Transactions on Computational and Scientific Methods | Vo. 4, No. 6, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Adaptive Fusion of Low-Rank and Soft-Gated Sparse Updates for Parameter-Efficient Instruction Tuning

Bingxu Chen
University of California, San Diego, San Diego, USA
vincechen0116@gmail.com

Abstract: This paper addresses the high cost and management challenges of fine-tuning large-scale pre-trained models in instruction-based scenarios, proposing a parameter-efficient fine-tuning algorithm framework that integrates low-rank adaptation and sparse updates. Under the premise of freezing basic parameters, linear layer weight updates are represented as structured incremental stacking. The low-rank component characterizes the main update direction in a two-factor manner, providing compressible global expressiveness; the sparse component generates soft masks through continuous gating and selects free increments element-wise, achieving selective reshaping of key connections. Furthermore, an intra-layer fusion coefficient is introduced to coordinate the contribution ratio of the two types of increments, and the update complexity is explicitly incorporated into the optimization objective through low-rank norm constraints and gating density regularization, thus balancing update stability and expressive flexibility under a unified training objective. For data processing, an open-source instruction-based fine-tuning dataset is used, and a standard input-output pairing and label masking mechanism is constructed to ensure that the training signal focuses on the response generation segment. Comparative experiments evaluate the proposed method and representative parameter-efficient fine-tuning strategies from dimensions such as language modeling reliability, semantic consistency, and answer matching, verifying that this integrated structural update has superior comprehensive performance across multiple metrics.

Keywords: Efficient parameter fine-tuning; structured incremental adjustments; soft gating mask; instruction alignment.

1. Introduction

Large-scale pre-trained models demonstrate strong generalization capabilities in tasks such as natural language processing, computer vision, and cross-modal understanding. However, their deployment in specific domains and tasks still heavily relies on fine-tuning. While traditional full-parameter fine-tuning can fully unleash the model's representational power, training and deployment costs increase linearly with parameter size, placing higher demands on GPU memory, computing power, time, and energy consumption, while also imposing a greater burden on storage and version management. In application environments with multiple scenarios and parallel iterations across multiple tasks, significantly reducing fine-tuning costs while ensuring model adaptability becomes a key issue in improving the usability and scalability of large-scale models[1].

Efficient parameter fine-tuning reduces computational and storage overhead by restricting the subspace of trainable parameters. Low-rank adaptation utilizes low-rank structures to characterize the main direction of parameter updates, achieving rapid adaptation to downstream tasks with small parameter increments, offering good portability and modularity. However, low-rank constraints may also exhibit limited expression in

complex distribution shifts or fine-grained capability transfer, especially in situations requiring local feature reshaping or long-tail pattern capture[2,3]. A single low-rank update cannot simultaneously meet the requirements of flexibility and robustness. Therefore, it is necessary to introduce a more selective update mechanism while maintaining the advantages of low-rank structures to improve the adaptability to key parameters and critical paths[4].

Sparse updates offer an alternative approach to this contradiction: by adjusting only a small number of key parameters through sparsification strategies, the specificity and interpretability of updates can be improved without significantly increasing training costs[5]. Sparse updates can form structured selection preferences in the parameter space, which is beneficial for controlling the update magnitude and avoiding perturbations of irrelevant parameters, thereby mitigating risks such as overfitting and catastrophic forgetting. However, relying solely on sparse updates may also face problems such as optimization instability, limited gradient propagation, and sensitivity to sparse structure design, leading to significant fluctuations in adaptation performance across different tasks and data distributions. Thus, the global representational advantages of low-rank structures and the local selectivity of sparse updates are naturally complementary[6].

