Task-Free Continual Learning via Order-Invariant Linearized Adaptation and Density-Guided Adapter Routing

Hang Thi-Thuy Le[1] **Nam-Quan Nguyen**[2] **Lam-Huy Nguyen**[2] **Dien Dinh**[2] **Minh Hoang**[3] **Trong Nghia Hoang**[1]

[1]Washington State University, Pullman, Washington, USA
[2]University of Science, VNU-HCM, Ho Chi Minh City, Vietnam
[3]Princeton University, Princeton, New Jersey, USA

Abstract

Task-free continual learning (TFCL) aims to adapt models to non-stationary data streams without knowing task boundaries. In TFCL, catastrophic forgetting arises from both the evolving data distributions and the order-sensitive nature of batch-streaming optimization. To address these challenges, we propose a new TFCL framework that mitigates optimization-induced forgetting via order-invariant linearized adaptation during learning and accommodates evolving data distributions via density-guided adapter routing for more accurate and effective adapter retrieval during inference. We also provide a theoretical characterization of retrieval error in terms of density estimation quality and cross-adapter embedding separability. Experiments across multiple benchmarks demonstrate that our proposed method consistently achieves higher accuracy and lower forgetting than strong TFCL baselines. For reproducibility, our experimental code is available at: `https://github.com/Alisia0303/HESTIA.git`.

1 INTRODUCTION

Neural networks are increasingly deployed in environments where data batches arrive sequentially and the corresponding data distributions also evolve over time. In such settings, models must adapt continuously without retraining from scratch and accessing historical data. This leads to a catastrophic forgetting challenge: parameter updates that improve performance on recent data may degrade performance on earlier data once those examples are no longer available [McCloskey and Cohen, 1989, French, 1999, Kirkpatrick et al., 2017]. Such challenge has been extensively studied in continual learning (CL) but many CL benchmarks assume explicit task identities or task-boundary signals while real-world data streams are often unsegmented and unlabeled. Such settings are often known as task-free continual learning (TFCL) where adaptation must proceed without task identifiers at both training and test time [Van de Ven et al., 2022, De Lange and Tuytelaars, 2021]. Compared with task-, class-, or domain-incremental settings where task identities are provided during training, TFCL is more challenging because the learner must simultaneously process data in small batches, detect distribution changes, and retrieve relevant knowledge without additional supervision.

A key challenge in TFCL is that catastrophic forgetting can accumulate from multiple sources. First, forgetting can emerge even within a single task stream because training still proceeds on randomly selected mini-batches and parameter updates can be biased toward recent batches. As a result, the model may lose previously learned information despite seeing data from the same distribution. We observe that even with strong pre-trained backbones, standard streaming optimization can still induce substantial forgetting (Section 4). Second, forgetting becomes more severe under rapid changes in task distribution. As the data stream transitions across task distributions, the learner must incorporate new patterns while avoiding interference with earlier knowledge, which can degrade past performance.

To mitigate these sources of forgetting, most recent TFCL methods implicitly adopt a three-stage strategy: (1) learn an online distribution- or task-specific parameters to reduce distribution interference; (2) detect change points (or distribution shifts) and expand model capacity when needed; and (3) retrieve the appropriate module at test time for unseen inputs. For example, replay-based methods partially restore the offline regime by interleaving current data with stored examples [Aljundi et al., 2019a, Jin et al., 2021, Ye and Bors, 2022c,b], which can reduce forgetting. However, effective replay in task-free streams requires difficult design choices about what to store, how to retrieve it, and how to allocate memory and computation over long data streams. Modular expansion methods reduce interference by assigning new capacity to novel data while preserving earlier modules. TFCL

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:3274–3297.

however introduces an additional challenge: reliable routing must be maintained under evolving representations and in the absence of task labels. Consequently, such approaches often rely on auxiliary shift detectors or learned retrievers trained incrementally [Ye and Bors, 2022a, 2024, Zhu et al., 2024], which can themselves suffer from forgetting.

Recent Transformer backbones [Vaswani et al., 2017] combined with parameter-efficient fine-tuning (PEFT) [Houlsby et al., 2019, Lester et al., 2021, Li and Liang, 2021, Hu et al., 2022, Liu et al., 2022] have significantly improved continual adaptation by restricting updates to a small set of trainable parameters, thereby reducing memory and computational overhead. Some PEFT-based methods, such as L2P Wang et al. [2022c] and DualPrompt Wang et al. [2022b] consolidate knowledge into shared adapter or prompt pools to compartmentalize task-specific knowledge and optimizes a prompt-selection mechanism to retrieve relevant prompts for each input. Alternatively, RanPAC [McDonnell et al., 2023] exploit statistics that are less sensitive to representation order to mitigate forgetting.

Despite these advances, important gaps remain for Transformer-based TFCL. Existing approaches can still be sensitive to optimization trajectory and stream order [Wang et al., 2022c,b, Ye and Bors, 2024]; others rely on heuristic representation choices or freezing strategies [McDonnell et al., 2023, Wu et al., 2025] that reduce forgetting but may degrade performance (see Section 4). Moreover, change detection and routing are often omitted due to predefined fixed-size adapter pools [Wang et al., 2022c,b] or treated as auxiliary [Ye and Bors, 2024] components rather than lightweight designed jointly for task-free adaptation.

To address these limitations in existing work, we propose HESTIA, a *comprehensive design of TFCL framework utilizing order-invariant linearized adaptation and density-guided adapter routing*. HESTIA reduces forgetting through complementary mechanisms: it mitigates order-induced drift during continual optimization, enables modular adaptation via a density-guided routing under distribution shift. Our contributions are summarized as follows:

1. Order-invariant continual adaptation via linearized fine-tuning. Within a task, we reinterpret streaming adaptation through a linearized formulation around pre-trained Transformer weights, revealing additive updates that accumulate sufficient statistics across batches and remove dependence on mini-batch order without replay buffers (Section 2.2). This directly mitigates order-induced forgetting and substantially reduces optimization drift as streams become longer (see Section 4).

2. Dynamic adapter expansion and change detection via density-guided approach. To address non-stationarity across data segments, we introduce density-based models (which represent *task keys*) associate with task-specific adapters. Inputs are routed by likelihood-based comparison against these task keys (Section 2.3). Because the routing mechanism introduces no cross-task learnable parameters, it avoids an additional source of forgetting. Furthermore, leveraging the same learned density-based models, we design a compact change-point detector based on these model convergences and distributional mismatch. This avoids auxiliary detector networks and reduces system complexity while maintaining high detection accuracy (see Section 2.3).

3. Theoretically verified retrieval-error hypothesis. To test our hypothesis on how retrieval accuracy depends on learned task keys, we provide a theoretical characterization of routing error in terms of (i) task-key (density) estimation quality and (ii) separability among adapter-induced embedding distributions (Section 3). Empirically, dynamic adapter expansion with our density-guided routing improves robustness under distribution shifts (see Section 4).

4. Extensive empirical validation. We evaluate HESTIA on Split CIFAR-10/100 (standard benchmarks), ImageNet-R (a more challenging downstream task), and VTAB (long-stream, real-world challenging scenarios) under task-free streaming protocols. Across all benchmarks, HESTIA consistently improves average and final accuracy, achieving gains of up to 60% over TFCL approaches. Moreover, it ranks top-1 in forgetting mitigation, showing substantially lower forgetting. Finally, HESTIA remains competitive in memory usage by avoiding replay buffers and maintaining only lightweight adapter and density statistics (Section 4).

2 METHODS

Approach Overview. Our goal is to mitigate catastrophic forgetting arising from two sources in TFCL: (i) *order-induced optimization drift* and (ii) *distribution shift*. We address these forgetting sources through complementary mechanisms: *order-invariant linearized adaptation* (Section 2.2) and *density-guided adapter routing* (Section 2.3).

Section Organization. Section 2.1 states problem setups and notation. Section 2.2 formalizes a linearized adaptation solution around pretrained weights, enabling stable streaming updates that are less sensitive to mini-batch order. Section 2.3 introduces *task keys*, implemented as density-based models such as Gaussian mixture models that are learned online from streaming data batches. At inference time, these keys route unseen inputs to the appropriate adapter via likelihood-based comparison. Section 2.3 also presents a lightweight change-point detector built from the same density models, providing a compact and accurate mechanism for distribution-shift detection.

2.1 PROBLEM SETUP AND NOTATIONS

We study task-free continual learning (TFCL) in batch-streaming settings where labeled data arrive sequentially

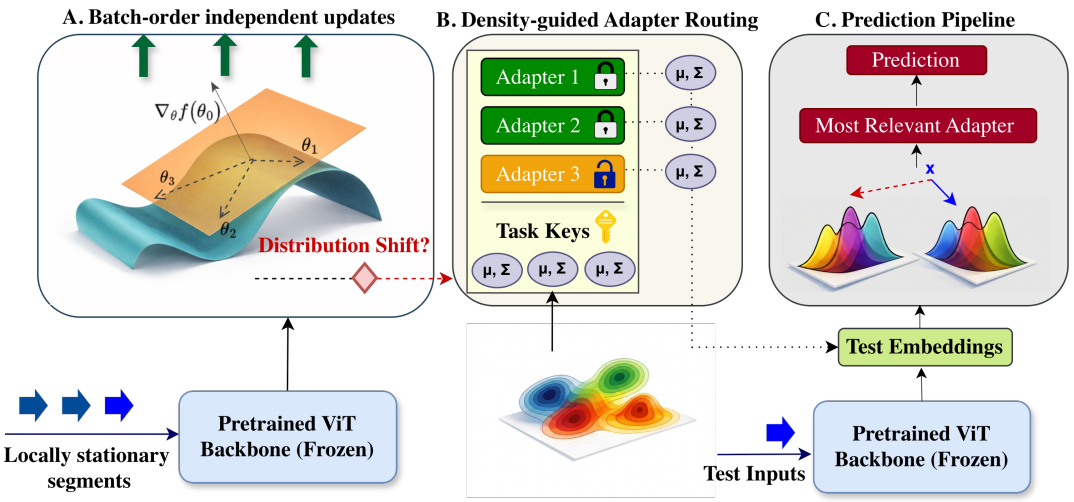

Figure 1: Overview of our task-free continual learning framework, HESTIA. **A:** Streaming batches are processed via order-invariant linearized adaptation to mitigate order-induced forgetting. **B:** When distributional change is detected, a new adapter is instantiated while previous adapters remain frozen. Each adapter maintains Gaussian mixture models (task keys) over its embedding space. **C:** At inference time, inputs are routed by likelihood-based comparison to these task key sets.

in small batches and task identities are unavailable at both training and test time. This is formalized below.

Let $\mathcal{S} = \{D_1, D_2, \ldots, D_B\}$ denote the stream of observed batches. Each batch $D_b = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_b}$ contains n_b labeled samples where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, C\}$. Due to limited storage, however, data batches cannot be stored for reuse. The data stream is also generated by an unknown sequence of data distributions $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T)$, where samples within a contiguous segment are drawn from the same distribution. The change points where the data stream switches from one distribution to another are unknown.

Let $h_\omega(\boldsymbol{x}) = h(x; \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega})$ denote the model with parameters $\boldsymbol{\omega} = \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega}$, and pretrained weights ω_0. We adopt a parameter-efficient fine-tuning (PEFT) setting where the backbone is frozen and only a small set of adapter parameters is updated during continual learning. In a centralized setting where all data batches are known a priori, PEFT can be viewed as finding a global optimizer via

$$\boldsymbol{\omega}_* = \arg\min_{\boldsymbol{\omega}} \mathcal{R}(\boldsymbol{\omega}, P) \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim P}\big[\ell\big(h_\omega(\boldsymbol{x}), y\big)\big] , \quad (1)$$

where P is the empirical data distribution and ℓ is an individual loss point-wise (e.g., cross-entropy loss). However, as past data batches are not accessible, TFCL needs to design a local minimizer procedure $\mathcal{R}(\boldsymbol{\omega}, D_b)$ to be executed at each streaming step b with local data batch D_b:

$$\begin{aligned} \boldsymbol{\omega}_b &= \boldsymbol{\omega}_{b-1} + \Delta\boldsymbol{\omega}_b, \\ \Delta\boldsymbol{\omega}_b &= \arg\min_{\Delta\boldsymbol{\omega}} \mathcal{R}(\boldsymbol{\omega}_{b-1} + \Delta\boldsymbol{\omega}, D_b), \end{aligned} \quad (2)$$

This procedure must be designed such that the generalized performance of $\boldsymbol{\omega}_T$ is similar to $\boldsymbol{\omega}_*$, which is the ultimate goal of mitigating forgetting. This is however non-trivial because this update schedule is inherently order-dependent and can induce forgetting as bias towards latest batches. Minimizing this order-dependence is therefore key to mitigating forgetting in TFCL. We will achieve this via two complementing update strategies.

