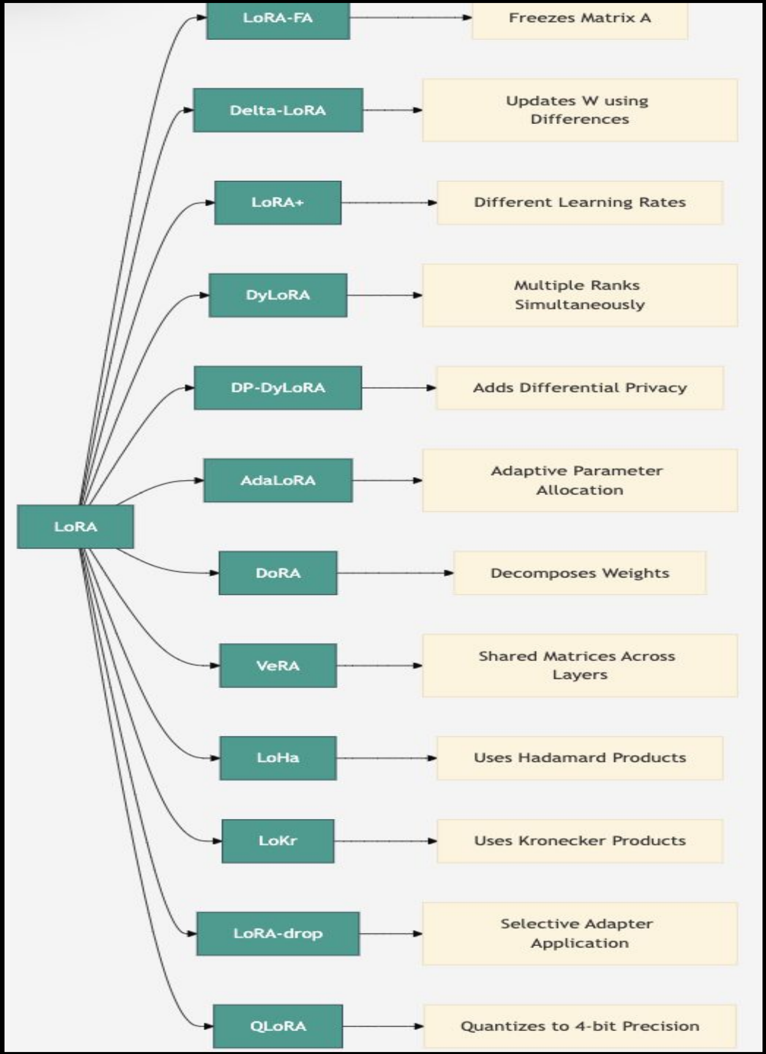
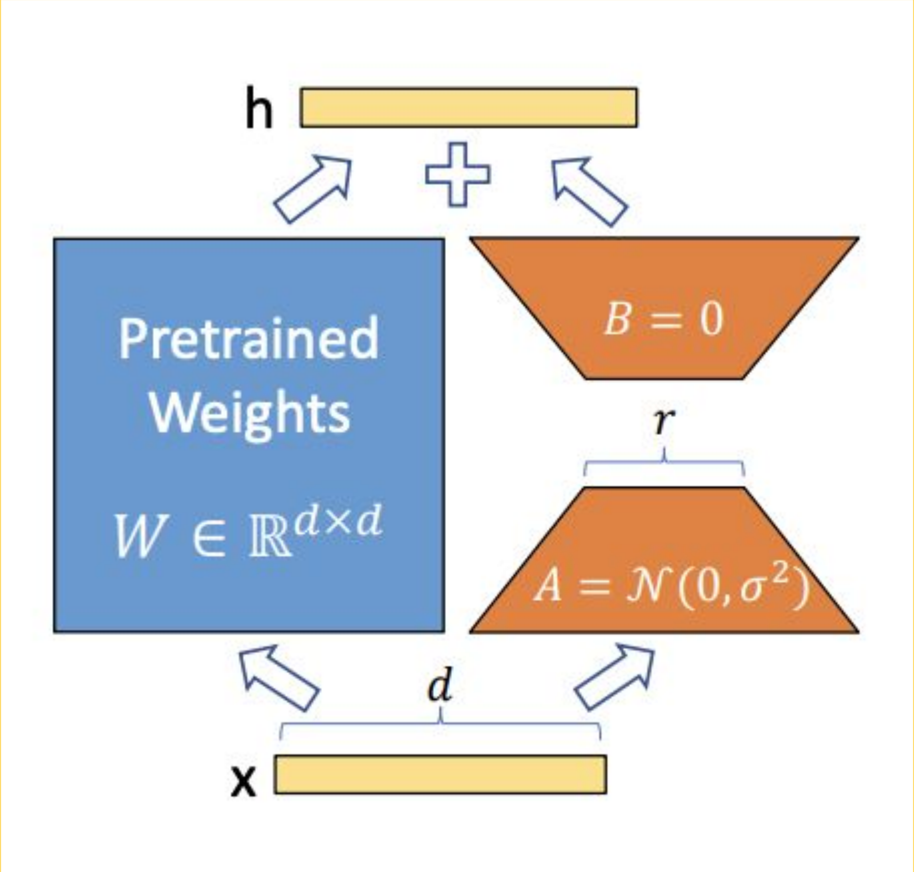
LoRA variants for efficient

