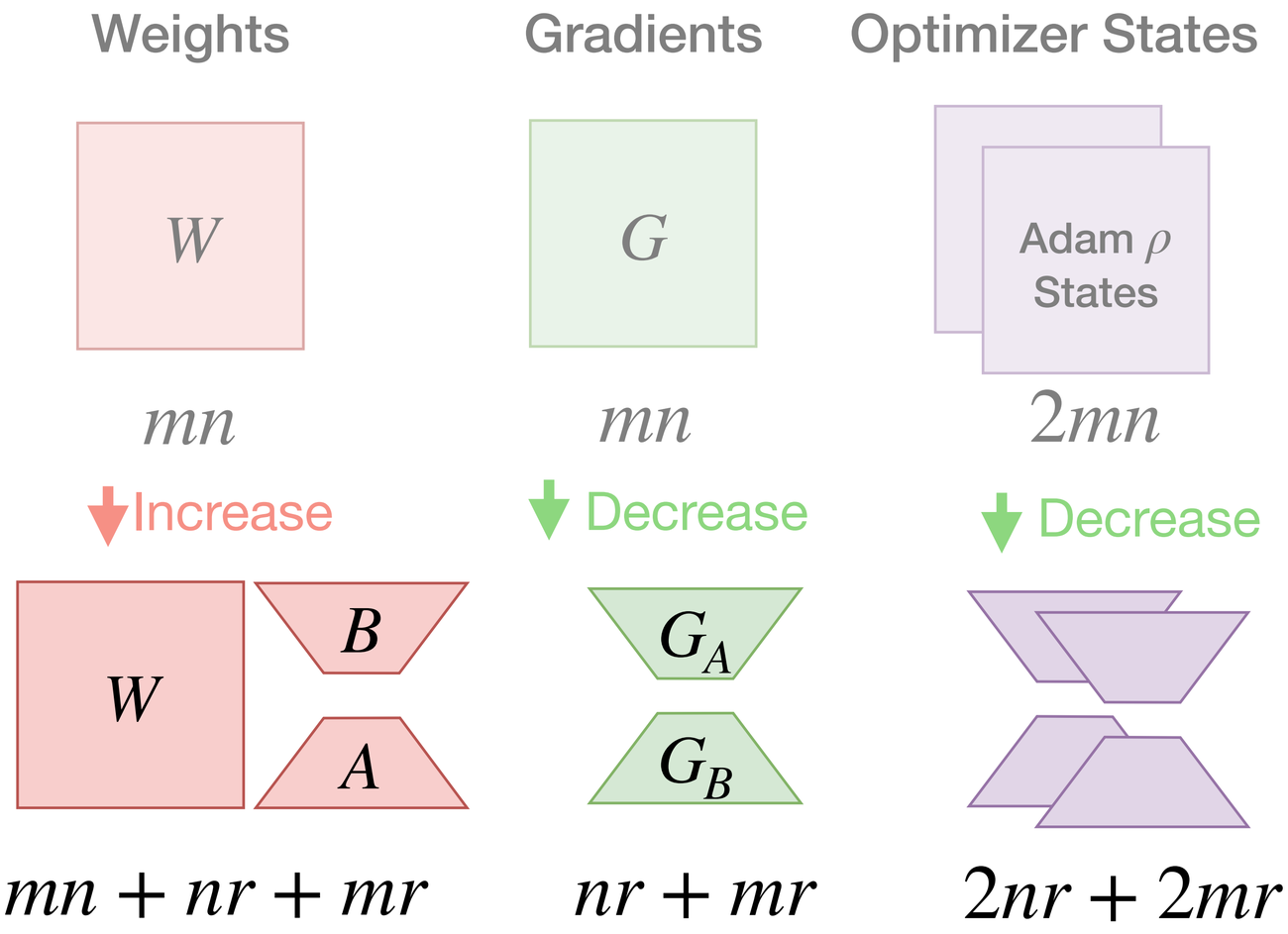
LoRA - Low-Rank Adaption

- **Gradient Descent**

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(G_t)$$

- **LoRA** (Hu et. al. 2021)

