Boosting methods for refining diffusion models

Ahmed Mehdi Inane

August 28, 2024

Version: Reviewer’s version

Université Paris-Dauphine

LAMSADE

Machine Intelligence and Learning Systems (MILES)

Master's thesis

Boosting methods for refining diffusion models

Ahmed Mehdi Inane

1. Supervisor Alexandre Vérine
Machine Intelligence and Learning Systems (MILES)
Université Paris-Dauphine

2. Supervisor Yann Chevaleyre
Machine Intelligence and Learning Systems (MILES)
Université Paris-Dauphine

August 28, 2024

Ahmed Mehdi Inane
Boosting methods for refining diffusion models
Master's thesis, August 28, 2024
Reviewers: Etienne Decencière and Gabriel Peyré
Supervisors: Alexandre Vérine and Yann Chevaleyre

Université Paris-Dauphine
Machine Intelligence and Learning Systems (MILES)

LAMSADE
Place du Maréchal de Lattre de Tassigny
75016 and Paris

Abstract

This thesis work presents two methods in refining the generation of generative models, particularly diffusion models. Popular generative models typically contain billions of parameters. They incur a hefty training cost, in both energy and time, and the architecture choice often restrict the estimation to have a suboptimal estimation of the target distribution.

As a result, we are interested in ways to boost generative models with smaller models, in order to mitigate generation errors, due to either estimation or sampling. Focusing our work on diffusion models, we thus propose two methods : improved discriminator guidance, which consists in using a discriminator to correct the estimation error, and f-Policy gradient, which circumvents the errors induced by solving the backward SDE of a diffusion process by using reinforcement learning and dense signals provided by measuring the f-divergence between the two distributions.

We provide theoretical guarantees proving the validity of our methods and demonstrate their effectiveness on standard benchmark datasets. Our experiments show that boosting improves the generation process without incurring a substantial cost that would otherwise be obtained by fine-tuning the model.

Acknowledgement

Many thanks to Alex for teaching me so many things, and for allowing such a great collaboration to happen. I thank him for his patience, enthusiasm and for setting up a great learning environment. You have been a great mentor ! I thank Yann for trusting me with this research project, and for providing meaningful advice and insightful mathematical observations.

During this internship, I worked at the MILES team, and I would like to thank all my colleagues that made this research experience incredible. A special mention to Matteo and Augustin for quality coffee breaks, Lucas and Erwan for their sound advice, and to all the team for their support and kindness.

Finally, I would like to thank my family and friends for their support and encouragement during this internship. Thank you Carla for being there for me.

This work has inspired me to continue research within machine learning, and I hope it will be the first of many more to come.

Contents

1 Introduction

During the past decade, the use of artificial intelligence (AI) and machine learning (ML) methods has spread across various sectors, driving innovation and efficiency at unprecedented scales. The development of hardware components such as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs), combined with the availability of large amounts of data has shown the potential of Machine Learning models to uncover latent patterns in data, and has driven advancements in fields such as image and natural language processing, autonomous driving, healthcare, and finance. A particular class of ML models, termed as generative modeling, has shown promise in the recent years by its ability to generate realistic samples by training on a dataset.

Generative modeling is a subfield of machine learning that focuses on learning the underlying distribution of a dataset to generate new samples. It has proven to be effective for numerous types of data, including language [Ope+24], image generation [Rom+22], and drug discovery [Jum+21]. Recent generative models are based on neural networks, and prominent models include Generative Adversarial Networks (GANs) [Goo+14], Variational Auto-Encoder (VAEs) [KW19], and Diffusion models [Son+21].

The success of generative models hides their training procedure that is usually long, computationally expensive, and requires a large amount of data. The training of generative models is challenging due to the high-dimensional nature of the data and the need to learn complex distributions. For instance, figure 1.1 displays an exponential increase on the number of parameters of architectures involved in generative models. This raises numerous questions on the *frugality* of the training procedure. Several research directions have been proposed to address this issue, including the use of transfer learning [Zhu+20], neural architecture search [VCC24] and pruning methods [Sun+24].

In this work, we focus on a subset of generative models, that is diffusion models. Diffusion models consists in gradually adding noise to the data and then learning to reverse this process to reconstruct the original data. This type of model has gained attention for its potential to generate high-quality and diverse samples, while providing a theoretically sound framework based on Markov chains [HJA20] and stochastic differential equations [Son+21]. We are particularly interested in

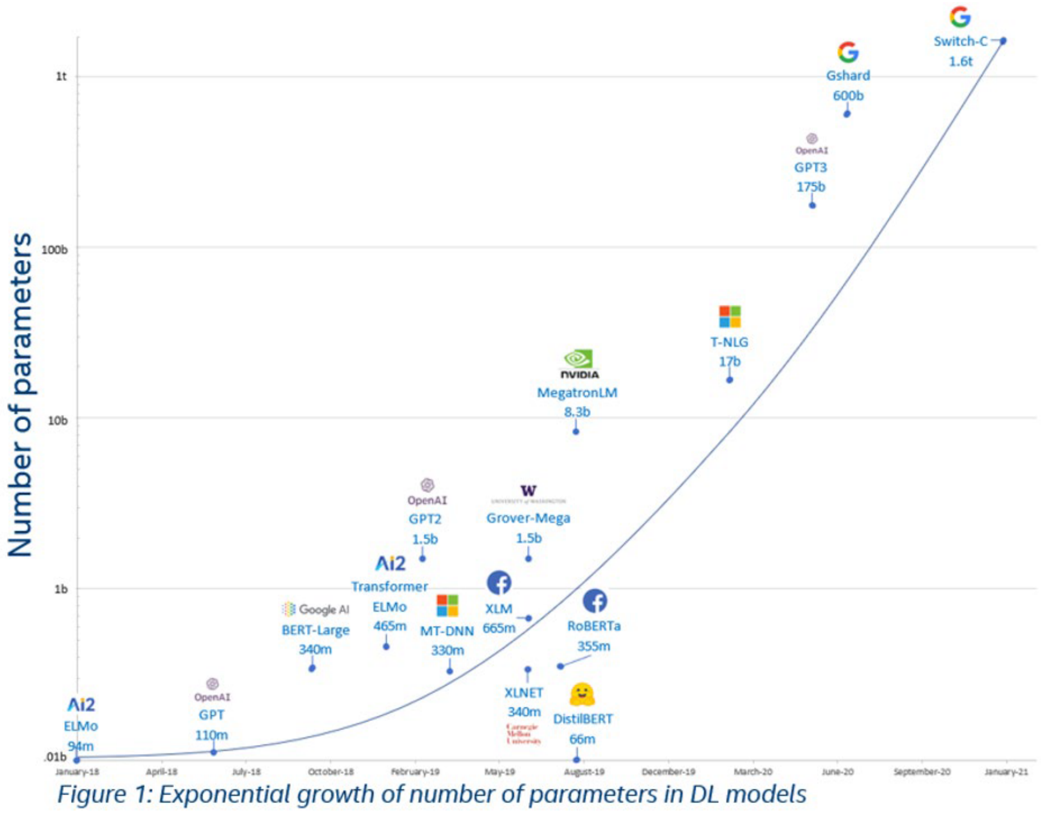


Fig. 1.1: Number of parameters of architectures used in large language models. Source: `https://www.webagesolutions.com/blog/generative-ai-engineering`

developing methods to refine a pre-trained diffusion model without re-training it. That is, improve its generation abilities without using the important computational power required to train the whole model. More specifically, we aim to investigate the potential of using boosting methods, i.e using auxiliary smaller models, to improve the generation capabilities of a pre-trained diffusion model. A smaller model will thus act as correcting the errors made by the pre-trained model.

The main sources of error that a diffusion model can make stem from *approximation* erros and *sampling* errors. Approximation errors denote the accuracy of estimation of the score compnent during the training phase, and sampling errors refer to the errors made during the sampling phase. We thus raise the following question :

Can we devise boosting methods to correct the errors made by a pre-trained diffusion model ?

We answer to this question by presenting two methods : discriminator guidance and f-Restart sampling. The former method attemps to correct the approximation error by using a smaller discriminator network to correct the score. f-Restart sampling attempts to correct the sampling error by using a reinforcement learning procedure during the sampling process, deciding between performing a denoising step or injecting noise as a correction term. The structure of our thesis is as follows :

- **Chapter 2 :** In this section, we give an overview of generative models, then present the derivation of diffusion models. Then, we list a few refinement methods proposed for generative models that are based on density ratio estimation technique, motivating the use of a discriminator.

- **Chapter 3 :** In this section, we present discriminator guidance as a method for correcting the approximation error involved in training a score network. We improve upon previous results by deriving an optimal objective function, and we expose the experimental results on standard image generation benchmarks.

- **Chapter 4 :** In this section, we present restart sampling as a boosting-free method for refining the sampling error of a diffusion model. We then propose f–Restart sampling, a method that improves upon the previous one by incorporating a reinforcement learning algorithm to dynamically denoise samples.

Background 2

This chapter introduces the necessary background that is relevant to our work. In the following, we provide an overview of generative modeling methods in machine learning. Then, we present diffusion models as the generative method we would like to improve. Finally, we present a few density ratio estimation techniques as this will be the backbone of our refinement techniques.

2.1 Generative modeling

Generative modeling denotes a branch of unsupervised learning that aims to construct a **model** of the data $x \in \mathcal{X}$. By considering that a dataset can be described by a probability distribution $p(x)$ - that we call a *target* distribution - the challenge is to estimate such a distribution by finding the best set of parameters able to approximate it, among a set of potential *generators* G. A notable ability of generative models is generating new data instances that resemble the observed data, by sampling from the *estimated* distribution $\hat{p}(x)$. Through the rapid development of deep learning and the abundance of training data over the past decade, generating data has attained impressive performances across various data types, often in high dimensions, such as language [Ope+24], visual data such as images [Rom+22] and videos [Bro+24], and speech [Oor+16]. This progress went on to include different types of input, termed as multimodal generative models [SM22].

2.1.1 Theoretical formulation

Assumption 1. *We assume that the domain of interest $\mathcal{D} \subset \mathcal{X}$ is governed by a probability distribution $P_{\mathcal{D}}$ of density $p(x)$.*

Given a training dataset $\mathbf{x}$ of n points sampled i.i.d from $P_{\mathcal{D}}$, we would like to approximate the target density $p(x)$ with a parametric family of models. Our focus is on deep learning based approaches involving neural networks. Thus, given a foxed architecture, we denote by Θ the set of all weights represented by this architecture. The traditional approach in density estimation with a parametric family of models is maximum likelihood estimation (MLE) or maximum a posteriori (MAP). The likelihood function $L(\theta \mid X)$ of a parameter $\theta \in \Theta$ given the data X is defined as the joint probability of the observed data:

$$L(\theta \mid X) = \prod_{i=1}^{n} p(x_i \mid \theta). \tag{2.1}$$

The maximum likelihood estimation problem involves finding the parameter θ that maximizes the likelihood function. Formally, we define the MLE of θ as:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta \mid X). \tag{2.2}$$

In practice, it is often more convenient to work with the log-likelihood function, since the logarithm is a monotonic function. The log-likelihood of θ is given by:

$$\ell(\theta \mid X) = \log L(\theta \mid X) = \sum_{i=1}^{n} \log p(x_i \mid \theta). \tag{2.3}$$

Thus, the MLE problem can also be formulated as maximizing the log-likelihood:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \ell(\theta \mid X). \tag{2.4}$$

This problem can be generalized as a **density estimation** problem, using the concept of statistical divergences.

Definition 1. *Given two probability distributions P, Q over $\mathcal{X}$, a statistical divergence $D(.\|.)$ is a function :*

$$D(P\|Q) : \mathcal{P} \times \mathcal{Q} \to [0, \infty), \tag{2.5}$$

where $\mathcal{P}$ and $\mathcal{Q}$ are the spaces of probability distributions on $\mathcal{X}$.

The function $D(P\|Q)$ satisfies the following properties:

- *Non-negativity: $D(P\|Q) \geq 0$ for all $P, Q \in \mathcal{P} \times \mathcal{Q}$.*
- *Identity of indiscernibles: $D(P\|Q) = 0$ if and only if $P = Q$ almost everywhere.*

The absence of the symmetry assumption excludes statistical divergences from being metrics. However, they are still effective in comparing probability distributions. Table 2.1 provides examples of statistical divergences of interest in our work. Given

Tab. 2.1: Generic Mathematical Expressions for Statistical Divergences

Category	Generic Mathematical Expression
F-Divergences	$D_f(P\|Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$
Renyi Divergences	$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \int p(x)^\alpha q(x)^{1-\alpha} dx$
Bregman Divergences	$D_\phi(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{p}) - \phi(\mathbf{q}) - \langle \nabla\phi(\mathbf{q}), \mathbf{p} - \mathbf{q} \rangle$

a divergence D, a target density $p(x)$ and a family of parametric models Θ, one can formulate the optimal density as the solution of :

$$\hat{\theta} = \arg\min_{\theta \in \Theta} D(\hat{p}(x;\theta) \| p(x)) \tag{2.6}$$

Once the density is estimated, one can sample from a generator $G_{\hat{\theta}}$ function, according to the type of generative model used.

2.1.2 Sampling from an estimated distribution

In this subsection, we give three examples of the training and sampling schemes of popular generative models in deep learning.

- **Generative adversarial networks (GANs)** [Goo+14] consist in simultaneaous training of two different networks having opposing objectives. The first network, called a discriminator D_ϕ discriminates between samples from the *target* distribution and samples from another distribution. The second network, called a generator G_θ, attempts to sample from the target distribution to fool the discriminator. Essentially, this consists in solving the following min-max problem [NCT16], given a convex, lower-semi continuous function f:

  $$\min_\theta \max_\phi V(D_\phi, G_\theta) = \mathbb{E}_{x \sim p(x)}[D_\phi(x)] - \mathbb{E}_{z \sim p_\theta(z)}[f^*(D_\phi(G_\theta(z)))] \tag{2.7}$$

  where f^* denotes the Fenchel conjugate of f. This objective function corresponds to a variational lower bound on the f-divergence between the target distribution and the generator distribution [NWJ10]. Once this objective is optimized, the generator G_θ is then used to sample from the estimated distribution. While GANs are celebrated for their sample quality, they often suffer from mode collapse which leads to a lack of diversity [ZLY18],and an unstable training due to the double optimization induced by the training objective.

- **Variational autoencoders (VAEs)** [KW22] are probabilistic generative models that frame the problem of data generation as a Bayesian inference task. The key idea is to encode inputs into a latent (hidden) space via a neural network, and then decode from this latent space back to the data space. The encoder learns a distribution $q_\phi(z|x)$ over the latent space conditioned on the input data x, aiming to approximate the true but intractable posterior $p(z|x)$. The decoder, parameterized by θ, attempts to reconstruct the input x from the latent variable z, sampled from $q_\phi(z|x)$. The loss function of a VAE can be described as:

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + D_{\text{KL}}(q_\phi(z|x) \| p(z)), \tag{2.8}$$

where the first term is the expected log likelihood of the decoder (which measures reconstruction accuracy), and the second term is the Kullback-Leibler divergence between the encoded distribution and a prior distribution (typically a standard normal), acting as a regularizer. The VAE thus balances between accurate reconstruction and a meaningful, well-structured latent space. Sampling from a VAE involves passing randomly drawn samples from the prior $p(z)$ through the decoder to generate new data points that resemble the training data. As opposed to GANs, VAEs have a mass covering advantage but lack the quality provided by GAN generated samples. A recent line of work attempts to combine both strengths of these models to provide both quality and diversity to the generation process [GZ22].

- **Diffusion Models** are another class of generative models that have recently gained significant attention due to their ability to generate high-quality samples. Similar to VAEs, diffusion models operate over a latent space and involve a gradual transformation process. However, instead of encoding and decoding directly, diffusion models start by gradually adding noise to the data until a simple noise distribution is reached, effectively 'diffusing' the data. By inspiring from theories in nonequilibrium thermodynamics, this process is then reversed to generate new samples from noise by learning to denoise through a series of steps. This is conceptually similar to the way VAEs attempt to map data to and from a latent space, but diffusion models do so over many more transformations, providing a smoother mapping and potentially richer generative capabilities. In the following section, we delve more in the theory behind diffusion models as it represents the main focus of our work.

2.2 Diffusion models

In this section, we derive the theoretical formulation of diffusion as a generative process. We begin by describing its derivation from Langevin dynamics, then present the different formulations and improvements over the past decade, and formulate the problem at hand in evaluating the estimation error induced by both the training and sampling processes.

2.2.1 Langevin diffusion

A popular method to accelerate sampling from a target distribution P_0 with density of the form $p_0(x) \propto e^{f(x)}$ is Langevin diffusion. In its discrete form, the algorithm consists in sampling a point $X_0 \sim \mathcal{N}(0, I_d)$ and then updating the points by a gradient descent like update :

$$X_{k+1} = X_k - \tau \nabla p_0(X_k) + \sqrt{2\tau} W_k \tag{2.9}$$

where $W_k \sim \mathcal{N}(0, I_d)$ denotes a brownian noise. Its continuous form consists in the following Langevin stochastic differential equation (SDE) of the following form :

$$dX_t = -\nabla p_0(X_t)dt + \sqrt{2} dW_t \tag{2.10}$$

where W_t is a Wiener process. Depending on the smoothness of f (CITATION), one can show that X_t converges in distribution to P_0 regardless of the distribution of X_0. This method establishes a clear link between SDEs and sampling methods, and is foundational in the formulation of diffusion models as generative models. Thus, if by estimating correctly $\nabla \log p(x)$, one can use the Langevin sampling algorithm in order to sample from distribution P. This raises the following question : How to estimate $\nabla \log p(x)$?