LLM

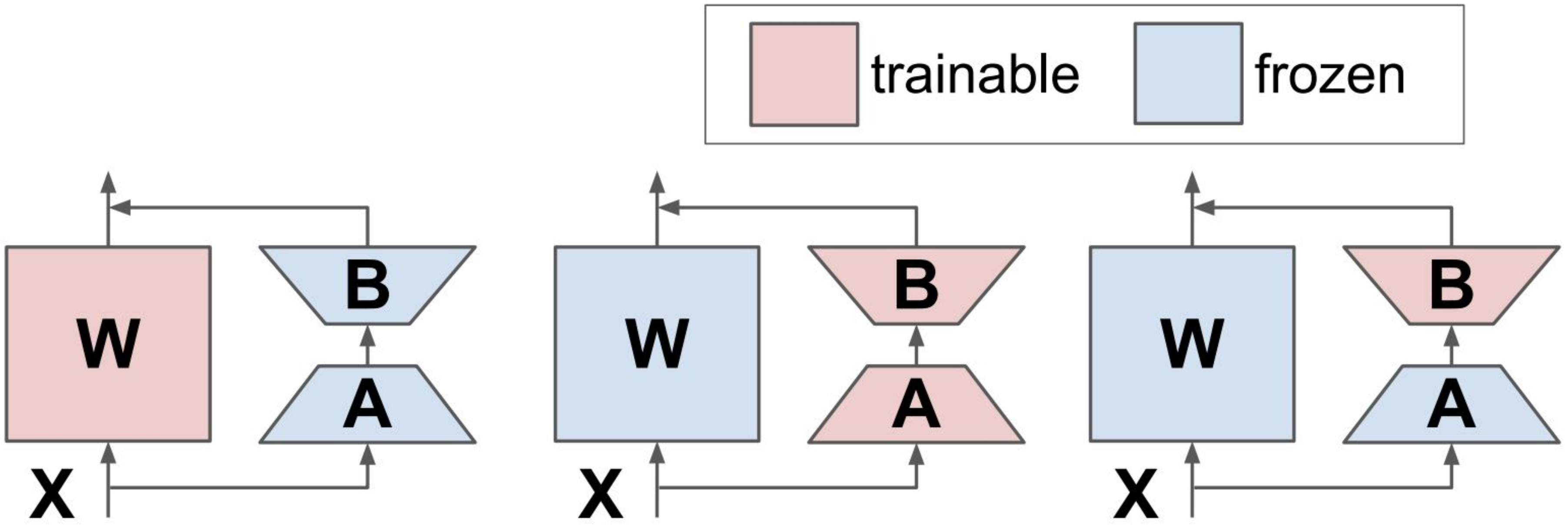
Fine-Tuning



1. LoRA (Low-Rank Adaptation)



2. LoRA-FA (Frozen-A)



(a) Full Parameter FT (b) LoRA (c) LoRA-FA

