Deep Generative Models

Supervised vs unsupervised learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$$x \rightarrow y$$

Examples: Classification, regression, object detection, semantic segmentation, etc.

Unsupervised Learning

Data: x
x is data, no labels!

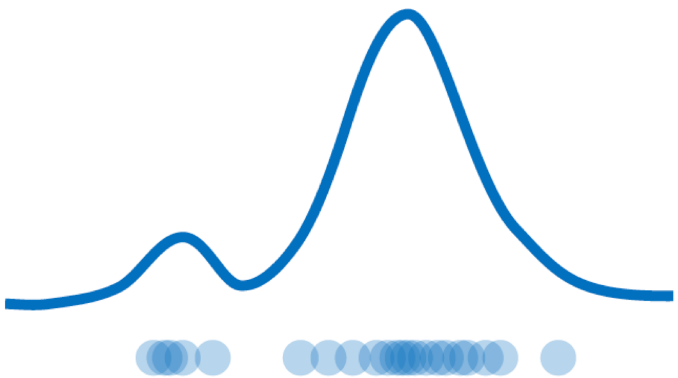
Goal: Learn some *hidden* or *underlying structure* of the data

Examples: Clustering, feature or dimensionality reduction, etc.

Generative modeling

Goal: Take as input training samples from some distribution and learn a model that represents that distribution

Density Estimation

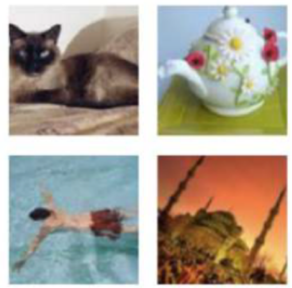


samples

Sample Generation

Input samples

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



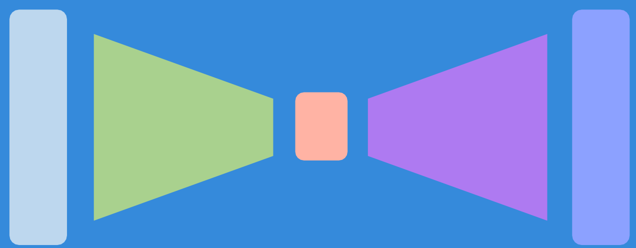
Generated samples

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

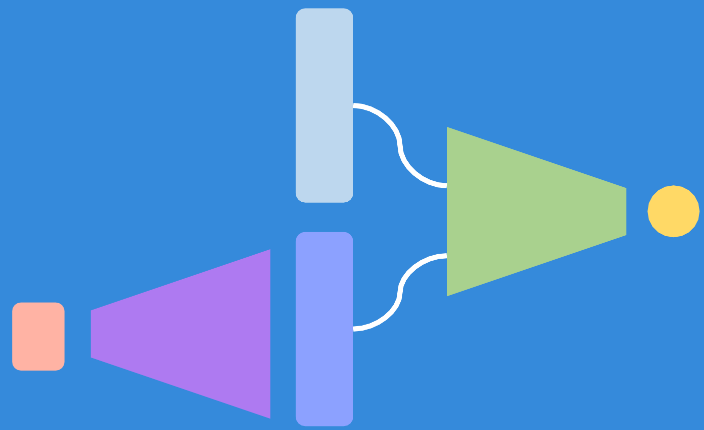
How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?

Latent variable models

Autoencoders and Variational Autoencoders (VAEs)



Generative Adversarial Networks (GANs)



What is a latent variable?

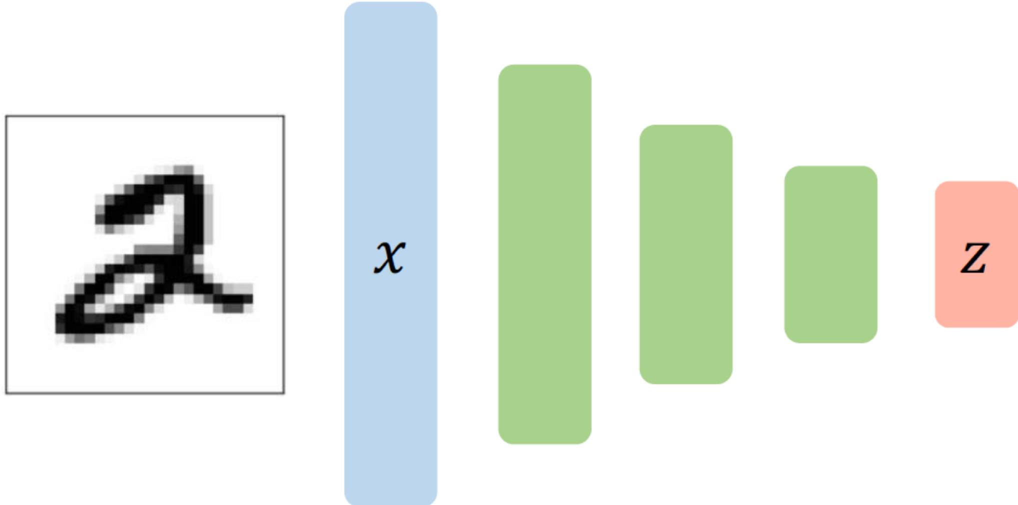
Myth of the Cave

Can we learn the true explanatory factors, e.g. latent variables, from only observed data?

Autoencoders

Autoencoders: background

Unsupervised approach for learning a **lower-dimensional** feature representation from unlabeled training data



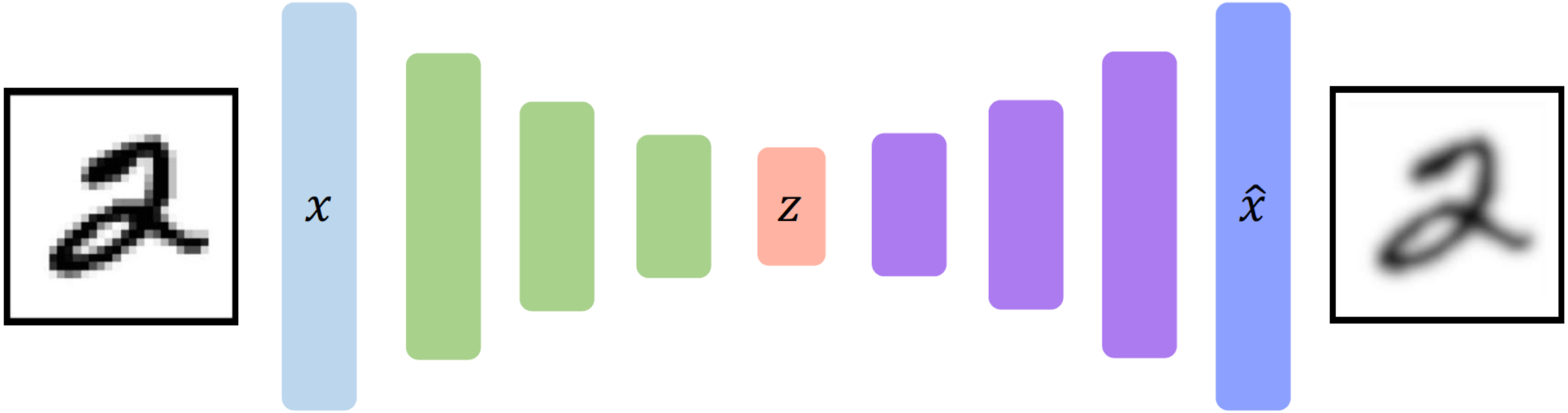
Why do we care about a low-dimensional z?

"Encoder" learns mapping from the data, x, to a low-dimensional latent space, z

Autoencoders: background

How can we learn this latent space?

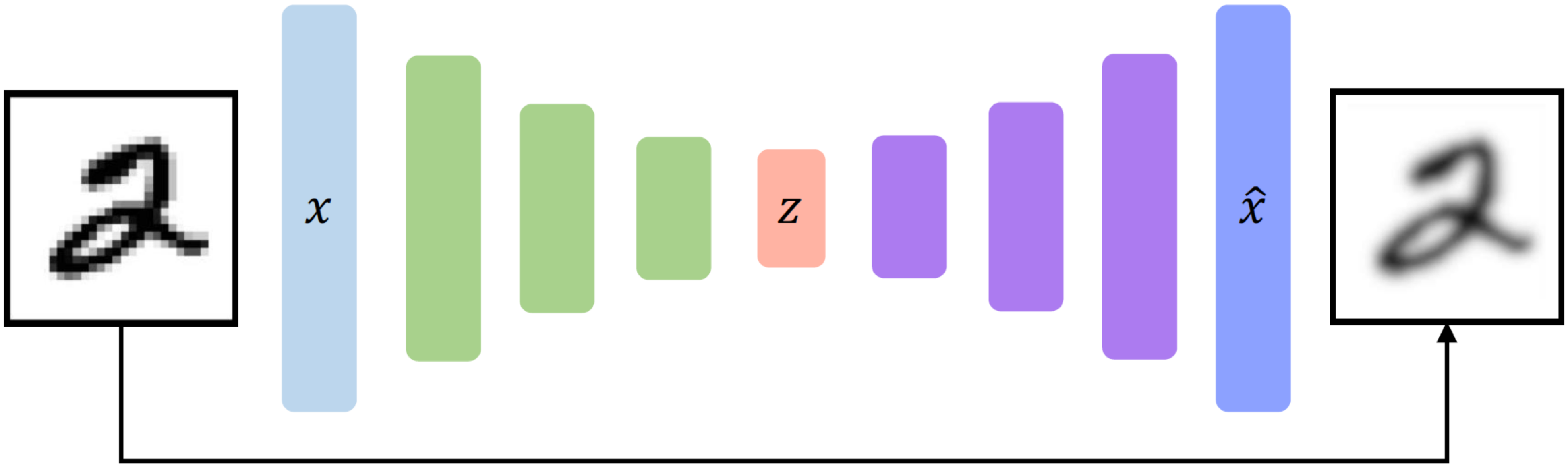
Train the model to use these features to **reconstruct the original data**



"Decoder" learns mapping back from latent, z, to a reconstructed observation, $\hat{x}$

Autoencoders: background

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data**



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

Loss function doesn't use any labels!!

Dimensionality of latent space → reconstruction quality

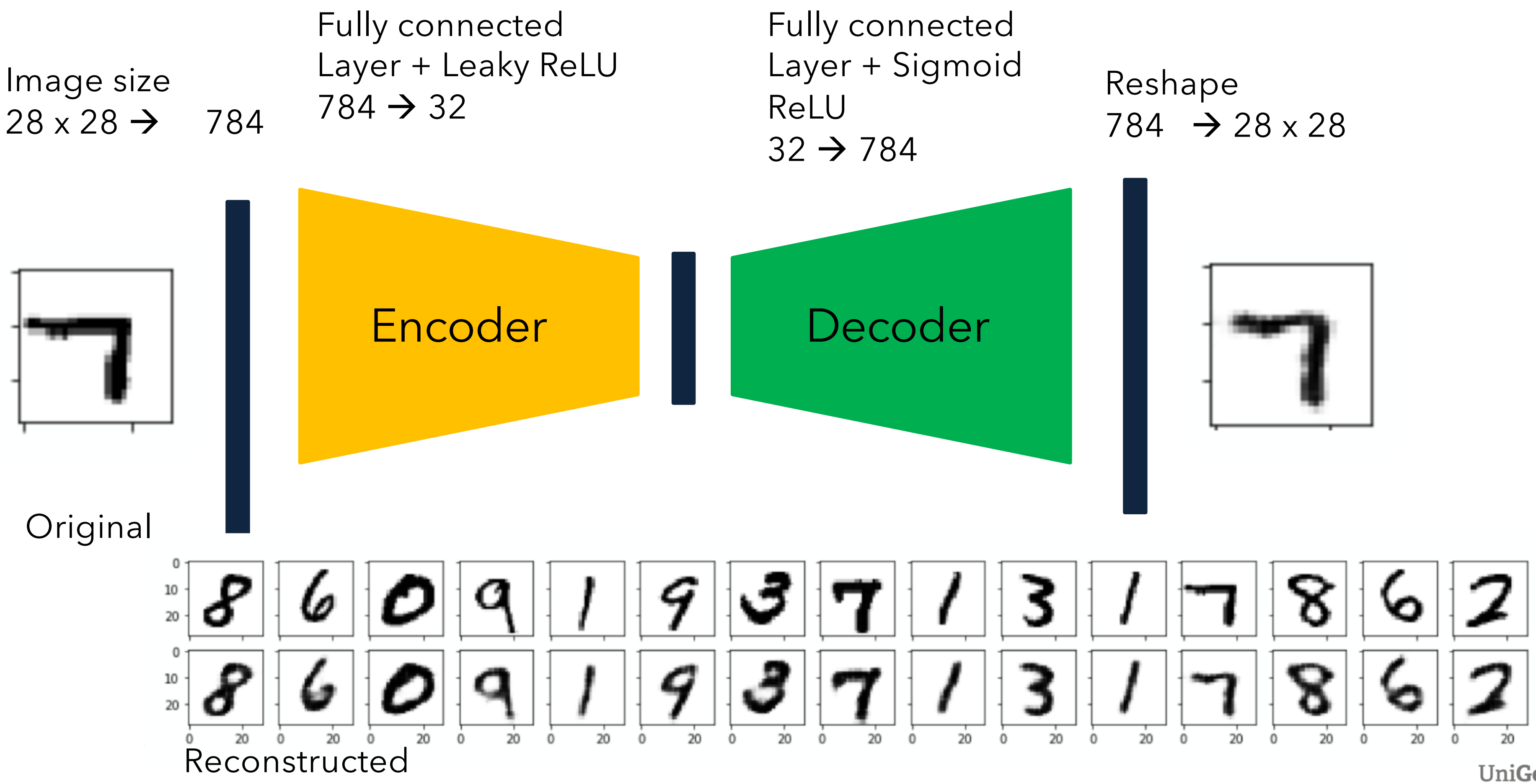
Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space

5D latent space

Ground Truth

An example: simple autoencoder

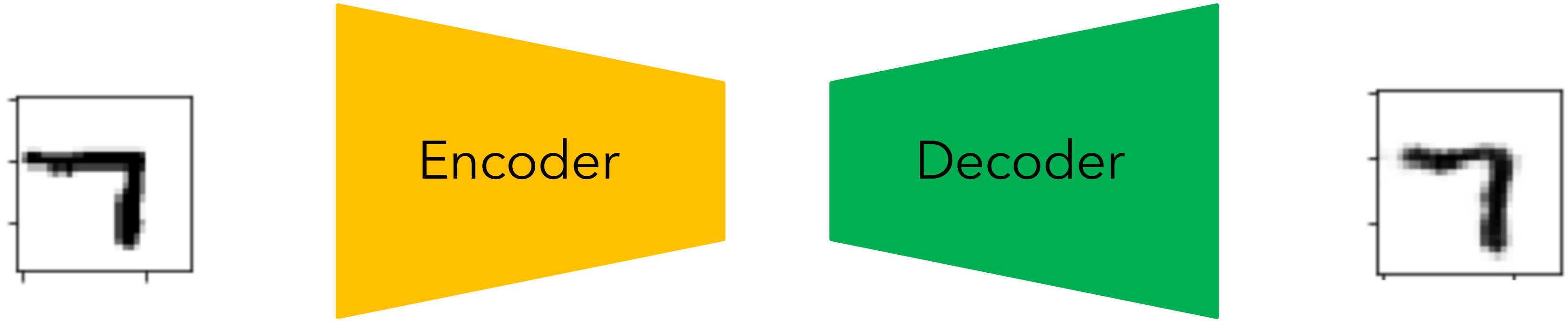


UniGe | MaLGa

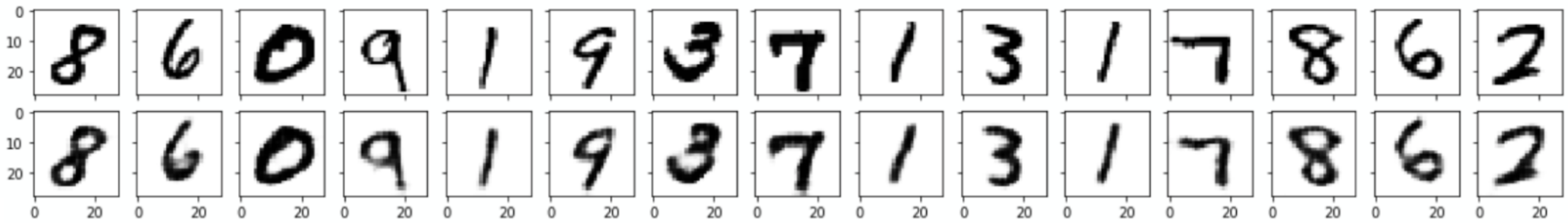
An example: Convolutional autoencoder

One or more convolutional layers

One or more de-convolutional layers



Original



Reconstructed

De-convolution

- When managing image data, encoder and decoder are made of, respectively, convolutional and de-convolutional layers

