ADAPTIVE DISTILLATION VIA ONLINE DRIFT PROJECTION FOR EXEMPLAR-FREE CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exemplar-free class-incremental learning must preserve old classes without storing raw examples. Prototype-based methods address this constraint by caching compact class statistics, but these statistics become stale as the feature extractor evolves. Representation drift can be heterogeneous: some old classes remain relatively stable, whereas others move substantially. This motivates adapting knowledge distillation to class-wise drift instead of weighting all old classes equally. We propose Online Drift Projection (ODP), a co-trained module that estimates class-wise drift during incremental training. ODP is optimized jointly with the evolving backbone and projects stored old-class prototypes into the current feature space. The induced prototype displacement provides an exemplar-free drift signal for each old class, without requiring access to past samples. We use this signal to construct drift-weighted distillation, which increases the contribution of high-drift classes to the distillation objective. Experiments on CIFAR-100, TinyImageNet, and ImageNet-100 show gains over AdaGauss and DPCR, while results on pretrained CUB-200 are mixed. Additional analyses show agreement between surrogate and oracle displacement, and an association between oracle displacement and class-wise forgetting, supporting online drift projection as a useful signal for adaptive distillation.

1 INTRODUCTION

Modern recognition systems are often deployed under strict privacy or memory budgets, yet they must keep expanding their label space over time. *Exemplar-free class-incremental learning* (EFCIL) captures this constraint within continual learning (Masana et al., 2023; Zhu et al., 2022; Shi & Ye, 2024): after each task, the model must recognize all seen classes without retaining raw samples from earlier tasks. When exemplars are allowed, replay-based strategies can be highly effective (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Prabhu et al., 2020; Buzzega et al., 2020), but retaining past inputs is incompatible with exemplar-free operation. Consequently, many EFCIL methods store compact class statistics for prototype-based inference or pseudo-feature replay (Zhu et al., 2021; Petit et al., 2023; Goswami et al., 2023; Rypeść et al., 2024).

The central difficulty of prototype-based EFCIL is *representation drift*. Because the encoder is updated using only current-task labels, the embedding geometry can shift between tasks, making cached old-class means and covariances stale and contributing to recency bias (Yu et al., 2020; Rypeść et al., 2024). Existing drift-compensation methods often regularize the current training stage and then relocate old prototypes with an additional projection or post-processing step (Yu et al., 2020; Goswami et al., 2024; Gomez-Villa et al., 2024; Rypeść et al., 2024; Magistri et al., 2024). However, the same distillation coefficient need not suit all old classes. Our diagnostics show heterogeneous old-class drift, with larger oracle drift associated with greater forgetting. This motivates adapting preservation strength to estimated class-wise drift when balancing stability and plasticity (Hadsell et al., 2020; Kim & Han, 2023).

We propose **Online Drift Projection (ODP)**, which jointly trains the current backbone with a lightweight module mapping old-space features into the current feature space. For each stored old-class prototype, the projected displacement estimates class-wise drift during incremental learning

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

without storing past data. We convert these estimates into continuous distillation weights, assigning larger loss coefficients to high-drift classes. ODP thus uses prototype displacement both to align cached statistics with the current space and to allocate distillation pressure.

Contributions. First, we introduce co-trained online drift projection to estimate class-wise representation drift from stored prototypes. **Second,** we propose class-adaptive preservation through continuous drift-weighted distillation without retaining exemplars. **Third,** we evaluate estimated drift against oracle drift, separately analyze the association between oracle drift and class-wise forgetting, and report accuracy gains on the evaluated benchmarks together with protocol-dependent limitations.

2 Related Work

Prototype-based exemplar-free class-incremental learning. EFCIL cannot store past inputs, so a common design is to cache compact class statistics, such as means and covariances, and perform nearest-prototype or Gaussian Bayes inference at test time. FeTrIL (Petit et al., 2023) translates current-class features around stored centroids to generate pseudo-features for a linear classifier while keeping the backbone frozen. FeCAM (Goswami et al., 2023) uses covariance-aware scoring; EFC (Magistri et al., 2024) and AdaGauss (Rypeść et al., 2024) maintain Gaussian prototypes; DPCR (He et al., 2025) uses calibrated class statistics to reconstruct a classifier. However, when the encoder changes across tasks, even well-estimated statistics become stale. ODP addresses this failure mode by learning an online projection from old prototype space to the current feature space and using the projected displacement as a drift signal.

Prototype transport and drift compensation. Instead of freezing the representation, drift-compensation methods keep the backbone plastic and explicitly correct old-class statistics after representation updates. SDC (Yu et al., 2020) estimates old-prototype drift from current-task feature displacements, whereas ADC (Goswami et al., 2024) uses adversarially perturbed current inputs as pseudo-exemplars for old classes. LDC (Gomez-Villa et al., 2024) and AdaGauss (Rypeść et al., 2024) learn mappings between old and new feature spaces to adapt stored statistics. EFC (Magistri et al., 2024) instead regularizes drift in important feature directions and updates Gaussian prototypes using an empirical feature metric. DPCR (He et al., 2025) estimates semantic shift through a dual-projection operator and reconstructs a calibrated classifier using stored statistics. ODP is complementary: it trains a projection module together with the incremental model and reuses the projected displacement to decide how strongly each old class should be preserved.

Adaptive distillation under heterogeneous drift. Knowledge distillation (Hinton et al., 2015) is widely used to preserve old knowledge without past inputs (Li & Hoiem, 2016; Dhar et al., 2019). Adaptive feature consolidation (Kang et al., 2022) also weights representation preservation by estimated importance, in an exemplar-based setting. Concept-drift adaptation has been studied in data streams (Gama et al., 2014; Ditzler et al., 2015), although changing data distributions differ from the encoder-induced representation drift considered here. ODP obtains a class-wise drift estimate from projected prototype displacement and uses it to reweight distillation, assigning larger loss coefficients to classes with larger estimated drift.

3 Problem Setup and Background

3.1 Problem Definition

We study **class-incremental** continual learning with a sequence of tasks $t \in \{1, \dots, T\}$. Each task introduces a disjoint label set $\mathcal{C}_t$, and after finishing task t, the model must classify examples from any class in $\mathcal{C}_{1:t} := \bigcup_{i=1}^{t} \mathcal{C}_i$ *without* being given a task identifier at test time. Let $f_t : \mathcal{X} \to \mathbb{R}^d$ denote the encoder, including the projection into the prototype space, after training task t. In the **exemplar-free** setting, training task t only provides the current dataset $\mathcal{D}_t = \{(x_i, y_i) \mid y_i \in \mathcal{C}_t\}$, and no raw data from $\{\mathcal{D}_1, \dots, \mathcal{D}_{t-1}\}$ can be stored or replayed.

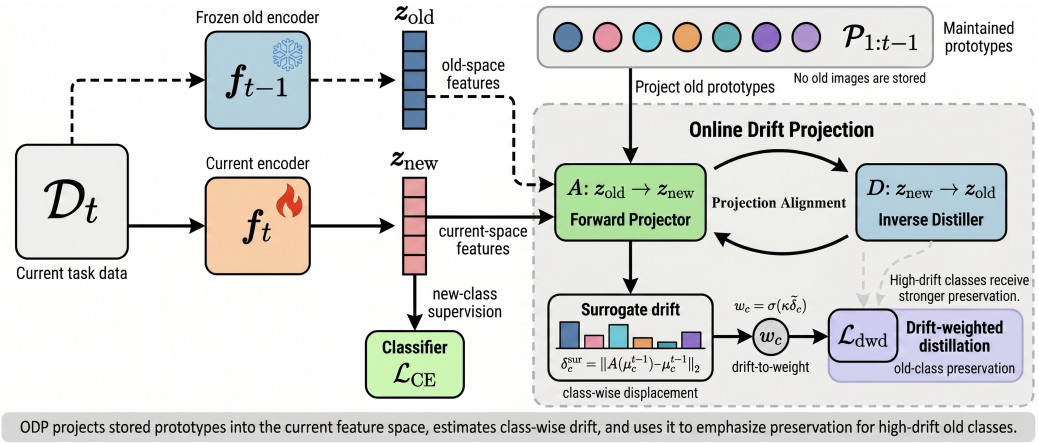


Figure 1: **Overview of Online Drift Projection (ODP).** The current encoder f_t trains on $\mathcal{D}_t$ while frozen f_{t-1} provides old-space features. Lightweight maps $A : z_{\text{old}} \to z_{\text{new}}$ and $D : z_{\text{new}} \to z_{\text{old}}$ are co-trained to align the spaces. Stored old-class prototypes $\mathcal{P}_{1:t-1}$ require no raw old images; their displacement under A supplies class-wise surrogate drift estimates. Drift-weighted distillation assigns larger loss coefficients to classes with larger estimated drift. Arrows are schematic and do not specify gradient routing; the depicted sigmoid scores are subsequently normalized to unit mean.

3.2 Prototype-Based Exemplar-Free CIL

Prototype-based EFCIL compresses past classes into simple statistics. For a class c introduced at task $t(c)$, we initially store its empirical mean and covariance in that task's feature space:

$$\boldsymbol{\mu}_c^{t(c)} = \frac{1}{|\mathcal{D}_c|} \sum_{x \in \mathcal{D}_c} f_{t(c)}(x), \tag{1}$$

where $\mathcal{D}_c$ contains the training samples of class c. At later tasks, the maintained statistics $(\boldsymbol{\mu}_c^t, \Sigma_c^t)$ are transported estimates, not means and covariances recomputed from unavailable old images.

Representation drift. After learning task t, the encoder changes from f_{t-1} to f_t, but old images are unavailable. Define the oracle mean and adjacent-encoder drift by

$$\boldsymbol{\mu}_{c,\text{ora}}^t = \frac{1}{|\mathcal{D}_c|} \sum_{x \in \mathcal{D}_c} f_t(x), \quad \delta_c^{\text{true}}(t) := \|\boldsymbol{\mu}_{c,\text{ora}}^t - \boldsymbol{\mu}_{c,\text{ora}}^{t-1}\|_2. \tag{2}$$

These oracle means are unavailable during subsequent-task training. Large drift can degrade decisions made with stale prototypes, motivating mechanisms that constrain or compensate for it without accessing old inputs.

4 Methodology

4.1 Setup

Let f_{t-1} be the frozen encoder from task $t-1$ and f_t the encoder trained on task t. For an input x, define

$$z_{\text{old}} := f_{t-1}(x) \in \mathbb{R}^d, \qquad z_{\text{new}} := f_t(x) \in \mathbb{R}^d. \tag{3}$$

We evaluate the *new* feature space with a prototype-based classifier: statistics for new classes are estimated from $\mathcal{D}_t$, while old-class prototypes cached at step $t-1$ are mapped into the current feature space by the forward projector. We instantiate two lightweight maps: an auxiliary inverse distiller $D : \mathbb{R}^d \to \mathbb{R}^d$ (new→old) and a forward projector $A : \mathbb{R}^d \to \mathbb{R}^d$ (old→new), implemented as linear layers or shallow MLPs. We use $\text{sg}(\cdot)$ for stop-gradient.

4.2 Online Drift Projection Module

Old-class statistics are available as maintained estimates in the previous feature space. Our *online drift projection* module jointly trains f_t, A, and D on $\mathcal{D}_t$, with D stabilizing cross-space alignment. Without old raw samples, A learns from paired old/new features of current-task samples and extrapolates to stored old-class prototypes. We do not assume exact extrapolation: our mechanistic analysis compares the surrogate with oracle-referenced measurements computed only for analysis.

4.2.1 Projection Alignment.

As shown in Fig. 1, ODP aligns the old and current feature spaces through a forward projector A and an inverse distiller D. We implement this projection alignment by coupling new→old feature preservation with old→new projection tracking:

$$\mathcal{L}_{\text{proj}} = \| D(z_{\text{new}}) - z_{\text{old}} \|_2^2 + \| A(z_{\text{old}}) - \text{sg}(z_{\text{new}}) \|_2^2. \tag{4}$$

The first term updates f_t and D through feature-level preservation, while the second updates A only, allowing the projector to follow the evolving new space without directly constraining f_t. To encourage round-trip reconstruction on the observed data support, we add a projection-consistency constraint:

$$\begin{aligned} \mathcal{L}_{\text{cons}} = &\| A(D(z_{\text{new}})) - \text{sg}(z_{\text{new}}) \|_2^2 \\ &+ \| D(A(z_{\text{old}})) - \text{sg}(z_{\text{old}}) \|_2^2. \end{aligned} \tag{5}$$

The first term updates (f_t, A, D) through its prediction branch, while the second updates (A, D) only. Detached targets do not block gradients through $A(D(z_{\text{new}}))$; see Appendix D.

4.2.2 Drift-Weighted Distillation (DWD)

Even with projection, drift is highly heterogeneous across old classes, while the distillation signal used during incremental training is typically class-agnostic. We address this mismatch by extracting an exemplar-free, per-class drift estimate from the co-trained projector and using it to weight distillation. For visual clarity, Fig. 1 summarizes this term as drift-weighted old-class preservation and omits the standard teacher–student logit paths.

