C-LoRA: Continual Low-Rank Adaptation for Pre-trained Visual Models

Xin Zhang
Institute of Intelligent Information Processing
Shanxi University
Taiyuan 030006, China
zhangxin8572@sxu.edu.cn

Liang Bai*
Institute of Intelligent Information Processing
Shanxi University
Taiyuan 030006, China
bailiang@sxu.edu.cn

Xian Yang
Alliance Manchester Business School
The University of Manchester
Manchester M13 9PL, United Kingdom
xian.yang@manchester.ac.uk

Abstract

Pre-trained visual models have become fundamental in computer vision, but they face challenges in continual learning scenarios where data and tasks evolve over time. Low-Rank Adaptation (LoRA) offers efficient fine-tuning capabilities but remains limited for such dynamic environments. Standard LoRA cannot distinguish important subspaces, causing critical knowledge to be overwritten in sequential training. Existing approaches address this by dynamically expanding the set of LoRA adapters—either maintaining a growing pool of task-specific modules or merging new adapters into prior ones—at the cost of unbounded parameter growth or increasing inference complexity. We propose Continual Low-Rank Adaptation (C-LoRA), a method that enables a single, shared LoRA adapter to handle sequential tasks without catastrophic forgetting—without requiring any module selection or fusion at inference. The core of C-LoRA is a learnable routing matrix $\boldsymbol{\mathcal{R}}$ that explicitly controls how each rank-one subspace contributes to the weight update. This matrix is decomposed into a stability component ($\boldsymbol{\mathcal{R}}_{\text{base}}$), which preserves knowledge from prior tasks, and a plasticity component ($\boldsymbol{\mathcal{R}}_{\delta}$), which drives adaptation to the current task—providing direct control over the stability-plasticity trade-off. We analyze how $\boldsymbol{\mathcal{R}}$ governs gradient flow during sequential training, and demonstrate competitive performance across multiple benchmarks. The code is available at https://github.com/lambor9973/C-LoRA.

1 Introduction

Pre-trained visual models Dosovitskiy [2020], Wang et al. [2022a,b] have revolutionized computer vision applications, but they face fundamental limitations when deployed in dynamic environments where tasks and data distributions continuously evolve Parisi et al. [2019]. This challenge, known as continual learning Wang et al. [2024], requires models to progressively accumulate knowledge while preserving performance on previously learned tasks.

Low-Rank Adaptation (LoRA) Hu et al. [2021] is a parameter-efficient fine-tuning method that adapts large-scale pre-trained models to downstream tasks by introducing low-rank matrices to update a

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

small subset of parameters while keeping the core pre-trained weights frozen. This approach preserves the general knowledge captured during pre-training, reduces computational and storage costs, and mitigates the risk of overfitting by limiting updates to task-specific components Ayoub Kamal et al. [2023]. LoRA has proven highly effective in fields such as computer vision, where pre-trained models are commonly deployed for diverse and evolving applications Yu et al. [2024], Zhou et al. [2024], Aljundi et al. [2017], Hu et al. [2023].

An overlooked question is why applying single LoRA to a task sequence leads to rapid forgetting, even when the adapter has sufficient capacity. We identify the cause: the standard update $\Delta\mathbf{W} = \mathbf{AB}$ implicitly assigns a uniform weight to every rank-one subspace $\mathbf{a}_i\mathbf{b}_i^\top$, providing no mechanism to distinguish subspaces that encode critical past knowledge from those that are underutilized. As a result, all subspaces are equally exposed to gradient updates from new tasks, regardless of their importance. Introducing a learnable routing matrix $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{r\times r}$ into the update, giving $\Delta\mathbf{W} = \mathbf{A}\boldsymbol{\mathcal{R}}\mathbf{B}$, addresses this by allowing the model to learn differential subspace weights. However, Theorem 1 shows that the gradients flowing into the shared bases $\mathbf{A}$ and $\mathbf{B}$ scale with $\|\boldsymbol{\mathcal{R}}\|_2$. As knowledge accumulates, $\|\boldsymbol{\mathcal{R}}\|_2$ grows, and the subspaces with the largest routing weights—those most critical for past tasks—experience the strongest gradient perturbations during new-task training. The subspaces most critical for stability are paradoxically the most vulnerable to disruption.

Prior approaches treats this conflict as a capacity problem: methods such as Zhou et al. [2024], Liang and Li [2024], Wu et al. [2025] expand the adapter pool with each new task—either retaining task-specific modules separately or merging them incrementally—at the cost of linear parameter growth and additional inference overhead from module selection or fusion. The implicit assumption is that stability and plasticity cannot be jointly satisfied within a fixed low-rank space. We show the conflict stems not from low-rank capacity, but from coupled gradient amplification that disproportionately disrupts critical subspaces. Resolving that mechanism directly makes expansion unnecessary.

We propose Continual Low-Rank Adaptation (C-LoRA), which mitigates the stability-plasticity conflict within a single, fixed-size adapter. C-LoRA decomposes $\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta$: the stability component $\boldsymbol{\mathcal{R}}_{\text{base}}$ accumulates knowledge from prior tasks and is excluded from gradient flow during new-task training; the plasticity component $\boldsymbol{\mathcal{R}}_\delta$ drives adaptation to the current task without disrupting consolidated knowledge. This decoupling breaks the conflict identified with theoretical guarantees and validated empirically across multiple benchmarks.

2 Related Work

Continual learning requires models to accumulate knowledge across a sequence of tasks without forgetting Wang et al. [2024]. Existing methods fall into four broad categories. Replay-based methods Liu et al. [2021], Luo et al. [2023], Wu et al. [2019], Buzzega et al. [2020], Wang et al. [2022c, 2021], Rebuffi et al. [2017] store a subset of past data and interleave it with new-task training. Knowledge distillation methods Dhar et al. [2019], Douillard et al. [2020], Li and Hoiem [2017], Simon et al. [2021], Tao et al. [2020] preserve the outputs of prior models via auxiliary losses. Architecture-based methods Shi et al. [2022], Zhao et al. [2020], Hu et al. [2023], Wang et al. [2023], Yang et al. [2022] allocate task-specific parameters within an expanding network. Regularization-based methods Zhou et al. [2023a], Aljundi et al. [2018], Zenke et al. [2017], Dhar et al. [2019], Li and Hoiem [2017] penalize changes to weights deemed important for past tasks.

LoRA Hu et al. [2021] achieves parameter-efficient adaptation by inserting trainable low-rank matrices into frozen pre-trained weights. It performs well in static settings Hayou et al. [2024], Li et al. [2023], Kopiczko et al. [2023], Fan et al. [2025], but applying a single LoRA adapter to a task sequence leads to rapid forgetting. The dominant response has been to expand the adapter set: InfLoRA Liang and Li [2024] and EASE Zhou et al. [2024] maintain a growing pool of task-specific modules; SD-LoRA Wu et al. [2025] introduces a new module per task and merges it into the previous one. These strategies treat the underlying conflict as a capacity problem—implicitly assuming that stability and plasticity cannot be jointly satisfied within a fixed low-rank space. Prior work has shown more broadly that parameter isolation strategies limit backward knowledge transfer by treating new-task updates as interference rather than as a source of refinement Lin et al. [2022]. The resulting systems also incur linear parameter growth and additional inference overhead from module selection or fusion.

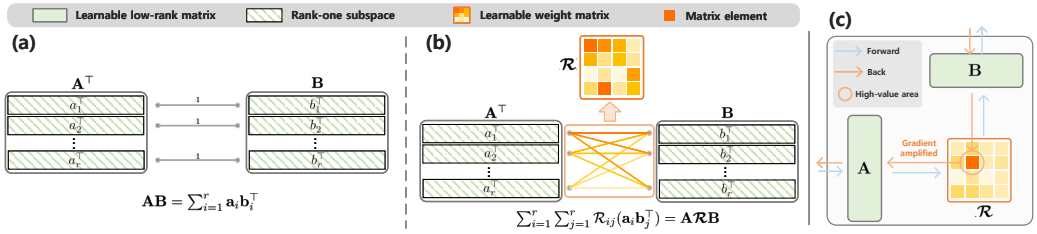

Figure 1: (a) The standard LoRA formulation implicitly assumes fixed, uniform weights (implicitly 1) and only allows interaction between corresponding rank-one subspaces ($a_i b_i^\top$). (b) C-LoRA introduces a learnable routing matrix $\boldsymbol{\mathcal{R}}$, enabling fine-grained control over subspace contributions and capturing complex cross-subspace interactions between a_i and b_j. (c) The inherent conflict in continual learning: High-magnitude areas in $\boldsymbol{\mathcal{R}}$ (hotspots) are crucial for stability (preserving past knowledge), but they simultaneously amplify the gradients flowing to the shared matrices $\mathbf{A}$ and $\mathbf{B}$ during backpropagation, thereby risking catastrophic forgetting.

Rather than expanding parameters, C-LoRA identifies and resolves it through a decomposed routing matrix within a single, fixed-size adapter. This shifts the problem from parameter allocation to gradient flow control.

