

STA 414/2104:
Probabilistic Machine Learning
Week 12-2: Diffusion Models

Murat A. Erdogdu

University of Toronto

These slides are based on:

- CVPR 2022 Tutorial: Denoising Diffusion-based Generative Modeling: Foundations and Applications, by Kreis, Gao, and Vahdat
- Lilian Weng's blogpost: What are diffusion models?

Generative Modeling

Common methods:

- Variational Autoencoders
- Generative Adversarial Networks (GAN)
- Flow-based models
- Today: Diffusion Models

Diffusion Models: Text-to-Image Generation and More



DALL-E 3, prompt: Tiny potato kings wearing majestic crowns, sitting on thrones, overseeing their vast potato kingdom filled with potato subjects and potato castles.

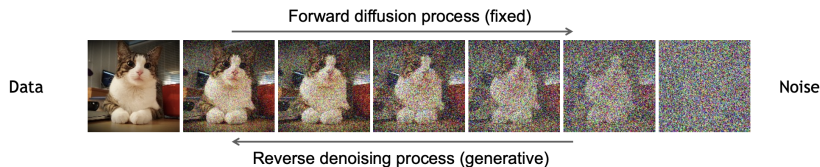


Stable Diffusion 3, prompt: Frog sitting in a 1950s diner wearing a leather jacket and a top hat. On the table is a giant burger and a small sign that says “froggy fridays”.

Diffusion Models

Diffusion models use two processes:

- A *forward process*, start from image and keep adding noise.
- A *reverse process*, start from noise and keep *denoising* it to recover an image.



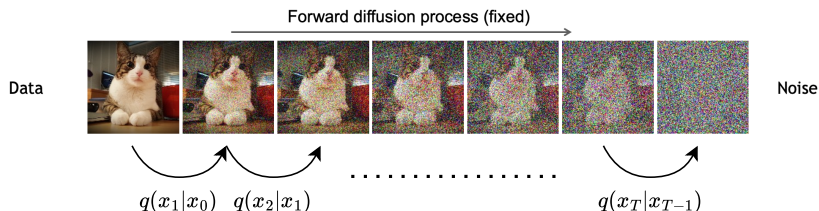
Forward Process

- The forward process is a Markov chain: $q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$
- Each step adds Gaussian noise:

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

Or equivalently,

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



Forward Process

Let $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^t \alpha_i$.

$$\begin{aligned}x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_{t-1} \\&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1 - \alpha_{t-1})}\epsilon_{t-2} + \sqrt{1 - \alpha_t}\epsilon_{t-1} \\&\stackrel{(d)}{=} \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).\end{aligned}$$

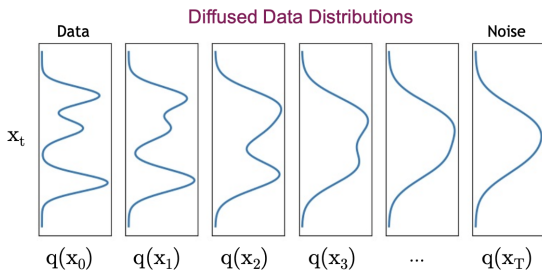
Therefore

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Forward Process and White Noise

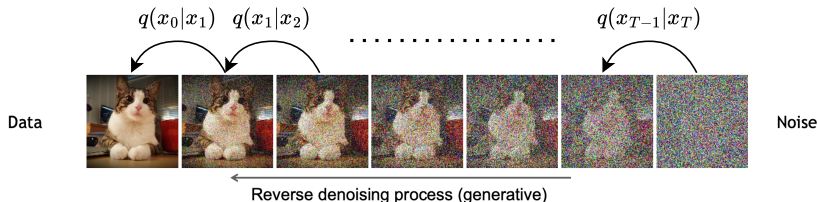
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

$(\beta_t)_{t=1}^T$ is chosen such that $\bar{\alpha}_T \rightarrow 0$, thus x_T converges to a standard normal random vector.



