MACHINE LEARNING
@ BERKELEY

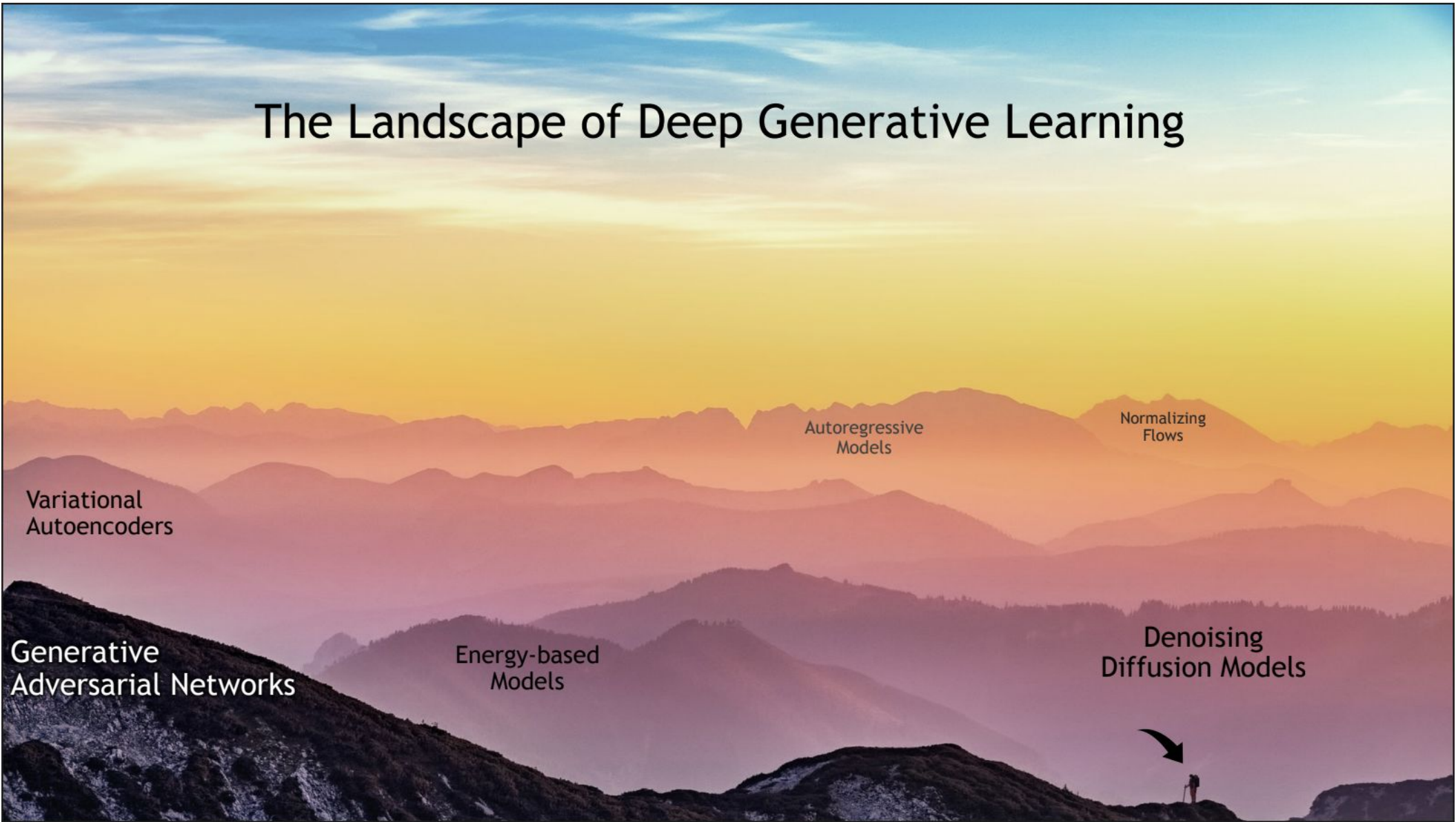
# Diffusion

By Aryan Jain

# Agenda

1. Theory of diffusion
2. Diffusion for image generation
3. Tricks to improve image synthesis models
4. Latent Diffusion Models
5. Examples of recent diffusion models

MACHINE LEARNING
@ BERKELEY


The Landscape of Deep Generative Learning
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models

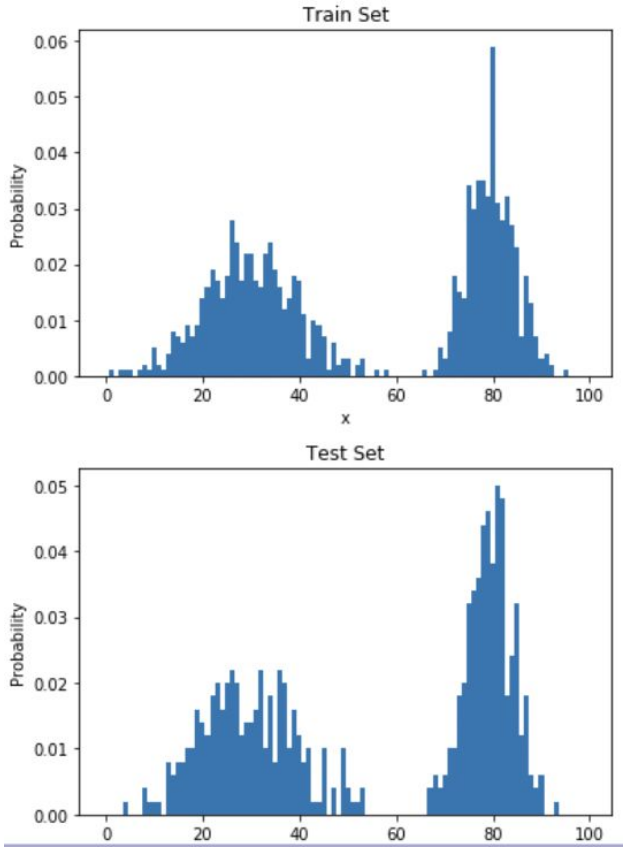
# Diffusion

MACHINE LEARNING
@ BERKELEY

# Density Modeling for Data Synthesis

Assume that all data comes from a distribution $p_{data}(x)$:

- The goal of generative machine learning models is to *learn* this distribution to the best of their ability — the distribution approximated by the model is denoted as $p_\theta(x)$ we want to learn
- We generate new data by *sampling* from the learned distribution
- In practice, train models to maximize the expected log likelihood of $p_\theta(x)$ (or minimizing negative log likelihood)/minimize divergence between $p_\theta(x)$ and $p_{data}(x)$



# Prior Methods

VAE:

$$
\begin{aligned}
L_{\text{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\text{VAE}}
\end{aligned}
$$

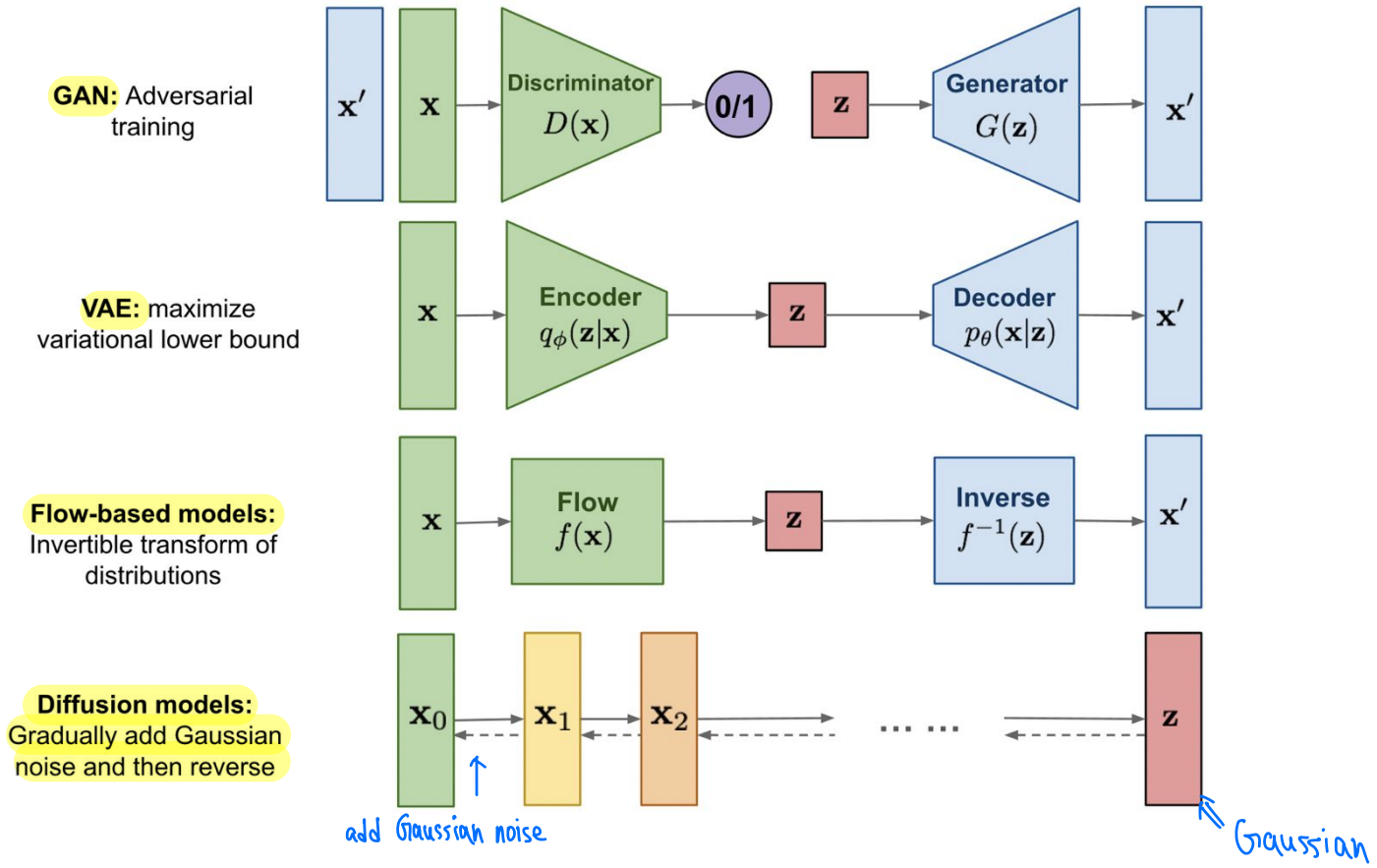
$$
\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}
$$

