REWARD-ALIGNING FEW-STEP FLOW MODELS WITH INTEGRATED REGULARIZERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based generative models are increasingly adapted to downstream rewards, preferences, and design objectives. Most existing approaches adapt the time-dependent drift of the underlying continuous flow. These continuous-time dynamics enable effective and probabilistically-principled reward alignment, but incur the cost of expensive simulations and require the challenging propagation of terminal reward gradients backwards through the trajectories. Recent accelerated one- and few-step samplers raise the prospect of improved fine-tuning efficiency, however such approaches side-step the continuous dynamics that are useful for effective and stable regularization. In this work, we propose a method for fine-tuning a flow map that preserves the advantages of both regimes, while also retaining the ability to perform any-step generation. Our approach directly optimizes the few-step generations using gradients of the reward function, while regularizing using a family of integrated divergences along the continuous-time probability flow induced by the model. Experiments on image and protein generation show that the proposed approach combines the efficiency of few-step generation with the stability of continuous-time regularization along the probability flow path.

1 INTRODUCTION

Reward-Alignment Generative models are increasingly deployed not merely to generate novel samples, but as models that can be adapted to downstream objectives (Bansal et al., 2024; Kim et al., 2023; Ma et al., 2025). For example, one may wish to bias samples toward high-reward regions, satisfy design criteria, or reflect user preferences. To do so, a natural approach is regularized reward-alignment; that is, given a pretrained generative model and a chosen reward function $r : \mathbb{R}^d \to \mathbb{R}$, we wish to improve the expected reward of generated samples while preventing collapse by keeping the adapted model close to the original.

Fine-tuning Flow Models For flow-based generative models (Song et al., 2020; Ho et al., 2020; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022; Liu et al., 2022; Peluchetti, 2023), this problem is usually formulated at the level of the continuous-time dynamics. Flow matching, diffusion models, and stochastic interpolants generate samples by specifying intermediate marginals connecting a simple prior to the data distribution, and learning an ODE or SDE whose drift realizes transport along these marginals. This continuous-time probabilistic structure is particularly useful for regularization, as it allows for comparing the base and adapted models along these induced intermediate distributions rather than only at the endpoint.

However, fine-tuning the continuous-time drift can be both costly and challenging. Firstly, they require repeated expensive simulation of the ODE or SDE trajectory. Secondly, rewards are typically defined only on clean terminal samples, whereas the drift acts at intermediate times along the generative process. Updating the transport therefore requires propagating reward information from the final sample back to the intermediate states along the generated trajectory. These factors become especially limiting when adaptation must be repeated for new rewards functions, preferences, or design objectives.

Few-Step Flow Models Motivated by the desire for faster generation, many flow-based models are no longer used as many-step ODE or SDE samplers but are instead distilled into one-step or few-step

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.

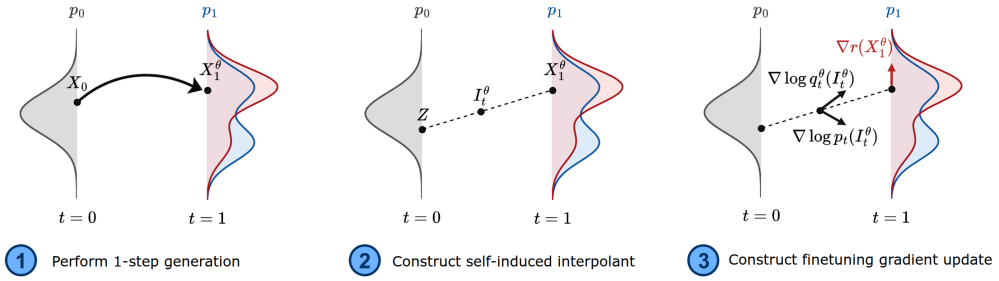


Figure 1: Overview of our proposed integrated regularization mechanism for fine-tuning a flow-map from the base distribution p_1 (blue) to the target regularized terminal measure q_1^* (red). (1) The flow map produces samples X_1^θ from the current q_1^θ using the few- or single-step generative procedure. (2) These samples define a self-induced interpolant between p_0 and q_1^θ, induced by intermediate states I_t^θ. (3) The fine-tuning gradient update is constructed using $\nabla r(X_1^\theta)$ evaluated with direct differentiation, and a regularization term comparing $\nabla \log p_t(I_t^\theta)$ and $\nabla \log q_t(I_t^\theta)$ along the interpolant. The flow map is also trained to be self-consistent with its own generations, so only a single set of trainable parameters is used and the model retains its flow-map structure.

generators that transport noise directly to samples. Such accelerated samples offer the opportunity to avoid the two fine-tuning pitfalls outlined above - they avoid expensive simulation, and also allow for direct reward optimization of the generative model rather than requiring propagating the information backwards through a trajectory.

However, a key question is how to regularize such few-step fine-tuning. Focusing only on the terminal law would compare the models at the least stable point, as the endpoint distributions may be concentrated near low-dimensional sets giving poorly behaved densities, or be difficult to compare from samples alone. In performing the accelerated sampling, such models side-step the continuous dynamics that are useful for regularizing flow-based models in the first place.

Contributions In this work, we show how a few- or single-step generator can be optimized directly, while retaining a regularization mechanism defined through the continuous-time flow dynamics. We focus on fine-tuning a *flow-map* (Boffi et al., 2024; 2025), an accelerated flow model that transports between any two timesteps along the generative trajectory, as this provides a single model that gives access to all the quantities necessary for integrated regularization along the probability path. Preserving the flow-map structure is also desirable, as any-step generation can be useful for downstream tasks such as inference-time steering.

In particular, we make the following contributions:

- Starting with a pretrained 'base' flow map, we finetune the model according to a chosen reward function. The reward function gradients are evaluated on the outputs of the fast generator itself, sidestepping the need to repeatedly simulate and update a continuous-time ODE or SDE.
- We regularize the generator along a continuous probability path induced by its own samples, using integrated KL or Fisher divergences between the current and base model's induced intermediate marginals. We show how a flow-map provides all the necessary terms needed for such training, meaning that the optimisation can be carried out with only a single set of trainable parameters in a simple manner. The same map generates terminal samples and provides the intermediate-time quantities needed for integrated divergence regularization, and is trained with self-induced flow matching and consistency losses so that the adapted model remains coherent as a flow map.
- We compare our proposed regularisation mechanisms in both image and protein generation tasks, and demonstrate strong performance that shares the efficiency of few-step generation with the stability of integrated regularization. In particular, we observe that regularizing earlier in the generative trajectory provides improved stability, suggesting integrated regularization along the probability flow path can be an effective mechanism for reward alignment of fast flow models.

2 BACKGROUND

Flow Matching We first briefly review the basics behind flow matching (Albergo et al., 2023; Lipman et al., 2022; Liu et al., 2022; Peluchetti, 2023), which constructs a generative model by learning a dynamic transport that transforms an initial distribution p_0, typically Gaussian, into the data distribution p_1. Letting $I_0 \sim p_0$ and $I_1 \sim p_1$, we can define the linear interpolant $I_t = (1-t)I_0 + tI_1$, which in turn induces a path of marginal distributions $p_t := \mathrm{Law}(I_t)$. Rather than learning the endpoint distribution directly, flow matching learns dynamics that transport along this entire path. To do so, flow matching constructs the drift function $b_t(x) := \mathbb{E}[I_1 - I_0 \mid I_t = x]$, which can then be used to construct an ODE

$$\dot{x}_t = b_t(x_t), \qquad x_0 \sim p_0, \tag{1}$$

which satisfies $\mathrm{Law}(x_t) = p_t$. Thus the interpolant I_t determines both a path of marginals $(p_t)_{t\in[0,1]}$, and a deterministic transport (1) that realizes this path.

When using a Gaussian source $p_0 = \mathcal{N}(0, I)$, the score of the marginals p_t can be written in terms of the flow matching drift,

$$\nabla \log p_t(x) = \frac{t\, b_t(x) - x}{1-t}. \tag{2}$$

The score and drift are therefore linear reparameterizations of each other and can be used interchangeably, which we will utilize in our proposed approach.

Flow Maps Flow-based generative models can produce high-quality samples, but their generation speed is limited due to the need for simulating the learned ODE or SDE at inference time. This has motivated recent efforts to speed up the inference procedure by learning to compress the generative trajectories into few or single network evaluations. One particular such approach aims to learn the *flow map* for the flow dynamics, which describes how states are carried from one time to another along the generative process (Song et al., 2023; Frans et al., 2025; Geng et al., 2025a; Boffi et al., 2024; 2025). We now briefly review flow maps, as they will form the basis of our approach.

Let $X_{s,t}$ denote the flow map of the ODE in (1); that is, if $(x_\tau)_{\tau\in[0,1]}$ is a trajectory of the ODE, then the flow map satisfies $X_{s,t}(x_s) = x_t$ for any $0 \le s \le t \le 1$. A flow map is therefore an any-step solution operator of the ODE, transporting a state at time s directly to its corresponding position in the trajectory at time t without explicitly integrating intermediate times. It is convenient to parametrize the map in residual form as $X_{s,t}(x) = x + (t-s)v_{s,t}(x)$, where $v_{s,t}(x)$ represents the average velocity over the interval $[s,t]$. Along the diagonal, this average velocity must recover the instantaneous flow-matching drift $v_{t,t}(x) = b_t(x)$. Additionally, a valid flow must also satisfy the semi-group identity; that is, for any $0 \le s \le u \le t \le 1$, we have $X_{u,t} \circ X_{s,u} = X_{s,t}$.

To train a flow map, one parametrizes a candidate map X^θ via v^θ in the same residual form, $X^\theta_{s,t}(x) = x + (t-s)v^\theta_{s,t}(x)$. The diagonal term $\hat{v}_{t,t}$ is trained using a flow matching loss,

$$\mathcal{L}_{\mathrm{fm}}(v^\theta) = \int_0^1 \mathbb{E}\left[\|v^\theta_{t,t}(I_t) - \dot{I}_t\|^2\right] dt, \tag{3}$$

and the off-diagonal terms $v^\theta_{s,t}$ are learned using any consistency loss that enforces the semigroup property $X_{u,t} \circ X_{s,u} = X_{s,t}$. Many consistency losses can be used, including Consistency Trajectory, Mean Flow, and Shortcut Models (Kim et al., 2024; Geng et al., 2025a; Frans et al., 2025).

3 METHOD

In this work, we study how to fine-tune a few- or single-step model, while retaining the appealing regularization benefits provided by the probabilistic structure of flow-based generative models. In particular, we observe that a flow map provides all the necessary objects required for such regularization, allowing the entire fine-tuning procedure to be carried out under a single-model framework.

Setup and Objective Suppose we have a generative model consisting of a map G_θ that sends Gaussian noise $Z \sim p_0 = \mathcal{N}(0, I)$ directly to a terminal sample $X^\theta_1 = G_\theta(Z)$ – for us, this will be our few- or single-step flow map sampler. Our aim is to *fine-tune* this model according to a

chosen reward function $r : \mathbb{R}^d \to \mathbb{R}$. Writing $q_1^\theta := \mathrm{Law}(X_1^\theta)$ for the terminal law induced by this generator, a natural fine-tuning objective is

$$\max_\theta \left\{ \mathbb{E}\big[r(X_1^\theta)\big] - D(q_1^\theta \| p_1) \right\}, \tag{4}$$

where D is a divergence between probability distributions. The reward term encourages samples towards high-reward regions, while the divergence term acts as regularizer to keep the adapted law close to the base terminal distribution and prevent collapse onto a narrow set of high-reward outputs.

3.1 Choice of Divergence

Two natural choices for D in (4) are the KL and Fisher divergences,

$$D_{\mathrm{KL}}(q\|p) := \int q(x) \log \frac{q(x)}{p(x)}\, dx, \qquad D_{\mathrm{FI}}(q\|p) := \int q(x) \left\| \nabla \log \frac{q(x)}{p(x)} \right\|^2 dx. \tag{5}$$

Using $D(q_1^\theta \| p_1)$ directly would involve comparing the endpoint distributions q_1^θ and p_1. However, as the data is often concentrated on low-dimensional sets, these distributions are often ill-behaved or difficult to estimate from samples alone, meaning that direct terminal comparison can be brittle. Rather than using endpoint divergences, an alternative is to compare the reward-aligned and base models along the probabilistic path induced by the interpolant. For $t < 1$, the marginals q_t^θ and p_t are Gaussian-smoothed versions of their respective endpoint laws, giving improved regularity compared to the endpoint distribution. Similarly to the construction of p_t, we can define an interpolant and marginal path for the adapted model by sampling independent $I_0 \sim p_0$ and $X_1^\theta \sim q_1^\theta$, and setting

$$I_t^\theta := (1-t)I_0 + tX_1^\theta, \qquad q_t^\theta := \mathrm{Law}(I_t^\theta). \tag{6}$$

Thus q_t^θ is not parameterized by θ directly; rather, it is induced by the currently-generated terminal law q_1^θ through the definition of the interpolant.

The above discussion therefore suggests replacing the endpoint KL and Fisher divergences by their *integrated* versions along the probability flow,

$$D_{\mathrm{iKL}}(q_1^\theta \| p_1) := \int_0^1 w(t)\, D_{\mathrm{KL}}(q_t^\theta \| p_t)\, dt, \qquad D_{\mathrm{iFI}}(q_1^\theta \| p_1) := \int_0^1 w(t)\, D_{\mathrm{FI}}(q_t^\theta \| p_t)\, dt, \tag{7}$$

where $w : [0,1] \to \mathbb{R}_+$ is a weight function, and q_1^θ together with the interpolant (6) define the family $(q_t^\theta)_{t\in[0,1]}$. These integrated KL and Fisher divergences have proven highly effective in prior works (Wang et al., 2023; Luo et al., 2023; Zhou et al., 2024; Luo et al., 2024) for distilling distribution models into one-step generators. They have also seen success for reward-aligning few-step generators by using them for regularizing towards the base model (Luo, 2025; Luo et al., 2025; Jiang et al., 2026). Here, we will show how they can be used under a single-model framework while retaining a flow-map structure.

3.2 Tractable Gradients

We now consider how to optimize the regularized objective (4). Differentiating the reward term is simple: one can differentiate through the generator $X_1^\theta \sim q_1^\theta$ directly, $\nabla_\theta \mathbb{E}[r(X_1^\theta)] = \mathbb{E}[\nabla_\theta r(X_1^\theta)]$. The more challenging question concerns how to differentiate the integrated divergences with respect to the model parameters θ. The key point is that we require formulations that avoid evaluating $\partial_\theta(\nabla_x \log q_t^\theta)$ explicitly—this is because q_t^θ is defined only implicitly through the generator, so we do not have an available parametric map $\theta \mapsto \nabla_x \log q_t^\theta(x)$ that can be differentiated.

KL Divergence Fortunately, for the integrated KL divergence such a formulation is immediate.

Proposition 3.1 (**Gradient of the integrated KL divergence,** Wang et al. (2023); Luo et al. (2023))**.** *The D_{iKL} has gradient equal to*

$$\nabla_\theta D_{\mathrm{iKL}}(q_1^\theta \| p_1) = \int_0^1 w(t)\, \mathbb{E}\Big[\big(\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\big)^\top \nabla_\theta I_t^\theta \Big]\, dt. \tag{8}$$

where the expectation is over the law of I_t^θ coming from (6).

From this result, we see that the gradient of the integrated KL divergence is determined by the score mismatch $\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)$ at the noised points I_t^θ along the probability flow. In particular, once the score $\nabla_x \log q_t^\theta$ is available at intermediate times, this gradient is straightforward to estimate. This expression for the gradient of D_{iKL} has been shown in prior work (Wang et al., 2023; Luo et al., 2023) to be highly effective for distilling flow-based models into a single step, and in Luo (2025) for reward-alignment. Examining the gradient expression (8), we see that the output is moved in a direction to reduce the mismatch between the fine-tuned and base score functions, giving a simple geometric intuition for why this term counteracts movement away from the base model.

Fisher Divergence For the integrated Fisher divergence, a naive differentiation would appear to be more problematic since the objective itself is defined in terms of $\nabla_x \log q_t^\theta$. Differentiating directly would introduce terms involving $\partial_\theta(\nabla_x \log q_t^\theta)$, precisely the dependence we want to avoid. Previous works in the distillation literature have overcome this by deriving expressions for the gradient requiring only *evaluations* of the score function, without requiring any derivatives with respect to θ (Zhou et al., 2024; Luo et al., 2024). Indeed, using the tower property of conditional expectation, we can express the Fisher divergence gradient in a form that avoids such θ-derivative terms.

