Carnegie Mellon University

Lecture 2: Denoising Diffusion Models

Yutong (Kelly) He

10-799 Diffusion & Flow Matching, Jan 15th, 2026

Modal aws

Housekeeping Announcements

- Homework 1 is out!
  - https://kellyyutonghe.github.io/10799S26/homework/
  - Due date: 1/24 Sat, Late Due date: 1/26 Mon
  - Training models take time! Start early!
- Modal is giving a guest lecture tomorrow (1/16) 5 PM SH 105
- We will be admitting students from the waitlist until Friday noon
  - We will send out the Modal coupons on Friday (on Discord)
  - Auditing students don't need to submit any form!
- We shall have our Quiz 1 next class (1/20 Tue)

What is probabilistic modeling?

I don't do absolutes, I do if buts and maybes

Carnegie Mellon University

Image from The Redmen TV

Generative modeling

Given a set of data {x} and some prior knowledge & assumptions

- **Data:** samples (e.g. images of bedrooms)
- **Prior knowledge & assumptions:** parametric form, loss function, optimization, etc

We want to learn a probability distribution $p_\theta(x)$ such that

- **Generation:** If we sample a new datapoint from $p_\theta(x)$, it'd look like a "real" sample (e.g. looks like a real image of bedroom)
- **Density estimation:** Given an existing datapoint x, we should be able to assign a probability to it (probability should be high if x looks "real")
- **Unsupervised learning:** We learn everything by just looking at the data

Carnegie Mellon University

Attempt 1: Autoregressive modeling

Given dataset $\{x^{(i)}\}$,

$$L(\theta) = -\sum_i \sum_k \log p_\theta(x_k^{(i)} | x_{<k}^{(i)})$$

$$= -\sum_i \sum_k (\log p_\theta \left(x_k^{(i)} \middle| x_{<k}^{(i)} \right) \cdot 1 + \sum_{x' \neq x_k^{(i)}} p_\theta(x' | x_{<k}^{(i)}) \cdot 0)$$

which is cross entropy loss when ground truth labels are one-hot

This is LLM!

Carnegie Mellon University

Attempt 3: Catch me if you can

Generator

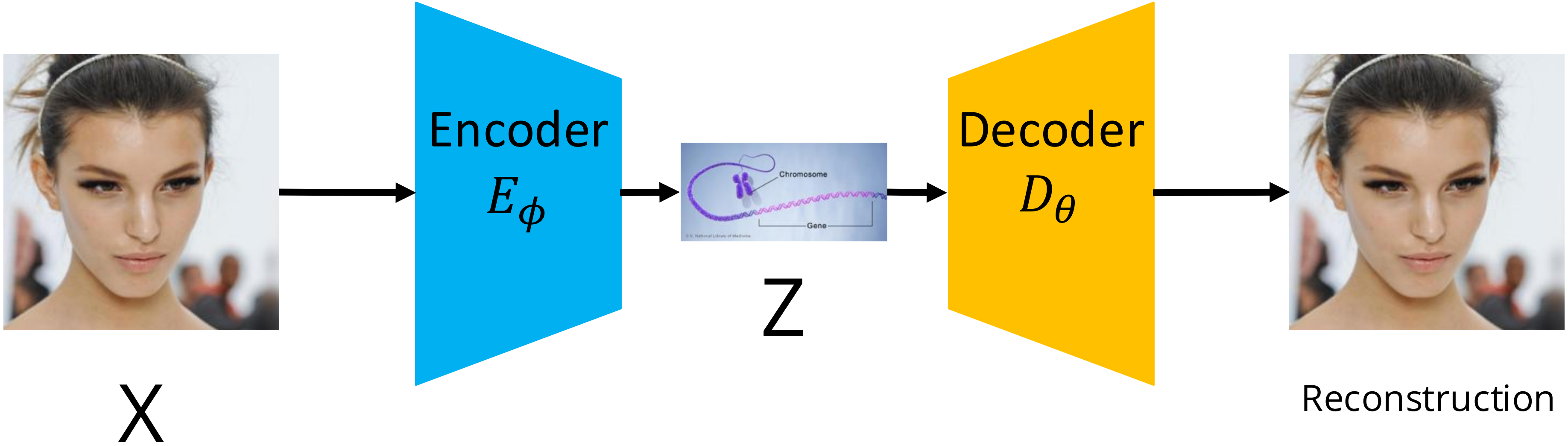
Tries to make the fake samples more and more realistic so that it can fool the discriminator

