WHEN PRESERVATION BECOMES STALE: CONTINUAL FUNCTIONAL REFACTORING FOR FIXED-CAPACITY CLIP CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual adaptation of pretrained vision-language models under a fixed parameter budget requires acquiring new knowledge without historical replay or task-specific capacity growth. Many existing methods protect historically important structures and constrain subsequent updates around them, effectively creating persistent capacity ownership. Through longitudinal subspace interventions, we identify Historical Necessity Drift: the functional necessity of historical subspaces can weaken, strengthen, or migrate as new tasks are learned. Directions once critical can become redundant, secondary directions can become essential, and others remain necessary throughout. This drift creates stale capacity ownership, where structure remains protected after its original functional role has diminished or migrated. More strikingly, retaining only a compact, functionally selected subspace can improve historical-task accuracy over the full adapter, showing that historical performance degradation can coexist with recoverable useful function and exposing functional interference from stale organization. Motivated by this finding, we propose Continual Functional Refactoring (CFR), which summarizes accumulated behavior with a fixed-size functional state, consolidates shared functionality into a compact core while retaining recent residuals in its complement, and reopens the full adapter for subsequent learning. Across CIFAR-100 and TinyImageNet sequences spanning 5 to 50 tasks, CFR ranks first in 11 of the 12 reported Average/Last accuracy evaluations. Its advantage is particularly evident over long horizons: on CIFAR-100 with 50 tasks, CFR achieves 84.96% Average and 76.74% Last accuracy, exceeding the strongest competing Last result by 1.08 percentage points. Code will be made publicly available upon acceptance.

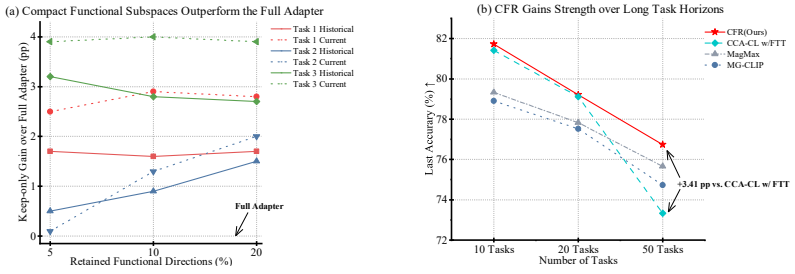


Figure 1: Functional interference and long-horizon behavior. (a) Compact functionally selected subspaces consistently outperform the full adapter under keep-only evaluation. (b) CFR's Last-accuracy advantage becomes most pronounced as the task horizon grows.

1 INTRODUCTION

Continual adaptation of pretrained vision-language models Radford et al. (2021) must absorb new knowledge within a fixed adaptation budget, without historical replay or task-specific capacity growth. Existing formulations largely treat this as a capacity-allocation problem: a finite set of adaptable parameters must support an expanding body of knowledge Liang & Li (2024); Luo et al.

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.

(2026); Qiu et al. (2026). Yet fixed capacity need not imply fixed functional organization. As new tasks reshape shared representations, the subspaces realizing earlier functions may become redundant, remain necessary, or take on new roles. This raises a central question: how can accumulated function be preserved when its parameter-space realization keeps changing?

Most existing approaches protect historically important structure and restrict later updates to low-interference directions, through subspace isolation, null-space projection, gradient decomposition, or parameter-efficient adaptationKang et al. (2025); Jo & Kim (2026). Despite their different implementations, they share a protection-oriented premise: structure important to past tasks should remain protected while new knowledge is written elsewhere. This premise improves stability but leaves open whether historical structures should retain the same functional responsibility as the model evolves.

We show that historical functional dependence is non-stationary. Directions indispensable when first learned can later become replaceable as their functions migrate elsewhere; previously secondary directions can become critical, while others remain persistently necessary. We term this continual reassignment Historical Necessity Drift. More importantly, as previewed in fig. 1(a), keep-only interventions reveal that compact functionally selected subspaces consistently outperform the full adapter across the examined retention ratios. Thus, historical performance degradation can coexist with recoverable useful function, exposing functional interference from stale organization rather than functional loss alone. Under fixed capacity, protecting such stale ownership can constrain future adaptation.

These findings motivate a functional view of continuity: preserve accumulated behavior while allowing its realization to move. We therefore propose Continual Functional Refactoring (CFR), a replay-free framework that periodically reorganizes accumulated function within the same fixed adapter budget. CFR summarizes accumulated behavior with a fixed-size functional state, consolidates shared functionality into a compact core, retains recent residuals in the complement, and reopens the full adapter for the next task. This learn–refactor–reopen cycle replaces permanent structural ownership with temporary functional organization, allowing the same capacity to be continually repurposed.

Across CIFAR-100 Krizhevsky et al. (2009) and TinyImageNet Le et al. (2015) sequences spanning 5 to 50 tasks, CFR ranks first in 11 of 12 reported Average/Last evaluations. Its advantage is most pronounced under long horizons: on CIFAR-100 with 50 tasks, CFR reaches 84.96% Average and 76.74% Last accuracy, improving the strongest competing Last result by 1.08%. Ablations further attribute the gains to function-aware refactoring, recent-residual preservation, and full reopening rather than additional capacity or a particular core dimension. Our main contributions are:

- We characterize Historical Necessity Drift through longitudinal functional-subspace interventions, showing that dependence on historical structure can weaken, strengthen, or migrate as new knowledge is acquired.
- We identify stale preservation and functional interference as distinct sources of rigidity in fixed-capacity continual learning. Keep-only interventions reveal that useful historical function can remain recoverable despite degraded full-model performance, while stronger preservation can even reduce retention by maintaining stale functional organization.
- We introduce CFR, which reorganizes accumulated behavior with a fixed-size functional state, compact core–complement refactoring, and full reopening. Experiments across 5–50 task sequences show consistent gains, with the clearest advantage under long task horizons.

2 Related Work

2.1 Continual Adaption of Vision-Language Models

Pretrained vision-language models (VLMs), particularly CLIP, provide strong transferable representations for continual learning, yet sequential adaptation can degrade both acquired capabilities and pretrained generalization Radford et al. (2021). Existing approaches mitigate this degradation through regularization, distillation, representation preservation, and parameter-efficient adaptation. More recently, subspace-based methods directly constrain where updates occur: DMNSP projects visual updates into dynamically constructed null spaces Kang et al. (2025); KeepLoRA separates

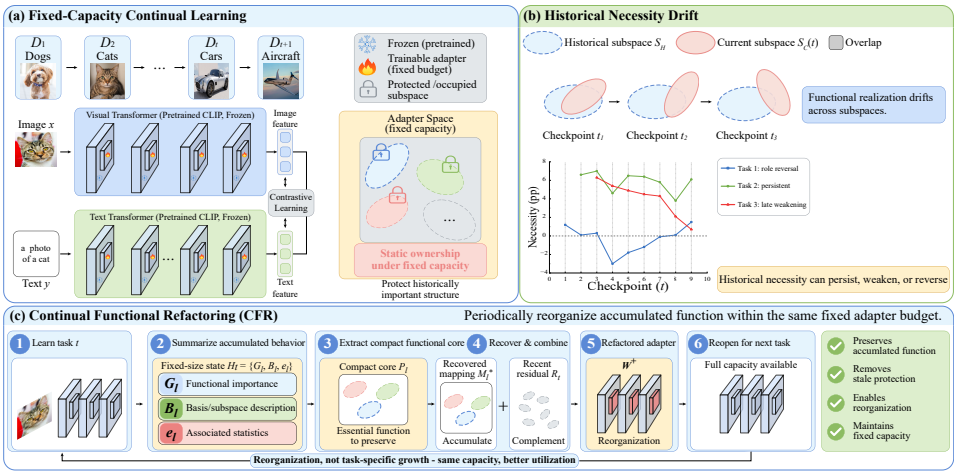


Figure 2: Historical Necessity Drift and Continual Functional Refactoring (CFR). (a) A fixed-capacity adapter is reused across tasks, whereas conventional protection reserves historical directions. (b) Historical necessity changes as task functions migrate across subspaces. (c) CFR uses a fixed-size functional state to consolidate shared functionality into a compact core, preserve recent residuals, and reopen the full adapter.

principal and residual directions for low-interference learning Luo et al. (2026); SplitLoRA partitions historical gradient space to balance stability and plasticity Qiu et al. (2026); and CoSO Cheng et al. (2026) constructs low-rank optimization spaces orthogonal to historical directions. Despite different implementations, these methods share a common objective: limiting interference by preserving or avoiding historically important structure.

2.2 Dynamic Capacity Management and Functional Reorganization

Recent studies have relaxed the assumption that importance and adaptation capacity remain static throughout continual learning. Evolving Parameter Isolation periodically updates isolation masks as parameter importance changes Lin et al. (2026), while Dynamic Orthogonal Continual Fine-tuning tracks drifting historical functional directions and regularizes new-task gradients against them Zhang et al. (2025). Other approaches adapt the representation of accumulated knowledge itself: Share Kaushik et al. (2026) incrementally integrates task-relevant directions into a shared low-rank subspace, continual merging consolidates sequential LoRA updates into a single fixed-size adapter, and Neural Subspace Reallocation manages low-rank task memories through compression, retrieval, and reallocation Qiao & Mahdavi (2026). These methods establish that protection criteria, optimization subspaces, and capacity organization can evolve during sequential learning.