2.2.2 Score matching

This consists in minimizing the *Fisher divergence* between the target distribution P and the model distribution $\hat{P}$:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathcal{J}_{SM}(\theta) = \arg\min_{\theta \in \Theta} \frac{1}{2} \mathbb{E}_p \left[||\nabla \log p(x) - s_\theta(x)||^2 \right] \tag{2.11}$$

where s_θ is the model parameterized by θ. In practice, the only access we have to a target distribution P_0 is through samples. Thus, the expression of $\nabla p_0(x)$ is unknown and the Fisher divergence cannot be computed. This is however alleviated by a family of methods, termed as **score matching** [Hyv05; Vin11]. [Hyv05] first show by using a partial integration trick that minimizing equation 2.11 is equivalent to minimizing the following quantity, that we term as *implicit score matching* :

$$J_{ISM}(\theta) = \mathbb{E}_p \left[||s_\theta(x)||^2 + \sum_{i=1}^{d} \frac{\partial_i s_\theta(x)}{\partial x_i} \right] \tag{2.12}$$

This quantity excludes the intractable term $\nabla \log p(x)$ and can be computed by having access to samples from the distribution P by Monte Carlo estimation. However, this method requires to compute the derivative of s_θ, which can be computationally expensive in high dimensions. Furthermore, the convergence of the optimization process can be slow for non convex functions J_{ISM} and lead to instabilities, which motivated [Vin11] to propose a new score matching loss, inspired from the denoising auto-encoder framework. This consists in deriving a *smoothing* of the distribution P, by injecting noise to its samples. The **denoising score matching** objective is thus formulated as, with a differentiable perturbation kernel $p_\sigma(\tilde{x}|x)$ and joint density $p_\sigma(\tilde{x}, x) = p_\sigma(\tilde{x}|x)p(x)$:

$$\mathcal{J}_{DSM}(\theta) = \mathbb{E}_{x,\tilde{x}\sim p_\sigma}\left[\frac{1}{2}\,||s_\theta(\tilde{x}) - \nabla \log p_\sigma(\tilde{x}|x)||^2\right] \tag{2.13}$$

Note that by setting $p_\sigma(\tilde{x}|x)$ as the density of $\mathcal{N}(x, \sigma^2 I_d)$, we have that $\nabla \log p_\sigma(\tilde{x}|x) = \frac{1}{\sigma^2}(x - \tilde{x})$, thus alleviating the need to have an explicit expression for the score function, and of computing high dimensional derivatives. This smoothing perturbation thus allows for estimating the score function needed to compute equation 2.11, and alleviating the computational burden induced by equation 2.12. The work of [Vin11] was critical in introducing the framework of diffusion models, that would be subsequently introduced. Once the score is estimated, one can sample from the *estimated* distribution $\hat{P}$ by using Langevin dynamics. The training and sampling procedures are described in algorithm 1, where $\mathcal{J}$ is either $\mathcal{J}_{ISM}$ or $\mathcal{J}_{DSM}$. Since the sampling requires first training a model, which induces an approximation error, and using Langevin dynamics, which introduces estimation errors, we consider that we sample from a distribution $\hat{P}$ that slightly deviates from P.

[Son+21] raised an important question : What happens in low density regions ? Since one only has access to P through its samples, estimating $\nabla \log p(x)$ for an x with low density requires access to large amounts of samples, which is not feasible in practice. The initial score matching loss $\mathcal{J}_{SM}$ (equation 2.11) can be unfolded as follows :

$$J_{SM}(\theta) = \frac{1}{2}\int p(x)||\nabla \log p(x) - s_\theta(x)||^2 dx \tag{2.14}$$

Thus, the weighting in $p(x)$ gives very little importance to low density regions, making the score estimation likely inaccurate in those regions. Figure 2.1 shows the issues graphically in a simple two dimensional case. As a result, the initialization x_0 used in Langevin dynamics is very likely to reside in those regions with low density $p(x)$. This makes the value $s_\theta(x_0)$ likely to be erroneous, which might lead to derail the Langevin sampling procedure. This raises the following question : **How can one account for accurate estimation of the score in low density regions ?**

Algorithm 1 Score matching and sampling with Langevin dynamics

```
Initialize parameters $\theta$
Choose learning rate $\eta$, noise level $\sigma$, number of training iterations $N$ and number of samples to generate $N_{\text{samples}}$
**Score Matching:**
**for** $n = 1$ to $N$ **do**
    Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta J(\theta)$
**end for**
**Langevin Dynamics for Sampling:**
**for** $n = 1$ to $N_{\text{samples}}$ **do**
    Sample initial point $x_0$ from some distribution
    **for** $t = 1$ to $T$ **do**
        Update sample using Langevin step:
            $$x_{t+1} = x_t + \frac{\eta}{2} s_\theta(x_t) + \sqrt{\eta} \cdot \mathcal{N}(0, \sigma^2)$$
    **end for**
**end for**
**Output:** Trained model $s_\theta$ and samples $\{x_1, \ldots, x_{N_{\text{samples}}}\}$
```

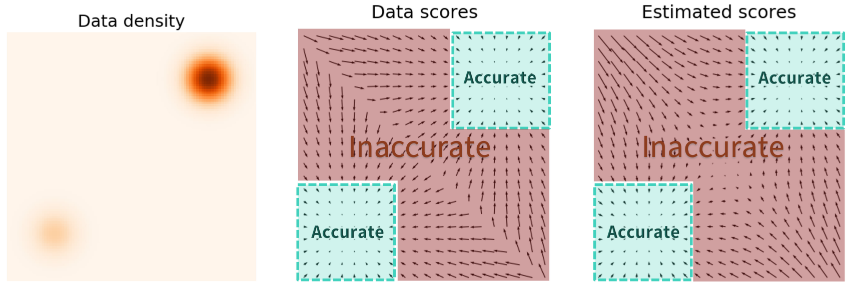


Fig. 2.1: Score estimation for low density regions in the case of Gaussian mixtures. Source : `https://yang-song.net/blog/2021/score/`

2.2.3 Injecting more noise

In order to account for accurately estimating the score for regions with low density, [SE20] propose to inject noise in several scales to the data points. When the noise scale σ is sufficiently high, regions with low density for distribution P are more likely to be populated for distribution P_Σ. This induces a trade-off between the magnitude of the added noise, i.e the size of σ, which populates the low density regions, and the alteration to the data distribution. In order to circumvent this issue, [SE20] propose *annealed Langevin dynamics*, which consists in altering data with several noise scales $\sigma_1, \sigma_2, \ldots \sigma_L$, and training the model to minimize the following sum of score matching losses at different noise scales :

$$J_{ASM}(\theta) = \sum_{i=1}^{L} \lambda(i) E_{p_{\sigma_i}} \left[||\nabla \log p(x) - s_\theta(x, i)||^2 \right] \tag{2.15}$$

where $\lambda(i)$ is a weighting schedule, and $s_\theta(x,i)$ is a model that takes in input the current point and the scale of the added noise. Note that the terms in equation 2.15 can be approximated using the aforementioned score matching techniques. Once a model is trained to minimize J_{ASM}, the sampling procedure generating samples from $\hat{P}$ is described by algorithm 0. [SD+15; HJA20] proposed a similar framework that was labeled as *denoising diffusion probabilistic models*. Akin to [SE20], they propose a noise scale $\beta 1, \beta 2, \dots, \beta_L$, and they create a Markov chain for each training point $x_0 \to \{x_0, x_1, \dots, x_L\}$, where each $x_i \sim \mathcal{N}(\sqrt{1-\beta_i}x_{i-1}, \beta_i I_d)$, thus $x_i \sim \mathcal{N}(\sqrt{\alpha_i}x_0, (1-\alpha_i)I_d)$, with $\alpha_i = \prod_{k=1}^{i}(1-\beta_k)$. By denoting $p(x_i|x_0) = \mathcal{N}(x_i|\sqrt{\alpha_i}x_0, (1-\alpha_i)I_d)$, a noise conditional model $s_\theta(\tilde{x}, i)$ is trained to minimize the following objective, that is similar to the one described by equation 2.15 :

Algorithm 2 Annealed Langevin Dynamics for Sampling [SE20]

```
Require: Sequence of noise levels $\{\sigma_i\}_{i=1}^{L}$, step size $\epsilon$, number of iterations $T$
Initialize $x_0$
for $i \leftarrow 1$ to $L$ do
    Set step size $\alpha_i \leftarrow \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$
    for $t \leftarrow 1$ to $T$ do
        Draw $z_t \sim \mathcal{N}(0, I)$
        Update $x_t \leftarrow x_{t-1} + \frac{\alpha_i}{2} s_\theta(x_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$
    end for
    Set $x_0 \leftarrow x_T$
end for
return $x_T$
```

$$\mathcal{J}_{DDPM}(\theta) = \sum_{i=1}^{L}(1-\alpha_i)\mathbb{E}_p\left[\mathbb{E}_{p(.|x)}\left[||s_\theta(\tilde{x},i) - \nabla \log p(x_i|x)||^2\right]\right] \tag{2.16}$$

The sampling process is done by starting from $x_L \sim \mathcal{N}(0, I_d)$, then reversing the Markov chain :

$$x_{i-1} = \frac{1}{1-\beta_i}(x_i + \beta_i s_\theta(x_i, i)) + \sqrt{\beta_i} z_i \tag{2.17}$$

with $z_i \sim \mathcal{N}(0, I_d)$ and $i \in \{L, \dots, 1\}$. The generative procedure described by [HJA20] is an alternative to using Langevin dynamics for generating samples using an estimated score, but one sees numerous similarities with the method proposed by [SE20]. Notably, both methods use increasing noise schedules to perturb the points, and estimate a noise conditional score model. One can thus ask : **Is there a unifying framework for generating with noise conditional score models ?**

2.2.4 Diffusion process

The answer to this question was obtained by answering another question : What happens when $L \to \infty$, i.e when the number of noise scales is infinite ?

When $L \to \infty$, the number of perturbations can be viewed as continuous and increasing noise perturbations. Thus, this can be seen as a stochastic process, and can sometimes be the solution of stochastic differential equations (SDEs). In order to generalize the idea of denoising score matching to continuous noise injections, we can mention diffusion processes. Diffusion processes are derived from modeling the dynamics of molecular systems, particularly from the theory of Langevin dynamics. In summary, a simplified model of a system can be created by compensating the omitted degrees of freedom through stochastic differential equations. Fundamentally, a diffusion process $\{x_t\}_{t=0}^{T}$ is a Markov process indexed by a continuous time variable $t \in [0, T]$. By adding iteratively noise to the samples, one can describe the *forward* diffusion process as a solution the following Itô SDE :

$$dx = f(x,t)dt + g(t)dw \tag{2.18}$$

where dw is the standard Wiener process (Brownian motion). $f(x,t)$ is called the *drift* coefficient of $x(t)$ and $g(t)$ is called the *diffusion* coefficient of $x(t)$. The distribution of x_T when $T \to \infty$ is called the *prior* distribution and can be carefully chosen by tailoring f and g in the aforementioned SDE [Son+21]. Equation 2.18 has a unique solution when the drift and diffusion coefficients are Lipschitz in all of their arguments [ØØ03]. Generally, one chooses f and g such that p_T is an easy distribution to sample from, like a zero-mean isotropic Gaussian distribution. For instance, with a linear drift $f(x) = -x$ and a constant diffusion coefficient $g(t) = \sqrt{2}$, one has that p_T converges exponentially fast to $\mathcal{N}(0, I_d)$ (CITATION). In the following, we denote P_t as the distribution of a sample from P at timestep t. We also denote by $p_t(x_t|x_0)$ the conditonal distribution of a noisy point x_t after starting with point x_0. When does the score intervene in this case ? [And82] proves that the reverse of a diffusion process, starting at distribution P_T and ending in distribution P is also a diffusion process, solution to the following SDE, that we label as the *backward SDE* :

$$dx = \left(f(x,t) - g(t)^2 \nabla \log p_t(x) dt\right) + g(t)dw \tag{2.19}$$

Thus, if one learns $\nabla \log p_t(x)$ for $t \in [0, T]$, one can sample from P by first sampling a point from P_T and solving equation 2.19. [Son+21] propose thus a continuous version of the discrete noise conditional score estimation, with the following objective function :

$$J_{\text{CSM}}(\theta) = \mathbb{E}_t \left[\lambda(t) \mathbb{E}_P \left[\mathbb{E}_{P_t} \frac{1}{2} \left\| s_\theta(x_t, t) - \nabla_{\mathbf{x}} \log p_t(x_t|x) \right\|^2 \right] \right] \tag{2.20}$$

Once the score is accurately estimated, one can solve the backward SDE using $s_\theta(x,t)$ as an estimator for $\nabla \log p_t(x)$. Several solving schemes are available, notably numerical solvers that discretize the backward SDE such as the stochastic Runge-Kutta and Euler-Maruyama methods [KP92]. Algorithm 0 describes this sampling procedure using the Euler Maruyama sampler. Furthermore, [Son+21] proposed a different sampling method by using the corresponding deterministic reverse process, labeled as the *probability flow* ordinary differential equation (ODE). It is expressed as :

$$dx = f(x,t) - \frac{1}{2}g(t)^2\nabla \log p_t(x)dt \tag{2.21}$$

and can be equivalently solved by ODE numerical solvers that discretize the ODE. This version provides a computational advantage as less discretization steps are needed for a fair quality, but there is a tradeoff between image quality by SDE generation and the number of function evaluations (computation cost) via ODE generation [Xu+23].

Algorithm 3 Training a Score-based Diffusion Model

```
Require: Dataset $\mathcal{D}$, learning rate $\eta$
 1: Initialize parameters $\theta$
 2: while not converged do
 3:     Sample mini-batch $\{x^{(i)}\}$ from $\mathcal{D}$
 4:     Update $\theta \leftarrow \theta - \eta\nabla_\theta \mathcal{J}_{CSM}(\theta)$
 5: end while
 6: return Trained model parameters $\theta$
```

Algorithm 4 Sampling via Discretized Backward SDE

```
Require: Trained parameters $\theta$, number of steps $N$, time horizon $T$
 1: Initialize $x_T \sim p_T(x)$ (e.g., Gaussian)
 2: Initialize current timestep $t_n = T$
 3: for $n = N$ down to 1 do
 4:     Set time step $\Delta t = T/N$
 5:     Update current timestep $t_n = t_n - \Delta t$
 6:     Sample noise $\epsilon_n \sim \mathcal{N}(0,I)$
 7:     Update using backward SDE discretization:
            $x_{n-1} = x_n - s_\theta(x_n, t_n)\Delta t + \sqrt{\Delta t}\epsilon_n$
 8: end for
 9: return Sample $x_0$
```

[Kar+22] proposed a unifying framework for diffusion-based generative models, and showed that different sampling schemes can be used with the same diffusion model. Table 2.2 lists a few configurations for popular diffusion based generative models.

Thus, diffusion appears as a viable choice for approximating probability distributions. However, as most distribution approximation methods, estimation errors arise in practice. For the diffusion framework, there is two main sources of error : **approximation** errors related to the training of the score, and **sampling** errors related to the discretization of the backward ODE or SDE. [Lai+23a; Cha+23] notably showed that diffusion models in practice are not conservative for large values of t, i.e does not define vector fields of real valued functions. This poses a new challenge, Tab. 2.2: Specific design choices employed by different model families

Model	**VP** [SE20]	**VE** [HJA20]	**EDM [Kar+22]**
ODE Sampling	Euler	Euler	2nd order Heun
Time steps t_i	$1 + \frac{i(1-\epsilon_0)}{N-1}$	$\sigma_{max}^2 \left(\frac{\sigma_{min}^2}{\sigma_{max}^2}\right)^{\frac{i}{N-1}}$	t
Training (Sec. 5)			
Noise distribution	$\sigma^{-1}(\sigma) \sim \mathcal{U}(\epsilon_t, 1)$	$\ln(\sigma) \sim \mathcal{U}(\ln(\sigma_{min}), \ln(\sigma_{max}))$	$\ln(\sigma) \sim \mathcal{N}(P_{mean}, P_{std}^2)$
Loss weighting $\lambda(\sigma)$	$\frac{1}{\sigma^2}$	$\frac{1}{\sigma^2}$	$\left(\frac{\sigma^2+\sigma_{data}^2}{\sigma \cdot \sigma_{data}}\right)^2$

which is to mitigate the error sources related to estimation. As generation consists of both training and sampling, this calls for improvement during both the training and the sampling stages. Thus, we ask the following question : **How to refine the generative process ?** In the following section, we will answer this broadly for generative models and specifically provide an answer for diffusion models in sections 3 and 4. By noting $\tilde{P}$ the distribution induced by training a generic generative model and sampling from it, one can check a discrepancy measure between the two distributions (Definition 1) to evaluate *how accurate* the estimation procedure is. In our work, we will focus on the family of f-divergences, that compute the following metric with respect to the Lebesgue measure :

$$D_f(\tilde{P}||P) = E_P\left[f(\frac{P}{Q})\right] = \int p(x) f(\frac{p(x)}{\tilde{p}(x)} dx \tag{2.22}$$

2.3 Expressivity of a generative model

The sources of error of interest in our case are the expressivity of the considered model, and the stopping iteration. When the number of iterations goes to infinity, the expressiveness of the family of parameters yield an optimal distribution $\tilde{P}$ for a given f-divergence, that is not necessarily equal to P. One controls the expressivity of a model by architecture design, choice of the objective function, and training paradigm.

2.3.1 Precision and recall as f-divergences

For generative models in general, the choice of an f in the minimization of an f-divergence yields different results in practice, for the same family of models. For instance, [Min+05] observed that optimizing Kullback-Leibler divergence($f(x) = x \log x$) tends to favor *mass-covering* models and that optimizing the reverse KL ($f(x) = -\log x$) and Jensen-Shannon $f(x) = \frac{1}{2}\left(x \log x - (x+1)\log\left(\frac{x+1}{2}\right)\right)$ tends to favor *mode-seeking* behaviors. Here, a mass covering model will produce diverse samples, covering a substantial part of the support of P. Conversely, a mode seeking model will display less diversity in the produced samples, but the generated samples will have a high target density p, which make them more *precise*. Depending on the use case, one would either want to produce samples of high quality or with high diversity. For example, using generative models for drug discovery would require precise samples, while using them to generate images for artistic synthesis might allow less precision for more diversity. INSERT PLOT HERE LIKE FIGURE 4.3 from alex thesis. This implicit trade-off was investigated by [Ver+24a] in the context of GANs and normalizing flows, and they derive a new divergence, termed as the precision and recall (P&R) divergence, that is dependent on a specified trade-off parameter λ. It is essentially an f-divergence, with a function f_λ defined as :

$$f_\lambda(x) = \begin{cases} \max(\lambda x, 1) - \max(\lambda, 1) & \text{if } \lambda \in [0, +\infty[\\ \mathbf{1}_{x=0} & \text{if } \lambda = +\infty. \end{cases} \tag{2.23}$$

In order to control the expressivity of a generative model with a specified trade-off between diversity and precision, [Ver+24a] propose a method for GANs and normalizing flows, that improves upon the traditional framework of f-GANs introduced by [NCT16]. Modifying the objective functions is thus a foundational method to control the expressivity of a generative model. This, in practice however, is not directly applicable to diffusion models as the loss is not expressed as an f-divergence, but rather as a score matching loss. In the following subsections, we will give examples that improve the expressivity of a score based model.

