Generative models: Diff-Instruct

Marceau Leclerc[1] and Grégoire Mourre[1]

[1]Master Data Science, École Polytechnique

January 2025

Contents

1 Introduction

Diffusion models (DMs) have gained a lot of traction in recent years, exhibiting a tremendous capacity to generate data such as images and other modalities [Ho et al., 2020]. They are arguably easy to train and tend to capture complex and various underlying data representations. One of their downsides is their efficiency both in term of memory usage and sampling time, due to their inherent iterative aspect. On the other hand generative adversarial networks (GANs) [Goodfellow et al., 2014] demonstrate opposite upsides and drawbacks. The - adversarial - nature of their training procedure is subject to many technical difficulties, despite being clearly able to produce remarkable results especially when leveraged to accomplish specific tasks given intricate architecture tricks [Karras et al., 2019]. One of their most prominent benefit is however their fast sampling. The method that we will present and showcase her, DiffInstruct (DI) [Luo et al., 2024], aims to use powerful DMs to refine simpler and faster generative models through a data-free procedure. This procedure can be used to distill knowledge directly using the weights of its teacher model into a student model, provided the samples it generates are differentiable wit respect to its parameters, e.g. GAN generators or even single-step DMs.
The DI article mentioned above does ground the method using the example of GANs and single-step DMs, but with a clear emphasis on the latter which is alsp evident from the provided implementation. In contrast to that, we will here focus particularly on applying DI to finetune a generator obtained with or without adversarial pre-training. As our goal is mainly to reproduce and observe the method ans its implications, we will ground our experiments using a simple yet representative framework. We aim to demonstrate that DI improves such generators, all in all enabling faster generation of higher quality images. Our results are contrasted, but allow us provide insights as pertains to the DI procedure itself. The Sections Section 2 & Section 3 are reminders as to the respective frameworks of DMs and GANs, and can be skipped by the reader. We next present the theoretical framework of DI in Section Section 4 followed by our experiments in Section Section 5, concluding by a discussion in Section Section 6.

2 Background on Diffusion Models

This section is highly inspired by [Ho et al., 2020].

2.1 Forward (Diffusion) Process

2.1.1 Notation and Setup

We consider data points $x_0 \in \mathbb{R}^d$ drawn from a data distribution $q(x_0)$. A forward noising process of length T is defined via a variance schedule $\{\beta_t\}_{t=1}^T$, where each $\beta_t \in (0,1)$. We set

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$$

We consider latent variables $\{x_t\}_{t=1}^T$. The joint distribution of the forward process, given an initial data point x_0, is

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),$$

where

$$q(x_t \mid x_{t-1}) = \mathcal{N}\Big(x_t;\, \sqrt{\alpha_t}\, x_{t-1},\, (1-\alpha_t)\mathbf{I}\Big).$$

2.1.2 Conditional Distributions in the Forward Process

By iterating these Gaussian transitions, one obtains the closed-form marginal:

$$q(x_t \mid x_0) = \mathcal{N}\Big(x_t;\, \sqrt{\bar{\alpha}_t}\, x_0,\, (1-\bar{\alpha}_t)\mathbf{I}\Big).$$

2.2 Reverse (Generative) Process

We aim to learn a reverse model

$$p_\theta(x_{t-1} \mid x_t),$$

so that starting from a standard Gaussian $x_T \sim \mathcal{N}(0, \mathbf{I})$, we can iteratively sample $x_{t-1}, x_{t-2}, \ldots, x_0$. We define

$$p_\theta(x_0, \ldots, x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t),$$

where typically $p(x_T) = \mathcal{N}(0, \mathbf{I})$.

2.3 Deriving the Variational Lower Bound (ELBO)

We want to maximize the log-likelihood $\mathbb{E}_{q(x_0)}[\log p_\theta(x_0)]$. Introducing the latent variables $x_{1:T}$ and the forward process as a variational distribution $q(x_{1:T} \mid x_0)$, we write:

$$\log p_\theta(x_0) = \log \int p_\theta(x_0, x_{1:T})\, dx_{1:T} = \log \int \frac{q(x_{1:T} \mid x_0)}{q(x_{1:T} \mid x_0)}\, p_\theta(x_0, x_{1:T})\, dx_{1:T}.$$

By Jensen's inequality:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\Big[\log p_\theta(x_0, x_{1:T}) - \log q(x_{1:T} \mid x_0)\Big].$$

Defining the negative of this expectation as the variational loss:

$$-\log p_\theta(x_0) \leq \mathbb{E}_{q(x_{1:T}|x_0)}\Big[\log q(x_{1:T} \mid x_0) - \log p_\theta(x_0, x_{1:T})\Big].$$

Taking an expectation over x_0 from $q(x_0)$ gives the full ELBO-based bound on the negative log-likelihood:

$$\mathcal{L} := \mathbb{E}_{q(x_0, x_{1:T})}\Big[\log q(x_{1:T} \mid x_0) - \log p_\theta(x_0, x_{1:T})\Big].$$

