A somewhat deep dive into diffusion models

Anthony Baryshnikov

August 23, 2021

1 Introduction

Diffusion models (DMs) are a new class of generative models that have recently achieved SOTA on a lot of popular benchmarks, including CIFAR-10, LSUN, and ImageNet.

The theory behind them is relatively hard to understand, and due to an increasing amount of publications about DMs in the past couple of months, it may seem to be intimidating to get the hang of them.

In this article I will explain the general concepts that are essential to understanding DMs and review the most important papers on the topic.

2 Generative Modeling by Estimating Gradients of the Data Distribution

Score estimation is an approach to generative modelling where instead of assessing the probability density $p(x)$ of data distribution we try to estimate the gradient of its logarithm, also called the Stein score function $s(x)$. One advantage of score based modelling is that we no longer have to care about the normalizing constant of our model due to the fact that $s(x)$ is independent of it:

$$\nabla_x \log \frac{q_\theta(x)}{Z_\theta} = \nabla_x \log q_\theta(x) - \nabla_x \log Z_\theta = \nabla_x \log q_\theta(x) = s_\theta(x), \tag{1}$$

where $q_\theta(x)$ is an unnormalized representation of $p(x)$, and Z_θ is its normalization constant.

2.1 Training and Inference

Such models are usually trained using Fischer divergence:

$$\mathcal{L}_\theta = \mathbb{E}_{p(x)}\Big[\big\|\nabla_x \log p(x) - s_\theta(x)\big\|_2^2\Big]. \tag{2}$$

It can be shown that under certain constraints the objective is equivalent to:

$$\mathcal{L}_\theta = \mathbb{E}_{p(x)}\Big[\operatorname{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\big\|s_\theta(x)\big\|_2^2\Big] + C, \tag{3}$$

where C is an independent constant. This allows us to train our model without having an access to the true score function $\nabla_x \log p(x)$. This reparametrization of our objective is called score matching. We could then generate samples from our distribution by some variation of gradient ascent in data space, such as Langevin dynamics, which is suggested in the original paper.

2.2 Challenges of Score Matching

The first challenge is that due to the manifold hypothesis the score function is undefined almost everywhere. Because of that score matching objective is no longer guaranteed to provide us with a consistent estimator and it is impossible to traverse between disconnected supports during the sampling process.

The second challenge is that our score estimate is going to be very inaccurate in low density regions, which makes it even harder to generate good samples.

2.3 Noise Conditional Score Networks

One way to overcome these issues is to perturb data with various levels of decreasing noise and simultaneously estimate scores corresponding to all noise levels by training a single conditional score network. The largest noise variance should be sufficiently large so that all challenges discussed previously are mitigated, and the smallest noise variance should be small enough to be indistinguishable from real data.

We can also employ a denoising score matching objective with a normal noise distribution which allows for an even prettier expression that does not require us to calculate the trace of score gradient:

$$l_\theta(\sigma) = \frac{1}{2}\mathbb{E}_{p(x)}\mathbb{E}_{\tilde{x}\sim\mathcal{N}(x,\sigma^2 I)}\left[\left\|s_\theta(\tilde{x},\sigma) + \frac{\tilde{x}-x}{\sigma^2}\right\|_2^2\right] \tag{4}$$

$$\mathcal{L}_\theta(\{\sigma_i\}_{i=1}^L) = \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)l_\theta(\sigma_i), \tag{5}$$

where $\{\sigma_i\}_{i=1}^L$ is the noise schedule, and $\lambda(\sigma_i)$ is a weighting constant. The paper suggests setting the noise schedule to a geometric progression such that $\sigma_1 = 10$ and $\sigma_{10} = 0.01$ for data scaled to $[0, 1]$. There also exists a followup paper from the same authors which discusses the choice of hyperparameters in detail.

We can then sample from the data distribution using annealed Langevin dynamics (fig. 1).

