HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Lesson 12
Generative models

Outline

- Introduction to generative models
- Autoencoder
- GANs

Introduction to generative models

Which face is real?

- http://www.whichfaceisreal.com/

Supervised vs. unsupervised learning

Supervised

- Data sample: (x, y)

x are input features, y is label

- Objective: Learn the mapping function x → y
- Eg.: Classification, regression, object recognition, semantic segmentation, machine translation,...

Unsupervised

- Data sample: x

x are input features, no label!

- Objective: Learn hidden structure and relationships from inputs
- Eg: clustering, association rules, dimension reduction,…

Generative models

- Deep generative models (DGM) are neural networks with many hidden layers trained to approximate complicated, high-dimensional probability distributions using samples.

- How to learn $p_{model}(x)$ which should be like $p_{data}(x)$?

Why generative models?

- Because it can uncover underlying hidden information in data

Homogeneous skin color, pose

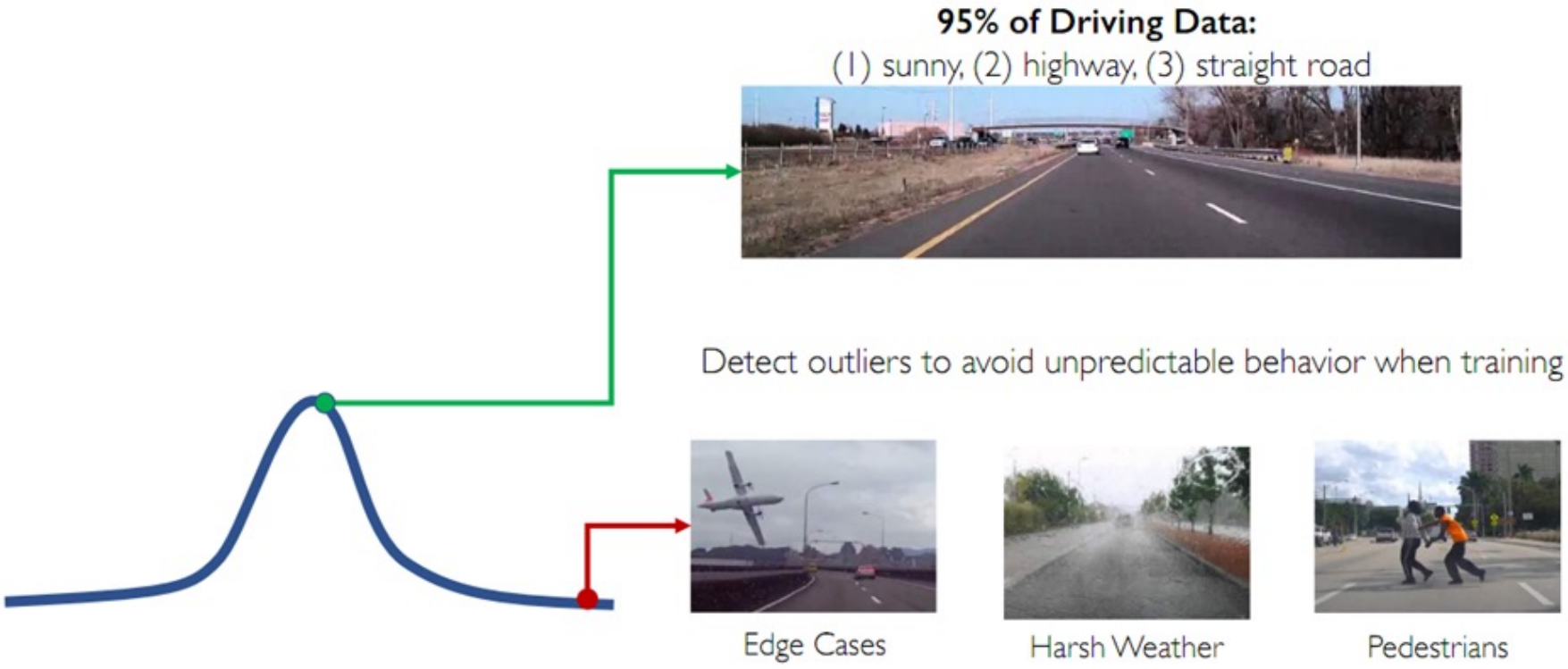
VS

Diverse skin color, pose, illumination

- Use learned latent distribution to generate more diverse and balanced data (debias)

Why generative models? (2)

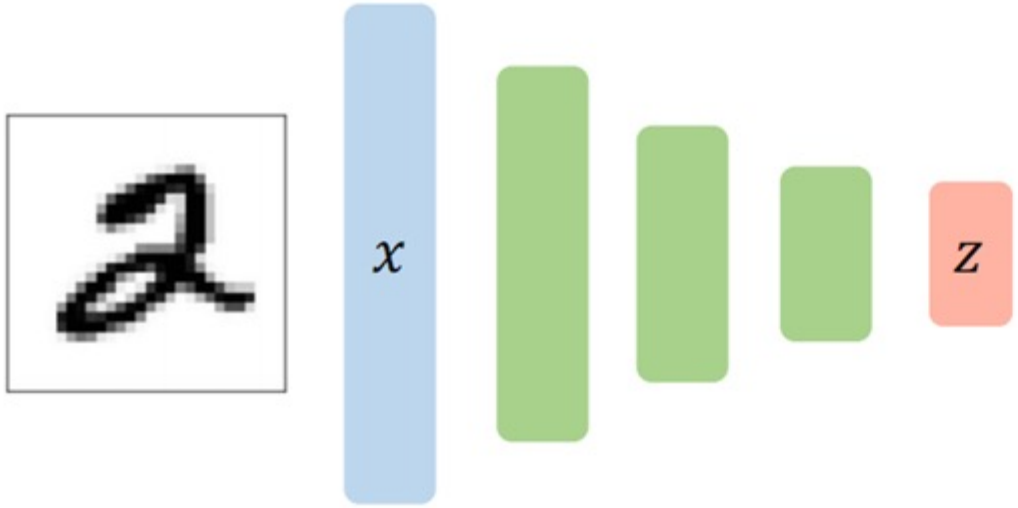
- Outlier detection: How to detect a rare occurrence of a new event?
  - Use the generative model to learn the distribution of the data, thereby identifying outliers based on the learned distribution.



Autoencoder

Autoencoder

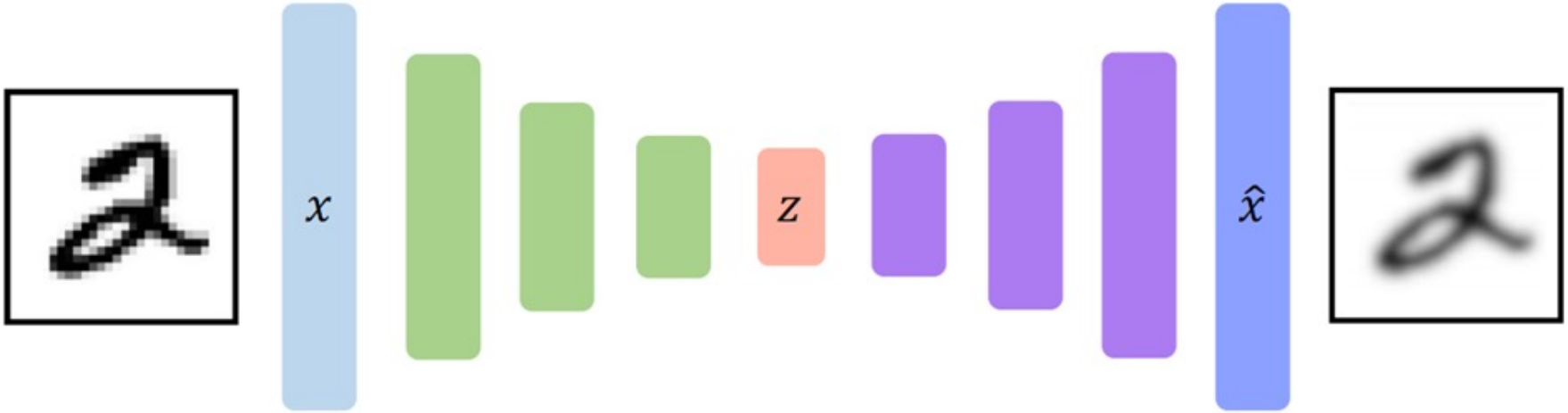
- It is an unsupervised learning model that allows to learn the feature representation with a smaller number of dimensions from the unlabeled training set
- "Encoder" learns mapping from data x into hidden space z of lower dimension



Why do we care about a low-dimensional z?

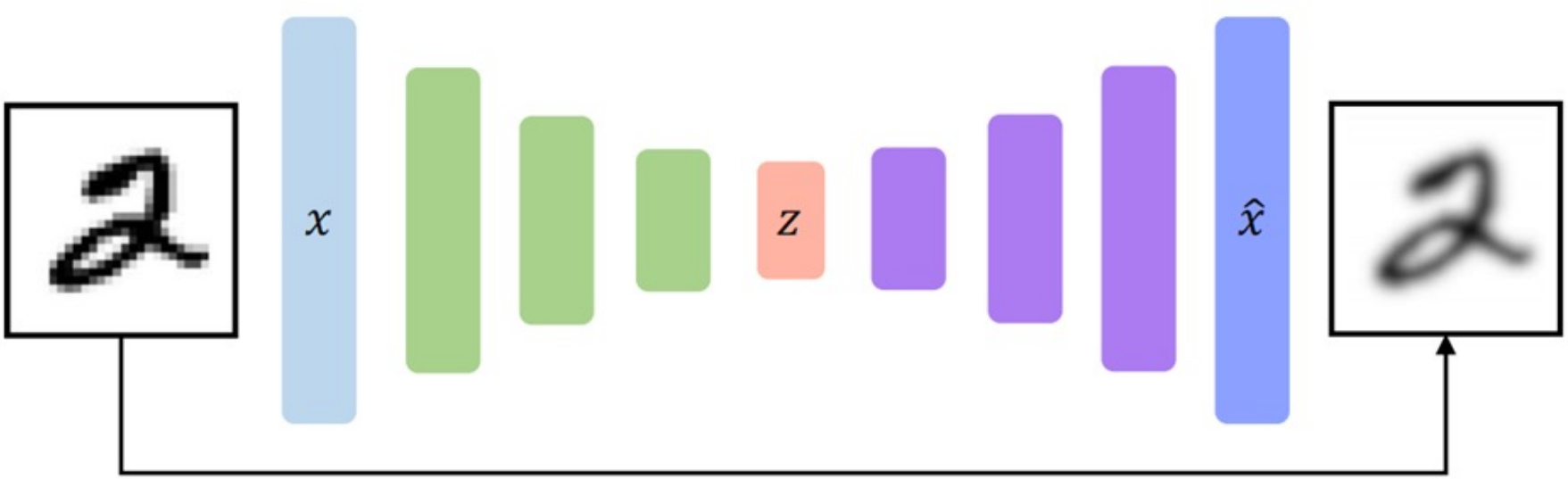
Autoencoder (2)

- How to learn hidden space?
- Train the model to use the hidden feature z to restore the original data
- "Decoder" maps the hidden feature z back to recover the input data information



Autoencoder (3)

- How to learn hidden space?
- Train the model to use the hidden feature z to restore the original data
- The objective function doesn't need a label!



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

The number of hidden dimensions affects the recovery quality

- Autoencoder is a type of data compression.
- The smaller the number of hidden dimensions, the larger the training bottleneck

2D latent space

5D latent space

Ground Truth

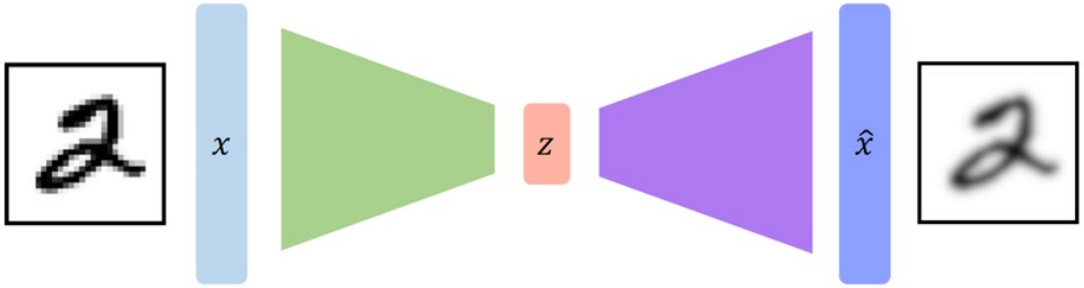
Autoencoders for learning representation

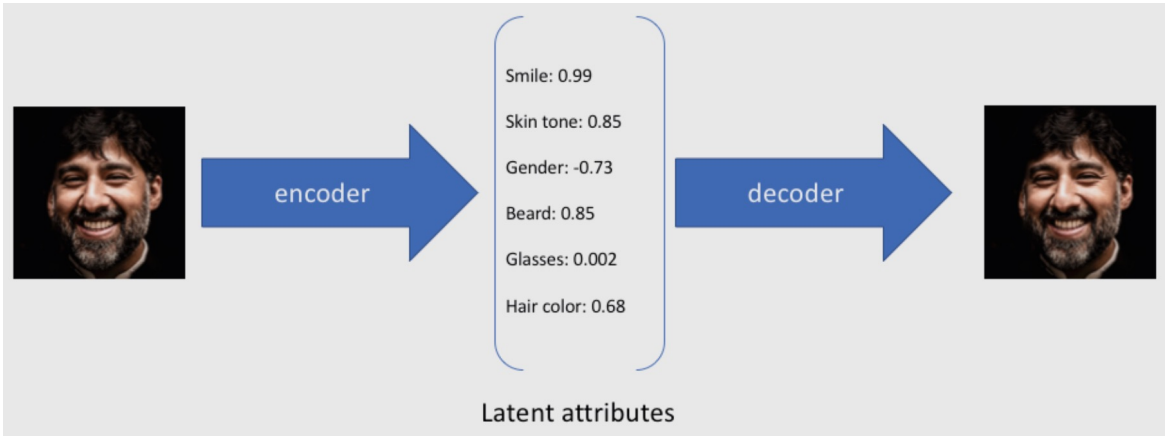
- Encoder
  - Hidden layers are bottlenecks that force the network to compress data representation
- Decoder
  - The objective function for reconstruction forces the hidden representation to encode as much information from the input as possible
- Autoencoding = **Auto**matically **encoding** data

Variational Autoencoders (VAEs)

Reviewing Autoencoders

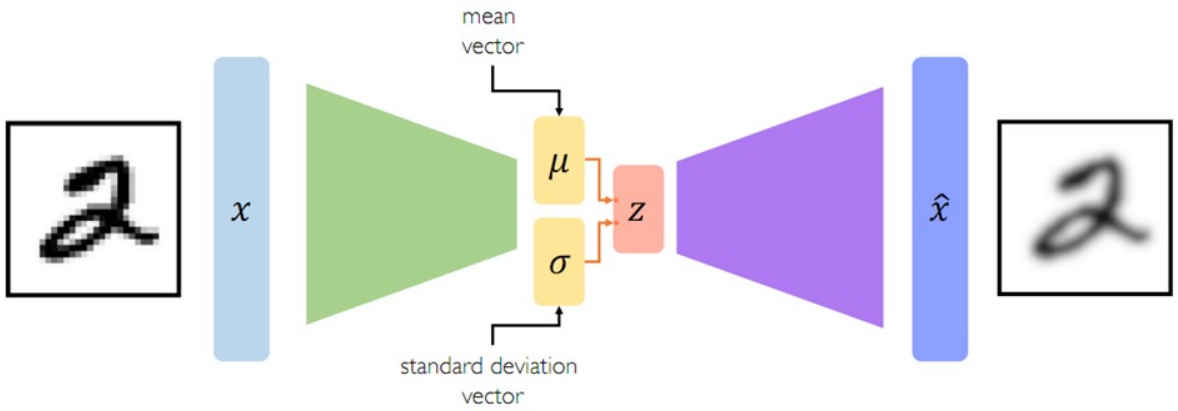
- Each dimension in the encoding vector contains a single value that represents a hidden attribute from the input data.

