Localized Low-Rank Adaptation within Clustered Parameter Subspaces

Jiahao Xiong[1]*, Yihe Liu[1]*, Xianming Hu[1]*, Hongbo Zhao[1], Nuoyi Chen[2], Jie Zhang[2], Kai Zhang[1]†

[1]School of Computer Science and Technology, East China Normal University
[2]Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University

Correspondence: xiongjiahao@stu.ecnu.edu.cn, kzhang@cs.ecnu.edu.cn

Abstract

Low-Rank Adaptation (LoRA) for large language models (LLMs) has achieved significant success in various domains. So far, most algorithms in the LoRA-family rely on global low-rank factors spanning the entire update weight matrix ($\Delta\mathbf{W}$). Through careful analysis, however, we observe that the $\Delta\mathbf{W}$ during fine-tuning typically exhibit heterogeneous subspace clusters, each corresponding to specific sub-sets of rows and columns. This structural heterogeneity suggests that global low-rank factors may not optimally capture the local variations needed for effective model adaptation. To address this limitation, we propose LoRA within Clustered Parameter Subspaces, or CPS-LoRA, which performs independent low-rank updates within clustered blocks of parameter matrices. The key idea is to group the rows/columns of the update matrix into locally coherent, and maximally uncorrelated subspaces, perform low-rank adaptations in each subspace, and iteratively update the partition and local adaptations. This allows adapting to local structures more precisely while preserving high efficiency. Theoretical analysis reveals that in case $\Delta\mathbf{W}$ can be partitioned into subspace blocks with non-overlapping basis, CPS-LoRA have superior parameter efficiency than global adaptations. Empirical evaluations further demonstrate better rank utilization of CPS-LoRA and its consistent improvements against LoRA (and variants) by up to 3.0% in absolute accuracy in various benchmarks.

1 Introduction

Adapting LLMs (Achiam et al., 2023; Grattafiori et al., 2024; Liu et al., 2024a) to specific domains is valuable for extending their capabilities. However, full-parameter fine-tuning can be costly. In recent years, Parameter-Efficient Fine-Tuning (PEFT) (Xu et al., 2023) emerged as a promising alternative. A prominent example is Low-Rank Adaptation (LoRA) (Hu et al., 2021), which decomposes weight updates into low-rank factors to significantly reduce the number of trainable parameters. Recent research further enhances its performance through improved initialization schemes (Meng et al., 2024), advanced architectural designs (Liu et al., 2024b; Xiong and Xie, 2025), optimized training strategies (Huang et al., 2025), parameter-sharing mechanisms (Kopiczko et al., 2024), and dynamic rank adaptation (Zhang et al., 2024).

Most existing algorithms in the LoRA-family adopt global low-rank factors that span across the entire collection of the rows and the columns in the update weight matrix. In other words, all the parameters in $\Delta\mathbf{W}$ are forced to share the same low-rank structure. A natural question thus arises: *Will the update matrix exhibit locally varying characteristics rather than uniform global patterns, and in such case will a global parameterization still be effective in achieving desired model adaptation?*

To answer this question, we carefully analyzed the update matrices in both the LLaMA2-7B (Touvron et al., 2023) and Qwen2.5-1.5B (Yang et al., 2024) models. As in Figure 1, in cases of both full-parameter fine-tuning and low-rank adaptation, the update matrix on many occasions could demonstrate block-wise structures, with each block corresponding to a specific subset of rows and columns that becomes visually prominent especially after reordering them through bi-clustering. Sometimes the boundaries of the blocks are blurred but a block-wise structure may still be observed.

The blockwise structure of $\Delta\mathbf{W}$ verifies the existence of non-uniform local patterns of the update parameters, i.e., distinct adaptation strategies might be needed across different parts of $\Delta\mathbf{W}$. So we suggest that partitioning the $\Delta\mathbf{W}$ matrix into homogeneous subspace blocks and independently learning low-rank adaptations within each block could be useful in mitigating the limitations

*Equal contribution.
†Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 26556–26572

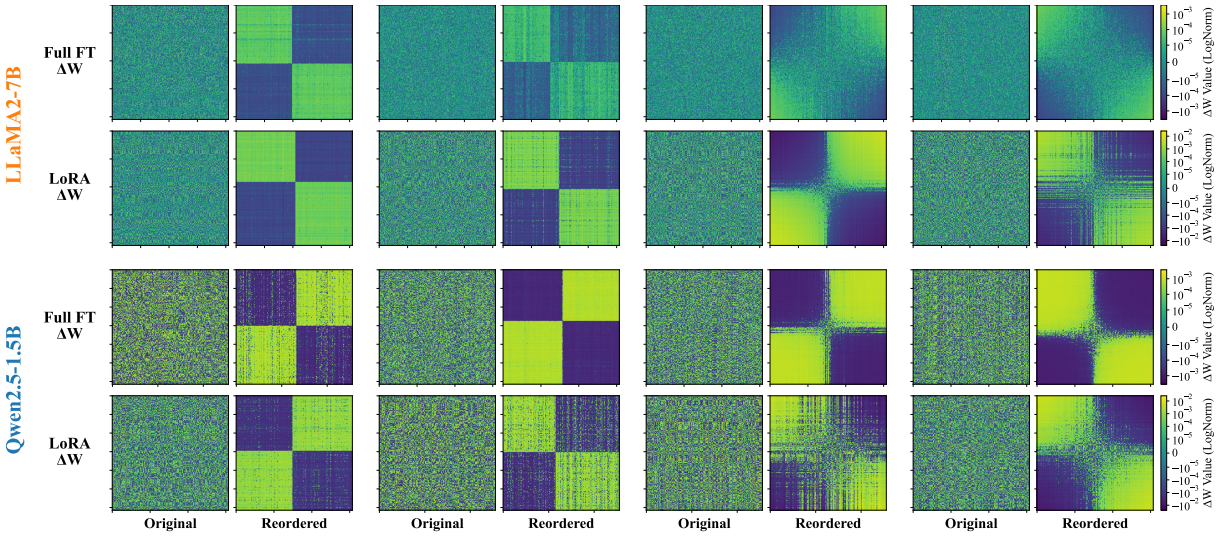


Figure 1: **Blockwise structure in $\Delta\mathbf{W}$ arises in fine-tuning**. Random Q,K,V,O layers in LLaMA2-7B and Qwen2.5-1.5B after row/column reordering, for both full-parameter fine-tuning and LoRA. Heatmap in **log-scale**.

of global parameterization, and in improving the representation power of low-rank adaptations.

Motivated by this, we propose CPS-LoRA, a local method that performs independent low-rank updates within clustered parameter subspaces. The idea is to group the rows and columns of the update matrix $\Delta\mathbf{W}$ into maximally non-overlapping subspaces and perform localized low-rank adaptation within each sub-block. To enhance the consistency (or uniformity) of each local structure, we reorder the rows and columns of $\Delta\mathbf{W}$ such that semantically similar vectors are grouped into the same subspace block. By doing this, more fine-grained and flexible weight adaptation can be enforced against different local structures to accommodate complex and diverse task requirements while remaining computationally efficient.

We have provided basic theoretical analysis to show that in case $\Delta\mathbf{W}$ contains subspace blocks with non-overlapping basis, local adaptations could be more advantageous than the global version. For example, when $\Delta\mathbf{W}$ exhibits a structure consisting of four equal-sized, orthogonal low-rank subspace blocks, CPS-LoRA achieves an effective rank twice that of standard LoRA under the same parameter budget, significantly expanding the parameter representational space and achieving greater parameter efficiency. Experimentally, CPS-LoRA improves the accuracy over both conventional LoRA and its variants by up to 3.0% on commonsense reasoning, mathematical reasoning and code generation across multiple datasets while incurring minimal computational overhead. This verifies the usefulness of block-wise low-rank updates in enhancing model fine-tuning and generalization capabilities.

In summary, our contributions are as follows:

• We propose CPS-LoRA to implement low-rank adaptations **locally in clustered subspace blocks of the update matrix**, allowing more fine-grained weight adaptation to accommodate complex tasks while maintaining computational efficiency.

• We provide theoretical analysis that CPS-LoRA achieves higher effective rank (thus parameter efficiency) than standard LoRA if $\Delta\mathbf{W}$ can be partitioned to orthogonal low-rank subspace blocks.

• We report encouraging results of CPS-LoRA across altogether 17 subtasks on commonsense reasoning, mathematical reasoning and code generation datasets against LoRA and its variants.

2 Related Work

Full-parameter fine-tuning requires significant computational resources. To address this, parameter-efficient fine-tuning (PEFT) is used to retain performance comparable to full-parameter fine-tuning while significantly reducing the number of trainable parameters. Examples include several categories, like additive-based methods (Houlsby et al., 2019; Wang et al., 2022; Li et al., 2025), prompt-based methods (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2024c), selection-based methods (Ben-Zaken et al., 2022; Kowsher et al., 2025; Wang et al., 2025), and reparameterization-based methods (Hu et al., 2021; Meng et al., 2024).

Additive-based methods enhance the adaptability of pre-trained models to downstream tasks by

inserting additional neural modules. For example, Adapters (Houlsby et al., 2019) insert lightweight modules between layers and fine-tune only these modules to reduce training costs. Such methods may lead to increased inference latency by introducing new modules into the model architecture.

Prompt-based methods aim to reduce inference latency by introducing trainable vectors while keeping pretrained model parameters fixed. Prompt Tuning (Lester et al., 2021) appends a learnable soft prompt to the input embeddings, optimized via standard back-propagation. Such methods are efficient and incur minimal parameter overhead, but may reduce the effective input sequence length by occupying part of the embedding space.

Selection-based methods improve fine-tuning efficiency and overcome the input length limitations of prompt-based methods by updating only specific parameter subsets, typically determined by layer type or model structure. For instance, BitFit (Ben-Zaken et al., 2022) fine-tunes only the bias terms in the model to enable task adaptation, while still maintaining strong performance.

