CorDA: Context-Oriented Decomposition Adaptation of Large Language Models for Task-Aware Parameter-Efficient Fine-tuning

Yibo Yang[1], **Xiaojie Li**[2,3], **Zhongzhu Zhou**[4], **Shuaiwen Leon Song**[4], **Jianlong Wu**[2]
Liqiang Nie[2,†], **Bernard Ghanem**[1,†]

[1]King Abdullah University of Science and Technology (KAUST)
[2]Harbin Institute of Technology (Shenzhen) [3]Peng Cheng Laboratory [4]University of Sydney
† : corresponding authors
https://github.com/iboing/CorDA

Abstract

Current parameter-efficient fine-tuning (PEFT) methods build adapters widely agnostic of the context of downstream task to learn, or the context of important knowledge to maintain. As a result, there is often a performance gap compared to full-parameter fine-tuning, and meanwhile the fine-tuned model suffers from catastrophic forgetting of the pre-trained world knowledge. In this paper, we propose **CorDA**, a **C**ontext-**or**iented **D**ecomposition **A**daptation method that builds learnable **task-aware adapters** from weight decomposition oriented by the context of downstream task or the world knowledge to maintain. Concretely, we collect a few data samples, and perform singular value decomposition for each linear layer of a pre-trained LLM multiplied by the covariance matrix of the input activation using these samples. The inverse of the covariance matrix is multiplied with the decomposed components to reconstruct the original weights. By doing so, the context of the representative samples is captured through deciding the factorizing orientation. Our method enables two options, the **knowledge-preserved adaptation** and the **instruction-previewed adaptation**. For the former, we use question-answering samples to obtain the covariance matrices, and use the decomposed components with the smallest r singular values to initialize a learnable adapter, with the others frozen such that the world knowledge is better preserved. For the latter, we use the instruction data from the fine-tuning task, such as math or coding, to orientate the decomposition and train the largest r components that most correspond to the task to learn. We conduct extensive experiments on Math, Code, and Instruction Following tasks. Our knowledge-preserved adaptation not only achieves better performance than LoRA on fine-tuning tasks, but also mitigates the forgetting of world knowledge. Our instruction-previewed adaptation is able to further enhance the fine-tuning performance to be comparable with full fine-tuning, surpassing the state-of-the-art PEFT methods such as LoRA, DoRA, and PiSSA.

1 Introduction

Large language models (LLMs) have shown remarkable abilities in a wide range of challenging tasks, including questing-answering [11, 51], common sense reasoning [6], and instruction following [85]. While being powerful, LLMs demand exorbitant computation and memory cost when fine-tuning the whole model on downstream tasks due to the huge model capacity. To enable resource-friendly adaptation on downstream tasks, parameter-efficient fine-tuning (PEFT) methods are proposed to largely reduce the number of trainable parameters, by only fine-tuning the newly added adapters [20, 21, 17] or tokens [31, 37, 53], with the original pre-trained weights frozen.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Among these PEFT methods, LoRA [21] is increasingly attractive because it is able to keep the model architecture unchanged after fine-tuning so does not induce extra burden in inference. LoRA suggests that the weight change in fine-tuning presents a low rank structure, and employs low-rank matrices with a low hidden dimension to approximate the adaptation [21]. Following studies introduce adaptive low rank choice among different layers [78, 59, 75], decouple the learning of magnitude and direction [42], combine LoRA with pruning or quantization [10, 65, 38, 77], and further reduce the number of trainable parameters [26, 55]. However, existing studies build learnable adapters without considering any data context. As a result, these initialized adapters are task-agnostic, and may not be the optimal choice for the downstream task to learn. Moreover, even if PEFT methods only train a small number of parameters, the fine-tuned model will still suffer from catastrophic forgetting, losing much of the world knowledge contained in the pre-trained LLM [13, 61, 83]. As shown in our visualization (Figures 4 and 5 in Appendix C) for the covariance matrices with input from different datasets, similar outlier patterns can be observed for inputs belonging to the same kind of task, which implies that the responsive parts of the LLM pre-trained weights are different when different tasks are triggered. Therefore, the adapter should be built upon the components most associated with the task of concern, *e.g.,* a new ability to learn or the QA ability to maintain pre-trained world knowledge.

To this end, we propose a **task-aware PEFT** method named as **CorDA**, based on **C**ontext-**or**iented **D**ecomposition **A**daptation. It adopts the same low-rank design as LoRA [21], namely introducing two low rank matrices for each linear layer as a learnable adapter, but associates the context of world knowledge or fine-tuning task with the process of building these adapters. First, we randomly collect a few data samples and assume that they contain representative context of the corresponding task. For example, the question from a question-answering dataset well indicates the ability of preserving the corresponding knowledge, and the query to write a code carries the context of the coding task. We feed these samples into a pre-trained LLM, and obtain the covariance matrix of the input activation of each linear layer, *i.e.,* $C = XX^T \in \mathbb{R}^{d_{in} \times d_{in}}$, where X is the input of this layer. We then perform singular value decomposition (SVD) for the weight $W \in \mathbb{R}^{d_{out} \times d_{in}}$ multiplied by the covariance matrix, *i.e.,* $\texttt{SVD}(WC) = U\Sigma V^T$, where U and V are singular vectors and Σ is the diagonal matrix with the singular values arranged in descending order. In this way, the representative context expressed by these covariance matrices is able to direct the factorizing orientation. Finally, the inverse of these covariance matrices is multiplied with the decomposed components to hold the same inference result with the original model at initialization, *i.e.,* $\hat{W} = U\Sigma V^T C^{-1}$, where $\hat{W}$ is the weight after decomposition and reconstruction.

Our method supports two optional modes for practitioners, **knowledge-preserved adaptation** and **instruction-previewed adaptation**. LLM fine-tuning on downstream tasks is always accompanied by the damage of world knowledge acquired from massive pre-training data [13]. Our knowledge-preserved adaptation enables to learn new tasks effectively while keeping world knowledge as sound as possible. In this mode, we use questions from question-answering dataset, such as TriviaQA [23] and Natural Questions [27, 29], to obtain the covariance matrices whose pattern corresponds to the LLM ability in retrieving knowledge. After SVD with these covariance matrices, we use the components with the smallest r singular values, *i.e.,* $U_{[:,-r:]}$, $\Sigma_{[-r:]}$, and $(V^T C^{-1})_{[-r:,:]}$ [1], to initialize a learnable low-rank adapter, and the other components that are key to preserving knowledge are frozen. Alternatively, when one only aims to achieve performance as high as possible on the fine-tuning task without concern for world knowledge maintenance, our instruction-previewed adaptation is suggested. In this mode, we use the instruction and response from the fine-tuning task, *e.g.* query to write a code and its answer, to produce the covariance matrices. Similarly, the pre-trained weights will be decomposed in an orientation such that the context of the fine-tuning task dominates in the principal singular values and vectors. Therefore, we use the largest r components, *i.e.,* $U_{[:,:r]}$, $\Sigma_{[:r]}$, and $(V^T C^{-1})_{[:r,:]}$, to initialize a learnable low-rank adapter, with the other components frozen. The adapter built upon the context of the fine-tuning task well accommodates the new ability, and thus leads to a better fine-tuning performance.

