

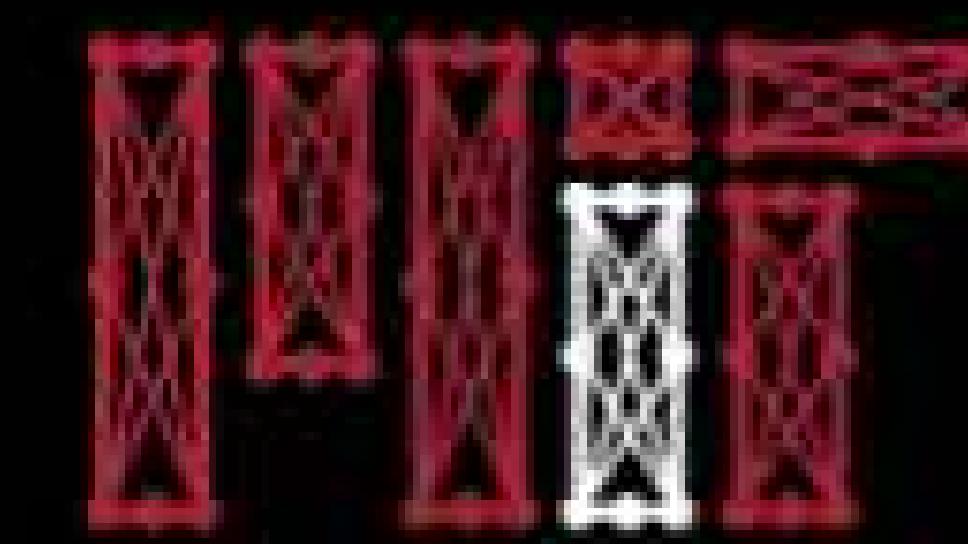


Deep Generative Modeling

Ava Soleimany

MIT 6.S191

January 25, 2022



Which face is real?



A



B



C

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.

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!

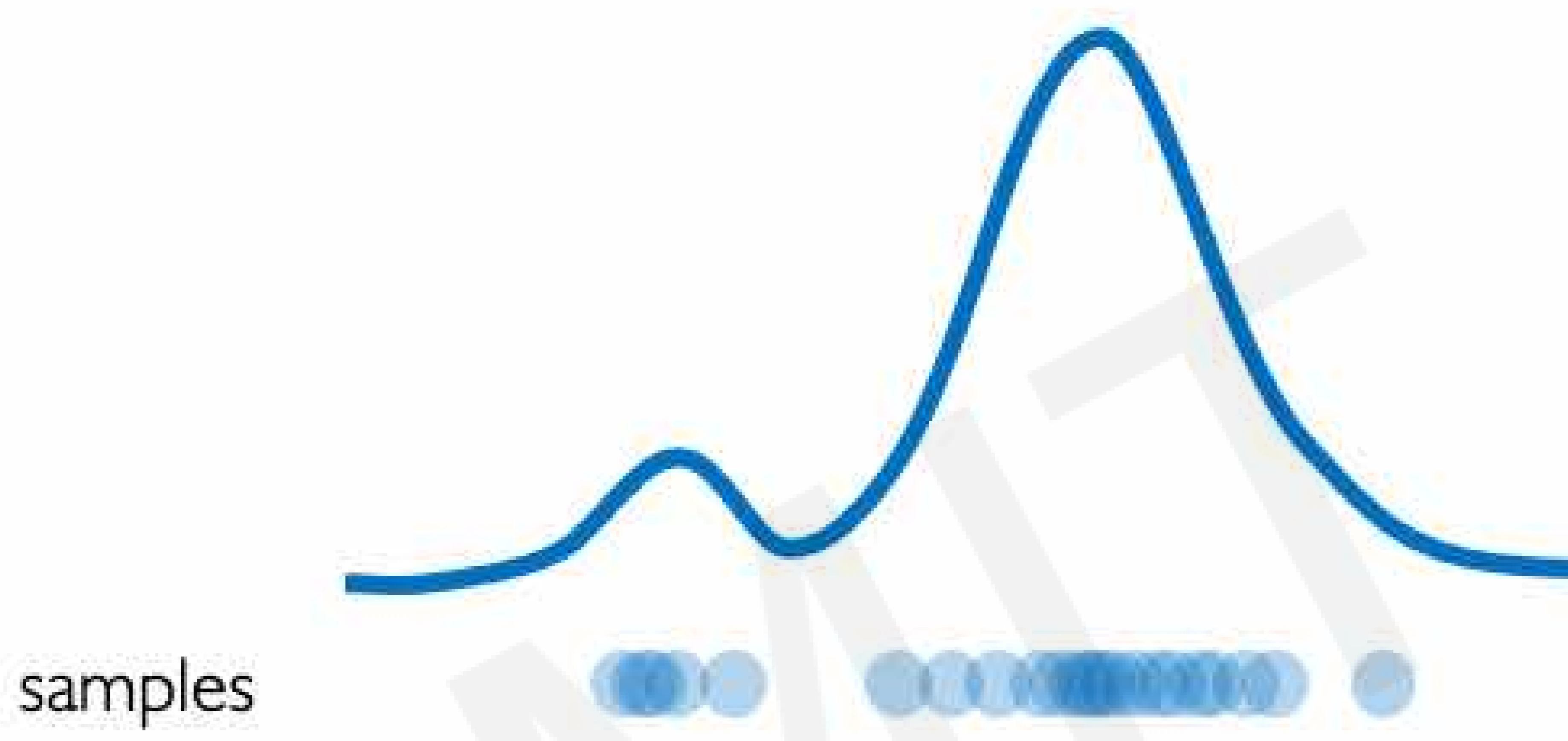
Goal: Learn the hidden or
underlying structure of the data

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

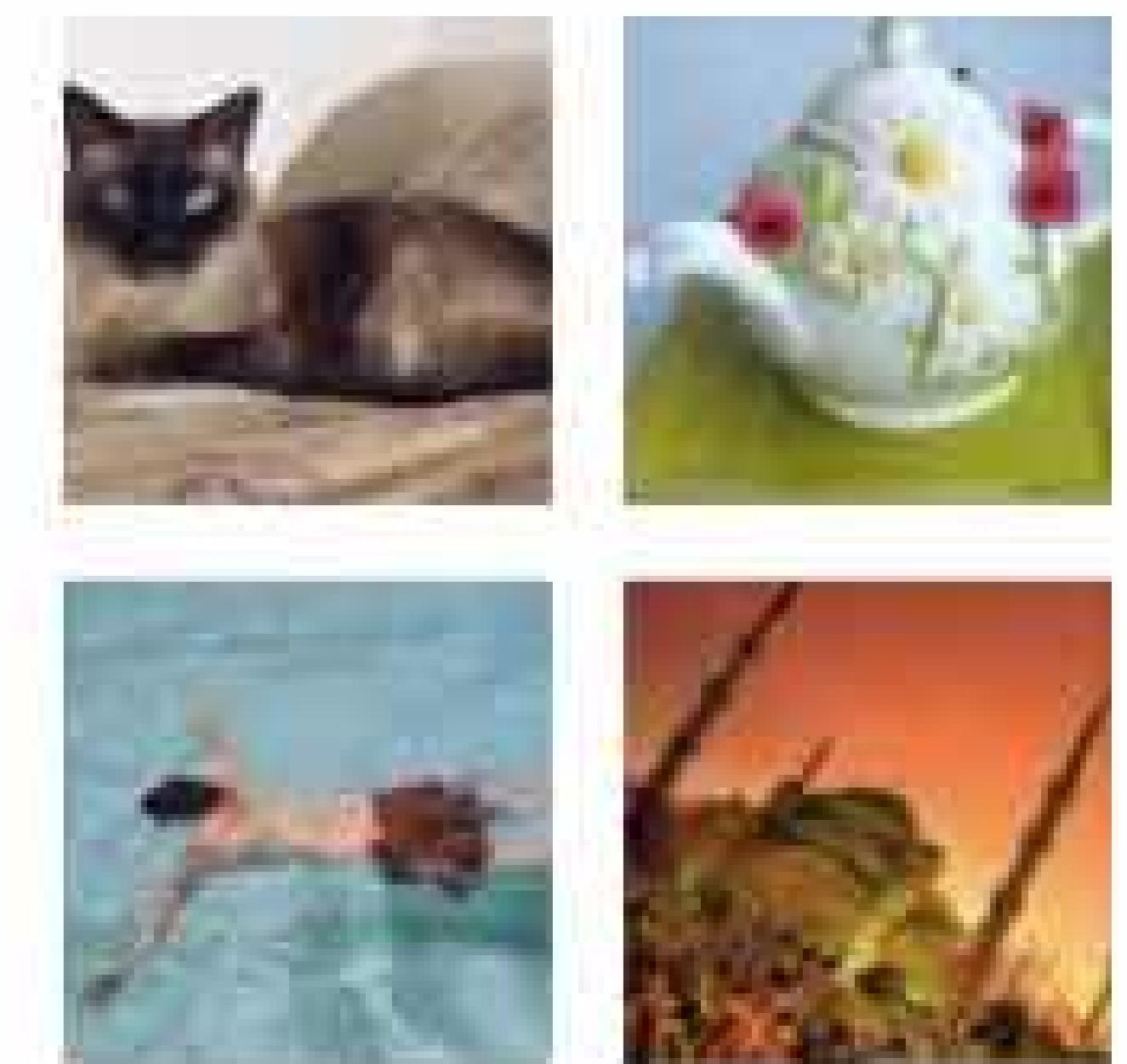
Density Estimation



Input samples

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

Sample Generation



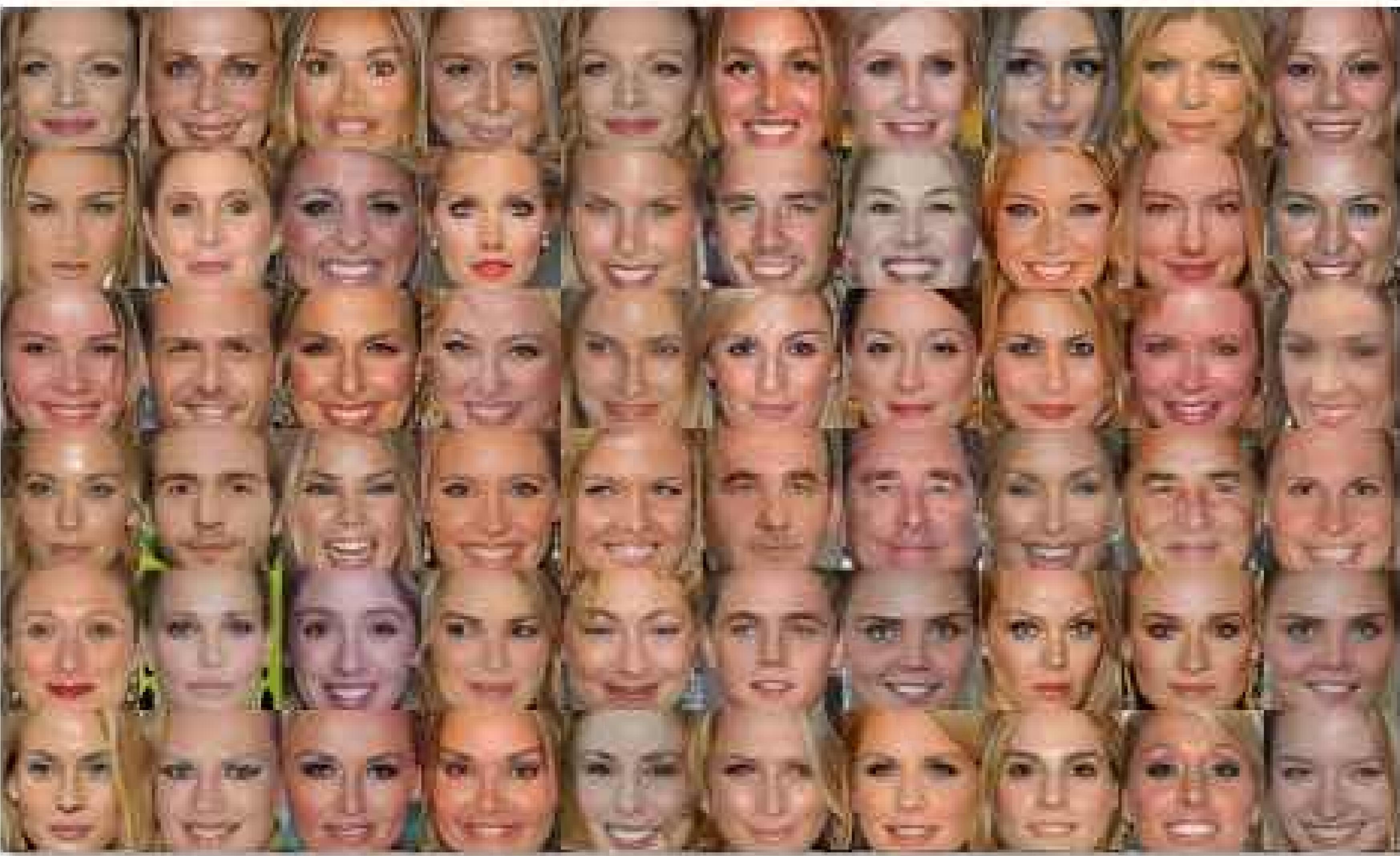
Generated samples

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

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

Why generative models? Debiasing

Capable of uncovering **underlying features** in a dataset



Homogeneous skin color, pose



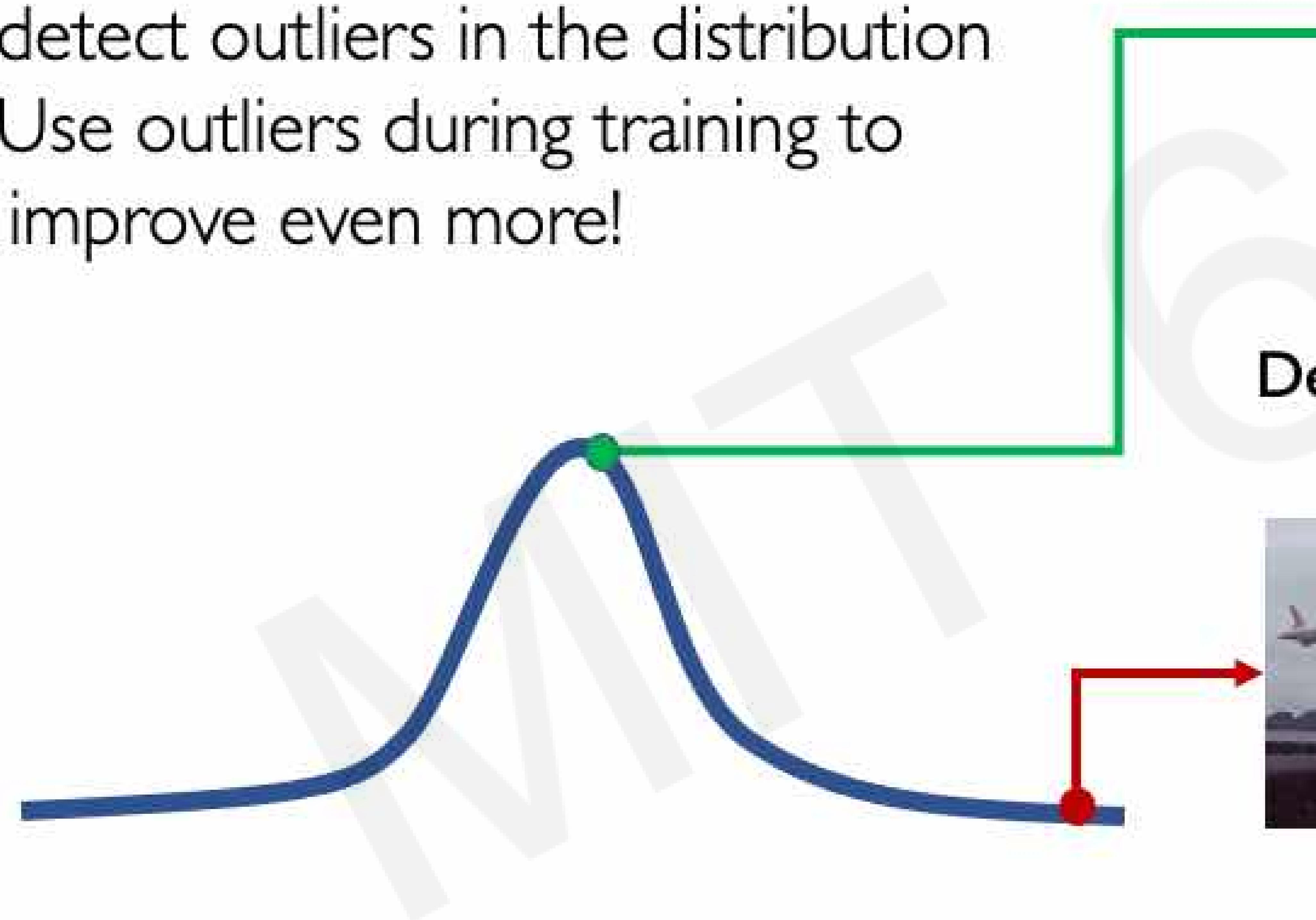
Diverse skin color, pose, illumination

How can we use this information 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



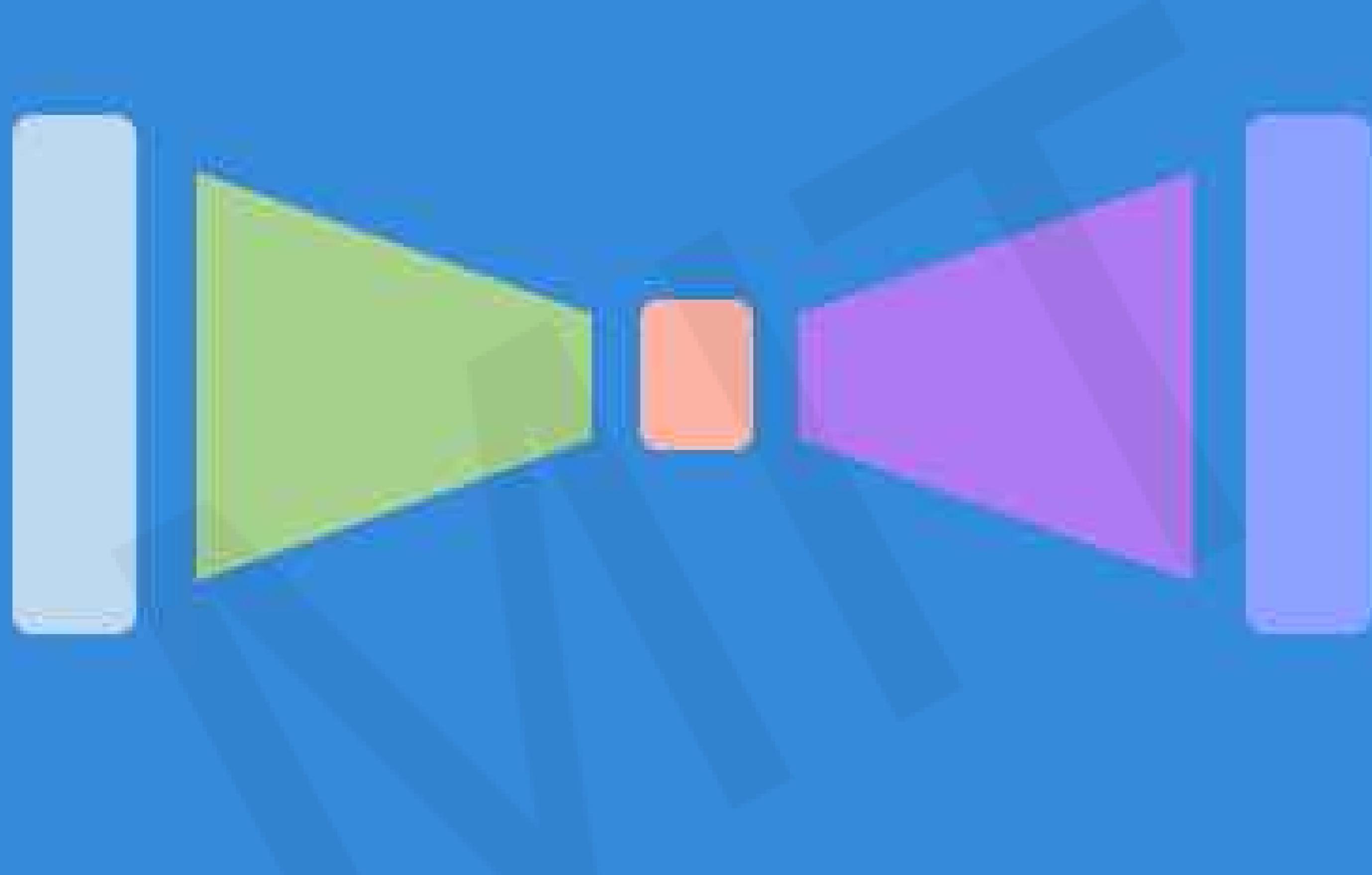
Harsh Weather



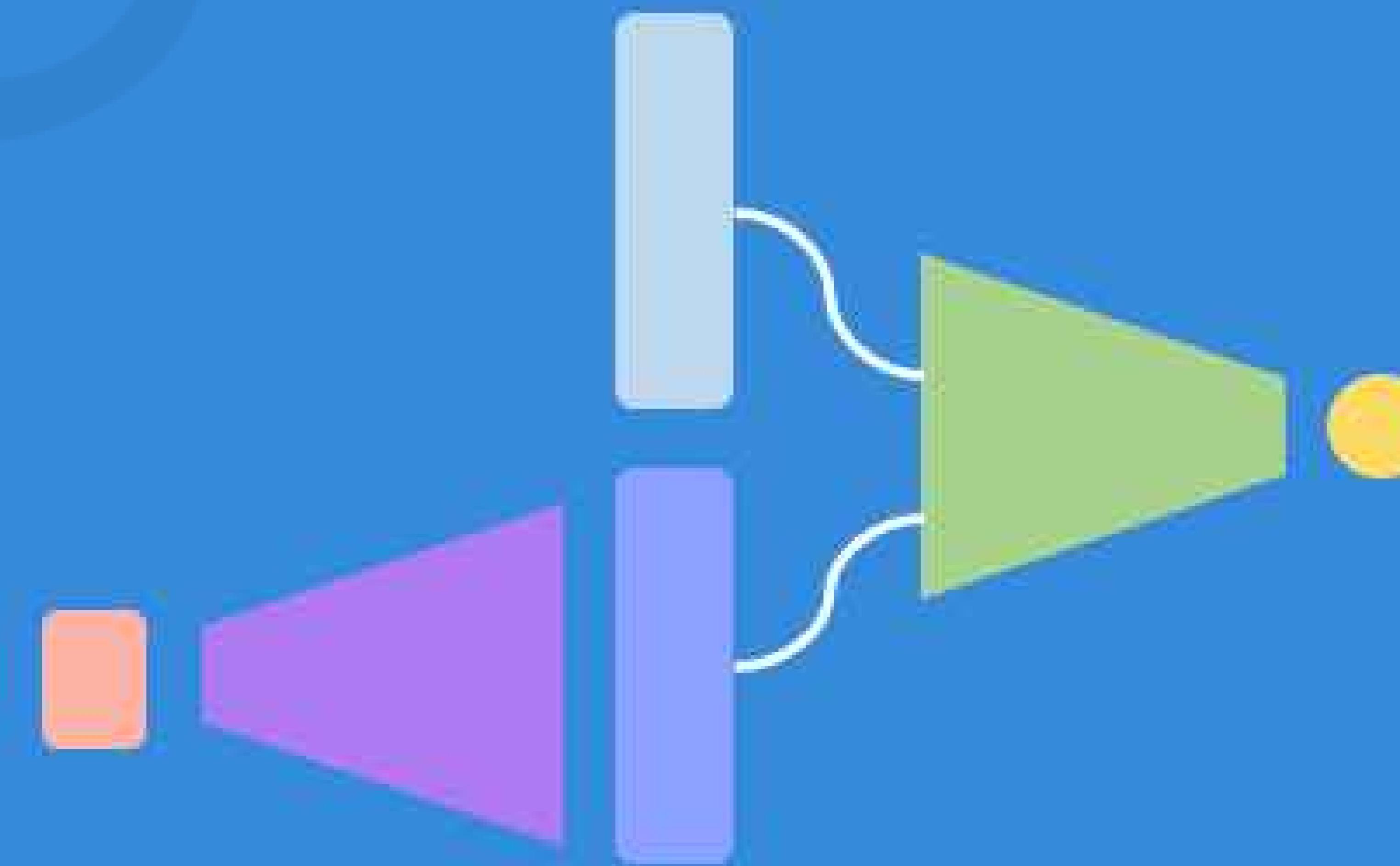
Pedestrians

Latent variable models

Autoencoders and Variational
Autoencoders (VAEs)



Generative Adversarial
Networks (GANs)



What is a latent variable?



Myth of the Cave

What is a latent variable?



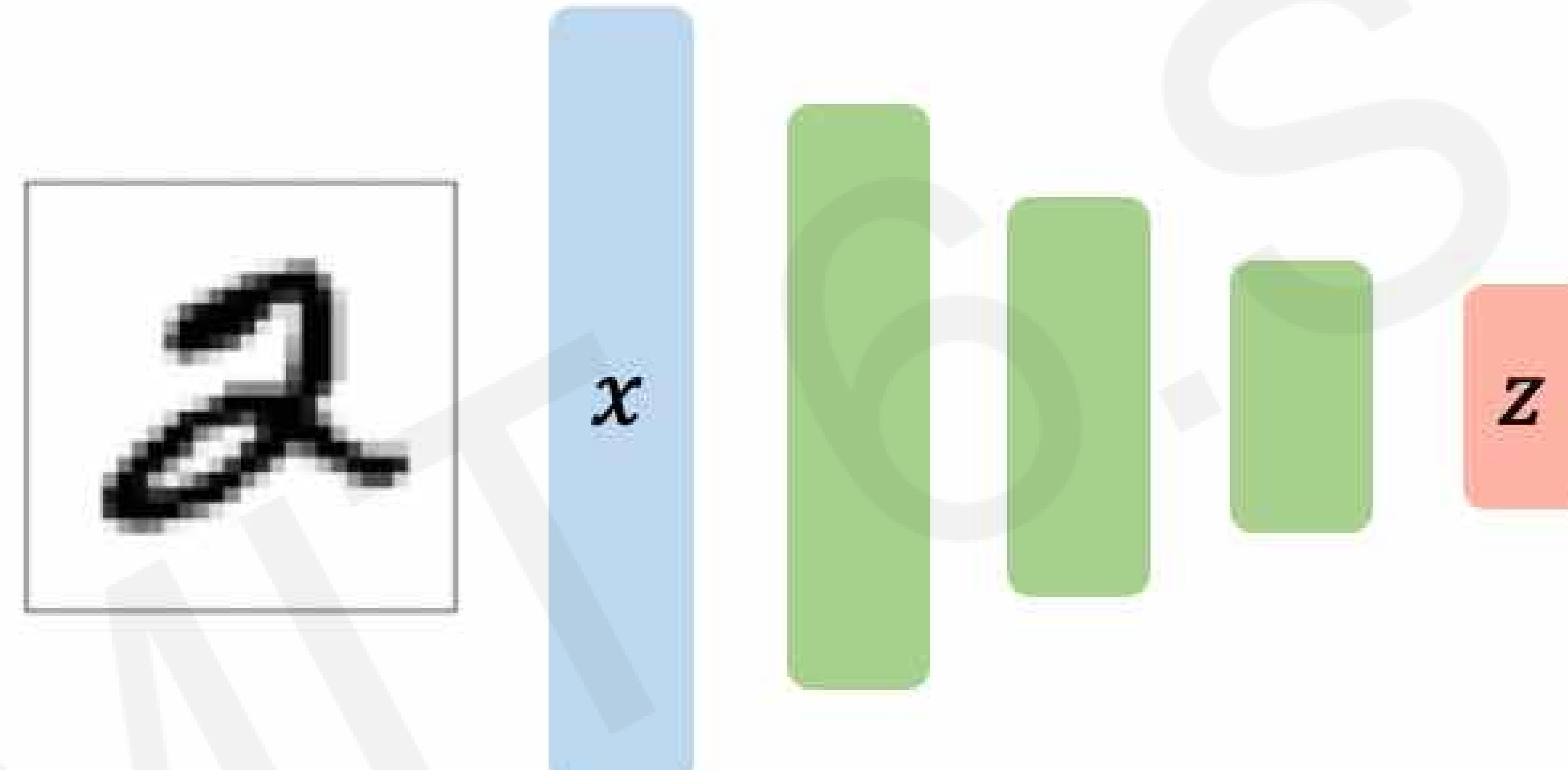
Can we learn the **true explanatory factors**, e.g. latent variables, from only observed data?

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 ?

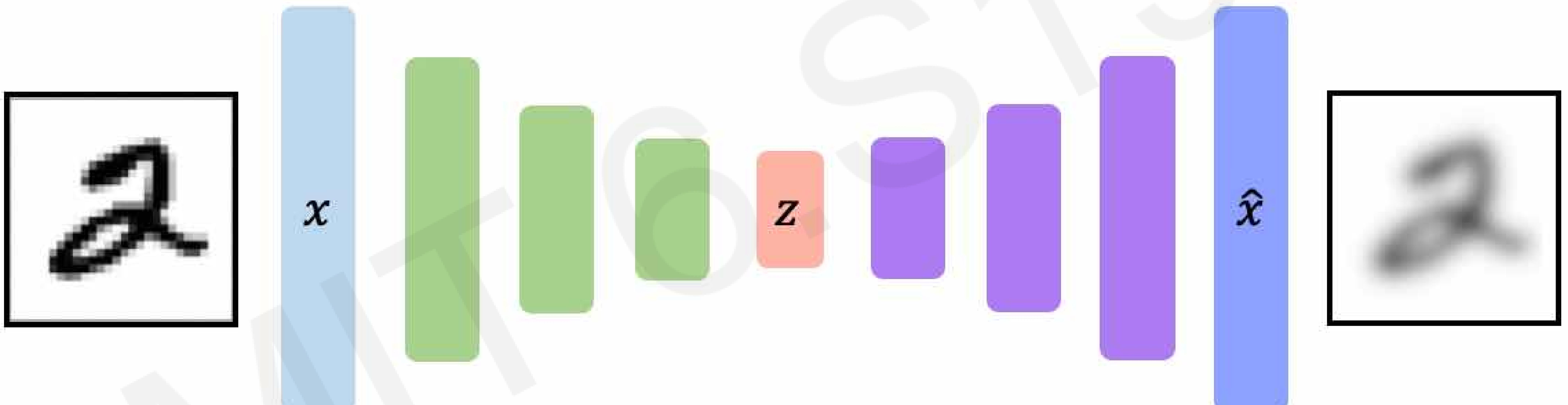


"Encoder" learns mapping from the data, x , to a low-dimensional latent space, z

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 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**



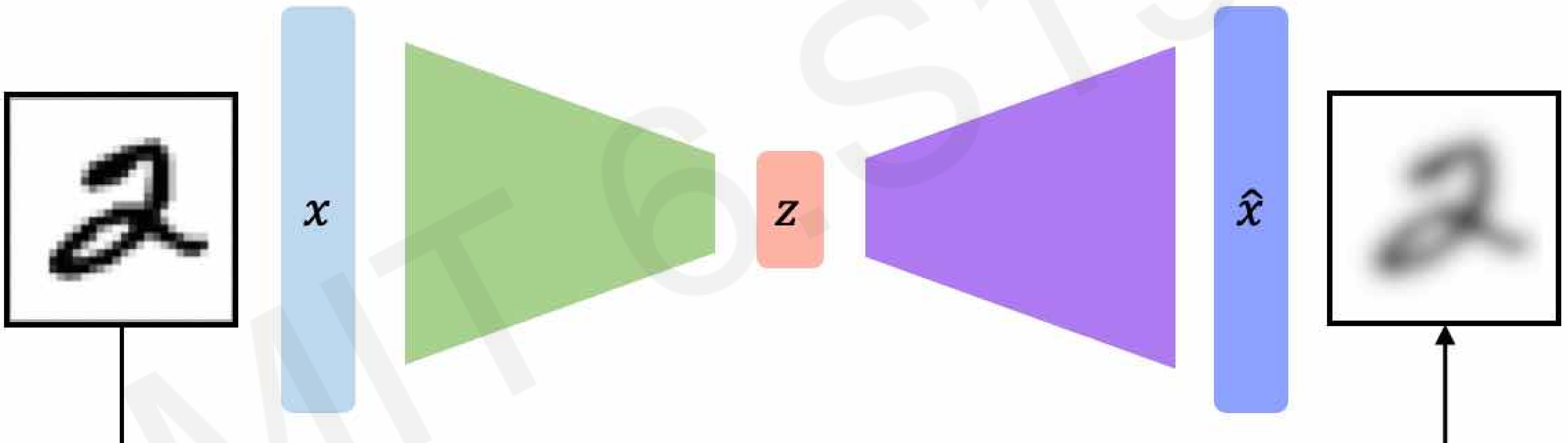
$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

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!!