Discriminator

Tries to be better and better at distinguishing fake samples from real ones

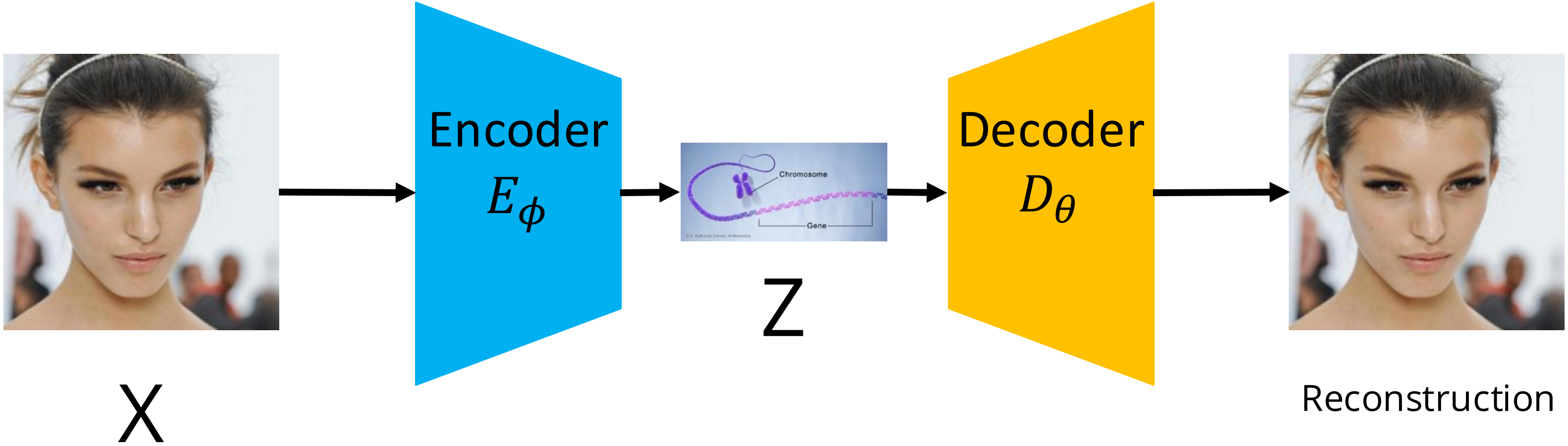
Carnegie Mellon University

Attempt 2: Variational autoencoder (VAE)



Kingma & Welling. "Auto-Encoding Variational Bayes". ICLR 2014. [*https://arxiv.org/pdf/1312.6114*](https://arxiv.org/pdf/1312.6114)

Attempt 2: Variational autoencoder (VAE)



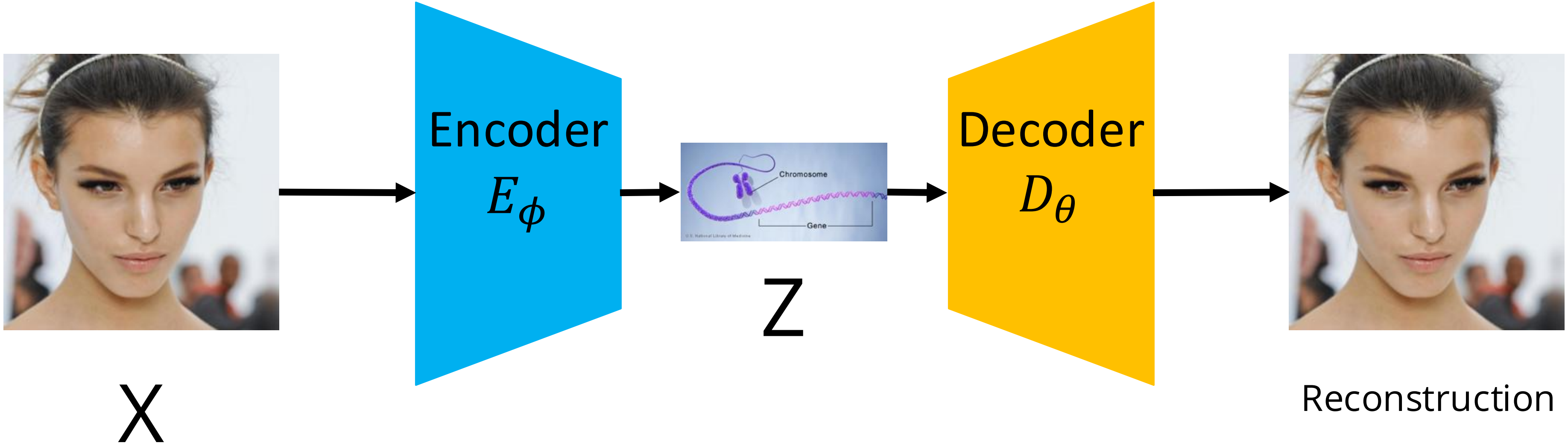
We need to do two things

- Maximize the likelihood of data X => maximize $\log p_\theta(x)$
- Make sure that the Z we get from encoding X can actually be decoded into the same X => minimize the "difference" between $q_\phi(z|x)$ and $p_\theta(z|x)$

Carnegie Mellon University

Kingma & Welling. "Auto-Encoding Variational Bayes". ICLR 2014. https://arxiv.org/pdf/1312.6114

Attempt 2: Variational autoencoder (VAE)



What we can design: $p_\theta(z), q_\phi(z|x), p_\theta(x|z)$

What we don't have: $p_\theta(x)$, $p_\theta(z|x)$

What we want: $p_\theta(x)$, $q_\phi(z|x)$, $p_\theta(z|x)$

Carnegie Mellon University

Kingma & Welling. "Auto-Encoding Variational Bayes". ICLR 2014. *https://arxiv.org/pdf/1312.6114*

VAE's evidence lower bound (ELBO)

We are trying to

- Maximize the likelihood of data X => maximize $\log p_\theta(x)$
- Make sure that the Z we get from encoding X can actually be decoded into the same X => minimize the "difference" between $q_\phi(z|x)$ and $p_\theta(z|x)$

$$\Rightarrow \text{argmax}_{\phi,\theta} \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x))$$

KL regularization

$$\Rightarrow \text{argmax}_{\phi,\theta} E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

Encode-Decode reconstruction loss

Evidence lower bound (ELBO)

Two ways to derive ELBO (1)

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x}) \\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))}
\end{aligned}
$$

Carnegie Mellon University

Kingma & Welling. "An Introduction to Variational Autoencoders". 2019. https://arxiv.org/pdf/1906.02691

Two ways to derive ELBO (1)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x}) \\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))}$$

$$E_{z\sim q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right] = E_{z\sim q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(z) + \log p_\theta(x|z) - \log q_\phi(z|x)]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - E_{z\sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z)]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

Carnegie Mellon University

Two ways to derive ELBO (2)

$$\log p_\theta(x) = \log \int p_\theta(x,z)\,dz$$

$$= \log \int \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x,z)\,dz = \log \mathrm{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

Jensen's Inequality

$$\geq \mathrm{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

For a convex function f, E[f(x)] >= f(E[x])

For a concave function f, f(E[x]) >= E[f(x)]

Not this Jensen!

