SLoRA: Balancing Plasticity and Forgetting in Large Language Models for Continual Learning

Lina Yang, Yusheng Liao, Yanfeng Wang, Yu Wang*
Shanghai Jiao Tong University
{alina_yln, liao20160907, wangyanfeng, yuwangsjtu}@sjtu.edu.cn

Abstract

Large language models (LLMs) have achieved remarkable success across diverse tasks through large-scale pretraining. However, they remain prone to catastrophic forgetting in continual learning. To the best of our knowledge, this is the first work to identify noise accumulation in LoRA updates as a key cause of forgetting in continual learning. A preliminary two-task experiment demonstrates that removing less important components of the second task's LoRA parameters improves performance on the first task, suggesting that later updates introduce noisy interference. Building on this insight, we propose **S**ubspace-Denoised **Lo**w-**R**ank **A**daptation (**SLoRA**), a simple and effective framework that filters noisy components from LoRA updates via subspace similarity with the base model. SLoRA is a regularization-free method without accessing data or gradients from previous tasks or modifying the training process. It offers two variants, SLoRA-Pre and SLoRA-Post, for online and offline continual learning, respectively. Extensive experiments across tasks and models validate the effectiveness of SLoRA. It improves final accuracy by up to 12%, reduces forgetting by 29%, and filters out over 30% of LoRA parameters identified as noisy. Our code is available at https://github.com/alina1031/SLoRA.

1 Introduction

Large language models (LLMs) have revolutionized artificial intelligence, achieving unprecedented performance in reasoning, generation, and generalization across diverse tasks (Brown et al., 2020; Achiam et al., 2023; Chowdhery et al., 2023). However, these models are typically trained under static task distributions and lack mechanisms for continual adaptation (Dhingra et al., 2022; Jang et al., 2022). In real-world scenarios, such as personalized assistants, domain-specific agents, or evolving user preferences, LLMs are expected to acquire new capabilities over time (Winata et al., 2023). This continual learning (CL) setting poses a significant challenge: *catastrophic forgetting* (McCloskey and Cohen, 1989), where learning new tasks interferes with previously acquired knowledge and degrades performance on earlier tasks.

To alleviate the catastrophic forgetting phenomenon, classical CL approaches attempt to add regularization during the training process (Kirkpatrick et al., 2017; Zenke et al., 2017; Wang et al., 2023a; Wu et al., 2024b). For example, GEM (Lopez-Paz and Ranzato, 2017) constrains updates using gradients stored from previous tasks, and O-LoRA (Wang et al., 2023a) restricts parameter updates to an orthogonal subspace. However, we found that such regularization prevents forgetting at the cost of reducing the LLMs' ability to learn new tasks (as shown in Sec. 6.2). Recent methods like SD-LoRA (Wu et al., 2025), while not explicitly imposing regularization, still restrict the learning of parameters at the subspace level, which is considered a form of implicit constraint. These findings lead to a question: *Can we mitigate catastrophic forgetting without restricting the LLMs' capacity for learning?*

In light of this challenge, we consider a regularization-free method to circumvent the limitation of the regularization-based methods in restricting the LLMs' learning capacity. This methodology is motivated by prior work showing that update parameters contain a significant amount of redundancy and noise (Zhang et al., 2025; Jiang et al., 2024, 2025). We hypothesize that the less important components of the new task are noise, which interfere with the knowledge retained from previous tasks, leading to catastrophic forgetting. To validate this hypothesis, we conduct a controlled pre-experiment where an LLM is sequentially trained on the Amazon Reviews (*Task1*) and Yahoo Answers (*Task2*) (details are shown in Appendix B.3).

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5437–5454

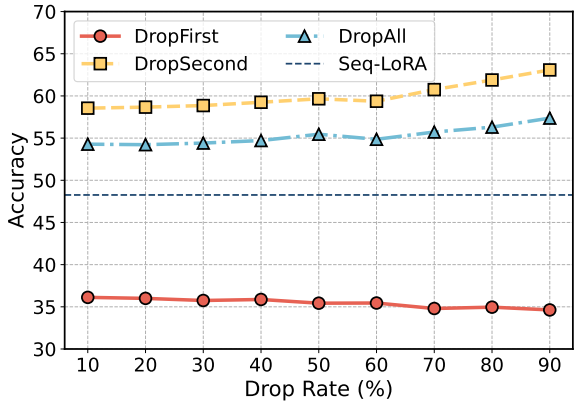


Figure 1: Performance of *Task1* under LoRA dropping with Seq-LoRA as the reference, including DropFirst, DropSecond, and DropAll. DropFirst degrades performance, while DropSecond yields clear improvements.

After training, we proportionally reduce the less important components in the LoRA parameters using three conditions: DropFirst (from *Task1*), DropSecond (from *Task2*), and DropAll (from both), ranging from 10% to 90%. As shown in Figure 1, dropping a moderate portion of *Task2* parameters raises *Task1* accuracy above the Seq-LoRA reference, whereas DropFirst consistently degrades *Task1*. This indicates that LoRA updates from later tasks introduce noisy components that interfere with knowledge retained from earlier tasks, leading to our key insight: *catastrophic forgetting can be mitigated by removing noisy components from LoRA updates to reduce cross-task interference*.

Motivated by this insight, we propose **S**ubspace-Denoised **Lo**w-**R**ank **A**daptation (SLoRA), a novel framework designed to mitigate catastrophic forgetting by selectively removing noisy components based on subspace similarity. At its core, SLoRA utilizes Singular Value Decomposition on each LoRA update. It then retains components that exhibit higher similarity to the base model's representation space, treating low-similarity components as noise to be discarded. We instantiate two distinct variants: **SLoRA-Pre**, which performs incremental denoising after each task, making it ideal for online continual learning settings; and **SLoRA-Post**, which performs batch denoising after all tasks, suitable for offline settings. Crucially, SLoRA is lightweight, requires no rehearsal data or storage of past gradients, and integrates seamlessly without any changes to the standard training process.

In summary, our contributions are as follows:

- We are the first to identify noise accumulation in LoRA as a key factor causing catastrophic forgetting. To address this, we propose a method to remove noise from LoRA parameters, significantly reducing interference caused by the following updated noise.

- Based on this finding, we propose **SLoRA**, a novel regularization-free method with two variants (SLoRA-Pre and SLoRA-Post). Our approach is efficient and practical as it requires no past data or gradients and no modifications to the training process.

- We empirically demonstrate that SLoRA outperforms all baselines on various continual learning tasks. Notably, on the TRACE benchmark, SLoRA improves final accuracy by up to 12% and reduces forgetting by up to 29% by removing over 30% of LoRA parameters.

2 Related Works

Continual Learning Continual Learning aims to enable models to learn from a sequence of tasks while retaining previously acquired knowledge and minimizing catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990; Wu et al., 2024a). Existing methods are broadly categorized into three families: (1) **Rehearsal-based methods** (Rolnick et al., 2019; Lopez-Paz and Ranzato, 2017; Han et al., 2020) utilize memory buffers or generative models to revisit data from prior tasks, which often introduces significant memory or computational overhead. (2) **Regularization-based methods** (Kirkpatrick et al., 2017; Li and Hoiem, 2017; Zenke et al., 2017) prevent forgetting by penalizing updates to important parameters. Notable examples include Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), which estimates parameter importance using the Fisher Information Matrix, and Orthogonal Gradient Descent (OGD) (Farajtabar et al., 2020), which constrains new gradients to be orthogonal to those of prior tasks. (3) **Architecture-based approaches** (Rusu et al., 2016; Yoon et al., 2018; Li et al., 2019) isolate task-specific components to mitigate interference, such as Progressive Prompts (Razdaibiedina et al., 2023) which assign soft prompts per task. While effective in small-scale settings, these classical approaches often struggle with scalability or parameter growth in high-dimensional LLM environments.

Parameter-Efficient Fine-Tuning PEFT has emerged as a practical alternative to full fine-tuning by adapting only a small, carefully selected subset of parameters (Houlsby et al., 2019; Zaken et al., 2022; Ding et al., 2023). Beyond adapters (Houlsby et al., 2019) and prompt-based tuning (Li and Liang, 2021; Lester et al., 2021), LoRA (Hu et al., 2021) has gained significant popularity by injecting trainable low-rank matrices into frozen weights. Recent variants further optimize this process by dynamically allocating rank budgets based on weight sensitivity (AdaLoRA) (Zhang et al., 2023) or leveraging principal and minor singular components (Pissa, MiLoRA) (Meng et al., 2024; Wang et al., 2024; Liang and Li, 2024). Despite their efficiency, standard PEFT updates still suffer from interference and forgetting when applied to sequential task adaptation.

PEFT for Continual Learning Increasingly, PEFT methods are extended to CL to achieve efficient adaptation while maintaining stability (Song et al., 2023; Qiao and Mahdavi, 2024). For example, O-LoRA (Wang et al., 2023a) enforces orthogonality between LoRA updates across tasks, while LB-CL (Qiao and Mahdavi, 2024) combines low-rank decomposition with sensitivity-based initialization. ConPET (Song et al., 2023) introduces a memory replay mechanism to manage parameter reuse. While concurrent work LoRI (Zhang et al., 2025) explores subspace filtering to reduce cross-task interference in multi-task learning, it applies filtering during training with access to all tasks. In contrast, our **SLoRA** is a **regularization-free** framework that applies subspace-based denoising *post-hoc*. Unlike prior methods relying on external modules such as prompt routing (Wang et al., 2022), reasoning traces (Wang et al., 2023b), or teacher-student distillation (Zhong et al., 2024), SLoRA filters noisy components directly via subspace similarity with the base model, offering a simple, modular, and scalable solution for long task sequences.

3 Preliminaries

Continual Learning Setup. We consider a standard continual learning scenario where a pretrained LLM is adapted to a sequence of tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$. Each task $\mathcal{T}_k$ is associated with a specific data distribution $\mathcal{D}_k$. The objective is to sequentially acquire new knowledge from the current task $\mathcal{T}_k$ while maintaining performance on all previously learned tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{k-1}\}$, ensuring that the model parameters evolve without suffering from catastrophic forgetting.

