PROJECTED ROUTING IN SCALED-UP MAPPINGS FOR CONTINUAL POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual post-training of pre-trained models offers a promising way to acquire new classes while retaining a shared representation and allocating adaptation capacity to individual tasks. However, this setting introduces a fundamental inference problem: the task associated with an input is unknown, yet selecting the correct task-specific adaptation is necessary for accurate classification. Existing approaches either concatenate features from all task-specific modules, which grows the representation dimensionality, or evaluate every module under a selection criterion such as distance or predictive entropy. The latter family is effective when the correct expert is selected, yielding high task-local classification accuracy. However, expert selection remains the bottleneck: batch-level criteria assume that test samples belong to the same task, while the per-sample alternative requires evaluating every stored module, making inference cost grow linearly with the number of tasks. We introduce PRISM (Projected Routing In Scaled-up Mappings) that explicitly separates task identification from classification. It learns a globally consistent router in a high-dimensional random feature space, enabling instance-wise selection of a single task-specific adaptation module without evaluating all adapters. This design provides constant-path inference while maintaining a shared routing function throughout the continual sequence. Across standard class-incremental benchmarks, PRISM outperforms existing methods, with particularly pronounced improvements under challenging distribution shifts. These results demonstrate that explicitly decoupling task identification from class prediction provides an effective and scalable mechanism in continual post-training.

1 INTRODUCTION

Class-incremental learning (CIL) considers a model that learns a sequence of tasks with disjoint class sets and must recognize all classes observed so far without access to task identity at inference. This setting is important for systems that continuously encounter new categories, where retaining and retraining on all historical data may be impractical due to storage, privacy, or computational constraints. Directly fine-tuning on successive tasks, however, can overwrite previously acquired knowledge and cause catastrophic forgetting (McCloskey & Cohen, 1989). Traditional CIL methods mitigate this through regularization, rehearsal, or architectural mechanisms (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Aljundi et al., 2018; Zenke et al., 2017; Rebuffi et al., 2017; Prabhu et al., 2020; Lopez-Paz & Ranzato, 2017; Aljundi et al., 2017; Yan et al., 2021; Wang et al., 2022a; Mallya & Lazebnik, 2018). More recently, strong pre-trained backbones have shifted CIL toward *continual post-training*, where the backbone remains largely fixed while lightweight, task-specific components or classifiers are adapted to new classes. This paradigm preserves a shared representation while providing task-specific plasticity, but introduces a distinct inference challenge: once multiple task-specific modules are accumulated, the task identity of a test sample is unknown, and the model must determine which adaptation should be applied before making the class prediction.

This routing problem is largely orthogonal to how the task-specific modules themselves are parameterized. A routing mechanism (i) should identify the appropriate task for each sample, (ii) remain stable as new tasks are introduced, and (iii) avoid coupling its decision to the composition of the test batch. Existing selection strategies can violate one or more of these requirements, particularly when test samples from different tasks are interleaved. Our preliminary experiments in Section 3.2 show that this issue can lead to substantial accuracy degradation under shuffled inference, motivating an explicit instance-wise routing mechanism for modular continual post-training.

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

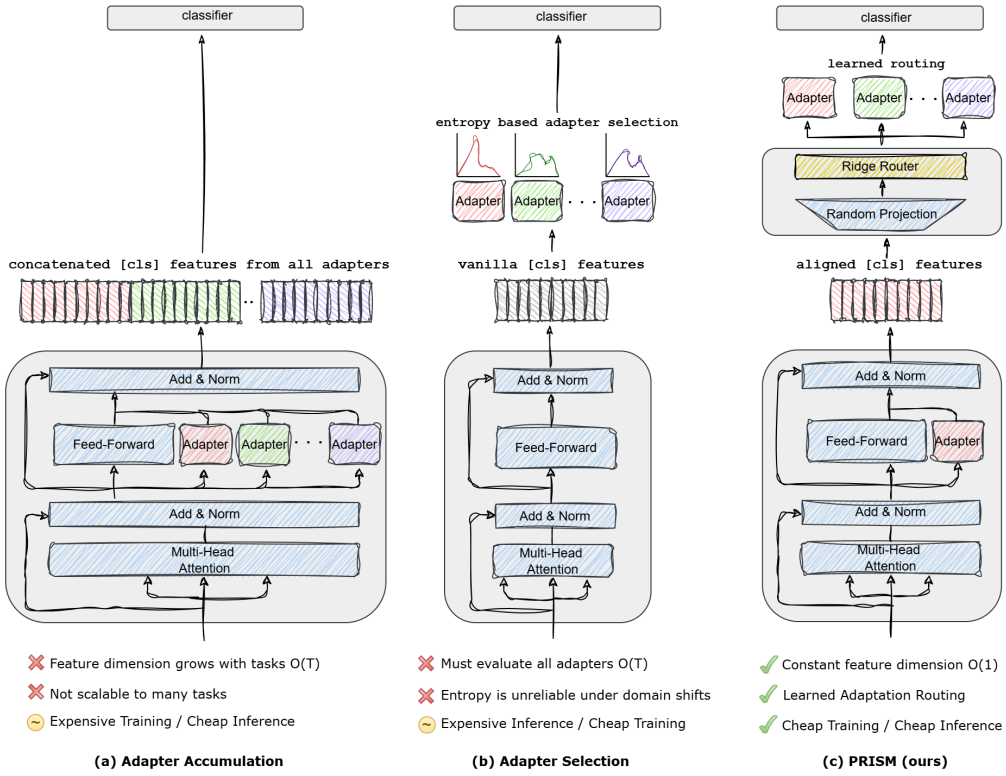


Figure 1: **Use of per-task adapters in continual post-training.** **(a)** Concatenating `[CLS]` features from all adapters grows and increases representation dimensionality with the number of tasks. **(b)** Entropy-based selection evaluates every adapter, which grows inference cost linearly. **(c)** PRISM gates the lifted `[CLS]` feature with a closed-form ridge router, evaluating one adapter at constant cost.

In this work, we introduce **PRISM** (**P**rojected **R**outing **I**n **S**caled-up **M**appings), which treats task identification as a separate prediction problem from class recognition. Given the frozen `[CLS]` representation, PRISM first maps each sample into a high-dimensional random feature space and uses a global ridge router to predict its task identity. The selected task-specific module is then used for class prediction, so each sample follows a single adaptation path regardless of the composition of the test batch. The resulting framework separates two complementary sources of capability. The shared router provides a stable and scalable mechanism for deciding *which* adaptation should be used, while the task-specific module provides the flexibility to determine *how* the input should be represented for classification. Importantly, the router is learned once over the shared representation and remains globally consistent as new tasks are introduced, avoiding the need to evaluate all task-specific modules at inference time. PRISM thus combines the plasticity of task-specific adaptation with efficient instance-wise routing, requiring only a single adaptation path per input rather than evaluating multiple experts. Our main contributions are summarized as follows:

- We identify task routing as a distinct inference problem in modular continual post-training. As task-specific adaptation modules accumulate, a model must determine the appropriate adaptation for each test instance without access to its task identity, while supporting arbitrary test-batch compositions.
- We introduce PRISM, an instance-wise router estimated in closed form over accumulated sufficient statistics in a high-dimensional random feature space. Because the router is never updated by gradient descent on the current task alone, it does not inherit the forgetting dynamics of the sequence it is meant to index. It activates exactly one task-specific module per input, yielding $O(1)$ adapter evaluations in the number of tasks.
- We demonstrate the effectiveness of PRISM across six standard class-incremental learning benchmarks, where it delivers larger improvements under challenging distribution shifts while substantially reducing the inference cost associated with multi-module evaluation.