3. Delta-LoRA

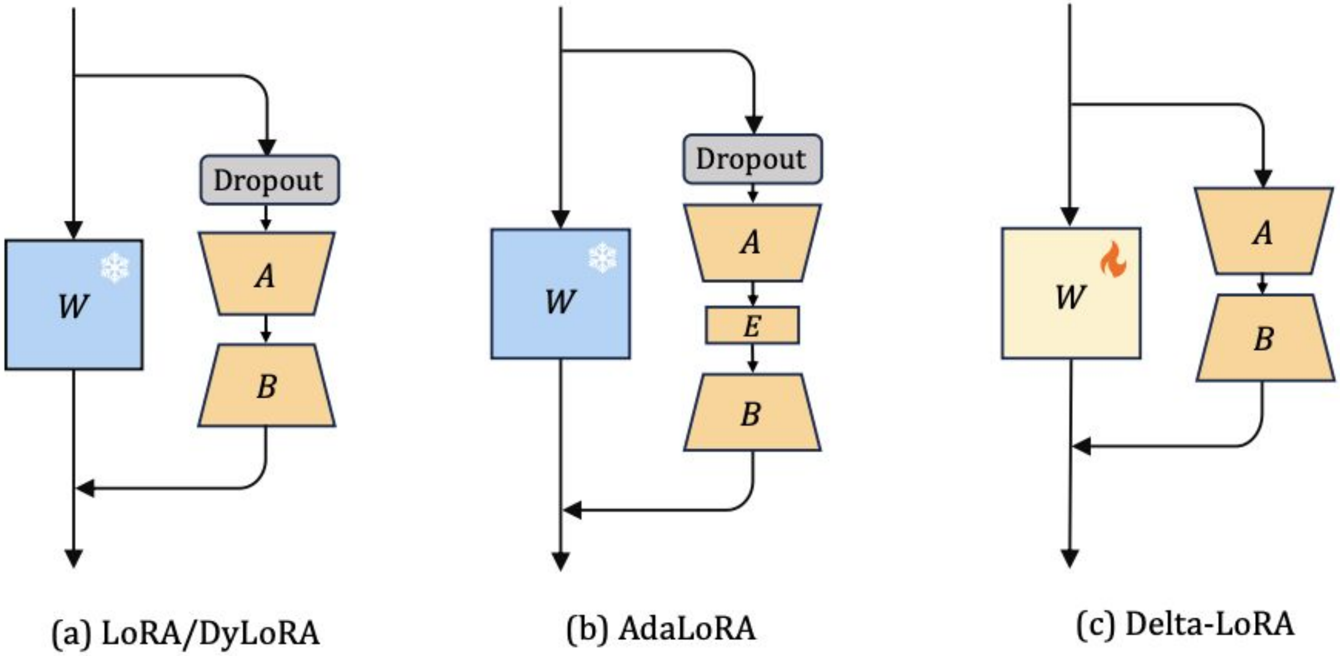
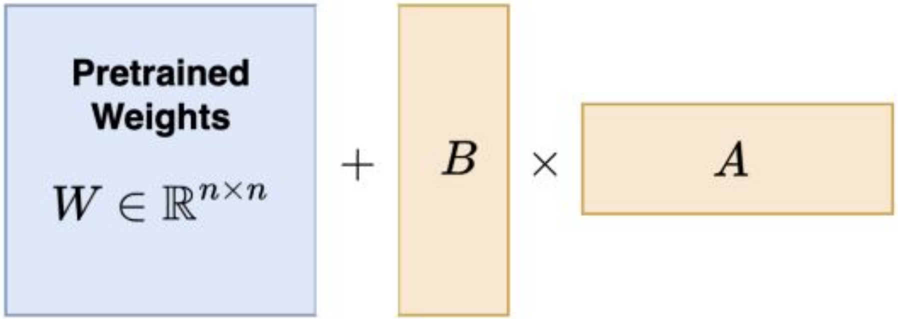
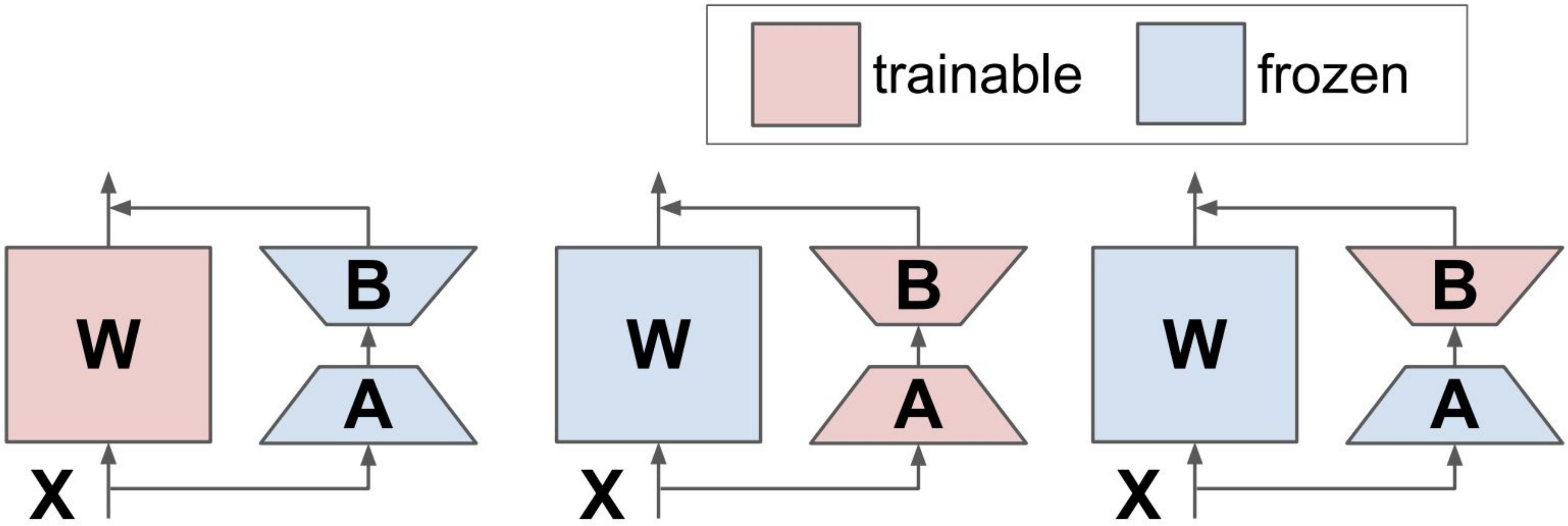