2.4 Decomposing the ELBO into $L_0 + L_1 + \cdots + L_T$

$$\begin{aligned}
L(\theta) = \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathcal{L} &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q\left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]
\end{aligned}$$

From this, we can bound $L(\theta)$ with the following decomposition:

$$L(\theta) \leq L_0 + \sum_{t=1}^{T-1} L_t + L_T,$$

where:

$$L_0 = \mathbb{E}_q \left[-\log p_\theta(x_0|x_1) \right],$$
$$L_t = \mathbb{E}_q \left[D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \right],$$
$$L_T = \mathbb{E}_q \left[D_{\mathrm{KL}}(q(x_T|x_0) \| p(x_T)) \right].$$

2.5 Derivation of L_t

Starting from:

$$L_t = \mathbb{E}_q \left[D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \right],$$

and assuming that both $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$ are Gaussian distributions, we define:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \sigma_t^2 I),$$

where $\sigma_t^2 = \frac{\beta_t}{1-\alpha_t}$ is a known variance, β_t and α_t are scalars derived from the forward process, and $\tilde{\mu}_t(x_t, x_0)$ is the mean of the posterior in the forward diffusion process.

The KL divergence for two Gaussian distributions can be computed explicitly as:

$$D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) = \frac{1}{2\sigma_t^2} \mathbb{E}_q \left[\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \|^2 \right] + C,$$

where C is a constant that does not depend on θ.

Thus:

$$L_t = \frac{1}{2\sigma_t^2} \mathbb{E}_q \left[\| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \|^2 \right] + C.$$

2.6 Derivation of the Reverse Law $q(x_{t-1}|x_t, x_0)$

The reverse process is derived from the forward diffusion process using Bayes' rule:

$$q(x_{t-1}|x_t, x_0) \propto q(x_t|x_{t-1}) q(x_{t-1}|x_0),$$

where the forward transition $q(x_t|x_{t-1})$ is defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I),$$

and the marginal $q(x_{t-1}|x_0)$ is:

$$q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}} x_0, (1 - \bar{\alpha}_{t-1}) I),$$

with $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ and $\beta_t = 1 - \alpha_t$.

The posterior $q(x_{t-1}|x_t, x_0)$ is proportional to the product of two Gaussian distributions. Expanding their densities yields:

$$q(x_{t-1}|x_t, x_0) \propto \exp\left(-\frac{\|x_t - \sqrt{\alpha_t} x_{t-1}\|^2}{2\beta_t} - \frac{\|x_{t-1} - \sqrt{\bar{\alpha}_{t-1}} x_0\|^2}{2(1 - \bar{\alpha}_{t-1})} \right).$$

By combining terms in the exponent, the posterior is identified as a Gaussian distribution:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I),$$

where the mean $\tilde{\mu}_t(x_t, x_0)$ and variance $\tilde{\beta}_t$ are obtained by completing the square. Specifically:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t,$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.$$

Let us explicitate the value of $\tilde{\mu}_t(x_t, x_0)$:

$$\begin{aligned}\tilde{\mu}_t(x_t, x_0) &= \tilde{\mu}_t\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t(x_0,\epsilon) - \sqrt{1-\bar{\alpha}_t\epsilon}\right)\right)\\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t(x_0,\epsilon) - \sqrt{1-\bar{\alpha}_t\epsilon}\right) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t\\ &= \frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t,t)\right).\end{aligned}$$

The parameters $\tilde{\mu}_t(x_t, x_0)$ and $\tilde{\beta}_t$ represent the posterior mean and variance, respectively. These are derived directly from the forward diffusion process parameters and ensure that the reverse process faithfully approximates the true posterior distribution.

2.7 Reparameterization and Noise-Prediction Diffusion Training

To simplify the optimization of the reverse process in a diffusion model, we reparameterize μ_θ, the reverse process mean function, as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t, t)\right),$$

where $\epsilon_\theta(x_t, t)$ is a neural network trained to predict the noise ϵ added during the forward diffusion process. This reparameterization allows μ_θ to depend directly on ϵ_θ, making the reverse process resemble a gradient-based update in the data space.

Given the parameterization above, sampling $x_{t-1} \sim p_\theta(x_{t-1} \mid x_t)$ during the reverse process follows:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z,$$

where $z \sim \mathcal{N}(0, I)$ is Gaussian noise and σ_t represents the variance of the diffusion noise at step t. This setup links the reverse process directly to a Langevin-like dynamics, where $\epsilon_\theta(x_t, t)$ serves as the learned gradient of the data density.

To train ϵ_θ, we rely on the fact that at each step t of the forward diffusion process, x_t can be written as:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon,$$

where $\bar{\alpha}_t$ is the cumulative product of the forward noise schedule coefficients α_t, and $\epsilon \sim \mathcal{N}(0, I)$ is the noise added to the clean data x_0. The network $\epsilon_\theta(x_t, t)$ is trained to approximate ϵ, minimizing the difference between the predicted and true noise.

