CLASS-INCREMENTAL LEARNING WITH ZERO-SHOT UNLEARNING VIA ACTIVATION DISENTANGLEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We study class-incremental learning (CIL) with zero-shot unlearning, where previously learned task-specific knowledge must be removed without access to past data. Existing CIL methods reduce forgetting during sequential learning, but they do not ensure safe unlearning: deleting a task-specific component can still perturb later-task activations and degrade retained performance. Motivated by this activation-space entanglement challenge, we introduce Anti-Entanglement of Activation (AEA), which enforces inter-task orthogonality in representation space to enable unlearning by parameter deletion. Building on this idea, we propose CAD, a buffer-free class-incremental learning framework with dual-branch LoRA adapters that separate shared and removable knowledge, allowing zero-shot unlearning through simple adapter detachment rather than retraining. The framework further facilitates unlearning by selectively deleting forgotten-task basis vectors from gradient-projection memory. Across multiple benchmarks, CAD achieves stronger retention–removal trade-offs and lower membership-inference risk than existing baselines. Our code is available here.

1 INTRODUCTION

Deep neural networks (DNNs) are typically trained offline, yet in many applications models must be updated over time as new tasks arrive. In such settings, learning new tasks can overwrite previously acquired knowledge, leading to *catastrophic forgetting* McCloskey & Cohen (1989). This has motivated extensive research in *continual learning* (CL), which seeks to retain prior knowledge while continually acquiring new capabilities. At the same time, growing privacy and security concerns have increased interest in *machine unlearning* (MU) Golatkar et al. (2020a); Wu et al. (2020); Golatkar et al. (2020b), which aims to remove the influence of specific data from a trained model without full retraining from the scratch. This has motivated the emerging *Continual Learning and Unlearning* (CLU) framework, which studies models that must continually acquire new tasks while supporting on-demand removal of previously learned knowledge.

Under the *Continual Learning and Unlearning* (CLU) framework, illustrated in Fig. 1, a model must sequentially acquire new tasks while removing previously learned knowledge when requested. This setting introduces a fundamental tension: CL encourages models to preserve and transfer prior knowledge, whereas privacy-aware unlearning requires selected knowledge to be erased after it has already influenced the model. Moreover, many MU methods assume access to the forget set, retain set, or auxiliary buffer data during unlearning, which conflicts with the standard CL assumption that previous-task data are no longer available. Therefore, privacy-aware CLU requires mechanisms that can support targeted removal without revisiting past samples.

In this paper, we focus on CLU under two practical constraints. First, following the standard continual learning assumption, data from previous tasks are not retained, so unlearning must be performed without access to past samples. This rules out replay-based approaches and many conventional unlearning methods that rely on retraining with retained data. Second, the framework should not depend on task identifiers at test time for inference since, in practical deployments, incoming data rarely carries explicit task labels, and the model must operate directly on task-agnostic inputs. Existing works on continual learning with unlearning Liu et al. (2022); Shibata et al. (2021); Adhikari et al. (2025); Chatterjee et al. (2024); Özdenizci et al. (2025) generally assume known task IDs for inference during test time, which limits their applicability in this practical setting.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

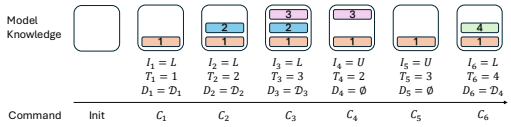


Figure 1: The CLU protocol. At each step t, the model receives a command $C_t = (I_t, T_t, D_t)$, with $I_t = L$ for learning and $I_t = U$ for unlearning. Colored blocks indicate retained task-specific knowledge.

Table 1: Comparison of continual learning and unlearning (CLU) methods. CAD is the framework supporting a fully generalized setup without strong assumptions.

CLU Method	Buffer-free	Task-ID-free-Test	Independent Unlearning	Arbitrary Learning
CLPU Liu et al. (2022)	✓	✗	✓	✗
LSF Shibata et al. (2021)	✓	✗	✗	✓
UniCLUN Chatterjee et al. (2024)	✗	✗	✓	✓
UnCLe Adhikari et al. (2025)	✓	✗	✓	✓
PALL Özdenizci et al. (2025)	✗	✗	✓	✓
CAD (Ours)	✓	✓	✓	✓

A central obstacle to zero-shot unlearning is that structural knowledge deletion is not guaranteed to be safe under existing continual learning methods. Namely, deleting one task's representation can still degrade retained tasks because task knowledge remains entangled in activation space. To address this challenge, we propose **CAD**, a **C**lass-Incremental Learning framework with Zero-Shot Unlearning via **A**ctivation **D**isentanglement. Our contributions are as follows:

- We identify activation-space entanglement as an overlooked obstacle to zero-shot unlearning: even with orthogonal weight updates, deleting a task-specific adapter can still perturb later-task activation paths and degrade retained performance. To address this, we introduce **Anti-Entanglement of Activation (AEA)**, which enforces inter-task separation in activation space and enables reliable structural deletion without disrupting retained tasks.
- We propose a **Dual-branch LoRA architecture** that captures shared representations in early task-agnostic adapters and isolates removable task-specific knowledge in deeper adapters, enabling efficient zero-shot unlearning with low storage overhead.
- We introduce a **Basis-Deletion Strategy** that selectively removes obsolete gradient-projection bases while preserving shared ones, improving memory efficiency and unlearning effectiveness.
- We validate the proposed framework on multiple benchmarks against existing continual learning and unlearning baselines.

2 Related Work

Continual Learning. To address catastrophic forgetting, numerous continual learning (CL) methods have been proposed. They can be broadly categorized into three groups. *Regularization-based methods* constrain weight updates to prevent significant changes to parameters deemed important for prior tasks. For example, EWC (Kirkpatrick et al., 2017) uses a Fisher information-weighted quadratic penalty, while LwF (Li & Hoiem, 2017) uses knowledge distillation to align current predictions with those of earlier models. These methods avoid storing data from previous tasks, but often provide limited protection against forgetting in challenging class-incremental settings. *Expansion-based methods* allocate task-specific parameters to isolate tasks. PNN (Rusu et al., 2016) expands the network by adding new columns per task while freezing earlier ones, whereas HAT (Serrà et al., 2018) learns binary masks to gate task-relevant neurons. While these methods reduce interference by separating task-specific computation, they typically require task identity at inference time to select the appropriate subnetwork. *Rehearsal-based methods* explicitly store and replay samples from previous tasks. ER (Rolnick et al., 2019) maintains a memory buffer and interleaves stored samples with new ones during training, while DER++ (Buzzega et al., 2020) and FDR (Benjamin et al., 2019) combine rehearsal with distillation to preserve output consistency. Although these methods improve retention, replay conflicts with data-free or privacy-sensitive settings. Overall, standard CL methods focus on preserving past knowledge but they are not designed to support targeted removal of learned tasks.

Machine Unlearning. Machine unlearning (MU) aims to remove specific knowledge from a trained neural network upon user request (Cao & Yang, 2015). The simplest solution, retraining from scratch without the data to be forgotten, is often impractical due to computational cost and limited access to the original dataset. Existing MU methods therefore seek to remove targeted information efficiently without full retraining, and can be grouped by the data available at unlearning time. Let the *forget set* denote the data to be erased and the *retain set* denote the data to be preserved. When both forget and retain sets are available, the forget set guides targeted removal while the retain set helps preserve performance on the remaining data. Representative methods include BadTeacher (Chundawat et al., 2023a), which distills from a randomly initialized teacher model; SalUn (Fan et al., 2024), which updates parameters most influenced by the forget set; and SSD (Foster et al., 2024), which uses Fisher information to identify parameters associated with the forget set. One-shot unlearning methods

require only the forget set; for example, JiT (Foster et al., 2025) minimizes gradient discrepancies between forget samples and their augmentations to suppress their influence. Zero-shot unlearning methods use neither forget nor retain data. GKT (Chundawat et al., 2023b) exemplifies this setting by generating pseudo-forget samples via a generative model and applying gated distillation. However, one-shot and zero-shot methods commonly struggle with instability and scalability, particularly for large-scale models or complex datasets.

Continual Learning with Unlearning Capability. Several recent works have explored unlearning in the context of continual learning. CLPU (Liu et al., 2022) maintains an auxiliary network per task and, upon user request, discards or distills the knowledge of the corresponding task into the main model. However, CLPU requires the user to specify at training time whether each task should ultimately be retained or removed, limiting flexibility and precluding *arbitrary task learning*. Moreover, CLPU does not support inference on tasks learned temporarily but not yet unlearned, restricting the model's ability to support inference across all learned tasks. UniCLUN (Chatterjee et al., 2024) adapts BadTeacher (Chundawat et al., 2023a) to the CL setting. For unlearning, a randomly initialized teacher injects noise associated with the forget set, while a stored model from the previous task serves as a good teacher for distilling knowledge from the retain set. However, this requires access to both forget and retain data in separate *buffer*, violating the standard continual learning assumption that prior-task data are unavailable. LSF (Shibata et al., 2021) introduces mnemonic codes that serve as task-specific anchors. Unlearning is triggered by deleting the associated code, which induces forgetting of the target task. Since unlearning relies mainly on forgetting rather than selective removal, this design does not support *independent unlearning*, limiting its flexibility. UnCLe (Adhikari et al., 2025) uses a hypernetwork (Ha et al., 2017) conditioned on task-specific embeddings to generate per-task parameters, and unlearning is performed by discarding the embedding of the target task. PALL (Özdenizci et al., 2025) learns task-specific sparse subnetworks and enables task-level unlearning by resetting the parameters trained on the target task, followed by rehearsal-based recovery for later tasks that reused those parameters. UnCLe and PALL are formulated for task-incremental learning, requiring task identifiers to select the appropriate embedding or subnetwork at inference. Furthermore, PALL also relies on *episodic memory buffers* during unlearning to prevent degradation of retained tasks. In summary, existing CLU methods often rely on restrictive assumptions, such as access to previous task data during unlearning or *availability of task IDs* for inference at test time. In contrast, CAD removes these constraints, enabling independent unlearning in a data-free and task-agnostic manner while supporting arbitrary task learning as shown in the Table. 1.

3 METHOD

3.1 PROBLEM FORMULATION

We consider a continual learner that receives a sequence of commands $\mathbb{C} = \{C_i\}_{i=1}^{|\mathbb{C}|}$, where each command $C_i = (I_i, T_i, D_i)$ is a triplet consisting of (1) a task instruction $I_i \in \{L, U\}$ indicating either a learning (L) or unlearning (U) operation, (2) a task index $T_i = t$, and (3) an associated dataset D_i. For a *learning command* ($I_i = L$), the model is trained on task t using dataset $D_i = \mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$. Under the standard assumption of privacy-preserving continual learning, data from previous tasks are discarded after training, i.e., no rehearsal buffer or replay is permitted. For an *unlearning command* ($I_i = U$), the model must selectively forget task t *without access* to its training data, i.e., $D_i = \emptyset$. This zero-shot unlearning is important in privacy-sensitive CL applications where data cannot be retained after use.

3.2 TASK-SPECIFIC LEARNING AND UNLEARNING

We formalize our class-incremental learning framework by building on the principle of orthogonal weight updates of InfLoRA (Liang & Li, 2024) based on GPM Saha et al. (2021). Let the model after training on the j-th task ($j < t$) be denoted by $\mathbf{W}_j = \mathbf{W}_0 + \sum_{i=1}^{j} \Delta\mathbf{W}_i$, where $\mathbf{W}_0 \in \mathbb{R}^{d_o \times d_i}$ is the initial pretrained backbone and $\Delta\mathbf{W}_i \in \mathbb{R}^{d_o \times d_i}$ represents the incremental weight update for task i. Let $\mathcal{D}_j$ denote the dataset used to train task j, and let $\mathbf{R}_j \in \mathbb{R}^{d_i \times n}$ and $\mathbf{H}_j \in \mathbb{R}^{d_o \times n}$ denote its corresponding input representations and output activations, where n denotes the length of token sequence. Then,

$$\mathbf{H}_j = (\mathbf{W}_0 + \sum_{i=1}^{j} \Delta\mathbf{W}_i)\mathbf{R}_j.$$

By design of GPM Saha et al. (2021), the adapter update for a new task t, denoted by $\Delta\mathbf{W}_t$, is constrained to be orthogonal to the subspace spanned by representations from previous tasks. Let

$$\mathrm{col}(\mathbf{M}_t) = \mathrm{span}\left(\bigcup_{i=1}^{t-1} \mathrm{col}(\mathbf{R}_i)\right)$$

denote this prior-task representation subspace, where $\mathbf{M}_t \in \mathbb{R}^{d_i \times m(t)}$ denotes the GPM matrix. The orthogonality constraint is $\Delta\mathbf{W}_t\mathbf{M}_t = \mathbf{0}$. This constraint prevents the update for task t from changing activations along the stored directions of previous tasks. After training on task t, the full model becomes $\mathbf{W}_t = \mathbf{W}_0 + \sum_{i=1}^{t} \Delta\mathbf{W}_i$, and evaluation on any earlier task $j < t$ satisfies

$$\left(\mathbf{W}_0 + \sum_{i=1}^{t} \Delta\mathbf{W}_i\right)\mathbf{R}_j = \left(\mathbf{W}_0 + \sum_{i=1}^{j} \Delta\mathbf{W}_i\right)\mathbf{R}_j = \mathbf{H}_j, \tag{1}$$

provided that subsequent updates satisfy

$$\Delta\mathbf{W}_i\mathbf{R}_j = \mathbf{0}, \quad \forall i \in \{j+1, \ldots, t\}. \tag{2}$$

This enables new tasks to be learned while preserving earlier-task activations without forgetting. However, as we demonstrate, it is insufficient for task-specific unlearning: although orthogonal updates preserve **earlier** tasks, they do not ensure that the contribution of a specific task can be removed without affecting the block activation of **later** tasks. To address this limitation, CAD introduces architectural and algorithmic mechanisms for task-specific knowledge removal.