Algorithm 1 Annealed Langevin dynamics.

```
Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.
1: Initialize $\tilde{\mathbf{x}}_0$
2: for $i \leftarrow 1$ to $L$ do
3:     $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$        ▷ $\alpha_i$ is the step size.
4:     for $t \leftarrow 1$ to $T$ do
5:         Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
6:         $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
7:     end for
8:     $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
9: end for
   return $\tilde{\mathbf{x}}_T$
```

Figure 1: Annealed Langevin dynamics.

3 Denoising Diffusion Probabilistic Models

Another approach to diffusion models comes from nonequilibrium thermodynamics. Let's perturb our data with a Markov chain that gradually adds Gaussian noise according to a variance schedule $\{\beta_t\}_{t=1}^T$.

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{6}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I). \tag{7}$$

The paper suggests a linear schedule such that $\beta_1 = 10^{-4}$ and $\beta_{1000} = 0.02$ for data scaled to $[-1, 1]$.

Our objective is to learn the reverse process $p_\theta(x_{t-1}|x_t)$, which is guaranteed to have the same functional form, normal distribution in this case, provided all β_t are small enough. We can further set reverse process variance to a fixed time-dependent constant $\sigma_t^2 I$. It has been shown experimentally that $\sigma_t^2 = \beta_t$ works well.

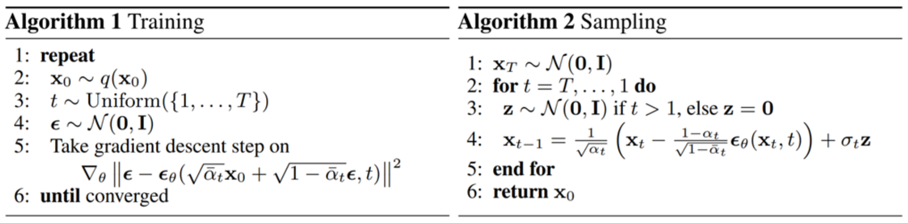
Training is performed by optimizing the variational lower bound on negative log likelihood, which can be shown to be equal to:

$$\mathcal{L}_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\varepsilon) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon\right) - \mu_\theta(x_t(x_0,\varepsilon),t)\right\|_2^2\right] + C \tag{8}$$

where $\alpha_t = 1 - \beta_t, \overline{\alpha}_t = \prod_{s=1}^t \alpha_s, \varepsilon \sim \mathcal{N}(0, I)$, and $\mu_\theta(x_t(x_0, \varepsilon), t)$ is the estimation of reverse process mean.

We can also reparametrize the objective in terms of added noise estimation:

$$\mathcal{L}_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha}_t)}\left\|\varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t} \cdot \varepsilon, t)\right\|_2^2\right] + C. \tag{9}$$



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 2: DDPM training and sampling algorithms.

This equation resembles the denoising score matching objective, which was discussed in Section 2.3. We can drop the weighting coefficient to get another, simplified objective:

$$\mathcal{L}_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\left\| \varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha}_t} x_0 + \sqrt{1-\overline{\alpha}_t} \cdot \varepsilon, t) \right\|_2^2 \right] + C. \tag{10}$$

The exact training and sampling algorithms can be seen in fig. 2.

4 Improved Denoising Diffusion Probabilistic Models

4.1 Learning Reverse Process Variance

Despite the fact that fixing reverse process variance to a time dependent constant works well, we can try to learn it. In order to do this we can predict a scalar v such that $\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1-v)\tilde{\beta}_t)$, where β_t and $\tilde{\beta}_t$ are the upper and lower bounds on reverse process variance, respectively. We then define a new hybrid objective:

$$L_{hybrid} = L_{simple} + \lambda L_{vlb} \tag{11}$$
$$L_{vlb} = L_0 + L_1 + \ldots + L_T \tag{12}$$
$$L_0 = -\log p_\theta(x_0|x_1) \tag{13}$$
$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \tag{14}$$
$$L_T = D_{KL}(q(x_T|x_0) \parallel p(x_T)), \tag{15}$$

where L_{simple} is the simplified objective proposed in Section 3, L_{vlb} is the variational lower bound of our model, and λ is a weighting term, which is set to 0.001 in the paper. The introduction of L_{vlb} is necessary due to the fact that L_{simple} is independent of $\Sigma_\theta(x_t, t)$.