This (Johan) Jensen

Carnegie Mellon University

How do we train a VAE

What we can design: $p_\theta(z), q_\phi(z|x), p_\theta(x|z)$

The ELBO:

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \; - \; E_{z\sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z)]$$

We need a sampler that we can differentiate through (so that we can take backprop)

We can choose $p_\theta(z)$ freely, so it better be something simple in log form

Reparameterization Trick

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - E_{z\sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z)]$$

Let's just choose the simplest $p_\theta(z)$ -- a standard normal Gaussian $N(0, I)$

Then what would be the easiest way to parametrize $q_\phi(z|x)$?

Also a Gaussian!

If $q_\phi(z|x)$ is a diagonal Gaussian, then we can literally write it out as $N\left(\mu_\phi(x), \sigma_\phi^2(x)I\right)$

Then we are literally just predicting two things: $\mu_\phi(x)$ and $\sigma_\phi^2(x)$

So to sample from $q_\phi(z|x)$, we can literally just do

1. Sample an $\epsilon \sim N(0, I)$
2. $z = \mu_\phi(x) + \sigma_\phi(x)\epsilon$

Carnegie Mellon University

Image edited by nano banana, originally from https://hgss.copernicus.org/articles/11/199/2020/

Reparameterization Trick

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \; - \; E_{z\sim q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z)\right]$$

If $q_\phi(z|x)$ is a diagonal Gaussian, then we can literally write it out as $N\left(\mu_\phi(x), \sigma_\phi^2(x)I\right)$

Then we are literally just predicting two things: $\mu_\phi(x)$ and $\sigma_\phi^2(x)$

So to sample from $q_\phi(z|x)$, we can literally just do

1. Sample an $\epsilon \sim N(0, I)$
2. $z = \mu_\phi(x) + \sigma_\phi(x)\epsilon$

Then you have the closed-form solution for the KL

$$E_{z\sim q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z)\right]$$

$$= \frac{1}{2}\sum_d \mu(x;\phi)_d^2 + \sigma(x;\phi)_d^2 - 1 - 2\log\sigma(x;\phi)_d$$

Carnegie Mellon University

Image edited by nano banana, originally from https://hgss.copernicus.org/articles/11/199/2020/

What about $p_\theta(x|z)$?

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \;-\; E_{z\sim q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z)\right]$$

We need to be able to calculate $\log p_\theta(x|z)$ easily as well

Also a Gaussian!

Let's assume $p_\theta(x|z) \sim N(\mu_\theta(z), \sigma^2 I)$, then the reconstruction term also has closed form solution

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

$$= \frac{1}{2} E_{z\sim q_\phi(z|x)}\left[-\frac{1}{2\sigma^2}||x - \mu_\theta(z)||^2\right] + C$$

$$\propto -E_{z\sim q_\phi(z|x)}\left[||x - \mu_\theta(z)||^2\right]$$

Carnegie Mellon University

How do we train a VAE

For existing data x,

1. Encode x and get $\mu_\phi(x)$ and $\sigma_\phi^2(x)$
2. Sample an $\epsilon \sim N(0, I)$
3. $z = \mu_\phi(x) + \sigma_\phi(x)\epsilon$
4. Calculate the loss

$$L(\phi, \theta; x) = \left|\left|x - \mu_\theta(z)\right|\right|^2 + \frac{1}{2}\sum_d \mu(x;\phi)_d^2 + \sigma(x;\phi)_d^2 - 1 - 2\log\sigma(x;\phi)_d$$

How to do sample from a VAE

At sampling time, all you need to do is

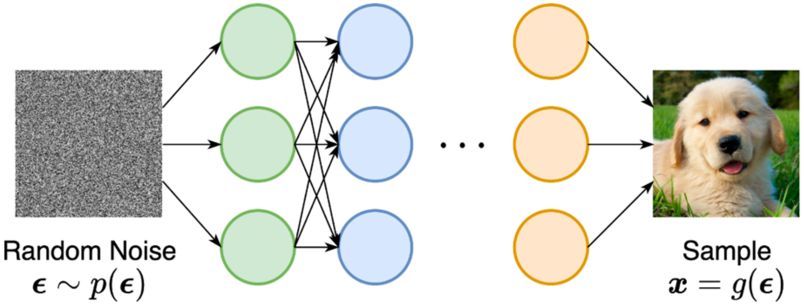
1. Sample an $z \sim N(0, I)$
2. Get $x = \text{Decoder}(z)$

That's it!

So far we have seen a bunch of generative models...

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM)
- **Likelihood Free:** Generative adversarial networks (GAN)

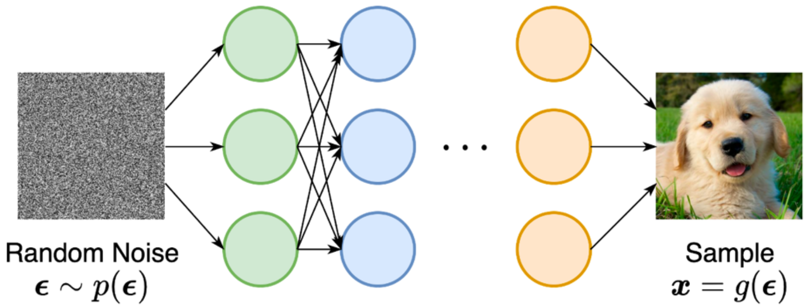


Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

Are they all perfect?