Research on model fine-tuning algorithms that integrate low-rank adaptation and sparse updates has significant theoretical and practical value. On the one hand, the fusion strategy is expected to simultaneously achieve low-dimensional structural constraints for parameter updates and selective reshaping of key parameters within a unified framework, thereby achieving a better trade-off between efficiency and expressive power, and providing a methodological foundation for building scalable and composable fine-tuning paradigms[7]. On the other hand, this direction can better support practical needs such as continuous iteration across multiple tasks, collaborative deployment between edge and cloud sides, and rapid model version switching, reducing the usage threshold and maintenance costs of large models in industry scenarios, and promoting the stable migration and efficient implementation of large model capabilities across multiple domains.

2. Datasets and Dataset Preprocessing

2.1 Dataset

This paper uses the Databricks Dolly 15k dataset as the fine-tuning dataset. This dataset consists of approximately 15,000 high-quality instruction and response samples, designed for supervised fine-tuning scenarios of large models. It covers common instruction types such as question answering, information extraction, classification, generation, and summarization, and can provide stable alignment signals for the model with a unified instruction format. It is suitable as a general fine-tuning corpus for evaluating the adaptability and update efficiency of efficient fine-tuning strategies with different parameters on multi-instruction tasks.

The data is released in an open-source format, allowing for use and redistribution in academic and commercial environments, facilitating reproducibility and comparative studies. The samples are organized around instruction and corresponding response fields, enabling direct use in supervising and fine-tuning data pipelines. Furthermore, it allows for algorithmic research on low-rank adaptation and sparse updates without altering the data semantics, to examine the expressive efficiency and generalization stability of parameter-constrained updates in multi-type instruction learning.

2.2 Dataset Preprocessing

The Dolly 15k raw data was uniformly parsed into a ternary structure required for instruction fine-tuning, including instruction, context, and response fields. The data was then cleaned based on field completeness, removing samples with missing responses, empty instructions, abnormal lengths, and broken formats. The text was normalized by deleting or normalizing redundant whitespace and invisible control characters, while maintaining semantically correct punctuation and capitalization. In sample construction, samples with context were concatenated into a single input sequence by combining instruction and context; samples without

context retained only the instruction as input. The output sequence was fixed as follows. The response is used to form a standard supervised fine-tuning input-output pair; then, a word segmenter consistent with the target model is used to complete the sub-word level encoding, and the input and output are truncated to the preset maximum length. Excessively long samples are truncated at the end to ensure training stability; the training objective adopts the label construction method of causal language modeling, setting the input part label as an ignored marker and only calculating the loss for the output part, to ensure that the parameter update focuses on the ability to generate instruction response; finally, random partitioning is performed without changing the task distribution to construct training and validation sets, and the random seed is fixed to ensure the reproducibility of data partitioning and preprocessing results. Figure 1 shows the comparison results of the dataset before and after preprocessing.

