

25  
SOICT

YEARS ANNIVERSARY

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# Bài 12:

## Mô hình sinh dữ liệu

# Nội dung

- Giới thiệu về mô hình sinh
- Mô hình tự mã hóa Autoencoder
- GANs

# Giới thiệu về mạng sinh dữ liệu

# Đâu là mặt thật, đâu là mặt giả?

- <http://www.whichfaceisreal.com/>



# Học giám sát và không giám sát

## Học giám sát

- Dữ liệu:  $(x, y)$

$x$  là dữ liệu,  $y$  là nhãn

- Mục đích: Học hàm số để ánh xạ  $x \rightarrow y$
- Ví dụ: Phân loại, hồi quy, phát hiện đối tượng, phân đoạn ngữ nghĩa, dịch máy...

## Học không giám sát

- Dữ liệu:  $x$

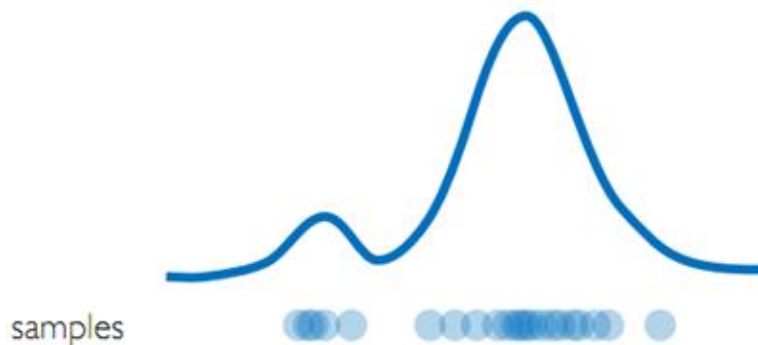
$x$  là dữ liệu, không nhãn!

- Mục đích: Học một cấu trúc ẩn hay một cấu trúc nền tảng nào đó của dữ liệu
- Ví dụ: phân cụm, giảm chiều...

# Mô hình sinh

- Mục đích: Nhận đầu vào một tập mẫu huấn luyện sinh ra từ một phân bố nào đó và học một mô hình để có thể biểu diễn lại phân bố đó

Density Estimation



Sample Generation



Input samples

Generated samples

Training data  $\sim P_{data}(x)$

Generated  $\sim P_{model}(x)$

- Làm sao để học  $p_{model}(x)$  tương tự với  $p_{data}(x)$ ?



# Tại sao cần mô hình sinh?

- Vì nó có thể khám phá ra các thông tin ẩn nền tảng trong dữ liệu



Homogeneous skin color, pose

VS



Diverse skin color, pose, illumination

- Sử dụng phân bố ẩn học được để sinh ra dữ liệu đa dạng và cân bằng hơn (debias)



# Tại sao cần mô hình sinh?

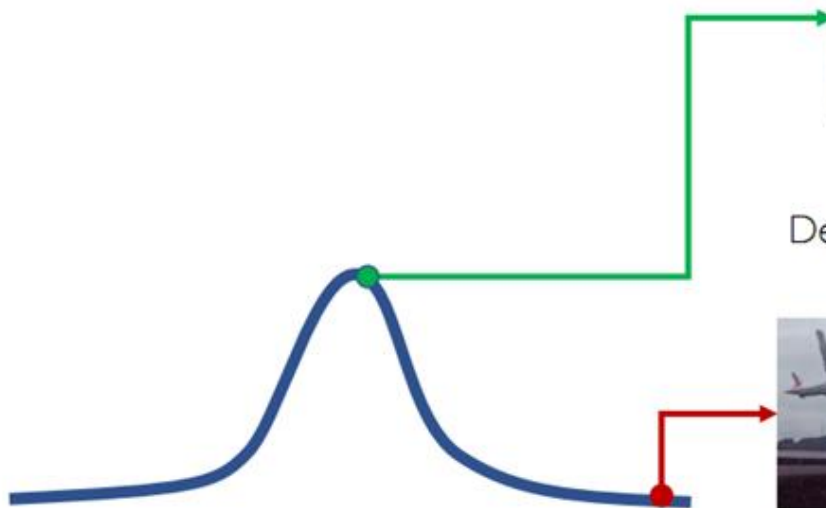
- Phát hiện ngoại lệ (outlier): Làm sao để phát hiện một sự kiện mới học hiếm xảy ra?
- Sử dụng mô hình sinh để học phân bố dữ liệu, từ đó xác định ngoại lệ dựa trên phân bố học được.

95% of Driving Data:

(1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training



Edge Cases



Harsh Weather

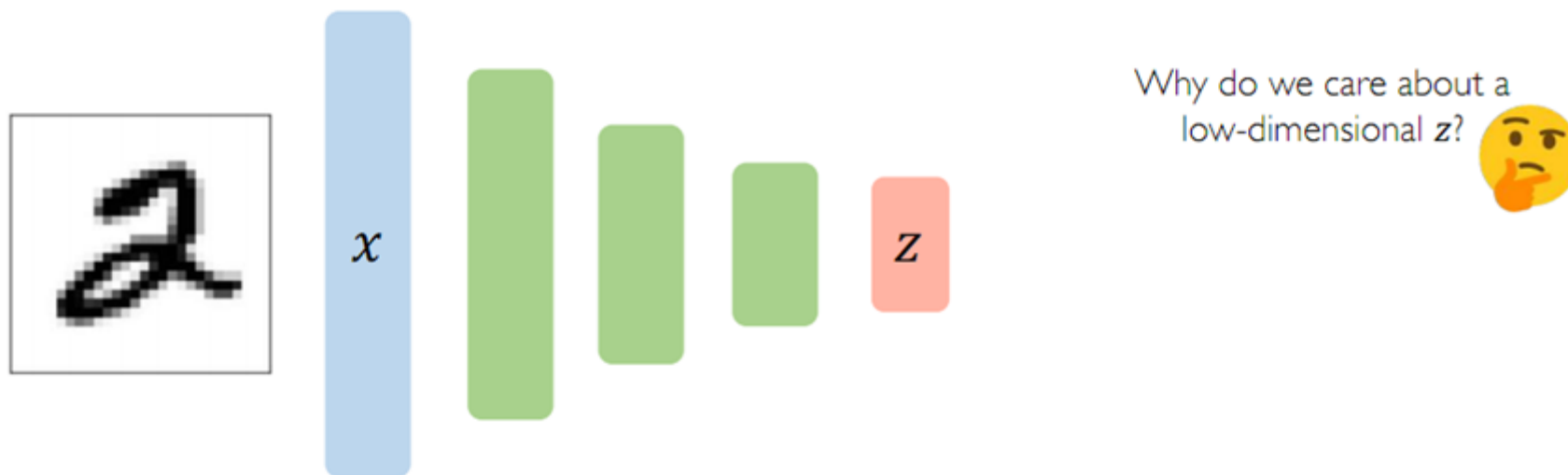


Pedestrians

# Mô hình tự mã hóa Autoencoder

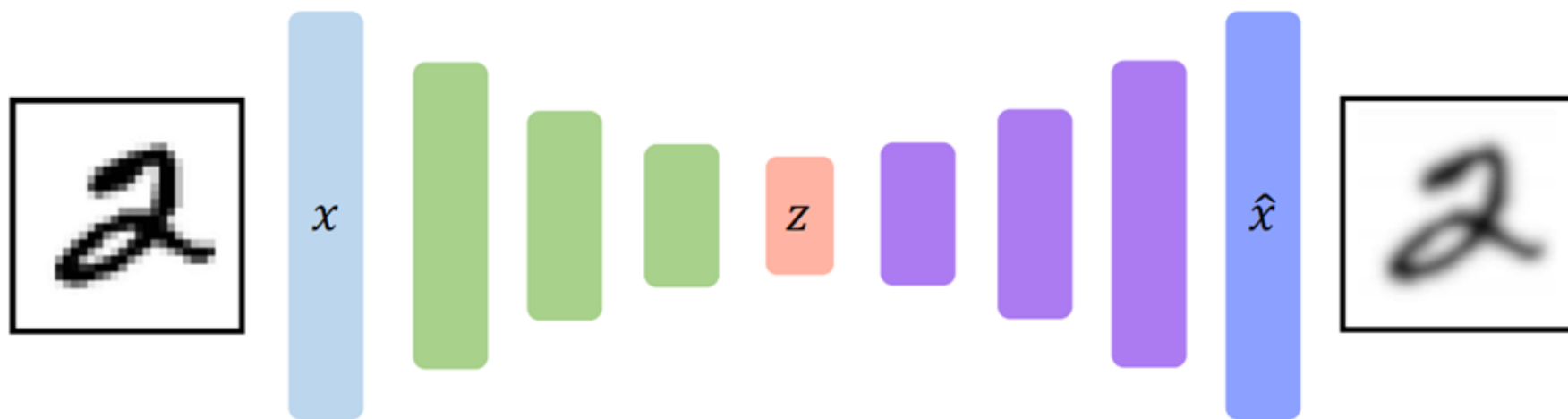
# Autoencoder

- Là mô hình không giám sát cho phép học biểu diễn đặc trưng với số chiều nhỏ hơn từ tập huấn luyện không có nhãn
- “Encoder” học ánh xạ từ dữ liệu  $x$  vào không gian ẩn  $z$  có số chiều thấp hơn



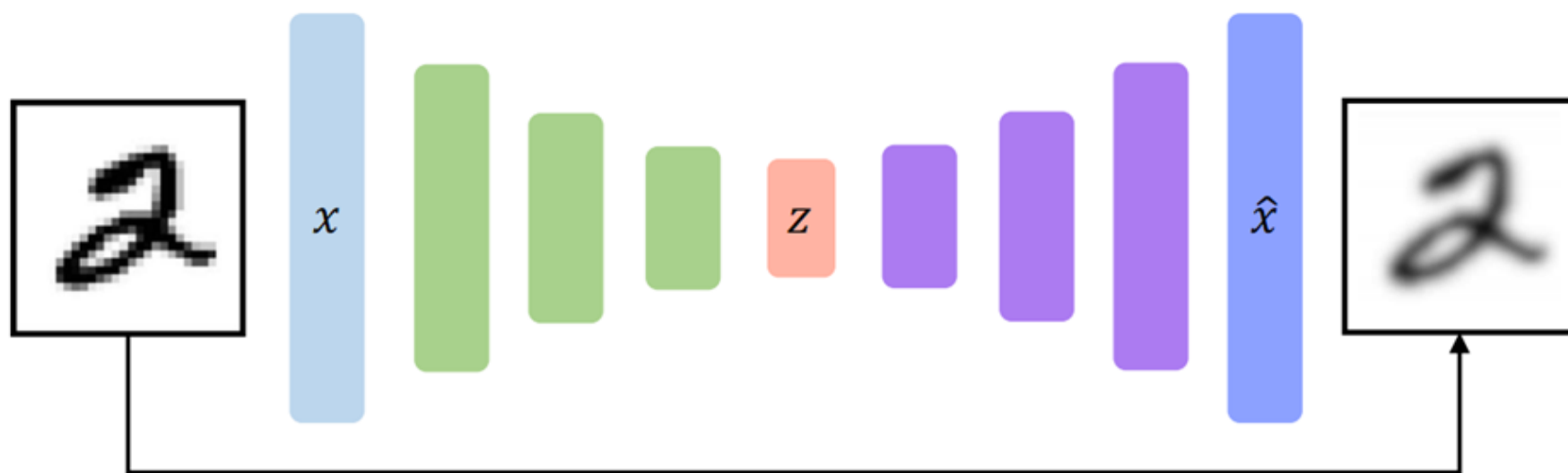
# Autoencoder

- Làm sao để học không gian ẩn?
- Huấn luyện mô hình sử dụng đặc trưng ẩn  $z$  để khôi phục lại dữ liệu gốc ban đầu
- “Decoder” ánh xạ đặc trưng ẩn  $z$  ngược trở lại để khôi phục thông tin dữ liệu đầu vào  $\hat{x}$



# Autoencoder

- Làm sao để học không gian ẩn?
- Huấn luyện mô hình sử dụng đặc trưng ẩn  $z$  để khôi phục lại dữ liệu gốc ban đầu
- Hàm mục tiêu không cần nhãn!



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

# Số chiều không gian ảnh hưởng chất lượng khôi phục dữ liệu

- Autoencoder là một kiểu nén dữ liệu.
- Số chiều không gian ảnh càng nhỏ càng tạo ra nút thắt cổ chai (bottleneck) lớn khi huấn luyện

2D latent space



5D latent space



Ground Truth





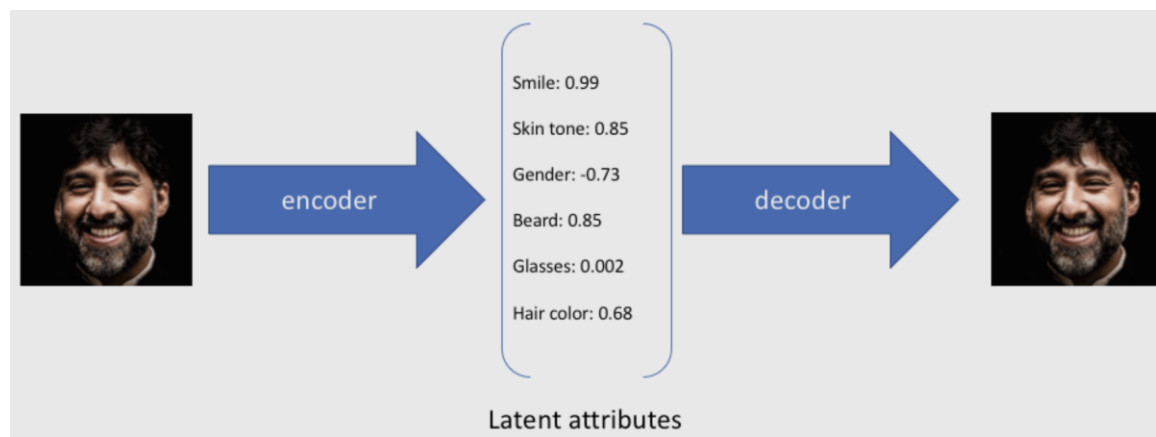
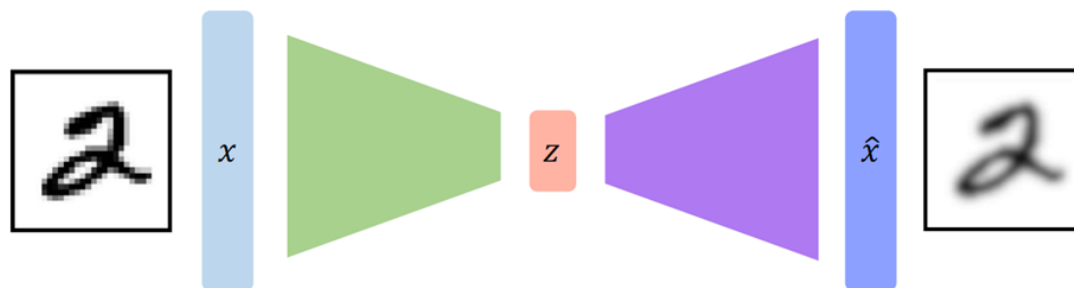
# Autoencoders để học biểu diễn

- Các lớp ẩn thất cổ chai ép mạng học biểu diễn ẩn nén thông tin dữ liệu vào
- Hàm mục tiêu tái tạo ép biểu diễn ẩn phải mã hóa được càng nhiều thông tin từ dữ liệu vào càng tốt
- **Autoencoding** = **Automatically encoding** data