Prototype displacement as an exemplar-free drift proxy. Let $(\boldsymbol{\mu}_c, \Sigma_c)$ denote the cached prototype of an old class $c \in \mathcal{C}_{1:t-1}$ at the end of task $t-1$ in the old feature space. We project its mean by $\widehat{\boldsymbol{\mu}}_c := A(\boldsymbol{\mu}_c)$ and define the displacement magnitude

$$\delta_c := \|\widehat{\boldsymbol{\mu}}_c - \boldsymbol{\mu}_c\|_2. \tag{6}$$

Because A is trained on current-task paired features rather than old raw samples, δ_c should be understood as a surrogate drift signal induced by the learned old-to-new projection, rather than as directly observed old-class drift. For nonlinear A, $A(\boldsymbol{\mu}_c)$ need not equal the mean of projected Gaussian samples used for task-end transport; here it serves only as a drift proxy.

From drift to weights. We normalize drift magnitudes within each step and map them to continuous weights:

$$\tilde{\delta}_c = \delta_c / (\max_j \delta_j + \epsilon), \qquad w_c = \sigma(\kappa \ \text{sg}(\tilde{\delta}_c)), \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid, $\kappa \geq 0$ controls sharpness, and $\epsilon > 0$ avoids division by zero. In practice, the sigmoid output in Eq. (7) is treated as an unnormalized preservation score; before applying the distillation loss, we rescale it to have unit mean over old classes:

$$w_c \leftarrow \frac{|\mathcal{C}_{1:t-1}| \, w_c}{\sum_{j \in \mathcal{C}_{1:t-1}} w_j}. \tag{8}$$

The positive sigmoid scores make this denominator nonzero. Normalization fixes the sum of class coefficients, but not the loss magnitude or gradient norm. We stop gradients through w_c so that A is optimized for projection, not for manipulating the weights. The following bound makes the role of the cached reference explicit.

Theorem 1: Cache-relative surrogate error. Fix a task t and an old class $c \in \mathcal{C}_{1:t-1}$. Let μ_c^{t-1} be the cached mean after task $t-1$ and let $\mu_{c,\text{ora}}^t$ be the oracle mean under f_t. Define the cache-relative oracle

displacement $\delta_c^{\mathrm{ora}}(t) := \|\mu_{c,\mathrm{ora}}^t - \mu_c^{t-1}\|_2$ and the surrogate drift $\delta_c^{\mathrm{sur}}(t) := \|A(\mu_c^{t-1}) - \mu_c^{t-1}\|_2$. Assume the projector maps the old mean to the current mean with bounded projection error:

$$\varepsilon_c(t) \;:=\; \|A(\mu_c^{t-1}) - \mu_{c,\mathrm{ora}}^t\|_2 \leq \varepsilon(t) \quad \forall c. \tag{9}$$

Then, for every class c,

$$|\delta_c^{\mathrm{sur}}(t) - \delta_c^{\mathrm{ora}}(t)| \;\leq\; \varepsilon_c(t) \;\leq\; \varepsilon(t). \tag{10}$$

Moreover, if two classes i, j satisfy $|\delta_i^{\mathrm{ora}}(t) - \delta_j^{\mathrm{ora}}(t)| > 2\varepsilon(t)$, then the surrogate preserves their ordering: $\delta_i^{\mathrm{ora}}(t) > \delta_j^{\mathrm{ora}}(t) \Rightarrow \delta_i^{\mathrm{sur}}(t) > \delta_j^{\mathrm{sur}}(t)$. [Proof in Appendix A.] This is a conditional bound, not a guarantee that projection error is small. For true adjacent-encoder drift, the bound also includes cache error $b_c(t-1) := \|\mu_c^{t-1} - \mu_{c,\mathrm{ora}}^{t-1}\|_2$: $|\delta_c^{\mathrm{sur}}(t) - \delta_c^{\mathrm{true}}(t)| \leq \varepsilon_c(t) + b_c(t-1)$.

Drift-weighted distillation on old classes. Let $s^T(x)$ and $s^S(x)$ be the frozen teacher and current student logits, respectively, both restricted to old classes. With temperature $\tau > 0$, define

$$p^T(x) := \mathrm{softmax}(s^T(x)/\tau), \tag{11a}$$

$$\log q^S(x) := \mathrm{logsoftmax}(s^S(x)/\tau). \tag{11b}$$

DWD applies class-wise weighting over old classes:

$$\mathcal{L}_{\mathrm{dwd}} = -\mathbb{E}_{x\sim\mathcal{D}_t}\Big[\sum_{c\in\mathcal{C}_{1:t-1}} w_c\, p_c^T(x)\, \log q_c^S(x)\Big]. \tag{12}$$

Larger estimated drift gives a no smaller coefficient to the corresponding teacher-weighted log-probability term. This weighted cross-entropy tilts the teacher distribution: for fixed x, its optimum is $q_c^S = w_c p_c^T / \sum_j w_j p_j^T$, generally not p_c^T. Thus DWD is a heuristic reweighting, not independent contraction of every class toward its original teacher output.

Theorem 2: Monotone allocation in a separable quadratic model. As a stylized allocation model, consider n independent scalar mismatches with magnitudes $a_c := |u_c| > 0$ and one-step dynamics $u_c^+ = (1 - \eta w_c)u_c$, $\eta > 0$. For $0 < W < n/\eta$, consider

$$\min_{\{0\leq w_c\leq 1/\eta\}} \max_c |u_c^+| \quad \text{s.t.} \quad \sum_{c=1}^n w_c = W. \tag{13}$$

The unique optimum is $w_c^\star = \eta^{-1}(1 - \xi/a_c)_+$, where $\xi \in (0, \max_c a_c)$ satisfies $\sum_c w_c^\star = W$ and $(v)_+ := \max(v, 0)$. Consequently, if a_c is nondecreasing in δ_c, then

$$\delta_i \geq \delta_j \quad \Longrightarrow \quad w_i^\star \geq w_j^\star. \tag{14}$$

This motivates monotone allocation under the stated independent dynamics; it does not derive Eq. (12), prove sigmoid optimality, or guarantee reduced forgetting. [Proof and scope in Appendix B.]

Overall objective. The full training objective combines new-class supervision, online projection, and drift-aware preservation:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{proj}}\mathcal{L}_{\mathrm{proj}} + \lambda_{\mathrm{cons}}\mathcal{L}_{\mathrm{cons}} + \lambda_{\mathrm{dwd}}\,\mathcal{L}_{\mathrm{dwd}}. \tag{15}$$

At the initial task there are no old classes or teacher, so only $\mathcal{L}_{\mathrm{CE}}$ is used. Appendix L specifies the DWD coefficient ramp-up used in the reported configuration.

5 Experiments

Baselines, protocol, and metrics. We benchmark a broad set of exemplar-free class-incremental learning methods. For classic regularization, we include EWC (Kirkpatrick et al., 2017) and LwF (Li & Hoiem, 2016) using the OCL reference (Mai et al., 2022). We also compare against recent strong baselines spanning prototype translation, drift compensation, and covariance-aware scoring: SDC (Yu et al., 2020), PASS (Zhu et al., 2021), FeTrIL (Petit et al., 2023), FeCAM (Goswami et al., 2023), EFC (Magistri et al., 2024), ADC (Goswami et al., 2024), LDC (Gomez-Villa et al., 2024), AdaGauss (Rypeść et al., 2024), and DPCR (He et al., 2025). Unless stated otherwise, we reproduce results from public implementations (FACIL (Masana et al., 2023), PyCIL (Zhou et al., 2023), or official

Table 1: **CIFAR-100 & TinyImageNet (from scratch).** Average incremental (A_{inc}) and final all-seen-class (A_{last}) accuracy (%, mean±std over five runs). Best in **bold**.

Method	CIFAR-100				TinyImageNet			
	T=10		T=20		T=10		T=20	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
EWC	30.9±1.9	50.4±1.7	17.0±1.6	34.2±2.1	18.5±1.8	34.3±2.3	11.3±1.9	26.8±2.5
LwF[ECCV16]	31.9±1.1	51.8±1.5	17.6±1.2	39.2±1.7	27.1±1.5	39.6±2.0	15.2±1.6	31.5±2.1
SDC[CVPR20]	40.6±0.9	56.2±1.3	32.3±1.0	46.6±1.4	29.5±1.1	43.8±1.5	26.3±1.2	40.6±1.7
PASS[CVPR21]	30.8±1.2	48.3±1.1	17.6±0.8	31.1±1.3	24.5±0.6	39.5±1.0	18.5±1.4	30.4±1.9
FeTrIL[WACV23]	34.9±0.5	51.2±1.1	23.3±1.4	37.9±1.2	31.0±0.9	45.3±1.8	25.9±1.2	39.9±1.2
FeCAM[NeurIPS23]	32.4±0.5	48.7±0.9	21.1±1.0	34.5±1.3	30.9±0.9	44.9±1.4	24.9±0.8	37.9±1.4
EFC[ICLR24]	43.5±0.8	58.1±1.2	32.4±0.9	47.0±1.3	34.5±1.1	47.9±1.5	28.4±1.2	42.1±1.6
ADC[CVPR24]	46.5±1.2	61.4±1.6	35.1±1.4	51.7±1.8	32.3±1.5	43.0±1.9	18.1±1.6	36.0±2.1
LDC[ECCV24]	45.4±1.6	59.5±1.9	35.5±1.9	51.9±2.3	34.2±1.1	46.8±1.6	24.9±2.2	38.2±2.7
AdaGauss[NeurIPS24]	46.8±1.2	60.9±1.0	37.9±1.0	54.4±0.8	32.9±0.9	45.8±1.3	27.5±1.2	39.5±1.1
DPCR[ICML25]	50.2±0.7	62.8±1.1	39.8±1.2	54.8±0.9	34.3±1.8	46.9±0.9	25.6±0.7	39.3±0.6
ODP	**51.7±0.7**	**64.4±0.6**	**42.3±1.1**	**56.8±1.3**	**38.0±0.9**	**50.8±1.1**	**32.5±0.8**	**45.0±1.2**

Table 2: **ImageNet-100 (from scratch) & CUB-200 (pretrained).** Average incremental (A_{inc}) and final all-seen-class (A_{last}) accuracy (%, mean±std). Best in **bold**. †: results as reported by (Gomez-Villa et al., 2024). DPCR results on CUB-200 are omitted due to numerical instability in the released implementation under the pretrained protocol.

Method	ImageNet-100				CUB-200			
	T=10		T=20		T=10		T=20	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
EWC	25.1±2.8	40.6±3.3	13.7±2.1	29.2±2.5	15.8±0.7	32.6±0.5	12.3±0.8	27.2±0.6
LwF[ECCV16]	33.4±2.2	51.5±1.6	18.6±1.6	41.3±1.9	30.4±1.1	46.1±1.0	19.4±1.6	34.7±1.8
SDC[CVPR20]	35.4±1.9	50.1±1.6	19.4±1.0	36.5±1.4	50.3±1.3	60.5±1.2	27.9±1.4	40.1±1.6
PASS[CVPR21]	26.4±1.3	45.7±0.2	14.4±1.2	31.7±0.4	27.0±0.9	42.3±0.9	18.1±1.2	36.9±1.1
FeTrIL[WACV23]	36.2±1.2	52.6±0.6	26.6±1.5	42.4±2.1	36.9±0.7	48.2±0.6	34.6±1.0	45.3±0.9
FeCAM[NeurIPS23]	38.7±1.0	54.8±0.5	29.0±1.3	44.6±2.0	40.2±0.8	54.9±1.0	36.2±1.1	48.9±1.3
EFC[ICLR24]	50.9±1.1	61.3±1.2	38.6±1.2	50.5±1.5	51.0±0.6	63.3±0.7	**46.1±1.0**	**59.3±1.3**
ADC[CVPR24]	38.3±1.2	55.5±1.5	25.1±1.3	43.4±1.7	49.5±0.9	58.8±1.1	35.4±1.4	48.3±1.4
LDC[ECCV24]	51.4 ± 1.2†	69.4 ± 0.6†	28.5±1.7	46.5±2.7	47.5±0.7	55.7±1.3	27.2±1.1	39.8±2.1
AdaGauss[NeurIPS24]	51.1±1.2	65.0±1.4	42.6±1.6	57.4±1.9	52.9±0.8	63.4±1.3	45.0±1.3	57.0±1.0
DPCR[ICML25]	49.9±0.8	64.8±1.1	37.3±1.6	54.7±0.7	–	–	–	–
ODP	**53.5±0.7**	67.6±0.9	**44.9±0.8**	**59.4±1.1**	**53.4±0.9**	**63.7±1.0**	43.1±1.1	55.4±1.2

repositories) with the authors' reported augmentations and hyper-parameters. For CIFAR-100, TinyImageNet, and ImageNet-100, we follow recent SOTA studies (EFC/ADC/LDC/AdaGauss/DPCR) and adopt a **cold-start** class-incremental protocol: models are trained from scratch and classes are evenly partitioned into $T \in \{10, 20\}$ disjoint tasks, with no enlarged first task. For clarity, while PASS/FeCAM/FeTrIL are often *reported* under a warm-start setting in their original papers, the numbers we report for these methods are obtained under the same cold-start protocol for fair comparison. For CUB-200-2011 (abbreviated CUB-200), we follow the **pretrained** protocol and report results separately from the from-scratch benchmarks. We report final accuracy over all seen classes A_{last} and average incremental accuracy A_{inc}. Diagnostic figures use zero-based task indices; their oracle drift is the cache-relative displacement defined in Appendix H. DPCR results on CUB-200 are omitted because our runs of the released implementation repeatedly failed numerically under the chosen pretrained protocol; this does not establish instability under other configurations.