2.3.2 Using the theory of diffusion as a refinement process

We will give two examples that improved the likelihood of the generated distribution of diffusion models. One is based on conservativity, a fundamental property that the estimated score should satisfy, and another is based on the FOkker-Planck equation, that is related to the SDE formulation of score based models.

Definition 2. *A vector field, represented by a vector valued function $v : \mathcal{X} \to \mathbb{R}^d$, is said to be conservative if it can be expressed as the gradient of a real valued function*

By definition, the ground truth score $s(x,t) = \nabla_x \log p_t(x)$ is thus conservative. However, [SH21; Cha+23] highlighted that notable diffusion models, that are defined as U-Nets in practice, are not conservative. In order to ensure conservativity, [SH21; Sar+18] propose to constrain the architecrure of the model such that its output vector field is modeled as a gradient of a real valued function. This, however, constrains the expressivity of the architecture overall. [Cha+23], on the other hand, add a regularization term based on higher orders of the score to ensure the conservativity. This method allowed for more flexible architecture choices, and an improvement in likelihood computations of the estimated distributions. We will now present another method that helped enforce conservativity, introduced by [Lai+23b]. The ground truth density p_t associated to the forward diffusion SDE (equation 2.18) satisfies the Fokker-Planck equation [ØØ03] :

$$\frac{\partial p_t(x)}{\partial t} = -\frac{\partial}{\partial x}\left[f(x,t)p_t(x)\right] + \frac{\partial^2}{\partial x^2}\left[g(t)p_t(x)\right] \tag{2.24}$$

[Lai+23b] derive from this equation another equation that the ground truth score should satisfy, labeled as **the score FPE** (c.f Proposition 3.1 and Appendix G in [Lai+23b]) :

$$\frac{\partial s(x,t)}{\partial t} = \nabla_x\left[\frac{1}{2}g^2(t)\mathrm{div}_x(s(x,t)) + \frac{1}{2}g^2(t)\left\|s(x,t)\right\|^2 - \langle f(x,t), s(x,t)\rangle - \mathrm{div}_x(f(x,t))\right] \tag{2.25}$$

where $s(x,t) = \nabla_x \log p_t(x)$ denotes the ground truth score, and $\mathrm{div}_x(s(x,t)) = \sum_{i=1}^{d}\frac{\partial s(x,t)}{\partial x_i}$ denotes the divergence of the vector field $s(x,t)$. In practice, it is observed that most trained diffusion models do not satisfy the score FPE. By denoting by $\mathcal{R}(.)$ as the linear mapping corresponding to the right hand side of equation 2.25, one can define a *residual* ϵ_θ corresponding to the difference $\frac{\partial s_\theta(x_t,t)}{\partial t} - \mathcal{R}[s_\theta](x_t,t)$, evaluating how far the current estimated score is from satisfying the score FPE. This residual is added as a regularization term in various forms, depending on the used score matching loss, and thus enforces the score FPE to be satisfied. [Lai+23b] demonstrate that notable implications of satisfying equation 2.25 are :

- Reducing the KL divergence between the target distribution P_0 and the estimated density with ODE sampling (solved by discretization of equation 2.21)

- Enforcing the conservativity of the vector field $s_\theta(x_t,t)$

These methods elaborate a novel framework that is suitable for a more theoretically grounded training for diffusion based models. One of the immediate setbacks of adopting these methods is their incompatibility with pre trained models, that already display a good performance. We thus ask : **Are there methods that improve the generative process without retraining a model from scratch ? cCan they be used for a wide array of models ?**

2.3.3 Fine-tuning methods

[Luc+19] first noticed that when a GAN is generating data in a label limited setting, there was a significant advantage in using class labels during generation. Using this idea, [DN21] presents a boosting method for diffusion models, by incorporating class-specific information during the sampling process through a classifier. This idea was presented in order to make diffusion models increase their precision at the expense of their diversity (c.f subsection 2.3.1. Specifically, given a specified class y, one can train a classifier with a parameterized model c_ϕ to estimate $\nabla \log p(y|x_t, t)$ for every perturbed point x_t at every diffusion step t. Then, the sampling transition is done by replacing $p(x_t|x_{t+1})$ by $p(x_t|y, x_{t+1})$, i.e the score approximation becomes $s_\theta(x_t, t) + \lambda c_\phi(x_t, t, y)$, where λ is a scaling parameter that determines the influence of classifier guidance. This technique is not involved in the training of the diffusion model, and can be used as a *refinement* method for fine-tuning the diffusion model. Notably, it can present advantages in terms of computational resources, as a smaller model can be used as a classifier. We will be exploring another boosting method, namely discriminator guidance, as a main improvement framework for diffusion models in chapter 3.

Note : This method was further extended by [HS22] to *classifier-free* guidance, by simultaneously training a model on samples x and a context c. This is a boosting-free method as only the training procedure is affected, and no additional model is trained. Stable diffusion [Rom+22], among others, used this method during its training, which explains its ability to generate data from user defined prompts. Adding context information is thus a viable technique to improve the generative process.

Another family of methods consists in using statistical sampling algorithms to improve the likelihood of a sample. In the following, we will describe rejection algorithms for generative models, as they rely on density ratio estimation techniques that will be important for the following chapters, notably chapters 3 and 4. Rejection sampling [von51] is a method for sampling from a target distribution P using a proposal distribution Q and a density ratio $\frac{p(x)}{q(x)}$ between both distributions. In the context of generative models, the idea is to create a new distribution $\hat{P}$ from the estimated distribution $\tilde{P}$ by using accepting or rejecting samples from $\tilde{P}$ with a rejection probability $a(x)$. The resulting density is given by $\hat{p}_a(x) = \frac{\tilde{p}(x)a(x)}{Z}$, where Z is a normalization constant. Thus, the rejection rate is given by $\mathbb{E}_{\tilde{P}}[a(x)] = \frac{1}{Z}$ or in other terms, one needs to draw $\frac{1}{Z}$ samples in average from $\tilde{P}$ to have one sample accepted. In order to exactly match the target density $p(x)$, the literature has proposed an optimal rejection rate $a(x)$ of the form :

$$a_{\text{opt}}(x) = \frac{p(x)}{M\tilde{p(x)}} \tag{2.26}$$

Algorithm 5 Generic Rejection Sampling Algorithm

```
Generate $x \sim \tilde{P}$
Generate $u \sim \mathcal{U}(0,1)$
**while** $u \leq a(x)$ **do**                         ▷ Rejection criterium
    $x \sim \tilde{P}$
    $u \sim \mathcal{U}(0,1)$                          ▷ Draw a uniform random number
**end while**
```

where $M = \sup_{\mathcal{X}} \frac{p(x)}{\tilde{p(x)}}$ and the resulting density function of $\hat{P}_{a_{\text{opt}}}$ is equal to p. However, in a high dimensional $\mathcal{X}$, [Mac03] showed that M can be very high and thus set a high rejection rate. For example, in the context of BigGAN [BDS19] trained on the CelebA dataset [Liu+15] (178×218 images), the rejection rate is as big as 10^6. Thus, a line of work investigates alternative rejection probabilities $a(x)$ to perform rejection sampling in high dimensions.

In the context of GANs, [Aza+19] propose to involve the discriminator in the *generation* process by using it to estimate the density ratio involved in equation 2.26. A discriminator model d parameterized by a set of parameters ϕ within the framework of f-GANs ([NCT16],2.7) can be used to estimate the density ratio using the following expression [Ver+24a]:

$$\frac{p(x)}{\hat{p}(x)} \approx \nabla f^*(d_\phi(x)) = r(x) \tag{2.27}$$

where f^* denotes the *Fenchel conjugate* of function f. In order to lower the rejection rate, [Aza+19] propose to use the following acceptance probability function :

$$a_\gamma(x) = \frac{r(x)}{r(x)(1 - e^\gamma) + e^\gamma M} \tag{2.28}$$

Negative values of γ are used to lower the rejection rate, but no theoretical contributions are given for the choice of γ. Another line of work explores rejection sampling methods under a *budget*. That is, under budget K, solve the following optimization problem :

$$\begin{aligned} \min_a \quad & D_f(P \,\|\, \tilde{P}_a) \\ \text{s.t.} \quad & \begin{cases} \mathbb{E}_{\hat{P}}[a(x)] \geq \frac{1}{K}, \\ \forall x \in X, \quad 0 \leq a(x) \leq 1. \end{cases} \end{aligned}$$

[Ver+24b] derives an optimal solution for this problem, with a rejection probability of the form :

$$a_{\text{K-opt}}(x) = \min\left(\frac{p(x)c_K}{\tilde{p(x)}M}, 1\right) \tag{2.29}$$

where c_K is a constant such that $\mathbb{E}_{\tilde{P}}[a_{\text{K-opt}}(x)] = \frac{1}{K}$ The density ratio ratio is computed by a discriminator as in [Aza+19], and the constant c_K is computed using dichotomy.

2.4 Conclusion

Diffusion models emerge as a viable family of generative models, as they emanate from a nice theoretical framework. The generation process is two-folds : training a model s_θ to estimate the target score function at each noise level $tin[0, T]$, then use the estimated score to solve the backward SDE (Equation 2.19) or ODE (Equation 2.21). While this separation allows for separately improving each step of the process, it might represent a concatenation of two error sources : an approximation error related to the training process, and an estimation error related to the sampling process. Several diffusion models are currently state of the art in generation, and the main question of our work is how to improve these pre-trained models, without retraining them and without additional data. In subsection 2

This raises the following questions, that we will try to answer in chapter 2.3, we presented a few ideas to improve generative models, and booting methods emerged as a viable technique to do so. This raises the following question :

- **Can classifier guidance be generalized to a boosting method that does not rely on class information ?** This question is answered in chapter 3, where we introduce discriminator guidance [Kim+23] and improve the theoretical framework of this method.

- **Can we extract further information from the density ratio estimated by a discriminator ?** This question will be answered in chapter 4, where we introduce reinforcement learning methods that uses information from an estimated density ratio as a reward function.

Discriminator guidance 3

How to improve the generation process of a diffusion model without re-training it ? In this chapter, we will present discriminator guidance, a concept introduced by [Kim+23] and improved in the context of this master's thesis. Overall, this method consists in using information from a trained discriminator during the sampling process, to guide the denoising path towards the target distribution. This can be seen as a generalization of classifier guidance [DN21] to the case of the absence of class information, and can also linked to discriminator rejection [Aza+19] if applied to diffusion models. We will first start by presenting the framework proposed by [Kim+23] in section 3.1, The formulation of this framework is theoretically grounded, but the presented training scheme of the discriminator is not justified. We show in section 3.2 that it is sub-optimal, and can worsen the sampling process in the over-fitting regime. Thus, in section 3.3, we propose a novel training objective that is optimal and demonstrate our results for standard generation benchmarks.
In the following, we recall the notation used in our work :

- P_t, p_t denotes the diffused *target* distribution and its corresponding density at time t
- $s_\theta(x,t)$ denotes a **trained** score based model with fixed parameter set θ
- $\tilde{P}_t, \tilde{p}_t$ denotes the diffused *estimated* distribution and its corresponding density at time t obtained by solving *exactly* the backward SDE (2.19). We assume that $s_\theta(x,t) = \nabla_x \log \tilde{p}_t(x)$
- The trainable discriminator d_ϕ is parameterized by a set of parameters ϕ
- We denote by $\pi(x)$ the density of the prior distribution used in the sampling process.
- The *refined* distribution induced by using both the score network s_θ and a discriminator d_ϕ is denoted by $\hat{P}_t$ and its corresponding density by $\hat{p}_t$

3.1 Discriminator guidance

3.1.1 Derivation

Discriminator guidance stems from the simple concept of estimating the training error $\nabla \log p_t(x) - s_\theta(x,t)$. In the following, we provide a definition that would introduce a key assumption in the derivation of discriminator guidance.

Definition 3. *The evidence lower bound $\mathcal{L}_\theta$ of a score model s_θ is given by :*

$$\mathcal{L}(s_\theta) = \frac{1}{2}\int_0^T \lambda(t)\mathbb{E}_P\left[||\nabla \log p_t(x) - s_\theta(x,t)||^2\right]dt \tag{3.1}$$

In the following, we make the following assumption :

Assumption 2. *The distribution $\tilde{P}_t$ induced by the backward process with score estimator s_θ has a log likelihood equal to the evidence lower bound of s_θ*

Define S_{sol} to be a family of time-conditioned score network s_θ that satisfies the following: there exists densities $p_{\theta,0}$ and $p_{\theta,t}$ such that$s_\theta(x,t) = \nabla logp_{\theta,t}(x)$ almost everywhere, where $p_{\theta,t}$ is the marginal density at time t of $dx = f(x,t)dt + g(t)dw$, starting from $x_0 \sim p_{\theta,0}$

Theorem 1. *[Kim+23] Suppose that the score network s_θ satisfies assumption 2. Then $s_\theta \in S_{\text{sol}}$. We thus denote by $\tilde{P}_t$ the distribution with density $\tilde{p}_t(x)$ such that $s_\theta(x,t) = \nabla \log \tilde{p}_t(x)$*

Theorem 2. *[Kim+23] Suppose that :*

- *Assumption 2 is satisfied by the estimated score network s_θ.*
- $s_\theta(x,T) = \nabla \log \pi(x)$.

Then, by noting $c_\theta(x,t) = \nabla \log p_t(x) - s_\theta(x,t) = \nabla \log \frac{p_t(x)}{s_\theta(x,t)}$, the backward SDE defined by :

$$dx = f(x,t) - \frac{1}{2}g(t)^2(s_\theta(x,t) + c_\theta(x,t))dt \tag{3.2}$$

coincides with the backward SDE of the diffusion process induced by the target distribution (Equation 2.19).

The proofs for theorems 1,2 are relegated to Appendix 5.
Thus, if one estimates exactly the density ratio $r_{\text{opt}}(x,t) = \frac{p_t(x)}{\tilde{p}_t(x)}$, then the *refined* score $s_{\theta,\text{opt}}(x,t) = s_\theta(x,t) + \nabla \log r_{\text{opt}}(x,t)$ allows for sampling from the target distribution

by solving the backward SDE. [Kim+23] train a discriminator $d_\phi(x,t)$ to estimate $r_{\text{opt}}(x,t)$ then computes $\nabla \log d_\phi$ as an estimator for c_θ. This raises the following question : **Does a generic estimation of the density ratio make the generative process better ?**

3.1.2 f-divergence between the target and refined distribution

The following theorem follows from the conservativity of s_θ (Assumption 3) :

Theorem 3. *Suppose that the assumptions of theorem 2 are satisfied. The backward SDE using as a score estimator $s_{\theta,\phi}(x,t) = s_\theta(x,t) + \nabla \log d_\phi(x,t)$ induces a distribution $\hat{P}_t$ with density $\hat{p}_t(x)$*

[Kim+22] provide an interesting result that upper bounds the Kullback-Leibler divergence between the target and refined distributions.

Theorem 4. *[Kim+23] If the assumptions of theorem 2 hold, then given the target distribution P_t, the score distribution $\tilde{P}_t$ and the refined distribution $\hat{P}_t$ we have :*

$$D_{KL}(p_0 \| \tilde{p}_0) = D_{KL}(p_T \| \pi) + E_\theta,$$
$$D_{KL}(p_0 \| \hat{p}_0) \le D_{KL}(p_T \| \pi) + E_\phi$$

where E_θ is the score error

$$E_\theta = \frac{1}{2} \int_0^T g^2(t) \mathbb{E}_{p_t} \left[\| \nabla \log p_t(x) - s_\theta(x,t) \|^2 \right] dt,$$

and E_ϕ is the discriminator-adjusted score error

$$E_\phi = \frac{1}{2} \int_0^T g^2(t) \mathbb{E}_{p_t} \left[\| \nabla \log r_{\text{opt}} - \nabla \log d_\phi \|^2 \right] dt.$$

Leading to the following inequality :

$$D_{KL}(p_0, \hat{p}_0) \le D_{KL}(p_0, \tilde{p}_0) + (E_\theta - E_\phi) \tag{3.3}$$

[Kim+23] provides no theoretical analysis on the positivity of the difference $(E_\theta - E_\phi)$, but argue that in practice it is initialized near 0 and gradually increases throughout training. Table 3.1 shows the value of the difference with different initializations of the discriminator. Notably, a completely blind discriminator $d_\phi(x,t) = \frac{1}{2}$ has $E_\phi = E_\theta$, and an optimal discriminator has zero error. There was however no analysis in the worst case scenario, where a discriminator is *badly* trained. In section 3.2, we present situations that are possible in practice that could make the generation process worse.

Tab. 3.1: Discriminator-adjusted score error $E_{\infty,\phi}$ and corresponding refinement value [Kim+23].

Discriminator	$E_{\theta,\phi}$	$E_\theta - E_{\theta,\phi}$
Blind $d_\phi(t = 0.5)$	E_θ	0
Optimal d_ϕ	0	E_θ (Maximum)
Untrained $d_\phi(\approx 0.5)$	$\approx E_\theta$	≈ 0
Trained d_ϕ	$\ll E_\theta$	$\gg E_\theta$

3.1.3 Training the discriminator

In order to train the discriminator, [Kim+23] propose the same training paradigm as in GAN training. Specifically, a discriminator is trained to minimize the binary cross-entropy loss, as suggested in the seminal GAN paper by [Goo+14] :

$$\mathcal{J}_{CE}(\phi) = \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[-\log d_\phi(x,t)\right] + \mathbb{E}_{\tilde{P}_t}\left[-\log(1 - d_\phi(x,t))\right] dt \tag{3.4}$$

and the estimated density ratio is given by $r_\phi(x,t) = \frac{d_\phi(x,t)}{1-d_\phi(x,t)}$. The corresponding correction term is thus given by $c_\phi(x,t) = \nabla \log r_\phi(x,t)$. Training is done by generating samples from the estimated score network to estimate the second expectation in equation 3.4, and samples from the dataset are used to estimate the first expectation in equation 3.4. The main advantage this loss is providing a relatively fast training time for the discriminator. However, no theoretical analysis is provided regarding its ability to approximate $c_\phi(x,t)$, that is $\nabla \log r_{\text{opt}}(x,t)$. Minimizing the cross entropy loss provides indeed a good estimate for the density ratio, but not necessarily for the gradient of the log density ratio.

