

# PMSS: Pretrained Matrices Skeleton Selection for LLM Fine-tuning

Qibin Wang<sup>1,4</sup>, Xiaolin Hu<sup>2,4</sup>, Weikai Xu<sup>3,4</sup>, Wei Liu<sup>4</sup>, Jian Luan<sup>4</sup>, Bin Wang<sup>4</sup>,

<sup>1</sup>Peking University, <sup>2</sup>Gaoling School of Artificial Intelligence, Renmin University of China,

<sup>3</sup>University of Electronic Science and Technology of China, <sup>4</sup>XiaoMi AI Lab,

Correspondence: wangqibin@stu.pku.edu.cn, wangbin11@xiaomi.com

## Abstract

Low-rank adaptation (LoRA) and its variants have recently gained much interest due to their ability to avoid excessive inference costs. However, LoRA still encounters the following challenges: (1) Limitation of low-rank assumption; and (2) Its initialization method may be suboptimal. To this end, we propose PMSS(Pre-trained Matrices Skeleton Selection), which enables high-rank updates with low costs while leveraging semantic and linguistic information inherent in pre-trained weight. It achieves this by selecting skeletons from the pre-trained weight matrix and only learning a small matrix instead. Experiments demonstrate that PMSS outperforms LoRA and other fine-tuning methods across tasks with much less trainable parameters. We demonstrate its effectiveness, especially in handling complex tasks such as DROP benchmark(+3.4%/+5.9% on LLaMA2-7B/13B) and math reasoning(+12.89%/+5.61%/+3.11% on LLaMA2-7B, Mistral-7B and Gemma-7B of GSM8K). The code and model will be released soon.

## 1 Introduction

Large language models (LLMs) have demonstrated exceptional capabilities across a wide range of natural language processing (NLP) tasks (Radford et al., 2019). The pre-training stage provides LLMs with foundational abilities for general tasks, but fine-tuning is typically required to better adapt them to specific downstream tasks (Dai and Le, 2015). Full fine-tuning, though effective in unlocking the potential of LLMs, is resource-intensive, introducing storage and computation challenges. As the scale of model training data and parameters continues to grow, the expense of full fine-tuning has become increasingly prohibitive, hindering the adoption of LLMs in scenarios where resources are limited.

To address this issue, Parameter-Efficient Fine-Tuning (PEFT) methods have been proposed to re-

duce the time and computation cost for fine-tuning pre-trained models(Houlsby et al., 2019; Hu et al., 2021; Lester et al., 2021; Li and Liang, 2021). Among these methods, LoRA(Hu et al., 2021) has gained particular success for its effectiveness and simplicity without altering model architecture or introducing any inference latency. However, LoRA still faces two fundamental limitations: first, its low-rank assumption may not generalize well to complex tasks, and second, its initialization method can result in slower or suboptimal convergence.

Recent studies have found that LoRA’s efficacy diminishes empirically in some complex tasks, especially those that differ from the pre-training dataset compared with full fine-tuning(Biderman et al., 2024). This phenomenon is hypothesized to stem from the inherent low-rank assumption underlying LoRA(Ji et al., 2024), which posits that an update of weight during fine-tuning occurs within a low-rank subspace and can be well-approximated by a low-rank matrix production. Chen et al. (2024) and Jiang et al. (2024) have evaluated the generalizability of the low-rank assumption, showing that specific complex tasks typically exhibit a higher intrinsic rank. Other researchers have focused on LoRA’s initialization method, where adapter matrix  $B$  is initialized with zeros and matrix  $A$  with Gaussian noise. PiSSA(Meng et al., 2024) and its follow-up works(Bařazy et al., 2024; Lingam et al., 2024; Wang et al., 2024; Yang et al., 2024), usually have employed low-rank approximations of the original pretrained matrices, such as low-rank SVD approximation, to initialize adapter matrices in LoRA. These studies demonstrate that alternative initialization methods can improve performance across different models and datasets(Mao et al., 2024). The success of these methods highlights the suboptimality in LoRA, further indicating that pretrained matrices contain rich semantic content highly pertinent to various downstream tasks. However, these works have not further examined the pre-trained

matrices themselves.

To address these challenges, we consider the two key factors simultaneously: (1) Overcome the limitations of low-rank assumption. Even in resource-constrained environments, it is essential to enable high-rank updates during fine-tuning to gain an advantage in handling more complex tasks, such as mathematical reasoning. (2) Leverage the semantic and linguistic information inherent in pretrained matrices rather than initialization in Gaussian noise or zeros, bridging the gap between the pre-training and fine-tuning stages.

To this end, we propose PMSS (**P**re-trained **M**atrices **S**keleton **S**election), a novel parameter efficient fine-tuning method designed to enhance the parameter efficiency of large language model while leveraging the intrinsic semantic structure of pre-trained matrices. As illustrated in Figure 1, by carefully selecting the row and column skeletons and freezing them during training, we ensure that the updates occur within the subspace spanned by these components. As hypothesized by ReFT (Wu et al., 2024), pre-training is likely the crucial stage in endowing models with capabilities, while instruction tuning acts merely as a form of style transfer. We select three representative and challenging downstream NLP tasks, commonsense reasoning, the challenging English reading comprehension benchmark, DROP, which requires discrete reasoning over paragraphs, and mathematical reasoning to demonstrate the efficacy in handling complex tasks. Our experiments demonstrate empirically that after pre-training on extensive and diverse datasets, over-parameterized models have already been positioned into a subspace that captures a wide range of linguistic and semantic patterns, meaning that only minimal adjustments are needed to adapt the model to specific downstream tasks.

The summary of our contributions is as follows:

- We introduce a novel fine-tuning method while preserving the intrinsic semantic structure of pre-trained matrices and enabling high-rank updates. Our method further reduces the number of trainable parameters compared to the state-of-the-art LoRA.
- We compare our method with LoRA and other parameter-efficient adaptation methods on the DROP, commonsense reasoning, and math reasoning benchmarks. Our method yields better results compared to LoRA, especially on

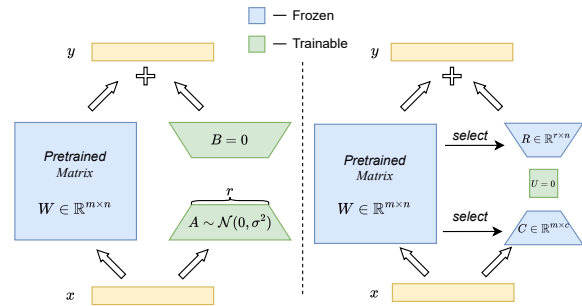


Figure 1: An overview of LoRA and our proposed PMSS. The distinction lies in that PMSS freezes  $C$  and  $R$  and only updates  $U$  during the fine-tuning stage. Note that *select* denotes we select the row and column skeletons from the original pre-trained matrices to construct matrices  $C$  and  $R$ , which ensures update happens in the subspace spanned by skeletons of the original weight. Further,  $C$  and  $R$  can be compactly represented by one-dimensional index vectors.

complex tasks such as DROP(+3.4%/+5.9% on LLaMA2-7B/13B) and math reasoning(+12.89%/+5.61%/+3.11% on LLaMA2-7B, Mistral-7B and Gemma-7B of GSM8K).

