

## 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!

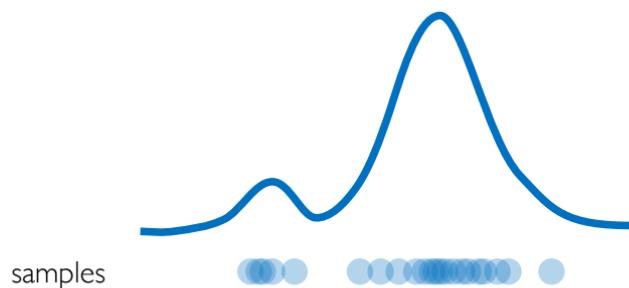
**Goal:** Learn some *hidden* or  
*underlying* structure of the data

**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**



**Sample Generation**



Input samples

Generated samples

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

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

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

# Generative models

## Why generative models? Debiasing

Capable of uncovering **underlying latent variables** in a dataset



Homogeneous skin color, pose

VS

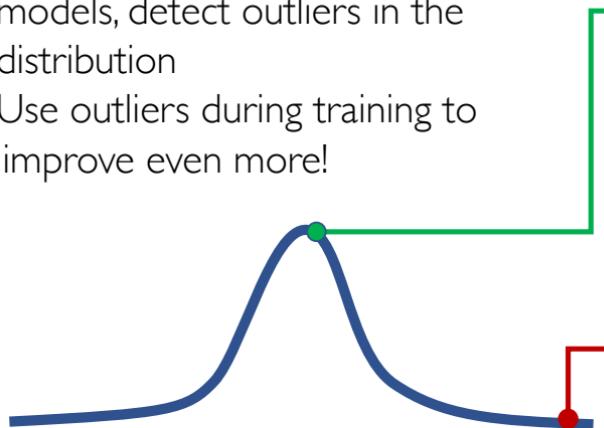


Diverse skin color, pose, illumination

How can we use latent distributions to create fair and representative datasets?

## Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!



**95% of Driving Data:**

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



Detect outliers to avoid unpredictable behavior when training



Edge Cases



Harsh Weather

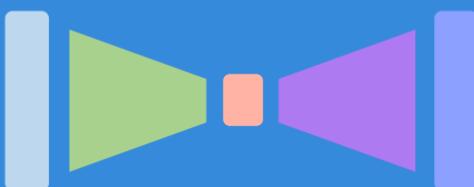


Pedestrians

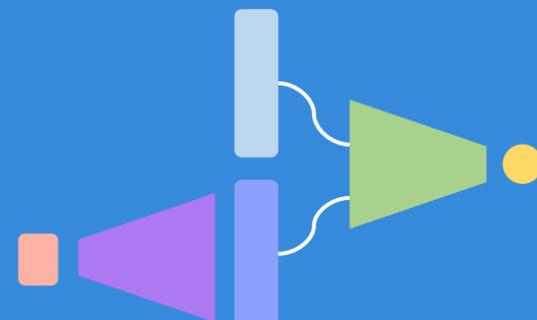
# Generative models

## Latent variable models

Autoencoders and Variational  
Autoencoders (VAEs)



Generative Adversarial  
Networks (GANs)

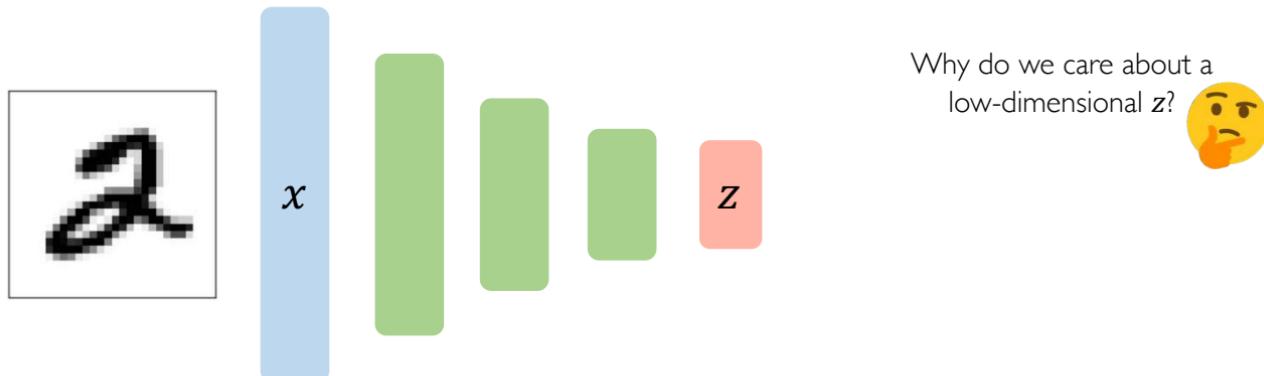


## Latent variable?

True explanatory factor – learnt only from observed data

## Autoencoders: background

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

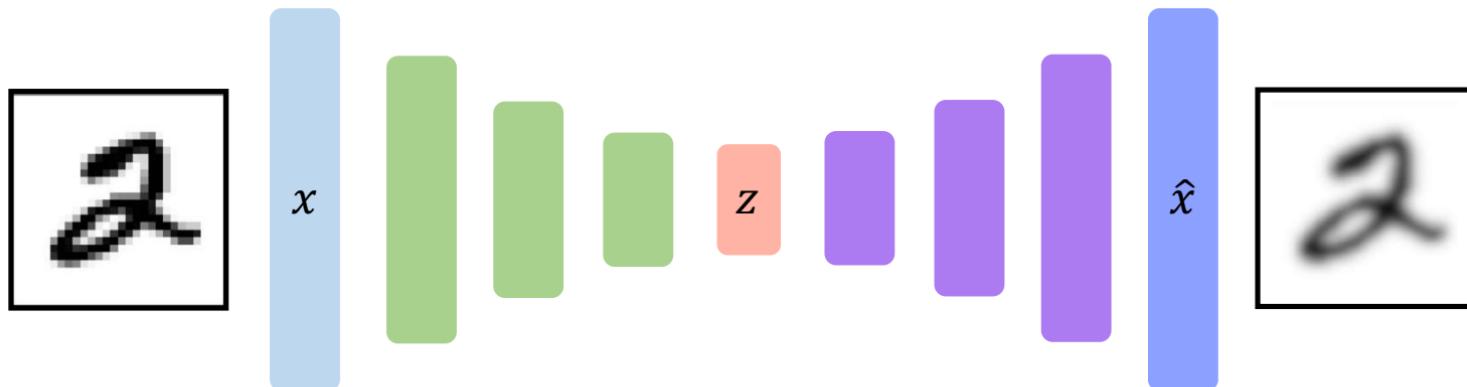


“Encoder” learns mapping from the data,  $x$ , to a low-dimensional latent space,  $z$

## Autoencoders: background

How can we learn this latent space?

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

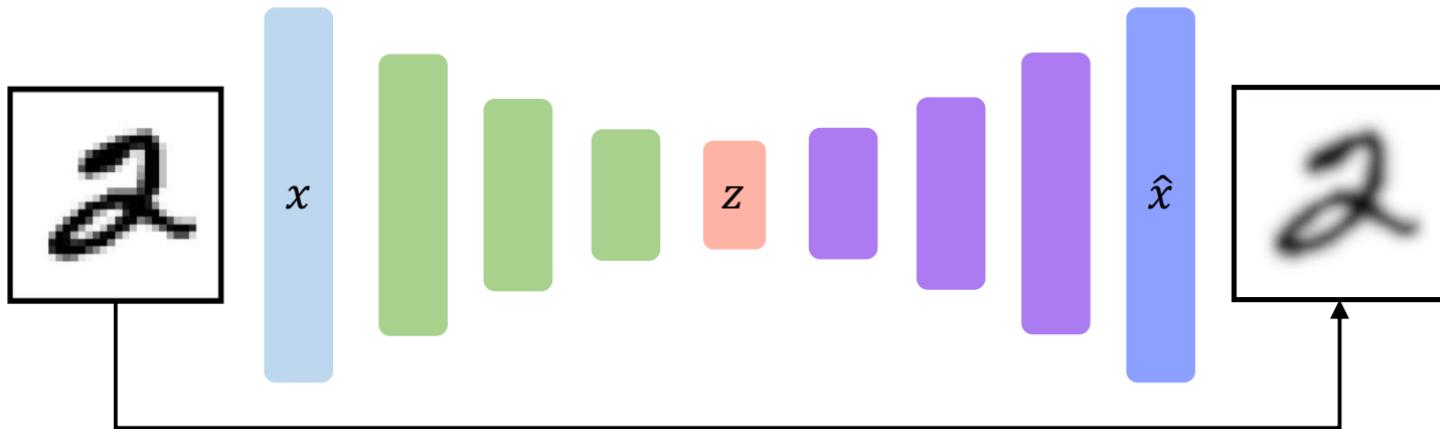


“Decoder” learns mapping back from latent,  $z$ , to a reconstructed observation,  $\hat{x}$

## Autoencoders: background

How can we learn this latent space?

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



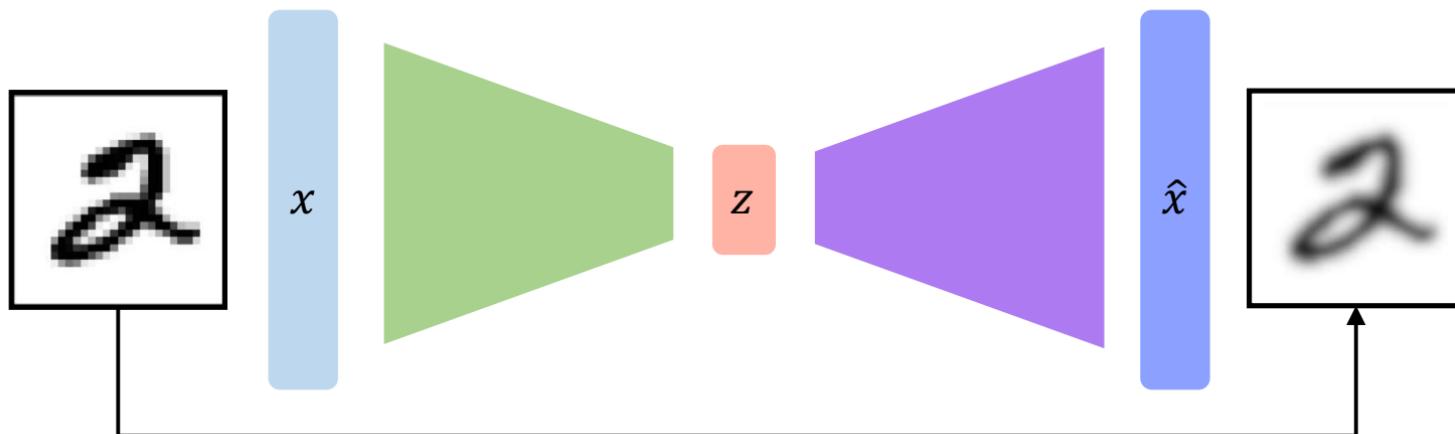
$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Loss function doesn't  
use any labels!!

