SPLITLORA: BALANCING STABILITY AND PLASTICITY IN CONTINUAL LEARNING THROUGH GRADIENT SPACE SPLITTING

Haomiao Qiu[1,2] , Miao Zhang[1]*, Ziyue Qiao[2]*, Weili Guan[1], Min Zhang[1], Liqiang Nie[1]
[1] Harbin Institute of Technology (Shenzhen)
[2] Great Bay University
24B951058@stu.hit.edu.cn, zhangmiao@hit.edu.cn, zyqiao@gbu.edu.cn
honeyguan@gmail.com, zhangmin2021@hit.edu.cn, nieliqiang@gmail.com

ABSTRACT

Continual Learning (CL) requires a model to learn multiple tasks in sequence while maintaining both stability—preserving knowledge from previously learned tasks, and plasticity—effectively learning new tasks. Gradient projection has emerged as an effective and popular paradigm in CL, where it partitions the gradient space of previously learned tasks into two orthogonal subspaces: a primary subspace and a minor subspace. New tasks are learned effectively within the minor subspace, thereby reducing interference with previously acquired knowledge. However, existing Gradient Projection methods struggle to achieve an optimal balance between plasticity and stability, as it is hard to appropriately partition the gradient space. In this work, we consider a continual learning paradigm based on Low-Rank Adaptation (LoRA), which has gained considerable attention due to its efficiency and wide applicability, and propose a novel approach for continual learning, called SplitLoRA. We first provide a theoretical analysis of how subspace partitioning affects model stability and plasticity. Informed by this analysis, we then introduce an effective method that derives the optimal partition of the gradient space for previously learned tasks. This approach effectively balances stability and plasticity in continual learning. Experimental results on multiple datasets demonstrate that the proposed method achieves state-of-the-art performance. The code is available at https://github.com/qhmiao/SplitLoRA.

1 INTRODUCTION

Continual Learning (CL) refers to a model's ability to sequentially learn new tasks while retaining knowledge from previously learned tasks (Parisi et al., 2019). This contrasts with traditional machine learning paradigms, which assume that models are trained on a fixed dataset where all data is available at once. In the CL setting, the challenge lies in maintaining performance on previous tasks while adapting to new ones, necessitating a balance between stability and plasticity. In recent years, orthogonal projection methods have demonstrated strong performance in continual learning tasks. These methods require storing the subspace spanned by the gradients of previous tasks in memory. During new task training, the gradient of the current task is projected onto the minor subspace of the previous task's gradient subspace, reducing the interference of new task updates with previously learned knowledge.

Parameter-Efficient Fine-Tuning (PEFT) (Hu et al., 2022; Houlsby et al., 2019; Jia et al., 2022) enables efficient fine-tuning for new tasks by keeping the pre-trained model parameters unchanged while introducing a small subset of trainable parameters. Due to its advantages in computational efficiency and performance, PEFT methods have gained increasing popularity in continual learning (Wang et al., 2022c; Smith et al., 2023; Gao et al., 2023; Liang & Li, 2024; Lu et al., 2024). Combining orthogonal projection with PEFT can better leverage the knowledge of the pre-trained model, allowing for faster adaptation to new tasks.

*Co-corresponding Authors.

However, existing methods (Liang & Li, 2024; Lu et al., 2024) typically determine the subspace dimension for each module within the model by using a predefined threshold based on the cumulative sum of squared singular values. This approach enforces a uniform partitioning rule across all modules, ignoring the fact that different modules contribute unequally to knowledge retention (Jiang et al., 2024; Dai et al., 2021; Geva et al., 2020). As a result, it fails to achieve an optimal trade-off between stability and plasticity.

In this paper, we theoretically analyze the relationship between the size of the minor gradient subspace of previous tasks and the upper bound of loss increments across all tasks. Furthermore, we model its impact on both stability and plasticity. In practice, we build upon the LoRA framework and propose a novel method called SplitLoRA for CL tasks. Specifically, to minimize the upper bound of the total task loss growth, we construct an optimization problem to determine the optimal size of minor subspace and derive an approximate solution to balance stability and plasticity.

Our contributions are summarized as follows:

- We theoretically model the impact of the gradient subspace size of previous tasks on stability and plasticity in orthogonal projection based continual learning in Theorem 4.2 and derive an approximate optimal minor subspace in CL.
- We introduce SplitLoRA, a novel PEFT framework. By projecting the minor subspace onto the LoRA dimension reduction matrix $\mathbf{A}_t$ via a random projection and optimizing only $\mathbf{B}_t$, SplitLoRA ensures that updates remain confined to the minor subspace, thereby achieving an effective balance between stability and plasticity.
- Our method achieves state-of-the-art performance across multiple datasets, surpassing existing CL methods by 2%–5% on different datasets.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning modifies pre-trained models by introducing a small set of trainable parameters while keeping the original model frozen, significantly reducing computational costs while maintaining strong performance. Adapter (Houlsby et al., 2019) fine-tunes small modules added to multiple layers, while Prompt-tuning (Lester et al., 2021) and Prefix-tuning (Li & Liang, 2021) inject trainable tokens into Transformer layers. LoRA (Hu et al., 2022) decomposes weight updates into low-rank matrices, tuning only these structures. Despite training fewer parameters, PEFT methods often achieve comparable or superior performance (Zaken et al., 2022; Fu et al., 2022; Hu et al., 2022; Mahabadi et al., 2021). Initially developed for NLP, PEFT has been extended to vision tasks, with methods such as Visual Prompt Tuning (VPT) (Jia et al., 2022) and AdapterFormer (Chen et al., 2022) achieving performance on par with full fine-tuning.

2.2 Continual Learning

Continual learning methods fall into three main categories: regularization-based, memory-based, and expansion-based. Regularization-based approaches (Zenke et al., 2017; Jung et al., 2020; Aljundi et al., 2018; Kirkpatrick et al., 2017) constrain significant changes to key parameters to mitigate catastrophic forgetting. Memory-based methods (Aljundi et al., 2019a;b; Sun et al., 2022; Liang & Li, 2023) retain prior task information in a buffer, allowing models to revisit past knowledge. Expansion-based techniques (Rusu et al., 2016; Hung et al., 2019; Li et al., 2019) dynamically expand the model architecture to accommodate new tasks while preserving learned representations.

Gradient Projection in CL. Gradient projection (Zeng et al., 2019; Farajtabar et al., 2020; Saha et al., 2021b) mitigates task interference by constraining updates to directions orthogonal to previous tasks. Orthogonal Weight Modulation (Zeng et al., 2019) learns a projector matrix to prevent new gradients from overwriting prior knowledge. Orthogonal Gradient Descent (Farajtabar et al., 2020) projects new gradients onto the orthogonal complement of previous task gradients. Gradient Projection Memory (GPM) (Saha et al., 2021b) stores subspace bases of old task data and projects new gradients onto their orthogonal complement. Trust Region Gradient (Lin et al., 2022a) enhances forward knowledge transfer by leveraging task-related representations.

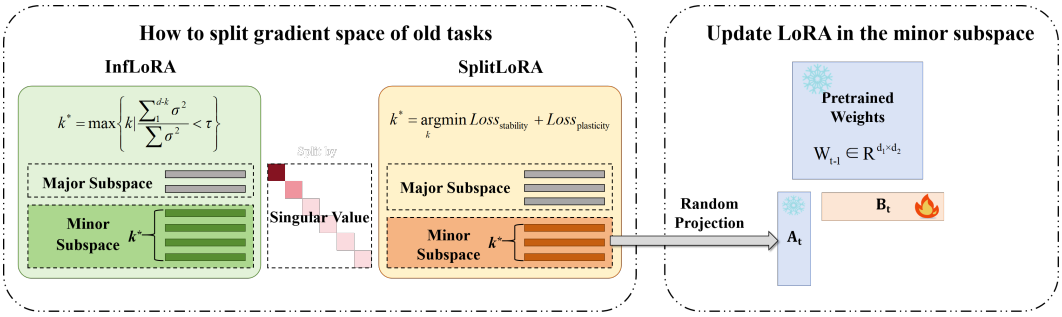

Figure 1: An overview of our proposed SplitLoRA. During the learning of the t-th task, the gradient space of tasks 1 to $t-1$ is decomposed into major and minor subspaces. InfLoRA determines k^* solely based on a predefined threshold, whereas SplitLoRA balances stability loss and plasticity loss to determine k^*. Then the minor subspaces are randomly projected onto the low-dimensional matrix A of LoRA and fixed, while only B is trained. Specifically, $W_{t-1} = W_0 + \Sigma_{i=1}^{t-1} A_i B_i$, where W_0 represents the pre-trained model weights, and k denotes the size of the minor subspace.

PEFT in CL. With the rise of pre-trained models (He et al., 2022; Dosovitskiy et al., 2021; Devlin et al., 2019), continual learning has shifted toward leveraging them rather than training from scratch. While some approaches (Boschini et al., 2022; Zheng et al., 2023) fine-tune pre-trained models fully, this is often inefficient. To address this, PEFT methods have been explored in continual learning, with studies (Smith et al., 2023; Wang et al., 2022c; Khan et al., 2023; Yu et al., 2024) integrating prompt-tuning to improve class-incremental learning. A unified framework (Gao et al., 2023) further combines various PEFT techniques, including prompt-tuning, LoRA, and Adapter, into continual learning.

