RoseLoRA: Row and Column-wise Sparse Low-rank Adaptation of Pre-trained Language Model for Knowledge Editing and Fine-tuning

Haoyu Wang[†], Tianci Liu[§], Ruirui Li[⋆], Monica Xiao Cheng[⋆], Tuo Zhao[*], and Jing Gao[§]
[†]SUNY Albany, Albany, NY, USA
[§]Purdue University, West Lafayette, IN, USA
[*]Georgia Institute of Technology, Atlanta, GA, USA
[⋆]Amazon, Palo Alto, CA, USA
[†]hwang28@albany.edu, [§]{liu3351,jinggao}@purdue.edu,
[*]tourzhao@gatech.edu , [⋆]{ruirul,chengxca}@amazon.com

Abstract

Pre-trained language models, trained on large-scale corpora, demonstrate strong generalizability across various NLP tasks. Fine-tuning these models for specific tasks typically involves updating all parameters, which is resource-intensive. Parameter-efficient fine-tuning (PEFT) methods, such as the popular LoRA family, introduce low-rank matrices to learn only a few parameters efficiently. However, during inference, the product of these matrices updates all pre-trained parameters, complicating tasks like knowledge editing that require selective updates. We propose a novel PEFT method, which conducts **r**ow and c**o**lumn-wise spar**se** **lo**w-**r**ank **a**daptation (RoseLoRA), to address this challenge. RoseLoRA identifies and updates only the most important parameters for a specific task, maintaining efficiency while preserving other model knowledge. By adding a sparsity constraint on the product of low-rank matrices and converting it to row and column-wise sparsity, we ensure efficient and precise model updates. Our theoretical analysis guarantees the lower bound of the sparsity with respective to the matrix product. Extensive experiments on five benchmarks across twenty datasets demonstrate that RoseLoRA outperforms baselines in both general fine-tuning and knowledge editing tasks.

1 Introduction

Pre-trained language models, trained on extensive and diverse general-domain corpora, exhibit robust generalization capabilities, benefiting various natural language processing (NLP) tasks, such as natural language understanding (Kenton and Toutanova, 2019; Liu et al., 2019) and generation (Touvron et al., 2023; Ouyang et al., 2022). To further adapt these pre-trained models to a specific downstream task, fine-tuning is typically performed. However, these models often comprise numerous parameters, rendering full fine-tuning resource-intensive.

To address this challenge, parameter-efficient fine-tuning (PEFT) methods (Ding et al., 2023b; Han et al., 2024) are proposed. These method introduce a small number of learnable parameters and update only the lightweight introduced parameters during fine-tuning. Among existing methods, LoRA family (Hu et al., 2021; Zhang et al., 2023; Ding et al., 2023b; Liu et al., 2024) has gained remarkable popularity because of its high efficiency and good performance. Conceptually, these LoRA methods add new low-rank matrices to model weights for fine-tuning. Unlike other PEFT methods such as Adapter (Houlsby et al., 2019), LoRA family does not modify the model architecture and is easier to incorporate.

LoRA family has demonstrated notable performance on tasks, such as commonsense reasoning and arithmetic reasoning (Hu et al., 2023; Liu et al., 2024), that mainly rely on a language model's ability to understand and generate text without requiring to modify its internal knowledge explicitly. However, some specialized tasks require updating this internal knowledge. For instance, in knowledge editing (Zhang et al., 2024; De Cao et al., 2021), a language model should incorporate new provided knowledge while preserving other existing knowledge simultaneously. On such tasks, the LoRA family of methods are less-suited due to the coarse-grained control they offer. In particular, the product of the low-rank matrices introduced by LoRA methods is a dense matrix, which is added to the pre-trained model weights during inference. Consequently, all pre-trained parameters are updated, making it challenging to selectively modify specific internal knowledge. This motivates a natural question: ***Is there a PEFT method that can be effectively employed for tasks that require editing the internal knowledge of language models?***

To answer this question, we propose a **r**ow and c**o**lumn-wise spar**se** **lo**w-**r**ank **a**daptation method (RoseLoRA). The motivation is to identify

and update only the most important and influential parameters in the pre-trained model concerning a specific task. In this way, the pre-trained model can be updated effectively with minimal impacts on knowledge that does not require modification. Specifically, RoseLoRA inherits the structure of LoRA to enable parameter-efficient fine-tuning. To selectively fine-tune the most important parameters, we introduce a sparsity constraint, i.e., the ℓ_0 norm, on the product of the low-rank matrices. However, this constraint is non-trivial to optimize. While ℓ_0 norm constraint is widely explored in model pruning (Zhu and Gupta, 2017; Wang et al., 2019; Sun et al., 2023), these methods can only address the sparsity constraint on each low-rank matrix individually. Unfortunately, even if each low-rank matrix is sparse, this does not guarantee that their product will be sparse. To overcome this challenge, we propose converting the original sparsity constraint to row and column-wise sparsity constraints on two low-rank matrices (i.e., $\boldsymbol{B}$ and $\boldsymbol{A}$ in LoRA). We provide a theoretical lower bound of the sparsity of the product of the two low-rank matrices. Furthermore, we propose using a sensitivity-based importance score to incrementally solve the row and column-wise sparsity constraints.

Beyond knowledge editing, the proposed RoseLoRA can also be applied to other general tasks, e.g., commonsense and arithmetic reasoning, instruction following, and natural language understanding. RoseLoRA updates the few most important parameters of the model via enforcing the row or column-wise sparsity for the low-rank matrices, and can match or even outperform LoRA performance with significantly fewer modified parameters.

The contributions are summarized as follows: 1) We propose RoseLoRA, a novel PEFT method that detects and optimizes the most important task-related parameters, resulting in highly precise and effective model updates while being more lightweight than existing methods. 2) We propose a novel row and column-wise sparsity constraint to control the sparsity of the product of two low-rank matrices. Additionally, we provide a theoretical sparsity lower bound for the proposed RoseLoRA. 3) We conduct extensive experiments on five benchmarks covering over twenty datasets. The experiments show that the proposed RoseLoRA can outperform baselines on both general fine-tuning tasks and knowledge editing tasks.

