TRAJECTORY-PROBING CONTINUAL POST-TRAINING: IDENTIFYING AND CLOSING THE READOUT GAP

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning with foundation models is dominated by methods that adapt pre-trained representations per task via prompts, adapters, or low-rank updates, under the assumption that feature adaptation is the primary driver of performance. We revisit this assumption and find that the picture is inverted: as tasks accumulate, a linear joint probe fit on the continually adapted backbone climbs toward joint-training while the incrementally trained classifier accuracy falls. We call this discrepancy the *readout gap* — the share of linearly decodable information that the incremental classifier fails to recover. The gap survives a closed-form readout that is invariant to task order, so it is not reducible to classifier bias. Representations are therefore not the bottleneck, the decision layer is. This does not imply that adaptation is unnecessary, but that task-specific adaptation is incompatible with an accumulated readout: it perturbs the feature geometry the classifier depends on, whereas shared, task-agnostic calibration improves features without destabilizing it. Guided by this, we introduce Scale-and-Shift Knowledge Injection (SASKIA), which calibrates representations at two granularities that are both shared across tasks: a global scale-and-shift module aligning the backbone to the target domain, and an instance-level hypernetwork producing per-sample corrections, refined online by EMA-based model merging rather than task-specific growth. New classes are absorbed by closed-form ridge regression in this progressively refined space, requiring no rehearsal or explicit task boundaries. SASKIA attains the smallest readout gap among all compared methods and matches or outperforms them across benchmarks, using under 1% of the backbone's parameters.

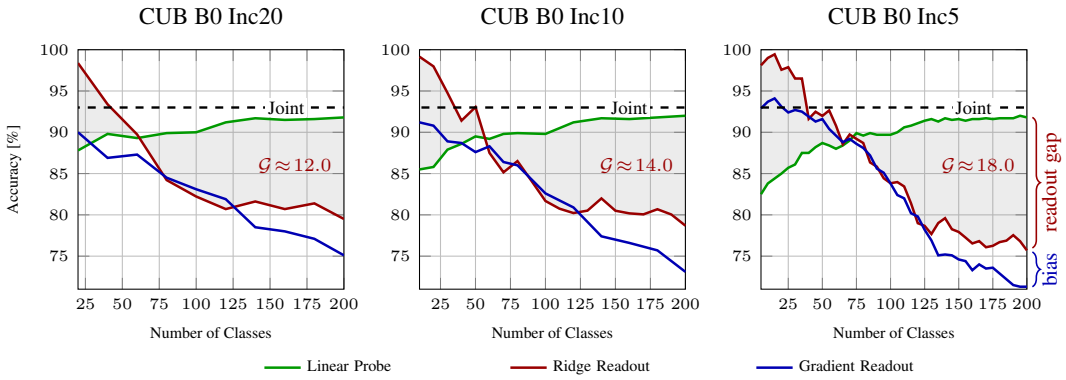


Figure 1: **The readout gap is the divergence between representation quality and continual retention.** Each panel tracks the same slowly adapting encoder under three readouts. The linear probe, refit from scratch at every stage on all data, measures what the current representation supports and rises toward joint training. The ridge readout and the gradient-fit readout are both accumulated across the sequence and both decline. Ridge is the informative one: its statistics are accumulated in a single pass and are invariant to task order, so on a fixed encoder it reproduces a jointly fitted classifier exactly and any gap beyond its frozen-encoder baseline is attributable to the moving geometry. The shaded band is therefore attributable to readout gap $\mathcal{G}$ accumulated in feature space, rather than to the under-representation of earlier classes that only biases a gradient-fit head; accounts for the narrower band below. Splitting the same data more finely widens the readout gap from 12 to 18 points while leaving probe accuracy nearly identical, so the gap scales with the number of times statistics are accumulated against a moving geometry.

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.

1 INTRODUCTION

Continual learning with foundation models has converged on *continual post-training*: freeze or lightly adapt powerful pre-trained representations and handle increments with prompts (Wang et al., 2022a;b; Smith et al., 2023), adapters (Zhou et al., 2024b; Sun et al., 2024; Yildirim et al., 2026a;b), or low-rank updates (Liang & Li, 2024; Wu et al., 2025). This inverts the classical setting, where representations were learned from scratch and catastrophic forgetting (McCloskey & Cohen, 1989) was treated as representational drift, mitigated by regularization (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Zenke et al., 2017), rehearsal (Rebuffi et al., 2017; Wu et al., 2019), or architectural isolation (Kang et al., 2023; Yildirim et al., 2025). Yet accuracy still degrades as tasks accumulate, and with the backbone largely fixed, weight interference can no longer explain it:

> **Research Question.**
> If representations are not heavily drifting, what actually degrades in continual post-training?

The readout gap. Section 3 answers this with a controlled experiment. Adapting the backbone slowly, we measure at every step both the continual model's accuracy and that of a linear classifier fit on the frozen features using the full dataset, giving an oracle upper bound on the encoder's linear separability over the complete label space. The probe rises monotonically toward joint-training accuracy while continual accuracy falls (Figure 1). We call the difference the *readout gap*: accuracy a linear map on the encoder achieves, but the incrementally trained classifier does not. Measured against a gradient-trained head, that difference can conflate two effects: (i) statistics that no longer describe the space they are read out in, and (ii) earlier classes appearing only as negatives once their task has passed. To separate them, we repeat the measurement with a closed-form ridge readout, whose statistics are obtained in a single pass and are invariant to task order, so on a fixed encoder it has no recency gap by construction. However, on the adapting encoder, it still exhibits a gap of 15 points, against 18 for the gradient-fit head. Class recency bias therefore accounts for only a small share of the deficit; the remainder is attributable to stale statistics induced by excessive representation shifts.

The gap tracks classifier updates. A second measurement confirms this. Splitting CUB into increments of 5, 10, or 20 classes varies the number of incremental classifier updates by a factor of four while holding the data and total training budget fixed. The linear probe reaches nearly identical accuracy in all three settings, while final continual accuracy degrades monotonically as the sequence is split more finely, widening the readout gap accordingly. The gap therefore cannot be attributed to differences in linearly decodable representation quality. This reveals a clear dissociation between representation quality and retention, in which an encoder improves to joint-training level over the very sequence. Crucially, the bias-free readout shows the same scaling where its gap widens from 12 to 18 points as the increments are refined. Therefore, what scales is the number of times statistics are accumulated against a geometry that has moved since they were computed.

Not less adaptation but a different adaptation. Two further observations show that how adaptation couples to the readout matters more than how much of it there is; we establish both in Section 3 and state here what they imply. First, on a frozen backbone, training only the classifier comes within a few points of prompt- and adapter-based methods, and a one-time adaptation paired with a closed-form readout outperforms all (Table 2, Figure 2 bottom). Second, replacing the linear head with closed-form ridge regression reduces forgetting by 7–16 points on *frozen identical features* (Table 1, Figure 2 top). On an adapting encoder, however, the same substitution recovers only 4 points, so a better readout is necessary but not sufficient: what remains is not the classifier's fault but the geometry's. The problem is thus not the amount of adaptation but how it couples to the readout, motivating a shared adaptation that refines the representation while preserving readout alignment.

Scale-and-Shift Knowledge Injection (SASKIA). We therefore build a method whose every adaptive component is task-agnostic. A global scale-and-shift module aligns the backbone to the target domain and is then frozen, while an instance-level hypernetwork supplies per-sample corrections and is refined by EMA merging, so the geometry evolves smoothly rather than in task-shaped jumps. This bounds inter-stage displacement and ensures smooth geometric evolution to preserve accumulated statistics, rather than guaranteeing strict feature invariance. New classes enter through single-pass closed-form ridge regression without rehearsal, task boundaries, or growing module sets. Shared across tasks, these modules add under 1% of backbone parameters while SASKIA yields the smallest readout gap and matches or outperforms competing methods across all benchmarks.

2 Related Work

Interrogating continual representations. Linear probes on frozen features reveal that continually learned representations retain far more information than end-task accuracy suggests (Davari et al., 2022), while simple classifiers over frozen ViT features can rival sophisticated continual learners (Janson et al., 2022). More broadly, Łapacz et al. (2025) show that the classification head can be a major source of the stability gap in continual learning, further separating representation quality from end-task performance. Prabhu et al. (2024) push this observation further, showing that fixed random representations can even outperform learned ones in online continual learning. Closest to our setting, Galashov et al. (2023) report modest gains from continual adaptation and attribute them to pretraining benchmark overlap. We ask a different question: not how representations compare at the *end* of training, but how they evolve *throughout* it. Probed representations improve monotonically toward joint-training performance even as end-task accuracy declines, revealing a dissociation that output accuracy alone cannot expose. This also refines the conclusion of Galashov et al. (2023): a small final benefit from adaptation does not imply that adaptation was unimportant, but can instead arise because improvements in the representation are masked by degradation in the classifier.

Stable feature geometry. One family keeps the feature space largely fixed once continual learning begins, whether by slow full-backbone updates with feature-statistic replay (Zhang et al., 2023), a frozen backbone with prototypical or analytic readout (Zhou et al., 2024a; Panos et al., 2023), or an additional random projection (McDonnell et al., 2024). They share the property our analysis identifies as scarce: accumulated classifier statistics stay valid, because the space they describe is the space in which later classes are read out. What they give up is tracking. A backbone fixed after the first task cannot align to later distributions.

Per-task adaptation. The largest family attaches task-specific parameters at every increment: prompts selected or composed from a pool (Wang et al., 2022b;a; Smith et al., 2023), adapters expanded, merged, or orthogonalized per task (Zhou et al., 2024b; Sun et al., 2024; Wang et al., 2025; Yildirim et al., 2026b), and low-rank updates constrained to limit interference (Liang & Li, 2024; Wu et al., 2025). This buys continual alignment at a cost our analysis makes precise: the geometry moves at every increment, so statistics accumulated earlier describe a space that no longer exists, and the classifier matches current features against a stale summary of classes it has already seen. Crucially, the interference these methods contain is *between task-specific modules*; the interference we identify is between the moving geometry and the classifier estimated over it, and module-level constraints do not address it.

Analytic classifiers. Closed-form incremental classification recasts class-incremental learning as recursive least squares for from-scratch CNN training (Zhuang et al., 2022; 2023; 2024a;b). RanPAC (McDonnell et al., 2024) carries the formulation into the foundation-model regime, and we adopt the same estimator. Our contribution is not the estimator but the pairing: a representation that keeps improving and a readout that stays aligned to it, which our diagnosis identifies as the condition neither frozen-backbone nor per-task-module methods satisfy.

3 Diagnosing the Readout Gap

We first formalize the continual learning setting and the readout gap, then present systematic experiments isolating the contributions of representation adaptation and classifier design. These findings create a basis for our approach.