3 Preliminary

3.1 Continual Learning Formulation

In CL, there are T tasks $\mathcal{T}_1, ..., \mathcal{T}_T$, each task includes data: $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$, where $\mathbf{x}_i^t \in \mathbb{R}^d$ is the input and $y_i^t \in \mathbb{R}$ is the label. The goal of CL is to achieve an overall optimal performance across all tasks. Let $\mathbf{W}_T$ represent the model parameters after training on the last task T. The loss on task t, denoted as $\mathcal{L}_t(\mathbf{W}_T)$, measures the performance of $\mathbf{W}_T$ on task t. Let $\mathbf{W}_T^*$ represent the optimal parameters. Then, the objective is to minimize the total loss across all tasks:

$$\mathbf{W}_T^* = \underset{\mathbf{W}_T}{\operatorname{argmin}}\, \mathcal{L}_{all}(\mathbf{W}_T) = \underset{\mathbf{W}_T}{\operatorname{argmin}} \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{W}_T). \tag{1}$$

3.2 LoRA-based Continual Learning

LoRA (Hu et al., 2022) is a low-rank based PEFT method that reduces the number of parameters by decomposing the weight matrix into the product of two low-rank matrices. Specifically, for a linear layer, the extra weight matrix $\Delta\mathbf{W}$ is decomposed into two low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ as : $\Delta\mathbf{W} = \mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_2}$, and r is the dimension of the low-rank. In this way, the number of parameters of the weight matrix is reduced from $d_1 d_2$ to $2dr$. During training, we learn $\mathbf{A}$ and $\mathbf{B}$ by minimizing the loss function of the current task. During testing, we recover $\mathbf{W} = \mathbf{W}_0 + \mathbf{AB}$ and use it for forward propagation.

CL generally initializes an additional LoRA for the new task t, while the LoRA of old tasks also participates in the forward process (Liang & Li, 2024; Wang et al., 2023b). In the current task t, the forward process of the linear layer is

$$\mathbf{Y} = \mathbf{W}_t\mathbf{X} = (\mathbf{W}_{t-1} + \mathbf{A}_t\mathbf{B}_t)\mathbf{X}, \tag{2}$$

and only $\mathbf{A}_t$ and $\mathbf{B}_t$ are trained during training.

4 SPLITLORA

In this section, we introduce SplitLoRA, a PEFT method for CL that mitigates catastrophic forgetting by partitioning the gradient space. Unlike existing Gradient Projection-based methods (Liang & Li, 2024; Lu et al., 2024; Jin et al., 2021), which typically define the minor subspace solely based on the sum of squared singular values being below a predefined threshold, SplitLoRA determines the optimal subspace size by analyzing the impact of the minor subspace on stability loss and plasticity loss. We first introduce gradient projection, then model the effect of subspace partitioning on learning dynamics and formulate an optimization problem to derive the approximate optimal size of the minor subspace. Finally, we present how to construct the low-rank projection matrix within this subspace to enhance CL. The entire process of SplitLoRA is illustrated in Figure 1.

4.1 ORTHOGONAL DECOMPOSITION BASED GRADIENT PROJECTION

Empirically, training on new tasks often leads to performance degradation on previously learned tasks due to interference between the gradients of new and old tasks, resulting in stability loss. To address this issue, GPM (Saha et al., 2021b) orthogonally decomposes the gradient space of previous tasks into a major subspace and a minor subspace, and constrains the update direction of the new task within the minor subspace. Our work is also built upon orthogonal decomposition.

In CL, we aim to maintain an average gradient space $\mathbf{G}^{\text{old}}$ for all previous tasks. Specifically, after training the task $t-1$, we re-feed the data from this task into the model and calculate the average gradient $\mathbf{G}_{t-1}^{\text{new}}$ of W throughout this process. Finally, we compute the average gradient space $\mathbf{G}_t^{\text{old}}$ for the previous $t-1$ tasks:

$$\mathbf{G}_t^{\text{old}} = \frac{1}{t-1}((t-2)\mathbf{G}_{t-1}^{\text{old}} + \mathbf{G}_{t-1}^{\text{new}}). \tag{3}$$

For the first task, $\mathbf{G}_1^{\text{old}}$ is equal to zero. Next, we receive the data from task t. We perform a Singular Value Decomposition (SVD) on the gradient $\mathbf{G}_t^{\text{old}}$. Larger singular values correspond to singular vectors that dominate in describing the vector's importance. We select the last k left singular vectors of $\hat{\mathbf{U}}_t$ as the minor subspace:

$$\hat{\mathbf{U}}_t, \hat{\mathbf{\Sigma}}_t, \hat{\mathbf{V}}_t^\top = \text{SVD}(\mathbf{G}_t^{\text{old}}), \qquad \hat{\mathbf{U}}_t^k = \hat{\mathbf{U}}_t[:, -k:]. \tag{4}$$

The gradient of previous tasks has a much smaller component in the minor subspace compared to the major subspace. Therefore, projecting the gradient of the new task onto the minor subspace will result in minimal interference.

A common projection method is to construct a projection transformation matrix. For simplicity, we consider a linear layer $\mathbf{W}$ in a model. As the model update $\Delta\mathbf{W}$ is determined by the gradient, projecting the model update onto the minor subspace is equivalent to constraining the gradient direction, which helps mitigate interference. Specifically, we project the model update $\Delta\mathbf{W}$ onto the minor subspace. The projection result is given by:

$$\Delta\hat{\mathbf{W}} = \text{proj}_{\text{col}(\hat{\mathbf{U}}_t^k)}(\Delta\mathbf{W}) = \hat{\mathbf{U}}_t^k \hat{\mathbf{U}}_t^{k\top} \Delta\mathbf{W}, \tag{5}$$

where $\hat{\mathbf{U}}_t^k$ is the projection subspace. Since the gradients of previous tasks are distributed primarily in the major subspace, therefore the projection ensures that the updates primarily benefit the new task while minimally affecting the performance of old tasks.

4.2 MINOR SPACE SETTING FOR OLD TASKS

Projecting the gradient onto the minor subspace can mitigate interference with previous tasks. And the larger the minor subspace size k, the larger the learning space for the new task, leading to better plasticity. However, as the gradient components of previous tasks in the minor subspace increase, stability deteriorates.

Previous methods (Jin et al., 2021; Lin et al., 2022a; Liang & Li, 2024; Lu et al., 2024) compute the sum of the squared singular values corresponding to the minor subspace, ensuring that it remains below a predefined threshold τ. Among all values of k that satisfy this condition, they select the

largest one:

$$k^* = \max\left\{k | \frac{\sum_{i=1}^{d-k}\sigma_i^2}{\sum\sigma^2} < \tau\right\}. \tag{6}$$

The size of the minor subspace, denoted as k, is a crucial parameter that affects both model stability and plasticity. However, previous methods determine k based on a predefined threshold τ, which is merely a hyperparameter and does not effectively balance stability and plasticity. Thus, we proceed to analyze how subspace selection impacts the loss across all tasks. Based on the smoothness of the loss function $\mathcal{L}$, we can derive an upper bound on the loss incurred due to parameter updates in CL.

Proposition 4.1 (Upper Bound on Loss Increase)**.** *Consider a model with a linear layer updated from $\mathbf{W}_{t-1}$ to $\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta\mathbf{W}_t$. Assume the loss function is L-smooth and that the first $t-1$ tasks were trained with updates constrained to be orthogonal to the gradients of previous tasks. Then, the total loss change over tasks $1, \ldots, t$ is bounded by:*

$$\sum_{i=1}^{t}\left(\mathcal{L}_i(\mathbf{W}_t) - \mathcal{L}_i(\mathbf{W}_{t-1})\right) \leq \underbrace{(t-1)\left\langle \Delta\mathbf{W}_t, \mathbf{G}_t^{old}\right\rangle}_{\textit{Stability Loss}} + \underbrace{\left\langle \Delta\mathbf{W}_t, \mathbf{G}_t\right\rangle}_{\textit{Plasticity Loss}} + \frac{(t-1)L}{2}\|\Delta\mathbf{W}_t\|_F^2, \tag{7}$$

where $\mathbf{G}_t = \nabla\mathcal{L}_t(\mathbf{W}_t)$ is the gradient for task t, and $\mathbf{G}_t^{old} = \frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{G}_i$ is the average gradient of previous tasks. $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product.

This result shows that parameter updates affect both the current and past tasks. The term $\langle\Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}}\rangle$ captures interference with past tasks (stability loss), while $\langle\Delta\mathbf{W}_t, \mathbf{G}_t\rangle$ reflects progress on the current task (plasticity gain). The squared norm $\|\Delta\mathbf{W}_t\|_F^2$ acts as a regularization term controlled by the smoothness constant L.

Next, we discuss how to choose the minor subspace in gradient projection to minimize the combined stability and plasticity losses. Building on Proposition 1, we can theoretically analyze how stability loss and plasticity loss vary as a function of k. From Eq. (7), after replacing $\Delta\mathbf{W}_t$ with $\Delta\hat{\mathbf{W}}_t$, where $\Delta\hat{\mathbf{W}}_t = \mathbf{U}_t^k\mathbf{U}_t^{k\top}\Delta\mathbf{W}_t$ is the projected update onto the minor subspace, we can express the stability loss $\mathcal{L}_t^S(\mathbf{W}_t)$ and the plasticity loss $\mathcal{L}_t^P(\mathbf{W}_t)$ as follows:

$$\mathcal{L}_t^S(\mathbf{W}_t) = (t-1)\left\langle\Delta\hat{\mathbf{W}}_t, \mathbf{G}_t^{\text{old}}\right\rangle, \tag{8}$$

$$\mathcal{L}_t^P(\mathbf{W}_t) = \left\langle\Delta\hat{\mathbf{W}}_t, \mathbf{G}_t\right\rangle. \tag{9}$$

The stability loss is proportional to the alignment between the projected update $\Delta\hat{\mathbf{W}}_t$ and the gradient of old tasks $\mathbf{G}_t^{\text{old}}$, while the plasticity loss depends on the alignment of $\Delta\hat{\mathbf{W}}_t$ with the gradient of the new task $\mathbf{G}_t$. Then we define the error function $\epsilon(k)$, which quantifies the proportion of these minor directions and is given by:

$$\epsilon(k) = \frac{\sum_{i=d-k+1}^{d}\sigma_i}{\sum_{i=1}^{d}\sigma_i}. \tag{10}$$

$\epsilon(k)$ measures the interference error caused by updating the model within the minor subspace.