2 Related Work

Foundation models have shifted CIL toward continual post-training, where the backbone is largely frozen, and only a small set of parameters is adapted (Wu et al., 2022; Zhou et al., 2024a; Yildirim et al., 2026a). These approaches preserve most of the pre-trained backbone while allocating an adequate amount of task-specific capacity for continual adaptation.

Analytic classifiers and random projections. One line of work freezes the pretrained backbone and fits the classifier analytically from its representations. SimpleCIL (Zhou et al., 2024a) represents classes using their prototypes, while RanPAC (McDonnell et al., 2023) maps frozen features into a higher-dimensional random feature space and fits a ridge classifier. More generally, random feature mappings provide an efficient mechanism for constructing expressive feature spaces without introducing additional learned backbone parameters (Rahimi & Recht, 2007). These approaches maintain a shared classifier over the observed class set, yielding a simple inference procedure whose computational structure is independent of the test-batch composition and the ordering of classes.

Task routing and modular inference. A major part of continual post-training methods introduces task-specific parameters to provide adaptation capacity. This design creates a distinct inference challenge: given an input with unknown task identity, the model must determine which task-specific adaptation should be activated. Several existing approaches mitigate this challenge by completely avoiding explicit task identification altogether. EASE (Zhou et al., 2024b) concatenates task-specific subspaces, while MOS (Sun et al., 2025) merges task-specific adapters into a shared representation. Prompt-based methods (Wang et al., 2022c;b) instead perform instance-dependent prompt selection. These can be viewed as alternative mechanisms for exploiting task-specific capacity without explicitly predicting the underlying task, but they either increase the dimensionality of the shared representation, merge task-specific knowledge, or require instance-dependent selection among learned prompts. Explicit expert selection provides a more direct formulation of the problem and is closely related to task routing and mixture-of-experts models (Jacobs et al., 1991; Shazeer et al., 2017), where a gating mechanism determines the computation path for an input. In continual post-training, however, the task identity is unavailable at inference, and the number of candidate paths grows as new tasks are introduced. TOSCA (Yildirim et al., 2026b) provides a representative approach by maintaining a separate adaptation module for each task and selecting among them using predictive entropy. Such selection requires evaluating candidate modules to determine which adaptation best explains the input, causing the routing cost to grow with the number of accumulated tasks.

PRISM instead treats task identification as an explicit prediction problem. It learns a global, instance-wise analytic router from the shared frozen representation and predicts the task-specific adaptation path. This formulation avoids both the need to combine all task-specific representations and the need to evaluate all candidate adaptations, requiring only the selected module for each input. Consequently, PRISM preserves task-specific adaptation capacity while keeping the number of adapter evaluations constant with respect to the number of tasks.

3 Background

In this section, we first formalize the continual post-training setting and then present a set of preliminary observations that motivate the need for instance-wise task routing.

3.1 Problem Setting

We consider an image classification model exposed to a sequence of T tasks, $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$. Each task $\mathcal{T}_t$ consists of a dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{n_t}$, where the labels belong to a task-specific class set Y_t. We consider the class-incremental setting, in which the class sets are disjoint, $Y_t \cap Y * t' = \emptyset$ for $t \neq t'$, while the underlying input distribution may change across tasks. The network consists of a pretrained vision foundation model $\Phi : \mathcal{X} \to \mathbb{R}^d$ and a unified classifier $W \in \mathbb{R}^{|\mathcal{Y}| \times d}$, where $\mathcal{Y} = \bigcup_{t=1}^{T} Y_t$ denotes the set of all observed classes. Given an input $\mathbf{x}$, the classifier produces $f(\mathbf{x}) = W\Phi(\mathbf{x})$ and predicts jointly over all classes in $\mathcal{Y}$. The objective of class-incremental learning is to learn a predictor $f : \mathcal{X} \to \mathcal{Y}$ that performs well across all observed tasks while **task identity is unavailable at inference time**. In addition, following the standard continual learning setting, the learner **cannot access or store training samples from previously observed tasks** (Wang et al., 2022c;b; Smith et al., 2023; Zhou et al., 2024b).

3.2 Preliminary Observations

Finding 1: Mixed-task inference exposes the limitations of expert selection. We first examine whether task-specific adaptation remains reliable when test samples from different tasks are interleaved within a batch. Table 1 compares task-homogeneous and mixed-task inference. Most methods show only minor changes under this intervention, whereas the best-performing method TOSCA exhibits a pronounced degradation, with its final accuracy decreasing from 65.37% to 9.45%. This behavior follows from its batch-level expert selection: a single task-specific module is selected for the entire batch, so when samples from multiple tasks are mixed, the selected expert can only be appropriate for a subset of the inputs. A natural mitigation is a per-sample variant that applies the same entropy criterion independently to each input while still evaluating all stored modules. This removes the dependence on batch composition by construction and is effective: it reaches 59.10 under mixed-task inference, the highest final accuracy of any configuration we measured on this benchmark. We take this as evidence that the decomposition into task-local experts is sound and that the experts themselves are strong: the collapse in the batch-level setting is attributable to the selection rule, not to the adaptation modules. The per-sample variant, however, converts the problem into a cost problem: it requires a forward pass through every stored module for every test sample, so inference scales as $O(T)$. Selection by entropy is also indirect, in that the lowest-entropy expert is not constrained to correspond to the true task. These observations motivate treating task identification as an explicit prediction problem, solved once over the shared representation, rather than as a by-product of evaluating all candidates.

Table 1: Final accuracy A_T under task-homogeneous and mixed-task inference on ImageNet-A B0 Inc20 scenario with ViT-B/16-IN21K.

Method	shuffle=False	shuffle=True	Δ
SimpleCIL	48.77	48.39	−0.38
RanPAC	54.76	54.73	−0.03
L2P	42.80	41.08	−1.72
DualPrompt	42.33	40.56	−1.77
CODAPrompt	47.18	41.12	−6.06
EASE	54.99	45.11	−9.88
MOS	54.60	53.89	−0.71
TOSCA	65.37	9.45	−55.92
TOSCA (per-sample)	59.10	59.10	0.00

Finding 2: The frozen [CLS] representation is weak for the task separability. We next ask whether task identity can be reliably inferred directly from the frozen [CLS] representation. Figure 2 illustrates the task-wise feature distributions before and after the random feature mapping. In the original feature space, samples from different tasks exhibit substantial overlap, suggesting that task identity is not easily recoverable from the frozen representation. Applying the random feature mapping substantially alters this geometry, yielding visibly stronger task-wise separation. Thus, the frozen representation does not provide a sufficiently discriminative space for task identification, whereas the projected space makes task structure substantially more accessible to a simple global classifier. This observation motivates performing task routing in a dedicated projected feature space rather than directly in the original backbone representation.

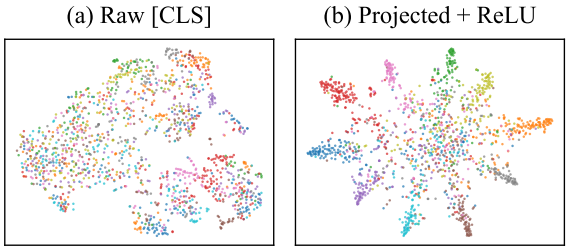


Figure 2: **Task separability on ImageNet-A.** t-SNE embeddings computed with identical samples and hyperparameters. (a) Tasks overlap substantially in the [CLS] space, (b) but form visibly more distinct groups after projection.