3.1 Problem Formulation

Continual Learning. We study an image classification model exposed to a sequence of tasks $\{\mathcal{T}_t\}_{t=1}^{T}$, where each task is composed of a dataset $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$ with n_t instances from a disjoint set of classes Y_t, ensuring $Y_t \cap Y_{t'} = \emptyset$ for $t \neq t'$, i.e. non-overlapping classes for different datasets, while the underlying data distribution may evolve over time. The objective is to obtain a learner $f(\mathbf{x}) = W^T\Phi(x)$ with $\Phi : \mathcal{X} \to \mathbb{R}^d$ is the vision foundation model pretrained on a large-scale source dataset and $W \in \mathbb{R}^{d\times|\mathcal{Y}|}$ is the unified classifier whose predictions are made jointly over all classes $\mathcal{Y} = Y_1 \cup Y_2 \cup \cdots Y_t$. To summarize, our goal is to find a model $f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$ that minimizes empirical risk across all test datasets **in the absence of task indices**, i.e., in the class-incremental learning setting, **without accessing and storing past samples** (Wang et al., 2022b;a; Smith et al., 2023; Zhou et al., 2024b).

Measuring the readouts. We use linear probing to measure what the encoder can achieve, independently of how the continual classifier was trained. At each step t we freeze the current encoder Φ_t and fit a linear classifier from scratch over the full label space $\mathcal{Y}$. We denote its accuracy A_t^{LP}, and write A_t^{CL} for the continual model's accuracy at the same step. Probing over the full label space rather than the seen subset holds the probe's task at fixed difficulty, so that A_t^{LP} is comparable across steps and increment sizes *vs.* joint training; it also grants the probe exactly the information budget available to the continual model, since both observe the same data. We define the **readout gap** as

$$G_t \;=\; A_t^{\mathrm{LP}} - A_t^{\mathrm{CL}}, \tag{1}$$

the accuracy a linear readout attains on the encoder but the incrementally trained classifier does not. The composition of G_t depends on the readout against which it is measured. For gradient-based classifiers, a small portion of the gap may be attributable to the class recency/bias effect (Wu et al., 2019), which refers to the classifier disproportionately favoring more recently introduced classes. In contrast, for readouts derived from prototypes or other order-invariant statistics, which are commonly used in the literature, this particular source of bias is eliminated by construction. At the final step, both quantities are evaluated on the same label space and data distribution, allowing the deficit relative to joint training to be decomposed exactly as:

$$\underbrace{A^{\mathrm{Joint}} - A_T^{\mathrm{CL}}}_{\text{observed forgetting}} \;=\; \underbrace{\left(A^{\mathrm{Joint}} - A_T^{\mathrm{LP}}\right)}_{\text{representation deficit}} + \underbrace{G_T}_{\text{readout gap}} . \tag{2}$$

For $t < T$, the intermediate G_t is best interpreted as a trend indicator rather than an exact decomposition; this is also the regime in which negative readout gaps and readout parity naturally arise. Our main claims rely on the final-step decomposition in Eq. 2, the saturated regime of constant A_t^{LP}, and class-matched comparisons across increment sizes, ultimately measuring *how much representation potential the classifier leaves unrealized*.

3.2 Findings

> **Finding 1.**
> Continual accuracy declines while the achievable linear readout improves, and the resulting gap scales with the number of readout updates, not with the quality of the representation.

Setup. We adapt the foundation model across tasks at a small learning rate and, after each task, evaluate two quantities on the same encoder: the accuracy of a classifier trained sequentially, and the accuracy of a linear probe fit from scratch on the same features. This separates the quality of the learned representation from the quality of the continually maintained readout. To vary the number of readout updates independently of the data, we repeat the experiment on CUB with increments of 5, 10, and 20 classes, giving 40, 20, and 10 tasks over an identical dataset and training budget.

Results. Probe accuracy rises monotonically toward joint-training accuracy while continual accuracy declines (Figure 1). At the final step, where the comparison is exact, the representation deficit is negligible and the entire deficit to joint training is carried by the readout gap. The saturated regime is the sharpest evidence: beyond roughly 140 classes the probe is flat at joint level, so the representation term in Eq. 2 is approximately zero, yet continual accuracy falls a further 5–8 points. Every point lost there is lost in the readout. The increment-size comparison locates the source. Final probe accuracy is essentially identical across all three settings, at roughly 91 points, so the three encoders end up equally linearly separable and differences in final continual accuracy cannot be explained by one setting learning a better representation. Continual accuracy nonetheless degrades monotonically as the sequence is partitioned more finely, widening the gap from 12 to 18 points between Inc20 and Inc5. What varies is not what the representation supports but how many times the readout is re-estimated over it.

The early sequence requires separate interpretation. The gap is negative before the parity point because the continual model is scored on the classes seen so far while the probe is scored over the full label space, so the difference reflects a changing evaluation problem rather than an advantage of the continual model; in the finest-grained setting this can even place continual accuracy above the joint reference. The parity point itself shifts with increment size, since coarser increments expose the encoder to more classes in the first task and give the probe more data earlier. We therefore draw our conclusions from the post-parity regime.

Finding 2.
On identical frozen features, the estimator used for the readout changes forgetting drastically; closed-form second-order estimation is stable where repeated first-order optimization is not.

Setup. Having established that the gap grows with readout updates, we ask how much of the degradation the classifier alone can avoid. We freeze the foundation model entirely and compare four readouts under identical features and data: *Linear*, trained by stochastic gradient descent and updated sequentially; *Nearest Class Mean (NCM)*, classifying by cosine similarity to class prototypes with no gradient updates; *Linear Discriminant Analysis (LDA)*, estimating class means with a shared covariance; and *Ridge*, a closed-form solution from the feature Gram matrix. Removing representation adaptation as a source of variation isolates the readout's contribution.

Results. The readout alone changes forgetting by 7–16 points on identical frozen features (Table 1), consistent with prior analyses (Davari et al., 2022; Lanzillotta et al., 2026) and with evidence that classifier-level degradation dominates the stability gap (Łapacz et al., 2025). The effect holds across the entire sequence rather than at the final task alone (Figure 2, top). LDA and Ridge, which estimate decision boundaries from second-order statistics of the accumulated features, retain substantially more than the gradient-trained linear classifier, and Ridge is best throughout, with its largest margins on out-of-distribution data such as ImageNet-A. This complements McDonnell et al. (2024), which uses second-order statistics for its readout, by isolating the benefit of the readout under a fixed representation. The distinction that matters is not linear versus nonlinear but how a boundary is estimated from continually accumulated data: closed-form second-order estimation is markedly more stable than repeated first-order optimization.

Ridge is also what makes the diagnosis measurable. Because $\mathbf{G}$ and $\mathbf{C}$ are sums of per-sample outer products, they do not depend on the order in which tasks arrive: accumulating them incrementally over a fixed encoder would yield exactly the classifier a joint fit would produce, so a ridge readout on a frozen encoder has no *order-dependent* gap by construction. It does retain some residual gap relative to the probe, which reflects the difference between least-squares and probe estimation, together with λ selection. Because this residual is present whether or not the encoder moves, it serves as a baseline: the part of a ridge head's gap on an *adapting* encoder that exceeds this baseline is attributable to the geometry having moved, with no recency-bias term to subtract.

Finding 3.
The gap originates in the mismatch between when classifier statistics are computed and when they are used; adaptation improves the representation but degrades the continual model, under a readout with no bias to blame.

Setup. Finding 1 measured the gap against a gradient-trained head, which conflates two effects: statistics that no longer describe the space they are read out in, and earlier classes appearing only as negatives once their task has passed. Finding 2 supplies a readout free of the second. We therefore repeat the trajectory measurement of Finding 1 with ridge in place of the gradient head, holding the encoder schedule, the data, and the probe protocol fixed, and compare against the same readout on a frozen encoder.

Results. On a frozen encoder, ridge retains almost all of its accuracy across the sequence (Table 1). Accumulated over a slowly adapting encoder the same readout loses 8.5 points on CUB, ending at 77.6 against 86.1 frozen. The encoder is not the reason: over that same sequence the linear probe *rises* from 85.6 to 92.1, so the adapted representation supports strictly more than the frozen one. Adaptation improves what a linear map can extract while costing almost ten points of what the deployed model extracts, and no class-imbalance term is available to explain the loss.

This isolates the mechanism the earlier findings could only suggest. A ridge classifier accumulates $\mathbf{G}$ and $\mathbf{C}$ from features computed at the moment each task arrives; when the encoder subsequently moves, those statistics summarize a space the model no longer inhabits, and the solve returns boundaries fitted to a geometry that no longer exists. The gradient-trained head loses a further 4 points on top of this (Figure 1), which is the scale of the recency bias Finding 2 removes. Bias is real but secondary; what dominates is the mismatch between when classifier statistics are computed and when they are used.

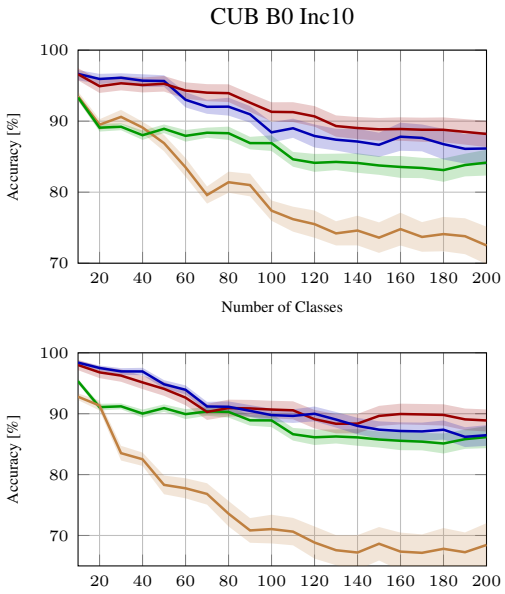


Table 1: Forgetting [↓] for different readouts.

Readout	CIFAR	CUB	ImageNet-A
Linear	11.27%	18.73%	24.66%
NCM	7.23%	9.51%	14.59%
LDA	6.08%	7.42%	11.41%
Ridge	**3.22%**	**4.30%**	**8.73%**

Table 2: Last accuracy $\mathcal{A}_T$ [↑] on different setups.

FM with	CIFAR	CUB	ImageNet-A
SimpleCIL	81.28%	86.73%	48.77%
CODAPrompt	81.46%	68.44%	45.73%
RanPAC	90.02%	89.27%	54.76%
EASE	85.80%	86.49%	55.24%
Joint	96.28%	92.81%	69.97%

Figure 2: **Analysis on ViT-B/16-IN21K. Top:** with features held fixed, readout choice alone changes forgetting by 7–16 points; closed-form estimation achieves the strongest retention, exposing the readout as a major source of degradation. **Bottom:** classifier-only training on a frozen backbone (SimpleCIL (Zhou et al., 2024a)) matches or outperforms continual prompt adaptation (CODAPrompt (Smith et al., 2023)). Furthermore, one-time random projection adaptation (RanPAC (McDonnell et al., 2024)) delivers higher performance compared to continually learned stacking adapters (EASE (Zhou et al., 2024b)), demonstrating that lightweight smooth feature adaptation paired with an optimal readout often matches or exceeds complex continual adaptation.

Finding 4.
Adaptation helps only insofar as it leaves the feature space a stable target for an accumulated readout; whether it is shared or task-specific matters more than how much of it there is.