Low-Rank Adaptation for Continual Learning. Given the immense computational cost of full fine-tuning, we adopt LoRA (Hu et al., 2021) as the backbone for efficient adaptation. We start with a pretrained LLM parameterized by frozen weights θ_0. LoRA injects trainable rank decomposition matrices into a specific subset of modules, denoted by $\mathcal{M}$. For a specific module $m \in \mathcal{M}$, let $\theta_0^m \in \mathbb{R}^{d' \times d}$ represent its frozen base weight. The adaptation for the k-th task is parameterized by a low-rank update $\Delta\theta_k^m = B_k^m A_k^m$, where $B_k^m \in \mathbb{R}^{d' \times r}$ and $A_k^m \in \mathbb{R}^{r \times d}$ are trainable matrices with rank $r \ll \min(d', d)$.

In a sequential fine-tuning setting (Seq-LoRA), the model parameters are updated incrementally. After training on T tasks, the final adapted weight for module m is obtained by accumulating the updates from all tasks:

$$\theta_T^m = \theta_0^m + \sum_{k=1}^{T} \Delta\theta_k^m, \quad m \in \mathcal{M}, \tag{1}$$

where $\Delta\theta_k^m$ denotes the raw LoRA update learned for task $\mathcal{T}_k$. Although mathematically concise, this direct accumulation often leads to interference, as $\Delta\theta_k^m$ may contain components that conflict with the knowledge encoded in θ_0^m or prior updates. Unlike previous works that add regularization terms to the training objective to constrain $\Delta\theta_k^m$, our method, SLoRA, focuses on identifying and filtering out noisy components within $\Delta\theta_k^m$ directly via subspace analysis, as detailed in the following section.

4 SLoRA: Subspace-Denoised LoRA

4.1 Method Overview

SLoRA is a post-hoc denoising framework designed to improve the robustness of continual learning by removing noisy components from task-specific updates. As illustrated in Figure 2, it integrates into standard LoRA fine-tuning pipelines without modifying the base model or accessing to data and gradients from previous tasks.

Consider a pretrained LLM with frozen parameters θ_0 and a set of target modules $\mathcal{M}$. For each incoming task $\mathcal{T}_k$, low-rank adapters $\{\Delta\theta_k^m\}_{m \in \mathcal{M}}$ are trained on top of the base weights $\{\theta_0^m\}_{m \in \mathcal{M}}$. Each update $\Delta\theta_k^m \in \mathbb{R}^{d' \times d}$ is typically formed by the product of two low-rank matrices of rank r,

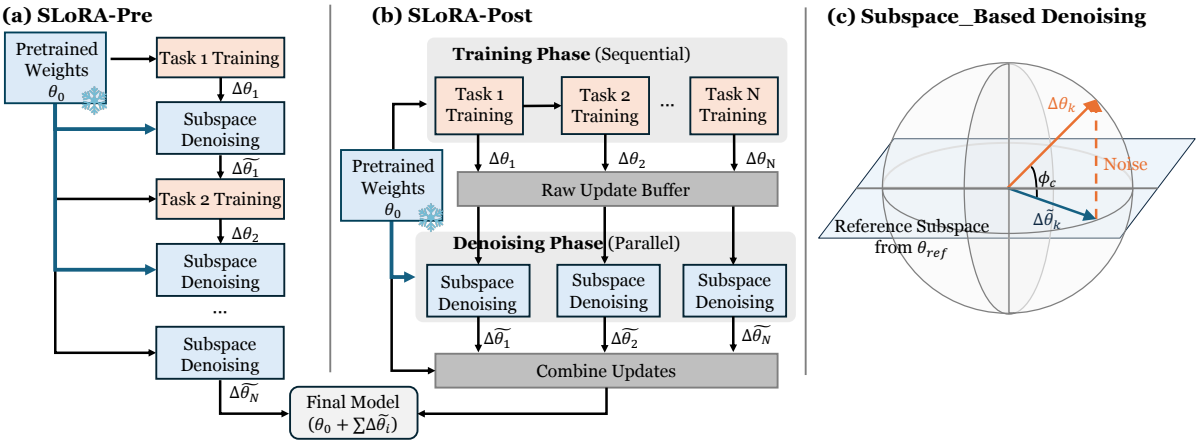


Figure 2: Overview of the SLoRA framework. SLoRA mitigates catastrophic forgetting by removing noise in LoRA updates based on subspace similarity to the base model. SLoRA-Pre performs denoising after each task, while SLoRA-Post applies it once after all tasks. Components with low similarity to the base subspace are dropped, and the remaining parts form the denoised update $\Delta\tilde{\theta}_k$.

typically with $r \ll \min(d', d)$. Crucially, instead of directly accumulating these raw updates, they are processed through a subspace-based denoising module. This module selectively filters components that are identified as noise, yielding denoised updates $\{\Delta\widetilde{\theta}_k^m\}$. The final parameters used for inference after K tasks are composed as:

$$\theta_K^m = \theta_0^m + \sum_{i=1}^{K} \Delta\widetilde{\theta}_i^m, \quad m \in \mathcal{M}, \tag{2}$$

where θ_K^m denotes the adapted weight of module m after K tasks, and $\Delta\widetilde{\theta}_i^m$ is the denoised LoRA update for each task $\mathcal{T}_i$. The framework operates in two variants, SLoRA-Pre and SLoRA-Post, as detailed in Section 4.4.

4.2 Automatic Subspace-Based Denoising

The core mechanism of SLoRA is the adaptive denoising procedure, which filters task-specific updates based on their alignment with a reference subspace, as shown in Figure 2(c). This approach is motivated by the observation that the principal singular components of pre-trained weights encapsulate the model's essential capabilities (Meng et al., 2024). We hypothesize that update components exhibiting high similarity to the base subspace represent meaningful adaptations that preserve general knowledge, whereas components with low similarity are likely task-specific noise or redundant deviations that exacerbate catastrophic forgetting. We formally define the denoising procedure below, omitting the module superscript m for clarity.

Definition 1 (Subspace-Based Adaptive Denoising). *Let $\Delta\theta_k \in \mathbb{R}^{d' \times d}$ be the raw update for task k. Singular Value Decomposition (SVD) is performed on the update such that $\Delta\theta_k = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices containing singular vectors and $\mathbf{\Sigma}$ is the diagonal matrix of singular values. Let θ_{ref} be the reference weights with principal basis $\mathbf{U}_{ref}$. Let $\mathcal{C} \subseteq \{1, \ldots, r\}$ denote a set of candidate ranks. For any rank $c \in \mathcal{C}$, the Grassmannian similarity ϕ_c is calculated as:*

$$\phi_c = \left\| (\mathbf{U}_{ref,c})^\top \mathbf{U}_c \right\|_F^2, \tag{3}$$

where $\mathbf{U}_c$ and $\mathbf{U}_{ref,c}$ represent the top-c left singular vectors of the update and the reference weights, respectively. The optimal rank c^* is automatically selected by maximizing this similarity score:

$$c^* = \arg\max_{c \in \mathcal{C}} \phi_c. \tag{4}$$

Consequently, the Denoised Update is reconstructed using the top-c^* singular components:

$$\Delta\widetilde{\theta}_k = \mathbf{U}_{c^*}\, \mathbf{\Sigma}_{c^*} (\mathbf{V}_{c^*})^\top, \tag{5}$$

where $\mathbf{U}_{c^*} \in \mathbb{R}^{d' \times c^*}$ and $\mathbf{V}_{c^*} \in \mathbb{R}^{d \times c^*}$ denote the sub-matrices containing the top-c^* left and right singular vectors, respectively, and $\mathbf{\Sigma}_{c^*} \in \mathbb{R}^{c^* \times c^*}$ is the diagonal matrix of the corresponding top-c^* singular values.

This subspace-based denoising removes noisy components with low similarity to the reference subspace, preserving only those that are more likely to contribute to effective knowledge transfer across tasks. To implement this efficiently, we utilize the randomized SVD (see Appendix C.2 for details).

4.3 Reference Subspace Selection

A critical component of the proposed denoising mechanism is the choice of the reference weight θ_{ref}. We select the fixed pre-trained base model θ_0 as the reference anchor. From a theoretical perspective, relying on the previous model θ_{t-1} as a reference introduces a risk of subspace drift, where the principal directions of new updates progressively diverge from the model's original knowledge structure. Our analysis demonstrates that the alignment between the current update and historical subspaces decays geometrically when the reference shifts sequentially. In contrast, anchoring the denoising process to θ_0 enforces a global consistency constraint, ensuring that the principal subspaces of learned updates maintain stable alignment throughout the continual learning sequence. We provide the formal definitions and detailed proofs of this alignment stability analysis in Appendix A.

4.4 Variants of SLoRA

Depending on the training setting, SLoRA operates in two variants as shown in Figure 2. **SLoRA-Pre** (Online) performs subspace-based denoising immediately after training task $\mathcal{T}_k$ (Figure 2(a)), integrating the denoised update $\Delta\widetilde{\theta}_k$ before proceeding to the next task. This is tailored for standard online continual learning. Conversely, **SLoRA-Post** (Offline) buffers raw updates from all tasks (Figure 2(b)) and performs denoising in parallel using θ_0 as the reference. This suits scenarios where raw checkpoints are preserved. Detailed pseudocode for both algorithms is provided in Appendix C.1.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on the TRACE benchmark (Wang et al., 2023b), which covers diverse tasks including domain-specific QA, summarization, multilingual classification, and reasoning. To ensure a rigorous continual learning evaluation, we follow a fixed sequential training protocol on the eight TRACE tasks in the following order: (1) C-STANCE, (2) FOMC, (3) MeetingBank, (4) Py150, (5) ScienceQA, (6) NumGLUE-cm, (7) NumGLUE-ds, and (8) 20Minuten. This sequence allows us to evaluate the model's ability to retain knowledge across significant domain and task shifts. Furthermore, we evaluate on a **standard CL benchmark** consisting of five text classification datasets (AG News, Amazon, Yelp, DBpedia, and Yahoo Answers) under three different task orders. Detailed statistics and task orders are provided in Appendix B.1.

