GANs and Diffusion Models

Yuchen Liang

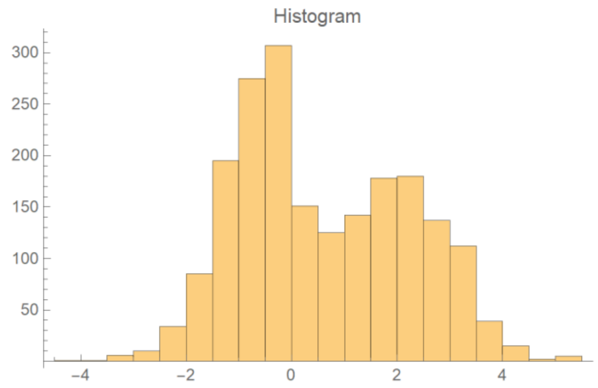
REU Summer 2025

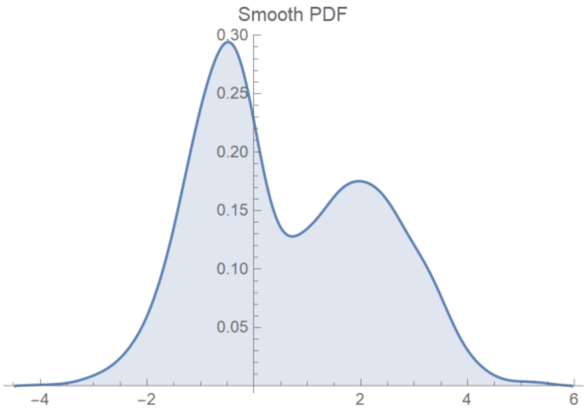
The Ohio State University
College of Engineering

Main Tasks for Generative Models

- **Key: Sample generation**

- Density estimation
- Representation learning
- Etc.



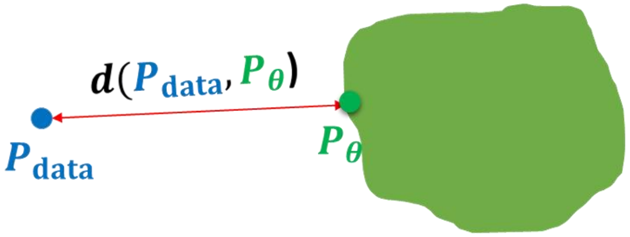


E.g., Kernel Density Estimator (KDE) can estimate density, but cannot sample…

Key Challenges

1. **Representation**: How do we model the **marginal/joint distribution** of many random variables?
2. **Learning**: What is the right way to **compare probability distributions**?

$x_i \sim P_{\text{data}}$
$i = 1, 2, \dots, n$



$\theta \in M$

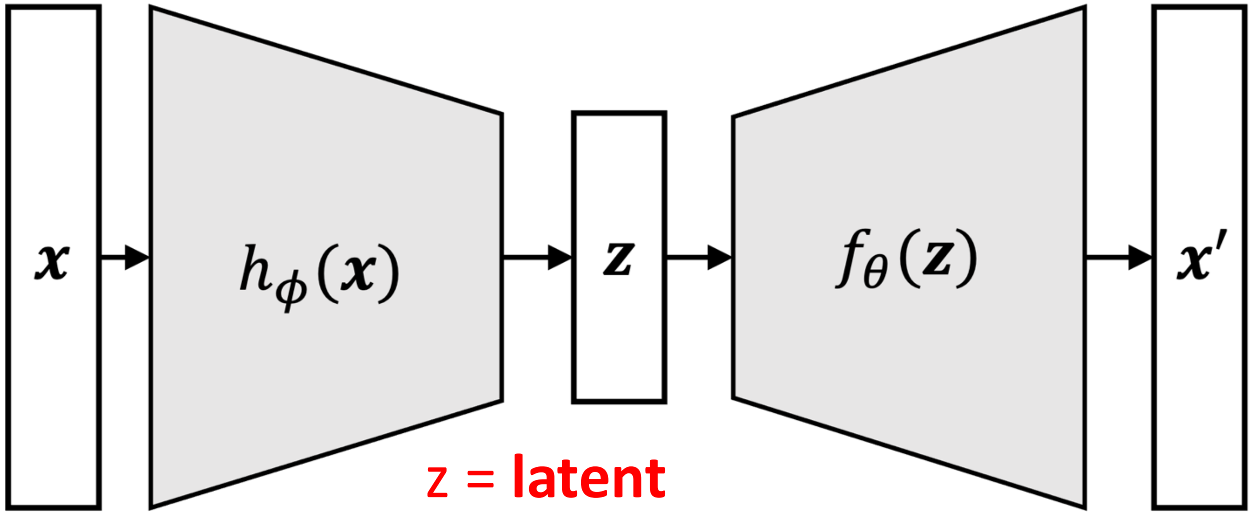
Model family

- **Other challenges:** How to obtain a sample? How to evaluate the performance?

Recap on VAE

- Distribution Representation: feed-forward neural network (FFNN)
- Structure: **Encoder** network h_ϕ + **Decoder** network f_θ

Say, an image of 28x28=784 dimensions



z = **latent** (compressed) data, say 50 dimensions

Recap on VAE Training

- Goal: maximize (log-)**likelihood**:

$$\log p_\theta(x_1, \dots, x_N) = \sum_i \log p_\theta(x_i)$$

- Practical algorithm: maximize the **ELBO**

$$\log p_\theta(x) \geq ELBO = \int \left(\log p_\theta(x, z) - \log q_\phi(z|x)\right) q_\phi(z|x)\, d\, z$$

- Disadvantages of ELBO:
  1. Inconsistent estimator (introducing asymptotic bias)
  2. Samples tend to have lower quality…

Rethinking the objective…

- Goal: maximize (log-)**likelihood**:

$$\max_{\theta} \mathbb{E}_{X\sim p_{data}}[\log p_{\theta}(X)]$$

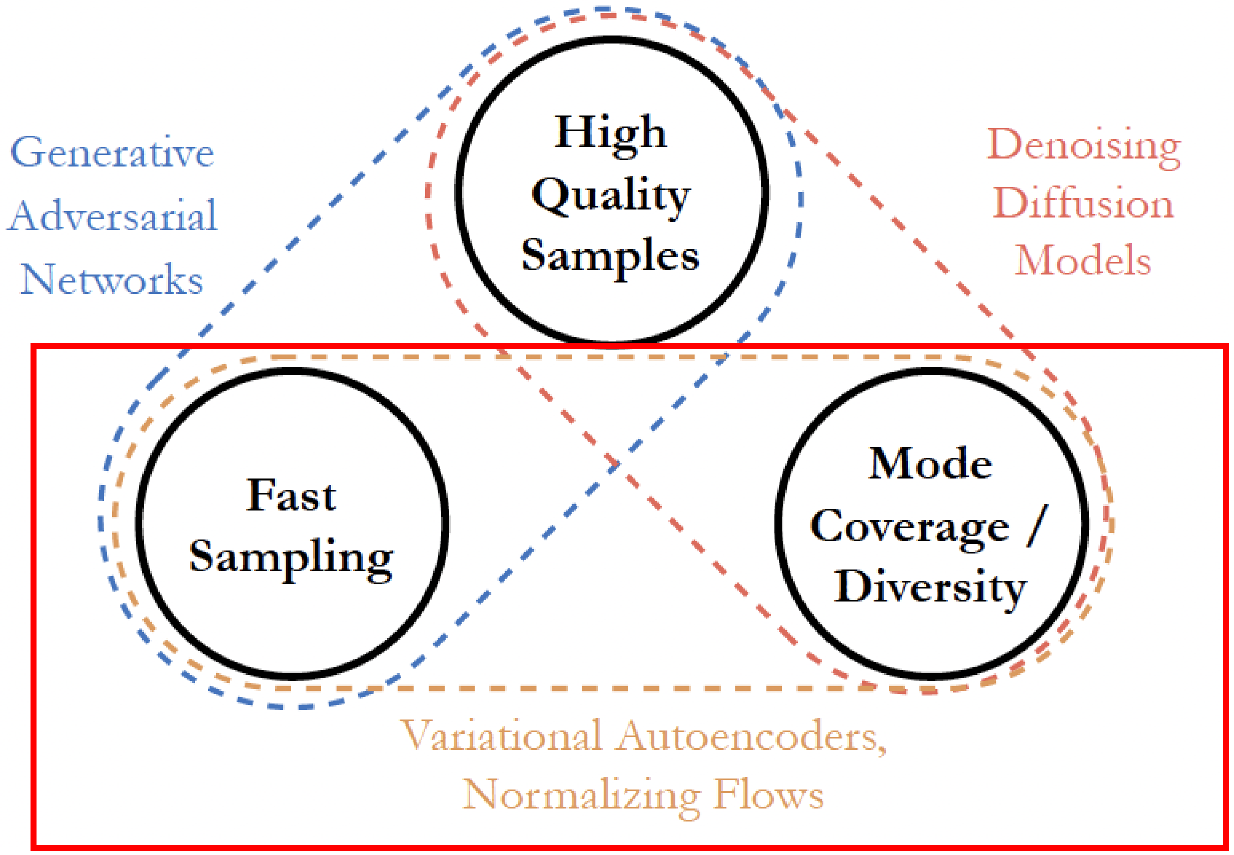
- Fact: **Optimal** generative model gives the **best** sample quality and **highest** likelihood
- Caveat: For **imperfect** models, higher likelihood **≠** better sample!
- Example: memorizing training data -> great samples, zero likelihood on test dataset