Setup. We compare four training paradigms from a common frozen backbone. SimpleCIL (Zhou et al., 2024a) as *classifier-only adaptation*, CODAPrompt (Smith et al., 2023) as *prompt adaptation*, EASE (Zhou et al., 2024b) as *adapter adaptation*, and RanPAC (McDonnell et al., 2024) as *one-time feature adaptation*. The first three employ a prototype-based classifier (NCM), while RanPAC uses a second-order ridge classifier. Both readout families are inherently free from class-imbalance and recency biases, allowing the effect of feature adaptation to be isolated.

Results. Updating only the classifier on a frozen model matches or outperforms continuous prompt adaptation methods that learn to steer attention modules (Table 2, Figure 2 bottom). Adapters help more, though unevenly: EASE gains 4.5 and 6.5 points on CIFAR and ImageNet-A but is level on CUB, so adaptation is not unnecessary. RanPAC adapts far less, only once on the first task, yet outperforms EASE on CIFAR and CUB and is level on ImageNet-A. The distinction is therefore not how much the feature space is adapted but whether it remains a stable target for the accumulated readout. Per-task modules move the space at every step, and the accumulated statistics silently cease to describe the space in which later classes are read out; adaptation shared across tasks leaves the readout intact, while task-specific adaptation does not. RanPAC avoids the problem by freezing after the first task, and pays for it in alignment with later distributions. What remains open is whether a feature space can be refined continually while remaining a stable target for a closed-form readout.

Collectively, Findings 1–4 suggest that continual post-training is constrained less by representation adaptation than by maintaining a readout aligned with a moving representation. Representation quality improves even as accuracy collapses, and the gap grows with the number of updates (Finding 1). Readout choice alone changes forgetting by 7–16 points under identical features (Finding 2), while adaptation still incurs almost 10-point loss under a recency-free readout despite improving the probe (Finding 3). Feature adaptation therefore helps only when it preserves compatibility with the accumulated readout (Finding 4).

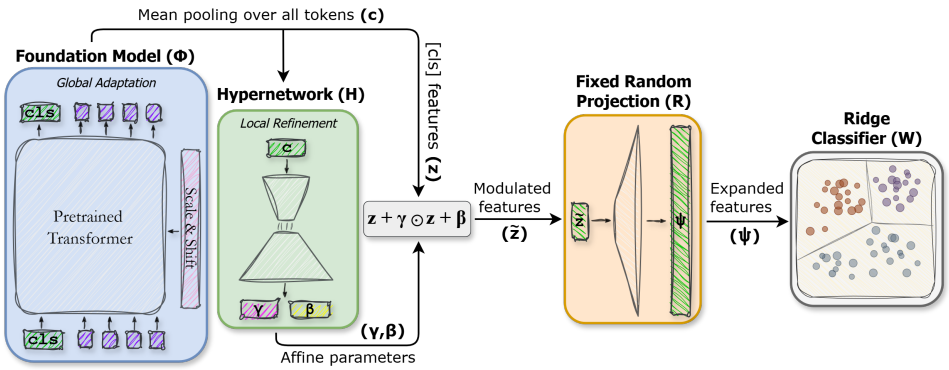


Figure 3: **SASKIA.** On the first session, the global module is trained jointly with an instance-level hypernetwork before being permanently frozen. Subsequent updates refine the hypernetwork via EMA merging, maintaining a smoothly evolving feature space and valid statistics. New classes enter through single-pass, closed-form ridge regression with zero replay or task-specific modules.

4 SASKIA

Our diagnosis gives the blueprint: rather than forcing the backbone to absorb each task, we build a globally calibrated feature space, refine it locally without letting any task dominate, and solve the remaining classification problem analytically. Given input $\mathbf{x}$, SASKIA computes

$$\mathbf{z} = [\Phi_{\mathcal{A}}(\mathbf{x})]_0, \quad (\boldsymbol{\gamma}, \boldsymbol{\beta}) = \mathcal{H}(\mathbf{c}), \quad \tilde{\mathbf{z}} = \big(1 + \tanh(\boldsymbol{\gamma})\big) \odot \mathbf{z} + \boldsymbol{\beta}, \quad \boldsymbol{\psi} = \mathrm{ReLU}(\mathcal{R}^\top \tilde{\mathbf{z}}), \tag{3}$$

where $\Phi_{\mathcal{A}}$ is a frozen foundation model carrying shared scale-and-shift modulation $\mathcal{A}$, $[\Phi_{\mathcal{A}}(\mathbf{x})]_0$ selects the `[CLS]` token, $\mathbf{c} = \frac{1}{N}\sum_n [\Phi_{\mathcal{A}}(\mathbf{x})]_n$ is the mean of final-layer tokens, $\mathcal{H}$ is a compact hypernetwork emitting per-sample modulation, $\mathcal{R} \in \mathbb{R}^{d \times M}$ is a fixed random projection, and $\odot$ is element-wise multiplication. Classes are read out from $\boldsymbol{\psi}$ by closed-form ridge regression. Every adaptive component is shared across tasks; none is task-conditioned. Please see our Appendix A.2 for a detailed algorithmic flow and mechanistic positioning diagram.

Global adaptation. After each layer normalization, attention block, and MLP block of the model we insert $\mathrm{SSF}_l(\mathbf{x}) = \mathbf{s}_l \odot \mathbf{x} + \mathbf{b}_l$ with $\mathbf{s}_l, \mathbf{b}_l \in \mathbb{R}^{d_l}$, giving $\mathcal{A} = \{(\mathbf{s}_l, \mathbf{b}_l)\}_l$. Backbone weights Φ stay frozen throughout; only $\mathcal{A}$ is optimized, during the initial adaptation phase on the first task, and frozen thereafter. Because $\mathcal{A}$ is shared and fixed, no task-specific routing is needed at inference.

Local refinement. Global alignment cannot correct per-sample deviations, so $\mathcal{H}$ maps the context $\mathbf{c}$ to $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ as in Eq. 3 and the $\tanh$ bounds the effective scale to $(0, 2)$, keeping $\tilde{\mathbf{z}}$ within the pretrained manifold's regime. To keep the space from being reshaped by whichever task came last, we hold an *active* copy for online training and a *shadow* copy for inference, merged after each task by $\mathcal{H}_{\text{shadow}} \leftarrow \alpha\mathcal{H}_{\text{shadow}} + (1-\alpha)\mathcal{H}_{\text{active}}$, with the active copy reset to the shadow weights at each task boundary. The representation geometry thus evolves smoothly instead of making abrupt, task-shaped jumps. Crucially, this merging process bounds inter-stage displacement to allow controlled adaptation and maintain the utility of accumulated ridge statistics, rather than guaranteeing or enforcing strict feature invariance.

Projection. Following McDonnell et al. (2024), $\mathcal{R} \in \mathbb{R}^{d \times M}$ has i.i.d. $\mathcal{N}(0, 1/d)$ entries drawn once and never updated, and $M \gg d$; the ReLU supplies the nonlinearity that makes the expanded space more linearly separable. Two properties matter here. The projection redistributes existing geometry rather than creating semantic structure, and because $\mathcal{R}$ is fixed, the projected space is identical at every task. This allows accumulated statistics always remain comparable.

Analytical readout. A stable deterministic feature map lets us discard iterative classifier training entirely. All information needed for the readout accumulates in two statistics, $\mathbf{G} = \sum_{t,i} \boldsymbol{\psi}_{t,i}\boldsymbol{\psi}_{t,i}^\top$ and $\mathbf{C} = \sum_{t,i} \boldsymbol{\psi}_{t,i}\mathbf{y}_{t,i}^\top$, with $\mathbf{y}$ one-hot and $\boldsymbol{\psi}$ computed under $\mathcal{H}_{\text{shadow}}$ after merging for task t while memory is $\mathcal{O}(M^2 + M|\mathcal{Y}|)$ independent from the number of samples seen. The ridge solution $W = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{C}$ uniquely minimizes $\|\mathbf{Y} - \Psi^\top W\|_F^2 + \lambda\|W\|_F^2$, with λ set by held-out grid search as in McDonnell et al. (2024), and the predictions are given by $\hat{y} = \arg\max_y \boldsymbol{\psi}_{\text{test}}^\top W_{:,y}$. Altogether, SASKIA turns continual learning into a sequence of analytical updates over a smoothly evolving feature space, with no replay, no task-specific modules, and no iterative optimization.

Table 3: **Main results under the 'offline → online' protocol.** Average accuracy $\bar{\mathcal{A}}$, last accuracy $\mathcal{A}_T$, and readout gap $\mathcal{G}_T$ with **ViT-B/16-IN21K**. $\mathcal{G}$ measures accuracy the representation supports but the incrementally trained classifier does not deliver, and remains comparable across methods despite their differing encoders. **Bold** marks the best result per column.

Method	CIFAR B0 Inc5			CUB B0 Inc10			IN-R B0 Inc20			IN-A B0 Inc20		
	$\bar{\mathcal{A}}\uparrow$	$\mathcal{A}_T\uparrow$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}\uparrow$	$\mathcal{A}_T\uparrow$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}\uparrow$	$\mathcal{A}_T\uparrow$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}\uparrow$	$\mathcal{A}_T\uparrow$	$\mathcal{G}\downarrow$
Joint	—	96.3±1.0	—	—	92.8±1.4	—	—	82.8±1.5	—	—	69.9±1.9	—
SimpleCIL	87.6±0.2	81.3±0.0	8.2±0.2	91.9±0.8	86.7±0.1	3.2±0.2	60.8±0.9	54.6±0.1	11.6±0.3	59.6±1.0	48.8±0.2	9.3±0.4
RanPAC	92.7±0.3	89.1±0.3	2.4±0.3	91.9±0.6	88.4±0.4	2.4±0.4	80.4±0.6	75.1±0.6	3.4±0.6	62.8±1.7	53.4±2.1	7.2±2.0
L2P	81.2±1.3	74.7±0.7	16.4±0.8	67.8±0.6	54.6±1.7	35.6±1.8	66.3±1.8	61.8±1.9	14.4±1.9	35.4±1.5	24.4±1.2	35.7±1.4
DualPrompt	82.9±0.6	76.6±0.4	15.1±0.6	76.5±2.8	63.6±2.1	26.7±2.2	65.8±1.2	60.1±1.6	17.0±1.6	46.2±3.5	34.2±2.1	26.4±2.3
CODAPrompt	84.4±0.3	76.8±0.6	15.2±0.7	68.5±1.3	58.7±1.4	31.5±1.5	67.9±1.5	63.4±1.6	14.6±1.7	49.0±2.2	37.1±1.6	24.3±1.8
EASE	88.8±1.0	82.8±1.0	9.3±1.0	90.4±1.4	85.6±0.5	5.3±0.6	74.2±0.8	68.6±0.5	9.4±0.6	55.7±1.1	45.2±1.1	15.6±1.3
MOS	88.8±0.5	85.0±0.5	7.4±0.6	89.5±0.8	83.8±0.2	7.3±0.4	74.1±1.3	69.4±0.8	10.0±0.9	57.7±1.7	49.5±1.6	12.5±1.8
TUNA	90.5±0.3	87.2±0.2	5.8±0.3	89.7±0.5	84.2±0.7	7.0±0.8	80.8±0.8	75.5±0.4	6.6±0.6	62.6±1.5	54.3±1.8	11.1±1.9
SASKIA	**93.9**±0.8	**90.2**±0.9	**2.1**±0.9	**93.6**±1.0	**89.8**±0.8	**1.6**±0.9	**84.0**±0.3	**79.0**±0.4	**2.2**±0.5	**67.2**±2.0	**58.8**±1.2	**3.6**±1.4

