

COMP547

DEEP UNSUPERVISED LEARNING

Lecture #8 - #9 – Generative Adversarial Networks



KOÇ
UNIVERSITY

Aykut Erdem // Koç University // Spring 2022

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, **StyleGAN**, **StyleGAN2**, **StyleGAN3**, StyleGAN-XL, VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

StyleGAN

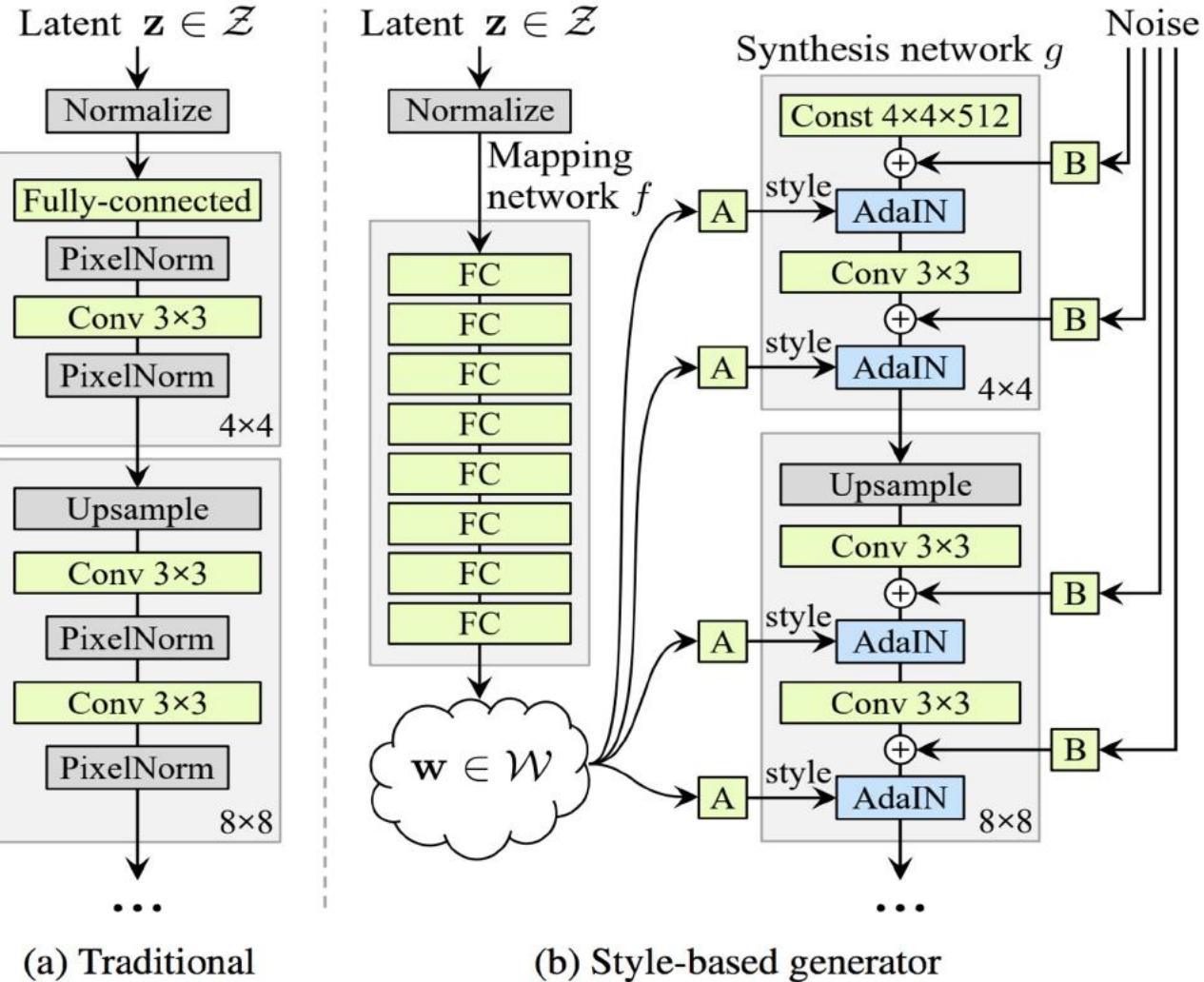
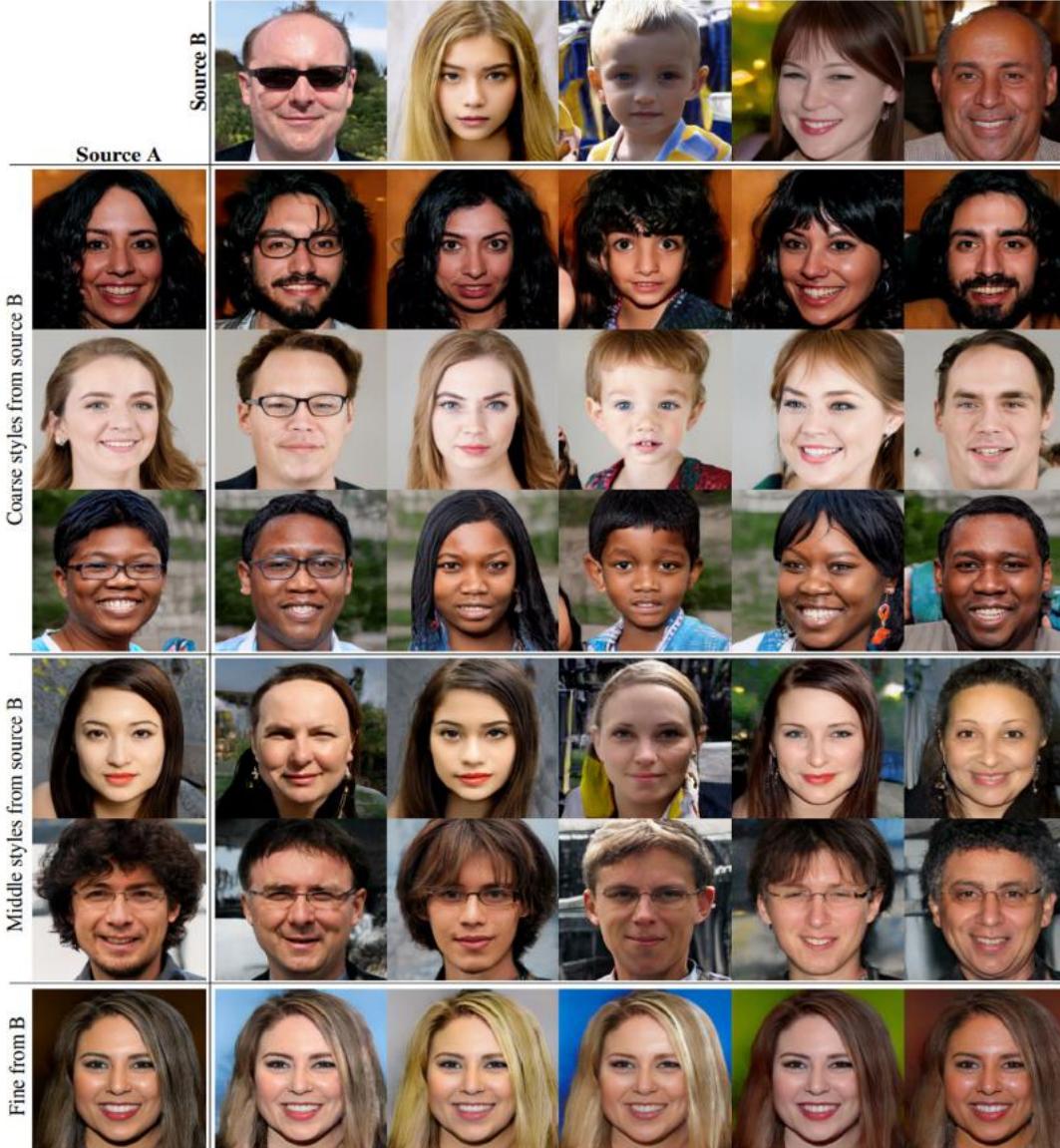


Figure 1. While a traditional generator [30] feeds the latent code through the input layer only, we first map the input to an intermediate latent space \mathcal{W} , which then controls the generator through adaptive instance normalization (AdaIN) at each convolution layer. Gaussian noise is added after each convolution, before evaluating the nonlinearity. Here “A” stands for a learned affine transform, and “B” applies learned per-channel scaling factors to the noise input. The mapping network f consists of 8 layers and the synthesis network g consists of 18 layers — two for each resolution ($4^2 - 1024^2$). The output of the last layer is converted to RGB using a separate 1×1 convolution, similar to Karras et al. [30]. Our generator has a total of 26.2M trainable parameters, compared to 23.1M in the traditional generator.

StyleGAN - Adaptive Instance Norm

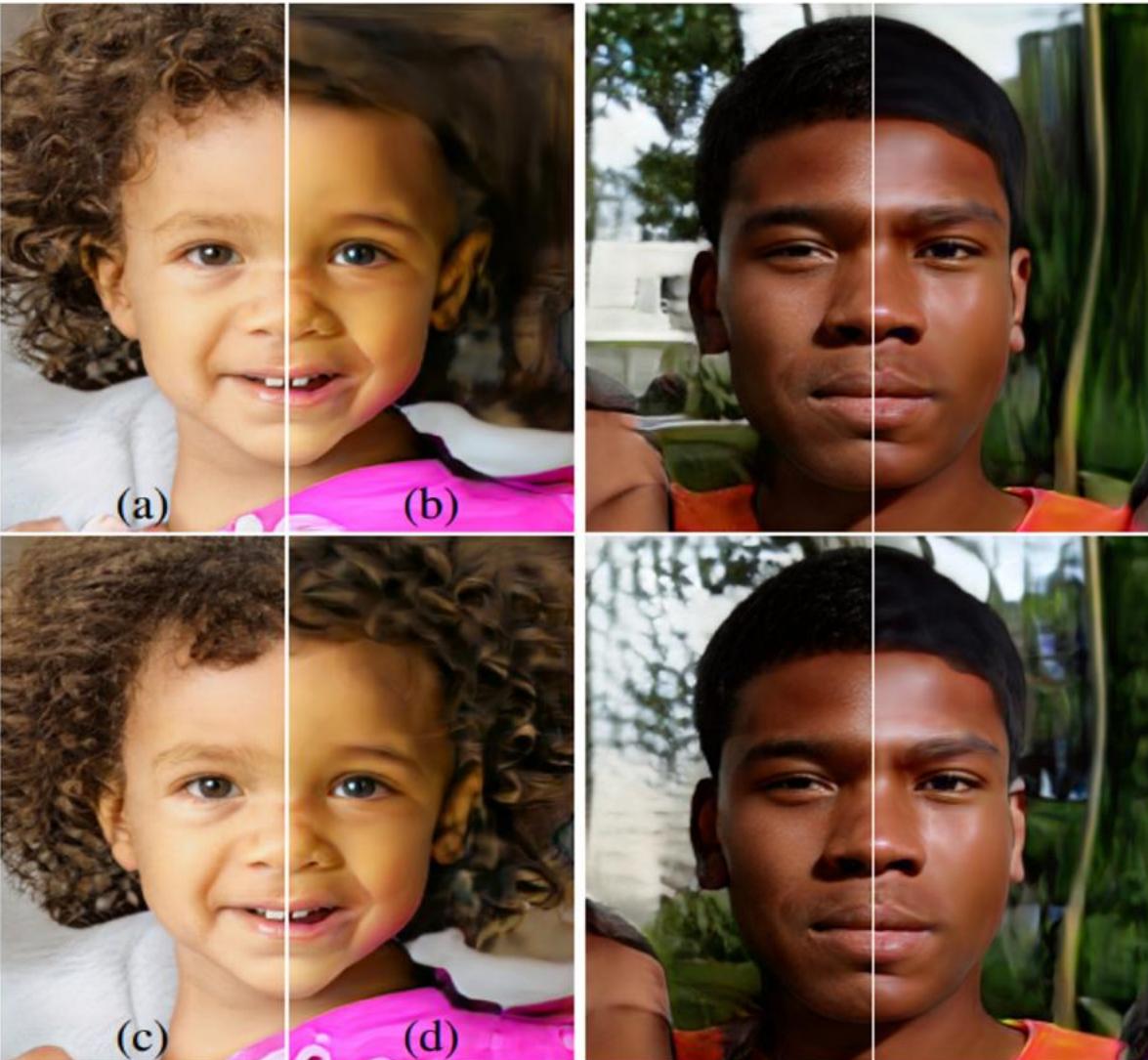
$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

StyleGAN - Style Transfer





StyleGAN - Effect of adding noise



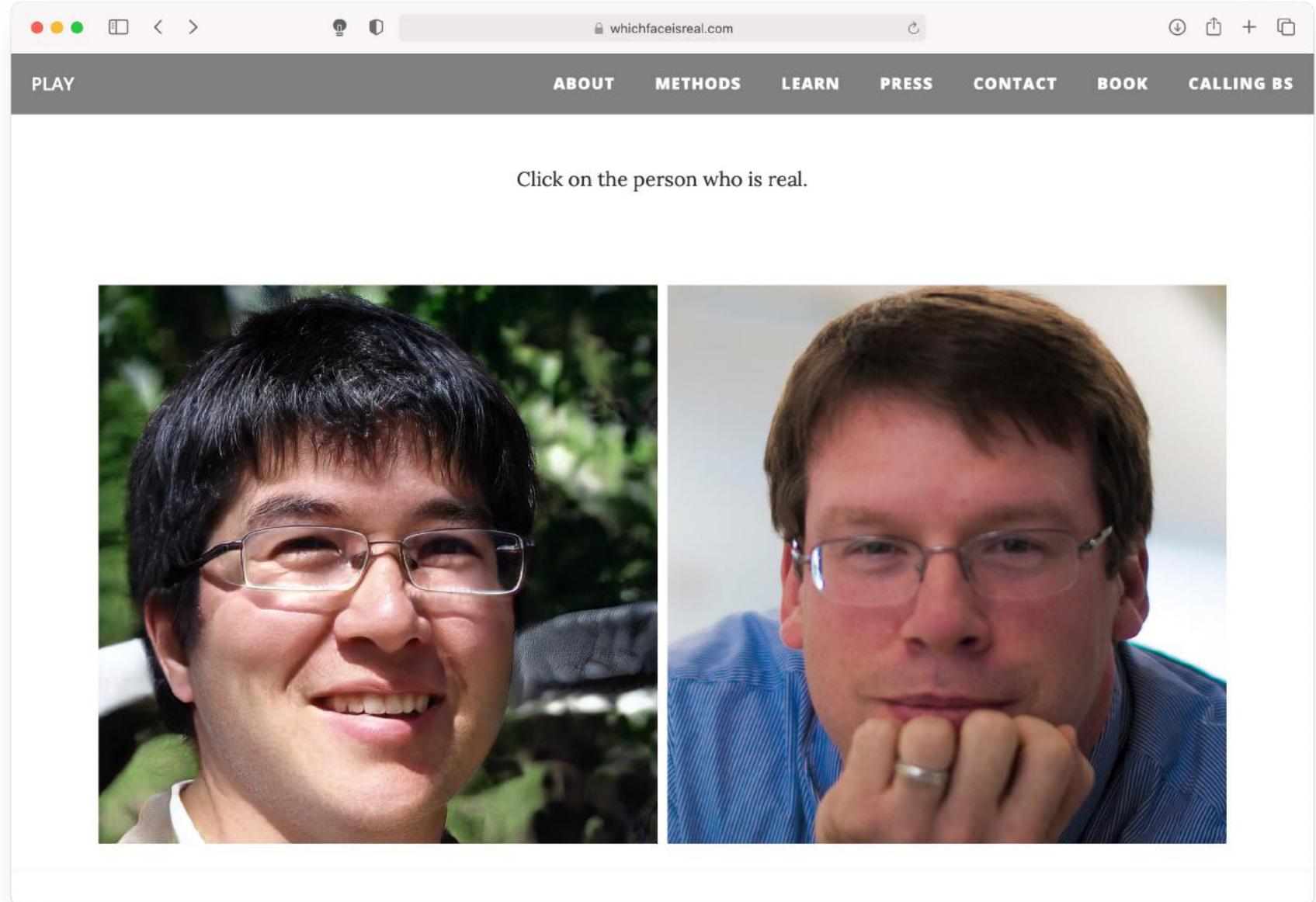
StyleGAN - Effect of noise



(a) Generated image

(b) Stochastic variation

(c) Standard deviation



<https://www.whichfaceisreal.com/learn.html>

StyleGAN Water Droplet-like Artifacts



Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

StyleGAN2

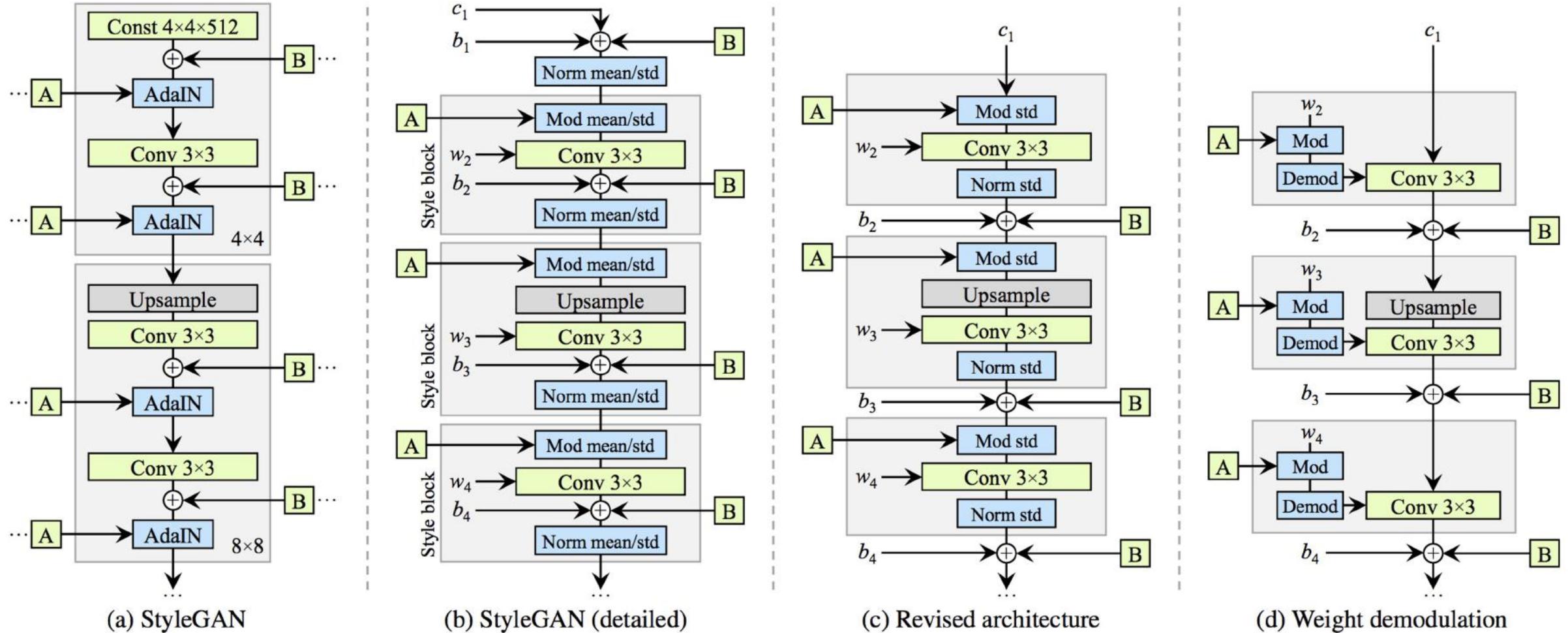


Fig. 2. We redesign the architecture of the StyleGAN synthesis network. (a) The original StyleGAN, where A denotes a learned affine transform from W that produces a style and B is a noise broadcast operation. (b) The same diagram with full detail. Here we have broken the AdaIN to explicit normalization followed by modulation, both operating on the mean and standard deviation per feature map. We have also annotated the learned weights (w), biases (b), and constant input (c), and redrawn the gray boxes so that one style is active per box. The activation function (leaky ReLU) is always applied right after adding the bias. (c) We make several changes to the original architecture that are justified in the main text. We remove some redundant operations at the beginning, move the addition of b and B to be outside active area of a style, and adjust only the standard deviation per feature map. (d) The revised architecture enables us to replace instance normalization with a “demodulation” operation, which we apply to the weights associated with each conv layer.

StyleGAN2 Phase Artifacts



Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

StyleGAN2 Phase Artifacts

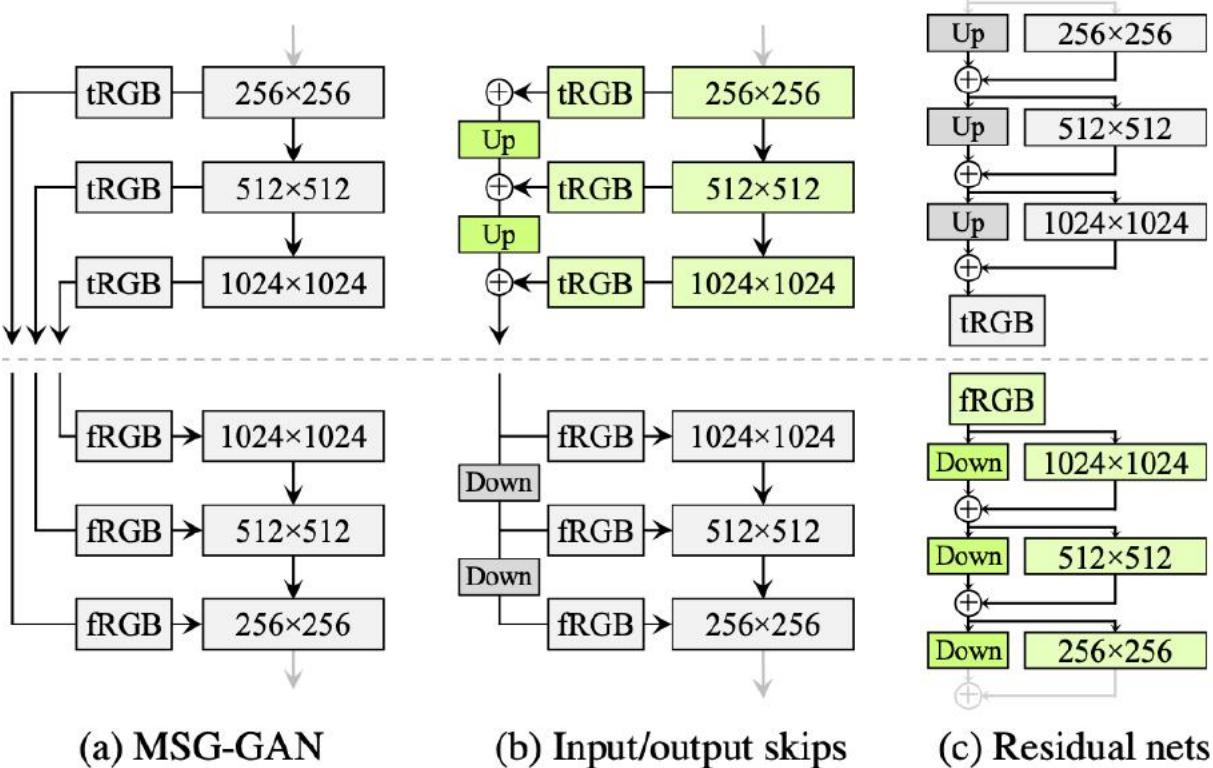
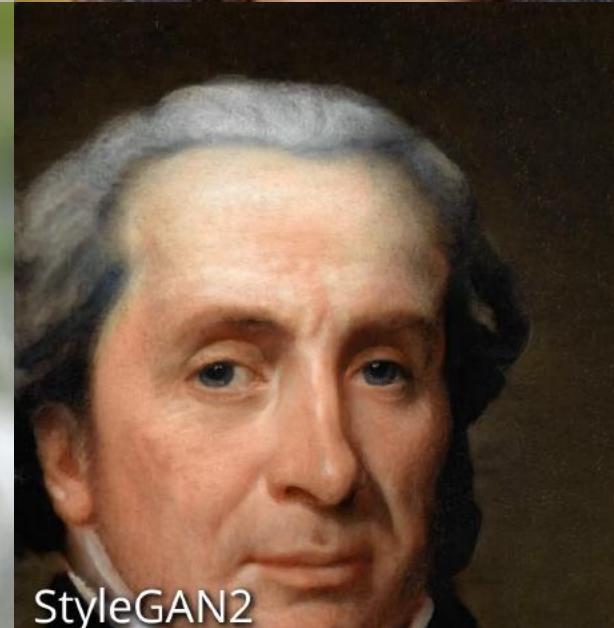
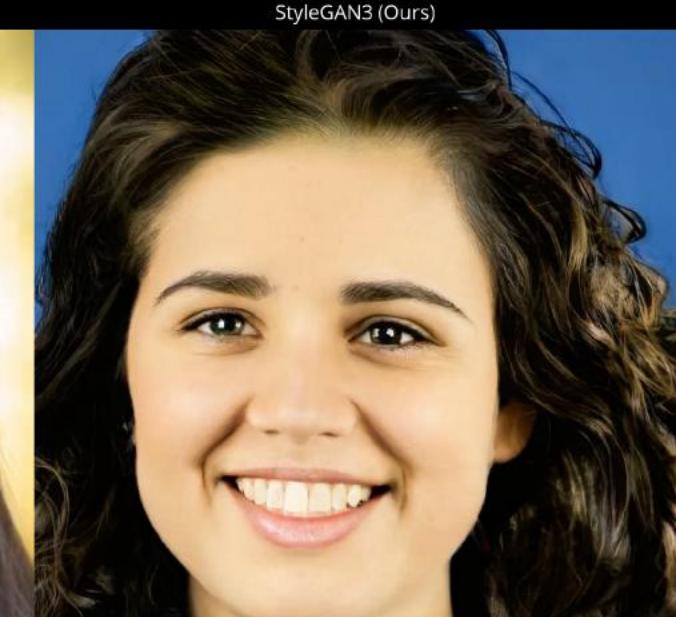
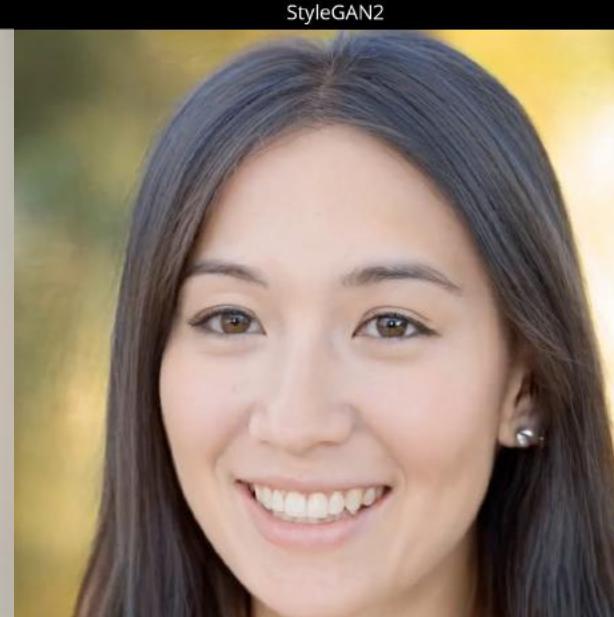


Figure 7. Three generator (above the dashed line) and discriminator architectures. **Up** and **Down** denote bilinear up and down-sampling, respectively. In residual networks these also include 1×1 convolutions to adjust the number of feature maps. **tRGB** and **fRGB** convert between RGB and high-dimensional per-pixel data. Architectures used in configs E and F are shown in green.

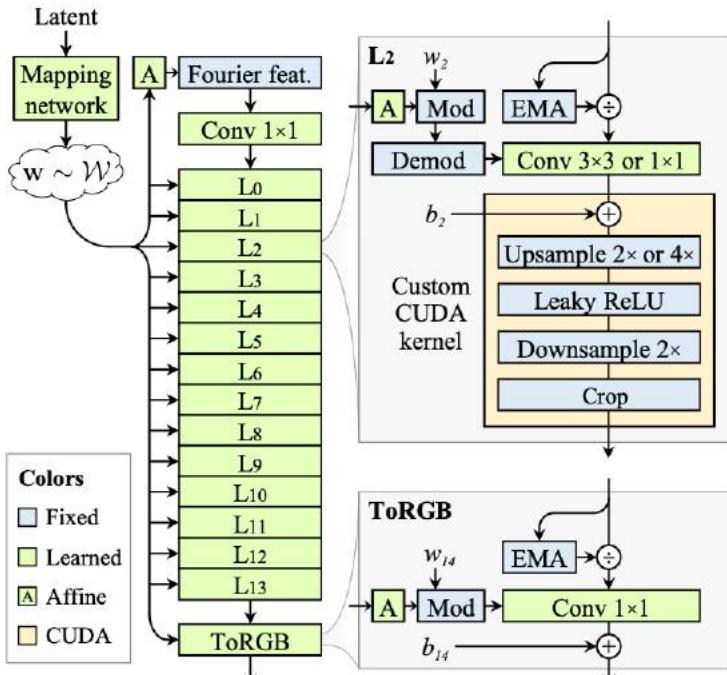


FFHQ, $\Psi = 0.50$

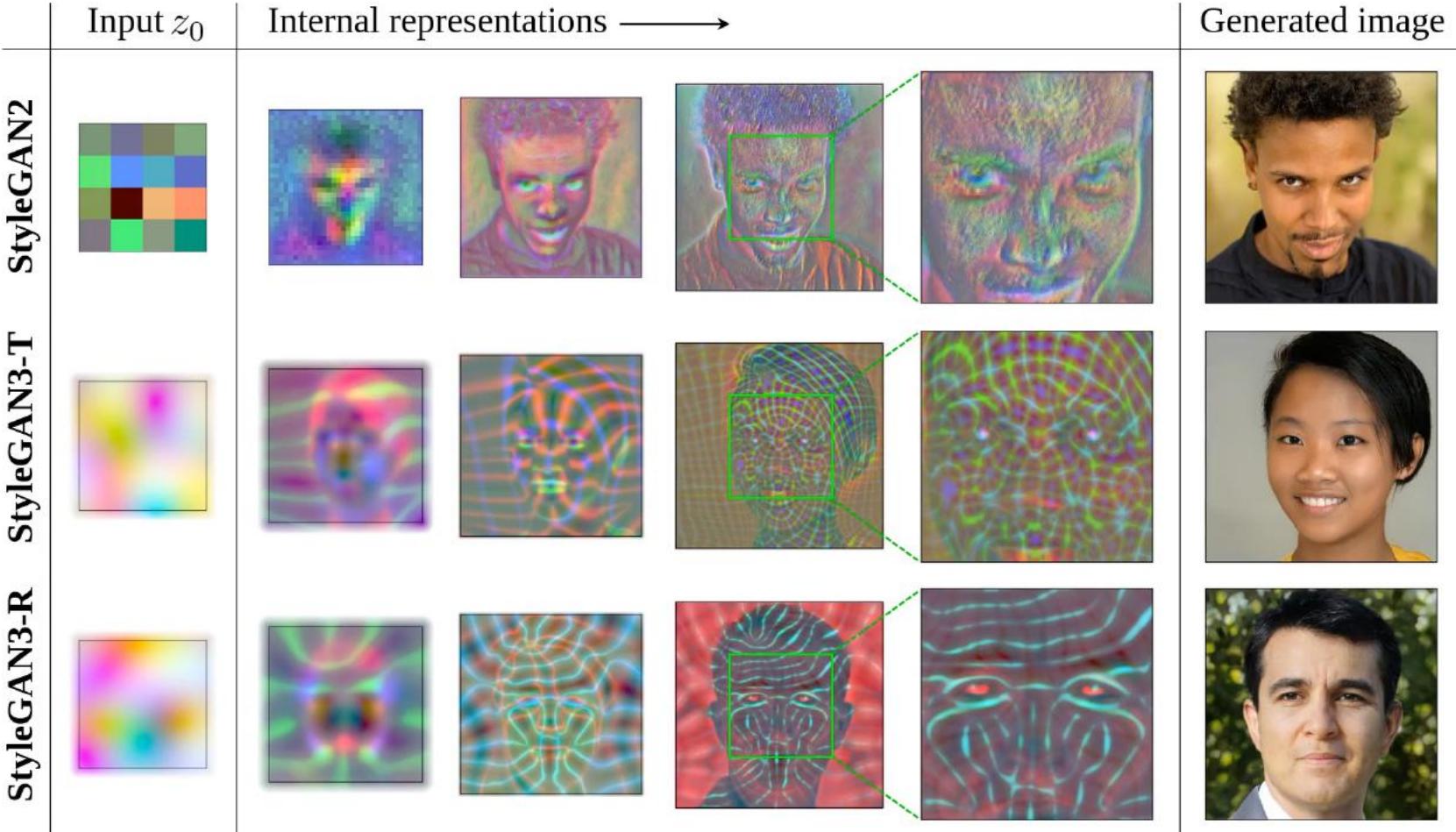
StyleGAN3 to resolve “texture sticking”



StyleGAN3



Implementation requires custom CUDA kernel



- Internal activations encode phase information
- Fully equivariant to translation and rotation even at subpixel scale

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN,BigGAN-Deep,StyleGAN,StyleGAN2,StyleGAN3,**StyleGAN-XL**,VIB-GAN
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

StyleGAN-XL

- StyleGAN was designed for controllability
- Its performance degrades on unstructured datasets such as ImageNet.
- StyleGAN-XL shows that it is possible with a carefully designed architecture and training schemes
 - StyleGAN3 framework
 - Projected GAN objective
 - Progressive growing
 - 1024×1024 images

StyleGAN-XL

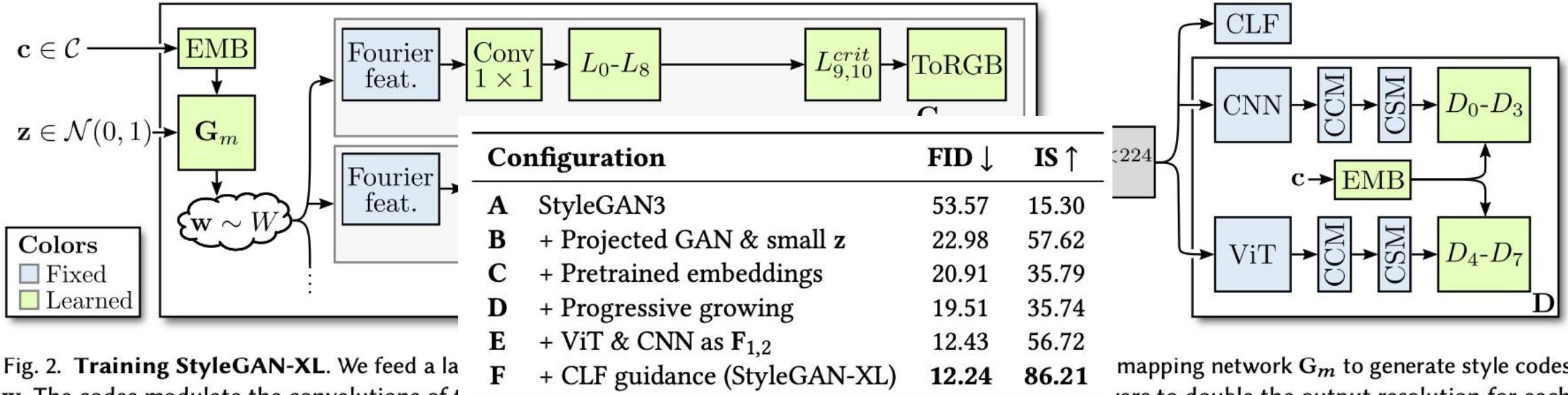


Fig. 2. **Training StyleGAN-XL.** We feed a latent vector w . The codes modulate the convolutions of the generator G . The mapping network G_m to generate style codes $w \sim W$ is trained to double the output resolution for each stage of the progressive growing schedule. We only train the latest layers while keeping the others fixed. The synthesized image is upsampled when smaller than 224^2 and passed through a CNN and a ViT and respective feature mixing blocks (CCM+CSM). At higher resolutions, the CNN receives the unaltered image while the ViT receives a downsampled input to keep memory requirements low but still utilize its global feedback. Finally, we apply eight independent discriminators on the resulting multi-scale feature maps. The image is also fed to classifier CLF for classifier guidance.

mapping network G_m to generate style codes $w \sim W$ is trained to double the output resolution for each stage of the progressive growing schedule. We only train the latest layers while keeping the others fixed. The synthesized image is upsampled when smaller than 224^2 and passed through a CNN and a ViT and respective feature mixing blocks (CCM+CSM). At higher resolutions, the CNN receives the unaltered image while the ViT receives a downsampled input to keep memory requirements low but still utilize its global feedback. Finally, we apply eight independent discriminators on the resulting multi-scale feature maps. The image is also fed to classifier CLF for classifier guidance.

StyleGAN-XL

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small z	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $F_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21



Fig. 3. Samples at Different Resolutions Using the Same w . The samples are generated by the models obtained during progressive growing. We upsample all images to 1024^2 using nearest-neighbor interpolation for visualization purposes. Zooming in is recommended.



Fig. 4. Inversion of a Given Source Image. For BigGAN, we invert to its latent space z , for StyleGAN-XL we invert to style codes w .

