Score-Based Diffusion Models

Winnie Xu, Chi Zhang

April 21 2022

1 Background: Diffusion Models

Idea A continuous time stochastic process that adds infinitesimal noise to input, gradually diffusing input input data into a noise vector.

1.1 Score Matching

The score of a pdf $p(x)$ is $\nabla_x \log p(x)$. Score-based models can be parameterized by any non-normalized statistical model because computing the score of such a distribution does not require computing the intractable normalizing constant.

If two differentiable functions have equal first derivatives we know that $f = g + C$, if these functions are log probability density functions, then we can use the the normalization requirement (aka $\int \exp(f)dx = 1$) to show that they're equal. From that property and from the fact that computing the score of an EBM is easier than computing the distribution itself, one can train EBMs by matching its score with the score of the data distribution [3].

1.1.1 Basic Score Matching

The basic Score Matching objective minimizes the Fisher Divergence between the distributions (half the square of the norm of the difference of the scores):

$$D_F(p_{data}(x)||p_\theta(x)) = \mathbb{E}_{p_{data}}\left[\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\partial E_\theta(x)}{\partial x_i}\right)^2 + \frac{\partial^2 E_\theta(x)}{(\partial x_i)^2}\right] + C$$

The problem with this basic idea is that it requires computing the trace of the Hessian, which is normally quadratic in the dimensionality of x, and so extremely costly (I wonder if there isn't some Jacobian-vector-product magic that could make it efficient).

1.1.2 Denoising Score Matching

The simple score matching objective requires many regularity conditions for $\log p_{data}$, for example it should be continuously differentiable and finite everywhere. But for example a distribution of digital images is discrete and bounded, as the value of the pixels range from 0 to 255 and are integers. That makes $p_{data} = 0$ outside that range, and so its logarithm is negative infinity, making SM not directly applicable. To alleviate this difficulty one can smooth the distribution by adding some noise $\tilde{x} = x + \epsilon$, with $p(\epsilon)$ being smooth. The resulting noisy data distribution $q(\tilde{x}) = \int q(\tilde{x}|x)p_{data}(x)dx$ is smooth, and so the Fisher Divergence is well-behaved. The new objective is:

$$\begin{aligned} D_F(q(\tilde{x})||p_\theta(\tilde{x})) &= \mathbb{E}_{q(\tilde{x})}\left[\frac{1}{2}||\nabla_x \log q(\tilde{x}) - \nabla_x \log p_\theta(\tilde{x})||_2^2\right] \\ &= \mathbb{E}_{q(x,\tilde{x})}\left[\frac{1}{2}||\nabla_x \log q(\tilde{x}|x) - \nabla_x \log p_\theta(\tilde{x})||_2^2\right] \end{aligned}$$

1.2 Denoising Diffusion Probabilistic Models

DDPM [2] is a parameterization of the reverse diffusion process that can be interpreted as sampling from a score-based model using Langevin dynamics:

$$x_{t+1} = x_t + \frac{\epsilon^2}{2\lambda}\nabla_x f_\theta(x) + \epsilon\alpha \qquad\qquad \alpha \sim \mathbb{N}(0, I)$$

with hyperparameters ϵ step-size and λ temperature. Lower temperature generates samples quicker for efficient training.

Problem Let x_0 be a data distribution that is hard to handle; we only observe a dataset containing iid samples from it, but do not know the closed form solution of this distribution. We want to estimate it. From here-on, there are two directions:

1. Select a *diffusion process* to perturb an input datum into a noise vector. This is a parameter-free T-step Markov chain describing a diffusion process from data $x_0 \sim p_{data}$ to latent variable x_T:

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

   where

2. *Reverse diffusion process* to generate samples from a data distribution by denoising noise vectors. This requires $\log p_t$ (distribution of perturbed distribution) and is a parameterized T-step Markov chain.

The reverse diffusion process' closed form requires solving a reverse SDE, which requires knowing the score function of the log density $\nabla \log p_t$, where p_t is the distribution of the noise perturbed data at time step t.

Limitations The choice of weighting function (the λ in $E[\lambda(t)||s(t,x) - \nabla_x log p_t(x)||)$ is incredibly important to the success of training. Given an SDE that diffuses data at t=0 to a normal at t=1, λ should essentially upweight the t=1 end of things, otherwise, for example a constant λ will fail. In tuition, the upweighting the t=0 end of things would be more important, as during inference that where we put the "finishing touches" on denoising the sample. In this project, we are going to reweight things during training and analyse the process

2 Methodology

2.1 Noising

We continuously add noise to images. For images x at any time t, the magnitude of corruption follows an SDE:

$$dx = f(x,t)dt + g(t)dW \tag{1}$$

For a total of N noise scales, each perturbation kernel $\{p_{\alpha_i}(x|x_0)\}_{i=1}^N$ of DDPM [2] corresponds to the distribution of x_i in the following Markov chain:

$$x_i = \sqrt{1-\beta_i}x_{i-1} + \sqrt{\beta_i}z_{i-1}, \qquad\qquad i = 1, 2, 3, \ldots, N \tag{2}$$

As N $\longrightarrow \infty$, Eq. (1) converges to the following SDE, that is, the Variance Preserving (VP) SDE:

$$dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dw \tag{3}$$

Then we apply the VP SDE to generate a noisy data sample to train on.