GAN:

$$
\begin{aligned}
\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
&= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]
\end{aligned}
$$

$$
\boxed{L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2}
$$

# Sampling from Noise



# Diffusion



- Another kind of generative modeling technique that takes inspiration from physics (non-equilibrium statistical physics and stochastic differential equations to be more exact)!
- Main idea: convert a well-known and simple *base distribution* (like a Gaussian) to the *target (data) distribution* iteratively, with small step sizes, via a Markov chain:
  - Treat the the output of the Markov Chain as the model's approximation for the *learned distribution*
  - Inspiration? Estimating and analyzing small step sizes is more tractable/easier than describing a single non-normalizable step from random noise to the learned distribution (which is what VAEs/GANs are doing)

# Anatomy of a Diffusion Model

1. Forward Process
2. Reverse Process

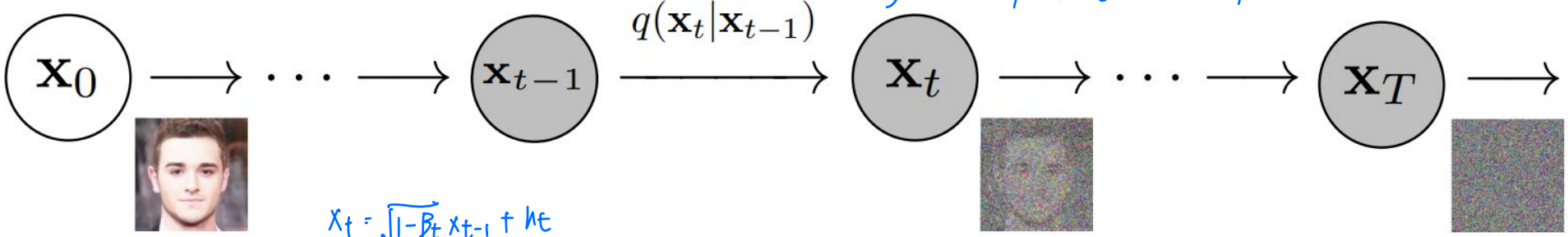
# Forward Process

$y = x + n$ (x ~ p(x), n ~ Gaussian)

$p_y(y) = x(y) \otimes n(y)$

$= \int x(y-t)\, n(t)\, dt$

$= \int x(t)\, n(y-t)\, dt$ ⇒ 縮小，圓弧化




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

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \{\beta_t \in (0,1)\}_{t=1}^{T}$$

- Take a datapoint x_0 and keep gradually adding very small amounts of Gaussian noise to it
  - Vary the parameters of the Gaussian according to a *noise schedule* controlled by beta_t
- Repeat this process for T steps — as the timesteps increase, the more features of the original input are destroyed
- You can prove with some math that as T approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)

# A neat (reparametrization) trick!

Define

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

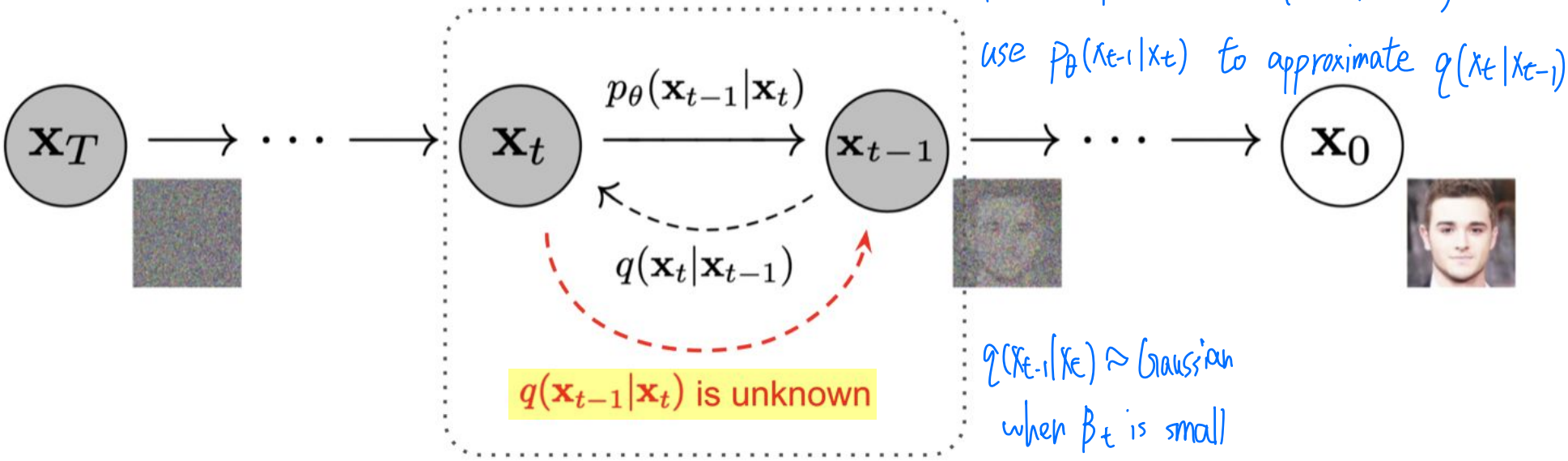
Then

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1},\ \beta_t\mathbf{I}\right)$$
$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}$$
$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbf{I}\right)$$

# Reverse Process

$$p(\underbrace{x_0, x_1, \ldots, x_T}_{z}) = p(x_T) \prod_{t=T}^{1} p(x_{t-1} | x_t)$$

MACHINE LEARNING @ BERKELEY




- The goal of a diffusion model is to **learn** the reverse *denoising* process to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