Refer to Theis et al., 2016

Agenda

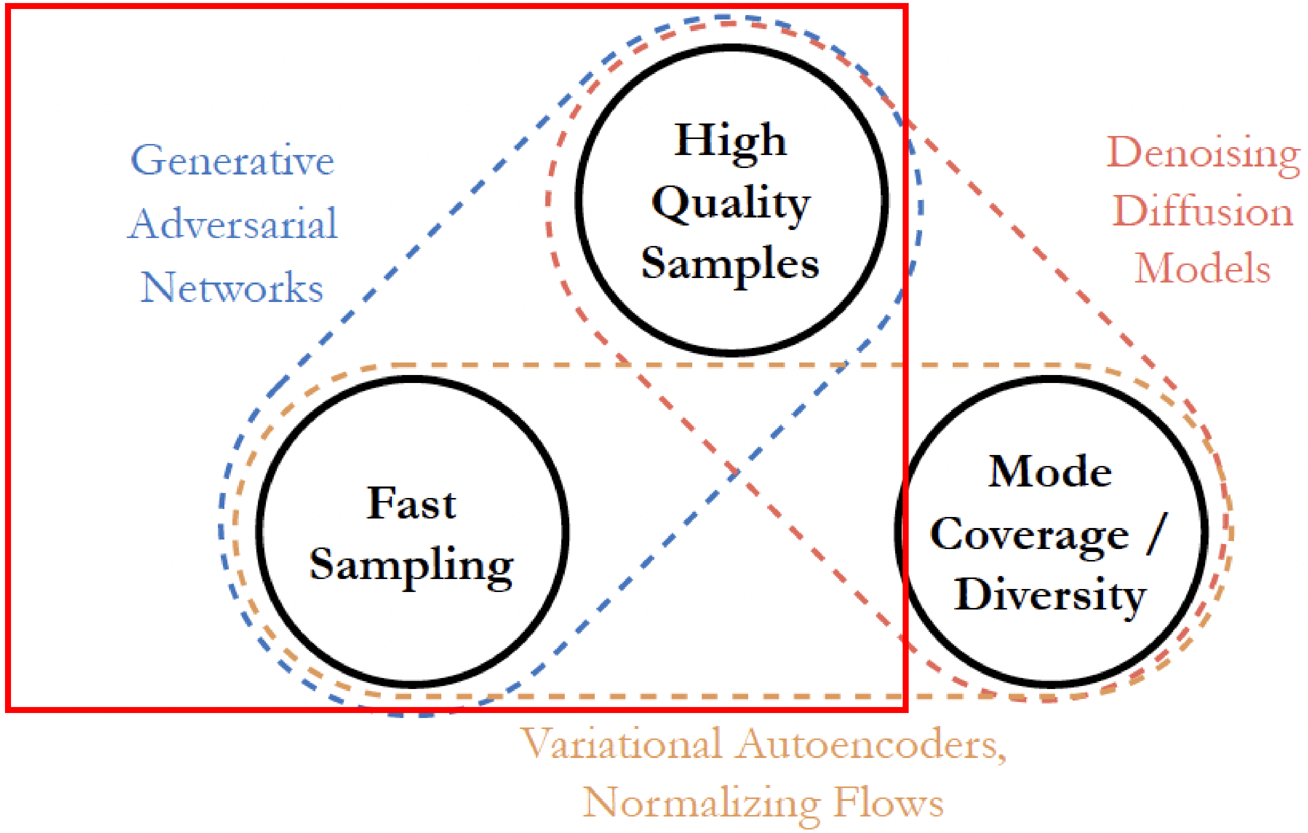
1. **Generative Adversarial Networks (GANs)**
2. Diffusion models