Implementation and reproducibility. We build on AdaGauss (Rypeść et al., 2024) using ResNet-18 (He et al., 2016), batch size 256, and a 64-D prototype space projected from the 512-D backbone

Table 3: **Component ablation.** We isolate the effects of projection alignment $\mathcal{L}_{\text{proj}}$, projection consistency $\mathcal{L}_{\text{cons}}$, drift-weighted distillation $\mathcal{L}_{\text{dwd}}$, and optional post-training projector refinement (PTR) on CIFAR-100 and TinyImageNet. PTR is included only as a diagnostic variant; the default ODP setting is without PTR.

Components				CIFAR-100				TinyImageNet			
				T=10		T=20		T=10		T=20	
$\mathcal{L}_{\text{proj}}$	$\mathcal{L}_{\text{cons}}$	$\mathcal{L}_{\text{dwd}}$	PTR	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
×	×	×	✓	46.8	60.9	37.9	54.4	32.9	45.8	27.5	39.5
✓	×	×	✓	49.4	63.1	40.2	55.8	34.0	47.8	29.0	43.5
×	✓	×	✓	47.8	61.8	39.0	54.9	33.2	47.0	28.2	42.8
×	×	✓	✓	48.3	61.9	39.5	55.2	37.1	49.9	31.5	**45.0**
✓	✓	×	×	49.9	62.2	38.9	54.7	34.5	48.5	29.5	43.2
✓	✓	×	✓	50.6	64.2	41.5	56.5	35.4	49.1	30.2	44.2
×	✓	✓	×	48.0	62.5	39.5	54.4	35.5	49.5	30.0	44.3
×	✓	✓	✓	48.8	63.3	40.0	54.7	36.3	49.9	30.8	44.6
✓	×	✓	×	49.9	63.0	40.2	55.3	36.8	49.6	31.1	43.9
✓	×	✓	✓	50.7	64.0	41.1	56.3	37.3	50.2	31.7	44.6
✓	✓	✓	✓	51.3	64.3	**42.5**	**56.9**	**38.2**	**51.2**	32.4	44.7
✓	✓	✓	×	**51.7**	**64.4**	42.3	56.8	38.0	50.8	**32.5**	**45.0**

features for Mahalanobis inference. Appendix I provides the reported optimization settings, loss weights, and prototype handling. We evaluate TinyImageNet on all 200 classes; the AdaGauss script version used in our experiments selected the first 100 classes before this adjustment.

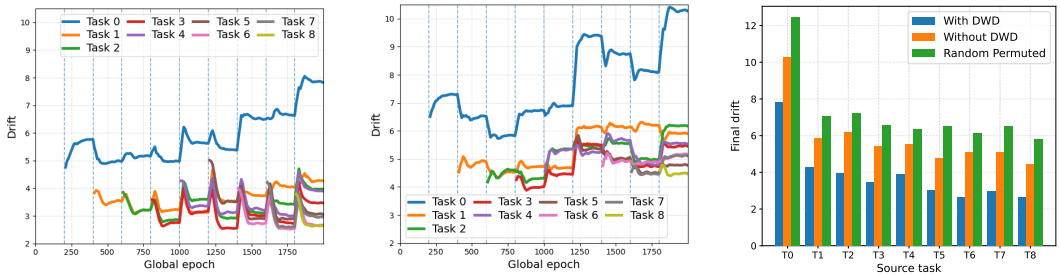
5.1 Main Results

Tables 1–2 report five-run means and standard deviations unless marked by †. The comparisons below describe reported means; no paired significance test is reported. ODP consistently improves over AdaGauss and DPCR on the from-scratch benchmarks. All gain pairs below refer to $A_{\text{last}}/A_{\text{inc}}$ in percentage points. On **CIFAR-100**, gains over AdaGauss are **+4.9/+3.5** at T=10 and **+4.4/+2.4** at T=20. On **TinyImageNet**, the gains are **+5.1/+5.0** and **+5.0/+5.5**, respectively; ODP also exceeds the next best baseline, EFC, by **+3.5/+2.9** and **+4.1/+2.9**. Compared with DPCR (He et al., 2025), ODP improves by **+6.9/+5.7** at T=20. On **ImageNet-100**, gains over AdaGauss are **+2.4/+2.6** at T=10 and **+2.3/+2.0** at T=20. At T=10, ODP has the best A_{last} among reproduced methods, while its A_{inc} trails the originally reported LDC† result by 1.8 pp. Rerunning public LDC code under our protocol gives A_{last}=41.7 $\pm$ 1.5% and A_{inc}=58.7 $\pm$ 1.7%. At T=20, ODP leads on both metrics. On pretrained **CUB-200**, gains are more modest: ODP leads at T=10, exceeding AdaGauss by **+0.5/+0.3**, but at T=20 EFC performs best and ODP trails AdaGauss by **1.9/1.6**. This regime may require further tuning of drift-weighted preservation, which we leave for future work.

5.2 Ablation Study and Mechanistic Analysis

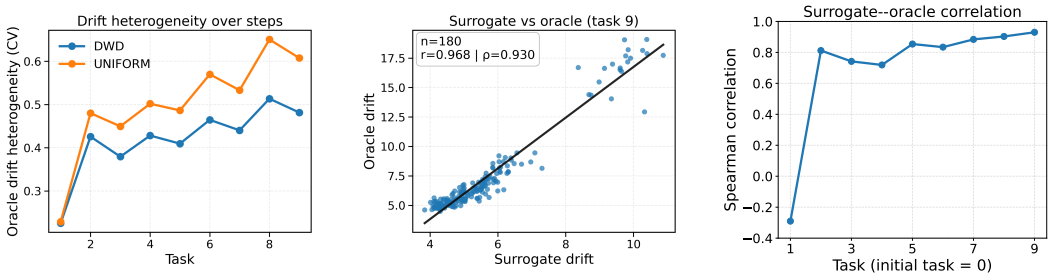
Overview. We isolate each objective term, compare DWD with uniform old-class preservation, and test whether DWD reduces the magnitude and dispersion of drift. We then validate the projector-based surrogate against oracle drift computed with held-out old data for analysis only, and connect drift reduction to forgetting and accuracy.

Model Component Ablation. Table 3 disentangles the effect of projection alignment $\mathcal{L}_{\text{proj}}$, projection consistency $\mathcal{L}_{\text{cons}}$, drift-weighted distillation $\mathcal{L}_{\text{dwd}}$, and post-training projector refinement (PTR). The first row corresponds to the AdaGauss-style baseline without our online projection losses or DWD. Adding $\mathcal{L}_{\text{proj}}$ consistently improves performance by coupling new$\rightarrow$old feature preservation with old$\rightarrow$new projection tracking. $\mathcal{L}_{\text{cons}}$ alone also improves over the baseline, suggesting that regularizing the projector–inverse-projector pair helps stabilize cross-space mapping. The key comparison for DWD is between the full non-PTR model and its non-DWD counterpart, i.e., the rows with $(\mathcal{L}_{\text{proj}}, \mathcal{L}_{\text{cons}})$ enabled and PTR disabled. Adding $\mathcal{L}_{\text{dwd}}$ improves CIFAR-100 by **+1.8/+2.2** pp at T=10 and **+3.4/+2.1** pp at T=20, and improves TinyImageNet by **+3.5/+2.3** pp at T=10 and



(a) **DWD** (b) **Uniform KD** (c) **Final drift**

Figure 2: **DWD reduces prototype drift** on TinyImageNet (T=10). (a,b) Drift trajectories grouped by source task, measured as displacement from the *initial* prototypes; blue dashed lines denote task boundaries. Compared with uniform KD, DWD produces smaller and less persistent drift, with noticeable within-stage correction. (c) Final drift after the last task follows *DWD* < *uniform KD* < *random-permute*, supporting the relevance of class-wise weight assignment. Matching the mean class weight does not fix the KD loss or gradient magnitude.



(a) **Drift heterogeneity (CV)** (b) **Surrogate vs. oracle (task 9)** (c) **Surrogate–oracle correlation**

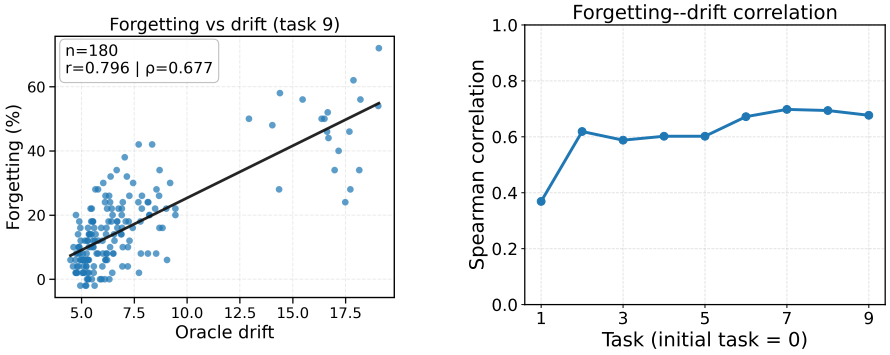
Figure 3: **Drift heterogeneity and drift-signal validity** on TinyImageNet (T=10). (a) DWD reduces drift heterogeneity measured by the coefficient of variation (CV) of per-class *oracle drift* across old classes. (b) At the last step (task 9), per-class surrogate drift strongly matches oracle drift. (c) Spearman coefficients reported in the corresponding scatter panels; no across-run uncertainty is estimated. After an initial negative correlation, agreement remains positive in the reported panels, supporting the surrogate as a drift-ranking signal.

+3.0/+1.8 pp at T=20. This supports the benefit of the DWD objective within the ODP pipeline; attributing the full gain specifically to class weighting requires a matched uniform-KD comparison. PTR, in contrast, is computationally expensive and provides only modest or inconsistent gains; therefore, we use the non-PTR setting as the default ODP configuration.

DWD Reduces Drift Magnitude. We measure each old class's drift as the displacement of its current mean from the initial prototype saved when it first appeared. Figures 2(a,b) show that DWD keeps drift smaller than uniform KD on TinyImageNet (T=10), with clear within-stage corrections. At the final checkpoint (Fig. 2(c)), DWD produces the lowest drift across source tasks, whereas randomly reassigning the same weight values to classes performs worst. This comparison supports the relevance of assigning weights to the corresponding classes, although matching mean class weights does not match the resulting loss or gradient magnitude.

DWD Also Reduces Drift Heterogeneity. We quantify heterogeneity using the coefficient of variation (CV) of per-class oracle drift over all old classes at each step. DWD consistently reduces CV relative to uniform KD (Fig. 3(a)), with a larger gap in later steps. Together with the drift-magnitude results, this indicates lower drift and lower relative dispersion in the evaluated stream.

Is the Drift Signal Reliable? Surrogate vs. Oracle Drift. We compare the exemplar-free surrogate with oracle drift estimated from held-out old-task data for analysis only. At the final step (TinyImageNet task 9; n=180 old classes), agreement is strong (Pearson r=0.968, Spearman ρ=0.930; Fig. 3(b)). The first step has negative rank correlation, while subsequent reported coefficients range



(a) **Forgetting vs. drift (task 9)** (b) **Correlation over steps**

Figure 4: **Prototype drift is strongly correlated with forgetting** on TinyImageNet (T=10). (a) At the last step (task 9), per-class forgetting increases with oracle drift (n=180, r=0.796, ρ=0.677). (b) Spearman coefficients reported in the corresponding scatter panels remain positive; no across-run uncertainty is estimated. These are associations, not evidence of causality.

from 0.719 to 0.930 (Fig. 3(c)), supporting the surrogate as a practical signal for class-wise weighting after the initial stage.

Drift Is Associated with Forgetting. Per-class oracle drift correlates positively with forgetting at the last step (task 9, n=180; Pearson r=0.796, Spearman ρ=0.677; Fig. 4(a)). Rank correlation remains positive across steps and tends to increase as tasks accumulate (Fig. 4(b)). Together with reduced drift magnitude and heterogeneity (Figs. 2–3), these trends are consistent with the motivation for DWD. They do not by themselves establish that drift causes forgetting or that the measured ranking is accurate throughout training. Appendix V further relates prototype drift to Top-1 accuracy on a fixed class set.

6 Conclusion

We presented *Online Drift Projection* (ODP), a co-trained drift-estimation framework for exemplar-free class-incremental learning. By projecting stored old-class prototypes into the evolving feature space, ODP obtains class-wise displacement estimates without accessing old examples and uses them to reweight distillation. Experiments show improvements over AdaGauss and DPCR on CIFAR-100, TinyImageNet, and ImageNet-100, with mixed results on pretrained CUB-200. Diagnostics show surrogate–oracle agreement and an association between oracle displacement and forgetting. The theoretical results provide a conditional error bound and a stylized allocation rationale, rather than a performance guarantee for the full training objective. The appendix provides proofs, implementation details, additional experiments, and limitations.