Central to this design is a dual-branch LoRA architecture that separates shared and task-specific components. Within a Vision Transformer (ViT) comprising N transformer blocks, the first L layers are equipped with *task-agnostic* LoRA adapters, which capture transferable knowledge across tasks. The remaining $N - L$ layers contain *task-exclusive* adapters, each specific to an individual task. This separation supports unlearning by design: forgetting a task reduces to detaching its corresponding task-exclusive adapters while leaving shared representations intact.

Task-Agnostic LoRA Branch. Given a learning command $C_i = (I_i = L,\ T_i = t,\ D_i = \mathcal{D}_t)$, the weight update for the task-agnostic LoRA branch is

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \Delta\mathbf{W}_t = \mathbf{W}_{t-1} + \mathbf{A}_t\mathbf{B}_t, \tag{3}$$

where $\mathbf{A}_t$ and $\mathbf{B}_t$ denote the low-rank matrices of the LoRA adapter fine-tuned on $\mathcal{D}_t$. After training, the adapter contribution $\Delta\mathbf{W}_t$ is merged into the base weights $\mathbf{W}_{t-1}$, and the adapter is re-initialized for future tasks. Thus, across the continual learning process, only one effective task-agnostic LoRA block is maintained, enabling parameter-efficient knowledge sharing.

Task-Exclusive LoRA Branch. The task-exclusive LoRA branch handles task-specific knowledge by maintaining a separate adapter for each task throughout the learning process. Given the same command $C_i = (I_i = L,\ T_i = t,\ D_i = \mathcal{D}_t)$, the model's weights evolve as

$$\mathbf{W}_t = \mathbf{W}_0 + \sum_{i=1}^{t} \Delta\mathbf{W}_i = \mathbf{W}_0 + \sum_{i=1}^{t} \mathbf{A}_i\mathbf{B}_i. \tag{4}$$

Upon receiving an unlearning command $C_k = (I_k = U,\ T_k = j,\ D_k = \emptyset)$, the task-exclusive component associated with task j is removed without access to its original data. Specifically, unlearning is performed by simply detaching the j-th LoRA adapter, resulting in

$$\mathbf{W}_t = \mathbf{W}_0 + \sum_{i \neq j} \Delta\mathbf{W}_i = \mathbf{W}_0 + \sum_{i \neq j} \mathbf{A}_i\mathbf{B}_i. \tag{5}$$

In contrast to traditional MU and CLU methods that require auxiliary optimization with extra buffer data, CAD supports data-free unlearning through highly lightweight pruning.

3.3 Anti-Entanglement of Activation

Removing an earlier adapter $\Delta\mathbf{W}_j = \mathbf{A}_j\mathbf{B}_j$ can degrade later tasks even when subsequent updates preserve earlier-task activations. We examine this asymmetry through projected adapter activations, define a decision-level criterion for evaluating their utility, and use it to motivate AEA.

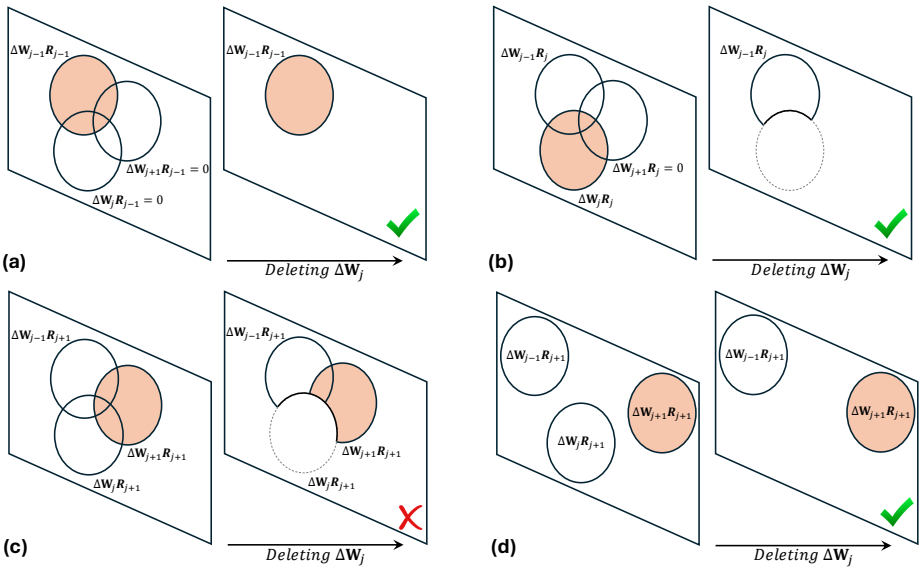


Figure 2: Effect of adapter deletion on activation subspaces. (a)–(c) Naive LoRA adapter deletion can degrade subsequent tasks by perturbing shared activation subspaces. (d) Anti-Entanglement of Activation encourages orthogonality across task activations, reducing downstream degradation after deletion.

After training through task t, consider three adjacent tasks $\mathcal{D}_{j-1}$, $\mathcal{D}_j$, and $\mathcal{D}_{j+1}$, with $j+1 < t$. At a fixed layer, under the orthogonality condition in Eq. 2, their activations satisfy

$$\mathbf{H}_{j-1} = (\mathbf{W}_0 + \Delta\mathbf{W}_1 + \cdots + \Delta\mathbf{W}_{j-1})\mathbf{R}_{j-1}, \tag{6}$$

$$\mathbf{H}_j = (\mathbf{W}_0 + \Delta\mathbf{W}_1 + \cdots + \Delta\mathbf{W}_j)\mathbf{R}_j, \tag{7}$$

$$\mathbf{H}_{j+1} = (\mathbf{W}_0 + \Delta\mathbf{W}_1 + \cdots + \Delta\mathbf{W}_{j+1})\mathbf{R}_{j+1}. \tag{8}$$

Thus, each task's activation depends on its own adapter and earlier adapters.

Definition 3.1 (Activation Subspace and Projector)**.** *For task i, let $\mathcal{S}_i := \mathrm{col}(\Delta\mathbf{W}_i\mathbf{R}_i) \subset \mathbb{R}^{d_o}$ be the subspace spanned by its adapter outputs, and let $\mathbf{P}_i$ denote the orthogonal projector onto $\mathcal{S}_i$.*

If $\mathbf{Q}_i$ is an orthonormal basis for $\mathcal{S}_i$, then $\mathbf{P}_i = \mathbf{Q}_i\mathbf{Q}_i^\top = (\Delta\mathbf{W}_i\mathbf{R}_i)(\Delta\mathbf{W}_i\mathbf{R}_i)^\dagger$. For an example x with layer input $\mathbf{R}_i(x)$, we retain the backbone and current-adapter contributions while projecting the earlier-adapter contributions onto $\mathcal{S}_i$. This gives

$$\mathbf{Z}_i(x) := \left(\mathbf{W}_0 + \mathbf{P}_i \sum_{k<i} \Delta\mathbf{W}_k + \Delta\mathbf{W}_i\right)\mathbf{R}_i(x), \tag{9}$$

$$\mathbf{E}_i(x) := (\mathbf{I} - \mathbf{P}_i)\sum_{k<i} \Delta\mathbf{W}_k\mathbf{R}_i(x), \tag{10}$$

where $\mathbf{Z}_i(x)$ and $\mathbf{E}_i(x)$ form the exact decomposition $\mathbf{H}_i(x) = \mathbf{Z}_i(x) + \mathbf{E}_i(x)$. We assess whether $\mathbf{Z}_i$ can serve as a proxy for $\mathbf{H}_i$ by measuring its effect on downstream classification risk.

Definition 3.2 (ϵ-Decision Sufficiency)**.** *Fix a layer, a task-i evaluation distribution p_i, and a loss ℓ. Let g_x denote the fixed downstream computation, with all other inputs held at their original values for example x. The risk of an activation map $\mathbf{V}$ is*

$$\mathcal{R}_i(\mathbf{V}) = \mathbb{E}_{(x,y)\sim p_i}[\ell(g_x(\mathbf{V}(x)), y)].$$

*For $\epsilon_i \geq 0$, we call $\mathbf{Z}_i$ ϵ_i-*decision-sufficient *relative to $\mathbf{H}_i$ when replacing $\mathbf{H}_i$ by $\mathbf{Z}_i$ increases risk by at most ϵ_i. We denote this relation by*

$$\mathbf{Z}_i \approx_{\epsilon_i} \mathbf{H}_i \quad \Longleftrightarrow \quad \mathcal{R}_i(\mathbf{Z}_i) \leq \mathcal{R}_i(\mathbf{H}_i) + \epsilon_i. \tag{11}$$

During task-i training, the current task-exclusive adapter is updated while earlier adapters remain frozen, so the decision-driving signal for task i flows primarily through the activation subspace $\mathcal{S}_i$. This motivates examining its output subspace $\mathcal{S}_i$ for task-relevant changes. The layerwise residual-removal experiment in Appendix F.7 provides empirical support for decision sufficiency on the task-exclusive LoRA branches. We therefore use $\mathbf{Z}_i$ as a decision-level proxy to study the effect of adapter deletion on these projected representations.

Projected effect of adapter deletion. Let the superscript $(-j)$ denote deletion of $\Delta\mathbf{W}_j$ at a fixed layer, with the original inputs and projectors held fixed. Suppressing x for readability, Eqs. 6–8 and Eq. 9 give

$$\mathbf{Z}_{j-1}^{(-j)} = \mathbf{Z}_{j-1}, \tag{12}$$

$$\mathbf{Z}_{j}^{(-j)} = \mathbf{Z}_{j} - \Delta\mathbf{W}_j\mathbf{R}_j, \tag{13}$$

$$\mathbf{Z}_{j+1}^{(-j)} = \mathbf{Z}_{j+1} - \mathbf{P}_{j+1}\Delta\mathbf{W}_j\mathbf{R}_{j+1}. \tag{14}$$

These identities distinguish the effects on earlier, deleted, and later tasks. The earlier task $j-1$ is unaffected because $\Delta\mathbf{W}_j\mathbf{R}_{j-1} = 0$ under Eq. 2 (Fig. 2(a)). For task j, deletion removes its own adapter contribution $\Delta\mathbf{W}_j\mathbf{R}_j$ (Fig. 2(b)). For the later task $j+1$, it can also remove a contribution within $\mathcal{S}_{j+1}$, changing the projected activation (Fig. 2(c)). To preserve $\mathbf{Z}_{j+1}$, we therefore require

$$\mathbf{P}_{j+1}\Delta\mathbf{W}_j\mathbf{R}_{j+1} = (\Delta\mathbf{W}_{j+1}\mathbf{R}_{j+1})(\Delta\mathbf{W}_{j+1}\mathbf{R}_{j+1})^{\dagger}\Delta\mathbf{W}_j\mathbf{R}_{j+1} = 0. \tag{15}$$

Lemma 3.3. *Let* $\mathbf{P}$ *be the orthogonal projector onto* $\mathrm{col}(\mathbf{X})$*, i.e.,* $\mathbf{P} = \mathbf{X}\mathbf{X}^{\dagger}$*. Then, for any* $\mathbf{Y}$ *of matching height,*

$$\mathbf{P}\mathbf{Y} = 0 \quad \Longleftrightarrow \quad \mathbf{X}^{\top}\mathbf{Y} = 0.$$

Proof is provided in Appendix A.1.