Reparameterization-based methods reduce the number of trainable parameters by low-rank representations, which incur no additional inference latency since the update matrices can be absorbed into the original weights during inference. A representative class of algorithms is low-rank adaptation methods (LoRA) (Hu et al., 2021), in which all parameters of the pretrained model are frozen, and the original weight matrix is updated with two low-rank matrices multiplied together.

Building on this, AdaLoRA (Zhang et al., 2023) incorporates singular value decomposition to evaluate the importance of different modules for finer-grained resource allocation. ROSA (Hameed et al., 2024) further improves expressivity by merging the low-rank updates into the original weights at the end of each training round and recomputing the SVD to iteratively generate new subspaces. More recently, more sophisticated structural designs have also been explored. DoRA (Liu et al., 2024b) decomposes the model weights into magnitude and direction components to improve training stability. PiSSA (Meng et al., 2024) constructs update directions using the principal singular values and vectors to improve convergence and adaptation efficiency. VERA (Kopiczko et al., 2024) employs a pair of shared low-rank matrices across layers with learned scaling vectors to reduce storage. RaSA (He et al., 2025) uses a layer-specific gating scheme to dynamically regulate the use of weights components (layer-specific or cross-layer shared).

Recently, GraLoRA (Jung et al., 2025) proposes to randomly partition the weight matrix into blocks and apply low-rank adaptation in each block. This strategy has shown improved performance over standard LoRA. However, random block partitioning lacks a theoretical justification in capturing within-block homogeneity and cross-block heterogeneity - factors we have proven to be critical to take into account in enhancing the representation capacity of block-wise low-rank adaptations. Motivated by this analysis, we propose LoRA within clustered parameter subspaces - it not only promotes more uniform rank utilization (i.e., a flatter singular value spectrum of $\Delta\mathbf{W}$), but also yields notable performance improvements over both standard LoRA and its random block-wise variant.

3 Methodology

We begin with preliminaries and a theoretical motivation, followed by detailed algorithm.

3.1 Preliminaries on Low-Rank Adaptation

LoRA (Hu et al., 2021) is motivated by the assumption that weight updates during fine-tuning exhibit low rank structure as $\Delta\mathbf{W} = \mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times n}$ with $r \ll \min(m, n)$. The model output is then:

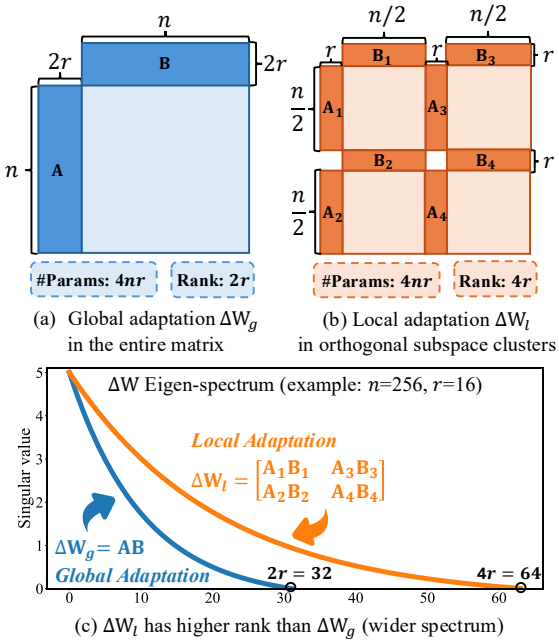
$$\mathbf{y} = \mathbf{x}(\mathbf{W}_0 + \mathbf{AB}), \tag{1}$$

where $\mathbf{W}_0$ is frozen and only $\mathbf{A}$, $\mathbf{B}$ are trainable. The $\mathbf{A}$ is initialized using a uniform Kaiming distribution (He et al., 2015), with $\mathbf{B} = 0$ to ensure that the model output remains unchanged initially.

3.2 Motivating Analysis

To illustrate the potential limit of global low-rank adaptation and inspire our method, we use a simple example in Figure 2. Consider updating an $n \times n$ weight matrix. Figure 2 (a) is a global low-rank adaptation with $2r$ rows and columns, and matrix $\mathbf{A}$ and $\mathbf{B}$ are both of rank $2r$. Figure 2 (b) is a local adaptation with four pairs of local factors, i.e., $\mathbf{A}_i$, $\mathbf{B}_i$, for $i = 1, 2, 3, 4$. Here, we assume that the update matrix $\Delta\mathbf{W}$ can be divided into 4 equal-sized sub-blocks with orthogonal basis, and $\mathbf{A}_i$'s, $\mathbf{B}_i$'s are exactly the basis of these 4 sub-blocks.

Under this setting, we can then compare the update matrix $\Delta\mathbf{W}_g$ (global setting) and $\Delta\mathbf{W}_l$ (local setting), in terms of the effective parameters and numerical ranks, as below (proof in Appendix A).



(a) Global adaptation $\Delta \mathbf{W}_g$ in the entire matrix

(b) Local adaptation $\Delta \mathbf{W}_l$ in orthogonal subspace clusters

(c) $\Delta \mathbf{W}_l$ has higher rank than $\Delta \mathbf{W}_g$ (wider spectrum)

Figure 2: Global update matrix (a) has a lower rank than local update matrix (b) with the same parameter budget under **subspace orthogonality condition** in Prop. 2; (c) singular-value spectrum of the two (numerical example).

Proposition 1 (Low-rank update in global matrix space)**.** *Assume the $n \times n$ update matrix $\Delta \mathbf{W}_g$ is written as $\Delta \mathbf{W}_g = \mathbf{AB}$, both of rank $2r$. Then, the rank of $\Delta \mathbf{W}_g$ is $2r$.*

Proposition 2 (Low rank updates in orthogonal subspace blocks)**.** *Assume the $n \times n$ update matrix $\Delta \mathbf{W}_l$ can be partitioned into four equal-sized blocks each being spanned by r low-rank factors ($r < \frac{n}{4}$), as $\Delta \mathbf{W}_l = \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_3\mathbf{B}_3 \\ \mathbf{A}_2\mathbf{B}_2 & \mathbf{A}_4\mathbf{B}_4 \end{bmatrix}$ with $\{\mathbf{A}_i\}_{i=1}^4 \in \mathbb{R}^{\frac{n}{2} \times r}$ and $\{\mathbf{B}_i\}_{i=1}^4 \in \mathbb{R}^{r \times \frac{n}{2}}$, each group having full rank r and being mutually orthogonal, i.e. $\langle \mathbf{A}_i, \mathbf{A}_j \rangle = \mathbf{0}$ and $\langle \mathbf{B}_i, \mathbf{B}_j \rangle = \mathbf{0}$ for all $i \neq j$. Then, the rank of $\Delta \mathbf{W}_l$ is $4r$.*

Comment 1. *Compare global update matrix $\Delta \mathbf{W}_g$ and local update matrix $\Delta \mathbf{W}_l$ as follows:*

- ***Effective # Params***. *Global: $4nr$; Local: $4nr$*
- ***Update Matrix Rank***. *Global: $2r$; Local: $4r$.*

That is, given the same parameter budget, localized parameterization enables a higher-rank update matrix than global version in case of orthogonal row-column subspaces.

The rank of a matrix reflects its degree of non-redundancy. Therefore, Proposition 2 indicates that when $\Delta \mathbf{W}$ exhibits orthogonal subspaces, localized adaptations can achieve greater parameter efficiency than global approaches, by attaining higher rank with the same number of parameters. Conversely, local adaptations require fewer parameters when approximating the same update matrix with orthogonal subspace basis (Appendix B). Theoretically, higher (cross) subspace orthogonality indicates higher parameter efficiency (Appendix C).

These observations motivate the use of localized adaptations within subspace blocks, forming the basis of CPS-LoRA. While subspace clusters in $\Delta \mathbf{W}$ may not be perfectly orthogonal in practice, empirical observations show that: (1) they are near-orthogonal with proper partition configuration (Figure 5); (2) localized adaptations within 2×2 clustered subspace-blocks do exhibit twice the effective rank of global LoRA with same parameter budget (Figure 4). Furthermore, the subspaces do not have to have exactly the same size (Appendix D).

CPS-LoRA include the following key steps:

1. Initialize the update weight matrix $\Delta \mathbf{W}$.
2. Reorder $\Delta \mathbf{W}$ into clustered sub-blocks.
3. Perform LoRA in each sub-block space.
4. Concatenate sub-blocks to obtain a new $\Delta \mathbf{W}$.
5. Absorb $\Delta \mathbf{W}$ into $\mathbf{W}$, and repeat step 2 - 3.

Step 2 and 3 are detailed in following subsections. Step 1 is based on a random initial partition of $\Delta \mathbf{W}$, discussed in conjunction with step 3.

3.3 Identifying Clustered Sub-blocks in $\Delta \mathbf{W}$

Based on our previous analysis, it is desirable to partition the update matrix $\Delta \mathbf{W}$ into subspace blocks such that: (1) different blocks are maximally orthogonal to each other in terms of their row and column basis; (2) vectors (row/column) within a single block are preferably similar to each other so that this block is easy to approximate with low-rank factors. To achieve these, we group the rows (or columns) of $\Delta \mathbf{W}$ into compact clusters.