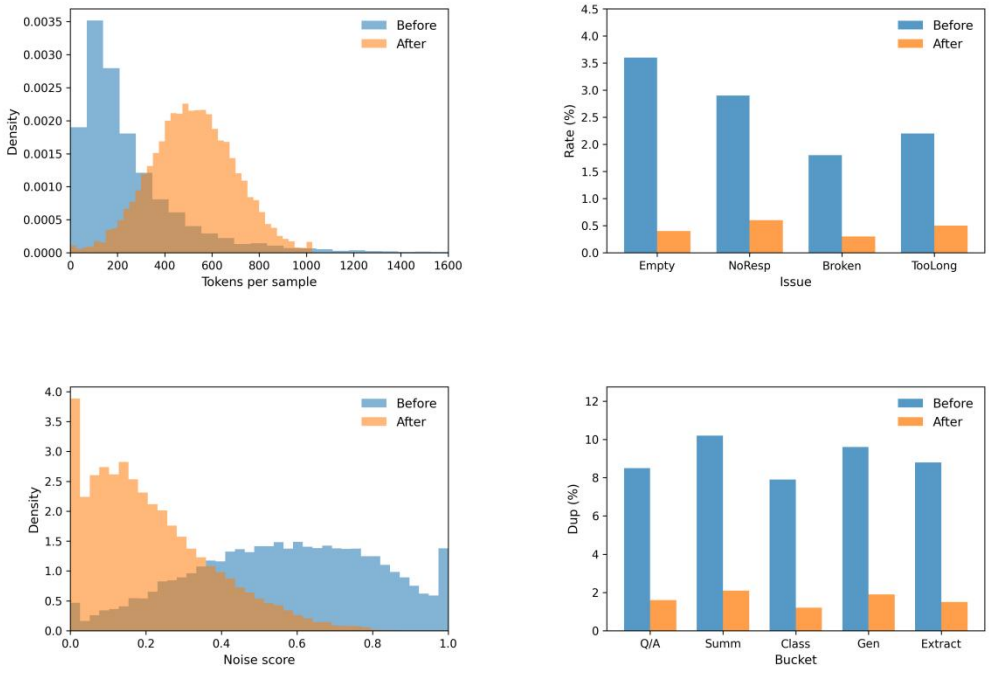

Figure 1. Comparison of dataset preprocessing results

Figure 1 compares the differences before and after data preprocessing from four dimensions: The top left subplot shows the length distribution of a single sample after word segmentation. After preprocessing, the length distribution is more concentrated, and the long tails converge significantly, indicating that truncation and removal of abnormal samples reduce the impact of extremely long samples on training stability. The top right subplot shows the proportion of invalid samples decomposed by question type, including four categories: Empty, NoResp, Broken, and TooLong. After preprocessing, the proportion of each category decreases overall, indicating that the cleaning rules effectively improve sample usability. The bottom left subplot shows the noise score distribution, used to characterize the intensity of text noise such as whitespace and control characters. After preprocessing, the scores shift to the left overall, indicating that normalization reduces non-semantic noise; The bottom right subplot shows the duplication rate (Dup) of different task buckets, used to measure the proportion of nearly duplicated samples. Bucket categories include Question Answering (QA), Summarization, Classification, Generating, and Extracting. After preprocessing, the duplication rate of each bucket decreases significantly, indicating that the deduplication strategy makes the data distribution more regular and less redundant, thereby improving the quality and consistency of the fine-tuning corpus.

3. Method

Assume the base model parameters are θ and remain frozen during the fine-tuning stage. Given an instruction input sequence x and a target output sequence y, the conditional distribution of the model under autoregressive generation is expressed as:

$$p_{\theta,\phi}(y|x) = \prod_{t=1}^{|y|} p_{\theta,\phi}\left(y_t|x, y_{<t}\right)$$

where ϕ denotes the set of incremental parameters introduced during fine-tuning. The training objective only accounts for the negative log-likelihood over the output segment and uses a mask to characterize valid labeled positions, expressed as:

$$\mathcal{L}_{\text{sft}}(\phi) = \sum_{(x,y)\in\mathcal{D}} \sum_{t=1}^{|y|} m_t \left(-\log p_{\theta,\phi}\left(y_t|x, y_{<t}\right)\right)$$

This formulation decouples the supervision signal from parameter updates, enabling the update mechanism to focus on local corrections of the generation distribution while avoiding unnecessary gradient interference on the input prompt segment. The overall architecture is presented in this article, as shown in Figure 2.

