ITERATIVE IMPORTANCE FINE-TUNING OF DIFFUSION MODELS

Alexander Denker
University College London
a.denker@ucl.ac.uk

Shreyas Padhy
University of Cambridge
sp2058@cam.ac.uk

Francisco A. Vargas Palomo
University of Cambridge
fav25@cam.ac.uk

Johannes Hertrich
Université Paris Dauphine-PSL
johannes.hertrich@dauphine.psl.eu

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 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., 2024), where the distribution defined by the pre-trained diffusion models $p_{\text{data}}(\boldsymbol{x})$ 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). The tilted distribution can also be expressed as the minimiser of an optimisation problem, i.e.,

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

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.

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. In this work, we propose a self-supervised framework for estimating the h-transform without access to samples from the tilted distribution. In particular, we iteratively sample from the diffusion model, resample on path-based importance weights and fine-tune the model using the resampled data.

Controlled Generation from Diffusion Models Controlled generation for diffusion models can be achieved via inference-time or post-training methods (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., 2024).

Iterative retraining can cause model degradation 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 Recently, a number of self-supervised frameworks have been proposed for sampling from unnormalised densities, for example FAB (Midgley et al., 2023) or iDEM (Akhound-Sadegh et al., 2024). In iDEM, the authors propose an iterative framework, where they train a diffusion model with a stochastic regression loss based on data sampled from the current estimation of the diffusion model.

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. Lastly, we present experiments on class conditional sampling for MNIST, and reward-based fine-tuning of Stable Diffusion (Rombach et al., 2022) in Section 4.

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}

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

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 $\overleftarrow{p}_{t_1|t_2}(\boldsymbol{x}_{t_1}|\boldsymbol{x}_{t_2})$ 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 guidiance 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 $\boldsymbol{H}_0 \sim p_{\text{tilted}}$, Denker et al. (2024); Vargas et al. (2023b) considered a generalisation of Doob's h-transform. In particular, they proved the following theorem.

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* $Q_T^{f_t}[p_{\text{tilted}}] := \int \vec{p}_{T|0}(\boldsymbol{x}|\boldsymbol{x}_0)p_{\text{tilted}}(\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_0$ *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* $\boldsymbol{H}_T \sim Q_T^{f_t}[p_{\text{tilted}}]$. *Then, it holds* $\boldsymbol{H}_0 \sim p_{\text{tilted}}$.

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

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

(iii) It holds that

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

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

Remark 2. *In Theorem 1 (i) the terminal distribution is given as* $Q_T^{f_t}[p_{\text{tilted}}]$ *instead of* $\mathbb{P}_T$ *in Equation* (4) *to remove the value function bias. In Domingo-Enrich et al. (2024), 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* $Q_T^{f_t}[p_{\text{tilted}}] \approx \mathbb{P}_T \approx \mathcal{N}(0, \mathbf{I}_n)$ *for all numerical experiments.*

Part (i) of Theorem 1 states conditions on h_t such that $\boldsymbol{H}_0 \sim p_{\text{tilted}}$ in (5) and the (ii), (iii) provide loss functions to learn such a h_t. For the commonly used VP-SDE (Song et al., 2021), the loss (7) in part (ii) reduces to

$$\mathcal{L}_{SM}(h_t) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_0 \sim p_{\text{tilted}} \\ t\sim \mathrm{U}(0,T), \boldsymbol{z} \sim \mathcal{N}(0,I)}} \left[\left\| (h_t(\gamma_t \boldsymbol{x}_0 + \nu_t \boldsymbol{z}) + s_t(\gamma_t \boldsymbol{x}_0 + \nu_t \boldsymbol{z})) - \boldsymbol{z}/\nu_t \right\|^2 \right], \tag{9}$$

where the parameters γ_t, ν_t are specified by the drift and diffusion coefficients. However, samples from the tilted distribution are often not available, making this approach limited in practice. Moreover, the loss function in (8) can be reformulated as a stochastic control (SC) loss, i.e.,

$$\mathcal{L}_{SC}(h_t) = \mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_h} \left[\frac{1}{2} \int_0^T \sigma_t^2 \|h_t(\boldsymbol{x}_t)\|^2 \mathrm{d}t - r(\boldsymbol{x}_0) \right]. \tag{10}$$

Differentiating $\mathcal{L}_{SC}$ requires differentiating through the sample generation of the SDE. Even though Denker et al. (2024) reduce the computational cost of this step by using the VarGrad (Richter et al., 2020) or trajectory balance loss (Malkin et al., 2022), the memory requirements for optimising $\mathcal{L}_{SC}$ still scale linearly with the number of steps in the SDE generation.