Proposition 3.2 (**Gradient of the integrated Fisher divergence,** Zhou et al. (2024)). *The D_{iFI} has gradient equal to*

$$\nabla_\theta D_{\mathrm{iFI}}(q_1^\theta \| p_1) = \nabla_\theta \int_0^1 w(t) \mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta)\big)^\top \Big(\nabla_x \log p_t(I_t^\theta) + \mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta) + \tfrac{2}{1-t} I_0\Big)\Big]\, dt, \quad (9)$$

where $sg[\cdot]$ indicates that $\nabla_x \log q_t^\theta$ is treated as constant with respect to θ when taking the gradient, and where the expectation is over I_t^θ and I_0, which are sampled as in (6).

Note that in the expression $\mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta)$, the gradient with respect to θ is stopped through the score function $\nabla_x \log q_t^\theta$, but is taken through the score function's argument I_t^θ.

Stop-gradient Fisher Divergence Following the distillation literature, we will also consider a *stop-gradient* version of the Fisher divergence, in which the gradient is not taken through sampling distribution in the expectation,

$$D_{\text{iFI-sg}}(q_1^\theta \| p_1) = \int_0^1 w(t)\, \mathbb{E}_{x \sim q_t^{\mathrm{sg}[\theta]}} \left[\left\| \nabla_x \log \frac{q_t^\theta(x)}{p_t(x)} \right\|^2 \right] dt \quad (10)$$

Proposition 3.3 (**Gradient of the integrated stop-gradient Fisher divergence** (10)**,** Zhou et al. (2024); Luo et al. (2024)). *The $D_{\text{iFI-sg}}$ has gradient equal to*

$$\nabla_\theta D_{\text{iFI-sg}}(q_1^\theta \| p_1) = 2\, \nabla_\theta \int_0^1 w(t) \cdot \mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta)\big)^\top \Big(\mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta) + \frac{1}{1-t} I_0\Big)\Big]\, dt, \quad (11)$$

where $sg[\cdot]$ indicates that $\nabla_x \log q_t^\theta$ is treated as constant with respect to θ when taking the gradient, and where the expectation is over I_t^θ and I_0, which are sampled as in (6).

This stop-gradient form of the Fisher divergence has also been shown to work well for diffusion model distillation (Zhou et al., 2024; Luo et al., 2024). We remark that in our case, introducing the stop-gradient means that we are no longer doing maximization on a fixed objective and it is thus somewhat unclear to what object the optimization (4) will converge. While this mechanism therefore lacks the principled variational characterization of the previous two options, we nevertheless include it in our comparisons due to its prevalence and success in the distillation literature.

3.3 Parameterizing with a Single Flow Map

One approach for optimizing (4) would be to train two separate objects: a fast generator G_θ for samples from q_1^θ, and an auxiliary score model for the intermediate-time scores $\nabla_x \log q_t^\theta$. Such a two-model strategy has been shown to be highly effective for distilling diffusion models into single-step generators (Wang et al., 2023; Luo et al., 2023; Zhou et al., 2024; Luo et al., 2024). More recently, similar two-model approaches have also been used for reward aligning few-step models (Luo, 2025; Luo et al., 2025).

In this work, our aim is to instead learn a *single* trainable object that provides everything needed for reward-alignment; that is, a few-step map that can generate samples, while also encoding the intermediate-time structure required by the integrated regularizer.

The natural object for this purpose is a flow map, as it allows for few-step generation while also providing access to the score function via its diagonal velocity. Indeed, suppose the fine-tuned path $(q_t^\theta)_{t\in[0,1]}$ were realized by a flow map $X_{s,t}^\theta$, written in residual form as $X_{s,t}^\theta(x) = x + (t-s)v_{s,t}^\theta(x)$. (that is, the flow-map is trained to fit its own self-induced terminal distribution q_1^θ). The corresponding scores can be obtained through the reparameterization of the diagonal $v_{t,t}^\theta$ as

$$\nabla_x \log q_t^\theta(x) = \frac{t\, v_{t,t}^\theta(x) - x}{1-t}. \tag{12}$$

A well-trained flow map would therefore provide all the objects required by the integrated regularizer objectives. Its endpoint map generates the fine-tuned terminal law $q_1^\theta = (X_{0,1}^\theta)_{\#}p_0$, while its diagonal velocity field gives access to the score of the induced path through the identity (12) above.

3.4 Joint Training

Of course, in practice the terminal distribution q_1^θ will be continually updated during optimization, meaning that the intermediate marginals $(q_t^\theta)_{t\in[0,1]}$ will also be changing. As we require the scores $\nabla_x \log q_t^\theta(x)$ for the regularization, we therefore need to keep the flow-map (or, at least, its diagonal drift $v_{t,t}^\theta$ used in (12)) consistent with its currently generated endpoint distribution q_1^θ.

We will therefore jointly optimize the model X^θ, training according to the reward-alignment objective while also updating to correspond to the flow map fitting the current self-induced endpoint q_1^θ. To do so, given independent $Z \sim p_0$ and $I_0 \sim p_0$ we form the terminal sample $X_1^\theta := X_{0,1}^\theta(Z)$ along with the corresponding self-induced interpolant

$$I_t^\theta := (1-t)I_0 + tX_1^\theta. \tag{13}$$

The reward term is evaluated on the generated sample, $\mathcal{L}_{\text{reward}}(\theta) := -\mathbb{E}\big[r(X_1^\theta)\big]$. The regularization term $\mathcal{L}_{\text{reg}}(\theta) := D(q_1^\theta \| p_1)$ is taken to be either the integrated KL divergence or the integrated Fisher divergence (full, or stop-gradient form). As shown above, its gradient can be evaluated using (8), (9) or (11) respectively.

Reward maximization and regularization alone, however, do not guarantee that X^θ is a coherent flow map, which is necessary for us to be able to extract the relevant scores $\nabla \log q_t^\theta$. We therefore add two structural losses. First, a self-induced flow matching term trains the diagonal of the map to match the dynamics induced by its own sample generator X_1^θ, ensuring we can extract the necessary score function $\nabla_x \log q_t^\theta(x)$ through the expression in (12):

$$\mathcal{L}_{\text{sfm}}(\theta) := \int_0^1 \mathbb{E}\Big[\|v_{t,t}^\theta(\text{sg}[I_t^\theta]) - \text{sg}[\dot{I}_t^\theta]\|^2\Big]\, dt. \tag{14}$$

Second, a consistency loss enforces the semigroup structure expected of a genuine flow map. Any choice of consistency loss is possible; In our experiments we will generally use the semigroup loss,

$$\mathcal{L}_{\text{cons}}(\theta) := \int_0^1 \int_s^1 \int_s^t \mathbb{E}\Big[\|X_{s,t}^\theta(\text{sg}[I_s^\theta]) - \text{sg}[X_{u,t}^\theta(X_{s,u}^\theta(I_s^\theta))]\|^2\Big]\, du\, dt\, ds. \tag{15}$$

Minimizing (14) and (15) ensure that X^θ is a valid flow map for the endpoint sampler. The final combined training objective we wish to minimize is therefore

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{reward}}(\theta) + L_{\text{reg}}(\theta) + \mathcal{L}_{\text{sfm}}(\theta) + \mathcal{L}_{\text{cons}}(\theta). \tag{16}$$

Algorithm 1 Reward-aligning flow maps with integrated regularization

1: **Input:** reward r; pretrained flow map $X_{s,t}$ with scores $\nabla_x \log p_t(x) = \frac{t v_{t,t}(x) - x}{1-t}$; weight $w(t)$.
2: **Initialize:** Initialize $X^\theta_{s,t}(x) = x + (t-s) v^\theta_{s,t}(x)$ at the base flow map $X_{s,t}$.
3: **repeat**
4: Sample $Z \sim p_0$ and generate $X^\theta_1 = X^\theta_{0,1}(Z)$.
5: Sample $t \in [0,1]$ and $I_0 \sim p_0$, and construct $I^\theta_t = (1-t) I_0 + t X^\theta_1$.
6: Construct gradient terms:
- $\nabla_\theta r(X^\theta_1)$ by direct differentiation
- $\nabla_\theta D$ using one of (8), (9), (11), using $\nabla \log p_t(I^\theta_t)$ and $\nabla \log q^\theta_t(I^\theta_t) = \frac{t v^\theta_{t,t}(I^\theta_t) - I^\theta_t}{1-t}$.
- $\nabla_\theta \mathcal{L}_{\text{sfm}}$ and $\nabla_\theta \mathcal{L}_{\text{cons}}$ using (14) and (15) (update flow map to fit self-induced q^θ_1).

7: Perform gradient step on θ using $\nabla_\theta \mathcal{L} = -\nabla_\theta r(X^\theta_1) + \nabla_\theta D + \nabla_\theta \mathcal{L}_{\text{sfm}} + \nabla_\theta \mathcal{L}_{\text{cons}}$.
8: **until** converged

The first two terms correspond to the reward-alignment objective itself – that is, improve reward, while remaining close to the base model. The last two terms ensure that the learned object is a flow map for its own generations, and thus in particular ensures we have access to the score function $\nabla \log q^\theta_t$ necessary for the regularization term. The combined training procedure is summarized in Algorithm 1, and an overview is given in Figure 1.

Multi-step generation Although we have presented the procedure above using the one-step map $X^\theta_{0,1}$, the same idea applies to any few-step generation $X^\theta_1 = X^\theta_{t_{K-1},t_K} \circ \cdots \circ X^\theta_{t_0,t_1}(Z)$, for $Z \sim p_0$. We note that, strictly speaking, the consistency term $\mathcal{L}_{\text{cons}}$ is not needed in the objective when using a single-step generator (as we only require the diagonal term to obtain the score function) but it is necessary when using multi-step generation. We include it here so that our learned object remains a flow map, and thus can be used for single- or multi-step generation in downstream tasks. This can be useful, for example, for inference-time steering applications (Sabour et al., 2025a; Potaptchik et al., 2026b; Holderrieth et al., 2026).

3.5 Exponential Reward Tilt

A common objective for finetuning flow-based models is to consider the KL-regularized objective defined on the terminal distribution, $\max_{q_1} \{\mathbb{E}_{X_1 \sim q_1}[r(X_1)] - D_{\text{KL}}(q_1 \| p_1)\}$, the solution of which is the exponentially-tilted distribution $q^*_1(x) \propto p_1(x) e^{r(x)}$.

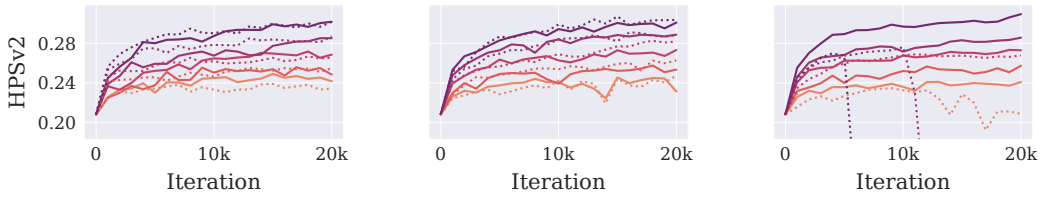
In general, the optimal terminal law q^*_1 for the integrated regularizer objective (4) depends on the choice of weight $w(t)$ and does not admit any such closed form according to the endpoint p_1. However, we note that for the particular case of the integrated Fisher divergence with the weight schedule $w(t) = \frac{1-t}{t}$, the optimal terminal law coincides with the exponentially-tilted terminal distribution. This choice of weighting schedule places most of the mass towards the start of the generative process, and we in fact found this to perform well in our experiments.

Proposition 3.4 (Exponential reward tilt). *Consider using the integrated Fisher divergence with weight $w(t) = \frac{1-t}{t}$. The optimal terminal law is*

$$q^*_1(x) \propto p_1(x) e^{r(x)}. \tag{17}$$

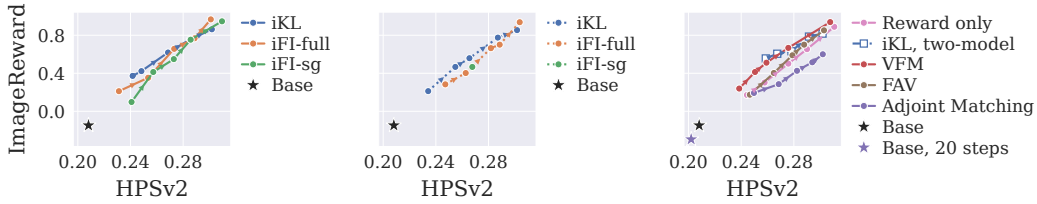
4 Experiments

To assess our proposed integrated regularization mechanisms, we will examine performance for image generation tasks for which flow-maps have shown success for fast generation. We focus our investigations on ImageNet 256×256 generation, which allows us to run detailed ablations and comparisons with existing methods due to the comparatively manageable training time. We then examine its scalability in a large-scale text-to-image setting reflective of current state-of-the-art applications by finetuning a flow-map distillation of FLUX (Black Forest Labs, 2024). In the Appendix, we also investigate applications for *de novo* protein generation.



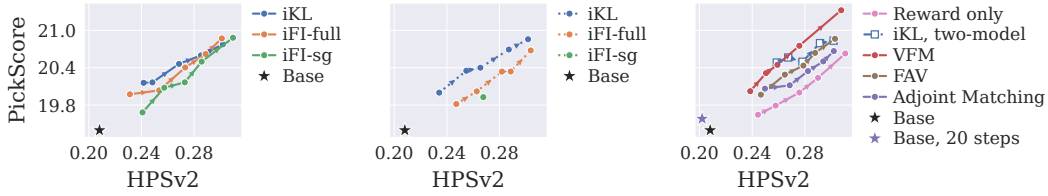
(a) iKL regulariser (b) iFI-full (c) iFI-sg

Figure 2: HPSv2 reward during fine-tuning for increasing reward strengths λ (darker is larger, λ values given in Table 4), using weight schedules $w(t) = \frac{1-t}{t}$ (solid —) and $w(t) = 1$ (dotted $\cdots$).



(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 3: ImageReward transfer when optimised for HPSv2 with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).



(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 4: PickScore transfer when optimised for HPSv2 with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).

ImageNet 256 We consider fine-tuning the one-step sampler of the pretrained flow map from Lee et al. (2025) using our three proposed integrated regularizers, which we denote **iKL**, **iFI-full**, and **iFI-sg**. To assess the effect of the integration weighting in the regulariser, we use both the constant weighting schedule $w(t) = 1$, and the 'memoryless' weighting schedule $w(t) = \frac{1-t}{t}$ which places significantly more mass towards the start of the probability flow path. We consider optimizing for the HPSv2 score (Wu et al., 2023), choose a range of reward strengths that obtain a similar range of final scores. Full details, and additional results including optimizing for alternate rewards ImageReward (Xu et al., 2023) and PickScore (Kirstain et al., 2023) are given in Appendix B.1.2.

In Figure 2, we plot the reward curves during training. We see that the proposed methods generally achieve stable training and exhibit the desired plateauing behaviour, suggesting that the regularization is effective in counteracting movement away from the base model. Generally, we found that the memoryless weighting trained more stably than the constant schedule, which we anticipate is a consequence of the greater smoothness of the marginal densities earlier along the generation path. This is supported by further investigations in Appendices B.1.2 and B.1.3. We also see that the combination of iFI-sg with the constant schedule often collapsed, which aligns with the fact that iFI-sg does not optimise a fixed variational objective.

In Figures 3 and 4 we report how the alternative rewards ImageReward and PickScore change as the HPSv2 score increase with reward strength, as an indication of whether the generations are truly improved rather than 'reward-hacking'. We see that optimizing for HPSv2 improved the other scores too, and the iKL regularizer generally gave the best transfer. When trained successfully, both the weighting schedules (left and middle plots) gave similar transfer ability. In the rightmost plots, we report results for ablated versions (no regularization, and a two-model version), as well as alter-



Figure 5: 4-step samples from the base Flux flow map and after fine-tuning with each regularizer, using $w(t) = \frac{1-t}{t}$.

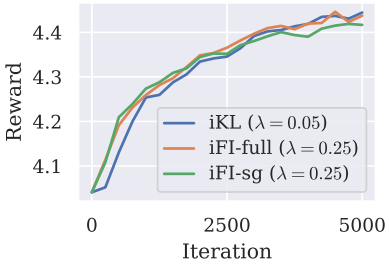


Figure 6: FLUX reward curves.

Table 1: FLUX GenEval score.

Base	iKL	iFI-full	iFI-sg
0.627	0.685	0.681	0.661

native few-step reward alignment approaches FAV (Lee et al., 2026) and VFM (Mammadov et al., 2026b). We also include results for Adjoint Matching using 20 discretisation steps, as a representative continuous-time approach. As above, we choose hyperparameters to span comparable HPSv2 ranges. We see that the transfer ability of our approach, particularly when using iKL, compares favourably with these methods. Additional results and more detailed discussion and comparisons are provided in the Appendix.