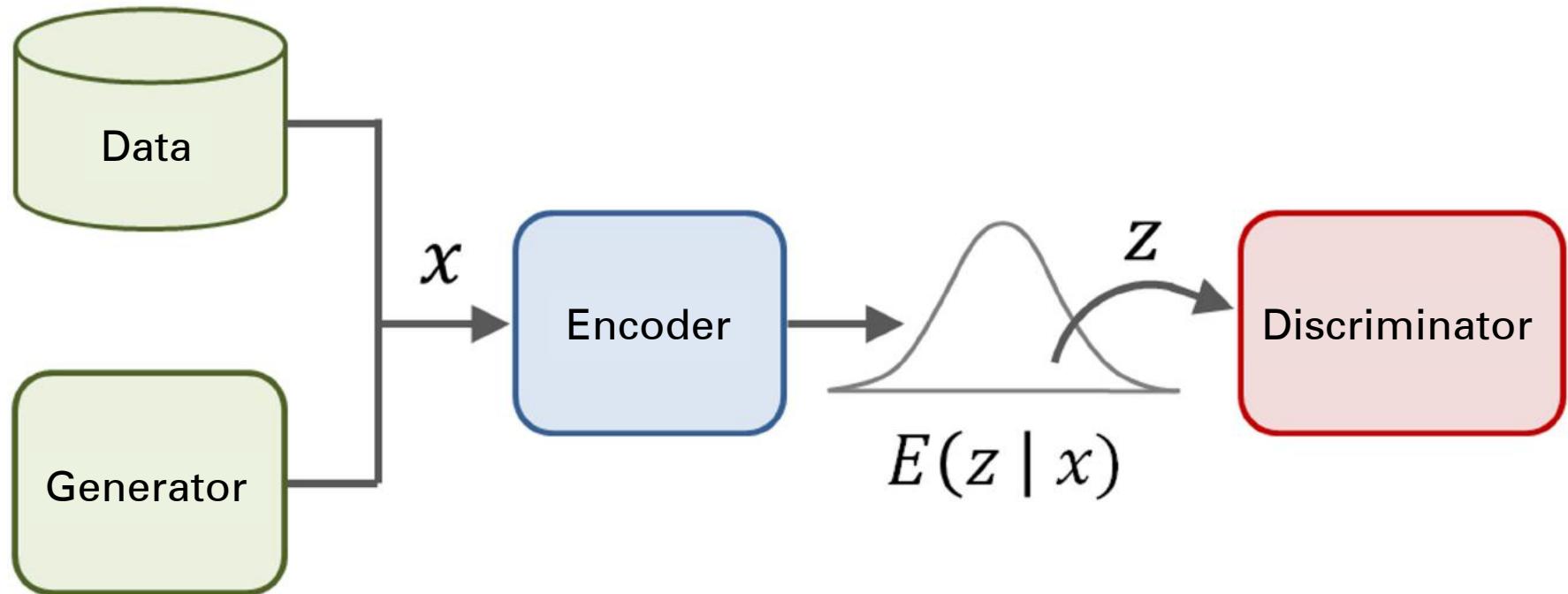


Style-based GANs achieve high image fidelity,

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- **GAN Progression**
 - DC GAN (Radford et al, 2016)
 - Improved Training of GANs (Salimans et al'16), Projected GAN (Sauer et al'21)
WGAN, WGAN-GP, Progressive GAN, SN-GAN, SAGAN
 - BigGAN, BigGAN-Deep, StyleGAN, StyleGAN2, StyleGAN3, StyleGAN-XL, VIB-GAN
- Conditional GANs
- GANs and Representations
- Applications

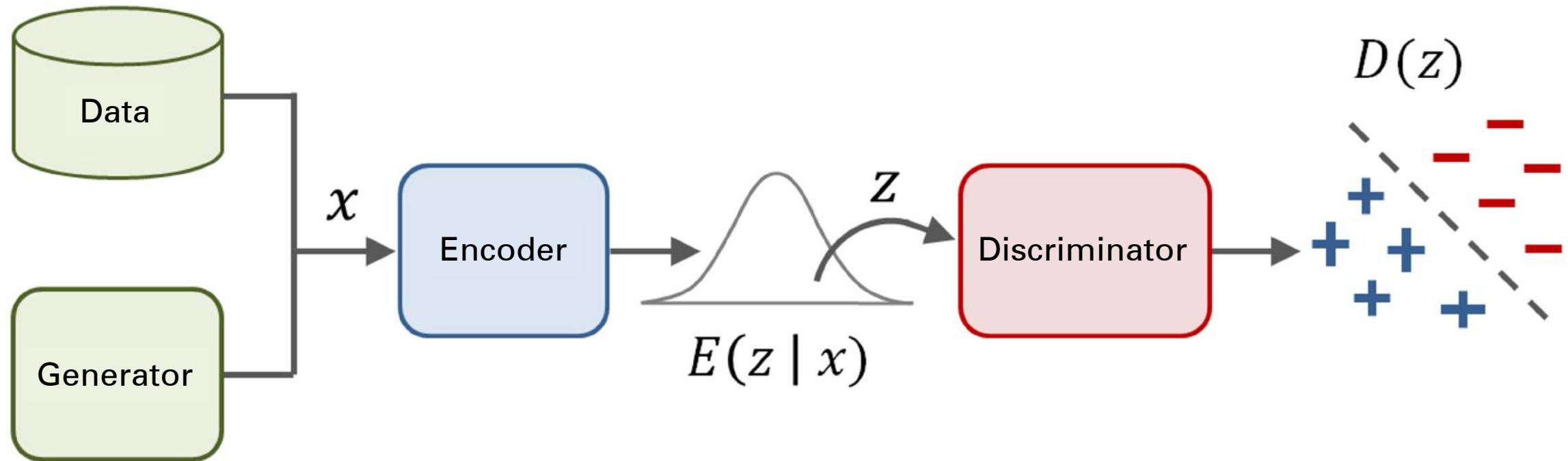
Information Bottleneck



Variational Information Bottleneck [Alemi et al., 2016]

Variational Information Bottleneck GAN [Peng et al, 2019]

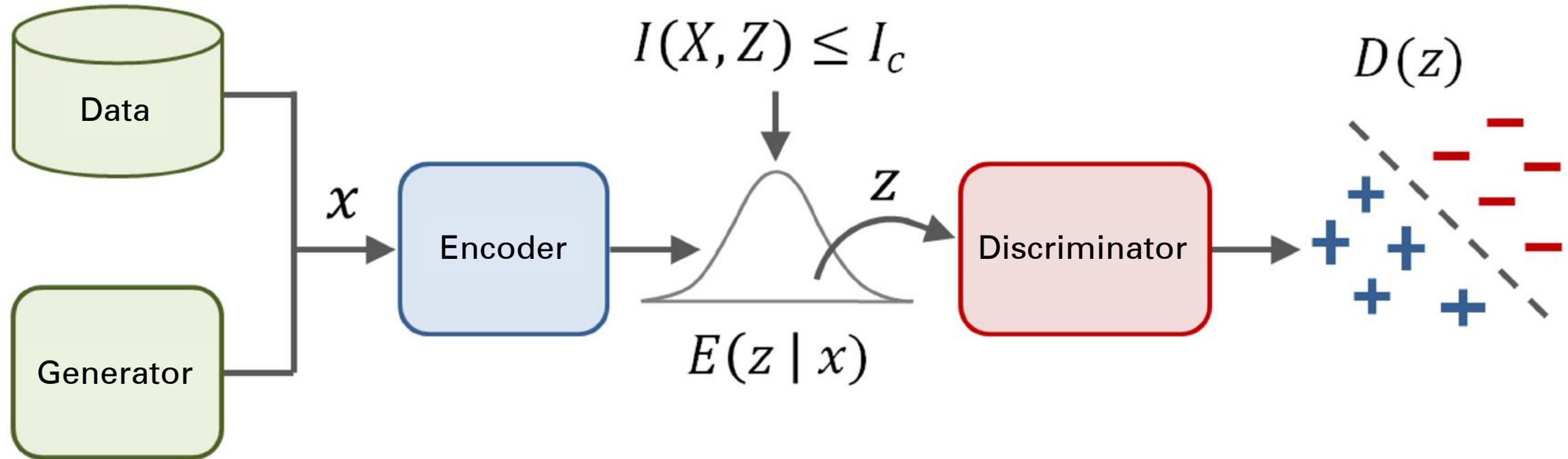
Information Bottleneck



Variational Information Bottleneck [Alemi et al., 2016]

Variational Information Bottleneck GAN [Peng et al, 2019]

Information Bottleneck



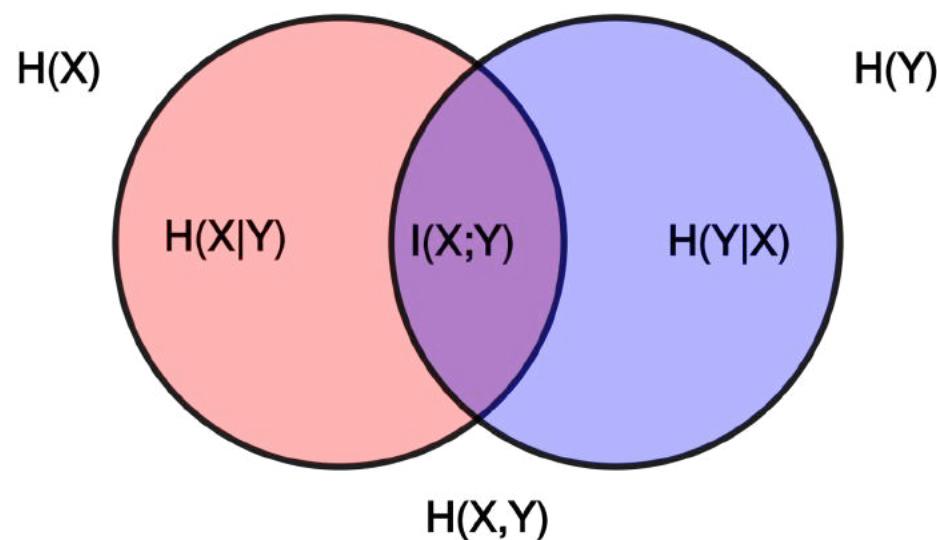
Variational Information Bottleneck [Alemi et al., 2016]

Variational Information Bottleneck GAN [Peng et al, 2019]

Mutual Information

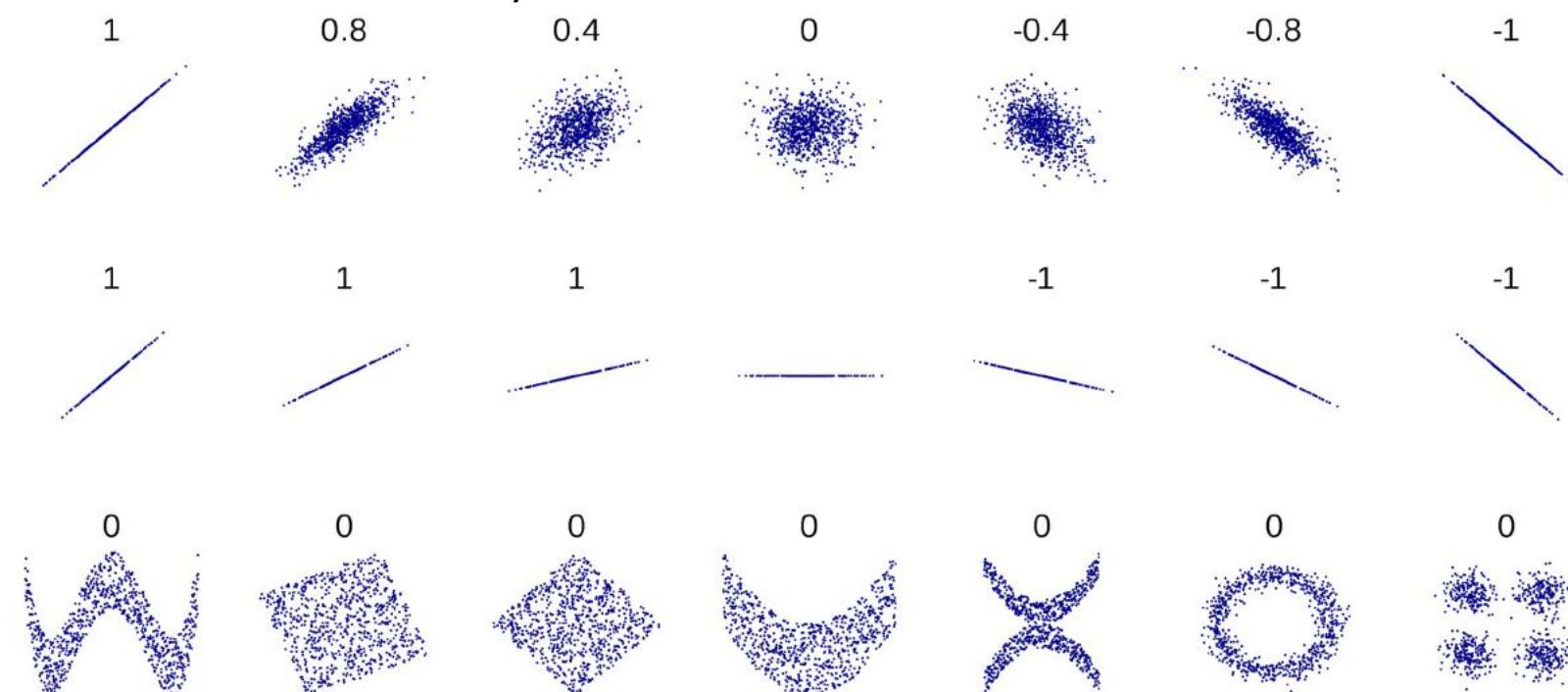
- Mutual information between two random variables X, Y : $I(X; Y)$ is defined as

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$



Mutual Information

- Mutual Information is a general way to measure dependency between two random variables
 - Unlike the more commonly used covariance



Estimating Mutual Information

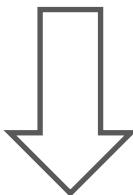
- We can try to estimate the mutual information between z and x in a latent variable model

$$\begin{aligned} I(z; x) &= H(z) - H(z|x) \\ &= H(z) - \mathbb{E}_{(z,x) \sim p(z,x)}[-\log p(z|x)] \\ &= H(z) + \mathbb{E}_{(z,x) \sim p(z,x)}[\log p(z|x) - \log q(z|x) + \log q(z|x)] \\ &\geq H(z) + \mathbb{E}_{(z,x) \sim p(z,x)}[\log q(z|x)] \end{aligned}$$

- Has intractable posterior $p(z|x)$ but we can estimate by introducing a variational distribution $q(z|x)$

Information Bottleneck

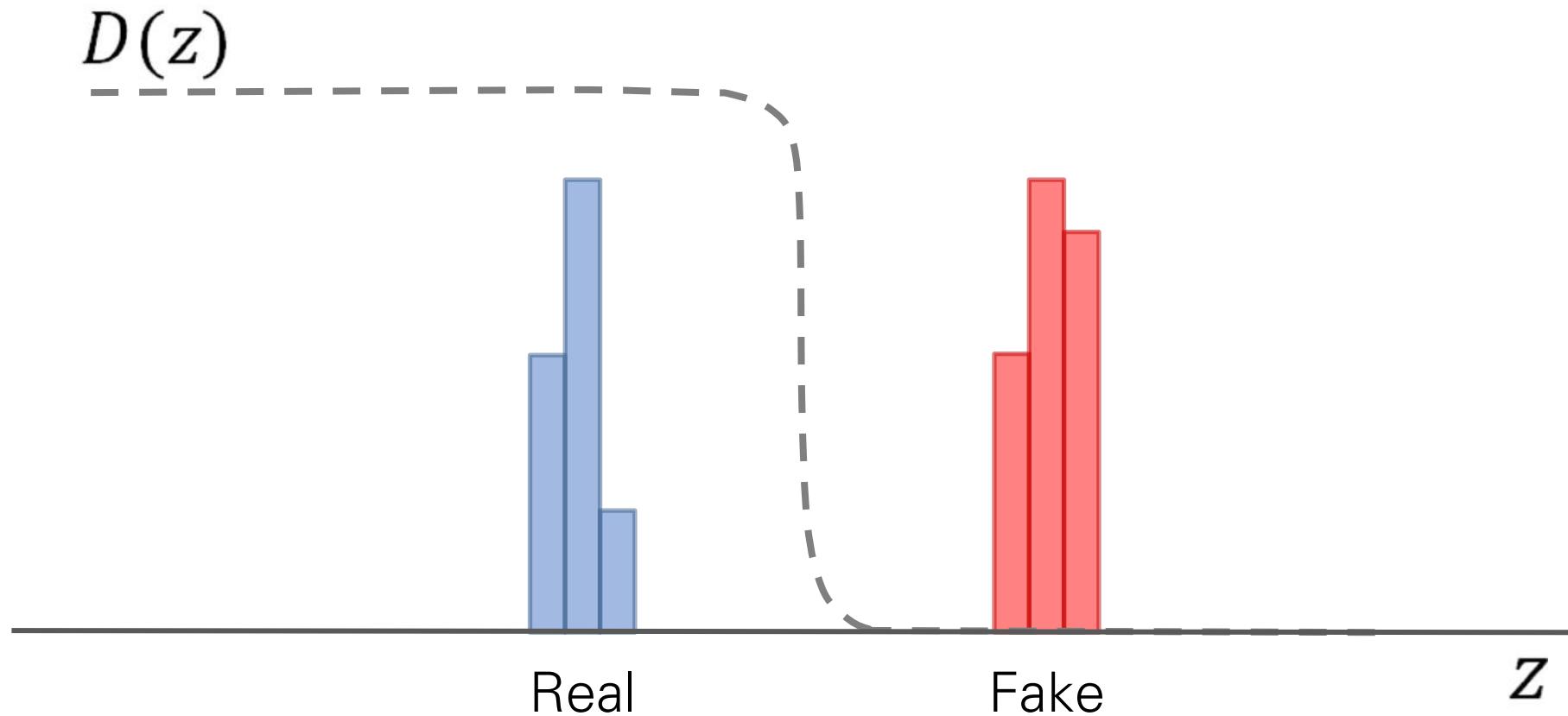
$$I(X, Z) \leq I_c$$



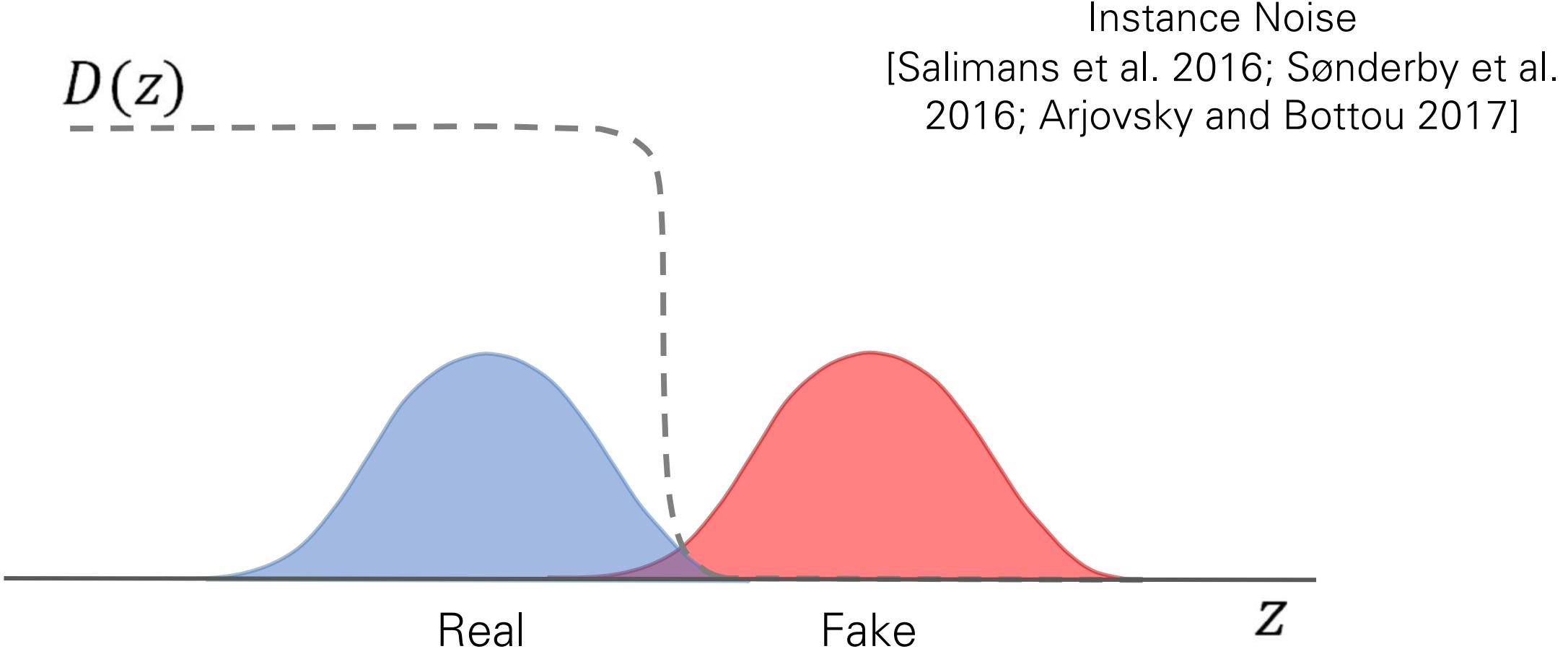
$$\mathbb{E}_{\mathbf{x} \sim \tilde{p}(\mathbf{x})} [\text{KL} [E(\mathbf{z}|\mathbf{x}) || r(\mathbf{z})]] \leq I_c$$

Variational Information Bottleneck (VIB)
[Alemi et al., 2016]

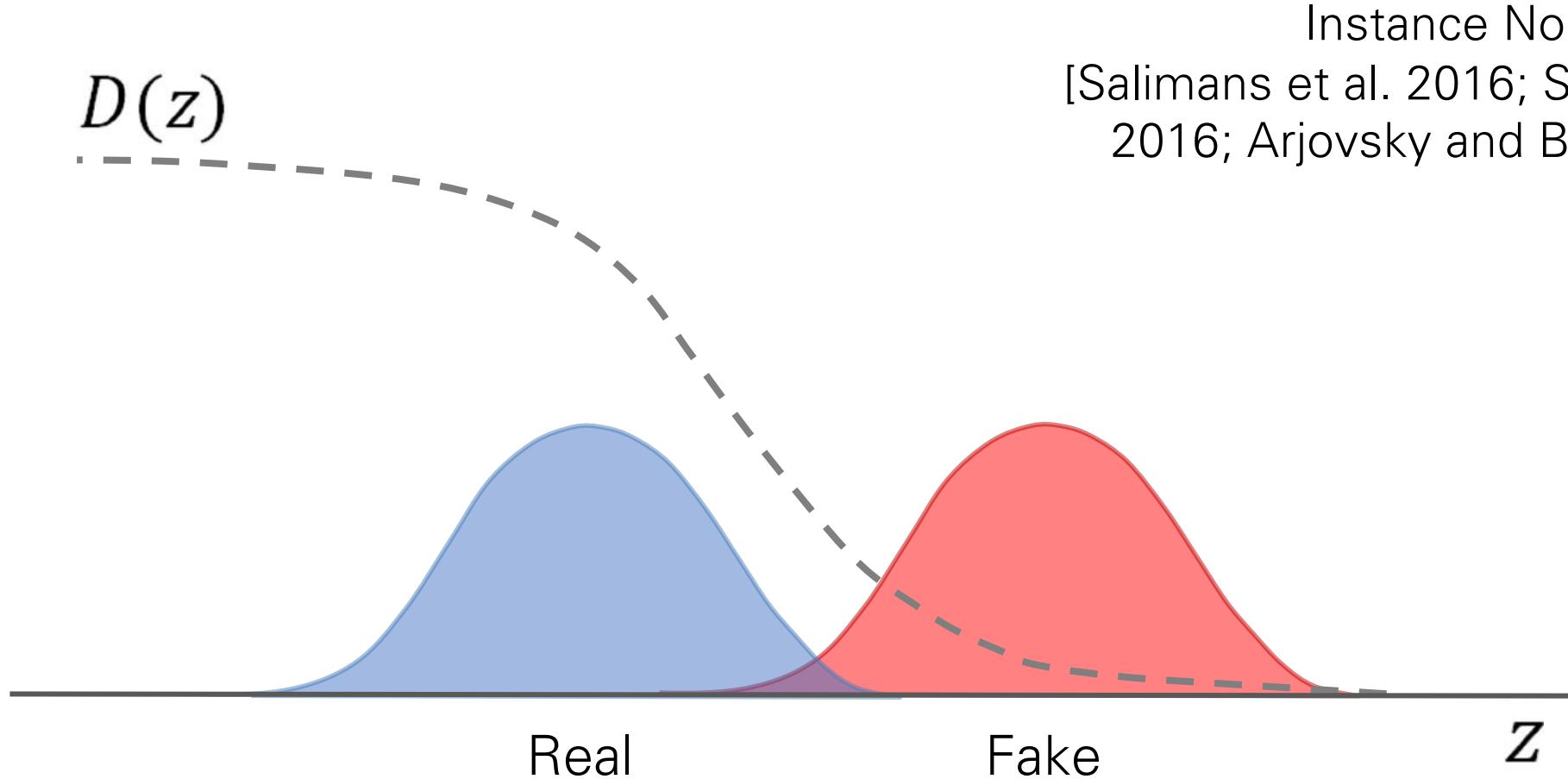
Variational Information Bottleneck



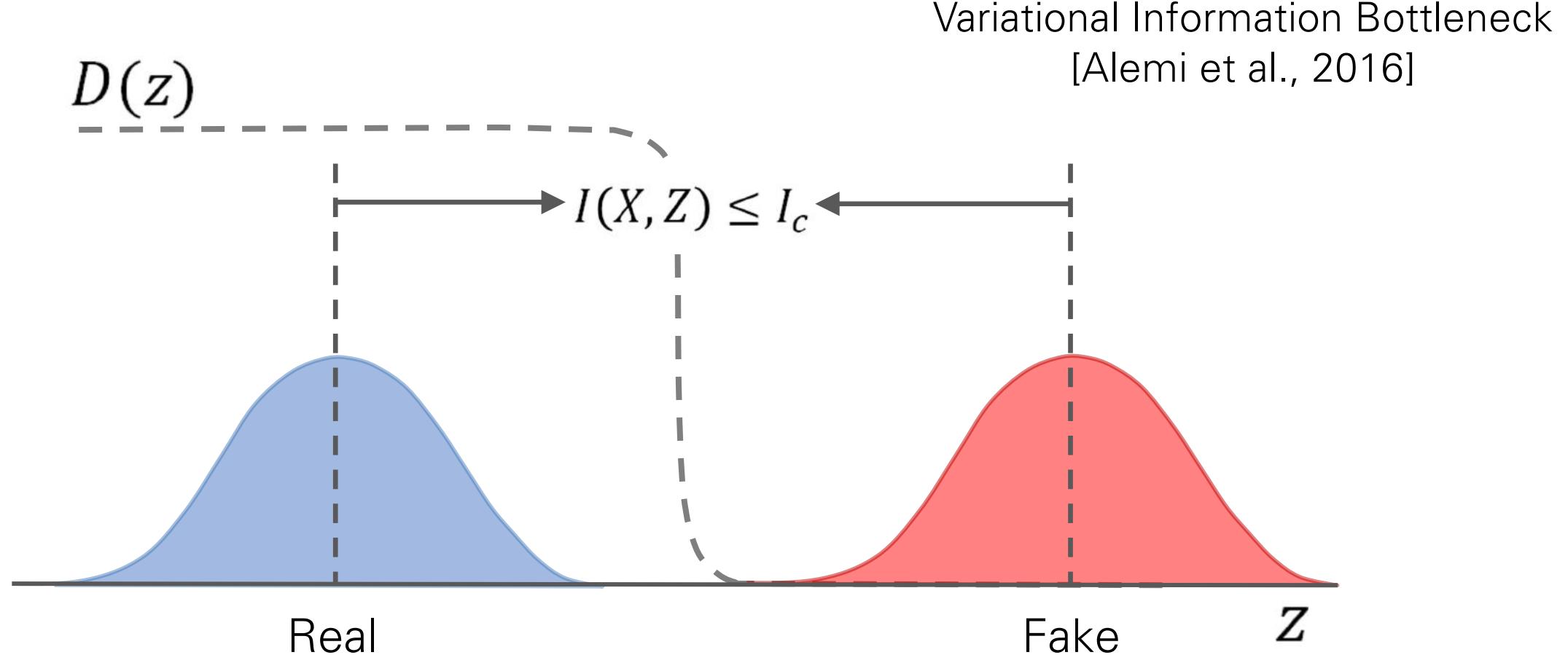
Variational Information Bottleneck



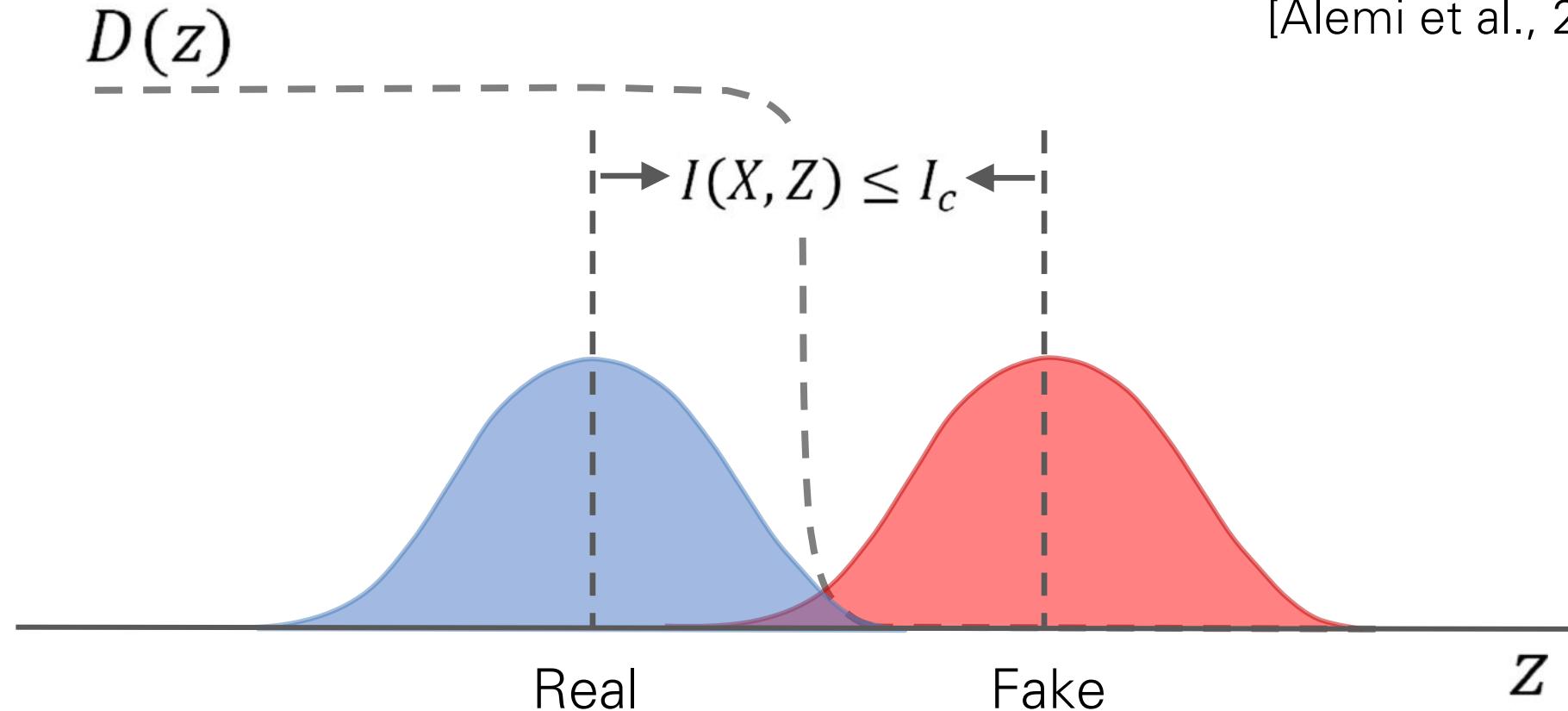
Variational Information Bottleneck



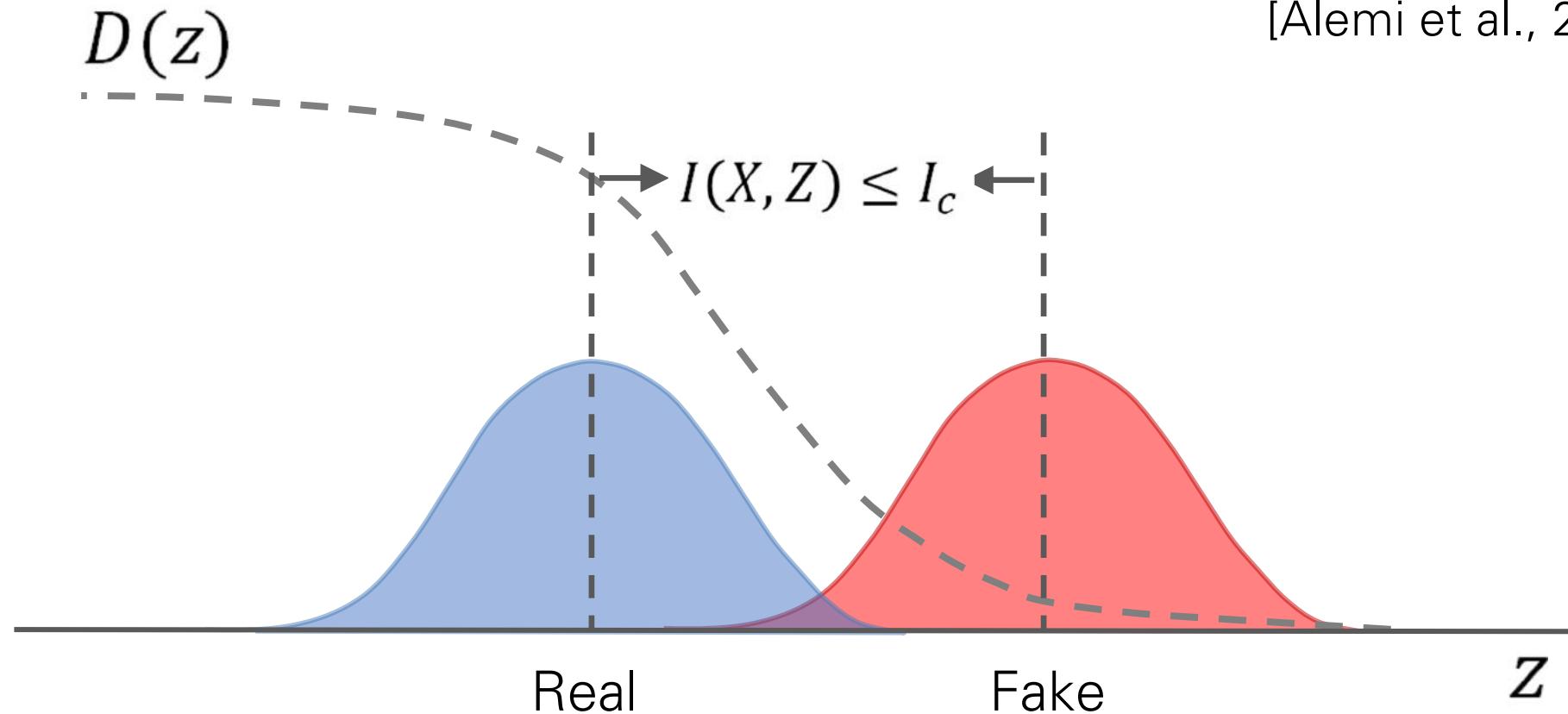
Variational Information Bottleneck



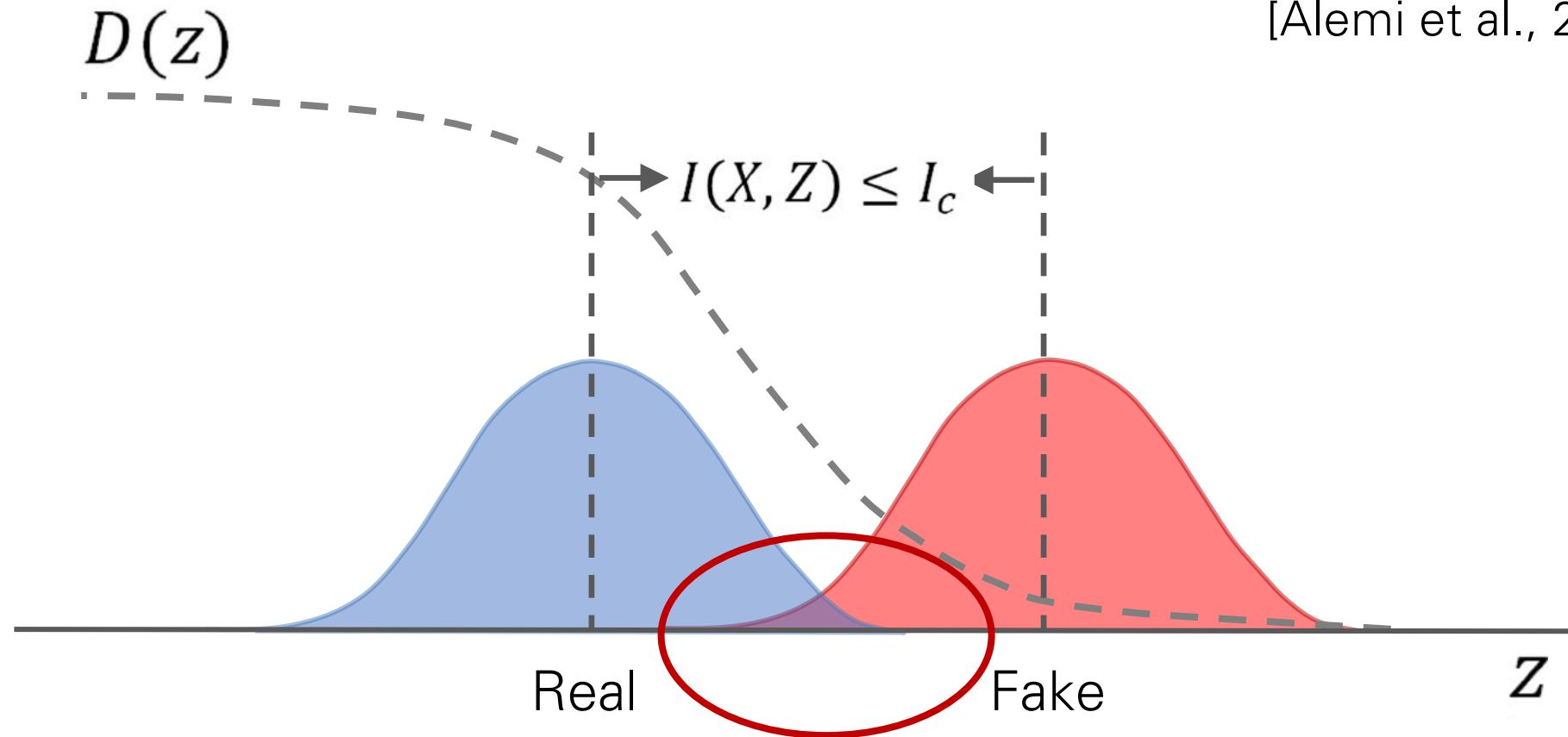
Variational Information Bottleneck



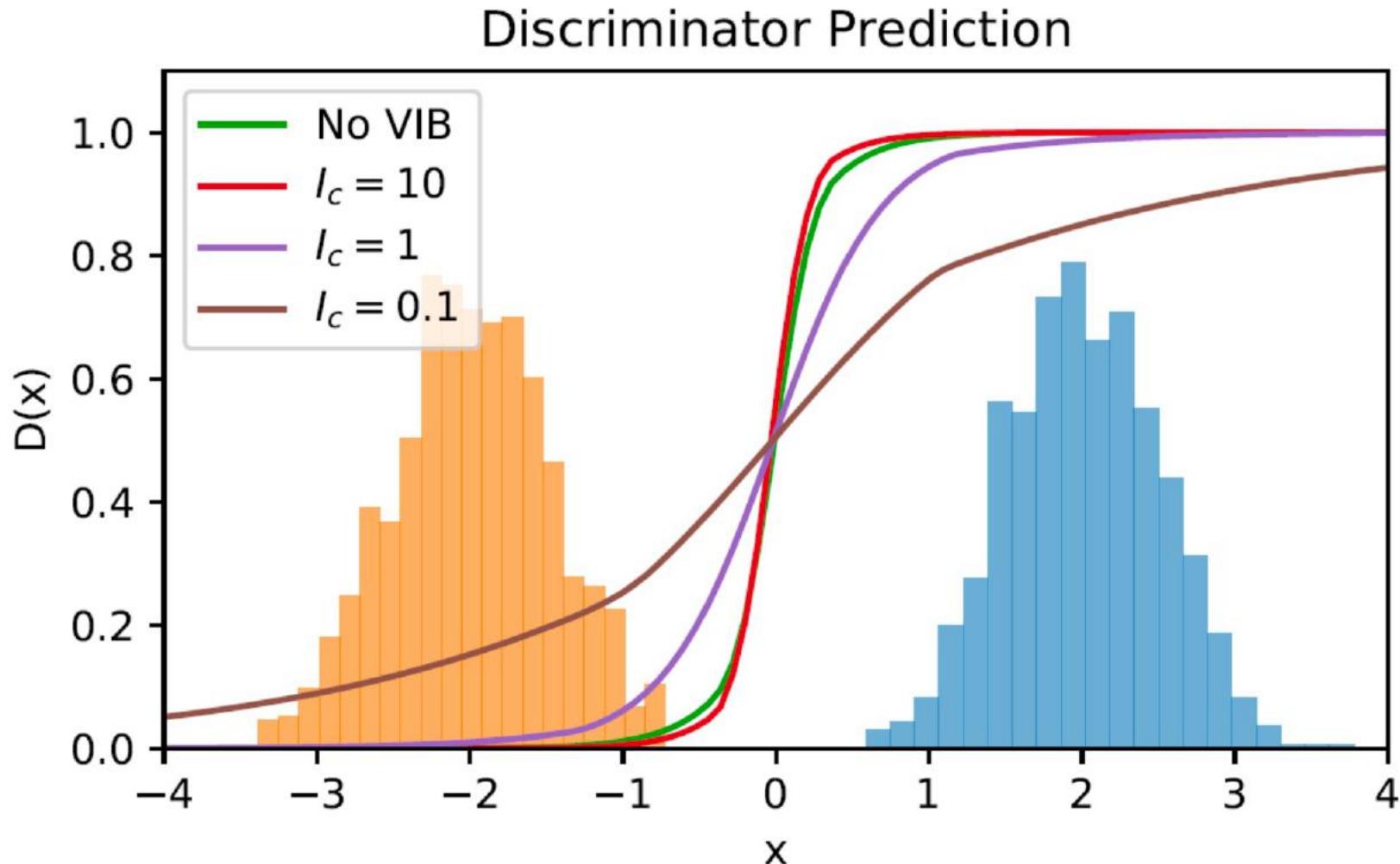
Variational Information Bottleneck



Variational Information Bottleneck



Variational Information Bottleneck GAN

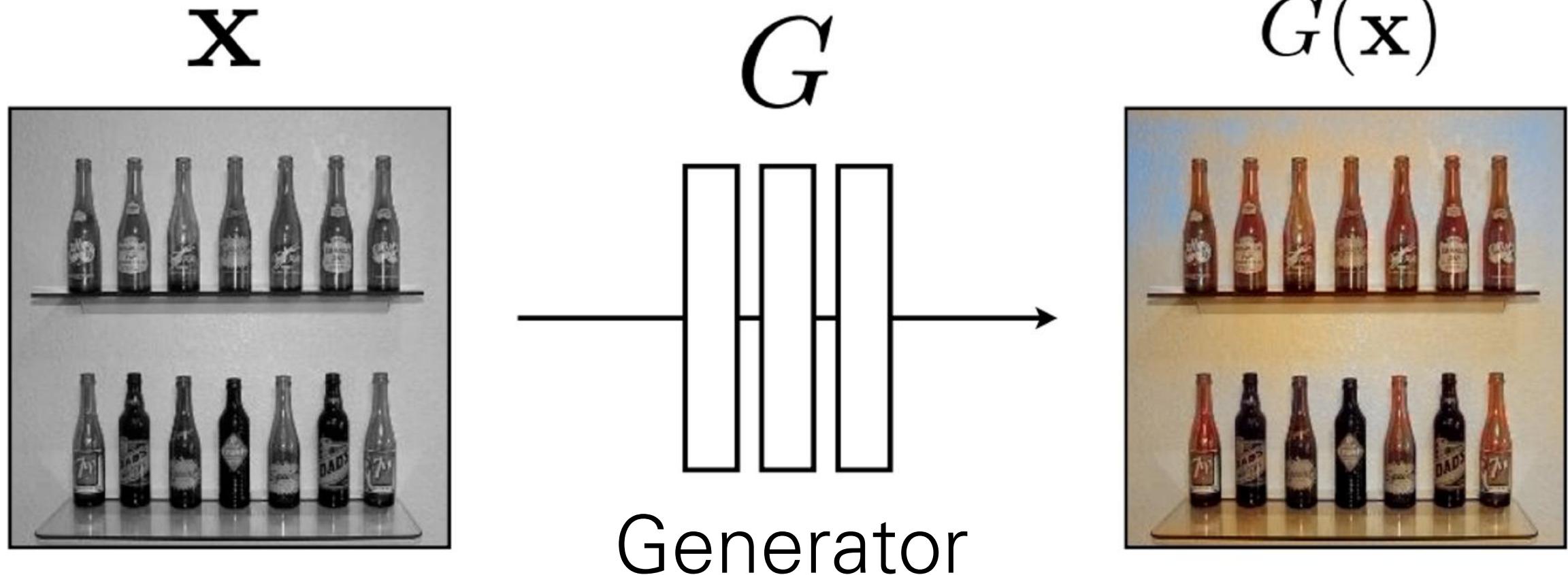


$$\begin{aligned} J(D, E) &= \min_{D, E} \quad \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x})} [\mathbb{E}_{\mathbf{z} \sim E(\mathbf{z}|\mathbf{x})} [-\log(D(\mathbf{z}))]] + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{x})} [\mathbb{E}_{\mathbf{z} \sim E(\mathbf{z}|\mathbf{x})} [-\log(1 - D(\mathbf{z}))]] \\ \text{s.t.} \quad & \mathbb{E}_{\mathbf{x} \sim \tilde{p}(\mathbf{x})} [\text{KL}[E(\mathbf{z} \mid \mathbf{x}) \parallel r(\mathbf{z})]] \leq I_c \end{aligned}$$

