Advanced Deep Learning Architectures
COMP 5214 & ELEC 5680

Instructor: Dr. Qifeng Chen
https://cqf.io

Generative Adversarial Networks

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!

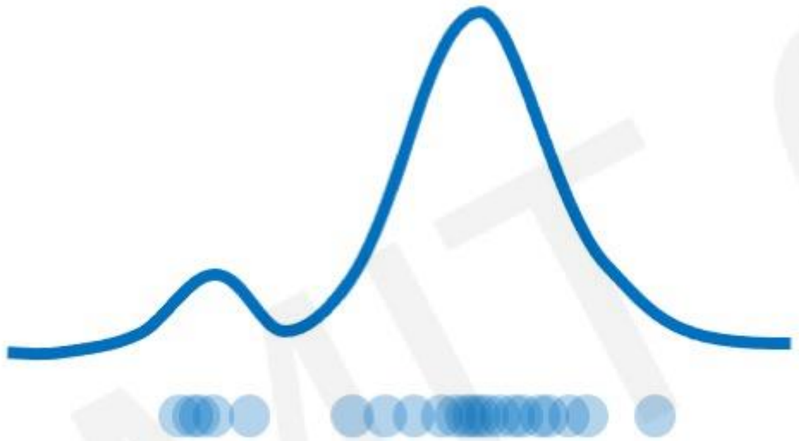
Goal: Learn the *hidden* or *underlying structure* of the data

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



samples

Sample Generation

Input samples

Training data ~ $P_{data}(x)$

Generated samples

Generated ~ $P_{model}(x)$

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

Why generative models? Debiasing

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

VS

Homogeneous skin color, pose

Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?

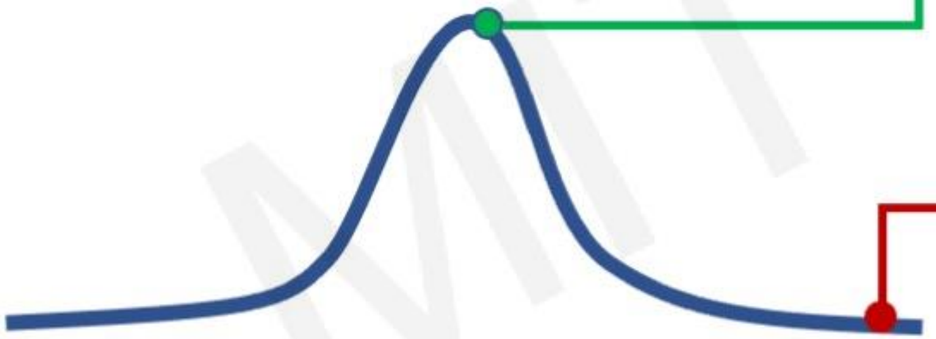
6.S191 Lab

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

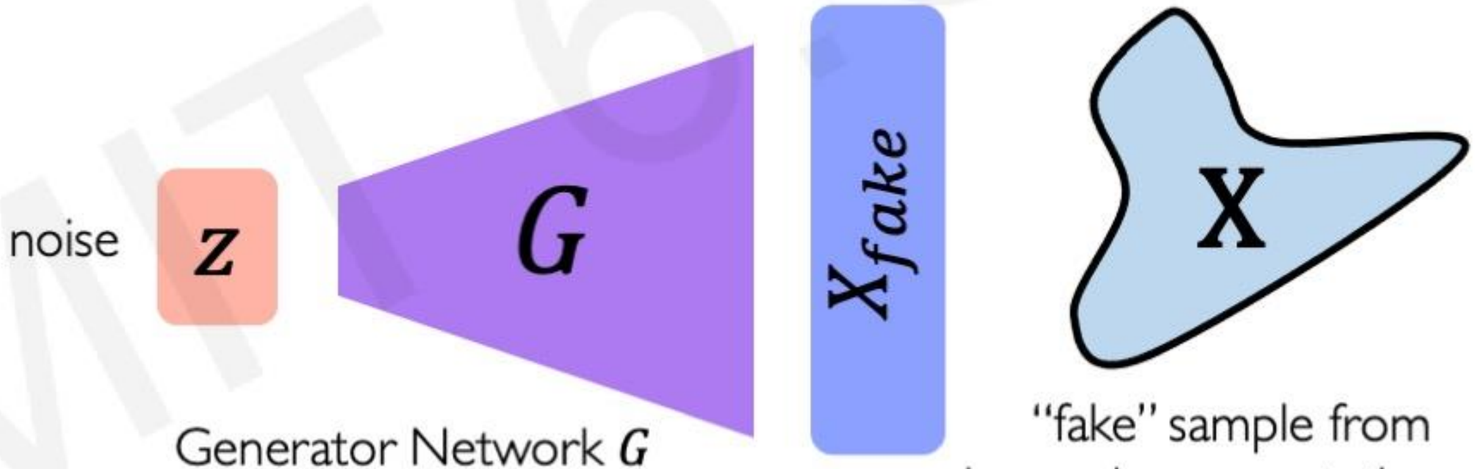
Generative Adversarial Networks (GANs)

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.

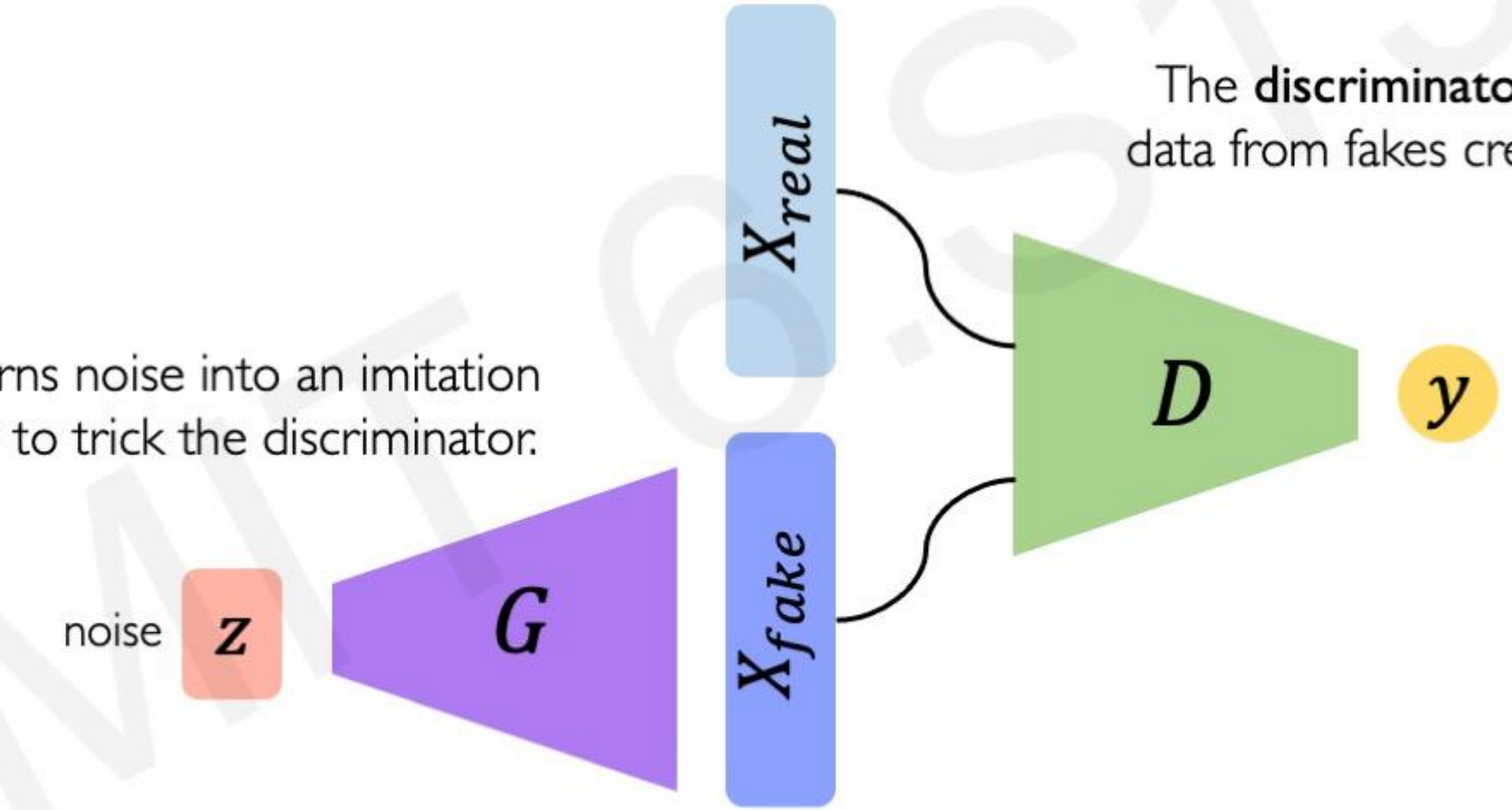


Generative Adversarial Networks (GANs)

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

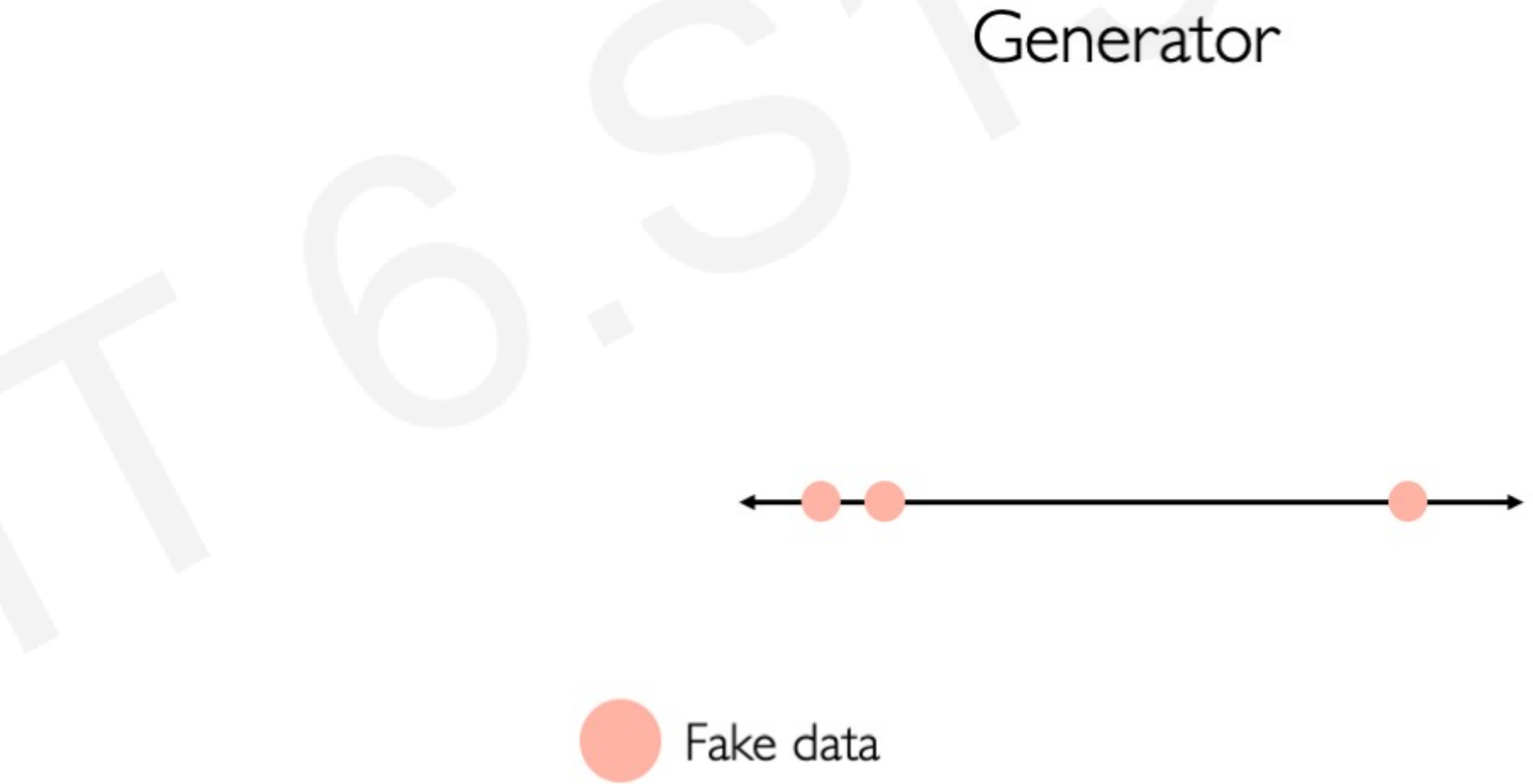
The **discriminator** tries to identify real data from fakes created by the generator.