Trichotomy of Existing Generative Models



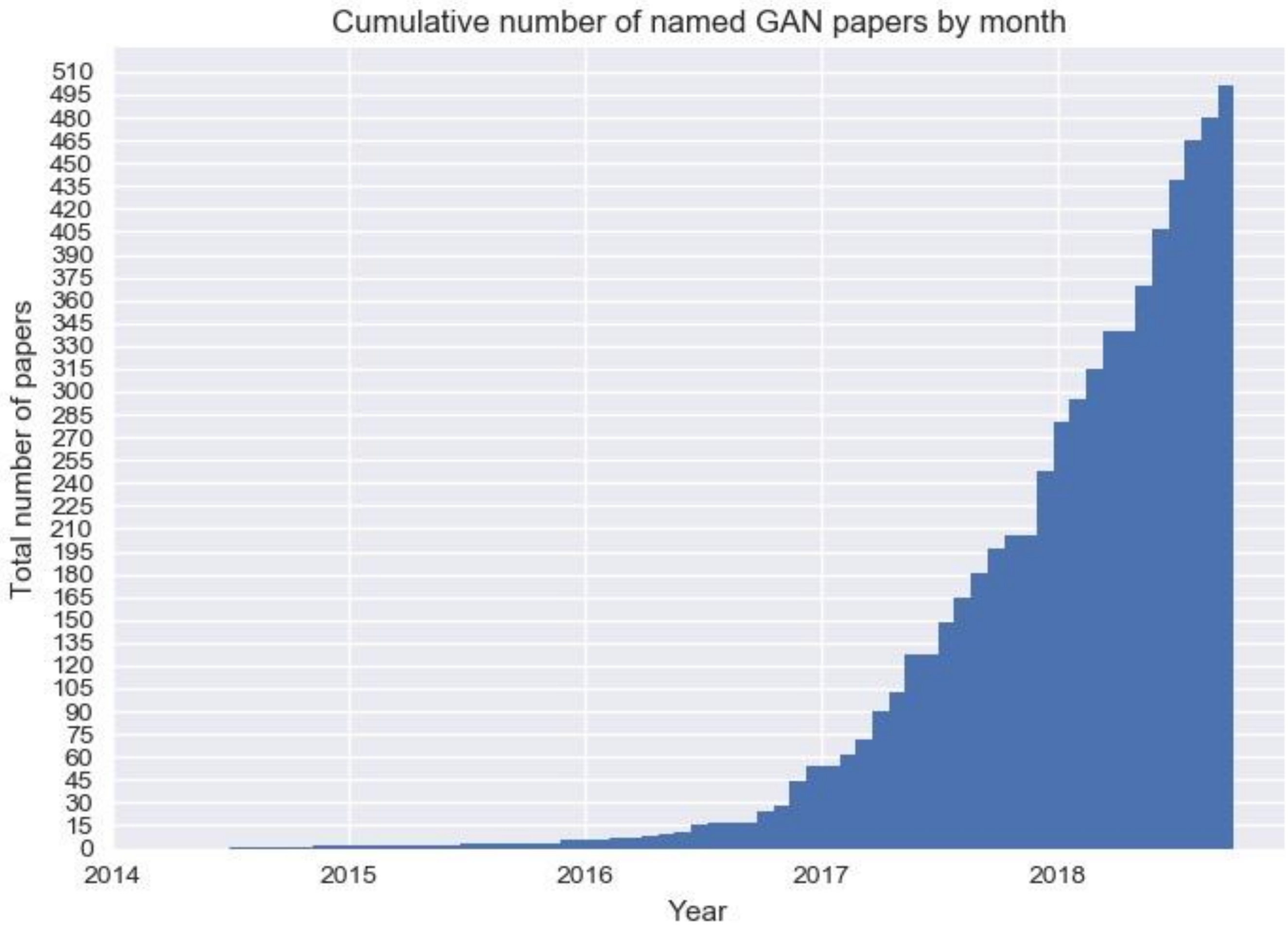
Xiao, 2022

Trichotomy of Existing Generative Models



Xiao, 2022

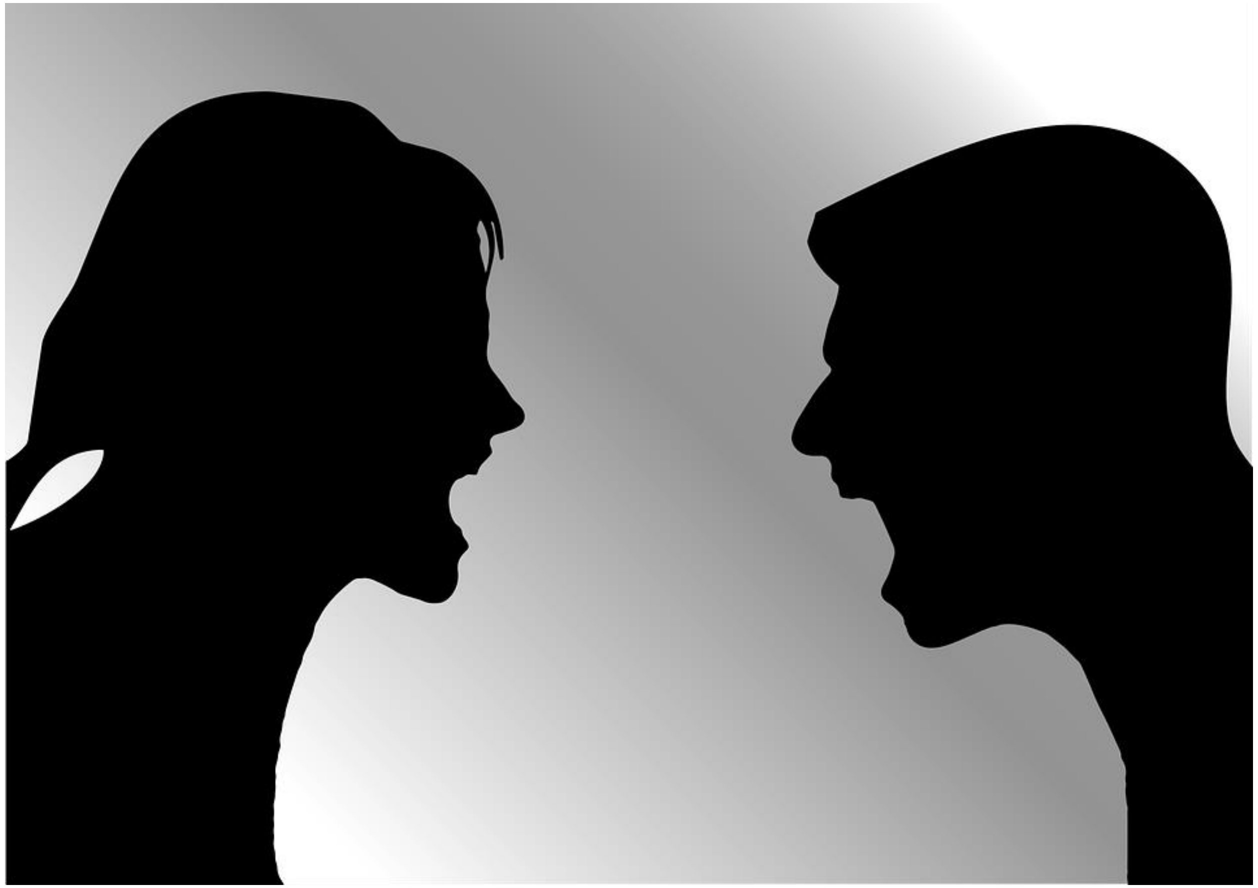
The GAN Zoo…



https://github.com/hindupuravinash/the-gan-zoo

Sample Video from StyleGAN3

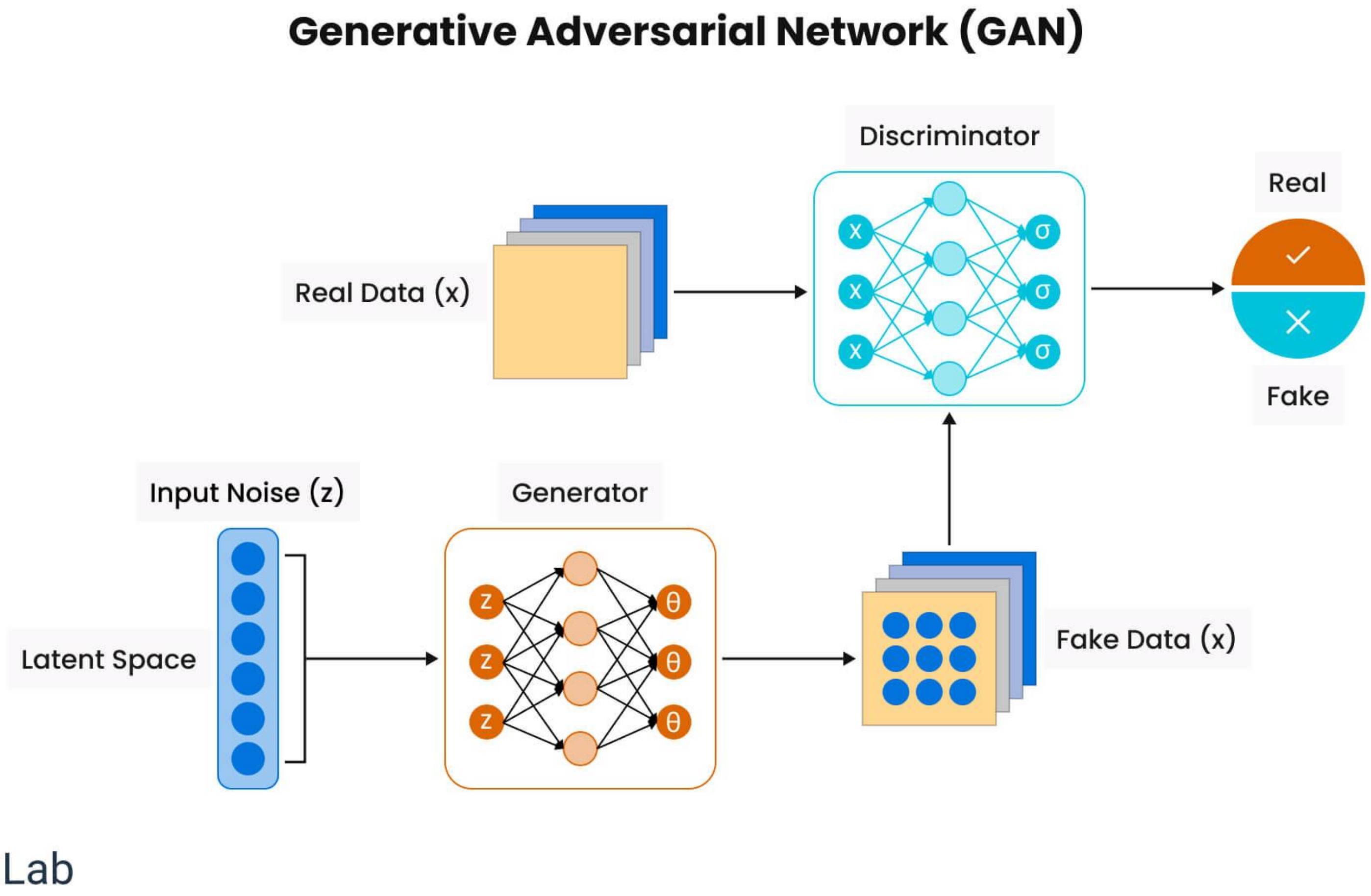
Intuition of GAN: Adversarial



GAN: Basic Structure

- GAN is the only **likelihood-free** generative model!
- **Discriminator**: Given an input image x, output the prob that it is real
  - If x is real, $\mathrm{D}(x) \approx 1$.
  - If x is artificial, $\mathrm{D}(x) \approx 0$.
- **Generator**: Generate some $x = G(z)$ such that $\mathrm{D}\big(G(z)\big) \approx 1$.
  - z is the initial latent noise (e.g., unit Gaussian)

GAN: Basic Structure



GAN: Training Objective

- Loss for **Discriminator**: For **fixed** G,

$$\max_D \left\{ \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_z \left[\log \left(1 - D(G(z)) \right) \right] \right\}$$

Higher D values for real data

Lower D values for fake data

- Loss for **Generator**: For **fixed** D,

$$\min_G \left\{ \mathbb{E}_z \left[\log \left(1 - D(G(z)) \right) \right] \right\}$$

Can we solve GAN's optimization problem?

