Supplementary Materials for *AIRA: Activation-Informed Low-Rank Adaptation for Large Models*

Lujun Li[1] Dezhi Li[2] Cheng Lin[3] Wei Li[4] Wei Xue[1]
Sirui Han[1*] Yike Guo[1*]
[1] Hong Kong University of Science and Technology, [2] Southeast University
[3] University of Electronic Science and Technology of China, [4] University of Birmingham
[1]lilujunai@gmail.com, {siruihan,yikeguo}@ust.hk, [2]lidezhi@seu.edu.cn,[3]linchengtech@gmail.com

Our appendix provides supplementary information to the main paper, offering in-depth insights into our experimental procedures, extended discussions, and detailed setup configurations.

A. Theoretical Analysis on Outlier-Driven Rank Assignment

Lemma A.1 (Outlier-Rank Correlation)**.** *For a layer l, the required rank r_l^* to achieve approximation error ϵ satisfies:*

$$r_l^* \geq C \cdot \mathcal{O}_l \cdot \log\left(\frac{1}{\epsilon}\right), \tag{1}$$

where $C > 0$ is a constant dependent on activation statistics. This indicates that outlier-rich layers require higher ranks for equivalent performance.

Proof. Using the SVD of the layer weight matrix W_l, the approximation error ϵ is bounded by the tail singular values. Outlier activations amplify the spectral decay of W_l, requiring more singular values (higher r_l) to achieve the same ϵ. The logarithmic relationship follows from the Eckart-Young theorem. □

Theorem A.2 (Optimality of OD-Rank)**.** *The solution $\mathbf{r}^*$ to the OD-Rank optimization problem maximizes the cumulative expressiveness:*

$$\sum_{l=1}^{L} \mathcal{O}_l \cdot \log(1 + r_l^*) \geq \sum_{l=1}^{L} \mathcal{O}_l \cdot \log(1 + r_l^{\textit{uniform}}), \tag{2}$$

where $r_l^{\textit{uniform}} = P_{\textit{budget}} / \sum_{l=1}^{L}(m_l + n_l)$ is the uniform rank allocation.

Proof. By the KKT conditions, the optimal $\mathbf{r}^*$ allocates higher ranks to layers with larger $\mathcal{O}_l$, maximizing the weighted objective. Uniform allocation r_l^{uniform} neglects $\mathcal{O}_l$, yielding suboptimal expressiveness. □

OD-Rank thus ensures that parameter budgets are concentrated where they maximally impact performance, achieving superior results compared to uniform allocation.

B. Theoretical Analysis on Activation-Informed Training

Proposition B.1 (Gradient Scaling Effect)**.** *The gradient of the loss L with respect to the adapter A satisfies:*

$$\nabla_A L \propto \mathbf{s}_{\textit{in}} \odot \nabla_{\mathbf{y}} L, \tag{3}$$

indicating that gradients are amplified for dimensions with significant activations.

Proof. By the chain rule:

$$\nabla_A L = \frac{\partial \mathbf{y}}{\partial A} \cdot \nabla_{\mathbf{y}} L \tag{4}$$

$$= (\mathbf{x} \odot \mathbf{s}_{\text{in}})^\top \cdot \nabla_{\mathbf{y}} L \cdot \alpha \tag{5}$$

$$\propto \mathbf{s}_{\text{in}} \odot \nabla_{\mathbf{y}} L. \tag{6}$$

This scaling ensures adaptation focuses on high-impact input dimensions. □

Lemma B.2 (Convergence Acceleration)**.** *The normalized update rule reduces gradient variance by a factor of $\|\mathbf{s}_{\textit{in}}\|_2^2$, accelerating convergence compared to unnormalized LoRA.*

Proof. The variance reduction follows from the scaling property of the normalization vector $\mathbf{s}_{\text{in}}$, which downweights low-magnitude activations that contribute less to the loss. □

This mechanism ensures adaptation prioritizes critical features, improving both efficiency and final performance.

C. Rank Allocation Methods Comparison

In Table 5 of the main paper, we compare various rank allocation strategies. Here, we provide detailed definitions of each method and present a comprehensive visualization of their resulting rank distributions across layers.

*Corresponding author.