MACHINE LEARNING
@ BERKELEY

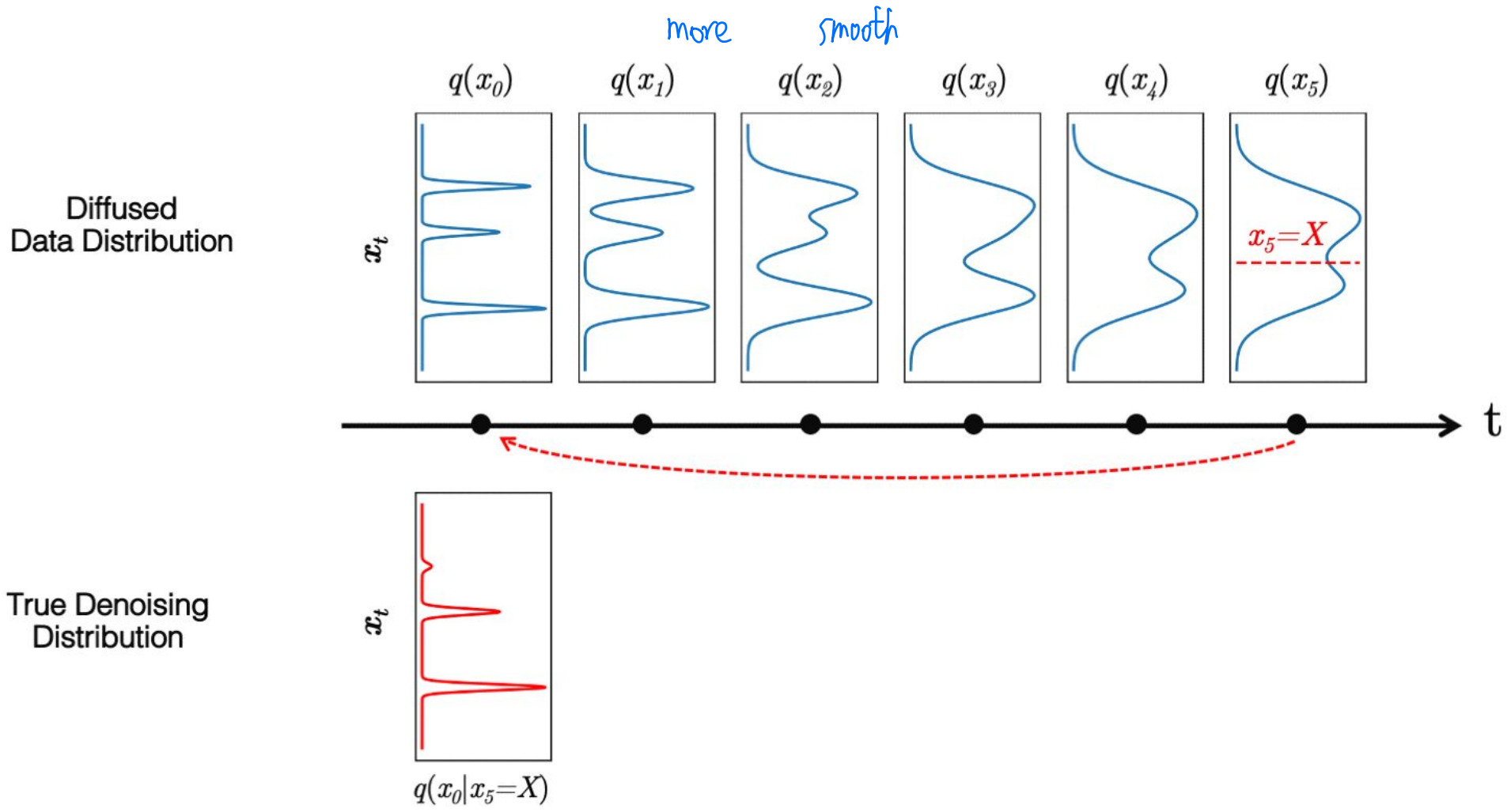
# Finding the exact distribution is hard

$$f(\theta \mid x) = \frac{f(\theta, x)}{f(x)} = \frac{f(\theta)\, f(x \mid \theta)}{f(x)} \quad \Longrightarrow \quad q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

$$q(x_t) = \int q(x_t \mid x_{t-1}) q(x_{t-1}) \mathrm{d}x$$

The distribution of each timestep and q(x_t | x_{t-1}) depends on the entire data distribution:

- Computing this is computationally intractable (where else have we seen this dilemma?)
- However, we still need those to describe the reverse process. Can we approximate them somehow?


more
smooth
$q(x_0)$
$q(x_1)$
$q(x_2)$
$q(x_3)$
$q(x_4)$
$q(x_5)$
Diffused
Data Distribution
$x_t$
$x_5=X$
t
True Denoising
Distribution
$x_t$
$q(x_0|x_5=X)$

# What should the distribution look like?

$q(x_{t-1}|x_t) = q(x_t|x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$ ← constant

↑ Gaussian

Turns out that for small enough forward steps, i.e.

$$\{\beta_t \in (0,1)\}_{t=1}^{T}$$

the reverse process step $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can be estimate is a Gaussian distribution too (take a course of stochastic differential equations if you want learn more)!

Therefore, we can parametrize the *learned* reverse process as

tries to learn this through neural network

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

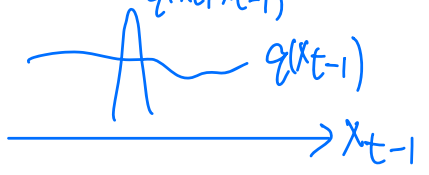
such that

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$q(x_t|x_{t-1}) q(x_{t-1})$ still Gaussian

⇑

$q(x_t|x_{t-1})$




$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \Rightarrow C \cdot \exp\left(\frac{1}{-2\beta_t}(x_t - \sqrt{1-\beta_t}x_{t-1})^T(x_t - \sqrt{1-\beta_t}x_{t-1})\right)$

# A preliminary objective

The VAE (ELBO) loss is a bound on the true log likelihood (also called the *variational lower bound*)

$$-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$

Apply the same trick to diffusion:

$$-\log p_\theta(\mathbf{x}_0) \leq \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] = L_{VLB}$$

Expanding out,

$$\begin{aligned} L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\ \text{where } L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\ L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\ L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \end{aligned}$$

$$-\log p_\theta(x_0) \le -\log p_\theta(x_0) + D_{KL}\left(q(x_{1:T}|x_0) \,\|\, p_\theta(x_{1:T}|x_0)\right)$$

$$= -\log p_\theta(x_0) + E_{q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)}\right], \quad \text{where } p_\theta(x_{1:T}|x_0) = \frac{p_\theta(x_{0:T})}{p_\theta(x_0)}$$

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

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

$$\Rightarrow \mathcal{L}_{VLB} = E_{q(x_0)}\left(E_{q(x_{1:T}|x_0)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]\right) = E_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

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

$$
\begin{aligned}
L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
$$

# A more thorough derivation

MACHINE LEARNING @ BERKELEY

$-\log q(x_1|x_0) + \log q(x_2|x_0)$

$-\log q(x_2|x_0) + \log q(x_3|x_0)$

$q(x_{0:T})$

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

$L_T$ $\qquad$ $L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \,\|\, p_\theta(x_t|x_{t+1}))$, for $1 \le t \le T-1$ $\qquad$ $L_0$

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

$t \ge 1 \rightarrow t > 1$

$q(x_t|x_{t-1})$

$= q(x_t|x_{t-1}, x_0)$

$= \dfrac{q(x_{t-1}|x_t, x_0)\, q(x_t|x_0)}{q(x_{t-1}|x_0)}$

$-\log q(x_1|x_0) + \log q(x_T|x_0)$

# A simplified objective

The reverse step conditioned on x_0 is a Gaussian:

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

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

After doing some algebra, each loss term can be approximated by

p and q variance same $\tilde{\beta}_t = \beta_t$

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2\right]$$

$$= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(\mathbf{x}_t, t)\right\|_2^2\right]$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right) \quad \alpha_t = 1 - \beta_t \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

proof :

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)}{q(x_t|x_0)} \frac{q(x_{t-1}|x_0)}{} \quad (\because \text{MDP} \Rightarrow q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1}))$$

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

$$\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{\alpha}_{t-1}} + \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{\alpha}_t}\right)\right)$$

$$= \exp\left(-\frac{1}{2}\left(\underbrace{\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2}_{\text{variance}} - \underbrace{\left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)}_{\text{mean}}x_{t-1} + C(x_t, x_0)\right)\right)$$

$$\mathcal{N}(\mu, \sigma) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}x^2 - \frac{2\mu}{\sigma^2}x + \frac{\mu^2}{\sigma^2}\right)\right)$$

$$\Rightarrow \frac{1}{\sigma^2} = \frac{1}{\tilde{\beta}_t} = \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = \frac{\alpha_t - \alpha_t\bar{\alpha}_{t-1} + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})} = \frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})} \quad \Rightarrow \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta$$

$$\Rightarrow \frac{2\mu}{\sigma^2} = \frac{2\tilde{\mu}_t(x_t, x_0)}{\tilde{\beta}_t} = \left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right) \quad \Rightarrow \tilde{\mu}(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

$\because x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t \quad \therefore x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t)$ 代入 $\tilde{\mu}(x_t, x_0)$

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

MACHINE LEARNING
@ BERKELEY

# A simplified objective

Instead of predicting the mu, Ho et al. say that we should predict epsilon instead!

