

Diffusion Generative Models

CS 274E Guest Lecture

Gavin Kerrigan

2022 17 November

gavin.k@uci.edu

Introduction

Diffusion Generative Models are a class of deep generative models that generate data by iterative denoising.

Introduction

Diffusion Generative Models are a class of deep generative models that generate data by iterative denoising.

Some history:

- First introduced in [Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015]
- Popularized by [Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020] and [Song and Ermon, Generative Modeling by Estimating Gradients of the Data Distribution, NeurIPS 2019]
- Huge amount of interest in last few years, both in research and applications

Introduction

Unconditional generation: Images

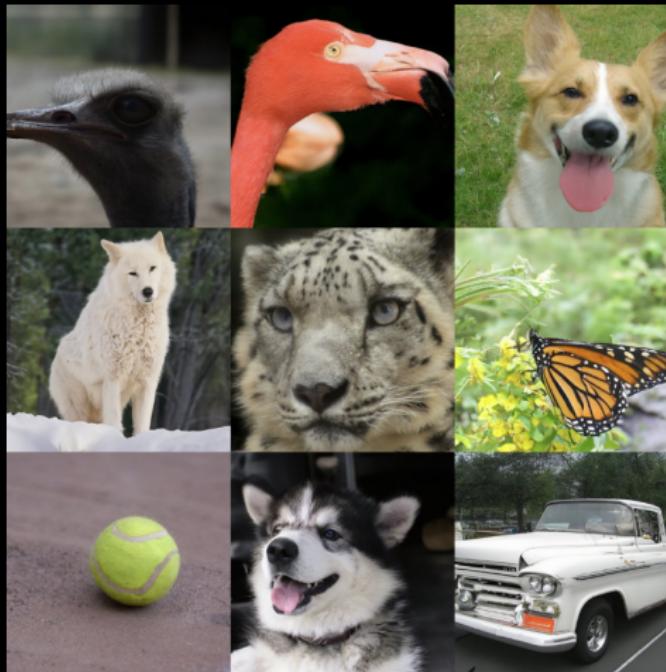


Figure 1: [Dhariwal and Nichol, Diffusion Models Beat GANs on Image Synthesis, NeurIPS 2021]

Introduction

Unconditional generation: Point clouds



Figure 2: [Cai et al., Learning Gradient Fields for Shape Generation, ECCV 2020]

Introduction

Conditional generation: Text → Image

- e.g. DALLE-2, Imagen, Stable Diffusion, etc.



Figure 3: “An oil painting of a cat wearing an ornate wizard hat and robe”

openai.com/dall-e-2

Introduction

Conditional generation: Image → Image

- e.g. Super-resolution



Figure 4: [Saharia et al., Image Super-Resolution via Iterative Refinement, ICCV 2021]

Introduction

Conditional generation: Graphs → 3D Molecules

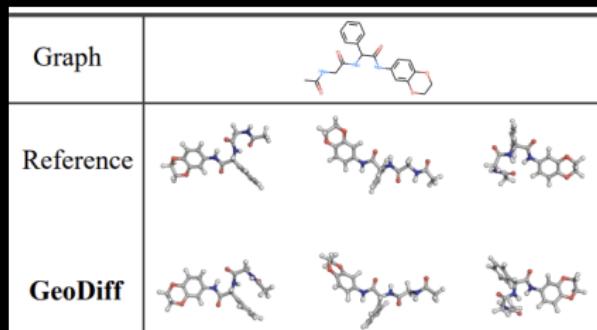


Figure 5: [Xu et al., GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation, ICLR 2022]

Denoising Diffusion Models

Denoising Diffusion Models

“Creating noise from data is easy; creating data from noise is generative modeling”¹.