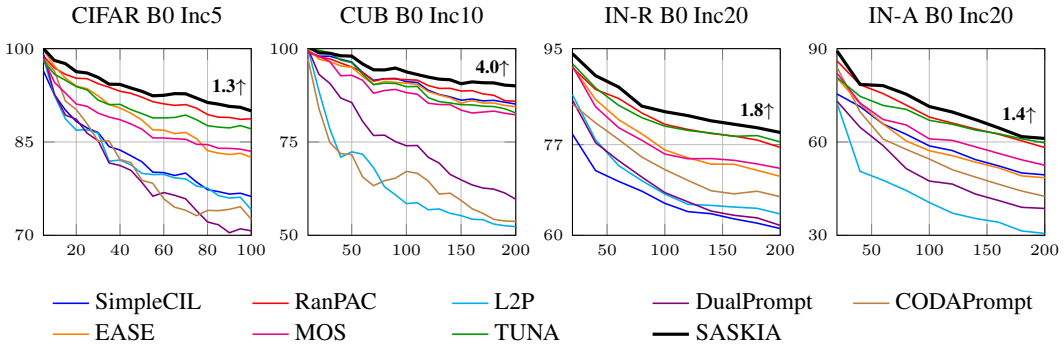


Figure 4: **Incremental accuracy under the 'offline → online' protocol, $\mathcal{A}_T$ at every increment.** Results use **ViT-B/16-IN1K**, a different pretraining corpus from Table 3, so the two together show the findings are not tied to one backbone. Each curve tracks accuracy over all classes seen so far as the sequence progresses; annotations give SASKIA's margin over the runner-up at the final increment. Two patterns hold across all four benchmarks: methods that re-optimize a readout at every task degrade fastest, while methods with closed-form or prototype readouts decline gradually despite adapting features far less. SASKIA leads throughout rather than only at the end, and its margin is widest on CUB, where fine-grained classes make the readout hardest to keep calibrated.

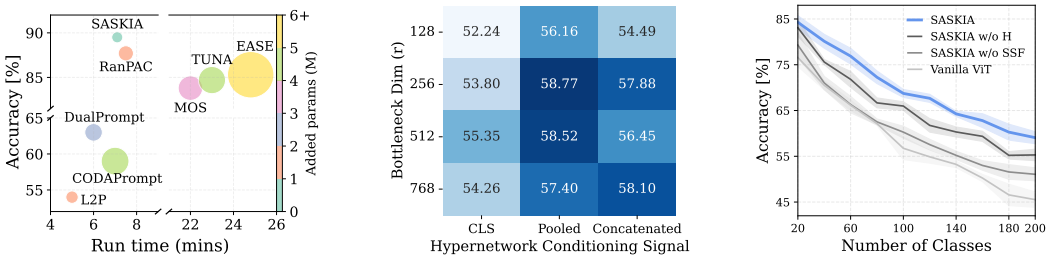
5 Empirical Evaluation

We report main results under the offline-to-online protocol as well as an always-offline budget to ensure baselines are evaluated in the setting they were designed for, and an ablation of SASKIA's components. Implementation details, further evaluation, and analysis are in Appendix A.1–A.4.

Main results. SASKIA leads on every benchmark and both metrics (Table 3), by 1.1–4.5% in last accuracy, and the same ordering holds on a second backbone (Figure 4). Methods separate by how they read out, not by how they adapt: closed-form readouts (RanPAC and SASKIA) keep $\mathcal{G}$ between 1.6–7.2%, while prompt-based methods that re-optimize classifier and selection mechanism at every task reach 24–36% on the fine-grained and out-of-distribution benchmarks. The comparison with TUNA is the sharpest case. Its final encoder supports a higher probe accuracy than ours on IN-R and IN-A, yet it finishes 3.6–4.4% behind, because its readout gap is about three times wider. Representation quality and retention come apart, and the readout decides which one the deployed model expresses. RanPAC is the instructive near-miss: restricting adaptation to the first task gives its ridge classifier a stable space and a small readout gap, but the space it stabilizes is never aligned to later distributions, which costs it 2–5% of probe accuracy relative to methods that keep adapting, largest on the OOD benchmarks. SASKIA achieves both, combining the smallest gap across all four benchmarks with a task-independent parameter footprint (Figure 5a) that adds under 1% of the backbone's parameters and maintains a fixed-capacity auxiliary state (Appendix A.4).

Table 4: **Comparison at offline budget.** Last accuracy $\mathcal{A}_T$ with **ViT-B/16-IN21K**. 'Always offline' runs each method under its released configuration, the setting these baselines were designed for. Δ is the cost of the 'offline $\rightarrow$ online' (o2o) protocol. Those that fit a classifier iteratively lose more in online scenario, while closed-form readouts lose under 2%. **Bold** marks the best result per column.

Method	CIFAR B0 Inc5			CUB B0 Inc10			IN-R B0 Inc20			IN-A B0 Inc20		
	offline	o2o	Δ	offline	o2o	Δ	offline	o2o	Δ	offline	o2o	Δ
SimpleCIL	81.28	81.28	0.00	86.73	86.73	0.00	54.55	54.55	0.00	48.77	48.77	0.00
RanPAC	90.02	89.12	−0.90	89.27	88.39	−0.88	75.44	75.07	−0.37	54.76	53.36	−1.40
L2P	80.06	74.70	−5.36	61.55	54.57	−6.98	69.77	61.81	−7.96	41.71	24.39	−17.32
DualPrompt	81.15	76.56	−4.59	70.51	63.64	−6.87	67.18	60.06	−7.12	41.67	34.21	−7.46
CODAPrompt	81.46	76.79	−4.67	68.44	58.73	−9.71	71.27	63.43	−7.84	45.73	37.04	−8.69
EASE	85.80	82.79	−3.01	86.49	85.60	−0.89	75.17	68.57	−6.60	55.24	45.21	−10.03
MOS	89.25	85.01	−4.24	90.07	83.75	−6.32	77.62	69.37	−8.25	56.90	49.49	−7.41
TUNA	90.74	87.18	−3.56	89.02	84.22	−4.80	79.42	75.45	−3.97	**59.78**	54.33	−5.45
SASKIA (ours)	**90.83**	**90.12**	−0.71	**90.35**	**89.77**	−0.58	**79.53**	**79.02**	−0.51	59.21	**58.77**	−0.44



(a) Parameter & compute cost (b) Hyperparameter analysis (c) Component ablation

Figure 5: **SASKIA from three perspectives. (a)** SASKIA's modules are shared across all tasks, so its footprint is fixed regardless of sequence length, unlike per-task expansion methods whose cost grows with the number of increments. **(b)** Mean-pooled tokens as the conditioning signal outperform `[CLS]`-only and concatenation at every bottleneck width r and performance is stable across a wide range of r. **(c)** Removing the instance-level hypernetwork costs accuracy consistently across the sequence, while removing the global SSF modulation costs more and does so from the earliest tasks, meaning global alignment must come first since local refinement can only correct a space that is already well conditioned.

Convergence budget. Underconvergence of the baselines designed for offline training may be a concern in the 'offline $\rightarrow$ online' protocol, so we also run every method to convergence on each task (Table 4). Prompt- and adapter-based methods optimize their adaptation parameters iteratively, so one might expect this setting to favour them twice: with better features and a better-fitted head. It does, and they still do not close the gap. Given their full budget, SASKIA leads on CIFAR, CUB and IN-R, trailing TUNA by 0.6% on IN-A alone. What separates the methods is instead how much they depend on that budget. TUNA loses 3.6% on CIFAR when each task is seen once, RanPAC 0.9%, and SASKIA 0.7%; across all four benchmarks, the two closed-form readouts stay within 1.5% of their offline results, while every gradient-fitted head loses between 3.6–17.3%.

Ablation. On IN-A B0 Inc20 (Figure 5b), conditioning $\mathcal{H}$ on mean-pooled tokens beats `[CLS]`-only and concatenation at every bottleneck width, with the best result at r=256 . Removing the hypernetwork $\mathcal{H}$ degrades accuracy across the sequence, and removing the global SSF modulation incurs greater cost and does so from the first tasks because local refinement can only correct a space that is already well-conditioned (Figure 5c). Please visit our Appendix A.5 for more ablation.

6 Conclusion

We asked what degrades in continual post-training since representations are no longer heavily drifting, and found that readout becomes the dominant bottleneck while the representations are reaching a linearly separable regime. The corrective is therefore applying an adaptation that the classifier can track: SASKIA pairs shared, task-agnostic calibration with a closed-form ridge readout, obtaining the smallest readout gap among all compared methods and matching or outperforming them across benchmarks, using under 1% of the backbone's parameters. Please visit our Appendix A.6 for the limitations and the future work.

AI USE STATEMENT

We used generative AI tools for writing assistance throughout this paper: condensing and rephrasing prose for length, drafting captions, and generating LaTeX formatting code for tables and plots. We did not use generative AI tools to generate research ideas, to produce experimental results, or to write the code used for training and evaluation. No AI-generated content was included without author review: all prose was read and edited by the authors, all numerical claims were checked against our own experimental logs, and all citations were verified against the sources. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

By reducing computational overhead, SASKIA may lower barriers to deploying adaptive systems in resource-constrained environments like edge devices. Its closed-form classifier updates offer transparency and predictability beneficial for safety-critical applications. However, high-stakes deployments require careful evaluation to ensure reliability under distribution shifts.

REPRODUCIBILITY STATEMENT

We take several measures to facilitate the reproducibility and reuse of SASKIA. An anonymous version of the complete source code is provided for review and will be made publicly available upon acceptance. The code includes detailed documentation and a user guide covering installation, configuration, and execution of SASKIA and the experimental pipeline. The proposed method is described in detail in Appendix A.2 (Algorithm 1), while the complete implementation and hyperparameters are provided in Appendix A.1 (Implementation Details). We evaluate SASKIA on four publicly available benchmark datasets, CIFAR100, CUB200, ImageNet-R, and ImageNet-A, with the corresponding experimental configurations and evaluation protocols provided in the appendix. All reported results are averaged over five random seeds, with mean and standard deviation reported where applicable. Experiments are conducted on NVIDIA A100 GPUs, and additional details necessary to reproduce the reported results are provided in the appendix and supplementary materials.

REFERENCES

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *CVPR*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Alexandre Galashov, Jovana Mitrovic, Dhruva Tirumala, Yee Whye Teh, Timothy Nguyen, Arslan Chaudhry, and Razvan Pascanu. Continually learning representations at scale. In *CoLLAs*, 2023.

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*.

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

Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. *arXiv preprint arXiv:2210.04428*, 2022.

Haeyong Kang, Jaehong Yoon, Sultan Rizky Hikmawan Madjid, Sung Ju Hwang, and Chang D. Yoo. On the soft-subnetwork for few-shot class incremental learning. In *ICLR*, 2023.

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.

Giulia Lanzillotta, Damiano Meier, and Thomas Hofmann. Heads collapse, features stay: Why replay needs big buffers. In *ICLR*, 2026.

