Stochastic Differential Equations for Generative Modeling

Anton Bolychev[1*]

Tuesday 2nd May, 2023

Abstract

In this work we studied score-based generative modelling with diffusion probabilistic modelling. Generation of images was modelled as stochastic process and was discretized with Score matching with Langevin dynamics (SMLD) and Denoising diffusion probabilistic modeling (DDPM) techniques. Performances of these methods were compared to Energy Based GAN (EBM GAN) for which we performed sampling in latent space of Generator as diffusion process. Frechet Inception Distance metric was employed as performance criterion.

Github repo: ML SDE

1. Introduction

Generative modelling is an important task in machine learning. The field was traditionally dominated by Generative Adversarial Networks, Autoencoders, and Flow Based models. However, recently diffusion models gained significant popularity and showed ability to beat GANs in image generation. One of the examples of a successful practical application of the method is DALL-E 2 by OpenAI.

Score-based generative modeling is a subclass of diffusion models which take inspiration from thermodynamics process and involves Stochastic Differential Equations (SDEs). These methods sequentially corrupt data with noise and then try to synthesize objects of interest from the noise.

One drawback of this method is very large number of steps in both training and inference modes as we need to compute SDE numerically with a sufficiently small discretization step (or apply some numerical SDE solver) - it is an especially computationally intensive task in case of high-resolution images. Hence, reproducibility is a major problem in this field as diffusion generative models are notoriously difficult to train. Another thing to consider is that results heavily depend on technical debt and specific technical solutions (Engstrom et al., 2020) as many papers either do not contain code or the implementation of the code gives different results.

Another difficulty is the relative unease of sampling for diffusion stochastic processes. Diffusion models produce continuous probability distributions which makes it computationally expansive to sample from them. Additionally, diffusion models often involve complex and nonlinear dynamics, which further complicates the sampling process. This makes it challenging to generate realistic samples because the model needs to be iteratively evaluated at every time step, and the likelihood of the data needs to be calculated for each sample, increasing the computational cost. This problem can be tackled with MCMC with Langevin dynamics and this approach can provide a fast and efficient sampling.

It is also possible to employ MCMC Langevin dynamics for the improvement of image generation with GANs. Adversarial generative process can be reformulated in an Energy-based setting and then Langevin dynamics for the latent space of the generator can be used to correct generator samples.

1.1. Main Contribution

- For the reproduction task, we compared two SDE solvers for generative modelling - ancestral sampling and reverse diffusion in terms of achieved Frechet Inception Distance (FID) on the CIFAR10 dataset. We tested the validity of the hypothesis if the usage of the predictor-corrector method would improve the performance of the generative model. To obtain performance metrics we built a pipeline and reproduced results from (Song et al., 2020) work and compared the author's results to ours.

- In the research task, we used pre-trained GAN, implemented the Energy function, and studied Energy based problem setting. We implemented the Langevin Dynamics sampling procedure from (Che et al., 2021) and observed how MCMC in the latent space changed the FID score of the generator. We obtained results as a graph and compared them to other papers.

* [1] Moscow State University, Moscow 143026, Russia.

1.2. Related works

Diffusion Models

Recently, diffusion models become an increasingly popular approach and achieved state-of-the-art results in image generation. Here are several papers that employed diffusion models. In (Song et al., 2020), the authors proposed a method for generating images from noise using a forward SDE function for noising an image and a backward SDE function for denoising an image. Meanwhile, an image was produced at random from noise. They combined different approaches and achieved record-breaking performance with an Inception score of 9.89 and FID of 2.20.

Another group of authors (Ho et al., 2020) demonstrated high-quality picture synthesis results utilizing diffusion probabilistic models, a type of latent variable model motivated by nonequilibrium thermodynamic considerations. Training on a weighted variational bound constructed according to a novel connection between probabilistic diffusion models and denoising score matching with Langevin dynamics yields the best results.