Figure 1: An overview of the proposed **Delta-LoRA** structure, compared to **LoRA**, **DyLoRA** and **AdaLoRA**. Note that **DyLoRA** and **LoRA** basically share the same architecture. $\boldsymbol{W}$ is the pre-trained weight which is frozen (signified by blue) when performing efficient-parameter fine-tuning in (a) and (b). Orange trapezoids $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{E}$ denote the trainable parameters. In our proposed Delta-LoRA, the light orange rectangle means that pre-trained weights can be updated via the delta. *Note that our proposed Delta-LoRA removes the Dropout layer to ensure reasonable delta for pre-trained matrix.*

4. LoRA+

	LoRA	LoRA+
Parameterization		
Training	$A \leftarrow A - \eta \times G_A$ $B \leftarrow B - \eta \times G_B$	$A \leftarrow A - \eta \times G_A$ $B \leftarrow B - \lambda \eta \times G_B$ $\lambda \gg 1$

5. DyLoRA (Dynamic LoRA)



(a) Full Parameter FT (b) LoRA (c) LoRA-FA

6. DP-DyLoRA

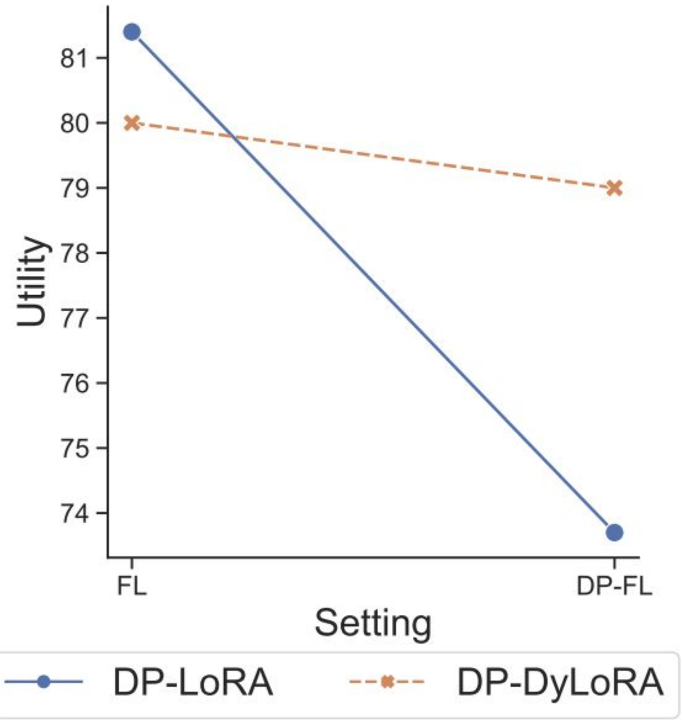


Fig. 1. Privacy-utility trade-offs of DP-LoRA and DP-DyLoRA on six datasets across three different domains under DP-FL. The utility is computed as the average of accuracy.