Flux To examine whether our approach scales to large-scale text-to-image generation, we now consider fine-tuning a flow-map distillation of FLUX.1-dev. Following Eyring et al. (2024), we use a linear combination of HPSv2, ImageReward, PickScore, and CLIP as our reward, and we use $w(t) = \frac{1-t}{t}$. Resulting generations are visualised in Figure 5, and the reward curves during training are plotted in Figure 6. Again, the curves show stable training and subsequent plateauing, indicating our proposed regularization mechanisms are effective in this large-scale setting too. We also report how the GenEval score (Ghosh et al., 2023) changes after finetuning in Table 1, and see the improvement transfers to this score too despite it not being part of the optimised reward.

Protein Generation In Appendix C, we also investigate applications for *de novo* protein generation, in which we fine-tune a Riemannian flow map according to a secondary structure reward.

5 Discussion

Limitations Our approach requires gradients of a reward function, so is not suitable for applications that require non-differentiable rewards. We also focus on reward-aligning flow-maps specifically, compared to other approaches (such as Lee et al. (2026)) that work for any few- or single-step generators. We also require that the score function remains close to that of the current self-induced terminal distribution q_t^θ, and lagging could cause issues for the optimization; we note however that we did not observe this to be an issue in our experiments. We noticed that our approach can be somewhat sensitive to the scaling of the consistency term, which sometimes caused the multi-step generation to deteriorate relative to the single-step generation. We found this could be addressed by suitable choices of hyperparameters, and examine this in more detail in Appendix B.1.3.

Conclusion In this work, we have introduced a method for reward-aligning accelerated flow models that enables direct optimization of the few-step sampler, while retaining a continuous-flow regularization mechanism along the probabilistic path. By considering flow maps, a single object provides access to both the accelerated sampler and the necessary intermediate-time quantities required for regularizing according to integrated KL and Fisher divergences. Our results indicate that optimizing flow maps using integrated regularization combines the efficiency of few-step generation with the stability afforded by regularization along the continuous probability flow path.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing code and making plots, to polish writing, to check for additional relevant literature, and to review mathematical results. We did not use AI for developing the theory or methodology. The remaining necessary disclosure tasks are not relevant to this work. We have reviewed all AI-assisted work, by checking any AI-assisted code and conducting our own manual literature review. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Detailed descriptions of the experimental setups, along with the reasons for choosing them, are given in Appendices B and E. In particular, details regarding the base models used, and the resulting fine-tuning setups are given in Appendix B, and all reward strengths λ used in the experiments are provided in Table 4. We also provide detailed descriptions of the baselines that we compare against in Appendix B, and provide links and licenses for all assets used in Appendix F.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2022.

Michael S. Albergo, Mark Goldstein, Nicholas M. Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. *arXiv:2310.03725*, 2023.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models, 2026.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow Map Matching: A unifying framework for consistency models. *arXiv:2406.07507*, June 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *arXiv preprint arXiv:2505.18825*, 2025.

Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2024.

John-Marc Chandonia, Lindsey Guan, Shiangyi Lin, Changhua Yu, Naomi K Fox, and Steven E Brenner. Scope: improvements to the structural classification of proteins – extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559, 01 2022.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İ Ceylan, Michael Bronstein, and Avishek J Bose. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084, 2024.

Oscar Davis, Michael Samuel Albergo, Nicholas Matthew Boffi, Michael M. Bronstein, and Joey Bose. Generalised flow maps for few-step generative modelling on riemannian manifolds. In *The Fourteenth International Conference on Learning Representations*, 2026.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. 2026.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Conference on Neural Information Processing Systems*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Stephanie Fu, Netanel Yakir Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. *arXiv preprint arXiv:2512.02012*, 2025b.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Jiaqi Han, Puheng Li, Qiushan Guo, Renyuan Xu, Stefano Ermon, and Emmanuel J Candès. One-step generative modeling via wasserstein gradient flows. 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, volume 33, pp. 6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *Advances in Neural Information Processing Systems*, 35:8633–8646, December 2022.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps. In *Forty-third International Conference on Machine Learning*, 2026.

Samuel Howard, Nikolas Nüsken, and Jakiw Pidstrigach. Control consistency losses for diffusion bridges. In *Forty-third International Conference on Machine Learning*, 2026.

Jerry Yuyang Huang, Justin Lin, Sheel Shah, Kartik Nair, and Nicholas Matthew Boffi. How to guide your flow: Few-step alignment via flow map reward guidance. In *Forty-third International Conference on Machine Learning*, 2026.

Dengyang Jiang, Dongyang Liu, Zanyi Wang, Qilong Wu, Liuzhuozheng Li, Hengzhuang Li, Xin Jin, David Liu, Changsheng Lu, Zhen Li, Bo Zhang, Mengmeng Wang, Steven Hoi, Peng Gao, and Harry Yang. Distribution matching distillation meets reinforcement learning, 2026.

Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion. *arXiv:2310.02279*, 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation, 2023.

Jaewoo Lee, Hyeongyu Kang, Dohyun Kim, Kyuil Sim, Woocheol Shin, Minsu Kim, Taeyoung Yun, Jeongjae Lee, Sanghyeok Choi, Tabitha Edith Lee, Jong Chul Ye, and Jinkyoo Park. Aligning few-step generative model via amortizing sample-based variational inference. In *ICML 2026 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2026.

Kyungmin Lee, Sihyun Yu, and Jinwoo Shin. Decoupled meanflow: Turning flow models into flow maps for accelerated sampling. *arXiv preprint arXiv:2510.24474*, 2025.

Zhiqi Li, Wen Zhang, and Bo Zhu. Flow-map grpo: Reinforcement learning for few-step flow-map generators via anchored stochastic composition, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhen Liu, Tim Z. Xiao, Carles Domingo-Enrich, Weiyang Liu, and Dinghuai Zhang. Value gradient guidance for flow matching alignment. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv:2410.11081*, 2025.

Weijian Luo. Diff-instruct++: Training one-step text-to-image generator model to align with human preferences. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Weijian Luo, Zemin Huang, Zhengyang Geng, J Zico Kolter, and Guo-Jun Qi. One-step diffusion distillation through score implicit matching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Weijian Luo, colin zhang, Debing Zhang, and Zhengyang Geng. David and goliath: Small one-step model beats large diffusion with score post-training. In *Forty-second International Conference on Machine Learning*, 2025.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025. URL https://arxiv.org/abs/2501.09732.

Abbas Mammadov, Jerry Y. Huang, Justin Lin, Partha Kaushik, Sheel Shah, Kartik Nair, Yee Whye Teh, and Nicholas M. Boffi. Wtf?! simulation-free reinforcement learning with wasserstein-tilted flow maps, 2026a.

Abbas Mammadov, So Takao, Bohan Chen, Ricardo Baptista, Morteza Mardani, Yee Whye Teh, and Julius Berner. Variational flow maps: Make some noise for one-step conditional generation, 2026b.

Stefano Peluchetti. Non-denoising forward-time diffusions. 2023.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

Peter Potaptchik, Lee Cheuk Kit, and Michael Samuel Albergo. Tilt matching for scalable sampling and fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026a.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael Samuel Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. In *Forty-third International Conference on Machine Learning*, 2026b.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Amirmojtaba Sabour, Michael S. Albergo, Carles Domingo-Enrich, Nicholas M. Boffi, Sanja Fidler, Karsten Kreis, and Eric Vanden-Eijnden. Test-time scaling of diffusions with flow maps, 2025a.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Max W. Shen, Mark Goldstein, Zichu Wang, Aahlad Puli, and Rajesh Ranganath. How i learned to stop worrying and love stopgrads: Stationarity, convergence, and a case study on flow map learning, 2026.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. 2024.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, July 2011. ISSN 0899-7667.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Dongyeop Woo, Marta Skreta, Seonghyun Park, Sungsoo Ahn, and Kirill Neklyudov. Riemannian meanflow, 2026.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 15903–15935. Curran Associates, Inc., 2023.

Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with se(3) flow matching. In *Proceedings of Neural Information Processing Systems Workshop: Machine Learning in Structural Biology*, 2023.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024b.

Ziyi Zhang, Sen Zhang, Yibing Zhan, Yong Luo, Yonggang Wen, and Dacheng Tao. Confronting reward overoptimization for diffusion models: A perspective of inductive and primacy biases. In *International Conference on Machine Learning*, 2024.

Zichen Zhong, Haoliang Sun, Yukun Zhao, Yongshun Gong, and Yilong Yin. Riemannian meanflow for one-step generation on manifolds, 2026.

Linqi Zhou, Mathias Parger, Ayaan Haque, and Jiaming Song. Terminal velocity matching. *arXiv preprint arXiv:2511.19797*, 2025.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

APPENDIX

The Appendix is structured as follows. In Appendix A, we given an extended discussion of related work. In Appendix B, we provide details and extended discussion of the experiments in the main body of the paper, along with additional comparisons and ablations. In Appendix C we investigate applying our method in the Riemannian setting for reward-aligning few-step models for *de novo* protein generation. In Appendix D we provide proofs for the results in the paper, and in Appendix E we display a comparison of the generated samples for each of our proposed integrated regularizer mechanisms. Finally, in Appendix F we provide the licenses for the assets used in this work.

A RELATED WORK

Fine-Tuning Flow Models Diffusion (Song et al., 2020; Ho et al., 2020) and flow-based generative models (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022; Liu et al., 2022; Peluchetti, 2023) have achieved state of the art performance across a wide range of modalities including images (Rombach et al., 2022), proteins (Watson et al., 2023), and videos (Ho et al., 2022). However, pretrained models sample from a fixed learned distribution, whereas practitioners increasingly wish to adjust the outputs according to chosen preferences, perhaps to improve sample quality or to encourage desirable properties in the generations.

Motivated by this, there has been substantial recent work devoted to fine-tuning such models, including Xu et al. (2023); Black et al. (2024); Clark et al. (2024); Domingo-Enrich et al. (2025); Liu et al. (2025); Potaptchik et al. (2026a); Bergmeister et al. (2026) to name a few. While effective, finetuning many-step models is typically computationally expensive due to the requirement for simulating the ODE or SDE trajectories, which poses issues for scalability and training speed.

Few-Step Flow Models A largely separate literature has focused on mitigating the bottleneck that is sampling speed in flow-based models, by distilling the full models into few or one-step generators. One line of works aims to train a one-step generator that matches the distribution of the pretrained flow model, using the scores of the base model and that of the current accelerated sampler. Score Distillation Sampling (Poole et al., 2023) uses a KL-based distillation objective, and subsequent works Variational Score Distillation (Wang et al., 2023) and Diff-Instruct (Luo et al., 2023) optimize according to the integrated KL divergence that we use in (8). Distribution Matching Distillation (DMD) (Yin et al., 2024b;a) use a similar formulation and include an additional regression and adversarial loss respectively. In contrast to these KL-based objectives, Score Identity Distillation (Zhou et al., 2024) uses a Fisher divergence objective, and Score Implicit Matching (Luo et al., 2024) generalizes this to use a broad family of divergences between a teacher score and the score of the current accelerated generator, providing tractable expressions for the gradients through the implicitly defined intermediate distributions, which we use in (11).

While the above line of works aim to accelerate inference by matching at the distributional level, a second approach considers distillation at the level of the individual trajectories. Consistency models (Song et al., 2023; Song & Dhariwal, 2024; Lu & Song, 2025) aim to learn the mapping to the terminal point from points along the trajectory. Flow maps (Boffi et al., 2024; 2025; Kim et al., 2024) generalize this construction by learning the map between arbitrary points along the ODE trajectory, and introduce Eulerian, Lagrangian, and semigroup losses to enforce this property. Several highly performant methods can be understood under this framework, including Shortcut Models (Frans et al., 2025), MeanFlow (Geng et al., 2025a;b), Align your Flow (Sabour et al., 2025b), and terminal velocity matching (Zhou et al., 2025) to name a few, and understanding the design space of such approaches is a highly active area of current research.

Given the increased adoption of accelerated flow-based generative models, this raises the prospect of eliminating the sampling bottleneck that arises in the fine-tuning methods discussed above. However, by sidestepping the sequential generation procedure it becomes unclear how to leverage the probabilistic structure in the regularization mechanism; this is the problem we tackle in this work. A few recent works have studied this same problem; in particular, it remains an active research question as to how to fine-tune a flow map while preserving its ability to transport between arbitrary times. Tilt matching (Potaptchik et al., 2026a) is applicable to both many-step and few-step generation, so could be applied for fine-tuning flow maps. Variational Flow Maps (Mammadov et al., 2026b)

learn a noise adapter in a combined flow map training procedure. The authors investigate applications to inverse problems, and also show that the method is effective for reward alignment too. Li et al. (2026) leverage within-batch relative rewards for reward alignment. We also highlight recent work Mammadov et al. (2026a), who similarly to us consider fine-tuning flow maps, but instead utilize an optimal transport regularizer. Some methods utilize comparisons between samples, often in a well-chosen feature space using similarity kernels, which are applicable to any one or few-step generators. For example, FAV (Lee et al., 2026) aligns few-step generative models by comparing samples from the current and base generators using similarity kernels, and training according to a fixed point based on Stein variational gradient descent dynamics.

A particularly relevant line of works seeks to reward-align one or few-step generators using integrated regularizers, similar to the objective we consider. DiffInstruct++ (Luo, 2025) utilized the same integrated KL divergence regularizer that we consider, and optimize by training an auxiliary score network similar to the distillation methods previously discussed. Follow-up work Luo et al. (2025) extends this to wider classes of integrated score-based divergences, similar the the stop-grad Fisher information regularizer that we consider as one of our options. Jiang et al. (2026) consider combining DMD with reinforcement learning, to guide the distilled generations to align with human preferences in a combined training procedure. As in the distillation literature, these methods use two separate models for the generator and score networks. In contrast, in this work we consider a single flow-map parameterization, allowing us to use a single set of trainable parameters and obtain a coherent reward-aligned flow-map model that can be used for any-step generation. This is often important for downstream tasks, for example for inference-time steering applications (Sabour et al., 2025a; Potaptchik et al., 2026b; Holderrieth et al., 2026).

B Experimental Details and Additional Results

B.1 ImageNet

Experimental Details For the ImageNet 256×256 experiment, we use a well-trained flow map checkpoint DMF XL/2+ (Lee et al., 2025) as the base model, which provides good quality single-step generations. We consider finetuning this model across all ImageNet classes using the prompt *"A high-quality, high-resolution photograph of a* $\{$*class*$\}$*."*. We use a batch size of 16 and train for 20,000 steps using the Adam optimizer with a learning rate 10^{-5}. All ImageNet experiments were ran on a single GH200 GPU.

For the results in Section 4, we consider fine-tuning according to the HPSv2 reward model (Wu et al., 2023), and report the HPSv2 reward training curves along with other standard reward and perceptual metrics ImageReward (IR) (Xu et al., 2023) and PickScore (Kirstain et al., 2023) for different reward strengths λ. Below, we report analogous results when optimising for the other rewards instead.

In each experiment, we run for 5 reward strenths λ. These are each chosen so that the methods span a similar range of the optimized reward to allow for comparison between the methods (chosen to be approximately 0.24-0.3 for HPSv2, 0.4-1.2 for ImageReward, and 20.5-22.0 for PickScore). The reward strengths for every experiment are given in Table 4, and the runtimes and memory requirements in Table 2. In these experiments, unless otherwise specified we use the semigroup consistency loss scaled by 0.01, which we found to give good performance. We examine the dependence on the consistency loss scaling in additional experiments below.

A comparison of samples for each of the regularization methods and weighting schedules, for each optimized reward function, is given in Figures 22 to 24.

Evaluation To assess performance when optimising for a particular chosen reward function, it is not appropriate to measure success by the optimised reward value alone, as this could conceal unstable optimization and 'reward-hacked' samples. Rather, we care about achieving stable training, and also that the obtained samples are genuinely improved rather than 'reward-hacked' for the particular reward function being optimized. We therefore design our analysis to assess these points.

To assess the training stability, we can examine the reward curve during training - unstable training would manifest as jagged curves or even collapsed runs, whereas stable training and effective regularization would give a smooth, plateauing curve. Assessing the quality of the generations is

somewhat subjective; in this work, we plot how the values of the other reward functions change as the the optimized reward value increases with increasing reward strength; this matches similar reporting conventions in other works, such as Zhang et al. (2024). A 'good' sample should score highly in the other rewards too, whereas 'reward-hacked' images are more likely to score poorly for the alternate reward functions. In our experiments, we report metrics evaluated using 512 generated samples, with the classes sampled uniformly from the 1000 ImageNet classes.