Wojciech Łapacz, Daniel Marczak, Filip Szatkowski, and Tomasz Trzciński. Exploring the stability gap in continual learning: The role of the classification head. In *WACV*, 2025.

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

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *CVPR*, 2024.

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*, 2024.

Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E. Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *ICCV*, 2023.

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

Ameya Prabhu, Shiven Sinha, Ponnurangam Kumaraguru, Philip Torr, Ozan Sener, and Puneet K. Dokania. Random representations outperform online continually learned representations. In *NeurIPS*, 2024.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 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*, 2024.

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

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *ICCV*, 2025.

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

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-loRA: Scalable decoupled low-rank adaptation for class incremental learning. In *ICLR*, 2025.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

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, Decebal Constantin Mocanu, and Joaquin Vanschoren. Self-regulated neurogenesis for online data-incremental learning. *arXiv preprint:2403.14684*, 2025.

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.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICCV*, 2023.

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.

Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. ACIL: Analytic class-incremental learning with absolute memorization and privacy protection. In *NeurIPS*, 2022.

Huiping Zhuang, Zhenyu Weng, Run He, Zhiping Lin, and Ziqian Zeng. Gkeal: Gaussian kernel embedded analytic learning for few-shot class incremental task. In *CVPR*, 2023.

Huiping Zhuang, Run He, Kai Tong, Ziqian Zeng, Cen Chen, and Zhiping Lin. Ds-al: A dual-stream analytic learning for exemplar-free class-incremental learning. In *AAAI*, 2024a.

Huiping Zhuang, Yuchen Liu, Run He, Kai Tong, Ziqian Zeng, Cen Chen, Yi Wang, and Lap-Pui Chau. F-oal: Forward-only online analytic learning with fast training and low memory footprint in class incremental learning. *NeurIPS*, 2024b.

A Appendix

This appendix complements the main paper in five parts. We first give the experimental setup: datasets, protocols, metrics, and the hyperparameters needed to reproduce our results. We then describe each compared method and give full pseudocode for SASKIA. The third part reports additional results: the second backbone under the offline-to-online protocol, an always-online protocol that removes the offline stage entirely, a large-base-class regime, and the linear-probe accuracies underlying the readout gap. The fourth accounts for memory and documents the numerical choices behind our analytic readout. The last reports sensitivity to the two hyperparameters SASKIA introduces and states the limitations of our conclusions.

A.1 Experimental Setup

Datasets. Since foundation models often carry substantial knowledge of upstream tasks, we adopt the evaluation framework of (Wang et al., 2022b;a; Smith et al., 2023; Zhou et al., 2024b; Sun et al., 2024; Wang et al., 2025; Yildirim et al., 2026b) and evaluate across benchmarks that differ in their relation to the pretraining distribution: CIFAR-100 (Krizhevsky et al., 2009), CUB-200 (Wah et al., 2011), ImageNet-R (Hendrycks et al.), and ImageNet-A (Hendrycks et al., 2021). The latter two exhibit significant domain shift relative to ImageNet (Deng et al., 2009), on which both backbones are pretrained. Following (Zhou et al., 2024b; Sun et al., 2024; Wang et al., 2025; Yildirim et al., 2026b; Zhou et al., 2024a) we write 'B-m Inc-n' for a split whose initial stage contains m classes (when it differs from later stages) and which introduces n classes per incremental stage.

Protocols. Our primary setting is the *offline-to-online* protocol: the first task is trained offline to convergence, and every subsequent task is seen in a single pass with no revisiting. This matches with the real-world deployment, where a model is adapted once on available data and thereafter encounters new classes as a stream. We also report an *always-online* protocol (Section A.3.2), which removes the offline stage entirely, and an *always-offline* setting in the main paper, which trains every method to convergence on every task.

Comparison methods. We compare against SimpleCIL (Zhou et al., 2024a), RanPAC (McDonnell et al., 2024), L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODAPrompt (Smith et al., 2023), EASE (Zhou et al., 2024b), MOS (Sun et al., 2024) and TUNA (Wang et al., 2025), all run under the foundation-model-based rehearsal-free CIL setting from their authors' released source code. We include 'Joint', which trains on all classes simultaneously, as an upper-bound reference.

Evaluation metrics. We use the standard continual learning metrics (Rebuffi et al., 2017). Writing $\mathcal{A}_t$ for Top-1 accuracy over all classes seen after stage t, we report the final-stage accuracy $\mathcal{A}_T$ and the average across stages $\bar{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{A}_t$. We additionally report the readout gap $\mathcal{G} = \mathcal{P}_T - \mathcal{A}_T$, where $\mathcal{P}_T$ is a linear probe fit from scratch on each method's own final encoder using all data seen up to that point (Section A.3.4).

Implementation details. All experiments run under identical conditions on an NVIDIA A100, reproduced with PyTorch (Paszke et al., 2019) and PILOT (Sun et al., 2023). Following (Zhou et al., 2024b; Sun et al., 2024; Wang et al., 2025; Yildirim et al., 2026b) we use two foundation models, ViT-B/16-IN21K and ViT-B/16-IN1K; both are pretrained on ImageNet21K, the latter further fine-tuned on ImageNet1K. For SASKIA, the SSF modulation and the hypernetwork are trained jointly during the offline phase with SGD (momentum 0.9, weight decay 5×10^{-4}), a cosine schedule from an initial learning rate of 10^{-2}, batch size 48, for 25 epochs. In the online phase the active hypernetwork is fine-tuned with the same optimizer for a single pass at batch size 10. We set the hypernetwork bottleneck to r=256, the projection dimension to M=10,000, the EMA decay to α=0.95, and select the ridge parameter λ by held-out grid search over $\{10^{-8}, \ldots, 10^{8}\}$. All results are reported as mean and standard deviation over five random seeds.

A.2 Compared Methods and SASKIA

Table A1: Conceptual comparison of continual post-training paradigms.

Method	Readout Compatibility	Representation Adaptation
SimpleCIL	High	None
RanPAC	High	One-time
Per-task modules	Low	Continual
SASKIA	High	Continual

SimpleCIL (Zhou et al., 2024a) uses the pretrained backbone unchanged with a prototypical classifier, constructing one prototype per class and classifying by cosine similarity. Because the prototypes are computed in a single pass over a fixed feature space, its readout is invariant to the training protocol, a property we use as a control throughout.

RanPAC (McDonnell et al., 2024) extends this by inserting layer-wise adapters trained only on the initial task, projecting the adapted features into a high-dimensional random space, and fitting an analytic ridge classifier.

L2P (Wang et al., 2022b) integrates visual prompts into class-incremental learning, placing prompts only at the initial embedding layer so that adaptation occurs early while the backbone stays frozen.

DualPrompt (Wang et al., 2022a) separates general prompts (G-Prompt), applied to earlier blocks for broad adaptation, from expert prompts (E-Prompt) in later blocks for task-specific tuning.

CODAPrompt (Smith et al., 2023) addresses instance-specific prompt selection through an attention mechanism that dynamically reweights prompts.

EASE (Zhou et al., 2024b) adds adapters at every layer for every task and concatenates the resulting task-specific feature representations, which increases model complexity in proportion to the sequence length.

MOS (Sun et al., 2024) also trains per-task adapters at every layer, adding adapter merging and replay generation for classifier correction. Both increase computational cost during training as the number of task-specific modules grows.

TUNA (Wang et al., 2025) learns task-specific orthogonal adapters alongside shared universal ones, a two-level scheme disentangling task-specific from shared representations, with rehearsal-based classifier alignment to mitigate interference.

SASKIA (ours) calibrates representations at two granularities through scale-and-shift modulation. Global SSF modulation aligns internal backbone distributions during offline training and is then permanently frozen, while an instance-level hypernetwork generates per-sample corrections to the output `[CLS]` embedding and is refined online by EMA merging. New classes are incorporated by single-pass analytic ridge regression without rehearsal or stored samples.

Algorithm 1 SASKIA: Scale-and-Shift Knowledge Injection for Online Continual Post-Training

Require: Tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, frozen backbone Φ, fixed projection $\mathcal{R}$, EMA rate α, ridge parameter λ

Initialize SSF modulation $\mathcal{A}$, active hypernetwork $\mathcal{H}_{\text{active}}$, $\mathbf{G} \leftarrow \mathbf{0}$, $\mathbf{C} \leftarrow \mathbf{0}$
// Offline phase (task 1)
Train $\mathcal{A}$ and $\mathcal{H}_{\text{active}}$ jointly on $\mathcal{D}_1$ for multiple epochs, using a cross-entropy loss through a temporary linear head that is discarded afterwards
Freeze $\mathcal{A}$ permanently; $\mathcal{H}_{\text{shadow}} \leftarrow \mathcal{H}_{\text{active}}$
Accumulate $\mathbf{G}, \mathbf{C}$ over $\mathcal{D}_1$ using $\mathcal{H}_{\text{shadow}}$
Solve $(\mathbf{G} + \lambda \mathbf{I})\, W = \mathbf{C}$ for W (Cholesky; see Section A.4.2)
// Online phase
for $t = 2, \ldots, T$ **do**
  Fine-tune $\mathcal{H}_{\text{active}}$ on $\mathcal{D}_t$ in a single pass with the same temporary-head objective
  $\mathcal{H}_{\text{shadow}} \leftarrow \alpha\, \mathcal{H}_{\text{shadow}} + (1 - \alpha)\, \mathcal{H}_{\text{active}}$
  Accumulate $\mathbf{G}, \mathbf{C}$ over $\mathcal{D}_t$ using $\mathcal{H}_{\text{shadow}}$
  Solve $(\mathbf{G} + \lambda \mathbf{I})\, W = \mathbf{C}$ for W
  $\mathcal{H}_{\text{active}} \leftarrow \mathcal{H}_{\text{shadow}}$
end for
// Inference
$\hat{y} = \arg\max_y \; \boldsymbol{\psi}_{\text{test}}^{\top} W_{:,y}$

Table A2: **Offline-to-online protocol with ViT-B/16-IN1K.** Average accuracy $\bar{\mathcal{A}}$, last accuracy $\mathcal{A}_T$, and readout gap $\mathcal{G}$, mean ± std over five seeds. 'IN-R/A' abbreviates ImageNet-R/A. **Bold** marks the best per column.

