

参赛学生姓名： JIAJUN LI

中学： 北京市第一〇一中学

省份： 北京市

国家/地区： 中国

指导老师姓名： 钟毅

指导老师单位： 清华大学

论文题目： LOOM: Weaving Multilayer
Representations through Prototypes for General
Continual Learning

LOOM: Weaving Multilayer Representations through Prototypes for General Continual Learning

Jiajun Li

Yi Zhong

Abstract

General continual learning (GCL) assumes dynamic, single-pass data streams without clear task boundaries. Pretrained models provide representations at multiple depths, yet analytic continual learning typically uses only the final representation or forms a jointly concatenated classifier whose second-order statistics grow quadratically with the number of selected layers. We introduce **LOOM**, a framework that combines sequential parameter-efficient tuning with prototype-based multilayer analytic classification. LOOM retains exact second-order statistics within each layer and reconstructs cross-layer Gram blocks from class prototypes, yielding a structured approximation that can be solved through layerwise and class-level systems without storing historical examples or individual features. The formulation supports prompts, adapters, and low-rank adaptation within the same procedure. Experiments under stochastic blurry streams show that every LOOM variant surpasses analytic baselines, and remains comparable to the full joint multilayer upper bound across all three benchmarks, while reducing statistical storage by more than 70% and cumulative solve time by 36.5%–50.7%. These results show that prototypes provide an effective interface for integrating complementary representations across network depths during continual adaptation.

Keywords: continual learning, lifelong learning, pretrained model, parameter efficient fine-tuning

Contents

1	Introduction	2
2	Related Work	3
3	LOOM: Multilayer Aggregation for GCL	5
3.1	Preliminaries	5
3.2	PTM-based Analytic CL	5
3.3	Cross-layer Moment Integration	7
4	Experiment	9
4.1	Experiment Setup	9
4.2	Result Analysis	10
5	Conclusion	12
	References	14
A	Implementation Details	17
A.1	Experiment Setup	17
A.2	Baselines	18
A.3	Evaluation Metrics	18
A.4	LOOM Algorithm	19
B	Additional Results	20
B.1	Cross-layer Representation Similarity	20
B.2	Resource Costs	20

1 Introduction

General continual learning (GCL) requires models to learn from a non-stationary data stream while remaining capable of prediction throughout training (Buzzega et al., 2020; De Lange et al., 2021; Koh et al., 2021). Classes may recur with changing frequencies, and task boundaries need not provide reliable cues for organizing learning (Moon et al., 2023). Under a single traversal of the stream, adaptation must make effective use of the current observations while preserving knowledge acquired earlier. Pre-trained models (PTMs) provide transferable representations for this purpose, and parameter-efficient tuning (PET) enables adaptation through a small set of trainable parameters (Wang et al., 2022d,c; Hu et al., 2021). Recent GCL methods exploit these representations through prompt selection, initialization, and expert routing (Moon et al., 2023; Kang et al., 2025; Yan et al., 2026). Alongside the organization of adaptation, a central question is how to retain and combine the information in these representations for prediction as the stream evolves.

Analytic continual learning offers a natural approach to information retention by accumulating feature statistics and updating a classifier through least-squares solutions (Zhuang et al., 2022, 2024a,b). With pretrained representations, RanPAC uses nonlinear random expansion to construct features for analytic classification (McDonnell et al., 2024). LayUP instead draws on representations already available at different network depths and fits a joint classifier to their concatenation (Ahrens et al., 2024). This use of intermediate representations extends the information available to the classifier beyond the final layer. However, a full joint classifier retains second-order statistics for every pair of selected layers, so its Gram storage grows quadratically with the number of layers at a fixed feature width. Fitting each layer independently reduces this requirement, but does not explicitly model their interactions within a joint system. These alternatives motivate an analytic classifier that preserves cross-layer relationships without maintaining the full concatenated Gram matrix (see Figure 1).

Our starting point is the structure of these cross-layer statistics. Each feature observation can be expressed as its class prototype plus a residual. The second moment between two layers then decomposes into a sum of count-weighted prototype outer products and a sum of within-class residual outer products. The prototypes provide a common class reference across depths, making it possible to reconstruct a component of every cross-layer block from class feature sums and counts. We use this decomposition to retain the full second-order statistics within each layer and represent interactions between layers through their prototypes. This construction preserves layer-specific variation while allowing all selected layers to participate in a single classifier. It also gives the approximation a precise interpretation: only cross-layer residual interactions are omitted.

Building on this construction, we introduce **LOOM**, a framework for prototype-based multilayer aggregation in GCL. LOOM continually updates shared PET parameters, collects intermediate representations, and accumulates layerwise second moments, class feature sums, and class counts. These statistics define a structured approximation to the joint Gram matrix and support an analytic classifier without retaining historical samples or individual feature vectors. The resulting matrix combines diagonal residual blocks with a low-rank prototype component. This structure admits an exact solution through layerwise solves and a class-level system. For equal-width layers, Gram storage grows linearly with the number of selected layers. LOOM therefore combines shared representation adaptation with a compact joint classifier and accommodates prompts, adapters, and low-rank adaptation (LoRA) (Hu et al., 2021) within the same statistical formulation.

We evaluate LOOM on CIFAR-100, ImageNet-R, and CUB-200 under stochastic blurry GCL streams (Moon et al., 2023). Our evaluation examines both anytime prediction and final performance, with comparisons that separate the effects of adaptation, feature depth, and cross-layer integration. Analyses of the accumulated Gram matrices and layer representations connect the statistical construction to the observed feature structure, while resource measurements quantify the cost of retaining and solving these statistics. Across the three benchmarks, LOOM-LoRA retains anytime accuracy within

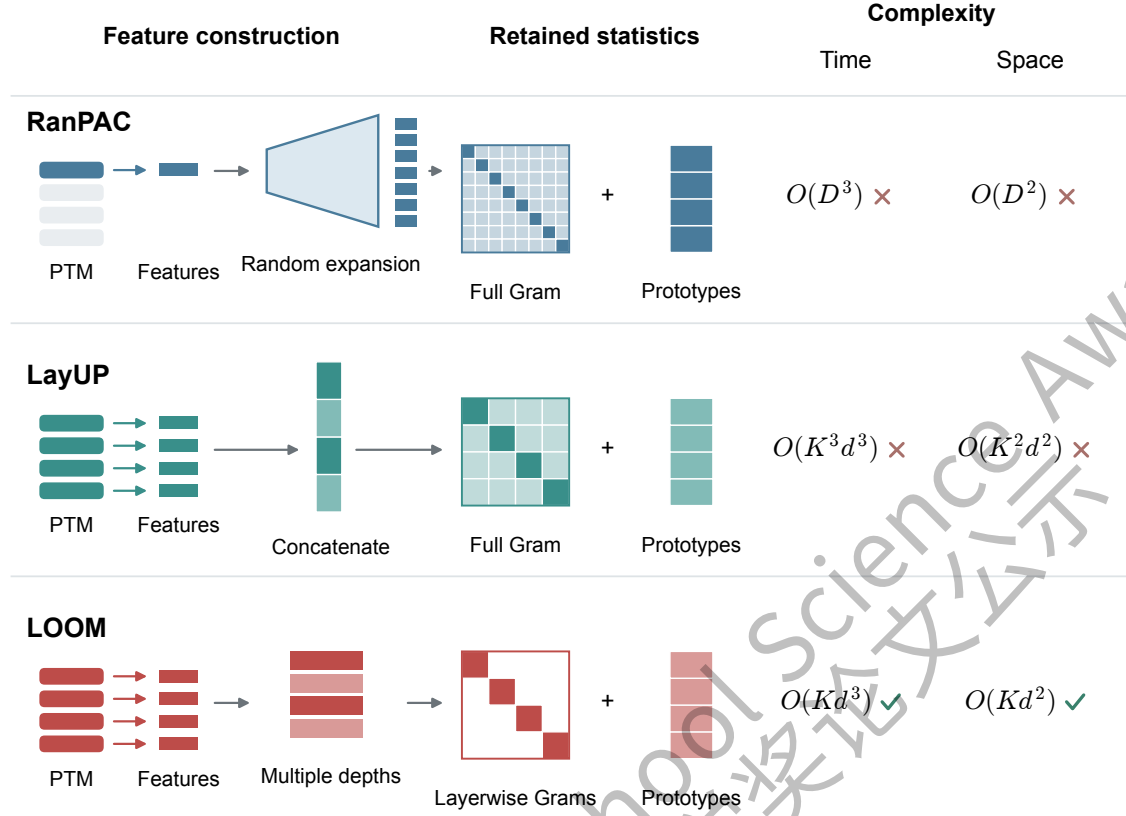


Figure 1: Analytic classifier construction in RanPAC, LayUP, and LOOM using pretrained models (PTMs). RanPAC expands final-layer features; LayUP concatenates layers and retains their full joint Gram; LOOM retains layerwise Gram blocks and reconstructs cross-layer blocks from prototypes. Prototype statistics are accumulated through class feature sums and counts. Time denotes classifier construction from accumulated statistics, and space denotes retained statistics. Dominant orders are shown for $C_t \leq \min(d, D)$, as in our benchmarks, where D is RanPAC’s random expansion width, d the per-layer feature width, K the number of selected layers, and C_t the number of seen classes.

0.59 percentage points of the full joint multilayer ridge classifier while reducing statistical storage by more than 70%. These results demonstrate that prototype-based integration preserves much of the predictive value of joint multilayer statistics under continual adaptation.

2 Related Work

Continual Learning (CL) studies how a model acquires knowledge from changing data distributions while retaining its ability to solve previously encountered problems (Parisi et al., 2019; Wang et al., 2024). Established approaches constrain parameter updates to preserve important knowledge (Kirkpatrick et al., 2017), distill previous predictions (Li & Hoiem, 2018), rehearse selected examples (Rebuffi et al., 2017b), or allocate additional capacity (Rusu et al., 2016). These strategies address different aspects of the *stability-plasticity trade-off*, including interference between old and new knowledge and the transfer of useful representations (Wang et al., 2021, 2022a). Their evaluation commonly follows task-, class-, or domain-incremental protocols, which differ in the evolution of the label space and the availability of task identity (Van de Ven & Tolias, 2019). Pretrained models (PTMs) provide a different starting point: substantial transferable knowledge is available before the stream begins, making the preservation and adaptation of this representation central to subsequent learning. For example,

SLCA combines slow backbone adaptation with classifier alignment to reduce the mismatch between representation learning and incremental classification (Zhang et al., 2023). Parameter-efficient tuning (PET) instead introduces a small set of trainable parameters, through prompts, prefixes, adapters, or low-rank updates, while keeping the pretrained weights fixed (Lester et al., 2021; Li & Liang, 2021; Rebuffi et al., 2017a; Hu et al., 2021).

