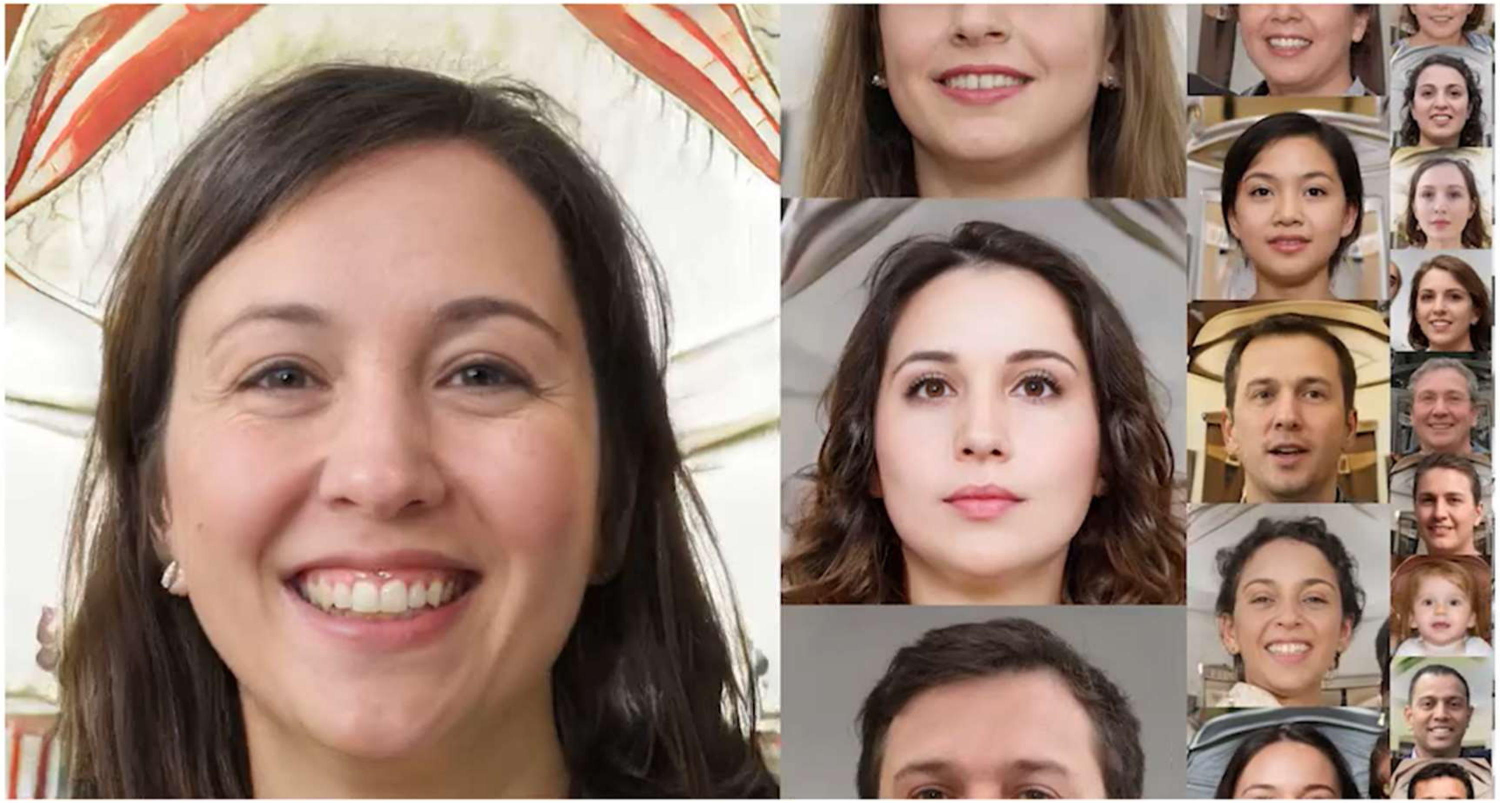
# 生成对抗网络
# Generative Adversarial Network (GAN)



# Taxonomy of Generative Models

介绍三种不同类型的
常用的深度生成模型

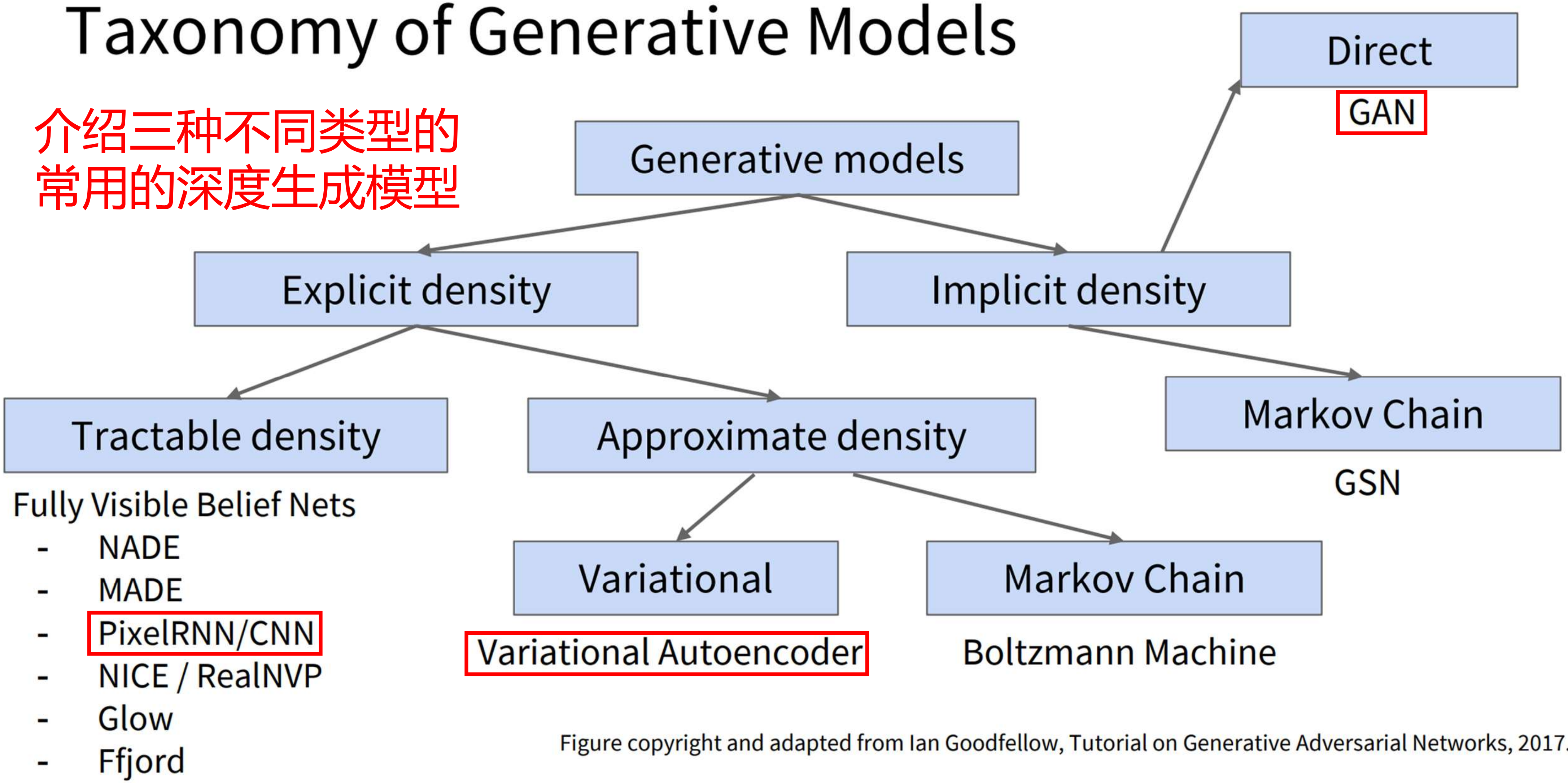



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# So far...

PixelRNN / PixelCNN **define tractable density function, optimize likelihood of training data:**

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs **define intractable density function with latent z:**

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

**Assume z a simple distribution, and optimize with Reconstruction and Regularization**

# So far...

PixelRNN / PixelCNN define tractable density function, optimize likelihood of training data：

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent z:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Assume z a simple distribution, and optimize with Reconstruction and Regularization

Both try to explicitly model the density

# So far...

PixelRNN / PixelCNN define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent z:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Assume z a simple distribution, and optimize with Reconstruction and Regularization

What if we give up on explicitly modeling density, and just want ability to sample?

# So far...

PixelRNN / PixelCNN define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent z:

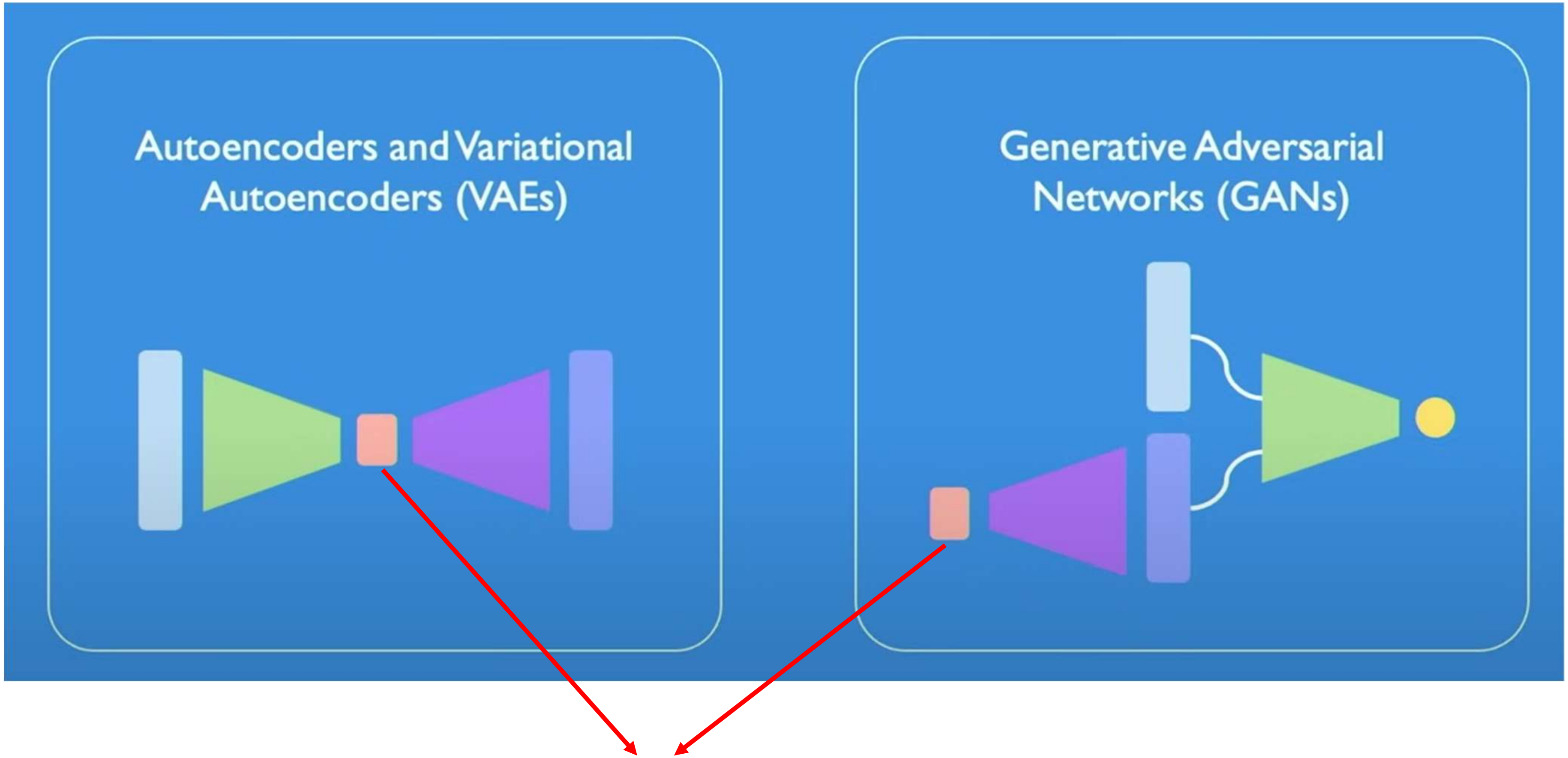
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Assume z a simple distribution, and optimize with Reconstruction and Regularization

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: do not model any explicit density function!

# VAEs v.s. GANs




Latent Variable (隐变量)

# GANs: Just to Sample

**Problem:** it's difficult to explicitly model complex distribution. No direct way to do this!

**Idea:** don't explicitly model density, and instead just sample to generate new instances.

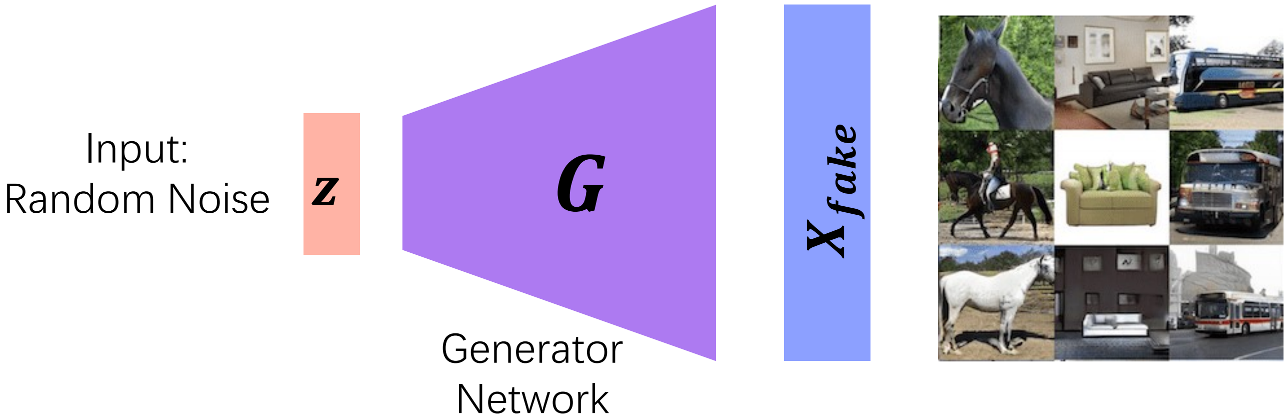
**Solution:** sample from something relatively simple (e.g., noise), learn a Transformation to the training data distribution.

# GANs: Just to Sample

**Problem:** it's difficult to explicitly model complex distribution. No direct way to do this!

**Idea:** don't explicitly  model density, and instead just sample to generate new instances.

**Solution:** sample from something relatively simple (e.g., noise), learn a Transformation to the training data distribution.

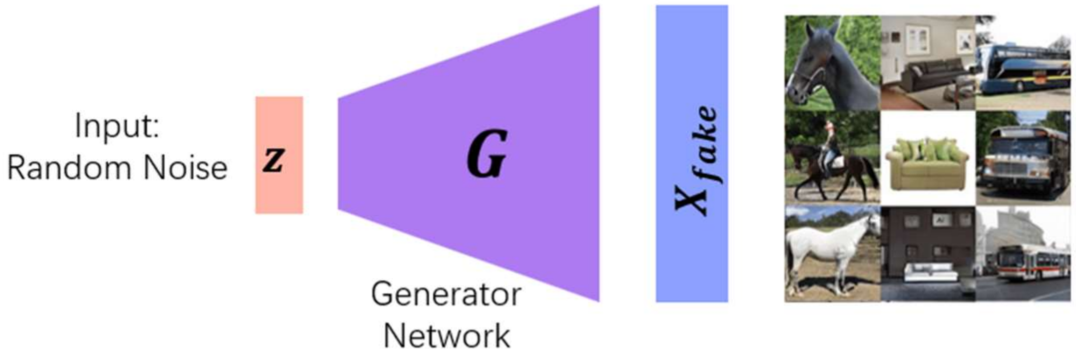


# Key challenges

**Problem:** it's difficult to explicitly model complex distribution. No direct way to do this!

**Solution:** sample from something relatively simple (e.g., noise), learn a Transformation to the training data distribution.