- Through our experiments, we demonstrate that fine-tuning happens in tiny subspaces related to subspaces spanned by skeletons of model parameters.

## 2 Related Work

### 2.1 Intrinsic Dimension and Subspace Learning

Li et al. (2018) first introduced the intrinsic dimension of objective landscape and demonstrated different tasks exhibit varying intrinsic dimensions through random subspace training. This work found that many tasks inherently have lower intrinsic dimensions. Following this, Aghajanyan et al. (2021) further elucidated that common pre-trained language models typically exhibit low intrinsic dimensions, with larger models often possessing even lower intrinsic dimensions. Gur-Ari et al. (2018) showed empirically that after a short period of training, the gradient dynamically converges to a tiny subspace, which is preserved over even long periods of training.

Based on intrinsic dimension and subspace learning, a series of methods (Gressmann et al., 2020; Li et al., 2022a,b; Gauch et al., 2022; Zhang et al., 2023) are proposed. Most of these works rely on either random projections or sampling from optimization trajectories to extract subspaces, allowing

large-scale model training within a tiny subspace.

## 2.2 Column Subset Selection, CUR and Interpolative Decomposition

Column Subset Selection Problem(CSSP) has been extensively studied within the theoretical computer science community(Boutsidis et al., 2009; Deshpande and Rademacher, 2010; Tropp, 2009; Altschuler et al., 2016), revolves around selecting a small subset of representative column skeletons of a matrix. The goal of CSSP is to identify column skeletons covering column space and capturing the essential information of the original matrix. Compared with SVD, CSSP provides a more interpretable way while preserving the underlying structure, such as sparsity and non-negativity.

Inspired by core idea of CSSP, one way to achieve low-rank matrix approximation leverages the self-expression of data, which is the notion that data is better represented by other data points rather than an abstract set of bases(Hamm and Huang, 2020). There are two representative methods: CUR Decomposition(also named Skeleton Decomposition)(Mahoney and Drineas, 2009) and Interpolative Decomposition(ID)(Cheng et al., 2005). We will formally define them in Section 3.

## 2.3 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) methods typically only train a small fraction of parameters while keeping the vast majority of parameters frozen to adapt large-scale models to downstream tasks. LoRA(Hu et al., 2021) has merged as a prominent fine-tuning technique of large pre-trained models, offering a computation and memory-efficient alternative to full fine-tuning. PiSSA(Meng et al., 2024) initializes the adapter matrices using a low-rank SVD of the original weights and only updates principal singular components. LoRA-XS(Balazy et al., 2024) performs a basis adaptation for frozen principal singular values and vectors. Concurrent with our work, CURLoRA(Fawi, 2024) adopted a CUR-modified method but involves the random selection of row and column skeletons with smaller norms to mitigate catastrophic forgetting. We will elucidate a comparison in detail in Section 4.4.

## 3 Preliminary

In this section, we will first provide a concise overview of pivoted QR factorization, Interpolative Decomposition (ID), and CUR decomposition.

Then, we will introduce the CUR-ID algorithm, which forms the basis of our proposed method.

**Pivoted QR factorization.** For a given matrix  $W \in \mathbb{R}^{m \times n}$  with real or complex entries, and set  $m \geq n$  without loss of generality.  $W(:, J)$  denotes the submatrix of  $W$  consisting of columns from  $W$  indexed by  $J$ .  $W(K, J)$  denotes the submatrix obtained by rows and columns from  $W$  indexed by  $K$  and  $J$  respectively(Golub and Van Loan, 2013). The (compact) QR factorization then takes the form

$$W \begin{matrix} P \\ \end{matrix} = \begin{matrix} Q \\ \end{matrix} \begin{matrix} R \\ \end{matrix}, \quad (1)$$

where  $P$  is a permutation matrix,  $Q$  is an orthogonal matrix and  $R$  is upper triangular. The permutation matrix  $P$  can be represented via a vector  $J \subset [n]$  of indices that  $P = I_n(:, J)$  where  $I_n$  is the  $n \times n$  identity matrix.

The QR-factorization is often computed via column pivoting, which results in factor  $R$  satisfying various decay conditions(Golub and Van Loan, 2013), such as:

$$R(k, k)^2 \geq \sum_{i=k}^j R(i, j)^2, \quad (2)$$

where  $j \geq k + 1$ , and  $R(i, j)$  denotes the element in the  $i$ -th row and  $j$ -th column of  $R$ .

**Interpolative decomposition(ID) and CUR decomposition.** Generally, CUR decomposition approximates matrix  $W$  by a product of three matrices  $C$ ,  $U$  and  $R$ , where matrices  $C$  and  $R$  consist of subset columns and rows from  $W$  and  $U$  is a small carefully constructed matrix to minimize the low-rank approximation error. Similarly, ID approximates a matrix  $W$  as a product of a matrix  $C$  consisting a small subset of columns from  $W$  and a coefficient matrix  $X$ .

From skeleton selection standpoint(Dong and Martinsson, 2023), given any arbitrary linearly independent column subset  $C = W(:, J)$  ( $W \in \mathbb{R}^{m \times n}$ ,  $J \subset [n]$ ), the rank- $|J|$  column ID of  $W$  with respect to column skeletons can be formulated as

$$\hat{W}_{*,J} \triangleq C(C^\dagger W), \quad (3)$$

where  $\dagger$  denotes the Moore-Penrose pseudoinverse,  $CC^\dagger$  is the orthogonal projector onto the spanning subspace of column skeletons. Analogously, the rank- $|K|$  column ID of  $W$  with respect to row skeletons can be formulated as

$$\hat{W}_{K,*} \triangleq (WR^\dagger)R, \quad (4)$$

where row subset  $R = W(K, :)(K \subset [m])$ .

With both column and row skeletons, we can construct low-rank approximation in two-sided ID and CUR decomposition. We define  $|K| = |J|$  and  $S \triangleq W(K, J)$  be an invertible two-sided skeleton, such that two-sided ID

$$\hat{W}_{K,J} \triangleq (CS^{-1})S(C^\dagger W), \quad (5)$$

and CUR decomposition

$$\tilde{W}_{K,J} \triangleq C(C^\dagger WR^\dagger)R, \quad (6)$$

where  $S^{-1}$  represents the inverse of  $S$ .

