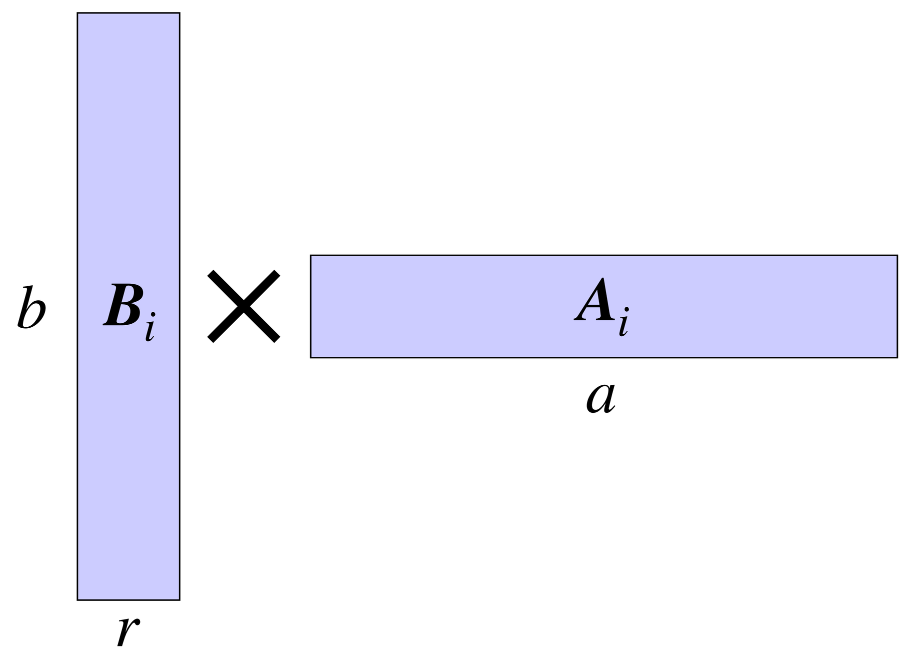
# RaSA: Rank-Sharing Low-Rank Adaptation

Zhiwei He Zhaopeng Tu Xing Wang Xingyu Chen Zhijie Wang Jiahao Xu Tian Liang
Wenxiang Jiao Zhuosheng Zhang Rui Wang