By Lemma 3.3, Eq. 15 is equivalent to

$$(\Delta\mathbf{W}_{j+1}\mathbf{R}_{j+1})^{\top}\Delta\mathbf{W}_j\mathbf{R}_{j+1} = 0, \tag{16}$$

$$\mathbf{R}_{j+1}^{\top}\mathbf{B}_{j+1}^{\top}\mathbf{A}_{j+1}^{\top}\mathbf{A}_j\mathbf{B}_j\mathbf{R}_{j+1} = 0. \tag{17}$$

A sufficient condition is $\mathbf{A}_{j+1}^{\top}\mathbf{A}_j = 0$. Generalizing to all task-exclusive adapters yields

$$\mathbf{A}_i^{\top}\mathbf{A}_j = 0, \quad \forall i \neq j \tag{18}$$

This condition preserves the local projected activation after deletion, complementing the continual-learning constraint in Eq. 2. Decision sufficiency alone does not guarantee prediction preservation under adapter deletion, which also changes residual activations and potentially later-layer inputs. The projectors are used for analysis; CAD encourages the condition through adapter training and evaluates retained-task performance empirically.

To enforce this orthogonality, we introduce an orthogonality loss $\mathcal{L}_o^t$ applied during training of task t,

$$\boxed{\mathcal{L}_o^t = \sum_{i=1}^{t-1} \|\mathbf{A}_i^{\top}\mathbf{A}_t\|_F} \tag{19}$$

With standard cross-entropy loss $\mathcal{L}_{ce}^t$, overall incremental learning objective becomes

$$\min_{\Theta} \ \frac{1}{T}\sum_{t=1}^{T}\big(\mathcal{L}_{ce}^t + \lambda\mathcal{L}_o^t\big), \quad \mathcal{L}_{ce}^t = \frac{1}{|\mathcal{D}_t|}\sum_{(\mathbf{x},y)\in\mathcal{D}_t} l(\mathbf{x}, y, \mathbf{W}_t). \tag{20}$$

3.4 Basis-Deletion Strategy

GPM update. After completing the t-th learning task, the Gradient Projection Memory (GPM) $\mathbf{M}_t$ is updated to include basis vectors that span the important representation subspace of task t. To extract the component of the input representation orthogonal to the existing memory, we compute

$$\hat{\mathbf{R}}_t = \mathbf{R}_t - \mathbf{M}_t\mathbf{M}_t^{\top}\mathbf{R}_t, \tag{21}$$

$$\hat{\mathbf{R}}_{t,k} \approx \hat{\mathbf{U}}_t^k\hat{\mathbf{\Sigma}}_t^k(\hat{\mathbf{V}}_t^k)^{\top}. \tag{22}$$

The rank k is selected using a cumulative variance threshold

$$\frac{\|\mathbf{M}_t\mathbf{M}_t^{\top}\mathbf{R}_t\|_F^2}{\|\mathbf{R}_t\|_F^2} + \frac{\|\hat{\mathbf{R}}_{t,k}\|_F^2}{\|\mathbf{R}_t\|_F^2} \geq \epsilon_{th}, \tag{23}$$

making sure that the retained directions collectively explain at least ϵ_{th} of the total variance in $\mathbf{R}_t$. Finally, the GPM is expanded by concatenating the top-k left singular vectors,

$$\mathbf{M}_{t+1} = [\mathbf{M}_t \mid \hat{\mathbf{U}}_t^k]. \tag{24}$$

Algorithm 1 Pseudo-algorithm of CAD

1: **for** $i = 1 : |\mathbb{C}|$ **do**
2: Receive command $C_i = (I_i, T_i, D_i) \in \mathbb{C}$
3: **if** $I_i = L$ **then**
4: $t \leftarrow T_i, \quad \mathcal{D}_t \leftarrow D_i$
5: Initialize $\Delta\mathbf{W}_t$ following InfLoRA.
6: **for** $\mathcal{B}_t \in \mathcal{D}_t$ **do**
7: Update $\mathbf{A}_t$ with loss $L_{ce}^t + L_o^t$
8: **for all** basis $\mathbf{v} = \mathbf{M}_t[\cdot, a]$ **do**
9: **if** $S_a \geq \varepsilon_e$ **then**
10: $\mathbf{v}.\text{task-list}.\text{append}(t)$
11: **else**
12: $\mathbf{v}.\text{task-list}.\text{append}(\text{None})$
13: GPM Update: $\mathbf{M}_{t+1} = [\mathbf{M}_t \mid \mathbf{U}_t^k]$
14: **for all** basis $\mathbf{v} \in \mathbf{U}_t^k$ **do**
15: $\mathbf{v}.\text{task-list} \leftarrow [\,]$
16: $\mathbf{v}.\text{task-list}.\text{append}(t)$
17: **else if** $I_i = U$ **then**
18: $j \leftarrow T_i$
19: Discard $\Delta\mathbf{W}_j$
20: **for all** basis $\mathbf{v} \in \mathbf{M}_t$ **do**
21: **if** $\mathbf{v}.\text{task-list}$ contains only j **then**
22: Delete $\mathbf{v}$
23: **else**
24: **continue**

Table 2: Benchmark results on CIFAR100 and ImageNet-R, showing average performance across $\mathbb{C}_1 - \mathbb{C}_3$. CAD consistently improves over baselines in continual learning and unlearning metrics, demonstrating its strong knowledge preservation-removal balance. The results are averaged across the three random seeds.

Methods	Scheme		CIFAR100						ImageNet-R					
	CLU	Buffer	ACC_r (↑)	BWT_r (↑)	ACC_f (↓)	BWT_f (↓)	S (↑)	MR (↓)	ACC_r (↑)	BWT_r (↑)	ACC_f (↓)	BWT_f (↓)	S (↑)	MR (↓)
R-Joint		0	82.52 ±0.94	-	-	-	-	-	80.99 ±0.62	-	-	-	-	-
R-Seq		0	69.15 ±0.69	-9.60 ±0.40	-	-	-	-	78.97 ±0.58	-3.48 ±0.57	-	-	-	-
DER		500	45.73 ±1.38	-49.07 ±1.58	26.17 ±0.94	-69.51 ±1.36	55.12 ±0.65	52.20 ±1.37	62.02 ±0.24	-25.93 ±1.85	42.20 ±1.57	-47.99 ±1.42	52.81 ±0.83	41.89 ±1.45
UniCLUN	✓	500	26.88 ±1.25	-68.40 ±1.64	3.77 ±0.64	**-92.07 ±0.35**	38.75 ±1.34	39.70 ±1.85	29.75 ±1.37	-58.74 ±1.46	3.75 ±0.52	**-87.06 ±0.48**	40.80 ±1.58	51.85 ±1.63
PALL	✓	500	40.34 ±1.58	-30.56 ±1.32	0.05 ±0.32	-73.29 ±1.36	51.75 ±0.47	44.74 ±1.75	45.26 ±1.48	-27.05 ±1.74	0.35 ±0.63	-73.96 ±1.58	56.03 ±0.36	7.54 ±0.65
LwF		0	17.38 ±1.34	-77.00 ±1.28	8.55 ±0.89	-86.62 ±0.65	28.18 ±1.69	67.22 ±0.84	36.14 ±0.24	-50.41 ±1.27	15.49 ±0.86	-73.56 ±1.24	46.87 ±1.76	25.70 ±1.95
EWC		0	10.53 ±0.46	-85.49 ±1.73	6.83 ±0.27	-89.55 ±0.85	17.91 ±0.58	53.19 ±1.48	13.48 ±0.73	-75.01 ±1.15	8.76 ±0.05	-81.76 ±0.37	21.76 ±0.43	30.98 ±1.37
LSF	✓	0	12.45 ±1.47	-80.38 ±1.84	7.10 ±0.62	-86.42 ±0.45	20.56 ±0.73	50.08 ±1.84	17.18 ±0.36	-69.71 ±1.95	12.13 ±0.84	-76.90 ±1.74	26.74 ±1.25	33.33 ±1.73
InfLoRA		0	65.50 ±0.93	-13.75 ±0.52	0.00 ±0.00	-83.27 ±0.73	71.82 ±0.21	27.67 ±1.52	73.05 ±0.93	-8.12 ±0.73	0.00 ±0.00	-85.16 ±0.95	78.08 ±0.78	6.04 ±0.52
CLPU	✓	0	57.00 ±0.05	-27.33 ±1.36	0.00 ±0.00	-88.00 ±0.26	67.74 ±0.72	38.99 ±1.88	71.63 ±0.14	-11.33 ±0.48	0.00 ±0.00	-86.58 ±0.36	77.83 ±0.38	7.33 ±0.59
CAD	✓	**0**	**70.70 ±1.86**	**-8.46 ±0.47**	**0.00 ±0.00**	-83.39 ±1.48	**75.67 ±0.75**	**27.10 ±1.98**	**75.51 ±0.82**	**-5.13 ±1.94**	**0.00 ±0.00**	-84.73 ±0.79	**79.58 ±0.26**	**5.24 ±1.44**

Basis Deletion. Upon receiving an unlearning command $C_i = (I_i = U, T_i = j, D_i = \emptyset)$, the bases introduced during task j should be removed when they are not reused by later tasks, both to support unlearning and to maintain memory efficiency. However, directly deleting $\hat{\mathbf{U}}_j^k$ can be unsafe, as its components may overlap with the representation subspaces of later tasks ($i > j$); that is, representations $\mathbf{R}_i$ may contain directions aligned with $\hat{\mathbf{U}}_j^k$. Naively removing these bases could therefore harm the performance of subsequent tasks (Fig. 5 in Appendix C).

To selectively identify the bases associated with each task, we define the projection coefficient matrix

$$\mathbf{C}_t = \mathbf{M}_t^\top \mathbf{R}_t \in \mathbb{R}^{m_t \times n}, \tag{25}$$

where $\mathbf{C}_t[a, b]$ is the inner product between the a-th basis vector $\mathbf{M}_t[\cdot, a]$ and the b-th input representation $\mathbf{R}_t[\cdot, b]$. The energy contribution of basis $\mathbf{M}_t[\cdot, a]$ to the input representation $\mathbf{R}_t$ is

$$S_a = \frac{\|\mathbf{C}_t[a, :]\|_2^2}{\|\mathbf{R}_t\|_F^2}. \tag{26}$$

We determine if this basis is associated with task t's representation subspace with the decision rule:

$$\begin{cases} \mathbf{M}_t[\cdot, a] \in \text{col}(\mathbf{R}_t), & S_a \geq \epsilon_e, \\ \mathbf{M}_t[\cdot, a] \notin \text{col}(\mathbf{R}_t), & S_a < \epsilon_e. \end{cases} \tag{27}$$

This criterion associates each basis vector with the set of task IDs whose representation subspaces it supports. Algorithm 1 summarizes the overall CAD procedure. When a new basis vector is added to the GPM, a list-like structure, task-list, is initialized with the current task ID (lines 15–16). Whenever that basis is later found, via the decision rule above, to lie within the representation subspace of a subsequent task, the corresponding task ID is appended to its task-list (line 10). Upon receiving an unlearning command for task j, the algorithm deletes any basis vector whose task-list contains only task j (lines 21–22); otherwise, if the basis is shared with later tasks, it is retained in the GPM.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Architecture and Datasets. We evaluate CAD on two standard continual learning and unlearning benchmarks: CIFAR-100 Krizhevsky et al. (2009) and ImageNet-R Hendrycks et al. (2021). Each dataset is divided into 10 disjoint tasks with 10 and 20 classes per task, respectively. To evaluate under the CLU framework, we define three command sequences, $\mathbb{C}_1$–$\mathbb{C}_3$, where each command $C_i = I_i \oplus T_i$ specifies an operation type ($I_i \in \{L, U\}$) and task index T_i:

$$\mathbb{C}_1 = \{L0, L1, U0, L2, U1, L3, L4, U4, L5, L6, U5, L7, L8, U7, L9\},$$
$$\mathbb{C}_2 = \{L0, L1, L2, L3, L4, L5, L6, L7, L8, L9, U0\},$$
$$\mathbb{C}_3 = \{L0, L1, L2, L3, L4, L5, L6, L7, L8, U5, U6, U7, U8, L9, U9\}.$$

For CIFAR-100, we use ViT-Tiny Wu et al. (2022) with 192-dimensional embeddings; for ImageNet-R, we use ViT-B/16 Dosovitskiy et al. (2021) with 768-dimensional embeddings. Both models have 12 transformer blocks and are pre-trained on ImageNet-21K.

Evaluation Metrics. We evaluate continual learning and unlearning performance using standard continual learning metrics Lopez-Paz & Ranzato (2017). Given a command sequence $\mathbb{C}$, we define the *Retain Set* $\mathcal{T}_\mathrm{r}$ and *Forget Set* $\mathcal{T}_\mathrm{f}$ as

$\mathcal{T}_\mathrm{r}(\mathbb{C}) \coloneqq \{T_i :\ I_i = L \wedge \nexists j > i,\ T_j = T_i \wedge I_j = U\}$, $\mathcal{T}_\mathrm{f}(\mathbb{C}) \coloneqq \{T_i :\ I_i = U \wedge \nexists j > i,\ T_j = T_i\}$.