Finding 3: Module routing itself is also susceptible to continual forgetting. We finally investigate whether the task-routing function can itself be learned incrementally using a conventional gradient-based predictor. Figure 3 shows the resulting task-routing accuracy as the number of observed tasks increases. Although the gradient-trained router initially learns to distinguish the observed tasks, its routing accuracy progressively deteriorates as new tasks are introduced, eventually approaching the 10%

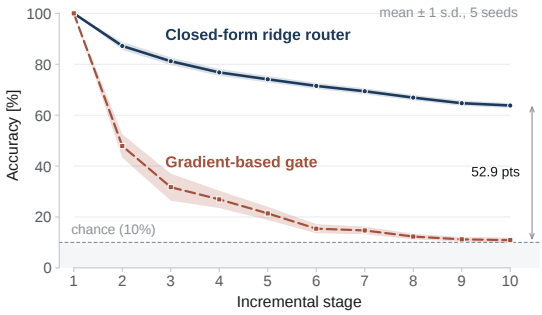


Figure 3: Sequential training causes router forgetting.

random-prediction level. This degradation indicates that the routing function is itself subject to continual forgetting: updating the router using only the current task progressively overwrites its ability to recognize previously observed tasks. In contrast, the closed-form router remains stable as the task sequence grows because its solution incorporates information from the accumulated task data rather than replacing the previously learned decision rule. Thus, task routing cannot be treated as an ordinary predictor that is repeatedly optimized on the current task. Instead, it requires a globally consistent estimation procedure that can incorporate newly observed tasks while preserving the routing structure established for earlier ones. This observation motivates the closed-form ridge formulation used by PRISM.

4 METHOD

We decompose class prediction into two estimation problems and solves each with the mechanism its scope demands. The task router must remain valid across the entire sequence, and Section 3 shows that such a component cannot be obtained by gradient descent on the arriving task without inheriting the forgetting. We therefore confine gradient training to components that observe one task and are then frozen. The remainder of this section derives the design from the constraints of Section 3 (§4.1), formalises the decomposition (§4.2), specifies the analytic router (§4.3) and the learned components (§4.4). For more detail please see out Appendix A.1–A.5.

4.1 DESIGN PRINCIPLE

Section 3 establishes three constraints on any routing mechanism for modular continual post-training. Routing must be *instance-wise*, since batch-level criteria are only valid when test samples share a task (Finding 1). It must be *nonlinear* in the frozen representation, since tasks are not separable in the `[CLS]` space (Finding 2). And it cannot be obtained by gradient descent on the arriving task, since a router updated this way is itself a class-incremental learner and inherits the forgetting it was introduced to remove (Finding 3). These constraints admit a narrow design space:

> **Insight (Key Design Principle).**
> Every component whose behaviour must span tasks is estimated in closed form from accumulated statistics. Every component trained by gradient descent observes one task and is never updated again.

The router spans all tasks and must be therefore estimated analytically. The shared backbone adaptation and task-specific modules observe one task each and must be frozen thereafter. We show in Section 5.3 that violating the invariant at the router collapses routing to chance regardless of the gate's architecture.

4.2 FACTORISATION OF THE PREDICTIVE DISTRIBUTION

Let $t(y)$ denote the task whose class set contains y; this is well defined because the class sets are disjoint. Conditioning on task identity gives

$$p(y \mid x) \;=\; \sum_{t=1}^{T} p(t \mid x)\, p(y \mid x, t) \;=\; p\big(t(y) \mid x\big)\, p\big(y \mid x, t(y)\big), \tag{1}$$

where the second equality uses $p(y \mid x, t) = 0$ for $t \neq t(y)$. Eq. 1 separates the problem into a *task posterior* $p(t \mid x)$ and a set of *task-conditional class posteriors* $p(y \mid x, t)$, and makes explicit that these are distinct problems: the first is T-way and shared, the second is $|\mathcal{Y}_t|$-way and local. Maximising Eq. 1 over all $y \in \mathcal{Y}$ requires every task-conditional model, and therefore $\mathcal{O}(T)$ evaluations per sample. Existing expert-selection methods incur exactly this cost, because they do not model $p(t \mid x)$ separately: they infer task identity *from* the task-conditional models, which requires evaluating all of them. PRISM instead models the task posterior directly with a component that is cheap to evaluate, and routes hard:

$$\hat{t} = \arg\max_{t}\; p(t \mid x), \qquad \hat{y} = \arg\max_{y \in \mathcal{Y}_{\hat{t}}}\; p(y \mid x, \hat{t}). \tag{2}$$

The design question is therefore how to estimate $p(t \mid x)$ well and cheaply, subject to the constraints of Section 4.1.

4.3 Analytic task router

Routing space. Let $z = \Phi(x) \in \mathbb{R}^d$ be the frozen representation. We draw a random projection $\mathbf{P} \in \mathbb{R}^{d \times D}$ once, before the sequence begins, with $D \gg d$, and define the routing feature

$$r(x) \;=\; \sigma\big(z^\top \mathbf{P}\big) \in \mathbb{R}^D, \qquad \sigma = \mathrm{ReLU}. \tag{3}$$

A linear model on $r(x)$ is a random-feature approximation to a kernel machine (Rahimi & Recht, 2007), so Eq. 3 supplies the nonlinearity required by Finding 2 while leaving the estimation problem linear in its parameters, and therefore solvable in closed form. Because $\mathbf{P}$ is sampled once and never updated, all tasks are embedded in a common routing space for the entire sequence; no realignment is needed when a task is added. We use the same construction as RanPAC (McDonnell et al., 2023), which applies it to build a classifier over the accumulated class set $\mathcal{Y}$; here it carries a T-way task posterior instead, and the classification problem is delegated to the task-conditional models.

Estimation. Let $\mathbf{R}_t \in \mathbb{R}^{n_t \times D}$ hold the routing features of task t's training samples and $\mathbf{Y}_t^{\text{task}} \in \{0,1\}^{n_t \times T}$ their one-hot task labels. The router solves ridge over all tasks observed so far,

$$\mathbf{W}_R \;=\; (\mathbf{G} + \lambda_R \mathbf{I})^{-1}\, \mathbf{C}, \qquad \mathbf{G} = \sum_{s \le t} \mathbf{R}_s^\top \mathbf{R}_s, \qquad \mathbf{C} = \sum_{s \le t} \mathbf{R}_s^\top \mathbf{Y}_s^{\text{task}}, \tag{4}$$

and predicts $p(t \mid x) \propto \big[\, r(x)^\top \mathbf{W}_R \,\big]_t$. The two quantities $\mathbf{G} \in \mathbb{R}^{D \times D}$ and $\mathbf{C} \in \mathbb{R}^{D \times T}$ are the sufficient statistics of the problem, and both are *sums over samples*. On arrival of task t we update

$$\mathbf{G} \leftarrow \mathbf{G} + \mathbf{R}_t^\top \mathbf{R}_t, \qquad \mathbf{C} \leftarrow \big[\, \mathbf{C} \;\big|\; \mathbf{R}_t^\top \mathbf{1}_{n_t} \,\big], \tag{5}$$

appending one column to $\mathbf{C}$ for the new task, and recompute $\mathbf{W}_R$ from Equation 4. No sample from any previous task is revisited or stored, satisfying the rehearsal-free constraint of Section 3.1.

Proposition (Order-independent ridge readout).
Assume the backbone Φ and projection $\mathbf{P}$ are fixed. Let $\mathbf{W}_R^{(T)}$ be the router obtained by applying Equation 5 over $\mathcal{T}_1, \ldots, \mathcal{T}_T$, and let $\mathbf{W}_R^\star$ be the ridge solution computed with simultaneous access to $\bigcup_{t=1}^{T} \mathcal{D}_t$. Then $\mathbf{W}_R^{(T)} = \mathbf{W}_R^\star$ for any permutation of the task sequence.

Proof. $\mathbf{G}$ and $\mathbf{C}$ are sums of per-sample outer products, and addition is commutative and associative, so both are independent of the order in which samples and tasks are accumulated and equal their values under joint access. $\mathbf{W}_R$ is a deterministic function of $(\mathbf{G}, \mathbf{C})$ by Equation 4. The projection $\mathbf{P}$ and backbone Φ are fixed, so $\mathbf{R}_s$ does not depend on the sequence either.