AI USE STATEMENT

AI assistance was used to polish the manuscript text and assist with writing code for this work. All AI-assisted text and code were reviewed and verified by the authors.

ETHICS STATEMENT

This work studies exemplar-free class-incremental learning. It does not collect new human-subject data. The methods inherit the capabilities and limitations of the underlying models and datasets; the reported benchmark improvements concern image-classification accuracy and do not establish suitability for safety-critical decisions.

REPRODUCIBILITY STATEMENT

Our implementation uses PyTorch (Paszke et al., 2019) and NVIDIA RTX 6000 Ada 48 GB GPUs. The full implementation will be publicly released. §4 and Appendix §E provides additional details of our method.

REFERENCES

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020.

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

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, 2019.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015. doi: 10.1109/MCI.2015.2471196.

Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014. doi: 10.1145/2523813.

Alex Gomez-Villa, Dipam Goswami, Kai Wang, Andrew D. Bagdanov, Bartlomiej Twardowski, and Joost van de Weijer. Exemplar-free continual representation learning via learnable drift compensation. In *ECCV*, 2024.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. FeCAM: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In *NeurIPS*, 2023.

Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bartlomiej Twardowski, and Joost van de Weijer. Resurrecting old classes with new data for exemplar-free continual learning. In *CVPR*, 2024.

Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Run He, Di Fang, Yicheng Xu, Yawen Cui, Ming Li, Cen Chen, Ziqian Zeng, and Huiping Zhuang. Semantic shift estimation via dual-projection and classifier reconstruction for exemplar-free class-incremental learning. In *ICML*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16071–16080, 2022.

Dongwan Kim and Bohyung Han. On the stability-plasticity dilemma of class-incremental learning. In *CVPR*, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Cs231n course project report, Stanford University, 2015. URL https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

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

Simone Magistri, Tomaso Trinci, Albin Soutif-Cormerais, Joost van de Weijer, and Andrew D. Bagdanov. Elastic feature consolidation for cold start exemplar-free incremental learning. In *ICLR*, 2024. URL https://openreview.net/forum?id=7D9X2cFnt1.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.10.021. URL https://www.sciencedirect.com/science/article/pii/S0925231221014995.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023. doi: 10.1109/TPAMI.2022.3213473.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *WACV*, January 2023.

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

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pp. 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

Grzegorz Rypeść, Sebastian Cygert, Tomasz Trzciński, and Bartłomiej Twardowski. Task-recency bias strikes back: Adapting covariances in exemplar-free class incremental learning. *NeurIPS*, 2024.

Wuxuan Shi and Mang Ye. Prospective representation learning for non-exemplar class-incremental learning. *Advances in Neural Information Processing Systems*, 37:995–1018, 2024.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical report, California Institute of Technology, 2011. URL https://authors.library.caltech.edu/records/cvm3y-5hh21.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, 2020.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning. *SCIENCE CHINA Information Sciences*, 66(9):197101, 2023. doi: https://doi.org/10.1007/s11432-022-3600-y.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, 2021.

Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9296–9305, 2022.

APPENDIX

This appendix complements the main paper with extra analyses, implementation details, proofs, metric definitions, and additional plots. For easier navigation, we summarize the contents below:

- §A. Proof of Theorem 1.
- §B. Proof of Theorem 2.
- §C. Additional details of Online Drift Projection.
- §D. Gradient routing and stop-gradient design in ODP.
- §E. Pseudocode of our algorithm.
- §F. Mahalanobis inference in the current feature space.
- §G. Scope of the Gaussian prototype assumption.
- §H. Metric definitions for drift, heterogeneity, and correlation analyses.
- §I. Experimental setup.
- §J. Computation overhead and wall-clock time.
- §K. Accuracy and forgetting metrics.
- §L. Sensitivity to regularization hyperparameters.
- §M. Choice of drift-to-weight mapping.
- §N. Random-permutation control for drift weights.
- §O. Cross-dataset surrogate–oracle agreement.
- §P. Surrogate–oracle agreement over time.
- §Q. Oracle drift is associated with class-wise forgetting.
- §R. Regime dependence on CUB-200 under pretrained fine-tuning.
- §S. ImageNet-1K experiment.
- §T. Scope of ViT experiments under the current protocol.
- §U. Limitations and future work.
- §V. Additional drift–accuracy analysis.
- §W. Warm-start evaluation.
- §X. One-step projection endpoint error.

A PROOF OF THEOREM 1

Let $a := A(\mu_c^{t-1})$, $b := \mu_{c,\mathrm{ora}}^{t}$, and $u := \mu_c^{t-1}$. By the triangle inequality,

$$\|a-u\|_2 \leq \|a-b\|_2 + \|b-u\|_2 \quad \Rightarrow \quad \delta_c^{\mathrm{sur}}(t) \leq \varepsilon_c(t) + \delta_c^{\mathrm{ora}}(t). \tag{16}$$

Similarly,

$$\|b-u\|_2 \leq \|b-a\|_2 + \|a-u\|_2 \quad \Rightarrow \quad \delta_c^{\mathrm{ora}}(t) \leq \varepsilon_c(t) + \delta_c^{\mathrm{sur}}(t). \tag{17}$$

Combining both inequalities gives $|\delta_c^{\mathrm{sur}}(t) - \delta_c^{\mathrm{ora}}(t)| \leq \varepsilon_c(t)$. For ordering, apply the bound to two classes i, j: $\delta_i^{\mathrm{sur}} \geq \delta_i^{\mathrm{ora}} - \varepsilon$ and $\delta_j^{\mathrm{sur}} \leq \delta_j^{\mathrm{ora}} + \varepsilon$. If $\delta_i^{\mathrm{ora}} - \varepsilon > \delta_j^{\mathrm{ora}} + \varepsilon$, i.e., the gap is larger than 2ε, then $\delta_i^{\mathrm{sur}} > \delta_j^{\mathrm{sur}}$. For the true adjacent-encoder drift, the reverse triangle inequality also gives

$$|\delta_c^{\mathrm{ora}}(t) - \delta_c^{\mathrm{true}}(t)| \leq \|\mu_c^{t-1} - \mu_{c,\mathrm{ora}}^{t-1}\|_2 = b_c(t-1).$$

Combining the two bounds yields $|\delta_c^{\mathrm{sur}}(t) - \delta_c^{\mathrm{true}}(t)| \leq \varepsilon_c(t) + b_c(t-1)$. Thus ranking guarantees for true drift require gaps exceeding the sum of both classes' projection and cache-error bounds. □

B Proof of Theorem 2

For the independent quadratic objective $\frac{1}{2}\sum_c w_c u_c^2$, one gradient step gives $u_c^+ = (1-\eta w_c)u_c$. The box constraints $0 \le w_c \le 1/\eta$ ensure $|u_c^+| = a_c(1-\eta w_c)$. To attain a maximum residual of at most $\xi \ge 0$, each class requires

$$w_c \ge h_c(\xi) := \frac{1}{\eta}\left(1 - \frac{\xi}{a_c}\right)_+ .$$

The function $H(\xi) := \sum_c h_c(\xi)$ is continuous and strictly decreasing on $[0, \max_c a_c]$, with $H(0) = n/\eta$ and $H(\max_c a_c) = 0$. Because $0 < W < n/\eta$, there is a unique $\xi^\star \in (0, \max_c a_c)$ such that $H(\xi^\star) = W$. Any smaller residual threshold would require total weight greater than W and is infeasible. The weights $w_c^\star = h_c(\xi^\star)$ meet the budget and attain threshold $\xi^\star$. Every optimal allocation must satisfy $w_c \ge h_c(\xi^\star)$; because these lower bounds already sum to W, equality holds for every class, proving uniqueness. For fixed $\xi^\star \ge 0$, $h_c(\xi^\star)$ is nondecreasing in a_c, so $a_i \ge a_j \Rightarrow w_i^\star \ge w_j^\star$. If a_c is nondecreasing in drift, the same ordering holds for drift. □

Scope relative to DWD. The independent quadratic dynamics are a stylized assumption, not a local derivation of the weighted softmax cross-entropy in Eq. (12). For one input, let $Z(x) := \sum_c w_c p_c^T(x)$ and $\bar{p}_c(x) := w_c p_c^T(x)/Z(x)$. Then its DWD loss is $Z(x)\,\mathrm{CE}(\bar{p}(x), q^S(x))$, with logit gradient

$$\frac{\partial \ell_{\mathrm{dwd}}(x)}{\partial s_j^S(x)} = \frac{Z(x) q_j^S(x) - w_j p_j^T(x)}{\tau}.$$

At $q^S = p^T$ this gradient generally does not vanish unless weights are uniform. Softmax also couples the class coordinates, and gradients through a shared encoder couple their parameter updates. Unit-mean class coefficients do not fix $Z(x)$, the loss magnitude, or the gradient norm. Theorem 2 therefore motivates an ordering in an idealized allocation problem; it does not establish optimality of the sigmoid rule or a performance guarantee for the implemented objective.

C Additional details of Online Drift Projection

ODP uses a lightweight projection module co-trained with the current backbone to estimate how stored old-class prototypes move in the evolving feature space. This appendix provides implementation details, pseudocode, metric definitions, proofs, and additional analyses for the proposed online drift projection and drift-weighted distillation objective.

The method is designed for prototype-based exemplar-free class-incremental learning, where old classes are represented only by compact feature statistics rather than stored images. The projection module plays two roles. First, it maps cached old-class statistics into the current feature space, allowing old and new classes to be compared by the same Mahalanobis classifier. Second, the induced prototype displacement provides an online, exemplar-free estimate of class-wise drift. DWD uses this estimate to allocate preservation strength non-uniformly across old classes.

The distinction of DWD is therefore not weighted distillation in the abstract, but the source and timing of the weighting signal. The weights are driven by online projection-induced per-class drift and are available during backbone training. This differs from post-hoc drift compensation, where a drift estimate or prototype correction is computed only after finishing a task.

Why shared updates can produce heterogeneous drift. Let $\theta \in \mathbb{R}^p$ parameterize a d-dimensional encoder and let $\mu_c(\theta) = \mathbb{E}_{x\sim\mathcal{D}_c}[f_\theta(x)]$. For a small update $\Delta\theta = -\eta g_t$ with $\eta > 0$, a first-order expansion gives

$$\Delta\mu_c \approx J_c \Delta\theta = -\eta J_c g_t, \qquad \|\Delta\mu_c\|_2 \approx \eta \|J_c g_t\|_2, \tag{18}$$

where $J_c = \partial\mu_c/\partial\theta \in \mathbb{R}^{d\times p}$. When differentiation and expectation can be interchanged, $J_c = \mathbb{E}_{x\sim\mathcal{D}_c}[\partial f_\theta(x)/\partial\theta]$. The update is shared, but its effect depends on the class-conditional Jacobian and its alignment with g_t. This provides a local explanation for unequal class-wise drift, not a guarantee of heterogeneity under every architecture or training regime. Figure 3a reports the measured dispersion for the TinyImageNet diagnostic.

Adjacent-space transport and scope of the evidence. At task t, the forward projector approximates the transformation from f_{t-1} to f_t using paired features of current-task inputs. Applying this map to old-class prototypes is an extrapolation across classes, rather than supervision by unavailable old examples. After the task, transported statistics are stored in the current space, and the next task learns the next adjacent-space map. The projector is therefore not required to retain every historical map; nevertheless, errors in transported statistics can accumulate across tasks. Theorem 1 gives a conditional error bound, not a guarantee that cross-class projection error is always small.

The component study in Table 3 measures the benefit of DWD within the ODP pipeline. It does not isolate a universally superior projector architecture. Similarly, PTR evaluates extra task-end refitting for prototype transport; it does not test alternative drift signals under an online or two-pass training protocol.

D GRADIENT ROUTING AND STOP-GRADIENT DESIGN IN ODP

A common concern with co-trained projection is stability during early training and potential circular dependency between the drift signal and the projector itself. The stated losses use **explicit gradient routing** via stop-gradient targets; this routing alone does not guarantee nondegenerate projections.

Which terms update which modules? Let $z_{\text{old}} = f_{t-1}(x)$ (frozen, no grad) and $z_{\text{new}} = f_t(x)$. The projection alignment loss is

$$\mathcal{L}_{\text{proj}} = \|D(z_{\text{new}}) - z_{\text{old}}\|_2^2 + \|A(z_{\text{old}}) - \text{detach}\,(z_{\text{new}})\,\|_2^2.$$

The first term updates (f_t, D) and acts as feature-level preservation in the new$\to$old direction. The second term updates only A because the target $\text{detach}\,(z_{\text{new}})$ stops gradients to f_t; hence A tracks the evolving feature space without pulling f_t backward.

The projection consistency loss is

$$\mathcal{L}_{\text{cons}} = \|A(D(z_{\text{new}})) - \text{detach}\,(z_{\text{new}})\,\|_2^2 + \|D(A(z_{\text{old}})) - \text{detach}\,(z_{\text{old}})\,\|_2^2,$$

whose first term updates (f_t, A, D) through $A(D(z_{\text{new}}))$ and whose second term updates (A, D) only because z_{old} is frozen. In particular, detaching the target does not detach the input to the prediction branch. This loss encourages round-trip reconstruction on observed features, but does not establish global invertibility or prevent collapse outside the observed support. Blocking its gradient to f_t would additionally require detaching the input z_{new}; that would be a different objective from the one written here.