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) \quad \Rightarrow \quad \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

Thus, our loss becomes

$$\begin{aligned}
L_{t-1} &= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)\right\|_2^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|_2^2\right]
\end{aligned}$$

original image back

It is trying to predict the noise that was added at 0~T, I can subtract noise to get the

# A simplified objective

The authors of DDPM say that it's fine to drop all that baggage in the front and instead just use

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right)\right\|_2^2\right]$$

Note that this is not a variational lower bound on the log-likelihood anymore: in fact, you can view it as a reweighted version of ELBO that *emphasizes* reconstruction quality!

$$L = L_T + L_{T-1} \cdots L_1 + L_0$$

$$\begin{aligned} L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\ \text{where } L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\ L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1 \\ L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \end{aligned}$$

where $L_T = D_{KL}(q(x_T|x_0) \| p(x_T))$ ∵ q don't have learnable parameter and $x_T$ is pure noise so omit this term as constant

# Training

**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

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

6: **until** converged

# Sampling

**Algorithm 2** Sampling

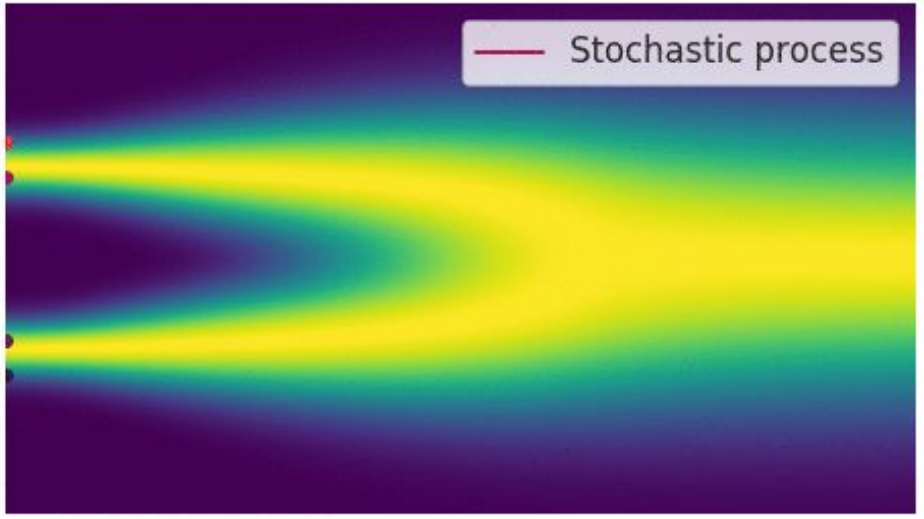
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

$\mu_\theta(x_t, t)$

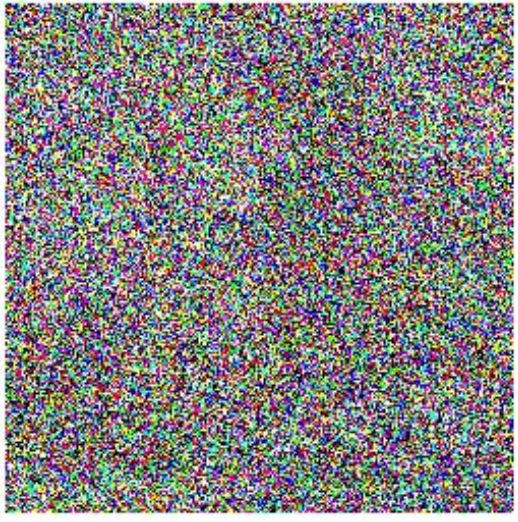
# Diffusion for Image Generation

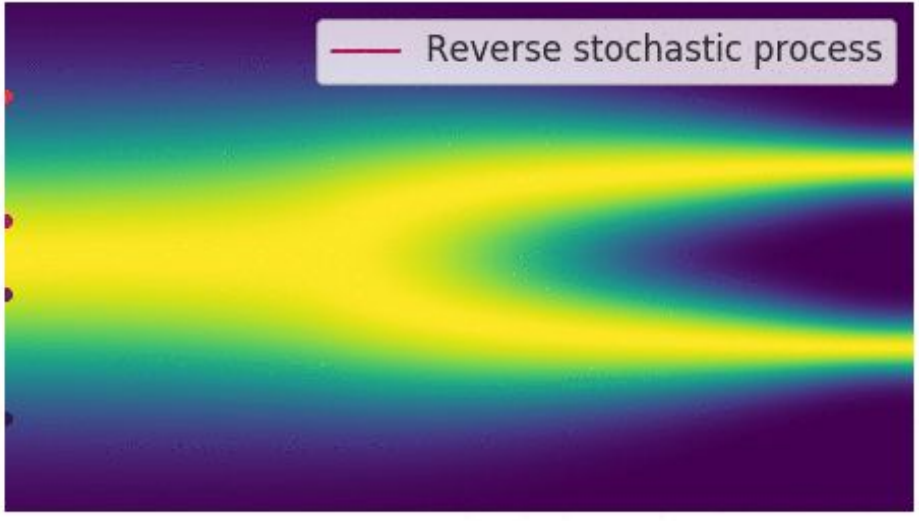
MACHINE LEARNING
@ BERKELEY

$x_0$ $x_T$




Forward process: converting the image distribution to pure noise





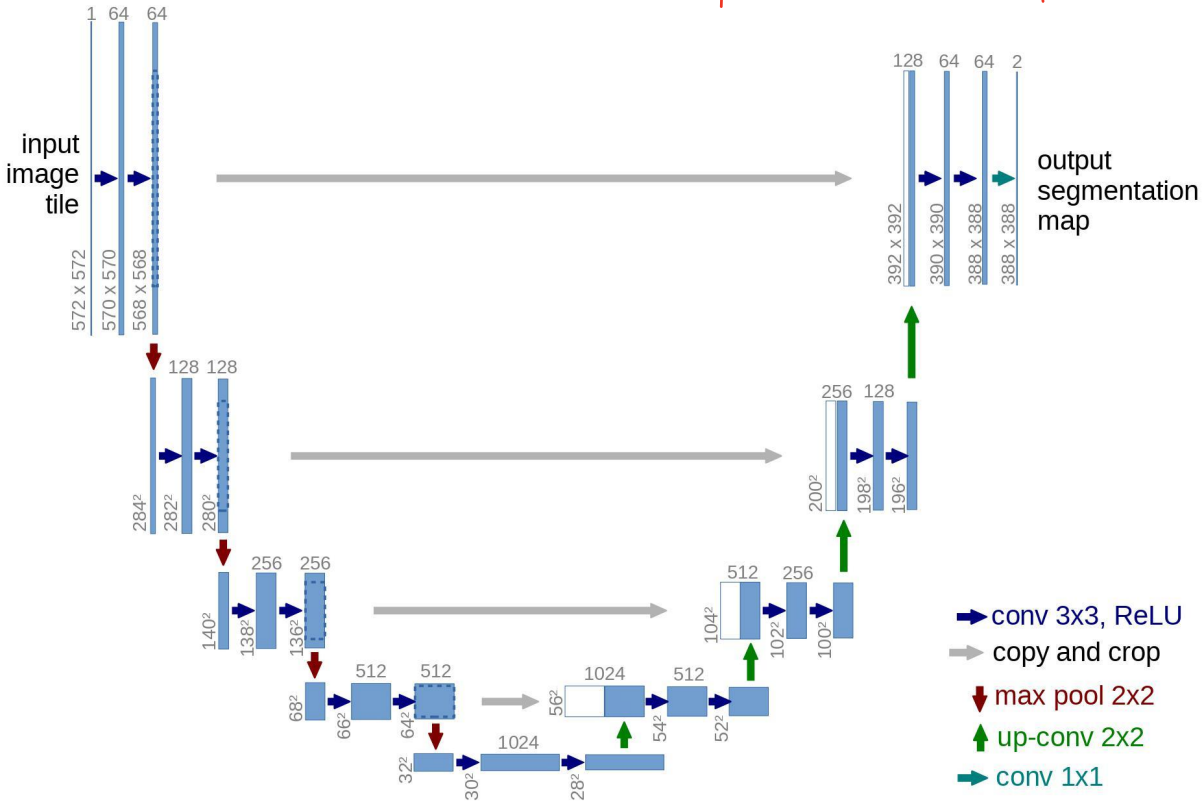

Reverse process: sampling from the image distribution, starting with pure noise

MACHINE LEARNING
@ BERKELEY

the noise must have the same dimensions as the image
⇒ input and the output have the same dimensions



# UNet + Other Stuff




Diffusion models typically use a U-Net on steroids as the noise predictive model — you take the good ol' model that you are already familiar with and add:

- Positional Embeddings
- ResNet Blocks
- ConvNext Blocks
- Attention Modules
- Group Normalization
- Swish and GeLU

It's a massive kitchen sink of modern CV tricks

# Tricks for Improving Generation

MACHINE LEARNING
@ BERKELEY

MACHINE LEARNING
@ BERKELEY

# Linear vs Cosine Schedule

$\beta_t$

- A linear noise schedule converts initial data to noise really quickly, making the reverse process harder for the model to learn.
- Researchers hypothesized that a cosine-like function that is changing relatively gradually near the endpoints might work better
    - Note: It did end up working better but this choice of cosine was completely arbitrary

Linear (top) vs Cosine (bottom)

# Learning a Covariance matrix

- DDPM authors said that it's better to use a fixed covariance matrix $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ where $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
  - The intuition is that covariance does not contribute as significantly as the mean does to the learned conditional distributions during the reverse process
  - However, it can still help us improve log-likelihood!
- So, Nichol and Dhariwal propose

predict with a neural network

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t)$$

This modification leads to better likelihood estimates while maintaining image quality!

if you have a higher likelihood this means that you are learning more of the initial data distribution and sample more and more images from it, likelihood can be view as a measure of diversity.