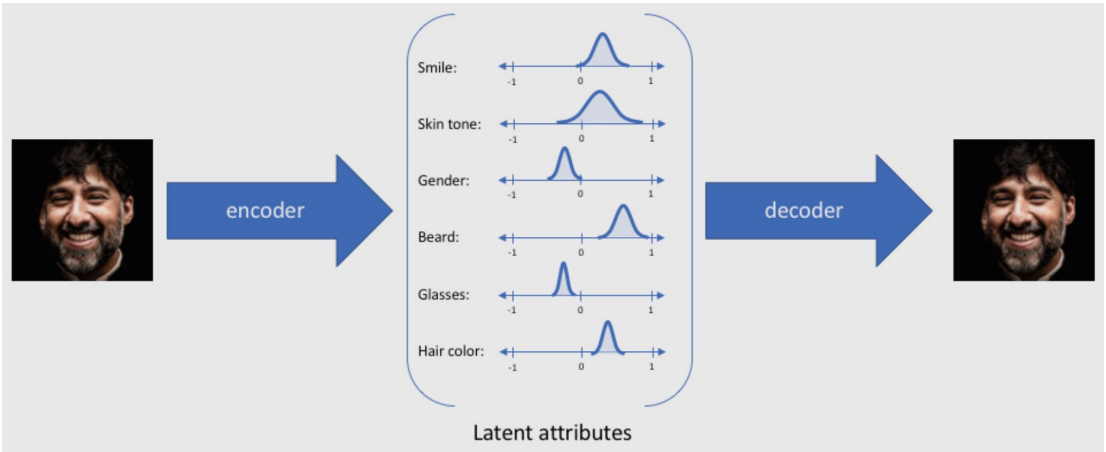




Variational Autoencoders

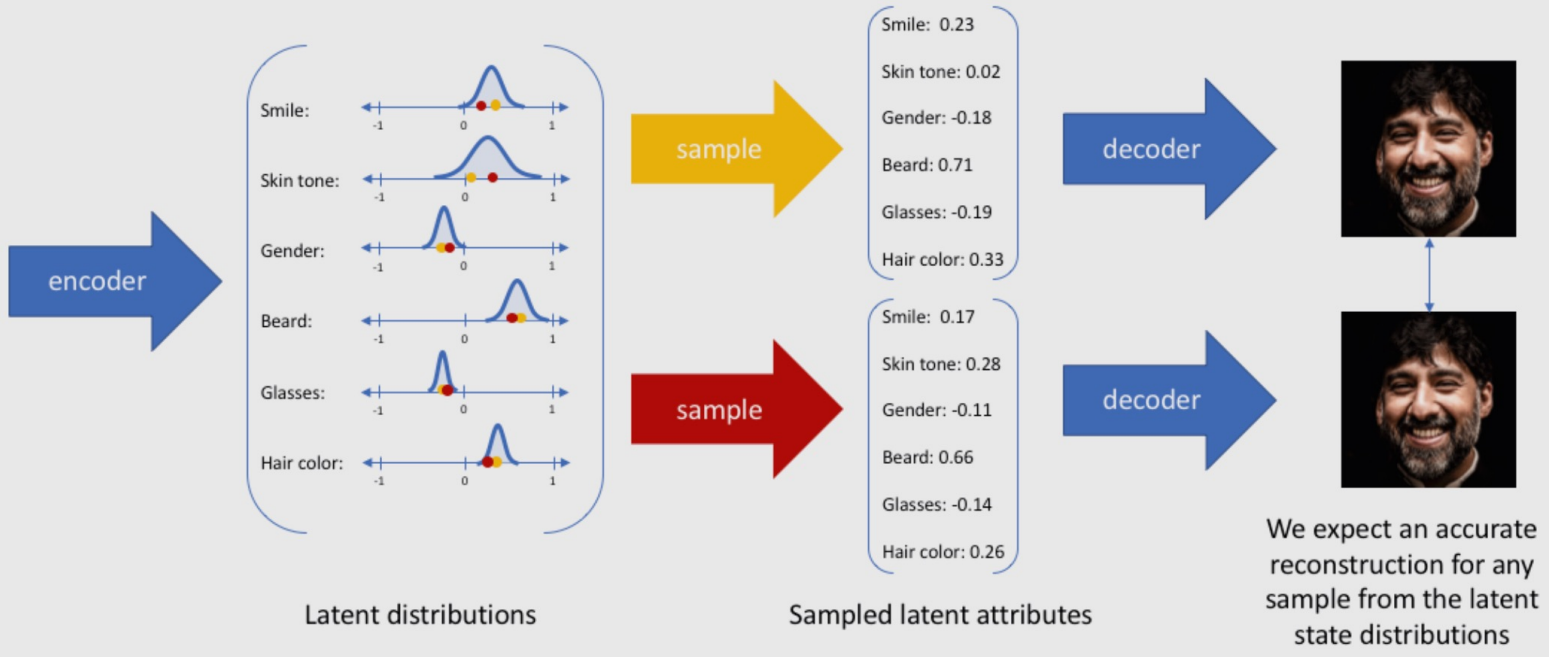
- Each hidden attribute is a probability distribution



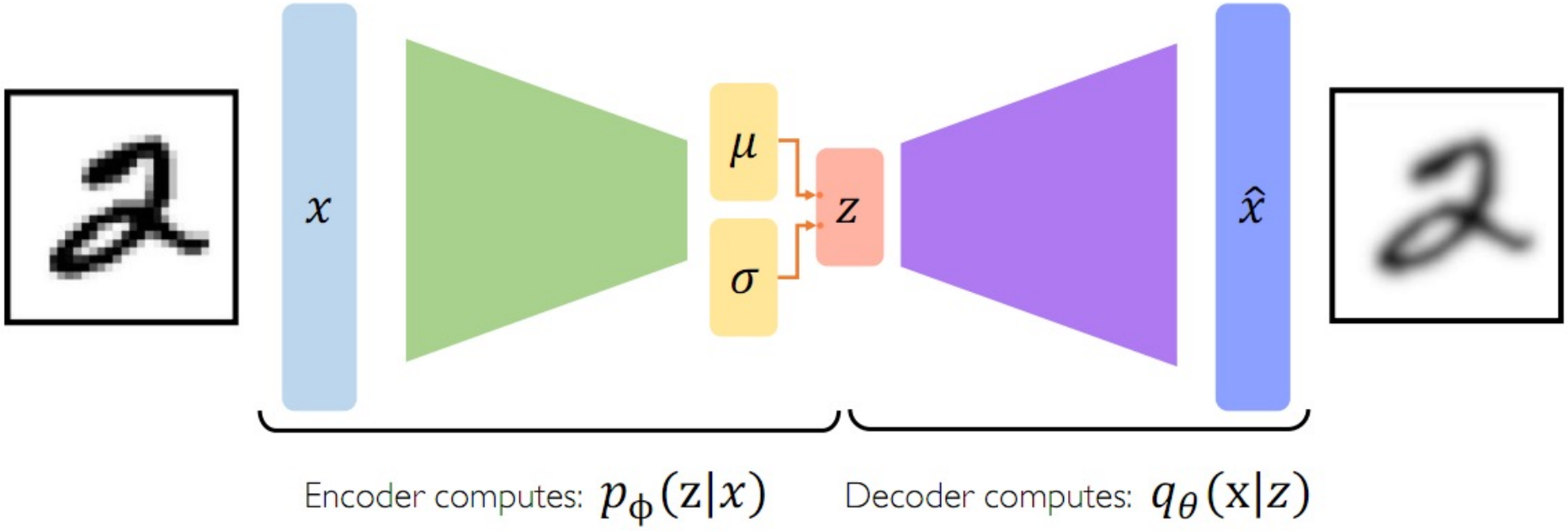


Decoder in VAEs

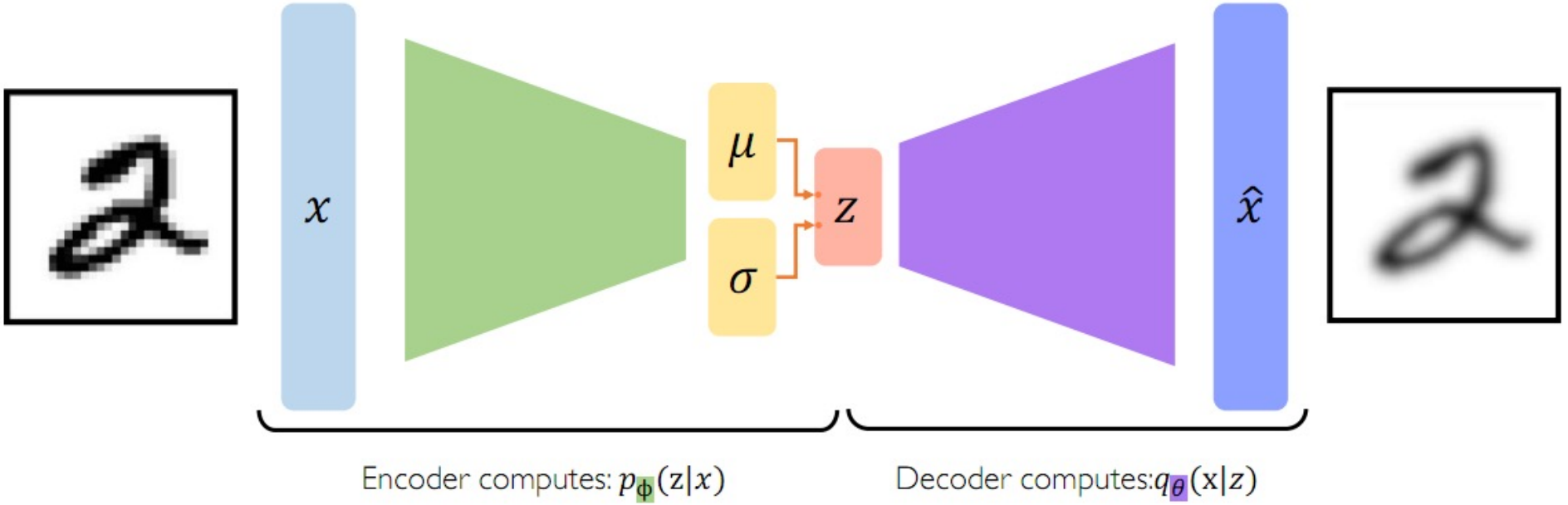
- When decoding from the hidden state, take a random sample from the distribution of each hidden attribute to generate the input vector for the decoder.



VAE optimization

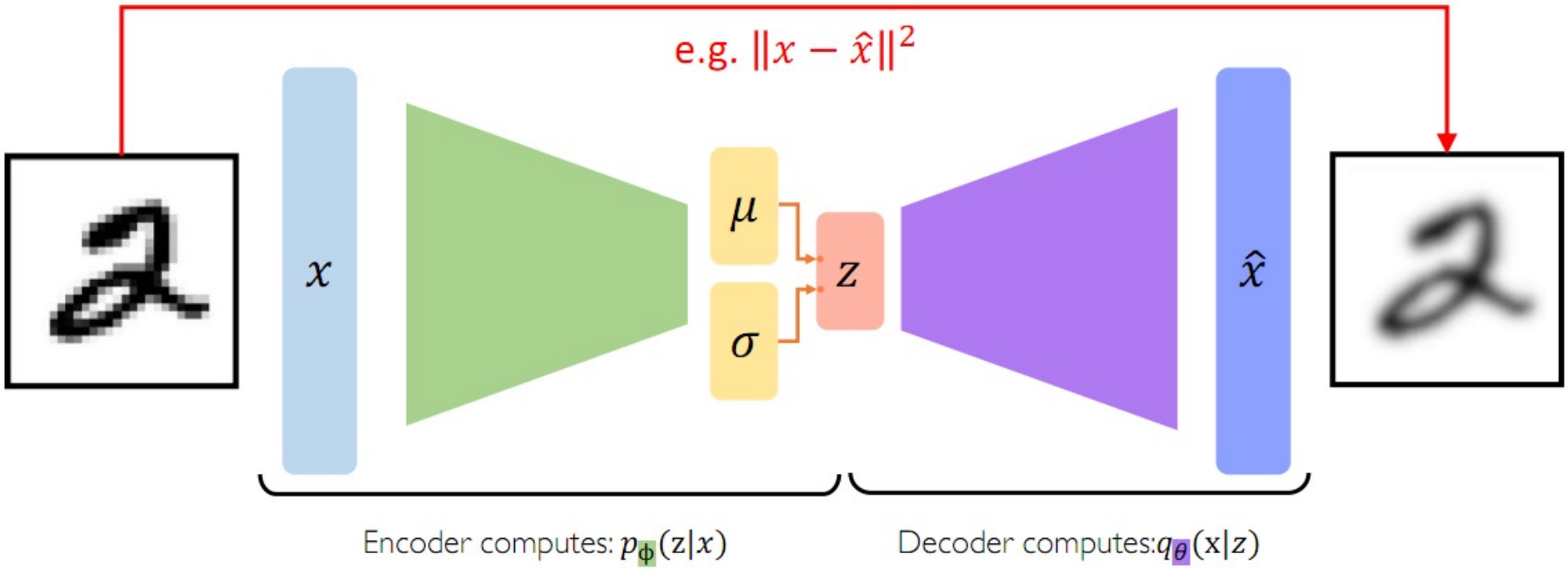


VAE optimization



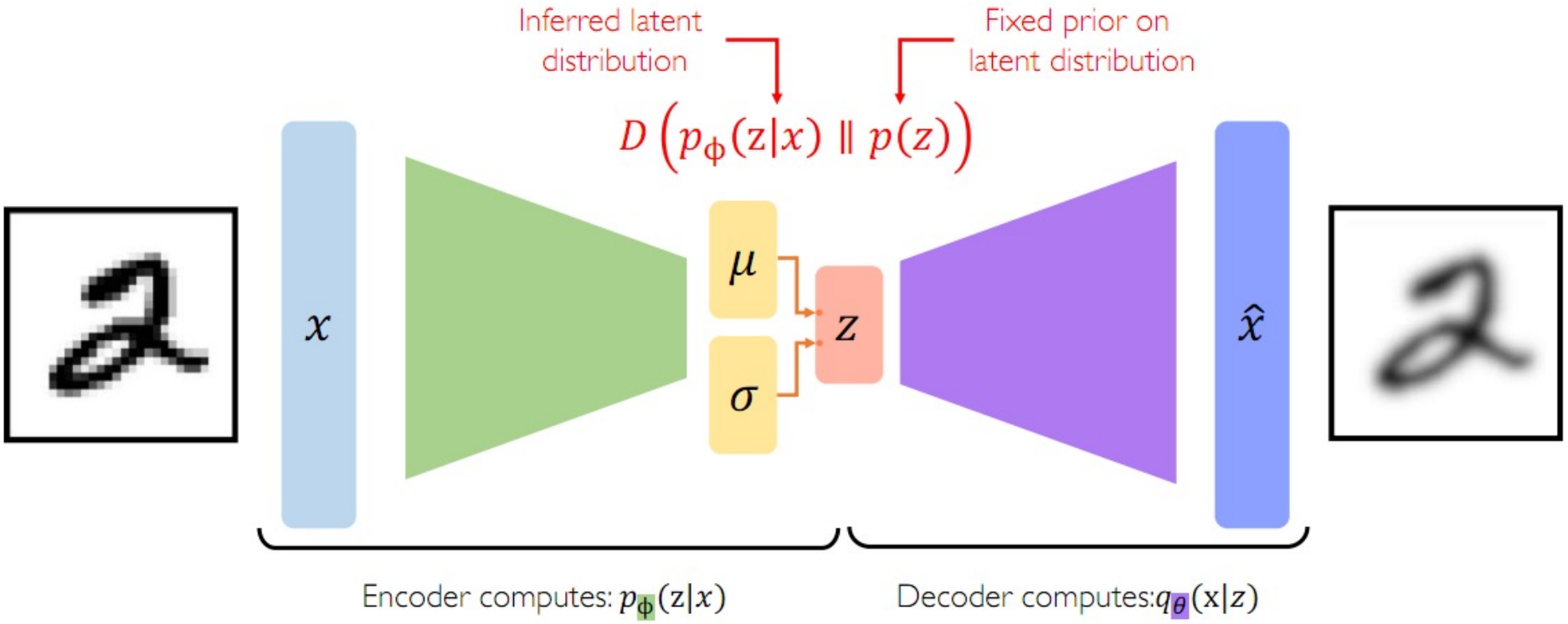
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