We thus raise the following question, that we answer in section 3.2 : **Even if the cross-entropy is minimized, is the discriminator guidance refining the diffusion model?**

3.2 Sub-optimality of cross entropy minimization

In this section, we first show in theorem 5 that an arbitrarily low cross-entropy loss can lead to an arbitrarily high KL-divergence between the target and refined distribution. We then show in 6 that over-fitting the discriminator to minimize the cross-entropy leads to a high KL divergence between P and $\hat{P}$.

Theorem 5. *Let $\{\boldsymbol{x}(t)\}_{t\in[0,T]}$ be a diffusion process defined by Equation 2.18. Assume that $\nabla \log \widetilde{p}_t = \boldsymbol{s}_\theta$ and $\nabla \log \widehat{p}_t = \boldsymbol{s}_\theta + \nabla \log r_\phi$ and that the induced distribution P, $\widetilde{P}$, and $\widehat{P}$ satisfies the assumptions detailed in Appendix 5.0.1. Then, for every $\varepsilon > 0$ and for every $\delta > 0$, there exists a discriminator $d : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ trained to minimize the cross-entropy such that:*

$$\mathcal{L}^d_{\mathrm{CE}}(\phi) \leq \varepsilon \quad \textit{and} \quad \mathcal{D}_{\mathrm{KL}}(P \| \widetilde{P}) \geq \delta, \tag{3.5}$$

where $\widehat{P}$ is the distribution induced by discriminator guidance with d.

Sketch of Proof. The detailed proof for Theorem 5 is provided in Appendix 5.0.2. The main idea for the proof is to show that a learned discriminator d_ϕ oscillating around the true optimal discriminator d^* will show a low cross-entropy. The smaller the amplitude of the oscillation, the lower the cross-entropy. However, the faster the oscillations, the larger the Kullback-Leibler divergence will be. □

Although theorem 5 shows the sub-optimality of the cross-entropy, it presents a pathologic case that one cannot find in practice. In the following theorem, we consider the situation where the discriminator overfits the training data, and show that the KL divergence between the target and refined distribution can explode.

Theorem 6. *Let P and $\widetilde{P}$ be two distributions on $\mathbb{R}$. Assume the intersection of their support is not empty, and assume they admit density functions p and $\tilde{p}$ which are both L-Lipschitz. Let $x_1 \ldots x_N \sim P^N$ and $x'_1 \ldots x'_N \sim \tilde{P}^N$. Assume that the or cross-entropy loss $-\log \sigma\left(d_\varphi(x_i)\right)$ and $-\log\left(1 - \sigma\left(d_\phi(x'_i)\right)\right)$ is at most ϵ on each example x_i and x'_i. Then, the expectation over the dataset of the mean square error exhibits the following behavior, for some constant c:*

$$\lim_{N\to\infty} \frac{\mathbb{E}\left[E_\phi\right]}{N} \to c. \tag{3.6}$$

In other words, the difference $E_\theta - E_\phi$ decreases linearly with the number of samples. This shows that the upper bound on the KL divergence between the target and refined distribution goes to $-\infty$ in an over-fitting regime.

Sketch of Proof. The Theorem is proved in Appendix 5.0.4. While we assume a one-dimensional diffusion process for the sake of simplicity, this result could be applied to higher dimensional. The key argument is to show that if the discriminator is overfitting the cross-entropy, the values of d_ϕ on real and fake samples will be very different. However, the closer the distributions P and $\widehat{P}$, the higher the probability of sampling a fake sample that is close to a real sample. Therefore, the discriminator gradient will be high between the two samples independently of the true gradient value. □

Intuitively, these results are due to the fact that [Kim+23] provide a good estimation for the density ratio $\frac{p_t(x)}{\hat{p}_t(x)}$, but not for the gradient of the log density ratio. This raises the following questions:

- **How can we train the discriminator to provide a good estimate for the gradient of the log density ratio ?**

- **Instead of training a model to estimate the density ratio, can we train a model to directly estimate the gradient of the log density ratio ?**

3.3 Improving the refinement process

In this section, we answer the first aforementioned questions by deriving an optimal loss for the refinement problem. Similarly to [Vin11; Son+21], we propose a denoising score matching loss to estimate $\nabla \log r_{\text{opt}}(x,t)$, and prove that minimizing our proposed loss minimizes the KL divergence between the target and refined distribution. We furthermore provide a few experiments to demonstrate the effectiveness of our method on image generation benchmarks.

3.3.1 Denoising score matching loss

In order to focus on estimating the gradient of the log-density ratio, we propose to minimize the following score matching loss :

$$\mathcal{J}_{SM}(\phi) = \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[||\nabla \log p_t(x) - s_\theta(x,t) - \nabla d_\phi(x,t)||^2\right] dt \tag{3.7}$$

However, the density $p_t(x)$ is intractable in practice, thus we propose the following *MSE* loss, similar to the proposed loss of [Vin11; Son+21] :

$$\mathcal{J}_{MSE}(\phi) = \int_0^T \lambda(t)\mathbb{E}_t\left[\mathbb{E}_{x_0\sim P_0}\left[\mathbb{E}_{x_t\sim P_{t|x_0}}\left[||\log p_t(x_t|x_0) - s_\theta(x,t) + \nabla d_\phi(x,t)||^2\right]\right]\right] dt \tag{3.8}$$

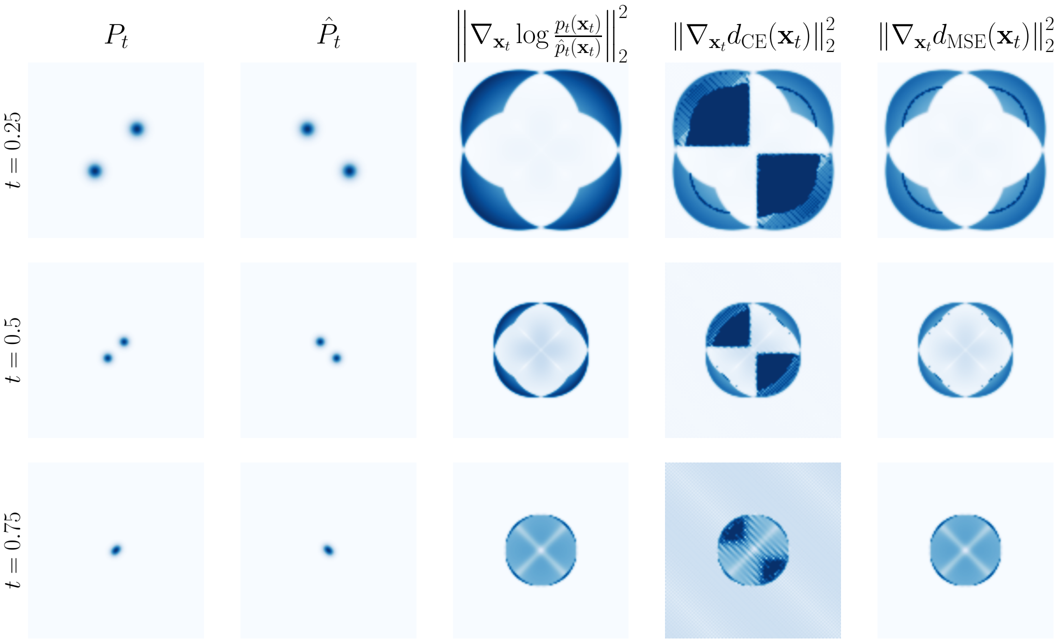


Fig. 3.1: Visualizing the estimation of $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) / \nabla_{\boldsymbol{x}_t} \log \hat{p}_t(\boldsymbol{x}_t)$ for the discriminator trained with low cross-entropy or low MSE loss. We plot the norm of the gradient for better readability. We can see here that the discriminator trained on minimizing $\mathcal{J}_{CE}$ provides a poor estimate of the gradient, while the one trained on $\mathcal{J}_{MSE}$ provides a sound estimate.

The following proposition states that miniming $\mathcal{J}_{MSE}$ is equivalent to minimizing $mathcal J_{SM}$.

Proposition 1. *Assume that P and $\widetilde{P}$ satisfy the assumptions detailed in Appendix 5.0.1. Then, the following holds:*

$$\operatorname*{argmin}_{\phi} \mathcal{J}_{\mathrm{SM}}(\phi) = \operatorname*{argmin}_{\phi} \mathcal{J}_{\mathrm{MSE}}(\phi). \tag{3.9}$$

Sketch of Proof. The proof is provided in Appendix 5.0.5, and is based on the fact that the two losses are equal up to an additive constant. □

3.3.2 Practical considerations

One advantage proposed by the approach of [Kim+23] is the usage of generated samples from $\hat{P}$ at each training step. Training a discriminator to minimize the MSE loss is more prone to overfitting, as no new samples are generated during training, as opposed to the cross-entropy loss. Furthermore, as our work focused on improving generation for already well trained model, the error $\nabla \log \frac{p_t(x)}{\tilde{p}_t(x)}$ is small. We thus

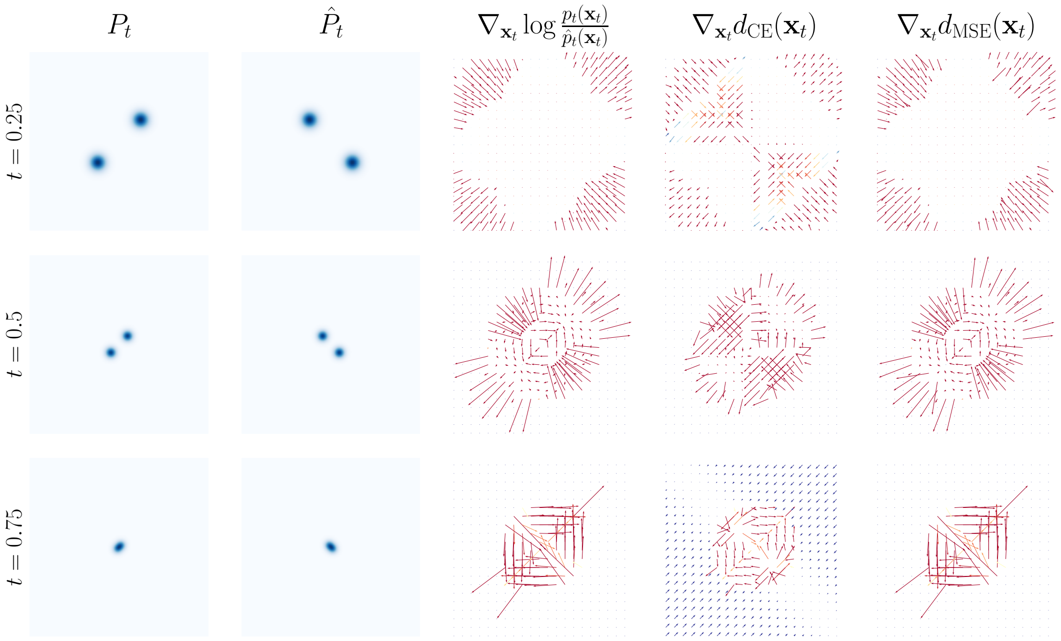


Fig. 3.2: Gradients of the estimated density ratio. d_{CE} represents a discriminator with low cross entropy, and d_{MSE} represents a discriminator with low MSE

propose to use leverage the generative abilities of the pretrained diffusion model s_θ to gain more information, by adding $\mathcal{J}_{CE}$ as a regularization term:

$$\mathcal{J}_{\mathrm{train}}(\phi) = \mathcal{J}_{\mathrm{MSE}}(\phi) + \gamma \mathcal{J}_{\mathrm{CE}}(\phi) \tag{3.10}$$

where γ is a hyperparameter controlling the regularization strength.

3.4 Experiments

3.4.1 Toy example

Setting: We consider two distinct mixtures of Gaussians in $\mathbb{R}^2$, that represent respectively P and $\tilde{P}$. We compute for a subset of timesteps $t \in [0,1]$ the closed form of the scores $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ and $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, and we use them for performing a discretized backward diffusion process. In order to concentrate on the effect of discriminator guidance, we first sample points from P, perform a forward diffusion process using subVP-SDE [Son+21], and start from this set of points as a prior distribution.

Evaluation of the optimization procedure : We also compute two estimations of the discriminator. The first estimate d_{CE} is based solely on minimizing $\mathcal{J}_{\mathrm{CE}}$ and the second d_{MSE} ensures a small $\mathcal{J}_{\mathrm{MSE}}$. We plot in Figure 3.1 the gradient estimates across the denoising process and show that the discriminator estimated by minimizing the cross-entropy has a bad gradient estimate. Figure 3.3 shows that the failure to accurately estimate the gradient is detrimental to the denoising process. We also provide in Appendix 3.2 the gradient plots of the different estimates.

3.4.2 Image generation

We implement our approach using the pre-trained EDM model [**karras**]. We will compare our approach with the pre-trained models (EDM) and the pre-trained models with discriminator guidance (EDM+DG). For a fair comparison to [Kim+23], we build the same architecture for the discriminator d_ϕ using pre-trained embedding of the ADM classifier models used in the work of Dhariwal and Nichol [DN21]. However, our method requires using the same framework as EDM, therefore we reimplemented DG for various datasets and resolutions. We use the same hyperpa-

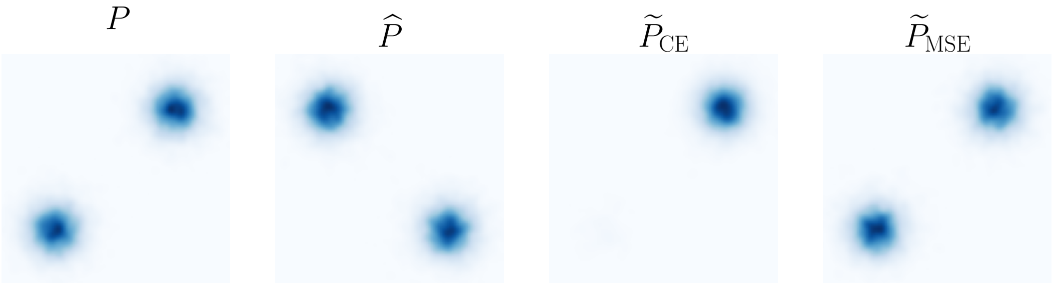


Fig. 3.3: Comparison of the sampling process by using a discriminator to minimize the cross-entropy loss defining a distribution $\tilde{P}_{\mathrm{CE}}$ and a discriminator to minimize the MSE loss $\mathcal{J}_{\mathrm{SM}}(\phi)$ defining a distribution $\tilde{P}_{\mathrm{MSE}}$. We can see that the cross entropy loss misses one mode of the target distribution

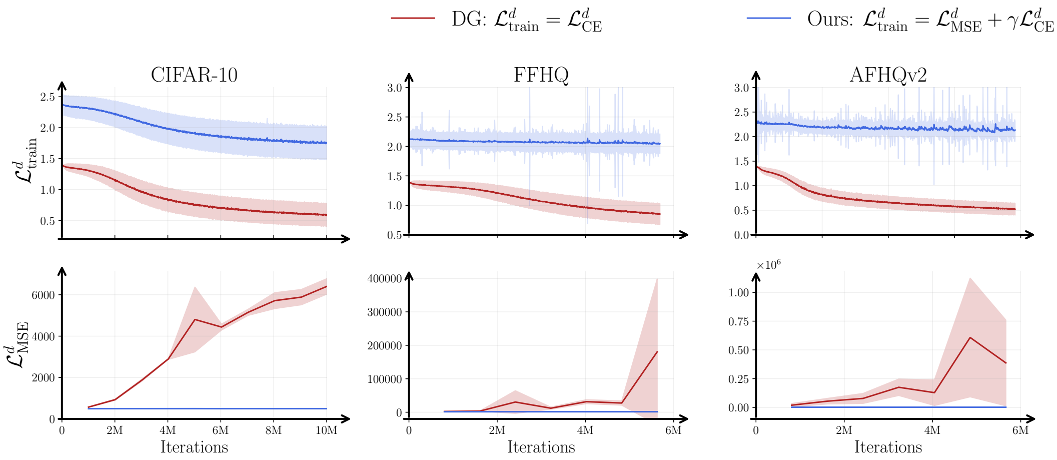


Fig. 3.4: Comparison of the proposed method with the method of Kim et al. [Kim+23] on the EDM model [**karras**] on CIFAR-10, FFHQ and AFHQ-v2 datasets. For the method proposed by Kim et al. [Kim+23], denoted as DG, the MSE loss $\mathcal{J}_{\mathrm{MSE}}$ increasing during training. The proposed method, denoted as Ours, shows a stable behavior. The error bars represent the standard deviation over 3 runs.

Tab. 3.2: Comparison of the proposed method with the method of Kim et al. [Kim+23] on CIFAR-10, FFHQ and AFHQv2. We give metrics averaged on 3 trainings procedure with one standard deviation.

Dataset	Resolution	Method	FID	P	R
CIFAR-10	32 × 32	EDM	2.03	99.12	76.84
		EDM + DG	3.03 ± 0.05	99.06 ± 0.10	72.56 ± 0.14
		EDM + Ours	2.00 ± 0.01	99.18 ± 0.04	76.70 ± 0.12
AFHQv2	64 × 64	EDM	2.62	99.92	84.66
		EDM + DG	2.66 ± 0.02	99.92 ± 0.00	83.70 ± 0.22
		EDM + Ours	2.61 ± 0.00	99.92 ± 0.00	84.62 ± 0.01
FFHQ	64 × 64	EDM	2.54	99.75	78.85
		EDM + DG	2.63 ± 0.03	99.74 ± 0.02	77.55 ± 0.06
		EDM + Ours	2.54 ± 0.00	99.75 ± 0.00	78.87 ± 0.02

rameters as Kim et al. [Kim+23] for the discriminator guidance method, and we apply both methods on the CIFAR-10, FFHQ and AFHQ-v2 datasets on, respectively, 8x GPU-V100, 8x GPU-A100 and 8x GPU-A100. First, we illustrate the behavior of the training loss ($\mathcal{J}_{\text{CE}}$ for DG and $\mathcal{J}_{\text{train}}$ for Ours) and the reconstruction loss during training in Figure 3.4. We can see that our approach enables us to mitigate the increase of the MSE loss, while the DG method shows a clear increase of the loss. However, we observe a decrease in the training loss that demonstrates that the density ratio is learned.