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

5D latent space

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

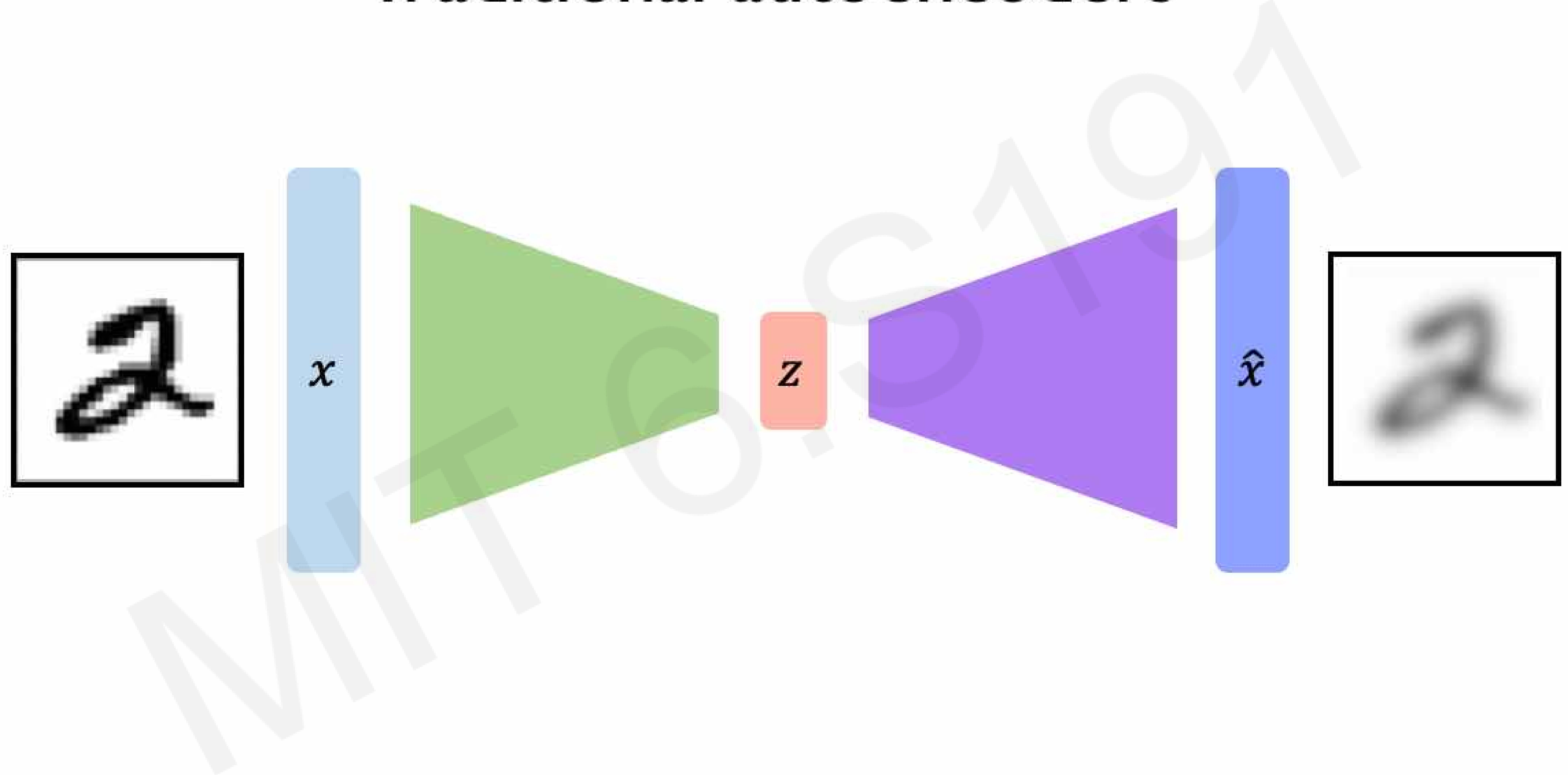
Reconstruction loss forces the latent representation to capture (or encode) as much “information” about the data as possible

Autoencoding = **A**utomatically **e**ncoding data; “**Auto**” = **s**elf-encoding

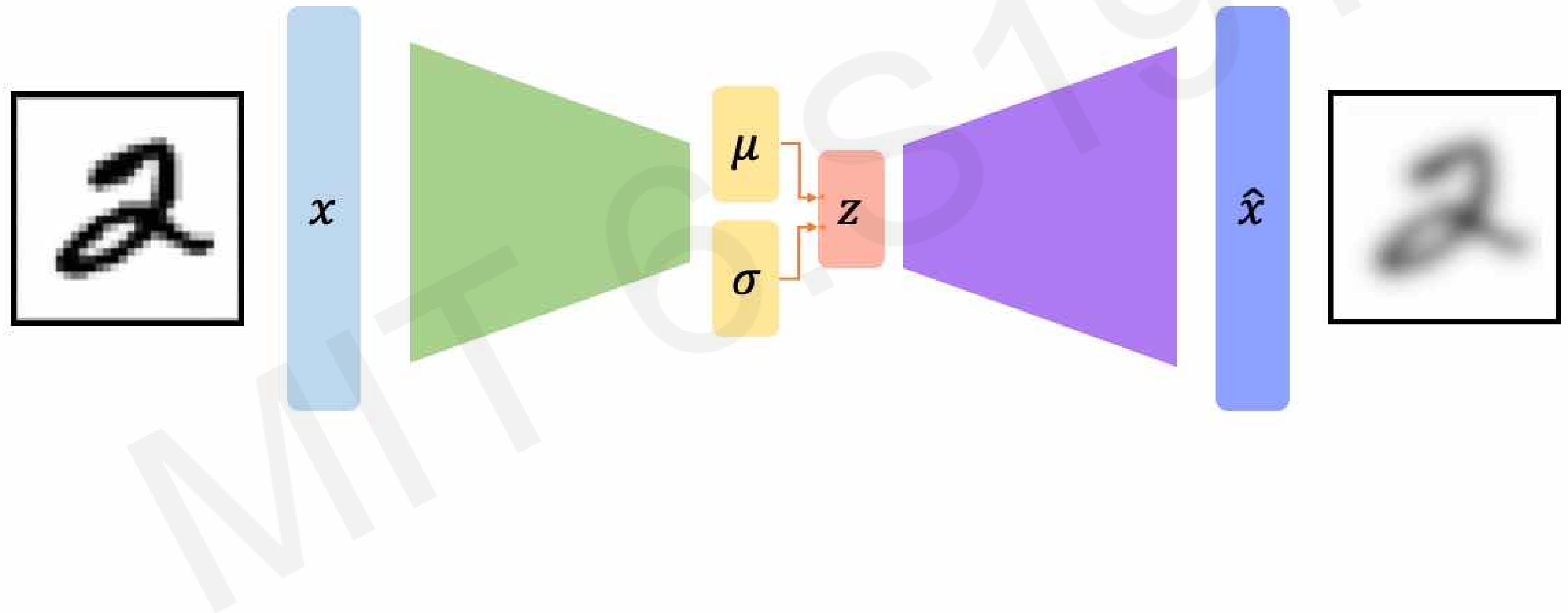
Variational Autoencoders (VAEs)



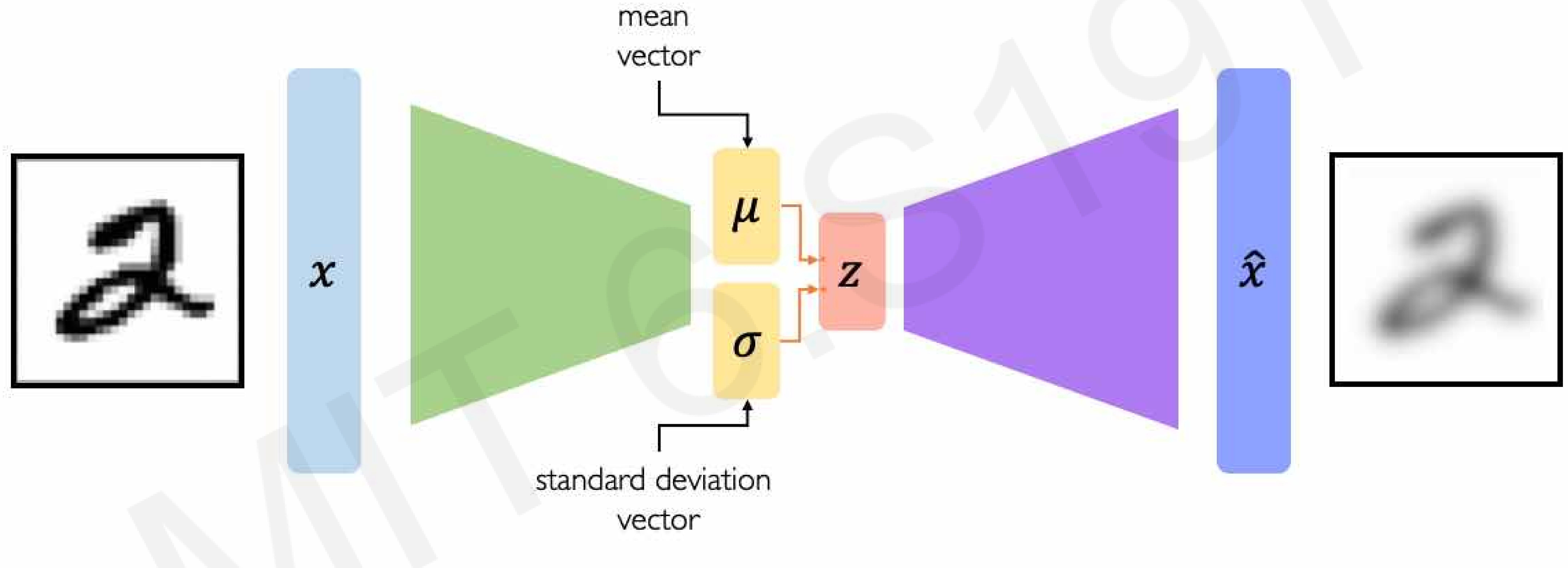
Traditional autoencoders



VAEs: key difference with traditional autoencoder



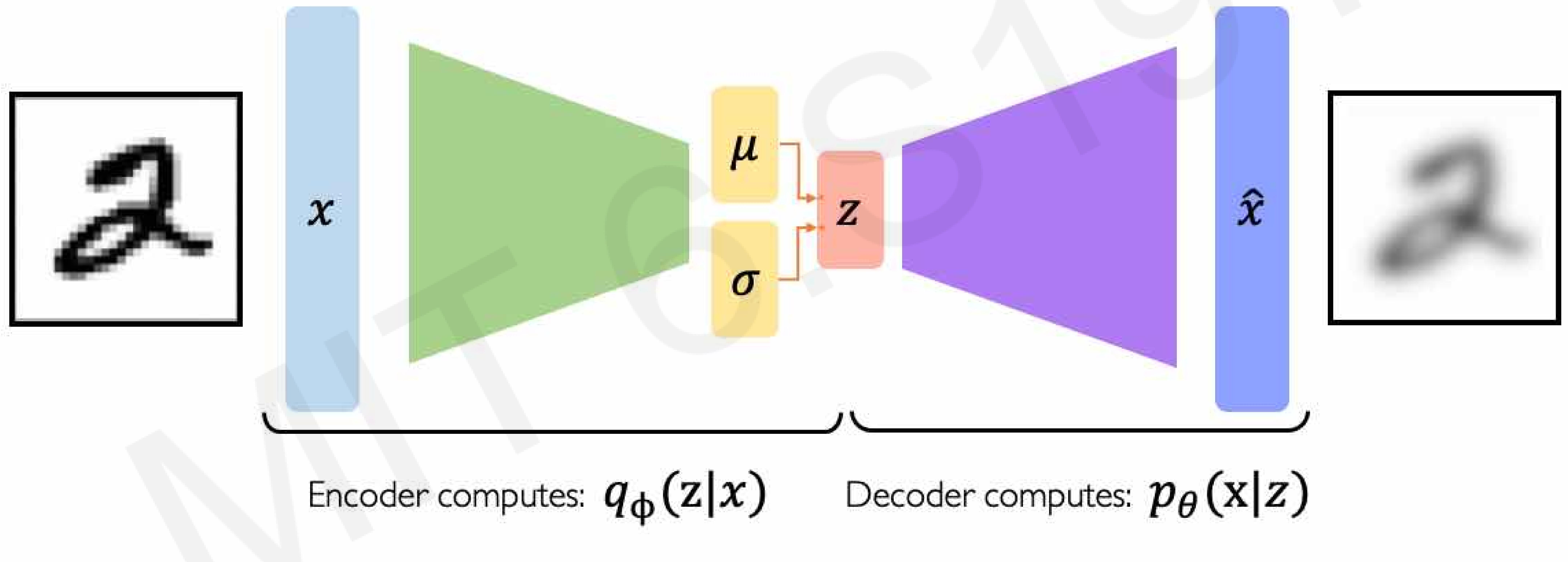
VAEs: key difference with traditional autoencoder



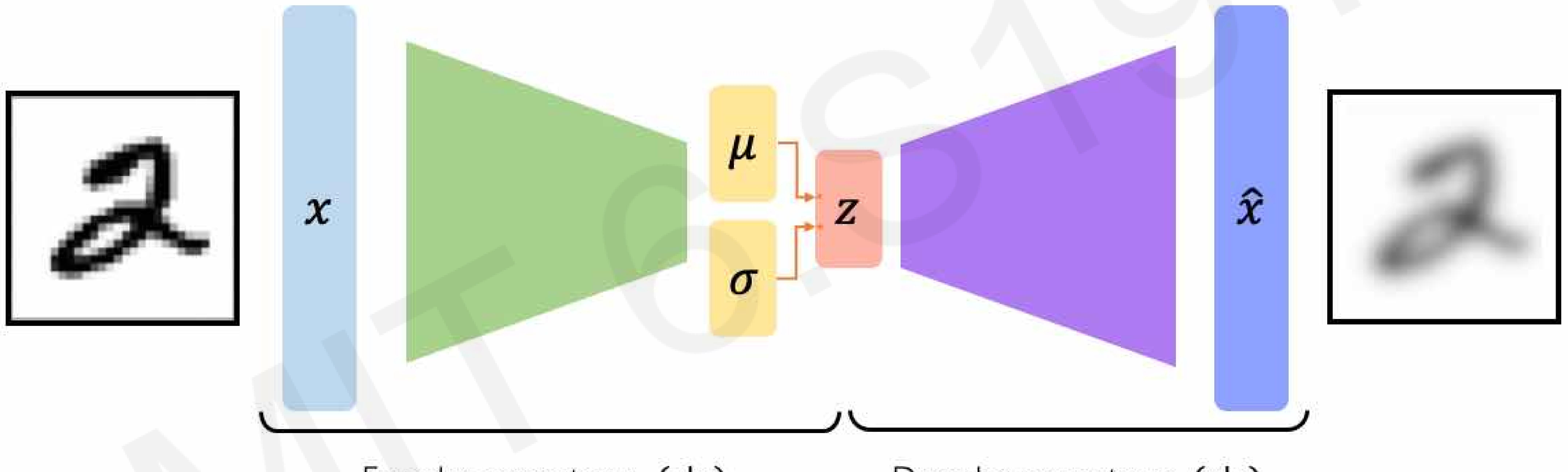
Variational autoencoders are a probabilistic twist on autoencoders!

Sample from the mean and standard deviation to compute latent sample

VAE optimization

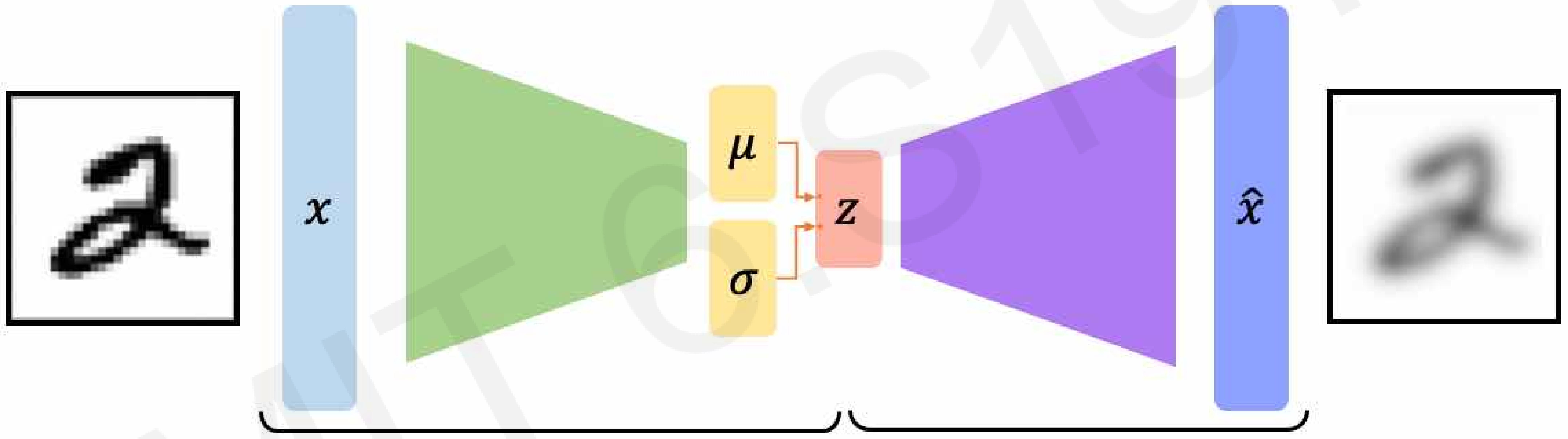


VAE optimization



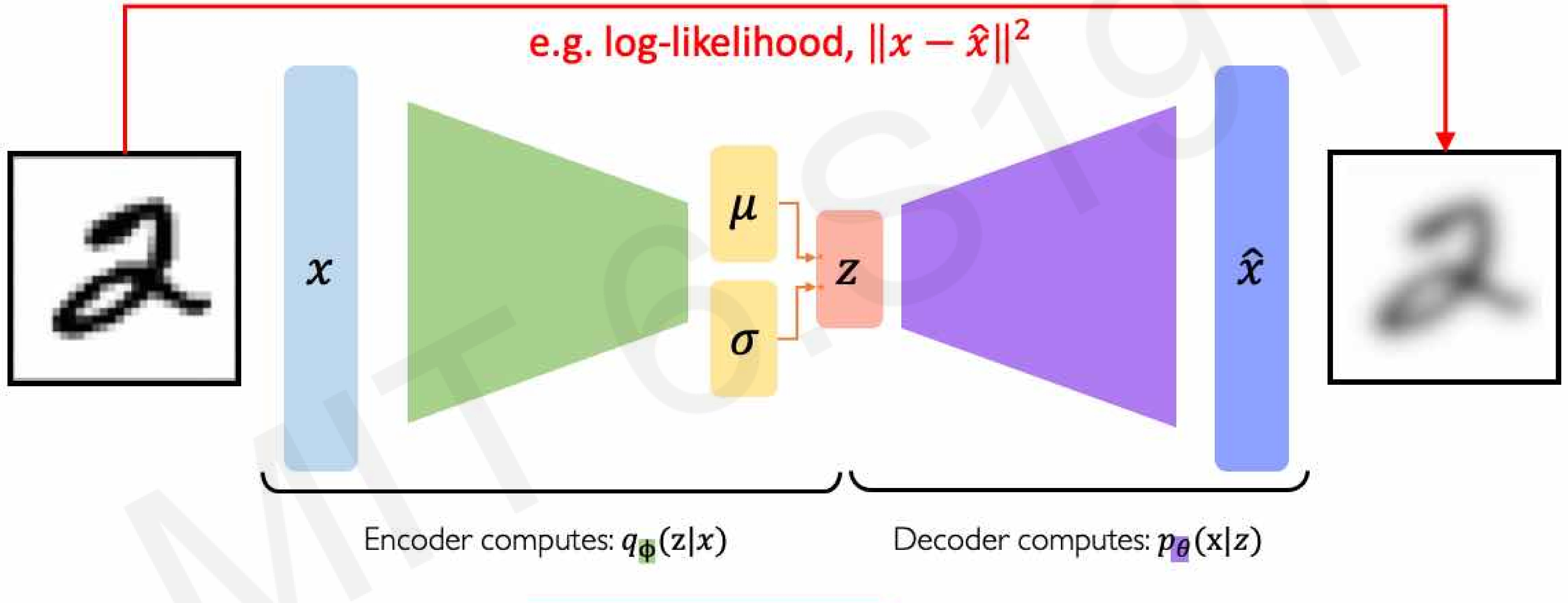
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

VAE optimization



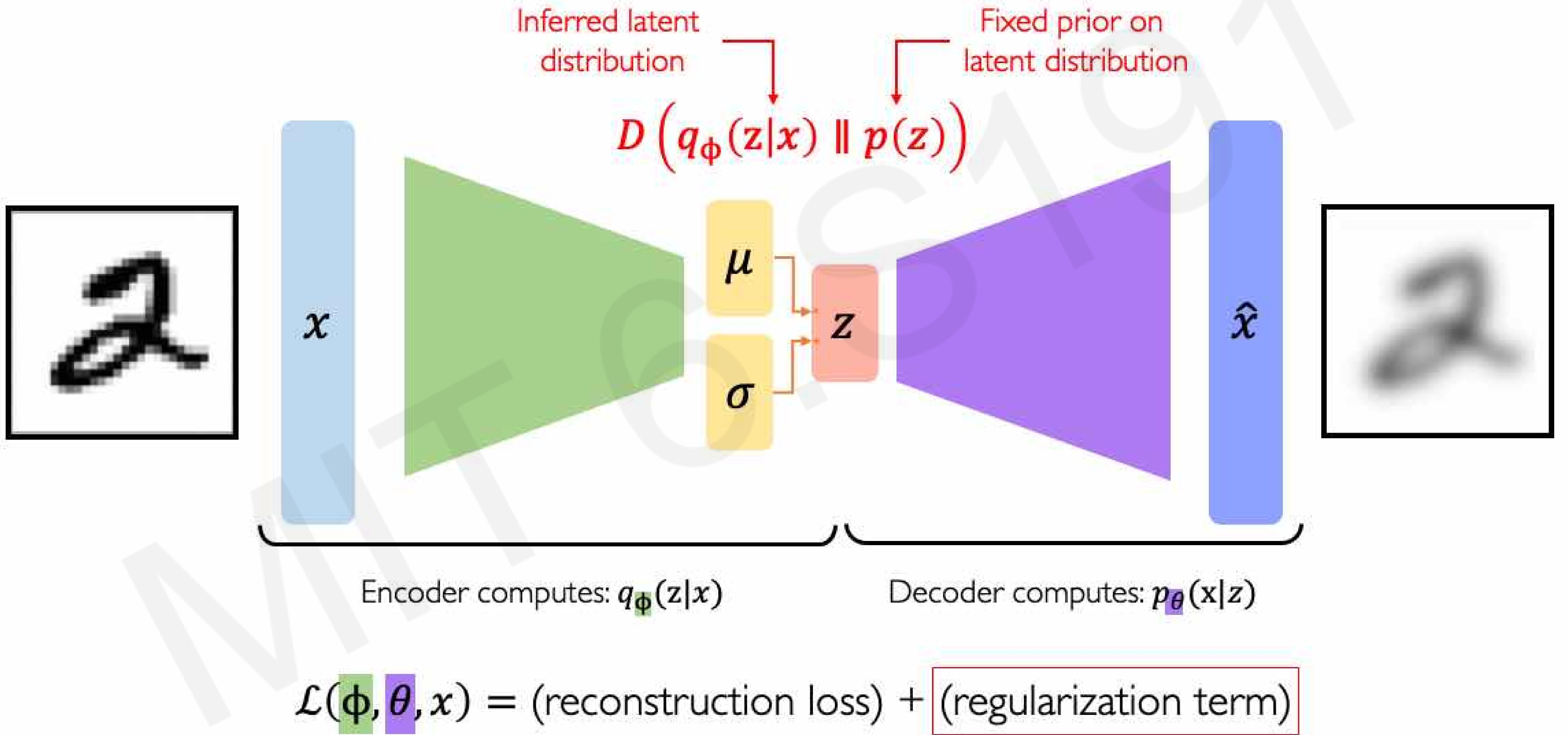
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

VAE optimization



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

VAE optimization

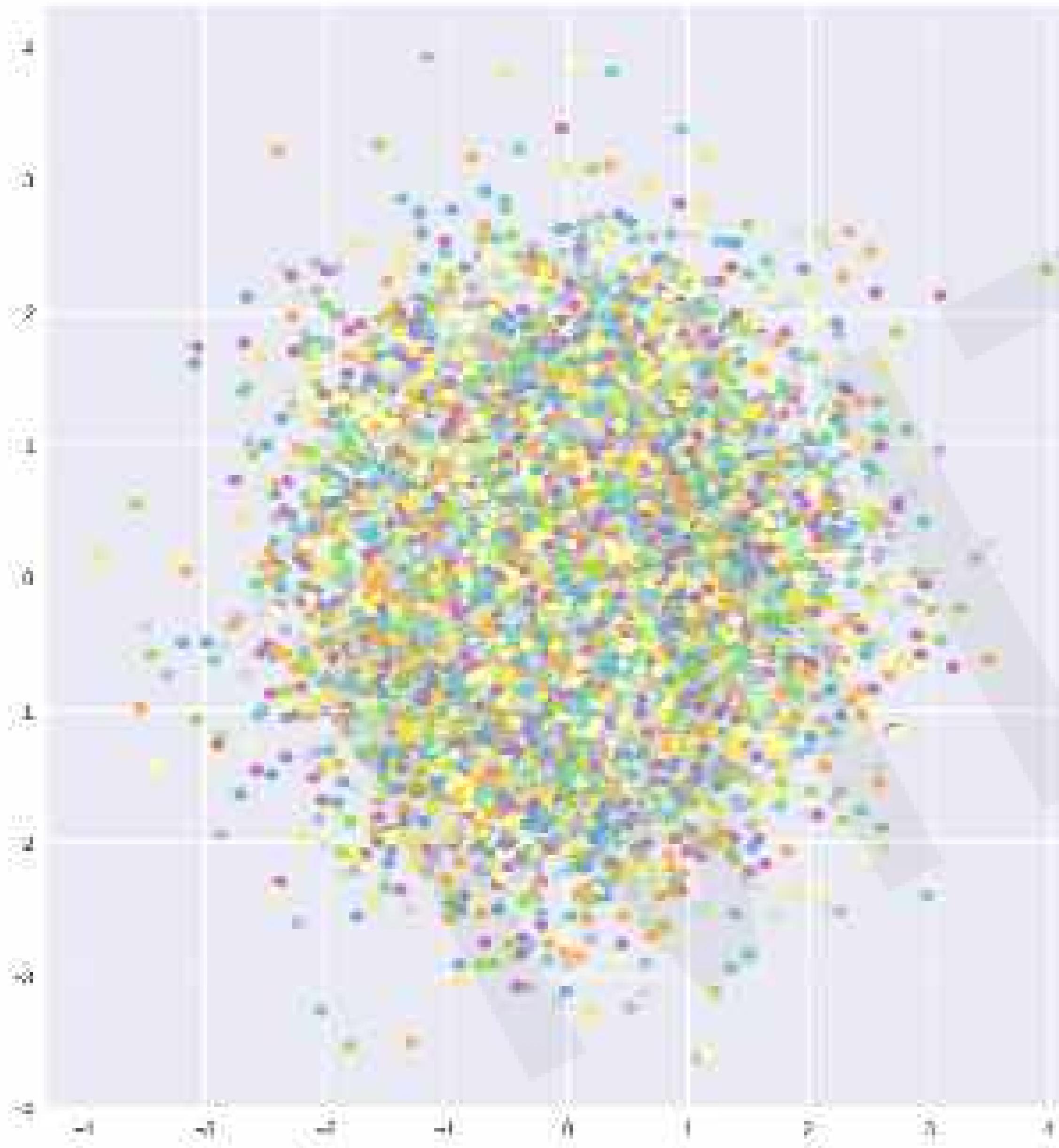


Priors on the latent distribution

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

Inferred latent
distribution

Fixed prior on
latent distribution



Common choice of prior – Normal Gaussian:

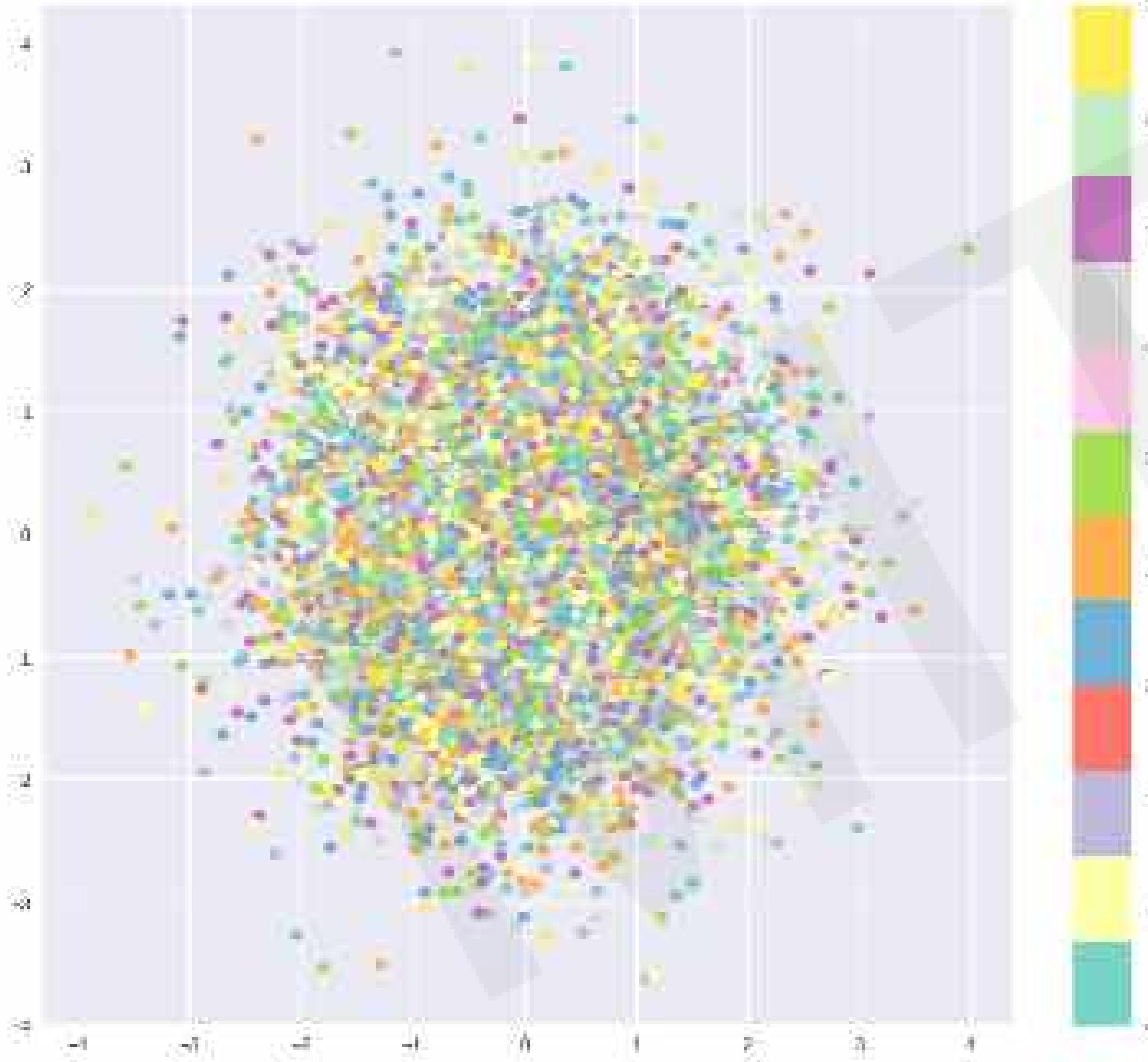
$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- 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

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

KL-divergence
between the two
distributions



Common choice of prior – Normal Gaussian:

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

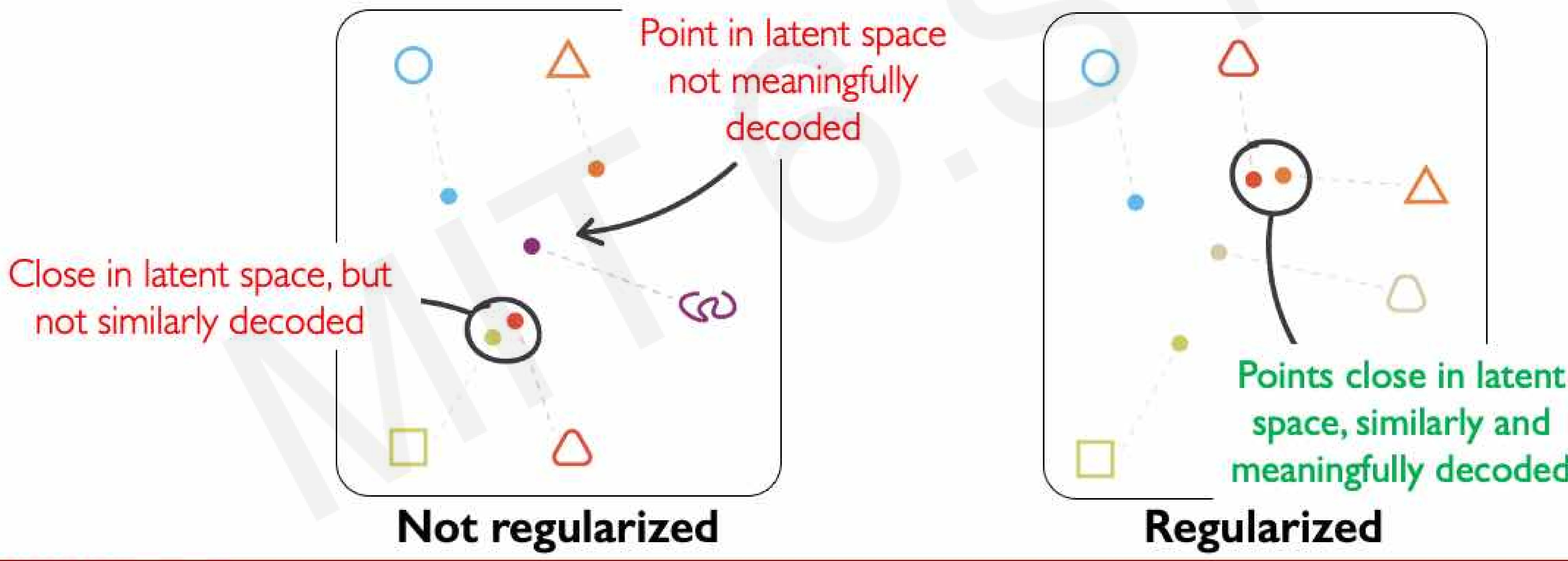
- Encourages encodings to distribute encodings evenly around the center of the latent space
- Penalize the network when it tries to “cheat” by clustering points in specific regions (i.e., by memorizing the data)

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



Intuition on regularization and the Normal prior

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

Encoding as a distribution does not
guarantee these properties!