We consider spectral bi-clustering (Kluger et al., 2003) on $\Delta \mathbf{W}$, a method that simultaneously embeds and clusters the rows and columns to reveal salient bi-clusters. It first obtains a non-negative $\Delta \mathbf{W}$ by $\mathbf{P} = \Delta \mathbf{W} - \Delta \mathbf{W}_{\min}$, which serves as the relational matrix among rows and columns. Then $\mathbf{P}$ is normalized on both sides as follows

$$\mathbf{S} = \mathbf{D}_r^{-\frac{1}{2}} \mathbf{P} \mathbf{D}_c^{-\frac{1}{2}}, \tag{2}$$

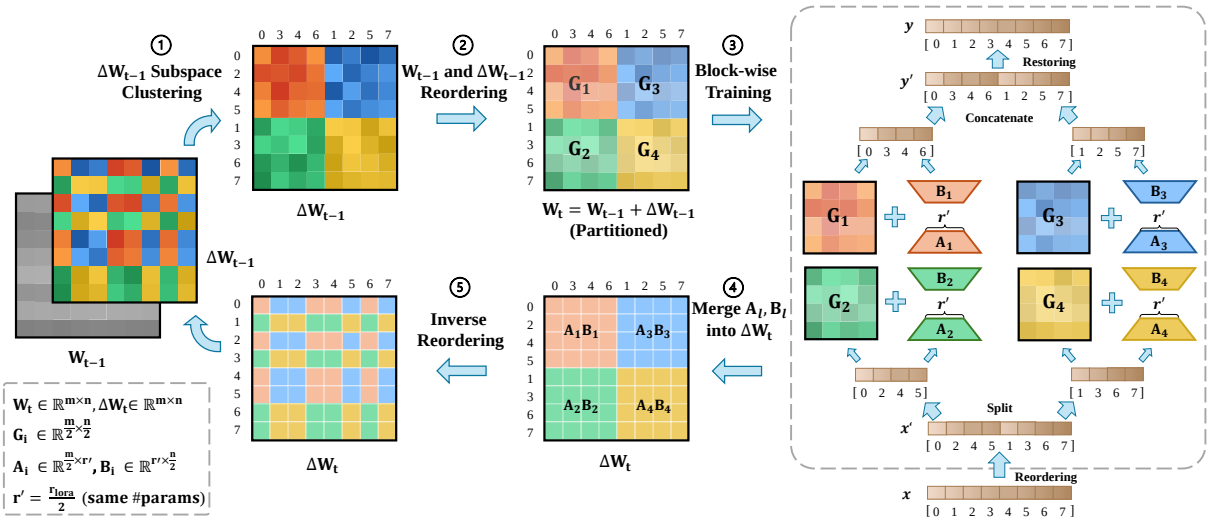


Figure 3: Overview of the CPS-LoRA using an 8×8 parameter matrix $\mathbf{W}$ and update matrix $\Delta\mathbf{W}$ as example.

where $\mathbf{D}_r = \mathrm{diag}\big(\sum_j \mathbf{P}_{ij}\big)$, $\mathbf{D}_c = \mathrm{diag}\big(\sum_i \mathbf{P}_{ij}\big)$ are row and column degree matrices, respectively, to encourage balanced clusters. Then we perform singular value decomposition (SVD) as follows

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top, \tag{3}$$

where the left singular vectors $\mathbf{U}$ and the right singular vectors $\mathbf{V}$ can be selected as row and column embeddings, respectively (by selecting the d singular vectors from the second largest singular value of the spectrum sorted in ascending order).

Having obtained the embedding of the rows and columns of the $\Delta\mathbf{W}$ matrix, we can then perform k-means to cluster these embeddings, yielding the row and column cluster indices $\boldsymbol{\pi}_r$ and $\boldsymbol{\pi}_c$, along with the corresponding partition index sets $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_k$ and $\mathcal{J} = \mathcal{J}_1, \ldots, \mathcal{J}_k$.

In case the clusters from spectral bi-clustering are imbalanced (say, a block with very small number of rows or columns), it could be desirable to re-balance this clustering by first sorting the rows and columns and then applying a balanced partition on the sorted sequence. This can be obtained by considering the first embedding-dimension in $\mathbf{U}$ and $\mathbf{V}$, denoted by $\mathbf{u}$ and $\mathbf{v}$, and then sort the elements in these vectors, as follows

$$\boldsymbol{\pi}_r = \mathrm{argsort}(\mathbf{u}), \quad \boldsymbol{\pi}_c = \mathrm{argsort}(\mathbf{v}). \tag{4}$$

Here, $\boldsymbol{\pi}_r$ and $\boldsymbol{\pi}_c$ are the permutations that rearrange rows and columns of $\Delta\mathbf{W}$ in ascending order based on univariate spectral embeddings. It is worthwhile to note that other embedding schemes like kernel PCA or cosine-similarity projection (similarity-score with a reference vector) can also be used to obtain univariate embedding of $\Delta\mathbf{W}$'s rows and columns, by incorporating nonlinear manifold structures or promoting structural coherence by grouping similarly oriented vectors together.

All above clustering/reordering procedures of $\Delta\mathbf{W}$ can be implemented equivalently on its factorized form instead of the $n \times n$ matrix, being more time and memory efficient (Appendix E).

3.4 Local Adaptation in Subspace Clusters

Suppose we are given $\mathbf{W}_{t-1}$ and $\Delta\mathbf{W}_{t-1}$ obtained in the previous iteration. We also have the permutations $\boldsymbol{\pi}_r$, $\boldsymbol{\pi}_c$ that sort the rows/columns of $\Delta\mathbf{W}_{t-1}$ into locally coherent blocks with respective row partitioning set $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ and column partitioning set $\mathcal{J} = \{\mathcal{J}_1, \ldots, \mathcal{J}_k\}$. Now we show how to incorporate these into CPS-LoRA training.

First, we reorder the weight matrix $\mathbf{W}_t \in \mathbb{R}^{m \times n}$, where $\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta\mathbf{W}_{t-1}$, according to the row and column permutations $\boldsymbol{\pi}_r$ and $\boldsymbol{\pi}_c$ to align it with the update matrix (since these two matrices are added directly and their corresponding rows and columns need to be strictly paired):

$$\mathbf{W}'_t = \mathbf{W}_t[\boldsymbol{\pi}_r, :][:, \boldsymbol{\pi}_c]. \tag{5}$$

Under this permutation, we use the partition $\mathcal{I}$ and $\mathcal{J}$ to break $\mathbf{W}_t$ into $k \times k$ blocks, as

$$\mathbf{W}'_t = \begin{bmatrix} \mathbf{G}_1 & \cdots & \mathbf{G}_{1+k\times(k-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_k & \cdots & \mathbf{G}_{k^2} \end{bmatrix}, \tag{6}$$

where each sub-block $\mathbf{G}_l = \mathbf{W}'_t[\mathcal{I}_i, \mathcal{J}_j]$ is indexed by $l = i + k \times (j-1)$. Here, permutation $\boldsymbol{\pi}_r$ (or

$\boldsymbol{\pi}_c$) and the partition $\mathcal{I}_i$'s (or $\mathcal{J}_j$'s) are consistent, in that the partitions $\mathcal{I}_i$'s are exactly "cuts" on the permuted row (or column) sequence.

Due to the permutations of the rows and columns in $\Delta\mathbf{W}_t$ as well as the corresponding weight matrix $\mathbf{W}_t$, the input vector $\mathbf{x}$ (see Eq. 7) should also be re-ordered based on the row permutation indices $\boldsymbol{\pi}_r$ in order to align to the parameter matrices and generate the correct training result, i.e.,

$$\mathbf{x}' = \mathbf{x}[:, \boldsymbol{\pi}_r]. \tag{7}$$

For each sub-block $\mathbf{G}_l$, we use a pair of low-rank factors $\mathbf{A}_l \in \mathbb{R}^{|\mathcal{I}_i| \times r'}$ and $\mathbf{B}_l \in \mathbb{R}^{r' \times |\mathcal{J}_j|}$. Here, $r' = r/k$ is the rank allocated to each sub-block, ensuring that the overall parameter count matches that of a standard LoRA module with rank r.

Consequently, the output can be expressed as:

$$\mathbf{y}' = \text{Concat}\left(\left[\sum_{i=1}^{k} \mathbf{x}'[\mathcal{I}_i](\mathbf{G}_l + \mathbf{A}_l\mathbf{B}_l)\right]_{j=1}^{k}\right), \tag{8}$$

where, as previously defined, $l = i + k \times (j-1)$.

After computing $\mathbf{y}'$, we apply the inverse permutation to $\mathbf{y}'$ to restore the output to its original semantic order, thereby ensuring that the subsequent module receives input in the correct order:

$$\mathbf{y} = \mathbf{y}'[:, \boldsymbol{\pi}_c^{-1}]. \tag{9}$$

With the output $\mathbf{y}$, we can then compute the loss term (Cross-Entropy), based on which $\mathbf{A}_i$ and $\mathbf{B}_i$ are optimized via backpropagation implemented in Hugging Face Transformers (Wolf et al., 2020). After computing the $\mathbf{A}_i$'s and $\mathbf{B}_i$'s for each sub-block, we then merge them together to recover the entire update matrix $\Delta\mathbf{W}'_t$ in the permuted space.

$$\Delta\mathbf{W}'_t = \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 & \cdots & \mathbf{A}_{1+k\times(k-1)}\mathbf{B}_{1+k\times(k-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_k\mathbf{B}_k & \cdots & \mathbf{A}_{k^2}\mathbf{B}_{k^2} \end{bmatrix} \tag{10}$$

By applying the inverse permutation $\boldsymbol{\pi}_r^{-1}$ and $\boldsymbol{\pi}_c^{-1}$, we recover $\Delta\mathbf{W}_t$ in the original order, which can be used for the next round of iterations.

$$\Delta\mathbf{W}_t = \Delta\mathbf{W}'_t[\boldsymbol{\pi}_r^{-1}, :][:, \boldsymbol{\pi}_c^{-1}]. \tag{11}$$

Finally, we describe how to initialize $\Delta\mathbf{W}$, whose partitioning is the key to finding orthogonal subspaces for localized low-rank adaptation. During the *initial stage* (e.g., the first epoch), we *randomly partition* the target matrix into balanced sub-blocks and apply *localized low-rank adaptations* within each sub-block. By the end of this stage, the corresponding low-rank factors $\mathbf{A}_i$'s and $\mathbf{B}_i$'s from each sub-block are merged to reconstruct the full $\Delta\mathbf{W}$ matrix. Starting from the *second epoch*, we *re-partition* $\Delta\mathbf{W}$ into new sub-blocks and *re-train* the low-rank factors within each updated partition. The algorithmic procedure is provided in Appendix F.