In (Lu et al., 2022) work authors used the notion of stability of ODEs and achieved 4.70 FID in 10 function evaluations and 2.87 FID in 20 function evaluations on the CIFAR10 dataset while performing calculations approximately 4-16 times faster. The main idea of the work is to consider the diffusion model as partly linear model and non-linear, learned by a neural network. The linear part thus can be computed analytically and the non-linear part can be approximated with great precision.

(Che et al., 2021) demonstrates that DDLS is more efficient in comparison to other methods that work in high-dimensional pixel space and may be used to improve on previously trained GANs of various varieties. Authors provide qualitative and quantitative evaluations of DDLS using synthetic and real-world datasets.

Generative models overview

There exist several approaches of generation models. A survey on generating models (Che et al., 2021) states these models as the most predominant :

- Generative Adversarial Net (GAN) (Goodfellow et al., 2014) trains generator to produce images from samples of latent space and in parallel trains an adverse model-discriminator to distringuish lifelike from generated images.

- Energy-Based Model (EBM) (Goodfellow et al., 2016) designs designs a suitable energy function for pairwise energy matching between conditions and samples, similar to a generative discriminator in GAN.

- Variational Auto-Encoder (VAE) (Kingma & Welling, 2022) employs projection onto latent space from which decoder than samples. The projection uses dimentionality reduction techniques.

- Normalizing flow (NF) (Papamakarios et al., 2021) uses idea that resembles that of SDE: a flow function gradually noises an image to latent space and specially designed reverse function can restore an image from the noise.

- Diffusion model gradually injects noise into the original data until it turns to the known noise distribution before reversing each step in the sampling steps.

FID Score usage

Frechet Inception Distance is widely used metric in comparing performance of generative models. It was used in (Tran et al., 2019) work on GANs and in (Munjal et al., 2019) for Variational Encoders.

2. Preliminaries

As we mentioned in the beginning, to solve backward stochastic differential equations, we used two different SDE solvers, namely, reverse diffusion and ancestral sampling solvers in combination with Langevin dynamics in a replication task. The goal was to replicate paper results (Song et al., 2020).

The sampling from the energy-based model implemented with the Langevin dynamics process was the second component of the final project.

So Langevin dynamics procedure was used for different goals in both sections of the project. The advantages of its applications and how it was used in the project will be discussed in the next paragraphs.

2.1. Langevin Dynamics

Langevin dynamics is a powerful tool for sampling from complex probability distributions. It is a stochastic process that simulates the motion of a particle in a potential energy landscape, subject to random forces and friction. In sampling applications, the Langevin dynamics algorithm is used to generate samples from a target probability distribution. The algorithm starts with an initial position x_0 and iteratively updates the position using the Langevin equation. After a sufficient number of iterations, the resulting sequence of positions represents a sample from the target distribution.

The key advantage of Langevin dynamics is that it can efficiently explore complex high-dimensional probability distributions with non-convex or multimodal shapes. By

introducing stochasticity and friction into the dynamics, it can overcome local minima and explore multiple modes of the distribution. Additionally, by tuning the parameters of the Langevin equation, such as the diffusion coefficient and the strength of the noise term, it is possible to control the exploration behavior and balance between exploration and exploitation.

2.2. Replication task background

As train dataset we have number of images. We can gradually apply stochastic noising process to an image and obtain several grades of noised image resulting in a completely noised image. By applying this transformation to every picture in the train dataset we can collect examples of noisy images from which we can collect a prior distribution from which we then can generate samples and use these samples as a seed for generation procedure. Generative process is a gradual denoising from grade to grade for determined number of steps until we obtain in image which resembles real images.

Langevin dynamics is utilized in this section to correct the denoising operation. We used Langevin dynamics not only for reverse time SDE, but also in conjunction with SDE solvers. As a result, it is critical to understand how Langevin dynamics affects stochastic differential equation solutions.

2.3. Research task background

There is a remarkable work (Che et al., 2021) that paves a way to an alternative interpretation of GANs that can be used to achieve a better quality of sampling. It states that GANs can be treated as an Energy-based model. All aforementioned difficulties in sampling apply to this formulation and MCMC with Langevin dynamics can be a solution here as well. However, it can be more time consuming and the result quality might be questionable. So in this work we provide a proof of concept which demonstrates the viability of this idea.