To analyze the effect of subspace partitioning, we need to handle the fact that the new task's update direction $\Delta\mathbf{W}_t$ is unpredictable. Since we have no prior knowledge about its distribution, we adopt a neutral assumption: $\Delta\mathbf{W}_t$ is treated as having equal expected projection across all feature directions of $\mathbf{G}_t$. This assumption does not imply that gradients are truly uniform in practice, but rather provides a distribution-free baseline that allows us to derive general bounds. Under the above assumption, we obtain the following theorem:

Theorem 4.2. *Let $\mathbf{W}_{t-1}$ denote the weight matrix of a linear layer in the model, updated as $\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta\hat{\mathbf{W}}_t = \mathbf{W}_{t-1} + \mathbf{U^k}_t\mathbf{U^k}_t^{\top}\Delta\mathbf{W}_t$. We provide the expected values of the stability loss :*

$$\mathbb{E}[\mathcal{L}_t^S(\mathbf{W}_t)] = (t-1)\epsilon_t(k_t)\left\langle\Delta\mathbf{W}_t, \mathbf{G}_t^{old}\right\rangle, \tag{11}$$

and the plasticity loss:

$$\mathbb{E}[\mathcal{L}_t^P(\mathbf{W}_t)] = \frac{k_t}{d}\left\langle\Delta\mathbf{W}_t, \mathbf{G}_t\right\rangle. \tag{12}$$

Algorithm 1 SplitLoRA

1: **Input:** Datasets $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$, for T tasks $\mathcal{T}_1, ..., \mathcal{T}_T$, a pre-trained ViT model $f_\Theta(\cdot)$ with l layers.
2: **Output:** The optimized $\mathbf{W}_T^l$ for each layer l.
3: **Initialization:** $\mathbf{G}_1^{\text{old}} = \mathbf{0}$
4: **for** $t = 1$ to T **do**
5: Train LoRA on task $\mathcal{T}_t$ using dataset $\mathcal{D}_t$
6: Update $\mathbf{G}_t^{\text{old}}$ using Eq. (3) for each layer ▷ See Algorithm 2 Step 1
7: Compute k_t using Eq. (15) for each LoRA module ▷ See Algorithm 2 Step 2
8: Initialize $\mathbf{A}_t$ using Eq. (17) for each LoRA module ▷ See Algorithm 2 Step 3
9: **end for**

The proof of this theorem can be found in Appendix A.3. Therefore, achieving an optimal balance between these two objectives requires solving the following optimization problem:

$$k_t^* = \underset{k}{\text{argmin}} \left(\mathbb{E}[\mathcal{L}_t^S(\mathbf{W}_t)] + \mathbb{E}[\mathcal{L}_t^P(\mathbf{W}_t)] \right). \tag{13}$$

Noting that $\Delta\mathbf{W}_t$ and $\mathbf{G}_t$ gradually change as training progresses, solving this optimization problem is challenging. To simplify this, we introduce a ratio parameter α:

$$\alpha = -\frac{\langle \Delta\mathbf{W}_t, \mathbf{G}_t \rangle}{\langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle}. \tag{14}$$

This substitution reformulates the optimization problem into the following form:

$$k_t^* = \underset{k}{\text{argmin}} \left((t-1)\epsilon_t(k_t) - \alpha \frac{k_t}{d} \right). \tag{15}$$

Since $\Delta\mathbf{W}_t$ benefits new tasks, it often interferes with previous task knowledge, leading to:

$$\langle \Delta\mathbf{W}_t, \mathbf{G}_t \rangle < 0, \quad \langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle > 0. \tag{16}$$

From Eq. (15), it is evident that increasing k_t leads to higher $\epsilon_t(k_t)$, which increases stability loss $\mathcal{L}_t^F$, while expanding the learning space and thus reducing plasticity loss $\mathcal{L}_t^P$. However, since both $\Delta\mathbf{W}_t$ and $\mathbf{G}_t$ evolve during training, the ratio α in Eq. (14) also varies dynamically, whereas the update subspace must be determined before training begins for task t. Therefore, we explicitly treat α as a fixed hyperparameter, serving as a strategic control knob for the stability–plasticity trade-off. Importantly, our experiments (Tab. 5, Fig. 3) demonstrate that SplitLoRA is highly robust to the choice of α; performance remains consistently superior to baselines across a wide range of values.

In the simplified optimization problem of Eq.(15), the parameter k_t is restricted to integer values within the range $[1, d]$. The optimal solution to this equation can be obtained by evaluating the objective function for all possible values of k_t and selecting the one that minimizes it as k_t^*. For example, in ViT-B/16 (Dosovitskiy et al., 2021), embedding dimension is 768 , and k can be selected as any integer between 1 and 768. It is worth noting that we compute k separately for each LoRA module, as weights at different layers and positions capture substantially different knowledge. A fixed threshold, as used in InfLoRA (Liang & Li, 2024), cannot effectively account for such variation across modules.

4.3 LoRA Updates in the Minor Subspace

To ensure LoRA updates remain within the minor subspace, we fix the projection matrix $\mathbf{A}_t$ and only optimize $\mathbf{B}_t$. LoRA parameterizes the weight update as: $\Delta\mathbf{W}_t = \mathbf{A}_t\mathbf{B}_t$. When $\mathbf{A}_t$ is fixed, the update is confined to its column space (Liang & Li, 2024; Wang et al., 2023b). To restrict this space to the minor subspace of previous tasks, we construct

$$\mathbf{A}_t = \hat{\mathbf{U}}_t^k \mathbf{R}, \tag{17}$$

where $\hat{\mathbf{U}}_t^k \in \mathbb{R}^{d \times k}$ is an orthonormal basis of the minor subspace and $\mathbf{R} \in \mathbb{R}^{k \times r}$ is a random Gaussian matrix. Importantly, this constraint only holds if $\mathbf{A}_t$ remains fixed during training, otherwise, the

Table 1: We present FAA (%) and CAA(%) on ImageNet-R under three incremental learning settings: "5-task," "10-task," and "20-task." All backbone networks are pre-trained on ImageNet-21K.

Method	Pub.	5-task		10-task		20-task	
		FAA (↑)	CAA (↑)	FAA (↑)	CAA (↑)	FAA (↑)	CAA (↑)
Upper-bound	–	84.09 ± 0.21	–	84.09 ± 0.21	–	84.09 ± 0.21	–
FT	–	18.74 ± 0.44	48.39 ± 0.58	10.12 ± 0.51	35.23 ± 0.92	4.75 ± 0.40	22.8 ± 0.37
FT++	–	60.42 ± 0.87	71.59 ± 0.50	48.93 ± 1.15	66.79 ± 0.92	35.98 ± 1.38	59.68 ± 0.95
L2P++ (Wang et al., 2022c)	CVPR22	70.83 ± 0.58	78.34 ± 0.47	69.29 ± 0.73	78.30 ± 0.69	65.89 ± 1.30	77.15 ± 0.65
Deep L2P++ (Wang et al., 2022c)	CVPR22	73.93 ± 0.37	80.14 ± 0.54	71.66 ± 0.64	79.63 ± 0.90	68.42 ± 1.20	78.68 ± 1.03
DualPrompt (Wang et al., 2022b)	ECCV22	73.05 ± 0.50	79.47 ± 0.40	71.32 ± 0.62	78.94 ± 0.72	67.87 ± 1.39	77.42 ± 0.80
CODA-P (Smith et al., 2023)	CVPR23	76.51 ± 0.38	82.04 ± 0.54	75.45 ± 0.56	81.59 ± 0.82	72.37 ± 1.19	79.88 ± 1.06
HiDe-Prompt (Wang et al., 2023a)	NeurIPS23	76.29 ± 0.10	78.77 ± 0.11	76.74 ± 0.18	78.76 ± 0.11	76.46 ± 0.06	78.76 ± 0.11
EvoPrompt (Kurniawan et al., 2024)	AAAI24	77.16 ± 0.18	82.22 ± 0.54	76.83 ± 0.08	82.09 ± 0.68	74.41 ± 0.23	80.96 ± 1.42
InfLoRA (Liang & Li, 2024)	CVPR24	79.82 ± 0.27	84.07 ± 0.48	78.10 ± 0.43	83.47 ± 1.23	73.81 ± 0.47	81.02 ± 0.56
VQ-Prompt (Jiao et al., 2024)	NeurIPS24	79.23 ± 0.29	82.96 ± 0.50	78.71 ± 0.22	83.24 ± 0.68	78.10 ± 0.22	82.70 ± 1.16
VPT-NSP^2 (Lu et al., 2024)	NeurIPS24	79.71 ± 0.22	84.54 ± 0.68	79.35 ± 0.19	84.92 ± 0.41	76.72 ± 0.44	82.91 ± 0.60
SD-LoRA (Yichen et al., 2025)	ICLR25	79.15 ± 0.20	83.01 ± 0.42	77.34 ± 0.35	82.04 ± 0.24	75.26 ± 0.37	80.22 ± 0.72
SplitLoRA	**This work**	**81.92 ± 0.29**	**85.83 ± 0.55**	**81.00 ± 0.17**	**85.84 ± 0.62**	**78.82 ± 0.28**	**84.57 ± 0.44**

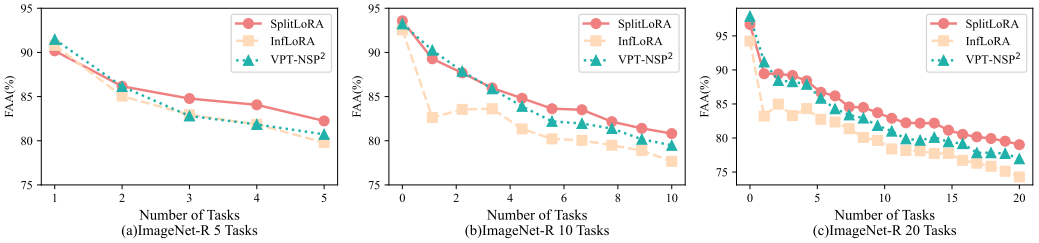

Figure 2: Variation of the performance of different methods during the learning of ImageNet-R.

update direction may drift out of the subspace. Fortunately, prior works (Zhang et al., 2023b; Liang & Li, 2024) verify that fixing $\mathbf{A}_t$ maintains sufficient model capacity while controlling interference. This design ensures that task updates are constrained to low-interference directions $\hat{\mathbf{U}}_t^k$, balancing stability and plasticity without additional memory or computational cost. The full procedure of SplitLoRA is summarized in Algorithm 1.