Reconstruction loss



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

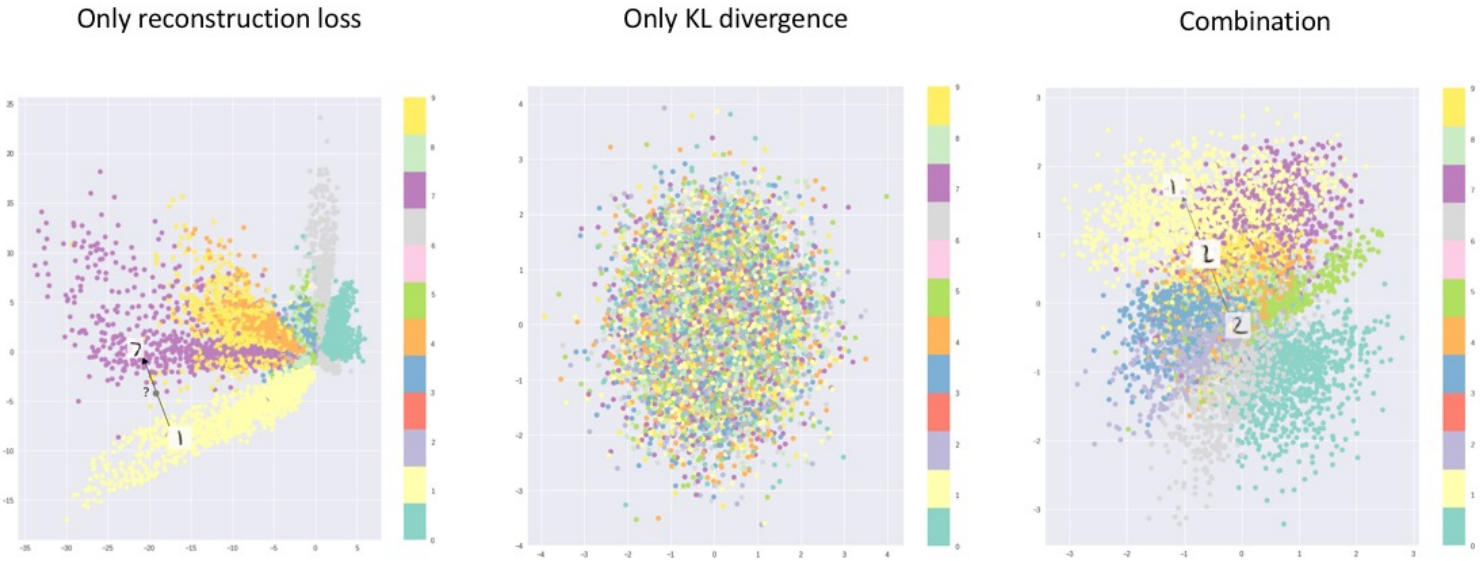
Regularization term (KL)



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

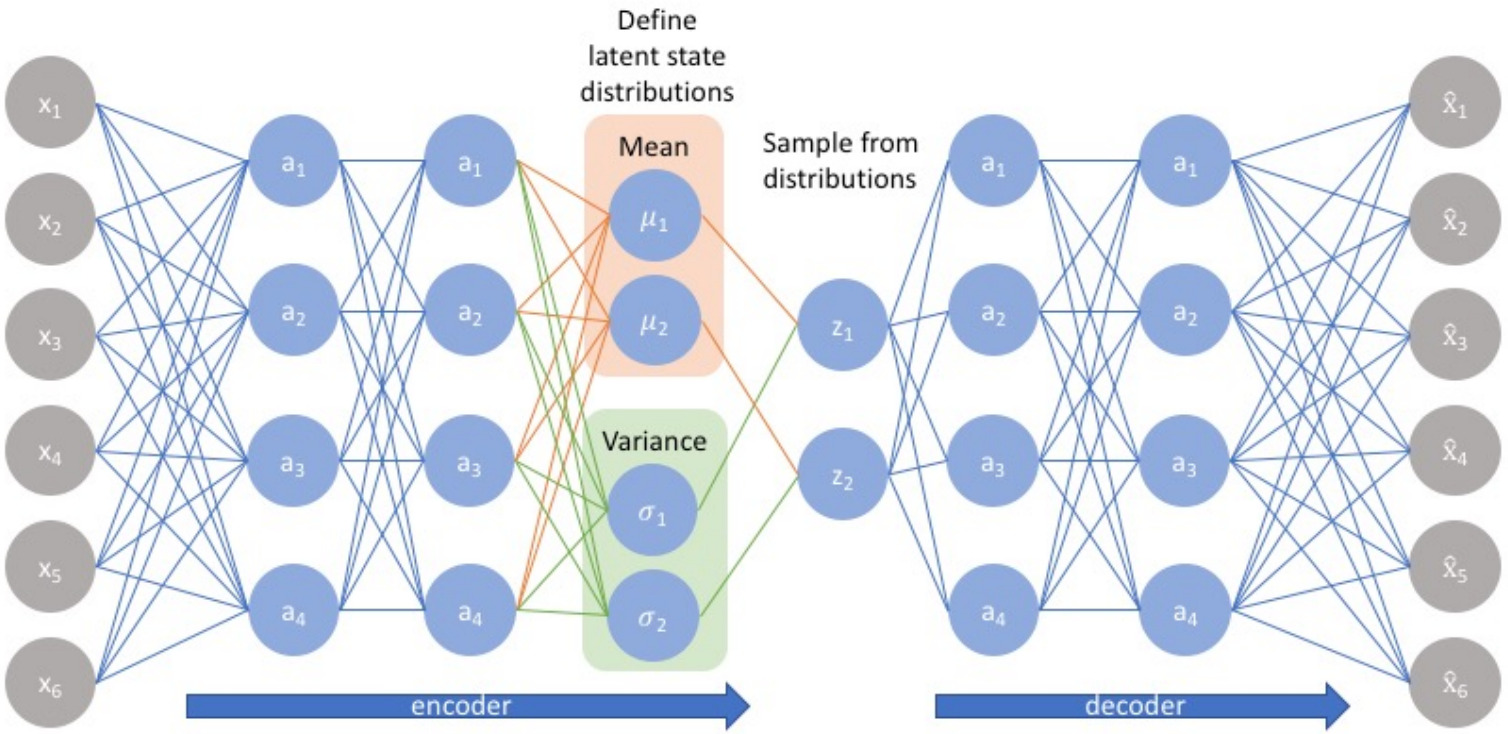
Hidden space representation

- Only reconstruction loss
  - uneven distribution, can not learn a smooth representation of hidden state
- Only KL divergence loss
  - Does not learn the representation of the original data, only memorize the input data



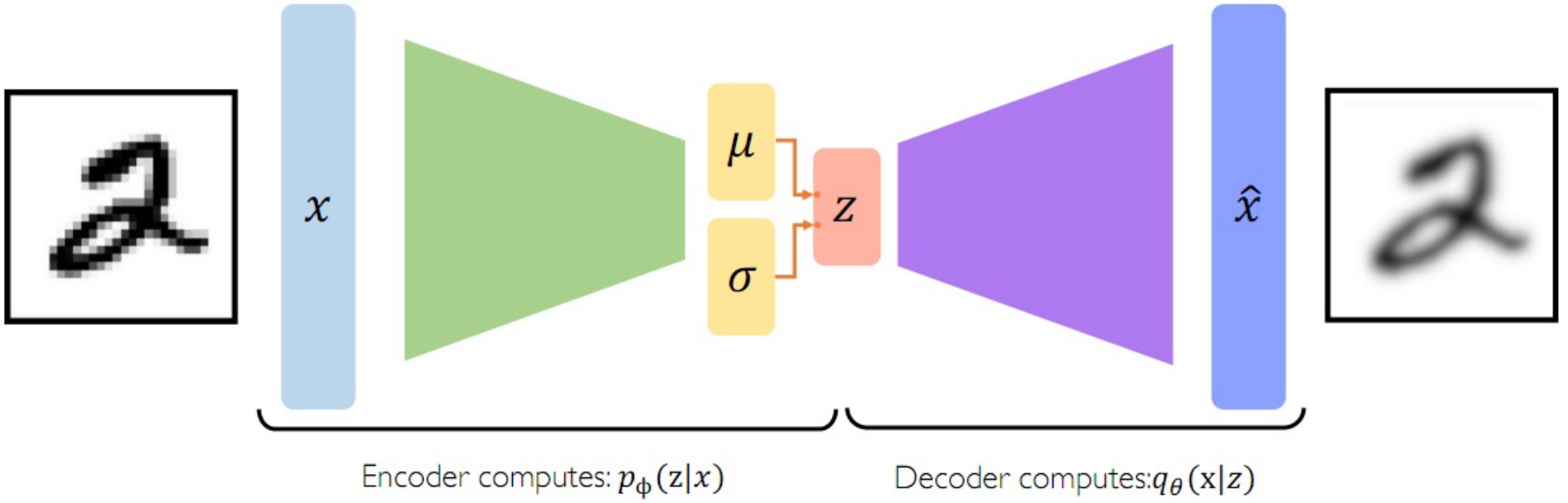
VAE implementation

- With the assumption that the hidden state distribution is a Gaussian distribution
- The output of the Encoder is two vectors describing the mean and the variance of the hidden state distribution
- Decoder takes random samples from hidden state and decodes



VAE computation graph

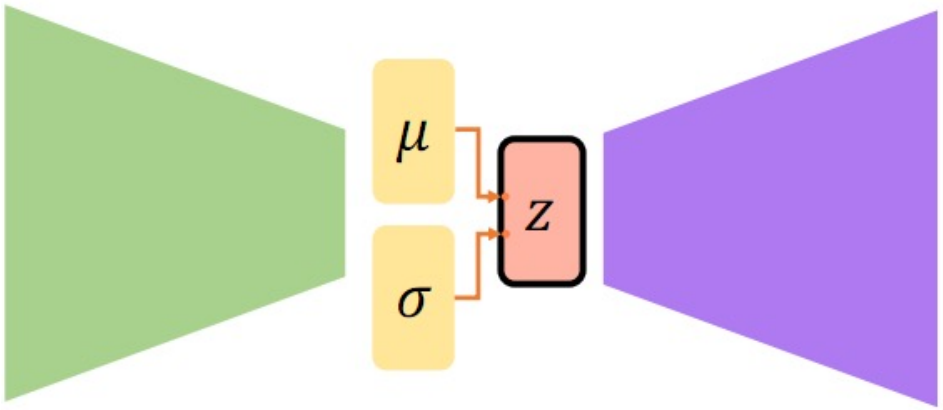
- Problem: Cannot back-propagate through the sampling layer!



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

Reparameterization trick

- Randomly sample ε from a unit Gaussian, and then shift the randomly sampled ε by the latent distribution's mean μ and scale it by the latent distribution's variance σ.



Key Idea:

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

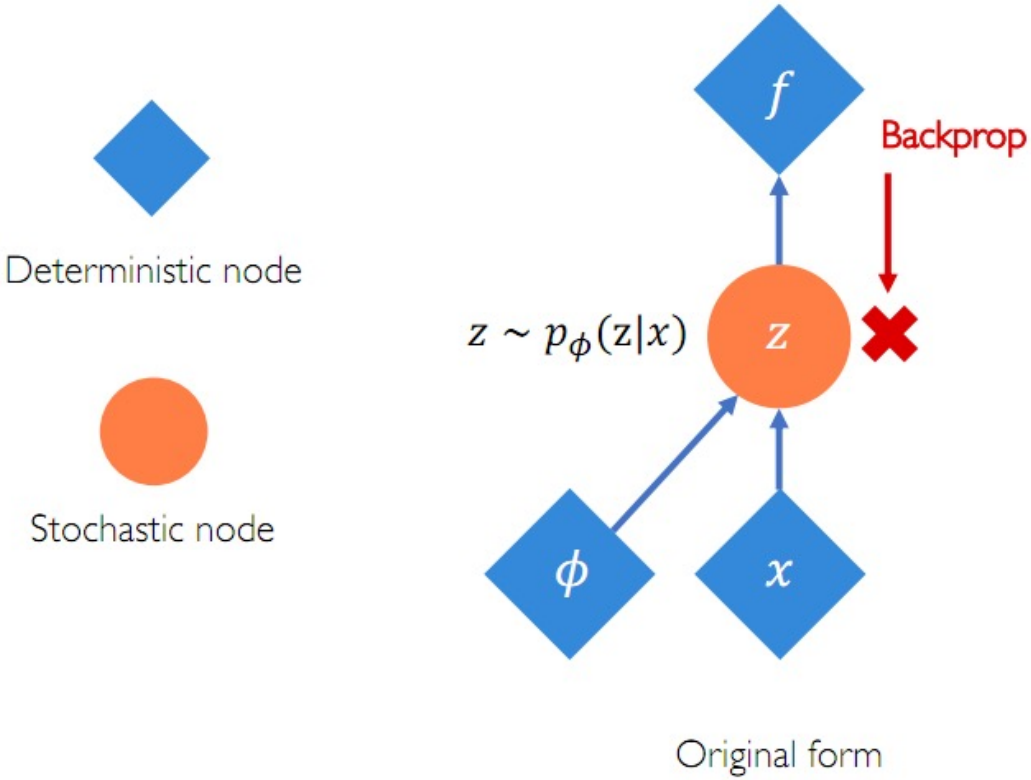
Consider the sampled latent vector as a sum of

- a fixed μ vector,
- and fixed σ vector, scaled by random constants drawn from the prior distribution