Drift weights do not backpropagate. In the drift-weighted distillation objective, the class weights w_c are treated as a *measured signal* rather than an optimizable variable. We stop gradients through w_c to prevent the model from trivially shaping the projector to manipulate the weights.

E PSEUDOCODE OF OUR ALGORITHM

Alg. 1 summarizes the overall training and prototype-update procedure of our method in the exemplar-free class-incremental learning (EFCIL) setting. At each task t, we jointly optimize the current backbone f_t with the ODP projector–inverse-distiller pair (A, D) using projection alignment losses, and apply drift-weighted distillation (DWD) to selectively regularize old classes based on projector-induced prototype displacement. After training, we project stored Gaussian prototypes from the previous feature space into the current space via the learned projector A to enable Mahalanobis inference without retaining past data.

F MAHALANOBIS CLASSIFICATION IN THE CURRENT FEATURE SPACE

We perform evaluation in the *current* feature space induced by f_t using Gaussian class statistics. Each class c is represented by a mean–covariance pair (μ_c^t, Σ_c^t) in that space: new-class statistics are estimated from $\mathcal{D}_t$, while old-class statistics are projected from $(\mu_c^{t-1}, \Sigma_c^{t-1})$ via the learned projector (Alg. 1).

Algorithm 1 Online Drift Projection (ODP) for EFCIL

Inputs: Task stream $\{\mathcal{D}_t\}_{t=1}^{T}$; encoder initialization for the chosen protocol;
classifier g; projector $A : \mathbb{R}^d \to \mathbb{R}^d$ (old→new); inverse distiller $D : \mathbb{R}^d \to \mathbb{R}^d$ (new→old);
initially empty prototype store;
hyperparameters $\lambda_{\text{proj}}, \lambda_{\text{cons}}, \lambda_{\text{dwd}}$, temperature τ, sharpness κ, ε, ramp fraction r, epoch count E, and sample count $M \geq 2$.
Outputs: Trained f_t for each task and (A, D) for $t > 1$; updated prototypes for inference.

for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Initialize f_1 and classifier g for the chosen protocol.
  else
    Freeze a copy of the previous encoder and old classifier as the teacher.
    Initialize $f_t \leftarrow f_{t-1}$; expand g for new classes; initialize A, D.
  end if
  # Online training on $\mathcal{D}_t$
  for each minibatch (x, y) during epochs $e = 0, \ldots, E-1$ **do**
    $z_{\text{new}} \leftarrow f_t(x)$
    $\mathcal{L}_{\text{CE}} \leftarrow \text{CE}(g(z_{\text{new}}), y)$
    if $t > 1$ **then**
      $z_{\text{old}} \leftarrow f_{t-1}(x)$ *no grad*
      $\mathcal{L}_{\text{proj}} \leftarrow \|D(z_{\text{new}}) - z_{\text{old}}\|_2^2 + \|A(z_{\text{old}}) - z_{\text{new}}^{(\text{detach})}\|_2^2$
      $\mathcal{L}_{\text{cons}} \leftarrow \|A(D(z_{\text{new}})) - z_{\text{new}}^{(\text{detach})}\|_2^2 + \|D(A(z_{\text{old}})) - z_{\text{old}}^{(\text{detach})}\|_2^2$
    else
      $\mathcal{L}_{\text{proj}}, \mathcal{L}_{\text{cons}} \leftarrow 0$
    end if
    DWD (only if $t > 1$): drift → class weights
    if $t > 1$ **then**
      for each old class $c \in \mathcal{C}_{1:t-1}$ **do**
        $\delta_c \leftarrow \|A(\mu_c^{t-1}) - \mu_c^{t-1}\|_2$
      end for
      $\tilde{\delta}_c \leftarrow \delta_c / (\max_j \delta_j + \varepsilon)$
      $a_c \leftarrow \sigma(\kappa\, \tilde{\delta}_c^{(\text{detach})})$ (raw preservation score)
      $w_c \leftarrow \dfrac{|\mathcal{C}_{1:t-1}|\, a_c}{\sum_{j \in \mathcal{C}_{1:t-1}} a_j}$ (unit-mean normalization)
      DWD loss on old classes
      Obtain teacher/student logits using their respective classifiers; restrict both to old classes.
      $p_T \leftarrow \text{softmax}(s_T(x)/\tau)$; $\log q_S \leftarrow \text{logsoftmax}(s_S(x)/\tau)$
      $\mathcal{L}_{\text{dwd}} \leftarrow -\frac{1}{B} \sum_{b=1}^{B} \sum_{c \in \mathcal{C}_{1:t-1}} w_c\, p_{T,b,c}\, \log q_{S,b,c}$
    else
      $\mathcal{L}_{\text{dwd}} \leftarrow 0$
    end if
    Total objective (current epoch e; $\gamma(e; r)$ from Appendix L)
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}} + \lambda_{\text{proj}} \mathcal{L}_{\text{proj}} + \lambda_{\text{cons}} \mathcal{L}_{\text{cons}} + \lambda_{\text{dwd}} \gamma(e; r) \mathcal{L}_{\text{dwd}}$
    Update (f_t, g) and, if $t > 1$, (A, D) by SGD.
  end for
  # Prototype projection (no training)
  for each old class $c \in \mathcal{C}_{1:t-1}$ **do**
    Load stored stats $(\mu_c^{t-1}, \Sigma_c^{t-1})$.
    Sample old features: draw $U = \{u_m\}_{m=1}^{M} \sim \mathcal{N}(\mu_c^{t-1}, \Sigma_c^{t-1})$.
    Push-forward to new space: $V = \{v_m\}_{m=1}^{M}$ with $v_m \leftarrow A(u_m)$.
    Re-estimate in new space:
      $\tilde{\mu}_c^{t} \leftarrow \frac{1}{M} \sum_{m=1}^{M} v_m, \quad \tilde{\Sigma}_c^{t} \leftarrow \frac{1}{M-1} \sum_{m=1}^{M} (v_m - \tilde{\mu}_c^{t})(v_m - \tilde{\mu}_c^{t})^\top$.
  end for
  Estimate new-class stats under f_t from $\mathcal{D}_t$: (μ_c^t, Σ_c^t) for all $c \in \mathcal{C}_t$.
  Build a new prototype collection using $\{(\tilde{\mu}_c^t, \tilde{\Sigma}_c^t)\}_{c \in \mathcal{C}_{1:t-1}}$ and $\{(\mu_c^t, \Sigma_c^t)\}_{c \in \mathcal{C}_t}$.
  For old classes set $(\mu_c^t, \Sigma_c^t) \leftarrow (\tilde{\mu}_c^t, \tilde{\Sigma}_c^t)$; store all seen-class statistics for the next task.
end for

Scoring rule. Given $z = f_t(x)$, we score each class by Mahalanobis distance

$$s_c(x) := (z - \mu_c^t)^\top (\Sigma_c^t)^{-1} (z - \mu_c^t), \qquad \hat{y} = \arg\min_c s_c(x), \tag{19}$$

and report task-agnostic Top-1 accuracy over all seen classes. This is a Mahalanobis decision rule. Full Gaussian Bayes scoring would additionally include $\log\det\Sigma_c^t - 2\log\pi_c$ for class prior π_c; these terms cancel only when they are class-independent.

Numerical stability. Before inversion, we symmetrize and ridge-regularize each covariance: $\Sigma \leftarrow \frac{1}{2}(\Sigma + \Sigma^\top) + \epsilon I$ with a small ϵ, and use a stable solver, e.g., Cholesky when SPD, for $(\Sigma_c^t)^{-1}$. The regularization coefficient affects the score and should be specified with the implementation.

G Scope of the Gaussian prototype assumption

ODP follows the common prototype-based EFCIL practice of representing each class by compact feature-space statistics. Each class is summarized by a mean and covariance and evaluated with the Mahalanobis rule above in the current feature space. This Gaussian parameterization is not a new assumption introduced by ODP; rather, it is the standard compact descriptor used by many prototype-based exemplar-free methods that cannot retain raw old samples.

The role of ODP is therefore not to argue that a single Gaussian is the most expressive possible class model. Instead, ODP improves how these compact statistics are aligned and regularized as the encoder changes over tasks. The forward projector maps stored old-class statistics into the current feature space, while the induced displacement provides an online drift signal for class-wise preservation.

A more expressive density model, such as a mixture of Gaussians or a learned latent-variable representation, could be incorporated in future work. However, such extensions would introduce a different modeling problem and additional optimization cost. Our analyses examine representation drift; they do not directly test whether the single-Gaussian approximation adequately captures each class distribution. That said, strongly multi-modal, noisy-label, open-set, or highly imbalanced settings may require richer class descriptors, and we view covariance-, margin-, or mixture-aware drift signals as promising directions beyond the compact Gaussian prototype setting considered in this work.

H Metric definitions for drift, heterogeneity, and correlation analyses

Oracle drift and surrogate drift. Let μ_c^{t-1} denote the maintained cached old-class mean after task $t-1$. For the correlation and CV diagnostics, the quantity referred to as *oracle drift* is the following cache-relative oracle displacement of an old class $c \in \mathcal{C}_{1:t-1}$:

$$\delta_c^{\mathrm{ora}}(t) := \left\|\hat{\mu}_c^t - \mu_c^{t-1}\right\|_2, \tag{20}$$

where $\hat{\mu}_c^t$ is estimated from the historical validation data of class c embedded by f_t. It includes accumulated cache error and differences between training and validation estimates; it is not pure adjacent-encoder drift unless the cached origin equals the previous oracle mean. These held-out data and oracle statistics are used only for analysis, never for training, weights, or prototype updates. Our method uses the projector A as a measurement device and defines the *surrogate drift*

$$\delta_c^{\mathrm{sur}}(t) := \left\|A(\mu_c^{t-1}) - \mu_c^{t-1}\right\|_2, \tag{21}$$

which is mapped to drift weights w_c via per-step normalization, sigmoid scoring, and unit-mean normalization.

Drift heterogeneity index (DHI). To quantify drift non-uniformity, we compute dispersion statistics over $\{\delta_c^{\mathrm{ora}}(t)\}_{c\in\mathcal{C}_{1:t-1}}$ using the coefficient of variation (CV). Dispersion measures heterogeneity of this cache-relative quantity; it does not by itself establish an optimal weighting rule.

Coefficient of variation (CV). For a fixed step t, let $\{\delta_c^{\mathrm{ora}}(t)\}_{c\in\mathcal{C}_{1:t-1}}$ be the oracle drift values over old classes. We compute drift heterogeneity in Fig. 3a of the main paper using the coefficient of variation

$$\mathrm{CV}(t) := \frac{\mathrm{Std}_{c\in\mathcal{C}_{1:t-1}}[\delta_c^{\mathrm{ora}}(t)]}{\mathrm{Mean}_{c\in\mathcal{C}_{1:t-1}}[\delta_c^{\mathrm{ora}}(t)] + \varepsilon}, \tag{22}$$

where ε is a small constant to avoid division by zero.

Correlations in Fig. 3 and Fig. 4 of the main paper. For Fig. 3b and Fig. 3c, at each step t we compute Pearson correlation r and Spearman rank correlation ρ between the vectors $(\delta_c^{\mathrm{sur}}(t))_{c\in\mathcal{C}_{1:t-1}}$ and $(\delta_c^{\mathrm{ora}}(t))_{c\in\mathcal{C}_{1:t-1}}$. For Fig. 4b, we compute Spearman ρ between per-class oracle drift $\{\delta_c^{\mathrm{ora}}(t)\}$ and per-class forgetting $\{f_c(t)\}$, defined in Appendix K, over the same old-class set.

Cumulative displacement and forgetting. The cumulative diagnostic uses the initial-prototype reference of Fig. 2. Let $t(c)$ be the task at which class c first appears and let μ_c^{init} be its prototype saved at that task. We measure cumulative displacement by

$$\Delta_c^{\mathrm{cum}}(t) := \left\|\hat{\mu}_c^t - \mu_c^{\mathrm{init}}\right\|_2 . \tag{23}$$

The initial prototype is held fixed for this analysis; it is not the cached prototype after repeated transport, and this displacement is not a sum of step-wise distances. Table 12 reports the Spearman correlation of final cumulative displacement with per-class forgetting $f_c(t) = a_c^{t(c)} - a_c^t$ over old classes. This differs from the step-wise drift–forgetting statistic in Fig. 4. Appendix X separately evaluates one-step projection error with both endpoint means recomputed from held-out data.

I Experimental setup

All experiments are run on a workstation with an NVIDIA RTX 6000 Ada GPU and an Intel Xeon Gold 6448Y CPU.

Datasets and task streams. We benchmark on CIFAR-100, TinyImageNet, ImageNet-100, and CUB-200. For each dataset, we create class-incremental streams by splitting the class set into disjoint groups with different granularities. Each class appears in exactly one task, and no raw samples from previous tasks are stored or replayed. Within each task, the model is trained for multiple epochs over the current-task data following the protocol and schedules used in the main experiments. We always use the official training and validation/test splits.

