SP-CLIP: STABLE-PLASTIC LORA CONSOLIDATION AND CONTEXTUAL BANDIT PROMPT ROUTING FOR CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning requires a model to adapt to new classes while retaining previously acquired knowledge. Although LoRA-based CLIP methods have shown promising performance, they remain limited in text prototype adaptation and cross-task parameter management. Fixed text templates cannot adequately accommodate instance variations and evolving visual representations, while sharing LoRA parameters across tasks may cause parameter interference and catastrophic forgetting. To address these limitations, we propose SP-CLIP, a two-stage framework for replay-free class-incremental learning that combines Stable-Plastic LoRA consolidation with LinUCB-based prompt routing. At task boundaries, visual and textual LoRA updates are consolidated into the base weights, and the LoRA modules are reinitialized to accumulate task knowledge while reducing cross-task interference. The consolidated CLIP model is then frozen, and LinUCB dynamically routes learnable prompts to construct instance-adaptive text prototypes. This design coordinates stability and plasticity through task-level parameter management and instance-level semantic adaptation.Experiments show that the two components provide complementary benefits, while SP-CLIP achieves consistent improvements over MG-CLIP on most benchmark datasets.

1 INTRODUCTION

With the development of large-scale vision–language pre-trained models, CLIP has become an important foundation model for visual recognition and continual learning due to its strong image–text alignment and zero-shot generalization capabilities (Radford et al., 2021). Unlike conventional vision models, CLIP can construct classifiers from class names or natural-language descriptions, reducing its reliance on task-specific supervision. Meanwhile, parameter-efficient fine-tuning methods such as LoRA adapt pre-trained models by optimizing only a small number of low-rank parameters while keeping most pre-trained weights frozen (Hu et al., 2021). This provides an effective way to apply vision–language models to continual learning. Representative methods such as MG-CLIP introduce LoRA into CLIP and demonstrate the potential of LoRA-based adaptation for replay-free class-incremental learning (Huang et al., 2025).

However, existing LoRA-based CLIP continual learning methods still face limitations in task-level parameter management and sample-level text-prototype adaptation. At the task level, existing methods typically reuse the same LoRA parameters across different tasks. As the task sequence progresses, updates from different tasks repeatedly act on the same low-rank parameter space. The optimization of a new task may therefore alter parameter directions formed by previous tasks, leading to inter-task parameter competition, visual and textual representation drift, and degraded performance on previously learned classes. Although maintaining an independent LoRA branch for each task can reduce direct interference, it causes the additional parameter and storage costs to grow linearly with the number of tasks, reducing parameter efficiency and deployment scalability.

At the sample level, fixed text templates usually produce relatively stable text prototypes for each class. However, the introduction of new classes, changes in background and visual styles, and intra-class appearance variations can cause visual representations to evolve throughout continual learning. A single text prototype generated by a fixed template may therefore be insufficient to match dynamic

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.

and diverse visual features. Conditional prompts and prompt pools provide more flexible text representations, but existing prompt-routing methods often rely on deterministic conditional mappings, feature similarity between the current image and candidate prompts, or locally greedy selection. Such methods make limited use of the actual classification benefits of different prompts on historical samples and rarely model the uncertainty of reward estimates explicitly. Consequently, prompt routing may become prematurely concentrated on a small subset of candidates, leaving other potentially effective prompts insufficiently explored. Moreover, approaches that rely on task identities and maintain task-specific prompts or parameter branches introduce additional overhead that grows with the number of tasks, making them less suitable for unified inference when the task identity is unknown.

To address these issues, we propose SP-CLIP, a two-stage framework for replay-free class-incremental learning. In the first stage, we introduce a Stable-Plastic LoRA consolidation mechanism for task-level parameter management. At each task boundary, the LoRA increments learned by the visual and text encoders are merged into the corresponding base weights, and the LoRA branches are reinitialized. This preserves the knowledge accumulated from previous tasks in the base model. The reinitialized LoRA branches then learn new parameter increments for the current task relative to the consolidated model, preventing updates from different tasks from continuously accumulating in the same branch, reducing inter-task parameter competition, and keeping the number of LoRA parameters independent of the number of tasks. In the second stage, we freeze the consolidated CLIP model and optimize only the Prompt Pool. For each input image, LinUCB (Li et al., 2010) combines the image context, historical classification feedback, and uncertainty in reward estimates to select a prompt, which is then combined with class names to construct sample-adaptive class text prototypes. The two stages separately address task-level parameter management and sample-level semantic adaptation. Freezing the consolidated CLIP prevents the backbone representation from continually changing during prompt-routing learning, reducing its interference with LinUCB context and reward estimation. This enables prompt routing to explore on a relatively stable vision–language representation while requiring no task identity during inference.

Our main contributions are summarized as follows:

- We propose a Stable-Plastic LoRA consolidation strategy for both the visual and text encoders. At each task boundary, the low-rank updates are merged into the base weights and the LoRA branches are reinitialized, reducing the interference caused by continuously reusing the same LoRA branch across tasks.
- We propose a LinUCB-based dynamic prompt-routing mechanism that exploits image context, historical classification feedback, and uncertainty in reward estimates to perform sample-adaptive class text-prototype selection.
- We develop a replay-free class-incremental learning framework that combines task-level LoRA management with dynamic prompt routing, and achieves better performance than MG-CLIP on multiple public benchmark datasets.

Our code and pre-trained models will be made publicly available upon acceptance.