5 Experiment

5.1 Experimental Settings

Datasets. We conducted experiments on three standard datasets: ImageNet-R Hendrycks et al. (2021a), CIFAR-100 Krizhevsky (2009), and DomainNet Peng et al. (2019). ImageNet-R is a variant of ImageNet with 200 classes. CIFAR-100 consists of 100 classes, each containing 600 images. DomainNet contains images from diverse domains, posing a challenge for cross-domain generalization. Following Gao et al. (2023); Wang et al. (2022c); Liang & Li (2024), we divided ImageNet-R into 5, 10, and 20 tasks, with each task comprising 40, 20, and 10 classes, respectively. CIFAR-100 was split into 10 tasks, each containing 10 classes, while DomainNet was uniformly partitioned into 5 tasks.

Baselines and Evaluation Metrics. We compare our method with several state-of-the-art continual learning approaches, including L2P++ (Wang et al., 2022c), Deep L2P++ (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), CODA-P (Smith et al., 2023), HiDe-Prompt (Wang et al., 2023a), EvoPrompt (Kurniawan et al., 2024), VQ-Prompt (Jiao et al., 2024), VPT-NSP^2, and InfLoRA (Liang & Li, 2024). The "Upper bound" represents the performance achieved by jointly training on all classes in one go. Results are averaged over three runs with different random seeds. Following Jiao et al. (2024), we report Final Average Accuracy (FAA) and Cumulative Average Accuracy (CAA). For details, please refer to the appendix A.4.

Table 2: We present FAA (%) and CAA(%) on CIFAR100: 10 tasks and DomainNet: 5 tasks. We report results over 3 trials. All backbone networks are pre-trained on ImageNet-21K.

Method	Pub.	CIFAR100		DomainNet	
		FAA (↑)	CAA (↑)	FAA (↑)	CAA (↑)
Upper-bound	–	91.92 ± 0.05	–	90.12 ± 0.13	–
DualPrompt Wang et al. (2022b)	ECCV22	84.42 ± 0.30	90.06 ± 0.07	72.14 ± 0.05	77.71 ± 0.06
CODA-Prompt Smith et al. (2023)	CVPR23	86.62 ± 0.11	91.08 ± 0.28	73.23 ± 0.13	78.72 ± 0.07
LAE Gao et al. (2023)	ICCV23	84.15 ± 0.16	89.84 ± 0.03	66.85 ± 0.40	75.01 ± 0.17
C-LoRA Smith et al. (2024a)	TMLR24	82.97 ± 0.47	88.81 ± 0.34	69.34 ± 0.16	75.25 ± 0.11
InfLoRA Liang & Li (2024)	CVPR24	87.06 ± 0.25	91.59 ± 1.43	78.26 ± 0.50	78.82 ± 0.34
VPT-NSP^2 (Lu et al., 2024)	NeurIPS24	88.04 ± 0.11	92.25 ± 0.80	83.83 ± 0.19	88.63 ± 0.10
CoSO (Cheng et al., 2025)	2025	88.76 ± 0.16	92.99 ± 0.23	74.27 ± 0.07	80.05 ± 0.04
SplitLoRA	**This work**	$\mathbf{90.33 \pm 0.73}$	$\mathbf{93.70 \pm 0.32}$	$\mathbf{84.31 \pm 0.23}$	$\mathbf{88.99 \pm 0.57}$

Implementation Details. As the mainstream of recent continual learning research has primarily centered on parameter-efficient fine-tuning with pre-trained ViT models, we also adopt this setting for our experiments. We follow prior works (Wang et al., 2022c;b; Smith et al., 2023; Wang et al., 2023a; Zhang et al., 2023a; Kurniawan et al., 2024) and adopt ViT-Base (Dosovitskiy et al., 2021) pre-trained on ImageNet-21K (Ridnik et al., 2021) as the backbone. The LoRA rank is set to 10, and the embedding dimension is $D = 768$, matching the feature dimension of ViT-BaseDosovitskiy et al. (2021). Following Liang & Li (2024), we insert SplitLoRA modules into the key and value projections in multi-head attention. Unless otherwise specified, we fix $\alpha = 20$. Our method is optimized using AdamW (Loshchilov & Hutter, 2018) with an initial learning rate of $1e{-}3$ for LoRA and $1e{-}2$ for the classification head. We use a batch size of 256 across all datasets, and each task is trained for 10 epochs. All experiments are conducted on a single NVIDIA GeForce L40S GPU. All results are reported as mean ± standard deviation over three random seeds.

5.2 Experimental Results

Results on ImageNet-R, CIFAR100 and Domainet. Table 1 presents the results of different methods evaluated on ImageNet-R with varying numbers of tasks. It highlights how our proposed method, SplitLoRA, achieves consistently higher accuracy compared to existing continual learning methods across different task setups. Additionally, Table 2 shows the results of these methods on CIFAR100 and DomainNet datasets. Across both tables, SplitLoRA outperforms other methods in FAA and CAA. Figure 2 shows the accuracy trends of various CL methods on ImageNet-R. Our method achieves the highest accuracy at the end and outperforms others throughout the learning curve.

Initialization strategies of $\mathbf{A}_t$. Table 3 compares different initialization strategies for $\mathbf{A}_t$. SplitLoRA achieves consistently better performance across task splits, demonstrating the effectiveness of using projected minor subspace over random or InfLoRA.

Table 3: Impact of different $\mathbf{A}_t$ initialization strategies on ImageNet-R.

Init of $\mathbf{A}_t$	5 tasks	10 tasks	20 tasks
Random	76.57	76.13	72.30
InfLoRA	78.92	78.10	73.81
SplitLoRA	**81.92**	**81.00**	**78.82**

Table 4: Efficiency of LoRA variants on ImageNet-R (10 tasks).

Method	Extra Fwd	Mem	Time
LoRA	None	22.80 GB	1h 37m
InfLoRA	2/task	23.06 GB	1h 48m
SplitLoRA	1/task	23.03 GB	1h 43m

Memory and Time Cost. Table 4 shows that SplitLoRA achieves a favorable trade-off between performance and efficiency. It introduces only 1 extra forward pass per task while maintaining similar memory and runtime overheads compared to InfLoRA, and its memory cost remains fixed without growing with the number of tasks.

Table 5: Evaluation of model performance under different values of α on ImageNet-R. A higher α may improve plasticity but could impact stability.

Method	5-task		10-task		20-task	
	FAA (↑)	CAA (↑)	FAA (↑)	CAA (↑)	FAA (↑)	CAA (↑)
InfLoRA	79.82	84.07	78.10	83.47	73.81	81.02
SplitLoRA($\alpha = 30$)	82.15	85.60	81.03	85.56	78.73	84.06
SplitLoRA($\alpha = 20$)	81.92	85.83	81.00	85.84	78.82	84.57
SplitLoRA($\alpha = 10$)	82.35	85.82	81.03	85.67	77.89	83.27
SplitLoRA($\alpha = 5$)	82.52	85.89	81.38	85.89	78.15	84.19
SplitLoRA($\alpha = 1$)	82.40	85.86	80.89	85.22	78.59	84.20

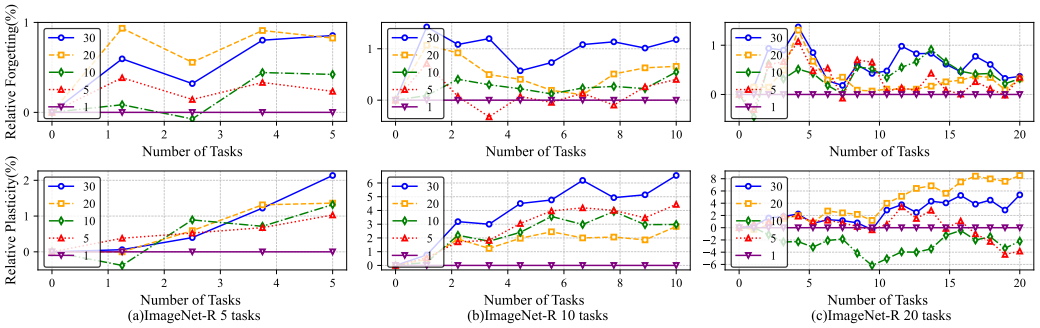

Figure 3: The impact of α on the stability and plasticity of the model in continual learning. As α increases, stability decreases (higher forgetting) while plasticity improves, illustrating the trade-off between retaining past knowledge and adapting to new tasks.

5.3 Hyperparameter Analysis and Discussion

We study the effect of the hyperparameter α on continual learning performance. As shown in Table 5, changing α has limited impact on final accuracy, and all settings consistently outperform InfLoRA. Model stability is measured by *forgetting*, defined as the average gap between each task's best historical accuracy and its current accuracy. Lower forgetting indicates better knowledge retention. For clarity, we define *relative forgetting* as the difference from the setting where $\alpha = 1$. Plasticity is evaluated by the model's accuracy on the current task. Similarly, *relative plasticity* is defined as the difference from the plasticity when $\alpha = 1$.

Figure 3 presents results on 5-, 10-, and 20-task splits of ImageNet-R. As α increases, forgetting grows (lower stability) while plasticity improves. These results show that α effectively controls the trade-off between retaining past knowledge and adapting to new tasks, while consistently maintaining better performance than InfLoRA across all settings.

5.4 Effectiveness of Optimizing Subspace Dimensions vs. Threshold Tuning

A central question is whether introducing a new hyperparameter for optimization offers a real advantage over directly tuning the threshold τ, as done in InfLoRA. Different layers in a model serve distinct roles (Zheng et al., 2025). Prior methods such as InfLoRA employ a fixed empirical threshold τ to determine subspace dimensions. However, this fails to accommodate the heterogeneous learning needs across layers, often resulting in shallow layers being overly constrained and deep layers retaining excessive capacity for past tasks.

Layer-wise Subspace Allocation. SplitLoRA replaces the single heuristic threshold with a principled optimization framework. Our objective function (Eq. 13) is derived from the total loss and allows us to analytically compute an optimal subspace dimension for each LoRA module within each task. This ensures that each layer selects a gradient space partitioning strategy that minimizes the overall loss. To illustrate, Tab. **??** compares the subspace dimensions selected by InfLoRA and SplitLoRA on the 10th task in the 20-task ImageNet-R setting. While InfLoRA under-allocates capacity in shallow

Table 6: Comparison of major subspace dimensions selected at different layers on the 10th task (ImageNet-R, 20-task setting).