4 Experiments

Benchamrk. Following the literature (Hu et al., 2023; Chaudhary, 2023; Hameed et al., 2024; He et al., 2025; Jung et al., 2025), we perform evaluation on three popular benchmark datasets comprising 17 diverse tasks, using the LLaMA2-7B, LLaMA2-13B, LLaMA3-8B-Instruct and Qwen2.5-1.5B as the base models.

Baselines. We include five strong baselines, including LoRA (Hu et al., 2021), DoRA (Liu et al., 2024b), PiSSA (Meng et al., 2024), RaSA (He et al., 2025), and GraLoRA (Jung et al., 2025). Other PEFT methods (e.g., adapter- or prompt-based) are excluded from consideration, as their architectural designs differ substantially from low-rank methods (such as larger number of epochs needed, dynamically changing the number of training parameters, or full fine-tuning warm-up).

Settings. To ensure fair comparisons, we adopted commonly used settings for all methods: (1) the number of trainable parameters ($r = 8$); (2) tuning Q, K, V, O matrices; (3) batch size as 4; and (4) training for 2 epochs with AdamW optimizer. More details are reported in the Appendix G.

4.1 Evaluation Results

Commonsense reasoning. As shown in Table 1, we fine-tuned LLaMA2-7B/13B, LLaMA3-8B-Instruct and Qwen2.5-1.5B on Commonsense-170K, and evaluated performance on the commonsense reasoning benchmark. Compared with standard LoRA, CPS-LoRA consistently improves performance across all models. On LLaMA2-7B, the average accuracy increases from 79.6% to 82.1% (+2.5 points). Similar gains are observed on larger models, reaching 84.7% on LLaMA2-13B and 87.0% on LLaMA3-8B-Instruct. On Qwen2.5-1.5B, CPS-LoRA also achieves strong results.

Mathematical reasoning. As shown in Table 2, we fine-tuned LLaMA2-7B/13B, LLaMA3-8B-Instruct and Qwen2.5-1.5B on Math10K and made evaluations on seven mathematical reasoning

Base Models	Methods	Params	BoolQ	PIQA	SIQA	ARC-C	ARC-E	OBQA	HellaSwag	Winogrande	Avg
LLaMA2 (7B)	LoRA	8.39M	68.5%	81.6%	79.9%	71.8%	85.7%	76.0%	92.8%	81.1%	79.6%
	DoRA	8.91M	68.9%	80.9%	79.7%	71.7%	86.4%	77.6%	92.9%	81.9%	80.0%
	PiSSA	8.39M	69.5%	**82.5%**	79.4%	72.6%	86.0%	79.2%	92.9%	83.0%	80.6%
	RaSA	8.39M	68.2%	80.8%	79.4%	71.5%	86.0%	78.4%	92.4%	79.3%	79.5%
	GraLoRA	8.39M	68.0%	81.7%	79.0%	69.5%	85.7%	77.6%	92.2%	79.2%	79.1%
	CPS-LoRA	8.39M	**70.9%**	**82.5%**	**81.7%**	**73.1%**	**86.3%**	**84.2%**	**93.9%**	**84.0%**	**82.1%**
LLaMA2 (13B)	LoRA	13.11M	71.7%	85.6%	81.0%	78.9%	89.3%	81.6%	95.0%	86.3%	83.7%
	DoRA	13.93M	71.6%	85.7%	81.3%	**79.9%**	89.3%	81.6%	94.9%	86.8%	83.9%
	PiSSA	13.11M	71.5%	**86.2%**	81.4%	77.0%	89.6%	82.8%	94.9%	85.5%	83.6%
	RaSA	13.11M	70.9%	85.3%	80.8%	76.5%	89.1%	80.6%	94.5%	86.7%	83.1%
	GraLoRA	13.11M	71.2%	85.3%	80.7%	75.6%	89.2%	79.8%	94.3%	85.8%	82.7%
	CPS-LoRA	13.11M	**73.7%**	86.1%	**82.1%**	78.4%	**90.4%**	**85.4%**	**95.6%**	**87.2%**	**84.9%**
LLaMA3 (8B) Instruct	LoRA	6.82M	73.0%	89.3%	81.7%	82.8%	93.6%	87.6%	95.9%	**88.8%**	86.6%
	DoRA	7.14M	72.8%	89.1%	81.6%	82.0%	**93.7%**	86.8%	**96.0%**	88.6%	86.3%
	PiSSA	6.82M	72.3%	88.3%	80.9%	83.1%	93.3%	86.2%	95.9%	87.2%	85.9%
	RaSA	6.82M	72.7%	88.4%	81.6%	81.8%	93.5%	86.0%	95.5%	87.3%	85.8%
	GraLoRA	6.82M	72.9%	88.4%	81.2%	81.6%	93.1%	87.0%	95.3%	86.4%	85.7%
	CPS-LoRA	6.82M	**75.0%**	**89.5%**	**82.2%**	**84.0%**	**93.7%**	**87.8%**	**96.0%**	87.6%	**87.0%**
Qwen2.5 (1.5B)	LoRA	2.18M	66.0%	80.7%	73.0%	75.4%	89.4%	76.8%	86.4%	69.5%	77.2%
	DoRA	2.28M	66.9%	80.7%	73.2%	75.4%	89.4%	77.2%	86.3%	69.0%	77.3%
	PiSSA	2.18M	66.7%	81.2%	73.4%	77.0%	89.6%	78.6%	87.4%	70.2%	78.0%
	RaSA	2.18M	65.5%	80.4%	72.7%	74.7%	89.0%	76.0%	84.8%	68.0%	76.4%
	GraLoRA	2.18M	64.5%	79.7%	73.1%	75.0%	88.8%	73.8%	84.1%	66.7%	75.7%
	CPS-LoRA	2.18M	**67.4%**	**82.4%**	**76.0%**	**78.6%**	**90.0%**	**83.6%**	**90.0%**	**75.2%**	**80.4%**

Table 1: Evaluation with LLaMA2/3 and Qwen2.5 models on commonsense reasoning task.

Base Models	Methods	Params	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	MAWPS	Avg
LLaMA2 (7B)	LoRA	8.39M	94.0%	41.0%	90.1%	21.7%	87.6%	58.7%	83.6%	68.1%
	DoRA	8.91M	94.2%	41.2%	89.1%	21.7%	86.0%	59.8%	85.3%	68.2%
	PiSSA	8.39M	94.0%	40.9%	89.6%	**26.0%**	90.2%	57.0%	86.1%	69.1%
	RaSA	8.39M	93.3%	39.0%	88.9%	24.8%	87.6%	58.8%	84.9%	68.2%
	GraLoRA	8.39M	93.3%	**41.4%**	88.4%	21.7%	87.4%	57.6%	84.0%	67.7%
	CPS-LoRA	8.39M	**97.0%**	40.1%	**90.4%**	24.4%	**91.3%**	**62.3%**	**86.6%**	**70.3%**
LLaMA2 (13B)	LoRA	13.11M	98.0%	52.0%	90.4%	23.2%	93.3%	69.2%	86.6%	73.2%
	DoRA	13.93M	98.5%	51.9%	89.9%	23.2%	92.9%	69.8%	**87.0%**	73.3%
	PiSSA	13.11M	98.3%	**56.6%**	**91.1%**	22.8%	93.5%	67.7%	86.1%	73.7%
	RaSA	13.11M	96.7%	50.6%	88.9%	**26.0%**	92.9%	70.6%	87.4%	73.3%
	GraLoRA	13.11M	95.0%	53.5%	88.1%	24.0%	93.7%	69.6%	84.9%	72.7%
	CPS-LoRA	13.11M	**98.7%**	54.0%	**91.1%**	23.2%	**94.9%**	**72.4%**	85.7%	**74.3%**
LLaMA3 (8B) Instruct	LoRA	6.82M	97.3%	74.9%	89.6%	32.3%	94.3%	**84.2%**	89.9%	80.4%
	DoRA	7.14M	97.5%	74.3%	89.6%	32.7%	94.7%	83.6%	89.5%	80.3%
	PiSSA	6.82M	97.3%	74.4%	89.9%	31.9%	**95.1%**	82.7%	91.2%	80.4%
	RaSA	6.82M	**98.2%**	74.7%	**90.9%**	30.3%	94.7%	82.6%	90.8%	80.3%
	GraLoRA	6.82M	97.7%	74.7%	90.1%	33.1%	94.9%	82.6%	90.8%	80.4%
	CPS-LoRA	6.82M	97.0%	**76.0%**	89.9%	**35.8%**	**95.1%**	83.7%	**92.0%**	**81.4%**
Qwen2.5 (1.5B)	LoRA	2.18M	97.3%	65.6%	90.9%	34.3%	96.5%	82.6%	91.6%	79.8%
	DoRA	2.28M	97.5%	65.6%	90.4%	35.4%	95.9%	81.4%	90.3%	79.5%
	PiSSA	2.18M	**98.8%**	66.7%	91.1%	31.1%	96.5%	80.9%	**92.0%**	79.6%
	RaSA	2.18M	97.8%	67.6%	89.9%	34.6%	96.5%	81.7%	91.6%	80.0%
	GraLoRA	2.18M	98.0%	67.2%	90.4%	**35.8%**	96.3%	81.7%	91.6%	80.1%
	CPS-LoRA	2.18M	97.7%	**68.5%**	**91.4%**	34.6%	**97.0%**	**82.8%**	91.6%	**80.5%**

Table 2: Evaluation with LLaMA2/3 and Qwen2.5 models on mathematical reasoning task.

datasets. CPS-LoRA consistently improves performance across all models. On LLaMA2-7B, the average accuracy increases from 68.1% to 70.3%. On the more advanced LLaMA3-8B-Instruct, CPS-LoRA exhibits a similar improvement trend, achieving an accuracy of 81.4%. These improvements further extend to models of varying scales, where CPS-LoRA achieves 74.3% on LLaMA2-13B and 80.5% on Qwen2.5-1.5B.