First, we will adopt a linearized fine-tuning formulation that reinterpret updates within a task as additive accumulation of sufficient statistics across batches which does not depend on the order in which data batches from the corresponding data distribution arrive (see Section 2.2).

Second, to mitigate dependence on the order in which different tasks (with different data distributions) arrive, we will introduce a change point detection mechanism. When a change point is detected, we expand the model capacity via adding new adapters dedicated to new data insights that the new task introduces. As new components are specifically created for each new data patterns discovered by the change point detection mechanism, their updates are less dependent on the order in which task arrives or equivalently, the sequence of change points. While this is not perfect due to (inevitably) imperfection in change point detection mechanism, it can be empirically shown to help achieve significantly better performance than existing TFCL baselines, demonstrating efficiency in mitigating task-order dependence or catastrophic forgetting (see Section 2.3).

Formally, let $\mathcal{G}$ denote the set of adapters, and let $G_i \in \mathcal{G}$ be the i-th adapter with its associated task key. Training starts with a single adapter and expands over time as new change points are detected. To reduce interference, only the

newly added adapter is updated, while previously learned adapters remain frozen. At inference time, a routing function compares an unseen input against task keys and selects relevant adapters. These mechanisms are detailed next.

2.2 ORDER-INVARIANT LINEARIZED ADAPTATION

Our key observation is that joint minimization of the global risk is inherently order-invariant, since all samples contribute symmetrically to the objective. In contrast, gradient-based continual fine-tuning replaces this global objective with a sequence of local optimizations, introducing order dependence and catastrophic forgetting. To recover the order-invariant structure of joint training in a streaming setting, we propose an alternative formulation in which the contribution of each batch can be accumulated additively, without requiring access to past data. We show that this becomes possible by analyzing fine-tuning in the tangent space around the pretrained parameters. Under a linearization of the model and an approximate quadratic loss, the global objective reduces to a least-squares problem whose sufficient statistics decompose across batches. This observation motivates our order-invariant linearized adaptation framework.

Linearization and Loss Approximation. We adopt a local approximation around pretrained weights ω_0 where cross-entropy is replaced by a squared-loss surrogate, and predictions are linearized for small updates,

$$\hat{p}_y(\boldsymbol{x}; \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega}) \approx \hat{p}_y(\boldsymbol{x}; \boldsymbol{\omega}_0) + J(\boldsymbol{x})\Delta\boldsymbol{\omega}, \qquad (3)$$

where we denote the Jacobian as $J(\boldsymbol{x}) = \nabla_{\omega_0}\hat{p}_y(\boldsymbol{x}; \boldsymbol{\omega}_0)$.

Additive Decomposition Across Batches. Substituting Eq. 3 into the approximate global objective results in a regularized least-squares (ridge) problem. For each batch D_b, let $\boldsymbol{A}_b$ denote the Jacobian matrix that stacks $J(\boldsymbol{x})$ of the data samples $\boldsymbol{x}$ in D_b. Let $\boldsymbol{r}_b$ denote the corresponding regression targets induced by the loss approximation. The resulting a closed-form expression for the optimizer that can be computed in an order-invariant manner:

$$\Delta\boldsymbol{\omega}^* = \left(\sum_b \mathbf{A}_b^\top \mathbf{A}_b + \lambda \mathbf{I}\right)^{-1} \left(\sum_b \mathbf{A}_b^\top \mathbf{r}_b\right). \qquad (4)$$

This form admits an additive decomposition over streaming data batches via the following accumulated statistics:

$$\boldsymbol{G}_b \;=\; \boldsymbol{G}_{b-1} + \boldsymbol{A}_b^\top \boldsymbol{A}_b \quad \text{and} \quad \boldsymbol{g}_b \;=\; \boldsymbol{g}_{b-1} + \boldsymbol{A}_b^\top \boldsymbol{r}_b\,. \quad (5)$$

Using these order-invariant accumulated statistics, we can compute the optimal update,

$$\Delta\boldsymbol{\omega}^* \quad = \quad (\boldsymbol{G} + \lambda \boldsymbol{I})^{-1}\boldsymbol{g}\,. \qquad (6)$$

This is formalized via the following result:

Algorithm 1 Task-Free Continual Learning

input backbone $h(\cdot; \boldsymbol{\omega}_0)$; hyper-parameters λ; and (w, k).
output adapter bank $\mathcal{B} = \{(\Delta\boldsymbol{\omega}^{(t)}, \boldsymbol{\theta}^{(t)})\}_{t=1}^{T}$.

```
initialize t ← 1, B ← ∅, G^(t) ← 0, g^(t) ← 0
initialize density parameters θ^(t)
initialize routing score buffer S ← ∅
for each arriving batch b do
    compute linearized adaptation (A_b, r_b) – Eq. (4)
    accumulate sufficient statistics (G^(t), g^(t)) – Eq. (5)
    compute routing score s_t ← S_t(h_t(x)) – Eq. (9)
    online learn θ^(t) by running mean/covariance
    add routing scores to buffer S ← s_t
    if change-point criterion – Eq. (11) – is satisfied then
        compute new adapter by solving Eq. (7)
        store (Δω^(t), θ^(t)) in B
        reinitialize G^(t) ← 0, g^(t) ← 0, and S ← ∅;
        t ← t + 1 # increase the no. of model components
    end if
end for
return B
```

Proposition 2.1 (Order Invariance)**.** *Under the linearized objective, sufficient statistics (G, g) decompose additively across batches. Consequently, the solution $\Delta\boldsymbol{\omega}^*$ is invariant to batch order within a locally stationary distribution.*

Continual adaptation within each task thus reduces to accumulating batch-wise sufficient statistics rather than replaying past data. Detailed derivations are deferred to Appendix A.1. Practical implementation to avoid expensive costs from computing Jacobian is provided in Appendix D.

2.3 DENSITY-GUIDED ADAPTER ROUTING AND CHANGE DETECTION

The linearized fine-tuning framework in Section 2.2 provides an order-invariant solution *within* a single task where the task distribution does not change across streaming batches. In task-free continual learning (TCFL), however, the stream might switch to another data distribution from another task whose optimal adaptations can differ substantially. A single shared linearized solution is therefore often insufficient and may cause representation mismatch under distribution shift (see Section 4).

To handle this, we maintain and adaptively grow a *bank of task-specific adapters*. Each adapter is paired with a *task key* that summarizes its induced embedding distribution using a Gaussian mixture model. At inference time, unseen inputs are routed to the most compatible adapter via likelihood-based comparison. We describe this mechanism below.

Adapter Instantiation. When a change point is detected, we finalize the accumulated statistics of the current task t

and compute a task-specific linearized update

$$\Delta\boldsymbol{\omega}^{(t)} \quad = \quad \Big(\boldsymbol{G}^{(t)} + \lambda \boldsymbol{I}\Big)^{-1} \boldsymbol{g}^{(t)} \,. \tag{7}$$

This update defines the t-th adapter (i.e., a task-specific adapted model) which induces an embedding function

$$h_t(\boldsymbol{x}) \quad = \quad h\Big(\boldsymbol{x}; \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega}^{(t)}\Big) \,. \tag{8}$$

Task Keys as Gaussian Mixture Models. For each adapter h_t, we model the distribution of induced embeddings $\boldsymbol{z} = h_t(\boldsymbol{x})$ using a mixture of Gaussians P_{θ_t} with parameters $\boldsymbol{\theta}_t = \{(\boldsymbol{m}_{t,i}, \boldsymbol{\Lambda}_{t,i})\}_{i=1}^{K}$, where $\boldsymbol{m}_{t,i}$ and $\boldsymbol{\Lambda}_{t,i}$ denote the mean and precision matrix of the i-th component. This Gaussian mixture serves as the *task key* for adapter t.

Given an embedding $\boldsymbol{z}$, we define its routing score under adapter t as follows:

$$S_t(\boldsymbol{z}) = \min_{i \in \{1,\ldots,K\}} (\boldsymbol{z} - \boldsymbol{m}_{t,i})^\top \boldsymbol{\Lambda}_{t,i} (\boldsymbol{z} - \boldsymbol{m}_{t,i}), \tag{9}$$

which corresponds to a negative log-likelihood score under the Gaussian mixture. The corresponding Gaussian component statistics i are then updated online using running mean/covariance estimates as new samples arrive. The overall mixture can therefore be interpreted as a piecewise Gaussian approximation of the embedding distribution.

Routing at Inference. For a test input $\boldsymbol{x}_*$, we evaluate the adapter-specific embedding under each adapter and select

$$t_* \quad = \quad \arg\min_t \;\; S_t(h_t(\boldsymbol{x}_*)) \,. \tag{10}$$

The final prediction is produced by the selected adapter t_*.

Density-Driven Change Detection. We use a *lightweight* change-detection mechanism that reuses the same density scores employed for adapter routing, avoiding any auxiliary detector network. The key idea is to monitor the evolution of Gaussian mixture model fit under the current adapter. Intuitively, once the score dynamics stabilize or convergence, the persistent arrival of poorly explained samples indicates a likely distribution shift.

For a training sample $\boldsymbol{x}$ processed by the current adapter t, let $S_t(h_t(\boldsymbol{x}))$ denote its routing score in Eq. (9). We maintain a sliding window of recent scores and declare *density convergence* when the score process becomes approximately stationary. Concretely, at index j we compute the mean μ_j and standard deviation σ_j over the previous window, then declare convergence if all scores in the next window remain within a tolerance band:

$$\max_{0 \le i < w} \big|S_{j+i} - \mu_j\big| \quad \le \quad k\,\sigma_j, \tag{11}$$

where w is the window size and k controls the strictness of the stationarity test. Intuitively, Eq. (11) detects when the density model has stabilized on the current stream segment; after convergence, sustained abnormal scores indicate mismatch and are treated as evidence of a distribution shift and trigger adapter expansion. We then finalize the linearized solution and its density parameters, add them to the adapter bank, and initialize fresh statistics for the new segment.

The overall workflow is illustrated in Figure 1, and the full training procedure is summarized in Algorithm 1.

3 THEORETICAL ANALYSIS

Our framework maintains multiple task-specific adapters, each paired with a density-based model (e.g., Gaussian mixture model). We hypothesize that retrieval error is governed by two complementary factors: (i) the *density modeling error* of each adapter-specific density model (captured by δ), and (ii) the *cross-adapter separation* between adapter-induced embedding distributions (captured by Δ). Intuitively, retrieval succeeds when embeddings produced by the correct adapter are well explained by their own density model, while being poorly explained by density models associated with other adapters. In this section, we formalize this hypothesis and provide a theoretical characterization of retrieval error under mild regularity assumptions. The result should be interpreted as a *theoretical test* of the above hypothesis: it shows that, under Assumptions A1–A3 (see Appendix A.2), retrieval error decreases as density fitting improves (smaller δ) and cross-adapter separation increases (larger Δ). The formal statement is given in Theorem 3.1, with full proof deferred to Appendix A.2.