This talk: Denoising Diffusion Probabilistic Models (DDPM)

- Derivations from [Ho et al., DDPM, NeurIPS 2020] and [Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015]
- Based in part on Arash Vahdat’s CVPR 2022 tutorial

¹Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations*, ICLR 2021

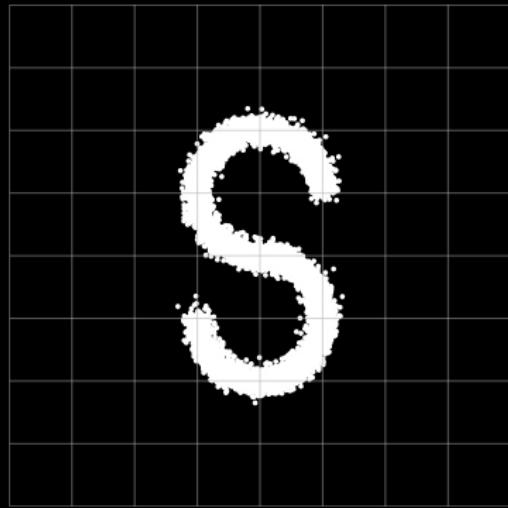
Denoising Diffusion Models

“Creating noise from data is easy; creating data from noise is generative modeling”².

²Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations*, ICLR 2021

Denoising Diffusion Models

Have samples from an unknown data distribution $q(x_0)$



Denoising Diffusion Models

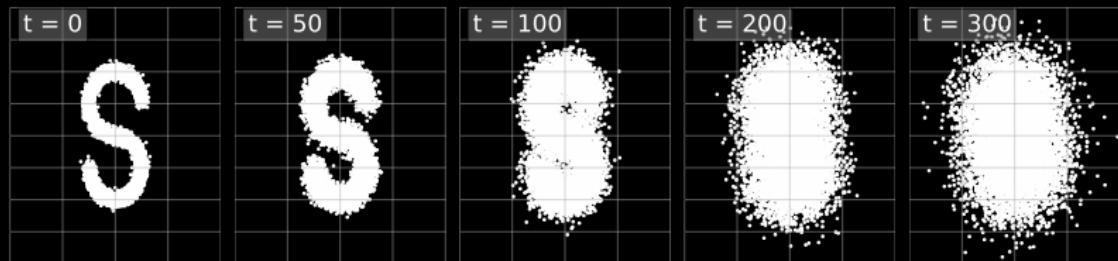
Creating noise from data is easy:

- Choose an integer T (typically large) and a variance schedule β_t
- β_t is often a linear interpolation between β_1 and β_T
- Slowly make your data noisier over T steps

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon \quad \epsilon \sim \mathcal{N}(0, I) \quad (1)$$

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t | \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \quad (2)$$

This defines the **forward** (diffusion) process:



Denoising Diffusion Models

Forward Process:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t \mid \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \quad (3)$$

- Gives you a joint distribution

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^T q(x_t \mid x_{t-1}) \quad (4)$$

Similar to a latent variable model / VAE:

- Encoder: q
- Latent variable(s): $x_{1:T}$
- Observed variable: x_0

Denoising Diffusion Models

Forward Process:

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t | \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \quad (5)$$

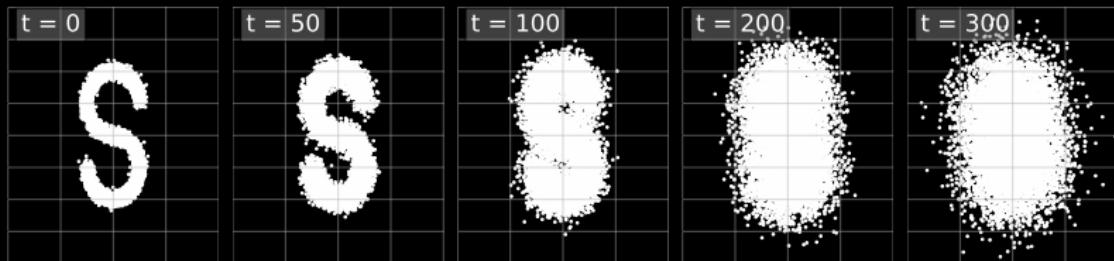
Cheap to sample at any time t . Set $\gamma_t = \prod_{s=1}^t (1 - \beta_s)$. Then:

$$x_t = \sqrt{\gamma_t} x_0 + \sqrt{(1 - \gamma_t)} \epsilon \quad \epsilon \sim \mathcal{N}(0, I) \quad (6)$$

$$q(x_t | x_0) = \mathcal{N}(x_t | \sqrt{\gamma_t} x_0, (1 - \gamma_t) I) \quad (7)$$

Proof (sketch): Write out the densities and compute.

