TRAJECTORY REPLAY: CONTINUOUS-TIME MIXTURE TRANSPORT FOR LIFELONG GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting pre-trained continuous-time generative models to sequentially incoming data without forgetting earlier tasks is a central challenge in lifelong learning. A leading strategy, generative replay, uses a generative model trained on historical tasks to augment its own continual update with synthetic samples from experience. However, it requires full trajectory simulations at every step that accumulate discretization errors and can degrade the historical distribution. Here, we propose *trajectory replay*, which bypasses sample generation by training an updated velocity field to match both a frozen historical velocity along its trajectories and the incoming task's velocity field, weighted by a stability-plasticity coefficient. Through the lens of continuous-time transport, we show the resulting optimum averages the two task velocities by their respective densities and generates the mixture of their probability paths. Combined with mean velocity distillation, trajectory replay accesses historical trajectories using only a few network evaluations per continual update. Empirically, it outperforms generative replay and distillation baselines on class-incremental image generation. In two biological case studies, trajectory replay extends generative perturbation models to new contexts and experimental batches across two data modalities while preserving treatment-effect prediction.

1 INTRODUCTION

Continuous-time generative models, such as diffusion and flow matching (Song et al., 2021; Lipman et al., 2023), have gained traction across image generation (Dhariwal & Nichol, 2021; Esser et al., 2024), language modeling (Li et al., 2022; Sriram et al., 2024), and scientific applications (Morehead et al., 2026). Trained on large corpora of data to capture complex, multi-modal distributions, they yield reusable generative priors for a range of downstream tasks (Rombach et al., 2022). As novel data accumulates, however, effective strategies are needed to update pre-trained models with incoming knowledge while preserving generative capabilities over the historical distribution. Joint retraining on the union of historical and incoming data is often computationally prohibitive as models and datasets scale, and infeasible when the original data is no longer accessible.

Incorporating new data without full retraining motivates continual learning (CL) strategies (Wang et al., 2024), whereby a seed model is updated on incoming data streams while mitigating forgetting of previous tasks. In generative modeling, a *task* means sampling from a novel data distribution absent from the historical dataset (Guo et al., 2025). CL approaches applied to generative models include architectural evolution (Smith et al., 2024) and regularization (Song et al., 2024). Among these, generative replay (Shin et al., 2017) is one of the most established (Van de Ven et al., 2022): the objective over incoming tasks is augmented with samples generated by a model over the historical distribution, alleviating catastrophic forgetting without storing or revisiting historical data.

Applying standard generative replay to continuous-time frameworks is expensive, as each replay sample requires iterative simulation of the underlying dynamics. Approximation errors compound along these trajectories, degrading replay fidelity and, in turn, subsequent CL updates (Gao & Liu, 2023). One line of work stabilizes replay by renoising generated samples and regressing the frozen historical velocity field at those points, rather than using them directly as training data (Masip et al., 2025), but generation is still required. Cheaper solutions that bypass data generation remain underexplored. Beyond this bottleneck, most work on CL for continuous-time models remains empirical and lacks a formal understanding of the distributional properties induced by sequential updates.

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

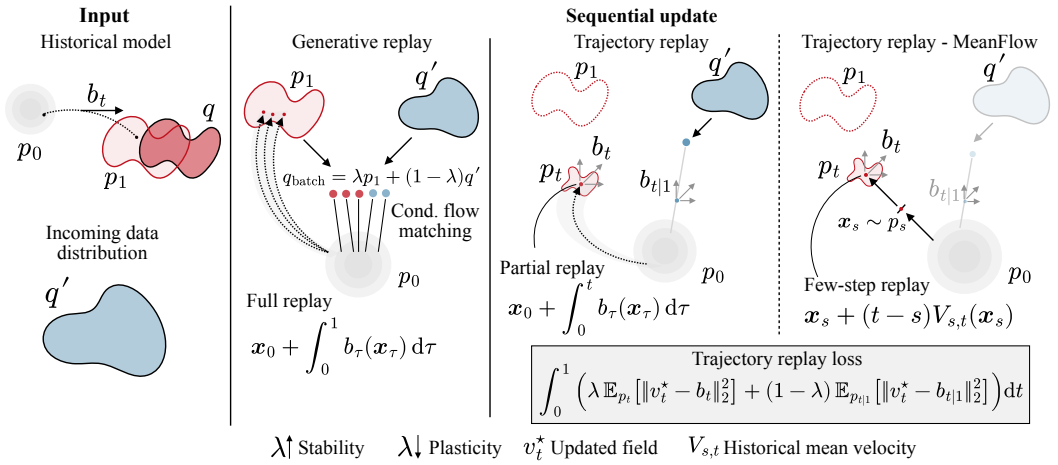

Figure 1: **Trajectory replay for generative CL.** Given a historical flow model and an incoming task (left), generative replay simulates full trajectories to draw replay samples and keeps training on the mixture q_{batch}, trajectory replay instead regresses a single field against the historical field b_t and the incoming task's data-conditioned velocity $b_{t|1}$ along their own paths, and the MeanFlow variant optimizes the same objective with intermediate states drawn through the historical mean velocity $V_{s,t}$.

To address these shortcomings, we introduce *trajectory replay*, a flow-based framework for generative replay for continuous-time models. Given a batch of incoming data and a vector field trained to generate the historical task distribution, our method splits the CL-update objective for a joint-task velocity field into two regression terms. The first is a standard conditional flow matching loss (Lipman et al., 2023) over the incoming data. The second regresses the joint-task velocity toward the frozen historical field along its own generative trajectory at intermediate times, so replay never runs through to fully generated samples. A stability-plasticity coefficient weights the two terms and controls how much mass the optimum assigns to the historical and the incoming density (see Fig. 1).

We validate trajectory replay in controlled simulations and class-incremental image-generation benchmarks, where it improves sample fidelity and retention of earlier tasks across sequential updates at a fraction of the network evaluations of existing CL strategies. Since continual adaptation is particularly important in biological discovery, where measurements arrive sequentially across cell lines, donors, and experimental batches, and where historical screens may be inaccessible, we evaluate two drug-discovery applications of generative perturbation prediction in single-cell gene expression and microscopy. Across these settings, trajectory replay offers the best stability–plasticity balance and generalizes to held-out conditions. In summary, our contributions are as follows:

- **Trajectory-level replay objective.** We formulate generative replay as velocity-field regression along the historical model's generative trajectories, and derive a matching continual update rule for its mean velocity (Geng et al., 2026a), reducing each update to fewer network evaluations.
- **Optimality and stability-plasticity control.** We show the population optimum of our trajectory objective is a density-weighted mixture of the historical and incoming probability paths, with the mixture weight directly controlling the resulting stability-plasticity trade-off of the update.
- **Gains across scales and domains.** Trajectory replay improves the fit-retention-generalization trade-off over baselines on simulations, image benchmarks, and two biological case studies.

2 Background and Preliminaries

2.1 Continuous-time generative flows

Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote a continuous data space of dimensionality d, and define $q \in \mathcal{P}(\mathcal{X})$, where $\mathcal{P}(\mathcal{X})$ is the space of probability density functions supported on $\mathcal{X}$. Continuous normalizing flows sample from a probability path $\{p_t\}_{t\in[0,1]}$, where $p_t \in \mathcal{P}(\mathcal{X})$, transitioning from a tractable prior

$p_0 := \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ to a terminal density p_1 approximating q (Chen et al., 2018). Here, p_t is generated by a *marginal velocity* $b_t : \mathcal{X} \to \mathbb{R}^d$ according to the continuity equation $\partial_t p_t = -\nabla \cdot (p_t b_t)$, so that learning the generative process reduces to regressing b_t with a parameterized function b_t^θ.

Flow matching. A prominent framework for learning these paths is flow matching (Lipman et al., 2023), which optimizes a vector field b_t^θ to approximate b_t across time via the following objective:

$$\mathcal{L}_{\text{FM}}(\theta) := \int_0^1 \mathbb{E}_{\boldsymbol{x}\sim p_t} \left[\|b_t^\theta(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2 \right] \mathrm{d}t \,. \tag{1}$$

As $b_t(\boldsymbol{x})$ is intractable, Lipman et al. (2023) instead minimize its tractable conditional counterpart:

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathbb{E}_{t\sim\mathcal{U}[0,1],\, \boldsymbol{x}_1\sim q,\, \boldsymbol{x}\sim p_{t|1}(\cdot|\boldsymbol{x}_1)} \left[\|b_t^\theta(\boldsymbol{x}) - b_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\|_2^2 \right] , \tag{2}$$

where $p_{t|1}(\cdot \mid \boldsymbol{x}_1) := \mathcal{N}(\alpha_t \boldsymbol{x}_1, \sigma_t^2 \mathbb{I}_d)$ is a Gaussian conditional path with scheduling parameters α_t, σ_t, and $b_{t|1}$ its associated conditional velocity. Crucially, Eq. (1) and Eq. (2) differ only by a term independent of θ, and therefore have identical gradients with respect to θ (details in Sec. A.1).

Mean flow distillation. Denote with $\boldsymbol{x}_t$ the integral solution of an ODE with marginal velocity b_t:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_t = b_t(\boldsymbol{x}_t), \qquad \boldsymbol{x}_t = \boldsymbol{x}_0 + \int_0^t b_\tau(\boldsymbol{x}_\tau)\, \mathrm{d}\tau \,, \tag{3}$$

where $\boldsymbol{x}_0 \sim p_0$, so $\boldsymbol{x}_t \sim p_t$ for all $t \in [0,1]$ (Sec. A.1). As solving Eq. (3) requires many discretization steps and is therefore expensive, recent work learns a *mean velocity* (Geng et al., 2026a;b), $V_{s,t}(\boldsymbol{x}_s) := \frac{1}{t-s}\int_s^t b_\tau(\boldsymbol{x}_\tau)\mathrm{d}\tau$, averaging b_τ over the interval $[s,t]$ for $0 \le s < t \le 1$ and $\boldsymbol{x}_s \sim p_s$. Differentiating this definition with respect to s yields the *MeanFlow Identity*, whose residual

$$\mathcal{R}_{s,t}(V,b)(\boldsymbol{x}) := V_{s,t}(\boldsymbol{x}) - \mathrm{sg}\Big[b_s(\boldsymbol{x}) + (t-s)\big(\partial_s V_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} V_{s,t}(\boldsymbol{x})\, b_s(\boldsymbol{x})\big)\Big] \tag{4}$$

vanishes exactly at the mean velocity of b, meaning $\mathcal{R}_{s,t}(V,b) = \mathbf{0}$ whenever $V = V_{s,t}$ as defined above. Here $\mathrm{sg}[\cdot]$ is the stop-gradient, so differentiating only outside it yields a *semigradient* rather than a true gradient (Sec. A.3). Minimizing $\|\mathcal{R}_{s,t}(V^\theta, b)\|_2^2$ thus gives a tractable objective for a parameterized field $V_{s,t}^\theta$, trained end-to-end or by distillation from a teacher field b (Sec. A.2). $V_{s,t}$ also defines a *flow map* $\boldsymbol{X}_{s,t}(\boldsymbol{x}_s) := \boldsymbol{x}_s + (t-s)V_{s,t}(\boldsymbol{x}_s)$, which maps a state at s directly to its corresponding state at t, satisfying $p_t = (\boldsymbol{X}_{s,t})_\sharp p_s$ (Boffi et al., 2025). Substituting the learned $V_{s,t}^\theta$ for $V_{s,t}$ in $\boldsymbol{X}_{s,t}$ therefore allows sampling in a few discrete steps, without simulating Eq. (3).

2.2 Continual learning and replay for generative models

Problem statement. We consider a CL paradigm where a generative model trained on an initial data distribution is extended to sequentially arriving data streams. We illustrate our setting across a single update step, though our framework extends to multiple iterations. Denote $q \in \mathcal{P}(\mathcal{X})$ as the historical density, the ground-truth distribution of the initial task dataset before sequential updates, and $q' \in \mathcal{P}(\mathcal{X})$ as the incoming data's ground-truth density for the subsequent CL step. The goal is to approximate a *target density* $\tilde{q} \in \mathcal{P}(\mathcal{X})$ that preserves the ability to sample from both, mitigating catastrophic forgetting on the old task while ensuring accurate adaptation to incoming information.

Generative replay. Consider a flow model $\{p_t\}_{t\in[0,1]}$ approximating q at $t = 1$ (Sec. 2.1). *Naive fine-tuning* on samples from q' leads to forgetting, as the velocity field is updated only where the incoming data lies and no longer transports noise onto q. Generative replay (Shin et al., 2017) mitigates this by augmenting training batches with samples generated from p_1 at rate $\lambda \in (0,1)$, modulating the *stability-plasticity* trade-off that governs how far the model may deviate from the historical task. However, generating each replay sample requires simulating the full generative trajectory illustrated in Eq. (3), making this augmentation expensive and susceptible to discretization errors.

3 Trajectory replay for continual learning

First, we characterize the terminal distribution a model generates after a replay update. Training on replay-augmented batches with the objective in Eq. (2) optimizes a vector field whose updated terminal marginal approximates $\tilde{q}$, the mixture of historical and incoming densities (details in Sec. A.4):

$$\tilde{q}(\boldsymbol{x}) = \lambda q(\boldsymbol{x}) + (1-\lambda) q'(\boldsymbol{x}). \tag{5}$$

This mixture behavior admits a variational justification: modeling CL updates as a λ-weighted mixture is equivalent to minimizing a forward Kullback-Leibler divergence objective. Specifically, one can show that the $\tilde{q}$ in Eq. (5) is the unique global minimizer of the following optimization problem:

$$\arg\min_{\rho \in \mathcal{P}(\mathcal{X})} \lambda D_{\mathrm{KL}}(q(\boldsymbol{x}) \parallel \rho(\boldsymbol{x})) + (1-\lambda) D_{\mathrm{KL}}(q'(\boldsymbol{x}) \parallel \rho(\boldsymbol{x}))\,. \tag{6}$$

The formalization and proof are in Sec. A.5. Since the forward $D_{\mathrm{KL}}(q \parallel \rho)$ penalizes low mass where q has support, its minimizer is mode-covering, discouraging collapse onto either task and thus retaining history while adapting to incoming data, as desired for continual updates. We develop a more efficient continuous-time solution for such mixture-based updates at the trajectory level.

3.1 A CONTINUOUS-TIME TRANSPORT FRAMEWORK FOR GENERATIVE CL

Task. We consider a historical flow generating p_t via velocity field b_t and an incoming one generating p'_t via b'_t (Sec. 2.2). We aim to find a path $\rho^\star_t$ mixing these two generative trajectories and recovering Eq. (5) at $t = 1$. To this end, we pose an optimization problem over a time-dependent velocity field $v := \{v_t\}_{t\in[0,1]}$, with $v_t : \mathbb{R}^d \to \mathbb{R}^d$, tasked with generating $\rho^\star_t$. We write $b := \{b_t\}_{t\in[0,1]}$, $b' := \{b'_t\}_{t\in[0,1]}$ for the respective families of historical and incoming task velocities.

Standing assumptions. Throughout this section, $p_t, p'_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ are conditional Gaussian paths with shared scheduling parameters α_t, σ_t satisfying $\alpha_t, \sigma_t > 0$ and $\dot{\alpha}_t\sigma_t - \alpha_t\dot{\sigma}_t > 0$ on $(0,1)$. They are generated through the continuity equation of Sec. 2.1 by velocity fields $b_t, b'_t \in C^0([0,1], (C^1(\mathbb{R}^d))^d)$, from the shared prior $p_0 = p'_0 := \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ to densities $p_1 := q$ and $p'_1 := q'$. We fix $\lambda \in (0,1)$ and assume $\lambda p_\tau + (1-\lambda)p'_\tau > 0$ on $[0,1] \times \mathcal{X}$.

Trajectory replay target. Under these assumptions, and taking v in the same regularity class as b and b', we define the *trajectory replay target* $v^\star$ as the minimizer of a two-term regression objective

$$\mathcal{J}(v; b, b') := \int_0^1 \Big(\lambda\, \mathbb{E}_{p_t}\big[\|v_t(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2\big] + (1-\lambda)\, \mathbb{E}_{p'_t}\big[\|v_t(\boldsymbol{x}) - b'_t(\boldsymbol{x})\|_2^2\big]\Big)\,\mathrm{d}t\,, \tag{7}$$

so that $v^\star = \arg\min_v \mathcal{J}(v; b, b')$. Each term regresses v_t onto one task velocity under that task's marginal, with λ weighing the old and new contributions. The objective admits a unique minimizer.

Proposition 3.1. *Under the standing assumptions, $\mathcal{J}(\cdot\,; b, b')$ in Eq. (7) admits*

$$v^\star_t(\boldsymbol{x}) := \frac{\lambda\, p_t(\boldsymbol{x})\, b_t(\boldsymbol{x}) + (1-\lambda)\, p'_t(\boldsymbol{x})\, b'_t(\boldsymbol{x})}{\lambda\, p_t(\boldsymbol{x}) + (1-\lambda)\, p'_t(\boldsymbol{x})}\,, \tag{8}$$

as unique minimizer, which generates the mixture path $\rho^\star_t := \lambda p_t + (1-\lambda)p'_t$, satisfying $\partial_t \rho^\star_t + \nabla_{\boldsymbol{x}} \cdot (v^\star_t \rho^\star_t) = 0$ with $\rho^\star_0 = \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ and $\rho^\star_1 = \tilde{q}$ (Eq. (5)).

The proof is in Sec. A.6. The optimum is a pointwise average of the two task velocities, weighted by their densities and by λ, generating a path $\rho^\star_t$ that interpolates the two trajectories and reduces to $\tilde{q}$ at $t = 1$. (**I**) λ fixes how much mass the updated model retains on the historical task relative to the incoming one. (**II**) Since both task paths share the Gaussian kernel $p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}; \alpha_t \boldsymbol{x}_1, \sigma_t^2 \mathbb{I}_d)$, $\rho^\star_t(\boldsymbol{x}) = \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, \tilde{q}(\boldsymbol{x}_1)\, \mathrm{d}\boldsymbol{x}_1$, so $\rho^\star_t$ is itself a conditional Gaussian path over $\tilde{q}$ (see Sec. A.7).

KL-optimality. The trajectory objective in Eq. (7) connects back to the static KL problem in Eq. (6). Evaluating the latter at the endpoints of the generative paths gives the weighted terminal divergence $\lambda D_{\mathrm{KL}}(p_1 \parallel \rho_1) + (1-\lambda) D_{\mathrm{KL}}(p'_1 \parallel \rho_1)$, measuring how well a candidate path ρ_t reproduces the two tasks at $t = 1$, which admits an exact trajectory-level expression.

Theorem 3.2. *Under the standing assumptions, let ρ_t be a candidate path from the same conditional Gaussian family, with velocity v_t and shared prior $\rho_0 = p_0$, and define $\kappa_t := \alpha_t / \big(\sigma_t(\dot{\alpha}_t\sigma_t - \alpha_t\dot{\sigma}_t)\big) > 0$. The weighted terminal divergence then satisfies*

$$\begin{aligned} &\lambda D_{\mathrm{KL}}(p_1 \parallel \rho_1) + (1-\lambda) D_{\mathrm{KL}}(p'_1 \parallel \rho_1) \\ &\quad = \int_0^1 \kappa_t \Big(\lambda\, \mathbb{E}_{p_t}\big[\|v_t(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2\big] + (1-\lambda)\, \mathbb{E}_{p'_t}\big[\|v_t(\boldsymbol{x}) - b'_t(\boldsymbol{x})\|_2^2\big]\Big)\mathrm{d}t\,. \end{aligned} \tag{9}$$

The proof is in Sec. A.8 and follows Albergo et al. (2025). Since $\kappa_t > 0$ reweights Eq. (7) in time without moving its pointwise minimizer, both share $v^\star$, whose path stays in the same conditional Gaussian family (Sec. A.7) and is thus admissible in Thm. 3.2. The same construction extends to *stochastic generative models*, recovering the same density-weighted mixture (Sec. A.10).

Mean velocity continual update. Evaluating Eq. (7) requires samples from p_t and p'_t, hence integrating Eq. (3) along both paths. At each continual update, we can therefore additionally learn a mean velocity $V^\star_{s,t}$ alongside the mixture velocity $v^\star_t$, averaging $v^\star$ over $[s,t] \subseteq [0,1]$:

$$V^\star_{s,t}(\boldsymbol{x}_s) := \frac{1}{t-s}\int_s^t v^\star_\tau(\boldsymbol{x}^\star_\tau)\,\mathrm{d}\tau \qquad \text{with} \qquad \boldsymbol{x}^\star_s = \boldsymbol{x}_s\,, \tag{10}$$

where $\boldsymbol{x}^\star_\tau$ solves Eq. (3) with velocity $v^\star$ from Prop. 3.1. Because the mixture path $\rho^\star_t$ becomes the historical path at the next sequential update, the flow map $\boldsymbol{X}^\star_{s,t}$ induced by $V^\star_{s,t}$ can supply the samples that the update requires in a few network evaluations rather than a full integration (Sec. 3.2). Writing $V := \{V_{s,t}\}_{0\le s\le t\le 1}$ for a candidate mean velocity, the counterpart of Eq. (7) at the level of mean velocities weights the consistency residuals of Eq. (4) along the two task paths:

$$\mathcal{K}(V; b, b') := \int_0^1\int_s^1 \Big(\lambda\,\mathbb{E}_{p_s}\big[\|\mathcal{R}_{s,t}(V,b)(\boldsymbol{x})\|_2^2\big] + (1-\lambda)\,\mathbb{E}_{p'_s}\big[\|\mathcal{R}_{s,t}(V,b')(\boldsymbol{x})\|_2^2\big]\Big)\,\mathrm{d}t\,\mathrm{d}s\,. \tag{11}$$

Proposition 3.3. *Under the standing assumptions, let V be continuously differentiable in s and $\boldsymbol{x}$ on $0 \le s \le t \le 1$. Then the semigradient of $\mathcal{K}(\cdot\,; b, b')$ vanishes if and only if V satisfies the MeanFlow identity for $v^\star$, that is $\mathcal{R}_{s,t}(V, v^\star) = 0$ for all $0 \le s < t \le 1$, with $v^\star$ from Eq. (8), whose unique solution is the mixture mean velocity $V^\star_{s,t}$ of Eq. (10).*

The proof is in Sec. A.9. Prop. 3.3 mirrors Prop. 3.1 one level up: regressing a single mean velocity against the two task residuals recovers the mean velocity of the mixture path, with the density weighting emerging rather than imposed. Since Eq. (11) involves only the instantaneous fields b_s, b'_s, the mixture mean velocity can be learned jointly with $v^\star$, without ever forming $v^\star$ explicitly.

3.2 Trajectory replay in practice

Sequential updates. Our framework extends to a stream of tasks $k \in \{1,\dots,K\}$ with densities q_k, where update k targets a mixture between the k tasks accumulated so far and the incoming task $k{+}1$. At each step we have only a model for the historical tasks, $b_t^{\theta^{\le k}}$, and only data for the incoming one, so the historical term regresses onto the previous model while the incoming term uses the *data-conditioned velocity*, which has the same gradient (Sec. 2.1). Each update trains a joint field $v_t^{\theta^{\le k+1}}$, which becomes $b_t^{\theta^{\le k+1}}$ at the next step, with the first task trained by flow matching alone (see Alg. 1):

$$\mathcal{J}^{k+1} = \lambda_k\,\mathbb{E}\big[\|v_t^{\theta^{\le k+1}}(\boldsymbol{x}_t) - b_t^{\theta^{\le k}}(\boldsymbol{x}_t)\|_2^2\big] + (1-\lambda_k)\,\mathbb{E}\big[\|v_t^{\theta^{\le k+1}}(\boldsymbol{x}) - b_{t|1}(\boldsymbol{x}\mid\boldsymbol{x}_1)\|_2^2\big]\,, \tag{12}$$

with $t \sim \mathcal{U}(0,1]$, $\boldsymbol{x}_t$ obtained by integrating Eq. (3) from $\boldsymbol{x}_0 \sim p_0$ under the frozen $b^{\theta^{\le k}}$ only as far as t, and $\boldsymbol{x}_1 \sim q_{k+1}$, $\boldsymbol{x} \sim p_{t|1}(\cdot \mid \boldsymbol{x}_1)$. In expectation, stopping early costs *half the evaluations* that generative replay needs, plus one for the target at the anchor. It also changes where discretization error lands: the frozen model supplies the velocity at whatever state the solver reaches, so the error only shifts which states the student sees rather than entering the loss as corrupted data (Sec. A.13).

Efficient sampling along the trajectory. Eq. (12) still integrates Eq. (3) at every training step, at a cost set by the solver. The *MeanFlow backend* removes that dependence: alongside $v^{\theta^{\le k+1}}$ it trains a mean velocity $V^{\theta^{\le k+1}}$ under the counterpart $\mathcal{K}^{k+1}$ of Eq. (11) with the same velocity teachers, so the model entering update k carries a flow map $\boldsymbol{X}_{s,t}^{\theta^{\le k}}$. Both fields are therefore passed forward together: the velocity field as the next update's regression target and the mean velocity as the sampler supplying its historical states. Composing that map over N sub-intervals of $[0,t]$ from $\boldsymbol{x}_0 \sim p_0$ reaches $\boldsymbol{x}_t$ in N evaluations regardless of t, against the $\lceil t/h \rceil$ spent by an Euler solver at step size h (see Alg. 2).

Choosing λ_k. Each value of λ_k fixes a different optimum to Eq. (8), striking a different balance between fitting the incoming task and forgetting earlier ones, so the best choice *depends on the metric at hand*. Holding λ_k constant is rarely appropriate, as task weights then decay geometrically with

age rather than balancing (Sec. A.12). We consider two schedules. *Equal-task* weighting sets $\lambda_k = k/(k+1)$, leaving every task seen so far with equal mass and needing no tuning. *Adaptive* weighting instead sweeps λ at each update and scores every candidate on old-task and new-task validation sets, which the CL protocol permits, picking the best point on the retention-adaptation trade-off.

4 Experiments

4.1 Controlled mixture recovery

Task. Before testing trajectory replay at scale, we verify in simulation that the trained field recovers the task mixture $\tilde{q} = \lambda q + (1-\lambda)q'$ predicted by Prop. 3.1. A bimodal Gaussian mixture lets us evaluate this recovery directly, since the ground-truth mixture weight and mode geometry are known exactly.

Setup. In $d = 2$, a historical flow model pretrained on $q := \mathcal{N}(\boldsymbol{\mu}_0, \sigma^2 \mathbb{I}_2)$ is updated with samples from $q' := \mathcal{N}(\boldsymbol{\mu}_1, \sigma^2 \mathbb{I}_2)$, where $\boldsymbol{\mu}_0 := (-1.5, 0)$, $\boldsymbol{\mu}_1 := (1.5, 0)$, $\sigma := 0.35$, 10,000 samples per task, three seeds throughout. Across the grid $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, we measure how closely the generated mass, computed by assigning samples to their nearest mode mean, matches λ, and how well mode shape is preserved, with joint training on the true mixture as an oracle reference (see Sec. D.1.1).

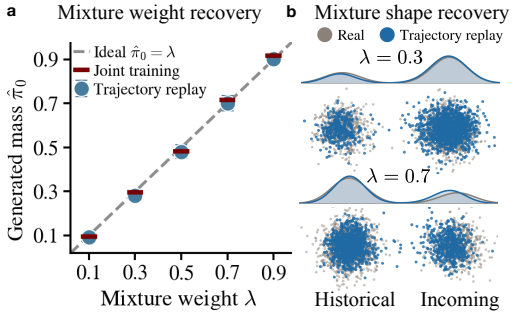

Figure 2: **(a)** Generated historical-mode mass versus mixture weight λ for trajectory replay and joint training (mean ± SD, three seeds). **(b)** Target and generated samples at $\lambda \in \{0.3, 0.7\}$. Historical and incoming modes are on the left and right.

Results. Trajectory replay recovers the prescribed historical mass across all λ, matching joint training and the ideal calibration line, and preserves mode location and geometry while shifting relative mass as prescribed (Fig. 2a,b), more cheaply than generative replay in higher dimensions (Fig. 7). Naive fine-tuning instead collapses onto the incoming mode, recovering no historical mass (Fig. 6b).

4.2 Sequential updates on a ring-structured Gaussian mixture

Task. We extend to a five-task stream of separated Gaussian modes with means arranged on a ring, testing whether trajectory replay preserves a balanced distribution across tasks without storing historical samples. Here, losses of historical mass or geometry are directly visible in the generated task proportions and in the distance to the uniform five-task target.

Data & Evaluation. We define five Gaussian tasks $T_0, \ldots, T_4$ arriving sequentially in $d = 2$, with means equally spaced on a circle of radius 3 and isotropic standard deviation parameter $\sigma := 0.35$. Each task provides 10,000/2,000/5,000 training/validation/test points, and the final target places 20% probability mass on each task (see Fig. 8a). We estimate per-task mass by labeling generated samples with their nearest mode mean and measure global recovery via energy distance to a balanced held-out mixture (details in Sec. D.2.1).