Let $\mathcal{D}_k$ denote the k-th distribution in the stream, and let h_{ω_k} be the embedding function induced by the linearized optimal parameters learned for that distribution. The corresponding embedding distribution is the pushforward $Q_k := (h_{\omega_k})_{\#}\mathcal{D}_k$. For each distribution k, we fit a Gaussian density model P_{θ_k} to samples from Q_k. Given a test input $\boldsymbol{x}$, we compute adapter-specific embeddings $\boldsymbol{z}_j = h_{\omega_j}(\boldsymbol{x})$, for all adapters j – see Eq. 7, and retrieve the adapter with the smallest negative log-likelihood score:

$$\hat{k}(\boldsymbol{x}) \quad = \quad \arg\min_j d(\boldsymbol{z}_j, P_{\theta_j}) \,, \tag{12}$$

Under this formulation, retrieval error occurs when an incorrect adapter-density pair assigns a better score than the correct one. We analyze this event in a pairwise manner.

Theorem 3.1 (Pairwise retrieval error bound)**.** *Under Assumptions A1–A3, for any incorrect adapter $s \neq k$, the pairwise retrieval error satisfies*

$$\Pr\Big(d(\boldsymbol{z}_k, P_{\theta_k}) \ge d(\boldsymbol{z}_s, P_{\theta_s})\Big) \le 2\exp\big(-c\,\mu_{k,s}^2\big) + 2\eta,$$

for some constant $c > 0$, where

$$\mu_{k,s} := KL\Big(Q_k \big\| P_{\theta_s}\Big) - KL\Big(Q_k \big\| P_{\theta_k}\Big) - \epsilon_+ \ge 0$$

and

$$\epsilon_+ := \mathbb{E}\Big[L\big\|\boldsymbol{z}_k - \boldsymbol{z}_s\big\|\Big].$$

As such, the pairwise retrieval error decreases as density modeling error $KL(Q_k\|P_{\theta_k})$ or δ becomes smaller and cross-adapter separation $KL(Q_k\|P_{\theta_k})$ or Δ becomes larger (up to the controllable discrepancy term ϵ_+).

We provide the full proof in Appendix A.2. Taken together, this analysis supports the central hypothesis behind density-guided routing: retrieval becomes reliable when task keys accurately model their own adapter-induced embedding distributions (small δ) and when different adapters induce sufficiently separated embedding distributions (large Δ).

This provides theoretical support for our design choice of **detecting change points and learning corresponding task-specific adapters** and **fitting density models to adapter-specific induced embeddings** to improve retrieval accuracy in task-free continual learning.

4 EMPIRICAL STUDIES

Our empirical study is designed to answer three questions: (i) How does our method perform relative to existing TFCL baselines across datasets, model sizes, and trainable-parameter budgets? (ii) To what extent do the proposed components mitigate catastrophic forgetting? (iii) What are the contributions and computational trade-offs of each component in our framework? We begin by briefly describing the datasets, evaluation protocols, and compared baselines (Section 4.1). We then present the main benchmark comparisons against prior TFCL methods (Section 4.2), followed by analyses of catastrophic forgetting and component effectiveness (Section 4.3). Finally, we report more ablation studies and complexity analyses in Appendices C, G.

4.1 EXPERIMENT SETUP

Datasets. We evaluate HESTIA on four datasets spanning standard and challenging TFCL settings: Split CIFAR-10/Split CIFAR-100 [Krizhevsky et al., 2009], Split ImageNet-R [Hendrycks et al., 2021], and VTAB5T [Zhai et al., 2019]. While Split CIFAR-10/100 provide standard benchmarks, Split ImageNet-R is more challenging due to substantial style variation and high intra-class diversity. To evaluate longer and more realistic streams, we use VTAB5T, a VTAB-based benchmark constructed from five semantically distinct task subsets spanning diverse visual domains. Additional dataset details are provided in Appendix E.

Baselines and Metrics. We compare HESTIA against representative methods, including classical TFCL such as MIR [Aljundi et al., 2019a], ER+GMED [Jin et al., 2021], CoPE [De Lange and Tuytelaars, 2021], CN-DPM [Ye and Bors, 2022a], Dynamic-OCM [Ye and Bors, 2022c], and ODDL [Ye and Bors, 2022b]; and Transformer-based TFCL such as TFDSViT [Ye and Bors, 2024], L2P [Wang et al., 2022c], DualPrompt [Wang et al., 2022b], RanPAC [McDonnell et al., 2023], and SD-LoRA [Wu et al., 2025]. We follow official and common baseline settings, but adopt a task-shared prompt pool for DualPrompt (as in L2P) and provide SD-LoRA with task IDs during training, since neither supports task-free change-point detection. We additionally report joint training as an upper-bound reference without continual learning constraints. For evaluation metrics, we measure and report the *average accuracy* (higher is better) $\mathrm{ACC}_\kappa = (1/\kappa)\sum_{j=1}^{\kappa}\mathrm{acc}\big(j,\kappa\big)$ where $\mathrm{acc}\big(j,\kappa\big)$ denotes model's prediction accuracy on task j after learning task κ; and *forgetting* (lower is better) $\mathrm{F}_\kappa = (1/(\kappa-1))\sum_{\tau=1}^{\kappa-1}\max_{\tau'\in\{1,\ldots,\kappa-1\}}(\mathrm{acc}(\tau,\tau')-\mathrm{acc}(\tau,\kappa))$ where $\mathrm{acc}(\tau,\kappa)$ denotes model's prediction accuracy on task τ after learning task κ. See Appendix E for additional metrics and experiment details.

Table 1: Final average accuracy (%) achieved by HESTIA and other baselines on Split CIFAR10/100 datasets. (*) are results reported from prior work. The best and second-best values are highlighted in **bold** and underline, respectively.

Methods	Split CIFAR10	Split CIFAR100
Upper-bound	99.79 ± 0.05	99.13 ± 0.01
MIR*	42.80 ± 2.22	20.02 ± 1.70
ER + GMED*	34.84 ± 2.20	20.93 ± 1.60
CoPE*	48.92 ± 1.32	21.62 ± 0.69
CN-DPM*	50.26 ± 1.36	28.76 ± 0.57
Dynamic-OCM*	51.27 ± 1.47	29.87 ± 0.69
ODDL*	52.69 ± 0.11	27.21 ± 0.87
TFDSViT*	55.46 ± 1.02	32.86 ± 0.56
RanPAC	73.99 ± 0.37	70.82 ± 0.11
L2P	89.88 ± 0.93	76.43 ± 0.56
DualPrompt	91.76 ± 1.42	78.49 ± 0.49
SD-LoRA	93.70 ± 3.99	83.82 ± 1.14
HESTIA	$\mathbf{98.51 \pm 0.30}$	$\mathbf{89.69 \pm 1.54}$

4.2 MAIN RESULTS

Table 2: Final average accuracy (%) achieved by HESTIA and Transformer-based baselines on Split ImageNet-R and VTAB5T datasets. The best and second-best values are highlighted in **bold** and underline colors, respectively.

Methods	Split ImageNet-R	VTAB5T
Upper-bound	87.97 ± 0.28	92.00 ± 0.79
RanPAC	50.28 ± 0.27	78.61 ± 1.36
L2P	51.68 ± 0.97	62.14 ± 1.21
DualPrompt	55.32 ± 0.48	63.88 ± 0.77
SD-LoRA	43.61 ± 2.41	54.56 ± 0.63
HESTIA	$\mathbf{58.87 \pm 0.65}$	$\mathbf{81.55 \pm 0.78}$

Table 1 reports the **average accuracy** of HESTIA and competing TFCL baselines on the two standard benchmarks, Split CIFAR-10 and Split CIFAR-100. HESTIA consis-

Table 3: Forgetting and number of trainable parameters comparisons between HESTIA and Transformer-based baselines are reported on Split CIFAR10/100, Split ImageNet-R, and VTABB5T.

Methods	Split CIFAR10		Split CIFAR100		Split ImageNet-R		Split VTAB5T	
	Forgetting ↓	TrainParams ↓	Forgetting ↓	TrainParams ↓	Forgetting ↓	TrainParams ↓	Forgetting ↓	TrainParams ↓
RanPAC	13.50 ± 1.43	155,146	6.28 ± 0.52	224,356	7.84 ± 0.95	301,256	5.77 ± 1.03	740,548
L2P	6.27 ± 1.90	330,250	10.94 ± 1.02	322,660	13.31 ± 12.39	322,760	31.34 ± 0.76	381,124
Dual-Prompt	5.69 ± 2.76	1,881,610	8.90 ± 0.41	1,950,820	12.52 ± 0.46	1,098,440	29.52 ± 1.12	1,337,284
SD-LoRA**	6.81 ± 4.69	368,661	9.88 ± 1.29	368,661	20.28 ± 5.96	368,661	31.695 ± 0.91	368,661
HESTIA	**0.04 ± 0.06**	**124,986**	**2.01 ± 0.55**	**144,196**	**4.62 ± 0.96**	**230,096**	**4.35 ± 0.86**	**206,120**

tently achieves the best performance across both datasets. In particular, it outperforms classical TFCL baselines by up to around 46% on Split CIFAR-10 and 60% on Split CIFAR-100, indicating a substantial margin. Compared with Transformer-based baselines, HESTIA also surpasses the current strong baseline SD-LoRA [Wu et al., 2025] by approximately 5% and 6%. More broadly, Transformer-based TFCL methods significantly outperform classical approaches, underscoring the advantages of pretrained Transformer backbones and PEFT-based adaptation strategies.

Under **longer and more challenging data streams**, we compare HESTIA with Transformer-based TFCL baselines on Split ImageNet-R and VTAB5T in Table 2. HESTIA further improves over the SOTA baseline by up to 15% on Split ImageNet-R and by 27% on VTAB5T. These results demonstrate the robustness of HESTIA in more realistic task-free streaming settings with more challenging and longer horizons scenarios. Comparing with offline upper-bound (UB) settings, HESTIA closes a substantial portion of the gap to the UB on Split CIFAR-10 and Split CIFAR-100, despite operating under the strict TFCL protocol (no task identities, no replay, mini-batch streaming). The larger gap observed on Split ImageNet-R and VTAB5T is consistent with their higher diversity and non-stationarity, which pose greater challenges in the task-free streaming setting.

Figure 2 shows the **average-accuracy progression** of HESTIA and Transformer-based baselines on Split CIFAR-10/100 over 5 and 20 continual-learning iterations, respectively (where iteration κ reports average accuracy over classes observed so far). HESTIA achieves the strongest performance across most iterations and maintains a substantially more stability as new classes arrive, while competing methods show a clearer downward trend due to cumulative forgetting. Although HESTIA may start slightly lower in the earliest iterations, it preserves prior knowledge more effectively and finishes with the highest final accuracy. This observation is consistent with our design: linearized adaptation constrains updates around pretrained weights, sacrificing some early accuracies for improved long-term performance.

Table 3 reports the **forgetting** results. HESTIA achieves the lowest forgetting among all compared methods and consistently ranks first across all four datasets, with extremely low forgetting on Split CIFAR-10/100 and substantially reduced forgetting on the remaining benchmarks. On Split

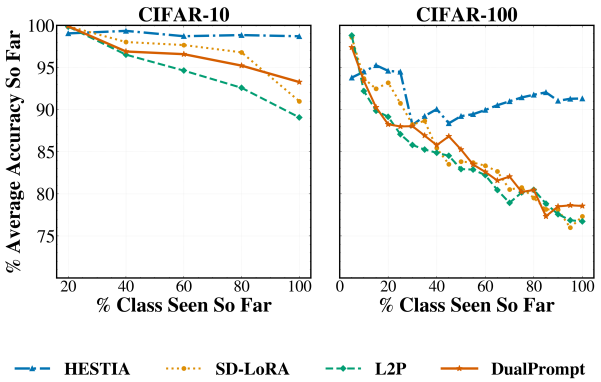

Figure 2: Performance progression on Split CIFAR-10/100. Average accuracy over iterations as new classes arrive, highlighting stable performance.