The optimization objective, derived from the variational bound, simplifies into a mean-squared error (MSE) loss:

$$\mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right].$$

Thus, the diffusion model training involves minimizing:

$$\mathcal{L}_{\text{simple}}(\theta) = \sum_{t=1}^{T} w_t \, \mathbb{E}_{x_0,\epsilon} \left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right],$$

where w_t are weighting coefficients that can be set to 1 for simplicity.

2.8 Sampling / Generation

Once trained, to sample from the model:

1. Draw $x_T \sim \mathcal{N}(0, \mathbf{I})$.
2. For $t = T, T-1, \ldots, 1$,

   $$x_{t-1} \sim p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1};\, \mu_\theta(x_t, t),\, \sigma_t^2 \mathbf{I}\big),$$

   where $\mu_\theta(x_t, t)$ is expressed via the network's noise prediction ϵ_θ.

The final x_0 is then a sample that resembles data from $q(x_0)$.

3 Background on Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), introduced by [Goodfellow et al., 2014], represent a class of implicit generative models that have gained significant attention due to their ability to synthesize high-quality data. GANs leverage two neural networks, a *generator* and a *discriminator*, which are trained adversarially in a minimax game setup.

The generator G maps samples from a simple latent distribution p_z (e.g., Gaussian or uniform) to the data space, aiming to approximate the true data distribution p_d. The discriminator D, on the other hand, is a classifier trained to distinguish between real data (drawn from p_d) and synthetic samples generated by G.

3.1 Objective Functions and Training Process

The training objective of a GAN can be expressed as:

$$\min_G \max_D \mathcal{L}(G, D) = \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))].$$

Here, the discriminator D is optimized to maximize the probability of correctly identifying real versus generated samples. Simultaneously, the generator G seeks to minimize the ability of D to distinguish between the two, effectively "fooling" the discriminator. The competition between G and D drives the generator to produce increasingly realistic samples.

3.2 Theoretical Insights and Divergence Metrics

GANs implicitly minimize a statistical divergence between the generated distribution p_g and the true distribution p_d. In its original formulation, GAN training corresponds to minimizing the Jensen-Shannon divergence:

$$D_{JS}(p_d \| p_g) = \frac{1}{2}\Big(D_{KL}(p_d \| m) + D_{KL}(p_g \| m)\Big),$$

where $m = \frac{1}{2}(p_d + p_g)$ is the mixture distribution, and D_{KL} denotes the Kullback-Leibler divergence.

Variants of GANs have since explored alternative divergence metrics, such as Wasserstein GANs, which replace the Jensen-Shannon divergence with the Earth Mover's distance for improved stability during training.

3.3 Challenges in Training GANs

Despite their effectiveness, training GANs remains challenging. Some of the key difficulties include:

- **Mode Collapse:** The generator may map diverse inputs to a limited subset of the data distribution, producing repetitive outputs.
- **Training Instability:** The adversarial setup can result in oscillatory behavior or failure to converge.
- **Vanishing Gradients:** If the discriminator becomes too powerful, the gradients provided to the generator may diminish, hindering its improvement.

Various techniques, such as gradient penalties, architectural modifications, and regularization, have been proposed to address these issues. For instance, WGAN-GP introduces a gradient penalty term to stabilize the training process.

3.4 Applications of GANs

GANs have been successfully applied in diverse domains, including:

1. **Image Generation:** Generating realistic images, such as faces, objects, and scenes.
2. **Data Augmentation:** Enhancing training datasets for machine learning tasks.
3. **Text-to-Image Synthesis:** Generating images conditioned on textual descriptions.
4. **Medical Imaging:** Synthesizing high-quality medical scans for diagnostics and research.

By leveraging the adversarial training framework, GANs have demonstrated exceptional capability in generating high-fidelity data, making them a cornerstone of modern generative modeling.

4 Diff-Instruct: A Universal Approach for Transferring Knowledge From Pre-trained Diffusion Models

4.1 Presentation

Diffusion models have garnered significant attention in generative modeling due to their remarkable stability during training. Unlike adversarial training methods, diffusion models ensure convergence with minimal tuning, making them an appealing choice for researchers and practitioners alike. However, their iterative sampling process inherently results in slow inference times, limiting their practical applications in scenarios where rapid generation is essential.

Conversely, Generative Adversarial Networks represent a class of implicit generative models celebrated for their ability to produce high-fidelity samples. Once trained, GANs offer state-of-the-art performance with swift inference capabilities. Yet, their training process is notoriously challenging, often requiring meticulous balancing between the generator and discriminator to avoid pitfalls like mode collapse or divergence.

Diff-Instruct was introduced in [Luo et al., 2024] and aims to unify the strengths of these two paradigms: the ease of training and stability from DMs, and the efficient inference and high-quality generation from GANs. The central idea is to distill the knowledge encapsulated in a pre-trained DM into a generator model, which can subsequently leverage the learned representations for rapid sample generation without compromising quality.