- Note that for large t , $q(x_t | x_0) \approx \mathcal{N}(0, I)$



Denoising Diffusion Models

“Creating noise from data is easy; creating data from noise is generative modeling”³.

³Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations*, ICLR 2021

Denoising Diffusion Models

Generate data by reversing the diffusion process

- Sample $x_T \sim \mathcal{N}(0, I)$
- Iteratively sample

$$x_{t-1} \sim q(x_{t-1} \mid x_t) \quad t = T, T-1, \dots, 1 \quad (8)$$

Denoising Diffusion Models

Generate data by reversing the diffusion process

- Sample $x_T \sim \mathcal{N}(0, I)$
- Iteratively sample

$$x_{t-1} \sim q(x_{t-1} \mid x_t) \quad t = T, T-1, \dots, 1 \quad (9)$$

- Problem:

$$q(x_{t-1} \mid x_t) = \frac{q(x_t \mid x_{t-1}) q(x_{t-1})}{q(x_t)} \quad (10)$$

- Know forward transitions, but marginals are intractable
- Variational approximation:

$$q(x_{t-1} \mid x_t) \approx p_\theta(x_{t-1} \mid x_t) \quad (11)$$

$$= \mathcal{N}(x_{t-1} \mid \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (12)$$

Denoising Diffusion Models

Variational approximation:

$$q(x_{t-1} \mid x_t) \approx p_\theta(x_{t-1} \mid x_t) \quad (13)$$

$$= \mathcal{N}(x_{t-1} \mid \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (14)$$

How can we train such a model?

- Want to maximize the model likelihood $p_\theta(x_0)$
- Can think of $p_\theta(x_{t-1} \mid x_t)$ as the decoder in a VAE

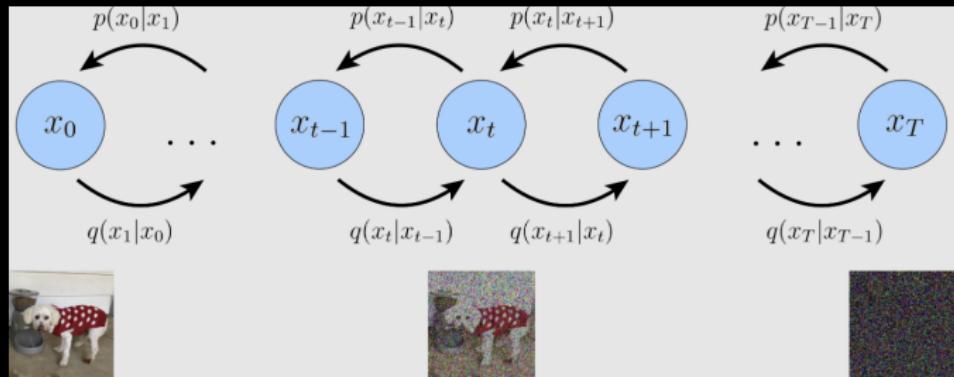


Figure 6: Image credit: Calvin Luo

Denoising Diffusion Models

The usual ELBO (treating $x_{1:T}$ as latents) is

$$\begin{aligned}\log p_\theta(x_0) &\geq \mathbb{E}_{x_{1:T} \sim q(\cdot | x_0)} \left[\log \frac{p_\theta(x_0, x_{1:T})}{q(x_{1:T} | x_0)} \right] \\ &= \mathbb{E}_{x_{1:T} \sim q(\cdot | x_0)} [\log p_\theta(x_0 | x_1)] - \text{KL}[q(x_{1:T} | x_0) || p_\theta(x_{1:T})]\end{aligned}\tag{15}$$

Let's analyze this to get something we can compute

Denoising Diffusion Models

Chain rule for the KL divergence: for any joint distributions,

$$\text{KL} [p(x, y) \mid q(x, y)] = \text{KL} [p(x) \parallel q(x)] + \mathbb{E}_{x \sim p(x)} \text{KL} [p(y|x) \parallel q(y|x)] \quad (16)$$

Proof (sketch):

Decompose the joint distributions into a product of marginal and conditional distributions. Plug into the definition of the KL divergence and compute.