Different from sampling-based methods that draw skeletons from proper probability distributions such as Mahoney and Drineas (2009), which utilized statistical leverage scores originating from statistics, Voronin and Martinsson (2017) proposed a novel CUR-ID algorithm drawing skeleton selections via more deterministic pivoting. They demonstrated that a CUR decomposition could be constructed using a two-sided ID, which can itself be built from pivoted QR factorization. The matrices  $C$  and  $R$  are selected by two successive one-sided IDs. The idea behind this work is that the matrix  $C$  can be directly obtained via ID, and then a subsequent full-rank ID on matrix  $C$  yields an index vector needed to construct matrix  $R$ .

**Notation.** Given a positive integer  $m$ , the notation  $[m]$  is defined as the set of the first  $m$  natural numbers  $\{1, 2, \dots, m\}$ . The notation  $\dagger$  denotes the Moore-Penrose pseudoinverse. In contrast to common conventions in computer science, all indices in this paper, unless otherwise specified, will begin from 1. This choice aligns with certain mathematical traditions and is made for consistency throughout the text. Full notations can be found in Appendix D.

## 4 Methodology

### 4.1 Formulation of PMSS

We present PMSS(Pre-trained Matrices Skeleton Selection), a novel parameter efficient fine-tuning method designed to enhance the parameter efficiency of large language model while preserving linguistic and semantic information. We reparameterize the weight update matrix  $\Delta W$  as the product of three matrices  $C, U$  and  $R$ . Unlike LoRA, where both  $A$  matrix initialized to zero and  $B$  matrix initialized with Gaussian noise are trainable, our algorithm adopts a different way. Once the  $C$  and  $R$  matrices are initialized, they are frozen and no longer

updated. Correspondingly, the  $U$  matrix remains trainable throughout the training stage, which leads to computational efficiency while retaining the relevant structure from the pre-trained weights. PMSS selects the most representative column and row skeletons of pre-trained matrices  $W \in \mathbb{R}^{m \times n}$  to construct matrices  $C \in \mathbb{R}^{m \times c}$  and  $R \in \mathbb{R}^{r \times n}$ . Consequently, we can represent  $C$  and  $R$  through compressed index vectors  $K \subset [m]$  and  $J \subset [n]$  respectively, further enhancing memory efficiency. The overview of PMSS is illustrated in Figure 1.

**Skeleton Selection.** We observe that different subspace initializations can impact the fine-tuning effectiveness of large language models. To capture the underlying structure (or skeleton) of the original weight more deterministically, we employ a two-sided ID algorithm. Initially, we apply a one-sided ID to the original weight matrix  $W$ , identifying the column skeleton (i.e., matrix  $C$ ) by performing a rank- $c$  column-pivoted QR factorization. This yields the row index vector  $J$ , which is used to construct  $C$ . Subsequently, we perform successive one-sided ID on the transpose of matrix  $C$  through a full-rank column-pivoted QR factorization to derive the row skeleton, which forms the matrix  $R$ . Then, we can generate a row index vector  $K$ . Ultimately, we only need to explicitly retain the index vectors. The overall algorithm is summarized in Algorithm 1.

**Forward Pass.** LoRA aims to reparameterize updates  $\Delta W \in \mathbb{R}^{m \times n}$  of pre-trained matrix  $W \in \mathbb{R}^{m \times n}$  in the form of low-rank approximation of adapter matrices  $A \in \mathbb{R}^{r \times n}$  and  $B \in \mathbb{R}^{m \times r}$  with rank  $r \ll \min(m, n)$ . LoRA’s forward pass is:

$$y = W'x = (W + BA)x, \quad (7)$$

where  $x \in \mathbb{R}^n$  is the input for the current layer, and  $y \in \mathbb{R}^m$  is the output of the current layer and pre-activation input for the next layer. Both  $A$  and  $B$  matrices are trainable.

Our proposed forward pass is:

$$\begin{aligned} y &= W'x = (W + CUR)x, \\ C &= W(:, J), R = W(K, :), \end{aligned} \quad (8)$$

where  $K \in \mathbb{R}^r, K \subset [m], J \in \mathbb{R}^c, J \subset [n]$ .  $K$  and  $J$  are compressed index vectors, where each scalar value represents a selected row or column from the original pre-trained matrix.  $C$  and  $R$  are frozen after selection and  $U$  retains trainable during training stage. We initialize the  $U$  with zero to prevent any weight drift in the beginning. We scale  $\Delta Wx$  by  $\frac{\alpha}{\max\{c, r\}}$ , where  $\alpha$  is a constant in  $c, r$ .

---

**Algorithm 1** PMSS Algorithm

---

**Input:** Pretrained matrix  $W \in \mathbb{R}^{m \times n}$ .**Parameter:** Column number parameter  $c$ , row number parameter  $r$ ,  $\min(m, n) \geq c \geq r$  WLOG.**Output:** Column index set  $J$  and row index set  $K$ .

- 1: Perform a rank  $c$  column pivoted QR factorization to get  $WP := QR$
  - 2: Define the column ordered index set  $J$  via  $I(:, J) = P$
  - 3: Define an interpolation matrix  $C := W(:, J(1:c))$
  - 4: Perform a full rank column pivoted QR factorization to get  $C^T P^* := Q^* R^*$
  - 5: Define the row ordered index set  $K$  via  $I(:, K) := P^*$
  - 6: Partition  $J := J(1:c)$ ,  $K := K(1:r)$
  - 7: **Return** column index set  $J$  and row index set  $K$
- 

## 4.2 Fine-tuning Happens in Constraining Skeleton Subspaces

In this subsection, we will compare the differences in gradient updates between LoRA and PMSS. Based on Equation 7, during back-propagation, the gradient of weight matrix  $W$  is:

$$\nabla_{W'} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial y} x^T, \quad (9)$$

where  $\mathcal{L}$  is the upstream loss and  $\frac{\partial \mathcal{L}}{\partial y}$  denotes the partial derivative of  $\mathcal{L}$  with respect to  $y$ .

