Lecture 14: Generative Models in Computer Vision

Instructor: Roni Sengupta

ULA: Andrea Dunn, William Li, Liujie Zheng

Course Website: Scan Me!

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Data: x

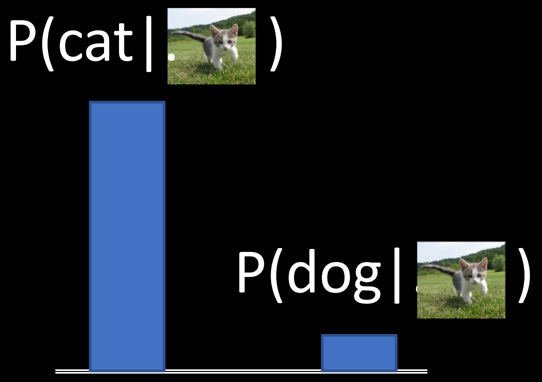
Label: y

Cat

Discriminative vs Generative Models

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

Data: x

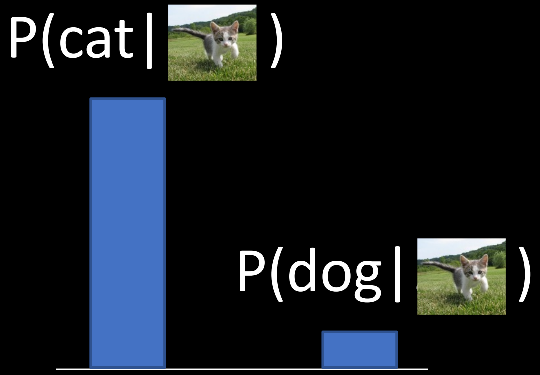


Generative Model: Learn a probability distribution p(x)

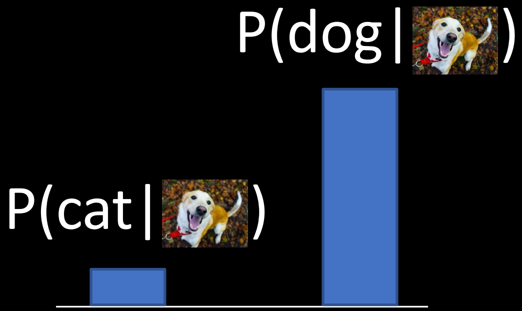
Conditional Generative Model: Learn p(x|y)

Discriminative vs Generative Models

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



Generative Model: Learn a probability distribution $p(x)$



Conditional Generative Model: Learn $p(x|y)$

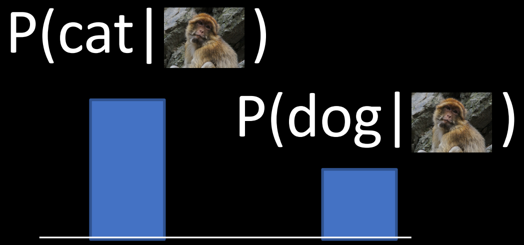
Discriminative model: the possible labels for each input "compete” for probability mass. But no competition between **images**

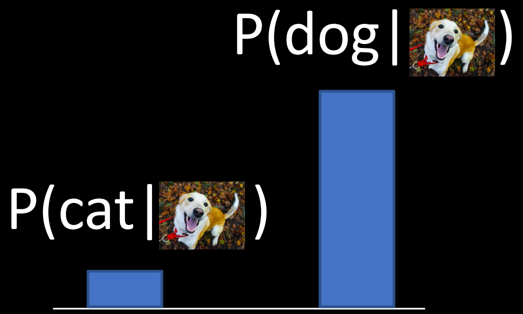
Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

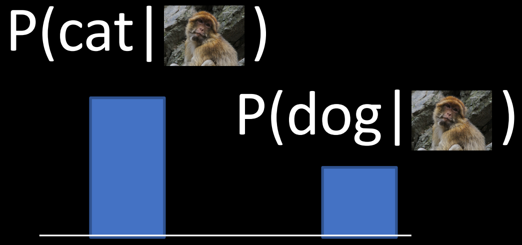




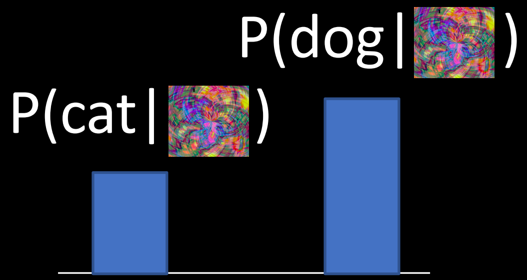
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

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



Generative Model:
Learn a probability distribution p(x)



Conditional Generative Model: Learn p(x|y)

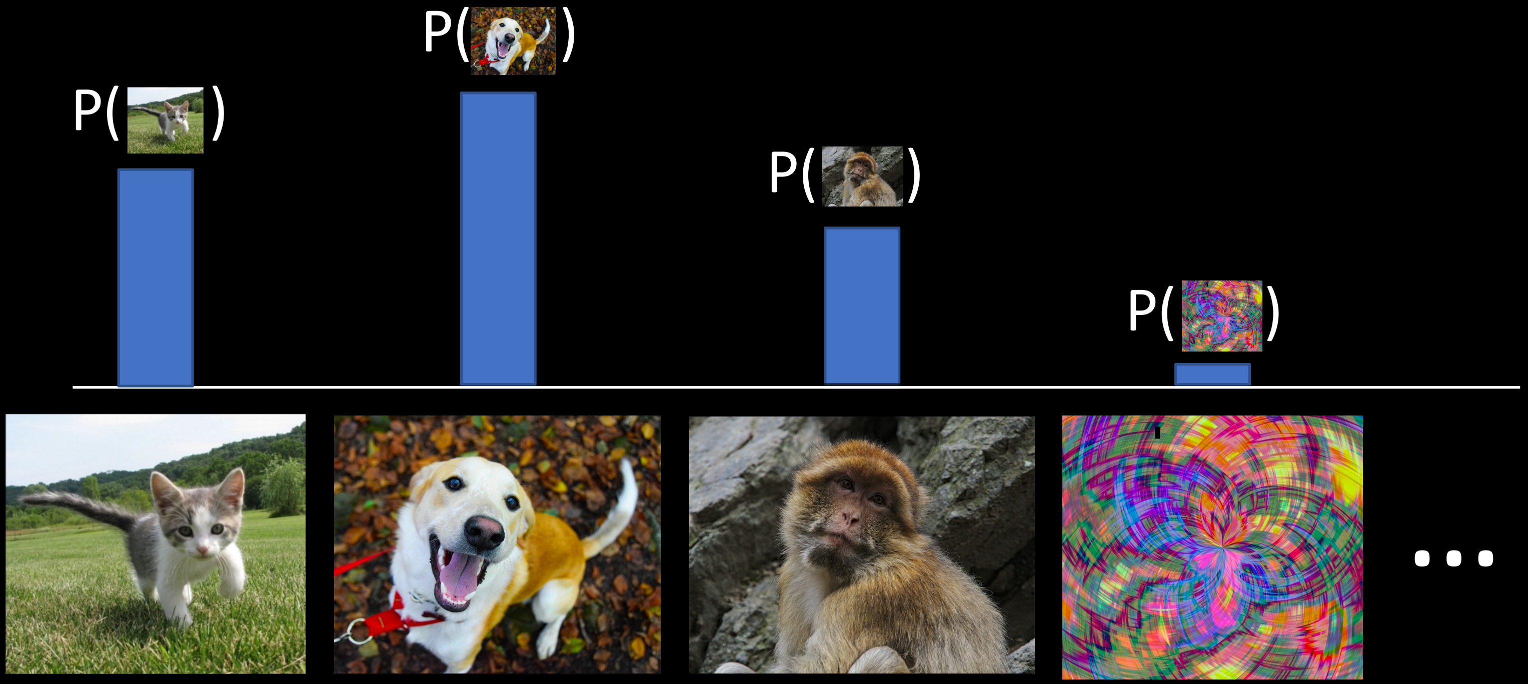
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



Generative model: All possible images compete with each other for probability mass

Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

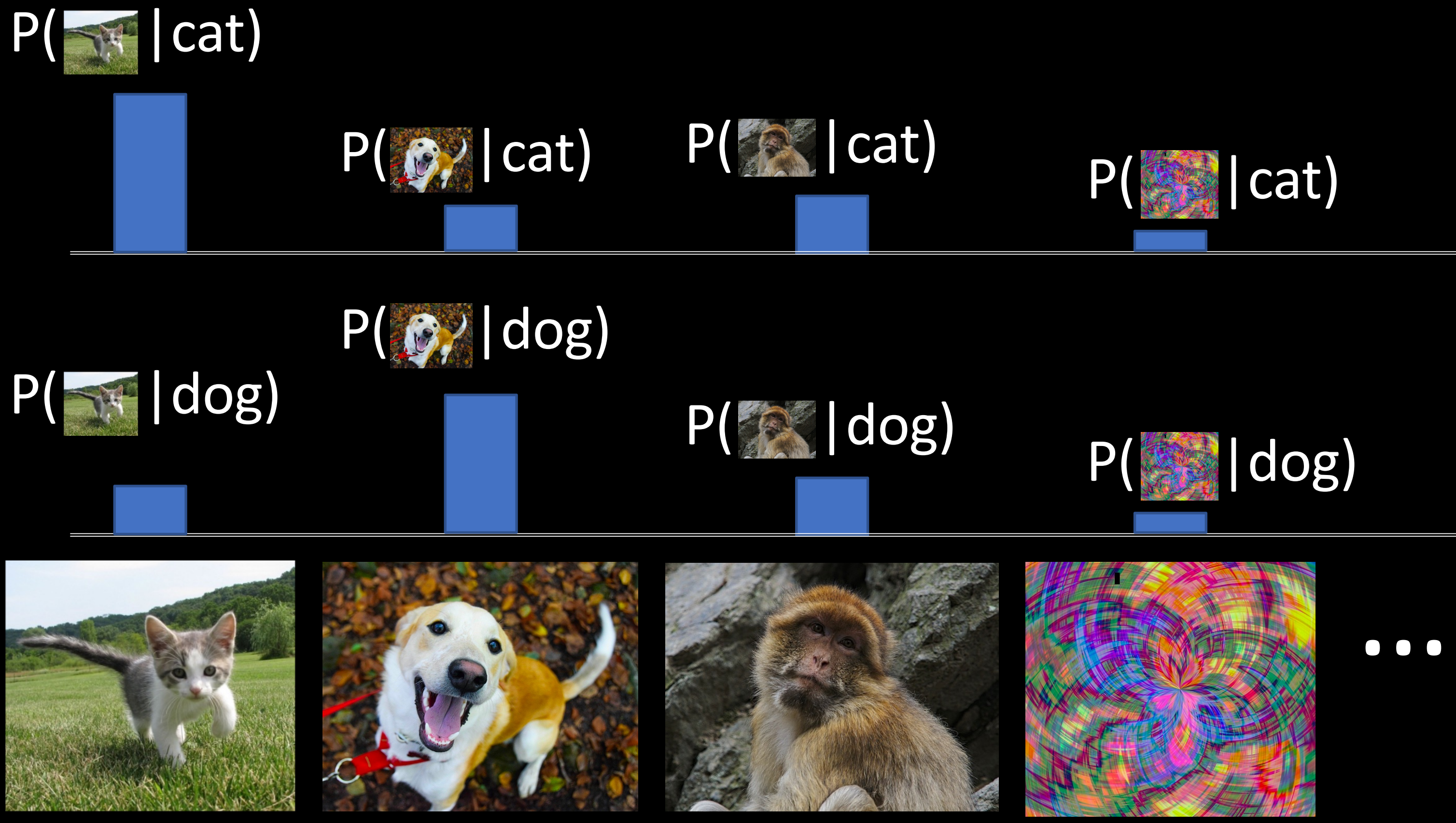
Model can “reject” unreasonable inputs by assigning them small values

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



Conditional Generative Model: Each possible label induces a competition among all images

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Recall **Bayes' Rule:**

$$P(x \mid y) = \frac{P(y \mid x)}{P(y)} P(x)$$

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Recall **Bayes' Rule:**

$$\underbrace{\boxed{P(x \mid y)}}_{\text{Conditional Generative Model}} = \frac{\overbrace{\boxed{P(y \mid x)}}^{\text{Discriminative Model}}}{\underbrace{\boxed{P(y)}}_{\text{Prior over labels}}} \overbrace{\boxed{P(x)}}^{\text{(Unconditional) Generative Model}}$$

We can build a conditional generative model from other components!

What can we do with a discriminative model?

- **Discriminative Model:** Learn a probability distribution p(y|x) → Assign labels to data; Feature learning (with labels)

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

- **Conditional Generative Model:** Learn p(x|y)

What can we do with a generative model?

- **Discriminative Model:** Learn a probability distribution p(y|x) → Assign labels to data; Feature learning (with labels)

- **Generative Model**: Learn a probability distribution p(x) → Detect outliers; Feature learning (without labels); Sample to **generate** new data

- **Conditional Generative Model:** Learn p(x|y)

What can we do with a generative model?

- **Discriminative Model:** Learn a probability distribution p(y|x) → Assign labels to data; Feature learning (with labels)

- **Generative Model**: Learn a probability distribution p(x) → Detect outliers; Feature learning (without labels); Sample to **generate** new data

- **Conditional Generative Model:** Learn p(x|y) → Assign labels, while rejecting outliers!; Generate new data conditioned on input labels

Introduction to Generative Models
(Conditional and Unconditional)

What cool things can we do with it?

PLAY ABOUT METHODS LEARN PRESS CONTACT BOOK CALLING BS

Click on the person who is real.

https://www.whichfaceisreal.com/index.php

Posted by u/Pit-Fiend_Fyrine-IV 4 days ago

1.6k

Testing animatediff + qr code monster

Animation | Video nsfw

AI Art!

:60 NSFW

SD animatediff + controlnet

48F @ 24 FPS | model: Photon | CFG 8 | Euler
CN_QRcode monster
512x512 > Upscale > Topaz
OCT 2023

Taxonomy of Generative Models

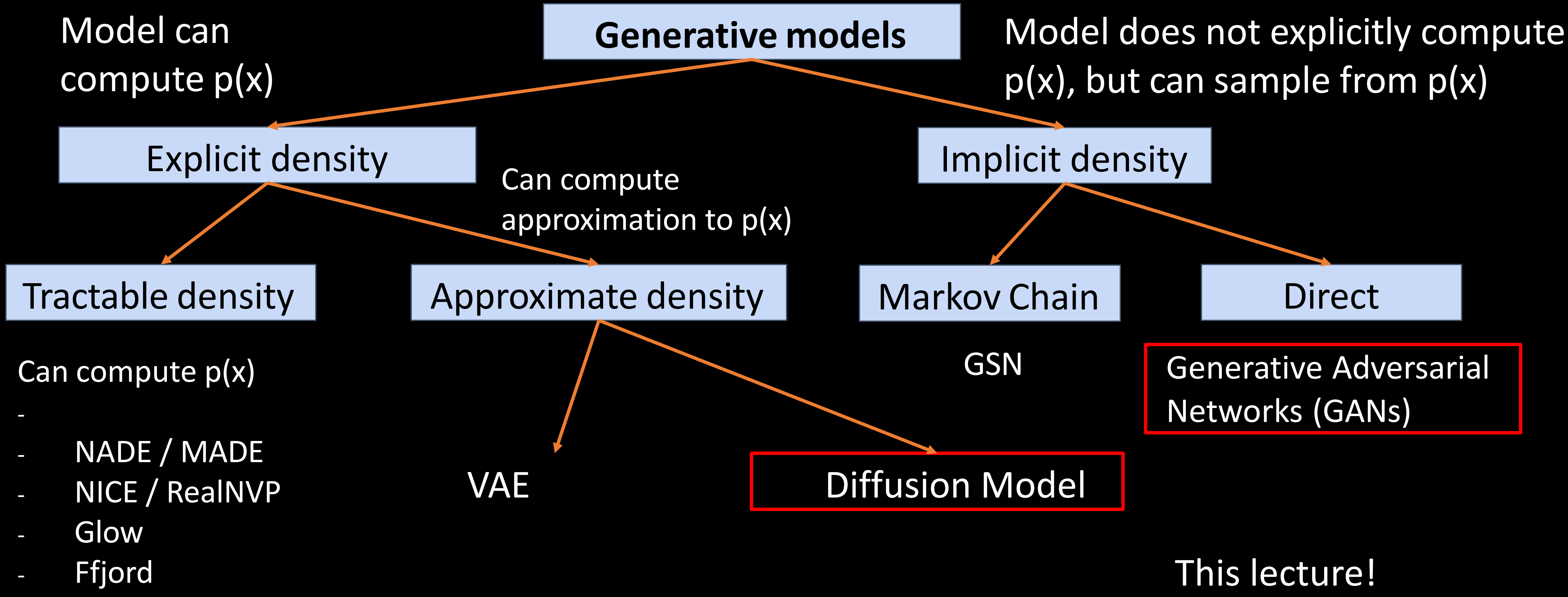


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Generative Adversarial Networks

Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: Introduce a latent variable z with simple prior p(z).