Unconditional Denoising Diffusion Models

Chain rule for KL divergences:

$$\text{KL}[p(x, y) \mid q(x, y)] = \text{KL}[p(x) \parallel q(x)] + \mathbb{E}_{x \sim p(x)} \text{KL}[p(y|x) \parallel q(y|x)] \quad (17)$$

Apply to the ELBO to condition on x_T :

$$\begin{aligned} \log p_\theta(x_0) &\geq \mathbb{E}_q [\log q_\theta(x_0 \mid x_1)] - \text{KL}[q(x_{1:T} \mid x_0) \parallel p_\theta(x_{1:T})] \\ &= \mathbb{E}_q [\log q_\theta(x_0 \mid x_1)] - \text{KL}[q(x_T \mid x_0) \parallel p_\theta(x_T)] \\ &\quad - \mathbb{E}_q \text{KL}[q(x_{1:T-1} \mid x_0, x_T) \parallel p_\theta(x_{1:T-1} \mid x_T)] \end{aligned}$$

Repeat to condition on $x_{T-1}, x_{T-2}, \dots, x_1$ to obtain

$$\begin{aligned} \log p_\theta(x_0) &\geq \mathbb{E}_q \left[\log p_\theta(x_0 \mid x_1) - \right. \\ &\quad \left. \text{KL}[q(x_T \mid x_0) \parallel p_\theta(x_T)] - \sum_{t=2}^T \text{KL}[q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)] \right] \end{aligned}$$

Denoising Diffusion Models

Three types of terms appear in the loss:

$$L_T := \mathbb{E}_q \text{KL} [q(x_T | x_0) || p_\theta(x_T)] \quad (18)$$

- Can be ignored: we set $p_\theta(x_T) = \mathcal{N}(0, I)$
- Justified because $q(x_T | x_0) \approx \mathcal{N}(0, I)$ for T sufficiently large

Denoising Diffusion Models

Three types of terms appear in the loss:

$$L_0 := \mathbb{E}_q [\log p_\theta(x_0 \mid x_1)] \quad (19)$$

- This is the decoder log-likelihood
- Exactly analogous to the reconstruction term in a VAE

Denoising Diffusion Models

Three types of terms appear in the loss:

$$L_{t-1} := \mathbb{E}_q \text{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)] \quad (20)$$

By Bayes' rule:

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1}, x_0) q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \quad (21)$$

Everything here is known and Gaussian, so (after a tedious but straightforward calculation)

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t) I) \quad (22)$$

$$\mu_q(x_t, x_0) = \frac{\sqrt{\gamma_{t-1}} \beta_t}{1 - \gamma_t} x_0 + \frac{\sqrt{1 - \beta_t}(1 - \gamma_{t-1})}{1 - \gamma_t} x_t \quad \sigma_q^2(t) = \frac{1 - \gamma_{t-1}}{1 - \gamma_t} \beta_t$$

Denoising Diffusion Models

The story so far:

$$L_{t-1} := \mathbb{E}_q \text{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)] \quad (23)$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t) I) \quad (24)$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (25)$$

How should we parametrize $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$?

Denoising Diffusion Models

$$L_{t-1} := \mathbb{E}_q \text{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)] \quad (26)$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t) I) \quad (27)$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (28)$$

Let's make our lives easy and set⁴

$$\Sigma_\theta(x_t, t) = \sigma_q(t)^2 I \quad (29)$$

The KL between Gaussians has a closed form:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} \left[\|\mu_\theta(x_t, t) - \mu_q(x_t, x_0)\|_2^2 \right] \right] + C \quad (30)$$

⁴You could also learn the variance, e.g. [Kingma et al., Variational Diffusion Models, NeurIPS 2021]

Denoising Diffusion Models

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} [||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2] \right] \quad (31)$$

Variational mean μ_θ needs to predict the denoised mean μ_q .