Reverse Process

- Ideally, to generate a sample:
 1. $x_T \sim \mathcal{N}(0, I) \approx q(X_T)$.
 2. $x_{t-1} \sim q(x_{t-1}|x_t)$ for $t = T, \dots, 1$.
- But $q(x_{t-1}|x_t)$ is intractable.



Reverse Process

- $q(x_{t-1}|x_t)$ is approximately normal if β_t is small.
- We approximate it with

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2).$$

- μ_θ comes from a trainable architecture (e.g. neural network) with parameter θ .
- For the reverse process

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t).$$

- To make $p_\theta(x_{1:T}|x_0)$ close to $q(x_{1:T}|x_0)$, we will use ideas from Variational Inference.

Evidence Lower Bound: ELBO

- Recall the ELBO:

$$KL(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)) + \mathbb{E}_{x_{1:T} \sim q} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \log p_\theta(x_0)$$

$$\implies \text{Minimize } \mathbb{E}_{x_{0:T} \sim q} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] =: L.$$

- Our goal becomes minimizing the loss L .

Variational Upper Bound

- While expressing $q(x_{t-1}|x_t)$ is difficult. By Bayes rule, expressing $q(x_{t-1}|x_t, x_0)$ is easy:

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ \implies q(x_t|x_{t-1}) &= \frac{q(x_t|x_0)q(x_{t-1}|x_t, x_0)}{q(x_{t-1}|x_0)}. \end{aligned} \quad (*)$$

Variational Upper Bound

$$\begin{aligned} L &= \mathbb{E}_{x_{0:T} \sim q} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \\ &= \mathbb{E}_{x_{0:T} \sim q} \left[\log \frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)} \right] \\ &= \mathbb{E}_{x_{0:T} \sim q} \left[\log \frac{q(x_T|x_0) \prod_{t=2}^T q(x_{t-1}|x_t, x_0)}{p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)} \right] \quad \text{By (*)} \\ &= \mathbb{E}_{x_{0:T} \sim q} \left[\underbrace{KL(q(x_T|x_0)||p_\theta(x_T))}_{l_T} + \sum_{t=2}^T \underbrace{KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{l_{t-1}} \right. \\ &\quad \left. + \underbrace{-\log p_\theta(x_0|x_1)}_{l_0} \right] \end{aligned}$$

Variational Upper Bound

- Let $L_t = \mathbb{E}_q[l_t]$ for $t = 0, \dots, T$.
- L_T is constant because x_T is just the standard normal random vector.
- To compute L_t for $t = 1, \dots, T - 1$, we first show $q(x_{t-1}|x_t, x_0)$ is Gaussian.

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &\propto q(x_t|x_{t-1})q(x_{t-1}|x_0) \\ &\propto \exp\left(-\frac{\|x_t - \sqrt{\alpha_t}x_{t-1}\|^2}{2\beta_t} - \frac{\|x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0\|^2}{2(1 - \bar{\alpha}_{t-1})}\right) \\ &\propto \exp\left(-\frac{\|x_{t-1} - \tilde{\mu}(x_t, x_0)\|^2}{2\tilde{\beta}_t}\right), \end{aligned}$$

Variational Upper Bound

- Therefore, $q(x_{t-1}|x_t, x_0) = \mathcal{N}(\tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$.
- Basic algebra shows

$$\begin{aligned}\tilde{\mu}(x_t, x_0) &= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 \\ \tilde{\beta}_t &= \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t.\end{aligned}$$

- Computing KL between two Gaussians is straightforward:

$$\begin{aligned}L_{t-1} &= \mathbb{E}_q [KL(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))] \\ &= \mathbb{E}_q \left[\frac{\|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|^2}{2\sigma_t^2} \right] + \text{const.}\end{aligned}$$

Parameterizing the Mean

- Recall $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$.
- By plugging in x_0 in terms of x_t and ϵ :

$$\tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right).$$

- As a result, we parameterize $\mu_\theta(x_t)$ to try to predict the noise using a neural network $\epsilon_\theta(x_t, t)$,

$$\mu_\theta(x_t, t) := \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

Training Objective

- The loss thus becomes

$$L_{t-1} = \underbrace{\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\alpha_t)}}_{\lambda_t} \mathbb{E}_{x_0 \sim q, \epsilon \sim \mathcal{N}(0, I)} [\|\epsilon_\theta(x_t, t) - \epsilon\|^2],$$

where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$.