3 Self-Supervised Importance Fine-tuning

To circumvent the limitations of the loss functions from Denker et al. (2024), we propose in this section a self-supervised method for fine-tuning diffusion models. To this end, 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.

3.1 IMPORTANCE-BASED REJECTION

A basic tool for reweighting and fine-tuning generative models are importance weights. To this end, let $\boldsymbol{x}_0$ be generated from the backward SDE (5) for some guidance term h. We assign to it the weight $\frac{p_{\text{tilted}}(\boldsymbol{x}_0)}{p_h(\boldsymbol{x}_0)}$, where p_h is the density of the generated distribution. Then, samples with a large importance weights are under-represented and samples with small importance weights are over-represented in p_h for approximating p_{tilted}.

Unfortunately, we do not have access to the explicit value of p_h. Therefore, we rewrite the importance weight with $p_{\text{tilted}}(\boldsymbol{x}) \propto p_{\text{data}}(\boldsymbol{x}) \exp(r(\boldsymbol{x}))$ as

$$\frac{p_{\text{tilted}}(\boldsymbol{x})}{p_h(\boldsymbol{x})} = \frac{p_{\text{tilted}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})} \frac{p_{\text{data}}(\boldsymbol{x})}{p_h(\boldsymbol{x})} \propto \exp(r(\boldsymbol{x})) \frac{p_{\text{data}}(\boldsymbol{x})}{p_h(\boldsymbol{x})}. \tag{11}$$

Now we approximate the quotient $\frac{p_{\text{data}}(\boldsymbol{x})}{p^h(\boldsymbol{x})}$ by the ELBO from Denker et al. (2024), such that

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

where $\mathbb{P}_h$ and $\mathbb{P}_{\text{data}}$ are the path measures for the SDEs corresponding to the prior and adjusted diffusion model and $\boldsymbol{x}_{[0:T]} = (\boldsymbol{x}_t)_{t\in[0,T]}$ is the whole generated trajectory. The Radon-Nikodym derivative $\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})$ can now be computed based on the framework from Vargas et al. (2024). We summarise the result in the following lemma. The derivations are included in Appendix A, both in continuous and in discrete time.

Lemma 3. *The right side of (12) can be rewritten as*

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

These path-wise importance weights can be computed concurrently with sampling without any computational overhead. For numerical stability, we perform these computations in the log-space.

In practice, importance weights have the disadvantage that they often suffer from highly imbalanced weights. 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. In practice, we use an adaptive choice of c such that a certain rate of samples is accepted, see (Hertrich & Gruhlke, 2025, Rem 10). 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 $\tilde{\mathbb{P}}_{h^k}^0$. 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 $h^{k+1} \in \arg\min_g \mathcal{L}_{FT}(g)$, where

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

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
Require: Rate of accepted samples r

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
    $t \sim U(\{1, \dots, T\})$
    $\epsilon \sim \mathcal{N}(0, I)$
    $\boldsymbol{x}_t^{(1:m)} \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{x}^{(1:m)} + \sqrt{1-\bar{\alpha}_t}\epsilon$
    $\ell \leftarrow \left\| \left((\epsilon_t^\theta(\boldsymbol{x}_t^{(1:m)}) + h_t^\varphi(\boldsymbol{x}_t^{(1:m)}) \right) - \epsilon \right\|_2^2$
    $\varphi \leftarrow \text{Optim}(\varphi, \nabla_\varphi \ell)$

We summarise our self-supervised importance fine-tuning in Algorithm 1.

Recall that the generalised h-transform from Theorem 1 minimises the function $\mathcal{F}(\mathbb{P}_h) = \mathrm{KL}(\mathbb{P}_h, \mathbb{P}_{\text{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]} \sim \mathbb{P}_h}[r(\boldsymbol{x}_0)]$. 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.1.

Theorem 4. *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* $\mathcal{F}(\mathbb{P}) = \mathrm{KL}(\mathbb{P}, \mathbb{P}_{\text{data}}) - \mathbb{E}_{\boldsymbol{x}_{[0,T]} \sim \mathbb{P}}[r(\boldsymbol{x}_0)] \propto \mathrm{KL}(\mathbb{P}, \mathbb{P}_{\text{tilde}})$

(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. Motivated by Midgley et al. (2023); Sendera et al. (2024), we make use of an unprioritised replay buffer with a fixed length. We save the accepted samples from the current model and append them the buffer. Once the fixed length is reached, we discard the oldest samples. During training, we randomly sample batches from the buffer.