Proposition is the formal content of Observation 3. The router does not *resist* forgetting; it has no mechanism by which forgetting could occur, since the estimate at session t is not a function of the estimate at session $t-1$. It is also exactly invariant to task ordering, a property no gradient-trained gate possesses, and one we verify empirically in Section 5.3. The router is estimated analytically because it spans the sequence. Everything else in PRISM is trained by gradient descent and obeys the second half of Design Principle.

4.4 Learned components

Shared representation. The backbone Φ is a foundation model with MLP adapters at every block, optimised on $\mathcal{T}_1$ and frozen thereafter. Since Φ is read by both the router and every expert, it is a shared component, and may therefore be trained only at the one point where doing so cannot overwrite anything; a later update would trade the current task's alignment against the representation on which the previous experts and the accumulated routing statistics were built.

Task experts. Each task-conditional distribution uses an expert module E_t and linear head $\mathcal{W}_t$ on the frozen representation, $p(y \mid x, t) \propto [\mathbf{W}_t^\top E_t(\Phi(x))]_y$. We adopt the parameterisation and objective of TOSCA (Yildirim et al., 2026b) without modification; our contribution being how a module is selected at inference rather than how it is parameterised. As $(E_t, \mathbf{W}_t)$ is fitted on $\mathcal{D}_t$ alone and distinct tasks own disjoint parameters, the experts cannot interfere. Every cross-task interaction is thus confined to $(\mathbf{G}, \mathbf{C})$, which combines information by summation, in closed form, and provably without regard to order (Proposition).

Table 2: Average accuracy ($\bar{\mathcal{A}}$) and last accuracy ($\mathcal{A}_T$) on six datasets with **ViT-B/16-IN21K**. 'IN-R/A' stands for 'ImageNet-R/A', and 'OmniBench' stands for 'OmniBenchmark.' We report all compared methods with their source code and show the best performance in bold.

Method	CIFAR B0 Inc5		CUB B0 Inc10		IN-R B0 Inc20		IN-A B0 Inc20		OmniBench B0 Inc30		VTAB B0 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Joint	–	96.21 ±1.0	–	92.62 ±1.1	–	81.92 ±1.4	–	67.97 ±1.9	–	85.44 ±1.2	–	94.96 ±1.2
SimpleCIL	86.48 ±0.72	81.28 ±0.00	91.62 ±1.12	86.77 ±0.00	61.40 ±0.72	55.11 ±0.00	58.67 ±0.90	48.39 ±0.00	79.68 ±1.33	73.30 ±0.00	90.65 ±0.98	84.45 ±0.00
RanPAC	93.43 ±0.63	90.02 ±0.27	93.09 ±0.61	89.27 ±0.49	83.87 ±0.52	78.86 ±0.17	63.47 ±1.45	54.73 ±0.73	85.84 ±0.87	79.44 ±0.35	94.91 ±0.57	91.93 ±0.69
L2P	84.55 ±1.74	79.75 ±1.15	73.96 ±2.00	62.38 ±0.42	75.60 ±0.59	70.93 ±0.20	48.21 ±0.83	41.08 ±0.96	73.74 ±1.74	64.65 ±1.00	75.38 ±5.53	59.37 ±6.79
DualPrompt	85.55 ±0.95	79.87 ±0.38	81.04 ±1.61	70.59 ±1.31	71.60 ±0.34	66.50 ±0.97	52.53 ±1.24	42.33 ±0.77	75.66 ±1.27	66.71 ±0.68	86.70 ±2.66	75.86 ±3.82
CODAPrompt	87.36 ±0.76	81.63 ±0.57	82.09 ±1.74	70.56 ±1.18	78.55 ±0.61	73.20 ±0.47	51.67 ±0.49	41.12 ±0.54	77.89 ±0.93	69.30 ±0.55	85.79 ±3.42	74.90 ±2.48
APER	89.54 ±0.88	84.90 ±0.31	91.63 ±1.12	86.77 ±0.03	75.76 ±0.78	68.24 ±0.50	59.05 ±0.91	48.94 ±0.14	80.48 ±1.17	74.07 ±0.29	90.60 ±0.96	84.34 ±0.20
EASE	90.82 ±0.67	85.97 ±0.70	90.17 ±1.21	84.47 ±1.06	82.30 ±0.55	76.42 ±0.21	57.29 ±0.86	45.11 ±1.62	76.04 ±1.59	68.31 ±0.54	89.78 ±1.21	82.94 ±0.78
MOS	93.15 ±0.63	89.73 ±0.23	93.54 ±0.52	90.07 ±0.18	82.98 ±0.69	78.59 ±0.35	61.64 ±1.80	53.89 ±1.11	85.86 ±1.00	80.08 ±0.09	95.44 ±0.36	91.78 ±0.55
TOSCA	73.85 ±2.55	69.86 ±1.14	62.34 ±3.25	49.04 ±1.72	30.54 ±1.22	13.87 ±1.39	24.43 ±1.87	9.45 ±1.43	52.28 ±0.65	35.23 ±1.61	68.11 ±5.12	42.94 ±2.44
PRISM	**94.45** ±0.65	**91.42** ±0.31	**93.62** ±0.59	**90.09** ±0.22	**85.84** ±0.36	**81.34** ±0.31	**67.27** ±1.79	**59.51** ±0.72	**86.42** ±0.93	**80.36** ±0.39	**95.87** ±0.43	**92.78** ±0.19

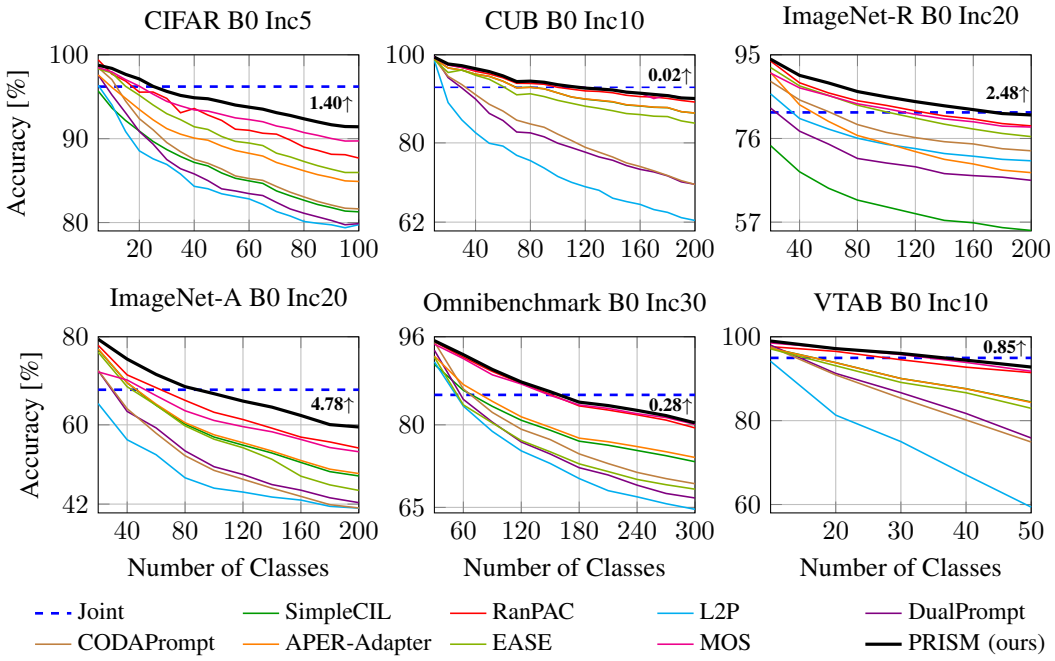


Figure 4: Last accuracy ($\mathcal{A}_T$) after each session under *shuffled* (instance-wise) inference, averaged over five seeds, using ViT-B/16-IN21K. Annotations give PRISM's gain over the runner-up at the final session.