Code generation. We fine-tuned the LLaMA3-8B-Instruct and Qwen2.5-1.5B on the CodeAlpaca20K dataset, and evaluated them on the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) datasets. As shown in Table 3, CPS-LoRA achieves the highest average accuracy of 61.2% across the two coding tasks, while also delivering the best performance on Qwen2.5-1.5B with an accuracy of 50.9%.

Base Models	Methods	MBPP	HumanEval	Avg
LLaMA3 (8B) Instruct	LoRA	63.4%	56.1%	59.8%
	DoRA	63.4%	56.7%	60.1%
	PiSSA	61.9%	57.3%	59.6%
	RaSA	63.4%	57.3%	60.4%
	GraLoRA	61.9%	57.9%	59.9%
	CPS-LoRA	**63.8%**	**58.5%**	**61.2%**
Qwen2.5 (1.5B)	LoRA	50.2%	37.2%	43.7%
	DoRA	50.6%	36.0%	43.3%
	PiSSA	51.4%	36.6%	44.0%
	RaSA	54.5%	42.7%	48.6%
	GraLoRA	51.8%	39.6%	45.7%
	CPS-LoRA	**56.8%**	**45.1%**	**50.9%**

Table 3: Evaluation with LLaMA3-8B-Instruct and Qwen2.5-1.5B models on code generation task.

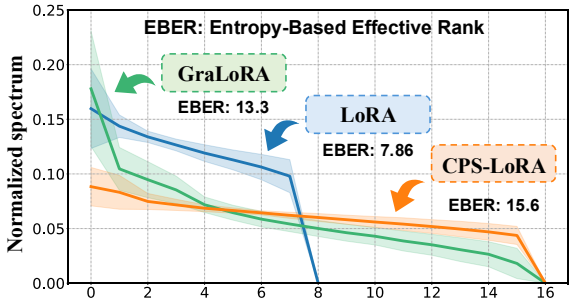


Figure 4: Normalized singular-val. spectra with entropy-based effective rank of LoRA, GraLoRA, and CPS-LoRA (with $n \times 8$ para. budget for each A or B). Each spectrum is averaged across all Q,K,V,O update matrices, color band denotes one standard deviation.

Overall, CPS-LoRA achieves better or comparable performance across the 17 evaluated tasks, and showing the generality of CPS-across different model architectures both small and large bases.

4.2 Ablation study

Effective Rank of ΔW. Figure 4 plots singular-value spectrum of LoRA, GraLoRA (random-partition) and CPS-LoRA (clustered subspace partition), under the fixed parameter budget (two 8×4096 matrices). The spectrum is normalized to sum up to 1, and averaged across 128 ΔW matrices from 32 layers of LLaMA2-7B. We can see that: (1) global methods like LoRA have rank 8, while local methods like CPS-LoRA (ours) and GraLoRA have 16, showing superior parameter efficiency (consistent with our theoretical analysis); (2) although both GraLoRA and our method attain twice the algebraic rank of global adaptations, our method has a much flatter singular value spectrum with s.t.d. 0.018, while that for GraLoRA is 0.053.

We further compute **Entropy-Based Effective Rank (EBER)** - the entropy of the normalized spectrum - which is upper bounded by the algebraic matrix rank but can better capture the "effective" rank and reveal how effectively that rank is utilized (Roy and Vetterli, 2007). CPS-LoRA has an EBER of 15.6, approaching the upper bound of the matrix rank (16), while GraLoRA is around 13.3. This indicates improved rank utilization of our method (with clustering based partition) over random partitions in practical finetuning tasks. See a detailed analysis for all competing methods in Appendix I.

Method	Rank-8	Rank-16	Rank-64
LoRA	68.1%	70.0%	71.0%
GraLoRA	67.7%	69.5%	71.1%
CPS-LoRA	**70.3%**	**70.5%**	**71.6%**

Table 4: Accuracy of LoRA, GraLoRA, and CPS-LoRA under varying ranks on Math10K dataset.

Reordering/Partition Strategy	Avg
Random Partitioning (GraLoRA)	67.7%
Epoch-wise Merging	68.3%
Random Partitioning + Epoch-wise Merging	68.6%
Spectral Bi-clustering	69.6%
Spectral Bi-clustering (balance-1)	70.0%
Spectral Bi-clustering (balance-2)	**70.3%**
Cosine Similarity Sorting	**70.3%**

Table 5: CPS-LoRA reordering/partition strategies.

Choice of the rank. Table 4 reports the accuracies of LoRA, GraLoRA (random-partition), and our CPS-LoRA (clustered subspace partition) with rank 8, 16, and 64 on Math10K with LLaMA2-7B. CPS-LoRA consistently achieves higher accuracies across all rank configurations. In particular, CPS-LoRA at rank 8 outperforms LoRA and GraLoRA at rank 16, indicating the usefulness of localized adaptations within clustered subspaces.

Partitioning Strategy. Spectral bi-clustering are prone to imbalanced clusters (Qian and Saligrama, 2013), so we consider two balancing strategies when the bi-clusters are skewed: (1) sorting rows/columns by the dominant singular vector (denoted as balance-1, see Eq.4); and (2) sorting by cosine similarity between row/column and a reference vector (balance-2). As in Table 5, balanced variants slightly outperform spectral bi-clustering on Math10K with LLaMA2-7B (0.4–0.7%). Intriguingly, using cosine similarity–based partitioning from scratch - rather than only correcting imbalanced bi-clustering results - gives an accuracy of 70.3%, matching the balanced versions. Given its

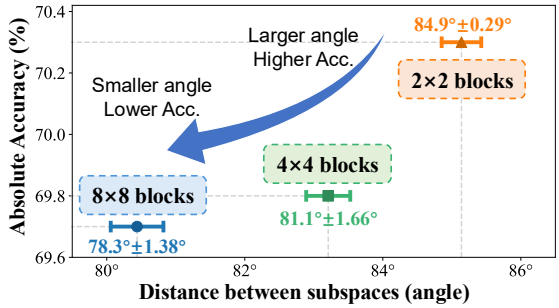


Figure 5: Model performance versus subspace angular distance under 2×2, 4×4, and 8×8 sub-block partitions.

simplicity and low cost, we therefore adopt cosine similarity–based partitioning in all experiments.

Epoch-wise merging. CPS-LoRA explicitly merges the low-rank update into the base model at each iteration. In contrast, GraLoRA (Jung et al., 2025) does not perform epoch-wise merging. To ensure a fair comparison, Table 5 additionally reports a GraLoRA variant with explicit merging at each epoch. The results show that, even under this setting, random partitioning remains approximately 1.7% inferior to structured partitioning. This indicates that the performance gains of CPS-LoRA primarily stem from the structured partitioning strategy rather than the merging scheme.

Efficiency. Overall, the training efficiency of LoRA, GraLoRA, and CPS-LoRA is comparable, requiring 71, 74, and 77 minutes, respectively, on Math10K with LLaMA2-7B base. Notably, the row and column reordering in CPS-LoRA can be equivalently performed on the input features and low-rank factors to avoid explicit permutation of the full weight matrix, thus only modest computational overhead was introduced during training. Further discussion on distributed training considerations, along with the detailed equivalent formulation, is provided in Appendix K. In inference time, CPS-LoRA is as efficient as standard LoRA because the update matrices are absorbed into the model without additional inference overhead.

Effect of Partition Granularity. Figure 5 evaluates how the number of blocks affects CPS-LoRA performance. Here we experiment with 2×2, 4×4, and 8×8 sub-block partitions on Math10K with LLaMA2-7B. For a fair comparison, we set the rank per sub-block to 4, 2, and 1, so that the total number of parameters remains constant across them. As shown, increasing the number of blocks leads to a slight performance degradation—from 70.3% with 2×2 partitions to 69.8% and 69.7% with 4×4 and 8×8 partitions.

To understand why finer-grained partitions impair performance, we analyze the angular distance between subspaces, as defined in Appendix L. As in Figure 5, increasing the number of sub-blocks leads to a decrease in the average angular distance, from $84.9°$ (2×2 partition) to $78.3°$ (8×8 partition). Namely, subspaces become increasingly homogeneous and their bases less orthogonal. Based on Prop. 2 and Appendix C, this could hamper parameter efficiency of CPS-LoRA and its performance. In practice, the 2×2 partition achieves the best performance, corresponding to the highest degree of subspace orthogonality and diversity.

Single-Epoch Training Analysis. Since CPS-LoRA performs subspace re-clustering from the second epoch onward, its effectiveness under extremely limited training budgets warrants examination. To evaluate this, we conduct a single-epoch experiment on the mathematical reasoning task with LLaMA2-7B, incorporating a brief 100-step warm-up to approximate the first epoch and enable subspace re-clustering. As shown in Table 6, CPS-LoRA consistently outperforms LoRA and its recent variants. In particular, CPS-LoRA improves performance by +2.2% over GraLoRA and +1.4% over LoRA. This suggests that clustered subspace initialization provides a strong inductive bias, enabling effective adaptation even under highly constrained training budgets.

Method	LoRA	PiSSA	DoRA	GraLoRA	RaSA	**CPS-LoRA**
Avg Acc.	65.7	65.9	65.5	64.9	64.8	**67.1**

Table 6: Effectiveness under a single-epoch training.

5 Conclusions

In this work, we propose CPS-LoRA, a localized low-rank adaptation method within clustered parameter subspaces, with both theoretical motivations and encouraging empirical performance when compared with a number of methods in the LoRA family. In the future, we will further explore adaptive rank assignment across subspace clusters so as to enhance the flexibility of local adaptations.

6 Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (62276099).