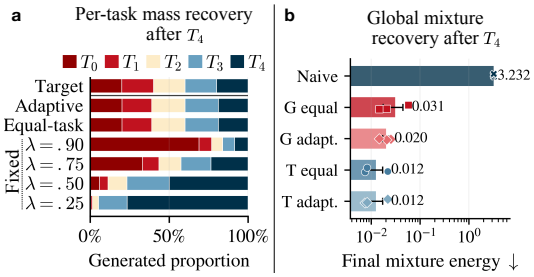

Figure 3: **(a)** Generated task mass after the final update for trajectory replay with fixed-λ, equal-task, and adaptive policies. **(b)** Energy distance to the balanced held-out mixture for naive fine-tuning and equal-task/adaptive generative (G) and trajectory (T) replay (mean ± SE, three seeds).

Baselines. We compare naive fine-tuning, which updates the model on incoming data alone, with generative and trajectory replay under equal-task ($\lambda_k := k/(k+1)$) and adaptive schedules (Sec. 3.2), the latter sweeping λ at each update and selecting by validation energy distance to the uniform mixture seen so far. Since that target is uniform, the equal-task rule *is already optimal here* and the adaptive selection should recover it. Trajectory replay is additionally ablated at constant λ.

Table 1: Final-stream performance after five tasks on Fashion-MNIST and CIFAR-10 and three tasks on ImageNet-100 (mean ± SE, three seeds). FID compares generated samples with pooled held-out data, KLD measures class-mass recovery, and MF-N uses N MeanFlow replay steps. Blue rows use trajectory replay, gray rows joint training with all data. Bold marks the best non-joint column mean.

Method	Fashion-MNIST		CIFAR-10		ImageNet-100
	↓ FID	↓ KLD	↓ FID	↓ KLD	↓ FID
Joint training (all data; upper bound)	12.5 (± 0.5)	0.040 (± 0.010)	25.9 (± 0.1)	0.100 (± 0.020)	50.7 (± 0.7)
Diffusion naive fine-tuning	67.7 (± 6.6)	4.140 (± 1.400)	53.6 (± 4.4)	1.100 (± 0.140)	105.8 (± 3.1)
Generative replay (diffusion) (Masip et al., 2025)	184.9 (± 12.6)	3.680 (± 1.320)	84.3 (± 9.1)	0.590 (± 0.110)	106.3 (± 2.5)
Generative distillation (diffusion) (Masip et al., 2025)	25.5 (± 1.1)	0.290 (± 0.040)	38.5 (± 4.4)	0.230 (± 0.080)	85.5 (± 3.4)
Flow matching naive fine-tuning	64.6 (± 11.4)	5.427 (± 2.446)	51.2 (± 4.0)	1.189 (± 0.172)	103.4 (± 2.6)
Generative replay (ODE, adaptive λ)	35.0 (± 2.4)	0.493 (± 0.131)	43.3 (± 2.3)	0.501 (± 0.046)	88.4 (± 1.3)
Generative replay (ODE, equal-task λ)	63.9 (± 3.5)	0.526 (± 0.161)	92.4 (± 15.0)	0.737 (± 0.203)	91.3 (± 4.5)
Generative replay (MF-1, adaptive λ)	52.2 (± 2.7)	0.748 (± 0.280)	55.8 (± 7.7)	0.531 (± 0.081)	151.0 (± 25.7)
Generative replay (MF-1, equal-task λ)	88.9 (± 7.1)	0.277 (± 0.030)	180.2 (± 37.4)	0.700 (± 0.254)	129.0 (± 6.4)
Generative replay (MF-4, adaptive λ)	37.8 (± 3.4)	0.504 (± 0.160)	45.2 (± 1.4)	0.508 (± 0.044)	140.6 (± 31.5)
Generative replay (MF-4, equal-task λ)	53.8 (± 4.3)	0.202 (± 0.029)	113.1 (± 14.4)	0.265 (± 0.066)	117.7 (± 4.1)
Trajectory replay (ODE, adaptive λ)	21.1 (± 3.6)	0.190 (± 0.090)	**31.4** (± 0.9)	0.272 (± 0.055)	**70.0** (± 5.8)
Trajectory replay (ODE, equal-task λ)	22.1 (± 3.2)	0.068 (± 0.039)	38.6 (± 3.5)	0.237 (± 0.066)	70.4 (± 9.2)
Trajectory replay (MF-1, adaptive λ)	20.2 (± 2.2)	0.140 (± 0.008)	33.3 (± 0.5)	0.332 (± 0.066)	70.7 (± 5.3)
Trajectory replay (MF-1, equal-task λ)	24.2 (± 0.8)	**0.065** (± 0.027)	36.8 (± 1.2)	0.237 (± 0.089)	83.0 (± 9.3)
Trajectory replay (MF-4, adaptive λ)	18.4 (± 1.5)	0.120 (± 0.023)	33.5 (± 1.7)	0.349 (± 0.056)	71.1 (± 2.7)
Trajectory replay (MF-4, equal-task λ)	**18.0** (± 1.7)	0.090 (± 0.030)	35.1 (± 3.1)	**0.211** (± 0.023)	75.3 (± 6.8)

Results. Naive fine-tuning collapses almost entirely onto T_4 (Fig. 8b), whereas equal-task and adaptive trajectory replay recover a near-uniform distribution across all five tasks (Fig. 3a). Fixing λ trades retention against fit, with high values holding earlier tasks and low values favoring T_4. Trajectory replay reaches the lowest energy distance to the balanced held-out mixture under both schedules, ahead of generative replay and with increased stability across tasks (Fig. 3b, Fig. 10). Expectedly, adaptive replay recovers the equal-task rule at every update, so the two tie (Fig. 9b).

4.3 Class-incremental image generation

Task. We evaluate class-incremental image generation, where disjoint classes arrive sequentially as CL tasks, following the same protocol as Masip et al. (2025) (see Sec. D.3.1 for full details).

Data & Evaluation. We use five-task streams on Fashion-MNIST (FMNIST) (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., 2009) with two disjoint classes per task, plus a three-task ImageNet-100 stream split $34/33/33$ (Deng et al., 2009). FMNIST and CIFAR-10 are modeled unconditionally in pixel space. To ensure quality, ImageNet-100 is modeled class-conditionally in frozen Stable-Diffusion (Rombach et al., 2022) latent space and decoded before evaluation. After the final update, we measure fit and retention jointly as Fréchet Inception Distance (FID) from $10{,}000$ generated samples against pooled held-out images from all seen classes. For FMNIST and CIFAR-10, a fixed classifier estimates generated class proportions, reported as $\mathrm{KL}(\pi_{\mathrm{data}} \| \pi_{\mathrm{gen}})$ (KLD), which scores retention alone. This is omitted for ImageNet-100, as class proportions are prescribed at sampling.

Baselines. Joint training on all tasks gives an upper bound, and naive sequential fine-tuning measures forgetting. We run fine-tuning separately for diffusion and flow matching, ruling out the generative objective as the source of any difference. We compare against diffusion-based generative replay and distillation (Masip et al., 2025), reproducing both under our protocol for ImageNet-100 since they were not evaluated there originally. For flow-based CL, generative and trajectory replay are each compared using numerical integration of the teacher flow or a MeanFlow approximation (Sec. 3.1) with one or four replay steps, under both equal-task and adaptive weighting (selecting λ by minimizing validation FID across seen tasks). In Tab. 3, we add adapter and regularization baselines.

Results. Trajectory replay is the strongest method family, achieving the best non-joint FID on all three datasets and the best KLD on both unconditional benchmarks, outperforming generative replay and diffusion-based baselines (Tab. 1). Adaptive weighting generally yields the lowest FID, whereas equal-task weighting remains competitive and consistently improves class-mass recovery. Trajectory replay also remains robust when ODE integration is replaced by one or four MeanFlow steps, while generative replay often degrades under the same approximation (Fig. 16). The runtime advantage of distillation emerges only as teacher steps grow (Fig. 17). Trajectory replay is cheaper end-to-end too, by $1.22\times$ to $1.65\times$, with the margin growing with generation cost (Fig. 12).

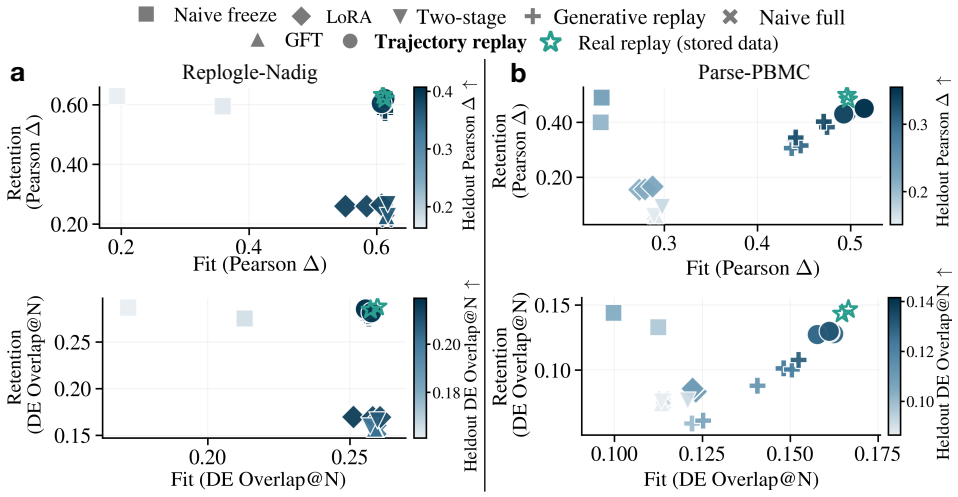

Figure 4: Final-checkpoint fit versus retention on (**a**) Replogle-Nadig and (**b**) Parse-PBMC, for Pearson Δ (top) and DE Overlap@N. Marker shape denotes the method family and each point one configuration within it, spanning the λ policies and teacher backends of Tabs. 6 and 7.

4.4 Application: Perturbation Modeling in Single-Cell Transcriptomics

Task. Data arrive as tuples $(\boldsymbol{x}, y, c)$ of a high-dimensional gene expression vector $\boldsymbol{x}$, a perturbation label y (a molecule or a gene suppression), and a context c (a cell line or a donor), and a generative model approximates the treatment response via $p(\boldsymbol{x} \mid c, y)$ (Palla et al., 2026). The initial task trains on a subset of contexts, and updates bring the same perturbations to novel ones. Success requires (**I**) preserving perturbation modeling on earlier contexts, (**II**) fitting the incoming ones, and (**III**) generalizing to unseen context-perturbation combinations from the incoming data.

Data. We use two datasets, each organized by context into a task stream (Sec. D.5.1): Replogle-Nadig (Replogle et al., 2022; Nadig et al., 2025), CRISPRi knockdowns of 2,023 essential genes, split by cell line into two tasks (T_0: Jurkat/K562/RPE1, T_1: HepG2), and Parse-PBMC (Oesinghaus et al., 2025), 90 cytokine stimulations of peripheral blood mononuclear cells, split by donor into five (T_0: eight donors, T_1–T_4: D1, D4, D9, D12). Models operate on PCA cell states fitted on T_0 (50 and 100 components, respectively), with frozen per-task affine decoders back to gene space.

Evaluation. At the final checkpoint, we report retention on previously learned strata, fit on newly introduced ones, and generalization on held-out perturbations, following the few-shot split of Adduri et al. (2026), where the held-out set covers 60% of the incoming context's perturbations, all seen in every other context, and is never used for training or λ selection. We score 500 generated cells per (*perturbation*, *context*) stratum with two metrics: macro-averaged *Pearson* Δ (correlation between predicted and observed control-subtracted effects) and *DE Overlap@*N (recovery of the N observed significant differentially expressed genes among the top-N predicted, N set per perturbation).

Baselines. We compare naive fine-tuning, full or with the velocity field or embeddings frozen in combination, gradual fine-tuning (GFT) (Thorkelsdottir & Banerjee, 2026), LoRA (Hu et al., 2022), two-stage adaptation, which first fits the new context embedding alone and then unfreezes the full model, and generative and trajectory replay. Real replay stores past tasks as a privileged reference.

Results. We jointly examine fit, retention, and held-out performance (Fig. 4; full tables in Tabs. 6 and 7). Replay is the only family reaching both axes: freezing retains without fitting, while LoRA, two-stage, and full fine-tuning either only fit or neither. Trajectory replay edges ahead of generative replay on the single-update Replogle-Nadig stream and clearly surpasses it across the five sequential tasks of Parse-PBMC, approaching real replay without storing cells. Point color confirms that held-out performance follows the upper-right frontier rather than either axis alone, since an unseen combination needs both the perturbation response carried from old contexts and the basal structure of the new one. As in the image streams, replacing ODE integration with one or four MeanFlow steps leaves trajectory replay at parity while generative replay degrades (Fig. 24).

4.5 Application: Continual phenotypic modeling in microscopy

Task. We extend Sec. 4.4 to microscopy, where $\boldsymbol{x}$ is an image, and the model captures the induced morphological phenotype. RxRx1 (Sypetkowski et al., 2023) images U2OS cells under gene knockdowns across experimental batches acquired weeks apart, and each batch is one task. The perturbation vocabulary is identical throughout, so the shift is technical.

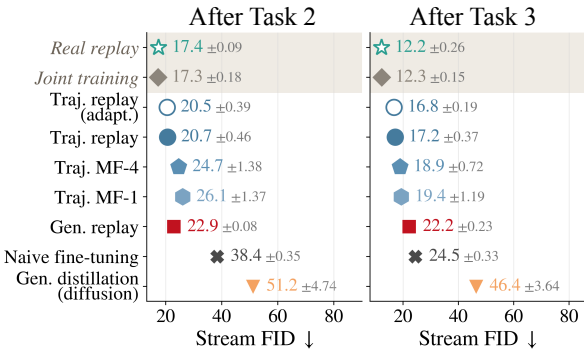

Figure 5: FID after the second and third tasks, computed on features pooled over all seen experiments (mean ± SE, three seeds). Gray rows are upper bounds.

Setup. We stream three U2OS experiments (01 → 02 → 03) over 1,071 perturbations, conditioning on perturbation and experiment identity, with six-channel 96×96 image crops. Baselines follow Sec. 4.3, adding real replay as a privileged reference. We generate one image per test condition and report FID against pooled test images of all seen experiments after each update (Zhang et al., 2025) (see Sec. D.6.1 and Tab. 9 for full results).

Results (Fig. 5). Diffusion distillation is the weakest replay method at both checkpoints. Trajectory replay with ODE integration improves over generative replay at each update, almost entirely on the oldest batch (see Fig. 27), and stays within a few points of real replay and joint training without storing images, qualitatively too (Fig. 29). Its MeanFlow variants rank just behind it on the final stream.

5 Related works

Lifelong learning for generative models. CL methods for generative models span architectural updates (Smith et al., 2024), regularization (Dong et al., 2024; Kirkpatrick et al., 2017), and replay (Sun et al., 2024; Wang et al., 2024; Guo et al., 2025), the last used to curb forgetting in sequential classification (Shin et al., 2017; Wu et al., 2018; Ye & Bors, 2020) and in continuous-time frameworks (Gao & Liu, 2023; Skierś & Deja, 2025; Liao et al., 2025). Similar to our setting, generation is treated not only as a replay source but as the task to preserve itself (Simone et al., 2025; Zając et al., 2023; Liu et al., 2026). Closest to our work is Masip et al. (2025), who use a diffusion model over historical data as a teacher for a sequential noise predictor over noise-augmented replayed samples.

Flow matching and density mixtures. Trajectory replay draws on flow matching (Lipman et al., 2023; Tong et al., 2024a) and distillation (Geng et al., 2026a; Boffi et al., 2025), and on continuous-time likelihood control (Albergo et al., 2025; Su et al., 2025; Römer et al., 2026) and mixture sampling (Liu, 2022), both mostly explored as inference-time control over frozen models (Skreta et al., 2025). Closest to our work is Thorkelsdottir & Banerjee (2026), who anneal the historical model toward the incoming task, whereas trajectory replay preserves old tasks while integrating new ones.

6 Discussion

Conclusion. We introduced trajectory replay, which replaces sample-level generative replay with velocity-field regression along the historical model's own trajectories. Its optimum is the density-weighted average of the two task velocities, generating the mixture of their probability paths, so the stability-plasticity coefficient sets what the update retains rather than only trading it off. A matching rule for the mean velocity carries this optimum to the flow-map level. Empirically, trajectory replay outperforms replay and distillation baselines on class-incremental image generation, stays robust under few-step replay, and extends perturbation models to incremental experimental measurements.

Limitations. Trajectory replay still involves partial simulation under a historical model, and its adaptive version, though favorable on image generation, requires expensive update-level sweeps. Mean velocity aids sampling but adds a costly gradient computation. Furthermore, estimation error tends to compound over long sequences, echoing forgetting bounds for continually fine-tuned Large Language Models (LLMs), and our reliance on flow matching leaves extension to autoregressive models open. Finally, future work will explore CL updates of more efficient distillation models.

AI USE STATEMENT

This work was carried out with the support of LLMs at specific stages of paper writing and development. We disclose these uses below, organized by task.

Generate synthetic datasets. Coding agents were used to accelerate the generation of the synthetic Gaussian datasets in Sec. 4.1 and Sec. 4.2. The design and ideation of these simulations were not carried out by AI agents.

Develop theoretical models, prove mathematical claims, and write proofs. All seed theoretical formulations, proofs, and conceptualization were formulated manually. LLMs were used afterward to validate correctness, enrich the set of assumptions, improve rigor, and refine language, with prompts designed to preserve the original formulation as closely as possible. AI agents specifically assisted in formulating Prop. 3.3 by providing theoretical background and connections on mean velocities and semigradients.

Propose hypotheses, design methodology, and interpret results. Ideation, hypothesis generation, experimental design, and result interpretation were not carried out by AI. AI agents were used only to help contextualize results in light of our theoretical framework and to provide feedback on formalization before experiments were run. All reasoning was human-guided.

Support qualitative and thematic data analysis. Not applicable, this work does not involve qualitative or thematic data analysis.

Implement methods, create or edit software code, assist with translation, create or modify scientific figures or images. Coding agents were used to support the implementation of experiments and to enhance plotting. AI models were not used for translation.

Draft parts of a research paper, edit a research paper to improve readability, propose a title. AI was used to look for language and typing errors, and to shorten sections of the paper while maintaining fixed tone and content. AI was used to brainstorm shorter versions of an original title. AI was not used to draft entire sections of the paper body. It was, however, used to draft the initial skeleton of a series of methods description sections in the Appendix. All these sections have been partially or fully rewritten and refined manually.

AI was not used for the following tasks:

- Formulate questions for surveys or interviews.
- Suggest experimental parameters.
- Transcribe recordings of research material.
- Discover research topics or identify gaps.
- Summarize or analyse existing literature.
- Clean and reformat datasets.
- Suggest a structure for a research paper.
- Format references.

ETHICS STATEMENT

Our biological case studies use publicly available, de-identified datasets, and do not involve novel human-subject experiments. As trajectory replay is designed to support continual updates of biological and treatment-response models as new experimental data arrives, downstream applications could in the future extend to more sensitive settings involving clinical or patient-derived data, which would call for the data-governance and privacy safeguards appropriate to that context.

REPRODUCIBILITY STATEMENT

Details to reproduce our work are fully disclosed in the Appendix. All theoretical claims are stated with their assumptions in Sec. A, and full proofs are given in Sec. A.6, Sec. A.8, Sec. A.9, and

Sec. A.10, each restating the result before proving it. The two continual update procedures are written out as algorithmic boxes in Sec. C, including the sampling of the historical branch, the loss weighting, and the stop-gradient structure of the mean velocity objective.

Every experiment is described in Sec. D, one subsection per setting, covering the data and preprocessing, the stream construction for continual updates and splits, the backbone and optimizer settings, the replay configuration, the baselines, and the evaluation protocol. Exact hyperparameter values for the image benchmark, single-cell transcriptomics example, and microscopy application are collected in per-experiment tables (Tab. 2, Tab. 4, Tab. 8). Hyperparameters for the two synthetic settings are described in Sec. D.1 and Sec. D.2. Moreover, we specify the generating distributions used for synthetic experiments in full, so they can be reproduced without any external data. The class-incremental image streams use open datasets (Sec. D.3) Fashion-MNIST, CIFAR-10, and ImageNet-100 with class orders drawn through Avalanche and fixed per reported seed, and reuse the backbone and protocol of Masip et al. (2025) so that our numbers are directly comparable to theirs. The two biological applications use public datasets with published splits: Replogle-Nadig and the PBMC cytokine screen under the few-shot definitions of Adduri et al. (2026) (Sec. D.5), and RxRx1 under the U2OS protocol of Zhang et al. (2025) (Sec. D.6). Architecture, latent-dimension, and training-budget choices are reported together with the selection studies that produced them (Fig. 11, Tab. 5), all run on a single seed before any continual-learning run. Across every experiment, model and λ selection uses validation data only, test splits are reserved for reporting, and each configuration is run for at least three seeds with results reported as means with standard errors unless noted.

Hardware constraints are reported in Sec. B.

REFERENCES

Abhinav K. Adduri, Dhruv Gautam, Beatrice Bevilacqua, Mohsen Naghipourfar, Alishba Imran, Rohan Shah, Noam Teyssier, Rishi Verma, Christopher Carpenter, Basak Eraslan, Francis Chalissery, Rajesh Ilango, Vishvak Subramanyam, Chiara Ricci-Tam, Sanjay Nagaraj, Aidan Winters, Mingze Dong, Stefanie Fellinger, Adam Krejci, Tilmann Burckstummer, Sravya Tirukkovular, Jeremy Sullivan, Brian S. Plosky, Nicholas D. Youngblut, Jure Leskovec, Luke A. Gilbert, Silvana Konermann, Patrick D. Hsu, Alexander Dobin, Dave P. Burke, Hani Goodarzi, and Yusuf H. Roohani. Predicting cellular responses to perturbation across diverse contexts with State. *Cell*, 189(19):5914–5931.e20, 2026. ISSN 0092-8674. doi: 10.1016/j.cell.2026.07.052.

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A PyTorch Library for Deep Continual Learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

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

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Sander Dieleman. Learning the integral of a diffusion model, 2026. URL https://sander.ai/2026/05/06/flow-maps.html.

Jiahua Dong, Wenqi Liang, Hongliu Li, Duzhen Zhang, Meng Cao, Henghui Ding, Salman Khan, and Fahad Khan. How to continually adapt text-to-image diffusion models for flexible customization? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Rui Gao and Weiwei Liu. DDGR: Continual learning with deep diffusion-based generative replay. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10744–10763. PMLR, 23–29 Jul 2023.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026b.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Da-Han Wang, et al. Continual learning for generative AI: From LLMs to MLLMs and beyond. *arXiv preprint arXiv:2506.13045*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Charles Jones, Emmanuel Noutahi, Jason Hartford, and Cian Eastwood. Elucidating the design space of flow matching for cellular microscopy. *arXiv preprint arXiv:2603.26790*, 2026.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24174–24184. IEEE, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Dominik Klein, Jonas Simon Fleck, Daniil Bobrovskiy, Lea Zimmermann, Sören Becker, Alessandro Palma, Leander Dony, Alejandro Tejada-Lapuerta, Guillaume Huguet, Hsiu-Chuan Lin, Nadezhda Azbukina, Fátima Sanchís-Calleja, Theo Uscidda, Artur Szalata, Manuel Gander, Aviv Regev, Barbara Treutlein, J. Gray Camp, and Fabian J. Theis. Cellflow enables generative single-cell phenotype modeling with flow matching. *bioRxiv*, 2025. doi: 10.1101/2025.04.11.648220.

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

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Yu-Chien Liao, Jr-Jen Chen, Chi-Pin Huang, Ci-Siang Lin, Meng-Lin Wu, and Yu-Chiang Frank Wang. Continual personalization for diffusion models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15511–15520. IEEE, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Jingren Liu, Zhong Ji, Shuning Xu, Yun Wang, and Xiangyu Chen. CCD: Continual Consistency Diffusion for Lifelong Generative Modeling. *IEEE Transactions on Image Processing*, 2026.

Qiang Liu. Rectified Flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost Van de Weijer. Generative feature replay for class-incremental learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 915–924. IEEE, 2020.

Sergi Masip, Pau Rodriguez, Tinne Tuytelaars, and Gido M van de Ven. Continual learning of diffusion models with generative distillation. In Vincenzo Lomonaco, Stefano Melacci, Tinne Tuytelaars, Sarath Chandar, and Razvan Pascanu (eds.), *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 431–456. PMLR, 29 Jul–01 Aug 2025.

Alex Morehead, Lazar Atanackovic, Akshata Hegde, Yanli Wang, Frimpong Boadu, Joel Selvaraj, Alexander Tong, Aditi Krishnapriyan, and Jianlin Cheng. Flow matching for generative modelling in bioinformatics and computational biology. *Nature Machine Intelligence*, pp. 1–18, 2026.

Ajay Nadig, Joseph M. Replogle, Angela N. Pogson, Mukundh Murthy, Steven A. McCarroll, Jonathan S. Weissman, Elise B. Robinson, and Luke J. O'Connor. Transcriptome-wide analysis of differential expression in perturbation atlases. *Nature Genetics*, 57(5):1228–1237, April 2025. ISSN 1546-1718. doi: 10.1038/s41588-025-02169-3.

Lukas Oesinghaus, Sören Becker, Larsen Vornholz, Efthymia Papalexi, Joey Pangallo, Amir Ali Moinfar, Jenni Liu, Alyssa La Fleur, Maiia Shulman, Simone Marrujo, Bryan Hariadi, Crina Curca, Alexa Suyama, Maria Nigos, Oliver Sanderson, Hoai Nguyen, Vuong K. Tran, Ajay A. Sapre, Olivia Kaplan, Sarah Schroeder, Alec Salvino, Guillermo Gallareta-Olivares, Ryan Koehler, Gary Geiss, Alexander B. Rosenberg, Charles M. Roco, Georg Seelig, and Fabian J. Theis. A single-cell cytokine dictionary of human peripheral blood. *bioRxiv*, 2025. doi: 10.64898/2025.12.12.693897.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. Featured Certification.

Giovanni Palla, Sudarshan Babu, Payam Dibaeinia, James D Pearce, Donghui Li, Aly A Khan, Theofanis Karaletsos, and Jakub M. Tomczak. Scalable single-cell gene expression generation with latent diffusion models. In *Forty-third International Conference on Machine Learning*, 2026.

Joseph M. Replogle, Reuben A. Saunders, Angela N. Pogson, Jeffrey A. Hussmann, Alexander Lenail, Alina Guna, Lauren Mascibroda, Eric J. Wagner, Karen Adelman, Gila Lithwick-Yanai, Nika Iremadze, Florian Oberstrass, Doron Lipson, Jessica L. Bonnar, Marco Jost, Thomas M. Norman, and Jonathan S. Weissman. Mapping information-rich genotype-phenotype landscapes with genome-scale Perturb-seq. *Cell*, 185(14):2559–2575.e28, 2022. ISSN 0092-8674. doi: 10.1016/j.cell.2022.05.013.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, Los Alamitos, CA, USA, June 2022. IEEE Computer Society.

Ralf Römer, Maximilian Seeliger, Saida Liu, Ben Sturgis, Marco Bagatella, Daniel Marta, Andreas Krause, and Angela P Schoellig. Uncertainty quantification for flow-based vision-language-action models. *arXiv preprint arXiv:2606.18043*, 2026.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Lorenzo Simone, Davide Bacciu, and Shuangge Ma. Continualflow: Learning and unlearning with neural flow matching. In *ICML 2025 Workshop on Machine Unlearning for Generative AI*, 2025.

Paweł Skierś and Kamil Deja. Joint diffusion models in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4380–4390, October 2025.

Marta Skreta, Lazar Atanackovic, Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposition of diffusion models using the itô density estimator. In *The Thirteenth International Conference on Learning Representations*, 2025.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-loRA. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Nan Song, Xiaofeng Yang, Ze Yang, and Guosheng Lin. Towards lifelong few-shot customization of text-to-image diffusion. *CoRR*, abs/2411.05544, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Anuroop Sriram, Benjamin Kurt Miller, Ricky T. Q. Chen, and Brandon M Wood. FlowLLM: Flow matching for material generation with large language models as base distributions. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Maojiang Su, Jerry Yao-Chieh Hu, Sophia Pi, and Han Liu. On flow matching KL divergence. *arXiv preprint arXiv:2511.05480*, 2025.

Endre Süli and David F Mayers. *An introduction to numerical analysis*, volume 1. Cambridge University Press, 2003.

Gan Sun, Wenqi Liang, Jiahua Dong, Jun Li, Zhengming Ding, and Yang Cong. Create your world: Lifelong text-to-image diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(9):6454–6470, 2024. doi: 10.1109/TPAMI.2024.3382753.

Maciej Sypetkowski, Morteza Rezanejad, Saber Saberian, Oren Kraus, John Urbanik, James Taylor, Ben Mabey, Mason Victors, Jason Yosinski, Alborz Rezazadeh Sereshkeh, Imran Haque, and Berton Earnshaw. RxRx1: A dataset for evaluating experimental batch correction methods. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4285–4294, June 2023.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Gudrun Thorkelsdottir and Arindam Banerjee. Gradual fine-tuning for flow matching models. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2026.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. Expert Certification.

Alexander Y. Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free Schrödinger bridges via score and flow matching. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1279–1287. PMLR, 02–04 May 2024b.