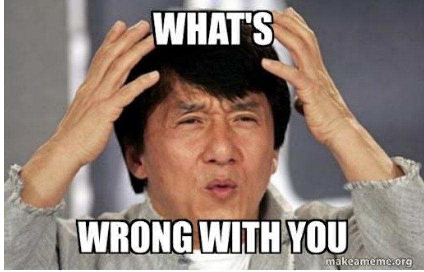
In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM)
- **Likelihood Free:** Generative adversarial networks (GAN)



Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

What’s wrong with previous models

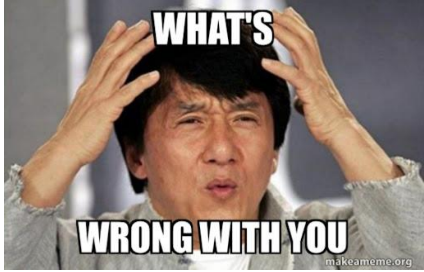


- **Autoregressive models:** you need to calculate stuff **one by one**
    - For text this may be ok (but your chatgpt is just gonna think for a very very long time ☺)
    - For image this is tragedy because this means you need to calculate things **pixel by pixel** (or maybe patch by patch, but same idea), if you have a 4k image, that means you need to do 3840 x 2160=8294400 forward passes of your model!

Carnegie Mellon University

Image from http://www.whatiswrongwithyou.info/uncategorized/08-what-is-wrong-with-you/

What’s wrong with previous models



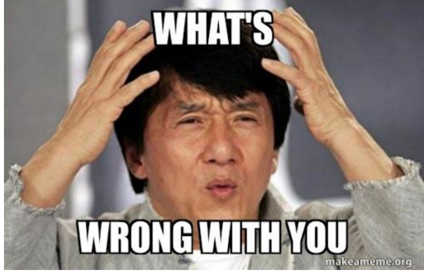
- **VAEs** are notorious for generating blurry images

> r/MachineLearning • 7y ago
> finallyifoundvalidUN
>
> **[D]Why are images created by GAN sharper than images by VAE?**
>
> Discussion
>
> There are different opinions . there is an opinion that the VAE loss function and in particular KL term is the main reason of getting such kind of blurry images, another one says the failure of VAE in generating sharp images implies that the model is not able to learn the true posterior distribution.
>
> GANs learn a loss function rather than using an existing one and they learn a loss that tries to classify if the output image is real or fake, while simultaneously training a generative model to minimize this loss but VAE kinda computes the euclidean distance between predicted and ground truth pixels which is not a good method of judging similarity so it yields blurry images

- If your encoder learns to map to different x’s into the same z region (which happens), then you are sort of just generating “average faces” all the time

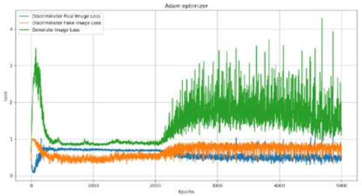
Screenshot from https://www.reddit.com/r/MachineLearning/comments/9t712f/dwhy_are_images_created_by_gan_sharper_than/

What’s wrong with previous models



- GANs are very unstable & suffers from mode collapse

- **Theorem (informal):** If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution
- **Unrealistic assumptions**!
- In practice, the generator and discriminator loss keeps oscillating during GAN training



Source: Mirantha Jayathilaka

- No robust stopping criteria in practice (unlike MLE)

- GANs are notorious for suffering from **mode collapse**
- Intuitively, this refers to the phenomena where the generator of a GAN collapses to one or few samples (dubbed as “modes”)

Arjovsky et al., 2017

Carnegie Mellon University

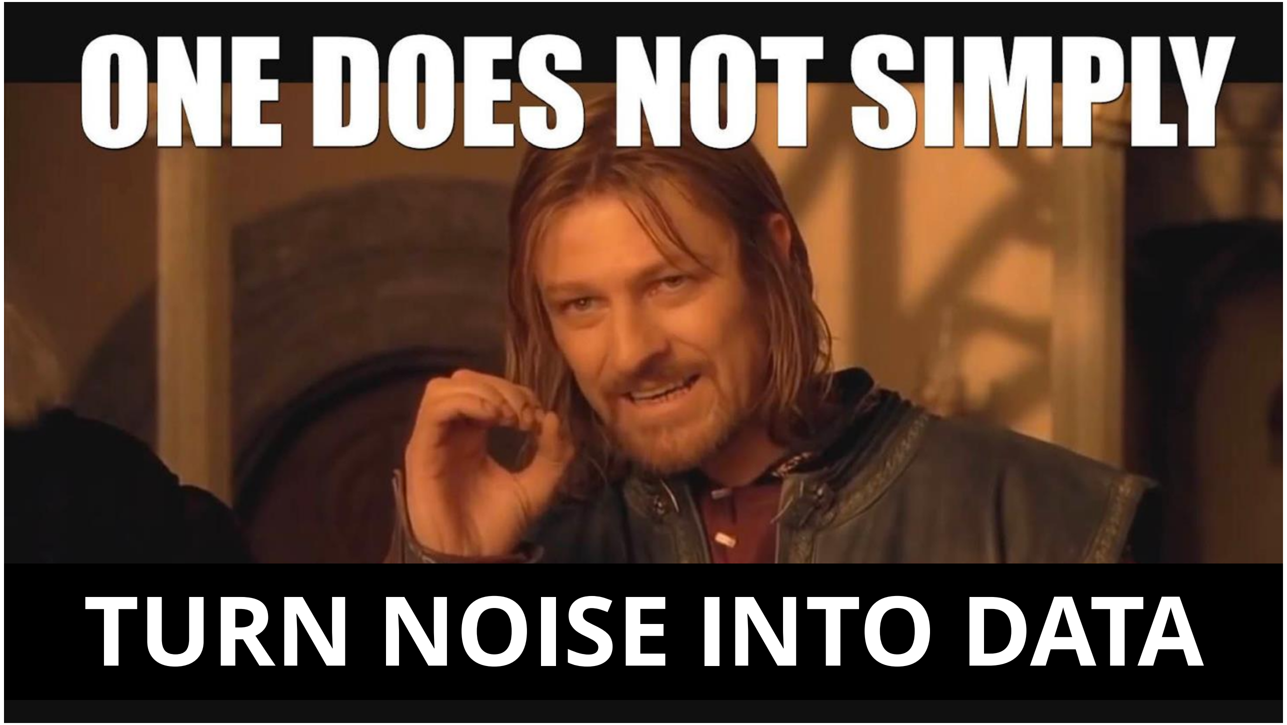
Screenshot from Stanford CS 236 by Prof. Stefano Ermon