5 Results

We evaluate in the task-agnostic inference setting introduced in Section 3, where samples from previously observed tasks may be arbitrarily interleaved within an evaluation batch. This setting directly tests whether a method can make predictions independently of task identity and batch composition.

5.1 State-of-the-art Comparison

Main experiments. Table 2 reports the full comparison: PRISM attains the highest score on all six benchmarks. The gains are not uniform, and we distinguish them: under a two-sample t-test over five seeds, the improvement in $\mathcal{A}_T$ is significant on ImageNet-A (+4.78), ImageNet-R (+2.48) and CIFAR (+1.40), all $p < 0.01$, whereas on CUB (+0.02), OmniBenchmark (+0.28) and VTAB (+0.85) it lies within seed noise and we claim parity rather than improvement. Figure 4 shows the advantage is not an artefact of the final session: PRISM leads at essentially every session, and the gap widens as the sequence grows, from 1.4 points after the first ImageNet-A session to 4.8 after the tenth. Its curves are also flatter, with EASE starting competitively at 76.48 but ending at 45.11, against PRISM's 79.44 → 59.51. Please see our Appendix A.6 for more results.

Figure 5: Router ablation on ImageNet-A, mean ± std over five seeds. Group 1 varies the routing space and projection dimension; Group 2 replaces the ridge with an incrementally learned gate; Group 3 disables the first-session adaptation. Best value per column among the routers in bold.

Router	Routing Acc. [%]	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
Upper bound: Oracle routing			
Ground-truth task	–	81.82 ± 0.95	–
Group 1: routing space and dimension D			
Raw `[CLS]` (no projection)	37.43 ± 1.48	34.59 ± 1.48	47.34 ± 2.74
Proj + ReLU, D=5000	62.34 ± 0.85	57.94 ± 0.94	66.16 ± 1.92
Proj + ReLU, D=10000	62.99 ± 0.56	58.66 ± 0.79	66.96 ± 1.96
Proj + ReLU, D=15000 (ours)	**63.86** ± 0.99	**59.34** ± 0.82	67.27 ± 1.91
Proj + ReLU, D=20000	63.57 ± 0.79	59.11 ± 0.74	**67.51** ± 1.87
Group 2: learned gate instead of ridge			
Linear gate w/o ReLU	10.70 ± 2.41	8.69 ± 2.35	23.69 ± 0.28
Linear gate w/ ReLU	10.69 ± 2.43	8.68 ± 2.37	23.62 ± 0.26
MLP gate w/ ReLU	10.69 ± 2.43	8.77 ± 2.29	23.65 ± 0.32
Group 3: no backbone adaptation			
Frozen ViT	58.74 ± 0.82	53.55 ± 0.53	61.57 ± 1.45

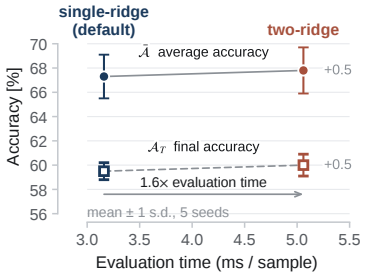


Figure 6: Linear Head vs. Ridge Head

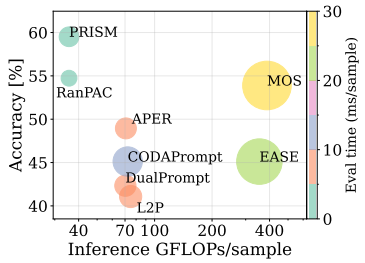


Figure 7: Efficiency vs. Performance

Gains are largest where the pre-trained representation is weakest. The structure of the improvement is informative. Ranking the benchmarks by SimpleCIL's final accuracy, a proxy for how well the frozen IN21K representation already separates the downstream classes, gives CUB (86.77), VTAB (84.45), CIFAR (81.28), OmniBenchmark (73.30), ImageNet-R (55.11) and ImageNet-A (48.39); PRISM's gain over the runner-up broadly follows this ordering in reverse, from +4.78 on ImageNet-A and +2.48 on ImageNet-R down to +0.02 on CUB. Where the pre-trained features are near-saturated, every competent method clusters within a point and little headroom remains to contest; where a genuine pre-training/downstream domain gap exists, PRISM separates from the field. As that regime is precisely the one in which continual adaptation is needed at all, we take it as the substantive result rather than the aggregate average.

Stability vs. plasticity. High $\bar{\mathcal{A}}$ indicates strong adaptation and serves as a proxy for plasticity, while the $\bar{\mathcal{A}} - \mathcal{A}_T$ gap measures cumulative degradation as classes accumulate and thus proxies stability. PRISM degrades least on all three domain-gap benchmarks: by 7.76 points on ImageNet-A against 8.74 for RanPAC, 10.28 for SimpleCIL and 12.18 for EASE; by 4.50 on ImageNet-R against 5.01 and 5.88 for RanPAC and EASE; and by 3.03 on CIFAR against 5.20 and 4.85 for SimpleCIL and EASE. PRISM therefore does not purchase retention at the cost of plasticity: it is simultaneously the most accurate method at the first session and the slowest to degrade thereafter.

5.2 Inference Efficiency and Scalability

PRISM is pareto-dominant. PRISM occupies the upper-left corner of the accuracy–compute trade-off, with no competing method achieving both higher accuracy and lower inference cost. It achieves the highest benchmark accuracy at $\approx$ 15–20 GFLOPs/sample, comparable to a single ViT-B/16 forward pass (17.6 GFLOPs), while also attaining the lowest measured latency ($\approx$ 3 ms/sample). This efficiency comes from a single backbone forward pass followed by a routed adaptation module whose cost is independent of the number of observed tasks. Thus, PRISM achieves $\mathcal{O}(1)$ inference complexity in T, with constant per-sample cost across sessions.

Compute does not buy accuracy in this regime. The evaluated methods occupy distinct compute regimes, yet substantially higher inference cost does not translate into higher accuracy. The most expensive cluster of MOS, EASE, and TOSCA requires approximately 380–400 GFLOPs/sample, or roughly 22× a single backbone forward pass, while achieving substantially lower final accuracy than PRISM. MOS provides the clearest comparison: at $\approx$ 28 ms/sample, it is nearly 10× slower than ours yet 5.62 points less accurate. A second cluster of L2P, DualPrompt, CODA-Prompt, and APER operates at $\approx$ 57–60 GFLOPs/sample and $\approx$ 8 ms/sample, consistent with their two-pass query-then-prompt inference, but remains 10–18 points below PRISM. Overall, these results show that additional inference compute does not compensate for a less effective inference design.

Relevant comparison. RanPAC provides the closest controlled comparison to PRISM: both operate with a single backbone forward pass and at comparable inference cost, requiring approximately 3 ms/sample and similar GFLOPs. At this matched operating point, PRISM improves final accuracy by 4.78 points (59.51 vs. 54.73). Importantly, this gain is not obtained by increasing test-time compute. PRISM basically builds on the RanPAC representation and augments it with a lightweight router that selectively activates the gated TOSCA modules, replacing TOSCA's entropy-based selection mechanism with effective routing. This enables additional task-specific adaptation while preserving the same single-pass inference budget and, consequently, essentially the same measured inference cost. This comparison therefore isolates the benefit of the router-enabled gated adaptation: PRISM retains RanPAC's efficient operating point while substantially improving final accuracy.