2 Related Work

2.1 Class-Incremental Continual Learning and Vision–Language Models

Class-incremental continual learning requires a model to learn new classes sequentially while retaining its recognition ability on previously observed classes (De Lange et al., 2021). In the replay-free setting, historical images are unavailable for rehearsal, making the model particularly susceptible to catastrophic forgetting. Existing approaches mainly include experience replay, parameter or feature regularization, parameter isolation, and classifier expansion (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Rebuffi et al., 2017). Replay-based methods preserve previous knowledge by storing historical samples but introduce additional memory costs. Regularization-based methods constrain changes in model parameters or features, whereas parameter-isolation methods allocate independent modules to different tasks to reduce interference, but typically increase the model size as the number of tasks grows.

Recently, pre-trained vision–language models such as CLIP have been introduced into continual learning to exploit general visual–semantic representations learned from large-scale image–text pre-training (Wu et al., 2022; Lee et al., 2023; Zhou et al., 2025a; Wang et al., 2023; Liu et al., 2023; Ni et al., 2023; D'Alessandro et al., 2023). Compared with training from scratch or full fine-tuning, continual learning with pre-trained models typically updates only a small number of parameters or auxiliary modules, thereby reducing adaptation costs while preserving pre-trained knowledge. Recent studies have further explored efficient CLIP adaptation, mixture-of-experts adapters, and probabilistic fine-tuning for continual learning (Thengane et al., 2022; Yu et al., 2024; Jha et al., 2024; Huang et al., 2025). Forgetting in vision–language models has also received increasing attention in continual learning (Zhou et al., 2025b). In the absence of historical data, how to manage model updates while balancing the retention of previous knowledge and adaptation to new classes remains an important challenge (Zhou et al., 2024).

2.2 Parameter-Efficient Fine-Tuning for Continual Learning

LoRA introduces low-rank updates alongside the pre-trained weights, allowing downstream adaptation with only a small number of trainable parameters (Hu et al., 2021). Because it substantially reduces training and storage costs, LoRA has been widely applied to large pre-trained models and vision–language models. Recent continual learning methods have also adopted LoRA or other parameter-efficient modules to reduce the overhead of sequential task adaptation (Yu et al., 2024; Jha et al., 2024; Huang et al., 2025).

From the perspective of cross-task parameter management, existing parameter-efficient continual learning methods can be broadly categorized into task isolation and parameter sharing. Task-isolation methods maintain independent adaptation branches for different tasks, which can reduce direct interference but cause the additional parameter and storage costs to grow with the number of tasks. Parameter-sharing methods reuse the same low-rank parameters across multiple tasks to control model size. However, updates from different tasks may continuously accumulate and interfere within the same low-rank subspace. For example, MG-CLIP (Huang et al., 2025) shares LoRA parameters across tasks to limit model growth, but sequential task updates may act on the same low-rank subspace and lead to inter-task parameter competition.

Another line of work uses model merging to consolidate knowledge from different tasks (Ilharco et al., 2022; Yadav et al., 2023). MAGMAX (Marczak et al., 2024) leverages model merging for continual learning while controlling model growth. In contrast, our method focuses on merging task-level LoRA increments and reinitializing the LoRA branch: at each task boundary, the LoRA updates are merged into the base weights and the LoRA branch is reinitialized. This preserves historical knowledge while providing a new low-rank adaptation space for subsequent tasks.

2.3 Prompt Learning and Contextual Decision-Making

Prompt learning optimizes contextual representations in the text input, enabling CLIP to construct more flexible text prototypes for downstream tasks (Jia et al., 2022; Khattak et al., 2023). CoOp replaces manually designed template contexts with learnable context tokens (Zhou et al., 2022b), while CoCoOp further generates instance-conditioned prompts from input images (Zhou et al., 2022a). In continual learning, prompt-pool methods such as L2P, DualPrompt, and CODA-Prompt maintain multiple candidate prompts to adapt to different tasks or input samples (Wang et al., 2022b;a; Smith et al., 2023; Nicolas et al., 2023). These methods typically select prompts using feature similarity, attention-based matching, or other deterministic routing mechanisms.

Contextual multi-armed bandits provide a general framework for dynamic action selection under contextual information (Li et al., 2010). LinUCB combines a linear reward estimator with an upper confidence bound, jointly considering the expected reward of an action and the uncertainty of its estimate. It therefore balances the exploitation of actions with strong historical performance and the exploration of potentially effective alternatives. Following this perspective, dynamic prompt selection can be formulated as a contextual multi-armed bandit problem, which motivates our use of LinUCB for sample-wise prompt routing.