From noise to data

"One ring to rule them all, …"

Healthier type of magic 🥬

From noise to data



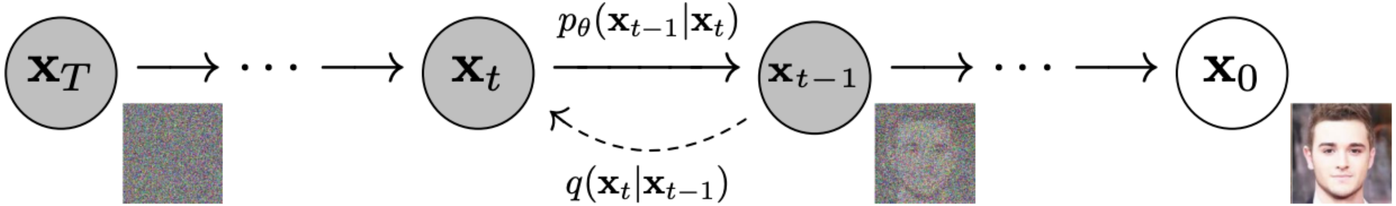
Carnegie Mellon University

Image from The Lord of The Rings

From noise to data

Now you have a diffusion model!

How do we model this process with math



Forward Process (adding noise):

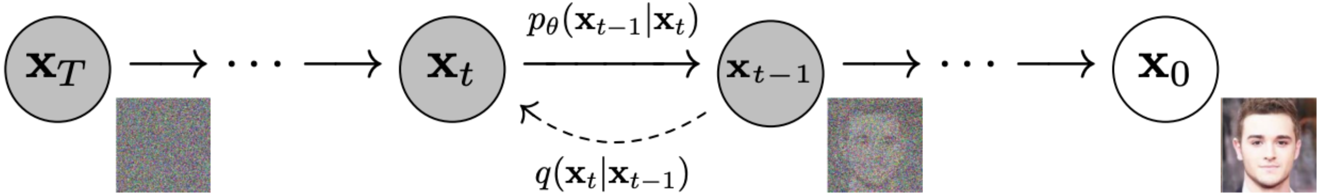
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Reverse Process (denoising):

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Carnegie Mellon University

How do we train this diffusion model



We want $\log p_\theta(x_0)$

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T}) dx_{1:T}$$

$$= \log \int q(x_{1:T}|x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} dx_{1:T}$$

$$= \log E_{q(x_{1:T}|x_0)}[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$$

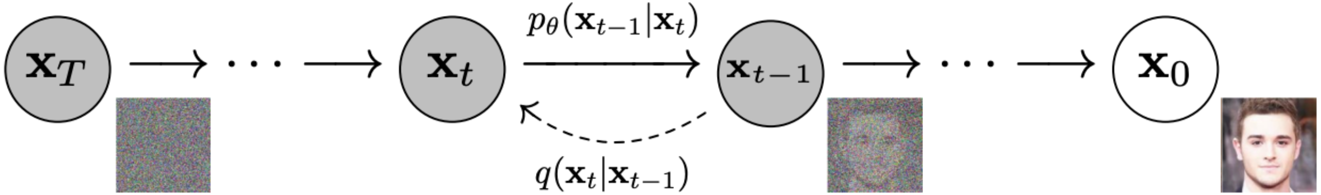
$$\geq E_{q(x_{1:T}|x_0)}[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$$

Jensen's Inequality

Carnegie Mellon University

Sohl-Dickstein et al. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. ICML 2015.

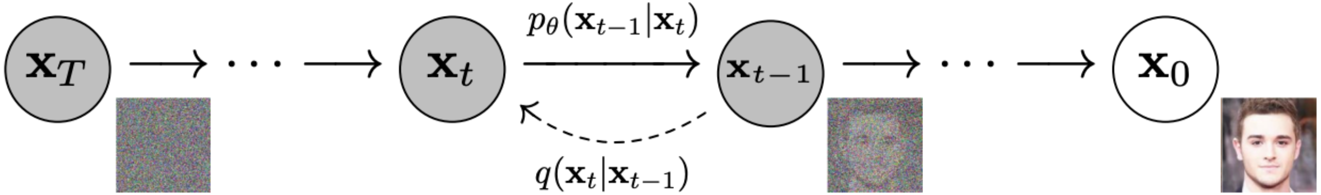
How do we train this diffusion model



$$\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]
\end{aligned}$$

Ho et al. Denoising Diffusion Probabilistic Models. NeurIPS 2020. https://arxiv.org/pdf/2006.11239