# Variational Autoencoders (VAEs)

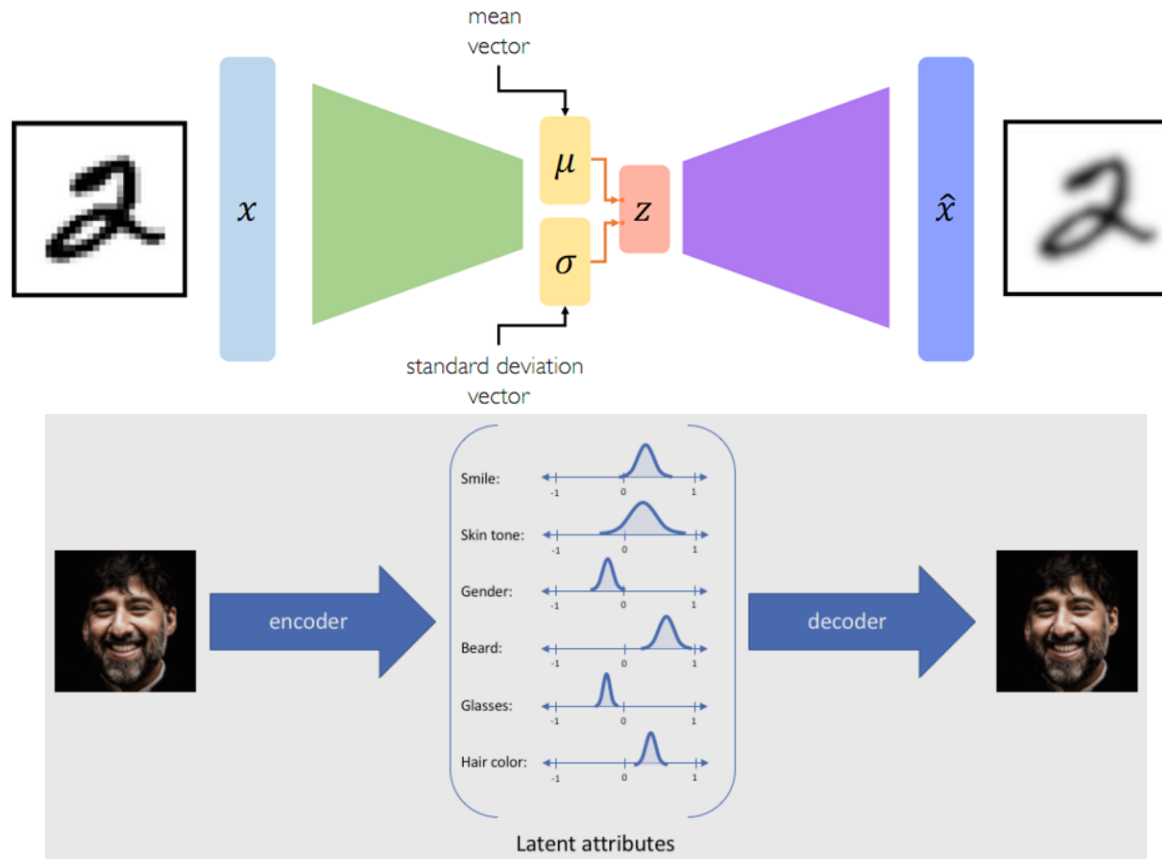
# Xem xét lại Autoencoders

- Mỗi chiều trong vector mã hoá (encoding vector) chứa giá trị đơn, là đại diện cho một thuộc tính ẩn từ dữ liệu đầu vào



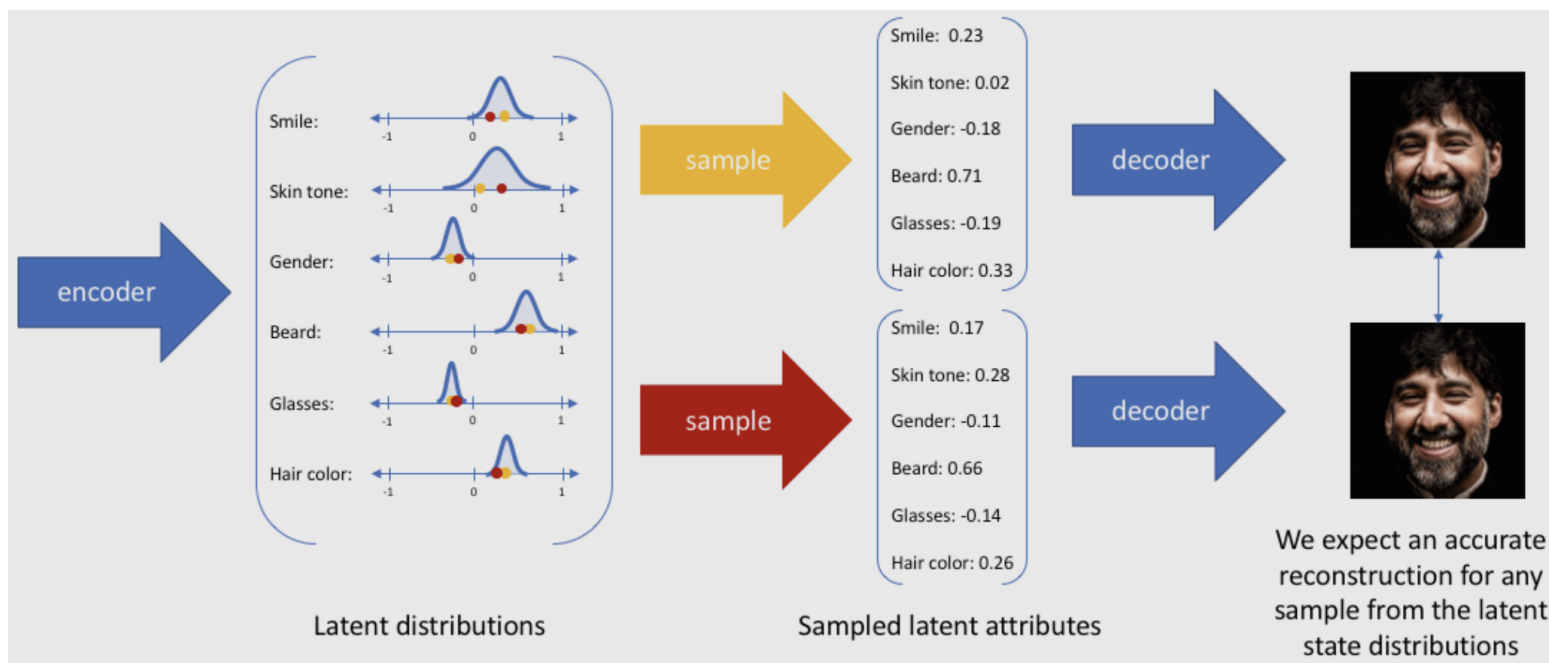
# Variational Autoencoders

- Mỗi thuộc tính ẩn là một phân bố xác suất

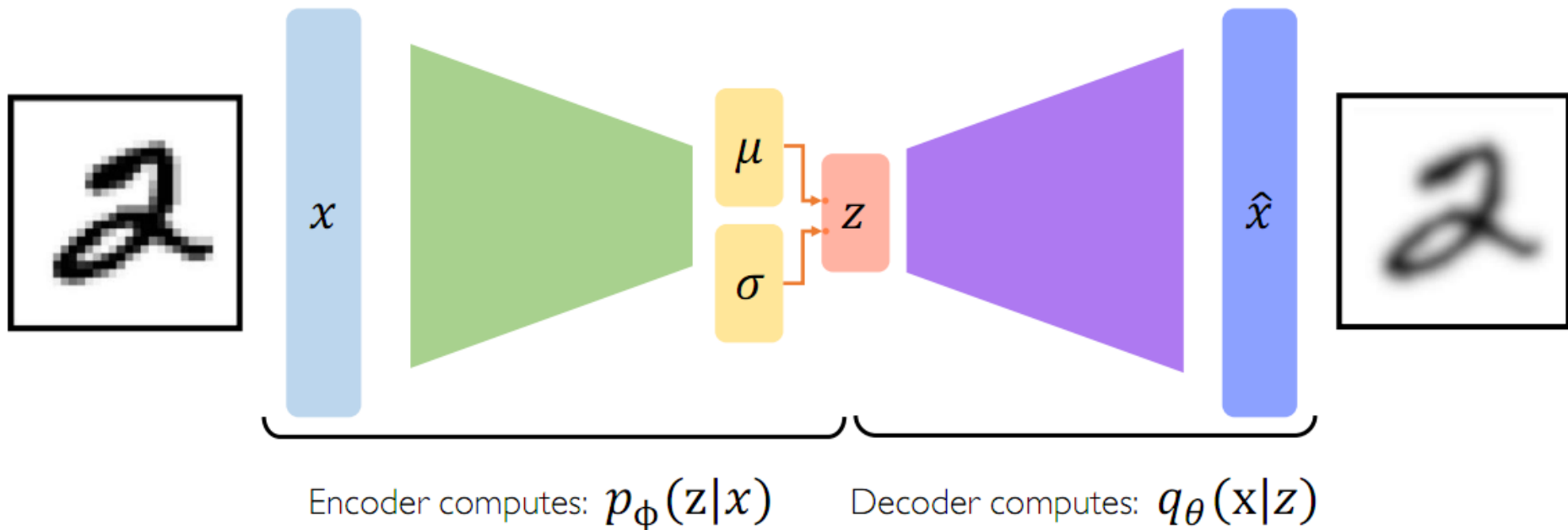


# Decoder trong VAEs

- Khi giải mã từ trạng thái ẩn, lấy mẫu ngẫu nhiên từ phân phối của mỗi thuộc tính ẩn để sinh ra vector đầu vào cho bộ giải mã