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



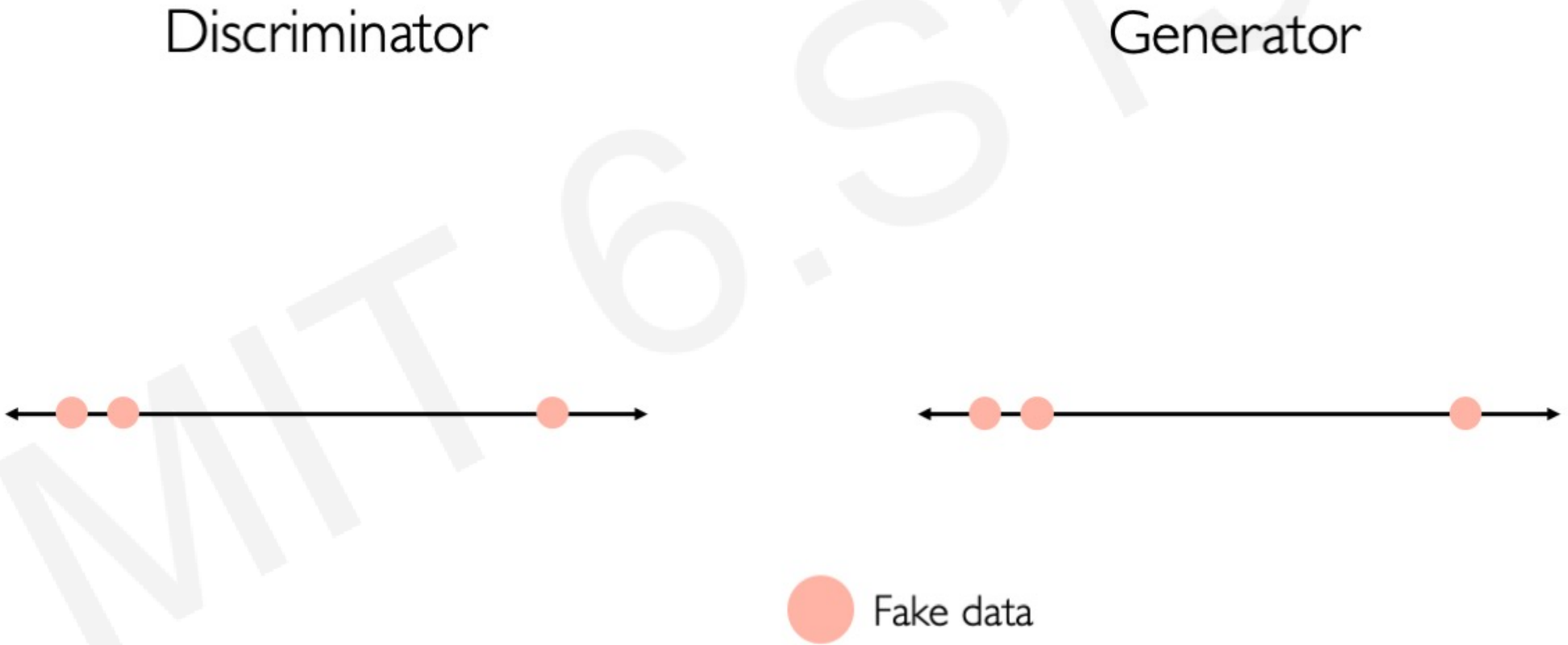
Intuition behind GANs

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



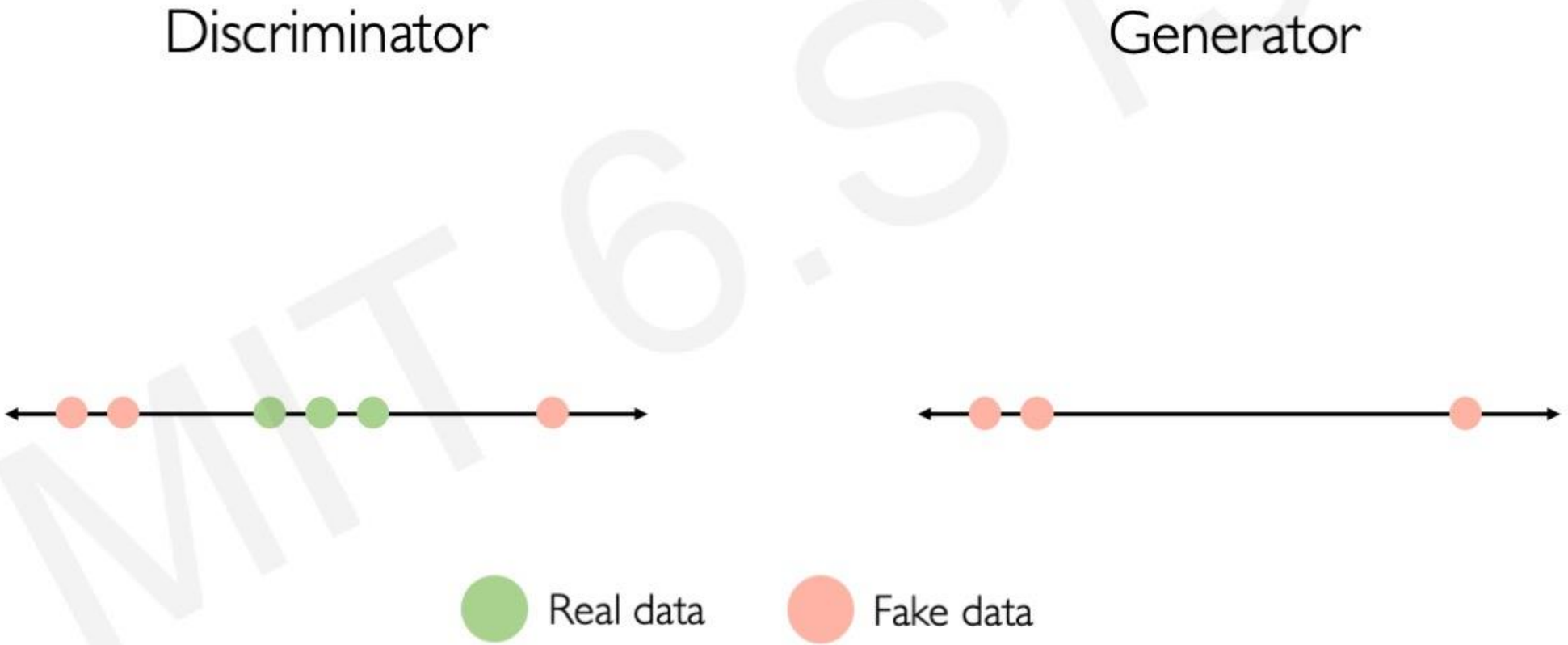
Intuition behind GANs

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



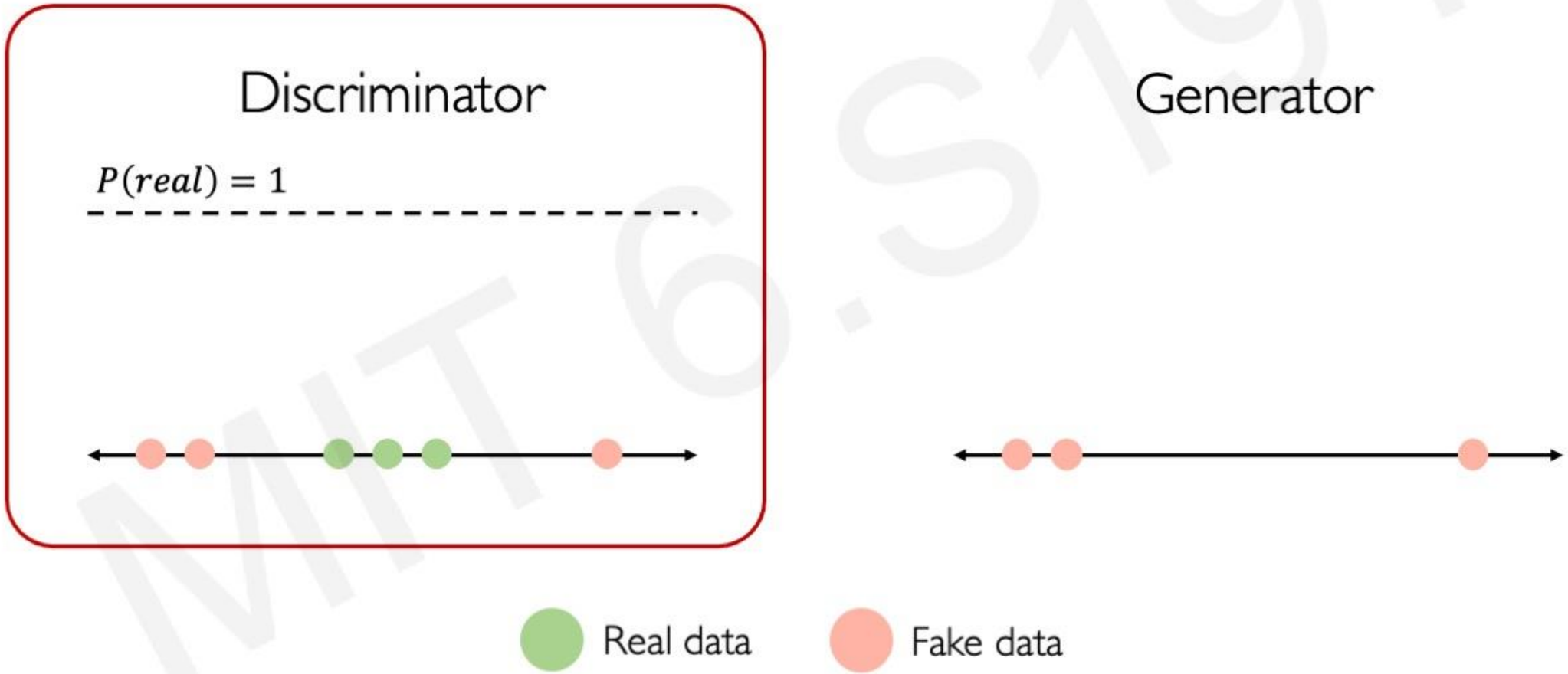
Intuition behind GANs

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



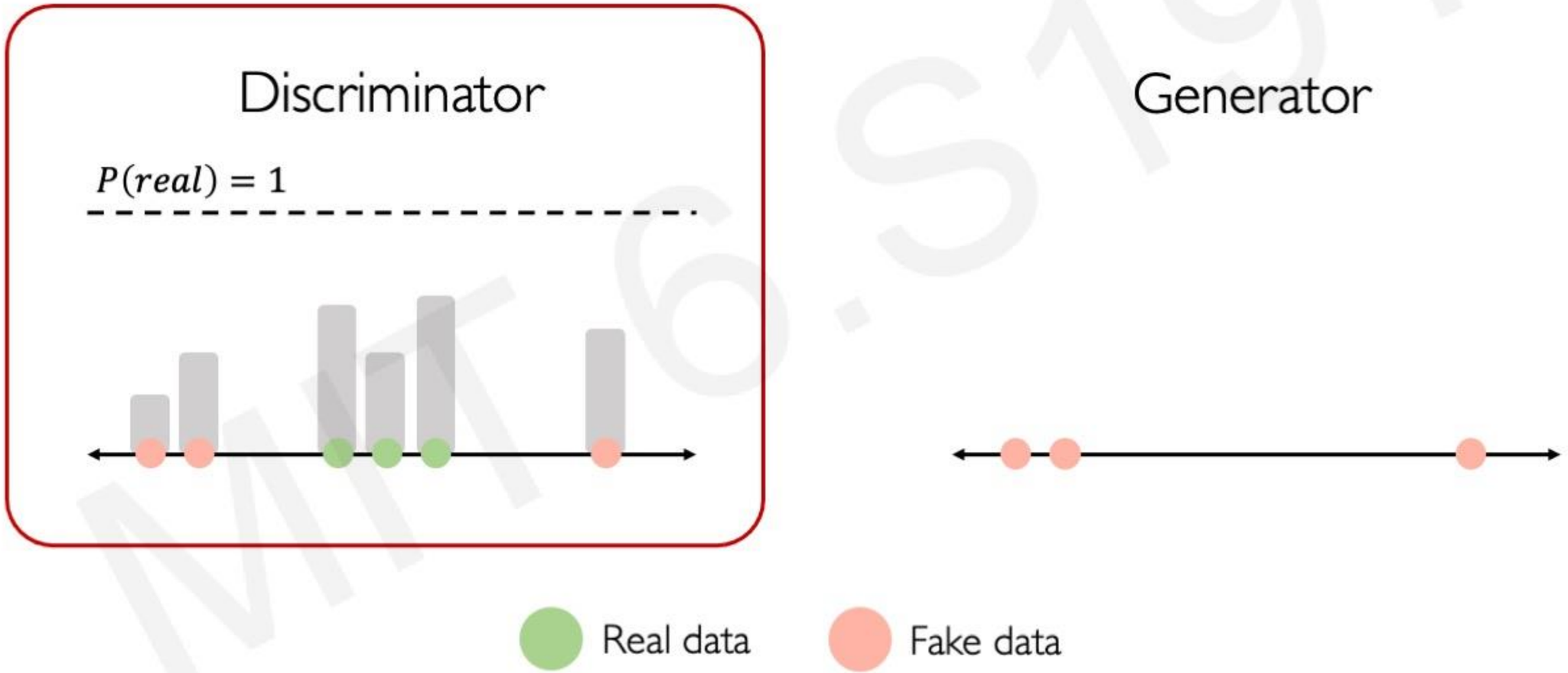
Intuition behind GANs

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



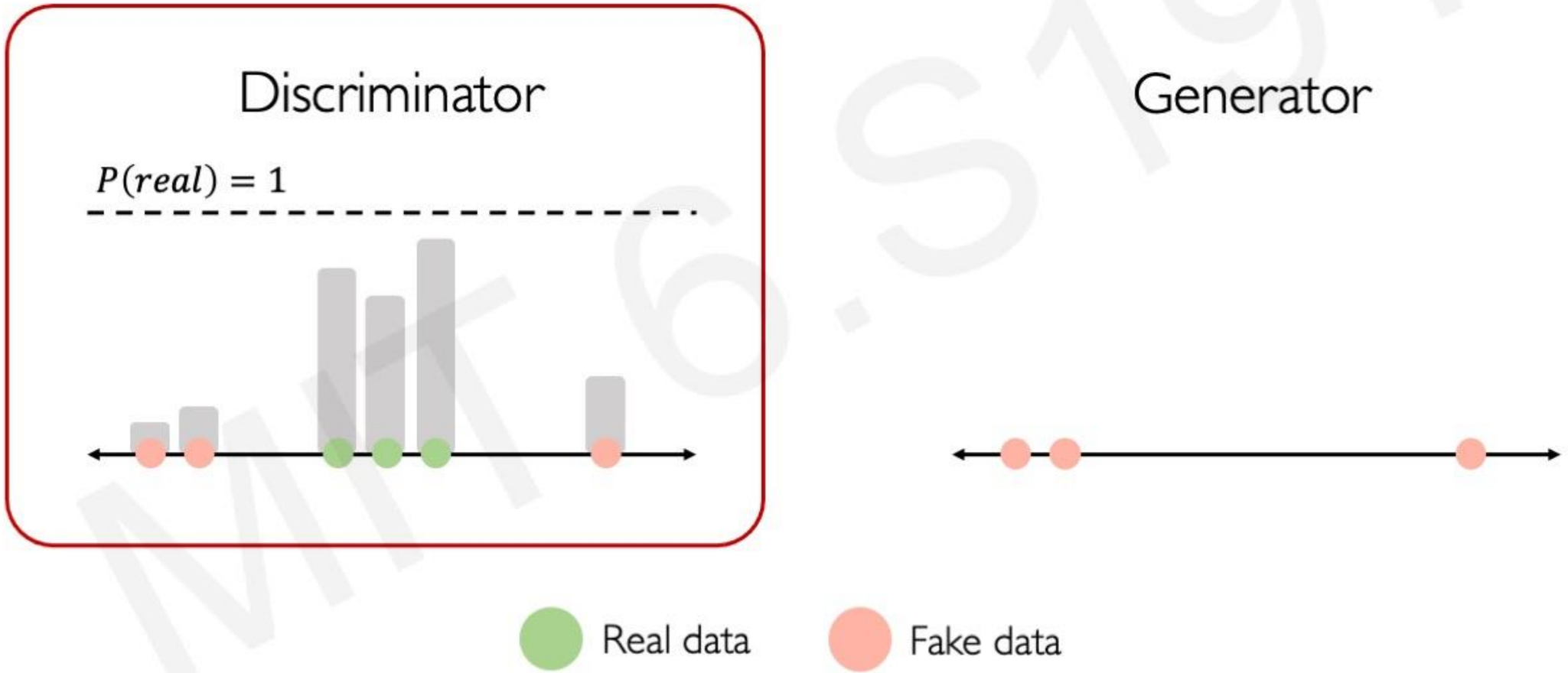
Intuition behind GANs

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



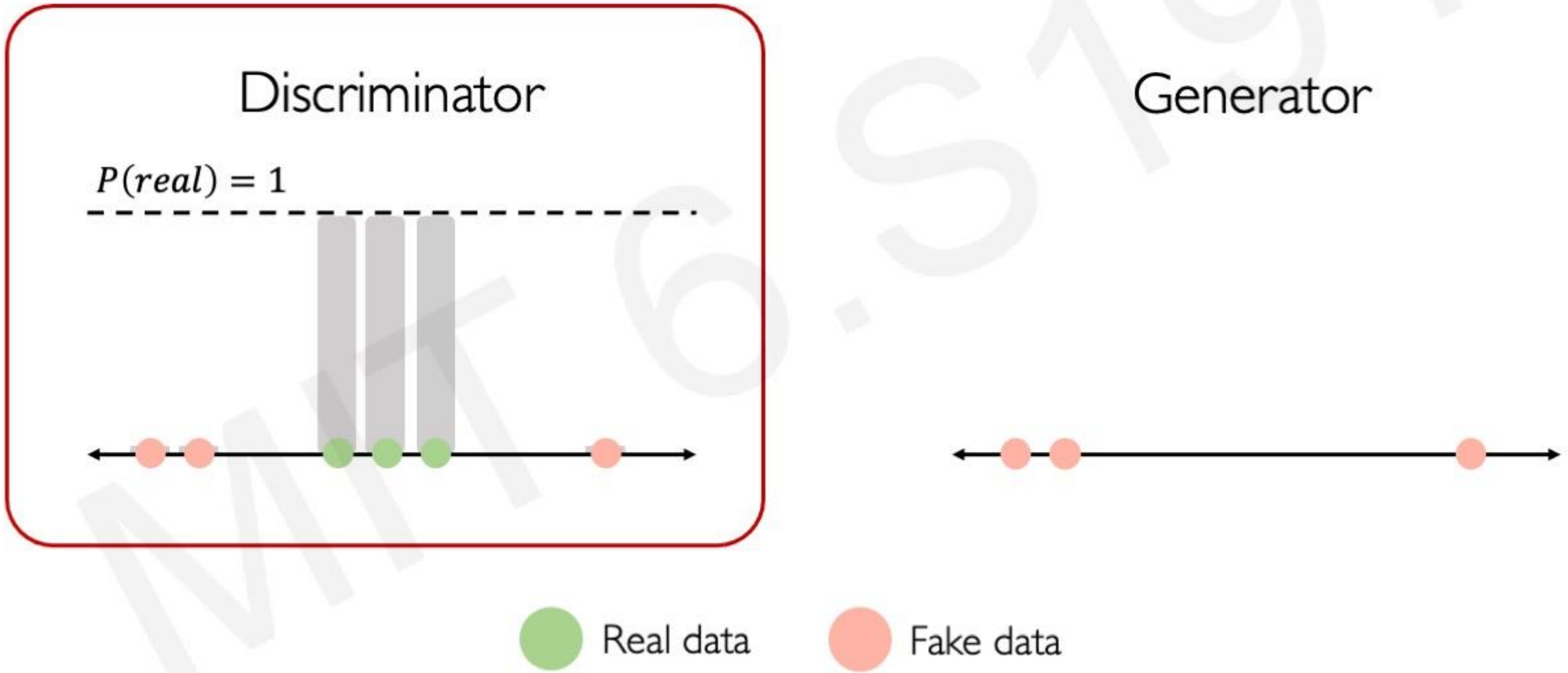
Intuition behind GANs

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



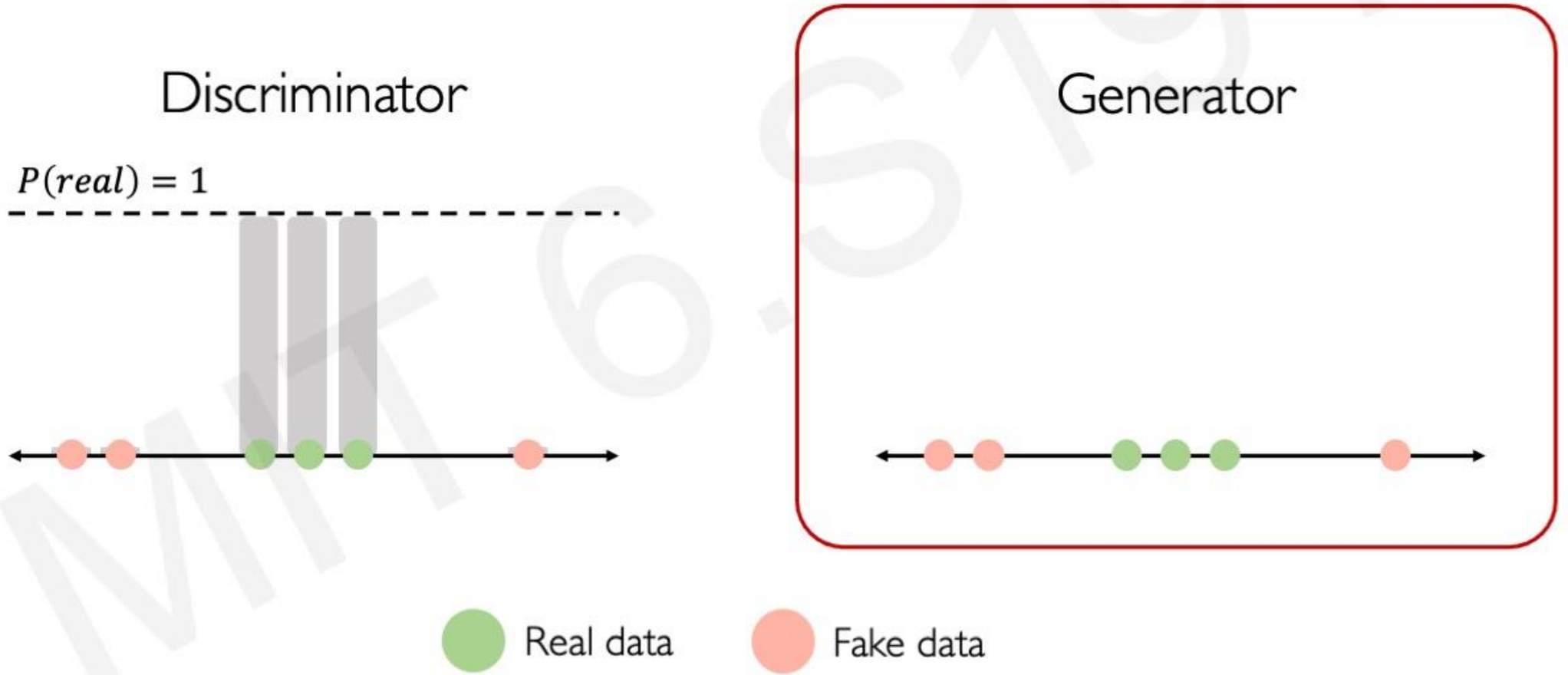
Intuition behind GANs

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



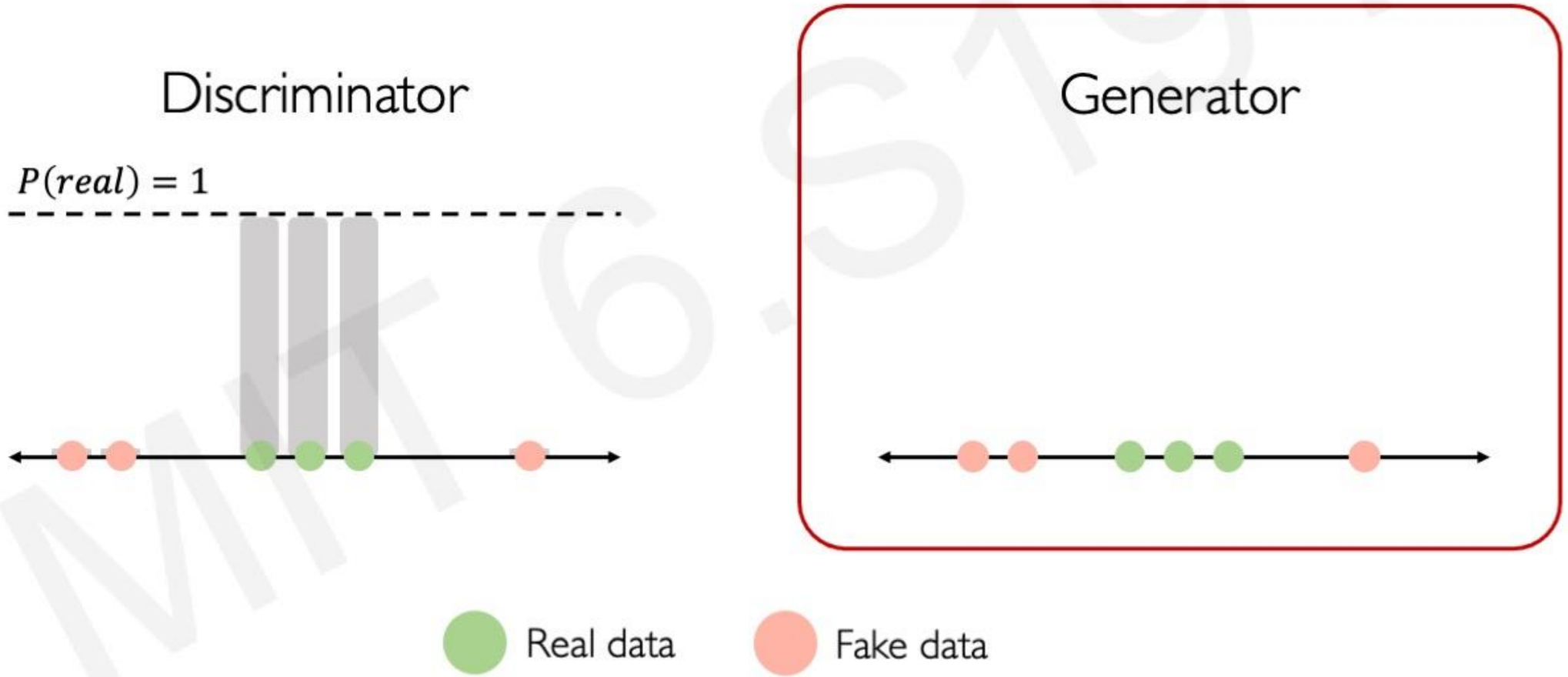
Intuition behind GANs

Generator tries to improve its imitation of the data.



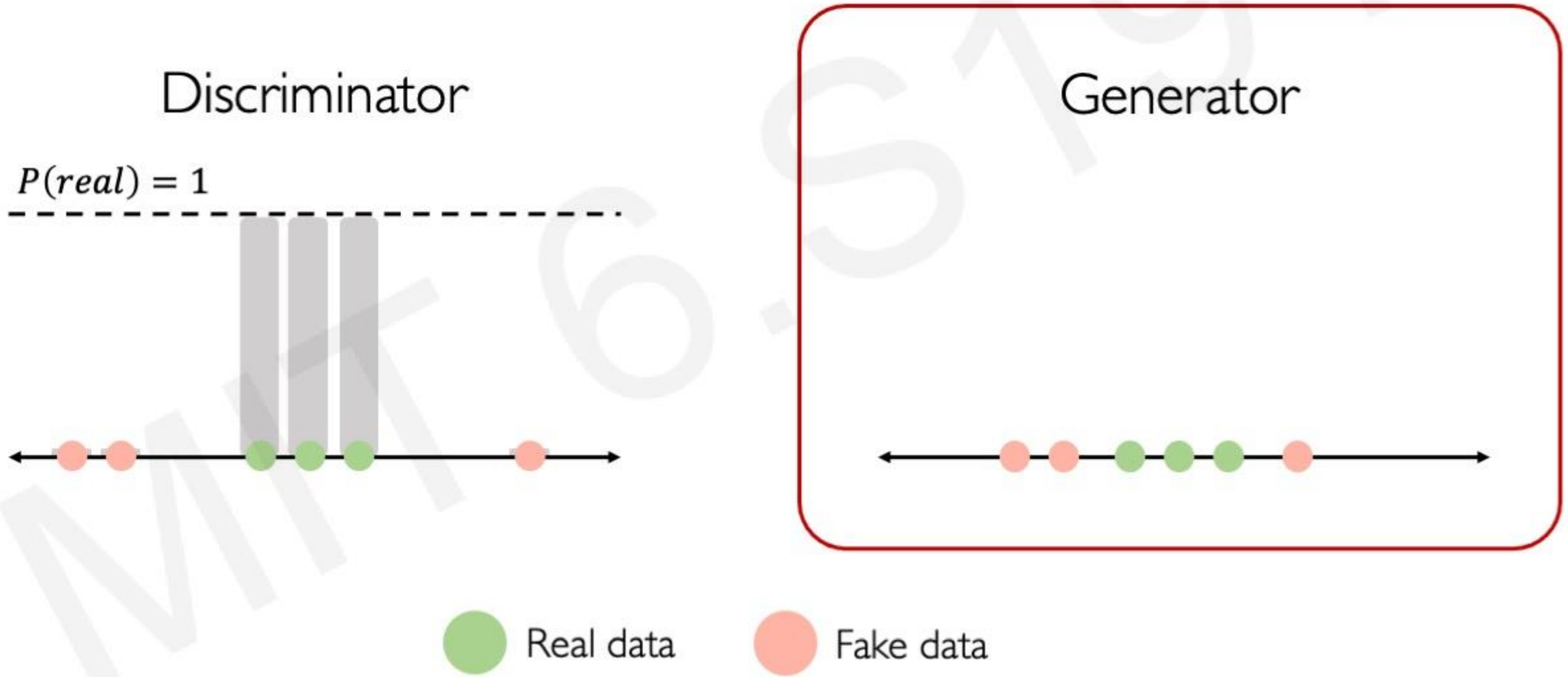
Intuition behind GANs

Generator tries to improve its imitation of the data.



