Diffusion Generative Models

CS 274E Guest Lecture

Gavin Kerrigan

2023 16 November

gavin.k@uci.edu

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

Unconditional generation: Images

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

Unconditional generation: Point clouds

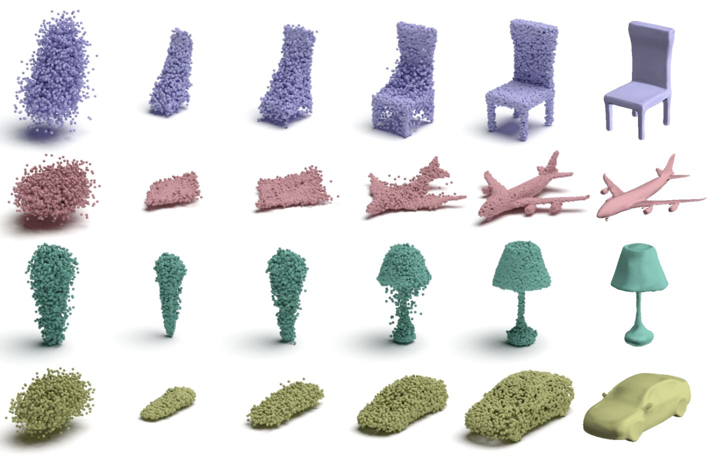


Figure 2: [Cai et al., Learning Gradient Fields for Shape Generation, ECCV 2020]

Introduction

Conditional generation: Text $\rightarrow$ 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 $\rightarrow$ Image

- e.g. Super-resolution

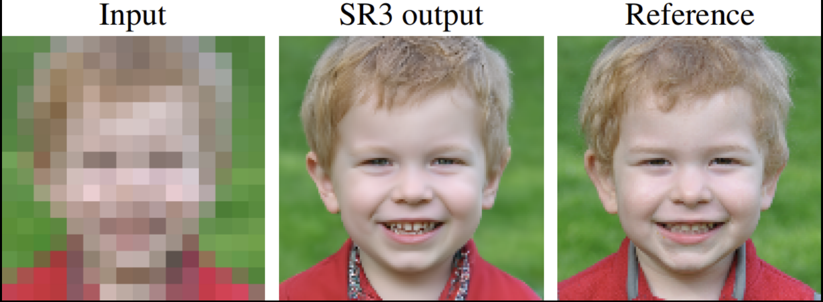
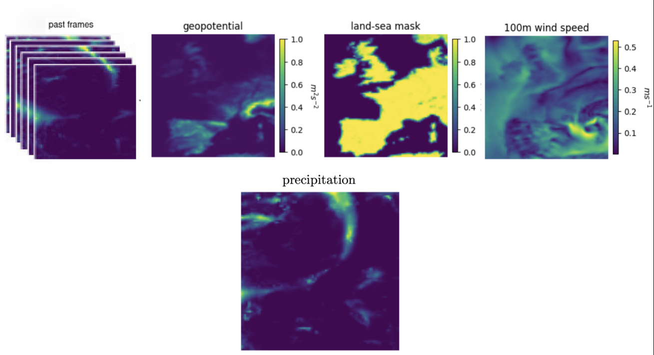


Figure 4: [Saharia et al., Image Super-Resolution via Iterative Refinement, ICCV 2021]

Conditional generation: Image → Image

- Precipitation forecasting



[Asperti et al., 2023]

Introduction

Conditional generation: Graphs → 3D Molecules

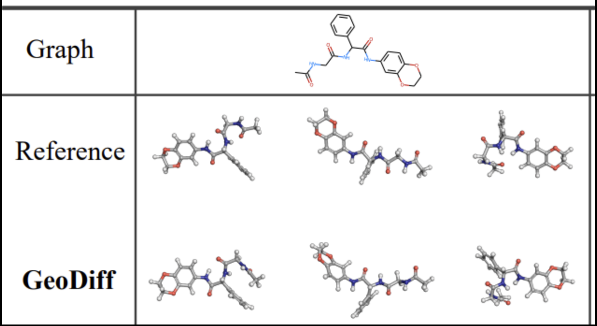


Figure 6: *[Xu et al., GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation, ICLR 2022]*