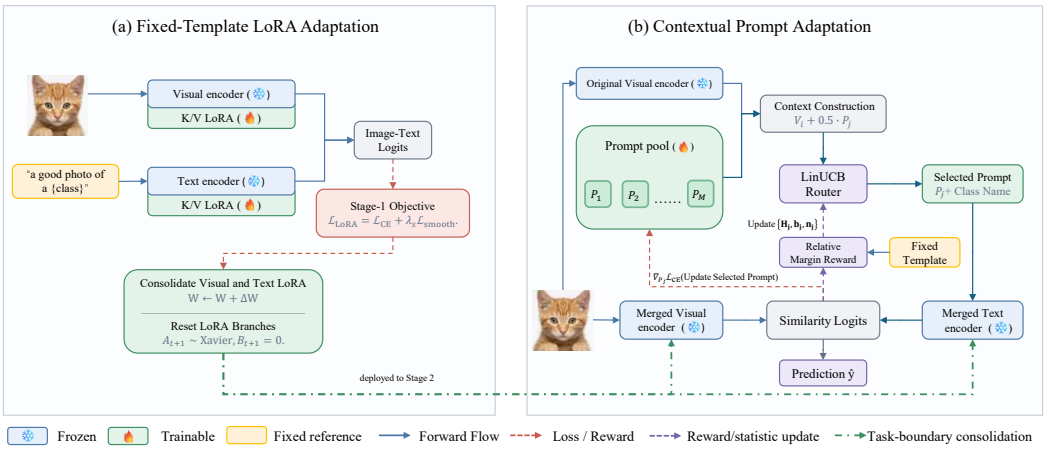


Figure 1: Overview of SP-CLIP. For each incremental task, Stage I learns visual and textual LoRA updates using a fixed template, merges the resulting updates into the backbone, and reinitializes the low-rank branches. Stage II freezes the consolidated CLIP model and dynamically selects an instance-specific prompt from the prompt pool using LinUCB.

3 Methodology

3.1 Problem Formulation

We study replay-free class-incremental continual learning without using task identities during inference. The model receives a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots$. For task t, the training set is $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$, where $y_i \in \mathcal{C}_t$ and $\mathcal{C}_t$ denotes the class set introduced by task t.

The class sets of different tasks are disjoint, i.e., $\mathcal{C}_t \cap \mathcal{C}_k = \emptyset$ for $t \neq k$. During task t, the model can access only the current-task data and cannot replay or revisit historical images or features.

After completing task t, the model is evaluated over all observed classes, $\mathcal{C}^{(t)} = \bigcup_{k=1}^{t} \mathcal{C}_k$. Task identities are not used for classification or prompt routing and are available only at training-time task boundaries for LoRA consolidation, branch reinitialization, and LinUCB state adjustment.

3.2 Overall Architecture

We adopt a sequential two-stage training process. Stage I learns and consolidates task-level visual–language adaptations with LoRA, while Stage II freezes the consolidated CLIP model and dynamically routes prompts with LinUCB for sample-level semantic adaptation.

3.2.1 Stage I: Stable-Plastic LoRA Consolidation

During Stage I of task t, the Prompt Pool is frozen and each class is represented with the fixed template `"a good photo of a {class}."`. The model is optimized using the standard classification cross-entropy loss over the current task classes $\mathcal{C}_t$.

To constrain excessive changes in projection outputs caused by LoRA, consider a projection layer with input h, base output Wh, and adapted output $o^{\text{eff}} = Wh + sB_tA_th$. We introduce the output-space smoothness regularizer

$$\mathcal{L}_{\text{smooth}} = \text{SmoothL1}\left(\frac{o^{\text{eff}}}{\text{Std}(Wh) + \varepsilon}, \frac{Wh}{\text{Std}(Wh) + \varepsilon}\right), \tag{1}$$

where $\text{Std}(Wh)$ denotes the empirical standard deviation and ε is a numerical stability constant. The Stage-I objective is $\mathcal{L}_{\text{LoRA}} = \mathcal{L}_{\text{CE}} + \lambda_s \mathcal{L}_{\text{smooth}}$.

At each task boundary, the accumulated LoRA update $\Delta W_t = sB_tA_t$ is merged into the base weight as $W_{\text{base}}^{(t)} = W_{\text{base}}^{(t-1)} + \Delta W_t$. The merge is performed in FP32 precision. For the next task, the low-rank branch is reinitialized, with A_{t+1} initialized using Xavier initialization and $B_{t+1} = 0$.

Since $B_{t+1} = 0$, the LoRA residual is zero at the beginning of task $t+1$, and the effective weight initially equals the consolidated base weight. This procedure preserves accumulated knowledge while providing a fresh trainable low-rank space for subsequent tasks and reducing direct competition between task-specific LoRA updates.

3.2.2 Stage II: Dynamic Prompt Routing via LinUCB

In Stage II, the consolidated CLIP model is frozen and only the Prompt Pool is optimized. For the j-th prompt $P_j = \{P_{j,1}, \ldots, P_{j,L_p}\}$, its routing representation is $e_j = \text{Norm}(\sum_{k=1}^{L_p} P_{j,k})$, where $\text{Norm}(\cdot)$ denotes ℓ_2-normalization.

For an input image x_i, the permanently frozen original image encoder produces $v_i = \text{Norm}(f_v^{\text{orig}}(x_i))$. Since the image feature and summed prompt representation are dimension-compatible, we fuse them directly:

$$c_{i,j} = \text{Norm}\left(v_i + \gamma e_j\right), \tag{2}$$

where $\gamma > 0$ controls the relative contribution of the prompt representation. This projection-free design preserves the original feature geometry and avoids introducing additional trainable parameters into the routing branch.

The selected prompt is combined with the class names of the current task and encoded by the frozen CLIP text encoder for classification. Only the selected prompt receives gradients from the classification loss.