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

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay GANs: Learning to generate new categories without forgetting. *Advances in neural information processing systems*, 31, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Fei Ye and Adrian G Bors. Learning latent representations across multiple data domains using lifelong VAEGAN. In *European Conference on Computer Vision*, pp. 777–795. Springer, 2020.

Michał Zając, Kamil Deja, Anna Kuzina, Jakub M Tomczak, Tomasz Trzciński, Florian Shkurti, and Piotr Miłoś. Exploring continual learning of diffusion models. *arXiv preprint arXiv:2303.15342*, 2023.

Yuhui Zhang, Yuchang Su, Chenyu Wang, Tianhong Li, Zoe Wefers, Jeffrey J Nirschl, James Burgess, Daisy Ding, Alejandro Lozano, Emma Lundberg, and Serena Yeung-Levy. CellFlux: Simulating cellular morphology changes via flow matching. In *Forty-second International Conference on Machine Learning*, 2025.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

A THEORETICAL INTEGRATION

A.1 THEORETICAL INTEGRATION ON FLOW MATCHING

Flows and the continuity equation. Flow matching (Lipman et al., 2023; Tong et al., 2024a) regresses a marginal vector field $\{b_t(\boldsymbol{x})\}_{t\in[0,1]}$ that generates a probability path $\{p_t(\boldsymbol{x})\}_{t\in[0,1]}$ through the continuity equation:

$$\frac{\partial}{\partial t}p_t(\boldsymbol{x}) = -\nabla\cdot\left(p_t(\boldsymbol{x})b_t(\boldsymbol{x})\right). \tag{13}$$

For sufficiently regular b_t, the solution map of Eq. (3) that sends an initial state to its state at time t is a diffeomorphism, called the *flow*, that maps noise vectors to samples from p_t. Let $\phi_t : \mathcal{X} \to \mathcal{X}$, for $t \in [0, 1]$, denote this map, with $\phi_0 := \mathrm{id}$ and $\phi_t(\boldsymbol{x}_0) := \boldsymbol{x}_t$. The mass conservation expressed by Eq. (13) then makes the flow push the prior forward along the path,

$$p_t = (\phi_t)_\sharp\, p_0\,, \tag{14}$$

that is, $\boldsymbol{x}_0 \sim p_0$ implies $\boldsymbol{x}_t \sim p_t$ for all $t \in [0, 1]$.

Gaussian conditional flows. It is common practice to use conditional Gaussian paths to evaluate the conditional flow matching objective in Eq. (2). Recall the definition of Gaussian paths:

$$p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1) := \mathcal{N}(\boldsymbol{x}; \alpha_t\boldsymbol{x}_1, \sigma_t^2\mathbb{I}_d)\,, \tag{15}$$

with $\boldsymbol{x}_1 \sim q$ and, for the linear schedule, $\alpha_t := t$ and $\sigma_t := 1 - t$, though other variants exist (Albergo et al., 2025; Tong et al., 2024b). Writing $\boldsymbol{x} = \alpha_t\boldsymbol{x}_1 + \sigma_t\boldsymbol{x}_0$ with $\boldsymbol{x}_0 \sim p_0$, differentiating in t gives the associated conditional velocity

$$b_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1) := \dot\alpha_t\boldsymbol{x}_1 + \dot\sigma_t\,\frac{\boldsymbol{x} - \alpha_t\boldsymbol{x}_1}{\sigma_t}\,, \tag{16}$$

which reduces to $\boldsymbol{x}_1 - \boldsymbol{x}_0$ under the linear schedule (Tong et al., 2024a). Conditional flow matching hinges on the fact that, in expectation, regressing $b_{t|1}$ along $p_{t|1}$ recovers the marginal path

$$p_t(\boldsymbol{x}) = \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, q(\boldsymbol{x}_1)\,\mathrm{d}\boldsymbol{x}_1\,, \tag{17}$$

whose generating velocity field is the posterior expectation of the conditional one over clean data given the noisy state (Lipman et al., 2023),

$$b_t(\boldsymbol{x}) = \int b_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})\,\mathrm{d}\boldsymbol{x}_1\,. \tag{18}$$

Relationship with the density score. A crucial property of flow matching with Gaussian conditional paths is that the marginal vector field is affine in the marginal probability path's score, with coefficients dependent on the scheduling parameters (Zheng et al., 2023):

$$b_t(\boldsymbol{x}) = a_t\,\boldsymbol{x} + c_t\,\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\,, \tag{19}$$

with $a_t := \dot\alpha_t/\alpha_t$ and $c_t := \frac{\sigma_t}{\alpha_t}\big(\dot\alpha_t\sigma_t - \alpha_t\dot\sigma_t\big)$. Wherever $\alpha_t, \sigma_t > 0$ and $\dot\alpha_t\sigma_t - \alpha_t\dot\sigma_t \neq 0$, so that $c_t \neq 0$, the relation is invertible, and learning the marginal velocity field and learning the marginal score are equivalent tasks. We use this equivalence in the proof of Thm. 3.2, whose assumptions include that condition on $(0, 1)$.

A.2 MEAN FLOW BASICS

From instantaneous to average velocity. Recall the definition of mean velocity in Sec. 2.1:

$$V_{s,t}(\boldsymbol{x}_s) := \frac{1}{t-s}\int_s^t b_\tau(\boldsymbol{x}_\tau)\mathrm{d}\tau\,, \tag{20}$$

with $\boldsymbol{x}_s \sim p_s$ and $0 \le s < t \le 1$. To distinguish our notation from Geng et al. (2026a), we set noise at $t = 0$ and data at $t = 1$, so s is the lower limit of integration. Multiplying Eq. (20) by $(t - s)$ and

differentiating both sides with respect to s, with t held fixed and along the trajectory through $\boldsymbol{x}_s$, yields the MeanFlow identity:

$$\frac{\mathrm{d}}{\mathrm{d}s}\Big[(t-s)\,V_{s,t}(\boldsymbol{x}_s)\Big] = \frac{\mathrm{d}}{\mathrm{d}s}\int_s^t b_\tau(\boldsymbol{x}_\tau)\,\mathrm{d}\tau$$

$$-V_{s,t}(\boldsymbol{x}_s) + (t-s)\frac{\mathrm{d}}{\mathrm{d}s}V_{s,t}(\boldsymbol{x}_s) = -b_s(\boldsymbol{x}_s)$$

$$V_{s,t}(\boldsymbol{x}_s) = b_s(\boldsymbol{x}_s) + (t-s)\frac{\mathrm{d}}{\mathrm{d}s}V_{s,t}(\boldsymbol{x}_s)\,, \tag{21}$$

where the left-hand side follows from the product rule and the right-hand side from the fundamental theorem of calculus. Note that the sign in Eq. (21) is flipped compared to Geng et al. (2026a), as we differentiate with respect to the earlier time, which is the lower limit of integration. Since $\boldsymbol{x}_s$ varies with s along the trajectory, the derivative is total and expands as

$$\frac{\mathrm{d}}{\mathrm{d}s}V_{s,t}(\boldsymbol{x}_s) = \partial_s V_{s,t}(\boldsymbol{x}_s) + \nabla_{\boldsymbol{x}_s} V_{s,t}(\boldsymbol{x}_s)\, b_s(\boldsymbol{x}_s)\,. \tag{22}$$

Boundary behavior and flow maps. As stated, Eq. (20) is undefined on the diagonal, but for continuous b the average converges to the instantaneous velocity as the interval closes (Geng et al., 2026a), so we extend V to $s = t$ by

$$V_{t,t}(\boldsymbol{x}_t) := \lim_{s\uparrow t} V_{s,t}(\boldsymbol{x}_s) = b_t(\boldsymbol{x}_t)\,. \tag{23}$$

A parameterized field V^θ is defined on the whole square and does not satisfy this automatically, so it is usually enforced with heuristics such as oversampling the time diagonal during distillation.

As mentioned in Sec. 2.1, the mean velocity $V_{s,t}$ defines a flow map:

$$\boldsymbol{X}_{s,t}(\boldsymbol{x}_s) := \boldsymbol{x}_s + (t-s)V_{s,t}(\boldsymbol{x}_s)\,, \tag{24}$$

which satisfies $\boldsymbol{X}_{t,t} = \mathrm{id}$ by construction, since the factor $(t-s)$ vanishes on the diagonal. Under this boundary condition, and for sufficiently regular b, the flow map is characterized equivalently by the Lagrangian condition, the Eulerian condition, or the semigroup property (Boffi et al., 2025), so any one of them can serve as a training target. We use the semigroup property, $\boldsymbol{X}_{s,t} = \boldsymbol{X}_{r,t} \circ \boldsymbol{X}_{s,r}$ for any $s < r < t$, which is what licenses few-step sampling in Sec. 3.2, and the Eulerian condition, which recovers the MeanFlow identity as follows.

Following Dieleman (2026), the Eulerian condition states that the flow map is invariant along the trajectory through its starting point,

$$\frac{\partial}{\partial s}\boldsymbol{X}_{s,t}(\boldsymbol{x}_s) + \nabla_{\boldsymbol{x}_s}\boldsymbol{X}_{s,t}(\boldsymbol{x}_s)\, b_s(\boldsymbol{x}_s) = \boldsymbol{0}\,. \tag{25}$$

Substituting Eq. (24) into Eq. (25) gives

$$\begin{aligned}\boldsymbol{0} &= -V_{s,t}(\boldsymbol{x}_s) + (t-s)\,\partial_s V_{s,t}(\boldsymbol{x}_s) + \Big(\mathbb{I}_d + (t-s)\nabla_{\boldsymbol{x}_s}V_{s,t}(\boldsymbol{x}_s)\Big)b_s(\boldsymbol{x}_s)\\ &= -V_{s,t}(\boldsymbol{x}_s) + b_s(\boldsymbol{x}_s) + (t-s)\Big(\partial_s V_{s,t}(\boldsymbol{x}_s) + \nabla_{\boldsymbol{x}_s}V_{s,t}(\boldsymbol{x}_s)\, b_s(\boldsymbol{x}_s)\Big)\,,\end{aligned} \tag{26}$$

which rearranges to Eq. (21). The two formulations are therefore equivalent: the MeanFlow identity is the Eulerian condition written in terms of the average velocity rather than the map itself.

A.3 Stop-gradient training and marginal-from-conditional learning

Semigradients. Recall the residual formula used to learn the mean velocity in Eq. (4). The gradient flows through the parameters θ of the neural mean velocity approximation only through the first term, treating the MeanFlow identity as a constant, and avoiding higher-order derivatives. Following Dieleman (2026), writing $\mathcal{R}$ for the residual and $\mathcal{L}(\theta) := \mathbb{E}[\|\mathcal{R}\|_2^2]$, the direction followed is

$$\widetilde{G}_\theta = 2\,\mathbb{E}\Big[\big(\partial_\theta V^\theta_{s,t}(\boldsymbol{x}_s)\big)^\top \mathcal{R}_{s,t}(V^\theta, b)(\boldsymbol{x}_s)\Big]\,, \tag{27}$$

rather than $\nabla_\theta\mathcal{L} = 2\,\mathbb{E}[(\partial_\theta\mathcal{R})^\top\mathcal{R}]$, since $V^\theta_{s,t}(\boldsymbol{x}_s)$ is the only θ-dependent term outside the stop-gradient. We refer to $\widetilde{G}_\theta$ as a *semigradient*: it is not the gradient of any objective, so the field it converges to is characterized as a *fixed point*, at which $\widetilde{G}_\theta$ vanishes, rather than as a minimizer. This is the sense in which Prop. 3.3 is stated.

Marginal from conditional. Write the target of Eq. (4) as $T_{s,t}(V, \boldsymbol{u})(\boldsymbol{x}) := \boldsymbol{c}(\boldsymbol{x}) + M(\boldsymbol{x})\,\boldsymbol{u}$, with $\boldsymbol{c} := (t-s)\,\partial_s V_{s,t}$ and $M := \mathbb{I}_d + (t-s)\nabla_{\boldsymbol{x}} V_{s,t}$. The residual, and hence the expression inside the expectation in Eq. (27), is therefore affine in the velocity slot $\boldsymbol{u}$. Conditioning on $\boldsymbol{x}_s$ and applying the tower property to $b_s(\boldsymbol{x}_s) = \mathbb{E}[b_{s|1}(\boldsymbol{x}_s \mid \boldsymbol{x}_1) \mid \boldsymbol{x}_s]$ then leaves $\widetilde{G}_\theta$ unchanged under substitution of the conditional velocity for the marginal one, so the mean velocity can be trained simulation-free from data (Geng et al., 2026a; Dieleman, 2026). This requires the stop-gradient, without which the expression is quadratic in $\boldsymbol{u}$ and the conditional variance of $b_{s|1}$ given $\boldsymbol{x}_s$ would shift the fixed point rather than only adding variance, and the evaluation at $\boldsymbol{x}_s$ of Eq. (21).

A.4 Replay as mixture modeling

Generative replay can be seen as modeling a mixture between historical and new data densities, where the mixture coefficient λ represents the rate at which samples are drawn from a generative model over previous tasks.

Given a flow model over historical data $\{p_t^\theta\}_{t\in[0,1]}$ and an incoming task distribution q', generative replay samples batches from the following distribution:

$$q_{\text{batch}} := \lambda p_1^\theta + (1-\lambda) q' \,. \tag{28}$$

Note that p_1^θ approximates the historical density q, hence q_{batch} is an approximation of Eq. (5). The resulting mixed batches are then used to train a generative model with the conditional flow matching loss in Eq. (2). There, the data density enters only as the law of $\boldsymbol{x}_1$, so training on replay-augmented batches amounts to minimizing the same objective with q replaced by q_{batch}. By the marginal-from-conditional construction in Eq. (17), its minimizer is the marginal velocity generating

$$\rho_t(\boldsymbol{x}) := \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, q_{\text{batch}}(\boldsymbol{x}_1)\, \mathrm{d}\boldsymbol{x}_1 \,, \tag{29}$$

whose terminal density is $\rho_1 = q_{\text{batch}}$, since the schedule satisfies $\alpha_1 = 1$ and $\sigma_1 = 0$, so that $p_{1|1}(\cdot \mid \boldsymbol{x}_1)$ is a point mass at $\boldsymbol{x}_1$.

Hence, up to an approximation error of q by p_1^θ, generative replay optimizes a flow matching model over Eq. (5).

A.5 Static KL as mixture minimizer

We restate the formal optimization objective over nonnegative integrable functions subject to normalization:

$$\begin{aligned} \tilde{q} = \arg\min_{\rho \geq 0} \left[\lambda D_{\mathrm{KL}}(q \parallel \rho) + (1-\lambda) D_{\mathrm{KL}}(q' \parallel \rho)\right] \,, \\ \text{s.t.} \quad \int \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1 \,. \end{aligned} \tag{30}$$

Expanding the Kullback-Leibler divergence definitions yields the following objective functional:

$$\begin{aligned} F(\rho) &= \lambda \int q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{\rho(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + (1-\lambda) \int q'(\boldsymbol{x}) \log \frac{q'(\boldsymbol{x})}{\rho(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\ &= \lambda \int q(\boldsymbol{x}) \log q(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \lambda \int q(\boldsymbol{x}) \log \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &\quad + (1-\lambda) \int q'(\boldsymbol{x}) \log q'(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - (1-\lambda) \int q'(\boldsymbol{x}) \log \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \,. \end{aligned} \tag{31}$$

Incorporating the density normalization constraint via a Lagrange multiplier $\gamma \in \mathbb{R}$, we construct the Lagrangian:

$$\Lambda(\rho, \gamma) = F(\rho) + \gamma \left[\int \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - 1\right] \,. \tag{32}$$

To find the optimal density, we compute the functional derivative of Eq. (32) with respect to $\rho(\boldsymbol{x})$ and set it to zero:

$$\frac{\delta \Lambda(\rho, \gamma)}{\delta \rho(\boldsymbol{x})} = -\lambda \frac{q(\boldsymbol{x})}{\rho(\boldsymbol{x})} - (1-\lambda) \frac{q'(\boldsymbol{x})}{\rho(\boldsymbol{x})} + \gamma = 0 \,. \tag{33}$$

Solving for $\rho(\boldsymbol{x})$ yields:

$$\rho(\boldsymbol{x}) = \frac{\lambda q(\boldsymbol{x}) + (1-\lambda) q'(\boldsymbol{x})}{\gamma} \,. \tag{34}$$

Integrating both sides over the domain ensures that the normalization constraint is satisfied, which requires:

$$\begin{aligned} \gamma &= \int \left[\lambda q(\boldsymbol{x}) + (1-\lambda) q'(\boldsymbol{x})\right] \mathrm{d}\boldsymbol{x} \\ &= \lambda \int q(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + (1-\lambda) \int q'(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &= 1 \,, \end{aligned} \tag{35}$$

since $q(\boldsymbol{x})$ and $q'(\boldsymbol{x})$ are valid probability density functions that integrate to unity. Substituting $\gamma = 1$ back into Eq. (34) gives the mixture $\tilde{q}(\boldsymbol{x}) = \lambda q(\boldsymbol{x}) + (1-\lambda) q'(\boldsymbol{x})$, which is nonnegative and integrates to unity, so it is feasible.

This critical point is the unique global minimizer: discarding terms independent of the optimized ρ, $F(\rho) = -\int \tilde{q}(\boldsymbol{x}) \log \rho(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \text{const} = D_{\mathrm{KL}}(\tilde{q} \parallel \rho) + \text{const}'$, which is nonnegative and vanishes only at $\rho = \tilde{q}$ almost everywhere, completing the proof.

A.6 Proof of Prop. 3.1

Proposition 3.1 (restated)**.** *Let $p_t, p'_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be density paths generated through the continuity equation in Sec. 2.1 by velocity fields $b_t, b'_t \in C^0([0,1], (C^1(\mathbb{R}^d))^d)$, with shared prior $p_0 = p'_0 := \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ and terminal densities $p_1 := q$, $p'_1 := q'$. Fix $\lambda \in (0,1)$ and assume $\lambda p_t + (1-\lambda) p'_t > 0$ on $[0,1] \times \mathcal{X}$. Then $\mathcal{J}(\cdot\,; b, b')$ in Eq. (7) admits*

$$v^\star_t(\boldsymbol{x}) := \frac{\lambda \, p_t(\boldsymbol{x}) \, b_t(\boldsymbol{x}) + (1-\lambda) \, p'_t(\boldsymbol{x}) \, b'_t(\boldsymbol{x})}{\lambda \, p_t(\boldsymbol{x}) + (1-\lambda) \, p'_t(\boldsymbol{x})} \,,$$

as unique minimizer, which generates the mixture path $\rho^\star_t := \lambda p_t + (1-\lambda) p'_t$, satisfying $\partial_t \rho^\star_t + \nabla_{\boldsymbol{x}} \cdot (v^\star_t \rho^\star_t) = 0$ with $\rho^\star_0 = \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ and $\rho^\star_1 = \tilde{q}$ (Eq. (5)).

The conditional Gaussian structure required by the standing assumptions of Sec. 3.1 is not used here: the minimizer and the continuity argument need only positivity and the stated regularity, so the density-weighted mixture holds for arbitrary probability paths. A shared family becomes necessary only in Thm. 3.2, where the score-to-velocity inversion relies on it.

Proof. **Minimizer.** As a first step, we rewrite the objective in Eq. (7) by expanding the expectation and grouping the integrals.

$$\mathcal{J}(v; b, b') = \int_0^1 \int \Big(\lambda \, \|v_t(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2 \, p_t(\boldsymbol{x}) + (1-\lambda) \, \|v_t(\boldsymbol{x}) - b'_t(\boldsymbol{x})\|_2^2 \, p'_t(\boldsymbol{x}) \Big) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \,. \tag{36}$$

By the calculus of variations, the optimization task in Eq. (7) becomes a pointwise minimization, as the objective does not contain any derivative of the minimizer v. It therefore suffices to minimize the integrand of Eq. (36) over $\boldsymbol{u} \in \mathbb{R}^d$ at each fixed $(t, \boldsymbol{x})$. Setting its gradient to zero,

$$2\lambda \, p_t(\boldsymbol{x}) \big(\boldsymbol{u} - b_t(\boldsymbol{x})\big) + 2(1-\lambda) \, p'_t(\boldsymbol{x}) \big(\boldsymbol{u} - b'_t(\boldsymbol{x})\big) = \mathbf{0} \,, \tag{37}$$

and collecting the terms in $\boldsymbol{u}$ gives $\big(\lambda p_t + (1-\lambda) p'_t\big)\boldsymbol{u} = \lambda p_t b_t + (1-\lambda) p'_t b'_t$, where all quantities are evaluated at $(t, \boldsymbol{x})$. The bracketed factor is strictly positive by assumption, so dividing through recovers Eq. (8). The integrand is a positively weighted sum of quadratics, with Hessian $2(\lambda p_t + (1-\lambda) p'_t)\mathbb{I}_d \succ 0$, hence strictly convex in $\boldsymbol{u}$, so this critical point is its unique global minimizer. Since the pointwise minimizer is unique at every $(t, \boldsymbol{x})$, any other field attaining the same value of $\mathcal{J}$ must agree with $v^\star$ almost everywhere on $[0,1] \times \mathcal{X}$.

Generating field. We prove that Eq. (8) is the generating field for $\rho_t^\star := \lambda p_t + (1-\lambda)p'_t$. Differentiating $\rho_t^\star$ in time and applying the continuity equation to each task path gives

$$\begin{aligned}\frac{\partial}{\partial t}\rho_t^\star &= \lambda\frac{\partial}{\partial t}p_t + (1-\lambda)\frac{\partial}{\partial t}p'_t \\ &= -\lambda\nabla\cdot(p_t b_t) - (1-\lambda)\nabla\cdot(p'_t b'_t) \\ &= -\nabla\cdot\left[\lambda p_t b_t + (1-\lambda)p'_t b'_t\right] \\ &= -\nabla\cdot\left[\rho_t^\star\,\frac{\lambda p_t b_t + (1-\lambda)p'_t b'_t}{\lambda p_t + (1-\lambda)p'_t}\right] \\ &= -\nabla\cdot\left(\rho_t^\star\, v_t^\star\right), \end{aligned} \tag{38}$$

where the fourth equality multiplies and divides by $\rho_t^\star = \lambda p_t + (1-\lambda)p'_t$, which is allowed by the positivity assumption, and the last uses Eq. (8). Hence $\rho_t^\star$ satisfies the continuity equation with velocity $v_t^\star$. The boundary conditions follow from those of the task paths: $\rho_0^\star = \lambda p_0 + (1-\lambda)p'_0 = \mathcal{N}(\mathbf{0}_d, \mathbb{I}_d)$ since the two priors coincide, and $\rho_1^\star = \lambda q + (1-\lambda)q' = \tilde{q}$. □

A.7 Closure of the marginal mixture path

Let p_t and p'_t be marginal paths induced by the same fixed conditional Gaussian kernel $p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}; \alpha_t \boldsymbol{x}_1, \sigma_t^2 \mathbb{I}_d)$ over the task densities q and q' respectively, as in Eq. (17). By linearity of the integral, the mixture path of Prop. 3.1 satisfies

$$\begin{aligned}\rho_t^\star(\boldsymbol{x}) &= \lambda\, p_t(\boldsymbol{x}) + (1-\lambda)\, p'_t(\boldsymbol{x}) \\ &= \lambda \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, q(\boldsymbol{x}_1)\, \mathrm{d}\boldsymbol{x}_1 + (1-\lambda)\int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, q'(\boldsymbol{x}_1)\, \mathrm{d}\boldsymbol{x}_1 \\ &= \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\left[\lambda\, q(\boldsymbol{x}_1) + (1-\lambda)\, q'(\boldsymbol{x}_1)\right] \mathrm{d}\boldsymbol{x}_1 \\ &= \int p_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\, \tilde{q}(\boldsymbol{x}_1)\, \mathrm{d}\boldsymbol{x}_1\,. \end{aligned} \tag{39}$$

The mixture path is therefore itself a marginal path induced by the same kernel, with data distribution $\tilde{q}$ in place of q or q'. Combined with Prop. 3.1, which identifies $v_t^\star$ as its generating field, the pair $(\rho_t^\star, v_t^\star)$ belongs to the conditional Gaussian family required by Thm. 3.2.

Note that the shared kernel is essential: if the two task paths used different scheduling parameters, the two integrals could not be combined, and the mixture would in general not be a marginal path of either kernel, for any data distribution.

A.8 Proof of Thm. 3.2

Theorem 3.2 (restated)**.** *Let p_t, p'_t be marginal paths induced by a shared conditional Gaussian kernel with scheduling parameters α_t, σ_t (Sec. A.7), generated by velocity fields b_t, b'_t, and let ρ_t be a candidate path from the same family, with velocity v_t and shared prior $\rho_0 = p_0 = p'_0$. Fix $\lambda \in (0,1)$ and let $\alpha_t, \sigma_t > 0$ and $\dot{\alpha}_t\sigma_t - \alpha_t\dot{\sigma}_t > 0$ on $(0,1)$. The weighted terminal divergence then satisfies*

$$\begin{aligned}&\lambda D_{\mathrm{KL}}(p_1 \parallel \rho_1) + (1-\lambda) D_{\mathrm{KL}}(p'_1 \parallel \rho_1) \\ &\quad = \int_0^1 \kappa_t \Big(\lambda\, \mathbb{E}_{p_t}\left[\|v_t(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2\right] + (1-\lambda)\, \mathbb{E}_{p'_t}\left[\|v_t(\boldsymbol{x}) - b'_t(\boldsymbol{x})\|_2^2\right]\Big)\mathrm{d}t\,, \end{aligned}$$

with $\kappa_t := \alpha_t / \big(\sigma_t(\dot{\alpha}_t\sigma_t - \alpha_t\dot{\sigma}_t)\big) > 0$.

Proof. **Dynamic formulation of terminal D_{KL}.** Albergo et al. (2025) show that the terminal KL objective can be rewritten as a dynamic product between score and vector field differences (Lemma 21). Exemplifying this on the first term in Eq. (9) yields:

$$D_{\mathrm{KL}}(p_1 \parallel \rho_1) = \int_0^1 \mathbb{E}_{p_t}\Big[\big(\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log \rho_t(\boldsymbol{x})\big)\cdot\big(b_t(\boldsymbol{x}) - v_t(\boldsymbol{x})\big)\Big]\,\mathrm{d}t\,, \tag{40}$$

where both continuity equations have been used, and the initial divergence drops out since the shared prior gives $D_{\mathrm{KL}}(p_0 \parallel \rho_0) = 0$ (see Albergo et al. (2025) for the derivation).

From scores to velocities. As, by assumption, ρ_t, p_t and p'_t are marginal Gaussian flows, the score can be rewritten in terms of the vector field and the scheduling parameters α_t and σ_t (Zheng et al., 2023). Rearranging terms in Eq. (19) and, again, exemplifying the historical branch as in Eq. (40) we obtain:

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) = \frac{\alpha_t b_t(\boldsymbol{x}) - \dot{\alpha}_t \boldsymbol{x}}{\sigma_t(\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t)} \,. \tag{41}$$

Since ρ_t belongs to the same family with the same schedule, the identical relation holds with v_t in place of b_t. Subtracting the two, the affine terms cancel, and the score difference collapses to a velocity difference:

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log \rho_t(\boldsymbol{x}) = \frac{\alpha_t}{\sigma_t(\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t)} \big(b_t(\boldsymbol{x}) - v_t(\boldsymbol{x})\big) = \kappa_t \big(b_t(\boldsymbol{x}) - v_t(\boldsymbol{x})\big) \,. \tag{42}$$

This cancellation is where the restriction to a shared family is used: were the two paths governed by different schedules, the drift terms would not agree, and no such proportionality would hold.