How do we train this diffusion model



$$L = \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]$$

$$= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]$$

$$= \mathbb{E}_q\left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,\|\, p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]$$

prior matching term

KL matching terms

reconstruction loss

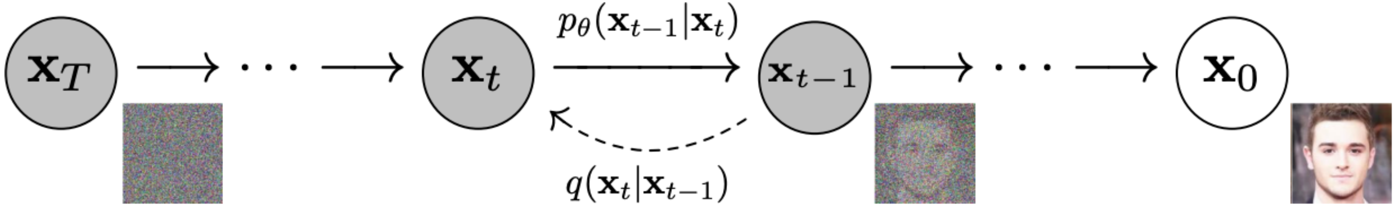
Carnegie Mellon University

Ho et al. Denoising Diffusion Probabilistic Models. NeurIPS 2020. https://arxiv.org/pdf/2006.11239

Good idea, but

Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. *(a)* Example training data. *(b)* Random samples generated by the diffusion model.

Can we do things in a simpler way?



So we know this thing is Markov

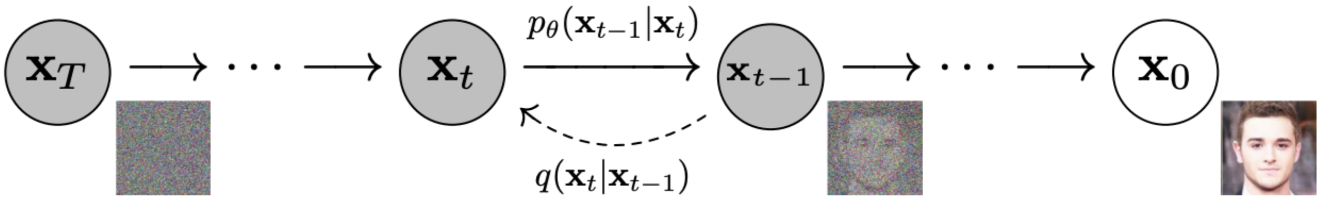
i.e. it just adds a small amount of noise at every time step

Then why don’t we just learn a noise predictor to predict the noise at each time step and then gradually reduce the noise?

Now you have DDPM!

Carnegie Mellon University

Denoising Diffusion Probabilistic Models



$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,\|\, p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

First of all, we can fix the forward process to make learning easier

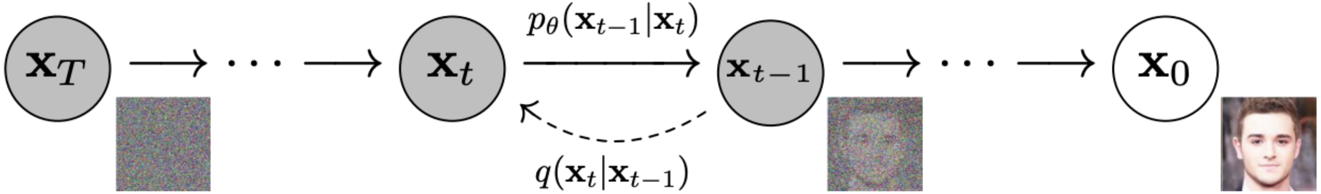
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$\alpha_t \coloneqq 1 - \beta_t \text{ and } \bar{\alpha}_t \coloneqq \prod_{s=1}^{t} \alpha_s$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Then L_T is constant because they are set to be standard Gaussians

Denoising Diffusion Probabilistic Models



$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big]$$

First of all, we can fix the forward process to make learning easier

$$\alpha_t := 1-\beta_t \text{ and } \bar{\alpha}_t := \textstyle\prod_{s=1}^t \alpha_s$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

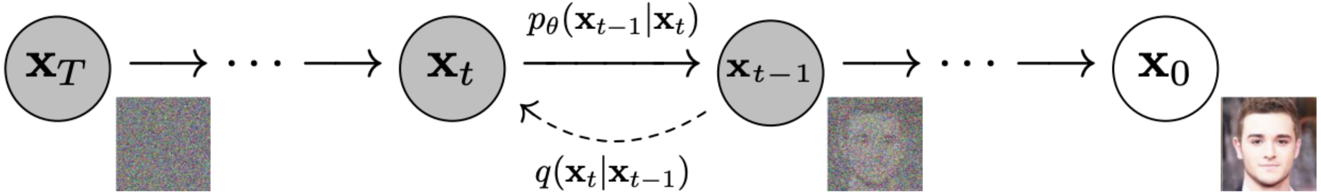
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0), \tilde{\beta}_t\mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Carnegie Mellon University

Ho et al. Denoising Diffusion Probabilistic Models. NeurIPS 2020. https://arxiv.org/pdf/2006.11239

Denoising Diffusion Probabilistic Models



$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big]$$

Similarly, we can also fix the variance of the reverse process and only learn the mean

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$$

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right] + C$$

Carnegie Mellon University

Ho et al. Denoising Diffusion Probabilistic Models. NeurIPS 2020. https://arxiv.org/pdf/2006.11239

Denoising Diffusion Probabilistic Models

$$\mathbb{E}_q\Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{L_T}+\sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Bigg]$$

Similarly, we can also fix the variance of the reverse process and only learn the mean

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)=\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$$

$$L_{t-1}=\mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right]+C$$

$$q(\mathbf{x}_t|\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_t;\sqrt{\bar{\alpha}_t}\mathbf{x}_0,(1-\bar{\alpha}_t)\mathbf{I})$$

Channeling reparameterization trick:

$$\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon})=\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\text{ for }\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$$

Carnegie Mellon University

Denoising Diffusion Probabilistic Models

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\right] + C$$

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \text{ for } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), t)\right\|^2\right]$$

Channeling high end Bayes Theorem:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) \coloneqq \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), t)\right\|^2\right]$$

Denoising Diffusion Probabilistic Models

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\underbrace{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)}_{\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)} - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t)\right\|^2\right]$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Now you have DDPM Training!