2.4. Fréchet Inception Distance

One of the metrics to evaluate quality of generated samples that has gained popularity in recent years is the Fréchet Inception Distance (FID). This metric compares the distribution of real data with the distribution of generated data using the activations of a pretrained Inception V3 neural network. In contrast to others, the FID is fast to compute, robust to noise, and correlates well with human judgment of sample quality.

The FID measures the distance between the distributions of real and generated data in feature space and is defined as follows:

$$d_F(\mu,\nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x-y\|^2 \, \mathrm{d}\gamma(x,y) \right)^{1/2},$$

where $\Gamma(\mu,\nu)$ is the set of all measures on $\mathbb{R}^n \times \mathbb{R}^n$ with marginals μ and ν on the first and second factors respectively. (The set $\Gamma(\mu,\nu)$ is also called the set of all couplings of μ and ν.). In other words, it is the 2-Wasserstein distance on $\mathbb{R}^n$.

This definition is difficult to use, however there is a closed form FID expression for multivariate normal distributions:

$$d_F(\mathcal{N}(\mu,\Sigma),\mathcal{N}(\mu',\Sigma'))^2 = \|\mu-\mu'\|_2^2 + \mathrm{tr}\left(\Sigma+\Sigma'-2\left(\Sigma^{\frac{1}{2}}\cdot\Sigma'\cdot\Sigma^{\frac{1}{2}}\right)^{\frac{1}{2}}\right)$$

where μ_1 and μ_2 are the mean vectors of the real and generated data distributions, respectively, and Σ_1 and Σ_2 are their covariance matrices. $Tr(\cdot)$ denotes the trace of a matrix and $||\cdot||_2$ denotes the Euclidean norm.

This FID formula measures the distance between the two multivariate normal distributions by comparing their mean vectors and covariance matrices. A smaller FID score indicates that the generated data is closer to the real data distribution. To compute theses statistics the activations of a pre-trained Inception V3 neural network are used. This network extracts features from the real and generated data distributions to find feature-space statistics instead of distributions of images themselves.

3. Score-based generative modeling through stochastic differential equations

3.1. Replication task pipeline

The task was to replicate a table 1 from paper (Song et al., 2020) results for SMLD and DDPM.

In general, the full pipeline boils down into the following steps:

- Train the model. For images noising procedure we apply forward-time SDE considering images from the dataset as initial conditions for the SDE we compute. By doing so for each image, we obtain corresponding noisy samples that has some prior distribution.
- This prior distribution can be used then to sample noisy images that will be used as initial condition for reverse-time SDE, the solution of which is essentially a generation procedure itself. It's important to mention that not only images can be generated but also video, audio etc.

We do not need to implement the first step and train the model because we used a pretrained model.

To solve backward stochastic differential equations, we used two different SDE solvers proposed in paper combined with

Langevin dynamics.

3.2. SMLD and DDPM overview

To use trained data, we need to sample the dataset. DDPM and SMLD are sampling methods that are score-based models. As a result, the following definitions must be considered:

- Score-based models are a class of machine learning models that directly learn the probability distribution of the data, rather than learning a discriminative function or a generative model. In score-based models, the goal is to learn a function that assigns a score to each data point, which represents the log-likelihood of that data point under the learned distribution.
- Sampling is used to select a subset of data from a larger dataset for training or testing purposes. It is important in machine learning to prevent overfitting, reduce computational costs, and improve the generalization performance of the model.

Both SMLD and DDPM are powerful techniques that are used to model the statistical properties of images and learn the probability distribution of image data.

SMLD is a method for estimating the probability distribution of a set of data points. It involves computing the gradient of the logarithm of the probability density function, known as the score function, and using it to iteratively adjust a set of parameters until the model distribution matches the data distribution. Langevin dynamics is then used to sample from the resulting model distribution.

DDPM, on the other hand, is a method for modeling the probability density function of images by applying a series of noise reduction and diffusion operations to the input image. The resulting model can then be used to generate new images by sampling from the probability distribution.