It has been shown experimentally that L_{vlb} is extremely noisy, so importance sampling has to be employed to make the optimization process more smooth:

$$L_{vlb} = \mathbb{E}_{t\sim p_t}\left[\frac{L_t}{p_t}\right] \tag{16}$$

$$p_t \propto \sqrt{\mathbb{E}[L_t^2]},\ \sum p_t = 1, \tag{17}$$

where L_t is the part of variational lower bound that depends on diffusion timestep t.

4.2 Improving Sampling Speed

The number of timesteps in the diffusion process required for good sample quality may reach up to several thousands. This causes sampling speed to be extremely slow, reaching up to several minutes per batch. Given an arbitrary subsequence of diffusion steps S we can calculate the new forward process variances to be:

$$\beta_{S_t} = 1 - \frac{\overline{\alpha}_{S_t}}{\overline{\alpha}_{S_{t-1}}} \tag{18}$$

$$\tilde{\beta}_{S_t} = \frac{1-\overline{\alpha}_{S_{t-1}}}{1-\overline{\alpha}_{S_t}}\beta_{S_t}. \tag{19}$$

Since Σ_θ is parametrized as a range between β_t and $\tilde{\beta}_t$, it will automatically be rescaled for the shorter diffusion process. By employing this trick we can increase sampling speed by several orders of magnitude. The paper suggests using $K < T$ evenly spaced timesteps as the new subsequence S.

It has also been shown experimentally that fixing reverse process variances and using L_{simple} produces much worse samples when combined with a shortened diffusion process.

4.3 Improving the Noise Schedule

The linear schedule proposed in Section 3 has a big downside. The end part of the diffusion is so noisy that it can easily be removed without affecting sample quality. The authors suggest a cosine schedule that mitigates this issue. It is defined as:

$$\overline{\alpha}_t = \frac{f(t)}{f(0)} \tag{20}$$

$$f(t) = \cos\left(\frac{t/T+s}{t+s}\cdot\frac{\pi}{2}\right)^2 \tag{21}$$

$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \tag{22}$$

where s is chosen to be 0.008. In practice, β_t is also clipped to be no larger than 0.999 to prevent singularities.

5 Diffusion Models Beat GANs on Image Synthesis

We can improve the quality of this model even further, which allows us to beat SOTA GANs on several popular benchmarks. Most of the proposed changes are architectural, however, the authors suggest an interesting idea called classifier guidance that improves conditional image sampling.

They define a conditional reverse process:

$$p_{\theta,\varphi}(x_t|x_{t+1}, y) = \frac{p_\theta(x_t|x_{t+1})p_\varphi(y|x_t)}{Z}, \tag{23}$$

where Z is a normalization constant. This allows us to use $\nabla_{x_t} \log p_\varphi(y|x_t)$ during sampling by training a separate classifier on noisy images and then using its gradients with respect to x. The formula for a reverse diffusion step becomes:

$$x_{t-1} \sim \mathcal{N}(\mu_\theta + s\Sigma_\theta \nabla_{x_t} \log p_\varphi(y|x_t), \Sigma_t), \tag{24}$$

where s is the gradient scale. Increasing s allows us to achieve higher precision while trading it off for recall.

6 Score-Based Generative Modelling through SDEs

Another way to advance the model is to consider the case of continuous diffusion. In this case a finite Markov chain turns into a stochastic differential equation:

$$\mathrm{d}x = f(x,t)\,\mathrm{d}t + g(t)\,\mathrm{d}w, \tag{25}$$

where $f(x,t)$ is a vector valued function called the drift coefficient, $g(t)$ is a scalar function called the diffusion coefficient, and w corresponds to the standard Wiener process.