Network parametrisation The parametrisation of the h-transform has a crucial effect on performance and convergence speed. 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.

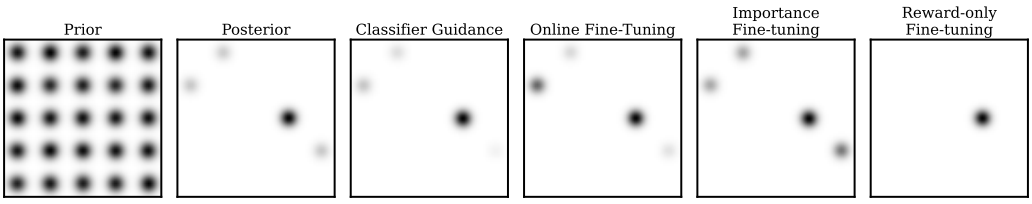

Figure 1: 2D Toy example for fine-tuning diffusion models. We show the prior the log-probability of the tilted distribution and samples from the conditional diffusion model using classifier guidance (Dhariwal & Nichol, 2021), online fine-tuning (Fan et al., 2024), our importance fine-tuning and a variation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

KL-Regularisation Similar to Fan et al. (2024), we found that using a KL regulariser was useful to 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 and finally preliminary results for reward fine-tuning of text-to-image diffusion models.

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 C.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). Instead of employing the gradient calculation of DPOK (Fan et al., 2024), we backpropagate gradients through the entire trajectory for this toy example. Further, we evaluate our importance fine-tuning and a variation where importance weights are calculated solely based on the reward, referred to as reward-only fine-tuning.

In Figure 1, we observe that classifier guidance is able to find all the modes of the posterior, 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 Class-conditional sampling

As another application, we consider class conditional sampling. 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 $r(x) = \ln p(c|\boldsymbol{x})$. We consider both the MNIST dataset and pre-train unconditional diffusion models using the U-Net architecture from Ho et al. (2020). We use a convolutional classifier with a 98.6% accuracy. We use 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. We show posterior prior samples in Figure 2, see Figure 4 for all classes. We compare both

Prior samples | Posterior: Six | Posterior: Even numbers

Figure 2: Class conditional sampling for MNIST. Left: Sampled from unconditional model. Middle: Samples for class 'six'. Right: Samples for even numbers.

Table 1: Expected reward and FID scores for both the single class and the even/odd task on MNIST. FID is computed for the features of the penultimate layer of the pre-trained classifier for 1024 samples. We compare against classifier guidance with scale $\gamma = 4.5$, online fine-tuning and our importance fine-tuning.

	Single class		**Even / Odd**	
	$\mathbb{E}[r(\boldsymbol{x})]$ ($\uparrow$)	FID ($\downarrow$)	$\mathbb{E}[r(\boldsymbol{x})]$ ($\uparrow$)	FID ($\downarrow$)
Classifier Guidance	-1.495	117.241	-2.572	193.984
Online Fine-tuning	-0.110	31.994	-0.065	198.871
Importance Fine-tuning	-0.152	30.814	-0.878	110.403

against classifier guidance and online fine-tuning. For online fine-tuning we optimise Equation (10) using VarGrad (Richter et al., 2020) to reduce the memory cost. In Table 1 we present the expectation of the reward and the FID (Heusel et al., 2017) based on the features of the penultimate layer of the pre-trained classifier. For single-class posteriors, online fine-tuning achieves a higher expected reward but results in a lower actual reward. A similar trend is observed for even/odd tasks. This indicates that online fine-tuning is more effective at maximising reward, although at the expense of reduced sample diversity, as reflected by the FID score.

4.3 Text-to-Image Reward Fine-tuning

Despite the progress in training text-to-image diffusion models, samples not always align with human preferences. In particular, the model can struggle with unnatural prompts such as "A green rabbit" (Venkatraman et al., 2024; Fan et al., 2024). To alleviate this problem, the diffusion model is fine-tuned to maximise some reward model, trained to imitate human preferences. 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. We refer to Appendix C.3 for experimental details. In Figure 3, we show examples for the prompt "A green rabbit". Here, we see that in contrast to the pre-trained model, we obtain samples which align better with the prompt. This can also be seen in the mean reward value of -0.18 for the base model and 0.67 for the fine-tuned model, evaluated over 60 images. However, for some samples the visual quality also degrades. Specifically, in the middle image of the bottom row, the output shows oversaturation in the green colour.