This approach employs a pre-trained DM as a teacher to guide the generator, ensuring that the latter inherits the superior data distribution knowledge embedded in the DM. By doing so, the generator is optimized to replicate the sampling quality of diffusion models while maintaining the computational efficiency characteristic of GANs. This framework represents a step towards bridging the gap between explicit and implicit generative models, paving the way for scalable and versatile generative solutions.

4.2 Integral Kullback-Leibler Divergence

To align the forward diffusion processes of the pre-trained diffusion model and the auxiliary diffusion model applied to the generator, we introduce the Integral Kullback-Leibler (IKL) divergence. The IKL divergence integrates the Kullback-Leibler divergence across the entire diffusion time horizon, providing a robust criterion for comparing distributions evolving under diffusion processes.

Definition 4.1 (Integral Kullback-Leibler Divergence)**.** *Given two distribution $q^{(0)}$ and $p^{(0)}$, the Integral Kullback-Leibler (IKL) divergence is defined as:*

$$\begin{aligned} D_{IKL}[q^{(0)}, p^{(0)}] &= \int_0^T w(t)\, D_{KL}(q^{(t)} \| p^{(t)})\, dt, \\ &= \int_0^T w(t)\, \mathbb{E}_{x_t \sim q^{(t)}} \left[\log \frac{q^{(t)}(x_t)}{p^{(t)}(x_t)} \right] dt, \end{aligned}$$

where $w(t)$ is a weighting function emphasizing specific diffusion time steps and $(q^{(t)})_{t\in[0,T]}$ and $(p^{(t)})_{t\in[0,T]}$ are the forward diffusion processes with marginal densities $p^{(0)}$ and $q^{(0)}$ at time 0.

The IKL divergence ensures alignment between the auxiliary diffusion model and the pre-trained diffusion model. Minimizing D_{IKL} guarantees that the generator g_θ, which induces the auxiliary diffusion process, learns a representation consistent with the pre-trained model.

To achieve this alignment, we optimize D_{IKL} with respect to the parameters θ of the generator. This requires a closed-form and tractable formula for the gradient of the IKL divergence, enabling efficient optimization. The following corollary provides the required gradient expression.

Corollary 1 (Gradient of the IKL Divergence)**.** *The gradient of $D_{IKL}[q, p]$ with respect to the generator parameters θ is given by:*

$$\nabla_\theta D_{IKL}[q, p] = \int_0^T w(t)\, \mathbb{E}_{z\sim p_z, x_0 = g_\theta(z), x_t \sim p_t(x_t|x_0)} \left[\left(s_q^{(t)}(x_t) - s_p^{(t)}(x_t) \right) \frac{\partial x_t}{\partial \theta} \right] dt,$$

where $s_p^{(t)}$ and $s_q^{(t)}$ are the score functions of the pre-trained diffusion model and the auxiliary diffusion model, respectively.

This corollary provides the key result for optimizing the generator. By minimizing D_{IKL} using its gradient, the forward diffusions from the auxiliary and pre-trained models are aligned as closely as possible, allowing the generator to inherit the high-quality sampling characteristics of the pre-trained diffusion model.

Proof. Recall the definition of $q^{(t)}$, where the sample is obtained by $x_0 = g_\theta(z)$ with $z \sim p_z$, and $x_t \sim p_t(x_t|x_0)$ according to the forward SDE (Equation (2.1)). Since the solution of the forward SDE is uniquely determined by the initial point x_0 and a trajectory of the Wiener process $w_{t\in[0,T]}$, we slightly abuse the notation and let $x_t = \mathcal{F}(g_\theta(z), w)$ to represent the solution of x_t generated by x_0 and w. Let $w_{[0,1]} \sim \mathbb{P}_w$ denote a trajectory from the Wiener process, where $\mathbb{P}_w$ represents the path measure of the Wiener process on $t \in [0, T]$.

The marginal density $q^{(t)}$ implicitly depends on θ since $q^{(t)}$ is initialized with $q^{(0)}$, which is generated by the generator. To emphasize this dependence, we may use $q_\theta^{(t)}$ to represent $q^{(t)}$.

The $p^{(t)}$ is defined through the pre-trained diffusion models with score functions $s_p^{(t)}$. The IKL divergence between $q_\theta^{(t)}$ and $p^{(t)}$ is defined as:

$$\begin{aligned}
D_{\mathrm{IKL}}^{[0,T]}(q_\theta^{(0)}, p^{(0)}) &= \int_0^T w(t) D_{\mathrm{KL}}(q_\theta^{(t)} \| p^{(t)})\, dt, \\
&= \int_0^T w(t)\, \mathbb{E}_{x_t \sim q_\theta^{(t)}} \left[\log \frac{q_\theta^{(t)}(x_t)}{p^{(t)}(x_t)} \right] dt, \\
&= \int_0^T w(t)\, \mathbb{E}_{z \sim p_z,\, x_0 = g_\theta(z),\, w \sim \mathbb{P}_w} \left[\log \frac{q_\theta^{(t)}(\mathcal{F}(g_\theta(z), w))}{p^{(t)}(\mathcal{F}(g_\theta(z), w))} \right] dt.
\end{aligned}$$