We don't know which sample z maps to which training image -> can't learn by reconstructing training images

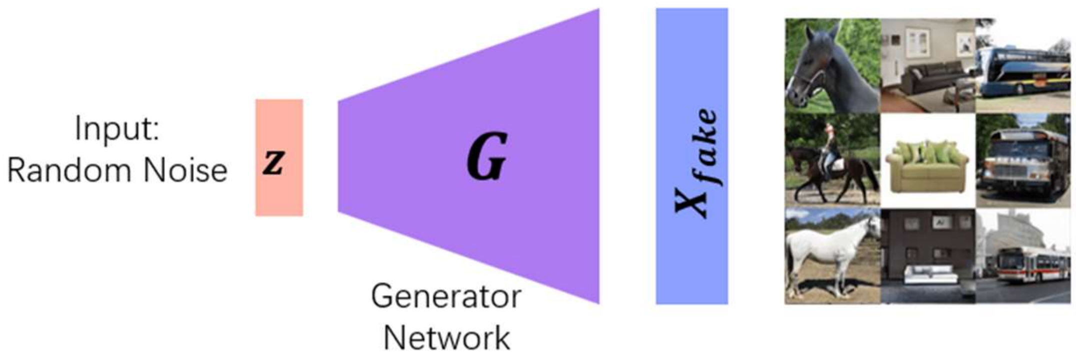


# Key challenges

**Problem:** it's difficult to explicitly model complex distribution. No direct way to do this!

**Solution:** sample from something relatively simple (e.g., noise), learn a Transformation to the training data distribution.

We don't know which sample z maps to which training image -> can't learn by reconstructing training images



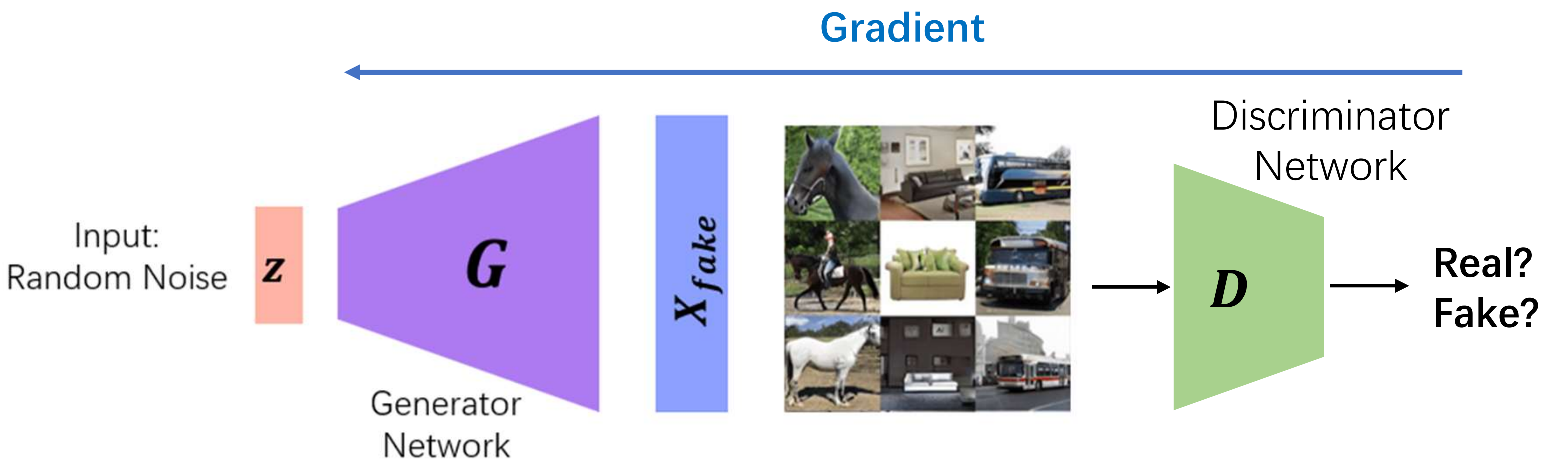

Objective: generated images should look "real"

# Key challenges

**Problem:** it's difficult to explicitly model complex distribution. No direct way to do this!

**Solution:** sample from something relatively simple (e.g., noise), learn a Transformation to the training data distribution.

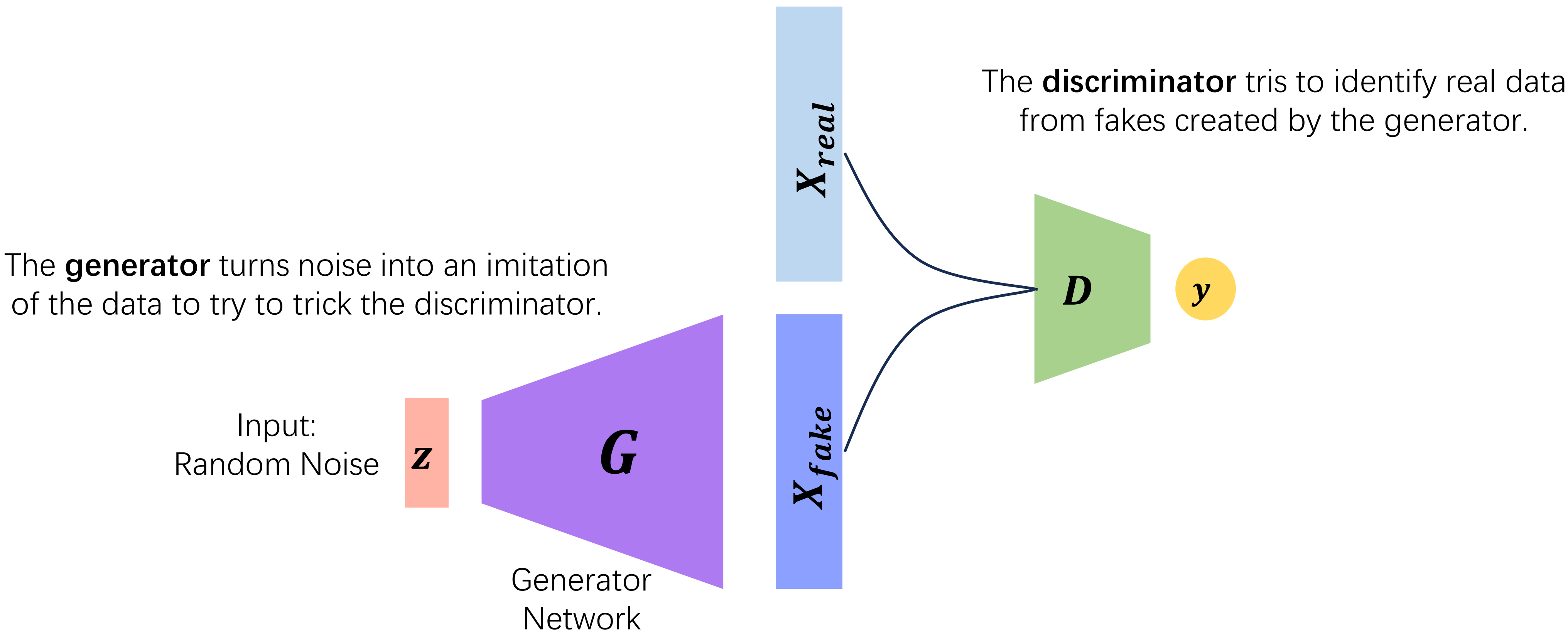
We don't know which sample z maps to which training image -> can't learn by reconstructing training images




Solution: Use a discriminator network to tell whether the generate image is within data distribution ("real") or not

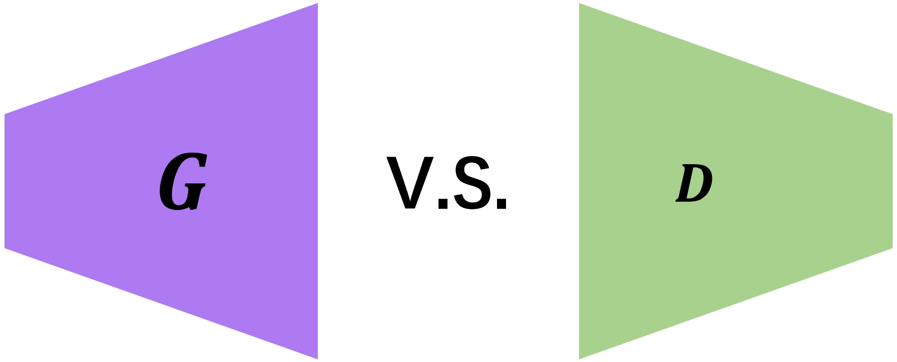
# Generative Adversarial Networks (GANs)

**Generative Adversarial Networks (GANs)** are a way to make a generative model by having two neural networks compete with each other!



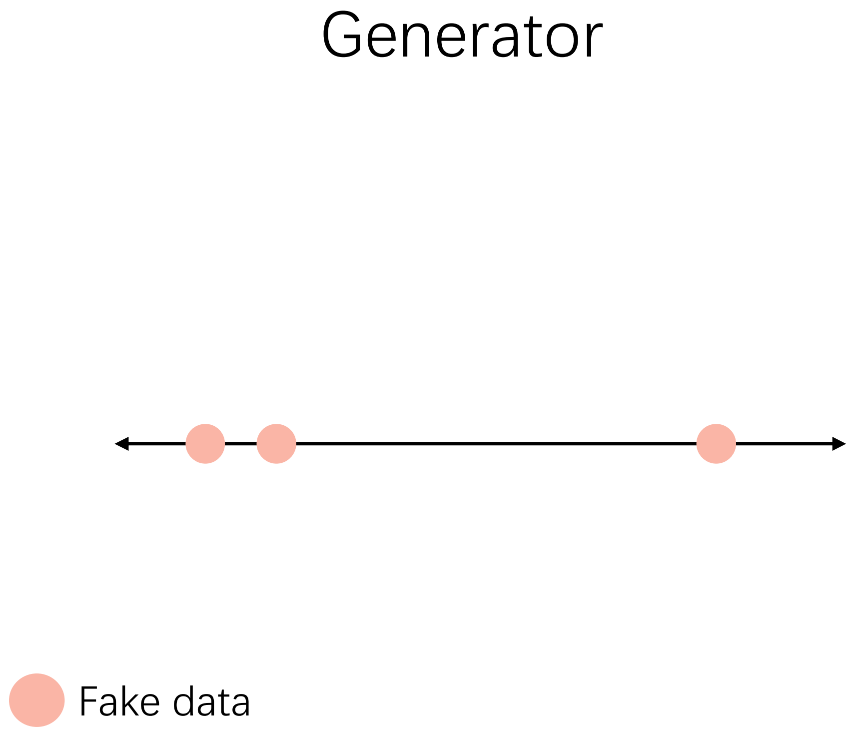
# Two-player game

**Generative Adversarial Networks (GANs)** are a way to make a generative model by having two neural networks compete with each other!

# Intuition behind GANs

**Generator** starts from noise to try to create an imitation of the data.



# Intuition behind GANs

**Discriminator** looks at both real data and fake data created by the generator.

Discriminator

Generator

Fake data

# Intuition behind GANs

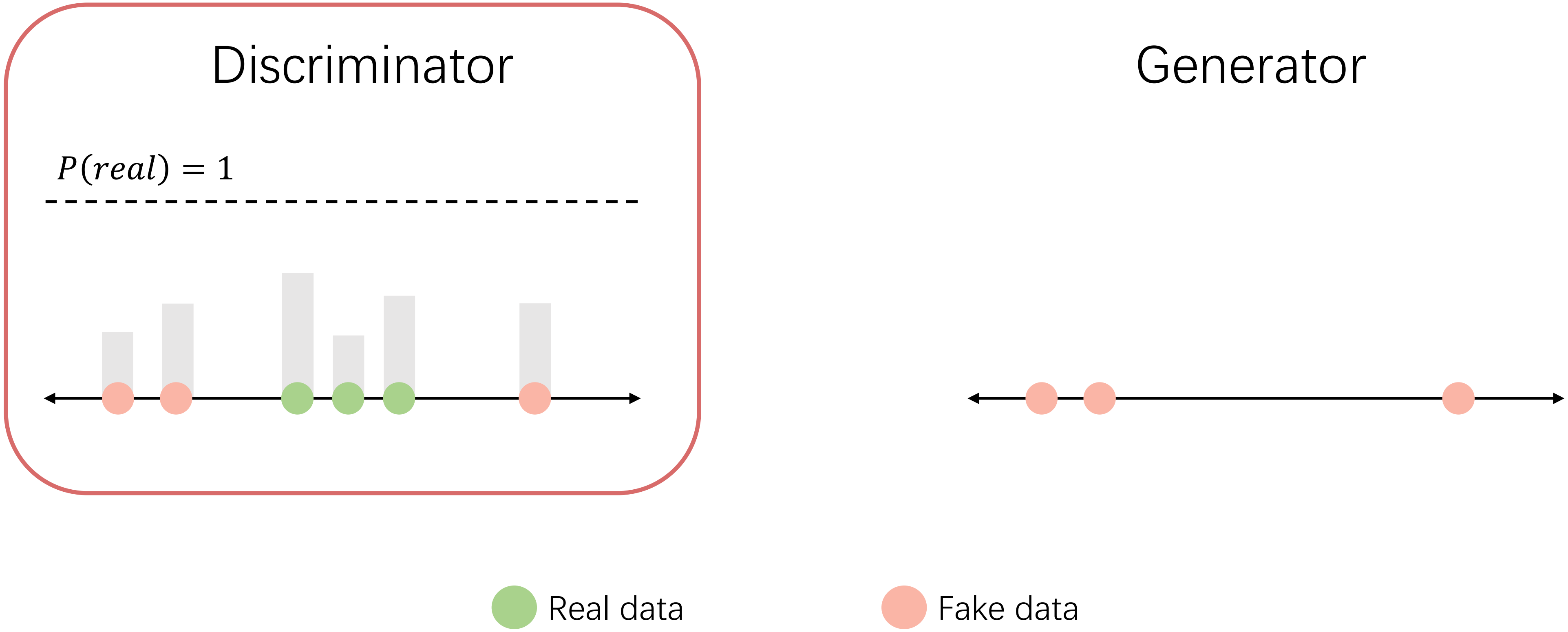
**Discriminator** looks at both real data and fake data created by the generator.