- **De-convolution** (often referred to as transposed convolution because, mathematically, deconvolution is in fact a different operation) allows to go from a lower resolution image to a higher resolution image

Regular convolution (in the encoder)

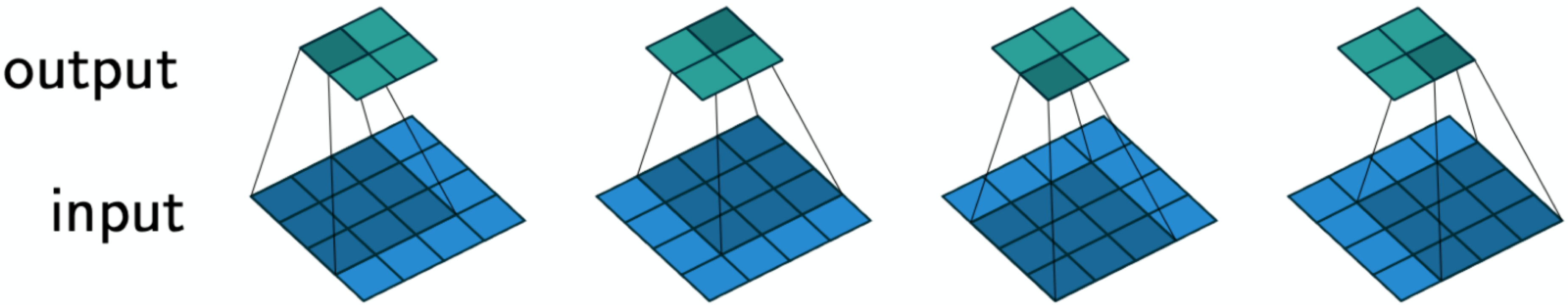
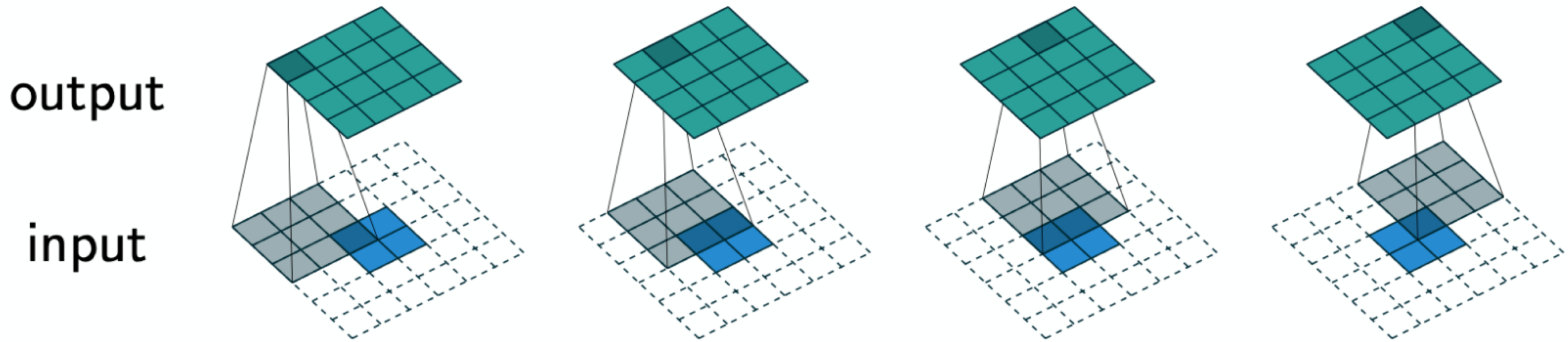


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

From https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/L15_autoencoder_slides.pdf and Dumoulin, Vincent, and Francesco Visin. A guide to convolution arithmetic for deep learning. arXiv preprint (2016)

De-convolution (in the decoder)



Dumoulin, Vincent, and Francesco Visin. "[A guide to convolution arithmetic for deep learning](#)." *arXiv preprint arXiv:1603.07285* (2016).

From https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/L15_autoencoder_slides.pdf and Dumoulin, Vincent, and Francesco Visin. A guide to convolution arithmetic for deep learning. arXiv preprint (2016)

Autoencoders for representation learning

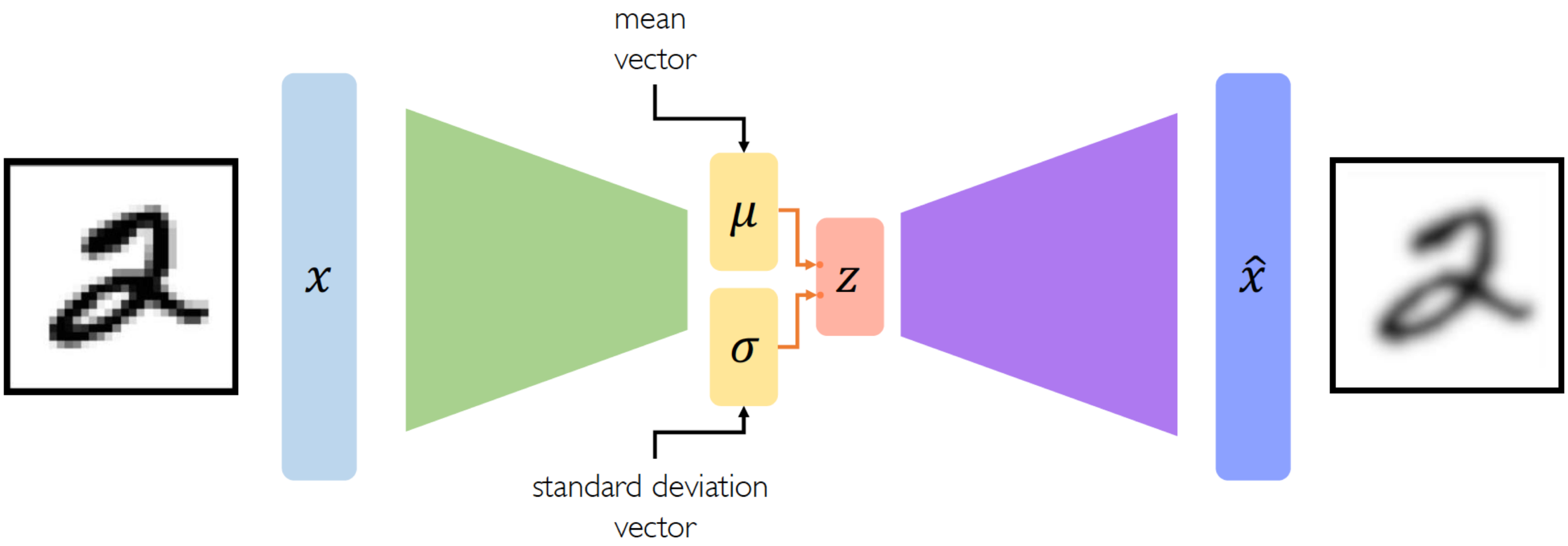
Bottleneck hidden layer: forces network to learn a compressed latent representation

Reconstruction loss: forces the latent representation to capture (or encode) as much "information" about the data as possible

Autoencoding = **Auto**matically **encoding** data

Variational Autoencoders (VAEs)

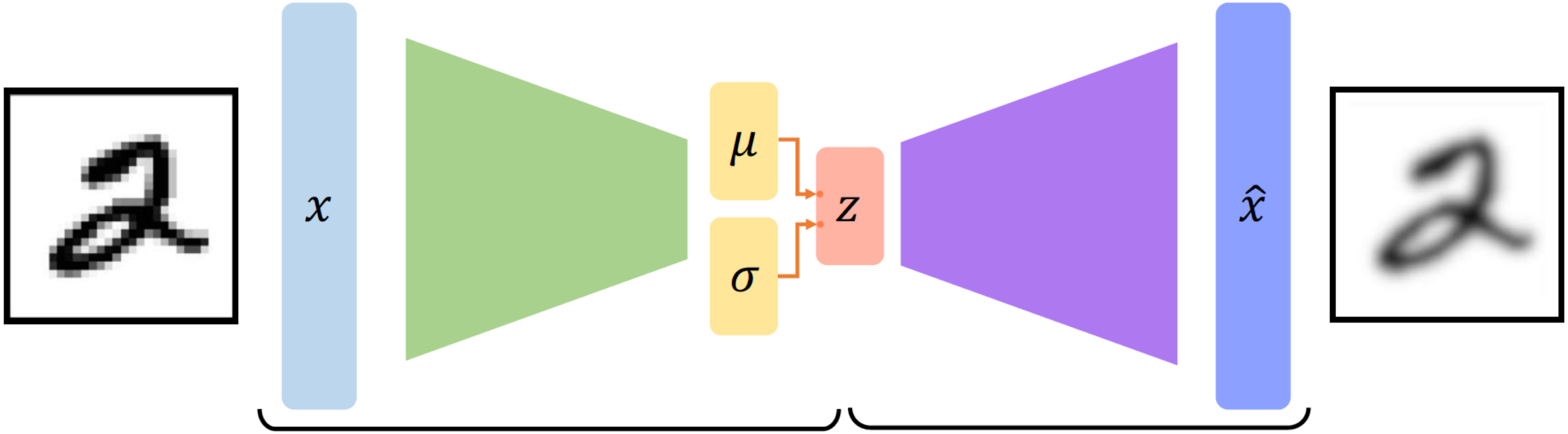
VAEs: key difference with traditional autoencoder



Variational autoencoders are a probabilistic twist on autoencoders!

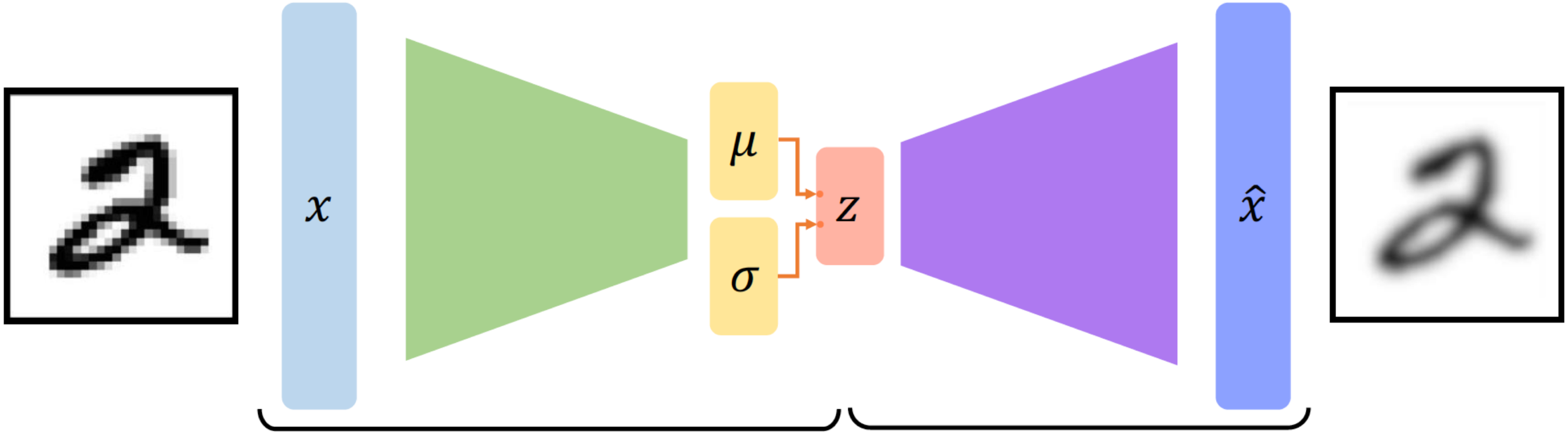
Sample from the mean and standard dev. to compute latent sample

VAE optimization



Encoder computes: $p_{\phi}(z|x)$ Decoder computes: $q_{\theta}(x|z)$

VAE optimization

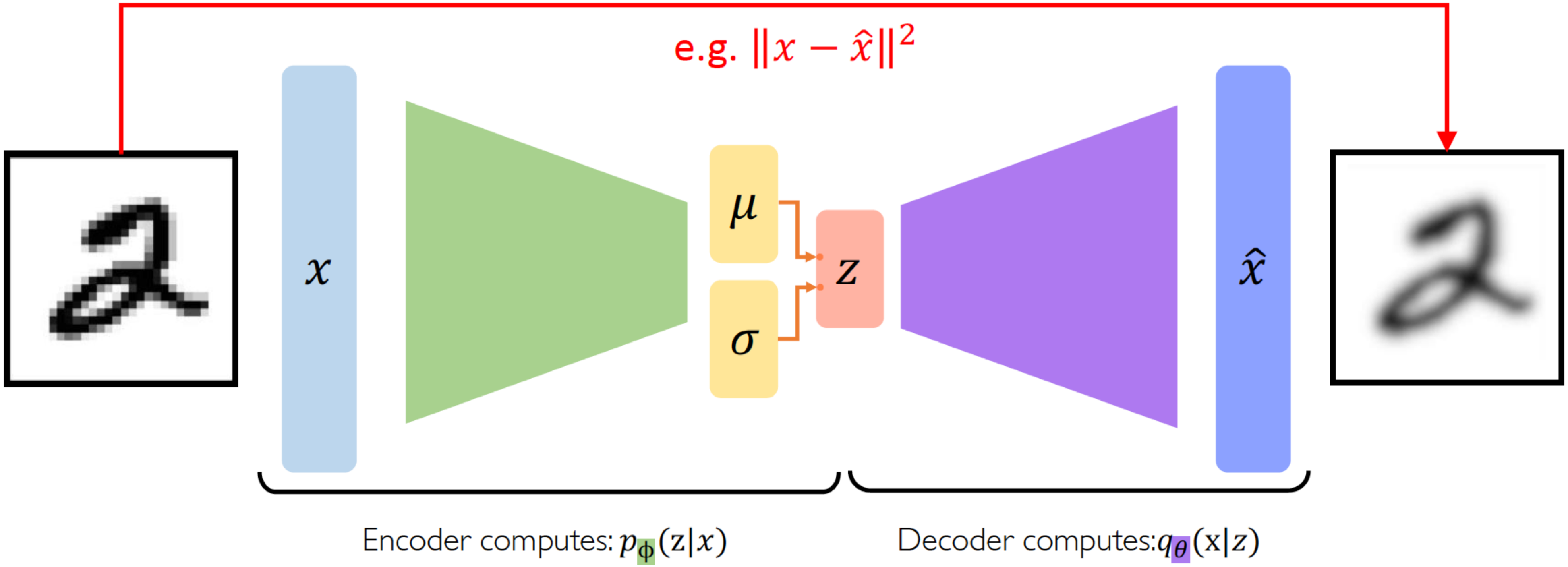


Encoder computes: $p_{\phi}(z|x)$

Decoder computes: $q_{\theta}(x|z)$

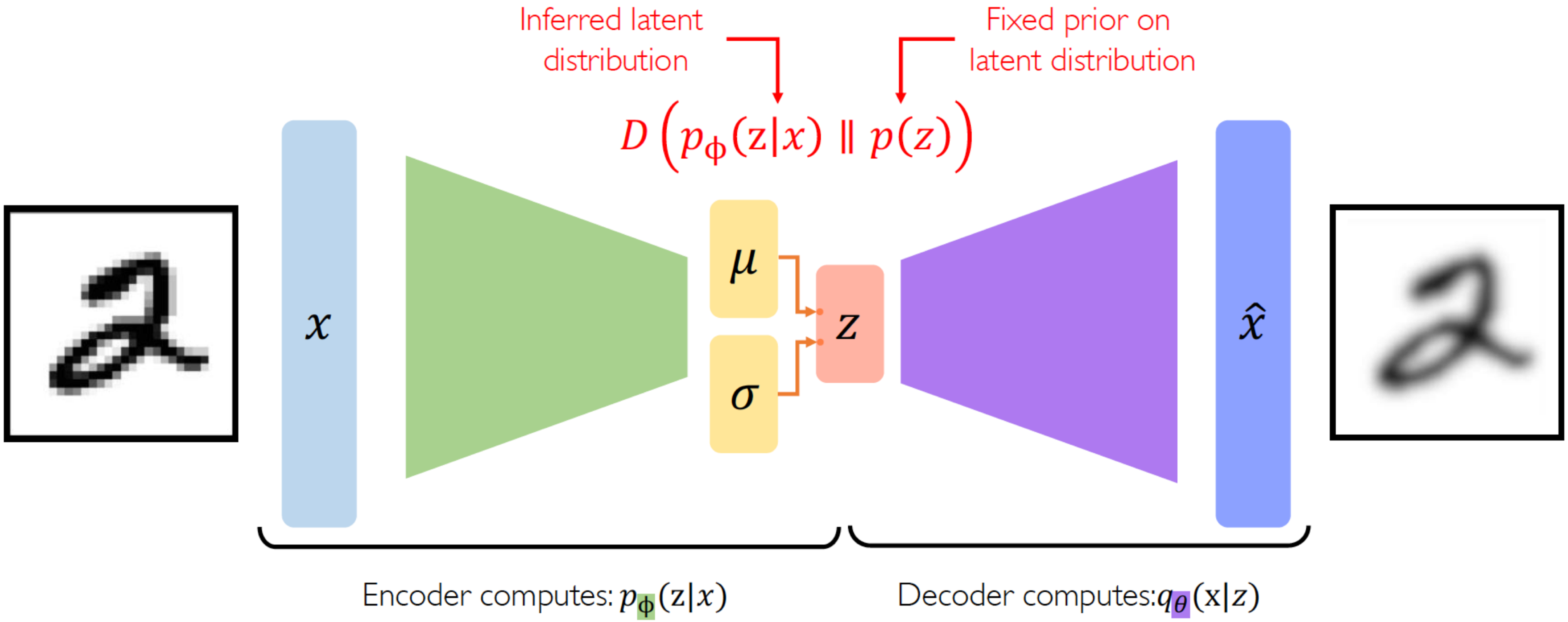
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