- Ho et al. [2020] observed that the performance improves if we simply choose $\lambda_t = 1$.
- The simplified loss is thus

$$L_{t-1}^{\text{simple}} = \mathbb{E}_{x_0 \sim q, \epsilon \sim \mathcal{N}(0, I)} [\|\epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t) - \epsilon\|^2].$$

Test-Time Sample Generation

- Start from $x_T \sim \mathcal{N}(0, I)$.
- For $t = T, \dots, 1$, sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$
 - ▶ Recall $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2)$.
 - ▶ As a result

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z, \quad z \sim \mathcal{N}(0, I).$$

Summary

Algorithm 1 Training

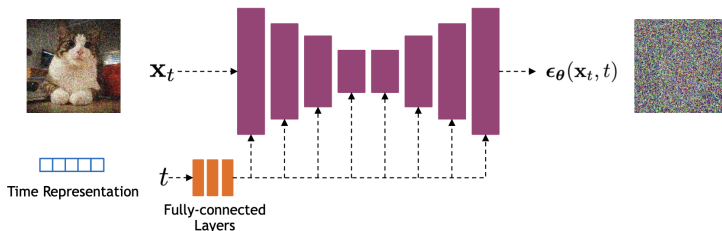
```
1: repeat  
2:  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$   
3:  $t \sim \text{Uniform}(\{1, \dots, T\})$   
4:  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
5: Take gradient descent step on  
    $\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

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:  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$   
5: end for  
6: return  $\mathbf{x}_0$ 
```

Design Choices: Architecture

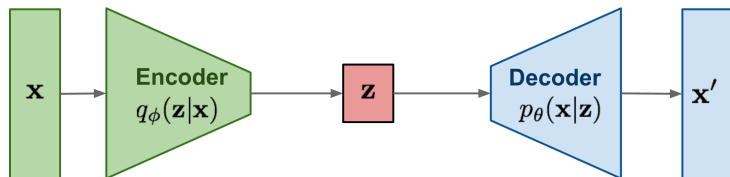
- $\epsilon_{\theta}(x_t, t)$ is implemented using a **U-Net** architecture, with **residual blocks** and **self-attention** layers.
- Weights are shared across time.



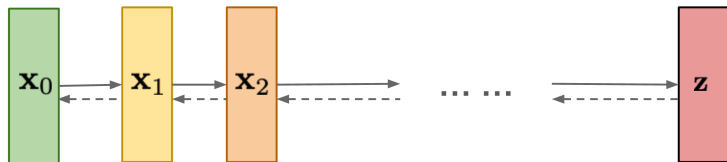
Design Choices: Hyperparameters

- β_t and σ_t control the variance of the forward and reverse process respectively.
- β_t is chosen linearly between $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$.
- $\sigma_t^2 = \beta_t$. We could instead consider a trainable full covariance matrix, i.e. $\Sigma_\theta(x_t, t)$.
- $T = 1000$ steps are taken.
- Fancier β_t schedules can further reduce loss [Nichol and Dhariwal, 2021].

Comparison with Variational Autoencoders (VAE)



VAE



Diffusion Model

Comparison with VAEs

- Diffusion models and VAEs both map to isotropic Gaussian.
- The latent space has the same dimension as the input space in DMs. In VAEs, it is smaller dimensional.
- The forward process is the encoder, which is **fixed**. This is trained in VAEs.
- The reverse process is the decoder, which is **trained**, similar to the VAEs.