Weighted combination. Substituting Eq. (42) into Eq. (40) turns the inner product into a squared norm,

$$D_{\mathrm{KL}}(p_1 \parallel \rho_1) = \int_0^1 \kappa_t \, \mathbb{E}_{p_t}\big[\|v_t(\boldsymbol{x}) - b_t(\boldsymbol{x})\|_2^2\big] \, \mathrm{d}t \,, \tag{43}$$

and the same argument applied to the incoming branch, with p'_t and b'_t in place of p_t and b_t, gives the analogous identity with the same κ_t, since both branches share α_t and σ_t. Taking the λ-weighted combination of the two and collecting the integrals yields Eq. (9). For the linear schedule of Sec. A.1, $\alpha_t = t$ and $\sigma_t = 1 - t$ give $\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t = (1 - t) + t = 1$, so that $\kappa_t = t/(1 - t)$. □

A.9 Proof of Prop. 3.3

Proposition 3.3 (restated)**.** *Let b_t, b'_t be the historical and incoming marginal velocities generating p_t, p'_t as in Prop. 3.1, let V be continuously differentiable in s and $\boldsymbol{x}$ on $0 \le s \le t \le 1$, and $\lambda \in (0, 1)$ with $\lambda p_s + (1 - \lambda) p'_s > 0$ on $[0, 1] \times \mathcal{X}$. Then the semigradient of $\mathcal{K}(\cdot\,; b, b')$ (Eq. (11)) vanishes if and only if $\mathcal{R}_{s,t}(V, v^\star) = 0$ for all $0 \le s < t \le 1$, with $v^\star$ from Eq. (8), whose unique solution is the mixture mean velocity $V^\star_{s,t}$ of Eq. (10).*

Proof. Recall the extended formula for the mean flow continual update objective:

$$\mathcal{K}(V; b, b') := \int_0^1 \int_s^1 \Big(\lambda \, \mathbb{E}_{\boldsymbol{x} \sim p_s}\big[\|\mathcal{R}_{s,t}(V, b)(\boldsymbol{x})\|_2^2\big] + (1 - \lambda) \, \mathbb{E}_{\boldsymbol{x} \sim p'_s}\big[\|\mathcal{R}_{s,t}(V, b')(\boldsymbol{x})\|_2^2\big]\Big) \, \mathrm{d}t \, \mathrm{d}s \,, \tag{44}$$

where, by Eq. (4), the two residuals read

$$\mathcal{R}_{s,t}(V, b)(\boldsymbol{x}) = V_{s,t}(\boldsymbol{x}) - \mathrm{sg}\Big[b_s(\boldsymbol{x}) + (t - s)\big(\partial_s V_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} V_{s,t}(\boldsymbol{x}) \, b_s(\boldsymbol{x})\big)\Big] \,, \tag{45}$$

$$\mathcal{R}_{s,t}(V, b')(\boldsymbol{x}) = V_{s,t}(\boldsymbol{x}) - \mathrm{sg}\Big[b'_s(\boldsymbol{x}) + (t - s)\big(\partial_s V_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} V_{s,t}(\boldsymbol{x}) \, b'_s(\boldsymbol{x})\big)\Big] \,. \tag{46}$$

As illustrated in Sec. A.3 and following Dieleman (2026), the semigradient induced by the stop-gradient operation ensures that the residual is affine in the velocity field. Denote $\boldsymbol{c} := (t - s) \, \partial_s V_{s,t}$ and $M := \mathbb{I}_d + (t - s) \nabla_{\boldsymbol{x}} V_{s,t}$, suppressing their dependence on $(s, t, \boldsymbol{x})$. We rewrite Eq. (45) and Eq. (46) as follows:

$$\mathcal{R}_{s,t}(V, b)(\boldsymbol{x}) = V_{s,t}(\boldsymbol{x}) - \mathrm{sg}\big[\boldsymbol{c}(\boldsymbol{x}) + M(\boldsymbol{x}) \, b_s(\boldsymbol{x})\big] \,, \tag{47}$$

$$\mathcal{R}_{s,t}(V, b')(\boldsymbol{x}) = V_{s,t}(\boldsymbol{x}) - \mathrm{sg}\big[\boldsymbol{c}(\boldsymbol{x}) + M(\boldsymbol{x}) \, b'_s(\boldsymbol{x})\big] \,. \tag{48}$$

The two share $\boldsymbol{c}$ and M, which depend on V but not on the velocity, and differ only in which field occupies the affine slot.

Stationarity. Write the expectations in Eq. (44) as integrals over $\boldsymbol{x}$ and perturb $V \to V + \varepsilon \, \eta$ for an arbitrary direction η. Because the stop-gradient holds $\boldsymbol{c}$ and M fixed, each residual moves by exactly $\varepsilon \, \eta$, so expanding the squares and collecting first-order terms gives

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon = 0} \mathcal{K}(V + \varepsilon \eta; b, b') = \int_0^1 \int_s^1 \int \widetilde{G}(s, t, \boldsymbol{x}) \cdot \eta(s, t, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t \, \mathrm{d}s \,, \tag{49}$$

with semigradient density

$$\widetilde{G}(s,t,\boldsymbol{x}) := 2\Big(\lambda\, p_s(\boldsymbol{x})\, \mathcal{R}_{s,t}(V,b)(\boldsymbol{x}) + (1-\lambda)\, p'_s(\boldsymbol{x})\, \mathcal{R}_{s,t}(V,b')(\boldsymbol{x})\Big)\,, \tag{50}$$

where the two densities appear as pointwise weights. This vanishes for every η if and only if $\widetilde{G}$ vanishes at every $(s,t,\boldsymbol{x})$. Substituting Eq. (47) and Eq. (48), the common term $V_{s,t}-\boldsymbol{c}$ is weighted by $\lambda p_s + (1-\lambda)p'_s$ while the velocity slot is weighted by $\lambda p_s b_s + (1-\lambda)p'_s b'_s$, so dividing by the total weight leaves

$$V_{s,t}(\boldsymbol{x}) = \boldsymbol{c}(\boldsymbol{x}) + M(\boldsymbol{x})\,\frac{\lambda p_s(\boldsymbol{x}) b_s(\boldsymbol{x}) + (1-\lambda)p'_s(\boldsymbol{x})b'_s(\boldsymbol{x})}{\lambda p_s(\boldsymbol{x}) + (1-\lambda)p'_s(\boldsymbol{x})} = \boldsymbol{c}(\boldsymbol{x}) + M(\boldsymbol{x})\, v^\star_s(\boldsymbol{x})\,, \tag{51}$$

where the division is admitted by the positivity assumption and the ratio is the mixture velocity of Eq. (8). Hence $\mathcal{R}_{s,t}(V,v^\star)=0$ for all $0\le s<t\le 1$: the density weighting is inherited rather than imposed, since the stop-gradient leaves each residual affine in its velocity, and a weighted sum of affine conditions is the condition at the weighted velocity.

Uniqueness. The stationarity condition constrains V only through its derivative. Following the sufficiency argument of Geng et al. (2026a, Sec. B.3), collecting $-V_{s,t}+(t-s)\frac{\mathrm{d}}{\mathrm{d}s}V_{s,t}$ into the form $\frac{\mathrm{d}}{\mathrm{d}s}\big[(t-s)V_{s,t}\big]$ by the product rule, $\mathcal{R}_{s,t}(V,v^\star)=0$ reads

$$\frac{\mathrm{d}}{\mathrm{d}s}\Big[(t-s)\, V_{s,t}(\boldsymbol{x}^\star_s)\Big] = -\, v^\star_s(\boldsymbol{x}^\star_s)\,, \tag{52}$$

along the trajectory $\boldsymbol{x}^\star_\tau$ solving Eq. (3) with velocity $v^\star$. Integrating Eq. (52) in s recovers the two sides only up to constants,

$$(t-s)\, V_{s,t}(\boldsymbol{x}^\star_s) + C_1 = \int_s^t v^\star_\tau(\boldsymbol{x}^\star_\tau)\,\mathrm{d}\tau + C_2\,, \tag{53}$$

which coincide since both sides vanish at $s=t$: the left by the factor $(t-s)$, for any V finite on the diagonal, and the right because the integration limits meet. Hence

$$(t-s)\, V_{s,t}(\boldsymbol{x}^\star_s) = \int_s^t v^\star_\tau(\boldsymbol{x}^\star_\tau)\,\mathrm{d}\tau\,, \tag{54}$$

which is Eq. (10): the stationary point of $\mathcal{K}(\cdot\,;b,b')$ is the mean velocity of the mixture path. As Geng et al. (2026a) note, this relies on parameterizing the average velocity rather than the displacement: had $S_{s,t} := (t-s)V_{s,t}$ been modeled directly, $S_{t,t}=0$ would have to be imposed as a separate boundary condition. □

A.10 A STOCHASTIC VIEW ON THE MIXTURE OPTIMIZATION PROBLEM

Consider the diffusion counterpart of Sec. A.1, where sampling from q and q' is governed by

$$\begin{aligned} \mathbb{P}:&\quad \mathrm{d}X_t = \mu_t(X_t)\,\mathrm{d}t + \epsilon_t\,\mathrm{d}B_t\,, \qquad X_0\sim\mathcal{N}(\mathbf{0}_d,\mathbb{I}_d),\ X_1\sim q\,,\\ \mathbb{P}':&\quad \mathrm{d}X_t = \mu'_t(X_t)\,\mathrm{d}t + \epsilon_t\,\mathrm{d}B_t\,, \qquad X_0\sim\mathcal{N}(\mathbf{0}_d,\mathbb{I}_d),\ X_1\sim q'\,, \end{aligned} \tag{55}$$

with B_t a standard Brownian motion and $\epsilon_t\in\mathbb{R}_+$ a time-dependent diffusion coefficient. The marginals p_t, p'_t at time t evolve under the corresponding Fokker–Planck equations,

$$\partial_t p_t = -\nabla\cdot(\mu_t p_t) + \tfrac{1}{2}\epsilon_t^2\Delta p_t\,, \qquad \partial_t p'_t = -\nabla\cdot(\mu'_t p'_t) + \tfrac{1}{2}\epsilon_t^2\Delta p'_t\,, \tag{56}$$

and the drifts decompose through the probability-flow identity as

$$\mu_t = b_t + \tfrac{1}{2}\epsilon_t^2\nabla_{\boldsymbol{x}}\log p_t\,, \qquad \mu'_t = b'_t + \tfrac{1}{2}\epsilon_t^2\nabla_{\boldsymbol{x}}\log p'_t\,, \tag{57}$$

where b_t, b'_t are the velocity fields of the probability-flow ODEs, which generate the same marginals p_t, p'_t through the continuity equation. Formulating the mixture problem over path measures rather than densities, Girsanov's theorem reduces the divergence between two such diffusions to an integrated quadratic drift mismatch, weighted by the inverse diffusion coefficient. Similar to Thorkelsdottir & Banerjee (2026) but applied to the forward KL between path measures, we obtain the following proposition.

Proposition A.1. *Let $\mathbb{P}, \mathbb{P}'$ be the path measures of Eq. (55), with marginals p_t, p'_t, and let $\tilde{\mathbb{P}}$ range over diffusions with the same $\epsilon_t > 0$ and initial condition. Fix $\lambda \in (0,1)$. Then*

$$\arg\min_{\tilde{\mathbb{P}}} \ \lambda D_{\mathrm{KL}}(\mathbb{P} \,\|\, \tilde{\mathbb{P}}) + (1-\lambda) D_{\mathrm{KL}}(\mathbb{P}' \,\|\, \tilde{\mathbb{P}}) \tag{58}$$

admits the unique minimizer $\tilde{\mathbb{P}}^\star : \ \mathrm{d}X_t = \mu_t^\star(X_t)\,\mathrm{d}t + \epsilon_t\,\mathrm{d}B_t$, with

$$\mu_t^\star(\boldsymbol{x}) := \frac{\lambda\, p_t(\boldsymbol{x})\, \mu_t(\boldsymbol{x}) + (1-\lambda)\, p'_t(\boldsymbol{x})\, \mu'_t(\boldsymbol{x})}{\lambda\, p_t(\boldsymbol{x}) + (1-\lambda)\, p'_t(\boldsymbol{x})}\,, \tag{59}$$

whose marginal law is the mixture path $\rho_t^\star = \lambda p_t + (1-\lambda) p'_t$ of Prop. 3.1.

Proof. **Minimizer.** Since $\mathbb{P}$, $\mathbb{P}'$ and $\tilde{\mathbb{P}}$ share the diffusion coefficient ϵ_t and the initial condition, Girsanov's theorem gives

$$D_{\mathrm{KL}}(\mathbb{P} \,\|\, \tilde{\mathbb{P}}) = \int_0^1 \frac{1}{2\epsilon_t^2}\, \mathbb{E}_{p_t}\left[\|\mu_t(\boldsymbol{x}) - \tilde{\mu}_t(\boldsymbol{x})\|_2^2\right] \mathrm{d}t\,, \tag{60}$$

and likewise for $\mathbb{P}'$ with p'_t and μ'_t. The weighted objective is therefore the functional of Eq. (36) with μ_t, μ'_t in place of b_t, b'_t and an extra positive weight $1/(2\epsilon_t^2)$, which does not affect the pointwise minimization. The argument of Sec. A.6 applies and yields Eq. (59) as the unique minimizer.

Marginal law. Taking the λ-weighted combination of the two equations in Eq. (56) and using $\mu_t^\star \rho_t^\star = \lambda p_t \mu_t + (1-\lambda) p'_t \mu'_t$, which is Eq. (59) cleared of its denominator, gives

$$\partial_t \rho_t^\star = -\nabla \cdot \left(\mu_t^\star \rho_t^\star\right) + \tfrac{1}{2}\epsilon_t^2 \Delta \rho_t^\star\,, \tag{61}$$

so $\rho_t^\star$ is the marginal law of $\tilde{\mathbb{P}}^\star$. The boundary conditions carry over as in Sec. A.6. □

Substituting Eq. (57) into Eq. (59) and using $p\,\nabla_{\boldsymbol{x}} \log p = \nabla_{\boldsymbol{x}} p$, the score terms recombine into the score of $\rho_t^\star$, leaving

$$\mu_t^\star = v_t^\star + \tfrac{1}{2}\epsilon_t^2 \nabla_{\boldsymbol{x}} \log \rho_t^\star\,, \tag{62}$$

with $v_t^\star$ exactly the velocity field of Eq. (8). The deterministic and stochastic formulations therefore select the same mixture: the optimal drift is the optimal velocity plus the score correction its own marginal requires.

A.11 Additional details on the sequential mean flow update

Parameterized objective. Following Sec. A.2 and mirroring Eq. (12) for the instantaneous field, we derive a counterpart of Eq. (11) across multiple sequential updates. Specifically, at step k:

$$\begin{aligned}\mathcal{K}^{k+1} = \int_0^1 \int_s^1 \Big(& \lambda_k\, \mathbb{E}_{\boldsymbol{x}_s \sim p_s^{\theta^{\le k}}}\left[\|\mathcal{R}_{s,t}(V^{\theta^{\le k+1}}, b^{\theta^{\le k}})(\boldsymbol{x}_s)\|_2^2\right] \\ & + (1-\lambda_k)\, \mathbb{E}_{\boldsymbol{x}_1 \sim q_{k+1},\, \boldsymbol{x} \sim p_{s|1}(\cdot|\boldsymbol{x}_1)}\left[\|\mathcal{R}_{s,t}(V^{\theta^{\le k+1}}, b_{\cdot|1}(\cdot \mid \boldsymbol{x}_1))(\boldsymbol{x})\|_2^2\right]\Big)\, \mathrm{d}t\, \mathrm{d}s\,.\end{aligned} \tag{63}$$

The two backends differ in how the historical samples $\boldsymbol{x}_s$ are obtained.

- *ODE backend.* No mean velocity is learned. Historical states come from integrating $b^{\theta^{\le k}}$ along Eq. (3), at a cost that grows with the solver budget, and only $v^{\theta^{\le k+1}}$ is carried forward.
- *MeanFlow backend.* Each update additionally trains a mean velocity network $V^{\theta^{\le k+1}}$ under Eq. (63), whose flow map supplies the historical states at the next update in a few evaluations. Both fields are carried forward, and the first task, having no history, trains them on its own data alone, by flow matching and single-teacher distillation, respectively.

$V^{\theta^{\le k+1}}$ is therefore auxiliary: it exists only to reduce the number of network evaluations needed to train the joint-task field at the following update. Note that the incoming branch is trained in a data-conditional way, as in Geng et al. (2026a) and explained in Sec. A.3: substituting the conditional velocity for the marginal one leaves the semigradient unchanged in expectation.

Departure from the population objective. Prop. 3.3 characterizes the fixed point of $\mathcal{K}$ under the true marginal fields and exact samples from p_s, p'_s, neither of which is available at training time. The historical field is replaced by the previous model $b_s^{\theta_{\leq k}}$, itself approximate. The incoming field is never formed, and while the conditional substitution is exact in expectation, it introduces variance at finite batch size, and samples are drawn through the learned flow map, so the sampling measure is $(\boldsymbol{X}_{0,s}^{\theta_{\leq k}})_\sharp p_0$ rather than p_s. Eq. (63) therefore differs from $\mathcal{K}$, and the fixed-point characterization transfers only up to these three approximations.

A.12 Integration on the selection of λ

The coefficient λ_k governs not only a single update but the weight each task retains across the whole stream. Unrolling Eq. (5) over the stream expresses the terminal density as a mixture over all tasks seen, with weights determined by the schedule $\{\lambda_k\}_{k=1}^{K-1}$. Holding $\lambda_k = \lambda$ constant gives the incoming task weight $1-\lambda$, the previous one $\lambda(1-\lambda)$, and any task introduced m updates earlier $\lambda^m(1-\lambda)$, with the initial task retaining λ^{K-1} since it never entered as an incoming task: the weight *decays geometrically* in task age, so a constant rate discounts older tasks rather than balancing them.

Uniform weight instead requires $\lambda_k = k/(k+1)$, since at step k the historical model carries k tasks at weight $1/k$ each, and weighting it by $k/(k+1)$ against $1/(k+1)$ for the incoming task leaves all $k+1$ tasks at $1/(k+1)$. Selecting λ_k is therefore a choice of how the stream is weighted overall, not only of the current stability-plasticity trade-off, and the two schedules above correspond to the settings compared in Sec. 4. The adaptive policy admits no such closed form, since the weights depend on validation outcomes at each update, so the terminal mixture it induces is known only after the fact.

A.13 Efficiency and stability gains

Evaluation count. Fix a solver step size h and write b for the frozen historical field. Generative replay integrates every replay sample from 0 to 1, costing $\lceil 1/h \rceil$ evaluations. Trajectory replay integrates only to $t \sim \mathcal{U}(0,1]$ and then evaluates b_t once at the anchor, costing $\lceil t/h \rceil + 1$, so that in expectation

$$\mathbb{E}_t\big[\lceil t/h \rceil + 1\big] \;\approx\; \frac{1}{2h} + 1 \qquad \text{against} \qquad \frac{1}{h}\,, \tag{64}$$

approaching a factor of two as $h \to 0$. At $h = 0.05$, for instance, a full generation costs 20 evaluations while trajectory replay averages 11.5, a saving of 42.5% that grows as the solver is refined.

Accumulated solver error. Intuitively, simulating less results in lower discretization error. Let $\boldsymbol{x}_\tau$ solve Eq. (3) exactly under the frozen field b, and let $\tilde{\boldsymbol{x}}_\tau$ denote its Euler approximation at step size h from the same initial state. Assume, in addition to the standing assumptions, that b is Lipschitz in $\boldsymbol{x}$ uniformly in time, with constant $L > 0$, and that the exact path has bounded curvature, $\sup_\tau \|\ddot{\boldsymbol{x}}_\tau\| \leq M_2$. The classical global error bound for Euler's method (Süli & Mayers, 2003) then gives, with exact initial data,

$$\|\tilde{\boldsymbol{x}}_t - \boldsymbol{x}_t\| \;\leq\; \frac{M_2 h}{2L}\Big(e^{Lt} - 1\Big)\,, \tag{65}$$

which is increasing in t: the local error of each step is $O(h^2)$, and the flow amplifies errors already committed by at most $(1 + hL)$ per step. Generative replay always pays Eq. (65) at $t = 1$, whereas trajectory replay pays it at a uniformly drawn anchor, so the two stand in the ratio

$$\frac{\mathbb{E}_t\big[e^{Lt} - 1\big]}{e^L - 1} = \frac{(e^L - 1)/L - 1}{e^L - 1} \;<\; \frac{1}{2}\,, \tag{66}$$

which approaches $1/2$ from below as $L \to 0$ and decays like $1/L$ for large L, since e^{Lt} concentrates near the endpoint and a uniformly drawn anchor mostly avoids it. Stopping early therefore reduces the worst-case error in the replayed state, and by more when the field varies rapidly.

Where the error lands. The two objectives also differ in what that error corrupts. Generative replay treats the simulated endpoint $\tilde{\boldsymbol{x}}_1$ as if it were a real historical sample, so the student is fit to a corrupted dataset: by Sec. A.4 its optimum reproduces the law of $\tilde{\boldsymbol{x}}_1$ rather than the historical density, and training longer or harder cannot recover the difference. Trajectory replay instead asks the frozen model what its velocity is at $\tilde{\boldsymbol{x}}_t$, and the answer is correct whatever $\tilde{\boldsymbol{x}}_t$ happens to be, since it is the *teacher's own value* at that point. Solver error then only shifts where along the trajectory the student is supervised, never what it is taught there, so the teacher is still reproduced exactly on the states that are visited.

Caveats with trajectory replay. The argument only covers the states the solver actually reaches. A solver that systematically avoids a region leaves the student unconstrained there, and with finite capacity the shift in which states are visited is not innocuous either, since supervising the wrong points trades approximation error between regions. Fig. 7 and the matched-backend comparisons of Sec. D.3 measure the combined effect of both mechanisms.

A.14 Relation to feature replay

Trajectory replay is loosely connected with feature replay approaches, sharing a similar motivating diagnosis: replaying generated images is expensive, and errors in those images enter training as if they were data. Liu et al. (2020) address this by splitting the network at a layer, freezing and distilling the feature extractor, and replaying intermediate features rather than images. A generator produces features for past classes, which are paired with their old labels and fed to the classifier. The gain is that feature distributions are lower-dimensional and far easier to model than pixels, so the generator can be accurate where an image generator would not be.

A continuous-depth view. The same structure appears in our setting. Under the neural ODE correspondence (Chen et al., 2018), integration time is depth and a flow is a *weight-tied residual network* (He et al., 2016) whose block at depth t is

$$\boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + \Delta t\, v_t(\boldsymbol{x}_t)\,, \tag{67}$$

so v_t is the residual branch and $\boldsymbol{x}_t$ an intermediate representation. Generative replay runs every block to obtain the output $\boldsymbol{x}_1$ and treats it as a sample. Trajectory replay runs a prefix and supervises the residual branch at the depth it reached. Two consequences follow. The expected prefix is half the depth, which is the accounting of Sec. A.13. And because the blocks share parameters, a single sampled t trains the whole network, so no layer-wise schedule is needed.

Where the two differ. The resemblance stops at what is replayed. Feature replay generates a *substitute for a real activation* and pairs it with an old label, so the generator's fidelity decides whether the training pair is correct, exactly as in standard generative replay. The requirement is moved to an easier space rather than removed. Trajectory replay generates no substitute for anything. The state $\boldsymbol{x}_t$ is a probe location, and the target is the frozen model's own velocity there, which is a valid input-output pair wherever the solver landed (Sec. A.13). The split also differs: feature replay's is architectural and fixed. Ours is temporal and resampled every step, so no layer is privileged.

B Hardware details

All experiments run on a Slurm cluster, one NVIDIA H100 80GB GPU per job, with no multi-GPU or multi-node training anywhere. Requested CPU and host memory scale with the input pipeline: 8 cores and 32GB for the synthetic simulations, 8 cores and 48GB for image continual learning, 12 cores and 64GB for microscopy, and 16 cores and 512GB for transcriptomics. Per-task wall time ranges from minutes on the synthetic streams to roughly 4 hours per sequential update on ImageNet-100, the most expensive setting.

C ALGORITHM

Algorithm 1: Trajectory replay continual update at step k, ODE backend

Input: Frozen historical field $b^{\theta_{\le k}}$; incoming dataset $\mathcal{D}_{k+1} \sim q_{k+1}$; weight λ_k; teacher step size h; time cutoff ε; optimizer steps T

Output: The updated joint-task field $v^{\theta_{\le k+1}}$

$\theta_{\le k+1} \leftarrow \mathrm{copy}(\theta_{\le k})$; // warm start; $b^{\theta_{\le k}}$ stays frozen

for $1, \dots, T$ **do**

  $t \sim \mathcal{U}(\varepsilon, 1]$;

  /* Historical branch: partial replay */

  $\boldsymbol{x}_0 \sim p_0$;

  $\boldsymbol{x}_t \leftarrow \mathrm{ODESolve}(b^{\theta_{\le k}}, \boldsymbol{x}_0, 0 \to t, h)$; // $\lceil t/h \rceil$ evals

  $\mathcal{L}_{\mathrm{hist}} \leftarrow \|v_t^{\theta_{\le k+1}}(\boldsymbol{x}_t) - b_t^{\theta_{\le k}}(\boldsymbol{x}_t)\|_2^2$; // one more eval

  /* Incoming branch: conditional flow matching */

  $\boldsymbol{x}_1 \sim \mathcal{D}_{k+1}, \boldsymbol{x} \sim p_{t|1}(\cdot \mid \boldsymbol{x}_1)$;

  $\mathcal{L}_{\mathrm{inc}} \leftarrow \|v_t^{\theta_{\le k+1}}(\boldsymbol{x}) - b_{t|1}(\boldsymbol{x} \mid \boldsymbol{x}_1)\|_2^2$;

  $\mathcal{J}^{k+1} \leftarrow \lambda_k \mathcal{L}_{\mathrm{hist}} + (1-\lambda_k)\mathcal{L}_{\mathrm{inc}}$; // Eq. (12)

  $\theta_{\le k+1} \leftarrow \mathrm{OptStep}(\theta_{\le k+1}, \nabla_{\theta_{\le k+1}} \mathcal{J}^{k+1})$;

return $v^{\theta_{\le k+1}}$;

Algorithm 2: Trajectory replay continual update at step k, MeanFlow backend

Input: Frozen field $b^{\theta_{\le k}}$ and frozen mean velocity $V^{\theta_{\le k}}$; incoming dataset $\mathcal{D}_{k+1} \sim q_{k+1}$; weight λ_k; replay jumps N; diagonal rate p_{diag}; time law $\mathcal{T}$; time cutoff ε; optimizer steps T

Output: The updated joint-task field $v^{\theta_{\le k+1}}$ and mean velocity $V^{\theta_{\le k+1}}$

$\theta_{\le k+1} = (\theta^v, \theta^V) \leftarrow \mathrm{copy}(\theta_{\le k})$; // warm start; teachers stay frozen

for $1, \dots, T$ **do**

  $t \sim \mathcal{U}(\varepsilon, 1]$; // one shared time per step

  /* Few-step replay through the frozen flow map */

  $\boldsymbol{x} \sim p_0, \quad u_0 \leftarrow 0$;

  for $n = 1, \dots, N$ **do**

    $u_n \leftarrow nt/N$;

    $\boldsymbol{x} \leftarrow \boldsymbol{x} + (u_n - u_{n-1})\, V_{u_{n-1}, u_n}^{\theta_{\le k}}(\boldsymbol{x})$;

  $\boldsymbol{x}_t \leftarrow \boldsymbol{x}$; // N evals, independent of t

  /* Velocity field */

  $\boldsymbol{x}_1 \sim \mathcal{D}_{k+1}, \quad \boldsymbol{x}' \sim p_{t|1}(\cdot \mid \boldsymbol{x}_1)$;

  $\mathcal{J}^{k+1} \leftarrow \lambda_k \|v_t^{\theta_{\le k+1}}(\boldsymbol{x}_t) - b_t^{\theta_{\le k}}(\boldsymbol{x}_t)\|_2^2 + (1-\lambda_k)\|v_t^{\theta_{\le k+1}}(\boldsymbol{x}') - b_{t|1}(\boldsymbol{x}' \mid \boldsymbol{x}_1)\|_2^2$;

  $\theta^v \leftarrow \mathrm{OptStep}(\theta^v, \nabla_{\theta^v} \mathcal{J}^{k+1})$;

  /* Mean velocity, carried to the next update */

  $s \leftarrow t$; // reuses the shared time and the rollout $\boldsymbol{x}_t$

  $\tau_{\mathrm{o}}, \tau_{\mathrm{n}} \leftarrow s + (1-s)u$ with $u \sim \mathcal{T}$, each set to s w.p. p_{diag};

  $\boldsymbol{x}'' \sim p_{s|1}(\cdot \mid \boldsymbol{x}_1)$;

  $\mathcal{K}^{k+1} \leftarrow \lambda_k \|\mathcal{R}_{s,\tau_{\mathrm{o}}}(V^{\theta_{\le k+1}}, b^{\theta_{\le k}})(\boldsymbol{x}_t)\|_2^2 + (1-\lambda_k)\|\mathcal{R}_{s,\tau_{\mathrm{n}}}(V^{\theta_{\le k+1}}, b_{\cdot|1}(\cdot \mid \boldsymbol{x}_1))(\boldsymbol{x}'')\|_2^2$;

  $\theta^V \leftarrow \mathrm{OptStep}(\theta^V, \widetilde{G}_{\theta^V}[\mathcal{K}^{k+1}])$; // semigradient, Sec. A.3

return $v^{\theta_{\le k+1}}$, $V^{\theta_{\le k+1}}$;

D EXPERIMENTAL DETAILS

D.1 CONTROLLED MIXTURE RECOVERY

D.1.1 EXPERIMENTAL DETAILS

Data. Each task is an isotropic Gaussian in $\mathbb{R}^d$ with $\sigma = 0.35$, with mode means on the first coordinate at ± 1.5 and all remaining coordinates pure noise axes. Task 0 (-1.5) is the historical mode and task 1 ($+1.5$) the incoming one. We draw 10,000 points per task at $d = 2$ and 100,000 at $d \in \{128, 256\}$, with data seed 0. At $d = 2$ we sweep $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and, for each value, sample an independent reference set of the same size from $\tilde{p} = \lambda p_{\text{hist}} + (1 - \lambda) p_{\text{new}}$, used only for evaluation. The higher-dimensional runs fix $\lambda = 0.5$.

Model. The velocity field is an unconditional MLP of width 256 and depth 4, with a 64-dimensional sinusoidal time embedding, trained on the linear interpolant $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$ with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$ (Tong et al., 2024a). We sample using Euler with step size 0.05 (20 evaluations). Given the task's simplicity, we did not ablate model capacity.