Our method brings flexibility in choosing between stronger fine-tuning performance or more preserved world knowledge, and can be adopted according to the practical demand. Both the two modes are as efficient as LoRA [21]. After fine-tuning, the adapter can be merged with the frozen part in the

[1] $U_{[:,-r:]}$, $\Sigma_{[-r:]}$, and $(V^T C^{-1})_{[-r:,:]}$ represent the last r columns of U, the last r diagonal elements of Σ, and the last r rows of $V^T C^{-1}$, respectively. $U_{[:,:r]}$, $\Sigma_{[:r]}$, and $(V^T C^{-1})_{[:r,:]}$ represent the first r columns of U, the first r diagonal elements of Σ, and the first r rows of $V^T C^{-1}$, respectively.

same manner as LoRA. In experiments, CorDA in knowledge-preserved adaptation not only enjoys better fine-tuning performance than LoRA on Math [9, 71], Code [8, 3], and Instruction Following [80], but also largely mitigates the deterioration of performance on world knowledge benchmarks including TriviaQA [23], NQ open [29], and WebQS [4]. The instruction-previewed adaptation is able to further strengthen the performance on fine-tuning tasks, surpassing the state-of-the-art PEFT methods including LoRA [21], DoRA [42], and PiSSA [47].

2 Related Work

Parameter-Efficient Fine-Tuning. Since large language models (LLMs) have tens and even hundreds of billions of parameters [1, 5], full-parameter fine-tuning will cause unbearable computation and memory cost [79, 67]. Parameter-efficient fine-tuning (PEFT) is developed to reduce resource consumption by only fine-tuning a small number of learnable parameters [12, 64]. Adapter-based methods introduce additional modules into LLMs and only fine-tune them during fine-tuning [20, 17, 30, 45, 49, 24]. Another line of research appends extra soft prompts into the input or hidden layers and only train these learnable vectors [31, 37, 53, 84]. However, most of these methods change the model architecture or increase the inference burden. Based on the insight that the weight change after fine-tuning possesses a low rank structure [32, 2], low-rank adaptation (LoRA) [21] proposes to use two low-rank small matrices as the learnable adapter, without modifying model architecture or bringing inference cost after fine-tuning. LoRA has inspired a range of variants that employ adaptive low rank in different layers [78, 59, 75], explore the adapter design [42, 7, 50, 79], combine LoRA with pruning [77], quantization [10, 65, 38], and mixture-of-expert [41, 13], and introduce alternative way to initialize the adapter [47]. Nevertheless, existing PEFT methods rarely consider data context when building the learnable adapter. Data context has been proved to be instrumental in guiding quantization and compression [40, 73, 28]. In our study, we utilize data context for PEFT, building adapters based on context-oriented decomposition to better maintain world knowledge or accommodate new ability.

Knowledge Forgetting. Deep learning models are prone to drastically forgetting the acquired knowledge when adapting to a new task, known as catastrophic forgetting [14, 54, 25, 52, 44, 70, 36]. A series of methods has been proposed to mitigate this issue using knowledge distillation [39, 19, 33], rehearsal [56, 69], and dynamic architecture [66]. In the era of large models [62], however, world knowledge is acquired by pre-training on massive data, which could be intractable to re-use in fine-tuning [34, 35]. The huge model capacity also hinders the feasibility of knowledge distillation and dynamic architecture, especially in the case of continuous fine-tuning [16, 74, 57, 15, 22]. Some studies introduce extra LLaMA layers [61] or mixture of experts [13] with the pre-trained layers frozen to strike a balance between keeping world knowledge and learning new tasks. Alternative approaches adopt merging schemes to enable diverse abilities [76, 72, 83]. Different from these studies, our method enables to achieve world knowledge maintaining in the process of parameter-efficient fine-tuning, without changing model architecture or relying on a post-merging step.

3 Method

We review the LoRA method in Sec. 3.1. We develop our context-oriented decomposition in Sec. 3.2, which provides the basis of our knowledge-preserved adaptation and instruction-previewed adaptation introduced in Sec. 3.3 and Sec. 3.4, respectively.

3.1 Preliminaries on Low-Rank Adaptation

LoRA suggests that the weight change in LLM fine-tuning presents a low rank structure, and thus proposes to use the product of two low rank matrices to learn the weight change with the pre-trained weights frozen during fine-tuning [21]. Given the pre-trained weight $W \in \mathbb{R}^{d_{out} \times d_{in}}$ from an LLM, the weight after fine-tuning can be formulated as:

$$W^* = W + \Delta W = W + BA, \tag{1}$$

where W^* is the weight after fine-tuning, ΔW is the weight change, and BA is the low rank decomposition of ΔW into two smaller matrices $B \in \mathbb{R}^{d_{out} \times r}$ and $A \in \mathbb{R}^{r \times d_{in}}$ with an intrinsic rank of $r \ll \min(d_{out}, d_{in})$. In this way, the number of learnable parameters can be largely reduced

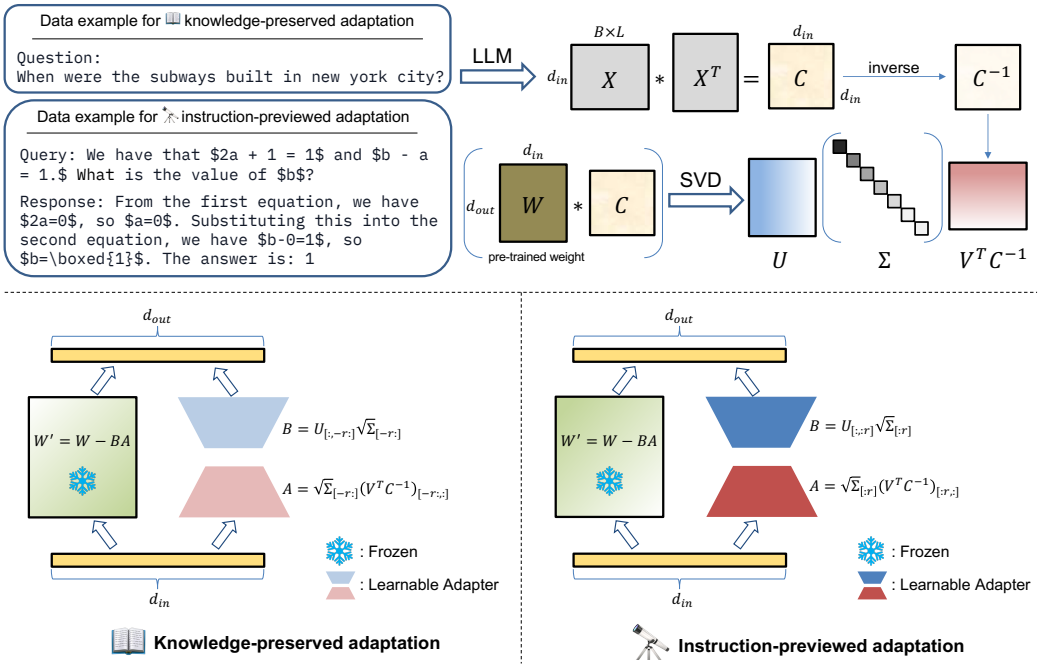


Figure 1: An overall illustration of our proposed method. We perform singular value decomposition oriented by the covariance matrix to aggregate task context into the principal components (up), which are frozen for maintaining world knowledge (down left) or utilized to initialize the learnable adapter for better fine-tuning performance (down right). The dark-colored adapter refers to the components with the largest r singular values, while the light one is composed of the smallest r components.

by freezing the pre-trained weight W and only fine-tuning the matrices B and A. LoRA adopts the Kaiming initialization [18] to randomly initialize A, and B is initialized as an all-0 matrix such that $\Delta W = 0$ at the start of training to circumvent a deviation from the pre-trained model. After fine-tuning, the learnable adapter BA can be merged into the pre-trained weight W without changing the original model architecture and introducing extra inference burden.

Despite the success of LoRA-based methods, when building the learnable adapter, existing studies widely ignore the data context from the target ability that users are particularly concerned with.