## 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!!

# Autoencoders

## Dimensionality of latent space → reconstruction quality

Autoencoding is a form of compression!

Smaller latent space will force a larger training bottleneck

2D latent space

7	2	/	0	9	/	9	9	8	9
0	6	9	0	1	5	9	7	8	9
9	6	6	5	4	0	7	4	0	1
3	1	3	0	7	3	7	1	2	1
1	7	9	2	3	5	1	2	9	4
6	3	5	5	6	0	4	1	9	8
7	8	9	3	7	9	6	4	3	0
7	0	2	9	1	9	3	2	9	7
9	6	2	7	3	9	7	3	6	1
3	6	9	3	1	9	1	7	6	9

5D latent space

7	2	/	0	9	/	4	9	9	9
0	6	9	0	1	5	9	7	3	4
9	6	6	5	4	0	7	4	0	1
3	1	3	4	7	2	7	1	2	1
1	7	4	2	3	5	1	2	9	4
6	3	5	5	6	0	4	1	9	8
7	8	9	3	7	4	6	4	3	0
7	0	2	9	1	7	3	2	9	7
9	6	2	7	8	4	7	3	6	1
3	6	9	3	1	4	1	7	6	9

Ground Truth

7	2	/	0	4	/	4	9	5	9
0	6	9	0	1	5	9	7	3	4
9	6	6	5	4	0	7	4	0	1
3	1	3	4	7	2	7	1	2	1
1	7	4	2	3	5	1	2	4	4
6	3	5	5	6	0	4	1	9	5
7	8	9	3	7	4	6	4	3	0
7	0	2	9	1	7	3	2	9	7
9	6	2	7	8	4	7	3	6	1
3	6	9	3	1	4	1	7	6	9

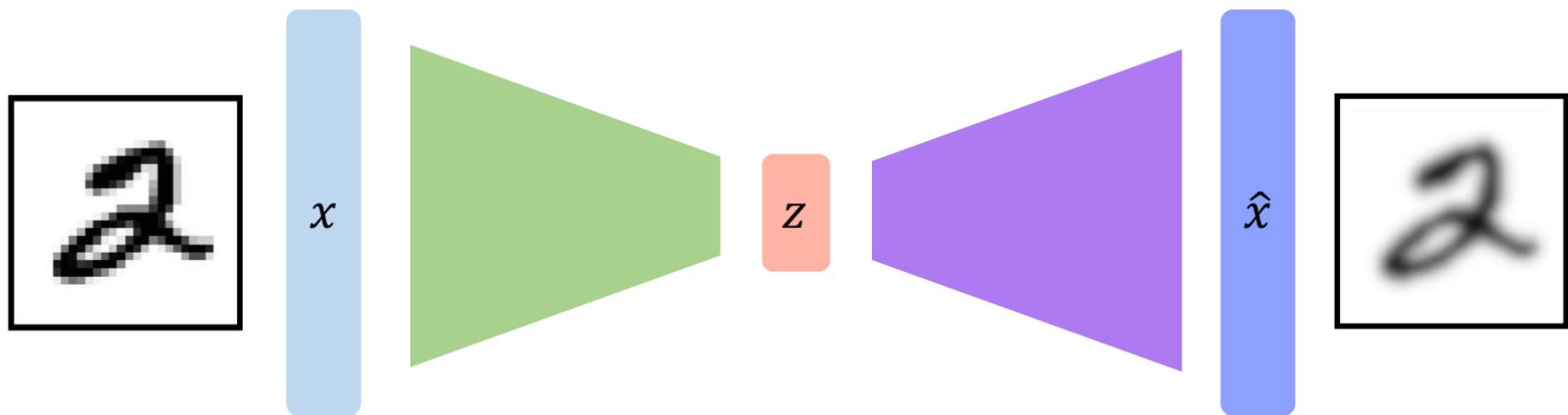
## Autoencoders for representation learning

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

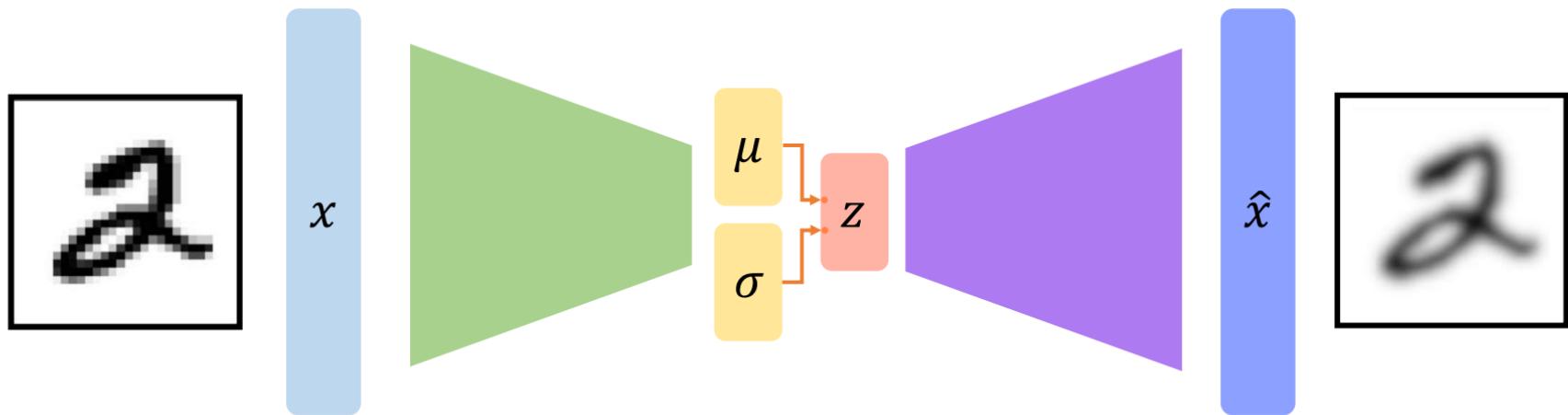
**Reconstruction loss** forces the latent representation to capture (or encode) as much “information” about the data as possible

**Autoencoding** = **A**uto**m**atically **e**nco**d**ing data

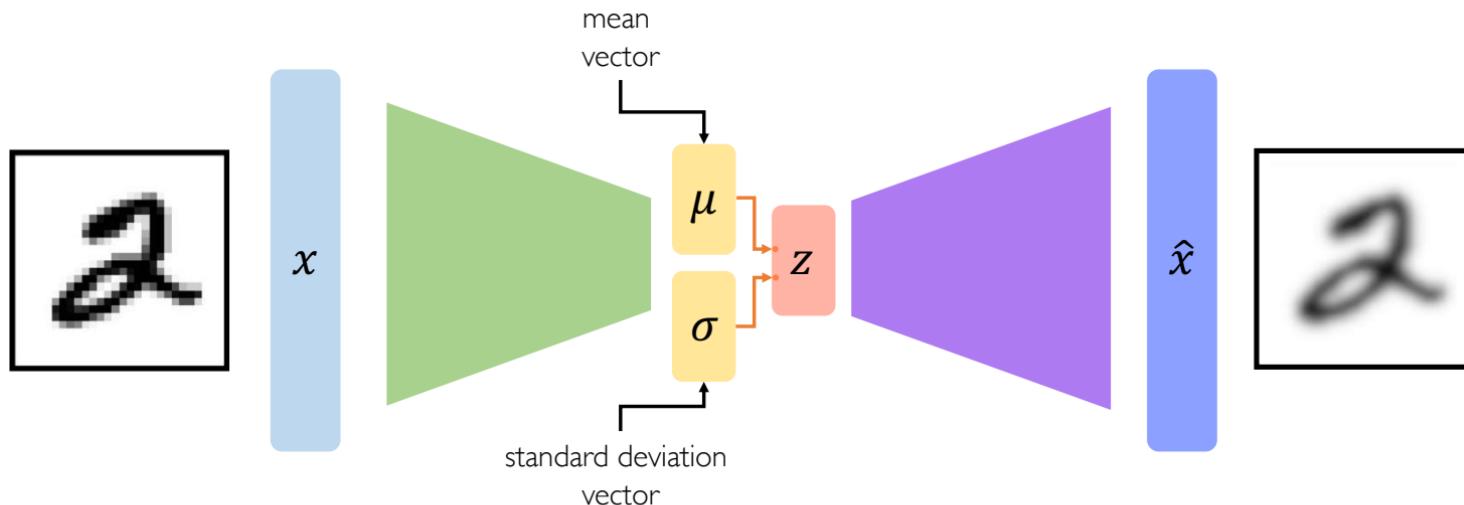
## VAEs: key difference with traditional autoencoder