Layer	0	1	2	6	9	11
InfLoRA	14	13	30	52	105	85
SplitLoRA	65	48	58	58	68	70

Table 7: Strategy transplantation results on ImageNet-R with 5, 10, and 20 tasks. Replacing layer-0 strategy of InfLoRA with SplitLoRA significantly improves performance, especially in reducing forgetting.

Method	5-task			10-task			20-task		
	FAA (↑)	CAA (↑)	Forget. (↓)	FAA (↑)	CAA (↑)	Forget. (↓)	FAA (↑)	CAA (↑)	Forget. (↓)
InfLoRA	79.82	84.07	10.03	78.10	83.47	9.40	73.81	81.02	12.34
w/ layer0→SplitLoRA	81.59	85.57	4.00	80.35	84.68	4.11	76.68	82.39	5.48
SplitLoRA	81.92	85.83	4.45	81.00	85.84	4.68	78.82	84.57	4.66

layers (layers 0–2), SplitLoRA learns to preserve significantly larger subspaces, implicitly protecting critical shallow representations from catastrophic forgetting.

Strategy Transplantation. To validate this observation explicitly, we designed a *strategy transplantation* experiment. In the 20-task ImageNet-R setting, we replaced the subspace selection strategy of layer 0 in InfLoRA with that of SplitLoRA. This modification significantly improved InfLoRA's performance, particularly in terms of reduced forgetting. Nonetheless, the modified InfLoRA still lagged behind SplitLoRA in new-task performance, suggesting that managing knowledge retention in shallow layers is a key factor, but not the sole determinant, of SplitLoRA's superiority.Table 6 reports results across ImageNet-R benchmarks with 5, 10, and 20 tasks. Compared to InfLoRA, transplanting only the layer-0 strategy from SplitLoRA already yields notable gains: forgetting is halved (e.g., from 10.03 to 4.00 in 5-task setting) and both FAA and CAA improve consistently. Nevertheless, SplitLoRA achieves the best overall balance between forward accuracy and forgetting, highlighting that its advantage lies not only in shallow layer protection but also in its holistic optimization framework.These results clearly demonstrate that SplitLoRA's learned subspace allocation strategy provides substantial advantages over direct threshold tuning. Protecting shallow layers plays a pivotal role in mitigating forgetting, while the full optimization framework ensures superior overall accuracy across continual learning tasks.

6 Conclusion

In this paper, we investigate the problem of continual learning based on pre-trained ViT models and propose a novel method, SplitLoRA. Specifically, we analyze the gradient space of previously learned tasks and partition it into two complementary components: a major subspace, which captures the most informative and stable directions, and a minor subspace, which contains the remaining, less significant directions. We then provide a theoretical formulation to model the effect of the minor subspace size on the trade-off between stability (the ability to retain past knowledge) and plasticity (the ability to acquire new knowledge). To make the method efficient and scalable, we employ random projection to map the minor subspace onto the low-dimensional matrices of LoRA. Extensive experiments on multiple benchmark datasets demonstrate that our approach achieves state-of-the-art performance, highlighting both its theoretical soundness and practical effectiveness.

Acknowledgment

Miao Zhang was partially sponsored by the National Natural Science Foundation of China under Grant 62306084 and U23B2051, Shenzhen College Stability Support Plan under Grant GXWD20231128102243003, and Shenzhen Science and Technology Program under Grant ZDSYS20230626091203008 and KJZD20230923115113026. Weili Guan was partially sponsored by the National Natural Science Foundation of China under Grant 62476071 and U24A20328.

The work of Ziyue Qiao was partially supported by the National Natural Science Foundation of China (No. 62406056) and the Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515140114).

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pp. 139–154, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pp. 11849–11860, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019b.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *NeurIPS*, 2019.

Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *Proceedings of the European Conference on Computer Vision*, pp. 692–709, 2022.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *NeurIPS*, 33:9900–9911, 2020.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, pp. 16664–16678, 2022.

Quan Cheng, Yuanyu Wan, Lingyu Wu, Chenping Hou, and Lijun Zhang. Continuous subspace optimization for continual learning. *arXiv preprint arXiv:2505.11816*, 2025.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *AISTATS*, pp. 3762–3773, 2020.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-Yi Lee. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. In *Findings of the Association for Computational Linguistics*, pp. 2608–2621, 2022.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11449–11459, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021b.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the International Conference on Machine Learning*, pp. 2790–2799, 2019.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems*, pp. 13647–13657, 2019.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of the European Conference on Computer Vision*, pp. 709–727, 2022.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. Taia: Large language models are out-of-distribution data learners. *arXiv preprint arXiv:2405.20192*, 2024.

Li Jiao, Qiuxia Lai, Yu Li, and Qiang Xu. Vector quantization prompting for continual learning. *arXiv preprint arXiv:2410.20444*, 2024.

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *NeurIPS*, 34:29193–29205, 2021.

Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *ICCV*, pp. 11847–11857, 2023.

Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in Neural Information Processing Systems*, pp. 3647–3658, 2020.

Muhammad Gul Zain Ali Khan, Muhammad Ferjad Naeem, Luc Van Gool, Didier Stricker, Federico Tombari, and Muhammad Zeshan Afzal. Introducing language guidance in prompt-based continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11463–11473, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

Muhammad Rifki Kurniawan, Xiang Song, Zhiheng Ma, Yuhang He, Yihong Gong, Yang Qi, and Xing Wei. Evolving parameterized prompt memory for continual learning. In *AAAI*, volume 38, pp. 13301–13309, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

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

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the International Conference on Machine Learning*, pp. 3925–3934, 2019.

Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems*, 2023.

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

Yan-Shuo Liang, Jiarui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. *arXiv preprint arXiv:2202.02931*, 2022a.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. In *International Conference on Learning Representations (ICLR)*. ICLR, 2022b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

Yue Lu, Shizhou Zhang, De Cheng, Yinghui Xing, Nannan Wang, Peng Wang, and Yanning Zhang. Visual prompt tuning in null space for continual learning. *arXiv preprint arXiv:2406.05658*, 2024.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, pp. 1022–1035, 2021.

Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *arXiv preprint arXiv:2307.02251*, 2023.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1406–1415, 2019.

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

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *NeurIPS*, 2021.

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

Gobinda Saha and Kaushik Roy. Continual learning with scaled gradient projection. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pp. 9677–9685. AAAI, 2023.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations (ICLR)*. ICLR, 2021a.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *ICLR*, 2021b.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11909–11919, 2023.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *archiveEprint: 2304.06027*, 2024a. URL https://arxiv.org/abs/2304.06027.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *Transactions on Machine Learning Research*, 2024b.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. *Advances in Neural Information Processing Systems*, pp. 27075–27086, 2022.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *arXiv preprint arXiv:2310.07234*, 2023a.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*, 2023b. URL https://arxiv.org/abs/2310.14152.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*, 2023c.

Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *AAAI*, volume 37, pp. 10209–10217, 2023d.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022a.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pp. 631–648, 2022b.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022c.

Chengyi Yang, Mingda Dong, Xiaoyue Zhang, Jiayin Qi, and Aimin Zhou. Introducing common null space of gradients for gradient projection methods in continual learning. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*. ACM, 2024.

Wu Yichen, Piao Hongming, Huang Long-Kai, Wang Renzhen, Li Wanhua, Pfister Hanspeter, Meng Deyu, Ma Kede, and Ying Wei. S-lora: Scalable low-rank adaptation for class incremental learning. *Under review at International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=5U1rlpX68A.

Hao Yu, Xin Yang, Xin Gao, Yan Kang, Hao Wang, Junbo Zhang, and Tianrui Li. Personalized federated continual learning via multi-granularity prompt. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4023–4034, 2024.

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

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pp. 3987–3995, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICCV*, 2023a.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023b. URL https://arxiv.org/abs/2308.03303.

Yuanhan Zhang, Zhenfei Yin, Jing Shao, and Ziwei Liu. Benchmarking omni-vision representation through the lens of visual realms. In *ECCV*, pp. 594–611. Springer, 2022.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. *arXiv preprint arXiv:2501.13453*, 2025.

Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. Preventing zero-shot transfer degradation in continual learning of vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19068–19079, 2023.

Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*, 2023.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*, 2024a.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024b.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Image bert pre-training with online tokenizer. In *ICLR*, 2022.

A APPENDIX

A.1 AUTHOR CONTRIBUTIONS AND THE USE OF LARGE LANGUAGE MODELS

The authors confirm that all core research contributions, encompassing the conceptualization, methodology, data analysis, and derivation of scientific conclusions, were performed entirely by the human authors without assistance from large language models. The fundamental intellectual content and technical execution of this work are original human efforts. Large Language Models (LLMs) were used only for editorial purposes to refine the presentation of the final manuscript, specifically for improving the grammar, clarity, and flow of the written text.

A.2 PROOF OF PROPOSITION 4.1

Before proving Proposition 4.1, we first establish a supporting lemma.

Lemma A.1 (Gradient Preservation under Orthogonal Updates)**.** *Let $\mathcal{L}_j : \mathbb{R}^d \to \mathbb{R}$ be a twice-differentiable loss function corresponding to task j, and let $\mathbf{W}_j$ be the model parameters after completing task j. Suppose at step $t > j$, the update direction $\tilde{g}_t$ for task t satisfies $\langle \nabla \mathcal{L}_j(\mathbf{W}_j), \tilde{g}_t \rangle = 0$. The updated parameter is given by: $\mathbf{W}_t = \mathbf{W}_j - \eta \tilde{g}_t$. Further assume that the second-order term $\eta H_j \tilde{g}_t$ in the Taylor expansion of $\nabla \mathcal{L}_j$ can be ignored. Then, the gradient of task j remains unchanged:*

$$\nabla \mathcal{L}_j(\mathbf{W}_t) = \nabla \mathcal{L}_j(\mathbf{W}_j).$$

Proof. Since $\mathcal{L}_j$ is twice-differentiable, we apply the first-order Taylor expansion of the gradient at point W_j in the direction of $\tilde{g}_t$:

$$\nabla \mathcal{L}_j(\mathbf{W}_t) = \nabla \mathcal{L}_j(\mathbf{W}_j - \eta \tilde{g}_t) = \nabla \mathcal{L}_j(\mathbf{W}_j) - \eta H_j \tilde{g}_t + o(\eta).$$

Now, under the assumption that $\eta H_j \tilde{g}_t$ is negligible (i.e., small learning rate and low curvature), we ignore the second-order term:

$$\nabla \mathcal{L}_j(\mathbf{W}_t) = \nabla \mathcal{L}_j(\mathbf{W}_j).$$

□

Based on the Lemma A.1, we denote $\nabla \mathcal{L}_i(\mathbf{W}_j)$ as $\mathbf{G}_i$. Next, we provide the proof of Proposition 4.1.