In LoRA, the adapter matrices  $A$  and  $B$  are both trainable, and the gradients for these are computed separately as follows:

$$\frac{\partial \mathcal{L}}{\partial A} = B^T \nabla_{W'} \mathcal{L}, \quad \frac{\partial \mathcal{L}}{\partial B} = \nabla_{W'} \mathcal{L} A^T, \quad (10)$$

From the above equations, it is evident that the gradient computations for the matrices  $A$  and  $B$  in LoRA are mutually coupled and continuously evolving. However, in PMSS, the only trainable matrix is  $U$ , and its gradient can be computed as follows based on Equation 8:

$$\frac{\partial \mathcal{L}}{\partial U} = C^T \nabla_{W'} \mathcal{L} R^T, \quad (11)$$

We then find PMSS update matrix  $U$  with SGD for every step  $t$  by

$$U_{t+1} \leftarrow U_t - \eta C^T \nabla_{W'} \mathcal{L}_t R^T, \quad (12)$$

where  $\eta$  is the learning rate. Therefore, putting it to Equation 8, we reparameterize  $\Delta W$  by

$$\Delta W = -\eta C C^T \left( \sum_{t=1}^T \nabla_{W'} \mathcal{L}_t \right) R^T R, \quad (13)$$

Let  $\sum_{t=1}^T \nabla_{W'} \mathcal{L}_t$  be  $M_T$  for convenience. This equation indicates the entire update is confined to the subspace spanned by  $C$  and  $R$ . Although the matrices  $C$  and  $R$  are typically not orthogonal and thus  $C C^T$  and  $R^T R$  do not form strict projection matrices. However,  $C C^T$  still constrains the rows of  $M_T$  to the column space of  $C$ , and  $R^T R$  constrains the columns of  $M_T$  to the row space of  $R$ . Since  $C$  and  $R$  are selected from matrix  $W$ , the update is effectively confined to a constrained subspace spanned by the row and column skeletons of  $W$ . In contrast, FLoRA (Hao et al., 2024) demonstrates that the update  $\Delta W$  in the vanilla initialization of LoRA can be approximated as:

$$\Delta W \approx -\eta \left( \sum_{t=1}^T \nabla_{W'} \mathcal{L}_t \right) A_0^T A_0, \quad (14)$$

where  $A_0$  is the initialization of adapter matrix  $A$  in LoRA. They reveal that LoRA updates can be viewed as performing random down and up projections to the gradient, whereas PMSS applies projections related to critical subspace of weight to the gradient.

## 4.3 Parameter Efficiency and Low-Cost High-Rank Updates

We demonstrate that our method achieves significant parameter efficiency compared to LoRA. For simplicity, let the number of layers for fine-tuning be  $L_t$  and let the dimension of weights be  $\mathbb{R}^{m \times n}$ . For each layer, LoRA introduces a pair of trainable adapter matrices  $A$  and  $B$ . The total number of trainable parameters  $\Theta_{LoRA}$  in LoRA is determined by the rank  $r_{LoRA}$  of adapter matrices:

$$\Theta_{LoRA} = L_t \times (m + n) \times r_{LoRA}, \quad (15)$$

Similarly, the total number of trainable parameters  $\Theta_{PMSS}$  in PMSS is determined by  $c_{PMSS}$  and  $r_{PMSS}$  of adapter matrix  $U$ :

$$\Theta_{PMSS} = L_t \times c_{PMSS} \times r_{PMSS}, \quad (16)$$

We observe PMSS's parameter efficiency arises from the fact that the number of trainable parameters is independent of the dimensions of the pre-trained weight matrix,  $m$  and  $n$ , which are typically

much larger in the large-scale model compared to the rank  $r_{LoRA}$  in LoRA.

PMSS enables higher-rank updates than LoRA when the budget for trainable parameters is the same. Without loss of generality, we set parameter  $c_{PMSS}$  equal to parameter  $r_{PMSS}$  in PMSS. We further observe the rank of updates in PMSS

$$r_{PMSS} = \sqrt{(m+n) \times r_{LoRA}} \gg r_{LoRA}, \quad (17)$$

when  $r_{LoRA} \ll \min(m, n)$ . Even in resource-intensive environments, PMSS enables high-rank updates without increasing memory and computation costs compared with LoRA.

#### 4.4 Comparison with Other Works

Compared to works such as LoRA-XS or CURLoRA, which only focus on updates at low ranks and typically underperform LoRA, our algorithm emphasizes low-cost, high-rank updates instead. Compared to selecting a set of abstract orthonormal bases via SVD, we argue that choosing elements directly from the original matrix provides greater interpretability. In contrast to our method, CURLoRA focuses on mitigating catastrophic forgetting during the fine-tuning stage. They induce implicit regularization by probabilistically sampling inversely proportional to the row and column norms of the matrix, aiming to deviate from the original weight matrix. Their core idea is to induce implicit regularization by deviating as much as possible from the original pre-trained weight matrix. However, they overlook the self-expressive capability of the weights in large pre-trained language models, treating them merely as a means to enforce a certain form of regularization against LoRA. By selecting less significant features or even noise from the original matrix, their work risks capturing suboptimal or even deficient subspaces within the  $C$  and  $R$ . This can lead to the loss of valuable information crucial for effective fine-tuning, thereby hindering the model’s ability to adapt to new tasks efficiently.

## 5 Experiments

In this section, we conduct a series of experiments on various NLP benchmarks to showcase the efficiency of PMSS. For convenience, we choose hyperparameter  $c$  equal to  $r$  in PMSS for all experimental settings. All experiments are conducted on the NVIDIA H800(80G) GPUs. We will list implementation details and hyperparameters of these experiments in Appendix A and C.

Model	Method	# Params%	$F_1$ Score
LLaMA2 <sub>7B</sub>	FT <sup>†</sup>	100%	59.4
	Series <sup>†</sup>	0.747%	58.8
	Parallel <sup>†</sup>	0.747%	59.0
	LoRA <sup>†</sup> <sub><math>r=8</math></sub>	0.062%	54.0
	LoRA <sup>†</sup> <sub><math>r=32</math></sub>	0.249%	54.8
	LoRA <sup>†</sup> <sub><math>r=128</math></sub>	0.996%	56.2
	CURLoRA <sub><math>c,r=128</math></sub>	0.016%	54.1
	CURLoRA <sub><math>c,r=256</math></sub>	0.062%	58.5
	CURLoRA <sub><math>c,r=512</math></sub>	0.248%	59.3
	CURLoRA <sub><math>c,r=640</math></sub>	0.388%	58.9
	PMSS <sub><math>c,r=128</math></sub>	0.016%	55.1
	PMSS <sub><math>c,r=256</math></sub>	0.062%	<b>59.6</b>
	PMSS <sub><math>c,r=512</math></sub>	0.248%	<b>59.6</b>
	PMSS <sub><math>c,r=640</math></sub>	0.388%	59.3
LLaMA2 <sub>13B</sub>	LoRA <sup>†</sup> <sub><math>r=8</math></sub>	0.050%	61.0
	CURLoRA <sub><math>c,r=128</math></sub>	0.010%	64.3
	PMSS <sub><math>c,r=128</math></sub>	0.010%	<b>66.9</b>

Table 1: Benchmark of various fine-tuning methods on the DROP dataset using LLaMA2 7B/13B models as the base model. We report  $F_1$  score as metric and higher score is better. All results with <sup>†</sup> are taken from Chen et al. (2024).