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 show initial evaluations for class conditional sampling 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)

Base Model Fine-tuned Model

Figure 3: Samples for the base model and the fine-tuned model for the prompt "A green colored rabbit.". Images were generated using the same seed.

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 iterative refinement method, the model is fine-tuned using "good", i.e., having a high importance weight, samples from the pre-trained model. 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 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 3. However, we can still compute importance weights using the RND in Proposition 5.

ACKNOWLEDGMENTS

Alexander Denker acknowledges support by the EPSRC programme grant EP/V026259/1. Johannes Hertrich acknowledges funding by the German Research Foundation (DFG) within the Walter Benjamin Programme with project number 530824055. Shreyas Padhy is funded by the University of Cambridge Harding Distinguished Postgraduate Scholars Programme.

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.

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.

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.

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.

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 TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

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.

Johannes Hertrich and Robert Gruhlke. Importance corrected neural JKO sampling. In *International Conference on Machine Learning*, 2025.

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.

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

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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.

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.

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.

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.

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, 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.

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

$$\mathrm{d}\boldsymbol{Y}_t = a_t(\boldsymbol{Y}_t)\,\mathrm{d}t + \sigma_t(\boldsymbol{Y}_t)\,\overrightarrow{\mathrm{d}\boldsymbol{W}}_t, \quad \boldsymbol{Y}_0 \sim \mu, \tag{18}$$

$$\mathrm{d}\boldsymbol{X}_t = b_t(\boldsymbol{X}_t)\,\mathrm{d}t + \sigma_t(\boldsymbol{X}_t)\,\overrightarrow{\mathrm{d}\boldsymbol{W}}_t, \quad \boldsymbol{X}_0 \sim \nu, \tag{19}$$

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

$$\ln\left(\frac{d\overrightarrow{\mathbb{P}}_a}{d\overrightarrow{\mathbb{P}}_b}\right)(\boldsymbol{Z}) = \ln\left(\frac{d\mu}{d\nu}\right)(\boldsymbol{Z}_0) + \int_0^T \sigma_t^{-2}(a_t - b_t)(\boldsymbol{Z}_t)d\overrightarrow{\boldsymbol{Z}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)(\boldsymbol{Z}_t)dt, \tag{20}$$

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

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

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

$$\mathrm{d}\boldsymbol{X}_t = \left(f_t(\boldsymbol{X}_t) - \sigma_t^2\nabla_{\boldsymbol{X}_t}\ln p_t(\boldsymbol{X}_t)\right)\,\mathrm{d}t + \sigma_t\overleftarrow{\mathrm{d}\mathbf{W}}_t, \quad \boldsymbol{X}_T \sim \mathcal{P}_T, \tag{22}$$

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

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

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 5 we get,

$$\ln\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_h}\right)(\boldsymbol{H}) = \int_0^T \sigma_t^{-2}(a_t - b_t)\mathrm{d}\overleftarrow{\boldsymbol{H}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)\mathrm{d}t \tag{24}$$

$$= \int_0^T \sigma_t^{-2}(a_t - b_t)b_t\mathrm{d}t + \int_0^T \sigma_t^{-2}(a_t - b_t)\sigma_t d\overleftarrow{\boldsymbol{W}}_t + \frac{1}{2}\int_0^T \sigma_t^{-2}(b_t^2 - a_t^2)\mathrm{d}t \tag{25}$$

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

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

$$= -\frac{1}{2}\int_0^T \sigma_t^2\|h_t\|_2^2\mathrm{d}t + \int_0^T \sigma_t h_t\mathrm{d}\overleftarrow{\boldsymbol{W}}_t, \tag{28}$$

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

$$p_\theta^{\text{data}}(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T) = p_T(\boldsymbol{x}_T)\prod_{t=1}^T p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad p_\theta^{\text{data}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\mu_\theta(\boldsymbol{x}_t, t); \tilde{\beta}_t^2 I) \tag{29}$$

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

$$p^{\mathrm{h}}_\varphi(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T) = p_T(\boldsymbol{x}_T) \prod_{t=1}^{T} p^{\mathrm{h}}_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad p^h_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\mu_h(\boldsymbol{x}_t, t); \tilde{\beta}_t^2 I), \tag{30}$$

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

$$\ln\left(\frac{p^{\text{data}}_\theta(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)}{p^{\mathrm{h}}_\varphi(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)}\right) = \sum_{t=1}^{T} \ln\left(\frac{p^{\text{data}}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p^h_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right), \tag{31}$$