Proposition 4.1. Assume the loss $\mathcal{L}_i(\mathbf{W})$ is L-smooth for all $i \in \{1, \ldots, t\}$. Let the model update be $\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta \mathbf{W}_t$. Then:

$$\sum_{i=1}^{t} \left(\mathcal{L}_i(\mathbf{W}_t) - \mathcal{L}_i(\mathbf{W}_{t-1}) \right) \leq (t-1) \left\langle \Delta \mathbf{W}_t, \mathbf{G}_t^{\text{old}} \right\rangle + \langle \Delta \mathbf{W}_t, \mathbf{G}_t \rangle + \frac{(t-1)L}{2} \|\Delta \mathbf{W}_t\|_F^2.$$

Proof. By L-smoothness of each $\mathcal{L}_i$, we have:

$$\mathcal{L}_i(\mathbf{W}_t) \leq \mathcal{L}_i(\mathbf{W}_{t-1}) + \langle \nabla \mathcal{L}_i(\mathbf{W}_{t-1}), \Delta \mathbf{W}_t \rangle + \frac{L}{2} \|\Delta \mathbf{W}_t\|_F^2.$$

Summing over $i = 1$ to t:

$$\sum_{i=1}^{t} \mathcal{L}_i(\mathbf{W}_t) - \sum_{i=1}^{t} \mathcal{L}_i(\mathbf{W}_{t-1}) \leq \sum_{i=1}^{t} \langle \nabla \mathcal{L}_i(\mathbf{W}_{t-1}), \Delta \mathbf{W}_t \rangle + \frac{tL}{2} \|\Delta \mathbf{W}_t\|_F^2.$$

Let $\mathbf{G}_t^{\text{old}} = \frac{1}{t-1} \sum_{i=1}^{t-1} \nabla \mathcal{L}_i(\mathbf{W}_{t-1})$, then:

$$\sum_{i=1}^{t-1} \langle \nabla \mathcal{L}_i(\mathbf{W}_{t-1}), \Delta \mathbf{W}_t \rangle = (t-1) \langle \mathbf{G}_t^{\text{old}}, \Delta \mathbf{W}_t \rangle.$$

Substituting back gives:

$$\sum_{i=1}^{t} \left(\mathcal{L}_i(\mathbf{W}_t) - \mathcal{L}_i(\mathbf{W}_{t-1}) \right) \leq (t-1) \langle \Delta \mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle + \langle \Delta \mathbf{W}_t, \mathbf{G}_t \rangle + \frac{(t-1)L}{2} \|\Delta \mathbf{W}_t\|_F^2.$$

□

A.3 PROOF OF THEOREM 4.2

Let $\mathbf{W}_{t-1}$ denote the weight matrix of a linear layer in the model, updated as $\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta\hat{\mathbf{W}}_t = \mathbf{W}_{t-1} + \mathbf{U}^{\mathbf{k}}{}_t \mathbf{U}^{\mathbf{k}}{}_t^{\top} \Delta\mathbf{W}_t$. We provide the expected values of the stability loss :

$$\mathbb{E}[\mathcal{L}_t^S(\mathbf{W}_t)] = (t-1)\epsilon_t(k_t) \left\langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \right\rangle, \tag{18}$$

and the plasticity loss:

$$\mathbb{E}[\mathcal{L}_t^P(\mathbf{W}_t)] = \frac{k_t}{d} \left\langle \Delta\mathbf{W}_t, \mathbf{G}_t \right\rangle. \tag{19}$$

Proof. Stability Loss:

By definition, the projected update is:

$$\Delta\hat{\mathbf{W}}_t = \mathbf{U}_t^k \mathbf{U}_t^{k\top} \Delta\mathbf{W}_t.$$

Thus, the expected stability loss is:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}_t^S] &= (t-1)\mathbb{E}[\langle \Delta\hat{\mathbf{W}}_t, \mathbf{G}_t^{\text{old}} \rangle] \\
&= (t-1)\mathbb{E}[\langle \mathbf{U}_t^k \mathbf{U}_t^{k\top} \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle] \\
&= (t-1)\mathbb{E}[\text{Tr}(\Delta\mathbf{W}_t^{\top} \mathbf{U}_t^k \mathbf{U}_t^{k\top} \mathbf{G}_t^{\text{old}})].
\end{aligned}$$

Let $\mathbf{G}_t^{\text{old}} = \sum_{i=1}^d \sigma_i \mathbf{u}_i \mathbf{v}_i^{\top}$ be the SVD. Then,

$$\mathbf{U}_t^k \mathbf{U}_t^{k\top} \mathbf{G}_t^{\text{old}} = \sum_{i=d-k_t+1}^{d} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\top}.$$

So:

$$\mathbb{E}[\mathcal{L}_t^S] = (t-1) \sum_{i=d-k_t+1}^{d} \sigma_i \cdot \mathbb{E}[\langle \Delta\mathbf{W}_t, \mathbf{u}_i \mathbf{v}_i^{\top} \rangle_F].$$

Under the uniform distribution assumption, all expected projections are equal:

$$\mathbb{E}[\langle \Delta\mathbf{W}_t, \mathbf{u}_i \mathbf{v}_i^{\top} \rangle_F] = c, \quad \forall i.$$

Then:

$$\mathbb{E}[\mathcal{L}_t^S] = (t-1) \cdot c \cdot \sum_{i=d-k_t+1}^{d} \sigma_i.$$

Also,

$$\langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle = \sum_{i=1}^{d} \sigma_i \cdot \langle \Delta\mathbf{W}_t, \mathbf{u}_i \mathbf{v}_i^{\top} \rangle_F = c \cdot \sum_{i=1}^{d} \sigma_i,$$

so:

$$c = \frac{\langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle}{\sum_{i=1}^d \sigma_i}.$$

Thus,

$$\mathbb{E}[\mathcal{L}_t^S] = (t-1) \cdot \epsilon_t(k_t) \cdot \langle \Delta\mathbf{W}_t, \mathbf{G}_t^{\text{old}} \rangle.$$

—

Plasticity Loss:

The plasticity loss is:

$$\mathbb{E}[\mathcal{L}_t^P] = \mathbb{E}[\langle \Delta\hat{\mathbf{W}}_t, \mathbf{G}_t \rangle] = \mathbb{E}[\langle \mathbf{U}_t^k \mathbf{U}_t^{k\top} \Delta\mathbf{W}_t, \mathbf{G}_t \rangle].$$

Let $\alpha_i = \langle \Delta \mathbf{W}_t, \mathbf{u}_i \rangle$, $\beta_i = \langle \mathbf{G}_t, \mathbf{u}_i \rangle$. Then:

$$\mathbb{E}[\mathcal{L}_t^P] = \mathbb{E}\left[\sum_{i=1}^{k_t} \alpha_i \beta_i\right].$$

Under the uniform assumption, the expected contribution over any direction is $\frac{1}{d}$, hence:

$$\mathbb{E}[\mathcal{L}_t^P] = -\frac{k_t}{d} \cdot \langle \Delta \mathbf{W}_t, \mathbf{G}_t \rangle.$$

□

A.4 Evaluation metrics

To evaluate continual learning performance, we track the average classification accuracy over all classes encountered so far at the end of each task's training following (Jiao et al., 2024). We denote by A_{ij} the average accuracy on the i-th task after training the j-th task. Below, we provide formal definitions for two key metrics: FAA and CAA.

(i) Final Average Accuracy (FAA). FAA measures the overall performance after learning all tasks, defined as:

$$\text{FAA} = \frac{1}{T} \sum_{i=1}^{T} A_{iT}, \tag{20}$$

where T is the total number of tasks and A_{iT} is the accuracy for task i after completing task T. A larger FAA indicates a stronger ability to learn while minimizing forgetting. In some literature, FAA is also referred to as "Last-Acc."

(ii) Cumulative Average Accuracy (CAA). CAA is the average of the FAA values computed after each task is learned, given by:

$$\text{CAA} = \frac{1}{T} \sum_{j=1}^{T} \frac{1}{j} \sum_{i=1}^{j} A_{ij}. \tag{21}$$

It captures the overall performance at every incremental step. This metric is sometimes referred to as "Inc-Acc."

A.5 The size of minor subspace evolves during training.

We tracked the evolution of the minor subspace size throughout training on ImageNet-R with 20 tasks. As shown in the Fig. 4, as the number of tasks increases, model stability becomes more critical, leading to a progressively smaller minor subspace. Furthermore, when comparing different layers of ViT, the minor subspace is larger in shallower layers and gradually decreases as the layer depth increases. This suggests that changes in the deep-layer parameters have a greater impact on model stability.

When the value of α changes, the learning space of the model varies significantly. Nevertheless, the model is still able to learn tasks effectively under different settings of α. Furthermore, as the number of tasks approaches infinity, the size of the minor subspace gradually shrinks toward zero, raising the question of whether the model can still maintain its learning ability in this limit. To provide intuition, we conduct a simple experiment to explain this phenomenon.

Table 8: Average accuracy of tasks 2–20 under different fine-tuning strategies.

Method	Avg. Acc (Tasks 2–20)
Only head	66.08
Only head and the first task	74.75
SplitLoRA	81.47

Table 9: We present FAA (%) and CAA(%) on ImageNet-R: 10-tasks. Backbones are with different self-supervised pre-training paradigms: iBOT-1K and DINO-1K.