PTM-based CL methods differ in how they share and select adaptation parameters. A shared module transfers its accumulated adaptation across the stream, whereas multiple modules can specialize but require an appropriate selection mechanism at inference. L2P retrieves prompts from a learned pool using input-dependent queries (Wang et al., 2022d); DualPrompt separates shared and expert prompts (Wang et al., 2022c); and CODA-Prompt composes prompt components through attention (Smith et al., 2023). S-Prompts explores domain-specific prompts (Wang et al., 2022b), while HiDe-Prompt and subsequent hierarchical analyses distinguish expert selection from within-expert prediction and account for the pretrained representation (Kim et al., 2022; Wang et al., 2023). NoRGa formulates prompt attention as a mixture-of-experts and introduces nonlinear residual gating to improve within-task prediction (Le et al., 2024); SD-LoRA extends parameter-efficient adaptation through decoupled learning of low-rank directions and magnitudes (Wu et al., 2025). These methods address different aspects of adaptation capacity, expert selection, and knowledge retention.

Analytic CL accumulates feature statistics for closed-form or recursive classifier updates. ACIL establishes the equivalence between incremental and joint least-squares learning under a fixed feature mapping (Zhuang et al., 2022), while GACL and F-OAL develop analytic learning for generalized class-arrival settings and online class-incremental streams, respectively (Zhuang et al., 2024a,b). With PTMs, RanPAC combines nonlinear random expansion with accumulated feature statistics (McDonnell et al., 2024). Multilayer information has also been exploited through intermediate supervision in MOSE (Yan et al., 2024) and joint analytic classification of concatenated PTM features in LayUP (Ahrens et al., 2024). Subsequent studies investigate distribution drift compensation and scalable statistical classifiers (Rao et al., 2026b,a). Together, these directions motivate compact integration of multilayer information under continual adaptation.

General Continual Learning (GCL) relaxes the clean task segmentation and repeated access to task datasets often assumed in conventional CL (Buzzega et al., 2020; De Lange et al., 2021; Mi et al., 2020). In an online stream, class frequencies and input distributions may change, previously observed classes may reappear, and transitions need not define a reliable task identity. The learner must therefore adapt with limited updates while remaining capable of inference throughout training. Task-free formulations remove explicit task boundaries (Aljundi et al., 2019a), blurry protocols allow classes to recur across sessions and emphasize anytime inference (Koh et al., 2021), and Si-Blurry introduces stochastic variation in class composition and sample allocation (Moon et al., 2023). These conditions couple fast adaptation with retention under class imbalance and limited memory. Replay remains a valid strategy within GCL: gradient-based sample selection, balanced memory construction, and replay objectives have been developed to use a restricted buffer (Aljundi et al., 2019b; Bang et al., 2021; Buzzega et al., 2020). Memory-centered baselines also illustrate how strongly performance can depend on storage and sampling choices (Prabhu et al., 2020).

PTMs support replay-free GCL by supplying reusable representations, but the remaining trainable parameters still face interference and unequal exposure to classes. MVP combines visual prompt tuning with masking under stochastic blurry streams (Moon et al., 2023). MISA improves prompt initialization through forgetting-aware initial-session adaptation using pretraining data and introduces a non-parametric logit mask (Kang et al., 2025). FlyPrompt combines an analytic router in a randomly expanded feature space with temporal-ensemble experts, addressing expert assignment and prediction stability (Yan et al., 2026). These methods emphasize the organization and adaptation of task experts. LOOM investigates a complementary question: how to use representations from multiple depths in a statistical classifier when adaptation is shared across the stream.

3 LOOM: Multilayer Aggregation for GCL

3.1 Preliminaries

We consider a labeled stream $\{\mathcal{B}_t\}_{t=1}^T$, where $\mathcal{B}_t = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^{n_t}$ is the mini-batch received at step t , $\mathbf{x}_{t,i} \in \mathcal{X}$, and $y_{t,i} \in \mathcal{Y}$. Let $\mathcal{C}_t = \bigcup_{s \leq t} \{y_{s,i}\}_{i=1}^{n_s}$ denote the classes observed so far and $C_t = |\mathcal{C}_t|$. The learner updates its state from the current batch and predicts over $\mathcal{C}_t$ without a test-time task identifier. The objective is to maintain accurate predictions on all observed classes as the stream evolves, rather than only on classes present in the latest batch. In conventional CL, task-incremental learning supplies a task identifier at inference, whereas class-incremental learning requires discrimination across an expanding label space without this information; domain-incremental learning preserves the label space while changing the input distribution (Van de Ven & Tolias, 2019). Disjoint class sessions are a common construction for the first two settings, but are not assumed here. We study online GCL with one traversal of the training stream, a bounded number of updates on each current batch, and no storage of past training examples. Class counts and feature statistics may be retained. Retained statistics may grow with the number of observed classes and feature dimensions.

We instantiate this setting with Si-Blurry (Moon et al., 2023). Its global label space is partitioned into disjoint and blurry classes,

$$\mathcal{Y} = \mathcal{Y}^D \cup \mathcal{Y}^B, \quad \mathcal{Y}^D \cap \mathcal{Y}^B = \emptyset, \quad r_D = \frac{|\mathcal{Y}^D|}{|\mathcal{Y}|}. \quad (1)$$

Disjoint classes are assigned to individual sessions, while examples from blurry classes can occur across sessions. Both class subsets are randomly partitioned among sessions with unequal partition sizes. Let $\mathcal{D}^B$ contain all training examples with labels in $\mathcal{Y}^B$; the blurry sample ratio r_B specifies a redistribution budget of $\lfloor r_B |\mathcal{D}^B| \rfloor$ examples, which are pooled from the initial blurry partitions and randomly reassigned across sessions (Moon et al., 2023; Kang et al., 2025). Consequently, a batch may contain both newly introduced and recurring classes with unequal frequencies. Sessions describe the benchmark construction and evaluation schedule; they need not correspond to separable prediction problems. In particular, the protocol does not imply that every pair of sessions must share classes. Throughout, t indexes an observed mini-batch.

Let $f_\theta(\mathbf{x}; \mathbf{p}_t)$ denote a PTM with fixed pretrained weights θ and PET parameters $\mathbf{p}_t$, instantiated by prompts, adapters, or low-rank updates. Although the backbone weights remain fixed, updating $\mathbf{p}_t$ changes the effective feature mapping. LOOM sequentially updates a shared set of PET parameters and integrates the resulting intermediate representations with an analytic classifier. Here, *analytic* refers only to the closed-form classifier solution conditioned on the extracted features.

3.2 PTM-based Analytic CL

We first describe analytic classification for a fixed feature mapping. Write $\mathbf{h}(\mathbf{x}) = f_\theta(\mathbf{x}; \mathbf{p}) \in \mathbb{R}^d$ for a column feature vector, with $\mathbf{p}$ held fixed. The classifier may use this representation directly or apply a fixed transformation $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^D$. For example, the random expansion used in RanPAC takes the form $\mathbf{z}(\mathbf{x}) = \sigma(\mathbf{R}^\top \mathbf{h}(\mathbf{x}))$, where $\mathbf{R} \in \mathbb{R}^{d \times D}$ is sampled once and σ is an elementwise nonlinearity (McDonnell et al., 2024). For batch s , let $\mathbf{Z}_s \in \mathbb{R}^{n_s \times D}$ contain feature vectors as rows and let $\mathbf{Y}_s^{(t)} \in \mathbb{R}^{n_s \times C_t}$ contain one-hot labels in a common class ordering at step t . We index classes in order of first appearance and identify each label with its fixed column index. When new classes appear, historical label matrices are conceptually extended with zero columns; historical examples need not be retained. The regularized least-squares classifier is

$$\mathbf{W}_t = \arg \min_{\mathbf{W} \in \mathbb{R}^{D \times C_t}} \sum_{s=1}^t \|\mathbf{Z}_s \mathbf{W} - \mathbf{Y}_s^{(t)}\|_F^2 + \lambda \|\mathbf{W}\|_F^2, \quad \lambda > 0. \quad (2)$$

We use a summed loss, so λ regularizes the accumulated objective without an implicit normalization by the number of observations. Its unique solution depends only on two sufficient statistics:

$$\mathbf{G}_t = \sum_{s=1}^t \mathbf{Z}_s^\top \mathbf{Z}_s, \quad \mathbf{Q}_t = \sum_{s=1}^t \mathbf{Z}_s^\top \mathbf{Y}_s^{(t)}, \quad \mathbf{W}_t = (\mathbf{G}_t + \lambda \mathbf{I}_D)^{-1} \mathbf{Q}_t, \quad (3)$$

where $\mathbf{G}_t \in \mathbb{R}^{D \times D}$ is the uncentered feature Gram matrix and $\mathbf{Q}_t \in \mathbb{R}^{D \times C_t}$ is the feature-label cross-product matrix. The identity matrix is denoted by $\mathbf{I}_D$. The solution can be computed by solving the corresponding linear system without explicitly forming the inverse.

These statistics admit additive online updates,

$$\mathbf{G}_t = \mathbf{G}_{t-1} + \mathbf{Z}_t^\top \mathbf{Z}_t, \quad \mathbf{Q}_t = \text{pad}_{C_t}(\mathbf{Q}_{t-1}) + \mathbf{Z}_t^\top \mathbf{Y}_t^{(t)}, \quad (4)$$

initialized from zero, where pad_{C_t} appends zero columns for newly observed classes. Each example contributes once, after which it can be discarded. Prediction uses

$$\hat{y}_t(\mathbf{x}) = \arg \max_{c \in \mathcal{C}_t} [\mathbf{W}_t^\top \mathbf{z}(\mathbf{x})]_c. \quad (5)$$

For fixed features and the same regularization, these updates recover the joint ridge solution on all observations, independent of their partition into batches (Zhuang et al., 2022). The equivalence concerns the ridge objective; previous-class accuracy can still change as observations accumulate. It also clarifies the relation to prototypes. For each observed class $c \in \mathcal{C}_t$ with $N_{t,c}$ examples, the prototype $\boldsymbol{\mu}_{t,c}$ and class-feature sum satisfy

$$\boldsymbol{\mu}_{t,c} = \frac{1}{N_{t,c}} \sum_{s=1}^t \sum_{i=1}^{n_s} \mathbf{1}\{y_{s,i} = c\} \mathbf{z}(\mathbf{x}_{s,i}), \quad \mathbf{q}_{t,c} = N_{t,c} \boldsymbol{\mu}_{t,c}, \quad (6)$$

where $\mathbf{q}_{t,c}$ is column c of $\mathbf{Q}_t$. Thus, prototypes together with class counts exactly recover the feature-label statistic, but prototypes alone do not determine $\mathbf{G}_t$. Replacing class sums by unweighted class means would change the solution in Eq. (2), especially under class imbalance. Likewise, nearest-prototype classification and analytic classification are distinct: the latter transforms class sums using the shared second-order statistic.