## Background & Motivation



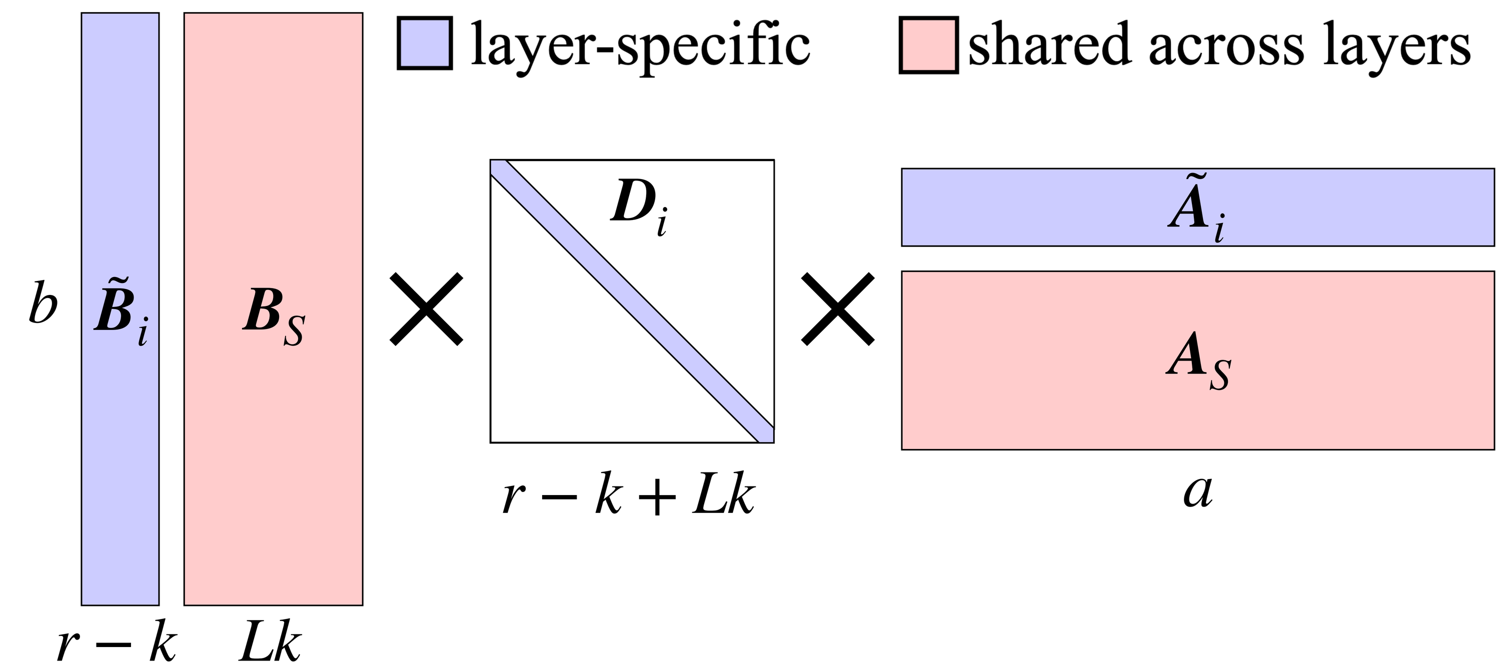

$$W + \Delta W = W + \frac{\alpha}{r} BA \quad (B \in \mathbb{R}^{b\times r}, A \in \mathbb{R}^{r\times a})$$

LoRA

- ✔ LoRA still lags behind full fine-tuning (FFT), particularly in scenarios involving large training datasets and complex tasks such as mathematical reasoning and code generation. A plausible explanation for this performance gap is that the low-rank constraint limits the expressive capacity of LoRA [1-2].
- ✔ Recent studies still indicate redundancy in LoRA's parameters, reducing LoRA by 1000 times without performance loss [3-6].
- ! This contradiction suggests that LoRA's parameters are still not being fully utilized.

[1] Mora: High-rank updating for parameter-efficient fine-tuning
[2] Lora learns less and forgets less
[3] Vera: Vector-based random matrix adaptation
[4] Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying
[5] VB-LoRA: extreme parameter efficient fine-tuning with vector banks

## Method




$$W_i + \Delta W_i = W_i + \underbrace{\begin{bmatrix} \tilde{B}_i & B_S \end{bmatrix}}_{\mathbb{R}^{b\times(r-k+Lk)}} D_i \underbrace{\begin{bmatrix} \tilde{A}_i \\ A_S \end{bmatrix}}_{\mathbb{R}^{(r-k+Lk)\times a}}$$