### 5.1 DROP Benchmark

**Datasets and Models.** We first conduct experiments on the DROP dataset (Dua et al., 2019), a challenging English reading comprehension benchmark that requires models to perform discrete reasoning over paragraphs. Through experiments using LoRA with varying ranks and subspace similarity analysis, Chen et al. (2024) demonstrate that DROP is a representative example of a higher intrinsic rank dataset compared to other NLP datasets, such as RTE dataset (Wang et al., 2019). We evaluate PMSS against LoRA and several baseline methods which include full fine-tuning, Series adapter (Series) (Houlsby et al., 2019), Parallel adapter (Parallel) (He et al., 2022) and CURLoRA (Fawi, 2024) by fine-tuning LLaMA-2 7B/13B (Touvron et al., 2023).

**Results.** As shown in Table 1, PMSS consistently outperforms other fine-tuning methods. We observe that PMSS achieves performance comparable to and in some cases exceeding, full fine-tuning while training only a small fraction of the parameters. This demonstrates the effectiveness of our method’s high-rank updates. To investigate the performance of these methods with the ranks scaling, we conducted experiments across varying ranks of PMSS and CURLoRA on LLaMA2-7B. As illustrated in Figure 2, we present a line graph depicting the variation of  $F_1$  score with respect to ranks (i.e., the number of trainable parameters) for LoRA,

Model	Method	#Params(%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	AVG
LLaMA2 <sub>7B</sub>	LoRA <sup>†</sup>	0.83	69.8	79.9	<b>79.5</b>	83.6	<b>82.6</b>	79.8	64.7	<b>81.0</b>	77.6
	CURLoRA	0.62	69.0	80.1	78.4	81.5	77.5	78.4	63.5	75.8	75.5
	PMSS	0.47	<b>70.8</b>	<b>82.8</b>	78.2	<b>88.4</b>	80.9	<b>82.5</b>	<b>68.0</b>	80.2	<b>79.0</b>
LLaMA3 <sub>8B</sub>	LoRA <sup>†</sup>	0.70	70.8	85.2	<b>79.9</b>	91.7	84.3	84.2	71.2	79.0	80.8
	CURLoRA	0.13	73.5	87.5	79.0	94.4	<b>83.4</b>	<b>90.2</b>	78.5	84.6	83.9
	PMSS	0.13	<b>73.8</b>	<b>87.8</b>	78.9	<b>94.6</b>	84.2	89.6	<b>78.7</b>	<b>85.8</b>	<b>84.2</b>

Table 2: Comparison of LLaMA-2 7B and LLaMA-3 8B with various PEFT methods on eight commonsense reasoning datasets. All results with <sup>†</sup> of baseline methods are taken from Liu et al. (2024). We report accuracy as metric and higher score is better.

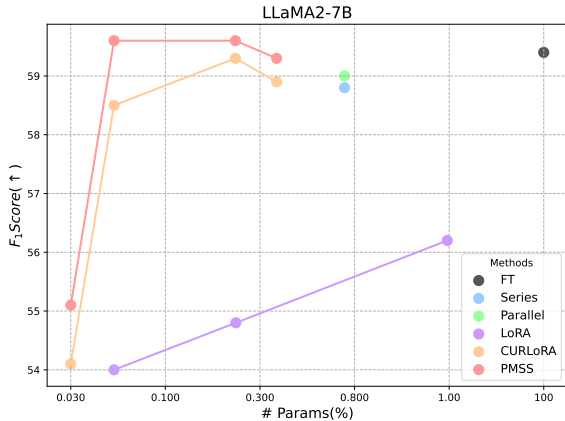


Figure 2: Benchmark of different fine-tuning methods on the DROP dataset. Illustration of the  $F_1$  score (y-axis) with different numbers ratio(%) of trainable parameters (x-axis) using LLaMA2-7B as the base model.

CURLoRA, and PMSS. As shown in the figure, the performance of PMSS, CURLoRA, and LoRA improve as the ranks increase, underscoring both the necessity and effectiveness of high-rank updates. As the rank increases, PMSS exhibits rapid performance gains, achieving superior results compared to other methods, including LoRA and CURLoRA, with minimal trainable parameters (0.062%). Conversely, LoRA only achieves subpar performance, though it improves with increased parameters.

## 5.2 Commonsense Reasoning

**Datasets and Models.** We fine-tune our models on COMMONSENSE170K(Hu et al., 2023), a comprehensive dataset of various commonsense reasoning questions. Eight different commonsense reasoning datasets are used for evaluation, including BoolQ(Clark et al., 2019), PIQA(Bisk et al., 2020), SIQA(Sap et al., 2019), HellaSwag(Zellers et al., 2019), WinoGrande(Sakaguchi et al., 2021), ARC-e, ARC-c(Clark et al., 2018) and OBQA(Ling et al., 2017). All commonsense reasoning tasks are formulated as multiple-choice or Yes/No questions, where the models are required to select the

Model	$q_{proj}$	$k_{proj}$	$v_{proj}$
LLaMA2-7B	$4096 \times 4096$	$4096 \times 4096$	$4096 \times 4096$
LLaMA3-8B	$4096 \times 4096$	$4096 \times 1024$	$4096 \times 1024$

Table 3: Comparison of module dimensions on LLaMA2-7B and LLaMA3-8B.

most appropriate answers and rationales. We evaluate PMSS against LoRA and CURLoRA by fine-tuning LLaMA-2 7B(Touvron et al., 2023) and LLaMA-3 8B(Dubey et al., 2024).

**Results.** The main results are reported in Table 2. PMSS outperforms LoRA and CURLoRA on most metrics for the LLaMA2-7B and LLaMA3-8B models. On LLaMA2-7B, PMSS outperforms LoRA and CURLoRA by 1.4% and 3.5% average accuracy scores. On LLaMA3-8B, PMSS exceeds LoRA and CURLoRA by 3.4% and 0.3% average accuracy scores.

However, PMSS underperforms LoRA on 3 out of 8 evaluation metrics on LLaMA2-7B, and underperforms LoRA on 2 metrics on LLaMA3-8B. It may be due to the inherently lower rank of commonsense reasoning tasks compared to more complex tasks(e.g., math). As a result, LoRA remains competitive in such tasks. Additionally, on LLaMA3-8B, our method performs closely to CURLoRA. As shown in Table 3, the asymmetric structure of weight( $k_{proj}$  and  $v_{proj}$ ) in LLaMA3-8B may make it easier to capture the critical subspace. We explore this point further in the ablation studies.

