ITERATIVE IMPORTANCE FINE-TUNING OF DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models are an important tool for generative modelling, serving as effective priors in applications such as imaging and protein design. A key challenge in applying diffusion models for downstream tasks is efficiently sampling from resulting posterior distributions, which can be addressed using the h-transform. This work introduces a self-supervised algorithm for fine-tuning diffusion models by estimating the h-transform, enabling amortised conditional sampling. Our method iteratively refines the h-transform using a synthetic dataset resampled with path-based importance weights. We demonstrate the effectiveness of this framework on class-conditional sampling, inverse problems and reward fine-tuning for text-to-image diffusion models.

1 INTRODUCTION

Diffusion models have emerged as a powerful tool for generative modelling (Ho et al., 2020; Dhariwal & Nichol, 2021). As training these models is expensive and requires large amount of data, fine-tuning existing models for new tasks is of interest. In particular, given large pre-trained foundation models such as Stable Diffusion (Rombach et al., 2022) for images or RFDiffusion (Watson et al., 2023) for protein generation, the goal is to use the model as a prior for various downstream applications. For example, in imaging applications, the same diffusion model might be used as a prior for inpainting, deblurring or super-resolution tasks (Rout et al., 2024).

Sampling from the resulting conditional distribution can be interpreted as sampling from a tilted distribution (Domingo-Enrich et al., 2025), where the distribution defined by the pre-trained diffusion model $p_{\text{data}}(\boldsymbol{x})$ is tilted by some reward or likelihood function $r : \mathbb{R}^n \to \mathbb{R}$. Then, we define the tilted distribution as

$$p_{\text{tilted}}(\boldsymbol{x}) \propto p_{\text{data}}(\boldsymbol{x}) \exp\left(\frac{r(\boldsymbol{x})}{\lambda}\right), \tag{1}$$

with a temperature $\lambda > 0$. This general framework includes several applications, such as statistical inverse problems with $r(\boldsymbol{x}) = \ln p(\boldsymbol{y}|\boldsymbol{x})$ as the log-likelihood given observed measurements $\boldsymbol{y}$; class conditional sampling with $r(\boldsymbol{x}) = \ln p(c|\boldsymbol{x})$ as the log-class probabilities for a class c or reward fine-tuning where $r(\boldsymbol{x})$ is learned explicitly as an image reward (Xu et al., 2024). Further, the tilted distribution can also be expressed as the minimiser of an optimisation problem, i.e.,

$$p_{\text{tilted}} = \arg\min_{p} \left\{-\mathbb{E}_{\boldsymbol{x}\sim p}[r(\boldsymbol{x})] + \lambda \mathrm{KL}(p, p_{\text{data}})\right\}. \tag{2}$$

Here, the first term maximises the reward, and the second term acts as a regulariser with the temperature λ as the regularisation strength. In the following, we directly incorporate λ into the reward r for an easier notation. sc We can sample from the tilted distribution by adding an additional term, the generalised h-transform, to the drift function of the reverse SDE (Vargas et al., 2023b). The h-transform is in general intractable in closed form and thus many works propose approximations, see for example (Jalal et al., 2021; Chung et al., 2023; Rout et al., 2024). As an alternative, Denker et al. (2024) propose a supervised framework to estimate the h-transform given a small dataset from the tilted distribution. However, this limits the method to domains where such a dataset is available.

1.1 CONTRIBUTIONS

In order to sample from the tilted distribution, we propose a self-supervised framework for estimating the h-transform without access to samples from the tilted distribution. Starting with an initial estimate of h, our method iterates the following three steps.

1. First, we *sample a batch of trajectories from the diffusion model* with our current estimate of the h-transform.
2. Next, we use path-based importance weights to *filter the dataset* by rejecting samples which do not fit to the tilted distribution. Consequently, the filtered dataset describes the filtered distribution better than the original one. The corresponding importance weights can be computed on the fly during the first step and do not cause additional costs.
3. Finally, we use the supervised fine-tuning loss from Denker et al. (2024) to *update our estimation of the h-transform*.

From a theoretical viewpoint, we prove in Theorem 6 that our iterative importance fine-tuning decreases a certain loss function in each iteration, where the global minimiser of this loss function is known to coincide with the h-transform. Finally, we apply our method for class conditional sampling on MNIST, super-resolution, and reward-based fine-tuning of Stable Diffusion. We emphasise that our importance-based fine-tuning *does not require to differentiate the score-function of the base-model*. Since we fine-tune using the score matching objective, we never have to backpropagate through the generation process. In particular, it can be applied for very large base-models where other methods run out of memory.

1.2 RELATED WORK

Controlled Generation from Diffusion Models Controlled generation for diffusion models can be achieved via inference-time or post-training methods, see the recent review by Uehara et al. (2025). Inference-time methods, such as classifier guidance (Dhariwal & Nichol, 2021) or reconstruction guidance (Chung et al., 2023), guide the reverse diffusion process without additional training but typically increase computational cost and are often sensitive to hyperparameters (Song et al., 2024). Post-training techniques instead fine-tune models for a specific application. Fine-tuning comes with a higher initial computational cost but often results in reduced sampling time compared to inference-time methods (Denker et al., 2024). Supervised post-training methods require an additional task-specific dataset for fine-tuning (Ruiz et al., 2023; Zhang et al., 2023; Xu et al., 2024). In contrast, online post-training methods directly optimise some objective given by the reward function via reinforcement learning (Venkatraman et al., 2024; Clark et al., 2024; Fan et al., 2024; Black et al., 2024) or stochastic optimal control (Denker et al., 2024; Domingo-Enrich et al., 2025; Pidstrigach et al., 2025). Further, diffusion models for sampling from Boltzmann densities via optimal control losses were considered by Albergo et al. (2023); Albergo & Vanden-Eijnden (2024); Berner et al. (2024); Blessing et al. (2025); Vargas et al. (2023a); Zhang & Chen (2022).

Self-supervised fine-tuning Aligning generative models via fine-tuning on their own high-reward outputs offers an alternative to online reinforcement learning. These approaches typically follow an iterative "sample $\rightarrow$ filter $\rightarrow$ fine-tune" process, using a reward function to rank trajectories generated by the model and then updating the model on the top-ranked subset, see e.g., REST (Gulcehre et al., 2023) and RAFT (Dong et al., 2023) for applications to large language models and diffusion models. In particular, REST and RAFT can be viewed as the limit case $\lambda^{-1} \rightarrow \infty$ of the proposed importance fine-tuning algorithm. Further, there are soft filtering approaches, which do not update the model only on the highest-ranking trajectories. Instead soft filtering incorporates all trajectories but weights their contributions according to the reward. This has strong connections to reward-weighted regression (Peters & Schaal, 2007) and has recently been used for fine-tuning text-to-image diffusion models (Lee et al., 2023; Fan et al., 2024).

Iterative retraining can cause model degradation However, iterative retraining of generative models on synthetic data has been shown to lead to performance degradation, including phenomena such as mode collapse (Alemohammad et al., 2024; Shumailov et al., 2023). Strategies to mitigate this issue have been proposed, such as mixing synthetic data with real data (Bertrand et al., 2024). Alternatively, training on curated synthetic datasets, i.e., choosing the synthetic data based on the

reward r, has also been demonstrated to improve retraining stability and model performance (Dong et al., 2023; Ferbach et al., 2024). As we re-sample the synthetic dataset based on the importance weights, our approach also falls into the category of retraining using curated data. However, as opposed to Ferbach et al. (2024) our selection process incorporates both the reward and the path probability, which can mitigate risks associated with iterative retraining.

Self-supervised training for sampling A number of self-supervised frameworks have been proposed for sampling from unnormalised densities, e.g., FAB (Midgley et al., 2023), iDEM (Akhound-Sadegh et al., 2024) or PDDS (Phillips et al., 2024). In iDEM, the authors propose an iterative framework, to train a diffusion model with a stochastic regression loss based on data sampled from the current estimation of the diffusion model. An importance-based acceptance-rejection scheme for improving the quality of generative models for sampling was proposed by Hertrich & Gruhlke (2025).

1.3 Outline

The rest of the paper is structured as follows. In Section 2 we give the necessary background on diffusion models and supervised fine-tuning. The path-based importance weights and the resampling step are presented in Section 3. We present experiments on class conditional sampling for MNIST, super-resolution, and reward-based fine-tuning of Stable Diffusion (Rombach et al., 2022) in Section 4. Conclusions are drawn in Section 5.

2 Background

Let us first recap the score-based generative modelling framework of Song et al. (2021); we start with a forward SDE, which progressively transforms the target distribution p_{data}

$$\mathrm{d}\boldsymbol{X}_t = f_t(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma_t \overrightarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{X}_0 \sim p_{\mathrm{data}}, \tag{3}$$

with drift f_t and diffusion σ_t. As usual, we denote by p_t the density of the solution $\boldsymbol{X}_t$ at time t and by $\vec{p}_{t_1|t_2}(\boldsymbol{x}_{t_1}|\boldsymbol{x}_{t_2})$ the conditional densities of $\boldsymbol{X}_{t_1}$ given $\boldsymbol{X}_{t_2}$. Under some regularity assumptions, there exists by Anderson (1982) a corresponding reverse SDE, that allows sampling from $\mathbb{P}_T$ (typically $\mathcal{N}(0, \mathbf{I}_n)$) and denoising them to generate samples from p_{data}. The reverse SDE is given by

$$\mathrm{d}\boldsymbol{X}_t = \left(f_t(\boldsymbol{X}_t) - \sigma_t^2 s_t(\boldsymbol{X}_t)\right)\,\mathrm{d}t + \sigma_t \overleftarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{X}_T \sim \mathbb{P}_T, \tag{4}$$

where the time flows backwards and $s_t(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \ln p_t(\boldsymbol{x})$ is the score function.

To sample from the tilted distribution p_{tilted} instead of p_{data}, we consider an additional guidance term in the reverse SDE. More precisely, we consider the SDE

$$\mathrm{d}\boldsymbol{H}_t = \left(f_t(\boldsymbol{H}_t) - \sigma_t^2 \left(s_t(\boldsymbol{H}_t) + h_t(\boldsymbol{H}_t)\right)\right)\,\mathrm{d}t + \sigma_t \overleftarrow{\mathrm{d}\mathbf{W}}_t, \tag{5}$$

where the time flows again backwards and we use $\boldsymbol{H}_t$ for the guided reverse SDE. To ensure that $\boldsymbol{H}_0 \sim p_{\mathrm{tilted}}$, Denker et al. (2024); Vargas et al. (2023b) represent the scores with respect to p_{tilted} as the sum $s_t(\boldsymbol{x}) + h_t(\boldsymbol{x})$, where h is a generalisation of Doob's h-transform. It can be computed by the next theorem. Note that Zhao et al. (2025) prove a similar results to Theorem 1 (iii) from a reinforcement learning perspective.

Theorem 1 (Proposition 2.2 and Theorem 3.1 in Denker et al., 2024)**.**
Assume $Z_r := \int \exp(r(\boldsymbol{x}_0))\mathrm{d}\boldsymbol{x}_0 < \infty$. *Further, let* $Q_T^{f_t}[p_{\mathrm{tilted}}] := \int \vec{p}_{T|0}(\boldsymbol{x}|\boldsymbol{x}_0)p_{\mathrm{tilted}}(\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_0$ *and*

$$h_t^*(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \ln p_t^r(\boldsymbol{x}), \quad \textit{where} \quad p_t^r(\boldsymbol{x}) = \int \frac{\exp(r(\boldsymbol{x}_0))}{Z_r} \overleftarrow{p}_{0|t}(\boldsymbol{x}_0|\boldsymbol{x})\mathrm{d}\boldsymbol{x}_0. \tag{6}$$

Then, the following holds true:

(i) Let $(\boldsymbol{H}_t)_t$ *be the solution of the SDE* (5) *with* $h_t = h_t^*$ *given in* (6) *and* $\boldsymbol{H}_T \sim Q_T^{f_t}[p_{\mathrm{tilted}}]$. *Then, it holds* $\boldsymbol{H}_0 \sim p_{\mathrm{tilted}}$.

