SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

Generative Modeling

What can we do with a generative model?

- **Discriminative Model:** Learn a probability distribution p(y|x)

→ Assign labels to data
Feature learning (with labels)

- **Generative Model:** Learn a probability distribution p(x)

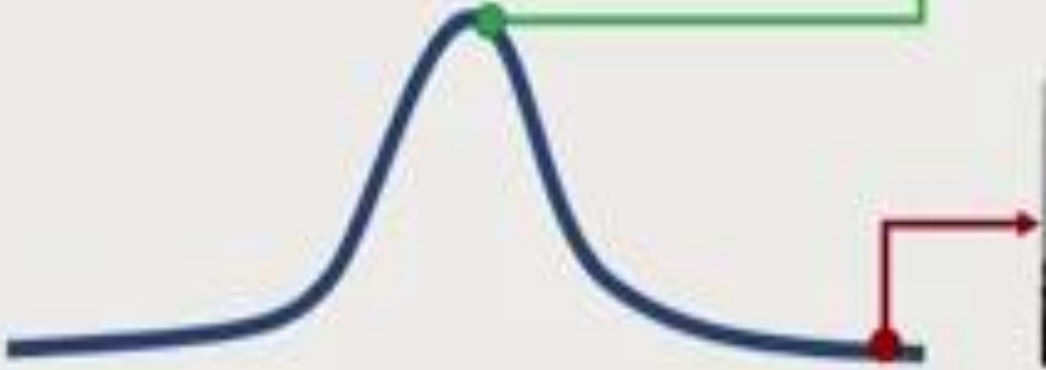
→ Detect outliers
Feature learning (without labels)
Sample to generate new data

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



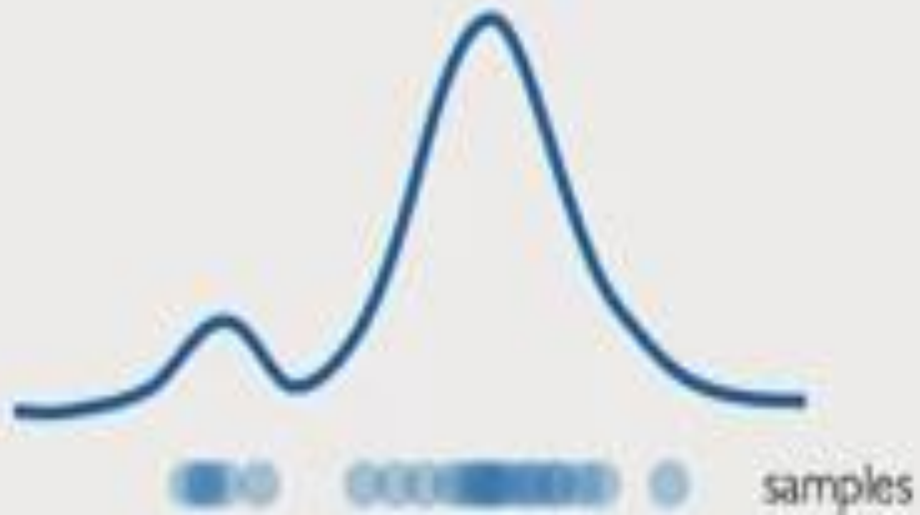
Edge Cases

Harsh Weather

Pedestrians

Why generative models? Sample generation

- **Generative models learn probability distributions**
- **Sampling** from that distribution → new data instances
- **Backbone of Generative AI:** generate language, images, and more



Natural Language

Images & Videos

Biology

Taxonomy of Generative Models

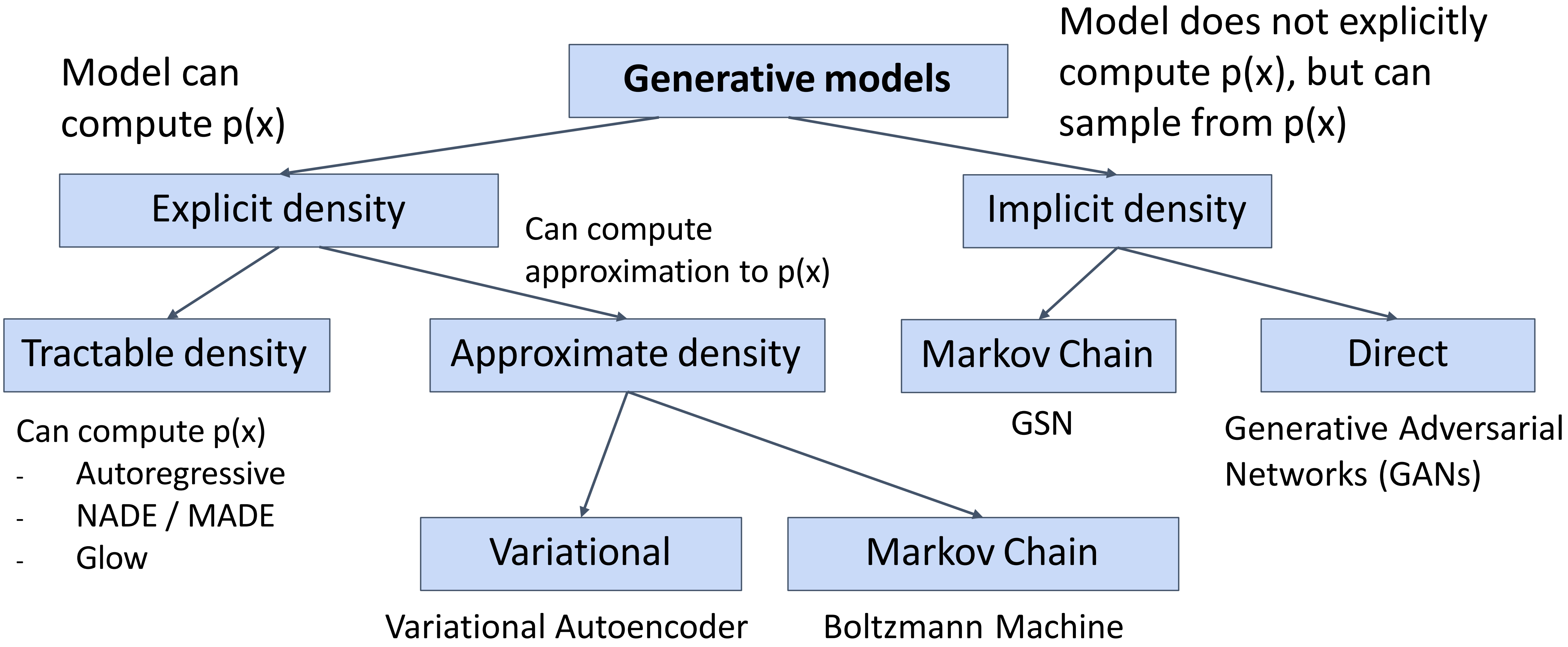


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Latent variable models

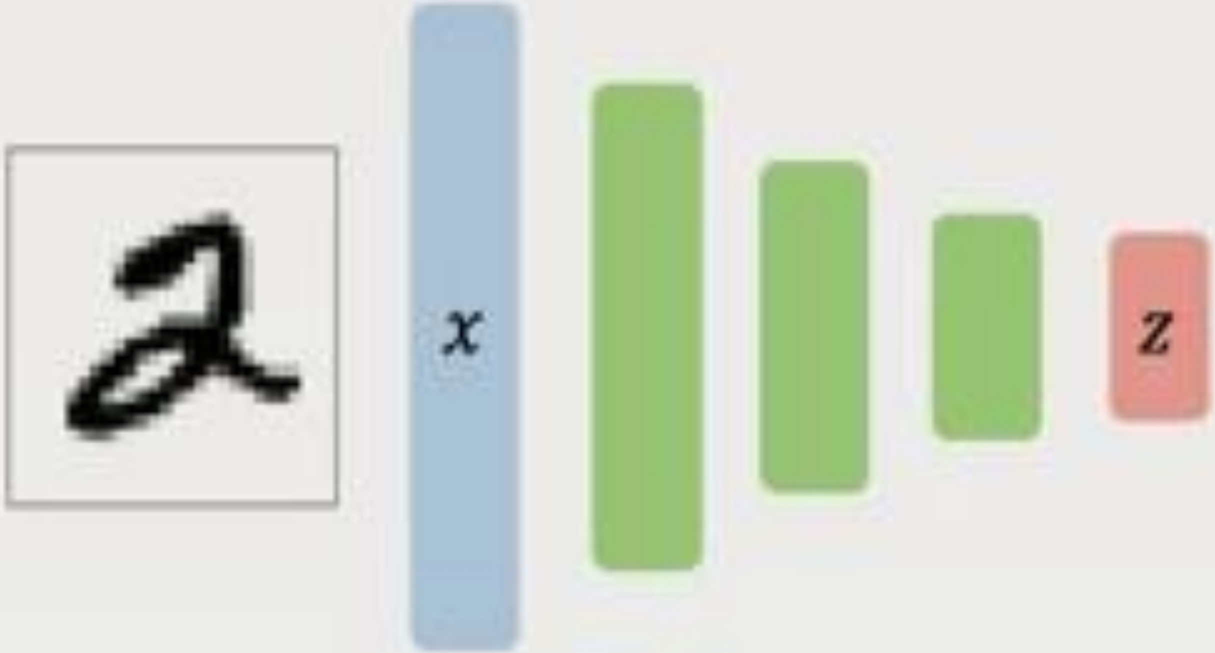
Autoencoders and Variational Autoencoders (VAEs)

Generative Adversarial Networks (GANs)

Autoencoders

Autoencoders: background

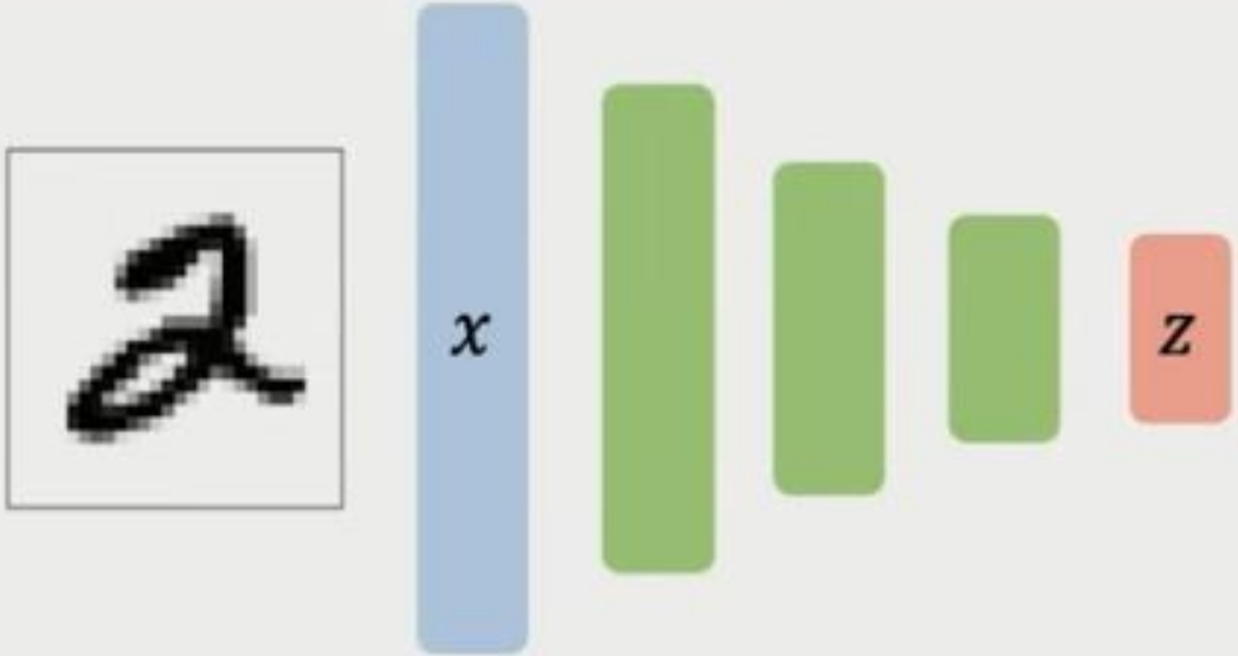
Unsupervised approach for learning a **lower-dimensional** feature representation from unlabeled training data