5.3 Ablation and Further Discussion

Two-ridge: Router + Classifier. Figure 6 compares the our existing design against a two-ridge variant where the classifier is also a ridge solution instead of linear head. Adding a second ridge increases per-sample evaluation time from 3.16 to 5.06 ms, corresponding to a $1.6\times$ efficiency drop, while improving both average ($\bar{A}$) and last ($\mathcal{A}_T$) accuracy by less than half a point. The two configurations therefore exhibit comparable accuracy despite the substantially higher latency of the two-ridge variant. This favorable accuracy–latency trade-off supports the single-ridge formulation as the default throughout our experiments.

Projecting router representations. Routing directly from the frozen `[CLS]` representation identifies the correct task for only 37.43% of samples and yields 34.59% final accuracy, consistent with the task overlap visible in Figure 2. Lifting the same representation through a random projection with a ReLU nonlinearity raises routing accuracy above 62% at every dimension tested, and final accuracy by more than 23 points, without introducing any learned parameters into the routing path. Within the projected regime the choice of D matters little: routing accuracy moves by 1.5 points across D and final accuracy peaks at D=15,000 before saturating. The determining factor is therefore the routing space rather than its dimension, and the gain comes from making task structure linearly accessible rather than from added capacity.

Where the remaining error lies. To separate routing error from expert error we replace the router with ground-truth task identity while leaving the task-specific modules unchanged. This oracle reaches 81.82% final accuracy, against 59.34% for PRISM. The two error sources can then be attributed by counterfactual: supplying perfect routing to the current experts is worth $+22.48$ points, whereas supplying perfect experts to the current router is worth only $+4.52$ points, since accuracy is then bounded above by the routing accuracy of 63.86%. Task identification is thus the dominant term by roughly a factor of five, and the 8.46-point residual gap to joint training reported in Section 5.1 is a routing gap rather than a capacity gap. We note that expert accuracy conditional on *correct* routing is $59.34/63.86 = 92.9\%$, appreciably higher than the 81.82% obtained under oracle routing over all samples, meaning that the samples the router identifies correctly are also the ones its expert classifies most reliably, so routing errors are concentrated on intrinsically harder inputs.

6 Conclusion

Continual post-training has largely treated task-specific capacity as the difficulty and selection as an implementation detail. Our results invert that emphasis: the decomposition into task-local experts is sound, and almost all remaining error is a failure to identify which expert to use. Routing, not adaptation, is the binding constraint in this setting, and it is a harder problem than it appears, since a router itself susceptible to forgetting and collapses to chance regardless of its architecture. PRISM makes this problem tractable: estimating the router in closed-form statistics breaks the recursion, giving a solution provably identical to one computed with simultaneous access to every task. This is enough to attain the highest accuracy of any method we evaluate, with gains of 1.4–4.8 points where a genuine domain gap exists and parity where the frozen representation is already near-saturated, at roughly $22\times$ less inference compute than the expert-selection methods it outperforms.

Limitations. Although PRISM returns state-of-the-art performance with 63.86% routing accuracy, it operates twenty points below its own oracle, and closing that gap through richer routing representations, calibrated rejection for ambiguous inputs, or expert sharing that reduces the number of candidates would yield more than any further improvement to task-specific capacity.

AI USE STATEMENT

Generative AI tools were employed as editorial aids during manuscript preparation. Their use was limited to improving clarity and conciseness of the text, refining wording, assisting with figure and table captions, and producing LaTeX code for formatting tables and visualizations. These tools were not used to formulate research hypotheses, design the proposed method, generate experimental findings, or implement the training and evaluation software. All AI-assisted material underwent author review and revision prior to inclusion. The authors assume full responsibility for all content presented in this paper, including any material that benefited from AI-assisted editing.

ETHICS STATEMENT

PRISM aims to improve the efficiency of continual adaptation, which may facilitate the deployment of adaptive machine learning systems on hardware with limited computational and memory resources. In addition, its analytical update procedures can provide a degree of interpretability and reproducibility that is desirable in safety-sensitive settings. Nevertheless, applications in high-risk domains should be accompanied by rigorous validation and monitoring, particularly when operating under distribution shifts or other forms of environmental change, to ensure dependable behavior.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

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 *ICCV*, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2021b.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991.

James Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

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

Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

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

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2007.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, et al. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, 2023.

Hai-Long Sun, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Pilot: A pre-trained model-based continual learning toolbox. *arXiv:2309.07117*, 2023.

Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *AAAI*, 2025.

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

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022a.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, 2022b.

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*, 2022c.

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 *CVPR*, 2022.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021.

Elif Ceren Gok Yildirim, Murat Onur Yildirim, and Joaquin Vanschoren. Pruned adaptation modules: A simple yet strong baseline for continual foundation models. *arXiv preprint:2603.21170*, 2026a.

Murat Onur Yildirim, Elif Ceren Gok Yildirim, and Joaquin Vanschoren. Unlocking [CLS] features for continual post-training. *Transactions on Machine Learning Research*, 2026b.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv:1910.04867*, 2019.

Yuanhan Zhang, Zhenfei Yin, Jing Shao, and Ziwei Liu. Benchmarking omni-vision representation through the lens of visual realms. In *ECCV*, 2022.

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. *IJCV*, 2024a.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024b.

A Appendix

A.1 Datasets, splits, and evaluation protocol

Datasets. Following the evaluation framework of foundation-model CIL (Wang et al., 2022c;b; Smith et al., 2023; Zhou et al., 2024a;b; Sun et al., 2025), we evaluate PRISM on six benchmarks: CIFAR-100 (Krizhevsky et al., 2009), CUB-200 (Wah et al., 2011), ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al., 2021b), OmniBenchmark (Zhang et al., 2022), and VTAB (Zhai et al., 2019). CIFAR-100 and CUB-200 provide standard classification settings, while ImageNet-R and ImageNet-A introduce substantial distribution shifts. OmniBenchmark and VTAB cover diverse visual domains. Following (Zhou et al., 2024a;b; Sun et al., 2025), we use the `B-`m `Inc-`n protocol, where m denotes the number of classes in the initial stage and n the number introduced at each subsequent stage.

Comparison Methods. We compare PRISM with state-of-the-art continual post-training methods in the same replay-free, foundation-model-based CIL setting: SimpleCIL (Zhou et al., 2024a), RanPAC (McDonnell et al., 2023), L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), CODAPrompt (Smith et al., 2023), APER-Adapter (Zhou et al., 2024a), EASE (Zhou et al., 2024b), MOS (Sun et al., 2025), and TOSCA (Yildirim et al., 2026b). MOS generates pseudo-replay from class statistics for classifier alignment, whereas the other methods do not store exemplars from previous tasks. All methods are reproduced using their released source code and evaluated with the same backbone and shuffled protocol.

Evaluation. Let $\mathcal{A}_t$ denote the Top-1 accuracy over all classes observed through stage t, and $\mathcal{A}_T$ the accuracy after the final stage. We report the final accuracy $\mathcal{A}_T$ and the average incremental accuracy $\bar{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{A}_t$. All metrics are computed under the shuffled inference protocol and averaged over five random seeds, with mean and standard deviation reported.

Implementation Details. All experiments use a ViT-B/16 backbone pre-trained on ImageNet-21K and are implemented in PyTorch (Paszke et al., 2019) using the Pilot toolkit (Sun et al., 2023). Following RanPAC, PRISM adapts the backbone only during the first session using AdaptMLP modules with bottleneck dimension 64, and keeps it frozen thereafter. The global router uses a fixed random projection to $D = 15{,}000$ dimensions followed by ReLU and a closed-form ridge classifier with $\lambda_R = 1000$; the projection matrix is sampled once and shared across all tasks. Each task-specific expert follows the TOSCA (Yildirim et al., 2026b) training recipe: SGD with batch size 48 for 20 epochs, cosine-annealed learning rate starting at 0.05, no weight decay, and an ℓ_1 sparsity coefficient of 5×10^{-4} on the module parameters. Every configuration is run with five random seeds, and we report the mean and standard deviation.