Pretraining. The task-0 model trains for 400 epochs with batch size 256 and Adam (Kingma & Ba, 2014) at learning rate 10^{-3}, seed 0. A single checkpoint is shared by every update run, so all differences across methods and seeds come from the update alone.

Update. Each update trains for 400 epochs with the same optimizer settings, for seeds $\{0, 1, 2\}$ at every λ in the grid. We compare naive fine-tuning, which continues training the historical checkpoint on incoming data alone with no replay term, against generative and trajectory replay, plus a joint-training reference fit directly on samples from $\tilde{p}$. Generative replay integrates the frozen teacher to $t = 1$ and treats the samples as data, drawing a fraction λ of every batch from the teacher and the remainder from the incoming task, then taking a *single flow-matching loss* over the combined batch. Trajectory replay splits each batch evenly, draws one shared $t \sim \mathcal{U}(10^{-4}, 1]$ per step, integrates the teacher only to t, and matches the student field against the teacher field there, with λ weighting the historical and incoming terms of the loss.

Evaluation. We generate 4,000 samples per run with the same sampler and report two quantities. Generated historical mass $\hat{\pi}_0$ (Fig. 2a) is the fraction of samples whose nearest mode mean is mode 0. Energy distance is computed between the generated samples and the held-out reference set, restricted to the active coordinate on which the mode means differ (Fig. 7). All figures show three seeds.

Sampling-budget comparison (Fig. 7). The teacher-cost curves use $d \in \{128, 256\}$ at $\lambda = 0.5$ (equal-task), training for 513 epochs (200,070 optimizer steps) with evaluation every 10 epochs on 2,000 samples, for seeds $\{0, 1, 2\}$. Cumulative teacher evaluations follow analytically from the fixed Euler protocol: at 100,000 points and batch size 256, there are 390 steps per epoch, generative replay spends $\lceil 1/0.05 \rceil = 20$ evaluations per step, and trajectory replay spends $\mathbb{E}[\lceil t/0.05 \rceil] + 1 = 11.5$, since t is uniform on $(0, 1]$, integrating only to an intermediate time plus one frozen-field evaluation at the anchor. Over the full run, this is 4.0×10^6 teacher evaluations for generative replay against 2.3×10^6 for trajectory replay.

D.1.2 ADDITIONAL RESULTS

We complement Fig. 2 with two additional views. Fig. 6a shows the mixture behavior at $\lambda \in \{0.1, 0.5, 0.9\}$, where generated samples track the prescribed weight across the full range. Fig. 6b contrasts this with naive fine-tuning, which collapses onto the incoming mode regardless of λ and recovers no historical mass.

Fig. 7 plots the energy distance against cumulative teacher evaluations rather than training steps, isolating the cost of replay from the cost of optimization. Trajectory replay reaches a given energy at a smaller budget in both dimensions, and the gap widens with d: at $d = 128$ the two curves converge once the target is essentially matched, whereas at $d = 256$ generative replay needs close to twice the budget to reach the level trajectory replay attains at $\approx 10^6$ evaluations.

Part of this follows from the accounting above: 11.5 evaluations per step against 20 for full generation. The rest is robustness to the solver, analyzed in Sec. A.13: discretization error displaces the replayed state but leaves the teacher's velocity there a valid target, whereas the same error corrupts a

generated sample that is then fit as data. The matched-backend comparisons in Sec. D.3 isolate this at equal replay cost.

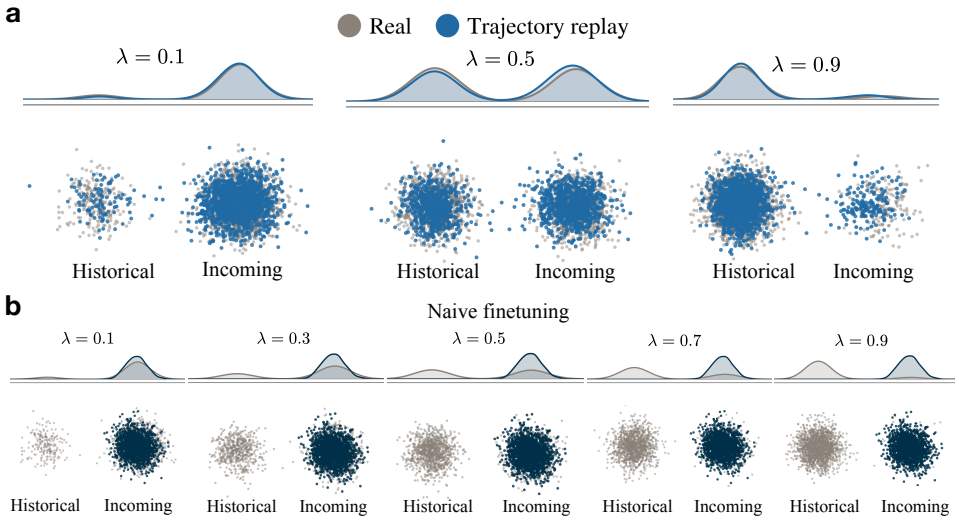

Figure 6: **Two-mode mixture updates.** (**a**) Samples and one-dimensional marginals from the real mixture and from trajectory replay after the update, for historical and incoming modes at $\lambda \in \{0.1, 0.5, 0.9\}$. (**b**) Samples and marginals from the real mixture and from naive fine-tuning after the update, shown for $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

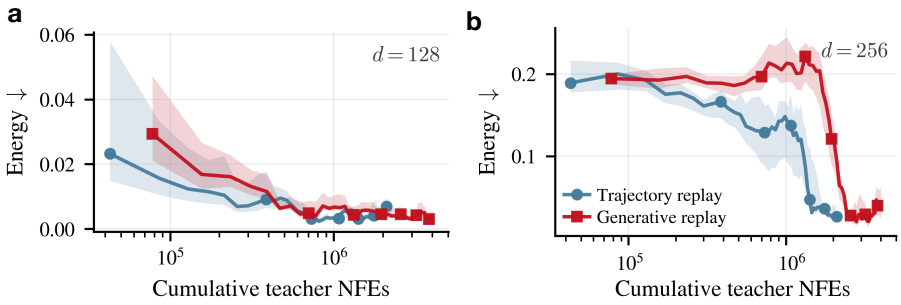

Figure 7: **Energy distance versus teacher sampling budget.** Energy distance to the target mixture, computed on the active coordinate that separates the modes, as a function of cumulative teacher NFEs for trajectory replay and generative replay, in (**a**) $d = 128$ and (**b**) $d = 256$. Lines show the mean and shaded bands the variability across three seeds. The x-axis is logarithmic and lower energy is better.

D.2 Sequential updates on a ring-structured Gaussian mixture

D.2.1 Experimental details

Data. Five isotropic Gaussians in $\mathbb{R}^2$ with $\sigma = 0.35$ are placed on a ring of radius 3.0, starting at the top and proceeding clockwise, giving a nearest-neighbor separation of 3.53 between consecutive task means. Tasks arrive in that order, T_0 through T_4. Each task has independent train, validation, and test splits of $10{,}000$, $2{,}000$, and $5{,}000$ points, with data seed 0. The balanced final target places 20% of the mass on each task.

Model. The velocity field is the same unconditional MLP as Sec. 4.1, of width 256 and depth 4 with a 64-dimensional sinusoidal time embedding, trained on the linear interpolant with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$. We sample using Euler with step size 0.05.

Training. Every task, including T_0, receives a budget of $20{,}000$ optimizer steps, rounded up to whole epochs at batch size 256. Optimization uses Adam at 10^{-3} with gradient-norm clipping at 1.0. The whole stream is repeated for seeds $\{0, 1, 2\}$, which vary initialization, minibatch order, and sampling noise.

Replay policies. We compare naive fine-tuning against generative and trajectory replay under three ways of setting the replay weight.

- *Constant.* $\lambda \in \{0.25, 0.5, 0.75, 0.9\}$ held fixed across the stream.
- *Equal-task.* $\lambda_k = k/(k+1)$ at update k, the parameter-free choice that keeps all seen tasks equally weighted.
- *Adaptive.* The full update is run once for each candidate in $\{0.25, 0.5, 2/3, 0.75, 0.8, 0.9\}$, and we keep the model whose $4{,}000$ generated samples minimize energy distance to a balanced validation mixture over all tasks seen so far, so the selection uses only validation data available at that point in the stream and no test data. The chosen model becomes the historical model for the next update, and the sweep is repeated there, which costs one full training run per candidate per update.

Running a policy to the end of the stream therefore fixes a target composition over tasks: equal-task yields the *uniform target*, whereas a constant λ discounts older tasks geometrically (Sec. A.12). The generated task mass is scored against that target.

Evaluation. After every update, we generate $10{,}000$ samples and report two quantities against the test splits of the seen tasks. Energy distance is computed against the balanced mixture formed by concatenating equal-size test sets across seen modes. Generated task mass assigns each sample to its nearest task mean, adds a pseudocount of 0.5 to keep the divergence finite when a task receives no samples, and is compared with the uniform target by forward categorical KL. All figures show three seeds.

D.2.2 Additional results

Stream layout and final samples. Fig. 8a shows the five-task ring layout and the order in which tasks arrive, and Fig. 8b shows the samples drawn after the final update: naive fine-tuning retains only T_4, generative replay recovers all five modes with visibly uneven mass, and trajectory replay comes closest to the balanced target.

Weighting policies. Fig. 9a places every final model on the trade-off between retaining earlier tasks and fitting the newest one. The constant-λ runs trace a clear frontier, from $\lambda = 0.9$ retaining history at the cost of T_4 to $\lambda = 0.25$ doing the reverse, with $\lambda = 0.5$ and 0.75 in between. Both scheduled policies sit near the uniform target, well inside the frontier that any single constant weight can reach, which is the practical case for scheduling λ_k rather than tuning it once. Fig. 9b shows that the adaptive sweep selects weights closely tracking the equal-task rule $\lambda_k = k/(k+1)$ at every update, for both replay methods. This is expected, since the five modes carry equal mass and the balanced criterion is optimized by exactly the composition equal-task produces in closed form: the agreement validates the adaptive procedure rather than separating the two schedules, and explains why they perform comparably in Fig. 3b.

Stream evolution. Fig. 10 follows both metrics as the stream progresses rather than only at the end. Naive fine-tuning degrades within the first update and stays flat thereafter, since each update

discards whatever the previous one retained. Both replay methods remain two to three orders of magnitude below it, and their curves separate progressively: generative replay drifts upward as tasks accumulate, on both energy distance and task-mass divergence, whereas trajectory replay stays roughly level. This is the compounding effect discussed in Sec. A.13, where simulation error in the replayed samples enters the next historical model and is inherited by every subsequent update.

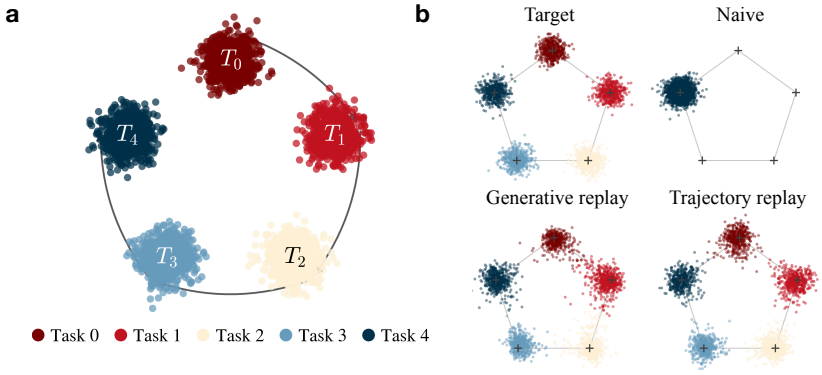

Figure 8: **Five-task Gaussian ring stream.** (**a**) The five task distributions $T_0, \ldots, T_4$ placed on a ring, with the arrow indicating the order in which they are introduced. (**b**) Samples drawn after the final update for the balanced target and for naive fine-tuning, generative replay, and trajectory replay. Crosses mark the task centers, and colors follow the task legend in (**a**).

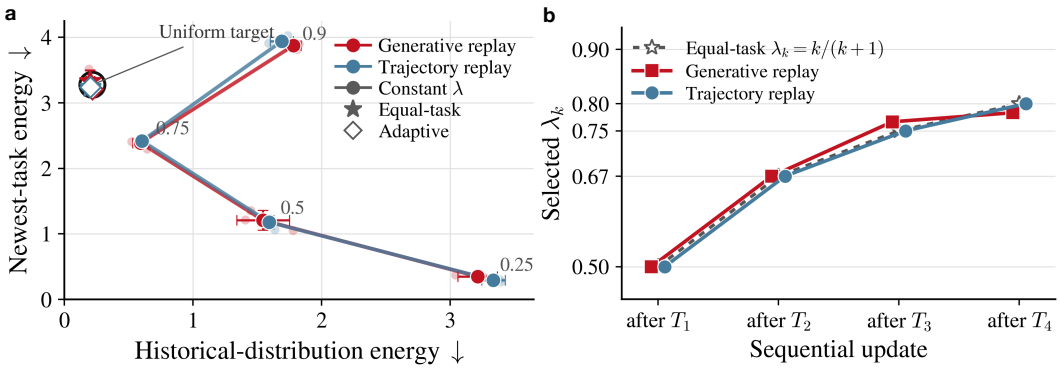

Figure 9: **Replay-weight policies on the five-task stream.** (**a**) Trade-off between retaining earlier tasks and fitting the newest one after the final update: each marker is one final model, positioned by its energy distance to the historical task distributions (x) and to the newest task (y), so points closer to the origin are better on both axes. Colors distinguish generative from trajectory replay, marker shapes distinguish constant-λ, equal-task, and adaptive policies, and the number next to each constant-λ marker is its λ value. The circled point near the origin marks the uniform target. (**b**) Replay weight λ_k selected at each sequential update for adaptive generative and trajectory replay, with the equal-task rule $\lambda_k = k/(k+1)$ shown as a dashed reference. Error bars show variability across seeds.

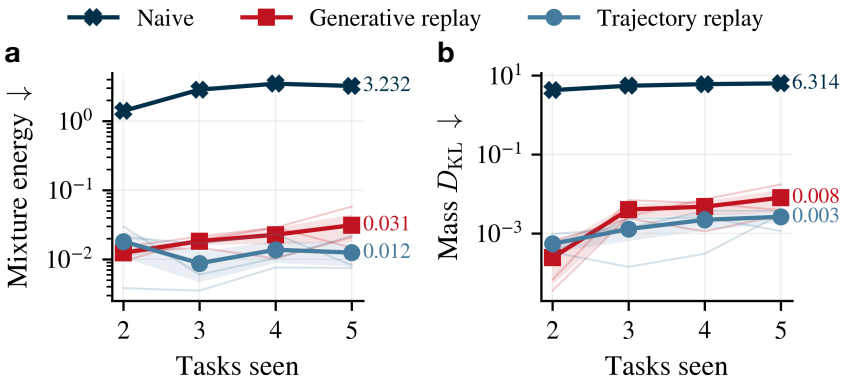

Figure 10: **Stream evolution on the five-task ring benchmark.** (**a**) Energy distance to the balanced mixture and (**b**) KL divergence of the generated task-mass composition from the uniform target, both as a function of the number of tasks seen, for naive fine-tuning, generative replay, and trajectory replay. Bold lines show the mean and faint lines three individual seeds. Numbers at the right give the final values, both axes are logarithmic, and lower is better.

D.3 Class-incremental image generation

D.3.1 Experimental details

Streams. Each dataset is split into disjoint class groups arriving sequentially, with class orders drawn per seed via the Avalanche package (Carta et al., 2023) so that the stream is identical across methods. We run seeds $\{42, 69, 1714\}$ (same as Masip et al. (2025)), each fixing the class order and training randomness. Split Fashion-MNIST and Split CIFAR-10 give five tasks of two classes each and share the same partition at a given seed. Split ImageNet-100 gives three tasks of $34/33/33$ classes. Within every task, a deterministic 10% of the training split is held out for validation, and the official test split is reserved for reporting.

Inputs. Fashion-MNIST is resized to 32×32 single-channel and CIFAR-10 to 32×32 RGB, both normalized to $[-1, 1]$, with train-time horizontal flips on CIFAR-10 only. ImageNet-100 is modeled in latent space: images are resized to 256×256 and encoded once with a frozen `stabilityai/sd-vae-ft-mse` autoencoder[1] into $32\times32\times4$ scaled latents, cached to disk, with train-time flips. Latents are decoded to pixels only for evaluation and sample grids.

Model. The velocity field is a Diffusers UNet2D (von Platen et al., 2022) on the 32×32 state, trained on the linear noise-to-data interpolant with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$. Fashion-MNIST and CIFAR-10 reuse the medium configuration of Masip et al. (2025), with block widths $(128, 256, 256, 256)$ and attention at one resolution (35.8M parameters), so our numbers are directly comparable to their diffusion baselines.

ImageNet-100 was not considered by Masip et al. (2025), so we selected its backbone and budget with a task-0 pilot crossing the medium configuration against a large latent-diffusion-style one, of widths $(128, 256, 512, 512)$ and attention at the two lowest resolutions (106.4M parameters), over optimizer budgets of $\{10, 20, 50, 100, 200\}$k steps, picking the lowest FID on the task-0 classes (Fig. 11). This selects the large backbone at 100k steps. The same pilot crossed conditional against unconditional variants, and we model ImageNet-100 *class-conditionally* since the unconditional runs were clearly worse at this scale, with label dropout 0.1 and classifier-free guidance (Zheng et al., 2023) 1.5 at sampling time.

Training. Every task is trained with Adam under the per-dataset budget in Tab. 2. Joint training reuses real data from all tasks seen so far: Fashion-MNIST and CIFAR-10 hold the epoch count fixed as the union grows, whereas ImageNet-100 holds the optimizer-step count fixed, so per-task compute does not grow with the stream.

[1] https://huggingface.co/stabilityai/sd-vae-ft-mse

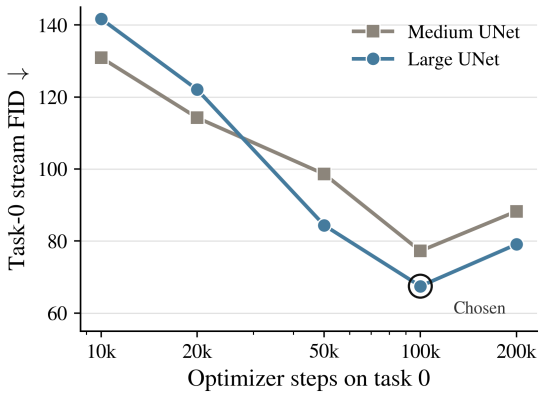

Figure 11: **Task-0 budget and backbone selection on ImageNet-100.** FID on the task-0 classes as a function of the number of optimizer steps on task 0, for two latent UNet backbones: the medium model with block widths $(128, 256, 256, 256)$ and attention at one resolution (35.8M parameters), and the large LDM-style model with widths $(128, 256, 512, 512)$ and attention at the two lowest resolutions (106.4M parameters). The circled point marks the configuration chosen for the continual-learning experiments; lower is better.

Replay. Both families share the stream and budget and differ only in what the historical term regresses onto.

- *Generative replay* draws a fraction λ of each batch from the frozen previous-task model and trains a single flow matching loss on the combined batch.
- *Trajectory replay* splits each batch evenly, samples one shared time per step, integrates the teacher only to that time, and matches the student field against the teacher field there, with λ weighting the historical and incoming terms.

On ImageNet-100, both condition the historical branch on labels cycled *uniformly* over all classes seen so far, and generate without guidance, unlike evaluation.

The equal-task policy sets $\lambda_k = k/(k+1)$, and the adaptive policy selects λ from the per-dataset grid by validation FID on 2,048 samples. The ODE backend integrates the teacher with Euler at the replay step size in Tab. 2. The MeanFlow backend replaces it with a distilled mean-velocity model using $N \in \{1, 4\}$ jumps, distilling task 0 post hoc at learning rate 10^{-4} with gradient clipping 1.0 for the per-dataset epochs in the table, then co-training the field and the mean velocity on later tasks with logit-normal time pairs, as in Geng et al. (2026a) (see Alg. 2).

Adaptive loss weighting. Task 0 distillation also uses adaptive loss weighting with the exponent in Tab. 2, but sequential updates do not: rescaling each branch by its own error would change their relative weight away from λ and $1 - \lambda$, and the fixed point of Prop. 3.3 would no longer be the mixture mean velocity.

Baselines. We report a series of reference points, all on the same stream, per-task budget, backbone, and evaluation protocol as our methods, following the scheme of Masip et al. (2025).

- *Joint training* retrains on the union of all real data seen so far, storing every past sample, and is therefore an upper target rather than a competing method.
- *Naive fine-tuning* continues training on the incoming task alone with no retention mechanism, giving the lower target. Since the baselines below use a different generative objective, we report it for both, as *diffusion naive fine-tuning* and *flow matching naive fine-tuning*, separating the gain attributable to replay from that attributable to the objective.
- *Generative replay (diffusion)* is the diffusion counterpart of generative replay above.
- *Generative distillation (diffusion)* (Masip et al., 2025) also replays from the frozen previous-task model, but supervises with the teacher's own noise prediction at a randomly drawn timestep rather than with the injected noise, so the student matches the teacher's denoising behavior along the

whole trajectory instead of only reproducing its final samples. This is the strongest published no-storage continuous-time baseline on these streams.

Evaluation. After the final task we generate 10,000 samples and report stream FID in Inception feature space (Szegedy et al., 2016) against pooled held-out images from all seen classes. Fashion-MNIST and CIFAR-10 are modeled unconditionally, so class composition is emergent and forgetting shows up as mass collapsing onto recent classes. Class-mass recovery (KLD) compares the classifier-predicted histogram against the target composition, using the pretrained convolutional neural network (CNN) classifiers released with the diffusion baselines (Masip et al., 2025), and isolates retention from the sample quality captured by FID. For ImageNet-100, labels are drawn from the target composition at sampling time, so the histogram matches by construction, and we report FID alone. All numbers are means with standard errors over seeds.

Table 2: Per-dataset configuration for the class-incremental image experiments. Sampler and replay steps are Euler step sizes, with the resulting number of function evaluations in parentheses.

	Fashion-MNIST	CIFAR-10	ImageNet-100
Data			
State	$1\times32\times32$	$3\times32\times32$	$4\times32\times32$ latent
Tasks $\times$ classes	5×2	5×2	$3 \times (34/33/33)$
Horizontal flips	no	yes	yes
Model			
UNet	medium	medium	large
Parameters	35.8M	35.8M	106.4M
Class-conditional	no	no	yes (CFG 1.5)
Optimization			
Epochs / task	100	200	667
Batch size	128	64	256
Learning rate	2×10^{-4}	2×10^{-4}	1×10^{-4}
Gradient clip	none	none	1.0
Joint schedule	fixed epochs	fixed epochs	fixed steps
Replay and sampling			
Sampler step (NFE)	0.1 (10)	0.05 (20)	0.02 (50)
Replay step (NFE)	0.1 (10)	0.05 (20)	0.033 (30)
λ grid	$\{0.1, 0.25, 0.5, 0.75, 0.9\}$	$\{0.25, 0.5, 0.75\}$	$\{0.25, 0.5, 0.75\}$
MeanFlow (μ, σ) (Geng et al., 2026a)	$(2.0, 2.0)$	$(2.0, 2.0)$	$(0.4, 1.0)$
MeanFlow diagonal rate (Geng et al., 2026a)	0.25	0.25	0.75
MeanFlow adaptive loss weight (T_0 only) (Geng et al., 2026a)	0.75	0.75	1.0
MeanFlow distillation epochs	100	200	150
Evaluation			
Validation samples	2,048	2,048	2,048
Test samples	10,000	10,000	10,000
Reported metrics	FID, KLD	FID, KLD	FID

D.3.2 Additional results

We report wall-time gains, degradation across the stream, the weights the adaptive policy selects, the quality-budget trade-off, when the MeanFlow backend beats the ODE teacher, and qualitative samples.

Runtime comparison. Fig. 12 compares the two replay families over the full CL pipeline under equal-task λ. Trajectory replay is faster on every stream, by $1.22\times$ on Fashion-MNIST, $1.51\times$ on CIFAR-10, and $1.65\times$ on ImageNet-100. The margin grows with the cost of the teacher: integrating to an intermediate time saves a fraction of the trajectory fixed by $\mathbb{E}[t]$, so the absolute saving scales with how expensive a full generation is, from 0.2h on Fashion-MNIST to 25.5h on ImageNet-100. The ratio is capped at a factor of two in Sec. A.13, but the hours saved keep growing.

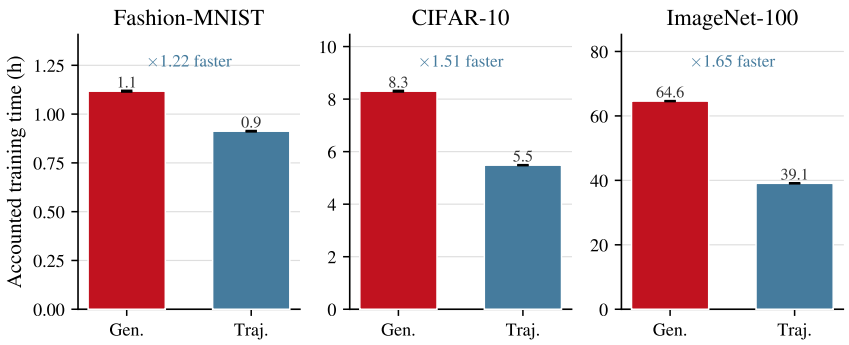

Figure 12: **Training wall-time comparison**. Full-CL-pipeline comparison of trajectory and generative replay in terms of accounted wall time across datasets. The two methods are compared in the equal-task λ setting.

Stream evolution. Fig. 13 tracks both FID and KLD metrics across the stream rather than only at the final task. Naive fine-tuning worsens monotonically everywhere. The replay families diverge in how they respond to the weighting policy. Generative replay under equal-task weighting degrades with each update, ending far above its adaptive counterpart on Fashion-MNIST and CIFAR-10. Trajectory replay is nearly flat under both. It therefore reaches comparable quality with the parameter-free schedule, which matters where a validation sweep at every update is the dominant cost.

Fig. 14 shows the resulting compositions: naive fine-tuning retains almost no mass on early classes, and trajectory replay is closest to the target histogram on both datasets, though its KLD remains higher than joint training (Tab. 1) and every method still favors recent classes. Unbiased retention across a long stream is not solved by reweighting alone and remains for *future work*.

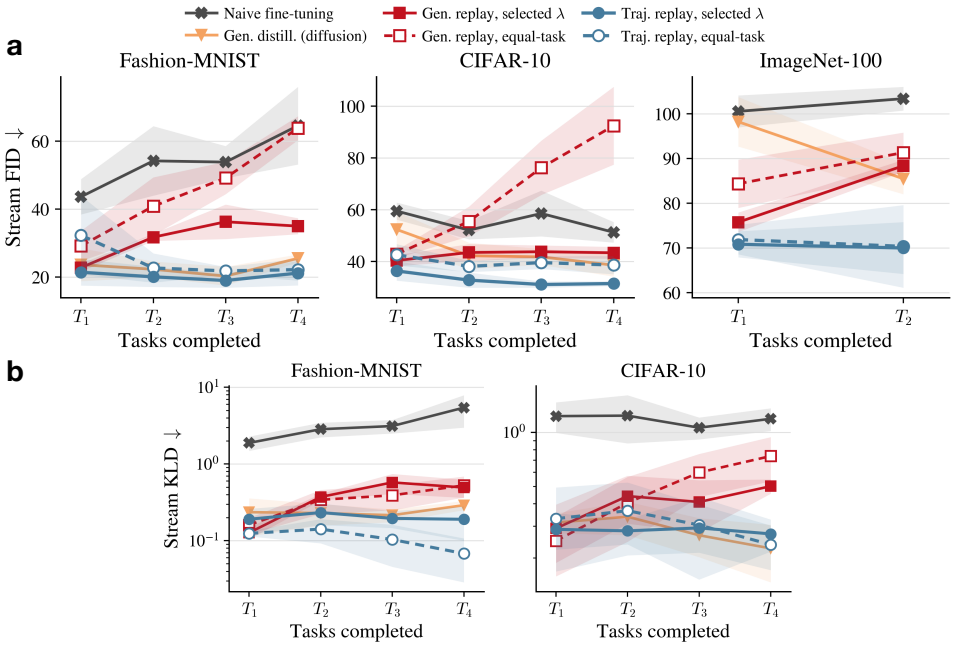

Figure 13: **Class-incremental image streams.** (**a**) Stream FID and (**b**) stream KLD as a function of tasks completed, for naive fine-tuning, diffusion distillation, and generative and trajectory replay under selected and equal-task λ. FID is shown for Fashion-MNIST, CIFAR-10, and ImageNet-100. KLD for Fashion-MNIST and CIFAR-10. Lines show the mean and shaded bands the variability across three seeds. Filled markers with solid lines denote selected λ and open markers with dashed lines equal-task λ, and lower is better on both metrics.