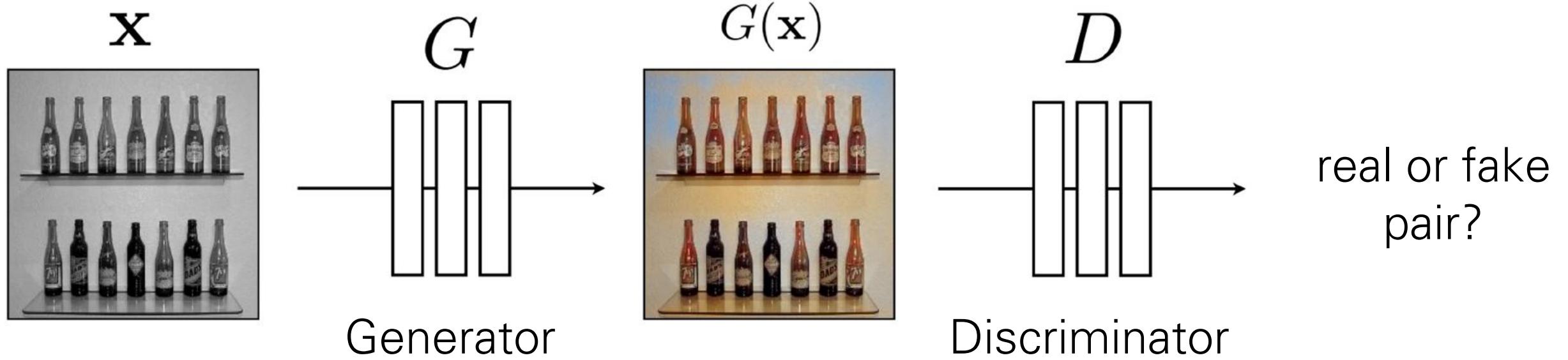
Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- **Conditional GANs**, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

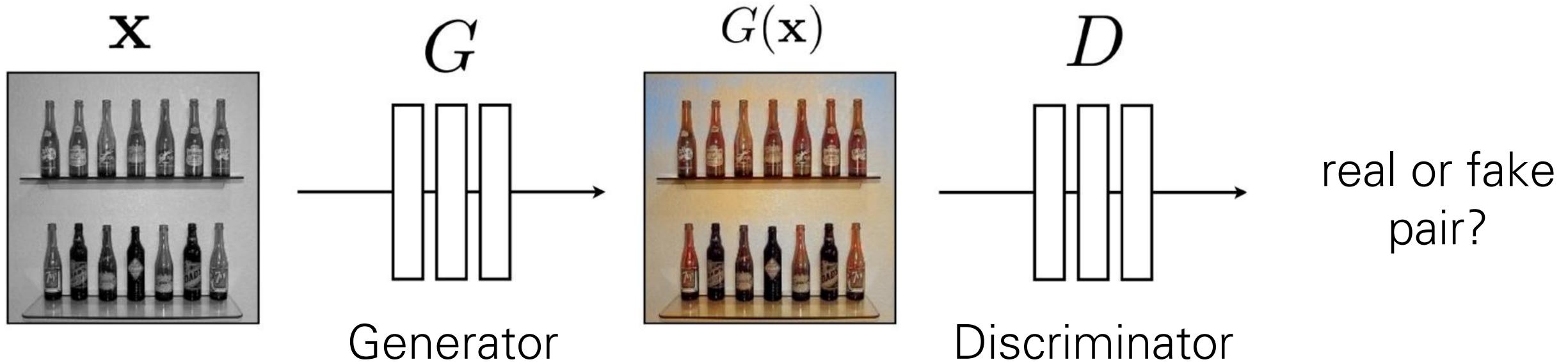
Conditional GANs / pix2pix



Conditional GANs / pix2pix



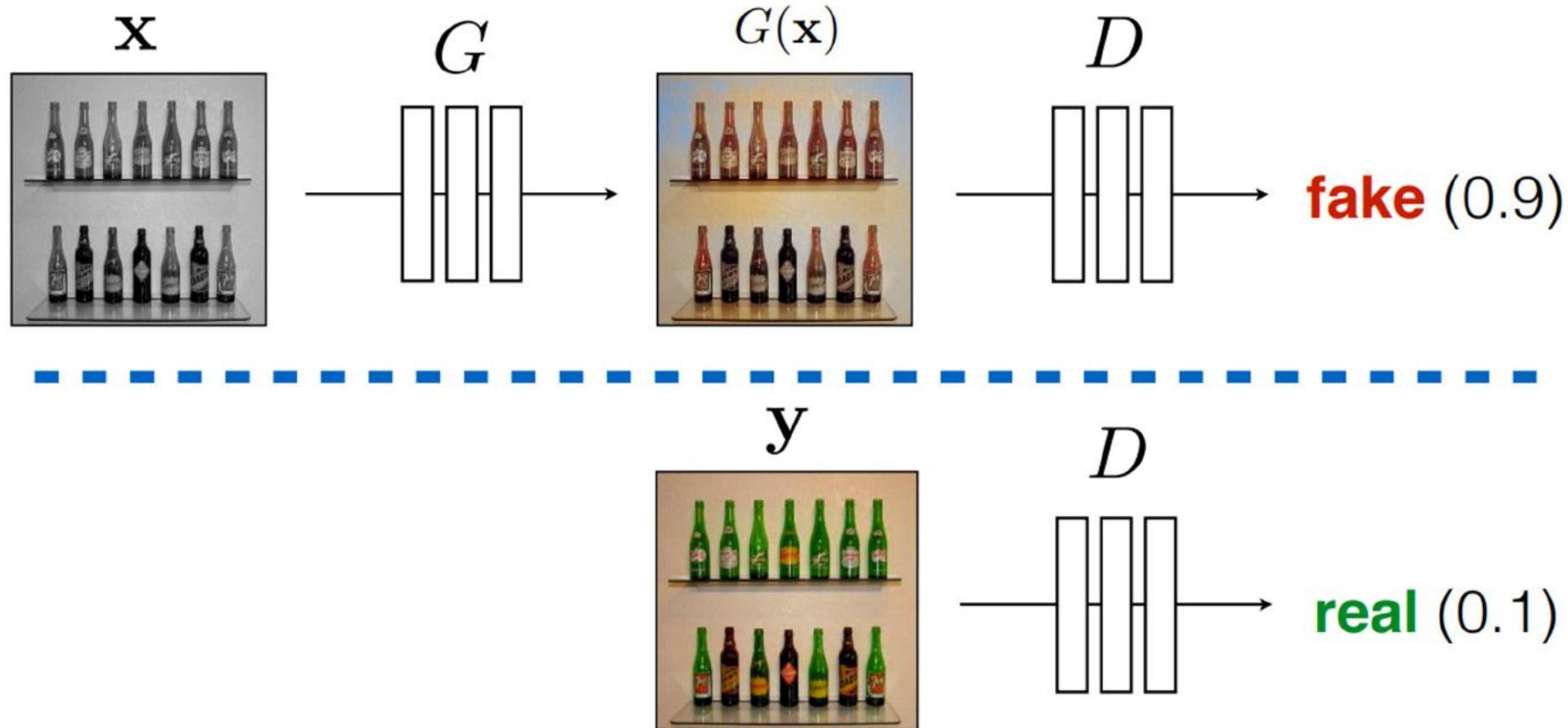
Conditional GANs / pix2pix



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

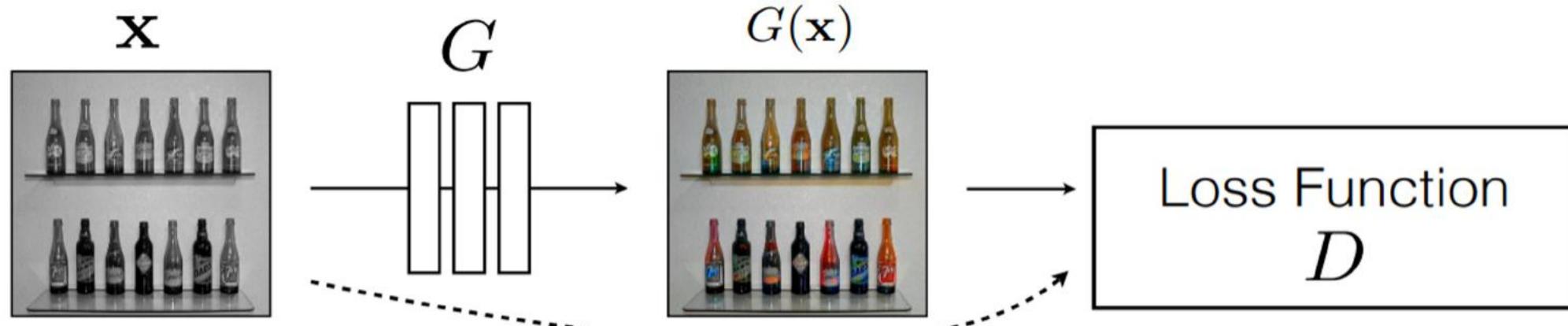
D tries to identify the fakes

Conditional GANs / pix2pix



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

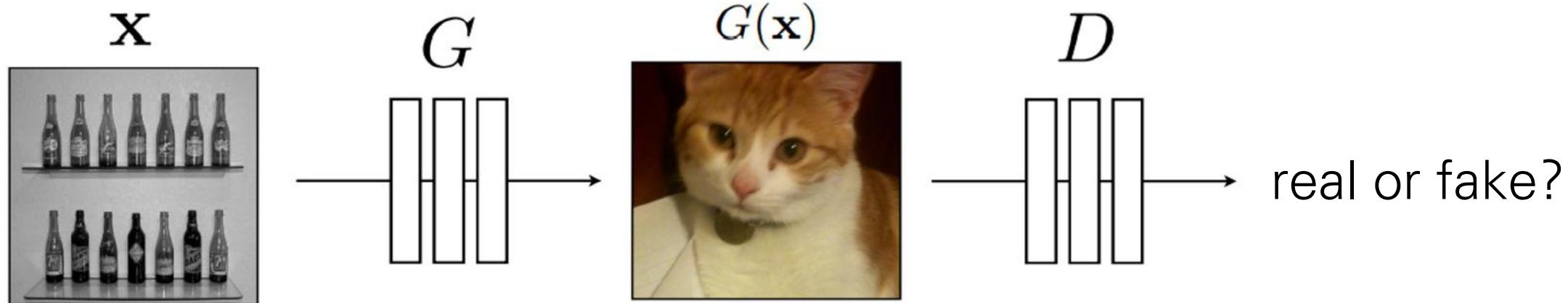
Conditional GANs / pix2pix



G 's perspective: D is a loss function.

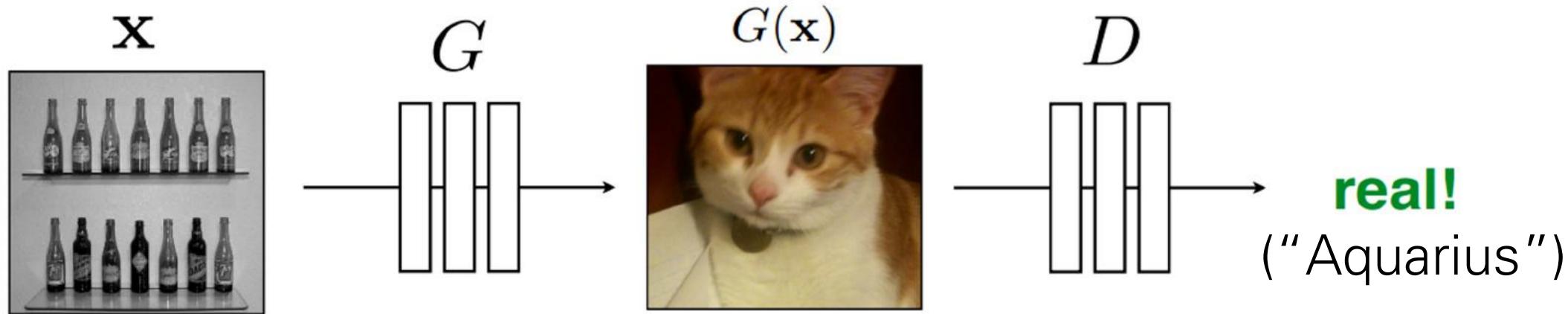
Rather than being hand-designed, it is learned.

Conditional GANs / pix2pix



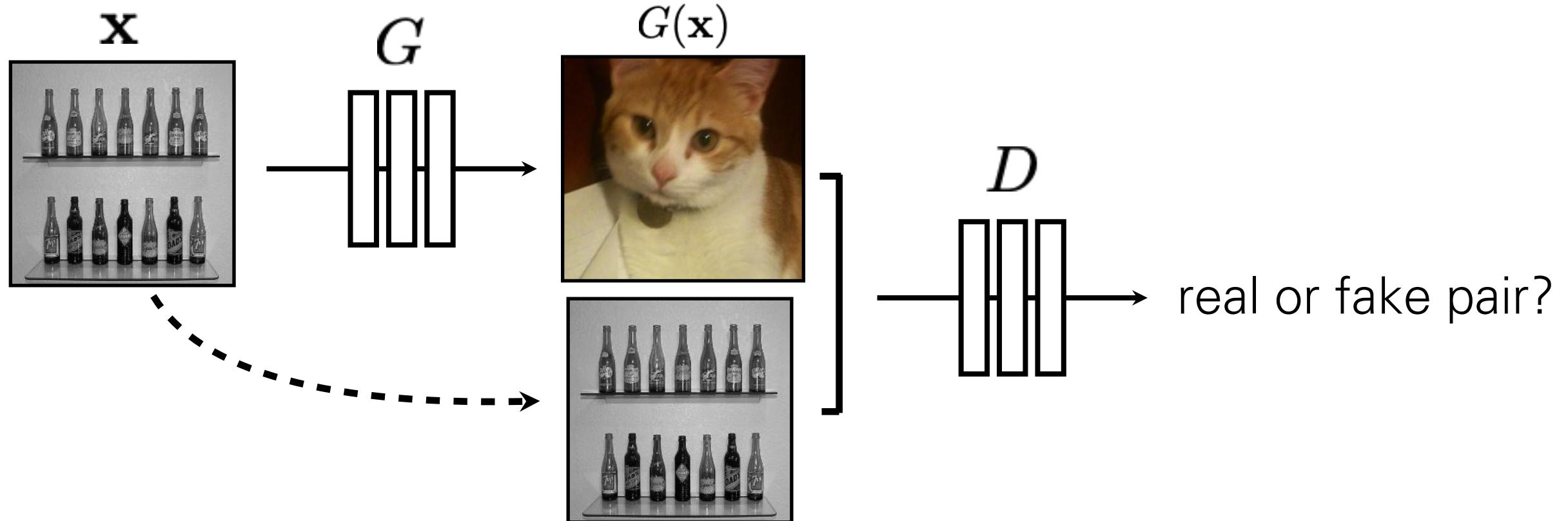
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

Conditional GANs / pix2pix



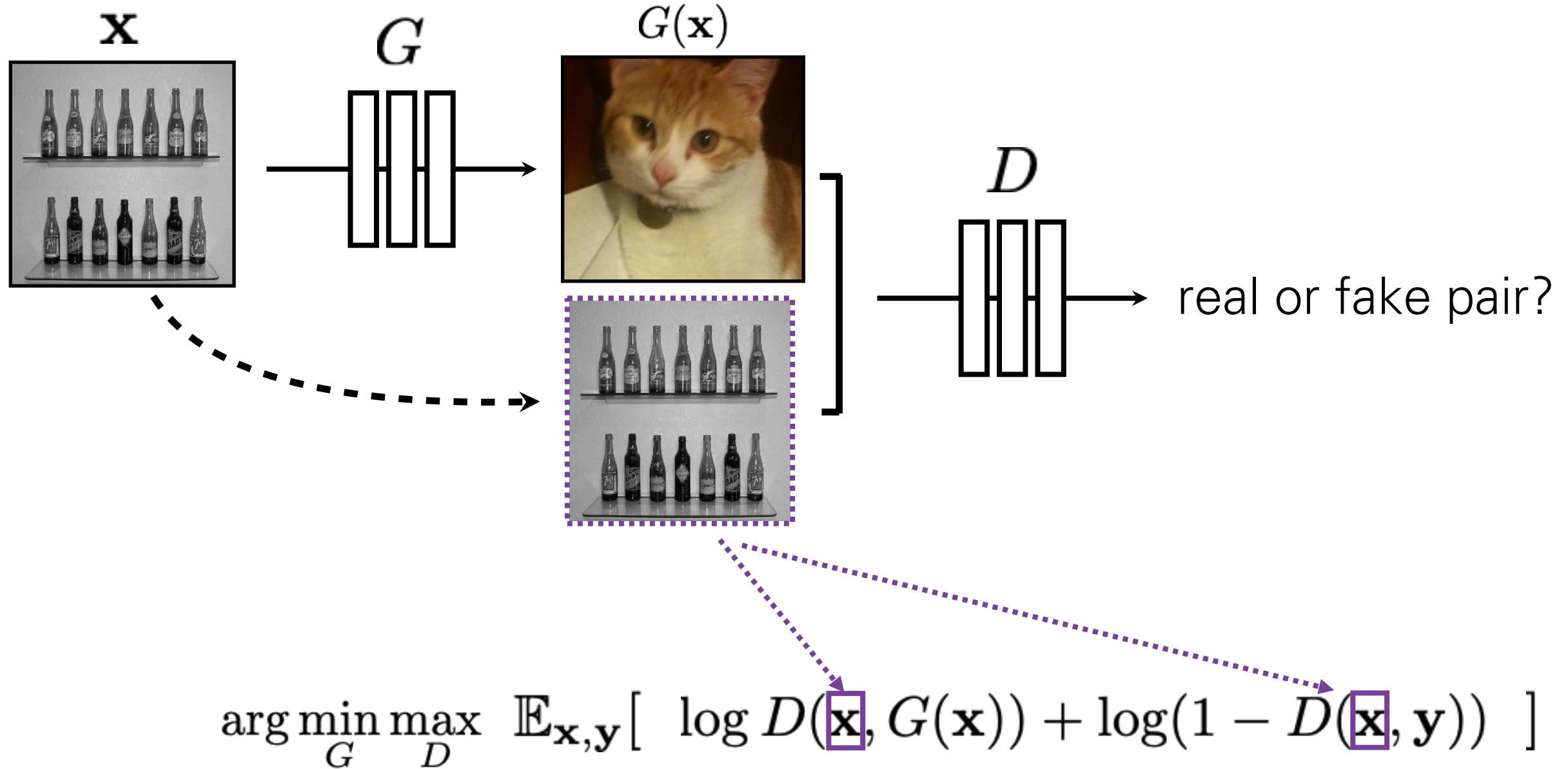
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

Conditional GANs / pix2pix

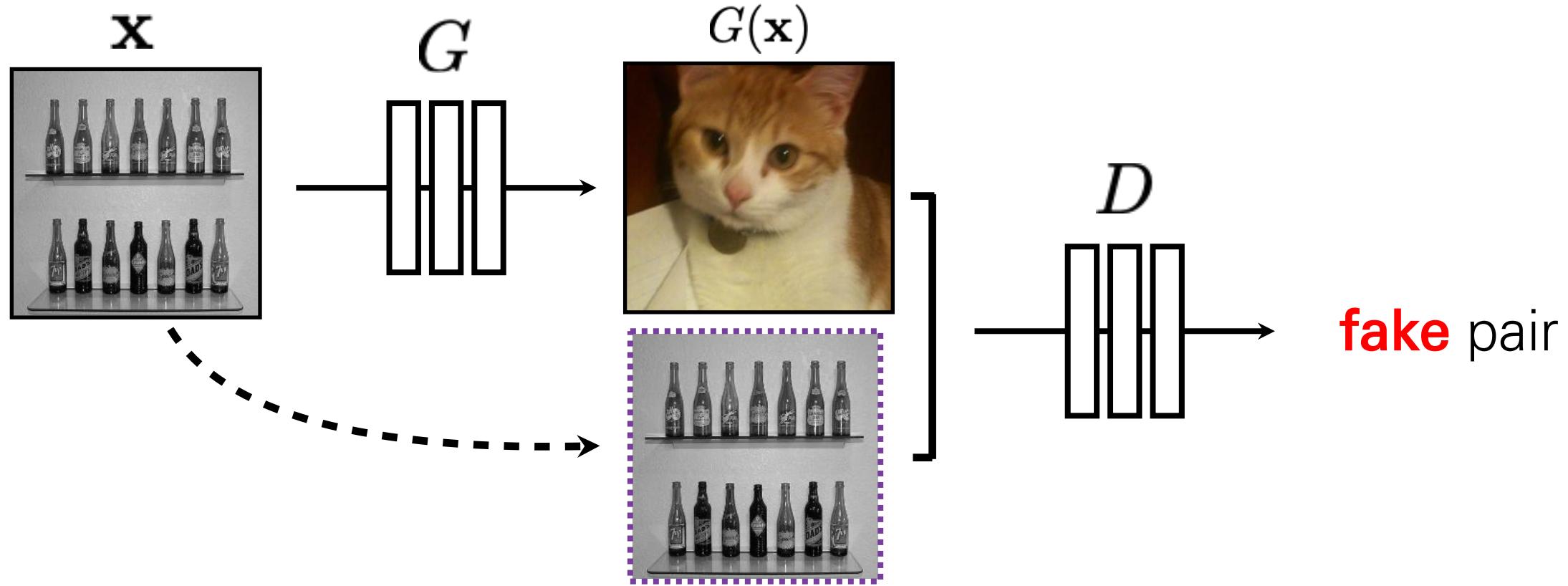


$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

Conditional GANs / pix2pix

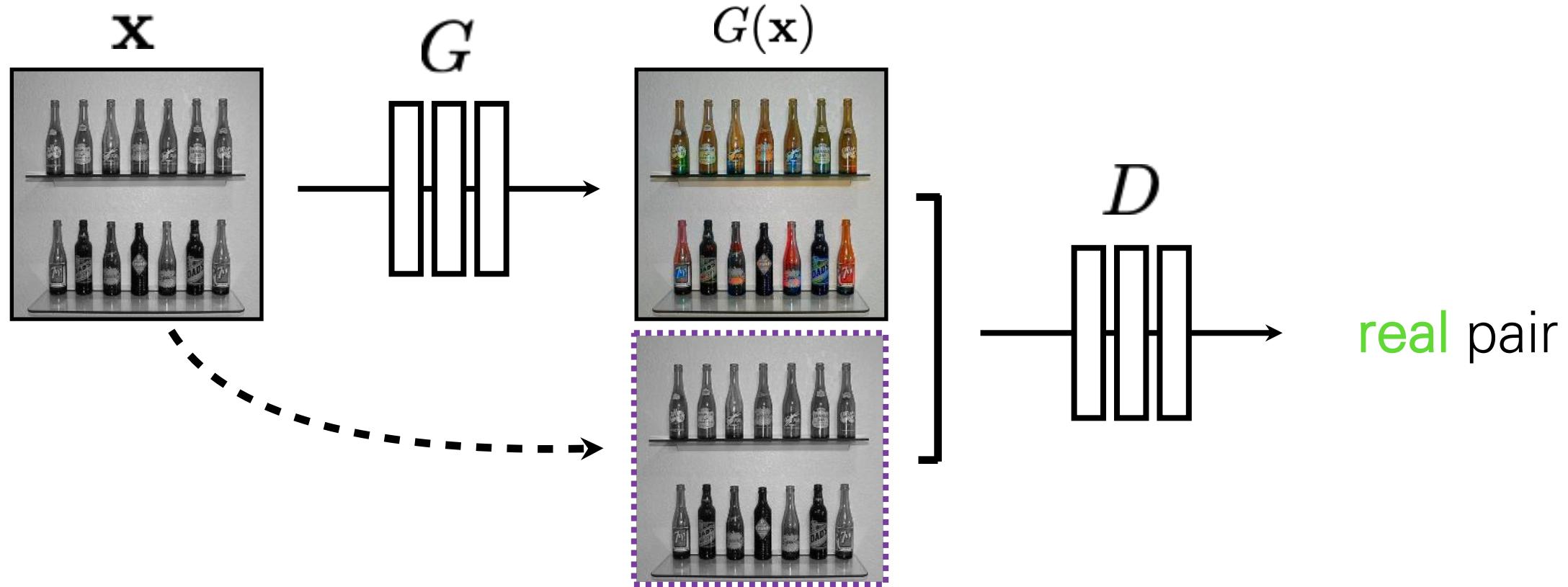


Conditional GANs / pix2pix



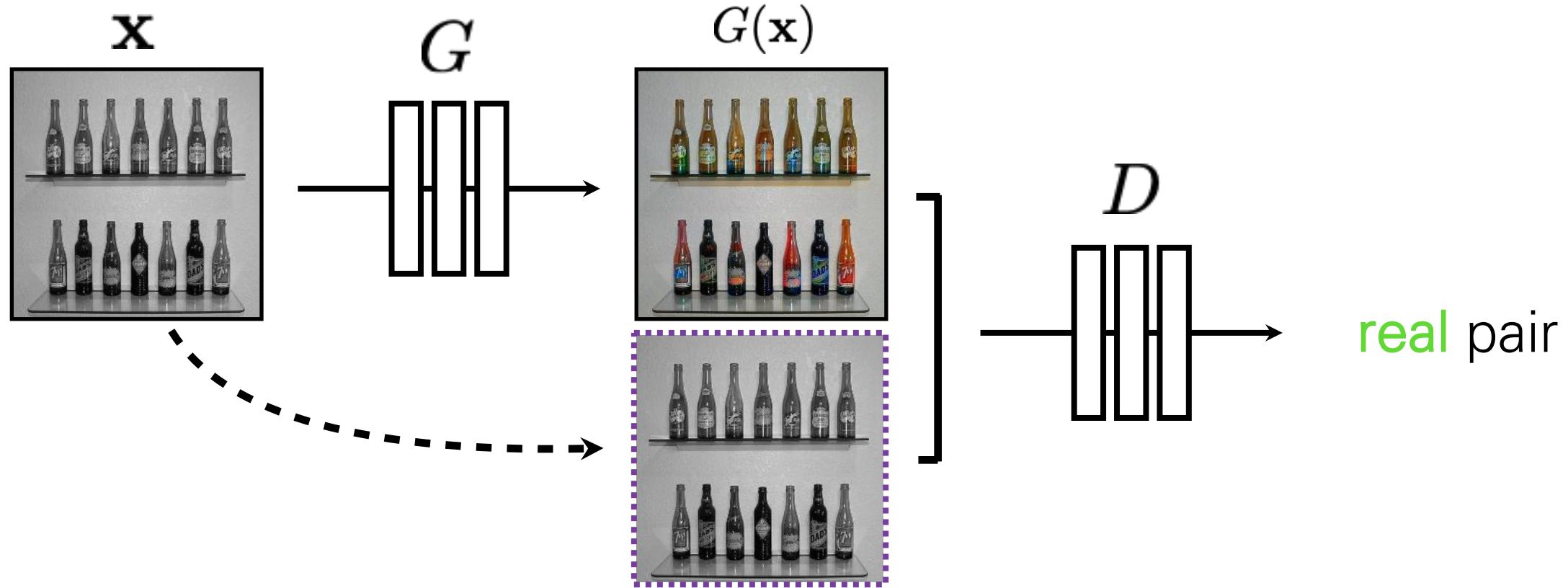
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))]$$

Conditional GANs / pix2pix



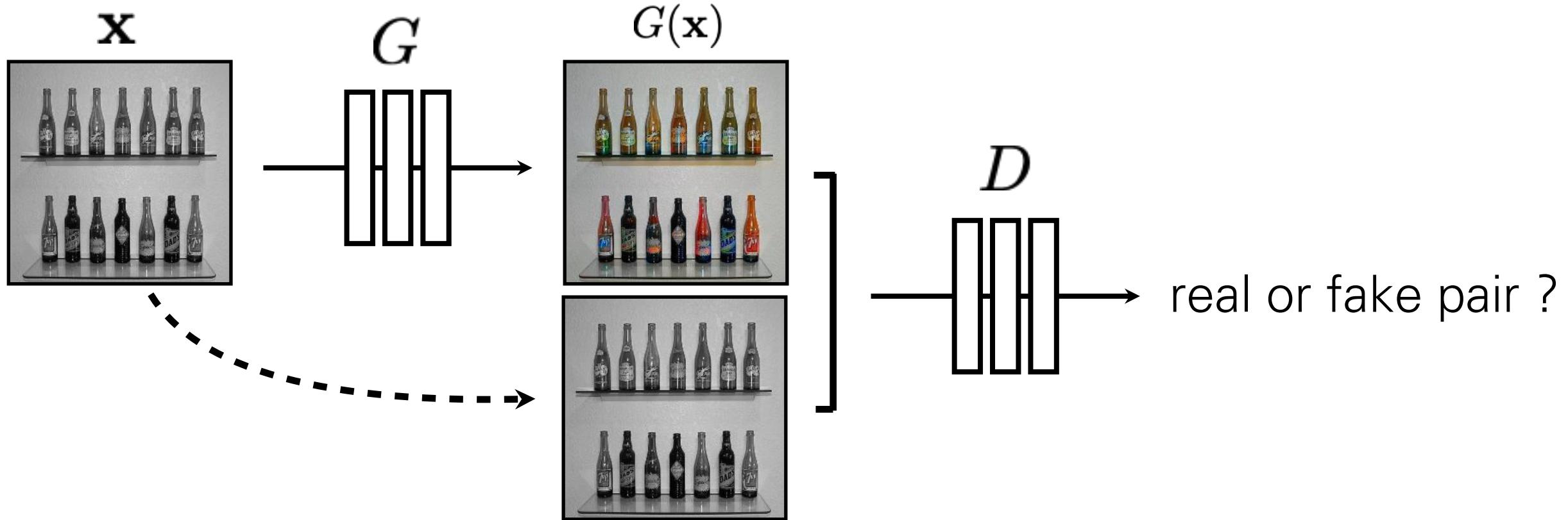
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))]$$

Conditional GANs / pix2pix



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

Conditional GANs / pix2pix



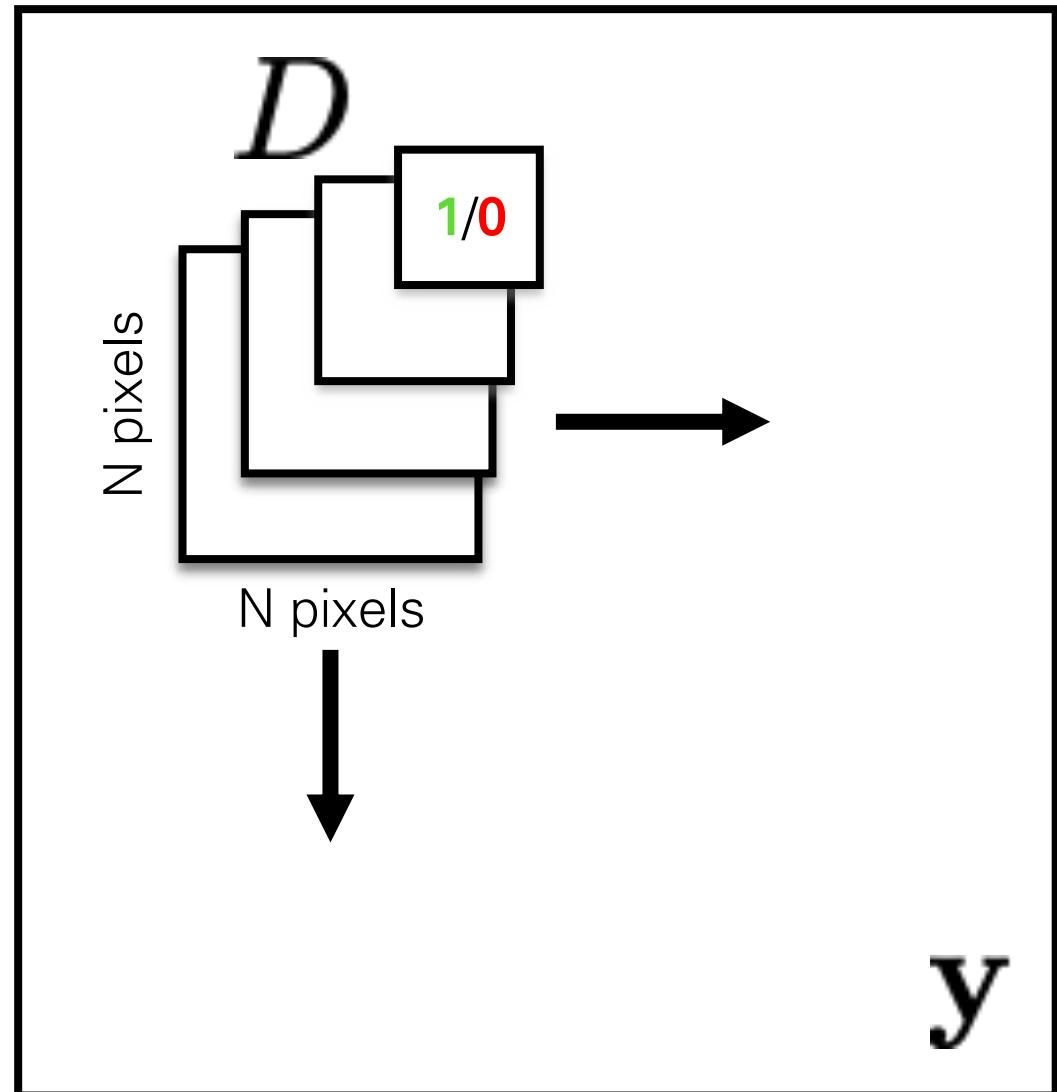
$$\boxed{\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))]}$$

Conditional GANs / pix2pix

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Conditional GANs / pix2pix

Shrinking the capacity:
Patch Discriminator

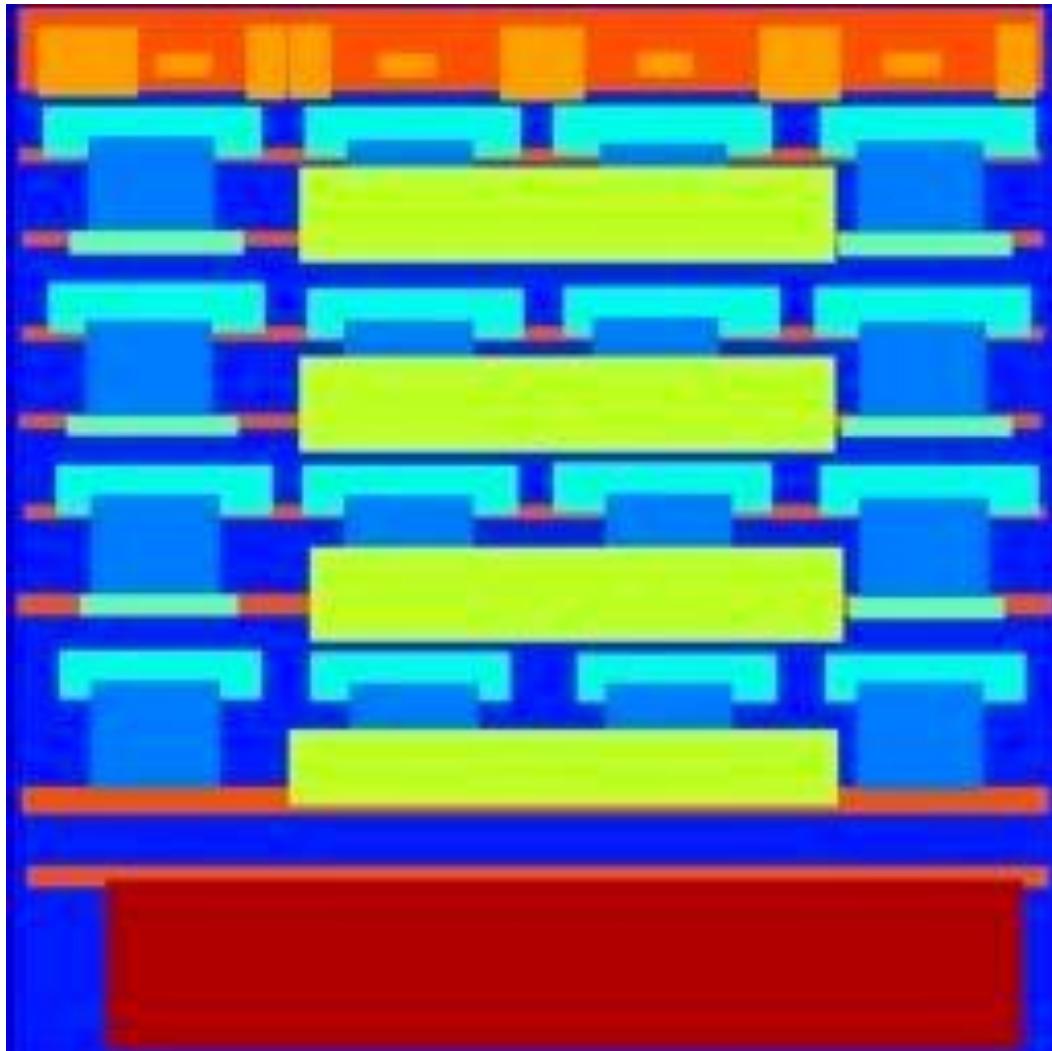


Rather than penalizing if output image looks fake, penalize if each overlapping patch in output looks fake