## VAEs: key difference with traditional autoencoder



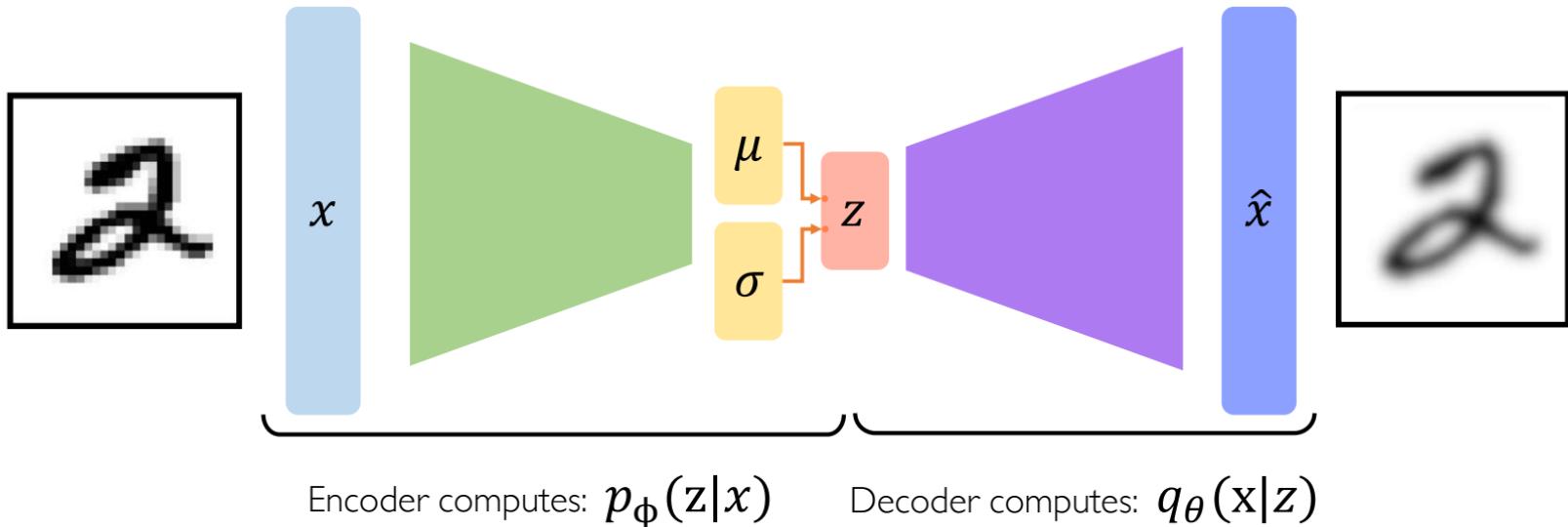
## VAEs: key difference with traditional autoencoder



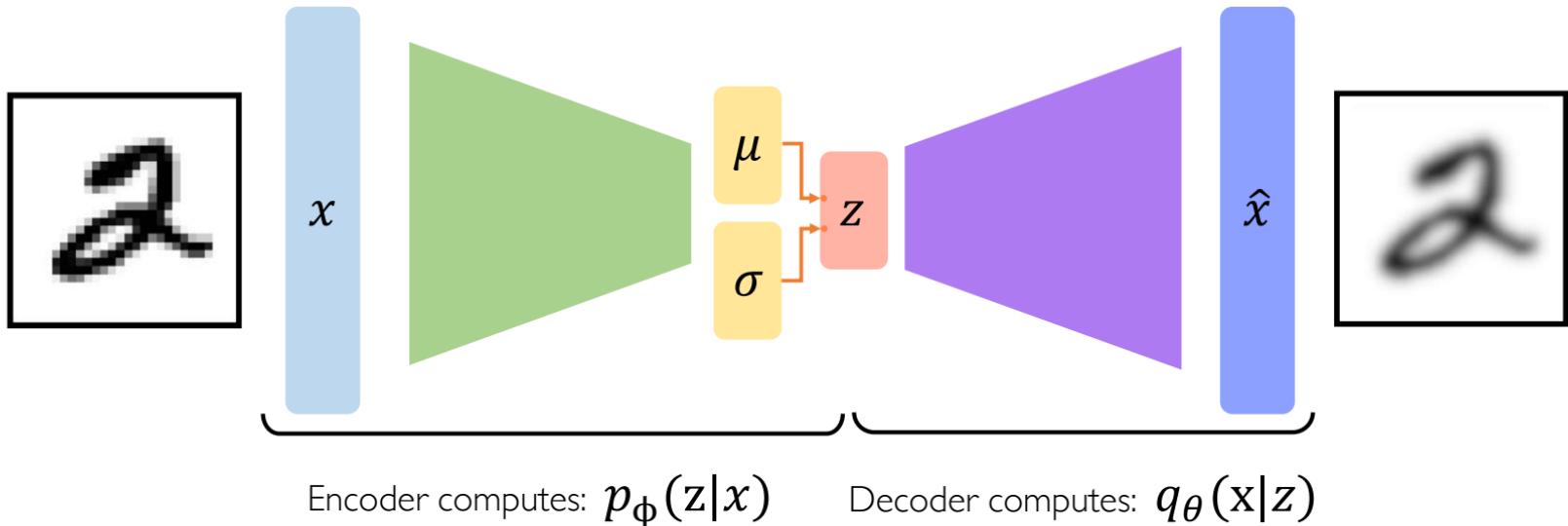
**Variational autoencoders are a probabilistic twist on autoencoders!**