Discriminator

Generator

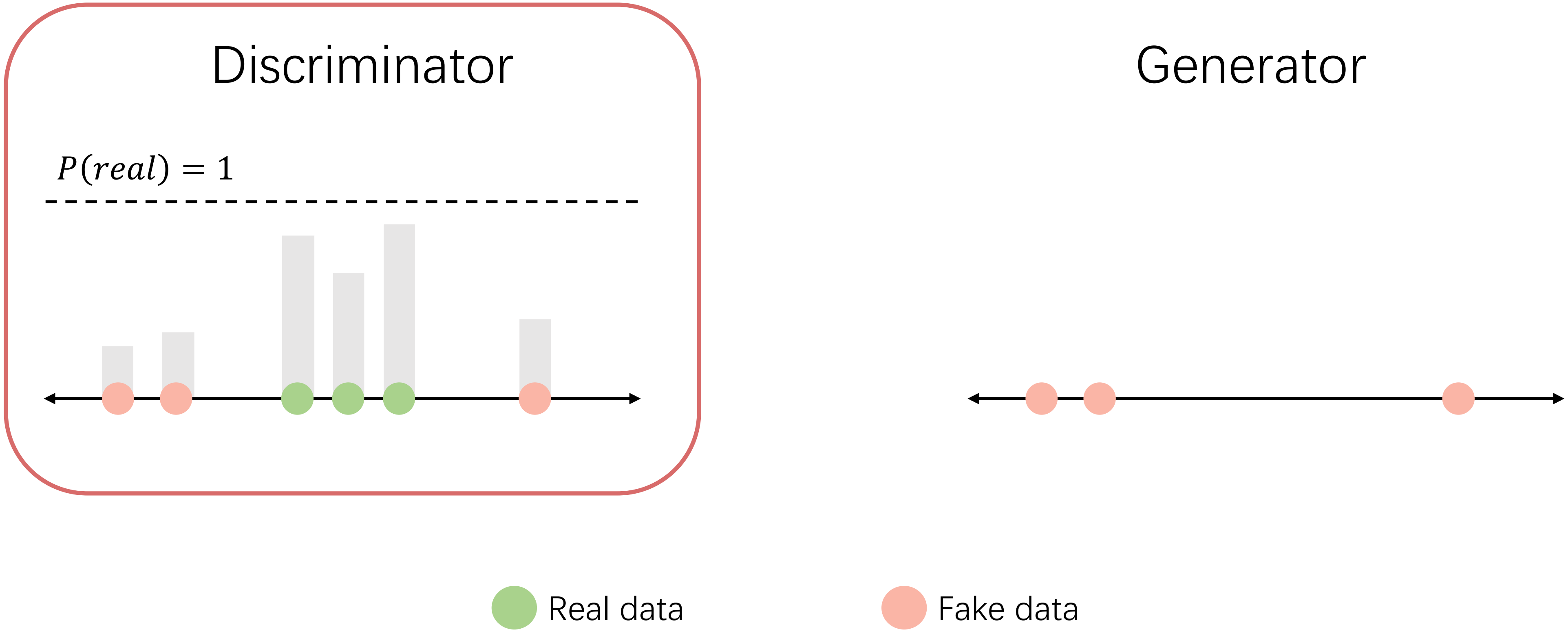
# Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.



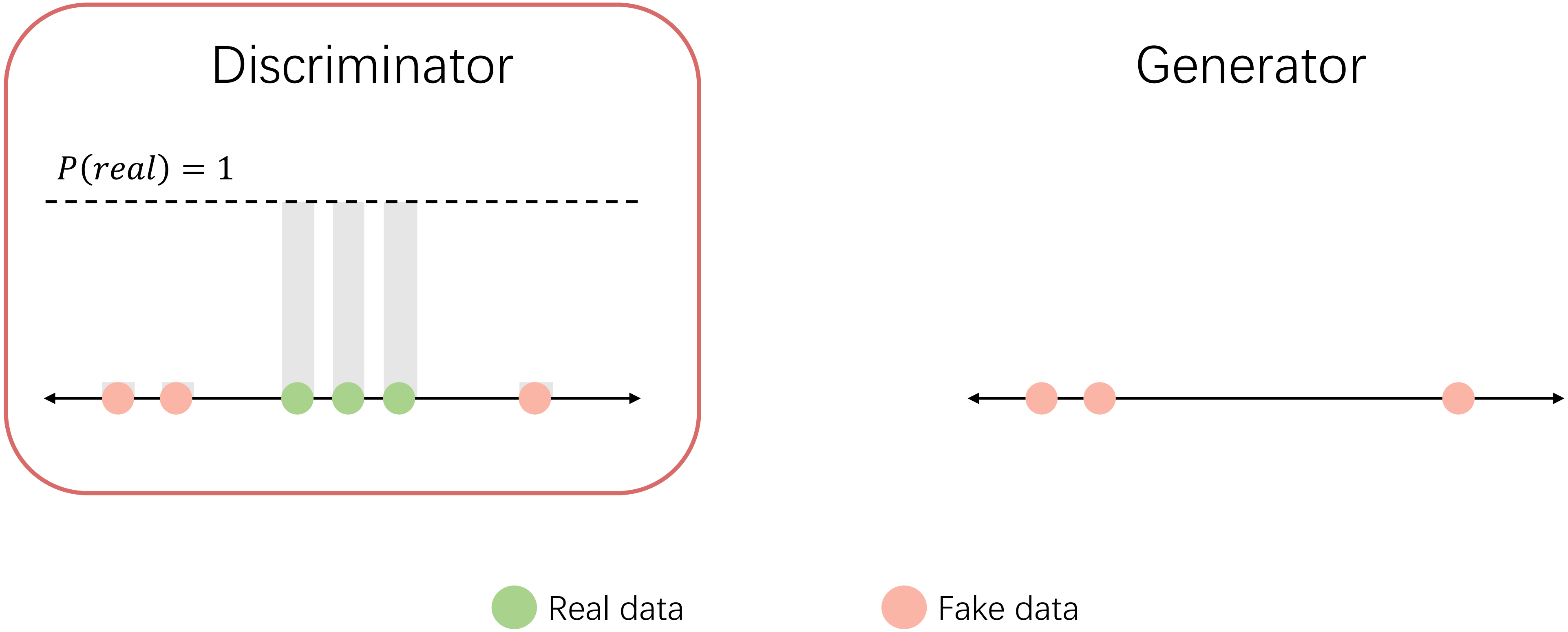
# Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.



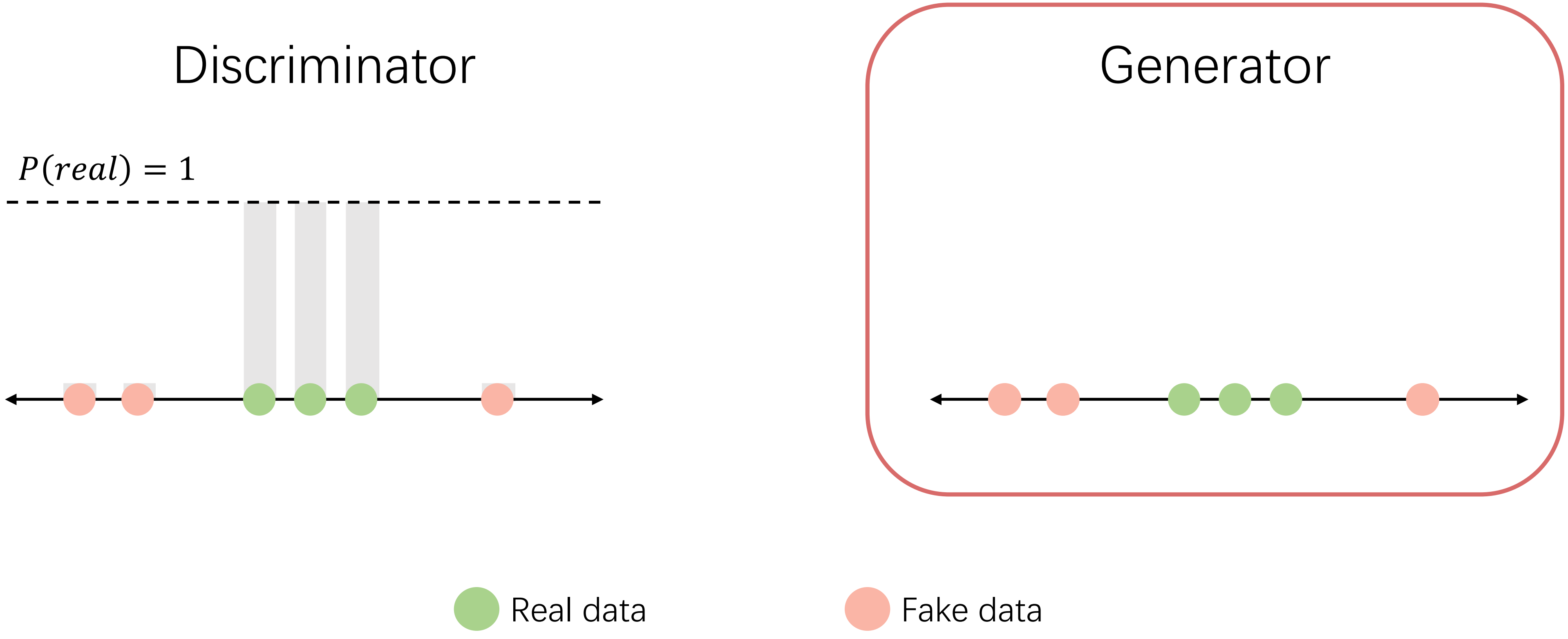
# Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.