Our question is different: rather than asking how importance estimates, optimization constraints, or stored subspaces should evolve, we ask whether a structure identified as important for a past task remains functionally necessary for that task after subsequent learning. We address this through longitudinal remove/keep interventions that directly measure the behavioral dependence of the current model on historical subspaces. This reveals Historical Necessity Drift: historical necessity can weaken, strengthen, or migrate even when the corresponding functionality remains available elsewhere, and full-adapter degradation can coexist with recoverable historical function. The resulting problem is therefore not only how to update or allocate capacity, but when historical capacity ownership has become stale.

3 Historical Necessity Drift in Fixed-Capacity Learning

3.1 Tracking Functional Dependence With Subspace Interventions

To track how historical functional dependence changes across checkpoints, we require task-relevant structure that is both comparable over time and directly tied to model behavior. We therefore operate

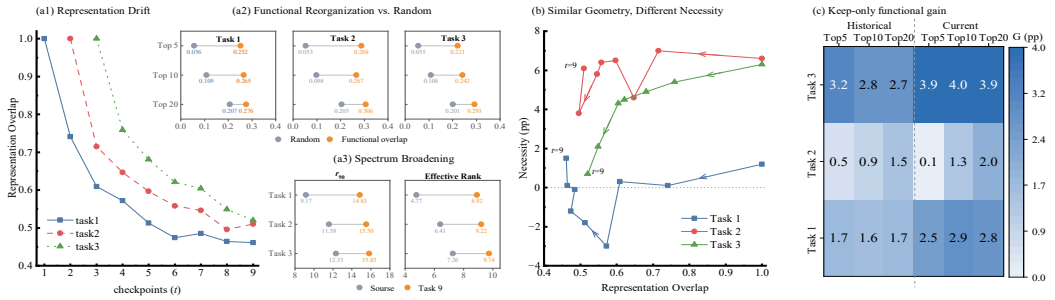


Figure 3: Diagnosing Historical Necessity Drift. (a) Adapter representations and functional subspaces reorganize across checkpoints, accompanied by spectrum broadening. (b) Similar representation overlap can correspond to markedly different necessity trajectories, showing that geometric drift alone does not determine functional dependence. (c) Keeping only the Top-5/10/20% functionally selected historical or current directions consistently improves historical-task accuracy over the full adapter, revealing interference from the remaining adapter directions.

in the shared adapter space and identify functional subspaces through behavioral intervention rather than activation geometry alone Raghu et al. (2017); Kornblith et al. (2019).

Consider a sequence of T class-incremental tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, with $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$. After task t, the model is evaluated over all classes observed so far. The pretrained CLIP backbone remains frozen and the same adapters are updated throughout, keeping the adaptation budget fixed, $|\Theta_A^{(t)}| = C$ for all t. Held-out samples from earlier tasks are used only for diagnosis to probe how historical functions are realized at later checkpoints.

Each of the $L = 12$ visual Transformer blocks contains an $r = 64$ bottleneck adapter Chen et al. (2022),

$$h_\ell(x) = \phi\left(x_\ell W_{\text{down}}^\ell\right) \in \mathbb{R}^r, \quad \Delta x_\ell = h_\ell(x) W_{\text{up}}^\ell, \tag{1}$$

where $W_\ell^{\text{down}} \in \mathbb{R}^{768\times 64}$ and $W_\ell^{\text{up}} \in \mathbb{R}^{64\times 768}$. Because all tasks reuse the same bottleneck space, it provides a fixed-dimensional substrate for tracing functional dependence over time.

For a source task s evaluated at checkpoint $t \geq s$, we collect bottleneck activations from a fixed diagnostic set $\mathcal{D}_s^{\text{diag}}$,

$$H_{s,\ell}^{(t)} = [h_\ell(x_i)]_{i=1}^N \in \mathbb{R}^{N\times r}. \tag{2}$$

An SVD of $H_{s,\ell}^{(t)}$ yields orthonormal candidate directions $V_{s,\ell}^{(t)}$. Since activation magnitude alone does not establish functional importance, we rank each direction by the behavioral effect of removing it:

$$I_{s,\ell,j}^{(t)} = \mathcal{L}_s\left(\theta_t; \text{Remove}(v_{s,\ell,j}^{(t)})\right) - \mathcal{L}_s(\theta_t). \tag{3}$$

Larger positive $I_{s,\ell,j}^{(t)}$ indicates stronger current dependence. We select the top fraction q across all Lr layer-direction candidates and denote their layer-wise spans by $S_s^{(t,q)}$. The historical subspace $S_s^H = S_s^{(s,q)}$ is identified when task s is learned, whereas $S_s^{C(t)} = S_s^{(t,q)}$ is re-estimated at checkpoint t.

Given a functional subspace S, let $Q_{s,\ell}$ be an orthonormal basis with projector $P_{s,\ell} = Q_{s,\ell}Q_{s,\ell}^\top$. We define complementary remove and keep interventions together with two behavioral scores:

$$\begin{aligned} h_\ell^{\text{remove}} &= h_\ell(I - P_{s,\ell}), & h_\ell^{\text{keep}} &= h_\ell P_{s,\ell}, \\ N_s^{(t)}(S) &= A_s^{\text{full}}(t) - A_s^{\text{remove}}(t;S), & G_s^{(t)}(S) &= A_s^{\text{keep}}(t;S) - A_s^{\text{full}}(t). \end{aligned} \tag{4}$$

Here, $N_s^{(t)}(S) > 0$ indicates that the current model still requires S for task s, whereas $G_s^{(t)}(S) \geq 0$ indicates that retaining the selected adapter directions alone matches or exceeds the full-adapter accuracy. Thus, N measures current indispensability, while G tests whether the same functional realization remains effective in isolation.

3.2 Historical Necessity Drifts Over Time

For each source task s, we fix its historical functional subspace S_s^H and evaluate $N_s^{(t)}(S_s^H)$ at every later checkpoint $t \geq s$, while independently re-estimating the current functional subspace $S_s^{C(t)}$ from the same diagnostic data. This separates persistence of a historical realization from migration of the function it once supported.

The resulting necessity trajectories are neither monotonic nor governed by a common decay pattern. Historical subspaces may become replaceable or interfering, recover after intermediate weakening, or remain persistently necessary. We term this task-dependent reassignment of functional necessity *Historical Necessity Drift*. It is therefore not simply the gradual erosion of historical structure: subsequent learning continually changes which historical directions the current model behaviorally depends on.

We further quantify how the realization itself moves. Let $Q_{s,\ell}^H$ and $Q_{s,\ell}^{C(t)}$ denote orthonormal bases of the historical and current functional subspaces. Their normalized overlap is

$$\Omega_{s,t} = \frac{1}{L} \sum_{\ell=1}^{L} \frac{\left\| Q_{s,\ell}^{H\top} Q_{s,\ell}^{C(t)} \right\|_F^2}{\min\left(\dim S_{s,\ell}^H,\, \dim S_{s,\ell}^{C(t)}\right)}. \tag{5}$$

Across the examined tasks, historical–current overlap eventually falls to 0.24–0.27, confirming substantial migration of the functional realization. Yet similar geometric change does not imply similar functional consequences: some historical subspaces remain necessary, others become replaceable, and some reverse their effect.

Although the bottleneck spectrum broadens and effective rank increases, tasks with similar late-stage representation overlap $(0.46 - 0.52)$ still exhibit sharply different necessity trajectories. Geometry therefore captures representational change but not functional requirement: functional importance is a time-dependent relation between a function and the current model state, rather than a persistent property of a historical subspace.

3.3 Beyond Forgetting: When Preservation Becomes Stale

A decline in historical-task accuracy does not necessarily imply the loss of useful historical function. As shown in Figure 3(c), keeping only the Top-5/10/20% functionally selected historical or current subspaces consistently outperforms the full adapter. For Task 3, the Top-10% historical/current subspaces increase accuracy from 75.8% to 78.6%/79.8%, indicating that useful adapter corrections remain concentrated in selected directions while the remainder introduces net interference. Functional usefulness can also diverge from indispensability: a historical subspace may satisfy $N_s^{(t)}(S_s^H) \approx 0$ while retaining $G_s^{(t)}(S_s^H) > 0$, remaining effective even after the current model no longer uniquely depends on it.

We call this mismatch stale preservation: historically important structure remains protected after its functional responsibility has migrated, become replaceable, or turned interfering, thereby maintaining outdated capacity ownership under a fixed budget. Yet replaceability for one historical task does not make a subspace globally disposable, since later tasks may have repurposed it. The appropriate objective is therefore reorganization rather than pruning: preserve necessary functionality, consolidate substitutable realizations, and reopen capacity that no longer requires permanent historical ownership.

4 Continual Functional Refactoring

Section 3 shows that functional continuity does not require structural permanence: accumulated function can persist even as its parameter-space realization changes or becomes replaceable. This motivates CFR, which preserves accumulated behavior while allowing its realization to be periodically reorganized within the same fixed adaptation budget.