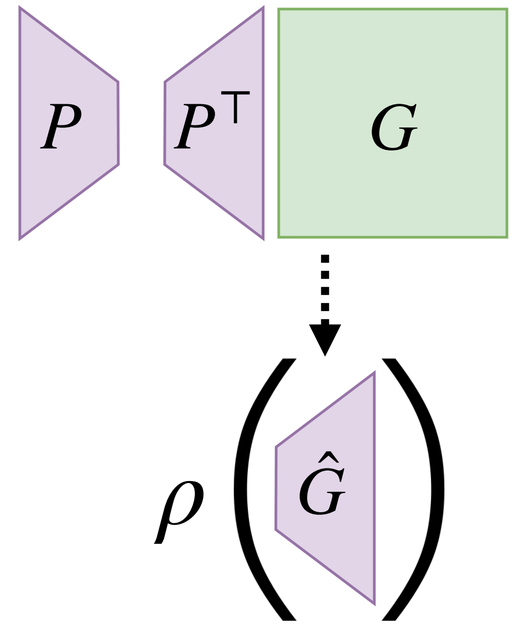
$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(A)\rho(B)$$



Low-Rank Projection of Gradient

- **GaLore** (Zhao et. al., 2024)

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \underbrace{\rho(P_t^\top G_t)}_{\in \mathbb{R}^{r \times n}}$$



Gradient Descent

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(G_t)$$

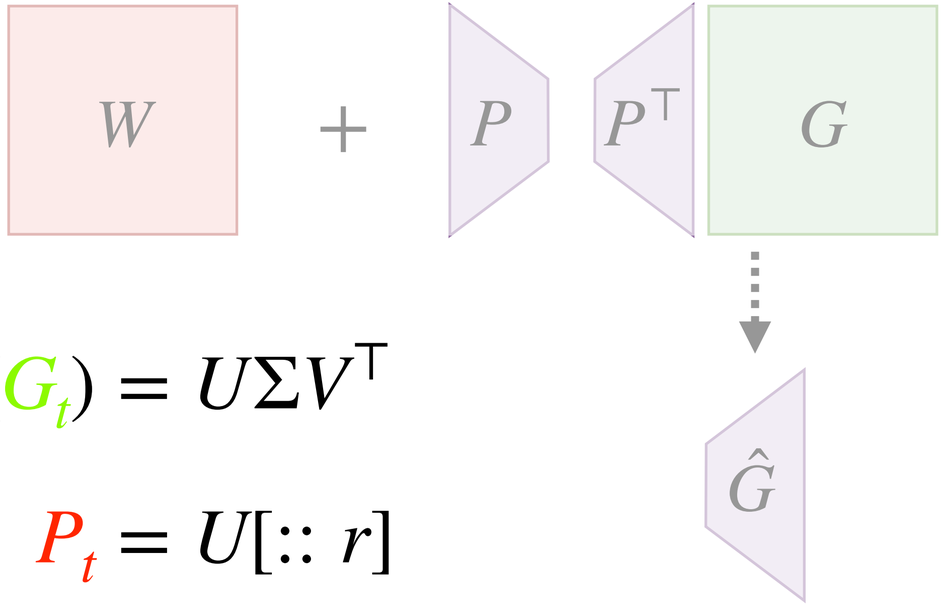
LoRA

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(A)\rho(B)$$

Low-Rank Projection of Gradient

- **GaLore** (Zhao et. al., 2024)

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \rho(\underbrace{P_t^\top G_t}_{\in \mathbb{R}^{r \times m}})$$