Method	Pub.	iBOT-1K		DINO-1K	
		FAA ↑	CAA ↑	FAA ↑	CAA ↑
Upper-bound	–	84.09 ± 0.21	–	81.98 ± 0.07	–
DualPrompt Wang et al. (2022b)	ECCV22	61.51 ± 1.05	67.11 ± 0.08	58.57 ± 0.45	64.89 ± 0.15
CODA-Prompt Smith et al. (2023)	CVPR23	66.56 ± 0.68	73.14 ± 0.57	63.15 ± 0.39	69.73 ± 0.25
HiDe-Prompt Wang et al. (2023a)	NeurIPS23	71.33 ± 0.21	73.62 ± 0.13	68.11 ± 0.18	71.70 ± 0.01
InfLoRA Liang & Li (2024)	CVPR24	71.84 ± 0.09	78.29 ± 0.09	68.31 ± 0.28	76.15 ± 0.05
VPT-NSP^2 Lu et al. (2024)	NeurIPS24	73.85 ± 0.23	80.34 ± 0.60	69.45 ± 0.74	76.38 ± 0.50
VQ-Prompt Jiao et al. (2024)	NeurIPS24	71.68 ± 0.72	76.66 ± 0.40	68.42 ± 0.28	74.43 ± 0.58
SplitLoRA	**This work**	**74.58 ± 1.05**	**81.45 ± 1.72**	**70.49 ± 0.31**	**78.15 ± 1.13**

As Tab. 8 shows, "Only head" means training only the classifier head for each task, which can be regarded as the extreme case where the minor subspace is nearly zero. "Only head and the first task" means training the first task's LoRA together with the classifiers for all tasks. SplitLoRA is used for comparison. It can be observed that even when the minor subspace is extremely small (or effectively zero, as in the "Only head" setting), the model can still learn tasks to a reasonable extent. Moreover, when subsequent tasks are fine-tuned on the first task's LoRA, performance further improves. This suggests that the knowledge contained in the pre-trained model and the beneficial knowledge retained from old tasks provide sufficient support for learning new tasks, even under highly restricted learning spaces.

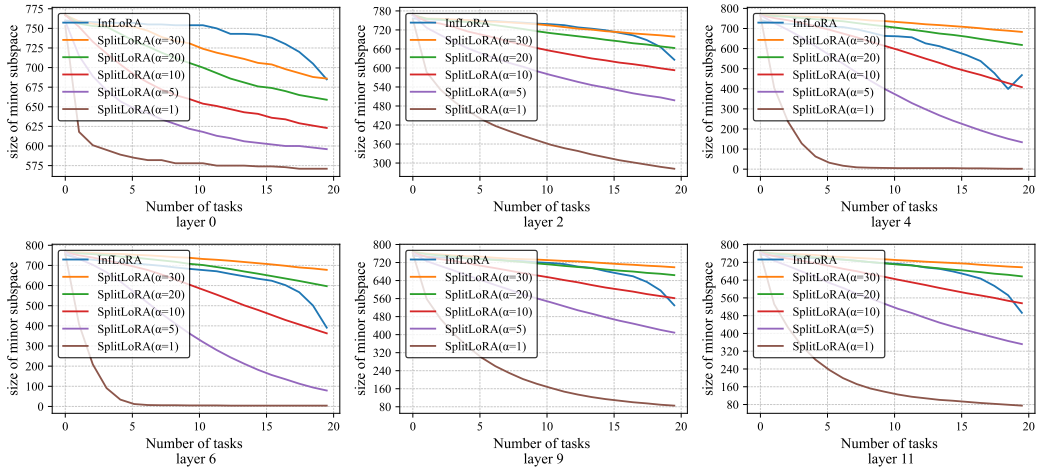

Figure 4: We recorded the evolution of the minor subspace size during training on ImageNet-R with 20 tasks.

A.6 Variant Pre-trained Models.

Tab. 9 provides a summary of experimental results on the 10-task ImageNet-R dataset using different self-supervised pre-training paradigms. Specifically, we evaluate our method with iBOT-1K (Zhou et al., 2022) and DINO-1K (Caron et al., 2021) pre-training frameworks. These results clearly demonstrate that SplitLoRA consistently outperforms state-of-the-art continual learning methods, irrespective of the pre-training paradigm used. This robustness underscores the generalizability and effectiveness of SplitLoRA in leveraging self-supervised pre-training for continual learning tasks.

A.7 More results on other benchmark.

We follow (Zhou et al., 2024a) to evaluate the performance on CIFAR100 (Krizhevsky, 2009), CUB200 (Wah et al., 2011), ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al.,

Table 10: Comparison with state-of-the-art methods on multiple benchmarks. We report CAA and FAA (%) on base and incremental sessions.

Method	CIFAR B0 Inc5	CUB B0 Inc10	IN-R B0 Inc5	IN-A B0 Inc20	Obj B0 Inc10	Omni B0 Inc30	VTAB B0 Inc10	Average
L2P	85.94 / 79.93	67.05 / 56.25	66.53 / 59.22	49.39 / 41.71	63.78 / 52.19	73.36 / 64.69	77.11 / 77.10	65.30
DualPrompt	87.87 / 81.15	77.47 / 66.54	63.31 / 55.22	53.71 / 41.67	59.27 / 49.33	73.92 / 65.52	83.36 / 81.23	67.11
CODA-Prompt	89.11 / 81.96	84.00 / 73.37	64.42 / 55.08	53.54 / 42.73	66.07 / 53.29	77.03 / 68.09	83.90 / 83.02	69.68
DAP	94.54 / 90.62	94.76 / 94.63	80.61 / 74.76	54.39 / 46.32	72.08 / 59.51	86.44 / 80.65	84.65 / 84.64	78.47
DAP w/o BI	68.07 / 58.16	65.27 / 52.05	50.40 / 37.99	34.48 / 21.84	50.47 / 37.55	65.43 / 52.53	79.63 / 79.87	53.83
SimpleCIL	87.57 / 81.26	92.20 / 86.73	62.58 / 54.55	59.77 / 48.91	65.45 / 53.59	79.34 / 73.15	85.99 / 84.38	72.53
ADAM + VPT-D	88.46 / 82.17	91.02 / 84.99	68.79 / 60.48	58.48 / 48.52	67.83 / 54.65	81.05 / 74.47	86.59 / 83.06	73.61
ADAM + SSF	87.78 / 81.98	91.72 / 86.13	68.94 / 60.60	61.30 / 50.03	69.15 / 56.64	80.53 / 74.00	85.66 / 81.92	74.02
ADAM + Adapter	90.65 / 85.15	92.21 / 86.73	72.35 / 64.33	60.47 / 49.37	67.18 / 55.24	80.75 / 74.37	85.95 / 84.35	74.93
RanPAC	93.51 / 89.30	93.13 / 89.40	75.74 / 68.75	64.16 / 52.86	71.67 / 60.08	85.95 / 79.55	92.56 / 91.83	79.17
EASE	91.51 / 85.80	92.23 / 86.81	78.31 / 70.58	65.34 / 55.04	70.84 / 57.86	81.11 / 74.85	93.61 / 93.55	78.39
HiDe-Prompt	91.22 / 89.92	89.75 / 89.46	76.20 / 74.56	61.41 / 49.27	70.13 / 62.84	76.60 / 77.01	91.24 / 92.78	78.02
ESN	87.15 / 80.37	65.69 / 63.10	60.69 / 55.13	44.06 / 31.07	63.73 / 52.55	75.32 / 66.57	81.52 / 62.15	63.50
SplitLoRA	93.11 / 90.84	91.52 / 87.46	81.74 / 74.81	66.11 / 58.22	68.60 / 61.33	82.30 / 76.86	94.39 / 91.95	**79.95**

2021b), ObjectNet (Barbu et al., 2019), Omnibenchmark (Zhang et al., 2022) and VTAB (Zhai et al., 2019) comparing with nine methods: L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), CODA-Prompt (Smith et al., 2023), DAP (Jung et al., 2023), SimpleCIL (Zhou et al., 2023), ADAM (Zhou et al., 2023), RanPAC (McDonnell et al., 2023), EASE (Zhou et al., 2024b), ESN (Wang et al., 2023d) and HiDe-Prompt (Wang et al., 2023a).

Tab. 10 compares SplitLoRA with state-of-the-art continual learning methods on seven benchmarks. SplitLoRA achieves the best average performance (79.95%) and consistently ranks top in individual tasks. In particular, it excels on ImageNet-R, Omni, and VTAB, demonstrating strong generalization and knowledge retention. This confirms that SplitLoRA effectively balances stability and plasticity across diverse scenarios.

A.8 LONG TASK SEQUENCES OF 50 AND 100 TASKS ON IMAGENET-R AND DOMAINNET.

To systematically evaluate the capacity of SplitLoRA, we further introduce long task sequences of 50 and 100 tasks on ImageNet-R and DomainNet. Results are shown below:

Table 11: Performance of SplitLoRA on long task sequences (50 and 100 tasks) on ImageNet-R and DomainNet.

Method	50-INR FAA	50-INR Forget	50-DN FAA	50-DN Forget	100-INR FAA	100-INR Forget	100-DN FAA	100-DN Forget
InfLoRA	59.00	11.02	69.96	9.51	38.16	15.11	44.32	17.85
VPT-NSP^2	69.48	6.51	71.28	11.36	62.23	12.13	57.35	13.82
SplitLoRA	**71.75**	7.59	70.53	10.18	**63.30**	12.29	**61.88**	9.27

From these 50-task and 100-task results, we observe that SplitLoRA maintains stable FAA even in extremely long sequences. When the number of tasks reaches 100, the model still retains strong continual learning ability.

A.9 MORE RESULTS ON CNN.

Our experiments is based on GPM (Saha et al., 2021a) and its extensions: TRGP (Lin et al., 2022b), SGP (Saha & Roy, 2023), and GPCNS (Yang et al., 2024). To ensure a fair comparison, we use the same network architectures as those employed in the baselines. For GPM-based experiments, we utilize a 5-layer AlexNet-like network as the backbone for the 10-Split and 20-Split CIFAR-100 datasets, and a reduced version of ResNet-18 for the 20-Split MiniImageNet experiment.