Evaluation Metrics. We follow the evaluation metrics specified in the TRACE benchmark. In our experiments, we compare the performance of each task individually across the sequence, as well as report the average performance across all tasks. The task-specific evaluation metrics are consistent with those defined in TRACE: accuracy for tasks like question answering and classification, ROUGE-L for summarization tasks, SARI for text simplification, and Edim similarity for code completion. For the standard CL benchmark, we evaluate the accuracy and report the average accuracy over all tasks at different orders.

Baselines. We compare SLoRA against baselines categorized into four groups. SLoRA belongs to the *Regularization-free* category, mitigating interference by denoising the LoRA subspace without extra constraints. The details are as follows:

(1) *Oracle* methods: **STL** (Single-Task Learning) trains one task in isolation and **MTL** (Multi-Task Learning) jointly trains on all tasks. Both methods represent the upper-bound performance in the continual learning setting.

(2) *Vanilla* methods: **Base** uses the pretrained model without any adaptation, and **ICL** performs 6-shot in-context learning without parameter updates.

(3) *Regularization-based* methods: **EWC** (Kirkpatrick et al., 2017) and **LwF** (Li and Hoiem, 2017) preserve knowledge via parameter penalization and distillation, respectively. **GEM** (Lopez-Paz and Ranzato, 2017) constrains gradients, **O-LoRA** (Wang et al., 2023a) enforces subspace orthogonality, and **SD-LoRA** (Wu et al., 2025) freezes learned directions while updating magnitudes.

(4) *Regularization-free* methods: **Seq-LoRA** applies sequential LoRA fine-tuning per task as a simple continual baseline, and **RCL** (Wang et al., 2023b) augments training data with reasoning traces generated by GPT-4.

Implementation Details. For the TRACE benchmark, we conduct experiments on Llama3-8B-Instruct, Llama3.1-8B-Instruct, Qwen2.5-7B-

Base Model	Methods	C-STANCE	FOMC	MTB.	Py150	Sci.QA	NumG.cm	NumG.ds	20Minuten	Avg.
Llama3.1-8B	STL†	56.30	68.55	61.22	70.72	93.45	70.37	73.23	41.83	66.96
	MTL†	57.90	66.94	59.72	71.10	92.30	64.20	66.15	41.50	64.98
	Base	36.15	43.55	14.97	38.66	87.15	43.21	24.62	38.26	40.82
	ICL	53.55	41.53	15.07	45.44	76.95	40.74	40.31	39.28	44.11
	EWC	41.65	52.82	33.35	29.08	74.80	25.93	39.38	40.53	42.19
	LwF	36.10	27.22	8.88	18.57	43.05	20.99	46.46	40.52	30.22
	GEM	40.60	36.89	35.55	37.16	79.70	37.04	67.38	41.15	46.93
	O-LoRA	53.80	59.88	25.47	46.57	84.10	35.80	48.00	40.86	49.31
	SD-LoRA	52.75	55.65	15.81	43.54	90.35	32.10	32.31	38.49	45.13
	Seq-LoRA	43.55	53.23	26.53	43.71	87.00	54.32	**72.31**	**41.85**	52.81
	RCL	52.00	55.44	26.02	53.63	83.40	56.79	63.08	39.47	53.73
	SLoRA-Post	49.45	59.48	23.78	53.87	**92.25**	59.26	58.46	40.87	54.68
	SLoRA-Pre	**53.80**	**60.48**	**50.80**	**58.23**	90.40	**61.73**	64.92	41.19	**60.19**
Qwen2.5-7B	STL†	58.95	70.56	59.13	71.46	93.00	67.90	69.23	41.95	66.52
	MTL†	61.55	68.55	55.87	72.86	91.40	60.49	62.15	41.38	64.28
	Base	56.05	57.26	12.96	45.93	85.80	54.32	37.85	38.79	48.62
	ICL	59.30	55.65	22.63	53.46	85.80	54.32	44.00	39.70	51.86
	EWC	45.90	33.27	29.53	56.32	37.50	46.91	63.38	40.91	44.22
	LwF	44.15	44.96	29.70	53.39	67.05	48.15	59.38	40.71	48.44
	GEM	43.85	36.29	32.56	58.61	64.10	37.04	**67.69**	40.87	47.63
	O-LoRA	50.15	42.14	19.85	31.20	84.40	51.85	51.08	41.07	46.47
	SD-LoRA	61.30	60.48	15.37	51.25	86.45	56.79	43.38	39.11	51.77
	Seq-LoRA	51.35	55.44	24.64	43.34	75.85	56.79	63.69	41.22	51.54
	RCL	**61.80**	61.49	34.05	**64.60**	86.60	60.49	55.08	39.62	57.97
	SLoRA-Post	60.65	**63.31**	33.51	56.87	89.00	61.73	58.46	41.51	58.13
	SLoRA-Pre	54.25	59.68	**45.96**	58.72	**89.65**	**61.73**	61.85	**41.72**	**59.20**

Table 1: Experimental results on the TRACE benchmark across three foundation LLMs. '†' indicates the oracle methods. We compare SLoRA (SLoRA-Pre and SLoRA-Post) with regularization-based and regularization-free baselines. For each task–model pair, **bold** and underlined represent the best and second-best results, respectively.

Instruct, Qwen2.5-14B-Instruct, and Qwen2.5-32B-Instruct. Regularization-based baselines (except O-LoRA and SD-LoRA) use the official TRACE codebase,[1] while all other methods, including SLoRA, are implemented within our unified framework. To ensure a rigorous and fair comparison, the STL and MTL oracle baselines are also implemented using LoRA with the same hyperparameter configurations as our proposed SLoRA. Unless otherwise specified, we employ LoRA with a rank of r=64 and a scaling factor of α=128. We utilize the AdamW optimizer with a learning rate of 2×10^{-4}, a 3% linear warmup, and a cosine annealing schedule. Following TRACE, the eight tasks are trained for $5/3/7/5/3/5/5/7$ epochs, respectively. For the standard CL benchmark, we evaluate on Llama3-8B-Instruct and follow O-LoRA for task sequences (Order1–Order3), task prompts, and evaluation protocol. [2] In this setting, each task is trained for a single epoch, with all other hyperparameters remaining consistent. All experiments are conducted on four NVIDIA A100 GPUs.

[1] https://github.com/BeyonderXX/TRACE

[2] https://github.com/cmnfriend/O-LoRA

5.2 Main Results

Table 1 reports results on the TRACE benchmark with Llama3.1-8B, Qwen2.5-7B, and additional results for Llama3-8B are in Table 10. The best and second-best scores for each task are highlighted in **bold** and underlined, respectively. *Oracle* methods are included for context and not ranked.

Vanilla methods exhibit limited capabilities. Base performs poorly on most tasks, particularly domain-specific ones like MeetingBank, while ICL provides only marginal gains as it does not update parameters. *Regularization-based* methods generally underperform compared to *Regularization-free* approaches. On Llama3.1-8B and Llama3-8B, methods such as EWC and GEM often fall below the simple Seq-LoRA baseline. While SD-LoRA shows modest gains on Qwen2.5-7B by retaining early tasks, it degrades on later tasks, supporting the observation that regularization constraints often alleviate forgetting at the cost of plasticity. Among *Regularization-free* baselines, Seq-LoRA suffers from substantial forgetting during sequential fine-tuning, while RCL improves performance but relies

Base Model	Methods	C-STANCE	FOMC	MTB.	Py150	Sci.QA	NumG. cm	NumG. ds	20Minuten	Avg.
Qwen2.5-14B	Base	18.45	54.44	15.28	53.41	90.45	66.67	40.31	39.69	47.34
	Seq-LoRA	48.75	59.07	**47.40**	69.12	**91.95**	59.26	50.69	**42.62**	58.61
	SLoRA-Post	**59.15**	**64.31**	45.05	67.02	91.40	**66.67**	52.92	41.15	**60.96**
	SLoRA-Pre	56.85	62.70	47.35	**69.91**	91.35	59.26	**54.77**	42.20	60.55
Qwen2.5-32B	Base	50.30	37.10	14.27	52.39	92.10	69.14	52.31	39.35	50.87
	Seq-LoRA	51.15	58.47	34.07	23.99	95.35	76.54	70.15	42.38	56.51
	SLoRA-Post	**59.95**	66.73	52.87	67.08	95.07	75.31	71.69	42.52	66.48
	SLoRA-Pre	59.25	**68.35**	**55.13**	**73.17**	**96.35**	**77.78**	**77.85**	**42.97**	**68.61**

Table 2: Experimental results on the TRACE benchmark with Qwen2.5-14B model and Qwen2.5-32B model. **Bold** and underlined denote the best and second-best results, respectively.

Method	Order-1	Order-2	Order-3	avg.
MTL†	83.02	83.02	83.02	83.02
Base	69.29	69.29	69.29	69.29
ICL	58.40	60.58	63.87	60.95
EWC	68.78	70.72	70.52	70.01
LwF	69.43	64.12	68.68	67.41
O-LoRA	72.59	77.22	**77.24**	75.68
SD-LoRA	72.69	72.61	72.14	72.48
Seq-LoRA	67.04	67.29	53.10	62.48
SLoRA-Post	75.83	76.29	69.19	73.77
SLoRA-Pre	**76.58**	**78.09**	73.92	**76.19**

Table 3: Performance on the standard CL benchmark with Llama3-8B, including average accuracy for Order-1, Order-2, Order-3, and overall average. **Bold** and underlined denote the best and second-best results. '†' indicates the oracle methods.