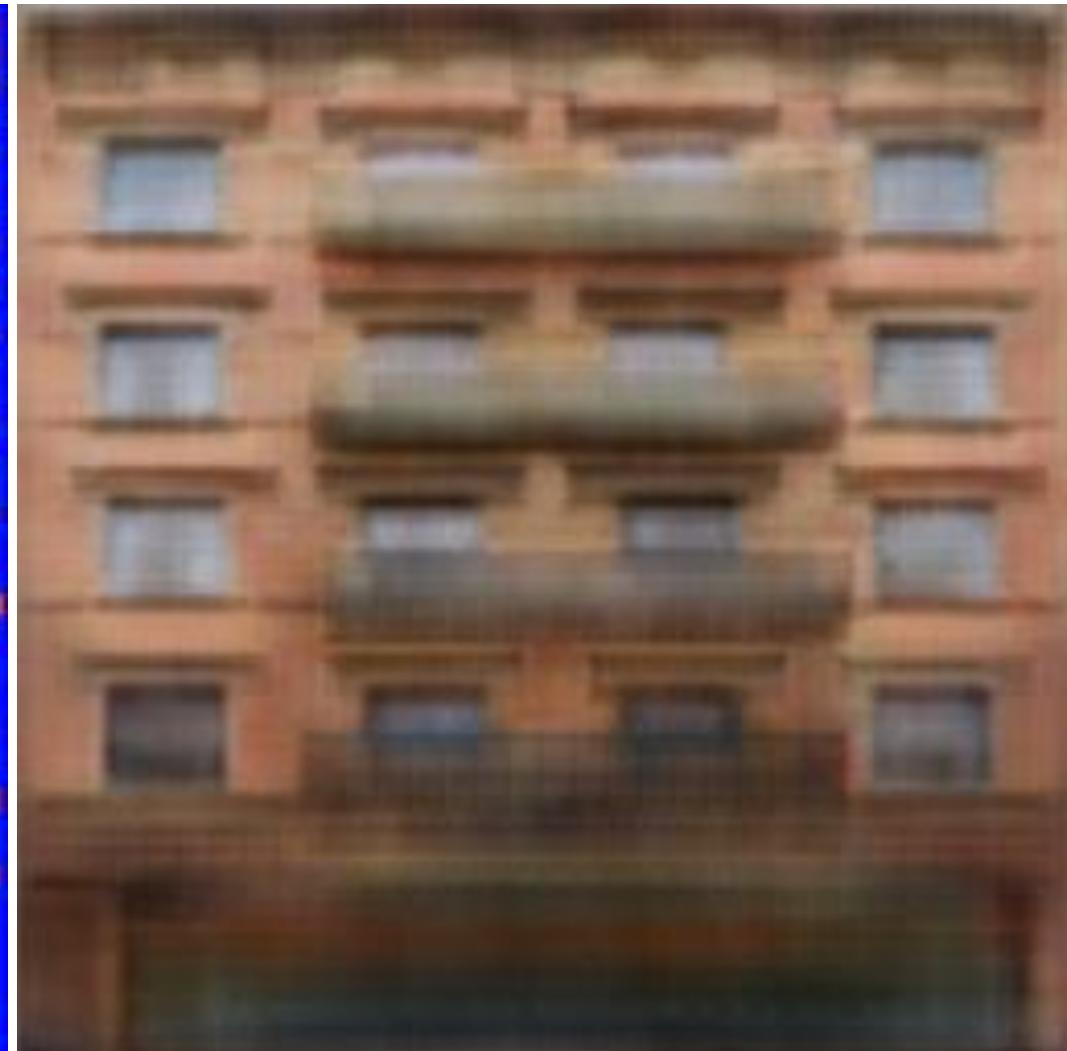
[Li & Wand 2016]
[Shrivastava et al. 2017]
[Isola et al. 2017]

Conditional GANs / pix2pix

Input

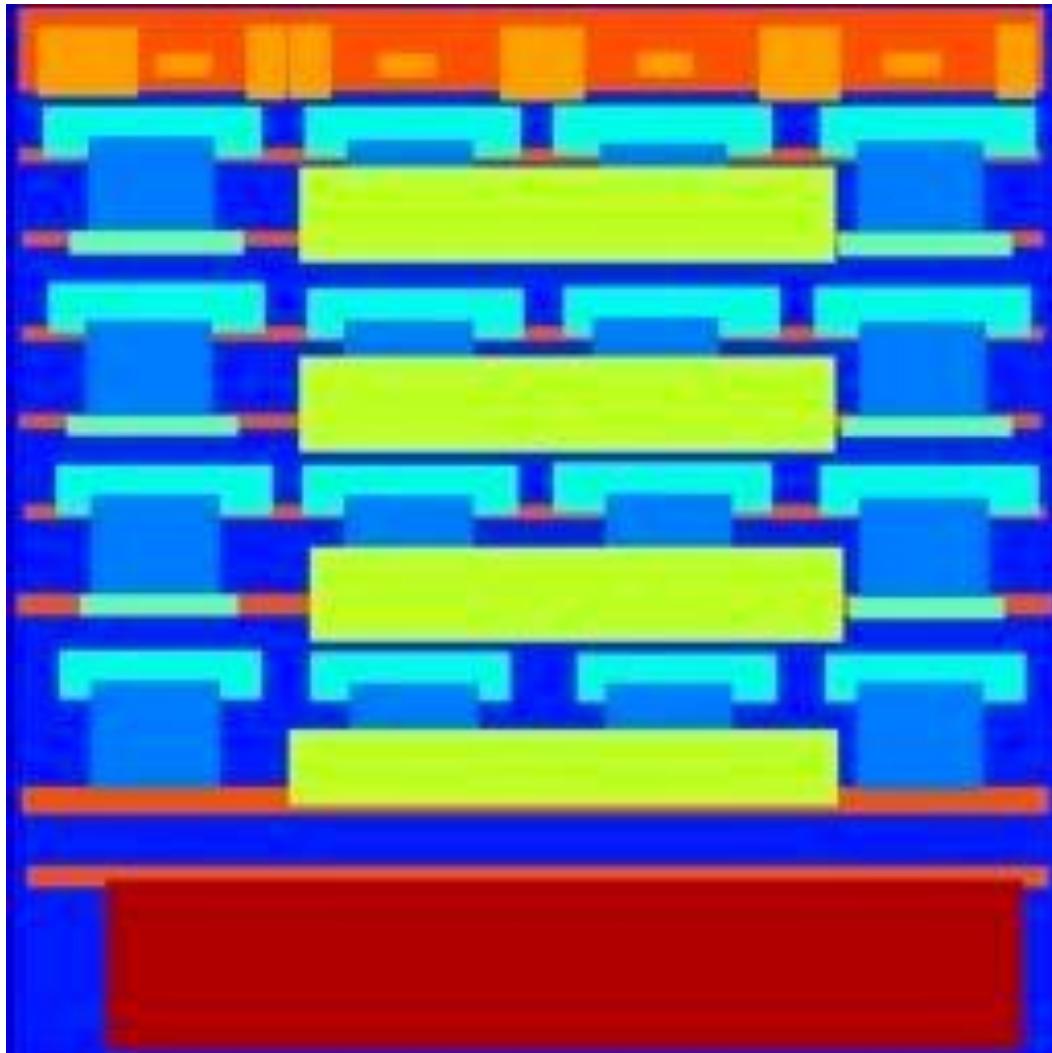


1x1 Discriminator



Conditional GANs / pix2pix

Input

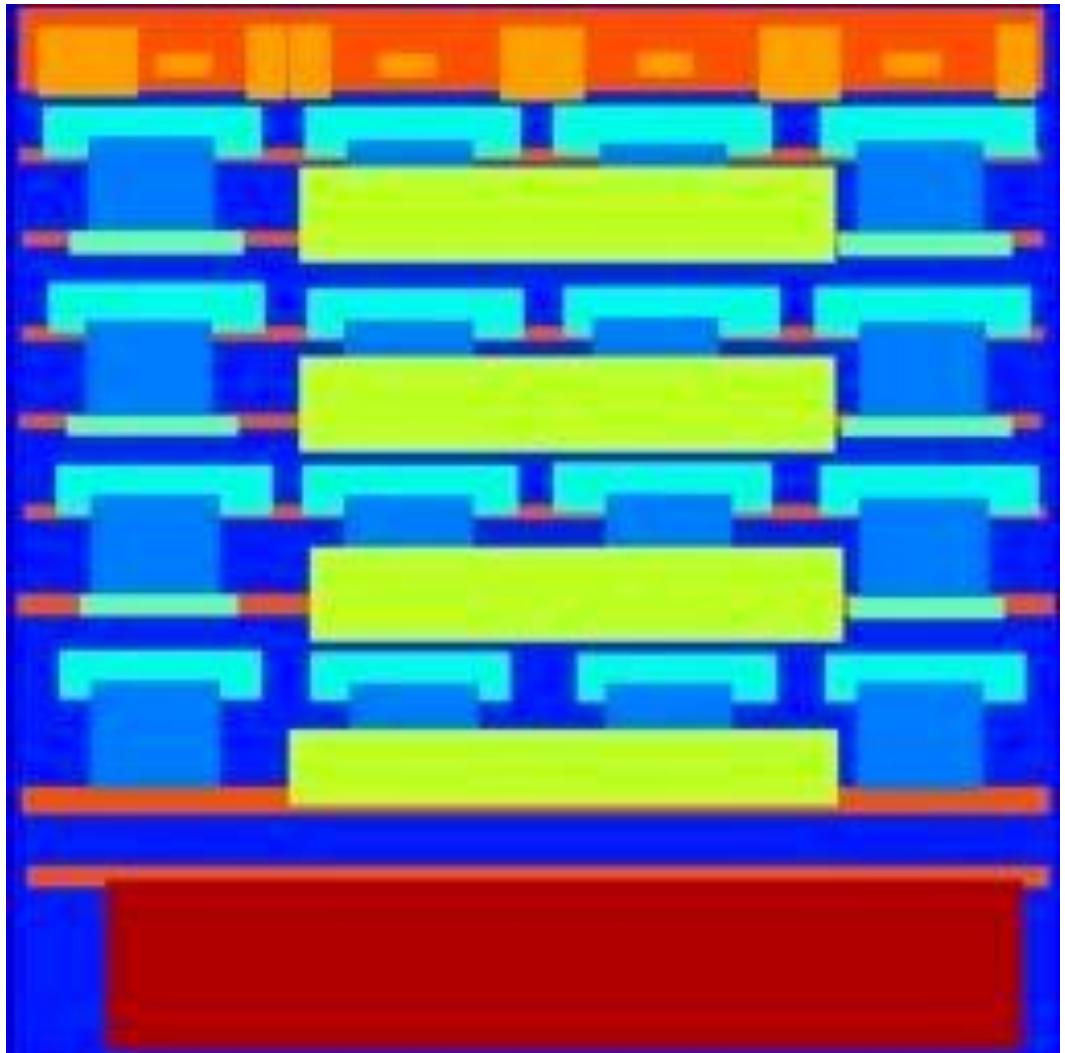


16x16 Discriminator



Conditional GANs / pix2pix

Input

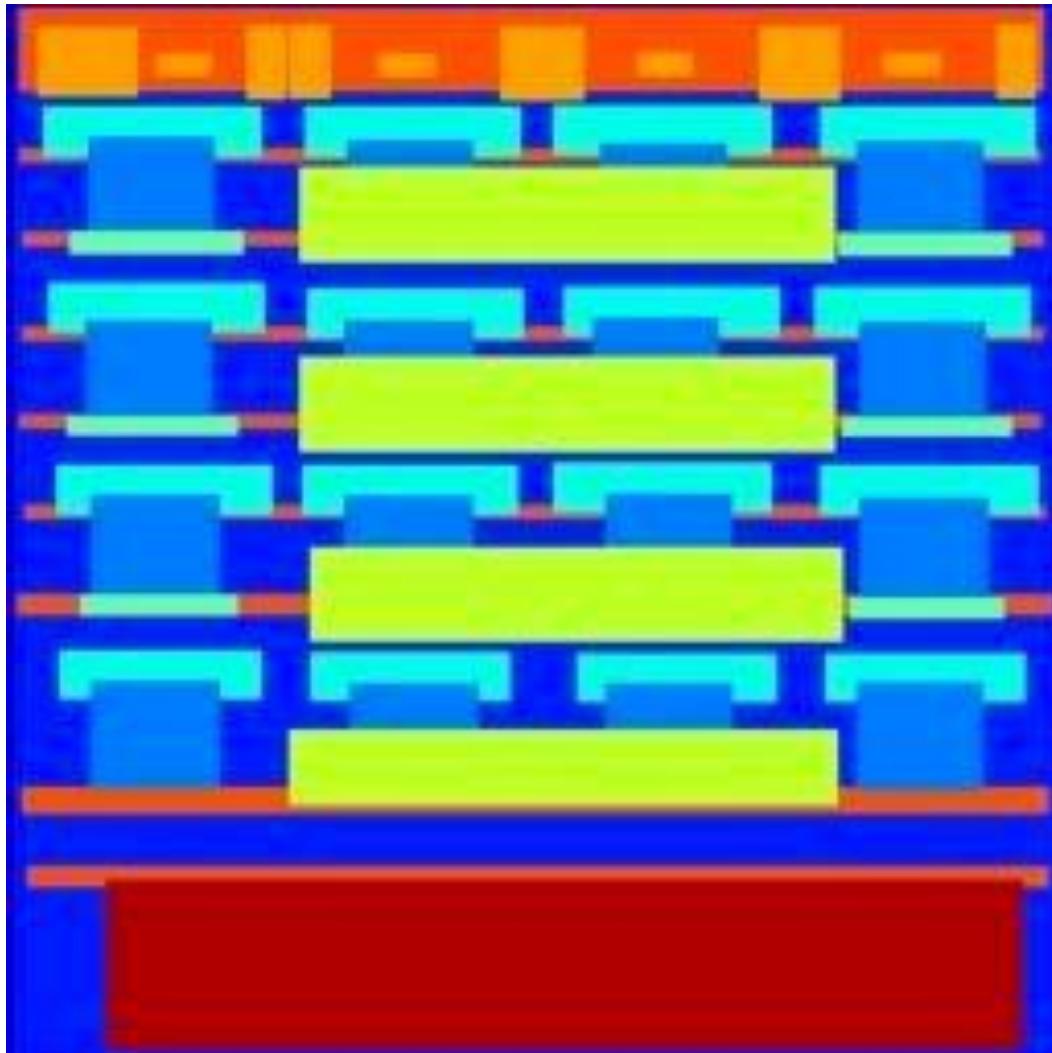


70x70 Discriminator



Conditional GANs / pix2pix

Input

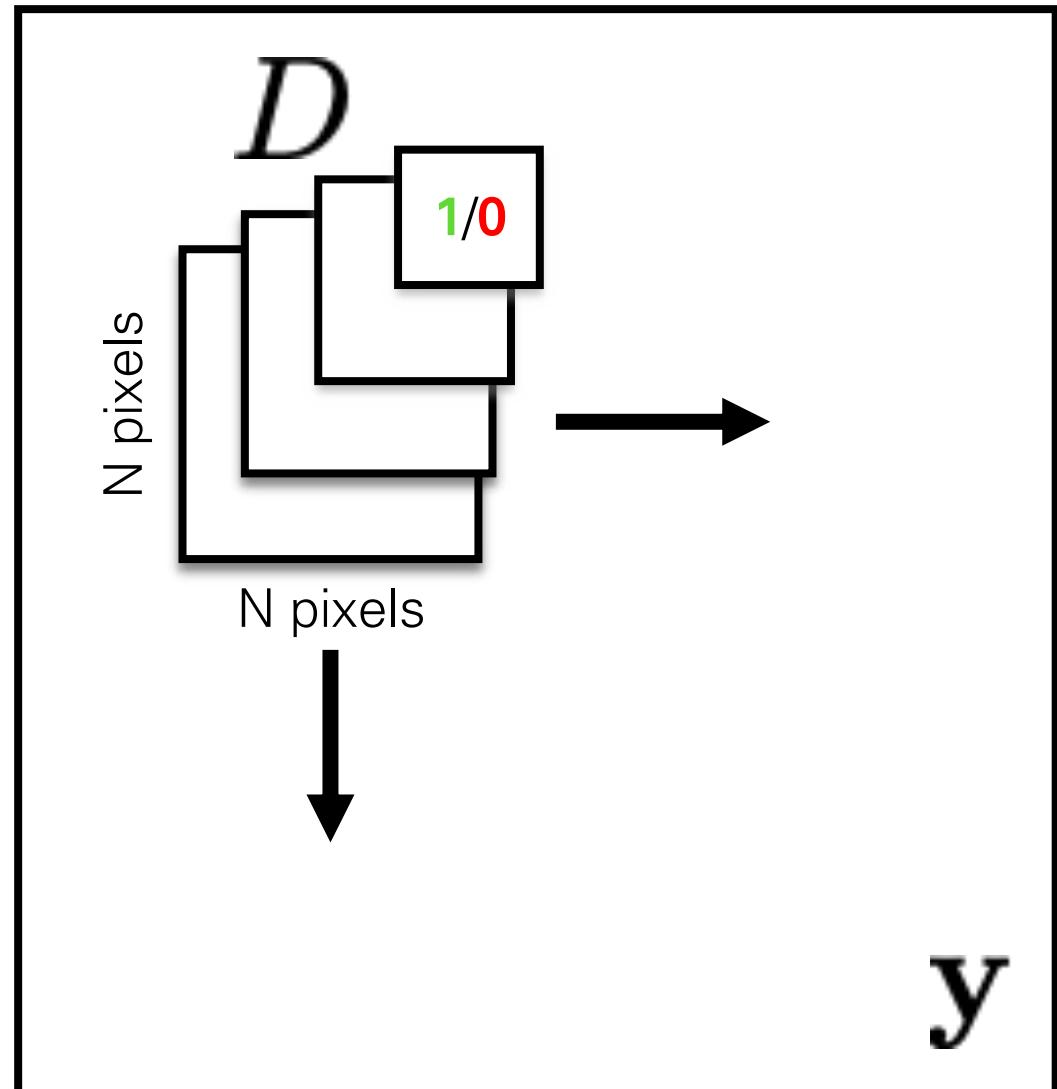


16x16 Discriminator



Conditional GANs / pix2pix

Shrinking the capacity:
Patch Discriminator

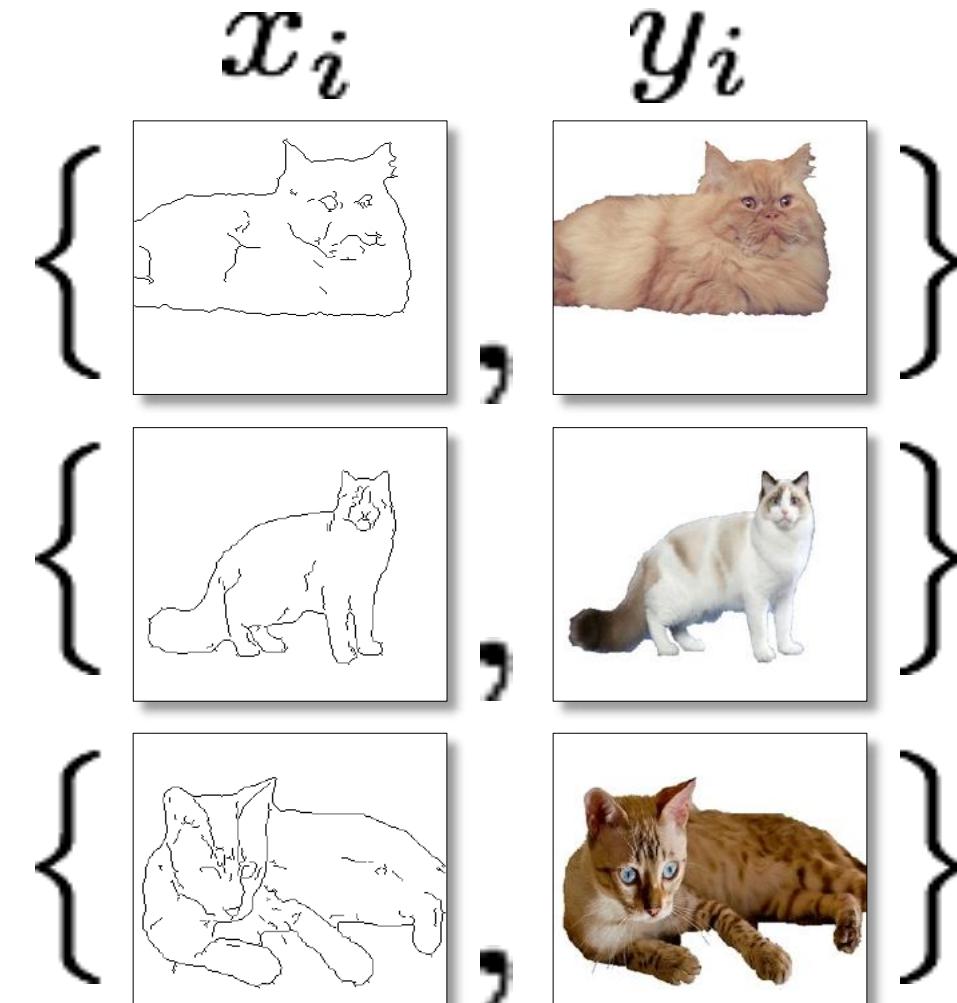


Rather than penalizing if output image looks fake, penalize if each overlapping patch in output looks fake

- Faster, fewer parameters
- More supervised observations
- Applies to arbitrarily large images

[Li & Wand 2016]
[Shrivastava et al. 2017]
[Isola et al. 2017]
120

Conditional GANs / pix2pix



Conditional GANs / pix2pix

BW → Color

Input



Output



Input



Output



Input

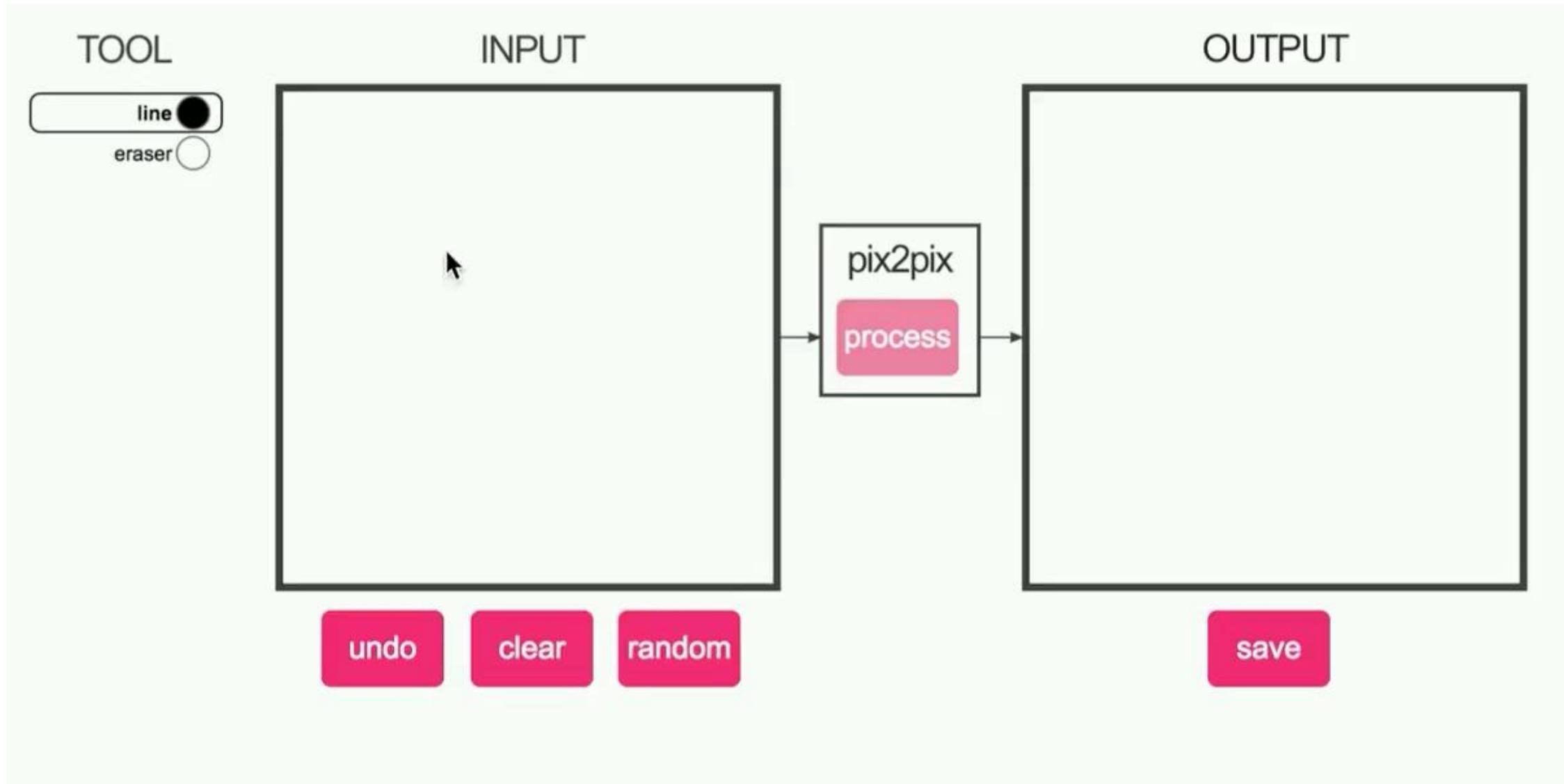


Output

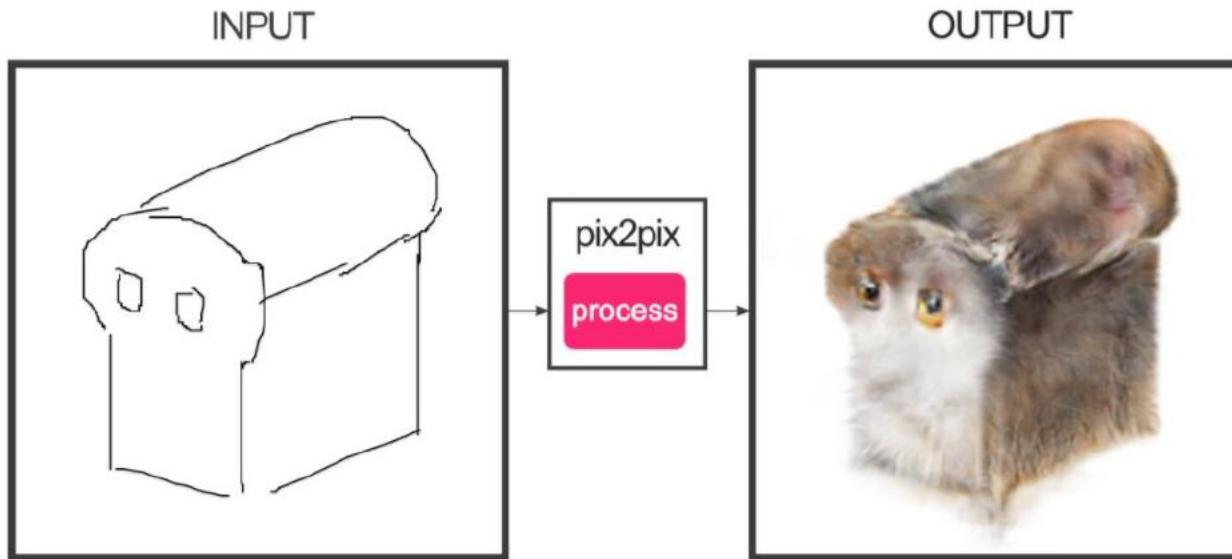


Data from [Russakovsky et al. 2015]

Conditional GANs / pix2pix #edges2cats [Chris Hesse]



Conditional GANs / pix2pix



Ivy Tasi @ivymyt



Vitaly Vidmirov @vvid

Conditional GANs / pix2pix

BW → Color

Input



Output



Input



Output



Input



Output



Conditional GANs / pix2pix

Structured Prediction

Input
 \mathbf{x}



Output
 $\hat{\mathbf{y}}$



Target
 \mathbf{y}



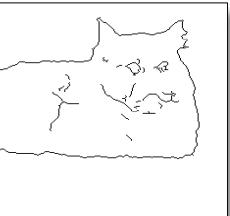
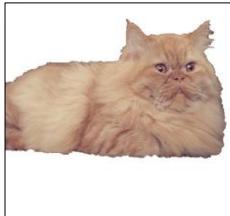
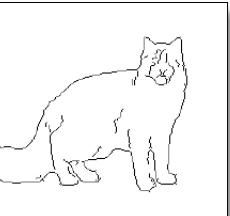
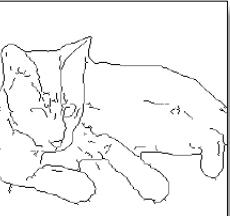
$$L(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2$$

Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, **Cycle-Consistent Adversarial Networks**
- GANs and Representations
- Applications

Cycle-Consistent Adversarial Networks

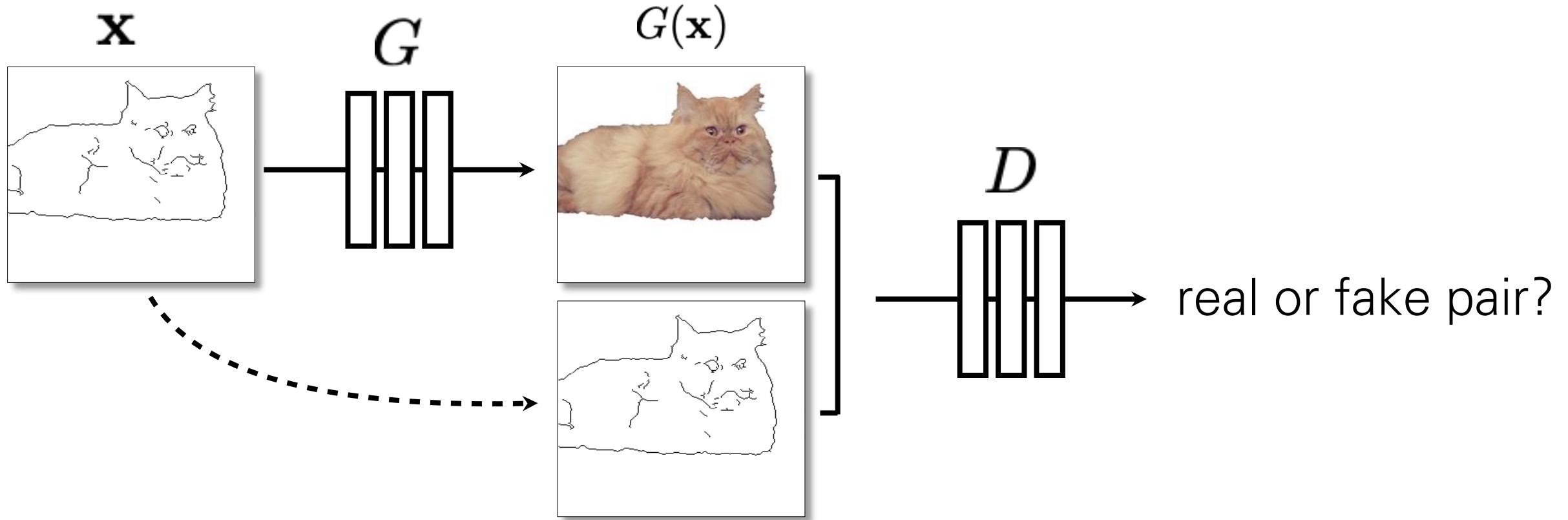
Paired data

x_i	y_i
	
	
	
⋮	⋮

Unpaired data

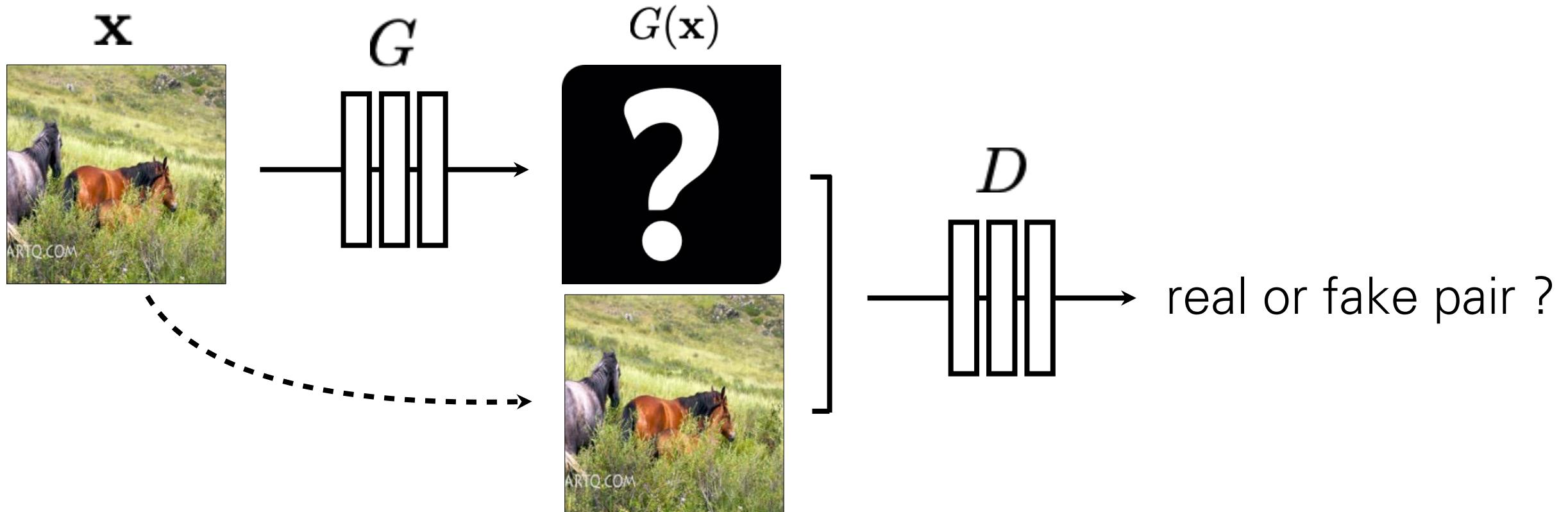
X	Y
	
	
	
⋮	⋮

Cycle-Consistent Adversarial Networks



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

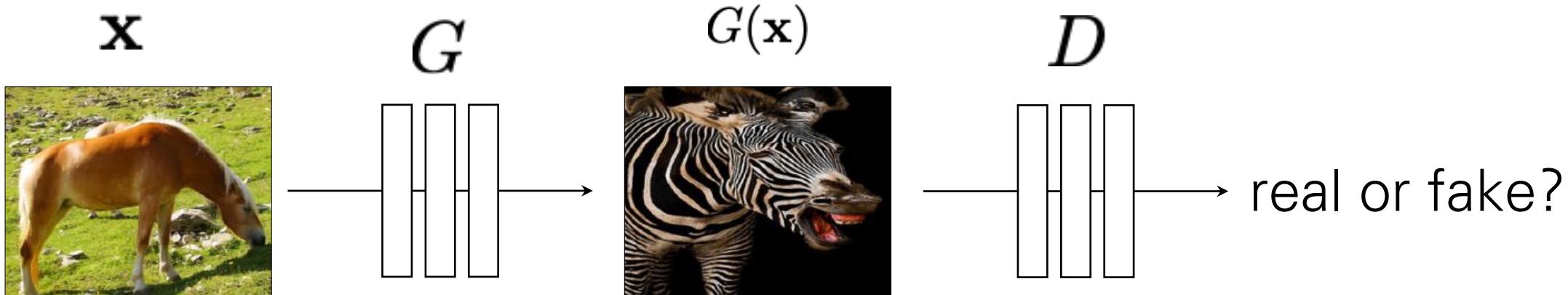
Cycle-Consistent Adversarial Networks



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

No input-output pairs!