- Optimization problem:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{y\sim p_G}[\log(1 - D(y))]$$

- For fixed G, take derivative w.r.t. D:

$$0 = \frac{\delta}{\delta D} V(G, D) = \int p_{\text{data}}(x) \frac{1}{D(x)} dx + \int p_G(x) \frac{-1}{1 - D(x)} dx$$

(Rule of derivative of log)

$$\implies 0 = p_{\text{data}}(x) \frac{1}{D(x)} - p_G(x) \frac{1}{1 - D(x)}$$

(b/c everything is non-negative)

$$\implies D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

(Rearrange the terms)

Can we solve GAN's optimization problem? (cont.)

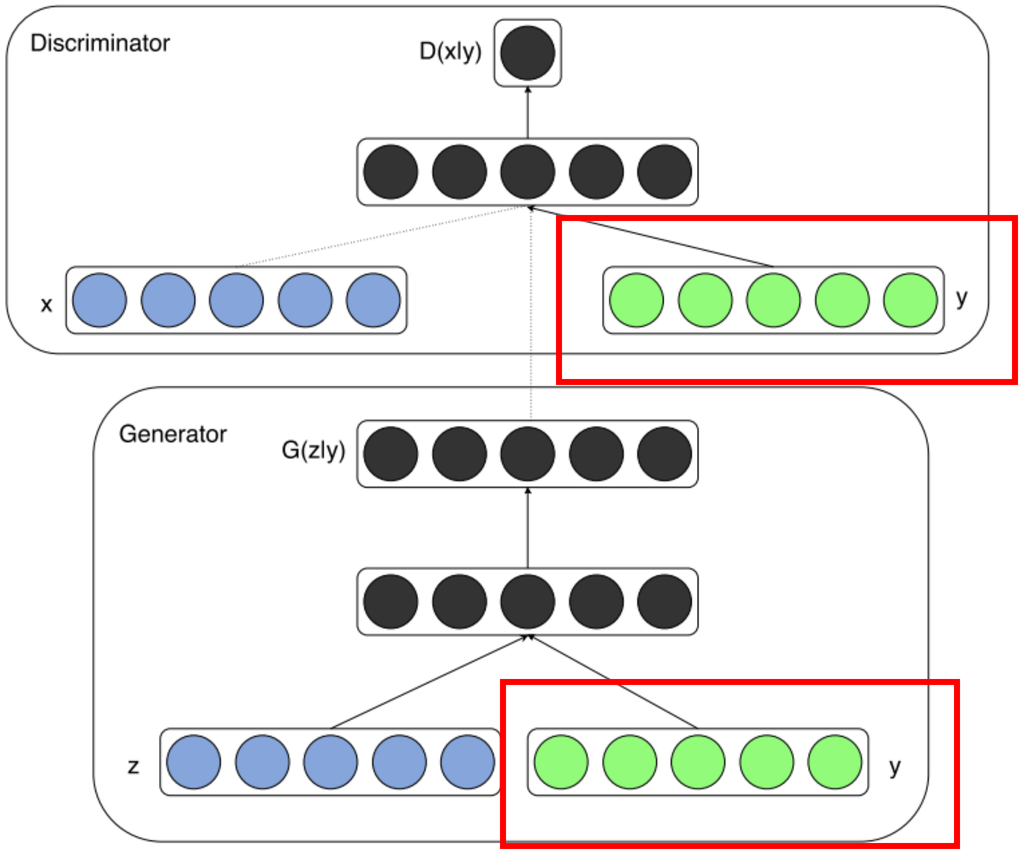
- The optimal training loss for fixed G:

$$V(G, D^*) = \int p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} dx + \int p_G(x) \log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)} dx = \text{JSD}(p_{\text{data}}(x), p_G(x))$$

  - JSD stands for **Jensen-Shannon Divergence**, measuring diff of dist's

- Optimal Generator: $p_G^* = p_{data}$, and $V^* = -\log 4$

Extension 1: Conditional GAN (CGAN)



https://arxiv.org/pdf/1411.1784

Extension 2: Deep Convolutional GAN (DCGAN)