Method	C-STANCE	FOMC	MTB.	Py150	Avg.
Selection Strategies					
min	54.30	62.30	44.15	60.43	55.30
minor	54.40	62.10	50.12	**64.64**	57.82
max (Ours)	**55.20**	**65.32**	**50.83**	64.23	**58.90**
Similarity Metrics					
L2	54.75	63.91	47.12	63.20	57.25
Cosine	54.95	63.51	48.63	62.74	57.46
PCA	53.10	64.31	**51.02**	**66.00**	58.61
Frobenius (Ours)	**55.20**	**65.32**	50.83	64.23	**58.90**
Reference Modules					
LoRA (last)	53.80	65.73	46.82	62.58	57.23
LoRA (all)	53.95	**65.93**	45.54	63.22	57.16
Base+Prev (merged)	52.70	62.30	**51.71**	**66.43**	58.29
Base+Last (separate)	55.05	65.93	47.50	59.28	56.94
Base+Prev (separate)	54.50	62.30	47.41	63.22	56.86
Base (Ours)	**55.20**	65.32	50.83	64.23	**58.90**

Table 4: Ablation Study of SLoRA on first four TRACE tasks with Llama3-8B and SLoRA-Pre. We evaluate different configurations of LoRA subspace selection strategies, similarity metrics, and reference modules. The best results in each block are highlighted in **bold**.

on external data augmentation.

In contrast, SLoRA attains the strongest averages across all base models, effectively narrowing the gap to Oracle bounds. It maintains a balanced profile between early and later tasks, indicating reduced cross-task interference without restricting learning capacity. We further verify the scalability of our approach on larger architectures in Table 2. SLoRA consistently outperforms the sequential baseline on both Qwen2.5-14B and Qwen2.5-32B. Notably, on Qwen2.5-32B, SLoRA-Pre achieves an average accuracy of 68.61%, significantly surpassing Seq-LoRA (20.41%). This demonstrates that our subspace denoising mechanism remains robust and effective as model scale increases.

5.3 Results on Standard CL Benchmark

We evaluate on the standard CL benchmark with Llama3-8B, as shown in Table 3. Among regularization-based methods, improvements over Base are limited except for O-LoRA and LwF even falls below Base. The datasets in this benchmark are relatively simple and the task types are closely aligned, which makes O-LoRA's orthogonality constraint effective in this setting and leads to the second-best result. SLoRA-Post performs comparably to O-LoRA and consistently surpasses SD-LoRA, while SLoRA-Pre achieves the best overall results with stronger robustness to task-order variation. These findings confirm the effectiveness and versatility of SLoRA. We further extend the evaluation to the 15-task long sequence following O-LoRA on Llama3-8B. As detailed in Appendix B.5, SLoRA remains the top performer, and SLoRA-Post shows greater stability across the long sequence, yielding the best final average.

5.4 Ablation Study

To validate the effectiveness and impact of design choices in SLoRA, We conduct ablation experiments on the first four TRACE tasks using Llama3-8B with SLoRA-Pre. We examine three dimensions: selection strategies, similarity metrics, and

Method	C-STANCE		FOMC		MeetingBank		Py150		Avg.	
	ACC↑	AFR↓	ACC↑	AFR↓	ACC↑	AFR↓	ACC↑	AFR↓	ACC↑	AFR↓
EWC	48.05	3.64	42.34	20.09	36.23	10.93	53.46	6.58	45.02	10.31
GEM	44.35	4.24	40.73	12.54	36.40	12.12	54.06	9.56	43.89	9.61
O-LoRA	46.05	12.11	51.50	20.86	31.11	8.24	54.77	3.36	45.86	11.14
SD-LoRA	52.15	**0.16**	55.04	**0.60**	17.22	**-0.68**	48.53	**-0.63**	43.24	**-0.14**
Seq-LoRA	47.95	7.06	**65.73**	10.85	30.90	13.06	55.54	6.26	50.03	9.31
SLoRA-Pre	52.60	4.19	53.63	10.48	**43.78**	3.17	**61.80**	1.30	**52.95**	4.78
SLoRA-Post	**52.70**	4.63	62.90	3.19	23.86	9.91	57.68	2.64	49.29	5.09

Table 5: Final performance (ACC) and forgetting rate (AFR) on the first four tasks in the TRACE benchmark using Llama3-8B. ↑ means higher is better; ↓ means lower is better. SLoRA variants achieve a favorable balance of stability and accuracy. **Bold** indicates the best value per column, and underline indicates the second-best.

reference modules. Table 4 reports the results and highlights the best within each group in **bold**.

Selection Strategies. We compare three heuristics for selecting singular directions from the LoRA update: `min`, which retains the least aligned components; `minor`, which keeps intermediate ones; and `max` (our default), which preserves the most aligned directions. Among the three, `max` attains the best average performance. These results support the denoising hypothesis: base-aligned directions contain less task noise and transfer more stably.

Similarity Metrics. We evaluate four measures for guiding subspace selection: L2 distance, cosine similarity, PCA-based projection overlap, and Frobenius-based subspace similarity. Frobenius-based similarity achieves the highest overall average. PCA is competitive on some tasks but is less consistent across the sequence. These results indicate that measuring overlap at the subspace level provides a more stable signal for denoising.

Reference Modules. We compare different reference representations used in subspace similarity computation, including previous LoRA adapters, the base model, and their combinations. These settings differ in whether similarity is computed against only the base model, specific prior tasks, or merged representations, as detailed in Appendix B.6. As shown in Table 4, the configuration using the base model alone consistently outperforms all other variants. Alternatives involving dynamic or merged histories fail to provide a stable reference frame. This validates our design choice that a fixed, task-agnostic base model serves as the most effective anchor for subspace denoising.

Overall, these results show that each of the core components plays a crucial role in the robustness and effectiveness of the proposed SLoRA.

Method	C-STANCE	FOMC	MeetingBank	Py150	Avg.
Seq-LoRA	55.75	70.36	55.74	60.55	60.60
GEM	50.20 (-5.55)	66.94 (-3.42)	53.26 (-2.48)	58.78 (-1.77)	57.30 (-3.31)
EWC	50.20 (-5.55)	68.95 (-1.41)	50.78 (-4.96)	56.35 (-4.20)	56.57 (-4.03)
O-LoRA	55.75 (+0.00)	59.48 (-10.88)	40.95 (-14.79)	57.23 (-3.32)	53.35 (-7.25)
SD-LoRA	52.70 (-3.05)	56.65 (-13.71)	15.69 (-40.05)	47.89 (-12.66)	43.23 (-17.37)
SLoRA-Post	58.50 (+2.75)	64.31 (-6.05)	46.12 (-9.62)	60.25 (-0.30)	57.30 (-3.31)
SLoRA-Pre	58.25 (+2.50)	71.37 (+1.01)	50.77 (-4.97)	64.23 (+3.68)	**61.16 (+0.56)**

Table 6: Performance on new tasks immediately after training on Llama3-8B. Parentheses report the score change vs. Seq-LoRA: positive (green) indicates improvement and negative (red) indicates degradation. **Bold** marks the best average.

6 Analysis

6.1 Earlier Tasks Forgetting

We analyze catastrophic forgetting on Llama3-8B using the first four TRACE tasks. Table 5 reports final accuracy (ACC) and average forgetting rate (AFR; formula in Appendix B.7). Seq-LoRA attains strong ACC on FOMC but shows high AFR on average. Classical regularization-based methods offer limited retention at LLM scale. SD-LoRA achieves near-zero or negative AFR yet loses accuracy on MeetingBank and Py150, which lowers its average. In contrast, both SLoRA variants offer a better balance: SLoRA-Pre reaches the best average ACC with a low AFR, and SLoRA-Post remains competitive with the second-lowest average AFR. These results indicate that subspace denoising reduces cross-task interference while preserving performance on earlier tasks.

6.2 New Tasks Learning Capacity

We examine how well models acquire new tasks when they are introduced. Table 6 reports performance immediately after training on each of the first four TRACE tasks on Llama3-8B. Numbers in parentheses denote the change relative to Seq-LoRA. Regularization-based methods gener-

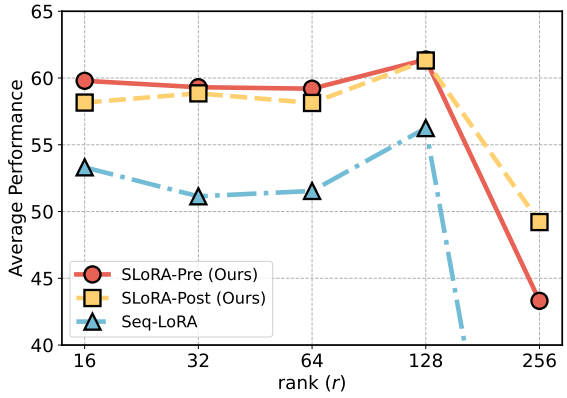


Figure 3: Impact of LoRA rank r on the average performance across the TRACE benchmark using Qwen2.5-7B, demonstrating that both SLoRA-Pre and SLoRA-Post consistently achieve superior accuracy compared to the Seq-LoRA baseline across all tested ranks.

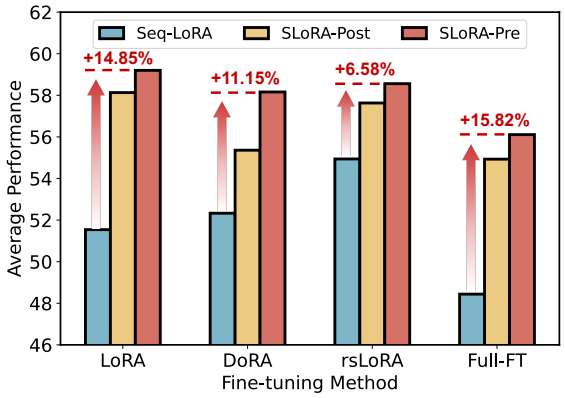


Figure 4: Universality of SLoRA across various fine-tuning methods (LoRA, DoRA, rsLoRA, and Full-FT) on the TRACE benchmark using Qwen2.5-7B. Percentages indicate the performance improvement of SLoRA-Pre over the Seq-LoRA baseline.

ally underperform Seq-LoRA. SD-LoRA shows the weakest adaptation among regularization-based approaches, with notable drops on MeetingBank and Py150. Seq-LoRA adapts well to new tasks but, as shown in the forgetting analysis, suffers pronounced interference later in the sequence. In contrast, both SLoRA variants maintain strong new-task performance, with SLoRA-Pre achieving the best overall average and SLoRA-Post remaining competitive, particularly on FOMC. The pattern indicates that subspace denoising preserves plasticity while controlling interference.