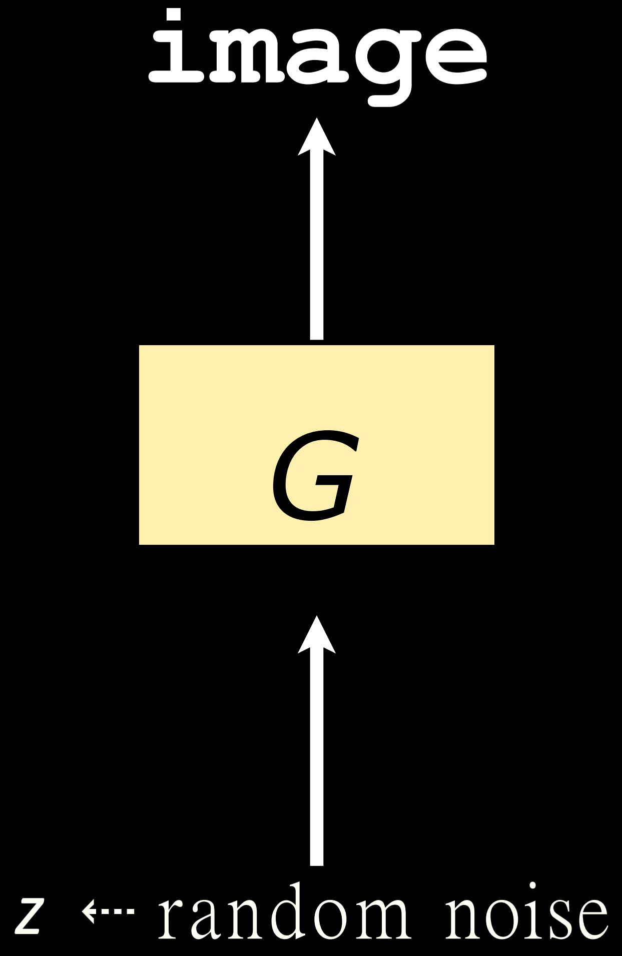
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)

Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Network takes a random input and produces a sample from the data distribution as output

Generator

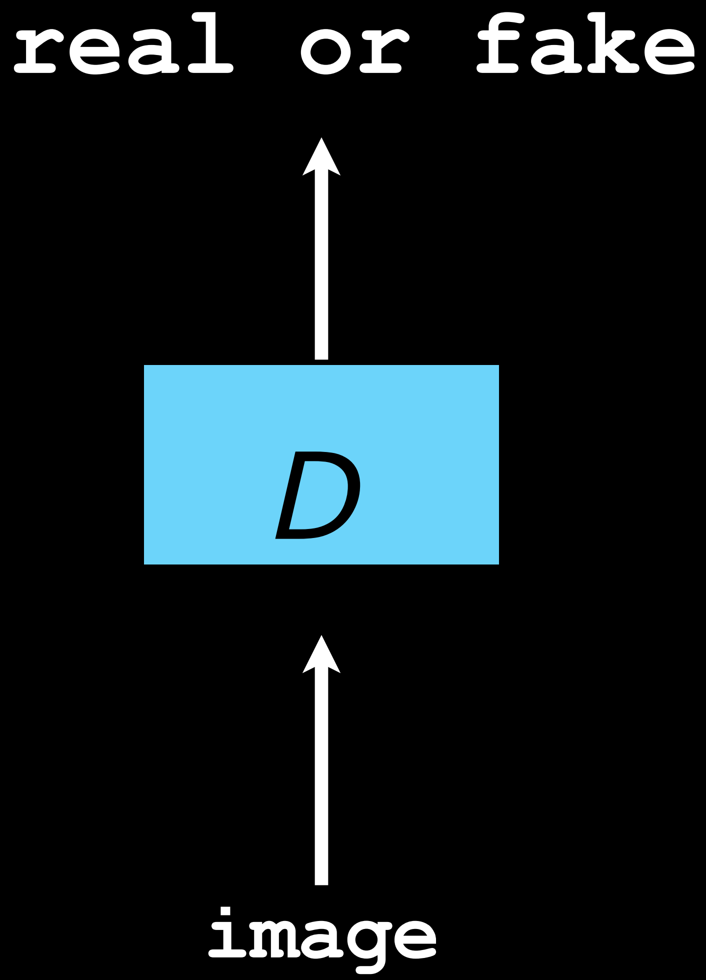


Network classifies input as “real” or “fake”

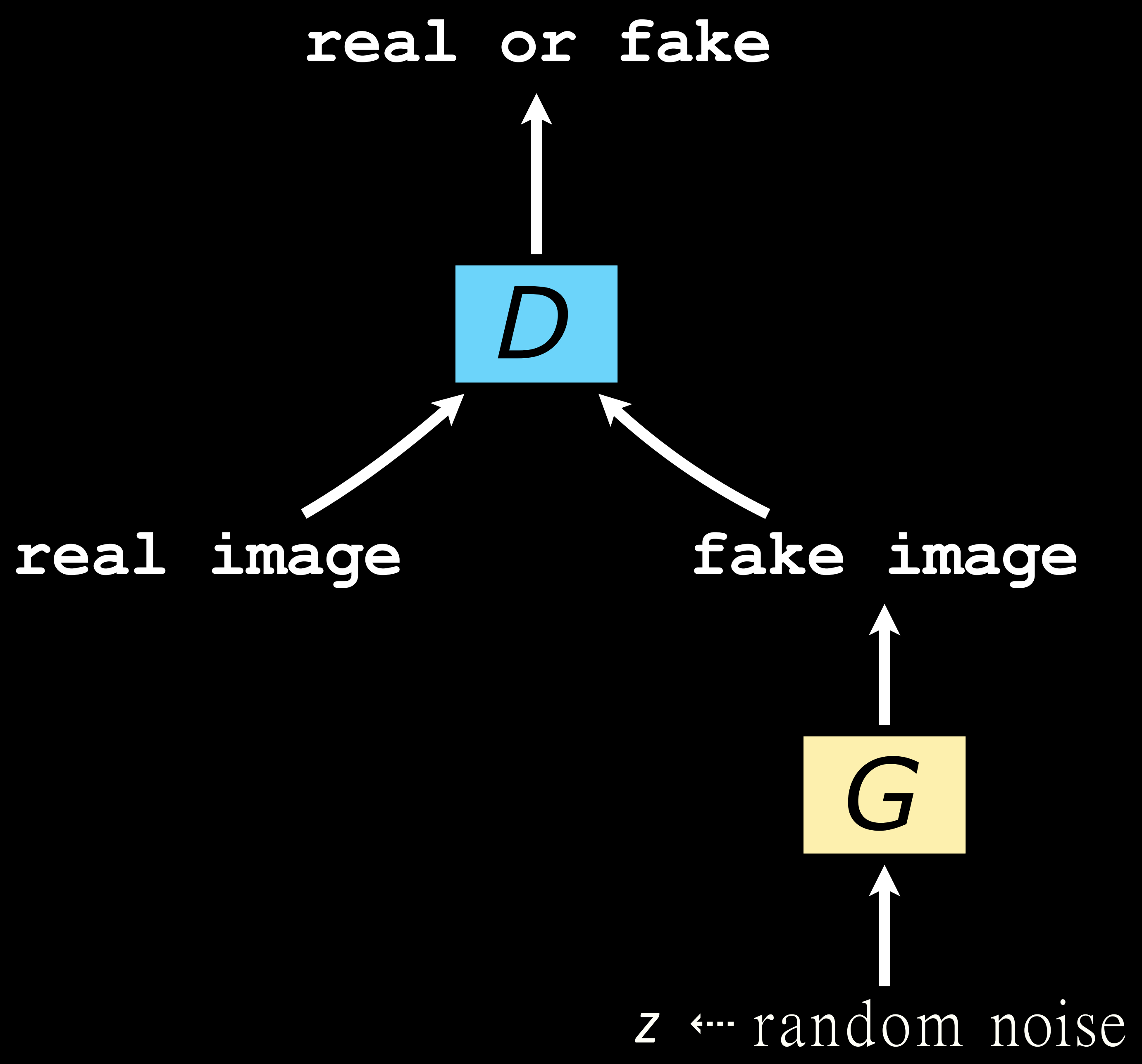
Discriminator

Network classifies input as “real” or “fake”

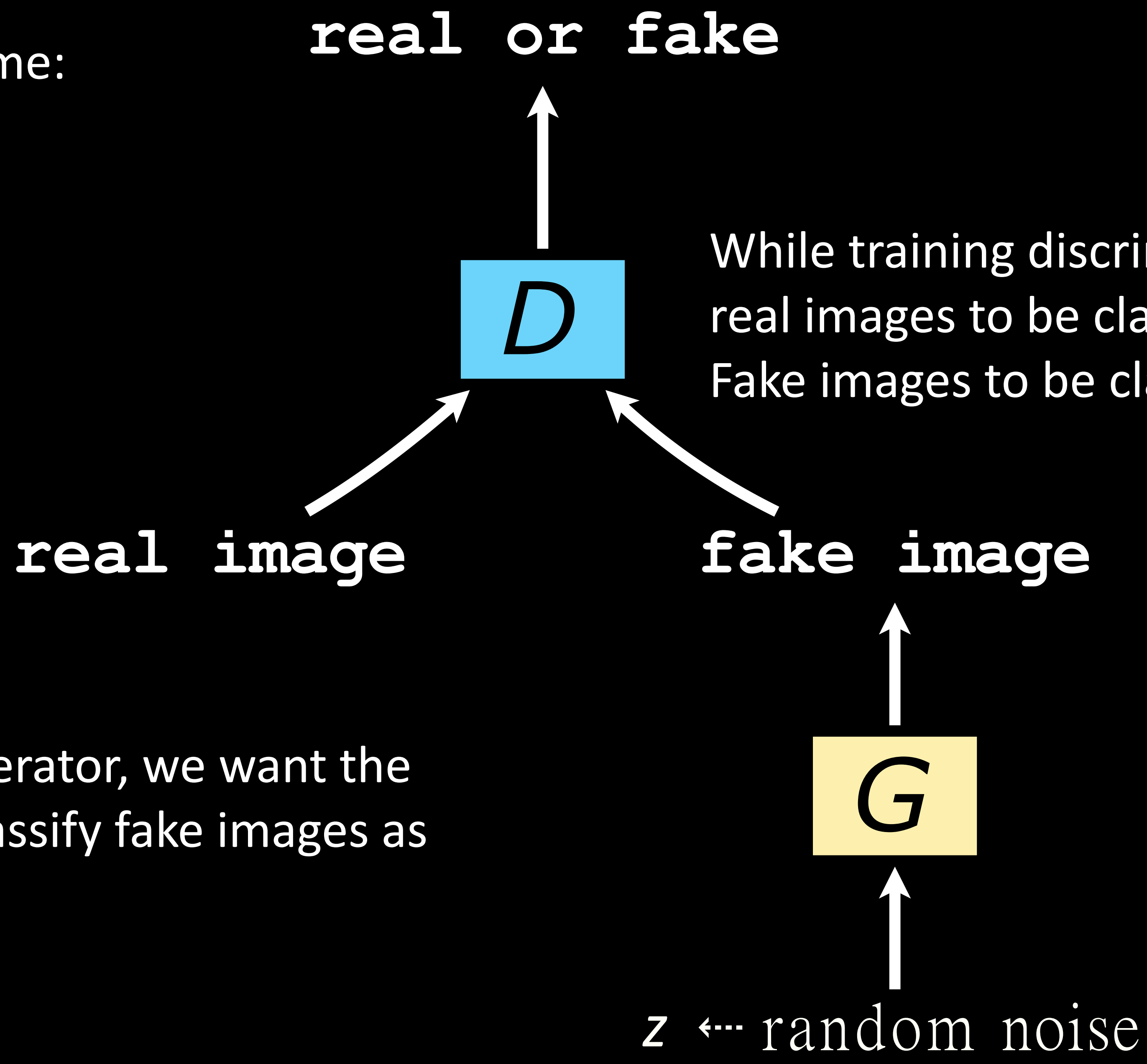
“fake” inputs come from the generator


real or fake
D
image


Discriminator

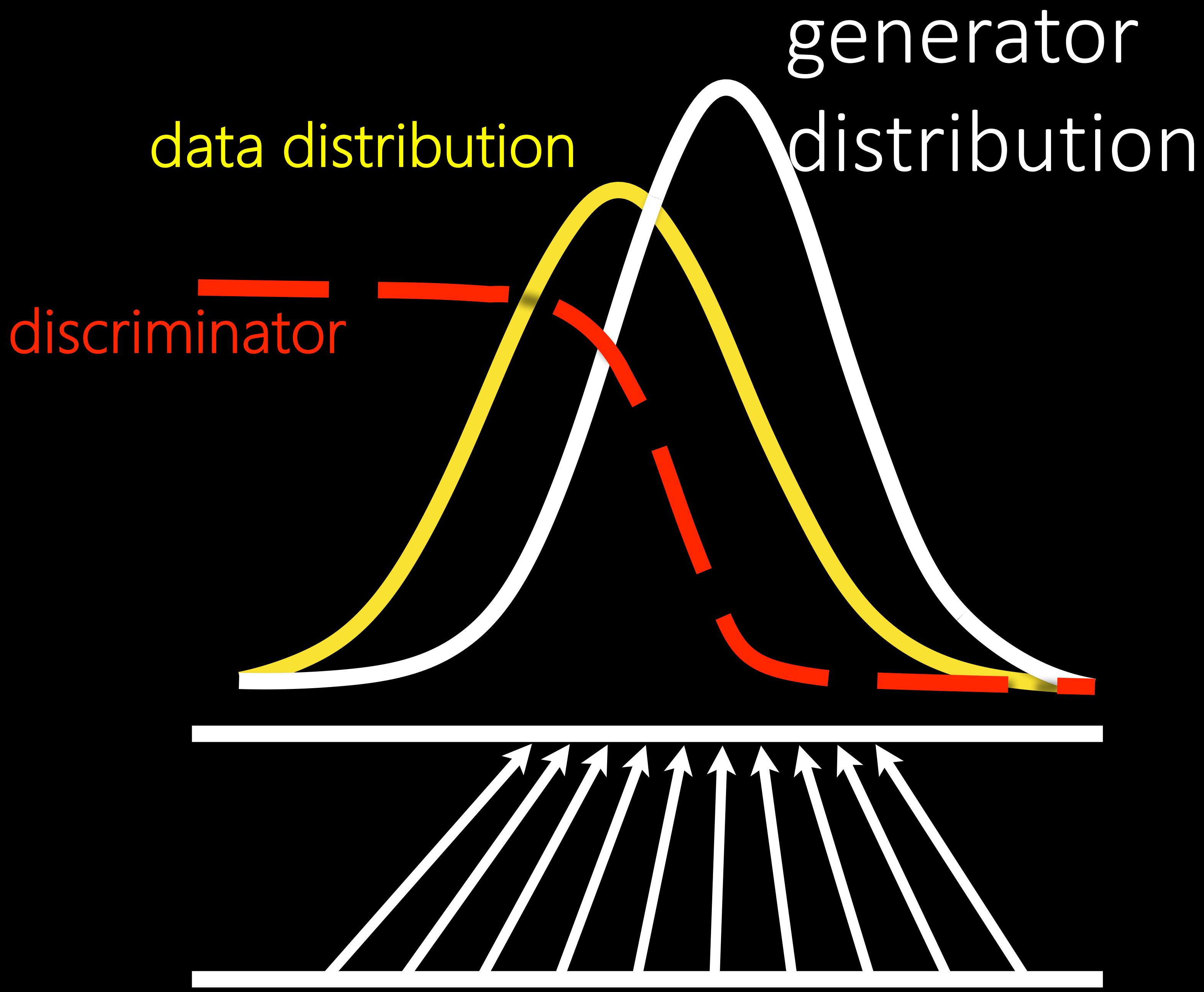

real or fake
D
real image
fake image
G
z ⟵ random noise

In practice we assume:
'real' label = 1
'fake' label = 0



While training discriminator, we want:
real images to be classified as 1
Fake images to be classified as 0

While training generator, we want the discriminator to classify fake images as real (label=1).


generator
distribution
data distribution
discriminator

GAN training

```
for number of training iterations do

   for k steps do
      sample m noise samples from noise prior
      sample m real examples from dataset
      update the discriminator by gradient ascent
   end for

   sample m noise samples from noise prior
   update generator by stochastic gradient ascent

end for
```

```
for number of training iterations do

   for k steps do
      sample m noise samples from noise prior
      sample m real examples from dataset
      update the discriminator by gradient ascent
   end for

   sample m noise samples from noise prior
   update generator by stochastic gradient ascent

end for
```

```
for number of training iterations do
    for k steps do
        sample m noise samples from noise prior
        sample m real examples from dataset
        update the discriminator by gradient ascent
    end for
    sample m noise samples from noise prior
    update generator by stochastic gradient ascent
end for
```

```
for number of training iterations do

    for k steps do
        sample m noise samples from noise prior
        sample m real examples from dataset
        update the discriminator by gradient ascent
    end for

    update generator by stochastic

end for
```

update discriminator

```
for number of training iterations do

   for k steps do
      sample m noise samples from noise prior
      sample m real examples from dataset
      update the discriminator by gradient ascent
   end for

   sample m noise samples from noise prior
   update generator by stochastic gradient ascent

end for
```

```
for number of training iterations do

   for k steps do
      sample m noise samples from noise prior
      sample m real examples from dataset
      update the discriminator by gradient ascent
   end for

   sample m noise samples from noise prior
   update generator by stochastic gradient ascent

end for
```