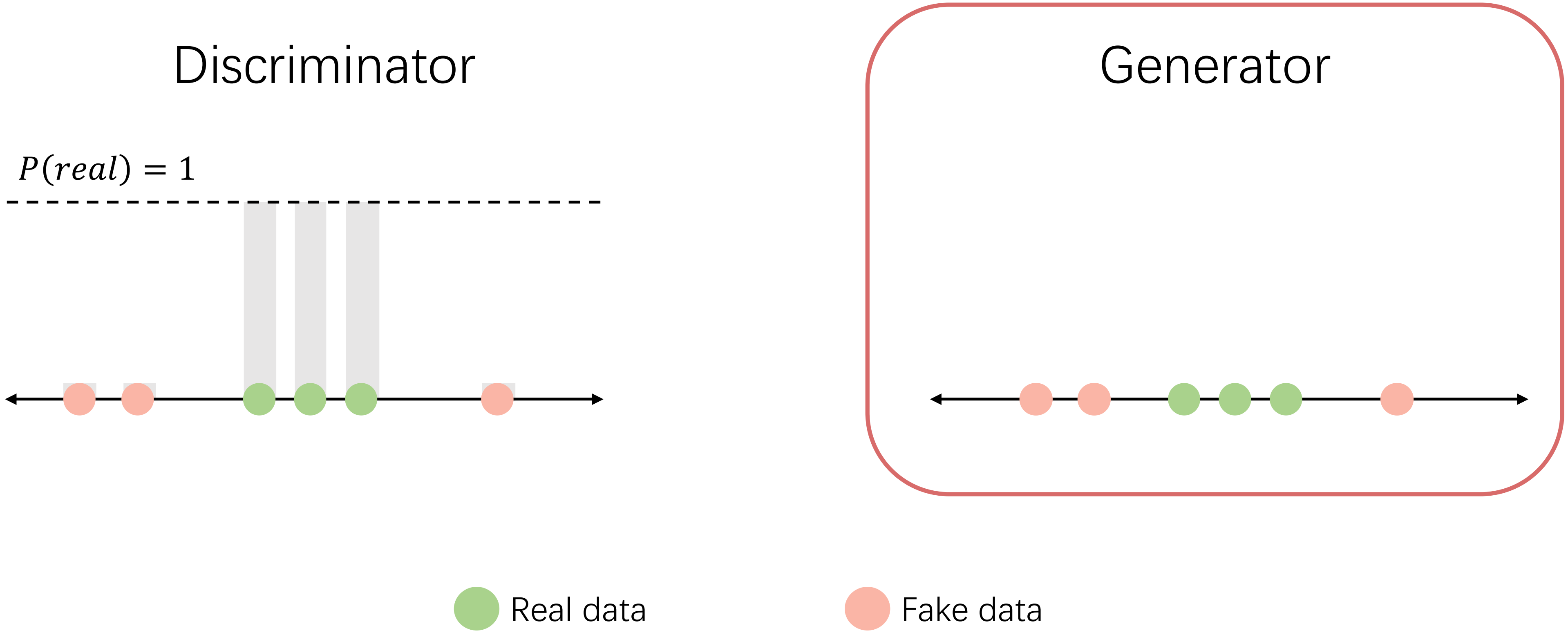
# Intuition behind GANs

**Generator** tries to improve its imitation of the data.



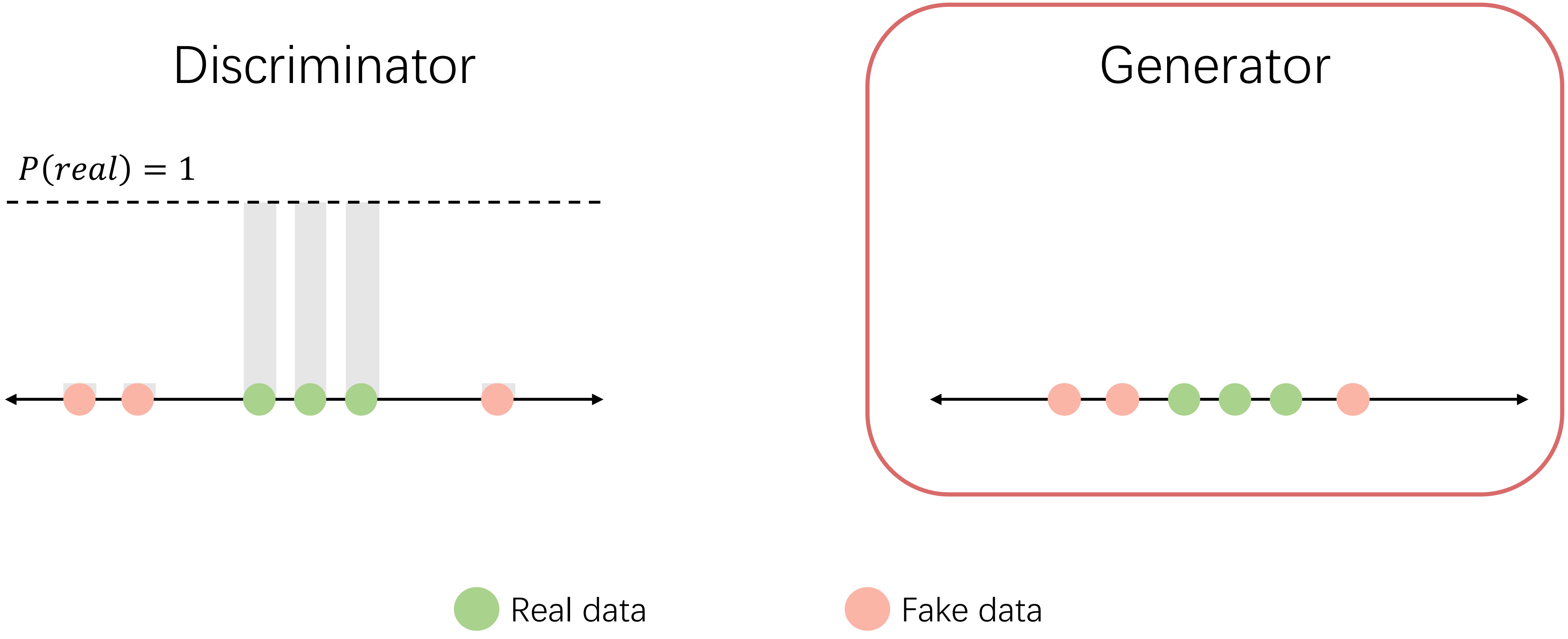
# Intuition behind GANs

**Generator** tries to improve its imitation of the data.



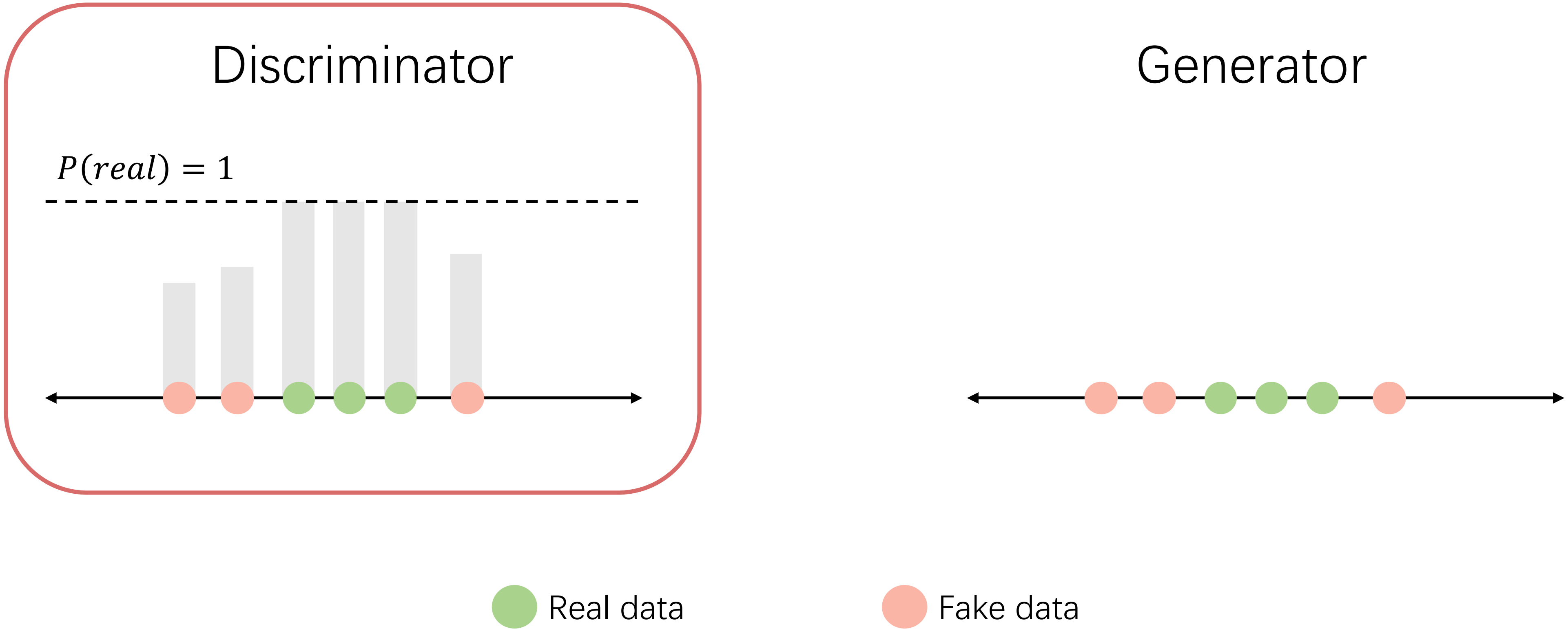
# Intuition behind GANs

**Generator** tries to improve its imitation of the data.



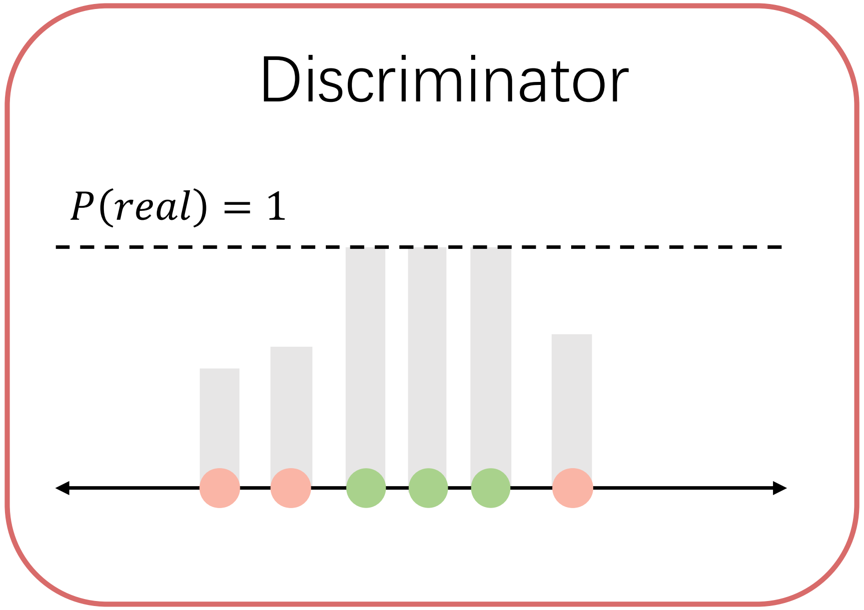
# Intuition behind GANs

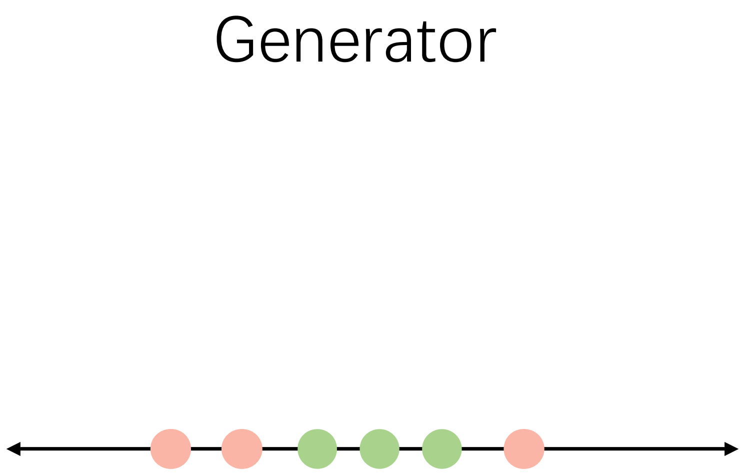
**Discriminator** tries to predict what's real and what's fake.



# Intuition behind GANs

**Discriminator** tries to predict what's real and what's fake.

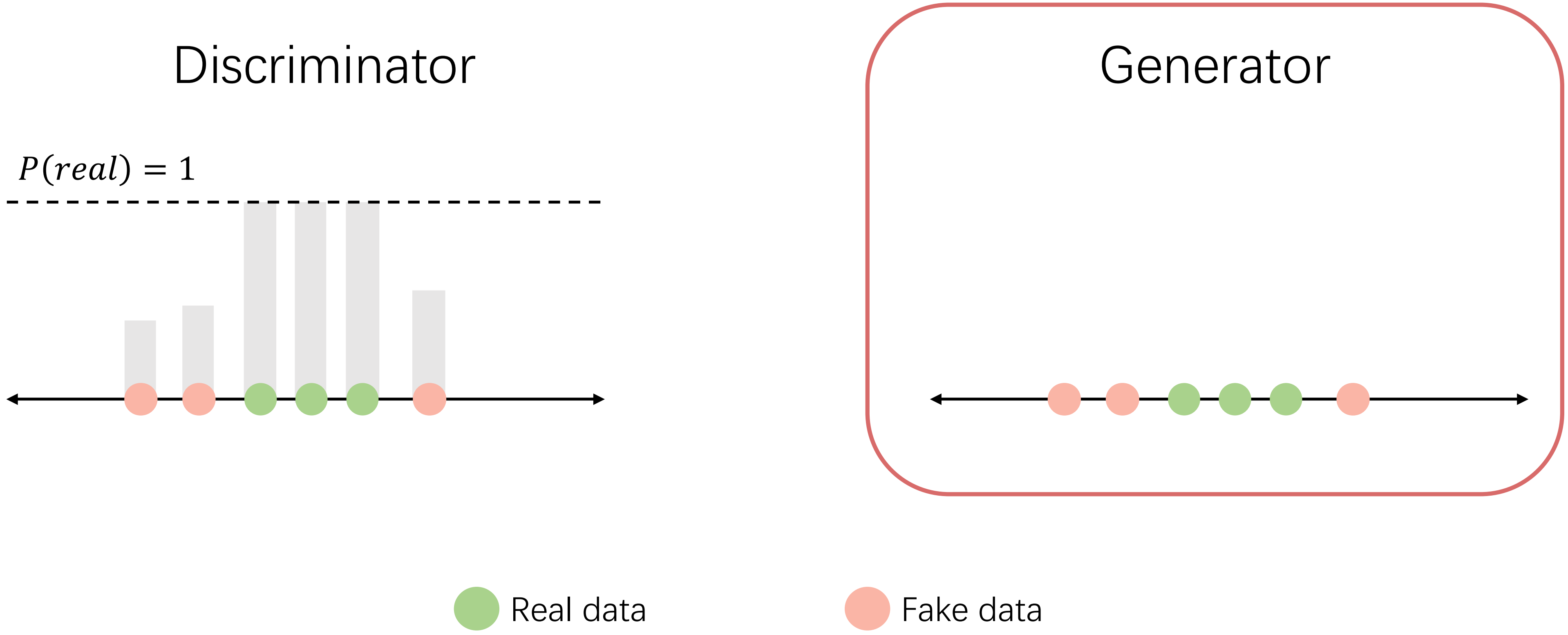



Real data    Fake data

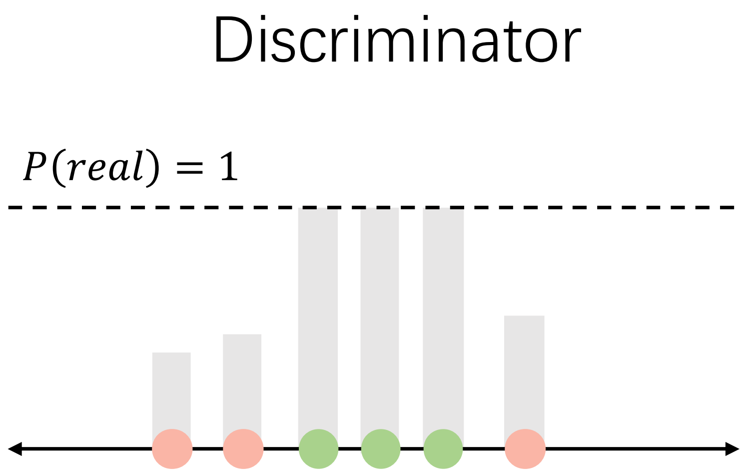
# Intuition behind GANs

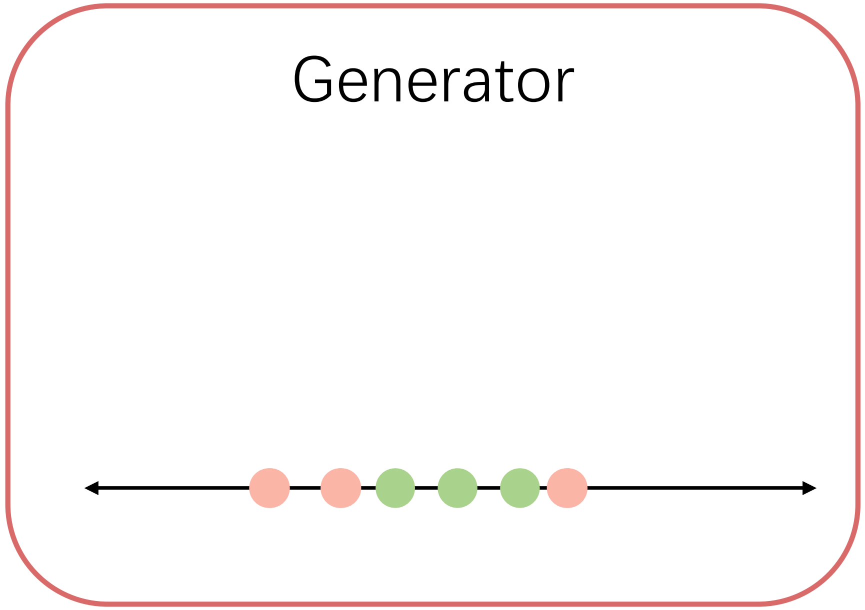
**Generator** tries to improve its imitation of the data.