RaSA

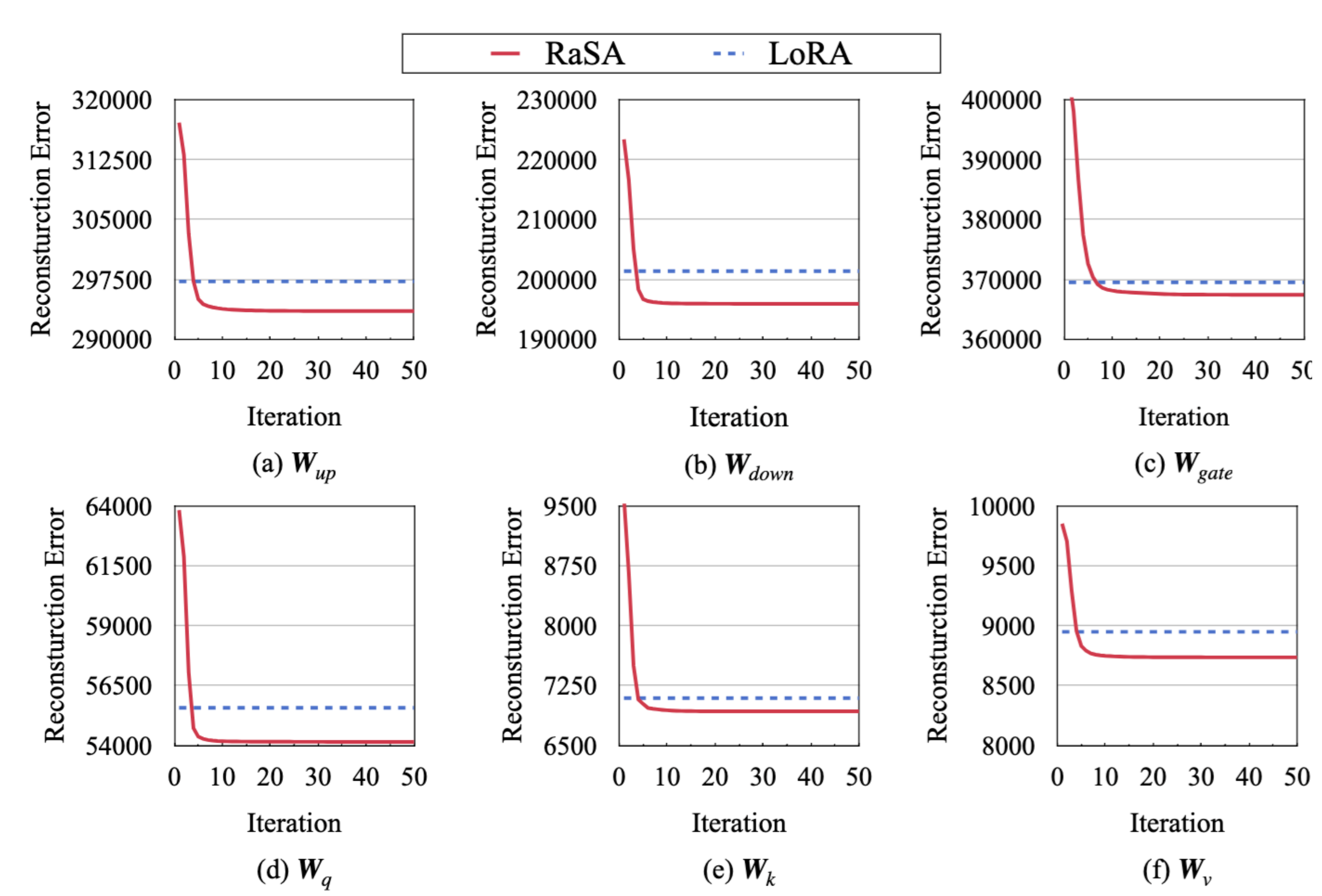
- RaSA extracts k ranks from each layer's LoRA update to form a rank pool of L × k ranks, which is shared across all layers with layer-specific weighting.

## Reconstruction Error

$$e_{\text{lora}} = \min_{B_i, A_i} \sum_{i=1}^{L} \| M_i - B_i A_i \|_F^2$$

$$e_{\text{rasa}(k)} = \min_{\tilde{B}_i, \tilde{A}_i, B_S, A_S, D_i} \sum_{i=1}^{L} \left\| M_i - \begin{bmatrix} \tilde{B}_i & B_S \end{bmatrix} D_i \begin{bmatrix} \tilde{A}_i \\ A_S \end{bmatrix} \right\|_F^2$$

- We prove $e_{\text{rasa}(k)} \le e_{\text{lora}}$ .




## Main Results


## RaSA Learns More Than LoRA