While training discriminator, we want:

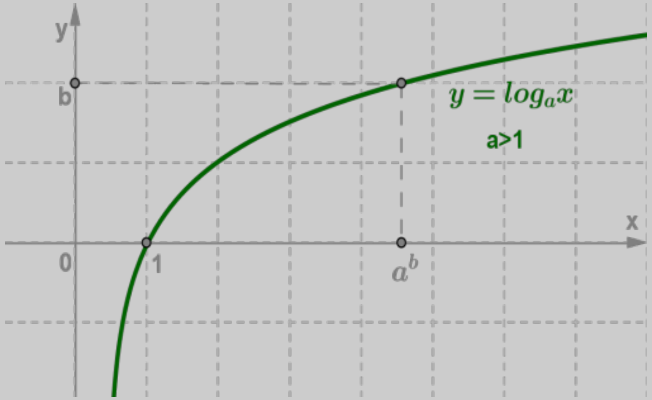
Real images to be classified as 1

Fake images to be classified as 0

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

sample m noise samples from noise prior

update the discriminator by gradient ascent



```
for number of training iterations do

    for k steps do
        sample m noise samples from noise prior
        sample m real examples from dataset
        update the discriminator by gradient ascent
    end for

    sample m noise samples from noise prior
    update generator by stochastic gradient ascent

end for
```

```
for number of training iterations do

    for k steps do
        sample m noise samples from noise prior
        sample m real examples from dataset
        update the discriminator by gradient ascent
    end for

    sample m noise samples from noise prior
    update generator by stochastic gradient ascent
```

update generator using modified objective

```
for number of training iterations do

   for k steps do
      sample m noise samples from noise prior
      sample m real examples from dataset
      update the discriminator by gradient ascent

   end for

   sample m noise samples from noise prior
   update generator by stochastic gradient ascent

end for
```

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

While training generator, we want the discriminator to classify fake images as real (label=1).
Discriminator weight is fixed, only update generator

GAN training

```
for number of training iterations do

    for k steps do
        sample m noise samples from noise prior
        sample m real examples from dataset
        update the discriminator by gradient ascent
    end for

    sample m noise samples from noise prior
    update generator by stochastic gradient ascent

end for
```

How is the quality of generated images assessed?

Two simple properties for evaluation metric:

- Fidelity: We want our GAN to generate *high* quality images.
- Diversity: Our GAN should generate images that are inherent in the training dataset.

Feature Distance:

- Use a pre-trained image classification model (neural network).
- Pass an image through the model and use the activation of intermediate layers as features.
- Calculate any distance metric (L2/L1) between the features of generated image and GT real image.
- LPIPS metric (Learned Perceptual Image Patch Similarity).

But often, we do not have the GT image to compare with.

What do we do?

FID (Frechet Inception Distance)

Frechet Distance between two univariate gaussian distribution

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

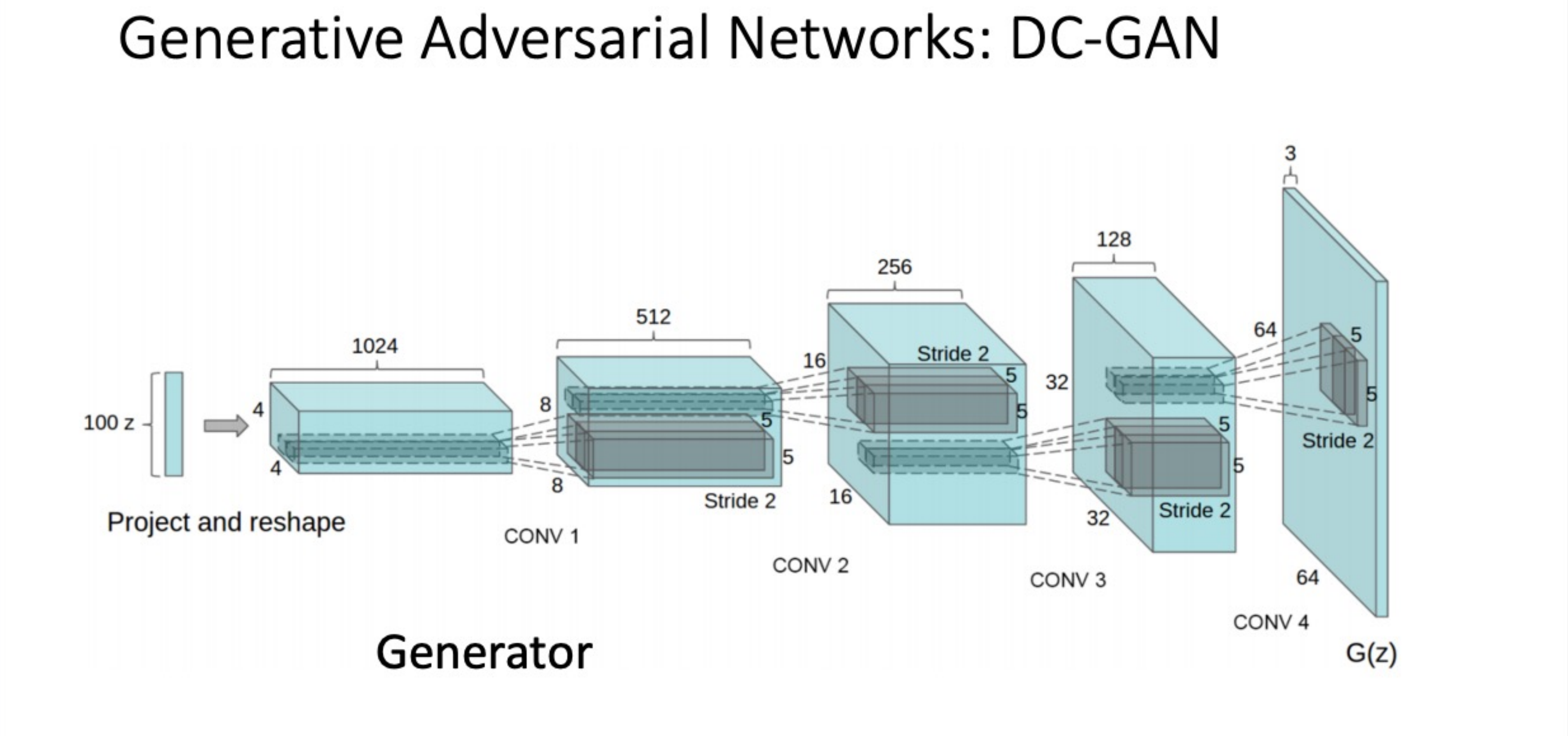
Frechet Distance between two multi-variate gaussian distribution

$$FID = ||\mu_X - \mu_Y||^2 - Tr(\sum_X + \sum_Y - 2 \quad \sum_X \sum_Y)$$

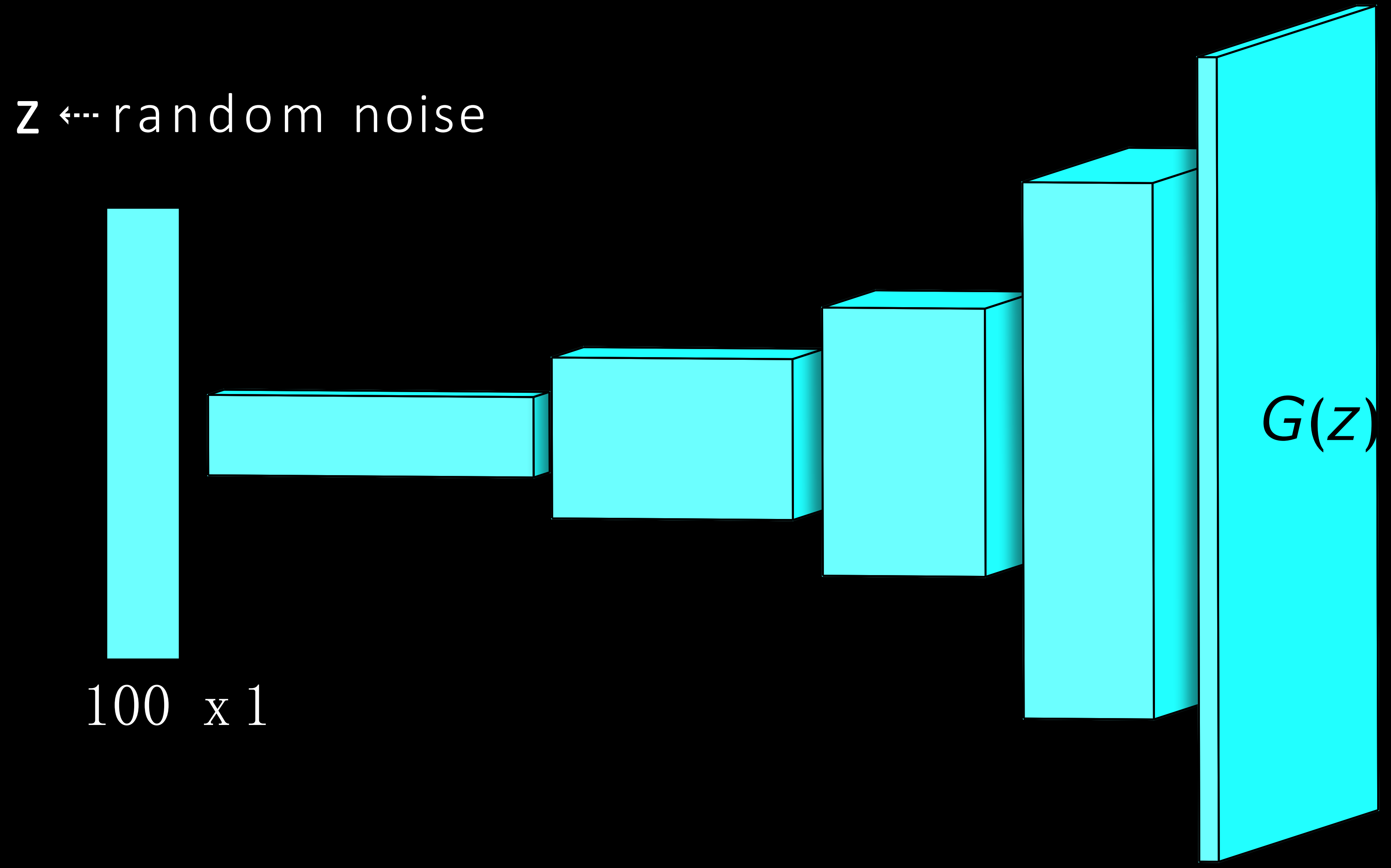
Frechet Inception Distance (FID), X and Y are features of Inception V3 classification model for real and fake images respectively.

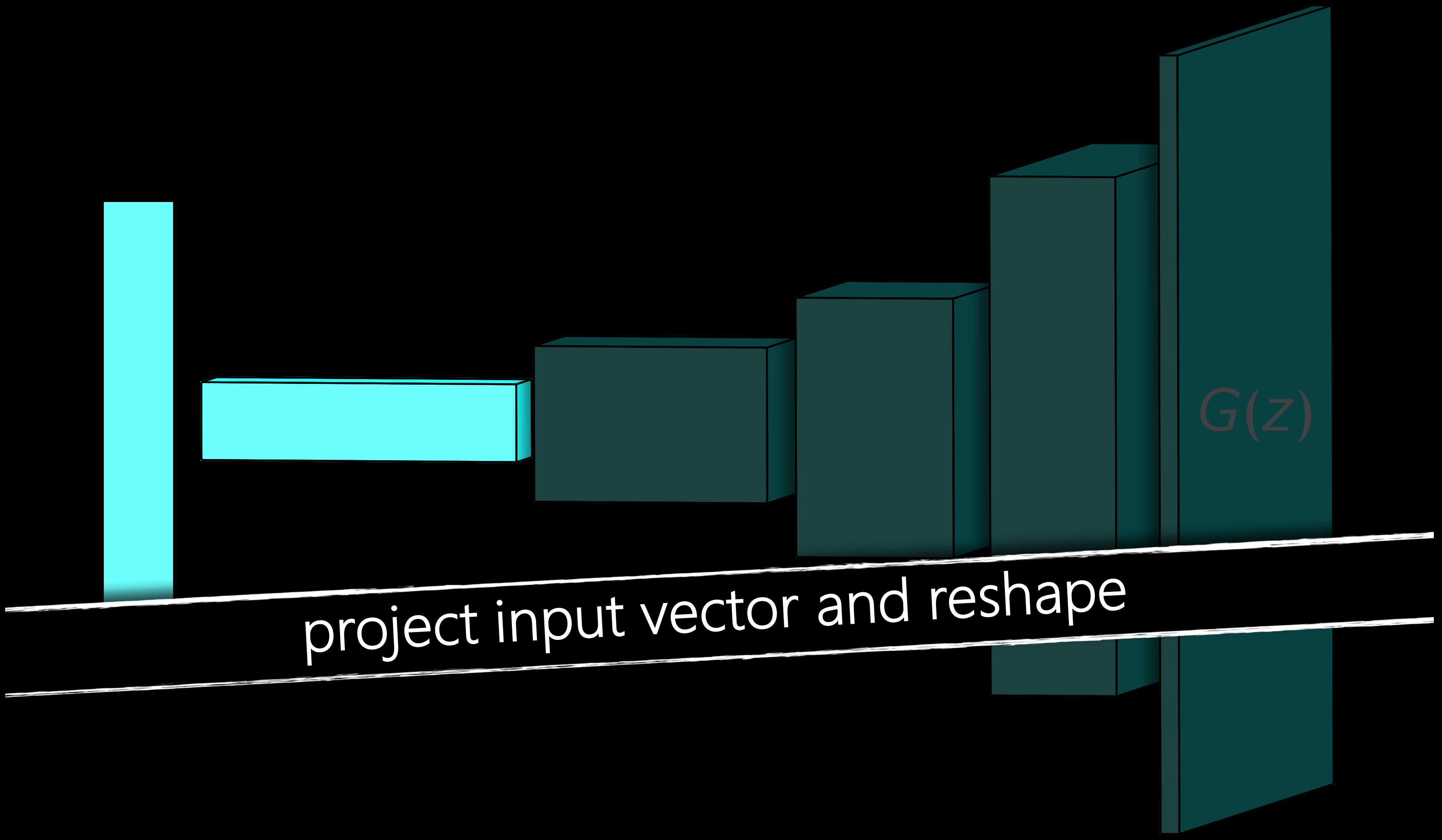
Note: The loss is between set of real and fake images, not individual real and fake image!

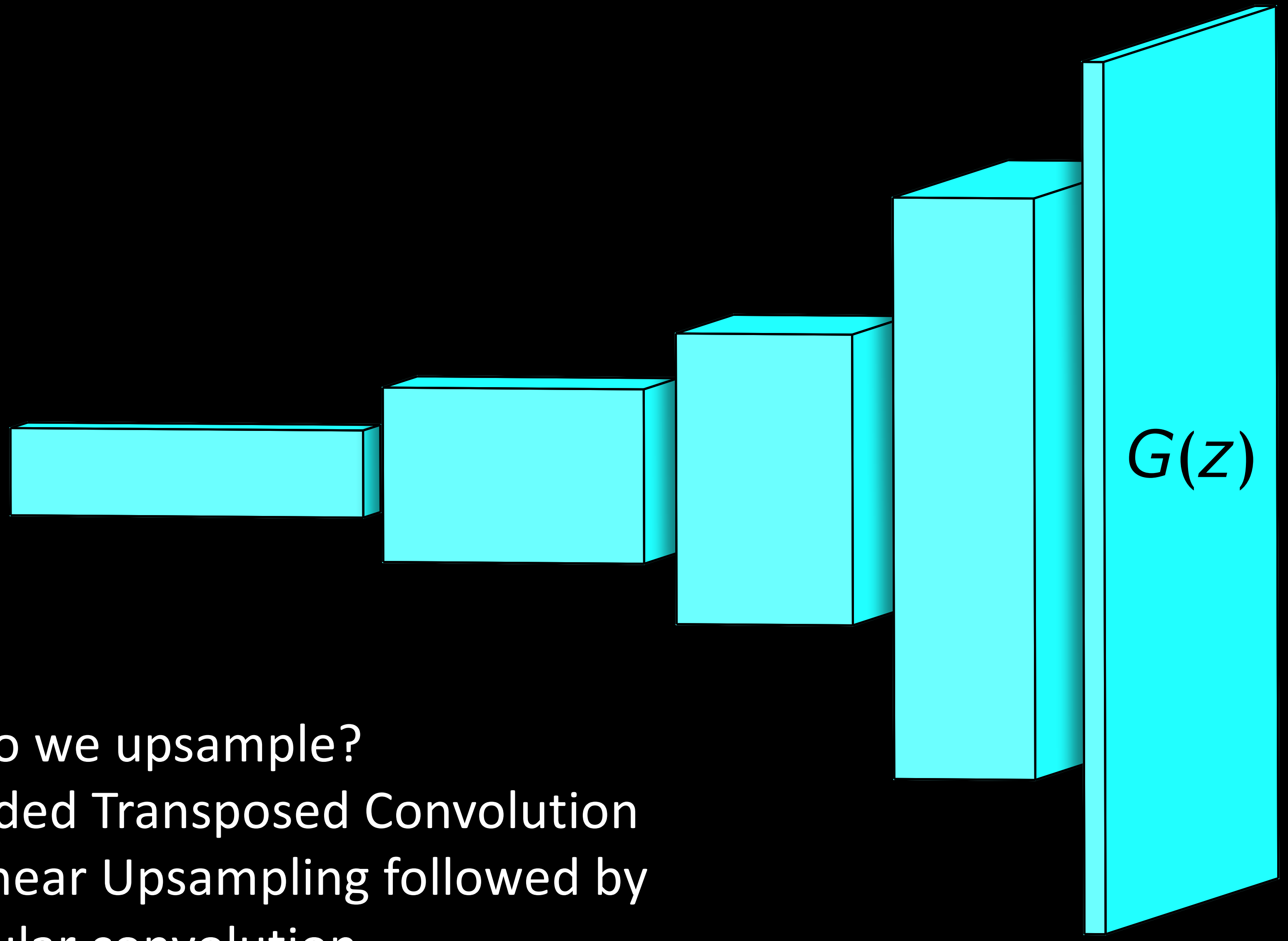
HQ image synthesis with GANs



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016


z ⇠ random noise
100 x 1
G(z)


G(z)
project input vector and reshape



How do we upsample?

