# Hebbian-Guided Bi-Directional Rank Adaptation for Parameter-Efficient Fine-Tuning

**Xing Zhao[1], Liu Yang[2]*, Xi-Le Zhao[3], Yueqi Yang[1], Tianqi Jiang[2],**

[1]School of Computer Science and Technology, Tianjin University
[2]School of Artificial Intelligence, Tianjin University
[3]University of Electronic Science and Technology of China, Chengdu, China
{zhaoxing975, yangliuyl, yang9868, jtq_3019}@tju.edu.cn, xlzhao122003@163.com

## Abstract

Low-Rank Adaptation (LoRA) is a widely used method to fine-tune large language models, extremely reducing computational and storage costs. But its fixed-rank design cannot well capture the varying importance across different layers, limiting its flexibility. Dynamic rank allocation methods mitigate this issue by adaptively allocating ranks during training, but most of them focus solely on either rank pruning or expansion, leading to redundant parameterization or insufficient representational capacity. To address this problem, we introduce Hebbian-Guided Bi-Directional Rank Adaptation (HeBiRA), a novel framework that bi-directionally reallocates low-rank capacity using Hebbian-inspired importance estimation. HeBiRA computes the contribution of each rank direction by measuring the synergy between activations and output gradients, and adjusts the rank bi-directionally by pruning uninformative directions while expanding those in critical layers. This mechanism flexibly redistributes the rank budget during training and also can be applied to PEFT methods such as DoRA, HiRA, and QLoRA. Experiments on multiple benchmarks and theoretical analysis show that HeBiRA consistently improves performance over baselines.

## 1 Introduction

Large language models (LLMs) (Dubey et al., 2024; DeepSeek-AI et al., 2025) have become the core infrastructure of natural language processing, advancing performance from general-purpose inference to domain-specific applications. However, the training and full parameter tuning of these models require huge computing resources and storage overhead. Therefore, parameter-efficient fine-tuning methods for large pre-trained models have become a research hotspot.

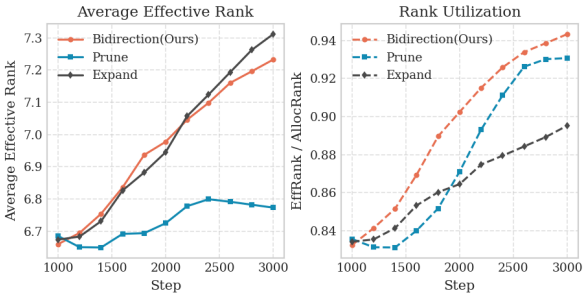



Figure 1: Effective Rank and Utilization on the STS-B dataset during training, using the same settings as Appendix D with an initial rank of 8.

Low-Rank Adaptation (LoRA) (Hu et al., 2022) reduces computational cost by decomposing model weights into trainable low-rank matrices. However, its fixed-rank allocation limits adaptability and parameter efficiency, as different layers contribute unequally to downstream tasks. Kalajdzievski (2023) showed that increasing the rank of LoRA with proper scaling can significantly improve performance. Yet higher ranks incur substantial memory, which has motivated the development of dynamic rank adaptation methods.

AdaLoRA (Zhang et al., 2023b) prunes ranks in weight matrices based on importance scores derived from gradients. IncreLoRA (Zhang et al., 2023a) incrementally allocates more parameters to important modules during training. TriAdaptLoRA (Liang et al., 2025) proposes an adaptive rank-growth strategy governed by dynamic thresholds. While the aforementioned methods are effective, they face two key limitations. First, they are restricted to either pruning or expansion, which may lead to suboptimal rank adjustments. We use the effective rank (Roy and Vetterli, 2007) as a capacity-aware metric to measure how many singular directions are meaningfully utilized (Appendix E). As shown in Figure 1, pruning alone achieves high utilization but results in few effective ranks, whereas expansion increases effective ranks but with low

*Corresponding author.



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 19112–19128

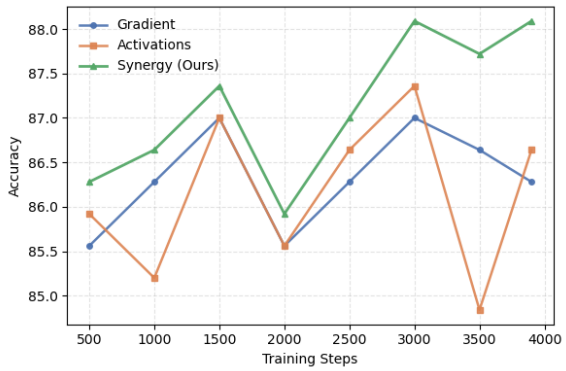



Figure 2: Accuracy comparison of different rank importance estimation strategies during training on RTE, using the same settings as Appendix D with an initial rank of 8.

utilization. Second, existing methods for rank importance estimation like PriLoRA (Benedek and Wolf, 2024) and AdaLoRA (Zhang et al., 2023b) overlook the synergy between gradients and input activations. Using only gradients reflects how strongly a direction affects the loss but does not indicate whether it is actually utilized during the forward pass. Conversely, using only input activations cannot reveal whether a direction contributes meaningfully to the task. As shown in Figure 2, rank directions that are both activated in the forward pass and influenced by gradients in the backward pass tend to be more useful, which aligns with the Hebbian principle (Do, 1949).

To address these limitations, we propose HeBiRA, a novel framework that dynamically reallocates low-rank capacity, built on two key ideas. First, we design a Hebbian-inspired importance estimator. Hebbian learning, a foundational principle in neuroscience, often summarized as "neurons that fire together wire together". In the LoRA setting, the Hebbian learning naturally translates into the interaction between input activations and output gradients. A rank direction becomes important when its input projection is strongly activated by the current input, and simultaneously, its output projection has a strong influence on the loss. Second, we introduce a bidirectional rank adjustment mechanism that goes beyond purely pruning or purely expanding ranks. Guided by the Hebbian importance signal, HeBiRA prunes redundant ranks and reallocates those in critical layers. As a result, HeBiRA maintains both high effective rank and high utilization throughout training (Figure 1), enabling the model to adapt its representational capacity as the task evolves, with better parameter efficiency. Table 1 provides a comparative overview of dynamic rank adaptation methods, demonstrating the advantages of our approach.

| Method | Pruning | Expansion | Adaptation |
|---|---|---|---|
| DyLoRA | × | × | Rank sampling |
| IncreLoRA | × | ✓ | Gradient |
| AdaLoRA | ✓ | × | Gradient |
| PriLoRA | ✓ | × | Activations |
| TriAdaptLoRA | × | ✓ | F-norms |
| **HeBiRA** | ✓ | ✓ | Grad-Act |

Table 1: Comparison of dynamic rank adaptation.

The main contributions are as follows:

1) We propose a Hebbian-inspired importance metric that combines input activations and output gradients, reflecting the functional contribution of each rank in a biologically motivated manner.

2) A bidirectional rank adaptation mechanism (HeBiRA) that dynamically prunes and expands ranks based on importance, enabling flexible allocation and improved parameter utilization.

3) HeBiRA framework can be to methods such as DoRA, HiRA, and QLoRA to achieve better results. Experimental results across diverse tasks demonstrate the effectiveness of HeBiRA over baseline methods, while theoretical analysis confirms its superior parameter efficiency.

## 2 Related Works

### 2.1 Low-rank adaptation

PEFT methods have evolved from adapter layers (Houlsby et al., 2019) and prompt tuning (Lester et al., 2021) to LoRA (Hu et al., 2022). The key insight of LoRA is that weight updates during adaptation can be effectively represented using low-rank decompositions. PiSSA (Meng et al., 2024) leverages principal SVD to initialize LoRA by truncating pre-trained weights, thereby accelerating fine-tuning convergence. QLoRA (Dettmers et al., 2023) extends LoRA with 4-bit quantization and gradient dequantization, enabling efficient fine-tuning of large models. DoRA (Liu et al., 2024) decomposes the weight update into two independent components, amplitude and direction, and applies LoRA adaptation only to the direction component.