ImageNet-R and VTAB5T, HESTIA reduces forgetting by up to approximately $5\times$ and $7\times$ relative to strong baselines respectively, highlighting the effectiveness of our design in mitigating catastrophic forgetting in TFCL. To further examine the contribution of each component to overall continual-learning performance and forgetting mitigation, we provide detailed ablation studies in Section 4.3.

4.3 ABLATION STUDIES

Linearized vs. full adaptation under batch streaming. We evaluate whether linearized adaptation reduces *order-induced forgetting* compared with full fine-tuning. To ensure a fair comparison, both methods use the same Transformer backbone, the same streaming protocol, and the same data stream; the only difference is the update rule (linearized update vs. standard gradient-based full adaptation). Figure 3a shows that HESTIA overally exhibits lower forgetting, with the gap larger as the stream becomes longer (i.e., at larger batch indices). This indicates that the proposed linearized solution substantially stabilizes streaming updates. It better preserves previously acquired knowledge and accumulates it more reliably as additional batches are observed.

Impact of task retrieval on performance and forgetting. Without retrieval, the model must represent all previously observed distributions within a single shared parameter space, which increases representation overlap and exacer-

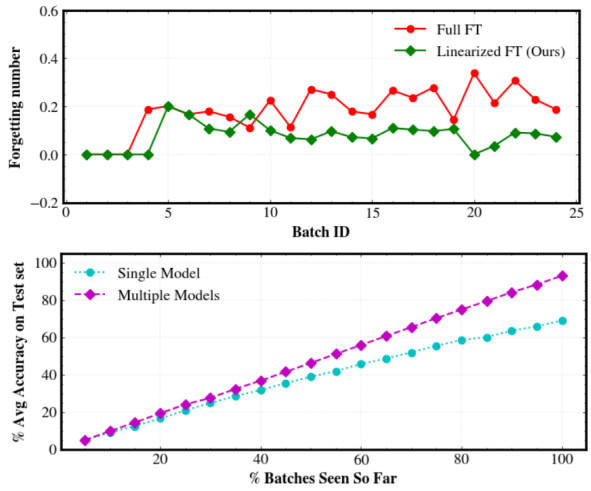

Figure 3: (**Top, a**) Forgetting (↓) for full fine-tuning vs. our linearized fine-tuning using the same Transformer backbone. (**Bottom, b**) Accuracy (↑) of a single linearized model vs. multiple task-specific linearized models on a non-stationary stream. Both experiments are on CIFAR-100.

bates catastrophic forgetting. In particular, as the stream becomes longer and more heterogeneous, a shared solution is more likely to induce cross-distribution interference, leading to degraded performance. By contrast, our routing allows the model to dynamically select the most compatible adapter for each input (Section 2.3), enabling task-specific adaptation. Empirically, HESTIA consistently achieves higher average accuracy and lower forgetting than its retrieval-free variant, shown in Figure 3b and Table 4.

Table 4: Forgetting and performance report of HESTIA with and without task retrieval on multiple datasets: Split CIFAR-10/100, Split ImageNet-R, and VTAB5T.

Datasets	W/o Task Retrieval		W/ Task Retrieval		
	F ↓	Acc ↑	F ↓	Acc ↑	Ret. Acc ↑
CIFAR10	19.51	76.59	0.04	98.51	100
CIFAR100	14.05	67.85	2.01	89.69	97.03
ImageNet-R	7.88	51.14	4.62	58.87	69.45
VTAB5T	26.31	43.72	4.96	82.10	97.48

Component-wise contribution. To further provide ablation experiments on isolating the contributions of the key components in our framework, we report overall performance (%) and retrieval accuracy (%) on 4 datasets with HESTIA's variants including (a) HESTIA without routing (use a single adapter only); (b) HESTIA without linearized fine-tuning (use standard stochastic gradient descent (SGD) [Bottou, 2010]); and (c) a full HESTIA in Table 5. The results indicate that both solution components are independently essential, with consistent gains from each across all four benchmarks. Specifically, the gap between (a) and (c) emphasizes routing benefits under heterogeneous distribution shifts and the gap between (b) and (c) reflects the cumulative benefit

Figure 4: Distribution shift detection on Split CIFAR-100. The solid curve plots routing scores – Eq. 9– over the stream. Ground-truth distribution boundaries are indicated by green markers, while detected distribution shifts produced by the proposed detector are shown by yellow markers.

of order-invariant updates in HESTIA as the stream grows.

Table 5: Comparison of HESTIA variants (Performance (%) / Retrieval Acc (%)) on four benchmarks.

Datasets	(a) w/o routing	(b) w/o lin. FT	(c) Full
CIFAR-10	73.98/NA	87.80/40.00	**98.51/100.00**
CIFAR-100	69.09/NA	83.95/32.34	**89.69/97.03**
ImageNet-R	50.97/NA	49.68/3.13	**58.87/69.45**
VTAB5T	60.22/NA	41.30/18.38	**82.10/97.48**

Performance of the distribution-shift detector. To evaluate our compact change-point detector, we compare the *predicted* change points against the *ground-truth* task boundaries used only to construct the streaming protocol (these boundaries are never provided to the model). As shown in Figures 4 and 7, our detector identifies distribution shifts accurately and with low delay. This result supports the design of our detector: it leverages stabilization of routing-score dynamics (convergence) and then flags sustained mismatches (e.g., consistently high out-of-distribution scores or recurring novel patterns) as evidence of a new distribution.

Change-point detector hyperparameter sensitivity. We study HESTIA's sensitivity to its three main hyperparameters: tolerance k, window size w (Eq. (11)), and number of Gaussian-mixture components K(Section 2.3). Figure 5 reports average accuracy on Split CIFAR-100 as each is varied independently, with the others fixed at default values ($k = 1.3$, $w = 10$, $K = 8$). HESTIA is robust to K across the full tested range but more sensitive to k and w when set too low: small values make the change-point detector overly sensitive, triggering spurious adapter expansion, while large values delay detection and merge distinct distributions; likewise, too few clusters under-model multi-modal embeddings, while too many increase estimation variance under limited samples. The values adopted throughout the

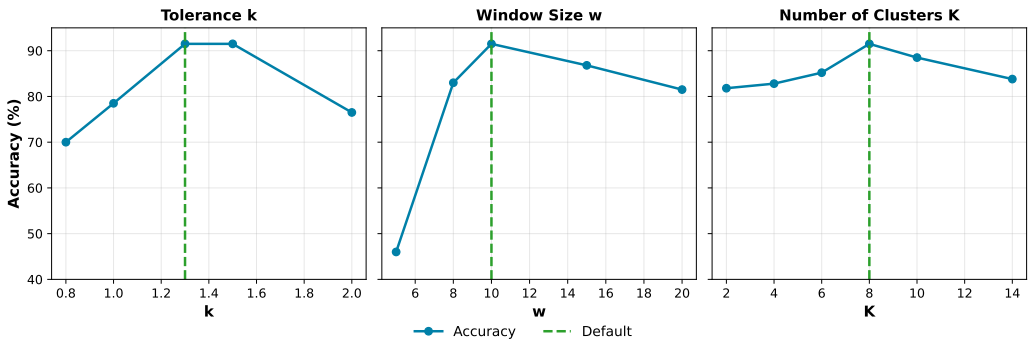

Figure 5: HESTIA performance on Split CIFAR-100 across tolerance k, window size w, and cluster count K. Main-experiment values ($k = 1.3$, $w = 10$, $K = 8$) are near-optimal for each curve.

paper ($k = 1.3$, $w = 10$, $K = 8$) are selected from these sensitivity curves and correspond to the best or near-best performance across the tested ranges.

Additional analyses and ablations. Table 3 summarizes trainable-parameter counts, showing that HESTIA achieves the best overall performance across datasets while using the fewest trainable parameters. We further report additional results on retrieval accuracy and distribution-shift behavior in Appendix G. scalability and efficiency are also evaluated via time and space complexity analyses (Appendices C and F).

5 RELATED WORKS

Task-free continual learning and streaming non-stationarity. Continual learning (CL) studies sequential learning under non-stationary data and the resulting stability–plasticity tension and catastrophic forgetting [McCloskey and Cohen, 1989, French, 1999, Parisi et al., 2019, Van de Ven et al., 2022]. A key distinction across CL formulations is the availability of task information: in *task-incremental* learning, task identity is provided at inference; in *class-incremental* (often termed task-agnostic) learning, task identity is absent at inference but task boundaries are typically available during training; and in *task-free* (streaming) settings, task identities and boundaries are not observed during either training or inference [Van de Ven et al., 2022]. Task-free continual learning (TFCL) is particularly relevant for realistic streams where data arrive in small, temporally correlated batches and latent distribution changes must be inferred online, which amplifies sensitivity to batch ordering and boundary ambiguity [De Lange and Tuytelaars, 2021].

Continual adaptation with pre-trained models. With strong pre-trained backbones, CL increasingly adopts parameter-efficient fine-tuning (PEFT), freezing the backbone while adapting lightweight modules (e.g., prompts or low-rank updates) [Houlsby et al., 2019, Lester et al., 2021, Li and Liang, 2021, Hu et al., 2022, Liu et al., 2022]. Prompt-based CFT methods such as L2P and DualPrompt learn prompt pools with instance-wise selection, enabling inference without task IDs [Wang et al., 2022c,b], and CODA-Prompt further improves end-to-end prompt composition and selection [Smith et al., 2023]; complementary prompt paradigms address domain/semantic shifts [Wang et al., 2022a, Kim et al., 2023]. PEFT with low-rank updates has also been explored for continual adaptation, including interference-aware designs [Liang and Li, 2024, Wei et al., 2025, Wu et al., 2025], while prototype-accumulation strategies such as RanPAC reduce forgetting by minimizing iterative feature updates [McDonnell et al., 2023]. Despite these advances, these methods remain fundamental challenges related to catatrophic forgetting because their adapted parameters (or retrieval/selection components) are optimized incrementally by local gradient steps or are assigned fixed rules. Additional related work is provided in Appendix I.

6 CONCLUSION

This work introduces a complementary view of task-free continual learning that jointly addresses two major sources of forgetting: order-induced drift within stationary distribution and interference under distribution shift. By combining order-invariant linearized adaptation with density-guided adapter routing, our framework improves performance, reduces catastrophic forgetting, and offers retrieval reliability in task-free streams. We also provide a theoretical characterization of retrieval error in terms of density estimation quality and the separability of adapter-induced embedding distributions. Extensive experiments and ablations show that our method consistently improves continual learning performance and yields the lowest forgetting among strong baselines. Our approach relies on a local linear approximation for adaptation and Gaussian-mixture task keys for routing, which may be less effective under large parameter shifts or highly overlapping, heavy-tailed embedding distributions; we discuss these limitations and concrete future directions in Appendix J.

ACKNOWLEDGEMENT

This work utilized GPU compute resources at SDSC and ACES through allocation CIS230391 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) program [Boerner et al., 2023], which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. Trong Nghia Hoang is supported by the National Science Foundation CAREER Award IIS-2544071.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019b.

Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, 2023.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, 2010.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

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 *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34:29193–29205, 2021.

Doyoung Kim, Susik Yoon, Dongmin Park, Youngjun Lee, Hwanjun Song, Jihwan Bang, and Jae-Gil Lee. One size fits all for semantic shifts: Adaptive prompt tuning for continual learning. *arXiv preprint arXiv:2311.12048*, 2023.

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

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

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Tian Yu Liu and Stefano Soatto. Tangent model composition for ensembling and continual fine-tuning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, 2023.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. 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. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

Nicolas Michel, Maorong Wang, Jiangpeng He, and Toshihiko Yamasaki. From offline to online memory-free and task-free continual learning via fine-grained hypergradients. *arXiv preprint arXiv:2502.18762*, 2025.

Saleh Momeni, Sahisnu Mazumder, and Bing Liu. Continual learning using a kernel-based method over foundation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Saleh Momeni, Changnan Xiao, and Bing Liu. Anacp: Toward upper-bound continual learning via analytic contrastive projection. *Advances in Neural Information Processing Systems*, 38:72923–72944, 2026.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113: 54–71, 2019.