The same construction extends to multiple network depths. For selected layers $\ell_1, \dots, \ell_K$, let $\mathbf{h}^{(\ell_k)}(\mathbf{x}) \in \mathbb{R}^{d_k}$ and define

$$\mathbf{z}(\mathbf{x}) = [\mathbf{h}^{(\ell_1)}(\mathbf{x})^\top, \dots, \mathbf{h}^{(\ell_K)}(\mathbf{x})^\top]^\top, \quad D = \sum_{k=1}^K d_k. \quad (7)$$

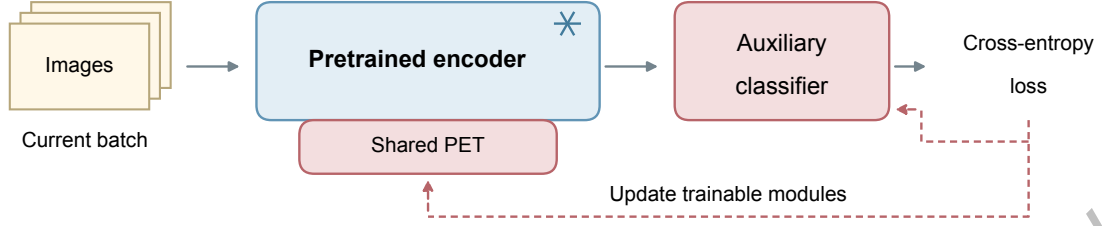
LayUP uses such a concatenated representation for joint analytic classification (Ahrens et al., 2024). Its Gram matrix contains both within-layer and cross-layer blocks:

$$\mathbf{G}_t^{(a,b)} = \sum_{s=1}^t \sum_{i=1}^{n_s} \mathbf{h}^{(\ell_a)}(\mathbf{x}_{s,i}) \mathbf{h}^{(\ell_b)}(\mathbf{x}_{s,i})^\top, \quad a, b \in \{1, \dots, K\}. \quad (8)$$

The off-diagonal blocks retain uncentered cross-layer second-order products; fitting separate ridge classifiers and averaging their scores does not, in general, recover this joint solution (Ahrens et al., 2024). Full accumulation requires $O(D^2 + DC_t)$ storage, and a dense factorization costs $O(D^3)$, followed by $O(D^2 C_t)$ work to solve for the classifier. These costs motivate retaining useful cross-layer information through a more compact statistical representation.

Finally, the fixed-feature condition must be distinguished from freezing only the pretrained weights. If PET parameters change while statistics are collected, the accumulated matrices contain features evaluated at their respective collection times. They still define an exact ridge problem over those stored feature observations, but generally differ from the statistics obtained by re-encoding all past inputs with the current PET parameters. For a given set of feature observations, analytic accumulation therefore avoids replay; by itself, however, it does not correct representation drift.

① Online adaptation



② Prototype-based multilayer integration

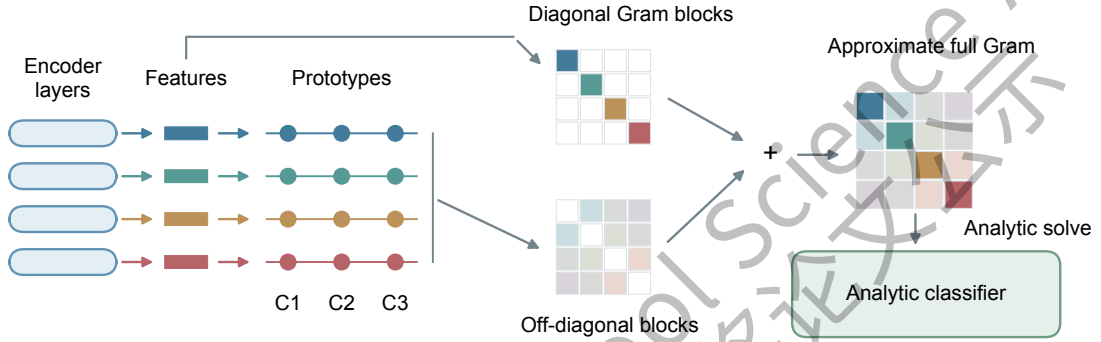


Figure 2: **LOOM overview.** (1) Shared parameter-efficient tuning (PET) modules and an auxiliary classifier (rose) adapt to each batch while the pretrained encoder (blue snowflake) remains frozen. (2) The selected encoder layers provide features for two parallel statistics: exact within-layer Gram blocks and class prototypes. Colored strands group prototype nodes by layer, with classes aligned horizontally. Products of these prototypes reconstruct the off-diagonal blocks. Combining these blocks with the exact diagonal Gram blocks yields an approximation to the full Gram matrix.

3.3 Cross-layer Moment Integration

LOOM integrates representations from different depths through class prototypes while retaining the full second-order statistics within each layer. Figure 2 summarizes its online updates and analytic classifier construction. The construction preserves a shared multilayer classifier, as in joint analytic classification (Ahrens et al., 2024), with a structured approximation to the cross-layer moments. At stream step s , we update the shared PET parameters and an auxiliary linear head on the current batch, then collect each example’s intermediate features once using the updated parameters $\mathbf{p}_s$. Let $\mathbf{z}_{s,i}$ denote the concatenated feature of example $(\mathbf{x}_{s,i}, \mathbf{y}_{s,i})$, and let $\mathbf{h}_{s,i}^{(\ell_a)}$ be its block from layer ℓ_a . In this subsection, $\mathbf{G}_t$, $\mathbf{Q}_t$, and $\boldsymbol{\mu}_{t,c}$ refer to these collection-time feature observations. Thus the following construction applies to the evolving representation without requiring historical inputs.

To separate the information carried by prototypes from the remaining feature variation, define the diagonal count matrix $\mathbf{N}_t = \text{diag}(N_{t,c} : c \in \mathcal{C}_t)$ and the count-weighted prototype factor

$$\mathbf{U}_t = \mathbf{Q}_t \mathbf{N}_t^{-1/2} = [\sqrt{N_{t,c}} \boldsymbol{\mu}_{t,c}]_{c \in \mathcal{C}_t} \in \mathbb{R}^{D \times C_t}. \quad (9)$$

All diagonal entries of $\mathbf{N}_t$ are positive because it includes only observed classes. Write $\mathbf{U}_t^{(a)}$ and $\boldsymbol{\mu}_{t,c}^{(a)}$ for the rows corresponding to layer ℓ_a . Centering each observation by its class prototype gives the

exact decomposition

$$\begin{aligned} \mathbf{G}_t^{(a,b)} &= \mathbf{U}_t^{(a)} \mathbf{U}_t^{(b)\top} + \mathbf{S}_t^{(a,b)}, \\ \mathbf{S}_t^{(a,b)} &= \sum_{s=1}^t \sum_{i=1}^{n_s} (\mathbf{h}_{s,i}^{(\ell_a)} - \boldsymbol{\mu}_{t,y_{s,i}}^{(a)}) (\mathbf{h}_{s,i}^{(\ell_b)} - \boldsymbol{\mu}_{t,y_{s,i}}^{(b)})^\top. \end{aligned} \quad (10)$$

The first term equals $\sum_{c \in \mathcal{C}_t} N_{t,c} \boldsymbol{\mu}_{t,c}^{(a)} \boldsymbol{\mu}_{t,c}^{(b)\top}$ and captures cross-layer products of class prototypes. The second term contains the corresponding within-class residual products. The mixed prototype-residual terms vanish because residuals sum to zero within each class. This is a decomposition of uncentered second moments and does not require a distributional assumption on the features.

LOOM retains the prototype component across all layers and the residual component within each layer:

$$\begin{aligned} \tilde{\mathbf{G}}_t &= \text{blockdiag}(\mathbf{S}_t^{(1,1)}, \dots, \mathbf{S}_t^{(K,K)}) + \mathbf{U}_t \mathbf{U}_t^\top, \\ \mathbf{S}_t^{(a,a)} &= \mathbf{G}_t^{(a,a)} - \mathbf{U}_t^{(a)} \mathbf{U}_t^{(a)\top}. \end{aligned} \quad (11)$$

Consequently, $\tilde{\mathbf{G}}_t^{(a,a)} = \mathbf{G}_t^{(a,a)}$: every within-layer second moment is preserved exactly. For $a \neq b$, the cross-layer block is reconstructed from the prototypes and their class counts. The prototype component $\mathbf{U}_t \mathbf{U}_t^\top$ has rank at most C_t . Layers therefore participate in a joint system through the same class-level statistics, while their individual within-class variation remains represented. The omitted information is precisely $\mathbf{S}_t^{(a,b)}$ for $a \neq b$; the approximation becomes exact when these aggregate cross-layer residual blocks vanish. Since each diagonal residual block is a sum of outer products, it is positive semidefinite, as is $\mathbf{U}_t \mathbf{U}_t^\top$. Hence $\tilde{\mathbf{G}}_t$ is positive semidefinite and $\tilde{\mathbf{G}}_t + \lambda \mathbf{I}_D$ is positive definite for $\lambda > 0$.