### 2.2 Dynamic rank adaptation

AdaLoRA (Zhang et al., 2023b) adapts ranks via a SVD formulation with importance-based pruning and orthogonality regularization, but requires a

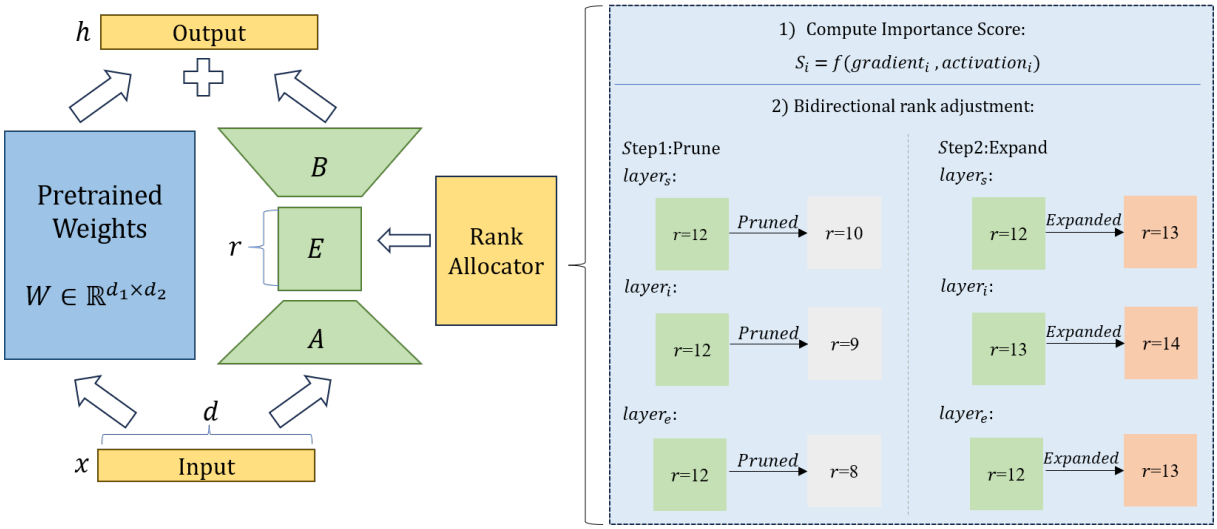



Figure 3: Overview of the proposed HeBiRA framework. **Left:** The base architecture with rank allocation. **Right:** The bidirectional rank adjustment procedure: (1) compute rank importance score $S$ based on Hebbian-inspired synergy of gradient and activation; (2) redistribute rank capacity by pruning low-importance ranks and expanding those in critical layers, allowing the model to adaptively allocate resources where they are most needed.

large initial parameter space. IncreLoRA (Zhang et al., 2023a) instead adopts a progressive rank expansion strategy, gradually increasing the model capacity, but starting with a small rank results in insufficient initial learning ability. DoRA (Mao et al., 2024) decomposes the LoRA parameter matrix into rank-one components and prunes them based on heuristic importance scores. AutoLoRA(Zhang et al., 2024) automates rank selection via meta-learned pruning of redundant singular components. TriAdaptLoRA (Liang et al., 2025) draws on neuroscience principles to introduce an adaptive rank-growth strategy controlled by dynamic thresholds. Despite their progress, these methods focus solely on either rank pruning or expansion, which may lead to suboptimal rank adjustments.

## 3 Preliminaries

### 3.1 Low-rank adaptation

LoRA (Hu et al., 2022) introduces a low-rank parameterization to model the update of a pre-trained weight matrix without modifying the original weights. Given a linear transformation $h = W^{(0)}x$, LoRA represents the update as:

$$h = W^{(0)}x + \Delta W x = W^{(0)}x + BAx, \quad (1)$$

where $W^{(0)}, \Delta W \in \mathbb{R}^{d_1 \times d_2}$, and the update is factorized into $A \in \mathbb{R}^{r \times d_2}$ and $B \in \mathbb{R}^{d_1 \times r}$ with $r \ll \{d_1, d_2\}$. During fine-tuning, the pre-trained weight $W^{(0)}$ is kept frozen, while $A$ and $B$ are trained. Typically, $A$ adopts a random Gaussion initialization and $B$ is initialized to zero.

### 3.2 Hebbian learning

Hebbian theory (Do, 1949) describes a fundamental principle of synaptic plasticity: the persistent and repeated co-activation of two neurons strengthens the synaptic efficacy between them. This idea is often summarized as "neurons that fire together wire together". Formally, the theory posits that when the presynaptic neuron first activates and consistently contributes to the activation of a postsynaptic neuron, the synaptic connection between them undergoes long-term strengthening.

Hebbian theory can generally be summarized as:

$$\Delta w_{ij} = \eta \cdot a_i \cdot o_j, \quad (2)$$

where $\Delta w_{ij}$ represents the change in the synaptic weight , $\eta$ is a suitable constant coefficient, $a_i$ is the activation value of the neuron $i$, and $o_j$ is the output value of the neuron $j$.

This co-activation principle offers an intuitive view of how low-rank adaptation behaves. When the forward activations provide a strong "presynaptic" drive and the backward gradients supply a strong "postsynaptic" signal, that direction exerts meaningful influence on learning. Consequently, its importance can be characterized by the interaction between activation strength and gradient sensitivity, mirroring the Hebbian notion that effective connections are those consistently co-activated.

## 4 Method

We propose HeBiRA, a parameter-efficient fine-tuning method based on dynamic rank adjustment. As shown in Figure 3, the LoRA incremental weight matrix is first parameterized in SVD form. A biologically inspired Hebbian importance score is then computed. Based on this score, HeBiRA performs bidirectional rank adaptation, pruning redundant ranks while expanding capacity in critical layers, followed by a final warmup stage.

### 4.1 SVD-form adaptation

We parameterize the weight increment in the form of singular value decomposition because it naturally aligns each rank direction with a singular mode of the weight matrix, providing a clear decomposition of input and output contributions. And we represent the incremental update of the pre-trained weight matrix as

$$\mathbf{W} = \mathbf{W}^{(0)} + \Delta\mathbf{W} = \mathbf{W}^{(0)} + \mathbf{BEA}, \quad (3)$$

where $\mathbf{B} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times d_2}$ are learnable factor matrices, $\mathbf{E} = \mathrm{diag}(e_1, \ldots, e_r)$ is a trainable diagonal matrix of singular values. $\mathbf{E}$ is initialized to zero, while $\mathbf{B}$ and $\mathbf{A}$ adopt a random Gaussian initialization.

LoRA's low-rank formulation can be viewed as decomposing weight updates into a set of independent "directions." In HeBiRA, we further apply SVD-form to reinterpret each rank as a triple unit. For a LoRA layer with rank $r$, each rank's corresponding triple is treated as the fundamental unit for computing importance scores and performing rank adjustments. It can be expressed as $\mathcal{G}_i = \left(\mathbf{b}_i, e_i, \mathbf{a}_i\right)$, where $\mathbf{b}_i$ and $\mathbf{a}_i$ are the $i$-th column of $\mathbf{B}$ and the $i$-th row of $\mathbf{A}$, respectively, and $e_i$ is the corresponding singular value, with $i = 1, \ldots, r$.

For a LoRA layer, let the set of rank triples at adjustment step $t$ be $\mathcal{G}^{(t)} = \{G_1^{(t)}, G_2^{(t)}, \ldots, G_r^{(t)}\}$. At the next adjustment stage, HeBiRA computes an importance score for each triple: $S_i^{(t)} = f(\mathbf{b}_i^{(t)}, e_i^{(t)}, \mathbf{a}_i^{(t)})$, where $f(\cdot)$ integrates activation-driven and gradient-driven in a Hebbian-inspired manner. Based on $S_i^{(t)}$, the triple set is updated through bidirectional rank adjustment: $\mathcal{G}^{(t+1)} = \left(\mathcal{G}^{(t)} \setminus \mathcal{G}^{(t)}_{\text{pruned}}\right) \cup \mathcal{G}^{(t)}_{\text{expanded}}$, where $\mathcal{G}^{(t)}_{\text{pruned}}$ contains the least important triples to be removed, and $\mathcal{G}^{(t)}_{\text{expanded}}$ introduces new triples initialized for critical directions.

### 4.2 Hebbian-inspired importance evaluation

Hebbian learning is a foundational principle in neuroscience that characterizes synaptic plasticity. Its classical formulation states that when the activation of a presynaptic neuron consistently precedes and contributes to the activation of a postsynaptic neuron, the synaptic connection between them is strengthened.

LoRA's low-rank parameterization naturally aligns with this view. Within the LoRA framework, the low-rank decomposition expresses the weight update as a sum of rank-1 directions. For a rank-r LoRA module, the update in SVD-form is: $\Delta\mathbf{W} = \mathbf{BEA}$. Each rank $i$ corresponds to a directional component parameterized by the pair $\left(\mathbf{b}_i, e_i, \mathbf{a}_i\right)$. From a functional perspective, these rank directions act as "connection pathways", whose contribution to learning is jointly determined by two factors: input-driven activation of the projection $\mathbf{a}_i$ and output-driven sensitivity captured by the gradient flowing through $\mathbf{b}_i$. These two factors naturally correspond to the pre-synaptic activation and post-synaptic activation in Hebbian learning, respectively.