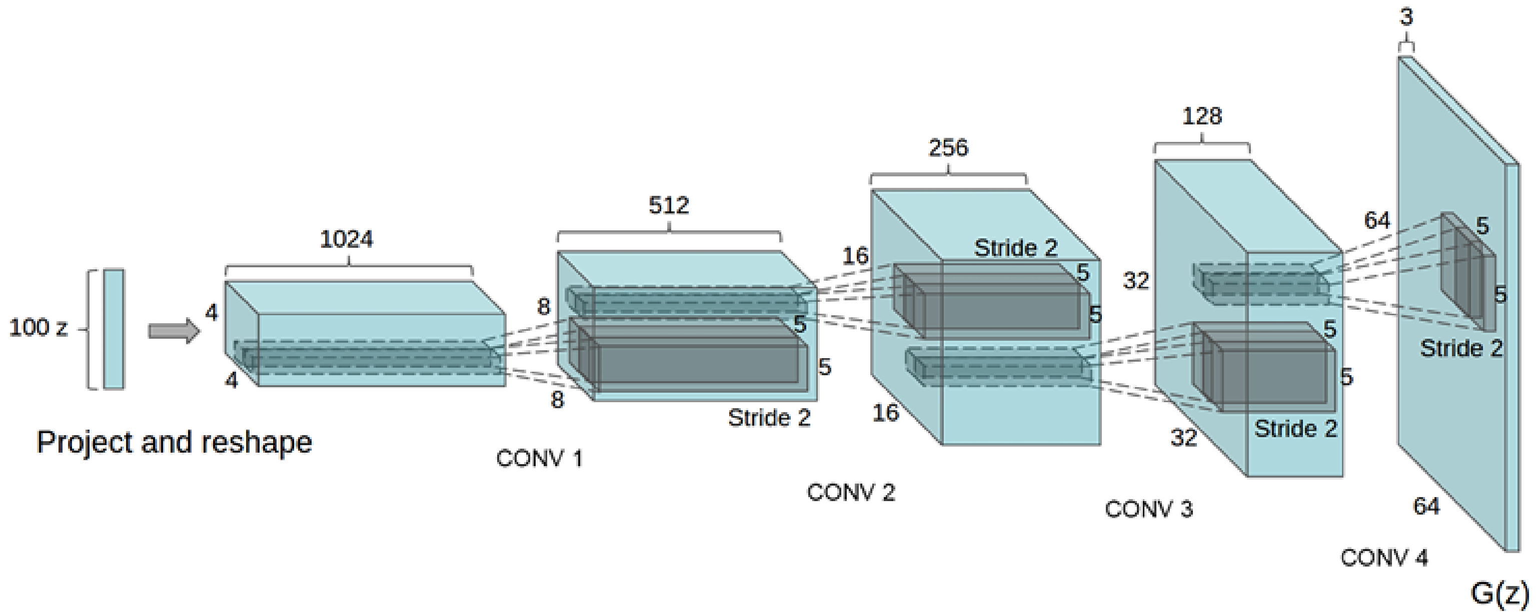


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, arXiv:1511.06434

DCGAN Simulation

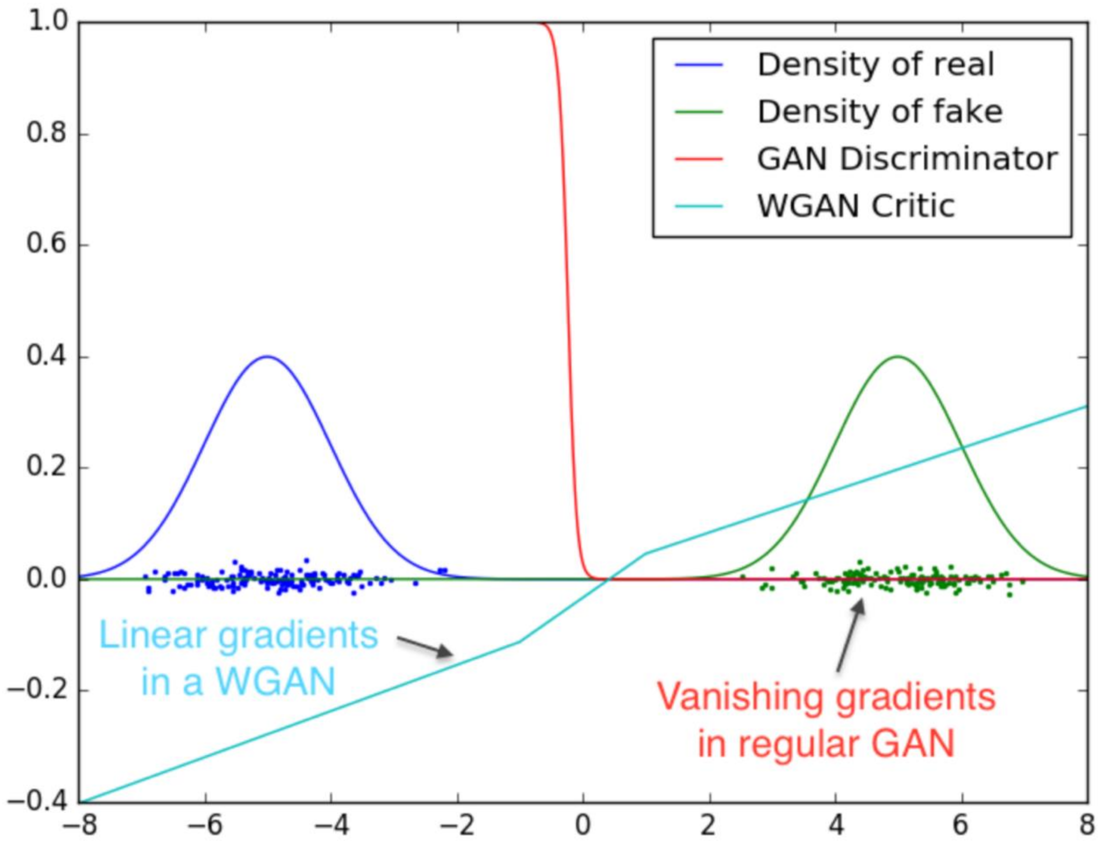
https://colab.research.google.com/drive/1pChdKaxL0ZhMJA_jxpD0k2WqE1C9yCf_?usp=sharing

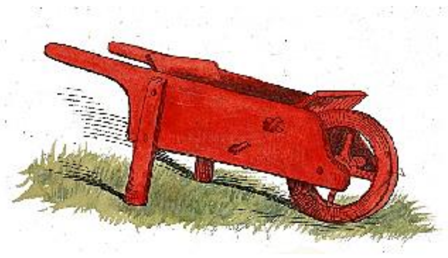
Some Issues with GAN

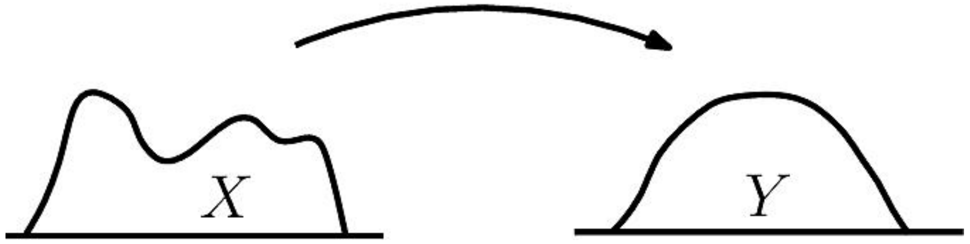
1. Vanishing gradient
2. Unstable optimization, even non-convergence
3. Mode collapse

- Thus, many (many) tricks: https://github.com/soumith/ganhacks
  - Initialize with Gaussian rather than Uniform
  - Avoid ReLU and MaxPool as they have sparse gradients
  - Add noise to inputs and let them decay
  - Don’t balance loss via statistics
  - …

Vanishing Gradient and Wasserstein GAN







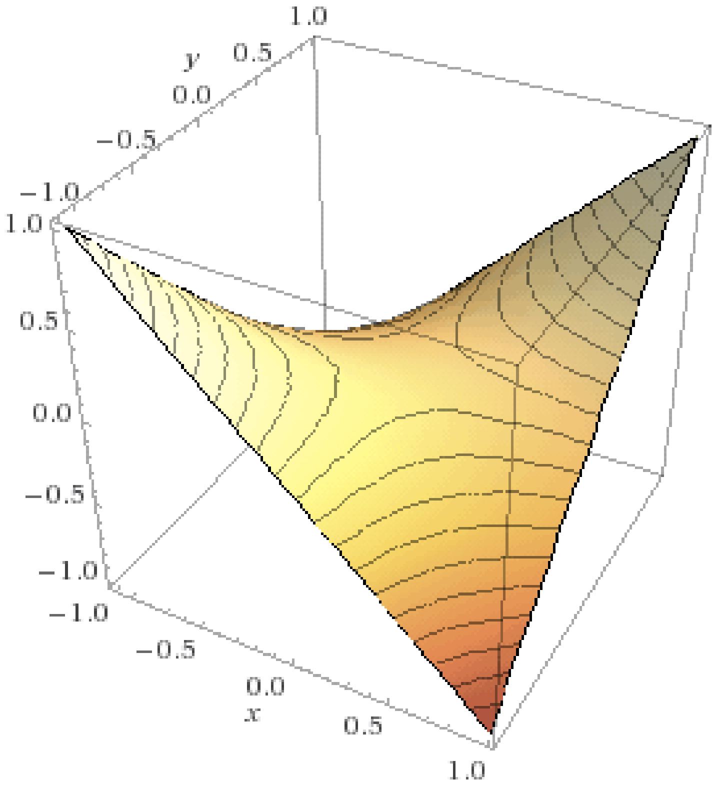
https://arxiv.org/pdf/1701.07875

Non-convergence

- GAN objective: $\min_x \max_y V(x, y)$
- Consider $V(x, y) = xy$
- Equilibrium (saddle point) at $x = y = 0$
- Using gradient descent results in a **spiral**

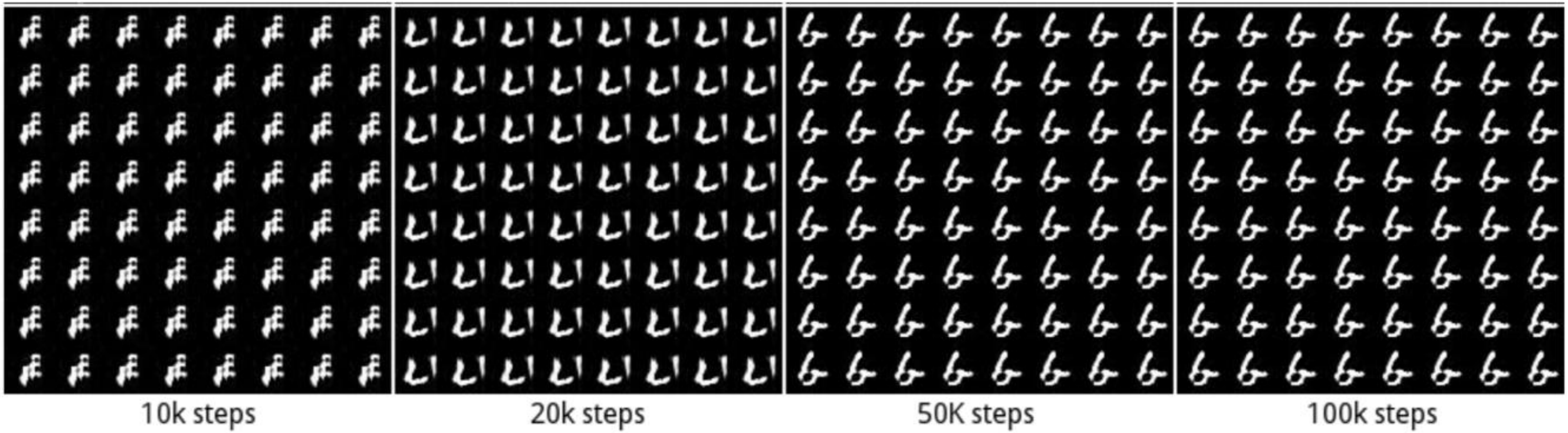
$$\frac{\partial x_t}{\partial t} = -\frac{\partial V(x_t, y_t)}{\partial x_t}, \qquad \frac{\partial y_t}{\partial t} = \frac{\partial V(x_t, y_t)}{\partial y_t}$$

$$x_t = x_0 \cos t - y_0 \sin t$$
$$y_t = x_0 \sin t + y_0 \cos t$$



(Goodfellow, 2016)