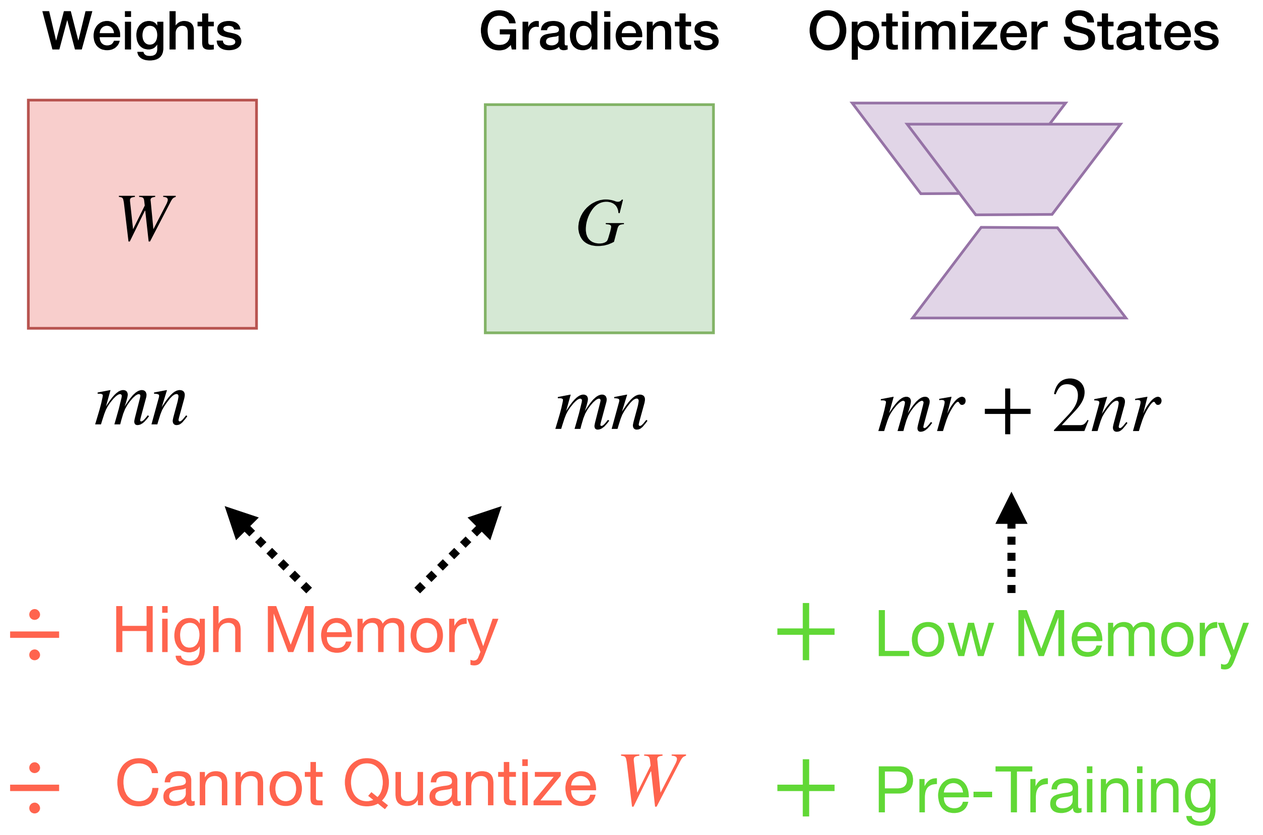
$$SVD(G_t) = U\Sigma V^\top$$

$$P_t = U[:: r]$$

Low-Rank Projection of Gradient

- **GaLore** (Zhao et. al., 2024)

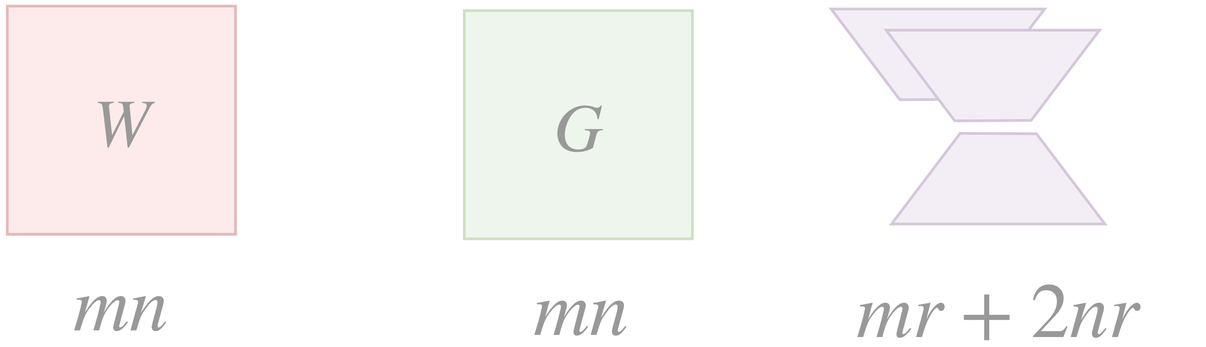
$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \underbrace{\rho(P_t^\top G_t)}_{\in \mathbb{R}^{r \times n}}$$



LoQT - Low Rank Adapters for Quantized Pre-Training

- **GaLore** (Zhao et. al., 2024)

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \underbrace{\rho(P_t^\top G_t)}_{\in \mathbb{R}^{r \times n}}$$



- **LoQT** LoRA factors: A B

LoRA: $W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(A)\rho(B)$

LoQT - Low Rank Adapters for Quantized Pre-Training

- **GaLore** (Zhao et. al., 2024)