3.3. Denoising score matching with langevin dynamics (SMLD)

(Song et al., 2020) proposed using a weighted sum of denoising score matching targets to train a Noise Conditional Score Network (NCSN):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})} \mathcal{L}(\boldsymbol{\theta}, \sigma_i, \mathbf{x}, \tilde{\mathbf{x}}) \quad (1)$$

where

$$\mathcal{L}(\boldsymbol{\theta}, \sigma_i, \mathbf{x}, \tilde{\mathbf{x}}) := \left\| \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \right\|_2^2$$

(Song et al., 2020) run M steps of Langevin MCMC to obtain a sample consecutively:

$$\mathbf{x}_i^m = \mathbf{x}_i^{m-1} + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}^*}\left(\mathbf{x}_i^{m-1}, \sigma_i\right) + \sqrt{2\epsilon_i}\mathbf{z}_i^m, \quad (2)$$

where $m = 1, 2, \cdots, M$

3.4. Denoising diffusion probabilistic models (DDPM)

(Che et al., 2021) consider the subsequent series of positive noise scales. A discrete Markov chain is built for each training data point. In the converse manner, a variational Markov chain is parameterized as $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$, and trained with a re-weighted variant of the evidence lower bound:

To obtain the best model samples, begin with x_N and follow the estimated reverse Markov chain as following:

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}}\left(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i)\right) + \sqrt{\beta_i}\mathbf{z}_i, \quad i = N, N-1, \cdots, 1 \quad (3)$$

The noise scales are prescribed as x_N, which are roughly distributed according to the following equation:

$$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (4)$$

3.5. Score-Based Generative modeling with SDEs

Background

The key idea behind generative modelling applying machinery of stochastic differential equations can be boiled down as follows:

1. Considering initial data samples as initial conditions for a given SDE, simulate this SDE within a time interval $[0, T]$ applying any SDE solver, so that $x_0 \to x_T$
2. Train a neural network $\mathbf{s}_{\boldsymbol{\theta}}$ to approximate $\nabla_x log\ p_t(x)$ (will be referred to as *score function*)
3. Generate new samples from distribution $p_0(x)$ assuming that all data was generated from distribution on some vector space (e.g. pixel space in case of pictures) by solving a reverse-time SDE of the form

$$dx = [f(x,t) - g^2(t)\nabla_x log\ p_t(x)]dt + g(t)d\bar{W}, \quad (5)$$

assuming that the score function $\nabla_x log\ p_t(x)$ was approximated by $\mathbf{s}_{\boldsymbol{\theta}}$ with sufficient quality.

The very first part of this algorithm is referred to as *perturbing data*. The last one is *a solution of reverse-time SDE* or, more briefly, a *generation procedure*.

Let us denote true distribution of data as p_{data}, Perturbation kernel $p(\tilde{x}, x)_\sigma = N(\tilde{x}; x, \sigma^2 I)$ and altered data distribution p_σ. Now, we are ready to describe the full noising and denoising routines.

Forward-time SDE Forward-time SDE is used for noising images, which is presented as following:

$$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w \tag{6}$$

We take this equation and implement noising y applying this procedure (from 3.6 - key idea behind) Simmulate stoh diffur is the sense for image noising.

Reverse-time SDE

By applying continuous stochastic transformation one can noise an image and transform it is known prior. Let us set x to be in image and $dx = f(x,t)dt + g(t)dW$ be stochastic noising transformation SDE. It is possible to reverse this transformation and obtain an image from prior. The inverse transformation (Anderson, 1982) is given by:

$$dx = [f(x,t) - g^2(t)\underline{\nabla_x \log p_t(x)}]dt + g(t)d\bar{W}. \tag{7}$$

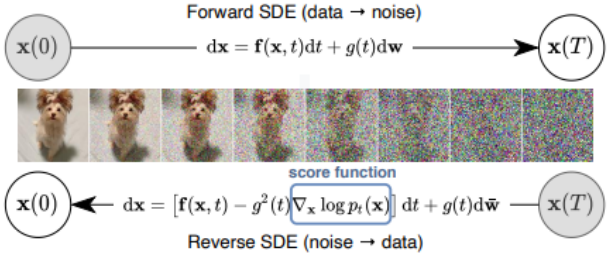

Figure 1. Diagram of transformation of an image to prior and restoration from prior with reverse-time SDE, (Song et al., 2020)

The Score is defined as $\nabla_x \log p_t(x)$ and it is to be trained by neural network by learning to reverse the transformation on train images.