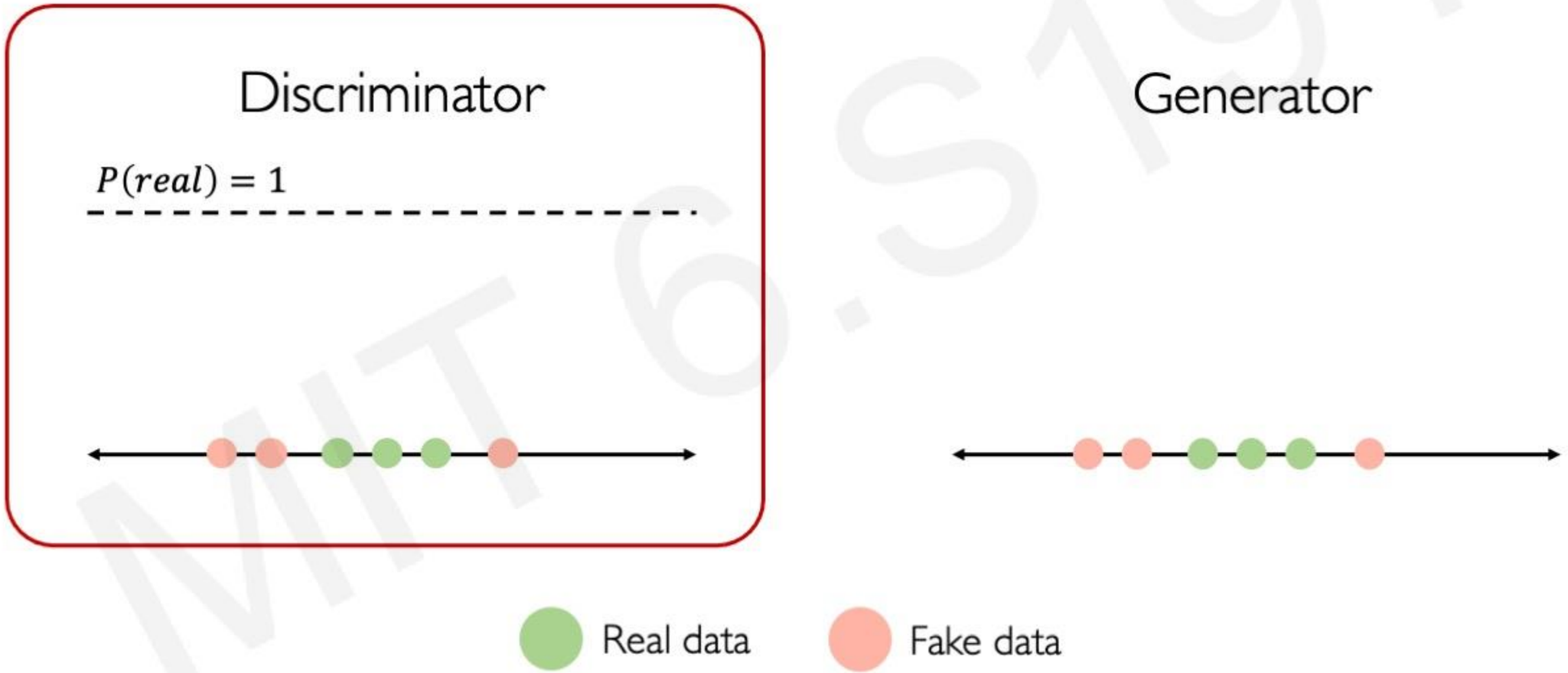
Intuition behind GANs

Generator tries to improve its imitation of the data.



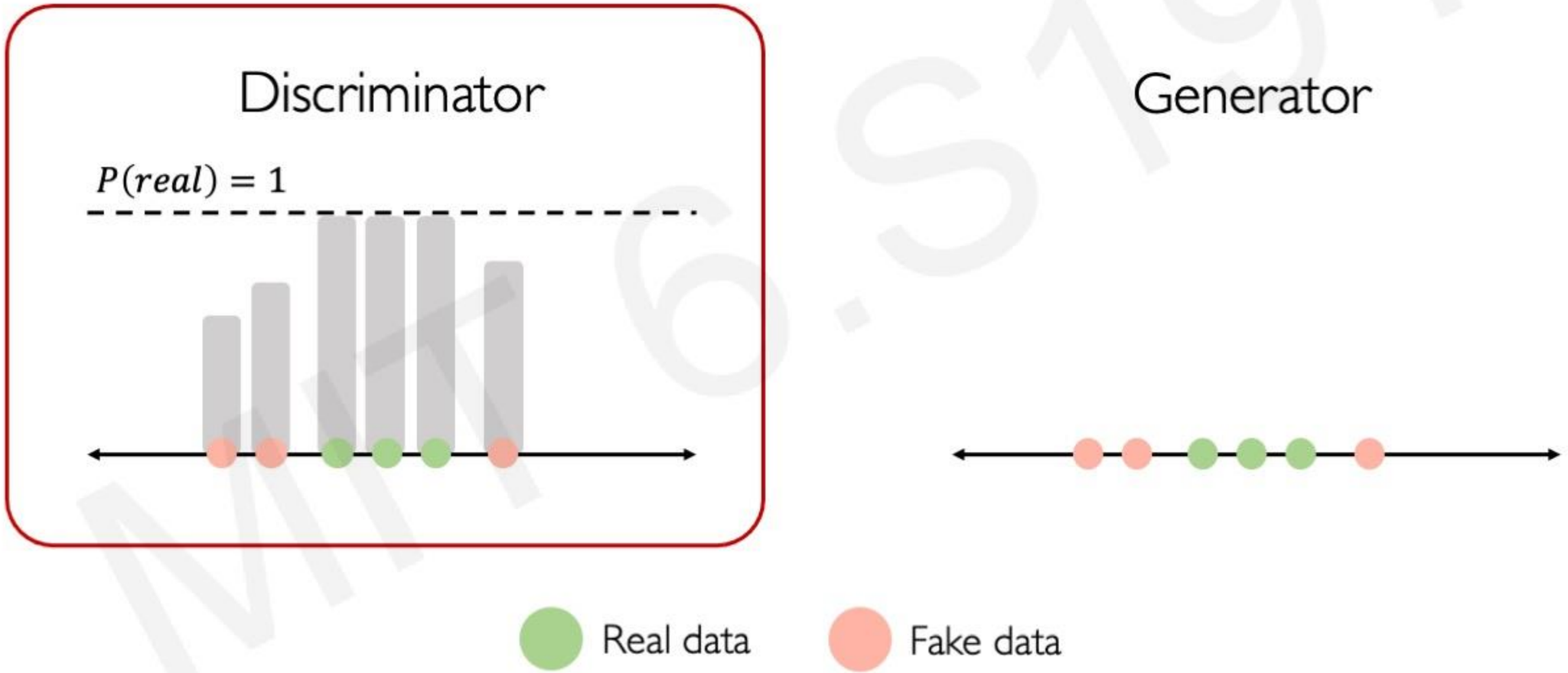
Intuition behind GANs

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



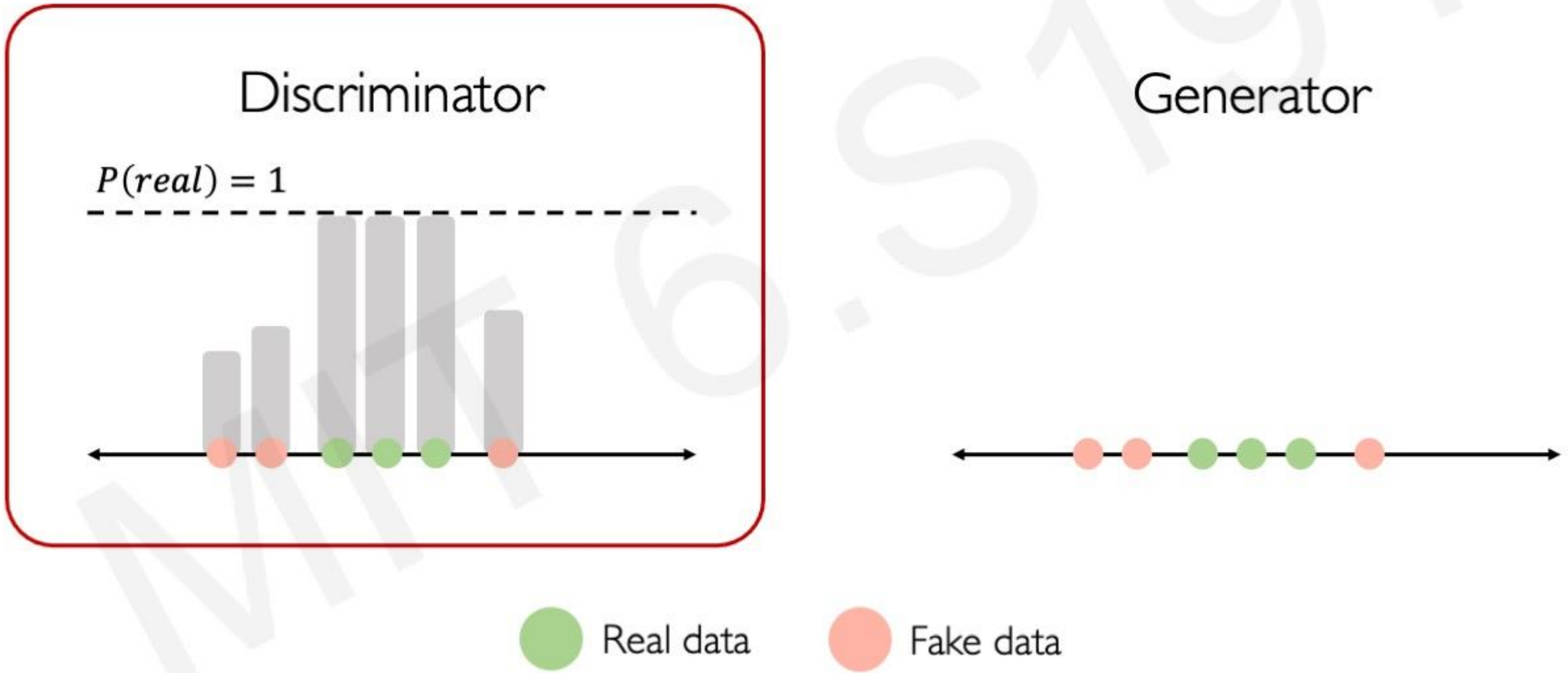
Intuition behind GANs

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



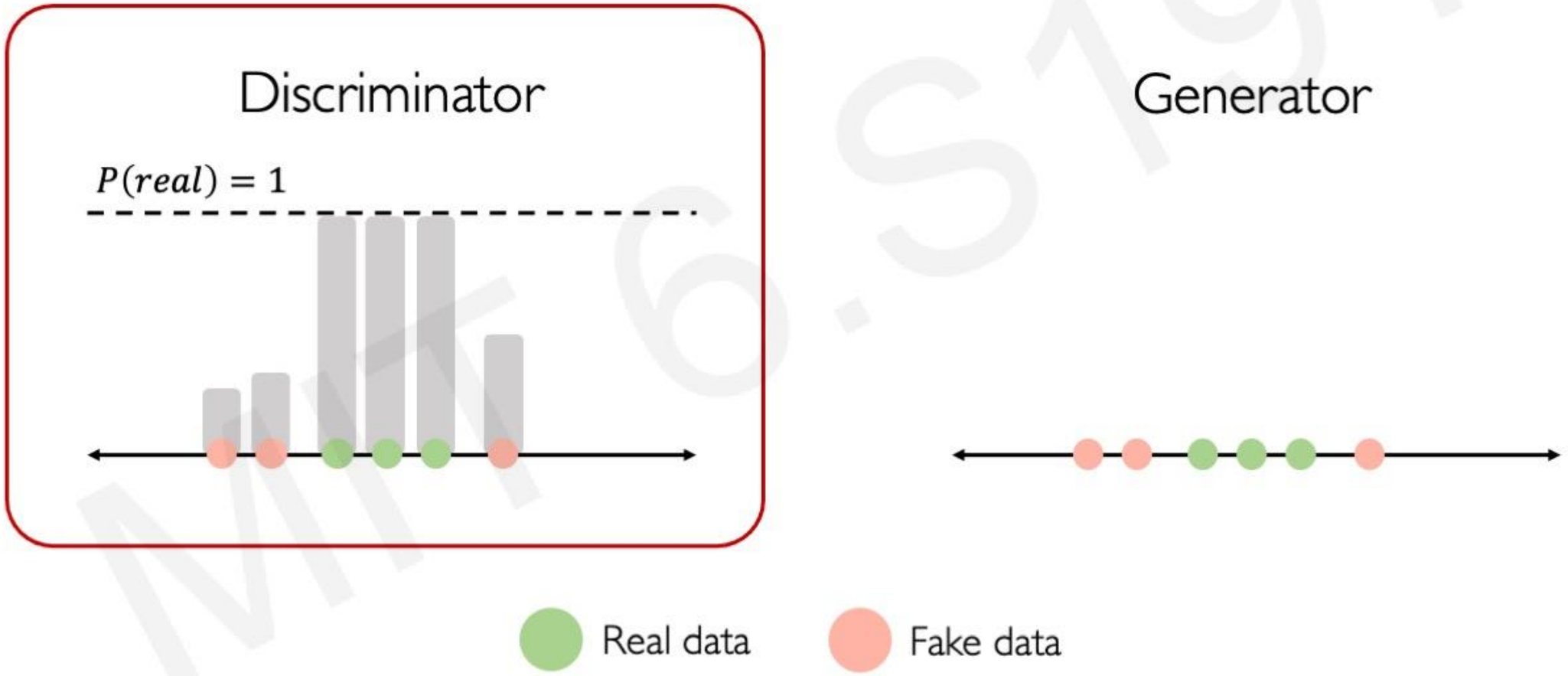
Intuition behind GANs

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



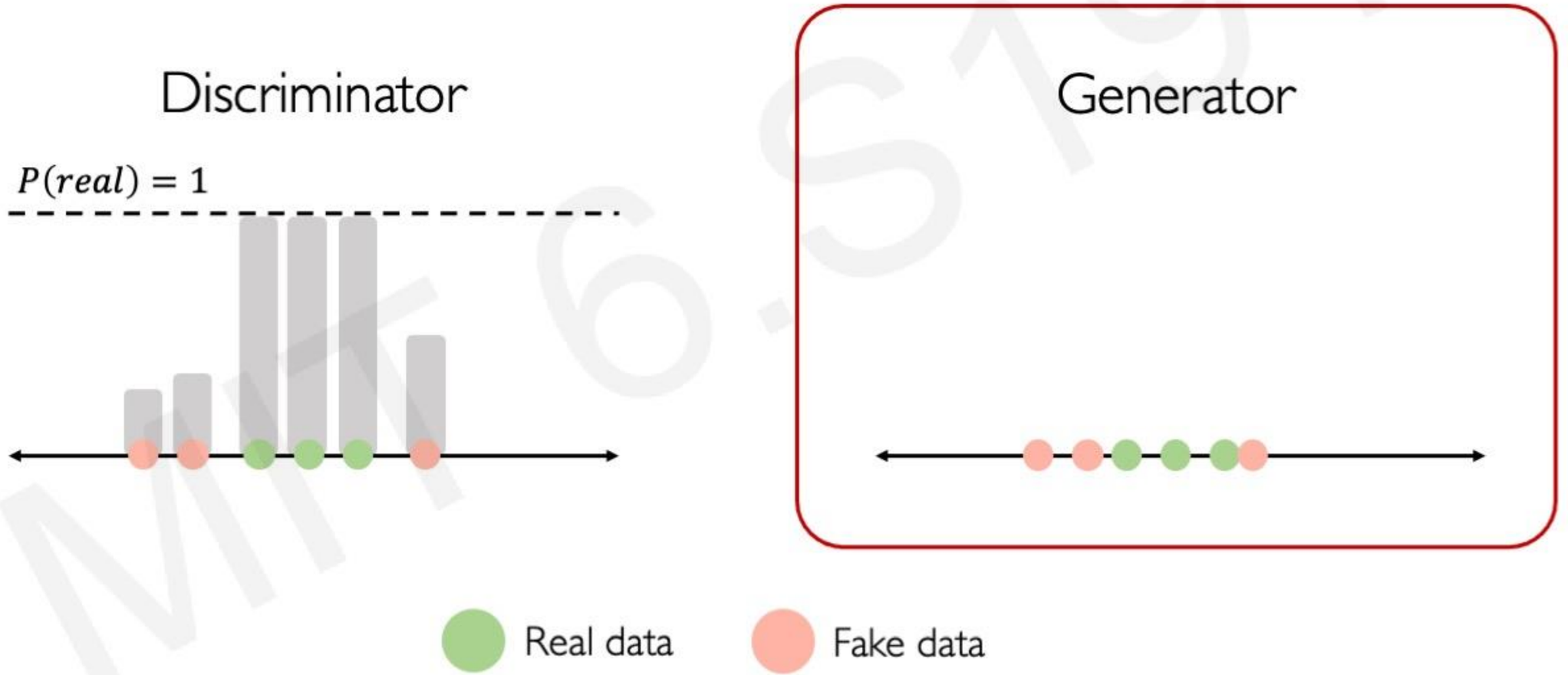
Intuition behind GANs

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



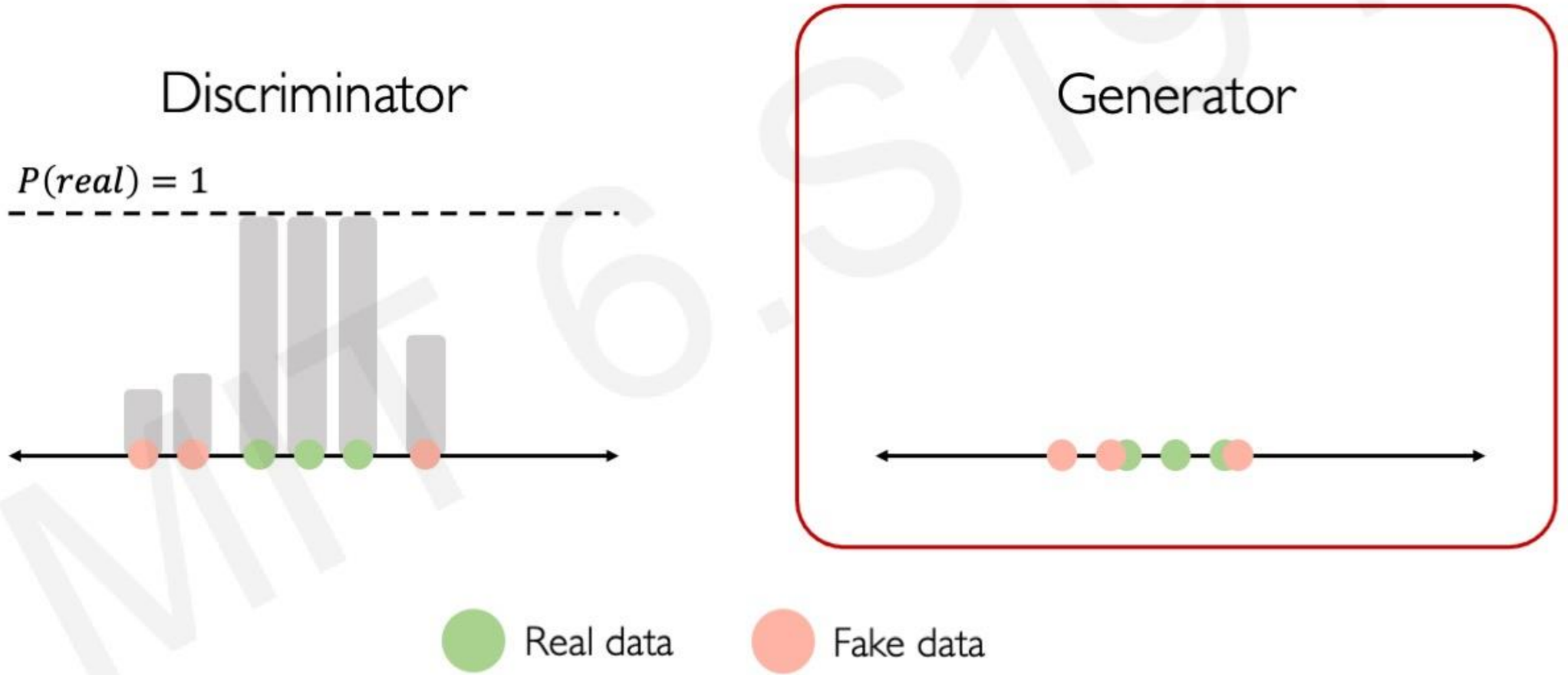
Intuition behind GANs

Generator tries to improve its imitation of the data.