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

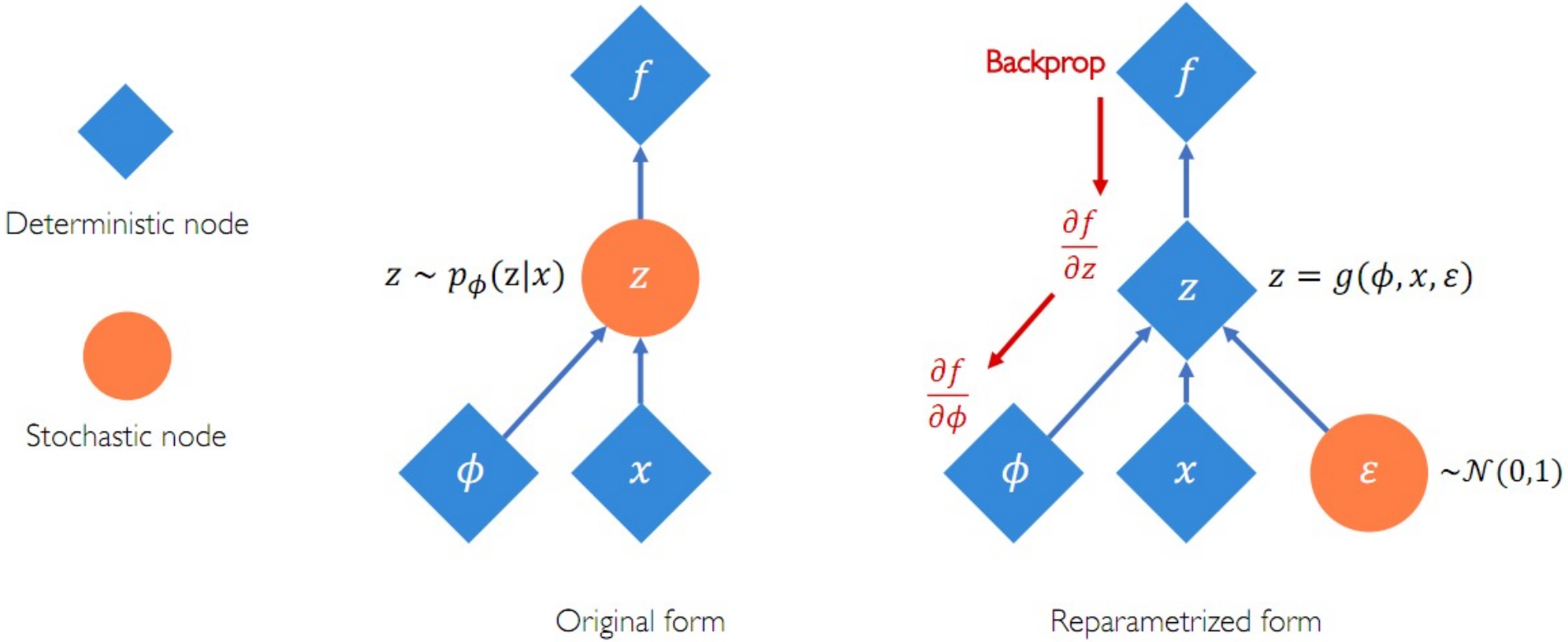
where $\varepsilon \sim \mathcal{N}(0,1)$

Reparameterization trick (2)



Original form

Reparameterization trick (3)

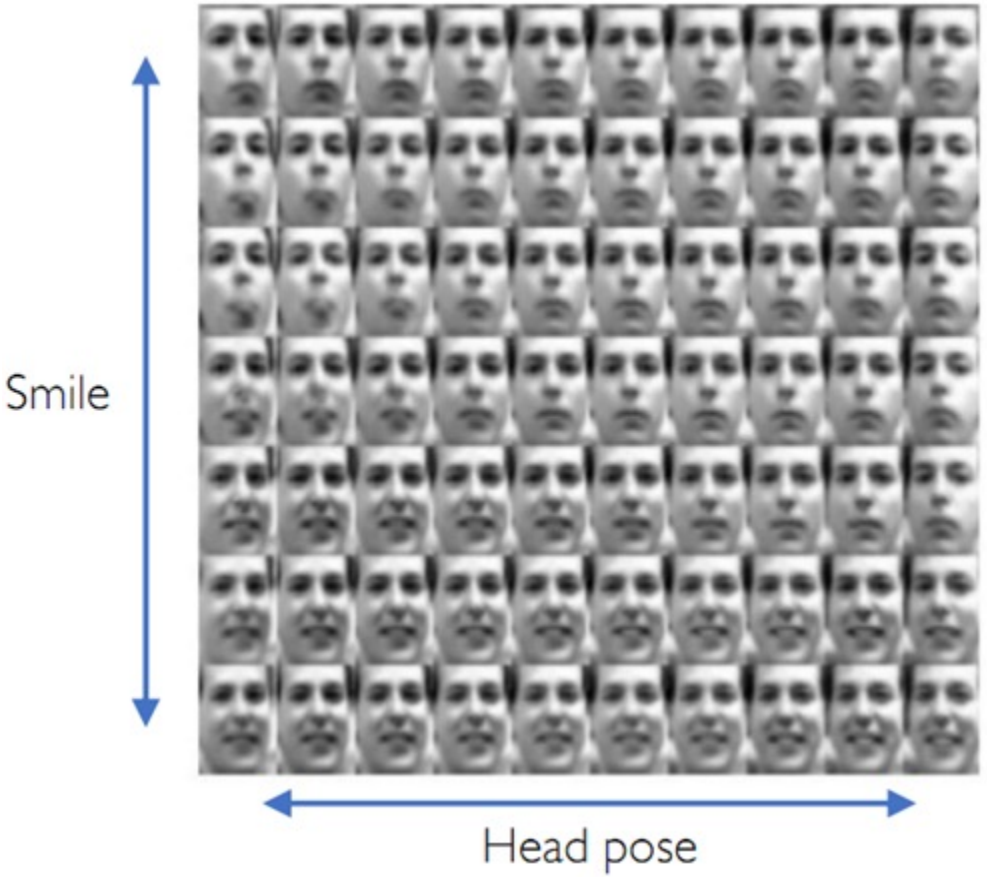


VAEs: hidden space transformation

- Slowly increase or decrease one hidden variable, keeping other variables fixed
- Each dimension of z encodes meaningful hidden features

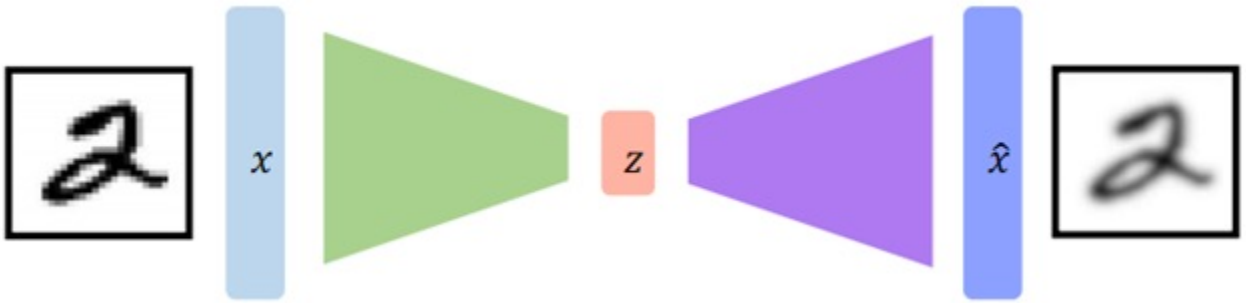
VAEs: hidden space transformation (2)

- Ideally, the hidden variables are completely independent of each other (uncorrelated).
- It is possible to add a cross-matrix constraint to force the hidden variables to be independent of each other (the correlation coefficient between different hidden variables is approximately 0)
- Disentanglement



VAEs: Summary

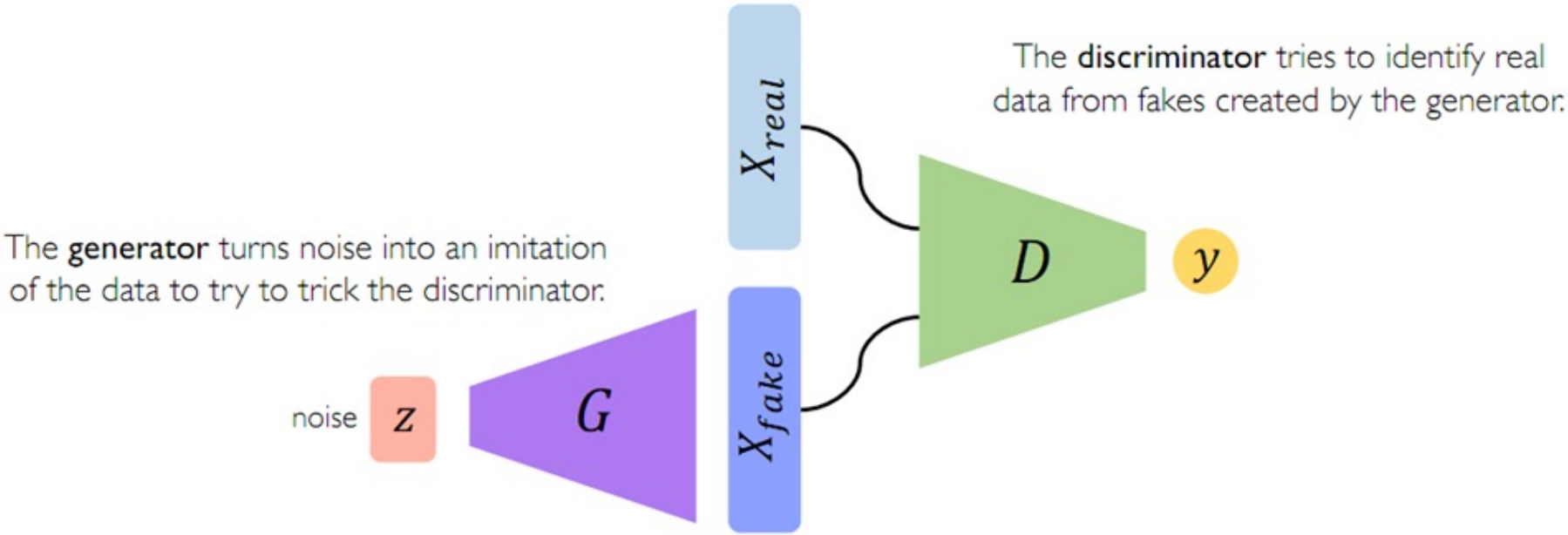
1. Reproducible objective function allows training without labels (unsupervised)
2. Using re-parameterization to enable end-to-end training
3. Interpret hidden variables by varying their values and observing
4. Generate new data



Generative adversarial networks (GANs)

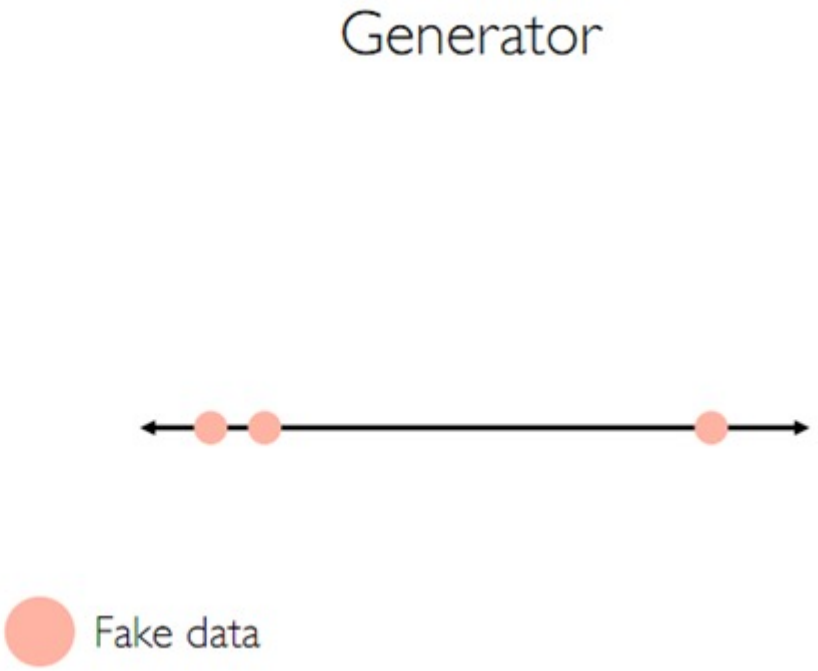
Generative adversarial networks

- GAN is a generative model containing two opposing neural networks
- The generator network turns a noise vector into an imitation of the data to fool the discriminator.
- The classifier network tries to distinguish between the real data and the fake data generated by the generator