2 Related Works

In this section we provide a concise overview of related works.

2.1 Parameter Efficient Fine-Tuning (PEFT)

PEFT injects a small fraction of trainable parameters into pre-trained large language models (LLMs) to adapt them to downstream tasks. Prefix Tuning (Li and Liang, 2021) prepends soft tokens to the input and learns their continuous embeddings while keeping the original parameters frozen. Adapter (Houlsby et al., 2019; He et al., 2021), on the other hand, inserts lightweight bottleneck neural network modules into the transformer blocks. The third paradigm, LoRA and its variants (Hu et al., 2021; Zhang et al., 2023; Ding et al., 2023a; Dettmers et al., 2024; Li et al., 2023b; Liu et al., 2024), learns low-rank matrices to approximate the desired updates of the original model weights and has achieved state-of-the-art performance. Recently, ReFT (Wu et al., 2024) learns low-rank updates on model representations instead of weights and achieves performance comparable to LoRA with significantly fewer parameters. However, the underlying linear representation hypothesis may not hold valid (Engels et al., 2024), which greatly undermines its generalization ability. In this work, we propose an effective method to learn sparse and low-rank updates on model weights, demonstrating superior performance using as few parameters as ReFT. Recent works such as AdaLoRA (Zhang et al., 2023) and SoRA (Ding et al., 2023a) have applied pruning to LoRA to increase its computational efficiency. However, it is worth mentioning that the proposed RoseLoRA is significantly different from these methods. In particular, these works prunes to control the rank of learned model updates, but the updates are still dense in the sense that all parameters are affected, and cannot offer precise updates as RoseLoRA thereof.

2.2 Knowledge Editing

Knowledge editing seeks to update outdated knowledge in pre-trained LLMs to accommodate a dynamic world. Early efforts involved fine-tuning their parameters directly but suffered from severe forgetting of original knowledge (Wang et al., 2023). For more precise editing, only a minimal amount of parameters should be updated (Wang et al., 2023). This requires sparse parameter updates, which proves NP-hard to solve (Natarajan, 1995). As a workaround, Zhu et al. (2020) used a relaxed L_2 norm constraint on the updates, and

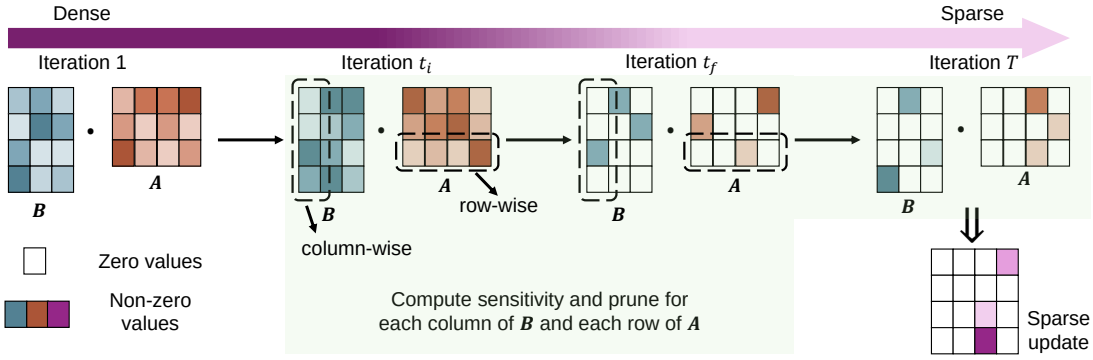

Figure 1: The framework of proposed RoseLoRA.

Huang et al. (2023); Dong et al. (2022) limited the updates to feed-forward network (FFN) layers based on findings that learned knowledge is often stored in these layers (Dai et al., 2021). For further refinement, the locate-and-edit paradigm (Meng et al., 2022a,b) identifies the layer storing specific knowledge and then modifies its parameters. Nonetheless, (Hase et al., 2024) found that updating parameters other than the located ones can also achieve competitive editing performance, questioning the extent to which the more computationally expensive locating process benefits editing.

Alternative solutions restore to external memory without updating original parameters, such as MEND (Mitchell et al., 2021), IKE (Zheng et al., 2023), and SERAC (Mitchell et al., 2022). However, these methods require hard-to-access data to retrieve from (e.g., IKE) or to train extra models on (e.g., MEND and SERAC), which limits their practicality. Recently, LoRA has also been applied for knowledge editing (Wu et al., 2023). However, they do not provide the aforementioned sparsity guarantee, which will be discussed shortly in the next section, so they are less effective and show unsatisfactory performance (Zhang et al., 2024).

3 Preliminary

In this section, we first briefly introduce the low-rank adaptation (LoRA) and then introduce importance-aware pruning.

3.1 Low-rank Adaptation

The LoRA models the efficient incremental update of pre-trained language models via the product of two learnable low-rank matrices. Specifically, the modified weight $\boldsymbol{W}$ can be represented as

$$\boldsymbol{W} = \boldsymbol{W}^o + \Delta = \boldsymbol{W}^o + \boldsymbol{B}\boldsymbol{A}, \tag{1}$$

where $\boldsymbol{W}^o, \Delta \in \mathbb{R}^{d_1 \times d_2}$ are the pre-trained weight matrix and the updated matrix respectively, $\boldsymbol{A} \in \mathbb{R}^{r \times d_2}$ and $\boldsymbol{B} \in \mathbb{R}^{d_1 \times r}$ with $r \ll \min\{d_1, d_2\}$. During fine-tuning, the pre-trained weight $\boldsymbol{W}^o$ is frozen and only lightweight matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ will be updated, which can be formulated as

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{B}\boldsymbol{A}), \tag{2}$$

where $\mathcal{D}$ is the training dataset.