Conditional generation: Graphs → 3D Molecules

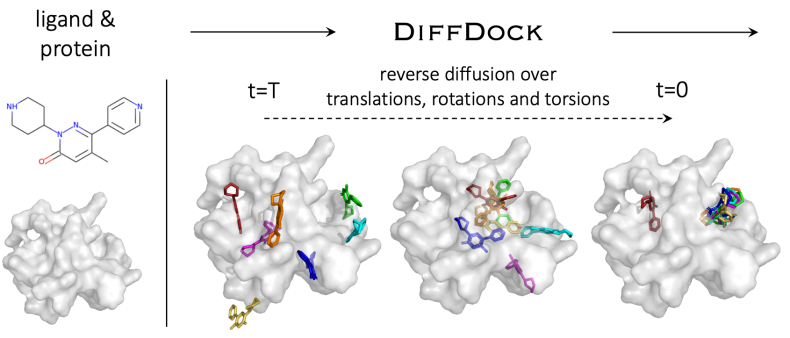


Figure 7: [Corso et al., DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking, ICLR 2023]

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)

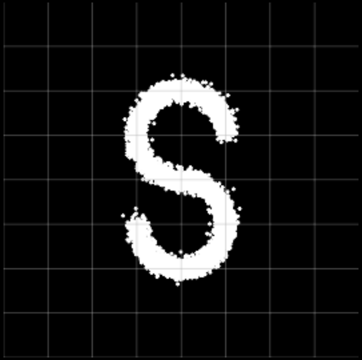
- Derivations from [Ho et al., DDPM, NeurIPS 2020] and [Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015]
- Based in part on Arash Vahdat's CVPR 2022 tutorial

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

"Creating noise from data is easy; creating data from noise is generative modeling"[2].

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

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



Forward Process

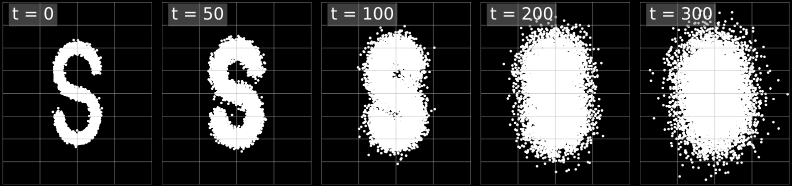
Creating noise from data is easy:

- 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

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

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

This defines the forward (diffusion) process:



Forward Process

Forward Process:

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

- Gives you a joint distribution

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

Similar to a latent variable model / VAE:

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

Forward Process:

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

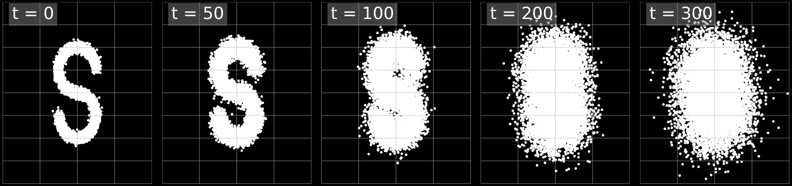
Cheap to sample at any time t. Set $\gamma_t = \prod_{s=1}^{t}(1-\beta_s)$. Then:

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

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t \mid \sqrt{\gamma_t}x_0, (1-\gamma_t)I\right) \tag{7}$$

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

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



"Creating noise from data is easy; creating data from noise is generative modeling"[3].

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

Reverse Process

Generate data by reversing the diffusion process

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

$$x_{t-1} \sim q(x_{t-1} \mid x_t) \qquad t = T, T-1, \ldots, 1 \tag{8}$$

Reverse Process

Generate data by reversing the diffusion process

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

$$x_{t-1} \sim q(x_{t-1} \mid x_t) \qquad t = T, T-1, \ldots, 1 \tag{9}$$

- Problem:

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

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

$$\begin{aligned} q(x_{t-1} \mid x_t) &\approx p_\theta(x_{t-1} \mid x_t) && (11) \\ &= \mathcal{N}(x_{t-1} \mid \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) && (12) \end{aligned}$$

Reverse Process

Variational approximation:

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

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

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

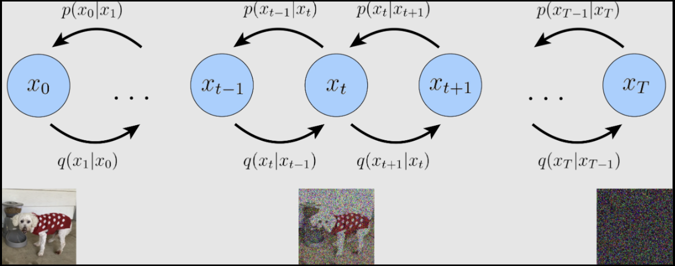


Figure 8: Image credit: Calvin Luo

Loss Analysis

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

$$\log p_\theta(x_0) \geq \mathbb{E}_{x_{1:T} \sim q(-|x_0)} \left[\log \frac{p_\theta(x_0, x_{1:T})}{q(x_{1:T} \mid x_0)} \right] \tag{15}$$

$$= \mathbb{E}_{x_{1:T} \sim q(-|x_0)} \left[\log p_\theta(x_0 \mid x_1) \right] - \mathsf{KL} \left[q(x_{1:T} \mid x_0) \mid\mid p_\theta(x_{1:T}) \right]$$

Let's analyze this to get something we can compute

Reminder: KL is the Kullback-Liebler divergence; a "distance" between probability distributions

$$\mathsf{KL} \left[q(x) \mid\mid p(x) \right] = \int \frac{q(x)}{p(x)} q(x) \, dx \tag{16}$$

KL Chain Rule

Chain rule for the KL divergence:

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

Proof (sketch):

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

KL Chain Rule

Chain rule for KL divergences:

$$\mathsf{KL}\left[p(x,y) \mid q(x,y)\right] = \mathsf{KL}\left[p(x) \mid\mid q(x)\right] + \mathbb{E}_{x \sim p(x)} \mathsf{KL}\left[p(y|x) \mid\mid q(y|x)\right] \quad (18)$$

Apply to the chain rule to condition on x_T:

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

KL Chain Rule

Chain rule for KL divergences:

$$\mathsf{KL}\left[p(x,y) \mid q(x,y)\right] = \mathsf{KL}\left[p(x) \mid\mid q(x)\right] + \mathbb{E}_{x\sim p(x)}\mathsf{KL}\left[p(y|x) \mid\mid q(y|x)\right] \quad (19)$$

Apply to the chain rule to condition on x_T:

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

Repeat to condition on $x_{T-1}, x_{T-2}, \ldots, x_1$:

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

Loss Analysis

Three types of terms appear in the loss:

$$L_T := \mathsf{KL}\left[q(x_T \mid x_0) \mid\mid p_\theta(x_T)\right] \tag{20}$$

$$L_0 := \mathbb{E}_q\left[\log p_\theta(x_0 \mid x_1)\right] \tag{21}$$

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t)\right] \tag{22}$$

Loss Analysis

$$L_T := \mathsf{KL}\left[q(x_T \mid x_0) \mid\mid p_\theta(x_T)\right] \tag{23}$$

This measures the error at the end of the forward process, i.e. $t = T$.

- We typically choose $p_\theta(x_T) = \mathcal{N}(0, I)$
- Note $q(x_T \mid x_0) \approx \mathcal{N}(0, I)$ for T sufficiently large
- Hence, L_T is negligible and is typically ignored during training

Loss Analysis

$$L_0 := \mathbb{E}_q \left[\log p_\theta(x_0 \mid x_1)\right] \tag{24}$$

Measures the error at the end of the backwards process, i.e. $t = 0$.

- This is essentially a decoder log-likelihood
- Analogous to the reconstruction term in a VAE
- Cheap to compute

Loss Analysis

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t)\right] \tag{25}$$

Measures the error between the at intermediate steps between

1. The model's reverse transitions $p_\theta(x_{t-1} \mid x_t)$
2. The true reverse transitions $q(x_{t-1} \mid x_t, x_0)$

Important note: the true reverse transitions are conditioned on x_0

- $q(x_{t-1} \mid x_t)$ is intractible
- ... but we'll see $q(x_{t-1} \mid x_t, x_0)$ is known!

Loss Analysis

By Bayes' rule:

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1}, x_0)\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \tag{26}$$

The right-hand side only involves the forward process

- ... so everything is known and Gaussian

After a tedious but straightforward calculation:

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

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

The story so far:

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t)\right] \tag{28}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t)\ I) \tag{29}$$

How should we parametrize the model $p_\theta(x_{t-1} \mid x_t)$?