Taking the gradient of the IKL divergence with respect to θ, we get:

$$\begin{aligned}
\frac{\partial}{\partial \theta} D_{\mathrm{IKL}}^{[0,T]}(q_\theta^{(0)}, p^{(0)}) &= \int_0^T w(t)\, \mathbb{E}_{z \sim p_z,\, w \sim \mathbb{P}_w} \frac{\partial}{\partial \theta} \left[\log \frac{q_\theta^{(t)}(\mathcal{F}(g_\theta(z), w))}{p^{(t)}(\mathcal{F}(g_\theta(z), w))} \right] dt \\
&= \int_0^T w(t)\, \mathbb{E}_{z \sim p_z,\, w \sim \mathbb{P}_w} \nabla_{x_t} \left[\log \frac{q_\theta^{(t)}(x_t)}{p^{(t)}(x_t)} \right] \frac{\partial x_t}{\partial \theta}\, dt \\
&\quad + \int_0^T w(t)\, \mathbb{E}_{x_t \sim q_\theta^{(t)}} \frac{\partial}{\partial \theta} \log q_\theta^{(t)}(x_t)\, dt.
\end{aligned}$$

The first term corresponds to:

$$A = \int_0^T w(t)\, \mathbb{E}_{x_t \sim q_\theta^{(t)}} \left[s_q^{(t)}(x_t) - s_p^{(t)}(x_t) \right] \frac{\partial x_t}{\partial \theta}\, dt,$$

where $s_p^{(t)}$ and $s_q^{(t)}$ are the score functions of the pre-trained and auxiliary diffusion models, respectively.

The second term vanishes because:

$$\begin{aligned}
B &= \int_0^T w(t)\, \mathbb{E}_{x_t \sim q_\theta^{(t)}} \frac{\partial}{\partial \theta} \log q_\theta^{(t)}(x_t)\, dt \\
&= \int_0^T w(t) \int \frac{\partial}{\partial \theta} q_\theta^{(t)}(x_t)\, dx_t\, dt \\
&= \int_0^T w(t) \frac{\partial}{\partial \theta} \int q_\theta^{(t)}(x_t)\, dx_t\, dt \\
&= \int_0^T w(t) \frac{\partial}{\partial \theta} 1\, dt = 0.
\end{aligned}$$

Thus, the gradient simplifies to:

$$\nabla_\theta D_{\mathrm{IKL}}^{[0,T]}(q_\theta^{(0)}, p^{(0)}) = \int_0^T w(t)\, \mathbb{E}_{x_t \sim q_\theta^{(t)}} \left[s_q^{(t)}(x_t) - s_p^{(t)}(x_t) \right] \frac{\partial x_t}{\partial \theta}\, dt.$$

This concludes the proof. □

4.3 Algorithm

The Diff-Instruct algorithm alternates between two optimization steps: updating the marginal score function s_ϕ and updating the generator g_θ. These updates are performed using stochastic gradient descent (SGD). In the Algorithm 1 below, we clearly describe how each term in the gradients is computed and provide the formal procedure. The convergence is characterized by $s_\phi(x_t,t) \approx s_{p^{(t)}}(x_t)$, i.e. when $\mathrm{Grad}(\theta) \approx 0$.

```
Algorithm 1: Diff-Instruct Algorithm
  Input: Pre-trained diffusion model scores $s_{p(t)}$, generator $g_\theta$, prior distribution $p_z$, initialized
         auxiliary diffusion model $s_\phi$, forward diffusion process $p_t(x_t \mid x_0)$.
  Output: Trained generator parameters $\theta$ and auxiliary model parameters $\phi$.
  while not converged do
      Step 1: Update the auxiliary score model $s_\phi$
      Compute the gradient:
          $\mathrm{Grad}(\phi) = \frac{\partial}{\partial \phi} \int_0^T w(t)\, \mathbb{E}_{z\sim p_z, x_0=g_\theta(z), x_t\sim p_t(x_t|x_0)} \left\| s_\phi(x_t,t) - \nabla_{x_t} \log p_t(x_t \mid x_0) \right\|_2^2 \, dt.$
      Explanation:
          • Sample $z \sim p_z$ from the prior distribution.
          • Compute $x_0 = g_\theta(z)$ using the generator.
          • Simulate $x_t$ using the forward diffusion $p_t(x_t \mid x_0)$.
          • Evaluate the score difference $s_\phi(x_t,t) - \nabla_{x_t} \log p_t(x_t \mid x_0)$.
          • Minimize the weighted squared error $\| \cdot \|_2^2$ over all $t \in [0,T]$.
      Update $\phi$ using SGD.
      Step 2: Update the generator $g_\theta$
      Compute the gradient:
          $\mathrm{Grad}(\theta) = \int_0^T w(t)\, \mathbb{E}_{z\sim p_z, x_0=g_\theta(z), x_t\sim p_t(x_t|x_0)} \left[ s_\phi(x_t,t) - s_{p(t)}(x_t) \right] \frac{\partial x_t}{\partial \theta} \, dt.$
      Explanation:
          • Sample $z \sim p_z$ and compute $x_0 = g_\theta(z)$.
          • Simulate $x_t$ using the forward diffusion $p_t(x_t \mid x_0)$.
          • Compute the difference between the auxiliary score $s_\phi(x_t,t)$ and the pre-trained score $s_{p(t)}(x_t)$.
          • Compute the Jacobian $\frac{\partial x_t}{\partial \theta}$ to propagate the gradient through $g_\theta$.
      Update $\theta$ using SGD.
  return $\theta, \phi$.
```