Consider a LoRA layer with weight $\Delta\mathbf{W}$ of shape $(d_1, d_2)$. For language models, this LoRA layer takes in input activations $X$ with a shape of $(N \times L, d_2)$, where $N$ and $L$ are batch and sequence dimensions respectively. For each triple $G_i$, we define the input activation strength:

$$\alpha_i = \left\| \mathbf{X}\mathbf{a}_i^\top \right\|_2, \quad (4)$$

where $\left\| \mathbf{X}\mathbf{a}_i^\top \right\|_2$ evaluates the $\ell_2$ norm of this rank aggregated across $N \times L$ different tokens.

And the output gradient sensitivity is obtained by taking the $\ell_2$ norm of the gradient with respect to the output direction $\mathbf{b}_i$ under the training loss $\mathcal{L}$:

$$\gamma_i = \left\| \nabla_{\mathbf{b}_i} \mathcal{L} \right\|_2. \quad (5)$$

Inspired by Hebbian co-activation, the input activation strength $\alpha_i$ measures pre-synaptic activity along $\mathbf{a}_i$, while the gradient sensitivity $\gamma_i$ captures the post-synaptic response along $\mathbf{b}_i$. The dependency and proof of $\gamma_i$ on $\alpha_i$ is formally shown in the Appendix A.

Thus, the final importance score is computed as the multiplicative interaction of the singular value, input-driven activation, and gradient-driven sensitivity:

$$S_i = f(e_i, \alpha_i, \gamma_i) = |e_i| \cdot \alpha_i \cdot \gamma_i. \quad (6)$$

To ensure temporal stability of rank importance and prevent fluctuations caused by noisy batches, the raw importance score is further smoothed using an exponential moving average:

$$\tilde{S}_i^{(t)} = \beta\, \tilde{S}_i^{(t-1)} + (1-\beta)\, S_i^{(t)}, \tag{7}$$

where $S_i^{(t)}$ is the raw importance at step $t$, $\tilde{S}_i^{(t)}$ is the smoothed score, and $\beta \in [0, 1)$.

This formulation naturally highlights rank directions that are both strongly driven by the current input activation and highly influential to the optimization objective. A rank direction with high activation but negligible gradient is uninformative for reducing the loss, while one with large gradient but weak activation is rarely engaged by the input.

### 4.3 Bidirectional rank adjustment strategy

**Pruning phase.** In the pruning stage, we first compute the Hebbian-inspired importance score $S_i$ for every rank-level triple $G_i = (\mathbf{b}_i, e_i, \mathbf{a}_i)$ across all LoRA layers. The scores from all layers are then aggregated into a global candidate pool and sorted in ascending order. We identify the $k$ least important rank directions and mark them for removal:

$$\mathcal{P} = \{\, G_{(1)}, G_{(2)}, \ldots, G_{(k)} \,\}, \tag{8}$$

where $G_{(j)}$ denotes the triple with the $j$-th smallest importance value. The set $\mathcal{P}$ determines the rank directions to be pruned in this adjustment cycle.

**Expansion phase.** After pruning, each LoRA layer retains a set of rank-level triples with their corresponding importance scores. To quantify the layer's need for additional capacity, we assign each layer a layer-level importance score defined as the minimum importance among its remaining rank directions:

$$S_{\text{layer}}^{(\ell)} = \min_{G_i \in \mathcal{G}^{(\ell)}} S_i, \tag{9}$$

which reflects the "weakest surviving direction" in layer $\ell$. A high value of $S_{\text{layer}}^{(\ell)}$ indicates that even the least important remaining direction in this layer is relatively significant, suggesting that the layer is capacity-limited and would benefit from expansion.

All layers are globally ranked based on $S_{\text{layer}}^{(\ell)}$, and the $k$ rank resources removed during pruning are reassigned to the top-$k$ layers with the highest scores:

$$\mathcal{E} = \{\, \ell_{(1)}, \ell_{(2)}, \ldots, \ell_{(k)} \,\}, \tag{10}$$

where $\ell_{(j)}$ denotes the layer with the $j$-th largest $S_{\text{layer}}^{(\ell)}$. For each layer in $\mathcal{E}$, one new rank direction is introduced, forming the expansion set and preserving strict global rank budget.

We employ a random initialization strategy for new rank directions, where new components $(A_{new}, B_{new}, E_{new})$ are initialized using a small-scale Gaussian distribution to ensure they start from a neutral but learnable state. To ensure training stability, we adopt a "Weight Inheritance with Optimizer Reset" approach. Furthermore, we use a dynamic scaling factor $(1/\text{ranknum})$ in the forward pass to automatically balance the output magnitude as the rank increases.

Under the constraint of a fixed total rank budget, the bidirectional adjustment strategy removes less important redundant parameters and reallocates them to more critical LoRA layers, thereby improving parameter efficiency and enhancing model performance. For detailed bidirectional rank adjustment strategy, please refer to Algorithm 1 in Appendix H.

We keep ranks fixed for the first $t_i$ steps to stabilize early representations, then perform rank adaptation every $\Delta_T$ steps. After this adaptive phase, the final rank configuration is frozen and the model is final fine-tuned for $t_f$ steps.

### 4.4 Parametric efficiency analysis

**Theorem 4.1 (Pareto-optimal efficiency)** *Consider $L$ LoRA layers with per-layer ranks $r_l$ constrained by a fixed total budget $R_{\text{total}}$: $\sum_{l=1}^{L} r_l = R_{\text{total}}$. Let $I_l$ denote the total importance score of layer $l$, defined as the aggregate importance of all ranks within that layer. The necessary condition for Pareto-optimal parameter efficiency is:*

$$r_l \propto I_l^{\frac{1}{\alpha+1}}, \quad 0 < \alpha < 1. \tag{11}$$

*That is, layers with higher importance scores should be assigned more ranks.*

**Inference: dynamic vs. static strategy** A dynamic rank adjustment strategy that updates $r_l$ in response to changes in $I_l$ during training can iteratively approach the Pareto-optimal condition 11. Static strategies, which fix $\{r_l\}$ at initialization, cannot adapt to evolving layer importance, and thus are generally less efficient in parameter utilization and model performance. See Appendix B for the complete derivation.

### 4.5 Consolidated hyperparameter table

We have now standardized all symbols and included a consolidated hyperparameter table to provide a clear, actionable guide for adaptation (Table 2).

| Symbol | Definition |
|---|---|
| $r$ | The initial rank of each incremental matrix. |
| $t_i$ | The steps of initial warmup. |
| $t_f$ | The steps of final warmup. |
| $\Delta_T$ | The time interval between two rank adjustments. |
| $k$ | The number of ranks pruned/expanded in one adjustment. |
| $\beta$ | The smoothing coefficient in the calculation of importance scores. |

Table 2: Notation and definitions.

## 5 Experiments

### 5.1 Models and datasets

**Natural language understanding (NLU).** We adopt DeBERTaV3-base (He et al., 2023) and fine-tune it on the GLUE benchmark (Wang et al., 2019), covering eight standard tasks.

**Mathematical reasoning and code generation.** We employ LLaMA-2-7B (Touvron et al., 2023) and LLaMA-3-8B (Dubey et al., 2024) for evaluation on mathematical reasoning, where the models are fine-tuned on MetaMathQA (Yu et al., 2024) and assessed on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). For code generation, the models are fine-tuned on CodeFeedback (Zheng et al., 2024) and evaluated on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).

**Summarization.** We use BART-large (Lewis et al., 2019) for summarization on XSum (Narayan et al., 2018), which evaluate the ability to generate concise and faithful summaries.

### 5.2 Baselines and settings

We compare our method against a broad range of parameter-efficient fine-tuning (PEFT) approaches, including LoRA and its variants, as well as dynamic rank adaptation methods:

- **LoRA-based methods:** LoRA (Hu et al., 2022), LoRA+ (Hayou et al., 2024) , PiSSA (Meng et al., 2024), DoRA (Liu et al., 2024), HiRA (Huang et al., 2025), RandLoRA (Albert et al., 2025), QLoRA (Dettmers et al., 2023), RaSA (He et al., 2025).
- **Dynamic rank methods:** AdaLoRA (Zhang et al., 2023b), IncreLoRA (Zhang et al., 2023a), DyLoRA (Valipour et al., 2023), TriAdaptLoRA (Liang et al., 2025).

The settings described in the Appendix D.