How should we parametrize $\mu_\theta(x_t, t)$?

- Most straightforward: just have network try to predict μ_q
- Can we do better?

Denoising Diffusion Models

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} [||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2] \right] \quad (32)$$

Idea: we can exploit the structure of μ_q to obtain a better parametrization

$$\mu_q(x_t, x_0) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1 - \gamma_t} x_0 + \frac{\sqrt{1 - \beta_t}(1 - \gamma_{t-1})}{1 - \gamma_t} x_t$$

Since the model has x_t as input, we can parametrize via

$$\mu_\theta(x_t, t) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1 - \gamma_t} x_\theta(x_t, t) + \frac{\sqrt{1 - \beta_t}(1 - \gamma_{t-1})}{1 - \gamma_t} x_t \quad (33)$$

i.e. network needs to predict noise-free input from x_t :

$$x_\theta(x_t, t) \approx x_0 \quad (34)$$

Denoising Diffusion Models

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} [||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2] \right] \quad (35)$$

$$\mu_\theta(x_t, t) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1-\gamma_t} x_\theta(x_t, t) + \frac{\sqrt{1-\beta_t}(1-\gamma_{t-1})}{1-\gamma_t} x_t \quad (36)$$

$$x_\theta(x_t, t) \approx x_0 \quad (37)$$

Loss simplifies to

$$L_{t-1} = \mathbb{E}_q [C_t ||x_\theta(x_t, t) - x_0||^2] \quad (38)$$

$$C_t = \frac{1}{2\sigma_q^2(t)} \frac{\gamma_{t-1}\beta_t^2}{(1-\gamma_t)^2} \quad (39)$$

Denoising Diffusion Models

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} [||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2] \right] \quad (40)$$

An alternative parametrization

Since $x_t = \sqrt{\gamma_t}x_0 + \sqrt{1 - \gamma_t}\epsilon$ for $\epsilon \sim \mathcal{N}(0, I)$:

$$\begin{aligned} \mu_q(x_t, x_0) &= \frac{\sqrt{\gamma_{t-1}}\beta_t}{1 - \gamma_t}x_0 + \frac{\sqrt{1 - \beta_t}(1 - \gamma_{t-1})}{1 - \gamma_t}x_t \\ &= \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \gamma_t}}\epsilon \right) \end{aligned}$$

We can thus parametrize μ_θ as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \gamma_t}}\epsilon_\theta(x_t, t) \right) \quad (41)$$

i.e. network tries to predict noise added to x_t

$$\epsilon_\theta(x_t, t) \approx \epsilon \quad (42)$$

Denoising Diffusion Models

This parametrization results in a fairly simple and interpretable loss:

$$L_{t-1} = \mathbb{E}_\epsilon \left[C_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{t})\|^2 \right] \quad \epsilon \sim \mathcal{N}(0, I) \quad (43)$$

- C_t is a time-dependent constant; often dropped during training for simplicity

$$C_t = \frac{\beta_t^2}{2\sigma_q^2(t)(1 - \beta_t)(1 - \gamma_t)} \quad (44)$$

- $\epsilon_\theta(x_t, t)$ is a network that tries to predict the **added noise** from the **noisy input** – i.e. it is *denoising*

Denoising Diffusion Models

Putting everything together:

$$L = \mathbb{E}_q \left[\log p_\theta(x_0 \mid x_1) - \sum_{t=2}^T C_t \mathbb{E}_\epsilon \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right] \quad (45)$$

Algorithm 1 Training

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

- C_t is ignored
- Likelihood term is assumed to be Gaussian
- Recall $x_t = \sqrt{\gamma_t} x_0 + \sqrt{1 - \gamma_t} \epsilon$ – pseudocode uses $\bar{\alpha}_t = \gamma_t$

Denoising Diffusion Models

Sampling:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \gamma_t}} \epsilon_\theta(x_t, t) \right) \quad (46)$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_q^2(t)I) \quad (47)$$