Each candidate prompt is treated as a separate LinUCB arm. For arm j, the algorithm maintains a context second-moment matrix H_j, a feedback vector b_j, and a selection count n_j. The reward parameter is estimated as $\hat{\theta}_j = (H_j + \lambda I)^{-1} b_j$, where $\lambda > 0$ is the ridge regularization coefficient. Let $S = \sum_{j=1}^{M} n_j$ denote the cumulative number of selections. The selection score is

$$\text{UCB}_{i,j} = c_{i,j}^{\top} \hat{\theta}_j + \alpha \sqrt{c_{i,j}^{\top} (H_j + \lambda I)^{-1} c_{i,j}} + \beta \sqrt{\frac{\log(S+1)}{n_j + 1}}. \tag{3}$$

The three terms represent the estimated reward, reward-estimation uncertainty, and frequency-based exploration, respectively.

An optional warm-up phase can be employed, during which candidate prompts are selected in a round-robin manner for a fixed number of steps before switching to LinUCB selection. When warm-up is not used, prompt selection starts directly with LinUCB.

The fixed template is used only as a reward reference and is not treated as an additional LinUCB arm. During training, classification probabilities are normalized over the current task class set $\mathcal{C}_t$. Let $p_t^q(c \mid x_i)$ denote the probability obtained under condition $q \in \{\text{prompt}, \text{fix}\}$. The corresponding margin is

$$m_i^q = p_t^q(y_i \mid x_i) - \max_{c \in \mathcal{C}_t \setminus \{y_i\}} p_t^q(c \mid x_i). \tag{4}$$

Both margins are computed using the CLIP model with the task-level LoRA updates already merged into the base weights. The relative reward is

$$r_i = \tanh\left(\frac{m_i^{\text{prompt}} - m_i^{\text{fix}}}{T_r}\right), \tag{5}$$

where $T_r > 0$ controls the sensitivity of the reward to margin differences.

For the selected prompt j_i, the LinUCB statistics are updated online via $H_{j_i} \leftarrow H_{j_i} + c_{i,j_i} c_{i,j_i}^{\top}$, $b_{j_i} \leftarrow b_{j_i} + r_i c_{i,j_i}$, and $n_{j_i} \leftarrow n_{j_i} + 1$, independently of backpropagation. At task boundaries, the states are discounted via $b_j \leftarrow \rho b_j$, $H_j \leftarrow \rho H_j + (1-\rho) I$, and $n_j \leftarrow 0$, where $\rho \in (0, 1)$ is the decay coefficient.

During evaluation, all model parameters and LinUCB states are frozen. For each test image, the model selects a prompt using the complete LinUCB score and performs classification over all observed classes in $\mathcal{C}^{(t)}$, without using task identities, computing rewards, or updating parameters.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate SP-CLIP on four class-incremental learning benchmarks: CIFAR-100 (Krizhevsky et al., 2009), ImageNet-100 (Deng et al., 2009), ImageNet-A (Hendrycks et al., 2021b), and ImageNet-R (Hendrycks et al., 2021a). CIFAR-100 and ImageNet-100 are divided into 10 sequential tasks with 10 classes per task, whereas ImageNet-A and ImageNet-R are divided into 10 tasks with 20 classes per task. Following the exemplar-free setting, the model only accesses samples from the current task and does not store or revisit historical data. Evaluation is performed in a single-head setting over all classes observed so far, without using task identifiers. The data splits for ImageNet-A and ImageNet-R follow the official MG-CLIP protocol.

Competing Methods. We compare SP-CLIP with representative continual learning methods, including L2P++ (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODA-Prompt (Smith et al., 2023), Continual-CLIP (Thengane et al., 2022), Aper-Adapter (Zhou et al., 2025a), MOE4CL (Yu et al., 2024), CLAP (Jha et al., 2024), MagMax (Marczak et al., 2024), and MG-CLIP (Huang et al., 2025). All experiments use CLIP ViT-L/14 as the backbone. Baseline results on CIFAR-100 and ImageNet-R are directly taken from the supplementary material of the original MG-CLIP paper (Huang et al., 2025), whereas results on ImageNet-A and ImageNet-100 are reproduced by us.

Evaluation Metrics. We report the average incremental accuracy (Avg) and the final accuracy (Last). Avg is computed by averaging the classification accuracy over all observed classes after each task, while Last denotes the accuracy after the final task.

Implementation Details. We insert rank-8 LoRA modules into the Key/Value projections of the attention layers in both the visual and text encoders. Stage I uses the fixed template `"a good photo of a {class}."` to train the LoRA parameters. At each task boundary, the LoRA updates are merged into the base weights and the low-rank branches are reinitialized. Stage II freezes the consolidated CLIP model and optimizes a Prompt Pool with 10 candidate prompts, each containing 5 learnable context tokens.

For CIFAR-100, both stages are trained for 2 epochs, using learning rates of 10^{-3} and 10^{-4} for the LoRA parameters and Prompt Pool, respectively. Warm-up settings are configured separately for each experiment. For CIFAR-100, we use 400 round-robin warm-up selections per task before switching to LinUCB. LinUCB uses $\alpha = 1.0$, $\beta = 0.03$, and ridge coefficient $\lambda = 1.0$.

4.2 MAIN COMPARISON RESULTS

Table 1 reports the class-incremental learning performance of different methods on the four benchmarks. SP-CLIP achieves a higher Avg than MG-CLIP on CIFAR-100, ImageNet-R, ImageNet-100, and ImageNet-A, demonstrating consistent gains across all evaluated datasets.

The improvements are particularly pronounced on ImageNet-R and ImageNet-A, suggesting that Stable-Plastic LoRA consolidation and instance-level prompt routing improve adaptation under substantial style variation and distribution shift. On ImageNet-100, SP-CLIP achieves a higher Avg than MG-CLIP, although the improvement in final-task accuracy is less consistent. This indicates that the proposed framework primarily improves the average performance throughout the incremental process.