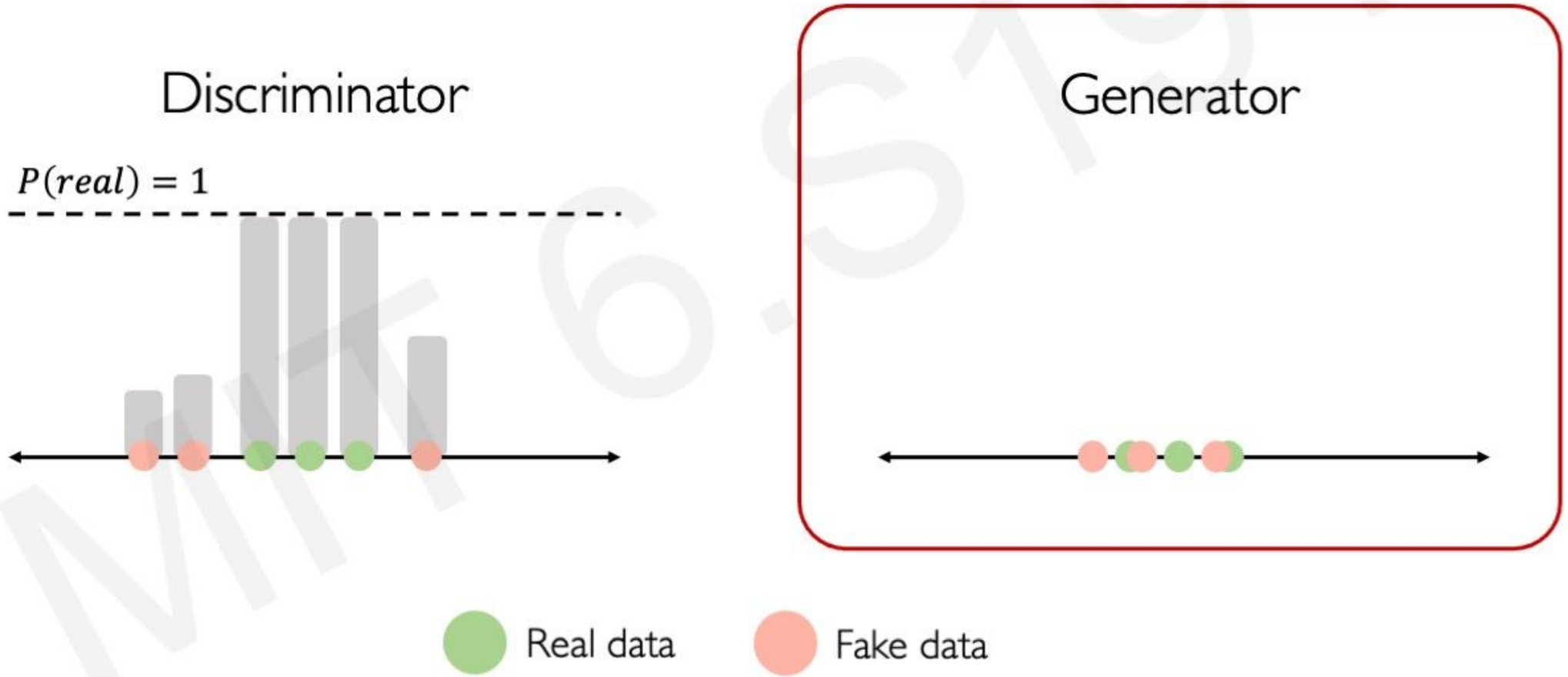
Intuition behind GANs

Generator tries to improve its imitation of the data.



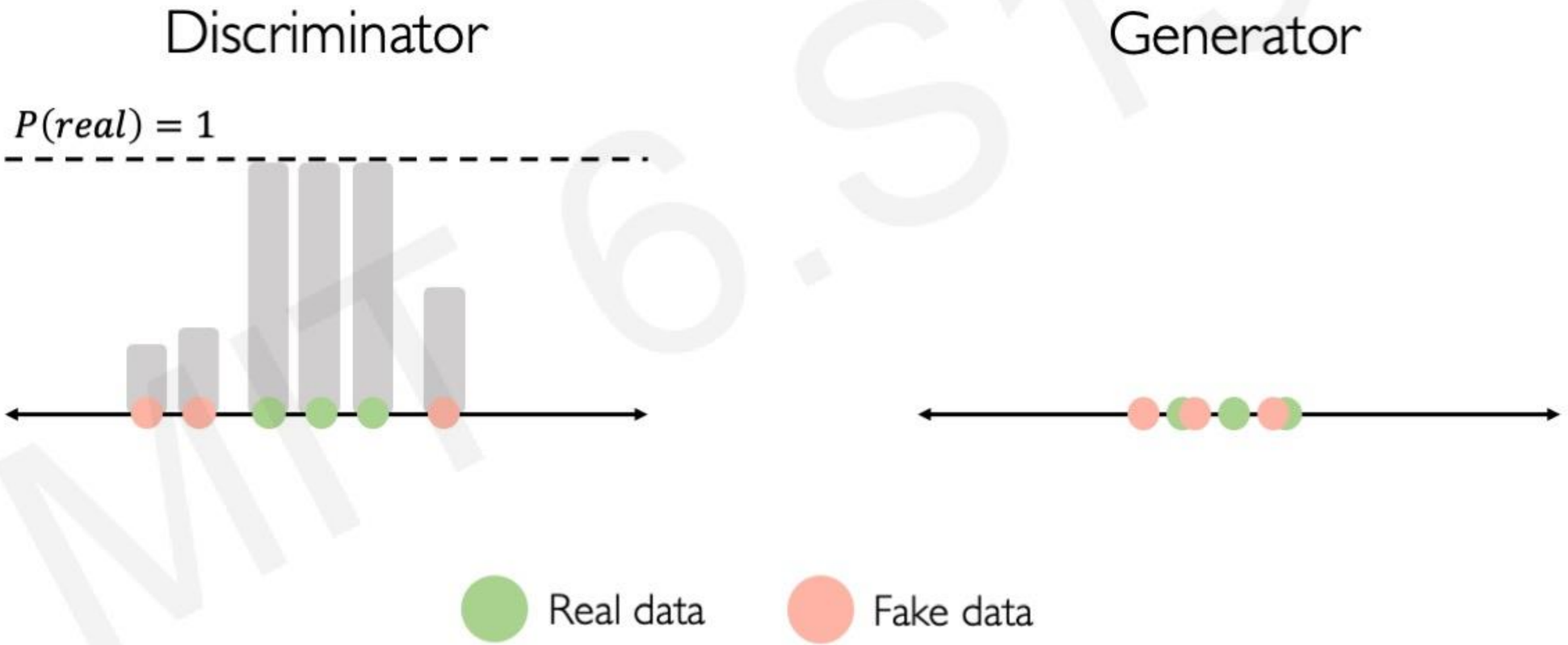
Intuition behind GANs

Generator tries to improve its imitation of the data.

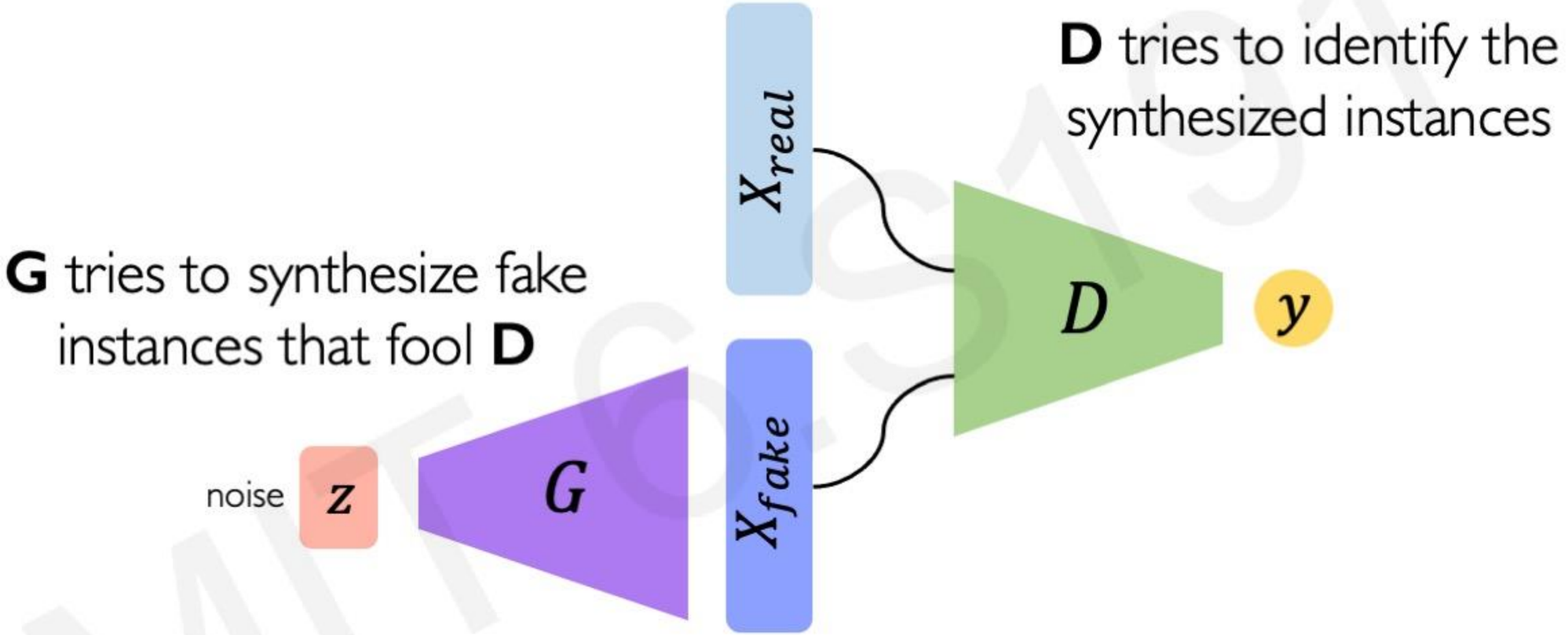


Intuition behind GANs

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



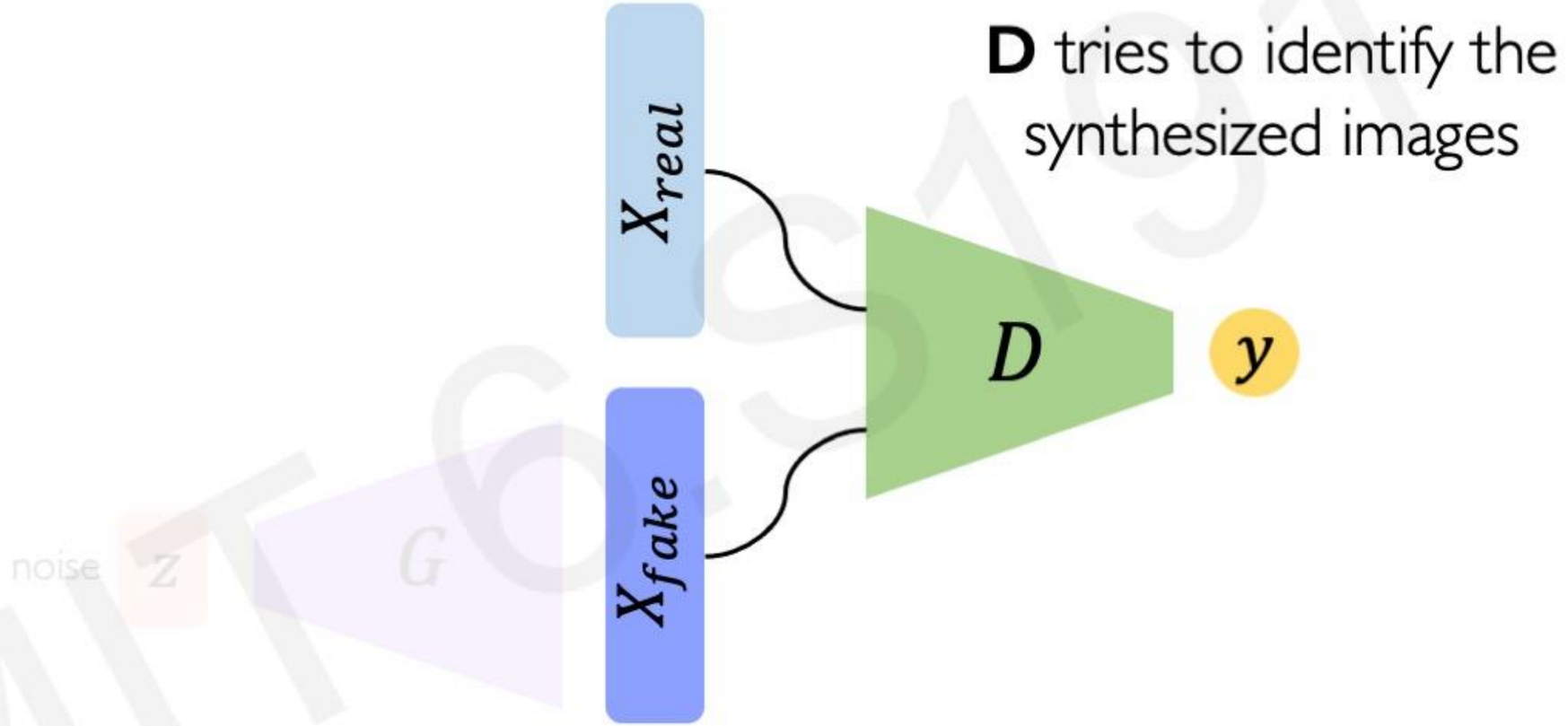
Training GANs



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

Global optimum: G reproduces the true data distribution

Training GANs: loss function



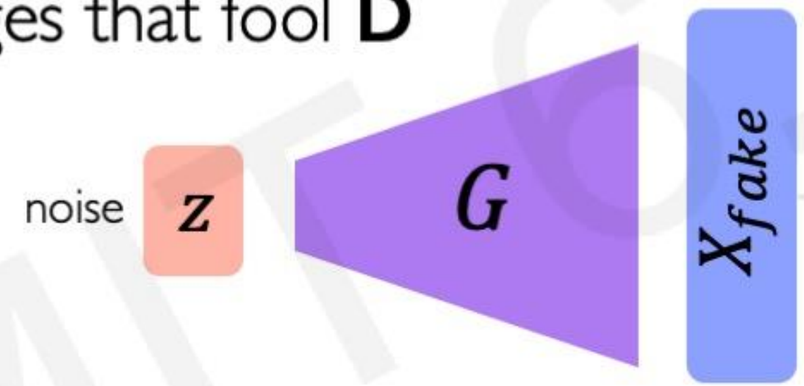
D tries to identify the synthesized images

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

Fake **Real**

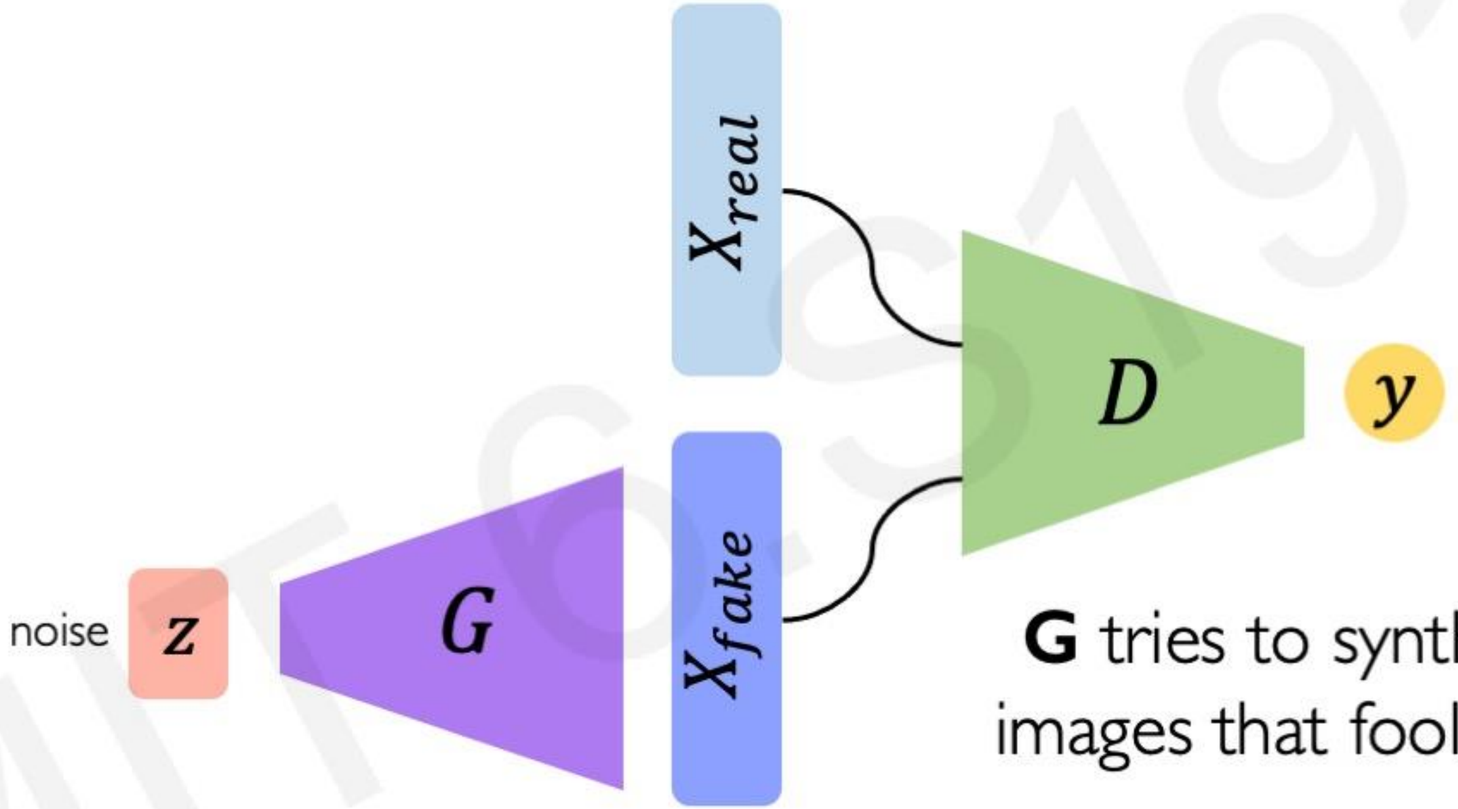
Training GANs: loss function

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



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

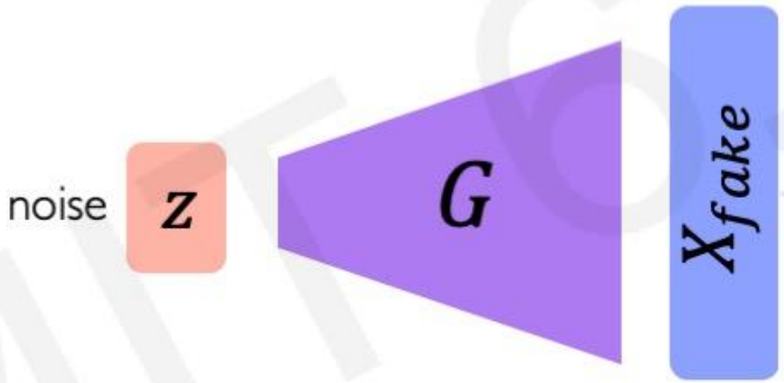
Training GANs: loss function



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

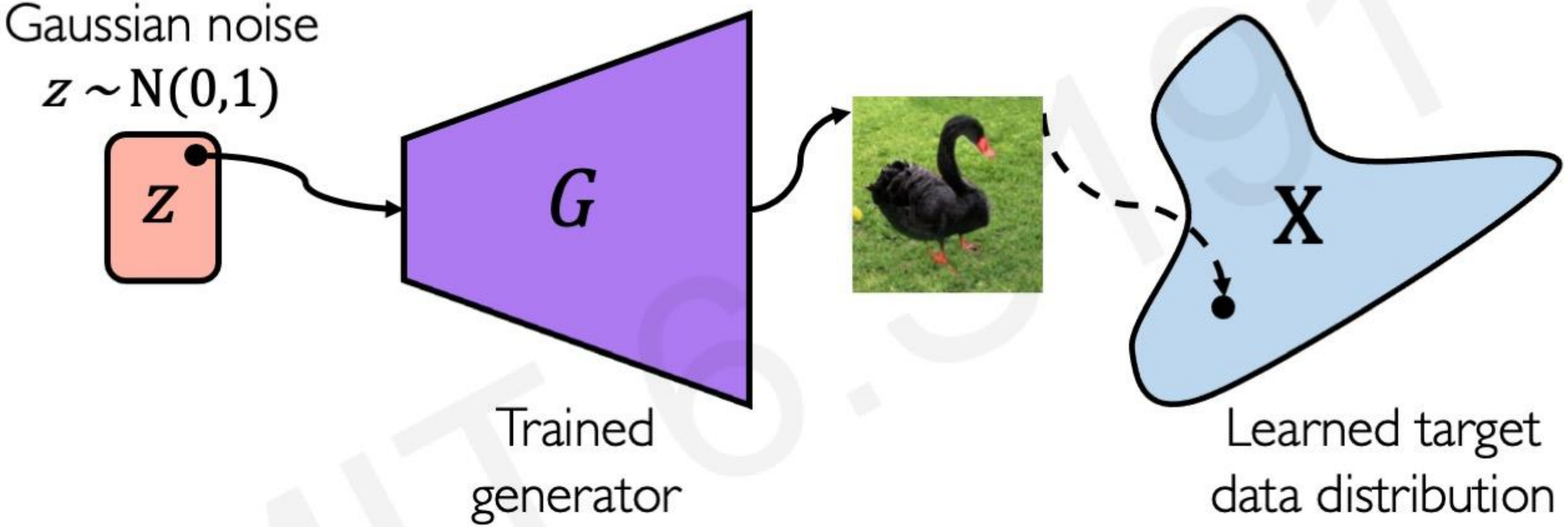
$$\arg\min_G \max_D \mathbb{E}_{\mathbf{z},\mathbf{x}}[\log D(G(\mathbf{z})) + \log(1 - D(\mathbf{x}))]$$