We operationalize this view through Continual Functional Refactoring (CFR). Let θ_{t-1}^+ denote the previously refactored model. Learning $\mathcal{T}_t$ yields the pre-refactoring state θ_t^-, which CFR reorganizes

into θ_t^+:

$$\theta_{t-1}^+ \xrightarrow{D_t} \theta_t \xrightarrow{\text{CFR}} \theta_t^+. \tag{6}$$

Rather than accumulating task-owned parameter regions, CFR maintains a fixed-size state of accumulated adapter behavior. At each task boundary, this state is updated in the current model coordinates, used to refactor accumulated function, and followed by full reopening of the adapter. This yields a recurring learn–refactor–reopen cycle under a fixed adaptation budget.

4.1 A Fixed-size State For Accumulated Function

Historical Necessity Drift makes parameter identity an unreliable preservation target, since the same function may migrate across adapter directions as learning proceeds. CFR therefore summarizes accumulated knowledge through adapter input–output behavior rather than historical parameters.

For visual block ℓ, let $h_\ell \in \mathbb{R}^r$ be the bottleneck activation and $z_\ell \in \mathbb{R}^d$ the corresponding up-projection output. We augment h_ℓ with a constant coordinate, $\bar{h}_\ell = [h_\ell; 1]$, so that the up-projection weight and bias are represented jointly by an affine map M_ℓ. For each block, CFR summarizes accumulated functionality using three second-order statistics: $G_\ell = \mathbb{E}\left[\bar{h}_\ell \bar{h}_\ell^\top\right]$, $B_\ell = \mathbb{E}\left[z_\ell \bar{h}_\ell^\top\right]$ and $e_\ell = \mathbb{E}\left[\|z_\ell\|_2^2\right]$. The collection over all blocks forms the accumulated functional state $\mathcal{H}_t = \{G_\ell, B_\ell, e_\ell\}_{\ell=1}^L$. Its size depends only on the adapter architecture and therefore remains constant with sequence length. Each task summary is normalized before aggregation to avoid domination by activation-scale differences.

These statistics are sufficient to evaluate the expected squared discrepancy of any candidate affine realization without replaying historical samples. For a candidate map M_ℓ,

$$\mathcal{E}_\ell(M_\ell; \mathcal{H}_t) = e_\ell - 2\langle M_\ell, B_\ell \rangle + \operatorname{tr}\left(M_\ell G_\ell M_\ell^\top\right). \tag{7}$$

4.2 Core–Complement Functional Refactoring

Given the accumulated state $\mathcal{H}_t$, we extract the bottleneck geometry $G_\ell^h \in \mathbb{R}^{r\times r}$ from G_ℓ and eigendecompose

$$G_\ell^h = U_\ell \Lambda_\ell U_\ell^\top. \tag{8}$$

The k dominant directions define the core projector

$$P_\ell = U_{\ell,k} U_{\ell,k}^\top. \tag{9}$$

We use $k = 16$ in the main configuration, selecting a compact support within the $r = 64$ bottleneck. Crucially, the geometry determines where the core lies, but not what behavior it should realize. We therefore recover the affine mapping from the accumulated input–output state:

$$M_\ell^\star = \arg\min_M \mathcal{E}_\ell(M; \mathcal{H}_t) + \lambda \|M\|_F^2. \tag{10}$$

where λ is a small ridge coefficient. Let $W_\ell^\star$ and $b_\ell^\star$ denote the resulting weight and bias. The core realization is $W_\ell^\star P_\ell$.

Thus, P_ℓ determines the compact support, while $\mathcal{H}_t$ determines the behavior realized on it. Since $\mathcal{H}_t$ aggregates all tasks up to t, the core represents accumulated rather than task-specific behavior.

Immediately after learning task t, newly acquired functionality may still lie outside P_ℓ and has not yet been integrated into the shared core. CFR therefore preserves this recent residual in the complementary space.

Let W_ℓ^- denote the pre-refactoring up-projection and $P_\ell^\perp = I - P_\ell$. CFR forms the refactored mapping

$$W_\ell^+ = W_\ell^* P_\ell + W_\ell^- P_\ell^\perp, \qquad b_\ell^+ = b_\ell^*. \tag{11}$$

The two terms serve complementary roles: the core reconstructs accumulated behavior from $\mathcal{H}_t$, whereas the complement preserves functionality learned most recently from θ_t^-. Both remain within the same r-dimensional adapter, so refactoring reorganizes functionality within a fixed capacity rather than reducing the adaptation space.

Table 1: Comparison on TinyImageNet under class-incremental learning. Average/Last accuracy (%) is averaged over three seeds; best/second-best results are shown in **bold**/underline.

Method	Venue	5 Tasks		10 Tasks		20 Tasks	
		Average ↑	Last ↑	Average ↑	Last ↑	Average ↑	Last ↑
CLIP Zero-shot	ICML'21	71.61	64.38	73.00	64.38	74.03	64.18
Continual-CLIP	arXiv'22	73.17	65.97	74.62	65.97	75.52	65.97
MagMax	ECCV'24	82.48	79.09	80.14	75.68	77.93	73.75
RAPF	ECCV'24	81.52	75.88	82.10	75.26	82.49	74.59
MoE4Adapters	CVPR'24	83.87	77.68	83.25	76.29	82.77	75.26
MG-CLIP	ICCV'25	84.43	78.10	83.82	<u>76.93</u>	<u>83.55</u>	<u>76.48</u>
DMNSP	ICCV'25	<u>84.52</u>	<u>78.15</u>	<u>83.89</u>	76.16	82.49	74.59
PROOF	TPAMI'25	76.18	68.36	75.95	67.85	75.26	67.38
CCA-CL w/ FTT	CVPR'26	82.89	76.16	83.67	76.47	82.36	74.17
CCA-CL w/o FTT	CVPR'26	77.93	70.53	79.33	70.49	79.77	70.48
CFR (Ours)	–	**85.77**	**80.09**	**85.12**	**78.25**	**84.17**	**76.67**

4.3 Temporary Organization And Full Reopening

Permanently freezing the compact core would gradually partition the adapter into historical and available regions, recreating the static ownership exposed by Historical Necessity Drift. CFR therefore treats the core–complement decomposition as a transient task-boundary organization rather than a persistent parameter partition.

After refactoring, no component of the adapter is frozen. Task $t+1$ is learned from the full refactored model

$$\theta_t^+ \xrightarrow{D_{t+1}} \theta_{t+1}^-. \tag{12}$$

without restricting updates to $P_\ell^\perp$ or allocating a task-specific module. All $r = 64$ bottleneck dimensions therefore return to the trainable adaptation space.

Released capacity means released ownership rather than empty coordinates: directions may still carry useful residual functionality, but they are no longer permanently reserved by historical importance and can be repurposed by subsequent tasks.

The core is recomputed at every task boundary from the updated functional state, so $P_\ell^{(1)}, P_\ell^{(2)}, \ldots$ represent successive checkpoint-specific realizations rather than progressively protected regions. Throughout the sequence, both the adapter budget and $\mathcal{H}_t$ remain fixed in size, allowing the same capacity to be repeatedly reorganized as task transitions accumulate.

5 Experiment

5.1 Experimental Setup

We evaluate CFR under class-incremental learning on CIFAR-100 Krizhevsky et al. (2009) with 10/20/50 tasks and TinyImageNet Le et al. (2015) with 5/10/20 tasks. At checkpoint t, evaluation covers all classes observed so far. These splits span short to long task horizons while keeping the adaptation budget fixed.

We use a frozen CLIP ViT-B/16 backbone with $r = 64$ bottleneck adapters inserted into all 12 Transformer layers of both the visual and text encoders. Both adapter branches are optimized during sequential learning, while CFR is applied only to the visual adapters. Unless otherwise specified, CFR uses $k = 16$, performs refactoring at each task boundary, and reopens all 64 visual bottleneck dimensions for subsequent learning. No historical replay or task-specific parameter growth is used, and all results are averaged over the same three random seeds.

We compare CFR with CLIP Zero-shot Radford et al. (2021), Continual-CLIP Ding et al. (2022), MagMax Marczak et al. (2024), RAPF Huang et al. (2024), MoE4Adapters Yu et al. (2024), MG-CLIPHuang et al. (2025), DMNSPKang et al. (2025), PROOFZhou et al. (2025), and CCA-CLZhang

Table 2: Comparison on CIFAR-100 under class-incremental learning. Average/Last accuracy (%) is averaged over three seeds; best/second-best results are shown in **bold**/underline.