5 Experiments

Even though the authors of DiffInstruct [Luo et al., 2024] present an application of the method to finetune StyleGAN, their work mainly focuses on single-step DMs. We will here go over the implementation of DI from scratch to finetune a GAN generator using a more powerful DM. After setting the framework, we will go over the pre-training of the models. From there we present our DI implementation, its training and it results. To assess of the quality of generated data beyond the qualitative human appreciation, we used the precision, recall and kernel inception distance (KID). We will elaborate on his choice when presenting our quantitative results. All the code necessary to replicate our experiments, as well as the models we trained, are available on this repository.

5.1 Experimental framework

We will here try and sample data using FashionMNIST as pre-training set for both our DM and our generator. This dataset holds 60k grayscale images of clothing articles from 10 classes, all in $(1 \times)28 \times 28$ resolution. We chose such a simple dataset to allow for fast training and inference as well as memory efficiency, without compromising the difficulty of the task. This dataset still allows to demonstrate generative models' capabilities in a fair way. A small sample of true data can be seen in Fig. 1.

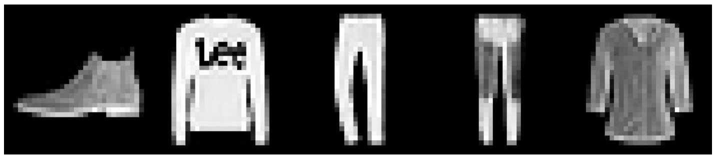

Figure 1: 5 true images from FashionMNIST

All the experiments described below were done using a 8GB laptop GPU and using a batch size of 128.

5.2 Pre-training

DiffInstruct relies on a powerful but slow-sampling DM distilling its knowledge into a faster-sampling originally weaker generator. To that end, we started by training a - satisfying - DM on MNIST, then adversarially pre-trained a - worse - generator.

5.2.1 Diffusion Model

Our DM relies on a UNet [Ronneberger et al., 2015] backbone. It incoporates 4 layers, 128 hidden channels and 2000 noising/denoising steps. We ended up with these hyperparameter after some manual tuning, and found this produced the best model quality - training speed ratio. The model was pretrained in around 1 hour over 110 epochs, training being stopped after an early stopping procedure was triggered, which relied on 20% of the training set (12k samples). We optimized for a Huber loss as reconstruction criterion. A few samples are provided in Fig. 2.

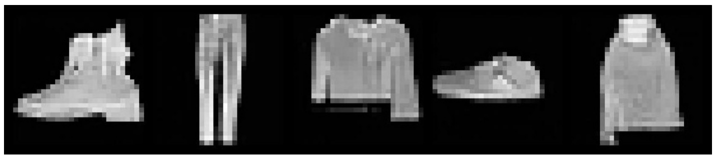

Figure 2: 5 images sampled from our DM

Qualitatively speaking, these are not top shelf samples. The model can also produce even worse ones. However it has to be underlined that the objects are still very recognizable - in their shape and contrast - which will likely be enough to finetune a generator downstream.

5.2.2 Generative Adversarial Networks

We trained a generator to reconstruct $1 \times 28 \times 28$ images from a latent space of dimension 128. It incorporates a shrinking convolutional architecture of 3 layers (dividing the dimensions by 2 in

between layers, finally projecting onto the desired resolution). The adversary discriminator has a mirror architecture.
After a few training runs we observed that such generator could quickly (within 15 epochs) match if not outperform or DM. In order to counteract that, we purposefully trained it for 3 epochs only. As it can be observed in Fig. 3, this imprecise generator produces very poor images although these are arguably not too noisy and still allow to distinct what can be qualitatively interpreted as approximation of real objects. All in all, we chose to move forward with this generator, estimating that it would be a good candidate to be our student model.