Mode collapse

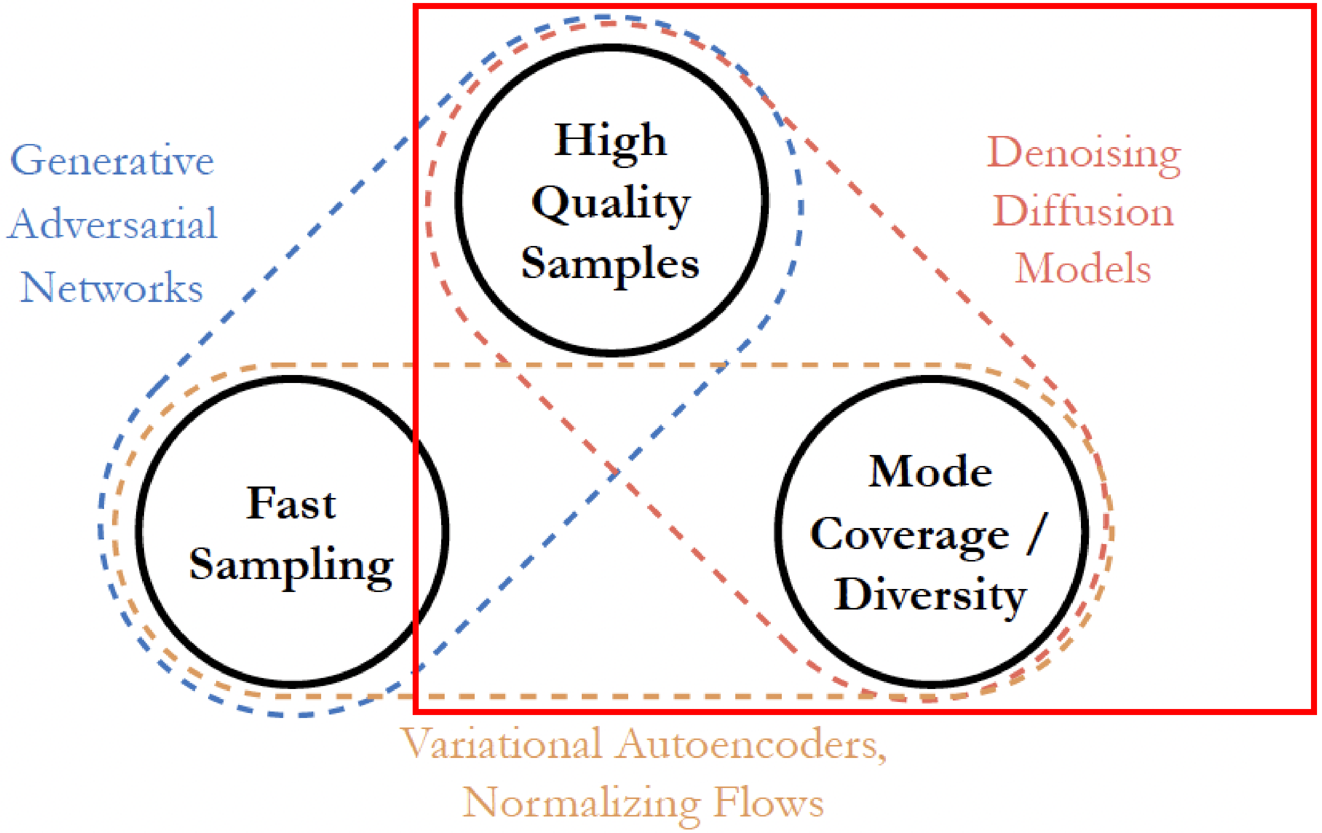


Source: Metz et al., 2017

Agenda

1. Generative Adversarial Networks (GANs)
2. **Diffusion models**

Trichotomy of Existing Generative Models



Xiao, 2022

High-Fidelity Generation for 1024x1024 Images

Diffusion Models: Motivation

what are the typical steps of drawing a picture for a human?

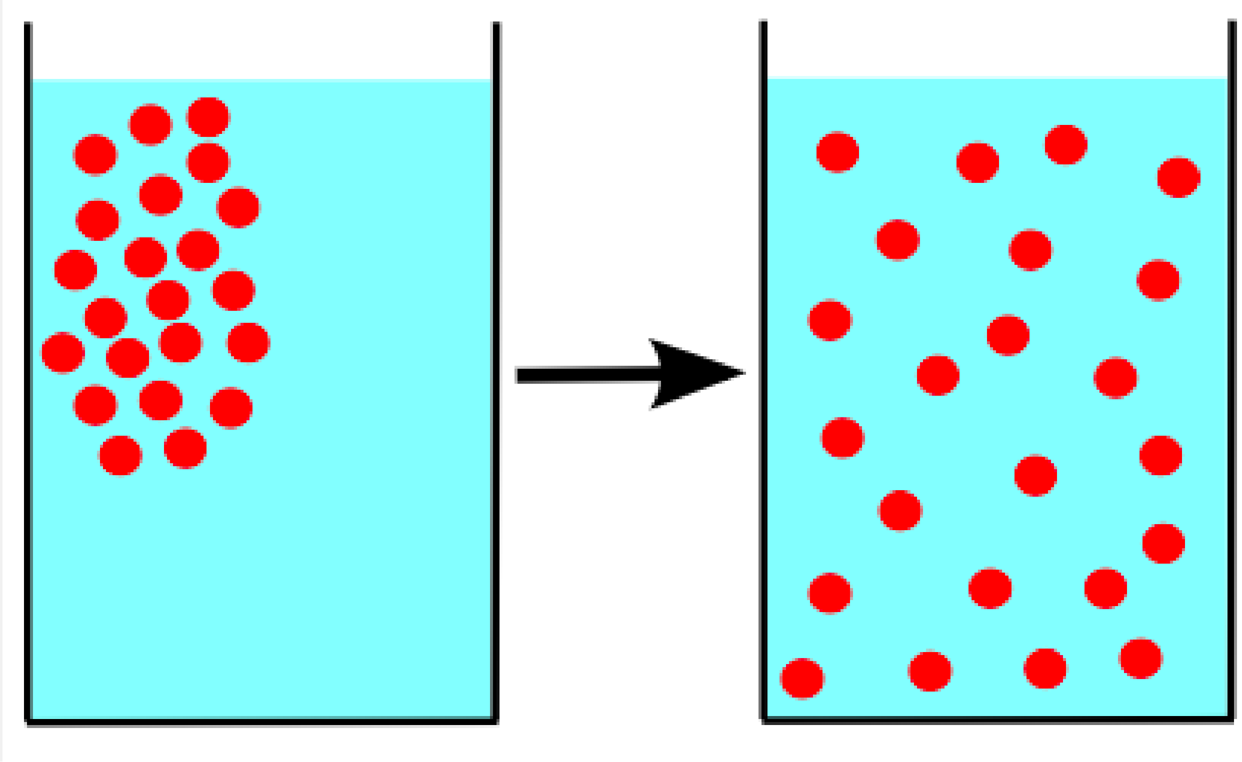
The typical steps a human follows when drawing a picture usually follow a loose structure that blends planning, sketching, refining, and finishing. While the exact process varies depending on the artist's style and medium, here's a general breakdown:

Diffusion Models: Motivation (cont.)

- Takeaway: There is (roughly) a unified **structure** for creation
- Previously, VAE and GAN generate samples in "one-shot" (one pass of Feed-forward NN)

- Question: Can we divide the generation process into **small, manageable** steps?
- **Diffusion Models!**

Diffusion Process = Noising

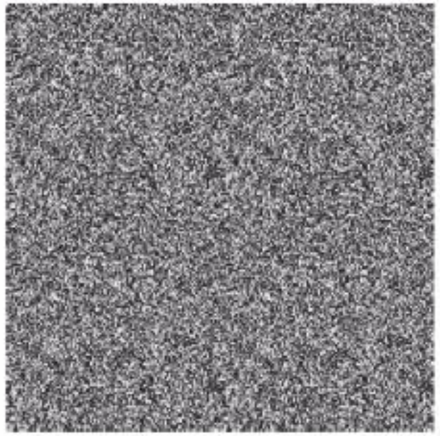


Diffusion Process = Noising (cont.)

From Denoisng to Image Generation

Question: How to use a **magic denoiser** to create new images?