3.2 Sensitivity-based Importance Score for Pruning

Importance-aware pruning (Sanh et al., 2020; Han et al., 2015; Molchanov et al., 2019; Zhang et al., 2022; Li et al., 2023c) aims to identify and set redundant model weights to zero based on estimated importance scores. Parameters with high importance scores are retained, while others are set to zero. Sensitivity (Sanh et al., 2020; Molchanov et al., 2019; Li et al., 2023c) is a popular importance metric that measures the approximate change in training loss when setting a parameter to zero. Formally, the sensitivity with respect to weight $\boldsymbol{W}_{ij}$ is defined by the product of the weight and its corresponding gradient:

$$I(\boldsymbol{W}_{ij}) = |\boldsymbol{W}_{ij} \cdot \nabla_{W_{ij}} \mathcal{L}|. \tag{3}$$

We denote the sensitivity at the t-th iteration based on the current mini-batch as $I^{(t)}$. To reduce the variance of sensitivity, Zhang et al. (2022) proposed to apply exponential moving average for smoothing:

$$\bar{I}^{(t)}(\boldsymbol{W}_{ij}) = \beta \bar{I}^{(t-1)}(\boldsymbol{W}_{ij}) + (1-\beta) I^{(t)}, \tag{4}$$

where β is a hyper-parameter.

4 Methodology

To efficiently fine-tune a pre-trained language model with selective updating, we propose RoseLoRA, a novel LoRA-style fine-tuning framework with sparse adaptation. The framework is

illustrated in Figure 1. We introduce row and column-wise sparsity constraints on the two low-rank matrices, respectively. We theoretically prove that the sparsity lower bound of the product of these low-rank matrices can be guaranteed under these constraints.

4.1 Row and Column-wise Sparse Low-rank Adaptation

We aim to update minimal parameters to enable the model to fit the training data, retain more previous knowledge, and become more lightweight. To achieve this goal, we build on the popular and effective parameter-efficient fine-tuning method LoRA, resulting in the following loss function:

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{B}} \quad & \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{B}\boldsymbol{A}) \\ \text{s.t.} \quad & \frac{\|\boldsymbol{B}\boldsymbol{A}\|_0}{d_1 d_2} \leq \tau, \end{aligned} \tag{5}$$

where τ is the sparsity threshold. However, Eqn. 5 is challenging to handle, with difficulty lie in two-fold. First, the ℓ_0 optimization is NP-hard. Despite that some effective approximate solutions have been proposed (Zhu and Gupta, 2017; Wang et al., 2019; Sun et al., 2023), they cannot be applied directly. In particular, due to the complex product-based parameterization, it is extremely hard to learn parameters in $\boldsymbol{A}, \boldsymbol{B}$ even if we know which entries in their product $\boldsymbol{B}\boldsymbol{A}$ should be 0. Furthermore, simply controlling the sparsity of $\boldsymbol{B}$ and $\boldsymbol{A}$ may not work, as shown in Example 1.

Example 1. *Let $s(\cdot)$ represent the sparsity (i.e., the portion of zero entries) of a vector or matrix. For sparse matrix $\boldsymbol{A} = [\boldsymbol{a}^\top; \boldsymbol{0}^{(r-1)\times d_2}]$ and $\boldsymbol{B} = [\boldsymbol{b}, \boldsymbol{0}^{d_1\times(r-1)}]$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ contains non-zero entries, we have $s(\boldsymbol{A}) = s(\boldsymbol{B}) = \frac{r-1}{r}$ that is reasonably large for $r > 1$. However, $s(\boldsymbol{B}\boldsymbol{A}) = s(\boldsymbol{b}\boldsymbol{a}^\top) = 0$, i.e., the product is a dense matrix.*

To summarize, it is non-trivial to incorporate sparsity in LoRA. To address this challenge, we propose controlling the sparsity of each row of $\boldsymbol{A}$ and each column of $\boldsymbol{B}$. In this way, the sparsity of $\boldsymbol{B}\boldsymbol{A}$ can be bounded by $s(\boldsymbol{A}_{i*})$ and $s(\boldsymbol{B}_{*i})$. We present the theoretical analysis in Proposition 1 and the empirical results in Fig. 2. Based on this finding, we can convert the optimization problem in Eqn. 5 as the following problem:

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{B}} \quad & \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{B}\boldsymbol{A}) \\ \text{s.t.} \quad & \frac{\|\boldsymbol{A}_{i*}\|_0}{d_2} \leq \tau, \frac{\|\boldsymbol{B}_{*i}\|_0}{d_1} \leq \tau, i = 1, ..., r. \end{aligned} \tag{6}$$

Proposition 1. *The sparsity of $\boldsymbol{B}\boldsymbol{A}$ is greater or equal to* $\max\{0, 1 + \sum_{i=1}^{r}(s(\boldsymbol{A}_{i*}) + s(\boldsymbol{B}_{*i}) - s(\boldsymbol{A}_{i*})s(\boldsymbol{B}_{*i})) - r\}$.

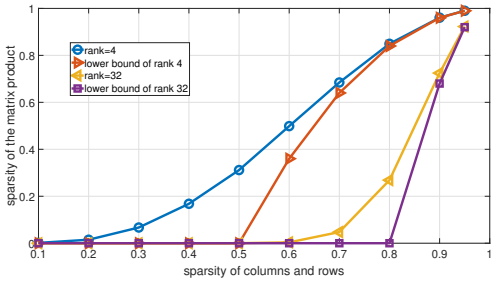

Figure 2: The sparsity of the product of matrix $\boldsymbol{B}$ and $\boldsymbol{A}$ with different column and row sparsity.