7 Limitations

We acknowledge several limitations of this work. CPS-LoRA is evaluated on base models ranging from 1.5B to 13B parameters. Owing to computational resource constraints, experiments on substantially larger models are not included. Although this range covers several widely used large language models, further validation on larger-scale architectures would help to better assess the scalability of the proposed approach. In addition, the current implementation adopts a relatively simple spectral bi-clustering strategy. More advanced or adaptive clustering methods are not explored.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 1–9. Association for Computational Linguistics (ACL).

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Marawan Gamal Abdel Hameed, Aristides Milios, Siva Reddy, and Guillaume Rabusseau. 2024. Rosa: Random subspace adaptation for efficient fine-tuning. *Preprint*, arXiv:2407.07802.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

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 *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533.

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)*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5254–5276.

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*.

Yeonjoon Jung, Daehyun Ahn, Hyungjun Kim, Taesu Kim, and Eunhyeok Park. 2025. Gralora: Granular low-rank adaptation for parameter-efficient fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. 2003. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4):703–716.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of NAACL*, pages 1152–1157.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Md Kowsher, Tara Esmaeilbeig, Chun-Nam Yu, Chen Chen, Mojtaba Soltanalian, and Niloofar Yousefi. 2025. Rocoft: Efficient finetuning of large language models with row-column updates. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26659–26678.

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)*, pages 3045–3059.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, Ruixuan Li, and 1 others. 2025. Beyond higher rank: Token-wise input-output projections for efficient low-rank adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4582–4597.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Zequan Liu, Yi Zhao, Ming Tan, Wei Zhu, and Aaron Xuxiang Tian. 2024c. Para: Parameter-efficient fine-tuning with prompt-aware representation adjustment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 728–737.

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 (NeurIPS)*, volume 37, pages 121038–121072.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*, pages 2080–2094.

Jing Qian and Venkatesh Saligrama. 2013. Spectral clustering with unbalanced data. *Preprint*, arXiv:1302.5134.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Runyu Wang, Peng Ping, Zhengyu Guo, Xiaoye Zhang, Quan Shi, Liting Zhou, and Tianbo Ji. 2025. Loki: Low-damage knowledge implanting of large language models. *arXiv preprint arXiv:2505.22120*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations

for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5744–5760.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pages 38–45, Online. Association for Computational Linguistics.

Yifeng Xiong and Xiaohui Xie. 2025. Oplora: Orthogonal projection lora prevents catastrophic forgetting during parameter-efficient fine-tuning. *arXiv preprint arXiv:2510.13003*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *Preprint*, arXiv:2312.12148.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

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 *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5048–5060.

A Proof of Proposition 2

Proposition 2. *Assume the $n \times n$ update matrix $\Delta\mathbf{W}_l$ can be partitioned into four equal-sized blocks each being spanned by r low-rank factors ($r < \frac{n}{4}$), as $\Delta\mathbf{W}_l = \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_3\mathbf{B}_3 \\ \mathbf{A}_2\mathbf{B}_2 & \mathbf{A}_4\mathbf{B}_4 \end{bmatrix}$ with $\{\mathbf{A}_i\}_{i=1}^4 \in \mathbb{R}^{\frac{n}{2}\times r}$ and $\{\mathbf{B}_i\}_{i=1}^4 \in \mathbb{R}^{r\times\frac{n}{2}}$, with each group having full rank r and being mutually orthogonal, i.e. $\langle \mathbf{A}_i, \mathbf{A}_j\rangle = \mathbf{0}$ and $\langle \mathbf{B}_i, \mathbf{B}_j\rangle = \mathbf{0}$ for all $i \neq j$. Then, the rank of $\Delta\mathbf{W}_l$ is $4r$.*

Proof. The update matrix $\Delta\mathbf{W}_l$ admits a factorization as the product of two matrices, $\mathbf{M}_A \in \mathbb{R}^{n\times 4r}$ and $\mathbf{M}_B \in \mathbb{R}^{4r\times n}$,satisfying $\Delta\mathbf{W}_l = \mathbf{M}_A\mathbf{M}_B$:

$$\mathbf{M}_A = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_3 & 0 & 0 \\ 0 & 0 & \mathbf{A}_2 & \mathbf{A}_4 \end{bmatrix},$$

$$\mathbf{M}_B = \begin{bmatrix} \mathbf{B}_1 & 0 \\ 0 & \mathbf{B}_3 \\ \mathbf{B}_2 & 0 \\ 0 & \mathbf{B}_4 \end{bmatrix},$$

where each $\mathbf{A}_i \in \mathbb{R}^{\frac{n}{2}\times r}$ and $\mathbf{B}_i \in \mathbb{R}^{r\times\frac{n}{2}}$ for $i = 1, 2, 3, 4$. Owing to the block-diagonal structure of the matrix $\mathbf{M}_A$, we have $\mathrm{rank}(\mathbf{M}_A) = \mathrm{rank}\left(\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_3\end{bmatrix}\right) + \mathrm{rank}\left(\begin{bmatrix}\mathbf{A}_2 & \mathbf{A}_4\end{bmatrix}\right)$. Since $\{\mathbf{A}_i\}$'s are all mutually orthogonal to each other, both blocks $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_3\end{bmatrix}$ and $\begin{bmatrix}\mathbf{A}_2 & \mathbf{A}_4\end{bmatrix}$ have a rank $2r$. Therefore, $\mathrm{rank}(\mathbf{M}_A) = 4r$.

For $\mathbf{M}_B$, we apply a row permutation, which preserves the rank of the matrix:

$$\mathrm{rank}(\mathbf{M}_B) = \mathrm{rank}\left(\begin{bmatrix} \mathbf{B}_1 & 0 \\ 0 & \mathbf{B}_3 \\ \mathbf{B}_2 & 0 \\ 0 & \mathbf{B}_4 \end{bmatrix}\right)$$

$$= \mathrm{rank}\left(\begin{bmatrix} \mathbf{B}_1 & 0 \\ \mathbf{B}_2 & 0 \\ 0 & \mathbf{B}_3 \\ 0 & \mathbf{B}_4 \end{bmatrix}\right).$$

Similar to $\mathbf{M}_A$, the rank of $\mathbf{M}_B$ is also $4r$. Since the rank of the product of two matrices is bounded above by the rank of each matrix, we have $\mathrm{rank}(\Delta\mathbf{W}_l) \leq \min(\mathrm{rank}(\mathbf{M}_A), \mathrm{rank}(\mathbf{M}_B)) = 4r$. On the other hand, by Sylvester's rank inequality $\mathrm{rank}(\Delta\mathbf{W}_l) \geq \mathrm{rank}(\mathbf{M}_A) + \mathrm{rank}(\mathbf{M}_B) - 4r = 4r$.

Therefore, combining both the upper and lower bounds, the rank of $\Delta\mathbf{W}_l$ is $4r$. □

B Number of Parameters Needed to Recover Update Matrix with Orthogonal Subspaces

To further illustrate the parameter efficiency advantage of localized adaptation in the presence of orthogonal subspaces, we consider an example in which the target matrix $\Delta\mathbf{W}$ consists of four orthogonal blocks. Then we show that global adaptations need more parameters to approximate this matrix than local adaptations.

As shown in Figure 6, in a numerical example with $n = 256$ and $r = 16$, local adaptation

achieves the same reconstruction accuracy as the global approach while using only half the number of parameters. These results show that, when the update matrix is composed of orthogonal subspaces, localized adaptation can offer significantly greater parameter efficiency compared to a global method.

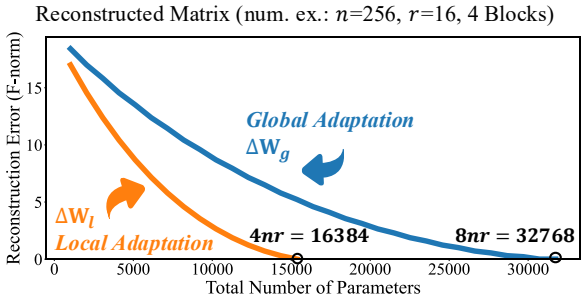


Figure 6: A numerical example comparing global and local adaptation for reconstructing a matrix composed of four mutually orthogonal blocks.

C Subspaces not Perfectly Orthogonal

The update matrix $\Delta\mathbf{W}_l$ is composed of four mutually orthogonal subspaces, achieving a rank of $4r$ with $4nr$ parameters. Under the same parameter budget, LoRA can only reach rank $2r$. This demonstrates that, for the same number of parameters, block-wise orthogonal parameterization yields higher rank and thus greater parameter efficiency.

In practice, the subspaces represented by different sub-blocks often exhibit a certain degree of redundancy. For example, if $\mathbf{A}_1$ and $\mathbf{A}_2$ in $\Delta\mathbf{W}_l$ share s common basis vectors in their column subspaces while the remaining subspaces are orthogonal, it is straightforward to show that the total rank of $\Delta\mathbf{W}_l$ becomes $4r - s$, which is lower than that in the fully orthogonal case.

Therefore, the more orthogonal the subspace bases are to each other, the greater the parameter efficiency gains compared to global low-rank parameterization. As the overlap among subspaces increases, redundancy reduces the achievable rank for same parameter budget. In fact, when the sub-blocks are quite similar, local adaptations are no longer desirable and parameter sharing among the sub-blocks instead should be encouraged.

D Sub-Blocks with Different Sizes / Ranks

It should be noted that, in practice, the blocks produced by row and column clustering are generally not strictly square or of uniform size; instead, each block may have different row and column dimensions as well as possibly different ranks. Nevertheless, our proof does not depend on these properties. Instead, it only requires that the four blocks are mutually orthogonal or non-overlapping. In this case, the rank of the update matrix is simply the sum of the ranks of the four individual blocks.