We report the FID[1] [Heu+17], Precision and Recall [Kyn+19] scores in Table 3.2. Metrics are computed using the Guided Diffusion library [DN21] using 50k samples for CIFAR-10 and FFHQ, and 15k samples for AFHQ-v2. We use $k = 5$ to compute the Precision and the Recall. We can see that our method systematically outperforms the DG method on all datasets even if the difference with the EDM model is small.

3.4.3 Comments on the results

Training a discriminator to refine the EDM model [**karras**] slightly improved the generation metrics, but the difference is not significant. We hypothesize that this is due to the fact that the EDM model is already well trained, and the discriminator guidance method is not able to provide a good estimate of the gradient of the log-density ratio. A more notable contribution would be to try this method on less trained models, and evaluate the improvement in FID, precision and recall. Thus, as future directions of this part, we propose the following :

[1] Note that we can observe some discrepancies between the results in [Kim+23] due to the reimplementation of the framework to closely match the generative process of EDM

- Our work also lacks the diversification of architectures, and should have been tried with other pre-trained models, for tasks different than image generation.

- Fine-tune the hyperparameter γ, or investigate other regularization terms to improve the training process.

- Restrict discriminator guidance to timesteps t where the score estimation is poor.

- Investigate the effect of discriminator guidance on less trained models.

3.5 Conclusion

In this chapter, we presented a method for refining pre-trained diffusion models using a smaller discriminator network. After introducing the method proposed by [Kim+23], we provided theoretical results that lead to modifying the training objective of the discriminator, and prove that our proposed objective is optimal. We then provided experimental results to evaluate the validity of our approach, and showed that it was not effective for the EDM model. We provided future directions to improve the method. Discriminator guidance attempts to correct the *approximation* errors related to the score estimation while training. Now, the following chapter answers the following question : **Can the sampling erros, related to the discretization of the backward process, be corrected by a boosting method ?**

4 Reinforcement Learning based methods

The idea of refining the sampling process of a diffusion model consists in correcting the sampling error related to the discretization of the backward SDE 2.19 or ODE. In section 2.3, we introduced *rejection* sampling as a method to correct the sampling error. This method relies on using a trained discriminator to reject samples that are not close to the true distribution. However, in high dimensions, the optimal acceptance probability of a sample becomes very small, which makes the rejection sampling method inefficient. We also would like to propose a method that is tailored for diffusion models, using the fact that the score network approximates the target distribution in different noise levels $t \in [0, T]$. In the first section, we introduce restart sampling [Xu+23] as a *boosting free* method to correct the sampling process. The method overall consists in improving the sampling process by solving the backward ODE 2.19, renoising, and restarting the process for K times. K in this case is a hyperparameter that is manually tuned, thus raising the question of finding an optimal K for a pre-trained diffusion model. We propose to use Reinforcement Learning (RL) to instead dynamically noise / denoise the samples. We thus present briefly reinforcement learning in the second section, and formalize the problem for diffusion models in the third section. Experiments are currently ongoing for the end of the internship.

4.1 Restart sampling

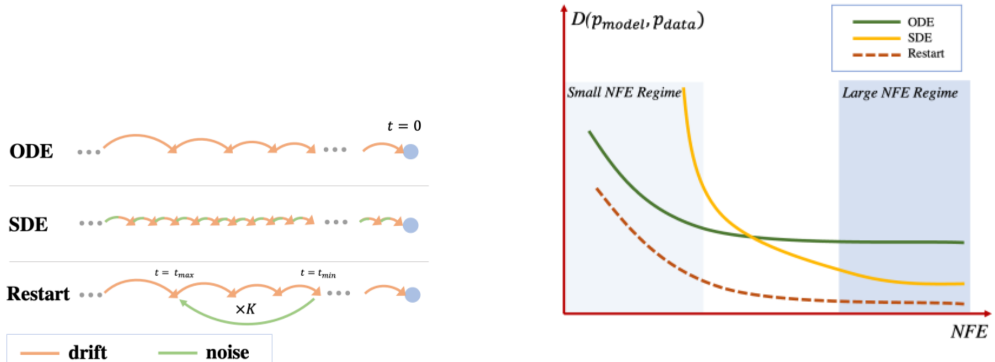
4.1.1 Trade-off between speed and quality

In order to sample from a diffusion model, the standard approach is to solve the backward ODE (Equation 2.21) / SDE (Equation 2.19) using numerical solvers, such as Heun's 2nd method and Euler-Maruyama's method [KP92]. The choice between reversing the ODE and the SDE is a trade-off between speed and quality :

- Sampling speed is evaluated for diffusion number by the number of function evaluations (NFE). That is, the number of calls to the score model s_θ during the sampling process. (Fast means low NFEs)

- Quality is commonly assessed by metrics depending on the data type. For image generation, the Fréchet Inception Distance (FID) [**heusel2017gans**] is a common metric. (High quality means low FID)

As the sampling speed increases (low NFE), the sample quality generally deteriorates (high FID). This is explained by the discretization error caused by using a larger step size in numerical differential equation solvers. As illustrated in figure 4.1, ODE solvers are efficient in the small NFE regime, providing a decent quality with a large step size. They are outperformed by SDE solvers for smaller step sizes (high NFE) in terms of quality, and fail to improve when increasing the NFE.



(a) Restart sampling method

(b) Number of function evaluations (NFE) vs sampling quality

Fig. 4.1: Taken from [Xu+23]. **4.1a** Illustration of restart sampling. **4.1b** Sample quality vs NFE for different numerical solvers. ODE solvers (green) allow for a better quality than SDE solvers (yellow) in the small NFE regime, but plateau quickly in the large NFE regime.

4.1.2 Sampling with ODE solvers vs SDE solvers

In the small NFE regime, the effectiveness of ODEs can be explained by comparing the local error of the two methods. For a step size δ, the Euler method for solving ODEs yields a local error of $O(\delta)^2$, while the Euler-Maruyama method for solving SDEs yields a local error of $O(\delta^{\frac{3}{2}})$ [DK19]. In the large NFE regime, the step size δ becomes small thus the discrepancy between the two local errors becomes negligible. The main source of error thus becomes the *approximation* error of the score model s_θ. Intuitively, adding noise by solving the SDE corrects the approximation error. The absence of a noise term in the ODE solvers thus can be explained by the domination of the approximation error over the discretization error, that cannot be corrected via noise. [Xu+23] provide a theorem that formalizes the behaviour difference between the two methods :

Theorem 7. *[Xu+23] Let $[t_{\min}, t_{\max}] \subset [0, T]$, and let $\tilde{p}_t^{ODE}, \tilde{p}_t^{SDE}$ denote the distributions of simulating the ODE and SDE respectively, with the estimated score s_θ. Assume for a given B that $\forall t \in [t_{\min}, t_{\max}], \|x_t\| \le \frac{B}{2}$ for any x_t in the support of $P_t, \tilde{P}_t^{ODE}, \tilde{P}_t^{SDE}$. Then we have the following upper bounds :*

$$W_1(\tilde{p}^{ODE}_{t_{\min}}, p_{t_{\min}}) \le B \cdot TV(\tilde{p}^{ODE}_{t_{\max}}, p_{t_{\max}}) + O(\delta + \varepsilon_{\text{approx}})(t_{\max} - t_{\min})$$
$$W_1(\tilde{p}^{SDE}_{t_{\min}}, p_{t_{\min}}) \le \underbrace{(1 - \lambda e^{-U})}_{\text{contractionterm}} \cdot B \cdot TV(\tilde{p}^{SDE}_{t_{\max}}, p_{t_{\max}}) + O(\sqrt{\delta t_{\max}} + \varepsilon_{\text{approx}})(t_{\max} - t_{\min})$$

where $W_1(.,.)$ denotes the 1-Wasserstein distance, $TV(.,.)$ denotes the total variation distance, $U = \frac{BL_1}{t_{\min}} + \frac{L_1^2 t_{\max}^2}{t_{\min}^2}$, $\lambda < 1$ is a contraction factor, L_1 and $\varepsilon_{\text{approx}}$ are uniform bounds on $\|t s_\theta(x, t)\|$ and the approximation error $\|t s_\theta(x, t) - t \nabla_x \log p_t(x)\|$ respectively.

Both right hand sides of the inequality feature two distinct terms in a sum. The first term denotes the *contracted* error, i.e the initial errors accumulated from both approximation and discretization errors during the simulation of the backward process, up until time $t_{\max}$. The second term denotes the *additional sampling* error. This term is essentially the same for both inequalities as δ gets small, but the contraction error is lowered by the term U as the denoising process progresses, explaining the role of the noise in correcting the approximation error. This raises the following question : **Is there a sampling procedure that maintains the same discretization error as ODE solvers, while correcting the approximation error as SDE solvers do ?**

4.1.3 Solving back and forth

[**xu_2023_restart**] propose *restart sampling*, a sampling technique consisting in doing back and forth steps during the sampling process during a pre-defined time interval $[t_{\min}, t_{\max}] \subset [0, T]$. As depicted in figure 4.1a, a restart backward step consists in performing an ODE step from t to $t - 1$. A restart forward step consists in adding noise following the diffusion forward process (Equation 2.18) from time $t_{\min}$ to time $t_{\max}$. This process is repeated K times, and the final sample is the output of the last ODE backward step, from $t_{\min}$ to 0. The authors provide the following upper bound on the 1-Wasserstein distance between the *refined* distribution $TildeP_{t_{\min}}^{\text{restart}}$ and the target distribution $P_{t_{\min}}$:

$$W_1(\tilde{P}_{t_{\min}}^{\text{restart}}, P_{t_{\min}}) \leq B \cdot (1-\lambda)^K TV(\tilde{P}_{t_{\max}}^{\text{restart}}, P_{t_{\max}}) + (K+1) O(\delta + \varepsilon_{\text{approx}})(t_{\max} - t_{\min}) \tag{4.1}$$

This inequality combines both advantages of ODE solvers (lower second term) and SDE solvers (approximation error contraction, exponential in K). However, this method introduces two additional degrees of freedom : the choice of the value of K and the restart interval $[t_{\min}, t_{\max}]$. This raises the following question : **Is it possible to dynamically noise / denoise the samples, without the need to manually tune K ?** We explore the option of using reinforcement learning as method to dynamically noise / denoise the samples, and present the formalization of the problem in the next section.

4.2 Reinforcement Learning

4.2.1 Background

Reinforcement learning is a machine learning paradigm that stems from control theory. It essentially attempts to learn an optimal *policy*, which is a probability distribution over actions given an observed state. It is described by a markov decision process (MDP) tuple (S, A, T, R, γ), where :

- S is the set of states of the environment, that can either be continuous or discrete
- A is the set of actions that the agent can take
- T is the transition model such that $T(s'|s,a) = P(S_{t+1} = s'|S_t = s, A_t = a)$
- R is the reward function that gives the reward r after taking action a in state s
- γ is the discount factor

A reinforcement learning *agent* interacts with the environment during episodes. Given $s_0 \sim \mu_0$, where μ_0 is the distribution over initial states, the agent takes an action a_0 according to its policy π_ψ, and receives a reward r_0 and a new state s_1.

The process is repeated until a terminal state is reached. The goal of the agent is to learn the optimal policy π^* that maximizes the expected return :

$$\mathcal{J}_{\mathrm{RL}}(\psi) = \mathbb{E}_{\pi_\psi}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{4.2}$$

We denote by $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_L)$ the trajectory of the agent by running an episode simulation in the environment. The field of RL has gained a lot of attention in the past decade, with the advent of deep reinforcement learning involving deep neural networks [Mni+13]. It has been applied to several tasks, notably robotics [Gup+19], protein synthesis [Jum+21], and famously for allowing the safe and aligned deployment of large language models through Reinforcement Learning from Human Feedback (RLHF) [Ouy+22]. Several methods were developed to solve reinforcement learning, such as Q-learning, policy gradient methods, and actor-critic methods. Our work focuses on policy-gradient methods, and more specifically on the Proximal Policy Optimization (PPO) algorithm [**schulman2017proximal**]. We present in the following section a modified version of the PPO algorithm that relies on $f-$divergences.

4.2.2 Matching state visitation distributions

The distribution over trajectories induced by the policy π_ψ can be defined by the density :

$$p_\psi(\tau) = \mu_0(s_0) \prod_{t=0}^{L-1} \pi_\psi(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{4.3}$$

Thus, the distribution over states induced by the policy π_ψ can be defined by the density :

$$\begin{align} p_\psi(s) &= \frac{\int p_\psi(\tau)\eta_\tau(s)d\tau}{Z} \tag{4.4} \\ &= \frac{\int \mu_0(s_0) \prod_{t=0}^{L-1} \pi_\psi(a_t|s_t) p(s_{t+1}|s_t, a_t)\eta_\tau(s)}{\int\int \mu_0(s_0) \prod_{t=0}^{L-1} \pi_\psi(a_t|s_t) p(s_{t+1}|s_t, a_t)\eta_\tau(s) d\tau ds} d\tau \tag{4.5} \end{align}$$

with $\eta_\tau(s) = \sum_{s_t \in \tau} \delta(s - s_t)$ the number of occurences of state s in trajectory τ. As a result, the optimal policy similarly defines a disribution over state visitation, that we write as P^* with density $p^*(s)$. This allows to formulate the following objective function :

$$\mathcal{J}_{\mathrm{RL}}(\psi) = D_f(p_\theta, p^*) \tag{4.6}$$

4.2.3 f-Policy gradient

As explained in section 4.3, this formulation allows to find an optimal policy without access to the reward function. This is crucial in our case since in the context of refining diffusion models, the only information we have are samples from the target distribution P_t and the estimated score s_θ. Computing the f−divergence in equation 4.6 is intractable, but [Aga+23] derive an expression for its gradient, in a similar flavor to the policy gradient theorem [Sut+99] :

$$\nabla \mathcal{J}_{\mathrm{RL}}(\psi) = \mathbb{E}_{P_\psi}\left[\left(\sum_{t=1}^{L} \nabla_\psi \log \pi_\psi(a_t|s_t)\right)\left(\sum_{t=1}^{L} f'\left(\frac{p_\psi(s_t)}{p_*(s_t)}\right)\right)\right] \tag{4.7}$$

In practice, computing this gradient is computationally intensive since it requires performing *on policy* updates. That is, the policy used to interact with the environment is updated on the go. This is not feasible for complex environments, as learning quickly becomes sample inefficient. In order to enforce off-policy learning, one idea [Aga+23] was to use importance sampling from another policy $\pi_{\psi'}$ to estimate the gradient. In order to provide a good estimation, the current policy π_ψ and the estimating policy $\pi_{\psi'}$ should be close. This is enforced by the Proximal Policy Optimization (PPO) algorithm [**schulman2017proximal**], that constrains the policy update to a certain distance from the previous policy. This is done by clipping the ratio of the new policy to the old policy, and using the clipped ratio to compute the loss function. This leads to the following expression for the gradient of $\mathcal{J}_{\mathrm{RL}}(\psi)$:

$$\nabla \mathcal{J}_{\mathrm{RL}}(\psi) = \mathbb{E}_{(s_t,a_t)\sim P_{\psi'}}\left[\min\left(r_\psi(s_t,a_t)F_{\psi'}(s_t), \mathrm{clip}\left(r_\psi(s_t,a_t), 1-\varepsilon, 1+\varepsilon\right)F_{\psi'}(s_t)\right)\right] \tag{4.8}$$

where $r_\psi(s_t,a_t) = \frac{\nabla_\psi \pi_\psi(a_t|s_t)}{\pi_{\psi'}(a_t|s_t)}$ is the importance sampling ratio, $F_{\psi'}(s_t) = \sum_{i=t}^{T} \gamma^i f'\left(\frac{p_{\psi'}(s_i)}{p_*(s_i)}\right)$, and ε is a hyperparameter that controls the clipping of the ratio. Algorithm 6 describes the generic training procedure for a given environment, and samples from the expert policy. In the following section, we will devise a procedure to apply the f-PG algorithm to the problem of refining diffusion models.

4.3 Application to diffusion models

4.3.1 Formalizing the restart sampling problem as a reinforcement learning problem

The sampling process of diffusion models can be seen as a trajectory in the state space of a reinforcement learning agent. We train an agent to decide whether an

Algorithm 6 f-PG

```
Let $\pi_\psi$ be the policy, $B$ be a buffer, $\mathcal{S}$ the set of trajectories from the expert policy
for $i = 1$ to num_iter do
    $B \leftarrow []$
    for $j = 1$ to num_traj_per_iter do
        Simulate trajectory $\tau$ using $\pi'_\psi$
        Train a discriminator $d_\phi$ to estimate the density ratio $\frac{p_{\psi'}(s)}{p_*(s)}$
        Store $f'\left(d_\phi(s)\right)$ for each $s$ in $\tau$
        $B \leftarrow B + \{\tau\}$
    end for
    for $j = 1$ to num_policy_updates do
        $\theta \leftarrow \theta - \alpha \nabla_\psi J_{\mathrm{RL}}(\psi)$
    end for
end for
```

ODE denoising step can be done, or if noise should be added to the current sample. The goal of the agent is to minimize the f−divergence between the state visitation distribution of the current policy and the optimal policy:

- The state space can be defined as the set of samples x_t of the discretized denoising process. Note that the timestep t refers to the noise level in the diffusion model. To make a difference with the time variable in the reinforcement learning context, we will refer to the RL timestep as l.

- The action space is discretized in two actions : perform one ODE denoising step to go from x_t to x_{t-1}, or perform add noise to go from x_t to x_{t+1}.

- The trajectory length is a fixed number of steps $L \geq T$

- The reward function is the f−divergence between the state visitation distribution of the current policy and the optimal policy.

- The samples from the expert (optimal) policy are the samples from the target distribution P_t. Each sample x_t is obtained from samples x_0 from the initial distribution P_0 (dataset), diffused according to the forward diffusion equation (Equation 2.18) up to time t.

Up to the current stage of the thesis, this approach is exploratory and we lacked time to provide theoretical guarantees on the validity of this method. The following section describes the training procedure, that we term as *f-Restart sampling*.