Denoising Diffusion Probabilistic Models

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right]$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)$$

$$\mathbf{x}_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)}_{\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)} + \sigma_t\mathbf{z}, \text{ where } \mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$$

Now you have DDPM sampling!

DDPM Algorithms

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
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** $\mathbf{x}_0$

Carnegie Mellon University

Image from https://makeameme.org/meme/thats-so-easy

The results are great!

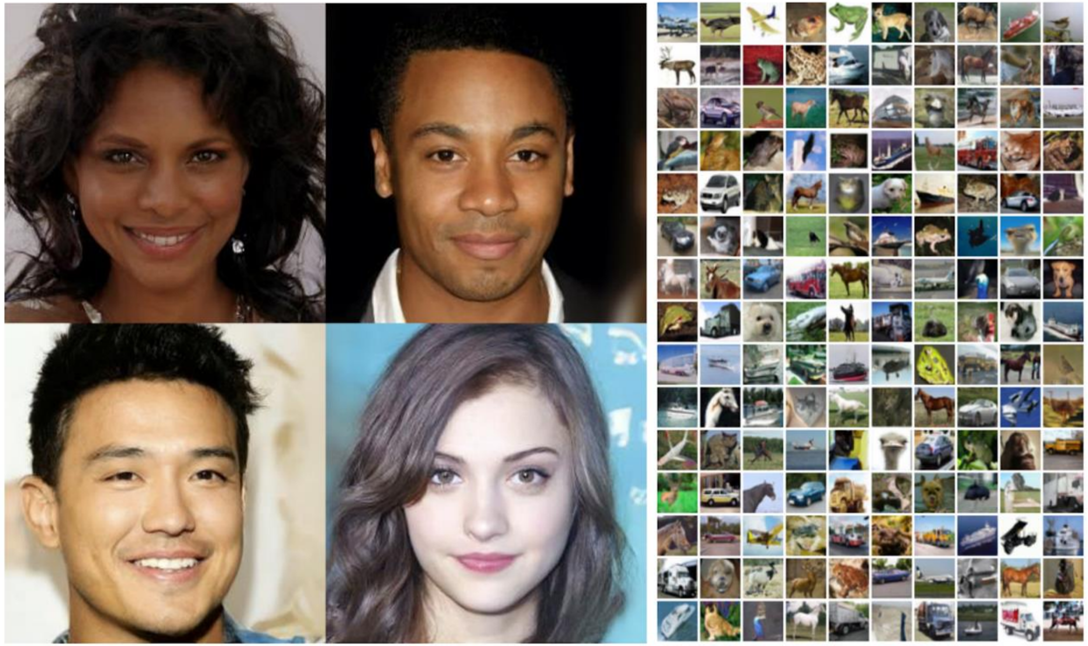
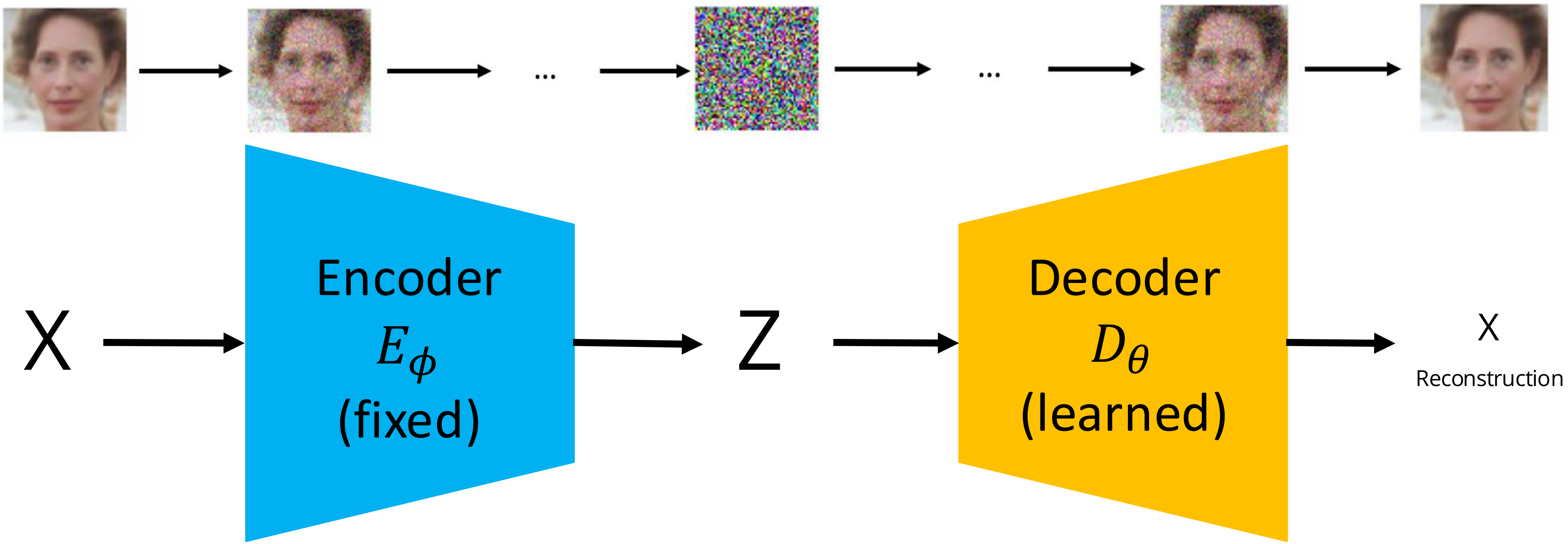