Algorithm 2 Sampling

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

- Notation: $\alpha_t = 1 - \beta_t$ and $\sigma_t = \sigma_q(t)$

Denoising Diffusion Models

Some practical details:

- $\epsilon_\theta(x_t, t)$ is typically implemented via the U-net architecture
- Time input t is discrete integer – usually handled via (learnable) embeddings
- Can tune forward process: number of steps T , variance schedule β_t
- Often β_t is linear interpolation between some fixed β_1, β_T

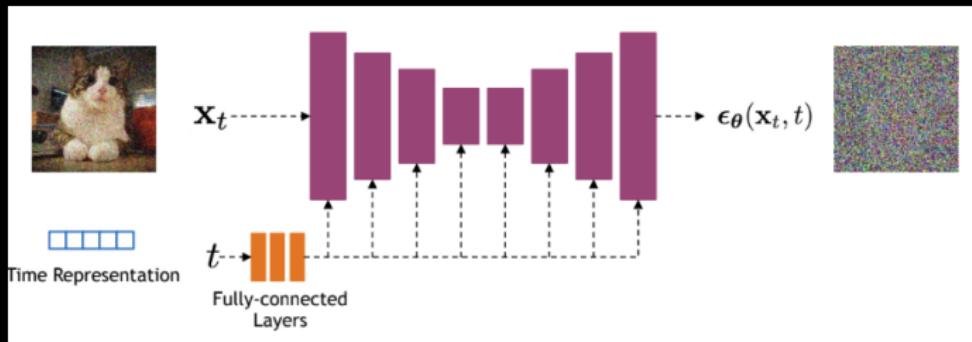


Figure 7: Image credit: Arash Vahdat

Denoising Diffusion Models

Some samples from a trained DDPM model [Ho et al., DDPM, 2020]



Connections to Score-Based Models

Score-Based Models

Tweedie's Formula:

$$z \sim \mathcal{N}(\mu_z, \Sigma_z) \quad (48)$$

$$\mathbb{E}[\mu_z | z] = z + \Sigma_z \nabla_z \log p(z) \quad (49)$$

Suppose we have a noisy measurement

$$z = x + \epsilon \quad \epsilon \sim \mathcal{N}(0, \Sigma) \quad (50)$$

Then Tweedie's formula says:

$$\mathbb{E}[x | z] = \int xp(x | z) dx = x + \Sigma \nabla_x \log p(x) \quad (51)$$

If you know $\nabla_x \log p(x)$, you don't need to know $p(x | z)$!

- The gradient $\nabla_x \log p(x)$ is called the **score** of $p(x)$

Score-Based Models

Tweedie's Formula:

$$z \sim \mathcal{N}(\mu_z, \Sigma_z) \implies \mathbb{E}[\mu_z | z] = z + \Sigma_z \nabla_z \log p(z) \quad (52)$$

Recall our forward process:

$$q(x_t | x_0) = \mathcal{N}(x_t | \sqrt{\gamma_t}x_0, (1 - \gamma_t)I) \quad (53)$$

$$x_t = \sqrt{\gamma_t}x_0 + \sqrt{1 - \gamma_t}\epsilon \quad \epsilon \sim \mathcal{N}(0, I) \quad (54)$$

By Tweedie's formula:

$$\mathbb{E}[x_0 | x_t] = \frac{1}{\sqrt{\gamma_t}} \left(x_t + \sqrt{1 - \gamma_t} \nabla \log p(x_t) \right) \quad (55)$$

Score-Based Models

Recall our setup:

$$L_{t-1} := \mathbb{E}_q \mathsf{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)] \quad (56)$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t) I) \quad (57)$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (58)$$

By Tweedie's formula:

$$\mu_q(x_t, x_0) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1 - \gamma_t} x_0 + \frac{\sqrt{1 - \beta_t}(1 - \gamma_{t-1})}{1 - \gamma_t} x_t \quad (59)$$

$$= \frac{1}{\sqrt{1 - \beta_t}} x_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} \nabla \log p(x_t) \quad (60)$$