The analytic classifier uses the unchanged feature-label statistic $\mathbf{Q}_t$ together with $\tilde{\mathbf{G}}_t$. Let $\mathbf{A}_t = \text{blockdiag}(\mathbf{S}_t^{(1,1)} + \lambda \mathbf{I}_{d_1}, \dots, \mathbf{S}_t^{(K,K)} + \lambda \mathbf{I}_{d_K})$. Applying the Woodbury identity (Hager, 1989) and using $\mathbf{Q}_t = \mathbf{U}_t \mathbf{N}_t^{1/2}$ yields

$$\begin{aligned} \tilde{\mathbf{W}}_t &= (\tilde{\mathbf{G}}_t + \lambda \mathbf{I}_D)^{-1} \mathbf{Q}_t \\ &= \mathbf{A}_t^{-1} \mathbf{U}_t (\mathbf{I}_{C_t} + \mathbf{U}_t^\top \mathbf{A}_t^{-1} \mathbf{U}_t)^{-1} \mathbf{N}_t^{1/2}. \end{aligned} \quad (12)$$

This expression requires independent solves with the layerwise blocks of $\mathbf{A}_t$ and one $C_t \times C_t$ system. It introduces no further approximation beyond the moment structure in Eq. (11). At inference, the concatenated representation $\mathbf{z}_t(\mathbf{x})$ is extracted using the current PET parameters $\mathbf{p}_t$, and prediction follows Eq. (5) with $\tilde{\mathbf{W}}_t$ and $\mathbf{z}_t(\mathbf{x})$. The auxiliary gradient-trained head supports adaptation, while the analytic classifier determines the prediction.

LOOM retains only the diagonal Gram blocks, class feature sums, and class counts. Each incoming batch adds its within-layer products to $\mathbf{G}_t^{(a,a)}$, updates $\mathbf{Q}_t$ as in Eq. (4), and increments the class counts. Prototypes and residual diagonal blocks are then recovered from these cumulative statistics, so recurring classes update the cross-layer approximation without storing individual historical features. Although the retained statistics accumulate additively, the reconstructed cross-layer blocks are recomputed from the cumulative prototypes. The statistical storage is $O(\sum_{a=1}^K d_a^2 + DC_t + C_t)$, compared with $O(D^2 + DC_t)$ for a full joint Gram matrix. With equal layer width d , this reduces the Gram storage from $O(K^2 d^2)$ to $O(K d^2)$. The structured solve costs $O(K d^3 + K d^2 C_t + K d C_t^2 + C_t^3)$ and uses an additional $O(C_t^2)$ workspace for the class-level system. When $C_t \ll D$, it avoids factorizing a dense $D \times D$ matrix while preserving prototype-based cross-layer interactions and exact within-layer moments. Algorithm 1 in Appendix A.4 summarizes the online updates and inference procedure.

Table 1: GCL performance of representative methods with the Sup-21K backbone. Bold and underlined values indicate the best and second-best mean accuracies in each column, respectively.

Method	CIFAR-100			ImageNet-R			CUB-200		
	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$	$F(\downarrow)$	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$	$F(\downarrow)$	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$	$F(\downarrow)$
SeqFT	19.71 $\pm$ 3.39	10.42 $\pm$ 4.92	32.25 $\pm$ 16.74	7.51 $\pm$ 3.94	2.29 $\pm$ 0.85	74.79 $\pm$ 23.89	3.47 $\pm$ 0.41	1.49 $\pm$ 0.42	54.01 $\pm$ 24.57
Linear Probe	49.69 $\pm$ 6.09	23.07 $\pm$ 7.33	11.59 $\pm$ 1.63	29.24 $\pm$ 1.26	16.87 $\pm$ 3.14	18.95 $\pm$ 1.44	28.96 $\pm$ 2.46	17.33 $\pm$ 3.08	28.82 $\pm$ 3.02
SeqFT w/ SL	64.90 $\pm$ 7.18	62.06 $\pm$ 1.89	21.09 $\pm$ 3.82	47.20 $\pm$ 1.47	39.60 $\pm$ 2.43	33.99 $\pm$ 7.14	56.16 $\pm$ 4.32	56.50 $\pm$ 3.08	29.05 $\pm$ 1.73
EWC	55.94 $\pm$ 8.36	58.13 $\pm$ 1.77	41.13 $\pm$ 4.59	36.29 $\pm$ 1.51	33.92 $\pm$ 1.86	50.28 $\pm$ 12.28	40.27 $\pm$ 4.03	35.57 $\pm$ 3.25	68.73 $\pm$ 13.00
LwF	62.14 $\pm$ 8.35	71.59 $\pm$ 3.20	18.47 $\pm$ 7.95	37.91 $\pm$ 1.60	36.66 $\pm$ 2.35	48.01 $\pm$ 13.05	41.08 $\pm$ 3.94	37.00 $\pm$ 3.24	67.05 $\pm$ 13.30
L2P	76.23 $\pm$ 2.73	79.11 $\pm$ 1.43	11.53 $\pm$ 1.44	44.40 $\pm$ 1.03	42.03 $\pm$ 1.72	19.16 $\pm$ 1.92	64.30 $\pm$ 2.18	61.42 $\pm$ 2.13	29.15 $\pm$ 3.19
DualPrompt	76.04 $\pm$ 3.32	76.62 $\pm$ 0.74	11.42 $\pm$ 0.91	46.13 $\pm$ 1.94	40.80 $\pm$ 1.04	18.24 $\pm$ 4.34	65.03 $\pm$ 2.24	62.43 $\pm$ 1.78	27.18 $\pm$ 2.91
CODA-P	79.13 $\pm$ 3.06	80.91 $\pm$ 0.70	9.95 $\pm$ 1.30	51.87 $\pm$ 2.81	48.09 $\pm$ 2.75	18.49 $\pm$ 2.44	66.01 $\pm$ 2.20	62.90 $\pm$ 2.46	27.43 $\pm$ 3.42
MVP	67.74 $\pm$ 4.96	63.22 $\pm$ 0.69	33.19 $\pm$ 2.33	39.50 $\pm$ 1.41	32.63 $\pm$ 3.95	43.39 $\pm$ 4.23	54.69 $\pm$ 3.14	50.07 $\pm$ 3.86	47.61 $\pm$ 5.73
MISA	80.35 $\pm$ 2.39	80.75 $\pm$ 1.24	9.67 $\pm$ 1.39	51.52 $\pm$ 2.09	45.08 $\pm$ 1.43	21.46 $\pm$ 4.25	65.40 $\pm$ 3.01	60.20 $\pm$ 1.82	30.06 $\pm$ 3.70
S-Prompt++	80.21 $\pm$ 2.55	83.48 $\pm$ 1.20	9.22 $\pm$ 1.43	52.14 $\pm$ 1.65	49.13 $\pm$ 1.60	17.59 $\pm$ 2.51	66.61 $\pm$ 2.21	64.73 $\pm$ 2.25	25.88 $\pm$ 4.21
HiDe-Prompt	77.10 $\pm$ 3.81	81.77 $\pm$ 2.00	8.80 $\pm$ 1.07	53.77 $\pm$ 1.09	49.87 $\pm$ 3.01	15.34 $\pm$ 2.00	67.05 $\pm$ 2.37	67.12 $\pm$ 0.50	24.21 $\pm$ 2.45
NoRGa	78.89 $\pm$ 3.33	83.03 $\pm$ 1.20	8.61 $\pm$ 1.04	54.12 $\pm$ 1.37	50.09 $\pm$ 3.66	15.79 $\pm$ 2.19	67.16 $\pm$ 2.44	67.06 $\pm$ 0.58	24.29 $\pm$ 2.59
SD-LoRA	79.26 $\pm$ 2.21	78.91 $\pm$ 2.48	16.72 $\pm$ 2.31	55.51 $\pm$ 1.30	51.97 $\pm$ 3.09	32.18 $\pm$ 4.21	64.12 $\pm$ 2.02	60.57 $\pm$ 0.77	31.31 $\pm$ 3.96
FlyPrompt	83.24 $\pm$ 2.23	86.76 $\pm$ 0.73	5.03 $\pm$ 1.16	56.58 $\pm$ 1.47	55.27 $\pm$ 0.91	13.42 $\pm$ 1.74	70.64 $\pm$ 2.85	73.40 $\pm$ 1.88	10.97 $\pm$ 1.52
RanPAC	77.88 $\pm$ 4.28	86.52 $\pm$ 1.15	4.17 $\pm$ 0.62	53.18 $\pm$ 2.22	54.71 $\pm$ 2.48	9.35 $\pm$ 1.91	74.64 $\pm$ 3.89	82.30 $\pm$ 1.09	6.54 $\pm$ 1.25
LayUP	71.39 $\pm$ 3.50	82.37 $\pm$ 0.00	2.64 $\pm$ 1.85	42.32 $\pm$ 0.83	49.62 $\pm$ 0.00	4.69 $\pm$ 1.13	74.44 $\pm$ 3.71	84.01 $\pm$ 0.00	1.08 $\pm$ 2.36
LOOM-Prompt	79.00 $\pm$ 3.34	86.97 $\pm$ 0.23	6.68 $\pm$ 0.88	58.95 $\pm$ 0.94	56.36 $\pm$ 2.55	19.00 $\pm$ 1.76	78.41 $\pm$ 3.78	81.78 $\pm$ 1.79	10.20 $\pm$ 2.11
LOOM-Adapter	79.25 $\pm$ 3.10	86.46 $\pm$ 0.84	7.71 $\pm$ 1.53	63.02 $\pm$ 1.31	61.14 $\pm$ 1.14	17.39 $\pm$ 2.78	77.75 $\pm$ 4.24	81.08 $\pm$ 1.82	11.21 $\pm$ 2.82
LOOM-LoRA	79.11 $\pm$ 3.37	86.45 $\pm$ 0.72	7.33 $\pm$ 1.44	63.95 $\pm$ 0.98	60.41 $\pm$ 1.78	20.20 $\pm$ 2.11	78.21 $\pm$ 3.83	81.43 $\pm$ 1.52	10.20 $\pm$ 1.48

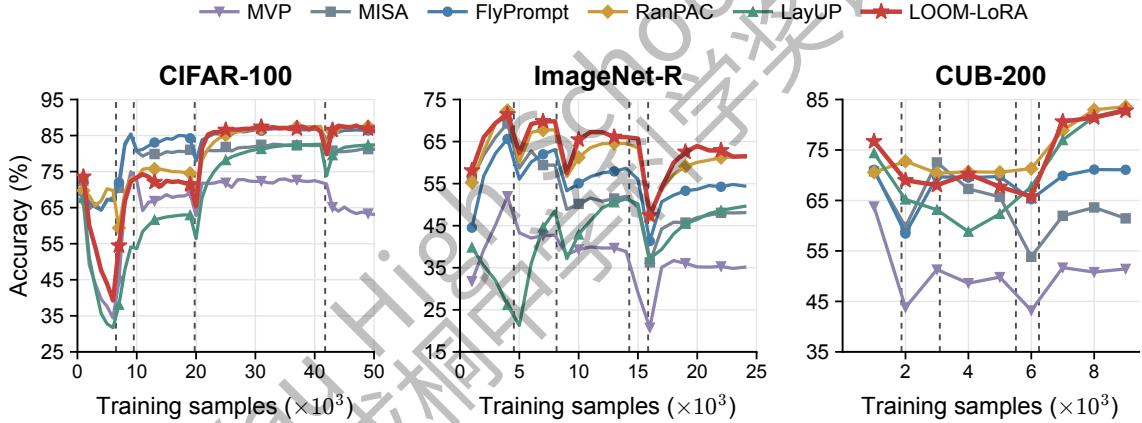


Figure 3: Anytime accuracy on three benchmarks with Sup-21K for six representative methods. LOOM-LoRA uses the last four layers with $\lambda = 10^3$. Black vertical dashed lines mark the actual Si-Blurry stage transitions. Curves connect periodic evaluations at scheduled 1,000-sample intervals.