Model Parametrization

The story so far:

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\right] \tag{30}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t)\ I) \tag{31}$$

How should we parametrize the model $p_\theta(x_{t-1} \mid x_t)$?

Since $q(x_{t-1} \mid x_t, x_0)$ is Gaussian, let's assume $p_\theta(x_{t-1} \mid x_t)$ is too:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{32}$$

- Now, we parametrize $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$

Model Parametrization

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\right] \tag{33}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(\mu_q(x_t, x_0), \sigma_q^2(t)\ I\right) \tag{34}$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right) \tag{35}$$

Let's make our lives easy and set

$$\Sigma_\theta(x_t, t) = \sigma_q(t)^2 I \tag{36}$$

The KL between Gaussians has a closed form:

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_q^2(t)} ||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2\right] + C \tag{37}$$

Model Parametrization

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} \left[||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2 \right] \right] \tag{38}$$

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, since this is known
- Can we do better?

Model Parametrization: Data Prediction

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} \left[||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2 \right] \right] \tag{39}$$

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

$$\mu_\theta(x_t, t) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1-\gamma_t} x_\theta(x_t, t) + \frac{\sqrt{1-\beta_t}(1-\gamma_{t-1})}{1-\gamma_t} x_t \tag{40}$$

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

$$x_\theta(x_t, t) \approx x_0 \tag{41}$$

Model Parametrization: Data Prediction

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} \left[||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2 \right] \right] \tag{42}$$

$$\mu_\theta(x_t, t) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1-\gamma_t} x_\theta(x_t, t) + \frac{\sqrt{1-\beta_t}(1-\gamma_{t-1})}{1-\gamma_t} x_t \tag{43}$$

$$x_\theta(x_t, t) \approx x_0 \tag{44}$$

Loss simplifies to

$$L_{t-1} = \mathbb{E}_q \left[C_t ||x_\theta(x_t, t) - x_0||^2 \right] \tag{45}$$

$$C_t = \frac{1}{2\sigma_q^2(t)} \frac{\gamma_{t-1}\beta_t^2}{(1-\gamma_t)^2} \tag{46}$$

Model Parametrization: Noise Prediction

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_q^2(t)} \left[||\mu_\theta(x_t, t) - \mu_q(x_t, x_0)||_2^2 \right] \right] \tag{47}$$

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:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \gamma_t}} \epsilon_\theta(x_t, t) \right) \tag{48}$$

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

$$\epsilon_\theta(x_t, t) \approx \epsilon \tag{49}$$

Denoising Diffusion Models: Noise Prediction

This parametrization results in a fairly simple and interpretable loss:

$$L_{t-1} = \mathbb{E}_{\epsilon}\left[C_t||\epsilon - \epsilon_\theta(x_t, t)||^2\right] \qquad \epsilon \sim \mathcal{N}(0, I) \tag{50}$$

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

$$C_t = \frac{\beta_t^2}{2\sigma_q^2(t)(1-\beta_t)(1-\gamma_t)} \tag{51}$$

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

Denoising Diffusion Models: Training

Putting everything together:

$$L = \mathbb{E}_q \left[\log p_\theta(x_0 \mid x_1) - \sum_{t=2}^{T} C_t \mathbb{E}_\epsilon ||\epsilon - \epsilon_\theta(x_t, t)||^2 \right] \tag{52}$$

Algorithm 1 Training

```
repeat
  x_0 ~ q(x_0)
  t ~ Uniform({1, ..., T})
  ε ~ N(0, I)
  Take gradient descent step on
      ∇_θ ||ε − ε_θ(√ᾱ_t x_0 + √(1 − ᾱ_t) ε, t)||^2
until converged
```

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

- 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 $\overline{\alpha}_t = \gamma_t$

Denoising Diffusion Models: Sampling

Sampling:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\gamma_t}}\epsilon_\theta(x_t, t)\right) \tag{53}$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_q^2(t) I) \tag{54}$$

Algorithm 2 Sampling

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

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

Denoising Diffusion Models

Some practical details:

- For images, $\epsilon_\theta(x_t, t)$ 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

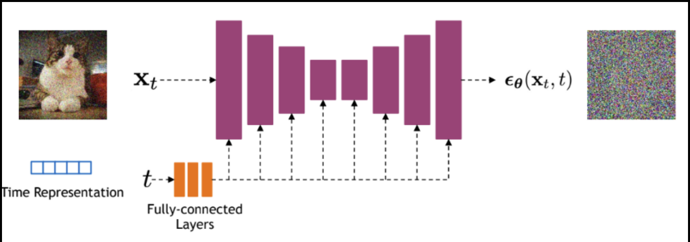


Figure 9: Image credit: Arash Vahdat

Denoising Diffusion Models

Some samples from a trained DDPM model

Figure 10: [Ho et al., DDPM, 2020]

Conditional Diffusion Models

Conditional Diffusion Models

Conditioning information c, e.g.