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

Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in neural information processing systems*, 32, 2019.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

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

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35, 2022a.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022c.

Xiwen Wei, Guihong Li, and Radu Marculescu. Online-lora: Task-free online continual learning via low rank adaptation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

Fei Ye and Adrian G Bors. Continual variational autoencoder learning via online cooperative memorization. In *European Conference on Computer Vision*, 2022a.

Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *Advances in Neural Information Processing Systems*, 35:23675–23688, 2022b.

Fei Ye and Adrian G Bors. Continual variational autoencoder learning via online cooperative memorization. In *European Conference on Computer Vision*, 2022c.

Fei Ye and Adrian G Bors. Task-free dynamic sparse vision transformer for continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Haoran Zhu, Maryam Majzoubi, Arihant Jain, and Anna Choromanska. Tame: Task agnostic continual learning using multiple experts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. Gacl: Exemplar-free generalized analytic continual learning. *Advances in neural information processing systems*, 37: 83024–83047, 2024.

Supplementary Material: Proofs and Additional Experiments

Hang Thi-Thuy Le[1] **Nam-Quan Nguyen**[2] **Lam-Huy Nguyen**[2] **Dien Dinh**[2] **Minh Hoang**[3] **Trong Nghia Hoang**[1]

[1]Washington State University, Pullman, Washington, USA
[2]University of Science, VNU-HCM, Ho Chi Minh City, Vietnam
[3]Princeton University, Princeton, New Jersey, USA

A PROOFS

A.1 DERIVATION OF THE LINEARIZED ADAPTATION SOLUTION-EQ 4

We consider a Transformer-based model $h(\cdot;\boldsymbol{\omega})$ initialized from pretrained weights $\boldsymbol{\omega}_0$. During continual adaptation, we update a small set of trainable parameters (e.g., adapters) by $\Delta\boldsymbol{\omega}$, yielding predictions based on $\boldsymbol{\omega} = \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega}$. For an input $\boldsymbol{x}$, we write the (pre-softmax) logits as $h(\boldsymbol{x};\boldsymbol{\omega})$ and the predicted class probabilities as

$$\hat{\boldsymbol{p}}(\boldsymbol{x};\boldsymbol{\omega}) = \mathrm{softmax}(h(\boldsymbol{x};\omega)) \in \mathbb{R}^C.$$

Cross-entropy loss. For a labeled example $(\boldsymbol{x}, y)$, the cross-entropy loss is

$$\ell_{\mathrm{ce}}(\boldsymbol{\omega};\boldsymbol{x},y) = -\sum_{c=1}^{C} \mathbb{I}(y=c)\log \hat{p}_c(\boldsymbol{x};\boldsymbol{\omega}) = -\log \hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}). \tag{13}$$

Global risk objective. Let P_i denote the empirical distribution of all samples observed up to time i (i.e., across all previously seen batches). The global risk under cross-entropy is

$$\mathcal{R}(\boldsymbol{\omega}, P_i) \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim P_i}\big[\ell_{\mathrm{ce}}(\boldsymbol{\omega};\boldsymbol{x},y)\big] = \frac{1}{N}\sum_{(\boldsymbol{x},y)\sim P_i} -\log \hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}), \tag{14}$$

where N is the number of samples accumulated so far.

Loss approximation near a strong pretrained model. Since we start from a strong pretrained model, we operate in a local regime where many samples have high confidence $\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}_0) \approx 1$. Using the second-order Taylor expansion of $-\log a$ around $a = 1$,

$$-\log a \;=\; (1-a) + \frac{1}{2}(1-a)^2 + o\big((1-a)^2\big),$$

remove higher-order residuals (which do not affect the ridge closed form up to constants), we obtain the squared-loss surrogate

$$-\log \hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}) \;\approx\; (1-a) + \frac{1}{2}\big(1-\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega})\big)^2 = \frac{1}{2}\left(2-\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega})\right)^2 - \frac{1}{2}. \tag{15}$$

Substituting Eq. (15) into Eq. (14) leads to the following approximate global objective

$$\boldsymbol{\omega}^* \;\approx\; \arg\min_{\omega} \frac{1}{N}\sum_{(\boldsymbol{x},y)\sim P_i} \frac{1}{2}\big(2-\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega})\big)^2. \tag{16}$$

Linearization around ω_0. Under the linearized fine-tuning regime, we assume the class probability $\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega})$ varies approximately linearly in a neighborhood of ω_0. For a small update $\Delta\boldsymbol{\omega}$, a first-order Taylor approximation gives

$$\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}_0+\Delta\boldsymbol{\omega}) \approx \hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}_0) + J(\boldsymbol{x})\,\Delta\boldsymbol{\omega}, \tag{17}$$

where $J(\boldsymbol{x}) := \nabla_\omega \hat{p}_y(\boldsymbol{x};\boldsymbol{\omega})\big|_{\omega=\omega_0}$ is the Jacobian row vector (or matrix, depending on parameterization) evaluated at ω_0.

Plugging Eq. (17) into Eq. (16) yields a regularized least-squares problem in $\Delta\omega$:

$$\Delta\boldsymbol{\omega}^* = \arg\min_{\Delta\omega} \frac{1}{N}\sum_{(\boldsymbol{x},y)\sim P_i} \frac{1}{2}\Big(J(\boldsymbol{x})\Delta\boldsymbol{\omega} - \big(2-\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}_0)\big)\Big)^2 + \frac{\lambda}{2}\|\Delta\boldsymbol{\omega}\|_2^2, \tag{18}$$

where $\lambda > 0$ is the ridge regularization coefficient. The resulting adapted parameters are $\boldsymbol{\omega}^* = \boldsymbol{\omega}_0 + \Delta\boldsymbol{\omega}^*$.

Closed-form ridge solution and batch decomposition. Assume the N samples are presented as a stream of batches $\{D_t\}_{t=1}^T$. For each batch D_t, define the batch Jacobian matrix and target vector:

$$\boldsymbol{A}_t := \big[J(\boldsymbol{x})\big]_{(\boldsymbol{x},y)\in D_t} \in \mathbb{R}^{n_t\times p}, \qquad \boldsymbol{r}_t := \big[2-\hat{p}_y(\boldsymbol{x};\boldsymbol{\omega}_0)\big]_{(\boldsymbol{x},y)\in D_t} \in \mathbb{R}^{n_t}, \tag{19}$$

where p is the number of trainable parameters and $n_t = |D_t|$.

Then Eq. (18) has the standard ridge closed form:

$$\Delta\boldsymbol{\omega}^* = \left(\sum_{t=1}^T \boldsymbol{A}_t^\top \boldsymbol{A}_t + \lambda \boldsymbol{I}\right)^{-1}\left(\sum_{t=1}^T \boldsymbol{A}_t^\top \boldsymbol{r}_t\right). \tag{20}$$

This expression motivates maintaining accumulated sufficient statistics

$$\boldsymbol{G} \leftarrow \boldsymbol{G} + \boldsymbol{A}_t^\top \boldsymbol{A}_t, \qquad \boldsymbol{g} \leftarrow \boldsymbol{g} + \boldsymbol{A}_t^\top \boldsymbol{r}_t,$$

so that $\Delta\boldsymbol{\omega}^* = (\boldsymbol{G}+\lambda\boldsymbol{I})^{-1}\boldsymbol{g}$ can be computed without storing past batches. This is the key to obtaining an order-invariant update within a locally stationary segment by aggregating batch-wise statistics.

A.2 PROOF OF RETRIEVAL ERROR BOUND

A.2.1 Notation and Assumptions

We follow the setting in Section 3. Let $x \sim \mathcal{D}_k$ be a test sample drawn from the k-th underlying data distribution, and let h_{ω_k} denote the embedding function induced by the linearized optimal parameters for that distribution. The corresponding embedding distribution is the pushforward

$$Q_k := (h_{\omega_k})_{\#}\mathcal{D}_k.$$

For each distribution k, we fit a density-based model (e.g., a Gaussian mixture) P_{θ_k} to embeddings from Q_k. Given a test input x, we compute adapter-specific embeddings $\boldsymbol{z}_j = h_{\omega_j}(x)$ and retrieve the adapter with the smallest negative log-likelihood:

$$\hat{k}(x) = \arg\min_j d(\boldsymbol{z}_j, P_{\theta_j}), \quad d(\boldsymbol{z}, P_\theta) := -\log p_\theta(\boldsymbol{z}). \tag{21}$$

We make the following assumptions.

A1. Gaussian mixture model fitting quality and inter-adapter separation (KL margin). For each k, the fitted GMM P_{θ_k} approximates the induced embedding distribution Q_k with bounded forward KL divergence:

$$\mathrm{KL}(Q_k \,\|\, P_{\theta_k}) \le \delta_k \le \delta. \tag{22}$$

Moreover, for every $s \neq k$, there exists a separation margin $\Delta_{k,s}$ such that

$$\mathrm{KL}(Q_k \,\|\, P_{\theta_k}) \le \delta_k \le \Delta_{k,s} \le \mathrm{KL}(Q_k \,\|\, P_{\theta_s}). \tag{23}$$

Let $\Delta := \min_k \min_{s \neq k} \Delta_{k,s}$. Then $\Delta - \delta \geq 0$.

This assumption captures two effects: (i) each task key fits its own adapter-induced embedding distribution well (small δ), and (ii) mismatched task keys are separated in KL under the correct embedding distribution (large Δ).

A2. Local smoothness of log-likelihood. For each GMM P_{θ_s}, the log-density $\log p_{\theta_s}(z)$ is L-Lipschitz on a high-probability region $B \subset \mathbb{R}^d$, and for any relevant adapter pair (k, s),

$$\Pr_{z_k \sim Q_k}(\boldsymbol{z}_k \in B) \geq 1 - \eta, \qquad \Pr(\boldsymbol{z}_s \in B) \geq 1 - \eta, \tag{24}$$

where $\boldsymbol{z}_s = h_{\omega_s}(\boldsymbol{x})$ for $\boldsymbol{x} \sim \mathcal{D}_k$. This is mild for Gaussian mixtures with bounded covariances on bounded embedding regions.

A3. Concentration of likelihood ratios and cross-adapter discrepancy. For $\boldsymbol{x} \sim \mathcal{D}_k$, define

$$\tilde{T}(\boldsymbol{z}_k) := \log p_{\theta_k}(\boldsymbol{z}_k) - \log p_{\theta_s}(\boldsymbol{z}_k), \qquad \boldsymbol{z}_k = h_{\omega_k}(\boldsymbol{x}), \tag{25}$$

and let $\boldsymbol{D} := \boldsymbol{z}_k - \boldsymbol{z}_s$ with $\boldsymbol{z}_s = h_{\omega_s}(x)$. Assume the centered variables

$$\tilde{T} - \mathbb{E}\big[\tilde{T}\big] \quad \text{and} \quad L\big\|\boldsymbol{D}\big\| - \mathbb{E}\Big[L\big\|\boldsymbol{D}\big\|\Big]$$

admit sub-Gaussian tails. This reflects the fact that all adapters share a pretrained backbone and differ only by lightweight updates.