(ii) It holds that h_t^* *is the unique minimiser of the score-matching (SM) loss function*

$$\mathcal{L}_{SM}(h_t) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_0 \sim p_{\mathrm{tilted}} \\ t \sim \mathrm{U}(0,T), \boldsymbol{x}_t \sim \vec{p}_{t|0}(\cdot|\boldsymbol{x}_0)}} \left[\left\| (h_t(\boldsymbol{x}_t) + s_t(\boldsymbol{x}_t)) - \nabla_{\boldsymbol{x}_t} \ln \vec{p}_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) \right\|^2 \right]. \tag{7}$$

(iii) It holds that

$$h_t^* \in \arg\min_{h_t} \mathcal{F}(\mathbb{P}_h), \quad \textit{where} \quad \mathcal{F}(\mathbb{P}) = \mathrm{KL}(\mathbb{P}, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}}[r(\boldsymbol{x}_0)] \tag{8}$$

and where $\mathbb{P}_h$ and $\mathbb{P}_{\mathrm{data}}$ are the path measures of the corresponding SDEs (5) and (4).

Remark 2. *In Theorem 1 (i) the terminal distribution is given as $Q_T^{f_t}[p_{\mathrm{tilted}}]$ instead of $\mathbb{P}_T$ in Equation (4) to remove the value function bias. In Domingo-Enrich et al. (2025), the authors deal with the value function bias by altering the noise of the controlled SDE. However, due to the mixing time in the commonly used VP-SDE the discrepancy, measured w.r.t. the total variation distance, decays exponentially (Denker et al., 2024, Proposition G.2) and we approximate $Q_T^{f_t}[p_{\mathrm{tilted}}] \approx \mathbb{P}_T \approx \mathcal{N}(0, \mathbf{I}_n)$ for all numerical experiments.*

Part (i) of Theorem 1 states conditions on h_t such that $\boldsymbol{H}_0 \sim p_{\mathrm{tilted}}$ in (5) and the (ii), (iii) provide loss functions to learn such a h_t. Minimising $\mathcal{F}(\mathbb{P}_h)$ in (8) can be reformulated as a stochastic control (SC) loss, i.e.,

$$\mathcal{L}_{SC}(h_t) = \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{1}{2}\int_0^T \sigma_t^2 \|h_t(\boldsymbol{x}_t)\|^2 \mathrm{d}t - r(\boldsymbol{x}_0)\right]. \tag{9}$$

Differentiating $\mathcal{L}_{SC}$ requires differentiating through trajectories of the SDE. While the computational cost of this step can be reduced by using approaches such as VarGrad (Richter et al., 2020) or trajectory balance (Malkin et al., 2022), the memory requirements still scale linearly with the number of time steps in the trajectory. Similar objectives have been considered from a reinforcement learning perspective, e.g. Fan et al. (2024) uses policy gradient methods (Schulman et al., 2017).

In contrast, the score-matching loss in part (ii) is computationally much lighter. It avoids simulating SDE trajectories and reduces to a simple mean squared error objective. However, it requires access to samples from the tilted distribution, the very distribution we want to sample from. In the next section, we develop a method that allows us to exploit the computational advantages of score-matching even when such ground truth samples are unavailable.

Remark 3. *The connection of training diffusion models to stochastic optimal control have also been successfully exploited for sampling from unnormalised probability density functions, see e.g. Zhang & Chen (2022); Nüsken & Richter (2021); Berner et al. (2024). Further, the proof for Theorem 1 (iii) relies on connections to Schrödinger half-bridges (Heng et al., 2021; Vargas et al., 2021).*

3 Self-Supervised Importance Fine-tuning

To circumvent the limitations of the supervised score-matching loss in (7), we propose a self-supervised method for fine-tuning diffusion models. For this, we combine the recently proposed rejection sampling steps from Hertrich & Gruhlke (2025) with the supervised fine-tuning from Theorem 1 (ii) and prove that this defines a descent algorithm for the loss function $\mathcal{F}$ from Theorem 1 (iii).

3.1 Importance-based Rejection

A basic tool for reweighting and fine-tuning generative models are importance weights. Concretely, let $\boldsymbol{x}_0$ be sampled from the backward SDE (5) with guidance term h. The corresponding weight is given by $\frac{p_{\mathrm{tilted}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)}$, where p_h is the density of the generated distribution. Direct evaluation of p_h is intractable. Using the factorisation $p_{\mathrm{tilted}}(\boldsymbol{x}_0) \propto p_{\mathrm{data}}(\boldsymbol{x}_0)\exp(r(\boldsymbol{x}_0))$, we can rewrite the weight as

$$\frac{p_{\mathrm{tilted}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)} = \frac{p_{\mathrm{tilted}}(\boldsymbol{x}_0)}{p_{\mathrm{data}}(\boldsymbol{x}_0)} \cdot \frac{p_{\mathrm{data}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)} \propto \exp(r(\boldsymbol{x}_0))\frac{p_{\mathrm{data}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)}. \tag{10}$$

The key term $\frac{p_{\mathrm{data}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)}$ can be linked to path measures of the SDE via the following lemma.

Lemma 4. *Assume that $\mathbb{P}_{data}$ and $\mathbb{P}_h$ are path measures of the solution for the SDE $\mathrm{d}\boldsymbol{X}_t = f_t(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma_t \mathrm{d}\overrightarrow{\mathbf{W}}_t$ with initial conditions $\boldsymbol{X}_0 \sim p_{data}$ and $\boldsymbol{X}_0 \sim p_h$, respectively. We have*

$$\frac{p_{data}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)} = \frac{\mathrm{d}\mathbb{P}_{data}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}). \tag{11}$$

Algorithm 1 Self-supervised estimation of the h-transform for DDPM

Require: Pre-trained noise prediction model $\epsilon_t^\theta(\boldsymbol{x}_t)$, noise schedule $\bar{\alpha}_t$
Require: Number of outer training steps K, number of inner training steps M, number of samples N, batch size m, buffer with maximum length

for $k = 1$ to K **do**
  Sample $\boldsymbol{x}_{[0:T]}^{(1:N)} \sim \mathbb{P}_{h_\varphi}$ ▷ Sample paths with current model
  for $i = 1$ to N **do** ▷ Resample using rejection sampling
    Compute acceptance probability $\alpha(\boldsymbol{x}_{[0:T]}^{(i)})$
    Sample $u \sim U([0,1])$
    If $u \leq \alpha(\boldsymbol{x}_{[0:T]}^{(i)})$, add $\boldsymbol{x}_{[0:T]}^{(i)}$ to buffer
  for 1 to M **do** ▷ Inner training loop using buffer
    Sample $\boldsymbol{x}^{(1:m)}$ from buffer
    $t \sim U(\{1, \dots, T\})$
    $\epsilon^{(1:m)} \sim \mathcal{N}(0, I)$
    $\boldsymbol{x}_t^{(1:m)} \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{x}^{(1:m)} + \sqrt{1-\bar{\alpha}_t}\epsilon$
    $\ell \leftarrow \left\| \left((\epsilon_t^\theta(\boldsymbol{x}_t^{(1:m)}) + h_t^\varphi(\boldsymbol{x}_t^{(1:m)}) \right) - \epsilon^{(1:m)} \right\|_2^2$
    $\varphi \leftarrow \text{Optim}(\varphi, \nabla_\varphi \ell)$

See Appendix B.1 for the proof. The assumptions are in particular satisfied if the guidance h minimises a score-matching loss function such as (7). Given Lemma 4 we have

$$\frac{p_{\text{tilted}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)} \propto \exp(r(\boldsymbol{x}_0)) \frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}), \tag{12}$$

where $\boldsymbol{x}_{[0:T]}$ denotes the full trajectory. The Radon–Nikodym derivative (RND) can be computed explicitly using the framework of Vargas et al. (2024), leading to the following result. The derivations are included in Appendix A, both in continuous and in discrete time.

Lemma 5. *For $x_{[0:T]} \sim \mathbb{P}_h$, we have*

$$\frac{p_{tilted}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)} \propto \exp\left(r(\boldsymbol{x}_0) - \frac{1}{2}\int_0^T \sigma_t^2 \|h_t(\boldsymbol{x}_t)\|_2^2 dt + \int_0^T \sigma_t h_t(\boldsymbol{x}_t)^\top dW_t \right). \tag{13}$$

These path-wise importance weights can be computed concurrently with sampling without any computational overhead, using the same time-discretiation for the integrals as in the SDE solver. For numerical stability, we perform these computations in the log-space. In practice, importance weights can be highly imbalanced. As a remedy, Hertrich & Gruhlke (2025) proposed a rejection sampling algorithm based on relaxed importance weights. Plugging in our approximation (12), this rejection sampling step amounts to computing, for some generated trajectory $\boldsymbol{x}_{[0,T]}$, the acceptance probability

$$\alpha(\boldsymbol{x}_{[0:T]}) = \min\left(1, \frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}) \frac{\exp(r(\boldsymbol{x}_0))}{c}\right), \tag{14}$$

where c is a hyper-parameter. Based on the observation that $\alpha(\boldsymbol{x}_{[0:T]})$ is decreasing in c, Hertrich & Gruhlke (2025, Rem 4.4) suggest to find c such that $\mathbb{E}[\alpha(\boldsymbol{x}_{[0:T]})] = \rho$ for some prescribed acceptance rate ρ. Such a c can easily be found by some bisection search. In this way it is always ensured that a sufficient amount of samples is accepted, which circumvents one of the major drawbacks from rejection sampling with importance weights. The acceptance ratio ρ is chosen for each experiment separately between 0.1 and 0.6. We will show in the next subsection that the distribution of accepted samples is closer to the tilted distribution than the distribution of the initially generated samples.

3.2 Self-supervised Importance Fine-Tuning

Next, we will combine the importance-based rejection steps with the supervised loss function from Theorem 1 (ii) to obtain a tractable fine-tuning algorithm. More precisely, starting with an initial guidance term h^0 we construct a sequence of guidance terms h^k for $k = 1, 2, \dots$ by the following

steps. First, we sample a batch $\{\boldsymbol{x}_0^{(i)}\}_{i=1,\dots,N}$ from the reverse SDE (5) with $h = h^k$ and compute the corresponding acceptance probabilities $\alpha_i = \alpha(\boldsymbol{x}_{[0,T]}^{(i)})$ from (14). Second, we keep any sample from the batch with probability α_i and discard the rest. We denote the distribution of remaining samples by $\tilde{\mathbb{P}}^0_{h^k}$. Finally, we update the guidance term h^k by using the supervised loss function for the h-transform from Theorem 1 (ii). That is, we define $h^{k+1} \in \arg\min_{h_t} \mathcal{L}_{FT}(h_t)$, where

$$\mathcal{L}_{FT}(h_t) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_0 \sim \tilde{\mathbb{P}}^0_{h^k} \\ t \sim \mathrm{U}(0,T), \boldsymbol{x}_t \sim \vec{p}_{t|0}(\cdot|\boldsymbol{x}_0)}} \left[\left\| (h_t(\boldsymbol{x}_t) + s_t(\boldsymbol{x}_t)) - \nabla_{\boldsymbol{x}_t} \ln \vec{p}_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) \right\|^2 \right]. \tag{15}$$

We summarise our self-supervised importance fine-tuning in Algorithm 1 for the discrete DDPM framework (Ho et al., 2020). Our methods belongs thus to the family of cross-entropy methods which iteratively fine-tune a model based on self-generated samples (De Boer et al., 2005).