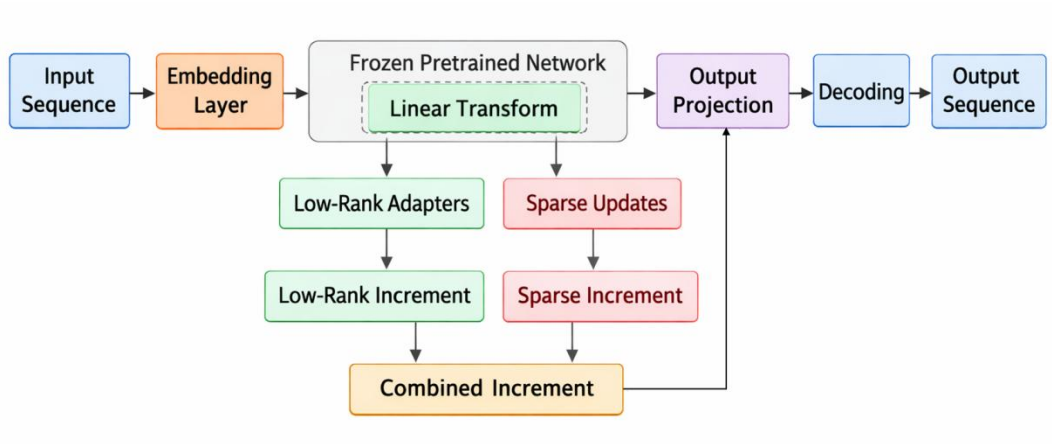

Figure 2. Overall Model Architecture

Consider an arbitrary linear transformation layer weight $W \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$. Fusion-based fine-tuning represents it equivalently as a structured increment added to the frozen weight, expressed as:

$$W' = W + \Delta W$$

To simultaneously achieve global compressibility and local selectivity, the increment is decomposed into the sum of a low-rank adaptation component and a sparse update component, expressed as:

$$\Delta W = \Delta W_{\text{lr}} + \Delta W_{\text{sp}}$$

The low-rank component adopts a rank-constrained two-factor parameterization to capture the principal directions of change, expressed as:

$$\Delta W_{\text{lr}} = BA$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$ with $r \ll \min(d_{\text{out}}, d_{\text{in}})$. This structure allows updates to be carried by a low-dimensional subspace to encode cross-dimensional coordinated shifts, while retaining sufficient degrees of freedom for subsequent sparse corrections to handle local anomalies and fine-grained requirements.

The sparse component is used to selectively reshape key connections by introducing a continuous gating matrix $G \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and defining a soft mask in a temperature-smoothed manner, expressed as:

$$M = \sigma\left(\frac{G - \tau}{T}\right)$$

where τ is the sparsity threshold, and T is the temperature coefficient. During training, M provides a differentiable sparse selection mechanism. Let the sparse free parameter be $S \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, then the sparse increment is defined as:

$$\Delta W_{\text{sp}} = M \odot S$$

Furthermore, a fusion coefficient $\alpha \in [0,1]$ is introduced to coordinate the relative contributions of the two types of updates within the same layer, expressed as:

$$\Delta W = \alpha BA + (1 - \alpha)(M \odot S)$$

This yields a complementary division of labor in the update form, where the low-rank component tends to capture macroscopic directions and shared shifts, while the sparse component tends to perform a small number of critical local patches. Both are jointly driven by the supervision signal within the same forward graph, thereby avoiding splitting expressive capacity and parameter efficiency into mutually independent training modules.

To explicitly inject structural bias into the optimization process, let $\phi = \{A, B, S, G, \alpha\}$, and impose complexity constraints on the low-rank and sparse components to control update magnitude and selection density, respectively. The low-rank component is constrained by the Frobenius norm, expressed as:

$$\mathcal{R}_{\text{lr}} = \| BA \|_F^2$$

The sparse component constrains its expected density via the l_1 norm of the soft mask, expressed as:

$$\mathcal{R}_{\text{sp}} = \| M \|_1$$

Combining the supervised fine-tuning objective with the structural regularization terms, the final optimization problem is expressed as:

$$\min_{\phi} \mathcal{L}_{\text{sft}}(\phi) + \lambda_{\text{lr}} \mathcal{R}_{\text{lr}} + \lambda_{\text{sp}} \mathcal{R}_{\text{sp}}$$

where λ_{lr} and λ_{sp} control the strengths of the low-rank and sparse constraints. This objective simultaneously shapes two inductive biases—low-dimensional structure and sparse selection—within a unified framework, enabling incremental updates to possess both compressible and reusable global directions as well as selective correction capability on key connections.