A.2.2 Proof of Theorem 1

Fix a true distribution index k and an incorrect adapter $s \neq k$. Retrieval incorrectly prefers s over k if

$$d(\boldsymbol{z}_k, P_{\theta_k}) \geq d(\boldsymbol{z}_s, P_{\theta_s}),$$

equivalently,

$$T := \log p_{\theta_k}(\boldsymbol{z}_k) - \log p_{\theta_s}(\boldsymbol{z}_s) \leq 0. \tag{26}$$

To compare the two log-likelihood terms at different embeddings, define the same-point likelihood ratio

$$\tilde{T} := \log p_{\theta_k}(\boldsymbol{z}_k) - \log p_{\theta_s}(\boldsymbol{z}_k). \tag{27}$$

On the event $\{\boldsymbol{z}_k \in B,\ \boldsymbol{z}_s \in B\}$, Assumption A2 implies

$$\log p_{\theta_s}(\boldsymbol{z}_s) \leq \log p_{\theta_s}(\boldsymbol{z}_k) + L\big\|\boldsymbol{z}_k - \boldsymbol{z}_s\big\|, \tag{28}$$

hence

$$T \geq \tilde{T} - L\big\|\boldsymbol{D}\big\|. \tag{29}$$

Therefore,

$$\begin{aligned} \Pr(T \leq 0) \leq& \Pr(\tilde{T} - L\big\|\boldsymbol{D}\big\| \leq 0) + \Pr(\boldsymbol{z}_k \notin B) + \Pr(\boldsymbol{z}_s \notin B) \\ \leq& \Pr(\tilde{T} - L\big\|\boldsymbol{D}\big\| \leq 0) + 2\eta. \end{aligned} \tag{30}$$

It remains to bound $\Pr(\tilde{T} - L\big\|\boldsymbol{D}\big\| \leq 0)$.

Step 1: Lower bound the mean of $\tilde{T}$. Since $\boldsymbol{z} \sim Q_k$,

$$\begin{aligned} \mathbb{E}\big[\tilde{T}\big] =& \mathbb{E}_{z \sim Q_k}\Big[\log p_{\theta_k}(\boldsymbol{z}) - \log p_{\theta_s}(\boldsymbol{z})\Big] \\ =& \mathrm{KL}\Big(Q_k\big\|P_{\theta_s}\Big) - \mathrm{KL}\Big(Q_k\big\|P_{\theta_k}\Big). \end{aligned} \tag{31}$$

By Assumption A1,

$$\mathbb{E}\big[\tilde{T}\big] \geq \Delta_{k,s} - \delta_k \geq \Delta - \delta. \tag{32}$$

Step 2: Define the effective margin. Let

$$\epsilon_+ := \mathbb{E}\Big[L\|\boldsymbol{D}\|\Big], \qquad \mu_{k,s} := \mathbb{E}[\tilde{T}] - \epsilon_+. \tag{33}$$

Using Eq. (32), we obtain the lower bound

$$\mu_{k,s} \geq (\Delta - \delta) - \epsilon_+. \tag{34}$$

The retrieval bound is informative when $\mu_{k,s} \geq 0$ (or more conservatively when $(\Delta - \delta) - \epsilon_+ > 0$).

Step 3: Concentration bound. If $\tilde{T} - L\|\boldsymbol{D}\| \leq 0$, then

$$\Big(\tilde{T} - \mathbb{E}[\tilde{T}]\Big) - \Big(L\|\boldsymbol{D}\| - \mathbb{E}\Big[L\|\boldsymbol{D}\|\Big]\Big) \leq -\mu_{k,s}. \tag{35}$$

By a union bound, for any $\mu_{k,s} \geq 0$,

$$\left\{\tilde{T} - L\|\boldsymbol{D}\| \leq 0\right\} \subseteq \left\{\tilde{T} - \mathbb{E}[\tilde{T}] \leq -\frac{\mu_{k,s}}{2}\right\} \cup \left\{L\|\boldsymbol{D}\| - \mathbb{E}\Big[L\|\boldsymbol{D}\|\Big] \geq \frac{\mu_{k,s}}{2}\right\}. \tag{36}$$

Hence,

$$\Pr\left(\tilde{T} - L\|\boldsymbol{D}\| \leq 0\right) \leq \Pr\left(\tilde{T} - \mathbb{E}[\tilde{T}] \leq -\frac{\mu_{k,s}}{2}\right) + \Pr\left(L\|\boldsymbol{D}\| - \mathbb{E}\Big[L\|\boldsymbol{D}\|\Big] \geq \frac{\mu_{k,s}}{2}\right). \tag{37}$$

Under Assumption A3 (sub-Gaussian tails), there exist constants $c_1, c_2 > 0$ such that

$$\Pr\Big(\tilde{T} - L\|\boldsymbol{D}\| \leq 0\Big) \leq \exp\Big(-c_1\mu_{k,s}^2\Big) + \exp\Big(-c_2\mu_{k,s}^2\Big) \leq 2\exp\Big(-c\mu_{k,s}^2\Big), \tag{38}$$

for $c := \min(c_1, c_2)$.

Combining with Eq. (30), we obtain

$$\Pr(T \leq 0) \leq 2\exp\Big(-c\mu_{k,s}^2\Big) + 2\eta. \tag{39}$$

This proves the pairwise retrieval error bound in Theorem 3.1.

Interpretation. The bound highlights two complementary factors controlling retrieval reliability: (i) *density modeling quality* (small δ), which increases the self-likelihood of embeddings under the correct task key, and (ii) *cross-adapter separability* (large Δ), which reduces the likelihood assigned by mismatched task keys. The term $\epsilon_+ = \mathbb{E}\Big[L\|\boldsymbol{D}\|\Big]$ captures cross-adapter embedding discrepancy evaluated through the smoothness of the density model, while η accounts for rare atypical embeddings outside the high-probability region.

Remark (controlling the discrepancy term). Under the first-order linearization of the embedding map around pretrained weights $\boldsymbol{\omega}_0$,

$$h_\omega(x) \approx h_{\omega_0}(x) + J_h(x;\boldsymbol{\omega}_0)(\boldsymbol{\omega} - \boldsymbol{\omega}_0),$$

the cross-adapter embedding discrepancy satisfies

$$\|\boldsymbol{z}_k - \boldsymbol{z}_s\| \approx \|J_h(\boldsymbol{x};\boldsymbol{\omega}_0)(\boldsymbol{\omega}_k - \boldsymbol{\omega}_s)\| \leq \|J_h(\boldsymbol{x};\boldsymbol{\omega}_0)\|_{\mathrm{op}}\,\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_s\|.$$

Consequently,

$$\epsilon_+ = \mathbb{E}\Big[L\big\|\boldsymbol{z}_k - \boldsymbol{z}_s\big\|\Big] \leq L\,\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_s\|\,\mathbb{E}\Big[\big\|J_h(\boldsymbol{x};\boldsymbol{\omega}_0)\big\|_{\mathrm{op}}\Big]$$

(up to higher-order terms). This shows that, in the PEFT regime with small adapter updates and bounded embedding Jacobian, the discrepancy penalty ϵ_+ remains controlled.

Algorithm 2 Inference Algorithm

input Backbone $h(\cdot;\omega_0)$ and classification head $f(\cdot)$; bank $\mathcal{B} = \{(\Delta\omega^{(t)}, \theta_t)\}_{t=1}^{T}$; test input $\boldsymbol{x}_*$
output Prediction $\hat{y}$

for $t = 1, \ldots, T$ **do**
  Compute adapter-specific embedding $\boldsymbol{z}_t \leftarrow h(\boldsymbol{x}_*; \omega_0 + \Delta\omega^{(t)})$ (Eq. (8))
  Compute routing score $s_t \leftarrow S_t(\boldsymbol{z}_t)$ (Eq. (9))
end for
Select $t_* \leftarrow \arg\min_t S_t$ (Eq. (10))
Predict $y_* \leftarrow \arg\max_c f\big(h(\boldsymbol{x}_*; \omega_0 + \Delta\omega^{(t_*)})\big)$
return $\hat{y}$

B INFERENCE ALGORITHM

C COMPLEXITY ANALYSIS

We analyze the computational complexity of HESTIA Training (Alg. 1) and HESTIA Inference (Alg. 2).

Let N and B be the number of streaming training batches and the batch size. Let T be the number of detected distributions stored in the bank $\mathcal{B}$ ($T \leq N$). The Gaussian mixture model is a diagonal-GMM with K clusters over d-dimensional features. The PEFT adapters ϕ are partitioned into L independent blocks with sizes $\{d_\ell\}_{\ell=1}^{L}$. Define

$$P \triangleq \sum_{\ell=1}^{L} d_\ell, \qquad D_2 \triangleq \sum_{\ell=1}^{L} d_\ell^2, \qquad D_3 \triangleq \sum_{\ell=1}^{L} d_\ell^3, \qquad d_{\max} \triangleq \max_{\ell} d_\ell. \tag{40}$$

Let E be the number of training epochs, $\mathcal{C}_{\text{fwd}}$ for one forward pass of the frozen backbone on a batch, and $\mathcal{C}_{\text{bwd},\phi}$ for one reverse-mode gradient evaluation w.r.t. ϕ on a batch (to compute Jacobian).

C.1 COMPLEXITY OF HESTIA TRAINING

The outer loop in Alg. 1 runs for N batches. We analyze the cost per batch.

1. **Head training**. For E epochs, the cost is upper-bounded by $\mathcal{O}(E\mathcal{C}_{\text{fwd}})$.
2. **Gaussian model update and shift scoring**. Computing diagonal Mahalanobis scores to K clusters over d-dimensional features costs $\mathcal{O}(BKd)$ plus one feature extraction $\mathcal{O}(\mathcal{C}_{\text{fwd}})$.
3. **Jacobian computation**. We compute $J_b = \nabla_\phi \hat{p}_y(x^b; \omega_0)$ yielding $\mathcal{O}(B \cdot \mathcal{C}_{\text{bwd},\phi})$.
4. **Build normal equations**. For each block ℓ, forming $A_\ell^\top A_\ell$ with $A_\ell \in \mathbb{R}^{B \times d_\ell}$ costs $\mathcal{O}(Bd_\ell^2)$. Summing over blocks is $\mathcal{O}(BD_2)$.
5. **Soft-target refinement + ridge solves**. Each iteration computes $A_\ell^\top z$ in $\mathcal{O}(Bd_\ell)$ and solves a dense $d_\ell \times d_\ell$ linear system in $\mathcal{O}(d_\ell^3)$. Summed over blocks, one iteration costs $\mathcal{O}(BP + D_3)$, and over E iterations is $\mathcal{O}(E(BP + D_3))$
6. **Global accumulation and finalization**. Updating global statistics adds $\mathcal{O}(D_2)$ and computing the current global ridge solution adds $\mathcal{O}(D_3)$ per batch. When a shift is detected, we store one additional ridge solution $\mathcal{O}(D_3)$; over the entire stream this contributes $\mathcal{O}(TD_3) \leq \mathcal{O}(ND_3)$ and is thus dominated by the per-batch terms.

Combining the above, the per-batch complexity is upper-bounded by

$$\mathcal{O}(E\mathcal{C}_{\text{fwd}} + \mathcal{C}_{\text{fwd}} + BKd + B\mathcal{C}_{\text{bwd},\phi} + BD_2 + E(BP + D_3) + (D_2 + D_3))\,, \tag{41}$$

and the total training complexity is

$$\mathcal{O}\Big(N\Big[B\big(\mathcal{C}_{\text{bwd},\phi} + D_2 + Kd\big) + E(BP + D_3 + \mathcal{C}_{\text{fwd}})\Big]\Big) \tag{42}$$

C.2 COMPLEXITY OF HESTIA INFERENCE

For each test input $x^\star$, inference performs: (i) one feature extraction $\mathcal{O}(\mathcal{C}_{\text{fwd}}(1))$, (ii) retrieval by scoring against T adapters, each costing $\mathcal{O}(Kd)$, and (iii) one forward pass under the retrieved PEFT update.

Thus, per-example inference complexity is $\mathcal{O}(TKd + \mathcal{C}_{\text{fwd}}(1))$. The retrieval overhead scales linearly with the bank size T and the number of clusters K.

D PRACTICAL IMPLEMENTATION WITH PARAMETER-EFFICIENT FINE-TUNING

Directly computing and storing Jacobians with respect to all model parameters is infeasible for large pretrained networks. We therefore instantiate the linearized fine-tuning framework on a small set of parameter-efficient fine-tuning (PEFT) parameters, while keeping the backbone frozen. We instantiate our framework using IA^3 because its scaling parameters are deterministically initialized to one, providing a natural linearization point around an identity transformation. This property ensures that the first-order Taylor approximation in Section 2.2 is well-conditioned from the start of training. More generally, our framework is compatible with parameter-efficient tuning methods whose trainable parameters are initialized identity and operate through smooth transformations. While we focus on IA^3 in this work, exploring other PEFT instantiations such as LoRA or adapters is an interesting direction for future study.