Generating new data with GANs

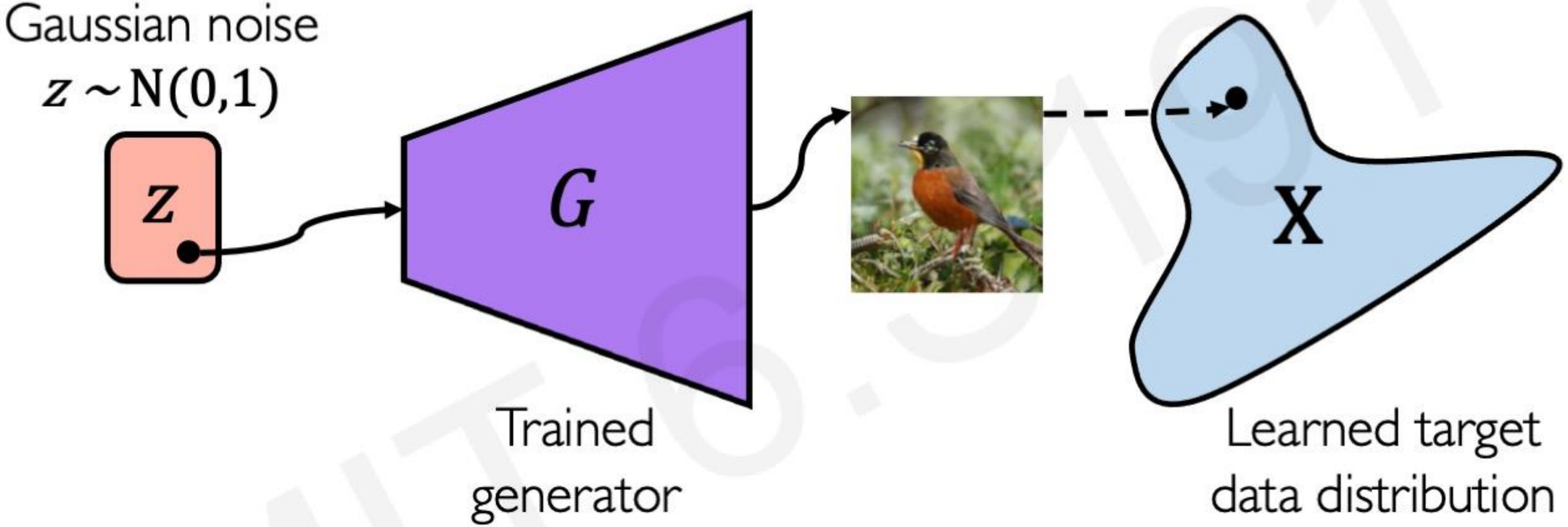


After training, use generator network to create **new data** that's never been seen before.

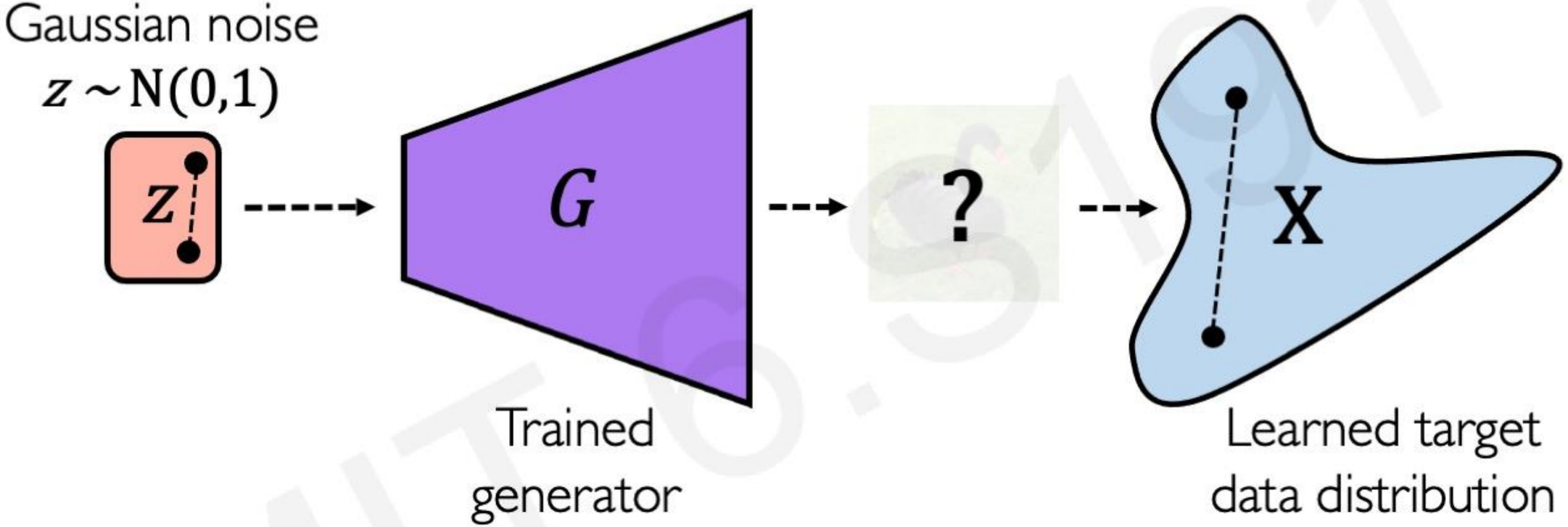
GANs are distribution transformers



GANs are distribution transformers

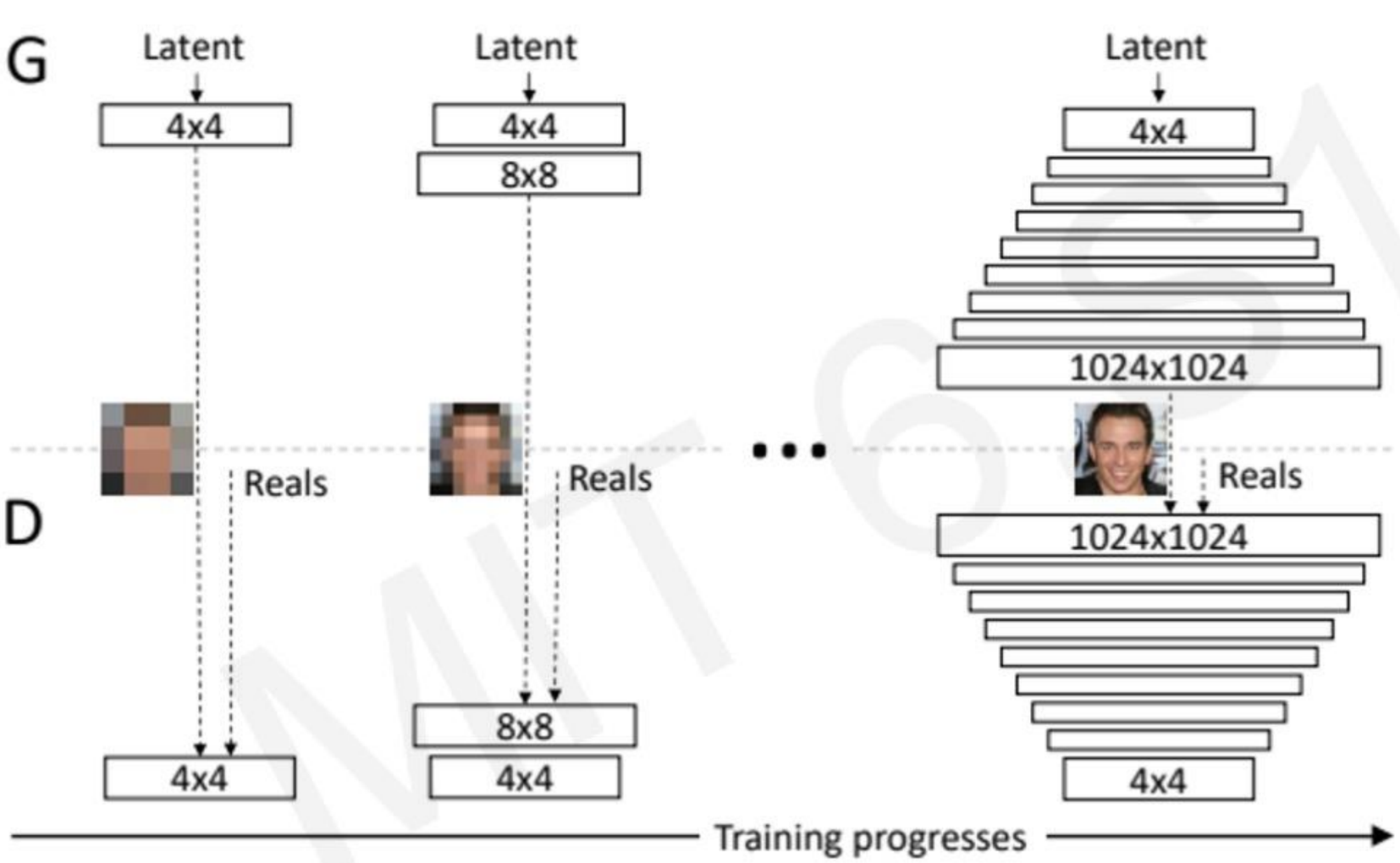


GANs are distribution transformers



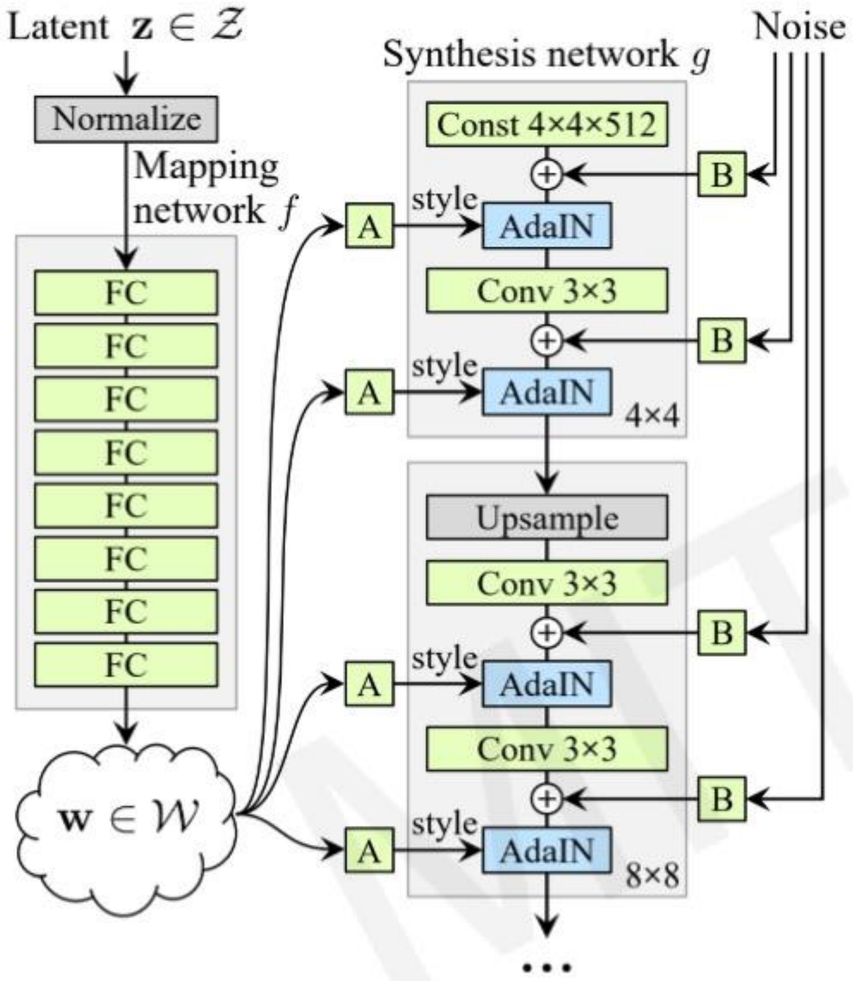
GANs: Recent Advances

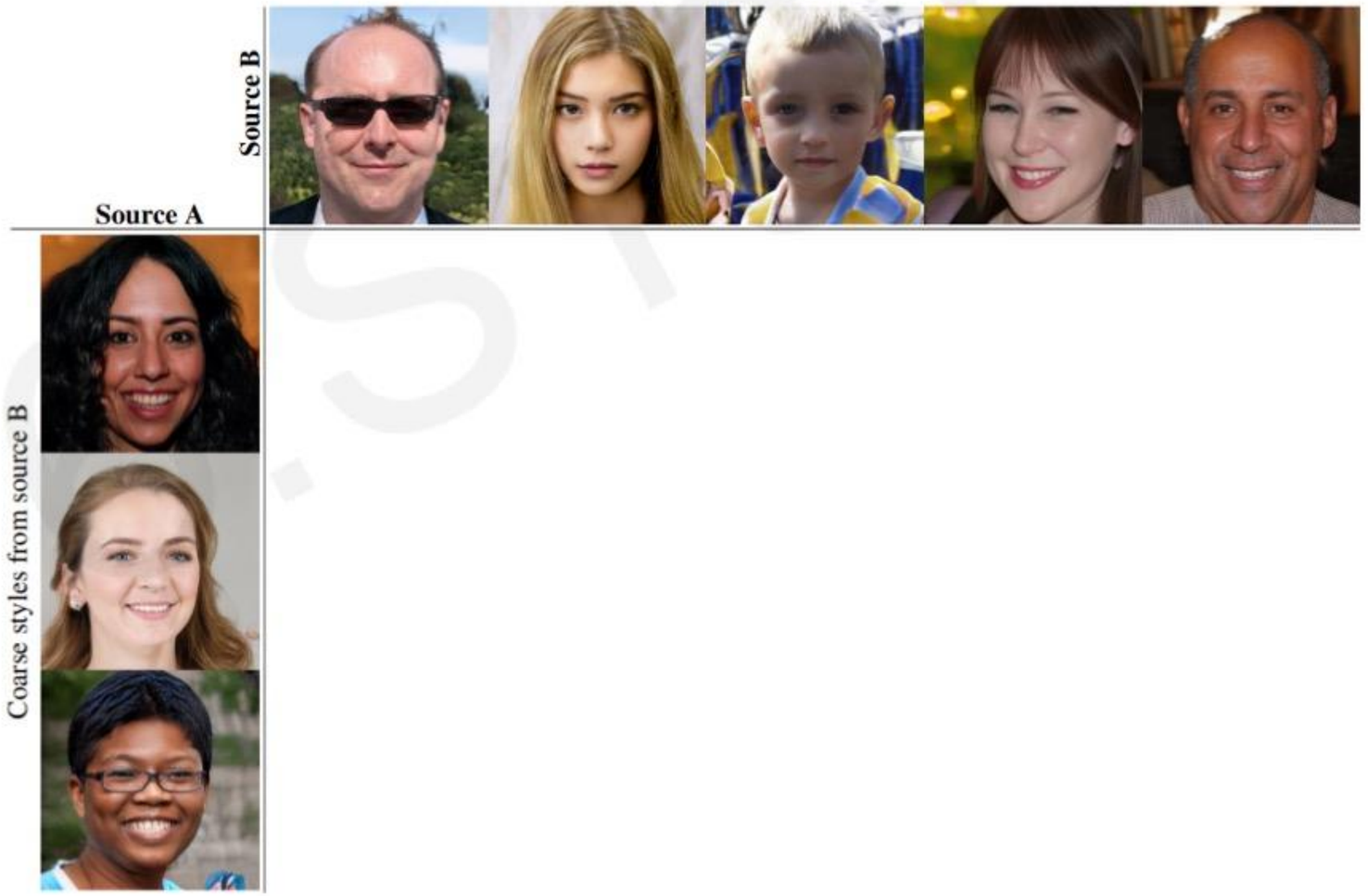
Progressive growing of GANs



Progressive growing of GANs: results

StyleGAN(2): progressive growing + style transfer





GANs for image synthesis: latest results

GANs for image synthesis: latest results

Applications of GANs

- Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
- Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks
- Generative Adversarial Text to Image Synthesis

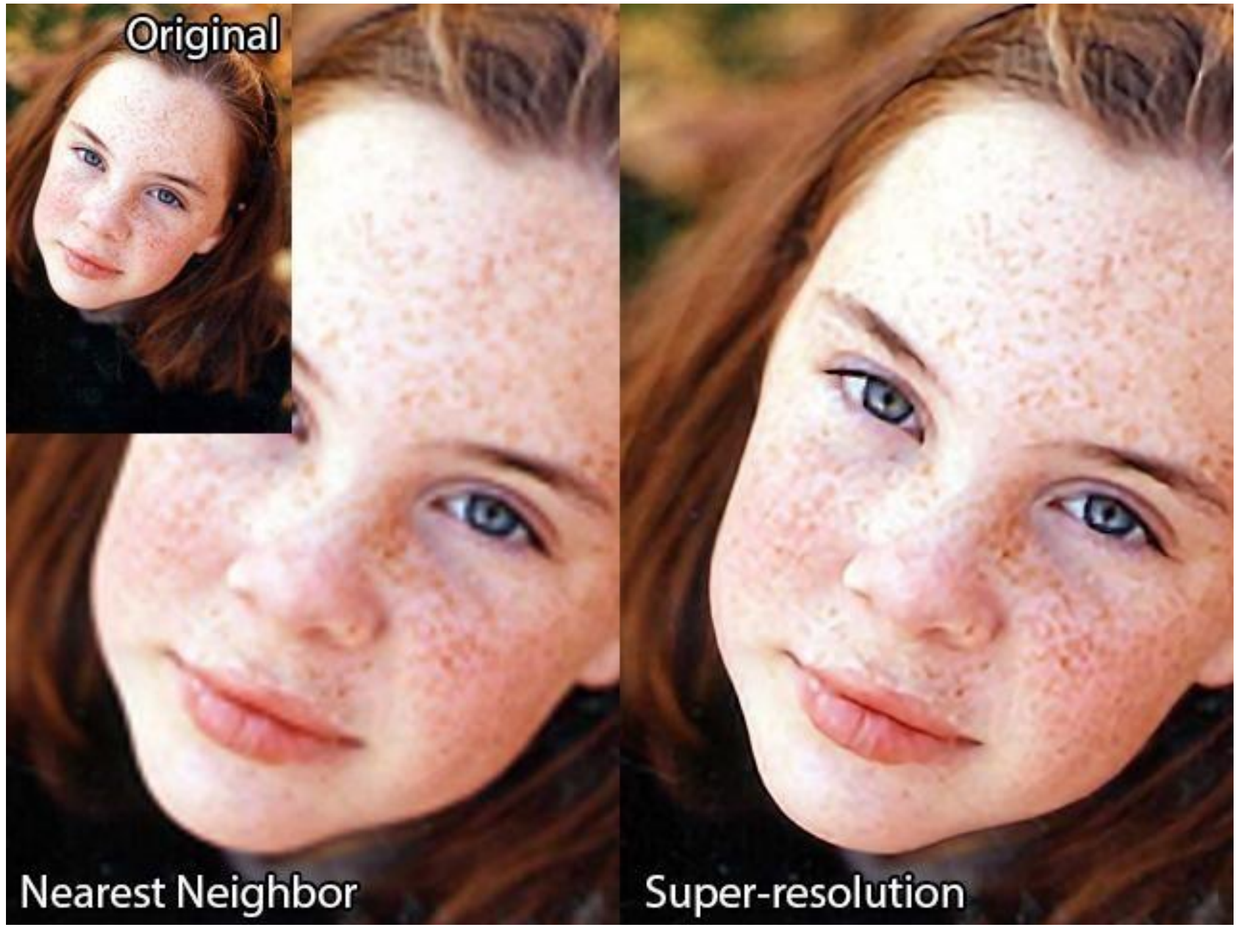
Using GANs for Single Image Super-Resolution

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi

Problem

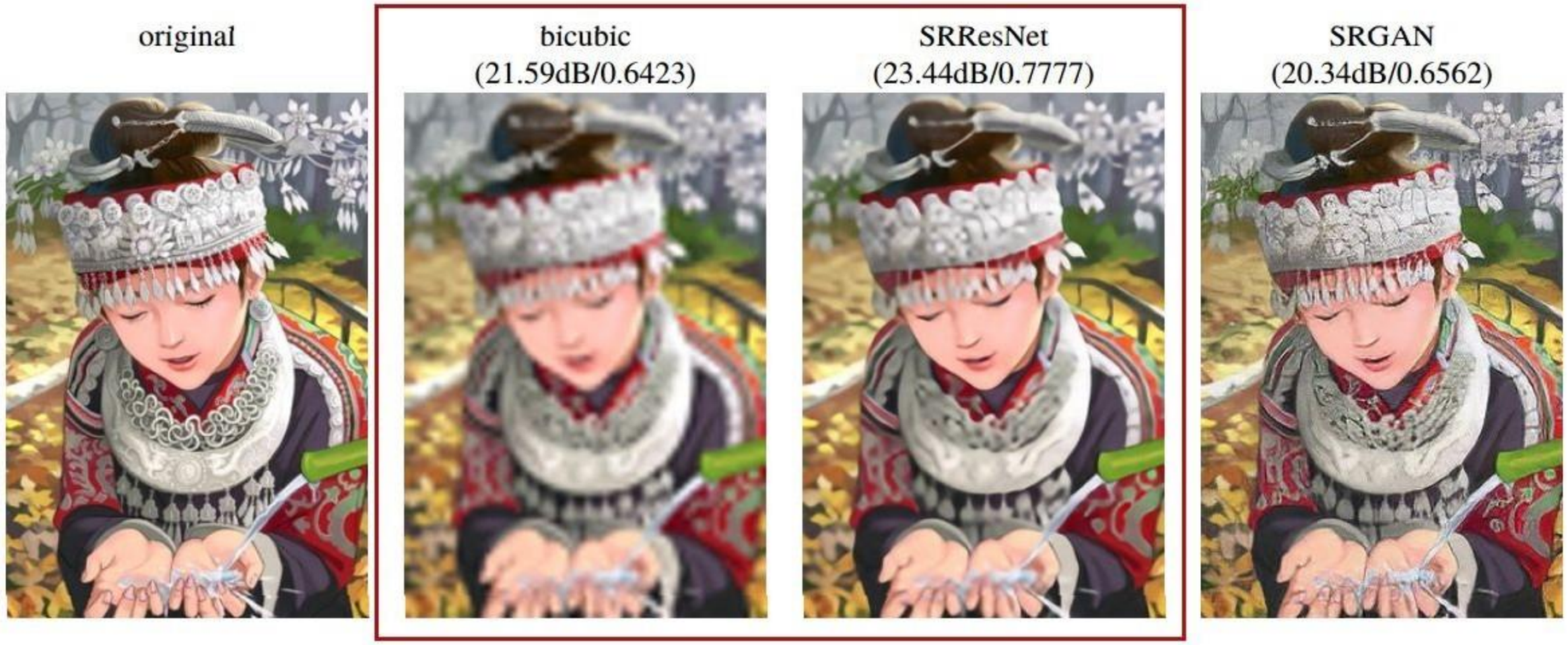
How do we get a high resolution (HR) image from just one (LR) lower resolution image?