Recall that the generalised h-transform from Theorem 1 minimises the function $\mathcal{F}(\mathbb{P}_h) = \mathrm{KL}(\mathbb{P}_h, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]} \sim \mathbb{P}_h}[r(\boldsymbol{x}_0)]$. The following theorem proves that our self-supervised importance fine-tuning is a descent algorithm for $\mathcal{F}$. More precisely, it holds that $\mathcal{F}(\mathbb{P}_{h^{k+1}}) \leq \mathcal{F}(\mathbb{P}_{h^k})$ for all $k = 1, 2, \dots$. We include the proof in Appendix B.2.

Theorem 6. *Let $\alpha(\boldsymbol{x}_{[0:T]}) = \min\left(1, \frac{\mathrm{d}\mathbb{P}_{data}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}) \frac{\exp(r(\boldsymbol{x}_0))}{c}\right)$ be the acceptance probability of a sample $\boldsymbol{x}_{[0,T]} \sim \mathbb{P}_h$ and denote by $\tilde{\mathbb{P}}_h$ the distribution of the accepted paths. Then, the following holds true:*

(i) $\frac{\mathrm{d}\tilde{\mathbb{P}}_h}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}) = \frac{\alpha(\boldsymbol{x}_{[0,T]})}{\mathbb{E}_{\boldsymbol{x}_{[0,T]} \sim \mathbb{P}_h}[\alpha(\boldsymbol{x}_{[0,T]})]}$

(ii) $\mathcal{F}(\tilde{\mathbb{P}}_h) \leq \mathcal{F}(\mathbb{P}_h)$, *where*

$$\mathcal{F}(\mathbb{P}) = \mathrm{KL}(\mathbb{P}, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]} \sim \mathbb{P}}[r(\boldsymbol{x}_0)] = \mathrm{KL}(\mathbb{P}, \mathbb{P}_{\mathrm{tilted}}) + \mathrm{const}$$

(iii) Let $h_t^* \in \arg\min_{g_t} \mathcal{L}_{FT}(g_t)$ with $\mathcal{L}_{FT}$ from (15) and let $\mathbb{P}_{h^*}$ be the path measure of the corresponding SDE. Then it holds $\mathcal{F}(\mathbb{P}_{h^*}) \leq \mathcal{F}(\tilde{\mathbb{P}}_h) \leq \mathcal{F}(\mathbb{P}_h)$.

3.3 Training and Network Parametrisation

Replay buffer Sampling from the current model is the most computationally expensive part of the algorithm. Inspired by Midgley et al. (2023); Sendera et al. (2024), we employ an un-prioritised replay buffer of fixed size. We store the accepted samples from the current model and add them to the buffer. Once the buffer reaches its maximum capacity, a portion of samples is removed. The samples selected for removal from the buffer can either be chosen randomly or as the oldest entries. In the random selection scenario, the replay buffer acts similarly to a moving average method, and a similar conclusion to Theorem 6 is applicable. Nevertheless, in practice, we choose to remove the oldest samples, which produces comparable results and achieves faster convergence. Additional details are provided in Appendix C. To minimise the fine-tuning loss (15), we randomly draw batches from the buffer.

Network parametrisation The parametrisation of the h-transform has a crucial effect on performance and convergence speed (He et al., 2025). Motivated by the network parametrisation in sampling applications with diffusion (Vargas et al., 2023a; Zhang & Chen, 2022) and fine-tuning approaches (Denker et al., 2024; Venkatraman et al., 2024), we make use of a *reward-informed inductive bias* given as

$$h_t^\theta(\boldsymbol{x}_t) = \mathrm{NN}_1(\boldsymbol{x}_t, t) + \mathrm{NN}_2(t) \nabla_{\hat{\boldsymbol{x}}_0} r(\hat{\boldsymbol{x}}_0), \tag{16}$$

where $\hat{\boldsymbol{x}}_0$ is the Tweedie estimate given the pre-trained unconditional diffusion model, NN_1 is a vector-valued and NN_2 a scalar-valued neural network. We initialise the last layer of NN_1 to be zero and NN_2 to be constant. However, for text-to-image diffusion models we instead make use of prior work, see e.g. (Venkatraman et al., 2024; Fan et al., 2024), and use the parameter efficient LoRA method (Hu et al., 2022).

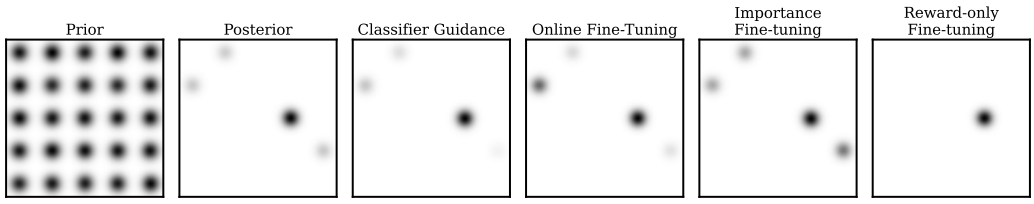


Figure 1: 2D GMM example for fine-tuning diffusion models. We show the prior the log-probability of the tilted distribution and samples from the conditional diffusion model using classifier guidance (Dhariwal & Nichol, 2021), online fine-tuning (Fan et al., 2024), our importance fine-tuning and a variation with only reward-based importance weights. For quantitative results see 5 in the Appendix.

KL-Regularisation Similar to Fan et al. (2024), we found that using a KL regulariser was useful to further improve diversity in the supervised setting. The KL divergence between p_h and p_{data} can be bounded as

$$D_{\text{KL}}(p_h, p_{\text{data}}) \leq D_{\text{KL}}(\mathbb{P}_h, \mathbb{P}_{\text{data}}) = \mathbb{E}_{\boldsymbol{H}_t \sim \mathbb{P}_h}\left[\int_0^T \sigma_t^2 \|h_t(\boldsymbol{H}_t)\|_2^2 dt\right], \tag{17}$$

i.e., the norm of the h-transform over trajectories, see Appendix A for a derivation. The naive addition of this term to the supervised training loss is expensive, as one has to backpropagate through the full trajectory. Instead, we estimate the integral using a single random time step.

4 Experiments

In this section, we present a toy example on a 2D dataset, class conditional sampling for MNIST, super-resolution, and finally results for reward fine-tuning of text-to-image diffusion models. The code is available[1].

4.1 2D Toy example

As an initial example we make use of a diffusion trained on samples of a Gaussian mixture model (GMM) with 25 modes, arranged on a grid. The goal is to sample from the tilted distribution $p_{\text{data}}(\boldsymbol{x}) \exp(r(\boldsymbol{x}))$, where the reward r is defined as the log-likelihood of a GMM with a reweighted subset of the modes, see Appendix D.1 for the detailed setup. Figure 1 illustrates both the prior and the density of the tilted distribution. We compare against Classifier Guidance (Dhariwal & Nichol, 2021), an inference time method, where the h-transform is approximated by the gradient of the reward $h_t(\boldsymbol{x}_t) = \gamma \nabla_{\boldsymbol{x}_t} r(\boldsymbol{x}_t)$. Further, we compare against online fine-tuning, where we directly optimise Equation (2). Further, we compare against our approach where importance weights are calculated solely based on the reward, referred to as reward-only fine-tuning.

In Figure 1, we observe that while classifier guidance is able to find all the modes of the posterior, it fails to capture the correct weighting. In contrast, both online fine-tuning and importance fine-tuning yield samples that more accurately represent the target distribution. Lastly, the reward-only fine-tuning method collapses to the highest mode.

4.2 Posterior Sampling for Inverse Problems

We consider the task of posterior sampling in inverse problems. Here, the reward function r is given by the likelihood $p^{\text{lkhd}}(\boldsymbol{y}|\boldsymbol{x})$, i.e., the conditional distributions of measurements $\boldsymbol{y}$ given images $\boldsymbol{x}$. We focus on the linear inverse problem of super-resolution, where the measurements are given as $\boldsymbol{y}^\delta = \boldsymbol{A}\boldsymbol{x} + \eta$. Here, $\boldsymbol{A}$ is the downsampling operator, $\eta \sim \mathcal{N}(0, \sigma_y^2 \mathbf{I})$ represents additive Gaussian noise, and the likelihood is given as $r(\boldsymbol{x}; \boldsymbol{y}^\delta) = \ln p^{\text{lkhd}}(\boldsymbol{y}^\delta|\boldsymbol{x}) = -\frac{1}{2\sigma_y^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}^\delta\|_2^2$.

[1] https://anonymous.4open.science/r/IterativeImportanceFinetuning-4606

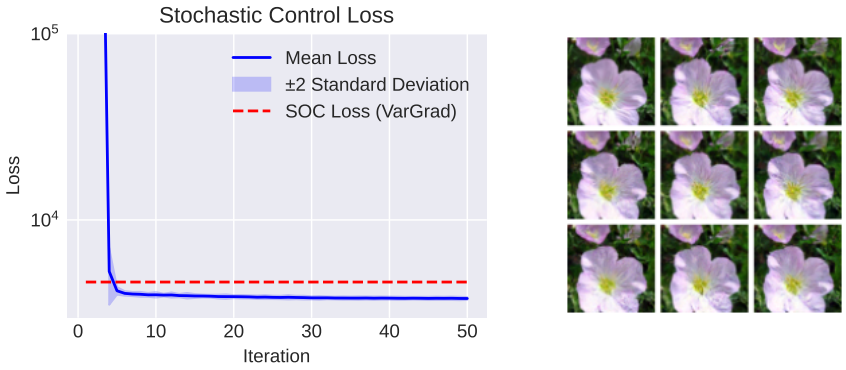
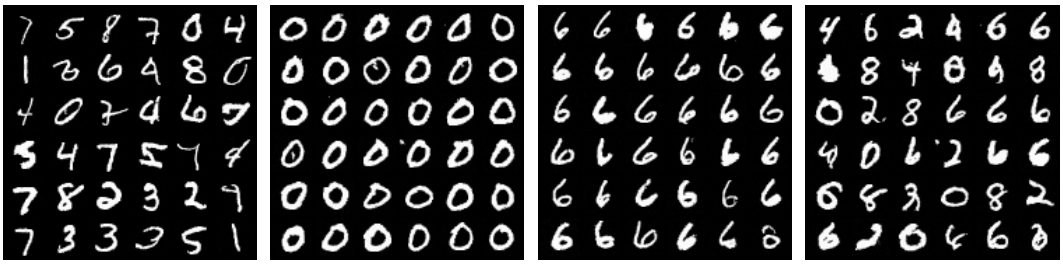


Figure 2: Posterior sampling for `Flowers`. Left: We show decay of the SC loss function $\mathcal{L}_{\text{SC}}(h_k)$ over fine-tuning iterations k. We show the mean loss over 10 independent training runs with ± 2 std intervals. We compare with VarGrad Richter et al. (2020) for minimising the SOC loss. Right: The upper left image is the ground truth image and the remaining 8 images are samples from the model.



Prior samples | Posterior: Zero | Posterior: Six | Posterior: Even numbers

Figure 3: Class conditional sampling for MNIST. Left: Sampled from unconditional model. Middle: Samples for class 'six'. Right: Samples for even numbers.

We train a diffusion model on the `Flowers` dataset (Nilsback & Zisserman, 2008) and consider a $2\times$ super-resolution task with 5% relative Gaussian noise. For the h-transform we make use of the reward-informed architecture (16), see Appendix D.2 for the detailed setup. Theorem 6 demonstrates that iterative importance fine-tuning functions as a descent algorithm for the SC loss $\mathcal{L}_{\text{SC}}$. Figure 2 presents both the SC loss across iterations and the final model's sample outputs, confirming that iterative fine-tuning operates as a descent algorithm for the SC loss in this scenario. We present additional qualitative results for different ground truth images in Appendix D.2.

4.3 Class-conditional sampling

As another application, we consider class conditional sampling for MNIST. Let c be the class and $p(c|\boldsymbol{x})$ the log class probabilities of a pre-trained classifier. The goal is to sample from the tilted distribution with the reward as $r(\boldsymbol{x}) = \ln p(c|\boldsymbol{x})$. We use a convolutional classifier with a 98.6% accuracy and parametrise the h-transform using the reward-informed architecture as in Eqn. (16).