### 5.3 Results

**Natural Language Understanding.** Table 3 reports the performance of different PEFT methods on eight tasks from the GLUE benchmark with rank of $r = 8$. Overall, our method consistently outperforms existing baselines, achieving the highest average score, demonstrating superior generalization across both sentence-level and sentence-pair classification tasks.

**Mathematical Reasoning and Code Generation.** As shown in Table 4, our method (HeBiRA) consistently surpasses the baselines. On LLaMA2-7B, HeBiRA improves performance over all four datasets. And for LLaMA3-8B, HeBiRA also delivers the best average score, demonstrating consistent advantages in both mathematical reasoning and code generation.

**Summarization.** Table 7 shows the results on summarization task. Compared with LoRA and AdaLoRA, our method (HeBiRA) achieves the best performance across all Rouge metrics while using the same parameter budget as LoRA (2.06M).

### 5.4 HeBiRA as a general paradigm

To further validate the generality of our proposed method, we integrate HeBiRA into several representative PEFT approaches, including DoRA, HiRA and QLoRA, across both natural language understanding (NLU) and code generation tasks. The results are reported in Table 5 and Table 6. These results suggest that HeBiRA can be applied as a paradigm to existing PEFT methods, improving their performance across models and tasks.

### 5.5 Analysis

#### 5.5.1 Ablation of bidirectional strategies

We propose a bidirectional rank adjustment strategy and validate it via an ablation study. From Table 8, compared with Prune-only or Expand-only variants, our method balances parameter allocation and model capacity.

| Method | #Params(%) | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| Full FT | 100 | 89.90 | 95.63 | 69.19 | 92.40 | 94.03 | 83.75 | 89.46 | 91.60 | 88.25 |
| LoRA | 0.71 | 90.65 | 94.95 | 69.82 | 91.99 | 93.87 | 85.20 | 89.95 | 91.60 | 88.50 |
| PiSSA | 0.71 | 90.30 | 95.53 | 71.41 | 91.92 | 94.07 | 88.09 | 90.20 | 91.54 | 89.13 |
| RandLoRA | 0.71 | 90.36 | 95.99 | 69.82 | 92.06 | 93.74 | 86.28 | 90.69 | 91.34 | 88.79 |
| LoRA+ | 0.71 | 90.28 | 95.30 | 70.25 | 92.09 | 94.01 | 86.28 | 90.93 | 91.54 | 88.84 |
| DyLoRA | 0.71 | 89.51 | 95.18 | 69.82 | 91.97 | 94.29 | 85.92 | 89.95 | 91.74 | 88.55 |
| IncreLoRA | 0.71 | 90.62 | 95.72 | 70.20 | 91.91 | 94.36 | 86.88 | 90.11 | 91.38 | 88.90 |
| AdaLoRA | 0.68 | **90.76** | **96.10** | 71.45 | 92.23 | 94.55 | 88.09 | 90.69 | 91.84 | 89.46 |
| TriAdaptLoRA | 0.71 | 90.64 | 95.68 | 71.60 | 92.09 | 94.37 | 87.84 | 90.77 | 91.79 | 89.35 |
| HeBiRA(Ours) | 0.71 | 90.59 | **96.10** | **72.44** | **92.38** | **94.65** | **88.45** | **91.67** | **92.08** | **89.80** |

Table 3: Performance comparison of different PEFT methods on GLUE benchmark (rank $r = 8$) on DeBERTaV3-base.

| Model | Method | #Params(%) | GSM8K | MATH | HumanEval | MBPP |
|---|---|---|---|---|---|---|
| LLaMA2-7B | LoRA | 0.15 | 52.9 | 7.6 | 26.0 | 34.7 |
| | DoRA | 0.17 | 56.1 | 9.8 | 31.7 | 38.4 |
| | QLoRA | 0.28 | 50.3 | 6.1 | 24.8 | 32.8 |
| | RaSA | 0.15 | 56.4 | 9.7 | 28.0 | 36.2 |
| | AdaLoRA | 0.22 | 52.0 | 8.2 | 26.2 | 35.2 |
| | HeBiRA | 0.15 | **57.5** | **10.1** | **32.3** | **38.9** |
| LLaMA3-8B | LoRA | 0.13 | 81.3 | 39.0 | 64.0 | 69.0 |
| | DoRA | 0.15 | **81.4** | 37.2 | 65.2 | 72.0 |
| | QLoRA | 0.23 | 81.1 | 39.6 | 67.1 | 70.6 |
| | RaSA | 0.13 | 81.0 | 36.2 | 67.1 | 69.6 |
| | AdaLoRA | 0.20 | 81.0 | 39.6 | 65.9 | 72.0 |
| | HeBiRA | 0.13 | 81.1 | **40.2** | **67.7** | **72.5** |

Table 4: Performance comparison of different PEFT methods on LLaMA2-7B and LLaMA3-8B.

#### 5.5.2 Layer-wise rank utilization

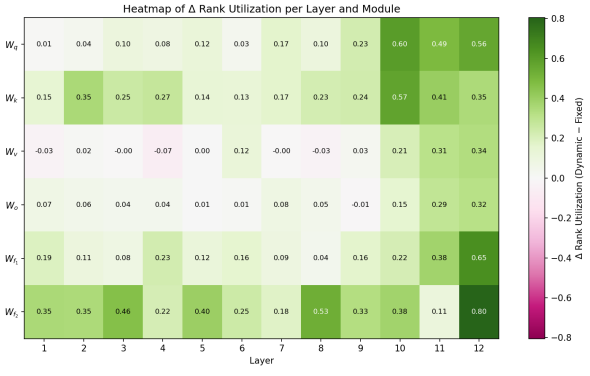



Figure 4: Heatmap of rank utilization differences between HeBiRA and fixed-rank LoRA.

To further understand how dynamic rank adaptation redistributes model capacity, we conduct a layer-wise utilization analysis comparing our method with fixed-rank LoRA. For each layer, we compute the effective rank (Appendix E) of the update matrix and normalize it by the allocated rank, obtaining a utilization score. We then compute the difference between the dynamic and fixed settings, denoted as $\Delta u = u_{\text{dynamic}} - u_{\text{fixed}}$. Figure 4 shows the rank utilization differences on QQP dataset, highlighting that ours method allocates capacity more effectively, with green indicating layers which outperforms the fixed-rank baseline.

#### 5.5.3 Rank adaptation across LoRA modules

We track the training dynamics on SST-2 and observe clear divergence across LoRA modules within `layer5`. In Figure 5, the `intermediate` layer maintains high importance and is allocated additional rank, whereas `key_proj` exhibits low importance and decreases its rank. Other modules stay relatively stable throughout training.

#### 5.5.4 Importance-driven causal testing

In Figure 6, we test causality by randomizing the importance order during training, making rank adjustment random. The performance drop across datasets confirms that importance-to-rank assign-

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| DoRA | 90.44 | 96.33 | 70.18 | 91.28 | 94.20 | 88.09 | 89.95 | 91.36 | 88.98 |
| DoRA+HeBiRA | 90.64 | 96.10 | 71.90 | 92.09 | 94.29 | 89.17 | 91.67 | 92.20 | **89.76** |
| HiRA | 90.32 | 95.64 | 70.44 | 91.97 | 94.27 | 86.28 | 90.44 | 91.66 | 88.88 |
| HiRA+HeBiRA | 90.44 | 95.99 | 72.72 | 92.35 | 93.68 | 87.73 | 90.69 | 92.02 | **89.45** |

Table 5: DeBERTaV3-base model NLU Benchmark results.

| Model | Method | MBPP | MBPP+ |
|---|---|---|---|
| LLaMA2-7B | DoRA | 38.4 | 28.3 |
| | +HeBiRA | **39.7** | **31.2** |
| | QLoRA | 32.8 | 27.0 |
| | +HeBiRA | **34.7** | 27.0 |
| LLaMA3-8B | DoRA | 72.0 | 61.4 |
| | +HeBiRA | **73.0** | **63.0** |
| | QLoRA | 70.6 | 60.6 |
| | +HeBiRA | **74.3** | **63.8** |

Table 6: LLaMA models code generation results.

| Method | Rouge-1 | Rouge-2 | Rouge-L |
|---|---|---|---|
| LoRA | 43.39 | 20.28 | 35.31 |
| AdaLoRA | 43.96 | 20.56 | 35.63 |
| HeBiRA | **44.04** | **20.85** | **35.96** |

Table 7: Performance comparison of different PEFT methods on XSum.

ment is effective.

#### 5.5.5 Comparison of importance criteria

We evaluate three pruning criteria: activation only, gradient only, and activation-gradient synergy, by iteratively removing the least important $k$ ranks under each metric using DeBERTaV3-base.