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

$$\ln\left(\frac{\mathcal{N}(x; \mu_1, \Sigma_1)}{\mathcal{N}(x; \mu_2, \Sigma_2)}\right) = -\frac{1}{2}[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)] + \frac{1}{2}\ln\frac{|\Sigma_2|}{|\Sigma_1|}, \tag{32}$$

which gets us

$$\ln\left(\frac{p^{\text{data}}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p^h_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = -\frac{1}{2\tilde{\beta}_t^2}\left[\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2\right]. \tag{33}$$

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

$$\boldsymbol{x}_{t-1} = \mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{34}$$

Given the parametrisation of the mean in the diffusion model

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\right), \tag{35}$$

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

we can reduce the terms on the RHS of Equation (33) to

$$\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2 \tag{37}$$

$$= \|\mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\mu_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon - \mu_h(\boldsymbol{x}_t, t)\|_2^2 \tag{38}$$

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

Combining Equation (39) and Equation (33), we obtain

$$\ln\left(\frac{p^{\text{data}}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p^h_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = -\frac{1}{2\tilde{\beta}_t^2}\left[\|\boldsymbol{x}_{t-1} - \mu_\theta(\boldsymbol{x}_t, t)\|_2^2 - \|\boldsymbol{x}_{t-1} - \mu_h(\boldsymbol{x}_t, t)\|_2^2\right] \tag{40}$$

$$= -\frac{1}{2\tilde{\beta}_t^2}\left[\|\Delta_h(\boldsymbol{x}_t, t) + \tilde{\beta}_t\epsilon\|_2^2 - \|\tilde{\beta}_t\epsilon\|_2^2\right] \tag{41}$$

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

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

For the full RND we obtain

$$\sum_{t=1}^{T} \ln\left(\frac{p^{\text{data}}_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{p^h_\varphi(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}\right) = \sum_{t=1}^{T}\left[-\frac{1}{2}\tilde{\beta}^{-2}\|\Delta_h(\boldsymbol{x}_t, t)\|_2^2 + \tilde{\beta}^{-1}\Delta_h(\boldsymbol{x}_t, t)^\top\epsilon\right], \tag{44}$$

which mimics a discretised version of the continuous RND in Equation (28).

B RESAMPLING STEP

B.1 PROOF TO THEOREM 4

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

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

On the other side, we have by definition that

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

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

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

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

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

Since $1 \leq \frac{\mathrm{d}\mathbb{P}_{\mathrm{data}}}{\mathrm{d}\mathbb{P}_h}(\boldsymbol{x}_{[0,T]})\frac{\exp(r(\boldsymbol{x}_0))}{c}$ if and only if $\ln\left(\frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\mathrm{data}}}(\boldsymbol{x}_{[0,T]})\frac{c}{\exp(r(\boldsymbol{x}_0))}\right) \leq 0$ 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

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

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) [1]:

$$\mathbb{P}_{h^*}(\cdot)=\mathbb{P}_{\text{forward}}(\cdot|\boldsymbol{x}_0)\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0) \tag{45}$$

$$=\mathbb{P}_{\text{forward}}(\cdot|\boldsymbol{x}_0)\mathbb{P}_{\text{data}}(\boldsymbol{x}_0)\cdot\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}(\boldsymbol{x}_0) \tag{46}$$

$$=\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}\mathbb{P}_{\text{data}}(\cdot) \tag{47}$$

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

$$\mathbb{P}_{h^*}=\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}\mathbb{P}_{\text{data}} \tag{48}$$

Then it follows that

$$\mathrm{KL}(\mathbb{P}_{h^*}||\mathbb{P}_{\text{data}})=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\mathbb{P}_{h^*}}\left[\ln\left(\frac{\mathrm{d}\mathbb{P}_{\text{data}}}{\mathrm{d}\mathbb{P}_{\text{data}}}(\boldsymbol{x}_{[0:T]})\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\text{data}}(\boldsymbol{x}_0)}\right)\right] \tag{49}$$

$$=\mathbb{E}_{\boldsymbol{x}_{[0:T]}\sim\frac{\tilde{\mathbb{P}}_h^0}{p_{\text{data}}}\mathbb{P}_{\text{data}}}\left[\ln\left(\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\text{data}}(\boldsymbol{x}_0)}\right)\right] \tag{50}$$

$$=\mathbb{E}_{\boldsymbol{x}_0\sim\tilde{\mathbb{P}}_h^0}\left[\ln\left(\frac{\tilde{\mathbb{P}}_h^0(\boldsymbol{x}_0)}{p_{\text{data}}(\boldsymbol{x}_0)}\right)\right]=\mathrm{KL}(\tilde{\mathbb{P}}_h^0,p_{\text{data}}) \tag{51}$$