4.3.2 f-Restart sampling

We train a neural network that represents the policy π_ψ to output a probability distribution over the set of actions {Noise,ODE step}. We train the policy using the f-PG algorithm, with the discriminator d_ϕ estimating the density ratio $\frac{p_{\psi'}(s)}{p_*(s)}$. The training dataset $\mathcal{S}$ is composed of samples from the target distribution P_t, and the buffer B is used to store the trajectories of the agent. The training process is done for a fixed number of iterations, with a fixed number of trajectories per iteration, and a fixed number of policy updates per iteration. The hyperparameters α and ε are manually tuned. The training process is done in an off-policy manner, with the policy $\pi_{\psi'}$ used to generate the trajectories. The training procedure is summarized in Algorithm 7

Algorithm 7 f-Restart sampling

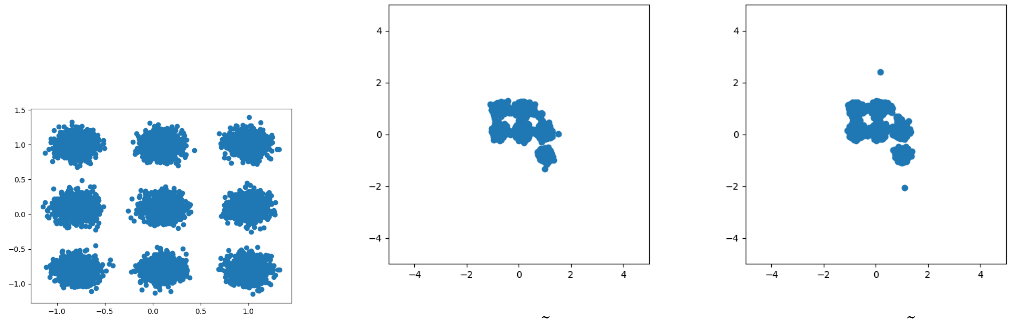
```
Let $\pi_\psi$ be the policy, $B$ be a buffer, $\mathcal{S}$ denote the training dataset (samples from the target distribution)
for $i = 1$ to num_iter do
    $B \leftarrow []$
    for $j = 1$ to num_traj_per_iter do
        sample $x_T^0$ from the prior distribution $\mathcal{N}(0, I_d)$
        $x_0 \leftarrow x_T^0$
        $t_0 \leftarrow T$
        for $l = 1$ to $L$ do
            Compute $\pi_\psi(a|x_l)$ for each action $a \in$ {Noise, ODE step} and choose the action with highest probability
            if action = Noise then
                $x_{l+1} \leftarrow$ Noise $x_t^l$ following the forward diffusion process $t_{l+1} \leftarrow t_l + 1$
            else
                $x_{l+1} \leftarrow$ Perform one step of ODE denoising $t_{l+1} \leftarrow t_l - 1$
            end if
            Store $(x_l, t_l, a_l, x_{l+1}, t_{l+1})$ in $\tau$
        end for
        Train a discriminator $d_\phi$ to estimate the density ratio $\frac{p_{\psi'}(s)}{p_*(s)}$
        Store $f'\left(d_\phi(s)\right)$ for each $s$ in $\tau$
        $B \leftarrow B + \{\tau\}$
    end for
    for $j = 1$ to num_policy_updates do
        $\theta \leftarrow \theta - \alpha \nabla_\psi J_{\mathrm{RL}}(\psi)$
    end for
end for
```

During this procedure, the quantities $f'\left(d_\phi(s)\right)$ can be seen as a guiding signal for the sampling algorithm to correct the approximation error. If at a denoising timestep t this signal is low, then the agent should add noise to the sample to correct the error, thus using the advantage of SDEs introduced in section 4.1. A high signal encourages the agent to continue the ODE denoising process.

4.3.3 Experiments and results

In order to test our proposed method, we will use a toy example of a 2D DDPM diffusion model. The target distribution is a mixture of 10 isotropic Gaussians, and we consider a trained score network that is trained on a samples stemming from a subset of modes of the target distribution. We use $f(u) = u$ for minimizing the KL divergence between the target distribution and the state visitation distribution. The first problematic behaviour observed is that the policy π_ψ favours the noising step over the ODE denoising step. After one training epoch, the probability of choosing the noise action is close to 1. This was expected as the distributions P_t and $\tilde{P}_t$ are very close when the noise step is high. A workaround to this issue is to use a *shaping* technique, consisting in encouraging the agent in the initial stages of training to perform the ODE denoising step.

Time weighting : In order to force the agent to perform the ODE denoising step at the inital stages of training, we add a denoising time weighting factor to the density ratio, which essentially puts more importance to the smaller values of t. In our experiments, we choose a weighting term $w(t)$ that is linearly decreasing with t. The effect of this shaping method was quite strong, and induced the reverse problematic behaviour : the agent would perform the ODE denoising step at each timestep. A few tweakings on the weighting scheme $w(t)$ were attempted, but the results were not satisfactory. Figure 4.2 shows that the agent does not change its behaviour post training, resulting in not improving the sampling process. We conclude that our shaping method was not effective, and provide the following directions for future work :



(a) Target distribution

(b) Samples from $\tilde{P}$ before training

(c) Samples from $\tilde{P}$ after training

Fig. 4.2: Reward shaping effect on the samples

- The optimality of the f-Restart sampling procedure was not proven. Further exploration on the theoretical aspect are needed to provide better insights for designing a training algorithm.

- Other shaping techniques should be considered to encourage the agent to perform the ODE denoising steps for high noise levels, but less frequently for low noise levels.

- Reinforcement learning relies on the Markov stationarity assumption, which might not be satisfied by our formulation of the state space. A more careful design of the state space should be considered.

4.4 Conclusion

In this chapter, we introduced the restart sampling method as a way to correct the sampling error of diffusion models. We then proposed to use reinforcement learning to dynamically noise / denoise the samples, and formalized the restart sampling procedure as a reinforcement learning problem. We presented the f-restart sampling algorithm, experimented on a toy example of a 2D diffusion model, and the results showed that our method was not effective. We provided directions for future work, notably on the theoretical aspect of the method, and on the design of the state space.

5 Conclusion

In this work, we were interested in proposing methods for refining pre-trained diffusion models. Error sources in diffusion mainly stem from two aspects : approximation error, related to the training process, and sampling error, rekated to the discretization of the backward SDE. In section 3, we presented discriminator guidance (DG) a method that is used to correct the approximation error. We improved upon the existing method by deriving an optimal training objective, and demonstrated our results on a two example and on image generation benchmarks. In section 4, we proposed f-Restart sampling, a method that is used to correct the sampling error. This method takes inspiration from restart sampling, an experimental method that mixes the advantages of a low discretization error of ODE samplers, and noise correction from SDE samplers. We propose to make this method more dynamic by using a reinforcement learning algorithm to decide when to add noise, and when to perform the denoising step. We perform experiments only on toy datasets, as this method is currently in its early stages of development.

Future work : Our study of discriminator guidance was mostly theoretical, and should be experimented with different pre-trained models, and data types. We proposed in our experiments training EDM to have a fair comparison with the original DG work, but we could have included more experiments to show the flexibility of our method to more models. In order to improve our final loss, other regularization term could be explored that leverage the generative possbilities of the score model. The f-Restart sampling method has many improvement potentialities. We believe that restraining the noise addition to the smaller denoising steps could improve the method. Although our weighting scheme was not effective, we believe it may be solved by providing a riforous analysis on the convergence of the method. Furthermore, other reinforcement learning schemes could be proposed, notably imitation learning approaches that learn a reward function given expert trajectories.

We believe that the scope of this work is important given the current usage of generative models. Boosting methods allow to improve the performance of large pre-trained models with much smaller models, and thus provide a consequential opportunity for efficiency and energy saving. In our work, both discriminator guidance and f restart sampling had the objective of minimizing an f-divergence between the refined distribution and the target distribution. However, other approaches can choose to minimize another objective, such as precision and recall of the generated samples, fairness metrics and bias reduction. We believe that the methods we proposed can be used as a basis for further research in the field of generative models.

Bibliography

[Aga+23] Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. „f-Policy Gradients: A General Framework for Goal-Conditioned RL using f-Divergences“. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 38).

[And82] Brian D.O. Anderson. „Reverse-time diffusion equation models“. In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326 (cit. on p. 13).

[Aza+19] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. *Discriminator Rejection Sampling*. 2019. arXiv: `1810.06758 [stat.ML]` (cit. on pp. 19–21).

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: `1809.11096 [cs.LG]` (cit. on p. 19).

[Bro+24] Tim Brooks, Bill Peebles, Connor Holmes, et al. „Video generation models as world simulators“. In: (2024) (cit. on p. 5).

[Cha+23] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, and Chun-Yi Lee. *On Investigating the Conservative Property of Score-Based Generative Models*. 2023. arXiv: `2209.12753 [cs.LG]` (cit. on pp. 14, 17).

[DK19] Arnak S. Dalalyan and Avetik Karagulyan. „User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient“. In: *Stochastic Processes and their Applications* 129.12 (Dec. 2019), 5278–5311 (cit. on p. 34).

[DN21] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: `2105.05233 [cs.LG]` (cit. on pp. 18, 21, 29, 30).

[Goo+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]` (cit. on pp. 1, 7, 24).

[Gup+19] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. *Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning*. 2019. arXiv: `1910.11956 [cs.LG]` (cit. on p. 37).

[GZ22] Jaime Roquero Gimenez and James Zou. *A Unified f-divergence Framework Generalizing VAE and GAN*. 2022. arXiv: `2205.05214 [stat.ML]` (cit. on p. 8).

[Heu+17] 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“. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 30).

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: `2006.11239 [cs.LG]` (cit. on pp. 1, 12, 15).

[HS22] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: `2207.12598 [cs.LG]` (cit. on p. 18).

[Hyv05] Aapo Hyvärinen. „Estimation of Non-Normalized Statistical Models by Score Matching“. In: *Journal of Machine Learning Research* 6.24 (2005), pp. 695–709 (cit. on p. 9).

[Jum+21] John Jumper, Richard Evans, Alexander Pritzel, et al. „Highly accurate protein structure prediction with AlphaFold“. In: *nature* 596.7873 (2021), pp. 583–589 (cit. on pp. 1, 37).

[Kar+22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. *Elucidating the Design Space of Diffusion-Based Generative Models*. 2022. arXiv: `2206.00364 [cs.CV]` (cit. on pp. 14, 15).

[Kim+22] Dongjun Kim, Byeonghu Na, Se Jung Kwon, et al. *Maximum Likelihood Training of Implicit Nonlinear Diffusion Models*. 2022. arXiv: `2205.13699 [cs.LG]` (cit. on p. 23).

[Kim+23] Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. *Refining Generative Process with Discriminator Guidance in Score-based Diffusion Models*. 2023. arXiv: `2211.17091 [cs.CV]` (cit. on pp. 20–24, 26, 27, 29–31, 52, 53).

[KP92] PE KLOEDEN KPj and E PLATEN. *Numerical Solutions of Stochastic Differential Equations*. 1992 (cit. on pp. 13, 33).

[KW19] Diederik P. Kingma and Max Welling. „An Introduction to Variational Autoencoders“. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), 307–392 (cit. on p. 1).

[KW22] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: `1312.6114 [stat.ML]` (cit. on p. 7).

[Kyn+19] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. „Improved Precision and Recall Metric for Assessing Generative Models“. In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.* arXiv: 1904.06991. Oct. 2019 (cit. on p. 30).

[Lai+23a] Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, et al. *FP-Diffusion: Improving Score-based Diffusion Models by Enforcing the Underlying Score Fokker-Planck Equation*. 2023. arXiv: `2210.04296 [cs.LG]` (cit. on p. 14).

[Lai+23b] Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, et al. „Fp-diffusion: Improving score-based diffusion models by enforcing the underlying score fokker-planck equation“. In: *International Conference on Machine Learning*. PMLR, 2023, pp. 18365–18398 (cit. on p. 17).

[Liu+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. „Deep Learning Face Attributes in the Wild“. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 19).

[Luc+19] Mario Lucic, Michael Tschannen, Marvin Ritter, et al. *High-Fidelity Image Generation With Fewer Labels*. 2019. arXiv: `1903.02271 [cs.LG]` (cit. on p. 18).

[Mac03] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003 (cit. on p. 19).

[Min+05] Tom Minka et al. *Divergence measures and message passing*. Tech. rep. Technical report, Microsoft Research, 2005 (cit. on p. 16).

[Mni+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: `1312.5602 [cs.LG]` (cit. on p. 37).

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. *f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization*. 2016. arXiv: `1606.00709 [stat.ML]` (cit. on pp. 7, 16, 19).

[NWJ10] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. „Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization“. In: *IEEE Transactions on Information Theory* 56.11 (Nov. 2010), 5847–5861 (cit. on p. 7).

[ØØ03] Bernt Øksendal and Bernt Øksendal. *Stochastic differential equations*. Springer, 2003 (cit. on pp. 13, 17).

[Oor+16] Aäron van den Oord, Sander Dieleman, Heiga Zen, et al. „WaveNet: A Generative Model for Raw Audio“. In: *Arxiv*. 2016 (cit. on p. 5).

[Ope+24] OpenAI, Josh Achiam, Steven Adler, et al. *GPT-4 Technical Report*. 2024. arXiv: `2303.08774 [cs.CL]` (cit. on pp. 1, 5).

[Ouy+22] Long Ouyang, Jeff Wu, Xu Jiang, et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: `2203.02155 [cs.CL]` (cit. on p. 37).

[Rom+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: `2112.10752 [cs.CV]` (cit. on pp. 1, 5, 18).

[Sar+18] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. *Deep Energy Estimator Networks*. 2018. arXiv: `1805.08306 [stat.ML]` (cit. on p. 17).

[SD+15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: `1503.03585 [cs.LG]` (cit. on p. 12).

[SE20] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: `1907.05600 [cs.LG]` (cit. on pp. 11, 12, 15).

[SH21] Tim Salimans and Jonathan Ho. „Should EBMs model the energy or the score?“ In: *Energy Based Models Workshop - ICLR 2021*. 2021 (cit. on p. 17).

[SM22] Masahiro Suzuki and Yutaka Matsuo. „A survey of multimodal deep generative models“. In: *Advanced Robotics* 36.5–6 (Feb. 2022), 261–278 (cit. on p. 5).

[Son+21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: `2011.13456 [cs.LG]` (cit. on pp. 1, 10, 13, 26, 28).

[SSK] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. „Density Ratio Estimation: A Comprehensive Review“. en. In: () (cit. on p. 54).

[Sun+24] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. *A Simple and Effective Pruning Approach for Large Language Models*. 2024. arXiv: `2306.11695 [cs.CL]` (cit. on p. 1).

[Sut+99] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. „Policy Gradient Methods for Reinforcement Learning with Function Approximation“. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999 (cit. on p. 38).

[Ueh+16] Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. *Generative Adversarial Nets from a Density Ratio Estimation Perspective*. arXiv:1610.02920 [stat]. Nov. 2016 (cit. on p. 54).

[VCC24] Manon Verbockhaven, Sylvain Chevallier, and Guillaume Charpiat. *Growing Tiny Networks: Spotting Expressivity Bottlenecks and Fixing Them Optimally*. 2024. arXiv: `2405.19816 [cs.AI]` (cit. on p. 1).

[Ver+24a] Alexandre Verine, Benjamin Negrevergne, Muni Sreenivas Pydi, and Yann Chevaleyre. „Precision-recall divergence optimization for generative modeling with GANs and normalizing flows“. In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on pp. 16, 19, 54).

[Ver+24b] Alexandre Verine, Muni Sreenivas Pydi, Benjamin Negrevergne, and Yann Chevaleyre. *Optimal Budgeted Rejection Sampling for Generative Models*. 2024. arXiv: `2311.00460 [cs.LG]` (cit. on p. 19).

[Vin11] Pascal Vincent. „A Connection Between Score Matching and Denoising Autoencoders“. In: *Neural Computation* 23.7 (July 2011), pp. 1661–1674 (cit. on pp. 9, 10, 26, 63).

[von51] John von Neumann. „Various techniques used in connection with random digits“. In: *Monte Carlo Method*. Ed. by A.S. Householder, G.E. Forsythe, and H.H. Germond. Washington, D.C.: U.S. Government Printing Office: National Bureau of Standards Applied Mathematics Series, 12, 1951, pp. 36–38 (cit. on p. 18).

[Xu+23] Yilun Xu, Mingyang Deng, Xiang Cheng, et al. *Restart Sampling for Improving Generative Processes*. Nov. 2023 (cit. on pp. 14, 33–35).

[Zhu+20] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, et al. *A Comprehensive Survey on Transfer Learning*. 2020. arXiv: `1911.02685 [cs.LG]` (cit. on p. 1).

[ZLY18] Zhaoyu Zhang, Mengyan Li, and Jun Yu. „On the convergence and mode collapse of GAN“. In: *SIGGRAPH Asia 2018 Technical Briefs*. SA ’18. Tokyo, Japan: Association for Computing Machinery, 2018 (cit. on p. 7).

List of Tables

Appendix

5.0.1 Assumptions

In the paper and in the following proofs, we make the following assumptions :

1. $p(\boldsymbol{x}) \in \mathcal{C}(\mathbb{R})$ and $\mathbb{E}_P\left[\|\boldsymbol{x}\|_2^2\right] < \infty$.

2. $q(\boldsymbol{x}) \in \mathcal{C}(\mathbb{R})$ and $\mathbb{E}_Q\left[\|\boldsymbol{x}\|_2^2\right] < \infty$.

3. $\forall t \in [0,T]$, $\boldsymbol{f}(\boldsymbol{x},t) \in \mathcal{C}(\mathcal{X})$ and $\exists C > 0$ such that $\forall \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T], \|\boldsymbol{f}(\boldsymbol{x},t)\|_2 \leq C(1 + \|\boldsymbol{x}\|_2)$.

4. $\exists C > 0$ such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d,\ \|\boldsymbol{f}(\boldsymbol{x},t) - \boldsymbol{f}(\boldsymbol{y},t)\|_2 \leq C\|\boldsymbol{x} - \boldsymbol{y}\|_2$.

5. $g \in \mathcal{C}([,\mathcal{T}])$ and $\forall t \in [0,T],\ |g(t)| > 0$.