Conditional Generation

- The original examples we saw were images generated conditioned on a text caption. More examples:



Midjourney, prompt: A close-up profile of a cute green-eyed kitten with a black nose and light cheeks sitting on top of a wooden floor under bright daylight.



DALL-E 3, prompt: Illustration of a chic chair with a design reminiscent of a pumpkin's form, with deep orange cushioning, in a stylish loft setting.

- But the diffusion model we learned about can only generate unconditioned images.

Conditional Generation: General Formulation

- Suppose we want to condition on y (e.g. class label or describing caption).
- The training data are pairs of (x_0, y) .
- Conditional reverse process:

$$p_{\theta}(x_{0:T}|y) = p(x_T) \prod_{t=1}^T p_{\theta}(x_{t-1}|x_t, y)$$

- We still model the transition probabilities as Gaussian:

$$p_{\theta}(x_{t-1}|x_t, y) = \mathcal{N}(\mu_{\theta}(x_t, t, c), \Sigma_{\theta}(x_t, t, y)).$$

Conditional Generation

- The new loss:

$$L = \mathbb{E}_q \left[l_T + \sum_{t \geq 2} KL(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t, \textcolor{red}{y})) - \log p_\theta(x_0|x_1, \textcolor{red}{y}) \right]$$

- How to incorporate y into the U-Net architecture?
 - ▶ Different techniques for different types of conditioning (labels, images, captions, ...)

Classifier Guidance

- To further strengthen conditioning, we can train a classifier $p_\phi(y|x)$ and incorporate its log-gradient into score with a scale s .
- [Nichol and Dhariwal, 2021]

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

```
Input: class label  $y$ , gradient scale  $s$ 
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$ 
for all  $t$  from  $T$  to 1 do
     $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$ 
     $x_{t-1} \leftarrow$  sample from  $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$ 
end for
return  $x_0$ 
```

Classifier-Free Guidance

- Instead of training a separate classifier, simultaneously train a conditional and an unconditional diffusion model.
- We have an implicit classifier by Bayes rule,

$$p(y|x_t) \propto_y \frac{p(x_t|y)}{p(x_t)}.$$

- We can simply use

$$\nabla_{x_t} \log p(y|x_t) = \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t).$$

Tradeoff: Sample Quality vs Diversity



$s = 0$



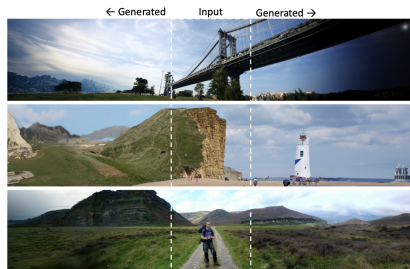
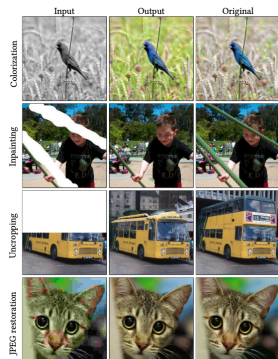
$s = 1$



$s = 3$

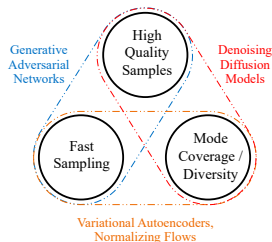
Conditioning Applications

Conditioning is not always on captions.



[Saharia et al., 2022]

Tradeoffs in Generative Modeling



[Xiao et al., 2021]

- Accelerating diffusion models can overcome the above trilemma.

- Diffusion Models: Forward and Reverse Processes
- Training via Variational Upper Bounds
- Conditional Generation
- Tradeoffs

Topics for Further Reading

- Denoising Score Matching [Song et al., 2021]
- Probability Flow ODE [Song et al., 2021]

References I

- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.
- Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.
- Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.
- Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.