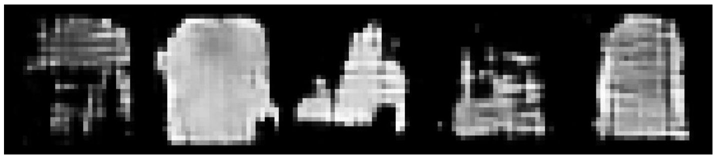

Figure 3: 5 images sampled from our pre-trained generator

5.3 DiffInstruct

5.3.1 Implementation

To implement the sequential training procedure that is DI, we strongly inspired ourselves from the original DI repository. We implemented a way for us to track the training logs (the loss associated to the implicit diffusion model ϕ as well as the gradient of the generator's parameters θ). We implemented the option to do several updates of ϕ or θ within the same round. Round here refers to a loop where both models are fully updated the desired number of time. By default as described in Algorithm 1 just incorporates a single update of both models.
Some lines of code that were present in the original repository were surprising. We are referring to curious sequence of freezing and unfreezing of the various models, sometimes being redundant with one another. We tried ablating such parts and did not observe major changes in the training procedure. In the final code we provide we kept our implementation as close as the original one, of course adapting it - as said code was meant to distill knowledge into single step DMs. We also implemented a way for us to save models after a select amount of rounds in order to somewhat keep track of the evolution of the training.

5.3.2 Distillation

The training itself proved to be challenging. This might partially be due to the training strategy itself. Indeed, it's sequential aspect might remind the behavior of an adversary procedure - while of course not being one. What we mean is that an update of ϕ relies on a provided generator, that changes as a consequence θ, itself moving ϕ slightly off the steepest parameter descent path, and so oin and so forth. This is observed in the very rough convergence of the loss associated to ϕ, see Fig. 4.
Now what we observed with various learning rates and various "round update strategies" (i.e. operating several updates of one and/or the other model within the same round) is that typically, the convergence as described in [Luo et al., 2024] does not occur. The implicit DM always converges first, before oscillating while Grad(θ) oscillates, typically far from 0, positvely or negatively. We do not have a satisfying explanation for it. Even after long (¿40 min) training procedures - with and without learning rate decay w.r.t the ϕ updates. This was also observed when updating one model more than the other one.
Now we will further discuss what happens progressively during the DI procedure.

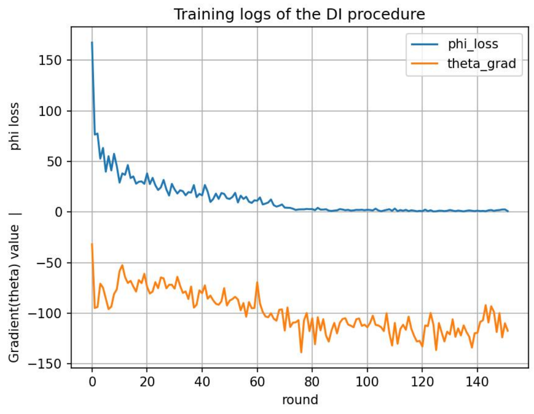

Figure 4: Training logs for DI (default)

5.4 Results

We will of course comment on the sampled images qualitatively as we already did. However to quantify our results, we will make use of the following metrics:

- The precision, to assess how the generated samples fall in the real samples' manifold; along with the recall, that quantifies how well the real samples can themselves be reconstructed by the generative model.
- The kernel inception distance (KID). Computing the maximum mean discrepancy on the features (from both real and generated samples) passed through a polynomial kernel, this metric quantifies the quality of the data capturing how close they are in the kernel space. It also can highlight mode collapse behavior. A value closer to 0 means a better KID.

We evaluate these metrics using 100 sampled images from each model. Our baseline metrics can be observed in Table 1.

	Precision	**Recall**	**KID**
DM	0.71	0.59	0.030831
pretrained generator	0.60	0.48	0.099935
real data (theoretical)	1	1	0

Table 1: Performance metrics comparison

To better grasp the procedure's behavior, we finetuned our generator with DI and saved a model every 15 epochs over 150 epochs. From there we sampled and evaluated the data. The metrics can be observed in Fig. 5 and a visualization of sampled data is provided in Fig. 6.
We clearly see both qualitatively and quantitatively that our implementation of DI gradually degrades the generator. Even though these metrics and data points were obtained from a single run, they fit the general pattern we observed trying a wide variety of hyper parameters and update strategies. We also tried a variety of pre-trained generators with different levels of quality as is. The only noticeable metric improvement. It is somewhat disappointing to not observe DI in action as described in the original paper. We will discuss these results in the final section.
One element that was observed - unsurprisingly so - was the difference in sampling times. Our DM model averages 2.31sec per generation, while our generator could sample 100 images in under a tenth of a second.