4 Experiment

4.1 Experiment Setup

Benchmarks. We study Si-Blurry GCL (Moon et al., 2023; Kang et al., 2025) on CIFAR-100 (Krizhevsky et al., 2009), ImageNet-R (Hendrycks et al., 2021), and CUB-200 (Wah et al., 2011), covering general object recognition, visual renditions, and fine-grained recognition, respectively. The datasets contain 100, 200, and 200 classes. Unless otherwise specified, each stream consists of five sessions with $r_D = 0.5$ and $r_B = 0.1$ and uses stochastic class and sample allocation. Training traverses each stream once, without retaining historical images, and evaluation covers all classes observed so far without a test-time task identifier. We report average anytime accuracy A_{auc} , final accuracy A_{last} , and classwise forgetting F , with higher accuracy and lower forgetting preferred. Anytime evaluation is performed at 1,000-sample intervals. Results are reported as the mean and standard deviation over five random

Table 2: Ablation studies of LOOM with Sup-21K under a frozen encoder or sequential LoRA. Ridge regression classifiers use $\lambda = 10^3$; multilayer classifiers use the last four layers. Bold and underlined values denote the best and second-best means within each adaptation setting, respectively. The full joint classifier is shown in gray as an upper bound and is excluded from the ranking.

Classifier	CIFAR-100		ImageNet-R		CUB-200	
	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$	$A_{\text{auc}}(\uparrow)$	$A_{\text{last}}(\uparrow)$
<i>Frozen encoder</i>						
Final-layer FC	71.46 ± 2.63	73.27 ± 1.22	41.56 ± 0.92	37.26 ± 1.54	64.26 ± 2.31	61.66 ± 2.04
Final-layer ridge	67.70 ± 4.12	77.58 ± 0.00	44.80 ± 0.68	44.07 ± 0.00	<u>73.94</u> ± 3.67	79.90 ± 0.00
Independent ridge	65.23 ± 5.38	<u>78.99</u> ± 0.00	<u>44.94</u> ± 0.98	<u>45.50</u> ± 0.00	71.73 ± 3.99	<u>82.10</u> ± 0.00
LOOM	<u>70.33</u> ± 3.80	79.76 ± 0.00	47.48 ± 0.89	47.53 ± 0.00	76.55 ± 3.74	83.67 ± 0.00
Joint (upper bound)	71.07 ± 3.67	80.93 ± 0.00	47.91 ± 0.79	48.77 ± 0.00	77.07 ± 3.54	83.80 ± 0.00
<i>Sequential LoRA</i>						
Final-layer FC	81.71 ± 2.01	83.05 ± 1.05	61.07 ± 1.12	56.08 ± 0.86	67.48 ± 2.22	64.09 ± 3.01
Final-layer ridge	78.15 ± 3.38	87.28 ± 0.46	<u>62.58</u> ± 1.03	61.61 ± 0.77	<u>76.30</u> ± 3.78	<u>80.73</u> ± 0.79
Independent ridge	73.45 ± 5.07	84.95 ± 0.74	60.14 ± 1.01	54.28 ± 3.63	74.23 ± 4.23	80.20 ± 1.75
LOOM	<u>79.11</u> ± 3.37	<u>86.45</u> ± 0.72	63.95 ± 0.98	<u>60.41</u> ± 1.78	78.21 ± 3.83	81.43 ± 1.52
Joint (upper bound)	79.70 ± 3.10	86.78 ± 0.68	64.06 ± 1.01	60.47 ± 2.26	78.53 ± 3.61	82.31 ± 0.99

seeds. Metric definitions are specified in Appendix A.

Baselines. We include sequential fine-tuning, its slow-backbone variant, and linear probing, together with EWC (Kirkpatrick et al., 2017) and LwF (Li & Hoiem, 2018). PTM-based comparisons cover L2P (Wang et al., 2022d), DualPrompt (Wang et al., 2022c), CODA-Prompt (Smith et al., 2023), S-Prompt++ (Wang et al., 2022b), HiDe-Prompt (Wang et al., 2023), NoRGa (Le et al., 2024), and SD-LoRA (Wu et al., 2025). MVP (Moon et al., 2023), MISA (Kang et al., 2025), and Fly-Prompt (Yan et al., 2026) provide comparisons developed for GCL. RanPAC (McDonnell et al., 2024) and LayUP (Ahrens et al., 2024) represent analytic classification using randomly expanded and multilayer features. Reference results for the established gradient-based methods follow the comparisons reported by Yan et al. (2026). To isolate the contribution of cross-layer integration, we additionally compare three classifier constructions under matched shared sequential PET adaptation and identical streams: a final-layer ridge classifier, independently fitted multilayer classifiers, and a full joint multilayer classifier.

Backbones and implementation. We use ViT-B/16 (Dosovitskiy et al., 2020), with 12 transformer blocks and embedding dimension 768, initialized by supervised pretraining on ImageNet-21K (Sup-21K) (Ridnik et al., 2021). LOOM is instantiated with prompts, adapters, and LoRA, each shared across the stream while the pretrained encoder weights remain fixed. We optimize the PET parameters and an auxiliary linear head with Adam at learning rate 5×10^{-3} , zero weight decay, batch size 64, and three updates per incoming batch. By default, the analytic classifier uses the final four block representations and a fixed ridge coefficient $\lambda = 10^3$. More details are specified in Appendix A.

4.2 Result Analysis

Overall Performance. Table 1 reports performance on the three benchmarks. On CIFAR-100, Fly-Prompt achieves the highest anytime accuracy, while LOOM-Prompt attains the highest final accuracy. On ImageNet-R, LOOM occupies the top two positions in both anytime and final accuracy; LOOM-LoRA reaches the highest A_{auc} at 63.95%, exceeding RanPAC by 10.77 percentage points and FlyPrompt by 7.37 points. On CUB-200, LOOM-Prompt achieves the highest anytime accuracy

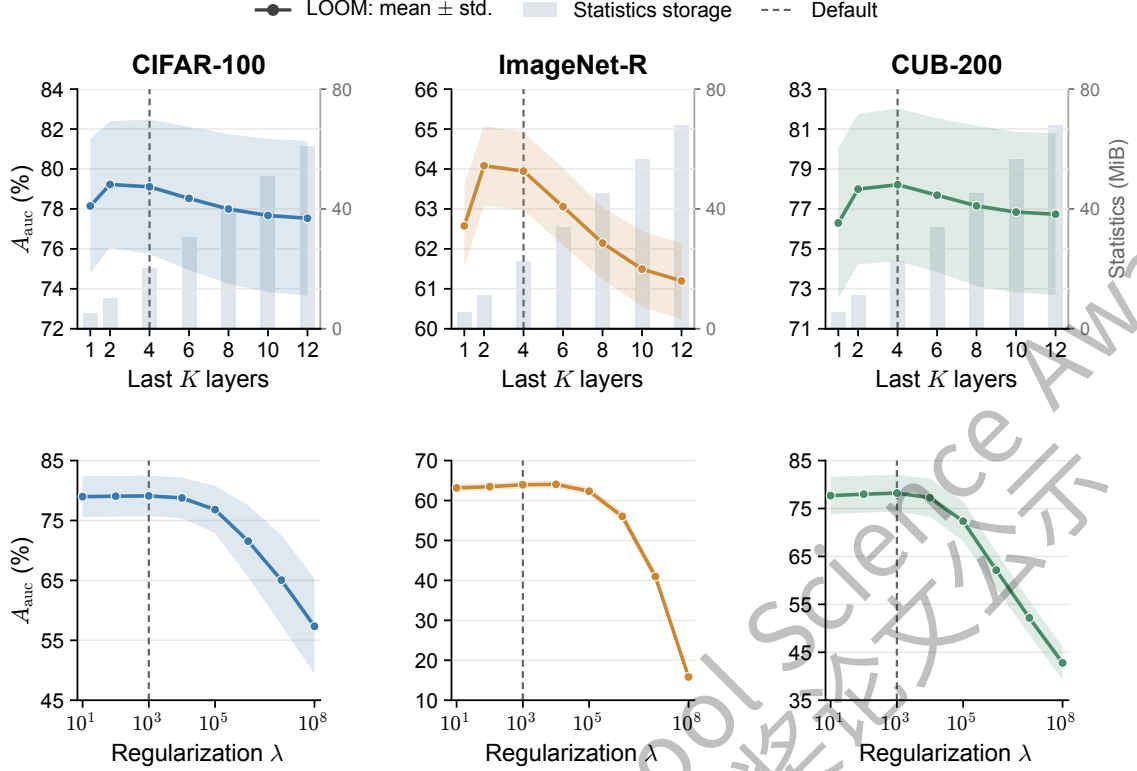


Figure 4: Sensitivity of LOOM-LoRA with Sup-21K. Top: aggregating the last K layers with $\lambda = 10^3$; bars show the retained statistics in MiB, excluding classifier weights. Bottom: varying λ with the last four layers. Curves and shaded regions show five-seed means and standard deviations. Dashed lines mark the default settings. Case $K = 1$ denotes the single-layer ridge classifier.

at 78.41%, compared with 74.64% for RanPAC and 70.64% for FlyPrompt, while LayUP retains the highest final accuracy. The strongest adaptation module varies across datasets: adapters perform best on CIFAR-100, LoRA on ImageNet-R, and prompts on CUB-200 in terms of anytime accuracy. The differences between the anytime and final rankings indicate that the gains reflect how information is used throughout the stream rather than the final model alone. Figure 3 illustrates these dynamics on a common stream realization, where changes in class composition produce substantial fluctuations in accuracy. Across all three benchmarks and all adaptation modules, LOOM consistently improves on LayUP in A_{auc} , showing that shared adaptation and multilayer analytic classification provide a strong combination for prediction throughout the stream.