Answer: We use super-resolution (SR) techniques.



http://www.extremetech.com/wp-content/uploads/2012/07/super-resolution-freckles.jpg

Previous Attempts



SRGAN


original
bicubic
(21.59dB/0.6423)
SRResNet
(23.44dB/0.7777)
SRGAN
(20.34dB/0.6562)

SRGAN - Generator

- G: generator that takes a low-res image I^{LR} and outputs its high-res counterpart I^{SR}
- θ_G : parameters of G
- l^{SR}: loss function measures the difference between the 2 high-res images

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR})$$

SRGAN - Discriminator

- D: discriminator that classifies whether a high-res image is I^{HR} or I^{SR}
- θ_D: parameters of D

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})} [\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})} [\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))]$$

SRGAN - Perceptual Loss Function

Loss is calculated as weighted combination of:

- ➔ Content loss
- ➔ Adversarial loss
- ➔ Regularization loss

SRGAN - Content Loss

Instead of MSE, use loss function based on ReLU layers of pre-trained VGG network. Ensures similarity of content.

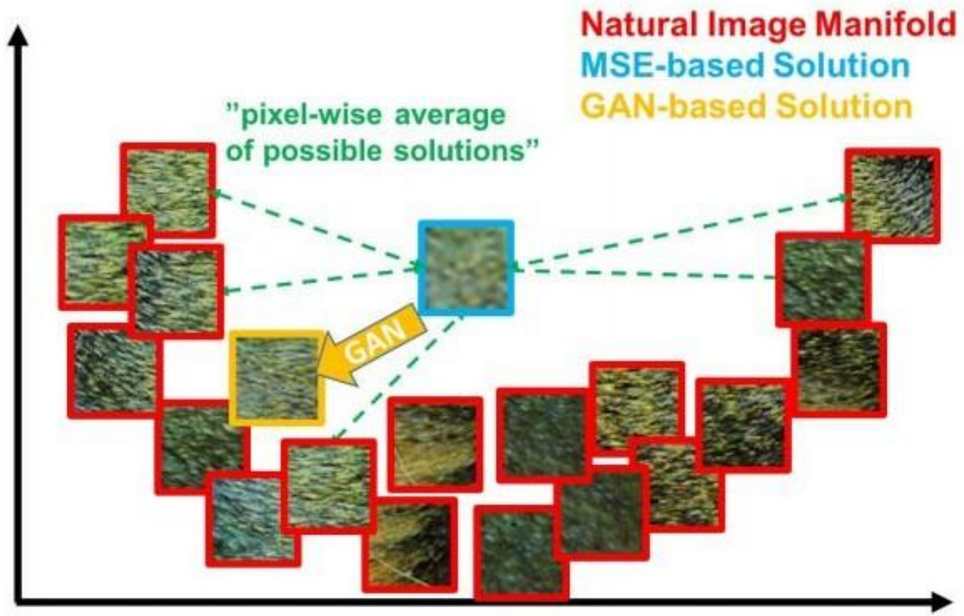
- $\Phi_{i,j}$: feature map of j^{th} convolution before i^{th} maxpooling
- $W_{i,j}$ and $H_{i,j}$: dimensions of feature maps in the VGG

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

SRGAN - Adversarial Loss

Encourages network to favour images that reside in manifold of natural images.

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$



SRGAN - Regularization Loss

Encourages spatially coherent solutions based on total variations.

$$l_{TV}^{SR} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} ||\nabla G_{\theta_G}(I^{LR})_{x,y}||$$

SRGAN - Examples

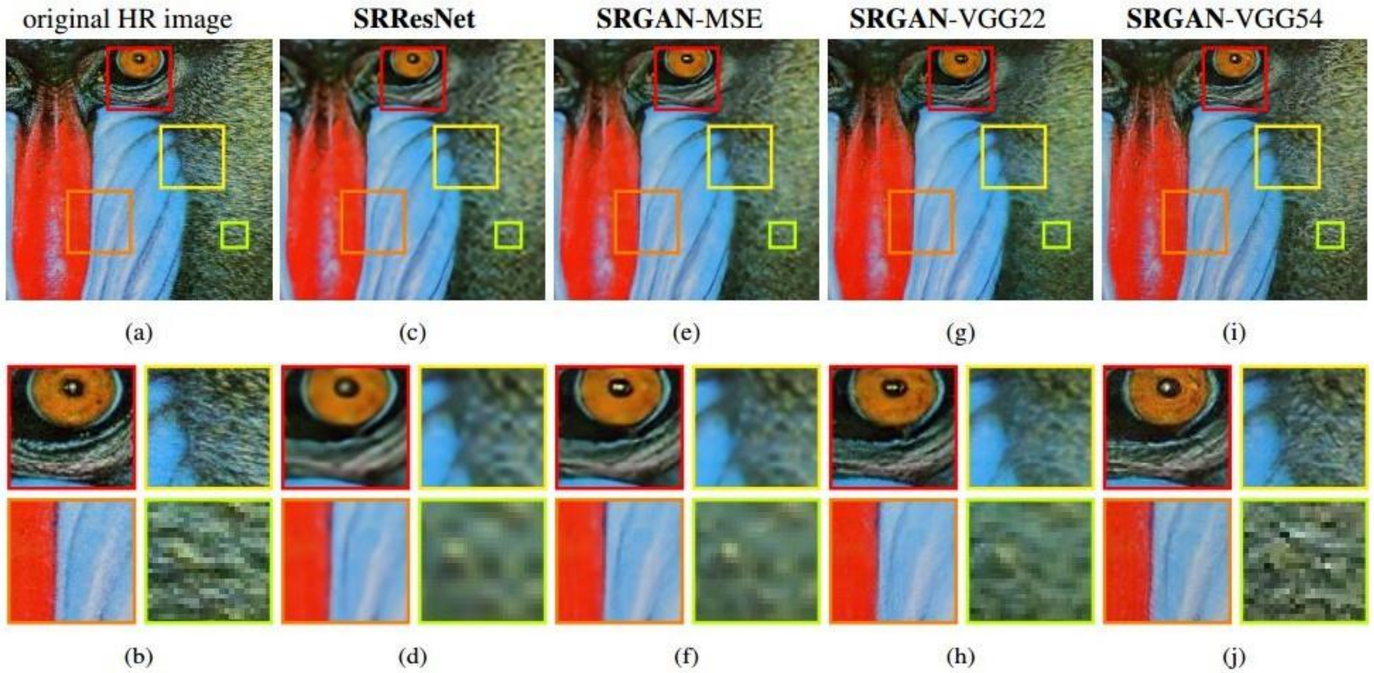
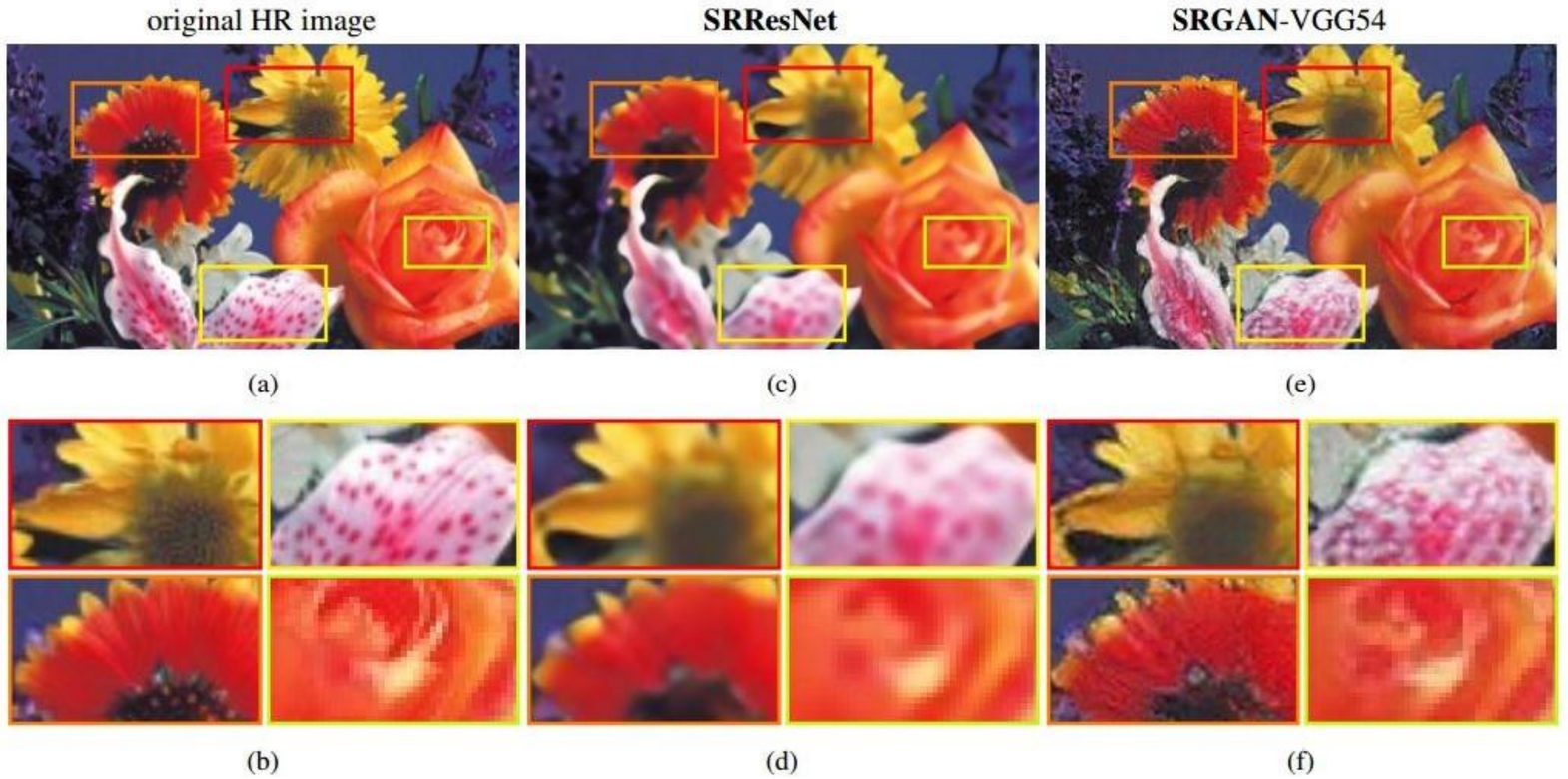


Figure 5: Reference HR image (left: a,b) with corresponding SRResNet (middle left: c,d), SRGAN-MSE (middle: e,f), SRGAN-VGG2.2 (middle right: g,h) and SRGAN-VGG54 (right: i,j) reconstruction results.

SRGAN - Examples



Generative Adversarial Text to Image Synthesis

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee

Author's code available at: https://github.com/reedscot/icml2016

Motivation

Current deep learning models enable us to...

- ➢ Learn feature representations of images & text
- ➢ Generate realistic images & text

- ☑ pull out images based on captions
- ☑ generate descriptions based on images
- ☑ answer questions about image content

"Two pizzas sitting on top of a stove top oven"

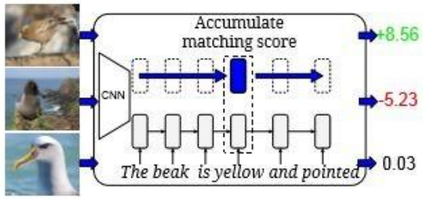
Problem - Multimodal distribution

- Many plausible image can be associated with one single text description

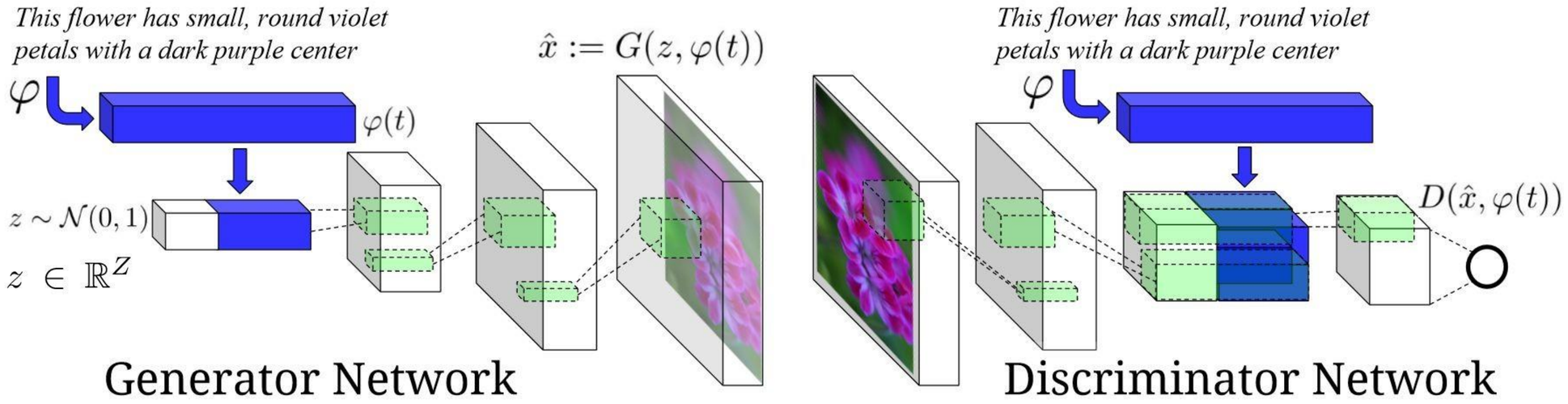
- Previous attempt uses Variational Recurrent Autoencoders to generate image from text caption but the images were not realistic enough. (Mansimov et al. 2016)

The Model - Basic CGAN

Pre-trained char-CNN-RNN



Learns a compatibility function of images and text -> joint embedding



$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z)))]$$

Training – Results: Flower & Bird

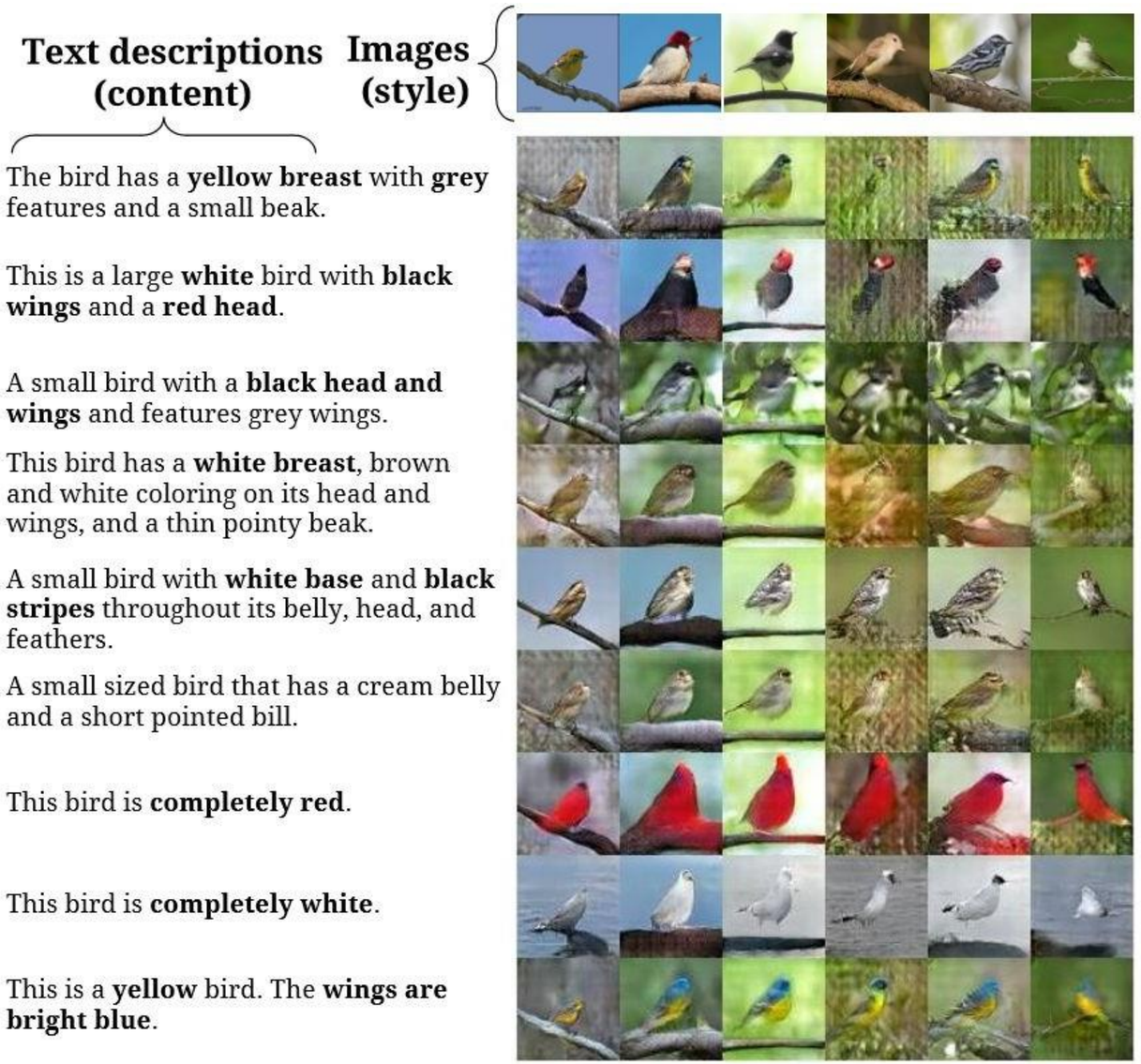
Training – Results: MS COCO

a large blue octopus kite flies above the people having fun at the beach.

a toilet in a small room with a window and unfinished walls.

a man in a wet suit riding a surfboard on a wave.