Method	CIFAR B0 Inc5			CUB B0 Inc10			IN-R B0 Inc20			IN-A B0 Inc20		
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$
Joint	—	95.9±1.0	—	—	90.2±1.4	—	—	83.9±1.5	—	—	74.1±1.9	—
SimpleCIL	82.8±0.2	76.2±0.0	8.8±0.2	91.0±0.9	85.2±0.1	3.2±0.2	67.1±0.9	61.3±0.2	10.6±0.3	59.7±1.0	49.4±0.1	9.2±0.3
RanPAC	90.6±0.1	88.7±0.1	2.6±0.2	91.9±0.5	85.6±3.2	2.6±1.0	81.6±0.4	76.6±0.1	3.4±0.3	67.4±1.6	58.2±0.6	4.5±1.1
L2P	80.0±1.9	73.5±3.4	16.2±3.5	64.7±1.0	51.9±0.5	35.5±0.8	69.8±1.7	63.7±1.7	14.4±1.8	41.5±2.3	30.2±0.7	31.1±1.0
DualPrompt	77.6±3.1	70.4±0.3	19.8±0.5	72.3±2.7	57.7±3.4	30.2±3.5	69.5±0.8	61.9±0.4	17.0±0.6	48.3±1.8	36.5±2.0	25.3±2.3
CODAPrompt	81.9±0.7	73.9±1.0	17.0±1.1	64.7±1.2	53.4±0.8	34.4±0.9	73.0±0.3	67.5±0.5	12.5±0.8	55.8±0.2	42.2±0.3	20.5±0.9
EASE	88.4±0.6	81.9±0.8	8.7±0.9	90.0±0.9	84.2±0.3	5.0±0.5	77.8±0.5	71.0±0.1	8.3±0.4	58.6±0.8	48.5±0.1	12.9±0.7
MOS	88.0±1.3	84.0±1.1	7.0±1.2	87.7±1.5	81.8±1.0	7.7±1.1	76.4±0.5	71.2±0.5	9.3±0.7	61.7±0.9	52.5±2.0	10.2±2.1
TUNA	90.6±0.3	87.0±0.2	4.8±0.3	88.5±1.4	82.2±0.6	7.6±0.7	82.3±0.5	77.5±0.1	5.6±0.3	67.5±1.6	59.7±0.2	6.5±0.7
SASKIA	**94.0**±0.7	**90.0**±0.9	**1.6**±1.0	**93.1**±0.9	**89.5**±0.2	**0.4**±0.4	**84.0**±0.4	**79.2**±0.2	**2.7**±0.3	**68.8**±1.3	**61.1**±1.2	**2.7**±1.4

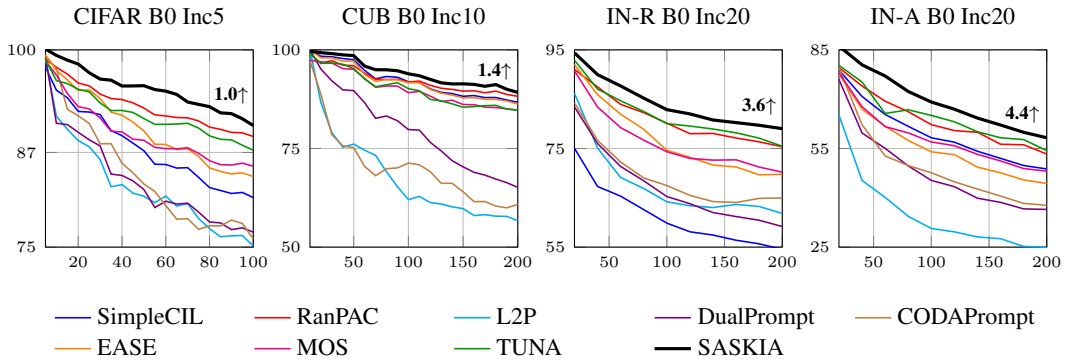


Figure A1: **Offline-to-online protocol with ViT-B/16-IN21K.** Last accuracy $\mathcal{A}_T$ after each stage. Annotations give SASKIA's margin over the runner-up at the final stage. SASKIA leads throughout rather than only at the end, and its margin is widest on the two out-of-distribution benchmarks.

A.3 Additional Results

This section reports results along three axes: the second foundation model under our primary protocol, two additional protocols that vary how much optimization each task receives, and the linear probe accuracies underlying the readout gap. Across all of them the ordering of methods is stable and the gap continues to track how a method reads out rather than how much it adapts.

A.3.1 Offline-to-online protocol

Figure A1 shows incremental accuracy curves under the offline-to-online protocol with ViT-B/16-IN21K, complementing Table 3 in the main paper; Table A2 reports the same protocol with ViT-B/16-IN1K, complementing the curves in Figure 4. SASKIA holds the highest accuracy at every stage on all four benchmarks, not only at the end. The margin is largest on the out-of-distribution benchmarks, where the gap between the pretraining and target distributions is widest, the regime in which the feature space is hardest to keep aligned, and in which methods that re-adapt representations per task inject variance into the space at every increment. SASKIA's one-time global alignment followed by EMA-refined local correction avoids this compounding. Prompt-based methods decline most steeply in later stages, more so than adapter-based ones. This is consistent with prompts providing too little representational capacity under a single-pass constraint: each task receives far less optimization signal than in the multi-epoch setting these methods were designed for, and the main paper's always-offline comparison confirms that they recover substantially when that budget is restored. Results agree across both pretraining regimes. This is worth noting because IN1K models have already been fine-tuned toward the ImageNet distribution, which should in principle reduce the benefit of SASKIA's global alignment. That the advantage persists suggests the instance-level hypernetwork contributes beyond what global alignment alone provides.

Table A3: **Always-online protocol with ViT-B/16-IN1K.** Every task, including the first, is processed in a single pass. Directly comparable to Table A2, which differs only in having an offline first task. SimpleCIL is identical in both, as neither its encoder nor its readout can use the extra optimization. **Bold** marks the best.

Method	CIFAR B0 Inc5			CUB B0 Inc10			IN-R B0 Inc20			IN-A B0 Inc20		
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$
Joint	—	95.9±1.0	—	—	90.2±1.4	—	—	83.9±1.5	—	—	74.1±1.9	—
SimpleCIL	82.8±0.2	76.2±0.0	8.8±0.2	91.0±0.9	85.2±0.1	3.2±0.2	67.1±0.9	61.3±0.2	10.6±0.3	59.7±1.0	49.4±0.1	9.2±0.3
RanPAC	91.0±0.4	86.7±0.3	3.0±0.4	91.7±0.5	85.8±3.9	2.0±1.9	78.0±0.7	72.7±0.5	3.4±0.6	65.7±0.6	56.8±0.1	4.0±0.5
L2P	80.8±1.6	74.7±1.0	14.2±1.2	65.4±1.0	51.3±1.9	35.4±2.0	70.3±1.0	66.1±0.5	11.1±0.9	39.9±0.5	31.4±2.3	29.0±2.4
DualPrompt	78.1±1.6	70.3±0.6	19.1±0.8	71.0±1.2	57.9±1.6	29.2±1.7	69.3±0.3	62.9±0.1	14.8±0.5	46.2±3.0	35.9±1.7	25.0±2.0
CODAPrompt	80.5±1.2	71.7±1.4	18.3±1.5	60.4±1.0	50.2±1.1	36.7±1.3	70.9±1.1	67.0±0.7	11.4±0.9	50.3±0.9	38.0±1.2	23.2±1.4
EASE	87.6±0.9	81.7±0.5	8.4±0.7	89.6±1.5	84.4±0.3	4.5±0.5	76.8±0.9	70.6±0.6	8.0±0.8	59.5±0.5	49.6±0.3	11.5±0.6
MOS	83.4±5.5	78.6±6.0	11.1±6.1	86.6±1.7	81.3±0.8	7.5±1.0	73.6±5.2	68.4±4.2	9.7±4.3	60.8±4.4	52.8±4.5	9.0±4.6
TUNA	88.5±0.7	85.3±0.4	5.5±0.5	86.7±1.4	81.8±0.5	7.3±0.7	79.1±0.7	75.5±0.3	6.1±0.5	64.0±1.6	56.0±0.8	8.3±1.0
SASKIA	**93.8**±0.7	**90.0**±0.6	**1.4**±0.7	**93.3**±0.6	**89.3**±0.6	**0.3**±0.7	**81.4**±0.6	**77.0**±0.4	**2.8**±0.6	**66.1**±2.9	**58.4**±1.8	**3.2**±2.0

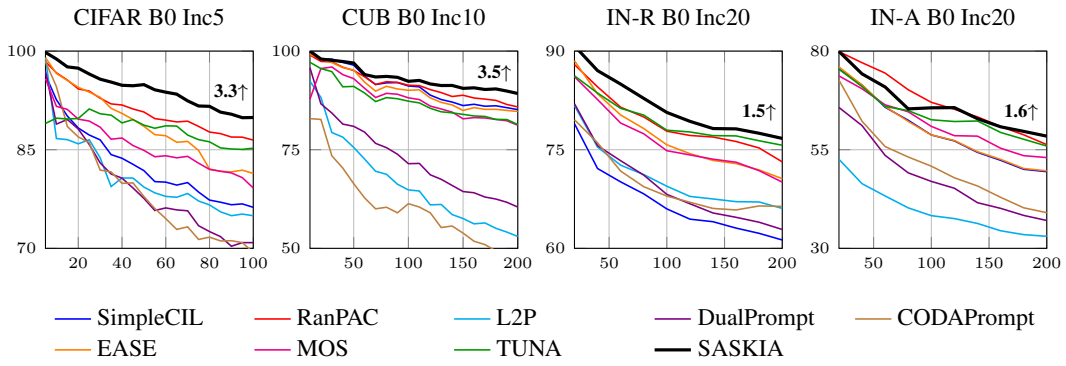


Figure A2: **Always-online protocol with ViT-B/16-IN1K.** Last accuracy $\mathcal{A}_T$ after each stage, with no offline phase at any point. Annotations give SASKIA's margin over the runner-up at the final stage.

A.3.2 Always-online protocol

To test all methods without any offline adaptation, we evaluate an always-online protocol in which every task, including the first, is processed under strict single-pass constraints. Table A3 and Figure A2 report results with ViT-B/16-IN1K, directly comparable to Table A2 above. SASKIA retains the highest final accuracy on all four benchmarks, and the ordering of methods is essentially unchanged, therefore hierarchy reflects algorithmic differences rather than access to an offline stage. The pattern of losses relative to Table A2 is more informative than the ranking. SimpleCIL is exactly invariant, as its readout cannot use the extra optimization.

The methods that lose most are those that accumulated readout statistics during the offline phase: RanPAC (-3.91 on IN-R), TUNA (-3.73 on IN-A), MOS (-5.38 on CIFAR) and CODAPrompt (-4.26 on IN-A). The prompt-based methods L2P and DualPrompt are nearly unaffected, and occasionally improve, because they were already failing to converge under the offline-to-online protocol—removing the offline phase takes away little they were using. EASE is likewise stable, changing around a point on every benchmark. SASKIA's own losses are concentrated where the offline phase was doing real work: under 0.25 points on CIFAR and CUB, but 2.28 on IN-R and 2.69 on IN-A. This is the same asymmetry noted in our limitations—the offline stage matters most where pretrained features transfer least well—and it indicates that the global SSF modulation, though small enough to converge meaningfully even in a single pass, contributes more on out-of-distribution targets. The instance-level hypernetwork compensates for the remainder, accumulating stable knowledge from the first task onward through EMA merging regardless of whether a dedicated offline phase was available.

Table A4: **Large base class (B50).** Last accuracy $\mathcal{A}_T$ and readout gap $\mathcal{G}$ at the final stage. SimpleCIL's probe is unchanged from B0, since its encoder is frozen and the probe eventually sees the same data; its gap therefore also matches B0 and serves as a reference point across protocols.