Ablation Study. Table 2 separates the effects of adaptation and classifier construction. With frozen representations, LOOM improves on final-layer ridge by 2.63, 2.68, and 2.61 points in A_{auc} on CIFAR-100, ImageNet-R, and CUB-200, respectively. Sequential LoRA further raises LOOM’s anytime accuracy by 8.78, 16.46, and 1.66 points, showing that the benefit of adaptation depends on the dataset. Under the same sequential adaptation, LOOM exceeds independently fitted multilayer classifiers by 5.66, 3.81, and 3.99 points. This comparison supports modeling cross-layer relationships within a joint classifier rather than combining independently learned predictions. The full joint classifier serves as an upper bound because it retains every cross-layer Gram block. LOOM approaches this bound closely, with A_{auc} gaps of only 0.59, 0.12, and 0.31 points, demonstrating that prototype-based reconstruction preserves most of the predictive benefit of the full cross-layer statistics in a substantially more compact form.

Table 3: Resource costs with Sup-21K and sequential LoRA on CIFAR-100, averaged over five seeds. Parameter counts include the encoder, LoRA, and auxiliary head; statistical storage excludes classifier weights. Peak denotes peak allocated GPU memory. Solve time is cumulative, whereas wall time includes data loading and evaluation.

Classifier	Parameters (M)		Statistics	Peak	Solve	Wall
	Total	Trainable	(MiB)	(GiB)	(s)	(min)
Final-layer ridge	85.95	0.154	5.09	5.25	0.39	23.30
Independent ridge	85.95	0.154	20.34	5.27	0.62	23.55
Full joint ridge	85.95	0.154	74.34	5.32	1.26	23.45
LOOM	85.95	0.154	20.34	5.27	0.80	23.49

Detailed Analysis. Figure 4 examines feature depth and regularization. At $\lambda = 10^3$, aggregating four layers improves A_{auc} over the final layer by 0.96, 1.37, and 1.92 points across the three datasets. Two and four layers yield similar average anytime accuracy, while adding earlier layers progressively reduces performance. Useful multilayer integration therefore depends on the selected representations rather than their number alone. We use four layers as the common configuration. Within this configuration, $\lambda = 10^3$ achieves the highest average A_{auc} across the benchmarks; ImageNet-R individually favors 10^4 . Much larger coefficients reduce accuracy on all three datasets. Table 3 reports the resource costs on CIFAR-100, with all three benchmarks included in Appendix Table 4. Relative to full joint ridge, LOOM reduces statistical storage by 72.6%, 70.4%, and 70.4%, and cumulative solve time by 36.5%, 50.7%, and 39.0% on CIFAR-100, ImageNet-R, and CUB-200, respectively. LOOM therefore achieves consistently lower resource use through a substantially smaller statistical state and faster analytic solves, with the same encoder and adaptation parameters.

Figure 5 examines the accumulated statistics in the six-layer configuration. LOOM preserves the layer-diagonal blocks exactly and reconstructs the cross-layer blocks from prototypes. The relative Frobenius errors over the off-diagonal blocks, averaged over five seeds, are 1.83%, 4.04%, and 2.17% on CIFAR-100, ImageNet-R, and CUB-200. These measurements directly quantify the information omitted by the approximation. Together with the joint-classifier ablation, the reconstruction results connect the proposed statistical structure to its predictive behavior. We further examine representation similarity across depths in Appendix B.1 to support the layerwise expertise argument.

5 Conclusion

In this work, we introduced **LOOM**, a prototype-based framework for integrating multilayer representations in GCL. LOOM combines shared parameter-efficient adaptation with an analytic classifier that preserves exact within-layer second moments and reconstructs cross-layer interactions from cumulative prototypes. This structure allows multiple network depths to participate in a single classifier without retaining the full concatenated Gram matrix or storing historical examples. Across three stochastic blurry benchmarks, LOOM consistently outperforms the multilayer analytic baseline LayUP and approaches the full joint classifier within 0.59 percentage points, while reducing statistical storage by more than 70%. The ablations and matrix analyses further show that both continual adaptation and cross-layer integration contribute to this performance.

Our evaluation is limited to image classification benchmarks, and the prototype-based approximation omits cross-layer residual correlations. Although the resulting gap to the full joint classifier is small in our experiments, richer approximations may capture these correlations when representations or data streams become more complex. Future work can extend LOOM beyond vision to language and embodied intelligence settings, where observations, tasks, and interaction contexts evolve continuously. Developing compact analytic mechanisms for these settings may help address broader and more practical GCL problems.

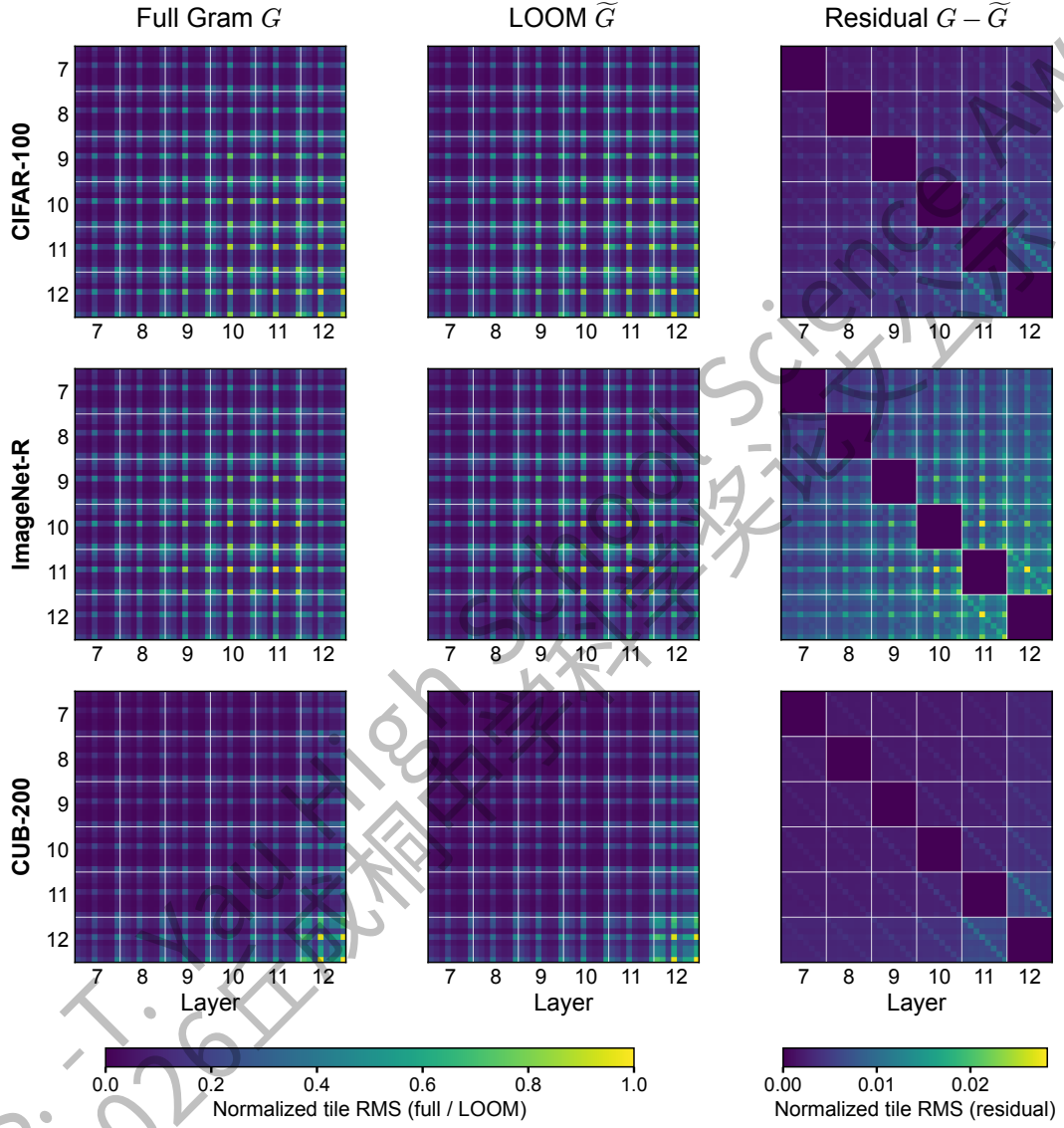


Figure 5: Final accumulated Gram statistics with sequential LoRA in a six-layer configuration. Columns show the full Gram, the LOOM approximation, and their residual. Each pixel summarizes a 96×96 feature-coordinate tile by its root-mean-square value; white lines separate layers 7–12. All three panels are normalized by the maximum tile value of the full Gram, then averaged over five seeds. The residual uses a separate color scale. Layer-diagonal blocks are retained exactly.