4.2 Optimization

In this section, we present how to solve the optimization problem in Eqn. 6. We prune each row of $\boldsymbol{A}$ and each column of $\boldsymbol{B}$ based on sensitivity iteratively. Specifically, we first conduct stochastic gradient decent with respective to $\boldsymbol{A}$ and $\boldsymbol{B}$, i.e.

$$\begin{aligned} \tilde{\boldsymbol{A}}^{(t)} &= \boldsymbol{A}^{(t)} - \nabla_{\boldsymbol{A}^{(t)}}\mathcal{L}, \\ \tilde{\boldsymbol{B}}^{(t)} &= \boldsymbol{B}^{(t)} - \nabla_{\boldsymbol{B}^{(t)}}\mathcal{L}. \end{aligned} \tag{7}$$

Then, we estimate the sensitivity-based importance scores based on Eqn. 4. Given the importance scores, the $\boldsymbol{A}$ and $\boldsymbol{B}$ are pruned following

$$\begin{aligned} \boldsymbol{A}_{i*}^{(t+1)} &= \mathcal{T}_A(\tilde{\boldsymbol{A}}_{i*}^{(t)}, \bar{I}^{(t)}(\boldsymbol{A}_{i*}^{(t)})), \\ \boldsymbol{B}_{*i}^{(t+1)} &= \mathcal{T}_B(\tilde{\boldsymbol{B}}_{*i}^{(t)}, \bar{I}^{(t)}(\boldsymbol{B}_{*i}^{(t)})), \end{aligned} \tag{8}$$

where $i = 1, 2, ..., r$, $\mathcal{T}_A$ is defined as

$$\begin{aligned} &(\mathcal{T}_A(\tilde{\boldsymbol{A}}_{i*}^{(t)}, \bar{I}^{(t)}(\boldsymbol{A}_{i*}^{(t)})))_j \\ &= \begin{cases} \tilde{\boldsymbol{A}}_{ij}^{(t)}, & \bar{I}^{(t)}(\boldsymbol{A}_{ij}^{(t)}) \text{ is top-}\tau^{(t)} \text{ in } \bar{I}^{(t)}(\boldsymbol{A}_{i*}^{(t)}), \\ 0, & \text{otherwise}, \end{cases} \end{aligned}$$

and $\mathcal{T}_B$ is defined as

$$\begin{aligned} &(\mathcal{T}_B(\tilde{\boldsymbol{B}}_{*i}^{(t)}, \bar{I}^{(t)}(\boldsymbol{B}_{*i}^{(t)})))_j \\ &= \begin{cases} \tilde{\boldsymbol{B}}_{ji}^{(t)}, & \bar{I}^{(t)}(\boldsymbol{B}_{ji}^{(t)}) \text{ is top-}\tau^{(t)} \text{ in } \bar{I}^{(t)}(\boldsymbol{B}_{*i}^{(t)}), \\ 0, & \text{otherwise}. \end{cases} \end{aligned}$$

Here, $\tau^{(t)}$ is the budget of the percentage of remaining parameters at the t-iteration. To enable the optimization to be more stable, we decrease the number of $\tau^{(t)}$ gradually following the cubic strategy (Li et al., 2023c):

$$\tau^{(t)} = \begin{cases} 1, & 1 \leq t \leq t_i, \\ \tau + (1-\tau)\left(1 - \frac{t - t_i}{t_f - t_i}\right)^3, & t_i \leq t \leq t_f, \\ \tau, & t_f \leq t \leq T, \end{cases}$$

where T is the number of total training iterations, and t_i, t_f are hyper-parameters.

5 Experiment

In the experiments, we evaluate the proposed RoseLoRA and answer the following questions: **RQ1**) How does the proposed RoseLoRA benefit knowledge editing tasks? **RQ2**) How does RoseLoRA perform compared to state-of-the-art PEFT methods on general tasks? **RQ3**) Does the proposed RoseLoRA alleviate the model forgetting issue? **RQ4**) How does the performance change with varying amounts of training data?

5.1 Datasets and Experiment Settings

Datasets. We conduct experiments on five different benchmarks: 1) **Knowledge Editing**, including $\text{WikiData}_{\text{recent}}$, $\text{WikiData}_{\text{counterfact}}$ (Cohen et al., 2024), ZsRE (Yao et al., 2023), and WikiBio (Hartvigsen et al., 2024); 2) **Commonsense Reasoning**, 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 (Mihaylov et al., 2018); 3) **Arithmetic Reasoning**, including AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021), MAWPS (Koncel-Kedziorski et al., 2016), and SVAMP (Patel et al., 2021); 4) **Instruction Following** with Ultrafeedback (Cui et al., 2023) as training data and evaluation on Alpaca-Eval v1.0 (Li et al., 2023a); 5) **Natural Language Understanding** consists of eight datasets from the GLUE benchmark (Wang et al., 2018). More details about datasets, metrics, and hyper-parameters we use can be found in the Appendix.

Baselines. Our baselines are constructed on a task basis. In specific, on each task the proposed RoseLoRA is compared with representative baselines from corresponding domain as listed below.

- On Knowledge Editing, we follow Zhang et al. (2024) and choose AdaLoRA (Zhang et al., 2023), ROME and FT-L (Meng et al., 2022a), and MEMIT (Meng et al., 2022b) as our baselines as they, same as us, do not require hard-to-access data or training additional models. In specific, AdaLoRA keeps unimportant weights in an LLM unchanged and achieves a highly efficient and precise PEFT. ROME applies a causal-tracing to identify the layer wherein the knowledge is stored and then learns a rank-one update. FT-L, on the other hand, directly finetunes the layer identified by ROME. Recently, MEMIT extends ROME to a large-scale setting, where the edits can be made more efficiently.
- On the other four tasks, we follow the setup from existing works (Hu et al., 2023; Liu et al., 2024; Wu et al., 2024) that evaluated a variety of representative PEFT methods including prefix tuning (Li and Liang, 2021), adapters (Houlsby et al., 2019), LoRA and its recent variants (Hu et al., 2021; Zhang et al., 2023), and ReFT (Wu et al., 2024). Due to page limitation we refer the readers to Hu et al. (2023); Wu et al. (2024) and reference therein for more details.