6.3 Sensitivity to LoRA Rank

We analyze the impact of LoRA rank r on the average performance of the TRACE benchmark using Qwen2.5-7B. As illustrated in Figure 3, both SLoRA-Pre and SLoRA-Post consistently outperform the Seq-LoRA baseline across the entire spectrum of ranks, ranging from $r = 16$ to $r = 256$. While we observe a performance drop at $r = 256$ across all methods, likely due to increased noise accumulation from the larger parameter space, SLoRA maintains a significant advantage over the baseline. This consistent stability confirms that our subspace denoising approach is inherently robust to variations in adapter size. Such results underscore the practicality of SLoRA, as it achieves superior performance without the necessity for fine-grained rank tuning.

6.4 Generality across Fine-tuning Methods

To assess the universality of our framework, we extend SLoRA to DoRA (Liu et al., 2024), rsLoRA (Kalajdzievski, 2023), and Full Fine-Tuning (Full-FT) on the TRACE benchmark using Qwen2.5-7B. As shown in Figure 4, SLoRA consistently outperforms the sequential baseline across all methods. Notably, SLoRA-Pre yields substantial improvements on DoRA and rsLoRA, confirming that noise accumulation affects various low-rank structures. While Full-FT suffers severe forgetting in Seq-LoRA, SLoRA-Pre effectively restores performance with a 15.82% improvement. These results demonstrate that subspace denoising is a versatile mechanism effective for both parameter-efficient and full-parameter adaptation.

7 Conclusion

In this paper, we are the first to identify that noise accumulation within low-rank updates is a key factor contributing to catastrophic forgetting in the continual learning of LLMs. Based on this insight, we propose SLoRA, a novel regularization-free framework designed to mitigate cross-task interference by filtering noisy components from LoRA updates via subspace similarity analysis with the frozen base model. This method is lightweight, requiring no rehearsal data or historical gradient storage while keeping the standard training process entirely intact. It offers two versatile variants, SLoRA-Pre and SLoRA-Post, which are specifically tailored for online and offline settings, respectively. Extensive experiments and analyses demonstrate that SLoRA effectively reduces forgetting without compromising the model's capacity to learn new tasks, thereby achieving a balance between stability and plasticity.

Limitations

While SLoRA demonstrates robust performance and scalability across various model sizes and fine-tuning paradigms (including Full-FT and DoRA), we acknowledge certain limitations. First, our subspace-based denoising mechanism relies on computing similarity with the base model's representation space. This requires white-box access to the pre-trained weights, limiting applicability in strictly black-box API scenarios where model parameters are inaccessible. Second, although the proposed randomized SVD approximation is highly efficient, the denoising step introduces a marginal computational cost compared to naive sequential fine-tuning. Finally, the offline variant (SLoRA-Post) necessitates buffering raw task updates prior to the unified denoising phase. While these low-rank updates are storage-efficient, this approach incurs a temporary memory overhead proportional to the sequence length during the training phase compared to purely online methods that discard updates immediately.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62576209) and STCSM (No. 2025SHZDZX025G05).

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Dan Gillick, Jacob Eisenstein, and William Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, and 1 others. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pages 3762–3773. PMLR.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6429–6440.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, KIM Gyeonghun, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2024. Taia: Large language models are out-of-distribution data learners. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2025. Fine-tuning with reserved majority for noise reduction. In *The Thirteenth International Conference on Learning Representations*.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. 2019. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International conference on machine learning*, pages 3925–3934. PMLR.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Yan-Shuo Liang and Wu-Jun Li. 2024. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

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. In *Forty-first International Conference on Machine Learning*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Fuli Qiao and Mehrdad Mahdavi. 2024. Learn more, but bother less: parameter efficient continual learning. *Advances in Neural Information Processing Systems*, 37:97476–97498.

Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems*, 32.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, Maosong Sun, and Tao Yang. 2023. Conpet: Continual parameter-efficient tuning for large language models. *arXiv preprint arXiv:2309.14763*.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2024. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2023a. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, and 1 others. 2023b. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149.

Genta Winata, Lingjue Xie, Karthik Radhakrishnan, Shijie Wu, Xisen Jin, Pengxiang Cheng, Mayank Kulkarni, and Daniel Preoţiuc-Pietro. 2023. Overcoming catastrophic forgetting in massively multilingual continual learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 768–777.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024a. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. 2025. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In *The Thirteenth International Conference on Learning Representations*.

Yichen Wu, Hong Wang, Peilin Zhao, Yefeng Zheng, Ying Wei, and Long-Kai Huang. 2024b. Mitigating catastrophic forgetting in online continual learning

by modeling previous task interrelations via pareto optimization. In *Forty-first International Conference on Machine Learning*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. In *Transactions of the Association for Computational Linguistics*, volume 4, pages 401–415.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2018. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *arXiv preprint arXiv:2504.07448*.

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 *The Eleventh International Conference on Learning Representations*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. 2024. Seeking neural nuggets: Knowledge transfer in large language models from a parametric perspective. In *The Twelfth International Conference on Learning Representations*.

A Theoretical Proofs

This appendix provides the mathematical derivation for the reference subspace selection analysis discussed in Section 4.3. We rigorously define the alignment metric and analyze its asymptotic behavior under two different reference strategies: using the previous model (θ_{t-1}) versus using the fixed base model (θ_0).

A.1 Definitions and Assumptions

We start by formalizing the relationship between the reference subspace and the learned update subspace.

Assumption 1 (Subspace Evolution Hypothesis). *We assume that the principal basis $\mathbf{U}_t$ of the denoised update for task t is derived from the reference basis $\mathbf{U}_{ref}$ perturbed by a task-specific innovation. Formally, $\mathbf{U}_t$ can be represented as a linear combination of the reference basis and an orthogonal innovation basis $\mathbf{N}_t$:*

$$\mathbf{U}_t = \mathbf{U}_{ref}\mathbf{R}_t + \mathbf{N}_t\mathbf{S}_t, \tag{6}$$

where $\mathbf{R}_t$ and $\mathbf{S}_t$ are coefficient matrices. This decomposition adheres to the following properties:

1. ***Subspace Alignment:*** *The denoising process ensures that $\mathbf{U}_t$ maintains significant alignment with the reference. This is quantified by the Frobenius norm of their product:*

$$\left\|(\mathbf{U}_{ref})^\top \mathbf{U}_t\right\|_F^2 = \gamma_t, \tag{7}$$

where $0 < \gamma_t < rank(\mathbf{U}_t)$ represents the magnitude of retained information.

2. ***Orthogonality of Innovation:*** *The innovation basis $\mathbf{N}_t$ captures new task knowledge that is orthogonal to the reference subspace, i.e., $(\mathbf{U}_{ref})^\top \mathbf{N}_t = \mathbf{0}$.*

3. ***High-Dimensional Sparsity:*** *Due to the high dimensionality of the parameter space, task-specific innovations from different tasks are mutually nearly orthogonal. For $t \neq i$:*

$$\left\|(\mathbf{N}_t)^\top \mathbf{N}_i\right\|_F^2 \approx 0. \tag{8}$$

To evaluate the consistency of the learned subspaces across the lifespan of the model, we define the following metric:

Definition 2 (Historical Alignment Score). *The Historical Alignment Score $\mathcal{A}(t)$ measures the average subspace consistency between the current task update and historical updates. Consistent with our denoising metric, it is defined as the Frobenius norm of the product of their principal bases:*

$$\mathcal{A}(t) = \frac{1}{t-1}\sum_{i=1}^{t-1}\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F^2, \tag{9}$$

where $\mathbf{U}_t$ and $\mathbf{U}_i$ represent the matrix of top-c^* left singular vectors (principal basis) of the denoised updates $\Delta\widetilde{\theta}_t$ and $\Delta\widetilde{\theta}_i$, respectively.

A high $\mathcal{A}(t)$ indicates that the principal directions of the current update remain aligned with the subspaces established by previous tasks. Based on this, we present the following theorem to validate our choice of θ_0.

A.2 Proof of Lemma 1

To prove the main theorem, we introduce a lemma describing how subspace alignment decays when the reference frame shifts sequentially.

Lemma 1 (Geometric Decay of Subspace Consistency). *Let the reference basis for task t be the principal basis of the previous task, i.e., $\mathbf{U}_{ref} = \mathbf{U}_{t-1}$. The subspace alignment between the current basis $\mathbf{U}_t$ and a historical basis $\mathbf{U}_i$ $(i < t)$ satisfies the recurrence relation:*

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F \leq \sqrt{\gamma_t} \cdot \left\|(\mathbf{U}_{t-1})^\top \mathbf{U}_i\right\|_F. \quad (10)$$

Consequently, the alignment score decays geometrically with the distance $t - i$:

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F^2 \leq C_{start} \cdot \prod_{k=i+1}^{t} \gamma_k, \quad (11)$$

where $\gamma_k < 1$ (normalized) acts as the retention factor.