3 Methodology

3.1 Preliminaries

For a pre-trained linear layer $\mathbf{W_0} \in \mathbb{R}^{d\times k}$, LoRA applies a low-rank update $\Delta\mathbf{W} = \mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times k}$ are trainable matrices with rank $r \ll \min(d, k)$. The updated weight for task t is:

$$\mathbf{W}_t = \mathbf{W}_0 + \Delta\mathbf{W}_t = \mathbf{W}_0 + \mathbf{A}_t\mathbf{B}_t, \tag{1}$$

where $\mathbf{A}_t$ and $\mathbf{B}_t$ are the task-specific low-rank matrices. By keeping $\mathbf{W}_0$ frozen and only updating $\mathbf{A}_t$ and $\mathbf{B}_t$, LoRA achieves significant parameter efficiency compared to full fine-tuning.

In continual learning, existing LoRA-based methods Wei et al. [2024], Zhou et al. [2024], Liang and Li [2024] introduce a separate update $\Delta\mathbf{W}_t = \mathbf{A}_t\mathbf{B}_t$ per task, maintaining a growing collection:

$$\{(\mathbf{A}_1, \mathbf{B}_1), (\mathbf{A}_2, \mathbf{B}_2), \ldots, (\mathbf{A}_T, \mathbf{B}_T)\}. \tag{2}$$

However, this approach treat the stability-plasticity conflict as a capacity problem rather than addressing its cause. We instead identify the cause within the structure of the update itself.

3.2 Generalized Low-Rank Adaptation via a Routing Matrix

The update $\Delta\mathbf{W} = \mathbf{AB}$ decomposes as a sum of r rank-one outer products:

$$\Delta\mathbf{W} = \sum_{i=1}^{r} \mathbf{a}_i \mathbf{b}_i^\top, \tag{3}$$

where $\mathbf{a}_i$ is the i-th column of $\mathbf{A}$ and $\mathbf{b}_i^\top$ is the i-th row of $\mathbf{B}$. As shown in Figure 1(a), each subspace $\mathbf{a}_i\mathbf{b}_i^\top$ contributes with a uniform, fixed weight of one. This leaves the adapter unable to differentially protect subspaces as knowledge accumulates.

Relaxing this constraint by introducing a learnable coefficient, $\mathcal{R}_{ij}$, for each potential interaction between a basis vector $\mathbf{a}_i$ and $\mathbf{b}_j$, as shown in Figure 1(b). This leads to a generalized formulation:

$$\Delta\mathbf{W} = \sum_{i=1}^{r}\sum_{j=1}^{r} \mathcal{R}_{ij}(\mathbf{a}_i \mathbf{b}_j^\top) = \mathbf{A}\boldsymbol{\mathcal{R}}\mathbf{B}, \tag{4}$$

where $\mathcal{R}_{ij}$ independently controls the contribution of each cross-subspace component $\mathbf{a}_i\mathbf{b}_j^\top$ (Figure 1(b)). Figure 2 confirms that $\boldsymbol{\mathcal{R}}$ develops structured, non-uniform weights during training: initially

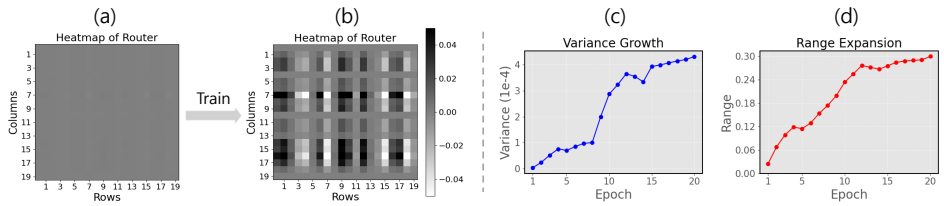

Figure 2: **(a):** Heatmap of the $\boldsymbol{\mathcal{R}}$ matrix at initialization, showing a nearly uniform distribution;**(b):** Heatmap of the $\boldsymbol{\mathcal{R}}$ matrix after training, with noticeable localised value changes; **(c):** Variance of matrix elements increasing with training epochs; **(d):** Range of matrix values expanding as training progresses.

near-uniform, the matrix develops localized high-magnitude regions and increasing element variance, indicating learned subspace prioritization.

3.3 Resolving the Stability-Plasticity Conflict via Decoupled Adaptation

Let θ_t represent the model parameters after learning task t. The metric for stability $S(\theta_t, \theta_{t+1})$ quantifies the degradation in performance on previous tasks due to the new adaptation:

$$S(\theta_t, \theta_{t+1}) = \mathbb{E}_{D_{old}}[\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)]. \tag{5}$$

Minimizing S is crucial for mitigating catastrophic forgetting. Our analysis relies on standard assumptions in optimization theory: the loss function $\mathcal{L}$ is L-smooth and the parameters are bounded (e.g., $\|\mathbf{A}\|_F, \|\mathbf{B}\|_F \leq M$).

The amplification problem. The gradients for the shared bases under $\mathbf{A}\boldsymbol{\mathcal{R}}\mathbf{B}$ are:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{A}} &= \boldsymbol{G}(\boldsymbol{\mathcal{R}}\mathbf{B})^\top = \boldsymbol{G}\mathbf{B}^\top\boldsymbol{\mathcal{R}}^\top, \\ \frac{\partial \mathcal{L}}{\partial \mathbf{B}} &= (\mathbf{A}\boldsymbol{\mathcal{R}})^\top\boldsymbol{G} = \boldsymbol{\mathcal{R}}^\top\mathbf{A}^\top\boldsymbol{G}, \end{aligned} \tag{6}$$

where $\boldsymbol{G} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ is the upstream gradient. A large $|\mathcal{R}_{ij}|$ therefore amplifies the gradients flowing into $\mathbf{A}$ and $\mathbf{B}$, as formalized below.

Theorem 1 *Assume the norms of the shared matrices and the upstream gradient are bounded, i.e., $\|\mathbf{A}\|_F, \|\mathbf{B}\|_F \leq M$ and $\|\boldsymbol{G}\|_F \leq G_{max}$. In a joint update scenario, the magnitude of the gradients flowing to the shared matrices $\mathbf{A}$ and $\mathbf{B}$ is directly bounded by the norm of the routing matrix $\boldsymbol{\mathcal{R}}$. Specifically,*

$$\left\|\frac{\partial \mathcal{L}}{\partial \mathbf{A}}\right\|_F \leq G_{max}M\|\boldsymbol{\mathcal{R}}\|_2, \quad \left\|\frac{\partial \mathcal{L}}{\partial \mathbf{B}}\right\|_F \leq G_{max}M\|\boldsymbol{\mathcal{R}}\|_2. \tag{7}$$

As tasks accumulate, $\|\boldsymbol{\mathcal{R}}\|_2$ grows, directly increasing the gradient magnitude on $\mathbf{A}$ and $\mathbf{B}$. The subspaces most important for stability—those with the largest $|\mathcal{R}_{ij}|$—are simultaneously the most susceptible to disruption. Figure 3 (top row) visualizes this: without intervention, high-magnitude regions in $\boldsymbol{\mathcal{R}}$ act as hotspots for destructive updates in subsequent tasks.

Decoupled routing. The amplification arises because $\boldsymbol{\mathcal{R}}$ simultaneously carries consolidated knowledge and modulates gradient flow. We decouple these roles by decomposing:

$$\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta \tag{8}$$

$\boldsymbol{\mathcal{R}}_{\text{base}}$ accumulates knowledge from all prior tasks and is treated as a constant during new-task training—gradient flow through it is stopped. $\boldsymbol{\mathcal{R}}_\delta$ is the trainable component for the current task. The update becomes:

$$\Delta\mathbf{W}' = \phi(\mathbf{A}\boldsymbol{\mathcal{R}}_{\text{base}}\mathbf{B}) + \mathbf{A}\boldsymbol{\mathcal{R}}_\delta\mathbf{B}, \tag{9}$$

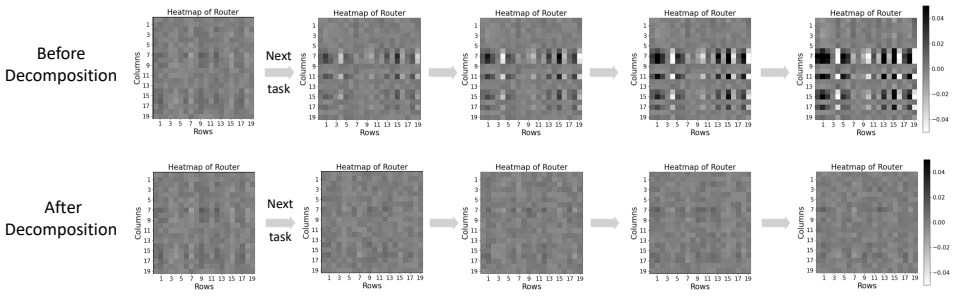

Figure 3: Evolution of the routing matrix $\boldsymbol{\mathcal{R}}$ heatmap during 5-task class-incremental learning. **Top Row (Before Decomposition):** Without decomposition, $\boldsymbol{\mathcal{R}}$ values progressively intensify in specific local regions. This highlights that subspaces critical for past tasks continue to receive strong updates when learning new tasks. **Bottom Row (After Decomposition):** With decomposition, updates primarily occur in the task-specific $\boldsymbol{\mathcal{R}}_\delta$, resulting in changes to $\boldsymbol{\mathcal{R}}$ that are more distributed across the matrix rather than being highly localized.

where $\phi(\cdot)$ denotes the stop-gradient operation. Gradients into $\mathbf{A}$ and $\mathbf{B}$ are now modulated by $\boldsymbol{\mathcal{R}}_\delta$ alone, whose norm is small at the start of each task. This directly breaks the amplification mechanism in Theorem 1. Figure 3 (bottom row) confirms that updates become distributed rather than concentrated at hotspots. After training, knowledge is consolidated: $\boldsymbol{\mathcal{R}}_{\text{base}} \leftarrow \boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta$.