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

Autoencoders: background

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



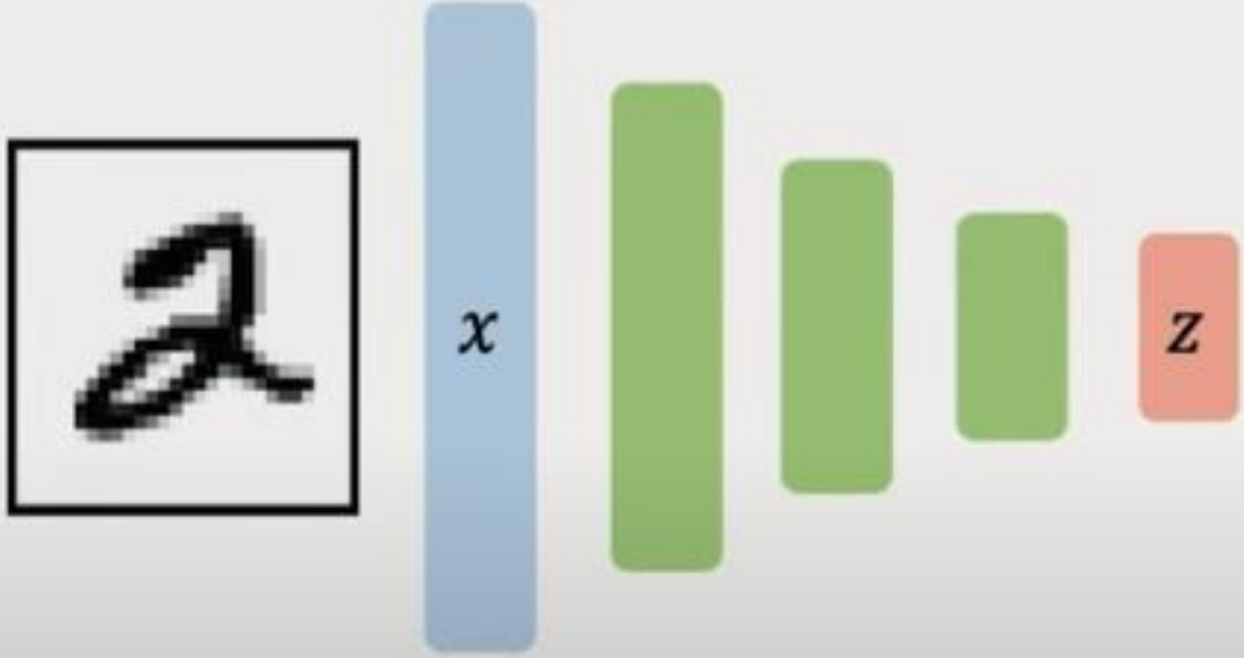
Why do we care about a low-dimensional z?

"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**

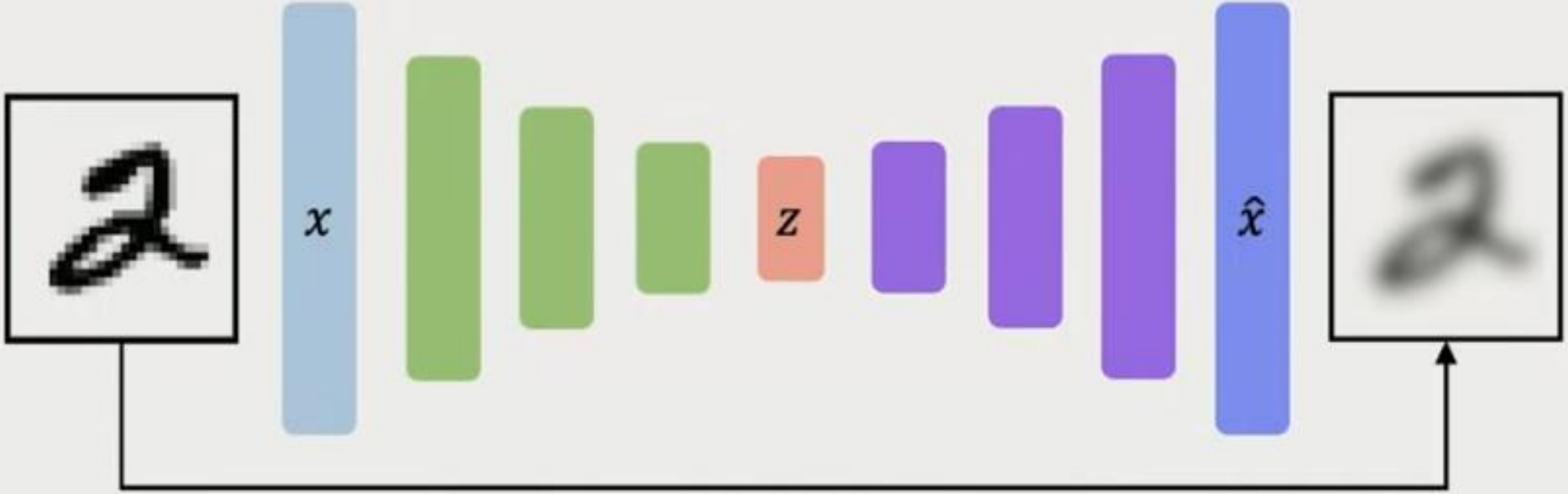


"Decoder" learns mapping back from latent space, z,
to a reconstructed observation, $\hat{x}$

Autoencoders: background

How can we learn this latent space?

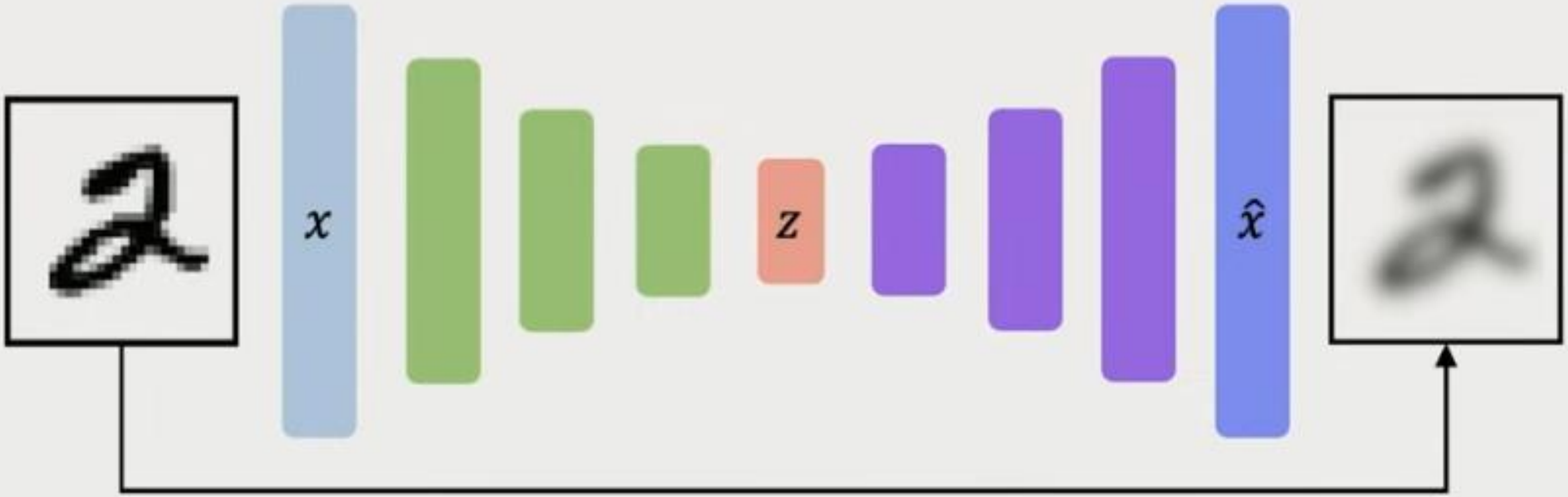
Train the model to use these features to **reconstruct the original data**



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

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!!

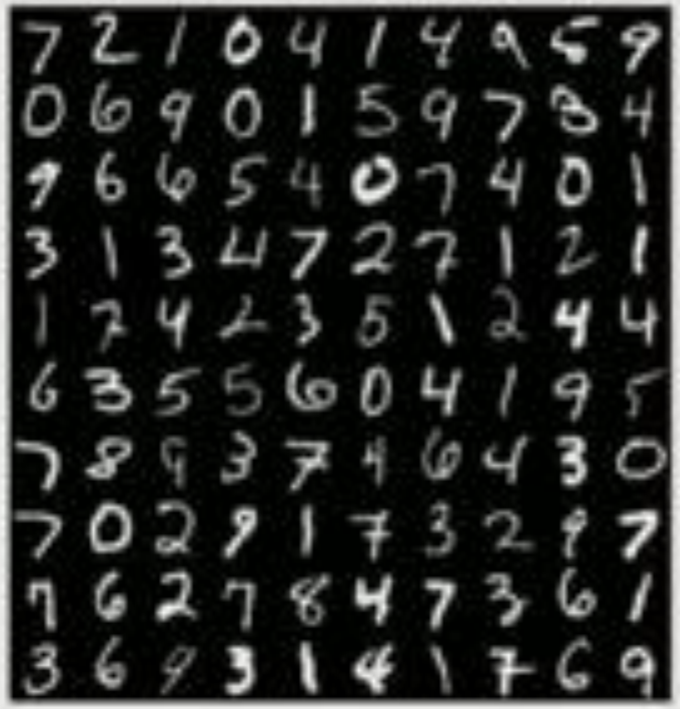
Dimensionality of latent space → reconstruction quality

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

2D latent space

5D latent space

Ground Truth



Autoencoders for representation learning

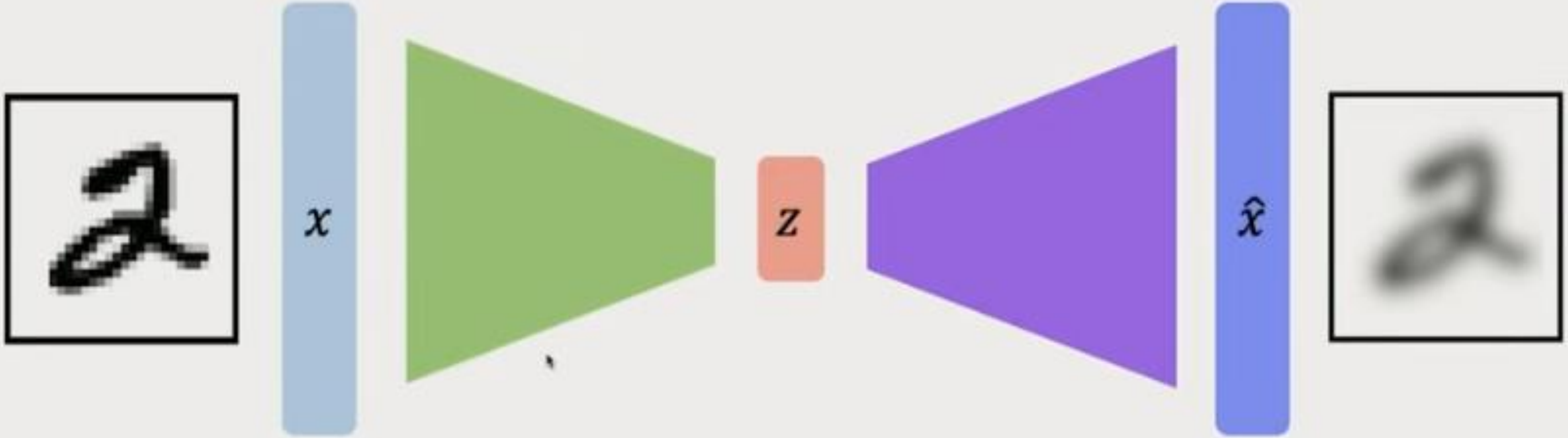
Bottleneck hidden layer forces network to learn a compressed latent representation