## 5.3 Math Reasoning

**Datasets and Models.** We train our models on MetaMathQA dataset(Yu et al., 2023), which comprises 395K samples augmented from other math instruction tuning datasets such as GSM8K(Cobbe et al., 2021) and MATH(Hendrycks et al., 2021), with higher diversity and complexity. We select GSM8K and MATH as the test datasets. We select LLaMA-2 7B(Touvron et al., 2023), Mistral-7B-

Model	Method	Params	GSM8K	MATH
LLaMA2 <sub>7B</sub>	Full FT <sup>†</sup>	6738M	49.05	7.22
	LoRA <sup>†</sup>	320M	42.30	5.50
	PiSSA <sup>†</sup>	320M	53.07	7.44
	CURLoRA	56M	54.51	9.30
	PMSS	56M	<b>55.19</b>	<b>9.74</b>
Mistral <sub>7B</sub>	Full FT <sup>†</sup>	7242M	67.02	18.6
	LoRA <sup>†</sup>	168M	67.70	19.68
	PiSSA <sup>†</sup>	168M	72.86	<b>21.54</b>
	CURLoRA	87.5M	72.40	20.40
	PMSS	87.5M	<b>73.31</b>	21.34
Gemma <sub>7B</sub>	Full FT <sup>†</sup>	8538M	71.34	22.74
	LoRA <sup>†</sup>	200M	74.90	31.28
	PiSSA <sup>†</sup>	200M	77.94	<b>31.94</b>
	CURLoRA	49M	76.65	30.20
	PMSS	49M	<b>78.01</b>	30.60

Table 4: Math reasoning evaluation results for LLaMA2-7B, Mistral-7B and Gemma-7B on math reasoning benchmarks. All results with <sup>†</sup> of baseline methods are taken from Meng et al. (2024). We report accuracy as metric and higher score is better.

Method	#Params(%)	AVG
Random	0.13	83.9
CURLoRA	0.0082	78.6
Random	0.0082	78.9
PMSS	0.0082	<b>79.5</b>

Table 5: Ablation study results on commonsense reasoning using LLaMA3-8B. We also report the result of random selection when the rank is kept the same as in Section 5.2.

v0.1(Jiang et al., 2023) and Gemma-7B(Team et al., 2024) as base models. We evaluate PMSS against full fine-tuning, LoRA, PiSSA, and CURLoRA as baseline methods.

**Results.** Table 4 presents the evaluation results on the GSM8K and MATH benchmarks. The results show that PMSS outperforms LoRA and CURLoRA and even surpasses full fine-tuning with a small fraction of parameters. On LLaMA2-7B, PMSS outperforms PiSSA by +2.12/2.30 on GSM8k and MATH. However, On both the Mistral-7B and Gemma-7B, PMSS outperforms PiSSA on the GSM8K but falls short of PiSSA on the MATH. This suggests that PiSSA’s core idea of utilizing the principal singular components of the weight matrices is effective. Nevertheless, PMSS still performs comparably to PiSSA with much fewer parameters(about 18%-52%), demonstrating the potential scalability of PMSS when handling even complex tasks like math reasoning.

Method	Params	GSM8K	MATH
Random	56M	53.68	9.38
CURLoRA	3.5M	46.70	6.76
Random	3.5M	45.64	7.04
PMSS	3.5M	<b>47.69</b>	<b>7.38</b>

Table 6: Ablation study results on math reasoning using LLaMA2-7B. We also report the result of random selection when the rank is kept the same as in Section 5.3.

## 5.4 Ablation Study

In this subsection, we provide ablation study results to empirically demonstrate how different skeleton selection strategies impact the experimental outcomes. In previous experiments, we set a relatively high rank (e.g., 512). High-rank updates may provide an inherent advantage by enabling the model to learn more effectively. When selecting a sufficient number of rows and columns from the original weight matrix to construct the skeletons, even a random selection method can yield good results, as it effectively covers the row and column spaces’ information of the weight matrix. We use random selection and CURLoRA as a baseline. We adhere to the hyperparameters used in previous experiments, modifying only the learning rate. We also report the results of the random selection method at higher-rank updates, using the exact same hyperparameter settings as in the previous experiments.

### Ablation Study on Commonsense Reasoning.

As shown in Table 5, we present the average scores of the eight commonsense evaluation sets. PMSS consistently outperforms baseline methods at lower ranks. We also report the result of random selection at the same high rank and find it performs closely to CURLoRA and PMSS, which may be attributed to the asymmetric structure of weight matrices on LLaMA3-8B,

**Ablation Study on Math Reasoning.** As shown in Table 6, we present the evaluation result on GSM8K and MATH sets. PMSS consistently outperforms baseline methods across all rank settings.

## 6 Conclusion

In this paper, we introduce PMSS, a novel fine-tuning method designed to enhance parameter efficiency while preserving the semantic and linguistic information contained in the weights. Our method effectively overcomes the low-rank limitation of LoRA and enables high-rank updates at a low cost. Experimental results demonstrate that PMSS out-



performs LoRA and other fine-tuning methods with much less trainable parameters. PMSS is expected to demonstrate superior learning capacity, especially in handling complex tasks.

## 7 Limitation

We have leveraged the inherent world knowledge embedded in the model’s weights. However, to better adapt the model to specific downstream tasks, task-specific knowledge should also be incorporated, which remains for future research. Due to time and objective conditions, it is still unclear whether PMSS is effective for other specific tasks, such as logical reasoning tasks. Additionally, the effectiveness of this method on models with larger parameter scales (e.g. 70B) remains to be verified.

## References

- Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328.
- Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. 2016. Greedy column subset selection: New bounds and distributed algorithms. In *International conference on machine learning*, pages 2539–2548. PMLR.
- Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*.
- Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Christos Boutsidis, Michael W Mahoney, and Petros Drineas. 2009. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM.
- Zhuo Chen, Rumen Dangovski, Charlotte Loh, Owen Dugan, Di Luo, and Marin Soljačić. 2024. Quanta: Efficient high-rank fine-tuning of llms with quantum-informed tensor adaptation. *arXiv preprint arXiv:2406.00132*.
- Hongwei Cheng, Zydrunas Gimbutas, Per-Gunnar Martinsson, and Vladimir Rokhlin. 2005. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404.
- 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.
- 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 preprint arXiv:1803.05457*.