References

- Kyra Ahrens, Hans Hergen Lehmann, Jae Hee Lee, and Stefan Wermter. Read between the layers: Leveraging multi-layer representations for rehearsal-free continual learning with pre-trained models. *TMLR*, 2024.
- Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, pp. 11254–11263, 2019a.
- Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *NeurIPS*, 32, 2019b.
- Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, pp. 8218–8227, 2021.
- Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *NeurIPS*, 33:15920–15930, 2020.
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *PAMI*, 44(7):3366–3385, 2021.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. doi: 10.1137/1031049. URL <https://doi.org/10.1137/1031049>.
- Dan Hendrycks, Steven Basart, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 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. *arXiv preprint arXiv:2106.09685*, 2021.
- Zhiqi Kang, Liyuan Wang, Xingxing Zhang, and Karteek Alahari. Advancing prompt-based methods for replay-independent general continual learning. In *ICLR*, 2025.
- Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. *NeurIPS*, 35:5065–5079, 2022.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.
- Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *arXiv preprint arXiv:2110.10031*, 2021.
- Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

- Minh Le, An Nguyen, Huy Nguyen, Trang Nguyen, Trang Pham, Linh Van Ngo, and Nhat Ho. Mixture of experts meets prompt-based continual learning. *Advances in Neural Information Processing Systems*, 37:119025–119062, 2024.
- Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- Zhizhong Li and Derek Hoiem. Learning without forgetting. *PAMI*, 2018.
- Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *NeurIPS*, 36, 2024.
- Fei Mi, Lingjing Kong, Tao Lin, Kaicheng Yu, and Boi Faltings. Generalized class incremental learning. In *CVPR Workshops*, pp. 240–241, 2020.
- Jun-Yeong Moon, Keon-Hee Park, Jung Uk Kim, and Gyeong-Moon Park. Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning. In *ICCV*, pp. 11731–11741, 2023.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.
- Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, pp. 524–540. Springer, 2020.
- Xuan Rao, Mingming Ha, Bo Zhao, Derong Liu, and Cesare Alippi. Scalable analytic classifiers with associative drift compensation for class-incremental learning of vision transformers, 2026a.
- Xuan Rao, Simian Xu, Zheng Li, Bo Zhao, Derong Liu, Mingming Ha, and Cesare Alippi. Compensating distribution drifts in continual learning with pre-trained vision transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(30):25090–25098, 2026b. doi: 10.1609/aaai.v40i30.39698.
- Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *NeurIPS*, 30, 2017a.
- Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017b.
- Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.
- A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv 1606.04671*, 2016. URL <https://arxiv.org/abs/1606.04671>.
- 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 *CVPR*, pp. 11909–11919, 2023.
- Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

- Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.
- Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. Afec: Active forgetting of negative transfer in continual learning. *NeurIPS*, 2021.
- Liyuan Wang, Xingxing Zhang, Qian Li, Jun Zhu, and Yi Zhong. Coscl: Cooperation of small continual learners is stronger than a big one. In *ECCV*, 2022a.
- Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *NeurIPS*, 2023.
- Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *PAMI*, 2024.
- Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam’s razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35:5682–5695, 2022b.
- Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pp. 631–648. Springer, 2022c.
- Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pp. 139–149, 2022d.
- Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In *ICLR*, 2025.
- Hongwei Yan, Liyuan Wang, Kaisheng Ma, and Yi Zhong. Orchestrate latent expertise: Advancing online continual learning with multi-level supervision and reverse self-distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23670–23680, 2024. doi: 10.1109/CVPR52733.2024.02234.
- Hongwei Yan, Guanglong Sun, Kanglei Zhou, Qian Li, Liyuan Wang, and Yi Zhong. FlyPrompt: Brain-inspired random-expanded routing with temporal-ensemble experts for general continual learning. In *International Conference on Learning Representations*, 2026.
- Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. *arXiv preprint arXiv:2303.05118*, 2023.
- Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. *NeurIPS*, 35:11602–11614, 2022.
- Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. Gacl: Exemplar-free generalized analytic continual learning. *NeurIPS*, 37:83024–83047, 2024a.
- Huiping Zhuang, Yuchen Liu, Run He, Kai Tong, Ziqian Zeng, Cen Chen, Yi Wang, and Lap-Pui Chau. F-OAL: Forward-only online analytic learning with fast training and low memory footprint in class incremental learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 41517–41538, 2024b. doi: 10.52202/079017-1314.

A Implementation Details

A.1 Experiment Setup

Backbone initialization. All experiments use ViT-B/16, with 12 transformer blocks, 16×16 patches, and a feature dimension of 768. We initialize the encoder from supervised pretraining on ImageNet-21K (Sup-21K) (Dosovitskiy et al., 2020; Ridnik et al., 2021) and keep its pretrained weights frozen throughout the stream.

Data stream and environment. CIFAR-100 uses its 50,000 training and 10,000 test images. ImageNet-R uses 24,000 training and 6,000 test images from the designated split. We follow the Si-Blurry configuration specified in the main text, with seeds 1, 2, 3, 4, 5 controlling stream construction and model randomness. Compared shared-PET variants use the same seed-specific streams. Session indices support stream construction, evaluation, and session-wise updates in baselines that require them; LOOM maintains one shared PET module throughout. Training images are resized to 224×224 , randomly cropped at the same resolution with four-pixel padding, and horizontally flipped. CIFAR-100 additionally uses the CIFAR-10 AutoAugment policy, and ImageNet-R uses the ImageNet policy; CUB-200 receives no additional AutoAugment operation. RGB channels are normalized using mean (0.5071, 0.4867, 0.4408) and standard deviation (0.2675, 0.2565, 0.2761) for CIFAR-100 and CUB-200, and (0.485, 0.456, 0.406) and (0.229, 0.224, 0.225) for ImageNet-R. Evaluation and analytic feature collection use deterministic resizing and normalization. LOOM uses Adam with a constant learning rate of 5×10^{-3} , zero weight decay, and batch size 64. Each current batch supports three optimization steps before being discarded; it contributes to the analytic statistics only once. The auxiliary head is trained with cross-entropy restricted to classes present in the current batch. All GCL experiments use one NVIDIA RTX 5090 per job on a Linux server with Intel Xeon Gold 6530 processors.

Shared adaptation modules. The three LOOM variants adapt the first five transformer blocks. LOOM-Prompt inserts 20 learned prompt tokens at each of these blocks, initialized uniformly at random. LOOM-Adapter uses a bottleneck of width 10 with a ReLU nonlinearity and a learned residual scale; its down-projection uses Kaiming initialization and its up-projection is initialized to zero. LOOM-LoRA applies rank-five updates to the key and value projections, with scaling $\alpha/r = 1/5$; one factor is randomly initialized and the other is zero. These parameters and the auxiliary classification head persist across sessions. The PET modules are initialized directly rather than loaded from pretrained prompt experts.

Analytic statistics and matched comparisons. After the gradient updates on a batch, we extract its features once in evaluation mode without gradients. We retain the unnormalized CLS outputs of the last four transformer blocks, before the final LayerNorm, yielding four 768-dimensional vectors. The auxiliary gradient-trained head uses the final normalized representation, whereas the reported LOOM predictions use the analytic head. The retained statistics and structured classifier follow the construction in Section 3.3. We use the absolute regularization coefficient $\lambda = 10^3$ in $\tilde{\mathbf{G}}_t + \lambda \mathbf{I}_D$, consistent with the summed objective in Eq. (2), and refresh the classifier before evaluation. No historical images or per-example feature buffers are retained by LOOM. Matched comparisons vary the classifier construction and retained statistics while keeping adaptation fixed: final-layer ridge uses one block; independent multilayer ridge fits four separate classifiers and sums their logits; full joint ridge retains all blocks of the concatenated Gram matrix. These comparisons share the PET architecture, update schedule, feature extraction, and ridge coefficient, allowing the effect of cross-layer integration to be assessed separately from adaptation capacity.

A.2 Baselines

Fine-tuning and regularization. Sequential fine-tuning updates the encoder and classifier as the stream arrives; linear probing freezes the encoder and updates only the classifier. The slow-learning variant reduces the encoder learning rate to 5×10^{-5} , limiting changes to the pretrained representation (Zhang et al., 2023). EWC penalizes changes to parameters with high estimated importance (Kirkpatrick et al., 2017), and LwF distills earlier predictions from a frozen teacher (Li & Hoiem, 2018). These methods provide comparisons based on backbone adaptation and explicit retention objectives, without prompt pools or analytic classifier updates.

Prompt-based and GCL methods. L2P uses a pool of 30 length-five prompts and retrieves five per input. DualPrompt uses length-five shared prompts in the first two blocks and length-20 expert prompts in the next three blocks, with ten expert slots; MVP uses the same lengths with five expert slots. CODA-Prompt uses ten length-20 components at each of the first five blocks. MISA loads pretrained shared and expert prompts, with shared length ten and expert length 20, and applies non-parametric logit masking (Kang et al., 2025). FlyPrompt assigns length-20 prompts to the first five blocks of each of its five experts. These methods follow the single-pass update schedule, with batch size 64, three updates per batch, and batch-seen logit masking. FlyPrompt uses a fixed random expansion of dimension 10^4 for its analytic expert router, router regularization of 10^4 , and two temporal-ensemble heads per expert with decay rates 0.9 and 0.99 (Yan et al., 2026).

Methods originally developed for offline CL. S-Prompt++, HiDe-Prompt, NoRGa, and SD-LoRA are compared under the single-pass GCL adaptations reported by Yan et al. (2026). S-Prompt++ uses five experts with length-20 prompts in the first five blocks. HiDe-Prompt and NoRGa use five prompt experts, each with length-five key and value prefixes in the first five blocks; the gating prediction selects one expert, whose class scores determine the final prediction. Feature statistics used for alignment are accumulated from the stream.

Analytic baselines. RanPAC forms ReLU features from a fixed random projection of dimension 10^4 and fits a ridge classifier (McDonnell et al., 2024). The implemented variant adapts adapters during the first session, collects features after current-batch updates, and freezes adaptation thereafter; it uses $\lambda = 10^5$. It temporarily buffers feature vectors and labels until classifier updates, so its storage includes per-example features in addition to aggregate statistics. LayUP uses a frozen encoder and the raw CLS outputs of the final six blocks, with a full joint Gram matrix and class feature sums (Ahrens et al., 2024). Its online configuration omits first-session adaptation, accumulates each incoming example once, and uses fixed $\lambda = 1$ with a linear-system solve. A frozen final-layer ridge classifier provides a corresponding single-layer control. These frozen controls complement the matched sequential-PET comparisons by separating the contribution of adaptation from that of multilayer classification.

A.3 Evaluation Metrics

At each evaluation, top-1 accuracy is measured on test images from the classes observed so far, with predictions restricted to the same label set. For E periodic evaluations and J session-end evaluations, denote these accuracies by a_e and a_j^{end} , respectively. We compute

$$A_{\text{auc}} = \frac{1}{E} \sum_{e=1}^E a_e, \quad A_{\text{last}} = a_J^{\text{end}}. \quad (13)$$

Here, A_{auc} is the discrete average of periodic anytime accuracies (Koh et al., 2021); the sampling interval is approximately 1,000 observed training examples. Session-end evaluations are recorded separately. A_{last} measures final test accuracy over all observed classes. In particular, each session-end accuracy evaluates the cumulative observed label space, not only the examples associated with that session.