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

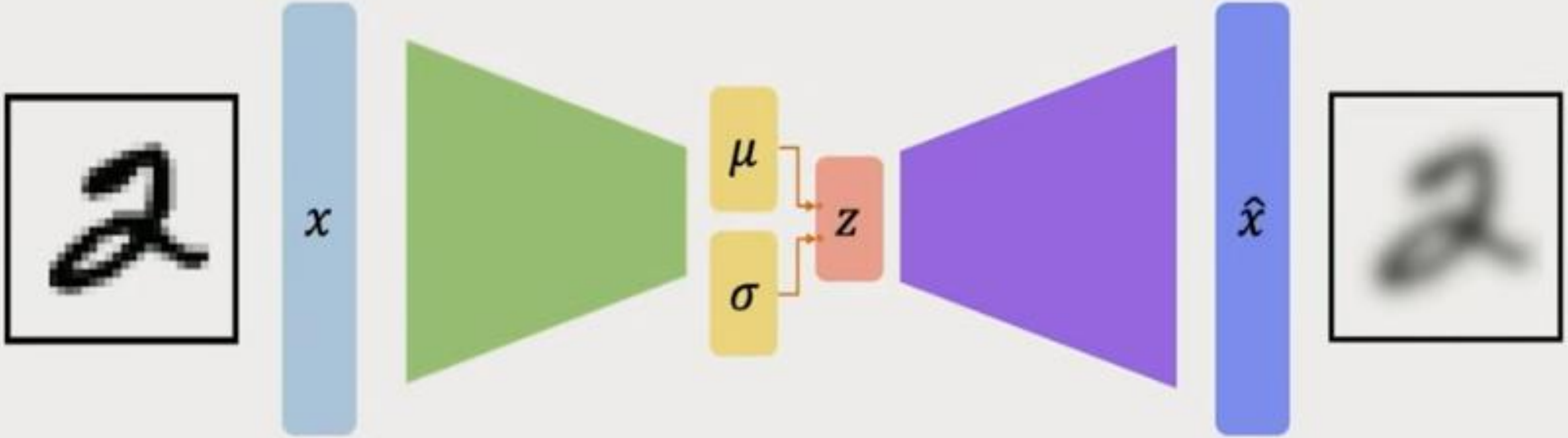
Autoencoding = **Auto**matically **encoding** data; "Auto" = **self**-encoding

Variational Autoencoders (VAEs)

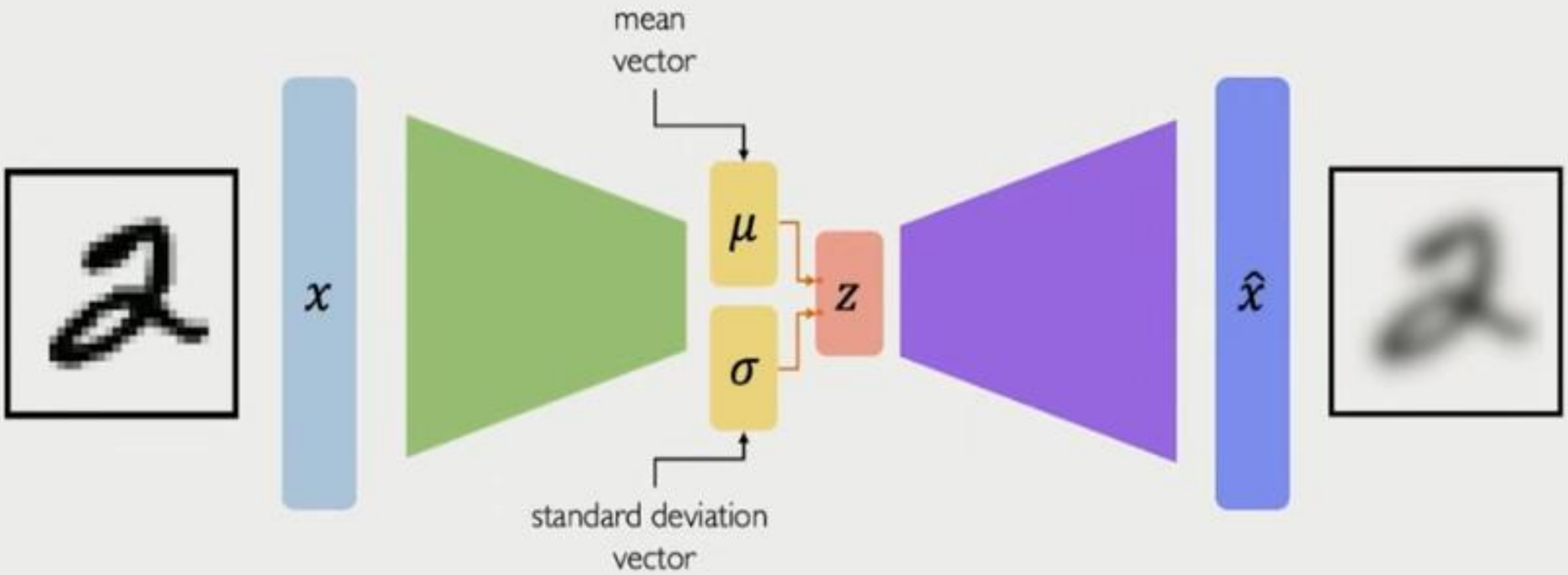
Traditional autoencoders



VAEs: key difference with traditional autoencoder



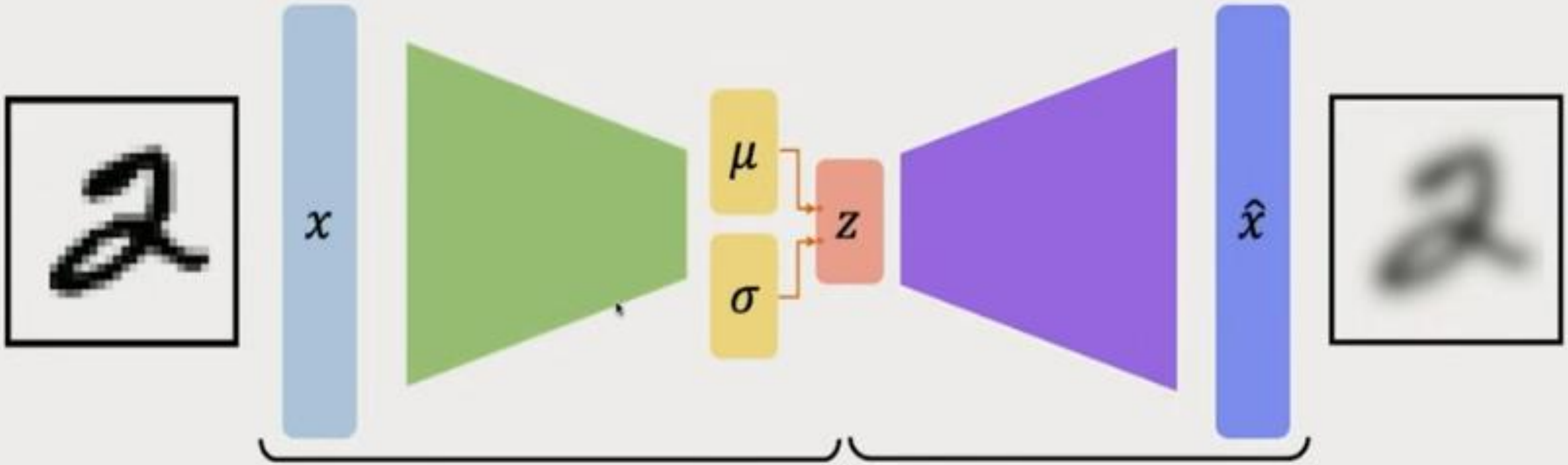
VAEs: key difference with traditional autoencoder



Variational autoencoders are a probabilistic twist on autoencoders!

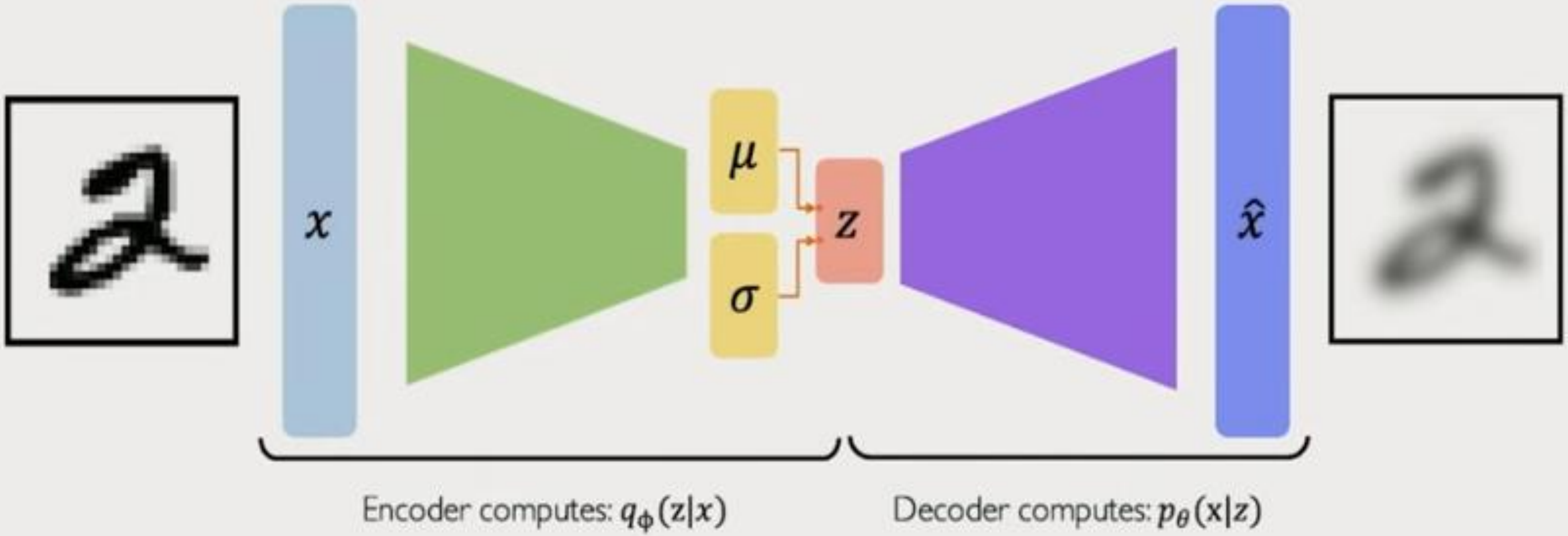
Sample from the mean and standard deviation to compute latent sample