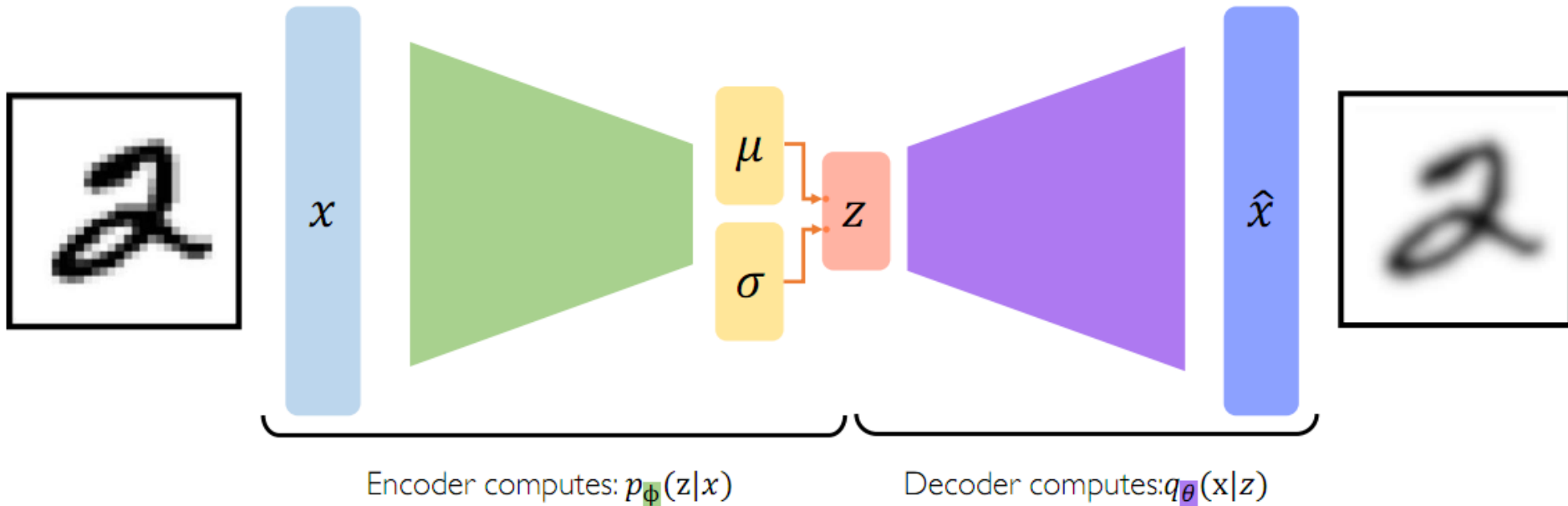


# Tối ưu VAE



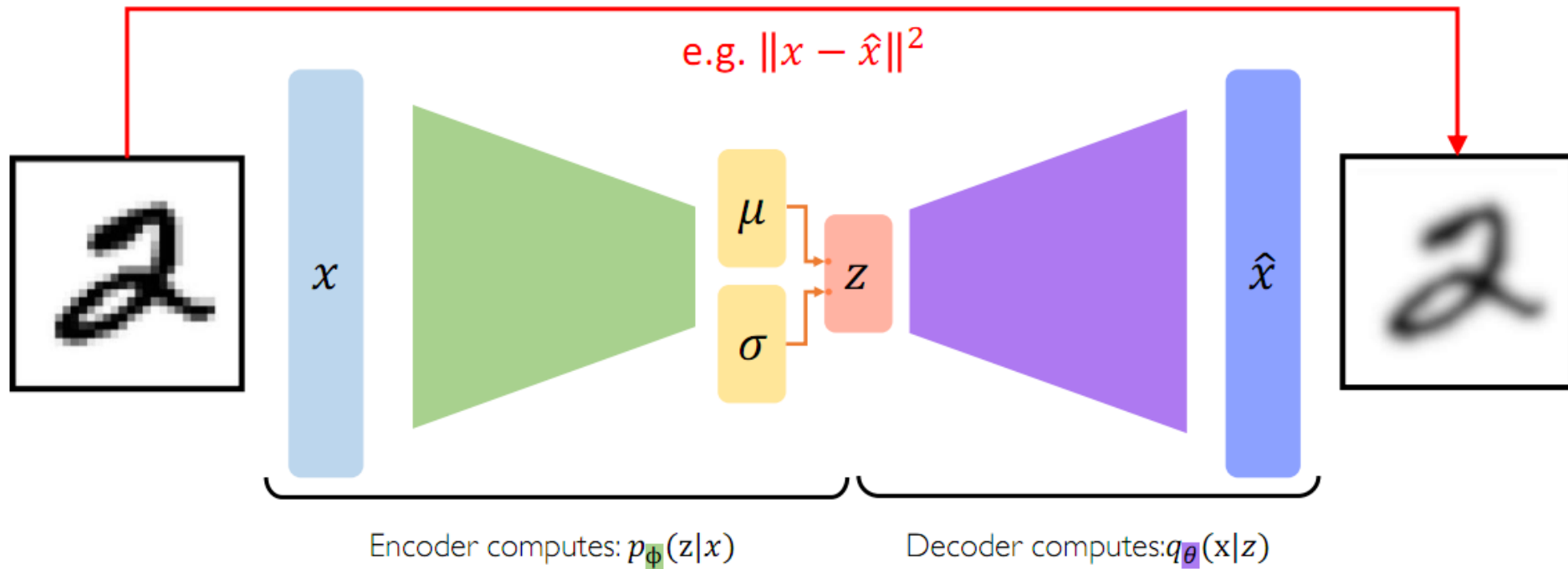


# Tối ưu VAE



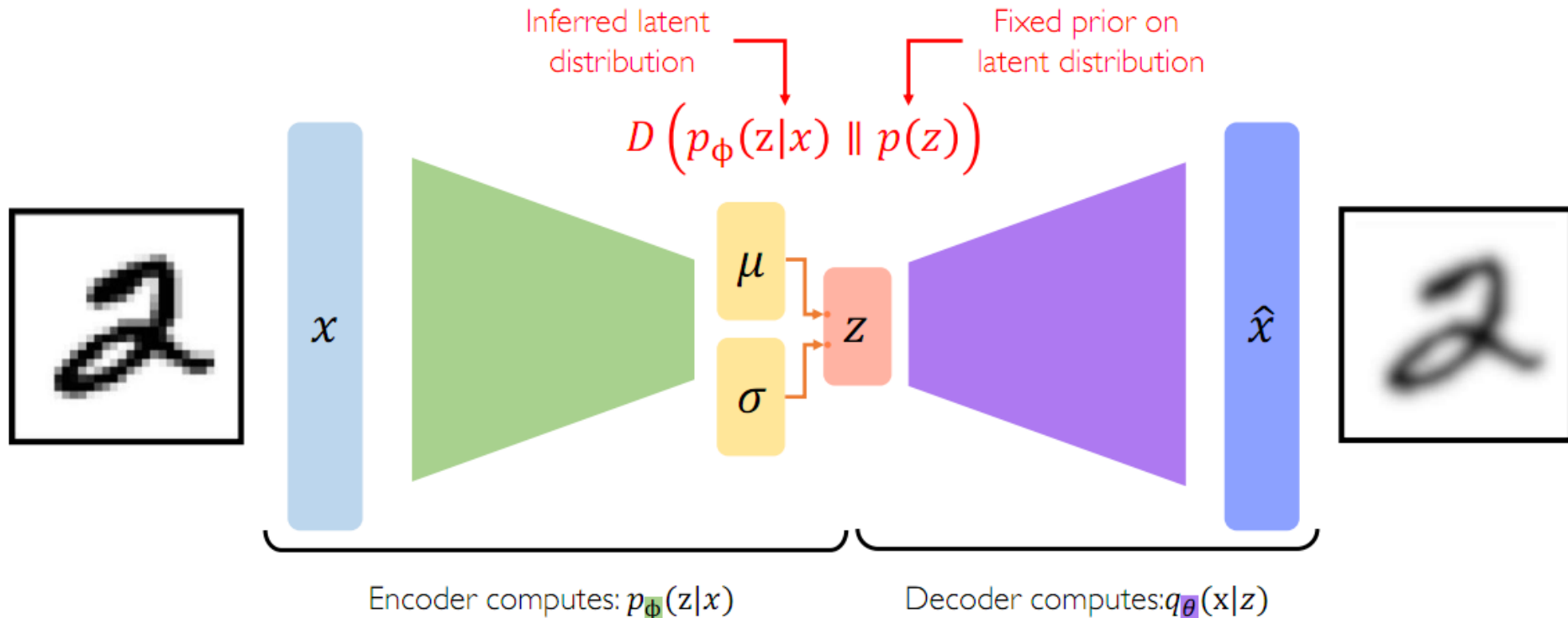
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

# Reconstruction loss



$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

# Regularization term (KL)

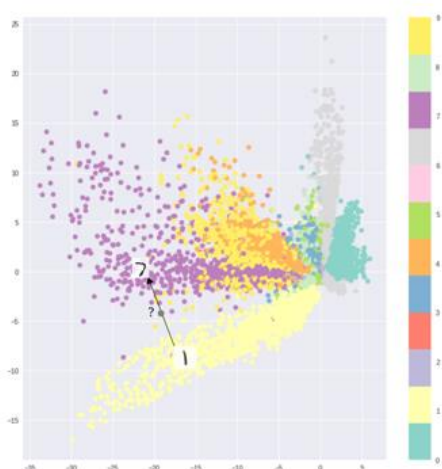


$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

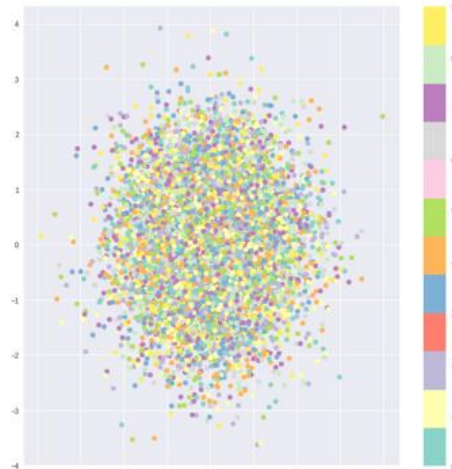
# Biểu diễn không gian ẩn

- Chỉ sử dụng reconstruction loss
  - phân phối không đều, không học được biểu diễn mịn của trạng thái ẩn
- Chỉ sử dụng KL
  - Không học được biểu diễn của dữ liệu gốc, không học chỉ ghi nhớ dữ liệu đầu vào

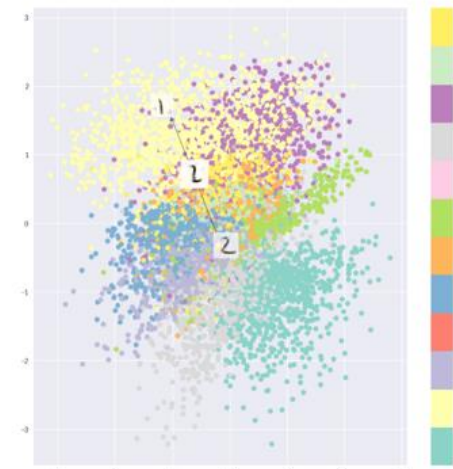
Only reconstruction loss



Only KL divergence

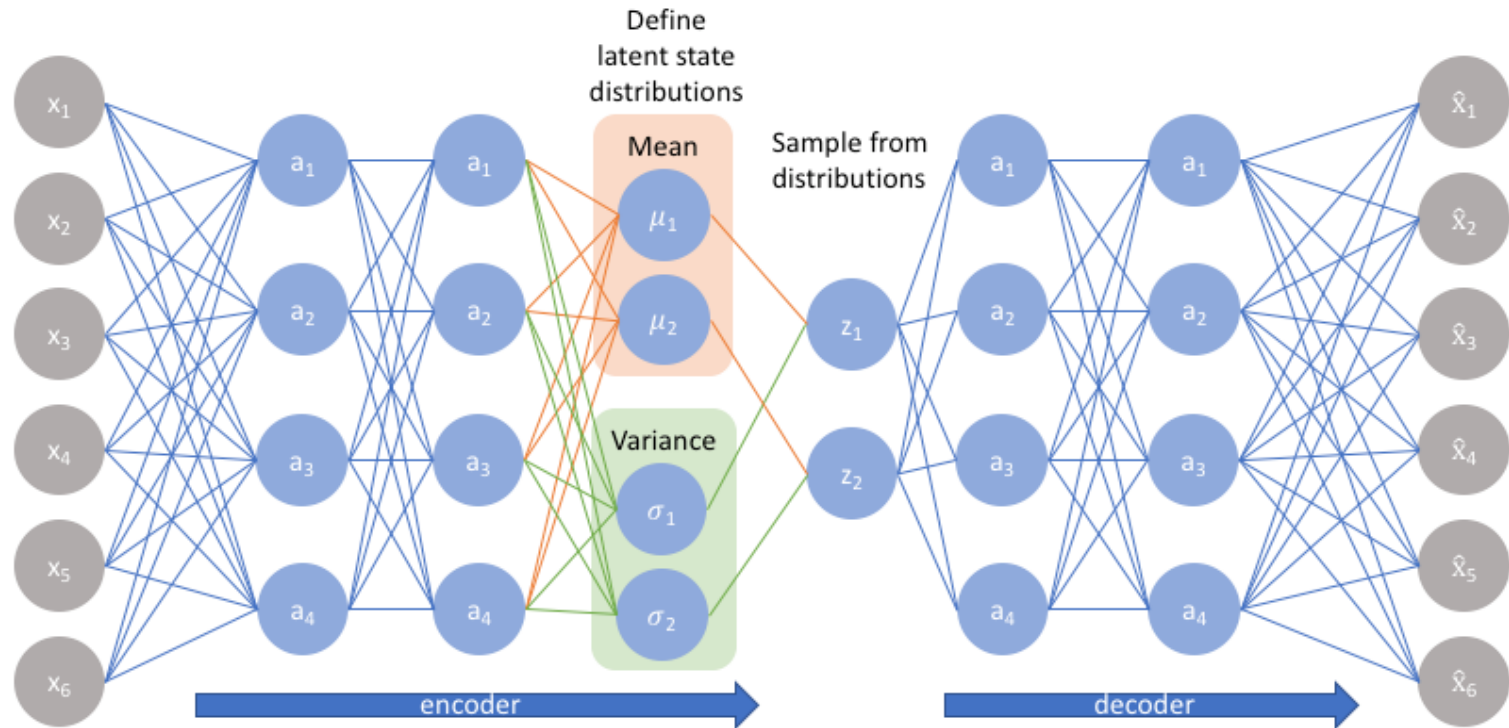


Combination



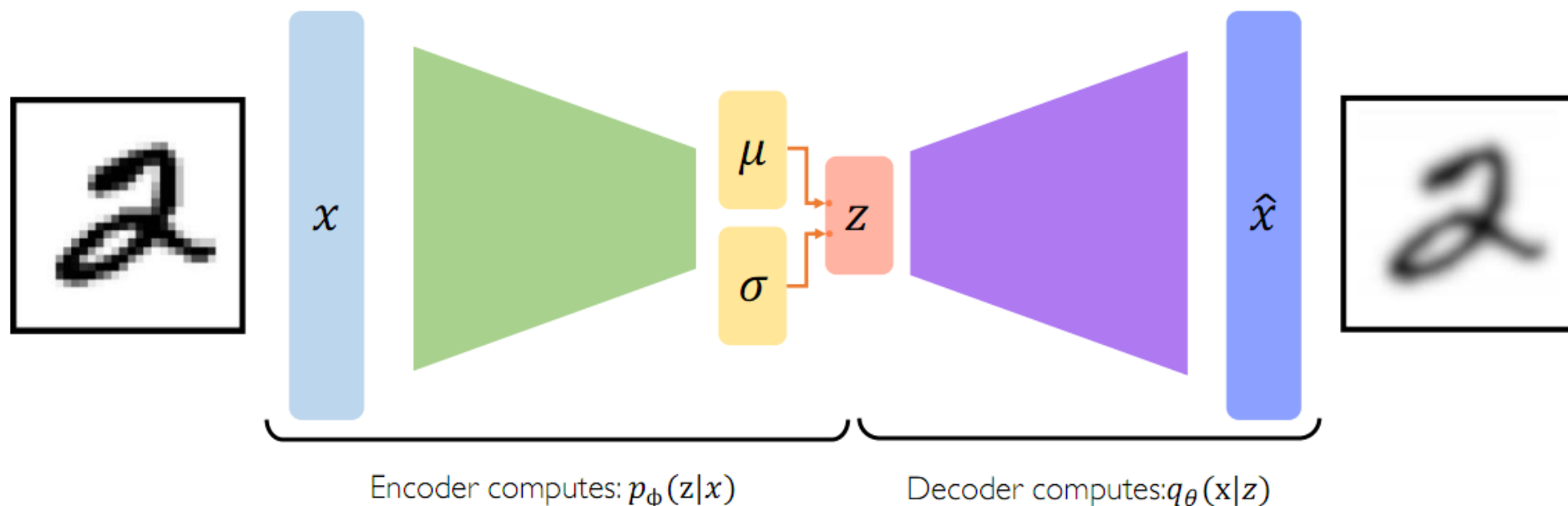
# Cài đặt VAEs

- Với giả định phân phối trạng thái ẩn là phân phối Gaussian
- Đầu ra của Encoder là hai vector mô tả trung bình và phương sai của phân bố trạng thái ẩn
- Decoder lấy mẫu ngẫu nhiên từ trạng thái ẩn và giải mã



# Đồ thị tính toán của VAE

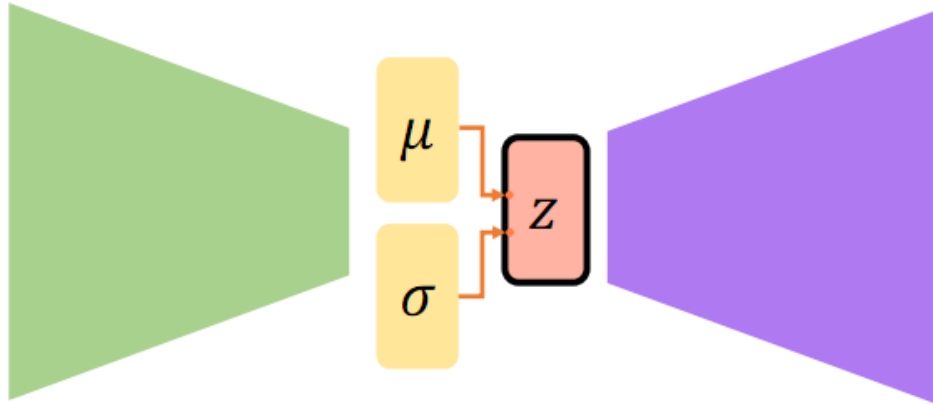
- Vấn đề: Không thể lan truyền ngược qua lớp lấy mẫu!



$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$



# Trick: tái tham số hóa lớp lấy mẫu



Key Idea:

$$- - z \sim \mathcal{N}(\mu, \sigma^2) - -$$

Consider the sampled latent vector as a sum of

- a fixed  $\mu$  vector,
- and fixed  $\sigma$  vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

where  $\varepsilon \sim \mathcal{N}(0,1)$

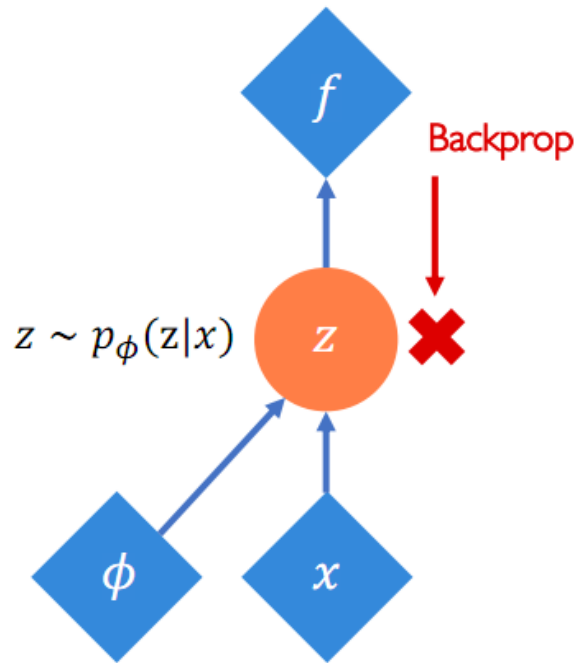
# Tái tham số hóa lớp lấy mẫu



Deterministic node



Stochastic node



Original form

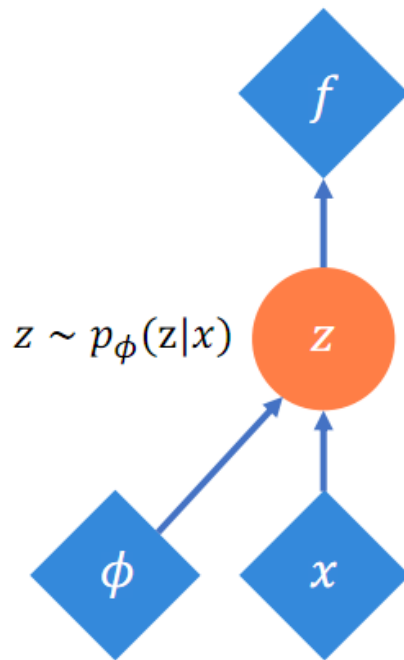
# Tái tham số hóa lớp lấy mẫu



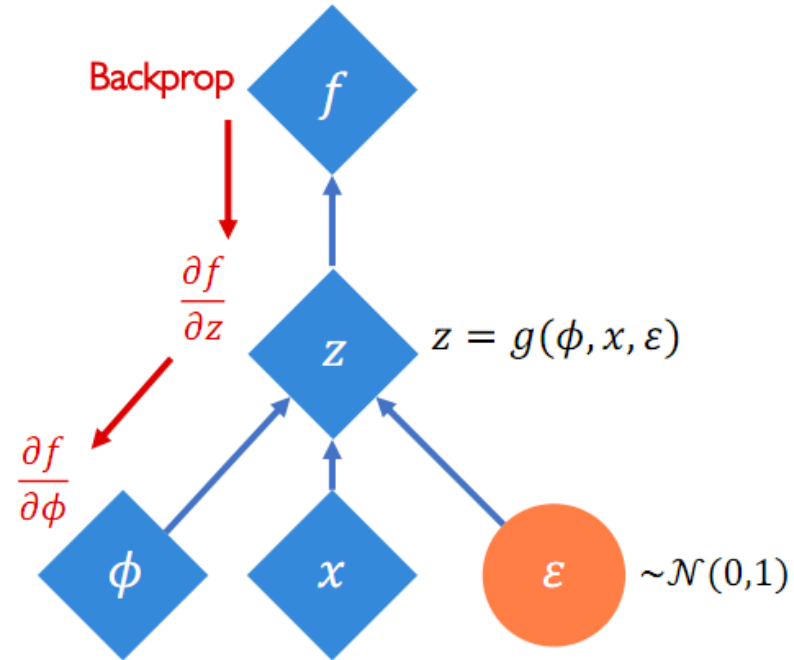
Deterministic node



Stochastic node



Original form



Reparametrized form

# VAEs: Biến đổi không gian ẩn

- Tăng hoặc giảm từ từ một biến ẩn, giữ các biến khác cố định
- Mỗi chiều của  $z$  mã hóa các đặc trưng ẩn có ý nghĩa khác nhau