Parameterization. Let the pretrained backbone be $h(x; \omega_0)$. We introduce a lightweight set of trainable parameters $\phi \in \omega$ through an IA^3-style re-scaling mechanism applied to attention and feed-forward activations. Only ϕ is optimized during continual learning, while ω_0 remains fixed. Under this design, the linearization in Section 2.2 is performed with respect to ϕ, and all Jacobians are computed as $J(x) = \nabla_\phi \hat{p}_y(x; \omega_0, \phi)$. This reduces both memory and computational cost.

Blockwise Jacobian Computation. The PEFT parameters ϕ are naturally partitioned into independent blocks corresponding to different transformer layers. We exploit this structure and compute Jacobians blockwise. Specifically, for each block ℓ, we accumulate $G_\ell \leftarrow G_\ell + \boldsymbol{A}_\ell^\top \boldsymbol{A}_\ell; g_\ell \leftarrow g_\ell + \boldsymbol{A}_\ell^\top \tilde{\boldsymbol{b}}$. where $\boldsymbol{A}_\ell$ denotes the Jacobian restricted to block ℓ. The global solution is obtained by solving a small ridge regression problem per block: $\Delta\phi_\ell^* = \big(G_\ell + \lambda I\big)^{-1} g_\ell$. Blockwise solving avoids forming a large dense system and enables scalable training.

Soft-Target Construction. From Section A.1, the squared-loss formulation requires targets of the form $b(x, y) = 2 - \hat{\boldsymbol{p}}_y(x; \omega_0)$. However, directly using these values may lead to unstable optimization, therefore we construct soft targets $\tilde{b}$ by iteratively refining the current prediction $\hat{p}$ toward b using a simple fixed-point update:

$$\hat{p}^{(j+1)} = (1 - \eta)\hat{p}^{(j)} + \eta b, \tag{43}$$

where $\eta \in (0, 1)$. After M iterations, we set $\tilde{b} = \hat{p}^{(M)}$. For each incoming batch, we compute predictions, construct soft targets, evaluate blockwise Jacobians, and update the corresponding sufficient statistics. The complete training procedure is summarized in Algorithm 1. This procedure provides numerically stable targets while preserving the squared-loss objective.

Classification Head. We followed the best practice from TMC [Liu and Soatto, 2023] to simply fine-tune the classification head using gradient-based optimization before performing our linearized adaptation solution on IA^3 parameters.

E DATASETS, HYPERPARAMETERS, AND PRE-TRAINED MODELS

Datasets and streaming protocols. We evaluate HESTIA on four task-free continual learning benchmarks: Split CIFAR-10, Split CIFAR-100 [Krizhevsky et al., 2009], Split ImageNet-R [Hendrycks et al., 2021], and VTAB5T [Zhai et al., 2019]. Each benchmark defines a stream order via a sequence of latent segments, but task identities and task boundaries are not provided to the learner during training or inference (TFCL protocol).

Split CIFAR-10 is streamed as 5 segments with two disjoint classes per segment. Split CIFAR-100 is streamed as 20 segments with five disjoint classes per segment. Split ImageNet-R contains 200 classes and is streamed using a fixed class order, split into sequential segments as commonly adopted in prior work. VTAB5T forms a substantially longer and more diverse stream by concatenating five semantically distinct VTAB subsets; we treat the resulting stream as task-free and do not reveal subset boundaries.

Evaluation metrics. We report (i) final average accuracy (AvgAcc), (ii) average forgetting, (iii) the number of additional parameters beyond the frozen backbone (TrainParams), and (iv) memory usage during training. AvgAcc is the mean

Table 6: Dataset statistics.

CL Datasets	# train samples	# val/test samples	# classes
Split CIFAR10	50,000	10,000	10
Split CIFAR100	50,000	10,000	100
Split ImageNet-R	24,000	6,000	200
VTAB5T	321,406	47,585	196

classification accuracy over all classes after processing the full stream. Forgetting is computed as the mean drop from the best historical accuracy on each latent segment to the final accuracy after training (definitions in Section 4.1). TrainParams counts all non-backbone parameters used by each method; for HESTIA, this includes both PEFT parameters and the density-model parameters used for routing (Section H).

Pre-trained backbone and configuration. All experiments use ViT-B/16 pre-trained on ImageNet-21k and fine-tuned on ImageNet-1K. The ViT backbone is frozen throughout continual learning. Inputs are resized to 224×224 and normalized using ImageNet statistics.

We follow the PEFT setup in Section D. The classification head is warm-started with AdamW and then the IA^3 parameters are adapted using our closed-form linearized updates. We use batch size 16 and dataset-specific learning rates for the head: 1×10^{-3} for Split CIFAR-10/100 and 5×10^{-4} for Split ImageNet-R and VTAB5T. All other components of HESTIA (ridge accumulation, density updates, and change-point detection) run online on the stream without storing past batches.

Hyperparameters. Hyperparameters are selected using a small validation split drawn from the training stream without using task labels or task boundaries. Across datasets, we keep the majority of hyperparameters fixed and adjust only the change-detection and density-model capacity to match stream difficulty. Table 7 summarizes the dataset-specific density and change-point hyperparameters.

Table 7: Signature learning hyperparameters used across datasets.

Datasets	#Clusters K	Window w
Split CIFAR-10	8	20
Split CIFAR-100	8	10
Split ImageNet-R	14	5
VTAB 5T	10	10

Computational Resources. All experiments are conducted on NVIDIA A40 GPUs with 48 GB of memory. We also report total memory usage to provide insight into the computational efficiency of HESTIA in Section F.

F MEMORY COST

We analyze the memory overhead incurred by HESTIA during continual training. Since the pretrained ViT backbone remains frozen, memory growth originates only from: (i) stored PEFT solutions, (ii) Gaussian mixture model statistics used for distribution retrieval, and (iii) global ridge accumulators maintained for closed-form updates.

Stored PEFT parameters. Each detected distribution contributes one stored PEFT solution. Using IA^3 adaptation, each transformer layer maintains three learned scaling vectors: $\ell_k \in \mathbb{R}^{768}$, $\ell_v \in \mathbb{R}^{768}$, and $\ell_{ff} \in \mathbb{R}^{3072}$. Thus, the total number of trainable parameters per distribution is

$$P = \sum_{\ell=1}^{L} (d_k + d_v + d_{ff}). \tag{44}$$

For ViT-B/16 ($L = 12$), this yields

$$P = 12 \times (768 + 768 + 3072) = 55{,}296. \tag{45}$$

Assuming 32-bit floating-point storage (4 bytes per parameter), the memory cost per stored distribution is

$$M_{\mathrm{PEFT}} = 4P \approx 0.21\,(\mathrm{MB}). \tag{46}$$

If T distributions are detected, the total memory required to store PEFT solutions scales linearly:

$$M_{\mathrm{PEFT,total}} = 0.21T\ (\mathrm{MB}). \tag{47}$$

Global ridge statistics. HESTIA maintains blockwise second-order accumulators,

$$\{A_\ell^\top A_\ell\}_{\ell=1}^{L}, \quad \{A_\ell^\top z\}_{\ell=1}^{L}, \tag{48}$$

which require storing

$$D_2 = \sum_{\ell=1}^{L} \left(d_k^2 + d_v^2 + d_{ff}^2\right) \tag{49}$$

floating-point values.

For ViT-B/16 ($L = 12$), each layer contains IA^3 vectors of dimensions $d_k = 768$, $d_v = 768$, and $d_{ff} = 3072$. Therefore,

$$D_2 = 12 \times \left(768^2 + 768^2 + 3072^2\right). \tag{50}$$

Since $768^2 = 589{,}824$ and $3072^2 = 9{,}437{,}184$, we obtain

$$D_2 = 12 \times 10{,}616{,}832 = 127{,}401{,}984. \tag{51}$$

Assuming FP32 storage (4 bytes per value), this corresponds to

$$M_{\mathrm{ridge}} = 127{,}401{,}984 \times 4 \approx 509.6\,\mathrm{MB}. \tag{52}$$

Importantly, this component is *constant* and does not grow with the number of detected distributions T.

Total memory growth. Combining all components, the total memory after observing T distributions is

$$M_{\mathrm{total}}(T) = 509.6 + 0.26T\ \mathrm{MB}. \tag{53}$$

Even for $T = 100$ detected distributions:

$$M_{\mathrm{total}}(100) \approx 535.6\,\mathrm{MB}, \tag{54}$$

which remains manageable on modern GPUs.

Notably, the total memory reaches approximately 1GB only when

$$T \approx 1886. \tag{55}$$

Memory growth in HESTIA is therefore linear in the number of detected distributions, with a small proportionality constant (approximately 0.26MB per distribution under CIFAR settings).

G ADDITIONAL RESULTS ON RETRIEVAL ACCURACY AND DISTRIBUTION SHIFT.

Retrieval Accuracy. We analyze the effectiveness of the proposed matching mechanism by evaluating its behavior after distribution-shift detection during incremental learning. Whenever a new distribution shift is detected, a new submodule is created, and the matching accuracy is evaluated across all existing submodules using data from the distributions observed so far. This evaluation protocol allows us to assess how accurately incoming samples are routed to the appropriate submodules as the data distribution evolves.

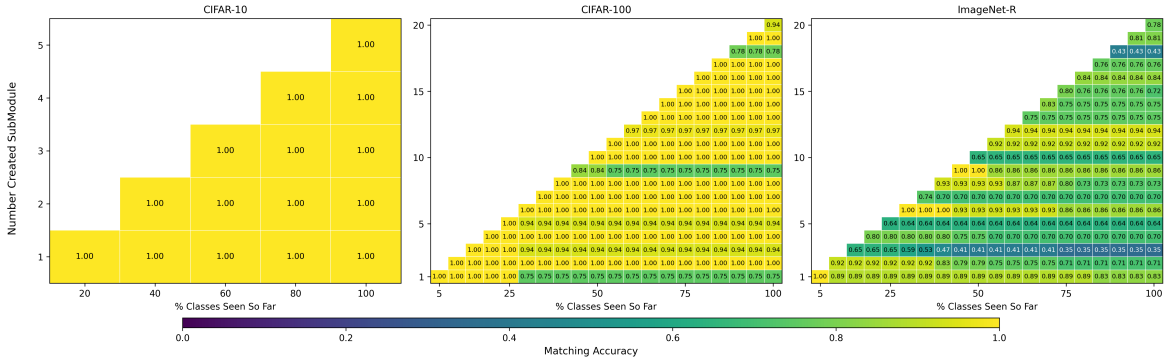

Figure 6: Matching accuracy evaluated after distribution-shift detection across incremental submodule creation. Each column corresponds to an evaluation stage after a new submodule is created (shown as the percentage of classes seen so far), and each row represents an existing submodule. The value at position (i, j) indicates the matching accuracy of submodule i when evaluation is performed after the j-th distribution shift is detected and a new submodule is introduced.

The matching accuracy over successive submodule creation stages is visualized in Fig. 6. Each column corresponds to an evaluation stage following the detection of a new distribution shift, represented as the percentage of classes seen so far, while each row denotes an existing submodule. The value at position (i, j) indicates the matching accuracy of submodule i when evaluation is performed after the j-th distribution shift has been detected and a new submodule has been introduced. This visualization provides insight into how the matching behavior changes as additional submodules are incrementally created.

From Fig. 6, we observe that newly created submodules consistently achieve high matching accuracy immediately after their creation, indicating that the matching mechanism effectively adapts to newly detected distributions. In addition, the matching accuracy of previously created submodules remains largely stable across subsequent distribution shifts, suggesting that the introduction of new submodules does not significantly interfere with the matching behavior of existing ones. These trends demonstrate that the proposed matching strategy can reliably associate incoming data with the correct submodule while maintaining robustness under non-stationary data distributions.