Then our goal is to approximate $\nabla_{\tilde{x}} log\ p_{\sigma_i}(\tilde{x}|x)$ with $s_\theta(x, \sigma_i)$:

4. Energy based GANs

4.1. EB GANs Background

Another more classical way to design a structural generative prediction is using Generative Adversarial Networks. The training processes of this powerful class of models can be defined as adversarial game between generator model and discriminator model in which the discriminator is a critic that measures ability of generator to produce samples from desired distribution. Usually, the discriminator is a classifier that is tasked with distinguishing real samples from true distribution and synthesized products of generative model.

This intuition can be implemented as follows:

$$L_D = -\mathbb{E}_{x\sim p_{data}}[\log D(x)] - \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z))]$$
$$L_G = -\mathbb{E}_{z\sim p_z}[\log D(G(z))]$$

Where L_D is discriminators loss functional to minimize and L_G is the generators loss functional.

Generally, there is no guarantee that the min-max game of the generator and Discriminator will converge to the optimal generator. This is why it is promising to let the critic operate differently: instead of solving a classification task, it can assign a score to the generator in the form of an Energy function. The Energy function is a measure of how closely generated objects resemble samples from the true distribution. The generator is trained to produce samples that have high energy scores, thereby improving the quality of generated samples. Unlike traditional GANs, EBGANs do not require a discriminator to classify the generated samples as real or fake. Instead, the generator is trained solely based on the energy score assigned by the critic. EBGANs have shown promising results in generating high-quality images, audio, and video data.

Formally, let us introduce the generator $G : \mathcal{Z} \to \mathcal{X}$. Latent space of the generator $\mathcal{Z}$ is equipped with prior distribution $p_0(z)$. Out goal is to gradually improve distribution $p_t(z)$. p_d - data distribution and p_g - generator distribution. The generator include distribution $p_d^* = G \circ p_t$. If out setup reaches optimally, then $p_d^* = d_d$ meaning that induced distribution matches true data distribution. Next we introduce a critic model that will measure quality of the generator.

Consider the discriminator D and logits of the discriminator as d. Assuming that $D \approx D^*$ meaning that the discriminator is near optimally then it holds:

$$D(x) \approx \frac{p_d(x)}{p_d(x)+p_g(x)} = \frac{1}{1+p_g(x)/p_d(x)} \approx \frac{1}{1+\exp(-d(x))}.$$

Rearranging terms we obtain $p_d(x) \approx p_g(x)e^{d(x)}$ and with normalization we get $p_d^*(x) = p_g(x)e^{d(x)}/K$ which is Energy-based formulation. In this formulation distribution p_d^* is hard to sample from as $p_g(x)$ is defined implicitly and cannot be computed directly while MCMC in $\mathcal{X}$ is impractical due to high dimentionality.

This observation gives us an opportunity to treat GAN as so-called an energy-based model (EBM) is defined by a Boltzmann distribution $p_t(z) = e^{-E(z)}/K$, where $E(z) : \mathcal{Z} \to R$ is the energy function and K is normalizing

constant. In order to sample from this distribution MCMC with Langevin dynamics is used.

4.2. Sampling from Energy Based GAN

In energy formulation of GAN problem from (Che et al., 2021) generator and discriminator collectively learn implicit energy-based model. Authors formulated energy over generator's latent space in simple tracable from.

The task of sampling from distribution $p(z)$ can be solved by a Langevin dynamics MCMC algorithm :

$$z_{i+1} = z_i - \epsilon/2\nabla_z E(z) + \sqrt{\epsilon}n, n \sim N(0, I) \quad (8)$$

We denote z sample from latent space of Generator, $x = G(z)$ is the batch of generated images. Here sample $z \sim p_t$ and $x = G(z) \sim G \circ p_t$. At the start of the sampling process $t = 0$, $z \sim p_0$ which is starting latent distribution of the Generator.