VAE optimization



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

VAE optimization



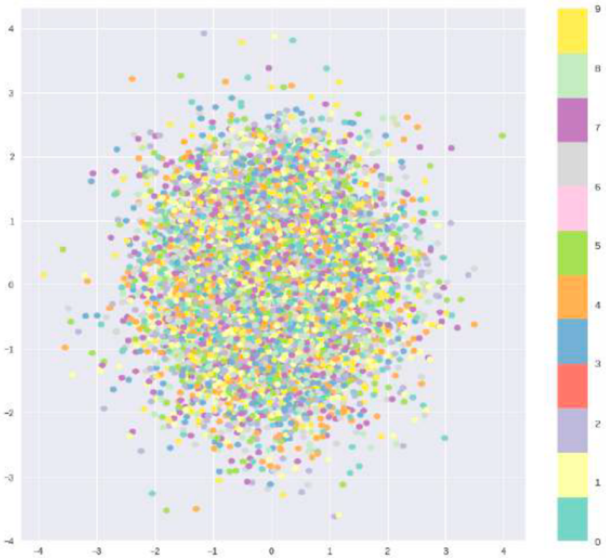
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

Priors on the latent distribution

$$D\left(p_\phi(z|x) \,\|\, p(z)\right)$$

Inferred latent distribution

Fixed prior on latent distribution



Common choice of prior:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

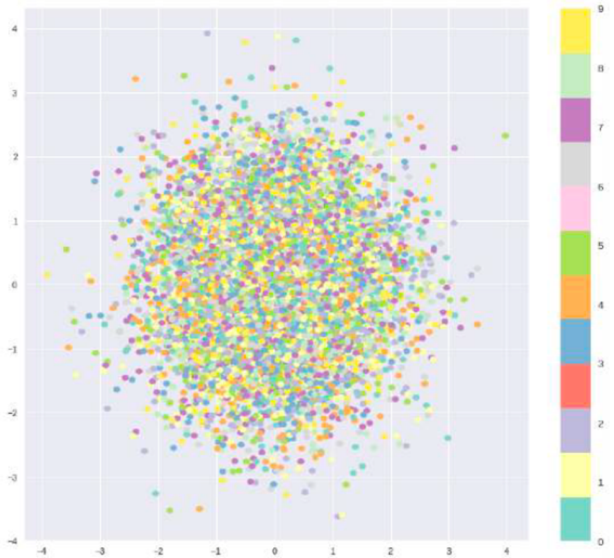
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (ie. memorizing the data)

Priors on the latent distribution

$$D\left(p_{\phi}(z|x) \parallel p(z)\right)$$

$$= -\frac{1}{2}\sum_{j=0}^{k-1}\left(\sigma_j + \mu_j^2 - 1 - \log \sigma_j\right)$$

KL-divergence between the two distributions



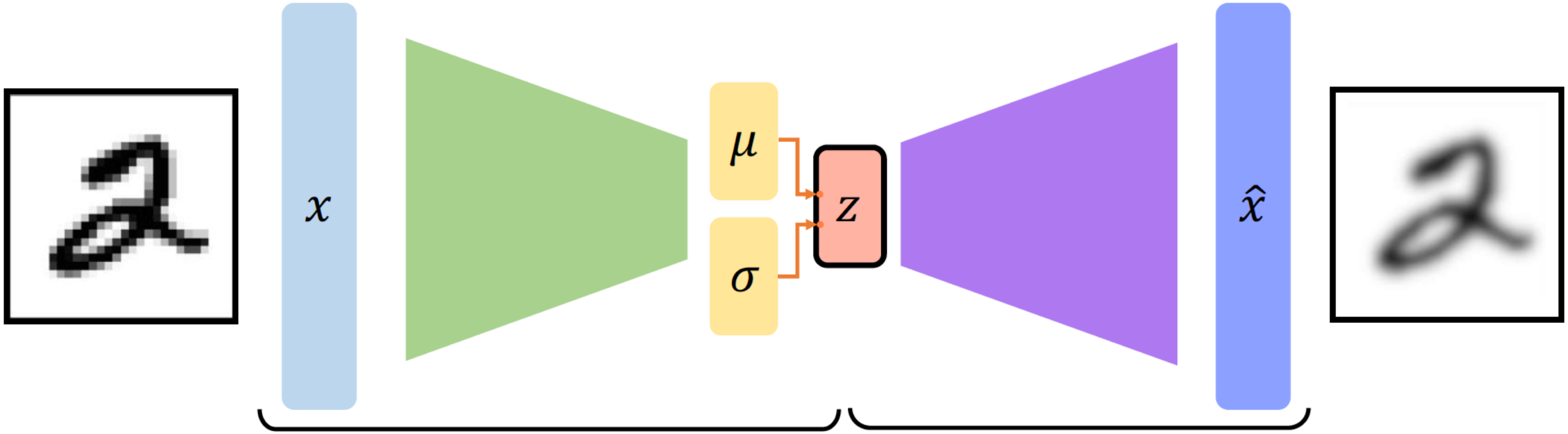
Common choice of prior:

$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (ie. memorizing the data)

VAEs computation graph

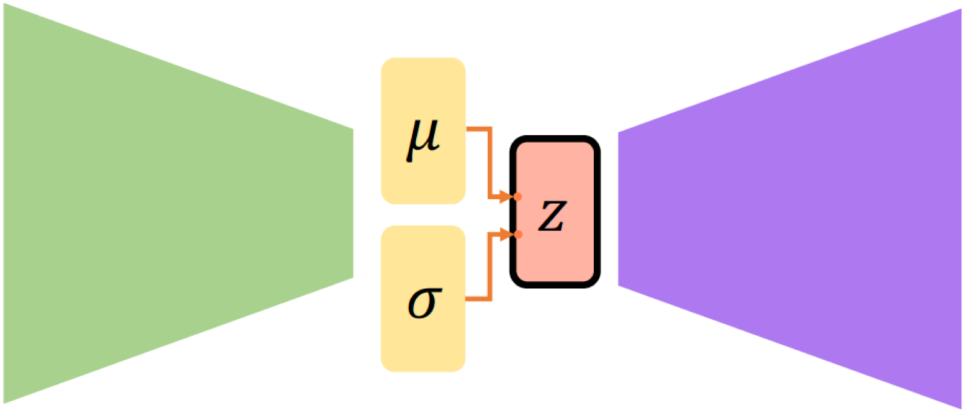
Problem: We cannot backpropagate gradients through sampling layers!



Encoder computes: $p_\phi(\mathrm{z}|x)$ Decoder computes: $q_\theta(\mathrm{x}|z)$

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

Reparametrizing the sampling layer



Key Idea:

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

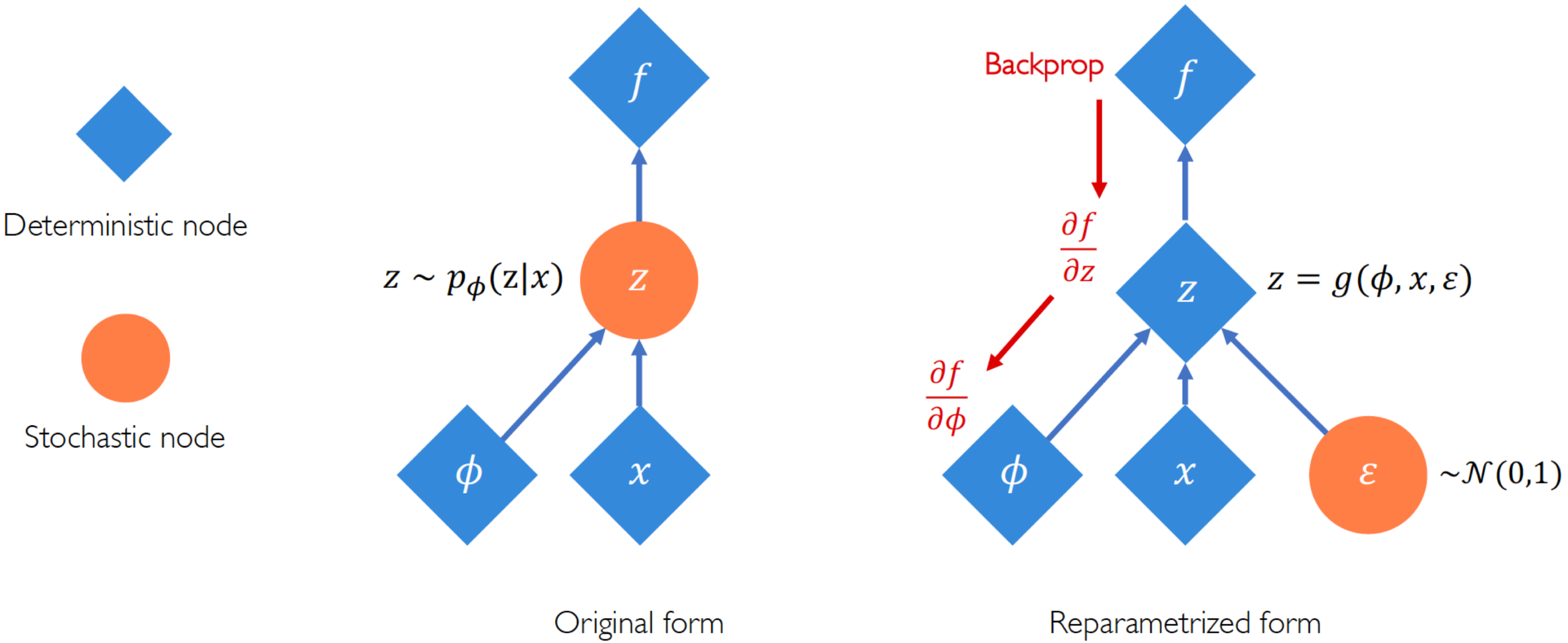
Consider the sampled latent vector as a sum of

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

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

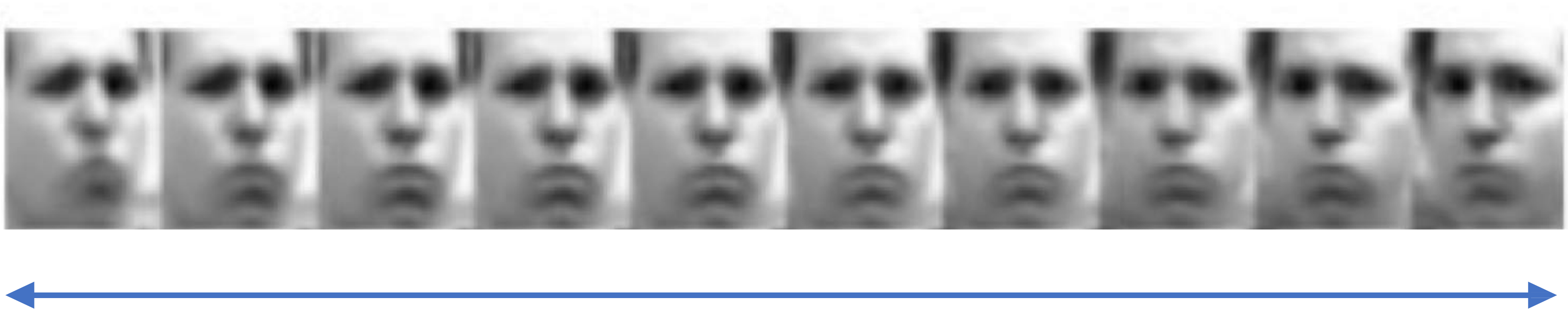
where $\varepsilon \sim \mathcal{N}(0,1)$

Reparametrizing the sampling layer



VAEs: Latent perturbation

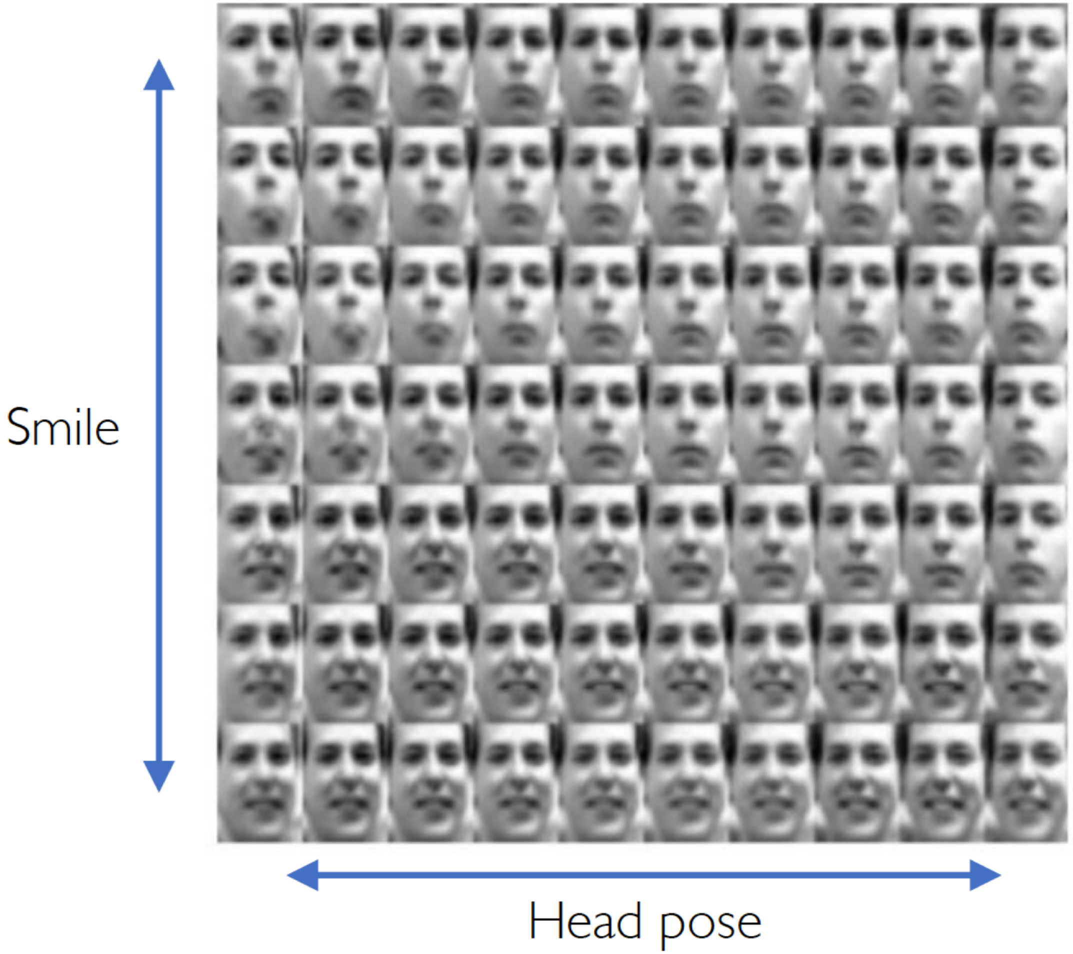
Slowly increase or decrease a **single latent variable**
Keep all other variables fixed