Small variances →
Pointed distributions

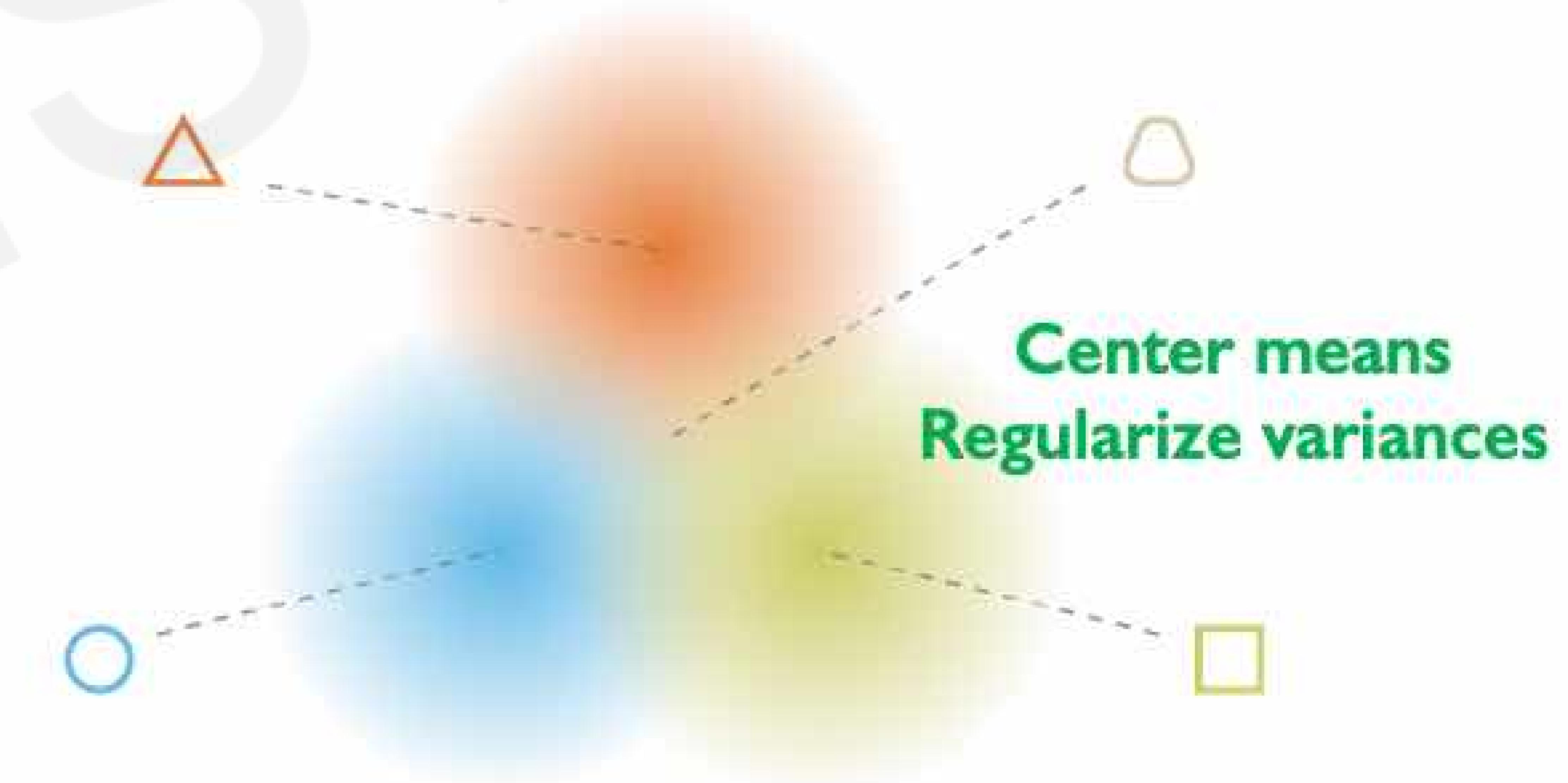
Different means →
Discontinuities

Not regularized

Normal prior →
continuity + completeness

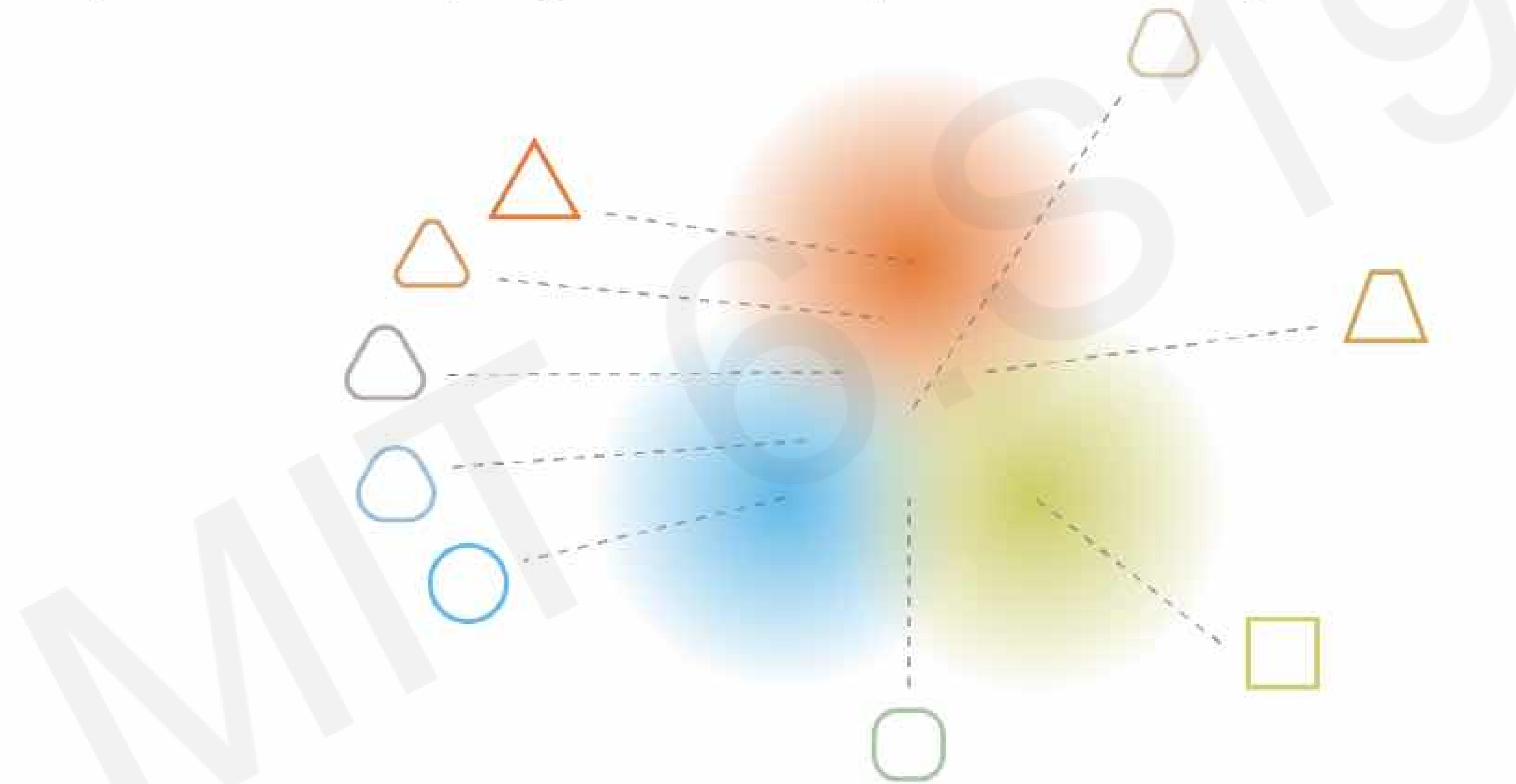
Center means
Regularize variances

Regularized



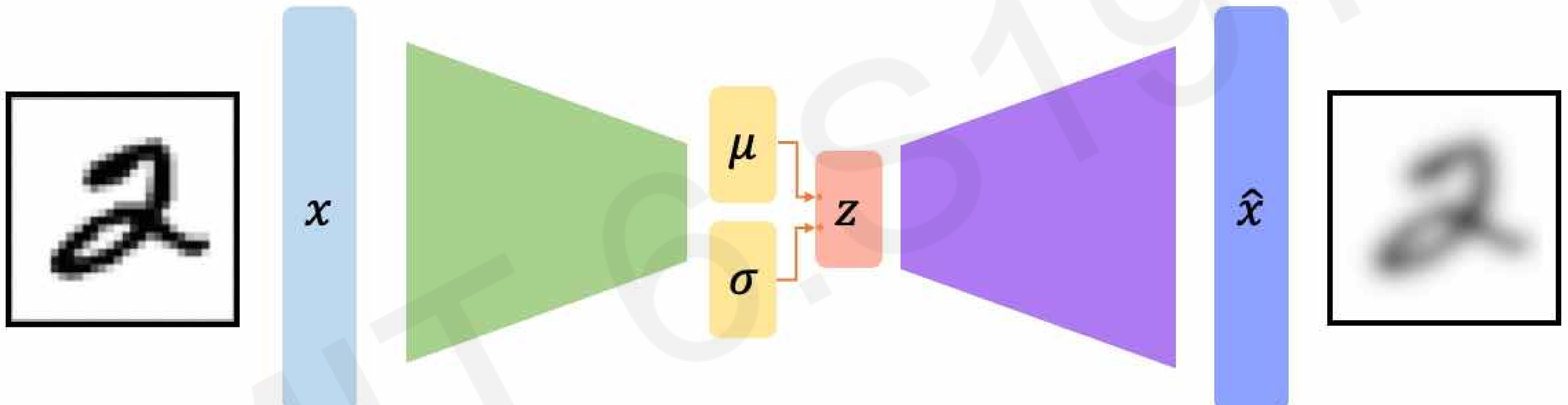
Intuition on regularization and the Normal prior

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



Regularization with Normal prior helps enforce **information gradient** in the latent space.

VAE computation graph



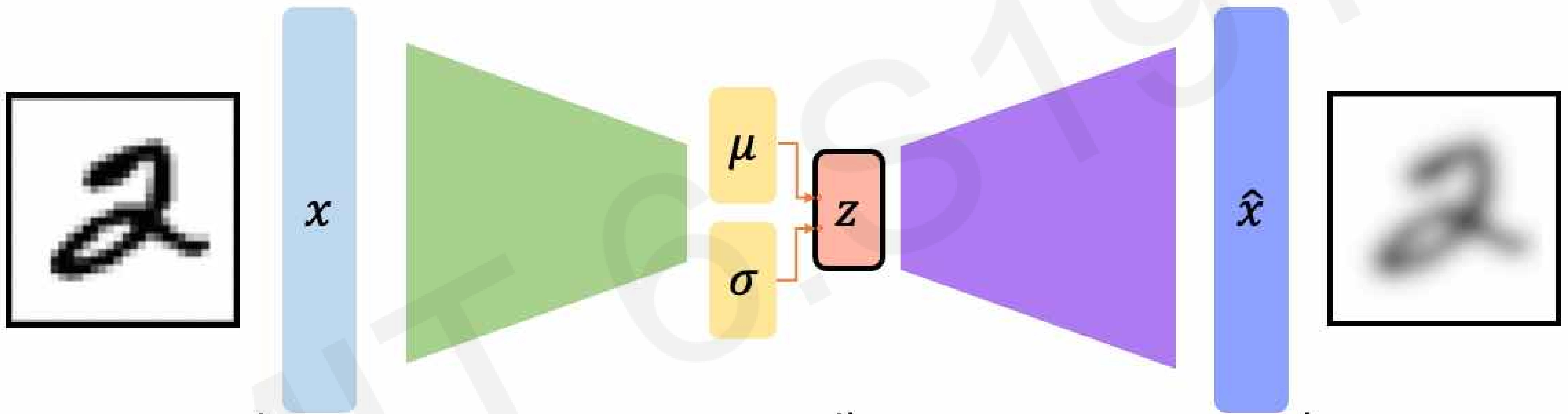
Encoder computes: $q_\phi(z|x)$

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

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

VAE computation graph

Problem: We cannot backpropagate gradients through sampling layers!

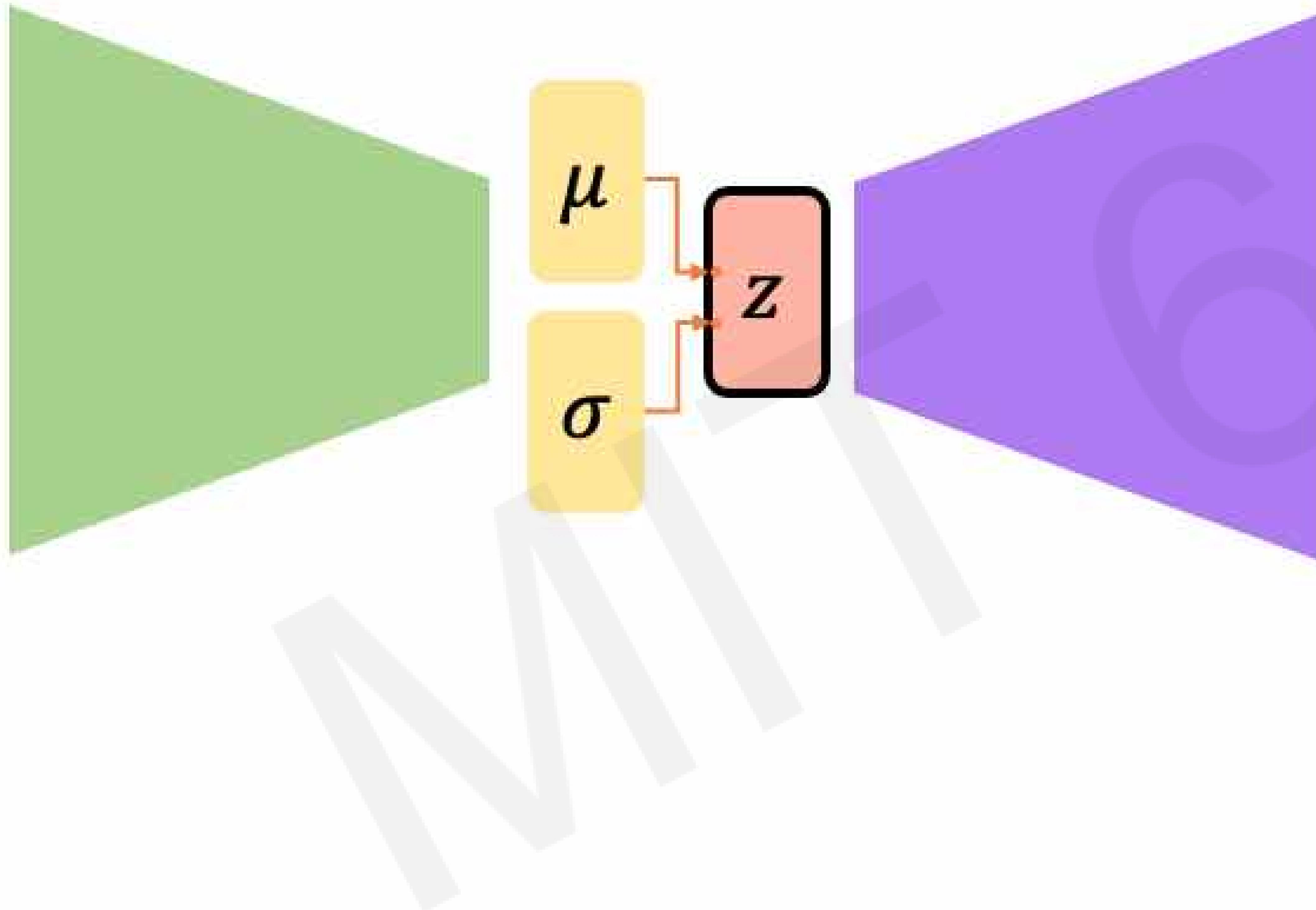


Encoder computes: $q_\phi(z|x)$

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

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

Reparametrizing the sampling layer



Key Idea:

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

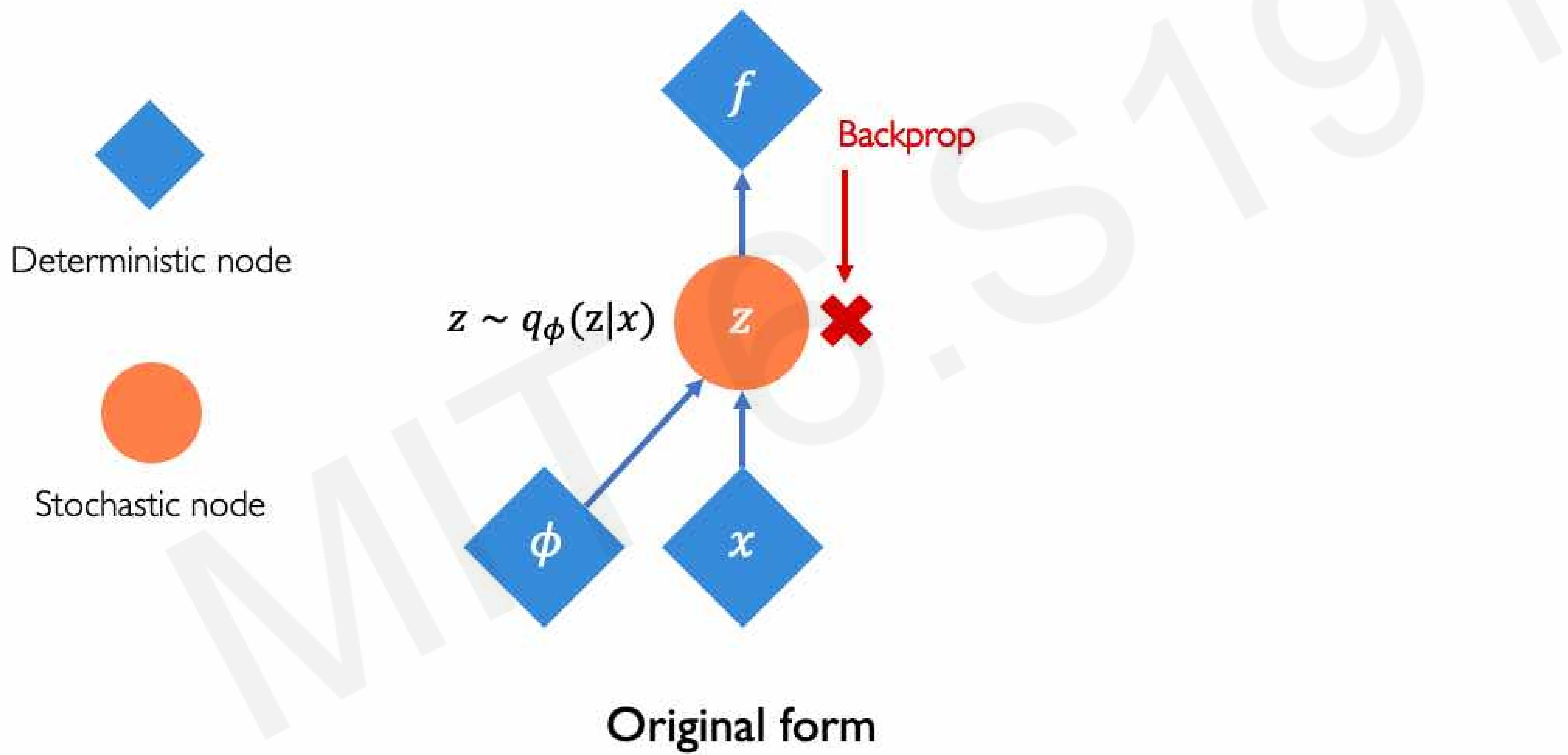
Consider the sampled latent vector z as a sum of

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

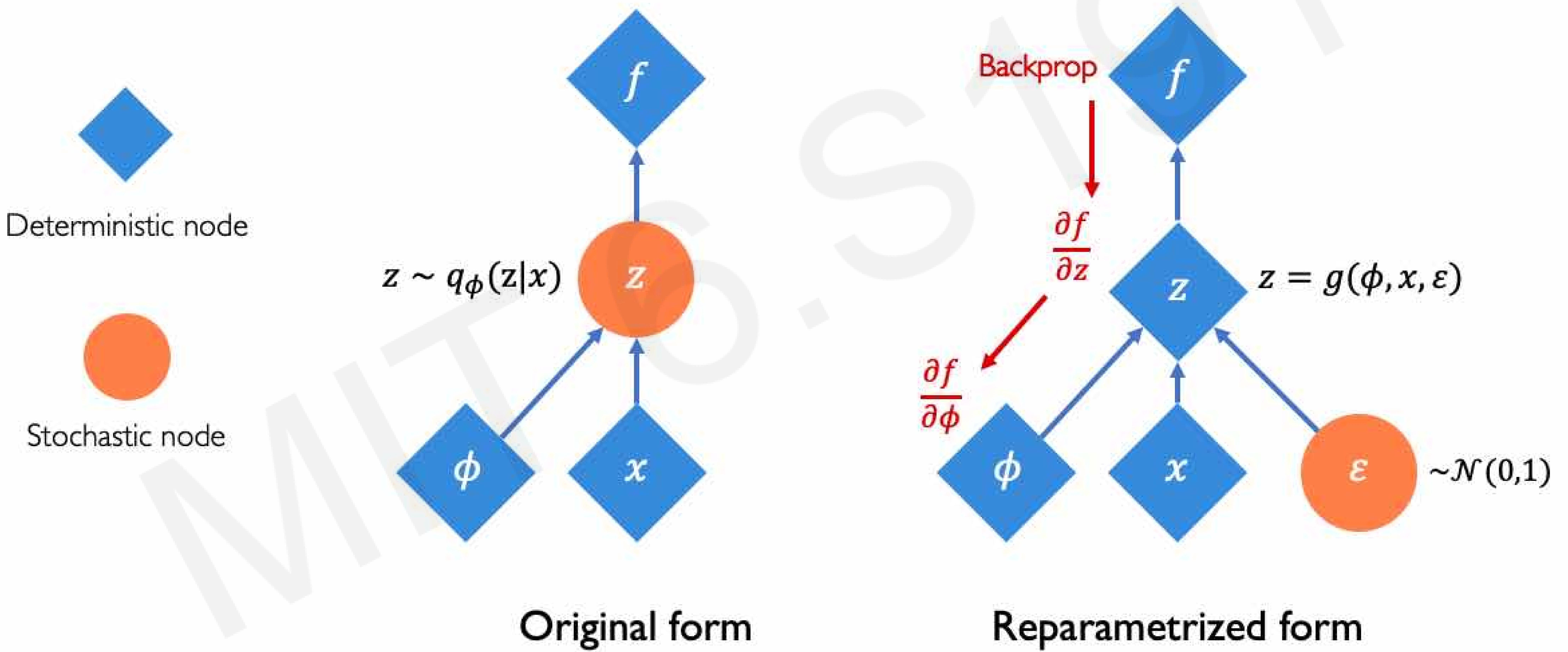
$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

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

Reparametrizing the sampling layer

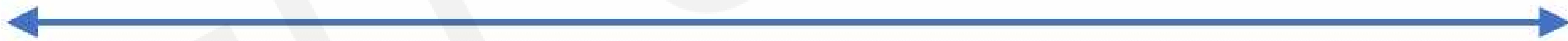


Reparametrizing the sampling layer



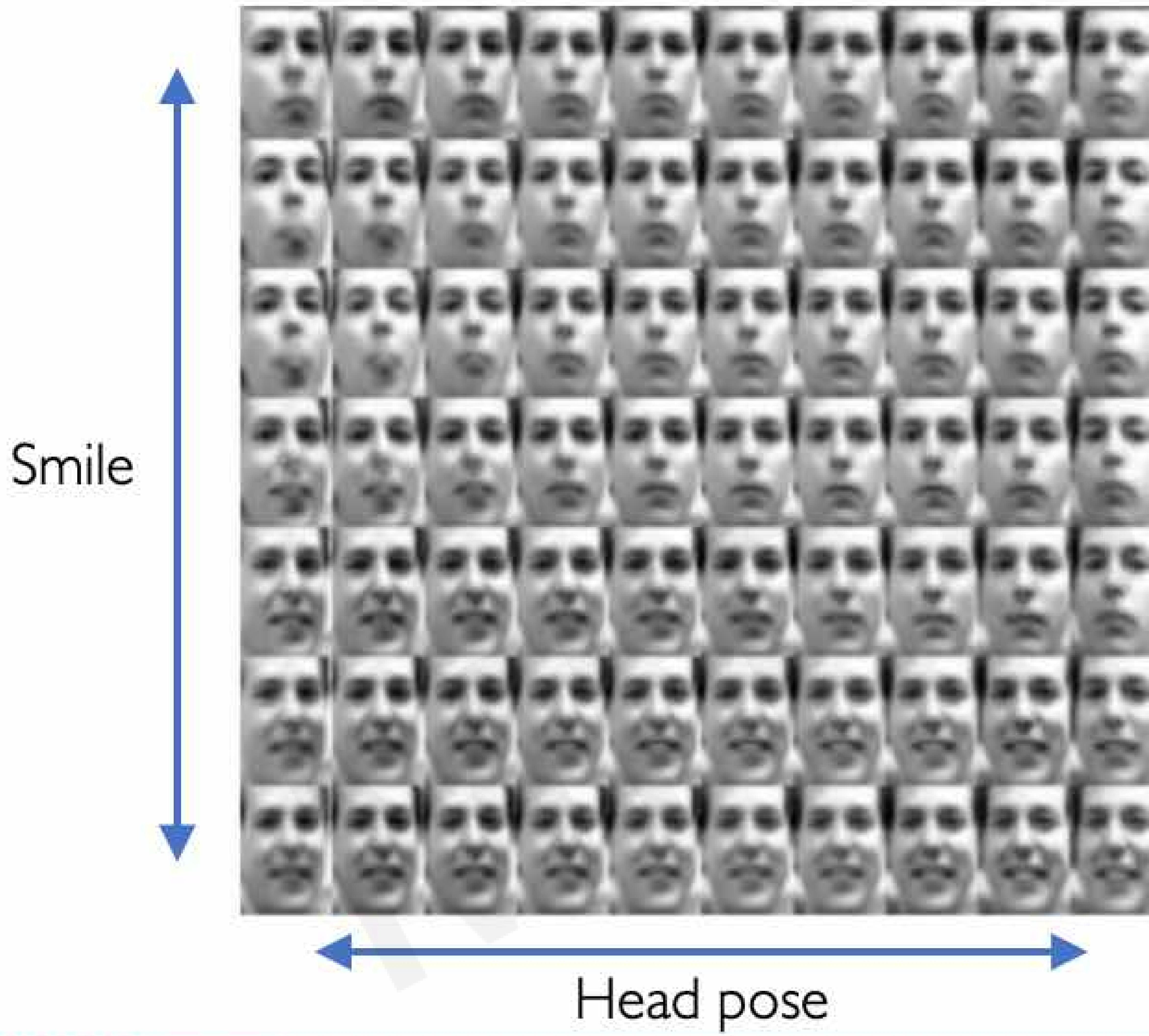
VAEs: Latent perturbation

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



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

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

Latent space disentanglement with β -VAE

8th Grade SAE Loss:

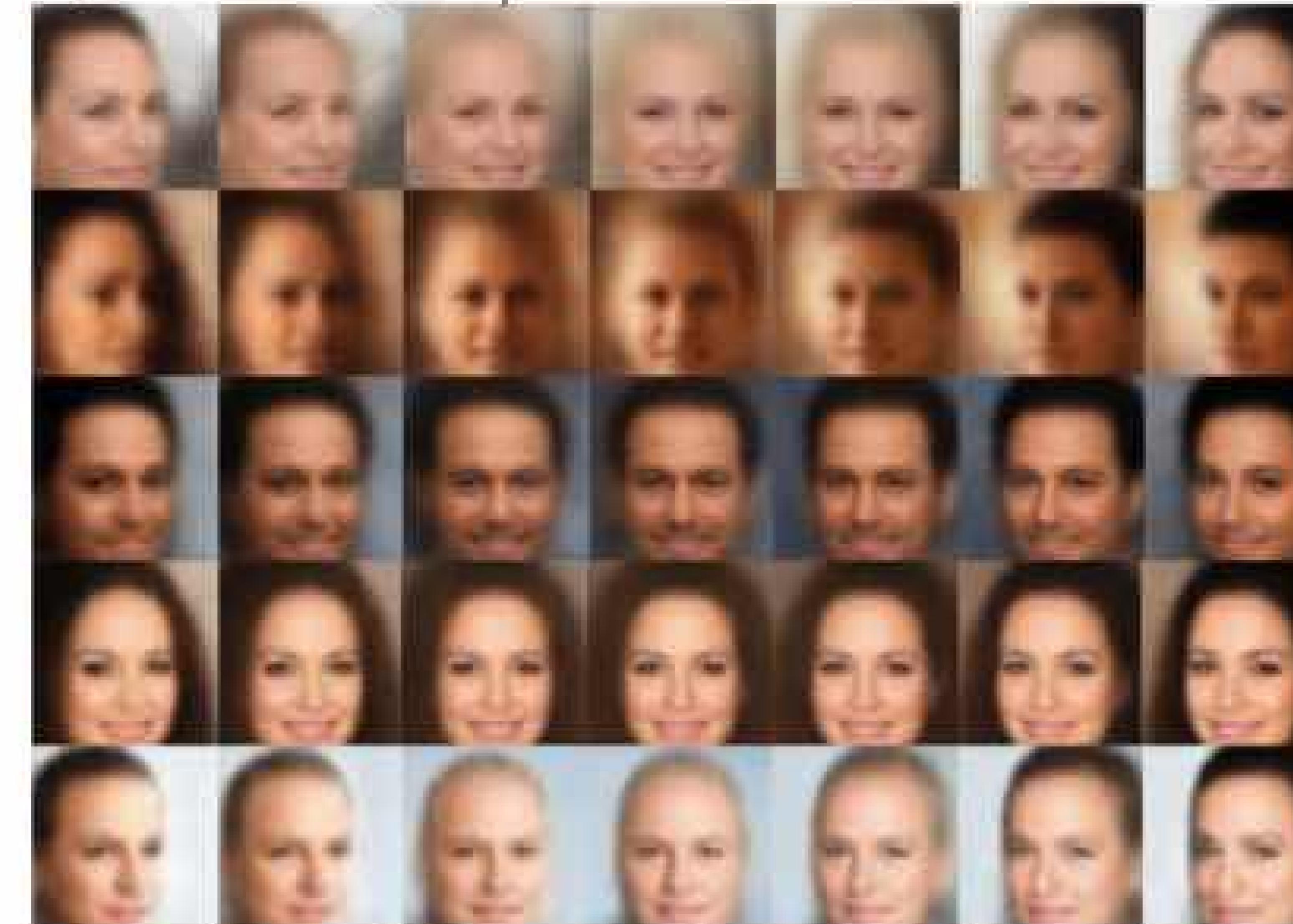
$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

$\beta > 1$: constrain latent bottleneck, encourage efficient latent encoding \rightarrow disentanglement

Head rotation (azimuth)



Smile also
changing!