5.2 Performance Comparison

Knowledge Editing When performing knowledge editing, we introduce an additional norm constraint for low-rank matrices, as detailed in the Appendix. The results of knowledge editing are presented in Table 1, addressing RQ1. From this table, we observe that the proposed RoseLoRA outperforms all state-of-the-art baselines in terms of average performance, achieving the highest edit success rate while preserving the most knowledge that should not be updated. Moreover, RoseLoRA demonstrates excellent generalization ability, as indicated by its high portability score which is a metric to measure if the edited model can reason correctly about the updated knowledge.

Commonsense Reasoning In this section, we present experiments on eight commonsense reasoning datasets to address RQ2, as shown in Table 2. The table indicates that the proposed RoseLoRA again outperforms all state-of-the-art parameter-efficient fine-tuning methods on average. Among the eight datasets, RoseLoRA ranks the first in five cases. Remarkably, its parameter numbers are the same as that of LoReFT, significantly smaller than PrefT, Adapter, LoRA, and DoRA. Yet, RoseLoRA still achieves higher accuracy on the commonsense reasoning datasets. This clearly demonstrates RoseLoRA's effectiveness of fine-tuning the most crucial parameters of LLaMA for commonsense reasoning tasks.

Arithmetic Reasoning In this section, we present experiments on four arithmetic reasoning datasets to address RQ2, with results shown in Table 3. The table indicates that LoRA achieves the

Table 1: Performance comparison of LLaMA-7b-chat against existing knowledge editing methods on four knowledge editing datasets. Results marked with "♡" are taken from Zhang et al. (2024). "AVG" means the average of edit success, locality, portability, and fluency. Because fluency is not at the same magnitude as other metrics, we leverage "fluency/10" when computing AVG values.

Dataset	Metric	FT-L♡	AdaLoRA♡	ROME♡	MEMIT♡	RoseLoRA
$\text{WikiData}_{\text{recent}}$	Edit Succ.(↑)	71.2	65.6	85.1	85.3	**98.4**
	Locality(↑)	63.7	55.8	66.2	64.8	**83.4**
	Portability(↑)	48.7	47.2	37.5	37.9	**54.3**
	Fluency(↑)	549	538	574	567	**585**
	AVG(↑)	59.6	55.6	61.5	61.2	**73.7**
$\text{WikiData}_{\text{counterfact}}$	Edit Succ.(↑)	51.1	72.1	83.2	83.4	**99.4**
	Locality(↑)	62.5	66.8	65.4	63.7	**90.9**
	Portability(↑)	39.1	55.2	38.7	40.1	**57.2**
	Fluency(↑)	545	554	579	569	**592**
	AVG(↑)	51.8	62.4	61.3	61.0	**76.7**
ZsRE	Edit Succ.(↑)	51.1	72.1	83.2	83.4	**100**
	Locality(↑)	62.5	66.8	65.4	63.7	**92.5**
	Portability(↑)	39.1	**55.2**	38.7	40.1	50.9
	Fluency(↑)	545	554	**579**	569	574
	AVG(↑)	54.6	62.1	58.2	54.0	**75.2**
WikiBio	Edit Succ.(↑)	66.3	97.0	95.1	94.3	**99.5**
	Locality(↑)	60.1	57.9	47.0	51.6	**92.5**
	Fluency(↑)	604	616	617	617	**620**
	AVG(↑)	62.3	72.2	67.9	69.2	**84.6**

Table 2: Accuracy comparison of LLaMA-7B against PEFT baselines on eight commonsense reasoning datasets. Results marked with "♡" are taken from Liu et al. (2024). "AVG" means the average accuracy of all datasets. For RoseLoRA, Params (%) is calculated by dividing the number of final low-rank matrices parameters by the number of parameters of the base LMs (number of low-rank matrix parameters times sparsity).

PEFT	Params (%)	Accuracy (↑)								
		BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	**AVG**
PrefT♡	0.11%	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
$\text{Adapter}^{\text{S}}$♡	0.99%	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
$\text{Adapter}^{\text{P}}$♡	3.54%	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
LoRA♡	0.83%	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
DoRA (half)♡	0.43%	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	77.5
DoRA♡	0.84%	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	78.1
LoReFT♡	0.03%	69.3	84.4	**80.3**	**93.1**	**84.2**	83.2	68.2	78.9	80.2
RoseLoRA	0.03%	**71.0**	**84.9**	75.5	92.6	82.6	**84.6**	**70.0**	**84.2**	**80.7**

highest average accuracy across the four datasets. However, the proposed RoseLoRA performs comparably, retaining 97% of LoRA's accuracy while updating only 22 times less parameters compared with LoRA. Additionally, compared to LoReFT, RoseLoRA updates a similar number of parameters while achieving approximately a 6.3% performance improvement.

Instruction Following In this section, we compare the proposed RoseLoRA with state-of-the-art baselines on the instruction-following task. To ensure fair comparisons, we use the same prompt templates from Taori et al. (2023). The model performance is shown in Table 4. Based on the table, it can be observed that the proposed RoseLoRA outperforms all baseline methods while updating the

Table 3: Accuracy comparison of LLaMA-7B against PEFT baselines on four arithmetic reasoning datasets. Results marked with "♡" are taken from Hu et al. (2023). "AVG" means the average accuracy of all datasets.