Head pose

Different dimensions of z encodes **different interpretable latent features**

VAEs: Latent perturbation

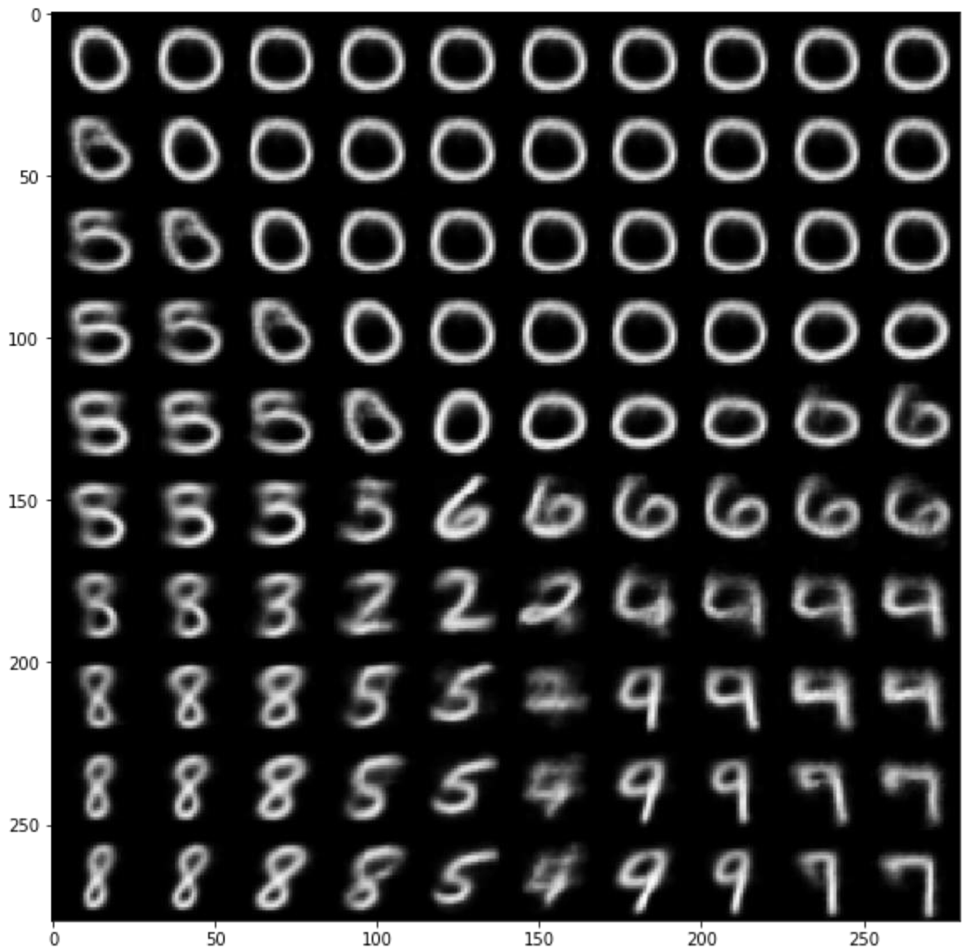


Ideally, we want latent variables that are uncorrelated with each other

Enforce diagonal prior on the latent variables to encourage independence

Disentanglement

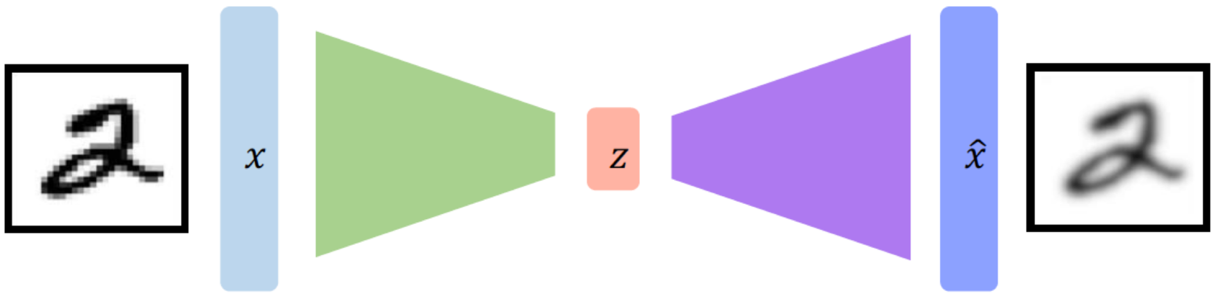
VAEs: Latent perturbation



VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)

4. Interpret hidden latent variables using perturbation
5. Generating new examples



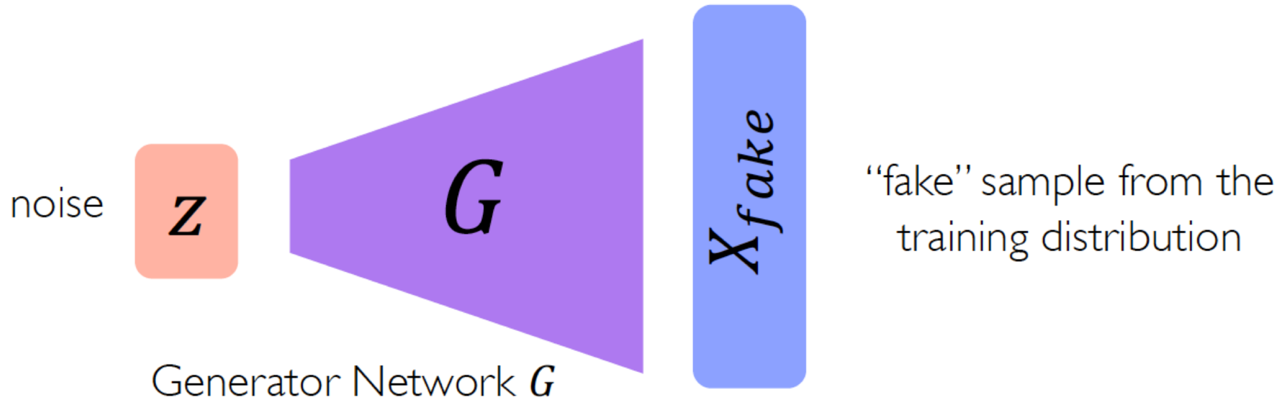
Generative Adversarial Networks (GANs)

What if we just want to sample?

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

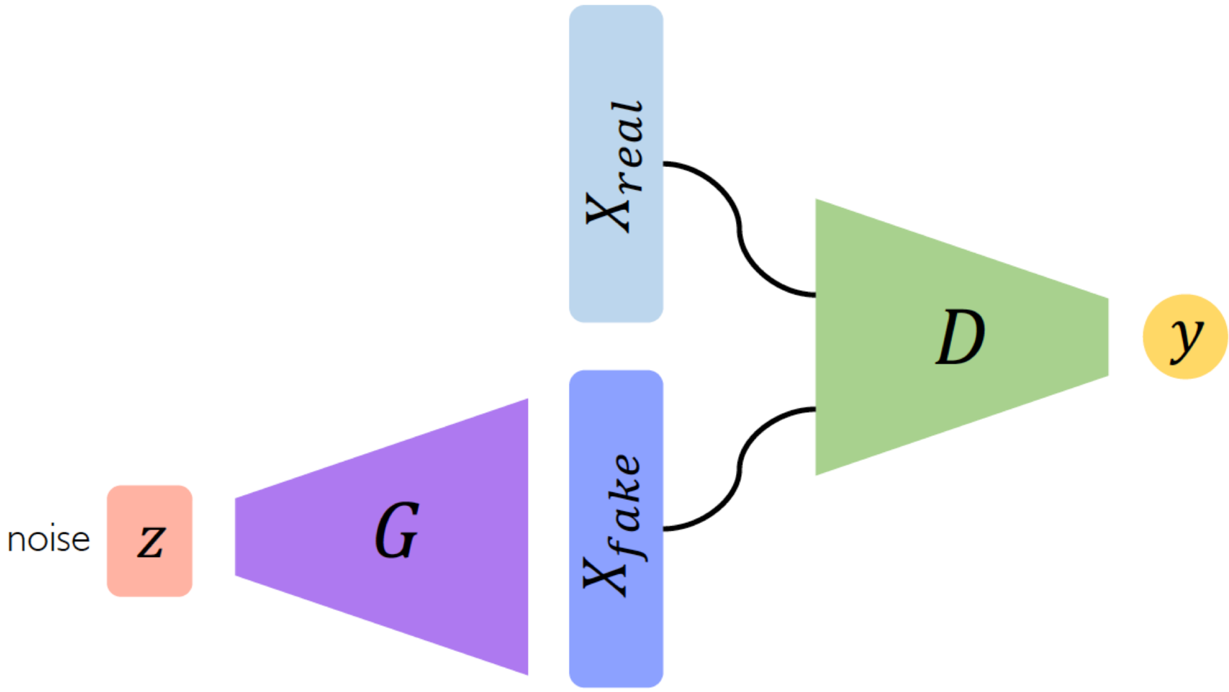
Problem: want to sample from complex distribution – can't do this directly!

Solution: sample from something simple (noise), learn a transformation to the training distribution.



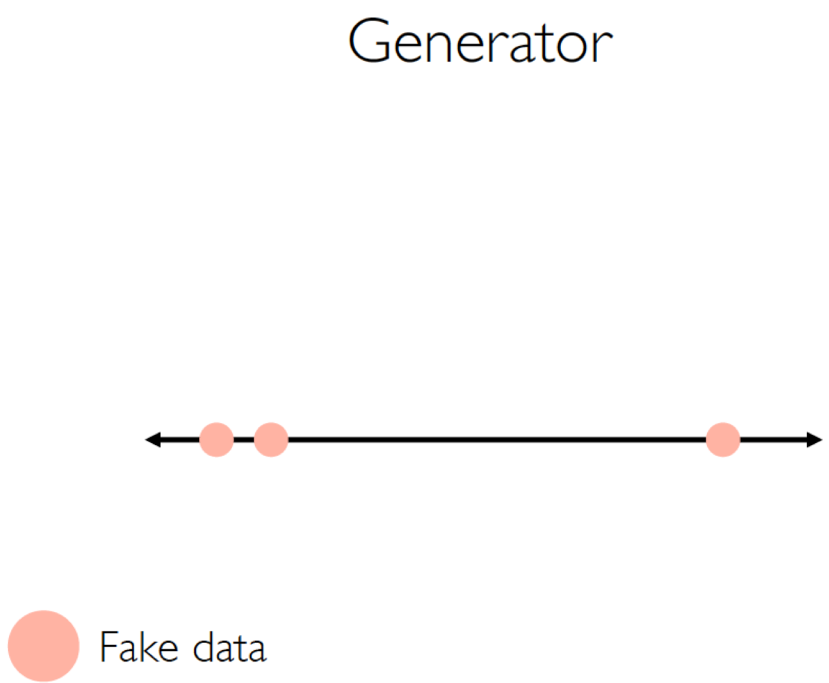
Generative Adversarial Networks (GANs)