- Strided Transposed Convolution
- Bilinear Upsampling followed by regular convolution.

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

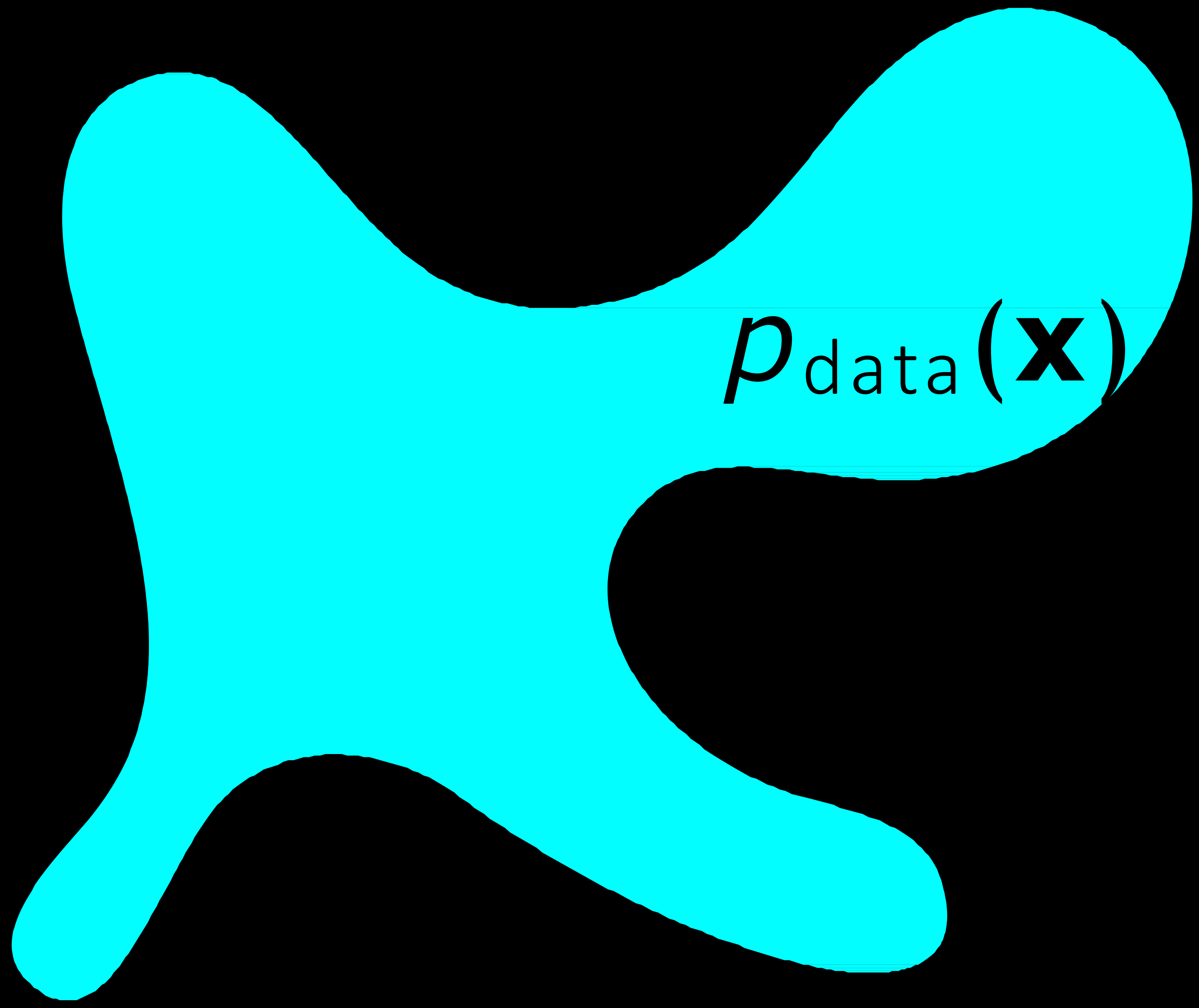
Alec Radford, Luke Metz and Soumith Chintala

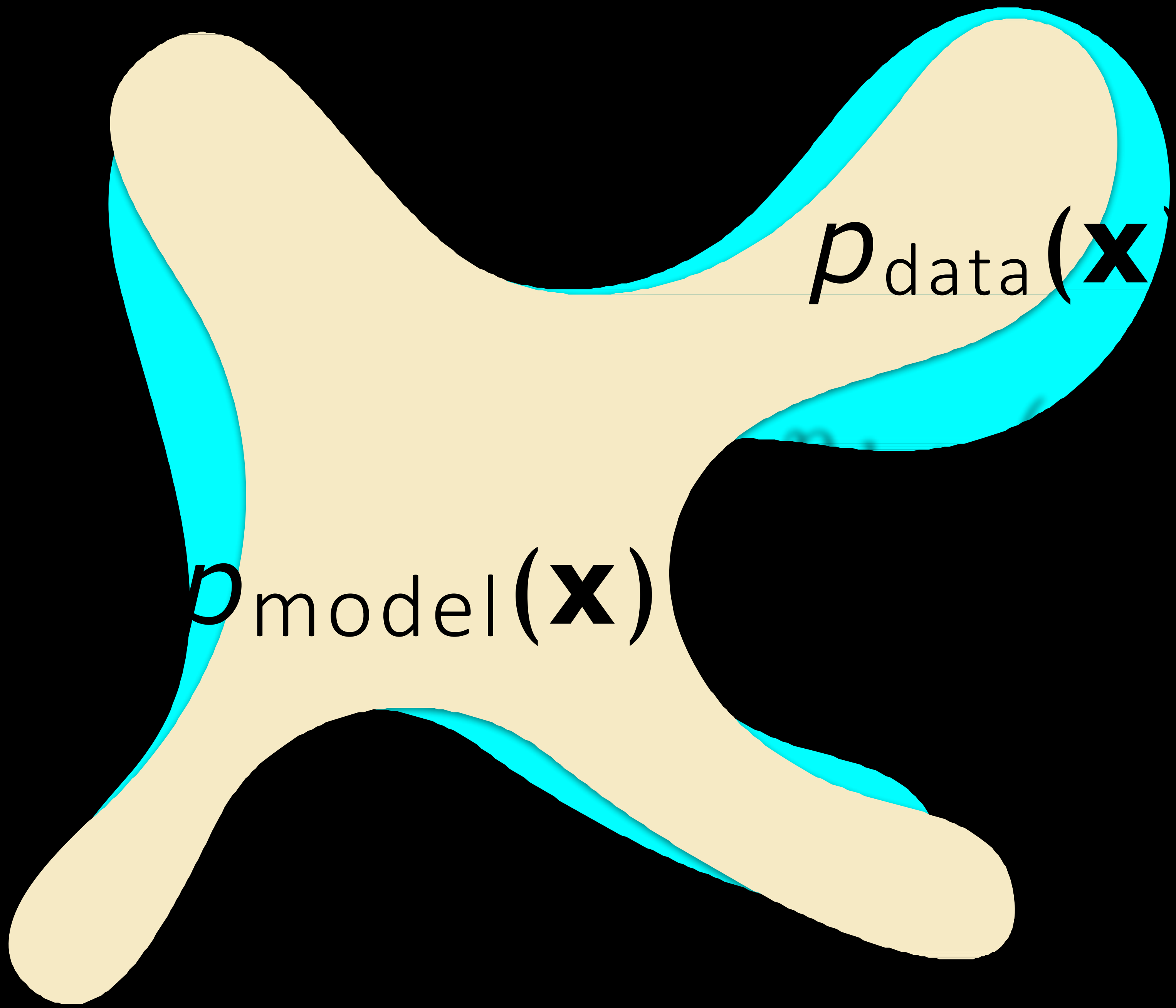
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

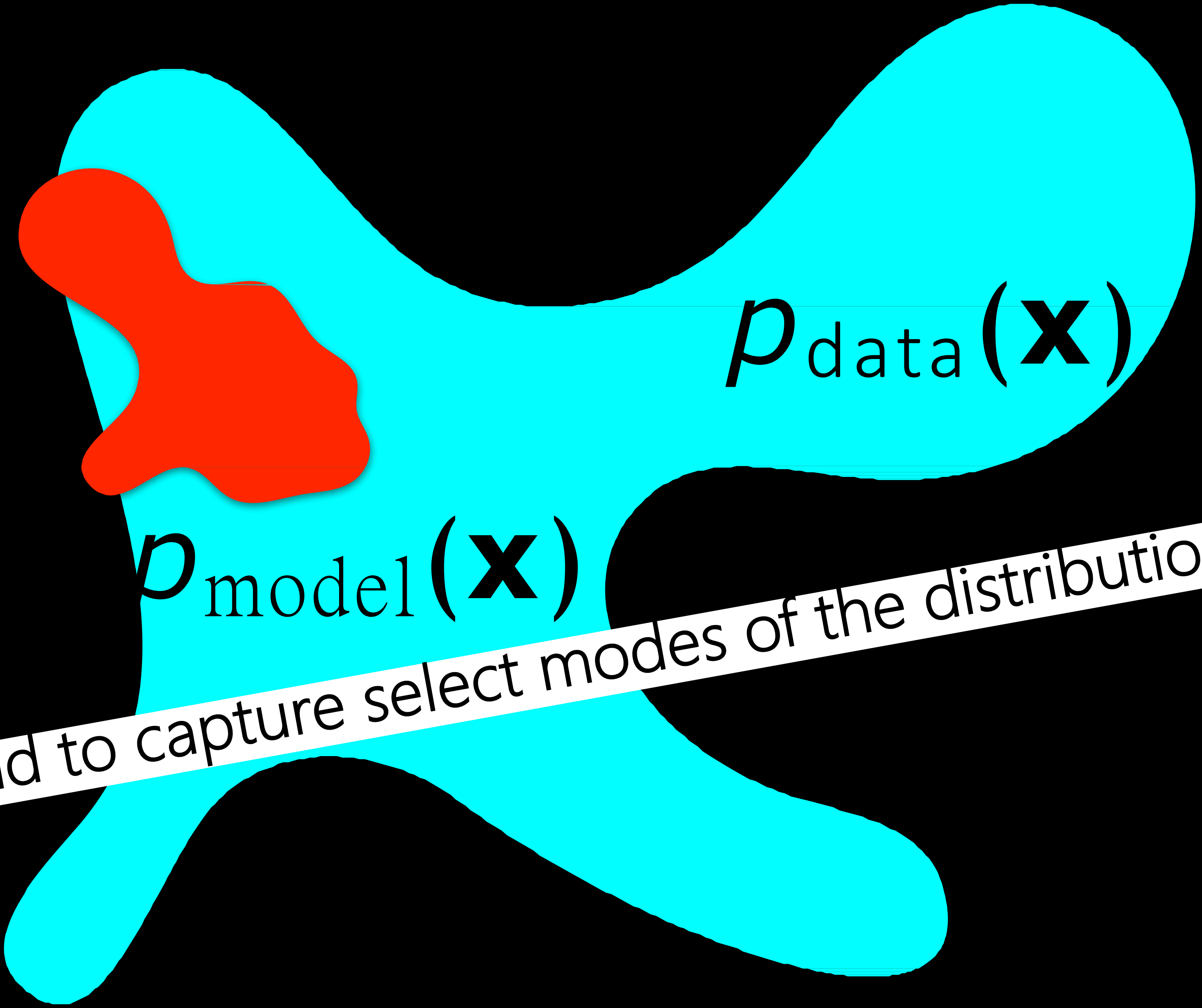
Alec Radford, Luke Metz and Soumith Chintala

GAN training can be

UNSTABLE


$p_{\text{data}}(\mathbf{x})$


$p_{\text{data}}(\mathbf{x})$
$p_{\text{model}}(\mathbf{x})$



tend to capture select modes of the distribution

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m}\sum_{i=1}^{m}\left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right)\right]$	$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

real=1, fake=0

Make fake into real (1)

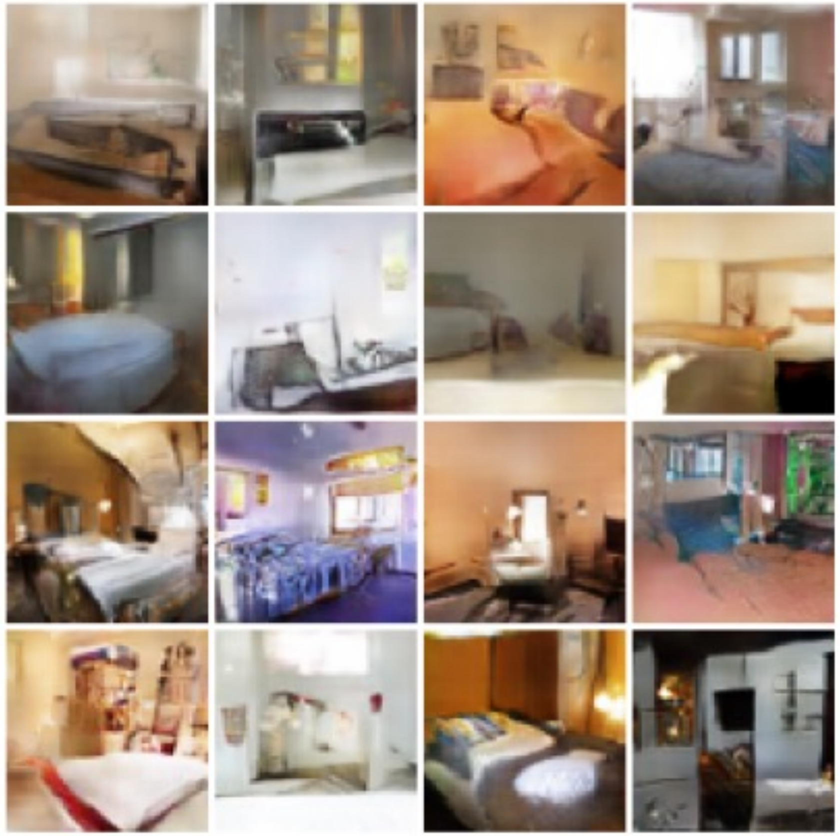
WGAN:
Instead of classifying fake and real image,
simply minimize the discriminator score for real image
and maximize the discriminator score for fake images.

Instead of classifying all fake images as real,
simply minimize the discriminator score.