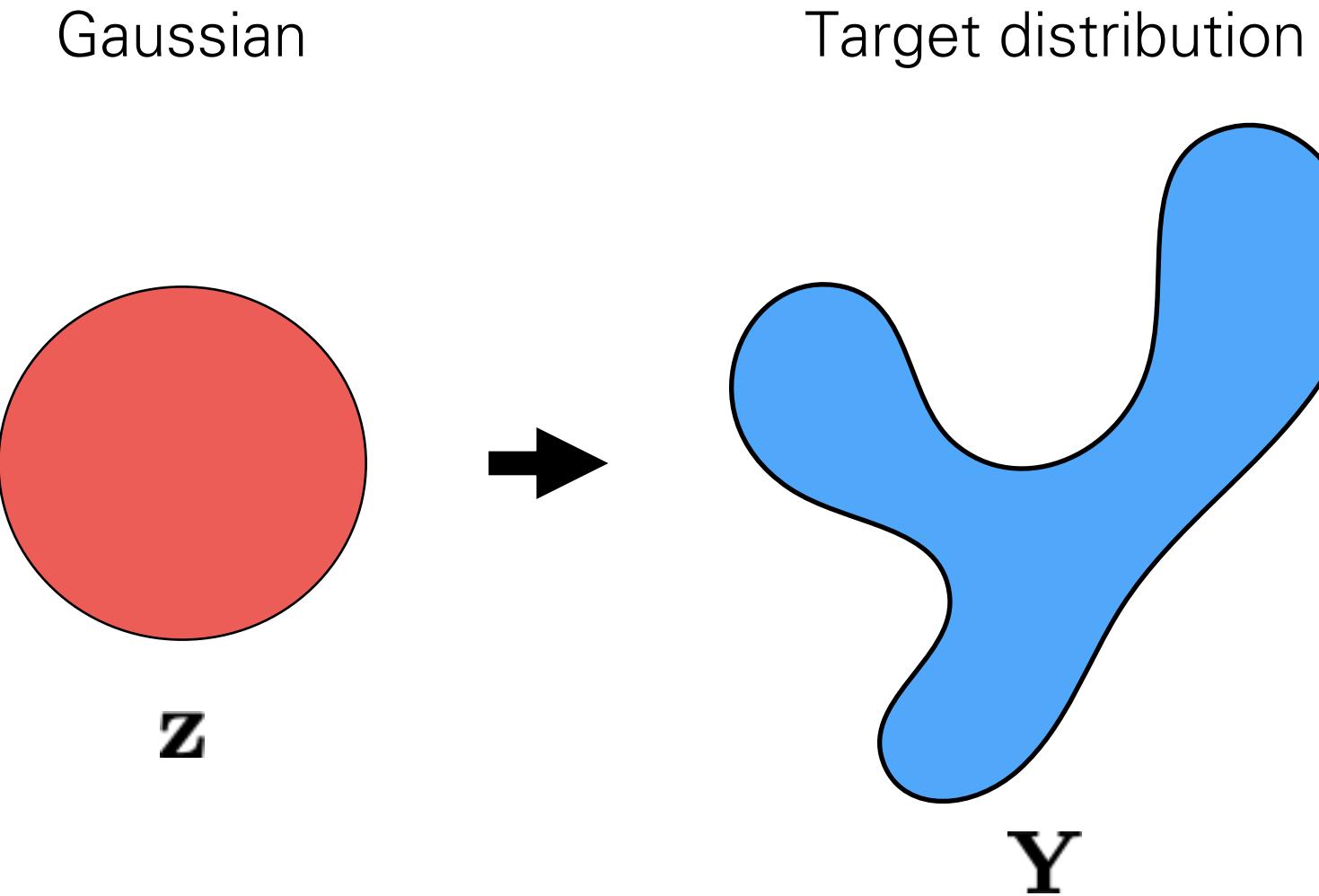
Cycle-Consistent Adversarial Networks



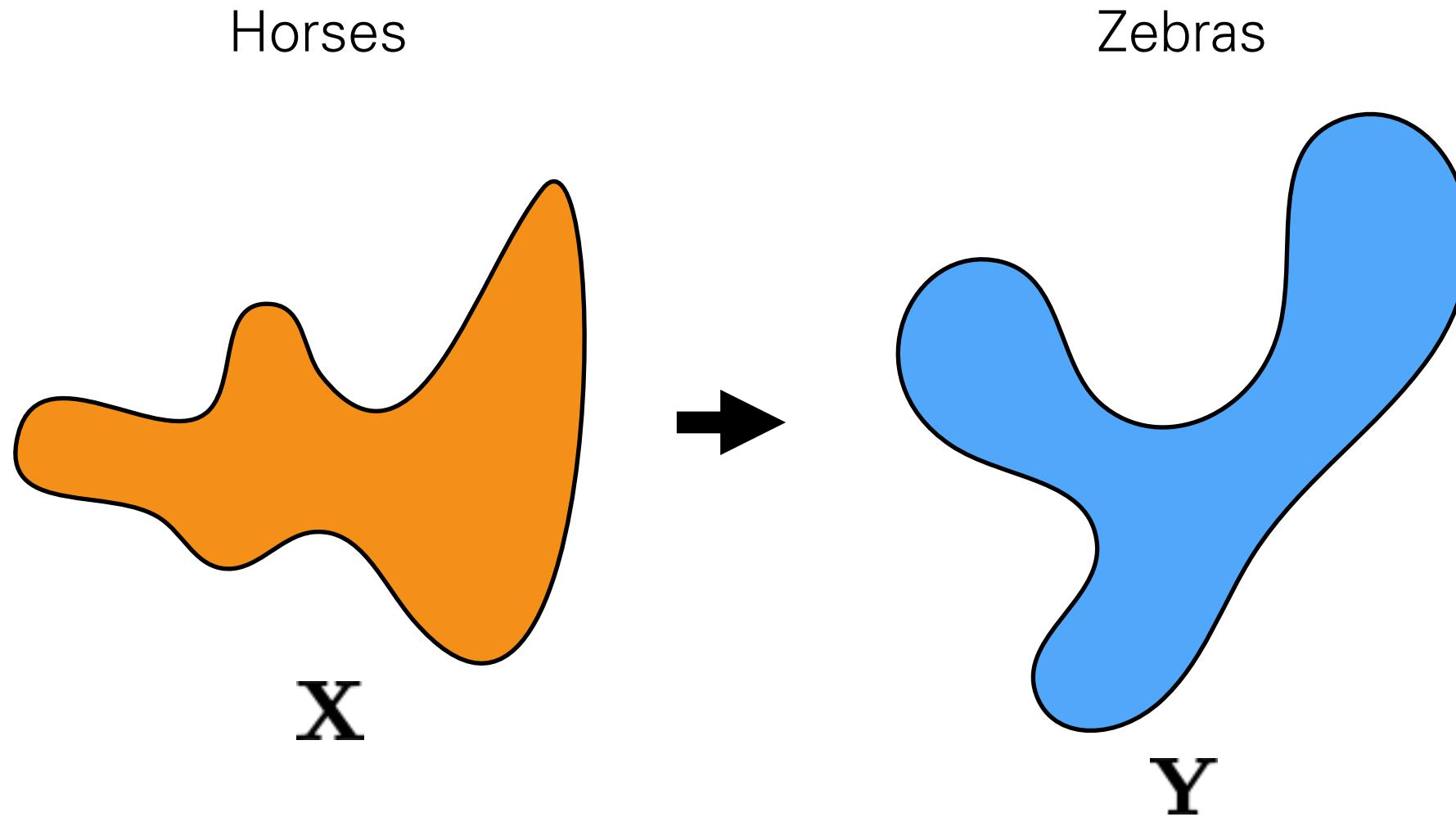
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

- Usually loss functions check if output matches a target instance
- GAN loss checks if output is part of an admissible set

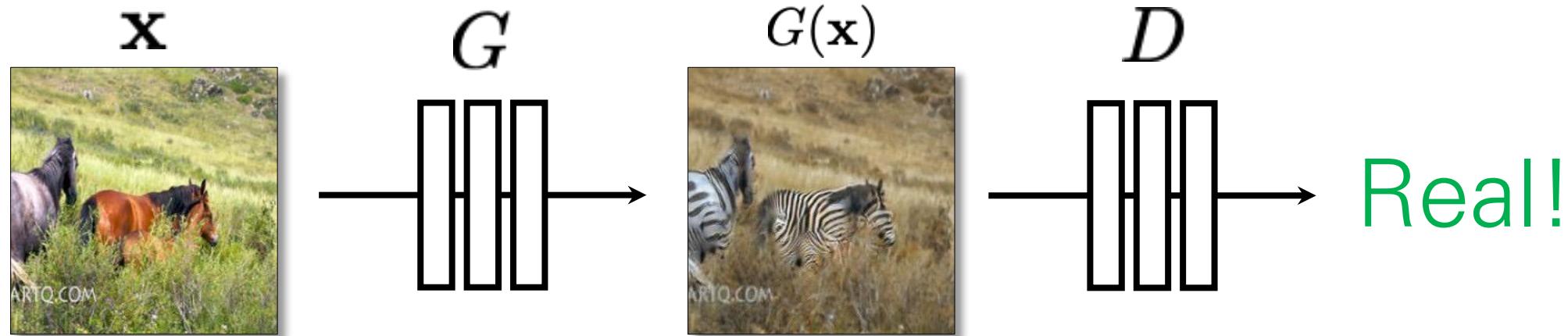
Cycle-Consistent Adversarial Networks



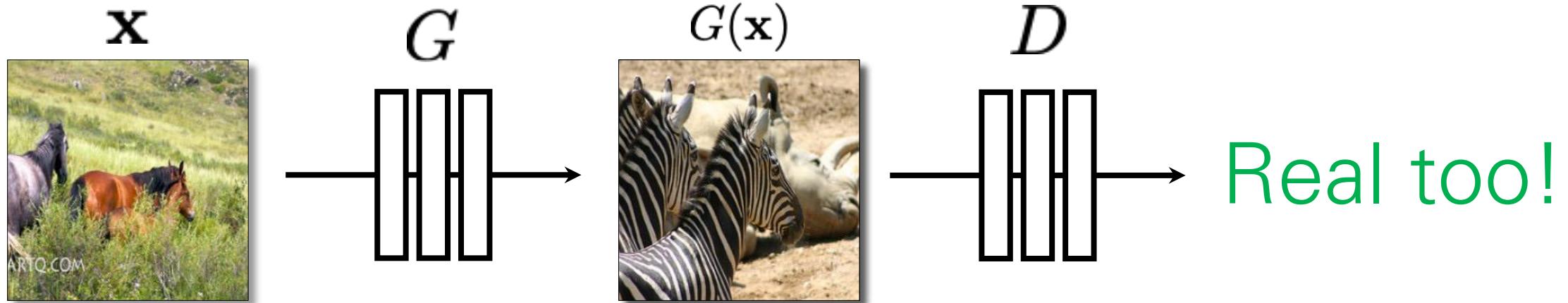
Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks

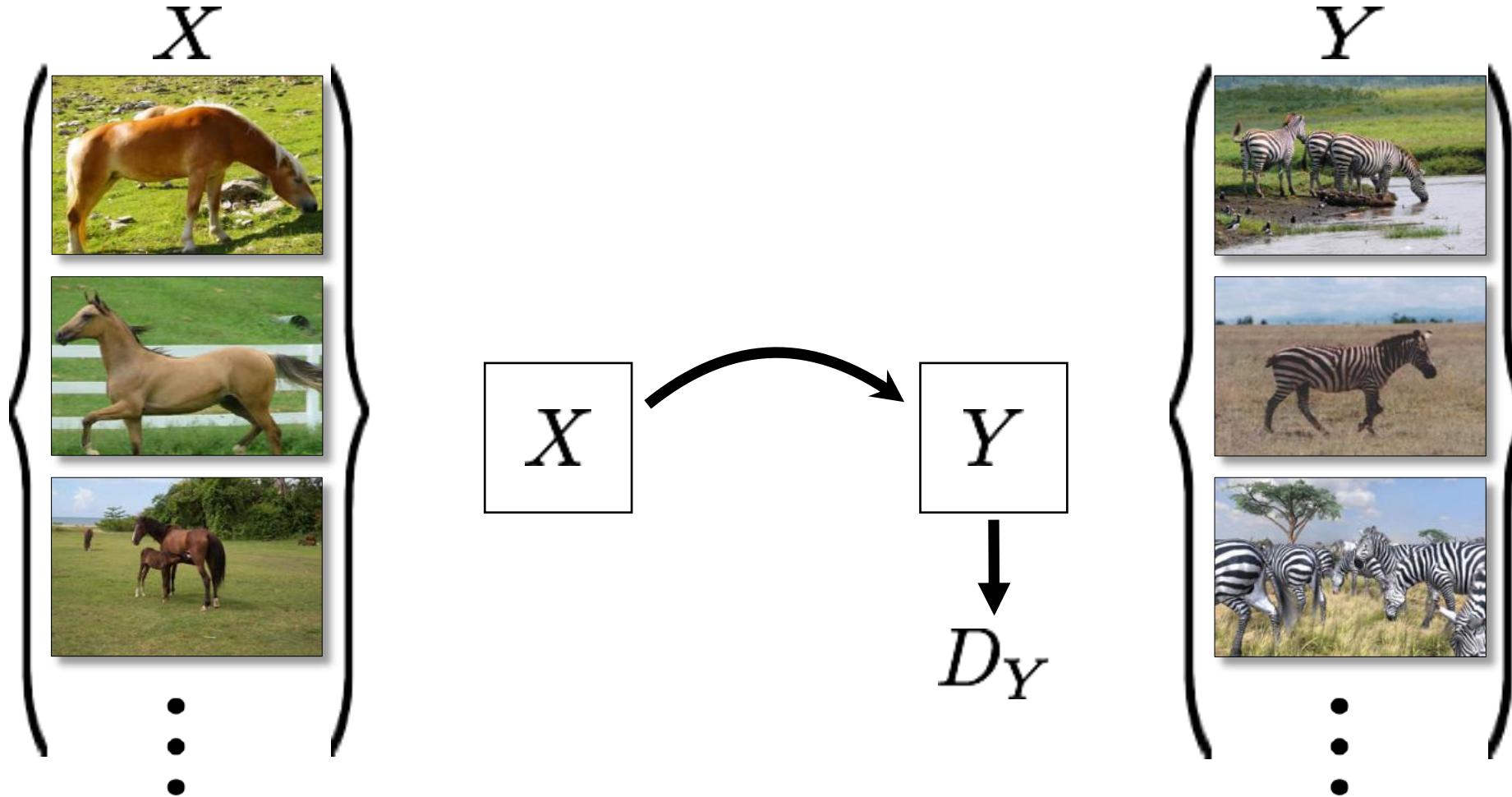


Cycle-Consistent Adversarial Networks



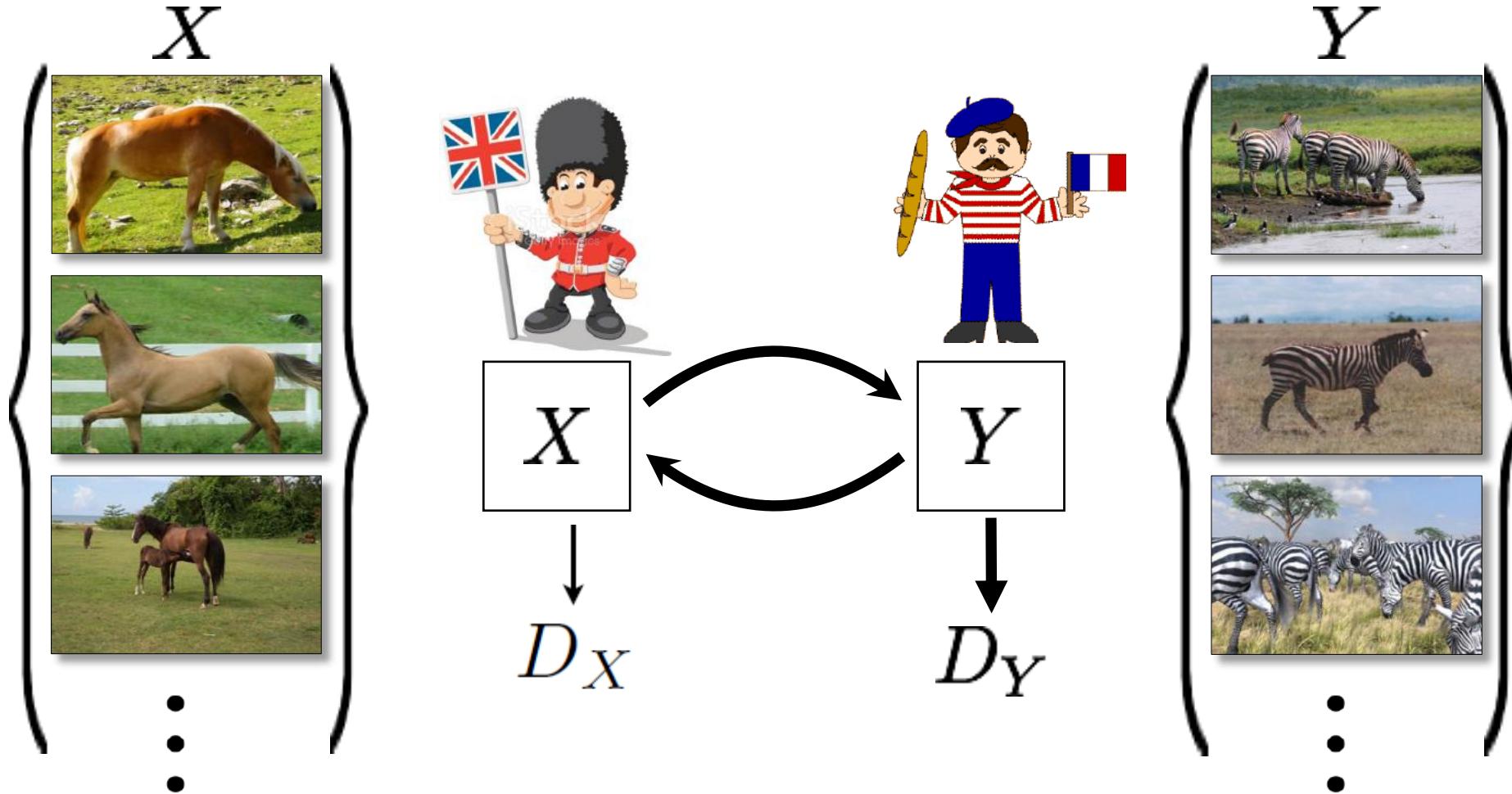
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks

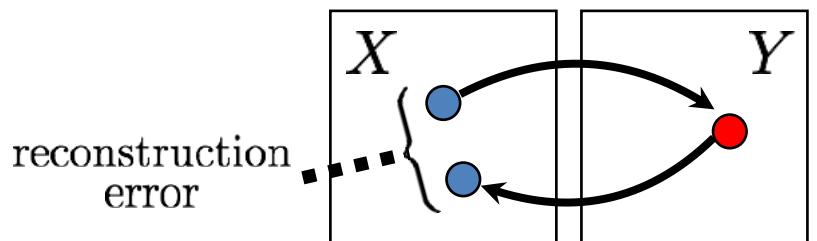
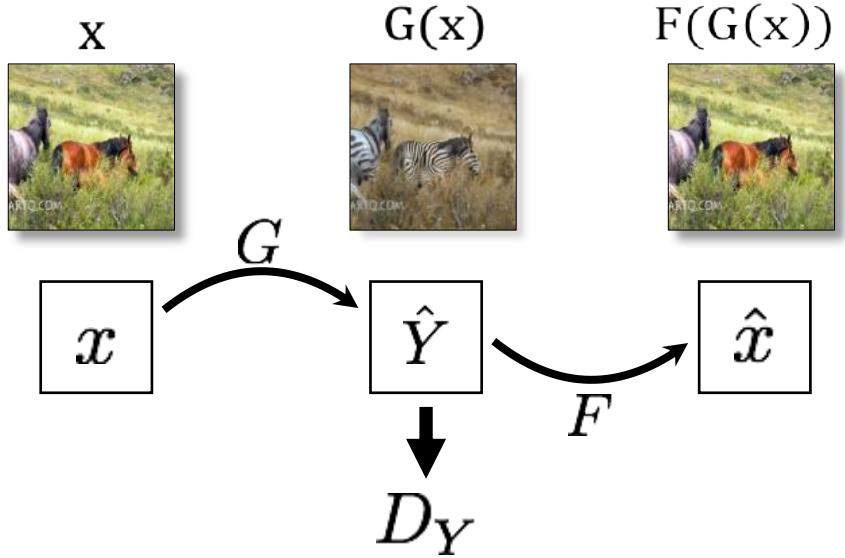


[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks

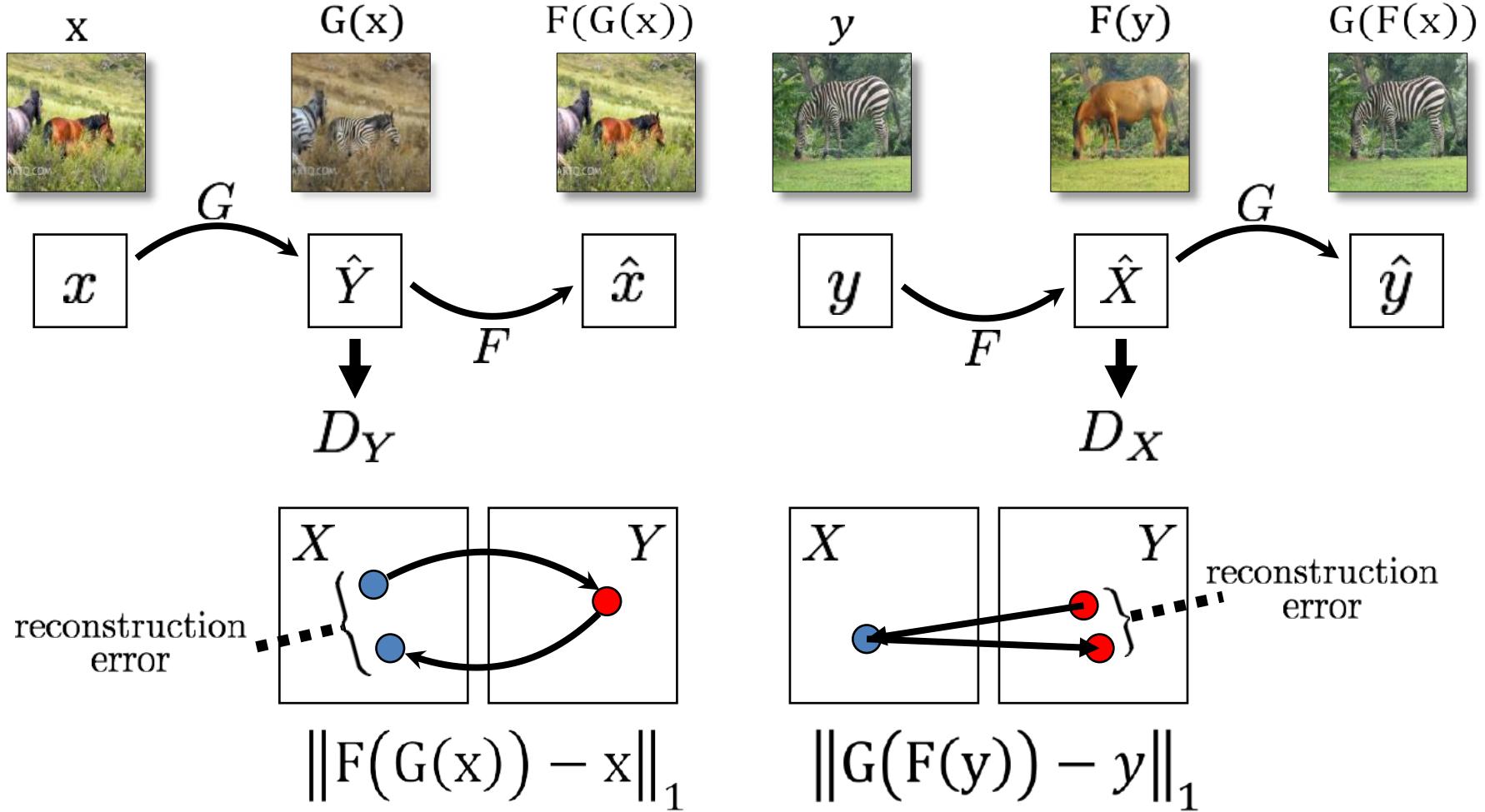


Cycle Consistency Loss



$$\|F(G(x)) - x\|_1$$

Cycle Consistency Loss



Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks



Photograph
@ Alexei Efros



Monet



Van Gogh



Cezanne



Ukiyo-e

Input



Monet



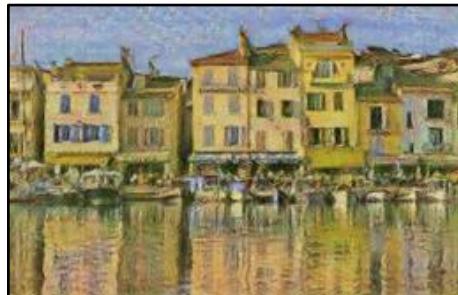
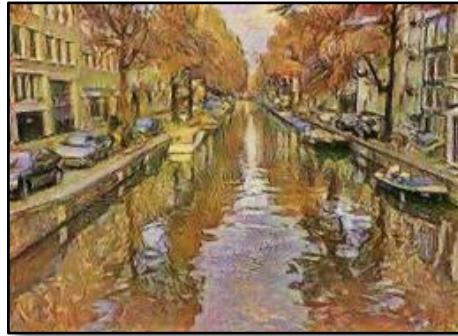
Van Gogh



Cezanne



Ukiyo-e



Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks



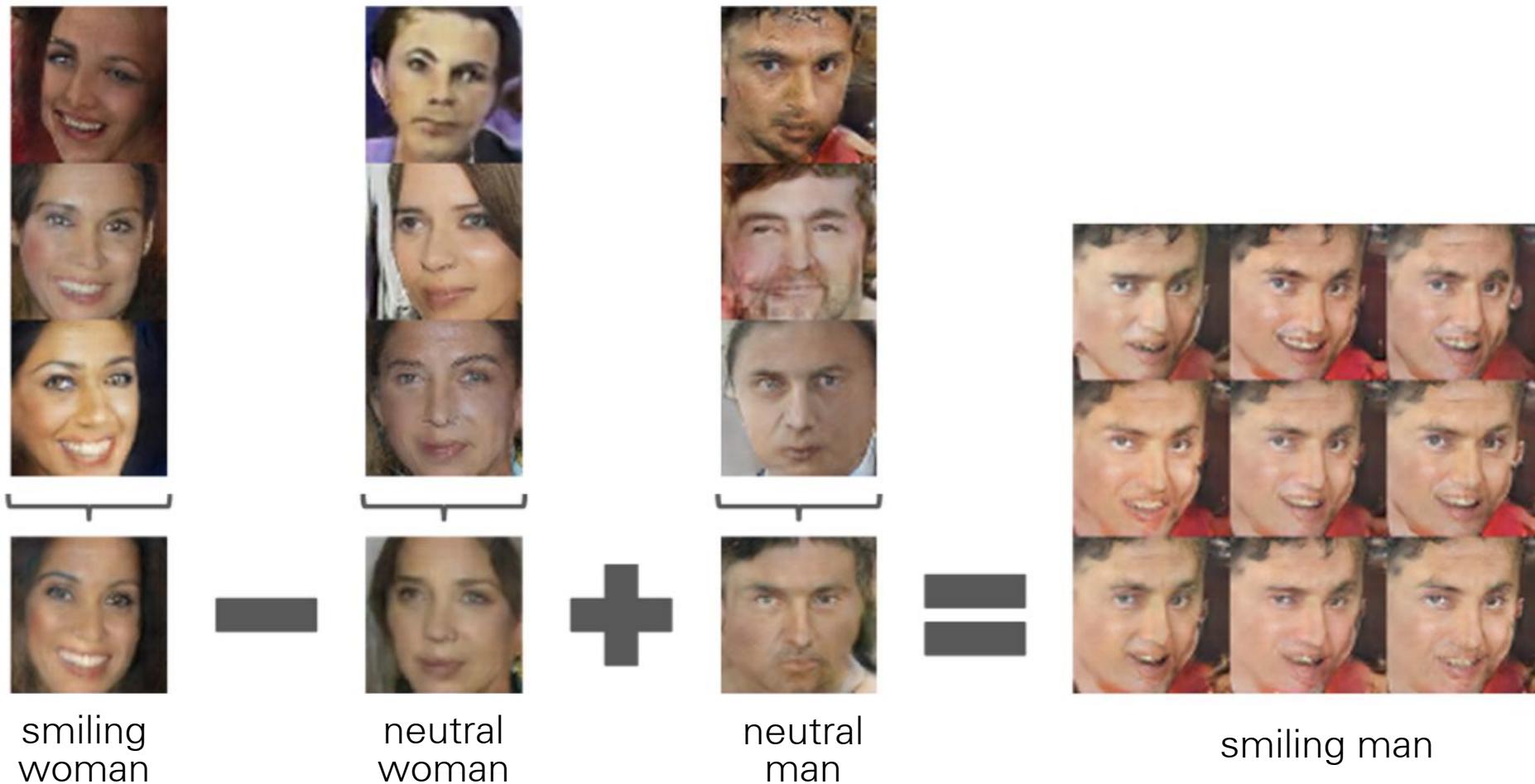
Cycle-Consistent Adversarial Networks



Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- **GANs and Representations**
- Applications

DCGAN Revisited: Vector Arithmetic

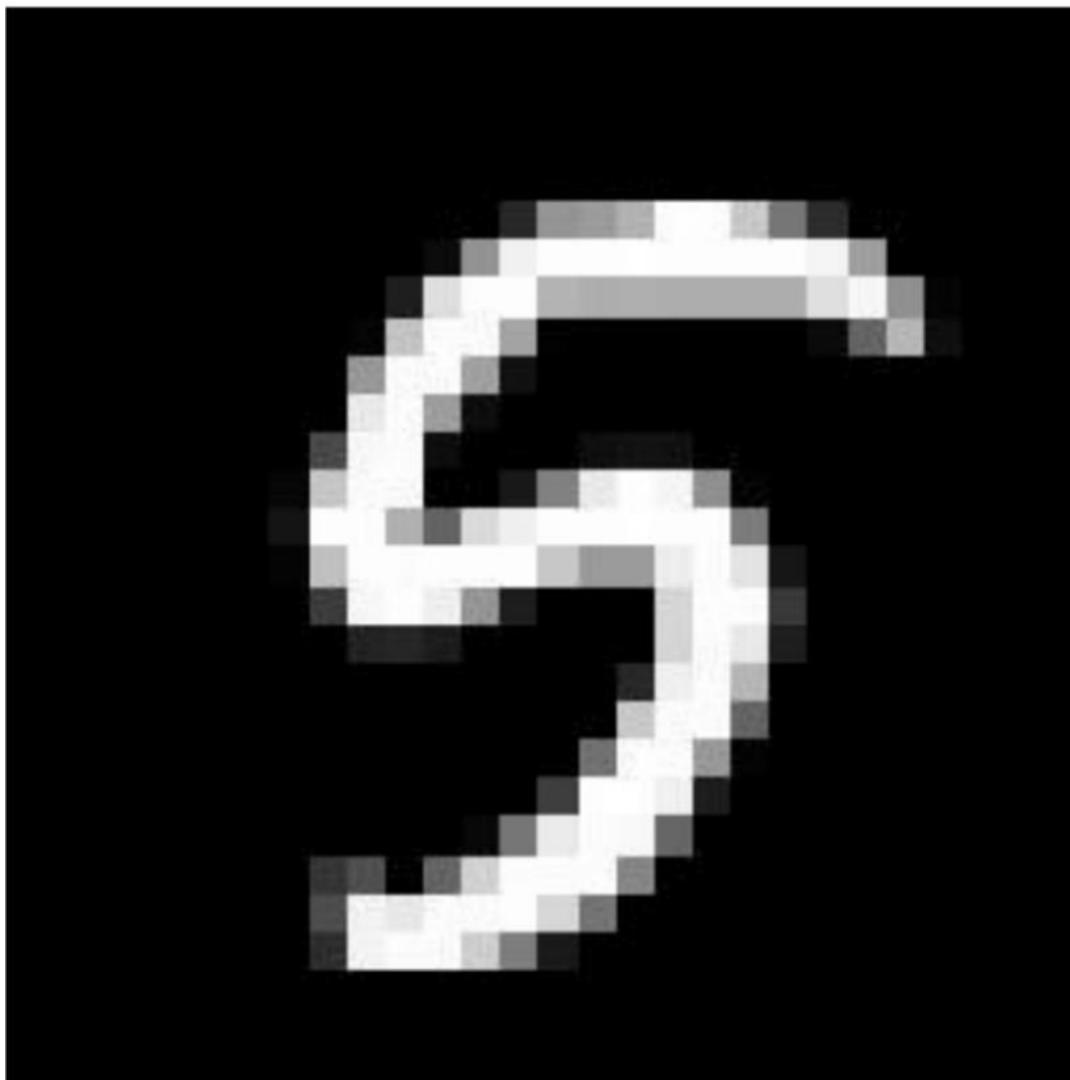


[Radford et al 2016]

GANs for unsupervised feature learning

- InfoGAN (Information Maximizing GAN)
- BiGAN (Bidirectional Generative Adversarial Networks)
ALI (Adversarially Learned Inference)
 - BigBiGAN (Big Bidirectional Generative Adversarial Networks)

InfoGAN



```
array([[151, 157, 250, ..., 20, 0, 0],  
       [148, 161, 242, ..., 15, 0, 0],  
       [235, 228, 255, ..., 3, 0, 0],  
       ...,  
       [252, 254, 176, ..., 240, 253, 253],  
       [253, 253, 253, ..., 253, 200, 200],  
       [253, 253, 253, ..., 253, 200, 200]]  
     , dtype=uint8)
```

InfoGAN

```
array([[151, 157, 250, ..., 20, 0, 0],  
       [148, 161, 242, ..., 15, 0, 0],  
       [235, 228, 255, ..., 3, 0, 0],  
       ...,  
       [252, 254, 176, ..., 240, 253, 253],  
       [253, 253, 253, ..., 253, 200, 200],  
       [253, 253, 253, ..., 253, 200, 200]]  
, dtype=uint8)
```

Data: x

Simple factors interact to create complex observations.

Digit type: “5”

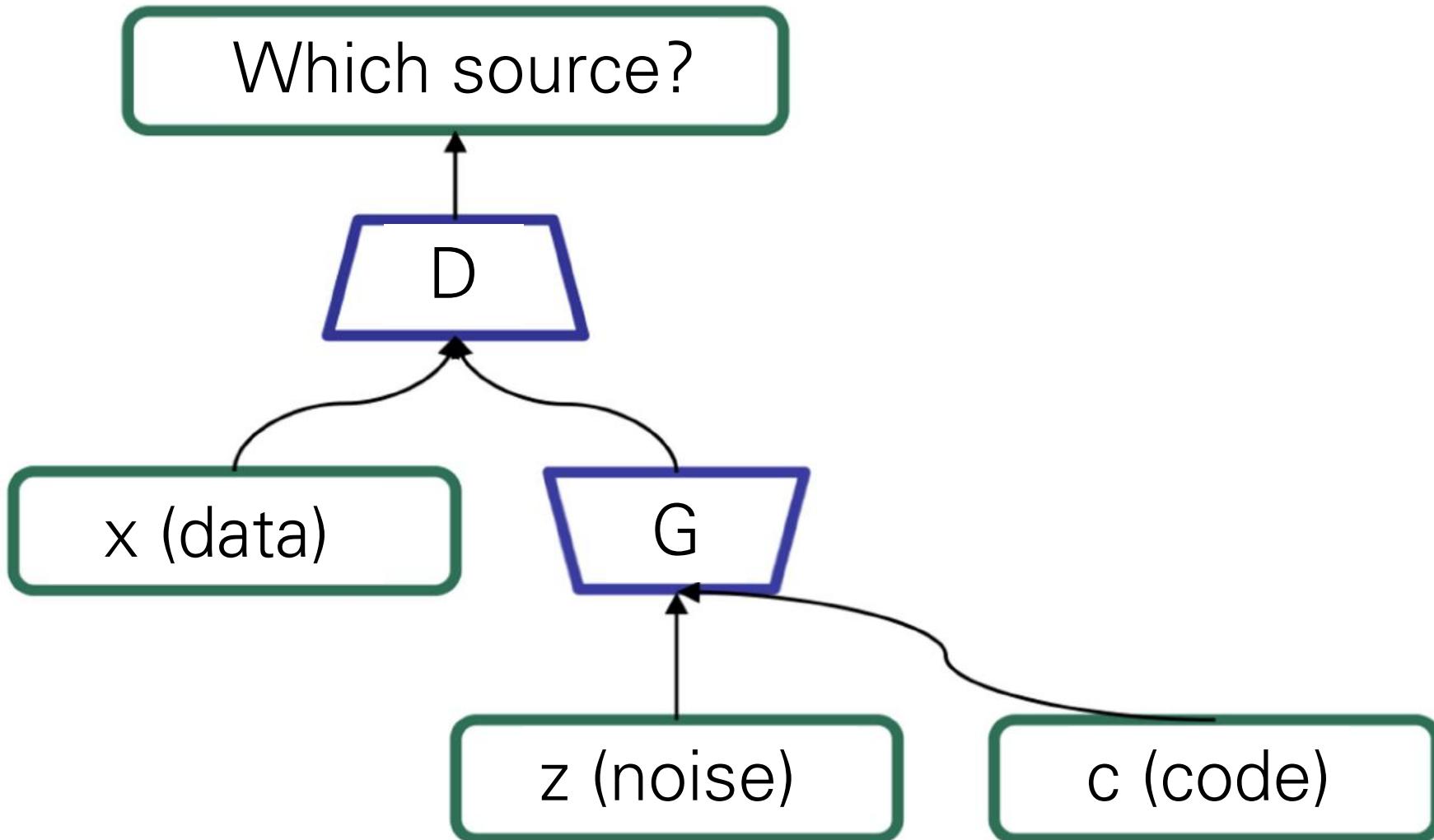
Rotation: Tilting to the right

Width: Medium

.....