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

## VAE optimization

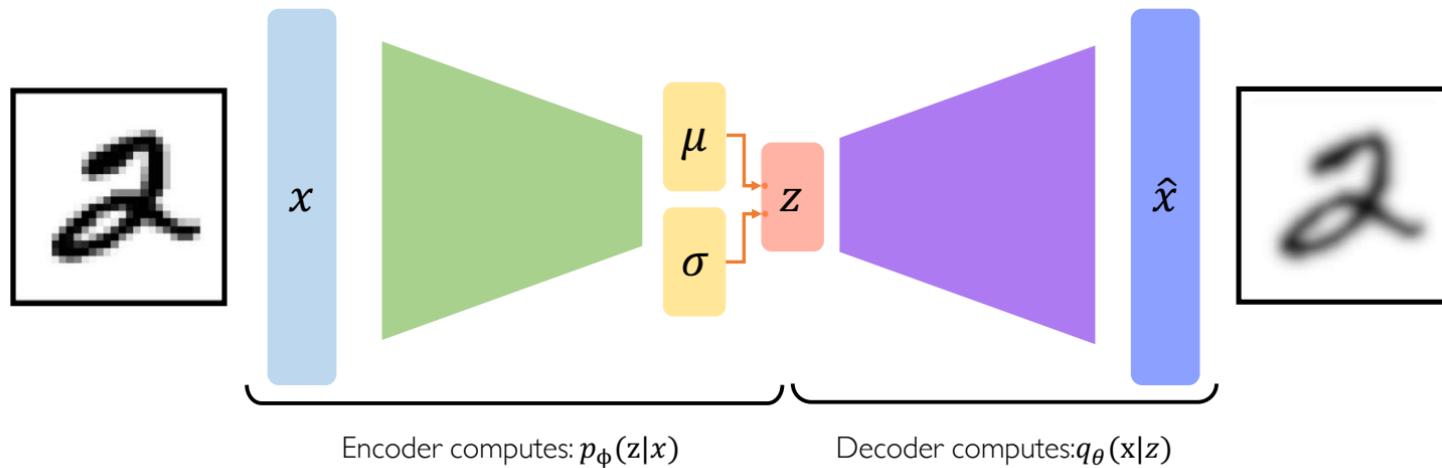


## VAE optimization



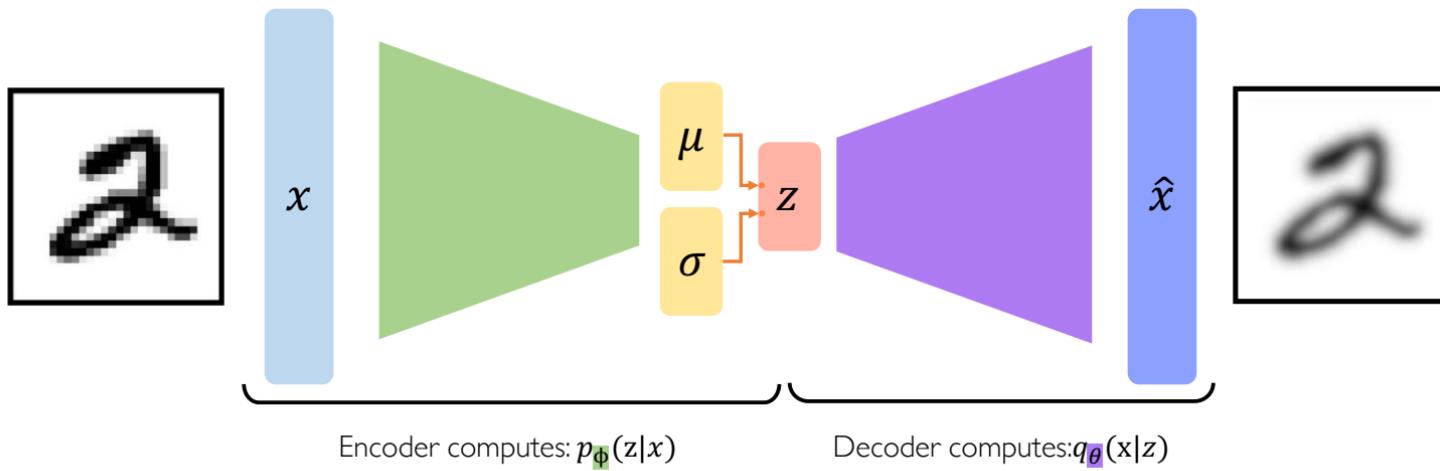
# Variational Autoencoders

## VAE optimization



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

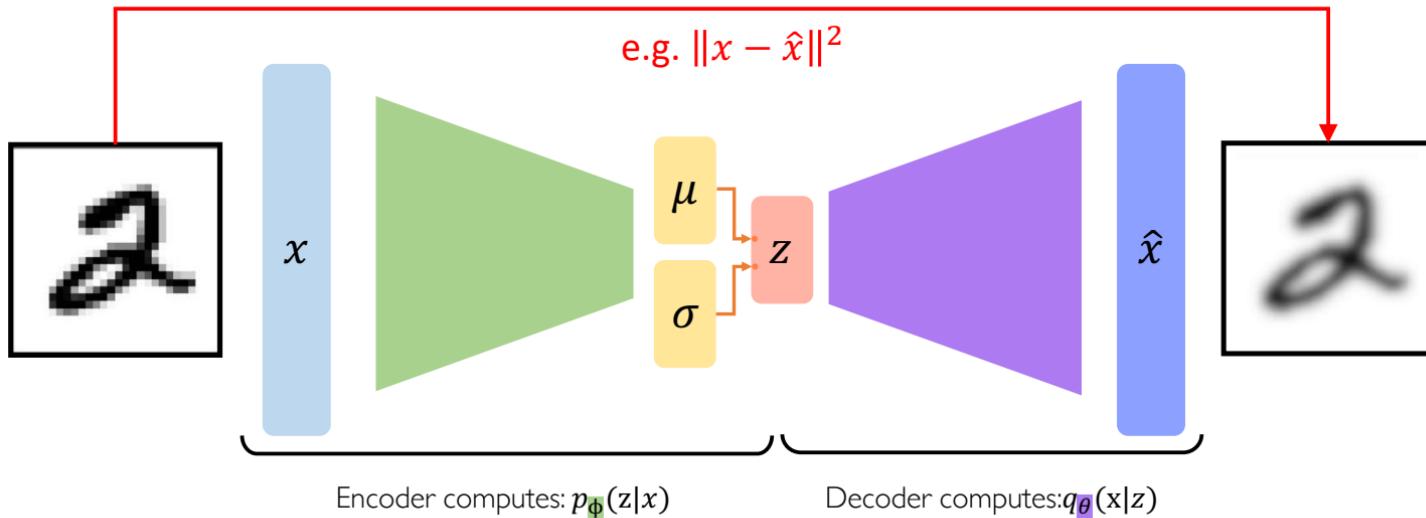
## VAE optimization



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

# Variational Autoencoders

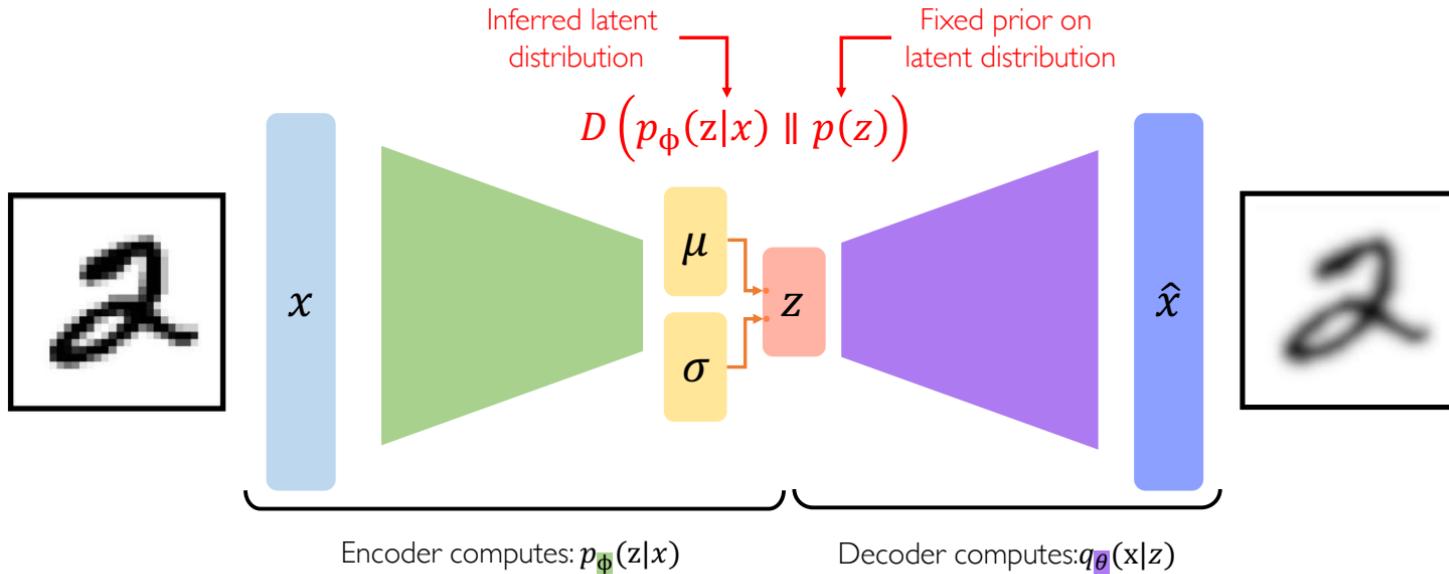
## VAE optimization



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

# Variational Autoencoders

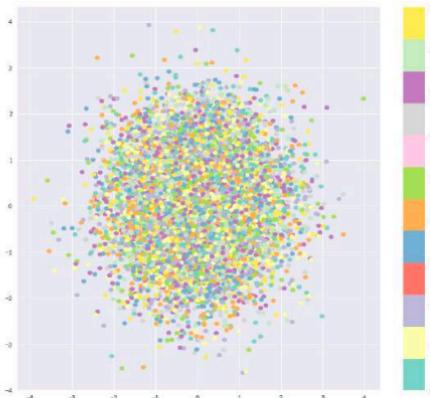
## VAE optimization



## Priors on the latent distribution

$$D(p_{\Phi}(z|x) \parallel p(z))$$

Inferred latent distribution      ↑      Fixed prior on latent distribution



**Common choice of prior:**

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

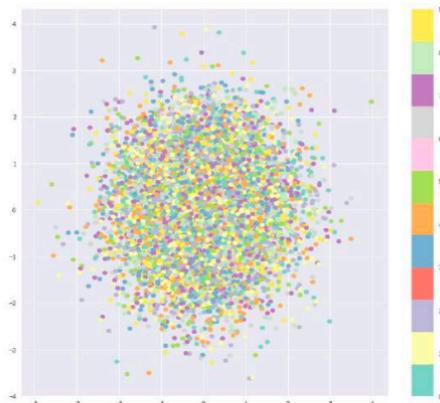
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to “cheat” by clustering points in specific regions (ie. memorizing the data)

## Priors on the latent distribution

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

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

KL-divergence between  
the two distributions



Common choice of prior:

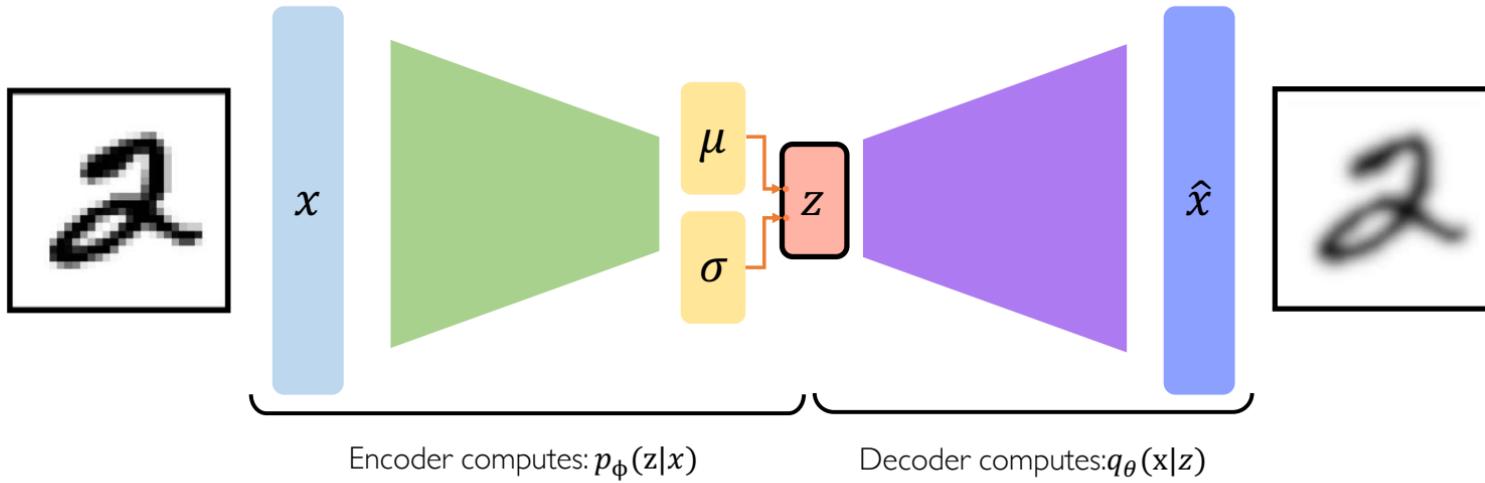
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Encourages encodings to distribute 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)

# Variational Autoencoders

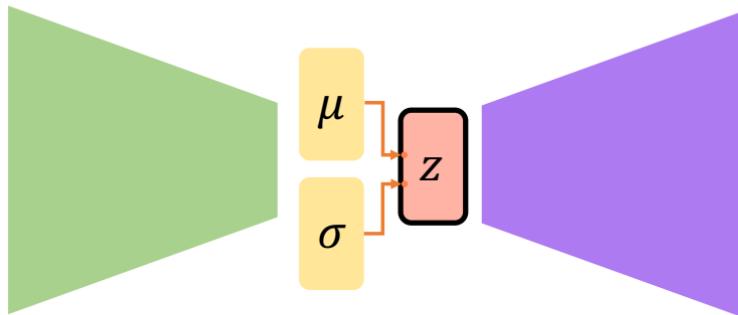
## VAEs computation graph

**Problem:** We cannot backpropagate gradients through sampling layers!



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

## Reparametrizing the sampling layer



**Key Idea:**

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

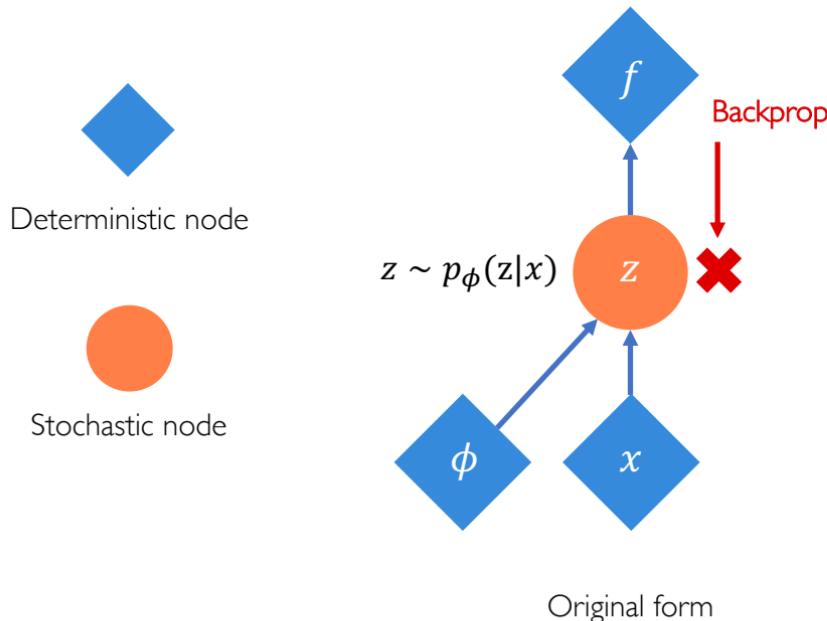
Consider the sampled latent vector as a sum of