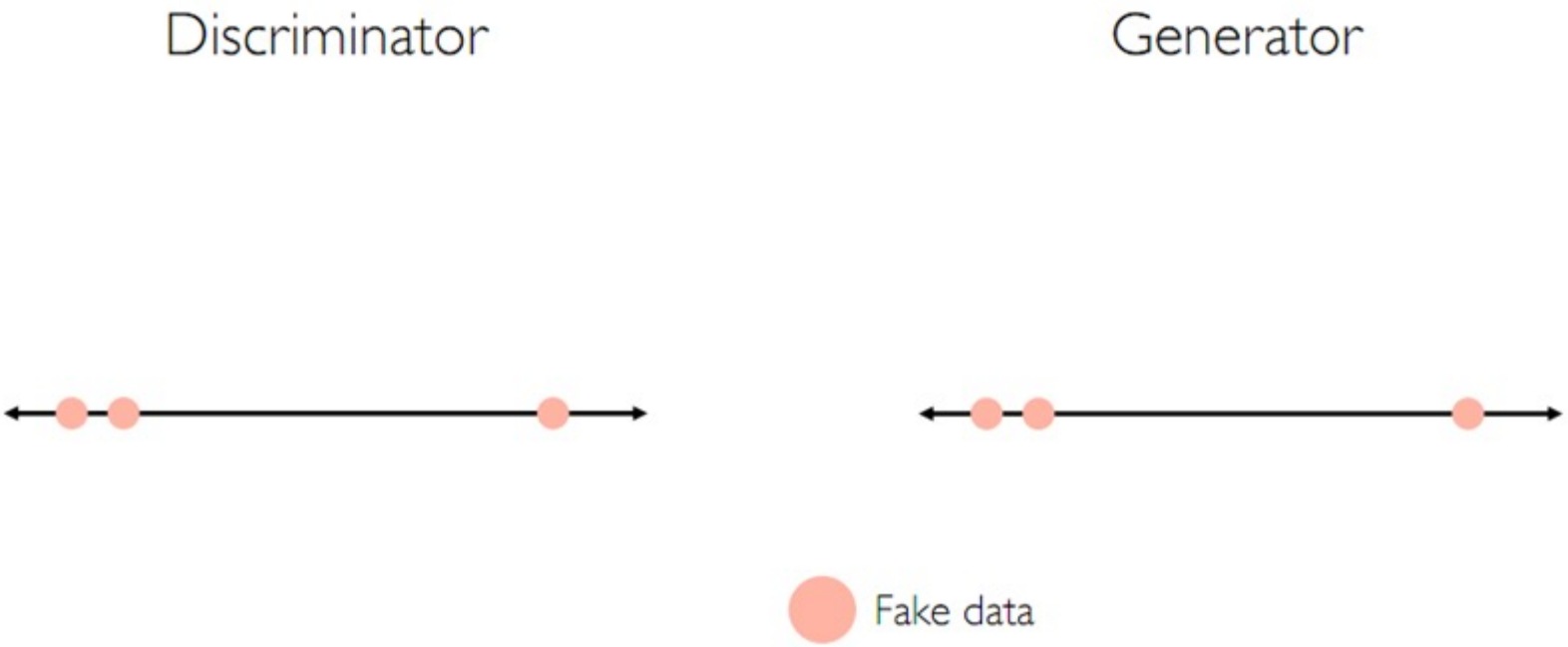
Initial intuition about GANs

- Generator generates fake data from noise



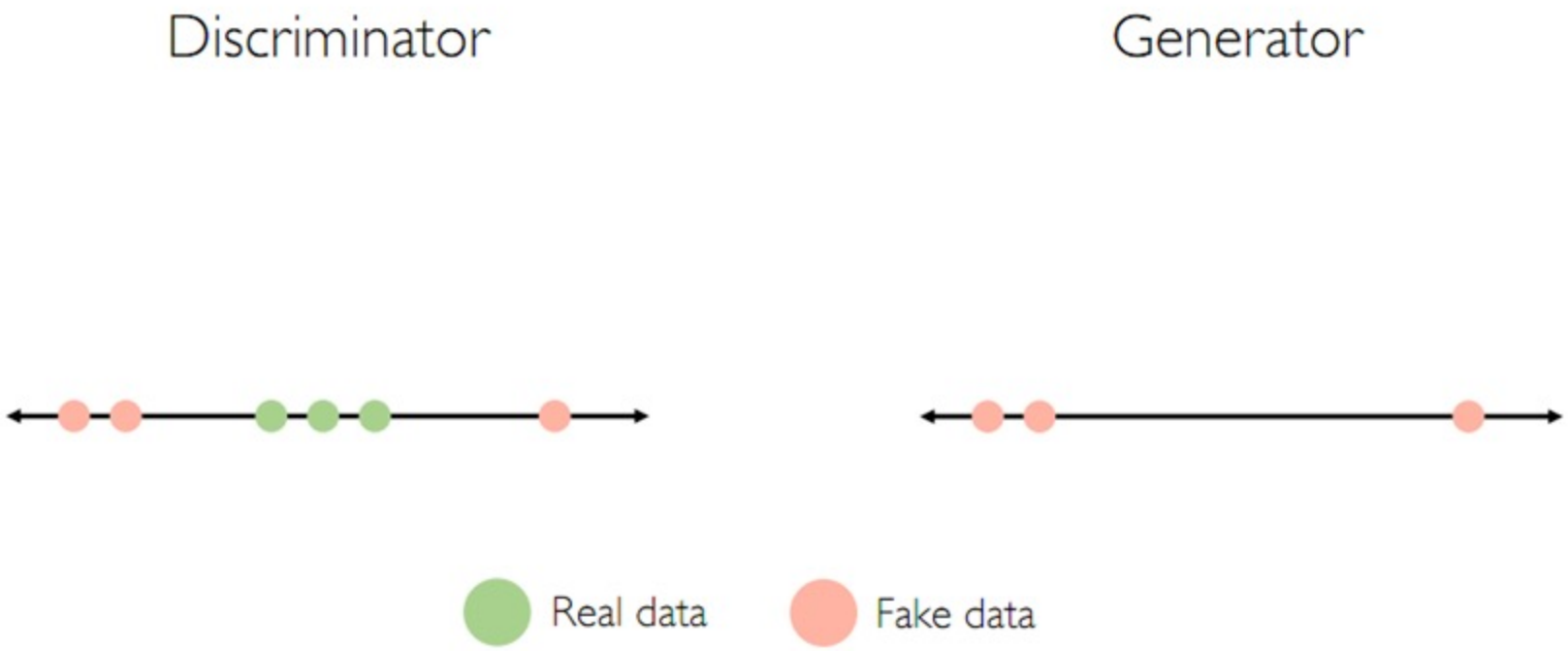
Initial intuition about GANs (2)

- Discriminator looks at real and fake data



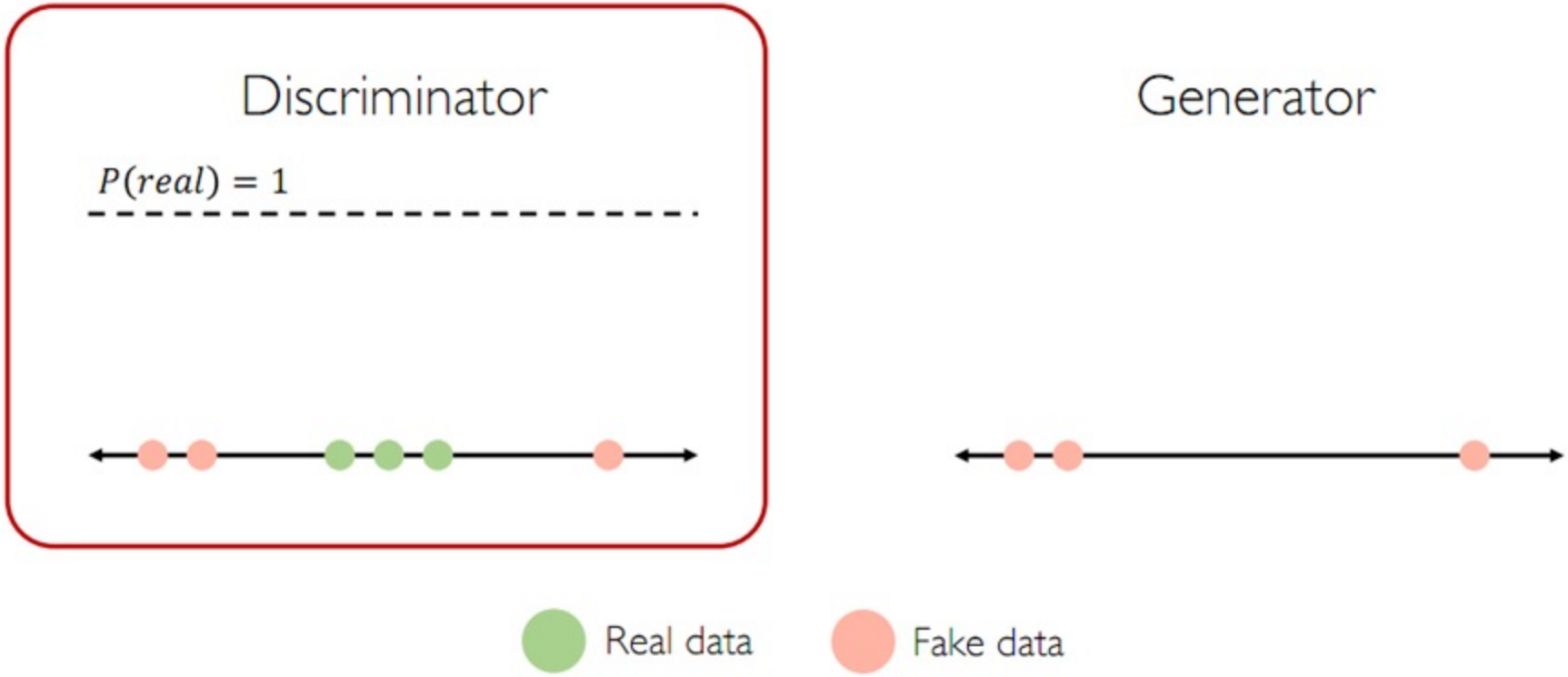
Initial intuition about GANs (3)

- Discriminator looks at real and fake data



Initial intuition about GANs (4)

- Discriminator predict what's real and what's fake

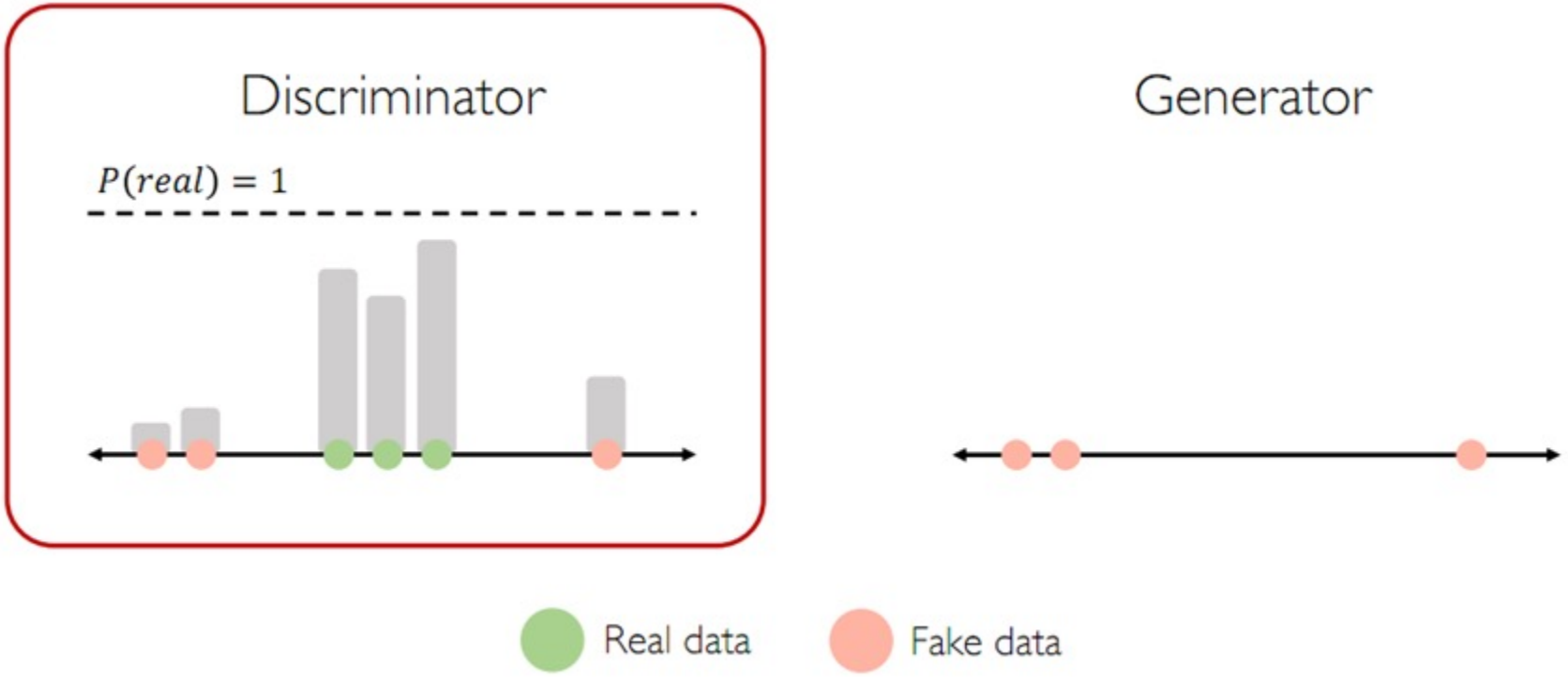


Initial intuition about GANs (5)

- Discriminator predict what's real and what's fake

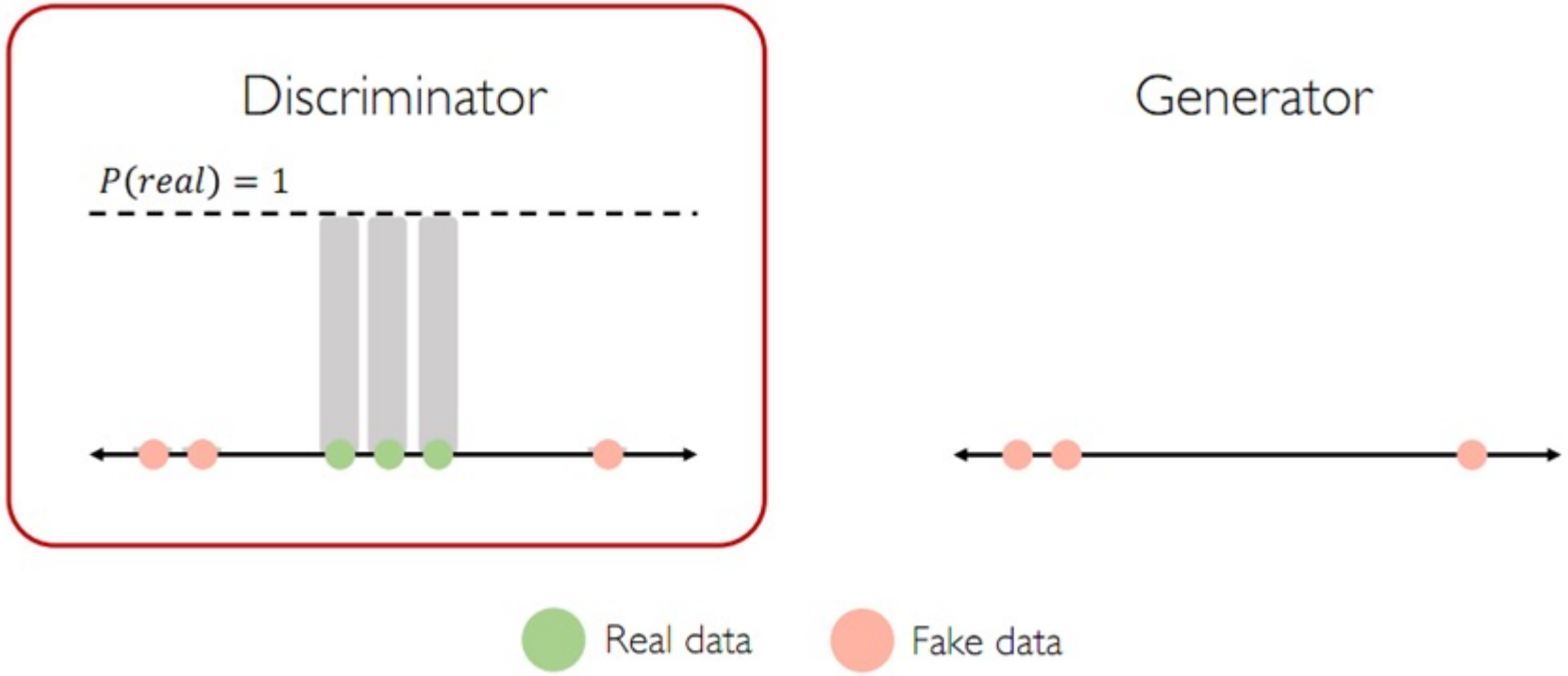
Initial intuition about GANs (6)

- Discriminator predict what's real and what's fake



Initial intuition about GANs (7)