We use importance fine-tuning to learn the posterior for different MNIST classes and for even/odd numbers. Posterior prior samples are shown in Figure 3, see also Figure 7 in the Appendix for all classes. We compare both against classifier guidance and online fine-tuning. In Table 1 we compare the expected reward and the FID (Heusel et al., 2017) based on the features of the penultimate layer of the pre-trained classifier. Here, we compare against classifier guidance and online fine-tuning, i.e., directly minimising (9) using VarGrad (Richter et al., 2020).

Table 1: Expected reward and FID scores for class conditional sampling on MNIST.

Method	**Single class**		**Even / Odd**	
	$\mathbb{E}[r(\boldsymbol{x})]$ (↑)	FID (↓)	$\mathbb{E}[r(\boldsymbol{x})]$ (↑)	FID (↓)
Classifier Guid.	-1.495	117.241	-2.572	193.984
Online FT	-0.110	31.994	-0.065	198.871
Importance FT	-0.152	30.814	-0.878	110.403

Table 2: Accuracy and FID score for class conditional sampling on MNIST with the non-smooth reward function. For Top-K we draw 100 independent samples.

Class	0		2		4		6		8	
	Accuracy	FID	Accuracy	FID	Accuracy	FID	Accuracy	FID	Accuracy	FID
Top-K ($k = 10$)	61.00	489.6	53.00	642.5	67.00	381.7	52.00	797.3	61.00	320.3
Top-K ($k = 20$)	82.00	200.7	71.00	347.0	87.00	98.70	71.00	375.8	76.00	192.9
Top-K ($k = 40$)	96.00	43.80	88.00	161.4	100	15.74	95.00	62.39	96.00	34.93
Importance FT (smooth reward)	98.24	38.53	93.65	49.78	96.97	16.40	91.21	28.32	93.06	25.36
Importance FT (non-smooth reward)	94.43	32.84	92.00	47.47	95.61	19.69	90.72	58.02	89.45	28.61

Online fine-tuning is able to achieve a higher expected reward both for single-class posterior and for the even/odd tasks. However, this comes at the expense of the FID. Here, our importance fine-tuning is able to obtain the best FID scores. In Appendix D.3 we include an ablation regarding the temperature λ, showing a trade-off between maximising the reward and keeping diversity.

Non-Smooth Reward Function We further evaluate our approach under a non-differentiable reward. In the previous experiment, the reward was given by the log class probability, which provides smooth and informative gradients that can be exploited by classifier guidance and other gradient-based fine-tuning procedures. Here, we instead define a binary reward that depends only on the predicted class

$$r(x) = \begin{cases} 1, & \text{if } c = \arg\max_{c'} \ln p(c'|x) \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

This reward is piecewise constant, and its gradient with respect to x is zero almost everywhere. Consequently, methods that rely on the reward gradient can not be applied. The reward-informed inductive bias from (16) is not applicable and we parametrise the h-transform as a time-dependent neural network, i.e.. $h_t^\theta(\boldsymbol{x}_t) = \mathrm{NN}_1(\boldsymbol{x}_t, t)$. Table 2 reports accuracy and FID for a subset of MNIST classes. Importance FT trained with the non-smooth reward achieves performance comparable to its smooth-reward counterpart, with modest degradation in some classes but remaining in the same overall range. In contrast, Top-K sampling only approaches similar performance when using very large k (e.g. $k = 40$), and even then its results are inconsistent across classes. These results highlight that our approach does not rely on differentiability of the reward, but can make use of it by using the reward-informed inductive bias for the network architecture.

4.4 Text-to-Image Reward Fine-tuning

Despite the progress in training text-to-image diffusion models, samples do not always align with human preferences. In particular, the model can struggle with unnatural prompts such as "A green colored rabbit" (Venkatraman et al., 2024; Fan et al., 2024). A common solution is to fine-tune the model to maximise a reward signal trained to approximate human judgments. We use the latent diffusion model Stable Diffusion-v1.5 (Rombach et al., 2022) and align it using the ImageReward-v1.0 (Xu et al., 2024) reward model, see Appendix D.4 for experimental details.

As baselines, we include Top-K sampling ($K = 6$), which selects the best of K generated candidates, at the cost of increasing sampling time by a factor of K. Further, we include a comparison to FK-Steering (Singhal et al., 2025) for an inference-time control method. We compare against recent fine-tuning approaches: DPOK (Fan et al., 2024), RTB (Venkatraman et al., 2024), and adjoint matching (Domingo-Enrich et al., 2025). DPOK, RTB, and our method adopt the LoRA parametrization, while adjoint matching typically fine-tunes the full model. We also report a LoRA variant of adjoint matching. Results are presented in Table 4 and images are shown in Figure 4, see also Appendix D.4 for additional results. We present both the mean reward and the diversity, measured as one minus the mean cosine similarity of CLIP embeddings. Across three representative prompts, all fine-tuning methods improve reward relative to the base model and Top-K sampling. Adjoint matching achieves the highest reward overall (e.g., 1.706 on "A green colored rabbit"), but at the expense of diversity (dropping to 0.050) and with higher memory requirements, i.e., adapting the full model instead of a low-rank parametrisation. Importance fine-tuning reaches a competitive reward with similar diversity and the same $O(B)$ memory footprint as DPOK. Further, similar to RTB and DPOK our approach only requires access to evaluations of the reward without using the gradient. Overall, importance fine-tuning offers more of a middle ground; achieving a lower reward than adjoint matching but with a more efficient and scalable training pipeline, making it an option for researchers without access to large-scale compute.

Base Model | DPOK | RTB | Adjoint Matching | Importance FT

Figure 4: Samples for the base model, DPOK, Adjoint Matching and our importance FT for the prompt "A green colored rabbit.". Images were generated using the same seed.

Table 4: Results for text-to-image. We compute the mean reward and diversity over 100 samples. For the GPU memory B refers to the batch size and T to the number of time steps during sampling.

	"A green colored rabbit."		**"Two roses in a vase."**		**"Two dogs in the park."**		$\nabla r(x)$ free	GPU memory
	Reward (↑)	Diversity (↑)	Reward (↑)	Diversity (↑)	Reward (↑)	Diversity (↑)		(training)
Base Model	−0.207	0.154	1.130	0.108	0.518	0.193	✓	N/A
Top-K Sampling ($K = 6$)	1.334	0.149	1.619	0.121	0.970	0.161	✓	N/A
FK-Steering ($K = 6$)	1.207	0.147	1.645	0.108	0.985	0.139	✓	N/A
DPOK	1.621	0.065	1.592	0.106	1.013	0.144	✓	$O(B)$
RTB	1.435	0.092	1.501	0.127	1.058	0.174	✓	$O(BT)$
Adjoint Matching	1.706	0.050	1.504	0.088	1.325	0.132	✗	$O(B)$
Adjoint Matching (LoRA)	1.435	0.092	1.230	0.137	0.780	0.162	✗	$O(B)$
Importance FT (ours)	1.462	0.051	1.534	0.077	1.010	0.120	✓	$O(B)$

Table 3: Results for varying the temperature λ for reward fine-tuning for the prompt "A green colored rabbit.".

λ^{-1}	0.1	1.0	10	20	100	∞
Reward (↑)	0.443	0.750	1.235	1.462	1.350	0.622
Diversity (↑)	0.101	0.089	0.069	0.051	0.054	0.113

Finally, we study the effect of varying the temperature parameter λ for the reward in the importance weights, see Table 3. For $\lambda^{-1} = \infty$ the importance weights are equal to the reward which recovers the setting from Dong et al. (2023). The results show that $\lambda^{-1} = 20$ yields the best trade-off of reward and diversity. Too small or too large values lead to either weak alignment or mode collapse. Finally, for reward-only learning we do not observe reward hacking, but instead a deteriorating image quality, leading to lower rewards, as also observed in Fan et al. (2024).

A key issue is how to handle the classifier-free guidance scale during fine-tuning. RL-based methods typically fix a scale and fine-tune the combined model directly. In contrast, our score-matching loss is independent of the guidance scale and we have to choose the guidance scale at sampling time. The default value of Stable Diffusion sometimes produces oversaturated outputs. In Fig. 13 (Appendix D.4) we show that tuning the guidance scale can yield cleaner, more natural images.

5 Conclusion

We propose an iterative approach for fine-tuning diffusion models for conditional sampling tasks. As the naive iterative retraining of generative models often lead to performance degradation (Shumailov et al., 2023), we introduce an additional resampling step based on path-based importance weights. We present experiments for class conditional sampling, inverse problems and reward fine-tuning of text-to-image diffusion models. Online fine-tuning methods often require additional tricks to increase training stability, e.g., Venkatraman et al. (2024) use loss clipping and disregard low reward trajectories or Fan et al. (2024) employ variance reduction techniques by additionally learning the value function. In contrast our importance fine-tuning method is trained using a score matching loss, see Theorem 1, leading to a more stable training.

Limitations In our approach, the diffusion model is fine-tuned using "good" samples, i.e., having a high importance weight. However, these samples necessarily lie in the support of the pre-trained model and we rely on the fact that such "good" samples exist. Thus, our refinement approach might struggle on domains where the distribution of the pre-trained model is sparse and high reward samples are. One possibility to alleviate this problem is to make use of *off-policy* samples (Venkatraman et al., 2024), which are obtained by some other method. For off-policy samples, we loose the compact formulation of the RND from Lemma 5. However, we can still compute importance weights using the RND in Proposition 7, see Appendix A.

Reproducibility Statement

For any theoretical claims and derivations in the paper, we provide proofs in the Appendices A and B. Moreover, we provide anonymized source code and instructions to reproduce our experiments in Section 4 at https://anonymous.4open.science/r/IterativeImportanceFinetuning-4606. A non-anonymized version will be released upon acceptance. Moreover, we describe the necessary implementation details and hyperparameters in Appendix D. All used data sets and models are publicly available and referenced in the paper. The final results might slightly vary due to the randomness in the algorithm.

References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from boltzmann densities. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=gVjMwLDFoQ.

Michael S Albergo and Eric Vanden-Eijnden. Nets: A non-equilibrium transport sampler. *arXiv preprint arXiv:2410.02711*, 2024.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go MAD. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ShjMHfmPs0.

Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=oYIjw37pTP.

Quentin Bertrand, Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. On the stability of iterative retraining of generative models on their own data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JORAfH2xFd.

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. URL https://openreview.net/forum?id=YCWjhGrJFD.

Denis Blessing, Julius Berner, Lorenz Richter, and Gerhard Neumann. Underdamped diffusion bridges with applications to sampling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Q1QTxFm0Is.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

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. URL https://openreview.net/forum?id=1vmSEVL19f.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon V Mathis, Riccardo Barbano, Vincent Dutordoir, Emile Mathieu, Urszula Julia Komorowska, and Pietro Lio. DEFT: Efficient fine-tuning of diffusion models by learning the generalised h-transform. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=AKBTFQhCjm.

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

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.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Damien Ferbach, Quentin Bertrand, Joey Bose, and Gauthier Gidel. Self-consuming generative models with curated data provably optimize human preferences. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=cyv0LkIaoH.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Jiajun He, Yuanqi Du, Francisco Vargas, Dinghuai Zhang, Shreyas Padhy, RuiKang OuYang, Carla Gomes, and José Miguel Hernández-Lobato. No trick, no treat: Pursuits and challenges towards simulation-free training of neural samplers. *arXiv preprint arXiv:2502.06685*, 2025.

Jeremy Heng, Valentin De Bortoli, Arnaud Doucet, and James Thornton. Simulating diffusion bridges with score matching. *arXiv preprint arXiv:2111.07243*, 2021.

Johannes Hertrich and Robert Gruhlke. Importance corrected neural JKO sampling. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=yQBZZeWPdQ.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Christian Léonard. Some properties of path measures. *Séminaire de Probabilités XLVI*, pp. 207–230, 2014.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XCTVFJwS9LJ.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Nikolas Nüsken and Lorenz Richter. Solving high-dimensional hamilton–jacobi–bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2(4):48, 2021.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 40688–40724. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/phillips24a.html.