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



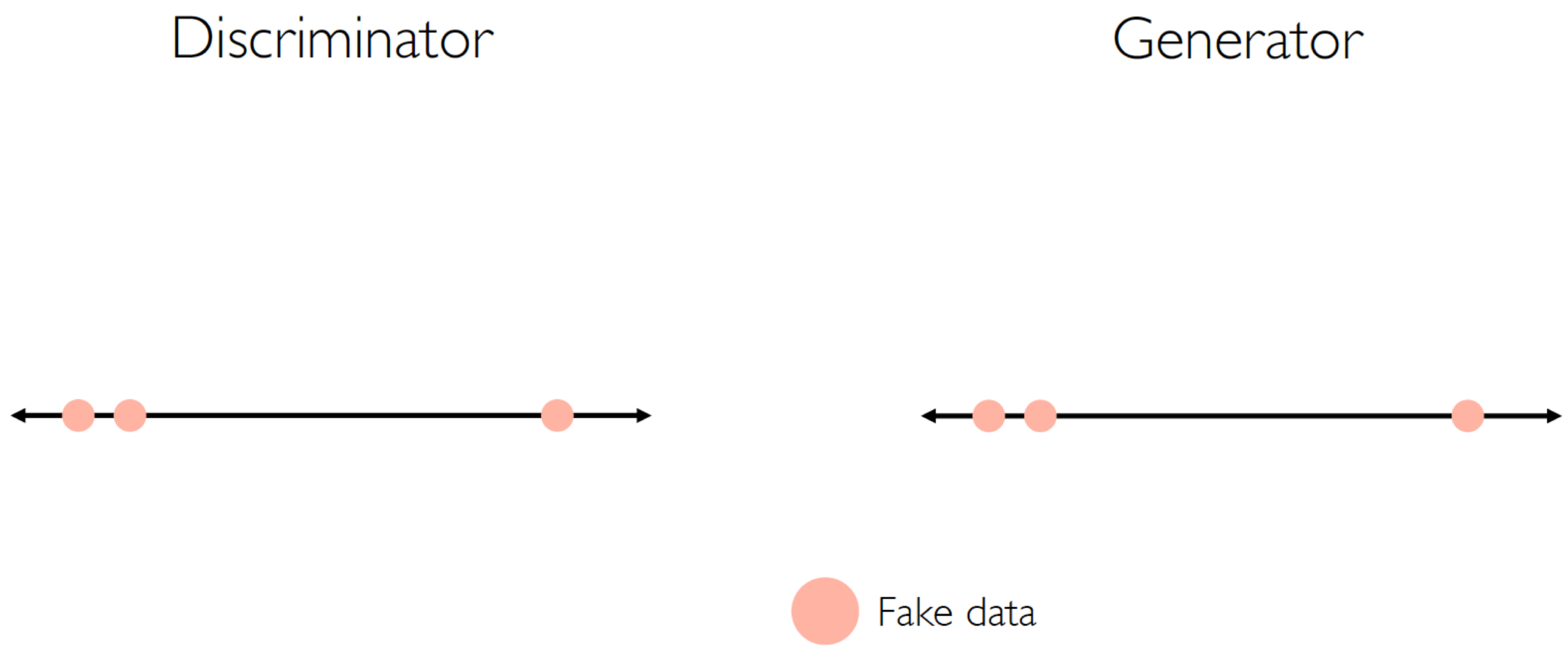
Intuition behind GANs

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



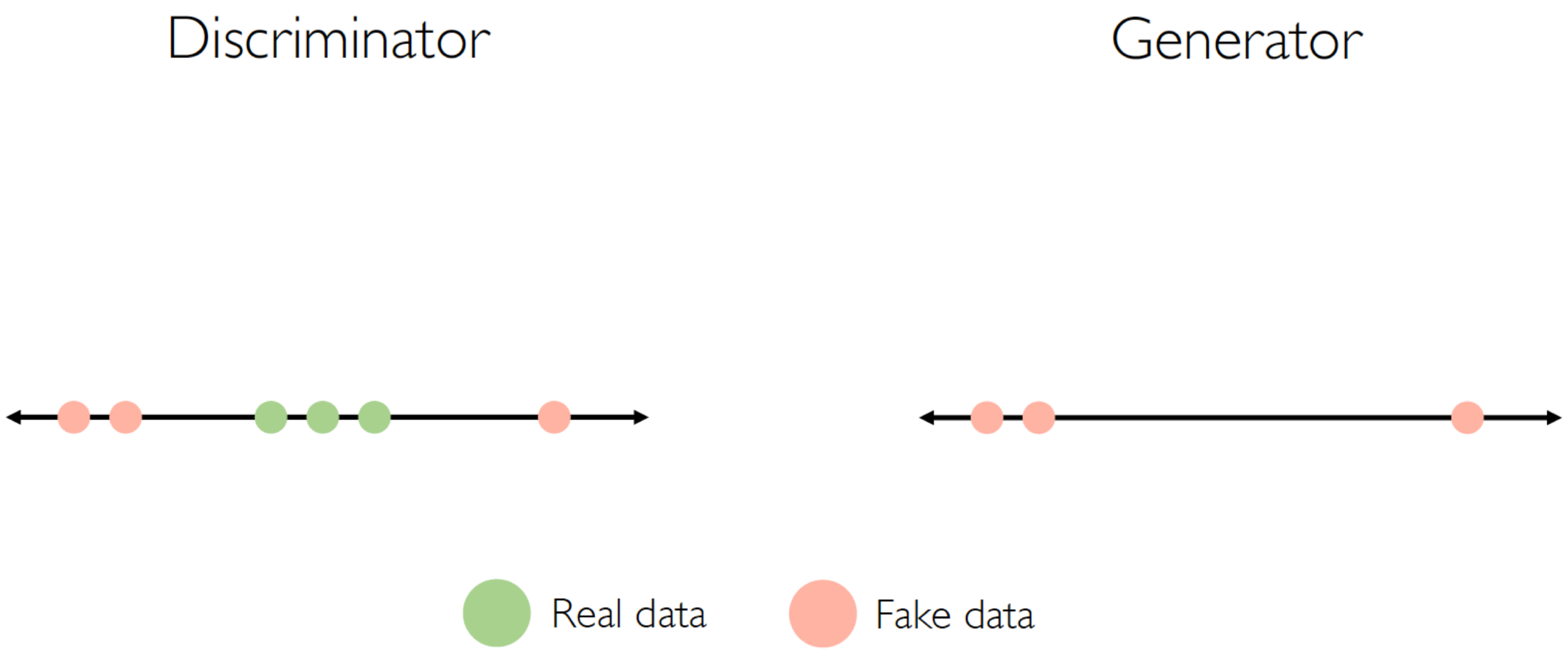
Intuition behind GANs

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



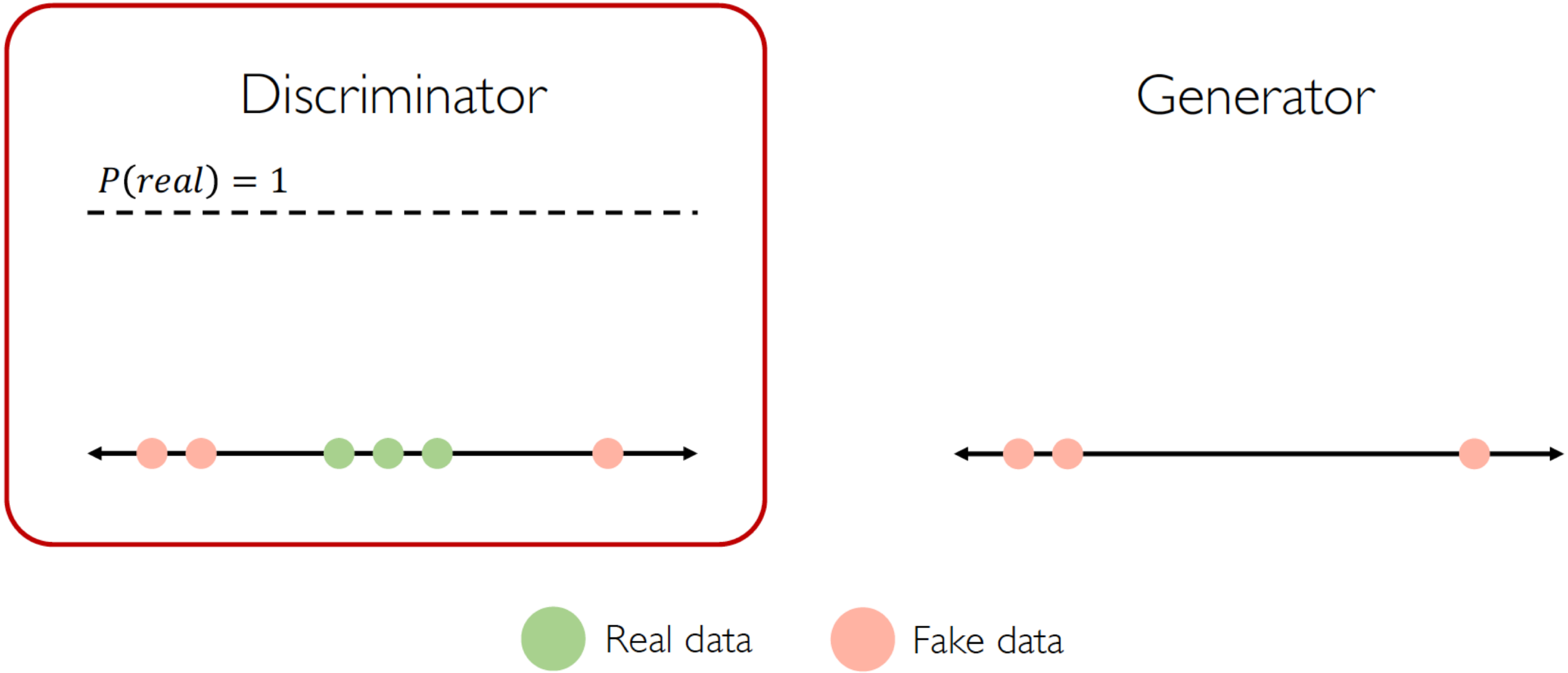
Intuition behind GANs

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



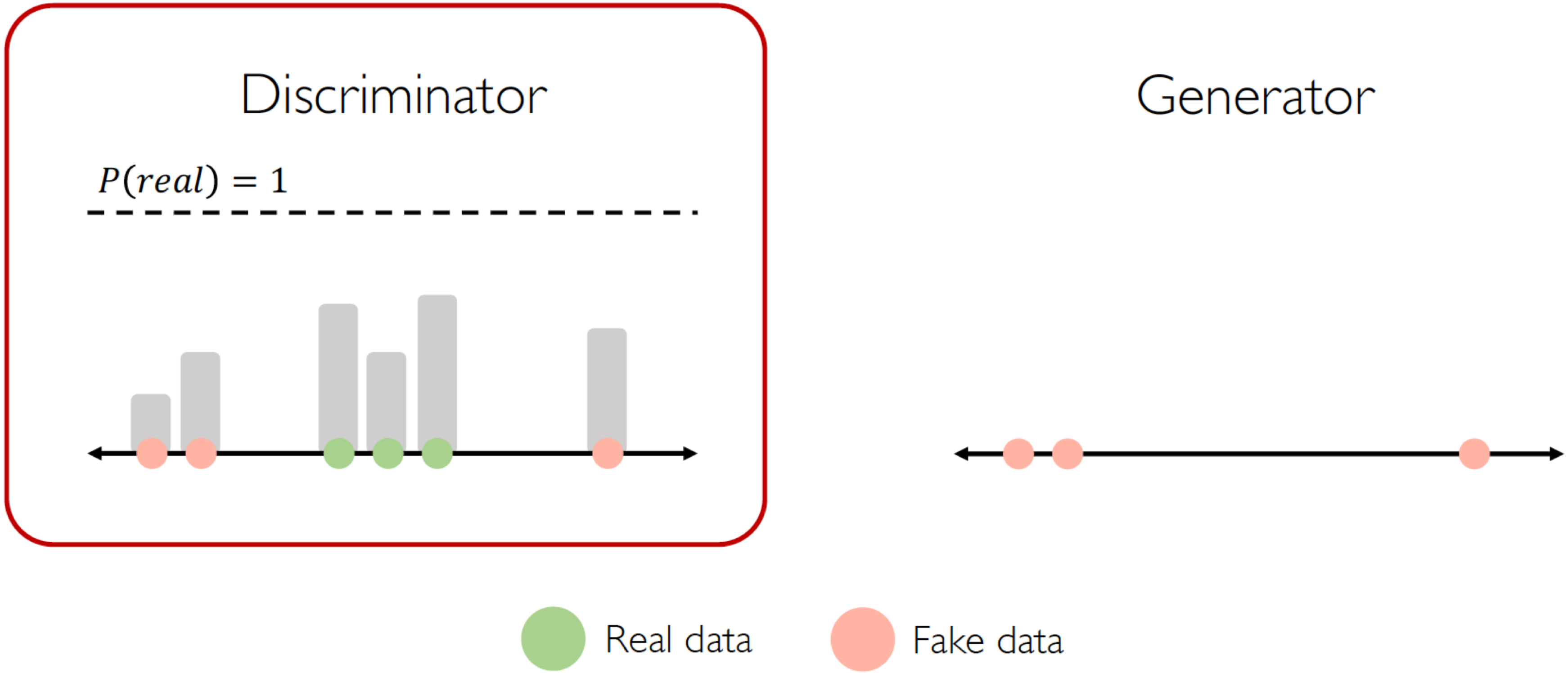
Intuition behind GANs

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



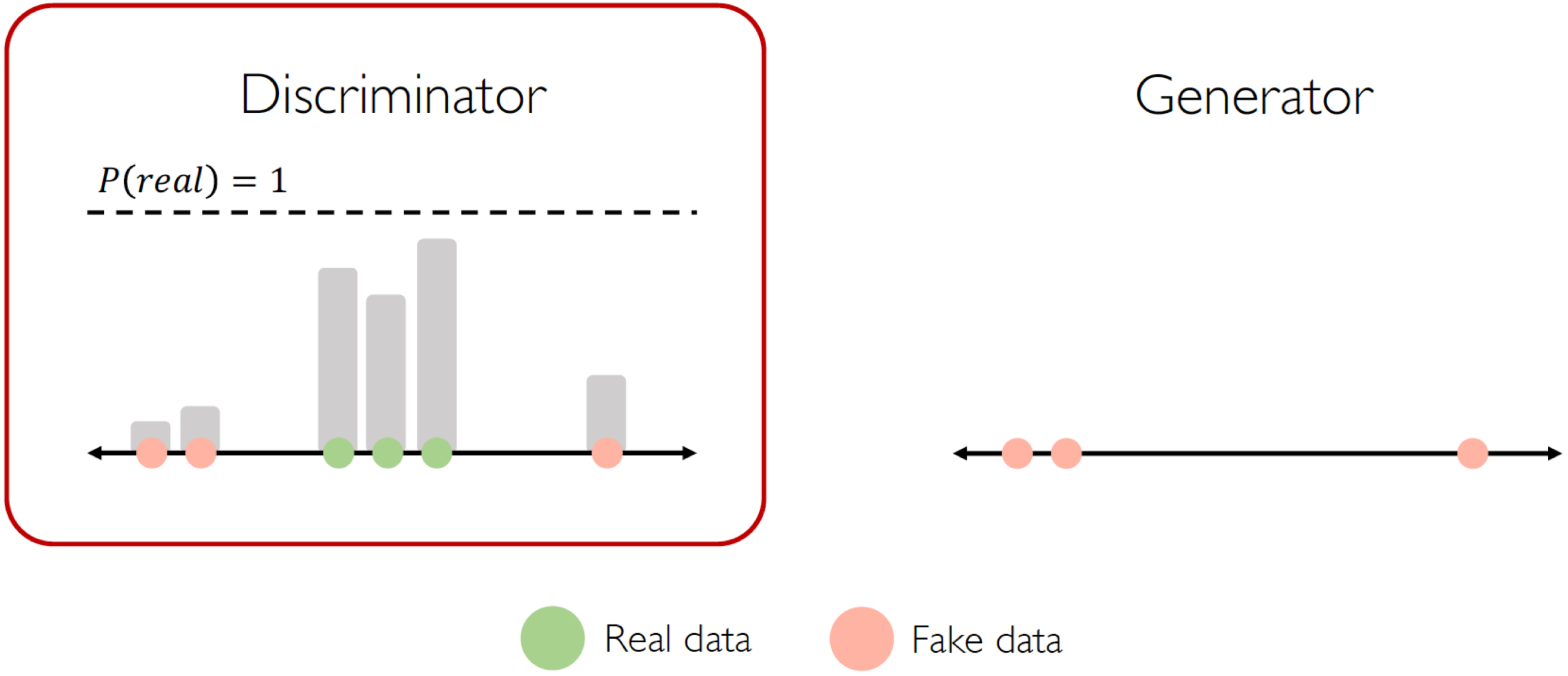
Intuition behind GANs

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



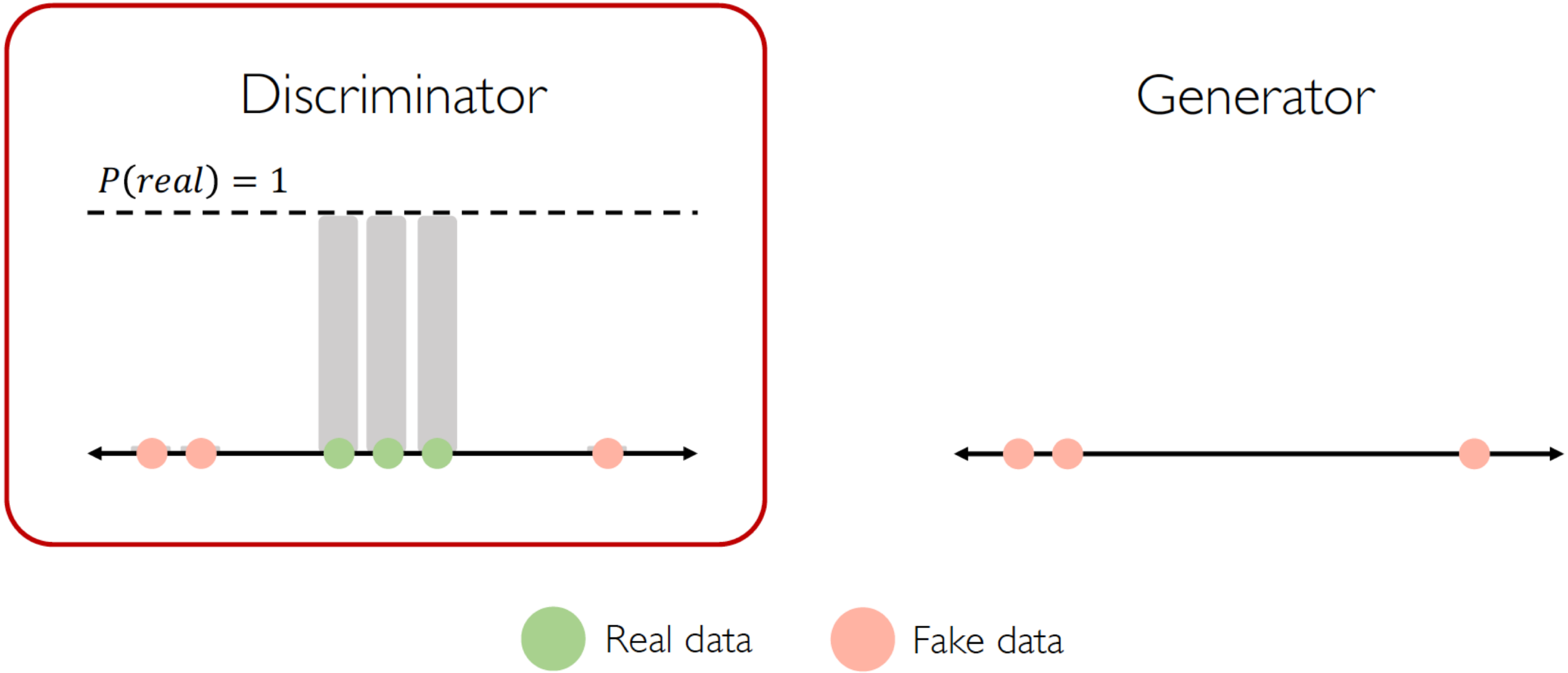
Intuition behind GANs

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