# Intuition behind GANs

**Generator** tries to improve its imitation of the data.



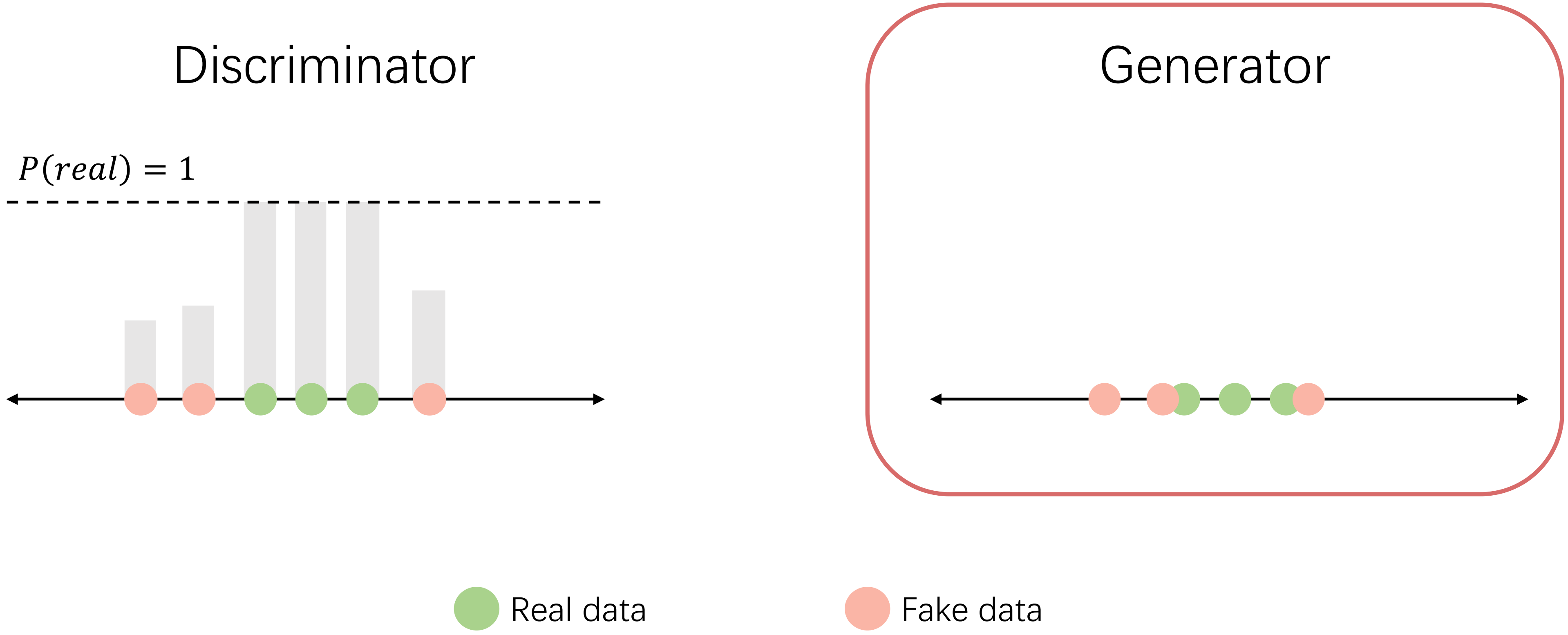




Real data

Fake data

# Intuition behind GANs

**Generator** tries to improve its imitation of the data.



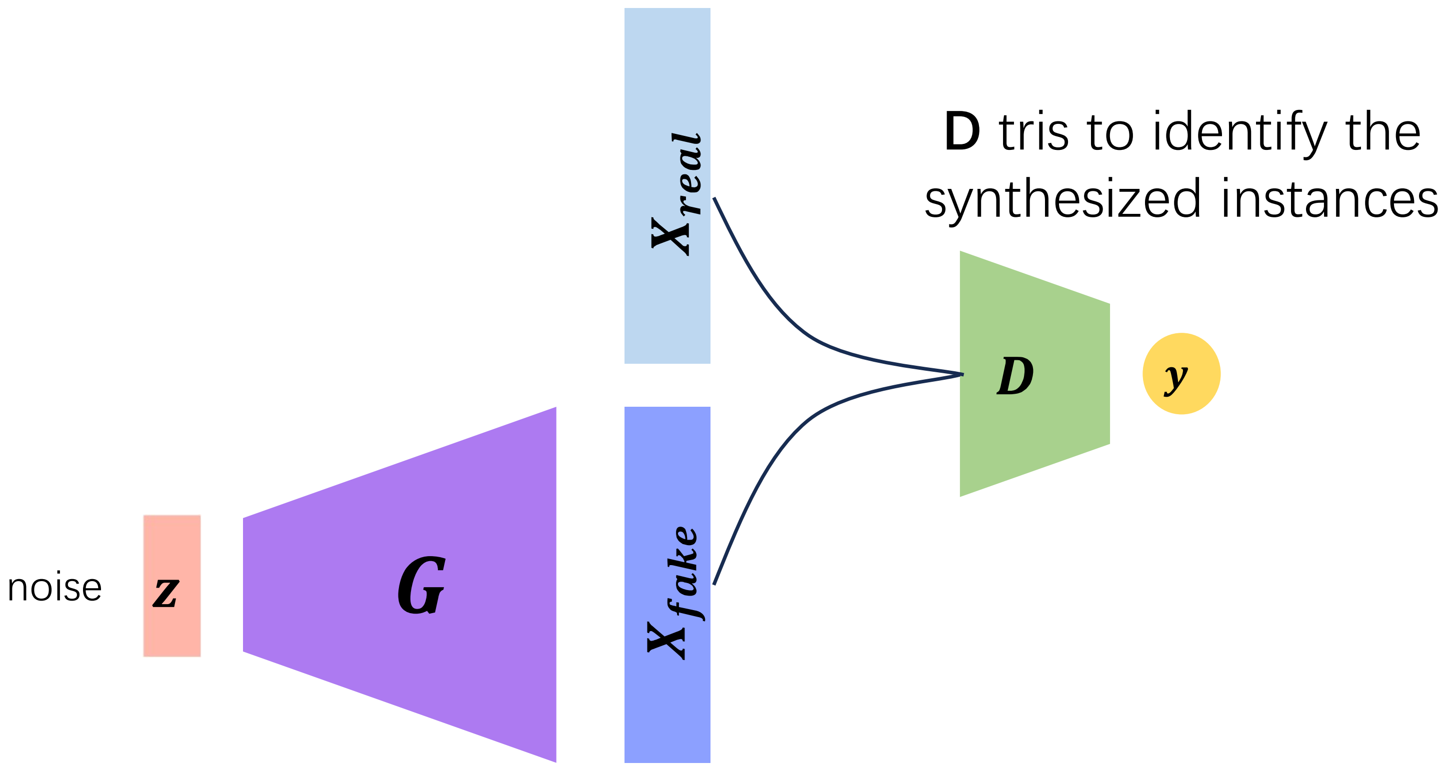
# Intuition behind GANs

**Discriminator** tris to identify real data from fakes created by the generator.
**Generator** tries to create imitations of data to trick the discriminator.

How to implement the intuition?

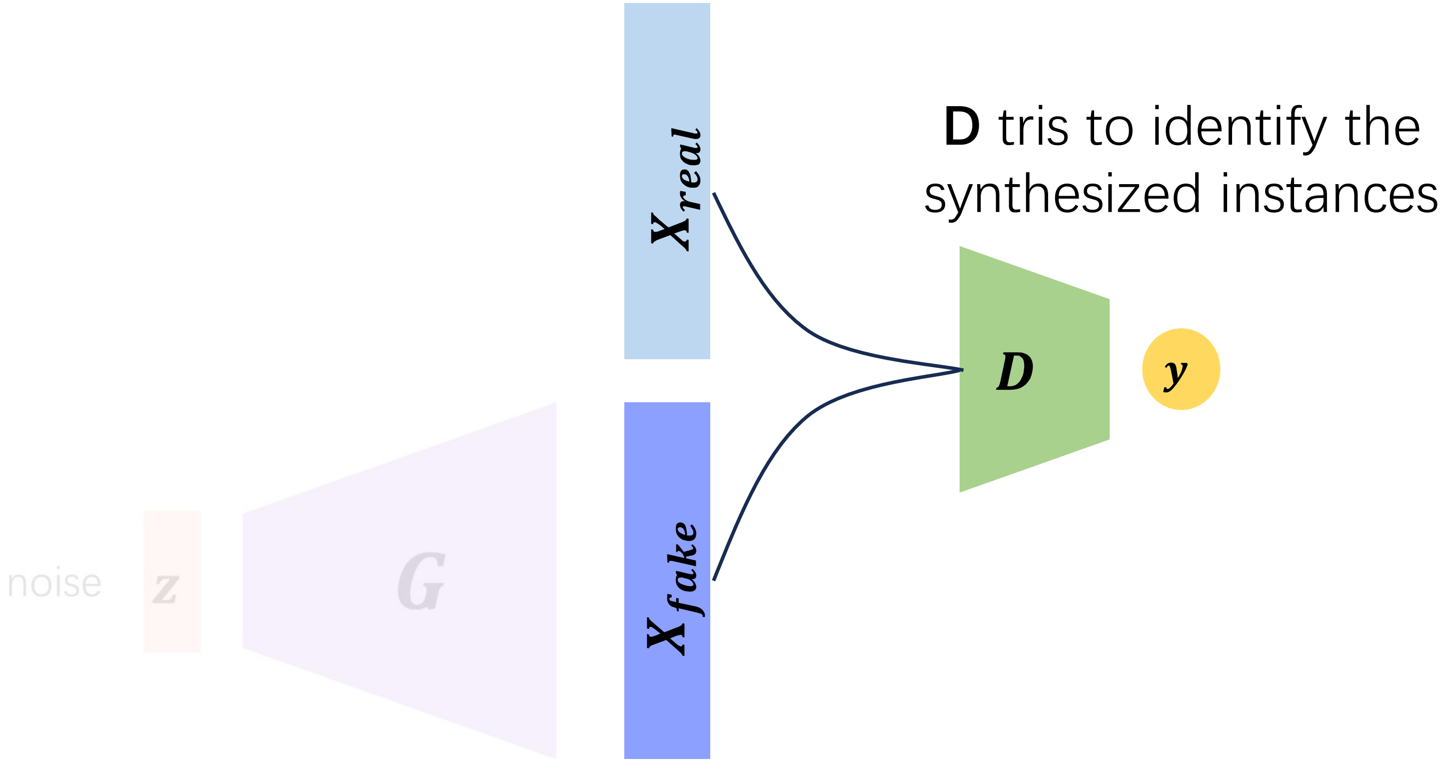
How to train a GAN?

# Training GANs



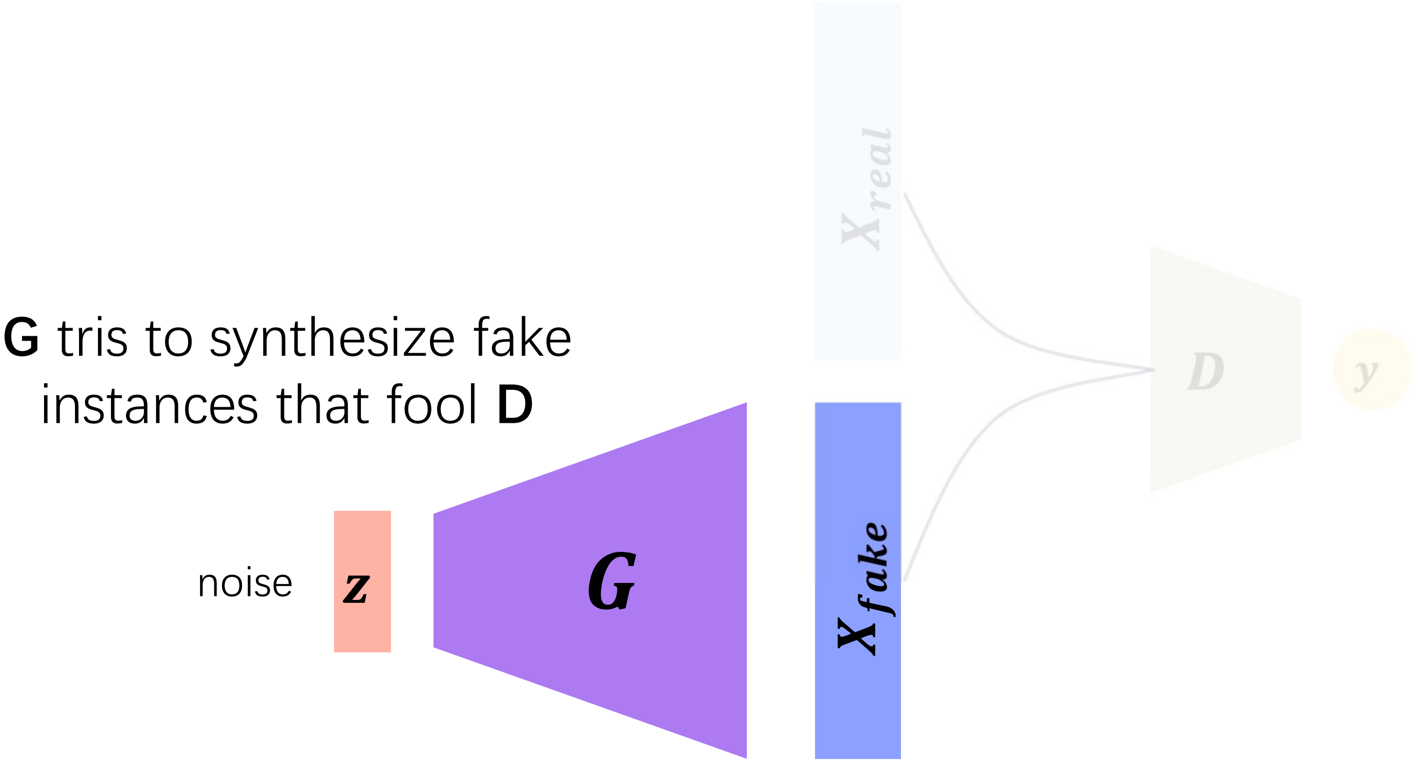

**Training:** adversarial objectives for **D** and **G**
**Global optimum: G** reproduces the true data distribution

# Training GANs



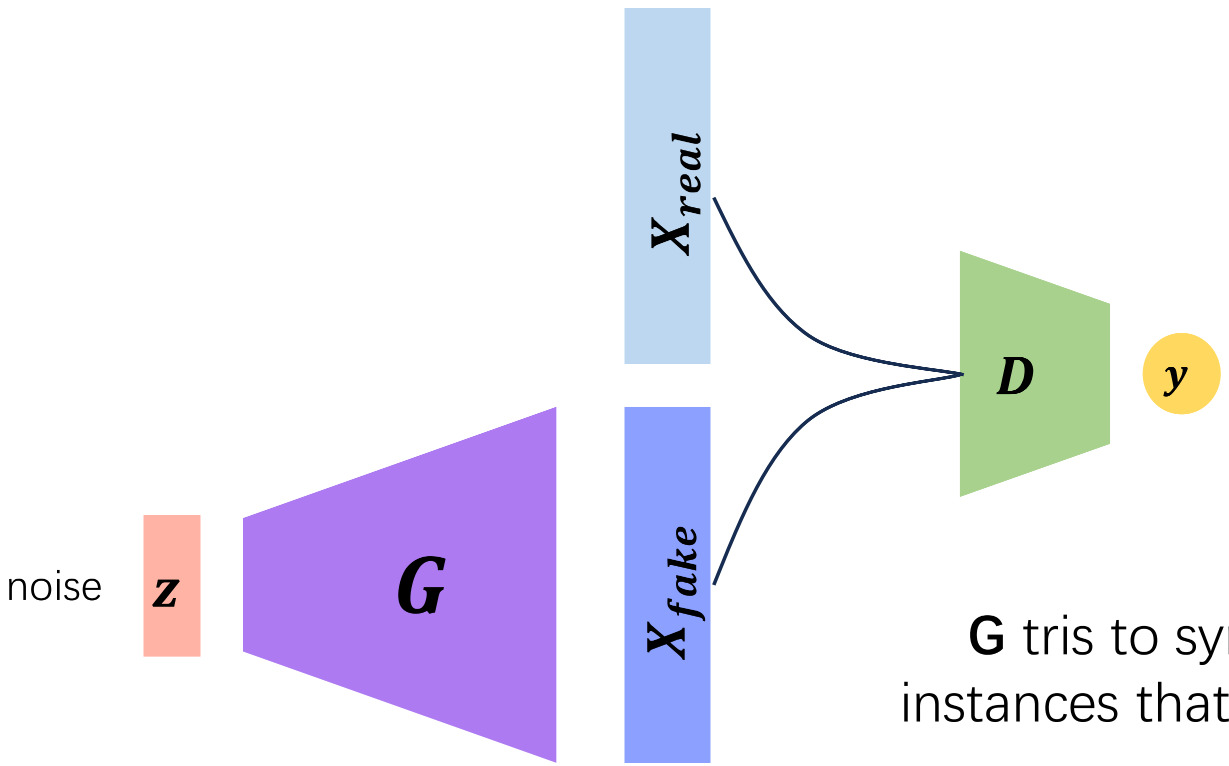

$$\arg\max_{D} E_{z,x}[\underbrace{\log D(x)}_{\text{Real}} + \underbrace{\log(1 - D(G(z)))}_{\text{Fake}}]$$

# Training GANs




$$\arg\min_{G} E_{z,x}[\log D(x) + \log(1 - D(G(z)))]$$

# Training GANs




**G** tris to synthesize fake instances that fool the *best* **D**

$$\arg\min_{G}\max_{D} E_{z,x}[\log D(x) + \log(1 - D(G(z)))]$$

# GANs: Mathematic Formulation

**D:** tris to distinguish between real and fake images
**G:** tris to fool **D** by generating real-looking images

Train jointly in **minimax game**

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ E_{x\sim p_{data}} \log D_{\theta_d}(x) + E_{z\sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right) \right]$$

**Generator objective**

**Discriminator objective**

# GANs: Mathematic Formulation

**D:** tris to distinguish between real and fake images
**G:** tris to fool **D** by generating real-looking images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[ E_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + E_{z \sim p(z)} \log \left( 1 - \underbrace{D_{\theta_d}\left(G_{\theta_g}(z)\right)}_{\text{Discriminator output for generated fake data G(z)}} \right) \right]$$

# GANs: Mathematic Formulation

**D:** tris to distinguish between real and fake images
**G:** tris to fool **D** by generating real-looking images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[ E_{x\sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + E_{z\sim p(z)} \log\left(1 - \underbrace{D_{\theta_d}\left(G_{\theta_g}(z)\right)}_{\text{Discriminator output for generated fake data G(z)}}\right)\right]$$

# GANs: Mathematic Formulation

**D:** tris to distinguish between real and fake images
**G:** tris to fool **D** by generating real-looking images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g}\max_{\theta_d}\left[E_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + E_{z\sim p(z)}\log\left(1-\underbrace{D_{\theta_d}\left(G_{\theta_g}(z)\right)}_{\text{Discriminator output for generated fake data G(z)}}\right)\right]$$

# GANs: Mathematic Formulation

**D:** tris to distinguish between real and fake images
**G:** tris to fool **D** by generating real-looking images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[ E_{x \sim p_{data}} \log D_{\theta_d}(x) + E_{z \sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right) \right]$$

- $\theta_d$ $in$ $\boldsymbol{D}$ wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- $\theta_g$ $in$ $\boldsymbol{G}$ wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

# Training GANs: Two-player Optimization

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ E_{x\sim p_{data}} \log D_{\theta_d}(x) + E_{z\sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)\right]$$

迭代优化:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[ E_{x\sim p_{data}} \log D_{\theta_d}(x) + E_{z\sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)\right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \left[ E_{z\sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)\right]$$