3.2 Context-Oriented Decomposition

Pre-trained large language models are endowed with multi-faced abilities, such as answering the question regarding world knowledge, common sense reasoning, and instruction following. When different kinds of input messages are fed into an LLM, *e.g.,* a question in some domain and a query to solve a math problem, even though they are processed by the same pre-trained weights, different abilities are triggered. The covariance matrix of each layer's activation will exhibit different outlier patterns as they are responsive to the task triggered to highlight different aspects of the pre-trained weight. Therefore, the covariance matrix is able to capture task context. Inspired by this insight, we leverage the covariance matrix inside LLM to build adapters catering to a certain ability.

The process of our context-oriented decomposition is shown in Figure 1. First, we randomly collect some samples from the training data of some task with interest, *e.g.,* question answering or Math, and feed these samples into the LLM used to fine-tune. Denote $X \in \mathbb{R}^{d_{in} \times BL}$ as the input activation of a linear layer where d_{in} is the input dimension, B is the number of samples we collect, and L represents the sequence length. We have the covariance matrix $C = XX^T \in \mathbb{R}^{d_{in} \times d_{in}}$. We then perform singular value decomposition for the weight multiplied by the covariance matrix as:

$$\texttt{SVD}(WC) = U\Sigma V^T = \sum_{i=1}^{R} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2}$$

where $W \in \mathbb{R}^{d_{out} \times d_{in}}$ is the weight of this linear layer, $U \in \mathbb{R}^{d_{out} \times d_{out}}$ and $V \in \mathbb{R}^{d_{in} \times d_{in}}$ are orthogonal matrices containing singular vectors $\mathbf{u}_i \in \mathbb{R}^{d_{out}}$ and $\mathbf{v}_i \in \mathbb{R}^{d_{in}}$, $\Sigma \in \mathbb{R}^{d_{out} \times d_{in}}$ is a diagonal matrix with singular values σ_i on its diagonal arranged in descending order, and R is the rank (the number of non-zero singular values) of WC, *i.e.,* $R \leq \min\{d_{out}, d_{in}\}$.

To not change the inference result at the initialization of fine-tuning, we reconstruct W by:

$$\hat{W} = \texttt{SVD}(WC)C^{-1} = U\Sigma(V^T C^{-1}) = \sum_{i=1}^{R} \sigma_i \mathbf{u}_i \hat{\mathbf{v}}_i^T, \tag{3}$$

where C^{-1} denotes the inverse of C, and $\hat{\mathbf{v}}_i^T$ is the i-th row vector of $V^T C^{-1}$. In case the covariance matrix C is not invertible, we adopt a strategy to dynamically add positive values on the diagonal elements of C to ensure invertible. Concretely, we multiply a positive coefficient with the average value of the diagonal elements of C, and add it on the diagonal. Then we calculate the ℓ_2 distance between CC^{-1} and an identity matrix. If it is higher than a threshold, we double the coefficient and perform this step again, until the distance reaches below the threshold.

After our context-oriented decomposition, the first several components of $\mathbf{u}_i$ and $\hat{\mathbf{v}}_i$ with the largest singular values σ_i depict the dominant characteristics of the task associated with C. We can decide either to maintain these key components to not sacrifice the corresponding ability, or to adapt them for better performance on the task, which leads to our two implementation modes in the following two subsections, respectively.

3.3 Mode 1: Knowledge-Preserved Adaptation

We introduce knowledge-preserved adaptation that enables to learn new tasks while maintaining world knowledge. In this mode, we use the question data from question-answering training data, such as TriviaQA [23] and Natural Questions [27, 29], to obtain the covariance matrices whose pattern corresponds to the knowledge retrieving ability of the LLM. When fine-tuning on a new task, as shown in Figure 1, we use the last r components with the smallest r singular values in Eq. (3) to build learnable adapters as:

$$W' = W - BA, \quad B = U_{[:,-r:]}\sqrt{\Sigma}_{[-r:]}, \quad A = \sqrt{\Sigma}_{[-r:]}(V^T C^{-1})_{[-r:,:]}, \tag{4}$$

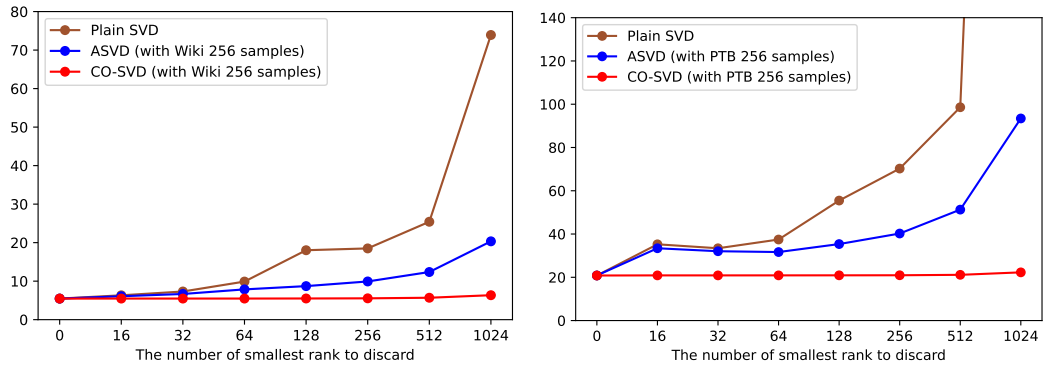
where $B \in \mathbb{R}^{d_{out} \times r}$ and $A \in \mathbb{R}^{r \times d_{in}}$ are the initialized matrices in the learnable adapter, $BA = \sum_{i=R-r+1}^{R} \sigma_i \mathbf{u}_i \hat{\mathbf{v}}_i^T$ corresponds to the last r components in Eq. (3), $\sqrt{\Sigma}_{[-r:]}$ is a diagonal matrix with the squared root of the smallest r singular values on the diagonal, and W' corresponding to the first $R - r$ components in Eq. (3) is frozen during fine-tuning. We have W' as the difference between W and BA instead of summing the first $R - r$ components to avoid the numerical error between $\hat{W}$ and W introduced by the decomposition and inversion operations. After fine-tuning, the learned matrices B^* and A^* can be merged into W' as $W^* = W' + B^* A^*$. This mode is featured by preserving world knowledge as the knowledge retrieving ability captured by the principal $R - r$ components is frozen. It is also more effective than a zero-initialized adapter in learning new abilities as verified by our experiments.

3.4 Mode 2: Instruction-Previewed Adaptation

In the circumstance that pursuing a higher performance on the fine-tuning task is the priority, our instruction-previewed adaptation will be favorable. In this mode, we collect instruction and response from the training data used for fine-tuning, *e.g.* the query to solve a math problem and its answer shown in Figure 1 as an example. The prompts are fed into the LLM to produce the covariance matrices whose pattern is associated with the task to learn. We use the first r components with the largest r singular values in Eq. (3) to build learnable adapters as:

$$W' = W - BA, \quad B = U_{[:,:r]}\sqrt{\Sigma}_{[:r]}, \quad A = \sqrt{\Sigma}_{[:r]}(V^T C^{-1})_{[:r,:]}, \tag{5}$$

where B and A are the initialized matrices in the learnable adapter, $BA = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \hat{\mathbf{v}}_i^T$ corresponds to the first r components in Eq. (3), $\sqrt{\Sigma}_{[:r]}$ is the squared root of the largest r singular values in a diagonal matrix. Similar to knowledge-preserved adaptation, W' containing the remaining $R - r$ components is frozen during fine-tuning. This mode enables the initialized adapters to pre-capture



(a) Perplexity on Wikitext-2

(b) Perplexity on PTB

Figure 2: Perplexity (lower is better) on (a) Wikitext-2 and (b) Penn TreeBank (PTB) after decomposing the LLaMA-2-7B weights and reconstruction discarding the smallest r singular values and their singular vectors. We compare our context-oriented decomposition (CO-SVD) with plain SVD and ASVD. The perplexity of plain SVD on PTB at $r = 1024$ is 763.4, which is out of the shown range.

the main characteristics of the fine-tuning task, leading to stronger performance after training. A recently proposed method [47] also performs SVD and uses the first r components to initialize an adapter for fine-tuning. However, their decomposition is agnostic with respect to any data context. Our adapter capturing the task context in advance can well accommodate the new ability and lead to better fine-tuning performances in our experiments.