Accordingly, we consider a more general block structure of the form $\Delta\mathbf{W}_l = \begin{bmatrix} \mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_3\mathbf{B}_3 \\ \mathbf{A}_2\mathbf{B}_2 & \mathbf{A}_4\mathbf{B}_4 \end{bmatrix}$, where for each $i = 1, 2, 3, 4$, $\mathbf{A}_i \in \mathbb{R}^{m_i \times r_i}$ and $\mathbf{B}_i \in \mathbb{R}^{r_i \times n_i}$; each set in $\{\mathbf{A}_i\}_{i=1}^4, \{\mathbf{B}_i\}_{i=1}^4$ is of full rank r_i and mutually non-overlapping. Under this assumption, if each block $\mathbf{A}_i\mathbf{B}_i$ is spanned by r_i, then the rank of $\Delta\mathbf{W}_l$ is $\sum_{i=1}^4 r_i$.

E Column/Row Clustering and Reordering on Factorized $\Delta\mathbf{W}$

Accelerating SVD in Spectral Bi-Clustering. For computational convenience and scalability, clustering or reordering of the rows and columns in $\Delta\mathbf{W} \in \mathbb{R}^{n\times n}$ can be equivalently implemented on the low-rank representation $\Delta\mathbf{W} = \mathbf{AB}$, where the choice of A and B can be referred to Appendix A. Spectral bi-clustering reordering algorithms typically rely on the Singular Value Decomposition (SVD) of $\Delta\mathbf{W}$ or its derived relational matrices $\mathbf{S}$ (as shown in Eq. 3). Performing SVD directly on the $n \times n$ matrix $\mathbf{S}$ incurs a computational complexity of $O(n^3)$. Leveraging the low-rank structure of $\mathbf{A}$ and $\mathbf{B}$, we derive an efficient solution as follows: First, we compute the QR decomposition of $\mathbf{A}$ and $\mathbf{B}^\top$: $\mathbf{A} = \mathbf{Q}_A\mathbf{R}_A$ and $\mathbf{B}^\top = \mathbf{Q}_B\mathbf{R}_B$, where $\mathbf{Q}_A, \mathbf{Q}_B \in \mathbb{R}^{n\times r}$ are column-orthogonal matrices, and $\mathbf{R}_A, \mathbf{R}_B \in \mathbb{R}^{r\times r}$ are upper triangular matrices. This yields $\mathbf{B} = \mathbf{R}_B^\top\mathbf{Q}_B^\top$. Next, we construct a small matrix $\mathbf{M} = \mathbf{R}_A\mathbf{R}_B^\top \in \mathbb{R}^{r\times r}$ and compute its SVD: $\mathbf{M} = \mathbf{U}_M\mathbf{\Sigma}\mathbf{V}_M^\top$. Finally, the singular vectors are synthesized as $\mathbf{U} = \mathbf{Q}_A\mathbf{U}_M$ and $\mathbf{V} = \mathbf{Q}_B\mathbf{V}_M$, with singular values in $\mathbf{\Sigma}$.

In practical spectral bi-clustering, a global offset is introduced to ensure that the $\Delta\mathbf{W}$ matrix is non-negative. This corresponds to a rank-one perturbation, which can be expressed as the outer product of two vectors with uniform entries. This term can be absorbed into $\mathbf{A}$ and $\mathbf{B}$, yielding the low-rank factorization of $\Delta\mathbf{W}$ used in spectral bi-clustering.

Accelerating Cosine Similarity Computation. When calculating cosine similarity directly based

on $\Delta\mathbf{W}$, we need to compute the inner product of a reference row $\mathbf{r}_{\text{ref}}$ and all rows in $\Delta\mathbf{W}$, and then divide each inner product by the norm of the corresponding row in $\Delta\mathbf{W}$ (since the norm of r remains constant).

Next we show how to compute the inner product scores and $\Delta\mathbf{W}$'s row-norms efficiently using the factorized form $\Delta\mathbf{W} = \mathbf{AB}$. Using $\mathbf{r}_{\text{ref}} \cdot \Delta\mathbf{W} = (\mathbf{r}_{\text{ref}} \cdot \mathbf{A}) \cdot \mathbf{B}$, the time cost of computing the row inner products can be reduced to linear in n. For the row norms of $\Delta\mathbf{W}$, they can be computed as $\text{diag}\left(\mathbf{A}\left(\mathbf{B}\mathbf{B}^\top\right)\mathbf{A}^\top\right)$. Therefore, if we pre-compute $\mathbf{B}\mathbf{B}^\top$, then we can recover the norms of the n rows in $\Delta\mathbf{W}$ by $\mathbf{A}_{[i,:]}(\mathbf{B}\mathbf{B}^\top)\mathbf{A}_{[i,:]}^\top$ for $i = 1, 2, \cdots, n$, whose complexity is again linear in n. In both cases, the complexity is quadratic with the rank of $\mathbf{A}$ and $\mathbf{B}$.

F Algorithm of CPS-LoRA Training

Algorithm 1 CPS-LoRA Training

Input: Pretrained weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$
Parameter: Grid number k, rank r, epochs T
Output: Final update $\Delta\mathbf{W}$

Randomly partition $\mathbf{W}$ into $k \times k$ balanced blocks $\mathbf{G}_l$ ($\mathbf{A}_l \leftarrow$ `Kaiming_init` ,$\mathbf{B}_l \leftarrow 0$)
for epoch $t = 1$ to T **do**
  if $t >= 2$ **then**
    Merge sub-block updates into $\Delta\mathbf{W}_{t-1}$
    Derive reordering indices $(\boldsymbol{\pi}_r, \boldsymbol{\pi}_c)$
    Reorder weight matrix $\mathbf{W}_t \leftarrow (\mathbf{W}_{t-1} + \Delta\mathbf{W}_{t-1})[\boldsymbol{\pi}_r, :][:, \boldsymbol{\pi}_c]$
    Partition the reordered $\mathbf{W}_t$ into blocks
    Reinitialize low-rank factors $\mathbf{A}_l$ and $\mathbf{B}_l$
  end if
  for each training batch $(\mathbf{x}, \hat{\mathbf{y}})$ **do**
    Reorder input features: $\mathbf{x}' = \mathbf{x}[:, \boldsymbol{\pi}_r]$
    Compute $\mathbf{y}'$ based on Eq. (8)
    Restore output order: $\mathbf{y} = \mathbf{y}'[:, \boldsymbol{\pi}_c^{-1}]$
    Compute loss $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ and update $\mathbf{A}_l, \mathbf{B}_l$
  end for
end for
Merge sub-block updates into $\Delta\mathbf{W}'$
Restore ordering: $\Delta\mathbf{W} = \Delta\mathbf{W}'[\boldsymbol{\pi}_r^{-1}, :][:, \boldsymbol{\pi}_c^{-1}]$
return $\Delta\mathbf{W}$

G Experimental Setups

The hyperparameter configurations for all tasks, including both CPS-LoRA and competing methods, are detailed in separate tables: Table 7 for LLaMA2-7B, Table 8 for LLaMA3-8B-Instruct, Table 9 for Qwen2.5-1.5B, and Table 10 for LLaMA2-13B. To ensure experimental reproducibility and fair comparison, all reported results are obtained from a single run with a fixed random seed of 42, following common practice in Natural Language Processing, and all experiments are conducted on a single NVIDIA A6000 GPU (48GB).

Task	Commonsense			Mathematical		
Method	CPS-LoRA	PiSSA	Other	CPS-LoRA	PiSSA	Other
LR	1e-4	2e-4	1e-4	3e-4	4e-4	3e-4
LR Scheduler	Linear					
Optimizer	AdamW					
Batch size	4					
Warmup Steps	500					
Epochs	2					
Rank	8					

Table 7: Hyperparameter configurations of all methods on LLaMA2-7B across commonsense and mathematical reasoning tasks.

Task	Commonsense			Mathematical			Code	
Method	CPS-LoRA	PiSSA	Other	CPS-LoRA	PiSSA	Other	CPS-LoRA	Other
LR	4e-5	2e-4	1e-4	4e-5	2e-4	1e-4	4.5e-5	5e-5
LR Scheduler	Linear							
Optimizer	AdamW							
Batch size	4							
Warmup Steps	500							
Epochs	2							
Rank	8							

Table 8: Hyperparameter configurations of all methods on LLaMA3-8B-Instruct across commonsense reasoning, mathematical reasoning and code generation tasks.

Task	Commonsense			Mathematical		Code			
Method	CPS-LoRA	PiSSA	Other	CPS-LoRA	Other	CPS-LoRA	PiSSA	GraLoRA	Other
LR	1e-4	2e-4	1e-4	5e-5	3e-4	6e-5	4.5e-5	5e-5	4e-5
LR Scheduler	Linear								
Optimizer	AdamW								
Batch size	4								
Warmup Steps	500								
Epochs	2								
Rank	8								

Table 9: Hyperparameter configurations of all methods on Qwen2.5-1.5B across commonsense reasoning, mathematical reasoning and code generation tasks.

Task	Commonsense			Mathematical	
Method	CPS-LoRA	PiSSA	Other	CPS-LoRA	Other
LR	1e-4	2e-4	1e-4	1e-4	3e-4
LR Scheduler	Linear				
Optimizer	AdamW				
Batch size	4				
Warmup Steps	500				
Epochs	2				
Rank	8				

Table 10: Hyperparameter configurations of all methods on LLaMA2-13B across commonsense reasoning and mathematical reasoning tasks.

Domain	Train Dataset	#Number	Eval Dataset	#Number	Answer
Commonsense	Commonsense170K (Hu et al., 2023)	170K	BoolQ (Clark et al., 2019)	3270	Yes/No
			PIQA (Bisk et al., 2020)	1830	Option
			Social IQA (Sap et al., 2019)	1954	Option
			ARC-C (Clark et al., 2018)	1172	Option
			ARC-E (Clark et al., 2018)	2376	Option
			OpenbookQA (Mihaylov et al., 2018)	500	Option
			HellaSwag (Zellers et al., 2019)	10042	Option
			Winogrande (Sakaguchi et al., 2021)	1267	Option
Mathematical	Math10K (Hu et al., 2023)	10K	MultiArith	600	Number
			GSM8K (Cobbe et al., 2021)	1319	Number
			AddSub (Hosseini et al., 2014)	395	Number
			AQuA (Ling et al., 2017)	254	Option
			SingleEq (Koncel-Kedziorski et al., 2015)	508	Number
			SVAMP (Patel et al., 2021)	1000	Number
			MAWPS (Koncel-Kedziorski et al., 2016)	238	Number
Code	CodeAlpaca20K (Chaudhary, 2023)	20K	MBPP (Austin et al., 2021)	257	Code
			HumanEval (Chen et al., 2021)	164	Code

Table 11: Dataset statistics for commonsense reasoning, mathematical reasoning, and code generation tasks.