For example, $\mathcal{T}_\mathrm{r}(\mathbb{C}_3) = \{0{-}4\}$ and $\mathcal{T}_\mathrm{f}(\mathbb{C}_3) = \{5{-}9\}$. For $i \in \{r, f\}$, we compute *average accuracy* and *backward transfer* as

$$\mathrm{ACC}_i = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \mathrm{acc}_t^{|\mathbb{C}|},\ \mathrm{BWT}_i = \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \left(\mathrm{acc}_t^{|\mathbb{C}|} - \mathrm{acc}_t^{t}\right),$$

where $\mathrm{acc}_t^{|\mathbb{C}|}$ denotes the final accuracy on task t after executing the full sequence $\mathbb{C}$, and acc_t^{t} denotes the accuracy immediately after learning task t. Following Shibata et al. (2021), we also report the *Learning with Selective Forgetting Measure* (LSFM),

$$\mathrm{S} = \frac{2|\mathrm{ACC}_r||\mathrm{BWT}_f|}{|\mathrm{ACC}_r| + |\mathrm{BWT}_f|}.$$

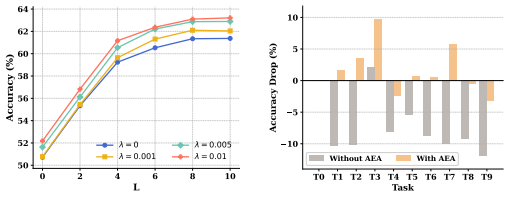
Finally, we report *Member Rate* (MR) to assess membership-inference risk Shokri et al. (2017).

Baselines. We compare CAD against representative continual learning and unlearning (CLU) methods. As the first class-incremental CLU framework, CAD is evaluated against existing CLU methods by adapting LSF Shibata et al. (2021), CLPU Liu et al. (2022), UniCLUN Chatterjee et al. (2024), and PALL Özdenizci et al. (2025) to the class-incremental setting without task IDs. Given the limited number of existing CLU baselines, we also adapt continual learning methods LwF Li & Hoiem (2017), EWC Kirkpatrick et al. (2017), DER++ Buzzega et al. (2020), and InfLoRA Liang & Li (2024) to the CLU setting by removing the task-specific mechanisms associated with the forgotten task. For example, in DER++, replay samples corresponding to the target task are discarded when an unlearning command is issued. Although our main focus is the data-free CLU setting, we additionally include data-permitted variants (DER++, UniCLUN) with 500 episodic samples for reference. Lastly, we provide joint training (R-Joint, where the retain set is trained jointly) and sequential training (R-Seq, where the retain set is trained sequentially) as reference baselines.

Training Details. All methods are trained with Adam Kingma & Ba (2015) for 5 epochs on CIFAR-100 and 10 epochs on ImageNet-R, using a batch size of 128. Following the original LoRA design Hu et al. (2022), adapters are inserted into the key and value projections of each attention block. Further experimental details are provided in Appendix D.2.

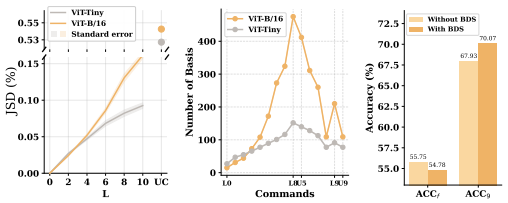
4.2 EXPERIMENTAL RESULTS

Main Results. Table 2 reports the main results, averaged over $\mathbb{C}_1$–$\mathbb{C}_3$. CAD achieves the strongest overall retention–forgetting trade-off, attaining the highest S score and the lowest MR. It preserves high Retain Set performance while driving Forget Set accuracy close to zero, indicating effective zero-shot unlearning with limited collateral degradation. In contrast, buffer-free baselines such as EWC and LSF suffer from severe forgetting, and naively applying InfLoRA degrades retained tasks due to inter-task entanglement. Although buffer-based methods improve stability, they still underperform CAD and require storing past data.



(a) (b)

Figure 3: Effect of Anti-Entanglement of Activation (AEA) on command sequence C_2 with CIFAR100. (a) Retain accuracy improves with AEA weight λ and deletion depth L. (b) AEA reduces downstream-task degradation after unlearning operation $U0$.



(a) (b) (c)

Figure 4: Results on command sequence C_3 with CIFAR100. (a) Jensen–Shannon distance vs. deletion depth L. (b) Number of GPM bases in layer 12 across commands. (c) Forget-set and task-9 accuracy with and without BDS.

Effect of AEA. We evaluate the effect of Anti-Entanglement of Activation (AEA) on retained-task performance using command sequence $\mathbb{C}_2$, focusing on how unlearning $U0$ affects subsequent tasks. Fig. 3(a) shows Retain Set accuracy as a function of deletion layer L and regularization weight λ. Accuracy drops when L moves toward earlier layers, especially below 4, because shared representations useful across tasks are removed. In contrast, larger λ improves Retain Set accuracy by encouraging greater separation between task representations. Fig. 3(b) shows that Anti-Entanglement of Activation (AEA) design substantially reduces the degradation of subsequent tasks after unlearning, confirming that task-exclusive deletion combined with AEA improves retained-task stability.

Effect of Deletion Layer. We next examine how the deletion layer L affects the fidelity of unlearning. Following Liu et al. (2022), we measure this using the *Jensen–Shannon Distance* (JSD) between the unlearned model and a retain-only reference model. Given a command sequence $\mathbb{C}$, let

$$\mathbb{C}_r := \{\, C_i \in \mathbb{C} : I_i = L \ \wedge\ \nexists j > i \text{ s.t. } T_j = T_i,\ I_j = U \,\} \subseteq \mathbb{C}$$

denote the subsequence containing only retained tasks. For a model $f(\mathbf{x};\theta)$ and a CLU algorithm A with initialization θ_0, we evaluate

$$\sum_{t\in\mathcal{T}_r} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_t}\big[\,\mathrm{JSD}(f(\mathbf{x};A(\theta_0,\mathbb{C})), f(\mathbf{x};A(\theta_0,\mathbb{C}_r)))\big].$$

For $\mathbb{C}_3$, we compare the final model $A(\theta_0,\mathbb{C})$ against the retain-only checkpoint $A(\theta_0,\mathbb{C}_r)$ after $L4$ (i.e., before unlearning). Fig. 4(a) shows that JSD is zero at $L = 0$, indicating exact unlearning. As L increases, JSD grows because fewer task-specific components are removed and more shared knowledge from later tasks ($L5$–$L9$) remains. CAD consistently achieves substantially lower JSD than UniCLUN Chatterjee et al. (2024), indicating closer agreement with the retain-only reference.

Effect of Basis-Deletion. Figure 4(b) shows the number of basis vectors maintained in the final (12-th) layer of the Gradient Projection Memory (GPM). During learning phases, the number of stored bases increases as the GPM expands to encode newly acquired task-relevant subspaces. During unlearning phases, it decreases as bases associated only with the forgotten task are removed by the proposed Basis-Deletion Strategy (BDS). This dynamic evolution of the GPM dimensionality indicates that BDS preserves shared subspace components while pruning obsolete ones, improving memory efficiency without sacrificing representational integrity.

Beyond this memory benefit, BDS also improves both learning and unlearning dynamics. By removing obsolete bases, it remove orthogonality constraints imposed by unlearned tasks, increasing plasticity. In task-agnostic branches, BDS exposes the forgotten task to *catastrophic forgetting*, serving as a unlearning mechanism of task-agnostic branch. Figure 4(c) reports accuracy on the Forget Set and the 9-th learning task ($L9$) under command sequence $\mathbb{C}_3$ when all layers are treated as task-agnostic branches ($L = \infty$). In this setting, unlearning occurs solely through catastrophic forgetting during training on $L9$. The results show that BDS improves plasticity for new tasks (higher ACC_9) while lowering Forget Set accuracy (lower ACC_f), indicating intended forgetting.

5 Conclusion

We introduced **CAD**, a class-incremental learning framework for zero-shot unlearning. The framework couples task-specific dual-adapter learning with activation-level anti-entanglement and selective basis deletion, enabling targeted unlearning while preserving performance on retained tasks. Experiments demonstrate its effectiveness for buffer-free continual learning and unlearning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with checking mathematical derivations, assisting with the implementation and debugging of research code, and interpreting experimental results. We also used generative AI tools to draft and edit portions of the manuscript for clarity, grammar, wording, and organization, and to refine code and presentation. We did not use generative AI tools to generate synthetic datasets, translate research material, clean or reformat datasets, or conduct qualitative or thematic data analysis.

All AI-assisted mathematical arguments were independently checked by the authors, and AI-assisted code was reviewed and tested against the intended algorithms. All reported experimental results, figures, tables, and quantitative claims were verified against the authors' experiments. Any AI-assisted suggestions regarding the literature were checked against the original sources. We reviewed all AI-assisted content for technical accuracy and consistency with the final manuscript. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies continual learning and machine unlearning under privacy-preserving constraints, with the goal of enabling learned task-specific knowledge to be removed without retaining or revisiting past training data. Our experiments use existing benchmark datasets and do not involve a new human-subject study or collection of user data. While effective unlearning may support privacy-sensitive applications and reduce the cost of responding to data-removal requests, the empirical measures considered in this work should not be interpreted as formal guarantees of complete information removal or privacy. In particular, low forgotten-task accuracy and the reported membership-inference results measure specific observable behaviors under the evaluated attacks and do not certify the absence of all residual information. Unlearning mechanisms may also have dual-use implications, for example by enabling selected behaviors or audit-relevant information to be removed without transparent tracking. We therefore encourage future work on stronger unlearning verification, adversarial recovery tests, and transparent auditing mechanisms for continual learning and unlearning systems.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce the proposed method and experiments throughout the paper and appendix. Section 3 describes the complete CAD framework, including the dual-branch LoRA architecture, Anti-Entanglement of Activation (AEA), the Basis-Deletion Strategy (BDS), and the overall procedure in Algorithm 1. Appendix A provides the proof supporting the activation-space analysis. Section 4 specifies the benchmark construction, command sequences, evaluation metrics, baselines, and training setup, while Appendix D provides dataset statistics, baseline configurations, and the hyperparameters used for CAD. Appendix E gives the complete membership-inference evaluation procedure, and Appendix F reports additional experiments, ablations, sensitivity analyses, and memory measurements. Experiments are repeated across three random seeds, and the corresponding implementation is provided through the anonymous code link accompanying the submission.

REFERENCES

Sayanta Adhikari, Vishnuprasadh Kumaravelu, and PK Srijith. An unlearning framework for continual learning. *arXiv preprint arXiv:2509.17530*, 2025.

Ari S. Benjamin, David Rolnick, and Konrad P. Körding. Measuring and regularizing networks in function space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkMwpiR9Y7.

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.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pp. 463–480. IEEE Computer Society, 2015. doi: 10.1109/SP.2015.35. URL https://doi.org/10.1109/SP.2015.35.

Romit Chatterjee, Vikram Chundawat, Ayush Tarun, Ankur Mali, and Murari Mandal. A unified framework for continual learning and unlearning. *arXiv preprint arXiv:2408.11374*, 2024.

Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan S. Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 7210–7217. AAAI Press, 2023a. doi: 10.1609/AAAI.V37I6.25879. URL https://doi.org/10.1609/aaai.v37i6.25879.

Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan S. Kankanhalli. Zero-shot machine unlearning. *IEEE Trans. Inf. Forensics Secur.*, 18:2345–2354, 2023b. doi: 10.1109/TIFS.2023.3265506. URL https://doi.org/10.1109/TIFS.2023.3265506.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=gn0mIhQGNM.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 12043–12051. AAAI Press, 2024. doi: 10.1609/AAAI.V38I11.29092. URL https://doi.org/10.1609/aaai.v38i11.29092.

Jack Foster, Kyle Fogarty, Stefan Schoepf, Zack Dugue, Cengiz Öztireli, and Alexandra Brintrup. An information theoretic approach to machine unlearning. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=t1utIThKHD.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9304–9312, 2020a.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374 of *Lecture Notes in Computer Science*, pp. 383–398. Springer, 2020b. doi: 10.1007/978-3-030-58526-6_23. URL https://doi.org/10.1007/978-3-030-58526-6_23.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

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, 2021.

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. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

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.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/abdeb6f575ac5c6676b747bca8d09cc2-Abstract.html.

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

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.

Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning. In *Conference on Lifelong Learning Agents*, pp. 243–254. PMLR, 2022.

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

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

H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016. URL https://api.semanticscholar.org/CorpusID:14955348.