- Discriminator predict what's real and what's fake

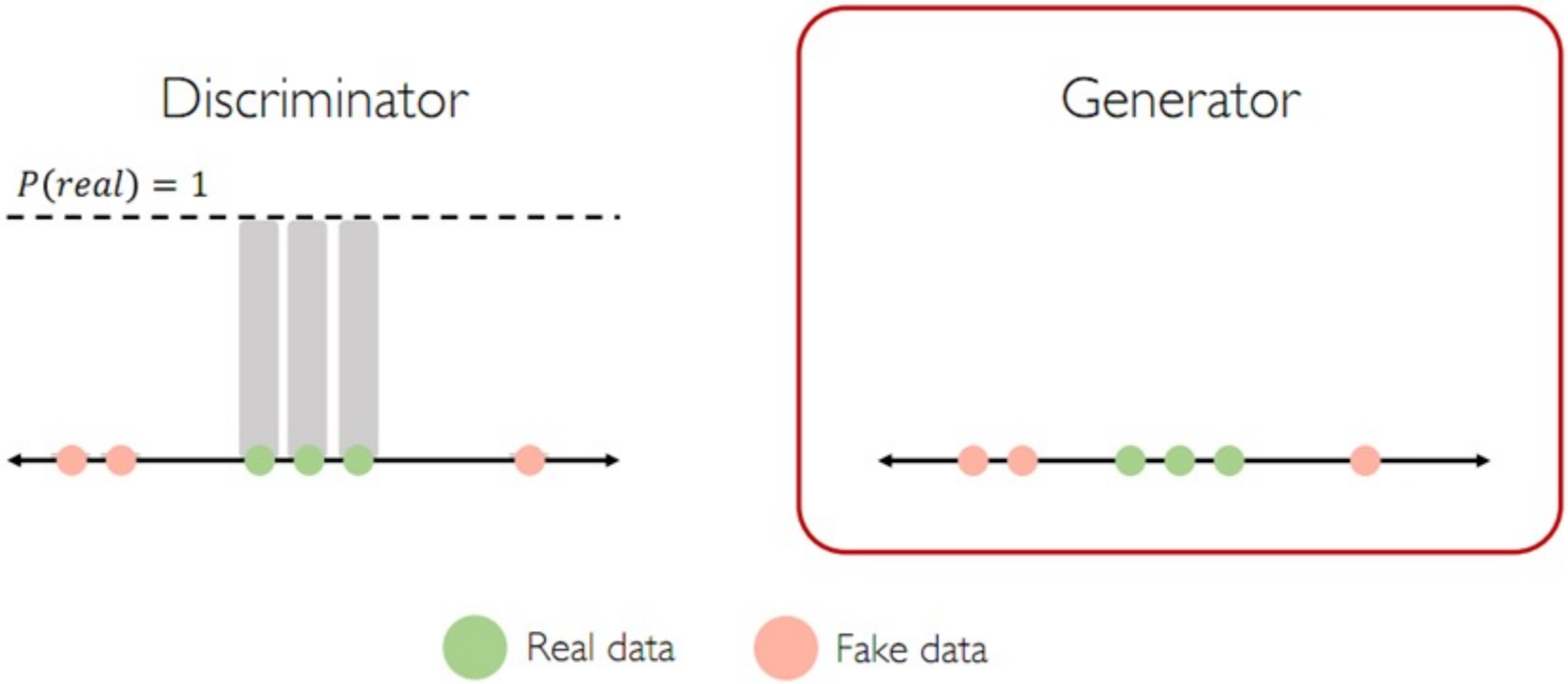


Initial intuition about GANs (8)

- Generator strives to improve the quality of fake data

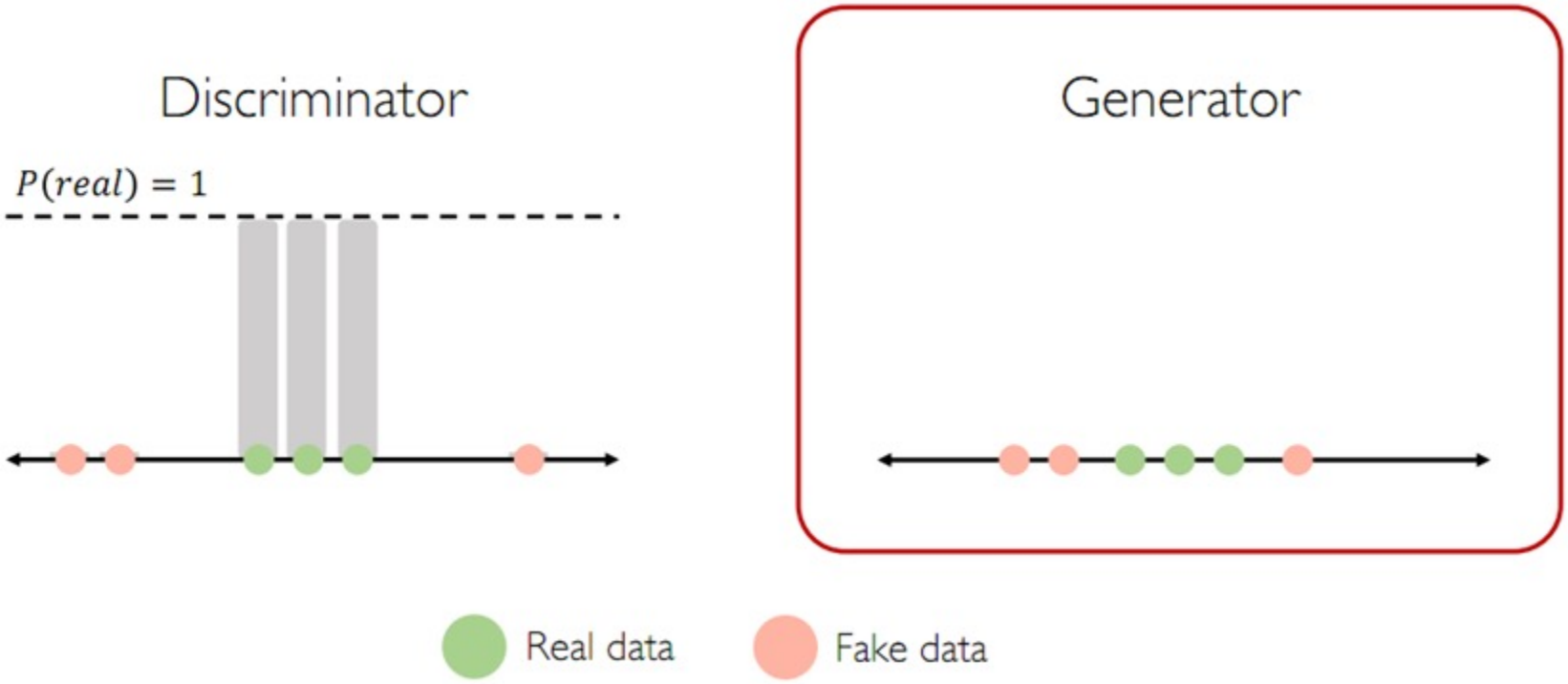
Initial intuition about GANs (9)

- Generator strives to improve the quality of fake data



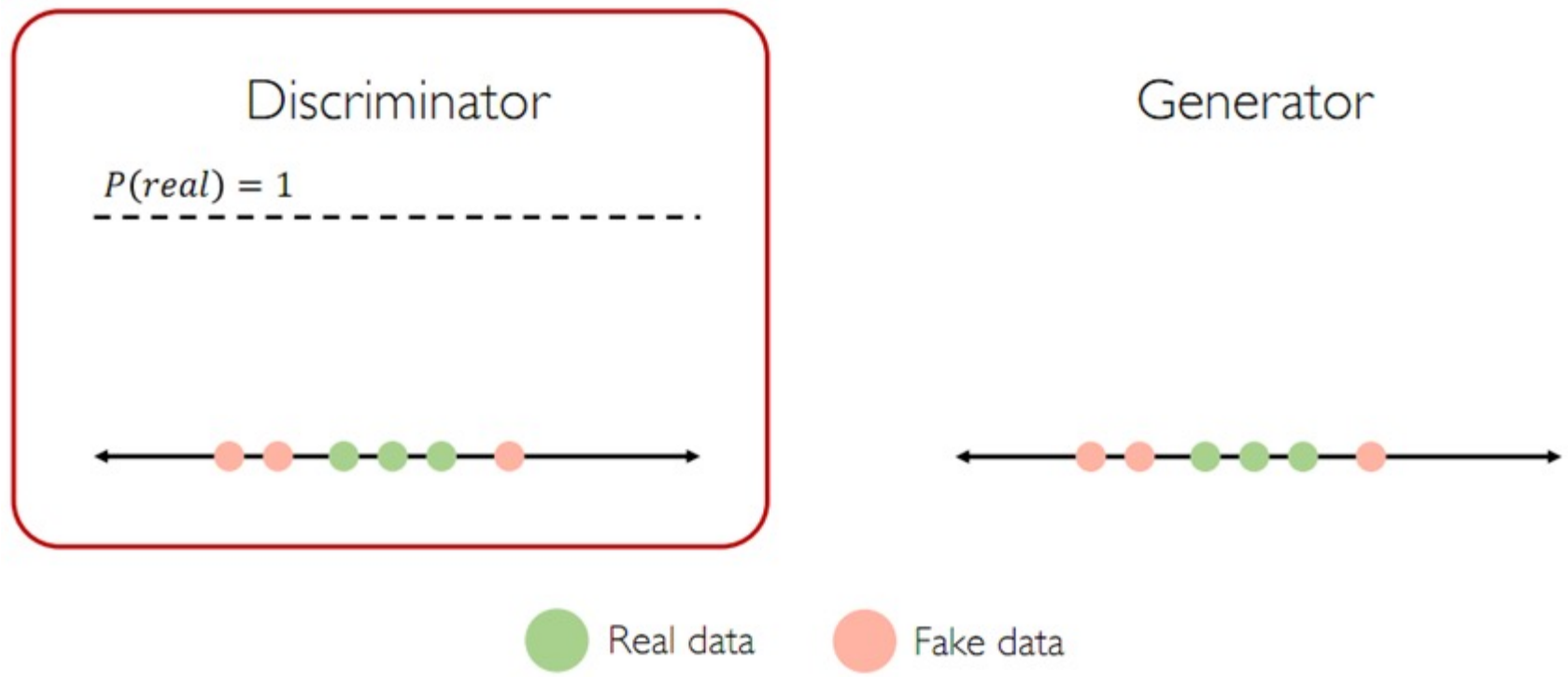
Initial intuition about GANs (10)

- Generator strives to improve the quality of fake data



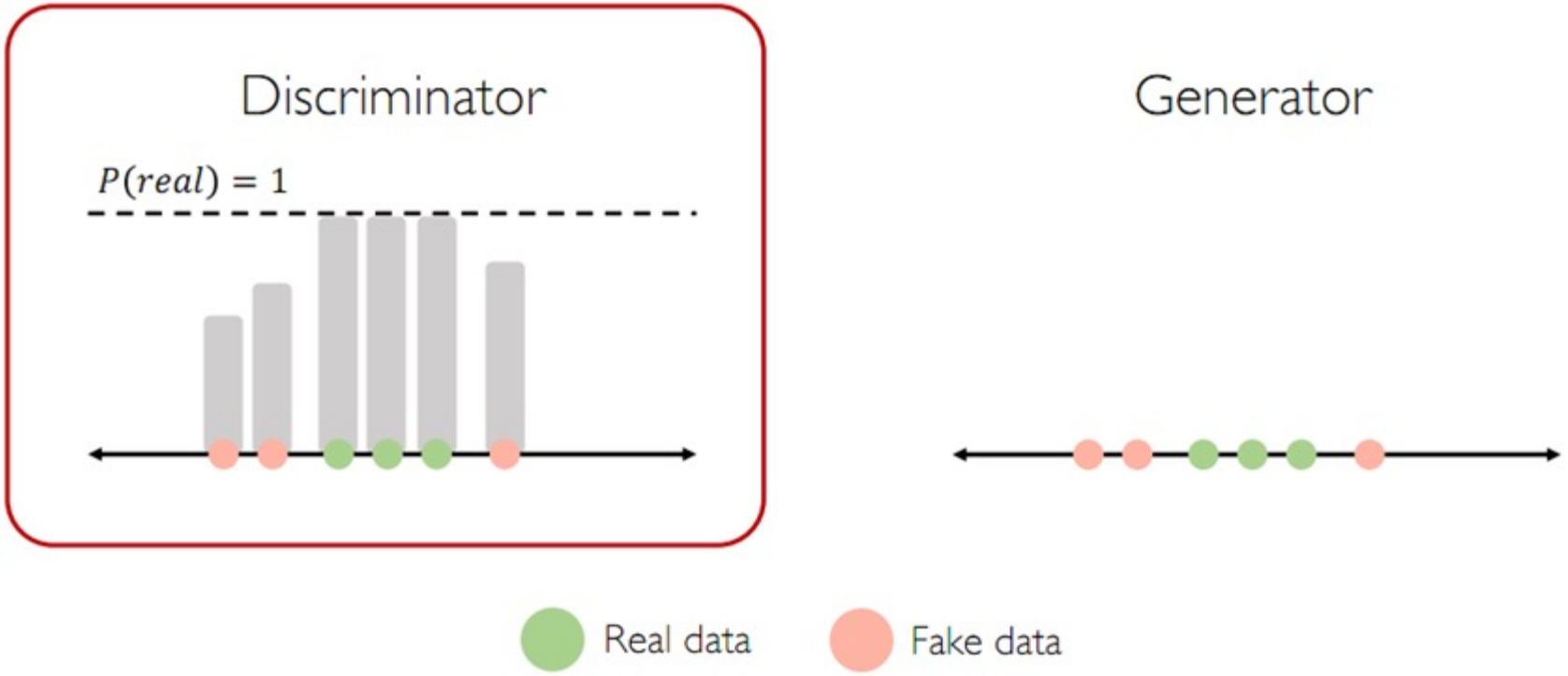
Initial intuition about GANs (11)

- Discriminator predict what's real and what's fake



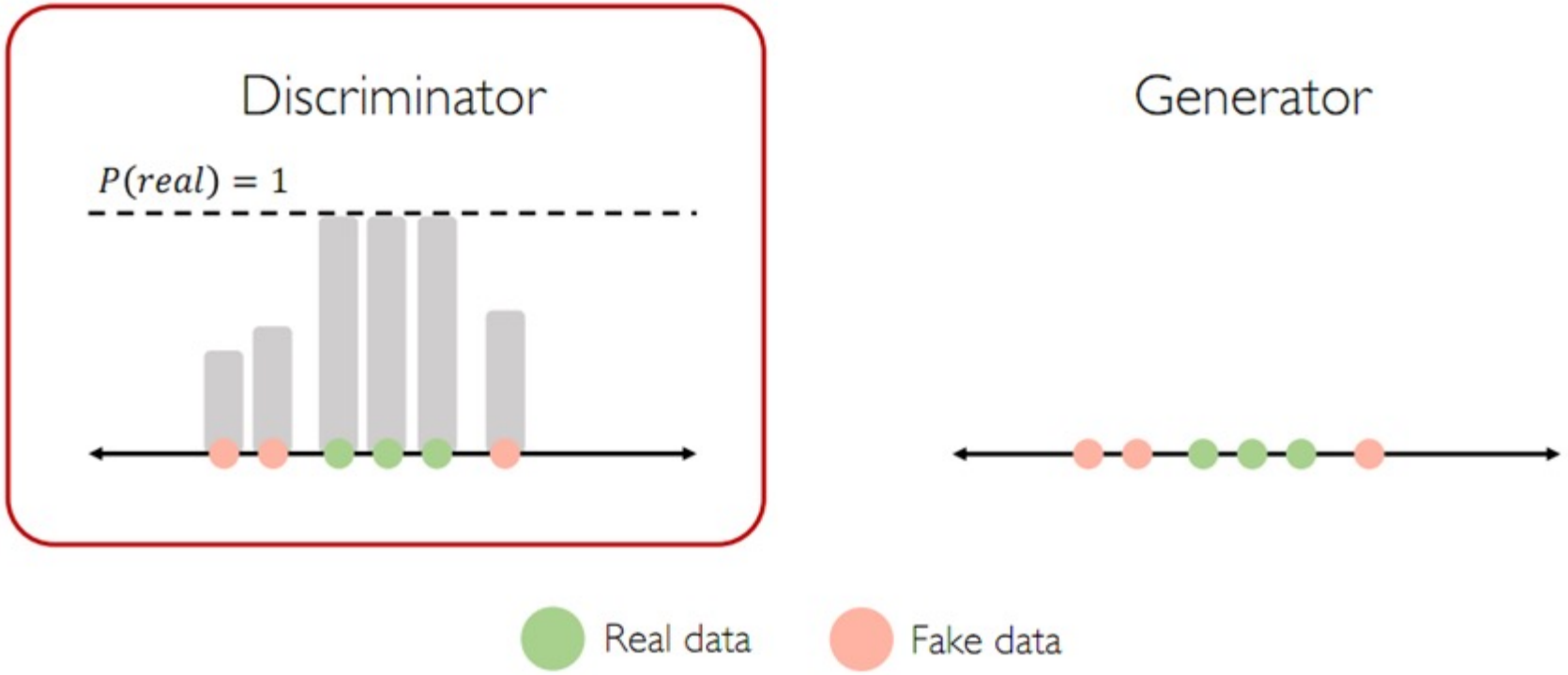
Initial intuition about GANs (12)