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

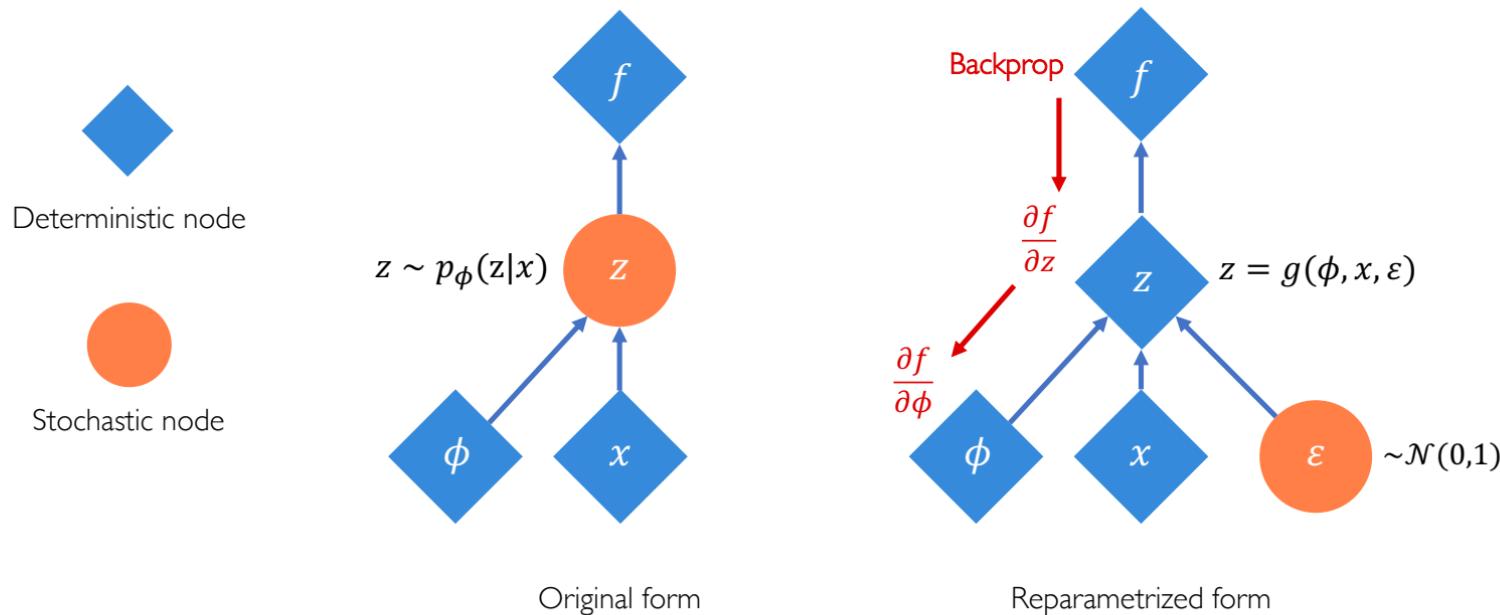
$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

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

## Reparametrizing the sampling layer

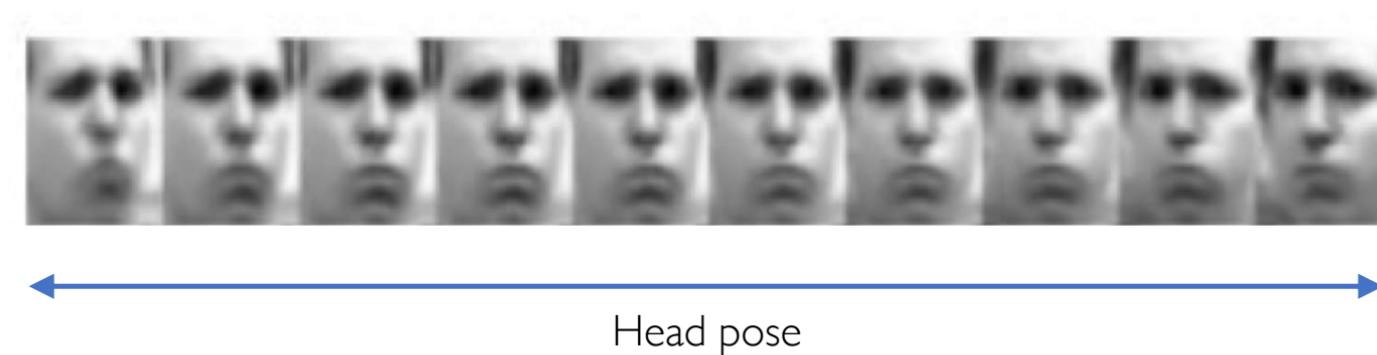


## Reparametrizing the sampling layer



## VAEs: Latent perturbation

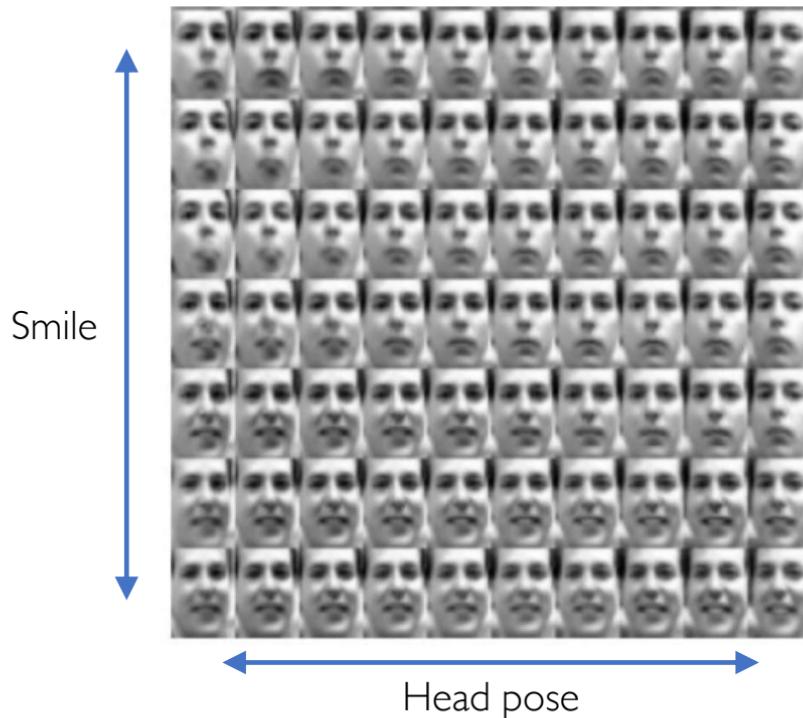
Slowly increase or decrease a **single latent variable**  
Keep all other variables fixed



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

# Variational Autoencoders

## VAEs: Latent perturbation



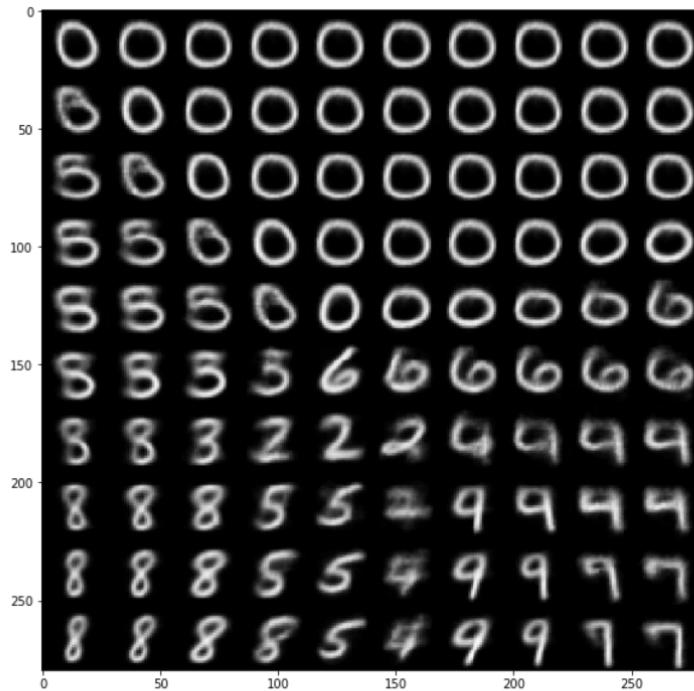
Ideally, we want latent variables that are uncorrelated with each other

Enforce diagonal prior on the latent variables to encourage independence

**Disentanglement**

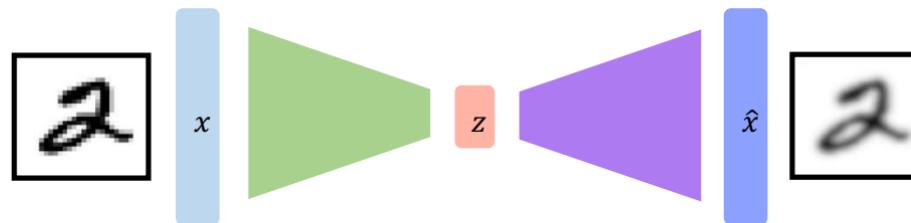
# Variational Autoencoders

## 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!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples



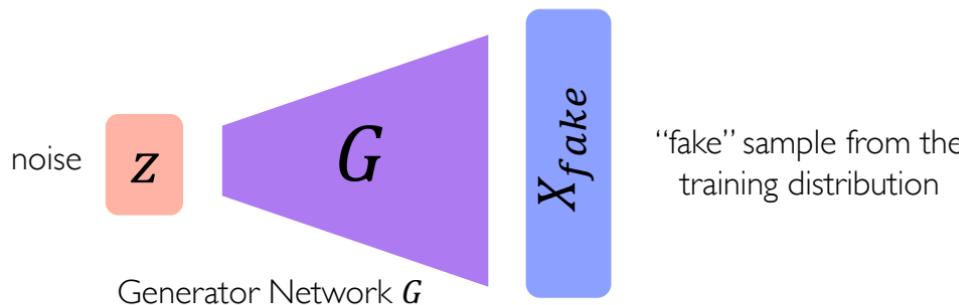
# Generative Adversarial Networks (GANs)