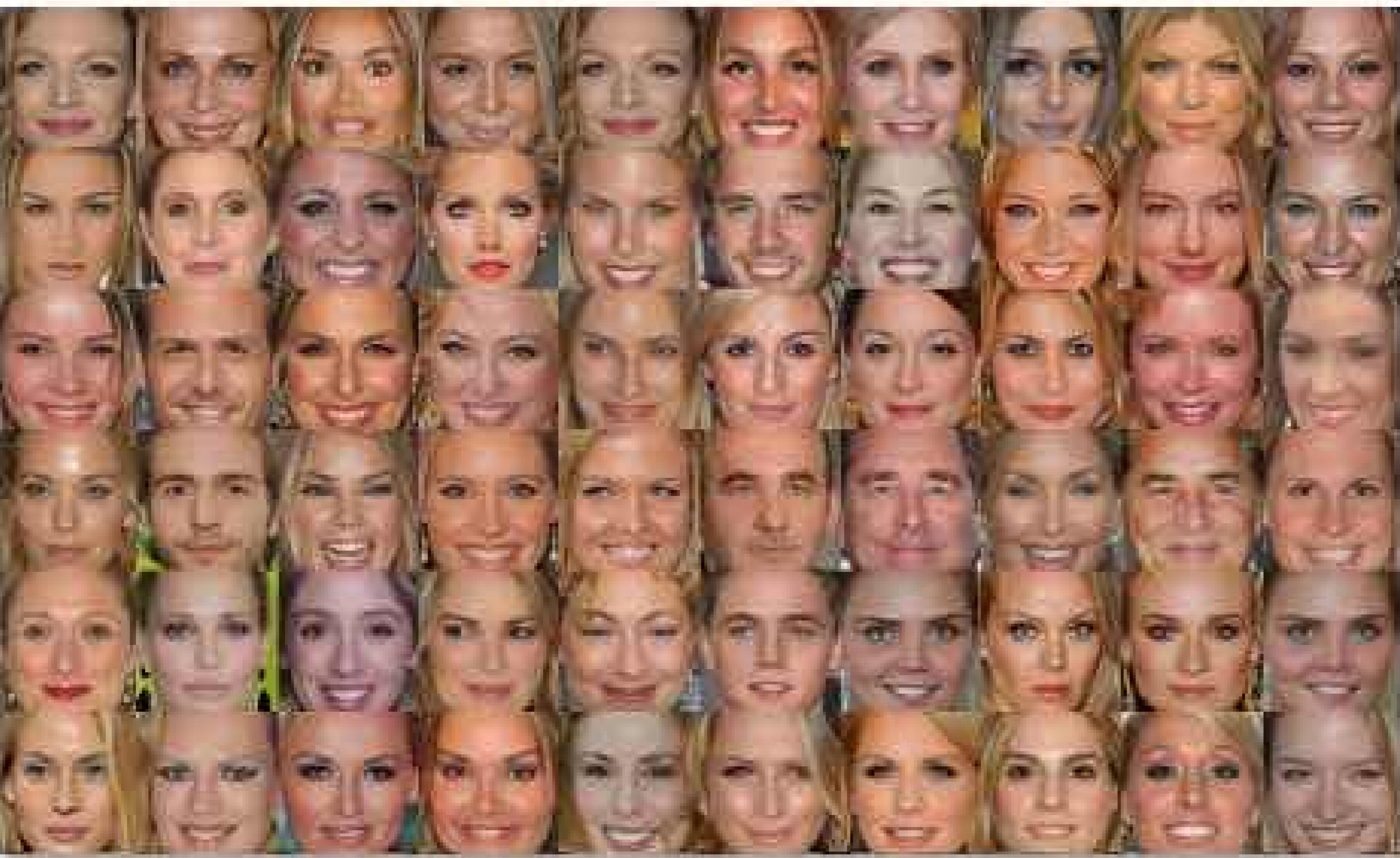


Smile relatively
constant!

Standard VAE ($\beta = 1$)

Why latent variable models? Debiasing

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



Homogeneous skin color, pose

VS



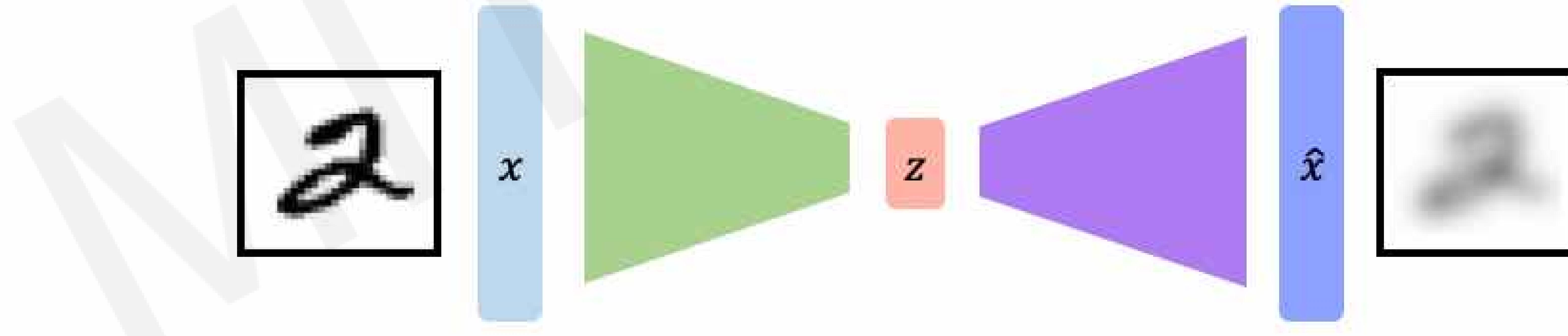
Diverse skin color, pose, illumination

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



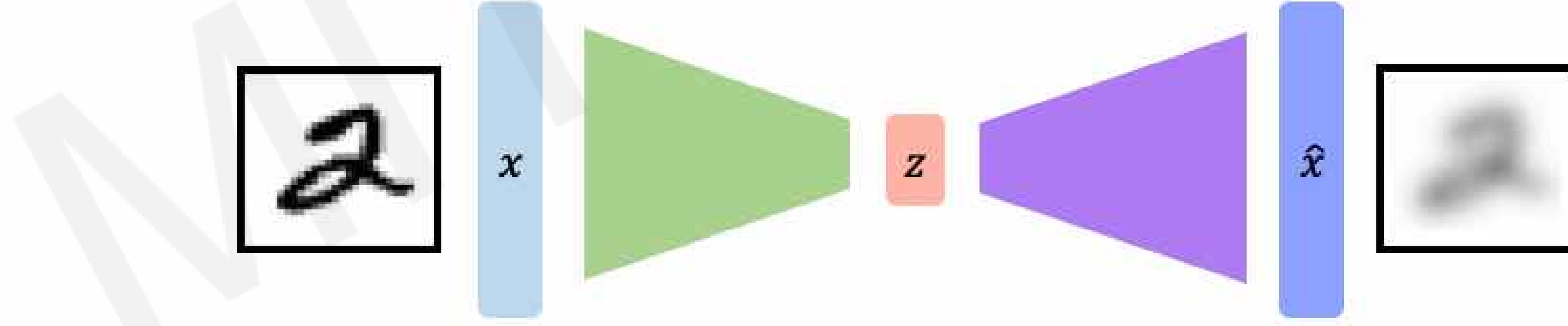
VAE summary

- I. Compress representation of world to something we can use to learn



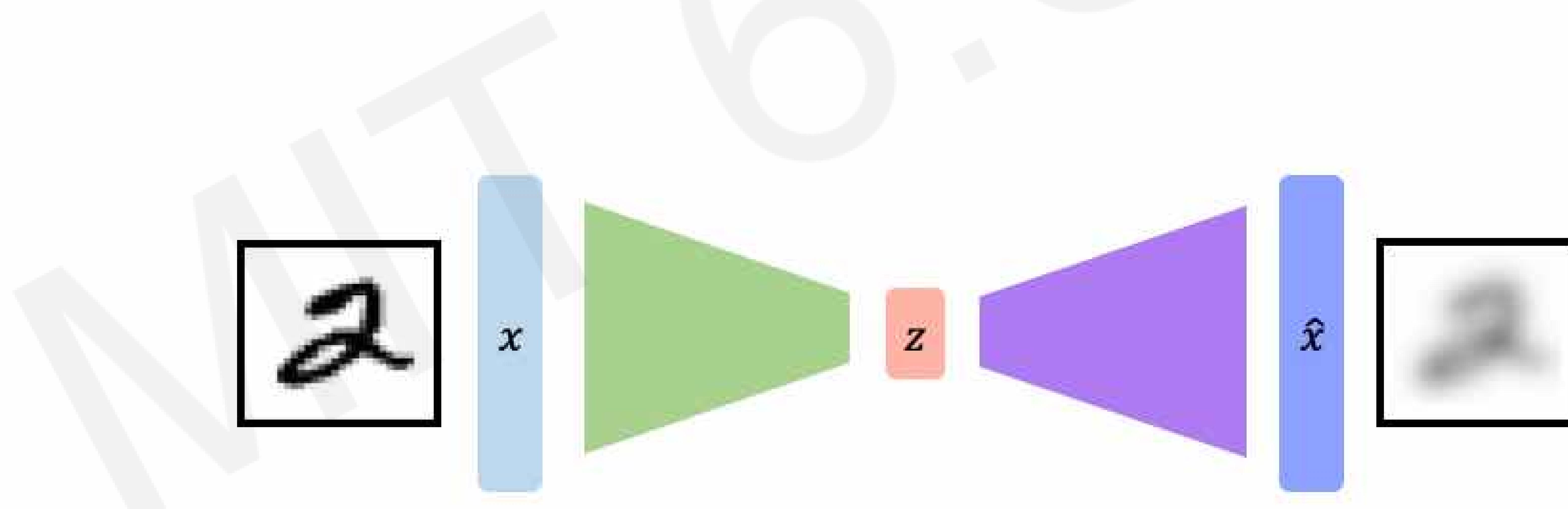
VAE summary

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



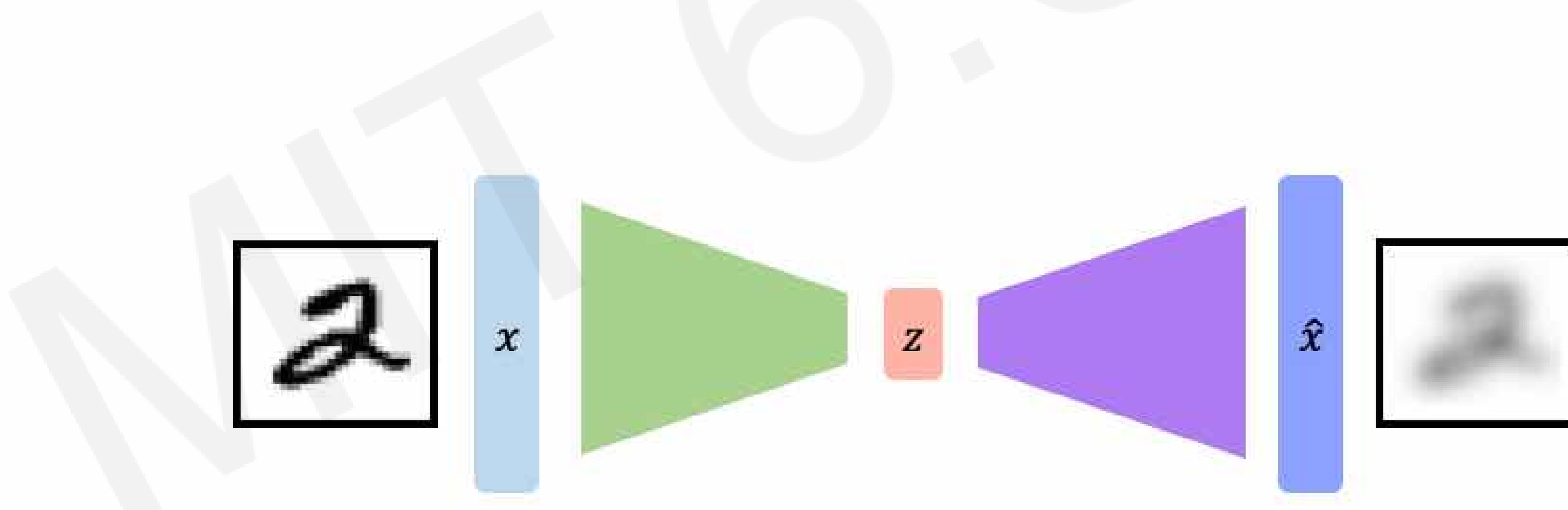
VAE summary

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



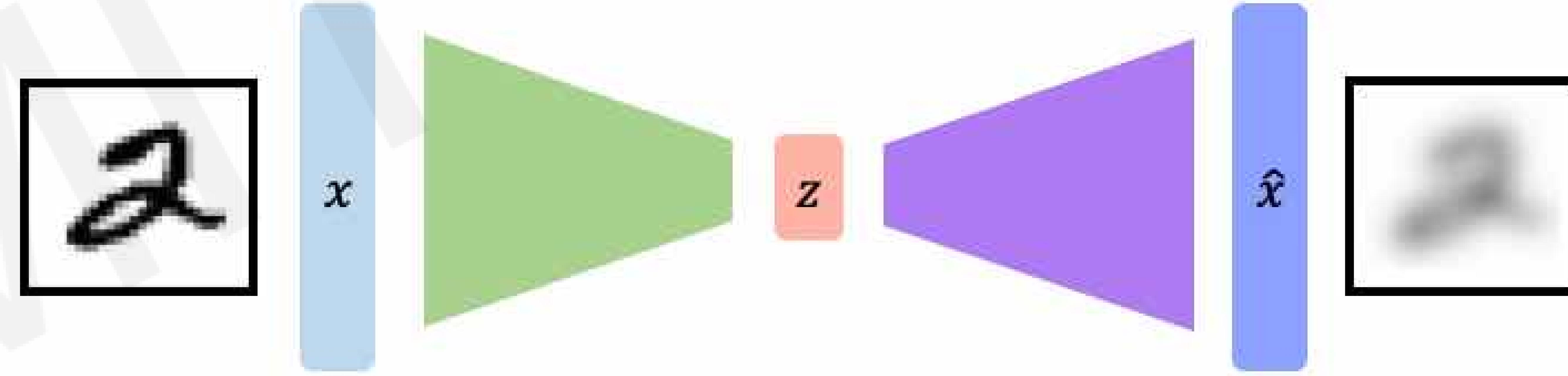
VAE summary

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



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)

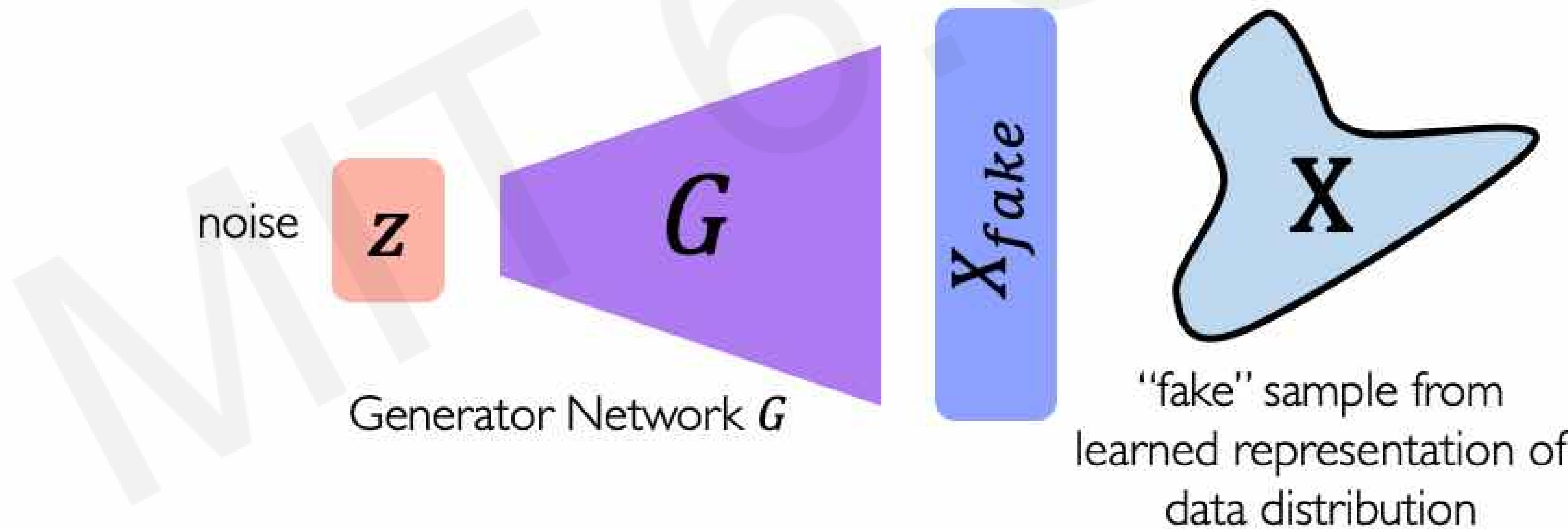


What if we just want to sample?

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!

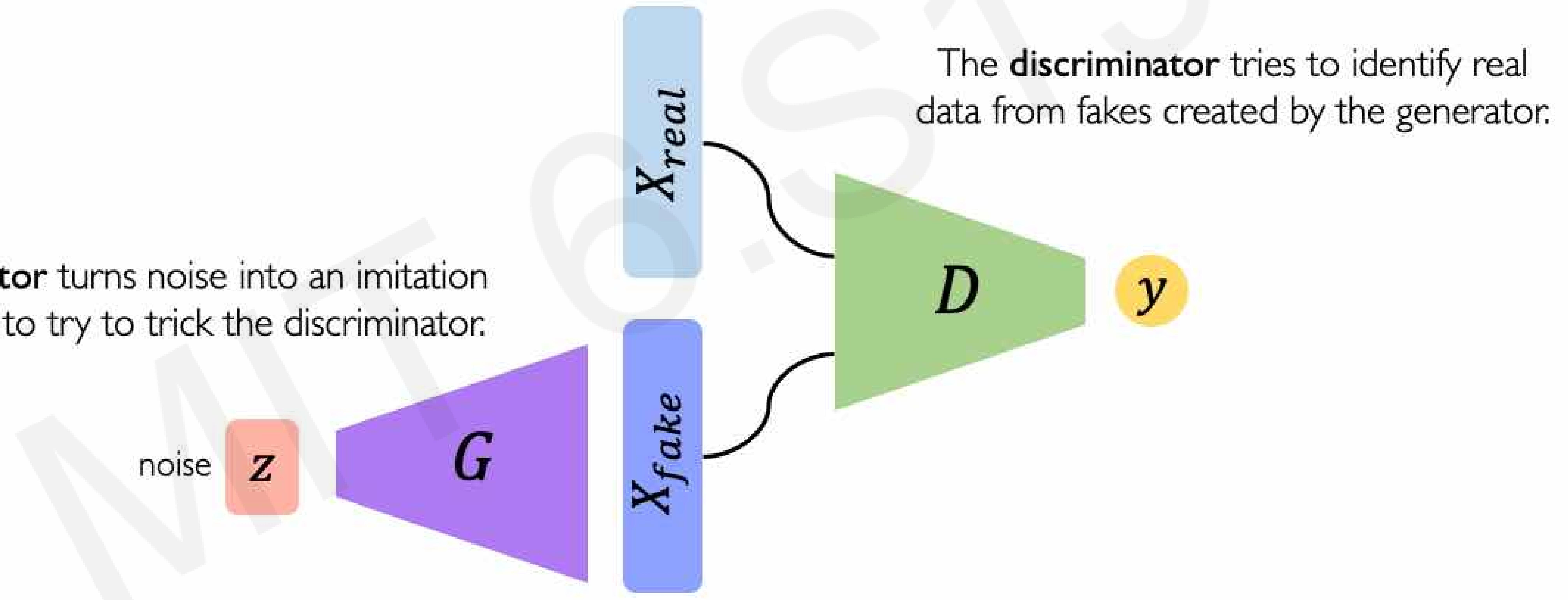
Solution: sample from something simple (e.g., noise), learn a transformation to the data distribution.



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.

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



Intuition behind GANs

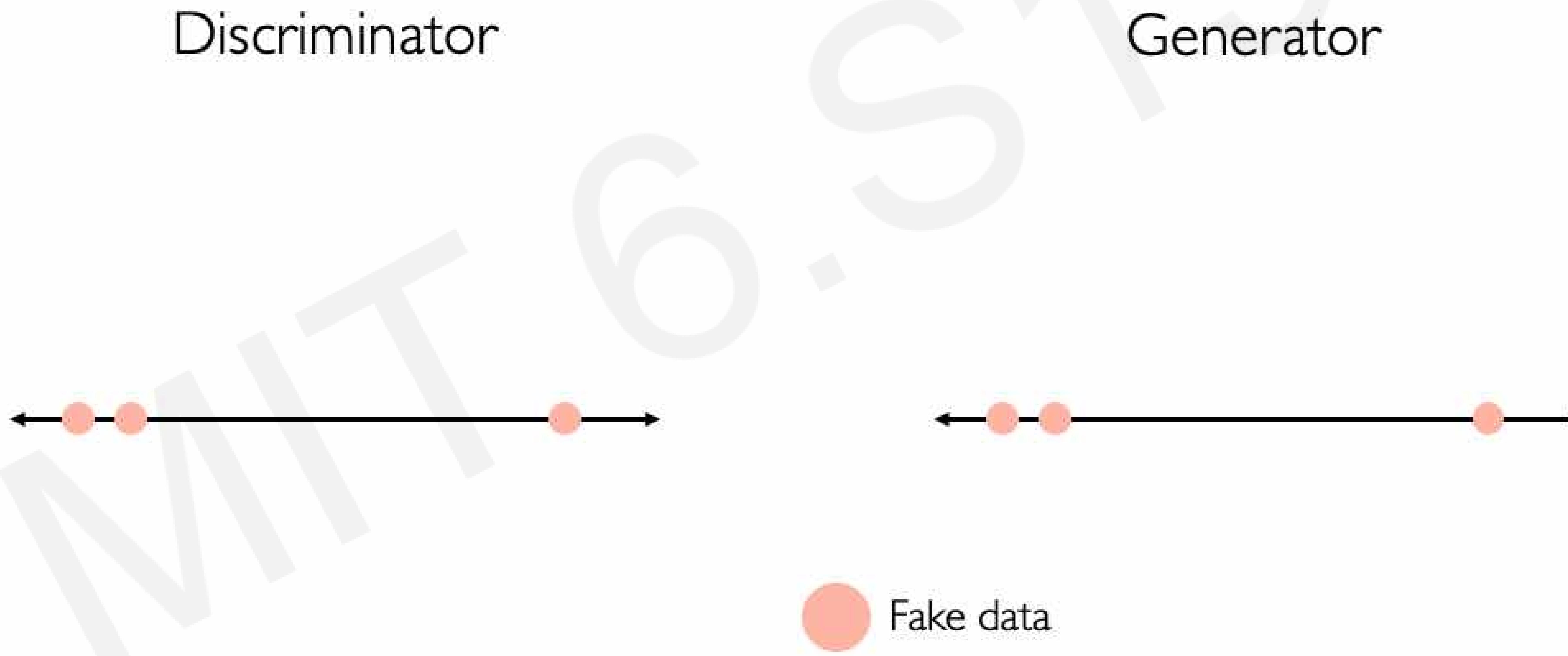
Generator starts from noise to try to create an imitation of the data.