- **Uniform**: The conventional approach that assigns equal ranks to all layers, disregarding their functional heterogeneity. Specifically, we set $r_l = r$ for all layers l, where r is determined by the total parameter budget.
- **Random**: This method randomly assigns ranks to different layers while respecting the minimum and maximum rank constraints. The ranks are sampled from a uniform distribution and then scaled to meet the parameter budget constraint.
- **Decay**: This approach allocates ranks in a linearly decreasing manner from earlier to later layers, following the intuition that earlier layers in model often capture more general features. Specifically, $r_l = r_{\max} - (l-1) \cdot \frac{r_{\max} - r_{\min}}{L-1}$, where L is the total number of layers.
- **Growth**: This strategy allocates ranks in a linearly increasing manner from earlier to later layers, following the alternative hypothesis that later layers may require more adaptation capacity for task-specific features. Specifically, $r_l = r_{\min} + (l-1) \cdot \frac{r_{\max} - r_{\min}}{L-1}$.
- **O-Driven (Ours)**: Our proposed Outlier-Driven Rank Assignment method that allocates ranks based on the layer-specific Outlier Ratio metric. This approach solves a constrained optimization problem to maximize the adaptation capacity for layers with significant activation outliers, as described in Section 3.2 of the main paper.

Figure 1 visualizes the rank distributions resulting from these different allocation strategies across layers. The visualization clearly demonstrates how our O-Driven method assigns ranks in a non-uniform pattern that correlates with the distribution of activation outliers, allocating higher ranks to layers with more significant outlier ratios. This targeted allocation leads to more efficient parameter utilization compared to the other strategies, as evidenced by the performance improvements reported in Table 5 of the main paper.

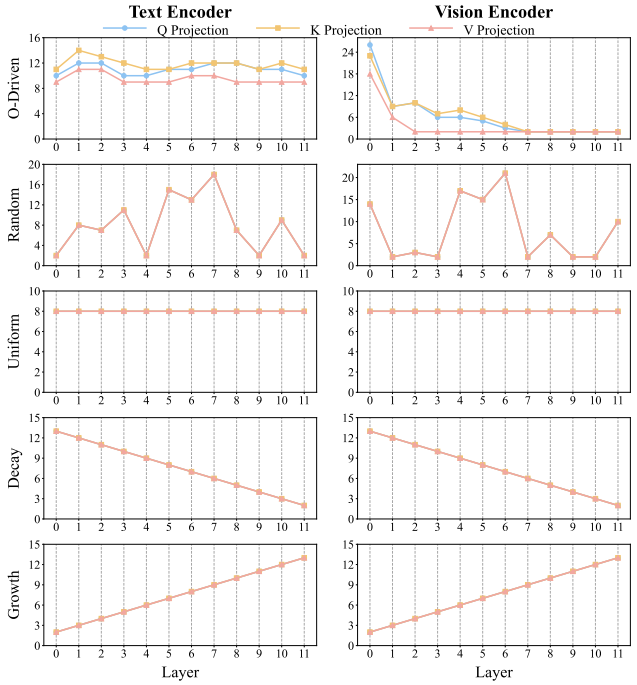

Figure 1. Comparison of rank distributions across layers for different allocation strategies: Uniform, Random, Decay, Growth, and our Outlier-Driven (O-Driven) method. The visualization shows how our O-Driven method assigns ranks based on activation outlier patterns, resulting in a non-uniform distribution that prioritizes layers with significant outlier ratios.

D. Impact of Calibration Data on Performance

As mentioned in the limitations section of the main paper, our method relies on input data for initialization and rank assignment. This is a common practice for efficient compression and optimization techniques that typically require calibration data, with research spanning model pruning [3, 13, 16], quantization [2, 5], specialized compression for Mixture-of-Experts models via SVD [17] and Delta Decompression [6], and knowledge distillation approaches [8, 9, 22]. A significant trend involves automating the discovery of efficient architectures and compression schemes through training-free, zero-cost proxies that eliminate the need for calibration data, including parametric proxies for NAS [4], automated proxy discovery for generative [14] and distillation-aware search [19], ViT architecture optimization [21], attention pattern discovery [15], automated student architecture search [1], Monte Carlo Tree Search for KD automation [10], universal knowledge distillers [12], and task-specific KD strategy optimization [11]. To better understand this dependency and provide insights for future improvements, we conduct additional experiments exploring how different types and quantities of calibration data affect performance.