Ozan Özdenizci, Elmar Rueckert, and Robert Legenstein. Privacy-aware lifelong learning. *arXiv preprint arXiv:2505.10941*, 2025.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1406–1415, 2018. URL https://api.semanticscholar.org/CorpusID:54458071.

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

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=3AOj0RCNC2.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018.

Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In *IJCAI*, volume 3, pp. 4, 2021.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

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. URL https://openaccess.thecvf.com/content/CVPR2023/html/Smith_CODA-Prompt_COntinual_Decomposed_Attention-Based_Prompting_for_Rehearsal-Free_Continual_Learning_CVPR_2023_paper.html.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXI*, volume 13681 of *Lecture Notes in Computer Science*, pp. 68–85. Springer, 2022. doi: 10.1007/978-3-031-19803-8_5. URL https://doi.org/10.1007/978-3-031-19803-8_5.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pp. 10355–10366. PMLR, 2020.

A Additional Proofs

A.1 Proof of Lemma 3.3

Proposition A.1 (Orthogonal projector via pseudoinverse)**.** *Let $\mathbf{X} \in \mathbb{R}^{m\times n}$ have rank r. Consider a (full) SVD of $\mathbf{X}$, $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, with*

$$\mathbf{U} = [\mathbf{U}_r \quad \mathbf{U}_\perp],\ \mathbf{V} = [\mathbf{V}_r \quad \mathbf{V}_\perp],\ \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma_r} & 0 \\ 0 & 0 \end{bmatrix},$$

where $\mathbf{U}_r \in \mathbb{R}^{m\times r}$, $\mathbf{V}_r \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\mathbf{\Sigma}_r = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_i > 0$. The Moore–Penrose pseudoinverse is

$$\mathbf{X}^\dagger = \mathbf{V}\,\mathbf{\Sigma}^\dagger\,\mathbf{U}^\top = \mathbf{V}_r\,\mathbf{\Sigma}_r^{-1}\,\mathbf{U}_r^\top.$$

Then

$$\mathbf{P} \;:=\; \mathbf{X}\,\mathbf{X}^\dagger \;=\; \mathbf{U}_r\,\mathbf{U}_r^\top$$

is the orthogonal projector onto $\operatorname{col}(\mathbf{X})$.

Proof. $\mathbf{X}\mathbf{X}^\dagger = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \cdot \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^\top = \mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\dagger)\mathbf{U}^\top = \mathbf{U}\begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{U}^\top = \mathbf{U}_r\mathbf{U}_r^\top$. This matrix is symmetric and idempotent, and its range is $\operatorname{col}(\mathbf{X})$, so it is an orthogonal projector. □

Lemma A.2 (Equivalent tests for orthogonality to $\operatorname{col}(\mathbf{X})$)**.** *Let $\mathbf{P} = \mathbf{X}\mathbf{X}^\dagger = \mathbf{U}_r\mathbf{U}_r^\top$ as above and $\mathbf{Y} \in \mathbb{R}^{m\times p}$. The following are equivalent:*

$$\text{(i) } \mathbf{P}\mathbf{Y} = 0 \quad\Longleftrightarrow\quad \text{(ii) } \mathbf{X}^\top\mathbf{Y} = 0 \quad\Longleftrightarrow\quad \text{(iii) } \mathbf{X}^\dagger\mathbf{Y} = 0.$$

Proof. Since $\mathbf{P} = \mathbf{U}_r\mathbf{U}_r^\top$, $\mathbf{P}\mathbf{Y} = 0$ iff $\mathbf{U}_r^\top\mathbf{Y} = 0$.

(i)⇒(ii): $\mathbf{X}^\top\mathbf{Y} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{Y} = \mathbf{V}_r\mathbf{\Sigma}_r(\mathbf{U}_r^\top\mathbf{Y}) = 0$.

(i)⇒(iii): $\mathbf{X}^\dagger\mathbf{Y} = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^\top\mathbf{Y} = \mathbf{V}_r\mathbf{\Sigma}_r^{-1}(\mathbf{U}_r^\top\mathbf{Y}) = 0$.

(ii)⇒(i): If $\mathbf{X}^\top\mathbf{Y} = 0$, then $\mathbf{V}^\top\mathbf{X}^\top\mathbf{Y} = \mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{Y} = 0$, so $\mathbf{\Sigma}_r(\mathbf{U}_r^\top\mathbf{Y}) = 0$. Since $\mathbf{\Sigma}_r$ is invertible, $\mathbf{U}_r^\top\mathbf{Y} = 0$ and hence $\mathbf{P}\mathbf{Y} = 0$.

(iii)⇒(i): If $\mathbf{X}^\dagger\mathbf{Y} = 0$, then $\mathbf{V}^\top\mathbf{X}^\dagger\mathbf{Y} = \mathbf{\Sigma}^\dagger\mathbf{U}^\top\mathbf{Y} = 0$, so $\mathbf{\Sigma}_r^{-1}(\mathbf{U}_r^\top\mathbf{Y}) = 0$, and hence $\mathbf{U}_r^\top\mathbf{Y} = 0$, so $\mathbf{P}\mathbf{Y} = 0$. □

B Jensen-Shannon Distance

Jensen–Shannon distance. Let p and q be probability distributions on a finite set $\mathcal{Y}$ and let $m := \frac{1}{2}(p+q)$. The Kullback–Leibler divergence is

$$D_{\mathrm{KL}}(p\|q) \;:=\; \sum_{y\in\mathcal{Y}} p(y)\,\log\frac{p(y)}{q(y)}.$$

The Jensen–Shannon *divergence* is defined as

$$\mathrm{JS}(p\|q) \;:=\; \tfrac{1}{2}D_{\mathrm{KL}}(p\|m) \;+\; \tfrac{1}{2}D_{\mathrm{KL}}(q\|m),\ m = \tfrac{1}{2}(p+q).$$

The Jensen–Shannon *distance* (JSD) is the square root of the divergence, i.e.,

$$\mathrm{JSD}(p,q) \;:=\; \sqrt{\mathrm{JS}(p\|q)}.$$

JSD is symmetric, equals 0 iff $p = q$, and is bounded, i.e., $0 \le \mathrm{JSD}(p,q) \le 1$ if log is base 2 (or $\le \sqrt{\log 2}$ with natural logs).

C MOTIVATION OF BASIS-DELETION STRATEGY

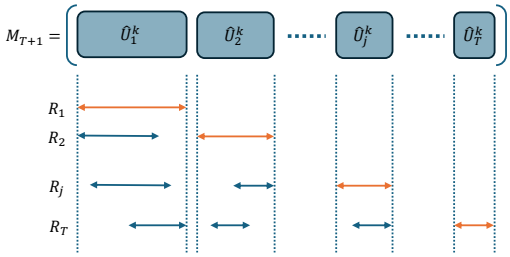


Figure 5: Possible basis configuration of GPM memory. Previously included bases can be used for future tasks.

Figure 6 illustrates the structure of the GPM memory after sequential task learning. Each task introduces a basis group into the memory, while later tasks may reuse bases from earlier tasks. Thus, task representations are not fully isolated, and knowledge of a task can persist through shared bases used by subsequent tasks.

This motivates the Basis-Deletion strategy. Simply removing task-specific parameters is insufficient for reliable unlearning, since residual information may remain in the shared GPM memory. Therefore, when unlearning a task, the corresponding task-related bases should also be selectively removed to eliminate remaining task information. Importantly, the dependency is directional: future tasks may reuse previous bases, but not vice versa, enabling selective basis deletion while preserving unrelated earlier knowledge.

D EXPERIMENTAL DETAILS

D.1 DATASETS

Table 3 summarizes statistic of datasets used in the main paper. Unless otherwise specified, ablation studies are conducted on the CIFAR-100 dataset. All experiments marked with ViT-B/16 are performed on the ImageNet-R dataset.

Table 3: Dataset statistics of CIFAR-100 and ImageNet-R.

	Split CIFAR-100	**Split ImageNet-R**
Image size ($C \times H \times W$)	$3 \times 32 \times 32$	$3 \times 224 \times 224$
Num. of tasks	10	10
# Classes / task	10	20
# Training samples	50,000	48,167
# Test samples	10,000	12,000
Memory per sample	3.072 KB	147.00 KB

D.2 HYPERPARAMETER SETTINGS AND BASELINES

Table 4: Hyper-parameters used for all baselines.

Method	Hyperparameters
DER++	α: 0.5, β: 0.5, episodic buffer size: 500, Rank of LoRA r: 10
UniCLUN	α_1: 0.1, α_2: 0.1, α_3: 0.3, ρ: 1.0 ,τ: 0.07, p: 0.7, γ: 0.5, m: 0.999, r: 10, buffer size: 500
LwF	λ_o: 5, Rank of LoRA r: 10
EWC	λ: 0.5, Rank of LoRA r: 10
LSF	λ_o: 5, γ_{SF}: 10.0, Rank of LoRA r: 10
CLPU	Rank of LoRA r: 10
InfLoRA	ϵ_{th}: 0.95, Rank of LoRA r: 10
CAD	ϵ_e: 0.01 ϵ_{th}: 0.95 λ: 0.01 Deletion layer L: 6 Rank of LoRA r: 10

Table 4 specifies all hyperparameters used in the frameworks evaluated in the main results. For each baseline, we follow the hyperparameter settings provided in their original implementations. All experiments are run three times with different random seeds (e.g., 1, 2, 3) on an NVIDIA GH200. We report the average performance. For CAD, we follow the same value of ϵ_{th} and LoRA rank r as introduced in InfLoRA. For λ, and L, we perform a 1-D modest hyperparameter search as shown in Figure 4(a). The results exhibit smooth and predictable trends, allowing the optimal values to be identified easily. Similarly, ϵ_e is selected via a modest 1D search over the range [0.001, 0.1].

The DER++, UniCLUN, LwF, EWC, and LSF baselines use a single LoRA adapter combined with a pretrained backbone and fine-tune the adapter using their respective continual learning algorithms. For unlearning in continual learning baselines adapted to the CLU setup (e.g., DER++, LwF, EWC), the corresponding continual learning components are removed to induce catastrophic forgetting of the target task. Specifically, for DER++, unlearning is performed by removing the target task's rehearsal samples from the memory buffer. For EWC, the Fisher information matrix of the corresponding task is discarded. In the case of LwF, distillation is halted for logits associated with the task to be unlearned. For CLU-native baselines (e.g., UniCLUN, LSF, CLPU), we follow the unlearning procedures as defined in their original literature. All state-of-the-art CLU baselines (e.g., UniCLUN, LSF, CLPU) rely on access to task identifiers at inference time, operating in a task-incremental setting. To ensure a fair comparison, we evaluate these methods without access to task IDs during inference. In particular, the naive version of CLPU assigns a separate network to each task and selects the corresponding network at unlearning time using the provided task ID. To enable CLPU to perform inference without access to task identifiers, we apply a model aggregation strategy similar to FedAvg McMahan et al. (2016). For unlearning, the corresponding LoRA adapter is discarded. In the case of InfLoRA, we assign a separate LoRA adapter to each task and remove the corresponding adapter to perform unlearning. In the case of PALL Özdenizci et al. (2025), we follow the original formulation, which maintains a shared LoRA adapter and learns a task-specific sparse subnetwork for each task via score-based pruning over the LoRA parameters, with previous tasks' masks protected by gradient masking and a rehearsal buffer kept throughout training. For unlearning, we discard the target task's mask and its rehearsal samples, reinitialize the LoRA parameters unique to the forgotten task, and fine-tune the parameters shared with remaining tasks using a DER++-style rehearsal loss. As with other CLU baselines, we evaluate PALL without access to task identifiers at inference time for a fair comparison.

E MEMBERSHIP INFERENCE ATTACK (MIA)

Membership Inference Attack (MIA) evaluates whether the model retains membership information about data that should be *forgotten* by unlearning. Given a trained classifier f and an input x, MIA outputs a binary decision: was x part of the training set (*member*) or not (*non-member*)? Effective unlearning should make forgotten training samples indistinguishable from non-members.

Notation. Let tasks be indexed by $t \in \{0, \dots, T-1\}$, each with a disjoint class set $\mathcal{C}_t$ of size K. For a set of tasks $\mathcal{T}$, let $\mathcal{C}_{\mathcal{T}} := \bigcup_{t \in \mathcal{T}} \mathcal{C}_t$. We denote retained (kept) tasks by $\mathcal{T}_{\text{ret}}$ and unlearned (forgotten) tasks by $\mathcal{T}_{\text{unl}}$ with $\mathcal{T}_{\text{ret}} \cap \mathcal{T}_{\text{unl}} = \emptyset$ and $\mathcal{T}_{\text{ret}} \cup \mathcal{T}_{\text{unl}} = \{0, \dots, T-1\}$. For any class set $\mathcal{C}$, $\mathcal{D}^{\text{train}}(\mathcal{C})$ and $\mathcal{D}^{\text{test}}(\mathcal{C})$ denote the training and test sets.