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28.
- Amit Deshpande and Luis Rademacher. 2010. Efficient volume sampling for row/column subset selection. In *2010 IEEE 51st annual symposium on foundations of computer science*, pages 329–338. IEEE.
- Yijun Dong and Per-Gunnar Martinsson. 2023. Simpler is better: a comparative study of randomized pivoting algorithms for cur and interpolative decompositions. *Advances in Computational Mathematics*, 49(4):66.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Muhammad Fawi. 2024. Curlora: Stable llm continual fine-tuning and catastrophic forgetting mitigation. *arXiv preprint arXiv:2408.14572*.
- Martin Gauch, Maximilian Beck, Thomas Adler, Dmytro Kotsur, Stefan Fiel, Hamid Eghbal-zadeh, Johannes Brandstetter, Johannes Kofler, Markus Holzleitner, Werner Zellinger, et al. 2022. Few-shot learning by dimensionality reduction in gradient space. In *Conference on Lifelong Learning Agents*, pages 1043–1064. PMLR.
- Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.
- Frithjof Gressmann, Zach Eaton-Rosen, and Carlo Luschi. 2020. Improving neural network training in low dimensional random bases. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 12140–12150.
- Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. 2018. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*.
- Keaton Hamm and Longxiu Huang. 2020. Perspectives on cur decompositions. *Applied and Computational Harmonic Analysis*, 48(3):1088–1099.
- Yongchang Hao, Yanshuai Cao, and Lili Mou. 2024. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. [Towards a unified view of parameter-efficient transfer learning](#). In *International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- 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 *International conference on machine learning*, pages 2790–2799. PMLR.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- 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*, pages 5254–5276.
- Yuheng Ji, Yue Liu, Zhicheng Zhang, Zhao Zhang, Yuting Zhao, Gang Zhou, Xingwei Zhang, Xinwang Liu, and Xiaolong Zheng. 2024. Advlora: Adversarial low-rank adaptation of vision-language models. *arXiv preprint arXiv:2404.13425*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.
- 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*, pages 3045–3059.
- Chunyuan Li, Heerad Farkhor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.
- Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. 2022a. Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3411–3420.

- Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. 2022b. Subspace adversarial training. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13399–13408. IEEE.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.
- Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. Svft: Parameter-efficient fine-tuning with singular vectors. *arXiv preprint arXiv:2405.19597*.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.
- Michael W Mahoney and Petros Drineas. 2009. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702.
- Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2024. A survey on lora of large language models. *arXiv preprint arXiv:2407.11046*.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- 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. Socialliqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Joel A Tropp. 2009. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 978–986. SIAM.
- Sergey Voronin and Per-Gunnar Martinsson. 2017. Efficient algorithms for cur and interpolative matrix decompositions. *Advances in Computational Mathematics*, 43:495–516.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.
- Hanqing Wang, Zeguan Xiao, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2024. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*.
- Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*.
- Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. 2024. Corda: Context-oriented decomposition adaptation of large language models. *arXiv preprint arXiv:2406.05223*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- 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*, pages 4791–4800.
- Zhong Zhang, Bang Liu, and Junming Shao. 2023. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1701–1713, Toronto, Canada. Association for Computational Linguistics.

## A Implementation Details

### A.1 DROP dataset

**Implementation Details.** We conduct experiments across different ranks and compare PMSS’s performance with LoRA and CURLoRA, highlighting the impact of high-rank updates. We follow the setting of [Chen et al. \(2024\)](#), selecting 2,000 samples from the training set of the DROP dataset as our training set, 800 samples as the validation set, and 1,200 samples from the validation set of DROP as the test set.  $F_1$ -score is used as the evaluation metric for measuring the closeness of the model’s output with the ground truth. The best checkpoint from the validation set is loaded as the final model for testing. [Chen et al. \(2024\)](#) select the number of epoch parameters arbitrarily between 3 and 6. For a fair comparison, we standardize our experiments to run for 3 epochs.

### A.2 Commonsense Reasoning Dataset

**Implementation Details.** We first fine-tune the model on the joint COMMONSENSE170K dataset and evaluate the fine-tuned model on eight downstream commonsense tasks. We follow the setting of [Liu et al. \(2024\)](#), splitting the COMMONSENSE170K dataset in a train set of 170,020 samples and a validation set of 400 samples. We optimize the hyperparameters on the validation set and load the best checkpoint on the validation set as the final model for evaluation. Accuracy is reported as a metric. We use the implementation of LLM-Adapters([Hu et al., 2023](#)) and standardize our experiments to run for 3 epochs.

### A.3 Math Reasoning Dataset

**Implementation Details.** In math reasoning experiments, we use the implementation of PiSSA([Meng et al., 2024](#)) and follow the setting of them while only fine-tuning the learning rate. All models are first trained on a subset containing 100K data points from the MetaMathQA. We load the last checkpoint as the final model for evaluation. Then the fine-tuned models are evaluated on GSM8K and MATH datasets to assess their capabilities in solving mathematical problems as specific downstream tasks. Accuracy is reported on the GSM8K and MATH datasets. All models are fine-tuned for only one epoch.

## B Case Studies

In this section, we present examples of three different tasks to help readers gain a deeper understanding of the specifics of the Three different benchmarks. Additionally, the process output of the Math task(GSM8K) further highlights the validity of the theoretical assumptions discussed in the main text.

### B.1 DROP

#### [An example in DROP]

**Context:**

As of the census of 2000, there were 325,957 people, 149,937 households, and 94,460 families residing in the county. The population density was 570 people per square mile (220/km<sup>2</sup>). There were 182,467 housing units at an average density of 319 per square mile (123/km<sup>2</sup>). The racial makeup of the county was 92.65% Race (United States Census), 4.18% Race (United States Census) or Race (United States Census), 0.22% Race (United States Census), 0.77% Race (United States Census), 0.03% Race (United States Census), 1.14% from Race (United States Census), and 1.02% from two or more races. 4.34% of the population were Race (United States Census) or Race (United States Census) of any race. 89.7% spoke only English language at home; 4.4% spoke the Spanish language, 1.3% German language, and 1.0% French language at home.

**Question:** Which group is smaller for the county according to the census: people or families?

**Answer:** families.

#### [PMSS Reasoning by LLaMA2-7B]

[Prediction: families.] (Right)

[CURLoRA Reasoning by LLaMA2-7B]

[Prediction: people.] (Wrong)

## B.2 ARC-e

[An example in ARC-e]

**Context:**

Below is an instruction that describes a task. Write a response that appropriately completes the request. Instruction: Please choose the correct answer to the question: Copper (Cu) reacts with oxygen (O) to form copper oxide (CuO). The properties of CuO are most likely

**Options:**

**Answer1:** different from copper or oxygen.

**Answer2:** similar to both copper and oxygen.

**Answer3:** similar only to copper.

**Answer4:** similar only to oxygen.

**Question:** Which one is the correct answer? answer1/answer2/answer3/answer4)

**Answer: the correct answer is answer1.**

[PMSS Reasoning by LLaMA3-8B]

[Prediction: the correct answer is answer1.] (Right)

[CURLoRA Reasoning by LLaMA3-8B]

[Prediction: the correct answer is answer2.] (Wrong)

## B.3 GSM8K

[An example in GSM8K]

**Context:**

question: Below is an instruction that describes a task. Write a response that appropriately completes the request. Helga was the fastest clog dancer in all of Slovenia. With both hands at her sides, she could tap her right foot at a rate of 300 taps per minute while simultaneously tapping her left foot at a rate of 250 taps per minute. When she raised her arms, her tap rate slowed down to 200 taps per minute with each foot.

**Question:** If she dances a total of 5 minutes, with her arms raised during only 2 of those minutes, what would be the combined total number of times that she taps both of her feet?