4 Experiments

In experiments, we fine-tune the pre-trained large language model LLaMA-2-7b [58] on Math, Code, and Instruction Following tasks, and also apply our method to the General Language Understanding Evaluation (GLUE) benchmark with $\text{RoBERTa}_{\text{base}}$ [43]. The world knowledge is evaluated by the exact match scores (%) on TriviaQA [23], NQ open [29], and WebQS [4]. Following the settings in [47], the Math ability is trained on MetaMathQA [71] and tested on GSM8k [9] and Math [71] validation sets. Code is trained on CodeFeedback [82] and tested on HumanEval [8] and MBPP [3]. Instruction following is trained on WizardLM-Evol-Instruct [63] and tested on MTBench [81]. The complete implementation details are described in Appendix A.

4.1 Analysis of the Ability to Capture Context

We conduct an experiment to demonstrate the ability of our proposed decomposition method in Sec. 3.2 to capture context in its principal components. We use different methods including the Plain SVD, ASVD [73], and our Context-Oriented SVD (CO-SVD), to perform full decomposition of the LLaMA-2-7B pre-trained weights in all layers, and then discard the smallest r singular values and their corresponding left and right singular vectors to reconstruct the weights for testing.

As shown in Figure 2, as the number of discarded ranks increases, the performances with Plain SVD on both Wikitext-2 [48] and PTB [46] are getting worse steeply. ASVD considers data context using activation absolute mean values, and helps to relieve the deterioration compared to Plain SVD. However, when discarding more than 256 ranks, the Perplexity also diverges sharply. In contrast, our method is able to maintain a stable performance very close to the original pre-trained weights even when the smallest 1024 components are discarded. The result indicates that our method is proficient with aggregating context from limited samples (256 Wiki or PTB samples in this example) into the principal components. It also evidences the potential of our method to maintain important knowledge when freezing the principal components and to learn new abilities when adapting these components. The detailed numbers of the experiment in Figure 2 are listed in Table 6 (Appendix B), where we also test the effect of sample number and dataset choice when collecting the covariance matrices.

Table 1: The experimental results of CorDA in the knowledge-preserved adaptation mode and comparison with full fine-tuning, LoRA, and PiSSA. LLaMA-2-7B is used to fine-tune on (a) Math, (b) Code, and (c) Instruction Following tasks. The rank r of LoRA, PiSSA, and CorDA is 128. CorDA is initialized with the NQ open samples to collect the covariance matrices. All methods are implemented by us under the same training and evaluation settings. The row of "LLaMA-2-7B" shows the world knowledge performance of the original pre-trained model.

(a) Math

Method	#Params	Trivia QA	NQ open	WebQS	GSM8k	Math	Avg
LLaMA-2-7B	-	52.51	14.99	5.86	-	-	-
Full fine-tuning	6738M	43.64	3.13	6.35	48.90	7.48	21.90
LoRA [21]	320M	44.17	1.91	6.64	42.68	5.92	20.26
PiSSA [47]	320M	39.71	1.02	6.30	**51.48**	**7.60**	21.22
CorDA (ours)	320M	**44.30**	**9.36**	**7.14**	44.58	6.92	**22.46**

(b) Code

Method	#Params	Trivia QA	NQ open	WebQS	HumanEval	MBPP	Avg
LLaMA-2-7B	-	52.51	14.99	5.86	-	-	-
Full fine-tuning	6738M	29.29	8.53	3.44	**25.42**	**25.64**	18.46
LoRA [21]	320M	**51.42**	9.30	8.46	16.8	21.51	21.50
PiSSA [47]	320M	47.07	9.16	8.14	19.48	23.84	21.54
CorDA (ours)	320M	50.02	**11.72**	**8.56**	18.36	20.91	**21.91**

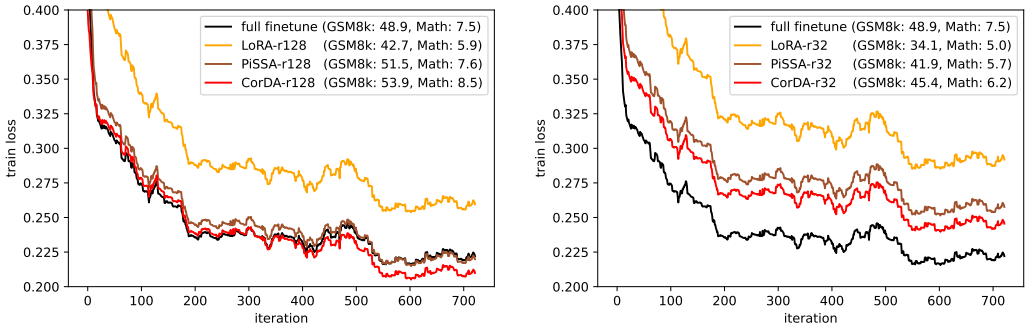
(c) Instruction Following

Method	#Params	Trivia QA	NQ open	WebQS	MTBench	Avg
LLaMA-2-7B	-	52.51	14.99	5.86	-	-
Full fine-tuning	6738M	26.6	8.45	6.84	4.85	11.69
LoRA [21]	320M	47.46	10.28	7.73	4.60	17.52
PiSSA [47]	320M	36.76	9.67	5.86	4.92	14.30
CorDA (ours)	320M	**50.34**	**14.43**	**8.17**	**5.05**	**19.50**

4.2 Knowledge-Preserved Adaptation Results

We fine-tune LLaMA-2-7B with full fine-tuning, LoRA, PiSSA, and our proposed CorDA on Math, Code, and Instruction Following tasks. In the knowledge-preserved adaptation mode, we randomly sample 256 questions from the NQ open training set and collect covariance matrices to initialize the adapters by Eq. (4). We report the fine-tuning performance and also the world knowledge performance of the fine-tuned model to manifest the overall ability of both new task learning and world knowledge maintaining. As shown in Table 1a, after fine-tuning on Math, all the three compared methods, full fine-tuning, LoRA, and PiSSA, suffer from drastic performance drop on TriviaQA and NQ open. Especially on NQ open, the ability is almost lost. PiSSA achieves the best accuracies on both GSM8k and Math, but meanwhile has the lowest results on the world knowledge benchmarks among the four methods. As a comparison, our method not only enjoys better fine-tuning performances than LoRA, but also achieves the best results on the three world knowledge benchmarks.

A similar pattern can be also observed in the fine-tuning of Code. As shown in Table 1b, full fine-tuning has the best ability on HumanEval and MBPP, but is the lowest on world knowledge performance. It is understandable that CorDA in knowledge-preserved adaptation is not as advantageous as PiSSA for fine-tuning performance, because PiSSA uses the largest singular values and their singular vectors as the adapter that dominates the weight update, while we keep them frozen and adapt the smallest components. Nevertheless, our method has the best average score in all the three fine-tuning tasks. A surprising result is achieved by our method in instruction following in Table



(a) $r = 128$ (b) $r = 32$

Figure 3: The training loss curves on MetaMath of full fine-tuning, LoRA, PiSSA, and CorDA with (a) rank 128 and (b) rank 32. The corresponding accuracies on GSM8k and Math are reported on the legends. Smoothing is performed for the loss curves.