Generator



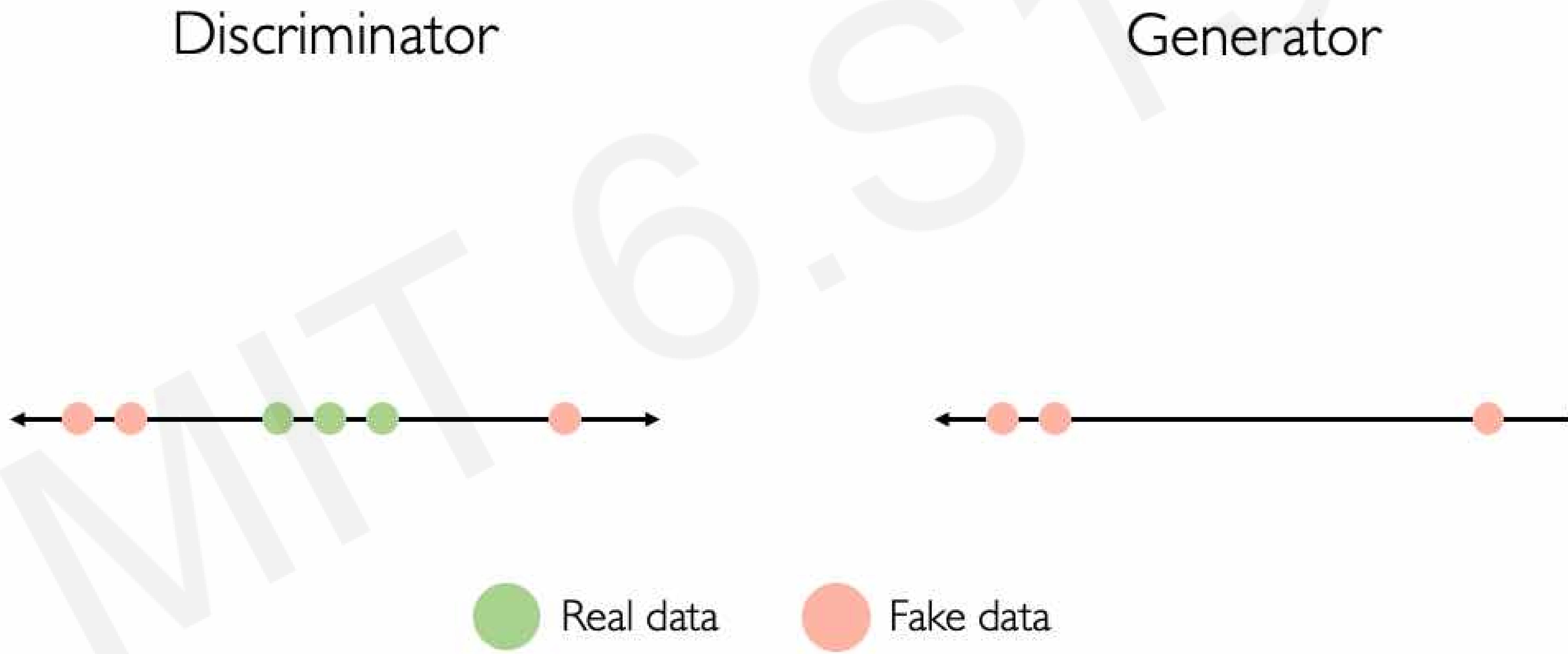
Intuition behind GANs

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



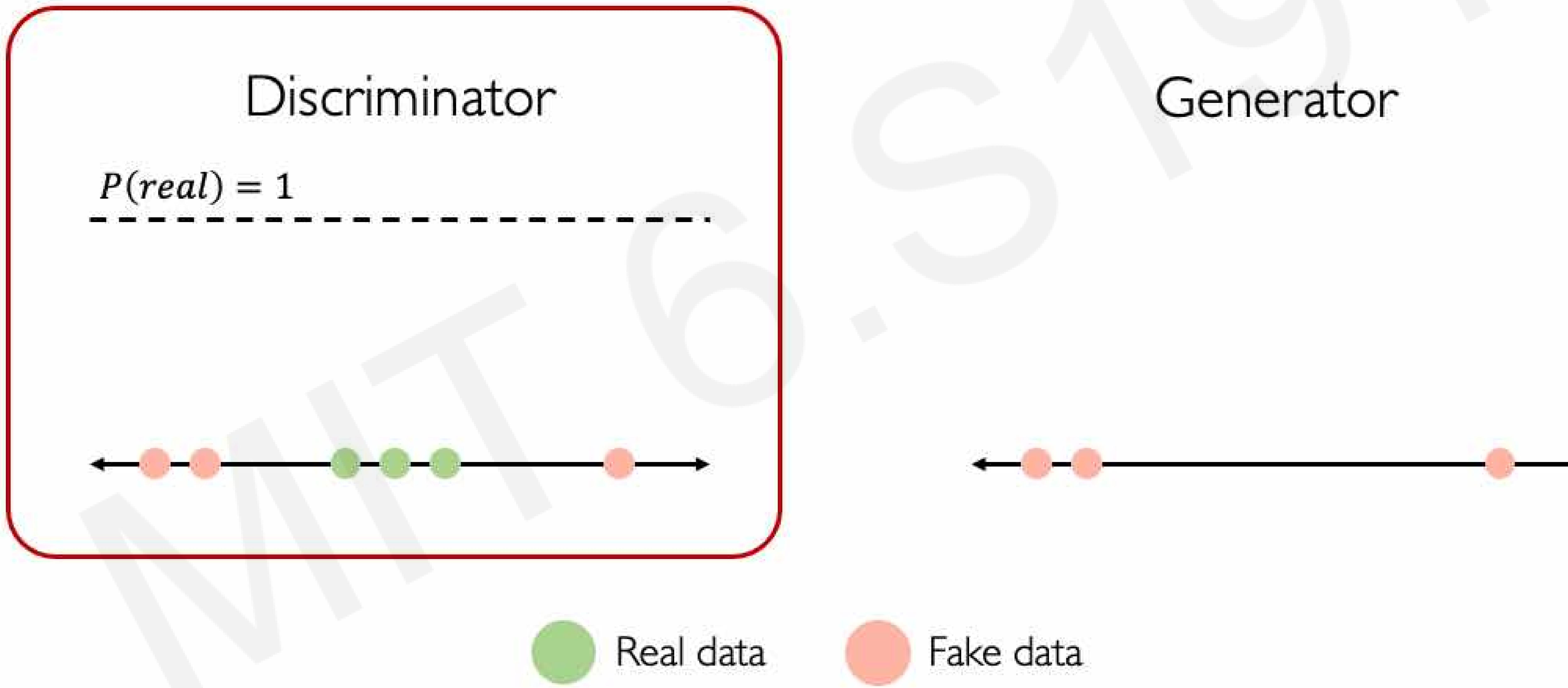
Intuition behind GANs

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



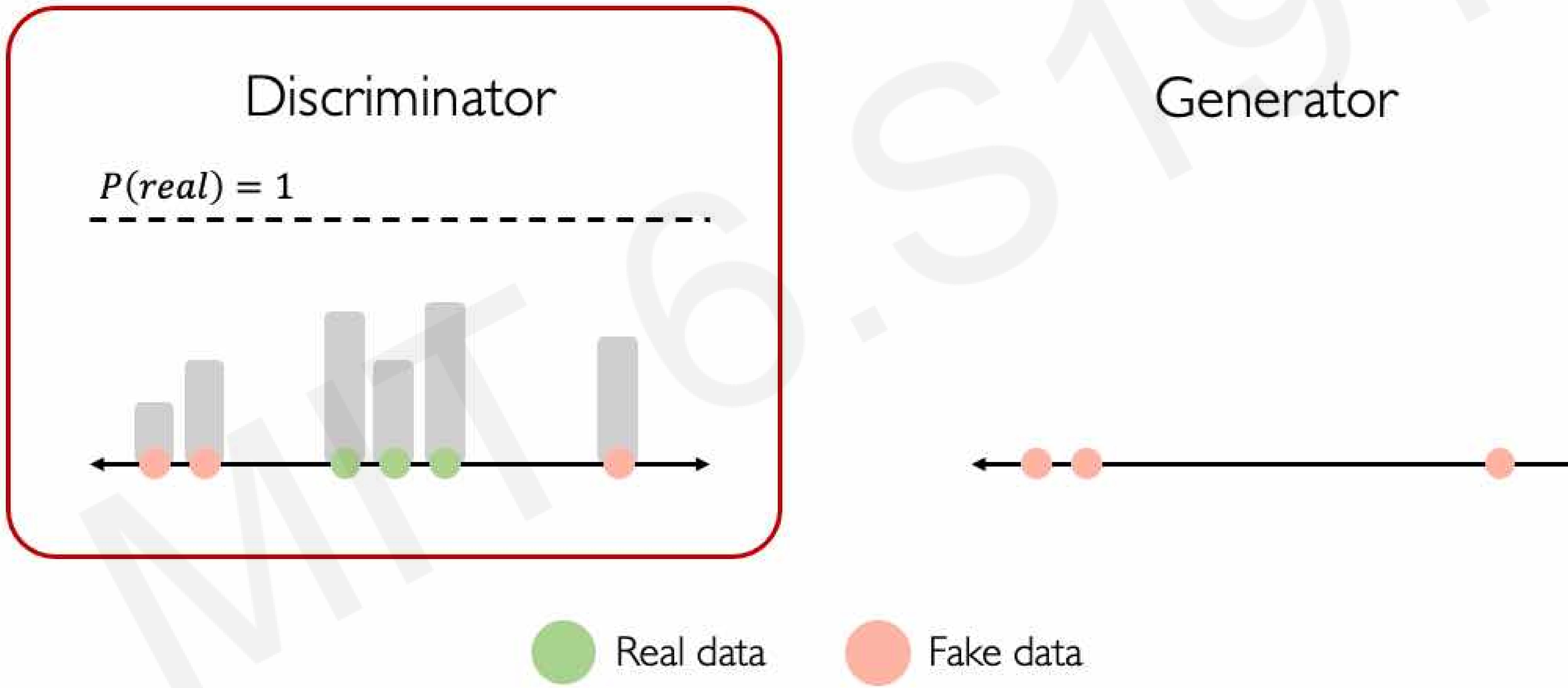
Intuition behind GANs

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



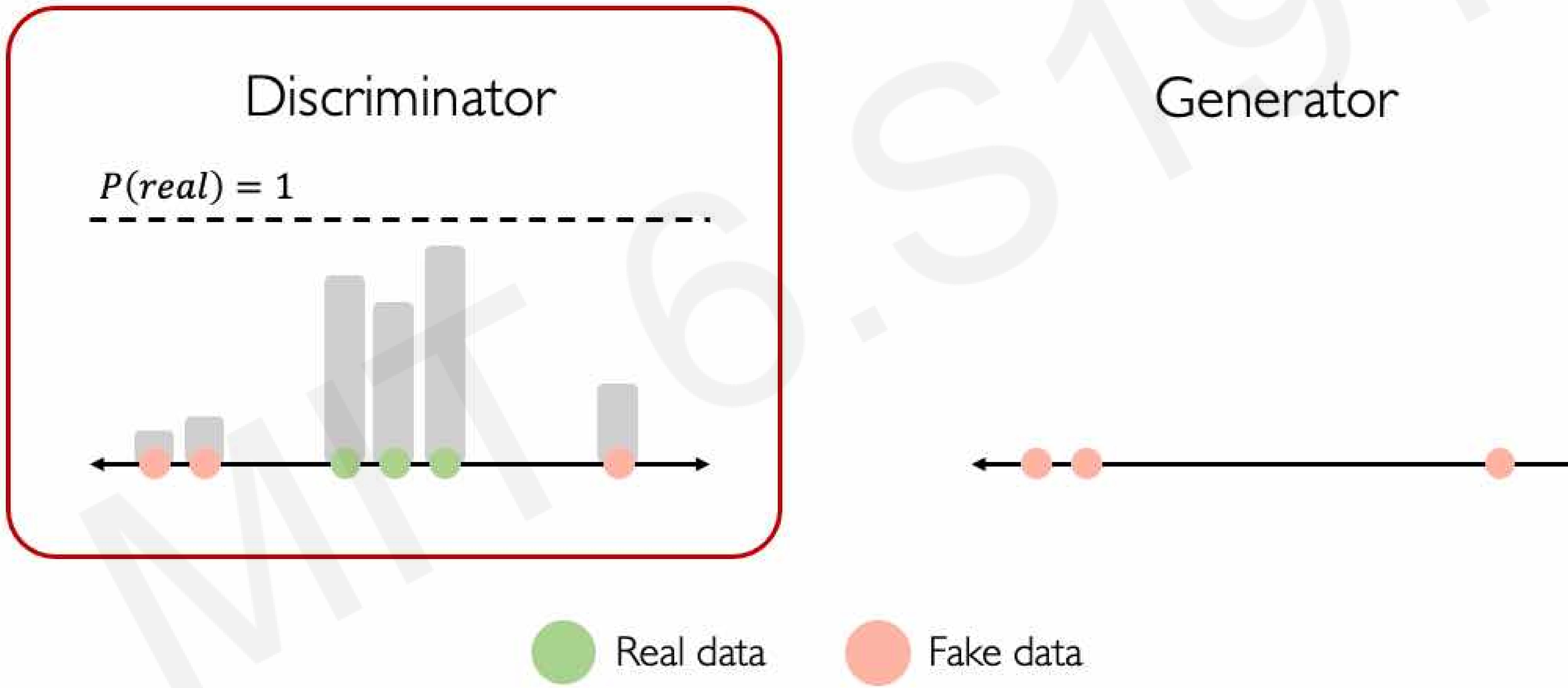
Intuition behind GANs

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



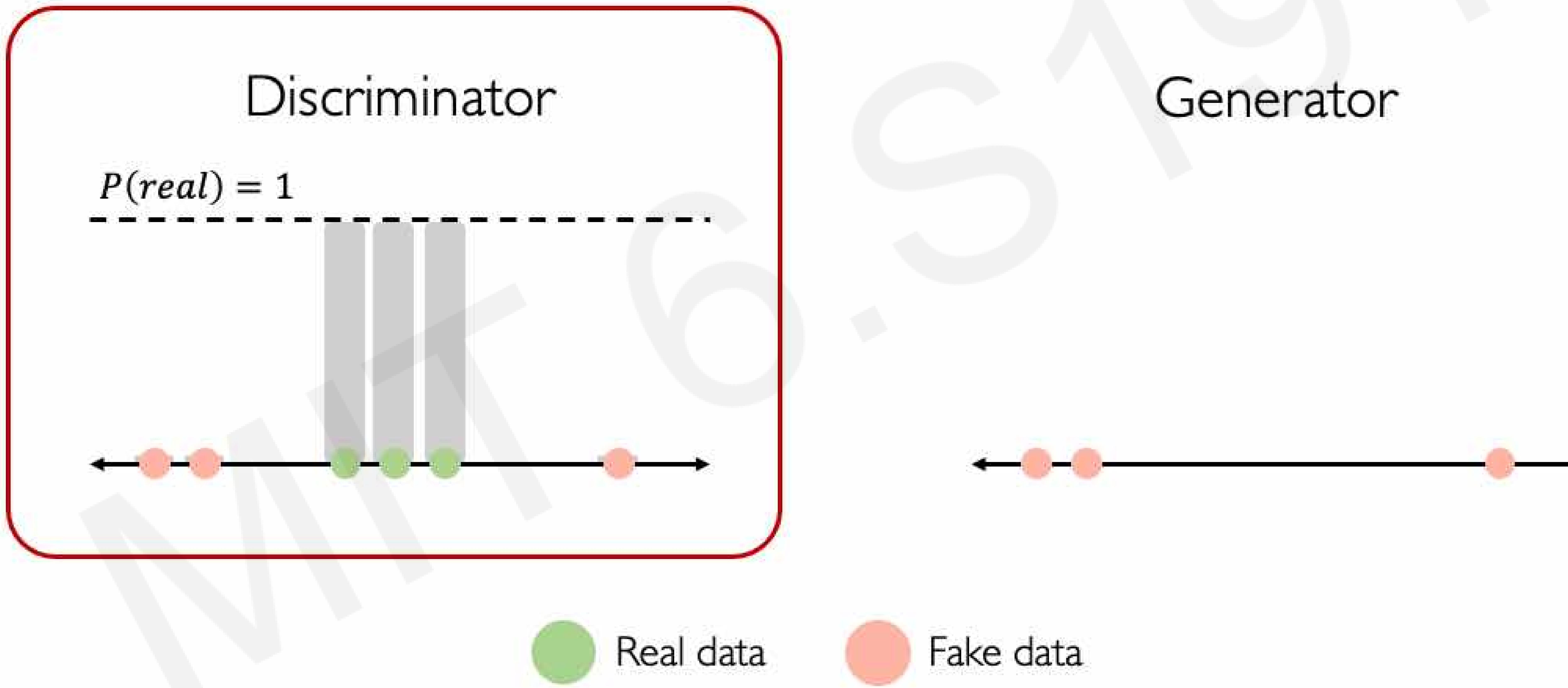
Intuition behind GANs

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



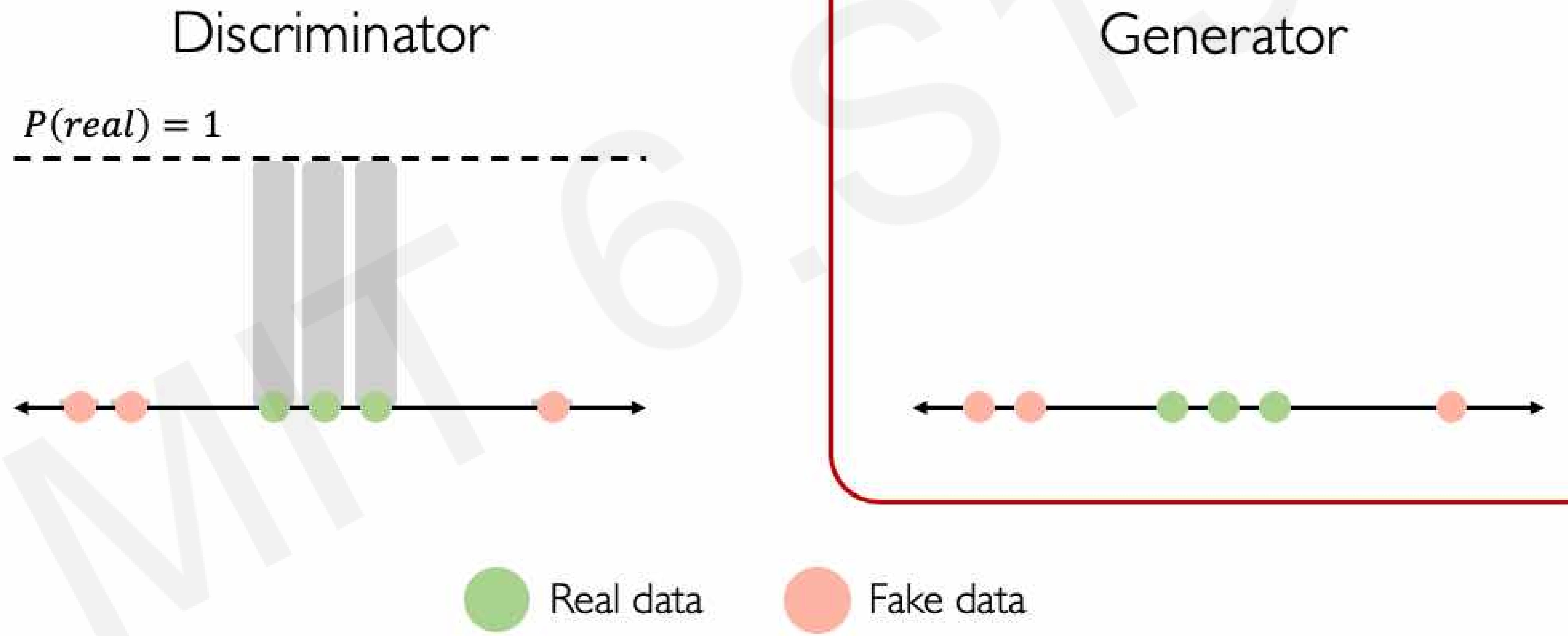
Intuition behind GANs

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



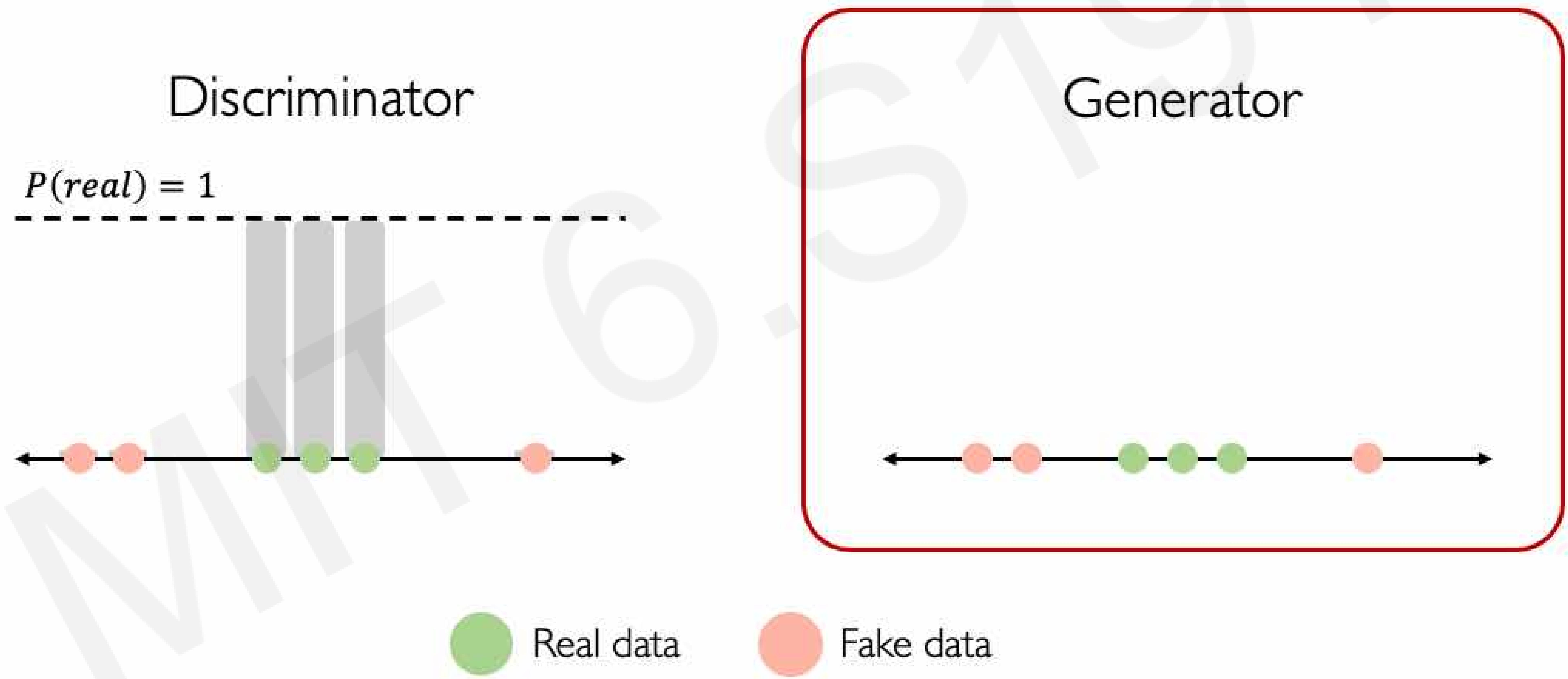
Intuition behind GANs

Generator tries to improve its imitation of the data.



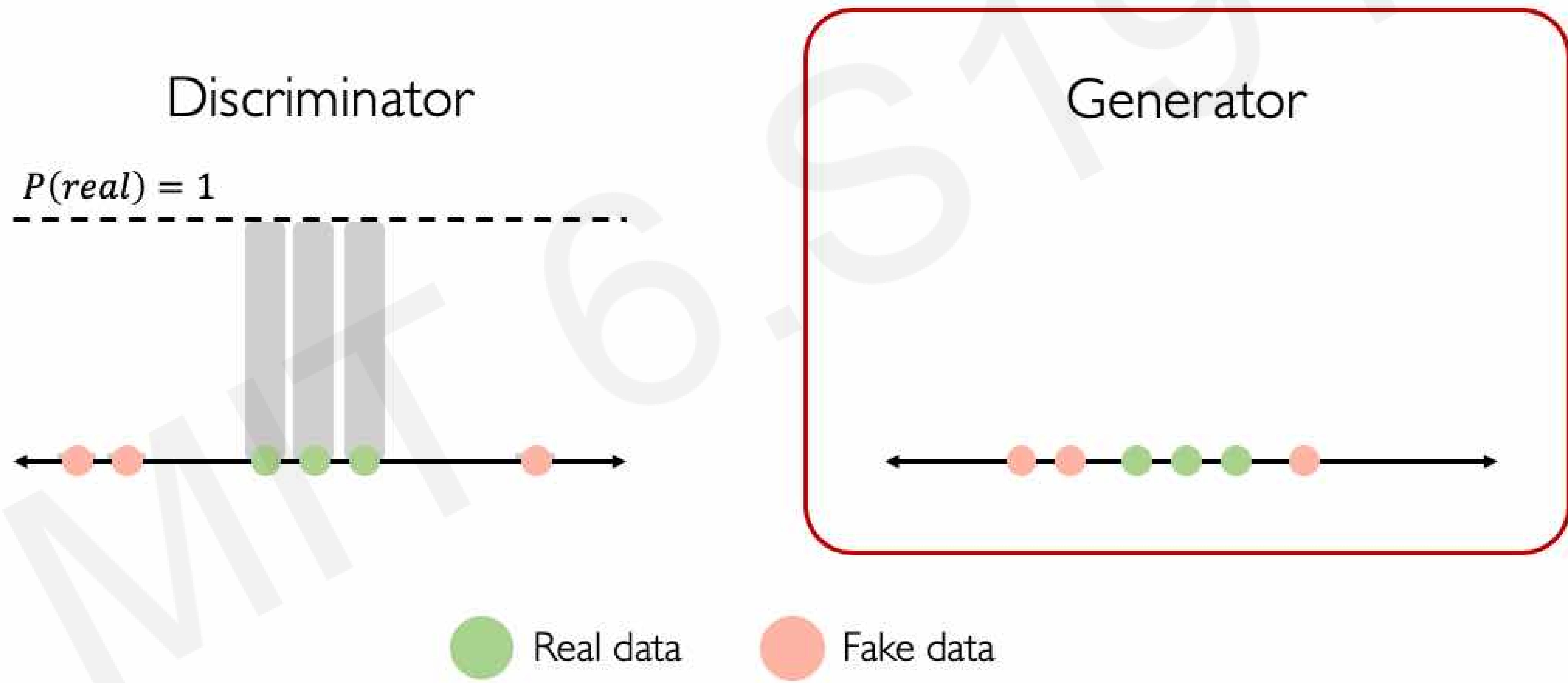
Intuition behind GANs

Generator tries to improve its imitation of the data.



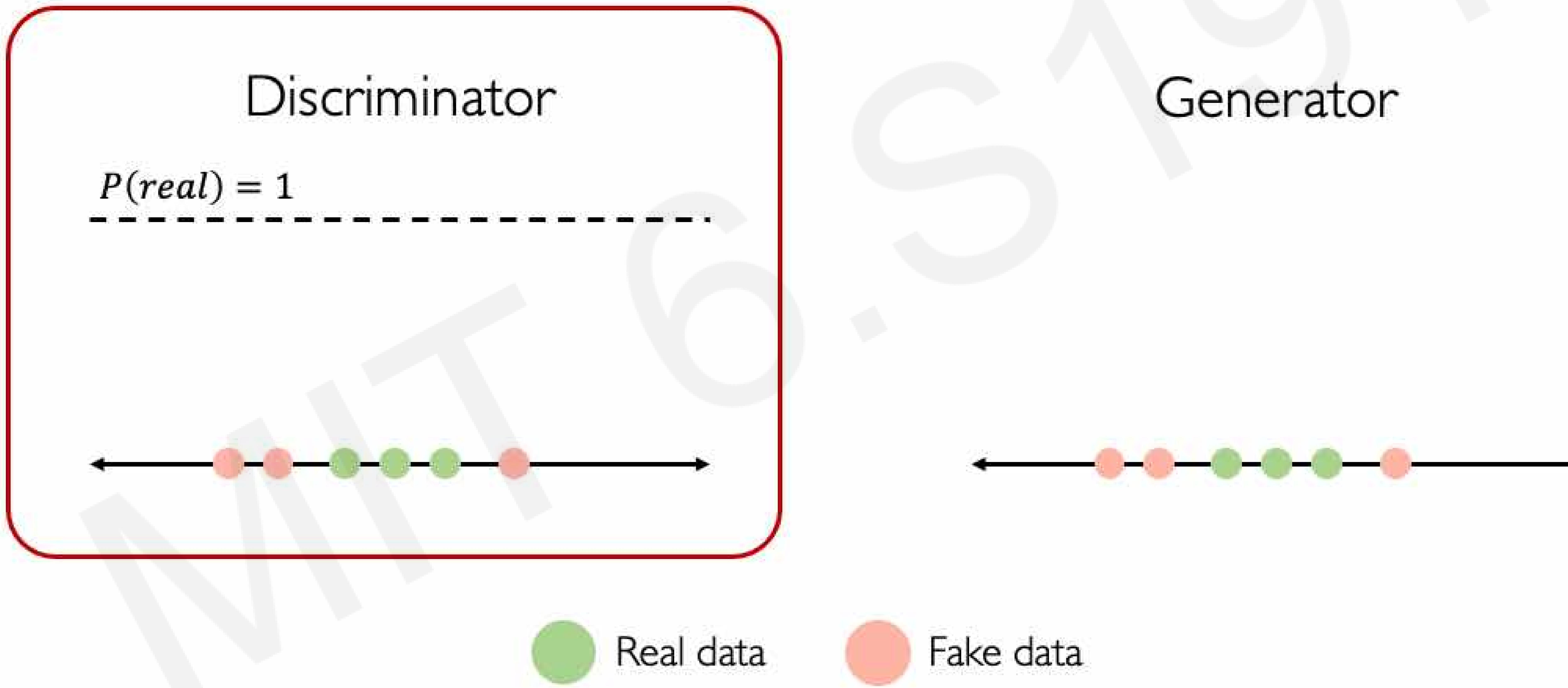
Intuition behind GANs

Generator tries to improve its imitation of the data.



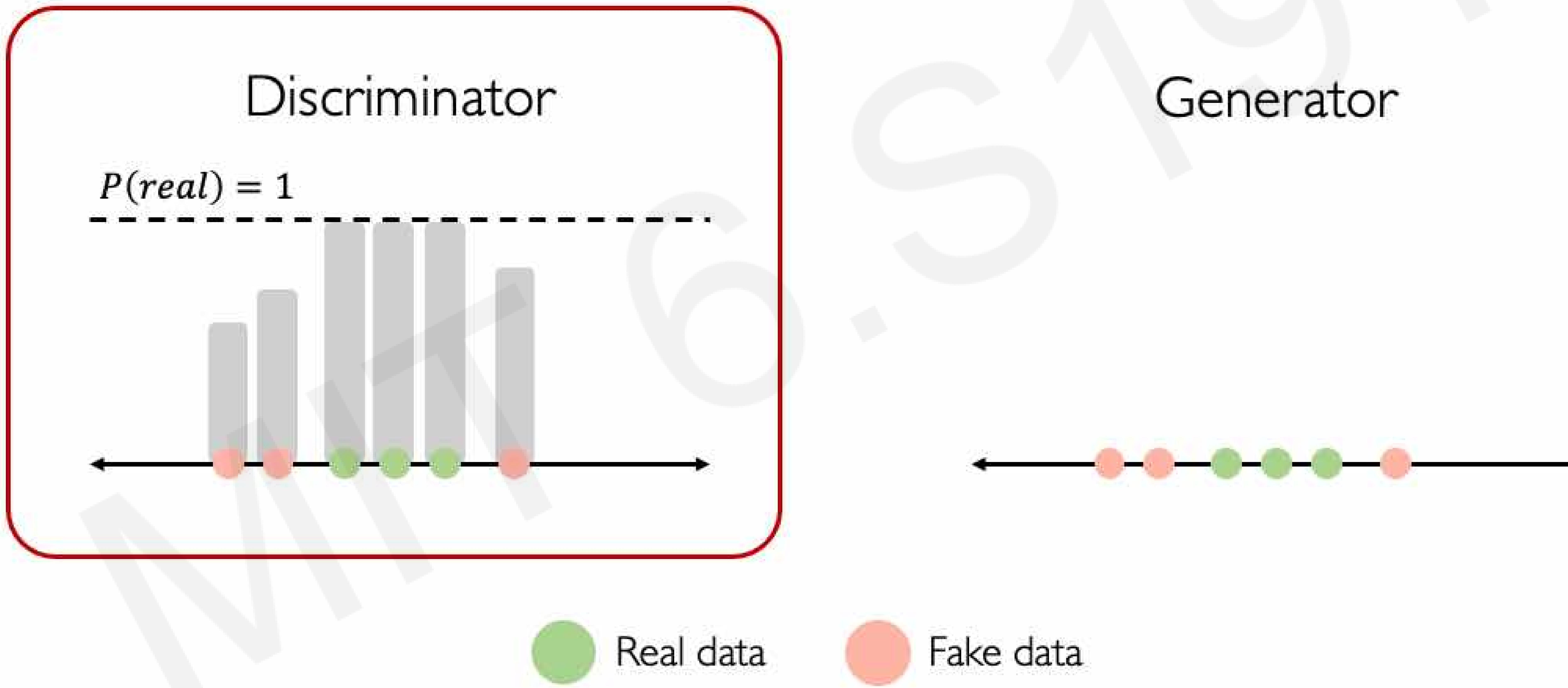
Intuition behind GANs

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



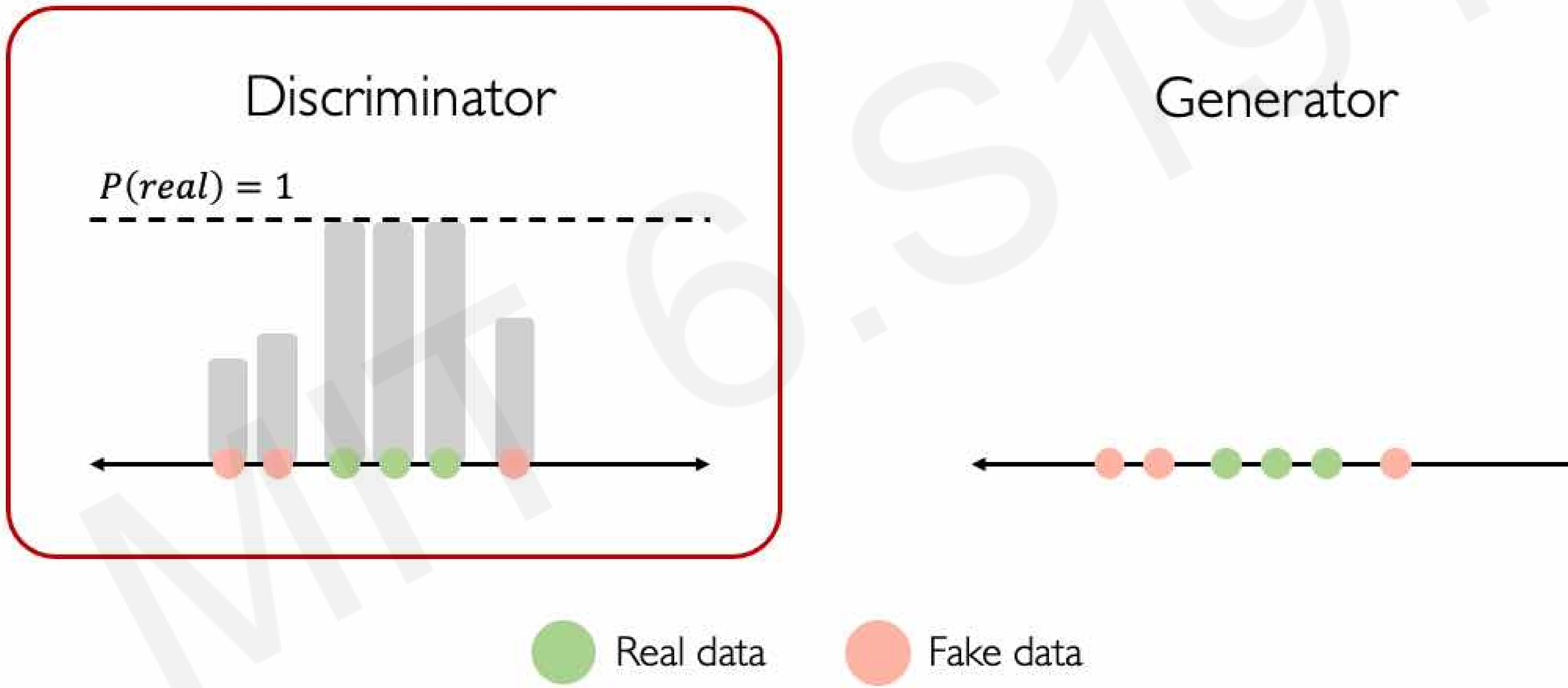
Intuition behind GANs

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



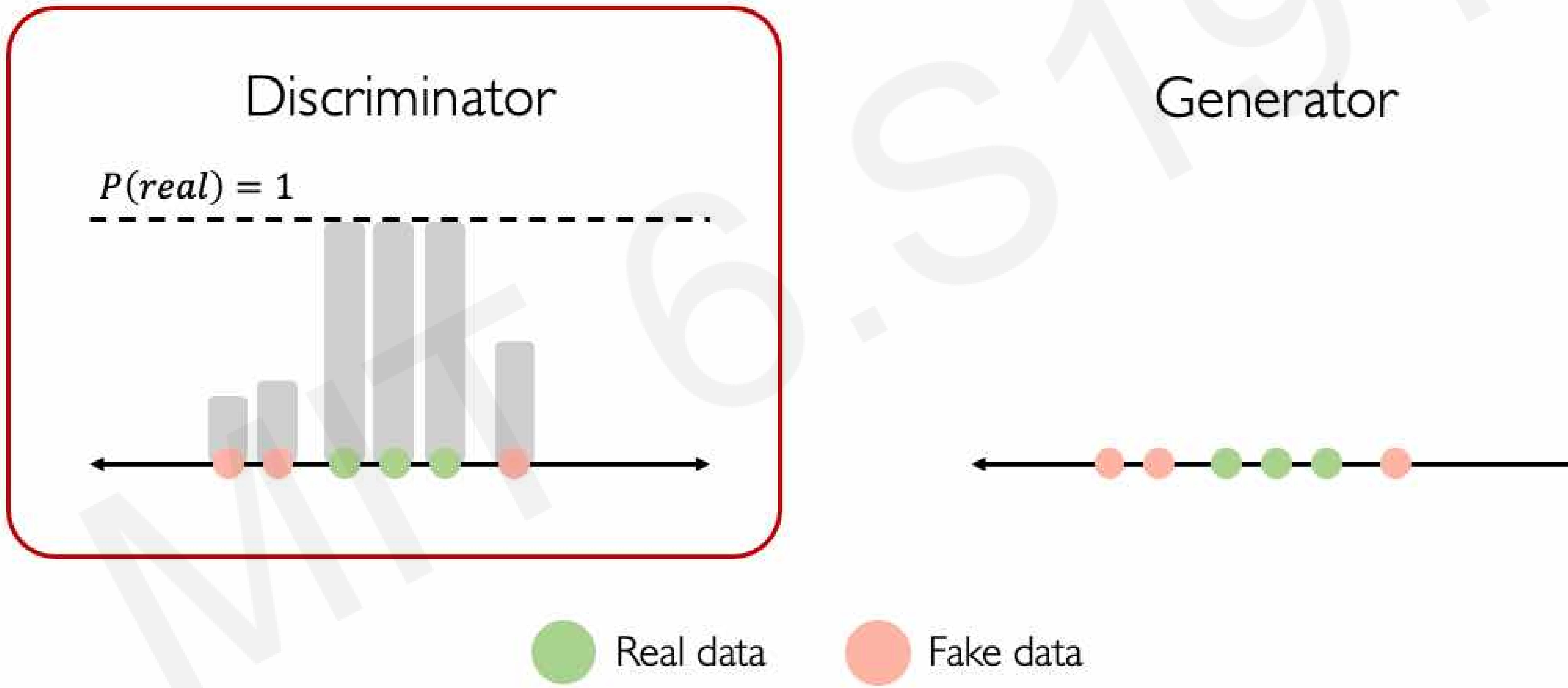
Intuition behind GANs

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



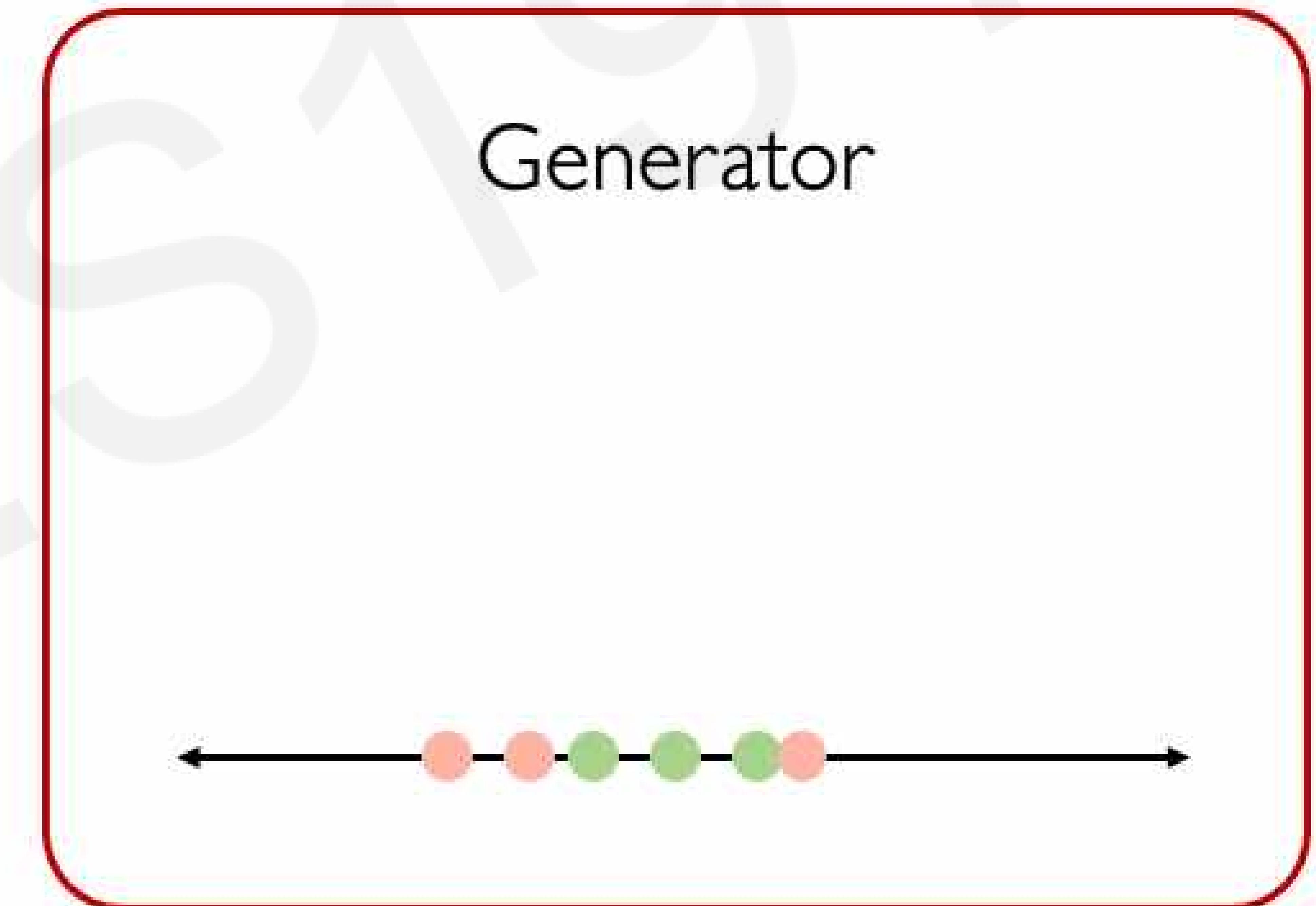
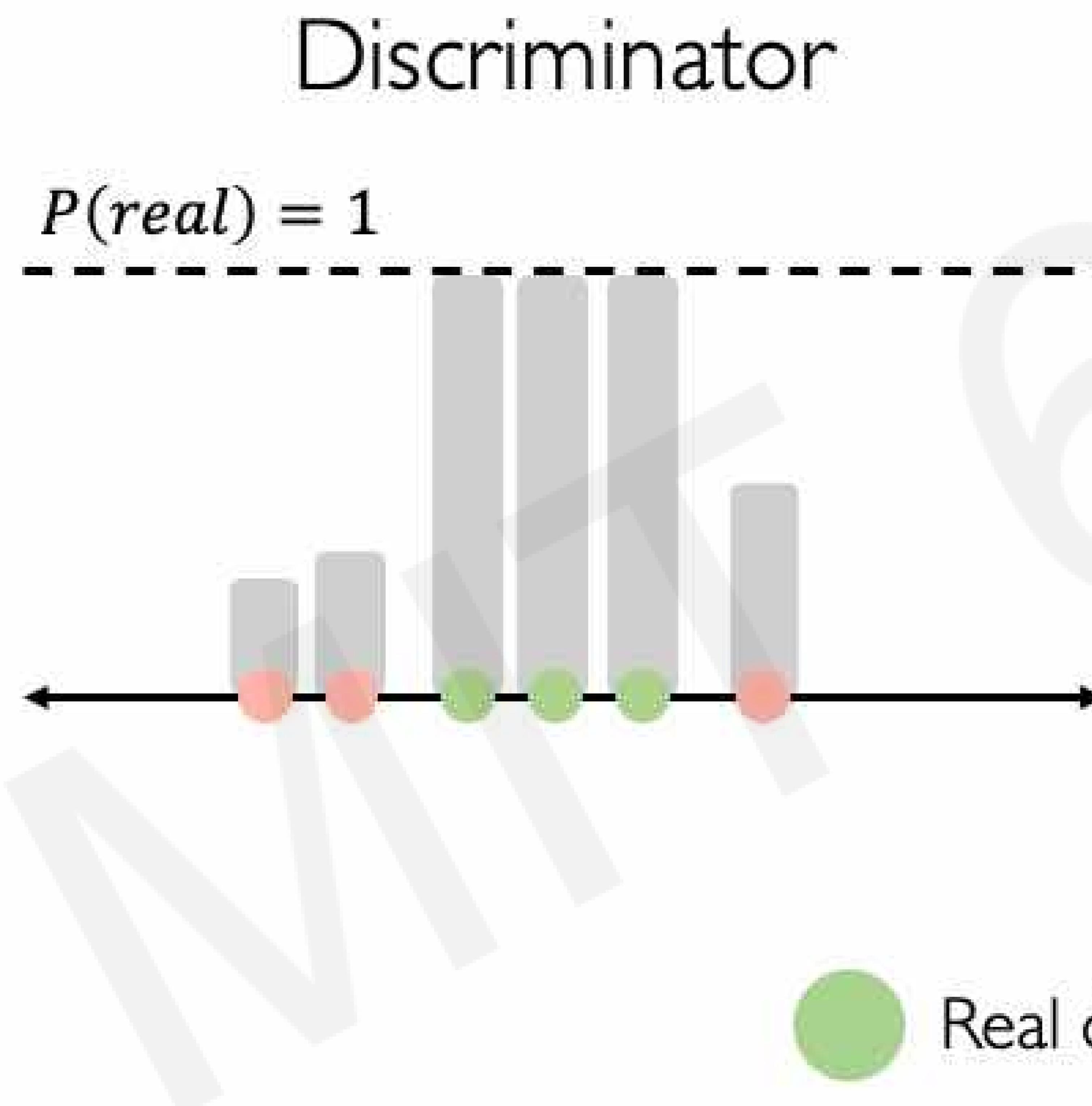
Intuition behind GANs

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



Intuition behind GANs

Generator tries to improve its imitation of the data.