1. Create some noise
2. Apply your magic denoiser
3. If still noisy, repeat step 2 again…
4. Done!



$X \in R^{1024}$

Diffusion Model: Structure Overview

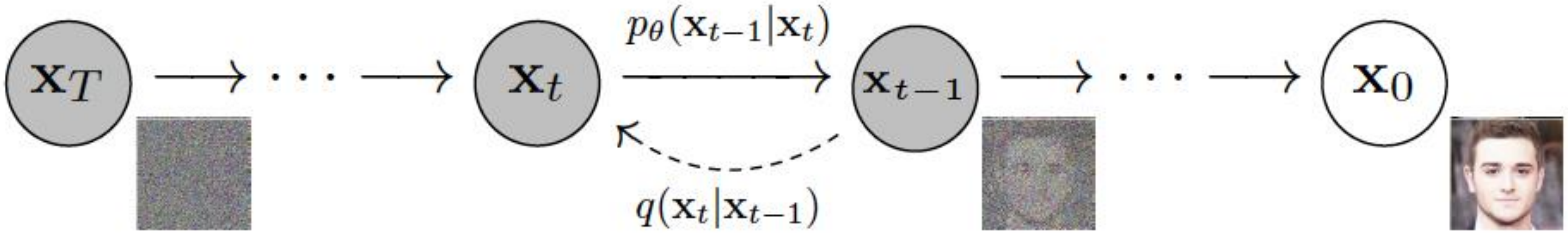


Figure 2: The directed graphical model considered in this work.

(Ho, 2020)

Diffusion Model: Forward Process

- **Forward** Process (for noising)

$$x_0 \to x_1 \to \cdots \to x_T$$

Original clean Image Veeery noisy version…

  - Divided into multiple steps (T could be 50-1000)

- During each step, a small amount of (Gaussian) noise is added:

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

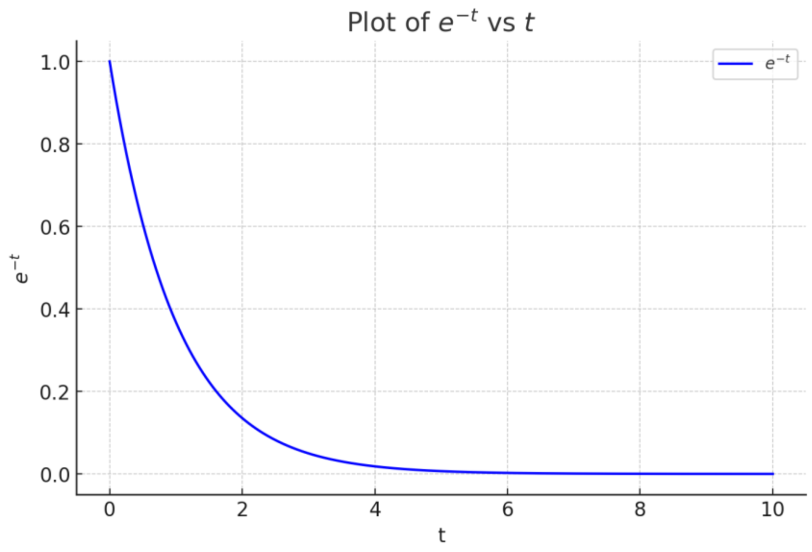
"noise schedule" Gaussian noise

- Check your understanding: Does larger β_t imply faster/slowlier noising?

What is the distribution of x_T?

$$
\begin{aligned}
x_T &= \sqrt{1-\beta_T}x_{T-1} + \sqrt{\beta_T}\epsilon_T \\
&= \sqrt{\alpha_T}x_{T-1} + \sqrt{1-\alpha_T}\epsilon_T \\
&= \sqrt{\alpha_T\alpha_{T-1}}x_{T-2} + \sqrt{\alpha_T(1-\alpha_{T-1})}\epsilon_{T-1} + \sqrt{1-\alpha_T}\epsilon_T \\
&= \cdots \\
&= \sqrt{\alpha_1\cdots\alpha_T}x_0 + \sqrt{1-\alpha_1\cdots\alpha_T}\bar{\epsilon}_T \\
&= \sqrt{\bar{\alpha}_T}x_0 + \sqrt{1-\bar{\alpha}_T}\bar{\epsilon}_T
\end{aligned}
$$

Thus, we can immediately sample x_t from x_0…



Diffusion Model: Reverse Process

- **Reverse** Process (for de-noising)

$$x_T \rightarrow x_{T-1} \rightarrow \cdots \rightarrow x_0$$

1. Create initial noise: $x_T \sim \mathcal{N}(0, I)$
2. Denoise at each step: $x_{t-1} = \frac{x_t - \beta_t \epsilon_t}{\sqrt{1-\beta_t}}$

- Any problems?
  1. We don't know the ϵ_t that produces x_t, which is **random**
  2. Even though $x_t | x_{t-1}$ is Gaussian, $x_{t-1} | x_t$ is **typically not** -> how to sample??

Diffusion Model: Reverse Process (cont.)

- Let's tackle the second issue first: instead of $x_{t-1}|x_t$, note that
$$q(x_{t-1}|x_0, x_t) = \mathcal{N}(\tilde{\mu}_t(x_0, x_t), \tilde{\beta}_t)$$
  - $\tilde{\beta}_t$: what we have
  - Gaussian: easy sampling
  - $\tilde{\mu}_t(x_0, x_t)$: known function

- Message: If we know x_0, we can easily sample for x_{t-1}
- How to obtain x_0??

Diffusion Model: Training

- We obtain x_0 by **training** for the noise $\bar{\epsilon}_t$ as a function of x_t
- For each (batch of) x_0:

1. Sample $\bar{\epsilon}_t \sim \mathcal{N}(0, I)$
2. Obtain $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \bar{\epsilon}_t$
3. Minimize $||\epsilon_\theta(x_t) - \bar{\epsilon}_t||^2$ (using GD/SGD/…)
    - Effectively, this maximizes the **ELBO** (Ho et al., 2020)

- During each sampling step, obtain $\hat{x}_0$ from $\epsilon_\theta(x_t)$, and use $\tilde{\mu}_t(\hat{x}_0, x_t)$ as Gaussian mean

Diffusion Model: Algorithms Overview

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ 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

(Ho et al., 2020)

Diffusion Model: Algorithms Overview

Algorithm 2 Sampling

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

(Ho et al., 2020)

Diffusion Model Simulation

https://colab.research.google.com/drive/1kzg-5iHQN9ZxW66uEsijLgr0w1rj2nai?usp=drive_link