Theorem 2 *Assume the loss function $\mathcal{L}$ is L-smooth. Let S_{Joint} and S_{Decomp} be the stability losses after one training step with learning rate η under the joint and decomposition strategies, respectively. If $\boldsymbol{\mathcal{R}}_\delta$ is initialized such that $\|\boldsymbol{\mathcal{R}}_\delta\|_2 \ll \|\boldsymbol{\mathcal{R}}_{base}\|_2$, the decomposition strategy achieves a significantly tighter upper bound on the stability loss.*

Beyond the magnitude of updates, the variability of these updates is also crucial for stability.As $\boldsymbol{\mathcal{R}}_{\text{base}}$ specializes across tasks, its elements develop high variance (Figure 2), whereas $\boldsymbol{\mathcal{R}}_\delta$ begins each task with low variance. To analyze the impact of this difference on the gradients, the following theorem assumes for simplicity that the elements of the residual matrices are independent and identically distributed (i.i.d.) with zero mean.

Theorem 3 (Reduced Gradient Variance via Decomposition) *Let the elements of $\boldsymbol{\mathcal{R}}_{base}$ and $\boldsymbol{\mathcal{R}}_\delta$ be modeled as independent and identically distributed (i.i.d.) random variables with zero mean and variances σ^2_{base} and σ^2_δ, respectively. Let $\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})$ denote the gradient w.r.t $\mathbf{A}$ modulated by $\boldsymbol{\mathcal{R}}$. If $\sigma^2_{base} > \sigma^2_\delta$, then the total variance of the gradient updates modulated by $\boldsymbol{\mathcal{R}}_\delta$ is strictly smaller than if they were modulated by $\boldsymbol{\mathcal{R}}_{base}$. Specifically,*

$$\mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}}_\delta)\|_F^2] < \mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}}_{base})\|_F^2], \tag{10}$$

and similarly for $\nabla_{\mathbf{B}}$.

Theorems 2 and 3 show that routing through $\boldsymbol{\mathcal{R}}_\delta$ produces updates that are both smaller in magnitude and lower in variance—two complementary properties for stability.

Fine-grained subspace gating. The decomposition mitigates gradient magnitude but treats all r^2 subspaces uniformly. Subspaces with large $|\mathcal{R}_{\text{base},ij}|$ are already heavily used and warrant additional protection. We introduce an importance-aware gate $\boldsymbol{\Gamma} \in [0,1]^{r\times r}$ that suppresses gradient flow into $\boldsymbol{\mathcal{R}}_\delta$ at dimensions important to prior tasks, while allowing free updates elsewhere:

$$\boldsymbol{\mathcal{R}}'_\delta = \boldsymbol{\Gamma} \odot \boldsymbol{\mathcal{R}}_\delta + (\mathbf{1} - \boldsymbol{\Gamma}) \odot \phi(\boldsymbol{\mathcal{R}}_\delta). \tag{11}$$

In the forward pass, $\boldsymbol{\mathcal{R}}'_\delta = \boldsymbol{\mathcal{R}}_\delta$ numerically. However, the stop-gradient operation $\phi(\cdot)$ ensures that during backward propagation, the effective gradient flowing to the parameters of $\boldsymbol{\mathcal{R}}_\delta$ is scaled by $\boldsymbol{\Gamma}$. When $\Gamma_{ij} \approx 0$ (important subspace), the gradient is suppressed, thereby protecting the consolidated knowledge by preventing adaptation in that dimension. When $\Gamma_{ij} \approx 1$ (low importance), the gradient flows freely.

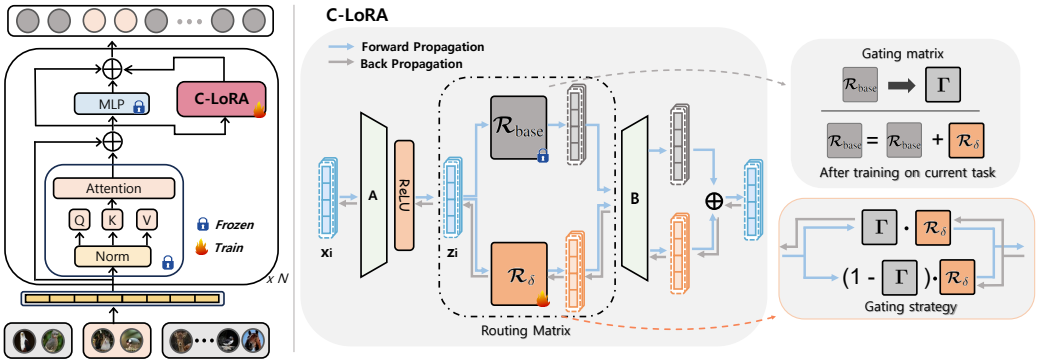

Figure 4: Illustration of the Proposed Model. **Left:** Vision Transformer (ViT) integrated with the C-LoRA module, where the adapter and local classifier are incrementally trained in each session. **Right:** Proposed architecture mitigates catastrophic forgetting by decoupling $\boldsymbol{\mathcal{R}}$. This isolates updates within a task-specific component, $\boldsymbol{\mathcal{R}}_\delta$, which in turn drives distributed updates to discover new routing configurations while preserving previously learned knowledge.

3.4 Implementation: C-LoRA in ViT

Building on our C-LoRA approach, this section details its application to Vision Transformers (ViT) Dosovitskiy [2020] for efficient continual learning. In a standard MLP block, the input features $\mathbf{x}_i \in \mathbb{R}^d$ are transformed as:

$$\text{MLP}(\mathbf{x}_i) = \sigma(\mathbf{x}_i \mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{12}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_2 \in \mathbb{R}^{h\times d}$ are the weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ are biases, and $\sigma(\cdot)$ is a non-linear activation function, such as ReLU. Directly updating these matrices for new tasks leads to significant parameter overhead and risks overwriting previously learned knowledge.

C-LoRA addresses these challenges by replacing full-rank updates with a low-rank adaptation mechanism. The modified MLP block operates by first projecting the input $\mathbf{x}_i$ into a low-rank subspace using a down-sampling matrix $\mathbf{A} \in \mathbb{R}^{d\times r}$, where $r \ll d$. The intermediate representation is computed as:

$$\boldsymbol{z}_i = \mathbf{x}_i \mathbf{A}, \tag{13}$$

where $\mathbf{A}$ reduces the dimensionality of the input while preserving its essential features.

Gating implementation. We implement the importance-aware gating matrix $\boldsymbol{\Gamma}$ to protect critical subspaces identified in $\boldsymbol{\mathcal{R}}_{\text{base}}$. The importance of each subspace is estimated by the magnitude of the corresponding element, $\mathbf{I}_{ij} = |\mathcal{R}_{\text{base},ij}|$. After the first task, importance scores are Z-score normalized:

$$Z_{ij} = \frac{\mathbf{I}_{ij} - \mu_I}{\sigma_I}, \tag{14}$$

where μ_I and σ_I are the mean and standard deviation of the importance matrix $\mathbf{I}$, respectively.

The gating matrix $\boldsymbol{\Gamma}$ is then computed using an element-wise sigmoid function:

$$\boldsymbol{\Gamma} = \text{Sigmoid}(-\beta \cdot \mathbf{Z}). \tag{15}$$

Here, $\beta > 0$ controls the sharpness of the gate. For the first task, $\boldsymbol{\Gamma}$ is initialized as a matrix of ones.

We employ the decoupled gating strategy in the update process. The effective routing component for the current task, $\boldsymbol{\mathcal{R}}'_\delta$, is calculated as:

$$\boldsymbol{\mathcal{R}}'_\delta = \boldsymbol{\Gamma} \odot \boldsymbol{\mathcal{R}}_\delta + (\mathbf{1} - \boldsymbol{\Gamma}) \odot \phi(\boldsymbol{\mathcal{R}}_\delta), \tag{16}$$

where $\phi(\cdot)$ denotes the stop-gradient operator.