Table 2: The experimental results of CorDA in the instruction-previewed adaptation mode on Math, Code, and Instruction Following tasks using LLaMA-2-7B. CorDA is initialized with samples from each of the fine-tuning datasets (MetaMathQA, CodeFeedback, and WizardLM-Evol-Instruct) for the three tasks, respectively. The rank r of LoRA, DoRA, PiSSA, and CorDA is 128. All methods are implemented by us under the same training and evaluation settings.

Method	#Params	GSM8k	Math	HumanEval	MBPP	MTBench	Avg
Full fine-tuning	6738M	48.9	7.48	**25.42**	**25.64**	4.85	22.46
LoRA [21]	320M	42.68	5.92	16.80	21.51	4.60	18.30
DoRA [42]	321M	41.77	6.20	16.86	21.60	4.48	18.18
PiSSA [47]	320M	51.48	7.60	19.48	23.84	4.92	21.46
CorDA (ours)	320M	**53.90**	**8.52**	21.03	24.15	**5.15**	**22.55**

1c, where CorDA surpasses all the three compared methods in both world knowledge performance and the new ability evaluated by MTBench. The world knowledge on NQ open is almost intact (14.43%) compared to the original performance (14.99%) before fine-tuning. These results reveal that our knowledge-preserved adaptation is an effective way to mitigate world knowledge forgetting and improve the overall ability.

4.3 Instruction-Previewed Adaptation Results

When the goal is to learn the target task as much as possible without concerning the loss of world knowledge, CorDA in the instruction-previewed adaptation mode satisfies this demand as it is able to further strengthen the fine-tuning performance. In this mode, we randomly sample instruction and response from the training data to collect the covariance matrices, and adopt the largest r components to initialize the adapters by Eq. (5). We compare our method with full fine-tuning, LoRA, DoRA, and PiSSA in the same three tasks, Math, Code, and Instruction Following. Compared with the knowledge-preserved adaptation in Table 1, CorDA in the instruction-previewed adaptation mode largely improves the fine-tuning performance on the five benchmarks, as shown in Table 2. Concretely, our method achieves the best performance on GSM8k, Math, MTBench, and the average score. Compared with PiSSA that adopts a similar adapter design but with no data context, our method has better results on all the five benchmarks. The training loss curves on Math are shown in Figure 3, where CorDA converges at a lower loss than LoRA and PiSSA in both $r = 128$ and $r = 32$. Only CorDA exhibits an obvious lower loss than full fine-tuning when $r = 128$. These results corroborate the benefits of data context to the initialized adapter, *i.e.,* the pre-captured task characteristic is able to accommodate the new ability and lead to a better performance.

We also apply our method to the General Language Understanding Evaluation (GLUE) benchmark [60] by fine-tuning the RoBERTa$_{\text{base}}$ model [43]. We adopt LoRA, DoRA, and our method with

Table 3: The experimental results of CorDA in the instruction-previewed adaptation mode on the GLUE benchmark using $\text{RoBERTa}_{\text{base}}$. CorDA is initialized with samples from each of the fine-tuning datasets. The rank r of LoRA, DoRA, and CorDA is 128. All methods are implemented by us under the same training and evaluation settings. Matthew's correlation and Pearson's correlation are the metrics of CoLA and STS-B, respectively. The metric of the other tasks is accuracy.

Method	#Params	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg
Full fine-tuning	125M	93.81	88.48	59.56	92.07	74.01	**90.49**	83.07
LoRA [21]	21M	**94.15**	82.84	54.24	92.48	64.26	88.58	79.43
DoRA [42]	21M	93.58	83.58	51.93	**92.59**	64.98	88.71	79.23
CorDA (ours)	21M	93.12	**89.71**	**59.60**	91.49	**76.17**	90.17	**83.38**

Table 4: Ablation experiments of the data choice used to collect covariance matrices and the adapter building manner in the knowledge-preserved adaptation mode. †: corresponds to the result of PiSSA that performs plain SVD and uses the largest r components to initialize the adapter.

Method	Context	Adapter	Trivia QA	NQ open	WebQS	GSM8k	Math	Avg
Plain SVD†	none	largest r	$39.71_{\pm 0.26}$	$1.02_{\pm 0.23}$	$6.30_{\pm 0.39}$	$51.48_{\pm 0.34}$	$7.60_{\pm 0.18}$	21.22
Plain SVD	none	smallest r	$39.94_{\pm 0.17}$	$4.21_{\pm 0.41}$	$6.25_{\pm 0.17}$	$43.29_{\pm 0.37}$	$5.96_{\pm 0.13}$	19.93
CO-SVD	Wikitext-2	smallest r	$42.93_{\pm 0.13}$	$7.20_{\pm 0.15}$	$6.40_{\pm 0.27}$	$42.99_{\pm 0.34}$	$5.80_{\pm 0.09}$	21.06
CO-SVD	Trivia QA	smallest r	$44.59_{\pm 0.34}$	$8.86_{\pm 0.20}$	$7.53_{\pm 0.14}$	$44.81_{\pm 0.28}$	$6.84_{\pm 0.16}$	22.53
CO-SVD	NQ open	smallest r	$44.30_{\pm 0.22}$	$9.36_{\pm 0.16}$	$7.14_{\pm 0.26}$	$44.58_{\pm 0.33}$	$6.92_{\pm 0.13}$	22.46

Table 5: The instruction following performance of CorDA using WizardLM-Evol-Instruct and Alpaca data to collect covariance matrices in the instruction-previewed adaptation mode.

Method	Context	MTBench
CorDA (ours)	WizardLM-Evol-Instruct	5.15
CorDA (ours)	Alpaca	5.06

a rank of 128 for all linear layers in the model except the classification head. For our method, we sample train data from each of the tasks to initialize adapters in the instruction-previewed mode and fine-tune the corresponding task. As shown in Table 3, our method achieves the best performance on the MRPC, CoLA, and RTE tasks, and the highest average score.

4.4 Discussions

Ablations. We ablate the data choice used to produce covariance matrices and the adapter building manner in Table 4. The first row corresponds to the implementation of PiSSA that uses the largest r singular values and their singular vectors as the initialized adapter by plain SVD. If we use the smallest r components by plain SVD to build adapters, there is no apparent improvement in world knowledge benchmarks. This implies that the plain SVD cannot precisely capture world knowledge related ability into the principal components and thus freezing them does not help to mitigate knowledge forgetting. When our context-oriented decomposition (CO-SVD) is adopted with Wikitext-2, which is not closely correlated with question answering, the performance on world knowledge is much improved. When the context collected by the covariance matrix is from question answering data, *i.e.,* TriviaQA or NQ open, the world knowledge performance is further improved by a significant margin, and the average score is also enhanced as a result. Therefore, data context is important to orientate the decomposition process such that the characteristics of the ability concerned can be better aggregated into the principal components for maintaining or adapting. As shown in Table 5, similar to TriviaQA and NQ open in the knowledge-preserved adaptation, collecting context from different data sources belonging to the same category, namely WizardLM-Evol-Instruct and Alpaca, also results in close performance in the instruction-previewed adaptation. It is noteworthy that CorDA with Alpaca on MTBench (5.06) is still the highest among the compared baselines in Table 2.

Limitations. The two adaptation modes developed in this paper highlight different aspects in usage. However, the knowledge-preserved mode, while being adept at maintaining world knowledge, is naturally not advantageous on fine-tuning performance compared with the instruction-previewed mode. How to develop an initialization strategy [68] for adapters combining the merits of the two modes to maximize both objectives deserves future exploration.