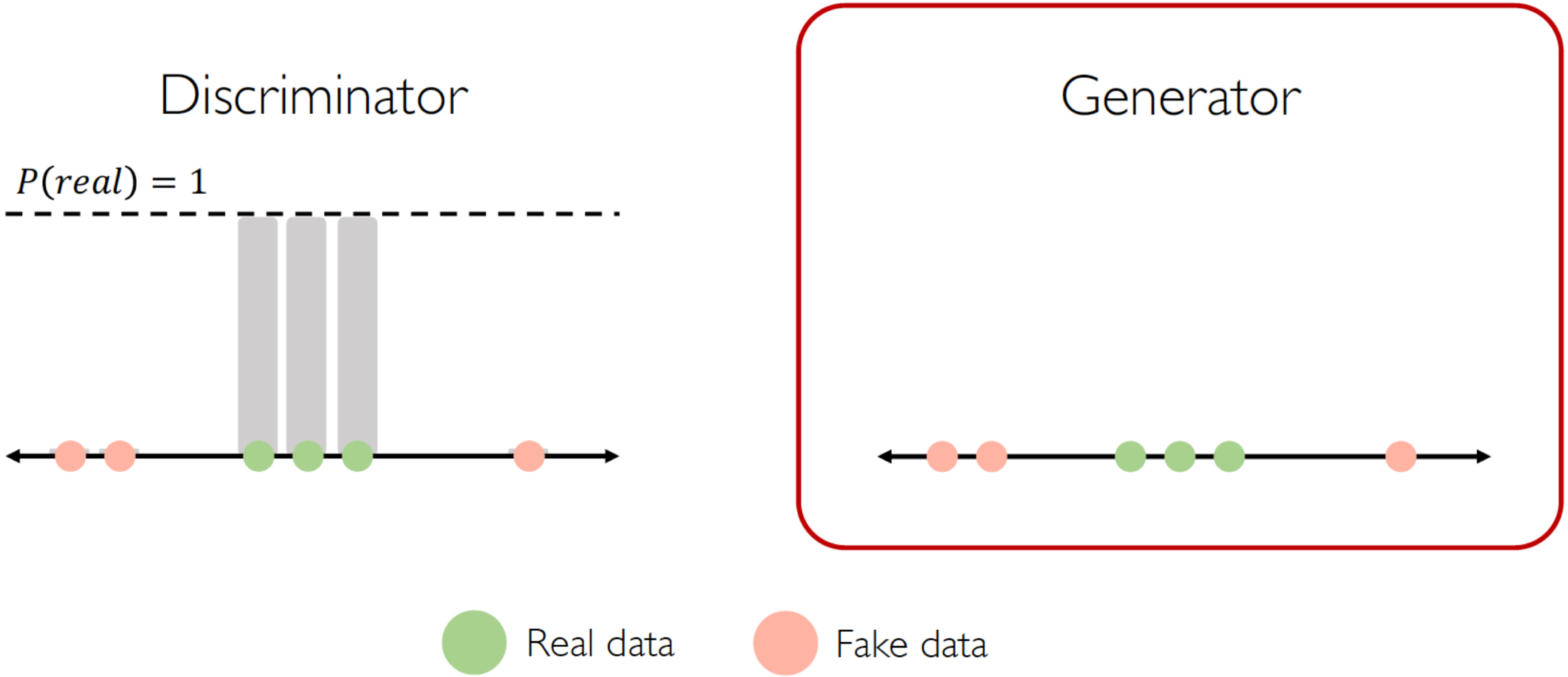
Intuition behind GANs

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



Intuition behind GANs

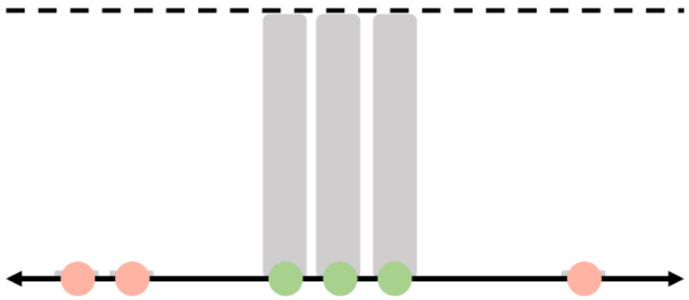
Generator tries to improve its imitation of the data.



Intuition behind GANs

Generator tries to improve its imitation of the data.

Discriminator

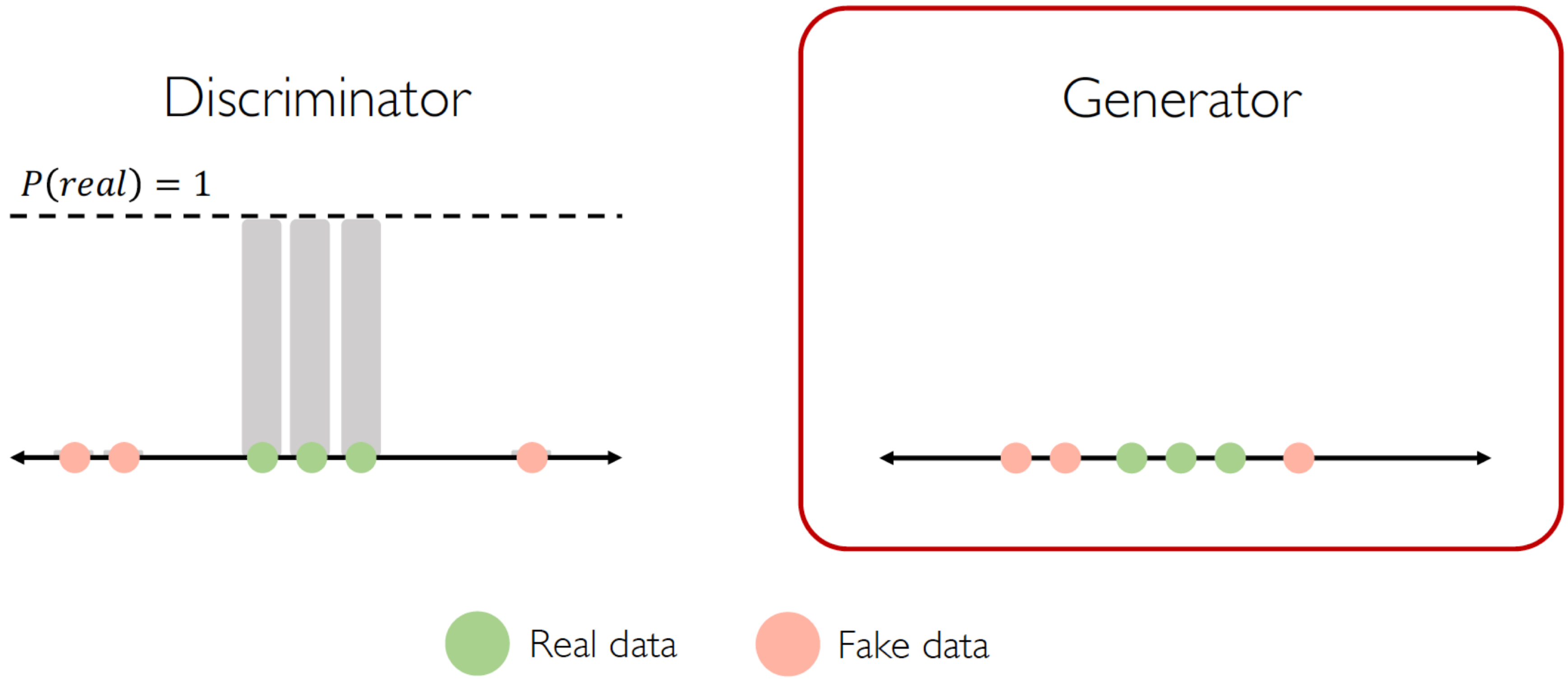


Generator

Real data   Fake data

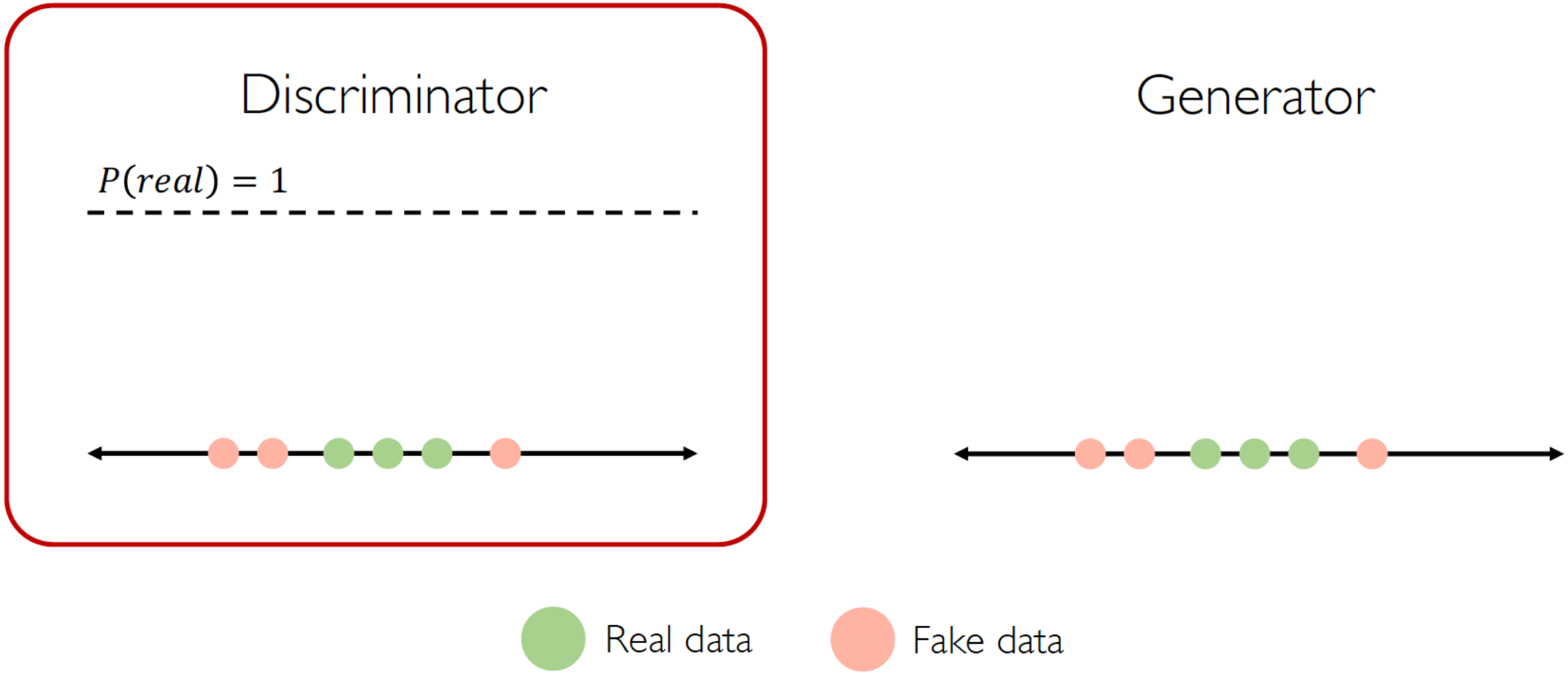
Intuition behind GANs

Generator tries to improve its imitation of the data.



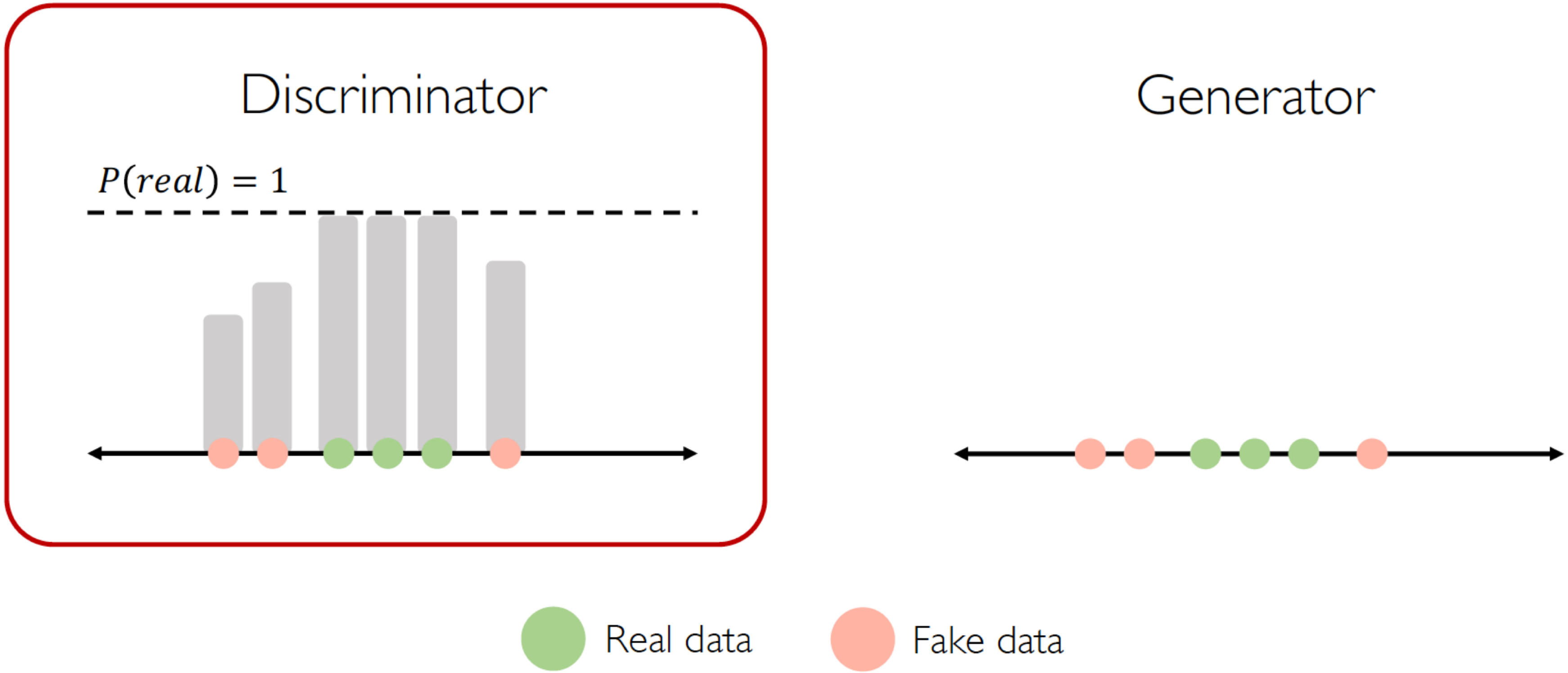
Intuition behind GANs

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



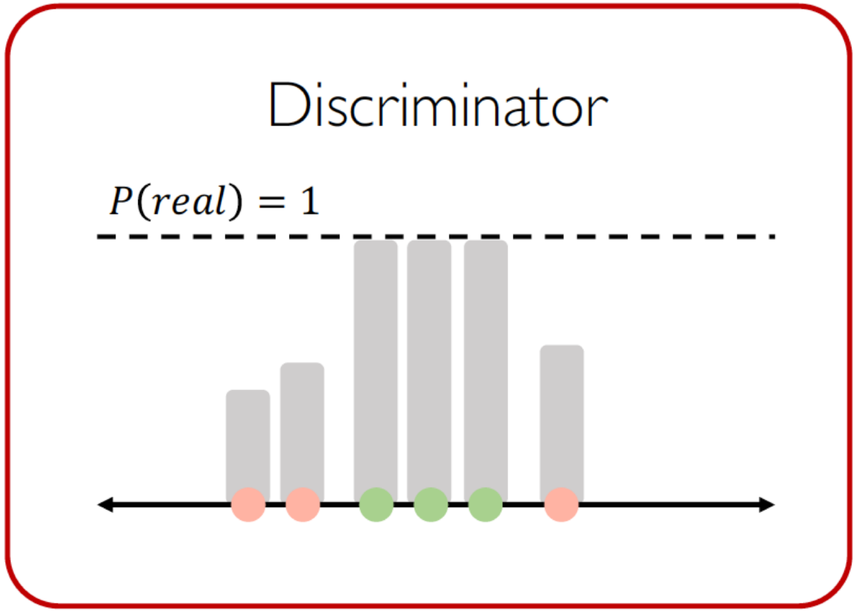
Intuition behind GANs

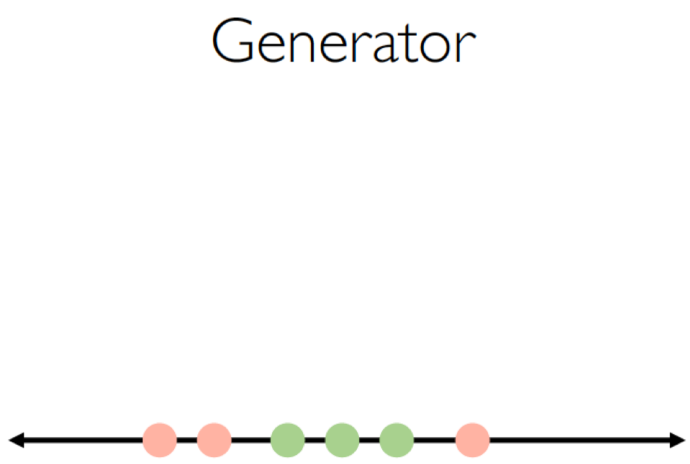
Discriminator tries to predict what's real and what's fake.