6. For any open bounded set $\mathcal{O} \subset \mathbb{R}$, $\int_0^T \int_{\mathcal{O}} \|p_t(\boldsymbol{x}_t)\|_2^2 + d\|\nabla_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t)\|_2^2 \mathrm{d}\boldsymbol{x}_t \mathrm{d}t < \infty$.

7. $\exists C > 0$ such that $\forall \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T] :\ \|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\|_2 \leq C(1 + \|\boldsymbol{x}_t\|_2)$.

8. $\exists C > 0$ such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d, t \in [0,T] :\ \|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}_t)\|_2 \leq C\|\boldsymbol{x}_t - \boldsymbol{y}_t\|_2$.

9. $\exists C > 0$ such that $\forall \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T] :\ \|\nabla_{\boldsymbol{x}_t} \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\|_2 \leq C(1 + \|\boldsymbol{x}_t\|_2)$.

10. $\exists C > 0$ such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d, t \in [0,T] :\ \|\nabla_{\boldsymbol{x}_t} \boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \boldsymbol{s}_\theta(\boldsymbol{y}_t, t)\|_2 \leq C\|\boldsymbol{x}_t - \boldsymbol{y}_t\|_2$.

11. Nokitov's condition: $\mathbb{E}_P\left[\exp\left(\frac{1}{2}\int_0^T \|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\|_2^2 \mathrm{d}t\right)\right] < \infty$.

12. $\forall t \in [0,T], \exists k > 0 :\ p_t(\boldsymbol{x}) = O(e^{-\|\boldsymbol{x}\|_2^k})$ as $\|\boldsymbol{x}\|_2 \to \infty$.

Theorem 8. *[Kim+23] Suppose that the score network s_θ satisfies assumption 2. Then $s_\theta \in S_{\mathrm{sol}}$. We thus denote by $\hat{P}_t$ the distribution with density $\hat{p}_t(x)$ such that $s_\theta(x,t) = \nabla \log \hat{p}_t(x)$*

Theorem 9. *[Kim+23] Suppose that assumption 2 is satisfied by the estimated score network s_θ. Suppose in addition that $s_\theta(x,T) = \nabla \log \pi(x)$. Then, by noting $c_\theta(x,t) = \nabla \log p_t(x) - s_\theta(x,t) = \nabla \log \frac{p_t(x)}{s_\theta(x,t)}$, the backward SDE defined by :*

$$dx = f(x,t) - \frac{1}{2}g(t)^2(s_\theta(x,t) + c_\theta(x,t))dt \tag{5.1}$$

coincides with the backward SDE of the diffusion process induced by the target distribution (Equation 2.19).

5.0.2 Proof of Theorem 5

Let $\{\boldsymbol{x}(t)\}_{t\in[0,T]}$ be a diffusion process defined by Equation (2.18). Assume that P and $\widehat{P}$ satisfy the assumptions detailed in Appendix 5.0.1. Then, for every $\varepsilon > 0$ and for every $\delta > 0$, there exists a discriminator $d : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ trained to minimize the cross-entropy such that:

$$\mathcal{L}^d_{\mathrm{CE}}(\phi) \leq \varepsilon \quad \text{and} \quad \mathcal{D}_{\mathrm{KL}}(P\|\widetilde{P}) \geq \delta, \tag{5.2}$$

where $\widetilde{P}$ is the distribution induced by discriminator guidance with d.

Finding a problematic case

The aim of discriminator guidance is to minimize the Kullback-Leibler divergence between the target distribution P, and the refined distribution $\tilde{P}$. Following the assumptions detailed in Appendix 5.0.1, this quantity can be written as :

$$\mathcal{D}_{\mathrm{KL}}(P\|\widetilde{P}) = \mathcal{D}_{\mathrm{KL}}(P_T\|Q) + \int_0^T g(t)^2 \mathbb{E}_{P_t}\left[\|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - \boldsymbol{s}_\theta(\boldsymbol{x}_t,t) - \nabla_{\boldsymbol{x}_t} d_\phi(\boldsymbol{x}_t,t)\|_2^2\right] \mathrm{d}t. \tag{5.3}$$

Thus, discriminator guidance minimizes the latter term of the equality, that we denote as :

$$E_\phi = \int_0^T g(t)^2 \mathbb{E}_{P_t}\left[\|\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - \boldsymbol{s}_\theta(\boldsymbol{x}_t,t) - \nabla_{\boldsymbol{x}_t} d_\phi(\boldsymbol{x}_t,t)\|_2^2\right] \mathrm{d}t. \tag{5.4}$$

where $d_\phi(\boldsymbol{x}_t)$ is the estimated density ratio by training the discriminator d_ϕ.

We consider the case of estimating d_ϕ by minimizing the cross-entropy up to ε. We would like to find a pathological case where this would not minimize the mean square error in Equation 5.4. The estimation error of the discriminator can be derived from the duality difference in estimating the g-Bregman divergence, with :

$$g(\boldsymbol{x}) = u\log(u) + (u+1)\log(u+1) \tag{5.5}$$

We denote by $\mathcal{D}_g(P\|P') = \inf_{f\in\mathcal{M}} \mathbb{E}_{(\boldsymbol{x})\sim p_t}\left[\log(\sigma(f(\boldsymbol{x}_t)))\right] - \mathbb{E}_{\boldsymbol{x}\sim\hat{p}_t}\left[-\log(1-\sigma(f(\boldsymbol{x}_t)))\right]$, where $\mathcal{M}$ denotes the set of all measurable functions. This quantity estimates the optimal cross-entropy that any measurable density ratio estimator f can achieve.

The estimated d_ϕ has minimized the following quantity $\mathcal{D}_g^\phi = inf_{\omega\in R^N}\mathbb{E}_{\boldsymbol{x}\sim p_t}\left[\log(\sigma(T_\omega(\boldsymbol{x}_t)))\right] - \mathbb{E}_{\boldsymbol{x}\sim\hat{p}_t}\left[-\log(1-\sigma(T_\omega(\boldsymbol{x}_t)))\right]$. We use the results of [Ueh+16; SSK; Ver+24a] to compute the estimation error of the cross-entropy :

For any discriminator $T_\phi : \mathcal{X} \to R$ and density ratio estimation $d_\phi = \nabla_{\boldsymbol{x}_t} g^*(T_\phi(\boldsymbol{x}))$, we have :

$$D_g(P\|\hat{P}) - D_g^\phi(P\|\hat{P}) = \mathbb{E}_{\hat{P}}\left[\text{Breg}_g\left(d_\phi(\boldsymbol{x}), \frac{p(\boldsymbol{x})}{\hat{p}(\boldsymbol{x})}\right)\right] \tag{5.6}$$

where $g*$ denotes the Fenchel conjugate of g.

Suppose a discriminator was trained on minimizing the cross-entropy D_{g}^ϕ such that it is ε-close to the optimal cross-entropy D_{g}. Thus, we can write :

$$D_g(P\|\hat{P}) - D_g^\phi(P\|\hat{P}) \leq \varepsilon \tag{5.7}$$

We consider the case when, for all $(\boldsymbol{x}) \sim \hat{P}$, $\text{Breg}_g(d_\phi(\boldsymbol{x}), \frac{p(\boldsymbol{x})}{\hat{p}(\boldsymbol{x})}) \leq \varepsilon$ Suppose moreover that the estimated density ratio $d_\phi(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{\hat{p}(\boldsymbol{x})} + h(\boldsymbol{x})$. We also note $\frac{p(\boldsymbol{x})}{\hat{p}(\boldsymbol{x})} = r_{\text{opt}}(\boldsymbol{x})$. Thus, we can write, for $\boldsymbol{x} \in \mathbb{R}$:

$$\text{Breg}_g\left(d_\phi(\boldsymbol{x}), \frac{p(\boldsymbol{x})}{\hat{p}(\boldsymbol{x})}\right) = g(d_\phi(\boldsymbol{x})) - g(r_{\text{opt}}(\boldsymbol{x})) - \nabla_{\boldsymbol{x}_t} g(r_{\text{opt}}(\boldsymbol{x}))(d_\phi(\boldsymbol{x}) - r_{\text{opt}}(\boldsymbol{x})))$$

With :

$$\begin{aligned} g(d_\phi(\boldsymbol{x})) &= (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}))\log(r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x})) \\ &\qquad - (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1)\log(r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1) \qquad (5.8) \\ &= (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}))\log(r_{\text{opt}}(\boldsymbol{x})) \qquad (5.9) \\ &\qquad + (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}))\log\left(1 + \frac{h(\boldsymbol{x})}{r_{\text{opt}}(\boldsymbol{x})}\right) \\ &\qquad - (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1)\log(r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1) \\ &\leq \begin{matrix} (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}))\log(r_{\text{opt}}(\boldsymbol{x})) + (r_{\text{opt}}(\boldsymbol{x}) + h)\dfrac{h(\boldsymbol{x})}{r_{\text{opt}}(\boldsymbol{x})} \\ - (r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1)\log(r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1) \end{matrix} \qquad (5.10) \end{aligned}$$

Moreover,

$$g(r_{\text{opt}}(\boldsymbol{x})) = r_{\text{opt}}(\boldsymbol{x})\log(r_{\text{opt}}(\boldsymbol{x})) + (r_{\text{opt}}(\boldsymbol{x}) + 1)\log(r_{\text{opt}}(\boldsymbol{x}) + 1) \qquad (5.11)$$

and the third term is given by

$$\nabla_{\boldsymbol{x}_t} g(r_{\text{opt}}(\boldsymbol{x}))(d_\phi(\boldsymbol{x}) - r_{\text{opt}}(\boldsymbol{x}))) = (\log(r_{\text{opt}}(\boldsymbol{x})) - \log(r_{\text{opt}}(\boldsymbol{x}) + 1))h(\boldsymbol{x}) \qquad (5.12)$$

Inequality 5.10 results from the Taylor expansion of $\log(1 + (\boldsymbol{x}))$ when $(\boldsymbol{x})$ is close to zero, as $\log(1+(\boldsymbol{x})) \leq (\boldsymbol{x})$. By summing the expressions 5.10,5.11,5.12, we obtain an upper bound on the g-Bregman divergence between the true and estimated density ratio :

$$\begin{aligned}\mathrm{Breg}_g\left(d_\phi(\boldsymbol{x}), r_{\mathrm{opt}}(\boldsymbol{x})\right) \leq & \left(1 + \frac{h^2(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x})}\right) + \left(r_{\mathrm{opt}}(\boldsymbol{x}) + 1\right)\log\left(r_{\mathrm{opt}} + 1\right) + h(\boldsymbol{x})\log\left(r_{\mathrm{opt}}(\boldsymbol{x}) + 1\right) \\ & - \left(r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1\right)\log\left(r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1\right)\end{aligned} \tag{5.13}$$

$$\begin{aligned}= & \left(h(\boldsymbol{x}) + r_{\mathrm{opt}}(\boldsymbol{x})\right)\frac{h(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x})} \\ & + \left[r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1\right]\left[\log(r_{\mathrm{opt}}(\boldsymbol{x}) + 1) - \log((r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1))\right]\end{aligned} \tag{5.14}$$

$$\begin{aligned}= & \left(h(\boldsymbol{x}) + r_{\mathrm{opt}}(\boldsymbol{x})\right)\frac{h(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x})} \\ & + \left(r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1\right)\log\left(1 - \frac{h(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1}\right)\end{aligned} \tag{5.15}$$

$$\leq \left(h(\boldsymbol{x}) + r_{\mathrm{opt}}(\boldsymbol{x})\right)\frac{h(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x})} - \left(r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1\right)\frac{h(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x}) + h(\boldsymbol{x}) + 1} \tag{5.16}$$

$$\leq \frac{h(\boldsymbol{x})^2}{r_{\mathrm{opt}}(\boldsymbol{x})} \tag{5.17}$$

One particular case of satisfaction of inequality 5.7 is when all the elements in the expectation $\mathbb{E}_{\hat{P}}$ are below ε, that is when :

$$\frac{h^2(\boldsymbol{x})}{r_{\mathrm{opt}}(\boldsymbol{x})} \leq \varepsilon \tag{5.18}$$

$$\Rightarrow h(\boldsymbol{x}) \leq \sqrt{\varepsilon r_{\mathrm{opt}}(\boldsymbol{x})} \tag{5.19}$$

This bound is notably satisfied for $h(\boldsymbol{x}) = \sin(\omega \boldsymbol{x})\sqrt{\varepsilon r_{opt}(\boldsymbol{x})}$, $\forall \omega \in \mathbb{R}$

Computing the gain

We will now show that the value of the E_ϕ (c.f Equation 5.4) can go to infinity for an estimated density ratio that has an ε-optimal cross-entropy. For this, we write, for $r(\boldsymbol{x}) = r_{opt}(\boldsymbol{x}) + h(\boldsymbol{x})$, with $h(\boldsymbol{x}) = \sin(\omega(\boldsymbol{x}))\sqrt{\varepsilon r_{opt}(\boldsymbol{x})}$: This bound is notably satisfied for $h(\boldsymbol{x}) = \sin(\omega(\boldsymbol{x}))\sqrt{\varepsilon r_{\mathrm{opt}}(\boldsymbol{x})}$, $\forall \omega \in \mathbb{R}$

5.0.3 Computing the gain

We will now show that the value of the gain (c.f Equation 5.4) can go to infinity for an estimated density ratio that has an ε-optimal cross-entropy. For this, we compute, for $d_\phi(\boldsymbol{x}) = r_{\text{opt}}(\boldsymbol{x}) + h(\boldsymbol{x})$, with $h(\boldsymbol{x}) = \sin(\omega(\boldsymbol{x}))\sqrt{\varepsilon r_{\text{opt}}(\boldsymbol{x})}$:

$$\nabla_{\boldsymbol{x}_t}\log(d_\phi(\boldsymbol{x})) = \nabla_{\boldsymbol{x}_t}\log(r_{\text{opt}}(\boldsymbol{x})) + \nabla_{\boldsymbol{x}_t}\log\left(1 + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\sin(\omega(\boldsymbol{x}))\right) \tag{5.20}$$

Moreover, we have :

$$\nabla_{\boldsymbol{x}_t}\log\left(1 + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\sin(\omega(\boldsymbol{x}))\right) = \frac{\sqrt{\varepsilon}\left(-\frac{1}{2}\nabla_{\boldsymbol{x}_t} r_{\text{opt}}(\boldsymbol{x}) r_{\text{opt}}^{-\frac{3}{2}}\sin(\omega(\boldsymbol{x})) + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\omega\cos(\omega(\boldsymbol{x}))\right)}{1 + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\sin(\omega(\boldsymbol{x}))} \tag{5.21}$$

Thus, the gain for a fixed timestep t is given by :

$$E_\phi^t = \mathbb{E}_{P_t}\left[||\nabla_{\boldsymbol{x}_t} log(r_{\text{opt}}(\boldsymbol{x})) - \nabla_{\boldsymbol{x}_t}\log d_\phi(\boldsymbol{x})||_2^2\right] \tag{5.22}$$

$$= \mathbb{E}_{P_t}\underbrace{\left\|\frac{\sqrt{\varepsilon}\left(-\frac{1}{2}\nabla_{\boldsymbol{x}_t}\left[r_{\text{opt}}(\boldsymbol{x})\right] r_{\text{opt}}^{-\frac{3}{2}}\sin(\omega(\boldsymbol{x})) + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\omega\cos(\omega(\boldsymbol{x}))\right)}{1 + \sqrt{\frac{\varepsilon}{r_{\text{opt}}(\boldsymbol{x})}}\sin(\omega(\boldsymbol{x}))}\right\|_2^2}_{B_\omega(\boldsymbol{x})} \tag{5.23}$$

By setting ω to be very large, and using the following properties :

- The set $\left\{\boldsymbol{x} \in \mathbb{R}^d \middle| \cos(\omega\boldsymbol{x}) = 0\right\})$ has a mass 0 with respect to P.
- $\mathbb{E}_P[.] = P(r_{\text{opt}}(\boldsymbol{x}) = \infty)\mathbb{E}_P[.|r_{\text{opt}}(\boldsymbol{x}) = \infty] + P(r_{\text{opt}}(\boldsymbol{x}) < \infty)\mathbb{E}_P[.|r_{\text{opt}}(\boldsymbol{x}) < \infty]$
- $\mathbb{E}_P[.|r_{\text{opt}}(\boldsymbol{x}) = \infty] = 0$ and $P(r_{\text{opt}}(\boldsymbol{x}) < \infty) > 0$

We have that :

$$E_\phi^t = P(r_{\text{opt}}(\boldsymbol{x}) = \infty)\mathbb{E}_{P_t}\left[B_\omega(\boldsymbol{x})|r_{\text{opt}}(\boldsymbol{x}) = \infty\right] + P(r_{\text{opt}}(\boldsymbol{x}) < \infty)\mathbb{E}_{P_t}\left[B_\omega(\boldsymbol{x})|r_{\text{opt}}(\boldsymbol{x}) < \infty\right] \tag{5.24}$$

Thus, $E_\phi^t \to \infty$ as $\omega \to \infty$, which concludes our proof. The effect of ω can be seen in Figures 5.1, 3.1,3.2, where high values lose all the gradient information necessary for correcting the sampling process.