PEFT	Params (%)	Accuracy (↑)				
		AQuA	GSM8K	MAWPS	SVAMP	AVG
PrefT♡	0.11%	14.2	24.4	63.4	38.1	35.0
Adapter^{S}♡	0.99%	15.0	33.3	77.7	**52.3**	44.6
Adapter^{P}♡	3.54%	18.1	35.3	**82.4**	49.6	46.4
LoRA♡	0.83%	18.9	**37.5**	79.0	52.1	**46.9**
LoReFT♡	0.03%	21.4	26.0	76.2	46.8	42.6
RoseLoRA	0.03%	**26.0**	33.0	79.8	44.7	45.9

fewest parameters. Additionally, for the instruction-following task, we find that significantly fewer parameters need to be updated compared to commonsense reasoning and arithmetic reasoning tasks. This suggests that fewer parameters are related to the instruction-following ability in the large language model.

Table 4: Performance comparison of LLaMA-2 7B on instruction tuning task on Alpaca-Eval v1.0. We compute the win-rate against text-davinci-003 using GPT-4 as the annotator. Results marked with "♡" are taken from Wu et al. (2024).

Model & PEFT	Params (%)	Win-rate (↑)
GPT-3.5 Turbo 1106♡	-	86.30
Llama-2 Chat 13B♡	-	81.10
Llama-2 Chat 7B♡	-	71.40
Llama-2 7B & FT♡	100%	80.93
Llama-2 7B & LoRA♡	0.1245%	81.48
Llama-2 7B & RED♡	0.0039%	81.69
Llama-2 7B & LoReFT♡	0.0039%	85.60
Llama-2 7B & RoseLoRA	0.0037%	**85.77**

Natural Language Understanding We conduct experiments on the GLUE to answer RQ2. We show the model performance in Table 5. According to the table, the proposed RoseLoRA outperforms the state-of-the-art baselines significantly. The best baseline LoRA achieves 88.1 average accuracy but the proposed RoseLoRA reaches about 89.0 accuracy on the eight datasets averagely. On RTE dataset, the proposed RoseLoRA even achieves 3.4% performance improvement. Compared to fully fine-tuning, the proposed RoseLoRA also achieves better performance. The potential reason may be that RoseLoRA only updates very few parameters and prevents overfitting on natural language understanding tasks. It demonstrates that the proposed RoseLoRA not only can be applied to decoder-only models but also can be applied to encoder-only language models.

5.3 Forgetting Test

In this section, we study if a fine-tuned model forgets knowledge learned from the pre-training stage to answer RQ3. To make fair comparisons, we evaluate LoRA and RoseLoRA after fine-tuning on Commonsense170K, Ultrafeedback, and Math10K in a zero-shot setting and using the same prompt templates. We report the experiment results in Table 6. According to the table, we can find that compared to LoRA, the RoseLoRA forgets less knowledge after fine-tuning. For example, after fine-tuning on the Commonsense170K dataset, LoRA leads to a significant performance drop on TriviaQA and MMLU. However, the proposed RoseLoRA still preserves over 90% performance of LLaMA-2. Besides, we can also find that both LoRA and RoseLoRA achieve good performance on ARC-c dataset. It may indicate that fine-tuning large language models on Commonsense170K, Ultrafeedback, or Math10K may not make them forget much general knowledge.

5.4 Sensitivity w.r.t. Training Data Size

In this section, we study how the model performance changes with different amounts of training data. We show the experiment results in Fig. 3. Based on the figure, we can find that with the decreasing amounts of training data, the performance gap between LoRA and RoseLoRA is becoming smaller. When using only 12.5% Math10K data as the training data to fine-tune the LLaMA 7B, RoseLoRA even outperforms LoRA on GSM8K. In conclusion, the proposed RoseLoRA shows more superiority on small data scenarios.

Table 5: Accuracy comparison of RoBERTa-large against PEFT baselines on the GLUE benchmark. Results marked with "♡" are taken from Wu et al. (2023). "AVG" means the average accuracy of all datasets.

PEFT	Params (%)	RTE	MRPC	QQP	STS-b	QNLI	CoLA	SST2	MNLI	AVG
FT♡	100%	85.8	91.7	91.5	92.6	93.8	68.2	96.0	88.8	88.6
Adapter♡	0.254%	85.3	**90.5**	**91.4**	91.5	94.6	65.4	95.2	90.1	88.0
LoRA♡	0.225%	86.3	89.8	90.7	91.7	**94.7**	65.5	96.0	90.2	88.1
Adapter^{FNN}♡	0.225%	84.8	90.5	91.3	90.2	94.3	64.4	96.1	90.3	87.7
RED♡	0.014%	86.2	90.3	88.8	91.3	93.5	68.1	96.0	89.5	88.0
LoReFT♡	0.014%	86.2	90.1	88.5	91.6	94.1	68.0	**96.2**	89.2	88.0
RoseLoRA	0.015%	**89.2**	90.2	91.1	**92.0**	**94.7**	**69.2**	95.2	**90.5**	**89.0**

Table 6: Accuracy of fine-tuned models on TriviaQA (knowledge reasoning), MMLU (general knowledge), and ARC-c (commonsense reasoning) dataset. "AVG" is the average accuracy of Humanities, Social Sciences, STEM, and Other fields on MMLU. The evaluation is conducted with Lm-Evaluation-Harness (Gao et al., 2023).

	TriviaQA	MMLU					ARC-c
		Humanities	Social Sciences	STEM	Other	AVG	
LLaMA 7B	48.6	29.9	29.4	26.3	33.4	29.8	41.7
After Commonsense170K							
LoRA	9.0	24.4	21.9	21.5	24.0	23.1	-
RoseLoRA	**47.8**	**36.8**	**42.7**	**31.4**	**42.3**	**38.1**	-
After Math10K							
LoRA	30.5	31.1	34.4	30.5	35.7	32.7	**42.2**
RoseLoRA	**51.3**	**37.9**	**43.0**	**32.1**	**43.9**	**39.0**	41.9
LLaMA-2 7B	52.5	38.9	46.1	34.3	47.1	41.2	43.4
After Ultrafeedback							
LoRA	23.5	41.3	49.4	43.0	49.3	43.0	41.2
RoseLoRA	**30.1**	**42.1**	**51.5**	**44.9**	**52.0**	**44.9**	**44.4**

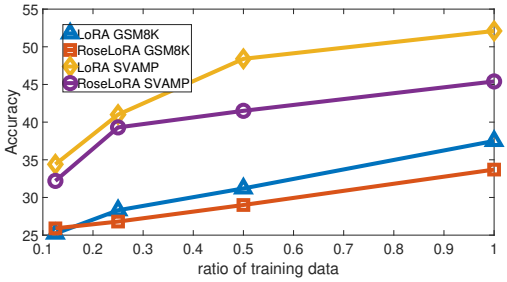

Figure 3: Accuracy of LoRA and RoseLoRA with different amount of Math10K training data on GSM8K and SVAMP.