# Training GANs: Two-player Optimization

**for** number of training iterations **do**

**for** $k$ steps **do**

- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

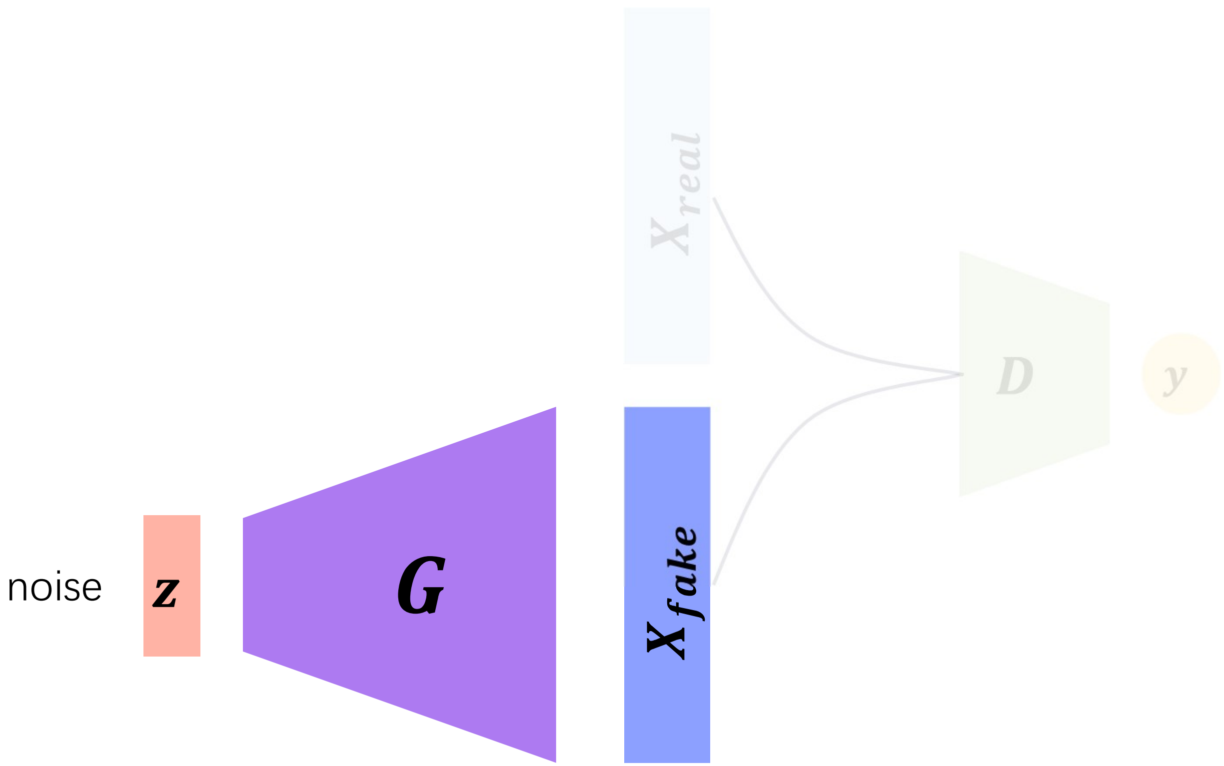
**end for**

- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

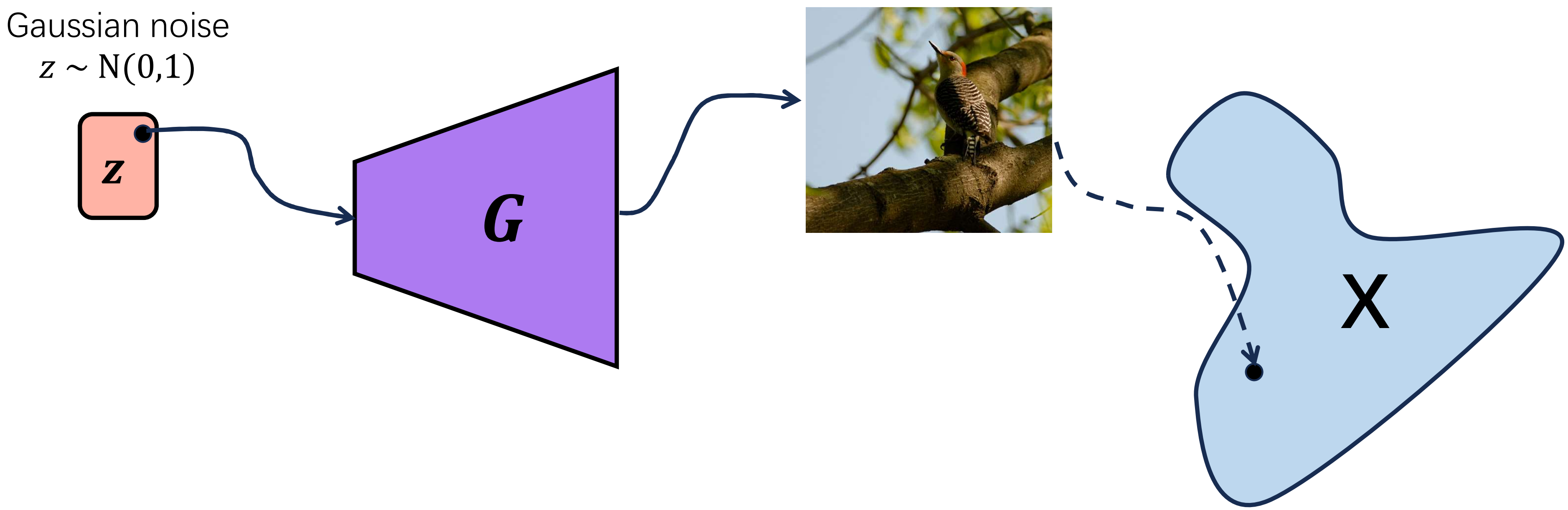
**end for**

# Generating new data with GANs

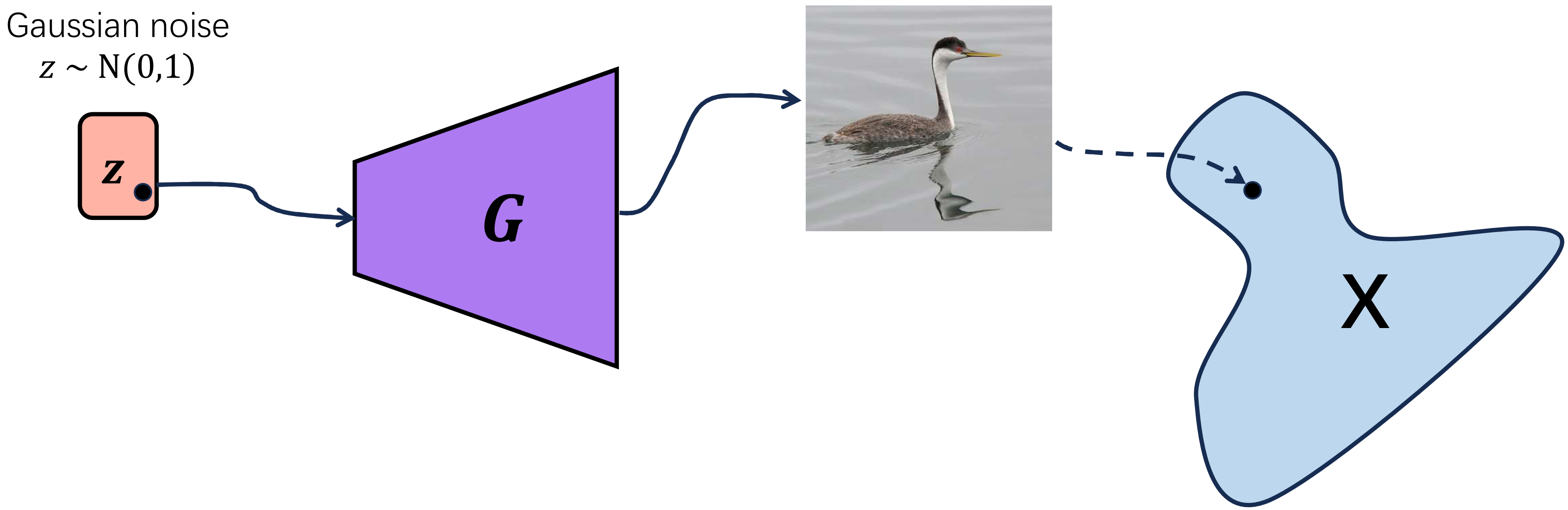



After training, use the generator network only to create **new data** that's never been seen before.

# GANs: Distribution Transformers

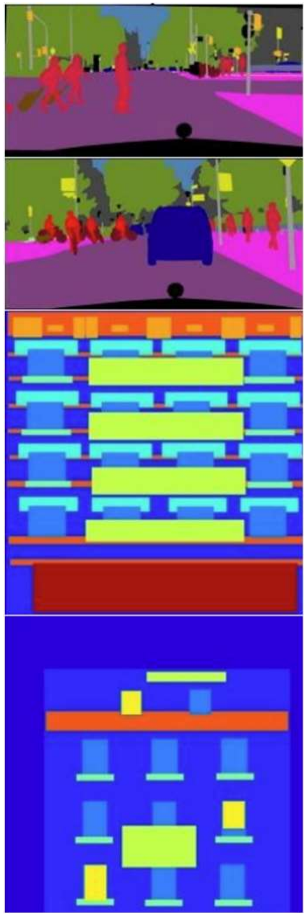


# GANs: Distribution Transformers

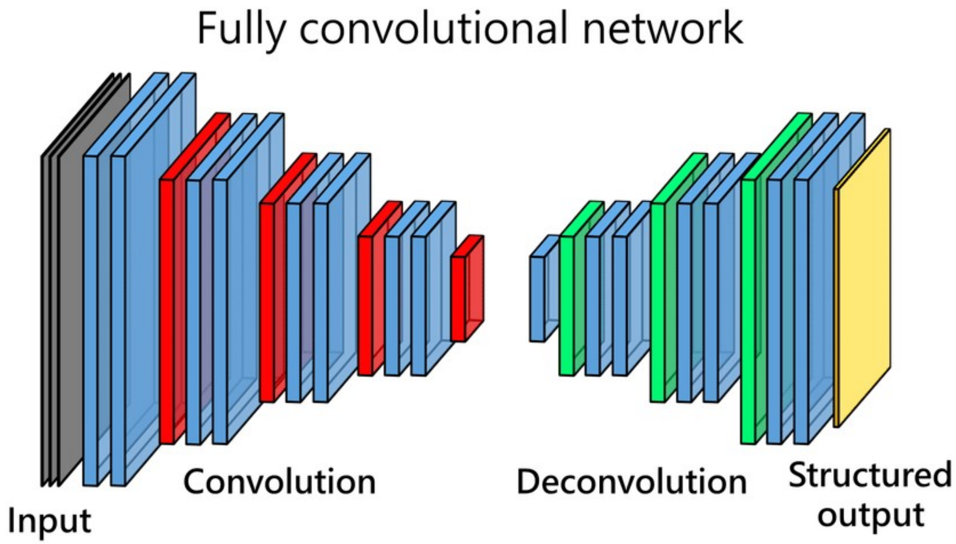


# GANs for better complex data generation

# GANs for better Assignment 2

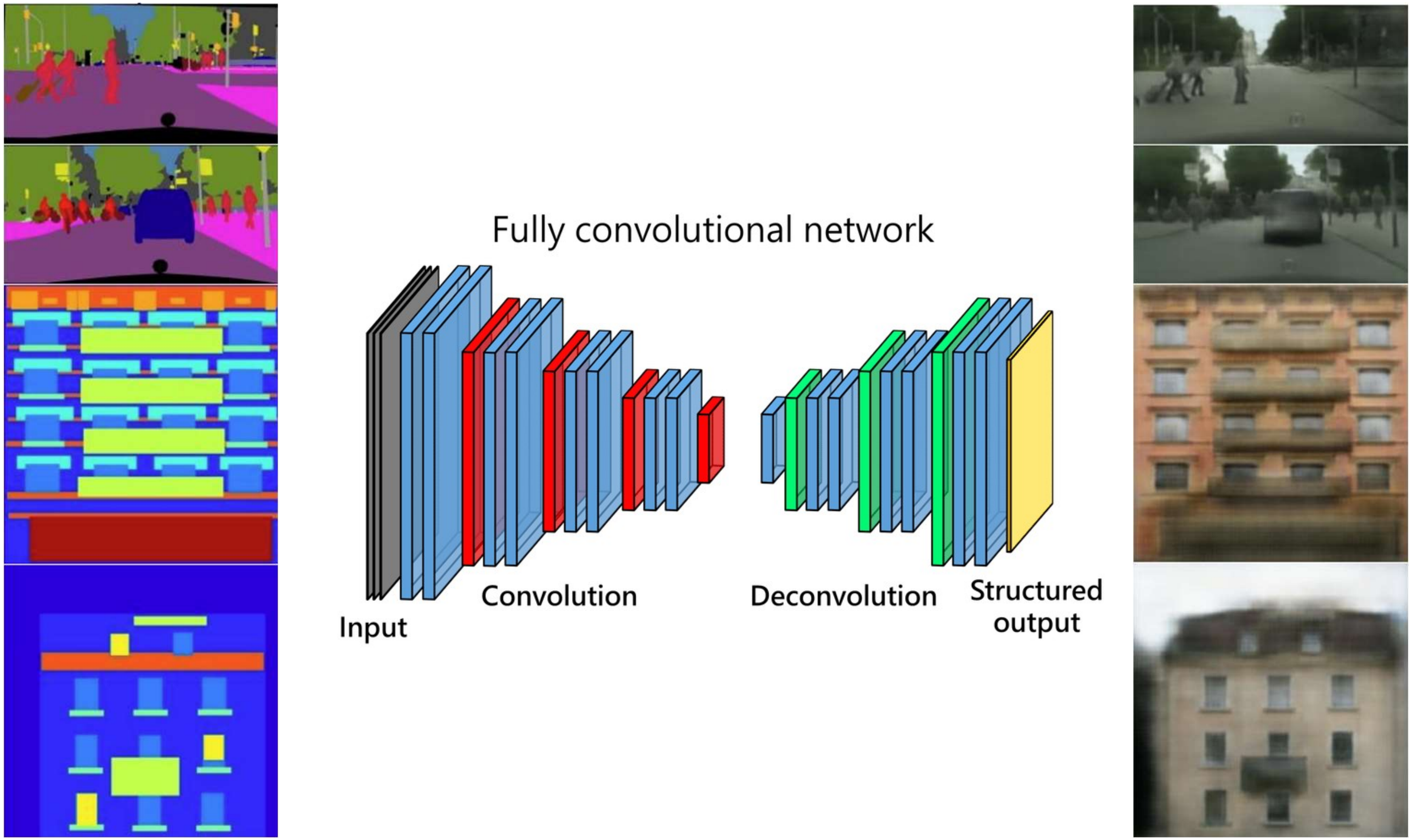


Input




Label
(Ground Truth)

# Expected results if you properly finish hw2



# Why results are not satisfactory?


L2 / L1: Average pixel distance maybe not the proper way to evaluate quality of images