5 Conclusion

In this paper, we propose a new parameter-efficient fine-tuning method, named context-oriented decomposition adaptation (CorDA). It performs singular value decomposition for pre-trained weights oriented by the covariance matrix that captures the context of the task concerned, and aggregates the context into the principal components for maintaining or adapting. Accordingly, our method is able to support two implementation modes, the knowledge-preserved adaptation to mitigate world knowledge forgetting and the instruction-previewed adaptation for better fine-tuning performance. In experiments, our knowledge-preserved adaptation not only achieves better fine-tuning performance than LoRA, but also maintains the world knowledge well, leading to the best average scores on three fine-tuning tasks. Our instruction-previewed adaptation is able to further enhance the fine-tuning performance, surpassing the state-of-the-art parameter-efficient fine-tuning methods DoRA and PiSSA.

Acknowledgments and Disclosure of Funding

The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) - Center of Excellence for Generative AI, under award number 5940.

This work was also supported in part by the National Natural Science Foundation of China under Grant 62376069, in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001, and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515012027.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] A. Aghajanyan, L. Zettlemoyer, and S. Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[3] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013.

[5] F. Bie, Y. Yang, Z. Zhou, A. Ghanem, M. Zhang, Z. Yao, X. Wu, C. Holmes, P. Golnari, D. A. Clifton, et al. Renaissance: A survey into ai text-to-image generation in the era of large model. *arXiv preprint arXiv:2309.00810*, 2023.

[6] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*, 2019.

[7] A. Chavan, Z. Liu, D. Gupta, E. Xing, and Z. Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023.

[8] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[9] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *NeurIPS*, 36, 2023.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[13] S. Dou, E. Zhou, Y. Liu, S. Gao, J. Zhao, W. Shen, Y. Zhou, Z. Xi, X. Wang, X. Fan, et al. Loramoe: Alleviate world knowledge forgetting in largelanguage models via moe-style plugin. *arXiv preprint arXiv:2312.09979*, 2023.

[14] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[15] K. Gupta, B. Thérien, A. Ibrahim, M. L. Richter, Q. G. Anthony, E. Belilovsky, I. Rish, and T. Lesort. Continual pre-training of large language models: How to re-warm your model? In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023.

[16] J. He, H. Guo, M. Tang, and J. Wang. Continual instruction tuning for large multimodal models. *arXiv preprint arXiv:2311.16206*, 2023.

[17] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[19] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019.

[20] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, pages 2790–2799. PMLR, 2019.

[21] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[22] A. Ibrahim, B. Thérien, K. Gupta, M. L. Richter, Q. Anthony, T. Lesort, E. Belilovsky, and I. Rish. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*, 2024.

[23] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

[24] R. Karimi Mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *NeurIPS*, 34:1022–1035, 2021.

[25] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[26] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano. Vera: Vector-based random matrix adaptation. In *ICLR*, 2024.

[27] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[28] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *AAAI*, volume 38, pages 13355–13364, 2024.

[29] K. Lee, M.-W. Chang, and K. Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[30] T. Lei, J. Bai, S. Brahma, J. Ainslie, K. Lee, Y. Zhou, N. Du, V. Y. Zhao, Y. Wu, B. Li, et al. Conditional adapters: Parameter-efficient transfer learning with fast inference. In *NeurIPS*, 2023.

[31] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059, 2021.

[32] C. Li, H. Farkhoor, R. Liu, and J. Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

[33] X. Li, S. He, J. Wu, Y. Yu, L. Nie, and M. Zhang. Mask again: Masked knowledge distillation for masked video modeling. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 2221–2232, 2023.

[34] X. Li, J. Wu, S. He, S. Kang, Y. Yu, L. Nie, and M. Zhang. Fine-grained key-value memory enhanced predictor for video representation learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 2264–2274, 2023.

[35] X. Li, Y. Yang, X. Li, J. Wu, Y. Yu, B. Ghanem, and M. Zhang. Genview: Enhancing view quality with pretrained generative model for self-supervised learning. In *ECCV*, 2024.

[36] X. Li, Y. Yang, J. Wu, B. Ghanem, L. Nie, and M. Zhang. Mamba-fscil: Dynamic adaptation with selective state space model for few-shot class-incremental learning. *arXiv preprint arXiv:2407.06136*, 2024.

[37] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, 2021.

[38] Y. Li, Y. Yu, C. Liang, N. Karampatziakis, P. He, W. Chen, and T. Zhao. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *ICLR*, 2024.

[39] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[40] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

[41] Q. Liu, X. Wu, X. Zhao, Y. Zhu, D. Xu, F. Tian, and Y. Zheng. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*, 2023.

[42] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[44] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, volume 30, 2017.

[45] R. K. Mahabadi, S. Ruder, M. Dehghani, and J. Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, 2021.

[46] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[47] F. Meng, Z. Wang, and M. Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

[48] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[49] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021.

[50] Z. Qiu, W. Liu, H. Feng, Y. Xue, Y. Feng, Z. Liu, D. Zhang, A. Weller, and B. Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *NeurIPS*, volume 36, pages 79320–79362, 2023.

[51] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[52] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell. Continual unsupervised representation learning. In *NeurIPS*, volume 32, 2019.

[53] A. Razdaibiedina, Y. Mao, M. Khabsa, M. Lewis, R. Hou, J. Ba, and A. Almahairi. Residual prompt tuning: improving prompt tuning with residual reparameterization. In *ACL*, pages 6740–6757, 2023.

[54] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.

[55] A. Renduchintala, T. Konuk, and O. Kuchaiev. Tied-lora: Enhacing parameter efficiency of lora with weight tying. *arXiv preprint arXiv:2311.09578*, 2023.

[56] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, , and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

[57] T. Scialom, T. Chakrabarty, and S. Muresan. Fine-tuned language models are continual learners. In *EMNLP*, 2022.

[58] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[59] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

[60] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

[61] C. Wu, Y. Gan, Y. Ge, Z. Lu, J. Wang, Y. Feng, P. Luo, and Y. Shan. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*, 2024.

[62] T. Wu, L. Luo, Y.-F. Li, S. Pan, T.-T. Vu, and G. Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

[63] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, Q. Lin, and D. Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *ICLR*, 2024.

[64] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

[65] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. ZHANG, and Q. Tian. QA-loRA: Quantization-aware low-rank adaptation of large language models. In *ICLR*, 2024.

[66] S. Yan, J. Xie, and X. He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021.

[67] Y. Yang, X. Li, M. Alfarra, H. A. A. K. Hammoud, A. Bibi, P. Torr, and B. Ghanem. Towards interpretable deep local learning with successive gradient reconciliation. In *ICML*, 2024.

[68] Y. Yang, H. Wang, H. Yuan, and Z. Lin. Towards theoretically inspired neural initialization optimization. In *NeurIPS*, volume 35, pages 18983–18995, 2022.

[69] Y. Yang, H. Yuan, X. Li, Z. Lin, P. Torr, and D. Tao. Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning. In *ICLR*, 2023.

[70] Y. Yang, H. Yuan, X. Li, J. Wu, L. Zhang, Z. Lin, P. Torr, D. Tao, and B. Ghanem. Neural collapse terminus: A unified solution for class incremental learning and its variants. *arXiv preprint arXiv:2308.01746*, 2023.

[71] L. Yu, W. Jiang, H. Shi, J. YU, Z. Liu, Y. Zhang, J. Kwok, Z. Li, A. Weller, and W. Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024.