Latent code: c

InfoGAN



InfoGAN

- Simple idea: Independent factors in latent code should maximally explain variations in generated images
- Formally: We want to maximize the mutual information between latent code and generated images:

$$\begin{aligned}\max_G I(c; x) &= H(x) - H(x|c) \\ &= H(c) - H(c|x)\end{aligned}$$

where $x = G(z, c)$

InfoGAN

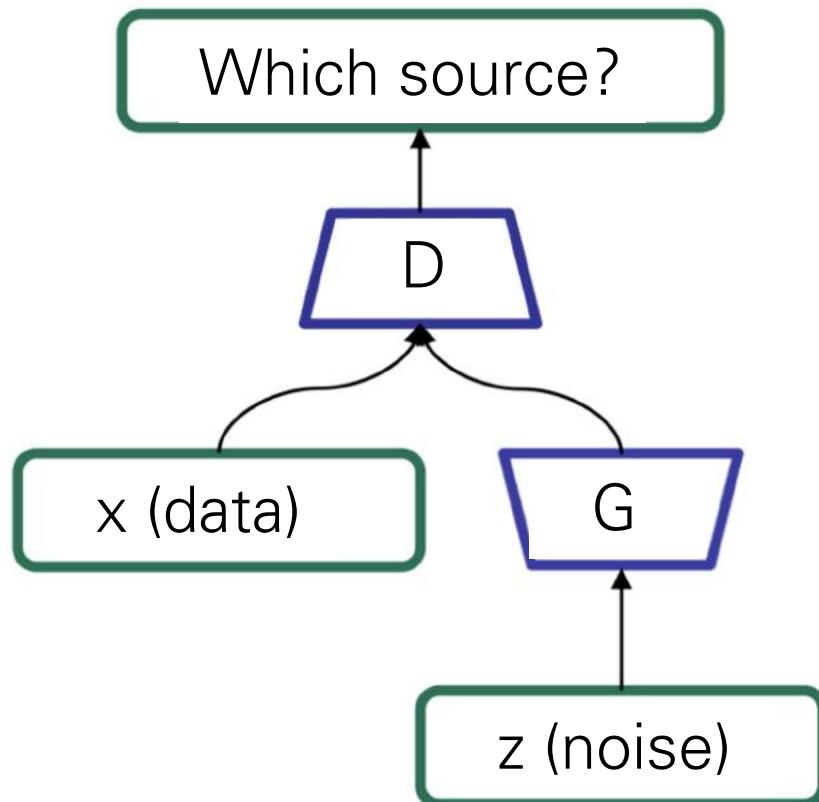
- Mutual information can be maximized easily with a variational lower bound:

$$\begin{aligned} I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\ &= H(c) + \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log P(c'|x)]] \\ &= H(c) + \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + \underbrace{D_{\text{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} \\ &\geq H(c) + \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] \end{aligned}$$

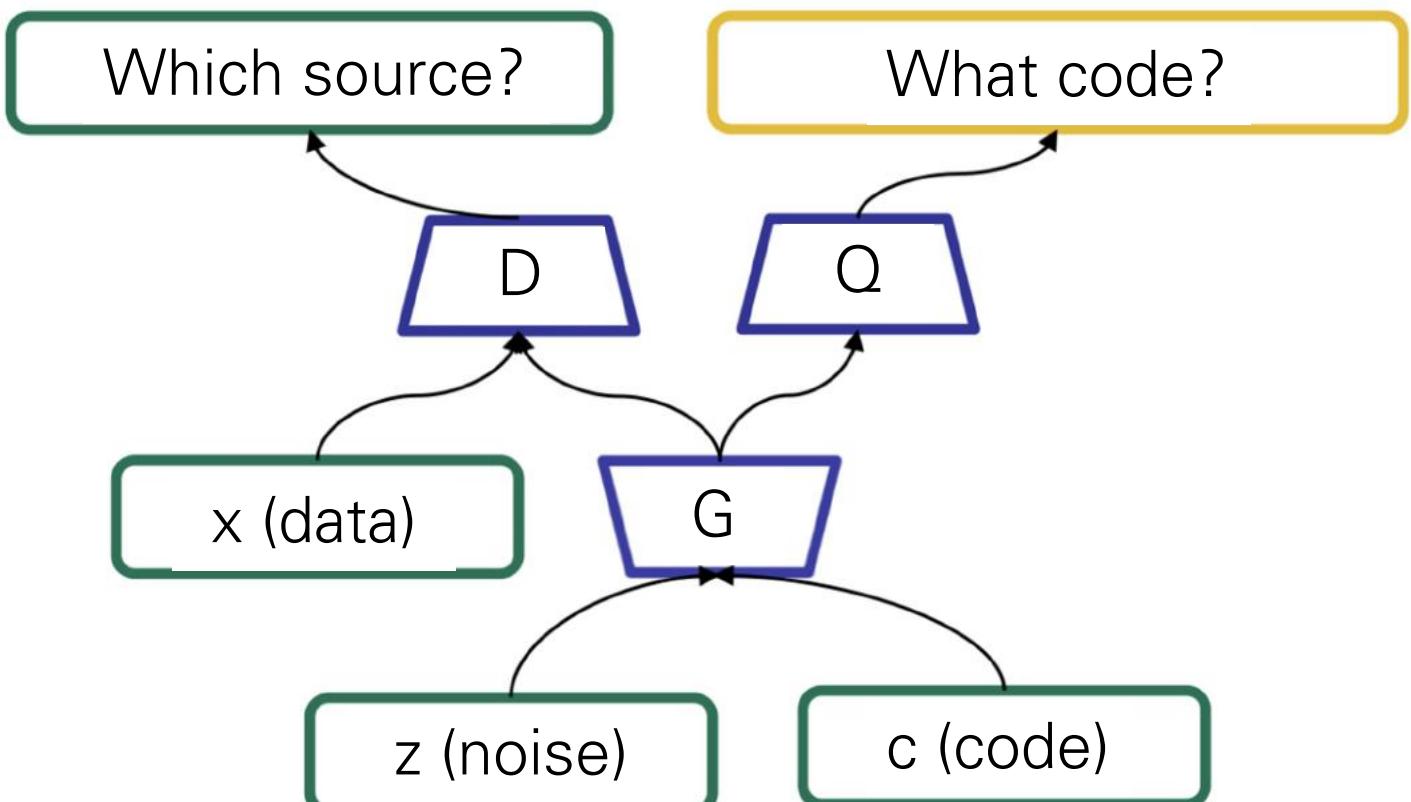


Simply MLE for a classifier/regressor

InfoGAN

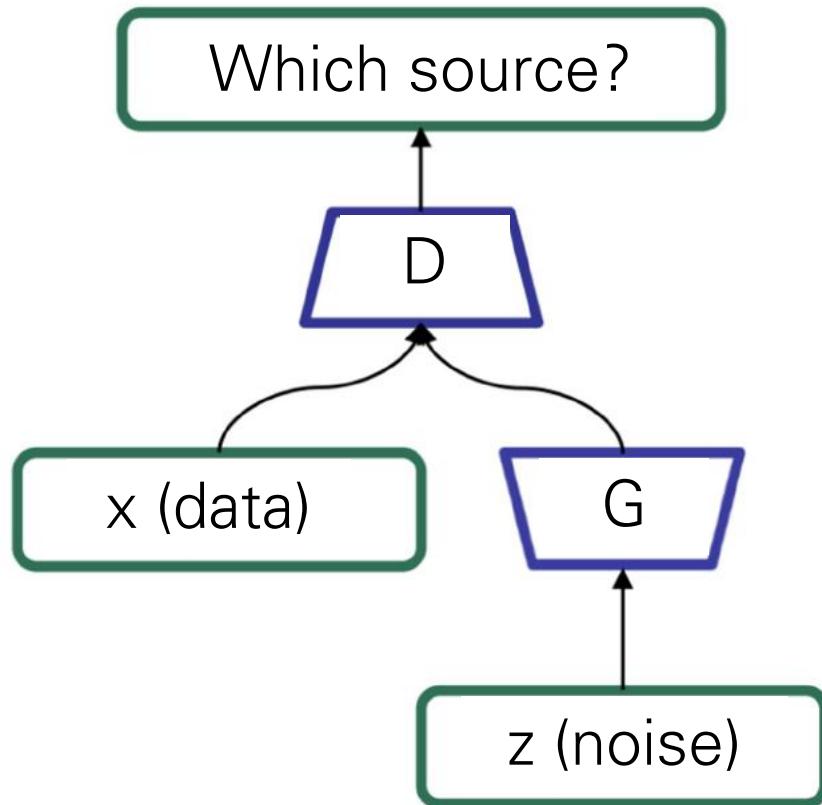


GAN

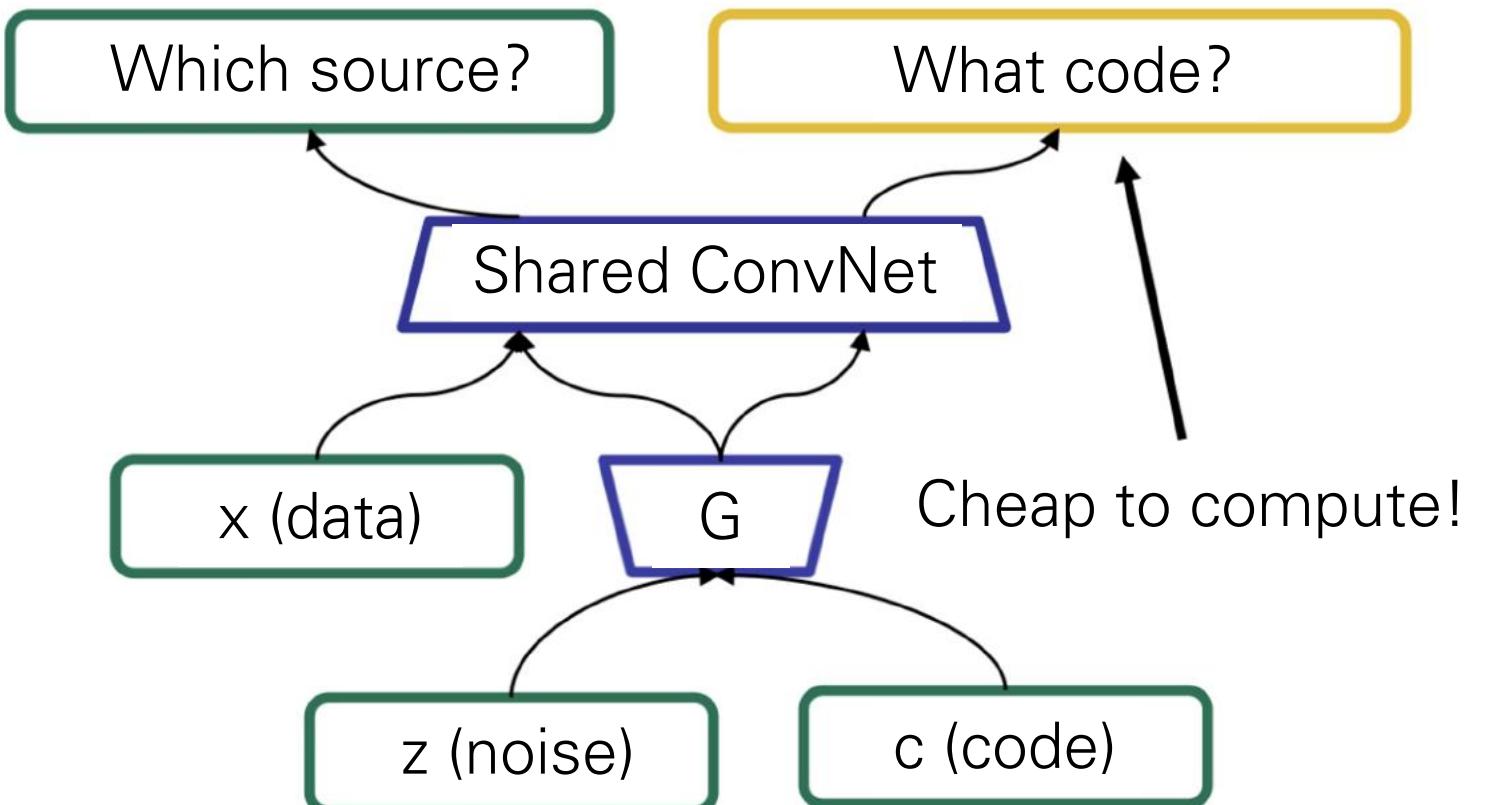


InfoGAN

InfoGAN

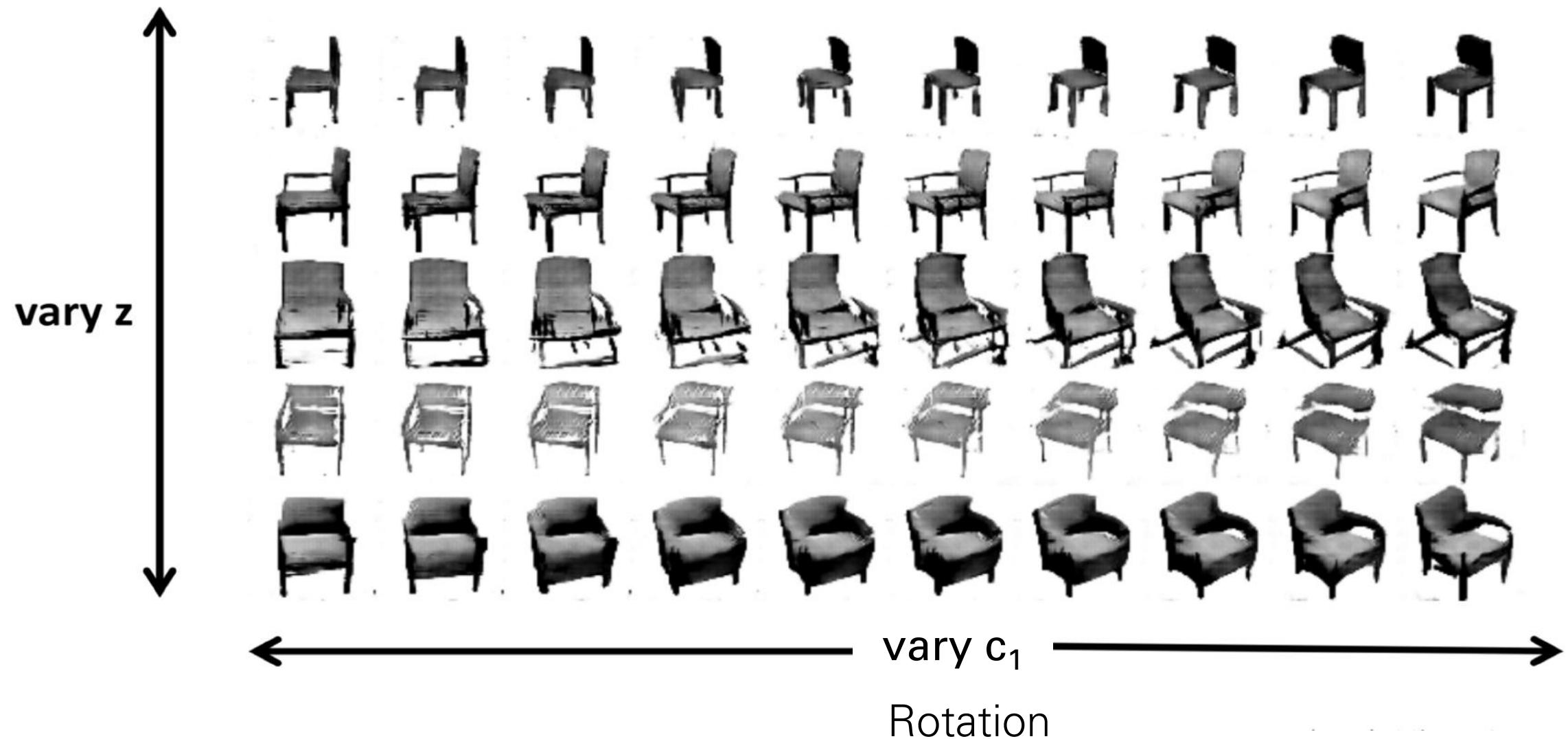


GAN

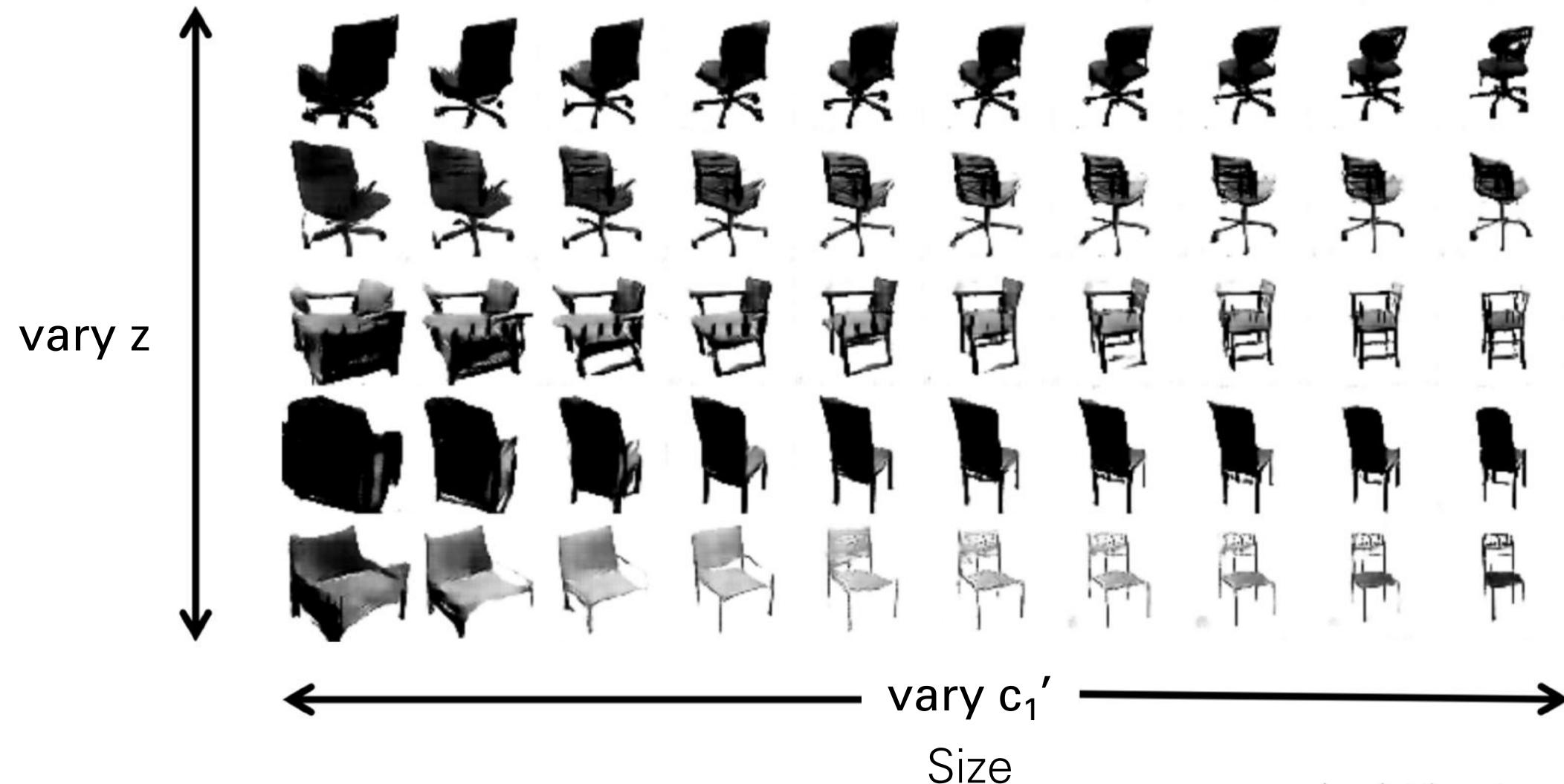


InfoGAN

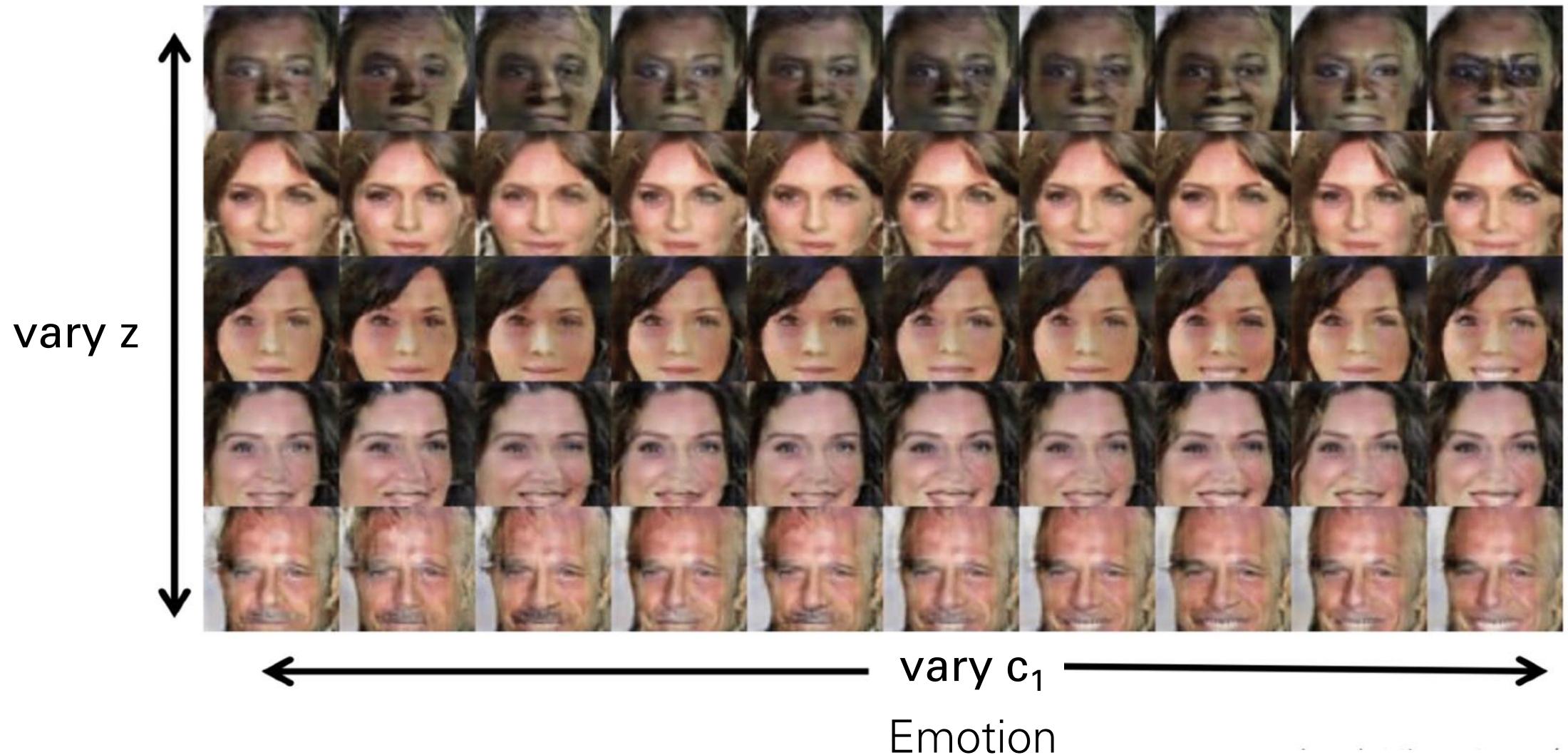
InfoGAN



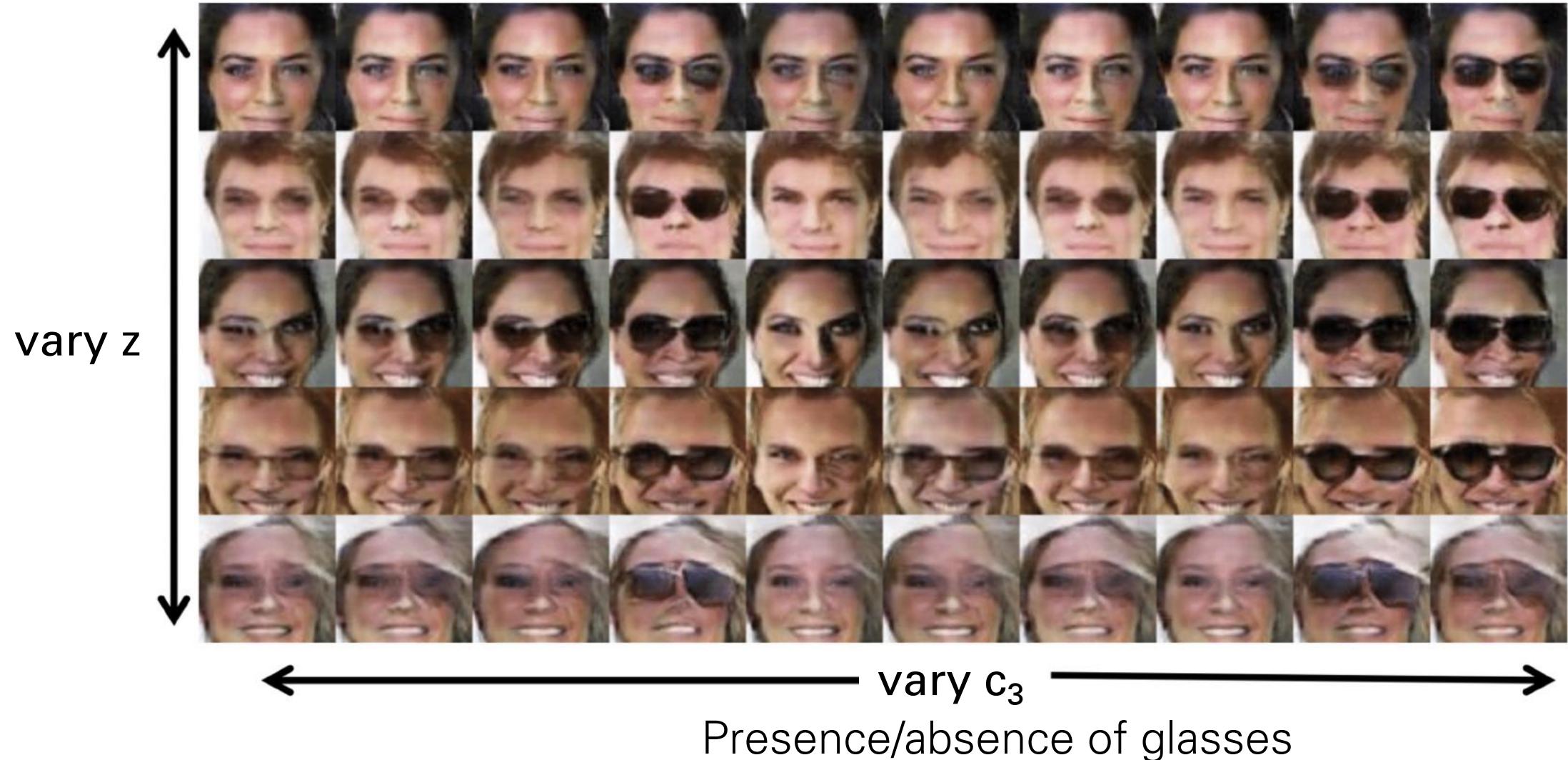
InfoGAN



InfoGAN

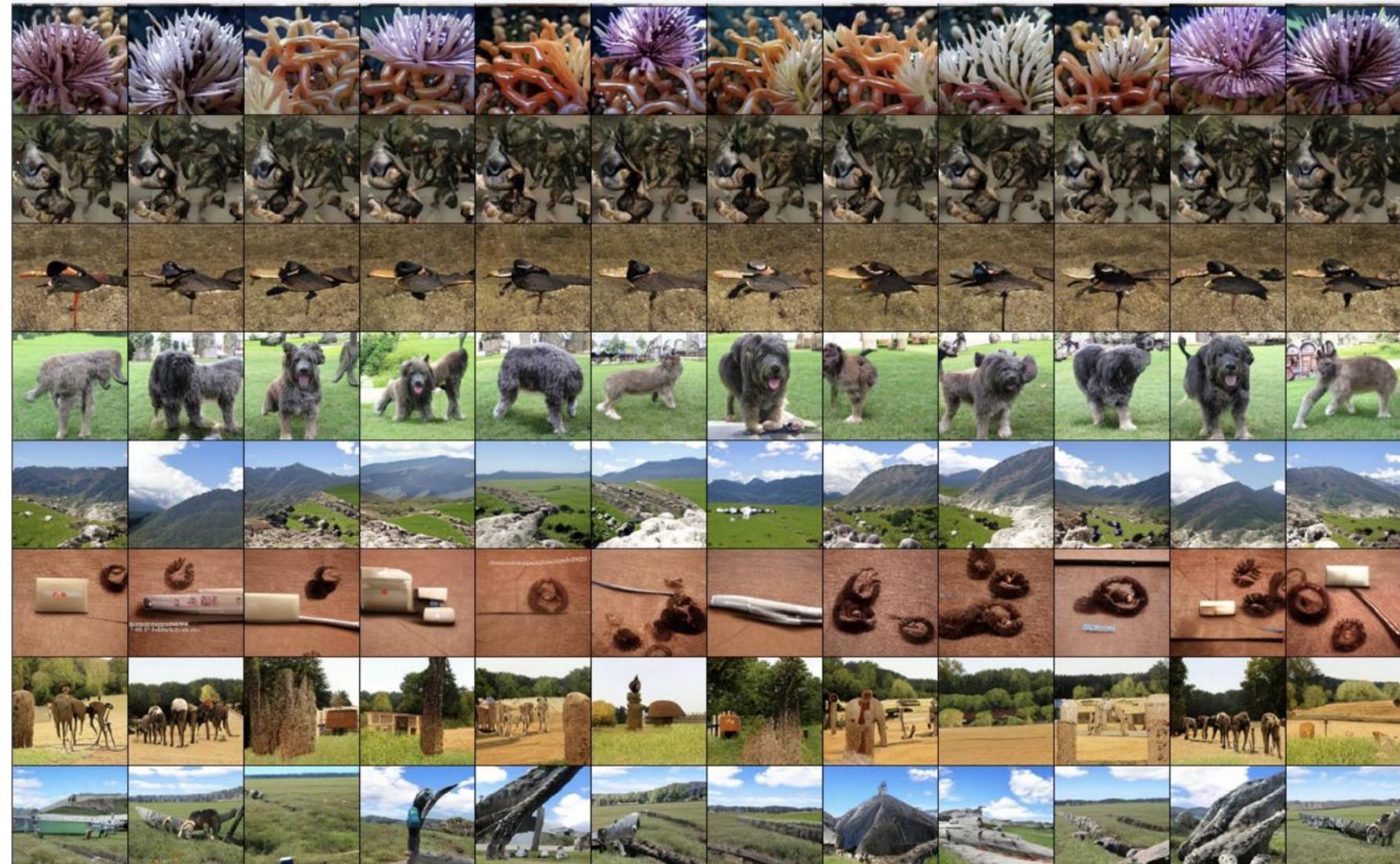


InfoGAN



Unsupervised Category Discovery - BigGAN

- Trained with no labels!
- $\mathbf{z} = \text{concat}([$
 - (a) $[\mathcal{N}(0, I)]^{120}$,
 - (b) $\text{UniformCateg}(1024)$ $])$
- Each row is one value of the categorical (b); columns are Gaussian samples (a)



Unsupervised Category Discovery - BigGAN

- Trained with no labels!
- $\mathbf{z} = \text{concat}([$
 - (a) $[\mathcal{N}(0, I)]^{120}$,
 - (b) $\text{UniformCateg}(1024)$ $])$
- Each row is one value of the categorical (b); columns are Gaussian samples (a)



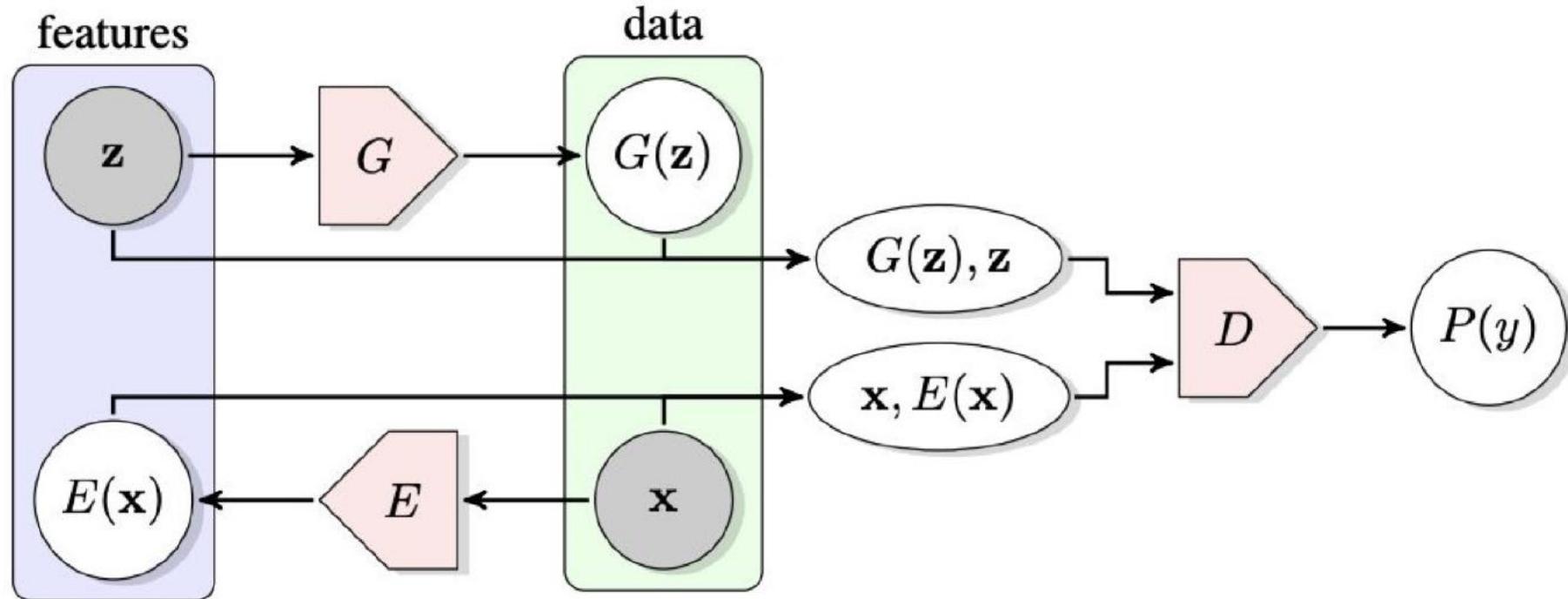
But what about inference...

- How can we use generative models?
 - GANs can generate content, but sometimes you want to make inference about observed data.
- Can we incorporate an inference mechanism into GANs?
- Can we learn an inference mechanism using an adversarial training paradigm?

Two papers, one model

- **ALI:** Vincent Dumoulin, Ishmael Belghazi, Olivier Mastropietro
ADVERSARIALLY LEARNED INFERENCE, ICLR 2017
Ben Poole, Alex Lamb, Martin Arjovsky
- **BiGAN:** Donahue, Krähenbühl and Darrell (2016), ADVERSARIAL
FEATURE LEARNING, ICLR 2017

Adversarially Learned Inference (ALI)



- Idea: Cast the learning of both an inference model (encoder) and a generative model (decoder) in a GAN-like adversarial framework
- Discriminator is trained to discriminate between joint samples (x, z) from:
 - Encoder distribution $q(x, z) = q(x) q(z | x)$, or
 - Decoder distribution $p(x, z) = p(z) p(x | z)$.
- Generator learns conditionals $q(z | x)$ and $p(x | z)$ to fool the discriminator.

x



$G(E(x))$



Adversarially Learned Inference (ALI)

- In the global optimum, E and G are inverses; for all x and z we have

$$\begin{aligned}-x &= G(E(x)) \\ -z &= E(G(z))\end{aligned}$$

- In practice, this inversion property does not hold perfectly

- Reconstructions still often capture interesting semantics

x



$G(E(x))$

x



$G(E(x))$

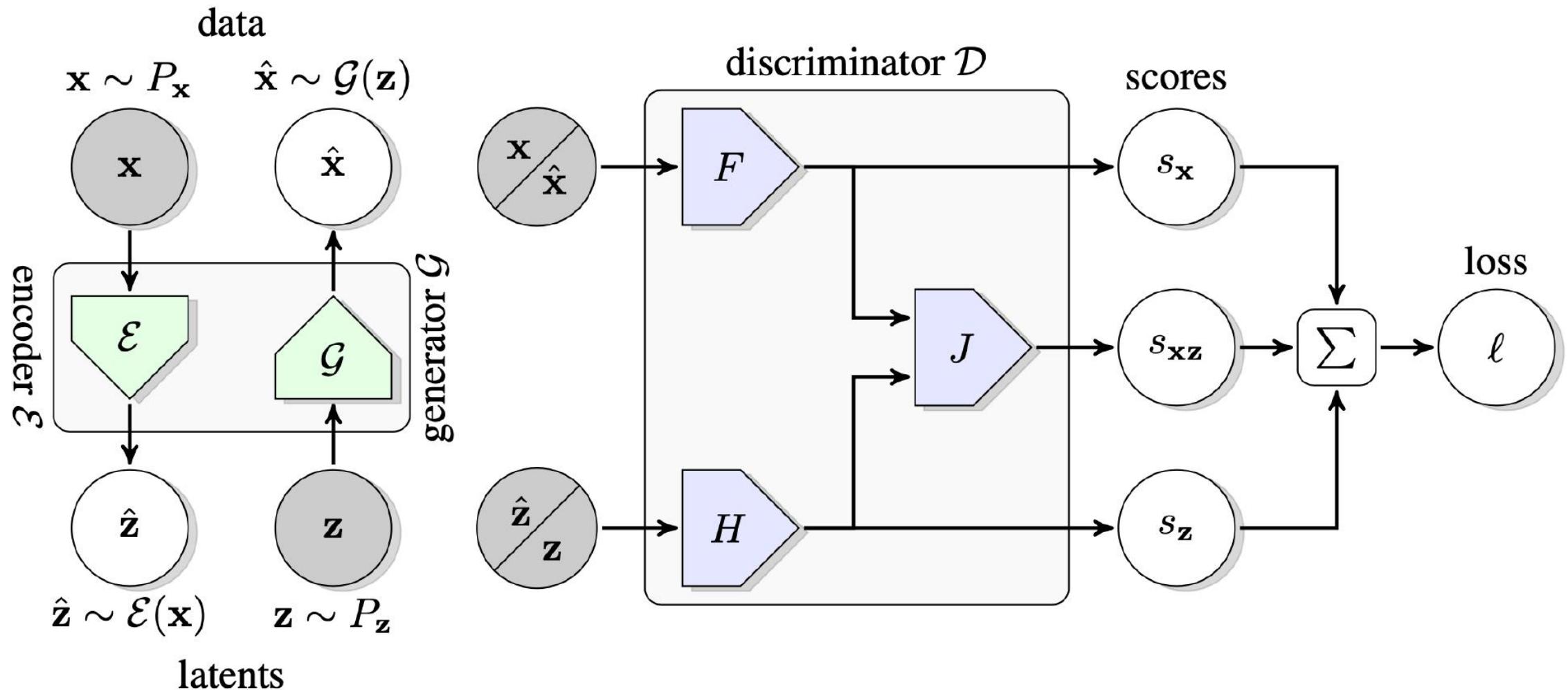
x



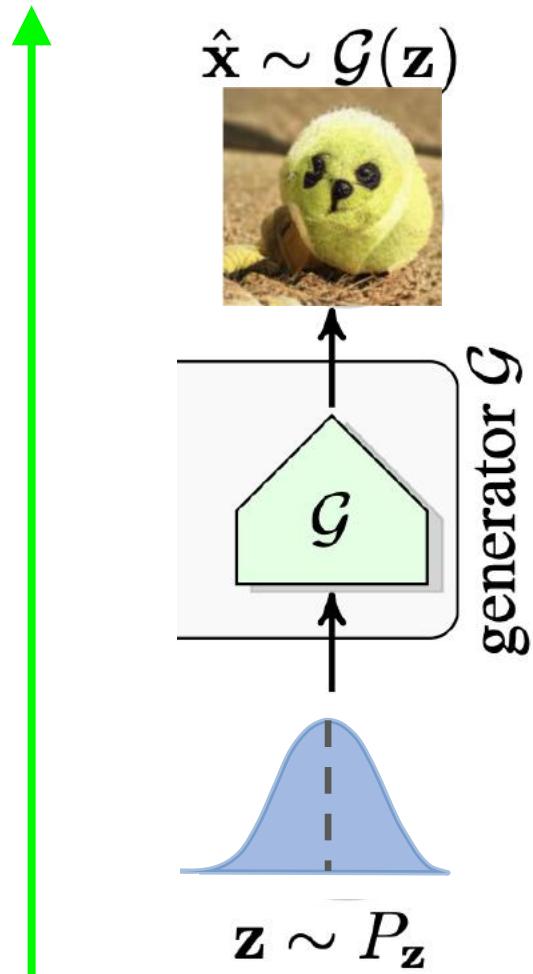
(a) CelebA samples.

(b) CelebA reconstructions.

Big Bidirectional GAN (BigBiGAN)



BigBiGAN

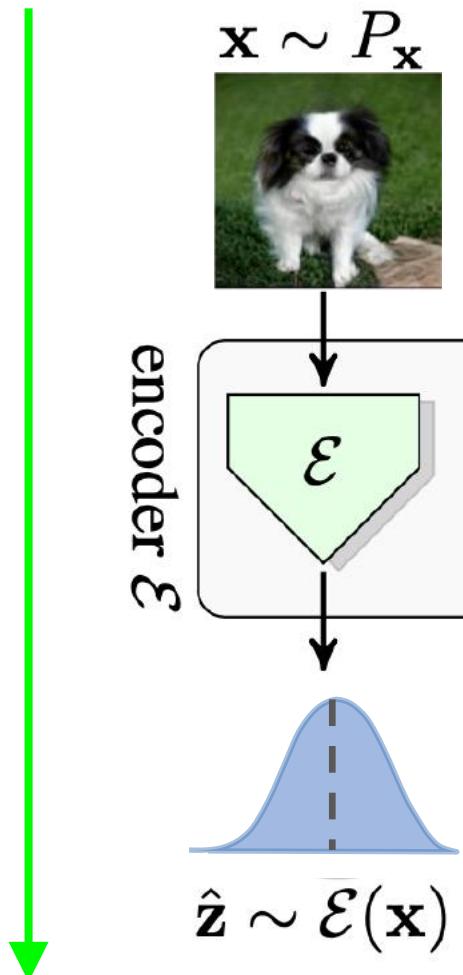


generated image

BigGAN generator convnet

latent sample

BigBiGAN



real image sample

image recognition model (ResNet)

predicted latent

BigBiGAN

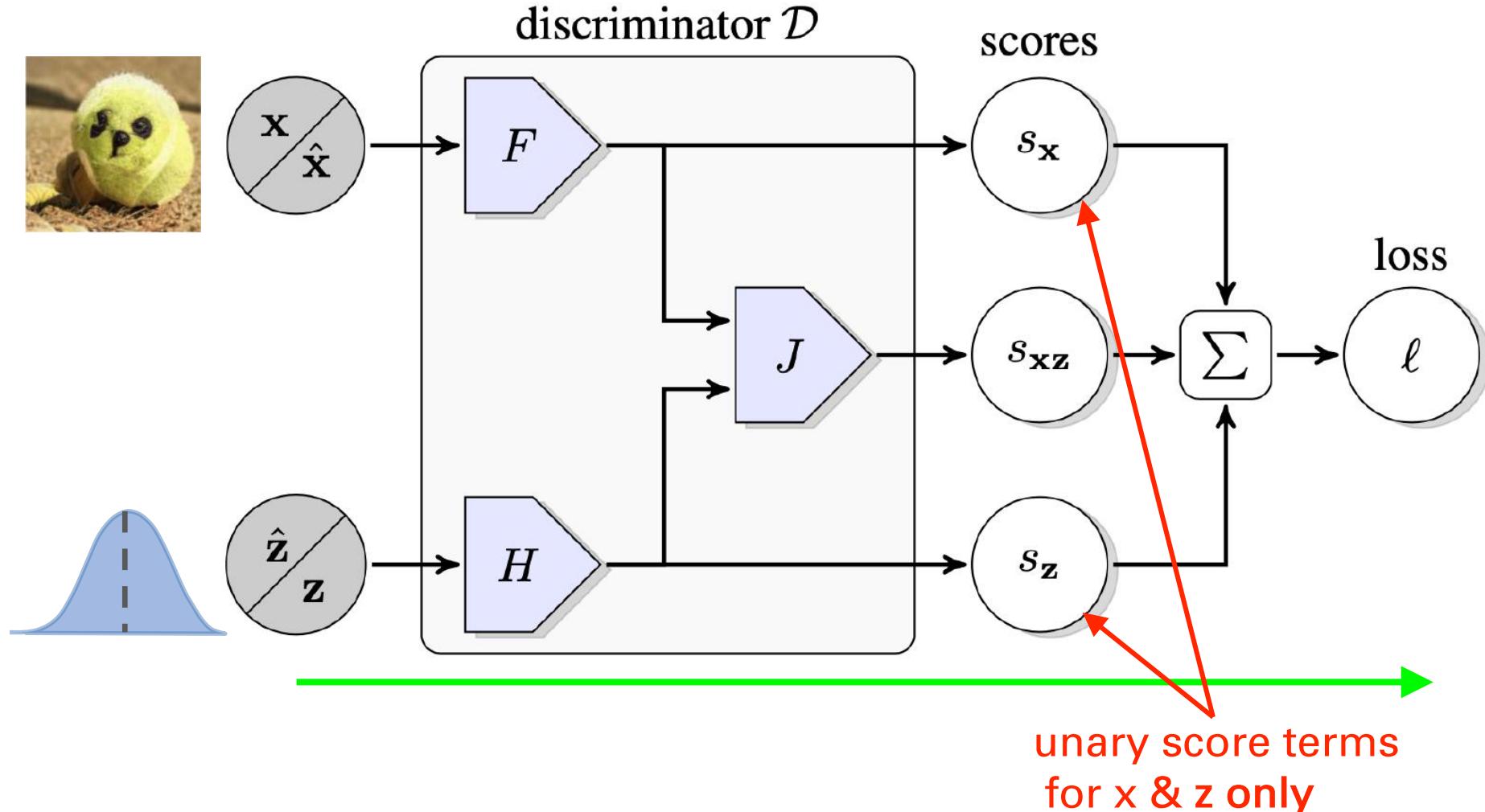
Discriminates
between input
pairs:

Encoder pair
 $(x, z' = E(x))$

vs.