Proof. Substituting Assumption 1 with $\mathbf{U}_{\text{ref}} = \mathbf{U}_{t-1}$, we express $\mathbf{U}_t$ as:

$$\mathbf{U}_t = \mathbf{U}_{t-1}\mathbf{R}_t + \mathbf{N}_t\mathbf{S}_t. \quad (12)$$

We examine the interaction between $\mathbf{U}_t$ and the distant historical basis $\mathbf{U}_i$. Computing the matrix product:

$$(\mathbf{U}_t)^\top \mathbf{U}_i = (\mathbf{R}_t)^\top (\mathbf{U}_{t-1})^\top \mathbf{U}_i + (\mathbf{S}_t)^\top (\mathbf{N}_t)^\top \mathbf{U}_i. \quad (13)$$

Based on the High-Dimensional Sparsity property (Assumption 1), the current innovation $\mathbf{N}_t$ is orthogonal to the historical basis $\mathbf{U}_i$ (which is composed of previous innovations and references), so $\|(\mathbf{N}_t)^\top \mathbf{U}_i\|_F \approx 0$. The expression simplifies to:

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F \approx \left\|(\mathbf{R}_t)^\top (\mathbf{U}_{t-1})^\top \mathbf{U}_i\right\|_F. \quad (14)$$

Using the sub-multiplicative property of matrix norms ($\|AB\|_F \leq \|A\|_2 \|B\|_F$) and noting that $\|\mathbf{R}_t\|_2 \approx \|\mathbf{R}_t\|_F$ for low-rank matrices in this context:

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F \leq \|\mathbf{R}_t\|_F \cdot \left\|(\mathbf{U}_{t-1})^\top \mathbf{U}_i\right\|_F. \quad (15)$$

From Assumption 1, $\left\|(\mathbf{U}_{t-1})^\top \mathbf{U}_t\right\|_F^2 = \gamma_t$. Since $\mathbf{U}_{t-1}$ has orthonormal columns, $\left\|(\mathbf{U}_{t-1})^\top \mathbf{U}_t\right\|_F = \|\mathbf{R}_t\|_F = \sqrt{\gamma_t}$. Thus:

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F \leq \sqrt{\gamma_t} \cdot \left\|(\mathbf{U}_{t-1})^\top \mathbf{U}_i\right\|_F. \quad (16)$$

Recursively applying this relation yields the geometric decay. □

A.3 Proof of Theorem 1

Theorem 1 (Alignment Stability Analysis). *Let $\mathcal{A}^{(prev)}(t)$ denote the alignment score when using the previous model θ_{t-1} as the reference, and $\mathcal{A}^{(base)}(t)$ denote the score when using the base model θ_0. As the number of tasks $t \to \infty$:*

$$\lim_{t\to\infty} \mathcal{A}^{(prev)}(t) = 0, \quad \lim_{t\to\infty} \mathcal{A}^{(base)}(t) = C, \quad (17)$$

where C is a positive constant.

Proof. The proof analyzes the asymptotic behavior of the subspace consistency metric $\mathcal{A}(t) = \frac{1}{t-1}\sum_{i=1}^{t-1} \|(\mathbf{U}_t)^\top \mathbf{U}_i\|_F^2$.

Case 1: $\theta_{\text{ref}} = \theta_{t-1}$

Here, the reference basis for task t is $\mathbf{U}_{t-1}$. According to Lemma 1, the alignment between $\mathbf{U}_t$ and any historical basis $\mathbf{U}_i$ decays as the distance $t - i$ increases. Let $\gamma_{\max} = \sup_k \gamma_k < 1$ (normalized retention factor). We bound the term:

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F^2 \leq C_0 \cdot \gamma_{\max}^{t-i}. \quad (18)$$

Substitute this into the alignment score:

$$\mathcal{A}^{(\text{prev})}(t) \leq \frac{C_0}{t-1}\sum_{i=1}^{t-1} \gamma_{\max}^{t-i} = \frac{C_0}{t-1}\sum_{j=1}^{t-1} \gamma_{\max}^{j}. \quad (19)$$

As $t \to \infty$, the sum of the geometric series converges to a constant $\frac{\gamma_{\max}}{1-\gamma_{\max}}$, but the factor $\frac{1}{t-1}$ drives the total value to 0:

$$\lim_{t\to\infty} \mathcal{A}^{(\text{prev})}(t) = 0. \quad (20)$$

Case 2: $\theta_{\text{ref}} = \theta_0$

In this case, the reference basis for all tasks is $\mathbf{U}_0$. According to Assumption 1, for any task k, $\mathbf{U}_k$ is derived from $\mathbf{U}_0$:

$$\mathbf{U}_k = \mathbf{U}_0\mathbf{R}_k + \mathbf{N}_k\mathbf{S}_k. \quad (21)$$

We evaluate the alignment term $\|(\mathbf{U}_t)^\top \mathbf{U}_i\|_F^2$:

$$\begin{aligned}(\mathbf{U}_t)^\top \mathbf{U}_i &= (\mathbf{R}_t)^\top (\mathbf{U}_0)^\top \mathbf{U}_0 \mathbf{R}_i + \dots \\ &\approx (\mathbf{R}_t)^\top \mathbf{R}_i.\end{aligned} \quad (22)$$

Note that the cross terms involving $\mathbf{N}_t$ and $\mathbf{N}_i$ vanish due to orthogonality with $\mathbf{U}_0$ and mutual orthogonality (Assumption 1). Thus, the alignment depends on the retained signal matrices $\mathbf{R}_t$ and $\mathbf{R}_i$.

$$\left\|(\mathbf{U}_t)^\top \mathbf{U}_i\right\|_F^2 \approx \left\|(\mathbf{R}_t)^\top \mathbf{R}_i\right\|_F^2. \quad (23)$$

Dataset	Source	Avg Len	Metric	Language	#Data
Domain-specific					
ScienceQA	Science	210	Accuracy	English	5000
FOMC	Finance	51	Accuracy	English	5000
MeetingBank	Meeting	2853	ROUGE-L	English	5000
Multi-lingual					
C-STANCE	Social media	127	Accuracy	Chinese	5000
20Minuten	News	382	SARI	German	5000
Code completion					
Py150	Github	422	Edim similarity	Python	5000
Mathematical reasoning					
NumGLUE-cm	Math	32	Accuracy	English	5000
NumGLUE-ds	Math	21	Accuracy	English	5000

Table 7: Overview of dataset statistics in the TRACE benchmark.

Assuming a consistent learning stability where the magnitude of information retained from the base subspace is stable across tasks (i.e., $\mathbb{E}[\|(\mathbf{R}_t)^\top \mathbf{R}_i\|_F^2] = C > 0$), the average score becomes:

$$\lim_{t\to\infty} \mathcal{A}^{(\text{base})}(t) = \lim_{t\to\infty} \frac{1}{t-1} \sum_{i=1}^{t-1} C = C. \quad (24)$$

This confirms that anchoring to θ_0 maintains a globally consistent principal subspace. □

B Experiment Details

B.1 Dataset Statistcs

TRACE To evaluate the robustness and generalizability of our method across diverse tasks and modalities, we conduct experiments on the TRACE benchmark, which consists of eight datasets spanning multiple domains, languages, and task types. As summarized in Table 7, the benchmark includes domain-specific tasks such as question answering (ScienceQA), financial document classification (FOMC), and meeting summarization (MeetingBank); multilingual tasks in Chinese and German (C-STANCE and 20Minuten); code completion in Python (Py150); and mathematical reasoning tasks (NumGLUE-cm and NumGLUE-ds). Each dataset contains 5,000 samples and is evaluated using appropriate metrics such as accuracy, ROUGE-L (Lin, 2004), SARI (Xu et al., 2016), or embedding similarity. [3] This diverse composition enables a comprehensive assessment of continual learning methods under heterogeneous task settings, linguistic variation, and input distributions.

standard CL benchmark For comparison with prior continual learning work, we also evaluate on the standard CL benchmark for language models. This benchmark contains five text classification datasets drawn from different domains (Zhang et al., 2015): AG News, Amazon reviews, Yelp reviews, DBpedia and Yahoo Answers. Table 8 summarizes the task types and domains. Following O-LoRA, we report results under three task sequences (Table 9) to evaluate robustness against task-order variation. Unlike TRACE, this benchmark focuses on homogeneous text classification tasks, making it a complementary setting for assessing continual learning methods.

All datasets do not contain personally identifiable information or offensive content

[3] https://github.com/seatgeek/fuzzywuzzy

Dataset	Task	Domain
Yelp	sentiment analysis	Yelp reviews
Amazon	sentiment analysis	Amazon reviews
DBpedia	topic classification	Wikipedia
Yahoo	topic classification	Yahoo Q&A
AG News	topic classification	news

Table 8: The details of datasets used in the standard CL benchmark.

Order	Task Sequence
1	dbpedia → amazon → yahoo → ag
2	dbpedia → amazon → ag → yahoo
3	yahoo → amazon → ag → dbpedia

Table 9: Three different task sequences of the standard CL benchmark used for continual learning experiments.

B.2 Additional Implementation Details

The open-source code, models, and datasets utilized in our experiments adhere to their respective licenses and are used in accordance with their intended purposes.

B.3 Pre-Experiments

We design a controlled two-task experiment to validate the effect of noisy components in LoRA updates. Following the standard CL benchmark, we select Amazon Reviews as *Task1* (sentiment analysis) and Yahoo Answers as *Task2* (topic classification), which differ in task type. We train Llama3-8B sequentially on *Task1* and *Task2* with standard Seq-LoRA.