VAE optimization



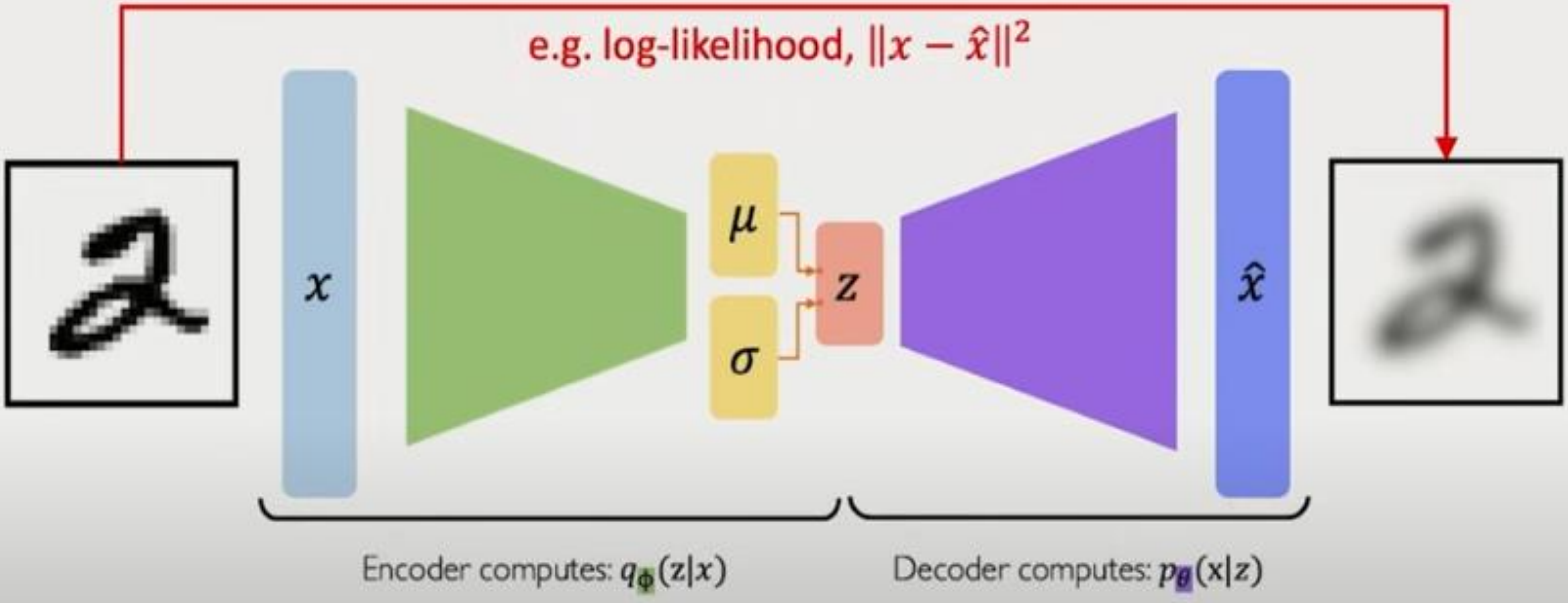
Encoder computes: $q_{\phi}(\mathrm{z}|x)$ Decoder computes: $p_{\theta}(\mathrm{x}|z)$

VAE optimization



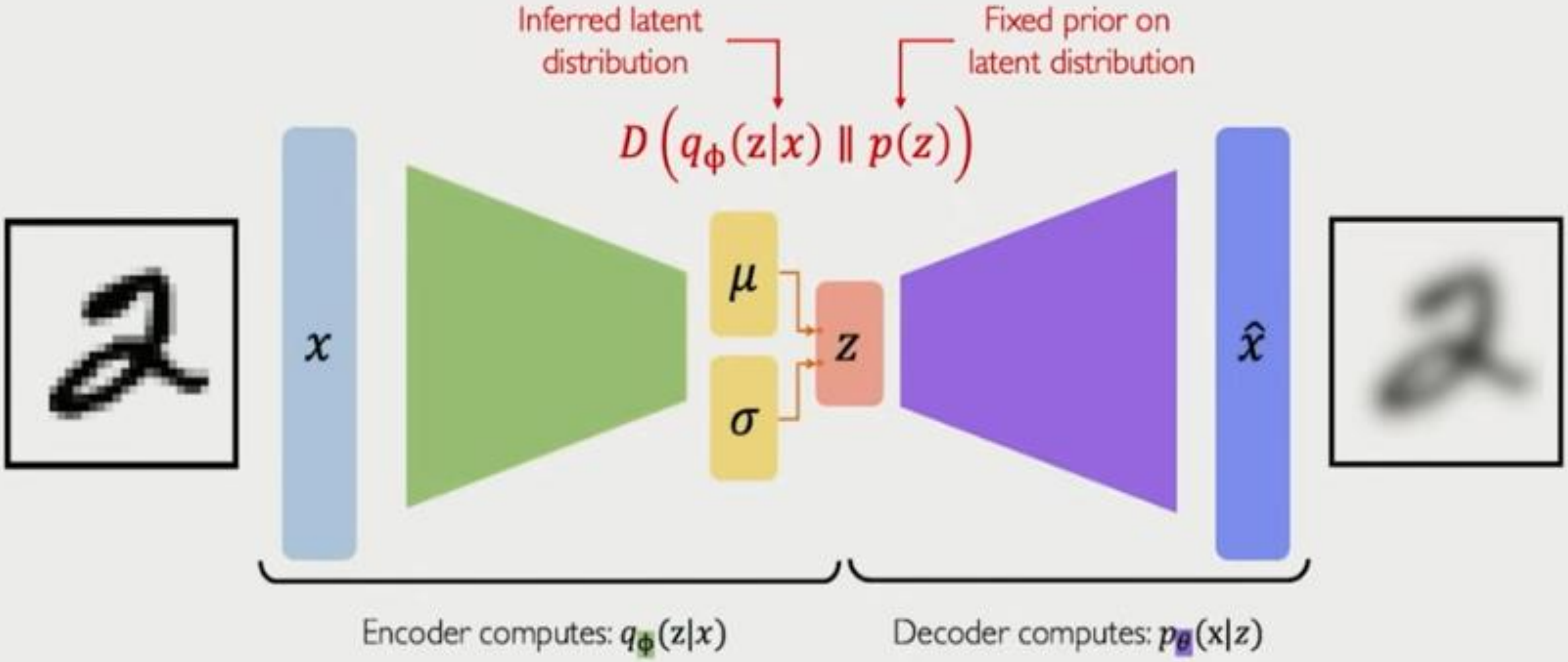
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

VAE optimization



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

VAE optimization



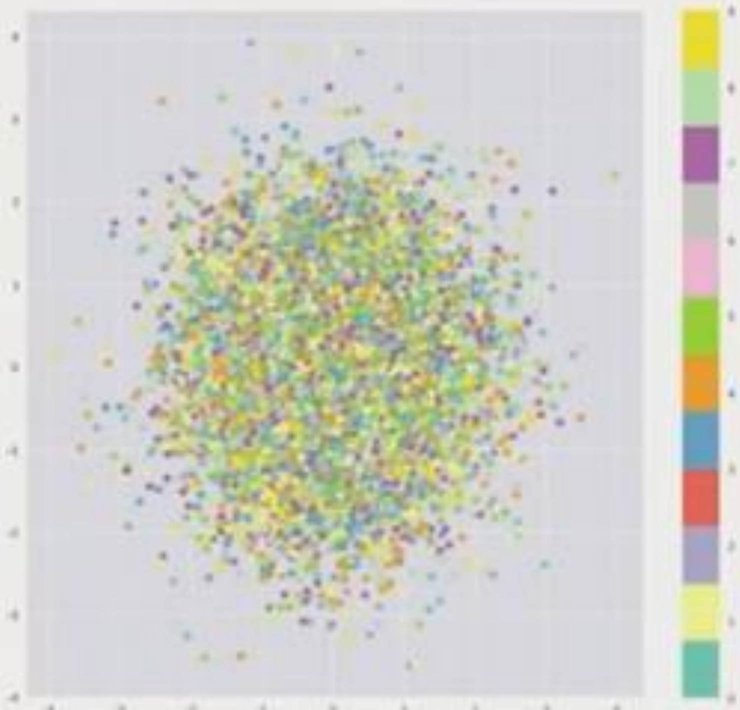
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + \boxed{(\text{regularization term})}$$

Priors on the latent distribution

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

Inferred latent distribution

Fixed prior on latent distribution



Common choice of prior – Normal Gaussian:

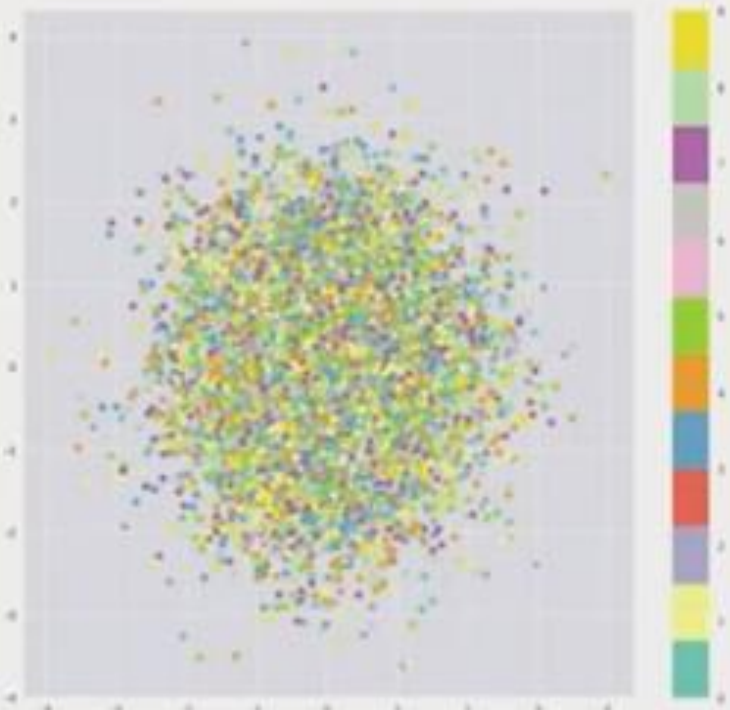
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- 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 (i.e., by memorizing the data)

Priors on the latent distribution

$$D\left(q_{\phi}(z|x) \parallel p(z)\right)$$
$$= -\frac{1}{2}\sum_{j=0}^{k-1}\left(\sigma_j + \mu_j^2 - 1 - \log\sigma_j\right)$$

KL-divergence between the two distributions



Common choice of prior – Normal Gaussian:

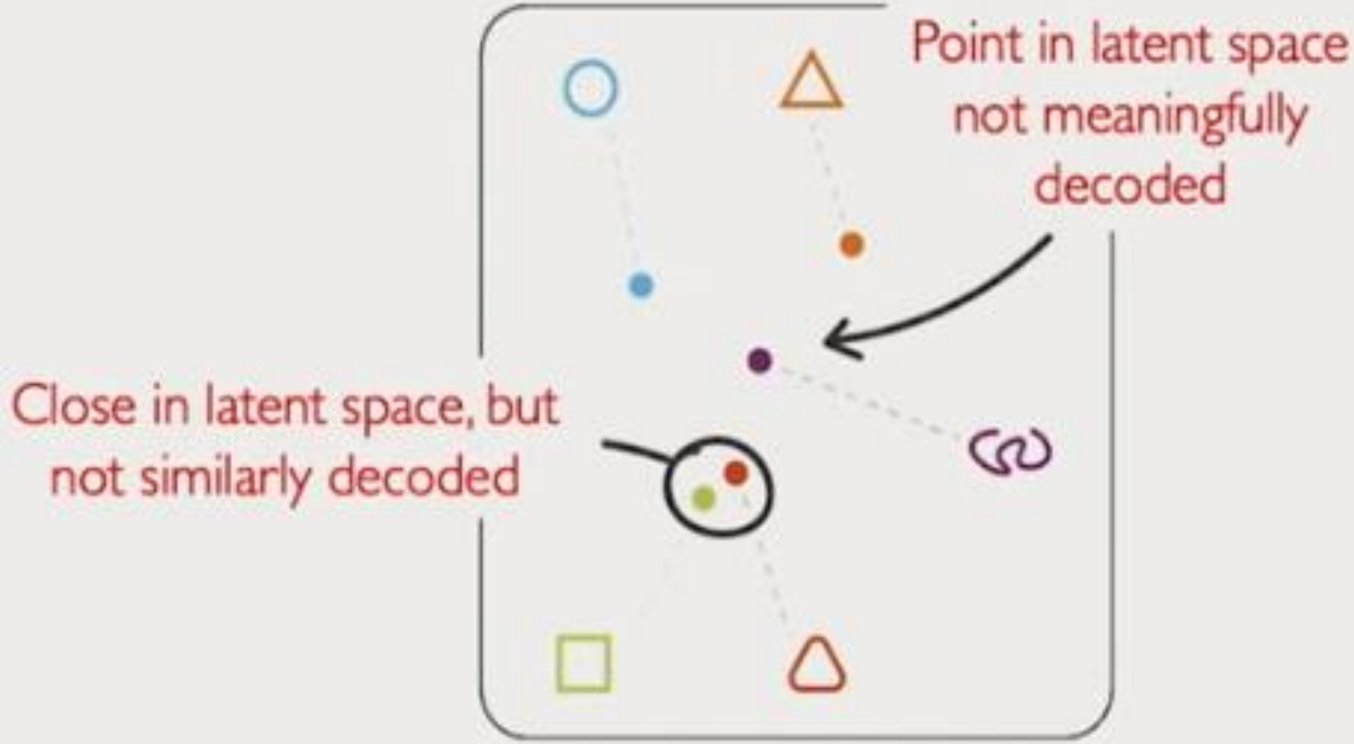
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- 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 (i.e., by memorizing the data)

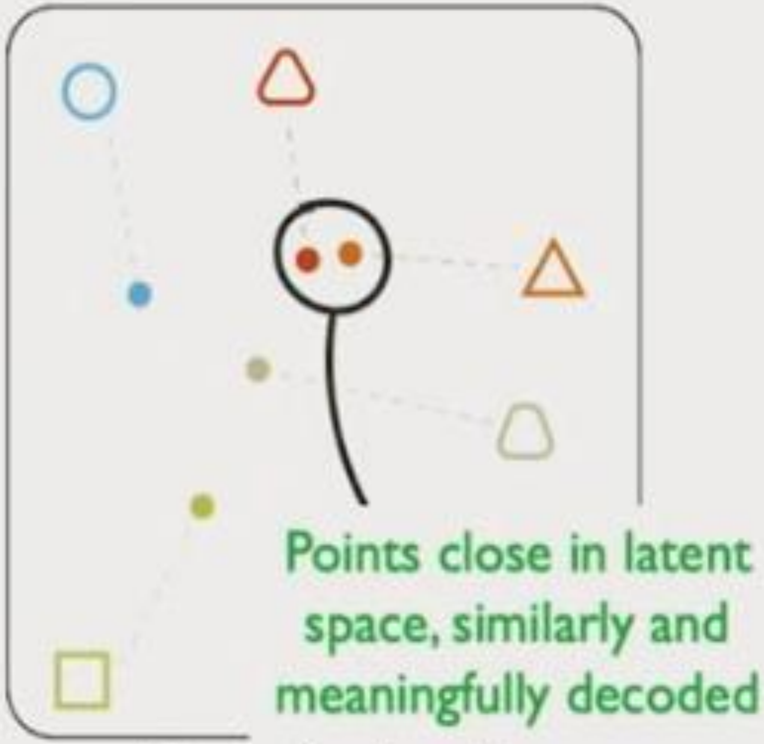
Intuition on regularization and the Normal prior

What properties do we want to achieve from regularization? 🤔

1. **Continuity:** points that are close in latent space → similar content after decoding
2. **Completeness:** sampling from latent space → "meaningful" content after decoding

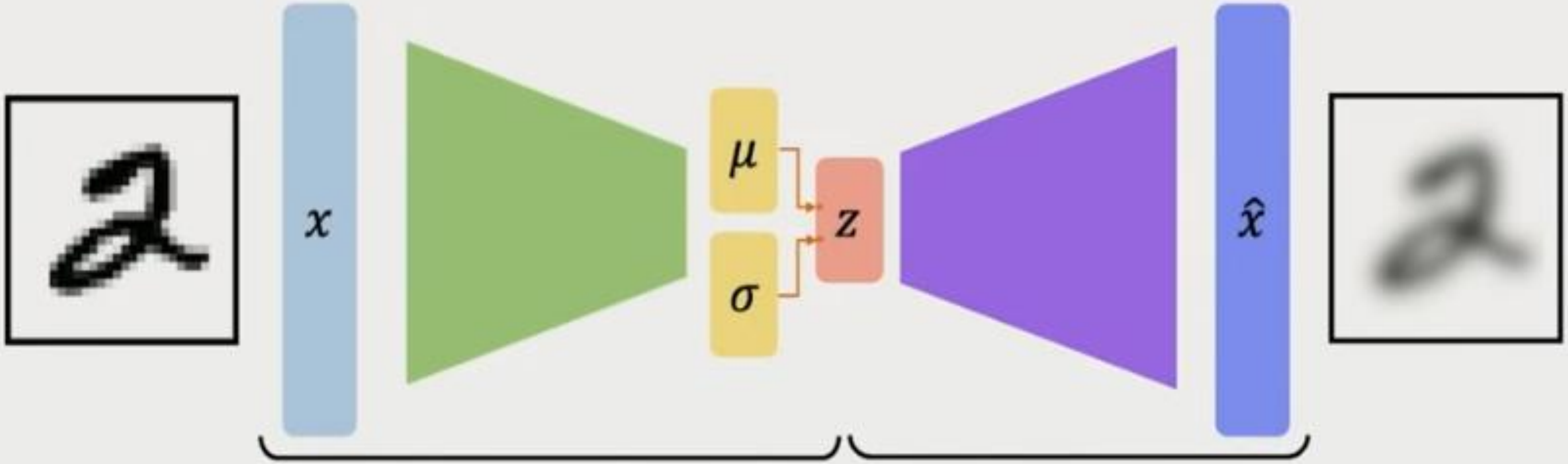


Not regularized



Regularized

VAE computation graph

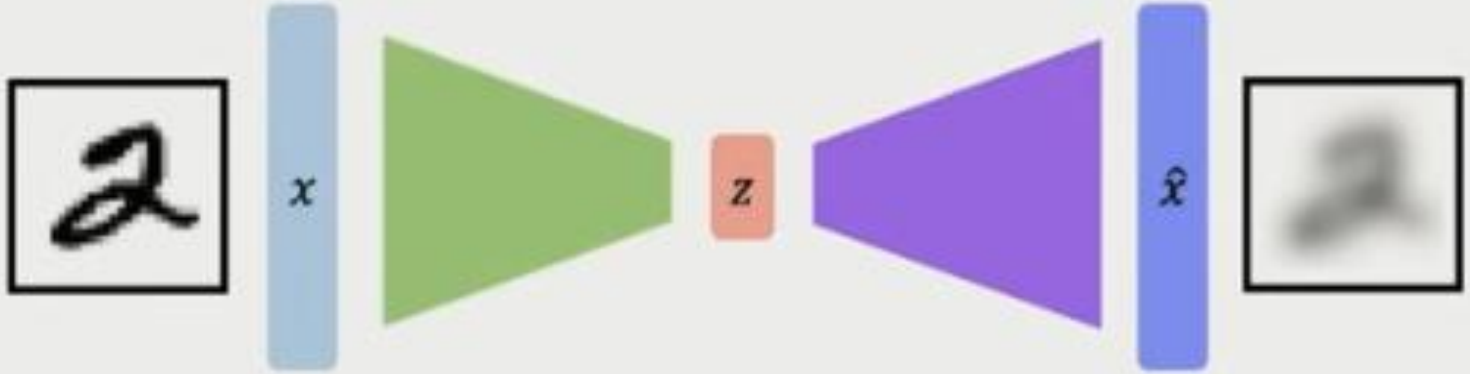


Encoder computes: $q_\phi(z|x)$ Decoder computes: $p_\theta(x|z)$

Sample from latent space and decode to generate new samples

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



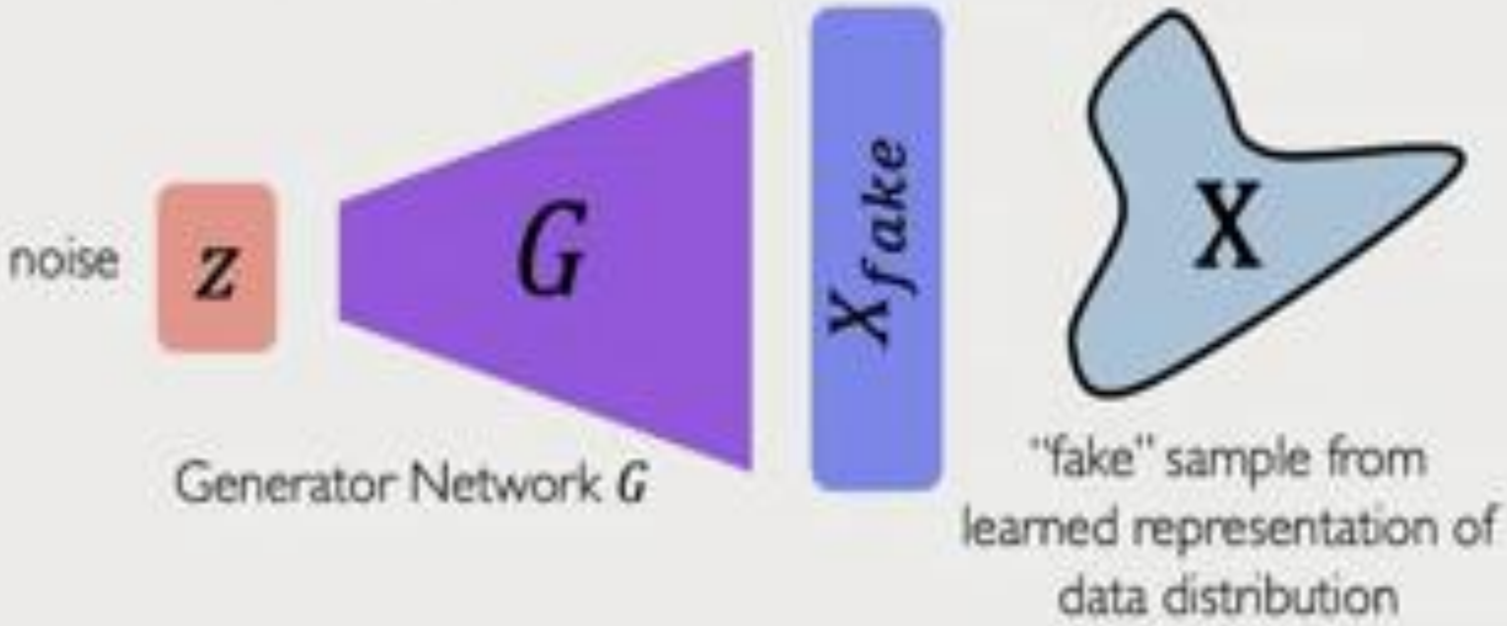
Generative Adversarial Networks (GANs)

What if we just want to sample?