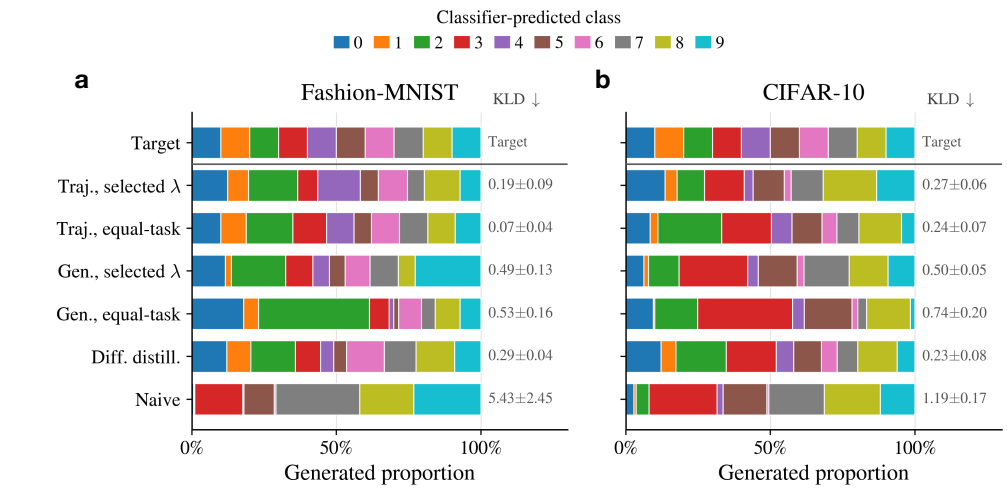

Figure 14: **Class composition of generated samples after the final task.** Stacked bars give the proportion of generated samples assigned to each class by the evaluation classifier, for (**a**) Fashion-MNIST and (**b**) CIFAR-10. Rows list the uniform target and each method: trajectory and generative replay under selected and equal-task λ, diffusion distillation, and naive fine-tuning. Numbers at the right report stream KLD from the target composition (mean $\pm$ SE over three seeds). Lower is better.

Selected replay weights. Fig. 15 reports which weights the adaptive λ-selection policy chooses. The criterion scores against a balanced reference, so the uniform composition that equal-task produces is what it rewards. On Fashion-MNIST and CIFAR-10, trajectory replay tracks that rule, rising with the number of tasks seen, whereas generative replay settles well below it and non-monotonically. Since λ sets how much mass the update assigns to history, the criterion appears to discount generative replay's own samples, whose fidelity is low enough that weighting them as heavily as real tasks costs more than the retention it buys. Trajectory replay faces no such trade-off, as its historical term regresses onto the frozen field rather than onto generated data. The ImageNet-100 panel runs the other way, though with only two updates and wide error bars.

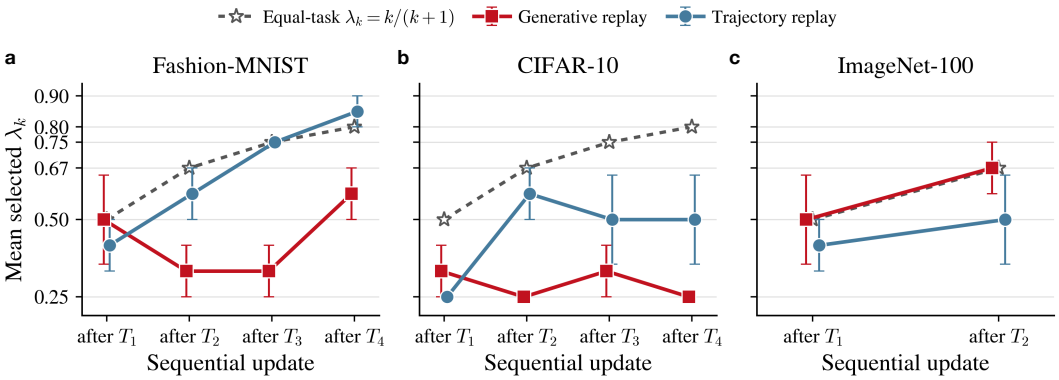

Figure 15: **Selected replay weights across the image streams.** Mean replay weight λ_k chosen at each sequential update for generative and trajectory replay on (**a**) Fashion-MNIST, (**b**) CIFAR-10, and (**c**) ImageNet-100, with the equal-task rule $\lambda_k = k/(k+1)$ shown as a dashed reference. Error bars show variability across three seeds.

Does the MeanFlow backend cost quality? Fig. 16 asks what replacing the ODE teacher with a distilled mean velocity gives up and what it saves. Panel a normalizes final FID by the matched ODE run at each backend's replay-field evaluations per update. Trajectory replay sits at parity across the range, so MF-1 and MF-4 *lose essentially nothing* relative to full integration and one or two evaluations suffice. Generative replay instead starts well above parity at the smallest budget and

only recovers as evaluations accumulate, as expected: a replay sample must be integrated to $t = 1$ before it can serve as data, whereas trajectory replay supervises at an intermediate time where a coarse trajectory is still informative. Panel b prices the same endpoint in accounted training time: the wall-clock cost of an update including replay sampling and, for the MeanFlow backends, the amortized cost of distilling the teacher. The gain is dataset-dependent: on Fashion-MNIST and CIFAR-10 the MeanFlow backends save little because the ODE teacher is already cheap, whereas on ImageNet-100 they reach comparable FID at a fraction of the time.

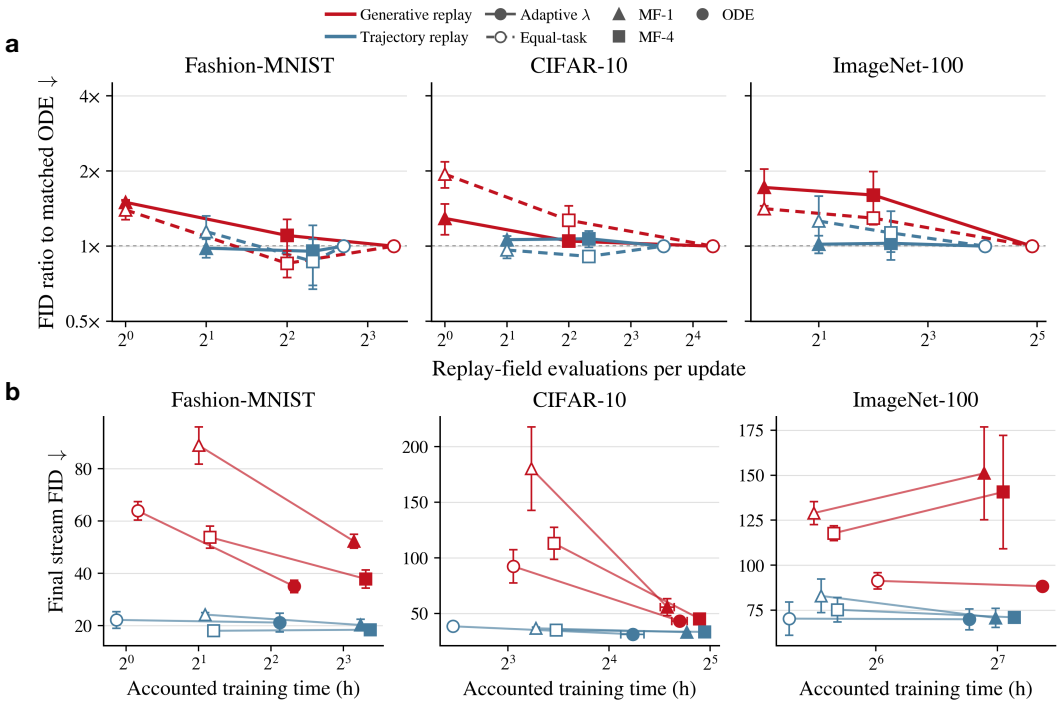

Figure 16: **Replay backend cost and quality on the image streams.** (**a**) Final stream FID relative to the matched ODE run as a function of replay-field evaluations per update, and (**b**) final stream FID against accounted training time, for Fashion-MNIST, CIFAR-10, and ImageNet-100. Colors distinguish generative from trajectory replay, marker shapes distinguish the MF-1, MF-4, and ODE backends, and filled markers with solid lines denote adaptive λ while open markers with dashed lines denote equal-task λ. The dotted line in (**a**) marks parity with the matched ODE run. Error bars show variability across three seeds, x-axes are logarithmic, and lower is better.

Parameter-efficient and regularization baselines. Tab. 3 reports LoRA and gradual fine-tuning under the same protocol. Neither is competitive: the best configuration reaches 65.1, 50.0, and 104.6 FID against 18.0, 31.4, and 70.0 for trajectory replay, with class-mass divergences up to two orders of magnitude larger. Capacity does not help, as raising the LoRA rank from 8 to 32 moves FID by at most 5.2 points, and the two β schedules are indistinguishable. Both families bound how far the velocity field may move but supply no target for the historical tasks, mirroring the transcriptomics streams (Tabs. 6 and 7).

When the mean velocity pays off. Fig. 17 sweeps the number of Euler integration steps used by the ODE teacher on Fashion-MNIST, from 5 to 100. Final FID is flat beyond roughly ten steps (panel a), so a coarse teacher loses nothing here, while the teacher's amortized cost per update grows with the step count and the MeanFlow teachers cost a *fixed amount set by co-training* rather than by the number of replay steps (panel b).

The two therefore cross at a moderate teacher budget: on Fashion-MNIST, where ten steps already saturate quality, the crossing sits above the operating regime, and the ODE backend is cheaper, whereas on ImageNet-100 (Fig. 16b) the teacher is expensive enough that the MeanFlow backends

Table 3: Memory-free baselines for class-incremental image generation. Final-stream means with standard errors over three seeds after five tasks (Fashion-MNIST and CIFAR-10) or three tasks (ImageNet-100). LoRA uses $\alpha = 2r$; GFT decays the teacher-regularization weight β from 1 to 0. Blue rows show trajectory-replay references (adaptive or equal-task λ; MF-N uses N replay steps). Lower is better.

Method	Fashion-MNIST		CIFAR-10		ImageNet-100
	↓ FID	↓ KLD	↓ FID	↓ KLD	↓ FID
LoRA (r=8, α=16)	68.4 (±11.1)	4.508 (±1.956)	50.9 (±3.4)	1.006 (±0.114)	109.8 (±0.5)
LoRA (r=16, α=32)	67.3 (±10.9)	5.129 (±2.154)	50.2 (±3.2)	1.055 (±0.123)	107.6 (±0.9)
LoRA (r=32, α=64)	66.2 (±11.2)	5.107 (±2.146)	50.0 (±3.0)	1.069 (±0.115)	104.6 (±2.3)
GFT (sigmoid β schedule)	65.1 (±12.0)	4.264 (±1.309)	50.7 (±4.0)	1.138 (±0.171)	119.5 (±2.7)
GFT (linear β schedule)	65.8 (±12.8)	4.873 (±2.188)	51.0 (±4.3)	1.101 (±0.176)	119.1 (±3.4)
Trajectory replay (ODE, adaptive λ)	21.1 (±3.6)	0.190 (±0.090)	31.4 (±0.9)	0.272 (±0.055)	70.0 (±5.8)
Trajectory replay (ODE, equal-task λ)	22.1 (±3.2)	0.068 (±0.039)	38.6 (±3.5)	0.237 (±0.066)	70.4 (±9.2)
Trajectory replay (MF-1, adaptive λ)	20.2 (±2.2)	0.140 (±0.008)	33.3 (±0.5)	0.332 (±0.066)	70.7 (±5.3)
Trajectory replay (MF-1, equal-task λ)	24.2 (±0.8)	0.065 (±0.027)	36.8 (±1.2)	0.237 (±0.089)	83.0 (±9.3)
Trajectory replay (MF-4, adaptive λ)	18.4 (±1.5)	0.120 (±0.023)	33.5 (±1.7)	0.349 (±0.056)	71.1 (±2.7)
Trajectory replay (MF-4, equal-task λ)	18.0 (±1.7)	0.090 (±0.030)	35.1 (±3.1)	0.211 (±0.023)	75.3 (±6.8)

match FID at a fraction of the cost. The mean velocity is thus not uniformly cheaper: it removes the dependence of replay cost on teacher fidelity, and pays off where that fidelity is expensive.

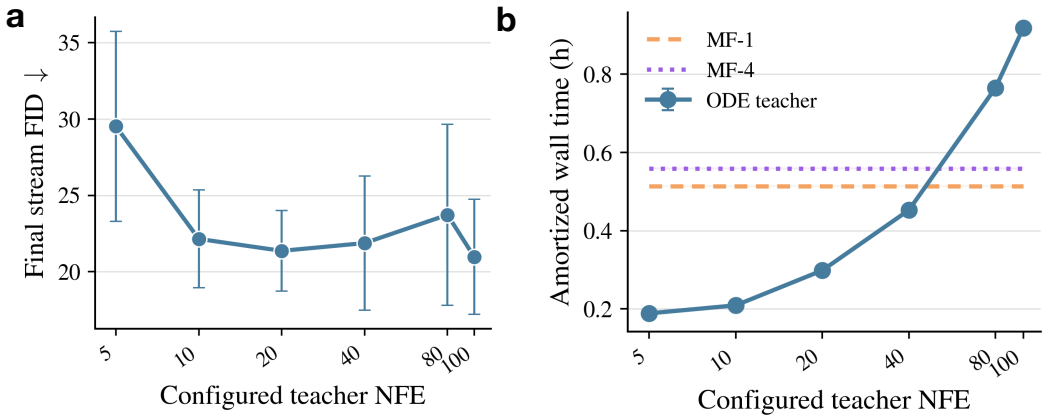

Figure 17: **Teacher NFE sweep on Fashion-MNIST.** (**a**) Final stream FID and (**b**) amortized wall time per update as a function of the configured teacher NFE for the ODE teacher. Dashed and dotted horizontal lines in (**b**) mark the corresponding cost of the MF-1 and MF-4 teachers. Error bars show variability across three seeds and the x-axis is logarithmic.

Qualitative samples. Figs. 18 to 20 show held-out and generated samples for all three streams, where trajectory replay retains recognizable class structure on early tasks that generative replay renders with visible degradation, especially on ImageNet-100.

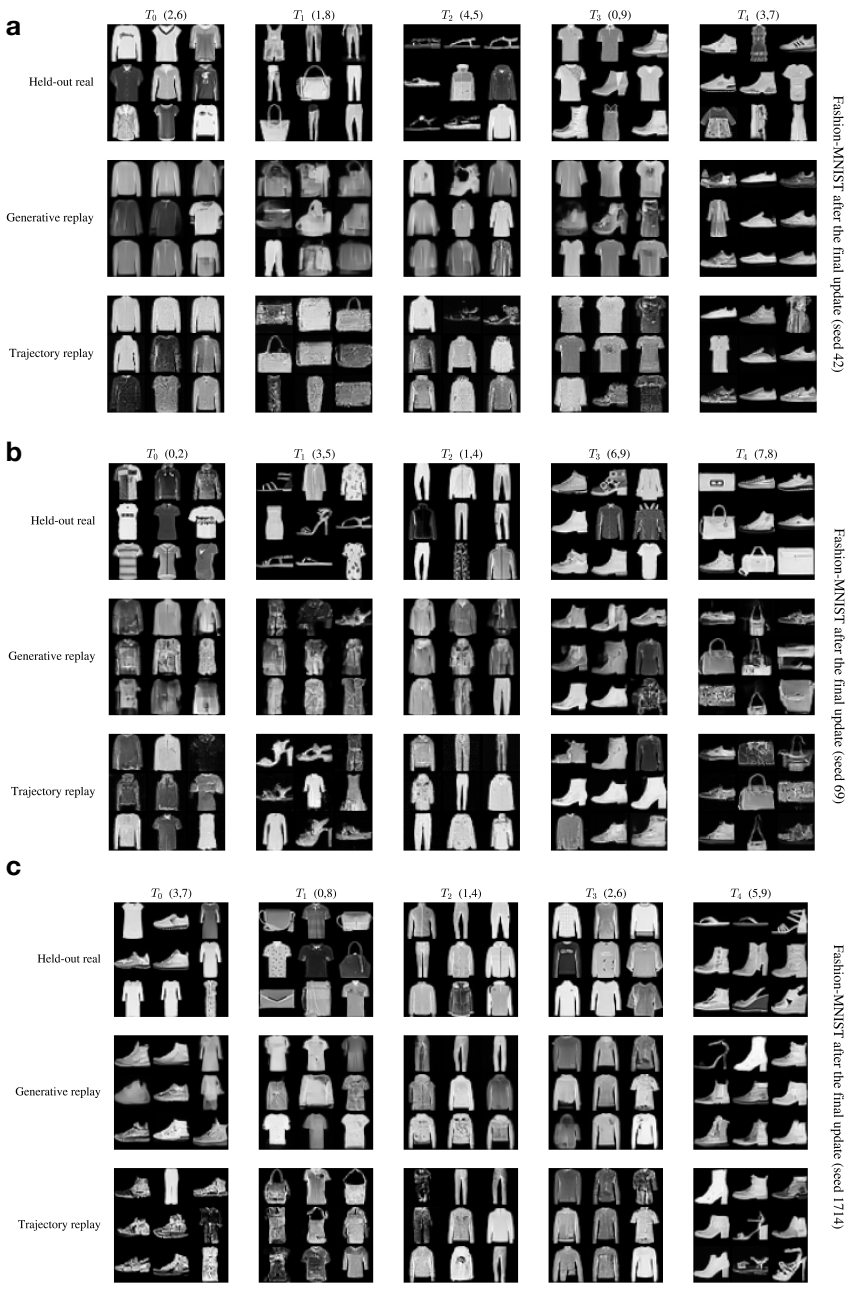

Figure 18: **Fashion-MNIST samples after the final update.** Held-out real images and samples from generative and trajectory replay, shown for seeds (**a**) 42, (**b**) 69, and (**c**) 1714. Columns correspond to the tasks of that seed's class stream, with the task index and its two classes given above each block. Rows within a block share the same task.

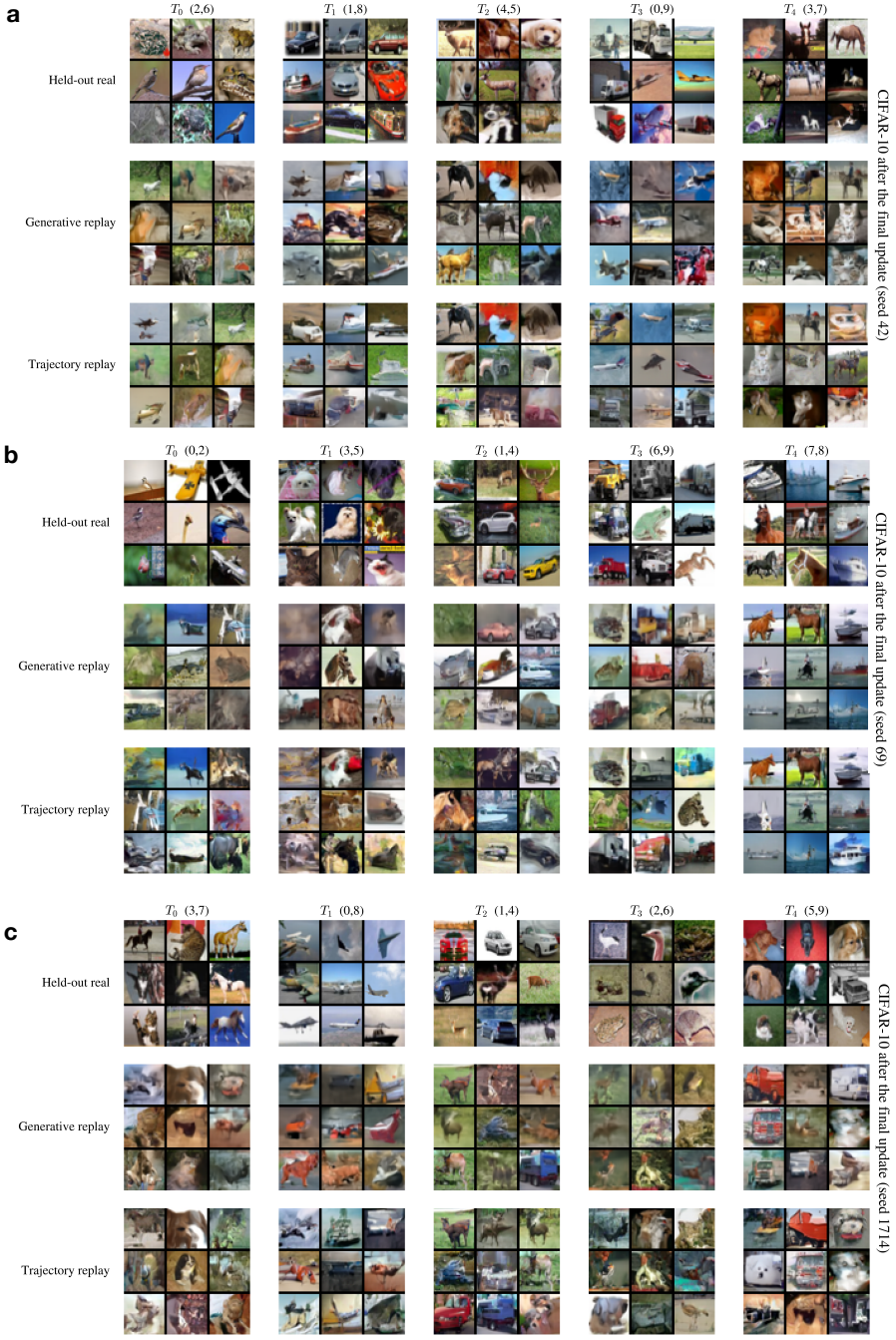

Figure 19: **CIFAR-10 samples after the final update.** Held-out real images and samples from generative and trajectory replay, shown for seeds (**a**) 42, (**b**) 69, and (**c**) 1714. Columns correspond to the tasks of that seed's class stream, with the task index and its two classes given above each block. Rows within a block share the same task.

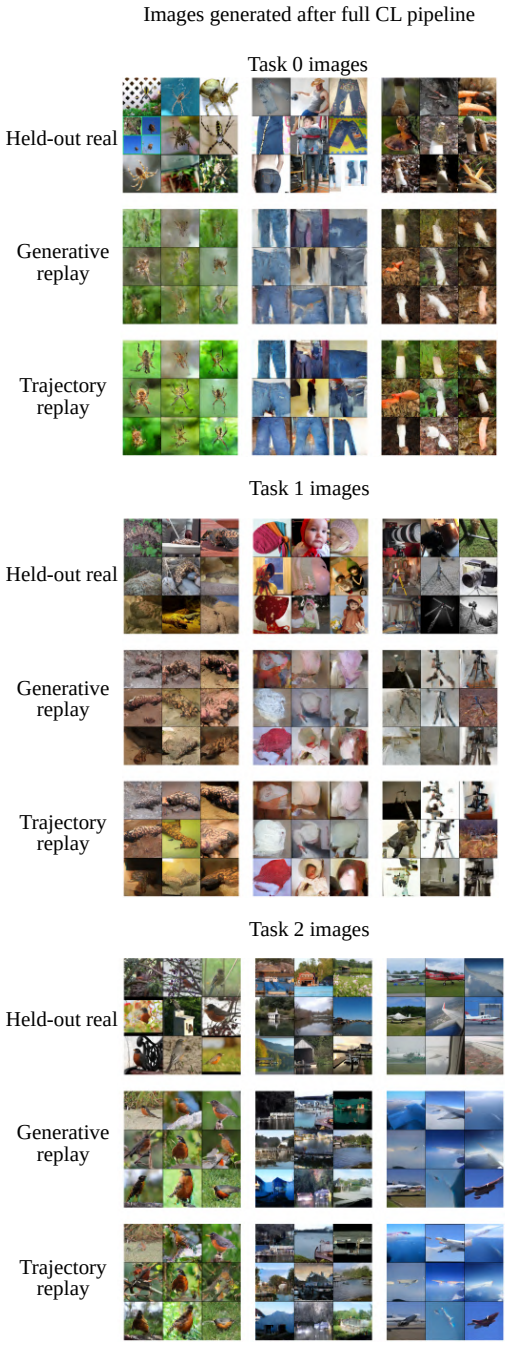

Figure 20: **ImageNet-100 samples after the final update.** Held-out real images and samples from generative and trajectory replay, grouped by the three tasks of the stream. Within each group, rows share the same source and columns show three classes drawn from that task.

D.4 SYNTHETIC PERTURBATION SHIFT GENERALIZATION

D.4.1 EXPERIMENTAL DETAILS

Task. Data arrive as pairs (c, y) of a context c and a perturbation y, and a generative model approximates $p(\boldsymbol{x} \mid c, y)$, mirroring Sec. 4.4 in a controlled setting where the ground truth is known. Contexts shift the Gaussian mean, and perturbations add a fixed offset shared across contexts, so perturbation effects are transferable by construction. The update introduces a new context but withholds one perturbation within it, leaving a single combination $(c^\star, y^\star)$ never observed in either phase. Recovering it requires (**I**) retaining the perturbation direction learned on historical contexts and (**II**) fitting the basal position of the new one, so held-out error reads out the balance between the two rather than either alone.

Data. We build a 6×4 grid of conditions in $\mathbb{R}^2$: six contexts c placed on a hexagon of radius 4, and four perturbations y applied as an additive offset of scale 1.5 shared across contexts, with $y = 0$ the unperturbed *control*. Each cell (c, y) is an isotropic Gaussian with $\sigma = 0.25$, providing 800 training and 200 validation points, and the single held-out cell provides $1{,}000$ test points, with data seed 0.

Stream. Pretraining uses contexts 0–4 with all four perturbations. The update introduces context 5 with perturbations 0–2 only, so the held-out combination is $(c^\star, y^\star) = (5, 3)$.

Model. The velocity field is a conditional MLP of width 256 and depth 4, with a 64-dimensional sinusoidal time embedding and 32-dimensional learned embeddings for context and perturbation, trained on the linear noise-to-data interpolant with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$. Sampling uses Euler at step size 0.05 (20 evaluations).

Pretraining. The context-0–4 model trains for $1{,}000$ epochs with batch size 1024 and Adam at 3×10^{-4}, with one checkpoint per seed.

Update. Each update trains for $2{,}000$ epochs with the same optimizer settings, for seeds $\{10, 20, 30, 40, 50\}$ at every λ in the grid, each loading the pretraining checkpoint of its own seed. We compare naive fine-tuning, which continues training the historical checkpoint on incoming data alone with no replay term, against generative and trajectory replay, sweeping $\lambda \in \{0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95\}$. Replay conditions are drawn from the pretraining support, so no data from the held-out combination enters training at any point.

Evaluation. After the update, we generate 500 samples conditioned on $(c^\star, y^\star)$ and report energy distance to that cell's held-out test set, with the naive fine-tuning value shown as a horizontal reference. Lower is better.

D.4.2 ADDITIONAL RESULTS

Fig. 21a shows the layout: contexts separate cleanly, and the shared perturbation offsets give each context the same internal geometry, so the held-out combination has a well-defined target location.

Fig. 21b traces held-out energy distance as a function of λ. Both replay methods trace a U, with a minimum near $\lambda = 0.5$ that is more than an order of magnitude below naive fine-tuning, which has no mechanism for retaining the perturbation direction. The shape is the point: held-out error rises at both extremes, so the unseen combination is recovered neither by staying close to the historical model nor by fitting the incoming context alone, but only where the update balances the two. The two replay methods are indistinguishable at the optimum and separate only at high λ, where trajectory replay degrades faster, since its historical term regresses onto a frozen field that has never seen the new context.

Fig. 21c shows the balance directly: the generated population coincides with the held-out target at $\lambda = 0.5$ and is displaced from it at either extreme. Recovering an unseen combination therefore requires fitting the incoming context without discarding the transferable structure carried from the historical ones.

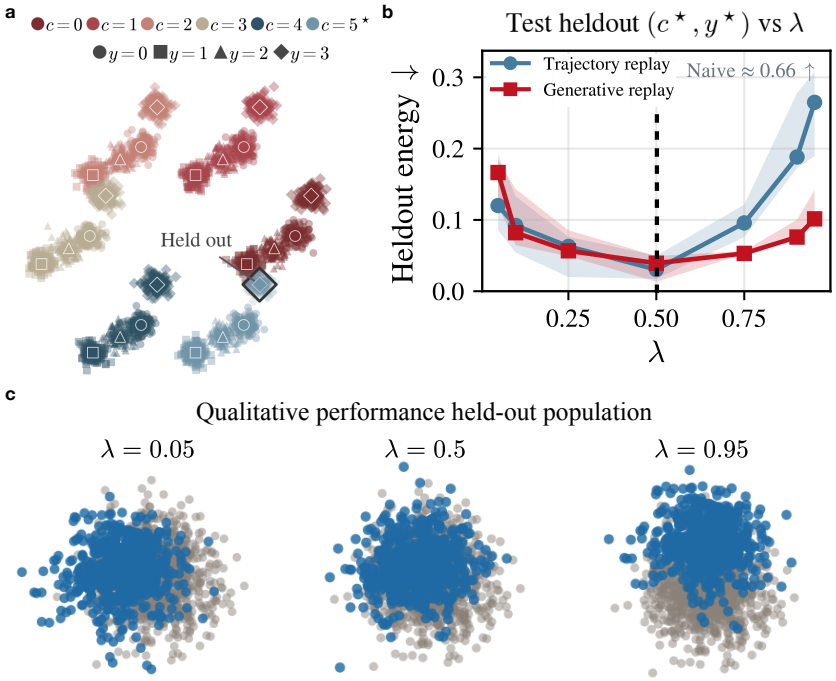

Figure 21: **Synthetic perturbation-shift generalization.** (**a**) Layout of the synthetic data, where color encodes the context c and marker shape the perturbation y. The held-out combination $(c^\star, y^\star)$ is marked separately. (**b**) Energy distance on the held-out combination as a function of the replay weight λ for generative and trajectory replay, with the dashed vertical line at $\lambda = 0.5$ and the naive fine-tuning value indicated at the top right. Lines show the mean and shaded bands the variability across seeds, and lower is better. (**c**) Samples generated by trajectory replay for the held-out combination (blue) against the observed held-out cells (grey), at $\lambda \in \{0.05, 0.5, 0.95\}$. The two clouds coincide at $\lambda = 0.5$, whereas both extremes displace the generated population away from its target, toward the incoming context at low λ and toward the historical one at high λ.