Method	ViT-B/16-IN21K				ViT-B/16-IN1K			
	IN-R B50 Inc10		IN-A B50 Inc10		IN-R B50 Inc10		IN-A B50 Inc10	
	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$
SimpleCIL	54.6	11.6	48.8	9.3	61.3	10.6	49.4	9.2
RanPAC	78.6	3.1	59.4	4.4	79.3	3.4	61.0	6.6
L2P	49.2	19.0	28.9	25.7	54.3	18.6	29.7	26.1
DualPrompt	47.8	19.7	27.8	26.1	55.5	18.1	31.5	25.4
CODAPrompt	55.8	15.1	35.9	20.5	61.0	14.8	40.1	18.5
EASE	60.6	13.7	45.0	14.7	67.6	11.6	49.2	12.9
MOS	74.7	7.4	55.1	9.1	75.8	7.2	58.5	8.8
TUNA	77.6	5.8	59.3	7.6	78.3	5.9	60.0	8.2
SASKIA (ours)	**79.8**	**2.5**	**61.3**	**3.8**	**80.5**	**2.7**	**61.7**	**5.3**

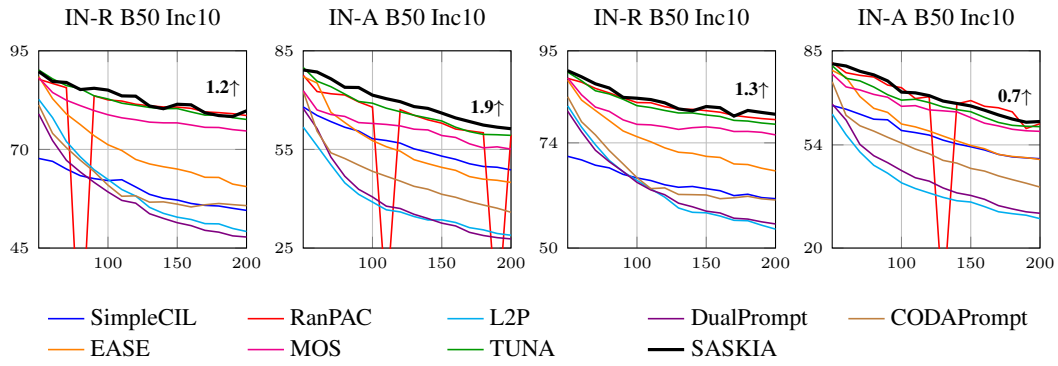


Figure A3: Incremental accuracy curves ($\mathcal{A}_T$) under the *offline-to-online protocol* with a **larger base class**. The first two figures utilize a ViT-B/16-IN21K backbone, while the last two utilize a ViT-B/16-IN1K backbone.

A.3.3 Larger base class

The B50 protocol pairs a large offline adaptation stage with fine-grained increments. It is the more realistic deployment shape, in which a substantial initial corpus followed by a long tail of new classes, and it exposes two failure modes that B0 hides. Table A4 reports final-stage accuracy and readout gap; Figure A3 shows the full trajectories.

Rigidity under per-task expansion. The limitations of per-task expansion are more pronounced here than under B0. The large initial adaptation induces a dominant representation, and the single-pass updates available to later tasks cannot reshape it, so accuracy declines steadily across the sequence rather than plateauing. The readout gap makes the same point from the other side: per-task expansion approaches retain gaps of 6–26 points at the final stage, so most of what their encoders support is also never expressed by their classifiers.

Ill-conditioning under a base-dominated readout. The analytic readouts face a different problem. With a large base set, the base classes dominate the spectrum of the accumulated Gram matrix $\mathbf{G}$, and later increments contribute comparatively little to it. This biases the classifier toward the base distribution, and it also degrades the conditioning of $\mathbf{G}$, which is a known practical concern for this family of methods. Consistent with (Prabhu et al., 2024) which reports instability in the RanPAC implementation, the consequence is visible in Figure A3: its accuracy collapses at isolated increments by 50 to 70 points at a single step and recovers within one or two further increments. We attribute this to numerical failure rather than to a learning dynamic. A method that had genuinely unlearned its classes would not recover from a single additional task, and the magnitude of the drops is inconsistent with any change in representation quality. The pattern is instead what an ill-conditioned solve produces: when an increment's features fall near the poorly-conditioned directions of a base-dominated $\mathbf{G}$, the solution for W becomes numerically unreliable, and the next increment perturbs the spectrum enough to restore it.

Table A5: **Linear probe accuracy** $\mathcal{P}_T$ on each method's own final encoder. Probes are fit from scratch on the frozen final encoder using all training data for classes seen up to the last stage, so $\mathcal{P}_T$ upper-bounds what a linear readout on that representation can achieve. Every readout gap reported in this paper is $\mathcal{G} = \mathcal{P}_T - \mathcal{A}_T$ using the corresponding entry here. Refitting per method is what makes $\mathcal{G}$ comparable across methods with different encoders. SimpleCIL's probe is identical across protocols because its encoder never changes, which also serves as a check that the two settings are otherwise matched. **Bold** marks the highest probe per column.

	IN21K, offline→online				**IN1K**, offline→online				**IN1K**, always-online			
Method	CIFAR	CUB	IN-R	IN-A	CIFAR	CUB	IN-R	IN-A	CIFAR	CUB	IN-R	IN-A
SimpleCIL	89.47	89.93	66.18	58.07	84.96	88.43	71.88	58.61	84.96	88.43	71.88	58.61
RanPAC	91.53	90.76	78.44	60.57	91.27	88.16	80.03	62.74	89.73	87.84	76.11	60.82
L2P	91.08	90.13	76.17	60.04	89.66	87.38	78.07	61.26	88.87	86.73	77.24	60.37
DualPrompt	91.64	90.37	77.08	60.58	90.24	87.92	78.85	61.77	89.42	87.11	77.74	60.94
CODAPrompt	91.96	90.18	78.02	61.34	90.86	87.83	79.95	62.68	89.95	86.94	78.36	61.16
EASE	92.07	90.91	77.96	60.84	90.58	89.17	79.34	61.43	90.06	88.88	78.57	61.05
MOS	92.36	91.02	79.37	62.03	90.97	89.46	80.47	62.73	89.72	88.76	78.13	61.82
TUNA	**92.98**	91.18	**82.03**	**65.47**	**91.83**	89.82	**83.07**	**66.18**	90.77	89.09	**81.58**	**64.27**
SASKIA (ours)	92.33	**91.39**	81.24	62.36	91.57	**89.88**	81.93	63.76	**91.41**	**89.57**	79.84	61.58

Why SASKIA is less exposed. Two design choices reduce this exposure, both of which follow from our diagnosis rather than from numerical considerations. The random projection is fixed for the entire sequence, so statistics accumulated at any stage describe the same space and remain mutually comparable. And EMA merging prevents any single stage—including the base stage—from determining the feature geometry, so the spectrum of $\mathbf{G}$ reflects the sequence as a whole rather than its largest task. SASKIA's B50 gaps (2.5–5.3 points) are close to its B0 gaps, indicating that a large base task does not degrade its readout alignment, and we observe no failed solves: we verify that $\mathbf{G} + \lambda\mathbf{I}$ admits a Cholesky factorization at every increment on every run.

A.3.4 Linear probe accuracies

Three patterns in Table A5 are not visible in accuracy alone. First, probe accuracy does not order the methods the way continual accuracy does: TUNA's final encoder supports the highest linear readout on the out-of-distribution benchmarks under every protocol, yet it finishes several points behind SASKIA on those same benchmarks. Its representation is better and its deployed model is worse, which is the dissociation our diagnosis predicts. Second, the frozen encoder trails the adapted ones by little on the in-distribution benchmarks (89.47 against 92.33 on CIFAR with IN21K) but by a wide margin out-of-distribution (66.18 against 81.24 on IN-R). Adaptation raises the achievable readout precisely where pretrained features transfer poorly, and contributes little where they already suffice. Third, comparing the two IN1K columns, removing the offline phase costs SASKIA 0.2–2.2 points of probe accuracy while its readout gap stays within 0.5 points; what the always-online protocol takes from SASKIA is representation quality, not readout alignment. RanPAC loses most in the same comparison (80.03 → 76.11 on IN-R), consistent with its design, which concentrates all adaptation in the first task.

A.4 Memory and Numerical Considerations

Figure 5a reports trainable parameters, which is not the same as memory. All compared methods are replay-free and store no raw images, so what distinguishes them is auxiliary state. We account for that state in full below, then document the implementation choices that keep our closed-form readout both cheap and well-conditioned.

A.4.1 What each method stores

Prompt- and adapter-based methods introduce task-specific parameters at multiple layers and retain them all at inference, so their footprint grows linearly with the number of tasks. Several also maintain dataset statistics: MOS (Sun et al., 2024) and TUNA (Wang et al., 2025) store per-class feature means and covariances for post-hoc classifier alignment, adding both memory and computation on top of their module growth. Methods with analytic readouts, namely RanPAC (McDonnell et al., 2024) and SASKIA, instead retain a fixed Gram matrix and a class-indexed correlation matrix: a large constant with almost no growth.

SASKIA's footprint. Table A6 decomposes it. Four of the five components are constant in the length of the task sequence. $\mathcal{R}$ costs nothing at rest: its entries are drawn once from a fixed seed and regenerated deterministically, so only the seed is stored. $\mathbf{G}$ is allocated once at its final size and never reallocated, and is not specific to our method; any closed-form ridge readout, including RanPAC's, carries the same term. The SSF modulation and hypernetwork have fixed capacity by construction and EMA merging updates weights in place and adds no parameters. Only $\mathbf{C}$ grows, by one M-dimensional column per class.

Table A6: **Memory decomposition for SASKIA** on ViT-B/16 (d=768) with M=10,000. $\mathbf{G}$ is symmetric and stored as its upper triangle in float16, and cast to float32 for accumulation and factorization (Section A.4.2); all other components are float32. Sizes use 1 MB $= 10^6$ bytes. SSF modulation is inserted at four sites per block across 12 blocks, all at width d; the hypernetwork uses r=256 and is held in two copies. Only $\mathbf{C}$ grows.

Component	Shape	Size	Growth
Gram matrix $\mathbf{G}$	$M \times M$	100.0 MB	0
Correlation $\mathbf{C}$	$M \times \lvert\mathcal{Y}\rvert$	0.04 MB/class	+0.04 MB per class
Projection $\mathcal{R}$	$d \times M$	0 MB	0
SSF modulation $\mathcal{A}$	$2d$ per site, 48 sites	0.3 MB	0
Hypernetwork $\mathcal{H}$ (active + shadow)	$2 \times (d \to r \to 2d)$	4.7 MB	0
Total at 200 classes		**113.0 MB**	+0.04 MB/class

Growth over a sequence. The trade-off is a large constant against near-zero growth (Table A7). At M=10,000 SASKIA uses more memory than the expansion methods on short sequences and less beyond roughly 32 tasks, where their linear growth overtakes our constant; at M=5000 the crossover falls to roughly 8 tasks and SASKIA is the cheapest method in every setting we evaluate. We state this plainly rather than claiming a uniform advantage: the constant is real, and on short sequences it dominates. It is also configurable, since $\mathbf{G}$ scales as M^2 and Section A.5 shows that reducing M to 5000 costs 0.56 points of accuracy while cutting $\mathbf{G}$ fourfold.