Table 3: This hyperparameters configuration is used for all six benchmarks.

Component	Hyperparameter	Value
First-session adaptation	AdaptMLP bottleneck r	64
	optimiser / lr / schedule	SGD (m= 0.9) / 0.01 / cosine
	epochs / batch size	20 / 48
	weight decay	5×10^{-4}
Task experts	bottleneck r_E	64
	ℓ_1 strength λ_T	5×10^{-4}
	optimiser / lr	SGD (m= 0.9) / 0.025
	epochs / batch size	20 / 48
Router	projection dimension D	15,000
	$\mathbf{P}$ distribution	$\mathcal{N}(0, 1/d)$
	nonlinearity σ	ReLU
	ridge λ_R	grid-selected per session
Protocol	seeds	$1993 - 1997$
	input resolution	224×224

A.2 Backbone and first-session adaptation

We use ViT-B/16 pretrained on ImageNet-21K, with patch size 16, hidden width $d = 768$, 12 transformer blocks and 12 attention heads. The pretrained weights are never updated. Adaptation capacity is provided by AdaptMLP modules inserted in parallel to the feed-forward sub-layer of *every* block. Each module is a bottleneck residual branch

$$\mathrm{AdaptMLP}(h) \;= \mathbf{W}_{\mathrm{up}}\,\sigma(\mathbf{W}_{\mathrm{down}}\,\mathrm{LN}(h))\,, \qquad \mathbf{W}_{\mathrm{down}} \in \mathbb{R}^{r\times d},\ \mathbf{W}_{\mathrm{up}} \in \mathbb{R}^{d\times r}, \tag{6}$$

with bottleneck width $r = 64$. Its output is added to the feed-forward output before the residual connection. $\mathbf{W}_{\mathrm{down}}$ uses Kaiming-uniform initialisation and $\mathbf{W}_{\mathrm{up}}$ is initialised to zero, so the adapted backbone is exactly the pretrained backbone at initialisation. Across 12 blocks this adds $\approx 1.2\%$ of the backbone.

These modules are trained on $\mathcal{T}_1$ only, jointly with a temporary linear head over $\mathcal{Y}_1$ that is discarded once the session ends, using cross-entropy. We use SGD with momentum 0.9, initial learning rate 0.01 with cosine decay, weight decay 5×10^{-4}, batch size 48, and 20 epochs. Training-time augmentation is random resized cropping to 224×224 and horizontal flipping; evaluation resizes the short side to 256 and centre-crops to 224. After session 1 the AdaptMLP parameters are frozen and Φ is fixed for the remainder of the sequence.

This is required because the routing statistics $(\mathbf{G}, \mathbf{C})$ operate and accumulate projected features $\mathbf{R}_s = \sigma(\Phi(\mathcal{D}_t)^\top \mathbf{P})$ computed at task t. Because previous data is discarded, these contributions cannot be recomputed later. Any post-hoc change to Φ would therefore leave $\mathbf{G}$ and $\mathbf{C}$ mixing features from different representations, and Proposition 1 would no longer hold. Freezing Φ after the first session is what makes the accumulated statistics a valid sufficient statistic for the joint problem.

A.3 Task experts

Each task t owns an adaptation module E_t operating on the frozen `[CLS]` representation and a linear head $\mathbf{W}_t \in \mathbb{R}^{d\times|\mathcal{Y}_t|}$ over that task's classes only. We use the module parameterisation of TOSCA (Yildirim et al., 2026b) unmodified; we summarise it here for completeness and refer to that work for the design rationale. E_t consist of two bottleneck residual blocks where one is responsible for adaptation and the second is responsible for calibration. Its training objective minimizes

$$\mathcal{L}_t = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \ell_{\mathrm{CE}}\big(\mathbf{W}_t^\top E_t(\Phi(x)), y\big) + \lambda_T \|\Theta_t\|_1, \tag{7}$$

where cross-entropy over $\mathcal{Y}_t$ with an ℓ_1 penalty on the module parameters at strength $\lambda_T = 5e^{-4}$, using SGD with momentum 0.9, learning rate 0.025 with cosine decay, batch size 48, and 20 epochs per session. Only $(E_t, \mathbf{W}_t)$ receive gradients; Φ, $\mathbf{P}$ and all previously trained experts are frozen. Because parameters are disjoint across tasks and no expert is revisited, expert training is independent of task order and of T.

A.4 Analytic router

Projection. $\mathbf{P} \in \mathbb{R}^{d\times D}$ has i.i.d. entries drawn from $\mathcal{N}(0, 1/d)$, sampled once from the run's seed before session 1 and never resampled. We use $D = 15{,}000$ throughout. The nonlinearity is $\sigma = \mathrm{ReLU}$, which combined with Gaussian $\mathbf{P}$ makes $r(x)$ a random-feature approximation to a first-order arc-cosine kernel (Rahimi & Recht, 2007). $\mathbf{P}$ can be generated identically with the fix random seed so requires no memory to store.

Solving. $\mathbf{W}_R = (\mathbf{G} + \lambda_R \mathbf{I})^{-1}\mathbf{C}$ is obtained by Cholesky factorisation of the regularised upper-triangular Gram matrix rather than by explicit inversion. λ_R is selected once per session over a logarithmic grid $\{10^{-8}, \ldots, 10^{8}\}$ on a held-out split of the current task's training data, choosing the value that maximises routing accuracy over all tasks seen so far. The factorisation is performed in float64 and $\mathbf{W}_R$ cast back to float32.

A.5 Training Algorithmic Flow and Inference

Here, we describe the complete PRISM procedure, covering both continual training and inference. Training separates shared representation adaptation, task-specific expert learning, and analytical router construction, while discarding each task's data after use. At inference, the learned router selects a single task expert, so prediction requires only one backbone pass and does not grow with the number of observed tasks.

Algorithm 1 PRISM — Continual Training

Require: pretrained backbone Φ_0; task sequence $\mathcal{T}_1, \ldots, \mathcal{T}_T$; projection dimension D; ridge grid Λ
Ensure: adapted backbone Φ, projection $\mathbf{P}$, router $\mathbf{W}_R$, experts $\{(E_t, \mathbf{W}_t)\}_{t=1}^{T}$

Initialisation ▷ before any task is observed
sample $\mathbf{P} \sim \mathcal{N}(0, 1/d)^{d \times D}$; freeze
$\mathbf{G} \leftarrow \mathbf{0}_{D \times D}$; $\mathbf{C} \leftarrow [\,]$ ▷ no task columns yet

for $t = 1$ **to** T **do**
(i) shared representation — first session only
  if $t = 1$ **then**
    insert AdaptMLP modules into every block of Φ_0
    train them with a temporary head on $\mathcal{D}_1$ by cross-entropy
    discard the temporary head; freeze Φ ▷ Φ fixed for all remaining sessions
  end if
(ii) task expert
  initialise $(E_t, \mathbf{W}_t)$
  train on $\mathcal{D}_t$ by minimising Eq. 7 ▷ Φ, $\mathbf{P}$, $\{E_s\}_{s<t}$ all frozen
  freeze $(E_t, \mathbf{W}_t)$
(iii) accumulate routing statistics
  for each minibatch $\mathcal{B} \subseteq \mathcal{D}_t$ **do** ▷ eval mode, no augmentation
    $\mathbf{R}_{\mathcal{B}} \leftarrow \sigma\big(\Phi(\mathcal{B})^\top \mathbf{P}\big)$
    $\mathbf{G} \leftarrow \mathbf{G} + \mathbf{R}_{\mathcal{B}}^\top \mathbf{R}_{\mathcal{B}}$
    $c_t \leftarrow c_t + \mathbf{R}_{\mathcal{B}}^\top \mathbf{1}$ ▷ every sample in $\mathcal{D}_t$ has task label t
  end for
  $\mathbf{C} \leftarrow [\,\mathbf{C} \mid c_t\,]$ ▷ append one column; existing columns unchanged
(iv) refit the router in closed form
  $\lambda_R \leftarrow \arg\max_{\lambda \in \Lambda}$ routing accuracy on held-out data
  $\mathbf{W}_R \leftarrow \mathrm{cholesky_solve}(\mathbf{G} + \lambda_R \mathbf{I},\ \mathbf{C})$ ▷ equals the joint solution, Prop. 1
  discard $\mathcal{D}_t$ ▷ no data is retained across sessions
end for

Training. PRISM processes the task sequence once, maintaining only the shared backbone, task-specific experts, and sufficient statistics for the router. The backbone is adapted on the first task and then frozen, while each subsequent task trains an independent expert without modifying previously learned parameters. After each task, its projected features are used to update the router's Gram matrix and label statistics; the corresponding data are then discarded. The router is refit from these accumulated sufficient statistics, so its solution incorporates all observed tasks without revisiting their data.