After training, we proportionally drop LoRA parameters under three conditions: DropFirst (operate on the *Task1* adapter only), DropSecond (operate

Base Model	Methods	C-STANCE	FOMC	MTB.	Py150	Sci.QA	NumG. cm	NumG. ds	20Minuten	Avg.
Llama3-8B	STL†	55.50	70.77	56.76	68.79	94.10	60.49	73.85	41.82	65.26
	MTL†	55.95	67.54	55.23	70.20	92.60	66.67	66.46	41.65	64.54
	Base	44.05	53.83	14.38	42.04	80.10	32.10	19.69	37.26	40.43
	ICL	46.85	58.27	13.30	44.81	84.70	40.74	25.85	38.79	44.16
	EWC	48.05	42.34	36.23	53.46	60.95	27.16	64.00	40.78	46.62
	LwF	40.90	38.31	6.56	26.81	40.15	12.35	53.23	39.50	32.23
	GEM	44.35	40.73	36.40	54.06	63.50	28.40	65.54	40.69	46.71
	O-LoRA	51.50	50.20	31.11	54.77	81.50	60.49	58.15	40.50	53.53
	SD-LoRA	52.15	55.04	17.22	48.53	87.00	33.34	30.46	39.32	45.38
	Seq-LoRA	47.95	**65.73**	30.90	55.54	82.00	49.38	**71.38**	41.11	55.50
	RCL	**58.20**	50.40	14.14	63.40	84.40	61.73	53.54	39.79	53.20
	SLoRA-Post	52.70	62.90	23.86	57.68	90.00	**66.67**	63.69	40.75	57.28
	SLoRA-Pre	52.60	53.63	**43.78**	**61.80**	**90.65**	59.26	67.38	**41.47**	**58.82**

Table 10: Experimental results on the TRACE benchmark in Llama3-8B. '†' indicates the oracle methods excluded from ranking. We compare SLoRA (SLoRA-Pre and SLoRA-Post) with regularization-based and regularization-free baselines. For each task–model pair, **bold** and underlined denote the best and second-best continual methods.

Methods	MNLI	CB	WIC	COPA	QQP	BoolQA	RTE	IMDB	Yelp	Amazon	SST-2	DBpedia	AG.	MultiRC	Yahoo	Avg.
Seq-LoRA	50.49	60.71	50.78	87.00	76.92	72.42	59.93	**95.75**	46.08	42.22	**94.15**	87.28	83.68	**82.20**	**74.01**	70.91
O-LoRA	43.65	58.93	**53.76**	91.00	68.07	67.98	54.87	95.29	63.87	57.09	92.09	**93.76**	83.92	77.91	70.34	71.50
SD-LoRA	58.40	75.00	52.35	90.00	**78.70**	78.10	57.76	95.24	63.08	57.89	92.09	87.66	84.51	75.27	69.50	74.37
SLoRA-Pre	42.80	60.71	50.16	85.00	70.57	77.25	53.07	95.59	**66.28**	**61.95**	93.69	92.34	**87.37**	73.78	71.72	72.15
SLoRA-Post	**64.47**	**75.00**	52.35	**92.00**	77.82	**79.82**	**64.26**	95.01	63.26	56.37	91.40	82.88	85.38	79.43	71.36	**75.35**

Table 11: Performance comparison of Seq-LoRA, O-LoRA, SD-LoRA, SLoRA-Pre and SLoRA-Post based on the Llama3-8B model on long-sequence benchmark. The task order follows Order-4 in the O-LoRA paper.

on the *Task2* adapter only), and DropAll (operate on both). The drop ratio ranges from 10% to 90%. Dropping is carried out by removing the less important components in the LoRA update subspace in proportion to the specified drop ratio.

B.4 Main Results on Llama3-8B

Table 10 reports results on the TRACE benchmark using Llama3-8B. Regularization-based approaches remain weak at this scale, often falling below Seq-LoRA. Seq-LoRA itself achieves strong results on some tasks such as FOMC and NumGLUE-ds but suffers from pronounced forgetting, while RCL excels on C-STANCE yet lacks consistency. In contrast, SLoRA-Pre attains the best average and SLoRA-Post is second-best among continual learners. SLoRA-Pre leads on MeetingBank, Py150, and ScienceQA, while SLoRA-Post performs best on NumGLUE-cm, confirming that subspace denoising improves retention without sacrificing adaptation.

B.5 Experiments on the Long-Sequence

To further evaluate continual learning under more challenging conditions, we follow the long-sequence setting introduced in O-LoRA and adopt the same task order (Order-4): `mnli → cb → wic → copa → qqp → boolqa → rte → imdb → yelp → amazon → sst-2 → dbpedia → ag → multirc → yahoo`. We conduct experiments on Llama3-8B across all 15 tasks.

Table 11 compares Seq-LoRA, O-LoRA, SD-LoRA, and our two variants of SLoRA. Seq-LoRA attains strong results on individual datasets such as IMDB and SST-2 but suffers from severe degradation on others, leading to a lower average. O-LoRA improves stability on tasks like WIC and DBpedia yet remains behind in overall performance. SD-LoRA achieves suboptimal average performance due to its better performance in early tasks. SLoRA-Pre shows notable gains on sentiment and topic classification datasets such as Yelp, Amazon, and AG News, while SLoRA-Post achieves the highest average by maintaining strong and balanced results across the entire sequence. These results confirm that SLoRA remains effective under long sequences, and that the post-hoc variant offers superior robustness in this setting.

B.6 Reference Modules for Subspace Similarity

In our ablation study on reference modules (Table 4), we evaluated six configurations to calculate subspace similarity during the denoising process.

These configurations determine which prior representations serve as the reference when measuring the alignment between the current LoRA update and historical model behavior. Specifically, we compare the following designs:

- **LoRA (last)**: The similarity is calculated with the LoRA adapter from the immediately preceding task T_{t-1}.
- **LoRA (all)**: The similarity is calculated with the union of all previous LoRA adapters $\{\Delta\theta_1, \ldots, \Delta\theta_{t-1}\}$.
- **Base+Prev (merged)**: All previous LoRA adapters are merged into the base model: $\theta_0 + \sum_{i=1}^{t-1} \Delta\theta_i$. The current update is then compared to this merged model, as an approximation of the full historical subspace.
- **Base+Last (separate)**: The similarity is computed separately with both θ_0 and $\Delta\theta_{t-1}$, and the scores are averaged.
- **Base+Prev (separate)**: The similarity is computed separately with θ_0 and each prior $\Delta\theta_i$ for $i < t$, and the final similarity score is averaged between them.
- **Base (Ours)**: Only the base model θ_0 is used as the reference, consistent with our default design in SLoRA.

All variants use the same underlying similarity metric (Grassmannian similarity via Frobenius norm). The difference lies solely in the reference subspace, which impacts the stability and generality of the denoising process.

To further validate the theoretical findings from Appendix B, particularly Theorem 1 which supports anchoring to the base model θ_0 to maintain consistent alignment, we compute the Historical Alignment Score $\mathcal{A}(t)$ (defined in Appendix A, Definition 2) for the variants above. The results in Table 12 show that explicitly anchoring to the fixed base model (**Base (Ours)**) yields the highest $\mathcal{A}(t)$ across tasks $t = 2, 3, 4$, demonstrating superior historical subspace consistency compared to methods that rely on accumulated or previous task updates. This numerical evidence supports the core design choice of selecting θ_0 as the fixed reference anchor for denoising.

Method	$\mathcal{A}(2)$	$\mathcal{A}(3)$	$\mathcal{A}(4)$
Seq-LoRA	1.7792	1.7750	1.7101
LoRA (last)	1.9592 (+10.12%)	1.8837 (+6.12%)	1.8276 (+6.87%)
LoRA (all)	1.5770 (-11.36%)	1.5581 (-12.22%)	1.5182 (-11.22%)
Base+Prev (merged)	1.9874 (+11.70%)	1.8807 (+5.95%)	1.8581 (+8.65%)
Base+Last (Separate)	1.5224 (-14.43%)	1.5304 (-13.78%)	1.5164 (-11.33%)
Base+Prev (Separate)	1.5309 (-13.96%)	1.5500 (-12.68%)	1.4756 (-13.71%)
Base (Ours)	**2.0026** (+12.56%)	**1.8966** (+6.85%)	**1.8693** (+9.31%)

Table 12: Historical Alignment Score $\mathcal{A}(t)$ (Higher is Better) for different reference selection strategies on the first four TRACE tasks. Percentage values in parentheses show the improvement relative to the Seq-LoRA.

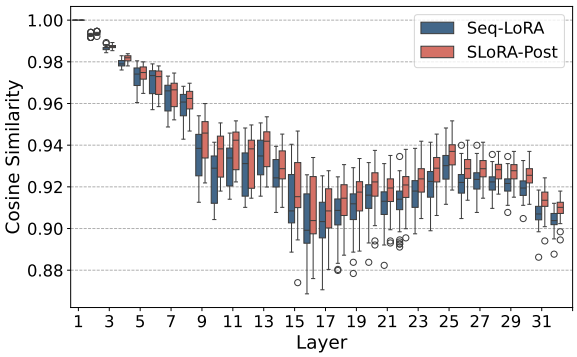


Figure 5: Cosine similarity between hidden states on the first task (C-STANCE), measured both immediately after task-specific training and after completing the full task sequence.

B.7 Average Forgetting Rate Analysis on Early Tasks

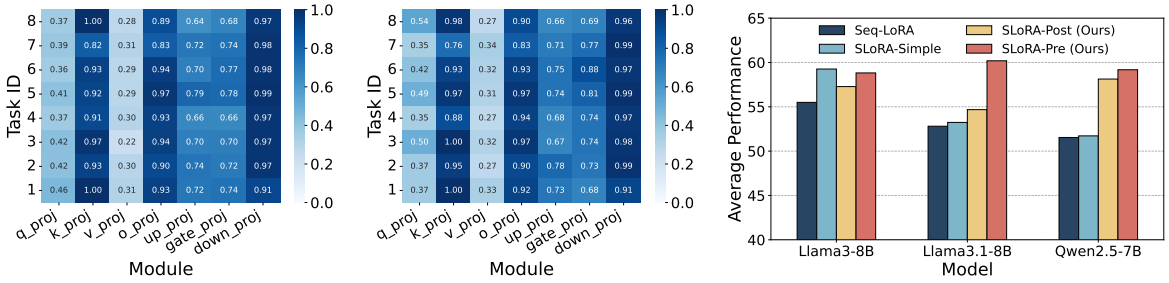
To better quantify catastrophic forgetting under continual learning, we compute the **Average Forgetting Rate (AFR)** for all methods on the first four tasks of the TRACE benchmark—C-STANCE, FOMC, MeetingBank, and Py150—using the LLaMA3-8B model. For a given task T_k, let $R_k(i)$ denote the model's performance on T_k after training up to task T_i. AFR is defined as:

$$\text{AFR}(T_k) = \frac{1}{K-k} \sum_{i=k+1}^{K} \left(R_k(k) - R_k(i)\right), \tag{25}$$

where K is the total number of tasks. AFR measures the average degradation in performance on T_k after subsequent tasks are learned, lower values indicate better retention.