Following papers such as Domingo-Enrich et al. (2025); Lee et al. (2026), we also assess the diversity of the resulting generations by computing average pairwise distances for DreamSim features (Fu et al., 2023). To compute this, we use 8 samples from 64 conditioning classes. Within each class, one minus the cosine similarity of the DreamSim features is computed for each pair, and then these values are averaged over the classes. Diversity results are presented in Appendix B.1.3.

B.1.1 Baselines

Here, we describe the baselines and ablated versions that we compare to in our plots. Generally, to implement the alternative methods we utilize the setups from their official implementations unless otherwise stated, and again choose hyperparameters so that the 5 runs span similar ranges of the optimised rewards.

No regularizer The first comparison is with optimizing the reward of the single-step model directly, without any regularization term constraining movement away from the base model (that is, using $\mathcal{L}_{\text{reward}}$ only). As expected, when ran with equivalent settings to our runs this resulted in the reward increasing until reward-hacking. In the figures, we therefore report results for this method in which the learning rate has been decreased to low levels so that the optimization reaches the desired range after 20,000 steps.

From the plots, it is clear that even when deliberately slowing the optimization to avoid reward-hacking, the transfer to the alternate rewards is worse than for other methods, including our proposed approached.

Two-model version As discussed in Section 3.3, one could perform integrated regularization using two models: one as the few-step generator, and a second auxiliary network to fit the score functions. Indeed, such two-model approaches have been highly successful for distilling diffusion models into few-step generators (Wang et al., 2023; Luo et al., 2023; Yin et al., 2024b;a; Zhou et al., 2024; Luo et al., 2024). They have also been utilised for reward-aligning few-step models (Luo, 2025; Luo et al., 2025; Jiang et al., 2026), and indeed this two-model variant is analogous to these methods. We instead consider optimizing a single model that provides access to these both objects, while also resulting in a coherent flow model which can be useful for downstream tasks. We therefore include a comparison against the analogous two-model version for the same optimizations. Our aim is therefore to achieve similar performance to this strong baseline, to indicate that one can use only a single model with minimal impact on performance. We provide a detailed comparison to this baseline in Appendix B.1.3.

FAV FAV (Lee et al., 2026) is a recent method for reward-aligning few- or one-step generative models by using dynamics determined by finite-sample kernel approximations to densities, usually in a suitable feature space, similar in spirit to the recent drifting-model framework (Deng et al., 2026; Han et al., 2026). As such, it lies in a somewhat different class to our approach since it requires a well-chosen feature space and kernel in order to learn from comparisons between samples. In contrast, our approach operates in the ambient space using the probability flow and compares scores rather than kernel density estimates.

Following the official FAV implementation (see Appendix F), we use 16 generations from the base and current fine-tuned model per class, and then use 8 classes for each gradient update. We optimize for 2,500 steps to match the number of samples from the fine-tuned model as used during training with our method. We use the same optimizer settings as in our method, which we found gave a similar plateau effect to our results. Following the paper, we use a Gaussian kernel in the feature space with a bandwidth of 0.3. We note that we only used FAV for optimizing for HPSv2 and PickScore, as here the choice of feature representations to use in the similarity kernel is clear (namely, the output of the CLIP image encoder used in the reward model itself).

Generally we found that FAV consistently performed well, giving stable training and effective regularization relative to the base model as indicated by plateauing reward curves during optimization. The reward increase also consistently transferred to other rewards, generally giving results fairly similar to our iFI-full regulariser. As this method relies on finite-sample kernel approximations to densities in feature space, it is possible that these results could be changed through different choices of kernel, feature space, and different numbers of samples in the kernel approximation. We remark that FAV gave noticably better diversity for comparable values of optimized reward than the other approaches (see Appendix B.1.3 below).

VFM For VFM (Mammadov et al., 2026b), we again follow the setup used in the official implementation. We note that this uses the EMA weights rather than the raw weights which follows the official implementation (we found that this helped to avoid reward-hacking), however this means that the results are not directly comparable. The VFM results show strong reward transfer when optimized for HPSv2, however it generally did not appear to exhibit as clear a plateau as for FAV or our integrated regularizer mechanisms. As we used the setup from the paper for training VFM in order to avoid reward hacking, the timings are not directly comparable due to the larger batch size. Nevertheless, VFM remains a useful comparison for reward transfer and training stability when each method is run under its recommended setup.

Adjoint Matching As a representative continuous-time finetuning approach, we also compare to adjoint matching (Domingo-Enrich et al., 2025). We use 20 discretisation steps when generating samples, and use a replay buffer consisting of 100 trajectories which we then used for training 8 gradient steps. As such continuous methods are significantly slower due to the need for ODE or SDE rollouts, we train for 5,000 optimization steps which gave a comparable plateau to our approach when optimizing the HPSv2 loss, though this took significantly longer than the few-step appraoches as seen in Table 2.

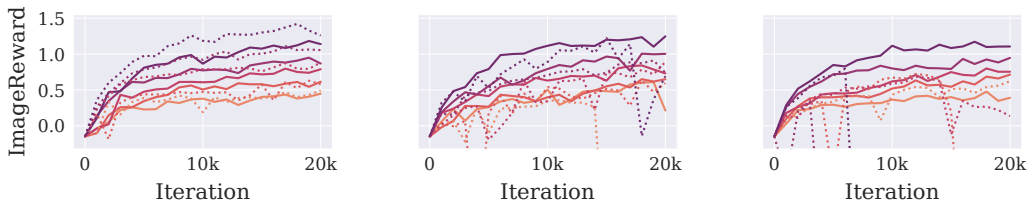
We found the results for adjoint matching to be lower than for the few-step methods when optimized for HPSv2, which we attribute to the reward gradient signal being indirectly propagated through the fixed point procedure rather than being optimized directly. It is also possible that using 20 steps for the discretisation is coarser than ideal, and using more steps might improve results at the cost of more expensive training. It did however perform well when optimized for PickScore, and ImageReward in particular. We remark that we also experimented with other forms of fixed-point consistency training (see Howard et al. (2026)), but found that the standard form of adjoint matching performed strongest in this generative modelling setting.

B.1.2 Optimising for Alternative Rewards

Figures 2 to 4 in the main body show the reward curves and reward transfer ability of our proposed integrated regularizer mechanism, along with that of ablated versions and alternative methods, when being optimized for the HPSv2 reward (Wu et al., 2023). In Figures 7-12 we show analogous plots to those in the main body, but instead optimising for ImageReward (Xu et al., 2023) and PickScore (Kirstain et al., 2023) respectively.

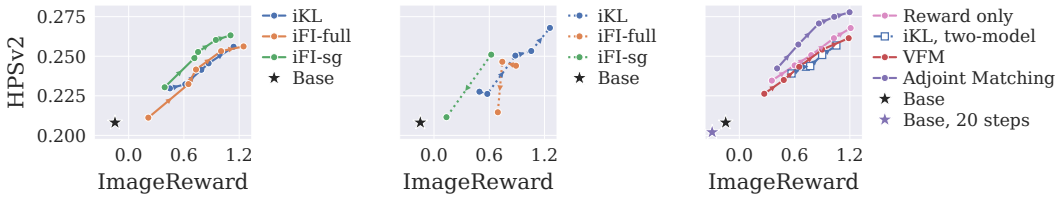
As before, we observe plateauing reward curves during training, and again see that the memoryless schedule gives more stable training than the constant schedule - indeed, using the constant schedule again often causes training to fail when using iFI-sg (as discussed previously, this reflects that this choice does not optimize a fixed objective). We also saw occasional failures with iFI-full and one instance of reward hacking with iKL when using the constant weight schedule, supporting the idea that stronger regularization towards the start of the generative process gives better stability.

Again, increasing the optimised reward generally increases the other rewards too. Interestingly, when optimising ImageReward, the iFI-sg regulariser often appeared to give better transfer to the other rewards in this case (though as discussed, it is generally a more unstable choice of objective). We remark that there is not always a uniform best-performing method across all the experiments; for example, the adjoint matching baseline appears to give better transfer to the HPSv2 score in this case. Overall, across all experiments we found the most consistent results were given using integrated regularization using iKL in both the single-model and analogous two-model version (particularly when using $w(t) = \frac{1-t}{t}$), as well as the kernel-based method FAV (Lee et al., 2026).



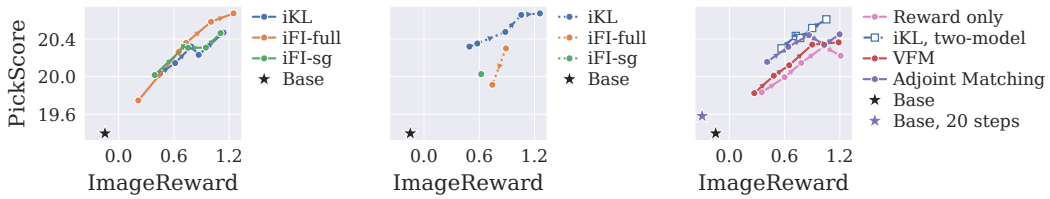
(a) iKL regulariser (b) iFI-full (c) iFI-sg

Figure 7: **ImageReward:** Reward during fine-tuning for increasing reward strengths λ (darker is larger, λ values given in Table 4), using weight schedules $w(t) = \frac{1-t}{t}$ (solid —) and $w(t) = 1$ (dotted $\cdots$).



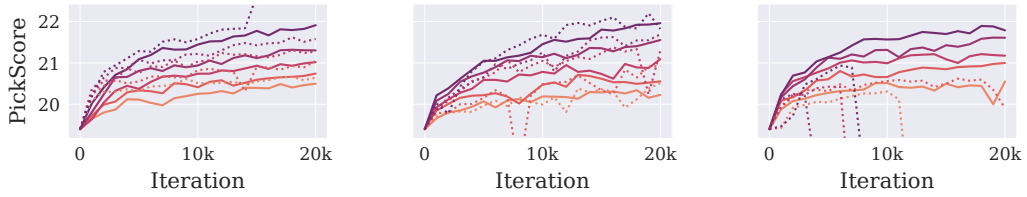
(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 8: **ImageReward:** HPSv2 transfer when optimised for ImageReward with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).



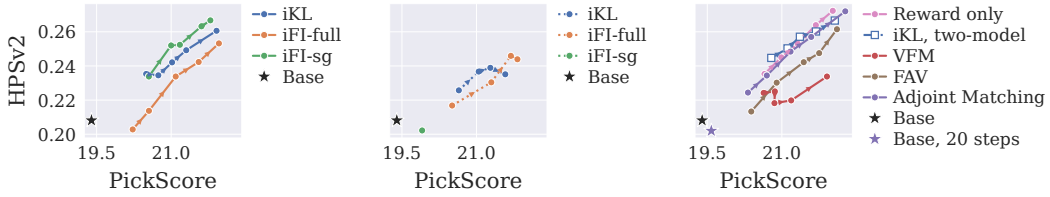
(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 9: **ImageReward:** PickScore transfer when optimised for ImageReward with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).



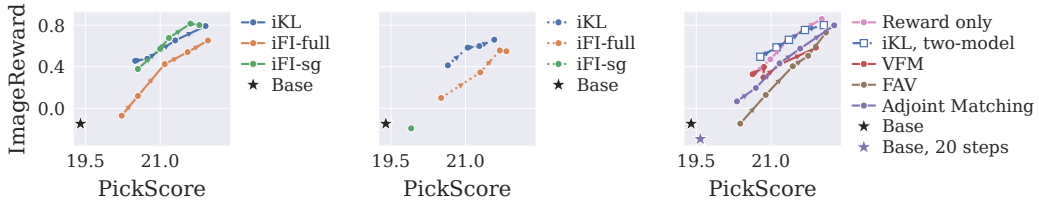
(a) iKL regulariser (b) iFI-full (c) iFI-sg

Figure 10: **PickScore:** Reward during fine-tuning for increasing reward strengths λ (darker is larger, λ values given in Table 4), using weight schedules $w(t) = \frac{1-t}{t}$ (solid —) and $w(t) = 1$ (dotted $\cdots$).



(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 11: **PickScore:** HPSv2 transfer when optimised for PickScore with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).



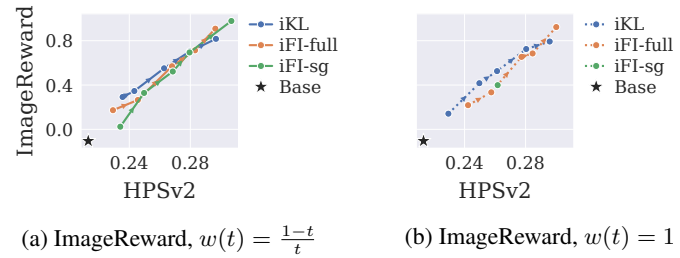
(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 12: **PickScore:** ImageReward transfer when optimised for PickScore with increasing reward strengths, using for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).

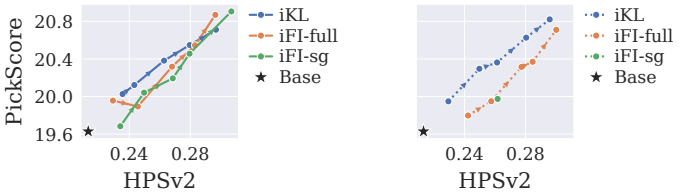
B.1.3 Ablations and Detailed Analysis

We now focus on ablations and specific analysis of important factors of the proposed approach. For these experiments, we focus on optimizing for the HPSv2 score as done in the experiments in the main body, and use the semigroup consistency loss with scaling 0.01 unless otherwise stated.

Few-Step Evaluation In our approach, we aim to retain the any-step generation ability of the flow-map during the fine-tuning procedure. To test whether optimizing one-step generations also benefits multi-step generation, we evaluate the fine-tuned models using multi-step generation, despite them having been fine-tuned using 1-step samples. In Figure 13, we report similar plots but instead evaluated using 4-step generations for the HPSv2 experiments in the main body. The 4-step generations similarly improve and plateau like the 1-step maps, indicating the the reward-alignment is transferred to the whole flow-map even though only the 1-step generation was used during training. The comparisons show a similar pattern to the 1-step results.



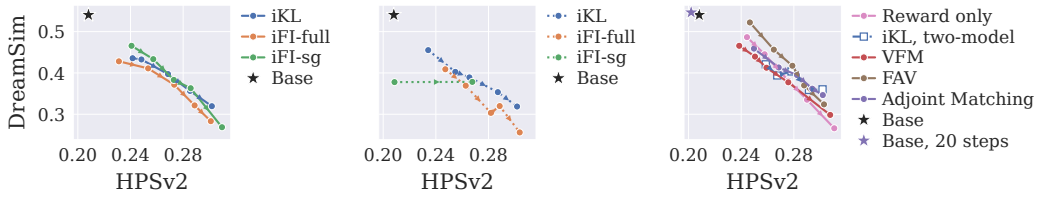
(a) ImageReward, $w(t) = \frac{1-t}{t}$ (b) ImageReward, $w(t) = 1$



(c) PickScore, $w(t) = \frac{1-t}{t}$ (d) PickScore, $w(t) = 1$

Figure 13: **Few-step generation also improves:** Reward transfer using 4-step generation, when trained for 1-step generation. Results for the iKL, iFI, and iFI-sg regularisers, optimised for HPSv2, under the weight schedules $w(t) = \frac{1-t}{t}$ (memoryless) and $w(t) = 1$ (constant).

Diversity Reward fine-tuning often trades sample diversity for reward, and a collapsed run can score well on every reward metric while producing near-identical images. Following Domingo-Enrich et al. (2025), in Figure 14 we therefore also plot the DreamSim diversity metric as the reward strengths increase. We generally see that most methods perform fairly similarly, though using $w(t) = \frac{1-t}{t}$ is noticably better than using $w(t) = 1$, and of the integrated regularizers, iKL appears to perform the best. We remark that FAV appears noticably stronger for diversity than the other methods.