- **CIFAR-100** (Krizhevsky, 2009): 50,000 train and 10,000 test images of size 32×32 from 100 classes.
- **TinyImageNet** (Le & Yang, 2015): 100,000 train and 10,000 validation images of size 64×64 from 200 classes.
- **ImageNet-100 (ImageNet-Subset)** (Deng et al., 2009): approximately 130,000 train and 5,000 validation images at 224×224 from 100 classes.
- **CUB-200-2011** (Wah et al., 2011): 11,788 images (5,994 train / 5,794 test) across 200 bird species; images are center-cropped and resized to 224×224 to match ImageNet-style preprocessing.

Evaluation. Unless stated otherwise, we evaluate with a batch size of 512 and without test-time augmentation. The code will be released upon publication.

I.1 Detailed training configuration

Implementation and reproducibility. We build on the public AdaGauss (Rypeść et al., 2024) implementation and integrate our projection module and DWD on top of it. Unless noted, we train a ResNet-18 (He et al., 2016) with batch size 256 from scratch; its 512-D backbone features are linearly projected to a 64-D prototype space where we maintain per-class Gaussians and perform Mahalanobis inference. For CIFAR-100, TinyImageNet, and ImageNet-100, we run SGD for 200 epochs with lr 1×10^{-1} and wd 5×10^{-4}. For CUB-200, we follow the pretrained setting and use lr 1×10^{-2} for the backbone and lr 1×10^{-1} for the task heads. The inverse projector and forward projector are optimized jointly with lr 5×10^{-2} and wd 1×10^{-4}. The TinyImageNet schedule study uses a default DWD ramp-up fraction $r = 0.05$ as specified in Appendix L. We set λ_{proj}=5, λ_{cons}=1, λ_{dwd}=5, temperature τ=2, and decay the learning rate by $\times0.1$ at epochs $\{60, 120, 180\}$. All remaining details, including prototype storage/sampling and covariance handling, are inherited from AdaGauss. Finally, the public AdaGauss TinyImageNet script averages over splits from the first 100 classes; we instead evaluate on all 200 classes and report the corrected numbers in our tables.

Table 4: **Computation overhead** comparison on CIFAR-100 (32×32). Train FLOPs/epoch are measured under a unified FLOPs accounting; inference FLOPs/epoch count backbone evaluation only and exclude class-dependent statistical scoring. "Extra FLOPs" denotes post-training or auxiliary-stage optimization beyond standard incremental training, not lightweight operations inside the ordinary training loop.

Method	Train FLOPs / epoch	Trainable Params	Inference FLOPs / epoch	Extra FLOPs
ODP	33.48 TFLOPs	11.73 M	1.11 TFLOPs	None
AdaGauss	33.42 TFLOPs	11.47 M	1.11 TFLOPs	0.12 GFLOPs / epoch (9.7 TF one-off caching)
EFC	33.80 TFLOPs	11.20 M	1.11 TFLOPs	27.9 TF one-off post-processing
LDC	34.20 TFLOPs	11.18 M	1.11 TFLOPs	11.40 TFLOPs / epoch projector training
ADC	34.20 TFLOPs	11.18 M	1.11 TFLOPs	26.70 TFLOPs one-off attack stage

Table 5: **Wall-clock training time** on CIFAR-100 with 10 splits. All numbers are measured in the same environment with a single RTX 6000 Ada GPU.

Method	Training time (min)
EFC	42.1±2.2
LDC	35.0±1.8
ADC	37.0±1.9
AdaGauss	36.4±2.0
DPCR	**32.0±1.6**
ODP	34.8±1.7

J Computation overhead and wall-clock time

For completeness, we provide an explicit breakdown of training and inference costs on CIFAR-100 under a unified FLOPs accounting. Table 4 reports per-epoch FLOPs, trainable parameters, inference FLOPs, and additional compute beyond standard incremental training.

The "Extra FLOPs" column refers specifically to post-training or auxiliary-stage optimization beyond the standard incremental training loop. Under this definition, ODP has no additional post-hoc fitting stage by default: the projection alignment losses and DWD are optimized jointly with the backbone during ordinary task training. This does not mean that the in-loop overhead is exactly zero. ODP still includes lightweight projection losses, a 64-D projector/distiller pair, and a task-end prototype/statistics update. However, these operations are small relative to the ResNet-18 backbone and are absorbed into the standard training phase.

We also report end-to-end wall-clock time on CIFAR-100 with 10 splits in Table 5. All methods are run in the same environment using a single NVIDIA RTX 6000 Ada GPU and PyTorch 2.7. ODP remains practically efficient: it is faster than AdaGauss, ADC, and EFC, and remains close to LDC and DPCR.

Wall-clock time does not always follow the exact ordering suggested by FLOPs. For example, EFC includes empirical feature-matrix construction, matrix decomposition, and prototype refresh. Such operations may contribute to its measured runtime, but the table does not provide an operation-level timing profile.

K Accuracy metrics

We summarize continual-learning performance using aggregate recognition accuracy over all classes observed so far. Let $a_i^{(K)}$ denote the accuracy on task i after training through step K, and let $|\mathcal{C}_i|$ be the number of classes introduced at step i. We report the **last-step class-count weighted accuracy** A_{last} and its **running mean across steps** A_{inc}:

$$A_{\text{last}} = \frac{\sum_{i=1}^{K} |\mathcal{C}_i|\, a_i^{(K)}}{\sum_{i=1}^{K} |\mathcal{C}_i|}, \qquad A_{\text{inc}} = \frac{1}{K} \sum_{j=1}^{K} A_{\text{last}}^{(j)}, \tag{24}$$

Table 6: **Parameter sensitivity** of DWD on TinyImageNet under T=10 and T=20. We vary the DWD loss weight λ_{dwd} and the ramp-up fraction r. A_{last} denotes final accuracy after the last task, and A_{inc} denotes average incremental accuracy across steps.

Hyper-parameters		**TinyImageNet**			
		T=10		T=20	
λ_{dwd}	r	A_{last}	A_{inc}	A_{last}	A_{inc}
5	0.05	38.0	50.8	**32.5**	**45.0**
0.5	0.05	37.1	50.2	30.7	44.1
1	0.05	37.4	50.6	31.7	44.5
10	0.05	**38.1**	**51.1**	32.3	44.8
5	0.00	37.9	50.7	**32.5**	**45.0**
5	0.10	37.7	50.6	32.2	44.7
5	0.30	37.5	50.4	32.0	44.6

where $A_{\text{last}}^{(j)}$ denotes A_{last} evaluated at step j. This class-count weighting agrees with pooled sample accuracy when task sample proportions match class proportions; in general they differ. It it equals macro-class accuracy when the task accuracies are themselves class-balanced. Intuitively, A_{last} captures a snapshot of performance at the current step over all seen classes with class-count weighting, while A_{inc} averages these snapshots across steps to reflect stability over the full stream.

Forgetting (per-class, used in Fig. 4 of the main paper). For correlation analyses between drift and forgetting, we compute forgetting at the *class level*. Let $a_c^{(t)}$ denote the Top-1 accuracy restricted to class c after training through step t, evaluated on the standard validation/test split. Let $t(c)$ be the step when class c is first introduced. We define per-class forgetting at step t as

$$f_c(t) := a_c^{(t(c))} - a_c^{(t)}. \tag{25}$$

This quantity can be negative if the model improves on class c after its introduction; we keep the signed value in Fig. 4 of the main paper since our goal is to study association trends rather than to construct a nonnegative forgetting score.

Averaged forgetting over old classes. When a single scalar summary is needed, we average per-class forgetting over the old-class set at step t:

$$\bar{F}(t) := \frac{1}{|\mathcal{C}_{1:t-1}|} \sum_{c \in \mathcal{C}_{1:t-1}} f_c(t). \tag{26}$$

L Sensitivity to regularization hyperparameters

Ramp-up schedule. Our full objective combines new-class supervision, online projection, and drift-aware preservation. In practice, we optionally apply a simple linear warm-up on the DWD coefficient to mitigate potential feature instability in the early stage of training, where the newly updated encoder f_t may deviate rapidly from the frozen teacher f_{t-1}. Concretely, for an optimization epoch index $e \in \{0, \ldots, E-1\}$ with a total number of epochs E, we define

$$\gamma(e; r) \;=\; \min\left(1,\; \frac{e+1}{\max(1, \lfloor rE \rfloor)}\right), \tag{27}$$

where $r \in [0, 1]$ is the ramp-up fraction. We then replace the constant DWD weight with a time-dependent coefficient $\lambda_{\text{dwd}}\gamma(e; r)$:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{proj}}\mathcal{L}_{\text{proj}} + \lambda_{\text{cons}}\mathcal{L}_{\text{cons}} + \lambda_{\text{dwd}}\gamma(e; r)\mathcal{L}_{\text{dwd}}. \tag{28}$$

This schedule down-weights DWD in the first rE epochs and gradually increases its influence as features stabilize.

Sensitivity to ramp-up. Table 6 suggests that DWD is not sensitive to the ramp-up fraction r on TinyImageNet. Removing warm-up (r=0) yields performance comparable to the default setting

Table 7: **Projection and consistency coefficient sensitivity** on TinyImageNet. Accuracy is reported in percent. Rows marked † reuse the corresponding non-PTR results from Table 3; other rows were obtained with an additional analysis implementation. All rows retain DWD with coefficient 5.

λ_{proj}	λ_{cons}	T=10		T=20	
		A_{last}	A_{inc}	A_{last}	A_{inc}
0†	1	35.5	49.5	30.0	44.3
0.5	1	34.9	48.9	30.0	43.5
1	1	37.8	**50.8**	30.2	44.6
5†	1	**38.0**	**50.8**	**32.5**	**45.0**
10	1	34.4	48.1	29.2	42.5
5†	0	36.8	49.6	31.1	43.9
5	0.5	36.6	50.6	30.9	44.0
5	2	36.8	50.7	31.5	44.8
0	0	34.2	47.7	28.7	42.7

(r=0.05), and increasing the warm-up duration (r=0.1, 0.3) induces only small changes in both A_{last} and A_{inc}. These point estimates show limited sensitivity to the tested scheduling choices; they do not quantify run-to-run variability.

Sensitivity to λ_{dwd}. Table 6 also evaluates the DWD loss weight under a fixed schedule (r=0.05). Small values such as $\lambda_{\text{dwd}} \in \{0.5, 1\}$ yield lower reported values of both A_{last} and A_{inc}. Increasing λ_{dwd} to 5 provides a strong operating point across task schedules, while λ_{dwd}=10 yields marginal gains on T=10 but slightly degrades T=20. We therefore use λ_{dwd}=5 as the default to balance stability and plasticity.

L.1 Projection and consistency coefficients

We additionally examine the projection-alignment coefficient λ_{proj} and consistency coefficient λ_{cons} on TinyImageNet under the T=10 and T=20 cold-start protocols. The DWD coefficient remains fixed at λ_{dwd}=5, and PTR is disabled throughout. Both the forward projector A and inverse distiller D are retained when a loss coefficient is zero; zero removes that loss contribution without removing the corresponding module.

The three marked rows provide reference values from the main component ablation, including the default $(5, 1)$ and the two individual loss-removal settings. The remaining values come from additional experiments run with an analysis implementation whose added diagnostics changed the execution path relative to the main experiments. Consequently, the table combines reference results with additional analysis results and does not constitute a comparison generated entirely by one implementation. Run counts and standard deviations for these additional settings are not available; the reported point estimates do not quantify statistical uncertainty or establish significance.

The reported values favor the default region over very weak or strong projection regularization. At fixed λ_{proj}=5, changing λ_{cons} from 1 to the other reported values changes A_{inc} by at most 1.2 pp for T=10 and 1.1 pp for T=20. Jointly removing both losses gives low accuracies, although it is not the lowest setting on every metric: $(10, 1)$ has lower T=20 A_{inc}.

These observations provide a descriptive complement to the DWD-strength and ramp-up analyses above. The shared coefficient configuration used in the main benchmarks is an experimental choice; this single-dataset study does not establish unique or universal optimality across datasets, initializations, or representation regimes.

M Choice of drift-to-weight mapping

DWD converts normalized surrogate drift scores into old-class preservation weights. The main experiments use a sigmoid mapping to obtain a raw preservation score,

$$a_c = \sigma(\kappa \tilde{\delta}_c),$$

Table 8: **Alternative drift-to-weight mappings** on CIFAR-100 and TinyImageNet under the 10-task setting. All variants use the same projection module and differ only in how normalized surrogate drift is mapped to old-class weights.

Weighting	CIFAR-100		TinyImageNet	
	A_{last}	A_{inc}	A_{last}	A_{inc}
Sigmoid	**51.7**	**64.4**	**38.0**	**50.8**
Linear	51.6	64.3	37.1	50.3
Temperature-scaled softmax (T=1)	51.2	64.1	37.3	50.4
Temperature-scaled softmax (T=0.5)	50.5	63.8	36.7	49.9

followed by unit-mean normalization over old classes,

$$w_c = \frac{|\mathcal{C}_{1:t-1}|\, a_c}{\sum_{j \in \mathcal{C}_{1:t-1}} a_j}.$$

This normalization keeps the average class coefficient equal to one. It does not fix the teacher-weighted sum $Z(x)$ or the overall loss and gradient magnitudes, as shown in Appendix B. The sigmoid score provides a smooth and bounded transition from low-drift to high-drift classes. High-drift classes receive no smaller coefficients, while all classes retain positive coefficients.