4. Experimental Results and Analysis

4.1 Experimental setup

4.2 The experimental environment of this study consists of a single-machine, single-card training node. The software stack is based on the Linux system and adopts mainstream deep learning frameworks to achieve efficient parameter fine-tuning. During training, a fixed random seed is used, and mixed precision is enabled to improve computational efficiency and memory utilization. The optimizer adopts an adaptive first-order method with weight decay, combined with linear warm-up and cosine annealing learning rate scheduling to achieve stable convergence. Batch construction is dynamically filled with sequence length as a constraint, and a maximum sequence length is set to avoid memory jitter caused by long-tailed samples. The hyperparameters related to low-rank adaptation and sparse update are kept consistent across all model layers to ensure reproducibility and comparability. The verification and preservation strategy adopts a fixed step size evaluation and retains the optimal checkpoint for subsequent analysis and reproduction. The hyperparameter settings are shown in Table 1.

Table 1: Hyperparameter settings

Category	Configuration	Value
Hardware	GPU	NVIDIA RTX 4090 24GB
Hardware	CPU	Intel Xeon 16 Cores
Hardware	Memory	64 GB
Hardware	Storage	1 TB NVMe SSD
System	Operating System	Ubuntu 22.04 LTS
Software	Python	3.10
Software	Deep Learning Framework	PyTorch 2.2
Software	CUDA	12.1
Software	cuDNN	8.9
Training	Mixed Precision	FP16
Training	Random Seed	42
Data	Max Input Length	1024
Data	Max Output Length	512
Data	Dynamic Padding	Enabled
Optimization	Optimizer	AdamW
Optimization	Learning Rate	2e-4
Optimization	Weight Decay	0.01
Optimization	Warmup Ratio	0.03
Optimization	Scheduler	Cosine
Training	Batch Size	8
Training	Gradient Accumulation	2
Training	Max Gradient Norm	1.0
Training	Epochs	3
PEFT	Low-Rank Rank	16
PEFT	Low-Rank Scaling Factor	32
PEFT	Sparsity Threshold	0.5
PEFT	Temperature Coefficient	0.2
PEFT	Fusion Coefficient	0.5
PEFT	Base-llm	Chat-GLM3-6B
Regularization	Low-Rank Regularization Coefficient	1e-4
Regularization	Sparse Regularization Coefficient	1e-5

4.2 Experimental Results and Analysis

To evaluate the adaptability and update quality of model fine-tuning algorithms that integrate low-rank adaptation and sparse updates in instruction fine-tuning scenarios, this paper selects representative methods in the field of efficient parameter fine-tuning as a baseline for comparison. Under unified data partitioning and training settings, a comprehensive comparison is conducted on dimensions such as generation quality, semantic consistency, answer matching degree, and language modeling stability. The corresponding comparison results are summarized in Table 2 in the form of multiple indicators to characterize the performance differences and update characteristics of each method on the fine-tuning task from different perspectives.

Table 2: Comparative experimental results

Method	PPL	ROUGE-L	BLEU-4	METEOR	BERTScore	Exact Match	Token-F1	Win Rate
Hu et al.[8]	25.98	71.90	37.93	76.76	55.73	39.15	60.11	47.68
Zhang et al.[9]	78.73	68.76	25.78	32.88	50.77	46.75	47.58	50.92
Ding et al.[10]	71.18	59.08	53.77	35.18	53.18	26.67	20.77	21.05
Huang et al.[11]	42.10	59.72	31.72	39.24	21.08	37.58	76.80	32.73
Zhang et al.[12]	46.73	34.79	60.70	28.93	35.88	67.15	79.43	64.28
Liang et al.[13]	28.09	45.52	20.80	70.02	60.24	63.40	51.65	24.09
Hu et al.[14]	63.12	34.99	66.97	75.77	27.89	48.61	51.69	31.83
Ours	10.11	89.34	78.38	79.45	86.64	76.71	82.67	77.13

Table 2 shows that the proposed method is more robust in terms of multidimensional generation quality and answer consistency metrics, achieving a more ideal balance between the reliability of language modeling and the usability of text generation. This result indicates that the incremental modeling approach, which integrates low-rank adaptation and sparse updates, can more fully absorb effective supervision signals from the instruction data under the constraint of limited parameter updates. This makes the output closer to the target distribution in terms of semantic fit, content coverage, and format matching, thereby improving the overall generation quality and readability.