# Architecture Improvements

Nichol and Dhariwal proposed several architectural changes that seem to help diffusion training:

1. Increasing model depth vs width (not both): both help but increasing width is computationally cheaper while providing similar gains as increased depth
2. Increasing number of attention heads and applying it to multiple resolutions
3. Stealing BigGAN residual blocks for upsampling and downsampling
4. *Adaptive Group Normalization* — hopes to better incorporate timestep (and potentially class) information during the training/reverse process

# Classifier Guidance

MACHINE LEARNING @ BERKELEY

Recall conditional GANs from the previous lectures: they can be conditioned on class labels to synthesize specific kinds of images. We can apply the same idea to diffusion!

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)$$

The main idea is this:

- Take a pre-trained unconditional diffusion model
- During sampling, inject the gradients of a classifier model (that is trained from scratch on noisy images) into the unconditional reverse process
- Classifier guidance trades off image diversity for model fidelity, allowing it to push the performance of a diffusion model past that of a GAN

準確度

# Classifier Guidance

MACHINE LEARNING
@ BERKELEY

**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 Guidance

MACHINE LEARNING @ BERKELEY

At a high level:

- FID and sFID captures image quality
- Precision measures image fidelity ("resemblance to training images")
- Recall measures image diversity/distribution coverage

Lower FID/sFID is better

Higher Precision and Recall is better

| Model | FID | sFID | Prec | Rec |
|---|---|---|---|---|
| **LSUN Bedrooms 256×256** | | | | |
| DCTransformer† [42] | 6.40 | 6.66 | 0.44 | **0.56** |
| DDPM [25] | 4.89 | 9.07 | 0.60 | 0.45 |
| IDDPM [43] | 4.24 | 8.21 | 0.62 | 0.46 |
| StyleGAN [27] | 2.35 | 6.62 | 0.59 | 0.48 |
| **ADM (dropout)** | **1.90** | **5.59** | **0.66** | 0.51 |
| **LSUN Horses 256×256** | | | | |
| StyleGAN2 [28] | 3.84 | 6.46 | 0.63 | 0.48 |
| **ADM** | 2.95 | **5.94** | 0.69 | **0.55** |
| **ADM (dropout)** | **2.57** | 6.81 | **0.71** | **0.55** |
| **LSUN Cats 256×256** | | | | |
| DDPM [25] | 17.1 | 12.4 | 0.53 | 0.48 |
| StyleGAN2 [28] | 7.25 | **6.33** | 0.58 | 0.43 |
| **ADM (dropout)** | **5.57** | 6.69 | **0.63** | **0.52** |
| **ImageNet 64×64** | | | | |
| BigGAN-deep* [5] | 4.06 | 3.96 | **0.79** | 0.48 |
| IDDPM [43] | 2.92 | **3.79** | 0.74 | 0.62 |
| **ADM** | 2.61 | **3.77** | 0.73 | 0.63 |
| **ADM (dropout)** | **2.07** | 4.29 | 0.74 | **0.63** |

| Model | FID | sFID | Prec | Rec |
|---|---|---|---|---|
| **ImageNet 128×128** | | | | |
| BigGAN-deep [5] | 6.02 | 7.18 | **0.86** | 0.35 |
| LOGAN† [68] | 3.36 | | | |
| **ADM** | 5.91 | **5.09** | 0.70 | **0.65** |
| **ADM-G (25 steps)** | 5.98 | 7.04 | 0.78 | 0.51 |
| **ADM-G** | **2.97** | **5.09** | 0.78 | 0.59 |
| **ImageNet 256×256** | | | | |
| DCTransformer† [42] | 36.51 | 8.24 | 0.36 | **0.67** |
| VQ-VAE-2†‡ [51] | 31.11 | 17.38 | 0.36 | 0.57 |
| IDDPM‡ [43] | 12.26 | 5.42 | 0.70 | 0.62 |
| SR3†‡ [53] | 11.30 | | | |
| BigGAN-deep [5] | 6.95 | 7.36 | **0.87** | 0.28 |
| **ADM** | 10.94 | 6.02 | 0.69 | 0.63 |
| **ADM-G (25 steps)** | 5.44 | 5.32 | 0.81 | 0.49 |
| **ADM-G** | **4.59** | **5.25** | 0.82 | 0.52 |
| **ImageNet 512×512** | | | | |
| BigGAN-deep [5] | 8.43 | 8.13 | **0.88** | 0.29 |
| **ADM** | 23.24 | 10.19 | 0.73 | **0.60** |
| **ADM-G (25 steps)** | 8.41 | 9.67 | 0.83 | 0.47 |
| **ADM-G** | **7.72** | **6.57** | 0.87 | 0.42 |

MACHINE LEARNING
@ BERKELEY

# Diffusion Models Beats GANs

look similar in these images

BigGAN

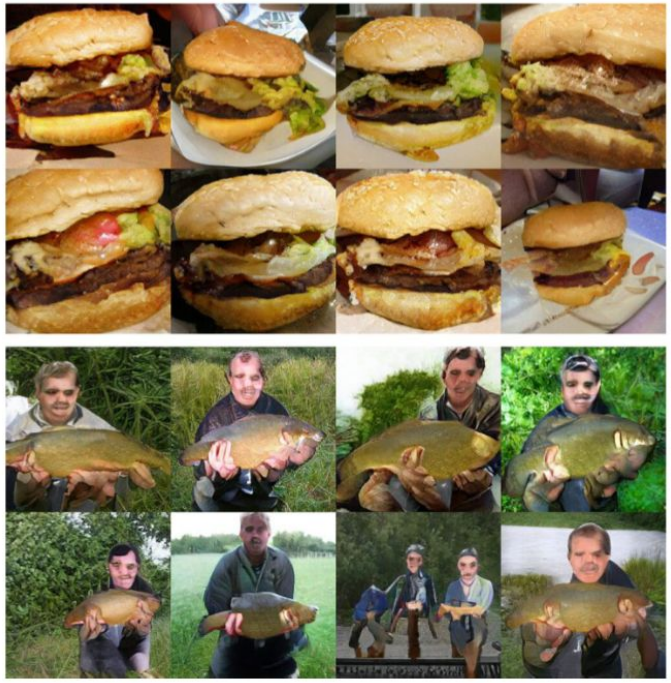
Diffusion

Training Set

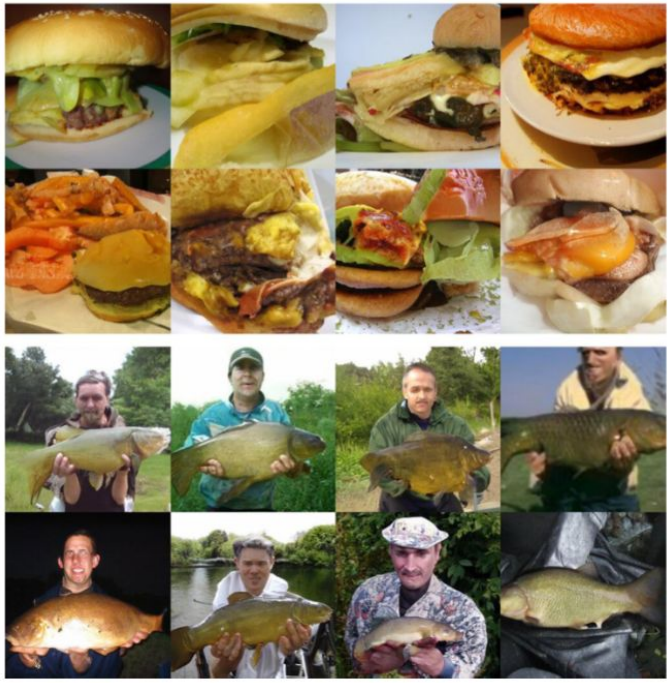
# Diffusion Models Beats GANs

MACHINE LEARNING
@ BERKELEY

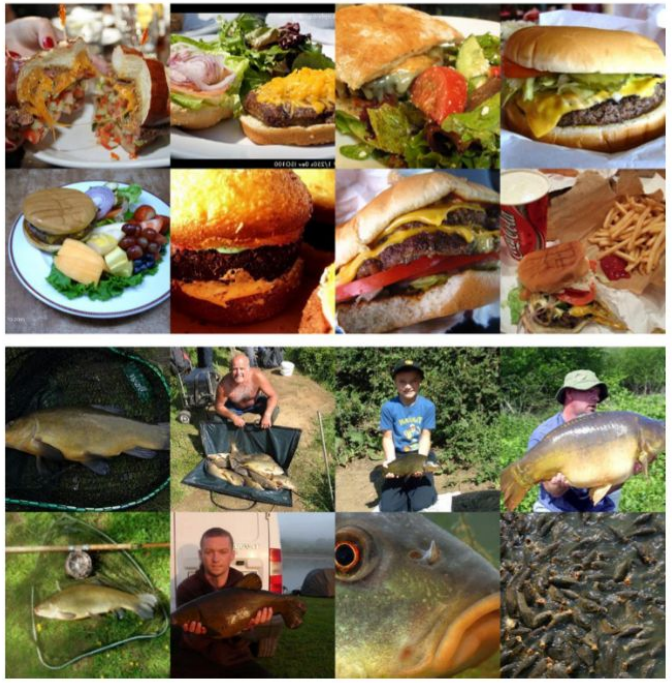
BigGAN



Diffusion



Training Set



# Classifier-Free Guidance

MACHINE LEARNING
@ BERKELEY

Classifier guidance worked great but

- Can't use a pre-trained classifier since classifier must be trained on noisy data
- Since classifier guidance injects classifier gradients into it's training process, is it really not just an adversarial attack that the classifier has learned to become robust against? It is hard to interpret what classifier guidance is actually doing
- To avoid these dilemmas, researchers proposed ignoring an external classifier all together and jointly training a class-conditioned and unconditional diffusion model simultaneously
- The goal of this paper was to understand the behavior of guidance, not to push the boundaries of image synthesis, but…