Add a GAN

We want the generated image look like real images

# Pix2Pix with GAN

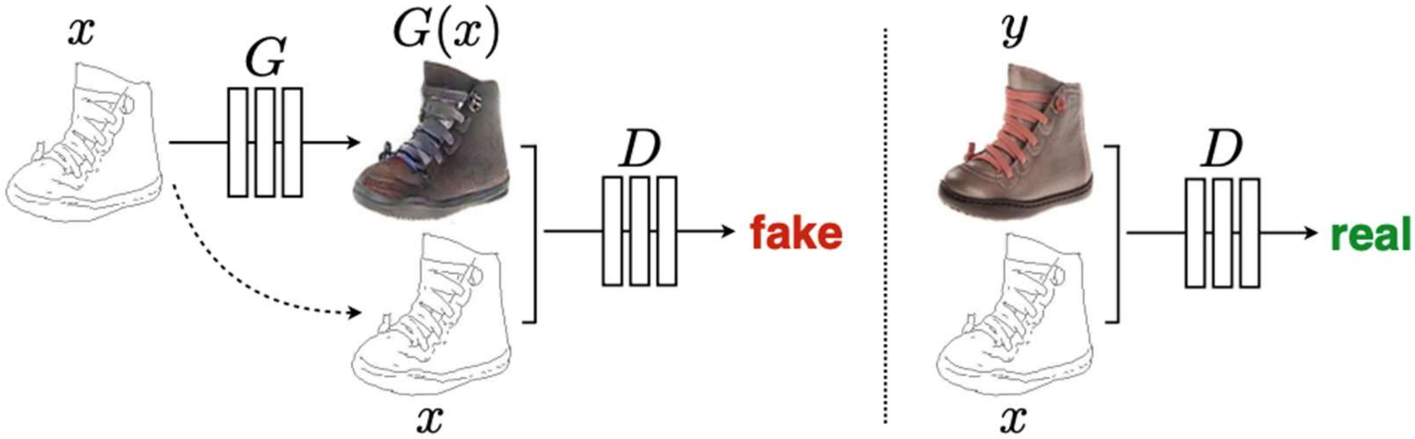
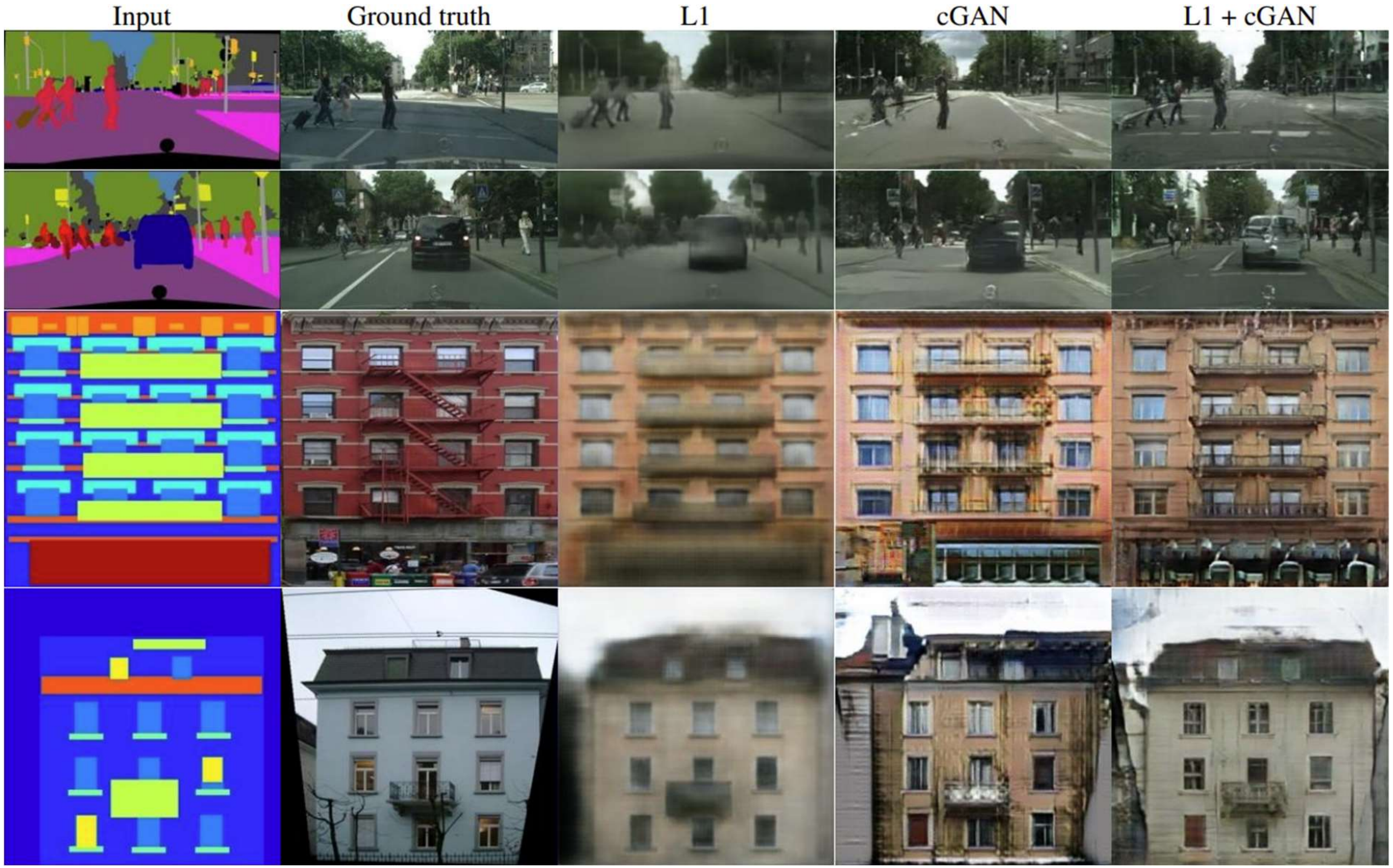


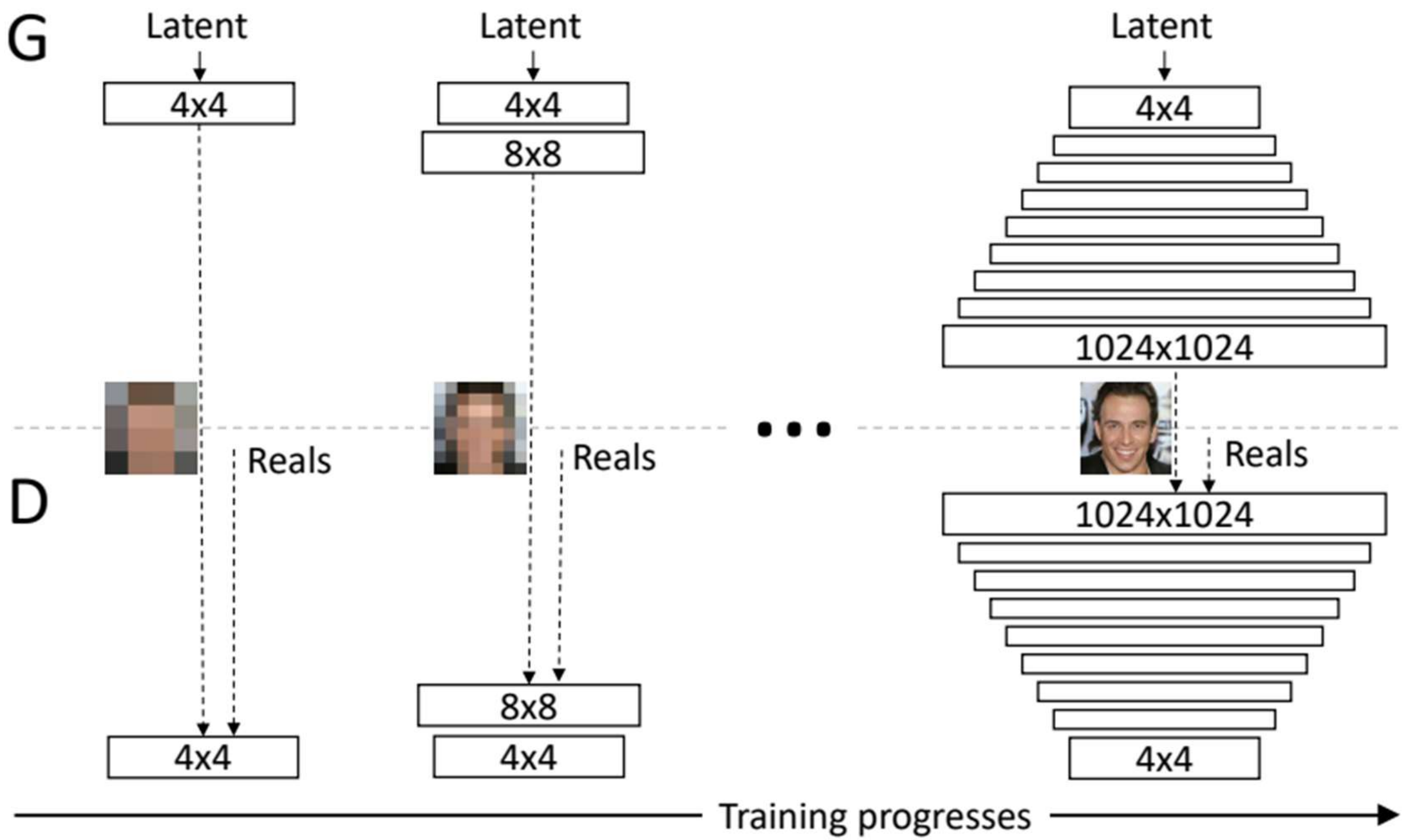

Figure 2: Training a conditional GAN to map edges→photo. The discriminator, $D$, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, $G$, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

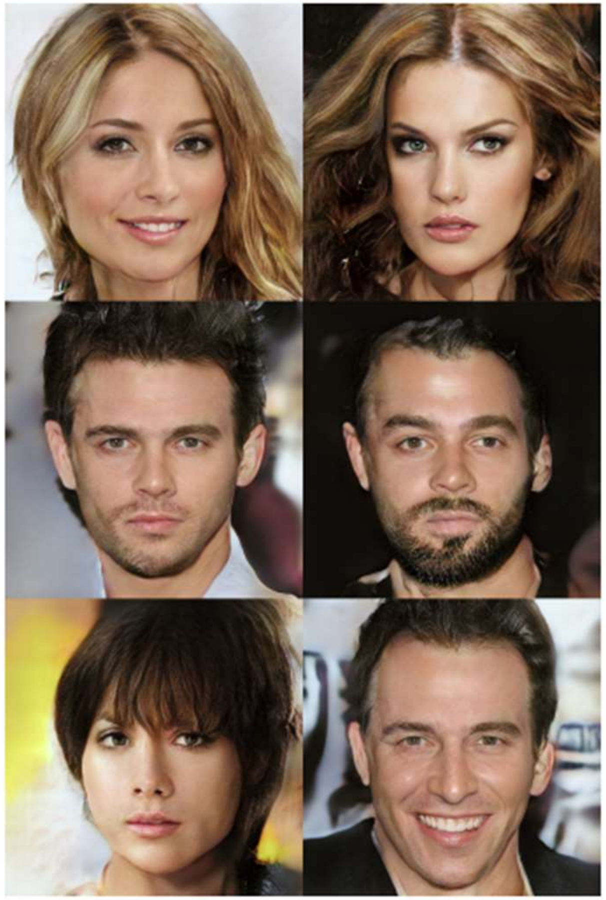
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

# Make results more like real



# GANs for high-resolution Images

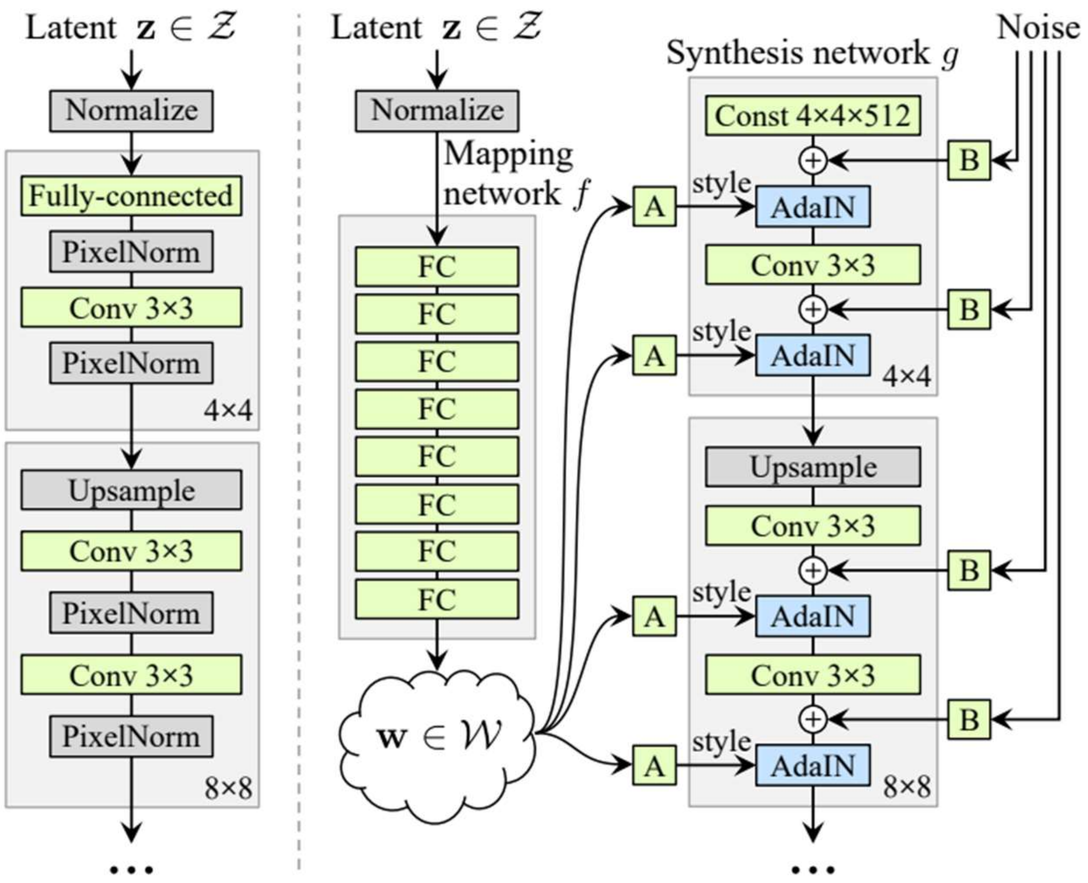


Progressive Growing of GANs for Improved Quality, Stability, and Variation. ICLR 2018.