At the final evaluation stage, the average matching accuracy across all submodules reaches 73.18% on ImageNet-R, indicating that approximately three quarters of samples are correctly matched to their corresponding submodules after training. For CIFAR-100, the average matching accuracy increases to 95.31%, while CIFAR-10 achieves perfect matching with an accuracy of 100%. These results further confirm that the proposed matching mechanism performs reliably across different datasets, while highlighting the increased difficulty of accurate submodule matching in more diverse and complex distribution-shift scenarios such as ImageNet-R.

Distribution Shift. We evaluate the ability of HESTIA to detect distribution shifts in non-stationary data streams. The detector monitors the routing score (Eq. 9), which quantifies the discrepancy between incoming features and the active distribution signature. A new distribution is instantiated when this discrepancy exceeds a predefined threshold.

Figure 7 shows the evolution of routing scores along the training stream for CIFAR-10 and ImageNet-R. Ground-truth distribution boundaries are indicated below the curve, while predicted shift points are shown above. Routing score peaks align closely with true transition points, with stable behavior within stationary segments.

H PARAMETER COMPARISON

All methods share the same frozen ViT-B/16 backbone; therefore, we compare methods by counting only additional parameters beyond the backbone. For HESTIA, these additional parameters consist of three parts: (i) IA^3 adapter parameters (one set per instantiated adapter/segment), (ii) a linear classification head, and (iii) parameters of the Gaussian density models (task keys) used for routing and change-point detection.

IA^3 parameters (per adapter). We adapt the backbone using IA^3 rescaling vectors (`l_k`, `l_v`, `l_ff`) per Transformer block. For a Transformer with L blocks, hidden size d, and MLP size d_{ff}, the number of IA^3 parameters in one adapter is

$$N_{\text{IA}^3} \;=\; L\,(2d + d_{\text{ff}}). \tag{56}$$

For ViT-B/16 ($L = 12$, $d = 768$, $d_{\text{ff}} = 3072$), this gives $N_{\text{IA}^3} = 55{,}296$ parameters per adapter.

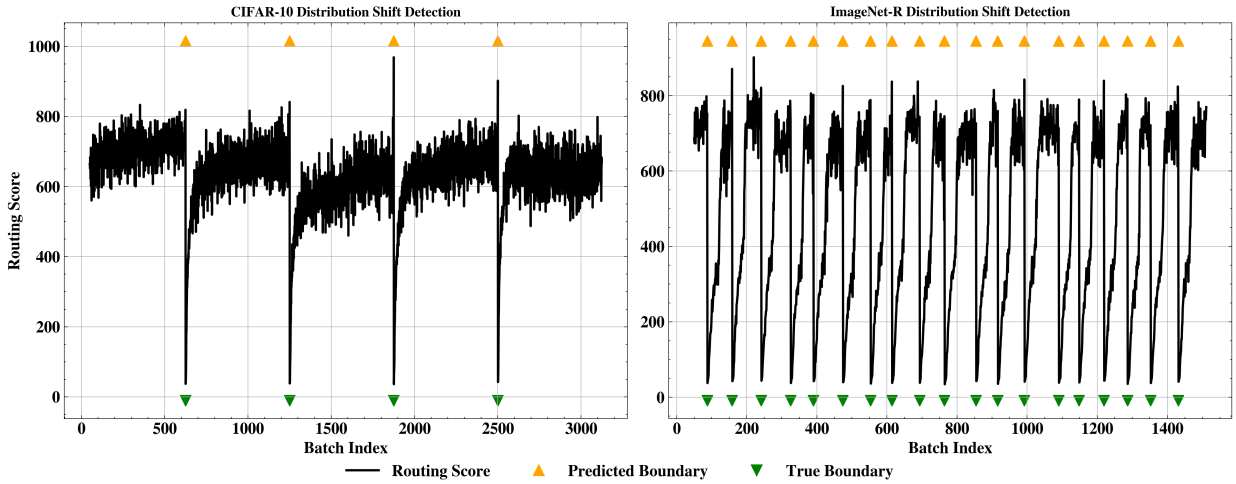

Figure 7: Distribution shift detection on CIFAR-10 and ImageNet-R data streams. The solid curve shows the evolution of recording routing score (see Eq. 9) over the training stream. Ground-truth distribution boundaries are indicated by markers below the curve, while detected distribution shifts produced by the proposed detector are shown by markers above the curve. The horizontal axis denotes the global batch index in the stream.

Classification head (shared). We train a linear classifier with C output classes,

$$N_{\text{head}} \;=\; d\,C + C. \tag{57}$$

This term is dataset-dependent through C (e.g., $C = 10, 100, 200, 196$ for Split CIFAR10/100, Split ImageNet-R, and VTAB5T, respectively).

Gaussian task keys (per adapter; diagonal covariance). Each adapter t stores a Gaussian mixture model with K components in the embedding space $\mathbb{R}^{d_z}$. We use diagonal covariances, meaning each component stores a mean vector $\boldsymbol{m}_{t,i} \in \mathbb{R}^{d_z}$ and a diagonal covariance (or diagonal precision) vector $\boldsymbol{\sigma}^2_{t,i} \in \mathbb{R}^{d_z}$. Optionally, mixture weights $\{\pi_{t,i}\}_{i=1}^{K}$ can be stored as K additional scalars. With diagonal covariances, the number of density parameters per adapter is therefore

$$N_{\text{GMM}} \;=\; K \cdot (2d_z) \;+\; K. \tag{58}$$

In our implementation, $d_z = 768$ (the [CLS] embedding dimension for ViT-B/16), so $N_{\text{GMM}} = 1537K$.

Total additional parameters after T detected segments. Let T be the number of adapters instantiated by change-point detection over the stream. The total number of non-backbone parameters used by HESTIA at the end of training is

$$N_{\text{total}}(T) \;=\; T \cdot \big(N_{\text{IA}^3} + N_{\text{GMM}}\big) \;+\; N_{\text{head}}. \tag{59}$$

During continual learning, only the current adapter is updated, while previously learned adapters remain frozen; task keys are updated online via running statistics.

Instantiating counts for ViT-B/16. With ViT-B/16 ($N_{\text{IA}^3} = 55{,}296$, $d = d_z = 768$), the classifier sizes are:

$$N_{\text{head}} = \begin{cases} 7{,}690 & (C = 10), \\ 76{,}900 & (C = 100), \\ 153{,}800 & (C = 200), \\ 150{,}724 & (C = 196). \end{cases}$$

The density parameters scale linearly with the number of mixture components K (Table 7) and the number of detected segments T (Eq. (59)).

I ADDITIONAL RELATED WORKS

Replay-based online and task-free continual learning. Replay is a dominant approach to TFCL, approximating joint training by interleaving current data with a bounded buffer of past examples [Rolnick et al., 2019]. The key challenge is

memory management under limited budget and non-i.i.d. streams [Chaudhry et al., 2019b]: deciding what to store and replay. Representative methods prioritize samples most vulnerable to interference (MIR) [Aljundi et al., 2019a], select diverse or gradient-informative examples (GSS) [Aljundi et al., 2019b], or modify stored samples to remain challenging as the model evolves (GMED) [Jin et al., 2021]. While strong baselines such as DER and GDumb demonstrate the effectiveness of replay [Buzzega et al., 2020, Prabhu et al., 2020], replay incurs memory overhead and remains tied to sequential gradient updates, making it sensitive to stream order and boundary ambiguity.

Regularization and constraint-based mitigation of forgetting. Beyond replay, a classical family of CL methods constrains parameter updates to preserve previous solutions, via importance-weighted penalties or gradient constraints [Kirkpatrick et al., 2017, Zenke et al., 2017, Aljundi et al., 2018, Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2019a, 2018]. While these approaches provide principled mechanisms to trade off stability and plasticity, their direct application to TFCL is complicated by the absence of explicit task segmentation (when to consolidate / refresh importance estimates) and by the accumulation of constraints over long streams. Moreover, the resulting training remains inherently *order-dependent* because it is still realized through incremental gradient steps on temporally localized batches.

Dynamic capacity, expert growth, and task-free routing. Another direction reduces interference by allocating separate capacity across time via modularization, expert growth, or mixture-of-experts style routing. In task-free settings, these approaches must additionally decide *when* to expand and *how* to route samples without task supervision. Bayesian nonparametric and expansion-based TFCL methods instantiate new experts in response to distributional novelty, e.g., CN-DPM [Lee et al., 2020] and CURL [Rao et al., 2019], while discrepancy-based analyses can motivate compact expansion and memory selection, as in ODDL [Ye and Bors, 2022b]. Recent task-free transformer architectures similarly employ dynamic experts with novelty/energy-based expansion and transfer mechanisms [Ye and Bors, 2024], and task-agnostic expert-switching approaches detect shifts online and route inference to expert networks [Zhu et al., 2024]. These methods can mitigate destructive interference, but introduce routing/selection complexity and may grow with stream length; selector modules may also require continual training and can forget.

Retrieval-based continual fine-tuning and parameter-free selection. Retrieval-based CFT frameworks mitigate interference by learning task- or segment-specific adaptation modules and selecting among them at inference [Wang et al., 2022c,b, Smith et al., 2023, McDonnell et al., 2023]. A recurring limitation is that many retrieval mechanisms are themselves learned and continuously updated (e.g., prompt selectors), which can introduce an additional locus of forgetting. Our approach instead emphasizes signature-based retrieval in which compact distributional summaries support both shift detection during training and principled adapter selection at inference.

Analytic and closed-form CIL methods. A complementary line of work derives closed-form, analytic updates for class-incremental learning (CIL) is based on frozen pretrained backbones. RanPAC projects features through fixed random layers and solves ridge regression in closed form to reduce forgetting [McDonnell et al., 2023]; GACL extends this idea to generalized CIL with recurring classes, maintaining a per-task autocorrelation matrix to achieve a solution weight-equivalent to joint training [Zhuang et al., 2024]; KLDA instead applies a kernel (RBF) expansion to pretrained features before solving linear discriminant analysis analytically [Momeni et al., 2025]; and AnaCP learns an analytic contrastive projection layer using per-task class means and covariance for feature whitening [Momeni et al., 2026]. These methods are effective and efficient, but they are inherently *task-aware*: they assume known task boundaries and require each task's data as a complete batch to compute the relevant statistics, which is incompatible with task-free streams where boundaries are unobserved and data arrive incrementally in small mini-batches. Online-LoRA [Wei et al., 2025] instead performs interference-aware low-rank adaptation directly in an online streaming setting, making it a more directly comparable baseline. Under our streaming protocol (batch size 16, no task IDs) on Split CIFAR-100, Online-LoRA achieves 4.68%, near-random performance. We attribute this to its sensitivity to batch size. The original paper uses batch size 64 and reports 49.4% while an independent reproduction [Michel et al., 2025] at batch size 100 reports only 35.35%. HESTIA achieves 89.69% on the same split, demonstrating substantially stronger robustness in low-batch streaming settings.

J LIMITATION

Despite the strong empirical performance of HESTIA in task-free continual learning, we foresee that our order-invariant update relies on a local linear approximation around pretrained weights; when effective adaptation requires large parameter changes, this approximation may hurt and reduce performance. Second, the Gaussian-mixture task keys may underfit when embedding distributions are highly overlapping, fine-grained, or heavy-tailed and make density-guided routing ambiguous

(e.g., under blurry or weakly separated shifts).

Addressing these limitations opens several directions for future work. For the routing limitation, more expressive non-Gaussian density models such as Student-t mixtures to better accommodate heavy tails, or normalizing-flow-based density estimators for highly overlapping or fine-grained distributions could improve routing robustness. For the adaptation limitation, exploring linearized fine-tuning in combination with other parameter-efficient fine-tuning (PEFT) methods, such as LoRA and prompt tuning, may extend the applicability of our approach beyond settings where a local linear approximation suffices. We leave these extensions to future works.