As shown in Table 9, the synergy-based criterion yields the highest accuracy, showing that the synergy of activation and gradient offers a more reliable measure of rank importance.

#### 5.5.6 Analysis of rank adjustment size

Our method performs rank updates every $T$ steps, where the adjustment size $k$ (number of ranks pruned/expanded) critically affects performance. We evaluate adjustment sizes $\{4, 8, 12, 24\}$ with initial rank $r = 8$ to analyze this effect.

As shown in Table 10, when the dynamic rank adjustment is small, the model's ability to improve is limited; when it is large, pruning trained ranks and introducing new ones can destabilize training. The optimal adjustment value depends on both the model architecture and the initial rank size.

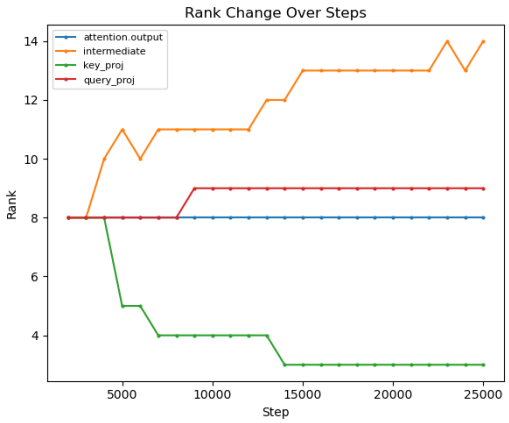



Figure 5: Evolution of dynamic rank allocation.

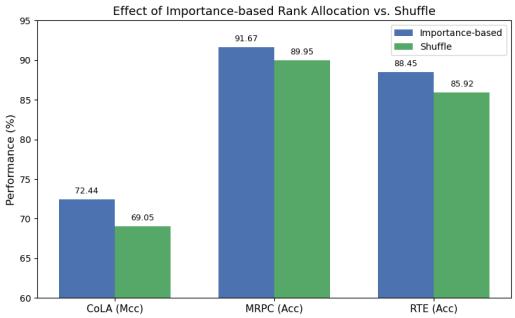



Figure 6: Causal test comparing importance-based allocation with random shuffle.

#### 5.5.7 Further experiments

We further provide complementary analyses in Appendix C, including analysis on the different rank budgets (Table 12) and update interval (Table 15), a comparison of importance estimation methods (Table 16), spearman correlation (Figure 8), results under low-rank budgets (Table 13) and statistical rigor (Table 11), offering a more comprehensive view of our design choices.

## 6 Conclusion

In this work, we introduced HeBiRA, a Hebbian-guided bidirectional rank adaptation framework for low-rank fine-tuning. Our method uses a biologically inspired importance estimator that captures

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 90.65 | 94.95 | 69.82 | 91.99 | 93.87 | 85.20 | 89.95 | 91.60 | 88.50 |
| +Prune | 90.45 | 95.53 | 71.46 | 92.16 | 94.45 | 88.09 | 89.71 | 91.98 | 89.23 |
| +Expand | 90.55 | 96.22 | 71.51 | 92.16 | 94.58 | 88.45 | 90.20 | 91.85 | 89.44 |
| HeBiRA | 90.59 | 96.10 | 72.44 | 92.38 | 94.65 | 88.45 | 91.67 | 92.08 | **89.80** |

Table 8: Performance comparison with different strategies.

| Dataset | LoRA | activation | grad | synergy |
|---|---|---|---|---|
| STS-B | 91.60 | 91.73 | 91.84 | **91.98** |
| RTE | 85.92 | 86.64 | 87.72 | **88.09** |

Table 9: Comparison of different importance criteria.

| **Dataset** | k=4 | k=8 | k=12 | k=24 |
|---|---|---|---|---|
| CoLA | 71.48 | 72.04 | **72.44** | 72.13 |
| MRPC | 91.18 | 90.69 | **91.67** | 92.08 |
| STS-B | 91.82 | 91.89 | **92.08** | 91.80 |

Table 10: Performance comparison with different rank adjustment sizes on DeBERTaV3-base.

the synergy between input activations and output gradients. This Hebbian signal enables effective pruning of redundant ranks and reallocation toward more informative layers. Furthermore, HeBiRA incorporates a bidirectional adjustment mechanism, allowing the model to maintain high effective rank and high utilization throughout training and to better align with the evolving needs of the task.

## Limitations

Although our method demonstrates promising results, several limitations remain. First, the rank adjustment size is controlled by a fixed hyperparameter, and learning it dynamically is left for future work. Second, we will explore more task-aware measures of importance, apply our method to larger models, and extend it to various tasks. And future research should consider responsible usage and safeguards in downstream applications.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62476194, U23B2049).

## References

Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Cristian Rodriguez Opazo, Anton van den Hengel, and Ehsan Abbasnejad. 2025. Randlora: Full rank parameter-efficient fine-tuning of large models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Nadav Benedek and Lior Wolf. 2024. Prilora: Pruned and rank-increasing low-rank adaptation. In *Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, March 17-22, 2024*, pages 252–263. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Hebb Do. 1949. The organization of behavior: A neuropsychological theory. *Wiley*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman,

Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhiwei He, Zhaopeng Tu, Xing Wang, Xingyu Chen, Zhijie Wang, Jiahao Xu, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, and Rui Wang. 2025. Rasa: Rank-sharing low-rank adaptation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *NeurIPS Datasets and Benchmarks*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. 2025. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *CoRR*, abs/2312.03732.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Yao Liang, Yuwei Wang, and Yi Zeng. 2025. Tri-adaptlora: Brain-inspired triangular adaptive low-rank adaptation for parameter-efficient fine-tuning. *CoRR*, abs/2501.08008.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11662–11675. Association for Computational Linguistics.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *CoRR*, abs/1808.08745.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference, EUSIPCO 2007, Poznan, Poland, September 3-7, 2007*, pages 606–610. IEEE.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *EACL*, pages 3266–3279.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *CoRR*, abs/2308.12043.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *NAACL-HLT*, pages 5048–5060.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 12834–12859. Association for Computational Linguistics.

## A Proof of Hebbian Correspondence in LoRA

We derive the gradient for the LoRA update matrix $\mathbf{B}$ and show its correspondence to Hebbian co-activation in the context of rank importance estimation.

Given the low-rank decomposition of the weight update in LoRA: $\Delta\mathbf{W} = \mathbf{BEA}$. where $\mathbf{B} \in \mathbb{R}^{d_1\times r}$ , $\mathbf{A} \in \mathbb{R}^{r\times d_2}$ and $\mathbf{E} = \mathrm{diag}(e_1, \ldots, e_r)$. the forward pass is:

$$\mathbf{Y} = \mathbf{X}\Delta\mathbf{W}^\top = \sum_{i=1}^{r} e_i(\mathbf{X}\mathbf{a}_i^\top)\mathbf{b}_i^\top. \tag{12}$$

Here, $\mathbf{X} \in \mathbb{R}^{(NL)\times d_2}$ is the input, and $\mathbf{Y} \in \mathbb{R}^{(NL)\times d_1}$ is the output. We define $\mathbf{u}_i = \mathbf{X}\mathbf{a}_i^\top$. And $\mathbf{G} = \frac{\partial\mathcal{L}}{\partial\mathbf{Y}} \in \mathbb{R}^{(NL)\times d_1}$ is the gradient of the loss with respect to the output.

Consider the $i$th rank contribution to the output: $\mathbf{Y}_i = e_i u_i \mathbf{b}_i^\top \in \mathbb{R}^{(NL)\times d_1}$. For the $m$-th sample, $\mathbf{G}_m = \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{i,m}} \in \mathbb{R}^{1\times d_1}$ and $\mathbf{y}_{i,m} = e_i u_{i,m}\mathbf{b}_i^\top \in \mathbb{R}^{1\times d_1}$ denotes the gradient vector and the output vector.

The Jacobian of $\mathbf{y}_{i,m}$ with respect to $\mathbf{b}_i$ is:

$$\frac{\partial\mathbf{y}_{i,m}}{\partial\mathbf{b}_i} = e_i u_{i,m} I, \tag{13}$$

where $I$ is the identity matrix. Thus, the gradient becomes:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{b}_i} &= \sum_{m=1}^{NL} \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{i,m}}\frac{\partial\mathbf{y}_{i,m}}{\partial\mathbf{b}_i} = \sum_{m=1}^{NL} \mathbf{G}_m\, e_i\, u_{i,m} I \\ &= \sum_{m=1}^{NL} \mathbf{G}_m\, e_i\, u_{i,m} = e_i\left(\mathbf{u}_i^\top \mathbf{G}\right).\end{aligned} \tag{14}$$

Therefore, the gradient update of $\mathbf{b}_i$ is indeed influenced by the forward activation passing through $\mathbf{X}\mathbf{a}_i^\top$.

which corresponds to the Hebbian learning rule:

$$\Delta w_{ij} \propto (\text{activation}) \times (\text{gradient sensitivity}).$$

This shows that the LoRA rank importance estimator is directly related to the Hebbian principle of co-activation.