Helps in problems of vanishing gradient

GAN Improvements: Improved Loss Functions

Wasserstein GAN (WGAN)



Arjovsky, Chintala, and Bouttou, “Wasserstein GAN”, 2017

WGAN with Gradient Penalty (WGAN-GP)

Gulrajani et al, “Improved Training of Wasserstein GANs”, NeurIPS 2017

GAN Improvements: Higher Resolution

256 x 256 bedrooms

1024 x 1024 faces

Karras et al, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

Progressive GAN

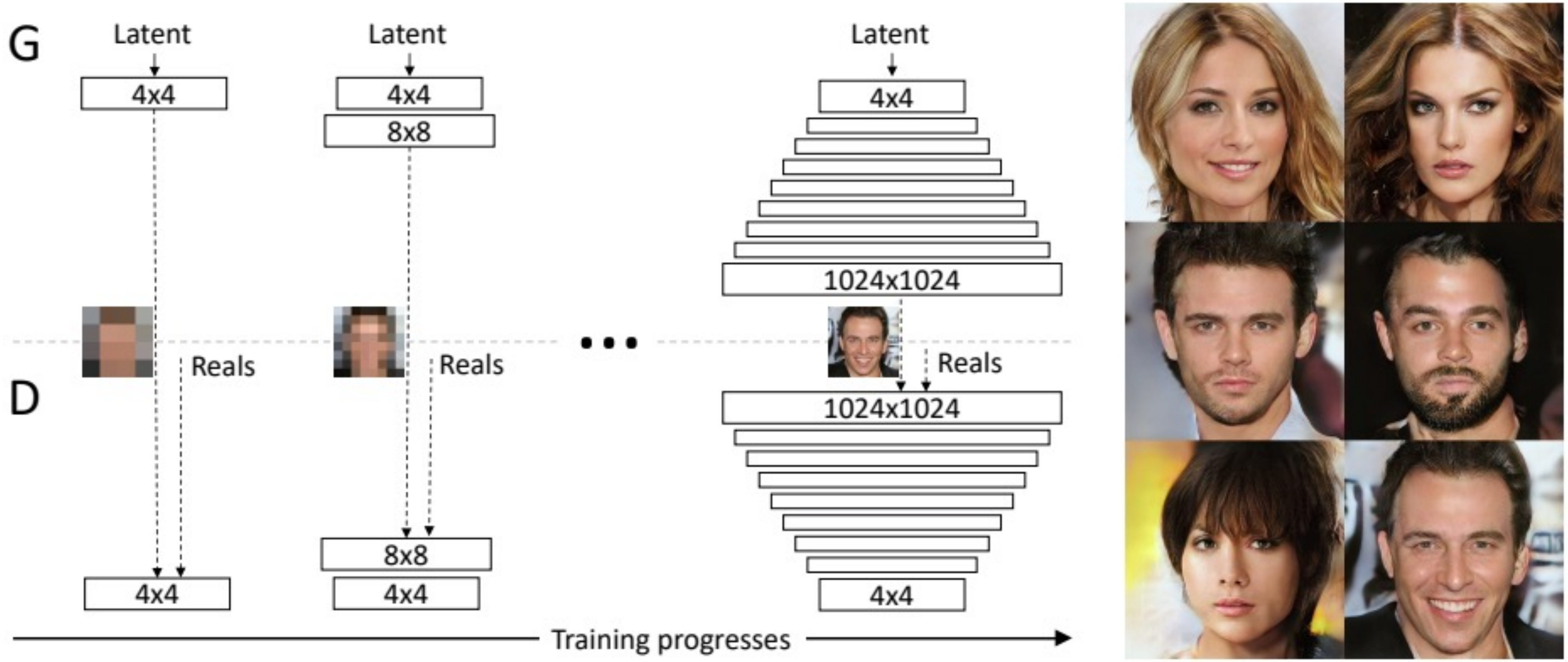
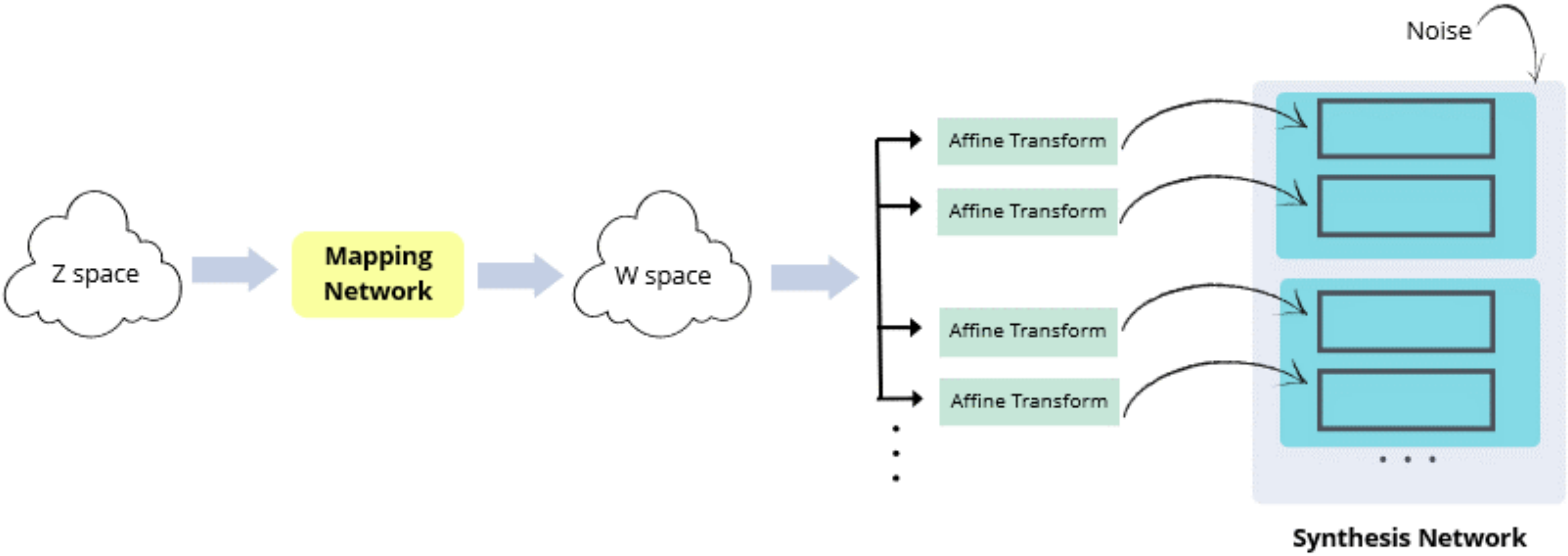
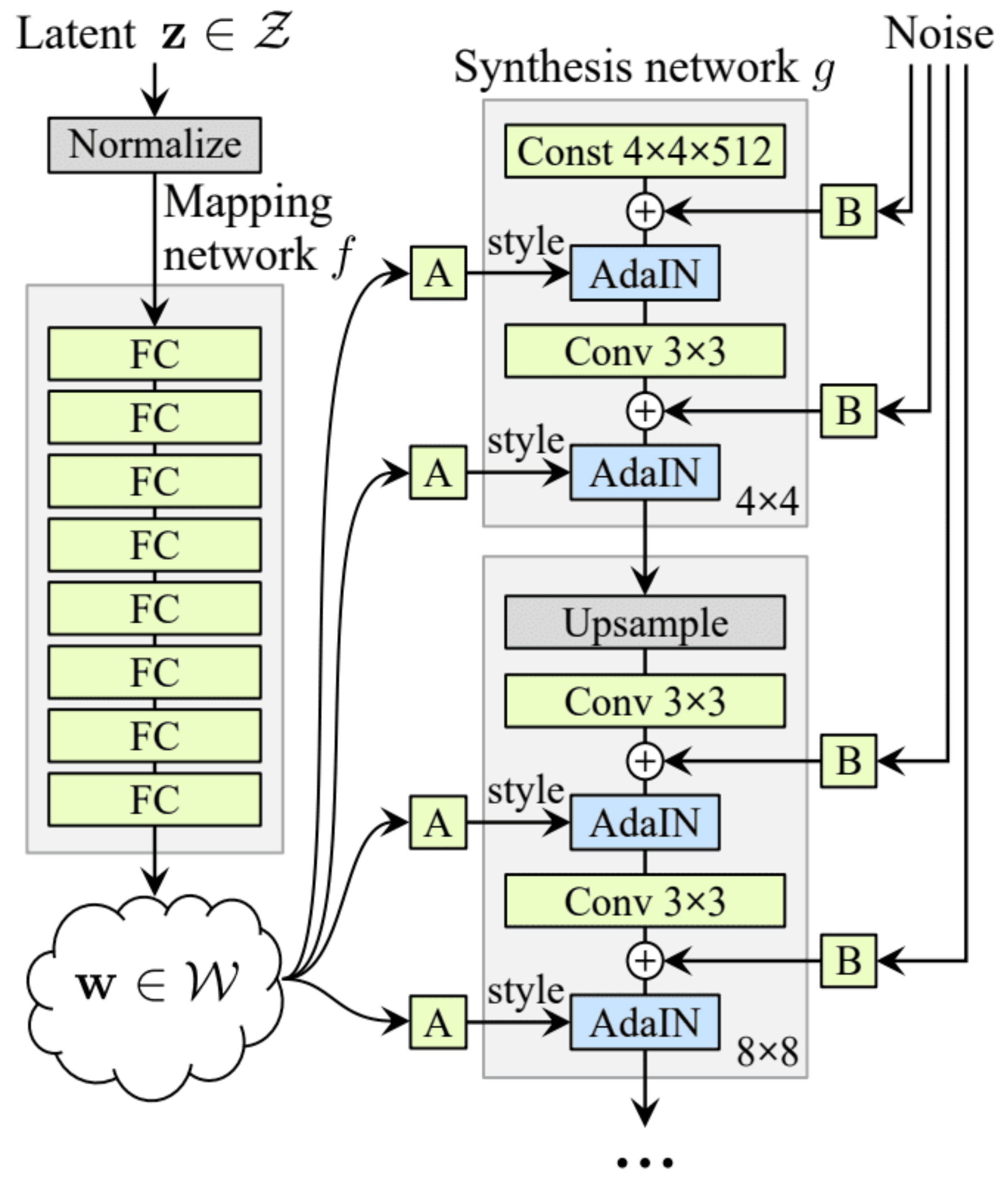


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

StyleGAN



Goal: Better disentanglement of features in latent space (W space)

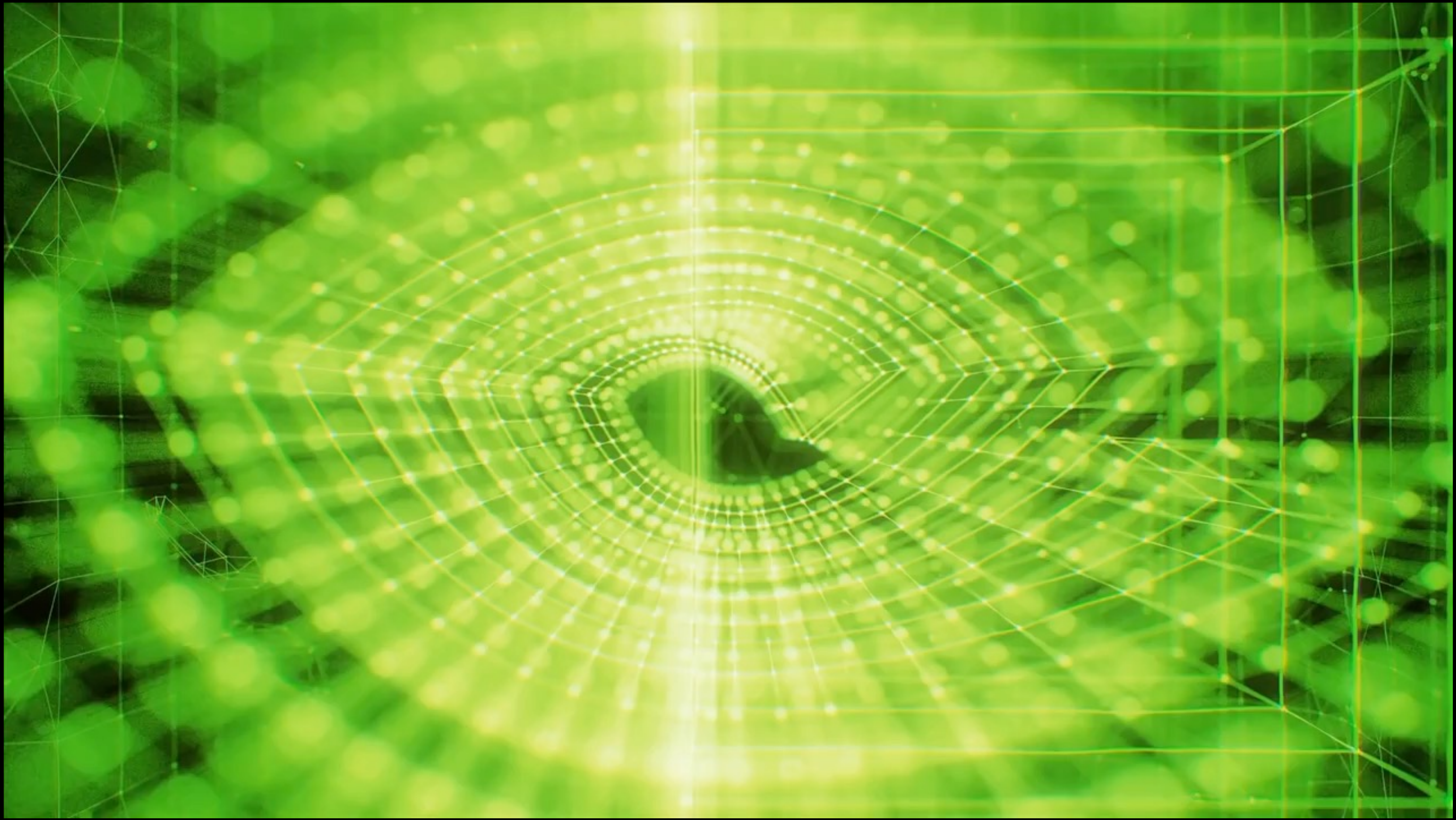


A = learned affine transformation block for AdaIN (predicts y)

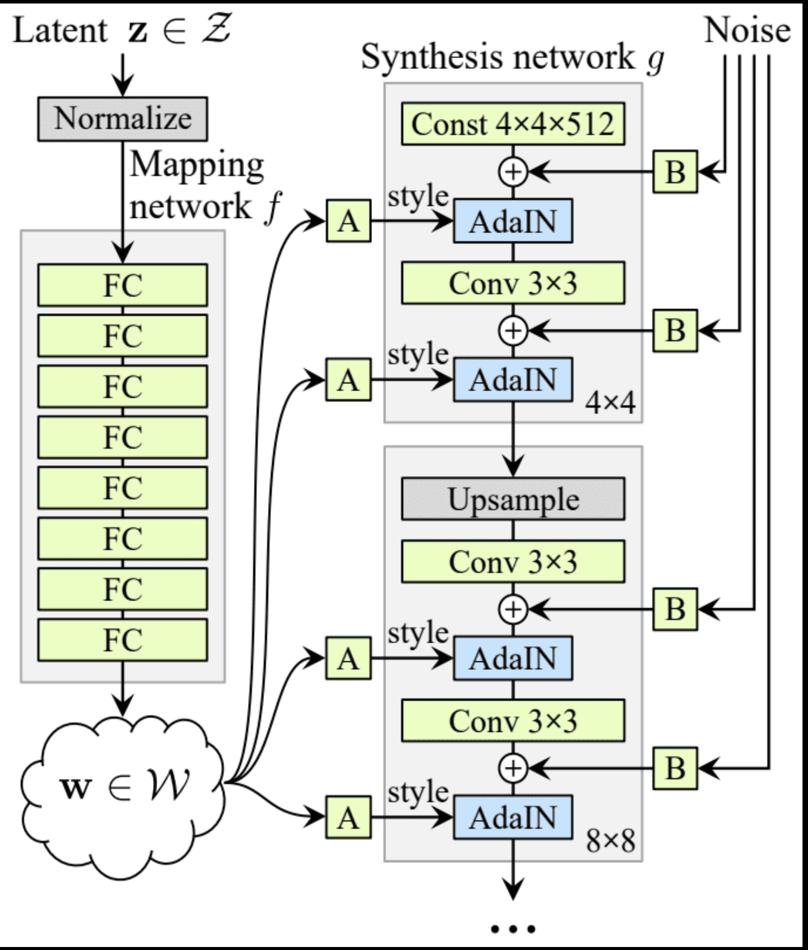
Adaptive Instance Normalization
(very effective in controlling styles)

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

B = learned per-channel scaling factor for noise input.



Which latent space to choose for embedding and editing?



- Z : 512 dimensional latent space (not good)
- W : 512 dimensional latent space (better but not perfect)
- W+ : 18x512 dimensional latent space (after affine transformation A has been applied)

- W is better for editing.
- W+ is better for reconstruction or embedding of real images.

Want to know more about embedding in W and W+ space?
Read: Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?