# Classifier-Free Guidance

| Model | FID (↓) | IS (↑) |
|---|---|---|
| BigGAN-deep, max IS (Brock et al., 2019) | 25 | 253 |
| BigGAN-deep (Brock et al., 2019) | 5.7 | 124.5 |
| CDM (Ho et al., 2021) | 3.52 | 128.8 |
| LOGAN (Wu et al., 2019) | 3.36 | 148.2 |
| ADM-G (Dhariwal & Nichol, 2021) | 2.97 | - |
| Ours | $T = 128/256/1024$ | |
| $w = 0.0$ | 8.11 / 7.27 / 7.22 | 81.46 / 82.45 / 81.54 |
| $w = 0.1$ | 5.31 / 4.53 / 4.5 | 105.01 / 106.12 / 104.67 |
| $w = 0.2$ | 3.7 / 3.03 / 3 | 130.79 / 132.54 / 130.09 |
| $w = 0.3$ | 3.04 / **2.43** / **2.43** | 156.09 / 158.47 / 156 |
| $w = 0.4$ | 3.02 / 2.49 / 2.48 | 183.01 / 183.41 / 180.88 |
| $w = 0.5$ | 3.43 / 2.98 / 2.96 | 206.94 / 207.98 / 204.31 |
| $w = 0.6$ | 4.09 / 3.76 / 3.73 | 227.72 / 228.83 / 226.76 |
| $w = 0.7$ | 4.96 / 4.67 / 4.69 | 247.92 / 249.25 / 247.89 |
| $w = 0.8$ | 5.93 / 5.74 / 5.71 | 265.54 / 267.99 / 265.52 |
| $w = 0.9$ | 6.89 / 6.8 / 6.81 | 280.19 / 283.41 / 281.14 |
| $w = 1.0$ | 7.88 / 7.86 / 7.8 | 295.29 / 297.98 / 294.56 |
| $w = 2.0$ | 15.9 / 15.93 / 15.75 | 378.56 / 377.37 / 373.18 |
| $w = 3.0$ | 19.77 / 19.77 / 19.56 | 409.16 / 407.44 / 405.68 |
| $w = 4.0$ | 21.55 / 21.53 / 21.45 | **422.29** / 421.03 / 419.06 |

# Classifier vs Classifier-Free Guidance

1. Classifier-free guidance is pretty simple to implement while classifier guidance requires training an external classifier on noisy data
2. Classifier-free guidance does not employ any kind of classifier gradients and cannot be interpreted as an adversarial attack: it is more in line with traditional generative models
3. Classifier-free guidance is slower than classifier guidance by the virtue of requiring twice as many reverse diffusion steps
4. Both lower the sample diversity to increase sample fidelity/quality: is this really acceptable?

MACHINE LEARNING
@ BERKELEY

Diversity
(unguided samples)
vs.
Fidelity
(guided samples)

# Latent Diffusion Models

MACHINE LEARNING
@ BERKELEY

# Latent Diffusion Models Motivation

- Training models in the pixel space is excessively computationally expensive (can easily multiple days on a V100 GPU)
  - Even image synthesis is very slow compared to GANs
  - Images are high dimensional → more things to model
- Researchers observed that most "bits" of an image contribute to its perceptual characteristics since aggressively compressing it usually maintains its semantic and conceptual composition
  - In layman's terms, there are more bits for describing pixel-level details while less bits for describing "the meaning" within an image
  - Generative models should learn the latter
- Can we separate these two components?

# Latent Diffusion Models

Latent Diffusion Models can be divided into two stages:

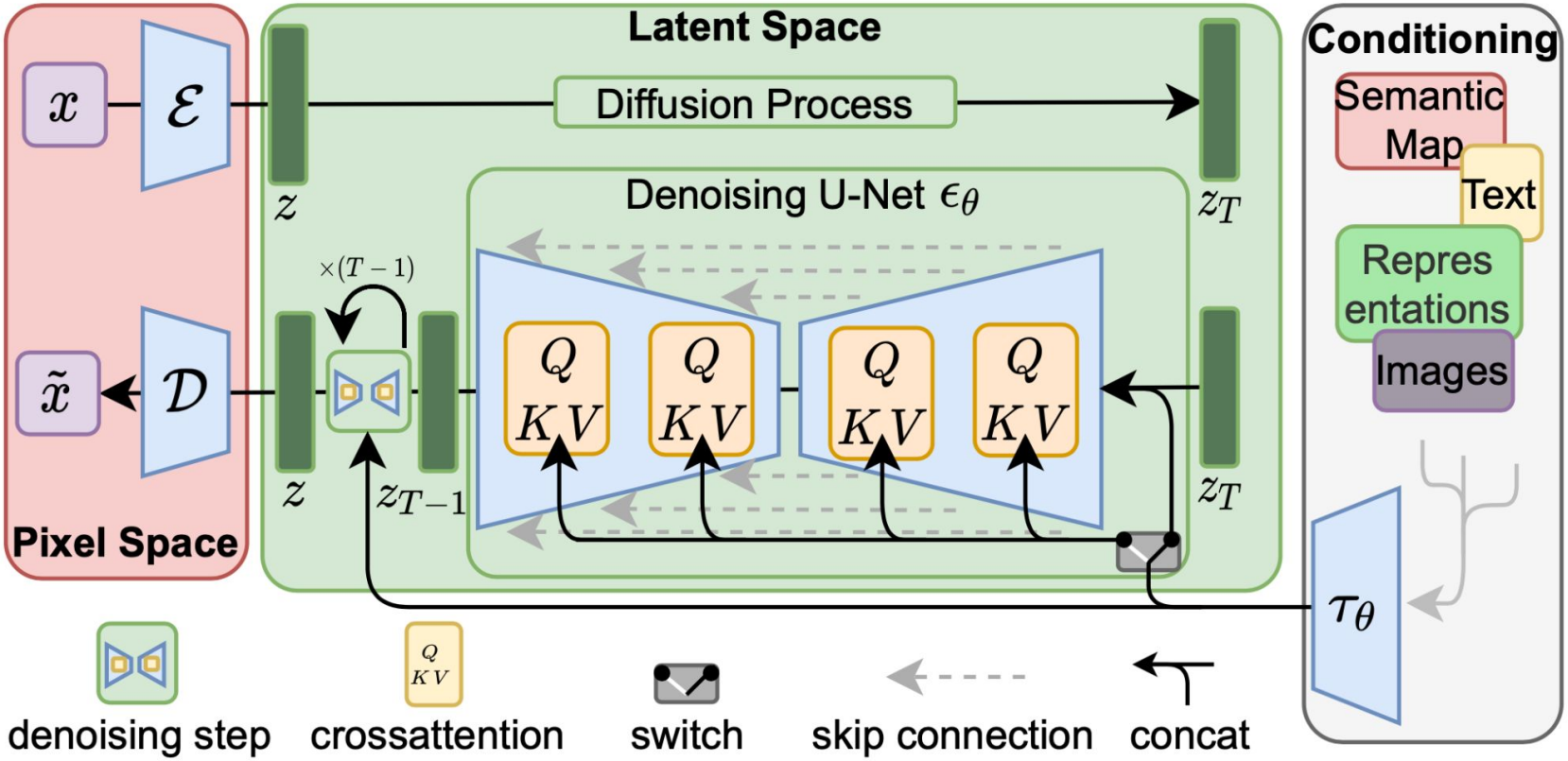
1. Training perceptual compression models that strip away irrelevant high-level details and learn a latent space that is semantically equivalent to the high level image pixel-space
    a. The loss is a combination of a reconstruction loss, an adversarial loss (remember GANs?) that promotes high quality decoder reconstruction, and regularization terms

$$L_{\text{Autoencoder}} = \min_{\mathcal{E},\mathcal{D}} \max_{\psi} \Big( L_{rec}(x, \mathcal{D}(\mathcal{E}(x))) - L_{adv}(\mathcal{D}(\mathcal{E}(x))) + \log D_{\psi}(x) + L_{reg}(x; \mathcal{E}, \mathcal{D}) \Big)$$