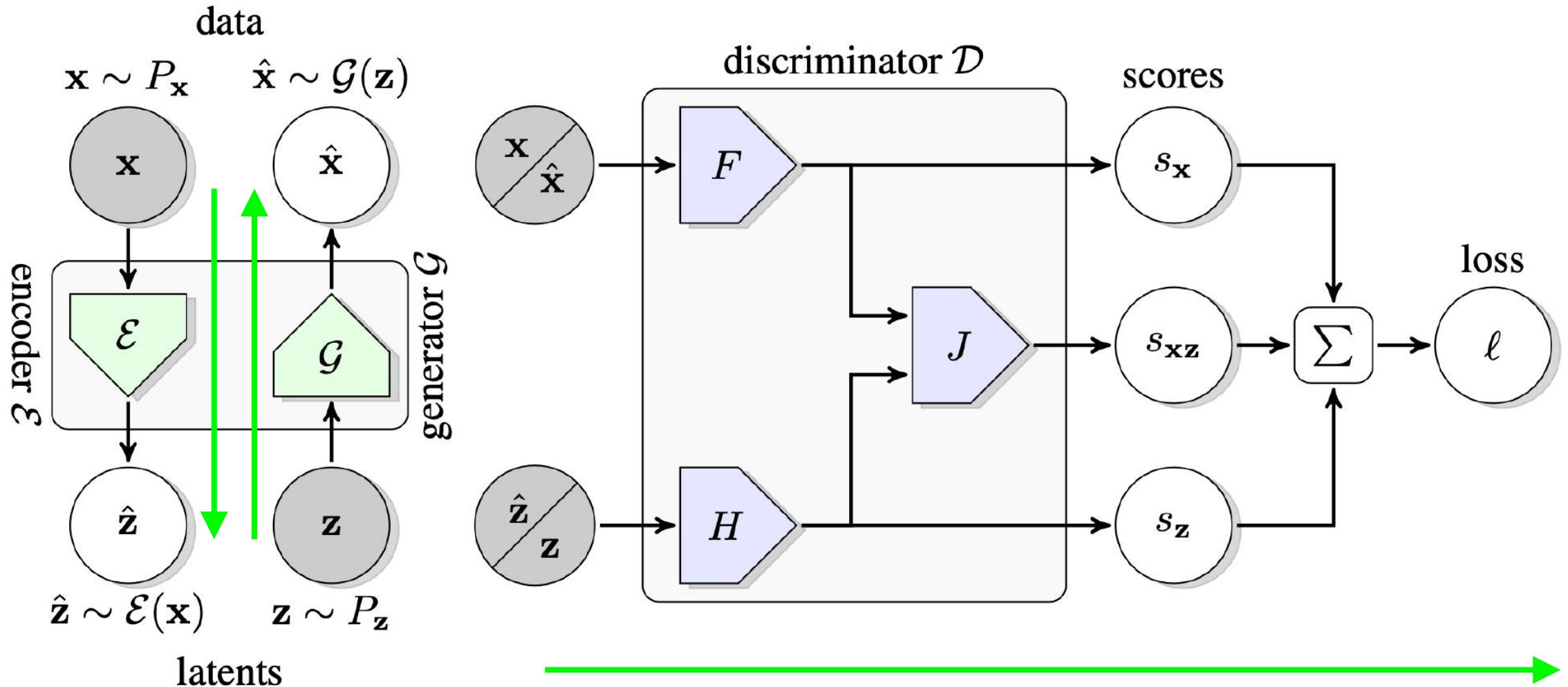
Generator pair
 $(x' = G(z), z)$

sees images x and latents z (not just images x)

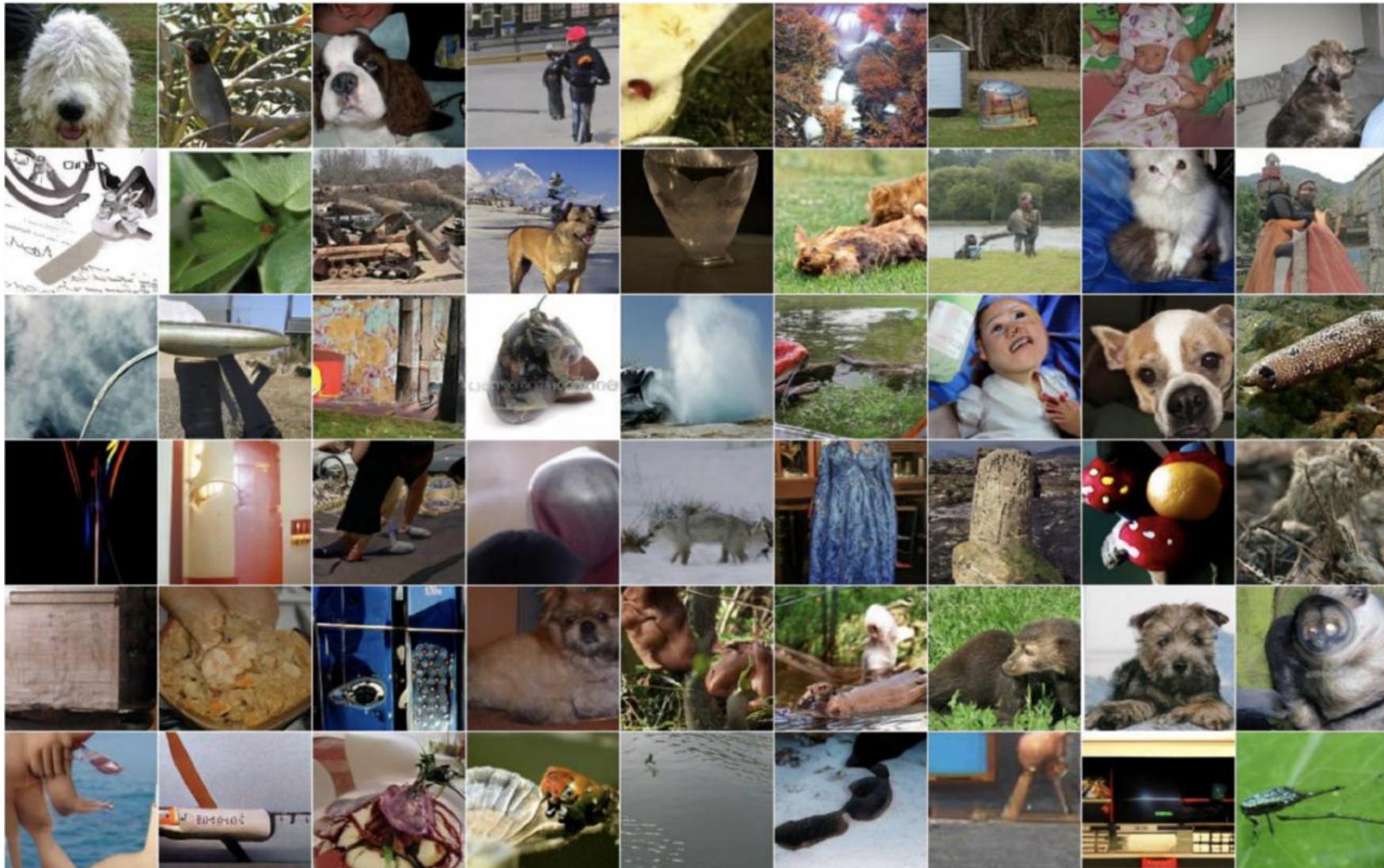


unary score terms
for x & z only

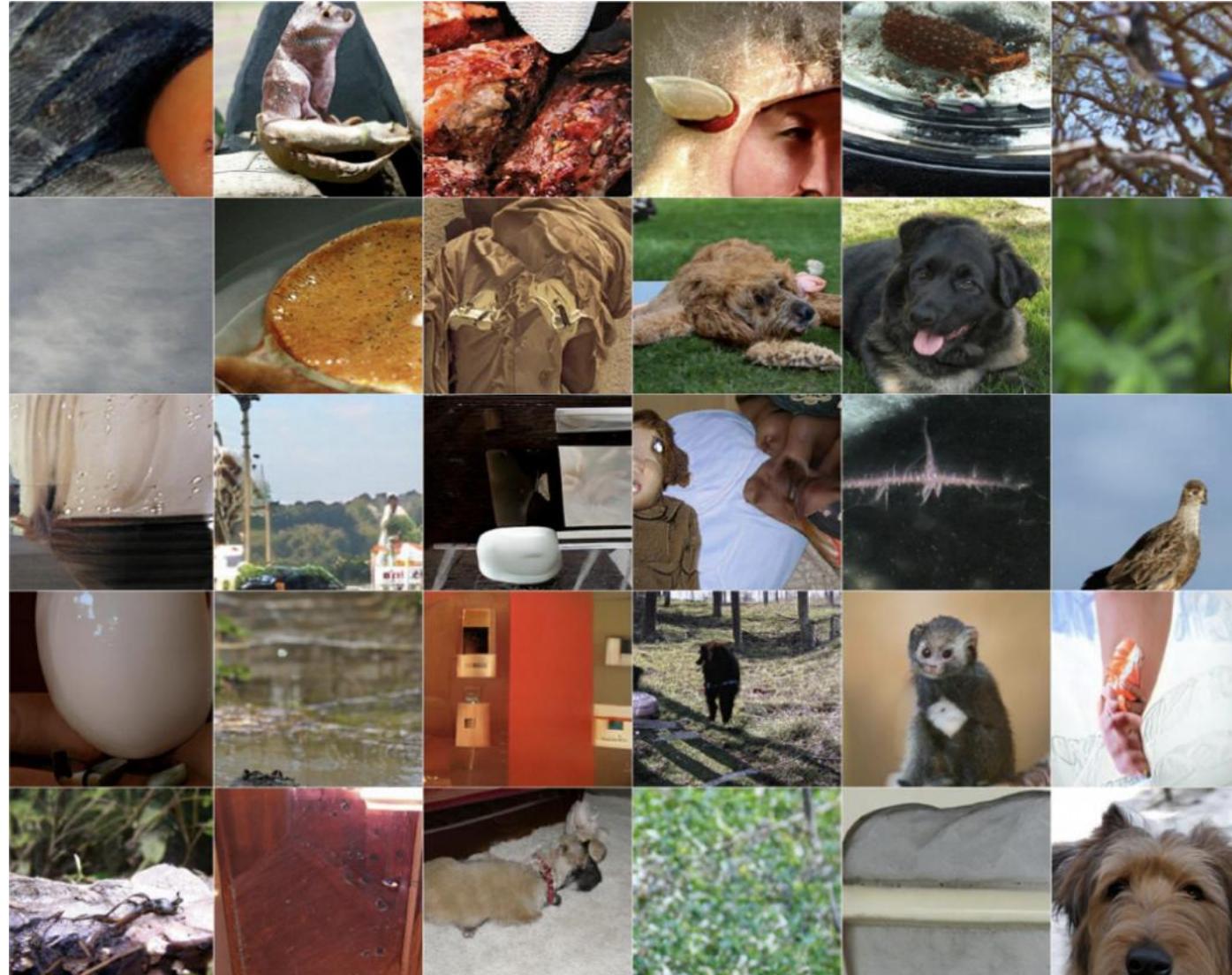
BigBiGAN



BigBiGAN: Unconditional Image Generation



BigBiGAN: Unconditional Image Generation



BigBiGAN: Representation Learning

Method	Architecture	Feature	Top-1	Top-5
BiGAN [7, 42]	AlexNet	Conv3	31.0	-
SS-GAN [4]	ResNet-19	Block6	38.3	-
Motion Segmentation (MS) [30, 6]	ResNet-101	AvePool	27.6	48.3
Exemplar (Ex) [8, 6]	ResNet-101	AvePool	31.5	53.1
Relative Position (RP) [5, 6]	ResNet-101	AvePool	36.2	59.2
Colorization (Col) [41, 6]	ResNet-101	AvePool	39.6	62.5
Combination of MS+Ex+RP+Col [6]	ResNet-101	AvePool	-	69.3
CPC [39]	ResNet-101	AvePool	48.7	73.6
Rotation [11, 24]	RevNet-50 \times 4	AvePool	55.4	-
Efficient CPC [17]	ResNet-170	AvePool	61.0	83.0
BigBiGAN (ours)	ResNet-50	AvePool	55.4	77.4
	ResNet-50	BN+CReLU	56.6	78.6
	RevNet-50 \times 4	AvePool	60.8	81.4
	RevNet-50 \times 4	BN+CReLU	61.3	81.9

BigBiGAN: Latent Space NNs

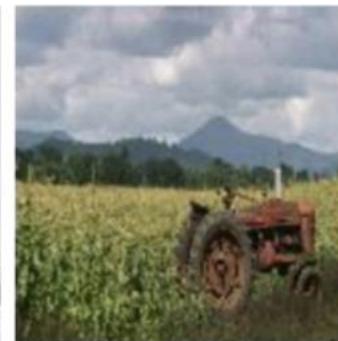
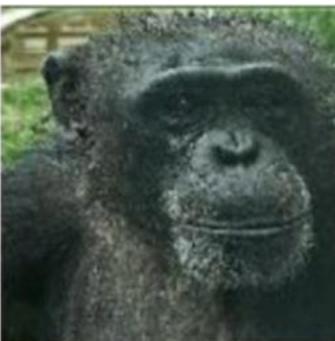


BigBiGAN Reconstructions

Computing a reconstruction $\mathbf{x}' = G(E(\mathbf{x}))$:

- (1) Sample a real image $\mathbf{x} \sim P_x$
- (2) Encoder predicts latents $\mathbf{z}' = E(\mathbf{x})$
- (3) Generator predicts reconstruction $\mathbf{x}' = G(\mathbf{z}')$

real images \mathbf{x}

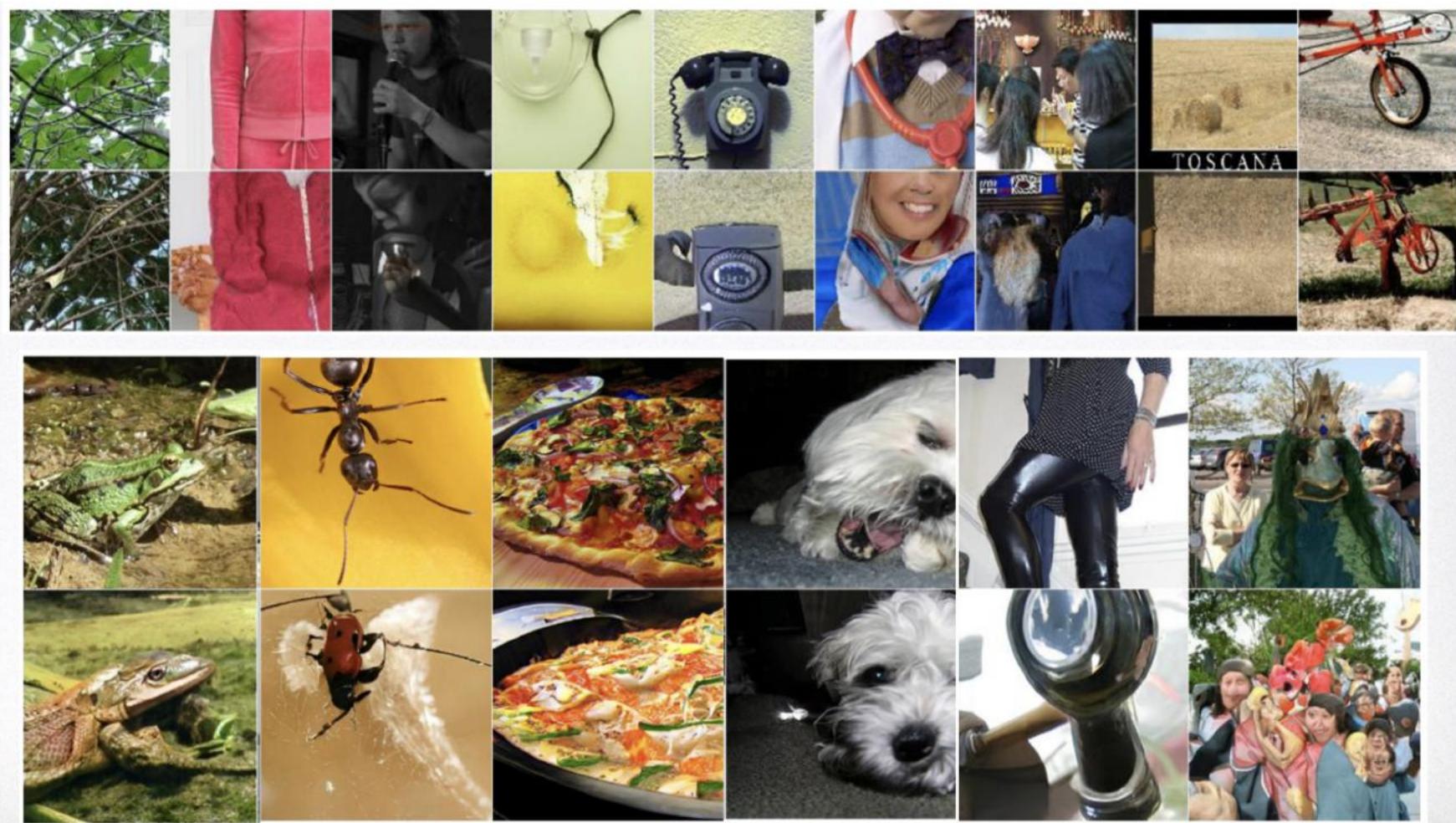


reconstructions $\mathbf{x}' = G(E(\mathbf{x}))$

(Big)BiGAN is not directly trained for reconstruction! =
Arises out of the objective: approx. reconstruction $\mathbf{x}' \cong G(E(\mathbf{x}))$
Optimally confuses the joint data-latent discriminator.

Reconstructions give insight into the semantics modeled.

BigBiGAN Reconstructions



Lecture overview

- Motivation and Definition of Implicit Models
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Frechet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- GANs and Representations
- Applications

Semi-supervised Classification

(Salimans et al., 2016;
Dumoulin et al., 2016)

SVNH

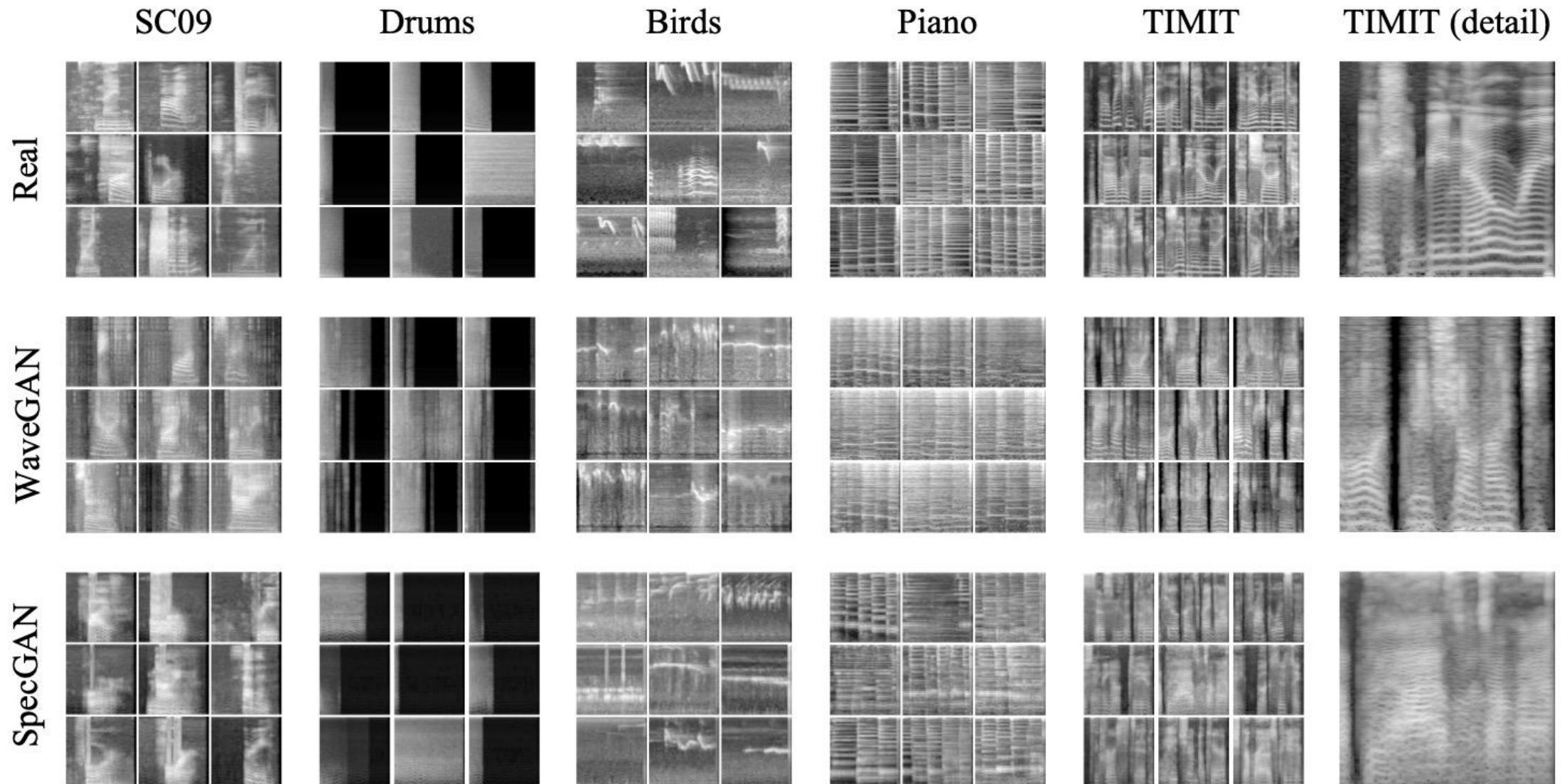
Model	Misclassification rate
VAE (M1 + M2) (Kingma et al., 2014)	36.02
SWWAE with dropout (Zhao et al., 2015)	23.56
DCGAN + L2-SVM (Radford et al., 2015)	22.18
SDGM (Maaløe et al., 2016)	16.61
GAN (feature matching) (Salimans et al., 2016)	8.11 ± 1.3
ALI (ours, L2-SVM)	19.14 ± 0.50
ALI (ours, no feature matching)	7.42 ± 0.65

Text Generation: MaskGAN (Fedus et al. 2018)

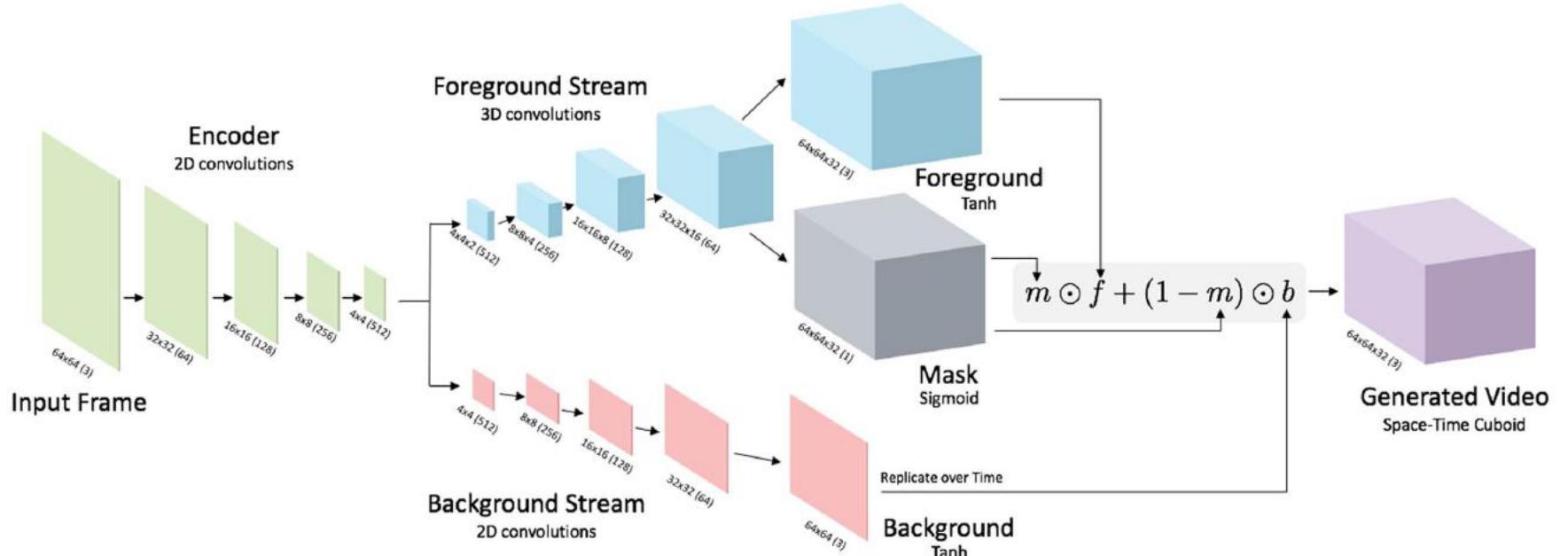
Ground Truth	Pitch Black was a complete shock to me when I first saw it back in 2000 In the previous years I
MaskGAN	Pitch Black was a complete shock to me when I first saw it back in <u>1979</u> I was really looking forward
MaskMLE	Black was a complete shock to me when I first saw it back in <u>1969</u> I live in New Zealand

Table 3: Conditional samples from IMDB for both MaskGAN and MaskMLE models.

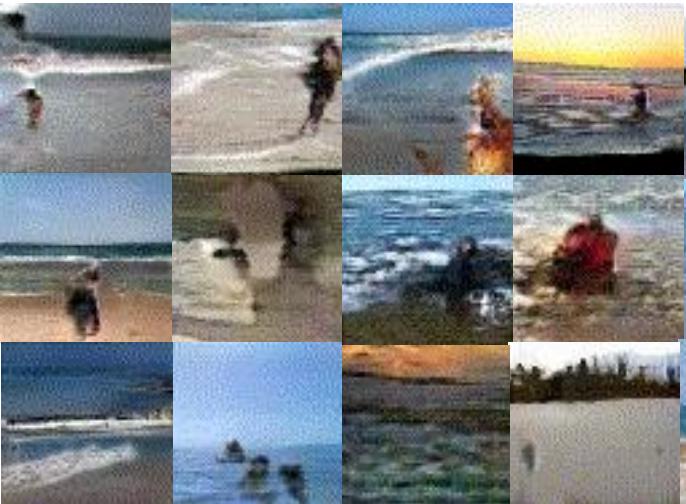
Audio Synthesis: WaveGAN (Donahue et al. 2020)



Video Generation (Vondrick et al., 2016)



Beach



Golf

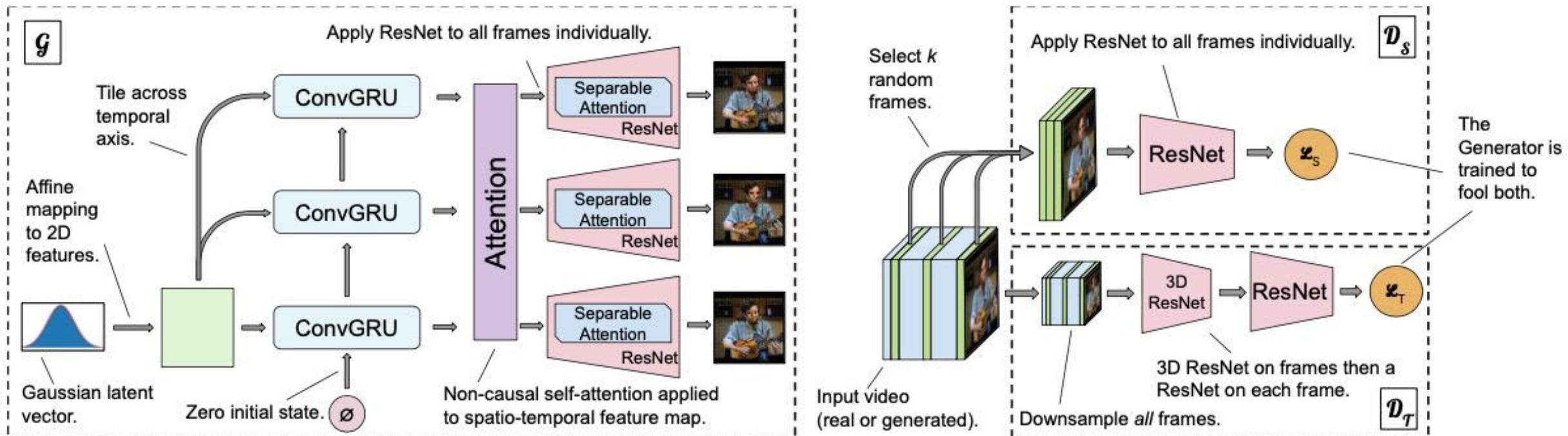


Train Station



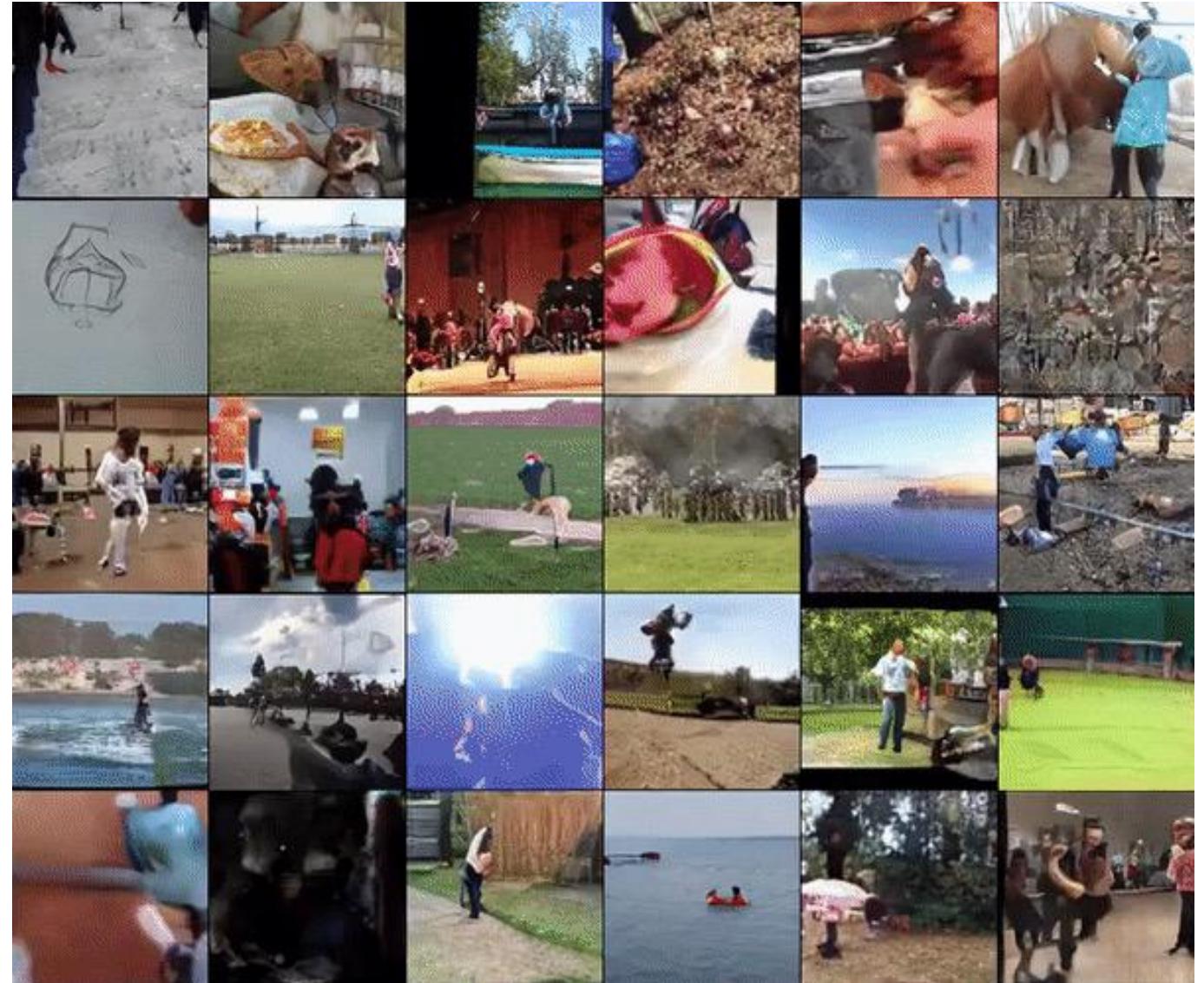
DVD-GAN: Efficient Video Generation

(Clark et al., 2019)



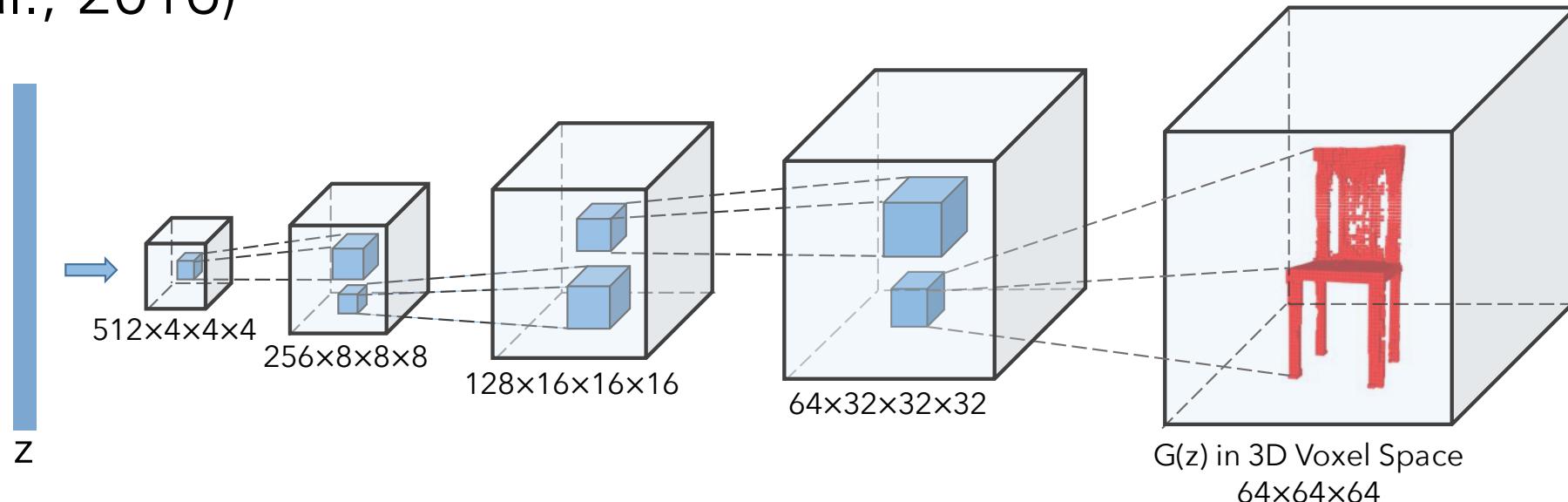
DVD-GAN: Efficient Video Generation

(Clark et al., 2019)

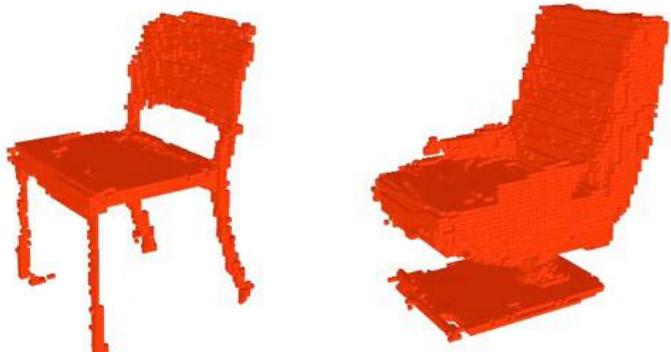


3DGAN: Generative Shape Modeling

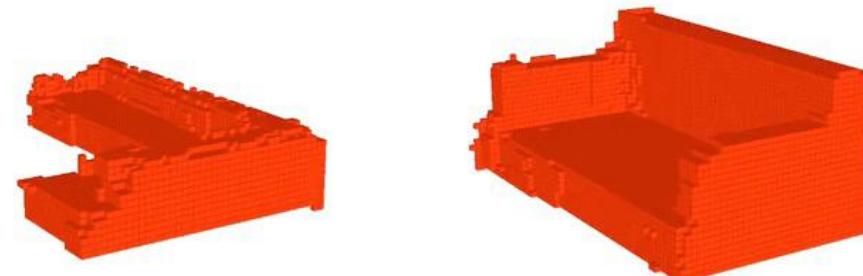
(Wu et al., 2016)