## B Proof of Pareto-optimal Parameter Efficiency

We formalize the rank allocation problem under a fixed total rank budget $R_{\text{total}}$ using a Lagrangian framework. Consider $L$ LoRA layers, each assigned a rank $r_l > 0$, with layer importance scores $I_l > 0$. We observe that the performance gain from increasing the rank exhibits diminishing returns, a phenomenon that is consistently observed across different rank budgets in the experiments of Huang et al. (2025). So we model the layer-wise loss contribution using a monotonically decreasing function with diminishing marginal effects.:

$$L(\{r_l\}) = \sum_{l=1}^{L} I_l r_l^{-\alpha}, \quad 0 < \alpha < 1, \tag{15}$$

subject to the total rank budget constraint

$$\sum_{l=1}^{L} r_l = R_{\text{total}}. \tag{16}$$

Our goal is to minimize 15 subject to 16, yielding the most parameter-efficient rank allocation.

**Lagrangian formulation.** We construct the Lagrangian

$$\mathcal{J}(\{r_l\}, \lambda) = \sum_{l=1}^{L} I_l r_l^{-\alpha} + \lambda\Big(\sum_{l=1}^{L} r_l - R_{\text{total}}\Big), \tag{17}$$

where $\lambda$ is the Lagrange multiplier for the total rank constraint.

**Optimality condition.** Taking the derivative of 17 with respect to $r_l$ and setting it to zero for optimality, we obtain

$$\frac{\partial \mathcal{J}}{\partial r_l} = -\alpha I_l r_l^{-(\alpha+1)} + \lambda = 0. \tag{18}$$

Equation 18 implies that

$$\alpha I_l r_l^{-(\alpha+1)} = \lambda, \quad \forall l. \tag{19}$$

Since the right-hand side is independent of $l$, we have

$$r_l^{\alpha+1} \propto I_l \quad \Longrightarrow \quad r_l \propto I_l^{\frac{1}{\alpha+1}}. \tag{20}$$

**Global optimality.** Each term $I_l r_l^{-\alpha}$ is strictly convex in $r_l > 0$, so the total objective 15 is strictly convex, and the constraint 16 is linear. Therefore, any stationary point satisfying 19 is the unique global minimizer. Thus, this is the globally Pareto-optimal rank allocation.

**Implications.** Compared to any static allocation (uniform $r_l = R_{\text{total}}/L$), the dynamic allocation achieves a strictly lower loss in 15 whenever the importance scores $I_l$ are not all equal. This formally establishes that allocating ranks proportionally to $I_l^{\frac{1}{\alpha+1}}$ is pareto-optimal under the given model.

For the SST-2 dataset, both LoRA(using SVD triples) and HeBiRA methods are used to verify the above results, We select $\alpha$ as $\alpha = \frac{1}{2}$, the special case recovers the prior result $r_l \propto I_l^{2/3}$.

As shown in Figure 7, under the dynamic strategy, the cross-layer variance of $I_l^{2/3}/r_l$ gradually decreases during training, indicating that the model progressively approaches the Pareto optimal condition. Moreover, the variance under the dynamic strategy is consistently lower than that of the static LoRA strategy, suggesting that the dynamic rank adjustment achieves a more pareto-efficient parameter allocation.

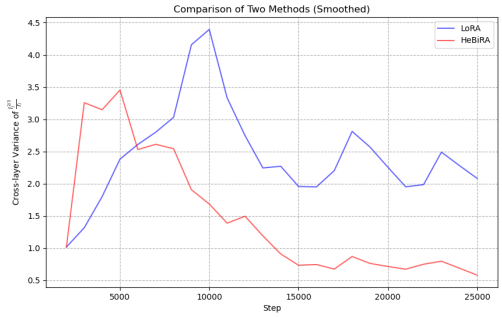



Figure 7: Comparison of Cross-layer variance of $I_l^{2/3}/r_l$ during SST-2 training.

## C Results of Further Experiments

### C.1 Statistical rigor

We have conducted experiments across three random seeds on the GLUE benchmark for both HeBiRA and several competitive baselines (including LoRA, PiSSA, DoRA, and AdaLoRA). The results, reported as mean ± standard deviation, are summarized in the Table 11. As shown, HeBiRA consistently maintains superior performance.

### C.2 Performance under different rank budgets

As shown in Table 12, we evaluate the fine-tuning performance of our method across various rank budgets on some datasets. The results demonstrate that our approach consistently achieves performance improvements under different rank configurations, highlighting its effectiveness and robustness.

### C.3 Analysis of update interval

Under a fixed number of rank adjustments, we evaluate the impact of different update intervals in Table 15. A small interval may not give newly expanded ranks enough time to learn effectively, whereas a large interval can limit the model's final refinement.

### C.4 Importance estimation comparison

We conduct a controlled study comparing the importance scoring mechanisms of AdaLoRA (Zhang et al., 2023b) and the Hebbian-based approach proposed in this work. All other components are kept consistent across methods to ensure a fair comparison. The results on GSM8K, QNLI, and QQP are summarized in Table 16. For GSM8K, the outer values correspond to LLaMA2-7B and the parenthesized values to LLaMA3-8B. In both settings,

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 89.97±0.49 | 95.10±0.11 | 69.66±0.23 | 91.96±0.03 | 94.02±0.19 | 85.80±0.45 | 89.87±0.12 | 91.67±0.06 | 88.51 |
| PiSSA | 90.39±0.13 | 95.41±0.09 | 71.33±0.09 | 91.79±0.11 | 94.13±0.05 | 88.09±0.30 | 90.44±0.53 | 91.57±0.08 | 89.14 |
| DoRA | 90.45±0.14 | 95.78±0.42 | 70.21±0.03 | 91.76±0.35 | 94.19±0.14 | 87.08±0.75 | 90.33±0.43 | 91.43±0.08 | 88.90 |
| AdaLoRA | 90.63±0.11 | 95.77±0.24 | 70.69±1.18 | 92.12±0.08 | 94.45±0.08 | 87.64±0.47 | 90.80±0.09 | 91.80±0.03 | 89.24 |
| HeBiRA | 90.57±0.03 | 96.06±0.15 | 72.29±0.27 | 92.32±0.06 | 94.53±0.11 | 88.57±0.45 | 91.67±0.20 | 92.10±0.06 | **89.76** |

Table 11: Statistical results on the GLUE benchmark over three random seeds (mean ± standard deviation).

| Rank | Method | CoLA | STS-B | RTE |
|---|---|---|---|---|
| r=2 | LoRA | 68.71 | 91.68 | 85.56 |
| | HeBiRA | **71.71** | **91.83** | **87.73** |
| r=8 | LoRA | 69.82 | 91.60 | 85.20 |
| | HeBiRA | **72.44** | **92.08** | **88.45** |
| r=16 | LoRA | 70.49 | 91.34 | 87.00 |
| | HeBiRA | **72.83** | **92.10** | **88.09** |
| r=32 | LoRA | 70.90 | 91.66 | 87.73 |
| | HeBiRA | **73.05** | **92.27** | **89.17** |

Table 12: Performance comparison between LoRA and HeBiRA under different rank budgets.

the Hebbian scoring achieves slightly better performance than Adalora, indicating the usefulness of the proposed importance measure.

### C.5 Spearman correlation analysis

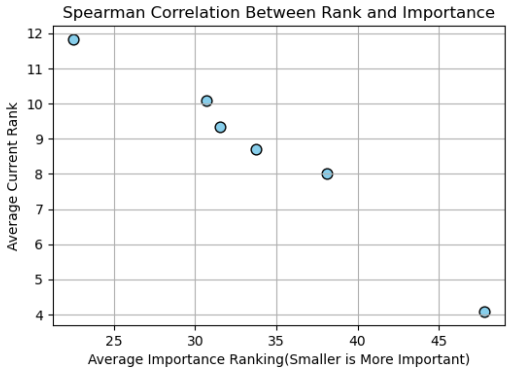



Figure 8: Spearman correlation between mean allocated rank and mean importance ranking.

In Figure 8, we analyze the correlation between layer importance and assigned rank across all modules within `layer5`. The strong spearman correlation confirms that dynamic rank assignment aligns well with learned importance.