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

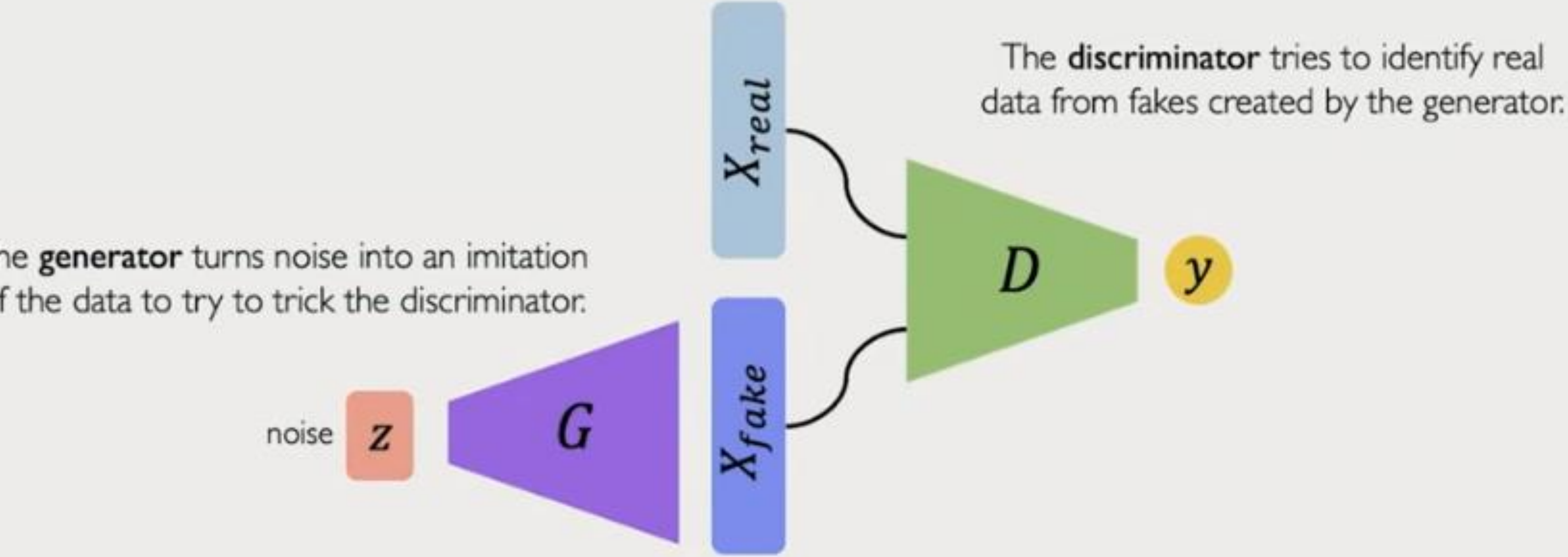
Problem: want to sample from complex distribution – can't do this directly!

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



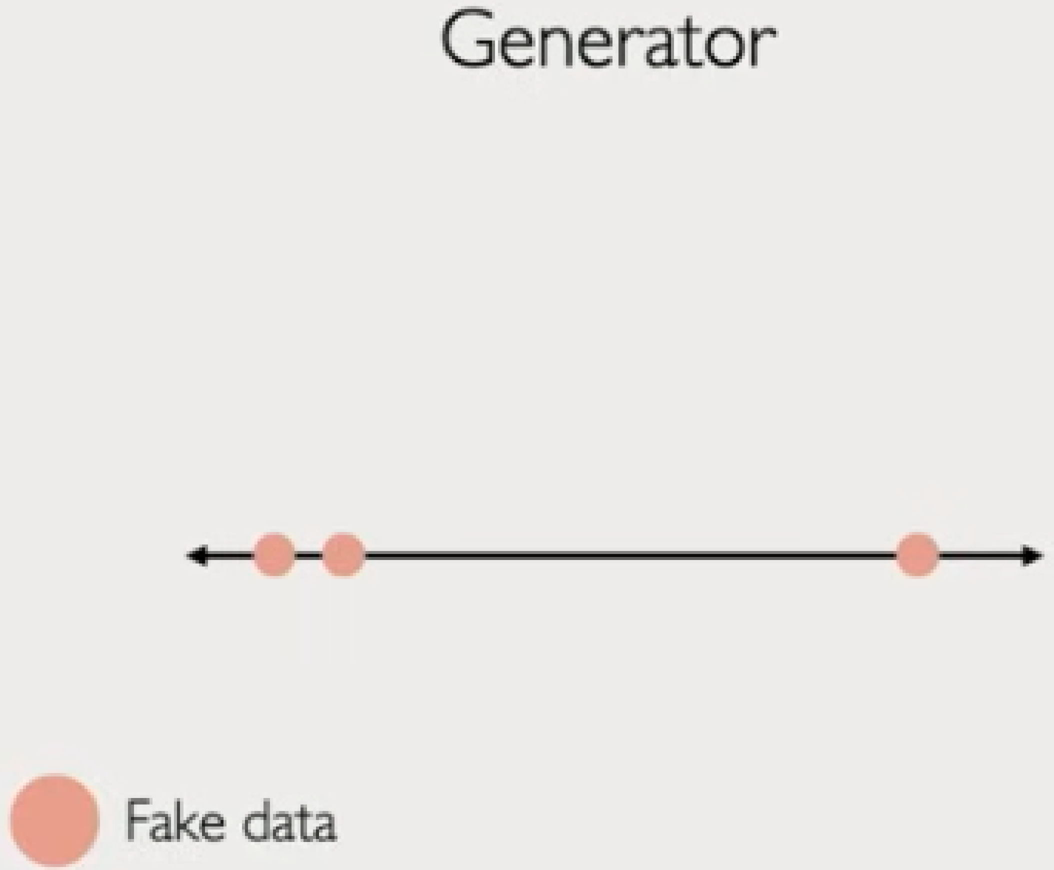
Generative Adversarial Networks (GANs)

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.