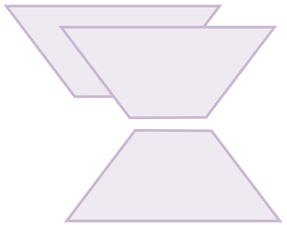
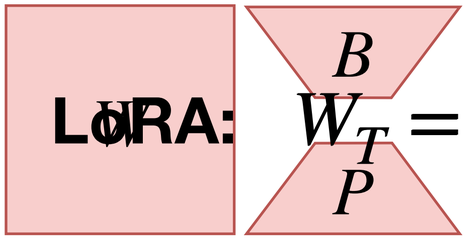
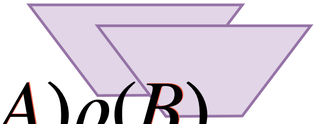
$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \underbrace{\rho(P_t^\top G_t)}_{\in \mathbb{R}^{r \times n}}$$

- **LoQT**

LoRA factors: A B

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} P_t \, \rho(B_t)$$

❄ Frozen 🔥 Optimize

	Weights	Gradients	Optimizer States
			
	W	G	
	mn	mn	$mr + 2nr$
	Increase	Decrease	Decrease
LoRA:			
	$W_T = W_0 + \eta \sum_{t=0}^{T-1} \rho(A)\rho(B)$		
	$mn + nr + mr$	nr	$2nr$

LoQT and GaLore have equivalent Gradient Updates

- **Forward pass for LoQT**: $y = xW + xPB$

- The gradient of the loss w.r.t. B using the Chain Rule:

$$\frac{\partial B}{\partial \mathcal{L}} = (xP)^\top \frac{\partial \mathcal{L}}{\partial y} = P^\top \underbrace{x^\top \frac{\partial \mathcal{L}}{\partial y}}_{G^W} = P^\top G^W$$

GaLore $$W_T = W_0 + \eta \sum_{t=0}^{T-1} P_t \rho(\overbrace{P_t^\top G_t}^{})$$

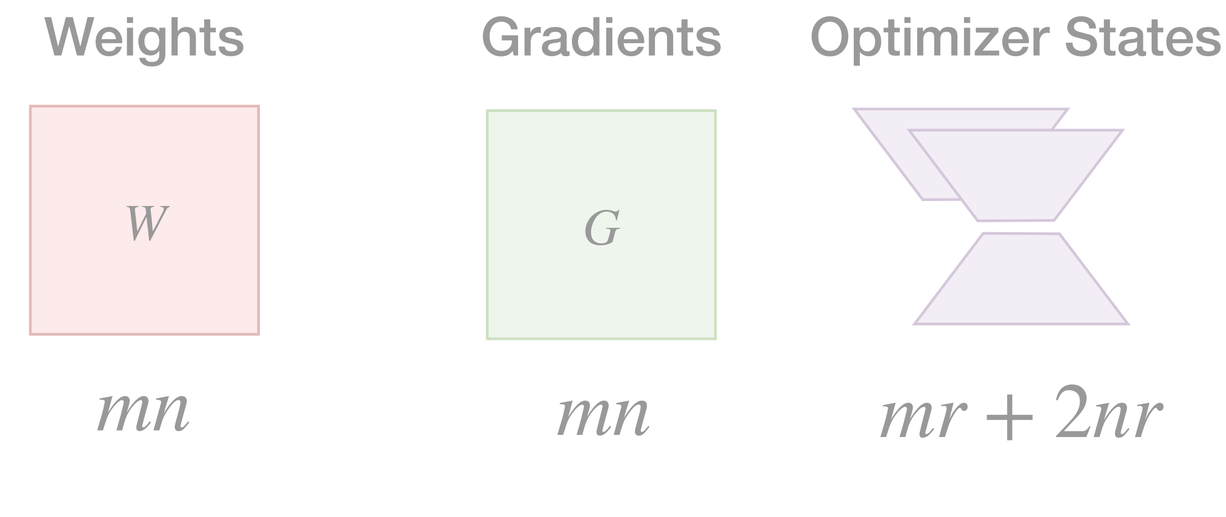
LoQT - Low Rank Adapters for Quantized Pre-Training

- **GaLore** (Zhao et. al., 2024)

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \underbrace{P_t}_{\in \mathbb{R}^{m \times r}} \rho(\underbrace{P_t^\top G_t}_{\in \mathbb{R}^{r \times n}})$$

- **LoQT**

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} P_t \rho(B_t)$$



Note: The gradient of the loss w.r.t. B_t: $\frac{\partial \mathcal{L}}{\partial B_t} = P_t^\top G_t$ → Same Gradient Updates