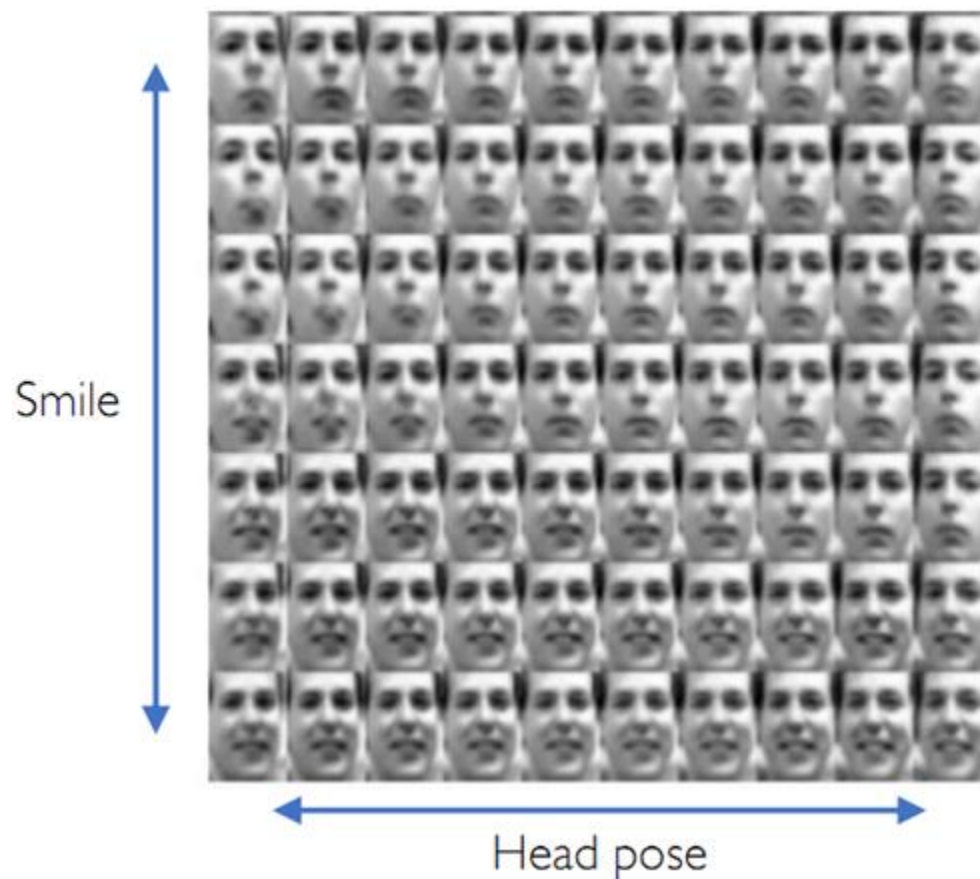


## Head pose

# VAEs: Biến đổi không gian ẩn

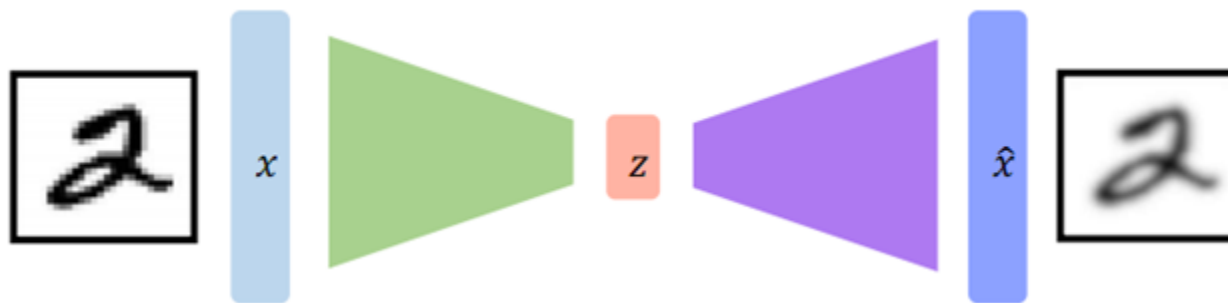
- Lý tưởng là các biến ẩn hoàn toàn độc lập với nhau (uncorrelated)
- Có thể thêm ràng buộc dạng ma trận chéo để ép các biến ẩn độc lập với nhau (hệ số tương quan giữa các biến ẩn khác nhau xấp xỉ 0)

## Disentanglement



# VAEs: Tổng kết

1. Hàm mục tiêu tái tạo cho phép huấn luyện không cần nhãn (không giám sát)
2. Sử dụng kỹ thuật tái tham số hóa để cho phép huấn luyện end-to-end
3. Diễn giải các biến ẩn bằng cách biến đổi giá trị của nó và quan sát
4. Sinh dữ liệu mới

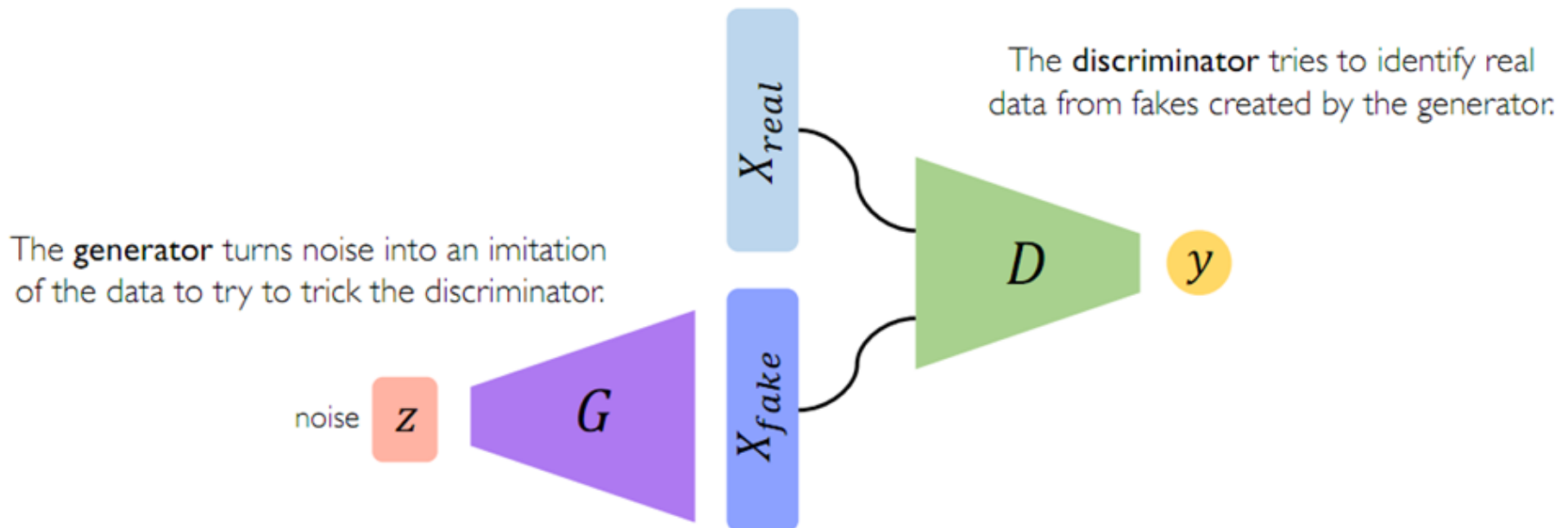




# Generative adversarial networks (GANs)

# Mạng sinh dữ liệu

- GANs là một mô hình sinh chứa hai mạng nơ-ron đối chọi lẫn nhau
- Mạng sinh (generator) biến véctơ nhiễu thành một dữ liệu giả để đánh lừa mạng phân loại (discriminator)
- Mạng phân loại (discriminator) cố gắng phân biệt đâu là dữ liệu thật, đâu là dữ liệu giả sinh bởi generator




# Trực giác ban đầu về GANs

- Generator tạo dữ liệu giả từ nhiễu

Generator



 Fake data


# Trực giác ban đầu về GANs

- Discriminator nhìn vào dữ liệu thật và giả

Discriminator

Generator



 Fake data

# Trực giác ban đầu về GANs

- Discriminator nhìn vào dữ liệu thật và giả

Discriminator

Generator



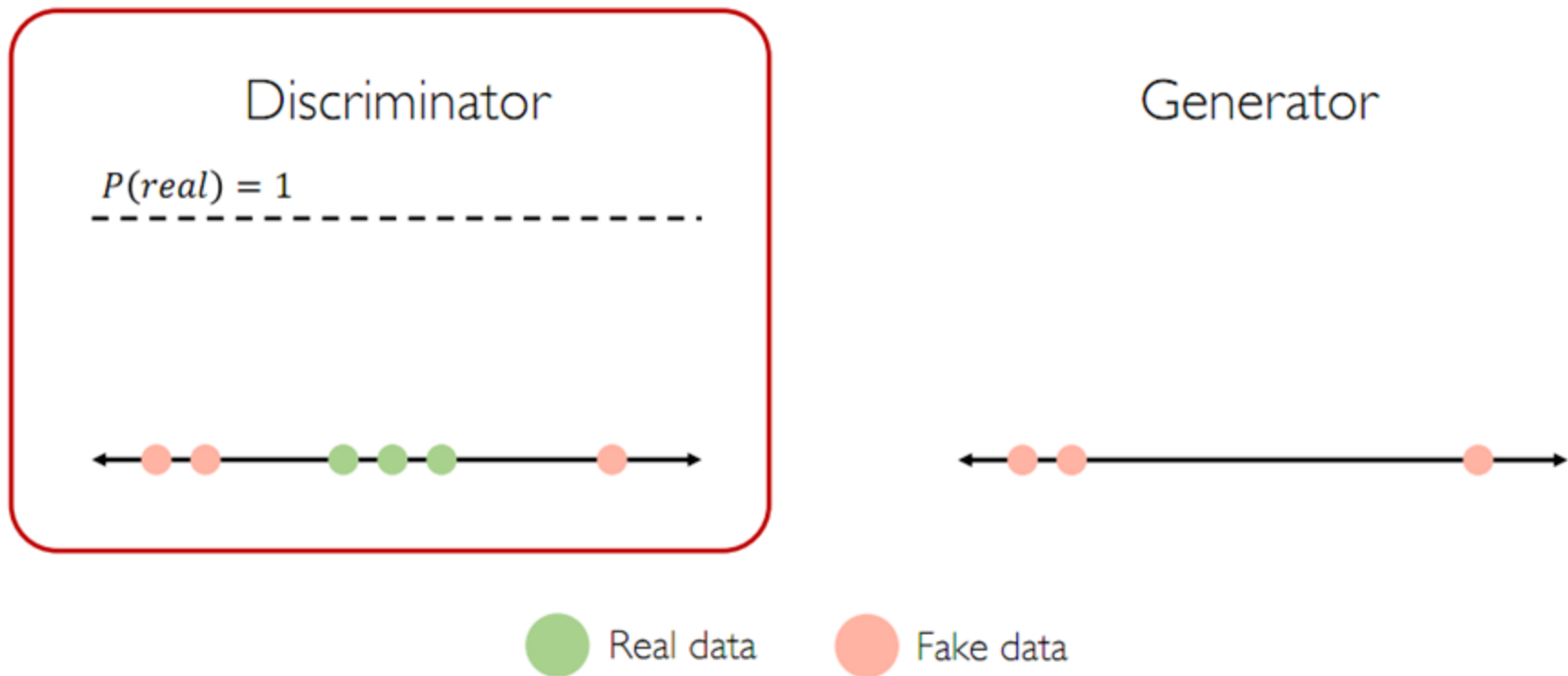
Real data



Fake data

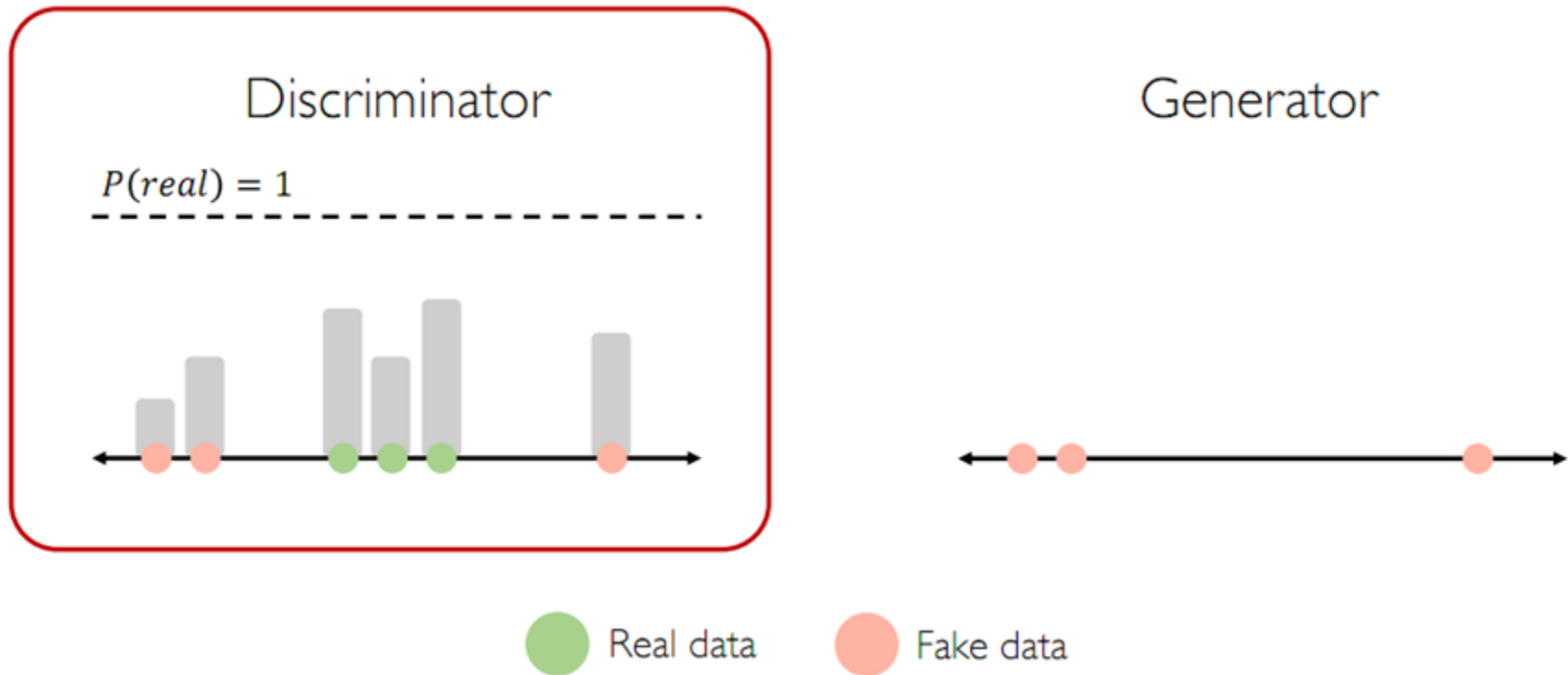
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật. đâu là giả