Conditional GAN

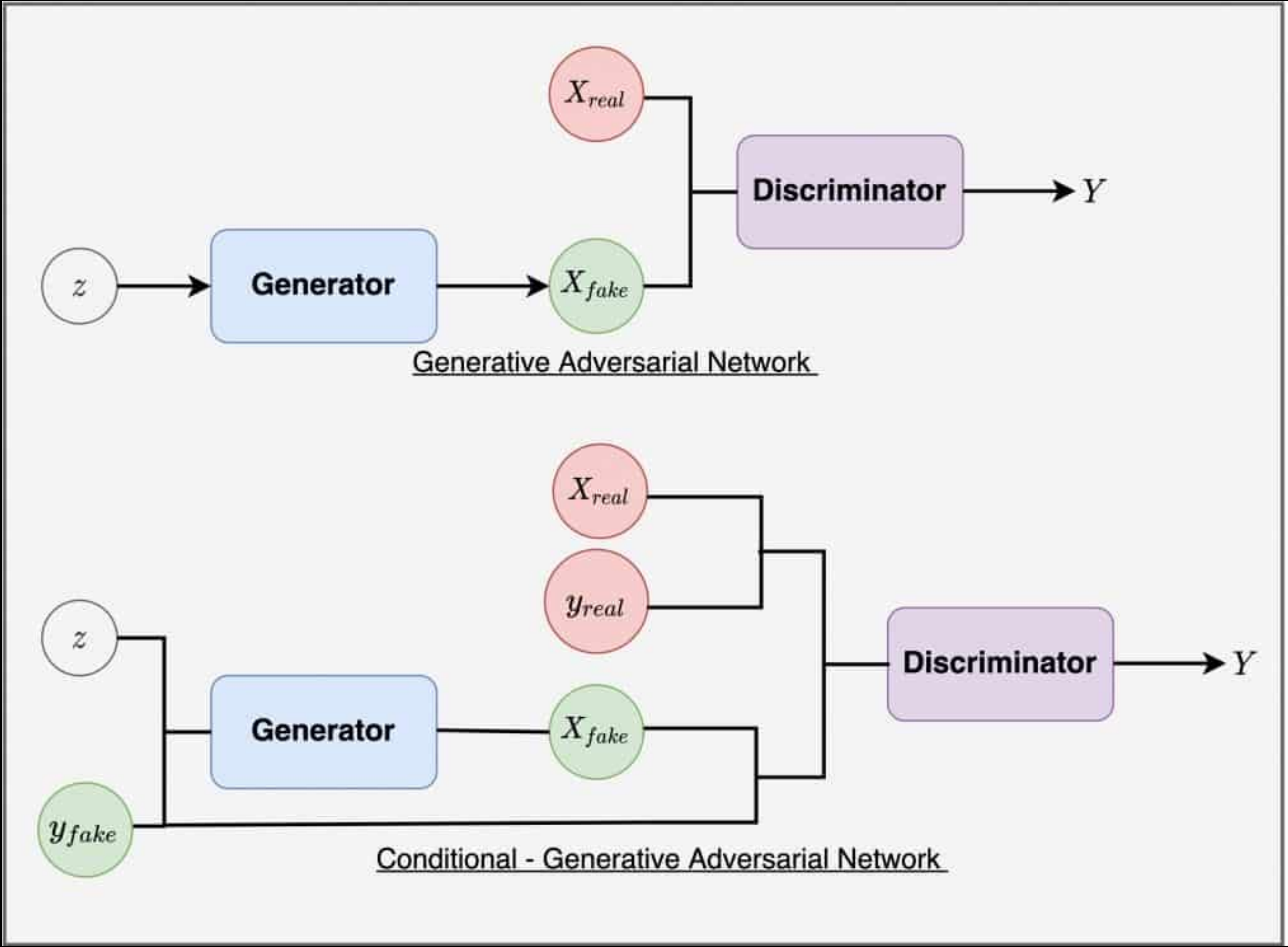
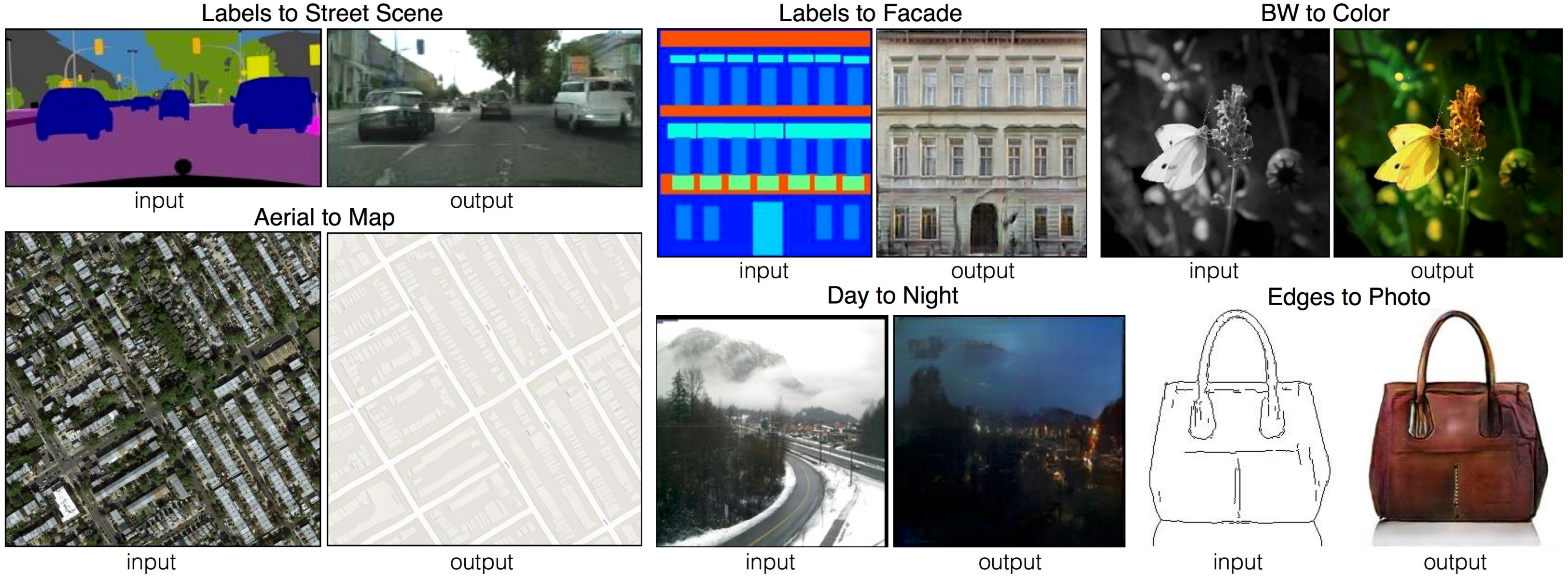

X_{real}
Discriminator
Y
z
Generator
X_{fake}
Generative Adversarial Network
X_{real}
y_{real}
z
Discriminator
Y
Generator
X_{fake}
y_{fake}
Conditional - Generative Adversarial Network

Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

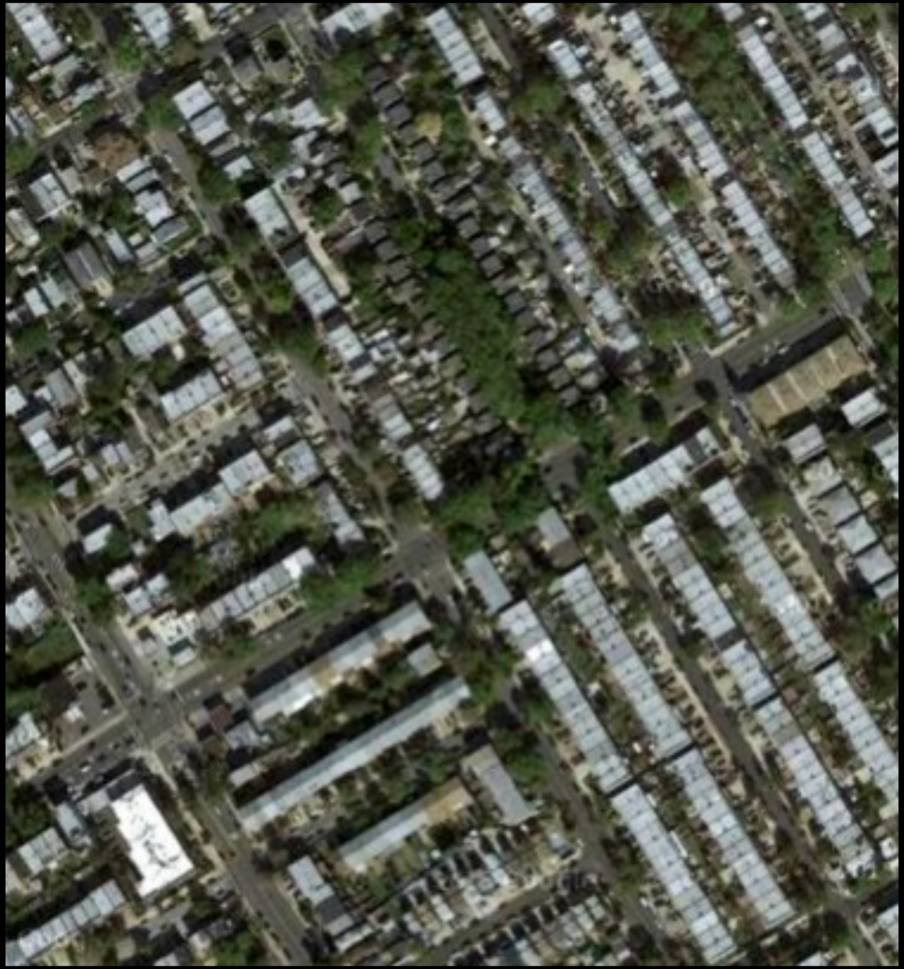
Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu



input: edges

output: image



input: satellite view

output: map

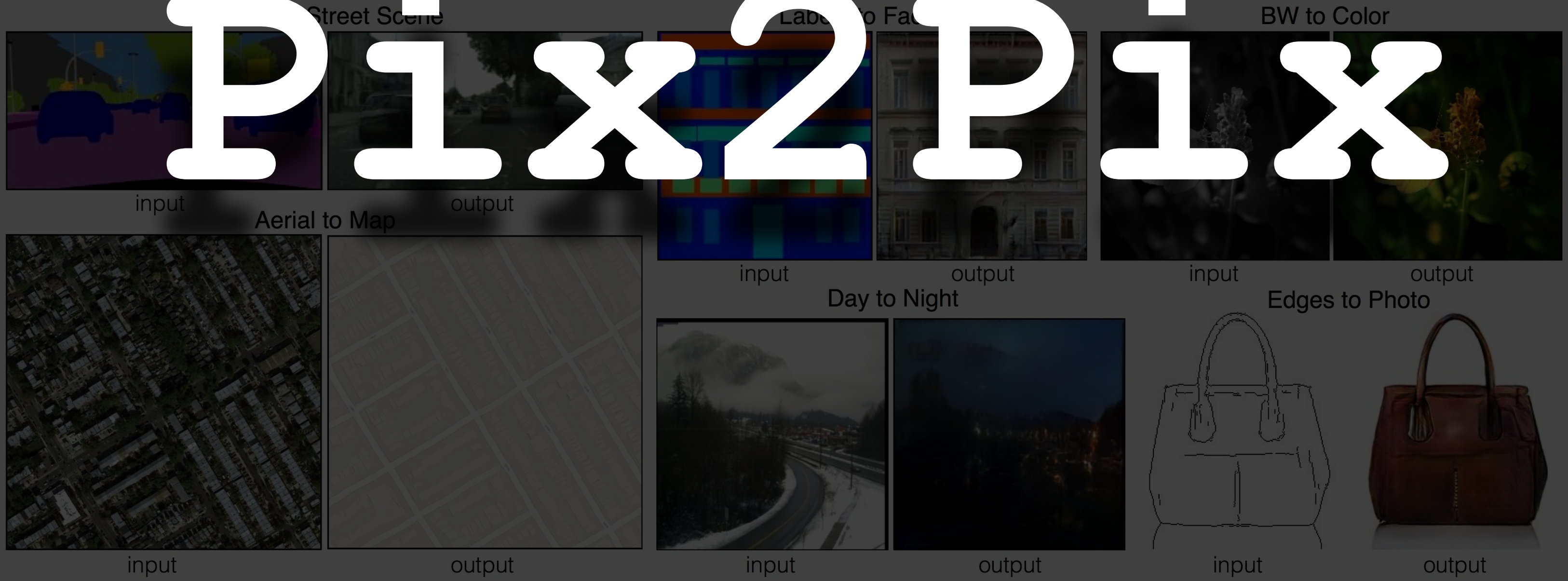
Pix2Pix

Image-to-Image Translation with Conditional Adversarial Networks

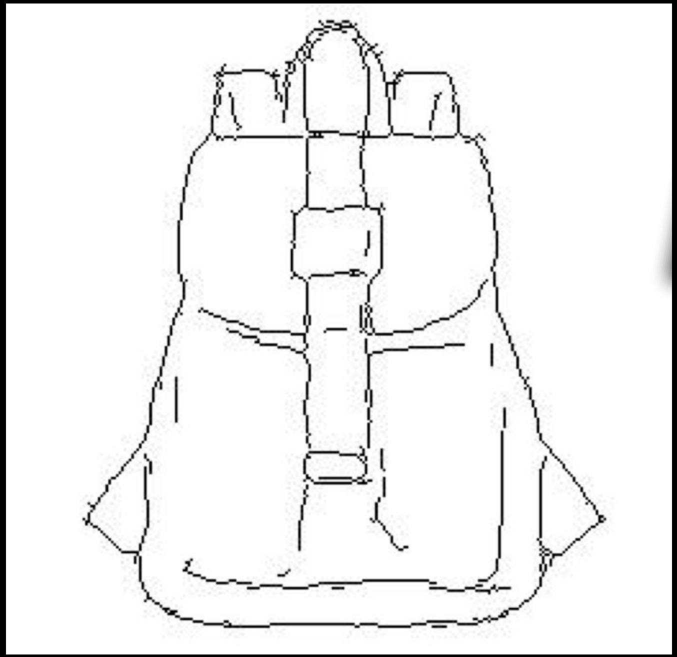
Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu



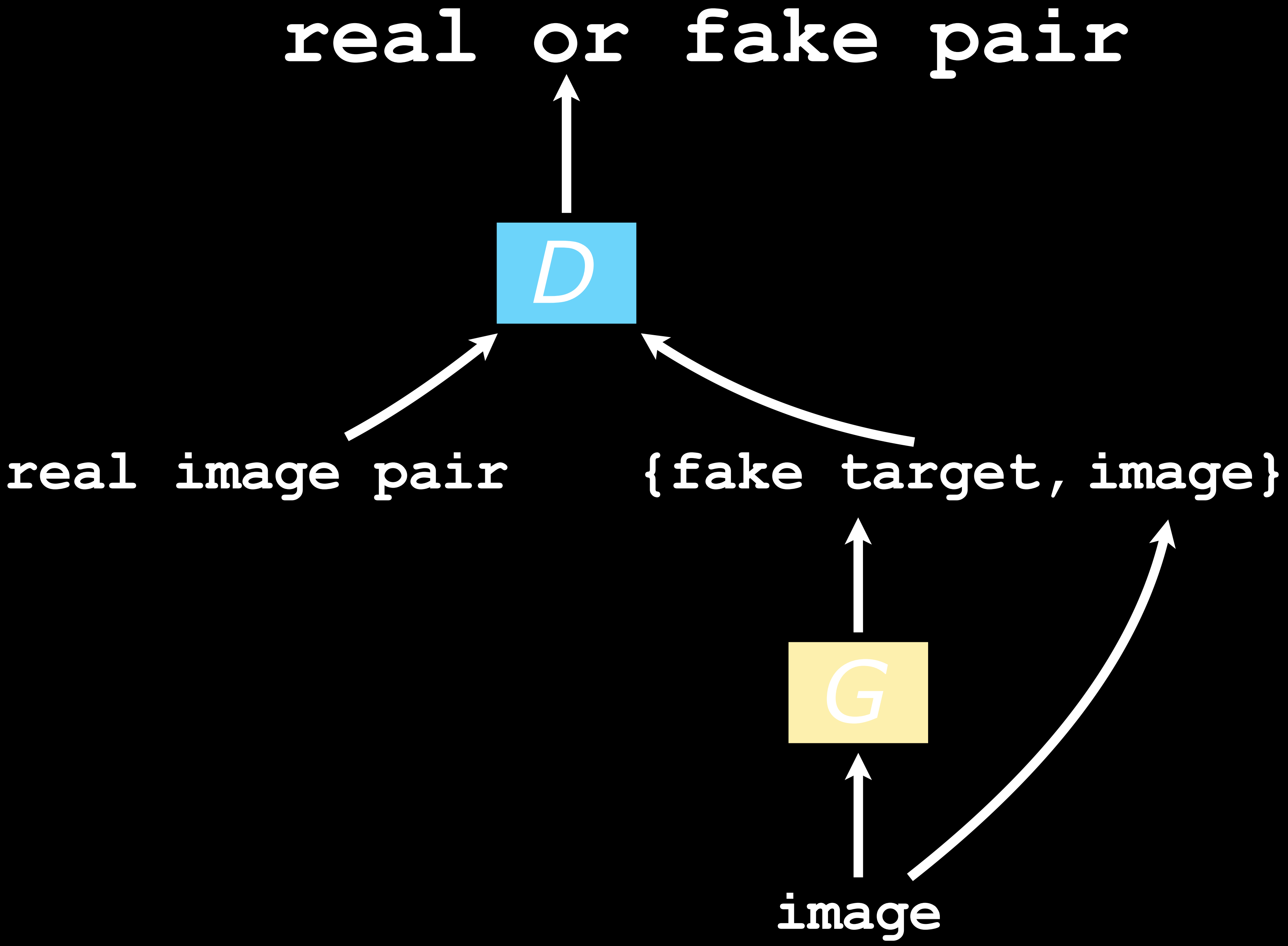
input: edges



output: image

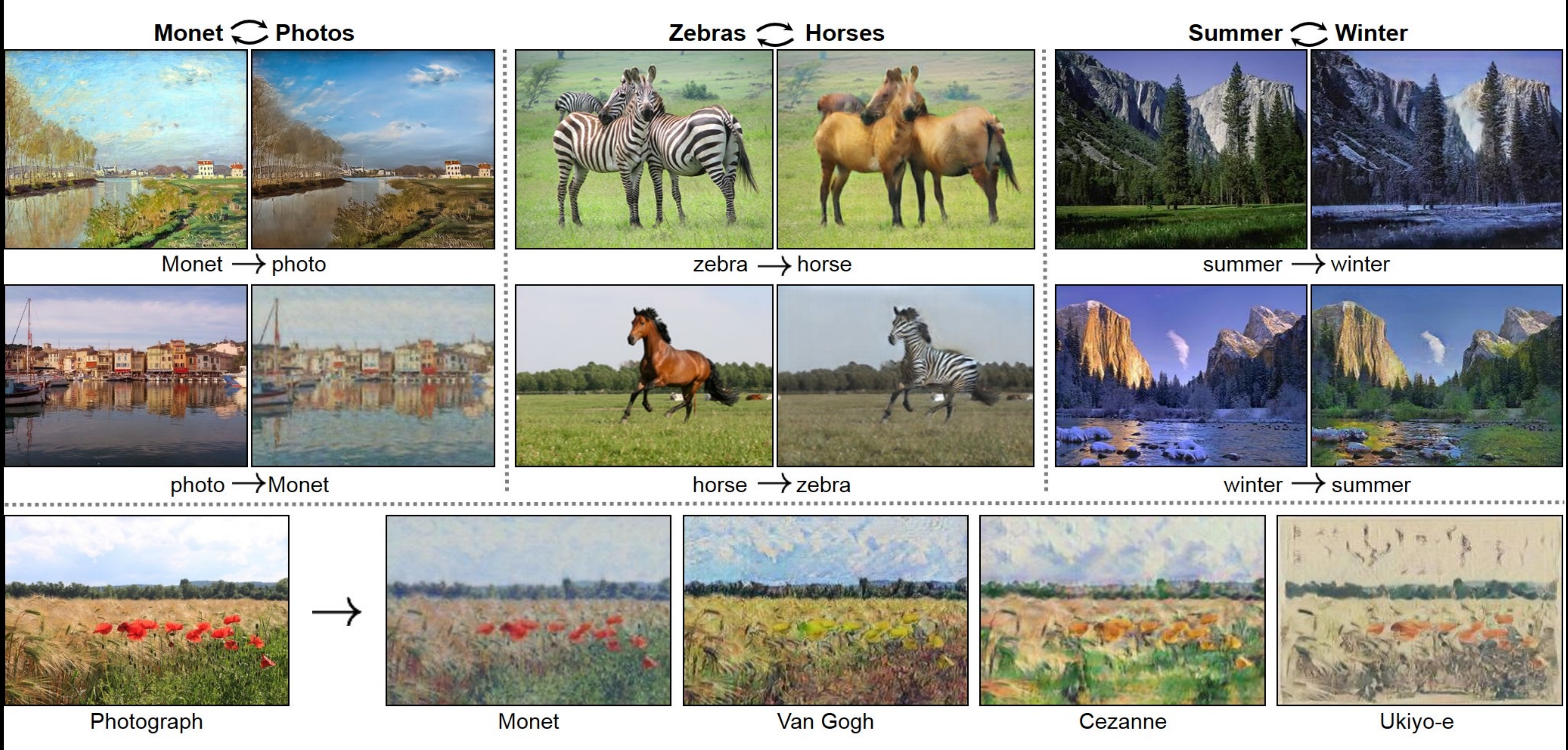
Training data consists of such pairs.

assumption



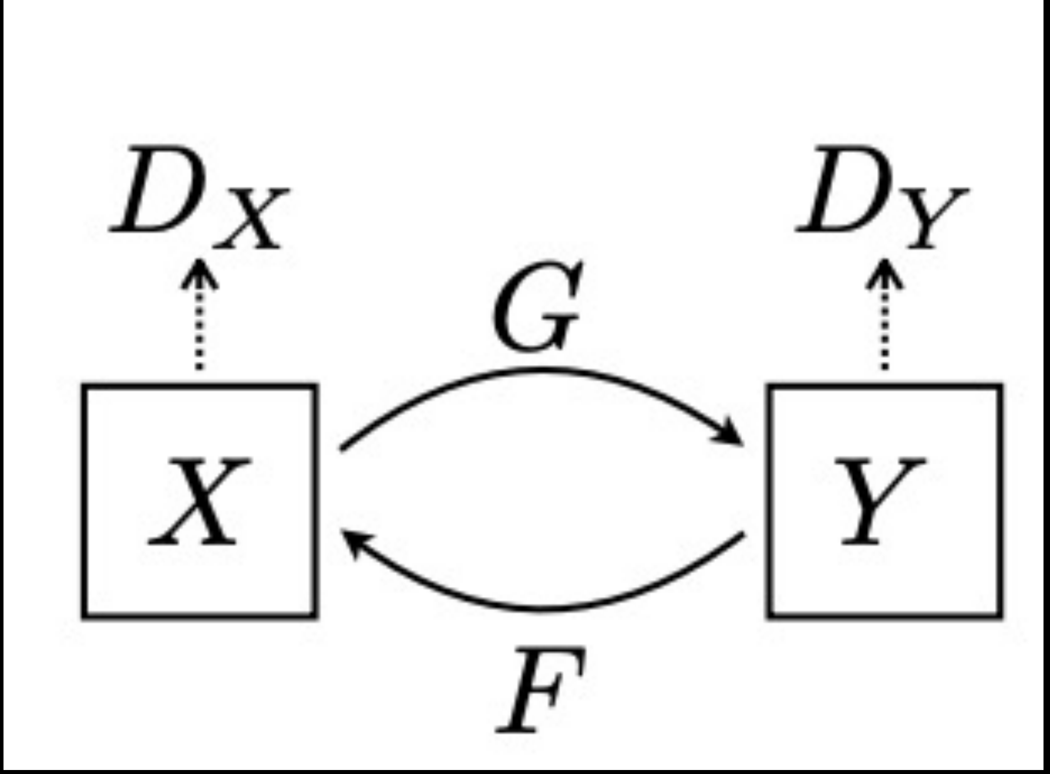
real or fake pair
D
real image pair
{fake target, image}
G
image

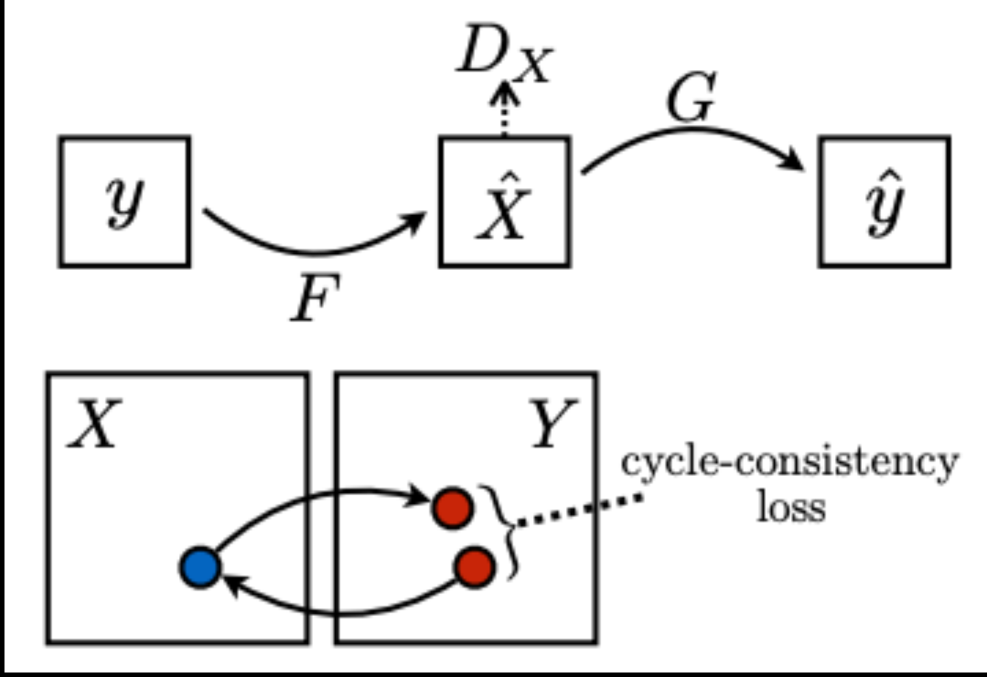
CycleGAN: Unpaired Image to Image Translation



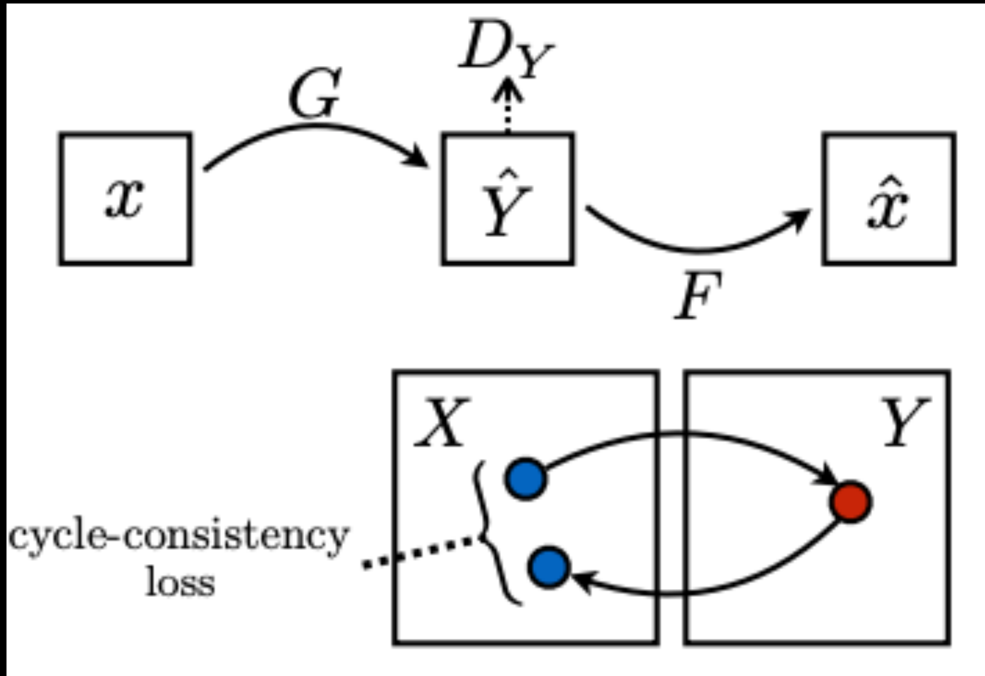
Training data: A set of images of style X + A set of images of style Y

Test: Given an image of style X, generate the same image in style Y

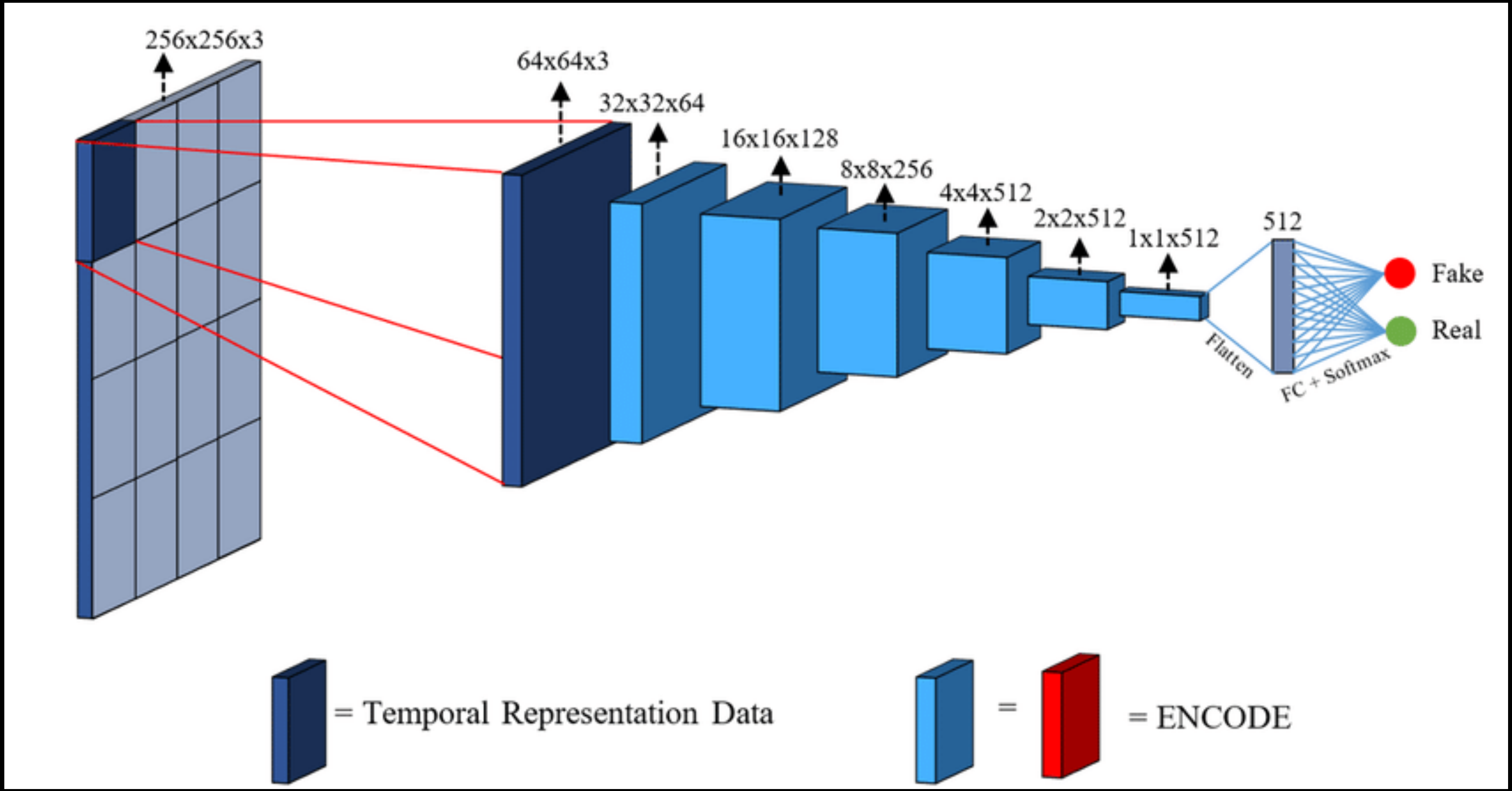




In addition to regular GAN loss on domain X and Y respectively, also add cycle-consistency loss for domain X and Y.



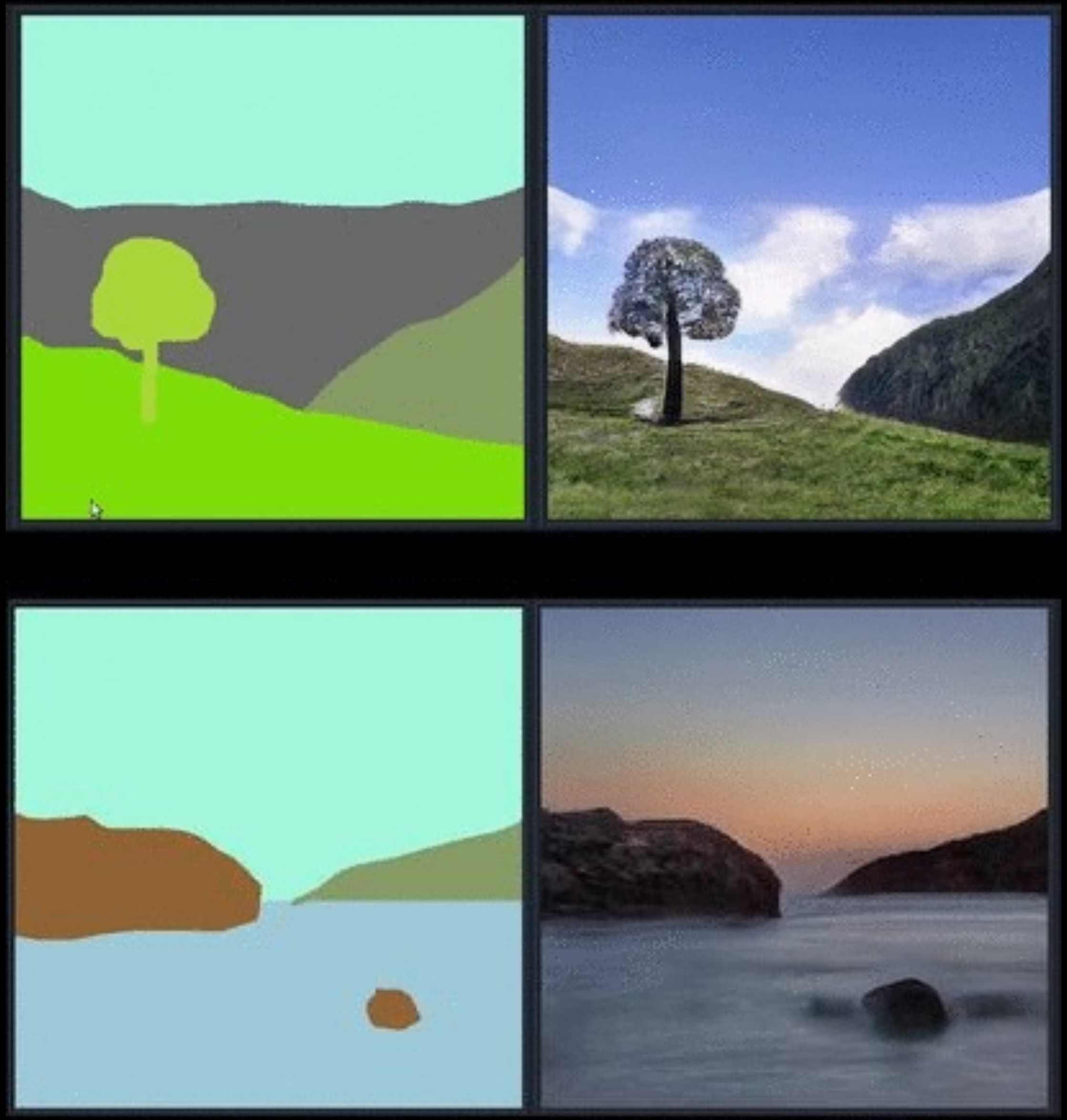
Patch Discriminator



In practice Patch discriminator is applied at multiple resolutions (e.g. 64x64, 128x128, 256x256), and often the patches are overlapping.