2. Performing a diffusion process *in this latent space*. There are several benefits to this:
    a. The diffusion process is only focusing on the relevant semantic bits of the data
    b. Performing diffusion in a low dimensional space is significantly more efficient

MACHINE LEARNING
@ BERKELEY

# U-Net



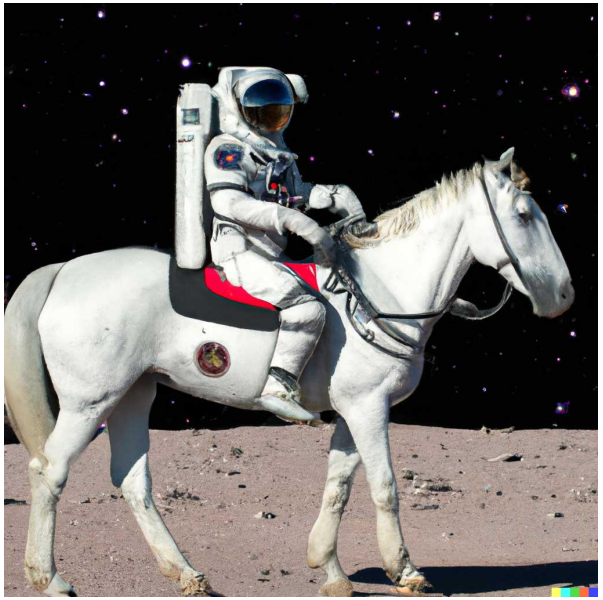
# Examples of Recent Diffusion Models

MACHINE LEARNING
@ BERKELEY

MACHINE LEARNING
@ BERKELEY

# DALLE 2 (Text-to-Image)

Teddy bears mixing sparkling chemicals as mad scientists



An astronaut riding a horse in a photorealistic style

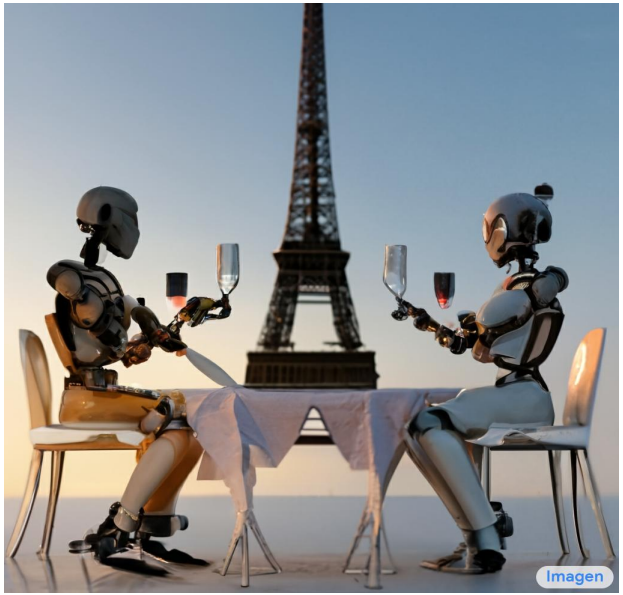
A bowl of soup as a planet in the universe

MACHINE LEARNING
@ BERKELEY

# Imagen (Text-to-Image)


A cute corgi lives in a house made of sushi


A majestic oil painting of a raccoon Queen wearing red French royal gown.




A robot couple fine-dining with the Eiffel Tower in the background

MACHINE LEARNING
@ BERKELEY

# Video Diffusion (Text-to-Video)

MACHINE LEARNING
@ BERKELEY

# Make-A-Video (Text-to-Video)


An artist's brush painting on a canvas close up


A young couple walking in heavy rain


Horse drinking water

MACHINE LEARNING
@ BERKELEY

# Make-A-Video (Text-to-Video)


A confused grizzly bear in a calculus class


A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

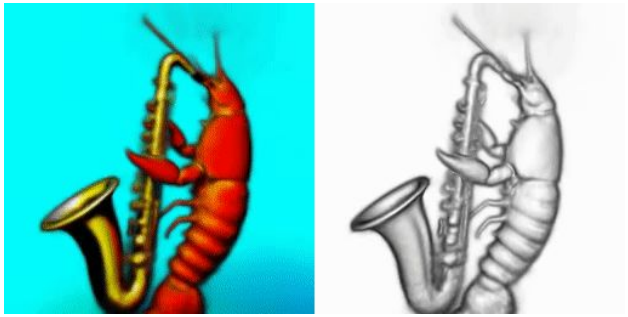

A panda playing on a swing set

MACHINE LEARNING
@ BERKELEY

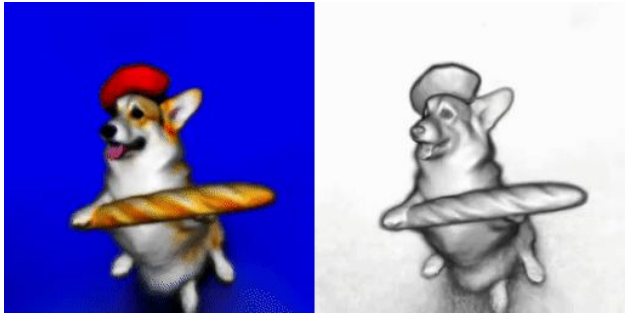
# Imagen Video (Text-to-Video)

# DreamFusion (Text-to-3D)

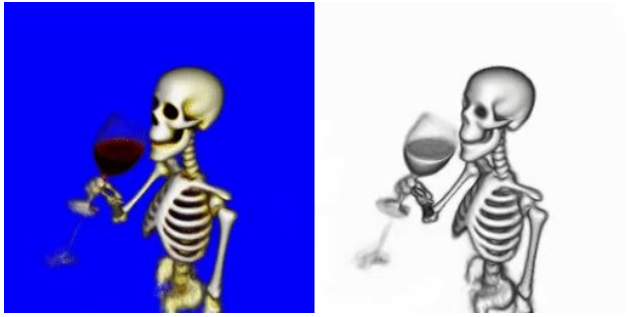
a fox holding a video game controller



a lobster playing the saxophone



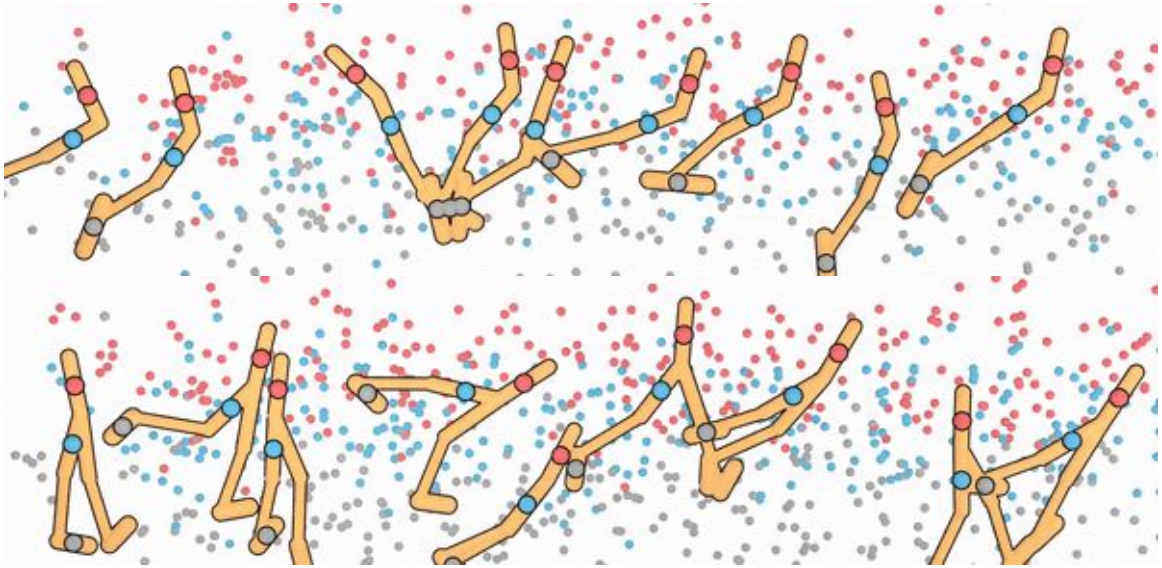
a corgi wearing a beret and holding a baguette, standing up on two hind legs

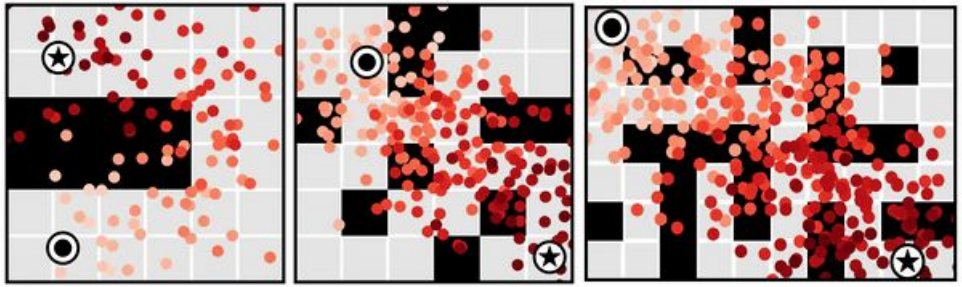


a human skeleton drinking a glass of red wine

MACHINE LEARNING
@ BERKELEY

# Diffuser (Trajectory Planning)





MACHINE LEARNING
@ BERKELEY

# Diffusion-QL (Offline RL)

Even applies to RL!