Simultaneously, the method also maintains its advantages in matching and preference metrics, indicating that it can not only generate semantically consistent text but also more stably meet task requirements and reduce interference from irrelevant content. These phenomena indirectly verify that the global expressive power of low-rank components and the local correction ability of sparse components can complement each other, enabling more precise adjustment of key parameter paths and thus presenting more consistent high-quality output under multi-metric evaluation.

The fusion of low-rank and sparse updates introduces a selective update mechanism in the parameter space, ensuring that only a subset of connections bears the primary adaptation load. Support set visualization visually represents the update locations selected by the gating mask and their spatial structure, thus characterizing the sparsity and structure of the update. Furthermore, simultaneously displaying the magnitude contributions of the sparse support set and the low-rank component reveals the complementary division of labor between the two types of increments within the same layer.

Figure 3 illustrates the effective parameter support set structure formed by fusing low-rank and sparse updates within the same layer. The gating mask exhibits distinct connected components and locally

concentrated regions, indicating that update selection is not random but rather forms a stable, structured subset around a small number of key connections. The high-activation regions in the support set graph are consistent with the main high-weight regions of the gating mask, demonstrating that the gating mechanism can achieve explicit update location selection under differentiable constraints, thus focusing parameter modifications on connections that contribute to the task.

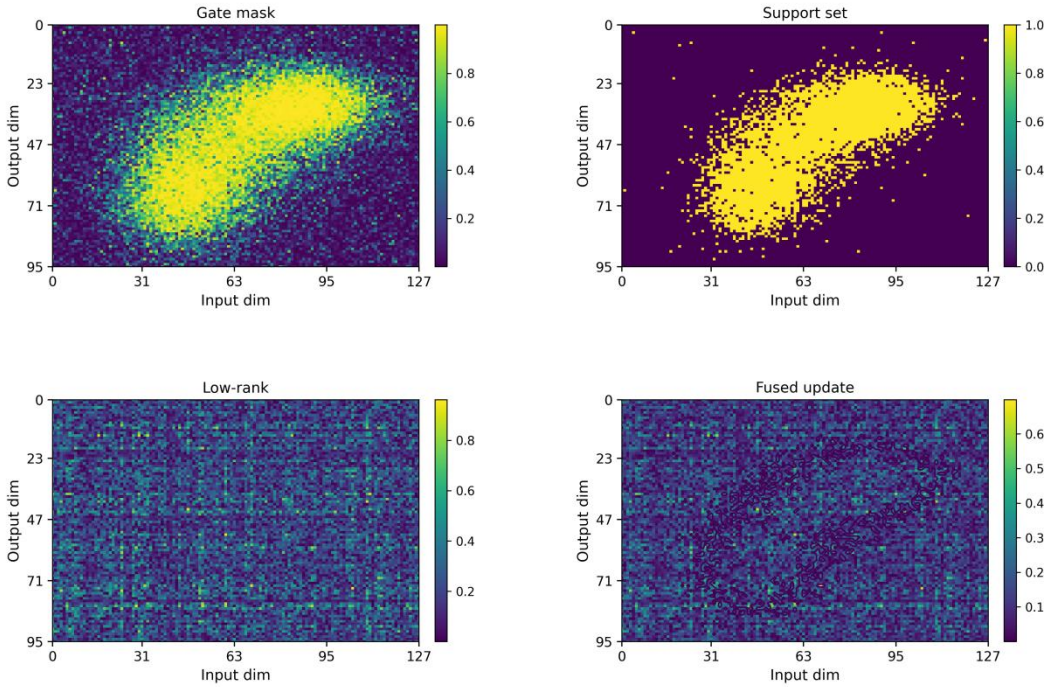

Figure 3. Visual analysis of parameter support sets updated by low-rank and sparse fusion

The low-rank component provides a continuous and smooth perturbation base globally, enabling updates to have cross-dimensional collaborative adjustment capabilities. The fused update further concentrates effective modifications into the gated regions, ensuring that the actual update retains both the compressibility of the overall expression and the ability to make pinpoint corrections to key local connections. The consistency between this structured support set and the fused increment demonstrates that the proposed method can achieve more efficient update allocation under finite incremental parameter constraints, making the update process more controllable and better aligned with the design goal of efficient parameter fine-tuning.