D.5 Application: perturbation modeling in single-cell transcriptomics

D.5.1 Experimental details

Datasets. We use two perturbation datasets whose context variable is a biological or technical batch rather than a semantic class. Throughout, a *stratum* is one context-by-perturbation combination, the unit at which conditional distributions are modeled and at which every metric below is computed. Exact values for every setting are in Tab. 4.

- *Replogle-Nadig* (Replogle et al., 2022; Nadig et al., 2025). Genome-scale CRISPRi screens in which the context is the cell line (Jurkat, K562, RPE1, HepG2) and the perturbation is the targeted gene, 2,023 of them plus a non-targeting control.
- *PBMC cytokine screen* (Oesinghaus et al., 2025). Cytokine stimulations in which the context is the donor, twelve in total, and the perturbation is the applied cytokine, 90 of them plus a PBS (phosphate-buffered saline) control. Cells additionally carry one of 18 annotated cell types, used as a covariate rather than as a context.
- *Preprocessing.* Following Adduri et al. (2026), both are restricted to the 2,000 highly variable genes, library-size normalized to 10^4 counts and $\log(1+x)$ transformed, with no further scaling.
- *Splits.* Split labels follow the few-shot definitions of Adduri et al. (2026): cells in non-few-shot contexts are training data, while in few-shot contexts the perturbations on that context's test list become held-out test cells and the remainder fine-tuning data. Controls stay trainable everywhere.
- *Latent space.* To avoid leakage, flow matching operates on a zero-centered PCA basis fit only on task-0 training cells, so later contexts never influence the coordinate system. We use 50 components on Replogle and 100 on PBMC, retaining 34.2% and 58.4% of the variance (see Tab. 5).
- *Decoding.* Latents return to gene space through a per-task decoder D_k composing the tied task-0 PCA inverse with a per-gene affine calibration $(a_{k,g}, b_{k,g})$ fit by scalar least squares on that task's training cells, so that $D_k(\boldsymbol{z}) = \mathbf{A}_k(\mathbf{W}^\top \boldsymbol{z} + \boldsymbol{\mu}) + \boldsymbol{b}_k$ with $\mathbf{A}_k$ diagonal. A single tied inverse cannot absorb the per-task shift in gene scale, which the calibration corrects, but being diagonal, it cannot recover variance discarded by the truncation to d components: it fixes gene scale, not the subspace. The bank is fit once per dataset, frozen, and shared by every method and seed, so no part of the reported gap can come from a better decoder. A qualitative reconstruction display on the pooled later-task datasets is in Fig. 26.

Streams. Replogle is a two-task stream: task 0 trains jointly on Jurkat, K562, and RPE1, and task 1 introduces HepG2. PBMC is a five-task stream over eight pretraining donors followed by Donor1, Donor4, Donor9, and Donor12. The perturbation vocabulary is shared across contexts, so each task adds a context row rather than a new label space. The model conditions on context and perturbation embeddings, plus a cell-type covariate on PBMC. Task 0 is trained once per seed and shared by every method.

Model and training. The velocity field is an MLP over PCA latents with 128-dimensional sinusoidal time and condition embeddings concatenated to the input, trained on the linear noise-to-data interpolant with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$. Replogle uses a flat SiLU MLP of width 2048 and depth 4, and PBMC uses a pre-normalized residual MLP of width 2048 and depth 6. Both were fixed by the two-stage selection study in Tab. 5.

- *PCA width* (panel a), from single-seed pilots at a fixed 1024×4 backbone. The fit falls monotonically with width on both datasets, giving 50 components on Replogle, while on PBMC 50 and 100 are within a thousandth of each other, and we take 100 following Klein et al. (2025).
- *Backbone* (panel b), sweeping width, depth, and residual connections under the final sequential protocol and selecting on task-0 held-out Δ. The flat MLP wins on Replogle and the smaller residual model on PBMC.

Both selection studies use seed 42 alone, and no method is tuned against them afterwards. Every method uses Adam at 10^{-4} with batch size 1024, under the budgets in Tab. 4, for seeds $\{42, 69, 1714\}$.

Replay. Replay follows the same formulation as in the image experiments, with the teacher integrated by Euler at step size 0.05 and equal-task weighting setting $\lambda_k = k/(k+1)$. Two things differ

here. Replay conditions are *sampled proportionally* to the stratum frequencies of the old training pool, since the conditioning space is a context-by-perturbation grid rather than a class label. The adaptive policy selects from $\lambda \in \{0.25, 0.5, 0.75\}$ by a balanced Pearson Δ, the unweighted mean of the old-task and new-task stratum macros, scored on the training strata of every task seen so far with $K = 200$ generated cells per stratum. The MeanFlow backends distill the task-0 teacher with logit-normal time sampling ($\mu = 0.4$, $\sigma = 1.0$) and diagonal fraction 0.25, under the same step budget as task-0 training itself (Tab. 4), then replay with $N \in \{1, 4\}$ jumps.

Table 4: Configuration for the continual transcriptomics experiments.

Setting	Replogle-Nadig	PBMC
Data		
Cells	643,413	9,697,974
Genes (highly variable)	2,000	2,000
Contexts	4 cell lines	12 donors
Perturbations	2,023 genes	90 cytokines
Control	non-targeting	PBS
Covariate conditioning	none	cell type (18)
Normalization	10^4 library size, $\log(1+x)$	
PCA width (task-0 basis)	50	100
Variance retained	0.342	0.584
Stream		
Tasks	2	5
Task 0	Jurkat, K562, RPE1	8 donors
Sequential tasks	HepG2	Donor1, 4, 9, 12
Model		
Backbone	MLP	residual MLP
Width / depth	2048 / 4	2048 / 6
Time embedding	128	128
Condition embedding	128	128
Training		
Optimizer	Adam, 10^{-4}	Adam, 10^{-4}
Batch size	1024	1024
Gradient clip / EMA	none	none
Steps, task 0	100,000	300,000
Steps, later tasks	50,000	150,000
Seeds	42, 69, 1714	
Replay		
Teacher sampler	Euler, step 0.05 (20 evaluations)	
Condition sampling	proportional to old stratum frequency	
Equal-task λ	$1/2$	$1/2, 2/3, 3/4, 4/5$
Adaptive λ grid	$\{0.25, 0.5, 0.75\}$	
Selection metric	old/new balanced Pearson Δ, $K = 200$	
Selection data	seen-task training strata	
MeanFlow jumps	$N \in \{1, 4\}$	
MeanFlow (μ, σ)	$(0.4, 1.0)$, logit-normal	
MeanFlow diagonal rate	0.25	
MeanFlow adaptive loss weight (T_0 only)	0.75	
MeanFlow optimizer	AdamW, 10^{-4}, $\beta_2 = 0.95$, clip 1.0	
MeanFlow distillation budget	same as task-0 training	
Evaluation		
Generated cells per stratum	500	500
Control rows	decoded observed control latents	
Tracks	fit, retention, stream, heldout	
Metrics	Pearson Δ, DE Overlap@N, residual Δ, L1 discr.	

Baselines. All baselines share the stream, backbone, budget, decoder bank, and evaluation protocol, differing only in what is updated or replayed.

- *Naive fine-tuning* trains on the incoming context alone, sweeping three independent freeze switches: whether the velocity field trains, whether perturbation rows seen earlier train, and whether earlier context rows train. Rows first appearing on the new task always train.

- *LoRA* adapts the MLP with rank $r \in \{4, 8, 16\}$ and $\alpha = 2r$, leaving the output layer frozen and merging the adapter once the task finishes.
- *Gradual fine-tuning* self-distills from the frozen previous model with a distillation weight annealed from 1 to 0, under sigmoid and linear schedules.
- *Two-stage fine-tuning* spends the first 10% or 50% of the task training only on the new context embedding row, then unfreezes the full field.
- *Real replay* mixes stored real cells from earlier tasks into each batch, a privileged reference rather than a competing method because it retains raw data.

Generative and trajectory replay are run with all three teacher backends and both λ policies.

Evaluation. All numbers come from a single evaluation of the final checkpoint, reported on four tracks.

- *Retention*, the macro over the task-0 training strata: Jurkat, K562 and RPE1 on Replogle, the eight pretraining donors on PBMC.
- *Fit*, the unweighted mean of the per-context macros for every context introduced sequentially thereafter: HepG2 on Replogle, and Donor1, Donor4, Donor9 and Donor12 on PBMC.
- *Stream*, the union of the two above (reported, for example, in Fig. 23).
- *Heldout*, the test perturbations of every context seen so far.

We generate $K = 500$ cells per non-control stratum and take the basal from decoded observed control latents rather than from generated controls, so only the treated side is synthesized. Following the *CellEval* metric suite in Adduri et al. (2026), the headline metric is Pearson Δ, the correlation between predicted and observed perturbation-minus-control profiles, macro-averaged over strata. We additionally report DE Overlap@N, the fraction of the N observed significant differentially expressed genes recovered among the top-N predicted, with N set per perturbation. Results are means with standard errors over seeds.

D.5.2 Additional results

Velocity model selection and full results. Tab. 5 reports the PCA dimension and backbone sweeps, both run on task 0 alone to avoid leakage. Fit falls monotonically with PCA width on both datasets (panel a), since wider bases spend capacity on directions that carry little perturbation signal. On Replogle, the 50-component basis is clearly best, whereas on PBMC 50 and 100 are within a thousandth of each other, and we take 100 following Klein et al. (2025). The backbone sweep (panel b) separates the two datasets: on Replogle, the flat MLP wins on fit and task-0 held-out with retention tied within a thousandth, while on PBMC the residual backbone is ahead on both retention and held-out, and among residual configurations the smaller 2048×6 model gives up little on retention while scoring higher on held-out, so we keep it.

Tabs. 6 and 7 give every configuration behind Fig. 4. On the single Replogle update, the replay families are close, with trajectory replay retaining slightly more than generative replay (0.604–0.620 against 0.573–0.594 Pearson Δ) and heldout tied at 0.39–0.41. Over the four sequential updates of PBMC, the gap widens on every axis, to 0.430–0.451 retention against 0.306–0.403 and 0.332–0.350 held-out against 0.289–0.330, placing trajectory replay level with real replay on fit and held-out despite storing no cells. The MeanFlow backends leave it unchanged on both streams but cost generative replay substantially on PBMC, where MF-1 retention drops to 0.306.

The freeze grid explains why a memory-free non-replay baseline can still top the retention column. Retention depends almost entirely on the velocity field, and adaptation almost entirely on it too, so the three switches are not three comparable knobs.

- *Field frozen, old embeddings frozen* ($\mathrm{vf}{=}0$, $\mathrm{pert}{=}0$, $\mathrm{ctx}{=}0$). Only the new context row is learned, so task 0 is preserved almost exactly, and PBMC retention is the highest of any memory-free method at 0.491. A single embedding vector cannot express the new context's perturbation responses, so fit stalls at 0.233 against 0.515 for trajectory replay.

- *Field frozen, old embeddings trainable* (vf=0, pert=1, ctx=1). The old rows drift to compensate for a field that cannot move, costing retention (0.400) for no fit at all (0.232) on PBMC. On Replogle, the same relaxation does buy fit, 0.359 against 0.193, at a smaller retention cost.
- *Field trainable* (vf=1, any embedding setting). Fit recovers to 0.289, and retention collapses to 0.056–0.064, and the embedding switches change neither, confirming that the field rather than the conditioning rows carries the historical task.

Freezing therefore buys retention only by refusing to adapt, which is why the two axes have to be read jointly rather than ranked one column at a time.

Table 5: **Transcriptomics architecture selection.** (**a**) PCA width sweep with a fixed MLP 1024×4 backbone, reporting fit Pearson Δ from single-seed pilots. (**b**) Backbone width and depth under the final sequential protocol with 500 generated cells per non-control stratum, reporting fit, retention, and task-0 heldout Pearson Δ. All rows use seed 42, and blue rows mark the selected configuration.

Dataset	PCA width	↑ Pilot fit Δ
(a) PCA width sweep		
Replogle	50	**0.486**
Replogle	100	0.458
Replogle	256	0.423
Replogle	512	0.389
PBMC	50	0.353
PBMC	100	0.352
PBMC	256	0.282
PBMC	512	0.263

Dataset	Backbone	Width	Depth	↑ Fit Δ	↑ Retention Δ	↑ T_0 heldout Δ
(b) Final-protocol model-capacity check ($K = 500$)						
Replogle	MLP	2048	4	**0.611**	0.608	**0.410**
Replogle	Residual MLP	2048	6	0.599	**0.609**	0.392
PBMC	MLP	4096	4	**0.546**	0.441	0.297
PBMC	MLP	3072	4	0.528	0.443	0.308
PBMC	Residual MLP	3072	6	0.532	**0.455**	0.352
PBMC	Residual MLP	2048	6	0.514	0.445	**0.355**

Table 6: **Continual perturbation-response modeling on Replogle-Nadig.** Final-checkpoint Pearson Δ and DE Overlap@N after adaptation from three task-0 cell lines to HepG2, reported as fit on the later task, retention on task 0, and heldout on perturbations excluded from training. Means with standard errors over seeds 42, 69, and 1714. Adaptive λ is selected on validation strata and equal-task uses $\lambda_k = k/(k+1)$. Gray marks the privileged real-replay reference and blue marks trajectory replay. Higher is better.

Method	Pearson Δ			DE Overlap@N		
	↑ Fit	↑ Retention	↑ Heldout	↑ Fit	↑ Retention	↑ Heldout
Naive (vf=0, pert=0, ctx=0)	0.193 (±0.004)	0.629 (±0.001)	0.163 (±0.004)	0.172 (±0.001)	0.287 (±0.001)	0.161 (±0.001)
Naive (vf=0, pert=1, ctx=1)	0.359 (±0.003)	0.595 (±0.001)	0.169 (±0.012)	0.213 (±0.000)	0.275 (±0.000)	0.165 (±0.002)
Naive (vf=1, pert=0, ctx=0)	0.616 (±0.001)	0.216 (±0.006)	0.351 (±0.002)	0.260 (±0.001)	0.156 (±0.001)	0.198 (±0.001)
Naive (vf=1, pert=0, ctx=1)	0.616 (±0.001)	0.216 (±0.006)	0.351 (±0.002)	0.260 (±0.001)	0.156 (±0.001)	0.198 (±0.001)
Naive (vf=1, pert=1, ctx=0)	0.616 (±0.001)	0.214 (±0.007)	0.353 (±0.002)	0.260 (±0.001)	0.156 (±0.001)	0.198 (±0.000)
Naive (vf=1, pert=1, ctx=1)	0.616 (±0.001)	0.214 (±0.007)	0.353 (±0.002)	0.260 (±0.001)	0.156 (±0.001)	0.198 (±0.000)
LoRA (r=4, α=8)	0.551 (±0.000)	0.259 (±0.005)	0.391 (±0.004)	0.251 (±0.000)	0.170 (±0.002)	0.216 (±0.000)
LoRA (r=8, α=16)	0.585 (±0.001)	0.260 (±0.002)	0.386 (±0.004)	0.258 (±0.002)	0.169 (±0.000)	0.214 (±0.001)
LoRA (r=16, α=32)	0.608 (±0.001)	0.264 (±0.011)	0.382 (±0.002)	0.261 (±0.001)	0.169 (±0.002)	0.212 (±0.001)
GFT (sigmoid, vf=1)	0.616 (±0.001)	0.226 (±0.006)	0.361 (±0.005)	0.259 (±0.001)	0.157 (±0.001)	0.202 (±0.001)
GFT (linear, vf=1)	0.615 (±0.001)	0.226 (±0.006)	0.363 (±0.005)	0.259 (±0.000)	0.157 (±0.001)	0.203 (±0.001)
Two-stage (10% new ctx → vf=1)	0.618 (±0.000)	0.227 (±0.004)	0.361 (±0.003)	0.257 (±0.001)	0.159 (±0.001)	0.201 (±0.000)
Two-stage (50% new ctx → vf=1)	0.617 (±0.001)	0.266 (±0.004)	0.377 (±0.001)	0.260 (±0.001)	0.166 (±0.000)	0.208 (±0.001)
Real replay (adaptive λ)	0.610 (±0.001)	0.632 (±0.001)	0.406 (±0.004)	0.260 (±0.000)	0.288 (±0.001)	0.218 (±0.001)
Real replay (equal-task λ)	0.613 (±0.001)	0.620 (±0.001)	0.401 (±0.001)	0.257 (±0.001)	0.285 (±0.002)	0.217 (±0.001)
Generative replay (ODE, adaptive λ)	0.611 (±0.001)	0.594 (±0.005)	0.403 (±0.005)	0.257 (±0.001)	0.276 (±0.002)	0.217 (±0.001)
Generative replay (ODE, equal-task λ)	0.613 (±0.000)	0.588 (±0.002)	0.400 (±0.003)	0.257 (±0.003)	0.275 (±0.003)	0.217 (±0.001)
Generative replay (MF-1, adaptive λ)	0.611 (±0.001)	0.580 (±0.005)	0.403 (±0.005)	0.257 (±0.001)	0.280 (±0.003)	0.217 (±0.000)
Generative replay (MF-1, equal-task λ)	0.613 (±0.000)	0.573 (±0.002)	0.400 (±0.002)	0.256 (±0.002)	0.277 (±0.002)	0.217 (±0.000)
Generative replay (MF-4, adaptive λ)	0.609 (±0.001)	0.588 (±0.003)	0.407 (±0.005)	0.256 (±0.001)	0.279 (±0.003)	0.219 (±0.001)
Generative replay (MF-4, equal-task λ)	0.613 (±0.000)	0.581 (±0.002)	0.400 (±0.002)	0.256 (±0.003)	0.278 (±0.002)	0.218 (±0.001)
Trajectory replay (ODE, adaptive λ)	0.612 (±0.001)	0.619 (±0.002)	0.399 (±0.002)	0.256 (±0.002)	0.285 (±0.001)	0.217 (±0.000)
Trajectory replay (ODE, equal-task λ)	0.610 (±0.001)	0.606 (±0.002)	0.398 (±0.001)	0.257 (±0.001)	0.282 (±0.001)	0.217 (±0.001)
Trajectory replay (MF-1, adaptive λ)	0.613 (±0.001)	0.620 (±0.001)	0.404 (±0.001)	0.256 (±0.002)	0.285 (±0.001)	0.218 (±0.001)
Trajectory replay (MF-1, equal-task λ)	0.608 (±0.002)	0.604 (±0.001)	0.392 (±0.002)	0.257 (±0.001)	0.282 (±0.001)	0.217 (±0.001)
Trajectory replay (MF-4, adaptive λ)	0.613 (±0.001)	0.620 (±0.001)	0.404 (±0.001)	0.256 (±0.002)	0.285 (±0.001)	0.218 (±0.001)
Trajectory replay (MF-4, equal-task λ)	0.608 (±0.002)	0.604 (±0.001)	0.392 (±0.002)	0.258 (±0.000)	0.282 (±0.001)	0.217 (±0.001)

Table 7: **Continual perturbation-response modeling on Parse-PBMC.** Final-checkpoint Pearson Δ and DE Overlap@N after four sequential donor updates, reported as fit on the later tasks, retention on task 0, and heldout on perturbations excluded from training. Means with standard errors over seeds 42, 69, and 1714. Adaptive λ is selected on validation strata and equal-task uses $\lambda_k = k/(k+1)$. Gray marks the privileged real-replay reference and blue marks trajectory replay. Higher is better.

Method	Pearson Δ			DE Overlap@N		
	↑ Fit	↑ Retention	↑ Heldout	↑ Fit	↑ Retention	↑ Heldout
Naive (vf=0, pert=0, ctx=0)	0.233 (±0.003)	0.491 (±0.001)	0.235 (±0.004)	0.100 (±0.004)	0.144 (±0.001)	0.104 (±0.003)
Naive (vf=0, pert=1, ctx=1)	0.232 (±0.005)	0.400 (±0.003)	0.207 (±0.002)	0.112 (±0.002)	0.133 (±0.001)	0.096 (±0.002)
Naive (vf=1, pert=0, ctx=0)	0.290 (±0.000)	0.056 (±0.000)	0.151 (±0.001)	0.114 (±0.001)	0.074 (±0.001)	0.087 (±0.002)
Naive (vf=1, pert=0, ctx=1)	0.290 (±0.000)	0.056 (±0.000)	0.151 (±0.001)	0.114 (±0.001)	0.074 (±0.001)	0.087 (±0.002)
Naive (vf=1, pert=1, ctx=0)	0.289 (±0.001)	0.064 (±0.001)	0.154 (±0.002)	0.114 (±0.002)	0.078 (±0.001)	0.092 (±0.001)
Naive (vf=1, pert=1, ctx=1)	0.289 (±0.001)	0.064 (±0.001)	0.154 (±0.002)	0.114 (±0.002)	0.078 (±0.001)	0.092 (±0.001)
LoRA (r=4, α=8)	0.273 (±0.004)	0.154 (±0.007)	0.214 (±0.002)	0.123 (±0.002)	0.083 (±0.003)	0.111 (±0.001)
LoRA (r=8, α=16)	0.279 (±0.003)	0.155 (±0.008)	0.217 (±0.001)	0.122 (±0.002)	0.085 (±0.003)	0.110 (±0.002)
LoRA (r=16, α=32)	0.287 (±0.003)	0.166 (±0.006)	0.225 (±0.003)	0.122 (±0.003)	0.086 (±0.004)	0.109 (±0.002)
GFT (sigmoid, vf=1)	0.291 (±0.001)	0.061 (±0.001)	0.156 (±0.001)	0.114 (±0.002)	0.075 (±0.001)	0.087 (±0.002)
GFT (linear, vf=1)	0.291 (±0.001)	0.060 (±0.001)	0.157 (±0.001)	0.113 (±0.001)	0.074 (±0.001)	0.087 (±0.002)
Two-stage (10% new ctx → vf=1)	0.289 (±0.001)	0.059 (±0.002)	0.155 (±0.002)	0.113 (±0.001)	0.076 (±0.000)	0.086 (±0.002)
Two-stage (50% new ctx → vf=1)	0.298 (±0.002)	0.091 (±0.002)	0.171 (±0.002)	0.121 (±0.001)	0.077 (±0.001)	0.092 (±0.002)
Real replay (adaptive λ)	0.498 (±0.002)	0.483 (±0.000)	0.350 (±0.003)	0.165 (±0.001)	0.143 (±0.001)	0.137 (±0.001)
Real replay (equal-task λ)	0.496 (±0.001)	0.500 (±0.002)	0.355 (±0.005)	0.167 (±0.002)	0.147 (±0.001)	0.141 (±0.002)
Generative replay (ODE, adaptive λ)	0.475 (±0.001)	0.383 (±0.002)	0.320 (±0.003)	0.122 (±0.001)	0.059 (±0.002)	0.100 (±0.002)
Generative replay (ODE, equal-task λ)	0.471 (±0.003)	0.403 (±0.004)	0.330 (±0.003)	0.125 (±0.001)	0.061 (±0.001)	0.107 (±0.001)
Generative replay (MF-1, adaptive λ)	0.437 (±0.001)	0.306 (±0.007)	0.289 (±0.002)	0.141 (±0.005)	0.088 (±0.005)	0.112 (±0.006)
Generative replay (MF-1, equal-task λ)	0.440 (±0.004)	0.340 (±0.001)	0.314 (±0.004)	0.148 (±0.001)	0.101 (±0.002)	0.124 (±0.002)
Generative replay (MF-4, adaptive λ)	0.446 (±0.002)	0.316 (±0.007)	0.305 (±0.001)	0.150 (±0.005)	0.101 (±0.004)	0.122 (±0.003)
Generative replay (MF-4, equal-task λ)	0.441 (±0.004)	0.345 (±0.007)	0.323 (±0.007)	0.152 (±0.001)	0.108 (±0.002)	0.130 (±0.000)
Trajectory replay (ODE, adaptive λ)	0.496 (±0.004)	0.435 (±0.003)	0.332 (±0.006)	0.162 (±0.002)	0.128 (±0.004)	0.129 (±0.002)
Trajectory replay (ODE, equal-task λ)	0.515 (±0.002)	0.450 (±0.003)	0.348 (±0.007)	0.161 (±0.003)	0.129 (±0.003)	0.131 (±0.003)
Trajectory replay (MF-1, adaptive λ)	0.493 (±0.003)	0.430 (±0.003)	0.333 (±0.003)	0.158 (±0.001)	0.127 (±0.001)	0.127 (±0.001)
Trajectory replay (MF-1, equal-task λ)	0.514 (±0.003)	0.449 (±0.007)	0.349 (±0.006)	0.161 (±0.004)	0.129 (±0.005)	0.132 (±0.004)
Trajectory replay (MF-4, adaptive λ)	0.493 (±0.003)	0.431 (±0.003)	0.333 (±0.003)	0.158 (±0.001)	0.127 (±0.001)	0.127 (±0.001)
Trajectory replay (MF-4, equal-task λ)	0.515 (±0.002)	0.451 (±0.005)	0.350 (±0.007)	0.161 (±0.005)	0.130 (±0.005)	0.132 (±0.004)

Alternative metrics. Fig. 22 repeats the fit-retention view under two metrics from Adduri et al. (2026) that stress different failure modes. L1 discrimination (top) ranks each predicted profile against the observed profiles of all strata by L_1 distance and reports how far its own target sits in that ranking, so it rewards predictions that are specific to their condition rather than merely close to the average response. Residual Pearson Δ (bottom) subtracts the mean perturbation effect across strata before correlating, leaving only the context-specific deviation, so a method predicting the shared response alone scores near zero. The layout is unchanged under both: freezing sits top-left, the adapters and fine-tuning variants low on retention, and the replay family in the upper-right corner with trajectory replay adjacent to real replay. Residual Δ matters most, since it rules out the reading that all methods recover the same global shift and differ only in how much of it they keep.

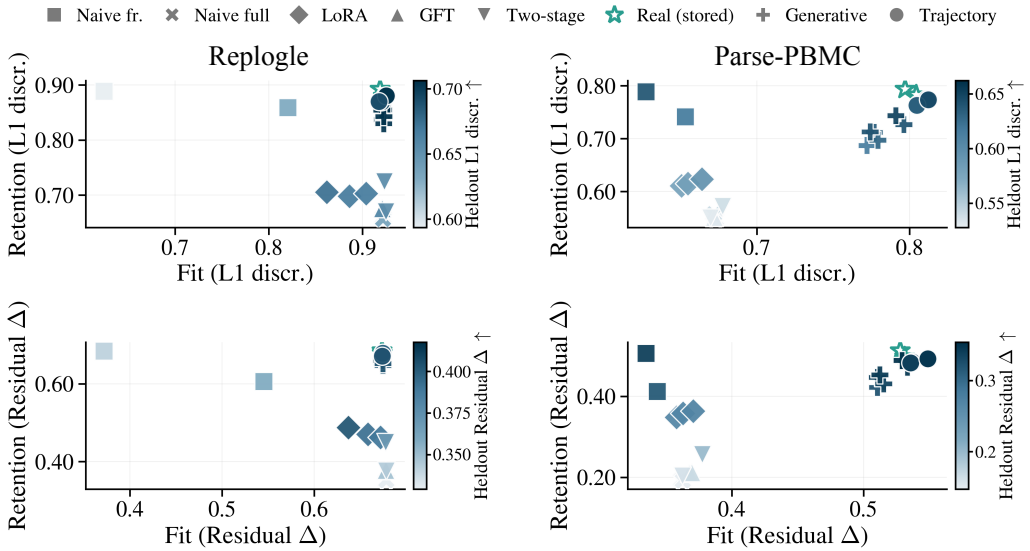

Figure 22: **Fit–retention trade-off under alternative metrics.** Retention against fit for every configuration on Replogle-Nadig (left) and Parse-PBMC (right), measured by L1 discrimination (top) and residual Pearson Δ (bottom). Marker shape denotes the method family and marker color the heldout score for the same metric, with a separate color scale per panel. Higher is better on all axes.

Sequential behavior on PBMC. Fig. 23a tracks held-out Pearson Δ after each donor update. Naive fine-tuning collapses at the first update, while the three replay variants decline in parallel from roughly 0.43 to 0.33–0.35, with trajectory replay tracking real replay. That the privileged reference declines equally points to the difficulty of covering more donors at fixed capacity rather than to forgetting specific to generated replay. Panel b shows all three selecting weights well below the equal-task schedule, real replay included, so the criterion favors the incoming donor rather than discounting generated samples, consistent with a shared perturbation vocabulary where each new context mainly needs its own basal position fitted.