4.3 ABLATION STUDIES AND MECHANISTIC ANALYSIS

Effectiveness of the Core Components. Due to the computational cost of evaluating all module combinations across multiple large-scale datasets, we first conduct a controlled component-wise

Method	CIFAR-100		ImageNet-R		ImageNet-A		ImageNet-100	
	Avg	Last	Avg	Last	Avg	Last	Avg	Last
L2P++	85.68	77.86	90.49	86.73	52.03	42.94	90.76	92.20
DualPrompt	86.63	79.12	90.66	87.14	66.68	55.72	80.23	69.00
CODA-Prompt	85.82	78.67	89.11	84.56	62.85	50.87	76.63	61.71
Continual-CLIP	80.48	73.46	86.99	83.05	68.36	69.63	81.78	81.80
Aper-Adapter	80.21	71.95	89.17	85.40	77.87	77.16	80.42	89.73
MOE4CL	90.98	85.83	93.27	90.42	61.62	71.60	88.87	80.73
CLAP	74.41	71.57	91.10	87.55	75.02	65.34	87.74	82.35
MagMax	90.16	86.06	93.22	89.55	70.66	68.57	92.37	84.58
MG-CLIP	91.78	87.03	93.66	91.08	85.82	81.85	93.19	**93.76**
SP-CLIP	**92.27**	**87.49**	**97.22**	**94.85**	**88.31**	**81.88**	**95.41**	93.08

Table 1: Class-incremental continual learning results on four benchmarks. All methods use CLIP ViT-L/14. "Avg" denotes the average incremental accuracy, and "Last" denotes the final accuracy after the last task. Baseline results on CIFAR-100 and ImageNet-R are directly taken from the supplementary material of the original MG-CLIP paper (Huang et al., 2025), whereas results on ImageNet-A and ImageNet-100 are reproduced by us and averaged over three random seeds.

LoRA Management	Prompt Strategy	Avg (%)	Last (%)
Shared LoRA	Fixed Template	91.13	85.86
Shared LoRA	LinUCB	91.77	86.46
Merge + Reset	Fixed Template	91.63	86.95
Merge + Reset	LinUCB	**92.27**	**87.49**

Table 2: Component-wise ablation of task-level LoRA management and instance-level prompt routing on CIFAR-100.

ablation on CIFAR-100. Table 2 compares two task-level LoRA management strategies, Shared LoRA and Merge + Reset, in combination with either a fixed text template or LinUCB-based prompt routing.

Shared LoRA continuously reuses the same LoRA branches across tasks and does not reset them at task boundaries. In contrast, Merge + Reset merges the current task's LoRA updates into the base weights and then reinitializes the low-rank branches.

With the fixed template, Merge + Reset consistently outperforms Shared LoRA in terms of both Avg and Last, indicating that consolidating task-specific LoRA updates and reinitializing the low-rank branches improve parameter adaptation across sequential tasks. Moreover, LinUCB routing further improves both LoRA management strategies, demonstrating that instance-level prompt adaptation is effective under both tested parameter management schemes. The combination of Merge + Reset and LinUCB achieves the best overall performance. These results show that task-level parameter management and instance-level text-prototype adaptation improve continual learning performance from different perspectives and can be effectively combined.

Cross-Dataset Effectiveness of the Prompt Stage. We further evaluate whether the LinUCB prompt stage provides consistent benefits across different data distributions. Table 3 compares the Stage-I model, which uses Merge + Reset with a fixed text template, against the full SP-CLIP model on four benchmarks. For each dataset, the two configurations use matched random seeds and the same Stage-I checkpoints, so that the performance difference measures the additional contribution of prompt learning and LinUCB routing under the same Stage-I initialization.

Adding the LinUCB prompt stage improves Avg across all four benchmarks, although the magnitude of the improvement varies across datasets. This result suggests that instance-level text-prototype adaptation provides additional benefits beyond task-level LoRA consolidation under both standard and distribution-shifted recognition settings. Last accuracy and per-seed results are provided in the appendix.

Dataset	Stage I	Full SP-CLIP	Δ Avg
CIFAR-100	92.08	92.27	+0.19
ImageNet-100	94.77	95.35	+0.58
ImageNet-A	88.85	89.32	+0.47
ImageNet-R	96.45	96.54	+0.11

Table 3: Cross-dataset effectiveness of the LinUCB prompt stage. Stage I denotes Merge + Reset with a fixed text template. The improvement Δ Avg is computed for each matched run before averaging.

Prompt Routing Strategy	Avg (%) $\uparrow$	Last (%) $\uparrow$
Fixed Template (Stage-I)	91.63	86.95
Key-Prompt retrieval	91.98	87.43
LinUCB with exploitation only	92.22	87.44
Full LinUCB	**92.27**	**87.49**

Table 4: Comparison of different prompt routing strategies on CIFAR-100. The fixed-template baseline denotes the Stage-I model using Merge + Reset without dynamic prompt routing. "LinUCB with exploitation only" uses only the estimated-reward term and excludes both the uncertainty term and the frequency-based exploration bonus.

Prompt Routing Strategies. We compare three prompt strategies using the same Merge + Reset model: a fixed template, Key-Prompt, and LinUCB. The fixed-template baseline corresponds to the Stage-I model without dynamic prompt routing. For Key-Prompt, we compute the cosine similarity between each candidate key and the current image feature, and select the candidate with the highest similarity. LinUCB additionally incorporates historical classification feedback, reward estimation, and uncertainty into the routing decision.

