Diffusion Models I

DDPMs

April 29th, 2024
Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal
OpenAI
prafulla@openai.com

Alex Nichol
OpenAI
alex@openai.com
Image generated by DALL-E 3 based on prompt “cartoon penguin riding a unicycle”.

Generated protein backbone from RFDiffusion.

Audio generation with WaveGrad.
What are Diffusion Models?

- Diffusion models are a kind of deep generative model, i.e., a model that learns to sample from an underlying probability distribution.

- Diffusion models generate samples through a noising/denoising process:
  - During training, “noise” is added to samples from the distribution, and the model learns to predict this noise.
  - During inference, a noisy sample is first generated. The model then iteratively removes the noise until it produces a final result, which should be a sample from the desired distribution.
• Why do this?
  • The noisy distribution is much simple to sample from.
  • The generation process is broken down into smaller, easier steps.
  • Most of the time, it is easy to add a desired amount of noise to a sample, making training simple.
Denoising Diffusion Probabilistic Models

• First diffusion model described in “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”

  • Inspired by methods in thermodynamics and statistics (in particular, Annealed Importance Sampling)

• This lecture will focus on the diffusion model proposed in “Denoising Diffusion Probabilistic Models” (DDPMs)

  • Arguably the most popular form of diffusion models.

  • Simpler to understand and implement.

• Notation between papers is somewhat inconsistent, but we will be following the notation in the DDPM paper.
• Underlying distribution: \( x_0 \sim q(x_0) \)

• Add Gaussian noise \( T \) times to get \( x_1, x_2, \ldots, x_T \)

• Forward process is a Markov chain: 
  \[ q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \]
  where \( 0 < \beta_i < 1 \) is the variance schedule

• Can sample at arbitrary \( t \) without stepping through MC: 
  \( \alpha_t := 1 - \beta_t \) and 
  \( \overline{\alpha}_t := \prod_{s=1}^{t} \alpha_s \) then 
  \[ q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1 - \overline{\alpha}_t)I) \]

Approaches standard normal distribution: \( \mathcal{N}(x_T; 0, I) \)
• Reverse process: \( p_\theta(x_T) = \mathcal{N}(x_T; 0, I) \)

• When \( \beta_t \) are small, the reverse process can also be written as Gaussian transitions: \( p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \)

• Determining \( \mu_\theta \) and \( \Sigma_\theta \) will determine the backward process. We set \( \Sigma_\theta = \beta_t I \) for simplicity.

• Training goal is to minimize the negative log likelihood
Given trained model, sample Gaussian noise and then step through reverse process MC to get a sample $x_0$. 

$\mathbf{x}_0 \rightarrow \mathbf{x}_{t-1} \rightarrow \mathbf{x}_t \rightarrow \mathbf{x}_{t+1} \rightarrow \mathbf{x}_T$

$p(x_0|x_1) \quad q(x_1|x_0) \\
p(x_{t-1}|x_t) \quad q(x_t|x_{t-1}) \\
p(x_t|x_{t+1}) \quad q(x_{t+1}|x_t) \\
p(x_{T-1}|x_T) \quad q(x_T|x_{T-1})$
• see “Understanding Diffusion Models: A Unified Perspective” for detailed derivation

\[ \mathbb{E} \left[ -\log p_\theta(x_0) \right] \leq \]

\[ \mathbb{E}_{q(x_1|x_0)} \left[ -\log p_\theta(x_0 | x_1) \right] + D_{KL}(q(x_T | x_0) \| p(x_T)) + \sum_{t>1} \mathbb{E}_{q(x_t| x_0)}[D_{KL}(q(x_{t-1} | x_t, x_0) \| p_\theta(x_{t-1} | x_t)] \]

• First term: reconstruction term. Can be optimized separately, but ultimately will be treated along with other terms.

• Second term: measure of difference between normal distribution and explicit prior distribution (does not depend on \( \theta \))

• Third term: measure of difference between backward process and the actual marginal distributions of the forward process.
• Let’s optimize the third term through gradient descent!

• KL divergence terms have exact formulas when distributions are normal.

  • Recall: \( p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \beta_t I) \); Just need other distribution.

\[
q(x_{t-1} \mid x_t, x_0) = \frac{q(x \mid x_{t-1}, x_0)q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \\
\ldots \\
= \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)
\]

where,

\[
\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\alpha_t} \beta_t}{1 - \bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t
\]
• Plugging into KL divergence...

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

• One could train a model off this using gradient descent, but there’s a simpler formulation not dependent on \(x_t\).

• Re-parameterize based on explicit formula for \(q(x_t \mid x_0)\). Knowing \(x_0\) allows easy sampling of \(x_t\):

\[
x_t(x_0, \epsilon) = \sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, 1)
\]

• Then substituting the equivalent value for \(x_0\) gives

\[
\tilde{\mu}_t(x_t(x_0, \epsilon), x_0) = \frac{1}{\sqrt{\alpha_t}} \left( x_t(x_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \alpha_t}}\epsilon \right)
\]
• Since our model knows $x_t$ at inference and needs to approximate $\tilde{\mu}$, a good parameterization of $\mu_\theta$ is

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

• So our model is now predicting $e$ given $x_t$ and the loss for fixed $t$ becomes

$$
\mathbb{E}_{x_0, e} \left[ \frac{\beta_t^2}{2\beta_t \alpha_t (1 - \alpha_t)} \| e - e_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}e, t) \|^2 \right]
$$

• Tempting to drop the time scaling out front, so let’s try it:

$$
L_{\text{simple}}(\theta) := \mathbb{E}_{t, x_0, e} \| e - e_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}e, t) \|^2
$$

• This ends up working very well.
Algorithm 1 Training

1: repeat
2: \( x_0 \sim q(x_0) \)
3: \( t \sim \text{Uniform}\{1, \ldots, T\} \)
4: \( \epsilon \sim \mathcal{N}(0, I) \)
5: Take gradient descent step on
\[
\nabla_\theta \|\epsilon - \epsilon_\theta (\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon, t)\|^2
\]
6: until converged

Algorithm 2 Sampling

1: \( x_T \sim \mathcal{N}(0, I) \)
2: for \( t = T, \ldots, 1 \) do
3: \( z \sim \mathcal{N}(0, I) \) if \( t > 1 \), else \( z = 0 \)
4: \( x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z \)
5: end for
6: return \( x_0 \)
Why did we use $L_{\text{simple}}(\theta)$ instead of the log-likelihood maximizing loss?

- **Empirical reason:** Using $L_{\text{simple}}(\theta)$ results in better sample quality (and is easier to implement).

- **Slightly more detailed reason:** the simplified loss more heavily weights small times in denoising. This is important in maintaining image quality when sampling.

- **Theoretical reason:** this loss learns the score of perturbed distribution (rewighted based on variances).

\[
\nabla \log q(x_t | x_0) = -\frac{1}{\sqrt{1 - \alpha_t}} \epsilon
\]

- **Two interpretations of DDPMs:** learning to remove noise or learning perturbed distributions.
Further Reading


• “Denoising Diffusion Probabilistic Models” by Ho, et al.

• “Understanding Diffusion Models: A Unified Perspective” by Luo