Jakiw Pidstrigach, Elizabeth Baker, Carles Domingo-Enrich, George Deligiannidis, and Nikolas Nüsken. Conditioning diffusions using malliavin calculus. *arXiv preprint arXiv:2504.03461*, 2025.

Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco Ruiz, and Omer Deniz Akyildiz. Vargrad: a low-variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems*, 33:13481–13492, 2020.

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.

Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22500–22510, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. Improved off-policy training of diffusion samplers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=vieIamY2Gi.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Jp988ELppQ.

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=j8hdRqOUhN.

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

Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Reward-guided controlled generation for inference-time alignment in diffusion models: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.

Francisco Vargas, Will Sussman Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=8pvnfTAbu1f.

Francisco Vargas, Andrius Ovsianas, David Fernandes, Mark Girolami, Neil D Lawrence, and Nikolas Nüsken. Bayesian learning via neural schrödinger–föllmer flows. *Statistics and Computing*, 33(1):3, 2023b.

Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PP1rudnxiW.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, Alexandre Adam, Jarrid Rector-Brooks, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Amortizing intractable inference in diffusion models for vision, language, and control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gVTkMsaaGI.

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, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 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. *Advances in Neural Information Processing Systems*, 36, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=_uCb2ynRu7Y.

Yulai Zhao, Masatoshi Uehara, Gabriele Scalia, Sunyuan Kung, Tommaso Biancalani, Sergey Levine, and Ehsan Hajiramezanali. Adding conditional control to diffusion models with reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=svp1EBA6hA.

A RADON-NIKODYM DERIVATIVE BETWEENS SDES.

To calculate the approximated importance weights, we make use of the RND between SDEs. In particular, we use the result from (Nüsken & Richter, 2021; Vargas et al., 2024).

Proposition 7. *(RND between SDEs (Nüsken & Richter, 2021; Vargas et al., 2024)) Given the following SDEs*

$$\mathrm{d}\boldsymbol{Y}_t = a_t(\boldsymbol{Y}_t)\,\mathrm{d}t + \sigma_t(\boldsymbol{Y}_t)\,\overrightarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{Y}_0 \sim \mu, \tag{19}$$

$$\mathrm{d}\boldsymbol{X}_t = b_t(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma_t(\boldsymbol{X}_t)\,\overrightarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{X}_0 \sim \nu, \tag{20}$$

with path probabilities $\overrightarrow{\mathbb{P}}_a$ and $\overrightarrow{\mathbb{P}}_b$, we get

$$\ln\left(\frac{d\overrightarrow{\mathbb{P}}_a}{d\overrightarrow{\mathbb{P}}_b}\right)(\boldsymbol{Z}) = \ln\left(\frac{d\mu}{d\nu}\right)(\boldsymbol{Z}_0) + \int_0^T \sigma_t^{-2}(a_t - b_t)(\boldsymbol{Z}_t)d\overrightarrow{\boldsymbol{Z}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)(\boldsymbol{Z}_t)dt, \tag{21}$$

and in particular, when evaluated on $\boldsymbol{Y}$:

$$\ln\left(\frac{d\overrightarrow{\mathbb{P}}_a}{d\overrightarrow{\mathbb{P}}_b}\right)(\boldsymbol{Y}) = \ln\left(\frac{d\mu}{d\nu}\right)(\boldsymbol{Y}_0) + \int_0^T \sigma_t^{-1}(a_t - b_t)(\boldsymbol{Y}_t)d\overrightarrow{\boldsymbol{W}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}\|b_t - a_t\|^2(\boldsymbol{Y}_t)dt. \tag{22}$$

Let $\mathbb{P}_{\text{data}}$ be the path probability of

$$\mathrm{d}\boldsymbol{X}_t = \left(f_t(\boldsymbol{X}_t) - \sigma_t^2\nabla_{\boldsymbol{X}_t}\ln p_t(\boldsymbol{X}_t)\right)\,\mathrm{d}t + \sigma_t\overleftarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{X}_T \sim \mathcal{P}_T, \tag{23}$$

and $\mathbb{P}_h$ the path probability of

$$\mathrm{d}\boldsymbol{H}_t = \left(f_t(\boldsymbol{H}_t) - \sigma_t^2(\nabla_{\boldsymbol{H}_t}\ln p_t(\boldsymbol{H}_t) + h_t(\boldsymbol{H}_t))\right)\,\mathrm{d}t + \sigma_t\overleftarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{H}_T \sim \mathcal{P}_T. \tag{24}$$

For the approximated importance weights we require the RND of $\mathbb{P}_h$ with respect to $\mathbb{P}_{\text{data}}$ evaluated at trajectories from $\mathbb{P}_h$. Let the drift of $\mathbb{P}_{\text{data}}$ be given as $a_t(\boldsymbol{x}) = f_t(\boldsymbol{x}) - \sigma_t^2\nabla_{\boldsymbol{x}}\ln p_t(\boldsymbol{x})$ and the drift of $\mathbb{P}_h$ as $b_t(\boldsymbol{x}) = f_t(\boldsymbol{x}) - \sigma_t^2\nabla_{\boldsymbol{x}}\ln p_t(\boldsymbol{x}) - \sigma_t^2 h_t(\boldsymbol{x})$. In particular, we have $a_t - b_t = \sigma_t^2 h_t$. For readability, we do not omit the dependence on $\boldsymbol{x}$ for the drift in the following. Using Proposition 7 we get,

$$\ln\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}\right)(\boldsymbol{H}) = \int_0^T \sigma_t^{-2}(a_t - b_t)\mathrm{d}\overleftarrow{\boldsymbol{H}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)\mathrm{d}t \tag{25}$$

$$= \int_0^T \sigma_t^{-2}(a_t - b_t)b_t\mathrm{d}t + \int_0^T \sigma_t^{-2}(a_t - b_t)\sigma_t d\overleftarrow{\boldsymbol{W}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)\mathrm{d}t \tag{26}$$

$$= \frac{1}{2}\int_0^T \sigma_t^{-2}[2a_t b_t - 2b_t^2 + b_t^2 - a_t^2]\mathrm{d}t + \int_0^T \sigma_t h_t\mathrm{d}\overleftarrow{\boldsymbol{W}}_t \tag{27}$$

$$= -\frac{1}{2}\int_0^T \sigma_t^{-2}(b_t - a_t)^2\mathrm{d}t + \int_0^T \sigma_t h_t\mathrm{d}\overleftarrow{\boldsymbol{W}}_t \tag{28}$$

$$= -\frac{1}{2}\int_0^T \sigma_t^2\|h_t\|_2^2\mathrm{d}t + \int_0^T \sigma_t h_t\mathrm{d}\overleftarrow{\boldsymbol{W}}_t, \tag{29}$$

evaluated on a trajectory $\boldsymbol{H}$ from $\mathbb{P}_h$.

A.1 Discrete Version for DDPM

We can also express the RND in the discrete setting. For this derivation, we make use of the DDPM schedule (Ho et al., 2020). The generalisation to different schedules is straightforward. The path probability of the pre-trained model is given as

$$p_\theta^{\text{data}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T) = p_T(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\mu_\theta(\boldsymbol{x}_t, t); \tilde{\beta}_t^2 I) \tag{30}$$

where the mean is parametrised as $\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t))$. Similar, we have path probabilities for the fine-tuned model as

$$p_\varphi^{\text{h}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T) = p_T(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\varphi^{\text{h}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad p_\varphi^{h}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\mu_h(\boldsymbol{x}_t, t); \tilde{\beta}_t^2 I), \tag{31}$$

with mean $\mu_h(\boldsymbol{x}_t, t) = \mu_\theta(\boldsymbol{x}_t, t) + \Delta_h(\boldsymbol{x}_t, t)$ is the original mean plus a delta given by the h-transform. We use $\tilde{\beta}_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t}$ for the standard deviation of the reverse kernel. Using this setting, we can write the RND as

$$\ln\left(\frac{p_\theta^{\text{data}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T)}{p_\varphi^{\text{h}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_T)}\right) = \sum_{t=1}^{T} \ln\left(\frac{p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p_\varphi^{h}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right), \tag{32}$$

where we assumed that the terminal distribution p_T is the same for both diffusion models. The log ratio of two Gaussian reduces to

$$\ln\left(\frac{\mathcal{N}(x; \mu_1, \Sigma_1)}{\mathcal{N}(x; \mu_2, \Sigma_2)}\right) = -\frac{1}{2}[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)] + \frac{1}{2}\ln\frac{|\Sigma_2|}{|\Sigma_1|}, \tag{33}$$

which gets us

$$\ln\left(\frac{p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p_\varphi^{h}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = -\frac{1}{2\tilde{\beta}_t^2}\left[\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2\right]. \tag{34}$$

To further simplify these terms, we need some information about the trajectory $\boldsymbol{x}_0, \dots, \boldsymbol{x}_T$. In particular, we assume that we have a *trajectory sampled from the fine-tuned model*, i.e.,

$$\boldsymbol{x}_{t-1} = \mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{35}$$

Given the parametrisation of the mean in the diffusion model

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\right), \tag{36}$$

$$\mu_h(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t) - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}h_\varphi(\boldsymbol{x}_t, t)\right), \tag{37}$$

we can reduce the terms on the RHS of Equation (34) to

$$\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2 \tag{38}$$

$$= \|\mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon - \mu_h(\boldsymbol{x}_t, t)\|_2^2 \tag{39}$$

$$= \|\Delta_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon\|_2^2 - \|\tilde{\beta}_t\epsilon\|_2^2. \tag{40}$$

Combining Equation (40) and Equation (34), we obtain

$$\ln\left(\frac{p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p_\varphi^{h}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = -\frac{1}{2\tilde{\beta}_t^2}\left[\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2\right] \tag{41}$$

$$= -\frac{1}{2\tilde{\beta}_t^2}\left[\|\Delta_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon\|_2^2 - \|\tilde{\beta}_t\epsilon\|_2^2\right] \tag{42}$$

$$= -\frac{1}{2\tilde{\beta}_t^2}\left[\|\Delta_h(\boldsymbol{x}_t, t)\|_2^2 + 2\Delta_h(\boldsymbol{x}_t, t)^\top\tilde{\beta}_t\epsilon + \|\tilde{\beta}_t\epsilon\|_2^2 - \|\tilde{\beta}_t\epsilon\|_2^2\right] \tag{43}$$

$$= -\frac{1}{2\tilde{\beta}_t^2}\|\Delta_h(\boldsymbol{x}_t, t)\|_2^2 - \frac{1}{\tilde{\beta}_t}\Delta_h(\boldsymbol{x}_t, t)^\top\epsilon. \tag{44}$$

For the full RND we obtain

$$\sum_{t=1}^{T} \ln\left(\frac{p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p_\varphi^h(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = \sum_{t=1}^{T}\left[-\frac{1}{2}\tilde{\beta}^{-2}\|\Delta_h(\boldsymbol{x}_t,t)\|_2^2 + \tilde{\beta}^{-1}\Delta_h(\boldsymbol{x}_t,t)^\top\epsilon\right], \tag{45}$$

which mimics a discretised version of the continuous RND in Equation (29).