D.0.1. Effect of Calibration Dataset Type

Table 1 presents the results of using different calibration datasets for computing activation statistics while testing on various target datasets. Interestingly, we observe that the choice of calibration dataset has a relatively minor impact on the final performance across different test datasets. For instance, when using DTD as the calibration dataset, we achieve 64.83% accuracy on DTD test data, which is very close to the 64.60% achieved when using Food101 for calibration. This suggests that our method can effectively capture general activation patterns that transfer well across different visual domains.

D.0.2. Effect of Calibration Dataset Size

We also investigate how the size of the calibration dataset affects performance. Table 2 shows the results of using different batch sizes for collecting activation statistics. Surprisingly, even with a relatively small calibration batch size of 64, our method achieves comparable or even slightly

Table 1. Effect of different calibration datasets with the ViT-B/16 as visual backbone. Top-1 accuracy (%) averaged over 3 random seeds is reported.

Calibration dataset	Test dataset	Test Accuracy
dtd	ucf101	78.64
dtd	stanford_cars	73.27
dtd	dtd	64.83
dtd	oxford_pets	93.73
dtd	food101	86.42
food101	food101	86.42
food101	dtd	64.60
food101	stanford_cars	73.52
food101	oxford_pets	93.76
food101	ucf101	78.98
oxford_pets	stanford_cars	73.50
oxford_pets	dtd	64.83
oxford_pets	food101	86.43
oxford_pets	oxford_pets	93.87
oxford_pets	ucf101	79.04

better performance than with larger batch sizes across all test datasets. This suggests that AIRA can effectively capture the essential activation patterns with a modest amount of calibration data, which is particularly advantageous for resource-constrained scenarios.

Table 2. Effect of different calibration dataset sizes with the ViT-B/16 as visual backbone. Top-1 accuracy (%) averaged over 3 random seeds is reported.

cal_batch_size	oxford_pets	food101	stanford_cars	dtd	ucf101
256	93.87	86.42	73.37	64.66	78.77
128	93.98	86.42	73.49	64.72	78.88
64	94.03	86.42	73.54	64.72	78.91

These findings have important implications for future research directions. First, they suggest that AIRA's performance is robust to the choice of calibration data, making it more practical for real-world applications where domain-specific calibration data might not be readily available. Second, the effectiveness of smaller calibration batch sizes indicates potential for further efficiency improvements. Future work could explore adaptive calibration strategies that dynamically adjust the amount and type of calibration data based on the target task, or investigate meta-learning approaches to predict optimal rank allocations without explicit calibration. Additionally, developing techniques to update activation statistics during fine-tuning could further enhance adaptation to domain shifts and task-specific requirements.

E. Experimental Setup and Hyperparameters

E.1. Model Configurations

- CLIP ViT-B/16 vision encoder: 86.19 Million parameters, 12 layers, 768 hidden size
- CLIP ViT-B/16 text encoder: 63.43 Million parameters, 12 layers, 512 hidden size
- Mistral-7B: 7 billion parameters, 32 layers, 4096 hidden size

E.2. Hardware and Software

- GPUs: 8 x NVIDIA V100S (32GB)
- Framework: PyTorch 2.2.1
- CUDA Version: 11.3

E.3. Hyperparameters

Instruction Tuning: We perform the instruction tuning experiments on Mistral-7B-v0.1 [7], Gemma-7B [20] and LlaMA-3 8B models. We use a batch size of 128 and train for 2 epochs on 100k samples of the MetaMathQA dataset. Models are evaluated on the GSM8K and MATH datasets. The learning rate is set to 7E-3 with the AdamW optimizer [18]. The warmup ratio is 0.02, and a cosine learning rate scheduler is used. The parameter α for AIRA modules is always equal to the rank. In our experiments, AIRA uses 166M parameters for Mistral-7B. We use 8 $\times$ V100S 32GB GPUs for the finetuning.

Fine-tuning of Vision-Language Models: Table 3 details our hyperparameter settings for CLIP ViT-B/16, which remain consistent across all 5 datasets.

Common hyperparameters across experiments:

- Batch size: 32
- Learning rate: 1e-4 (AdamW optimizer)
- Weight decay: 0.01
- Warmup steps: 500
- Max steps: 20,000

Table 3. Our hyperparameter configuration on fine-tuning of Vision-Language model experiments.