Intuition behind GANs

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

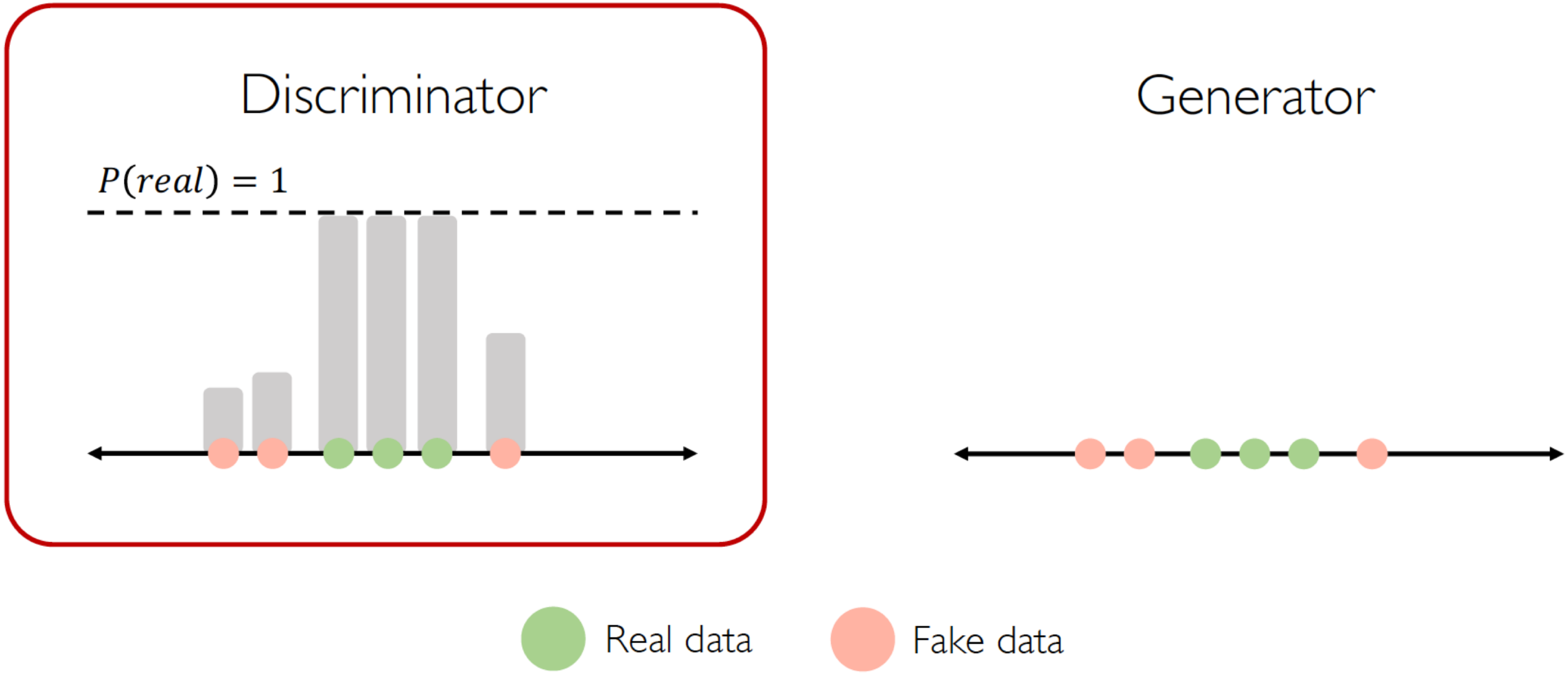




Real data   Fake data

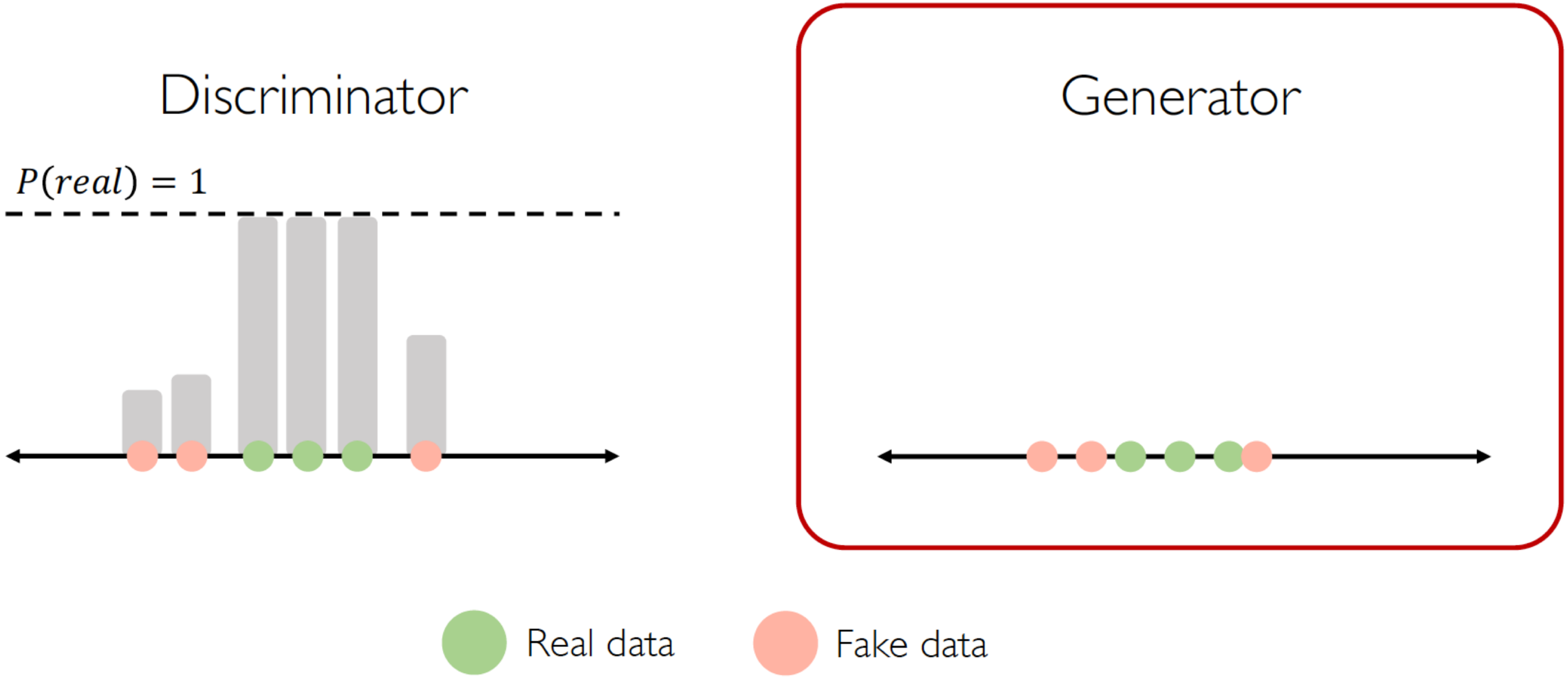
Intuition behind GANs

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



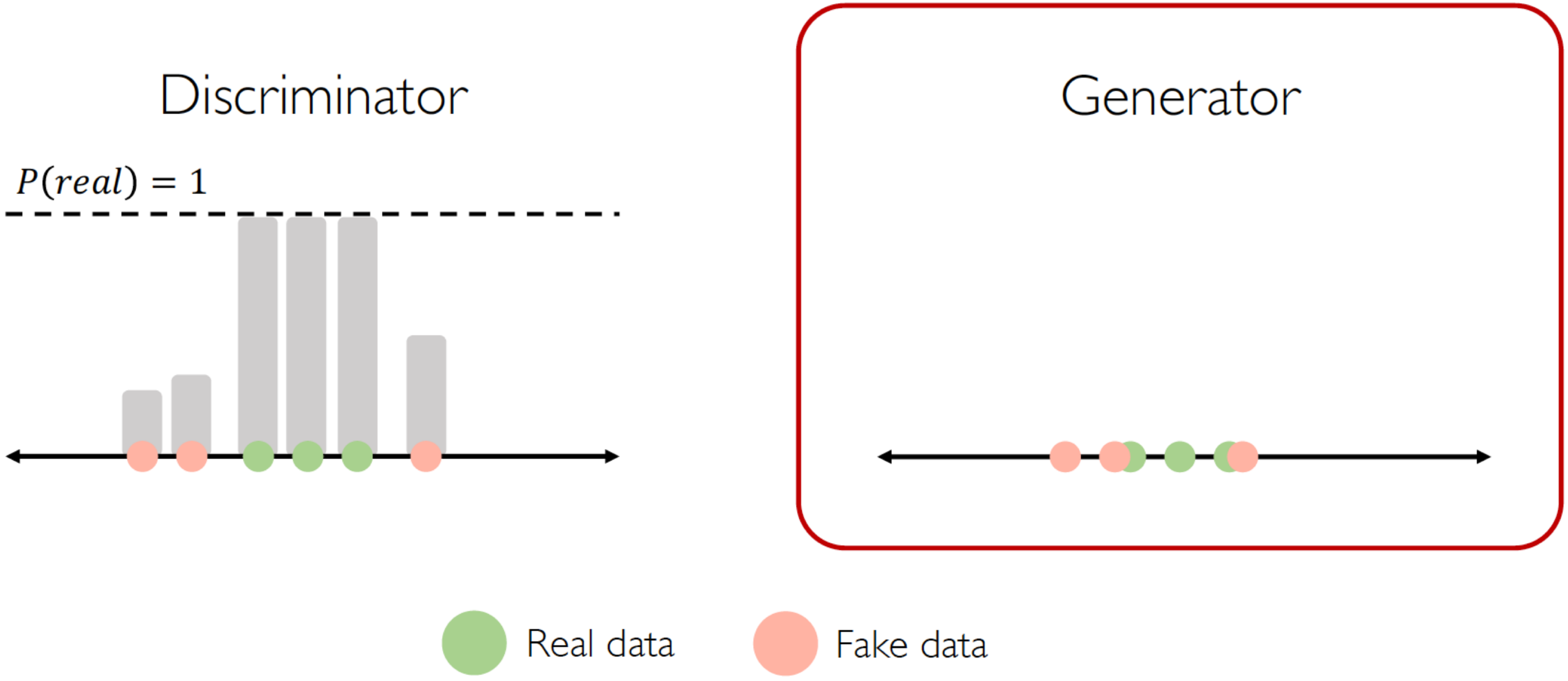
Intuition behind GANs

Generator tries to improve its imitation of the data.



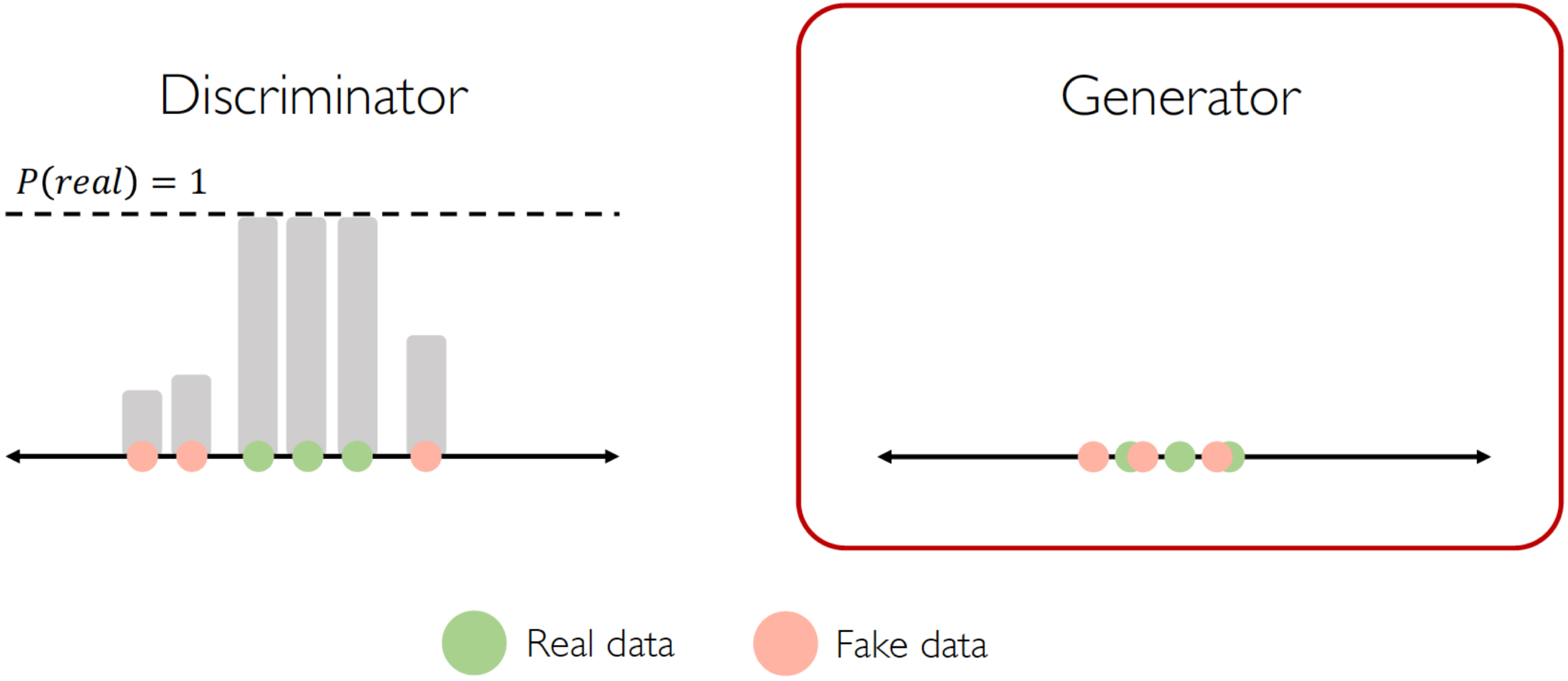
Intuition behind GANs

Generator tries to improve its imitation of the data.