Inference. For a test input x, PRISM computes $z = \Phi(x)$, forms $r(x)$ by Equation 3, routes by Equation 2, and evaluates the single selected pair $(E_{\hat{t}}, \mathbf{W}_{\hat{t}})$:

$$\hat{y} \;=\; \arg\max_{c \in \mathcal{Y}_{\hat{t}}} \Big[E_{\hat{t}}(\Phi(x))^\top \mathbf{W}_{\hat{t}} \Big]_c , \qquad \hat{t} = \arg\max_t \big[r(x)^\top \mathbf{W}_R \big]_t . \tag{8}$$

The cost is one backbone forward pass, one projection, and one adaptation module, none of which depends on T; inference is therefore $\mathcal{O}(1)$ in the number of observed tasks, against $\mathcal{O}(T)$ for methods that infer task identity by evaluating candidates. Equation 8 depends on x alone, so predictions are unchanged under any permutation or recomposition of the evaluation batch.

A.6 Results and Discussions (cont.)

Table 4: Task-wise accuracy ($\mathcal{A}_t$) on ImageNet-A over five seeds with **ViT-B/16-IN21K**, reported as mean ± standard deviation. Here $\mathcal{A}_t$ is the top-1 accuracy over *all classes seen through stage* t (Tt), i.e. $20t$ classes, not the accuracy on task t's own classes. PRISM is highlighted; its average incremental accuracy, the highest of all methods, is shown in bold.

Method	Task										Avg
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	
SimpleCIL	77.13 ± 2.47	69.56 ± 1.55	64.44 ± 1.83	60.15 ± 2.08	57.23 ± 1.35	55.39 ± 1.57	53.60 ± 0.96	51.46 ± 1.14	49.37 ± 1.10	48.39 ± 0.00	58.67 ± 0.90
RanPAC	78.06 ± 2.88	71.34 ± 3.48	68.44 ± 2.21	65.56 ± 3.05	62.85 ± 1.70	61.21 ± 1.79	59.24 ± 0.94	57.21 ± 1.47	56.11 ± 0.65	54.73 ± 0.73	63.47 ± 1.45
L2P	64.83 ± 4.26	56.61 ± 2.53	53.21 ± 1.75	47.97 ± 1.75	45.62 ± 0.76	44.73 ± 0.60	43.60 ± 1.16	42.91 ± 1.37	41.49 ± 1.57	41.08 ± 0.96	48.21 ± 0.83
DualPrompt	72.30 ± 1.79	63.04 ± 2.93	59.30 ± 2.72	54.08 ± 2.34	50.49 ± 1.61	48.72 ± 1.79	46.43 ± 1.54	45.16 ± 1.33	43.43 ± 1.36	42.33 ± 0.77	52.53 ± 1.24
CODAPrompt	72.36 ± 3.29	63.66 ± 2.72	58.09 ± 0.99	52.98 ± 1.70	49.65 ± 0.41	47.64 ± 1.05	45.62 ± 0.77	43.75 ± 0.76	41.87 ± 1.02	41.12 ± 0.54	51.67 ± 0.49
APER	77.25 ± 2.58	69.54 ± 1.24	64.70 ± 1.77	60.48 ± 2.06	57.74 ± 1.40	55.93 ± 1.48	54.05 ± 1.01	51.90 ± 1.07	49.93 ± 1.02	48.94 ± 0.14	59.05 ± 0.91
EASE	76.48 ± 2.14	68.66 ± 1.81	64.46 ± 2.51	59.77 ± 2.52	56.74 ± 2.33	54.80 ± 1.45	52.18 ± 1.34	48.34 ± 1.36	46.36 ± 1.71	45.11 ± 1.62	57.29 ± 0.86
MOS	72.02 ± 5.74	70.18 ± 1.85	66.62 ± 2.18	63.20 ± 2.73	60.99 ± 1.76	59.59 ± 2.26	58.42 ± 1.78	56.68 ± 1.29	54.84 ± 1.45	53.89 ± 1.11	61.64 ± 1.80
TOSCA	73.65 ± 4.52	41.81 ± 3.36	30.06 ± 2.74	20.86 ± 2.52	18.16 ± 2.66	15.07 ± 1.61	13.68 ± 1.24	11.55 ± 1.51	10.05 ± 0.76	9.45 ± 1.43	24.43 ± 1.87
PRISM	79.44 ± 3.20	74.93 ± 3.23	71.45 ± 3.27	68.67 ± 2.87	67.07 ± 2.03	65.39 ± 1.87	64.02 ± 1.70	62.12 ± 1.07	60.05 ± 0.97	59.51 ± 0.72	**67.27** ± 1.79

Task-wise performance. Table 4 reports task-wise incremental accuracy on ImageNet-A under the B0 Inc20 protocol. PRISM achieves the highest average incremental accuracy, reaching $67.27\% \pm 1.79$, outperforming the strongest prior baseline, RanPAC, by 3.80 percentage points and MOS by 5.63 points. The advantage is consistent throughout the entire sequence: PRISM attains the best performance at every evaluation stage from T1 to T10. While all methods experience some degradation as the number of observed classes increases, PRISM exhibits the slowest decline, decreasing from 79.44% at T1 to 59.51% at T10, whereas RanPAC falls from 78.06% to 54.73% and most prompt-based methods deteriorate substantially more. Notably, the gap between PRISM and competing methods widens as the sequence progresses, indicating that its combination of task-specialized experts and analytic routing mitigates the accumulation of classifier error that typically emerges in long continual-learning streams.

Oracle routing exceeds the joint upper bound, and why that matters. The oracle attains 81.82% against a joint upper bound of 67.97%. This is not a contradiction: under correct routing each expert discriminates among the 20 classes of a single task, whereas the joint model discriminates among all 200. The routed formulation is not solving an easier problem overall — the router must recover task identity from the input alone, with no task label at test time — but it does factor the problem so that no single classifier faces the full label space. A method scored over all seen classes cannot exceed the joint oracle for that class set; doing so indicates that task identity has entered through the evaluation protocol. This is the concrete reason we evaluate all methods under instance-wise inference, and it applies to any batch-dependent selection rule rather than to TOSCA specifically.

Closing the gap to joint training. Averaged across the six benchmarks, PRISM ends 3.94 points below the joint oracle, against 5.45 for the best baseline in each data set, closing roughly 28% of the remaining gap. The result is sharpest on ImageNet-R, where PRISM reaches 81.34 against a joint upper bound of 81.92, i.e. within 0.58 points of an oracle that sees all 200 classes simultaneously, while the best prior method remains 3.06 points short. ImageNet-A remains the outstanding challenge for all methods, PRISM included, with an 8.46-point residual gap.