[72] L. Yu, B. Yu, H. Yu, F. Huang, and Y. Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

[73] Z. Yuan, Y. Shang, Y. Song, Q. Wu, Y. Yan, and G. Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

[74] Y. Zhai, S. Tong, X. Li, M. Cai, Q. Qu, Y. J. Lee, and Y. Ma. Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In *Conference on Parsimony and Learning (Proceedings Track)*, 2024.

[75] F. Zhang, L. Li, J. Chen, Z. Jiang, B. Wang, and Y. Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*, 2023.

[76] J. Zhang, J. Liu, J. He, et al. Composing parameter-efficient modules with arithmetic operation. *NeurIPS*, 36:12589–12610, 2023.

[77] M. Zhang, C. Shen, Z. Yang, L. Ou, X. Yu, B. Zhuang, et al. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023.

[78] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*, 2023.

[79] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *ICML*, 2024.

[80] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, volume 36, 2023.

[81] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[82] T. Zheng, G. Zhang, T. Shen, X. Liu, B. Y. Lin, J. Fu, W. Chen, and X. Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

[83] D. Zhu, Z. Sun, Z. Li, T. Shen, K. Yan, S. Ding, K. Kuang, and C. Wu. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. In *ICML*, 2024.

[84] W. Zhu and M. Tan. SPT: Learning to selectively insert prompts for better prompt tuning. In *EMNLP*, 2023.

[85] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Table 6: The detailed numbers and more results of the experiment in Figure 2.

Test Data	Method	discarded ranks							
		0	16	32	64	128	256	512	1024
Wikitext-2	Plain SVD	5.47	6.32	7.31	9.89	18.03	18.5	25.42	73.92
	ASVD [73] (with 256 Wiki samples)	5.47	6.08	6.67	7.86	8.71	9.92	12.37	20.34
	CO-SVD (with 32 Wiki samples)	5.47	5.48	5.48	5.49	5.52	5.58	5.79	6.62
	CO-SVD (with 256 Wiki samples)	5.47	5.48	5.48	5.48	5.5	5.54	5.69	**6.35**
	CO-SVD (with 256 PTB samples)	5.47	5.49	5.5	5.52	5.57	5.74	6.25	8.69
PTB	Plain SVD	20.82	35.25	33.42	37.46	55.47	70.25	98.6	763.44
	ASVD [73] (with 256 PTB samples)	20.84	33.42	32.05	31.67	35.36	40.23	51.28	93.42
	CO-SVD (with 32 PTB samples)	20.75	20.75	20.76	20.78	20.83	20.91	21.17	22.68
	CO-SVD (with 256 PTB samples)	20.88	20.88	20.88	20.89	20.91	20.94	21.14	**22.28**
	CO-SVD (with 256 Wiki samples)	20.34	20.34	20.32	20.41	20.59	21.25	22.94	29.69

A Appendix: Implementation Details

A.1 Fientuning on Math, Code, and Instruction Following

For fine-tuning tasks on Math, Code, and Instruction Following, we adopt the same training setting as PiSSA [47]. Concretely, optimization is performed with the AdamW optimizer, a batch size of 128, and a learning rate of 2e-5. We employ cosine annealing schedules with a warmup ratio of 0.03 and do not apply weight decay. Training is conducted exclusively on the first 100,000 conversations from the dataset for one epoch, with loss computation solely based on the response. Our experiments are executed on the NVIDIA A100-SXM4(40/80GB) GPUs. Publicly available platforms are utilized for the evaluation of world knowledge (TriviaQA, NQ open, and Web QS) [2], Code (HumanEval and MBPP) [3], and Instruction Following (MTBench) [4].

A.2 GLUE Benchmark

To ensure fair comparison across Full fine-tuning, LoRA, DoRA, and CorDA in the GLUE benchmark, we implement all methods under the same training and evaluation settings. The AdamW optimizer is used with a batch size of 32 and a learning rate of 4e-5 for 3 epochs, following a linear learning rate schedule. The max token length is set as 128. The rank of LoRA, DoRA, and our CorDA is 128. For covariance matrix collection of CorDA, we concatenate the representative content of each training sample to form a text sequence. From this sequence, 256 text segments, each containing 256 tokens, are randomly sampled. The selected content for each task is as follows: MRPC, RTE, and STS-B using "sentence1", CoLA and SST-2 using "sentence", and QNLI using "question". All methods are trained on a single NVIDIA A100-SXM4(40/80GB) GPU.

B Appendix: More Results

Table 6 lists the detailed numbers and more results of the experiment in Figure 2. It is shown that the number of sampled data only has a very limited impact. When the smallest 1024 ranks are discarded, using 32 samples is slightly worse than 256 samples in both Wikitext-2 and PTB. It implies that a small number of samples is enough to capture context into the principal components. Besides, collecting samples from the same dataset as the one used to test is able to attain a better performance after discarding a large number of ranks. For example, when discarding the smallest 1024 ranks, CO-SVD (with 256 Wiki samples) is better than CO-SVD (with 256 PTB samples) on Wikitext-2 (6.35 v.s. 8.69), and CO-SVD (with 256 PTB samples) is better than CO-SVD (with 256 Wiki samples) on PTB (22.28 v.s. 29.69). This also reveals that precisely capturing the data context in our decomposition is crucial for better maintaining the task characteristics into the principal components, and explains why our method is superior to the PEFT methods without considering data context.

[2] https://github.com/EleutherAI/lm-evaluation-harness
[3] https://github.com/bigcode-project/bigcode-evaluation-harness
[4] https://github.com/lm-sys/FastChat