### C.6 Low-rank comparison (r=2)

We additionally evaluate method under a strict low-rank setting of r=2. Across tasks, our method outperforms competing approaches, demonstrating its robustness under constrained rank budgets. The results are shown in Table 13.

### C.7 Matching final rank

To complement the bidirectional adjustment ablation in the main text, where all methods use the same initial rank, we further conduct an experiment in which methods start with different initial ranks but are required to reach the same total rank after adjustment. As shown in Table 14, HeBiRA achieves the best performance, indicating that coordinated pruning and expansion more effectively utilizes the final rank budget than single-direction strategies.

### C.8 Comparison of training cost

HeBiRA matches AdaLoRA in both memory usage and total training time across RTE and SST-2 in Table 17, with only negligible differences. This demonstrates that our bidirectional rank adjustment method introduces virtually no additional training cost while providing its performance benefits.

### C.9 Statistical robustness of layer-wise allocation

To quantify each layer's requirement for additional capacity, we define the layer-level importance score as the minimum importance among its remaining rank directions after the pruning phase is completed: $S^{(\ell)}_{\text{layer}} = \min_{G_i \in \mathcal{G}^{(\ell)}} S_i$. This design is rooted in identifying the "representation bottleneck" of the model. According to the "Buckets Effect," a layer's need for expansion is best reflected by its "weakest surviving direction". A high $S^{(\ell)}_{\text{layer}}$ indicates that even the least significant rank direction in layer $\ell$ still contributes substantially to the model's performance, suggesting that the layer has exhausted its current capacity. Conversely, aggregators like the mean or higher quantiles can be skewed by a few highly dominant directions, potentially masking the insufficiency of the underlying rank space.

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 90.30 | 94.95 | 68.71 | 91.61 | 94.03 | 85.56 | 89.71 | 91.68 | 88.32 |
| DoRA | 90.33 | 96.10 | 70.25 | 91.97 | 94.23 | 88.09 | 90.20 | 91.80 | 89.12 |
| AdaLoRA | 90.66 | 95.80 | 70.04 | 91.48 | 94.49 | 87.36 | 90.44 | 91.63 | 88.99 |
| HeBiRA | 90.48 | 96.10 | 71.71 | 92.07 | 94.33 | 87.73 | 90.20 | 91.83 | **89.31** |

Table 13: Performance comparison with different methods (r=2).

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 90.65 | 94.95 | 69.82 | 91.99 | 93.87 | 85.20 | 89.95 | 91.60 | 88.50 |
| +Prune | 90.61 | 96.33 | 70.61 | 92.09 | 94.44 | 88.81 | 91.18 | 91.96 | 89.50 |
| +Expand | 90.50 | 96.22 | 71.12 | 92.11 | 94.44 | 87.73 | 90.69 | 91.73 | 89.32 |
| HeBiRA | 90.59 | 96.10 | 72.44 | 92.38 | 94.65 | 88.45 | 91.67 | 92.08 | **89.80** |

Table 14: Performance comparison with different strategies under same final rank.

| **Dataset** | **Update interval** | | |
|---|---|---|---|
| MRPC | T=100 | T=150 | T=200 |
| | 91.18 | 91.67 | 90.69 |
| STS-B | T=100 | T=200 | T=300 |
| | 91.94 | 92.08 | 91.89 |

Table 15: Ablation of update interval.

| **Method** | **GSM8K** | **QNLI** | **QQP** |
|---|---|---|---|
| AdaLoRA | 56.3 (80.82) | 94.58 | 92.03 |
| Hebbian | **57.5 (81.1)** | **94.65** | **92.38** |

Table 16: Comparison of AdaLoRA and Hebbian importance estimation under identical settings.

To empirically validate this strategy, we compared the Minimum aggregator against Mean aggregator on the GLUE benchmark. As shown in the Table 18, the Minimum aggregator achieves superior performance.

# D Experimental Settings

For all baseline methods, we follow the hyperparameter configurations reported in their original papers whenever available; otherwise, we adopt the same settings as used in our method for a fair comparison. All experiments are conducted on a single NVIDIA A6000 GPU (48GB memory).

## D.1 Training details on GLUE benchmark

In the GLUE Benchmark, the model we used was DebertaV3-Base, with a rank size of 8. The specific details of the experimental hyperparameters are shown in Table 19.

| Dataset | Method | GPU | Time |
|---|---|---|---|
| RTE | AdaLoRA | 14.45GB | 1372s |
| | HeBiRA | 14.17GB | 1379s |
| SST-2 | AdaLoRA | 6.95GB | 7679.28s |
| | HeBiRA | 6.91GB | 7828.56s |

Table 17: Comparison of practical training cost between AdaLoRA and HeBiRA.

## D.2 Training details on mathematical reasoning and code generation

In the mathematical reasoning and code generation task, the initial rank size is 4. For LLaMA2-7B we set the learning rate to $2 \times 10^{-4}$, batch size to 16, and train for 5 epochs. Ranks are held fixed for the initial $t_i = 2000$ steps to stabilize early representations. Rank adaptation is then performed every $\Delta_T = 1000$ steps, with $k = 12$ ranks adjusted at each update. After the adaptation phase completes, ranks are frozen and the model undergoes a final warmup fine-tuning of length $t_f$ ($t_f = 11250$ for MetaMath, $t_f = 12765$ for Python). For LLaMA3-8B, the schedule and timings are identical while the learning rate is $1 \times 10^{-4}$.

# E Effective Rank Formulation

The effective rank was introduced by Roy and Vetterli (2007) as a stable surrogate for matrix rank based on the Shannon entropy of its singular value distribution. It serves as a capacity-aware metric that reflects how many singular directions are meaningfully utilized, rather than merely present.

Formally, it is defined as the exponential of the Shannon entropy of the normalized singular value

| Method | MNLI | SST-2 | CoLA | QQP | QNLI | RTE | MRPC | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| Mean | 90.38 | 95.98 | 71.43 | 92.14 | 94.53 | 88.09 | 91.18 | 91.98 | 89.46 |
| Minimum(Ours) | 90.59 | 96.10 | 72.44 | 92.38 | 94.65 | 88.45 | 91.67 | 92.08 | **89.80** |

Table 18: Performance comparison with different methods (r=2).

| Dataset | metric | learning rate | batch size | # epochs | $t_i$ | $\Delta_T$ | $t_f$ | $k$ |
|---|---|---|---|---|---|---|---|---|
| MNLI | Accuracy | $5 \times 10^{-4}$ | 32 | 7 | 3000 | 1000 | 65000 | 12 |
| RTE | Accuracy | $1.2 \times 10^{-3}$ | 32 | 50 | 300 | 100 | 2600 | 12 |
| QNLI | Accuracy | $9 \times 10^{-4}$ | 32 | 5 | 1000 | 500 | 10000 | 12 |
| MRPC | Accuracy | $1 \times 10^{-3}$ | 32 | 30 | 600 | 150 | 1100 | 12 |
| QQP | Accuracy | $6 \times 10^{-4}$ | 32 | 9 | 5000 | 1000 | 80000 | 12 |
| SST-2 | Accuracy | $8 \times 10^{-4}$ | 32 | 24 | 1000 | 1000 | 25000 | 12 |
| CoLA | Matthews corr | $1 \times 10^{-3}$ | 32 | 35 | 700 | 100 | 7000 | 12 |
| STS-B | Pearson corr | $2.2 \times 10^{-3}$ | 32 | 25 | 800 | 200 | 1500 | 12 |

Table 19: Hyper-parameter setup of HeBiRA for GLUE benchmark.

distribution computed on a matrix $M$, expressed:

$$\text{erank}(M) = \exp\left(-\sum_{i=1}^{n} p_i \log p_i\right), \qquad (21)$$

where $p_i = \frac{\sigma_i}{\sum_j \sigma_j}$ and $\sigma_i$ are the singular values of $M$.

## F The Resulting Rank Distribution

As shown in Figure 9, methods were applied to STS-B using the DeBERTaV3-base model and the respective final rank distributions were saved. Under limited resource budgets, the proposed method offers a higher upper bound on rank capacity.

## G Dataset Statistics

**GLUE** (Wang et al., 2019) benchmark consists of a broad set of natural language understanding tasks, created to assess model performance over varied linguistic challenges. The statistics of these datasets are provided in Table 20.