The Discriminator $D(x)$ and logit of Discriminator are as follows: $d(x)$: $D(x) = \sigma(d(x))$. Then the Energy function is defined as: $E(z) = -\log p_0(z) - d(G(z))$.

In this simple form MCMC algorithm with Langevin dynamics is employed to obtain high-quality samples from GAN.

Algorithm 1 Langevin Dynamics Sampling

Input: N$\in \mathbf{N}_+$, $\epsilon > 0$
Output: Latent code $z_N \sim p_t(z)$
Sample $z_0 \sim p_0(z)$
for $i = 1$ **to** N **do**
  $n_i \sim N(0, 1)$
  $z_{i+1} = z_i - \epsilon/2\nabla_z E(z_i) + \sqrt{\epsilon}n_i$
end for

Weights ϵ decay for every n interations.

5. Experiments and Results

Code for our experiments can be found here: ML SDE

5.1. Replication

In this task we run experiments with reversed stochastic process (Predictor) and Langevin Dynamics (Corrector). We run predictors with different number of steps, meaning that for each step we would consider a level of noisiness and predictor would make one denoising step from one level of noise to another. Step of corrector is one simulation of Langevin dynamics.

In total we run following experiments:

- Predictor only: We use only reversed stochastic process from sample from prior distribution to generated image. We run experiments for 1000 and 2000 steps.
- Corrector Only: 1000 and 2000 runs of Langevin dynamics
- Predictor Corrector: 1000 times of predictor denoising step and after each predictor step a correction with Langevin dynamics.

We used pretrained models from Yang Song's GitHub and run several experiments. For performance evaluation we use generated images and samples from CIFAR10 dataset. The only evaluation metric is FID so we do not have error bars. The task does not require data collection, we only need real samples from CIFAR10 for evaluation.

We compared results ours against authors:

	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
FID↓ Sampler / Predictor	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	4.98 ± .06	4.88 ± .06		**3.62** ± .03	3.24 ± .02	3.24 ± .02		**3.21** ± .02
reverse diffusion	4.79 ± .07	4.74 ± .08	20.43 ± .07	**3.60** ± .02	3.21 ± .02	3.19 ± .02	19.06 ± .06	**3.18** ± .01
probability flow	15.41 ± .15	10.54 ± .08		**3.51** ± .04	3.59 ± .04	3.23 ± .03		**3.06** ± .03

Figure 2. Frechet Inception Distance results from (Song et al., 2020)

FID	P1000	P2000	**PC1000**	C1000	C2000
ancestral sampling	31.76	31.7	**30.57**	173.77	71.51
reverse diffusion	31.98	31.43	**30.96**		

Table 1. Our results for Variance Exploding method

FID	P1000	P2000	**PC1000**	C1000	C2000
ancestral sampling	30.55	30.53	**29.74**	169.89	69.68
reverse diffusion	31.01	30.32	**30.01**		

Table 2. Our results for Variance Preserving method

We ran experiments on Skoltech Lab's GPU. In the inference mode allocated GPU memory was around 4 GB. We ran experiments in pairs, so approximately 8 GB at any time was allocated.

Hyperparameters in this task are modes of work (P, C, PC) and number of steps.

5.2. Research

We used pretrained on CIFAR10 models checkpoints from (DCGAN Github). We implemented Energy function from the (Che et al., 2021) paper and built optimization procedure that replicates Langevin Dynamics in the latent space of the generator.

We leveraged PyTorch functionality for loss optimization for our setting. We were interested in dynamics in latent space of the generator and we tasked PyTorch optimizer

with loss optimization with respect to z - originally a sample from prior distribution. Loss function was initialized as Energy function which we implemented earlier. The Langevin dynamics could be interpreted as slightly altered Gradient Descent with additional step of adding n_i sample from the prior distribution at each step.