As shown in Table 4, Key-Prompt provides no clear improvement over the fixed-template baseline, indicating that routing based solely on instantaneous feature similarity may not yield consistent benefits. In contrast, Full LinUCB improves Avg by 0.19 percentage points over the fixed-template baseline and achieves the best overall performance. Compared with the exploitation-only variant, Full LinUCB further improves both Avg and Last by 0.05 percentage points, indicating that the uncertainty and frequency-based exploration terms provide additional benefits.

To further examine prompt utilization, we report the normalized prompt selection frequencies and the entropy of the selection distribution for Key-Prompt and LinUCB in the appendix. When warm-up is enabled, the corresponding round-robin selections are excluded from these statistics to avoid bias from the initialization stage.

Mechanistic Analysis of Cross-Task LoRA Management. To analyze the effect of different LoRA management strategies on cross-task parameter interference, we conduct a controlled comparison between Shared LoRA and Merge + Reset on CIFAR-100. Both methods use the same model initialization, class order, LoRA configuration, and training budget, and differ only in how LoRA states are managed across task boundaries.

As shown in Figure 2, Merge + Reset generally maintains higher old-class accuracy during the later stages of the task sequence. It also exhibits lower old-class feature drift in Tasks 2–3, indicating that it can suppress abrupt changes to the base visual representation during the early stages of continual learning. This trend is generally maintained in later tasks. Although the negative cosine ratio fluctuates during the early stages, Merge + Reset achieves a lower ratio on most tasks from Task 4 onward, suggesting that merging historical updates and resetting the low-rank branch can reduce the frequency of oppositely directed updates between the current and previous tasks to some extent.

Overall, these observations are consistent with the design motivation that Merge + Reset reduces persistent competition within a shared low-rank parameter space. Merging consolidates the knowledge acquired from the current task, while branch reinitialization provides a new trainable low-rank space for subsequent tasks and thus preserves plasticity by design. These metrics mainly reflect knowl-

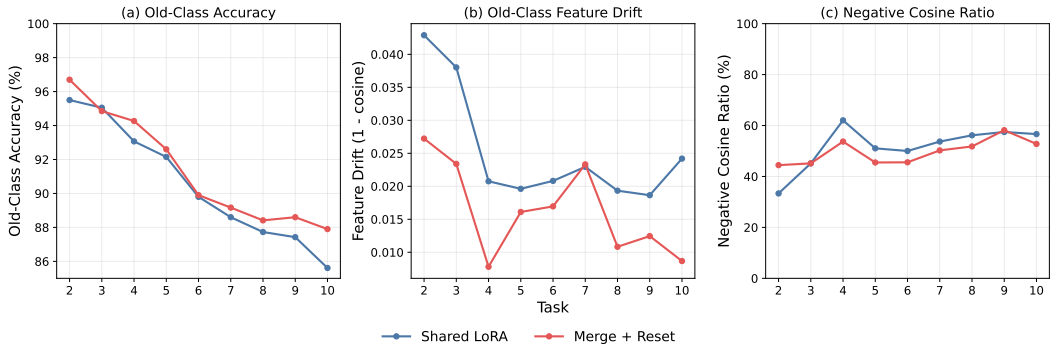


Figure 2: Mechanistic comparison between Shared LoRA and Merge + Reset on CIFAR-100. (a) Accuracy on previously learned classes after each task. (b) Visual feature drift on previously learned-class samples, measured by cosine distance between image features extracted at consecutive task boundaries. (c) Ratio of negative cosine similarities between the current task-level effective weight update and previous task updates. The first task is omitted because these metrics require previously learned classes or cross-task update pairs.

edge retention and representation stability rather than directly establishing improved adaptation or gradient conflict.

4.4 Parameter Overhead

The trainable parameters of SP-CLIP consist only of the LoRA modules and the Prompt Pool. For CLIP ViT-L/14, rank-8 LoRA modules introduce approximately 1.08×10^6 parameters (~ 4.13 MiB), while a Prompt Pool with $M = 10$ candidates, $L_p = 5$, and $d_p = 768$ contains 3.84×10^4 parameters (~ 0.15 MiB). Since the two stages are optimized sequentially, these parameter sets do not need to be optimized simultaneously. After the Stage-I LoRA updates are merged into the base backbone, no task-specific adapter branches are retained during inference. LinUCB also introduces no trainable parameters. Thus, for a fixed prompt-pool size, the additional trainable-parameter and adapter-storage overhead remains constant with respect to the number of incremental tasks. Detailed analyses of the non-gradient routing state and inference complexity are provided in the supplementary material.

5 Conclusion

We presented SP-CLIP, a replay-free class-incremental learning framework that combines task-level Stable-Plastic LoRA consolidation with instance-level prompt routing. At each task boundary, the task-specific LoRA updates are merged into the CLIP backbone, while the low-rank branches are reinitialized to provide a fresh trainable parameter space for subsequent tasks. In addition, LinUCB dynamically selects prompts according to image features, historical classification feedback, reward estimation, and exploration uncertainty.