Method	Venue	10 Tasks		20 Tasks		50 Tasks	
		Average ↑	Last ↑	Average ↑	Last ↑	Average ↑	Last ↑
CLIP Zero-shot	ICML'21	75.17	66.68	75.94	66.68	76.94	66.68
Continual-CLIP	arXiv'22	75.15	66.69	75.94	66.69	76.48	66.69
MagMax	ECCV'24	83.74	79.33	81.49	77.83	78.81	75.66
RAPF	ECCV'24	85.91	79.10	85.90	77.09	83.29	72.69
MoE4Adapters	CVPR'24	85.67	77.12	84.87	77.16	83.98	74.68
MG-CLIP	ICCV'25	86.67	78.90	84.32	77.52	84.70	74.73
DMNSP	ICCV'25	87.03	79.16	84.89	76.18	83.66	73.91
PROOF	TPAMI'25	83.74	79.83	79.15	70.57	76.26	69.01
CCA-CL w/ FTT	CVPR'26	88.05	81.42	**87.22**	79.11	82.36	73.33
CCA-CL w/o FTT	CVPR'26	84.08	75.90	84.11	75.02	81.20	72.06
CFR (Ours)	–	**88.30**	**81.72**	86.46	**79.21**	**84.96**	**76.74**

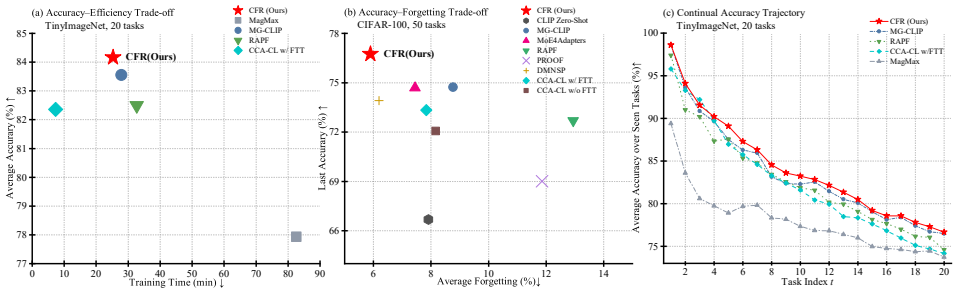


Figure 4: Efficiency and continual-learning dynamics. (a) Average accuracy versus training time on TinyImageNet-20. (b) Last accuracy versus Average Forgetting on CIFAR-100-50. (c) Seen-class accuracy across the TinyImageNet-20 task sequence. CFR combines moderate training cost with strong long-horizon retention and sustained accuracy across task boundaries.

et al. (2026) with and without first-task fine-tuning (FTT). We report Average accuracy over the continual-learning trajectory and Last accuracy after the final task, with higher values indicating better performance.

5.2 Main Results across Task Horizons

Table 1 and table 2 compare CFR across task horizons spanning 5 to 50 tasks. CFR ranks first in 11 of the 12 reported Average/Last evaluations. On TinyImageNet, it achieves the best result in all six metrics across the 5-, 10-, and 20-task settings, outperforming the strongest competing result by 0.19–1.32 percentage points. Thus, the benefit of functional refactoring is already evident over short and medium horizons.

The key trend emerges as the task horizon grows on CIFAR-100. Under 10 tasks, CFR slightly surpasses the strong CCA-CL w/ FTT baseline, reaching $88.30/81.72$ versus $88.05/81.42$ in Average/Last accuracy. At 20 tasks, the two remain closely matched: CFR is 0.76 points lower in Average accuracy but 0.10 points higher in Last accuracy. At 50 tasks, however, the gap widens sharply. CFR reaches 84.96% Average and 76.74% Last accuracy, exceeding CCA-CL w/ FTT by 2.60 and 3.41 points, respectively. Even when compared with the strongest competing result for each metric, CFR remains ahead by 0.26 points in Average and 1.08 points in Last accuracy. The notable result is therefore not merely strong endpoint performance, but the transition from near parity at shorter horizons to a clear separation after repeated task transitions.

This horizon-dependent separation directly reflects the challenge targeted by CFR. As task boundaries accumulate, the same fixed adapter must repeatedly accommodate changing functional demands, increasing the pressure imposed by persistent historical capacity ownership. CFR instead consolidates accumulated behavior, releases stale ownership, and reopens the full adapter for sub-

Table 3: Ablation and sensitivity analysis on TinyImageNet with 20 tasks. All variants share the same CLIP ViT-B/16 backbone, $r = 64$ bottleneck adapters, training protocol, and fixed adaptation budget. Higher accuracy and lower forgetting are better. The full CFR uses $k = 16$.

Variant	k	Accuracy (%) ↑		Forgetting (%) ↓	
		Avg.	Last	Avg.	Last
Mechanism ablations					
Shared Adapter / No Refactor	–	82.96	73.84	9.44	13.74
Random-Core CFR	16	83.64	75.38	8.36	12.08
CFR w/o Recent Residual	16	83.86	76.00	5.34	8.01
CFR w/o Reopening	16	83.76	76.12	**5.18**	**7.76**
CFR (Full)	**16**	**84.17**	**76.67**	5.51	7.87
Core-size sensitivity					
CFR	8	84.19	76.45	5.61	8.57
CFR	32	84.20	76.42	5.44	8.16

sequent reuse. The substantially larger margin at 50 tasks is consistent with this mechanism: repeated functional reorganization becomes increasingly consequential as more knowledge must coexist within the same fixed capacity. At the same time, CFR retains strong short-horizon performance, indicating that this long-sequence advantage is achieved without trading away effectiveness earlier in the stream.

5.3 Efficiency and continual dynamics

Figure 4 summarizes efficiency and continual-learning dynamics. On TinyImageNet-20, CFR achieves 84.17% Average accuracy at moderate training cost, outperforming MG-CLIP and RAPF while remaining slower than lightweight CCA-CL. On CIFAR-100-50, CFR combines the highest Last accuracy (76.74%) with one of the lowest Average Forgetting values. Its seen-class trajectory on TinyImageNet-20 is also strongest at nearly all task boundaries. Together, these results show CFR maintains its advantage throughout continual learning rather than only at the final checkpoint.

5.4 Ablations and Sensitivity

Table 3 isolates the main components of CFR. Removing refactoring reduces Average/Last accuracy from $84.17/76.67\%$ to $82.96/73.84\%$, while replacing the functional core with a random subspace yields $83.64/75.38\%$, indicating that the gain depends on function-aware organization rather than low-dimensional reorganization alone. Removing the recent residual further lowers performance to $83.86/76.00\%$, supporting its role in preserving newly acquired functionality that has not yet been consolidated into the core. Disabling reopening slightly reduces Average/Last Forgetting from $5.51/7.87\%$ to $5.18/7.76\%$, but also decreases accuracy to $83.76/76.12\%$, reflecting the stability–plasticity trade-off introduced by stronger structural preservation. Finally, $k = 8, 16, 32$ yield nearly identical Average accuracy $(84.19, 84.17, 84.20\%)$ and similar Last accuracy, showing that CFR is not sensitive to the exact core dimension.

6 Conclusion

We study fixed-capacity CLIP continual learning from the perspective of functional rather than structural continuity. Longitudinal interventions reveal Historical Necessity Drift: historical functional dependence can weaken, strengthen, or migrate, while compact functionally selected subspaces can outperform the full adapter, exposing stale preservation and functional interference. Motivated by these findings, we introduce CFR, which summarizes accumulated behavior with a fixed-size functional state, reorganizes it through core–complement refactoring, and fully reopens the adapter for subsequent learning. Experiments across multiple datasets and task horizons consistently validate CFR, with its advantages becoming most pronounced over long sequences. Overall, our results suggest that continual functionality is better sustained through adaptive functional reorganization than permanent structural ownership.

AI USE STATEMENT

In this work, we used generative AI tools to assist with literature retrieval and discovery, including identifying potentially relevant prior work, and to polish the writing of the manuscript for clarity and readability. All AI-assisted literature suggestions were independently checked against the original sources, and all AI-assisted revisions were reviewed and edited by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies continual adaptation of pretrained vision-language models using standard image-classification benchmarks, including CIFAR-100 and TinyImageNet. It does not involve human-subject studies, collection of personal or sensitive information, or deployment in high-stakes settings. Our experiments use existing benchmark datasets and pretrained models solely for methodological evaluation. As our approach builds on pretrained vision-language representations, it may inherit biases or limitations already present in the underlying pretrained model and benchmark data. We do not identify additional ethical risks specific to the proposed continual functional refactoring procedure beyond those associated with the underlying models and datasets.

REPRODUCIBILITY STATEMENT

We provide detailed experimental and implementation information to facilitate reproduction of our results. The continual-learning protocol, model architecture, optimization settings, diagnostic procedure, and CFR implementation are described in Appendix A, with the complete procedure summarized in Algorithm 1 and the default configuration listed in Table S1. The main paper specifies the functional-subspace interventions and CFR formulation in Sections 3 and 4, while additional diagnostic results, task-boundary analyses, ablations, and robustness studies are reported in Appendices B and C. All reported results use the same three random seeds within each experimental setting and are averaged across the three runs. We plan to release the implementation code upon acceptance.

REFERENCES

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in neural information processing systems*, 35:16664–16678, 2022.

Quan Cheng, Yuanyu Wan, Lingyu Wu, Chenping Hou, and Lijun Zhang. Continuous subspace optimization for continual learning. *Advances in Neural Information Processing Systems*, 38: 15376–15398, 2026.

Yuxuan Ding, Lingqiao Liu, Chunna Tian, Jingyuan Yang, and Haoxuan Ding. Don't stop learning: Towards continual learning for the clip model. *arXiv preprint arXiv:2207.09248*, 2022.

Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with clip: Adaptive representation adjustment and parameter fusion. In *European Conference on Computer Vision*, pp. 214–231. Springer, 2024.

Linlan Huang, Xusheng Cao, Haori Lu, Yifan Meng, Fei Yang, and Xialei Liu. Mind the gap: Preserving and compensating for the modality gap in clip-based continual learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3777–3786. IEEE, 2025.

Yujin Jo and Taesup Kim. Memory-free continual learning with null space adaptation for zero-shot vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 106089–106106, 2026.

Borui Kang, Lei Wang, Zhiping Wu, Tao Feng, Yawen Li, Yang Gao, and Wenbin Li. Dynamic multi-layer null space projection for vision-language continual learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2077–2086. IEEE, 2025.

Prakhar Kaushik, Ankit Vaidya, Shravan Chaudhari, Rama Chellappa, and Alan Yuille. Shared lora subspaces for almost strict continual learning. In *European Conference on Computer Vision*, pp. 480–497. Springer, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

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

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

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

Zekai Lin, Chao Xue, Di Liang, Xingsheng Han, Peiyang Liu, Xianjie Wu, Lei Jiang, Yu Lu, Bob Simons, Shuang Liang, et al. Parameter importance is not static: Evolving parameter isolation for supervised fine-tuning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32846–32864, 2026.

Mao-Lin Luo, Zi-Hao Zhou, Yi-Lin Zhang, Yuanyu Wan, Min-Ling Zhang, and Tong Wei. Keeplora: Continual learning with residual gradient adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 136017–136040, 2026.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pp. 379–395. Springer, 2024.

Fuli Qiao and Mehrdad Mahdavi. Merge before forget: A single lora continual learning via continual merging. In *International Conference on Learning Representations*, volume 2026, pp. 8133–8159, 2026.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, Weili Guan, Min Zhang, and Liqiang Nie. Splitlora: Balancing stability and plasticity in continual learning through gradient space splitting. In *International Conference on Learning Representations*, volume 2026, pp. 48458–48480, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

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

Huan Zhang, Shuyu Dong, Yujin Zheng, Dingwen Wang, Shenghua Fan, and Fan Lyu. Subspace alignment for clip-based continual learning via canonical correlation analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32257–32266, 2026.

Zhixin Zhang, Zeming Wei, and Meng Sun. Dynamic orthogonal continual fine-tuning for mitigating catastrophic forgettings. *arXiv preprint arXiv:2509.23893*, 2025.

Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4489–4504, 2025.

A Experimental Details and Reproducibility

A.1 Continual-Learning Protocol

We evaluate all methods under the class-incremental learning setting on CIFAR-100 and TinyImageNet. CIFAR-100 is divided into sequences of 10, 20, and 50 tasks, while TinyImageNet is divided into sequences of 5, 10, and 20 tasks. These settings span increasingly long task horizons while preserving the same fixed adaptation architecture throughout the sequence.

At task t, optimization uses only samples from the current task $\mathcal{D}_t$. Samples from previously observed tasks are not retained or replayed for subsequent training. After each task, the model is evaluated over all classes observed up to the current checkpoint without access to task identity at inference time.

Let $a_{t,j}$ denote the accuracy on task j after completing task t, where $j \le t$. The seen-task accuracy at checkpoint t is

$$A_t = \frac{1}{t}\sum_{j=1}^{t} a_{t,j}.$$

We define Average and Last Accuracy as

$$A_{\text{avg}} = \frac{1}{T}\sum_{t=1}^{T} A_t, \qquad A_{\text{last}} = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}.$$

For forgetting, the forgetting of an earlier task j at checkpoint t is measured relative to its best accuracy before task t:

$$f_{t,j} = \max_{k=j,\dots,t-1} a_{k,j} - a_{t,j}, \qquad j < t.$$

The mean forgetting at checkpoint t is therefore

$$F_t = \frac{1}{t-1}\sum_{j=1}^{t-1} f_{t,j}.$$

We report Average and Last Forgetting as

$$F_{\text{avg}} = \frac{1}{T-1}\sum_{t=2}^{T} F_t, \qquad F_{\text{last}} = \frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{k=j,\dots,T-1} a_{k,j} - a_{T,j}\right).$$

Thus, Average Accuracy and Average Forgetting summarize performance over the entire continual-learning trajectory, whereas Last Accuracy and Last Forgetting characterize the model after completing the full task sequence. Higher accuracy and lower forgetting are better. For all methods and experimental settings, we use the same three random seeds and report the mean over the three runs. The same task construction and training protocol are maintained across compared methods within each setting.

A.2 Architecture and Optimization

All experiments use CLIP ViT-B/16 as the pretrained vision-language backbone. The pretrained backbone parameters remain frozen throughout continual learning. Bottleneck adapters are inserted into all 12 Transformer layers of both the visual and text encoders, and only the inserted adapters are optimized during sequential training. We use a bottleneck dimension of $r = 64$.

The visual and text adapters are optimized jointly on each incoming task, but CFR is applied exclusively to the visual branch. Specifically, the visual adapters participate in functional-state construction, core extraction, core–complement refactoring, and subsequent reopening. The text-side adapters are not refactored and are updated throughout the sequence using ordinary gradient-based optimization. This asymmetric design allows CFR to reorganize the visual adaptation space while retaining the standard continual optimization procedure on the text side.

Algorithm 1 Continual Functional Refactoring (CFR)

Require: Frozen pretrained CLIP backbone; task sequence $\{\mathcal{D}_t\}_{t=1}^{T}$; visual and text adapters; core rank k

Initialize the visual and text adapters
Initialize the functional state $\mathcal{H}_0 = \{(G_{0,\ell}, B_{0,\ell}, e_{0,\ell})\}_{\ell=1}^{L}$ with zeros
for $t = 1, \ldots, T$ **do**
  Store the visual-adapter state θ_{t-1}^{+}
  Train all adapter parameters on $\mathcal{D}_t$ using the current-task objective
  Obtain the pre-refactoring state θ_t^{-}
  if $t > 1$ **then**
    for each visual Transformer block ℓ **do**
      Estimate alignment $R_{t,\ell}$ from paired activations of θ_{t-1}^{+} and θ_t^{-}
      Form the augmented alignment $\bar{R}_{t,\ell} = \mathrm{diag}(R_{t,\ell}, 1)$
      Transport the accumulated functional state

$$G_{t-1,\ell} \leftarrow \bar{R}_{t,\ell}^{\top} G_{t-1,\ell} \bar{R}_{t,\ell},$$
$$B_{t-1,\ell} \leftarrow B_{t-1,\ell} \bar{R}_{t,\ell},$$

    end for
  end if
  for each visual Transformer block ℓ **do**
    Collect bottleneck activations h_ℓ and responses $z_\ell = W_\ell^{-} h_\ell + b_\ell^{-}$ on $\mathcal{S}_t^{\mathrm{sum}}$
    Form the augmented activation $\bar{h}_\ell = [h_\ell; 1]$
    Compute the task-level statistics

$$\widetilde{G}_{t,\ell} = \frac{1}{N_t} \sum \bar{h}_\ell \bar{h}_\ell^{\top},$$
$$\widetilde{B}_{t,\ell} = \frac{1}{N_t} \sum z_\ell \bar{h}_\ell^{\top},$$
$$\widetilde{e}_{t,\ell} = \frac{1}{N_t} \sum \|z_\ell\|_2^2.$$

    Normalize with $\gamma_{t,\ell} = \|\widetilde{G}_{t,\ell}^{h}\|_F$, where $\widetilde{G}_{t,\ell}^{h}$ is the bottleneck block of $\widetilde{G}_{t,\ell}$

$$G_{t,\ell}^{\mathrm{task}} = \widetilde{G}_{t,\ell} / \gamma_{t,\ell},$$
$$B_{t,\ell}^{\mathrm{task}} = \widetilde{B}_{t,\ell} / \gamma_{t,\ell},$$
$$e_{t,\ell}^{\mathrm{task}} = \widetilde{e}_{t,\ell} / \gamma_{t,\ell}.$$

    Accumulate the functional state

$$G_{t,\ell} = G_{t-1,\ell} + G_{t,\ell}^{\mathrm{task}},$$
$$B_{t,\ell} = B_{t-1,\ell} + B_{t,\ell}^{\mathrm{task}},$$
$$e_{t,\ell} = e_{t-1,\ell} + e_{t,\ell}^{\mathrm{task}}.$$

    Extract the top-k eigenspace of $G_{t,\ell}^{h}$ and form projector $P_{t,\ell}$
    Recover the accumulated affine mapping

$$[W_\ell^{*}, b_\ell^{*}] = B_{t,\ell} (G_{t,\ell} + \Lambda_{t,\ell})^{-1}$$

    Refactor the visual-adapter up-projection

$$W_\ell^{+} = W_\ell^{*} P_{t,\ell} + W_\ell^{-} (I - P_{t,\ell}),$$
$$b_\ell^{+} = b_\ell^{*}.$$

  end for
  Keep visual down-projections and text adapters unchanged
  Obtain the refactored model θ_t^{+}
end for
  return θ_T^{+}

Each task is trained for 4 epochs with a batch size of 64. We use AdamW with a learning rate of 1×10^{-3} and weight decay of 0. Unless otherwise specified, these optimization settings are shared across all tasks and experimental configurations. Experiments are conducted on a single NVIDIA GeForce RTX 5090 D V2 GPU.