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

$$\mathrm{KL}(\tilde{\mathbb{P}}_h,\mathbb{P}_{\text{data}})=\mathbb{E}[\mathrm{KL}(\tilde{\mathbb{P}}_h(\cdot|\boldsymbol{x}_0),\mathbb{P}_{\text{data}}(\cdot|\boldsymbol{x}_0))]+\mathrm{KL}(\tilde{\mathbb{P}}_h^0,p_{\text{data}}) \tag{52}$$

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

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

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

$$\mathcal{F}(\mathbb{P}_{h^*})=\mathrm{KL}(\mathbb{P}_{h^*},\mathbb{P}_{\text{data}})-\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\mathbb{P}_{h^*}}[r(\boldsymbol{x}_0)] \tag{53}$$

$$=\mathrm{KL}(\mathbb{P}_{h^*},\mathbb{P}_{\text{data}})-\mathbb{E}_{\boldsymbol{x}_{[0,T]}\sim\tilde{\mathbb{P}}_h}[r(\boldsymbol{x}_0)] \tag{54}$$

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

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

[1] 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.

C Experimental Details and Additional Results

C.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 the 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 $\alpha_{\text{KL}} = 0.2$. We initialise the buffer with samples from the initial score-model. Then we perform 40 fine-tuning steps with 200 gradient updates per iteration. We choose c such that 30% of the samples will be rejected. The total training time is approximately 1.5min on a single NVIDIA GeForce RTX 4090.

C.2 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 Equation (10) 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

$$D_{\text{logvar}}(\mathbb{P}_h, \mathbb{P}_{\text{data}}; \mathbb{W}) = \text{Var}_{\boldsymbol{H}^{g_t}_{0:T} \sim \mathbb{W}} \left[\ln \frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}} (\boldsymbol{H}^{g_t}_{0:T}) \right], \tag{56}$$

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

$$\begin{aligned} \boldsymbol{H}_T &\sim Q^{f_t}_T[p_{\text{tilted}}] \\ \mathrm{d}\boldsymbol{H}_t &= \left(f_t(\boldsymbol{H}_t) - \sigma_t^2 (s_t(\boldsymbol{H}_t) + g_t(\boldsymbol{H}_t)) \right) \mathrm{d}t + \sigma_t \, \overleftarrow{\mathrm{d}\mathbf{W}}_t. \end{aligned} \tag{57}$$

In particular, if we choose $g_t = \text{stop\_grad}(h_t)$ as a detached copy of the current estimation h_t, we obtain

$$\begin{aligned} \ln \frac{\mathrm{d}\mathbb{P}_h}{\mathrm{d}\mathbb{P}_{\text{data}}} (\boldsymbol{H}^{g_t}_{0:T}) = -\frac{1}{2} \int_0^T \sigma_t^2 \|h_t(\boldsymbol{H}^{g_t}_t)\|^2 \mathrm{d}t + \int_0^T \sigma_t^2 (g_t^\top h_t)(\boldsymbol{H}^{g_t}_t) \mathrm{d}t - r(\boldsymbol{x}^{g_t}_0) \\ + \int_0^T \sigma_t h_t^\top (\boldsymbol{H}^{g_t}_t) \overleftarrow{\mathrm{d}\mathbf{W}}_t, \end{aligned} \tag{58}$$

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.

In Figure 4 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" inthe image, giving evidence that this particular model has not fully converged.

C.3 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 $\epsilon^\theta_t(\boldsymbol{x}_t)$ and a conditional $\epsilon^\theta_t(\boldsymbol{x}_t, c)$ noise-prediction model are learned by randomly masking out the text prompt c. During sampling the linear

Figure 4: 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.

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 4. We used a buffer of 200 images and sample 32 new images at every iteration, the parameter c was chosen such at 40% of the images were accepted. We used a total of 50 iteration, where we did 60 gradient descent steps with a batch size of 4 at each iteration. Fine-tuning took about 4 hours on a single NVIDIA Geforce RTX 4090. We used the AdamW optimiser (Loshchilov, 2017) with a learning rate of 1×10^{-4} and a cosine decay to 1×10^{-5}. For the final sampling evaluation we used the DDIM scheduler with 50 time steps. We used the default guidance scale of 7.5 for Stable Diffusion.