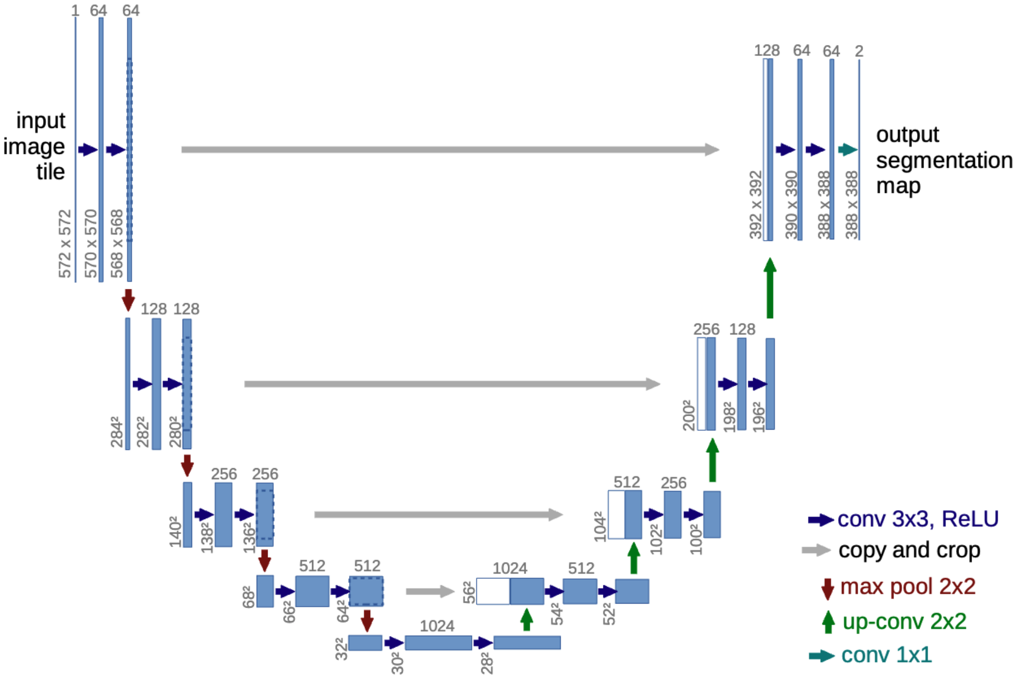
Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Diffusion models are lowkey VAEs



Carnegie Mellon University

Which model architecture was used for DDPM?

A  Net

Carnegie Mellon University

Ronneberger et al. U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2020. https://arxiv.org/abs/1505.04597

U-Net is good for diffusion (at least for now)

- Builds both coarse features (via downsampling) & fine features (via upsampling)
- Skip connections help preserve information
- Convolution is good inductive bias for images
- Easy to setup so that input and output are of the same spatial dimensionality

Later people have developed alternatives, but U-Net dominated the diffusion model architecture for the beginning years.

Re-reparametrization

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Channeling reparameterization trick:

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \text{ for } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

Equivalently

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$\mu_\theta(x_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathrm{x}_{0,\theta}(\mathrm{x}_t, \mathrm{t}) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathrm{x}_\mathrm{t}$$

Carnegie Mellon University

Re-reparametrization

Equivalently

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$\mu_\theta(x_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathrm{x}_{0,\theta}(\mathrm{x}_t, \mathrm{t}) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathrm{x}_t$$

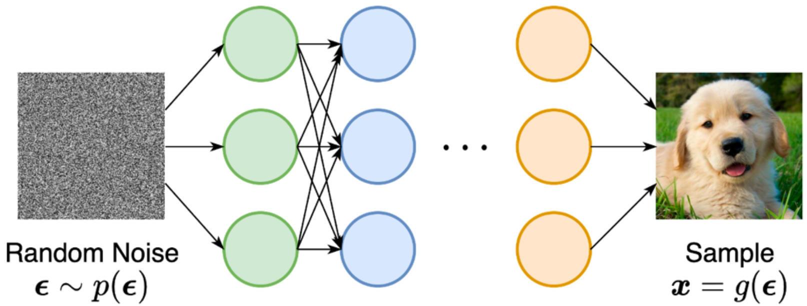
Plug in here!

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\right] + C$$

Now you know what a diffusion model is!

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM), **Diffusion model**
- **Likelihood Free:** Generative adversarial networks (GAN)



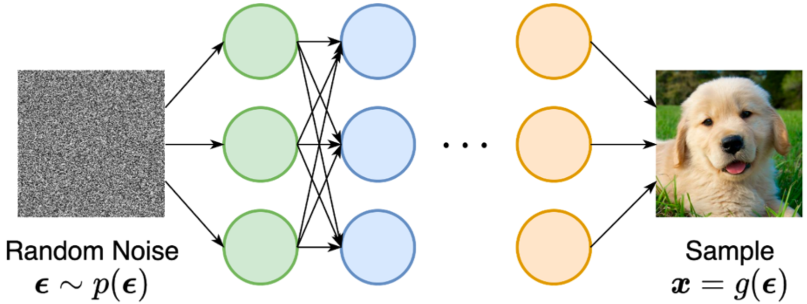
Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

Carnegie Mellon University

In the next class, we will derive the same diffusion model from a different perspective (with new techniques that we haven't seen so far)

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM), **Diffusion model**
- **Likelihood Free:** Generative adversarial networks (GAN), **Diffusion model?**



Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!