B Proofs

B.1 Proof to Lemma 4

Proof. Denote by $\mathbb{P}_{\text{data}} = p_{\text{data}} \times \mathbb{P}_{\text{data}|0}$ and $\mathbb{P}_h = p_h \times \mathbb{P}_{h|0}$ the disintegrations of $\mathbb{P}_{\text{data}}$ and $\mathbb{P}_h$ with respect to time 0. Since $\mathbb{P}_{\text{data}}$ and $\mathbb{P}_h$ are both path measures for solutions of the SDE $\mathrm{d}\boldsymbol{X}_t = f_t(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma_t \mathrm{d}\overrightarrow{\mathbf{W}}_t$ we have that for any x_0 that $\mathbb{P}_{\text{data}|x_0}$ and $\mathbb{P}_{h|x_0}$ coincide with the path measure of this SDE with initial condition $\boldsymbol{X}_0 \sim \delta_{x_0}$. In particular, the disintegrations $\mathbb{P}_{\text{data}|x_0}$ and $\mathbb{P}_{h|x_0}$ coincide. Hence, we can conclude

$$\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]}) = \frac{p_{\text{data}}(x_0)}{p_h(x_0)}\frac{\mathrm{d}\mathbb{P}_{\text{data}|x_0}}{\mathrm{d}\mathbb{P}_{h|x_0}}(\boldsymbol{x}_{[0,T]}) = \frac{p_{\text{data}}(x_0)}{p_h(x_0)}. \tag{46}$$

□

B.2 Proof to Theorem 6

Proof. Part (i) is a Bayes theorem.

The proof of part (ii) follows similar ideas as (Hertrich & Gruhlke, 2025, Prop 8 (ii)). To this end, let $Z = \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}[\alpha(\boldsymbol{x}_{[0,T]})]$. Then, we can estimate by Jensens' inequality that

$$\begin{aligned}
-\ln(Z) &= -\ln\left(\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}\left[\min\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}, 1\right)\right]\right) \\
&\leq -\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}\left[\ln\left(\min\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}, 1\right)\right)\right] \\
&= -\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}\left[\min\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}\right), 0\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}\left[\max\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right), 0\right)\right]
\end{aligned}$$

On the other side, we have by definition that

$$\begin{aligned}
\mathrm{KL}(\tilde{\mathbb{P}}_h, \mathbb{P}_{\text{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\tilde{\mathbb{P}}_h}[r(\boldsymbol{x}_0)] &= \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\tilde{\mathbb{P}}_h}\left[\ln\left(\frac{\mathrm{d}\tilde{\mathbb{P}}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right] - \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\tilde{\mathbb{P}}_h}[r(\boldsymbol{x}_0)] \\
&= \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\tilde{\mathbb{P}}_h}\left[\ln\left(\frac{cZ}{\exp(r(\boldsymbol{x}_0))}\frac{\mathrm{d}\tilde{\mathbb{P}}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right] - \ln(cZ).
\end{aligned}$$

By taking the expectation over $\mathbb{P}_h$ instead of $\tilde{\mathbb{P}}_h$ this is equal to

$$\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{\mathrm{d}\tilde{\mathbb{P}}_h}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\ln\left(\frac{cZ}{\exp(r(\boldsymbol{x}_0))}\frac{\mathrm{d}\tilde{\mathbb{P}}_h}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right] - \ln(cZ).$$

Inserting the formula from part (i) and then the definition of α, this is equal to

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{\alpha(\boldsymbol{x}_{[0,T]})}{Z}\ln\left(\frac{c}{\exp(r(x))}\alpha(\boldsymbol{x}_{[0,T]})\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right]-\ln(cZ)\\
&=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{\alpha(\boldsymbol{x}_{[0,T]})}{Z}\ln\left(\frac{c}{\exp(r(x))}\min\left(1,\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}\right)\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right]-\ln(cZ)\\
&=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{\alpha(\boldsymbol{x}_{[0,T]})}{Z}\ln\left(\min\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))},1\right)\right)\right]-\ln(cZ)\\
&=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\frac{\alpha(\boldsymbol{x}_{[0,T]})}{Z}\min\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(cZ)\\
&=\frac{1}{Z}\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\min\left(\alpha(\boldsymbol{x}_{[0,T]})\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(cZ)\\
&=\frac{1}{Z}\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\min\left(\min\left(1,\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}\right)\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(cZ).
\end{aligned}$$

Since $1\leq\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}$ if and only if $\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right)\leq 0$ the minimum is attained either for both $\min$ in the above formula in the first argument or it is attained for both $\min$ in the second argument. Consequently the above formula is equal to

$$\begin{aligned}
&\frac{1}{Z}\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\min\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(cZ)\\
&\leq\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\min\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(Z)-\ln(c),
\end{aligned}$$

where the inequality comes from the fact that $Z\in(0,1]$ and that the expectation is non-positive (since the integrand is non-positive). Inserting the formula of $-\ln(Z)$ from the beginning of the proof, this is equal to

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\min\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]\\
&\quad+\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_h}\left[\max\left(\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right),0\right)\right]-\ln(c)\\
&=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right)\right]-\ln(c)\\
&=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}\left[\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0,T]})\right)\right]-\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}[r(\boldsymbol{x}_0)]\\
&=\mathrm{KL}(\mathbb{P}_h,\mathbb{P}_{\text{data}})-\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h}[r(\boldsymbol{x}_0)].
\end{aligned}$$

This concludes the proof of part (ii).

For part (iii), we observe that the loss in (15) is learning the marginal score of a forward SDE (in our case a VP-SDE intiliased at $\tilde{\mathbb{P}}_h^0$, by construction the associated path measure of this process is given by (See appendix A (Vargas et al., 2023a) for a similar sketch) [2]:

$$\mathbb{P}_{h^*}(\cdot)=\mathbb{P}_{\text{forward}}(\cdot|\boldsymbol{x}_0)\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0) \tag{47}$$

$$=\mathbb{P}_{\text{forward}}(\cdot|\boldsymbol{x}_0)\mathbb{P}_{\text{data}}(\boldsymbol{x}_0)\cdot\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}(\boldsymbol{x}_0) \tag{48}$$

$$=\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}\mathbb{P}_{\text{data}}(\cdot) \tag{49}$$

Thus, the path measure minimising the score-matching loss satisfies

$$\mathbb{P}_{h^*}=\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}\mathbb{P}_{\text{data}} \tag{50}$$

[2]Note we slightly abuse notation here as these equalities are meant to be understood in an RND sense but we omit the full ratio notation for brevity.

Then it follows that

$$\mathrm{KL}(\mathbb{P}_{h^*}||\mathbb{P}_{\mathrm{data}}) = \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_{h^*}}\left[\ln\left(\frac{\mathrm{d}\mathbb{P}_{\mathrm{data}}}{\mathrm{d}\mathbb{P}_{\mathrm{data}}}(\boldsymbol{x}_{[0:T]})\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\mathrm{data}}(\boldsymbol{x}_0)}\right)\right] \tag{51}$$

$$= \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\frac{\tilde{\mathbb{P}}_h^0}{p_{\mathrm{data}}}\mathbb{P}_{\mathrm{data}}}\left[\ln\left(\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\mathrm{data}}(\boldsymbol{x}_0)}\right)\right] \tag{52}$$

$$= \mathbb{E}_{\boldsymbol{x}_0\sim\tilde{\mathbb{P}}_h^0}\left[\ln\left(\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\mathrm{data}}(\boldsymbol{x}_0)}\right)\right] = \mathrm{KL}(\tilde{\mathbb{P}}_h^0, p_{\mathrm{data}}) \tag{53}$$

Via the disintegration theorem (Léonard, 2014, Theorem 1.6. and Theorem 2.4), the KL divergence can be decomposed as

$$\mathrm{KL}(\tilde{\mathbb{P}}_h, \mathbb{P}_{\mathrm{data}}) = \mathbb{E}[\mathrm{KL}(\tilde{\mathbb{P}}_h(\cdot|\boldsymbol{x}_0), \mathbb{P}_{\mathrm{data}}(\cdot|\boldsymbol{x}_0))] + \mathrm{KL}(\tilde{\mathbb{P}}_h^0, p_{\mathrm{data}}) \tag{54}$$

Where $\mathbb{E}[\mathrm{KL}(\tilde{\mathbb{P}}_h(\cdot|\boldsymbol{x}_0), \mathbb{P}_{\mathrm{data}}(\cdot|\boldsymbol{x}_0))] \geq 0$ and thus

$$\mathrm{KL}(\mathbb{P}_{h^*}, \mathbb{P}_{\mathrm{data}}) = \mathrm{KL}(\tilde{\mathbb{P}}_h^0, p_{\mathrm{data}}) \leq \mathrm{KL}(\tilde{\mathbb{P}}_h, \mathbb{P}_{\mathrm{data}}).$$

Since the marginals of $\mathbb{P}_{h^*}$ and $\tilde{\mathbb{P}}_h$ at time 0 coincide, we obtain

$$\mathcal{F}(\mathbb{P}_{h^*}) = \mathrm{KL}(\mathbb{P}_{h^*}, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_{h^*}}[r(\boldsymbol{x}_0)] \tag{55}$$

$$= \mathrm{KL}(\mathbb{P}_{h^*}, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\tilde{\mathbb{P}}_h}[r(\boldsymbol{x}_0)] \tag{56}$$

$$\leq \mathrm{KL}(\tilde{\mathbb{P}}_h, \mathbb{P}_{\mathrm{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\tilde{\mathbb{P}}_h}[r(\boldsymbol{x}_0)] = \mathcal{F}(\tilde{\mathbb{P}}_h), \tag{57}$$

which shows the first inequality from the claim. The second inequality was proven in part (ii). □

C Replay Buffers as Moving Average

As outlined in Section 3.3 we are using replay buffers for the numerical implementation of the importance fine-tuning algorithm. Once the buffer is

Replacing Random Samples from the Buffer In the case, that we are replace random samples from the buffer, it can be interpreted as moving average method and argue that $\mathcal{F}(\mathbb{P}_{h^k})$ still decreases when using them.

To this end, let h^k be the estimation of the h-transform in step k and recall that $\mathbb{P}_{h^k}$ is the path distribution of the backward SDE (5) with guidance h^k. Since h^k was trained with the fine-tuning loss (15), the samples in the replay buffer are distributed as $\mathbb{P}_{h^k}^0$. Now let $\tilde{\mathbb{P}}_{h^k}$ be the accepted samples in step $k+1$. Then, the trajectories corresponding to the samples in the replay buffer at time $k+1$ are distributed as $(1-\gamma)\mathbb{P}_{h^k}^0 + \gamma\tilde{\mathbb{P}}_{h^k}^0$, where γ is the number of accepted samples at time $k+1$ divided by the buffer size. At the same time, we know by the convexity of $\mathcal{F}$ in the linear space of measures (up to a constant it coincides with the KL divergence) that

$$\mathcal{F}((1-\gamma)\mathbb{P}_{h^k} + \gamma\tilde{\mathbb{P}}_{h^k}) \leq (1-\gamma)\mathcal{F}(\mathbb{P}_{h^k}) + \gamma\mathcal{F}(\tilde{\mathbb{P}}_{h^k}) = \mathcal{F}(\mathbb{P}_{h^k}) - \gamma(\mathcal{F}(\mathbb{P}_{h^k}) - \mathcal{F}(\tilde{\mathbb{P}}_{h^k})) \leq \mathcal{F}(\mathbb{P}_{h^k}),$$

where the last inequality follows from Theorem 6 (ii). In particular, following the same arguments as Theorem 6 (iii), we obtain that the minimiser h^{k+1} of the fine-tuning loss (15) using a dataset with the distribution $(1-\gamma)\mathbb{P}_{h^k}^0 + \gamma\tilde{\mathbb{P}}_{h^k}^0$ fulfils $\mathcal{F}(\mathbb{P}_{h^{k+1}}) \leq \mathbb{F}((1-\gamma)\mathbb{P}_{h^k} + \gamma\tilde{\mathbb{P}}_{h^k}) \leq \mathcal{F}(\mathbb{P}_{h^k})$.

Replacing the Oldest Samples from the Buffer In practice, we typically substitute the oldest samples from the buffer rather than choosing them at random. This is based on the intuition that more recent samples are produced by an improved model and thus are more in line with the target distribution. Although we cannot derive a similar justification as we do for random replacement, we have observed numerically that both approaches yield comparable final results while replacing the oldest samples results in faster convergence.

Table 5: Quantitative results for the GMM-2d example in Section 4.