## 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!

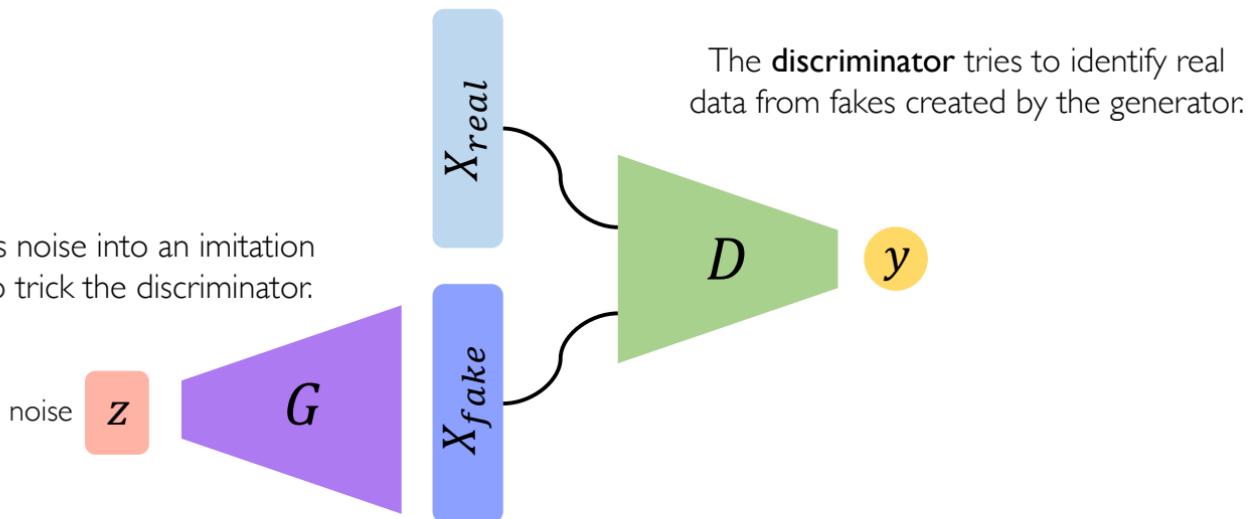
**Solution:** sample from something simple (noise), learn a transformation to the training distribution.



## 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.

The **generator** turns noise into an imitation of the data to try to trick the discriminator.