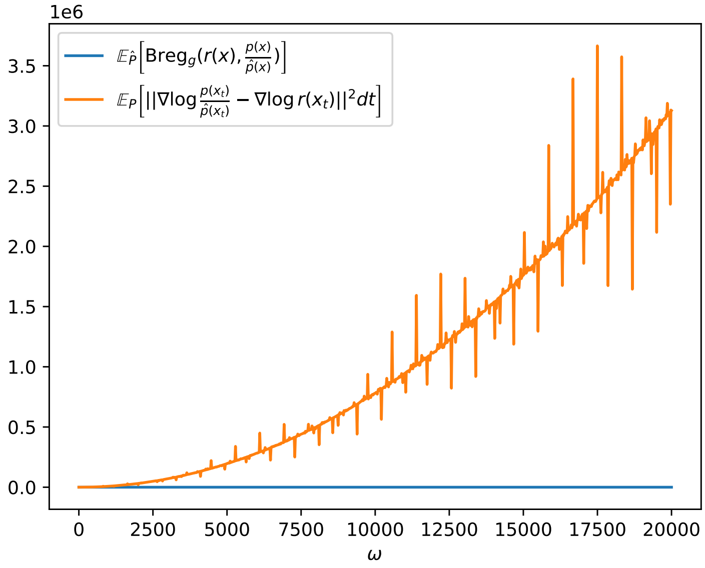


Fig. 5.1: 1-dimensional example : Density ratio between two Gaussians, with a discriminator having an ε-optimal cross-entropy and a gain that goes to infinity

5.0.4 Proof of Theorem 6

Proof. Define the measures $\tilde{\mu} = \min\left(P, \hat{P}\right), \mu_P = \max\left(0, P - \hat{P}\right)$ and $\mu_{\hat{P}} = \max\left(0, \hat{P} - P\right)$. Observe that $P = \tilde{\mu} + \mu_P$, $\hat{P} = \tilde{\mu} + \mu_{\hat{P}}$ and that $\tilde{\mu}\left(\mathbb{R}\right) = 1 - TV$ where TV is the total variation distance between P and $\hat{P}$. Define the distribution $\mu = \frac{\tilde{\mu}}{\tilde{\mu}(\mathbb{R})}$ and let f_μ and $f_{\tilde{\mu}}$ be the density of μ and $\tilde{\mu}$ with respect to Lebesgue measure. Note that because $\tilde{\mu}$ is not normalized, $f_{\tilde{\mu}}$ does not integrate to one.

Let us build two sets S and S' iteratively using the following rejection sampling procedure: First, start with $S = S' = \varnothing$. Then, for each i, add x_i (respectively x'_i) to the set S (respectively S') with probability $\frac{f_{\tilde{\mu}}(x_i)}{p(x_i)}$ (respectively $\frac{f_{\tilde{\mu}}(x'_i)}{\hat{p}(x'_i)}$). For now, denote $M_1 = |S|$ and $M_2 = |S'|$. It is easy to check using standard properties of rejection sampling that $\mathbb{E}\left[M_1\right] = \mathbb{E}\left[M_2\right] = N \times \left(1 - TV\right)$. Finally, take the largest of both sets S and S' and remove its last added elements until both sets have same cardinality $M = \min\left(M_1, M_2\right)$. Note that in all three sets S,S' and $S \cup S'$, examples are distributed i.i.d. from μ.

Let us define the mean square error:

$$\begin{aligned}
MSE &= \mathbb{E}_{x\sim P}\left[\left(\nabla_x d^\star(x) - \nabla_x d_\phi(x)\right)^2\right] \\
&\geq \mathbb{E}\left[\left(\nabla_x d_\phi\right)^2\right] - 2\mathbb{E}\left[\nabla_x d^\star . \nabla_x d_\phi\right] \\
&\geq \mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right] - 2\sqrt{\mathbb{E}_p\left[(\nabla_x d^\star)^2\right]}\sqrt{\mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right]}
\end{aligned}$$

The last line follows from Cauchy-Schwartz inequality.

We are interested into lower-bounding the *expected* mean square error, noted $\mathbb{E}MSE$, where the expectation is over the training set $\{x_1, x'_1 \dots\}$ on which the discriminator d_ϕ is trained. So we would like to bound it.

To derive this bound, we will first lower bound $\mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right]$.

First, consider an arbitrary interval $B \subset \mathbb{R}$ of size β containing at least one point $z \in S$ and one point $z' \in S'$ such that $z < z'$.

Then we can write

$$\begin{aligned}
\int_{\mathbb{R}} \left(\nabla d_\phi\right)^2 dP(x) &\geq \int_B \left(\nabla d_\phi\right)^2 dP(x) \\
&\geq \int_B \left(\nabla d_\phi\right)^2 d\tilde{\mu}(x) = \int_B \left(\nabla d_\phi\right)^2 f_{\tilde{\mu}}(x) dx \\
&\geq \left(\inf_{x\in B} f_{\tilde{\mu}}(x)\right) \int_B \left(\nabla d_\phi\right)^2 dx \\
&\geq \left(\inf_{x\in B} f_{\tilde{\mu}}(x)\right) \int_z^{z'} \left(\nabla d_\phi\right)^2 dx \\
&= \left(\inf_{x\in B} f_{\tilde{\mu}}(x)\right) (z'-z) \int_z^{z'} \frac{\left(\nabla d_\phi\right)^2}{z'-z} dx \\
&\geq \left(\inf_{x\in B} f_{\tilde{\mu}}(x)\right) (z'-z) \left(\int_z^{z'} \frac{\nabla d_\phi}{z'-z} dx\right)^2 \quad \text{(by Jensen)} \\
&= \frac{\inf_{x\in B} f_{\tilde{\mu}}(x)}{z'-z} \left(\int_z^{z'} \nabla d_\phi dx\right)^2 \\
&\geq \frac{\inf_{x\in B} f_{\tilde{\mu}}(x)}{\beta} \left(\int_z^{z'} \nabla d_\phi dx\right)^2
\end{aligned}$$

Because both p and $\hat{p}$ are L-Lipschitz, $f_{\tilde{\mu}}$ has also this Lipschitz property. So for any $u \in B$ we have $f_{\tilde{\mu}}(u) - \inf_{x\in B} f_{\tilde{\mu}}(x) \leq L\beta$, so $\inf_{x\in B} f_{\tilde{\mu}}(x) + L\beta \geq f_{\tilde{\mu}}(u)$, so $\inf_{x\in B} f_{\tilde{\mu}}(x) + L\beta \geq \frac{1}{\beta}\int_B f_{\tilde{\mu}}(x)dx = \frac{\tilde{\mu}(B)}{\beta}$.So we can write

$$\int_B \nabla d_\phi^2 dP(x) \geq \left(\frac{\tilde{\mu}(B)}{\beta^2} - L\right)\left(\int_z^{z'} \nabla d_\phi dx\right)^2$$

Note that if $z > z'$ we would get by the same reasoning

$$\int_B \nabla d_\phi^2 dP(x) \geq \left(\frac{\tilde{\mu}(B)}{\beta^2} - L\right)\left(\int_{z'}^{z} \nabla d_\phi dx\right)^2$$

To bound $\int_z^{z'} \nabla d_\phi dx$, recall that the cross-entropy for each i is such that $\log\left(1 + e^{-d_\phi(x_i)}\right) \leq \epsilon$ and $\log\left(1 + e^{d_\phi(x_i')}\right) \leq \epsilon$. Thus, $1 + e^{-d_\phi(x_i)} \leq e^\epsilon$ and $1 + e^{d_\phi(x_i')} \leq e^\epsilon$. So $d_\phi(x_i) \geq \alpha$ and $d_\phi(x_i') \leq -\alpha$ where $\alpha = -\log(e^\epsilon - 1)$. This also applies to z and z', so $\int_{\min(z,z')}^{\max(z,z')} \nabla d_\phi dx = \pm 2\alpha$. Finally, we can summarize our findings by:

$$\text{if both } S \text{ and } S' \text{ intersect with } B \text{ then } \int_B \nabla d_\phi dP(x) \geq \left(\frac{\tilde{\mu}(B)}{\beta^2} - L\right) 4\alpha^2$$

Our goal now will be to show that there exists a small enough interval B containing two points z and z' from S and S' with high probability.

First, let us build an interval A such that $\mu(A) \geq \frac{1}{2}$. By Chebyshev's inequality, we have that $\mu(\{x : |x - \mathbb{E}_\mu x| \geq b\}) \leq \frac{Var_\mu}{b^2}$ for any b, so using the bound on the variance of lemma 1, we get that there exists a finite interval $A = [\mathbb{E}_\mu X - b, \mathbb{E}_\mu X + b]$ where $b = \frac{\min\left(\sqrt{Var_P}, \sqrt{Var_{\hat{P}}}\right)}{1 - TV}$ such that $\mu(A) \geq \frac{1}{2}$.

Next, because $\mu(A) \geq \frac{1}{2}$, for any $0 < \beta < 2b$ there exists an interval $B \subset A$ of size β such that $\mu(B) \geq \frac{\beta}{4b}$. Let us bound the probability that both S and S' intersect with B:

$$\begin{aligned}
\mathbb{P}\left[B \cap S \neq \varnothing \wedge B \cap S' \neq \varnothing\right] =& 1 - \mathbb{P}\left[B \cap S = \varnothing \vee B \cap S' = \varnothing\right] \\
=& 1 - \mathbb{P}\left[B \cap S = \varnothing\right] - \mathbb{P}\left[B \cap S' = \varnothing\right] + \mathbb{P}\left[B \cap S' = \varnothing \wedge B \cap S' = \varnothing\right] \\
=& 1 - \mu\left(\bar{B}\right)^M - \mu\left(\bar{B}\right)^M + \mu\left(\bar{B}\right)^{2M} \\
\geq& 1 - 2\left(1 - \mu(B)\right)^M \geq 1 - 2\exp\left(-M\mu(B)\right) \\
\geq& 1 - 2\exp\left(-\frac{M\beta}{4b}\right)
\end{aligned}$$

For any non negative random variable Z, we know by the law of total expectation that $\mathbb{E}Z \geq \mathbb{E}\left[Z \mid condition\right]\mathbb{P}\left(condition\right)$, so in our case we have (here the expectation is over the dataset and the rejection sampling procedure):

$$\begin{aligned}
\mathbb{E}\int_B \left(\nabla d_\phi\right)^2 dP(x) \geq& \mathbb{E}\left[\int_B \left(\nabla d_\phi\right)^2 dP(x) \mid B\cap S \neq \varnothing \wedge B \cap S' \neq \varnothing\right] \times \mathbb{P}\left[B\cap S \neq \varnothing \wedge B \cap S' \neq \varnothing\right] \\
&\geq \left(\frac{\tilde{\mu}\left(B\right)}{\beta^2} - L\right)4\alpha^2 \times \left(1 - 2\exp\left(-\frac{M\beta}{4b}\right)\right) \\
&\geq \left(\frac{\mu\left(B\right)\left(1-TV\right)}{\beta^2} - L\right)4\alpha^2 \times \left(1 - 2\exp\left(-\frac{M\beta}{4b}\right)\right) \\
&\geq \left(\frac{1-TV}{4b\beta} - L\right)4\alpha^2 \times \left(1 - 2\exp\left(-M\frac{\beta}{4b}\right)\right)
\end{aligned}$$

Choosing $\beta = \frac{b}{M}4\log 4$ we get a $\left(1 - 2\exp\left(-M\frac{\beta}{4b}\right)\right) = 1/2$ and

$$\begin{aligned}
\mathbb{E}\int_B \left(\nabla d_\phi\right)^2 dP(x) &\geq \mathbb{E}\left(\frac{1-TV}{4b\beta} - L\right)2\alpha^2 \\
&= \left(\frac{(1-TV)\mathbb{E}\left[M\right]}{8b^2\log 4} - 2L\right)\left(\log\left(e^\epsilon - 1\right)\right)^2
\end{aligned}$$

It is known that $\mathbb{E}M = \mathbb{E}\min\left(M_1, M_2\right) \geq \frac{1}{2}\mathbb{E}M_1 = \frac{N\times(1-TV)}{2}$. So we finally get

$$\begin{aligned}
\mathbb{E}\int_\infty \left(\nabla d_\phi\right)^2 dP(x) &\geq \left(N\frac{(1-TV)^2}{16b^2\log 4} - 2L\right)\left(\log\left(e^\epsilon - 1\right)\right)^2 \\
&= \left(N\frac{(1-TV)^4}{16\log 4\min\left(Var_P, Var_{\hat{P}}\right)} - 2L\right)\left(\log\left(e^\epsilon - 1\right)\right)^2
\end{aligned}$$

Finally, we can bound the expected MSE:

$$\begin{aligned}
\mathbb{E}MSE &\geq \mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right] - 2\sqrt{\mathbb{E}_p\left[\left(\nabla_x d^\star\right)^2\right]}\sqrt{\mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right]} \\
&\geq \mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right] - 2\sqrt{\mathbb{E}_p\left[\left(\nabla_x d^\star\right)^2\right]}\sqrt{\mathbb{E}_p\left[\left(\nabla_x d_\phi\right)^2\right]}
\end{aligned}$$

□

Lemma 1. *Let P and $\hat{P}$ be two distributions over $\mathbb{R}$ and let $\tilde{\mu} = \min\left(P, \hat{P}\right)$. Then*

$$Var_{\mu} \leq \frac{1}{2(1-TV)^2} \min\left(Var_P, Var_{\hat{P}}\right)$$

Proof. Define $\mu = \frac{\tilde{\mu}}{\tilde{\mu}(\mathbb{R})}$. As before, $\mu = \frac{\tilde{\mu}}{1-TV}$, so we have

$$\begin{aligned} Var_{\mu} &= \mathbb{E}_{X \sim \mu}\left[(X - \mathbb{E}X)^2\right] = \frac{1}{2}\mathbb{E}_{X,X' \sim \mu}\left[(X - X')^2\right] \\ &= \frac{1}{2}\int\int (x - x')^2 d\mu(x) d\mu(x') \\ &= \frac{1}{2(1-TV)^2}\int\int (x - x')^2 d\tilde{\mu}(x) d\tilde{\mu}(x') \\ &\leq \frac{1}{2(1-TV)^2}\min\left(\int\int (x-x')^2 dp dp, \int\int (x-x')^2 d\hat{p} d\hat{p}\right) \\ &= \frac{1}{2(1-TV)^2}\min\left(Var_P, Var_{\hat{P}}\right) \end{aligned}$$

□

5.0.5 Proof of Proposition 1

Proposition 2. *Assume that P and $\widehat{P}$ satisfy the assumptions detailed in Appendix 5.0.1. Then, the following holds:*

$$\operatorname*{argmin}_{\phi} \mathcal{L}^d_{\mathrm{SM}}(\phi) = \operatorname*{argmin}_{\phi} \mathcal{L}^d_{\mathrm{MSE}}(\phi). \tag{5.25}$$

First, we can show that:

$$\mathcal{L}^{d}_{\text{MSE}}(\phi) = \int_0^T \lambda(t)\mathbb{E}_{P_0,P_{t|\boldsymbol{x}_0}}\left[\|\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) - \nabla_{\boldsymbol{x}_t}\log\widehat{p}_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t)\|_2^2\right]\mathrm{d}t \tag{5.26}$$

$$\begin{aligned}
&= \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t)\right\|_2^2\right]\mathrm{d}t \qquad (5.27)\\
&\quad + \int_0^T \lambda(t)\mathbb{E}_{P_0,P_{t|\boldsymbol{x}_0}}\left[\left\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}{\widehat{p}_t(\boldsymbol{x}_t)}\right\rangle\right]\mathrm{d}t\\
&\quad + \int_0^T \lambda(t)\mathbb{E}_{P_0,P_{t|\boldsymbol{x}_0}}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}{\widehat{p}_t(\boldsymbol{x})}\right\|_2^2\right]\mathrm{d}t\\
&= \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t)\right\|_2^2\right]\mathrm{d}t + J(\phi) + C_1, \qquad (5.28)
\end{aligned}$$

where C_1 is a constant independent of ϕ. And we can write J as:

$$\begin{aligned}
J(\phi) &= \int_0^T \lambda(t)\mathbb{E}_{P_0,P_{t|\boldsymbol{x}_0}}\left[\left\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}{\widehat{p}_t(\boldsymbol{x}_t)}\right\rangle\right]\mathrm{d}t \qquad (5.29)\\
&= \int_0^T \lambda(t)E_{P_0,P_{t|\boldsymbol{x}_0}}\left[\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\rangle\right]\mathrm{d}t \qquad (5.30)\\
&\quad - \int_0^T \lambda(t)E_{P_0,P_{t|\boldsymbol{x}_0}}\left[\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\widehat{p}_t(\boldsymbol{x}_t)\rangle\right]\mathrm{d}t.
\end{aligned}$$

Using the result of [Vin11], we can show the term in the first integral can be rewritten as:

$$E_{P_0,P_{t|\boldsymbol{x}_0}}\left[\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\rangle\right] = E_{P_t}\left[\left\langle\nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\right\rangle\right]. \tag{5.31}$$

Thus, we can rewrite J as:

$$\begin{aligned}
J(\phi) &= \int_0^T \lambda(t)E_{P_t}\left[\left\langle\nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)\right\rangle\right]\mathrm{d}t \qquad (5.32)\\
&\quad - \int_0^T \lambda(t)E_{P_0,P_{t|\boldsymbol{x}_0}}\left[\langle\nabla_{\boldsymbol{x}_t}d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\widehat{p}_t(\boldsymbol{x}_t)\rangle\right]\mathrm{d}t\\
&= \int_0^T \lambda(t)E_{P_t}\left[\left\langle\nabla_{\boldsymbol{x}_t}d_\phi(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t)}{\widehat{p}(\boldsymbol{x}_t)}\right\rangle\right]\mathrm{d}t. \qquad (5.33)
\end{aligned}$$

And on the other side, we have:

$$\begin{aligned}
\mathcal{L}_{SM}^{d}(\phi) &= \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\|\nabla_{\boldsymbol{x}_t} d_\phi(\boldsymbol{x}_t,t)\|_2^2\right]\mathrm{d}t && (5.34)\\
&= \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t} d_\phi(\boldsymbol{x}_t,t)\right\|_2^2\right]\mathrm{d}t && (5.35)\\
&\quad + \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\left\langle \nabla_{\boldsymbol{x}_t} d(\boldsymbol{x}_t,t), \nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t)}{\widehat{p_t}(\boldsymbol{x}_t)}\right\rangle\right]\mathrm{d}t\\
&\quad + \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log\frac{p_t(\boldsymbol{x}_t)}{\widehat{p_t}(\boldsymbol{x}_t)}\right\|_2^2\right]\mathrm{d}t\\
&= \int_0^T \lambda(t)\mathbb{E}_{P_t}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t} d_\phi(\boldsymbol{x}_t,t)\right\|_2^2\right]\mathrm{d}t + J(\phi) + C_2 && (5.36)
\end{aligned}$$

using Equation (5.33) and C_2 is a constant independent of ϕ. Thus, we have $\mathcal{L}_{\mathrm{MSE}}^{d}(\phi) = \mathcal{L}_{\mathrm{SM}}^{d}(\phi) + C_3$, where C_3 is a constant independent of ϕ. Thus, the minimizer of $\mathcal{L}_{\mathrm{MSE}}^{d}(\phi)$ is the same as the minimizer of $\mathcal{L}_{\mathrm{SM}}^{d}(\phi)$.