Pseudo code for our implementation can be seen below:

Algorithm 2 Langevin Dynamics SGD implementation

Input: N$\in \mathbf{N}_+, \epsilon > 0$
Output: Latent code $z_N \sim p_t(z)$
Optimizer = SGD with full gradient
Sample $z_0 \sim p_0(z)$
for $i = 1$ **to** N **do**
  Compute Loss
  Back propagate Loss
  SGD step: $z_{i+1} \leftarrow z_i - \epsilon/2\nabla_z E(z_i)$
  $n_i \sim N(0, 1)$
  $z_{i+1} \leftarrow z_{i+1} + \sqrt{\epsilon} n_i$
end for

We ran experiments on several Google Colabs.

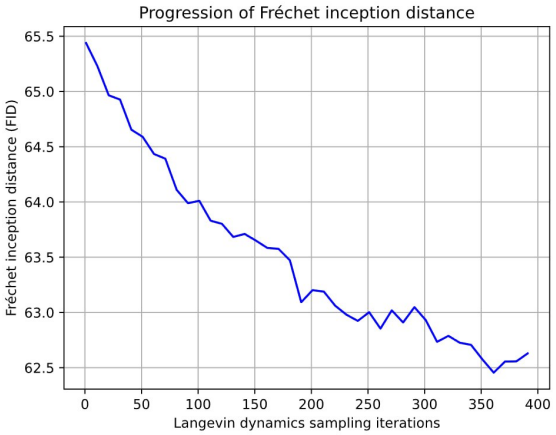

Figure 3. Diagram of transformation of an image to prior and restoration from prior with reverse-time SDE, (Song et al., 2020)

5.3. FID Calculation

To compute the FID, we first pass both the real and generated data through the Inception V3 neural network. We then use the activations of the mixed 8x8 and mixed 7x7 layers of the neural network as the features for the real and generated data distributions. These activations are flattened to obtain a feature vector for each sample. We then compute the mean and covariance of the feature vectors for both the real and generated data distributions and use them to compute the FID score using the formula above.

6. Conclusion

References

Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. ISSN 0304-4149. doi: https://doi.org/10.1016/0304-4149(82)90051-5. URL https://www.sciencedirect.com/science/article/pii/0304414982900515.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling, 2021.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep policy gradients: A case study on ppo and trpo, 2020.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

Munjal, P., Paul, A., and Krishnan, N. C. Implicit discriminator in variational autoencoder, 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *CoRR*, abs/2011.13456, 2020. URL https://arxiv.org/abs/2011.13456.

Tran, N.-T., Tran, V.-H., Nguyen, N.-B., and Cheung, N.-M. An improved self-supervised gan via adversarial training. *ArXiv*, abs/1905.05469, 2019.

Appendix A. Contributions of the Team Members

Every team member contributed to this project trying to do their best. As this project was really mathematically intensive, Georgiy and Oleg applied their mathematical background to understand the whole mathematical background of the related work and were focused mostly on research task. Anastasia and Nikolay were working mainly on pipeline creation and experiments managing and were working on replication task. Vadim helped them to evaluate the experiments and suggested an approach for FID metrics evaluation and sorted out a lot of problems in original code that lack of clear dependencies and contains bugs that he gracefully corrected and thus contributed greatly to the whole project. The whole team worked in very organized manner and appeared to be well-balanced in terms of skills.

Appendix B. Reproducibility Checklist

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used. Yes. Approximately 30% of code in replication task is ours. We used code and checkpoints of pretrained models from Notebook of original paper for replication task and we used checkpoints of pretrained models from DCGAN Github.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report. Yes

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report. Yes

4. A complete description of the data collection process, including sample size, is included in the report. No

5. A link to a downloadable version of the dataset or simulation environment is included in the report. No

6. An explanation of any data that were excluded, description of any prepossessing step are included in the report. No

7. An explanation of how samples were allocated for training, validation and testing is included in the report.No

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report. No

9. The exact number of evaluation runs is included. No

10. A description of how experiments have been conducted is included. Yes

11. A clear definition of the specific measure or statistics used to report results is included in the report. Yes

12. Clearly defined error bars are included in the report. No

13. A description of the computing infrastructure used is included in the report. Yes