## Intuition behind GANs

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

Generator



## Intuition behind GANs

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

Discriminator



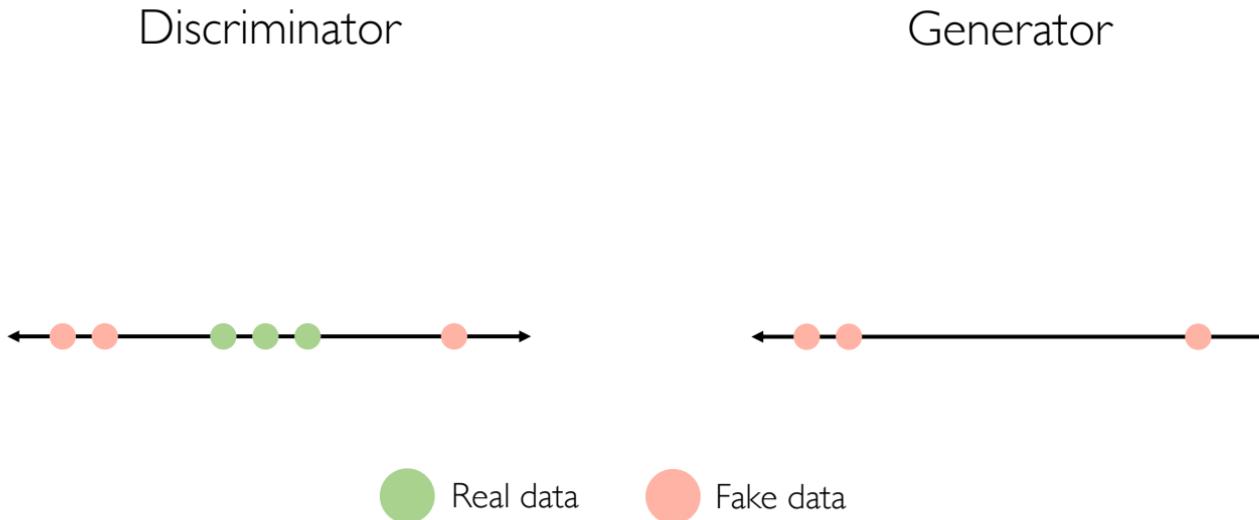
Generator



Fake data

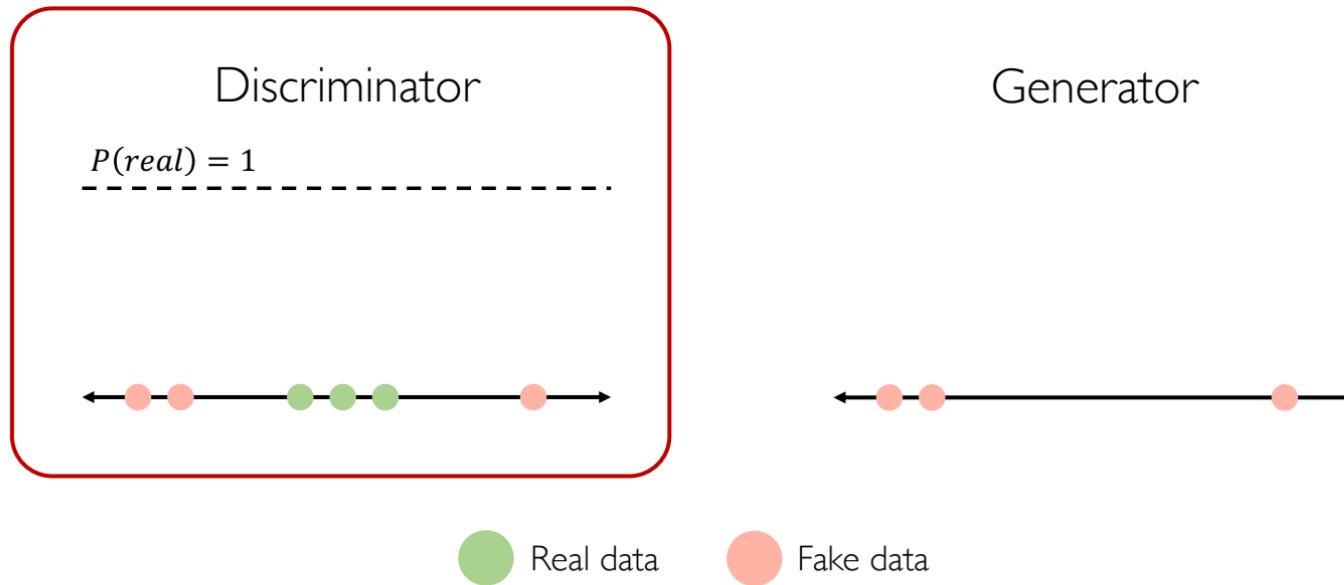
## Intuition behind GANs

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



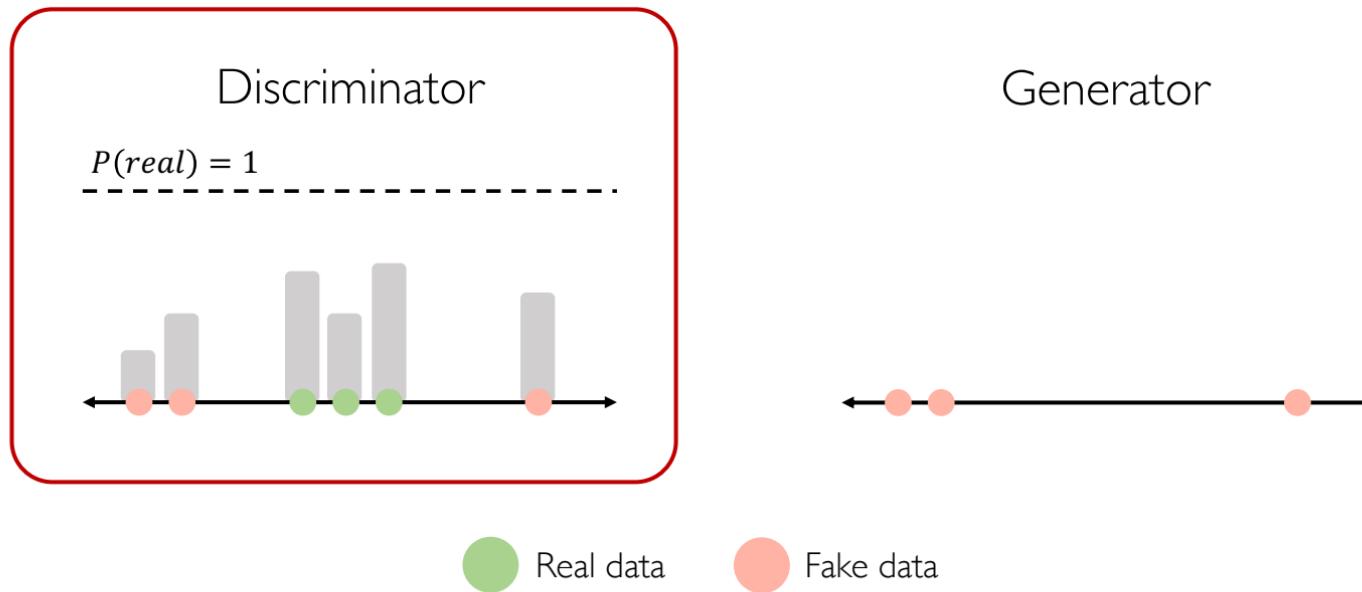
## Intuition behind GANs

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



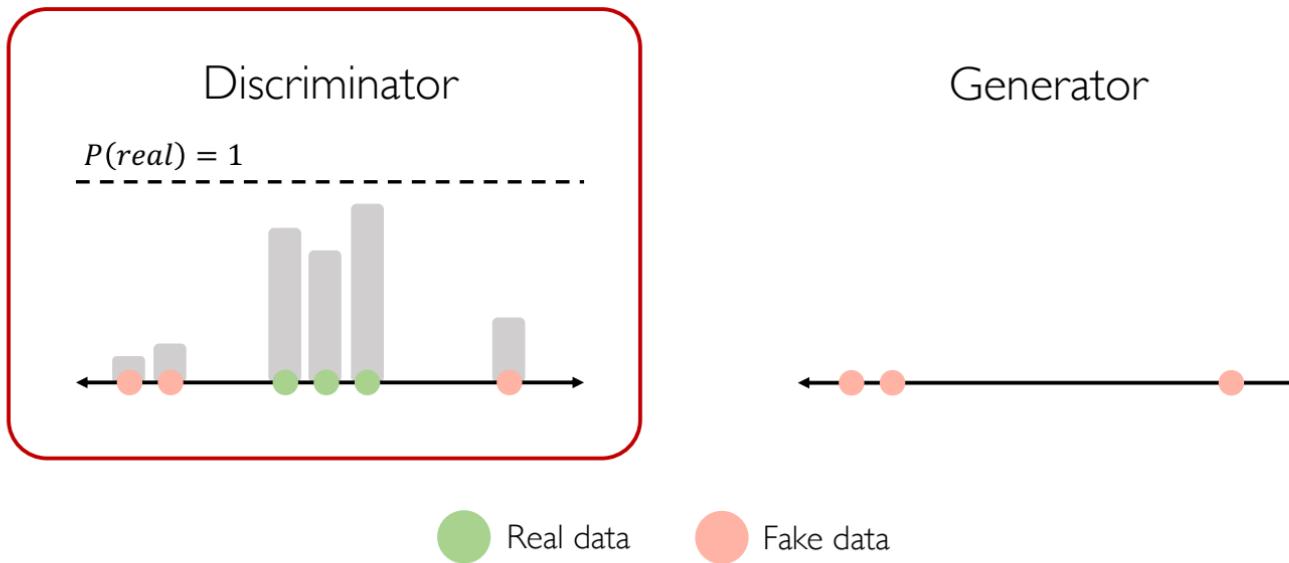
## Intuition behind GANs

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



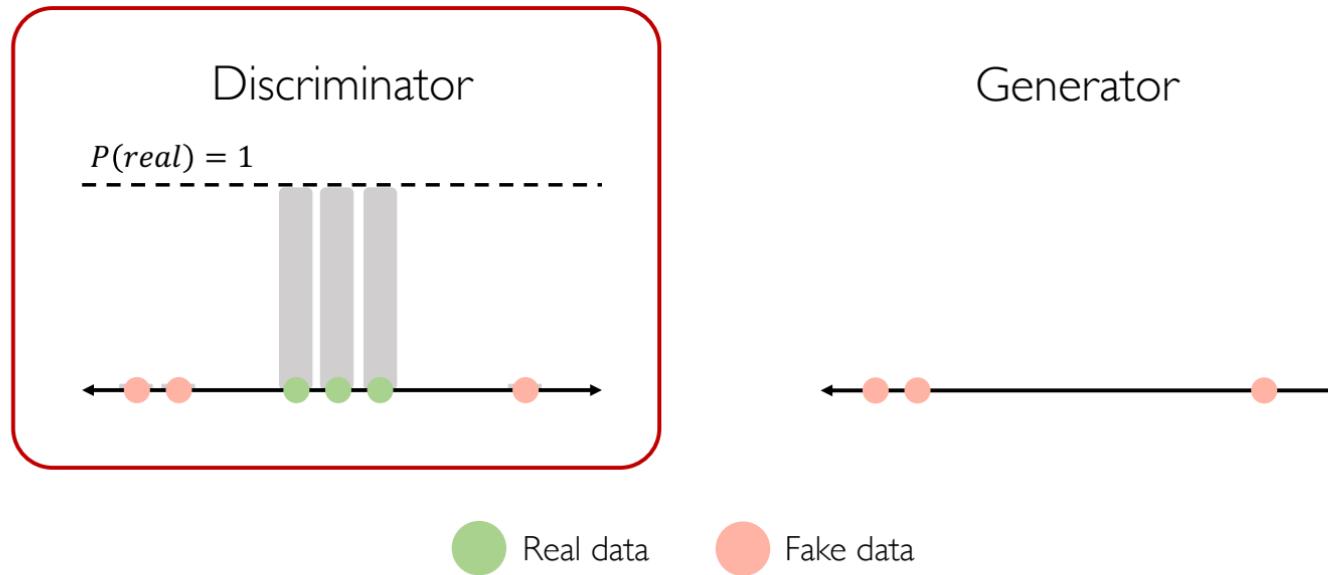
## Intuition behind GANs

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



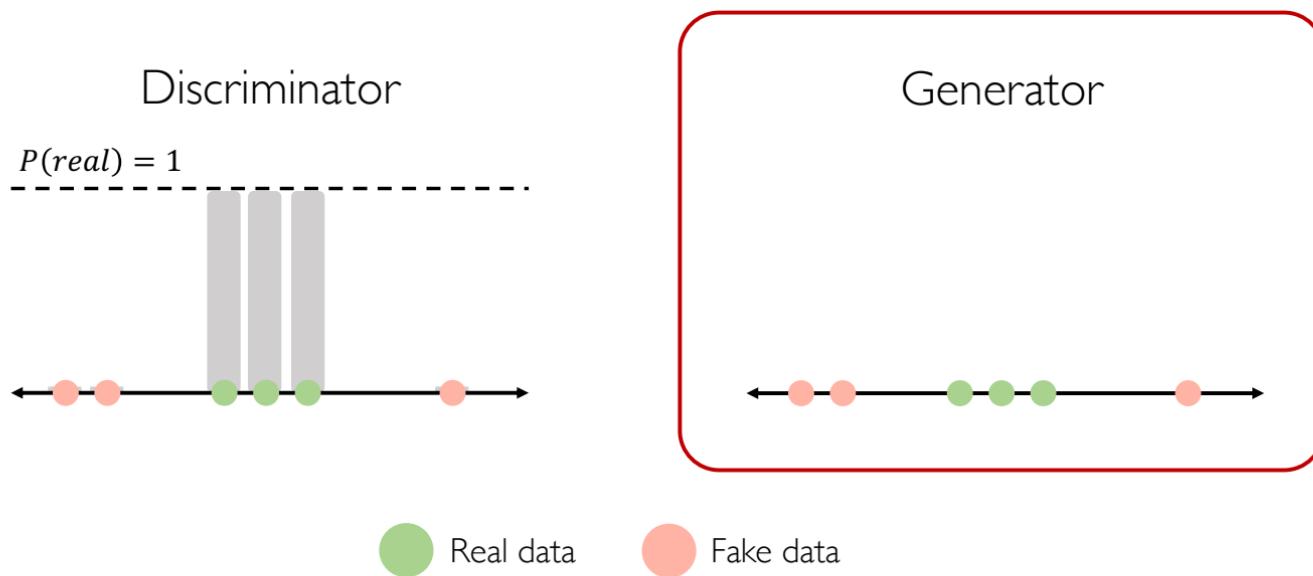
## Intuition behind GANs

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



## Intuition behind GANs

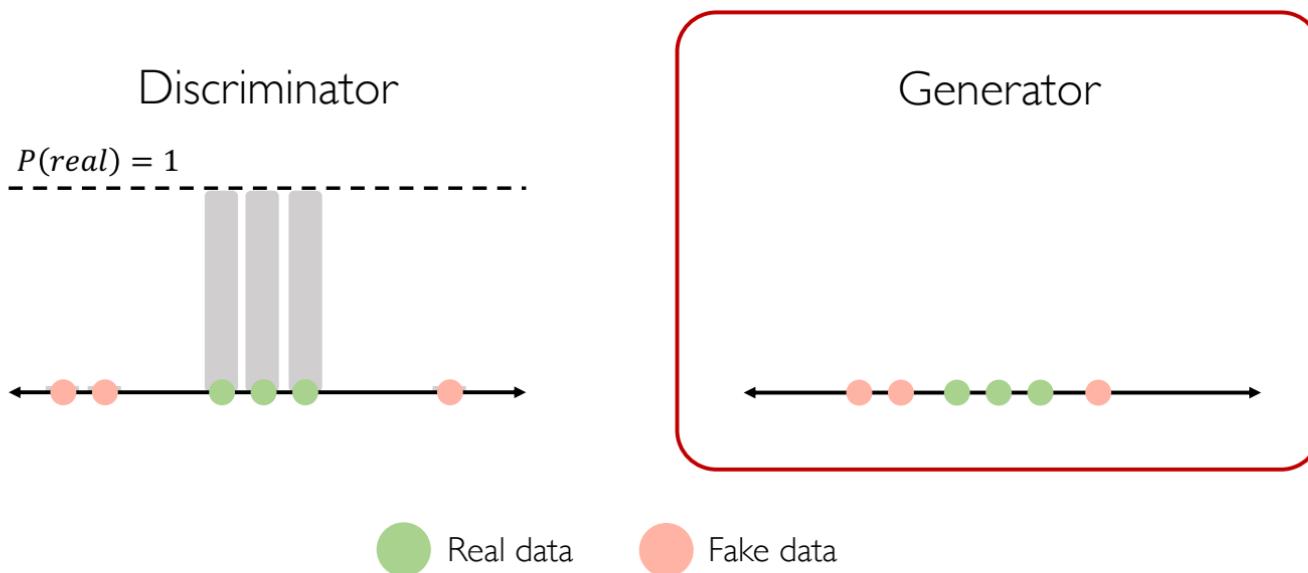
**Generator** tries to improve its imitation of the data.