- text (embedding)
- class label
- image(s)

Condition reverse chain on c:

$$p_\theta(x_{0:T} \mid c) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t, c) \tag{55}$$

Loss can be derived in an analogous way:

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

Conditional Diffusion Models

Condition reverse chain on c:

$$p_\theta(x_{0:T} \mid c) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t, c) \tag{56}$$

Loss can be derived in an analogous way:

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

- Basic idea still holds for conditional models
- Challenge: building architectures to best make use of c

Conditional Diffusion Models

How do you model

$$p_\theta(x_{t-1} \mid x_t, c)? \qquad (57)$$

Some examples:

- Scalar c (e.g. class labels, time): pass through small MLP; mix with hidden layers

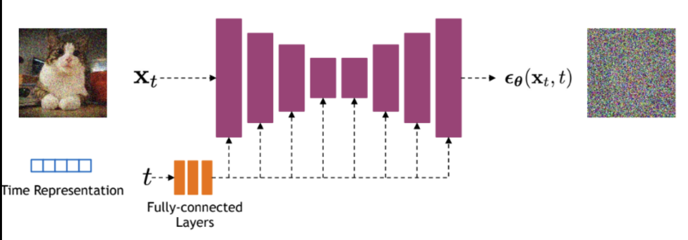


Figure 11: Image credit: Arash Vahdat

How do you model

$$p_\theta(x_{t-1} \mid x_t, c)? \tag{58}$$

Some examples:

- Image c: concatenate channel-wise with x_t
- Can be easily combined with scalar information

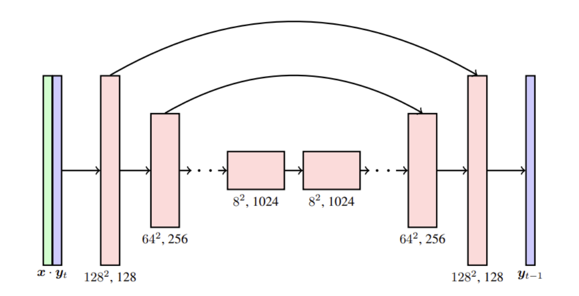


Figure 12: [Saharia et al., Image Super-Resolution via Iterative Refinement, 2021]

Conditional Diffusion Models

Case study: Imagen text-to-image model

- Text prompt embedded into a latent space (via T5)
- Cascaded image-to-image super resolution models

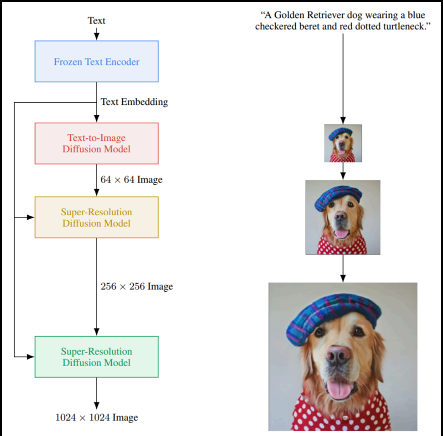


Figure 13: [Saharia et al., *Photorealistic Text-to-Image Diffusion Models...*, 2022]]

Connections to Score-Based Models

Tweeedie's Formula (1956):

$$z \sim \mathcal{N}(\mu_z, \Sigma_z) \tag{59}$$

$$\mathbb{E}[\mu_z \mid z] = z + \Sigma_z \nabla_z \log p(z) \tag{60}$$

Given a sample z from a Gaussian, our best guess for the mean is to perturb z in the direction that most increases the log density.