**Answer: 2450.**

[PMSS Reasoning by Gemma-7B]

**Reasoning Process:**

- In the first 3 minutes, Helga taps her right foot at a rate of 300 taps per minute, so she taps her right foot  $300 * 3 = 900$  times.
- In the first 3 minutes, Helga taps her left foot at a rate of 250 taps per minute, so she taps her left foot  $250 * 3 = 750$  times.
- In the first 3 minutes, Helga taps both feet a total of  $900 + 750 = 1650$  times.
- In the next 2 minutes, Helga taps her right foot at a rate of 200 taps per minute, so she taps her right foot  $200 * 2 = 400$  times.
- In the next 2 minutes, Helga taps her left foot at a rate of 200 taps per minute, so she taps her left foot  $200 * 2 = 400$  times.

- In the next 2 minutes, Helga taps both feet a total of  $400 + 400 = 800$  times.
- In the total 5 minutes, Helga taps both feet a total of  $1650 + 800 = 2450$  times.

The answer is 2450

[Prediction: 2450.] (Right)

#### [CURLoRA Reasoning by Gemma-7B]

##### Reasoning Process:

- In the first 3 minutes, Helga dances with her arms at her sides, so she taps each foot at a rate of 300 taps per minute and 250 taps per minute, respectively.
- In the first 3 minutes, she taps her right foot  $3 \text{ minutes} \times 300 \text{ taps per minute} = 900$  times.
- In the first 3 minutes, she taps her left foot  $3 \text{ minutes} \times 250 \text{ taps per minute} = 750$  times.
- In the first 3 minutes, she taps both feet a total of  $900 \text{ taps} + 750 \text{ taps} = 1650$  times.
- In the next 2 minutes, Helga dances with her arms raised, so she taps each foot at a rate of 200 taps per minute..
- In the next 2 minutes, she taps her right foot  $2 \text{ minutes} \times 200 \text{ taps per minute} = 400$  times.
- In the next 2 minutes, she taps her left foot  $2 \text{ minutes} \times 200 \text{ taps per minute} = 400$  times.
- In the next 2 minutes, she taps both feet a total of  $400 \text{ taps} + 400 \text{ taps} = 800$  times.
- In the last minute, Helga dances with her arms at her sides again, so she taps each foot at a rate of 300 taps per minute and 250 taps per minute, respectively.
- In the last minute, she taps her right foot  $1 \text{ minute} \times 300 \text{ taps per minute} = 300$  times.
- In the last minute, she taps her left foot  $1 \text{ minute} \times 250 \text{ taps per minute} = 250$  times.
- In the last minute, she taps both feet a total of  $300 \text{ taps} + 250 \text{ taps} = 550$  times.
- In the last minute, she taps both feet a total of 550 times.

[Prediction: 550.] (Wrong)

## C Hyperparameters

## D Notations

We summarize key notations used in this paper in Table 12.

Hyperparameters	PMSS	CURLoRA
Batch Size		4
Optimizer		AdamW
Scheduler		Linear Scheduler
Weight Decay		0
Dropout		0
Modules		q_proj, v_proj
Number of GPUs		2
Epochs		3
Model	$c, r, \alpha$	Learning Rate
LLaMA2-7B	128-128-128	3e-4
	256-256-256	4e-4
	512-512-512	7e-5
	640-640-640	7e-5
LLaMA2-13B	128-128-128	1e-3

Table 7: Hyperparameters used for DROP dataset for PMSS and CURLoRA on LLaMA2-7B and LLaMA2-13B.

Hyperparameters	LLaMA2-7B	LLaMA2-13B
Batch Size		16
Warmup Steps		100
Optimizer		AdamW
Scheduler		Linear Scheduler
Weight Decay		0
Dropout		0.05
Modules	q_proj, v_proj, k_proj, up_proj, down_proj	
Number of GPUs		1
Epochs		3
$c, r, \alpha$ (PMSS)	448-448-896	256-256-512
Learning Rate(PMSS)	1e-4	1e-4
$c, r, \alpha$ (CURLoRA)	512-512-1024	256-256-512
Learning Rate(CURLoRA)	2e-4	2e-4

Table 8: Hyperparameter configurations of PMSS and CURLoRA for LLaMA2-7B, and LLaMA3-8B on the commonsense reasoning tasks.

Hyperparameters	PMSS	Random	CURLoRA
$c, r, \alpha$		64-64-128	
Learning Rate		7e-4	

Table 9: Ablation study on the commonsense reasoning tasks. We report hyperparameter configurations of PMSS, Random and CURLoRA for LLaMA3-8B.

Hyperparameters	LLaMA2-7B	Mistral-7B	Gemma-7B
Batch Size		128	
Warmup Ratio		0.03	
Optimizer		AdamW	
Scheduler		Cosine	
Weight Decay		0	
Dropout		0	
Modules	q_proj, v_proj, k_proj, o_proj, gate_proj, up_proj, down_proj		
Number of GPUs		1	
Epochs		3	
$c, r, \alpha$	512-512-1024	640-640-1280	512-512-1024
Learning Rate(PMSS)	2e-4	1e-3	7e-4
Learning Rate(CURLoRA)	4e-4	7e-4	7e-4

Table 10: Hyperparameter configurations of PMSS and CURLoRA for LLaMA2-7B, Mistral-7B and Gemma-7B on the math reasoning tasks.

Hyperparameters	PMSS	Random	CURLoRA
$c, r, \alpha$		128-128-256	
Learning Rate	2e-3	1e-3	2e-3

Table 11: Ablation study on the commonsense reasoning tasks. We report hyperparameter configurations of PMSS, Random and CURLoRA for LLaMA2-7B.



<b>Notation</b>	<b>Description</b>
$[m]$	Denotes the set of the first $m$ natural numbers, i.e., $1, 2, \dots, m$ . This is used to simplify expressions involving iterating over the first $m$ elements.
$\dagger$	Denotes the Moore-Penrose pseudoinverse of a matrix.
$W(:, J)$	Denotes the submatrix of matrix $W$ consisting of the columns indexed by the set $J$ , where $J$ is an ordered index set. For example, if $J = \{j_1, j_2, \dots, j_k\}$ , then $W(:, J)$ refers to the matrix consisting of the columns $W(:, j_1), W(:, j_2), \dots, W(:, j_k)$ .
$W(K : J)$	Denotes the submatrix of matrix $W$ obtained by selecting rows indexed by the set $K$ and columns indexed by the set $J$ . If $K = \{k_1, k_2, \dots, k_p\}$ and $J = \{j_1, j_2, \dots, j_q\}$ , then $W(K, J)$ is the matrix formed by taking the intersection of the rows $W(k_1, :), W(k_2, :), \dots, W(k_p, :)$ and the columns $W(:, j_1), W(:, j_2), \dots, W(:, j_q)$ .

Table 12: Table of Notations.