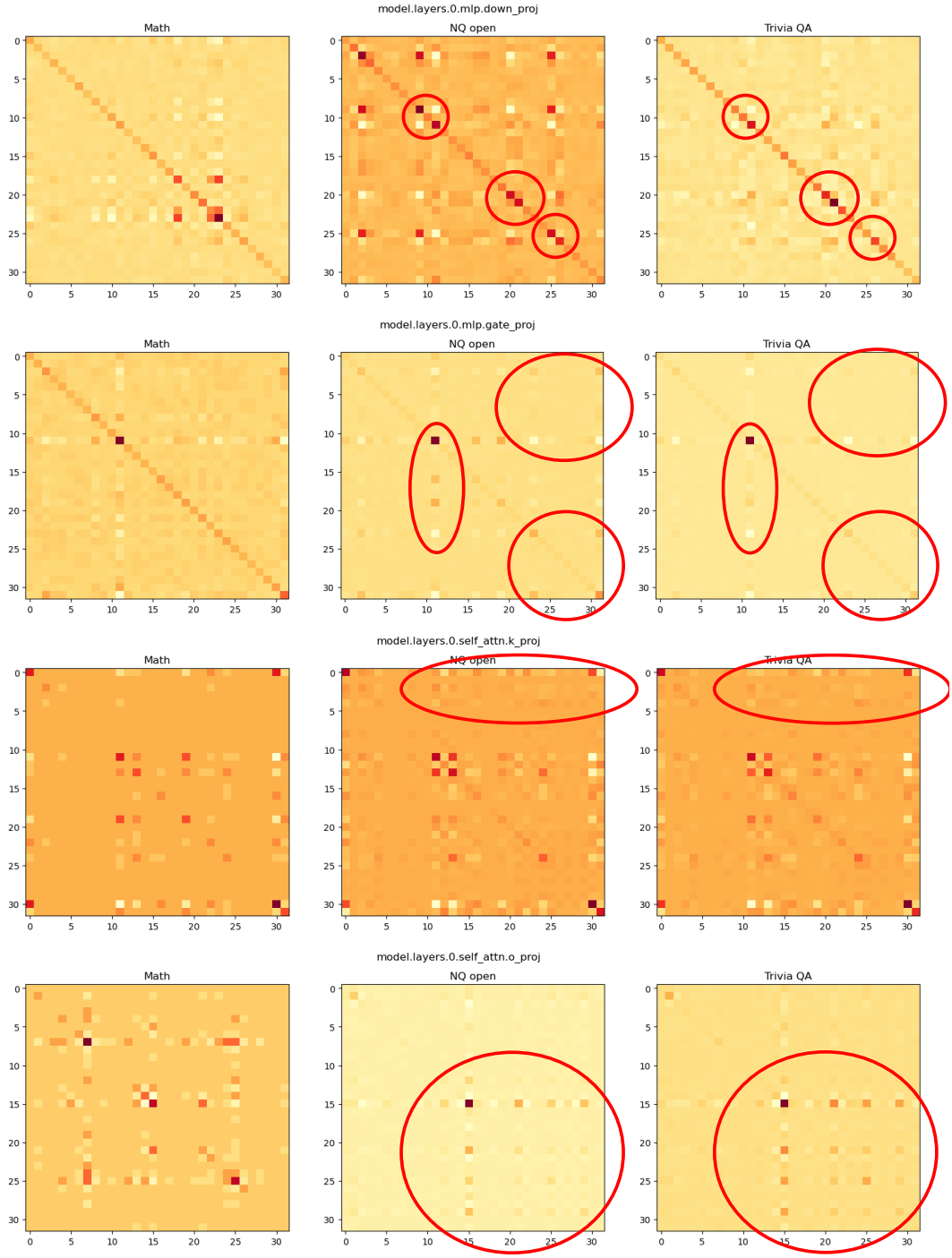


Figure 4: Covariance visualization results for “self_attn.k_proj”, “self_attn.o_proj”, “mlp.down_proj”, and “mlp.gate_proj” weights in the 0-th layer. Please zoom in for a better view.

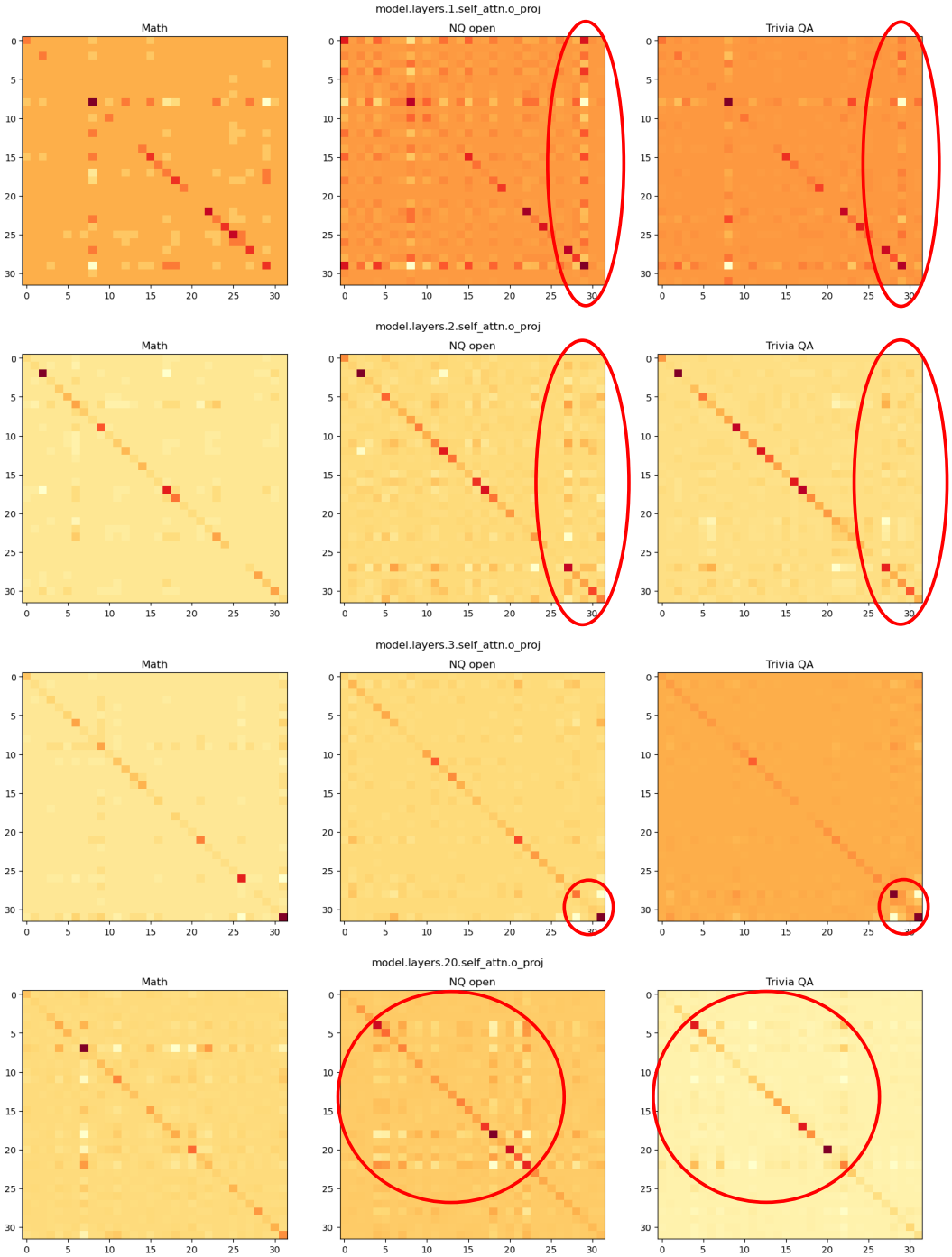


Figure 5: Covariance visualization results for “self_attn.o_proj” weights in different depth layers. Please zoom in for a better view.

C Appendix: Covariance Visualization Results

We provide the visualization results of the covariance matrices collected from three tasks MetaMath, NQ open, and Trivia QA in Figures 4 and 5.

Since the original dimension in 4096 or 11008 will be too large to be informative, we down-sample the covariance matrices into 32×32 and visualize their heatmaps. We provide the results from the activations before different linear weights including `self_attn.k_proj` (the same as `self_attn.q_proj` and `self_attn.v_proj` due to the same input), `self_attn.o_proj`, `mlp.down_proj`, and `mlp.gate_proj` (the same as `mlp.up_proj`) in the first layer, and the `self_attn.o_proj` weight in later layers. It is shown that the heatmaps from NQopen and TriviaQA (both are QA tasks) share some similar patterns (marked in red circles), which do not appear in the heatmap from the different task MetaMath. Therefore, when the inputs of different tasks are fed into an LLM, the covariance matrix from activations will exhibit different patterns. The visualization result empirically supports that the covariance matrix patterns can be used to characterize the triggered task. We use such patterns to orientate the decomposition of LLM pretrained weights, to make the resulting adapter initialization task-dependent.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our Introduction section does not claim any contribution out of the scope. All the contributions mentioned are supported in the later sections, see Sec. 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in Sec. 4.4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theory and proof are concerned by this study.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The complete implementation details to produce our experimental results are described in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and trained models are publicly available with detailed instructions to implement our method.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify all the training and test details in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean and standard deviation values in Table 4 by running each result 3 times with different seeds. For the other results, the repeated experiment of full fine-tuning will consume too much time and resource.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the type of GPU in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We obey the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our study is a parameter-efficient fine-tuning method for better fine-tuning performance and mitigating knowledge forgetting. No societal impact is concerned by our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the papers whose datasets are used in our experiments. We also explicitly mention the code used for evaluation in Appendix A.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No experiment involving crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No experiment involving crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.