Experiments on four class-incremental learning benchmarks demonstrate that SP-CLIP achieves consistent improvements in average incremental accuracy over MG-CLIP. Component-wise ablations show that Merge + Reset and LinUCB-based prompt routing provide complementary benefits. Further cross-dataset experiments confirm that the prompt stage improves the performance of the consolidated model under different data distributions. Mechanistic analysis also suggests that Merge + Reset improves old-class retention and reduces visual feature drift during continual learning.

Nevertheless, the current routing mechanism relies primarily on global image features, which may limit its effectiveness in fine-grained local recognition. Future work will investigate local visual representations and more adaptive prompt-routing mechanisms for fine-grained continual learning.

REFERENCES

Marco D'Alessandro, Alberto Alonso, Enrique Calabrés, and Mikel Galar. Multimodal parameter-efficient few-shot class incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3393–3403, 2023.

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. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8320–8329. IEEE, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021b.

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.

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.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Saurav Jha, Dong Gong, and Lina Yao. Clap4clip: Continual learning with probabilistic finetuning for vision-language models. *Advances in neural information processing systems*, 37:129146–129186, 2024.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727. Springer, 2022.

Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19113–19122. IEEE, 2023.

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

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

Kuan-Ying Lee, Yuanyi Zhong, and Yu-Xiong Wang. Do pre-trained models benefit equally in continual learning? In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6474–6482. IEEE, 2023.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

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

Xialei Liu, Xusheng Cao, Haori Lu, Jia-wen Xiao, Andrew D Bagdanov, and Ming-Ming Cheng. Class incremental learning with pre-trained vision-language models. *arXiv preprint arXiv:2310.20348*, 2023.

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.

Zixuan Ni, Longhui Wei, Siliang Tang, Yueting Zhuang, and Qi Tian. Continual vision-language representation learning with off-diagonal information. In *International Conference on Machine Learning*, pp. 26129–26149. PMLR, 2023.

Julien Nicolas, Florent Chiaroni, Imtiaz Ziko, Ola Ahmad, Christian Desrosiers, and Jose Dolz. Mop-clip: A mixture of prompt-tuned clip models for domain incremental learning. *arXiv preprint arXiv:2307.05707*, 2023.

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.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5533–5542. IEEE, 2017.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11909–11919. IEEE, 2023.

Vishal Thengane, Salman Khan, Munawar Hayat, and Fahad Khan. Clip model is an efficient continual learner. *arXiv preprint arXiv:2210.03114*, 2022.

Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhu Lü, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3654–3663, 2023.

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

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 139–149. IEEE, 2022b.

Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefano Soatto. Class-incremental learning with strong pre-trained models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9591–9600. IEEE, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

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

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

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133(3):1012–1032, 2025a.

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, 2025b.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 16795–16804. IEEE, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International journal of computer vision*, 130(9):2337–2348, 2022b.

A Supplementary Experiments and Overhead Analysis

A.1 Task-Update and Representation-Subspace Analysis

To further investigate the effect of different parameter-update strategies, we compare Shared LoRA with Merge + Reset in terms of task-update cosine similarity and representation-subspace overlap. The results are shown in Fig. 3. The diagonal entries correspond to comparisons between a task and itself and are excluded from the cross-task analysis.

Figure 3(a) shows the cosine similarity between text-side task updates. Under Shared LoRA, the off-diagonal entries exhibit non-negligible similarity, indicating that different tasks tend to share part of their update directions. Such sharing may facilitate knowledge transfer, but it can also introduce interference when updates for subsequent tasks accumulate in a common parameter space.

In contrast, the off-diagonal entries under Merge + Reset are generally close to zero. This suggests that merging the learned update into the backbone and resetting the task-specific adaptation space leads to more independent update directions across tasks.

Figure 3(b) shows the overlap between text-side representation subspaces. Shared LoRA produces more evident cross-task subspace overlap, whereas Merge + Reset substantially reduces the overlap between different tasks. This observation is consistent with the task-update cosine similarities and indicates weaker coupling between task-specific text representations.

Figures 3(c) and 3(d) present the corresponding results for the visual branch. Similar trends can be observed: Shared LoRA retains stronger correlations and subspace overlap across tasks, whereas Merge + Reset yields lower cross-task similarity and weaker subspace overlap.

Overall, these results suggest that Merge + Reset reduces the coupling between task updates and task-specific representation subspaces. This mechanism helps mitigate the accumulation of cross-task interference over a long sequence of incremental tasks, while still allowing the consolidated backbone to preserve knowledge acquired from previous tasks.

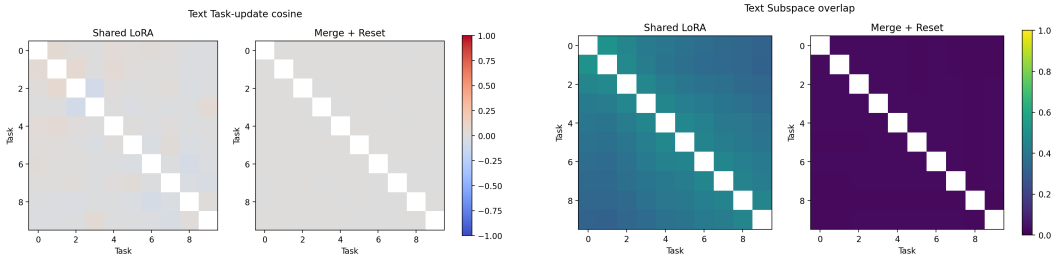
A.2 Parameter, Routing-State, and Inference Overhead