2.2 Denoising

2.2.1 Apply reverse SDE to denoise data

$$dx = [f(x,t) - g^2(t)\nabla_x \log p_t(x)]dt + g(t)d\bar{w} \tag{4}$$

$\nabla_x \log p_t(x)$ in Eq. (4) is the learned score function.

2.2.2 Score-based Model Objective

To estimate the gradient $\nabla_x \log p_t(\mathbf{x})$, a score-based neural network model $s_\theta(\mathbf{x}, t)$ is used. Note that unlike flow or autoregressive models, you do not need to specify an architecture respecting causality or invertibility.

Minimize The expected square difference between the ground truth gradient and the score model:

$$\mathcal{J}_{SM}(\theta; \lambda(\cdot)) := \frac{1}{2}\int_0^T \mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)||\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - s_\theta(\mathbf{x}, t)||_2^2]dt \tag{5}$$

where $\lambda(t)$ is a positive weighting function. In practise, we select this coefficient function to beget a low variance objective. Since the ground truth is unknown, the objective cannot be directly evaluated. Instead we are getting an unbiased estimator (integral does not need to be evaluated in inclusive form, just need to 1) sample t, 2) sample a data point $x \sim p_t$) via denoising score-matching.

Denoising SM Transforms Eq (5) to the following objective which is equivalent up to a constant independent of θ:

$$\mathcal{J}_{DSM}(\theta; \lambda(\cdot)) := \frac{1}{2}\int_0^T \mathbb{E}_{p_t(\mathbf{x})p(\mathbf{x}'|\mathbf{x})}[\lambda(t)||\nabla_{\mathbf{x}} \log p_t(\mathbf{x}'|\mathbf{x}) - s_\theta(\mathbf{x}', t)||_2^2]dt \tag{6}$$

where $\mathbf{x} \sim p(\mathbf{x})$ and $\mathbf{x}' \sim p_{0_t}(\mathbf{x}'|\mathbf{x})$, the transition density tractably a Gaussian if drift $f_\theta(\mathbf{x}, t)$ is linear in $\mathbf{x}$.

2.3 Reweighting

As described in Section 1.2, we expect that rescheduling the weight schedule (i.e. $\lambda(t)$ in Eq. (5)(6)) will have changes on the sample quality. It is not clear why certain choices are worse than others and how to select this in a princpled way without relying on the underlying properties of the SDE.

3 Experimental Details

Below we trained a score-based generative model with the denoising objective while interpreting the forward corruption process as a VP SDE. Using the corresponding variance of the converging transity density as the weighting function, we obtain the following result by sampling with the probability flow ODE:

$$dx = \left[f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)\right]dt$$

The MNIST model was trained for 100k iterations. Due to computational constraints, the CIFAR10 model was only trained for 60k iterations. As a further enhancement to the model architecture, we use time-dependent layers as introduced in [1] called `ConcatSquash`, which is a linear layer that takes in t (time) and is commonly used in training neural differential equation models. Further, we explore

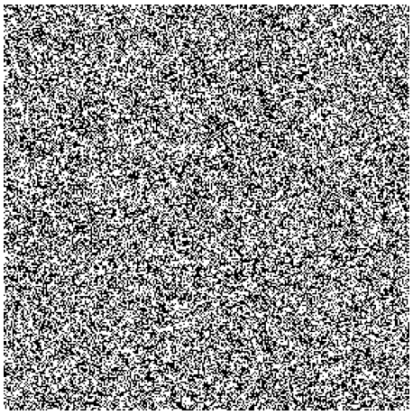
the use of an *MLPMixer2d* (Mixer model with 2D convolutions) from [4], which is significantly faster to train compared to the UNet architecture commonly used by the diffusion model community.