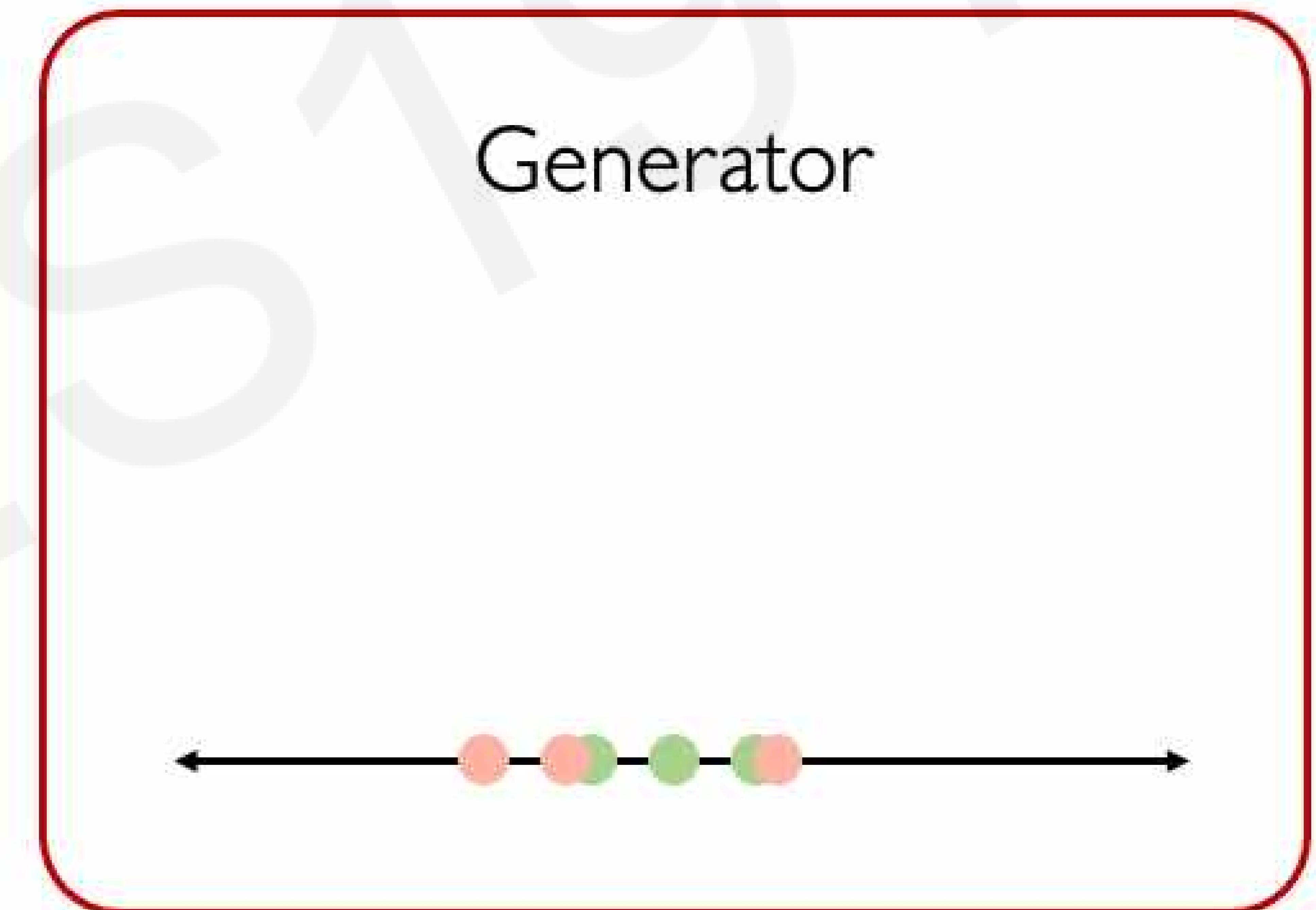
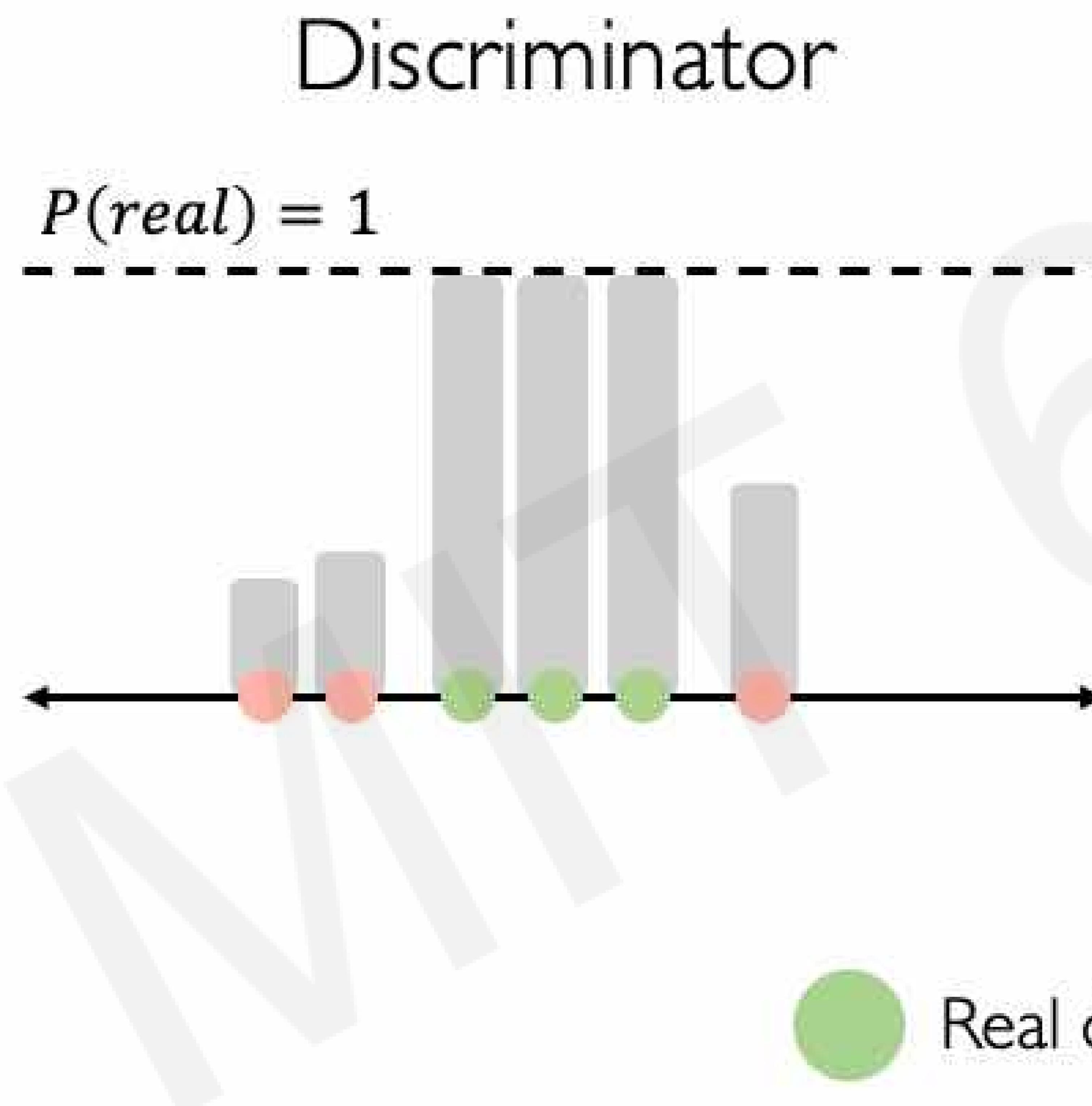


Real data

Fake data

Intuition behind GANs

Generator tries to improve its imitation of the data.

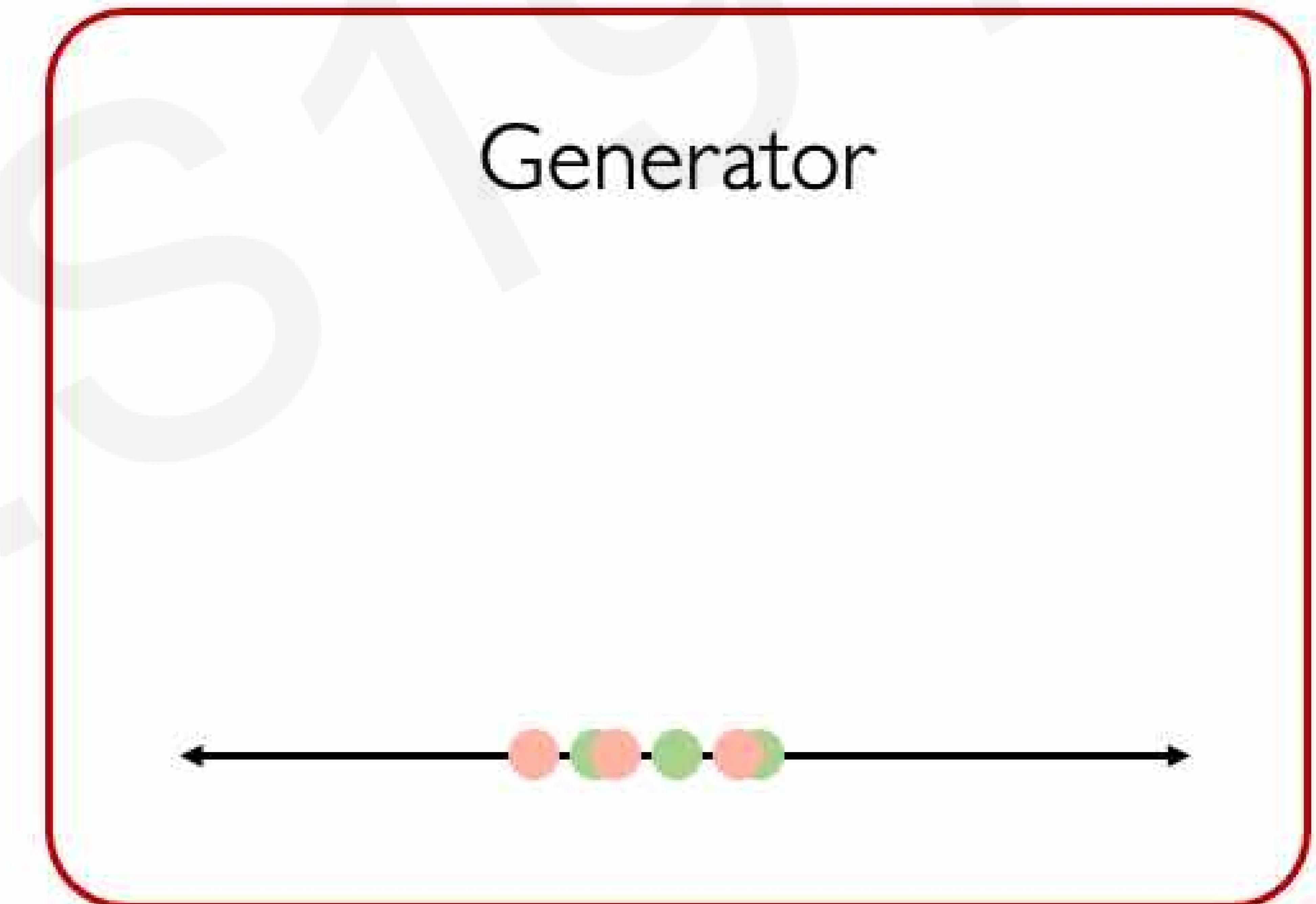
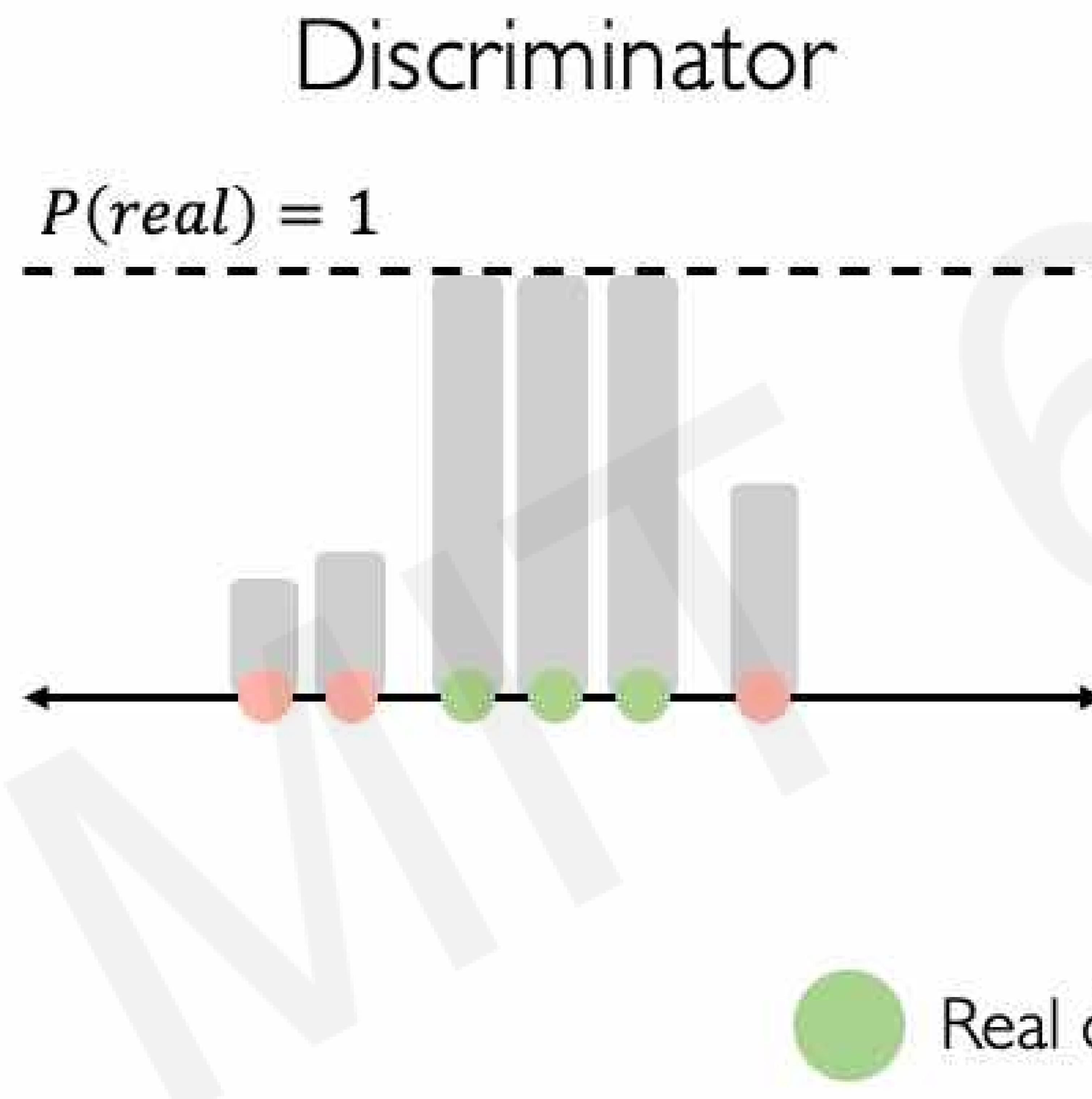


Real data

Fake data

Intuition behind GANs

Generator tries to improve its imitation of the data.



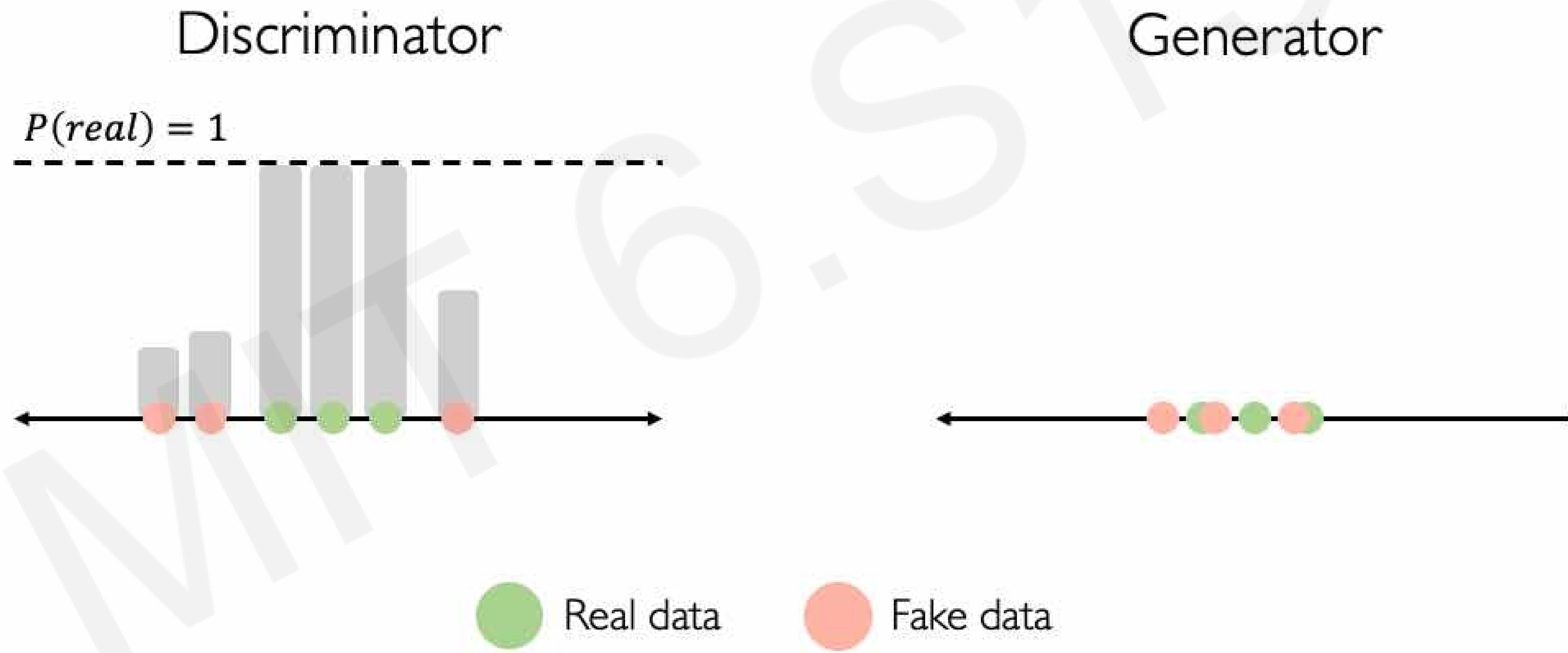
Real data

Fake data

Intuition behind GANs

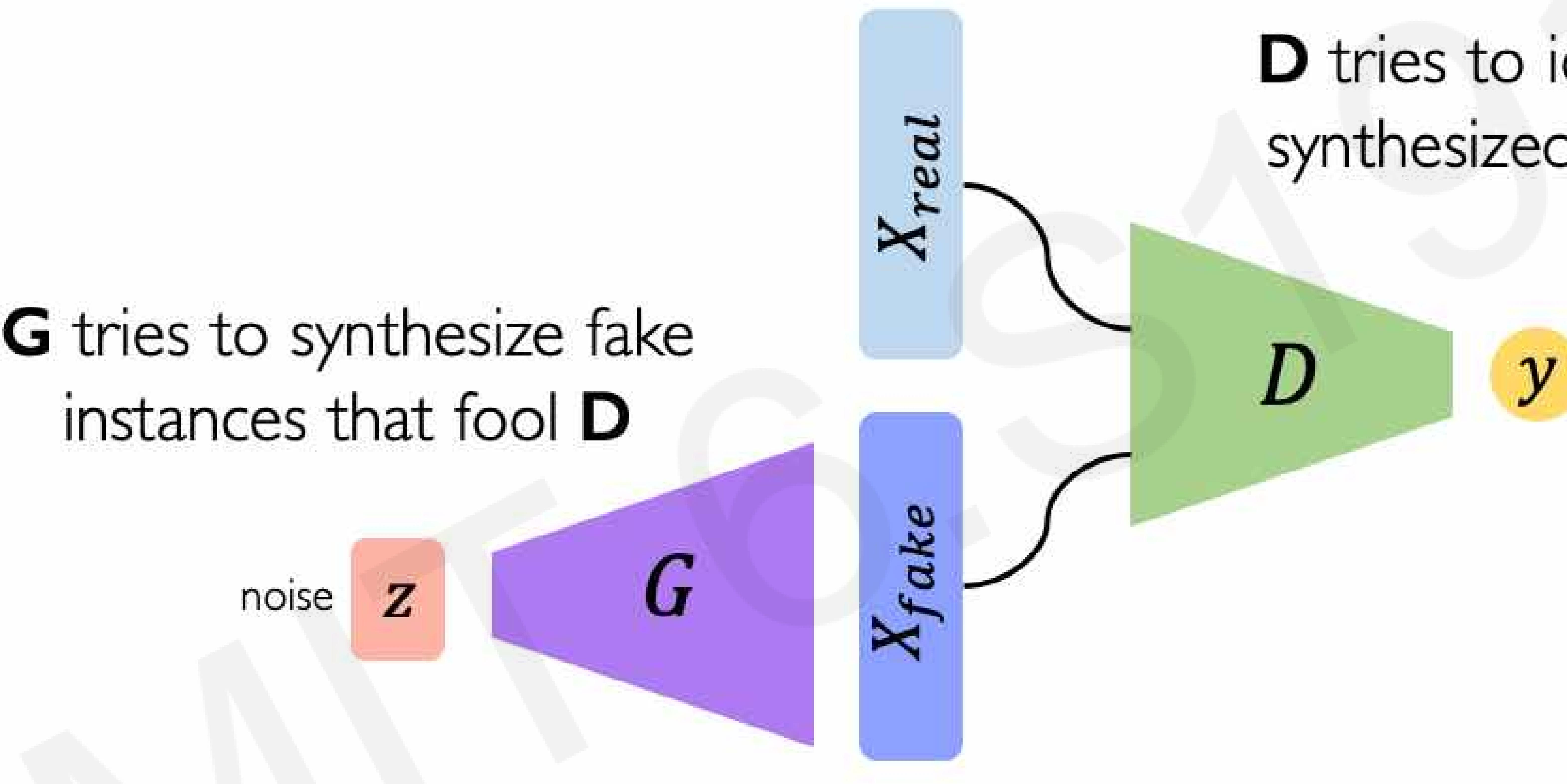
Discriminator tries to identify real data from fakes created by the generator.

Generator tries to create imitations of data to trick the discriminator.



Training GANs

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

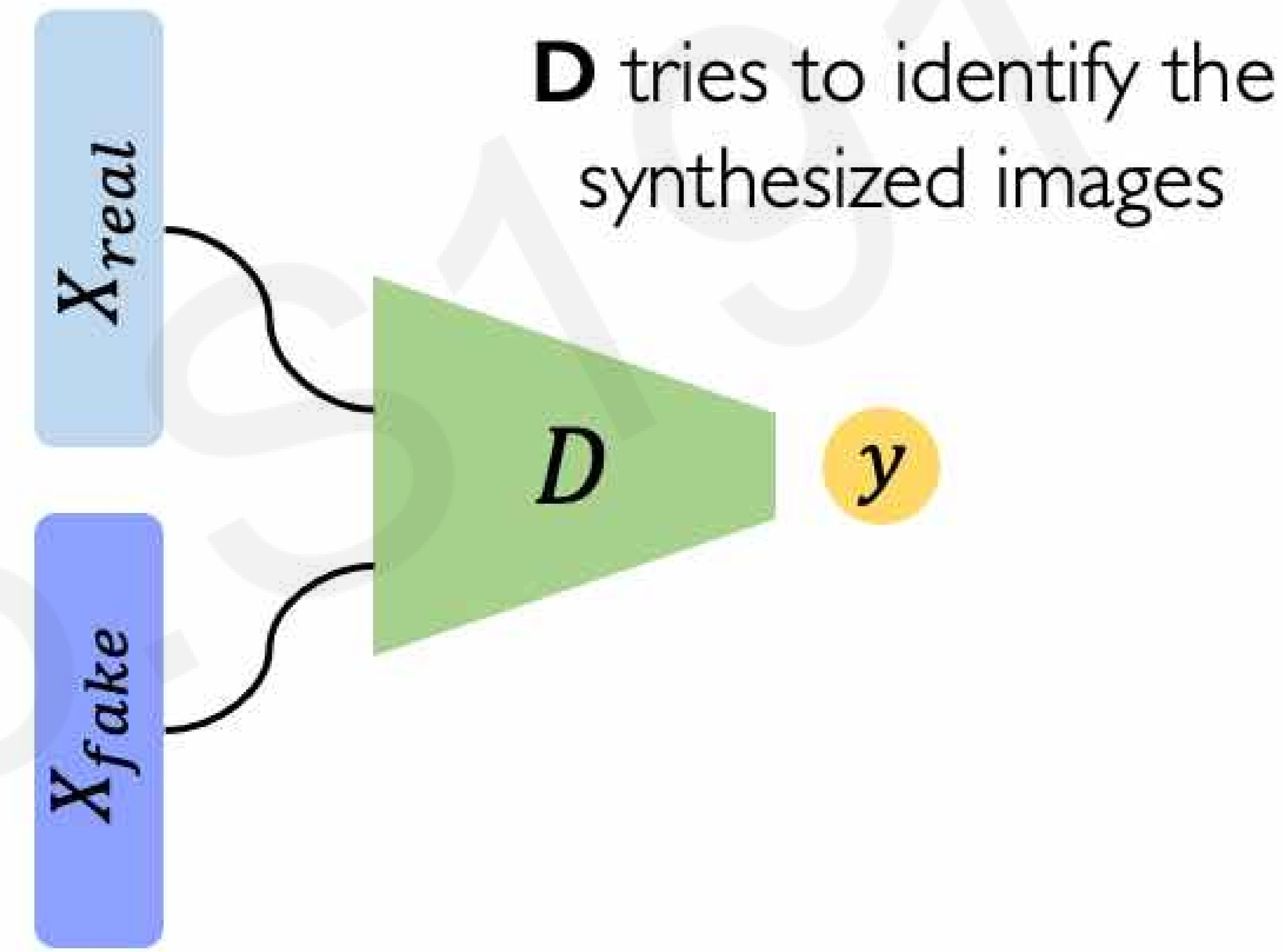


D tries to identify the synthesized instances

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

Global optimum: **G** reproduces the true data distribution

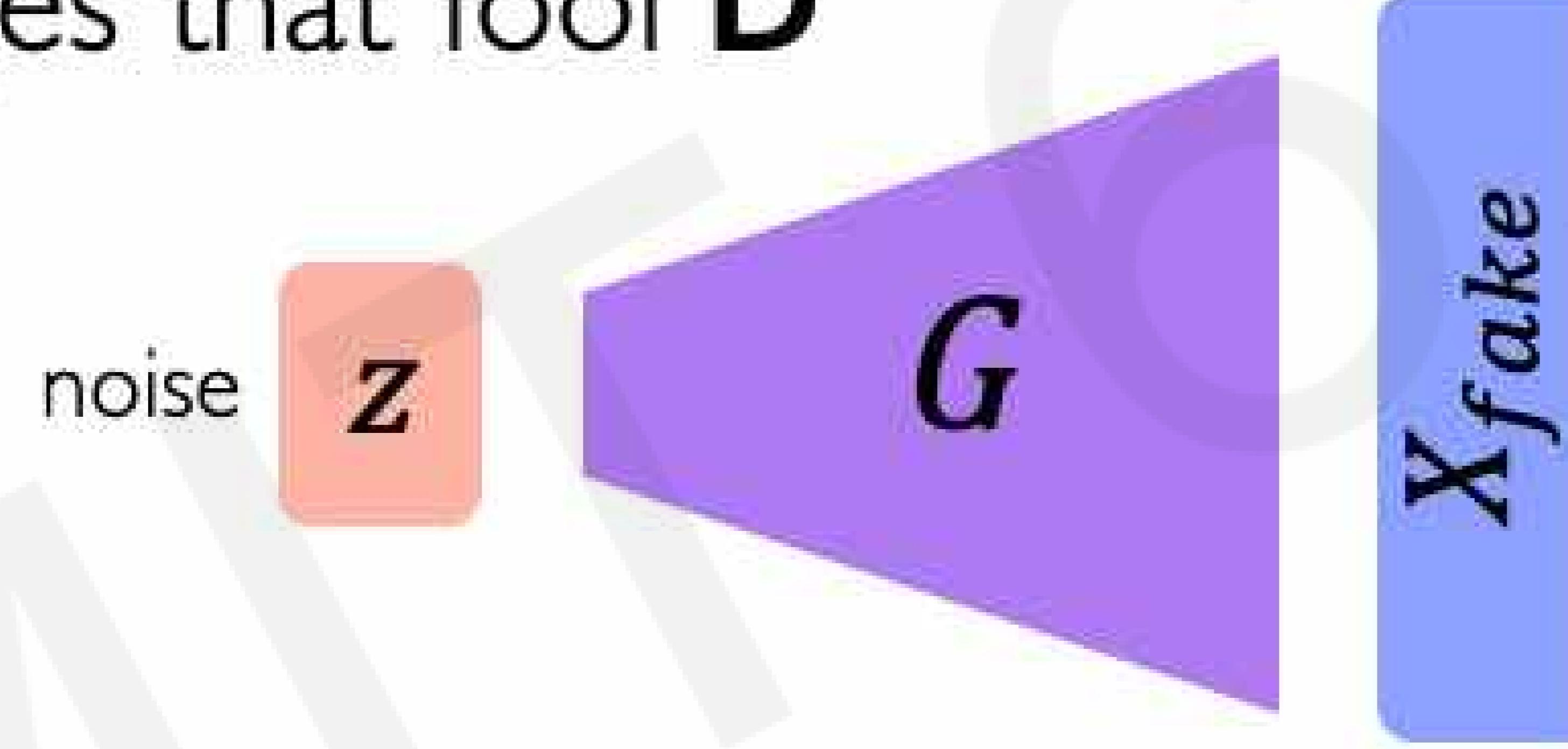
Training GANs: loss function



$$\arg \max_D \mathbb{E}_{\mathbf{z}, \mathbf{x}} [\underbrace{\log D(G(\mathbf{z}))}_{\text{Fake}} + \underbrace{\log (1 - D(\mathbf{x}))}_{\text{Real}}]$$