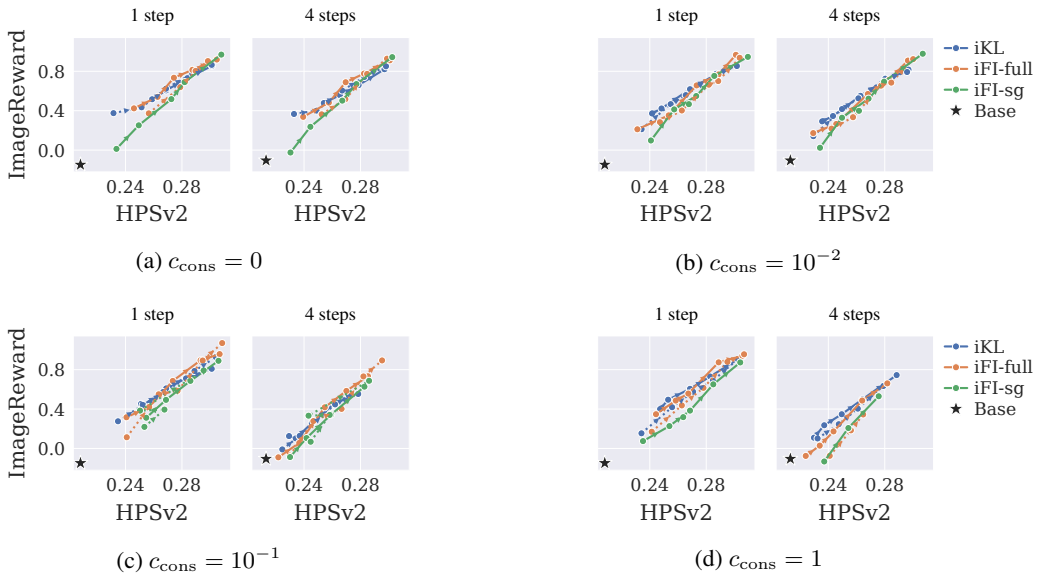


(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1$ (c) Method comparison

Figure 14: **Diversity:** DreamSim diversity (mean pairwise DreamSim distance between 8 samples of the same class, averaged over 64 classes; higher is more diverse) when optimized for HPSv2, using weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).

Consistency Loss Ablations We now consider the sensitivity of our approach to the scaling of the consistency loss. In Figure 15, we plot the reward transfer curves for both the 1 and 4-step generations, to examine whether the 4-step performance remains similar to that of the 1-step generations that we are optimizing according to. We run for scaling coefficients 0.01, 0.1, and 1.0, as well as with no $\mathcal{L}_{\text{cons}}$ term, denoting this term c_{cons}. Somewhat surprisingly, we found that increasing the scaling to 0.1 and 1.0 began to degrade the 4-step generations rather than aiding them, suggesting that too large a weight can add instability to the training. We anticipate that this could be mitigated by incorporating adaptive weights or gradient surgery, and note that understanding and improving the behaviour of such fixed point training is a active area of research (Shen et al., 2026).

More surprisingly, we found that the 4-step generations remained good even without including the consistency term $\mathcal{L}_{\text{cons}}$ at all, despite the fact that there is nothing to keep the model a coherent flow map. We anticipate this is a consequence of starting with a well-trained flow map, and that this multi-step ability may begin to deteriorate with more training. Based on these results, we used a scaling factor of 0.01 in our experiments, which provided good single- and multi-step performance while actively encouraging the model to retain the flow-map structure.



(a) $c_{\text{cons}} = 0$ (b) $c_{\text{cons}} = 10^{-2}$

(c) $c_{\text{cons}} = 10^{-1}$ (d) $c_{\text{cons}} = 1$

Figure 15: **Comparison of Consistency Loss Scaling:** ImageReward transfer when optimised for HPSv2 using various consistency coefficients for the semigroup consistency loss, evaluated with 1 and with 4 sampling steps. Comparison of iKL, iFI, and iFI-sg regularisers, for weight schedules $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).

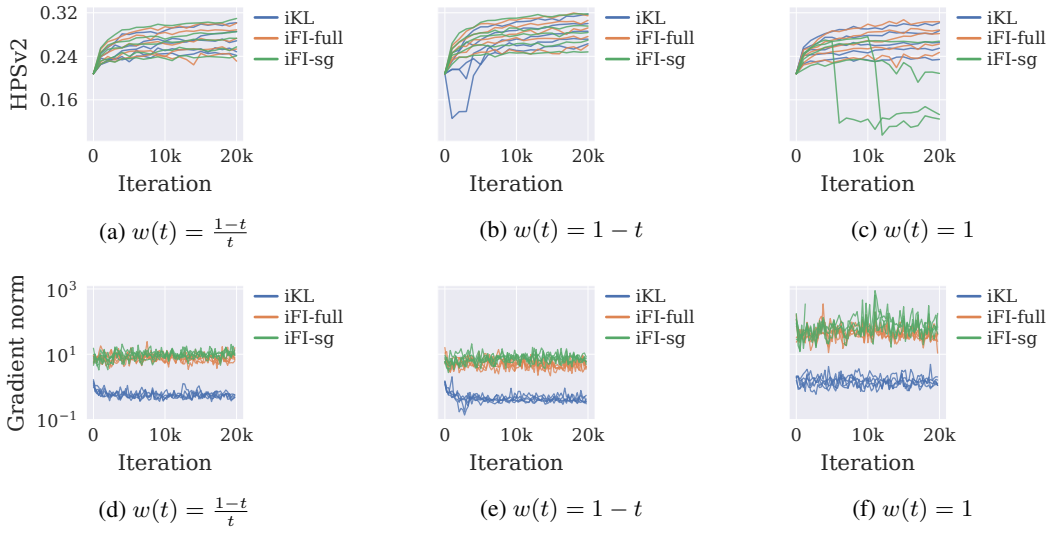
Ablation over Regularizer Integrated Range Recall from Section 3.1 that we motivated using regularizers that integrate along the probability flow path, because the improved regularity of the intermediate marginals p_t and q_t^θ offer the promise of more stable training.

Here, we now examine to what extent this intuition holds in practice, by comparing three possible weighting schedules that place the integrated mass at different places along the generative trajectory. The first that we consider is the so-called *'memoryless'* schedule $w(t) = \frac{1-t}{t}$, which places most of the mass towards the start of the generative process. As a middle ground, the second option is a linearly decaying weighting function $w(t) = 1 - t$, which again places more weight towards the start but in a less extreme manner. Finally, we also consider the constant weight schedule $w(t) = 1$, which places the mass evenly along the generative trajectory.

In the top row of Figure 16, we show all the reward curves during training (for each strength, with the different regularization methods indicated by different colours). These results clearly show that the regularisers that place more weight towards the start give more stable training. All the runs using $w(t) = \frac{1-t}{t}$ trained successfully with smooth reward curves. The runs with $w(t) = 1 - t$ also all trained successfully but resulted in noticeably more jagged reward curves, with a few that temporarily collapsed before recovering during training. The constant schedule $w(t) = 1$ again gave more jagged curves, with several collapsing completely (though these were generally for the iFI-sg regulariser, which as discussed does not do direct optimisation on a fixed objective and appears to be more unstable as a result).

We can also relate the collapsing runs to the gradient norms and their variance, visible on the bottom row of Figure 16. We see that the weighting schedules that place more mass towards the start have significantly lower gradient norms and lower variances, which may explain the improved stability. It is also worth noting that iKL gives much smaller gradient norms than the two Fisher information regularizers.

These results support the motivation that comparing marginals earlier in the trajectory gives more stable training while till providing effective regularisation behaviour. They also suggest that using iKL with the memoryless schedule is the most stable choice among those we tested, which aligns with our experience when running the experiments.



(a) $w(t) = \frac{1-t}{t}$ (b) $w(t) = 1 - t$ (c) $w(t) = 1$

(d) $w(t) = \frac{1-t}{t}$ (e) $w(t) = 1 - t$ (f) $w(t) = 1$

Figure 16: **Comparing Integrated-Regularisation Range:** *(Top row)* Reward curves when optimizing for HPSv2, using the three weight schedules $w(t) = \frac{1-t}{t}$, $w(t) = 1 - t$ and $w(t) = 1$, to examine stability during training. *(Bottom row)* Gradient norms of the various integrated regularizer terms $\nabla_\theta \mathcal{L}_{\text{reg}}$, for each weight schedule $w(t) = \frac{1-t}{t}$, $w(t) = 1 - t$ and $w(t) = 1$.

Detailed Comparison with Two-Model Version In our approach, we consider using a flow-map as it provides both a fast generator, and access to the score functions along the probabilistic path needed for the integrated regularization. One could in fact parameterize this with two separate models which are both trained at the same time. Indeed, this is often done in the distillation literature (Wang et al., 2023; Luo et al., 2023; Yin et al., 2024b;a; Zhou et al., 2024; Luo et al., 2024), and has also recently been utilised for reward-alignment (Luo, 2025; Luo et al., 2025; Jiang et al., 2026) However such methods necessitate training two large models (the generator, and an auxiliary score network) and do not result in a flow-based model that can be used for any-step generation, which can be useful in downstream tasks (for example, in applications that use inference-time steering (Sabour et al., 2025a; Potaptchik et al., 2026b; Holderrieth et al., 2026)). Our aim in this paper is to optimize only a single set of trainable parameters under the flow-map parameterization. It is therefore instructive to compare against this two-model variant as a strong baseline analogous to methods like Luo (2025) and Luo et al. (2025), to which we want to attain similar performance using only a single model.

The performance of the two-model version has been included in several of the plots. In Figure 17 we provide a more detailed comparison of for each regularization mechanism, for the weight schedules $w(t) = \frac{1-t}{t}$ and $w(t) = 1$. Generally, the performance difference between the two-model version and the single flow map is minimal, indicating that little performance is lost by restricting to a single model. For the iKL, the two-model version seems to give slightly stronger transfer to the other reward functions, but for the iFI-full this is approximately equal, and for iFI-sg the single-model version with $w(t) = \frac{1-t}{t}$ is in fact better.

We note from Table 2 that our approach incurs more training time than the two-model version, as a result of the additional network evaluations needed for the flow-map consistency loss. Our method does have slightly lower memory requirements due to training only a single set of parameters. Based on our experiments, both our single-model approach and its two-model counterpart are strong candidates for few-step reward-alignment, and result in comparable final quality. Our approach would in particular be preferable when a coherent flow map is desired, for example for any-step generation or for downstream tasks that rely on the flow-map structure.

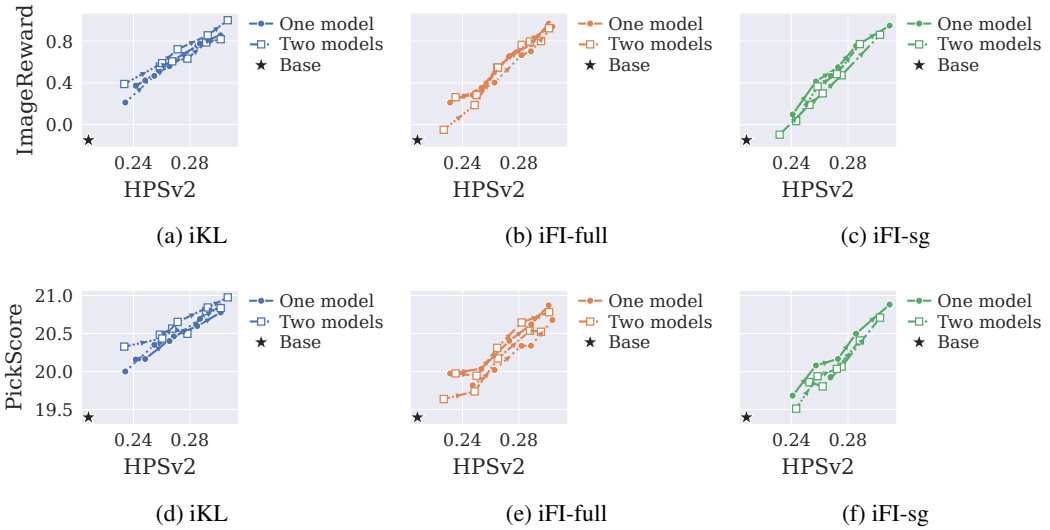


Figure 17: **Comparison with two-model version:** Reward transfer for ImageReward (top) and PickScore (bottom) when optimised for HPSv2, comparing our one-model approach (circles) with its two-model counterparts (squares) for each integrated regularization mechanism. Results shown for both $w(t) = \frac{1-t}{t}$ (solid) and $w(t) = 1$ (dotted).

Runtimes and Memory Requirements Table 2 lists the number of optimization steps, the wall-clock time and the peak GPU memory of each method for the HPSv2 runs. We note that these results are not all directly comparable between runs, as the different methods necessitate different design choices such as batch sizes and the use of EMA weights, which were chosen so that each method trained successfully to comparable levels of reward alignment. One could change these results significantly with different hyperparameter choices. Where it is possible to compare computational costs, we analyse these in the earlier discussions in the Appendix. We remark that the iFI updates are slightly slower than iKL because they require backpropagating through the score networks with respect to their input arguments, which is not required for the iKL regularizer.

Table 2: Number of optimisation steps, the wall-clock time and the peak GPU memory, for the ImageNet experiments optimizing HPSv2. Note that these results are not all directly comparable - see the comments in Appendix B.1 for a discussion of the relative computational costs.

Method	Steps	Wall-clock	Memory
One model (ours)			
iKL	20k	3 h 35 min	45 GiB
iFI-full	20k	4 h 5 min	53 GiB
iFI-sg	20k	3 h 55 min	53 GiB
Two-model version			
iKL	20k	3 h 15 min	48 GiB
iFI-full	20k	3 h 20 min	56 GiB
iFI-sg	20k	3 h 25 min	56 GiB
Baselines			
Reward only	20k	3 h	38 GiB
VFM	10k	9 h 35 min	54 GiB
FAV	2.5k	2 h 45 min	34 GiB
Adjoint Matching	5k	13 h 25 min	34 GiB

B.2 FLUX

For the FLUX reward-alignment experiment, we consider fine-tuning a flow-map distillation (`huggingface.co/gabeguofanclub/flux-1-dev-flowmap-lsd`) of FLUX.1-dev (Black Forest Labs, 2024). This flow map checkpoint has previously been used in Holderrieth et al. (2026) and Huang et al. (2026) for guiding sampling at inference-time. As this model was trained using the Lagrangian consistency loss, we choose this as the consistency loss in our fine-tuning procedure. We sampled using 4 steps during training and for evaluation.

For the reward function, following Eyring et al. (2024) we use a linear combination of HPSv2, ImageReward, PickScore, and CLIP. We used the coefficients 5, 0.03, 0.1 and 0.01 respectively, which were chosen to balance the gradient sizes. The weighting schedule was chosen to be $w(t) = \frac{1-t}{t}$ due to its good performance in our earlier ImageNet experiments. We use a reward scaling value of $\lambda = 0.05$ for iKL, and $\lambda = 0.25$ for iFI-full and iFI-sg, which resulted in similar training curves and resulting levels of reward-alignment over the 5,000 training steps. Due to the size of the model, we optimise a rank-32 LoRA, and we use a learning rate of 1e-5. We trained with a batch size of 16, split across 4 GH200 GPUs (4 per GPU). Training took around 15.5 hours for iKL and 18 hours for both iFI-full and iFI-sg. We use the training and evaluation prompts used in Domingo-Enrich et al. (2025).

A selection of generated samples initialized at the same noise are given in Figure 5, for the prompts *'a concept art of a vehicle, cyberpunk'*, 'a cute cat', and *'landscape photography, mountains with some forests, small lake in the center, fog in the background, sunrays, golden hour, high'*. The overall composition of each image is largely preserved throughout the optimization. As is common in such experiments, the resulting images pick up the usual polished and stylised effects associated to the reward functions, though we remark this is not overly extreme in our case due to the combined reward function and the effective regularization given by our approach to the base model (as indicated by the plateau in Figure 6). Different choices of reward strength could control such effects. We additionally report the GenEval score (Ghosh et al., 2023) of the fine-tuned models relative to the base

model, which assess how well the generated images adhere to the given prompts. We see that the fine-tuning improves this score too, despite not being part of the reward used during optimization.

C Application to Riemannian Flow Maps

So far in Sections 4 and Appendix B, we have considered applying our approach to image generation tasks which operate in the standard Euclidean regime. Many data modalities, for example protein or geoscience data, possess intrinsic geometric structure and are therefore more naturally modelled on general Riemannian manifolds rather than in Euclidean space. Flow-based modelling techniques have been extended to this setting (Chen & Lipman, 2024), allowing the learned dynamics and resulting generations to respect the underlying geometry of the data. Such methods have achieved strong generation ability in applications such as protein generation (Yim et al., 2023; Bose et al., 2024) and DNA sequence design (Stark et al., 2024; Davis et al., 2024; Cheng et al., 2024).

Riemannian Flow Maps Motivated by the success of flow maps for single- and few-step generation in Euclidean space, recent works have sought to extend the methodology to the Riemannian setting (Davis et al., 2026; Woo et al., 2026; Zhong et al., 2026). These works derive the Riemannian analogues of the Eulerian, Lagrangian, and Semigroup flow-map losses, allowing the training of flow-maps that respect the manifold structure.