Low-rank adaptation compresses updates into a low-dimensional subspace through parameterization with restricted rank, thereby achieving effective regulation of model behavior with smaller increments. Singular value spectra can characterize the energy distribution pattern of low-rank increments and reflect the structural changes in the main update directions during training. Visualizing the evolution of the singular value spectrum helps to understand how the expressive load of low-rank components is redistributed among different principal directions from a linear algebra perspective; the experimental results are shown in Figure 4.

This figure illustrates the structure and energy distribution of the singular value spectrum of the low-rank increment matrix during training. The spectral heatmap shows that the main energy is concentrated in a few principal directions, and the overall structure remains compact. This indicates that low-rank updates effectively compress parameter changes into a finite-dimensional subspace, thereby achieving efficient adjustment of the core representation. The spectral slice curves exhibit consistent principal component dominance at different stages, indicating that the representational load distribution of incremental updates is relatively stable. It can continuously focus on key update directions while maintaining parameter efficiency

constraints, enabling the proposed method to have more reliable update organization capabilities and more controllable structured update effects under the low-rank adaptation mechanism.

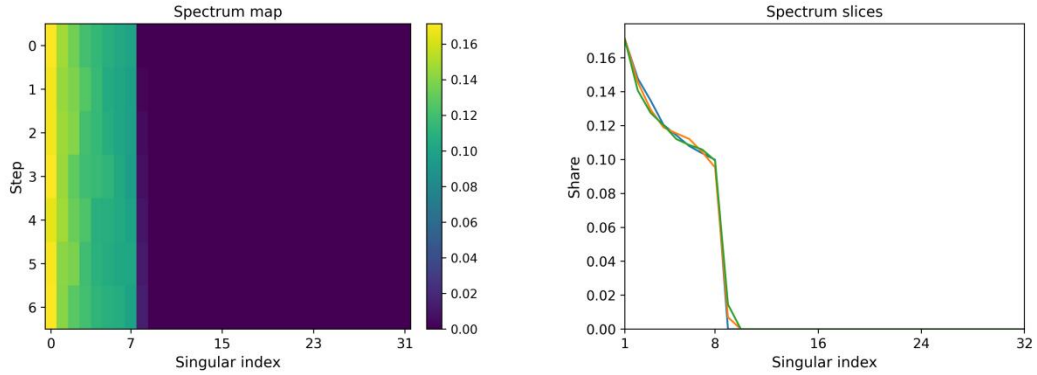

Figure 4. Visual analysis of the singular value spectrum evolution of low-rank increment matrices.

5. Conclusion

Addressing the common challenges of high fine-tuning costs, heavy version maintenance, and uncontrollable updates in the deployment of large-scale pre-trained models across multiple scenarios, a unified structured incremental fine-tuning framework is constructed. This framework, while freezing the basic parameters, restricts updates to a low-rank subspace and a sparse support set, enabling parameter increments to possess both compressible global expressiveness and selectable local correction capabilities. The resulting fine-tuning paradigm establishes a clearer structural trade-off between efficiency and expressiveness, providing a more interpretable modeling perspective and implementation path for efficient parameter fine-tuning.