B.8 Representation Drift in Hidden States

As shown in Figure 5, the cosine similarity tends to decrease with layer depth for both methods, suggesting that deeper layers experience greater representational changes during continual learning. Notably, SLoRA-Post consistently maintains higher



(a) (b) (c)

Figure 6: Module retention and effects of denoising strategies. (a) and (b) show module-wise retention ratios after denoising on Llama3-8B for SLoRA-Pre and SLoRA-Post. (c) Average performance across three base models comparing SLoRA-Pre and SLoRA-Post with SLoRA-Simple, which removes `q_proj` and `v_proj` at inference.

similarity than Seq-LoRA, demonstrating that our denoising strategy is effective in preserving task-specific representations while mitigating representation drift.

B.9 Module Retention after Denoising

We analyze module-wise parameter retention after denoising on Llama3-8B for SLoRA-Pre and SLoRA-Post. Figures 6(a) and 6(b) report average retention ratios across tasks. A consistent pattern emerges: `q_proj` and `v_proj` retain the fewest parameters, `up_proj` and `gate_proj` show moderate retention, while `k_proj`, `o_proj`, and `down_proj` are largely preserved. Motivated by the low retention in `q_proj` and `v_proj`, we introduce a simple variant, SLoRA-Simple, which removes these two LoRA modules at inference. Figure 6(c) shows that SLoRA-Simple brings marginal gains on Llama3-8B but reduces stability on Llama3.1-8B and Qwen2.5-7B. In contrast, SLoRA-Pre achieves the highest overall average, indicating that similarity-driven filtering is more effective and robust than fixed module removal.

To understand how continual learning affects internal model representations, we measure the cosine similarity between hidden states on the first task (C-STANCE) at two points: immediately after training on the task itself and after completing the full sequence of continual learning tasks. This similarity reflects how much the model's internal representations shift over time, with lower values indicating greater drift. We conduct this analysis on the Llama3-8B model using 100 randomly selected examples from the C-STANCE test set. For each example, we compute the cosine similarity between hidden states from the two checkpoints across all 32 transformer layers.

Algorithm 1 SLoRA-Pre: Online Subspace-Based Denoising

Require: Full base model weights θ_0, LoRA target modules $\mathcal{M}$, task sequence $\{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$, candidate ranks $\mathcal{C}$

for each $m \in \mathcal{M}$ **do**
  Compute SVD: $\theta_0^m = \mathbf{U}_0^m \mathbf{\Sigma}_0^m (\mathbf{V}_0^m)^\top$
end for
for $k = 1$ to K **do**
  for each $m \in \mathcal{M}$ **do**
    Train LoRA update $\Delta\theta_k^m$ on task $\mathcal{T}_k$ over θ_{k-1}^m
    Compute SVD: $\Delta\theta_k^m = \mathbf{U}^m \mathbf{\Sigma}^m (\mathbf{V}^m)^\top$
    for each $c \in \mathcal{C}$ **do**
      Compute similarity: $\phi_c^m = \left\| (\mathbf{U}_{0,c}^m)^\top \mathbf{U}_c^m \right\|_F^2$
    end for
    $c^* \leftarrow \arg\max_{c \in \mathcal{C}} \phi_c^m$
    Reconstruct $\widetilde{\Delta\theta}_k^m = \mathbf{U}_{c^*}^m \, \mathbf{\Sigma}_{c^*}^m \, (\mathbf{V}_{c^*}^m)^\top$
    $\theta_k^m \leftarrow \theta_{k-1}^m + \widetilde{\Delta\theta}_k^m$
  end for
end for
return Final weights $\{\theta_K^m\}$ where

$$\theta_K^m = \begin{cases} \theta_0^m + \sum_{k=1}^K \widetilde{\Delta\theta}_k^m, & \text{if } m \in \mathcal{M} \\ \theta_0^m, & \text{otherwise} \end{cases}$$

C Implementation and Algorithmic Details

C.1 Algorithm: Subspace-Based Denoising in SLoRA

We present the full pseudocode for both variants of SLoRA. Algorithm 1 describes SLoRA-Pre,

Algorithm 2 SLoRA-Post: Offline Subspace-Based Denoising

Require: Full base model weights θ_0, LoRA target modules $\mathcal{M}$, task sequence $\{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$, candidate ranks $\mathcal{C}$

for each $m \in \mathcal{M}$ **do**
  Compute SVD: $\theta_0^m = \mathbf{U}_0^m \mathbf{\Sigma}_0^m (\mathbf{V}_0^m)^\top$
end for
for $k = 1$ to K **do**
  for each $m \in \mathcal{M}$ **do**
    Train LoRA update $\Delta\theta_k^m$ on task $\mathcal{T}_k$ over θ_0^m
    Store $\Delta\theta_k^m$
  end for
end for
for $k = 1$ to K **do**
  for each $m \in \mathcal{M}$ **do**
    Compute SVD: $\Delta\theta_k^m = \mathbf{U}^m \mathbf{\Sigma}^m (\mathbf{V}^m)^\top$
    for each $c \in \mathcal{C}$ **do**
      Compute similarity: $\phi_c^m = \left\| (\mathbf{U}_{0,c}^m)^\top \mathbf{U}_c^m \right\|_F^2$
    end for
    $c^* \leftarrow \arg\max_{c \in \mathcal{C}} \phi_c^m$
    Reconstruct $\widetilde{\Delta\theta}_k^m = \mathbf{U}_{c^*}^m \, \mathbf{\Sigma}_{c^*}^m \, (\mathbf{V}_{c^*}^m)^\top$
  end for
end for
for each $m \in \mathcal{M}$ **do**
  $\theta_K^m \leftarrow \theta_0^m + \sum_{k=1}^K \widetilde{\Delta\theta}_k^m$
end for
return Final weights $\{\theta_K^m\}$ where

$$\theta_K^m = \begin{cases} \theta_0^m + \sum_{k=1}^K \widetilde{\Delta\theta}_k^m, & \text{if } m \in \mathcal{M} \\ \theta_0^m, & \text{otherwise} \end{cases}$$

which applies subspace-based denoising immediately after each task and incrementally accumulates denoised LoRA updates. Algorithm 2 describes SLoRA-Post, which performs standard LoRA training across all tasks and applies denoising once after the full sequence is completed.

C.2 Efficient Computation of Low-Rank SVD

To improve efficiency, we approximate the singular value decomposition (SVD) of each LoRA update $\Delta\theta_k^m \in \mathbb{R}^{d' \times d}$ using a randomized projection method. As each update is formed via low-rank decomposition (i.e., $\Delta\theta_k^m = B_k^m A_k^m$ with $r \ll \min(d', d)$), computing a full-rank SVD is unnecessary.

We follow a standard randomized SVD pipeline with orthogonalization and projection:

1. **Random projection.** Sample a Gaussian random matrix $\mathbf{P} \in \mathbb{R}^{d \times c}$ and compute the sketch matrix $Y = \Delta\theta_k^m \mathbf{P} \in \mathbb{R}^{d' \times c}$.

2. **Orthonormal basis.** Perform QR decomposition $Y = \mathbf{QR}$, where $\mathbf{Q} \in \mathbb{R}^{d' \times c}$ has orthonormal columns.

3. **Low-dimensional projection.** Project the original matrix into the subspace spanned by $\mathbf{Q}$:
$$\mathbf{B} = \mathbf{Q}^\top \Delta\theta_k^m \in \mathbb{R}^{c \times d}. \tag{26}$$

4. **SVD in projected space.** Compute SVD of $\mathbf{B}$:
$$\mathbf{B} = \hat{\mathbf{U}} \mathbf{\Sigma} \mathbf{V}^\top, \tag{27}$$
where $\hat{\mathbf{U}} \in \mathbb{R}^{c \times c}$, $\mathbf{\Sigma} \in \mathbb{R}^{c \times c}$, and $\mathbf{V} \in \mathbb{R}^{c \times d}$.

5. **Approximate SVD.** Recover the approximate left singular vectors in the original space:
$$\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}} \in \mathbb{R}^{d' \times c}. \tag{28}$$

We then construct the rank-c approximation:

$$\Delta\theta_k^{m(c)} = \mathbf{U}_c \cdot \mathrm{diag}(\mathbf{\Sigma}_c) \cdot \mathbf{V}_c, \tag{29}$$

where $\mathbf{U}_c \in \mathbb{R}^{d' \times c}$ and $\mathbf{V}_c \in \mathbb{R}^{c \times d}$ denote the top-c singular directions, and $\mathrm{diag}(\mathbf{\Sigma}_c)$ forms a diagonal matrix from the top-c singular values.

To ensure reliable similarity measurement in the denoising stage, we apply a final SVD to $\Delta\theta_k^{m(c)}$ and retain the top-r left singular vectors as the subspace basis for alignment scoring.

Comparison with Standard SVD. The full SVD of a $d' \times d$ matrix costs $\mathcal{O}(d' \times d^2)$ time. In contrast, the randomized SVD reduces the complexity to $\mathcal{O}(d \times r^2)$, assuming $r \ll d'$, by operating in a lower-dimensional space. Our experiments show this approximation preserves denoising quality while significantly reducing computational overhead, enabling scalable continual learning with LLMs.

D AI Assistance Statement

Language editing and stylistic refinement of the draft were performed with the aid of large language models, including ChatGPT [4] and Google Gemini [5].

[4] https://chatgpt.com/
[5] https://gemini.google.com/app/