Thus we have an alternative parametrization:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1 - \beta_t}} x_t + \frac{\beta_t}{\sqrt{1 - \beta_t}} s_\theta(x_t, t) \quad (61)$$

$$s_\theta(x_t, t) \approx \nabla \log p(x_t) \quad (62)$$

Score-Based Models

Further connections:

$$s_\theta(x_t, t) \approx \nabla \log p(x_t) = \int q(x_0) \nabla \log q(x_t | x_0) x_0 \quad (63)$$

$$= \int q(x_0) \left(-\frac{x_t - x_0}{1 - \gamma_t} \right) \quad (64)$$

$$= -\mathbb{E}_{x_0} \left(\frac{\epsilon}{\sqrt{1 - \gamma_t}} \right) \quad \epsilon \sim \mathcal{N}(0, 1) \quad (65)$$

That is:

$$s_\theta(x_t, t) \approx -\frac{1}{\sqrt{1 - \gamma_t}} \epsilon_\theta(x_t, t) \quad (66)$$

Predicting the score is the same (up to a time-dependent constant) as predicting the noise

Score-Based Models

Thus an alternative form of the loss is:

$$L_{t-1} = \mathbb{E}_q [C_t || s_\theta(x_t, t) - \nabla \log p(x_t) ||_2^2] \quad (67)$$

i.e. we can predict the **score** rather than the added noise

Note that

$$\nabla \log p(x_t) = \int q(x_0) \nabla p(x_t | x_0) x_0 \quad (68)$$

is intractable as written

- Requires specialized techniques for **score-matching**
- Beyond the scope of this lecture

Conditional Diffusion Models

Conditioning information c , e.g.

- text (embedding)
- class label
- etc.

Condition reverse chain on c :

$$p_{\theta}(x_{0:T} \mid c) = p_{\theta}(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1} \mid x_t, c) \quad (69)$$

Loss can be derived in an analogous way:

$$\begin{aligned} \log p_{\theta}(x_0 \mid c) &\geq \mathbb{E}_q \left[\log p_{\theta}(x_0 \mid x_1, c) - \text{KL} [q(x_T \mid x_0) \parallel p_{\theta}(x_T)] \right. \\ &\quad \left. - \sum_{t=2}^T \text{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_{\theta}(x_{t-1} \mid x_t, c)] \right] \end{aligned}$$

Conditional Diffusion Models

Condition reverse chain on c :

$$p_{\theta}(x_{0:T} \mid c) = p_{\theta}(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1} \mid x_t, c) \quad (70)$$

Loss can be derived in an analogous way:

$$\begin{aligned} \log p_{\theta}(x_0 | c) &\geq \mathbb{E}_q \left[\log p_{\theta}(x_0 \mid x_1, c) - \text{KL} [q(x_T \mid x_0) \parallel p_{\theta}(x_T)] \right. \\ &\quad \left. - \sum_{t=2}^T \text{KL} [q(x_{t-1} \mid x_t, x_0) \parallel p_{\theta}(x_{t-1} \mid x_t, c)] \right] \end{aligned}$$

- Basic idea still holds for conditional models
- Challenge: building architectures to best make use of c
- Score-based models can be conditioned via guidance

Conclusions and Summary

Conclusions and Summary

Diffusion Generative Models are a class of deep generative models that generate data by iterative denoising.

- Can be applied to a wide array of conditional and unconditional generation tasks
- The forward process is a Markov chain that turns our data into noise
- We learn to undo this procedure via a variational approximation to the time-reversed chain

Conclusions and Summary

Diffusion Generative Models are a class of deep generative models that generate data by iterative denoising.

There are many complementary perspectives on diffusion models:

- Hierarchical VAEs; Latent variable models
- From x_t , predicting:
 - Denoised input x_0
 - Added noise ϵ
 - Score $\nabla \log p(x_t)$

Conclusions and Summary

Not covered today:

- A lot!
- Improvements to forward process [Kingma et al., Variational Diffusion Models, NeurIPS 2021]
- Techniques to speed up generation [Song et al., Denoising Diffusion Implicit Models, ICLR 2021]
- Conditional generation methods [Ho et al., Classifier Free Guidance]

Thanks!

gavin.k@uci.edu

gavinkerrigan.github.io