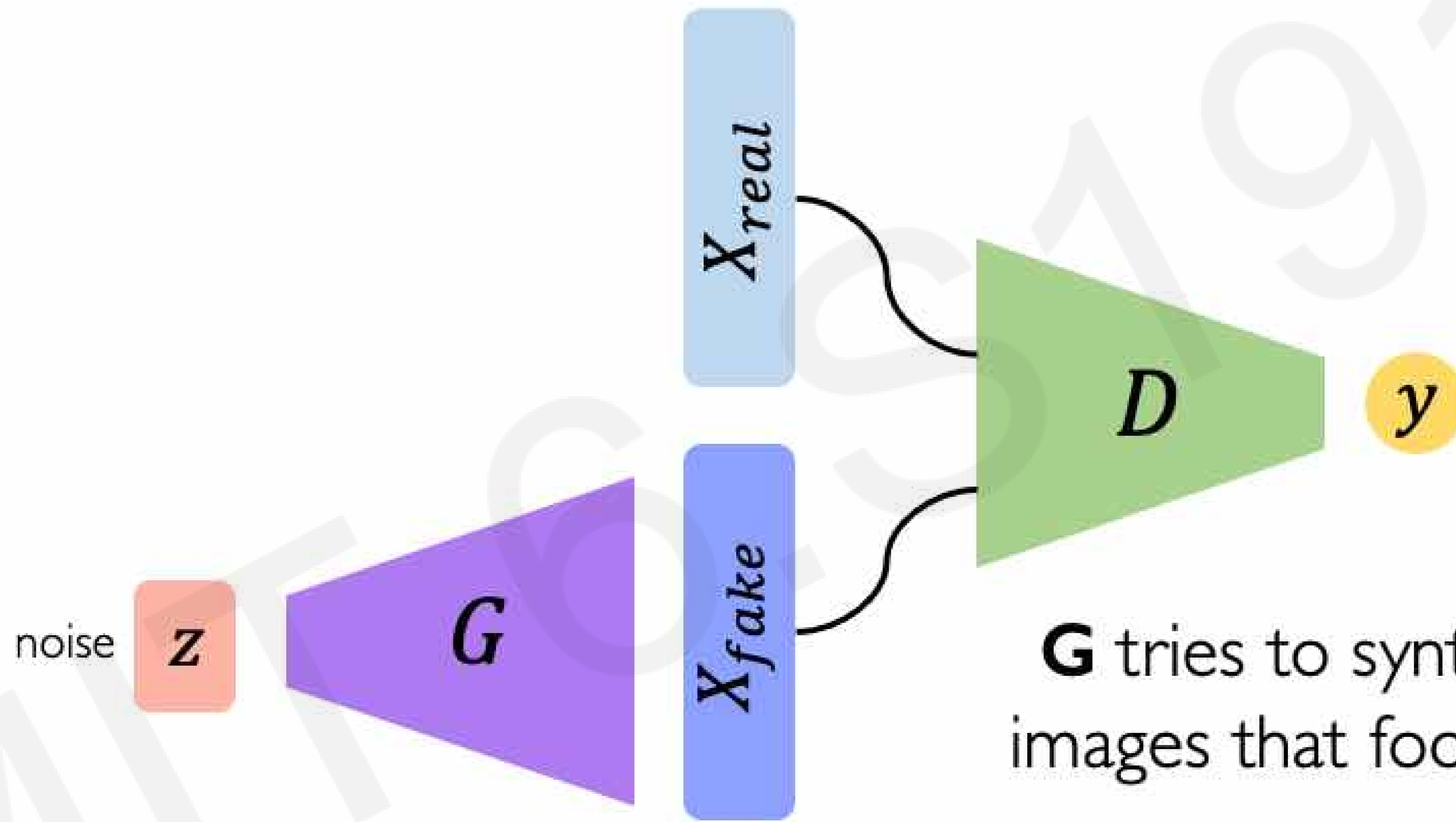
Training GANs: loss function

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



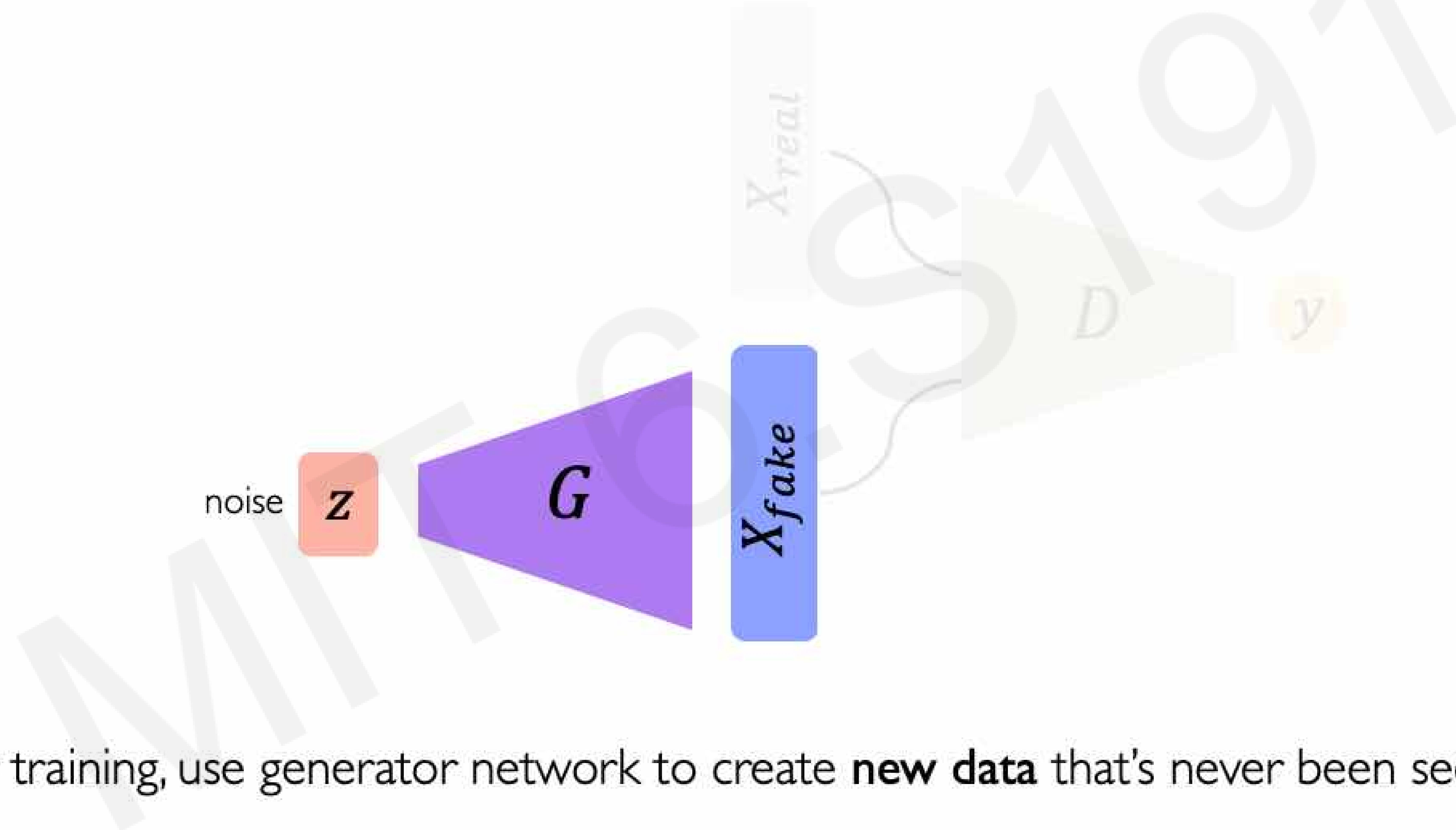
$$\arg \min_G \mathbb{E}_{\mathbf{z}, \mathbf{x}} [\log D(G(\mathbf{z})) + \log (1 - D(\mathbf{x}))]$$

Training GANs: loss function

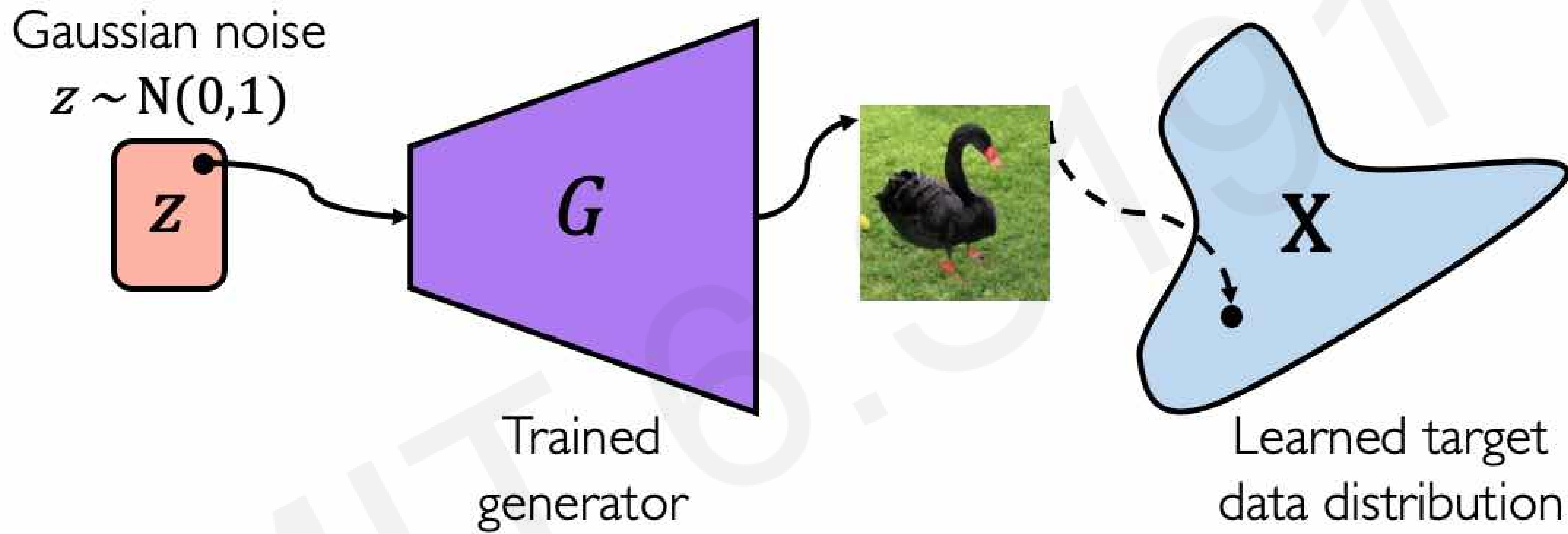


$$\arg \min_G \max_D \mathbb{E}_{\mathbf{z}, \mathbf{x}} [\log D(G(\mathbf{z})) + \log (1 - D(\mathbf{x}))]$$

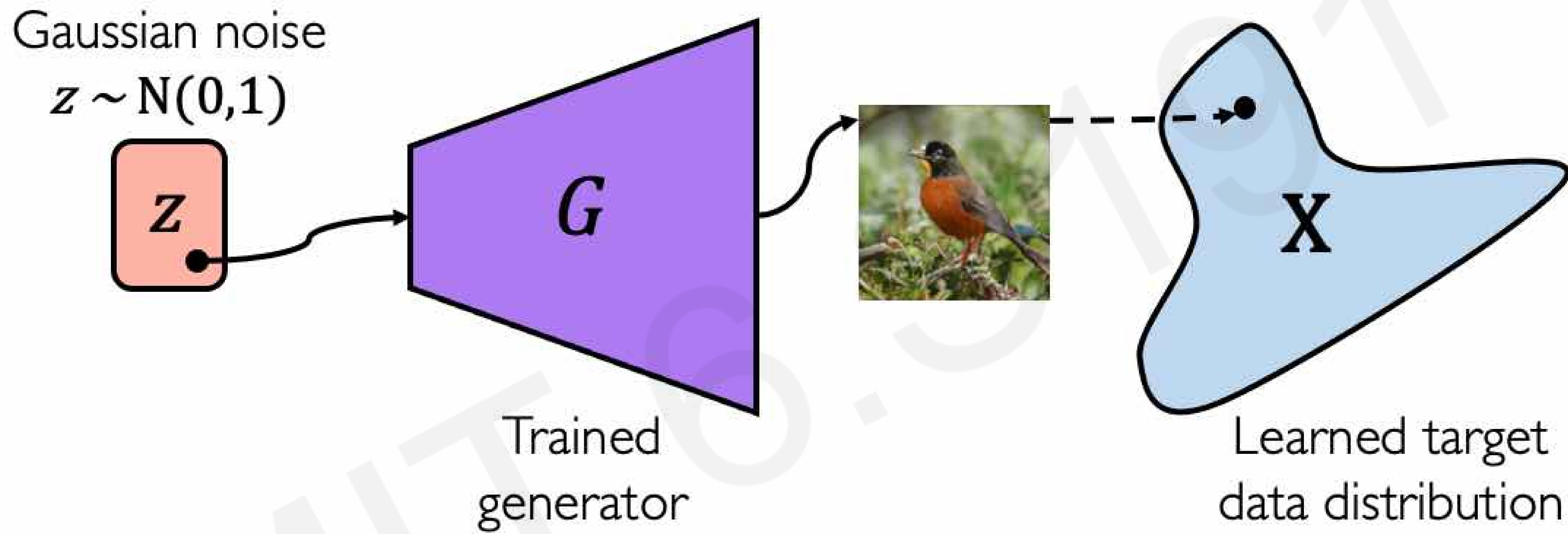
Generating new data with GANs



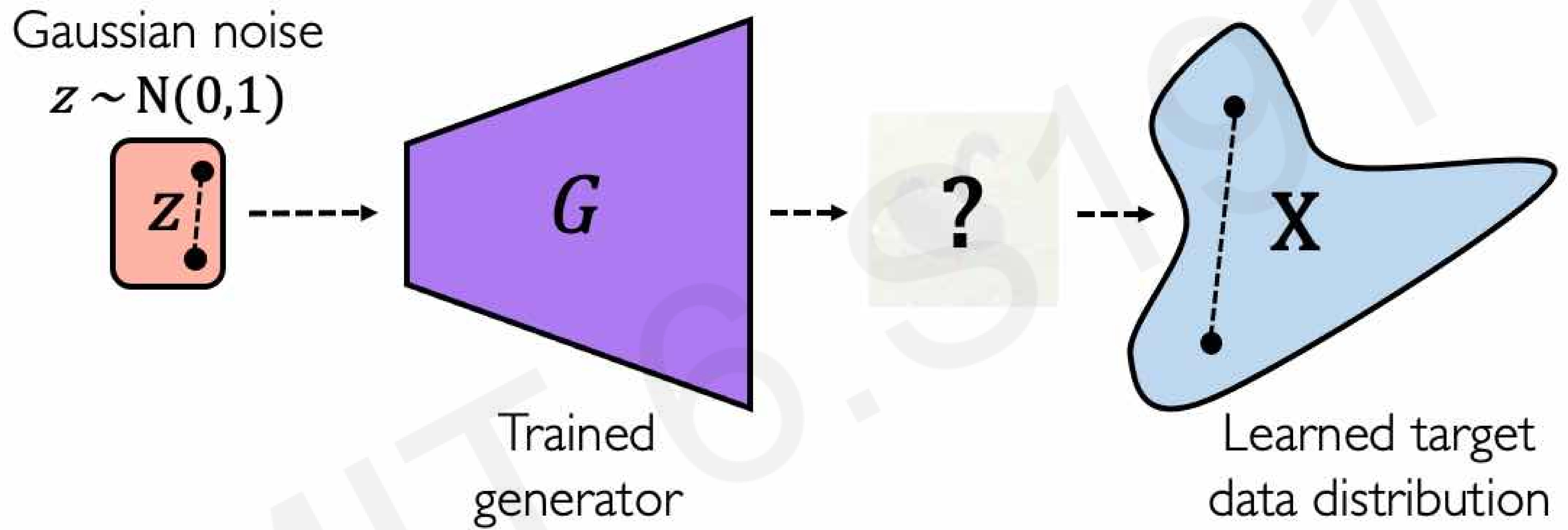
GANs are distribution transformers



GANs are distribution transformers



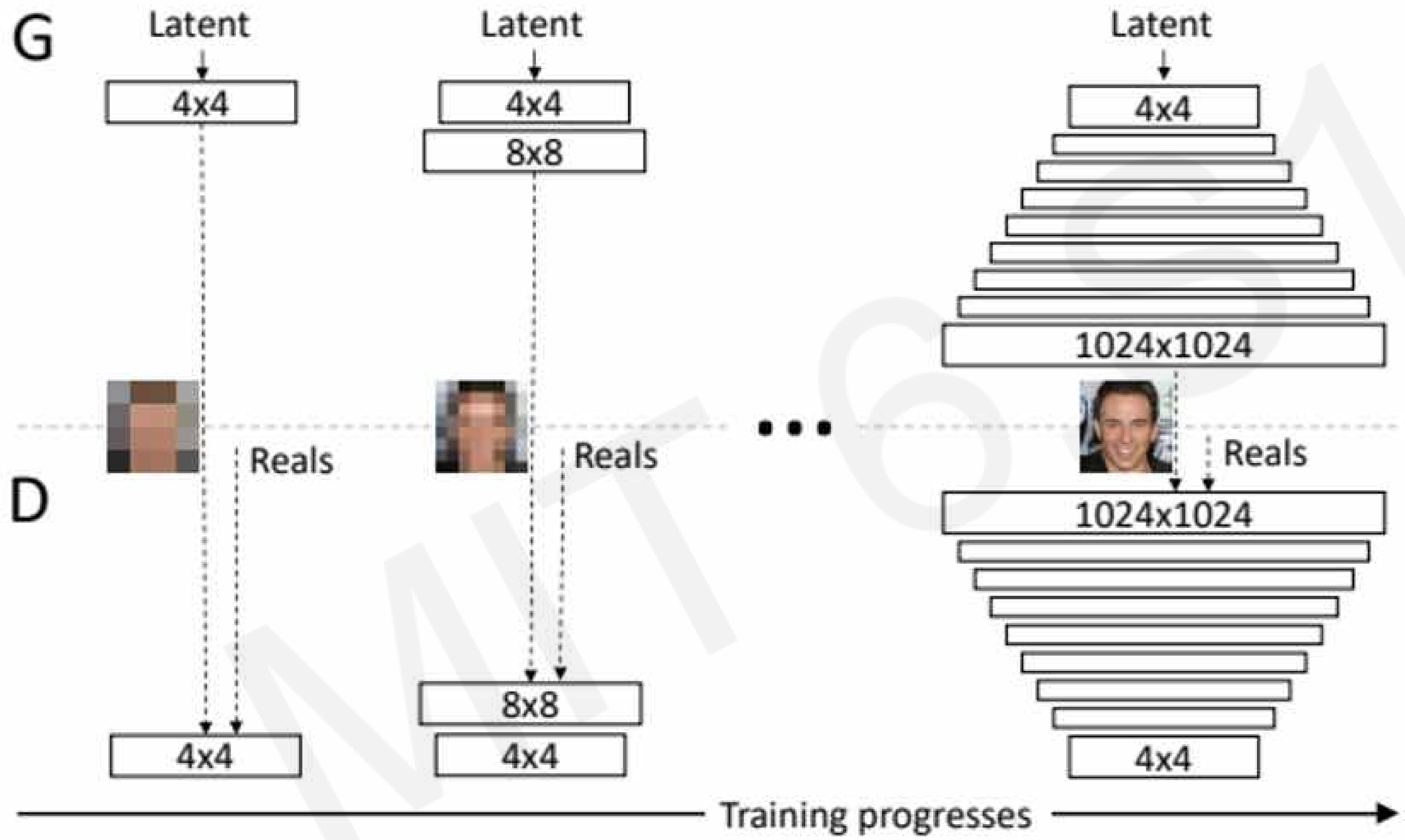
GANs are distribution transformers



GANs: Recent Advances



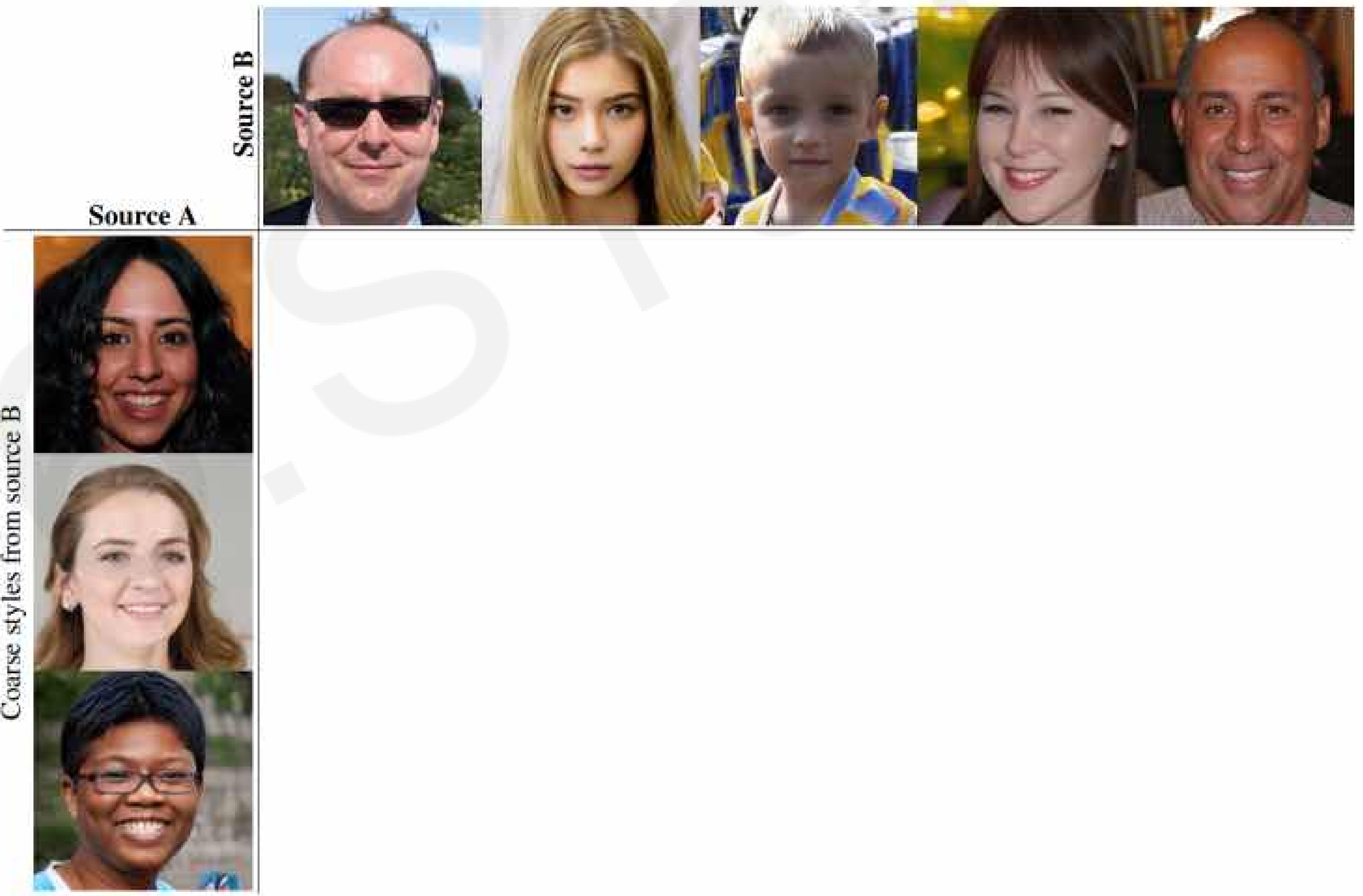
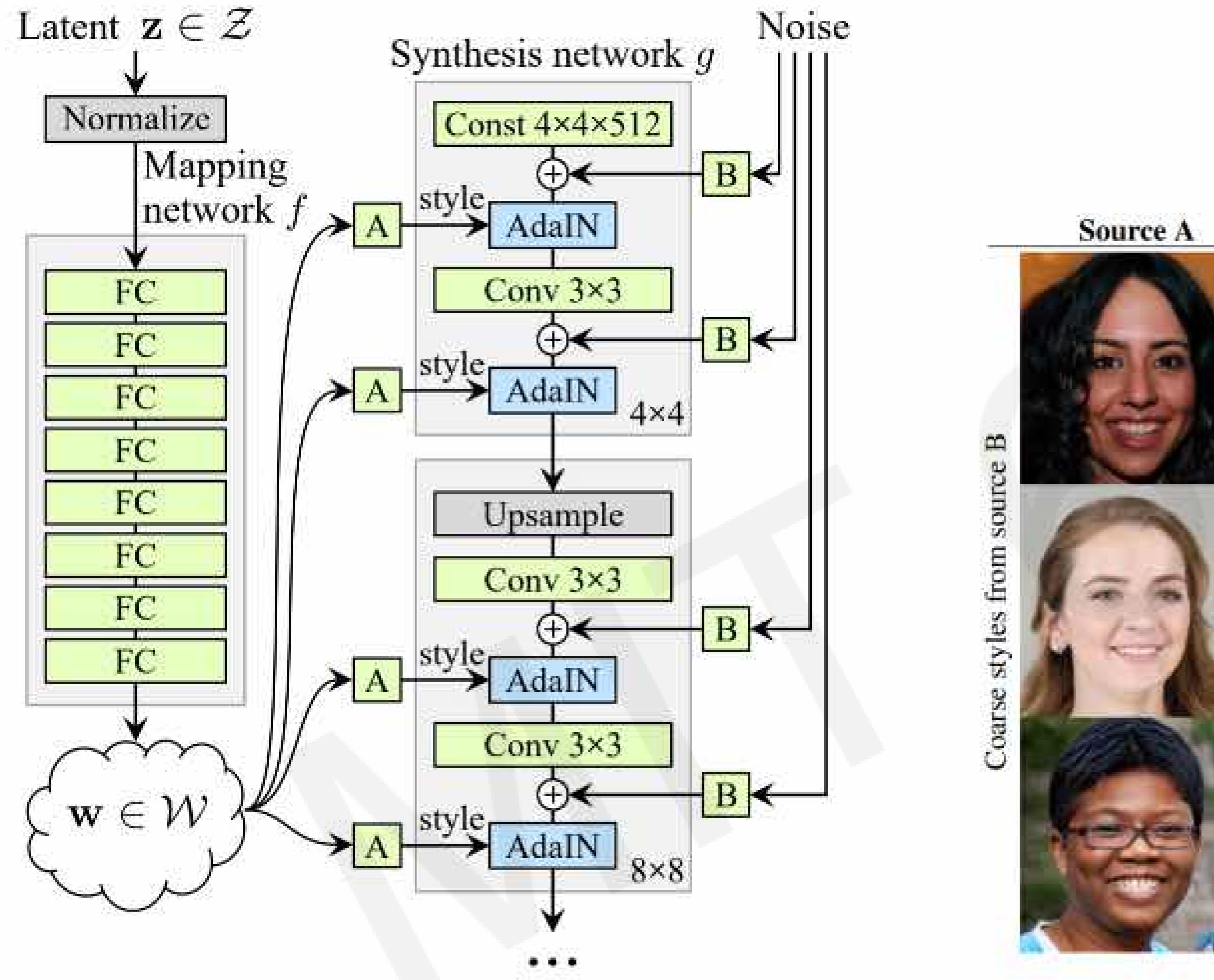
Progressive growing of GANs



Progressive growing of GANs: results



StyleGAN(2): progressive growing + style transfer



GANs for image synthesis: latest results

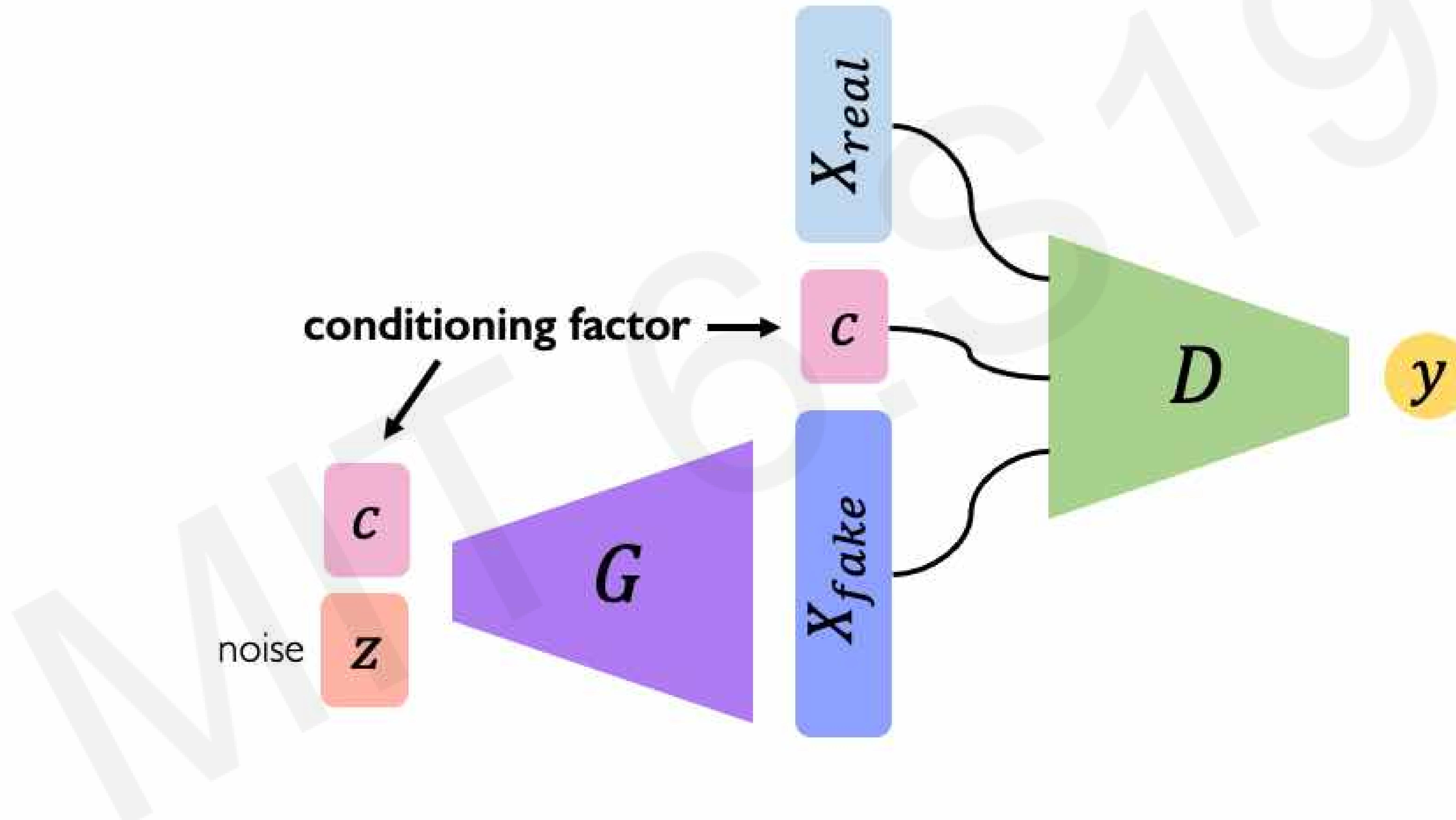


GANs for image synthesis: latest results

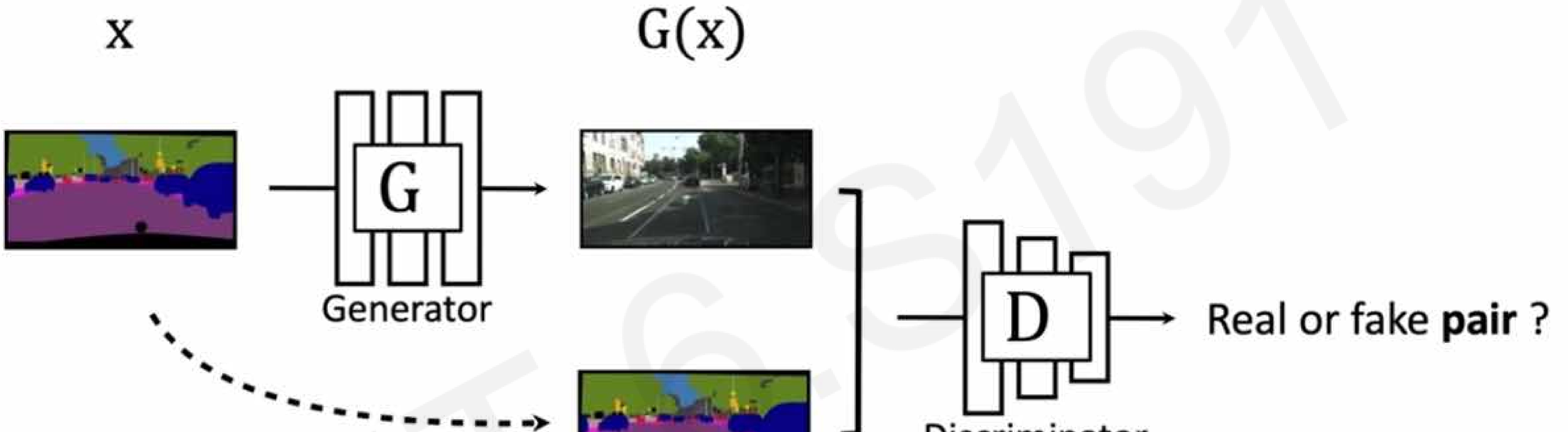


Conditional GANs

What if we want to control the nature of the output, by **conditioning** on a label?