The representation $\boldsymbol{z}_i$ is modulated before being up-sampled by $\mathbf{B} \in \mathbb{R}^{r\times d}$. The combined C-LoRA output is expressed as:

$$\text{C-LoRA}(\boldsymbol{x}_i) = \underbrace{\phi\big(\boldsymbol{z}_i \boldsymbol{\mathcal{R}}_{\text{base}} \mathbf{B}\big)}_{\text{stop-gradient for base}} + \underbrace{\big(\boldsymbol{z}_i \boldsymbol{\mathcal{R}}'_\delta \mathbf{B}\big)}_{\text{gated update}}. \tag{17}$$

This mechanism ensures that shared knowledge is preserved while allowing flexibility for task-specific updates. After training on the task is complete, the learned knowledge is consolidated: $\boldsymbol{\mathcal{R}}_{\text{base}} \leftarrow \boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}'_{\delta}$.

Integration into ViT: As shown in Figure 4, C-LoRA modifies the MLP block in ViT by combining its residual connection, original MLP computation, and the C-LoRA updates. The final output is computed as:

$$\boldsymbol{out} \;=\; \boldsymbol{x}_i + \text{MLP}(\boldsymbol{x}_i) \;+\; \text{C-LoRA}(\boldsymbol{x}_i). \tag{18}$$

This formulation maintains the original representational power of the MLP block while incorporating task-specific adaptability through C-LoRA. Over successive tasks, the accumulated updates in $\boldsymbol{\mathcal{R}}_{\delta}$ enable the model to adapt to new tasks without compromising performance on previously learned ones.

Loss function. Each task is trained with a cosine-similarity cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N_t}\sum_{j=1}^{N_t} \log \frac{\exp(z_{j,c^*})}{\sum_{\ell \in \mathcal{Y}(t)} \exp(z_{j,\ell})}, \tag{19}$$

where N_t is the total number of samples in the current task t's dataset, and $\mathcal{Y}(t)$ represents the set of class labels for the current task. The term $z_{j,\ell} = \frac{s(\mathbf{w}_\ell^\top \mathbf{f}_j)}{\|\mathbf{w}_\ell\|\|\mathbf{f}_j\|}$ represents the scaled cosine similarity between the feature vector $\mathbf{f}_j$ of sample j and the weight vector $\mathbf{w}_\ell$ of class ℓ, with s being a scaling factor to stabilize optimization.

In class-incremental learning, we typically cannot access raw data from previous tasks. To overcome this, our method re-samples synthetic features, not raw images, from a frozen feature extractor after the initial training is complete. We do this by using the mean and covariance of each class's features Zhang et al. [2023a] to generate representative feature vectors. This allows us to align the classification layer to the new class distribution while keeping the backbone frozen. This approach ensures that the model retains its discriminative power on old tasks without requiring direct access to their data.

4 Experiments

4.1 Experimental Setup

Datasets. Following previous studies McDonnell et al. [2024], Zhang et al. [2023a], Tan et al. [2024], we evaluate our method on four primary benchmark datasets. CIFAR-100 Krizhevsky et al. [2009] consists of 100 classes with 60,000 images, approximately 50,000 for training (83%) and 10,000 for testing (17%). ImageNet-A Hendrycks et al. [2021] includes 200 classes and roughly 7,500 test samples. CUB-200 Wah et al. [2011] contains 11,788 bird images across 200 subclasses, with about 6,000 for training (51%) and 5,800 for testing (49%). Car196 Krause et al. [2013] comprises 196 car models with 16,185 images, split into around 8,150 for training (50%) and 8,035 for testing (50%). For the continual learning setting, these datasets are divided into a sequence of tasks (incremental sessions). We evaluate performance under three scenarios: 5, 10, and 20 incremental sessions.

Implementation Details. Consistent with recent studies McDonnell et al. [2024], Zhang et al. [2023a], we use the ViT-B/16 model pre-trained on ImageNet-21K Ridnik et al. [2021] as the frozen backbone. During each incremental session, we train the C-LoRA module and the classifier using the SGD optimizer with a learning rate of 0.01, coupled with a cosine annealing scheduler starting at an initial value of 0.0005. The batch size is set to 48. The plasticity component of the routing matrix, $\boldsymbol{\mathcal{R}}_{\delta}$, is initialized near zero at the beginning of each new session. Baseline methods using LoRA adapters Zhou et al. [2024], Liang and Li [2024] used their recommended ranks, and C-LoRA adopts a fixed rank of $r = 20$ throughout all experiments.

Evaluation Metrics. We use two standard metrics to assess continual learning performance:

- **Last-Acc (%):** The average classification accuracy across all classes seen so far, measured after the final incremental session.
- **Inc-Acc (%):** The average of the Last-Acc values measured at the end of each incremental session. This metric provides a holistic view of the model's performance throughout the entire learning process.

All experiments are repeated with three different random seeds, and we report the mean and standard deviation of the results. "Joint-Training" represents the upper bound performance.

4.2 Comparison with State-of-the-Art Methods

Table 1 and Figure 5 present the primary comparison under the 10 session setting, which balances task granularity and sequence length. C-LoRA shows competitive results on all four datasets. The advantage is particularly pronounced on fine-grained benchmarks: on CUB-200, C-LoRA achieves 89.93% Last-Acc, substantially exceeding Adam+adapter (87.04%) and SLCA (84.55%), demonstrating that our stability-plasticity mechanism scales effectively to longer task sequences. Full results under the 5 and 20 session settings are provided in Appendix.

Table 1: Accuracy comparison of different continual learning methods across four benchmark datasets, divided into 10 incremental sessions.

Method	**Split CIFAR-100**		**Split ImageNet-A**		**Split CUB-200**		**Split CAR196**	
	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)
Joint-Training	93.22	-	79.61	-	90.78	-	78.66	-
EWC Kirkpatrick et al. [2017]	90.59 ± 0.63	93.80 ± 0.97	56.24 ± 1.56	66.30 ± 1.71	85.10 ± 0.31	92.36 ± 0.51	45.58 ± 0.19	60.41 ± 0.94
L2P Wang et al. [2022b]	84.17 ± 0.91	88.20 ± 1.30	43.47 ± 0.89	50.52 ± 2.29	67.13 ± 2.03	79.62 ± 1.56	45.81 ± 0.98	57.85 ± 1.92
DualPrompt Wang et al. [2022a]	81.70 ± 1.07	86.42 ± 0.98	46.41 ± 0.20	55.95 ± 1.34	68.68 ± 0.35	80.75 ± 0.87	39.36 ± 2.32	53.93 ± 2.40
SLCA Zhang et al. [2023a]	91.07 ± 0.45	94.06 ± 1.07	58.96 ± 0.88	66.39 ± 2.16	84.55 ± 0.19	90.69 ± 0.65	66.49 ± 2.87	76.43 ± 0.61
Adam+VPT-S Zhou et al. [2023b]	68.67 ± 12.40	75.22 ± 9.72	39.08 ± 9.51	49.08 ± 9.11	80.03 ± 11.42	86.60 ± 9.14	31.63 ± 10.15	41.78 ± 10.92
Adam+VPT-D Zhou et al. [2023b]	84.61 ± 3.01	88.83 ± 2.70	37.00 ± 13.81	46.29 ± 14.94	86.73 ± 0.33	91.91 ± 0.15	38.04 ± 12.80	48.00 ± 14.05
Adam+adapter Zhou et al. [2023b]	87.32 ± 0.28	91.21 ± 1.35	48.94 ± 0.10	58.95 ± 1.38	87.04 ± 0.13	92.13 ± 0.27	37.65 ± 0.09	49.22 ± 1.66
Adam+ssf Zhou et al. [2023b]	85.26 ± 0.40	89.92 ± 0.84	49.60 ± 0.80	58.90 ± 2.22	86.32 ± 0.43	91.80 ± 0.55	42.83 ± 0.72	53.97 ± 1.94
EASE Zhou et al. [2024]	87.53 ± 0.38	91.67 ± 1.04	60.74 ± 0.93	69.43 ± 1.31	81.55 ± 2.38	88.63 ± 1.84	44.07 ± 0.95	57.88 ± 1.12
InfLoRA Liang and Li [2024]	84.31 ± 0.28	90.31 ± 0.28	40.42 ± 1.15	52.20 ± 1.51	69.58 ± 0.78	81.57 ± 0.43	51.45 ± 3.36	68.05 ± 0.62
SD-LoRA Wu et al. [2025]	86.65 ± 1.54	90.78 ± 1.25	44.11 ± 9.81	53.93 ± 9.59	71.33 ± 1.21	82.69 ± 0.53	52.47 ± 1.72	65.16 ± 2.72
C-LoRA (Ours)	**91.70 ± 0.14**	**94.24 ± 0.85**	**63.86 ± 0.84**	**71.18 ± 2.10**	**89.93 ± 0.49**	**93.48 ± 0.11**	**70.73 ± 1.68**	**78.86 ± 1.32**

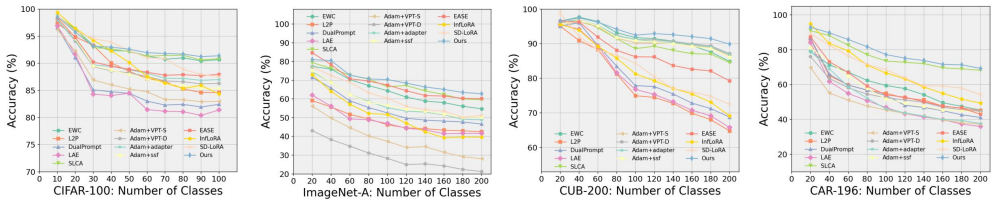

Figure 5: Accuracy performance during the 10 incremental sessions on CIFAR-100, ImageNet-A, CUB-200, and CAR196.