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

Tweeedie's Formula:

$$z \sim \mathcal{N}(\mu_z, \Sigma_z) \tag{61}$$

$$\mathbb{E}[\mu_z \mid z] = z + \Sigma_z \nabla_z \log p(z) \tag{62}$$

Suppose we have a noisy measurement

$$z = x + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \Sigma) \tag{63}$$

Then Tweedie's formula says:

$$\mathbb{E}[x \mid z] = \int x p(x \mid z)\, dx = z + \Sigma \nabla_x \log p(z) \tag{64}$$

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

Tweeedie's Formula:

$$z \sim \mathcal{N}(\mu_z, \Sigma_z) \implies \mathbb{E}[\mu_z \mid z] = z + \Sigma_z \nabla_z \log p(z) \tag{65}$$

Recall our forward process:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t \mid \sqrt{\gamma_t} x_0, (1-\gamma_t) I\right) \tag{66}$$

$$x_t = \sqrt{\gamma_t} x_0 + \sqrt{1-\gamma_t} \epsilon \qquad \epsilon \sim \mathcal{N}(0, I) \tag{67}$$

By Tweedie's formula:

$$\mathbb{E}[x_0 \mid x_t] = \frac{1}{\sqrt{\gamma_t}} \Big(x_t + \sqrt{1-\gamma_t} \nabla \log p(x_t) \Big) \tag{68}$$

Score-Based Models

Recall our setup:

$$L_{t-1} := \mathbb{E}_q \mathsf{KL}\left[q(x_{t-1} \mid x_t, x_0) \mid\mid p_\theta(x_{t-1} \mid x_t)\right] \tag{69}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\mu_q(x_t, x_0), \sigma_q^2(t)\ I) \tag{70}$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{71}$$

By Tweedie's formula (plug in for x_0):

$$\mu_q(x_t, x_0) = \frac{\sqrt{\gamma_{t-1}}\beta_t}{1-\gamma_t}x_0 + \frac{\sqrt{1-\beta_t}(1-\gamma_{t-1})}{1-\gamma_t}x_t \tag{72}$$

$$= \frac{1}{\sqrt{1-\beta_t}}x_t + \frac{\beta_t}{\sqrt{1-\beta_t}}\nabla \log p(x_t) \tag{73}$$

Thus we have an alternative parametrization:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}x_t + \frac{\beta_t}{\sqrt{1-\beta_t}}s_\theta(x_t, t) \tag{74}$$

$$s_\theta(x_t, t) \approx \nabla \log p(x_t) \tag{75}$$

Further connections:

$$s_\theta(x_t, t) \approx \nabla \log p(x_t) = \int q(x_0) \nabla \log q(x_t \mid x_0)\, dx_0 \tag{76}$$

$$= \int q(x_0) \left(-\frac{x_t - x_0}{1 - \gamma_t} \right) dx_0 \tag{77}$$

$$= -\mathbb{E}_{x_0} \left(\frac{\epsilon}{\sqrt{1 - \gamma_t}} \right) \qquad \epsilon \sim \mathcal{N}(0, 1) \tag{78}$$

$$= -\frac{\epsilon}{\sqrt{1 - \gamma_t}} \tag{79}$$

That is:

$$s_\theta(x_t, t) \approx -\frac{1}{\sqrt{1 - \gamma_t}} \epsilon_\theta(x_t, t) \tag{80}$$

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

Thus an alternative form of the loss is:

$$L_{t-1} = \mathbb{E}_q \left[C_t' || s_\theta(x_t, t) - \nabla \log p(x_t) ||_2^2 \right] \tag{81}$$

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

Note that

$$\nabla \log p(x_t) = \int q(x_0) \nabla p(x_t \mid x_0) x_0 \tag{82}$$

is intractable as written

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

Conditional Diffusion Models

Condition reverse chain on c:

$$p_\theta(x_{0:T} \mid c) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t, c) \tag{83}$$

Loss can be derived in an analogous way:

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

- 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!
- Continuous-time perspectives via Stochastic Differential Equations (SDEs)
- 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, 2022]

Additional Resources

- CVPR 2022 tutorial: *cvpr2022-tutorial-diffusion-models.github.io*
- Calvin Luo's blog: *calvinyluo.com/2022/08/26/diffusion-tutorial.html*
- Yang Song's blog: *yang-song.net/blog/2021/score/*

Thanks!

gavin.k@uci.edu

gavinkerrigan.github.io