	Classifier Guidance	**Online FT**	**Reward-only FT**	**Importance FT**
Energy distance (↓)	0.626	0.307	0.656	0.353
$\mathbb{E}[r(x_0)]$	1.954	1.385	3.187	2.474

D Experimental Details and Additional Results

D.1 2D Toy Example

We train the initial diffusion model on a Gaussian mixture model with modes 25 equally weighted modes where the set of means is defined by $\{-2.5, -1.25, 0, 1.25, 2.5\}^2$ and the covariance matrix is given by $0.1I$. The reward function r is defined as the negative log likelihood function of a GMM with four modes with means $(-2.5, 1.25)$, $(-1.25, 2.5)$, $(1.25, 0)$ and $(2.5, -1.25)$, covariance matrix $0.1I$ and mode weights $\frac{1}{8}$, $\frac{1}{8}$, $\frac{5}{8}$ and $\frac{1}{8}$. For classifier guidance, we choose the parameter γ by maximising the expected reward, which is $\gamma = 0.3$. For the importance and reward-only fine-tuning we use a batch size of 4096, a buffer size of 6000 and a KL regularisation with $\alpha_{\text{KL}} = 0.06$. We initialise the buffer with samples from the initial score-model. Then we perform 20 fine-tuning steps with 500 gradient updates per iteration. We choose c such that 40% of the samples will be rejected. The total training time is approximately 1min on a single NVIDIA GeForce RTX 4090.

For our importance fine-tuning we use the acceptance probability from (14), whereas for the reward-only fine-tuning we use the acceptance probability

$$\alpha(\boldsymbol{x}_{[0:T]}) = \min\left(1, \frac{\exp(r(\boldsymbol{x}_0))}{c}\right), \tag{58}$$

which does not take into account the likelihood under the pre-trained model.

We provide an additional numerical evaluations in Table 5 of the distributions shown in Figure 1. For this, we compute the energy distance $D(p_{\text{post}}, p_h)$ defined by

$$D(p_{\text{post}}, p_h)^2 = 2E[||X - Y||] - E[||X - X'||] - E[||Y - Y'||],$$

where $X, X' \sim p_{\text{post}}$ i.i.d. and $Y, Y' \sim p_h$ i.i.d. This metric is well-suited here as it can be computed directly from samples, without requiring density estimates. In addition, we report the expected reward under each method. While reward-only FT achieves the highest expected reward (it concentrates mass on the highest mode) it performs poorly in terms of distributional alignment, as reflected in the higher energy distance. In contrast, importance FT achieves a better balance, yielding lower energy distance while still maintaining high expected reward. This highlights that using the full importance weights during resampling is important for accurate fine-tuning.

D.2 Posterior Sampling for Inverse Problems

We train a unconditional score-based diffusion model on the `Flowers` dataset with the VP-SDE with $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 20.0$. We parametrise the score model using a small Attention UNet (Dhariwal & Nichol, 2021) with approx. 10M parameters. We parametrise the h-transform as in Equation (16) with approx. 0.8M parameters. The scaling network $\text{NN}_2(t)$ was initialised as a linear function going from 0 to 0.001. In this already at the start of training, we obtain trajectories which are similar to the ground truth leading to more stable optimisation. We used 50 importance fine-tuning iterations with 20 gradient updates per iteration. For every iteration we sampled 64 new trajectories and used a resampling rate of 50%. We used a maximal buffer size of 256 images and no KL-regularisation. The total training time is approximately 8min on a single NVIDIA GeForce RTX 4090.

At the start of fine-tuning the end points $\boldsymbol{x}_0$ of most trajectories are not consistent with the measurements $\boldsymbol{y}^\delta$, leading to a low reward $r(\boldsymbol{x}_0; \boldsymbol{y}^\delta)$. This can sometimes lead to instabilities at the start of training, if no valid high reward trajectory was sampled. We found that it was helpful to amortise

over measurements, such that we consider the amortised loss function

$$\mathcal{L}_{FT}(g) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_0\sim\tilde{\mathbb{P}}^0_{hk},\boldsymbol{y}\sim p^{\text{lhkd}}(\boldsymbol{y}|\boldsymbol{x}_0)\\ t\sim\mathrm{U}(0,T),\boldsymbol{x}_t\sim\overrightarrow{p}_{t|0}(\cdot|\boldsymbol{x}_0)}} \left[\left\|(g_t(\boldsymbol{x}_t,\boldsymbol{y})+s_t(\boldsymbol{x}_t))-\nabla_{\boldsymbol{x}_t}\ln\overrightarrow{p}_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\|^2\right], \tag{59}$$

where the simulated measurements $\boldsymbol{y} \sim p^{\text{lhkd}}(\boldsymbol{y}|\boldsymbol{x}_0)$ are added as an additional input to the h-transform.

We provide additional results for the first five test images in Figure 5, following the same experimental protocol. For each image, we draw 64 samples and compute both the per-pixel mean and the corresponding standard deviation. We also include the first sample from each batch for direct visual comparison. The standard deviation consistently rises along the flower boundary, where uncertainty is expected, and stays comparatively low in homogeneous background regions. Notably, the uncertainty remains stable across all images rather than collapsing.

In Figure 6, we compare samples generated by our importance FT method against DPS Chung et al. (2023), Top-K sampling, and an optimal-control approach using VarGrad Richter et al. (2020). For DPS, the guidance scale was tuned to maximise the reward. We also report the mean reward, defined as $r(\boldsymbol{x}) = -\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}^\delta\|_2^2$, estimated using 64 samples. The samples from importance FT and VarGrad are visually similar, but importance FT consistently achieves a larger reward. The samples from DPS are too smooth and lacking texture. Top-K sampling (with $K = 64$) fails on this super-resolution task; the reward landscape is too sharply peaked, making it unlikely to draw an image consistent with the observation $\boldsymbol{y}^\delta$ by chance.

D.3 Class conditional sampling

We pre-train a score-based diffusion model on the MNIST dataset with the VP-SDE with $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 20.0$. We parametrise the score model using a small Attention UNet (Dhariwal & Nichol, 2021) with approx. 1.1M parameters. We parametrise the h-transform as in Equation (16) with approx. 0.8M parameters. For all experiments, we use a batch size of 256, a buffer size of 2048 and a reward scaling $\lambda = 4.0$. We perform 50 importance fine-tuning iterations with 50 gradient updates per iterations. We use the KL regulariser with $\alpha_{\text{KL}} = 0.001$. Lastly, we adaptively choose c such that 10% of the samples will be accepted for the first 10 steps. After the initial 10 steps, we accept 30% of the samples. We start with 10% as MNIST has 10 classes with roughly similar class probabilities. The total training time is about 10min on a single NVIDIA GeForce RTX 4090.

For the comparison methods in Table 1, we use classifier guidance and online fine-tuning. For classifier guidance, we used a scaling $\gamma = 4.5$ maximising sample quality. For online fine-tuning we directly minimise (9) using VarGrad (Richter et al., 2020). VarGrad enables us to detach the trajectory from the backpropagation, thus saving memory cost and enabling us to train with a larger batch size. In particular, we use the formulation in Denker et al. (2024) as

$$D_{\text{logvar}}(\mathbb{P}_h, \mathbb{P}_{\text{data}}; \mathbb{W}) = \mathrm{Var}_{\boldsymbol{H}^{g_t}_{0:T}\sim\mathbb{W}}\left[\ln\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{H}^{g_t}_{0:T})\right], \tag{60}$$

evaluated at a reference process $\mathbb{W} = \mathrm{Law}(\boldsymbol{H}^{g_t}_{0:T})$, given by

$$\begin{aligned}\boldsymbol{H}_T &\sim Q^{f_t}_T[p_{\text{tilted}}]\\ \mathrm{d}\boldsymbol{H}_t &= \left(f_t(\boldsymbol{H}_t) - \sigma_t^2(s_t(\boldsymbol{H}_t) + g_t(\boldsymbol{H}_t))\right)\mathrm{d}t + \sigma_t\,\overline{\mathrm{d}\mathbf{W}}_t.\end{aligned} \tag{61}$$

With $g_t = \mathrm{stop\_grad}(h_t)$ as a detached copy of the current estimation h_t, we obtain

$$\begin{aligned}\ln\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{H}^{g_t}_{0:T}) = -\frac{1}{2}\int_0^T \sigma_t^2\|h_t(\boldsymbol{H}^{g_t}_t)\|^2\mathrm{d}t + \int_0^T \sigma_t^2(g_t^\top h_t)(\boldsymbol{H}^{g_t}_t)\mathrm{d}t - r(\boldsymbol{x}^{g_t}_0)\\ + \int_0^T \sigma_t h_t^\top(\boldsymbol{H}^{g_t}_t)\overline{\mathrm{d}\mathbf{W}}_t,\end{aligned} \tag{62}$$

using the RND for time reverse SDEs see Equation 64 in (Vargas et al., 2024). We use the same reward-informed network architecture to parametrise h_t.

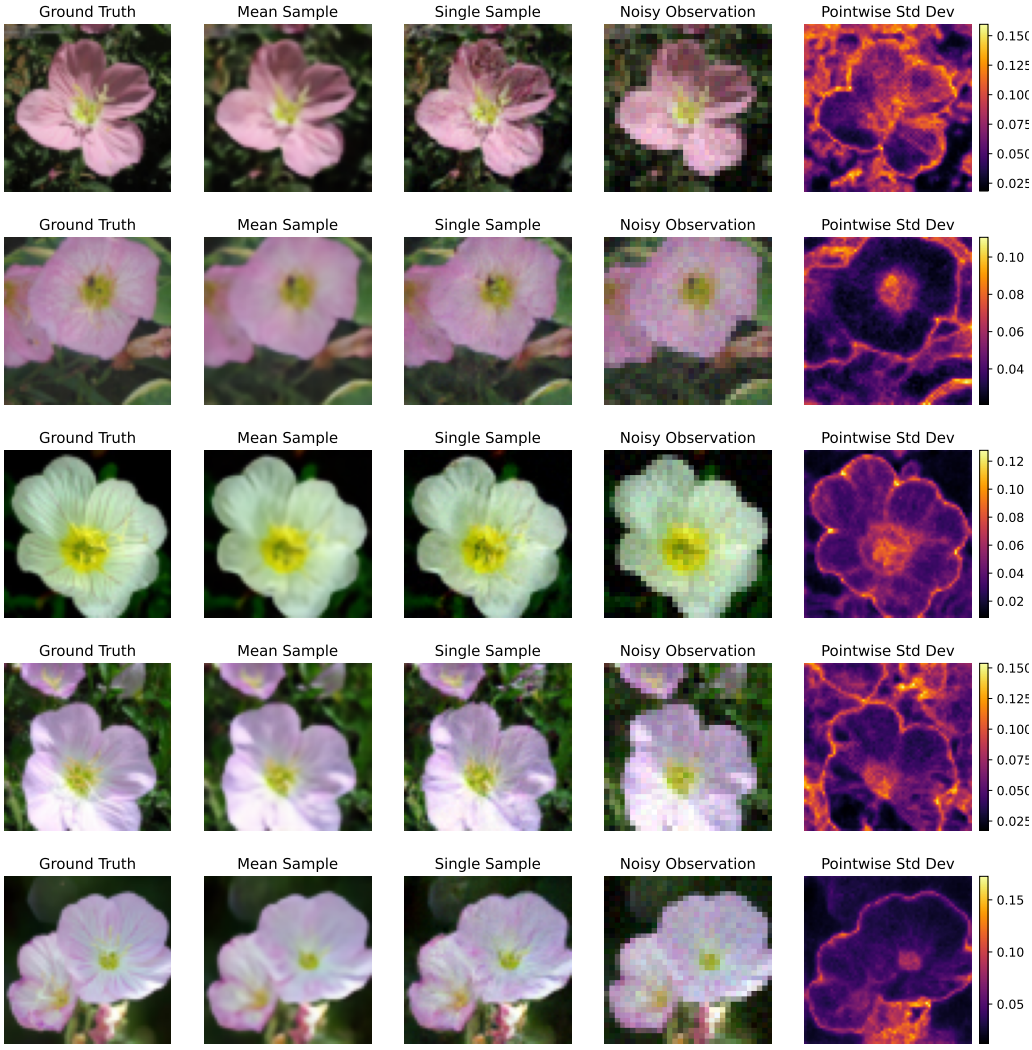


Figure 5: Additional results for different images from the test set of the `Flowers` dataset.