7. AdaLoRA (Adaptive LoRA)

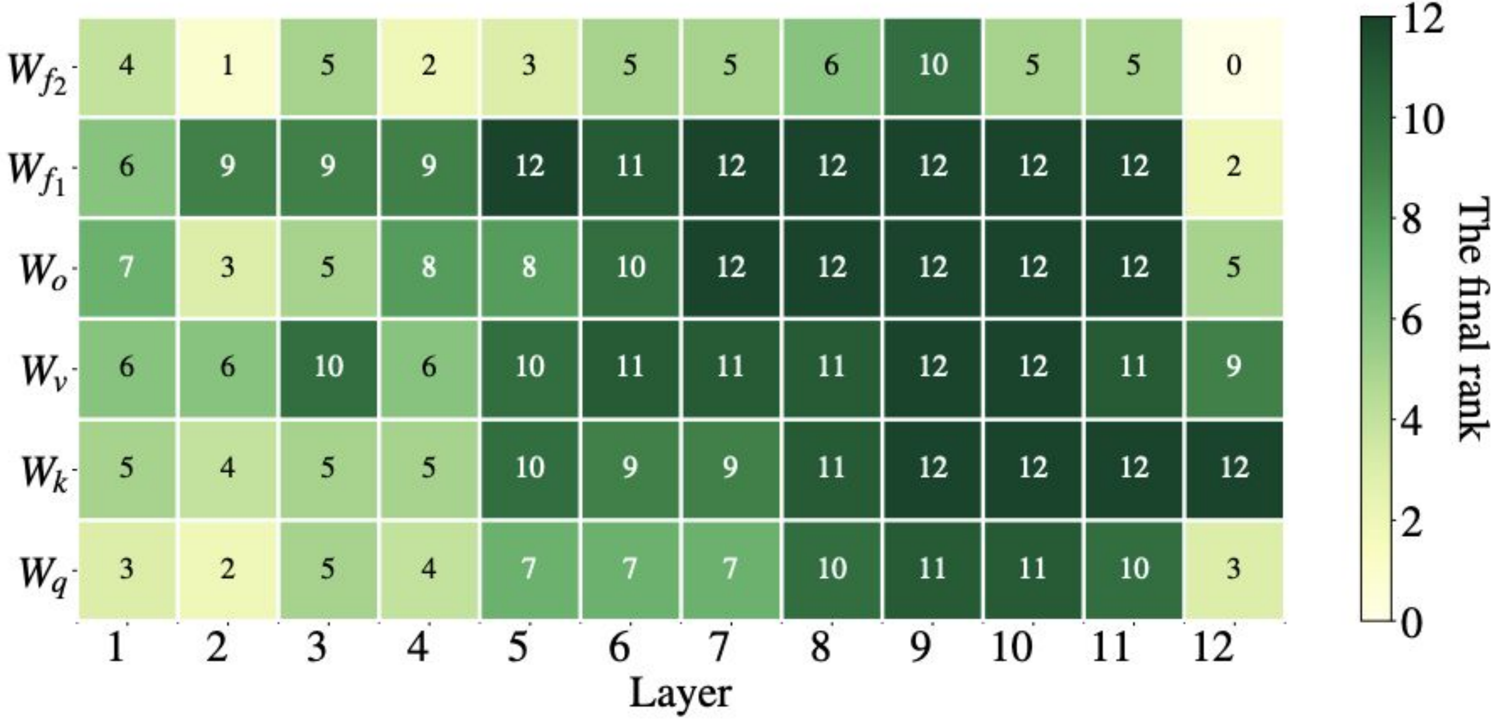


Figure 3: The resulting rank of each incremental matrix when fine-tuning DeBERTaV3-base on MNLI with AdaLoRA. Here the x-axis is the layer index and the y-axis represents different types of adapted weight matrices.

8. DoRA (Decomposed LoRA)

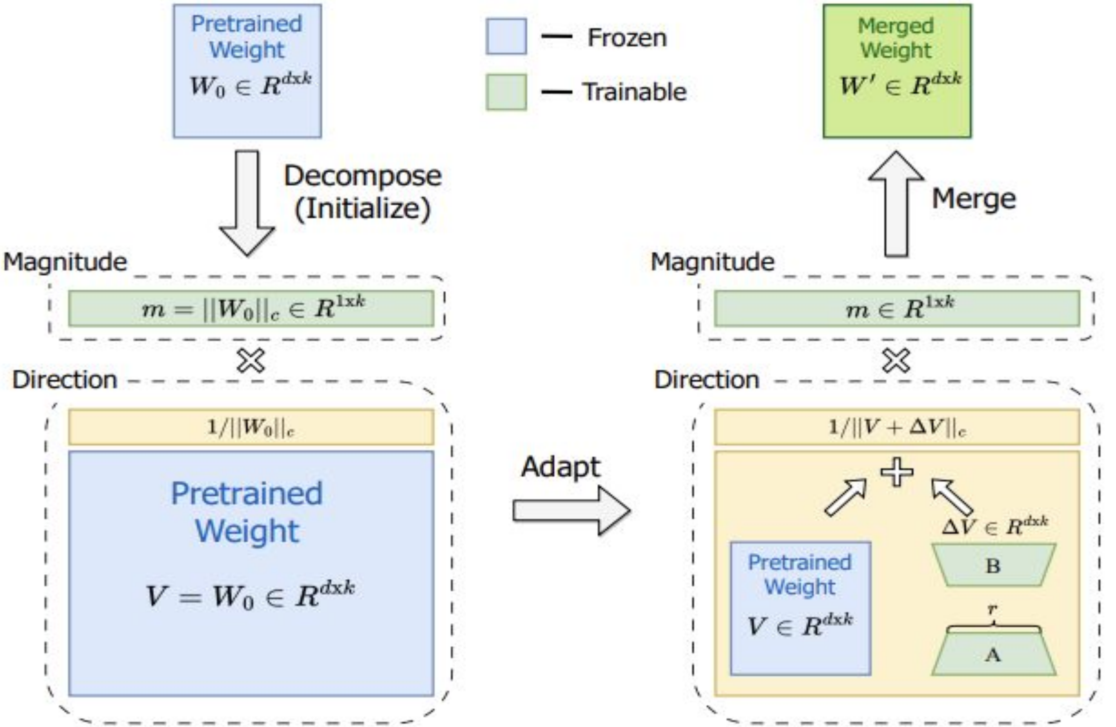


Figure 1. An overview of our proposed DoRA, which decomposes the pre-trained weight into *magnitude* and *direction* components for fine-tuning, especially with LoRA to efficiently update the direction component. Note that $|| \cdot ||_c$ denotes the vector-wise