The model architecture parameters were:

- `patch_size`: 4
- `hidden_size`: 64
- `mix_patch_size`: 512
- `mix_hidden_size`: 512
- `num_blocks`: 4

4 Results and Discussion

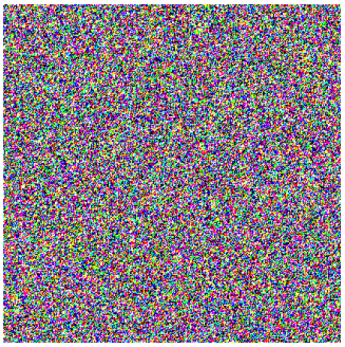
Evidently, longer training times are conducive to better results. Below are the baseline results with the weight function $1 - \exp(-\int \beta(t))$.



(a) Samples from 100 steps of Probability Flow ODE on denoising MNIST early on in training

(b) Samples from 100 steps of Probability Flow ODE on denoising MNIST

Figure 1



(a) Samples from 100 steps of Probability Flow ODE on denoising CIFAR10

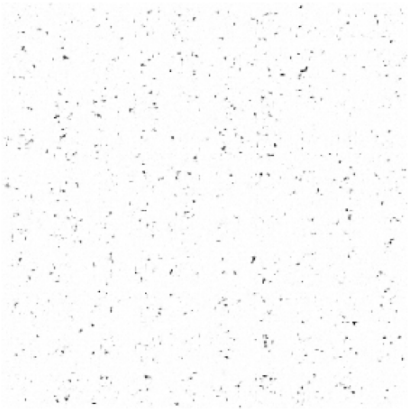
Figure 2

We then tried different weighting schemes setting to different functions that weight different aspects of the time horizon differently, and notice that the results change drastically from the default weighting function:

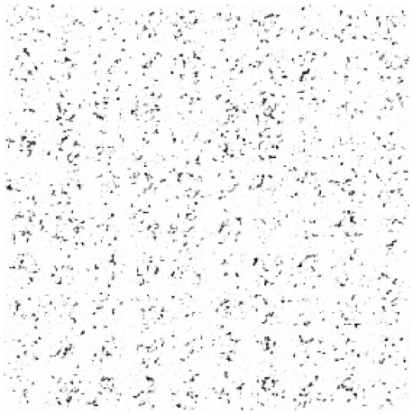
(a) Samples from 100 steps of Probability Flow ODE using $\exp\{\int \beta - 1/t\}$ function transformation

Figure 3

Notice that adding small perturbations that don't drastically change the degree to which different ends of the time horizon are updated (e.g. Figure 3) affects the sample quality in a slightly less harmful manner. As per the nature of noising, using a non-monotonic weighting function such as cos leads to poor results.

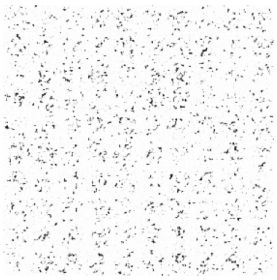


(a) Samples from 100 steps of Probability Flow ODE using exp as the $\int \beta$ function transformation

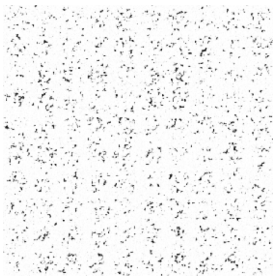


(b) Samples from 100 steps of Probability Flow ODE using $exp \times 100$ as the $\int \beta$ function transformation

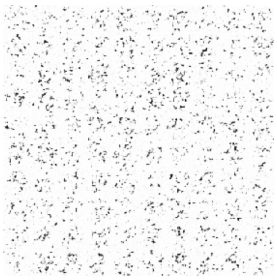
Figure 4



(a) Samples from 100 steps of Probability Flow ODE using cos as the $\int \beta$ function transformation

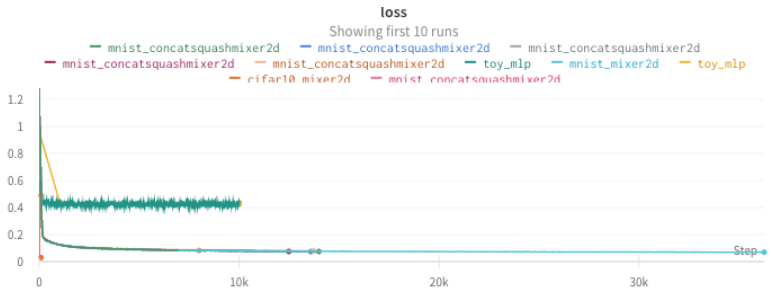


(b) Samples from 100 steps of Probability Flow ODE using $\cos \times 100$ as the $\int \beta$ function transformation



(c) Samples from 100 steps of Probability Flow ODE using $\cos / 100$ as the $\int \beta$ function transformation

Figure 5



(a) Loss curve indicates poor training with sub-optimal weighting functions.

Figure 6

References

[1] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[4] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34, 2021.