Conditional GANs and pix2pix: paired translation

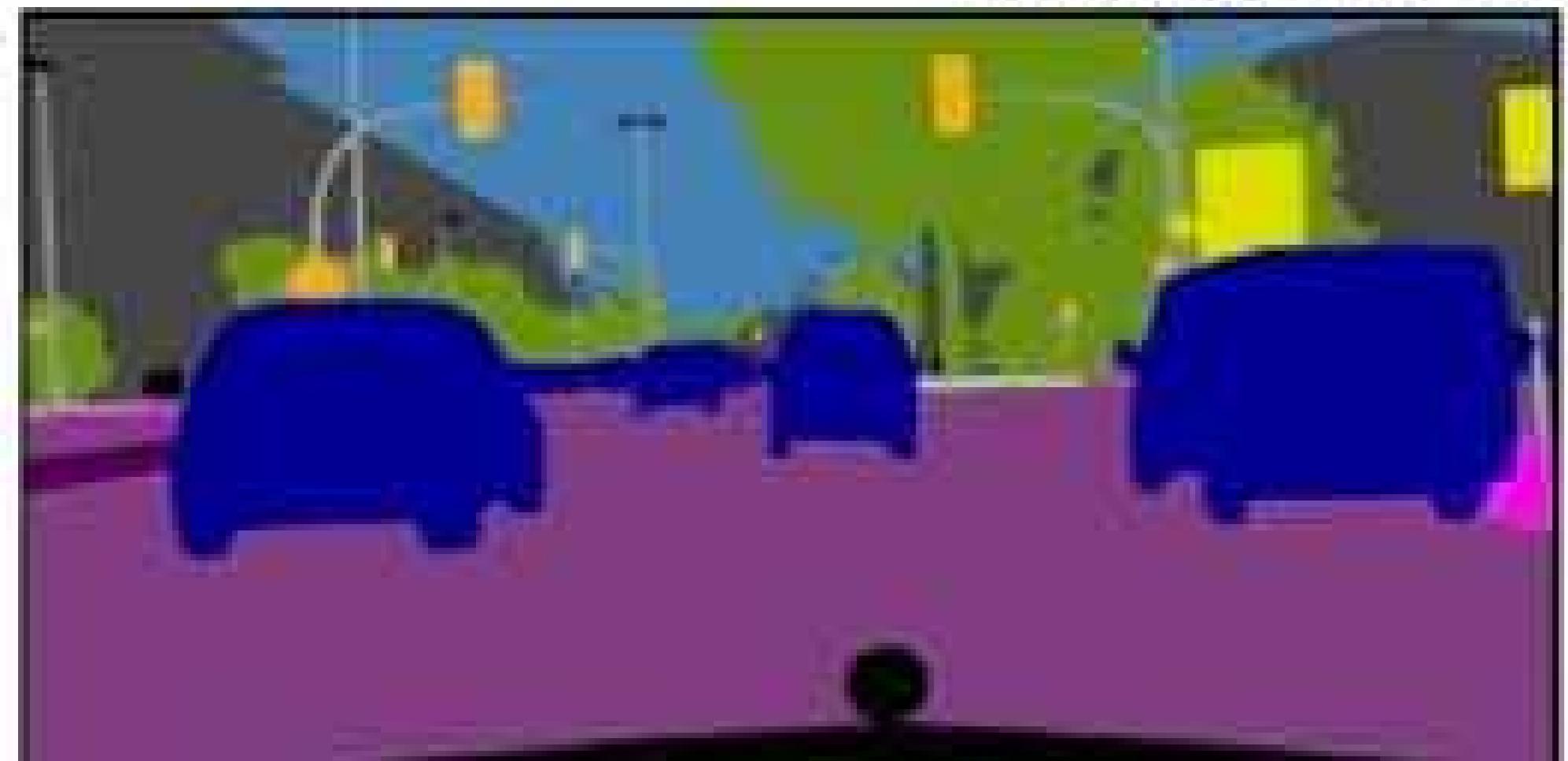


The discriminator, D, classifies between fake and real **pairs**.

The generator, G, learns to fool the discriminator.

Applications of paired translation

Labels to Street Scene



input



output

Paired translation: results

Map → Aerial View



input

output

Aerial View → Map



input

output

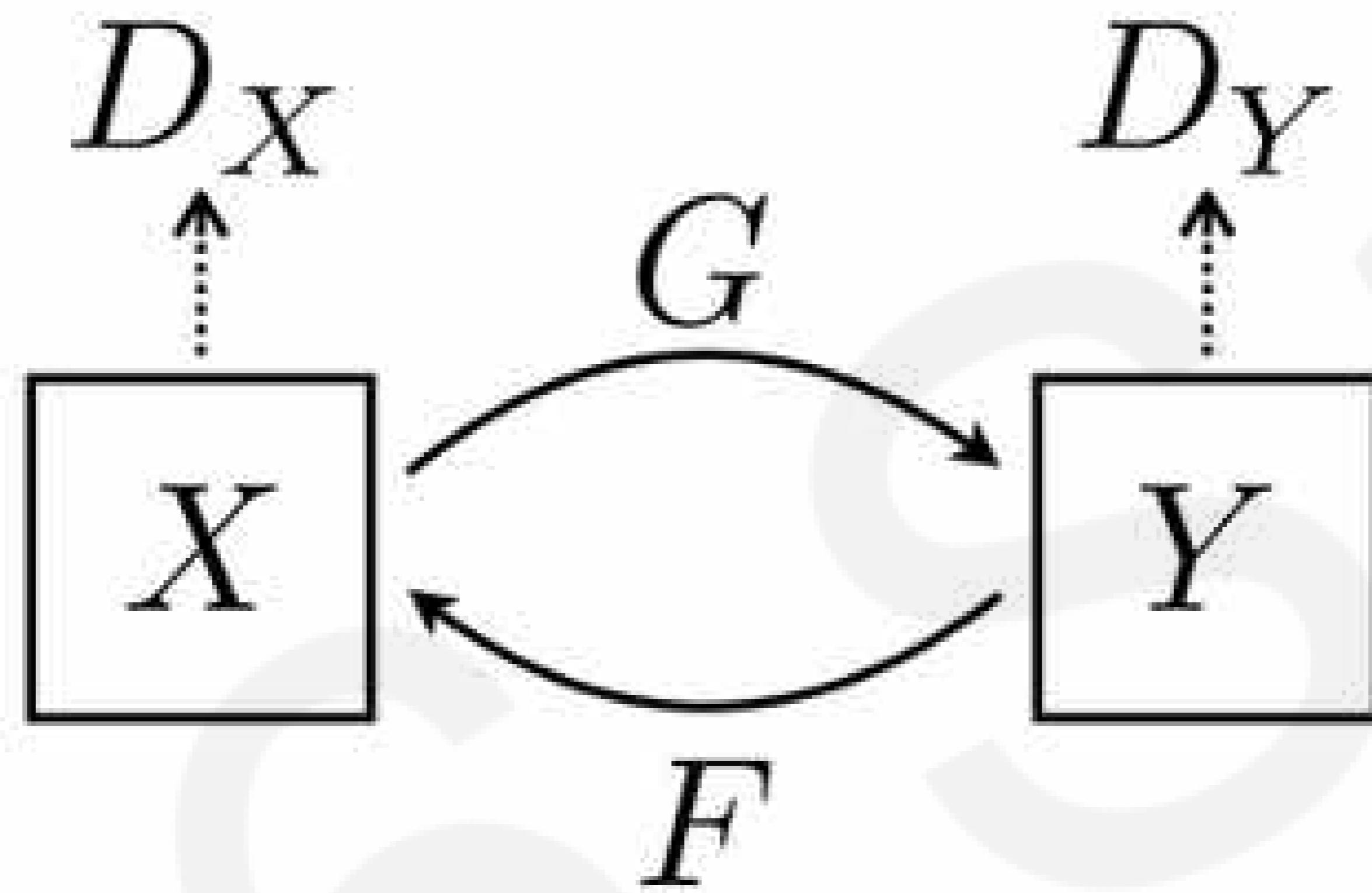
Paired translation: coloring from edges

MIT 6



CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.



Distribution transformations

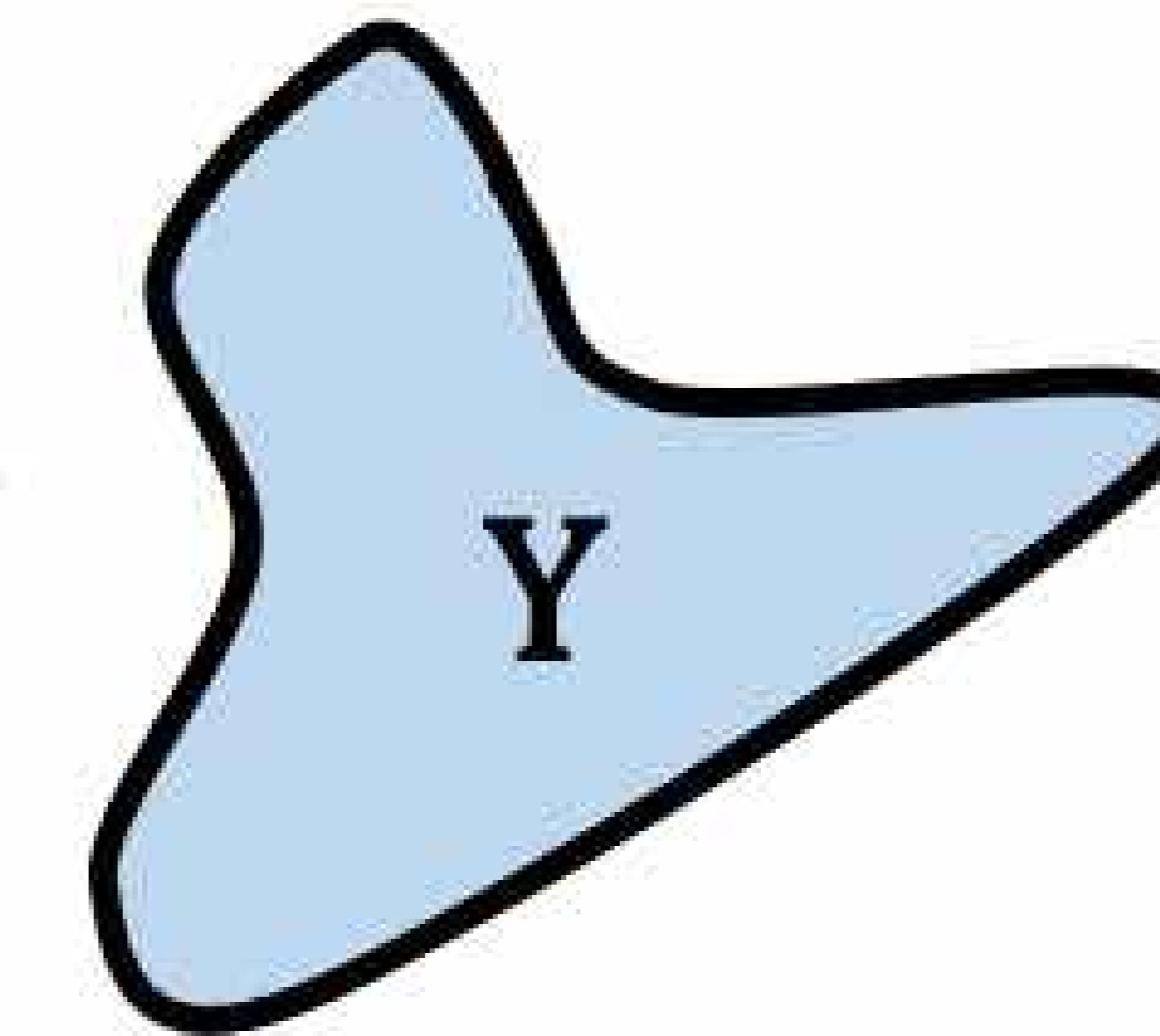
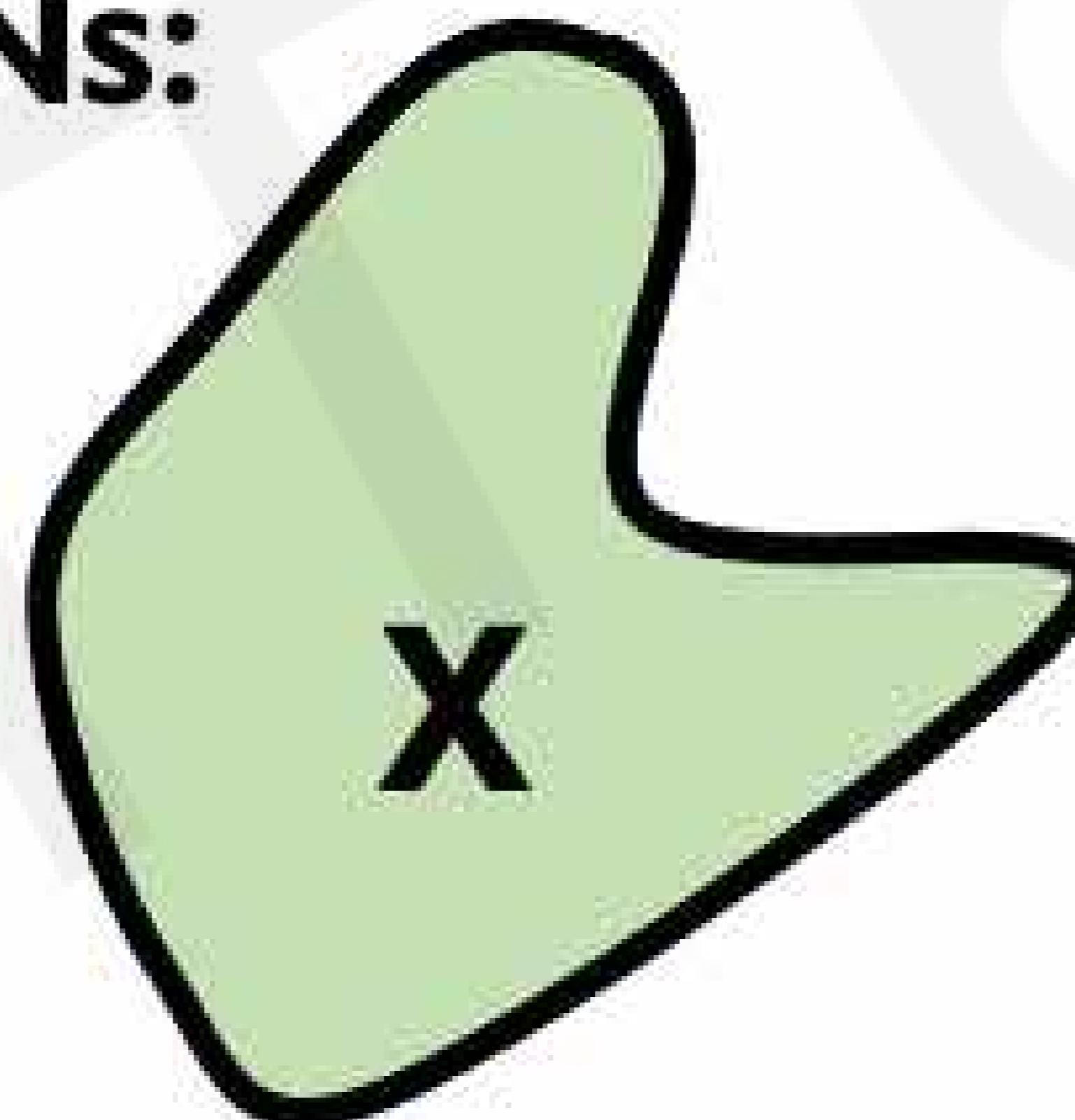
GANs:

Gaussian noise
 $z \sim N(0,1)$



Gaussian noise → target data manifold

CycleGANs:



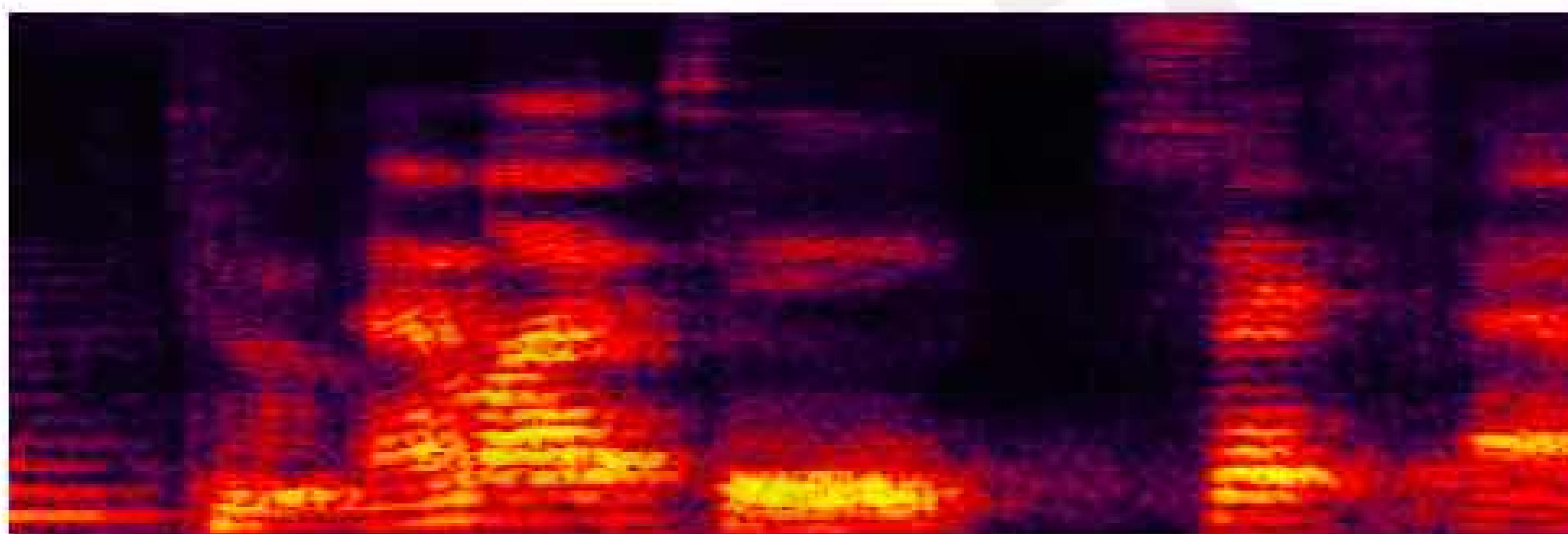
data manifold X → data manifold Y

CycleGAN: transforming speech

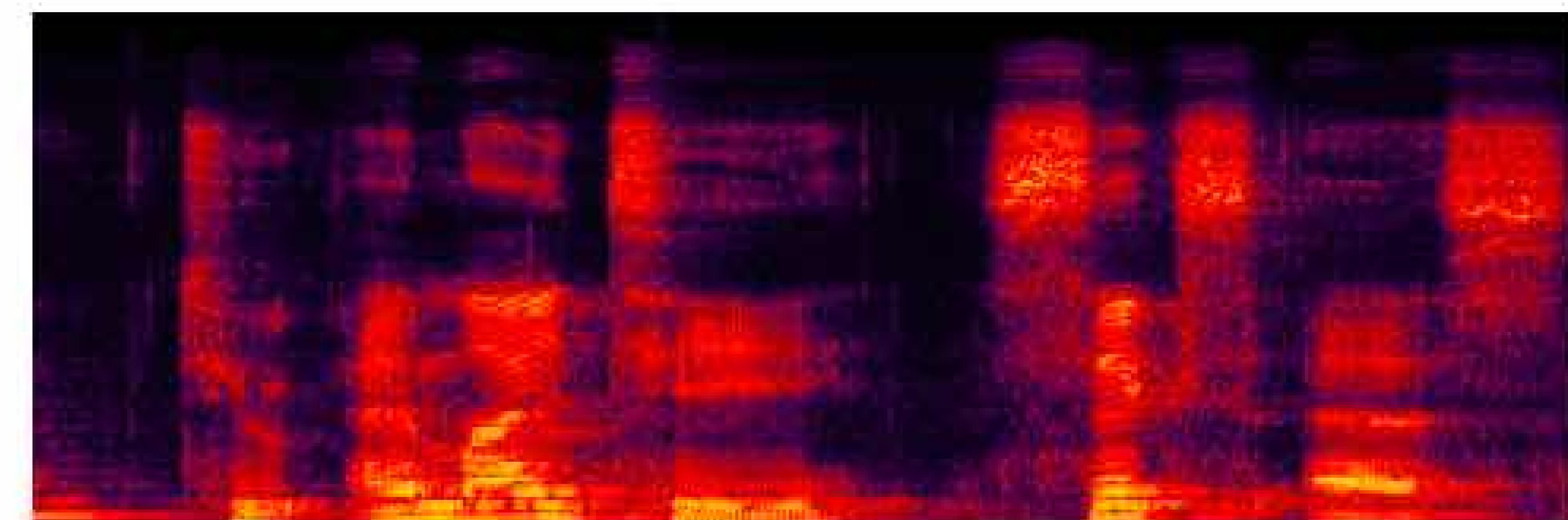
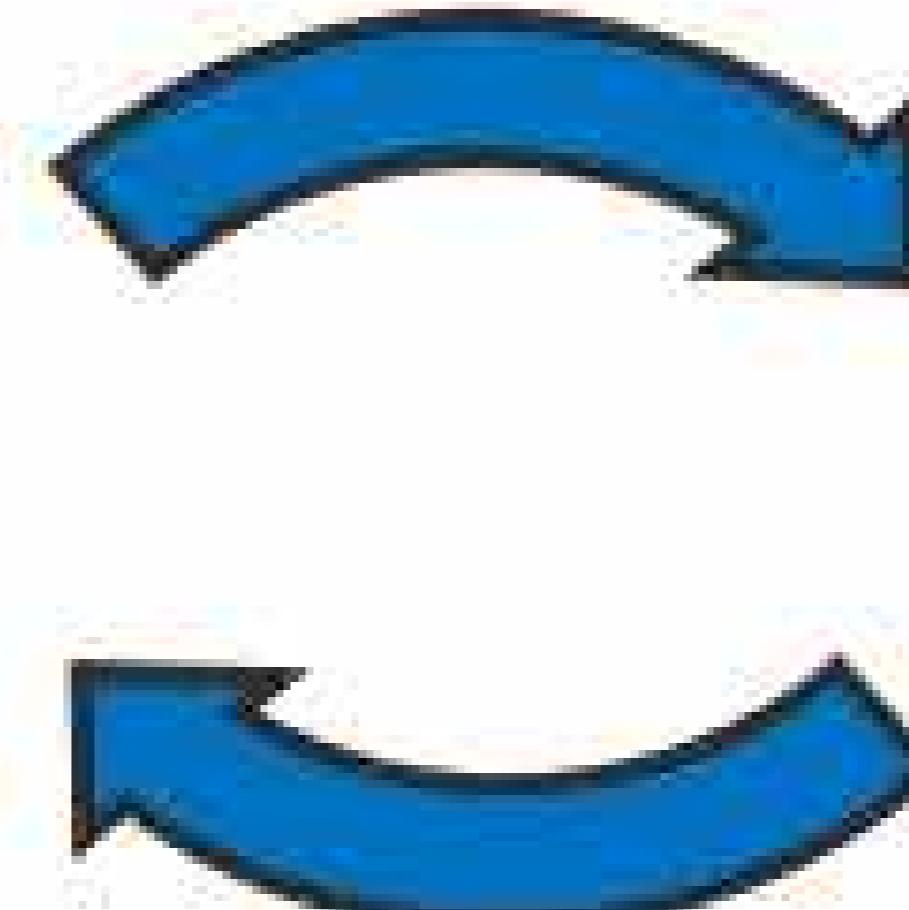
Audio waveform (A)



Audio waveform (B)



Spectrogram image (A)



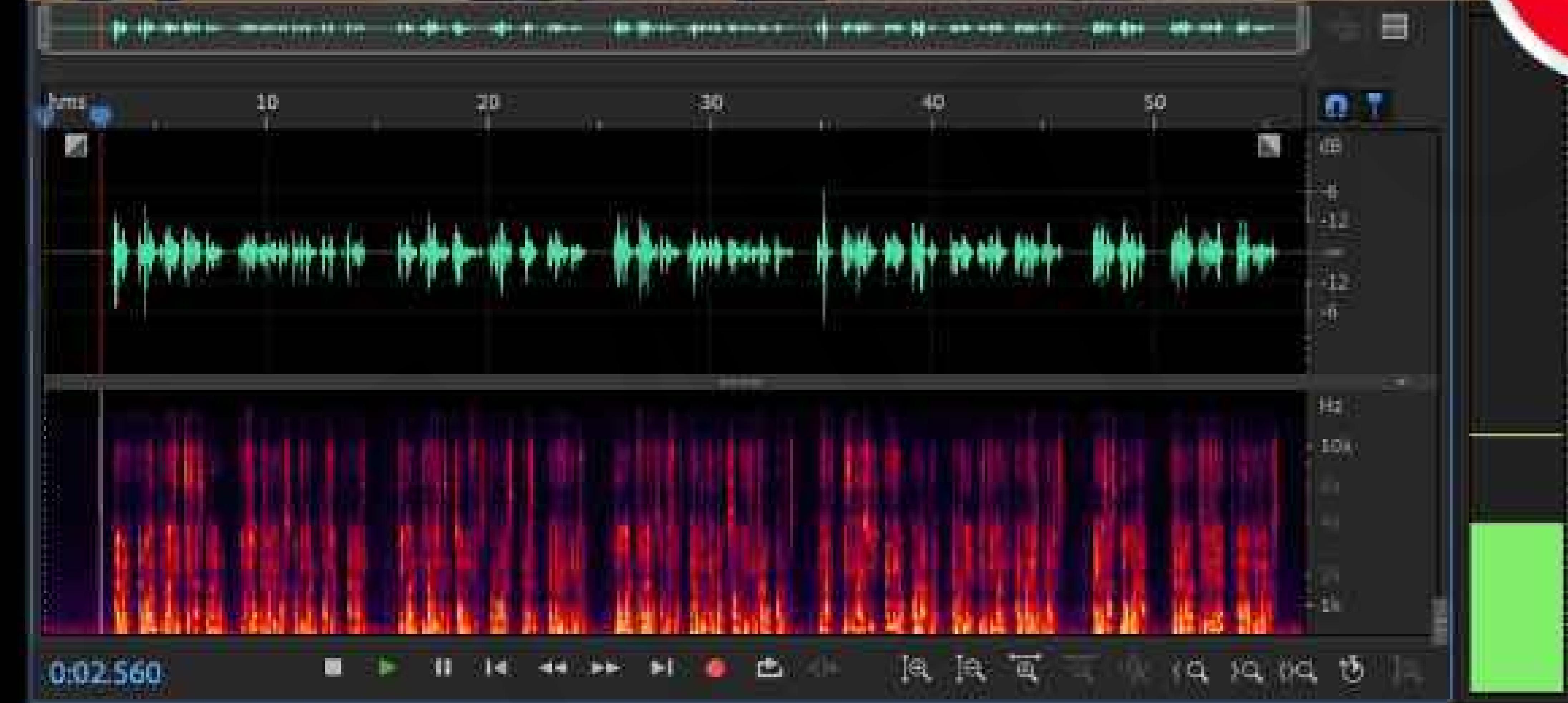
Spectrogram image (B)

Original (Amini)



Synthesized (Obama)

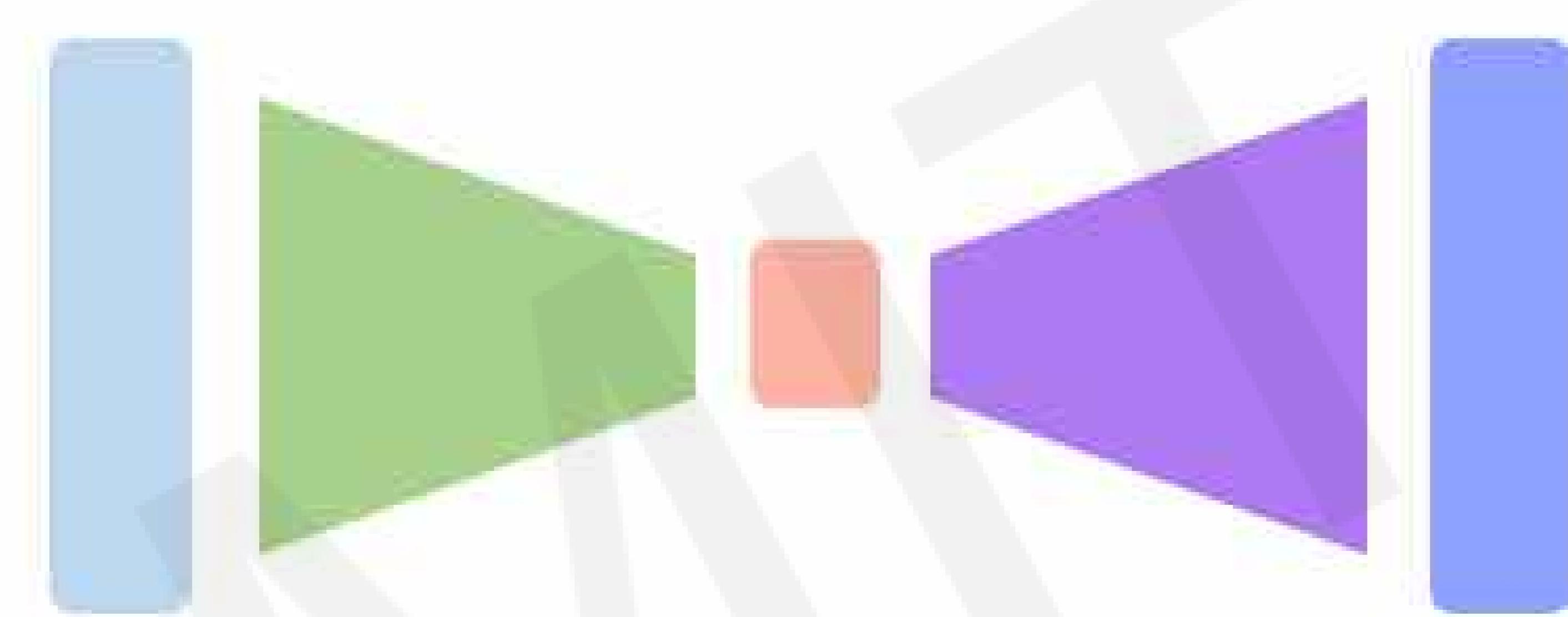
CANNYAI



Deep Generative Modeling: Summary

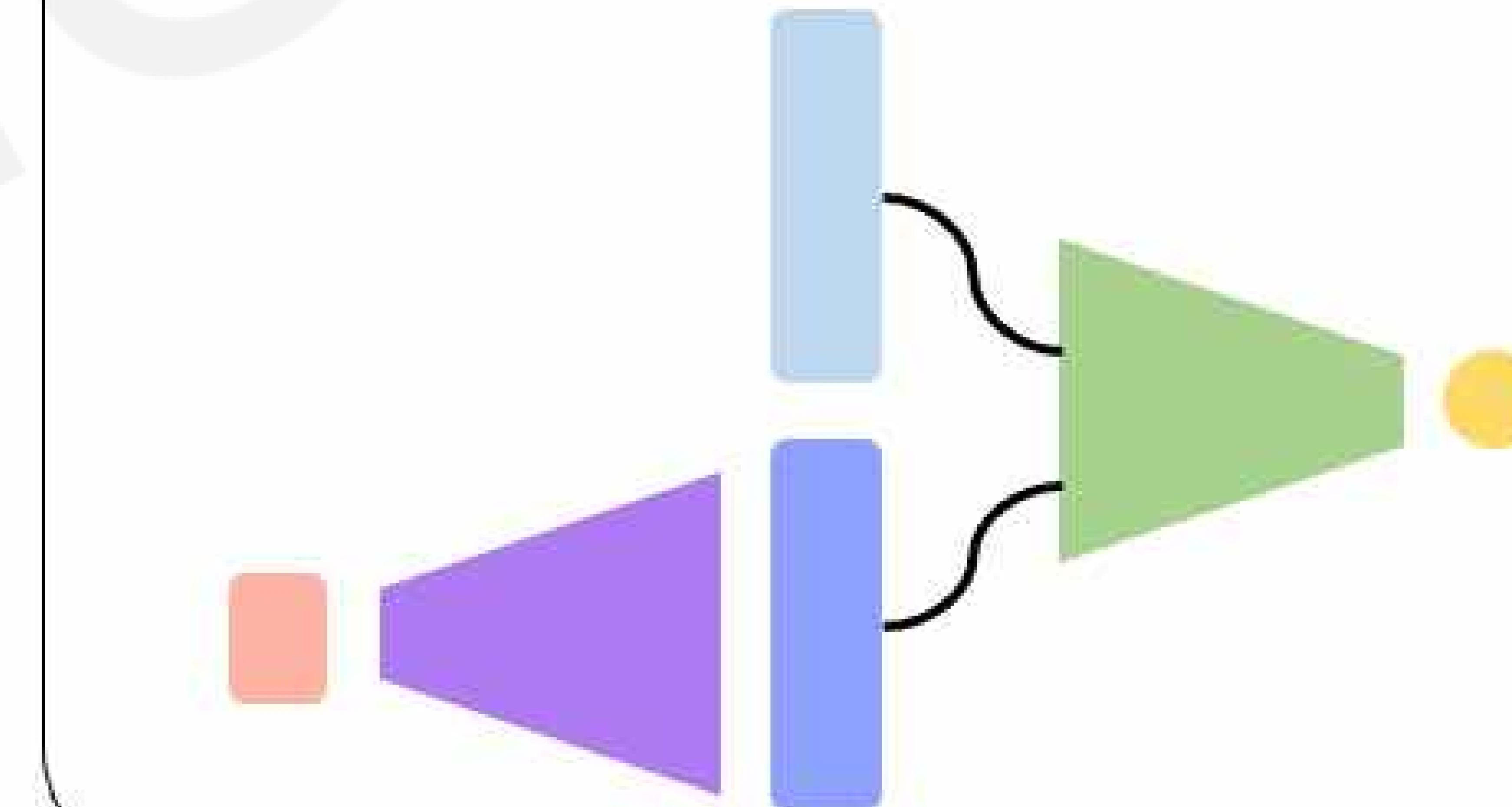
Autoencoders and Variational Autoencoders (VAEs)

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



Generative Adversarial Networks (GANs)

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





6.S191:

Introduction to Deep Learning

Lab 2: Computer Vision

Link to download labs:
<http://introtodeeplearning.com#schedule>

1. Open the lab in Google Colab
2. Start executing code blocks and filling in the #TODOs
3. Need help? Come to the 10-250 / Gather.Town!