Training – Results Style disentangling



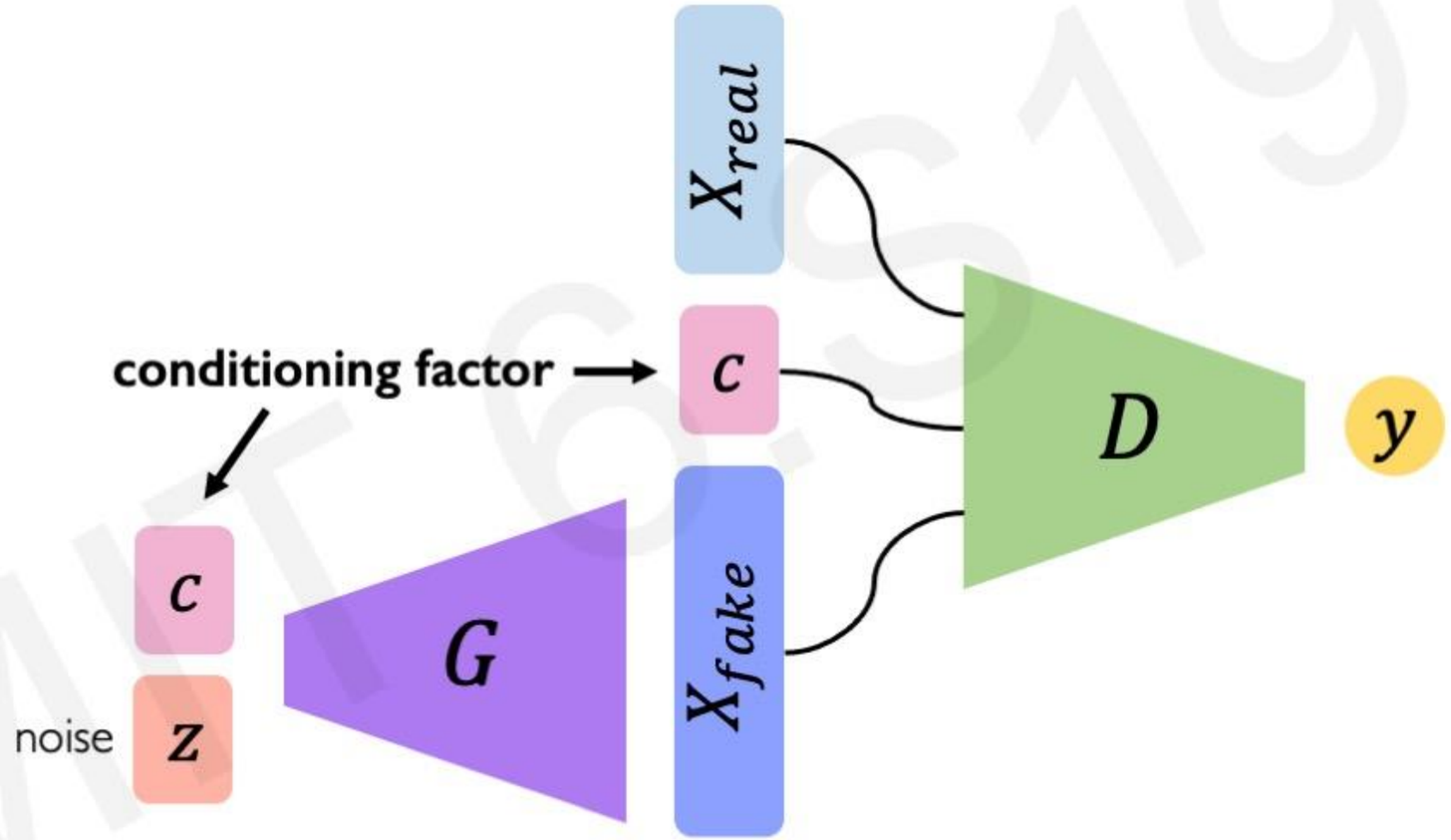
$$s \leftarrow S(x)$$

$$\hat{x} \leftarrow G(s, \varphi(t))$$

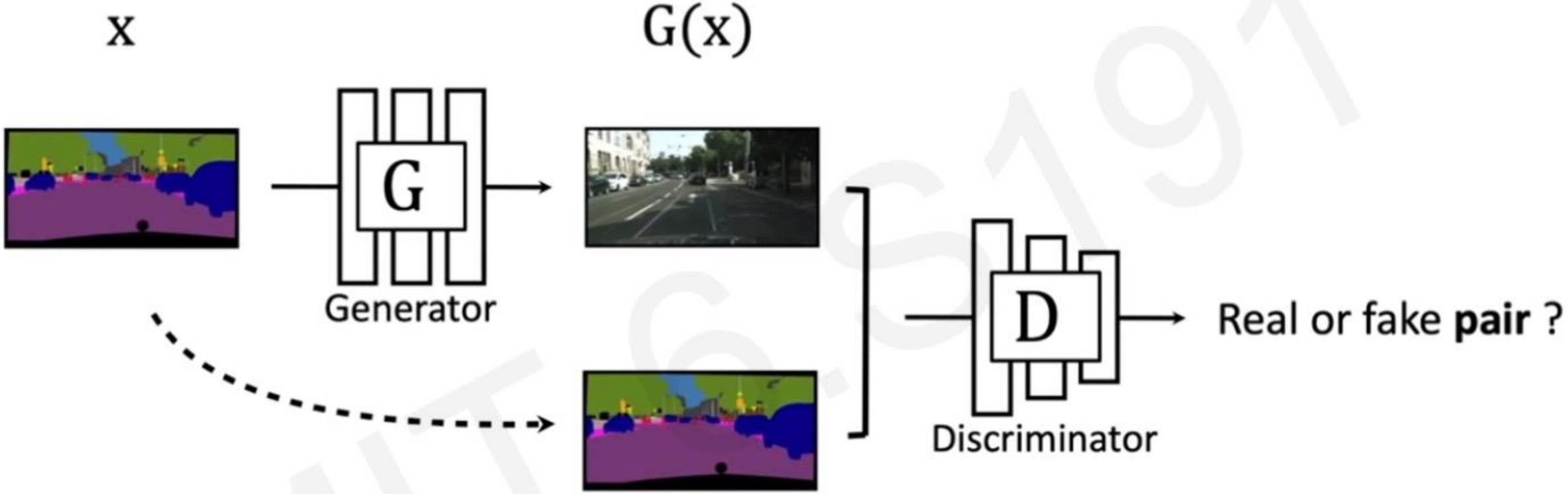
Image-to-Image translations

Conditional GANs

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

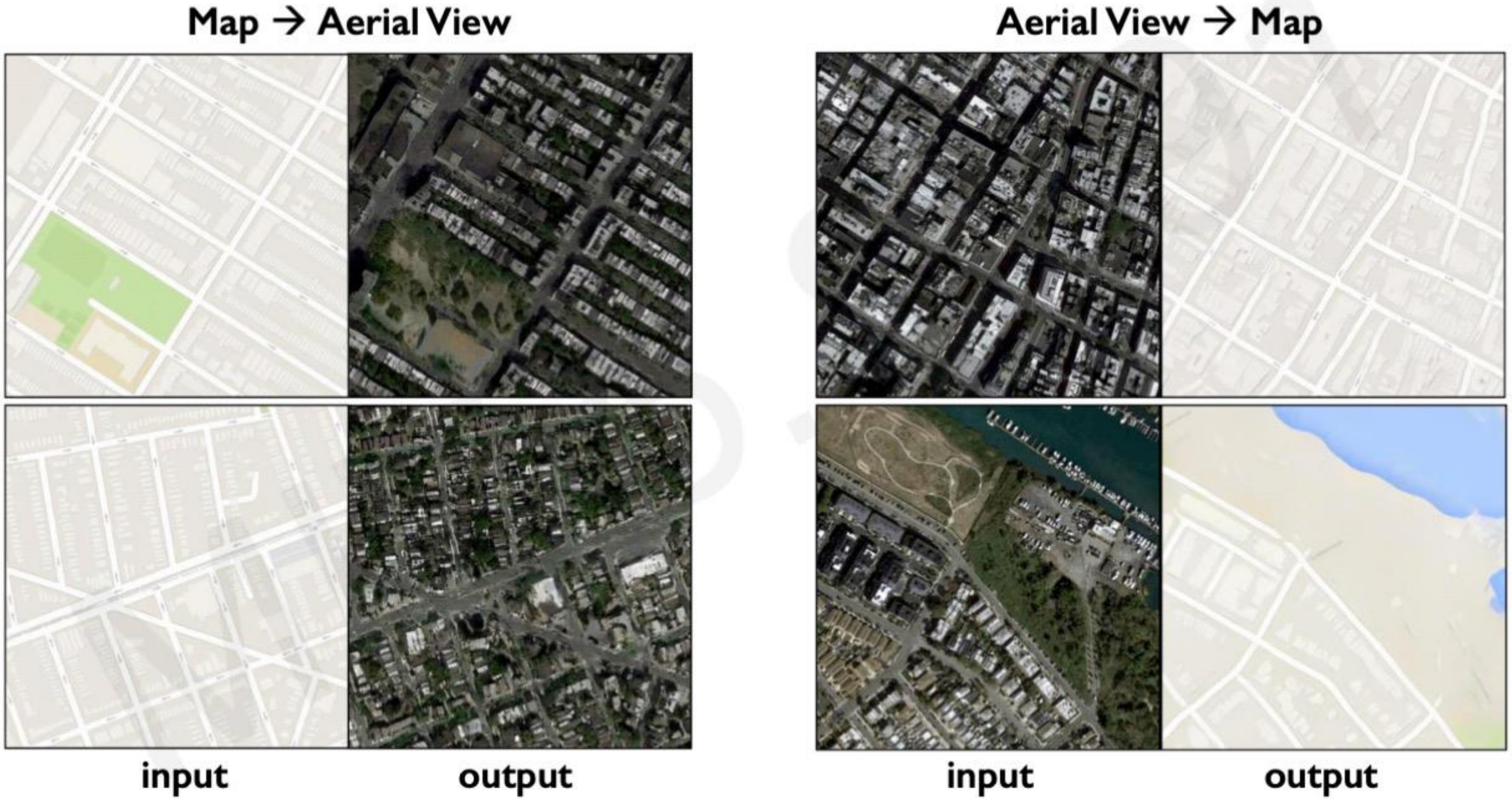


Conditional GANs and pix2pix: paired translation



The discriminator, D, classifies between fake and real **pairs**.
The generator, G, learns to fool the discriminator.

Paired translation: results



Isola et al. (2017) use a GAN-like setup to address this issue for the translation of images with pixel-to-pixel correspondence:

- edges to realistic photos,
- semantic segmentation,
- gray-scales to colors, etc.

P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. CVPR 2017

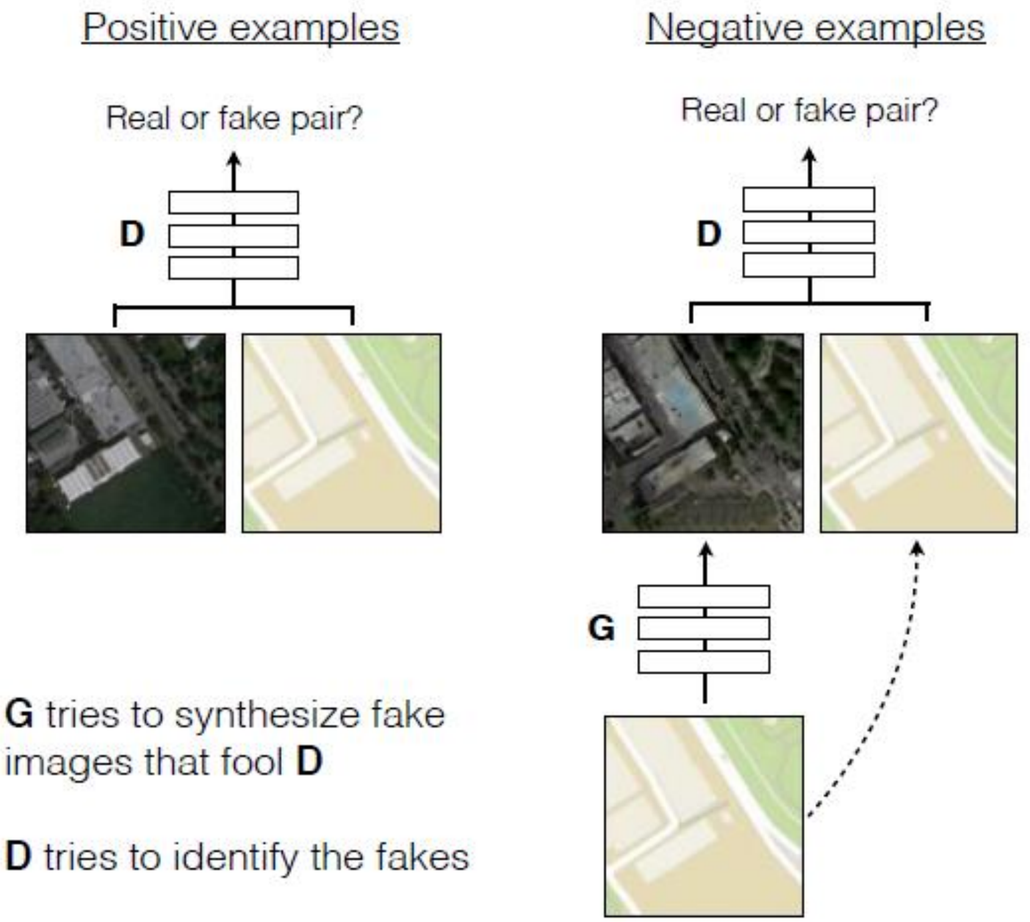


Figure 2: Training a conditional GAN to predict aerial photos from maps. The discriminator, D, learns to classify between real and synthesized pairs. The generator learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe an input image.

They define

$$V(\mathbf{D},\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu}\Big[\log \mathbf{D}(Y,X)\Big] + \mathbb{E}_{Z\sim\mu_Z, X\sim\mu_X}\Big[\log(1-\mathbf{D}(\mathbf{G}(Z,X),X))\Big],$$

$$\mathscr{L}_{L^1}(\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu, Z\sim\mathcal{N}(0,I)}\Big[\,\|Y-\mathbf{G}(Z,X)\|_1\,\Big],$$

and

$$\mathbf{G}^* = \operatorname*{argmin}_{\mathbf{G}} \max_{\mathbf{D}} \; V(\mathbf{D},\mathbf{G}) + \lambda \mathscr{L}_{L^1}(\mathbf{G}).$$

The term $\mathscr{L}_{L^1}$ pushes toward proper pixel-wise prediction, and V makes the generator prefer realistic images to better fitting pixel-wise.

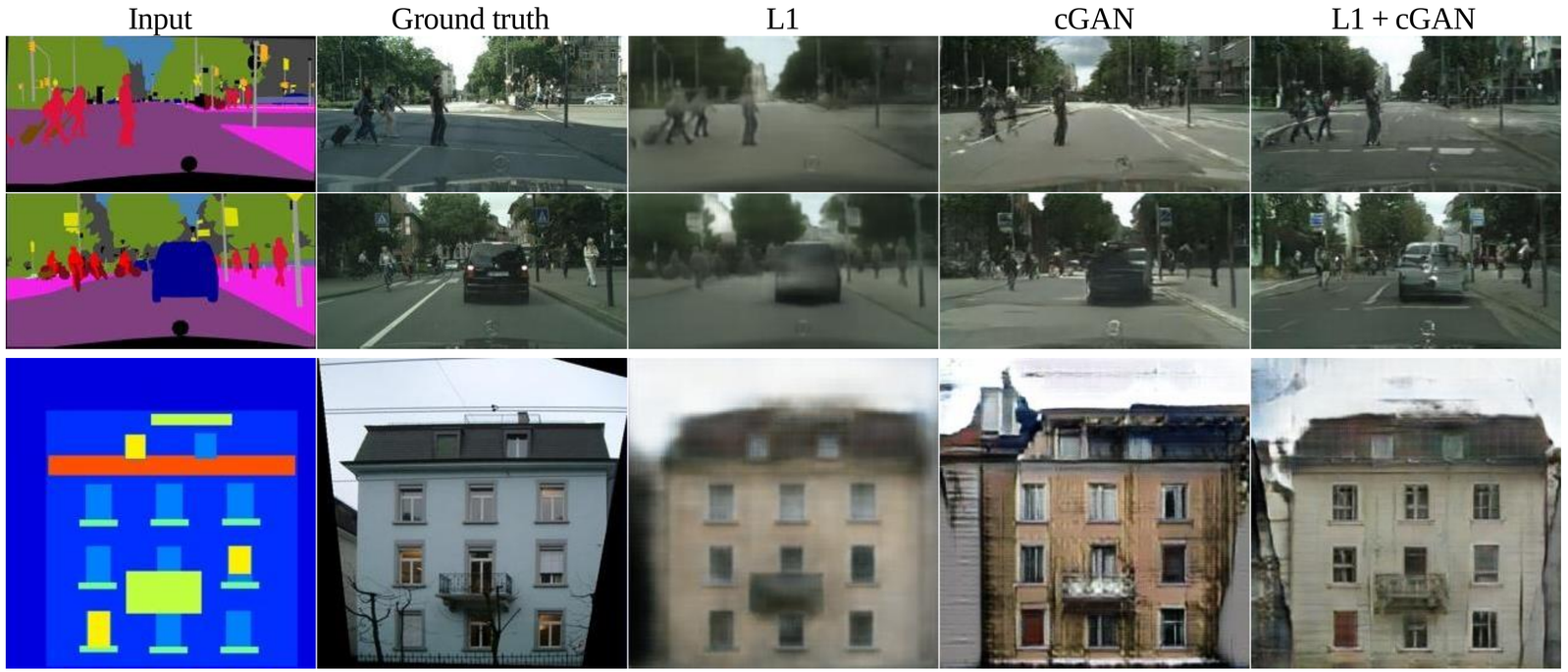


Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see https://phillipi.github.io/pix2pix/ for additional examples.

Figure 6: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70x70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (coforfulness) dimensions. The full 256x256 ImageGAN produces results that are visually similar to the 70x70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 2). Please see `https://phillipi.github.io/pix2pix/` for additional examples.

Figure 8: Example results on Google Maps at 512x512 resolution (model was trained on images at 256x256 resolution, and run convolu- tionally on the larger images at test time). Contrast adjusted for clarity.

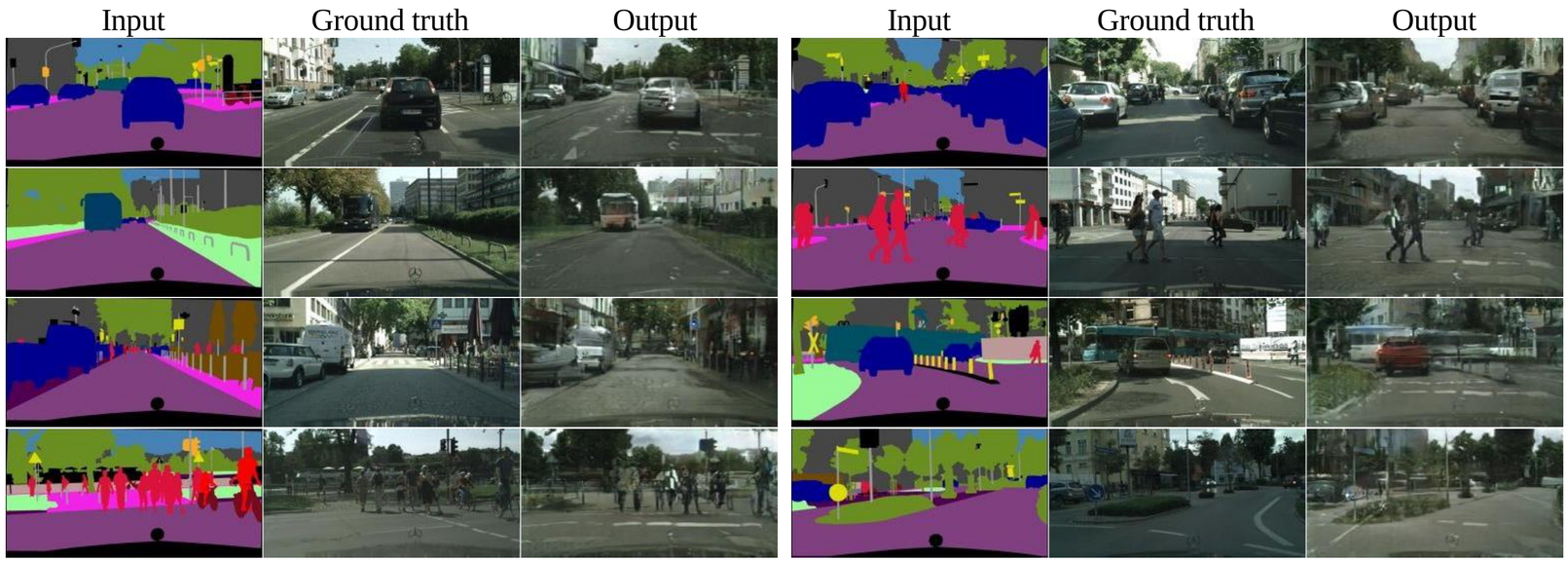


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.