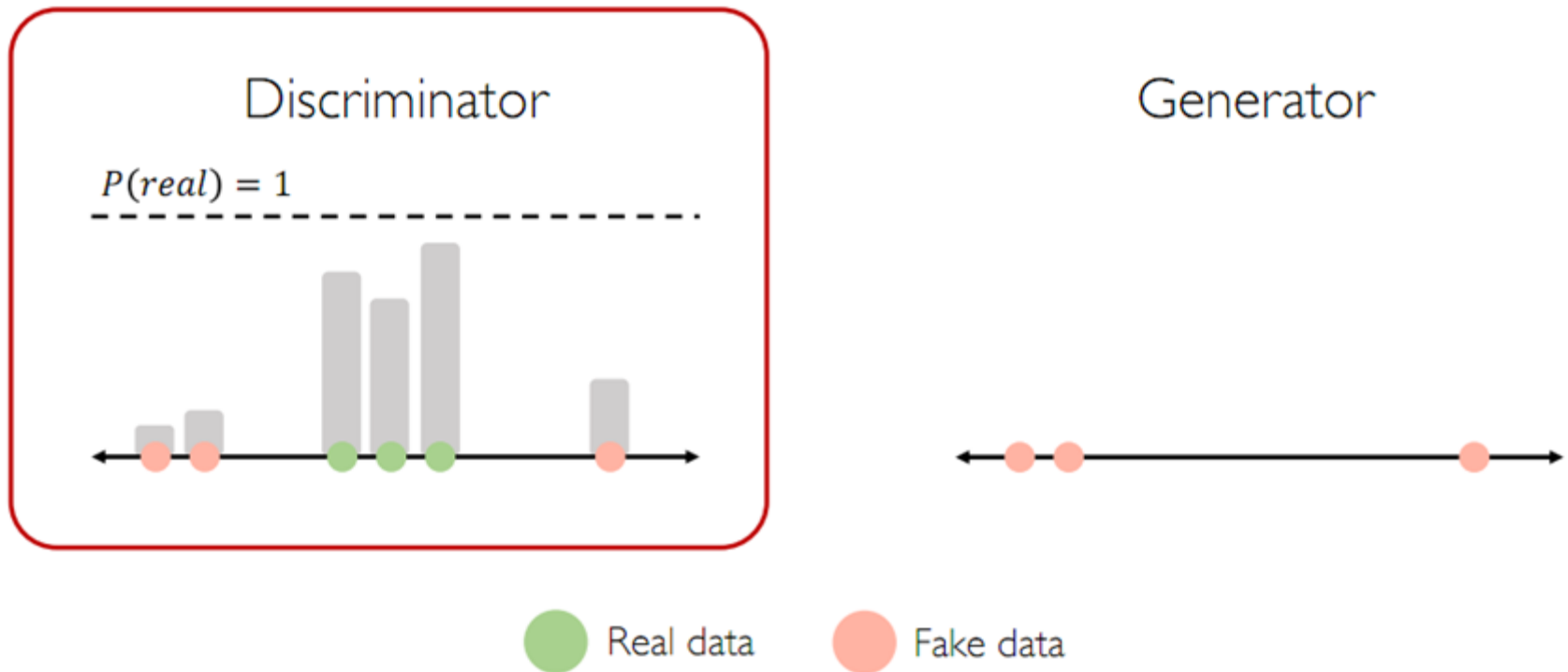
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật, đâu là giả



# Trực giác ban đầu về GANs

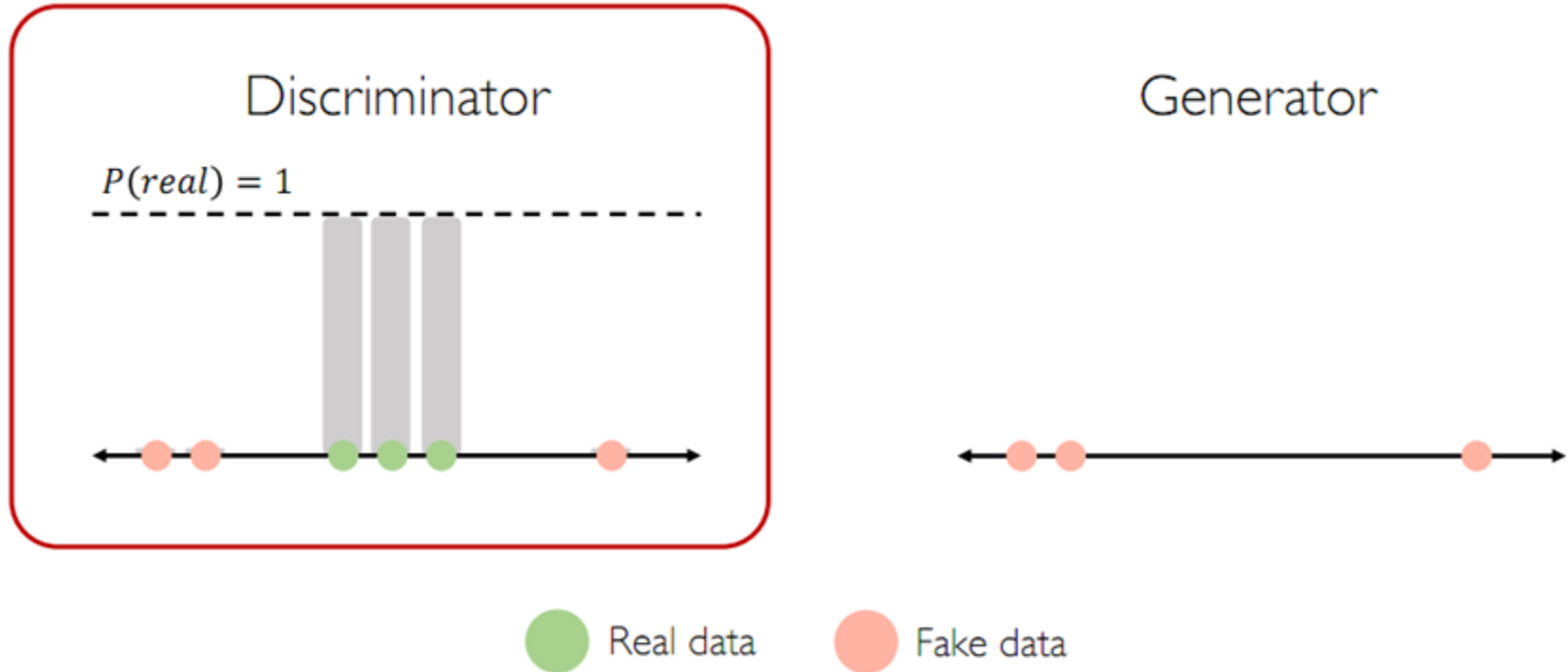
- Discriminator đoán đâu là thật, đâu là giả





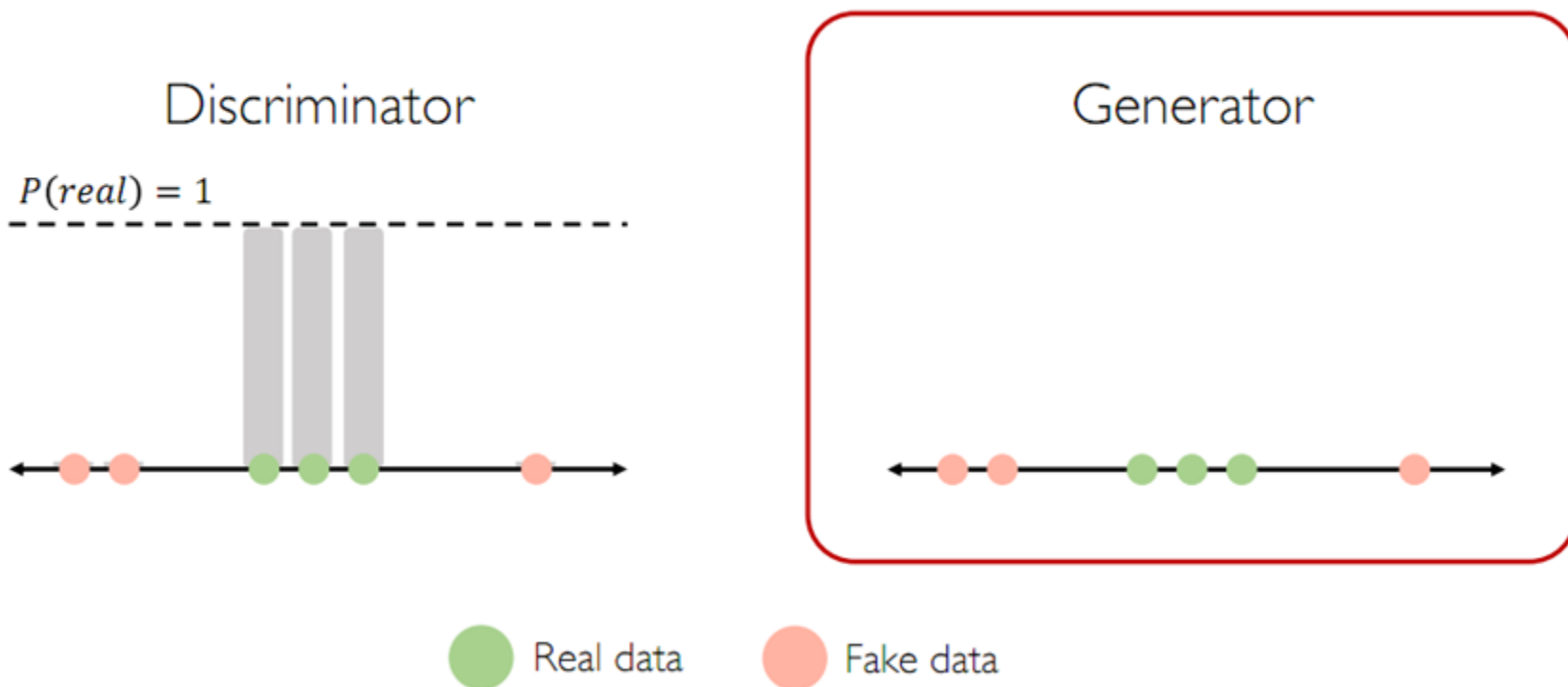
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật, đâu là giả



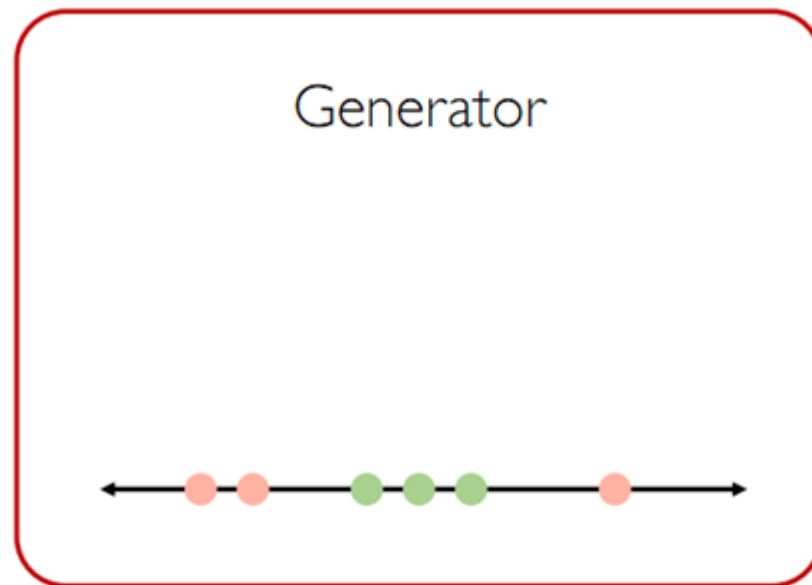
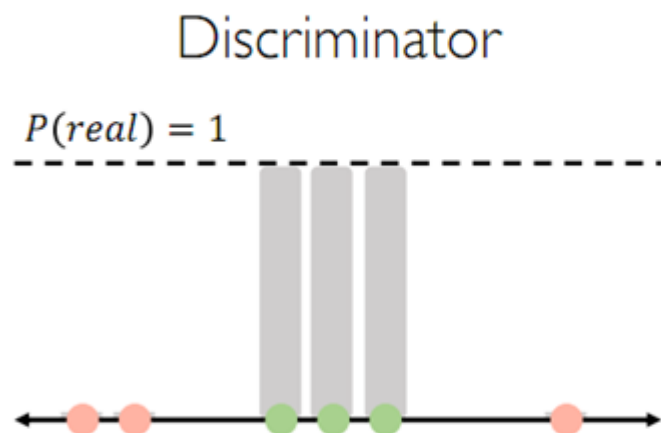
# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