6 Conclusion

In this paper, we address the limitations of existing parameter-efficient fine-tuning (PEFT) methods, particularly the LoRA family, in handling tasks requiring selective knowledge updates while still being effective for other general NLP tasks. We introduced a novel method, **r**ow and c**o**lumn-wise spar**se** **lo**w-**r**ank **a**daptation (RoseLoRA), which selectively updates the most important parameters for specific tasks, maintaining efficiency while minimizing unnecessary changes to the pre-trained model's knowledge. RoseLoRA applies a row and column-wise sparsity constraint to the product of low-rank matrices, ensuring efficient updates without modifying the model architecture. Our theoretical analysis lower bounds the sparsity of product matrices that affect model's knowledge, and our sensitivity-based importance scoring effectively fulfilled the sparsity constraints. Through extensive experiments on five benchmarks encompassing over twenty datasets, RoseLoRA demonstrated superior performance on both general-purposed fine-tuning and knowledge editing tasks compared to existing methods. This highlights its potential as a robust and efficient fine-tuning solution for a wide range of NLP applications.

Limitations

The proposed RoseLoRA framework introduces a hyper-parameter β to smooth the sensitivity estimation, which might require additional effort to tune. Fortunately, we observe that the model performance is not sensitive to the hyper-parameter and we set it to a fixed value to achieve good performance in this paper.

Acknowledgement

This work is supported in part by the US National Science Foundation under grant NSF IIS-1747614 and NSF IIS-2141037. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

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.

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. *arXiv preprint arXiv:1905.10044*.

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*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023a. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023b. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

Joshua Engels, Isaac Liao, Eric J Michaud, Wes Gurnee, and Max Tegmark. 2024. Not all language model features are linear. *arXiv preprint arXiv:2405.14860*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

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.

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. *Preprint*, arXiv:2106.09685.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023b. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023c. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR.

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*.

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*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

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. *arXiv preprint arXiv:1809.02789*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272.

Balas Kausik Natarajan. 1995. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

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.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389.

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

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Suhang Wu, Minlong Peng, Yue Chen, Jinsong Su, and Mingming Sun. 2023. Eva-kellm: A new benchmark for evaluating knowledge editing of llms. *arXiv preprint arXiv:2308.09954*.

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*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

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 *International Conference on Learning Representations*.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

A Proof of Proposition 1

Lemma 1. *For $\boldsymbol{a} \in \mathbb{R}^{1\times d_2}$ and $\boldsymbol{b} \in \mathbb{R}^{d_1\times 1}$, where the sparsity of them is $s(\boldsymbol{a}) = s_a$ and $s(\boldsymbol{b}) = s_b$ respectively, we have $s(\boldsymbol{ba}) = s_a + s_b - s_a s_b$.*

Proof. Define the number of zero values in a vector or matrix as $z(\cdot)$. Consider the i-th row of $\boldsymbol{ba}$, i.e. $\boldsymbol{b}_i\boldsymbol{a}$. If $\boldsymbol{b}_i = 0$, then $\boldsymbol{b}_i\boldsymbol{a} = \boldsymbol{0}$. If $\boldsymbol{b}_i \neq 0$, then the number of zeros depends on the number of zeros of $\boldsymbol{a}$. Therefore, we have

$$z(\boldsymbol{b}_i\boldsymbol{a}) = \begin{cases} d_2, & \boldsymbol{b}_i = 0, \\ s_a d_2, & \boldsymbol{b}_i \neq 0. \end{cases} \tag{9}$$

Then we have

$$\begin{aligned} z(\boldsymbol{ba}) =& \sum_{i=1}^{d_1} z(\boldsymbol{b}_i\boldsymbol{a}) \\ =& d_2 s_b d_1 + s_a d_1 d_2 (1 - s_b) \\ =& d_1 d_2 (s_a + s_b - s_a s_b). \end{aligned} \tag{10}$$

So the sparsity of $\boldsymbol{ba}$ is

$$\begin{aligned} s(\boldsymbol{ba}) =& \frac{d_1 d_2 (s_a + s_b - s_a s_b)}{d_1 d_2} \\ =& s_a + s_b - s_a s_b. \end{aligned} \tag{11}$$

□

Proposition 1. *The sparsity of $\boldsymbol{BA}$ is greater or equal to $\max\{0, 1 + \sum_{i=1}^{r}(s(\boldsymbol{A}_{i*}) + s(\boldsymbol{B}_{*i}) - s(\boldsymbol{A}_{i*})s(\boldsymbol{B}_{*i})) - r\}$.*

Proof. First, we have

$$\begin{aligned} (\boldsymbol{BA})_{ij} =& \sum_{k=1}^{r} \boldsymbol{B}_{ik}\boldsymbol{A}_{kj} \\ =& \sum_{k=1}^{r} (\boldsymbol{B}_{*k}\boldsymbol{A}_{k*})_{ij}. \end{aligned} \tag{12}$$

Consider the worst case: the positions of nonzero value of $\{\boldsymbol{B}_{*k}\boldsymbol{A}_{k*}\}$ does not have any overlapping, we at least have $\max\{0, d_1 d_2 - \sum_{i=1}^{r}(1 - s(\boldsymbol{B}_{*i}\boldsymbol{A}_{i*}))d_1 d_2\}$ zero values.