Integrated Regularizers We now additionally consider whether our proposed integrated regularization mechanisms can be effective for Riemannian flow maps too. For a general Riemannian manifold, it is not generally possible to obtain the Riemannian score as a function of the diagonal velocity field. The fact that this is possible in the standard Euclidean case is a consequence of the linear relationship which holds for the conditional interpolations, which is retained when taking expectations. However, we can nevertheless still use the diagonal $v_{t,t}$ to obtain effective integrated regularizer mechanisms. Following the three regularizing methods proposed for the Euclidean case, we therefore propose to use the following three expressions in the parameter update.

Regularizers for the Riemannian Setting
Analogous to (8), (9), and (11) respectively, using the diagonal $v^\theta_{t,t}$ rather than the scores, we use the following expressions in the parameter update:

$$\text{(i)} \quad \int_0^1 w(t)\,\mathbb{E}\Big[\big(v^\theta_{t,t}(I^\theta_t) - v_{t,t}(I^\theta_t)\big)^\top \nabla_\theta I^\theta_t\Big]\,dt, \tag{18}$$

$$\text{(ii)} \quad \nabla_\theta \int_0^1 w(t)\,\mathbb{E}\Big[\big(v_{t,t}(I^\theta_t) - \mathrm{sg}[v^\theta_{t,t}](I^\theta_t)\big)^\top \big(v_{t,t}(I^\theta_t) + \mathrm{sg}[v^\theta_{t,t}](I^\theta_t) - 2\dot{I}^\theta_t\big)\Big]\,dt, \tag{19}$$

$$\text{(iii)} \quad 2\,\nabla_\theta \int_0^1 w(t)\,\mathbb{E}\Big[\big(v_{t,t}(I^\theta_t) - \mathrm{sg}[v^\theta_{t,t}](I^\theta_t)\big)^\top \big(\mathrm{sg}[v^\theta_{t,t}](I^\theta_t) - \dot{I}^\theta_t\big)\Big]\,dt. \tag{20}$$

The first option (18) is analogous to the integrated KL regularization (8). In the Riemannian case, this expression does not necessarily correspond to the gradient of a divergence. However, it retains the same geometric interpretation motivating its use: the output should be moved to reduce the mismatch between the fine-tuned and base vector fields, thus regularizing towards the base dynamics.

The second option (19) is analogous to the integrated Fisher divergence regularization (9). Rather than the Fisher divergence, this instead updates according to the gradient of the expression

$$\int_0^1 w(t)\mathbb{E}_{I^\theta_t \sim q^\theta_t} \|v^\theta_t(I^\theta_t) - v_t(I^\theta_t)\|^2 dt. \tag{21}$$

The final option (20) is analogous to the integrated stop-gradient Fisher divergence (11). Similarly to the Euclidean case, this updates according to the gradient of a stop-gradient version of the above expression,

$$\int_0^1 w(t)\mathbb{E}_{I^\theta_t \sim \mathrm{sg}[q^\theta_t]} \|v^\theta_t(I^\theta_t) - v_t(I^\theta_t)\|^2 dt. \tag{22}$$

C.1 PROTEIN GENERATION

To investigate the performance of our proposed integrated regularization mechanisms in the Riemannian setting, we consider the protein backbone design experimental setting from Woo et al. (2026), in which they train a Riemannian flow map for unconditional *de novo* protein generation using the SCOPe dataset (Chandonia et al., 2022). They demonstrate few-step generative ability comparable to existing state-of-the-art flow matching baselines, with significantly improved generation in the few-step regime.

Secondary Structure Reward It is often desirable to control the proportions of different secondary structures in the generations, for example to control the proportions of α-helices and β-sheets in the generated samples. Woo et al. (2026) show that their Riemannian flow map can be used as a one-step lookahead to steer the generations at inference time, thus enabling increasing the proportions of residues involved in the different structure types. Following Woo et al. (2026), we consider modifying the base model to encourage more α-helices and more β-sheets respectively. Rather than inference-time steering, we will instead consider *fine-tuning* the flow map using our proposed integrated regularizer mechanisms.

Experimental Details We follow a similar experimental setup to Woo et al. (2026), and consider fine-tuning their RMF-S model which is trained on the SCOPe dataset (Chandonia et al., 2022) for unconditional *de novo* protein generation. The model uses an SE(3)-equivariant architecture from Yim et al. (2023). For details, see Woo et al. (2026). We train for 2,000 fine-tuning steps, with a batch size of 16, and use a small learning rate of 3e-6 which gave smooth reward curves during the optimization. For the flow map terms in the loss function, we use the same setup as used for the pretraining. Similarly to the pretraining, the finetuning is performed for residue lengths uniformly sampled from 60 to 128, with results provided for length 128 similarly to the results in Woo et al. (2026). The sampling was done with 5 steps (as used in Woo et al. (2026)), and was done deterministically (in Woo et al. (2026), a stochastic inference procedure was used, but here we use deterministic sampling for simplicity). Training was done on a single A100 GPU, and took around 2 hours per run.

Reward Functions The standard method for assigning the secondary structure to residues in a protein is the Define Secondary Structure of Proteins (DSSP) algorithm (Kabsch & Sander, 1983), an open source implementation of which is given by PyDSSP. This function is non-differentiable, so as a reward function we use the differentiable proxy DiffDSSP introduced in Woo et al. (2026). DiffDSSP assigns a scalar value [0,1] reflecting a soft proportion of residues involved in α-helix or β-sheet structures, denoted by $\text{DiffDSSP}_\alpha(x)$ and $\text{DiffDSSP}_\beta(x)$ respectively. To encourage increasing the proportion of α-helices and β-sheets in the generated samples, we use the differentiable rewards

$$r_\alpha(x) = -(1 - \text{DiffDSSP}_\alpha(x))^2, \qquad r_\beta(x) = -(1 - \text{DiffDSSP}_\beta(x))^2, \tag{23}$$

respectively. We remark that this differs slightly from the reward construction used in Woo et al. (2026); there, they optimize both structure proportions simultaneously according to a combined reward $r(x) = -(w_\alpha - \text{DiffDSSP}_\alpha(x))^2 - (w_\beta - \text{DiffDSSP}_\beta(x))^2$, with weights w_α, w_β chosen appropriately to steer towards the desired structure and away from the other. In our experiments, we found that the $\text{DiffDSSP}_\alpha(x)$ could be altered more easily and by a significantly larger amount than the $\text{DiffDSSP}_\beta(x)$, which we anticipate is because α-helices are more local structures than β-sheets so can be generated more easily. We therefore did not include the term steering away from α-helices in the r_β reward, as this term dominated the fine-tuning optimization. We used a constant weight function $w(t) = 1$ for the integrated regularizers.

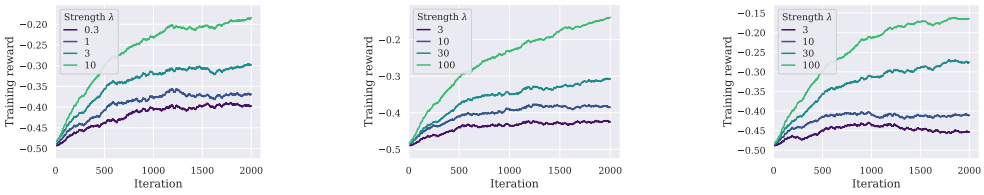
Results Results for each regularization mechanism are provided in Table 3, computed using 500 samples with residue length 128. We report the value of the differentiable reward used in the fine-tuning, as well as the DSSP (for the secondary structure being targeted), the mean DSSP of the top 50 samples, and the proportion of samples whose DSSP was improved over the base model.

Training curves for the differentiable reward (as used for finetuning during training, plotted with EMA parameter 0.99) and the mean DSSP value (over 500 samples for residue length 128) are plotted in Figures 18 and 19 for the α-helix experiment, and Figures 20 and 21 for the β-sheet experiment respectively. As desired, the differentiable reward curves show a smooth increase before plateauing

Reward	Reg. type	λ	Diff. reward ↑	Mean ↑	Top-50 mean ↑	Frac. improved ↑
α-helix	Base	—	-0.5082 ± 0.1028	0.30 ± 0.18	0.55	—
	(18)	0.3	-0.3924 ± 0.1638	0.39 ± 0.23	0.74	0.76
		1.0	-0.3545 ± 0.1719	0.41 ± 0.24	0.77	0.80
		3.0	-0.2744 ± 0.1568	0.50 ± 0.22	0.82	0.95
		10.0	-0.1622 ± 0.1229	0.65 ± 0.20	0.89	1.00
	(19)	3	-0.4311 ± 0.1300	0.35 ± 0.20	0.66	0.77
		10	-0.3817 ± 0.1411	0.39 ± 0.20	0.68	0.90
		30	-0.2944 ± 0.1430	0.48 ± 0.20	0.77	0.99
		100	-0.1191 ± 0.0800	0.75 ± 0.15	0.91	1.00
	(20)	3	-0.4556 ± 0.1361	0.32 ± 0.19	0.58	0.62
		10	-0.4231 ± 0.1353	0.33 ± 0.18	0.56	0.73
		30	-0.2700 ± 0.1390	0.51 ± 0.20	0.80	0.99
		100	-0.1453 ± 0.0883	0.70 ± 0.15	0.88	1.00
β-sheet	Base	—	-0.6675 ± 0.0004	0.18 ± 0.12	0.34	—
	(18)	1	-0.6673 ± 0.0004	0.24 ± 0.13	0.41	0.65
		10	-0.6672 ± 0.0005	0.22 ± 0.13	0.41	0.59
		100	-0.6668 ± 0.0007	0.24 ± 0.16	0.45	0.65
		1000	-0.6666 ± 0.0008	0.24 ± 0.15	0.47	0.73
	(19)	300	-0.6674 ± 0.0005	0.19 ± 0.12	0.35	0.50
		1,000	-0.6671 ± 0.0006	0.20 ± 0.14	0.40	0.56
		3,000	-0.6670 ± 0.0007	0.22 ± 0.13	0.40	0.67
		10,000	-0.6666 ± 0.0007	0.26 ± 0.13	0.43	0.81
	(20)	300	-0.6673 ± 0.0005	0.20 ± 0.12	0.35	0.54
		1,000	-0.6670 ± 0.0006	0.23 ± 0.13	0.42	0.73
		3,000	-0.6668 ± 0.0007	0.23 ± 0.14	0.41	0.72
		10,000	-0.6664 ± 0.0008	0.31 ± 0.14	0.40	0.89

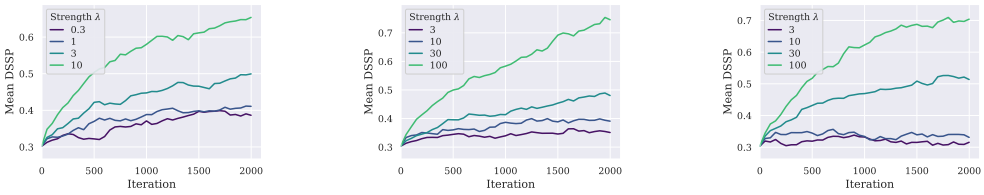
Table 3: Results for α-helix and β-sheet rewards for different λ values for 500 generated samples, using regularisation types (18) (similar to iKL), (19) (similar to iFI-full), and (20) (similar to iFI-sg).

at a level determined by the tilting strength (though we note that some of the stronger-strength runs did not flatten out within the 2,000 steps). We found that the differentiable proxy generally induced a similar increase in the corresponding DSSP values, though we note this correspondence was not as clear in some of the β-sheet examples.



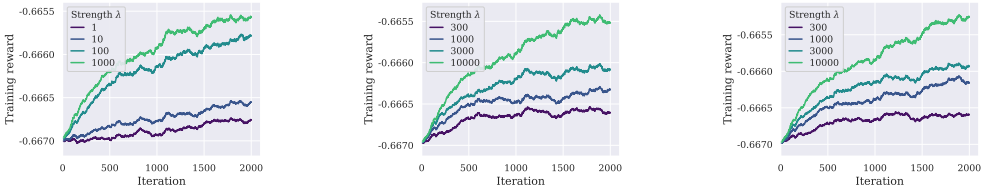
(a) (18) - similar to iKL (b) (19) - similar to iFI-full (c) (20) - similar to iFI-sg

Figure 18: Differentiable α-helix reward during fine-tuning, for four steering strengths λ under each regularizer. Curves are exponential moving averages with parameter 0.99. The reward is evaluated at the randomly sampled training length $L \in [60, 128]$.



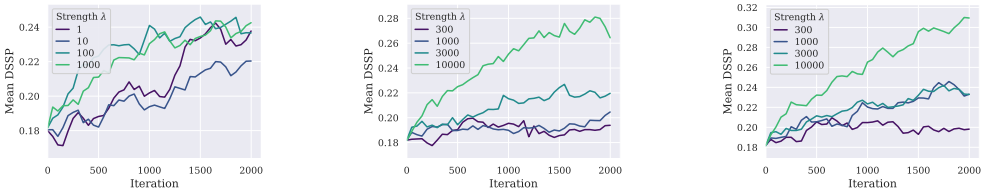
(a) (18) - similar to iKL (b) (19) - similar to iFI-full (c) (20) - similar to iFI-sg

Figure 19: Mean DSSP α-helix fraction at $L = 128$ over fine-tuning, for four steering strengths λ under each regularizer.



(a) (18) - similar to iKL (b) (19) - similar to iFI-full (c) (20) - similar to iFI-sg

Figure 20: Differentiable β-sheet reward during fine-tuning, for four steering strengths λ under each regularizer. Curves are exponential moving averages with parameter 0.99. The reward is evaluated at the randomly sampled training length $L \in [60, 128]$.



(a) (18) - similar to iKL (b) (19) - similar to iFI-full (c) (20) - similar to iFI-sg

Figure 21: Mean DSSP β-sheet fraction at $L = 128$ over fine-tuning, for four steering strengths λ under each regularizer.

D PROOFS

Here, we provide proofs of the results used in the paper. We remark that many of these results (or similar variants) have also appeared previously in works in the distillation literature; we provide full proofs here for completeness.

D.1 KL DIVERGENCE REGULARIZER

Proposition 3.1 (Gradient of the integrated KL divergence, Wang et al. (2023); Luo et al. (2023)). *The D_{iKL} has gradient equal to*

$$\nabla_\theta D_{\mathrm{iKL}}(q_1^\theta \| p_1) = \int_0^1 w(t)\, \mathbb{E}\Big[\big(\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\big)^\top \nabla_\theta I_t^\theta \Big]\, dt. \tag{8}$$

where the expectation is over the law of I_t^θ coming from (6).

Proof. Exchanging the derivative through the integrals and applying the chain rule, we have

$$\begin{aligned}
\nabla_\theta D_{\mathrm{iKL}}(q_1^\theta \| p_1) &= \int_0^1 w(t)\, \nabla_\theta D_{\mathrm{KL}}(q_t^\theta \| p_t)\, dt && (24)\\
&= \int_0^1 w(t)\, \mathbb{E}\Big[\frac{\partial}{\partial \theta} \Big(\log q_t^\theta(I_t^\theta) - \log p_t(I_t^\theta) \Big) \Big]\, dt && (25)\\
&= \int_0^1 w(t)\, \mathbb{E}\Big[\nabla_\theta \log q_t^\theta(I_t^\theta) + \nabla_x \log q_t^\theta(I_t^\theta)^\top \nabla_\theta I_t^\theta - \nabla_x \log p_t(I_t^\theta)^\top \nabla_\theta I_t^\theta \Big]\, dt && (26)\\
&= \int_0^1 w(t)\, \mathbb{E}\Big[\big(\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\big)^\top \nabla_\theta I_t^\theta \Big]\, dt. && (27)
\end{aligned}$$

Here, the expectation is over the randomness used to generate I_t^θ. In the final line, we use that the parameter-score term integrates to 0,

$$\mathbb{E}\big[\nabla_\theta \log q_t^\theta(I_t^\theta)\big] = \int q_t^\theta(x) \nabla_\theta \log q_t^\theta(x) dx = \int \nabla_\theta q_t^\theta(x) dx = \nabla_\theta \int q_t^\theta dx = 0. \tag{28}$$

□

D.2 FISHER DIVERGENCE REGULARIZER

To prove Proposition 3.2, we will use the following well-known *score-projection identity* (Vincent, 2011; Zhou et al., 2024; Luo et al., 2024). We provide its short proof here for completeness.

Lemma D.1 (Score-projection identity, (Vincent, 2011; Zhou et al., 2024; Luo et al., 2024)). *For a measurable $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ with finite second moment, we have*

$$\mathbb{E}\Big[\varphi(I_t^\theta)^\top \big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0 \big) \Big] = 0, \tag{29}$$

where the expectation is over the randomness in the coupling (I_t^θ, I_0) induced by the interpolant construction.