Let's denote by p_t the diffusion probability density at moment t. By starting samples from $x(T) \sim p_T$, we can obtain samples from $x(0) \sim p_0$ using a reverse diffusion process, running backwards in time and given by the reverse-time SDE:

$$\mathrm{d}x = [f(x,t) - g(t)^2 \nabla_x \log p_t(x)]\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{w}, \tag{26}$$

where $\overline{w}$ is a standard Wiener process when time runs backwards. Our objective then becomes to learn the score of each marginal distribution at every timestep:

$$\mathcal{L}_\theta = \mathbb{E}_{t\sim \text{Uniform}(0,T)} \left[\lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} \left[\left\| s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0)) \right\|_2^2 \right] \right], \tag{27}$$

where $\lambda(t)$ is a weighting coefficient, and p_{0t} is the transition kernel from moment 0 to moment t. This equation very much resembles eq. 2. In case of affine $f(x,t)$ the transition kernel is always a Gaussian distribution and can be obtained in closed form. In other cases either solving the forward equation or simulating the SDE is required.

6.1 Connection to NCSNs and DDPMs

Both NCSNs and DDPMs can be thought of as a discretization of this SDE and their continuous versions are:

$$\mathrm{d}x = \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\,\mathrm{d}w \tag{28}$$

$$\mathrm{d}x = -\frac{1}{2}\beta(t)x\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}w, \tag{29}$$

respectively. The first SDE always yields exploding variance as $t \to \infty$, and the second one gives a process with fixed variance. Because of these properties they are called Variance Exploding (VE) and Variance Preserving (VP) SDEs. The authors also propose another variation of continuous diffusion called sub-VP SDE defined by this process:

$$\mathrm{d}x = -\frac{1}{2}\beta(t)x\,\mathrm{d}t + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)\,\mathrm{d}s})}\,\mathrm{d}w, \tag{30}$$

which is reported to work particularly well on likelihoods.

6.2 Sampling

Now we can use any black-box SDE solver for sampling. However, we can go even further and employ score-based MCMC approaches to correct the solution of a numerical SDE solver at timestep t. This procedure is called predictor-corrector (PC) sampling. Interestingly enough, DDPMs completely avoid the corrector step, and NCSNs avoid the predictor step, so PC sampling can be thought of as a generalization of the two.

Both PC samplers and black-box SDE solvers allow us to explicitly trade off between quality and speed.

6.3 Probability Flow

Another interesting result of this approach is the fact that we can obtain a deterministic process, called the probability flow ODE, that has the same marginal probabilities as the original SDE:

$$\mathrm{d}x = \left[f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)\right]\mathrm{d}t, \tag{31}$$

This gives us several benefits. First of all, we can compute the exact likelihood on any input data by following the results of Neural ODEs (Chen et. al., 2018). Second, any sample obtained from p_T now corresponds to a unique point in p_0, therefore allowing us to manipulate latent representations like other generative models, such as GANs and VAEs.

6.4 Controllable generation

Similar to classifier guidance discussed in Section 5, we can use conditional scores for controllable sampling. The corresponding SDE is:

$$\mathrm{d}x = \{f(x,t) - g(t)^2[\nabla_x \log p_t(x) + \nabla_x \log p_t(y|x)]\}\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{w}. \tag{32}$$

We can either train a separate model to get an estimate of $\nabla_x \log p_t(y|x)$, such is the case for class-conditional sampling, use domain knowledge, or employ a general approximation method. For detailed explanation of controllable generation please refer to Appendix I of the original paper.

7 Variational Diffusion Models

Let's look a little more into how the choice of forward diffusion can impact our model. The generalized kernel is:

$$q(z_t|x) = \mathcal{N}(\alpha_t x, \sigma_t^2 I), \tag{33}$$

where α_t and σ_t^2 are non-negative scalar-valued functions such that α_t^2/σ_t^2 is strictly monotonically decreasing, and z_t is a latent variable for which the diffusion happens. Also, we confine timestep t to $[0,1]$ for simplicity's sake.