A key advantage of C-LoRA is its *fixed* parameter budget. As shown in Table 2, LoRA-expansion methods incur parameter costs that grow linearly with the number of tasks (e.g., EASE requires 23.6M total expanded parameters over 20 tasks), whereas C-LoRA maintains a constant overhead parameter regardless of sequence length. This fixed budget also eliminates the need for module selection or fusion at inference time, thereby simplifying deployment.

Table 2: Comparison of parameter efficiency for different LoRA-based continual learning methods under a 20-task setting. C-LoRA maintains a fixed parameter budget, while others grow linearly with the number of tasks.

Method	**Total Expanded Parameters (M)**	**Trainable Parameters per Task (M)**
EASE	23.6	1.18
InfLoRA	7.4	0.37
SD-LoRA	7.4	0.37
C-LoRA (Ours)	**0.36**	**0.36**

4.3 Ablation Studies

To ensure controlled comparisons across ablation variants, all ablation experiments are conducted with a single fixed random seed. Table 3 ablates the two core components of C-LoRA on ImageNet-A and CUB-200 (10 sessions): the decomposition of $\mathcal{R}$ into $\mathcal{R}_{\text{base}}$ and $\mathcal{R}_{\delta}$, and the importance-aware

gating $\mathbf{\Gamma}$. We additionally report Backward Transfer (BWT), where higher values indicate less forgetting. Removing both components causes severe forgetting (BWT: -28.90 on ImageNet-A, -11.78 on CUB-200), confirming that unconstrained routing updates overwrite prior knowledge. The decomposition alone substantially recovers performance, as freezing $\boldsymbol{\mathcal{R}}_{\text{base}}$ shields prior subspaces from gradient interference. Adding gating further reduces forgetting by providing fine-grained, per-subspace control, and the progressive benefit of each component is shown in Figure 6.

Table 3: Ablation study on the core components of C-LoRA (10 sessions).

Method	**ImageNet-A**			**CUB-200**		
	Last-Acc (%)	Inc-Acc (%)	BWT	Last-Acc (%)	Inc-Acc (%)	BWT
w/o Decomp & Gate	55.76	67.15	-28.90	85.67	90.61	-11.78
w/o Gate	62.21	71.09	-13.33	89.74	93.36	-4.47
C-LoRA (Ours)	**64.06**	**73.38**	$\mathbf{-9.77}$	**90.25**	**93.63**	$\mathbf{-3.78}$

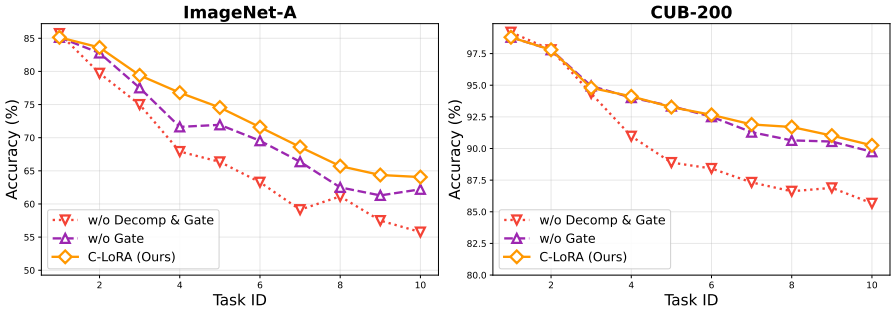

Figure 6: Per-task accuracy on ImageNet-A (left) and CUB-200 (right).

To isolate the contribution of the C-LoRA module from that of the feature resampling strategy (Stage 2), Table 4 compares C-LoRA against EASE, InfLoRA, and SD-LoRA under the 10 session setting on CIFAR-100 and ImageNet-R Boschini et al. [2022]. The variant without Stage 2 ("Ours w/o stage 2") already surpasses all LoRA-expansion baselines on both datasets, achieving 88.49% on CIFAR-100 compared to 87.53% for EASE. This demonstrates that the performance gains of C-LoRA stem primarily from the design of the routing matrix rather than from bias correction alone. Incorporating Stage 2 yields additional improvements (91.70% on CIFAR-100; 77.94% on ImageNet-R), confirming that feature resampling further reduces the recency bias introduced by class-incremental training.

Table 4: Comparison with LoRA-expansion methods and ablation on the feature resampling strategy (Stage 2) on CIFAR-100 and ImageNet-R (10 sessions).

Method	**Split CIFAR-100**		**Split ImageNet-R**	
	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)
LoRA-Expansion Methods				
EASE	87.53 ± 0.38	91.67 ± 1.04	74.52 ± 0.93	80.22 ± 1.31
InfLoRA	84.31 ± 0.28	90.31 ± 0.28	74.75 ± 0.64	80.67 ± 0.55
SD-LoRA	86.65 ± 1.54	90.78 ± 1.25	75.72 ± 1.15	80.39 ± 0.82
C-LoRA (Ours)				
Ours w/o stage 2	88.49 ± 0.24	92.19 ± 1.24	76.79 ± 0.21	82.22 ± 0.59
Ours (Full)	$\mathbf{91.70} \pm \mathbf{0.14}$	$\mathbf{94.24} \pm \mathbf{0.85}$	$\mathbf{77.94} \pm 0.41$	$\mathbf{82.99} \pm 0.37$

5 Conclusion

In this work, we presented Continual Low-Rank Adaptation (C-LoRA), a novel framework that addresses catastrophic forgetting in pre-trained visual models during sequential task adaptation. By employing a learnable routing matrix $\boldsymbol{\mathcal{R}}$, strategically decomposed to manage shared and task-specific

knowledge, C-LoRA significantly enhances knowledge retention. Our extensive empirical evaluations on diverse benchmarks demonstrated C-LoRA's competitive performance.

6 Acknowledgement

This work is supported by the National Natural Science Foundation of China (No.62432006, 62276159) and the Fundamental Research Program of Shanxi Province (No. 202303021223004).

References

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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 *European Conference on Computer Vision*, pages 631–648. Springer, 2022a.

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*, pages 139–149, 2022b.

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.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Muhammad Ayoub Kamal, Muhammad Mansoor Alam, Aznida Abu Bakar Sajak, and Mazliham Mohd Su'ud. Requirements, deployments, and challenges of lora technology: A survey. *Computational Intelligence and Neuroscience*, 2023(1):5183062, 2023.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23219–23230, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23554–23564, 2024.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.

Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11858–11867, 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*, pages 23638–23647, 2024.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. *arXiv preprint arXiv:2501.13198*, 2025.

Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems*, 34:3478–3490, 2021.

Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11371–11380, 2023.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing Hong, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. *arXiv preprint arXiv:2202.06592*, 2022c.

Liyuan Wang, Bo Lei, Qian Li, Hang Su, Jun Zhu, and Yi Zhong. Triple-memory networks: A brain-inspired method for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):1925–1934, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XX 16*, pages 86–102. Springer, 2020.

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

Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1591–1600, 2021.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 254–270. Springer, 2020.

Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16722–16731, 2022.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217, 2020.

Liyuan Wang, Xingxing Zhang, Qian Li, Mingtian Zhang, Hang Su, Jun Zhu, and Yi Zhong. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, 5(12):1356–1368, 2023.

Binbin Yang, Xinchi Deng, Han Shi, Changlin Li, Gengwei Zhang, Hang Xu, Shen Zhao, Liang Lin, and Xiaodan Liang. Continual object detection via prototypical task correlation guided gating mechanism. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9255–9264, 2022.

Qinhao Zhou, Xiang Xiang, and Jing Ma. Hierarchical task-incremental learning with feature-space initialization inspired by neural collapse. *Neural Processing Letters*, 55(8):10811–10827, 2023a.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

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

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

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

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023.

Chenghao Fan, Zhenyi Lu, Sichen Liu, Xiaoye Qu, Wei Wei, Chengfeng Gu, and Yu Cheng. Make lora great again: Boosting lora with adaptive singular values and mixture-of-experts optimization alignment. *arXiv preprint arXiv:2502.16894*, 2025.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Beyond not-forgetting: Continual learning with backward knowledge transfer. *Advances in Neural Information Processing Systems*, 35: 16165–16177, 2022.

Xiwen Wei, Guihong Li, and Radu Marculescu. Online-lora: Task-free online continual learning via low rank adaptation. *arXiv preprint arXiv:2411.05663*, 2024.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158, 2023a.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuwen Tan, Qinhao Zhou, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. Semantically-shifted incremental adapter-tuning is a continual vitransformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23252–23262, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

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*, pages 15262–15271, 2021.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

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. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Da-Wei Zhou, Zi-Wen Cai, 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*, 2023b.

Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *European conference on computer vision*, pages 692–709. Springer, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 30534–30544. IEEE, 2025.

A Mathematical Proofs

This appendix provides the detailed proofs for theorems presented in the main text.