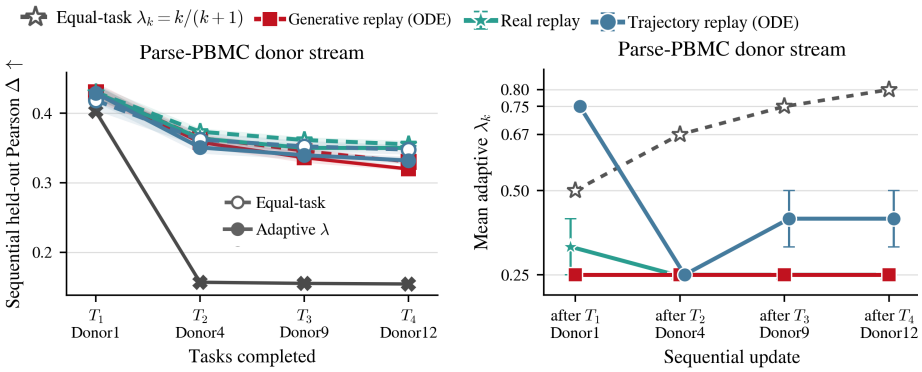

Figure 23: **Parse-PBMC donor stream.** (**a**) Held-out Pearson Δ after each donor update for naive fine-tuning, real replay, and generative and trajectory replay with an ODE backend. Filled markers with solid lines denote adaptive λ and open markers with dashed lines equal-task λ, and shaded bands show variability across seeds. (**b**) Mean replay weight λ_k selected at each update by the adaptive policy for the same replay methods, with the equal-task rule $\lambda_k = k/(k+1)$ as a dashed reference. Error bars show variability over seeds 42, 69, and 1714. Higher is better in (**a**).

Replay budget. Fig. 24 asks what the MeanFlow backends give up. Each point is a backend, MF-1, MF-4, or the ODE teacher, plotted at its replay-field evaluations per update and normalized by the same method's ODE run, so parity means the distilled teacher matches full integration. On Replogle, both families sit at parity across the whole range, so a single update is short enough that teacher fidelity does not matter. On PBMC, the two separate: trajectory replay stays at parity under both weighting policies, whereas generative replay loses about ten percent at one evaluation and only recovers as the budget grows. The asymmetry is the same as on the image streams, since a replay sample must be integrated to $t = 1$ before it can serve as data, while trajectory replay supervises at an intermediate time where a coarse trajectory is still informative.

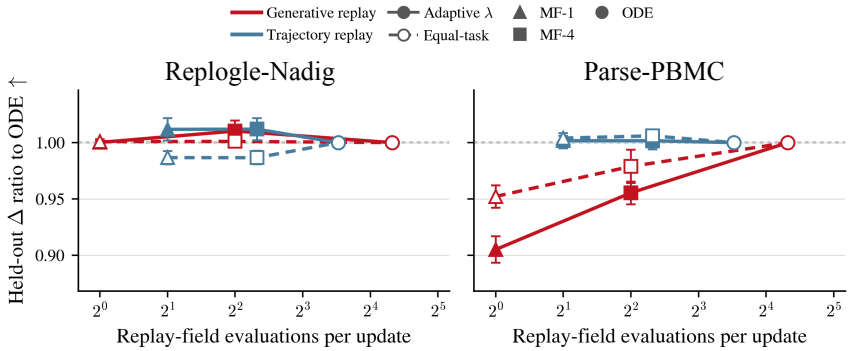

Figure 24: **Replay budget on the transcriptomics streams.** Final held-out Pearson Δ relative to the matched ODE run as a function of replay-field evaluations per update, for Replogle-Nadig (left) and Parse-PBMC (right), with the MF-1, MF-4, and ODE backends labeled by their evaluation count. Colors distinguish generative from trajectory replay, filled markers with solid lines denote adaptive λ and open markers with dashed lines equal-task λ, and the dotted line marks parity with the matched ODE run. Error bars show variability over seeds 42, 69, and 1714. Higher is better.

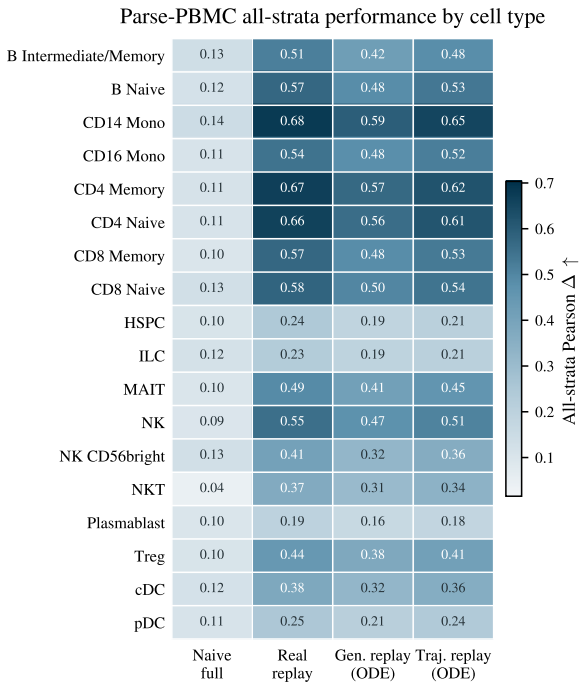

Figure 25: **Parse-PBMC performance by cell type.** Pearson Δ for each annotated cell type under naive full fine-tuning, real replay, and generative and trajectory replay with an ODE backend, all using equal-task λ. Each cell averages over the donor-cytokine strata of that cell type and over seeds 42, 69, and 1714. Higher is better.

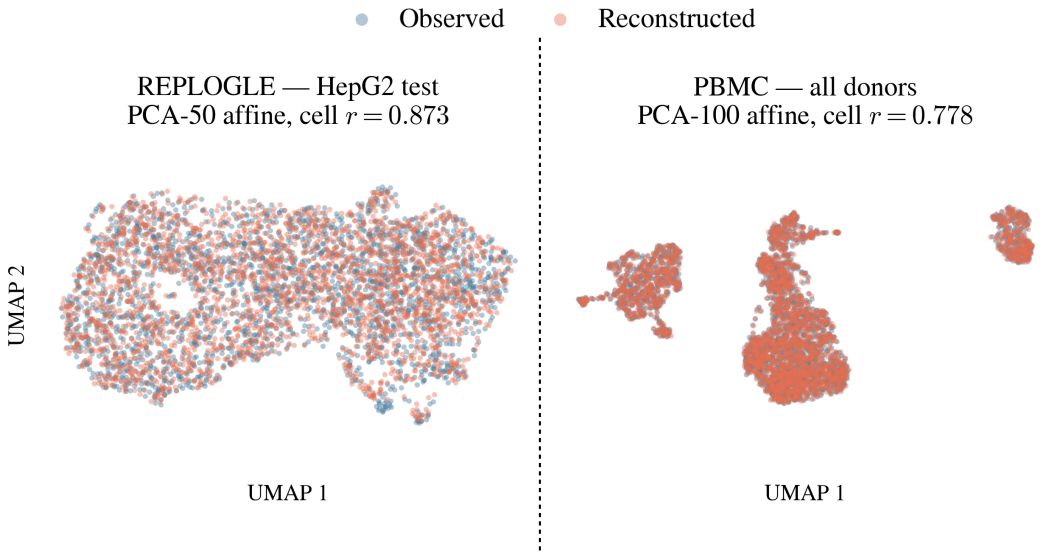

Figure 26: **Decoder reconstruction quality.** Joint UMAP of observed cells and their reconstructions from the observed PCA latents through the frozen affine decoder, for Replogle-Nadig HepG2 test cells with PCA-50 (left) and Parse-PBMC cells from all donors with PCA-100 (right). Panel titles report the mean per-cell reconstruction correlation. Each panel embeds observed and reconstructed cells together, so no flow-matching generation is involved.

Performance by cell type. Fig. 25 breaks PBMC down by annotated cell type. The ordering is identical in all eighteen: naive fine-tuning near zero, then generative replay, then trajectory replay,

then real replay, with trajectory replay closer to the stored-data reference than to generative replay throughout. Absolute values track cell-type abundance rather than method, from 0.65 on CD14 monocytes to 0.18 on plasmablasts, so the residual difficulty is in rare populations for which few cells are available per donor-cytokine stratum rather than in any method-specific failure.

Decoder quality. Fig. 26 embeds observed cells together with their reconstructions through the frozen affine decoder, with no generation involved. The two clouds overlap throughout on both datasets, so the PCA basis and decoder preserve the population structure that the metrics are computed on, and the decoder is not a confound in the method comparison. It does, however, bound what any method can reach: per-cell correlation is 0.873 on Replogle and 0.778 on PBMC, so reported scores sit below a ceiling set by the task-0 PCA basis, which later tasks must also be expressed in.

D.6 Application: Continual phenotypic modeling in microscopy

D.6.1 Experimental details

Data. We use RxRx1 (Sypetkowski et al., 2023), a fluorescence-microscopy screen of gene knockdowns acquired in separate experimental batches, following the dataset choice, preprocessing pipeline, and modeling approach of Zhang et al. (2025) and Jones et al. (2026). Exact values for every setting below are in Tab. 8.

- *Content.* We restrict to the U2OS cell line as Zhang et al. (2025), spanning three acquisition experiments and 1,071 perturbation identities, of which 100 form the official evaluation subset. Since every crop is U2OS, cell-type conditioning carries no information and is disabled.
- *Crops.* Each sample is a 96×96 crop with all six fluorescence channels retained, modeled natively rather than as an RGB projection or a latent code.
- *Normalization.* Pixels are mapped to $[-1, 1]$. At training time, we add uniform dequantization noise to avoid fitting the quantization lattice, and apply random flips. Neither is applied at evaluation.
- *Conditioning.* The model receives perturbation and experiment identity as independent learned embeddings. Experiment identity is an observed acquisition covariate, not a phenotype.
- *Channel projection.* Generated and real tensors are mapped to RGB through the same fixed linear projection and re-quantized before any metric, so the comparison never mixes projections.

Stream. Tasks arrive in the fixed order U2OS-01 $\rightarrow$ U2OS-02 $\rightarrow$ U2OS-03, so the shift is technical rather than semantic: the perturbation vocabulary is *identical* in every task and only the acquisition batch changes. A deterministic 10% of each experiment's training rows is held out for validation and λ selection, and the official test rows are never trained on, replayed from, or used for selection. Task 0 is trained once per seed and shared by every method within each objective family, the flow matching and the diffusion one, so all differences arise from the updates.

Model and training. The velocity field is the ADM U-Net (Dhariwal & Nichol, 2021) adapted to $96\times96\times6$ inputs, trained on the linear noise-to-data interpolant with target $\boldsymbol{x}_1 - \boldsymbol{x}_0$. Every task receives the same budget of 25,000 optimizer steps with Adam and a power-function EMA (Karras et al., 2024), and all reported samples come from the EMA weights.

Replay. Generative and trajectory replay follow the same formulation as in the image experiments (Sec. D.3), with the teacher integrated by Euler at step size 0.05. Equal-task weighting gives $\lambda = 1/2$ at task 1 and $2/3$ at task 2, and the adaptive policy selects from $\{0.25, 0.5, 0.75\}$ by macro validation FID across the experiments seen so far. As in the image streams, we also ask whether a distilled mean velocity can replace the ODE teacher: we distill a MeanFlow model at task 0 for 25,000 steps, matching the per-task budget but at batch size 16 rather than 64, and replay with $N \in \{1, 4\}$ jumps. Here this is restricted to trajectory replay under *equal-task* weighting, since the MeanFlow residual requires a Jacobian-vector product whose activation footprint scales with the $96\times96\times6$ state, making co-training substantially more expensive than at 32×32. The reduced batch is a direct consequence of the same cost.

Baselines. We report four reference points, all on the same stream, per-task budget, backbone, and evaluation protocol as our methods (Tab. 8).

- *Joint training* retrains on the union of all real data seen so far, storing every past sample, and is therefore an upper target rather than a competing method.
- *Real replay* mixes stored historical images into each batch. It is likewise privileged but respects the sequential budget, isolating the cost of replaying generated rather than stored samples.
- *Naive fine-tuning* continues training on the incoming experiment alone with no retention mechanism, giving the lower target.
- *Generative distillation (diffusion)* (Masip et al., 2025) replays samples from the frozen previous-task model but supervises them with the teacher's own noise prediction at a randomly drawn timestep rather than with the injected noise, so the student matches the teacher's denoising behavior along the whole trajectory instead of only reproducing its final samples. We run it on the identical backbone, stream, budget, and EMA as trajectory replay, with a full current batch plus a

Table 8: Configuration for the continual microscopy experiments on RxRx1.

Setting	Value
Data	
Cell line / experiments	U2OS, 3 (U2OS-01/02/03)
Input	$6 \times 96 \times 96$, `uint8` $\rightarrow [-1, 1]$
Train augmentation	uniform dequantization $1/128$, random H/V flips
Perturbations	1,071 total, 100 evaluated
Validation split	10% of each experiment's train rows
Model	
Backbone	Config-B ADM U-Net, 56.2M parameters
Base channels / blocks	128 / 4 per resolution
Channel multipliers	$(2, 2, 2)$, attention at one resolution
Dropout / label dropout	0.3 / 0.15
Conditioning	perturbation + experiment (cell type disabled)
Training	
Steps per task	25,000
Batch size	64 (one H100)
Optimizer	Adam, 10^{-4}
Gradient clip	0.5 (flow matching), 1.0 (diffusion)
EMA	power-function, $\sigma_{\text{rel}} = 0.01$
Seeds	42, 69, 1714
Replay	
Teacher sampler	Euler, step 0.05 (20 evaluations)
Equal-task λ	$k/(k+1)$, i.e. $1/2$ then $2/3$
Adaptive λ grid	$\{0.25, 0.5, 0.75\}$, macro validation FID
MeanFlow jumps	$N \in \{1, 4\}$, trajectory replay only
MeanFlow (μ, σ)	$(2.0, 2.0)$, logit-normal
MeanFlow diagonal rate	0.25
MeanFlow optimizer	AdamW, 10^{-4}
MeanFlow batch size	16
MeanFlow distillation steps	25,000 (same as task-0 training)
Evaluation	
Generated samples	one per test condition (composition-matched)
Benchmark size	2,942 treated test images (961/1000/981)
Sampler	Dopri5, EMA weights, eval seed 2027
Metrics	Inception FID, DINOv2 FDD
Pooling	features concatenated across experiments

full generated batch ($32+32$), task-proportional weighting at replay coefficient 0.75, the post-task online model as teacher, and zero-terminal-SNR DDIM with 20 teacher and 50 evaluation steps.

Because Masip et al. (2025) evaluated only natural-image streams, every diffusion entry here is our *own matched run* of their method on RxRx1.

Evaluation. At the final checkpoint, we evaluate each sequential experiment separately and then pool (Zhang et al., 2025). Following prior work, with only a few dozen test images per perturbation, Fréchet distances cannot be estimated conditionally, so we report them marginally over perturbations and rely on composition matching to make the comparison meaningful: we generate exactly one image per test condition, so the generated and reference sets share the same perturbation-by-experiment composition and no perturbation is over-represented. The benchmark comprises 2,942 treated test images, with the 120 untreated controls excluded. We report Inception FID alongside a Fréchet distance in DINOv2 (Oquab et al., 2024) feature space (FDD) at a pinned revision, sampling with an adaptive Dopri5 solver. All numbers are means with standard errors over seeds.

D.6.2 Additional results

Final-checkpoint performance. Tab. 9 reports every configuration at the end of the stream. Trajectory replay is the best memory-free method on both pooled metrics, at 16.8 FID and 96.1 FDD against 22.0 and 119.0 for generative replay, narrowing the gap to the privileged references. The per-

experiment columns locate the difference: the families are comparable on the two later experiments, but on U2OS-01 generative replay degrades to 70.9–71.4 while trajectory replay stays at 39.6–40.5. Since the perturbation vocabulary is identical across tasks, this is the oldest acquisition batch being overwritten rather than any loss of semantic coverage. The MeanFlow backends cost a little, 19.4 FID at one jump against 17.2 for the matched ODE run, but remain ahead of generative replay at any backend. Diffusion distillation, the strongest published no-storage baseline on natural-image streams, is the weakest replay method here on every column at 46.4 FID and 218.2 FDD.

Table 9: **Continual phenomic generation on RxRx1.** Final-checkpoint performance after the U2OS-01→U2OS-02→U2OS-03 experiment stream. Per-experiment FID compares composition-matched generated and test images within each biological experiment and stream metrics pool all three experiments, with FDD denoting Fréchet distance in DINOv2 (Oquab et al., 2024) feature space. Means with standard errors over seeds 42, 69, and 1714. Adaptive λ is selected using validation FID and equal-task uses $\lambda_k = k/(k+1)$. Gray marks the privileged references that retain real historical data and blue marks trajectory replay. Bold marks the best memory-free mean in each column. Lower is better.

Method	Per-experiment FID ↓			Pooled stream ↓	
	U2OS-01	U2OS-02	U2OS-03	FID	DINOv2 FDD
Joint training (all data; upper bound)	29.7 (±0.2)	19.9 (±0.2)	26.6 (±0.2)	12.3 (±0.2)	66.4 (±1.4)
Real replay (stored data)	29.7 (±0.4)	19.6 (±0.2)	26.3 (±0.3)	12.2 (±0.3)	66.1 (±1.0)
Naive fine-tuning	**36.0** (±0.6)	83.0 (±1.6)	**26.0** (±0.1)	24.5 (±0.3)	110.1 (±1.3)
Generative distillation (diffusion) (Masip et al., 2025)	80.9 (±6.9)	82.1 (±2.6)	42.0 (±0.3)	46.4 (±3.6)	218.2 (±13.4)
Generative replay (ODE, adaptive λ)	71.4 (±0.3)	27.1 (±0.2)	27.0 (±0.2)	22.0 (±0.1)	119.0 (±1.6)
Generative replay (ODE, equal-task λ)	70.9 (±1.1)	26.8 (±0.1)	27.7 (±0.1)	22.2 (±0.2)	121.4 (±1.7)
Trajectory replay (ODE, adaptive λ)	39.6 (±0.3)	**22.5** (±0.6)	28.8 (±0.1)	**16.8** (±0.2)	**96.1** (±1.9)
Trajectory replay (ODE, equal-task λ)	40.5 (±0.4)	23.1 (±0.6)	28.6 (±0.2)	17.2 (±0.4)	99.5 (±2.9)
Trajectory replay (MF-1, equal-task λ)	48.9 (±4.5)	25.4 (±0.9)	28.8 (±0.2)	19.4 (±1.2)	102.2 (±3.2)
Trajectory replay (MF-4, equal-task λ)	46.4 (±3.4)	24.7 (±0.8)	28.9 (±0.1)	18.9 (±0.7)	101.7 (±2.4)

Stream evolution. Fig. 27 tracks the same quantities after each update, and panel a shows the final checkpoint hiding a non-monotonic effect. Naive fine-tuning loses U2OS-01 at the first update, rising to roughly 97 FID, then falls back to 36.0, nominally the best memory-free value in that column. This is not retention but a *similarity artifact*: U2OS-01 and U2OS-03 are visually close in illumination and staining intensity, whereas U2OS-02 is the outlier (Fig. 29), so training on the third experiment partially restores the first. The excursion in between is the real measurement, and its size, a threefold FID increase in a single update, is what makes the point about forgetting under a purely technical shift. Both replay families avoid it, with trajectory replay flat across the stream and generative replay drifting upward from the second update. Panel b shows the pooled metric over a growing reference set, where trajectory replay separates at the first update and holds the margin.

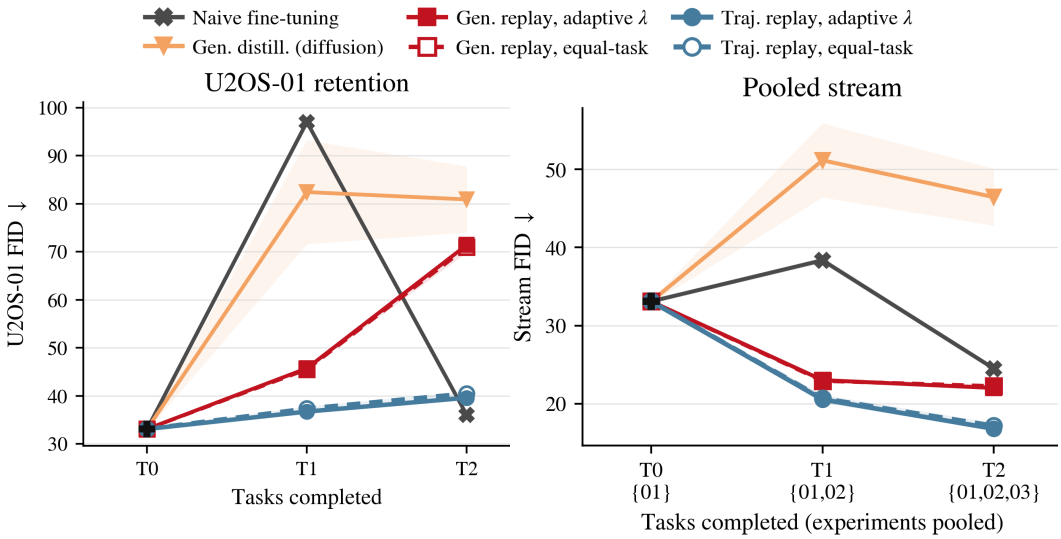

Figure 27: **RxRx1 experiment stream.** (**a**) FID on U2OS-01 and (**b**) pooled stream FID as a function of tasks completed, for naive fine-tuning, diffusion distillation, and generative and trajectory replay under adaptive and equal-task λ. The pooled reference set in (**b**) grows with each task, as indicated below the tick labels, whereas (**a**) keeps U2OS-01 fixed. Lines show the mean and shaded bands the variability over seeds 42, 69, and 1714. Lower is better.

Weighting and backend. Fig. 28a separates the two knobs. On the left, the choice of weighting policy barely moves either family, 16.8 against 17.2 FID for trajectory replay and 22.0 against 22.2 for generative replay, so the gap between them is not an artifact of how λ was set. On the right, the selected weights nonetheless differ: trajectory replay tracks the equal-task rule across both updates, while generative replay settles below it and declines, the same pattern as on the image streams and consistent with the criterion discounting replayed samples whose fidelity is low.

Panel b sweeps the backend. The ODE teacher integrates to an intermediate time at step size 0.05, averaging 11.5 evaluations per update against the 20 a full generation would cost, and two MeanFlow evaluations reach 19.4 FID against its 17.2, with the FDD ordering the same at a proportionally smaller spread. A near-sixfold reduction in replay cost therefore buys back about two FID points, which is the regime in which the mean velocity is worth using.

Qualitative samples. Fig. 29 shows real and trajectory-replay crops for three perturbations across the three experiments. The batch structure is visible directly: U2OS-01 and U2OS-03 share a warm, brightly stained appearance while U2OS-02 is darker and shifted toward green, which is the similarity that lets naive fine-tuning recover U2OS-01 after training on U2OS-03. Generated crops reproduce the staining balance and nuclear morphology of their own experiment rather than defaulting to the appearance of the most recent one.

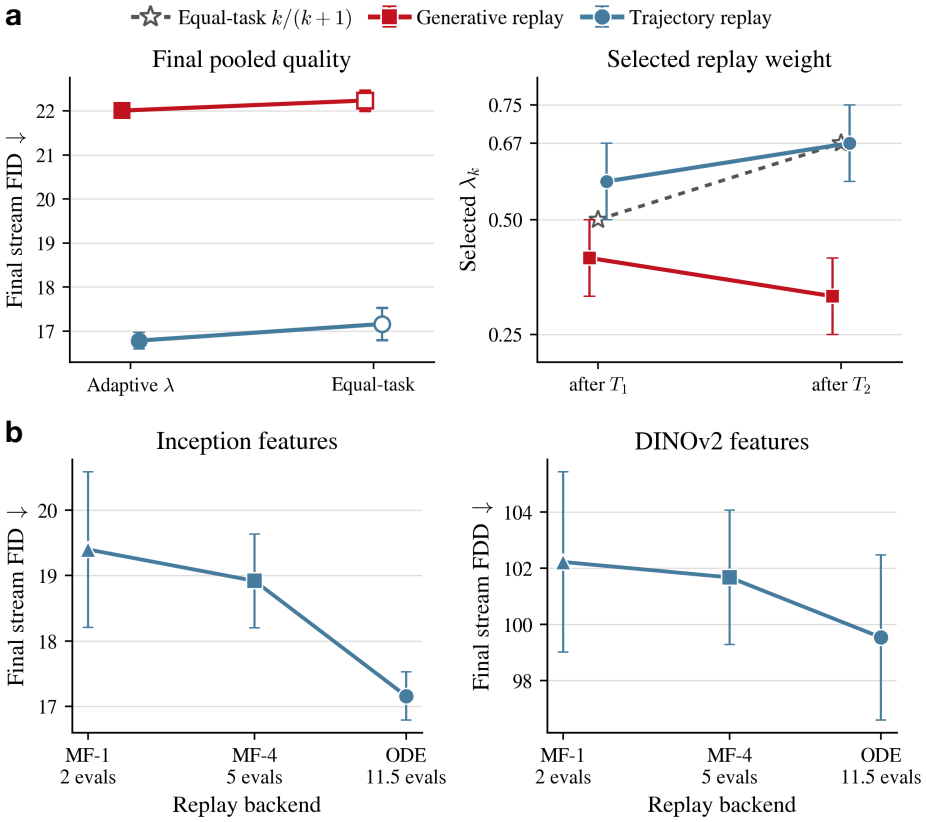

Figure 28: **Replay-weight and backend ablations on RxRx1.** (**a**) Final pooled stream FID under adaptive and equal-task λ for generative and trajectory replay (left), with filled markers denoting adaptive λ and open markers equal-task, and the mean replay weight λ_k selected at each update (right), with the equal-task rule $\lambda_k = k/(k+1)$ as a dashed reference. (**b**) Final pooled stream FID in Inception features (left) and final pooled stream Fréchet DINOv2 distance (FDD) in DINOv2 features (right) for trajectory replay with the MF-1, MF-4, and ODE backends, labeled by their replay-field evaluations per update. Error bars show variability over seeds 42, 69, and 1714. Lower is better for both metrics.

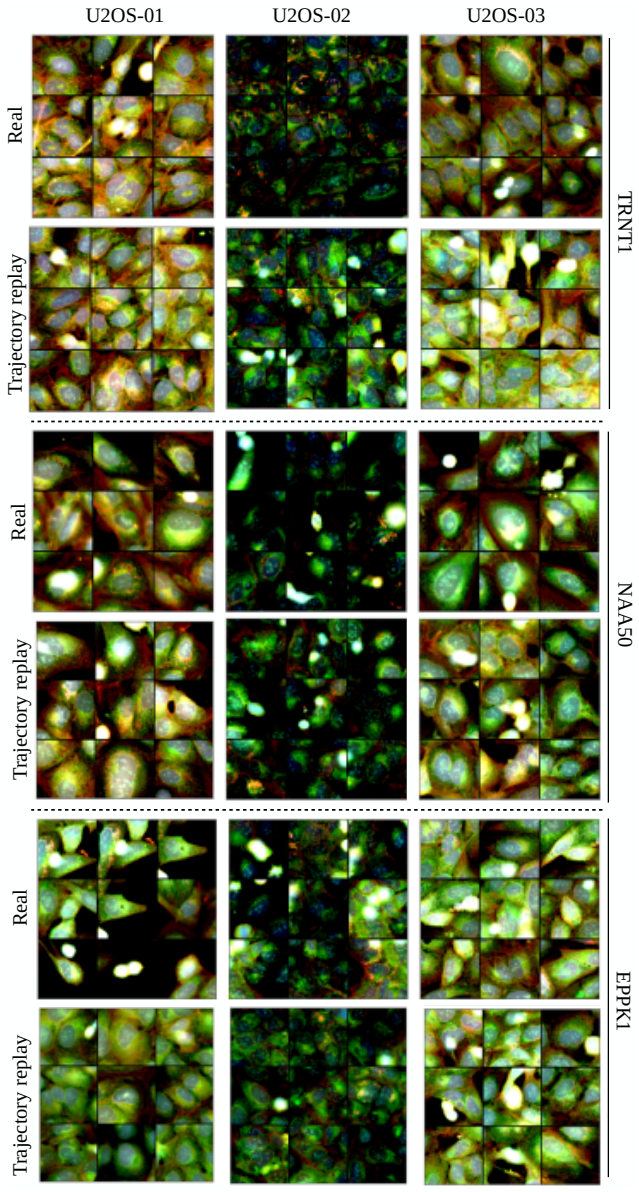

Figure 29: **RxRx1 samples after the final update.** Real test images and trajectory-replay samples for the perturbations TRNT1, NAA50, and EPPK1, shown for each biological experiment of the stream. Each block contains nine images from the corresponding perturbation and experiment, rendered through the fixed six-channel-to-RGB projection.