Proof. Recall that by Tweedie's formula, for the Gaussian initial distribution and the linear interpolant (6) we can write the score of the marginal q_t^θ as a conditional expectation,

$$\nabla_x \log q_t^\theta(I_t^\theta) = \mathbb{E}\Big[-\tfrac{1}{1-t} I_0 \,|\, I_t^\theta \Big]. \tag{30}$$

Rearranging gives $\mathbb{E}\left[\nabla_x \log q_t^\theta(I_t^\theta) + \frac{1}{1-t} I_0 \,|\, I_t^\theta \right] = 0$. The result then follows from the tower property, conditioning on I_t^θ.

$$\mathbb{E}\Big[\varphi(I_t^\theta)^\top \big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0 \big) \Big] = \mathbb{E}\Big[\varphi(I_t^\theta)^\top \mathbb{E}\Big[\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0 \,|\, I_t^\theta \Big] \Big] = 0. \tag{31}$$

□

Proposition 3.2 (**Gradient of the integrated Fisher divergence,** Zhou et al. (2024)). *The D_{iFI} has gradient equal to*

$$\nabla_\theta D_{\mathrm{iFI}}(q_1^\theta \| p_1) = \nabla_\theta \int_0^1 w(t)\mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta)\big)^\top \Big(\nabla_x \log p_t(I_t^\theta) + \mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta) + \tfrac{2}{1-t} I_0\Big)\Big]\, dt. \tag{9}$$

where $sg[\cdot]$ indicates that $\nabla_x \log q_t^\theta$ is treated as constant with respect to θ when taking the gradient, and where the expectation is over I_t^θ and I_0, which are sampled as in (6).

Proof. Consider first the Fisher divergence between the marginals q_t^θ, p_t at an individual time t, that is

$$D_{\mathrm{FI}}(q_t^\theta \| p_t) = \mathbb{E}_{I_t^\theta \sim q_t^\theta}\left[\|\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\|^2\right]. \tag{32}$$

By the score-projection identity, taking $\varphi = \nabla_x \log p_t - \nabla_x \log q_t^\theta$, we have (again with the expectation over the joint distribution of (I_t^θ, I_0))

$$\mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \nabla_x \log q_t^\theta(I_t^\theta)\big)^\top \big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)\Big] = 0. \tag{33}$$

We can therefore perform the rearrangements

$$\begin{aligned}
D_{\mathrm{FI}}(q_t^\theta \| p_t) &= \mathbb{E}_{I_t^\theta \sim q_t^\theta}\Big[\big\|\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\big\|^2\Big] && (34)\\
&= \mathbb{E}\Big[\big\|\nabla_x \log q_t^\theta(I_t^\theta) - \nabla_x \log p_t(I_t^\theta)\big\|^2\Big] && (35)\\
&\quad + 2\,\mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \nabla_x \log q_t^\theta(I_t^\theta)\big)^\top \big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)\Big] && (36)\\
&= \mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \nabla_x \log q_t^\theta(I_t^\theta)\big)^\top && (37)\\
&\quad \Big(\big(\nabla_x \log p_t(I_t^\theta) - \nabla_x \log q_t^\theta(I_t^\theta)\big) + 2\big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)\Big)\Big] && (38)\\
&= \mathbb{E}\Big[\big(\nabla_x \log p_t(I_t^\theta) - \nabla_x \log q_t^\theta(I_t^\theta)\big)^\top \big(\nabla_x \log p_t(I_t^\theta) + \nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{2}{1-t} I_0\big)\Big] && (39)\\
&= \mathbb{E}\Big[\big\|\nabla_x \log p_t(I_t^\theta) + \tfrac{1}{1-t} I_0\big\|^2 - \big\|\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big\|^2\Big]. && (40)
\end{aligned}$$

In this expression, the first term is already of the form that we require. Consider the gradient with respect to θ of the second term. Applying the chain rule, we have

$$\begin{aligned}
\nabla_\theta\, \mathbb{E}\left[\|\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\|^2\right] &= \mathbb{E}\Big[2\big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)^\top \partial_\theta\big(\nabla_x \log q_t^\theta\big)(I_t^\theta) \\
&\qquad + 2\big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)^\top \nabla_x^2 \log q_t^\theta(I_t^\theta)\, \nabla_\theta I_t^\theta\Big] && (41)\\
&= \mathbb{E}\Big[2\big(\nabla_x \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t} I_0\big)^\top \nabla_x^2 \log q_t^\theta(I_t^\theta)\, \nabla_\theta I_t^\theta\Big] && (42)\\
&= \nabla_\theta\, \mathbb{E}\left[\Big\|\mathrm{sg}\big[\nabla_x \log q_t^\theta\big](I_t^\theta) + \tfrac{1}{1-t} I_0\Big\|^2\right], && (43)
\end{aligned}$$

where in the second line we again used the score-projection identity with $\varphi = \partial_\theta\big(\nabla_x \log q_t^\theta\big)$ to remove the first term of the expansion. In the final line, we observe that the remaining gradient term is the same as if the score-function had parameters θ frozen.

Integrating over time t, and performing a simple rearrangement, we obtain the result (9). □

D.3 STOP-GRAD FISHER DIVERGENCE REGULARIZER

To prove this result we need another additional lemma, which is obtained by differentiating the score-projection identity.

Lemma D.2. *For a measurable $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ with finite second moment, we have*

$$\mathbb{E}_{I_t^\theta \sim q_t^\theta}\left[\varphi(I_t^\theta)^\top \partial_\theta\big(\nabla_x \log q_t^\theta\big)(I_t^\theta)\right] = -\frac{d}{d\theta}\mathbb{E}\left[\varphi(I_t^\theta)^\top\big(\operatorname{sg}\big[\nabla_x \log q_t^\theta\big]\,(I_t^\theta) + \tfrac{1}{1-t}I_0\big)\right]. \quad (44)$$

Proof. Recall from the score-projection identity (Lemma D.1) that we have $\mathbb{E}\left[\nabla_x \log q_t^\theta(I_t^\theta) + \frac{1}{1-t}I_0 \,|\, I_t^\theta\right] = 0$. Taking gradients with respect to θ, we obtain three terms (one for each dependence on θ),

$$\begin{aligned} 0 &= \frac{d}{d\theta}\mathbb{E}\left[\varphi(I_t^\theta)^\top\big(\nabla \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t}I_0\big)\right] && (45)\\ &= \mathbb{E}\Big[\big(\nabla_x\varphi(I_t^\theta)\,\partial_\theta I_t^\theta\big)^\top\big(\nabla \log q_t^\theta(I_t^\theta) + \tfrac{1}{1-t}I_0\big)\Big] \\ &\quad + \mathbb{E}\Big[\varphi(I_t^\theta)^\top\nabla_x^2 \log q_t^\theta(I_t^\theta)\nabla_\theta I_t^\theta\Big] \\ &\quad + \mathbb{E}\Big[\varphi(I_t^\theta)^\top\,\partial_\theta(\nabla_x \log q_t^\theta)(I_t^\theta).\Big] && (46)\end{aligned}$$

Of these three terms, the first two combined constitute the gradient of $\frac{d}{d\theta}\mathbb{E}\left[\varphi(I_t^\theta)^\top\big(\operatorname{sg}\big[\nabla_x \log q_t^\theta\big]\,(I_t^\theta) + \frac{1}{1-t}I_0\big)\right]$. We therefore have

$$0 = \frac{d}{d\theta}\mathbb{E}\left[\varphi(I_t^\theta)^\top\big(\operatorname{sg}\big[\nabla_x \log q_t^\theta\big]\,(I_t^\theta) + \tfrac{1}{1-t}I_0\big)\right] + \mathbb{E}_{I_t^\theta\sim q_t^\theta}\Big[\varphi(I_t^\theta)^\top\,\partial_\theta(\nabla_x \log q_t^\theta)(I_t^\theta).\Big]. \quad (47)$$

Rearranging gives the result. □

Note that on the left hand side of (44), the θ-gradient arises through the score-function only, and not through the argument I_t^θ. By contrast, on the right hand side the gradient flows through the argument I_t^θ only, and does not flow through the score-function. The proof of Proposition 3.3 follows from an application of this lemma. In the remainder, we will let $\bar\theta$ be shorthand stopping the gradient with respect to θ.

Proposition 3.3 (Gradient of the integrated stop-gradient Fisher divergence (10), Zhou et al. (2024); Luo et al. (2024)). *The $D_{\text{iFI-sg}}$ has gradient equal to*

$$\nabla_\theta D_{\text{iFI-sg}}(q_1^\theta\|p_1) = 2\,\nabla_\theta\int_0^1 w(t)\cdot$$
$$\mathbb{E}\left[\big(\nabla_x \log p_t(I_t^\theta) - \operatorname{sg}\big[\nabla_x \log q_t^\theta\big]\,(I_t^\theta)\big)^\top\left(\operatorname{sg}\big[\nabla_x \log q_t^\theta\big]\,(I_t^\theta) + \frac{1}{1-t}I_0\right)\right]dt, \quad (11)$$

where $sg[\cdot]$ indicates that $\nabla_x \log q_t^\theta$ is treated as constant with respect to θ when taking the gradient, and where the expectation is over I_t^θ and I_0, which are sampled as in (6).

Proof. For a particular time t, taking gradients of $D_{\text{iFI-sg}}(q_t^\theta\|p_t)$, we have

$$\begin{aligned}\nabla_\theta D_{\text{iFI-sg}}(q_t^\theta\|p_t) &= \nabla_\theta\mathbb{E}_{I_t^{\bar\theta}\sim q_t^{\operatorname{sg}[\theta]}}\left[\left\|\nabla_x \log q_t^\theta(I_t^{\bar\theta}) - \nabla \log p_t(I_t^{\bar\theta})\right\|^2\right] && (48)\\ &= \mathbb{E}_{I_t^{\bar\theta}\sim q_t^{\operatorname{sg}[\theta]}}\left[2\left(\nabla_x \log q_t^\theta(I_t^{\bar\theta}) - \nabla \log p_t(I_t^{\bar\theta})\right)^\top\partial_\theta(\nabla \log q_t^\theta)(I_t^{\bar\theta})\right]. && (49)\end{aligned}$$

Applying Lemma D.2 with $\varphi = \nabla \log q_t^{\bar\theta} - \nabla \log p_t$, we obtain the desired result. □

D.4 Other results

Riemannian Setting Recall from Appendix C that the expressions (19) and (20) correspond to the gradients of the following expressions, with or without a stop-gradient on the expectation,

$$\int_0^1 w(t)\mathbb{E}_{I_t^\theta\sim q_t^\theta}\|v_t^\theta(I_t^\theta)-v(I_t^\theta)\|^2dt, \qquad \int_0^1 w(t)\mathbb{E}_{I_t^\theta\sim \mathrm{sg}[q_t^\theta]}\|v_t^\theta(I_t^\theta)-v(I_t^\theta)\|^2dt.$$

The proofs of these results proceed exactly as in the proofs of Propositions 3.2 and 3.3, but with the following two lemmas in place of Lemmas D.1 and D.2 respectively.

Lemma D.3. *For a fixed φ, we have*

$$\mathbb{E}\left[\langle\varphi(I_t^\theta), v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta\rangle\right]=0, \tag{50}$$

where the expectation is over the randomness used to generate I_t^θ.

Proof. Recall that by the definition of the diagonal $v_{t,t}^\theta$, we have $v_{t,t}^\theta(I_t^\theta)=\mathbb{E}\left[\dot I_t^\theta \mid I_t^\theta\right]$. Rearranging gives $\mathbb{E}\left[v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta \mid I_t^\theta\right]=0$. The result then again follows from the tower property, conditioning on I_t^θ,

$$\mathbb{E}\left[\langle\varphi(I_t^\theta), v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta\rangle\right]=\mathbb{E}\left[\langle\varphi(I_t^\theta), \mathbb{E}_{I_t^\theta\sim q_t^\theta}\left[v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta \mid I_t^\theta\right]\right]=0. \tag{51}$$

□

Lemma D.4. *For a measurable $\varphi:\mathbb{R}^d\to\mathbb{R}^d$ with finite second moment, we have*

$$\mathbb{E}\left[\langle\varphi(I_t^\theta), \partial_\theta(v_{t,t}^\theta)(I_t^\theta)\rangle\right]=-\frac{d}{d\theta}\mathbb{E}\left[\langle\varphi(I_t^\theta), \mathrm{sg}\left[v_{t,t}^\theta\right](I_t^\theta)-\dot I_t^\theta\rangle\right]. \tag{52}$$

Proof. From Lemma D.3, we have $\mathbb{E}\left[\langle\varphi(I_t^\theta), v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta\rangle\right]=0$. Taking gradients, we therefore have

$$0=\frac{d}{d\theta}\mathbb{E}\left[\langle\varphi(I_t^\theta), v_{t,t}^\theta(I_t^\theta)-\dot I_t^\theta\rangle\right] \tag{53}$$

$$=\mathbb{E}\left[\langle\varphi(I_t^\theta), \partial_\theta(v_{t,t}^\theta)(I_t^\theta)\rangle\right]+\frac{d}{d\theta}\mathbb{E}\left[\langle\varphi(I_t^\theta), \mathrm{sg}[v_{t,t}^\theta](I_t^\theta)-\dot I_t^\theta\rangle\right]. \tag{54}$$

Rearranging gives the desired result. □

Exponential Reward Tilt We also provide a proof of the exponential reward-tilt results stated in Section 3.5.

Proposition 3.4 (Exponential reward tilt). *Consider using the integrated Fisher divergence with weight $w(t)=\frac{1-t}{t}$. The optimal terminal law is*

$$q_1^*(x)\propto p_1(x)e^{r(x)}. \tag{17}$$

Proof. To show this result, we consider the evolution of the KL divergence $\frac{\mathrm{d}}{\mathrm{d}t}D_{\mathrm{KL}}(q_t\|p_t)$, where p_t and q_t are two interpolant paths started at $p_0=q_0$ that satisfy the continuity equations associated with their respective drifts b_t^p and b_t^q:

$$\partial_t p_t+\nabla\cdot(p_t b_t^p)=0, \tag{55}$$

$$\partial_t q_t+\nabla\cdot(q_t b_t^q)=0. \tag{56}$$

Under mild assumptions, by taking the derivative through the integral, substituting in the continuity equations, and integrating by parts, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t} D_{\mathrm{KL}}(q_t \| p_t) = \frac{\mathrm{d}}{\mathrm{d}t} \int q_t(x) \log \frac{q_t(x)}{p_t(x)} \,\mathrm{d}x \tag{57}$$

$$= \int \left(\log \frac{q_t}{p_t} + 1 \right) \partial_t q_t - \frac{q_t}{p_t} \partial_t p_t \,\mathrm{d}x \tag{58}$$

$$= -\int \left(\log \frac{q_t}{p_t} + 1 \right) \nabla_x \cdot (q_t b_t^q) \,\mathrm{d}x + \int \frac{q_t}{p_t} \nabla_x \cdot (p_t b_t^p) \,\mathrm{d}x \tag{59}$$

$$= \int q_t b_t^q \cdot \nabla_x \log \frac{q_t}{p_t} \,\mathrm{d}x - \int p_t b_t^p \cdot \nabla_x \left(\frac{q_t}{p_t} \right) \mathrm{d}x \tag{60}$$

$$= \int q_t \left(b_t^q - b_t^p \right)^\top \nabla_x \log \frac{q_t}{p_t} \,\mathrm{d}x, \tag{61}$$

where in the final line, we used $\nabla_x \left(\frac{q_t}{p_t} \right) = \frac{q_t}{p_t} \nabla_x \log \frac{q_t}{p_t}$. That is,

$$\frac{\mathrm{d}}{\mathrm{d}t} D_{\mathrm{KL}}(q_t \| p_t) = \mathbb{E}_{X_t \sim q_t} \left[\left(b_t^q(X_t) - b_t^p(X_t) \right)^\top \left(\nabla_x \log q_t(X_t) - \nabla_x \log p_t(X_t) \right) \right]. \tag{62}$$

Writing the drifts in terms of the corresponding score functions, this can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t} D_{\mathrm{KL}}(q_t \| p_t) = \frac{1-t}{t} \mathbb{E}_{X_t \sim q_t} \left[\left\| \nabla_x \log q_t(X_t) - \nabla_x \log p_t(X_t) \right\|^2 \right] \tag{63}$$

$$= \frac{1-t}{t} D_{\mathrm{FI}}(q_t \| p_t). \tag{64}$$

As both interpolant paths begin at the same distribution, $q_0 = p_0$, integrating (64) from 0 to 1 gives

$$D_{\mathrm{KL}}(q_1 \| p_1) = \int_0^1 \frac{1-t}{t} D_{\mathrm{FI}}(q_t \| p_t) \,\mathrm{d}t. \tag{65}$$

Thus, the objective with the integrated Fisher divergence and weight $w(t) = \frac{1-t}{t}$ coincides with the KL-regularized objective defined only on the terminal distribution, completing the result. □

E GENERATIONS

In Figures 22 to 24, we display a selection of the generated samples from the ImageNet experiments. For the same initial noises and class conditioning, we plot 1-step samples from the base model and the three integrated regularization methods, optimized for each of the three reward functions HPSv2, ImageReward, and PickScore respectively. We display generations for two reward strengths λ for both weighting schedules $w(t) = \frac{1-t}{t}$ and $w(t) = 1$, with the particular λ selected because they gave similar values of the optimized reward function (shown on the right column). Despite similar values of the optimized rewards, we can see qualitative differences between the runs.