Attack Principle. Let $z(x) \in \mathbb{R}^C$ be the logits and $p(x) = \text{softmax}(z(x))$ with temperature fixed at T=1. We define:

$$
\begin{aligned}
H(x) &= -\sum_{c=1}^{C} p_c(x)\, \log p_c(x) \quad \text{(entropy)}, \\
\widehat{H}(x) &= H(x)/\log C \quad \text{(normalized entropy)}, \\
\text{maxP}(x) &= \max_c\, p_c(x) \quad \text{(max confidence)}, \\
\text{margin}(x) &= p^{(1)}(x) - p^{(2)}(x) \quad \text{(top–2 gap)}.
\end{aligned}
$$

Empirically, training samples tend to induce *lower* entropy (higher confidence) than non-members. Thus, an attacker can infer membership from entropy-based features. We use a four-dimensional feature vector

$$
\phi(x) \;=\; \big[\, H(x),\; \widehat{H}(x),\; \text{maxP}(x),\; \text{margin}(x) \,\big] \in \mathbb{R}^4.
$$

The following steps describes the MIA procedure used to infer membership of the forget set by attacker.

1. **Feature extraction.** Compute $\phi(x)$ for all $x \in \mathcal{D}^{\text{train}}(\mathcal{C}_{\text{ret}})$ (members; positive class) and $x \in \mathcal{D}^{\text{test}}(\mathcal{C}_{\text{ret}})$ (non-members; negative class), with temperature fixed at T=1. Form $X_r = [X^+; X^-]$ with labels $Y_r = [\mathbf{1}; \mathbf{0}]$.
2. **Attacker training.** Standardize the features to have zero mean with unit variance. Then fit a regularized logistic regression with class balancing.
3. **Evaluation on the forgotten set.** Compute $\phi(x)$ for $x \in \mathcal{D}^{\text{train}}(\mathcal{C}_{\text{unl}})$ and obtain predictions $\hat{Y}_{\text{unl}} \in \{0, 1\}$ from the trained classifier.

Metrics. Our primary metric is the *member rate* (MR) on the forgotten training data defined as

$$
\text{MR} \;=\; \frac{1}{|X_{\text{unl}}|} \sum_i \mathbf{1}\Big[\, \hat{Y}_{\text{unl},i} = 1 \,\Big].
$$

Lower value of MR implies more effective unlearning. A value near 50% implies the attacker makes random guesses(i.e., there is no residual membership signal). The values $\ll 50\%$ indicate that forgotten points look even *less* like members than typical test data (over-forgetting), while values $\gg 50\%$ indicate residual membership leakage.

F ADDITIONAL RESULTS

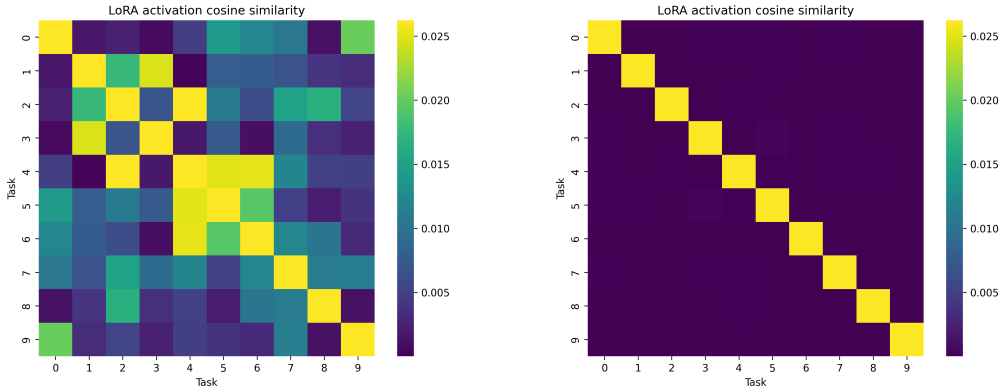
F.1 UNLEARNING TIME COMPARISON

Table 5: Average unlearning time comparison between CAD and UniCLUN.

	ViT-Tiny	**ViT-B/16**
UniCLUN	1.283 sec	17.60 sec
CAD	**0.139** sec	**0.147** sec

Table 6: Experimental result on longer command sequence $\mathbb{C}_4$

Methods	Scheme		Retain Set		Forget Set			
	CLU	Buffer	ACC_r (↑)	BWT_r (↑)	ACC_f (↓)	BWT_f (↓)	S (↑)	MR (↓)
DER++		500	59.28	-30.98	17.90	-73.28	65.54	22.60
UniCLUN	✓	500	47.71	-43.11	1.16	-90.56	62.49	29.31
LwF		0	19.84	-69.07	2.26	-87.45	32.34	40.51
EWC		0	9.48	-81.49	0.10	**-91.44**	17.17	39.57
LSF	✓	0	13.93	-75.48	1.04	-88.96	24.09	43.89
CAD	✓	**0**	**76.33**	**-4.64**	**0.00**	-82.43	**79.26**	**6.40**



(a) (b)

Figure 6: Heatmap illustrating the effect of Anti-Entanglement of Activation. (a) Before applying AEA. (b) After applying AEA.

As mentioned in the main paper, CAD supports arbitrary task learning and independent unlearning without requiring any additional training phase. Table 5 demonstrates that the computational cost of CAD's unlearning operation is very low compared to the baseline method, UniCLUN. This advantage becomes even more pronounced with larger models (e.g., ViT-B/16), where CAD achieves up to 120× faster unlearning.

F.2 Experiments on longer command sequence

In this section, we validate CAD on a longer command sequence. We split the ImageNet-R dataset into 20 non-overlapping chunks, each forming a distinct learning task. Then, we randomly insert 10 unlearning commands, resulting in a 30-step command sequence:

$$\begin{aligned}\mathbb{C}_4 = \{&L0, L1, L2, L3, U1, U3, L4, L5, L6, U5, L7, L8,\\ &L9, U8, L10, L11, U0, U2, L12, L13, L14, L15,\\ &U10, U12, L16, U15, L17, L18, U17, L19\}\end{aligned}$$

Table 6 shows the experimental results on command sequence $\mathbb{C}_4$ with ViT-B/16. All hyperparameters and experimental configurations follow those introduced in Appendix D.2. Due to the increased catastrophic forgetting associated with longer sequences, the performance of buffer-free baselines on the Retain Set deteriorates rapidly to $\sim 10\%$, while the performance of buffer-permitted baselines is also degraded compared to the results on the shorter sequence $\mathbb{C}_1$. On the other hand, CAD maintains robust performance on the Retain Set while achieving selective forgetting, yielding the highest S score and the lowest MR.

F.3 Effect of AEA in representation space

In this section, we visualize the effect of Anti-Entanglement of Activation (AEA). To this end, we measure the cosine similarity between all possible pairs of output activations produced by each LoRA adapter after completing 10 learning tasks. The output activations are extracted from the final layer

Table 7: Continual Learning performance evaluation of CAD with command sequence $\mathbb{C}_5$.

Methods	Scheme		CIFAR100		ImageNet-R	
	CLU	Buffer	ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)
DER++		500	42.17	-52.64	58.62	-29.78
UniCLUN	✓	500	27.97	-67.12	40.20	-48.40
LwF		0	16.71	-77.47	26.58	-60.29
EWC		0	10.16	-85.90	13.07	-76.04
LSF	✓	0	11.95	-80.72	17.28	-69.72
CLPU	✓	0	43.77	-36.80	67.72	-14.86
InfLoRA		0	59.90	-13.57	73.37	-6.79
CAD	✓	**0**	**61.89**	**-10.95**	**73.65**	**-5.78**

Table 8: Experimental results on DomainNet with baselines

Methods	Scheme		Retain Set		Forget Set			
	CLU	Buffer	ACC_r (↑)	BWT_r (↑)	ACC_f (↓)	BWT_f (↓)	S (↑)	MR (↓)
DER++		500	30.11	-47.61	8.24	-70.01	42.11	51.63
UniCLUN	✓	500	22.30	-53.53	0.33	-77.06	34.58	53.08
LwF		0	35.43	-41.43	2.56	-75.35	48.20	55.26
EWC		0	26.63	-53.11	0.00	-79.45	39.89	43.55
LSF	✓	0	30.69	-43.84	0.60	-75.54	43.65	49.77
InfLoRA		0	47.21	-21.55	0.00	-74.28	57.73	47.81
CLPU	✓	0	38.51	-33.16	0.00	-76.62	51.26	47.82
CAD	✓	**0**	**60.41**	**-6.97**	**0.00**	-73.95	**66.50**	**42.50**

of the network (e.g., the 12th layer), using the dataset from the (randomly chosen) 9-th task (i.e., $\mathcal{D}_9$). Specifically, we compute

$$\cos(i,j) = \frac{\langle \Delta\mathbf{W}_i\mathbf{R}_9 \cdot \Delta\mathbf{W}_j\mathbf{R}_9 \rangle_F}{||\Delta\mathbf{W}_i\mathbf{R}_9||_F||\Delta\mathbf{W}_j\mathbf{R}_9||_F}, \ \forall_{i,j\in 0,1,\ldots,9},$$

where $\langle A, B\rangle_F = \text{tr}(A^\top B)$ is the Frobenius inner product, equivalent to the dot product of the flattened matrices.

Figure 6 shows the heatmap where each entry is indicative of $\cos(i,j)$ with and without the Anti-Entanglement of Activation. Higher values indicate greater overlap in the output representation space, reflecting stronger entanglement. We can observe that introducing the AEA design effectively suppresses inter-task entanglement.

F.4 Effect of AEA on Continual Learning Performance

In this section, we evaluate the pure class-incremental learning performance of CAD in the absence of unlearning operations. To this end, we construct a command sequence $\mathbb{C}_5$ composed solely of learning commands:

$$\mathbb{C}_5 = \{L0, L1, L2, L3, L4, L5, L6, L7, L8, L9\}. \tag{28}$$

Table 7 reports the class incremental learning performance of CAD compared to the baselines on the CIFAR100 and ImageNet-R. Across both benchmarks, CAD consistently achieves strong learnability with minimal forgetting, as reflected by the highest ACC and lowest BWT. We attribute this to the Anti-Entanglement of Activation (AEA) mechanism, which reduces inter-task interference and mitigates the impact of new task learning on previously acquired knowledge, thus further minimizing forgetting.

F.5 Experiments on DomainNet

In Table 2 of the main paper, we evaluated CAD on the CIFAR-100 and ImageNet-R benchmarks. In this section, we present additional results on DomainNet Peng et al. (2018) to further validate the scalability of CAD. DomainNet contains 345 classes, which we partition into 5 disjoint tasks of 69 classes each, forming 5 learning tasks. We then construct the following command sequence:

$$\mathbb{C}_6 = \{L0, L1, L2, L3, L4, U0, U2\}. \tag{29}$$

As shown in Table 8, CAD outperforms all baselines—achieving strong retention on the retain set, exact forgetting on the unlearned tasks, the highest S score, and the lowest membership rate (MR).

F.6 PERFORMANCE VARIATION ACROSS THE COMMAND SEQUENCE

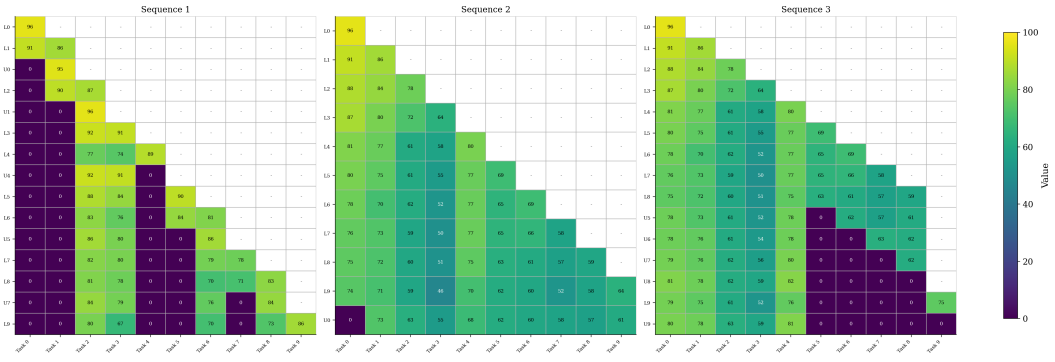


Figure 7: Heat-map illustrating Class-Incremental Learning accuracy of CAD across the three command sequence ($\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3$).