For forgetting, let $r_{j,c}$ be the test accuracy of class c at the end of session j . Following the classwise convention used by the evaluation framework, define $\mathcal{H} = \{c : \max_{1 \leq j < J} r_{j,c} > 0\}$. The reported forgetting is

$$F = \frac{1}{|\mathcal{H}|} \sum_{c \in \mathcal{H}} \left(\max_{1 \leq j < J} r_{j,c} - r_{J,c} \right). \quad (14)$$

This compares final class accuracy with its best recorded accuracy before the last session, excluding classes with no positive earlier accuracy. The final session is excluded from the maximum, so a class that improves beyond its earlier best can contribute negatively. All reported metrics are expressed in percent and aggregated across seeds. Missing forgetting values are left unreported.

A.4 LOOM Algorithm

Algorithm 1 LOOM: online adaptation and cross-layer moment integration

- 1: **Inputs:** stream $\{\mathcal{B}_t\}_{t=1}^T$, frozen backbone f_θ , selected layers $\{\ell_a\}_{a=1}^K$, regularization $\lambda > 0$, and gradient-update budget per batch.
 - 2: **Initialize:** shared PET parameters $\mathbf{p}_0$ and an auxiliary linear head; $\mathcal{C}_0 \leftarrow \emptyset$, $\mathbf{G}_0^{(a,a)} \leftarrow \mathbf{0}$ for all a , and $\mathbf{Q}_0 \in \mathbb{R}^{D \times 0}$ with empty class counts.
 - 3: **for** $t = 1$ to T **do**
 - 4: Extend $\mathcal{C}_{t-1}$ with newly observed labels to obtain $\mathcal{C}_t$, preserving class order; set $N_{t-1,c} = 0$ for each new class c .
 - 5: Starting from $\mathbf{p}_{t-1}$ and the retained auxiliary head, update both on $\mathcal{B}_t$ within the prescribed budget, keeping θ frozen; denote the updated PET by $\mathbf{p}_t$. Use cross-entropy restricted to classes present in $\mathcal{B}_t$.
 - 6: Collect $\mathbf{Z}_t = [\mathbf{z}_{t,1}^\top; \dots; \mathbf{z}_{t,n_t}^\top]$ once using $\mathbf{p}_t$ and deterministic preprocessing; let $\mathbf{Z}_t^{(a)}$ be its layer- ℓ_a column block.
 - 7: **for** $a = 1$ to K **do**
 - 8: $\mathbf{G}_t^{(a,a)} \leftarrow \mathbf{G}_{t-1}^{(a,a)} + (\mathbf{Z}_t^{(a)})^\top \mathbf{Z}_t^{(a)}$.
 - 9: **end for**
 - 10: Form one-hot labels $\mathbf{Y}_t^{(t)}$ over $\mathcal{C}_t$ and update $\mathbf{Q}_t \leftarrow \text{pad}_{\mathcal{C}_t}(\mathbf{Q}_{t-1}) + \mathbf{Z}_t^\top \mathbf{Y}_t^{(t)}$.
 - 11: $N_{t,c} \leftarrow N_{t-1,c} + |\{i : y_{t,i} = c\}|$ for each $c \in \mathcal{C}_t$.
 - 12: Discard the current training images and per-example features.
 - 13: **if** evaluation is requested after batch t **then**
 - 14: $\mathbf{N}_t \leftarrow \text{diag}(N_{t,c} : c \in \mathcal{C}_t)$; $\mathbf{U}_t \leftarrow \mathbf{Q}_t \mathbf{N}_t^{-1/2}$.
 - 15: **for** $a = 1$ to K **do**
 - 16: $\mathbf{S}_t^{(a,a)} \leftarrow \mathbf{G}_t^{(a,a)} - \mathbf{U}_t^{(a)} \mathbf{U}_t^{(a)\top}$.
 - 17: **end for**
 - 18: $\mathbf{A}_t \leftarrow \text{blockdiag}(\mathbf{S}_t^{(1,1)} + \lambda \mathbf{I}_{d_1}, \dots, \mathbf{S}_t^{(K,K)} + \lambda \mathbf{I}_{d_K})$.
 - 19: Compute $\widetilde{\mathbf{W}}_t$ by Eq. (12), using the layerwise blocks of $\mathbf{A}_t$ and the $C_t \times C_t$ system.
 - 20: For each test input $\mathbf{x}$, extract $\mathbf{z}_t(\mathbf{x})$ using $\mathbf{p}_t$. Predict
 - 21: $\hat{y}_t(\mathbf{x}) = \arg \max_{c \in \mathcal{C}_t} [\widetilde{\mathbf{W}}_t^\top \mathbf{z}_t(\mathbf{x})]_c$.
 - 22: **end if**
 - 23: **end for**
-

B Additional Results

B.1 Cross-layer Representation Similarity

Figure 6 reports linear centered kernel alignment (CKA) between the raw CLS representations of all 12 transformer blocks after sequential adaptation. Adjacent blocks exhibit the strongest similarity, while the correspondence decreases as their depth separation grows. The selected late-layer representations therefore remain closely related but are not redundant. This structure supports integrating several late layers while retaining layer-specific second-order statistics, rather than treating their features as interchangeable.

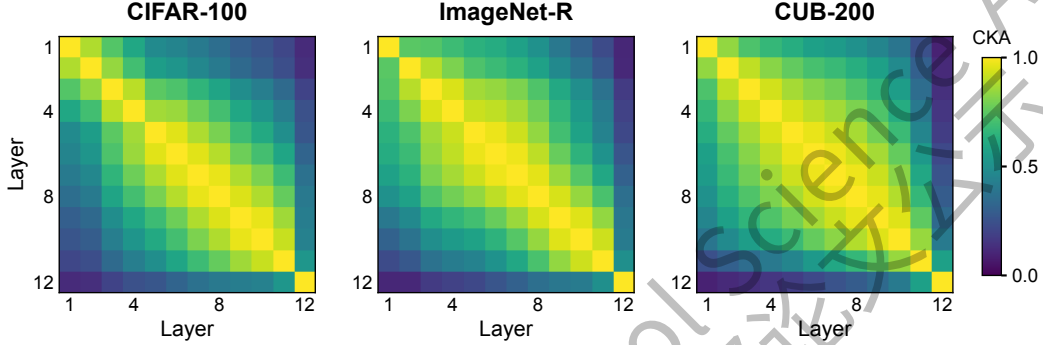


Figure 6: Linear CKA between the raw CLS representations of all 12 blocks in the final LOOM-LoRA encoder, initialized from Sup-21K. The CKA matrices are averaged over five seeds.

B.2 Resource Costs

Table 4: Recorded resource costs with Sup-21K and sequential LoRA, averaged over five seeds. Parameter counts include the encoder, LoRA, and auxiliary head; statistical storage excludes classifier weights. Peak denotes peak allocated GPU memory. Solve time is cumulative, whereas wall time includes data loading and evaluation.

Classifier	Parameters (M)		Statistics	Peak	Solve	Wall
	Total	Trainable	(MiB)	(GiB)	(s)	(min)
<i>Dataset: CIFAR-100</i>						
Final-layer ridge	85.95	0.154	5.09	5.25	0.39	23.30
Independent ridge	85.95	0.154	20.34	5.27	0.62	23.55
Full joint ridge	85.95	0.154	74.34	5.32	1.26	23.45
LOOM	85.95	0.154	20.34	5.27	0.80	23.49
<i>Dataset: ImageNet-R</i>						
Final-layer ridge	86.03	0.231	5.67	5.26	0.19	8.86
Independent ridge	86.03	0.231	22.69	5.28	0.35	9.40
Full joint ridge	86.03	0.231	76.69	5.33	0.75	8.99
LOOM	86.03	0.231	22.69	5.28	0.37	8.97
<i>Dataset: CUB-200</i>						
Final-layer ridge	86.03	0.231	5.67	5.26	0.14	2.70
Independent ridge	86.03	0.231	22.69	5.28	0.22	2.71
Full joint ridge	86.03	0.231	76.69	5.33	0.41	2.71
LOOM	86.03	0.231	22.69	5.28	0.25	2.60

致谢

在本论文完成之际，我的心中充满兴奋与感激。

本项的选题源于我对于持续学习领域所面临的问题的好奇与思考。持续学习长久以来一直面对着灾难性遗忘的问题，模型需要保证稳定性和可塑性，既要学习新的类别的知识，也不能忘记旧类别。现有提示微调方法多聚焦于“选择”层面（路由与集成），却忽视了记忆保持的根本机制。本研究受神经科学中生物脑系统的神经网络层次化记忆机制启发，重新思考如何更高效地利用预训练模型的多层信息，优化了以表征为基础的岭回归式解析分类算法，在通用持续学习场景的三个基准上验证了方法的有效性。

在此由衷的感谢一直带我的钟毅老师。导师在整个项目中无偿给予我明晰的指导与有力支持。本研究的实验设计、数据获取、分析计算、实验实施以及论文撰写等工作均由我在指导老师的指导下独立完成。在导师的指导下，我独立完成了本项目的全部工作。我主要负责开展复现实验并记录实验数据；在导师确定的总实验方向下分步推进实验，尝试多种模型修改方案，调试代码，记录并对比各组实验结果；论文撰写阶段，我在导师搭建好的框架下完成内容的填写与修改。

这次项目中，我在基线评估、模型迭代等环节都曾遇到困难，其中一部分已经克服，还有一部分可在未来继续深入研究、尝试解释。在最初修改模型的时候又出现了信息泄露问题。早期方法在测试时使用了真实任务标签判断样本新旧，导致结果虚高。通过指导老师的指导，排查评估逻辑，删除所有对测试标签的依赖，使方法回到任务无关设定，并重新复现了干净基线。并重新思考改进模型的策略，进行后续实验。在实验进行到后续的时候，经过近 10 种尝试，改进方向几乎都失败了。在修复基线后，尝试的时间集成、路由、提示长度等多个方向均无显著提升，一度陷入瓶颈。我发现一个反常现象——两阶段路由将路由正确率从 0.556 提升到 0.9 以上、但准确率几乎不变——反证出瓶颈不在“选择”而在“记忆”，从而转向可塑性调节与原型记忆方法，实现了突破。

这段科研经历对我意义重大。我经历了一次完整的科研过程，将自己的精力倾注于人工智能持续学习领域难题背后的原理探究。感谢所有支持我的老师和同学。我相信这将是未来学习与科研之路的起点，我会继续秉持热情与严谨，去探索更大的世界。

在实验过程中我使用了 deepseek v4 pro 模型，主要用于统计实验数据，以及修改部分论文段落提升可读性，使用频率大约是一周两次左右。所有 AI 辅助的修改都已经过作者的审查，我们对文章的全部内容的真实性和原创性负责。