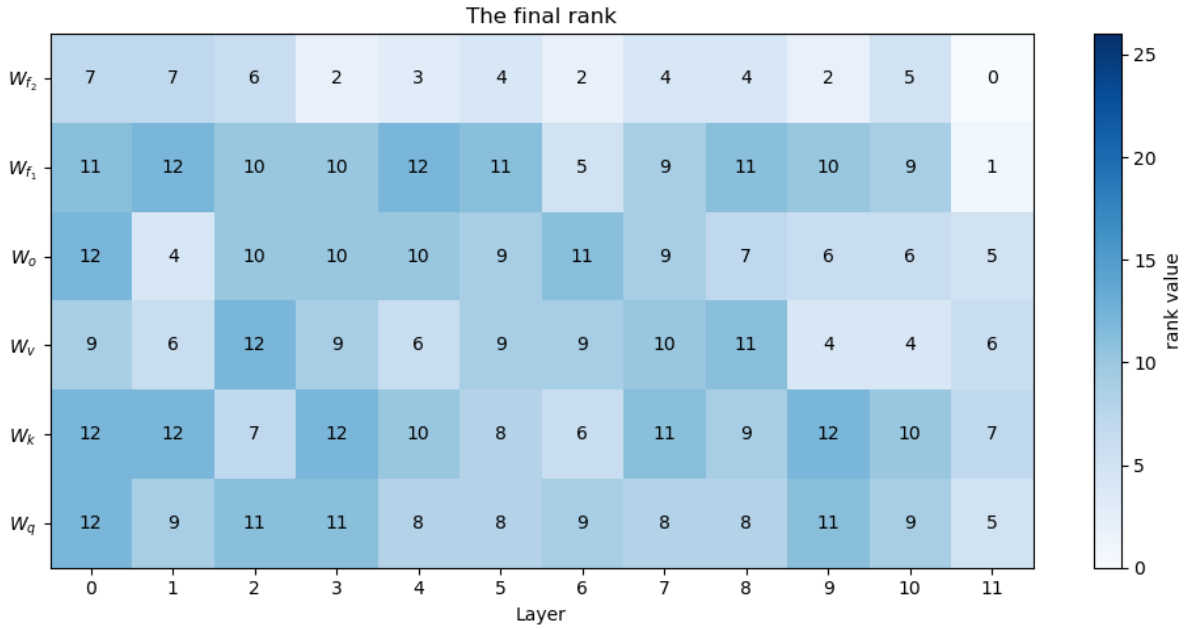
| **Corpus** | **Train** | **Valid** | **Test** | **Metrics** |
|---|---|---|---|---|
| RTE | 2.5k | 277 | 3k | Accuracy |
| MRPC | 3.7k | 408 | 1.7k | Accuracy |
| STS-B | 5.7k | 1.5k | 1.4k | Pearson corr |
| CoLA | 8.5k | 1,043 | 1,063 | Matthews corr |
| SST-2 | 67k | 872 | 1.8k | Accuracy |
| QNLI | 105k | 5.5k | 5.5k | Accuracy |
| QQP | 364k | 40.4k | 391k | Accuracy |
| MNLI | 393k | 20k | 20k | Accuracy |

Table 20: Statistics of the GLUE Benchmark Datasets

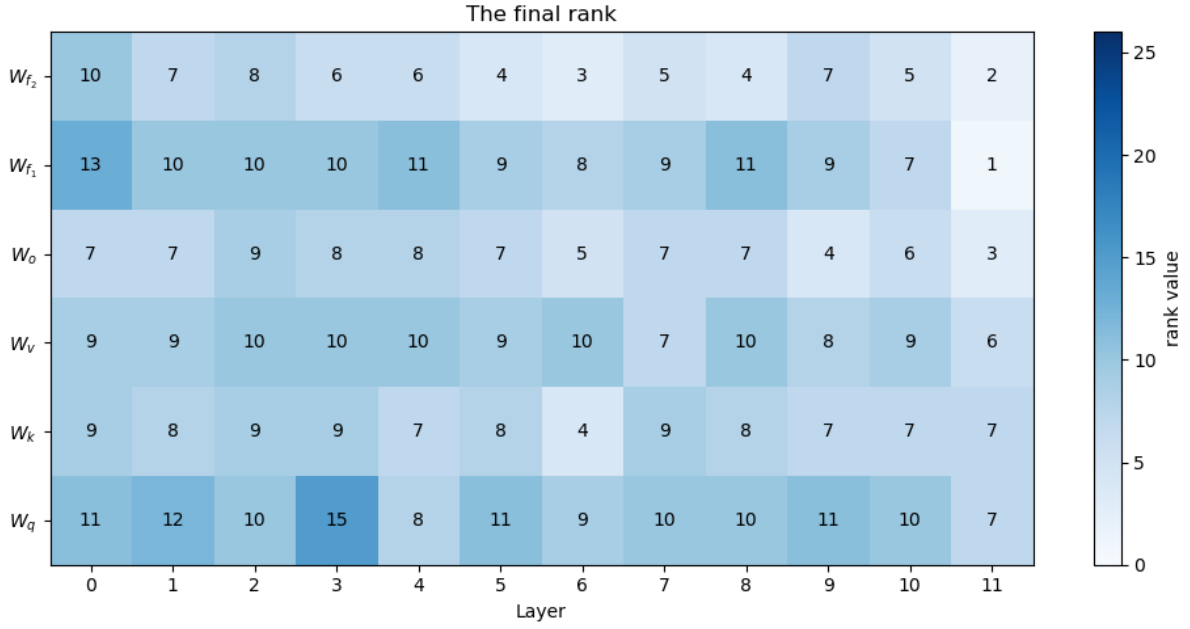
**XSum** (Narayan et al., 2018) is a dataset tailored for single-sentence news summarization, where the goal is to condense an article into a brief abstract. It contains roughly 227K news articles sourced from the BBC, each paired with a professionally authored one-sentence summary. The dataset is considered highly challenging because the summaries are extremely short and demand genuine abstraction rather than surface-level extraction. In contrast to many summarization benchmarks that emphasize selecting salient portions from the text, XSum requires models to produce concise, informative, and fluent summaries that capture the main idea of the article, often involving synthesis and substantial rephrasing.

**HumanEval** (Chen et al., 2021) is composed of 164 manually authored programming problems released by OpenAI. Each problem provides a function signature, a natural-language description, example usages, a partially completed function body, and a set of unit tests (averaging 7.7 per task).

**MBPP** (Austin et al., 2021) contains 974 Python programming exercises that have been manually validated. The benchmark spans basic programming concepts, standard library operations, and related skills. Every task includes a natural-language prompt, a reference code solution, and three automatically executable test cases.




(a) The resulting rank of AdaLoRA




(b) The resulting rank of HeBiRA

Figure 9: Comparative rank allocation patterns across model layers.

## H The Algorithm of Bidirectional Rank Adjustment

Algorithm 1 outlines the proposed bidirectional rank adjustment procedure.

**Algorithm 1** Bidirectional rank adjustment with hebbian-inspired importance

**Input:** LoRA layers with rank-level triples $\{G_i = (\mathbf{a}_i, e_i, \mathbf{b}_i)\}$, total rank budget $r$, adjustment size $k$
**Output:** Updated low-rank representation $\Delta W_{\text{adjusted}}$

**Step 1: Compute hebbian-inspired importance scores**
**for** each rank-level triple $G_i = (\mathbf{a}_i, e_i, \mathbf{b}_i)$ **do**
    Compute importance score:
        $S_i = |e_i| \cdot \left\|\mathbf{X}\mathbf{a}_i^\top\right\|_2 \cdot \|\nabla_{\mathbf{b}_i}\mathcal{L}\|_2$
**end for**

**Step 2: Pruning phase**
Aggregate all importance scores into global candidate pool
Sort triples by $S_i$ in ascending order
Select $k$ triples with lowest scores for pruning:
    $\mathcal{P} = \{G_{(1)}, G_{(2)}, \ldots, G_{(k)}\}$
Remove pruned triples:
    $\Delta W_{\text{pruned}} = \sum_{G_i \notin \mathcal{P}} G_i$

**Step 3: Expansion phase**
**for** each LoRA layer $\ell$ **do**
    Compute layer-level importance score:
        $S_{\text{layer}}^{(\ell)} = \min_{G_i \in \mathcal{G}^{(\ell)}} S_i$
**end for**
Sort layers by $S_{\text{layer}}^{(\ell)}$ in descending order
Select top-$k$ layers for expansion:
    $\mathcal{E} = \{\ell_{(1)}, \ell_{(2)}, \ldots, \ell_{(k)}\}$
Add one new rank direction to each layer in $\mathcal{E}$:
    $\Delta W_{\text{adjusted}} = \Delta W_{\text{pruned}} + \sum_{\ell \in \mathcal{E}} G_\ell^{\text{new}}$

**return** $\Delta W_{\text{adjusted}}$