- Recall that diffusion is basically a method for learning unknown distributions.
- In offline RL, the goal is to train an agent from offline datasets
- To ensure

| Gym Tasks | BC | AWAC | Diffuser | MoRel | Onestep RL | TD3+BC | DT | CQL | IQL | Diffusion-QL |
|---|---|---|---|---|---|---|---|---|---|---|
| halfcheetah-medium-v2 | 42.6 | 43.5 | 44.2 | 42.1 | 48.4 | 48.3 | 42.6 | 44.0 | 47.4 | **51.1** ± 0.5 |
| hopper-medium-v2 | 52.9 | 57.0 | 58.5 | **95.4** | 59.6 | 59.3 | 67.6 | 58.5 | 66.3 | 90.5 ± 4.6 |
| walker2d-medium-v2 | 75.3 | 72.4 | 79.7 | 77.8 | 81.8 | 83.7 | 74.0 | 72.5 | 78.3 | **87.0** ± 0.9 |
| halfcheetah-medium-replay-v2 | 36.6 | 40.5 | 42.2 | 40.2 | 38.1 | 44.6 | 36.6 | 45.5 | 44.2 | **47.8** ± 0.3 |
| hopper-medium-replay-v2 | 18.1 | 37.2 | 96.8 | 93.6 | 97.5 | 60.9 | 82.7 | 95.0 | 94.7 | **101.3** ± 0.6 |
| walker2d-medium-replay-v2 | 26.0 | 27.0 | 61.2 | 49.8 | 49.5 | 81.8 | 66.6 | 77.2 | 73.9 | **95.5** ± 1.5 |
| halfcheetah-medium-expert-v2 | 55.2 | 42.8 | 79.8 | 53.3 | 93.4 | 90.7 | 86.8 | 91.6 | 86.7 | **96.8** ± 0.3 |
| hopper-medium-expert-v2 | 52.5 | 55.8 | 107.2 | 108.7 | 103.3 | 98.0 | 107.6 | 105.4 | 91.5 | **111.1** ± 1.3 |
| walker2d-medium-expert-v2 | 107.5 | 74.5 | 108.4 | 95.6 | **113.0** | 110.1 | 108.1 | 108.8 | 109.6 | 110.1 ± 0.3 |
| **Average** | 51.9 | 50.1 | 75.3 | 72.9 | 76.1 | 75.3 | 74.7 | 77.6 | 77.0 | **88.0** |

| AntMaze Tasks | BC | AWAC | BCQ | BEAR | Onestep RL | TD3+BC | DT | CQL | IQL | Diffusion-QL |
|---|---|---|---|---|---|---|---|---|---|---|
| antmaze-umaze-v0 | 54.6 | 56.7 | 78.9 | 73.0 | 64.3 | 78.6 | 59.2 | 74.0 | 87.5 | **93.4** ± 3.4 |
| antmaze-umaze-diverse-v0 | 45.6 | 49.3 | 55.0 | 61.0 | 60.7 | 71.4 | 53.0 | **84.0** | 62.2 | 66.2 ± 8.6 |
| antmaze-medium-play-v0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.3 | 10.6 | 0.0 | 61.2 | 71.2 | **76.6** ± 10.8 |
| antmaze-medium-diverse-v0 | 0.0 | 0.7 | 0.0 | 8.0 | 0.0 | 3.0 | 0.0 | 53.7 | 70.0 | **78.6** ± 10.3 |
| antmaze-large-play-v0 | 0.0 | 0.0 | 6.7 | 0.0 | 0.0 | 0.2 | 0.0 | 15.8 | 39.6 | **46.4** ± 8.3 |
| antmaze-large-diverse-v0 | 0.0 | 1.0 | 2.2 | 0.0 | 0.0 | 0.0 | 0.0 | 14.9 | 47.5 | **56.6** ± 7.6 |
| **Average** | 16.7 | 18.0 | 23.8 | 23.7 | 20.9 | 27.3 | 18.7 | 50.6 | 63.0 | **69.6** |

| Adroit Tasks | BC | SAC | BCQ | BEAR | BRAC-p | BRAC-v | REM | CQL | IQL | Diffusion-QL |
|---|---|---|---|---|---|---|---|---|---|---|
| pen-human-v1 | 25.8 | 4.3 | 68.9 | -1.0 | 8.1 | 0.6 | 5.4 | 35.2 | 71.5 | **72.8** ± 9.6 |
| pen-cloned-v1 | 38.3 | -0.8 | 44.0 | 26.5 | 1.6 | -2.5 | -1.0 | 27.2 | 37.3 | **57.3** ± 11.9 |
| **Average** | 32.1 | 1.8 | 56.5 | 12.8 | 4.9 | -1.0 | 2.2 | 31.2 | 54.4 | **65.1** |

| Kitchen Tasks | BC | SAC | BCQ | BEAR | BRAC-p | BRAC-v | AWR | CQL | IQL | Diffusion-QL |
|---|---|---|---|---|---|---|---|---|---|---|
| kitchen-complete-v0 | 33.8 | 15.0 | 8.1 | 0.0 | 0.0 | 0.0 | 0.0 | 43.8 | 62.5 | **84.0** ± 7.4 |
| kitchen-partial-v0 | 33.8 | 0.0 | 18.9 | 13.1 | 0.0 | 0.0 | 15.4 | 49.8 | 46.3 | **60.5** ± 6.9 |
| kitchen-mixed-v0 | 47.5 | 2.5 | 8.1 | 47.2 | 0.0 | 0.0 | 10.6 | 51.0 | 51.0 | **62.6** ± 5.1 |
| **Average** | 38.4 | 5.8 | 11.7 | 20.1 | 0.0 | 0.0 | 8.7 | 48.2 | 53.3 | **69.0** |

# Wrap-Up

MACHINE LEARNING
@ BERKELEY


# Summary

We went over

- A quick tour of generative modeling and how image synthesis can be viewed as sampling from a density
- Preliminary theory of diffusion (don't worry if this is confusing; this is a very theory rich subject and even I don't know all the details!)
- Some tricks that modern diffusion models employ for image generation:
  - A U-Net architecture equipped with all kinds of modifications
  - Other architecture improvements
  - Several implementation tricks (different noise schedules, covariance parametrizations)
  - Classifier and classifier-free guidance
- Latent diffusion models for improving diffusion quality and efficiency

# Main papers referenced here!

Disclaimer: some of the foundational work done on Diffusion is relatively math and notation heavy!

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics: https://arxiv.org/pdf/1503.03585.pdf
- Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2006.11239.pdf
- Improved Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2102.09672.pdf
- Diffusion Models Beat GANs on Image Synthesis: https://arxiv.org/pdf/2105.05233.pdf
- Classifier-free Diffusion Guidance: https://arxiv.org/pdf/2207.12598.pdf
- High Resolution Image Synthesis with Latent Diffusion Models: https://arxiv.org/pdf/2112.10752.pdf

MACHINE LEARNING
@ BERKELEY

# Other cool papers to check out!

Disclaimer: some of the foundational work done on Diffusion is relatively math and notation heavy!

- Denoising Diffusion Implicit Models: https://arxiv.org/pdf/1503.03585.pdf
- Generative Modeling by Estimating Gradients of the Data Distribution: https://yang-song.net/blog/2021/score/
- Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions: https://arxiv.org/pdf/2209.11215.pdf

# Even more resources!

Other resources:

- Lillian Weng's Blog: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- The Annotated Diffusion Model: https://huggingface.co/blog/annotated-diffusion
- The Illustrated Stable Diffusion: https://jalammar.github.io/illustrated-stable-diffusion/
- PyTorch implementation of the DDPM Unet: https://nn.labml.ai/diffusion/ddpm/unet.html
- Guidance: a cheat code for diffusion models: https://benanne.github.io/2022/05/26/guidance.html
- Understanding Diffusion Models: A Unified Perspective: https://arxiv.org/pdf/2208.11970.pdf