In continuous case this corresponds to this SDE:

$$\mathrm{d}z_t = [f(t)z_t - g^2(t)s_\theta(z_t,t)]\,\mathrm{d}t + g(t)\,\mathrm{d}w \tag{34}$$

$$f(t) = \frac{\mathrm{d}\log \alpha_t}{\mathrm{d}t} \tag{35}$$

$$g^2(t) = \frac{\mathrm{d}\sigma^2(t)}{\mathrm{d}t} - 2\frac{\mathrm{d}\log \alpha_t}{\mathrm{d}t}\sigma^2(t). \tag{36}$$

7.1 SNR

We will now consider the case of variance preserving diffusion only. Let's define signal-to-noise ratio as:

$$\mathrm{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}. \tag{37}$$

It has several interesting properties. For example, the variational lower bound can be simplified considerably by using SNR:

$$\mathcal{L}_{\text{discrete}} = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I), i\sim\text{Uniform}\{1,T\}}\left[\frac{T}{2}(\mathrm{SNR}(e_{i-1}) - \mathrm{SNR}(e_i))\left\|x - \hat{x}_\theta(z_t,t)\right\|_2^2\right] \tag{38}$$

$$\mathcal{L}_{\text{continuous}} = -\frac{1}{2}\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I), t\sim\text{Uniform}(0,1)}\left[\mathrm{SNR}'(t)\left\|x - \hat{x}_\theta(z_t,t)\right\|_2^2\right], \tag{39}$$

where $e_i = i/T$, and $\hat{x}_\theta$ is our estimate of x from its noisy version z_t. The given objectives are for discrete and continuous case, respectively.

It can be shown that continuous time loss depends on $\mathrm{SNR}_{max} = \mathrm{SNR}(0)$ and $\mathrm{SNR}_{min} = \mathrm{SNR}(1)$, and is otherwise invariant to the noise schedule. By using a change of variables formula our new objective becomes:

$$\mathcal{L}_{\text{continuous}} = \frac{1}{2}\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)} \int_{\mathrm{SNR}_{min}}^{\mathrm{SNR}_{max}} \left\| x - \hat{x}_\theta(z_v, v) \right\|_2^2 \mathrm{d}v. \tag{40}$$

It is also possible to add a weighting term $w(v)$ to the integral. This captures all the different objectives discussed in other papers. In practice, this objective is optimized using a Monte Carlo estimate:

$$\mathcal{L}_{\text{continuous}}^{MC} = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I), t\sim\text{Uniform}(0,1)} \left[\frac{1}{2}\gamma'(t) w(\gamma(t)) \left\| \varepsilon - \hat{\varepsilon}_\theta(z_t, \gamma(t)) \right\|_2^2 \right], \tag{41}$$

where $\gamma(t) = -\log \mathrm{SNR}(t)$.

7.2 Learning the Noise Schedule

It turns out that it is possible to jointly learn the noise schedule of our diffusion process. Let's parametrize $\mathrm{SNR}(t) = \exp\bigl(-\gamma_\eta(t)\bigr)$, where $\gamma_\eta(t)$ is a monotonically increasing neural network.

In discrete time we simply optimize the variational lower bound. However, as shown before, the continuous case is different, as its objective does not depend on SNR apart from its values at points 0 and 1. What does depend on SNR though is the variance of the Monte Carlo estimate. The exact optimization process for minimizing $\mathbb{V}[\mathcal{L}_{\text{continuous}}^{MC}]$ is discussed in Appendix D.1 of the original paper.

7.3 Fourier Features for Improved Fine Scale Prediction

At smallest noise levels the discrete nature of 8-bit data leads to peaked marginal distributions. Adding Fourier features helps with this issue. The proposed formula is:

$$f_{i,j,k}^n = \sin\bigl(z_{i,j,k} 2^n \pi\bigr) \tag{42}$$

$$g_{i,j,k}^n = \cos\bigl(z_{i,j,k} 2^n \pi\bigr), \tag{43}$$

where $z_{i,j,k}$ is the pixel at position (i, j) and channel number k, and n corresponds to the feature frequency. The paper suggests using $n \in \{7, 8\}$. The features are then concatenated to the input data and can be viewed like additional channels.