- Discriminator predict what's real and what's fake



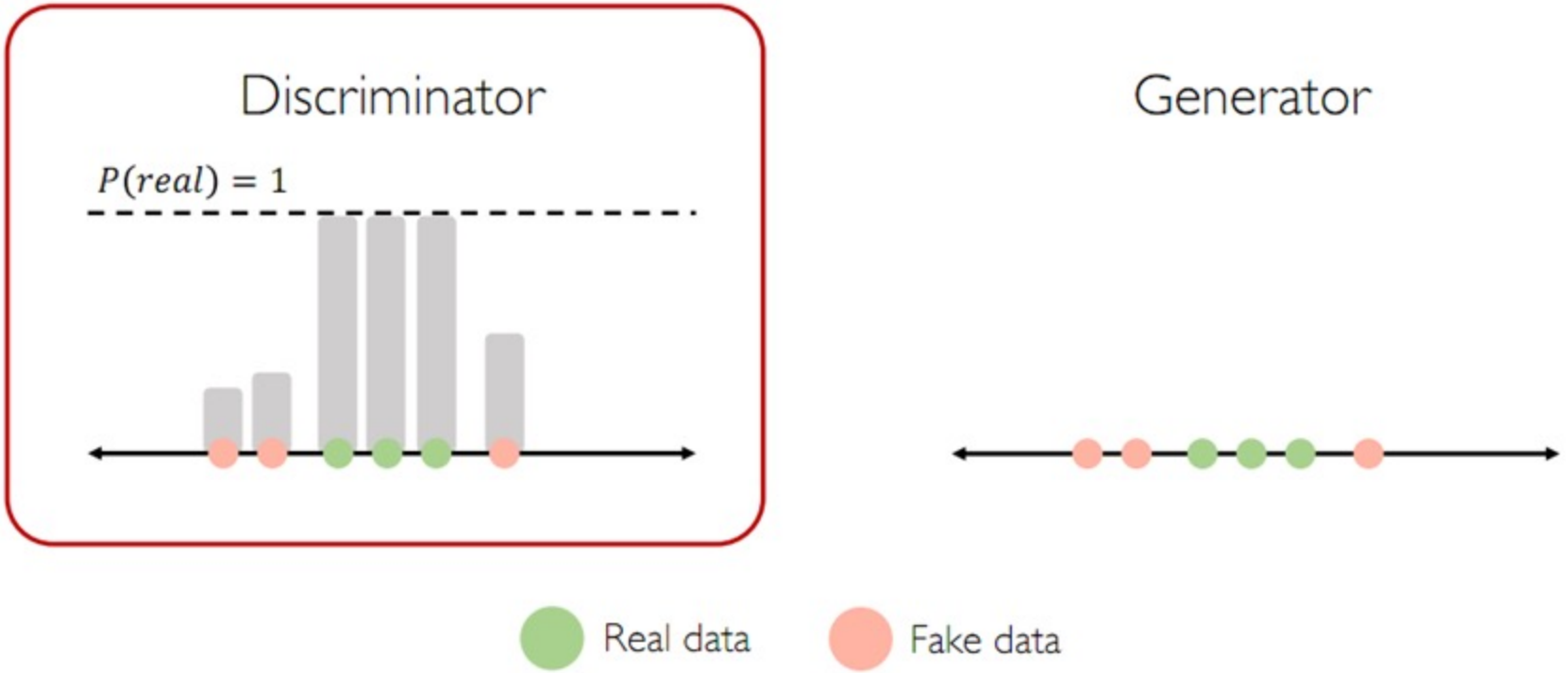
Initial intuition about GANs (13)

- Discriminator predict what's real and what's fake



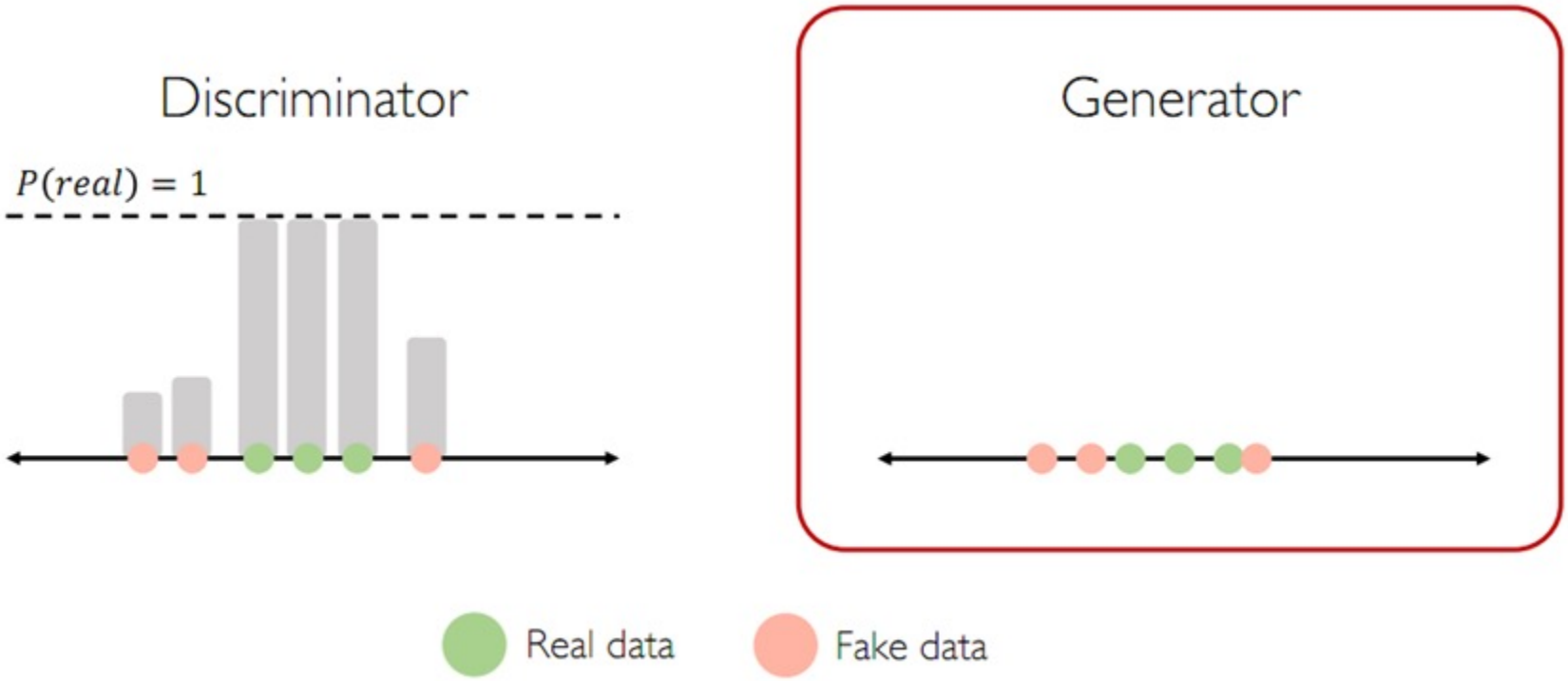
Initial intuition about GANs (14)

- Discriminator predict what's real and what's fake



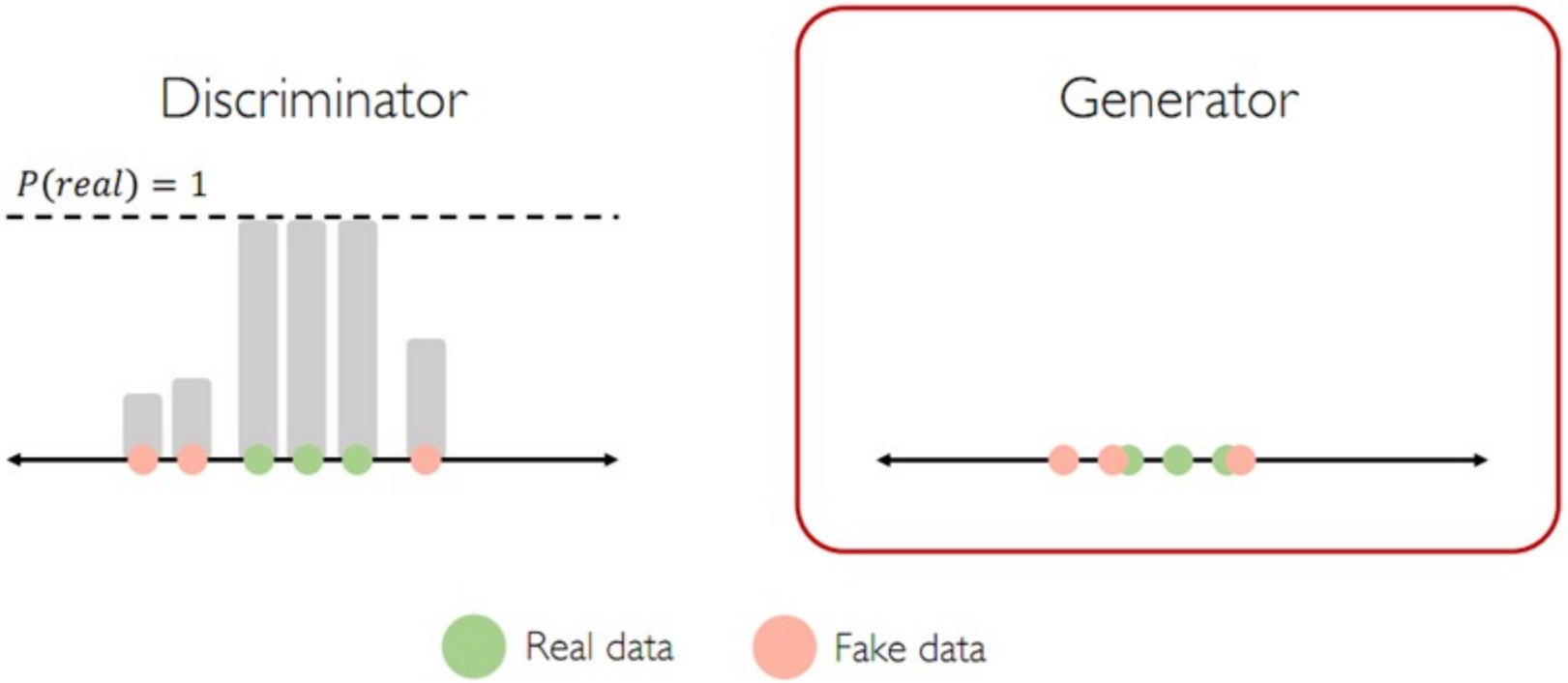
Initial intuition about GANs (15)

- Generator strives to improve the quality of fake data



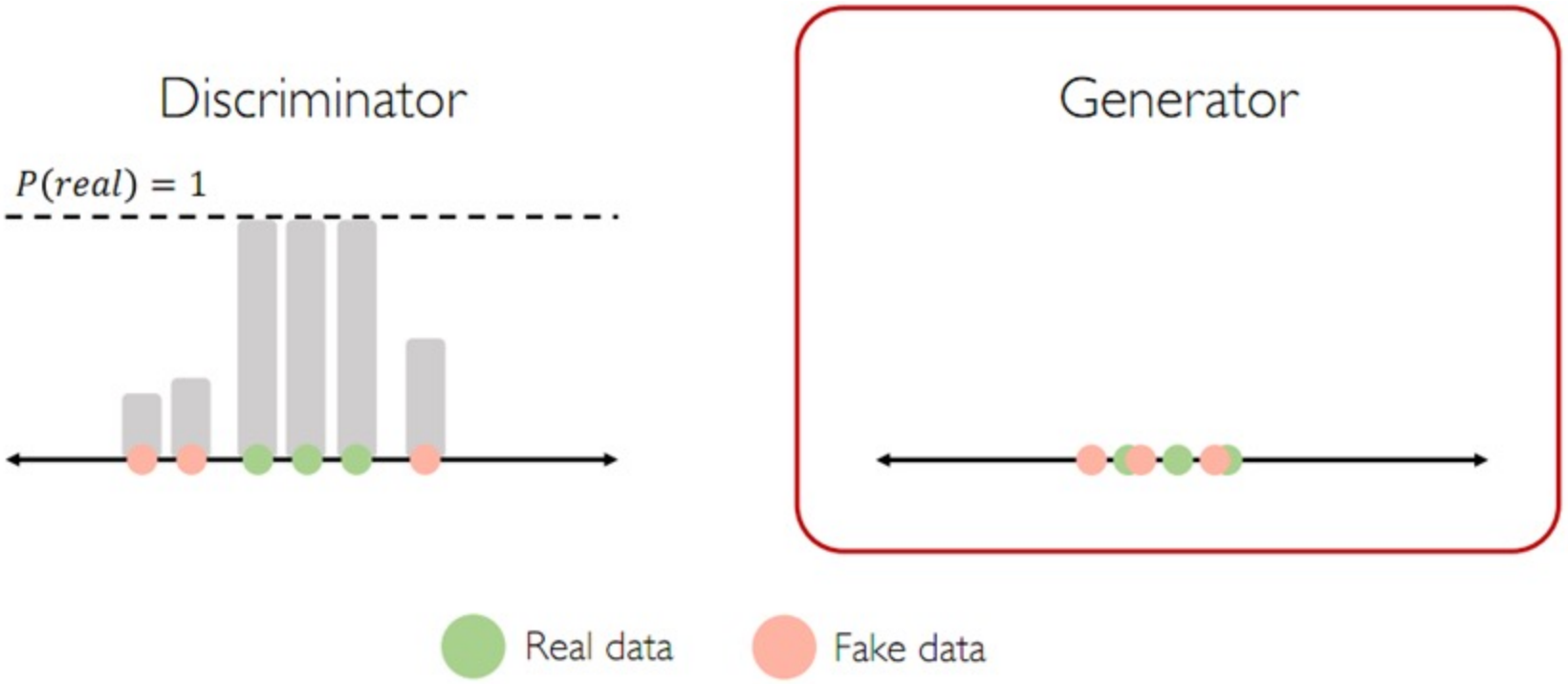
Initial intuition about GANs (16)

- Generator strives to improve the quality of fake data



Initial intuition about GANs (17)

- Generator strives to improve the quality of fake data



Initial intuition about GANs (18)

- Discriminator tries to distinguish real and fake data
- Generator tries to improve the quality of fake data to trick discriminator

Training GANs

- Discriminator tries to distinguish real and fake data
- Generator tries to improve the quality of fake data to trick 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.