# Progressive Growing of GANs

# Add more control with StyleGANs




(a) Traditional

(b) Style-based generator

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Style | Content | Ours

# Add more control with StyleGANs

Generative adversarial networks
learn to generate entirely new images
that mimic the appearance of real photos

However, they offer very limited control
over the generated images

# Use GAN as persudo 3D renderer

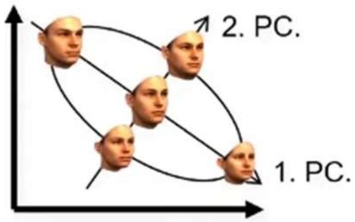
A vector space of 3D shapes and colors of a class of objects

- linear combinations of shapes $\mathbf{S}$ and textures $\mathbf{T}$

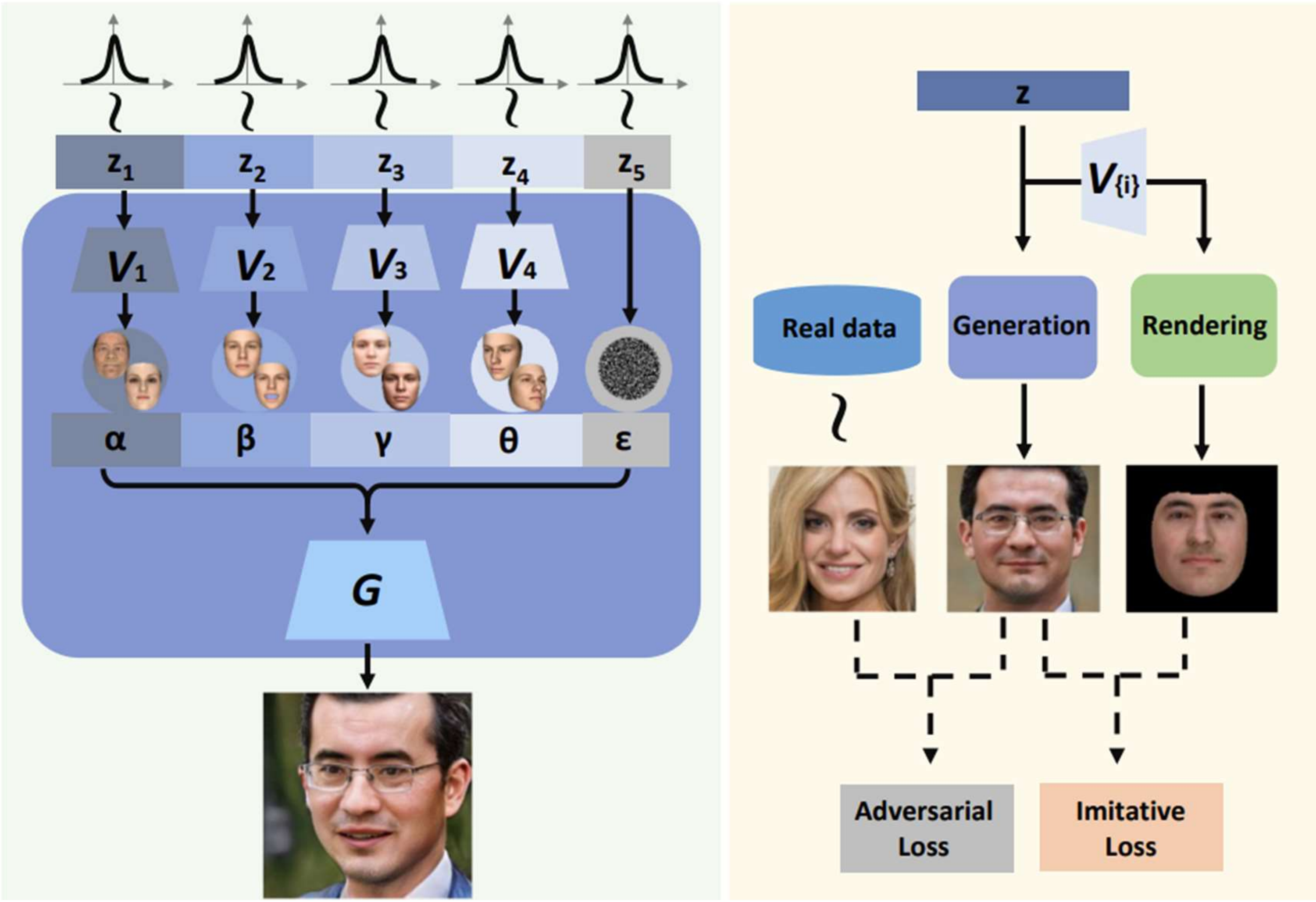
$$\mathbf{S} = \sum_i \alpha_i \boldsymbol{S}_i = \alpha_1 \cdot + \alpha_2 \cdot + \alpha_3 \cdot + \alpha_4 \cdot + \ldots$$

$$\boldsymbol{T} = \sum_i \beta_i \boldsymbol{T}_i = \beta_1 \cdot + \beta_2 \cdot + \beta_3 \cdot + \beta_4 \cdot + \ldots$$

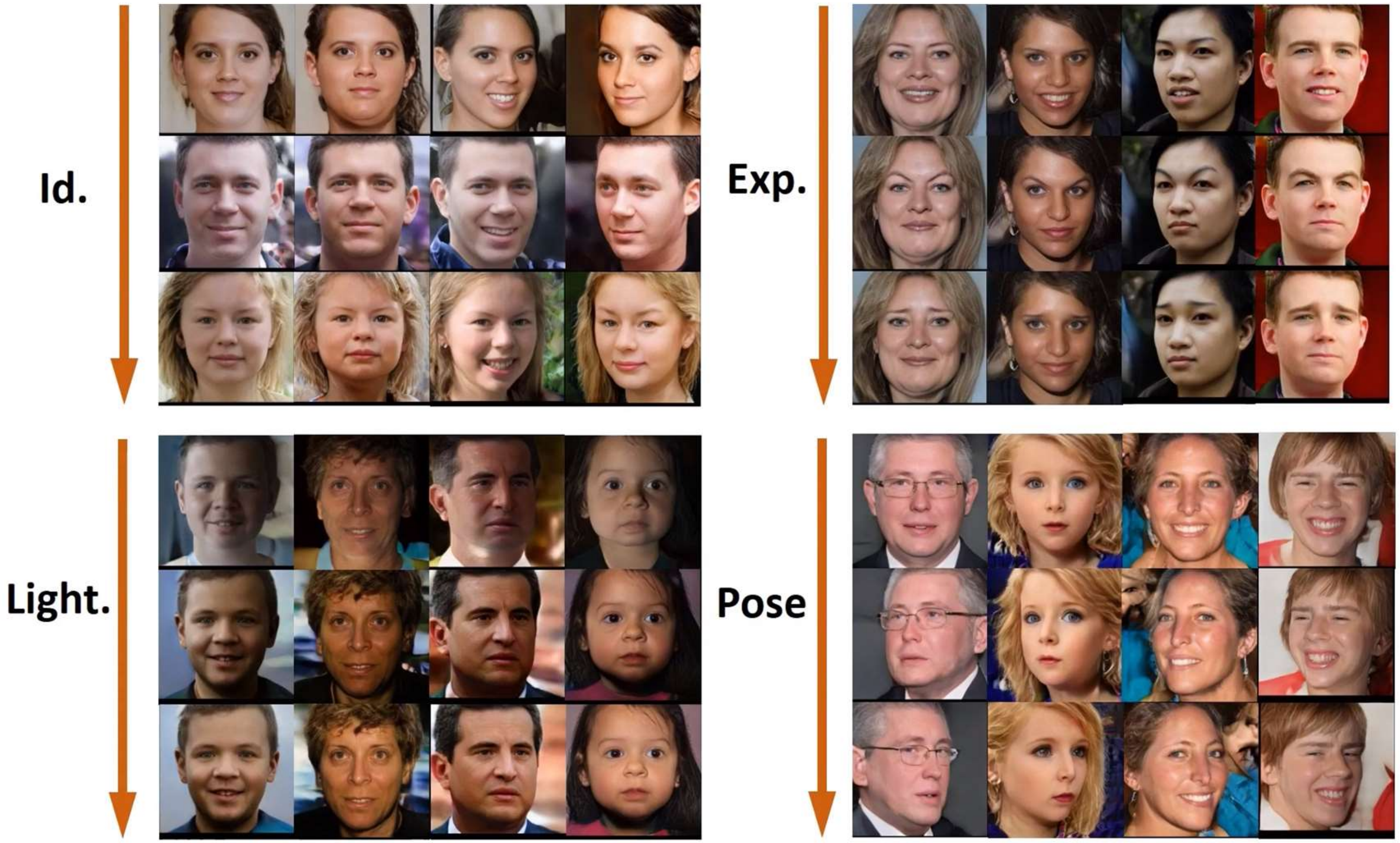
- Often: Principal Component Analysis (PCA)



# Use GAN as persudo 3D renderer



# DiscoFaceGAN



# 3D GAN

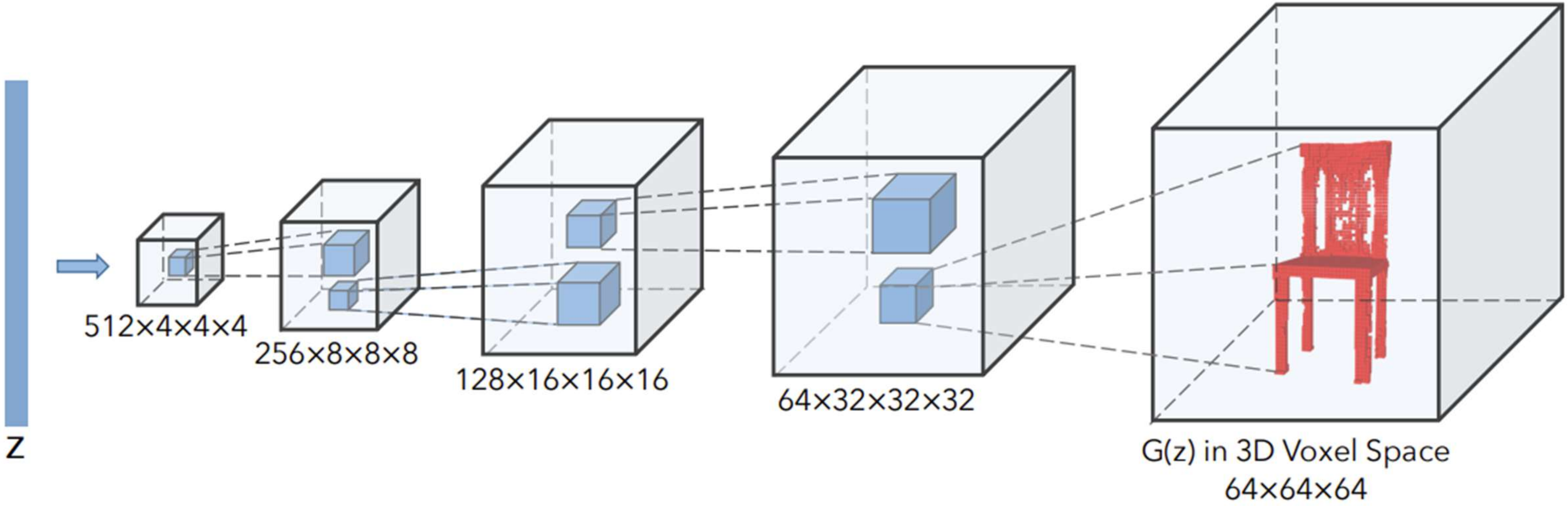



Figure 1: The generator in 3D-GAN. The discriminator mostly mirrors the generator.

**Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. NeurIPS 2016.**

# 3D GAN

## Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling

NIPS 2016





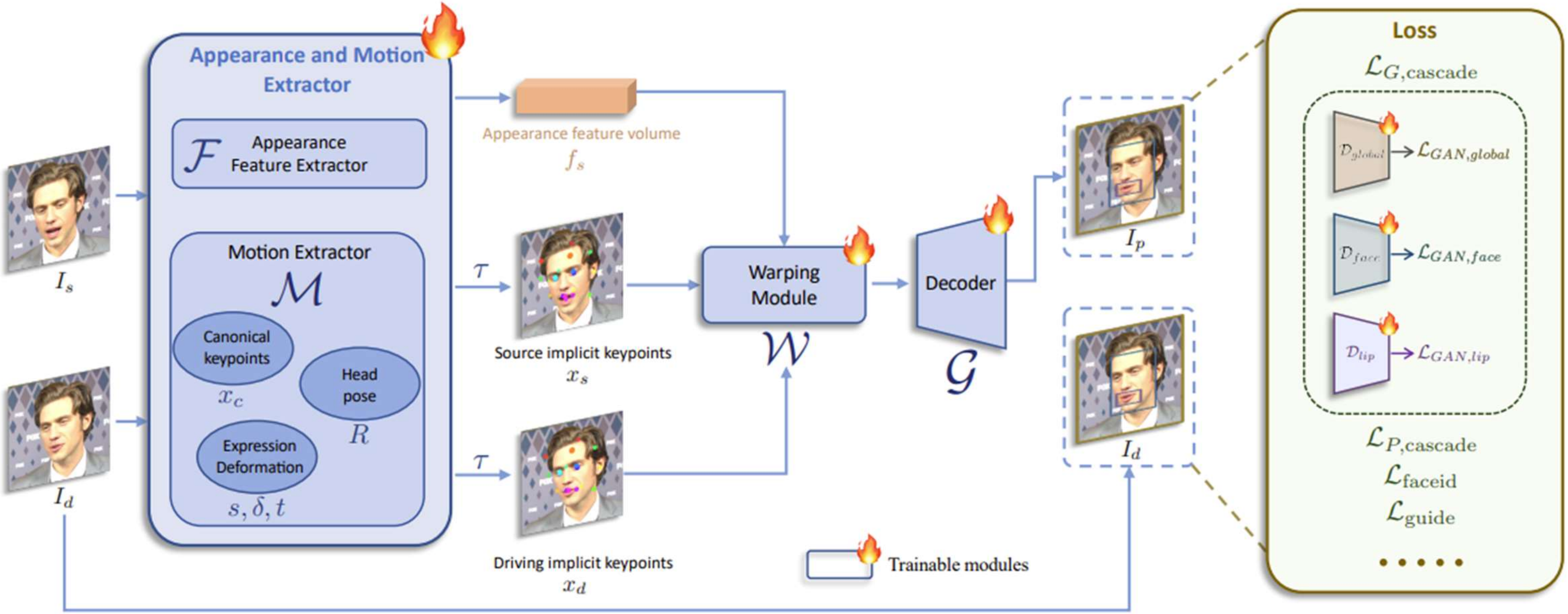
Jiajun Wu*    Chengkai Zhang*    Tianfan Xue    Bill Freeman    Josh Tenenbaum

MIT CSAIL    Google Research

* indicates equal contribution

**Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. NeurIPS 2016.**

# GAN for Animation




LivePortrait: Efficient Portrait Animation with Stitching and Retargeting Control. 2024.

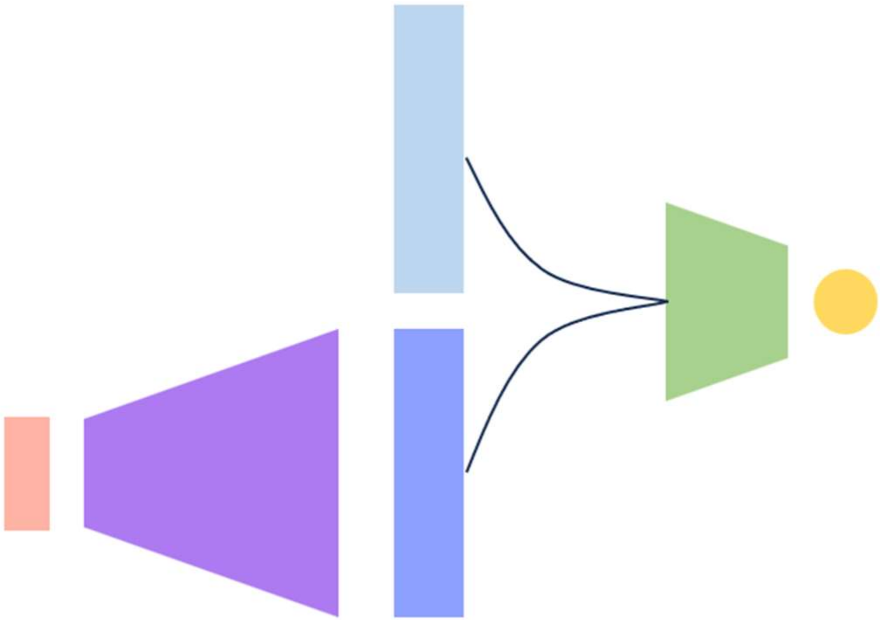
# GAN for Animation

Driving video

Live Portrait Animation From a Still Image

# GAN for Complex Video Generation

# Course Summary



**Generative Adversarial Networks (GANs)**

- Not modeling explicit density
- Directly sample from noise
- Competing **Generator** and **Discriminator** networks
- Many applications for complex generation …

中国科学技术大学
University of Science and Technology of China

# 谢谢观看！