9. VeRA (Vector-based Random Matrix Adaptation)

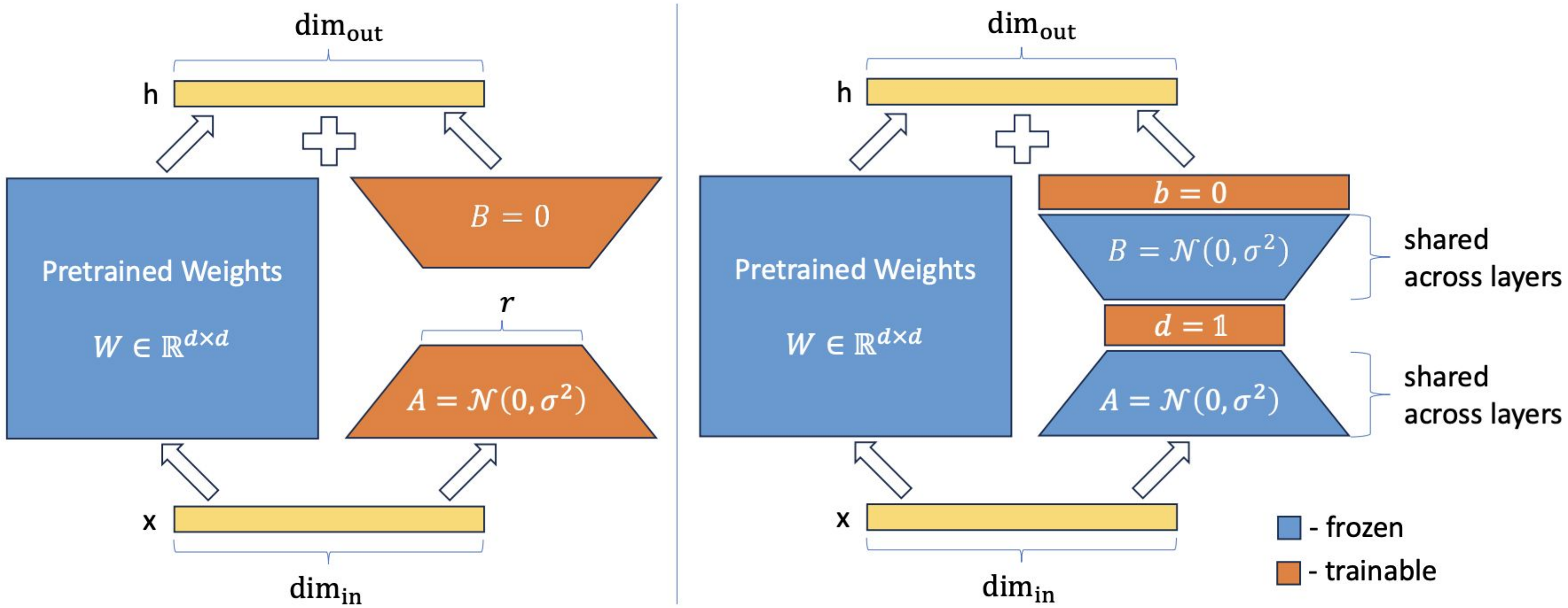


Figure 1: Schematic comparison of LoRA (left) and VeRA (right). LoRA updates the weights matrix W by training the low-rank matrices A and B, with intermediate rank r. In VeRA these matrices are frozen, shared across all layers, and adapted with trainable vectors d and b, substantially reducing the number of trainable parameters. In both cases, low-rank matrices and vectors can be merged into original weights matrix W, introducing no additional latency.