# Generative Adversarial Networks (GANs)

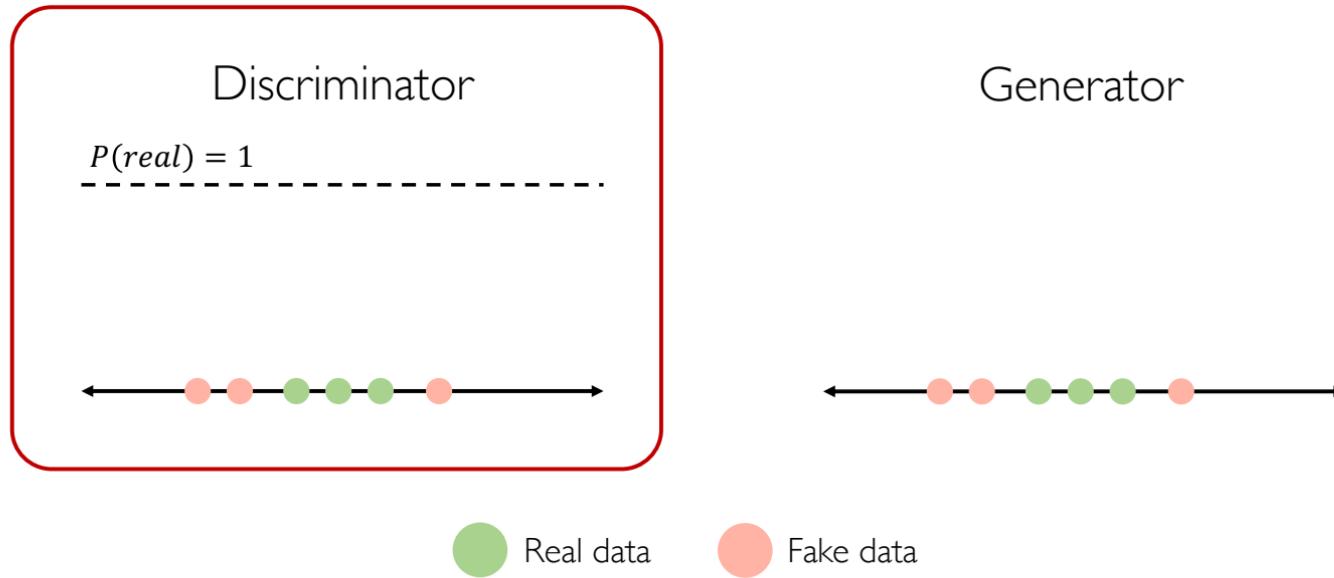
## Intuition behind GANs

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



## Intuition behind GANs

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

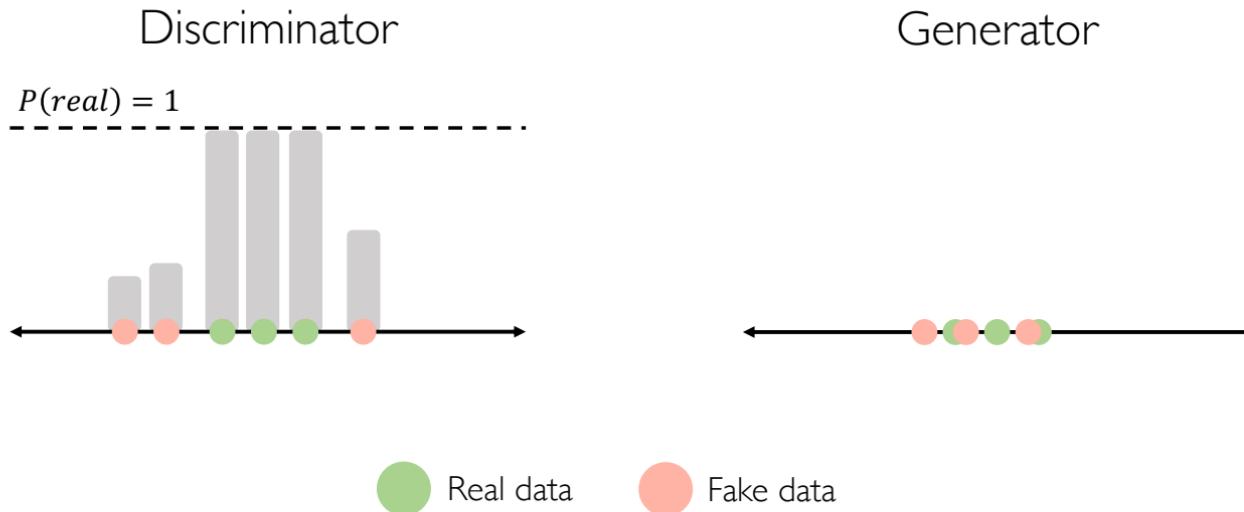


# Generative Adversarial Networks (GANs)

## 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.



## Training GANs

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

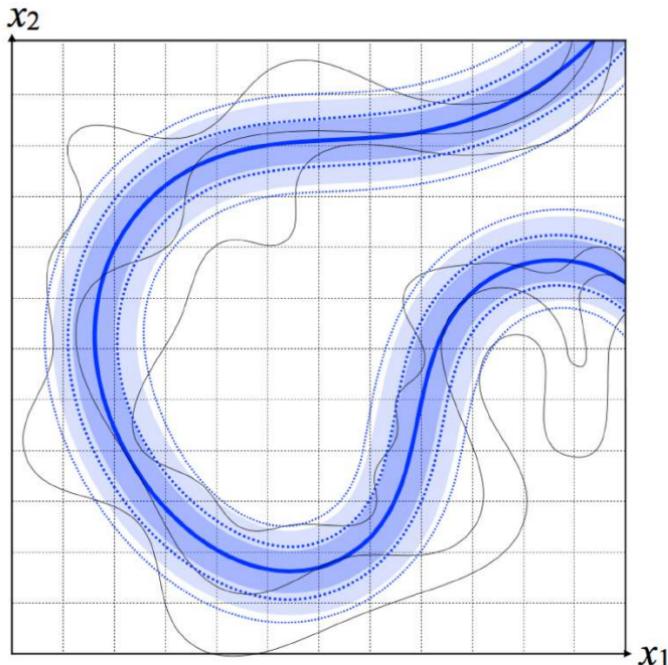
Train GAN jointly via **minimax** game:

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

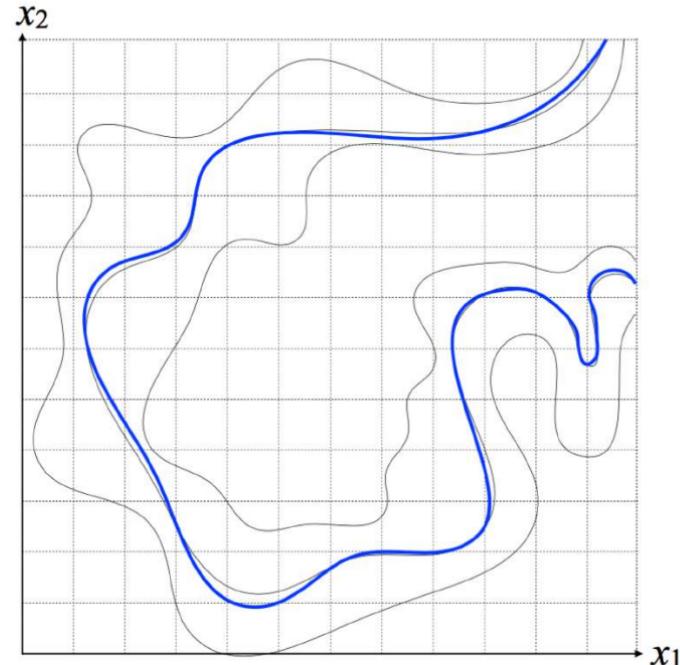
**Discriminator** wants to maximize objective s.t.  $D(x)$  close to 1,  $D(G(z))$  close to 0.  
**Generator** wants to minimize objective s.t.  $D(G(z))$  close to 1.

# Generative Adversarial Networks (GANs)

## Why GANs?



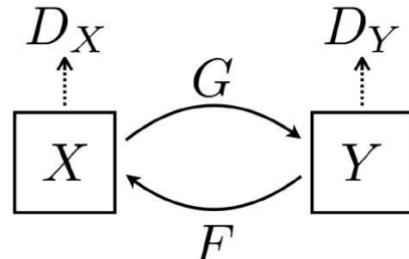
more traditional max-likelihood approach



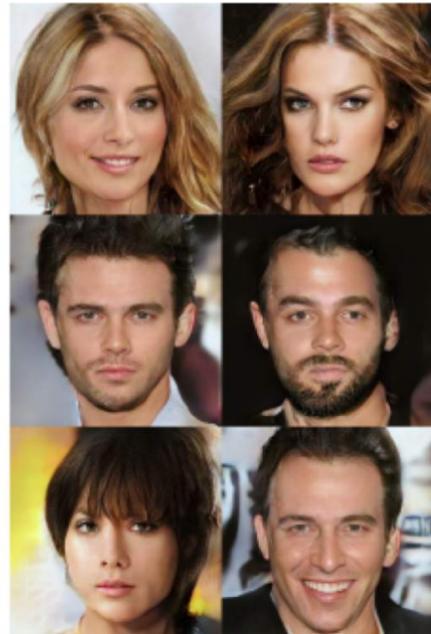
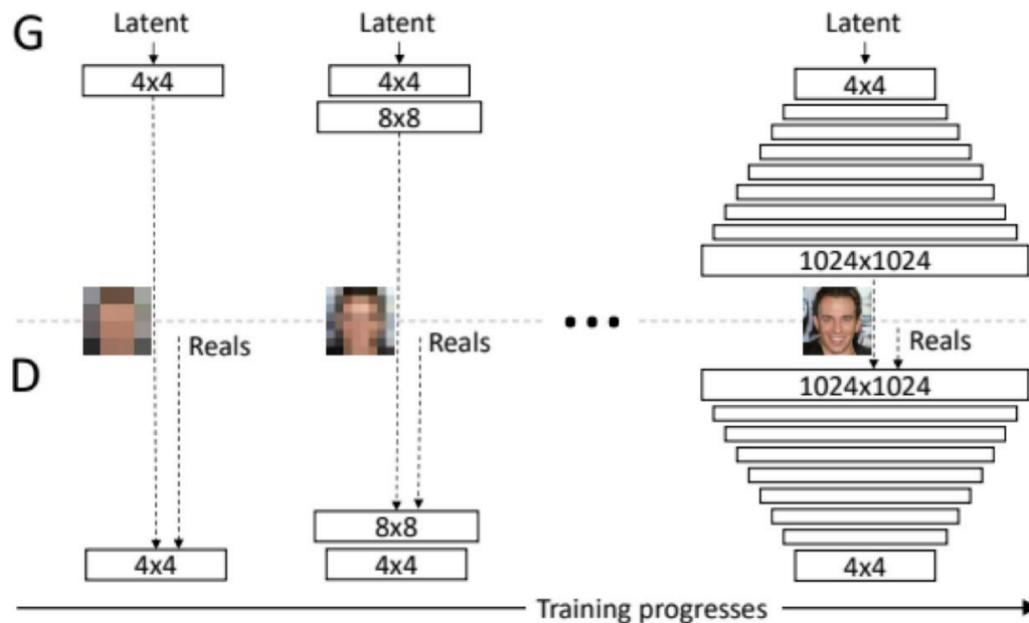
GAN

## CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.



## Progressive growing of GANs (NVIDIA)



Karras et al., ICLR 2018.