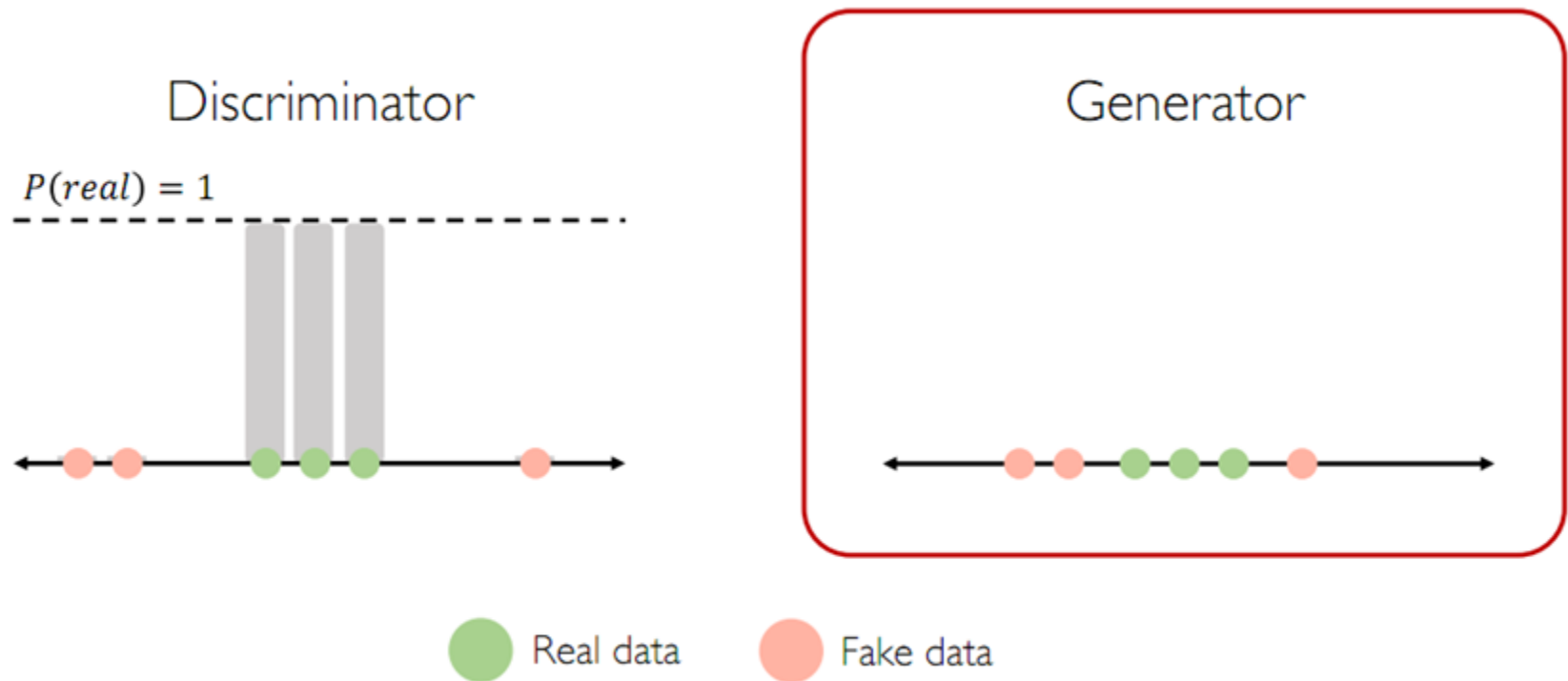
Real data



Fake data

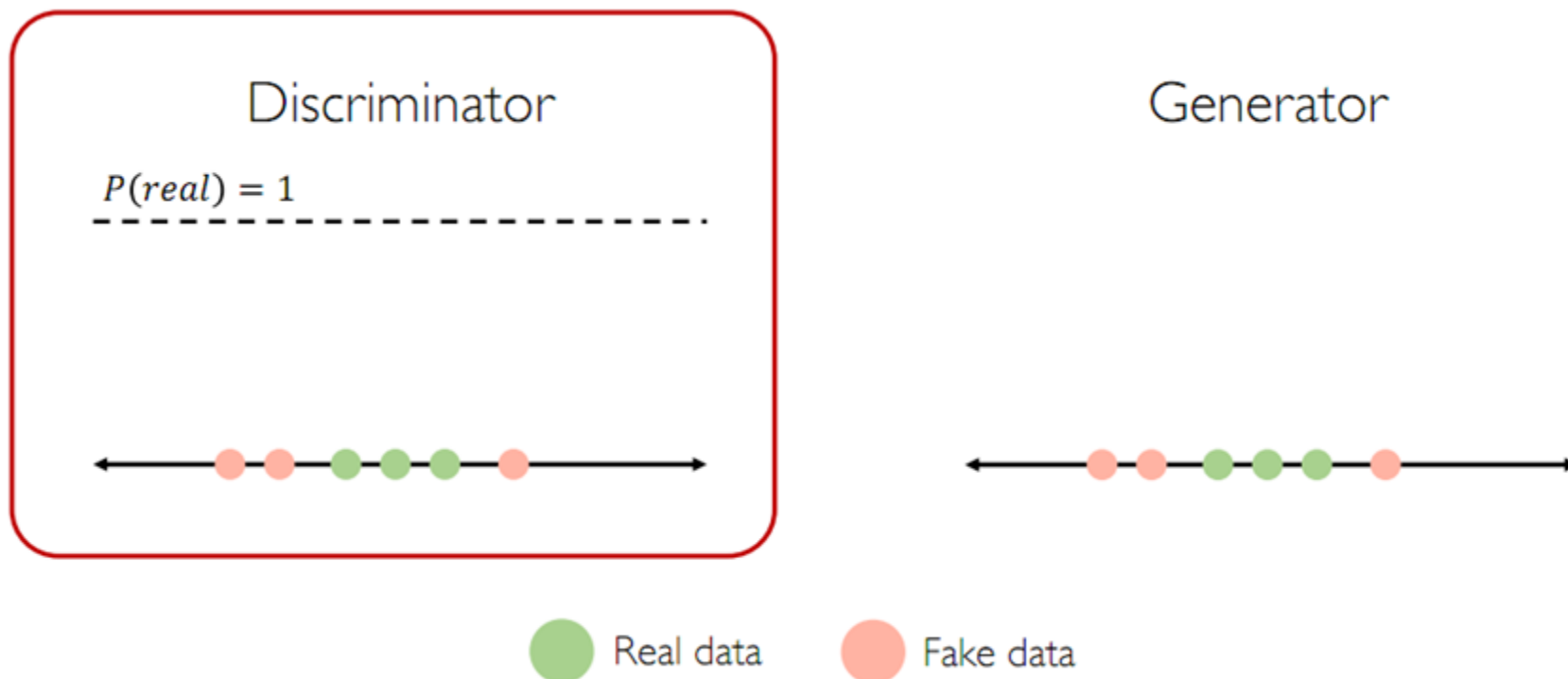
# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



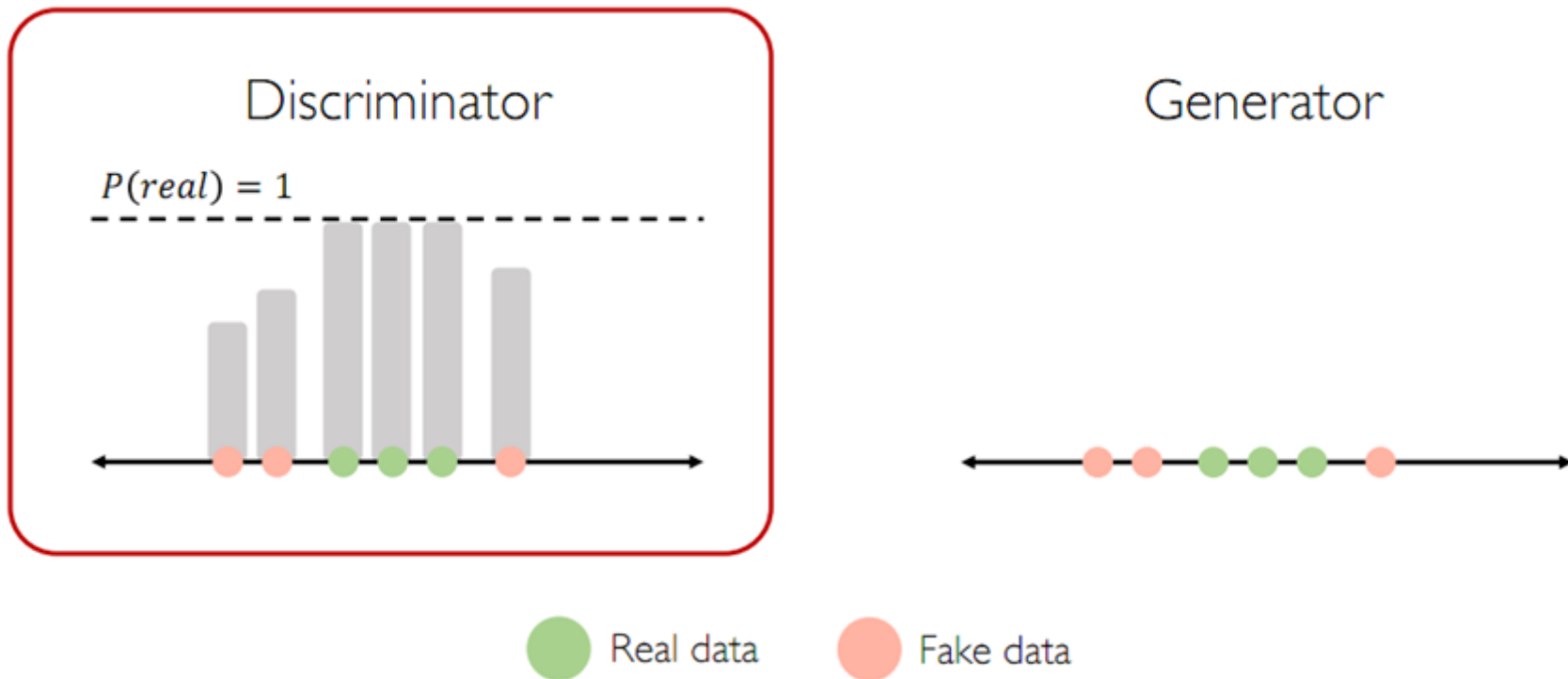
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật, đâu là giả



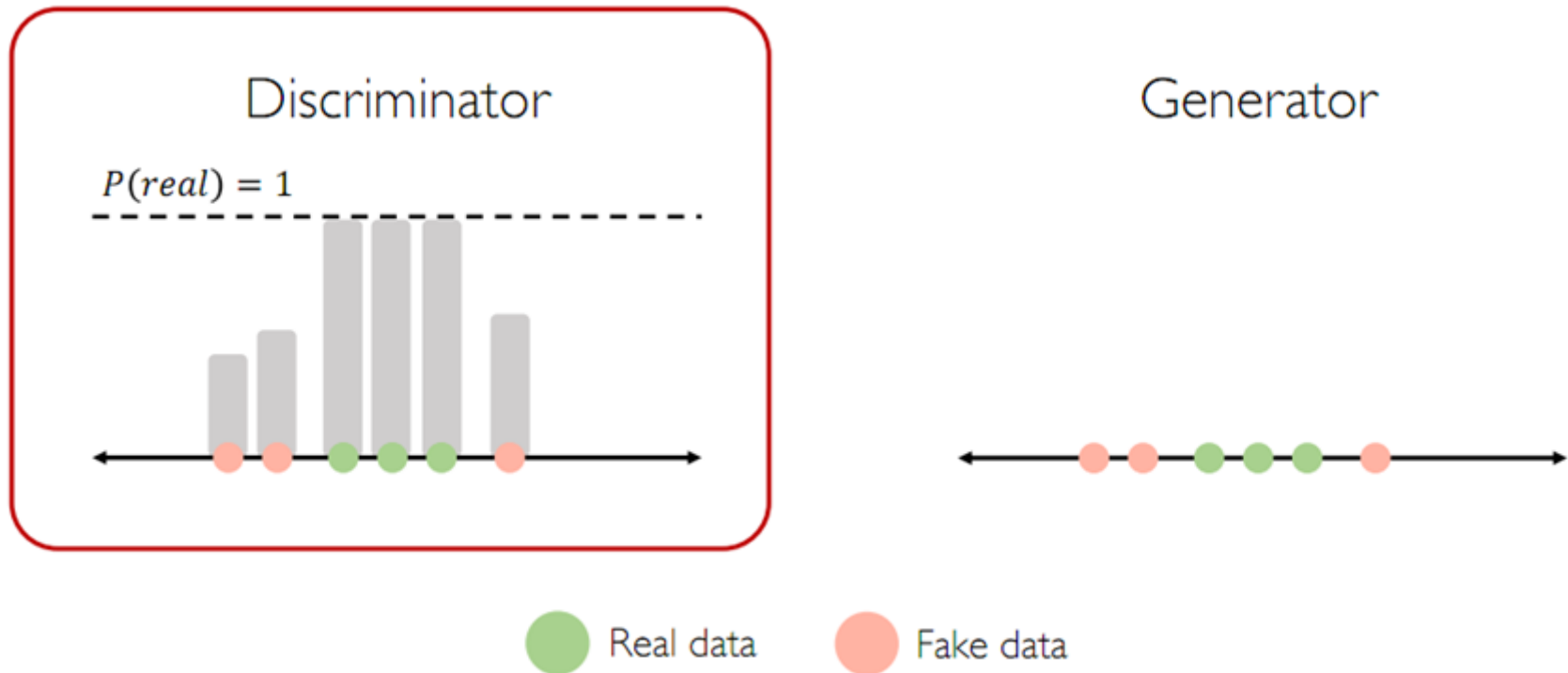
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật, đâu là giả