For CFR, the visual bottleneck dimension is $r = 64$, and the default core dimension is $k = 16$. Refactoring is performed once after completing the optimization of each task. After refactoring, no visual adapter direction remains frozen: all 64 bottleneck dimensions are reopened and are jointly trainable when learning the next task. The text adapters continue directly from their current parameters without any task-boundary refactoring.

A.3 Diagnostic and Training Protocols

The Historical Necessity Drift analysis in Section 3 and the CFR procedure in Section 4 use distinct information flows.

For diagnosis, we construct a fixed held-out diagnostic set for each previously observed source task. These samples are used to collect visual-adapter bottleneck activations, identify candidate functional directions, measure their behavioral importance through remove/keep interventions, and track historical and current functional subspaces across later checkpoints.

For a given source task, the same diagnostic samples are used across checkpoints so that temporal differences reflect changes in the model rather than changes in the evaluated samples.

The historical diagnostic samples are used only for the analyses of functional dependence and are never included in continual optimization. They are not replayed when learning subsequent tasks and are not used to update the CFR functional state.

CFR instead operates from current-task visual-adapter behavior and a fixed-size accumulated functional state. After learning task t, the visual branch is summarized by

$$\mathcal{H}_t = \{G_\ell, B_\ell, e_\ell\}_{\ell=1}^{L},$$

where G_ℓ, B_ℓ, and e_ℓ are the second-order statistics defined in Section 4.1. Each task-level summary is normalized before aggregation, and the resulting state has a size determined only by the adapter architecture rather than the number of previously observed tasks. This state is then used to construct the visual functional core and recover its accumulated input–output mapping without accessing historical samples.

The text-side adapters do not maintain an analogous functional state and are not involved in the refactoring operation. They are updated only through the standard task-training objective. Consequently, the diagnostic samples, the CFR functional state, and the normal optimization of the text adapters remain separate throughout the continual-learning sequence.

A.4 Continual Functional Refactoring Procedure

Algorithm 1 summarizes the complete training procedure. Starting from the previously refactored model, both visual and text adapters are first optimized on the current task. The resulting visual-adapter behavior is then incorporated into the accumulated functional state. For each visual Transformer block, CFR extracts a compact functional core from the accumulated bottleneck geometry and reconstructs its behavior from the input–output statistics. The complementary component of the newly trained visual adapter is retained to preserve recent functionality that may not yet have been consolidated into the core.

The refactored visual adapter therefore combines accumulated functionality in the core with the most recent residual in its complement. This organization is temporary: after refactoring, all visual-adapter dimensions are reopened before the next task. In contrast, the text adapters undergo no refactoring and proceed directly to subsequent optimization. Historical training samples are discarded and are not required by later CFR operations.

A.5 Implementation Summary

Table S1 summarizes the default configuration used in the main experiments.

B Extended Historical Necessity Drift Analysis

B.1 Full Historical Necessity Trajectories

Table S2 reports the complete checkpoint-wise necessity trajectories for the three source tasks visualized in the main paper. Recall that $N = A_{\text{full}} - A_{\text{remove}}$, such that $N > 0$ indicates that removing the historical functional subspace degrades performance, whereas $N < 0$ indicates that the same subspace is currently interfering with the task.

Table S1: Default implementation and training configuration.

Configuration	Setting
Backbone	CLIP ViT-B/16
Backbone parameters	Frozen
Visual Transformer layers	12
Text Transformer layers	12
Adapter placement	Visual + Text
Adapter bottleneck dimension r	64
CFR branch	Visual only
CFR core dimension k	16
Epochs per task	4
Batch size	64
Optimizer	AdamW
Learning rate	1×10^{-3}
Weight decay	0
Refactoring frequency	Every task boundary
Historical replay	None
Task-specific parameter growth	None
Number of runs	3 seeds
Hardware	NVIDIA GeForce RTX 5090 D V2

The three source tasks exhibit markedly different temporal behaviors. For Source Task 1, the historical Top-10% subspace changes from beneficial at its source checkpoint ($N = 1.2$) to interfering at intermediate checkpoints, reaching $N = -3.0$ at Task 4, before becoming beneficial again at Task 9 ($N = 1.5$). In contrast, Source Task 2 remains strongly necessary throughout the sequence: its Top-10% necessity stays positive at every checkpoint, from 6.6 at Task 2 to 6.1 at Task 9. Source Task 3 follows a third pattern, with its historical necessity progressively weakening from 6.3 at Task 3 to only 0.7 at Task 9. The same qualitative behaviors are consistently observed for the Top-5%, Top-10%, and Top-20% selections.

These accuracy-based trajectories are corroborated by the corresponding loss changes in table S3. In particular, removing the historical Top-10% subspace of Source Task 1 at Task 4 not only improves accuracy by 3.0 percentage points, but also decreases the loss by 0.1002, confirming that the sign reversal is not an artifact of accuracy fluctuations. Source Task 2, by comparison, maintains positive loss increases after removal throughout the sequence, while Source Task 3 shows a gradual reduction in the loss penalty toward later checkpoints. Together, these results show that historical functional necessity is non-stationary and task dependent: a previously important realization may remain indispensable, become temporarily interfering, or gradually become replaceable as continual learning proceeds.

B.2 Representation Reorganization and Spectrum Broadening

We next examine how the representation geometry associated with each source task evolves as continual learning proceeds. Table S4 reports the overlap between the representation subspace identified at the source checkpoint and the corresponding subspace recomputed at later checkpoints, while keeping the source-defined rank fixed. For all three source tasks, the overlap decreases substantially over time. By Task 9, the overlap falls to 0.461, 0.510, and 0.520 for Source Tasks 1, 2, and 3, respectively. These values show that the dominant representation span of an old task is not preserved as a fixed geometric object, but undergoes substantial reorganization during subsequent learning.

Importantly, this reorganization is not accompanied by a progressive concentration of the old-task representation into fewer dominant directions. Table S5 and table S6 show the opposite trend. For Source Task 1, the mean r_{90} increases from 9.17 at its source checkpoint to 14.83 at Task 9, while the mean effective rank increases from 4.77 to 8.92. Source Task 2 exhibits a similar evolution, with mean r_{90} increasing from 11.58 to 15.50 and mean effective rank from 6.41 to 9.22. Source Task 3 follows the same endpoint trend, increasing from 12.33 to 15.83 in mean r_{90} and from 7.26 to 9.74 in effective rank. Thus, across all three source tasks, later continual adaptation broadens rather than compresses the activation spectrum.

Table S2: Checkpoint-wise accuracy and necessity-based Top-$K\%$ results for three source tasks.

Checkpoint	Full-adapter Acc. (%)	N, Top-5%	N, Top-10%	N, Top-20%
Source Task 1				
Task 1	94.3	1.2	1.2	1.3
Task 2	91.9	0.1	0.1	0.1
Task 3	89.9	0.4	0.3	0.4
Task 4	85.5	**-2.8**	**-3.0**	**-2.9**
Task 5	83.3	**-1.4**	**-1.8**	**-1.8**
Task 6	80.1	-0.2	**-1.2**	**-1.3**
Task 7	82.1	-0.4	-0.1	0.1
Task 8	79.2	-0.3	0.1	-0.4
Task 9	78.4	**1.7**	**1.5**	**1.5**
Source Task 2				
Task 2	89.2	5.6	6.6	7.1
Task 3	82.4	6.3	7.0	7.4
Task 4	78.0	3.8	4.6	4.8
Task 5	75.0	5.9	6.5	7.0
Task 6	67.9	6.4	6.4	7.5
Task 7	72.3	5.2	5.8	6.7
Task 8	65.0	3.8	3.8	4.6
Task 9	66.5	**5.5**	**6.1**	**7.2**
Source Task 3				
Task 3	88.6	6.1	6.3	7.1
Task 4	87.0	4.9	5.4	5.7
Task 5	84.4	4.3	4.9	5.0
Task 6	82.5	4.6	4.5	5.0
Task 7	83.4	4.2	4.3	4.9
Task 8	76.4	2.3	2.1	1.8
Task 9	75.8	**0.7**	**0.7**	**-0.3**

Table S3: Checkpoint-wise changes in necessity for different source tasks.

Source	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
Task 1	+0.0113	+0.0010	+0.0063	**-0.1002**	-0.0374	-0.0567	-0.0254	+0.0185	**+0.0125**
Task 2	—	+0.2520	+0.2690	+0.1920	+0.3030	+0.1860	+0.2390	+0.2010	**+0.2550**
Task 3	—	—	+0.1850	**+0.1810**	+0.1260	+0.1540	+0.1460	+0.0500	**+0.0720**

The checkpoint-wise results in table S5 further show that this broadening is not strictly monotonic at every intermediate checkpoint. For example, the mean r_{90} of Source Task 1 temporarily decreases from 12.08 at Task 3 to 10.42 at Task 4 before subsequently increasing to 14.83. Source Task 2 exhibits a similar temporary contraction. Nevertheless, the long-term change from the source checkpoint to Task 9 is consistent across all three tasks: both r_{90} and effective rank increase substantially. The relevant phenomenon is therefore a progressive redistribution of representation energy over longer horizons, rather than a monotonic checkpoint-by-checkpoint increase.