Therefore, based on Lemma 1 the sparsity of $\boldsymbol{BA}$ satisfies

$$\begin{aligned} & s(\boldsymbol{BA}) \\ \geq& \frac{\max\{0, d_1 d_2 - \sum_{i=1}^{r}(1 - s(\boldsymbol{B}_{*i}\boldsymbol{A}_{i*}))d_1 d_2\}}{d_1 d_2} \\ =& \max\{0, 1 + \sum_{i=1}^{r} s(\boldsymbol{B}_{*i}\boldsymbol{A}_{i*}) - r\} \\ =& \max\Big\{0, 1 + \sum_{i=1}^{r} \big(s(\boldsymbol{A}_{i*}) + s(\boldsymbol{B}_{*i}) \\ & - s(\boldsymbol{A}_{i*})s(\boldsymbol{B}_{*i})\big) - r\Big\}. \end{aligned} \tag{13}$$

□

B Datasets, Metrics and Hyper-parameters

We conduct experiments on five different benchmarks:

- Knowledge editing consists of four datasets, including $\text{WikiData}_{\text{recent}}$, $\text{WikiData}_{\text{counterfact}}$ (Cohen et al., 2024), ZsRE (Yao et al., 2023), and WikiBio (Hartvigsen et al., 2024). For the knowledge editing tasks, the model should memorize new knowledge while preserving knowledge which does not need to update. Following Zhang et al. (2024), we use four metrics to evaluate the editing performance: 1) **Edit Success**, which estimates the accuracy with respect to both the knowledge needed to be updated and the similar expressions of the knowledge, 2) **Locality**, which shows if the post-edited model keeps its original answer on the locality set, 3) **Portability**, which is to measure if the post-edited model can reason correctly about the updated knowledge, and 4) **Fluency**, which measures the model's generation ability after editing via calculating the weighted average of bi-gram and tri-gram entropies.

- Commonsense reasoning contains of eight datasets, 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 (Mihaylov et al., 2018). These tasks are multiple choice problems. Following Hu et al. (2023); Wu et al. (2024), we fine-tune the LLM on a combined training dataset named Commonsense170K of these tasks and evaluate the Accuracy on individual test sets.

Table 7: Hyper-parameters used in knowledge editing, commonsense reasoning and arithmetic reasoning.

Dataset	lr	Rank	Batch size	Sparsity	β	α	Target modules
WikiData recent	2e-4	4	1	0.95		3e-3	"up_proj", "down_proj", "gate_proj"
WikiData counterfact	2e-4	4	1	0.95		3e-3	"up_proj", "down_proj", "gate_proj"
ZsRE	2e-4	4	1	0.95		3e-3	"up_proj", "down_proj", "gate_proj"
WikiBio	2e-4	4	1	0.95	0.8	3e-3	"up_proj", "down_proj", "gate_proj"
Commonsense170K	2e-4	32	8	0.865		-	"q_proj","v_proj"
Math10K	3e-4	32	32	0.865		-	"q_proj","v_proj"
Instruction tuning	3e-4	32	32	0.85		-	"q_proj","v_proj"

Table 8: Hyper-parameters and metrics used in GLUE benchmark.

Dataset	Metric	lr	Rank	Batch size	Sparsity	β	Target modules
CoLA	Matthews corr	2e-4	6	16	0.95	0.8	"query", "key", "value", "output.dense", "intermediate.dense"
SST-2	Accuracy	2e-4		32			
MRPC	Accuracy	2e-4		32			
QQP	Accuracy	1e-4		32			
STS-B	Pearson corr	2e-4		32			
MNLI	Accuracy	2e-4		32			
QNLI	Accuracy	2e-4		32			
RTE	Accuracy	6e-4		32			

- Arithmetic reasoning consists of four math reasoning datasets: AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021), MAWPS (Koncel-Kedziorski et al., 2016), and SVAMP (Patel et al., 2021). Models need to generate correct answers and we use Accuracy as the evaluation metric following Hu et al. (2023) as well. Again, we replicate the setup in Wu et al. (2024) and fine-tune the models on the combined training data named Math10K of the four tasks.

- Instruction-following measures if the model can follow human instructions. Same as before, we follow Hu et al. (2023); Wu et al. (2024) and use Ultrafeedback (Cui et al., 2023) as the training data, and evaluate the model performance by Alpaca-Eval v1.0 (Li et al., 2023a).

- Natural language understanding consists of eight datasets from the GLUE benchmark (Wang et al., 2018). We adopt the evaluation metrics and setups from Wu et al. (2023).

We show the hyper-parameters we use in Table 8 and Table 7. We conduct experiments based on libraries LLM-Adapters[1], EasyEdit[2], and lm-evaluation-harness[3].

[1]https://github.com/AGI-Edgerunners/LLM-Adapters
[2]https://github.com/zjunlp/EasyEdit
[3]https://github.com/EleutherAI/lm-evaluation-harness

C Implementation of Knowledge Editing

To enable the minimal modification of the LLM, following (Zhang et al., 2024), we add one ℓ_2 norm on the low-rank matrices:

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{B}} \quad & \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{B}\boldsymbol{A}) \\ \text{s.t.} \quad & \frac{\|\boldsymbol{A}_{i*}\|_0}{d_2} \leq \tau, \frac{\|\boldsymbol{B}_{*i}\|_0}{d_1} \leq \tau, i = 1, ..., r, \\ & \|\boldsymbol{A}\|_F^2 \leq \alpha, \|\boldsymbol{B}\|_F^2 \leq \alpha, \end{aligned} \tag{14}$$

where α is a hyper-parameter. In each step, after pruning $\boldsymbol{A}$ and $\boldsymbol{B}$, we clip them to make them satisfy the ℓ_2 norm constraint.