A.1 Proof of Theorem 1

Proof of Theorem 1. We aim to establish an upper bound on the Frobenius norm of the gradient $\nabla_{\mathbf{A}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}}$. The gradient is given by $\nabla_{\mathbf{A}}\mathcal{L} = \boldsymbol{G}\mathbf{B}^\top\boldsymbol{\mathcal{R}}^\top$. We invoke the property for the Frobenius norm, which states that for conformable matrices X and Y, $\|XY\|_F \leq \|X\|_F\|Y\|_2$, where $\|\cdot\|_2$ denotes the spectral norm.

$$\begin{aligned}\|\nabla_{\mathbf{A}}\mathcal{L}\|_F &= \|\boldsymbol{G}(\boldsymbol{\mathcal{R}}\mathbf{B})^\top\|_F \\ &\leq \|\boldsymbol{G}\|_F\|\boldsymbol{\mathcal{R}}\mathbf{B}\|_2.\end{aligned} \tag{20}$$

By the assumption that $\|\boldsymbol{G}\|_F \leq G_{\text{max}}$, and utilizing the sub-multiplicativity property of the spectral norm ($\|\boldsymbol{\mathcal{R}}\mathbf{B}\|_2 \leq \|\boldsymbol{\mathcal{R}}\|_2\|\mathbf{B}\|_2$), we obtain:

$$\|\nabla_{\mathbf{A}}\mathcal{L}\|_F \leq G_{\text{max}}\|\boldsymbol{\mathcal{R}}\|_2\|\mathbf{B}\|_2. \tag{21}$$

Since the spectral norm is bounded by the Frobenius norm, $\|\mathbf{B}\|_2 \leq \|\mathbf{B}\|_F$. By the hypothesis that $\|\mathbf{B}\|_F \leq M$, we conclude:

$$\|\nabla_{\mathbf{A}}\mathcal{L}\|_F \leq G_{\text{max}}M\|\boldsymbol{\mathcal{R}}\|_2. \tag{22}$$

The bound for $\nabla_{\mathbf{B}}\mathcal{L}$ is derived symmetrically by considering $\|\boldsymbol{\mathcal{R}}^\top\mathbf{A}^\top\boldsymbol{G}\|_F$, utilizing the assumption $\|\mathbf{A}\|_F \leq M$, and the fact that $\|\boldsymbol{\mathcal{R}}^\top\|_2 = \|\boldsymbol{\mathcal{R}}\|_2$. □

A.2 Proof of Theorem 2

Proof of Theorem 2. Let $\mathcal{L}_{old}(\theta) = \mathbb{E}_{D_{old}}[\mathcal{L}(\theta)]$. The stability loss is $S(\theta_t, \theta_{t+1}) = \mathcal{L}_{old}(\theta_{t+1}) - \mathcal{L}_{old}(\theta_t)$. Since $\mathcal{L}_{old}$ is L-smooth, we can bound the stability loss using the Descent Lemma (derived from the second-order Taylor expansion):

$$S(\theta_t, \theta_{t+1}) \leq \langle \nabla\mathcal{L}_{old}(\theta_t), \Delta\theta\rangle + \frac{L}{2}\|\Delta\theta\|_F^2, \tag{23}$$

where $\Delta\theta = \theta_{t+1} - \theta_t$ is the parameter update resulting from a single gradient descent step on the new task loss $\mathcal{L}_{new}$, i.e., $\Delta\theta = -\eta\nabla_\theta\mathcal{L}_{new}$.

We compare the magnitude of the updates to the shared parameters $\mathbf{A}$ and $\mathbf{B}$.

Case 1: Joint Strategy. The update is modulated by $\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta$. Applying Theorem 1:

$$\|\Delta\mathbf{A}_{Joint}\|_F = \eta\|\nabla_{\mathbf{A}}\mathcal{L}_{new}\|_F \leq \eta G_{max}M\|\boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta\|_2. \tag{24}$$

Case 2: Decomposition Strategy. The gradient flow through $\boldsymbol{\mathcal{R}}_{\text{base}}$ is stopped, so the update is modulated only by $\boldsymbol{\mathcal{R}}_\delta$:

$$\|\Delta\mathbf{A}_{Decomp}\|_F \leq \eta G_{max}M\|\boldsymbol{\mathcal{R}}_\delta\|_2. \tag{25}$$

By the hypothesis that $\|\boldsymbol{\mathcal{R}}_\delta\|_2 \ll \|\boldsymbol{\mathcal{R}}_{\text{base}}\|_2$, it follows immediately that $\|\Delta\mathbf{A}_{Decomp}\|_F$ is significantly smaller than $\|\Delta\mathbf{A}_{Joint}\|_F$. The same holds for $\Delta\mathbf{B}$.

Consequently, the total parameter shift $\|\Delta\theta_{Decomp}\|_F$ is significantly smaller than $\|\Delta\theta_{Joint}\|_F$. Substituting this into the L-smoothness bound, both the first-order term (bounded via Cauchy-Schwarz by $\|\nabla\mathcal{L}_{old}(\theta_t)\|_F\|\Delta\theta\|_F$) and the second-order term ($\frac{L}{2}\|\Delta\theta\|_F^2$) are substantially tighter for the decomposition strategy, establishing a significantly improved upper bound on S_{Decomp}. □

A.3 Proof of Theorem 3

Proof of Theorem 3. We analyze the variability of the gradients by examining their total variance under the defined stochastic model. Since the entries of $\boldsymbol{\mathcal{R}}$ are assumed to have zero mean, the expected gradient $\mathbb{E}[\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})]$ is the zero matrix. The total variance (the sum of the variances of all elements) is therefore equivalent to the expected squared Frobenius norm, $\mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})\|_F^2]$.

Let $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{r\times r}$ be a random matrix with i.i.d. entries such that $\mathbb{E}[\mathcal{R}_{ij}] = 0$ and $\text{Var}(\mathcal{R}_{ij}) = \sigma^2$. The gradient is $\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}}) = \boldsymbol{G}\mathbf{B}^\top\boldsymbol{\mathcal{R}}^\top$. Let $\mathbf{M} = \boldsymbol{G}\mathbf{B}^\top$. We treat $\mathbf{M}$ as fixed.

The (i, j)-th element of the gradient is $[\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})]_{ij} = \sum_{k=1}^r M_{ik}\mathcal{R}_{jk}$. We compute its variance. Due to the independence of the entries of $\boldsymbol{\mathcal{R}}$:

$$\begin{aligned}\text{Var}([\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})]_{ij}) &= \text{Var}\left(\sum_{k=1}^r M_{ik}\mathcal{R}_{jk}\right)\\ &= \sum_{k=1}^r M_{ik}^2\text{Var}(\mathcal{R}_{jk})\\ &= \sigma^2\sum_{k=1}^r M_{ik}^2.\end{aligned} \tag{26}$$

The total variance is the sum over all entries (i, j). Let d be the row dimension of $\mathbf{A}$ (and $\mathbf{M}$).

$$\begin{aligned}\mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})\|_F^2] &= \sum_{i=1}^d\sum_{j=1}^r \text{Var}([\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}})]_{ij})\\ &= \sum_{i=1}^d\sum_{j=1}^r\left(\sigma^2\sum_{k=1}^r M_{ik}^2\right)\\ &= \sigma^2\sum_{j=1}^r\left(\sum_{i=1}^d\sum_{k=1}^r M_{ik}^2\right)\\ &= \sigma^2 r\|\mathbf{M}\|_F^2.\end{aligned} \tag{27}$$

By the hypothesis $\sigma_{\text{base}}^2 > \sigma_\delta^2$, and assuming $\|\mathbf{M}\|_F^2 > 0$ (non-zero upstream gradient), we have:

$$\begin{aligned}\mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}}_\delta)\|_F^2] &= \sigma_\delta^2 r\|\mathbf{M}\|_F^2\\ &< \sigma_{\text{base}}^2 r\|\mathbf{M}\|_F^2\\ &= \mathbb{E}[\|\nabla_{\mathbf{A}}(\boldsymbol{\mathcal{R}}_{\text{base}})\|_F^2].\end{aligned} \tag{28}$$

A symmetric argument holds for $\nabla_{\mathbf{B}}$ by analyzing the variance of $\boldsymbol{\mathcal{R}}^\top(\mathbf{A}^\top\boldsymbol{G})$. □

B Additional Experimental Results

This appendix provides the full set of experimental results omitted from the main text for brevity, including benchmark comparisons under 5 and 20 session settings.

Table 5 summarises performance when each dataset is divided into 5 sessions. C-LoRA achieves competitive results across nearly all benchmarks. On CIFAR-100, C-LoRA achieves a Last-Acc of

Table 5: Accuracy comparison of different continual learning methods across four benchmark datasets, divided into 5 sessions.