The reorganization is also layer dependent. Table S7 reports the layer-wise r_{90} values for Source Task 3 at its source checkpoint and at Task 9. Several middle-to-late visual blocks exhibit particularly large increases. For example, the r_{90} of Layer 7 increases from 2 to 14, and that of Layer 8 from 1 to 17, whereas several other layers change much less. This indicates that continual adaptation does not reorganize the visual adapter uniformly across depth; instead, different layers contribute differently to the expansion and redistribution of the old-task representation.

Taken together, these results establish two geometric properties of the evolving adapter representation. First, historical representation subspaces are only partially preserved and undergo substantial

Table S4: Checkpoint-wise trajectories for three source tasks. Each row starts from its source checkpoint and tracks the quantity across subsequent checkpoints.

Source Task	Source	+1	+2	+3	+4	+5	+6	+7	+8
Task 1	1.000	0.741	0.609	0.572	0.513	0.474	0.485	0.464	**0.461**
Task 2	1.000	0.715	0.647	0.597	0.558	0.546	0.496	**0.510**	—
Task 3	1.000	0.759	0.681	0.621	0.604	0.549	**0.520**	—	—

Table S5: Evolution of r_{90} and effective rank across checkpoints for Source Tasks 1 and 2.

Checkpoint	Mean r_{90}	Total r_{90}	Mean Effective Rank
Source Task 1			
Task 1	9.17	110	4.77
Task 2	11.25	135	6.32
Task 3	12.08	145	7.05
Task 4	10.42	125	6.24
Task 5	11.67	140	6.97
Task 6	12.83	154	7.45
Task 7	12.92	155	7.43
Task 8	14.25	171	8.29
Task 9	**14.83**	**178**	**8.92**
Source Task 2			
Task 2	11.58	139	6.41
Task 3	12.08	145	6.97
Task 4	10.42	125	6.25
Task 5	11.75	141	7.07
Task 6	12.58	151	7.64
Task 7	12.83	154	7.50
Task 8	13.92	167	8.48
Task 9	**15.50**	**186**	**9.22**

restructuring over time. Second, the corresponding activation spectra generally become broader rather than lower-rank. Therefore, the functional changes observed in Section B.1 cannot be explained by a simple picture in which continual learning progressively compresses old-task representations into a smaller and more stable subspace. In Section B.3, we further show that even highly similar geometric reorganization can correspond to markedly different functional necessity trajectories.

Table S6: Changes in r_{90} and effective rank from the source checkpoint to Task 9 for three source tasks.

Source Task	Source Mean r_{90}	Task-9 Mean r_{90}	Source Eff. Rank	Task-9 Eff. Rank
Task 1	9.17	14.83	4.77	8.92
Task 2	11.58	15.50	6.41	9.22
Task 3	12.33	**15.83**	7.26	**9.74**

B.3 Similar Geometry, Different Functional Fate

Table S8 compares the historical and current Top-10% functional subspaces at Task 9. Although the three source tasks exhibit similar geometric evolution, their functional trajectories differ substantially. Their representation overlaps at Task 9 are 0.461, 0.510, and 0.520, while the historical–current functional overlaps are similarly close at 0.265, 0.267, and 0.242. Nevertheless, Source Task 1 undergoes role reversal, Source Task 2 remains persistently necessary, and Source Task 3 shows late weakening.

Table S7: Layer-wise r_{90} for Source Task 3 at its source checkpoint and after learning through Task 9.

Checkpoint	0	1	2	3	4	5	6	7	8	9	10	11
Source	7	10	14	16	14	22	4	2	1	20	23	15
Task 9	10	13	13	19	17	18	7	**14**	**17**	24	23	15

This contrast is also reflected in necessity. At Task 9, the historical and current Top-10% subspaces yield $(N_H, N_C) = (1.5, 3.6)$, $(6.1, 5.0)$, and $(0.7, 2.4)$ for Source Tasks 1–3, respectively. Most notably, Source Task 3 retains a strong historical keep-only gain ($G_H = +2.8$) despite weak historical necessity ($N_H = 0.7$), indicating that a functional realization can remain effective while becoming replaceable. These results show that geometric similarity alone does not determine functional necessity.

Table S8: Historical–current comparison of Top-10% functional directions for three representative source tasks.

Metric	Task 1	Task 2	Task 3
Historical N at Source	1.2	**6.6**	**6.3**
Historical N at Task 9	1.5	**6.1**	**0.7**
Current N at Task 9	**3.6**	5.0	**2.4**
Historical–Current Functional Overlap	0.265	0.267	0.242
Representation Overlap at Task 9	0.461	0.510	0.520
Trajectory Pattern	Role reversal	Persistent necessity	Late weakening
Historical G at Task 9	+1.6	+0.9	**+2.8**
Current G at Task 9	**+2.9**	+1.3	**+4.0**

B.4 Extended Remove/Keep-only Interventions

Table S9 reports the complete Task 9 removal and keep-only results across selection ratios of 5%, 10%, and 20%. Across all source tasks and subspace sizes, the keep-only accuracy exceeds that of the full adapter, showing that a compact functionally selected subset can realize the task more effectively than the complete mixed representation.

At the same time, necessity and standalone effectiveness remain distinct. Source Task 2 retains high historical necessity ($N = 5.5$–7.2), whereas Source Task 3 shows weak historical necessity ($N = -0.3$–0.7) but strong keep-only gains ($G = +2.7$–$+3.2$). Thus, a historical realization can remain highly effective while no longer being indispensable to the current model.

Table S10 further shows that historical and current functional subspaces retain non-random overlap, but are far from identical. Together, these results support a picture of partial functional continuity with substantial reorganization, and motivate CFR to consolidate useful functionality rather than permanently preserve stale historical structure.

C Understanding Continual Functional Refactoring

C.1 Functional Redistribution at Refactoring Boundaries

Table S11 examines the immediate effect of CFR at task boundaries on the CIFAR-100 50-task sequence. Refactoring consistently improves historical retention: old-task accuracy increases by 2.92 points on average and improves at 48 of the 49 effective boundaries. Consequently, the average accuracy over all seen tasks also increases by 2.55 points.

This gain is accompanied by reduced current-task accuracy (-4.45 points on average), indicating that refactoring redistributes capacity away from task-specific specialization toward accumulated

Table S9: Task-9 removal and keep-only results for historical and current functional subspaces. $N = \text{Full} - \text{Remove}$ and $G = \text{Keep-only} - \text{Full}$.

Subspace	Ratio	Full	Remove	N	Keep-only	G
Source Task 1						
Historical	5%	78.4	76.7	1.7	80.1	+1.7
Historical	10%	78.4	76.9	1.5	80.0	+1.6
Historical	20%	78.4	76.9	1.5	80.1	+1.7
Current	5%	78.4	74.9	3.5	80.9	+2.5
Current	10%	78.4	74.8	3.6	81.3	+2.9
Current	20%	78.4	74.8	3.6	81.2	+2.8
Source Task 2						
Historical	5%	66.5	61.0	5.5	67.0	+0.5
Historical	10%	66.5	60.4	6.1	67.4	+0.9
Historical	20%	66.5	59.3	7.2	68.0	+1.5
Current	5%	66.5	62.0	4.5	66.6	+0.1
Current	10%	66.5	61.5	5.0	67.8	+1.3
Current	20%	66.5	61.0	5.5	68.5	+2.0
Source Task 3						
Historical	5%	75.8	75.1	0.7	79.0	+3.2
Historical	10%	75.8	75.1	0.7	78.6	+2.8
Historical	20%	75.8	76.1	-0.3	78.5	+2.7
Current	5%	75.8	73.6	2.2	79.7	+3.9
Current	10%	75.8	73.4	2.4	79.8	+4.0
Current	20%	75.8	73.0	2.8	79.7	+3.9

Table S10: Overlap between historical and current functional subspaces at different selection ratios, compared with the corresponding random expectation.

Source Task	Ratio	Hist.–Current Overlap	Random Expectation
Task 1	5%	0.252	0.056
Task 1	10%	**0.265**	0.109
Task 1	20%	0.276	0.207
Task 2	5%	0.288	0.053
Task 2	10%	**0.267**	0.098
Task 2	20%	0.306	0.205
Task 3	5%	0.221	0.055
Task 3	10%	**0.242**	0.108
Task 3	20%	0.293	0.201

functionality. Thus, CFR acts as an explicit functional consolidation step rather than a behavior-preserving reparameterization.

C.2 Why Accumulated Functional State Matters

Table S12 compares the full accumulated functional state with a variant that retains only the current-task statistics. Accumulation improves Average Accuracy from 84.24 to 84.99 and reduces Average Forgetting from 8.37 to 6.19. It also increases the mean old-task accuracy by 0.88 points.