A.2.1 Trainable-Parameter Overhead

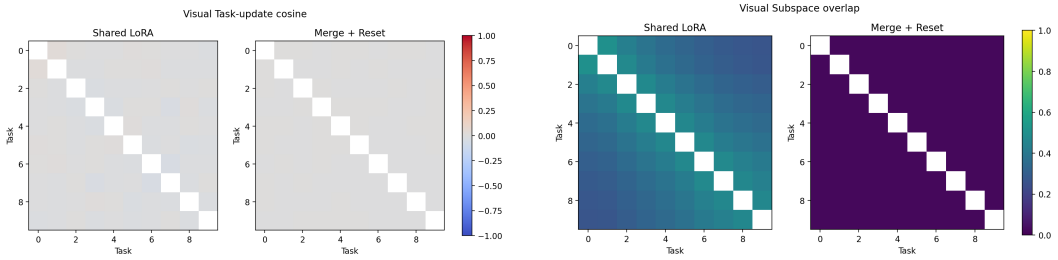
The trainable parameters of SP-CLIP consist only of the LoRA modules and the Prompt Pool. For CLIP ViT-L/14, the rank-8 LoRA modules introduce approximately 1.08×10^6 parameters, corresponding to approximately 4.13 MiB in FP32 precision.

In general, the number of LoRA parameters can be written as

$$N_{\mathrm{LoRA}} = \sum_{\ell \in \mathcal{L}} r \left(d_{\ell,\mathrm{in}} + d_{\ell,\mathrm{out}} \right), \tag{6}$$



(a) Text task-update cosine similarity **(b)** Text subspace overlap



(c) Visual task-update cosine similarity **(d)** Visual subspace overlap

Figure 3: Comparison of Shared LoRA and Merge + Reset in terms of task-update cosine similarity and representation-subspace overlap. (a) Text-side task-update cosine similarity; (b) text-side representation-subspace overlap; (c) visual-side task-update cosine similarity; (d) visual-side representation-subspace overlap. The diagonal entries correspond to self-comparisons and are excluded from the cross-task analysis.

where $\mathcal{L}$ denotes the set of adapted linear layers, r is the LoRA rank, and $d_{\ell,\text{in}}$ and $d_{\ell,\text{out}}$ denote the input and output dimensions of layer ℓ, respectively.

The Prompt Pool contains M candidate prompts, each consisting of L_p context tokens with embedding dimension d_p. Its parameter count is therefore

$$N_{\text{prompt}} = M L_p d_p. \tag{7}$$

With $M = 10$, $L_p = 5$, and $d_p = 768$, the Prompt Pool contains

$$N_{\text{prompt}} = 10 \times 5 \times 768 = 3.84 \times 10^4 \tag{8}$$

parameters, corresponding to approximately 0.15 MiB in FP32 precision.

The LoRA consolidation stage and the Prompt routing stage are optimized sequentially, so these two parameter sets do not need to be optimized simultaneously. After the Stage-I LoRA updates are merged into the base backbone, no task-specific adapter branches are retained during inference. Therefore, for a fixed Prompt Pool size, the additional trainable-parameter and adapter-storage overhead remains constant with respect to the number of incremental tasks.

A.2.2 Non-Gradient Routing-State Memory

LinUCB does not introduce any trainable gradient parameters. For each Prompt arm j, the router maintains a second-order matrix

$$H_j \in \mathbb{R}^{d_c \times d_c}, \tag{9}$$

a feedback vector

$$b_j \in \mathbb{R}^{d_c}, \tag{10}$$

and a selection count n_j.

The total number of routing-state scalars is therefore

$$N_{\text{state}} = M \left(d_c^2 + d_c + 1 \right). \tag{11}$$

For $d_c = 768$ and $M = 10$, this gives

$$N_{\text{state}} = 10\left(768^2 + 768 + 1\right) = 5{,}905{,}930 \tag{12}$$

scalars. Assuming FP32 storage, the corresponding memory consumption is

$$S_{\text{state}} = \frac{4N_{\text{state}}}{2^{20}} \approx 22.5 \text{ MiB}. \tag{13}$$

The routing-state memory grows linearly with the Prompt Pool size M, but it is independent of the number of incremental tasks. Thus, the router does not need to maintain a separate full routing state for every historical task.

A.2.3 Inference-Time Computational Overhead

During inference, an input image x is first processed by the permanently frozen image encoder. Once the Prompt Pool and the class set are fixed, the Prompt–class text features can be computed in advance and cached. Consequently, text encoding does not need to be repeated for every input sample.

For the j-th Prompt arm, LinUCB computes an upper-confidence score of the form

$$\text{UCB}_j(x) = \theta_j^\top x + \alpha\sqrt{x^\top H_j^{-1} x}, \tag{14}$$

where

$$\theta_j = H_j^{-1} b_j \tag{15}$$

and α is the exploration coefficient.

When the inverse matrices or their factorizations are cached, computing the score for one Prompt arm requires $\mathcal{O}(d_c^2)$ operations. The total complexity for all M Prompt arms is therefore

$$\mathcal{O}(Md_c^2). \tag{16}$$

Hence, the additional inference computation consists primarily of one forward pass through the frozen image encoder and the LinUCB routing scores. Since the image encoder remains frozen and the routing state depends on the Prompt Pool size rather than the number of historical tasks, the additional inference overhead does not grow linearly with the number of incremental tasks.

The above memory estimates assume FP32 storage and do not include CUDA workspace memory, temporary activation storage, or optimizer-specific buffers.