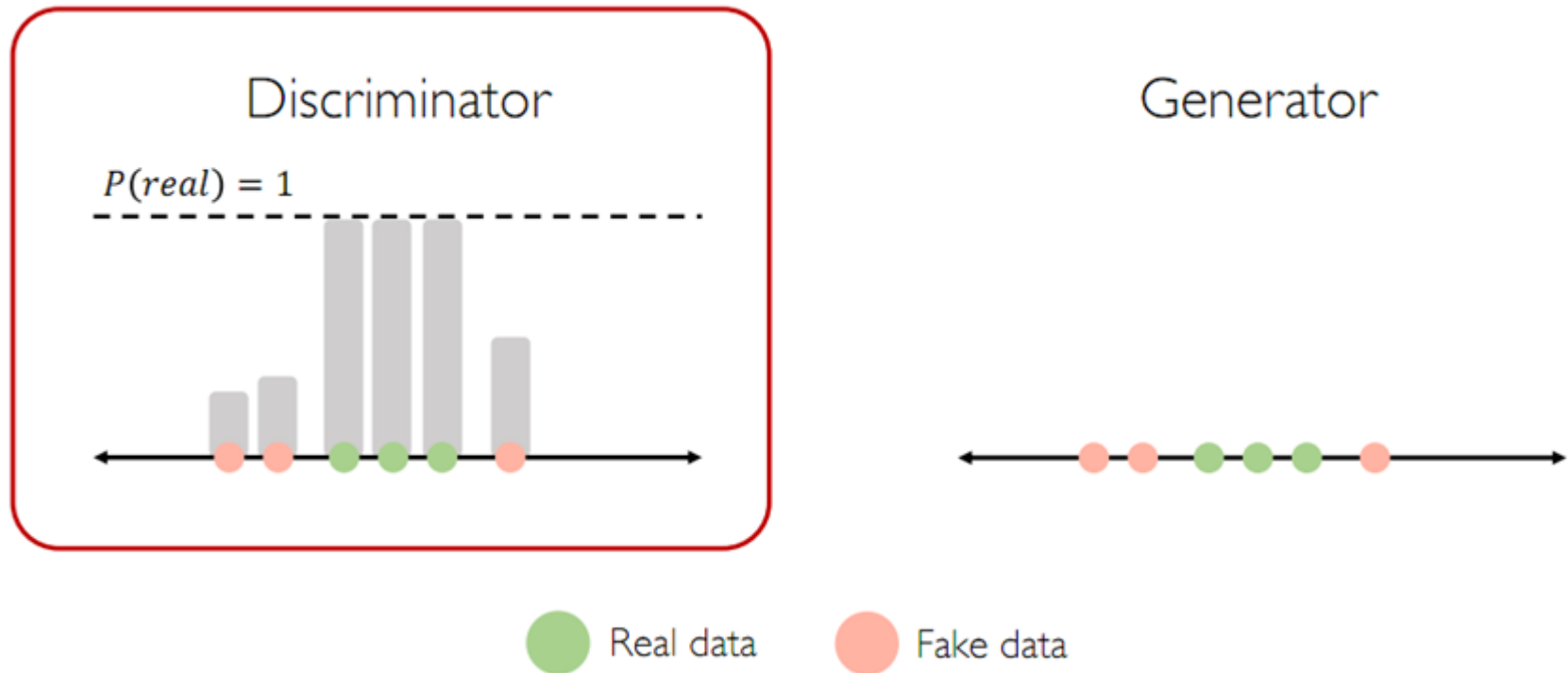
# Trực giác ban đầu về GANs

- Discriminator đoán đâu là thật, đâu là giả



# Trực giác ban đầu về GANs

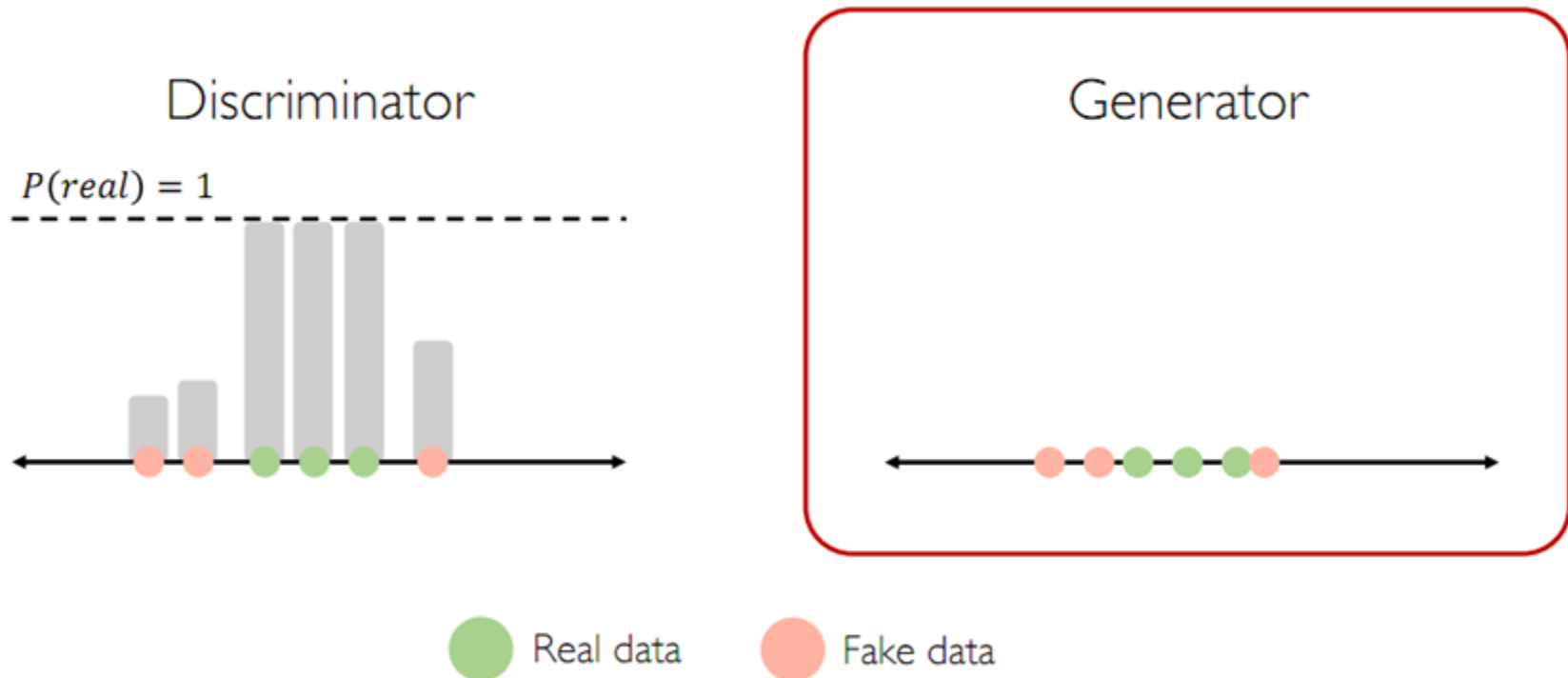
- Discriminator đoán đâu là thật, đâu là giả





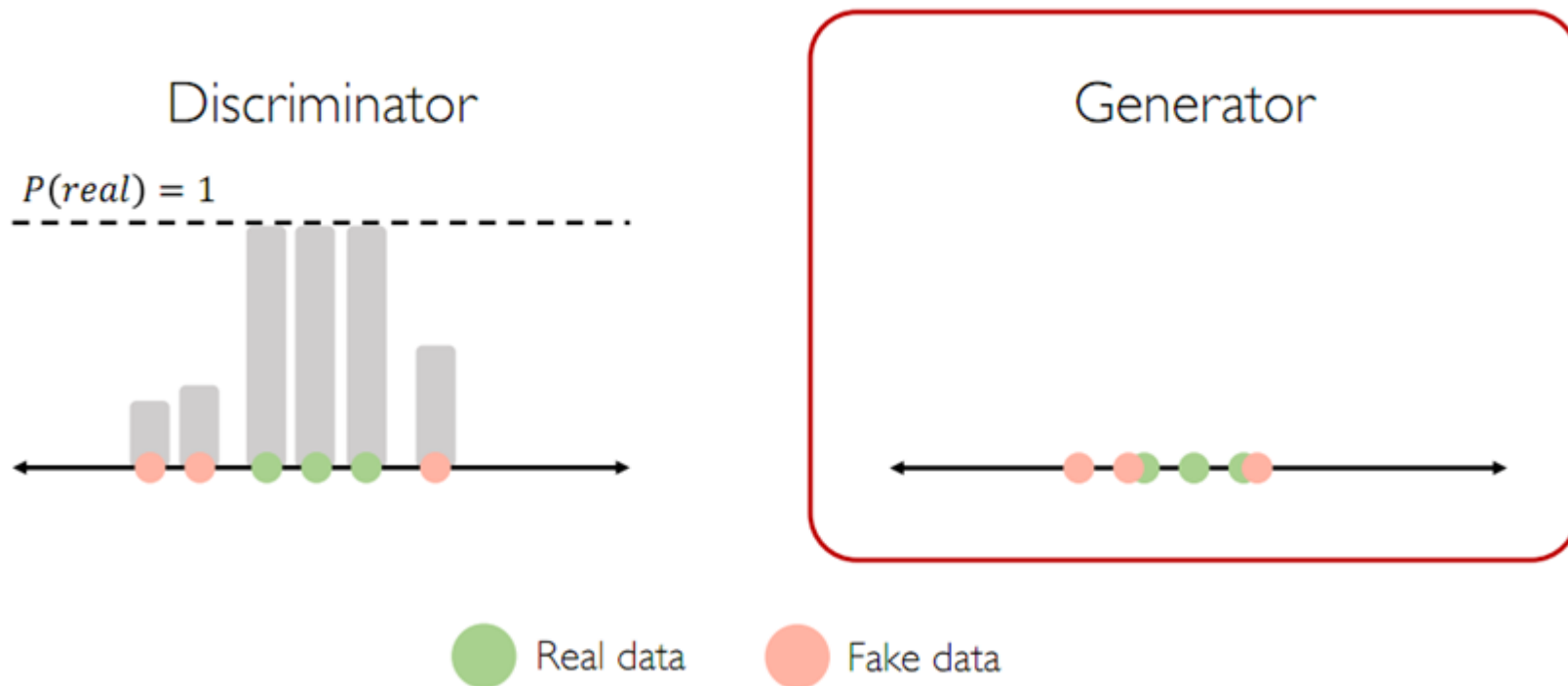
# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



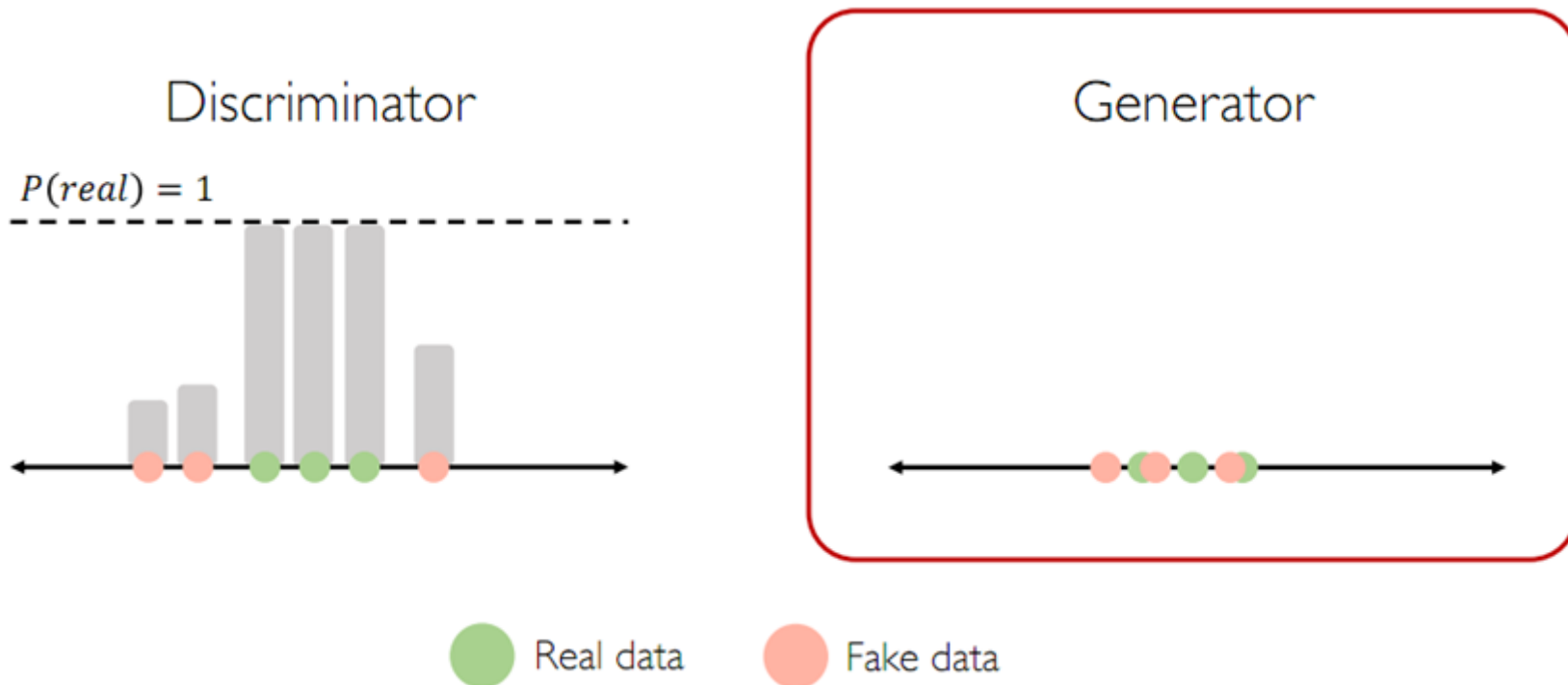
# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



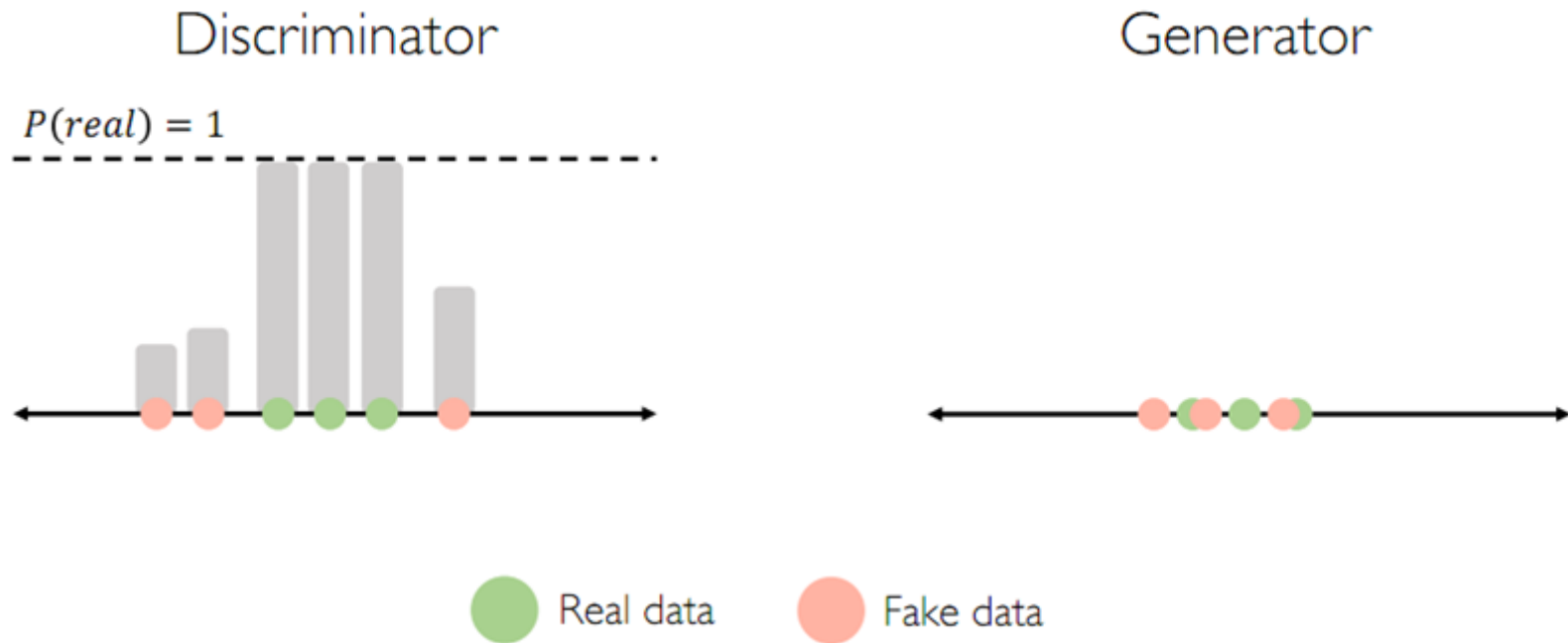
# Trực giác ban đầu về GANs

- Generator cố gắng cải tiến chất lượng dữ liệu giả



# Intuition behind GANs

- Discriminator cố gắng phân biệt dữ liệu thật và giả
- Generator cố gắng cải tiến chất lượng dữ liệu giả để lừa discriminator



# Huấn luyện GANs

- Discriminator cố gắng phân biệt dữ liệu thật và giả
- Generator cố gắng cải tiến chất lượng dữ liệu giả để lừa discriminator

Train GAN jointly via **minimax** game:

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

**Discriminator** wants to maximize objective s.t.  $D(x)$  close to 1,  $D(G(z))$  close to 0.

**Generator** wants to minimize objective s.t.  $D(G(z))$  close to 1.

# Huấn luyện GANs

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator,  $k$ , is a hyperparameter. We used  $k = 1$ , the least expensive option, in our experiments.