We also evaluate alternative mappings, including a linear mapping and temperature-scaled softmax. Table 8 shows that multiple drift-aware mappings are effective, supporting the main claim that class-wise drift reallocation is beneficial. Sigmoid and linear weighting perform similarly on CIFAR-100, while sigmoid is more consistent on TinyImageNet. Temperature-scaled softmax gives lower point estimates, especially at lower temperature; without repeated-run uncertainty these values do not establish training instability.

Overall, sigmoid is not the only viable choice, but it provides the most consistent trade-off across datasets in our experiments. These comparisons support evaluating online class-wise drift signals, while the effect of weighting sharpness in noisy early stages remains an empirical question.

M.1 Bounded soft weighting and sigmoid sharpness

For normalized drift $\tilde{\delta}_c \in [0, 1]$ and finite $\kappa \geq 0$, the raw score $a_c = \sigma(\kappa \tilde{\delta}_c)$ satisfies

$$\frac{1}{2} \leq a_c \leq \sigma(\kappa), \qquad \frac{\max_c w_c}{\min_c w_c} = \frac{\max_c a_c}{\min_c a_c} \leq 2\sigma(\kappa) < 2.$$

The equality follows because normalization multiplies every score by the same factor. Thus, this mapping provides bounded soft reweighting rather than hard class selection. At $\kappa = 0$, all weights are equal. As κ increases, scores for classes with strictly positive normalized drift saturate toward one, so larger sharpness does not imply indefinitely increasing contrast or hard top-class selection. The coefficient and mapping studies above do not constitute a direct sweep of κ; its numerical sensitivity remains a separate calibration question.

N Random-permutation control for drift weights

To examine whether the assignment of drift-derived weights to classes matters, we include a random-permutation control in Fig. 2c of the main paper. At each task $t > 1$, we sample a fixed random permutation π over the old-class set $\mathcal{C}_{1:t-1}$ once at the beginning of the task. During training, we compute drift weights $\{w_c(t)\}$ from surrogate drift in the same way as the full method and update these values using the same schedule as DWD, but apply them to mismatched classes via $\tilde{w}_c(t) = w_{\pi(c)}(t)$ inside the KD loss. At each update, permutation preserves the computed weight multiset and its unit mean while breaking its alignment with class identity. It does not preserve the teacher-weighted sum $Z(x)$, gradients, or subsequent training trajectory. The comparison tests the importance of class assignment, but does not isolate that effect from changes in effective loss magnitude.

Table 9: **Spearman correlation between surrogate and oracle drift** across incremental steps. Task 1 is the earliest old-class stage and is comparatively noisy. CUB-200 uses the pretrained protocol; its Task-1 measurement is not available and is marked by a dash. Diagnostic task indices start at 0 for the initial task.

Dataset	T1	T2	T3	T4	T5	T6	T7	T8	T9
CIFAR-100	0.03	0.78	0.71	0.89	0.80	0.85	0.89	0.91	0.90
TinyImageNet	-0.29	0.81	0.74	0.72	0.85	0.83	0.88	0.90	0.93
ImageNet-100	0.03	0.83	0.69	0.87	0.79	0.85	0.89	0.92	0.89
CUB-200	–	0.65	0.74	0.71	0.68	0.71	0.73	0.73	0.76

Table 10: **Uniform KD at Task 1.** Replacing DWD by uniform KD only at the first old-class stage produces similar results, suggesting that the overall gains do not require drift weighting at the earliest old-class stage.

Variant	**CIFAR-100**		**TinyImageNet**		**ImageNet-100**	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
ODP	51.7	64.4	38.0	50.8	53.5	67.6
ODP-Task1UniKD	51.8	64.4	38.4	51.0	53.7	67.9

O Cross-dataset surrogate–oracle agreement

The main text provides detailed surrogate–oracle drift analysis on TinyImageNet. To verify that this behavior is not dataset-specific, Table 9 reports the Spearman rank correlation between surrogate drift and oracle drift across incremental steps on CIFAR-100, TinyImageNet, ImageNet-100, and pretrained CUB-200 under the ten-task protocol. Since DWD uses drift primarily as a ranking signal for class-wise reweighting, Spearman correlation is the most relevant measure.

The first reported old-class stage has near-zero or negative correlation; the smaller old-class set and limited accumulated displacement are possible explanations. From Task 2 onward, the rank correlation remains positive across all four datasets. For pretrained CUB-200, the mean over Tasks 2–9 is 0.714 and the final value is 0.76, indicating useful ranking beyond the from-scratch setting. Appendix R extends this evaluation to the more demanding 20-task CUB-200 stream. This supports the use of projection-induced displacement as a practical online drift signal for DWD, even though the theoretical bound itself is not intended as a tight per-epoch empirical equality.

We additionally test whether the noisy first stage is critical by replacing DWD with uniform KD only at Task 1. Table 10 shows that this variant performs similarly to the default method, and in some cases slightly better. This is consistent with a low-signal initial stage; the later reported correlations are higher, without establishing reliability at every training epoch.

P Surrogate–oracle agreement over time

In the main paper, Fig. 3b visualizes the surrogate–oracle relationship at the last step, and Fig. 3c summarizes the Spearman correlation across steps. For completeness, Fig. 5 reports the underlying per-step scatter plots for tasks 1–9 on the same TinyImageNet $T{=}10$ stream. Each point corresponds to one previously learned class; hence, n increases from 20 to 180 as the stream progresses. The first step is comparatively noisy, but from task 2 onward the surrogate drift becomes strongly predictive of oracle drift, matching the trend highlighted in Fig. 3c. This supports using surrogate drift as a reliable drift signal for class-aware weighting without requiring access to oracle drift during training.

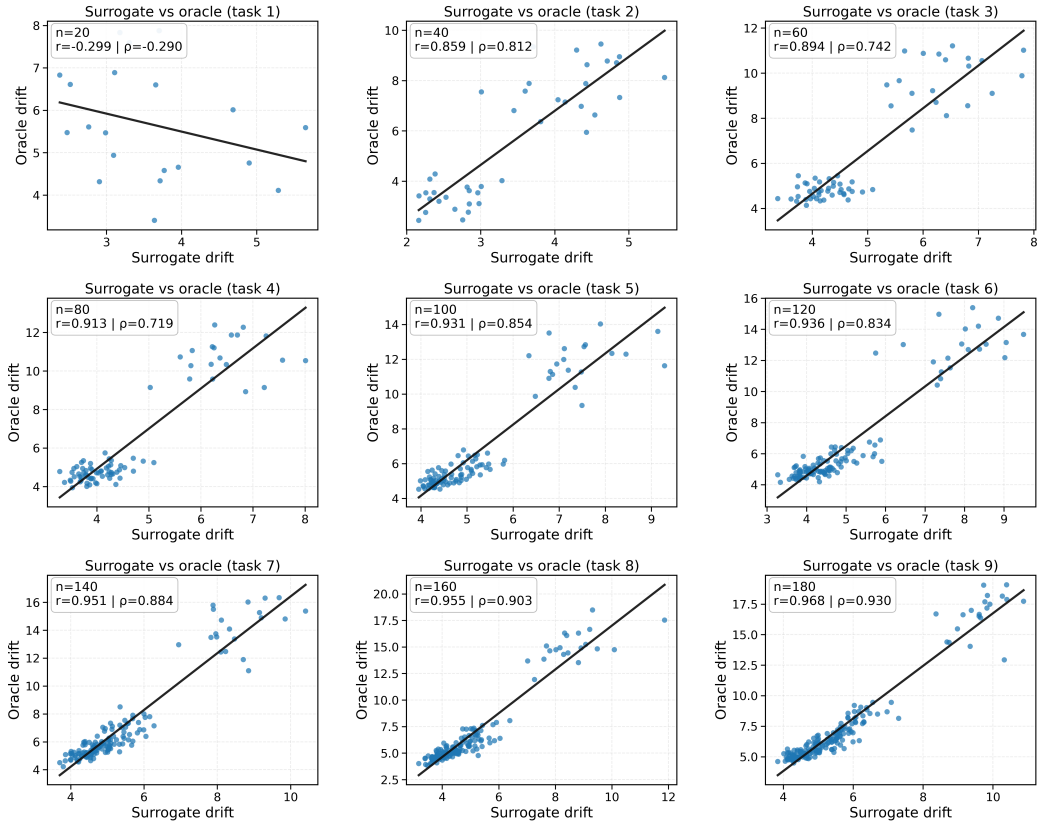


Figure 5: **Surrogate drift vs. oracle drift across incremental steps (TinyImageNet, T=10).** Each panel plots per-class surrogate drift against the corresponding oracle drift for previously seen classes after learning task t. The number of points n equals the number of old classes at that step, and each plot reports Pearson r and Spearman ρ. The task-9 panel reproduces the main-paper snapshot in Fig. 3b; tasks 1–8 are provided here for completeness and to visualize how the surrogate–oracle agreement evolves over the stream, complementing Fig. 3c.

Q Oracle drift is associated with class-wise forgetting

The main paper reports a last-step snapshot in Fig. 4a and a step-wise summary of rank correlation in Fig. 4b. For completeness, Fig. 6 provides the per-step scatter plots underlying Fig. 4b. Across steps, classes with larger oracle drift tend to experience larger forgetting, yielding consistently positive correlations. This association is already visible early in the stream and becomes stronger as more tasks accumulate. These associations motivate drift-aware weighting but do not establish that displacement causes forgetting.

R Regime dependence on CUB-200 under pretrained fine-tuning

CUB-200 uses an ImageNet-pretrained ResNet-18 with a 64-dimensional prototype space and a lower backbone learning rate than the from-scratch benchmarks. As shown in Table 2, ODP is competitive under the 10-task protocol but trails the strongest baselines under the 20-task protocol. We examine whether this difference coincides with a loss of surrogate–oracle agreement or a difference in the association between drift and forgetting. All oracle measurements use held-out old-class data for analysis only.

Surrogate ranking under the 20-task protocol. Table 11 complements the 10-task correlations in Table 9. The initial task is indexed by zero; the reported stages are tasks 2–19. Surrogate–oracle Spearman correlation remains positive throughout these stages, with a mean of 0.679 and a final value

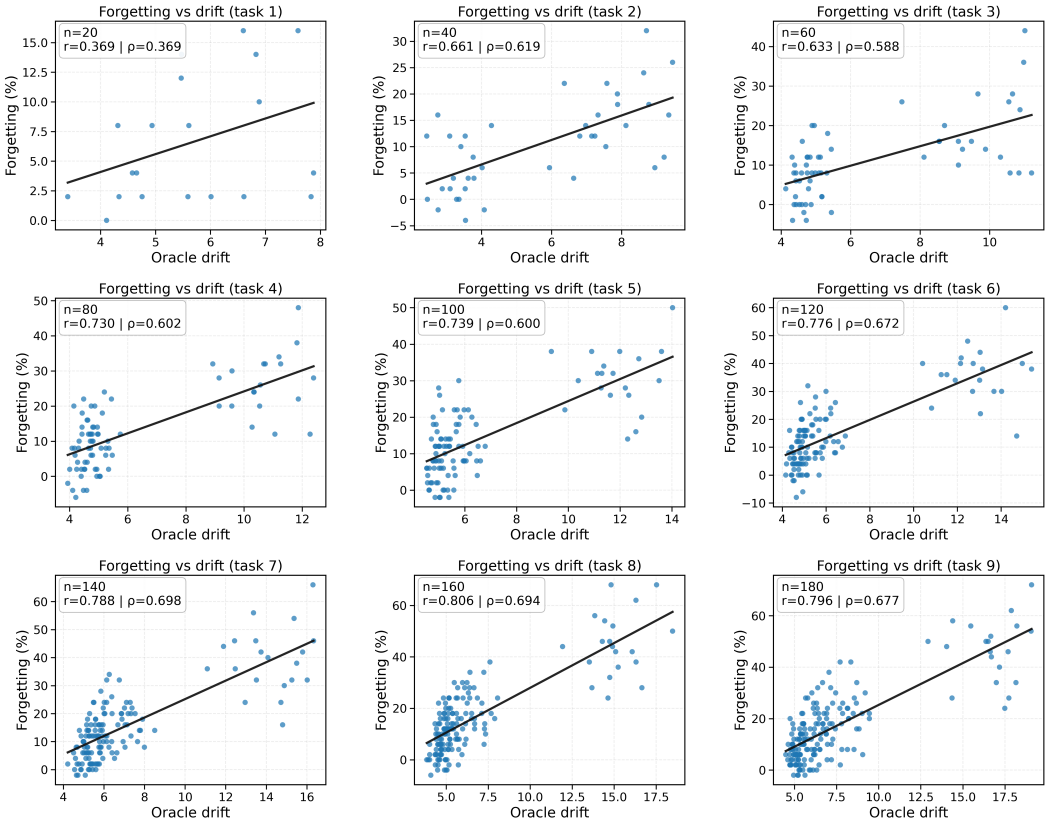


Figure 6: **Forgetting vs. oracle drift across incremental steps (TinyImageNet 10-task stream).** Each panel shows the relationship between per-class oracle drift (x-axis) and per-class forgetting (y-axis) after learning task t. The number of points n equals the number of previously seen classes at that step; each panel reports Pearson r and Spearman ρ. The task-9 panel reproduces the main-paper snapshot in Fig. 4a, while tasks 1–8 visualize how the drift–forgetting association evolves over time, complementing Fig. 4b.