A.10 MORE FINE-GRAINED COMPUTATION OVERHEAD ANALYSIS.

We have added detailed training and evaluation (T&E) time, as well as the projection-update (PU) time, for the 5-tasks ImageNet-R setting. The table below reports all training and projection-update times in **minutes**.

Both InfLoRA and SplitLoRA use the fixed LoRA matrix $\mathbf{A}$ as the projection basis, meaning no additional forward computation is introduced during training. However, SplitLoRA requires significantly less projection-update overhead than InfLoRA, as reflected by the PU times above.

Table 12: Performance comparison of GPM-based experiments.

Method	20-Split CIFAR-100		10-Split CIFAR-100		20-Split MiniImageNet	
	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑	ACC(%)↑	BWT(%)↑
FS-DGPM	80.50	-3.30	74.33	-3.00	-	-
GPM	77.34	0.01	71.81	-0.11	63.90	-1.30
TRGP	81.68	-0.13	75.01	-0.01	62.68	-1.04
SGP	80.21	-0.88	74.97	-0.98	66.99	-2.46
GPCNS	78.63	-1.93	71.84	-3.44	62.85	-1.90
SplitLoRA	81.93	-1.06	75.03	-0.50	66.17	0.20

Table 13: Time costs on ImageNet-R 5 tasks.

Method	Task1 (T&E)	Task1 PU	Task2 (T&E)	Task2 PU	Task3 (T&E)	Task3 PU	Task4 (T&E)	Task4 PU	Task5 (T&E)	PU Total
LoRA	18.89	–	20.61	–	18.23	–	20.61	–	18.94	–
InfLoRA	18.61	1.18	20.43	1.28	18.26	1.07	20.61	1.08	18.58	5.69
SplitLoRA	18.91	0.68	19.88	0.72	18.35	0.44	20.70	0.38	18.24	2.60

A.11 More results on language models.

Following existing CL methods (Razdaibiedina et al., 2023), we evaluate different methods on SuperNI (Wang et al., 2022a) and Long Sequence (Razdaibiedina et al., 2023) benchmarks. SuperNI benchmark includes various types of NLP tasks, including dialogue generation, information extraction, question answering, summarization, and sentiment analysis. Three tasks are selected from each type, resulting in 15 tasks. These tasks are arranged into two different task sequences with different orders, referred to as Order 1 and Order 2.

Table 14: Details of different tasks in SuperNI Benchmark.

Dataset name	Task Type	Metric
Task639_multi_woz_user_utterance_generation	summarization	Rouge-L
Task1590_diplomacy_text_generation	summarization	Rouge-L
Task1729_personachat_generate_next	summarization	Rouge-L
Task181_outcome_extraction	information extraction	Rouge-L
Task748_glucose_reverse_cause_event_detection	information extraction	Rouge-L
Task1510_evalution_relation_extraction	information extraction	Rouge-L
Task002_quoref_answer_generation	dialogue generation	Rouge-L
Task073_commonsenseqa_answer_generation	dialogue generation	Rouge-L
Task591_sciq_answer_generation	dialogue generation	Rouge-L
Task511_reddit_tifu_long_text_summarization	question answering	Rouge-L
Task1290_xsum_summarization	question answering	Rouge-L
Task1572_samsum_summary	question answering	Rouge-L
Task363_sst2_polarity_classification	sentiment analysis	Accuracy
Task875_emotion_classification	sentiment analysis	Accuracy
Task1687_sentiment140_classification	sentiment analysis	Accuracy

We compare our method with state-of-the-art CL methods, including IncLoRA (Wang et al., 2023c), C-LoRA (Smith et al., 2024b), O-LoRA (Wang et al., 2023c), InfLoRA (Liang & Li, 2024), and GainLoRA (Liang et al.). Additionally, we introduce a simple baseline called SeqLoRA, which does not expand new LoRA branches but sequentially updates old LoRA parameters for new tasks and lacks mechanism to mitigate forgetting.

Evaluation Metric. We use $\mathrm{A}_{j,i}$ to denote the model's performance on the i-th task once the model learns the j-th task. Specifically, $\mathrm{A}_{j,i}$ represents accuracy for classification tasks and Rouge-L for other types of tasks. Following traditional CL works (Chaudhry et al., 2020), we employ average

Table 15: The order of different task sequences for experiments.

Benchmark	Order	Task Sequence
SuperNI Benchmark	1	task1572 → task363 → task1290 → task181 → task002 → task1510 → task639 → task1729 → task073 → task1590 → task748 → task511 → task591 → task1687 → task875
	2	task748 → task073 → task1590 → task639 → task1572 → task1687 → task591 → task363 → task1510 → task1729 → task181 → task511 → task002 → task1290 → task875

performance (AP) and forgetting (FT) to evaluate the model's performance. The formulas for these two metrics are defined as

$$\mathrm{AP} = \frac{1}{T}\sum_{i=1}^{T} \mathrm{A}_{T,i}, \quad \mathrm{FT} = \frac{1}{T-1}\sum_{i=1}^{T-1} (\max_{l\in\{1,2,\ldots,T-1\}} \mathrm{A}_{l,i} - \mathrm{A}_{T,i}), \tag{22}$$

where T denotes the total number of tasks in the task sequence. AP evaluates the model's final performance, and FT quantifies the forgetting.

Table 16: T5-Large Results.

Method	Order 1 AP↑	Order 1 FT↓	Order 2 AP↑	Order 2 FT↓
SeqLoRA	7.30	47.60	7.03	47.97
IncLoRA	12.33	41.93	16.65	36.56
C-LoRA	22.69	24.25	32.81	11.60
O-LoRA	26.37	19.15	32.83	11.99
Gain-OLoRA	47.84	2.26	46.84	2.91
InfLoRA	39.78	7.64	39.57	8.93
Gain-InfLoRA	46.21	2.40	46.44	2.61
SplitLoRA	**51.64**	1.14	**52.89**	0.73

Table 17: LLaMA2-7B Results.

Method	Order 1 AP↑	Order 1 FT↓	Order 2 AP↑	Order 2 FT↓
O-LoRA	39.37	15.84	37.55	20.23
Gain-OLoRA	51.10	4.96	51.14	5.57
InfLoRA	42.93	11.23	39.94	15.00
Gain-InfLoRA	51.27	2.84	50.17	4.71
SplitLoRA	**53.18**	3.01	**52.43**	3.72

These results demonstrate that the core idea of SplitLoRA continues to hold even for large language models, further validating the **generality and scalability** of our framework.

Algorithm 2 Projection update between tasks for SplitLoRA

Input: datasets $\{\mathcal{D}_\tau\}_{\tau=1}^{t}$, current weights $\mathbf{W}_t$, previous average gradient $\mathbf{G}_t^{\text{old}}$ (Eq. (3))
Output: updated $\mathbf{G}_{t+1}^{\text{old}}$, minor–subspace bases $\{\hat{\mathbf{U}}_{t+1}^{k}\}$, and LoRA bases $\{\mathbf{A}_{t+1}\}$ for task $t{+}1$
// Step 1: Update average gradient of old tasks (GPM-style, using LoRA representations)
Initialize per-layer accumulators $\mathbf{G}_t^{\text{new},(l)} \leftarrow \mathbf{0}$, $c^{(l)} \leftarrow 0$ for all layers l
for each mini-batch $B \subset \mathcal{D}_t$ **do**
    Forward pass through $f_{\mathbf{W}_t}$ to obtain LoRA inputs $x^{(l)}$ for all LoRA modules
    for each layer l with a LoRA module **do**
        Accumulate: $\mathbf{G}_t^{\text{new},(l)} \leftarrow \mathbf{G}_t^{\text{new},(l)} + x^{(l)}x^{(l)\top}$
        $c^{(l)} \leftarrow c^{(l)} + 1$
    end for
end for
for each layer l **do**
    $\mathbf{G}_t^{\text{new},(l)} \leftarrow \mathbf{G}_t^{\text{new},(l)}/c^{(l)}$
end for
Update the running average of old-task gradients using Eq. (3):

$$\mathbf{G}_{t+1}^{\text{old}} = \frac{1}{t}\big((t{-}1)\mathbf{G}_t^{\text{old}} + \mathbf{G}_t^{\text{new}}\big)$$

// Step 2: Compute minor subspace and optimal dimension k_{t+1}
for each layer l with a LoRA module **do**
    Perform SVD on $\mathbf{G}_{t+1}^{\text{old}}$:

$$\hat{\mathbf{U}}_{t+1}^{(l)}, \hat{\mathbf{\Sigma}}_{t+1}^{(l)}, \hat{\mathbf{V}}_{t+1}^{(l)\top} = \text{SVD}(\mathbf{G}_{t+1}^{\text{old},(l)})$$

    For each candidate k, compute $\epsilon_{t+1}^{(l)}(k)$ and the objective in Eq. (15)
    Select $k_{t+1}^{(l)} = \arg\min_k\big((t)\,\epsilon_{t+1}^{(l)}(k) - \alpha\frac{k}{d}\big)$
    Form the minor-subspace basis:

$$\hat{\mathbf{U}}_{t+1}^{k,(l)} = \hat{\mathbf{U}}_{t+1}^{(l)}[:, -k_{t+1}^{(l)}:]$$

end for
// Step 3: Initialize LoRA projection bases $\mathbf{A}_{t+1}$ **in the minor subspace**
for each layer l with a LoRA module **do**
    Sample a random Gaussian matrix $\mathbf{R}^{(l)} \in \mathbb{R}^{k_{t+1}^{(l)} \times r}$
    Construct $\mathbf{A}_{t+1}^{(l)}$ using Eq. (17):

$$\mathbf{A}_{t+1}^{(l)} = \hat{\mathbf{U}}_{t+1}^{k,(l)}\mathbf{R}^{(l)}$$

end for
return $\mathbf{G}_{t+1}^{\text{old}}$, $\{\hat{\mathbf{U}}_{t+1}^{k,(l)}\}$, $\{\mathbf{A}_{t+1}^{(l)}\}$