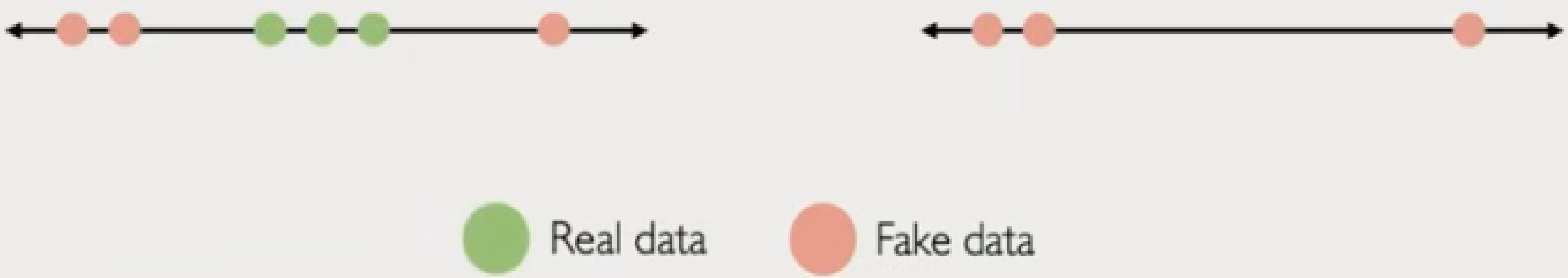


Intuition behind GANs

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

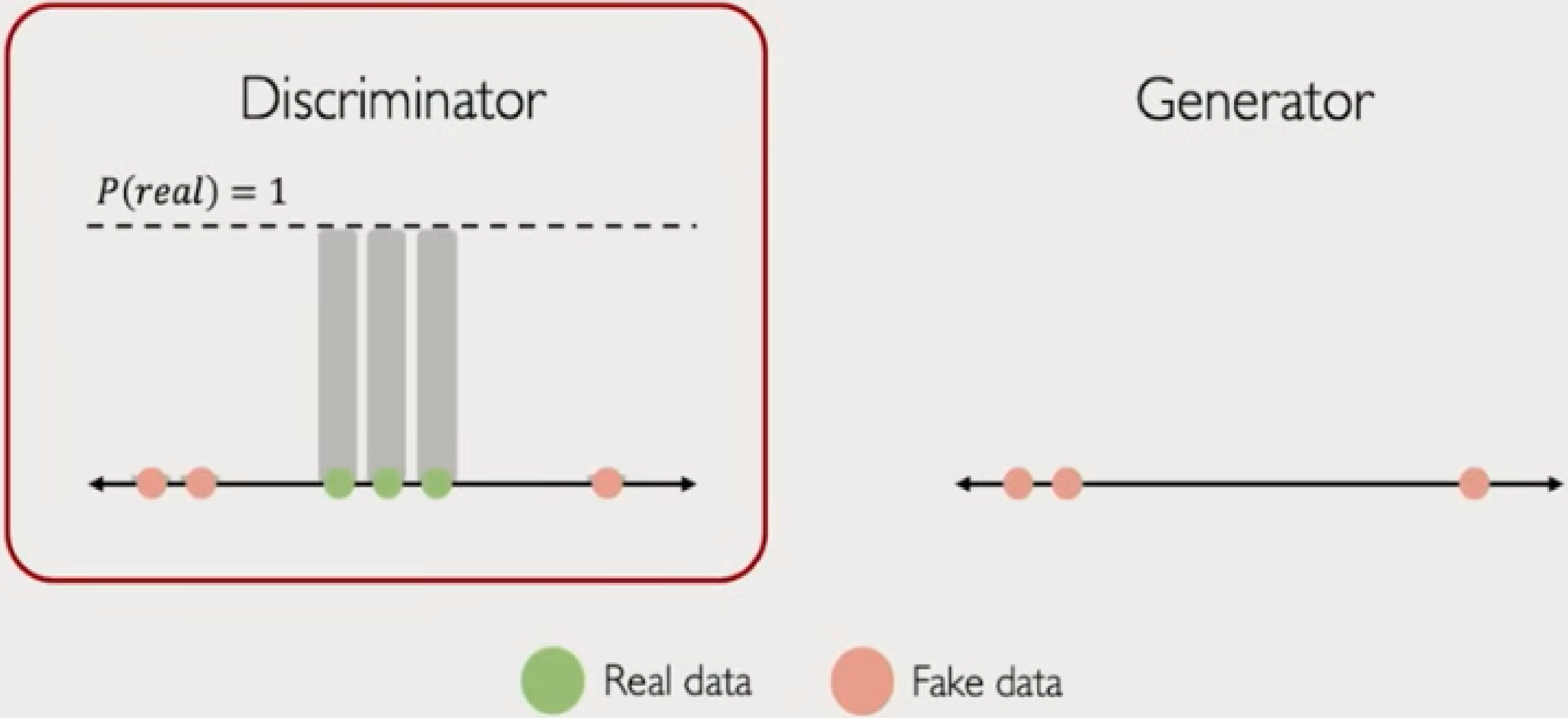
Discriminator

Generator



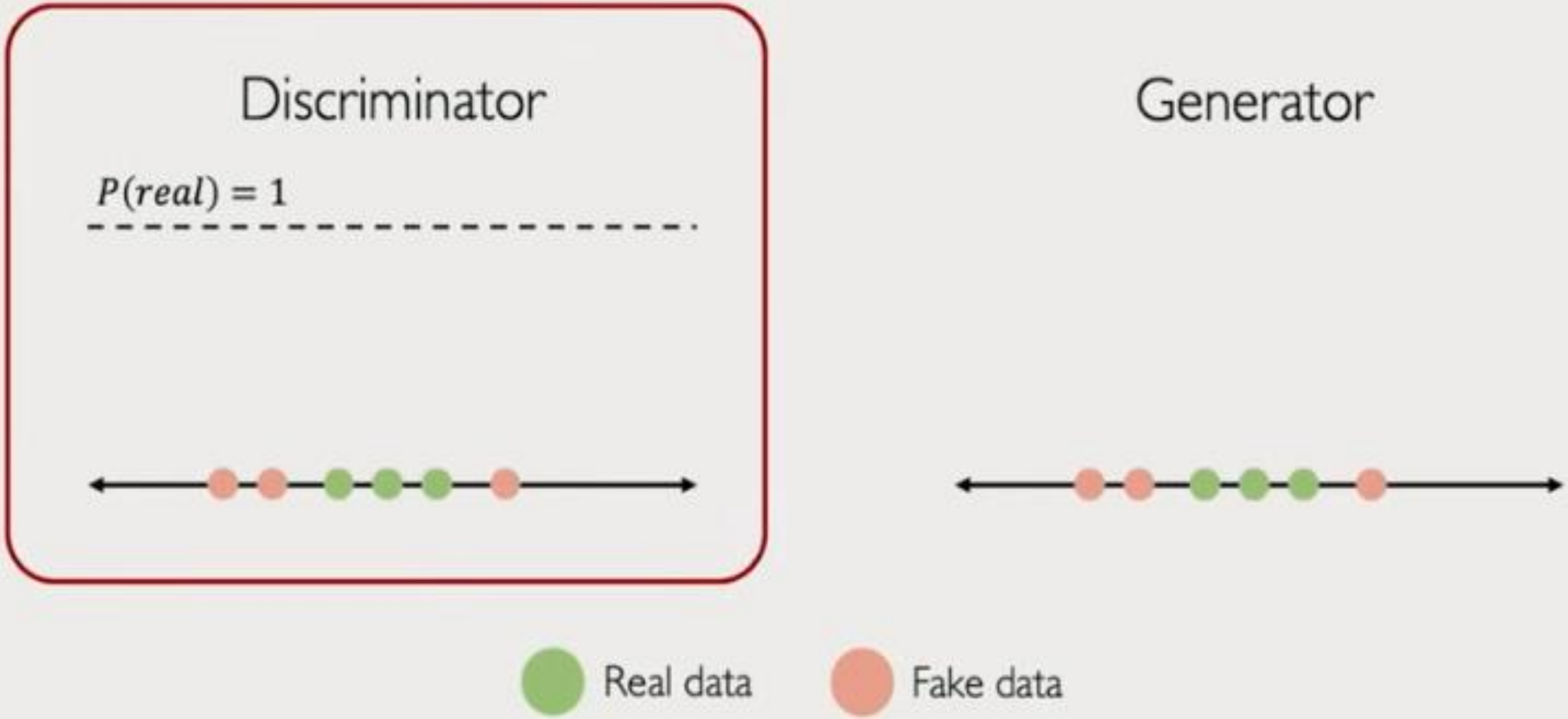
Intuition behind GANs

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



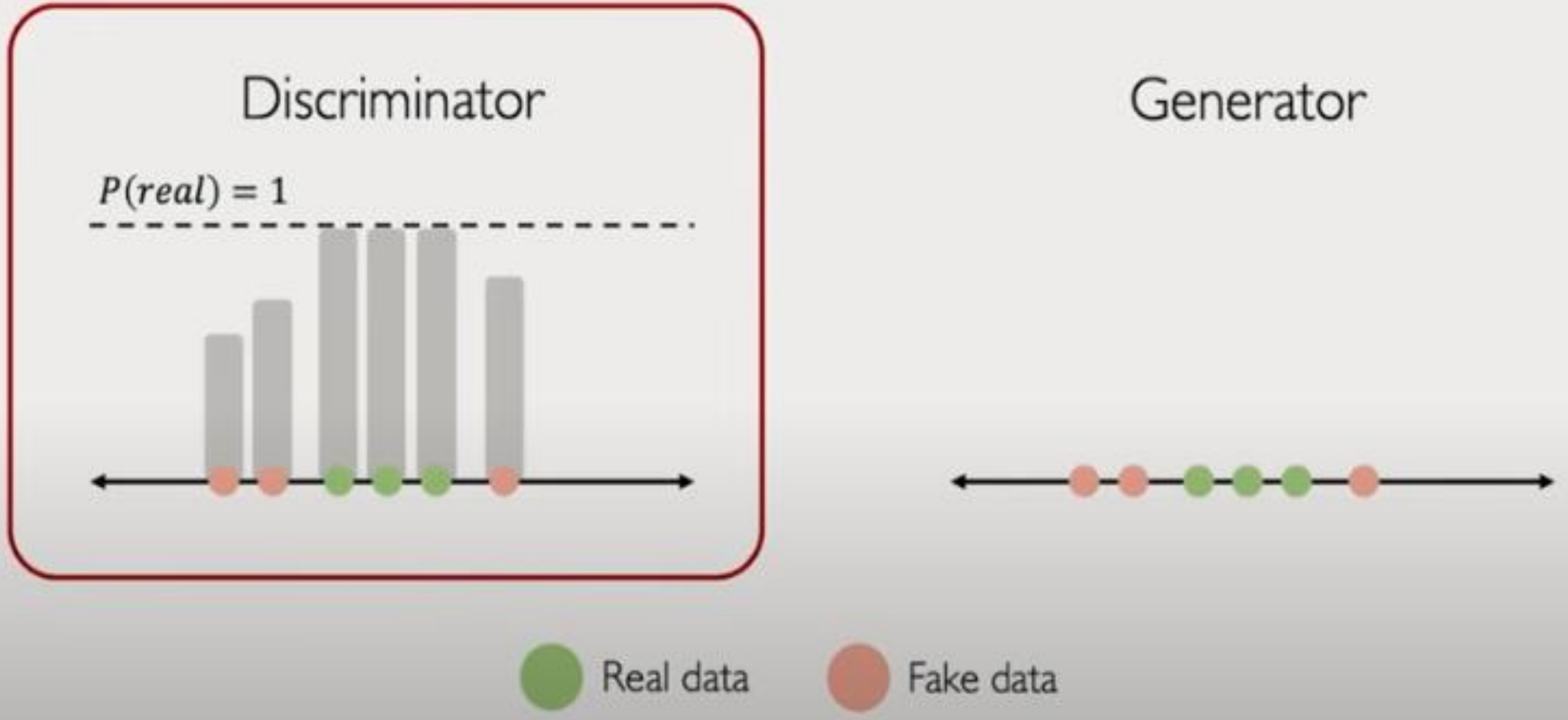
Intuition behind GANs

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



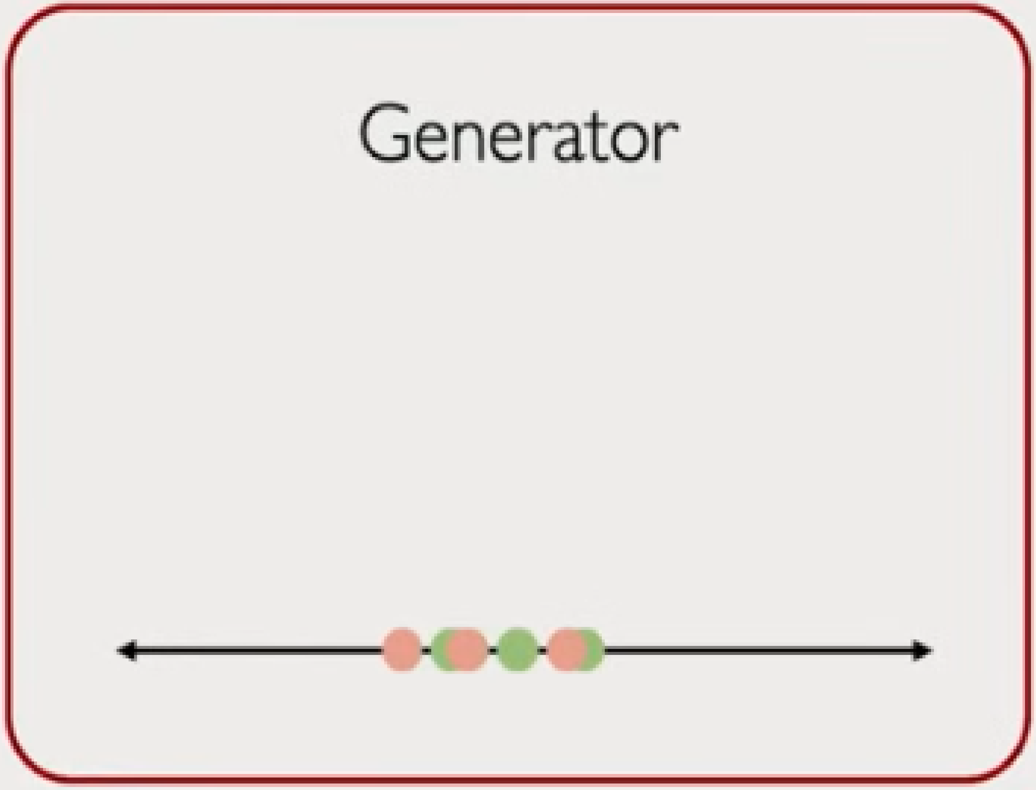
Intuition behind GANs

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



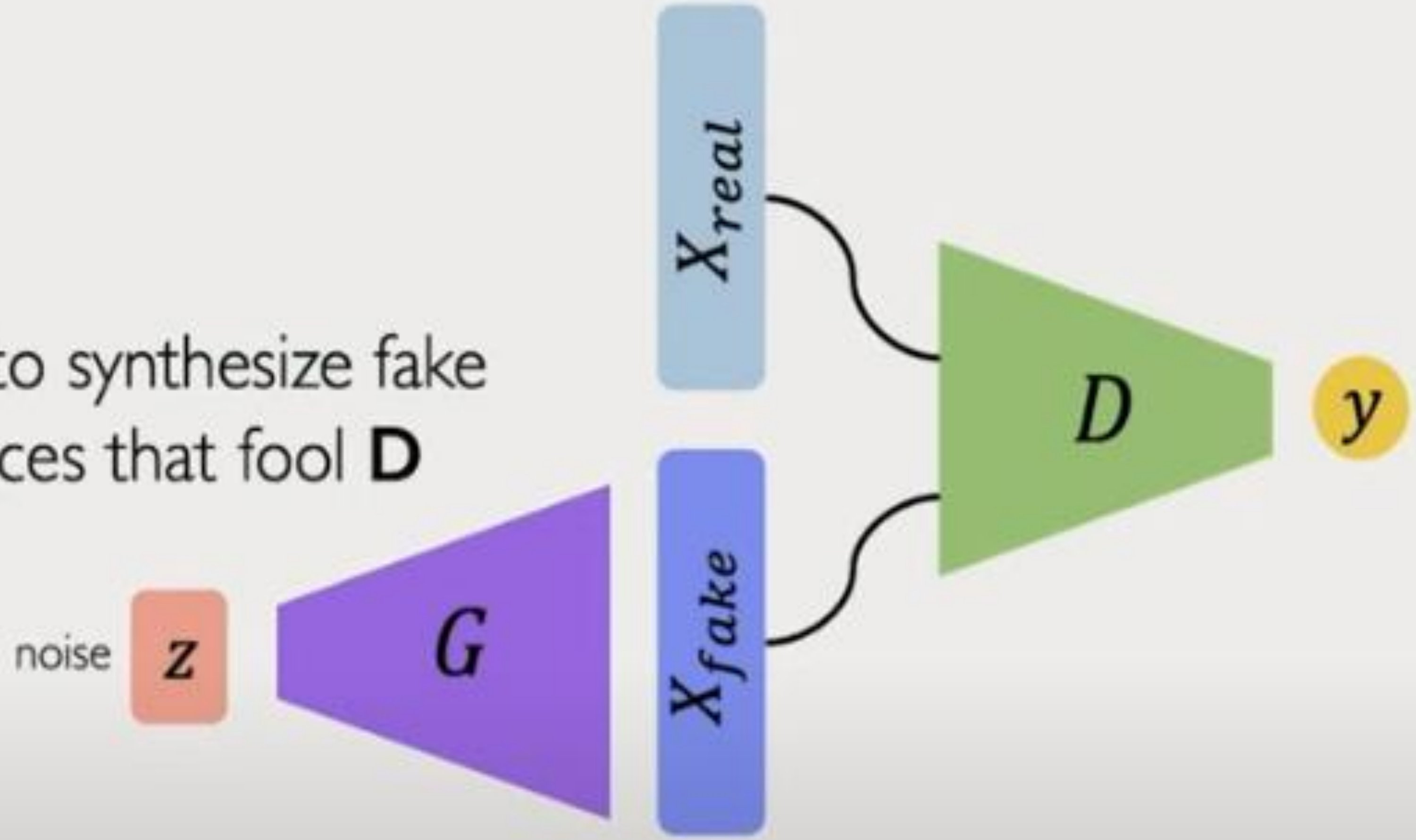
Intuition behind GANs

Generator tries to improve its imitation of the data.