Table A7: **Memory growth per task** and total auxiliary memory after T tasks of 10 classes each. SASKIA's footprint is dominated by a constant, the expansion methods' by growth; the columns cross before T=20 at M=10,000, and before T=10 at M=5000.

Method	Constant	Per task	T=10	T=20	T=40	T=100
EASE (Zhou et al., 2024b)	∼1.0 MB	+3.7 MB	38.0 MB	75.0 MB	149.0 MB	371.0 MB
MOS (Sun et al., 2024)	∼1.0 MB	+6.6 MB	67.0 MB	133.0 MB	265.0 MB	661.0 MB
TUNA (Wang et al., 2025)	∼1.0 MB	+6.0 MB	61.0 MB	121.0 MB	241.0 MB	601.0 MB
SASKIA (M=10,000)	105.0 MB	+0.4 MB	109.0 MB	113.0 MB	121.0 MB	145.0 MB
SASKIA (M=5,000)	30.0 MB	+0.2 MB	32.0 MB	34.0 MB	38.0 MB	50.0 MB

A.4.2 Implementation of the analytic readout

Several implementation choices make the analytic readout both memory-efficient and numerically stable in practice. While none of them changes the underlying solution, they substantially reduce computational overhead and avoid common sources of instability that arise in large-scale closed-form classifiers.

Symmetric storage. $\mathbf{G} = \sum_{t,i} \boldsymbol{\psi}_{t,i} \boldsymbol{\psi}_{t,i}^\top$ is symmetric by construction. We therefore store only its upper triangular part, reducing the dominant memory cost by roughly a factor of two. Statistics are accumulated directly into this compact representation, avoiding the need to materialize an additional $M \times M$ buffer during updates.

Mixed precision. Gram matrix $\mathbf{G}$ is the fixed and heaviest component in our approach. To reduce memory consumption, $\mathbf{G}$ is stored in float16 between updates and cast to float32 whenever arithmetic is performed. Both accumulation and factorization therefore operate entirely in float32, ensuring that reduced precision affects storage only. This halves the memory footprint while avoiding the numerical degradation that would result from repeatedly accumulating thousands of outer products in float16.

Solving rather than inverting. We never explicitly compute $(\mathbf{G}+\lambda\mathbf{I})^{-1}$. Since $\mathbf{G}+\lambda\mathbf{I}$ is symmetric positive definite for $\lambda > 0$, the classifier weights are obtained via a Cholesky factorization followed by two triangular solves against $\mathbf{C}$. This approach is faster, requires no additional $M \times M$ allocation, and is numerically preferable because explicit inversion amplifies conditioning issues by effectively squaring the condition number.

Amortized selection of λ. The ridge parameter λ is selected by grid search on held-out data after each task. Performing a separate factorization for every candidate would be unnecessarily expensive. Instead, we eigendecompose $\mathbf{G} = U\Lambda U^\top$ once and reuse the decomposition throughout the sweep. For any candidate λ, $W = U(\Lambda+\lambda\mathbf{I})^{-1}U^\top\mathbf{C}$, reducing evaluation of additional grid points to a simple diagonal rescaling. Consequently, the cost of hyperparameter selection is negligible relative to the single decomposition it reuses.

Conditioning. Poor conditioning of $\mathbf{G}$ is a well-known issue for this class of analytic readouts. Prabhu et al. (2024) report numerical instability in the original RanPAC implementation and offer some corrections. The problem is particularly pronounced when a subset of tasks dominates the accumulated statistics, as can occur under the large base-task setting of Section A.3.3. SASKIA mitigates this risk in two ways. First, the random projection remains fixed throughout the entire sequence, ensuring that all statistics are accumulated in a consistent feature space. Second, the EMA-merged hypernetwork prevents any individual task from inducing abrupt changes in feature geometry. In all experiments we verify that $\mathbf{G}+\lambda\mathbf{I}$ remains Cholesky-factorizable at every task.

A.5 Component Decomposition and Sensitivity to M and α

Table A8: **Sensitivity to projection dimension M and EMA decay α** on IN-A B0 Inc20. α=1 keeps the shadow hypernetwork fixed after the offline phase, and α=0 removes smoothing entirely, so the geometry is set by the most recent task; both are reported to bound the interior sweep. $|\mathbf{G}|$ is the size of the Gram matrix under the symmetric float16 storage of Table A6. The right-hand block gives the probe accuracy $\mathcal{P}_T$ and readout gap $\mathcal{G}$ at M=10,000; since $\mathcal{A}_T = \mathcal{P}_T - \mathcal{G}$, it shows where each setting loses accuracy. Both endpoints lose mainly in the gap, but for different reasons: without merging the accumulated statistics go stale, while with a frozen shadow nothing moves, so its gap is estimator residual alone.

	$\mathcal{A}_T$ by projection dimension M				At M=10,000	
α	1,000	5,000	10,000	20,000	$\mathcal{P}_T$	$\mathcal{G}\downarrow$
0 (no merging)	48.06	52.62	56.10	56.24	61.32	5.22
0.50	52.13	55.87	57.30	57.41	61.70	4.40
0.90	52.22	56.94	58.13	58.15	62.38	4.25
0.95	54.83	58.21	**58.77**	58.81	62.36	**3.59**
0.99	53.31	57.08	57.53	57.85	62.17	4.64
1 (frozen shadow)	50.44	54.09	56.81	56.82	61.56	4.75
$\lvert\mathbf{G}\rvert$	1 MB	25 MB	100 MB	400 MB		

Projection dimension. Accuracy rises steeply from M=1000 to M=5000 (+3.4 points at α=0.95) and then flattens: M=10,000 adds a further 0.56, and M=20,000 only 0.04 for four times the memory. We therefore use M=10,000 throughout, matching RanPAC (McDonnell et al., 2024), while noting that M=5000 retains all but 0.56 points at a quarter of the memory (Section A.4). Only M=1000 is genuinely too small, where the projected space no longer supports the linear separability the readout depends on.

EMA decay and where the accuracy goes. Accuracy peaks at α=0.95 and falls on both sides, by 2.67 points without merging (α=0) and 1.96 points with a frozen shadow (α=1). Because $\mathcal{A}_T = \mathcal{P}_T - \mathcal{G}$, each loss splits exactly into a change in what the representation supports and a change in how much of it the readout recovers (Table A8). At both endpoints most of the loss appears in the gap, 1.63 points without merging and 1.16 with a frozen shadow, against probe losses of 1.04 and 0.80. The two gaps, however, cannot have the same origin. Without merging, the hypernetwork follows each new task, so statistics accumulated under earlier tasks describe a geometry that has since moved; this is the staleness our diagnosis identifies. With a frozen shadow, the geometry never

Table A9: **Component decomposition.** Probe accuracy $\mathcal{P}_T$, last accuracy $\mathcal{A}_T$, and readout gap $\mathcal{G} = \mathcal{P}_T - \mathcal{A}_T$ with ViT-B/16-IN21K under the offline-to-online protocol. Rows 1–3 share a frozen encoder and differ only in the readout, so their probe is identical by construction and every change in $\mathcal{G}$ is attributable to the classifier. Rows 4–7 change the encoder. Row 4 is RanPAC; row 5 replaces its first-session adapters with our shared scale-and-shift modulation at a seventeenth of the trainable parameters. Row 6 adds the instance-level hypernetwork without EMA merging, and row 7 adds the merging. **Params** counts trainable parameters outside the readout.

	Encoder adaptation	Readout	CUB B0 Inc10			IN-A B0 Inc20			Params
			$\mathcal{P}_T$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	$\mathcal{P}_T$	$\mathcal{A}_T$	$\mathcal{G}\downarrow$	
1	none (frozen)	linear (SGD)	89.93	78.50	11.43	58.07	41.25	16.82	0
2	none (frozen)	ridge	89.93	85.95	3.98	58.07	48.22	9.85	0
3	none (frozen)	ridge + RP	89.93	87.20	2.73	58.07	50.64	7.43	0
4	first-session adapters (RanPAC)	ridge + RP	90.76	88.39	2.37	60.57	53.36	7.21	1.19M
5	global SSF	ridge + RP	90.45	88.95	1.50	60.34	55.80	4.54	0.07M
6	global SSF + $\mathcal{H}$ (no EMA)	ridge + RP	91.05	89.10	1.95	61.32	56.10	5.22	0.66M
7	global SSF + $\mathcal{H}$ + EMA	ridge + RP	**91.39**	**89.77**	**1.62**	**62.36**	**58.77**	**3.59**	0.66M

moves after the first task, so no staleness is possible: the larger gap there can only be estimator residual, which grows when the representation stops adapting to the classes that arrive later, the same limitation that costs the frozen shadow 0.80 points of probe accuracy. Between the endpoints, the probe stays within 0.2 points of its best value for $0.90 \leq \alpha \leq 0.99$, so the accuracy differences there are carried almost entirely by the gap. The decay rate therefore trades one source of gap against the other: too little smoothing lets accumulated statistics go stale, while too much leaves a representation the readout fits less well. Measured against no merging, EMA merging is worth 2.67 points.

The components are not interchangeable increments. Table A9 separates what each part of SASKIA contributes, and the decomposition is not monotone. Rows 1–3 hold the encoder fixed: replacing a gradient-trained head with ridge regression closes 7.4 points of gap on CUB and 7.0 on IN-A without touching the representation, and the random projection closes a further 1.3 and 2.4. This is the largest single improvement anywhere in the table, and it involves no adaptation at all.

Rows 4 and 5 compare adaptation strategies at a fixed readout. Replacing RanPAC's first-session adapters with our shared scale-and-shift modulation improves both the probe and the gap while using 17× fewer trainable parameters, so the gain is not a matter of added capacity.

Rows 6 and 7 are the informative pair. Adding the instance-level hypernetwork *without* EMA merging raises the probe—the representation genuinely improves, by 0.60 on CUB and 0.98 on IN-A—but the gap *widens*, from 1.50 to 1.95 and from 4.54 to 5.22. This is our diagnosis reproduced inside our own method: per-sample adaptation that moves the feature geometry every task improves what the representation supports while invalidating the statistics accumulated over it, and the deployed model captures almost none of the improvement. EMA merging is what makes the two compatible, recovering 0.33 and 1.63 points of gap and turning a representation gain into an accuracy gain. The hypernetwork and the merging are therefore not two independent improvements stacked on a strong baseline: one creates the problem that the other solves, and neither is useful without the other.

A.6 Limitations

Three limitations bound our conclusions. First, because our experiments are limited strictly to vision tasks, our diagnosis reflects linear decodability within the visual domain rather than general representation quality; a representation could improve in ways a linear map cannot express, which our evaluation would fail to capture. Second, Galashov et al. (2023) caution that effects attributed to continual representation adaptation may partly reflect overlap between pretraining and benchmark distributions. This concern applies to our CIFAR and CUB results, though the readout gap is also highly visible on ImageNet-A and ImageNet-R, which are constructed to be adversarial to ImageNet pretraining. Third, on those same out-of-distribution targets, a gap to methods trained to convergence per task persists (main paper, Table 4), and SASKIA's own always-online losses concentrate there (Section A.3.2). Both suggest that single-pass adaptation still leaves accuracy on the table where pretrained features transfer least well.