Diffusion Models as sequential VAE

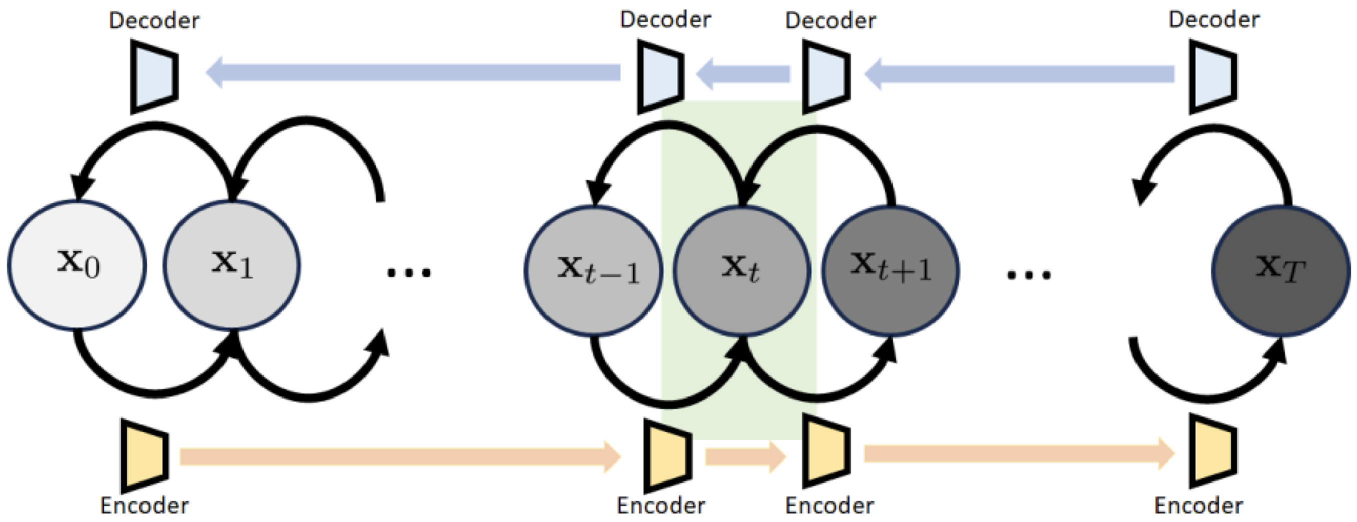
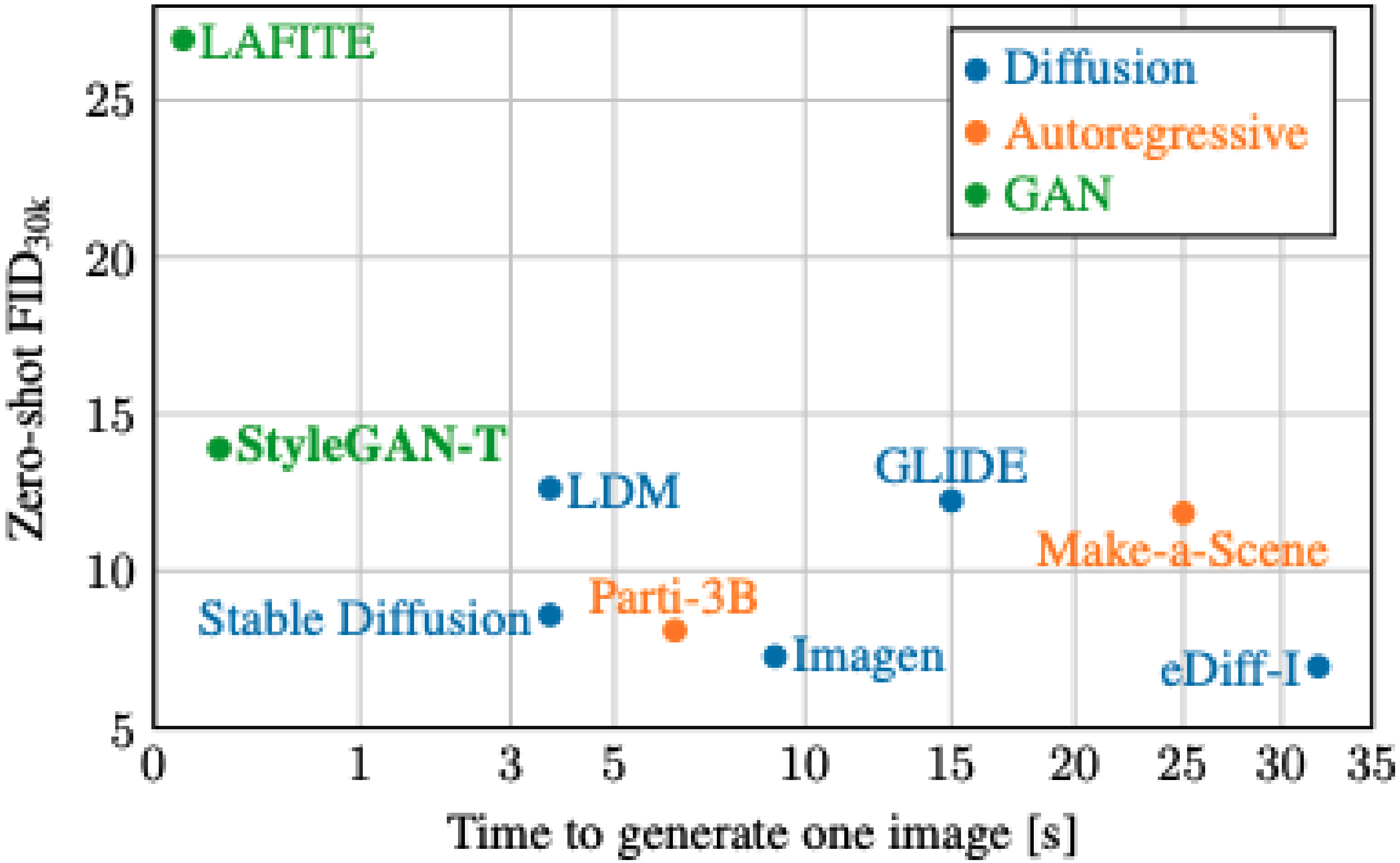


Figure 2.1: Variational diffusion model by Kingma et al [22]. In this model, the input image is $\mathbf{x}_0$ and the white noise is $\mathbf{x}_T$. The intermediate variables (or states) $\mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$ are latent variables. The transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ is analogous to the forward step (encoder) in VAE, whereas the transition from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ is analogous to the reverse step (decoder) in VAE. In variational diffusion models, the input dimension and the output dimension of the encoders/decoders are identical.

https://arxiv.org/pdf/2403.18103

Issues with Diffusion Model

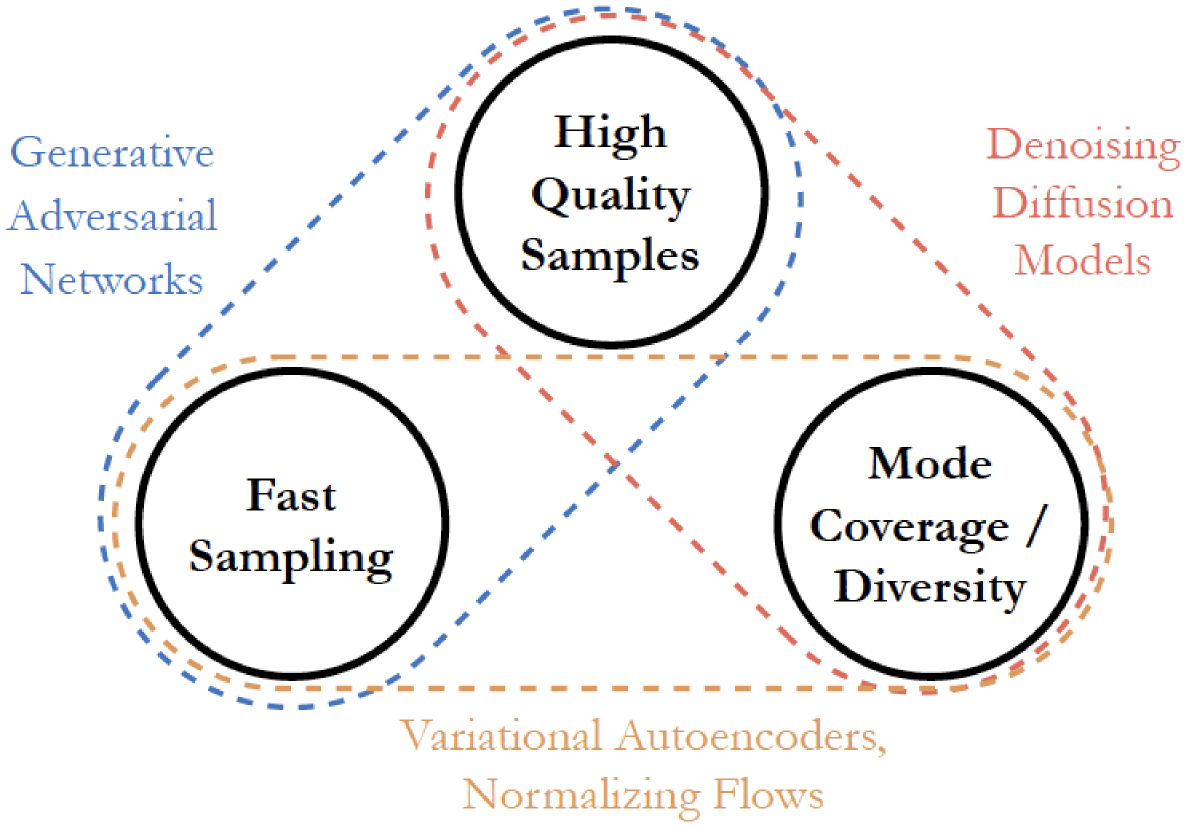


https://arxiv.org/abs/2301.09515

Last words: How to choose?

GANs	Diffusion Models
• Pros: fast sampling • Cons: unstable training, mode collapse	• Pros: high-quality samples, stable training, theoretical guarantees • Cons: slow!!!

Trichotomy of Existing Generative Models



Xiao, 2022

Which model would you choose?

- Entertainment and gaming?
- Medical imaging?
- Data imputation (for missing data)?
- Autonomous driving?
- Drug discovery?
- …

Homework

For 2 of 3 models below, change at least 3 parameters of the model in class; examine any differerence (quantitatively or qualitatively)

1. VAE
2. DCGAN
3. Diffusion Model

- Send your report to chen.11020@buckeyemail.osu.edu

References

- Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks, https://arxiv.org/pdf/1701.00160
- DDPM original paper: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
- A blog on DDPM: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/#nice