Hyperparameters	**AIRA**
Batch size	64
Learning rate	5e-4
Scheduler	CosineAnnealingLR
Optimizer	AdamW
Weight decay	0.05
Dropout rate	0.25
Placement	query, key, value
Encoder	TextEncoder&VisionEncoder
Lora alpha	1
n_iters	300
Max rank	32
Min rank	2
Rank budget	0.7M
Outlier Ratio threshold (τ)	5
Objective function	$-\log(x)$

Task-specific adjustments:

- GSM8K and Math: Increased max steps to 30,000

- Few-shot CLIP: Reduced batch size to 16, max steps to 5,000

E.4. Evaluation Metrics

- NLP tasks: Accuracy, F1 score
- Math reasoning: Pass@1 score
- Few-shot image classification: Top-1 accuracy

References

[1] Peijie Dong, Lujun Li, and Zimian Wei. Diswot: Student architecture search for distillation without training. In *CVPR*, 2023. 2

[2] Peijie Dong, Lujun Li, Zimian Wei, Xin Niu, Zhiliang Tian, and Hengyue Pan. Emq: Evolving training-free proxies for automated mixed precision quantization. *arXiv preprint arXiv:2307.10554*, 2023. 2

[3] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *ICML*, 2024. 2

[4] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Zimian Wei, Qiang Wang, and Xiaowen Chu. Parzc: Parametric zero-cost proxies for efficient nas. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16327–16335, 2025. 2

[5] Peijie Dong, Lujun Li, Yuedong Zhong, Dayou Du, Ruibo Fan, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Yike Guo, et al. Stbllm: Breaking the 1-bit barrier with structured binary llms. In *ICLR*, 2025. 2

[6] Hao Gu, Wei Li, Lujun Li, Zhu Qiyuan, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for moe-based llms compression. *arXiv preprint arXiv:2502.17298*, 2025. 2

[7] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 3

[8] Lujun Li. Self-regulated feature learning via teacher-free feature distillation. In *ECCV*, 2022. 2

[9] Lujun Li and Zhe Jin. Shadow knowledge distillation: Bridging offline and online knowledge transfer. In *NeuIPS*, 2022. 2

[10] Lujun Li, Peijie Dong, Zimian Wei, and Ya Yang. Automated knowledge distillation via monte carlo tree search. In *ICCV*, 2023. 2

[11] Lujun Li, Yufan Bao, Peijie Dong, Chuanguang Yang, Anggeng Li, Wenhan Luo, Qifeng Liu, Wei Xue, and Yike Guo. Detkds: Knowledge distillation search for object detectors. In *ICML*, 2024. 2

[12] Lujun Li, Peijie Dong, Anggeng Li, Zimian Wei, and Ya Yang. Kd-zero: Evolving knowledge distiller for any teacher-student pairs. *NeuIPS*, 2024. 2

[13] Lujun Li, Peijie, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. Discovering sparsity allocation for layer-wise pruning of large language models. In *NeuIPS*, 2024. 2

[14] Lujun Li, Haosen Sun, Shiwen Li, Peijie Dong, Wenhan Luo, Wei Xue, Qifeng Liu, and Yike Guo. Auto-gas: Automated proxy discovery for training-free generative architecture search. ECCV, 2024. 2

[15] Lujun Li, Zimian Wei, Peijie Dong, Wenhan Luo, Wei Xue, Qifeng Liu, and Yike Guo. Attnzero: efficient attention discovery for vision transformers. In *ECCV*, 2024. 2

[16] Wei Li, Lujun Li, Mark Lee, and Shengjie Sun. Als: Adaptive layer sparsity for large language models via activation correlation assessment. In *NeuIPS*, 2024. 2

[17] Wei Li, Lujun Li, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Structured mixture-of-experts LLMs compression via singular value decomposition. In *ICML*, 2025. 2

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[19] Haosen Sun, Lujun Li, Peijie Dong, Zimian Wei, and Shitong Shao. Auto-das: Automated proxy discovery for training-free distillation-aware architecture search. ECCV, 2024. 2

[20] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 3

[21] Zimian Wei, Peijie Dong, Zheng Hui, Anggeng Li, Lujun Li, Menglong Lu, Hengyue Pan, and Dongsheng Li. Autoprox: Training-free vision transformer architecture search via automatic proxy discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. 2

[22] Liu Xiaolong, Li Lujun, Li Chao, and Anbang Yao. Norm: Knowledge distillation via n-to-one representation matching. In *ICLR*, 2023. 2