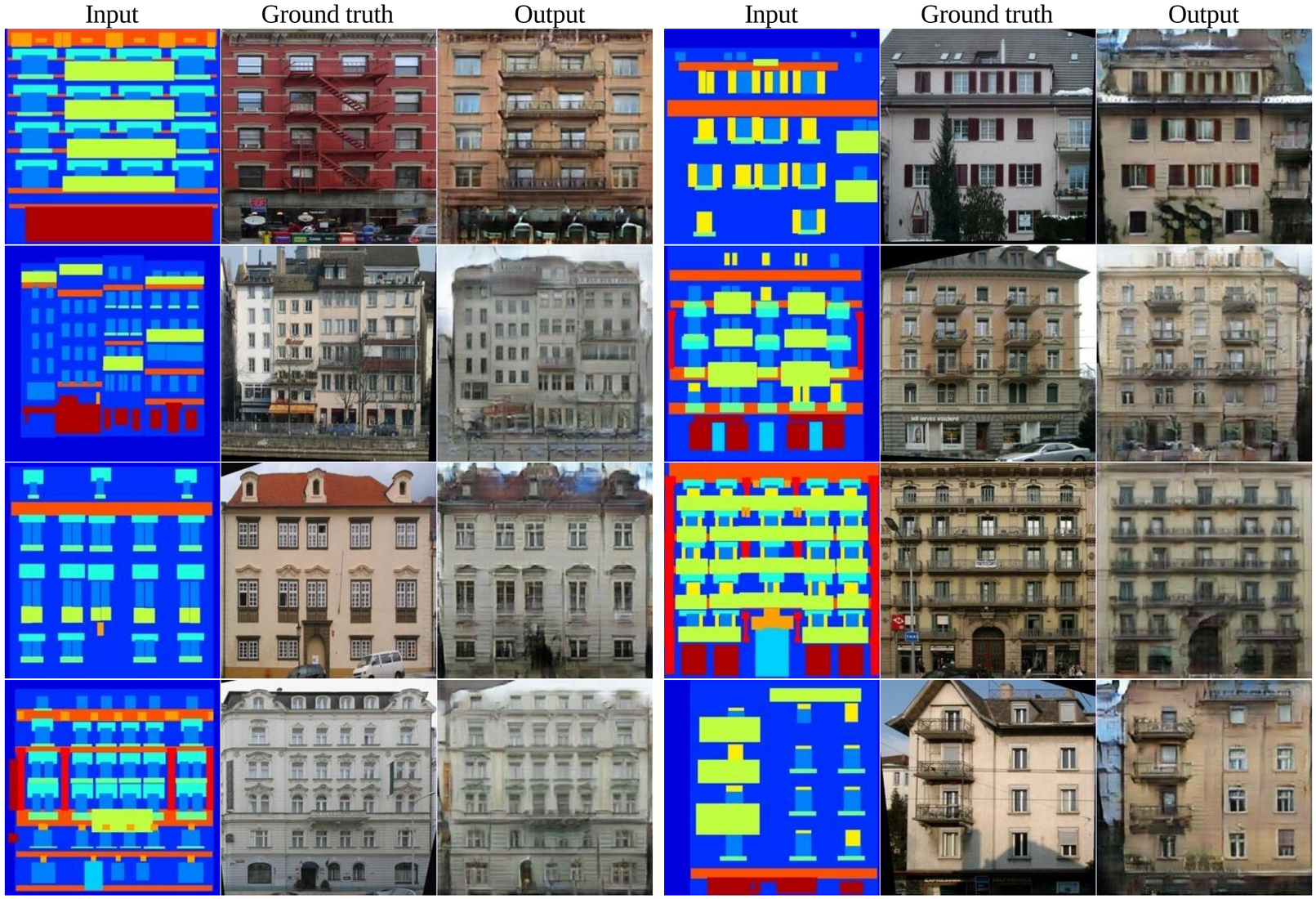


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

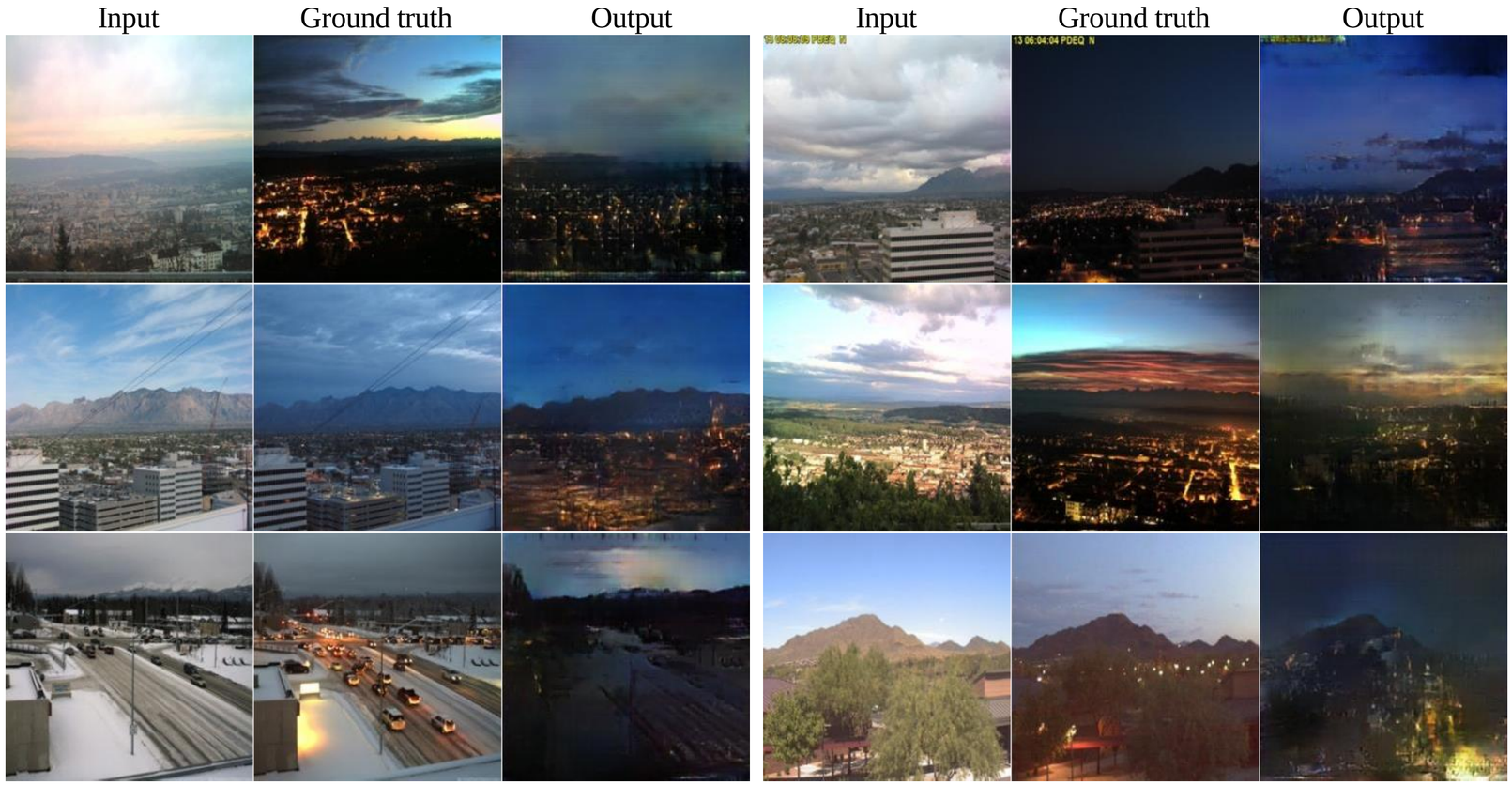


Figure 13: Example results of our method on day→night, compared to ground truth.

Figure 14: Example results of our method on automatically detected edges→handbags, compared to ground truth.

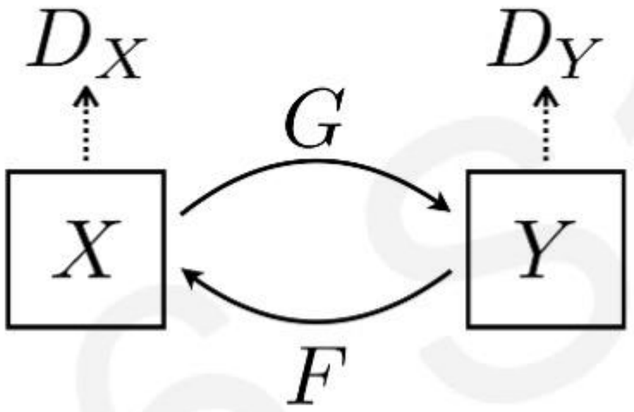
Figure 16: Example results of the edges→photo models applied to human-drawn sketches from [10]. Note that the models were trained on automatically detected edges, but generalize to human drawings

The main drawback of this technique is that it requires pairs of samples with pixel-to-pixel correspondence.

In many cases, one has at its disposal examples from two densities and wants to translate a sample from the first (“images of apples”) into a sample likely under the second (“images of oranges”).

CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.

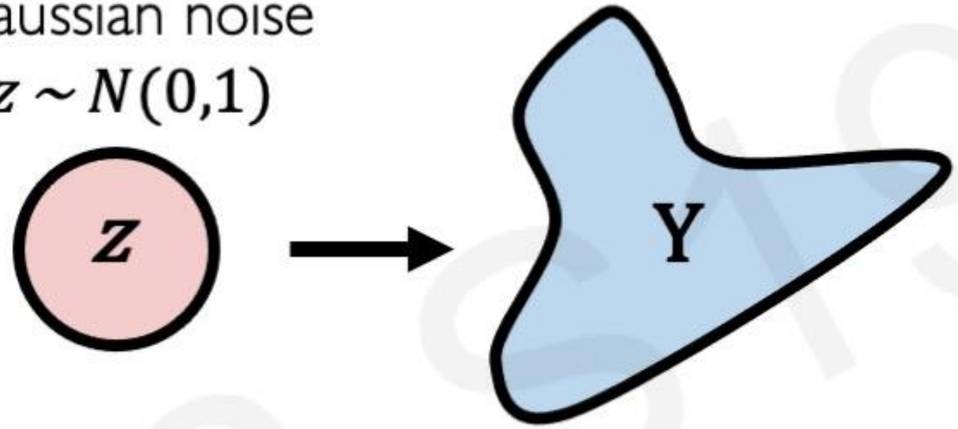


Distribution transformations

GANs:

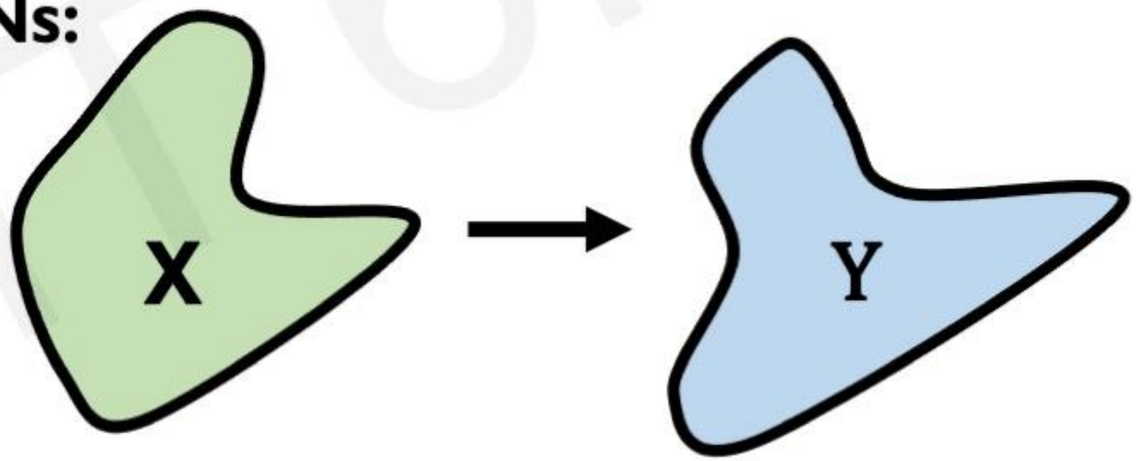
Gaussian noise

$z \sim N(0,1)$



Gaussian noise → target data manifold

CycleGANs:



data manifold **X** → data manifold **Y**

CycleGAN

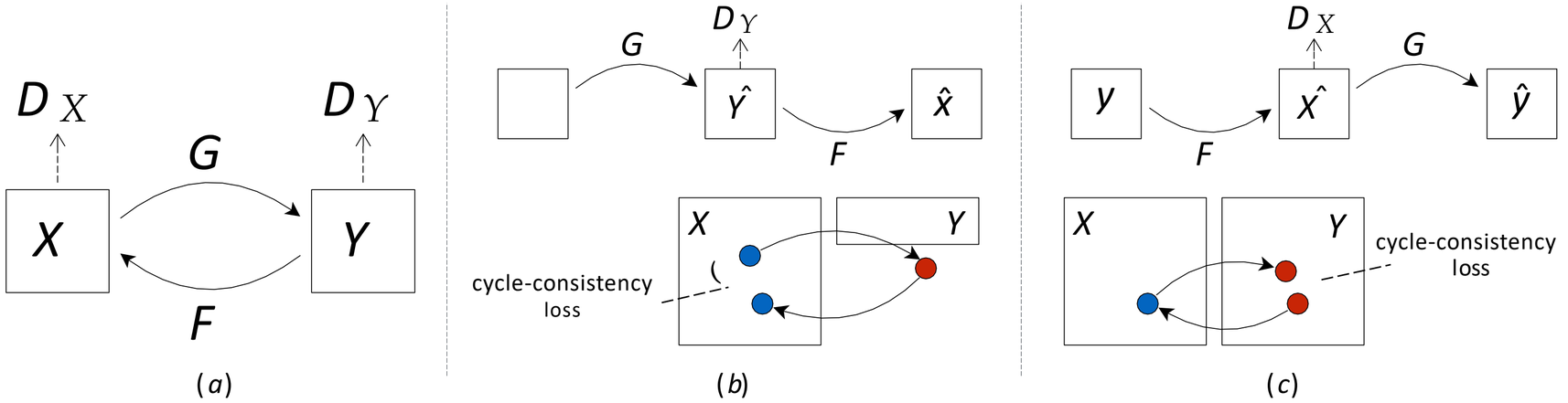
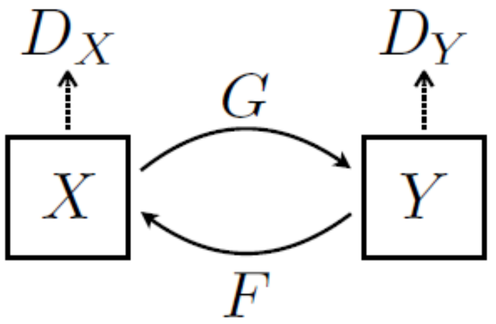


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

(Zhu et al., 2017)

J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. ICCV 2017.



The loss optimized alternatively is

$$V^*(\mathbf{G}, \mathbf{F}, \mathbf{D}_X, \mathbf{D}_Y) = V(\mathbf{G}, \mathbf{D}_Y, X, Y) + V(\mathbf{F}, \mathbf{D}_X, Y, X) + \lambda \left(\mathbb{E}\Big[\|\mathbf{F}(\mathbf{G}(X)) - X\|_1 \Big] + \mathbb{E}\Big[\|\mathbf{G}(\mathbf{F}(Y)) - Y\|_1 \Big] \right)$$

where V is a quadratic loss, instead of the usual log (Mao et al., 2016)

$$V(\mathbf{G}, \mathbf{D}_Y, X, Y) = \mathbb{E}\Big[(\mathbf{D}_Y(Y) - 1)^2 \Big] + \mathbb{E}\Big[\mathbf{D}_Y(\mathbf{G}(X))^2 \Big].$$

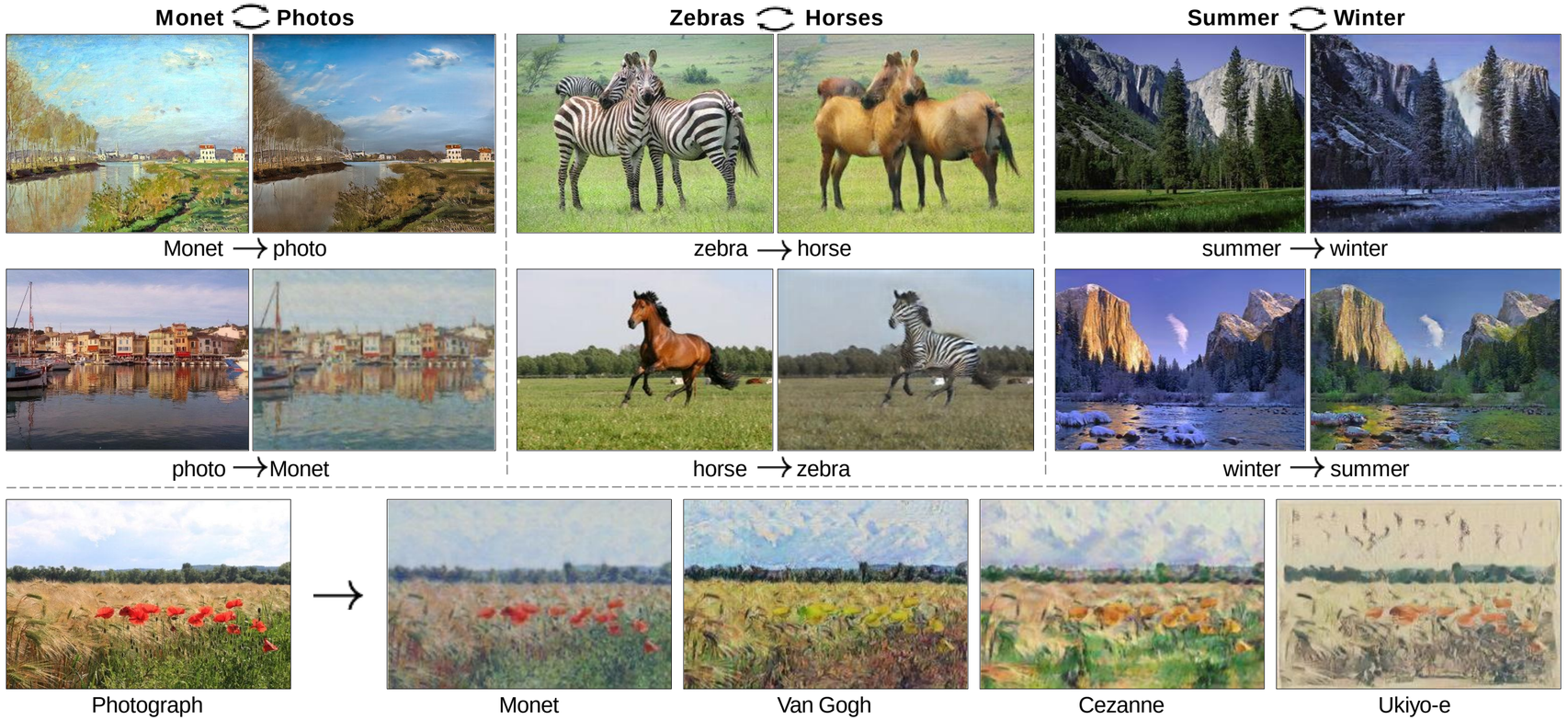


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

(Zhu et al., 2017)

zebra → horse

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

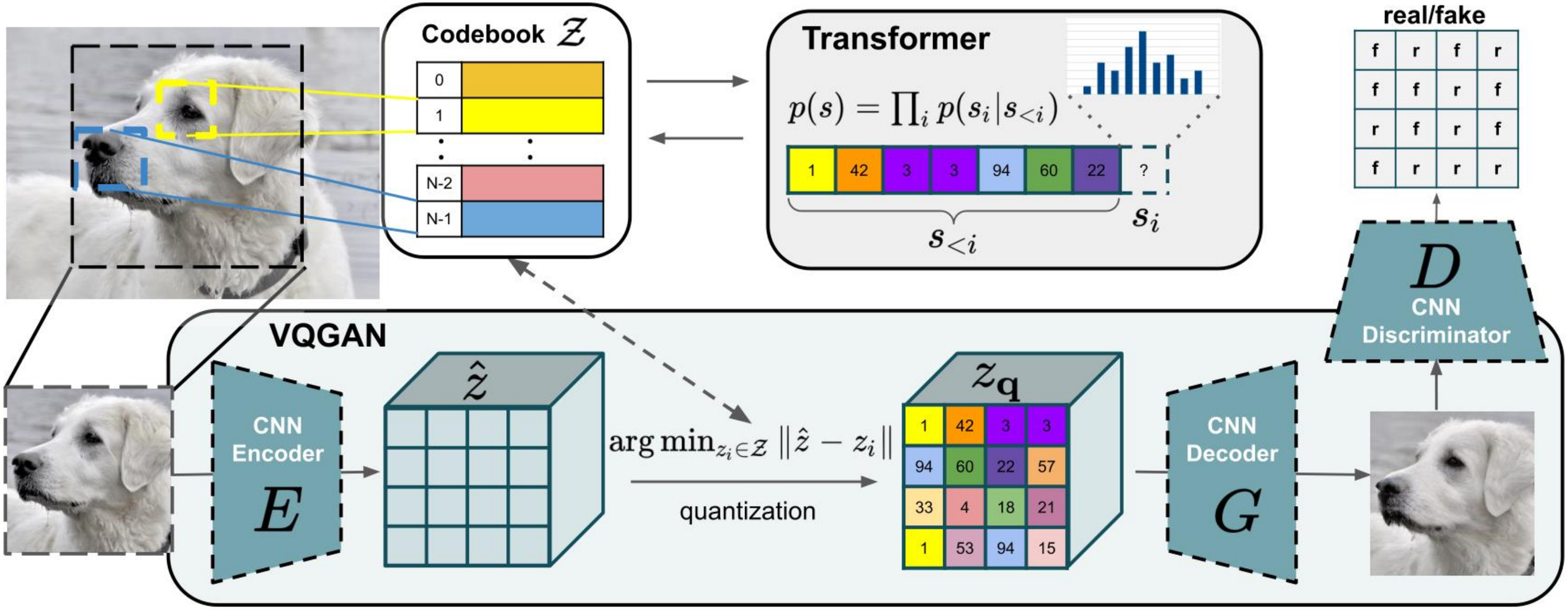
(Zhu et al., 2017)

apple → orange

orange → apple

(Zhu et al., 2017)

TAMING TRANSFORMERS FOR HIGH-RESOLUTION IMAGE SYNTHESIS (A.K.A #VQGAN)



https://compvis.github.io/taming-transformers/

The End