Figure 7 visualizes CAD's task-wise accuracy (acc_i^j) over the full command sequences $\mathbb{C}_1$, $\mathbb{C}_2$, and $\mathbb{C}_3$ on CIFAR100. Here, acc_t^i denotes the inference accuracy on task t after the i-th command. In each heatmap, command sequences $C_i = (I_i, T_i, D_i)$ are shown along the y-axis, and tasks are shown along the x-axis. Notably, immediately following each unlearning command (e.g., $U0$), the corresponding task's accuracy drops to nearly zero – indicating successful unlearning. CAD's unlearning mechanism – based on the dual-branch LoRA architecture combined with Anti-Entanglement of Activation design – effectively prevents the undesirable performance degradation on subsequent tasks when prior tasks are unlearned. We occasionally observe improved performance on some subsequent tasks after unlearning a prior one. This stems from a characteristic of class-incremental learning: once a task is unlearned, its corresponding logit outputs are severely perturbed and become effectively meaningless (e.g., noisy), reducing their influence during inference. As a result, the model is able to focus only on the remaining valid logits, simplifying the prediction space and sometimes improving classification accuracy on retained tasks.

F.7 LAYERWISE EVALUATION OF DECISION SUFFICIENCY

Activation decomposition and decision criterion. Fix a layer and its input $\mathbf{R}_i(x)$ for task i. Under Eq. 2, subsequent adapters do not contribute to this activation. Define the contributions of earlier and current adapters as

$$\mathbf{U}_i(x) = \left(\sum_{k<i} \Delta\mathbf{W}_k\right)\mathbf{R}_i(x), \qquad \mathbf{X}_i(x) = \Delta\mathbf{W}_i\mathbf{R}_i(x). \tag{30}$$

Then $\mathbf{H}_i(x) = \mathbf{W}_0\mathbf{R}_i(x) + \mathbf{U}_i(x) + \mathbf{X}_i(x)$. Using $\mathbf{I} = \mathbf{P}_i + (\mathbf{I} - \mathbf{P}_i)$ gives the exact identity

$$\begin{aligned} \mathbf{H}_i(x) &= \mathbf{Z}_i(x) + \mathbf{E}_i(x), \\ \mathbf{Z}_i(x) &= \mathbf{W}_0\mathbf{R}_i(x) + \mathbf{P}_i\mathbf{U}_i(x) + \mathbf{X}_i(x), \\ \mathbf{E}_i(x) &= (\mathbf{I} - \mathbf{P}_i)\mathbf{U}_i(x). \end{aligned} \tag{31}$$

Each column of $\mathbf{E}_i(x)$ lies in $\mathcal{S}_i^{\perp}$.

Definition 3.2 evaluates the effect of replacing $\mathbf{H}_i$ by $\mathbf{Z}_i$ through the risk of the fixed downstream computation. Under zero-one loss, ϵ_i is the permitted increase in expected classification error on the specified task distribution incurred by this replacement. The inequality is one-sided, so a reduction in error also satisfies the criterion. Even $\epsilon_i = 0$ does not require identical predictions on every example.

Layerwise intervention. We evaluate the empirical counterpart of Definition 3.2 using ViT-Tiny on CIFAR-100 after the learning-only sequence $\mathbb{C}_5 = \{L0, L1, \ldots, L9\}$. For task 9 at each tested layer l, we use the decomposition in Eq. 31, keep the input and all model parameters fixed, and replace $\mathbf{H}_9^{(l)}(x)$ by $\mathbf{Z}_9^{(l)}(x)$ by setting $\mathbf{E}_9^{(l)}(x)$ to zero. Each intervention affects one layer, with the other inputs to the downstream computation held at their original values for each example.

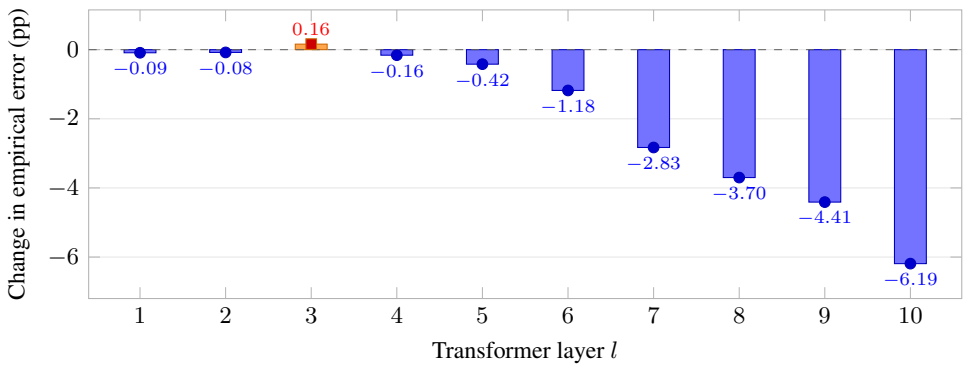


Figure 8: Layerwise residual intervention on task 9 of CIFAR-100 with ViT-Tiny. Each bar corresponds to a separate intervention at one layer. Negative values indicate lower empirical error after removing the residual; the dashed line marks unchanged error.

Let $\mathcal{E}_9$ denote the evaluation examples used in this experiment. For an activation map $\mathbf{V}$, define the empirical risk

$$\widehat{\mathcal{R}}_9^{(l)}(\mathbf{V}) = \frac{1}{|\mathcal{E}_9|} \sum_{(x,y)\in\mathcal{E}_9} \ell_{0\text{-}1}\big(g_x^{(l)}(\mathbf{V}(x)), y\big), \tag{32}$$

where $\ell_{0\text{-}1}$ is zero-one classification loss and $g_x^{(l)}$ is the fixed downstream computation for the intervention at layer l. We report the change in empirical error in percentage points,

$$\widehat{\Delta}_9^{(l)} = 100\left(\widehat{\mathcal{R}}_9^{(l)}(\mathbf{Z}_9^{(l)}) - \widehat{\mathcal{R}}_9^{(l)}(\mathbf{H}_9^{(l)})\right). \tag{33}$$

The smallest nonnegative tolerance satisfying the empirical decision-sufficiency criterion at that layer is

$$\widehat{\epsilon}_9^{(l)} = \frac{\max\{0, \widehat{\Delta}_9^{(l)}\}}{100}. \tag{34}$$

Thus, $\widehat{\epsilon}_9^{(l)}$ is expressed on the zero-one risk scale, while $\widehat{\Delta}_9^{(l)}$ is expressed in percentage points.

Empirical results. Figure 8 shows lower empirical error at nine of the ten tested layers. The largest observed increase is 0.16 percentage points at layer 3, while the largest reduction is 6.19 points at layer 10. Based on the reported values, $\max_{l\in\{1,\dots,10\}} \widehat{\epsilon}_9^{(l)} = 0.0016$. Hence, all ten separate interventions satisfy the empirical version of Definition 3.2 with tolerance 0.0016, and nine satisfy it with tolerance zero.

These observations support decision sufficiency on the evaluated task and layers. The tolerance is computed from the observed error changes.

F.8 Layerwise growth of representation subspaces

To examine the layer partition used by the dual-branch architecture, we measure the average number of newly added GPM basis vectors per task in ViT-B/16 under $\mathcal{C}_1$. Each layer has representation dimension 768. The measured quantity counts new representation directions admitted by the GPM update criterion.

As shown in Figure 9, later layers generally add more basis vectors, increasing from 1.2 at layer 1 to 46.0 at layer 12, with a decrease from layer 10 to layer 11. Under the GPM update criterion, this pattern suggests that early-layer representations are more readily covered by directions stored from previous tasks. The result provides empirical motivation for sharing early layers, while leaving the layer partition as an architectural choice. It does not establish the absence of task-specific information in the shared branch.

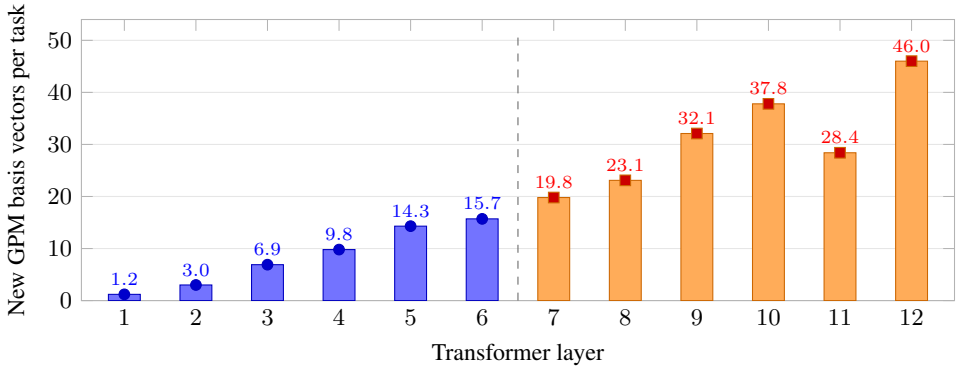


Figure 9: Average number of newly added GPM basis vectors per task for ViT-B/16 under $\mathcal{C}_1$. The dashed line marks the partition at $L = 6$: blue bars correspond to the shared branch and orange bars to the task-exclusive branch. The representation dimension is 768 at every layer.

Table 9: Task-incremental evaluation with task identifiers available at inference. Accuracy is reported in percent and BWT in percentage points; S uses the same percentage scale as in the main experiments. Dashes indicate metrics not reported for the retain-only reference.

Method	$\mathrm{ACC}_r\uparrow$	$\mathrm{BWT}_r\uparrow$	$\mathrm{ACC}_f\downarrow$	$\mathrm{BWT}_f\downarrow$	$S\uparrow$
CIFAR-100					
R-Joint	82.53	—	—	—	—
UniCLUN	56.25	-39.31	27.28	-68.73	59.98
LSF	91.10	-1.82	85.78	-7.97	14.19
CLPU	88.99	-7.30	10.01	-86.41	87.56
PALL	68.73	-6.35	14.01	-62.63	65.11
CAD	94.64	-1.28	10.01	-85.89	90.05
ImageNet-R					
R-Joint	76.86	—	—	—	—
UniCLUN	47.85	-40.48	16.50	-74.05	56.43
LSF	84.80	-1.60	81.90	-6.92	12.56
CLPU	82.87	-5.85	7.17	-84.53	83.59
PALL	59.98	-13.68	8.32	-66.79	62.91
CAD	85.22	-2.31	6.17	-83.11	84.14

F.9 Evaluation with task identifiers at inference

Our primary evaluation uses task-ID-free class-incremental inference. To complement this comparison, we also evaluate the methods with task identifiers available at inference. This task-incremental evaluation helps distinguish performance under task-conditioned inference from the task-ID-free setting targeted by CAD.

Table 9 shows that CAD attains the highest retained accuracy and S on both datasets in this evaluation. Its retained accuracy is 94.64% on CIFAR-100 and 85.22% on ImageNet-R, while forgotten accuracy is 10.01% and 6.17%, respectively. The results also show that the relative ordering depends on the metric: LSF has less negative retained BWT on ImageNet-R, but retains high accuracy on the tasks requested for removal. These task-incremental results complement the task-ID-free comparison rather than replacing it.

F.10 Evaluation with shared semantic superclasses

Disjoint class labels do not imply disjoint representations. To evaluate CAD when tasks share coarse semantic structure, we additionally use a CIFAR-100 superclass split with ViT-Tiny. Tasks contain disjoint fine-grained labels while sharing the same 20 coarse semantic categories. This setting tests retention after task deletion when related semantic features occur across tasks.

Table 10: Results on the CIFAR-100 superclass split with ViT-Tiny. Accuracy and MR are percentages; BWT is in percentage points. MR denotes the member-rate statistic of the specified membership attacker.

Method	$\mathrm{ACC}_r\uparrow$	$\mathrm{BWT}_r\uparrow$	$\mathrm{ACC}_f\downarrow$	$\mathrm{BWT}_f\downarrow$	$S\uparrow$	MR
UniCLUN	20.88	-73.59	10.39	-84.83	32.28	40.52
InfLoRA	76.62	-7.26	0.00	-87.09	80.43	29.08
CLPU	73.01	-13.69	0.00	-89.68	79.06	31.44
PALL	47.47	-14.10	0.72	-67.32	55.16	39.04
CAD	81.24	-2.38	0.00	-86.93	83.59	28.80

In Table 10, CAD improves retained accuracy over InfLoRA and CLPU by 4.62 and 8.23 percentage points, respectively. Its retained BWT is also less negative. All three methods achieve zero forgotten-task accuracy, so their principal difference in this experiment is retained-task utility after the learning and deletion sequence. The result supports the usefulness of CAD when tasks share coarse semantic structure; zero forgotten accuracy alone does not establish removal of all training-data influence.