H Dataset Statistics

Table 11 summarizes the datasets used in our experiments across different domains, including commonsense reasoning, mathematical reasoning, and code generation tasks. All scientific artifacts employed in this study adhere to the licenses specified in their original publications or websites and are utilized exclusively for research purposes.

I Extended Spectral and EBER Analysis

Figure 7 extends the singular-value spectrum analysis in the main text to each competing method, including LoRA, DoRA, PiSSA, RaSA, GraLoRA, and CPS-LoRA. All methods are evaluated under the same parameter budget, corresponding to two 8×4096 matrices[1]. The spectra are averaged across all 128 update matrices $\Delta\mathbf{W}$ extracted from all 32 layers of LLaMA2-7B. Consistent with the observations reported in Figure 4, global adaptation methods are constrained to a lower algebraic rank, while methods based on localized or partitioned updates attain a higher nominal rank under the same parameter budget.

To quantitatively characterize the differences in spectral distributions observed above, we adopt the Entropy-Based Effective Rank (EBER) as a scalar measure of rank utilization. Given an update matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ with singular values $\{\sigma_i\}_{i=1}^{r}$, where $r = \mathrm{rank}(\mathbf{W})$, the entropy-based effective rank is

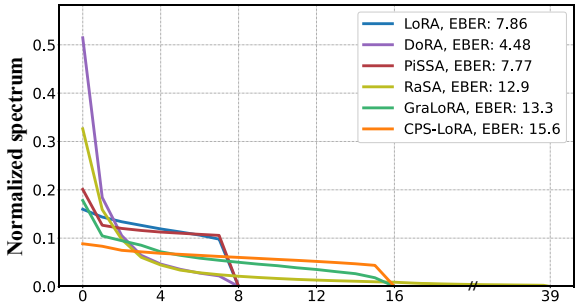


Figure 7: Normalized singular value spectra with entropy-based effective rank (EBER) for all competing methods.

defined as

$$\mathrm{EBER}(\mathbf{W}) = \exp\left(-\sum_{i=1}^{r} p_i \log p_i\right),$$
$$p_i = \frac{\sigma_i}{\sum_{j=1}^{r}\sigma_j}, \quad i = 1,\ldots,r.$$

Intuitively, EBER captures not only the nominal rank of a matrix but also how effectively that rank is utilized. A higher EBER corresponds to a more uniform distribution of spectral energy, whereas a lower EBER indicates that the update is dominated by a small number of singular directions.

Figure 7 plots the normalized singular value spectrum for all competing methods. To avoid visual clutter, we do not plot the standard deviation bands. As can be seen, global adaptation methods such as LoRA, DoRA and PiSSA generally yield relatively low EBER values with an upper bound of 8, reflecting limited rank utilization under a fixed parameter budget.

[1]RaSA has 39 columns for each $\mathbf{A}$ and $\mathbf{B}$ due to a global parameter sharing scheme, see discussion below.

Method	Mathematical			Commonsense		
	Seed1	Seed2	Seed3	Seed1	Seed2	Seed3
LoRA	68.7	67.4	67.9	80.1	80.3	79.8
DoRA	68.2	67.9	68.6	79.7	80.2	79.7
PiSSA	68.0	68.3	69.2	79.9	79.5	80.6
RaSA	67.7	68.2	68.9	79.2	79.1	79.7
GraLoRA	67.8	67.7	68.2	78.7	79.2	79.1
CPS-LoRA	**69.5**	**69.3**	**70.1**	**81.9**	**81.8**	**81.6**

Table 12: Results across three random seeds on math and commonsense reasoning tasks using LLaMA2-7B.

For local adaptation methods such as GraLoRA and CPS-LoRA (ours), they consistently attain higher effective ranks than global methods, indicating a broader utilization of the available subspace. Notably, CPS-LoRA (with clustered subspace partitions) consistently attains a higher EBER than GraLoRA (with random partitions), the former approaching the upper-bound 16 and approximately 17.3% higher than that of GraLoRA. This difference directly reflects the flatter singular-value spectrum observed in Figure 7 and provides quantitative evidence that clustering-based subspace partitioning improves rank utilization in practice.

Note that RaSA is a special example in which a larger pool of low-rank factors of dimension 32 are shared across all layers with layer-specific adapter of dimension 7. This parameter reuse strategy effectively improves the dimensionality of each low-rank factor matrices ($\mathbf{A}$ or $\mathbf{B}$) from 8 to 39, thus leading to a higher EBER (12.9) than other global adaptation methods. However, compared with the full algebraic rank 39, the EBER still accounts for a lower proportion of the full algebraic rank. On the other hand, the reuse of low-rank update matrices may lead to higher parameter redundancies that could affect the expressiveness of the model. In the empirical evaluations, the accuracies of RaSA is slightly worse than LoRA on selected benchmarks, according to Table 1, Table 2, Table 3 and studies in the previous literature (Jung et al., 2025).

J Additional Multi-Seed Evaluation

To further examine the statistical stability of the observed improvements, we conduct additional experiments with multiple random seeds. Specifically, we evaluate LoRA and several variants on both mathematical reasoning and commonsense reasoning tasks using LLaMA2-7B. Each method is trained with three different random seeds, and the corresponding accuracies are reported in Table 12. Across all seeds, CPS-LoRA consistently outperforms LoRA and other recent variants on both tasks, suggesting that the observed improvements are not attributable to a favorable random initialization but remain robust across different training runs.

Furthermore, CPS-LoRA exhibits smaller performance variations across seeds. For example, on the commonsense reasoning task, the standard deviation of CPS-LoRA is approximately 0.02, which is noticeably lower than that of the baseline methods (0.06–0.31). This observation indicates that CPS-LoRA leads to more stable optimization behavior and improved robustness in practice.

K Discussion on Distributed Training

CPS-LoRA introduces row and column reordering operations during the subspace clustering stage. A natural question is whether such operations may introduce additional engineering complexity in distributed training scenarios where model parameters are partitioned across devices. In this section, we briefly discuss several practical considerations.

K.1 Computational Overhead

In our single-GPU experiments, the additional operations introduced by CPS-LoRA (mainly subspace clustering and row/column reordering) incur only modest computational overhead. Compared with standard LoRA, the overall training time increases by approximately 8.4%, while the method consistently achieves a 2–3% absolute improvement in accuracy, demonstrating a favorable efficiency–performance trade-off.

K.2 Permutation Implementation Strategy

Importantly, explicitly reordering the rows and columns of the weight matrix $\mathbf{W}$ is not strictly required during training. Instead, an equivalent formulation can be derived by applying permutations only to the input features and the low-rank factors.

Let $\boldsymbol{\pi}_r$ and $\boldsymbol{\pi}_c$ denote the row and column permutation index vectors, and let $\boldsymbol{\pi}_r^{-1}$ and $\boldsymbol{\pi}_c^{-1}$ denote their inverse permutations. Under a naïve permutation scheme, the output of a layer $\mathbf{y} = \mathbf{x}(\mathbf{W} + \Delta\mathbf{W})$ can be written as

$$\mathbf{y} = (\mathbf{x}[:, \boldsymbol{\pi}_r](\mathbf{W}[\boldsymbol{\pi}_r, \boldsymbol{\pi}_c] + \mathbf{AB}))[:, \boldsymbol{\pi}_c^{-1}],$$

which can be equivalently transformed into

$$\mathbf{y} = \mathbf{xW} + (\mathbf{x}[:, \boldsymbol{\pi}_r]\mathbf{AB})[:, \boldsymbol{\pi}_c^{-1}].$$

This formulation avoids explicitly permuting the full weight matrix $\mathbf{W}$ and instead applies permutation operations only to the input feature vector

$\mathbf{x}$ and the low-rank factors $\mathbf{A}$ and $\mathbf{B}$. Since these tensors are significantly smaller than $\mathbf{W}$, the associated computational and memory overhead is substantially reduced.

K.3 Implications for Distributed Training

In distributed fine-tuning scenarios, communication overhead is an inherent challenge common to most LoRA-style methods. Under the permutation formulation described above, CPS-LoRA does not require explicit reshuffling of the full weight matrix. Instead, only the low-rank adaptation parameters and the corresponding permutation index vectors need to be synchronized. Since these components are negligible compared to the full model parameters, the additional communication cost is expected to be small. Therefore, CPS-LoRA can be integrated into standard distributed fine-tuning pipelines with minimal additional overhead.

L Computation of Subspace distance

The computation of subspace angles serves to quantify the directional dissimilarity between different sub-blocks, thereby evaluating whether a given partitioning or reordering strategy effectively promotes relative independence across blocks. Specifically, we perform singular value decomposition (SVD) on each sub-block and extract the top k dominant singular vectors to form a low-dimensional orthonormal basis that represents the principal directions in either the row or column space. The principal angles between every pair of sub-blocks are then computed using the `subspace_angles` function from the SciPy library. This function derives the angles by computing the singular values of the product of the two subspace basis matrices; larger angles indicate greater independence between the subspaces. To comprehensively assess block-wise independence, we compute angles using both the left singular vectors (row space) and the right singular vectors (column space), and retain the larger of the two as the final subspace angle. When the partitioning is effective and structural information is well localized, we expect large angles between sub-blocks, indicating that their dominant directions are substantially different—thus reflecting stronger subspace independence. This metric facilitates comparison across different partitioning granularities (e.g., 2×2, 4×4, 8×8) in terms of their ability to reveal localized structure.