Training GANs

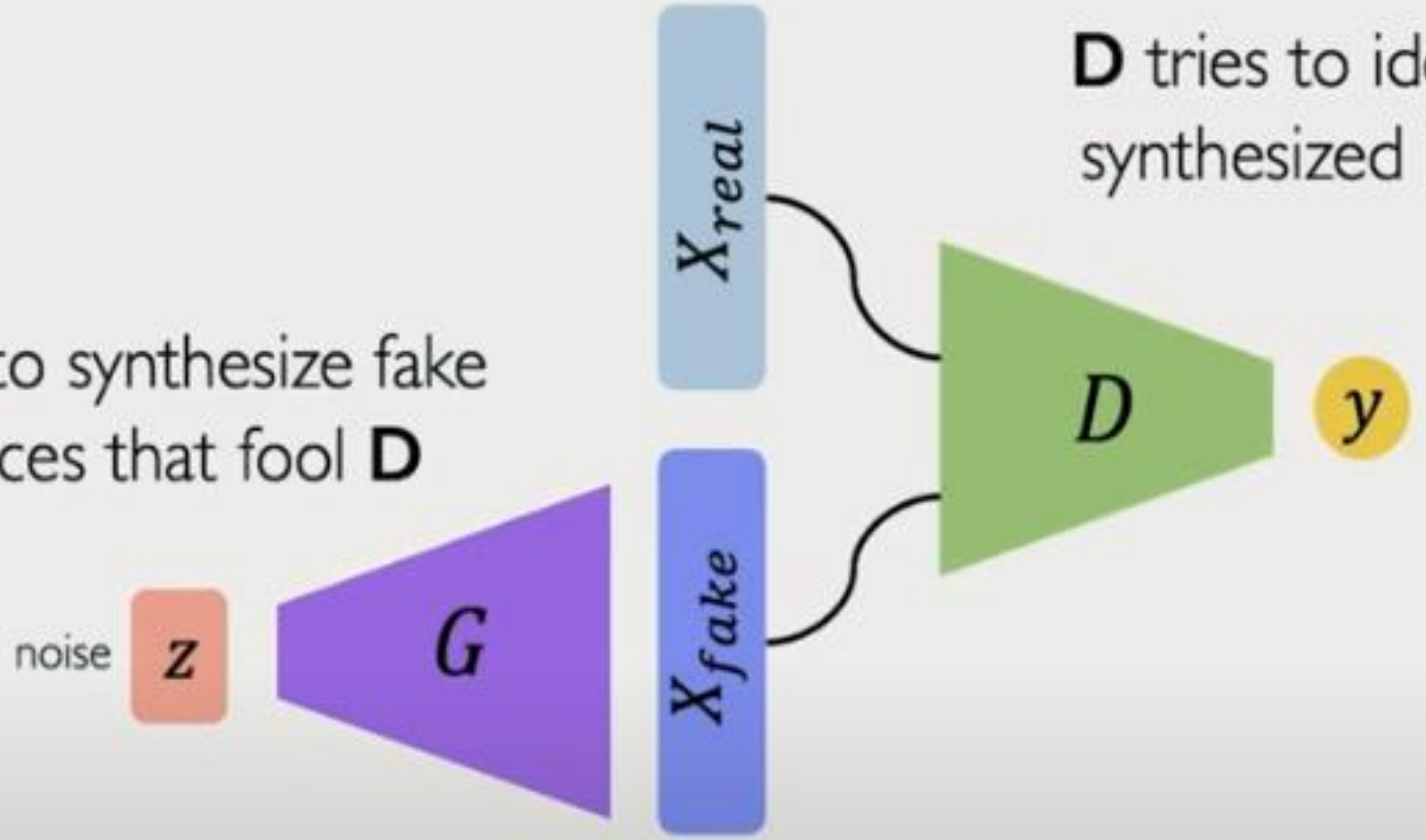
G tries to synthesize fake instances that fool **D**



Training GANs

D tries to identify the synthesized instances

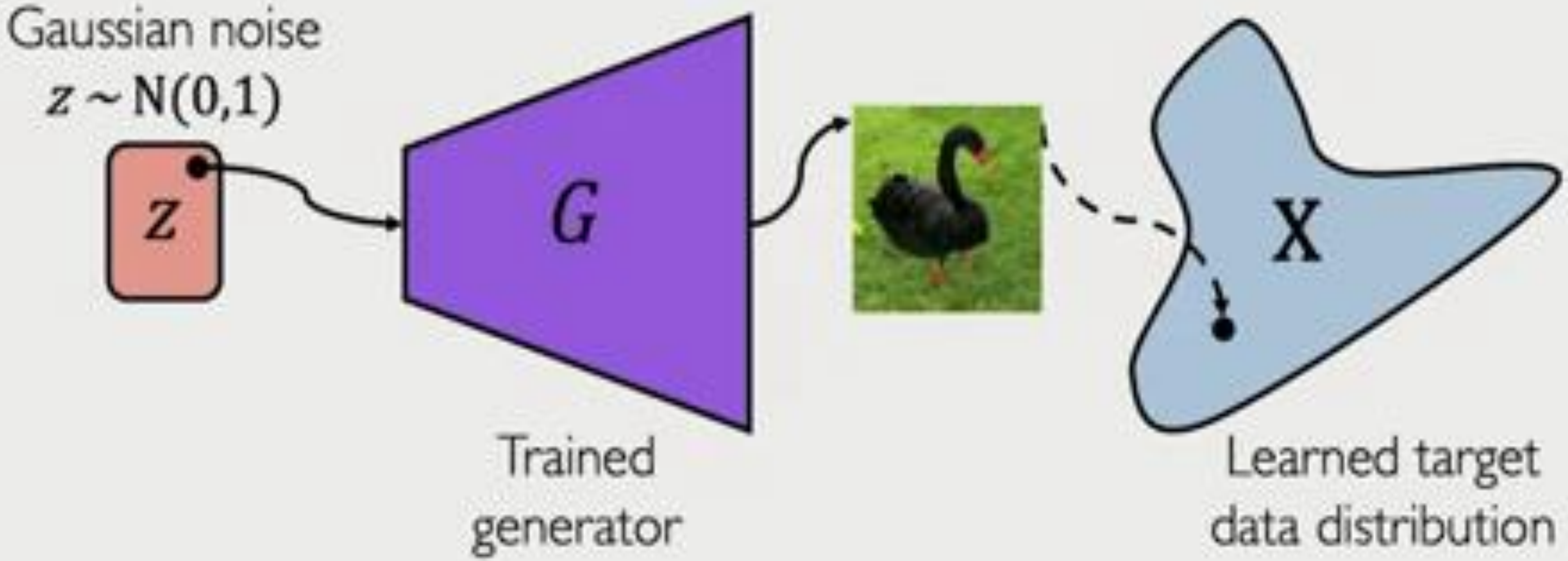
G tries to synthesize fake instances that fool **D**



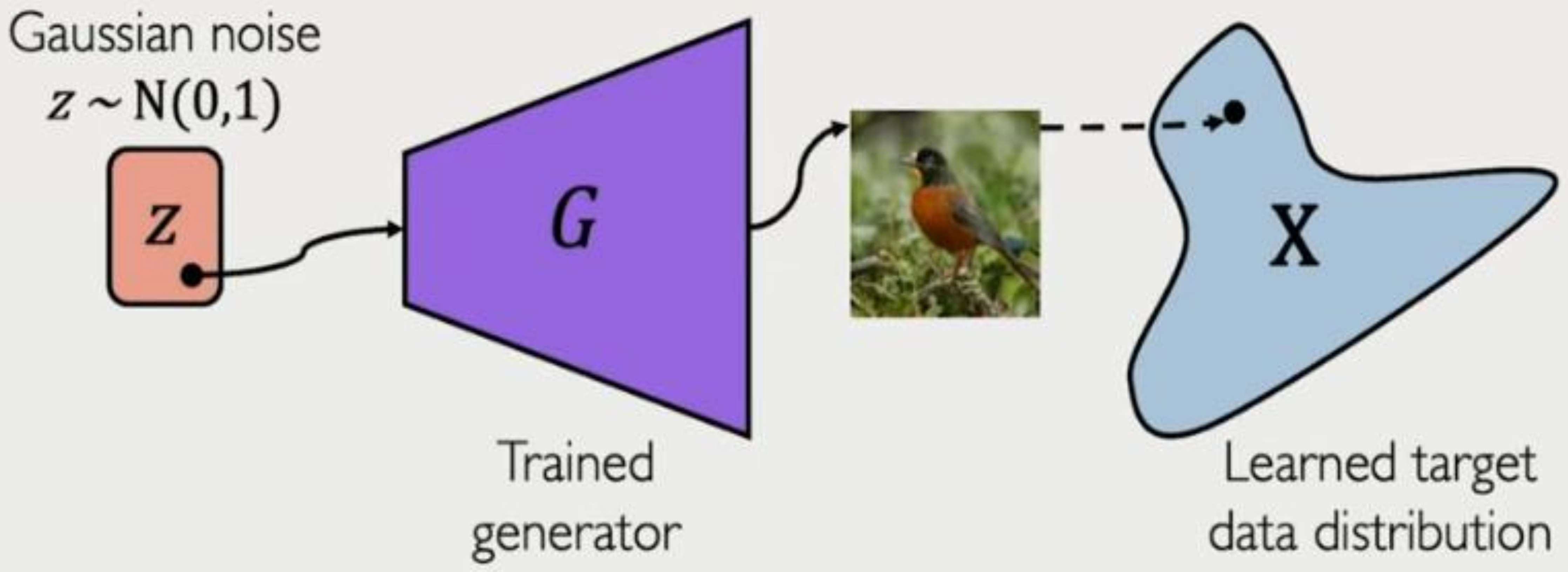
Training: adversarial objectives for **D** and **G**

Global optimum: G reproduces the true data distribution

GANs are distribution transformers



GANs are distribution transformers



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