Examining the samples, we generally see that the samples $w(t) = \frac{1-t}{t}$ remain more visually consistent with the base model than those using $w(t) = 1$. Indeed, using $w(t) = 1$ often caused unusual artifacts in the samples or partially collapsed training, particularly when optimizing for ImageReward or PickScore. The visualization generally align with our analysis of the stability in Appendix B.1.3, and again supports the idea that weighting the regularization earlier in the probability path gives improved stability.

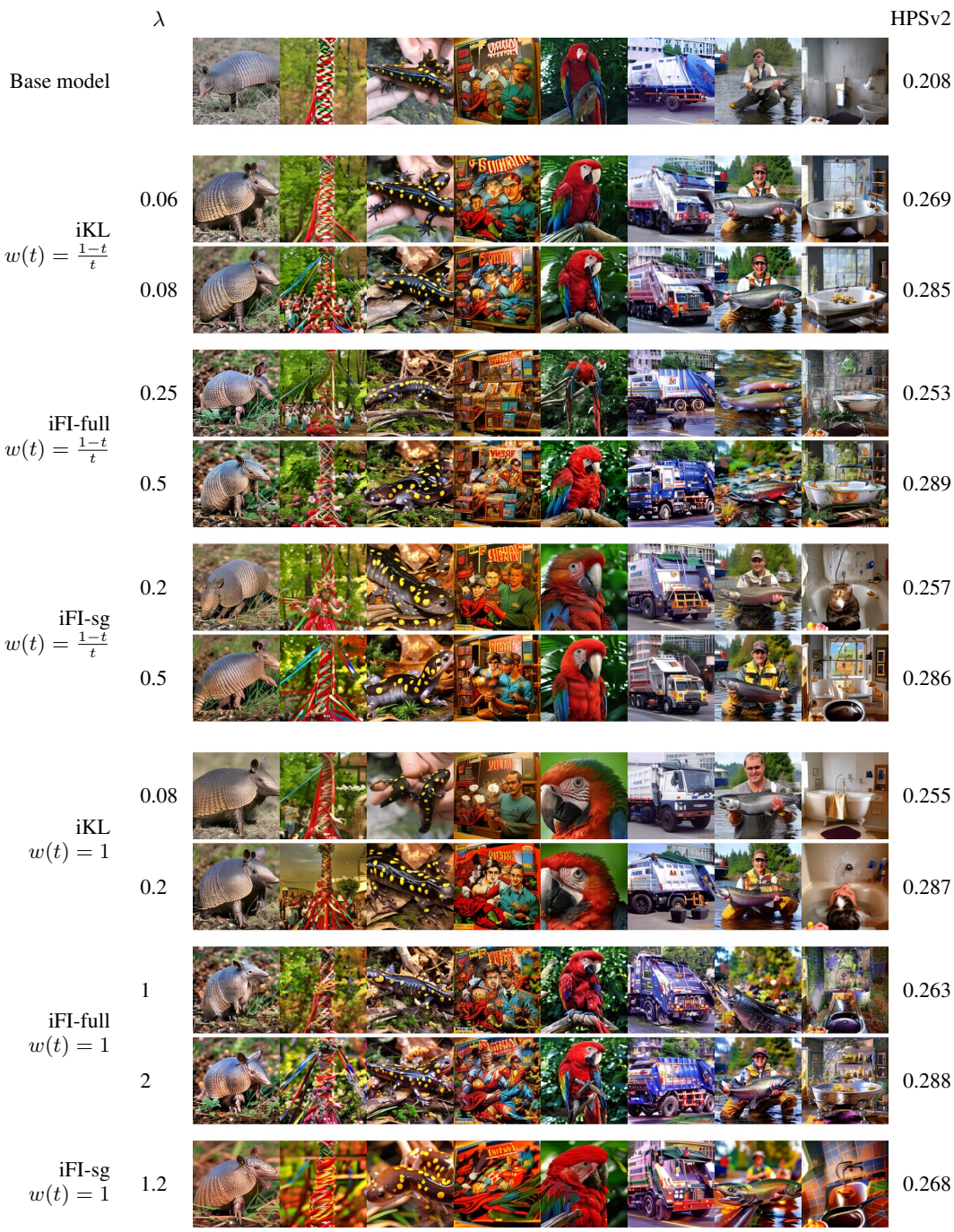


Figure 22: ImageNet generations when optimizing for HPSv2 (Wu et al., 2023).

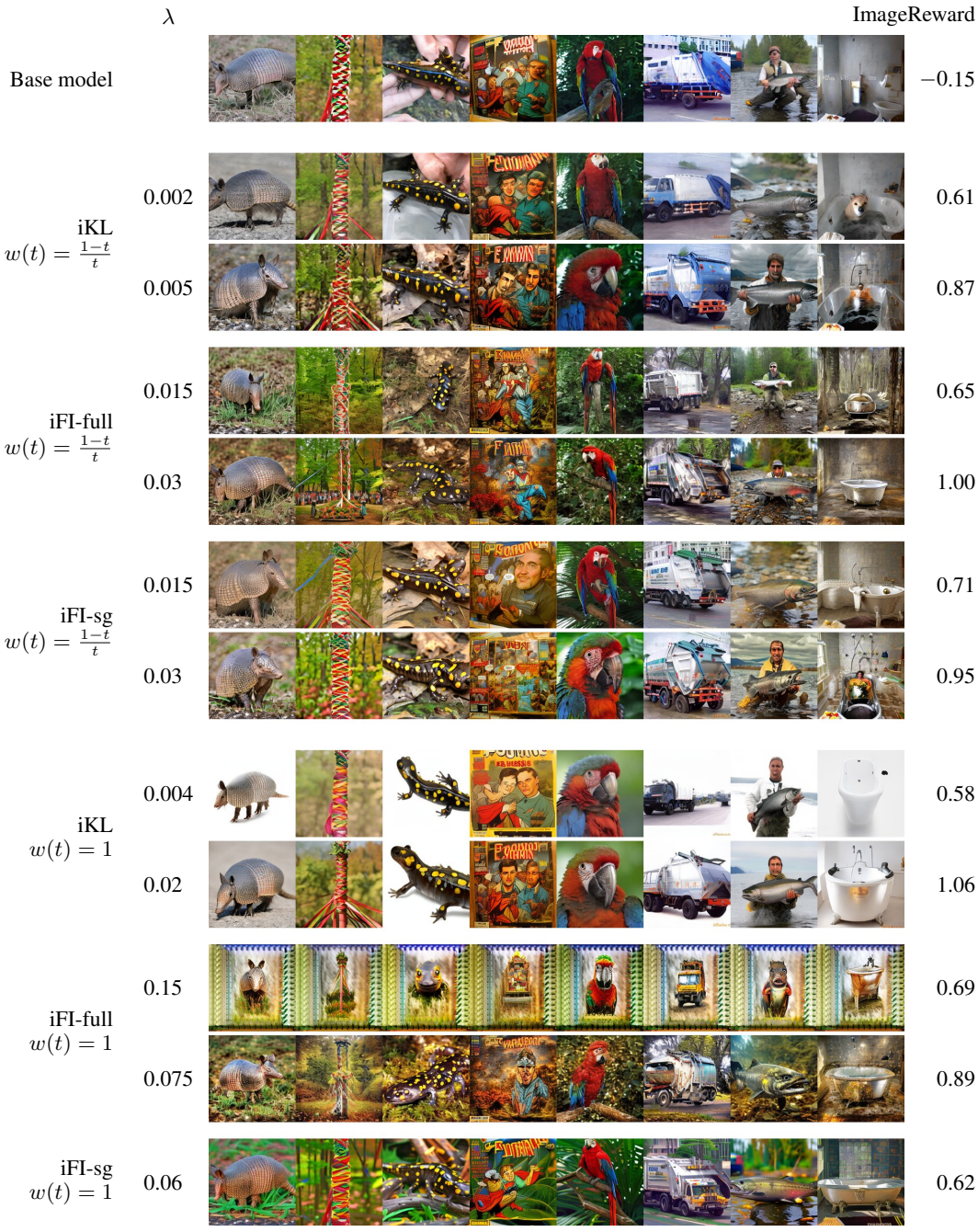


Figure 23: ImageNet generations when optimizing for ImageReward (Xu et al., 2023).

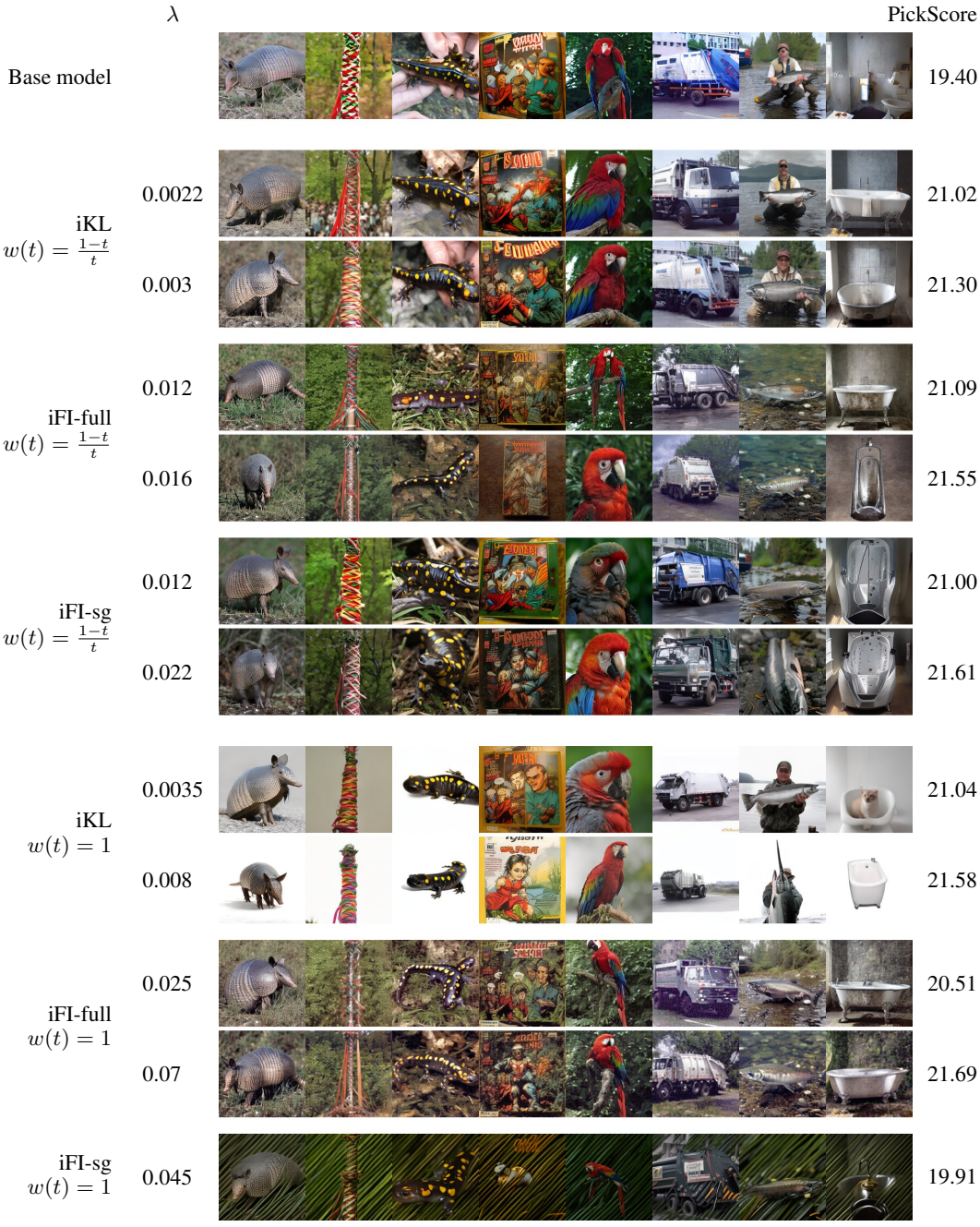


Figure 24: ImageNet generations when optimizing for PickScore (Kirstain et al., 2023).

Reward Strengths Table 4 lists the reward strengths used for each method in the ImageNet experiment, for each optimised reward. These were selected to induce comparable ranges of the optimised reward after training, to allow for fair assessment of how well the increase transferred to the other reward functions.

Table 4: Reward strengths used for the ImageNet experiments (5 per method, for each reward). For FAV, this corresponds to the parameter β in Lee et al. (2026), and for the reward-only runs this corresponds to the learning rate (which was made small to ensure the unregularized optimization covered the desired reward range).

Method	$w(t)$	HPSv2	ImageReward	PickScore
iKL	$\frac{1-t}{t}$	0.03, 0.04, 0.06, 0.08, 0.12	0.001, 0.002, 0.003, 0.005, 0.008	0.001, 0.0015, 0.0022, 0.003, 0.0045
	1	0.04, 0.08, 0.12, 0.2, 0.3	0.002, 0.004, 0.01, 0.02, 0.05	0.002, 0.0035, 0.0055, 0.008, 0.012
iFI-full	$\frac{1-t}{t}$	0.18, 0.25, 0.35, 0.5, 0.7	0.012, 0.015, 0.02, 0.03, 0.045	0.006, 0.009, 0.012, 0.016, 0.021
	1	0.7, 1, 1.5, 2, 3	0.04, 0.055, 0.075, 0.1, 0.15	0.025, 0.035, 0.05, 0.07, 0.09
iFI-sg	$\frac{1-t}{t}$	0.12, 0.2, 0.35, 0.5, 0.9	0.01, 0.015, 0.02, 0.03, 0.05	0.008, 0.012, 0.017, 0.022, 0.03
	1	0.3, 0.7, 1.2, 2, 3	0.06, 0.08, 0.1, 0.13, 0.18	0.03, 0.045, 0.06, 0.08, 0.1
Reward only*	–	1, 2, 4, 6, 10	4, 7, 10, 15, 20	3, 5, 7, 10, 12
VFM	–	0.03, 0.06, 0.1, 0.3, 1	0.001, 0.002, 0.003, 0.01, 0.03	0.003, 0.01, 0.03, 0.1, 0.12
FAV (β)	–	2, 3, 4, 5, 7	–	0.03, 0.04, 0.06, 0.08, 0.1
Adjoint Matching	–	3, 10, 20, 40, 60 $(\times 10^3)$	1, 3, 10, 30, 100 $(\times 10^3)$	0.1, 0.3, 1, 3, 10 $(\times 10^3)$

F LICENSES

The following assets were used in this work.

- HPS v2 (Wu et al., 2023), Apache-2.0 License
  `https://github.com/tgxs002/HPSv2`
- ImageReward (Xu et al., 2023), Apache-2.0 License
  `https://github.com/zai-org/ImageReward`
- PickScore (Kirstain et al., 2023), MIT License
  `https://github.com/yuvalkirstain/pickscore`
- DreamSim (Fu et al., 2023), MIT License
  `https://github.com/ssundaram21/dreamsim`
- GenEval (Ghosh et al., 2023), MIT License
  `https://github.com/djghosh13/geneval`
- DMF XL/2+ (Lee et al., 2025)
  `https://github.com/kyungmnlee/dmf`
- FLUX.1 dev (Black Forest Labs, 2024), Apache-2.0 License
  `https://github.com/black-forest-labs/flux`
- Distilled FLUX Flow Map
  `https://huggingface.co/gabeguofanclub/flux-1-dev-flowmap-lsd`
- RMF (Woo et al., 2026)
  `https://github.com/dongyeop3813/Protein-RMF`
- FAV (Lee et al., 2026), MIT License
  `https://github.com/Jaewoopudding/FAV`
- VFM (Mammadov et al., 2026b), MIT License
  `https://github.com/abbasmammadov/VFM`
- Adjoint Matching (Domingo-Enrich et al., 2025), MIT License
  `https://github.com/microsoft/soc-fine-tuning-sd`