Chairs

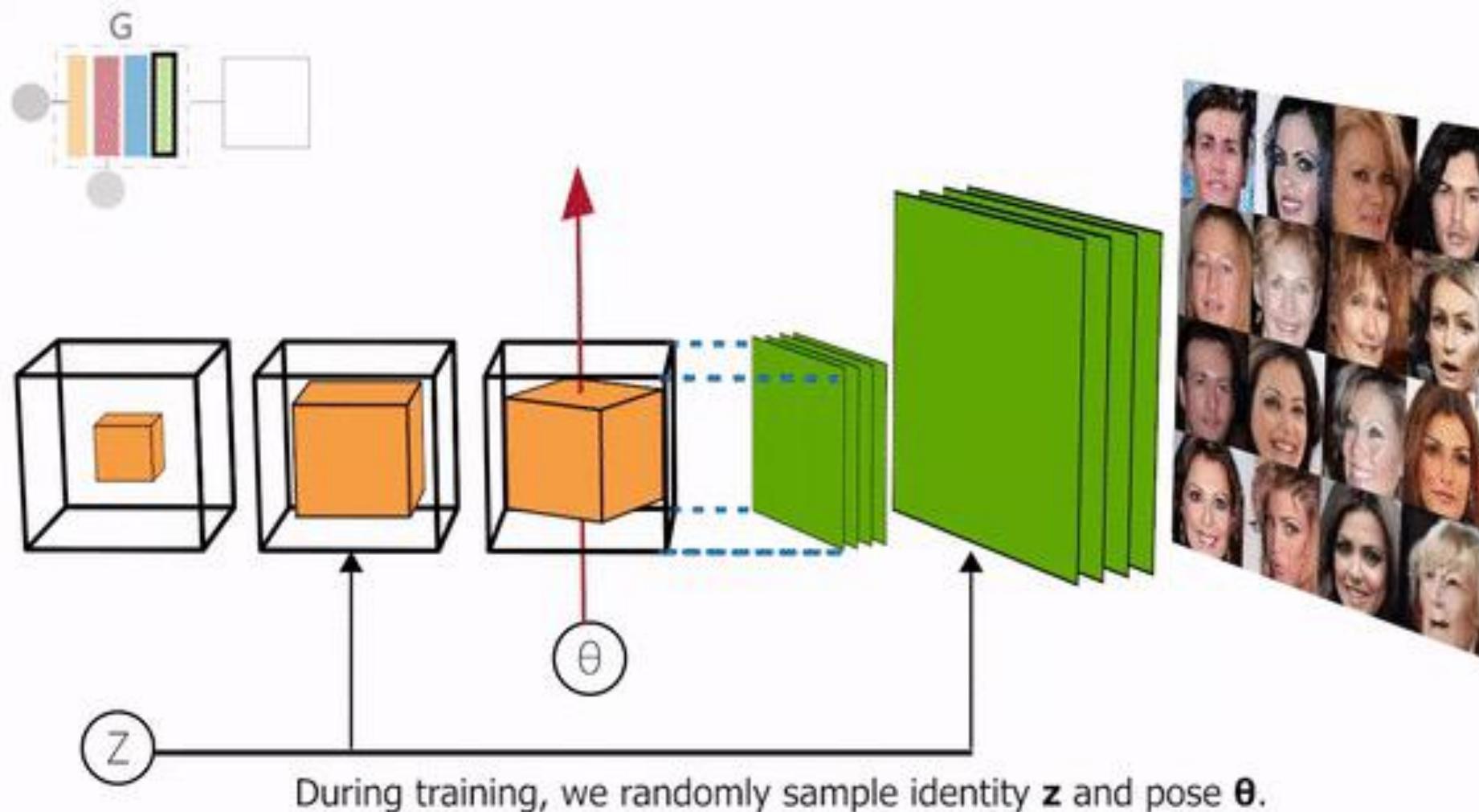


Sofas



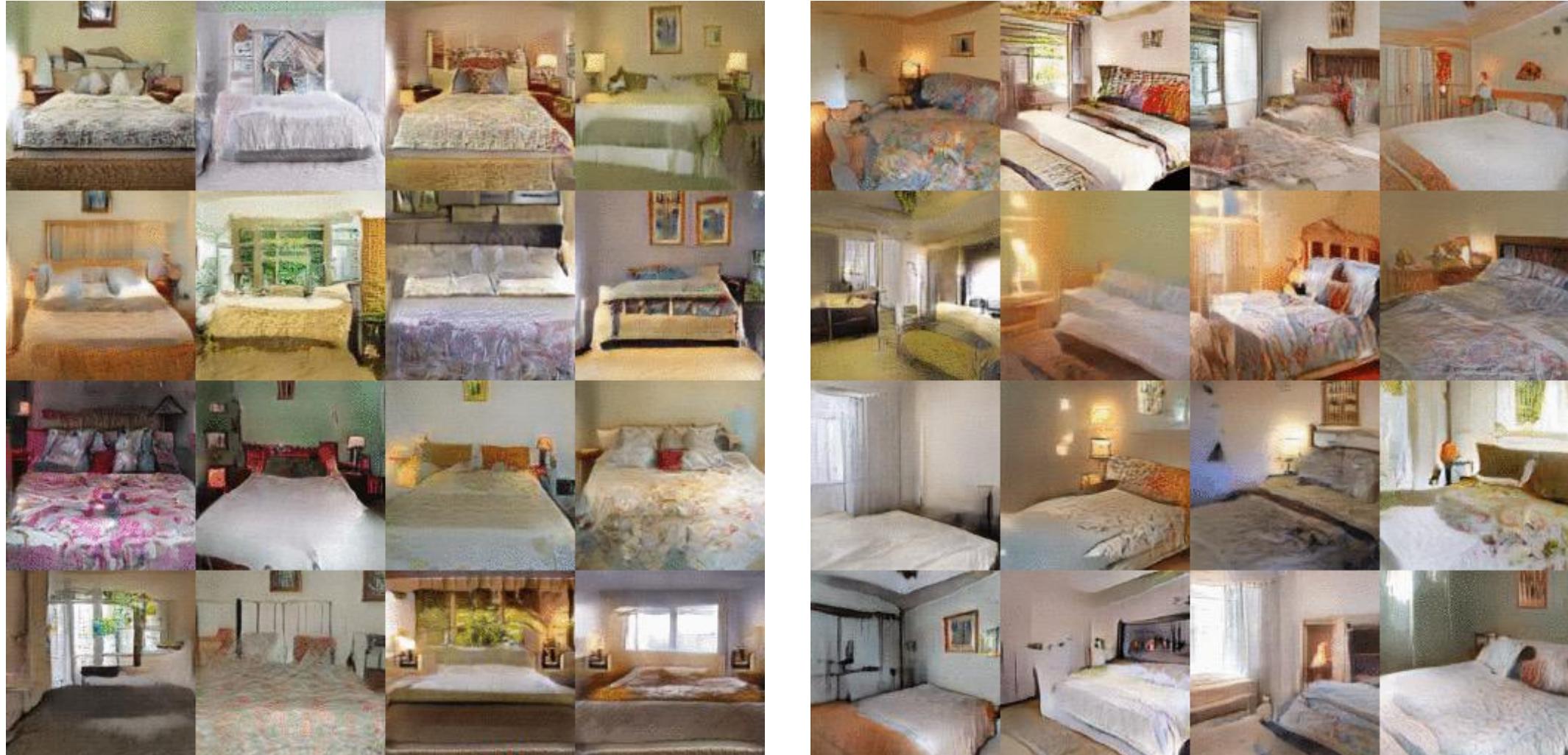
HoloGAN: Learning 3D Representations from Images

(Nguyen-Phuoc et al., 2020)



HoloGAN: Learning 3D Representations from Images

(Nguyen-Phuoc et al., 2020)



Motion Transfer: Everybody Dance Now



Vid2Vid: Video to Video Synthesis



StackGAN: Text-to-Image Synthesis (Zhang et al.'16)

The small bird has a red head with feathers that fade from red to gray from head to tail



The petals of this flower are white with a large stigma

A unique yellow flower with no visible pistils protruding from the center

This flower is pink and yellow in color, with petals that are oddly shaped

This is a light colored flower with many different petals on a green stem

This flower is yellow and green in color, with petals that are ruffled

The flower have large petals that are pink with yellow on some of the petals

A flower that has white petals with some tones of yellow and green filaments



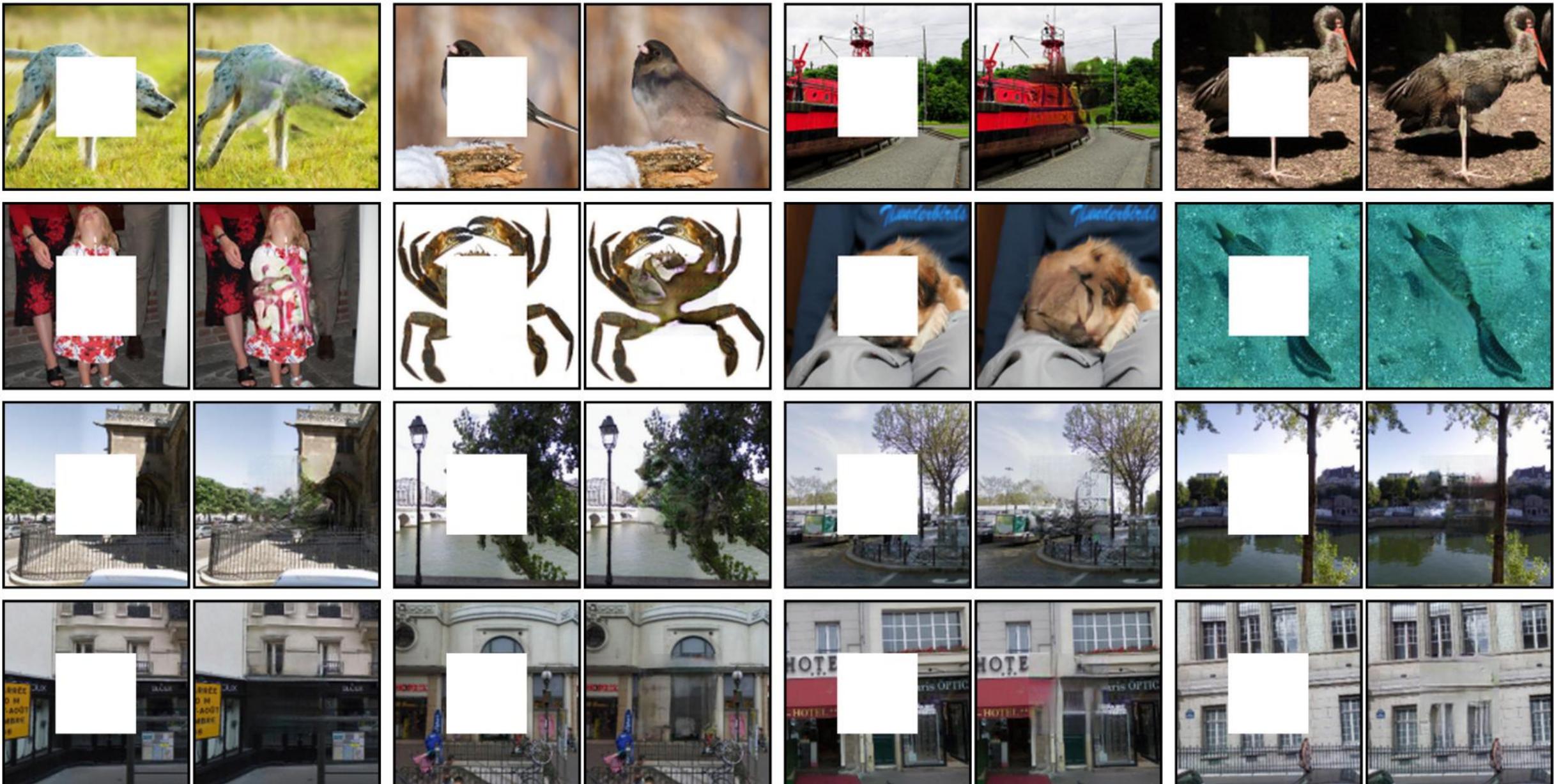
SRGAN: Single Image Super-Resolution

(Ledig et al., 2017)

- Combine content loss with adversarial loss



Image Inpainting (Pathak et al., 2016)



Unsupervised Domain Adaptation (Bousmalis et al., 2016)

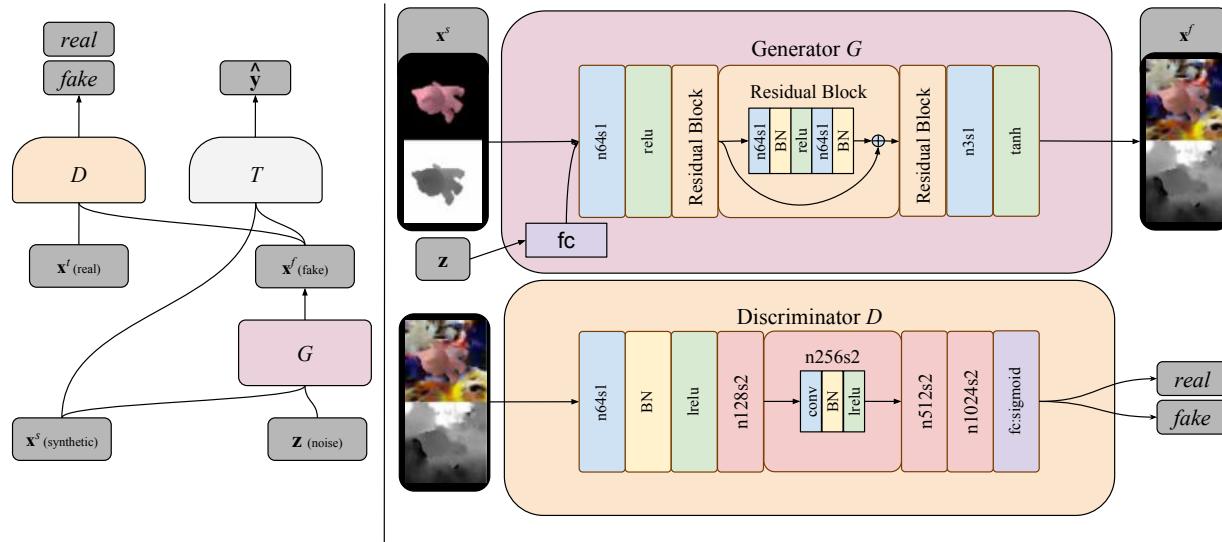
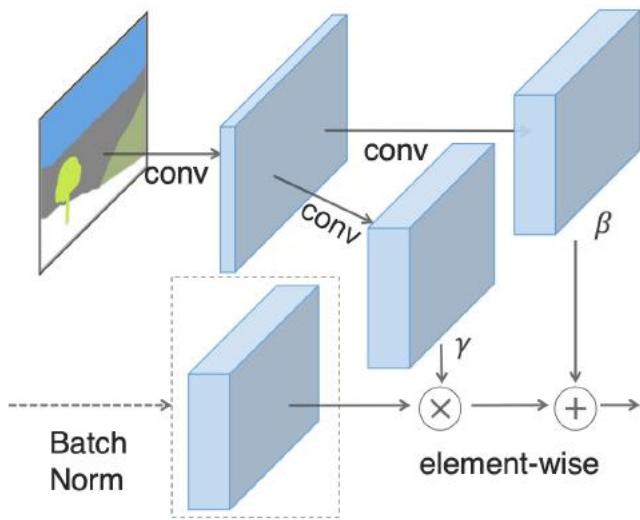


Image examples from the Linemod dataset



RGDB image samples
(conditioned on a synthetic image)

Semantic Image Editing: GauGAN

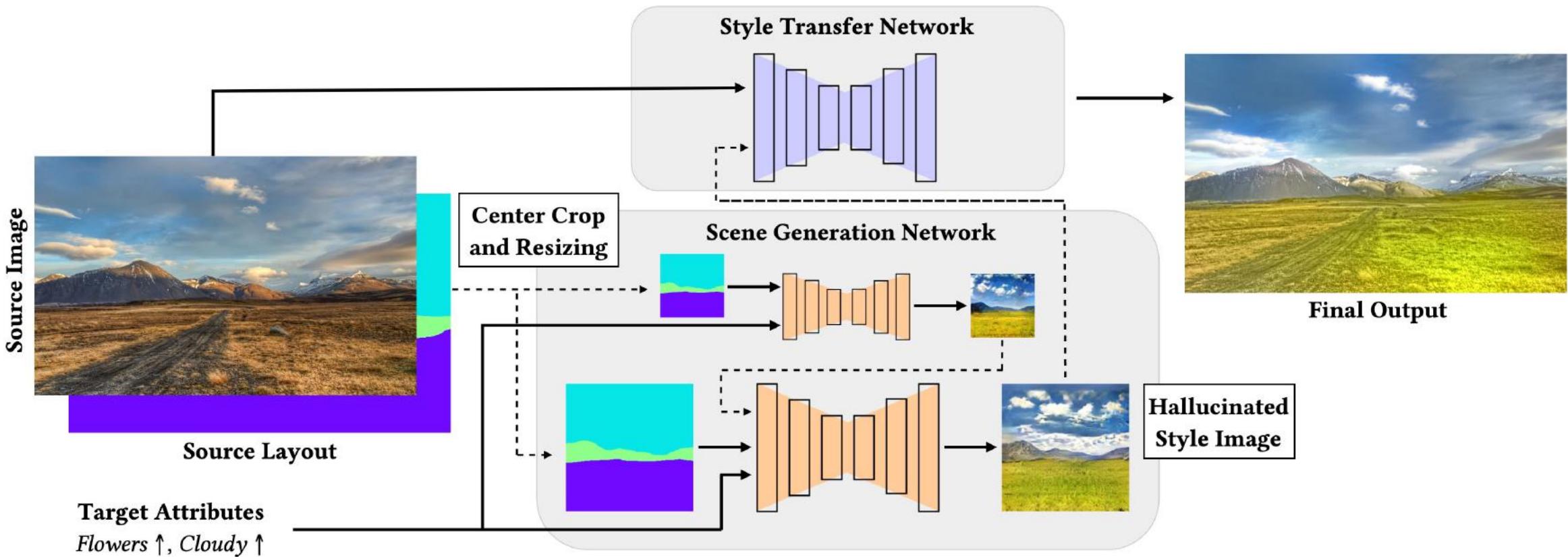


(Park et al. 2019)



Semantic Image Editing

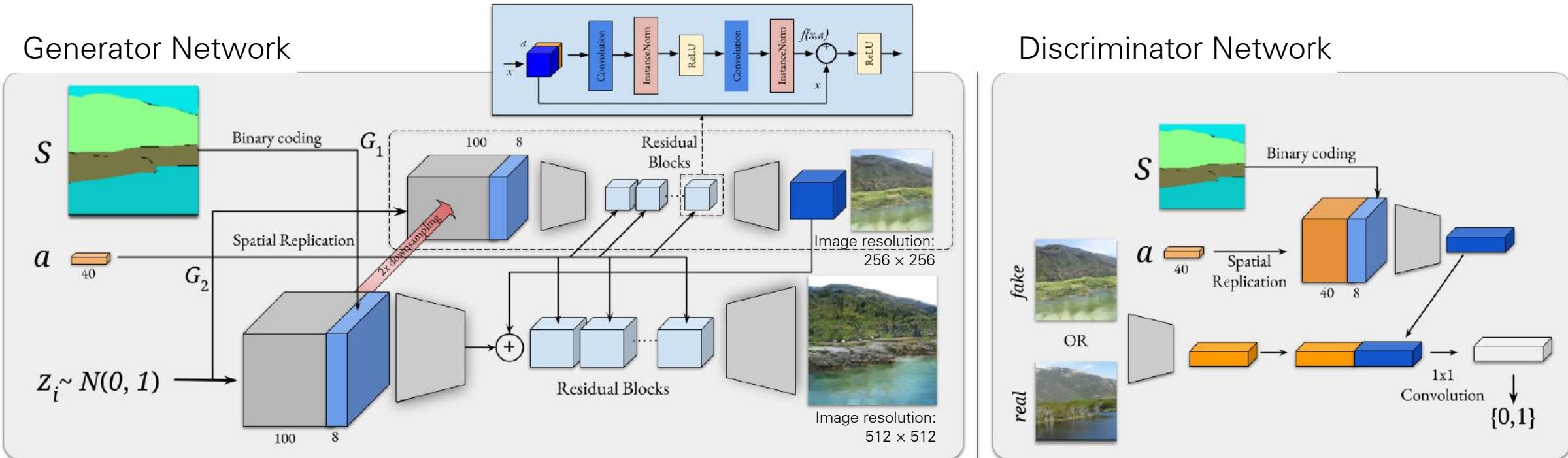
(Karacan et al. 2020)



https://hucvl.github.io/attribute_hallucination/

Scene Generation Network (SGN)

- The semantic layout categories are encoded into 8-bit binary codes
- The transient attributes are represented by a 40-d vector.



- An architecture similar to Pix2pixHD model (Wang et al. 2018)
- **Generator network:** A coarse-to-fine model with 2 generator networks
- **Discriminator network:** A combination of three different discriminator networks operating at an image pyramid of 3 scales

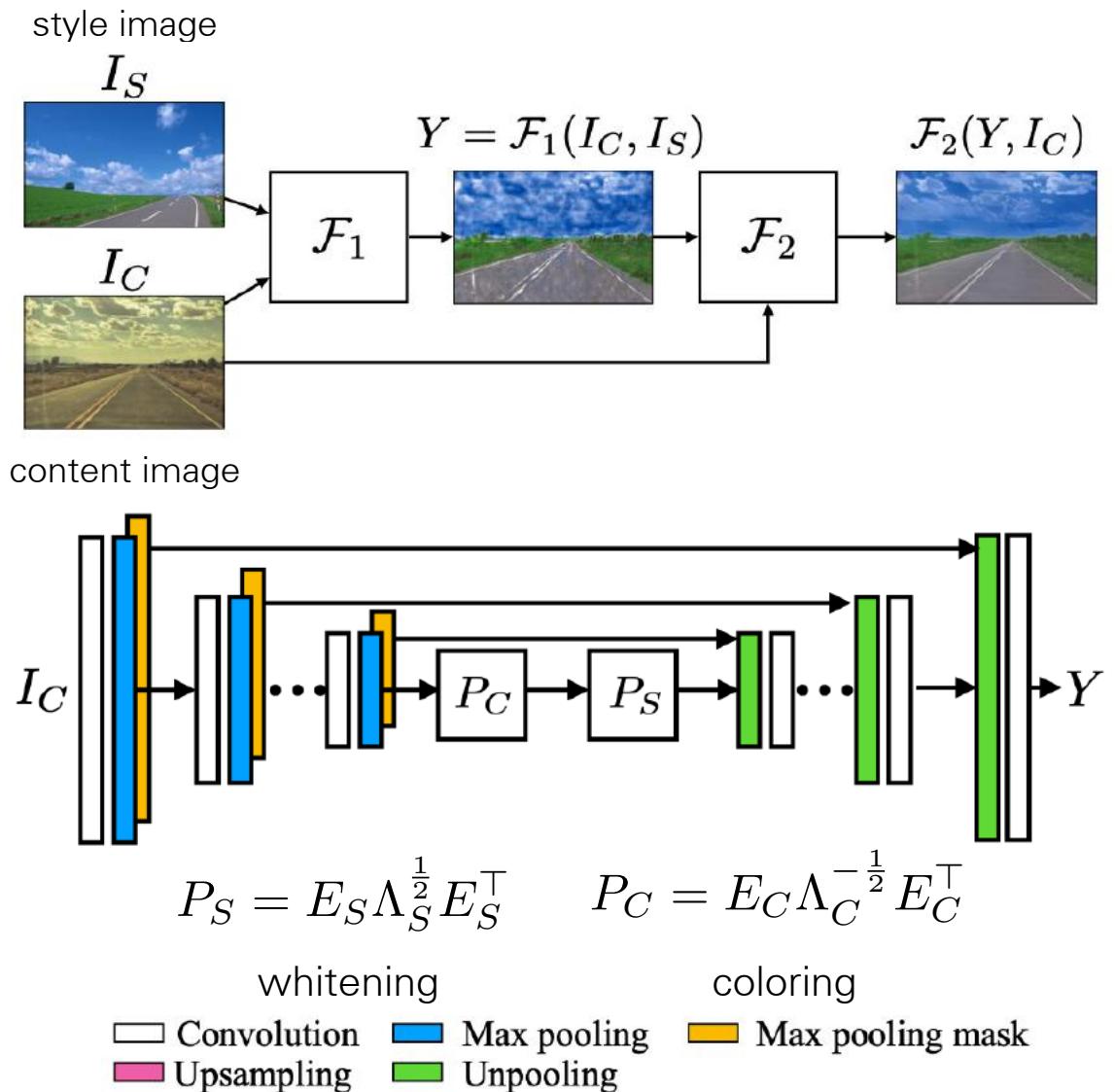
Training Objective of SGNs

$$\mathcal{L}_{SGN} = \min_G \left(\left(\max_{D=\{D_1, D_2, D_3\}} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k) \right) + \lambda \mathcal{L}_{percep}(G) \right)$$

- **Relative Negative Mining (RNM)**
 - real image, relevant attributes and layout
 - vs.
 - fake image, relevant attributes and layout
 - real image, mismatching layout (chosen from hard negatives)
 - or mismatching attributes
- **Layout-Invariant Perceptual Loss**
 - $\mathcal{L}_{percep}(G) = E_{z \sim p_z(z); x, S, a \sim p_{data}(S, a)} \left[\|f_P(x) - f_P(G(z, a, S))\|_2^2 \right]$
 - f_P : CNN encoder for the scene parser network (Zhou et al., 2018)

Style Transfer Network

- The FPST method of (Li et al., 2018), which is composed of two steps with close-form solutions:
 - Stylization step \mathcal{F}_1
 - Smoothing step \mathcal{F}_2
$$I_{out} = \mathcal{F}_2(\mathcal{F}_1(I_C, I_S), I_C)$$
- The **stylization step** is based on the whitening and coloring transform to stylize images via feature projections
 - Style information encoded by the covariance matrix of VGG features
- The **smoothing step** ensures spatially consistent stylizations via a manifold ranking operator.



ALS18K Dataset

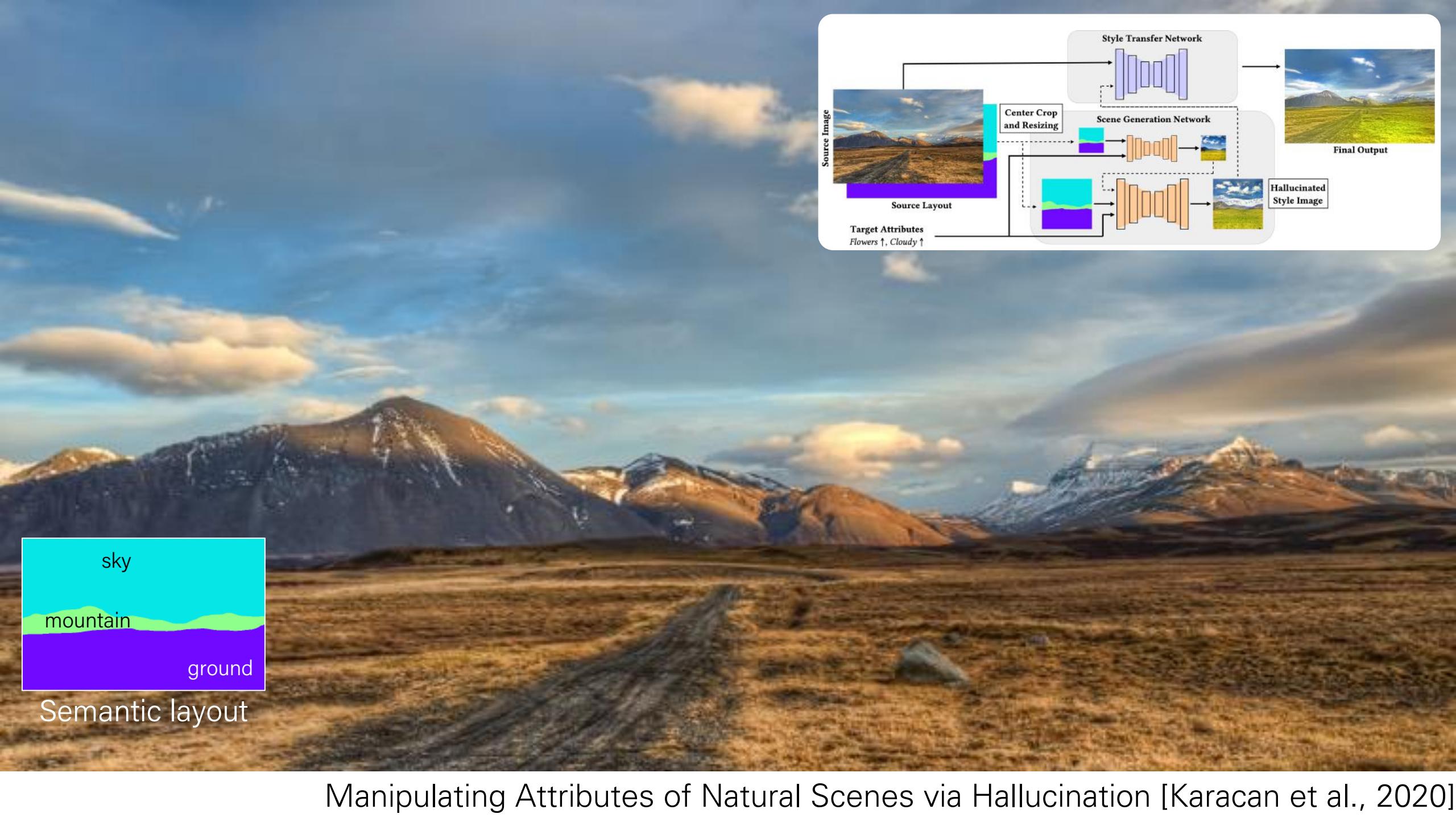
- A dataset of 17772 outdoor images with layout and transient attribute labels, formed by combining and annotated images from
 - Transient Attributes dataset (Laffont et al., 2013)
 - ADE20K dataset (Zhou et al., 2017)
- 16434 images for training, 1338 images for testing
- 150 semantic categories
- 40 transient attributes in five categories

bottle
bathtub
clock
radiator
monitor
fan
computer
scorncie
streetlight
mirror
step
column
base
shef
bus
airplane
ship
boat
bicycle
minibike
van
car
cradle
buffet
bookcase
chest of drawers
wardrobe
ottoman
bench
sofa
swivel chair
armchair
coffee table
pool table
counter
desk
cabinet
bed
escalator
stairs
sidewalk
road
screen
countertop
ceiling
pier
stage
runway
floor
fountain
tower
awning
hovel
bridge
booth
fireplace
signboard
house
banister
bar
fence
screen door
skyscraper
wall

conveyer belt
traffic light
Poster
trade name
grandstand
lake
waterfall
river
sea
food
water
sand
sky
land
towel
kitchen island
field
hill
mountain
rock
animal
person
grass
palm
flag
plate
swimming pool
air track
plaything
microwave
oven
stove
dishwasher
washer
refrigerator
chandelier
shower
toilet
pillow
book
sculpture
painting
hood
blanket
apparel
rug
curtain
bulletin board
windowpane
pole
ball
television receiver
glass
ashcan
tray
pot
bag
basket
case
box
vase
tank
barrel

lighting: sunrise/sunset, bright, daylight, etc.
weather: sunny, warm, moist, foggy, cloudy, etc.
seasons: spring, summer, autumn, winter
subjective impressions: gloomy, soothing, beautiful, etc.
additional attributes: active/busy, cluttered, dirty/polluted, lush vegetation, etc.







Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

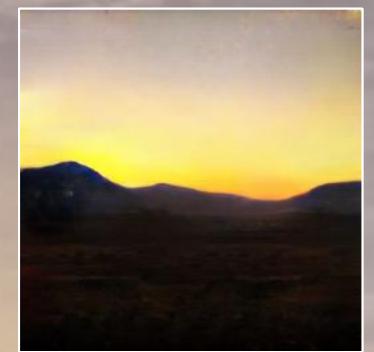


night

prediction

A wide-angle photograph of a sunset over a mountain range. The sky is filled with warm, orange and yellow hues, with scattered clouds. In the foreground, there is a dark, flat landscape, possibly a plain or a large body of water. The mountains in the background are silhouetted against the bright sky, their peaks catching some light.

sunset



prediction



snow



prediction



winter



prediction

Spring and clouds



prediction



Moist, rain and fog



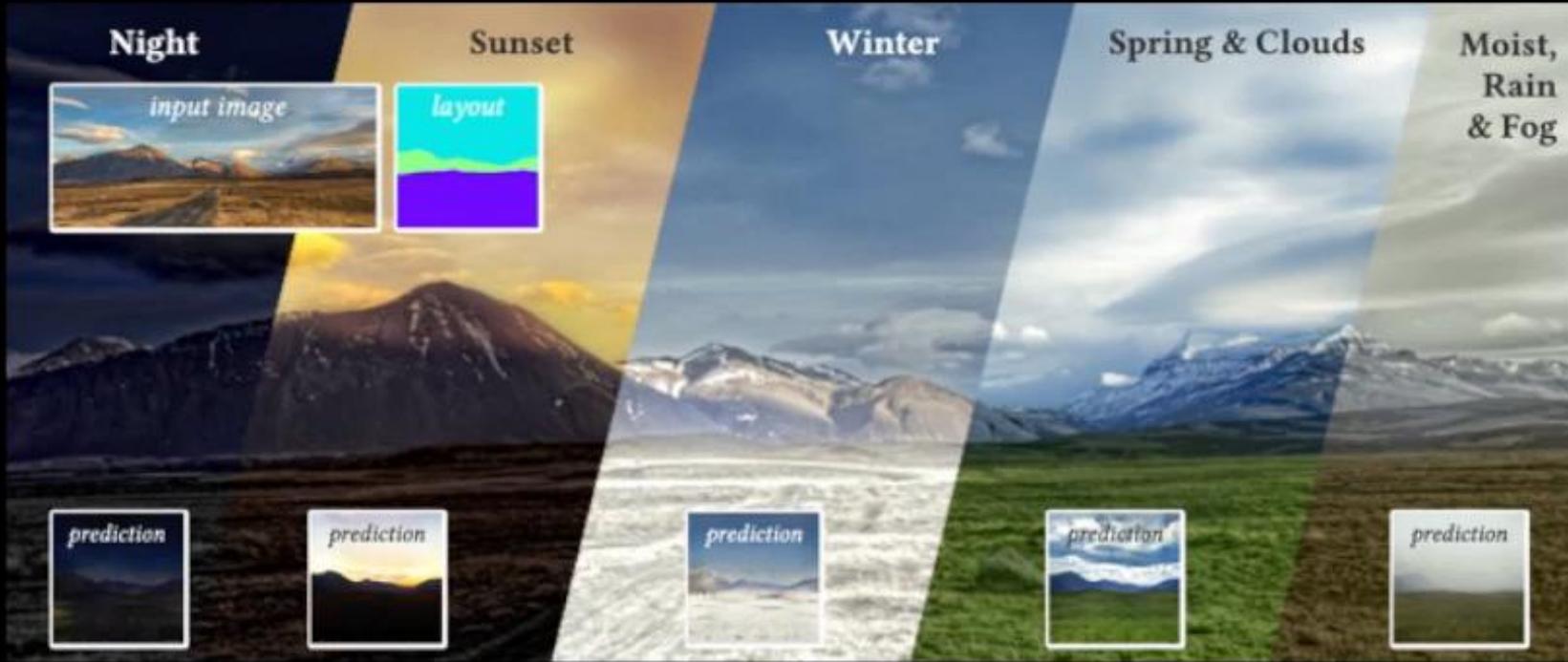
prediction



flowers



prediction

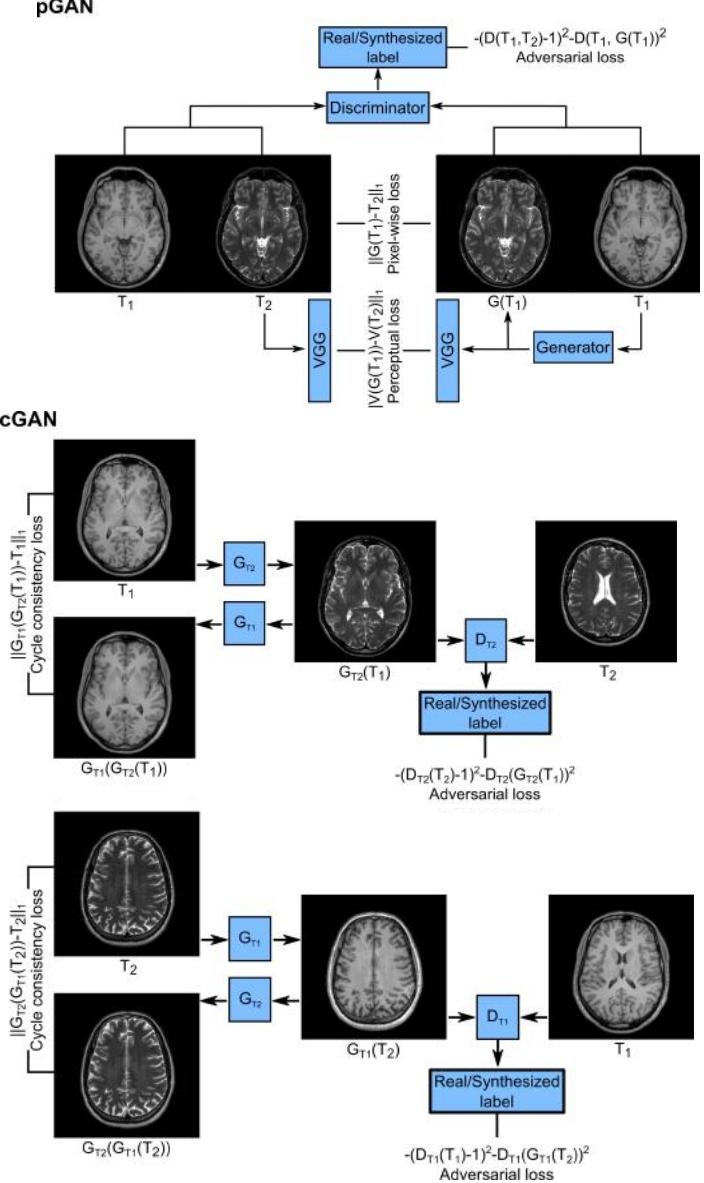
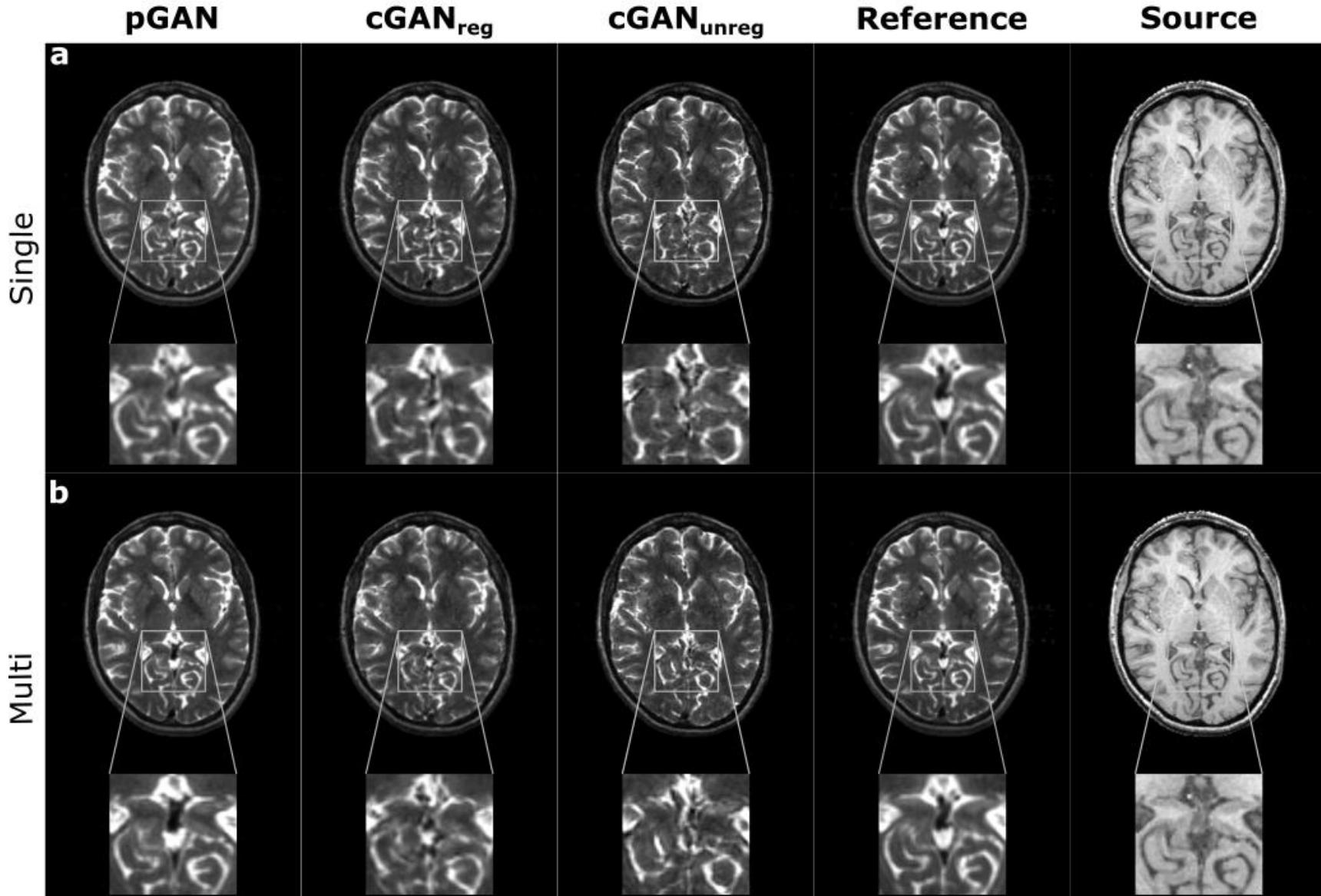


Manipulating Attributes of Natural Scenes via Hallucination

Levent Karacan, Zeynep Akata, Aykut Erdem, Erkut Erdem

ACM Transactions on Graphics

Demo



- Image Synthesis in Multi-Contrast MRI [UI Hassan Dar et al. 2019]

Next lecture:
Score-Based and
Denoising Diffusion Models