Table 11: Surrogate–oracle Spearman correlation on CUB-200 under the 20-task pretrained protocol.

Task	2	3	4	5	6	7	8	9	10
ρ	0.54	0.50	0.58	0.61	0.65	0.70	0.72	0.71	0.77
Task	11	12	13	14	15	16	17	18	19
ρ	0.75	0.73	0.70	0.74	0.77	0.69	0.66	0.68	0.72

of 0.72. The mean is 0.626 over tasks 2–9 and 0.721 over tasks 10–19. Thus, the lower recognition accuracy does not coincide with a collapse of the measured drift ranking. These correlations assess class ordering rather than exact reconstruction of old-class feature means.

Cross-regime association with forgetting. Table 12 compares the pretrained CUB-200 protocol with the from-scratch CIFAR-100 and TinyImageNet protocols. The table reports the Spearman correlation between final cumulative displacement and forgetting, using the initial-prototype reference defined in Eq. (23). This association is positive on all three datasets but weaker on CUB-200.

Taken together, the positive surrogate–oracle correlations and weaker drift–forgetting association distinguish ranking quality from the relevance of drift to forgetting. They are compatible with less benefit from drift-based reallocation in the pretrained fine-grained regime, but do not establish a causal explanation of the accuracy gap or directly measure the dispersion of DWD weights. Class-mean

Table 12: Cumulative drift–forgetting association across representation regimes. The Spearman coefficient is computed at the final task and measures association with forgetting, rather than surrogate–oracle ranking agreement.

Dataset	Protocol	Final cumulative drift–forgetting ρ
CIFAR-100	From scratch	0.580
TinyImageNet	From scratch	0.727
CUB-200	Pretrained	0.462

Table 13: **ImageNet-1K single-seed sanity check** under a 10-task cold-start stream. These results use each method's closest public configuration and should be interpreted as a supplementary check rather than a fully controlled benchmark.

Method	A_{last}	A_{inc}
ADC	31.34	50.95
LDC	35.15	53.88
DPCR	35.49	54.22
AdaGauss	35.11	54.08
ODP	**36.16**	**54.62**

displacement may also omit covariance changes or local decision boundary movement. Evaluating such distribution- and margin-based signals is a direction for future work.

S IMAGENET-1K EXPERIMENT

The main experiments focus on CIFAR-100, TinyImageNet, ImageNet-100, and CUB-200, which together cover small-scale, medium-scale, higher-resolution, and fine-grained settings. Extending the same cold-start protocol to full ImageNet-1K is substantially more expensive because it increases both the number of classes and the total number of incremental optimization steps. A fully controlled ImageNet-1K benchmark would also require carefully matched hyper-parameters for all baselines, which are not consistently available in public implementations.

As a supplementary check, we ran a single-seed ImageNet-1K experiment using each method's closest public TinyImageNet/ImageNet-100 configuration. Because this setup is not a fully tuned or multi-seed benchmark, we do not include it in the main comparison. The result is reported only as an indicative large-scale sanity check in Table 13.

T SCOPE OF VIT EXPERIMENTS UNDER THE CURRENT PROTOCOL

Our main benchmark uses ResNet-18 trained from scratch, following the directly comparable exemplar-free CIL baselines. Under the tested recipe, ViT-tiny obtains lower accuracy than ResNet-18. Without training-loss diagnostics and matched backbone-specific tuning, this comparison does not establish underfitting or a general limitation of ViTs.

A completely frozen pretrained ViT has no backbone-induced drift; any trainable adapters or prompts can still change its effective representation. Such methods represent a different protocol from our from-scratch comparison. Such pretrained-transformer continual learning is an important direction, but it is outside the from-scratch prototype-based EFCIL setting studied in this work.

U LIMITATIONS AND FUTURE WORK

Limitations. ODP relies on the assumption that prototype displacement induced by the learned projection is a meaningful surrogate for old-class representation drift. Although our mechanistic

Table 14: **Backbone sanity check** on CIFAR-100 under the from-scratch protocol.

Model	A_{last}	A_{inc}
ODP (ResNet-18)	**51.7**	**64.4**
ODP (ViT-tiny)	26.4	37.6

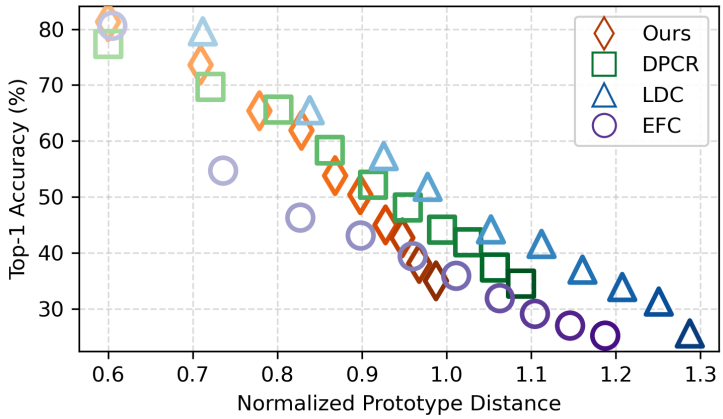


Figure 7: **Top-1 accuracy vs. normalized prototype distance** on TinyImageNet (T=10). Each marker denotes one incremental step, with transparency decreasing over time.

analysis shows strong agreement between surrogate and oracle drift, this relationship may weaken when the old and current feature spaces undergo highly nonlinear or class-dependent transformations that cannot be captured by the chosen projector. ODP improves over AdaGauss and DPCR on the reported from-scratch benchmarks, but does not lead all metrics or protocols; it trails EFC and AdaGauss on CUB-200 at T=20 and FeCAM/EFC under warm start. This suggests that the optimal amount of drift-weighted preservation may depend on the backbone initialization, dataset granularity, and task construction.

Future Work. Next, we will study more expressive yet stable projection mechanisms for estimating drift under nonlinear representation evolution, including low-rank, mixture-based, or locally adaptive transport maps. Another promising direction is to extend drift-weighted preservation beyond class means by incorporating covariance drift, boundary uncertainty, or class-wise confusion structure into the distillation weights. Since ODP currently uses a fixed weighting schedule and global hyperparameters, future variants could adapt the sharpness and strength of DWD online based on validation-free stability–plasticity indicators.

V Additional Drift–Accuracy Analysis

Drift–Accuracy Relationship. To relate drift reduction to recognition performance, we further examine Top-1 accuracy versus a normalized prototype distance on TinyImageNet (T=10), computed on fixed task-0 data. At each incremental step, we compute the mean normalized Euclidean distance between the current class means under f_t and the initial task-0 prototypes, and evaluate Top-1 accuracy on the same class set. As shown in Fig. 7, larger prototype distance is consistently associated with lower accuracy. The displayed ODP trajectory ends at smaller distance and higher accuracy than the compared methods. This descriptive comparison is consistent with a drift-related explanation but does not isolate a causal effect.

W WARM-START EVALUATION

The main experiments use a cold-start stream with equally sized tasks. We additionally evaluate a warm-start protocol in which the initial task contains 50% of the classes and the remaining classes are introduced in ten equal increments. This gives eleven stages in total: $50 + 10 \times 5$ classes for CIFAR-100 and $100 + 10 \times 10$ classes for TinyImageNet. It is therefore distinct from the ten-task, equally partitioned protocol in Table 1. All methods use their released configurations without warm-start-specific retuning. In addition to the external baselines, we evaluate the ODP pipeline with uniform old-class KD to isolate the contribution of drift weighting within this setting.

Table 15: **Warm-start evaluation** with half of the classes in the initial task followed by ten increments. Values are accuracies in percent. ODP with uniform KD retains the same projection pipeline and removes class-wise drift weighting. Point estimates are shown; the table does not establish statistical significance.

	CIFAR-100		**TinyImageNet**	
Method	A_{last}	A_{inc}	A_{last}	A_{inc}
FeCAM	**62.1**	**68.9**	**50.8**	**57.7**
EFC	60.9	68.2	50.4	57.5
ADC	41.9	54.7	25.3	34.6
LDC	43.8	55.3	26.1	35.4
AdaGauss	55.1	62.0	45.8	54.1
DPCR	45.5	57.0	24.6	33.5
ODP with uniform KD	57.7	64.8	47.0	54.8
ODP	58.6	65.9	47.5	55.2

Table 15 shows that DWD improves over the otherwise identical uniform-KD variant by $0.9/1.1$ percentage points in $A_{\text{last}}/A_{\text{inc}}$ on CIFAR-100 and $0.5/0.4$ points on TinyImageNet. These gains are smaller than the DWD-versus-non-DWD differences under the cold-start protocol in Table 3, although that component comparison does not by itself identify a matched uniform-KD control. ODP outperforms ADC, LDC, AdaGauss, and DPCR on both metrics and datasets, while FeCAM and EFC obtain higher accuracy in this regime. Thus, the advantage of drift-aware allocation depends on initialization and task construction. A broader initial representation is a plausible reason for the reduced benefit, but this accuracy comparison alone does not measure teacher stability or establish a causal relationship with drift magnitude.

Table 16: **One-step projection endpoint error.** The first two numeric columns report task-averaged normalized class statistics. The last three columns summarize raw endpoint norms $e_c(t)$ across old classes at the final step: mean, median, and 90th percentile (P90). All measurements use held-out old-class data for analysis only.

Dataset	Task-averaged normalized error		Final-step raw $e_c(t)$		
	$r_c(t)$	$s_c(t)$	Mean	Median	P90
CIFAR-100	0.470	0.230	0.827	0.786	1.006
TinyImageNet	0.600	0.291	1.008	0.989	1.183
CUB-200 (T=20)	0.696	0.340	2.561	2.420	3.398

X ONE-STEP PROJECTION ENDPOINT ERROR

We evaluate how accurately the forward projector predicts old-class feature means after a single task update. For an old class c, let $\mu_{c,\mathrm{ora}}^{t-1}$ and $\mu_{c,\mathrm{ora}}^{t}$ be oracle means estimated from held-out examples embedded by f_{t-1} and f_t, respectively. Both endpoints are computed strictly for analysis: these held-out examples and their statistics are not used to train the projector, construct DWD weights, or update the maintained prototypes. For the projector A_t learned during task t, define

$$e_c(t) := \left\| A_t\big(\mu_{c,\mathrm{ora}}^{t-1}\big) - \mu_{c,\mathrm{ora}}^{t} \right\|_2, \tag{29}$$

$$\delta_c^{\mathrm{ora,orig}}(t) := \left\| \mu_{c,\mathrm{ora}}^{t} - \mu_{c,\mathrm{ora}}^{t-1} \right\|_2, \tag{30}$$

$$\delta_c^{\mathrm{sur,orig}}(t) := \left\| A_t\big(\mu_{c,\mathrm{ora}}^{t-1}\big) - \mu_{c,\mathrm{ora}}^{t-1} \right\|_2. \tag{31}$$

The superscript orig distinguishes these oracle-origin quantities from the cache-origin diagnostics in Appendix H. Here the surrogate is evaluated from the oracle origin specifically to diagnose the current map. The normalized and symmetric errors are

$$r_c(t) := \frac{e_c(t)}{\delta_c^{\mathrm{ora,orig}}(t) + \varepsilon}, \qquad s_c(t) := \frac{e_c(t)}{\delta_c^{\mathrm{sur,orig}}(t) + \delta_c^{\mathrm{ora,orig}}(t) + \varepsilon}, \tag{32}$$

where $\varepsilon > 0$ is a numerical stabilizer. The symmetric denominator reduces sensitivity to near-zero oracle drift when the predicted displacement remains nonzero. It does not imply exact endpoint reconstruction.

Table 16 reports task-averaged normalized errors and final-step raw errors. The nonzero errors confirm that the learned map is approximate. Normalized errors permit comparison without equating absolute feature distances across protocols; their values depend on both endpoint error and the drift used for normalization. The final-step raw norms document endpoint calibration within each protocol and should not be compared directly across datasets with different feature scales.

This diagnostic isolates one-step mapping error from accumulated transport error. During ordinary incremental learning, cached prototypes may have undergone several previous transports. Applying A_t to such a maintained prototype also reflects historical approximation errors; that quantity is distinct from Eq. (29), whose origin is recomputed under the previous encoder.

Theorem 1 is conditional on bounded projection error, rather than asserting that this error is universally small. For the oracle-origin quantities above, the reverse triangle inequality likewise gives $|\delta_c^{\mathrm{sur,orig}}(t) - \delta_c^{\mathrm{ora,orig}}(t)| \le e_c(t)$. Endpoint calibration and rank agreement are complementary measurements: DWD needs an informative class ordering, which can coexist with imperfect reconstructed means. The correlation analyses assess that ordering separately; the endpoint statistics alone do not establish it or guarantee performance in every representation regime.