The current-task-only variant retains slightly stronger short-term plasticity, yielding higher current-task and Last accuracy. This trade-off shows that accumulated functional statistics primarily improve long-horizon stability rather than immediate task specialization, supporting their role in consolidating functionality across the continual sequence.

Table S11: Task-boundary effect of CFR. We report the average accuracy immediately before and after refactoring, together with the mean change and the number of task boundaries exhibiting positive, zero, or negative changes.

Accuracy Metric	Before	After	Mean Δ	$\Delta > 0$	$\Delta = 0$	$\Delta < 0$
Seen-task Avg.	82.13	**84.69**	**+2.55**	46	2	1
Old-task Avg.	81.87	**84.79**	**+2.92**	48	1	0
Current-task	**89.11**	84.66	**-4.45**	3	3	43

C.3 Robustness to Adapter Capacity

To examine whether CFR depends on a particular adapter size, we vary the bottleneck width as $r \in \{32, 64, 128\}$ while keeping the core ratio fixed at $k/r = 1/4$. For each capacity, CFR is compared with a matched no-refactoring baseline under the same TinyImageNet-20 setting.

As shown in table S13, CFR consistently improves continual-learning performance across all three capacities. Average Accuracy increases by 0.46, 1.06, and 1.04 points for $r = 32$, 64, and 128, respectively, while Last Accuracy improves by 1.24–2.28 points. More notably, the reduction in forgetting is stable across capacity scales: Average Forgetting decreases by 3.68, 3.77, and 4.87 points, and Last Forgetting decreases by 4.79, 5.36, and 6.12 points.

The absolute performance of CFR changes only moderately as the adapter width increases, whereas its advantage over the matched baseline remains consistent. This indicates that the benefit of functional refactoring is not tied to the default $r = 64$ configuration, but persists under both tighter and larger fixed-capacity budgets.

C.4 Behavioral Roles of the Refactored Components

We further examine how the two components of the refactored visual adapter contribute to model behavior. Recall that CFR constructs

$$W_\ell^+ = W_\ell^* P_\ell + W_\ell^-(I - P_\ell),$$

where $W_\ell^* P_\ell$ is the reconstructed functional core and $W_\ell^-(I - P_\ell)$ is the recent residual retained in the complementary space. We intervene on these two terms after refactoring while leaving the remaining model unchanged. The intervention is applied to the visual up-projection weights; the shared reconstructed bias is retained in all variants so that the comparison isolates the contribution of the two weight components. Table S14 reports results averaged across the CIFAR-100 50-task

Table S12: Effect of accumulating functional state across tasks. Current-task retains only the current-task functional state, whereas Accumulated CFR preserves information across tasks.

State	Avg. Acc. ↑	Last Acc. ↑	Avg. Forget. ↓	Last Forget. ↓	Old-task Acc. ↑	Current-task Acc. ↑
Current-task	84.24	**76.73**	8.37	11.65	83.91	**86.38**
Accumulated CFR	**84.99**	75.86	**6.19**	**10.39**	**84.79**	84.66
Accum. − Current	**+0.75**	-0.87	**-2.19**	**-1.27**	**+0.88**	-1.71

sequence. *Seen-task Accuracy* denotes the average accuracy over all tasks observed at a checkpoint, *Old-task Accuracy* averages over tasks preceding the current one, and *Current-task Accuracy* evaluates the newly learned task. For each intervention, the corresponding drop is defined relative to the full CFR model,

$$\Delta A = A_{\text{Full CFR}} - A_{\text{Intervention}},$$

so a larger value indicates a larger functional contribution of the removed component.

Removing the functional core causes a substantial degradation across all three evaluations: Seen-task, Old-task, and Current-task accuracy decrease by 22.19, 22.55, and 21.18 percentage points, respectively. In contrast, removing the recent residual changes them by only 0.09, 0.08, and 0.46

points. The reconstructed core therefore becomes the dominant carrier of accumulated functionality after refactoring, including most of the newly acquired behavior. The recent residual has a much smaller immediate contribution, but its effect is relatively more concentrated on the current task (0.46 versus 0.08 points on old tasks), consistent with its role as a supplementary channel for recent functionality not fully captured by the consolidated core.

These results refine, rather than impose, a strict historical–recent partition: CFR does not assign old knowledge exclusively to the core and new knowledge exclusively to the complement. Instead, refactoring rapidly integrates most behavior into a shared functional core, while retaining a small current-biased residual before the full adapter is reopened for subsequent learning.

Table S13: Performance of CFR across different adapter capacities on TinyImageNet-20.

Adapter Width r	Core Rank k	Variant	Avg. Acc. ↑	Last Acc. ↑	Avg. Forget. ↓	Last Forget. ↓
32	8	No Refactor	82.90	74.53	8.80	12.58
32	8	**CFR**	**83.36**	**75.77**	**5.12**	**7.79**
		CFR − No Refactor	**+0.46**	**+1.24**	**-3.68**	**-4.79**
64	16	No Refactor	83.01	74.05	9.35	13.68
64	16	**CFR**	**84.07**	**76.33**	**5.58**	**8.33**
		CFR − No Refactor	**+1.06**	**+2.28**	**-3.77**	**-5.36**
128	32	No Refactor	83.11	74.73	9.85	13.94
128	32	**CFR**	**84.15**	**76.56**	**4.98**	**7.82**
		CFR − No Refactor	**+1.04**	**+1.83**	**-4.87**	**-6.12**

Table S14: Ablation of the core and recent-residual components in CFR. Performance drops are measured relative to the full CFR model.

Intervention	Seen-task Acc. ↑	Old-task Acc. ↑	Current-task Acc. ↑	Seen-task Drop ↓	Old-task Drop ↓	Current-task Drop ↓
Full CFR	**84.91**	**84.71**	**84.79**	—	—	—
w/o Core	62.72	62.15	63.61	22.19	22.55	21.18
w/o Recent Residual	84.81	84.63	84.33	**0.09**	**0.08**	**0.46**

Table S15: Comparison of forgetting on TinyImageNet under class-incremental learning. Average/Last forgetting (%) is averaged over three seeds; lower is better, and best/second-best results are shown in **bold**/underline.

Method	Venue	5 Tasks		10 Tasks		20 Tasks	
		Average ↓	Last ↓	Average ↓	Last ↓	Average ↓	Last ↓
CLIP Zero-shot	ICML'21	8.82	9.24	8.29	9.38	7.77	9.65
Continual-CLIP	arXiv'22	7.72	8.03	7.50	<u>8.29</u>	7.21	<u>8.61</u>
MoE4Adapters	CVPR'24	7.37	9.08	7.02	9.14	6.44	9.11
MG-CLIP	ICCV'25	7.55	10.00	7.39	9.50	7.08	9.32
DMNSP	ICCV'25	8.19	8.91	6.90	8.96	<u>6.21</u>	8.95
CCA-CL w/ FTT	CVPR'26	<u>6.52</u>	<u>7.99</u>	<u>6.32</u>	8.31	6.51	9.21
CCA-CL w/o FTT	CVPR'26	9.14	10.63	8.70	10.91	8.33	11.05
CFR (Ours)	–	**5.98**	**7.44**	**5.74**	**7.71**	**5.51**	**7.87**

C.5 Extended Forgetting Results

Table S15 and Table S16 report Average and Last Forgetting across all task horizons. CFR achieves the lowest forgetting in 10 of the 12 reported metrics and the second-lowest in the remaining two. On TinyImageNet, CFR obtains the best Average and Last Forgetting under all 5-, 10-, and 20-task

Table S16: Comparison of forgetting on CIFAR-100 under class-incremental learning. Average/Last forgetting (%) is averaged over three seeds; lower is better, and best/second-best results are shown in **bold**/underline.

Method	Venue	10 Tasks		20 Tasks		50 Tasks	
		Average ↓	Last ↓	Average ↓	Last ↓	Average ↓	Last ↓
CLIP Zero-shot	ICML'21	8.77	10.66	8.44	10.87	7.91	10.97
Continual-CLIP	arXiv'22	8.70	10.57	8.40	10.80	7.95	10.99
MoE4Adapters	CVPR'24	7.89	11.99	7.35	10.37	7.45	12.24
MG-CLIP	ICCV'25	7.46	10.48	7.14	11.46	8.77	14.18
DMNSP	ICCV'25	5.48	8.20	6.18	9.93	<u>6.19</u>	11.53
CCA-CL w/ FTT	CVPR'26	**4.61**	**6.61**	<u>5.35</u>	<u>8.12</u>	7.84	<u>10.47</u>
CCA-CL w/o FTT	CVPR'26	6.54	9.04	6.48	9.22	8.17	10.55
CFR (Ours)	–	<u>4.78</u>	<u>7.12</u>	**5.26**	**7.80**	**5.90**	**8.85**

settings. On CIFAR-100, it remains competitive at 10 tasks and becomes strongest as the sequence length increases, reaching $5.26/7.80$ at 20 tasks and $5.90/8.85$ at 50 tasks.

The improvement is particularly clear in the long-horizon regime, consistent with the main-paper results: repeated functional refactoring reduces accumulated forgetting while maintaining strong continual-learning accuracy. These results further support CFR's advantage under sustained fixed-capacity adaptation.