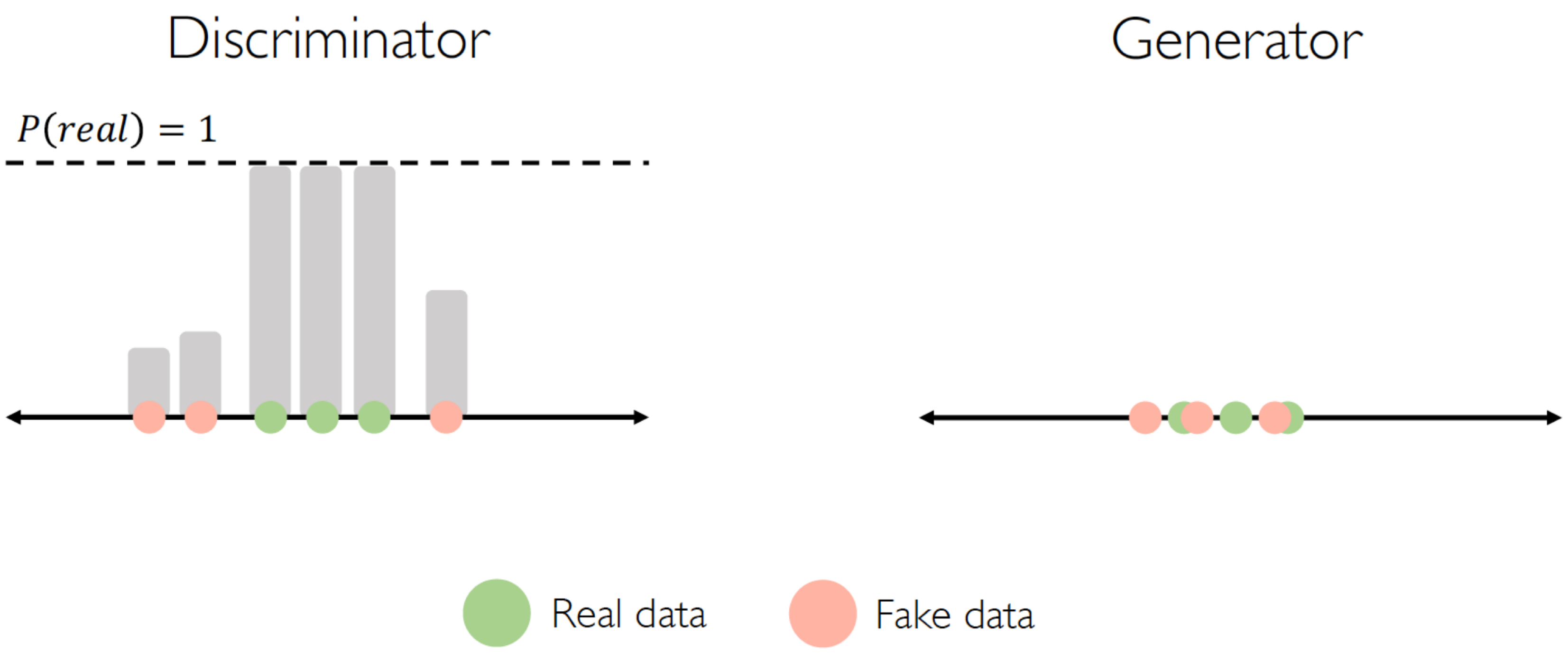
Intuition behind GANs

Generator tries to improve its imitation of the data.

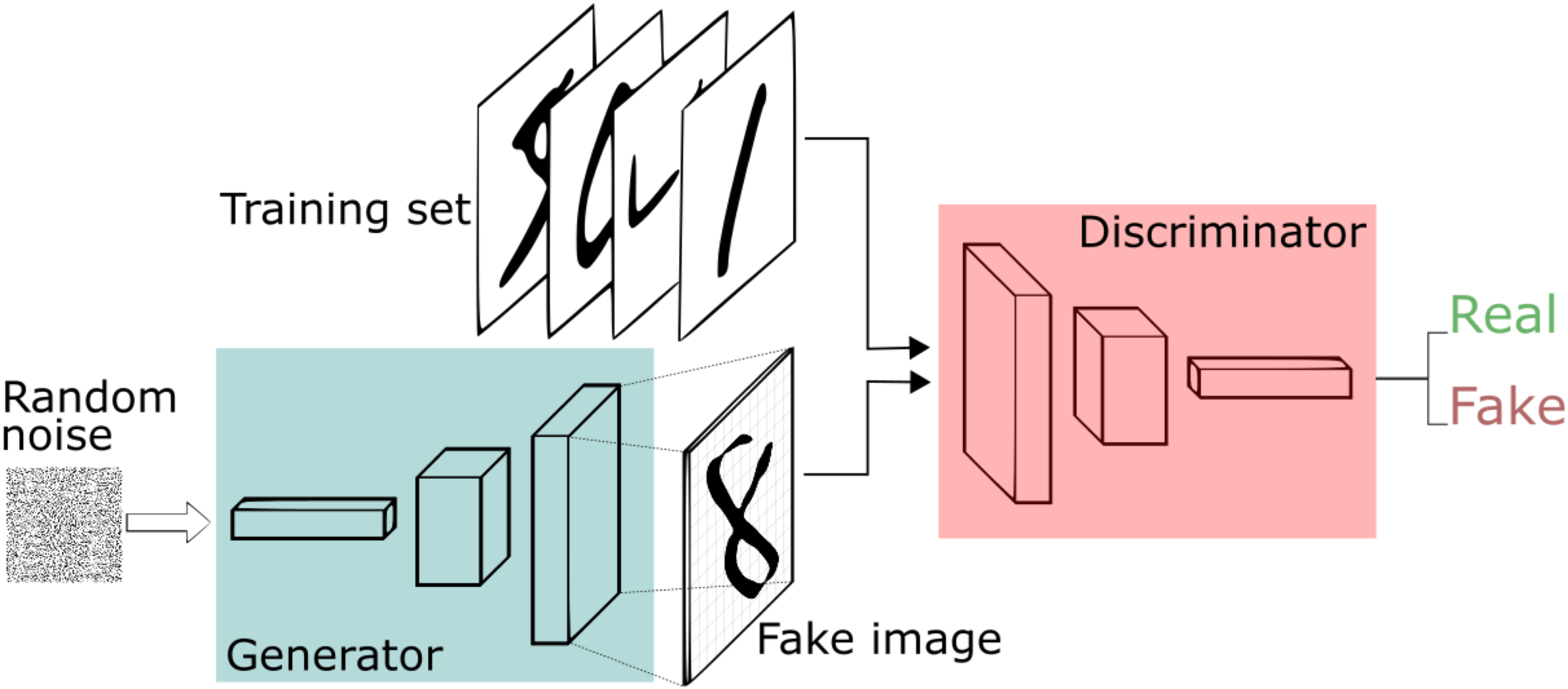


Intuition behind GANs

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



Another example



Training GANs

Given the Generator:

$G(z, \Theta_g), G: z \rightarrow x,$

Its goal is:
$max\ D(G(z))$

Given the Discriminator

$D(x, \Theta_d), D: x \rightarrow (0,1)$

Its goal is:
$max\ D(x), \min D(G(z))$

After training:
- *G* produces realistic synthetic data
- *D* is unable to distinguish real from fake

Training GANs

$$G^* = \arg\min_G \max_D v(G, D)$$

Input samples

Input random noise samples

$$\nabla_{W_D} \frac{1}{n} \sum_{i=1}^{n} [log(D(x_i)) + log(1 - D(G(z_i)))]$$

This predicts well
on real images

This predicts well
on fake images

$$\nabla_{W_G} \frac{1}{n} \sum_{i=1}^{n} log\left(1 - D(G(z_i))\right)$$

This predicts badly on
fake images

Training GANs

$$G^* = \arg\min_G \max_D v(G, D)$$

- The optimization drives the discriminator to learn to correctly classify samples as real or fake. Simultaneously, the generator attempts to fool the classifier

- At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs 0.5 everywhere

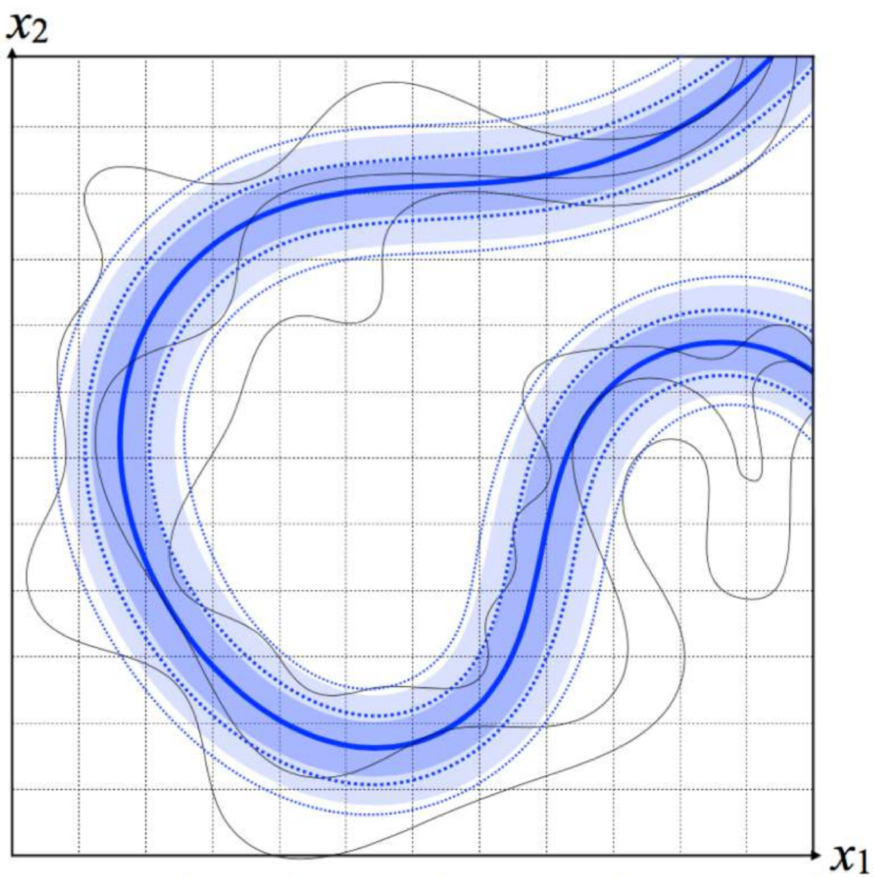
Training GANs

$$G^* = \arg\min_G \max_D v(G, D)$$

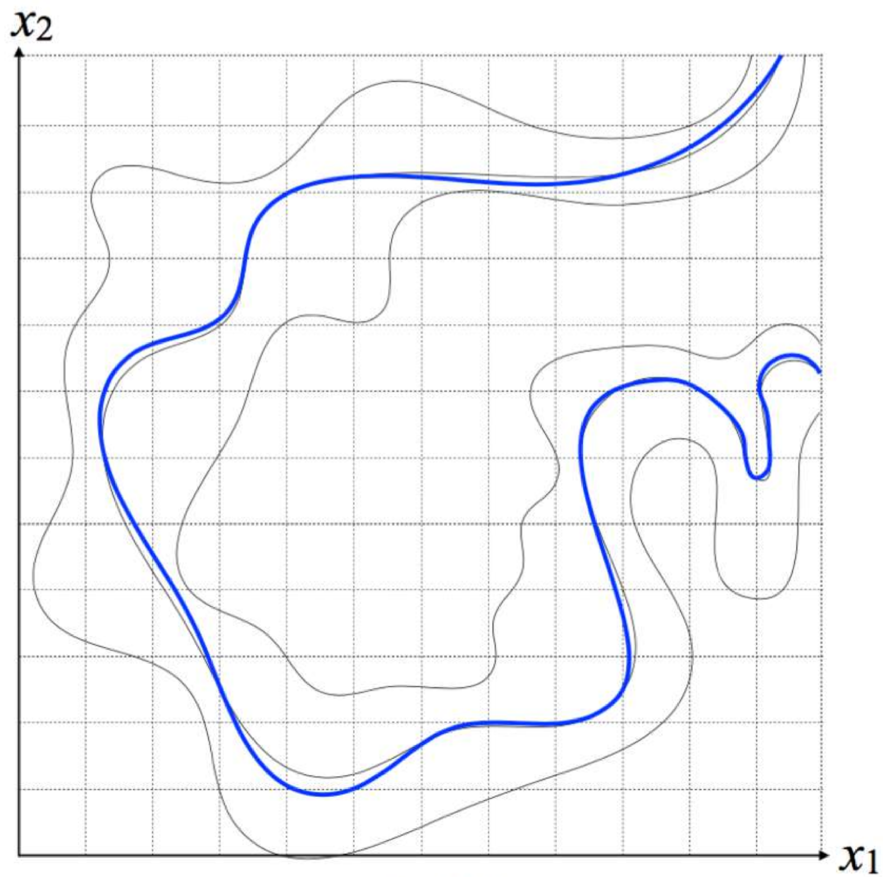
- In other words the convergence is reached when the actions of one of the players do not change depending on the actions of the other players

- As you can imagine, training can be very slow

Why GANs?



more traditional max-likelihood approach

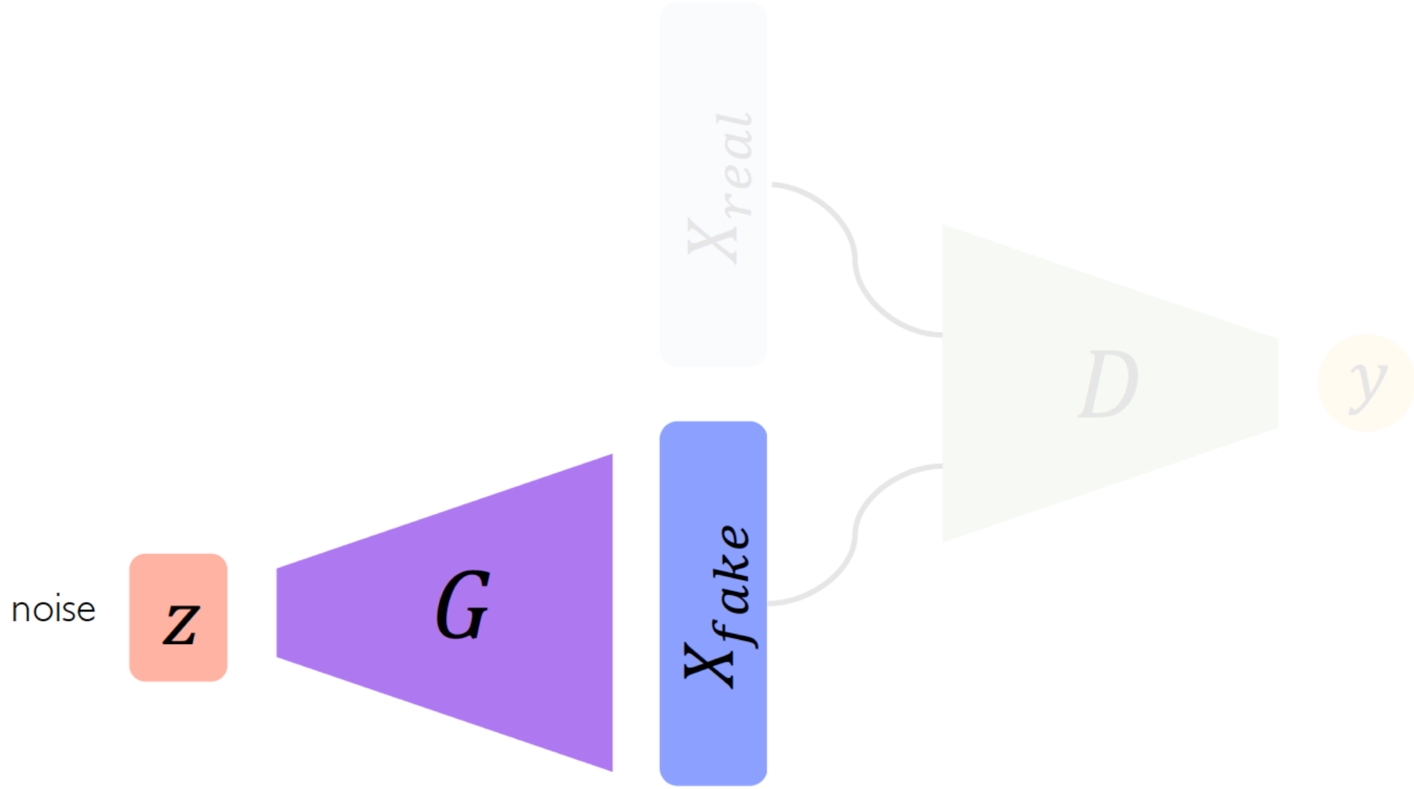


GAN

A. Courville, 6S191 2018.

Generating new data with GANs

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



GANs: Recent Advances

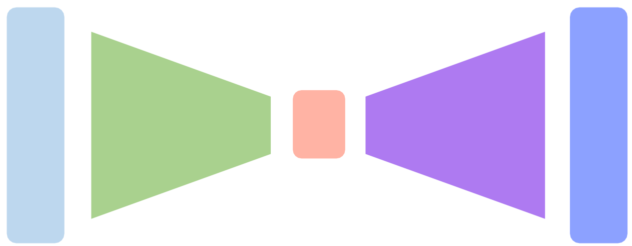
Some Impressive application

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Deep Generative Modeling: 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