Importance FT $\mathbb{E}[r(x)] = -15.28$ | DPS $\mathbb{E}[r(x)] = -78.95$ | VarGrad $\mathbb{E}[r(x)] = -18.28$ | Top-K $\mathbb{E}[r(x)] = -1619.74$

Figure 6: Comparison of samples for the super-resolution task on the `Flowers` dataset.

In Figure 7 we show the posterior for all 10 classes. For all experiments, we chose the same hyper-parameters as well as the same initial seed for the samples presented. We observe that these settings work for most classes. However, for class "Four" we see a single digit "Six" in the image, giving evidence that this particular model has not fully converged.

Posterior: Zero | Posterior: One | Posterior: Two | Posterior: Three | Posterior: Four

Posterior: Five | Posterior: Six | Posterior: Seven | Posterior: Eight | Posterior: Nine

Figure 7: Class conditional sampling for MNIST for all classes. We used the same initial seed. The model for "Posterior: Four" has apparently not fully converged, as we still see a "6" in the image.

$\lambda^{-1} = 0.01$ | $\lambda^{-1} = 0.1$ | $\lambda^{-1} = 1.0$ | $\lambda^{-1} = 2.0$ | $\lambda^{-1} = 100.0$

Figure 8: Samples for changing the temperature λ for MNIST class conditional sampling for the digit "Two".

Ablation: Changing the temperature We perform an ablation, where we change the temperature λ in (1). In particular, a small λ^{-1} will lead to a tilted distribution which is closer to the base distribution, whereas a large λ^{-1} will put more weight onto the reward. Quantitative results are presented in Table 6 for the digit "Two", see also Figure 8 for a visual comparison. We report the FID, the expected reward and the accuracy on 1024 samples. Note, that we do not expect an accuracy of 100% as the original classifier only reaches an accuracy of 98.6%. For $\lambda^{-1} = 0.01$, we obtain mostly samples of the base model and the classifier has no effect, leading to a low accuracy of 7.91%. The best results with respect to FID are obtained for $\lambda^{-1} = 2.0$. For a smaller temperature, i.e., $\lambda^{-1} = 100$ the diversity of samples decreases, leading to a higher accuracy, but also a worse FID.

D.4 Reward Fine-tuning

Diffusion models can be used for conditional generation via classifier free guidance (Ho & Salimans, 2022). In classifier free guidance both an unconditional $\epsilon_t^\theta(\boldsymbol{x}_t)$ and a conditional $\epsilon_t^\theta(\boldsymbol{x}_t, c)$ noise-prediction model are learned by randomly masking out the text prompt c. During sampling the linear combination $\bar{\epsilon}_t^\theta(\boldsymbol{x}_t, t) = (1 + w)\epsilon_t^\theta(\boldsymbol{x}_t, c) - w\epsilon_t^\theta(\boldsymbol{x}_t)$, with a guidance scale w, is used. Despite the progress in training text-to-image diffusion models, the samples are not always aligned with human preferences. For the alignment we make use of `ImageReward-v1.0`, a human preference reward model (Xu et al., 2024). These reward models $r(\boldsymbol{x}; c)$ are trained to produce a reward from a given text prompt c and an image $\boldsymbol{x}$, corresponding to human preferences. Here, we learn the tilted distribution for a given text prompt and directly fine-tune the classifier free model $\bar{\epsilon}_t^\theta(\boldsymbol{x}_t, t)$ using the LoRA parametrisation (Hu et al., 2022) instead of the reward-informed parametrisation. We use a LoRA for all attention layers with a rank of 10. We used a buffer of 64 images and sample 128 new images at every iteration, the parameter c was chosen such at 10% of the images were accepted. We

Table 6: Results for varying the temperature λ for class conditional sampling on MNIST for class 2.

λ^{-1}	0.01	0.1	1.0	2.0	4.0	10.0	100.
FID (↓)	1457.99	934.26	123.51	84.96	122.25	151.28	149.72
$\mathbb{E}[r(x)]$ (↑)	−13.54	−8.49	−0.094	−0.078	−0.102	−0.043	−0.062
Accuracy (↑)	7.91	17.28	98.53	98.34	98.53	99.12	99.02

Table 7: Comparison of run-time and peak GPU memory for the different methods for reward fine-tuning.

	Base Model	DPOK	RTB	Adjoint Matching	Importance FT
Training Time (h)	N/A	28	17	4	7
Peak GPU Memory (GB)	9	34	20	36	16

used a total of 25 iteration, where we did 40 gradient descent steps with an effective batch size of 64 at each iteration (batch of 4 and 16 gradient accumulation steps). We use a weight of 2.5 for the KL-regulariser in Eqn. (17). Fine-tuning took about 7 hours on a single NVIDIA Geforce RTX 4090. We used the AdamW optimiser (Loshchilov & Hutter, 2019) with a learning rate of 1×10^{-4} and a cosine decay to 1×10^{-5}. The weight decay is chosen as 0.001. For the final sampling evaluation we used the DDIM scheduler with 50 time steps. We used a guidance scale of 5.0 for Stable Diffusion. We show additional results for changing the temperature λ in Figure 12.

We compare against Top-K sampling (also often referred to as Best-of-K sampling), DPOK (Fan et al., 2024), Adjoint Matching (Domingo-Enrich et al., 2025) and RTB Venkatraman et al. (2024). We give the experimental details in the following paragraphs.

Additional images are shown in Figure 10 and Figure 11. We also observe the reward and diversity during iterations, see Figure 9.

Table 7 reports approximate training times and peak GPU memory usage for the different methods. Note that DPOK and Adjoint Matching were executed on a machine with an A100 GPU. These numbers should be interpreted cautiously, as training time and GPU memory can be traded off, for example, via gradient accumulation to increase effective batch size or gradient checkpointing. The main purpose of this comparison is to illustrate that our approach can be run on smaller GPUs, whereas the default configurations of DPOK or Adjoint Matching generally require larger GPUs.

Top-K Sampling is a widely known approach to increase the performance of text-to-image diffusion models without additional training. Here, we sample a batch of K images and only return the image with the highest reward. In our implementation we use $K = 6$. Top-K sampling is easy to implement and requires no additional training loss, but often decrease diversity and increased the sampling time by a factor of K.

FK-Steering was proposed by Singhal et al. (2025) as a general inference-time approach to control diffusion models. We use the implementation by the authors using thee hyperparameters from the original publication[3]. FK-Steering is a particle-based framework. The original implementation used $k = 4$ particles for Stable Diffusion. However, we used $k = 6$ to be consistent with the Top-K sampling baseline. We observe slightly higher scores for $k = 6$, which is consistent with the observations by Singhal et al. (2025).

DPOK was proposed by (Fan et al., 2024) to fine-tune diffusion models using a differentiable reward. We use the implementation by the authors using the provided hyperparameters[4]. DPOK used LoRA for the fine-tuning parametrisation, we use the default parameters as suggested by the authors. Fine-tuning takes 28h on a single A100.

Adjoint Matching reformulates fine-tuning as a stochastic optimal control problem and casting it as a regression problem. We use the implementation by the authors with the provided hyperparame-

[3] https://github.com/zacharyhorvitz/Fk-Diffusion-Steering

[4] https://github.com/google-research/google-research/tree/master/dpok

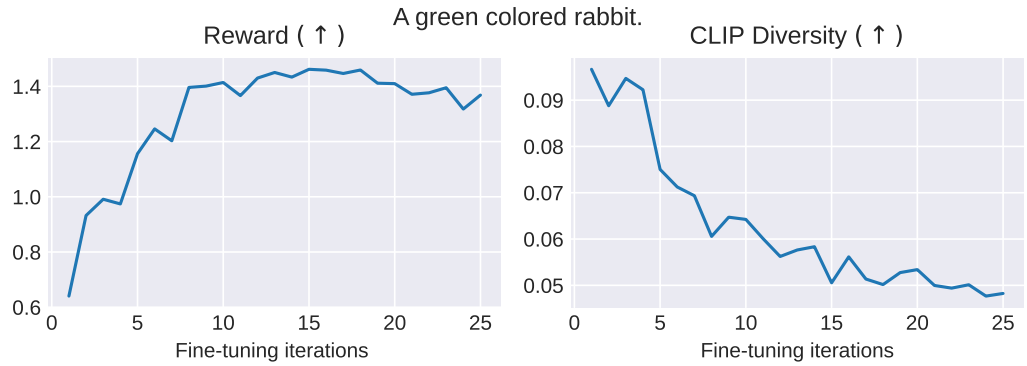


Figure 9: Reward and diversity for text-to-image fine-tuning on "A green colored rabbit." during training. As known for LoRA fine-tuning the diversity decreases over iterations.

Base Model | DPOK | RTB | Adjoint Matching | Importance FT

Figure 10: Samples for the base model, DPOK, Adjoint Matching and our importance FT for the prompt "Two roses in a vase.". Images were generated using the same seed.

ters[5]. Adjoint Matching fine-tunes the full model and does not use LoRA. This requires large GPUs and the experiments cannot be performed on a NVIDIA Geforce RTX 4090. We use the adjoint matching variant which uses CFG to sample trajectories, but not in the adjoint computation. This setting is not covered by theory, but leads to the best empirical results.

Relative Trajectory Balance (RTB) proposed by Venkatraman et al. (2024) arises from a GFlowNet perspective on diffusion models. We use the implementation by the authors with the provided hyperparameters[6].

[5] https://github.com/microsoft/soc-fine-tuning-sd
[6] https://github.com/GFNOrg/diffusion-finetuning

Base Model | DPOK | RTB | Adjoint Matching | Importance FT

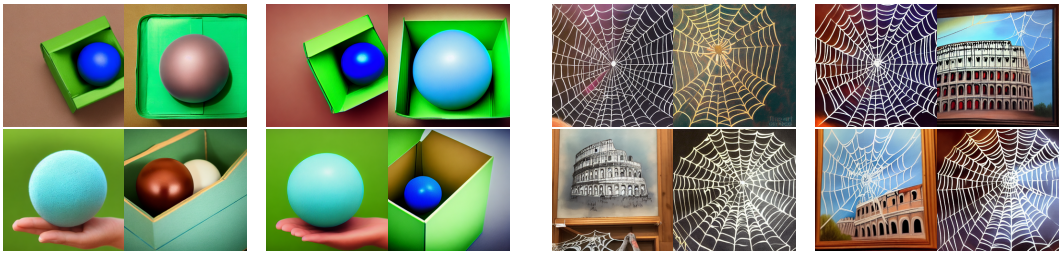
Figure 11: Samples for the base model, DPOK, Adjoint Matching and our importance FT for the prompt "Two dogs in the park.". Images were generated using the same seed.

$\lambda^{-1} = 0.1$ | $\lambda^{-1} = 1.0$ | $\lambda^{-1} = 10.$ | $\lambda^{-1} = 100.$

Figure 12: Samples for importance FT for the prompt "A green colored rabbit." with different temperatures λ^{-1}. Images were generated using the same seed.

$w = 2.0$ | $w = 4.0$ | $w = 5.0$ | $w = 7.5$

Figure 13: Samples for importance FT for the prompt "A green colored rabbit." with different guidance scales w for classifier free guidance. Images were generated using the same seed.



Base Model (mean reward -0.18) | Importance FT (mean reward 1.02) | Base Model (mean reward -1.15) | Importance FT (mean reward 0.18)

"A blue ball in a green box" | "A spider web and a painting of the Colosseum"

Figure 14: We run the importance fine-tuning on two additional prompts.