Method	Split CIFAR-100		Split ImageNet-A		Split CUB-200		Split CAR196	
	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)
Joint-Training	93.22	-	79.61	-	90.78	-	78.66	-
EWC Kirkpatrick et al. [2017]	91.21 ± 0.53	93.93 ± 0.67	60.13 ± 2.28	68.74 ± 0.87	84.73 ± 0.74	91.07 ± 0.38	51.64 ± 1.36	61.47 ± 1.10
L2P Wang et al. [2022b]	84.60 ± 1.74	89.24 ± 1.27	46.94 ± 0.71	53.28 ± 1.13	73.07 ± 1.74	81.99 ± 1.34	56.03 ± 2.86	66.43 ± 2.40
DualPrompt Wang et al. [2022a]	82.66 ± 2.00	87.23 ± 0.88	50.54 ± 0.98	58.18 ± 1.30	74.28 ± 0.26	83.60 ± 0.44	48.90 ± 0.76	61.83 ± 0.44
SLCA Zhang et al. [2023a]	92.19 ± 0.56	**94.56 ± 0.58**	63.14 ± 1.39	68.65 ± 0.72	86.54 ± 1.27	91.51 ± 0.96	75.73 ± 1.27	82.01 ± 0.72
Adam+VPT-S Zhou et al. [2023b]	84.10 ± 0.88	87.89 ± 1.49	24.49 ± 2.35	31.16 ± 1.73	86.64 ± 0.17	91.25 ± 0.14	30.86 ± 11.39	39.34 ± 13.77
Adam+VPT-D Zhou et al. [2023b]	85.89 ± 4.11	89.41 ± 3.03	48.76 ± 4.90	56.59 ± 5.78	86.67 ± 0.13	91.23 ± 0.10	23.52 ± 13.70	30.80 ± 17.23
Adam+adapter Zhou et al. [2023b]	88.63 ± 0.13	91.98 ± 0.85	49.79 ± 0.67	58.38 ± 1.33	87.25 ± 0.03	91.66 ± 0.23	40.75 ± 0.29	50.07 ± 1.81
Adam+ssf Zhou et al. [2023b]	86.98 ± 0.26	90.91 ± 0.98	52.36 ± 1.53	61.08 ± 2.60	86.73 ± 0.05	91.18 ± 0.19	46.07 ± 0.66	55.58 ± 1.90
EASE Zhou et al. [2024]	89.33 ± 0.16	92.53 ± 0.92	62.92 ± 0.83	70.19 ± 1.18	83.93 ± 1.48	89.55 ± 0.84	54.98 ± 0.80	67.09 ± 1.08
InfLoRA Liang and Li [2024]	87.38 ± 0.48	92.13 ± 0.08	48.87 ± 1.36	59.54 ± 0.56	77.32 ± 1.61	86.28 ± 0.79	66.27 ± 1.27	78.10 ± 0.48
SD-LoRA Wu et al. [2025]	88.52 ± 0.45	92.16 ± 1.16	54.86 ± 1.12	62.06 ± 2.88	77.52 ± 2.28	86.21 ± 1.27	66.70 ± 1.85	74.98 ± 1.63
C-LoRA (Ours)	**92.27 ± 0.25**	94.49 ± 0.64	**65.28 ± 0.66**	**71.52 ± 2.01**	**89.67 ± 0.60**	**93.15 ± 0.44**	**77.07 ± 0.67**	**83.10 ± 1.32**

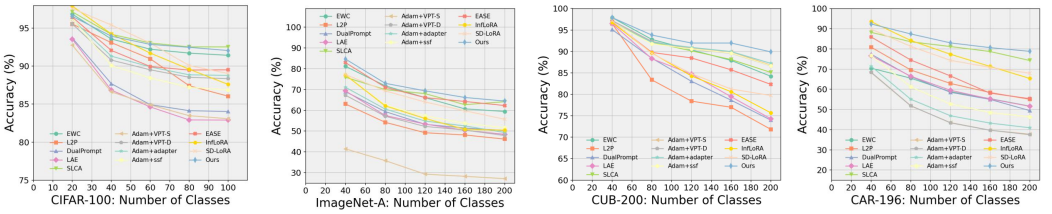

Figure 7: Accuracy performance during the 5 sessions on CIFAR-100, ImageNet-A, CUB-200, and CAR196.

Table 6: Accuracy comparison of different continual learning methods across four benchmark datasets, divided into 20 sessions. This setting emphasizes the challenge of longer task sequences.

Method	Split CIFAR-100		Split ImageNet-A		Split CUB-200		Split CAR196	
	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)
Joint-Training	93.22	-	79.61	-	90.78	-	78.66	-
EWC Kirkpatrick et al. [2017]	89.95 ± 0.15	93.31 ± 1.02	54.62 ± 1.39	63.55 ± 1.53	87.25 ± 1.00	93.45 ± 0.48	44.42 ± 0.56	61.75 ± 1.06
L2P Wang et al. [2022b]	79.32 ± 1.49	84.10 ± 1.11	40.07 ± 1.42	49.35 ± 1.37	60.30 ± 3.80	73.53 ± 3.34	29.83 ± 3.31	42.82 ± 1.14
DualPrompt Wang et al. [2022a]	76.57 ± 0.60	82.97 ± 1.78	41.48 ± 1.63	52.84 ± 1.57	65.22 ± 1.29	77.86 ± 1.31	25.11 ± 2.09	40.89 ± 0.77
SLCA Zhang et al. [2023a]	90.37 ± 0.56	**93.64 ± 0.99**	53.76 ± 5.54	61.82 ± 4.74	82.37 ± 0.42	90.12 ± 0.93	56.09 ± 3.89	68.50 ± 2.42
Adam+VPT-S Zhou et al. [2023b]	81.41 ± 1.00	86.48 ± 1.46	40.66 ± 5.37	51.32 ± 4.68	82.42 ± 5.75	89.46 ± 3.65	29.72 ± 8.31	41.08 ± 8.90
Adam+VPT-D Zhou et al. [2023b]	73.60 ± 15.86	79.59 ± 13.27	37.07 ± 15.26	48.41 ± 15.17	86.67 ± 0.56	92.09 ± 0.87	36.76 ± 6.24	49.34 ± 7.03
Adam+adapter Zhou et al. [2023b]	84.86 ± 0.29	89.41 ± 1.32	48.72 ± 0.07	59.53 ± 1.02	86.80 ± 0.02	92.42 ± 0.31	37.69 ± 0.02	50.56 ± 1.40
Adam+ssf Zhou et al. [2023b]	82.96 ± 1.21	87.90 ± 0.05	47.86 ± 4.13	58.58 ± 4.27	86.37 ± 0.20	92.21 ± 0.63	41.98 ± 0.42	54.35 ± 1.29
EASE Zhou et al. [2024]	84.88 ± 1.09	90.08 ± 1.01	55.50 ± 1.16	65.73 ± 1.77	78.26 ± 1.02	87.14 ± 1.19	31.83 ± 1.38	47.90 ± 1.89
InfLoRA Liang and Li [2024]	77.28 ± 1.19	85.82 ± 0.28	30.90 ± 1.16	43.12 ± 2.98	65.24 ± 2.71	77.89 ± 0.84	35.79 ± 1.56	55.10 ± 2.21
SD-LoRA Wu et al. [2025]	84.14 ± 1.39	88.73 ± 2.74	40.51 ± 2.78	51.33 ± 3.18	60.64 ± 1.51	74.50 ± 1.29	26.53 ± 10.48	42.02 ± 13.26
C-LoRA (Ours)	**90.70 ± 0.20**	93.30 ± 1.08	**59.40 ± 3.36**	**69.25 ± 2.31**	**90.03 ± 0.14**	**93.96 ± 0.41**	**62.45 ± 2.13**	**73.85 ± 1.28**

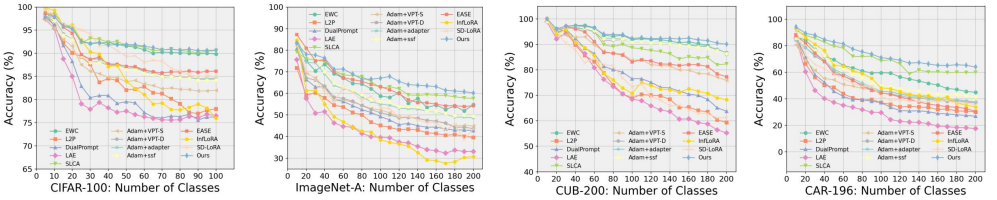

Figure 8: Accuracy performance during the 20 sessions on CIFAR-100, ImageNet-A, CUB-200, and CAR196.

92.27%, slightly ahead of the competitive SLCA (92.19%). On the more challenging ImageNet-A, CUB-200, and CAR196 datasets, C-LoRA consistently outperforms all baselines. Notably, on CUB-200, C-LoRA (89.67%) comes within 1.1 percentage points of the Joint-Training upper bound (90.78%). The per-session accuracy curves in Figure 7 further illustrate the stability of C-LoRA throughout the learning process. Table 6 and Figure 8 further report results under the more demanding 20 session setting. A noteworthy observation is that on CUB-200, C-LoRA's Last-Acc slightly increases with the number of sessions. We attribute this to the substantial overlap between CUB-200's

bird categories and the pre-training corpus, reducing the forgetting pressure that would otherwise accumulate over longer task sequences.

Table 7: CUB-200 per-task accuracy (%) after learning all tasks.