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

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

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

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

**end for**

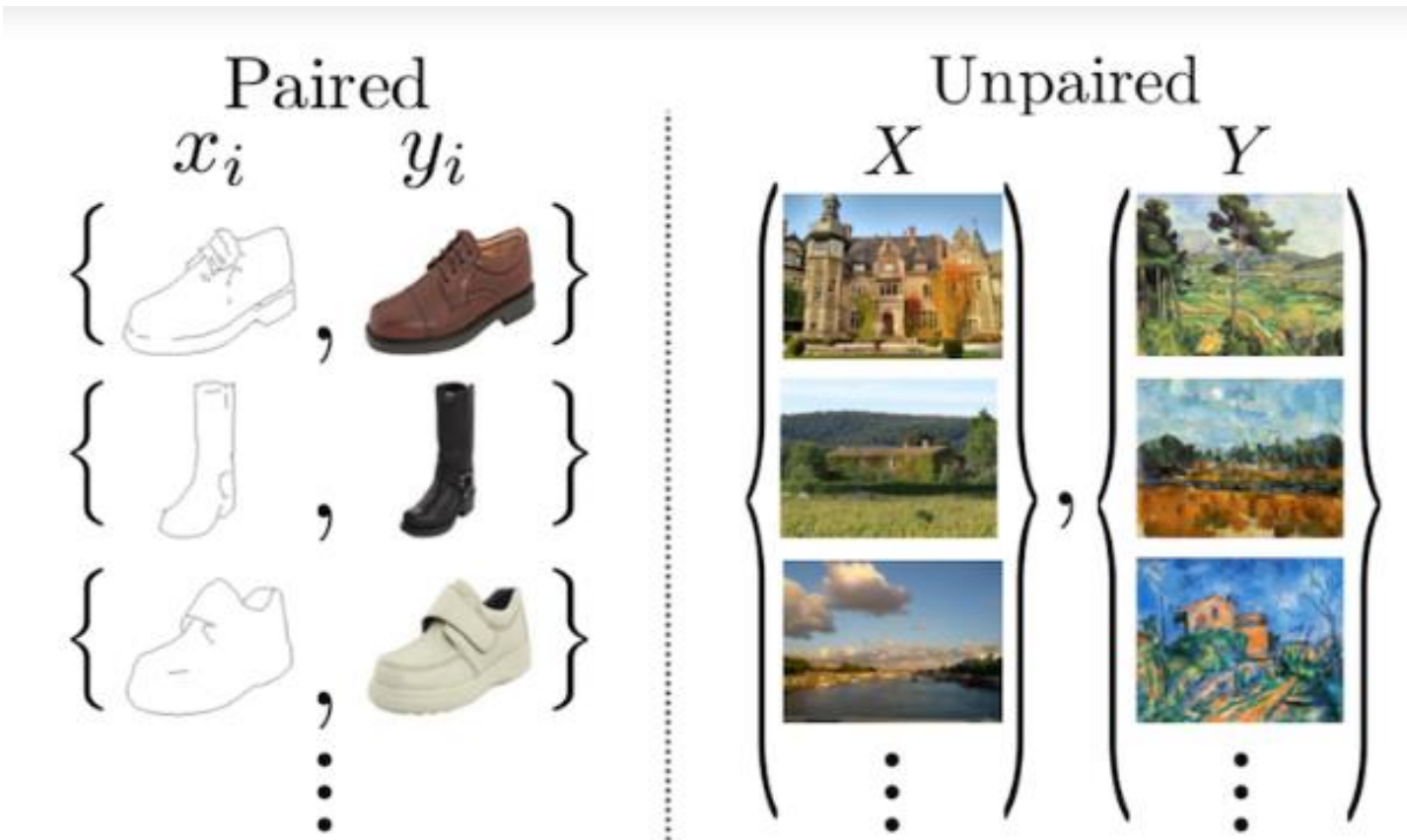
- Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left( 1 - D(G(z^{(i)})) \right).$$

**end for**

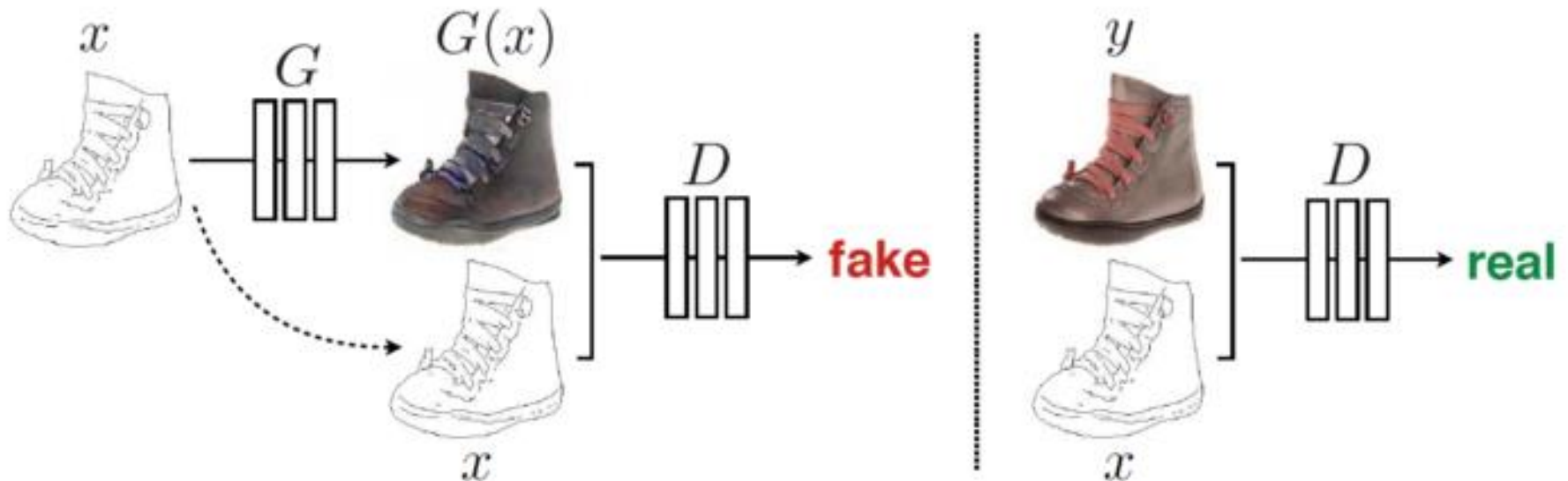
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

# Biến đổi Image-to-image



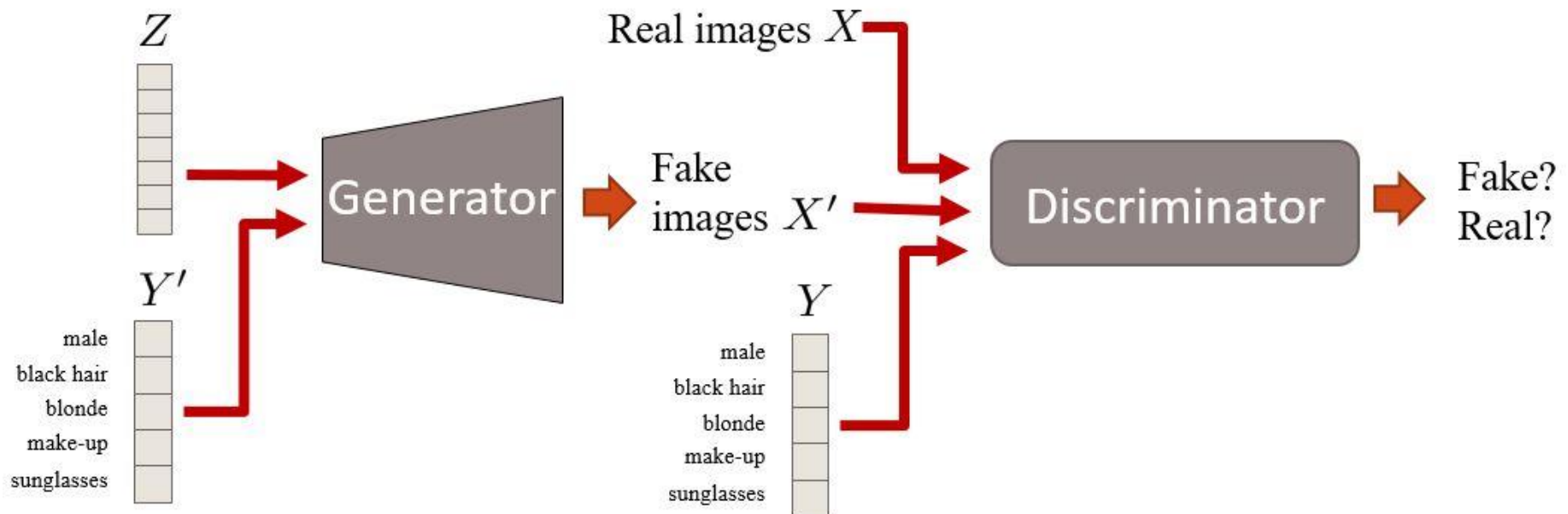
# pix2pix

- 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



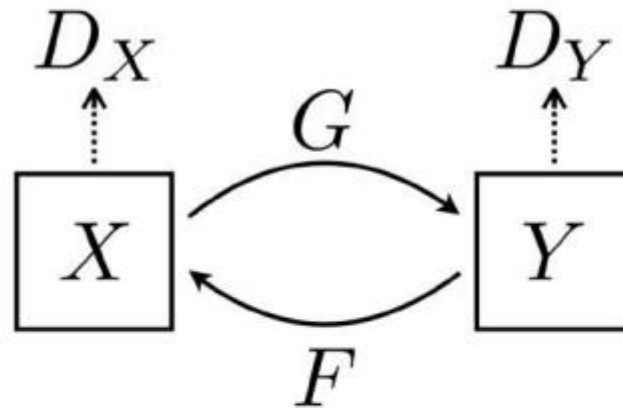


# Conditional GAN

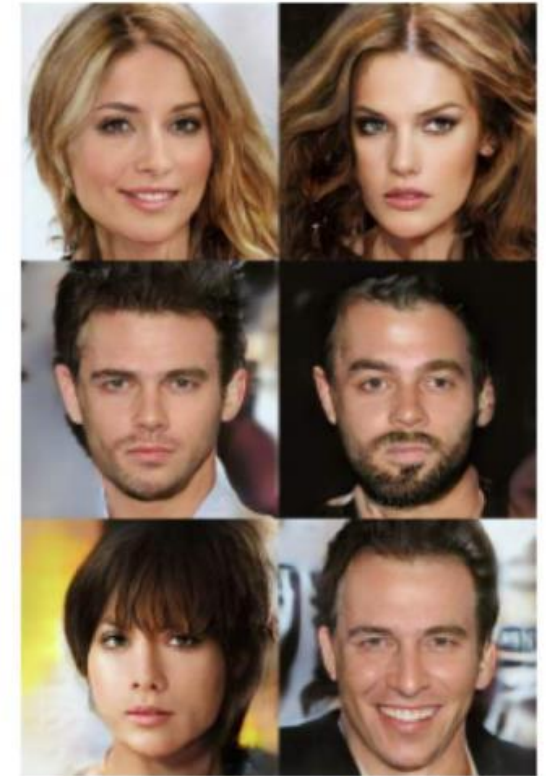
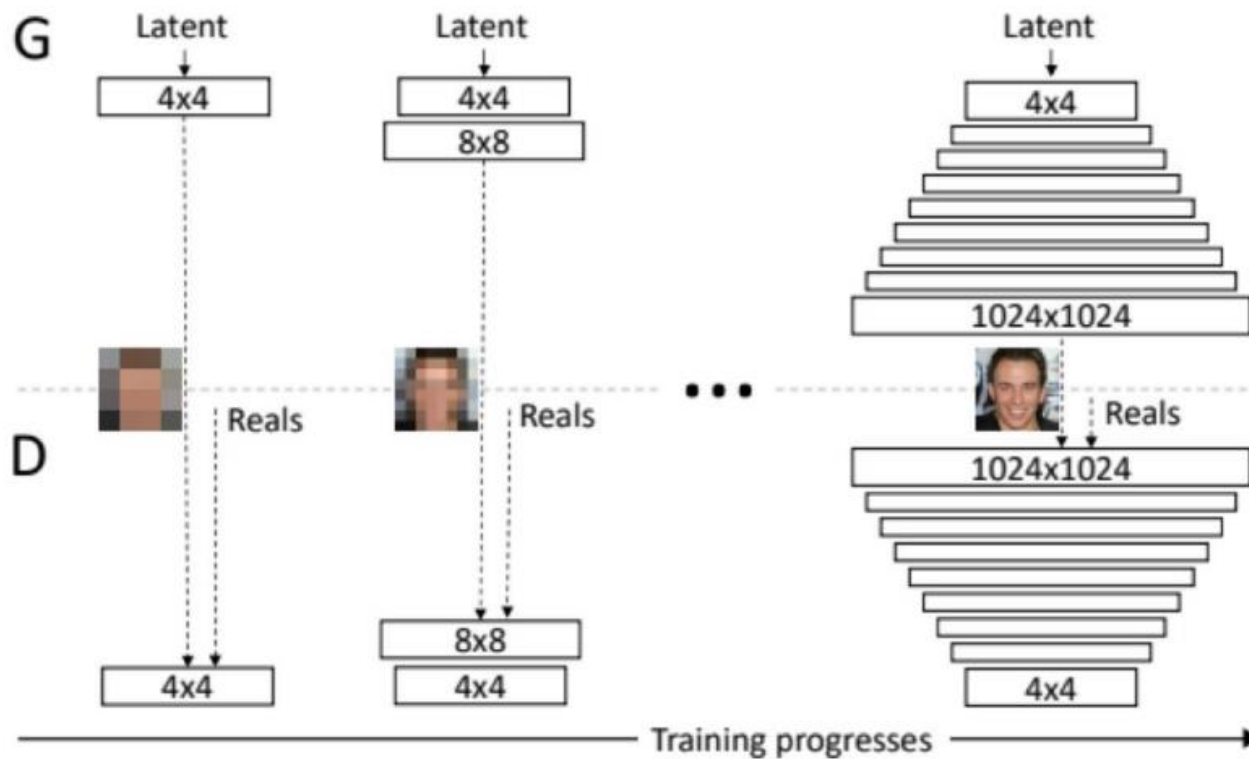


# CycleGAN

CycleGAN learns transformations across domains with unpaired data.



# Progressive growing of GANs (NVIDIA)



Karras et al., ICLR 2018.

# Progressive growing of GANs (NVIDIA)



Karras et al., ICLR 2018.



# Style-based generator



Karras et al., Arxiv 2018.

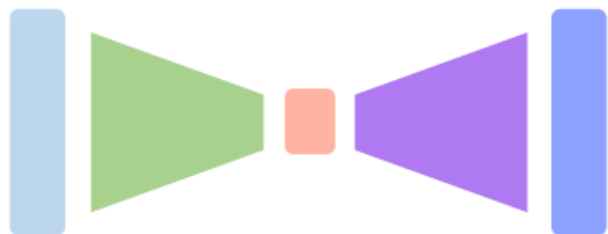
# Style-based transfer



# Tổng kết mô hình sinh

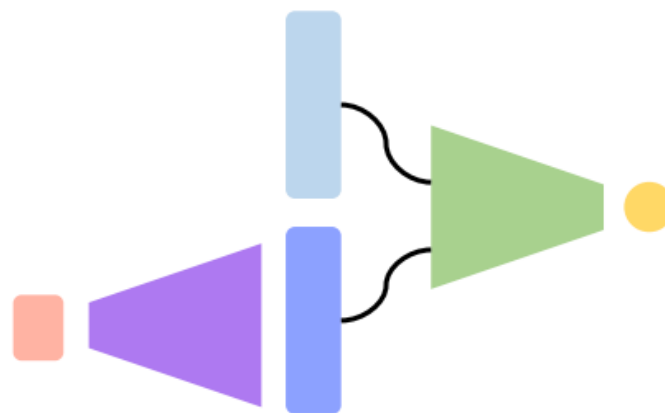
## Autoencoders and Variational Autoencoders (VAEs)

Learn **lower-dimensional** latent space and **sample** to generate input reconstructions



## Generative Adversarial Networks (GANs)

Competing **generator** and **discriminator** networks



# Tài liệu tham khảo

1. Khóa MIT Deep Learning 6.S191:

<http://introtodeeplearning.com/>

2. Khóa cs231n của Stanford:

[http://cs231n.stanford.edu/slides/2020/lecture\\_11.pdf](http://cs231n.stanford.edu/slides/2020/lecture_11.pdf)





25 YEARS ANNIVERSARY  
**SOICT**

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Chân thành  
cảm ơn!!!



[soict.hust.edu.vn/](http://soict.hust.edu.vn/)



[fb.com/groups/soict](https://fb.com/groups/soict)