F.11 Additional task-ID-free baselines

We expand the task-ID-free comparison with a modular expert baseline and a prompt-based continual learner. The modular baseline, denoted OOD, maintains an independent LoRA expert for each task and selects an expert using a Mahalanobis routing score motivated by the scoring approach of Lee et al. (2018). Unlearning removes the requested expert and its routing statistics. This is a custom routed-expert baseline, with no externally supplied task identifier at inference.

We also adapt CODA-Prompt Smith et al. (2023) to the task-level unlearning protocol by deleting the prompt component associated with the requested task together with its corresponding key and attention parameters. This deletion rule is our adaptation of the continual-learning method.

Table 11 shows that CAD improves retained accuracy over OOD by 5.25 points on CIFAR-100 and 8.13 points on ImageNet-R. Relative to the adapted CODA-Prompt baseline, the improvements are 5.86 and 6.27 points, respectively. CAD also attains higher S on both datasets, whereas CODA-Prompt has less negative retained BWT on ImageNet-R. OOD, CODA-Prompt, and CAD all reach zero forgotten-task accuracy. Thus, the final retained accuracy and S distinguish these methods more clearly than forgotten-task accuracy in this evaluation.

The OOD comparison shows that independent modules with inferred task selection are a viable alternative, while CAD attains higher retained accuracy in the tested configurations. The experiment does not separately measure routing error and expert quality, so their individual contributions to the observed gap remain unquantified. Throughout these experiments, MR is an empirical member-rate statistic under the specified attacker, rather than a certificate of privacy or exact unlearning.

F.12 Joint ablation of AEA and basis deletion

To quantify the contributions of AEA and BDS under a common setting, we conduct a 2×2 ablation on ImageNet-R with ViT-B/16 using $\mathcal{C}_1$. Each configuration retains the adapter-detachment mechanism and varies whether AEA and BDS are enabled.

Enabling AEA increases retained accuracy by 2.44 points without BDS and by 2.26 points with BDS. Enabling BDS increases retained accuracy by 0.53 points without AEA and 0.35 points with AEA, while reducing GPM storage from 2.25 to 1.81 MB and from 2.04 to 1.80 MB, respectively. BDS also reduces the final member rate in both comparisons. The full configuration achieves the highest retained accuracy and S, together with the smallest reported GPM footprint; BDS alone has the lowest MR.

All configurations reach zero forgotten-task accuracy, consistent with adapter detachment being the direct mechanism for behavioral task removal. The final-state accuracy improvements reflect the complete sequence of learning and deletion operations. In particular, they do not isolate deletion-induced damage from differences in initial learning performance. Similarly, pruning GPM bases changes subsequent learning constraints; any resulting changes to shared inference weights occur through later training rather than through basis deletion alone.

Table 11: Expanded class-incremental comparison without task identifiers at inference. CIFAR-100 uses ViT-Tiny and ImageNet-R uses ViT-B/16. OOD and CODA-Prompt are evaluated with the task-removal procedures described in the text. Accuracy and MR are percentages; BWT is in percentage points.

Method	$\mathrm{ACC}_r\uparrow$	$\mathrm{BWT}_r\uparrow$	$\mathrm{ACC}_f\downarrow$	$\mathrm{BWT}_f\downarrow$	$S\uparrow$	MR
CIFAR-100 / ViT-Tiny						
R-Joint	82.53	—	—	—	—	—
R-Seq	69.15	-9.60	—	—	—	—
DER	45.73	-49.07	26.17	-69.51	55.12	52.20
UniCLUN	26.88	-68.40	3.77	-92.07	38.75	39.70
PALL	40.34	-30.56	0.05	-73.29	51.75	44.74
LwF	17.38	-77.00	8.55	-86.62	28.18	67.22
EWC	10.53	-85.49	6.83	-89.55	17.91	53.19
LSF	12.45	-80.38	7.10	-86.42	20.56	50.08
InfLoRA	65.50	-13.75	0.00	-83.27	71.82	27.67
CLPU	57.00	-27.33	0.00	-88.00	67.74	38.99
OOD	65.45	-9.32	0.00	-79.87	70.91	42.26
CODA-Prompt	64.84	-11.55	0.00	-80.49	70.65	44.04
CAD	70.70	-8.46	0.00	-83.39	75.67	27.10
ImageNet-R / ViT-B/16						
R-Joint	76.86	—	—	—	—	—
R-Seq	77.46	-4.08	—	—	—	—
DER	62.02	-25.93	42.20	-47.99	52.81	41.89
UniCLUN	29.75	-58.74	3.75	-87.06	40.80	51.85
PALL	45.26	-27.05	0.36	-73.97	56.03	7.55
LwF	36.14	-50.41	15.49	-73.56	46.87	25.70
EWC	13.48	-75.01	8.76	-81.76	21.76	30.98
LSF	17.18	-69.71	12.13	-76.90	26.74	33.33
InfLoRA	73.05	-8.12	0.00	-85.16	78.08	6.04
CLPU	71.63	-11.33	0.00	-86.58	77.83	7.33
OOD	67.38	-6.93	0.00	-81.61	73.43	15.57
CODA-Prompt	69.24	-1.89	0.00	-76.27	72.48	7.30
CAD	75.51	-5.13	0.00	-84.73	79.58	5.24

Table 12: Joint AEA/BDS ablation on ImageNet-R with ViT-B/16 under $\mathcal{C}_1$. GPM storage is measured in MB. Accuracy and MR are percentages; BWT is in percentage points. All metrics are evaluated after the command sequence.

AEA	BDS	$\mathrm{ACC}_r\uparrow$	$\mathrm{BWT}_r\uparrow$	$\mathrm{ACC}_f\downarrow$	$\mathrm{BWT}_f\downarrow$	$S\uparrow$	MR	GPM (MB)
Off	Off	75.62	-3.47	0.00	-85.21	80.13	12.03	2.25
Off	On	76.15	-3.64	0.00	-85.07	80.36	5.19	1.81
On	Off	78.06	-3.23	0.00	-85.41	81.57	10.97	2.04
On	On	78.41	-3.92	0.00	-85.49	81.79	6.63	1.80

F.13 Sensitivity to the LoRA rank

We evaluate LoRA ranks $r \in \{4, 8, 10, 16, 32\}$ on ImageNet-R with ViT-B/16 to examine whether performance depends strongly on the default choice $r = 10$.

Table 13: Sensitivity to LoRA rank on ImageNet-R with ViT-B/16. Accuracy and MR are percentages; BWT is in percentage points.

r	$\mathrm{ACC}_r\uparrow$	$\mathrm{BWT}_r\uparrow$	$\mathrm{ACC}_f\downarrow$	$\mathrm{BWT}_f\downarrow$	$S\uparrow$	MR
4	78.37	-2.72	0.00	-84.55	81.35	4.91
8	78.73	-3.27	0.00	-85.00	81.74	7.01
10	78.41	-3.92	0.00	-85.49	81.79	6.63
16	77.77	-4.57	0.00	-85.56	81.48	5.40
32	77.55	-5.08	0.00	-86.08	81.59	6.37

Table 14: Backbone and per-block LoRA adapter storage for rank $r = 10$.

	ViT-Tiny	**ViT-B/16**
M_W	26.35MB	348.38MB
$M_{\Delta W}$	0.0293MB	0.1172MB

Table 15: Layerwise basis sizes and total storage of the Gradient Projection Memory (GPM) after executing command sequence $\mathbb{C}_1$.

Network	**Basis count × representation dimension**	M_{GPM}
ViT-Tiny	9 × 192, 14 × 192, 25 × 192, 42 × 192, 62 × 192, 71 × 192, 85 × 192, 98 × 192, 67 × 192, 87 × 192, 98 × 192, 89 × 192	0.547MB
ViT-B/16	7 × 768, 14 × 768, 26 × 768, 30 × 768, 42 × 768, 41 × 768, 46 × 768, 50 × 768, 80 × 768, 82 × 768, 31 × 768, 82 × 768	1.55MB

Across the tested ranks, retained accuracy varies from 77.55% to 78.73%, a range of 1.18 percentage points, while S varies by 0.44 points. Forgotten-task accuracy remains zero throughout. The best retained accuracy occurs at $r = 8$, whereas the highest S occurs at $r = 10$. These results show modest variation within the evaluated range and support $r = 10$ as a representative operating point.

F.14 Memory Profile

We analyze CAD's storage requirements by separating the backbone, LoRA adapters, and gradient-projection memory, then compare with UniCLUN and examine the effects of deletion depth and command-sequence length.

Storage components. Let N be the number of transformer blocks, L the number of shared (task-agnostic) blocks, and T the number of retained tasks. CAD's total storage is

$$M_{\mathrm{total}} = M_W + M_{\Delta W} L + M_{\Delta W}(N - L)T + M_{\mathrm{GPM}}, \tag{35}$$

where M_W is backbone storage, $M_{\Delta W}$ is the storage of the LoRA adapters in one transformer block, and M_{GPM} is the storage of the Gradient Projection Memory (GPM). The two adapter terms account for the shared blocks and the task-exclusive blocks, respectively. Both ViT-Tiny and ViT-B/16 have $N = 12$ blocks. Unless varied explicitly, we use deletion depth $L = 6$ and LoRA rank $r = 10$, matching the main experiments. After $\mathbb{C}_1$, $T = 5$ tasks remain.

Table 14 reports backbone and per-block adapter storage, and Table 15 reports the layerwise GPM basis sizes after $\mathbb{C}_1$. GPM occupies 0.547 MB for ViT-Tiny and 1.55 MB for ViT-B/16. Including the backbone and adapters, CAD's reported total storage is 27.95 MB and 354.14 MB, respectively.

Comparison with UniCLUN. For UniCLUN Chatterjee et al. (2024), the storage accounting includes the backbone, adapters for the good teacher, bad teacher, and student across all N blocks, and a rehearsal buffer:

$$M_{\mathrm{UniCLUN}} = M_W + N\big(M_{\Delta W_{\mathrm{GT}}} + M_{\Delta W_{\mathrm{BT}}} + M_{\Delta W_{\mathrm{S}}}\big) + M_{\mathrm{buffer}}.$$

Here, $M_{\Delta W_{\mathrm{GT}}}$, $M_{\Delta W_{\mathrm{BT}}}$, and $M_{\Delta W_{\mathrm{S}}}$ denote the per-block adapter storage of the good teacher, bad teacher, and student, respectively, and M_{buffer} denotes rehearsal-buffer storage. The bad-teacher formulation Chundawat et al. (2023a) used by UniCLUN permits a lightweight network or a random predictor. For a conservative storage comparison, we therefore set $M_{\Delta W_{\mathrm{BT}}} = 0$ and use $M_{\Delta W_{\mathrm{GT}}} = M_{\Delta W_{\mathrm{S}}} = M_{\Delta W}$. For a buffer of 500 samples, the reported buffer storage is 1.50 MB for CIFAR-100 and 71.78 MB for ImageNet-R. The resulting UniCLUN totals are 28.55 MB for ViT-Tiny and 422.97 MB for ViT-B/16. Table 16 compares these totals with CAD. CAD uses less storage in both configurations, with a larger difference on ImageNet-R because storing its higher-resolution replay samples is more costly.

Effect of deletion depth. Increasing L shares more adapter blocks and reduces the number stored separately for each retained task.

Table 16: Storage after executing command sequence $\mathbb{C}_1$. CAD uses deletion depth $L = 6$ and LoRA rank $r = 10$; UniCLUN uses a replay buffer of 500 samples.

	ViT-Tiny	**ViT-B/16**
UniCLUN	28.55MB	422.97MB
CAD	**27.95MB**	**354.14MB**

Table 17: ViT-B/16 storage across deletion depths under $\mathbb{C}_1$. The UniCLUN reference uses a replay buffer of 500 samples.

Deletion depth L	0	2	4	6	8
CAD storage (MB)	356.95	356.01	355.07	354.14	353.20
UniCLUN reference: 422.97 MB					

Table 17 shows a reduction of 3.75 MB between $L = 0$ and $L = 8$. Thus, deletion depth has a modest effect on total storage within this range, where the backbone dominates the memory footprint. The corresponding retention and deletion-fidelity trade-offs should also be considered when choosing L.

Effect of sequence length. We next examine $\mathbb{C}_4$, which contains 20 learning and 10 unlearning commands and ends with $T = 10$ retained tasks. We use the same rank $r = 10$ and deletion depth $L = 6$ as in the main configuration.

Under $\mathbb{C}_4$, the reported storage is 358.11 MB, comprising 348.38 MB for the backbone, 7.72 MB for adapters, and 2.01 MB for GPM. This is 3.97 MB above the $\mathbb{C}_1$ footprint and 64.86 MB below the UniCLUN reference. The backbone accounts for most of the total storage in both sequences, while adapter storage grows with the number of retained tasks according to Equation 35. The observed increase is modest over the tested sequences, but the storage requirement is not constant for arbitrarily long streams.