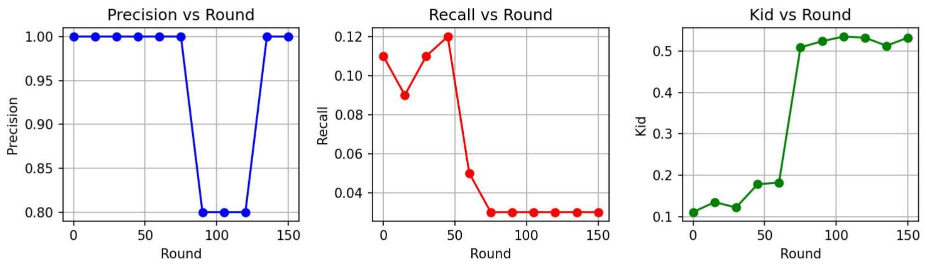

Figure 5: Scores of the DI generator as training goes

6 Discussion

Despite the strong theoretical analysis and results presented in [Luo et al., 2024], we couldn't transpose the results of DI to our miniature framework. We struggle to explain why the convergence in our implementation does not work. We do have a few leads.
It may very well be that DI is supposed to work on much more advanced model, e.g. the example closest to our implementation from the paper refers to the finetuning of StyleGAN [Karras et al., 2019], a very advanced and heavily pre-trained model. If this hypothesis is right, it would mean that the framework we set up stand no chance to emulate the results presented in the paper.
Another plausible explanation could be an error in the implementation, either stemming from us when transcribing and adapting it or the original repository itself. However, this appears somewhat unlikely to us as we did try out a lot of configuration and alternative implementations.
It is disappointing to us not to have been able to make it work even after extensively exploring strategy that seemed wise. All in all, these results made us question the Algorithm 1, but after thorough investigation we can't seem to find the root of our observations. DI is undoubtedly a powerful strategy, but does require a challenging training procedure that we couldn't quite nail.

References

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[Ho et al., 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

[Karras et al., 2019] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks.

[Luo et al., 2024] Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., and Zhang, Z. (2024). Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

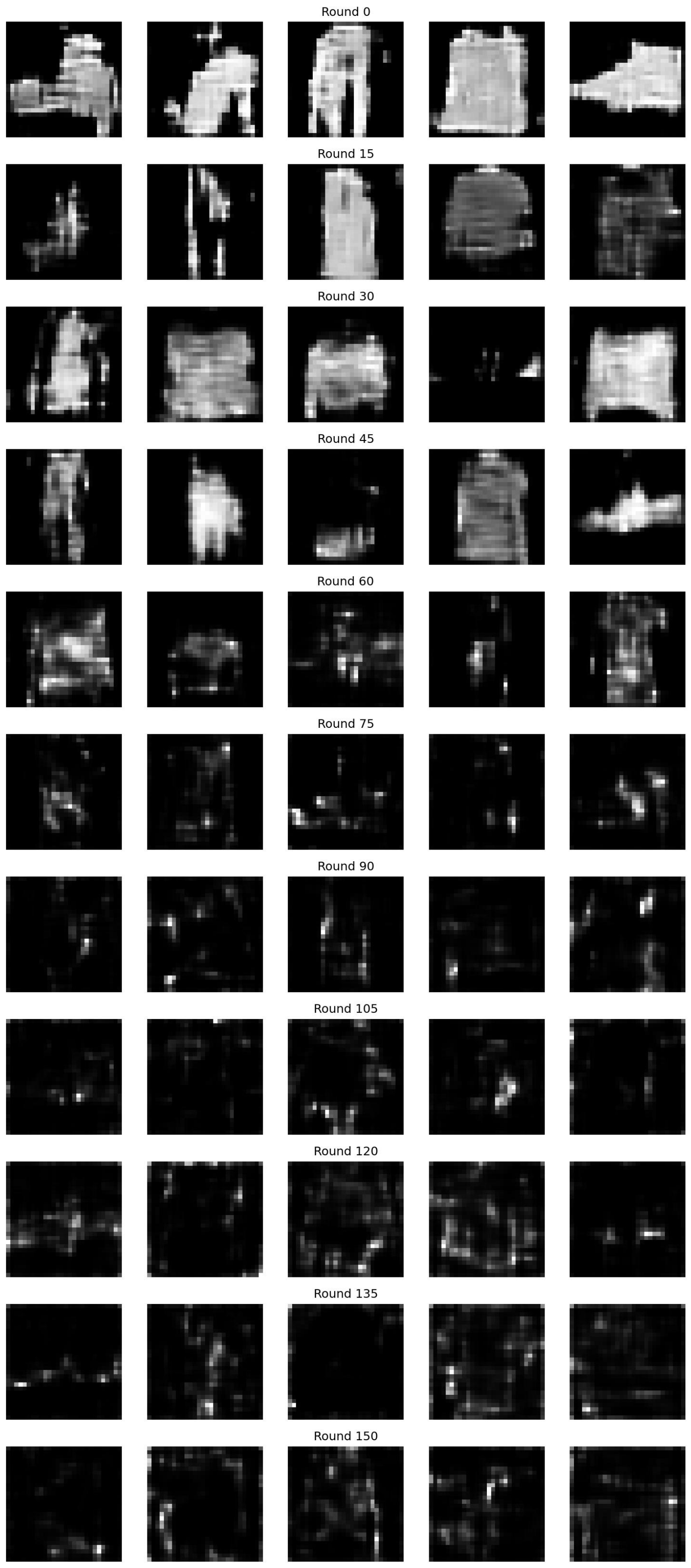

Figure 6: Visualization of the samples as training goes