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

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

  end for

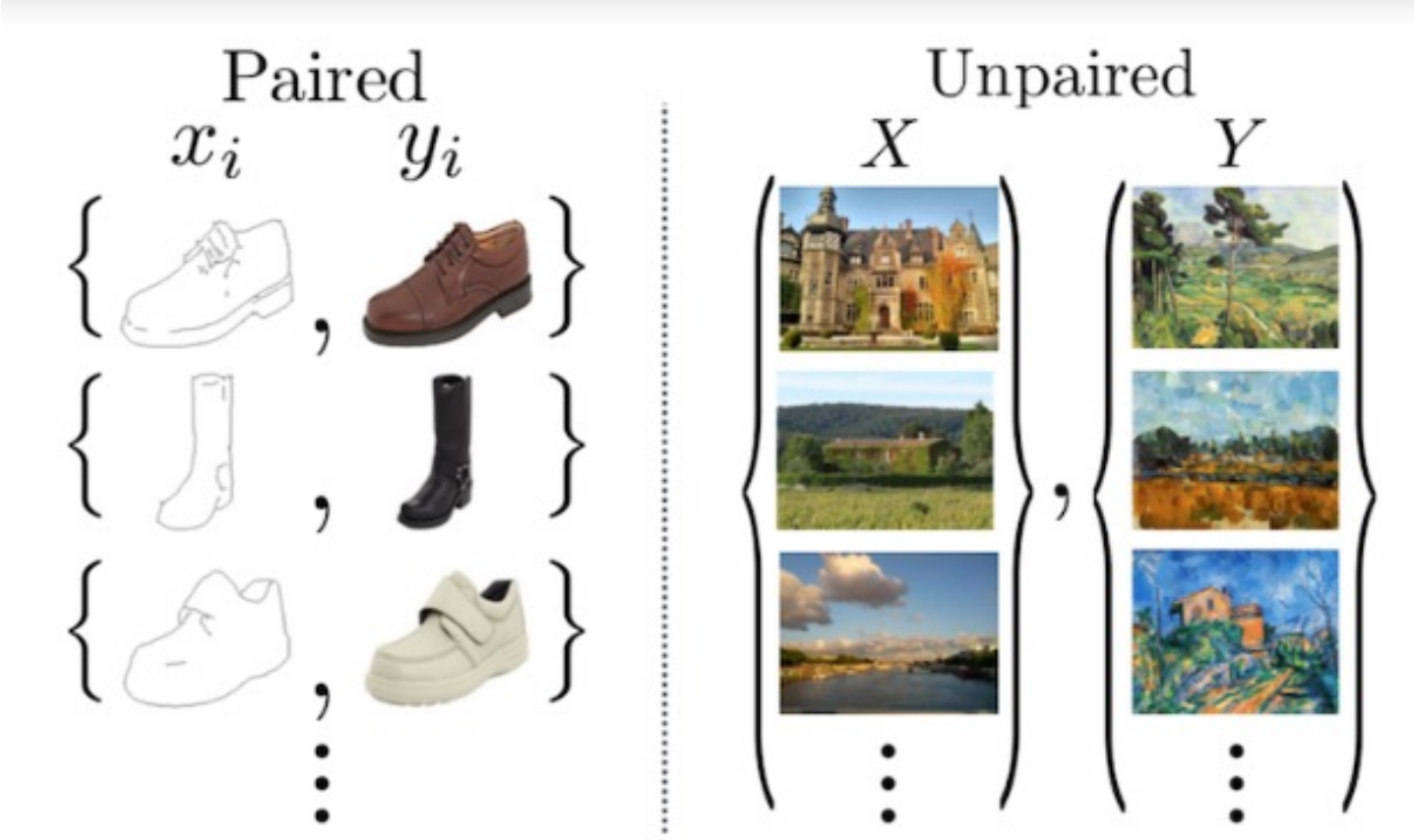
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

end for

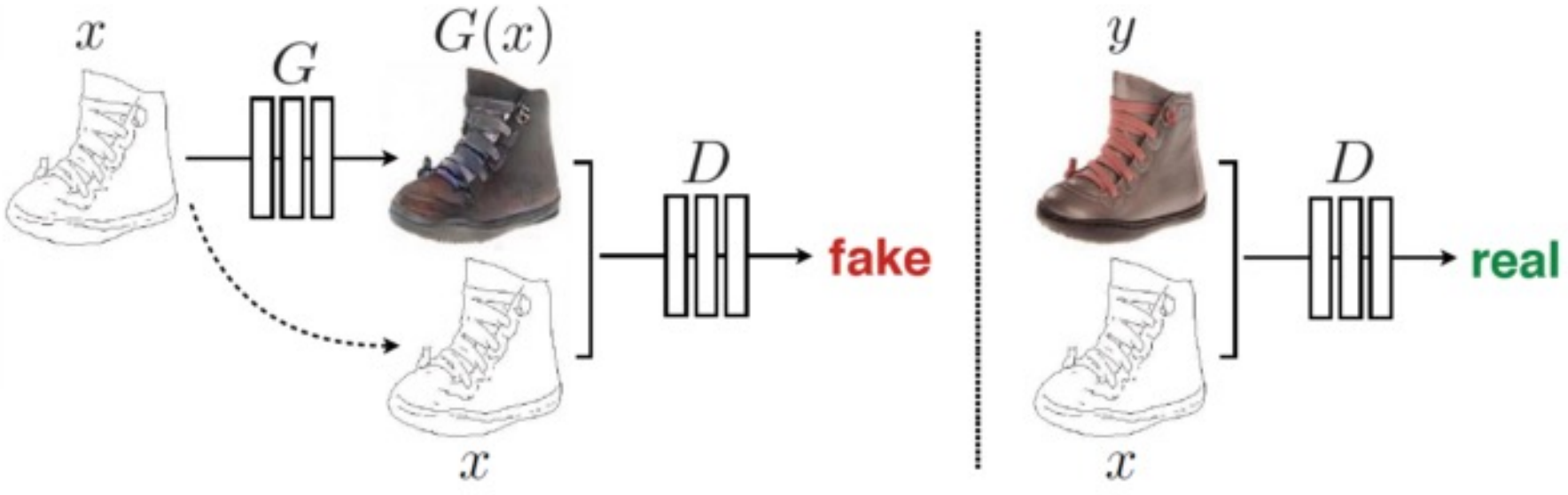
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Image-to-image transformation

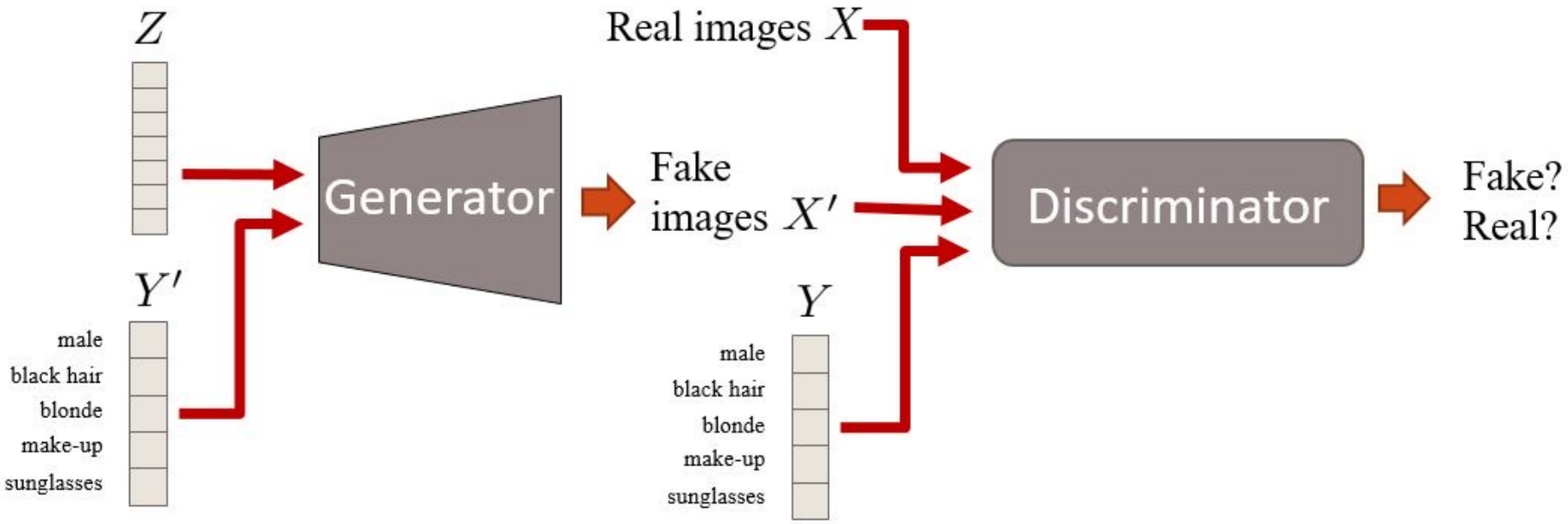


pix2pix

- 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

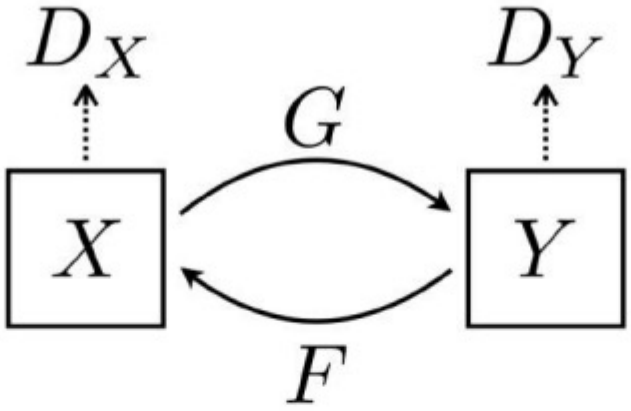


Conditional GAN

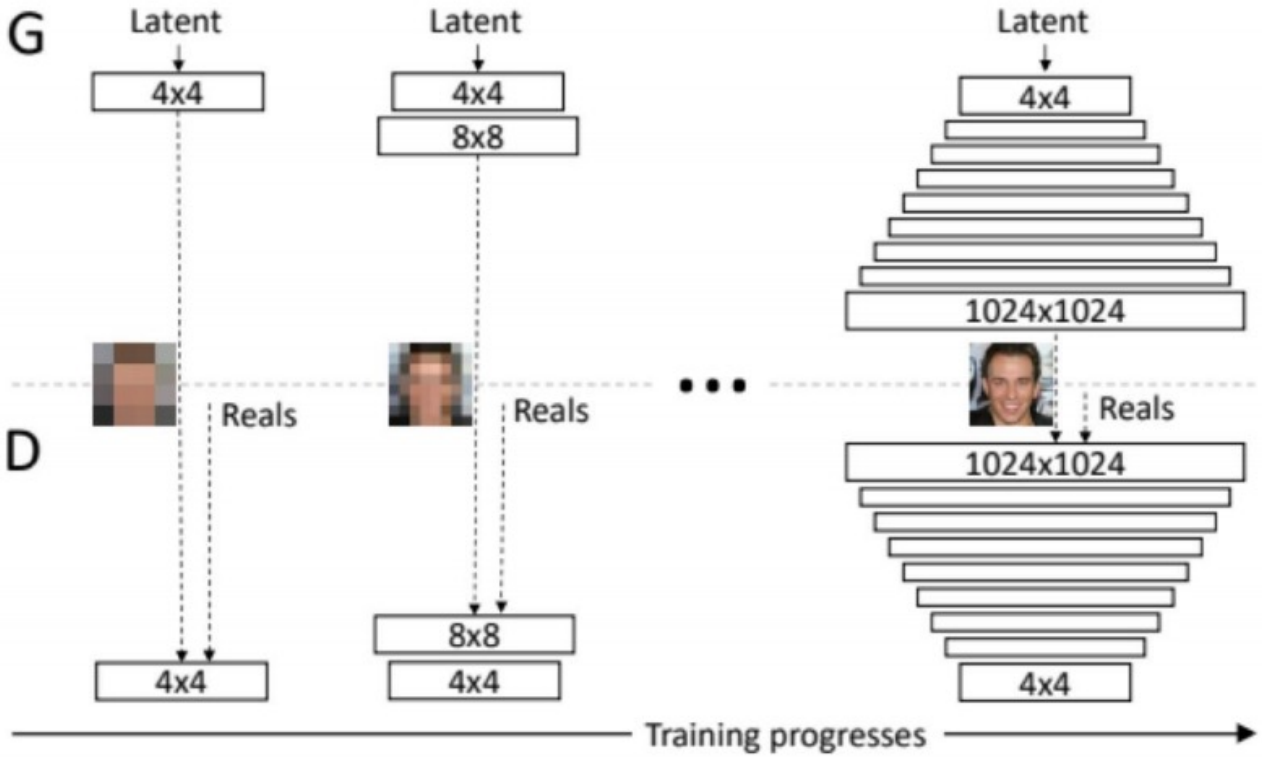


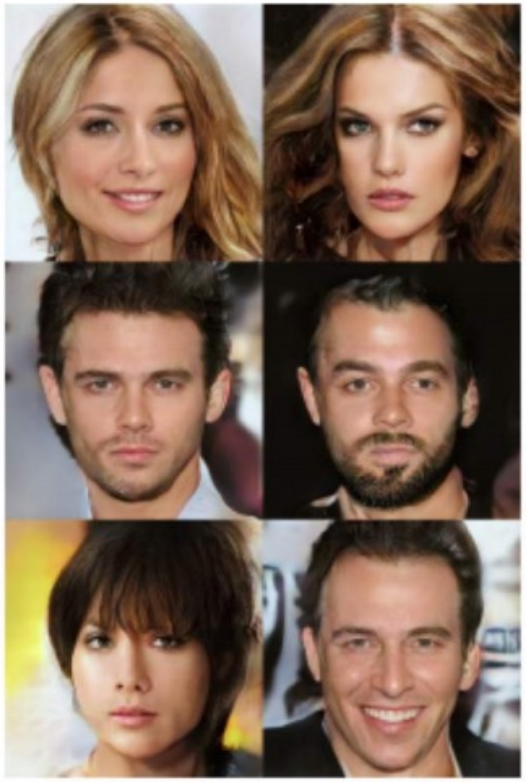
CycleGAN

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

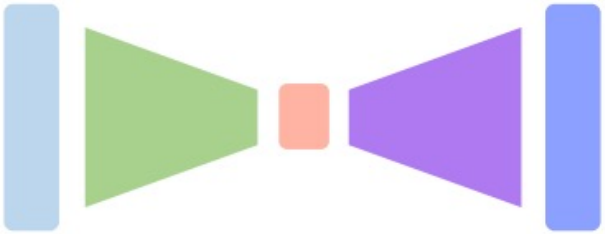
Karras et al., Arxiv 2018.

Style-based transfer

Summary

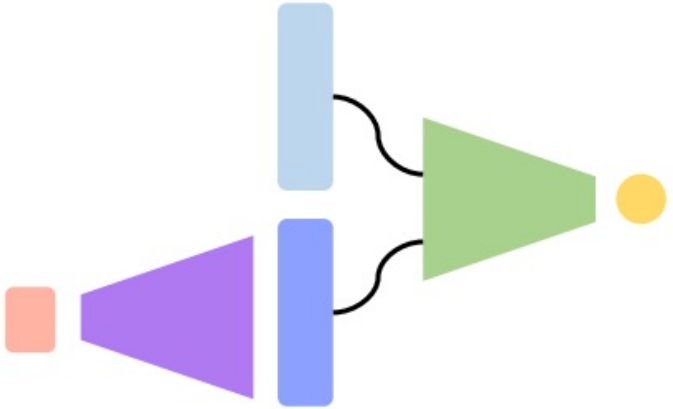
Autoencoders and Variational Autoencoders (VAEs)

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



Generative Adversarial Networks (GANs)

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



References

1. MIT Deep Learning 6.S191:

http://introtodeeplearning.com/

2. Stanford cs231n :

http://cs231n.stanford.edu/slides/2020/lecture_11.pdf

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!!!

soict.hust.edu.vn/ fb.com/groups/soict