Methodologically, the low-rank component carries cross-dimensional collaborative changes and maintains the overall continuity of the update, while the sparse component selects key connections through a differentiable gating mechanism to achieve more targeted local reshaping. Both work synergistically under the same optimization objective, ensuring that the update maintains a stable structural bias while avoiding widespread perturbations to irrelevant parameters. This design explicitly incorporates structural constraints into the training process, making the update complexity controllable through regularization and gating density control, thereby improving the repeatability and manageability of the fine-tuning process. Overall, this framework provides a finer-grained update organization method for large model fine-tuning, helping to alleviate common problems such as update drift and maintenance complexity during continuous iteration.

From an application impact perspective, the proposed integrated fine-tuning method provides more realistic technical support for deploying large models in resource-constrained environments and multi-task parallel iteration environments. In industry scenarios, models typically need to frequently and rapidly adapt to new data, new specifications, and new requirements, while being constrained by computing power budgets, storage costs, and delivery cycles. Structured incremental fine-tuning can lower the update threshold and improve iteration efficiency. Furthermore, the combination of low-rank and sparse update forms provides a natural interface for modular delivery and version management, enabling model capabilities to be combined, replaced, and rolled back in an incremental module manner. This promotes the standardization of engineering processes for knowledge updates and task switching, enhancing the sustainable maintenance and stable service capabilities of large models in real business processes.

Future research can focus on three directions. First, the gating selection mechanism can further introduce stronger structural priors to achieve sparsity targeting hierarchical, channel, or block structures, thereby improving the structural consistency and transferability of the sparse support set while maintaining differentiable optimization. Second, cross-task and cross-domain incremental sharing and conflict resolution

mechanisms can be explored, enabling more reasonable reuse and isolation of low-rank basis and sparse support set in multi-task adaptation, improving robustness and controllability in continuous learning scenarios. Third, for practical deployment, the storage, loading, and combination strategies of incremental parameters can be deeply integrated with the inference system to form an incremental delivery solution that coordinates the edge and cloud sides, thereby further expanding the application scope and industry impact of efficient parameter fine-tuning in fields such as intelligent customer service, content generation, knowledge-assisted decision-making, and professional text processing.

References

[1] Liu H, Tam D, Muqeeth M, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 1950-1965.

[2] S. Y. Huang, "Retrieval-Augmented Long-Text Reasoning with Semantic Conflict Detection and Cross-Paragraph Evidence Integration," Artificial Intelligence and Computing Innovations, vol. 4, no. 1, 2024.

[3] Zaken E B, Goldberg Y, Ravfogel S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2022: 1-9.

[4] Wu Z, Arora A, Wang Z, et al. Reft: Representation finetuning for language models[J]. Advances in Neural Information Processing Systems, 2024, 37: 63908-63962.

[5] Jain A, Chaudhuri S, Reps T, et al. Prompt tuning strikes back: Customizing foundation models with low-rank prompt adaptation[J]. Advances in Neural Information Processing Systems, 2024, 37: 47297-47316.

[6] Li Y, Song L, Hou H. Loran: Improved low-rank adaptation by a non-linear transformation[C]//Findings of the Association for Computational Linguistics: EMNLP 2024. 2024: 3134-3143.

[7] N. Ansell, E. Ponti, A. Pfeiffer, et al., "What Makes Good Parameter-Efficient Transfer Learning for NLP?," in Proc. ACL, 2021, pp. 1054-1065.

[8] Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models[J]. Iclr, 2022, 1(2): 3.

[9] Zhang Q, Chen M, Bukharin A, et al. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning[J]. arXiv preprint arXiv:2303.10512, 2023.

[10] Ding N, Lv X, Wang Q, et al. Sparse low-rank adaptation of pre-trained language models[C]//Proceedings of the 2023 conference on empirical methods in natural language processing. 2023: 4133-4145.

[11] J. L. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in Proc. ICLR, 2019.

[12] S. Han, J. Pool, J. Tran, and W. Dally, "Learning Both Weights and Connections for Efficient Neural Networks," in Proc. NeurIPS, 2015.

[13] A. Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient Low-Rank Hypercomplex Adapter Layers," in Proc. NeurIPS, 2021.

[14] J. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," arXiv:2210.17323, 2022.