Method	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Avg**
w/o Decomp & Gate	63.2	75.4	83.9	85.8	86.2	90.3	88.4	93.2	93.7	94.8	85.5
w/o Gate	79.8	89.4	89.0	89.5	92.2	93.4	89.3	93.6	92.0	89.6	89.8
C-LoRA (Ours)	**83.8**	**90.8**	**90.4**	**89.1**	**92.2**	92.5	**90.2**	92.8	91.6	89.1	**90.3**

Table 8: CUB-200 per-task forgetting (%) after learning all tasks.

Method	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Avg**
w/o Decomp & Gate	36.0	21.7	11.5	10.9	10.4	4.8	5.3	4.0	1.3	0.0	10.6
w/o Gate	18.6	8.2	5.1	4.9	2.6	−0.4	0.5	−0.8	1.7	0.0	4.0
C-LoRA (Ours)	**14.6**	**6.8**	**4.1**	6.1	**2.2**	**−0.9**	**−1.3**	1.6	**0.8**	0.0	**3.4**

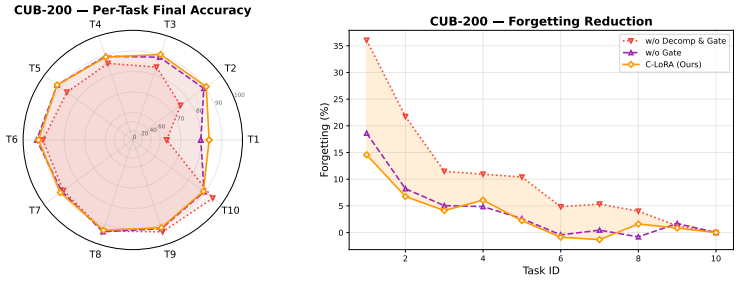

Figure 9: CUB-200: Per-Task Final Accuracy & Forgetting Reduction

Without decomposition or gating, early tasks suffer the most—T1 loses 36.0% accuracy as subsequent training freely overwrites shared routing directions. Routing decomposition halves the average forgetting from 10.6 to 4.0, and C-LoRA reduces it further to 3.4%. Tables 7 and 8 report per-task accuracy and forgetting on CUB-200 after all 10 sessions are learned.

To examine the relationship between $\|\boldsymbol{\mathcal{R}}\|_2$ and the gradients on the shared bases described in Theorem 1, we log $\|\partial\mathcal{L}/\partial\mathbf{A}\|_F$ and $\|\boldsymbol{\mathcal{R}}\|_2$ at every training step and report the 12-layer mean in Table 9 (10 sessions). Without decomposition, $\|\boldsymbol{\mathcal{R}}\|_2$ keeps growing across sessions and the gradient on $\mathbf{A}$ grows with it. With decoupled routing, $\|\boldsymbol{\mathcal{R}}\|_2$ stays close to its initial scale and the gradient on $\mathbf{A}$ is $24\times$ to $8836\times$ smaller.

Table 10 compares C-LoRA with three controlled variants under the same backbone, classifier head, Stage-2 resampling, and protocol as Table 3 (10 sessions). **Vanilla LoRA** uses the standard update $\Delta\mathbf{W} = \mathbf{AB}$ at the same rank. **w/o Decomp & Gate** uses a shared $\mathbf{A}\boldsymbol{\mathcal{R}}\mathbf{B}$ with a fully trainable $\boldsymbol{\mathcal{R}}$. **+ LR scaling** further multiplies the learning rates of $\mathbf{A}$ and $\mathbf{B}$ by κ_t^2 at each layer, where $\kappa_t = \|\boldsymbol{\mathcal{R}}_\delta\|_2/\|\boldsymbol{\mathcal{R}}_{\text{base}} + \boldsymbol{\mathcal{R}}_\delta\|_2$, matching the effective step size of decomposition without decomposing $\boldsymbol{\mathcal{R}}$.

Adding a shared routing matrix without decomposition improves over Vanilla LoRA by at most 1.91 points in Last-Acc, so the extra capacity of $\boldsymbol{\mathcal{R}}$ alone does not explain the gains. Learning-rate scaling recovers a large part of the improvement, and C-LoRA still achieves the best accuracy on all three datasets.

We additionally compare C-LoRA with AdaLoRA Zhang et al. [2023b], obtained by replacing the routing module of C-LoRA with the AdaLoRA parameterization under otherwise identical configurations, and with CL-LoRA He et al. [2025]. Results on ImageNet-A and CUB-200 (10 sessions) are reported in Table 11.

Table 9: Gradient norm $\|\partial\mathcal{L}/\partial\mathbf{A}\|_F$ and routing-matrix norm $\|\boldsymbol{\mathcal{R}}\|_2$ across sessions (12-layer mean, 10 sessions). "w/o Decomp" denotes the variant without decomposition and gating; Ratio = w/o Decomp / C-LoRA.

Session	$\|\partial\mathcal{L}/\partial\mathbf{A}\|_F$			$\|\boldsymbol{\mathcal{R}}\|_2$		
	C-LoRA	w/o Decomp	Ratio	C-LoRA	w/o Decomp	Ratio
ImageNet-A						
1	0.0013	0.0506	39×	1.09	1.56	1.43×
5	0.0005	0.1118	234×	1.17	2.65	2.26×
9	0.0007	0.1638	236×	1.24	3.25	2.63×
CUB-200						
1	3×10^{-6}	0.0019	659×	1.03	1.51	1.47×
5	1×10^{-6}	0.0072	4929×	1.03	2.52	2.44×
9	1×10^{-6}	0.0124	8836×	1.04	3.06	2.95×
Cars-196						
1	0.0006	0.0140	24×	1.12	1.53	1.37×
5	0.0005	0.0274	60×	1.31	2.49	1.90×
9	0.0003	0.0373	129×	1.50	3.05	2.03×

Table 10: Controlled comparison of capacity and learning-rate scaling (10 sessions). "–" denotes not evaluated.

Method	**ImageNet-A**		**CUB-200**		**Cars-196**	
	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc
Vanilla LoRA	53.85	–	85.33	–	69.38	–
w/o Decomp & Gate	55.76	67.15	85.67	90.61	69.69	79.19
+ LR scaling	60.11	69.85	89.99	–	70.44	78.98
C-LoRA (Ours)	**64.06**	**73.38**	**90.25**	**93.63**	**71.97**	**80.01**

Table 11: Comparison with AdaLoRA and CL-LoRA on ImageNet-A and CUB-200 (10 sessions).

Method	**ImageNet-A**		**CUB-200**	
	Last-Acc (%)	Inc-Acc (%)	Last-Acc (%)	Inc-Acc (%)
AdaLoRA	57.41	68.56	83.67	91.52
CL-LoRA	58.92	70.06	80.03	87.66
C-LoRA (Ours)	**64.06**	**73.38**	**90.25**	**93.63**

B.1 Computational Resources

All experiments were conducted on a GPU server running Ubuntu 20.04.6 LTS. The server was equipped with an Intel Core i9-14900k processor and used a single NVIDIA GeForce RTX 4090 GPU. The implementation was built using the PyTorch framework, version 2.3.1. For the most extensive scenario involving 20 incremental sessions, the average training time required per session on each dataset was approximately as follows: 92 minutes for CIFAR-100, 13 minutes for ImageNet-A, 52 minutes for CUB-200, and 124 minutes for CAR196. These times reflect the average duration to complete the training for one incremental task within the 20 session sequence.

B.2 Limitations and Future Work

The theoretical analysis in Theorems 1–3 relies on simplified assumptions, including bounded parameter norms and an i.i.d. zero-mean model for routing matrix elements. These assumptions do not hold exactly during training. Nevertheless, the consistent empirical gains across all benchmarks suggest that the core mechanism—decoupling gradient flow via $\boldsymbol{\mathcal{R}}_{\text{base}}$ and $\boldsymbol{\mathcal{R}}_{\delta}$—is robust beyond the idealized setting. Deriving tighter guarantees under more realistic assumptions is a natural direction for future work.

The experiments in this paper focus on class-incremental image classification with ViT backbones. The routing matrix design is architecture-agnostic in principle, but its effectiveness on other tasks—such as object detection or segmentation—and on other backbone families remains to be verified. Extending C-LoRA to these settings is a direct avenue for future investigation.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction include the claims made in the paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations concerning theoretical assumptions and empirical scope are discussed in Appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Assumptions are stated and acknowledged in Appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Methodology, implementation details including hyperparameters and setup, and computational resources are provided.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide the source code in the supplemental material.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Experimental setup, datasets, evaluation protocol, and hyperparameters are detailed in main paper. Compute details are in Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Results presented in Tables 6, 5, 1 report the mean and standard deviation over 3 independent runs.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide related information in appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research focuses on algorithmic development for continual learning using standard public datasets and does not present obvious ethical concerns.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Potential positive impacts and broader challenges are discussed at the end of Appendix.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: No data or models are released as part of this work, so safeguards for responsible release are not applicable.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the datasets and baseline papers in the experiment section we used.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or direct experiments with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: The research does not involve human subjects requiring IRB approval.

    Guidelines:

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

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were only used for writing, editing, and formatting purposes, and not as a core method in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.