10. LoHa (Low-Rank Hadamard Product)

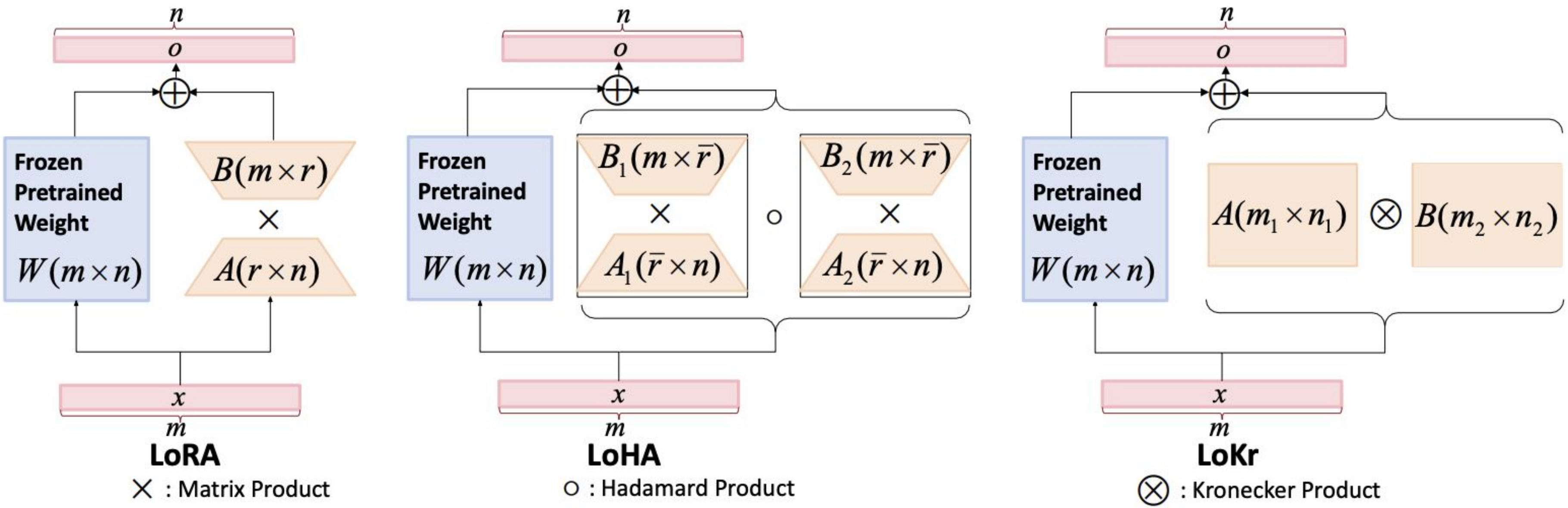
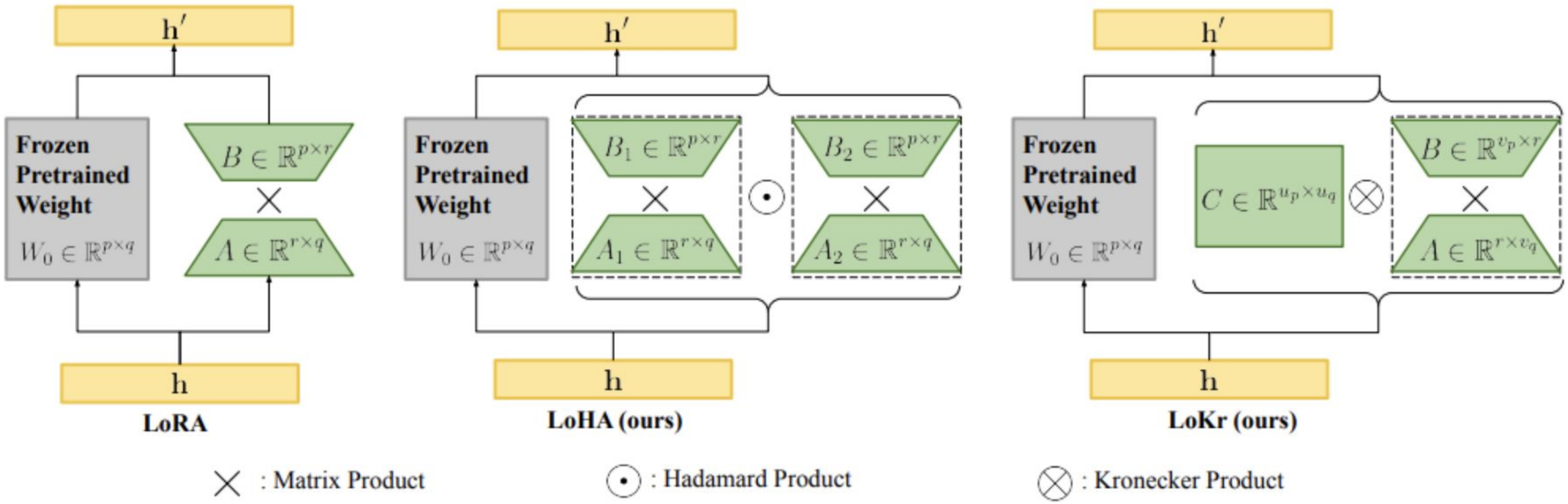


Figure 2: Diagram illustrating LoRA, LoHA, and LoKr (KronA).

11. LoKr (Low-Rank Kronecker Product)



12. LoRA-drop

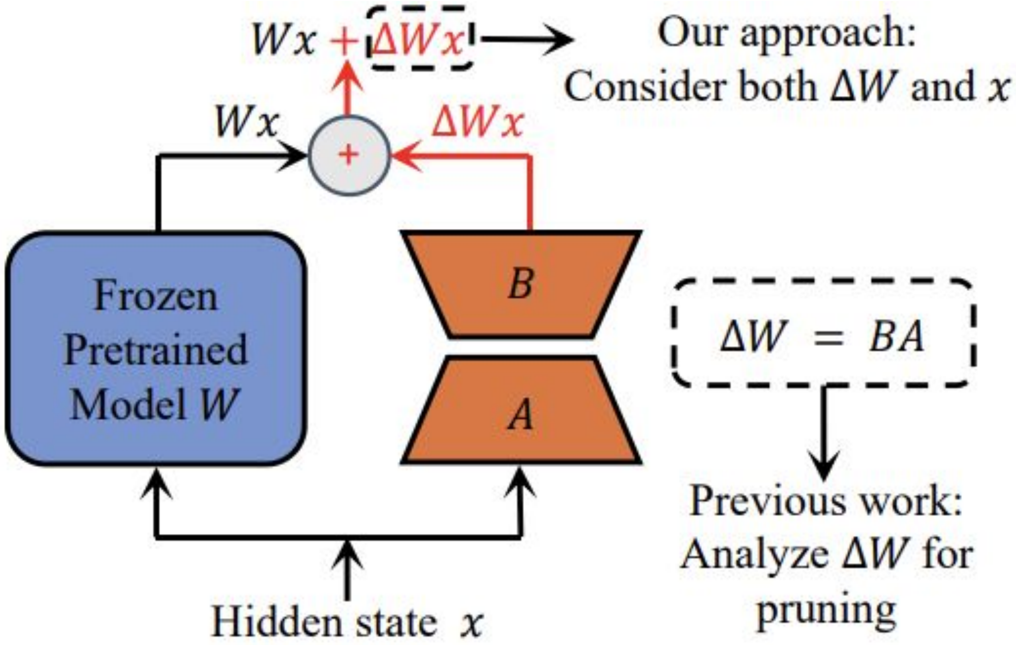


Figure 1: The diagram of LoRA. LoRA influences the pre-trained model through its output $\Delta \boldsymbol{W}\boldsymbol{x}$. This paper's method measures the importance of LoRA based on its output.

13. QLoRA (Quantized LoRA)

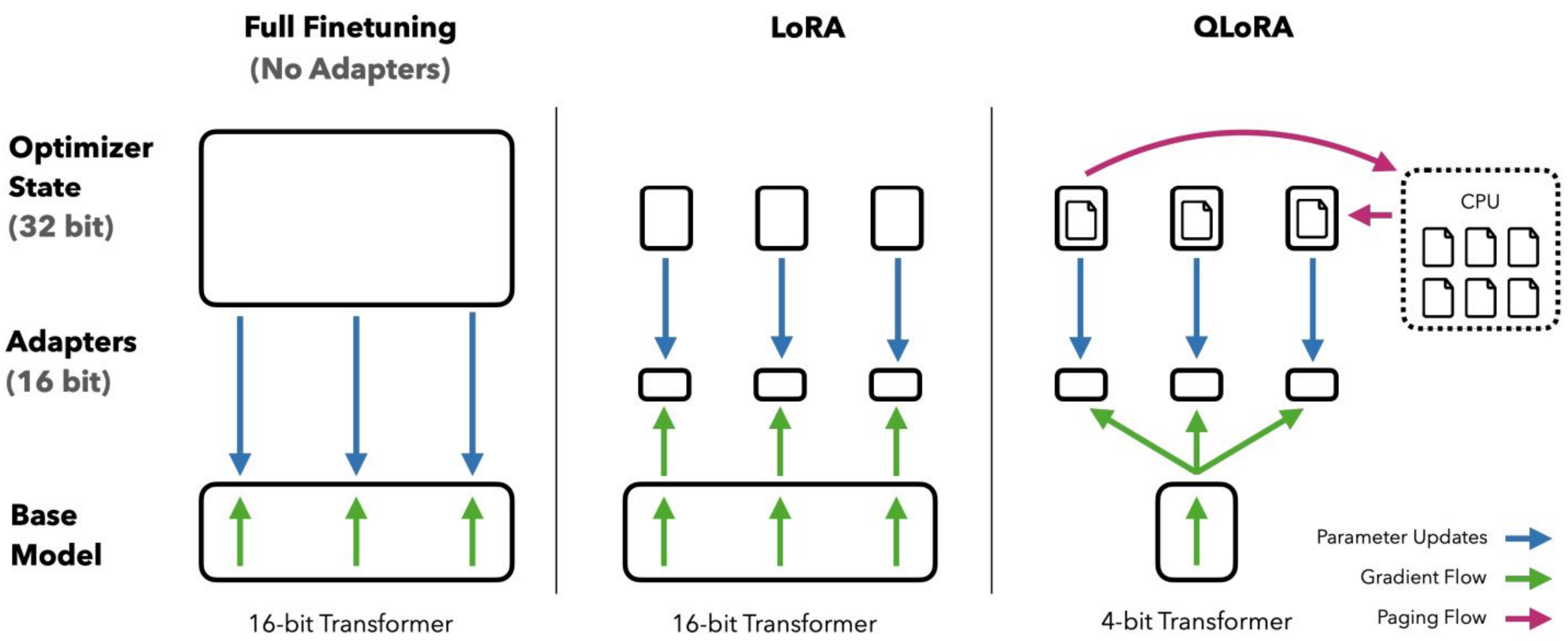


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Download Complete Guide ⬇️

https://github.com/Abonia1/lora-llm-fientuning