Figure courtesy: Unsupervised Anomaly Detection and Localization Based on Deep Spatiotemporal Translation Network

GauGAN: Semantic Image Synthesis with Spatially-Adaptive Normalization



Denoising Diffusion Models

Emerging as powerful generative models, outperforming GANs

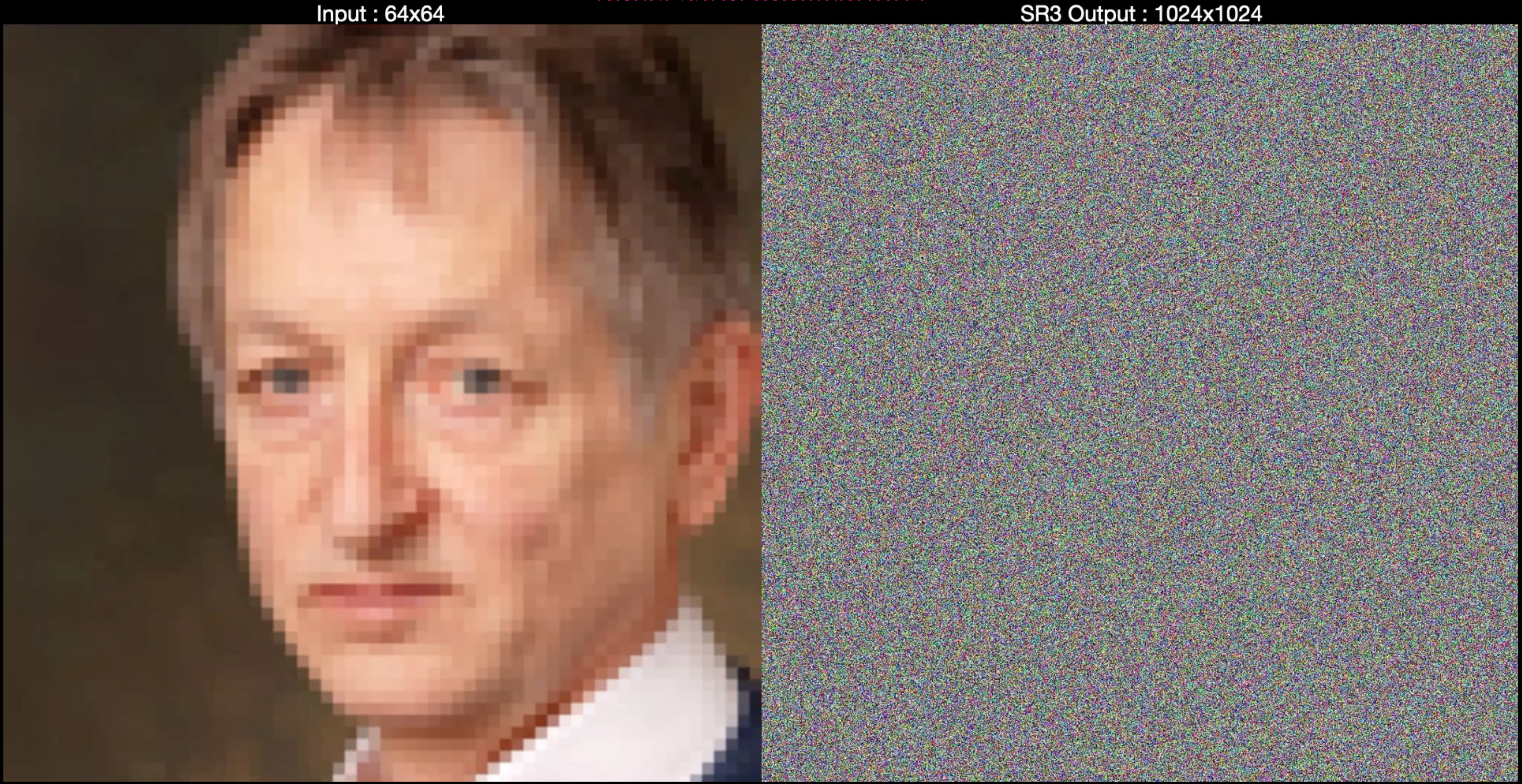
[“Diffusion Models Beat GANs on Image Synthesis”
Dhariwal & Nichol, OpenAI, 2021]

[“Cascaded Diffusion Models for High Fidelity Image Generation”
Ho et al., Google, 2021]

Diffusion Model

Image Super-resolution

Successful applications



Saharia et al., Image Super-Resolution via Iterative Refinement, ICCV 2021

Text-to-Image Generation

DALL·E 2

“a teddy bear on a skateboard in times square”

[“Hierarchical Text-Conditional Image Generation with CLIP Latents” Ramesh et al., 2022]

Imagen

A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.

[“Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, Saharia et al., 2022]

Text-to-Image Generation

Stable Diffusion

Stable Diffusion Applications: Twitter Mega Thread

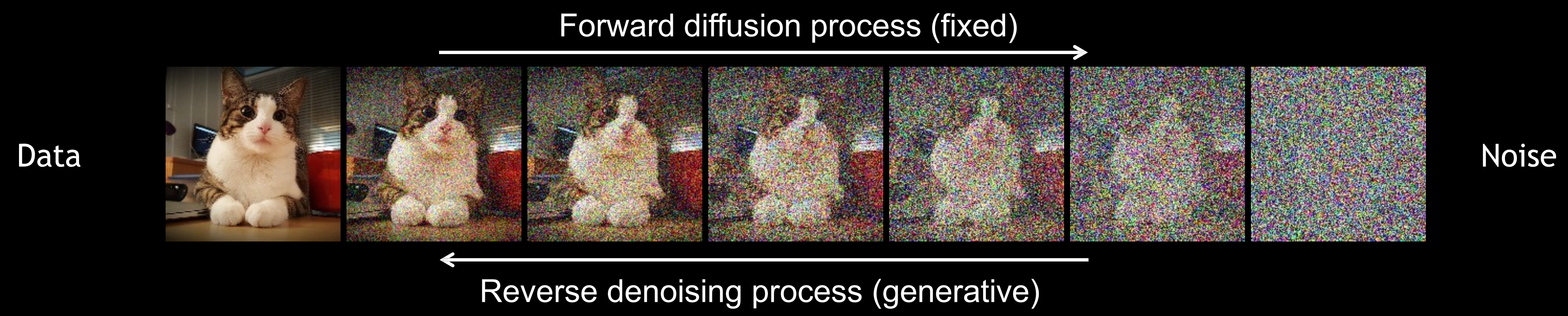
“High-Resolution Image Synthesis with Latent Diffusion Models” Rombach et al., 2022

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

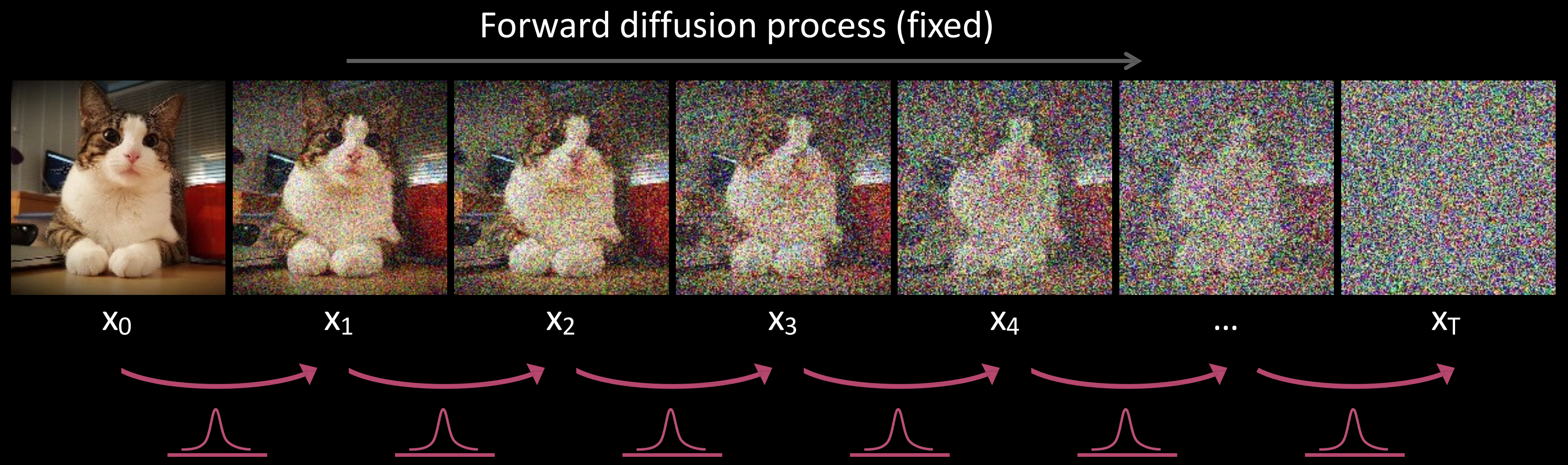


[Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015]
[Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020]
[Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021]

Forward Diffusion Process

The formal definition of the forward process in T steps:

Data



Noise

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \underbrace{\sqrt{1-\beta_t}\mathbf{x}_{t-1}}_{\text{mean}}, \underbrace{\beta_t\mathbf{I}}_{\text{variance}})$$

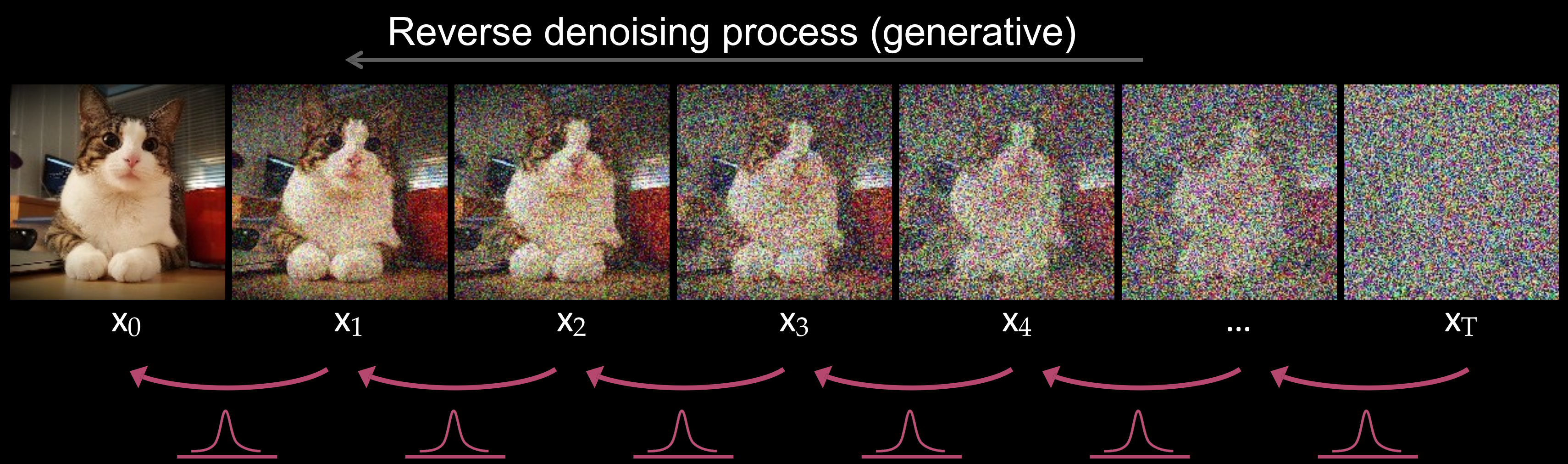
Sample: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}$

where, $\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:

Data



Noise

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\substack{\text{Trainable network} \\ \text{(U-net, Denoising Autoencoder)}}}, \sigma_t^2\mathbf{I})$$

Autoencoder predicts the mean of the denoised image x(t-1) given x(t).

How do we train? (summary version)

What is the loss function? (Ho et al. NeurIPS 2020)

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\mathcal{U}(1,T)}\left[\left|\left|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)\right|\right|^2\right]$$

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1,\ldots,T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5: $\quad$ Take gradient descent step on
$$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boxed{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}, t)\right\|^2$$
6: **until** converged

U-Net autoencoder takes x(t) as input and simply predict a noise. The goal of the training is to generate a noise pattern that is unit normal.

Summary

Training and Sample Generation

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boxed{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}, t\right) \right\|^2$$

6: **until** converged

Algorithm 2 Sampling

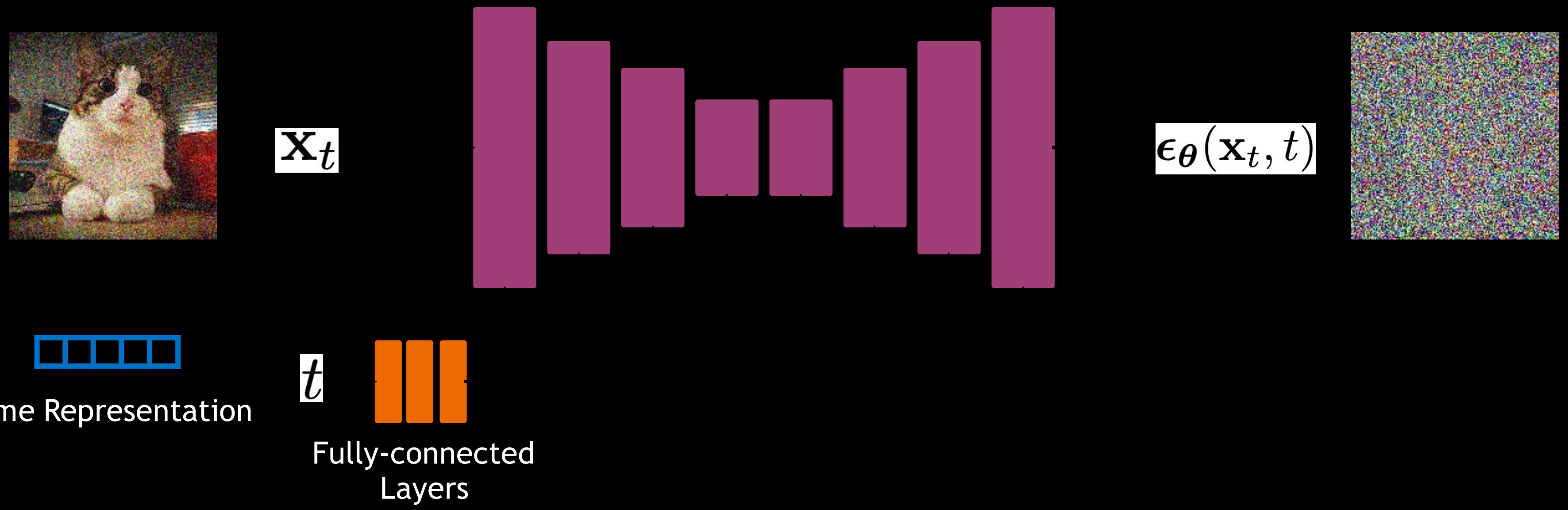
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\quad \mathbf{x}_{t-1} = \boxed{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)} + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Intuitively: During forward process we add noise to image. During reverse process we try to predict that noise with a U-Net and then subtract it from the image to denoise it.

Implementation Considerations

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

Content-Detail Tradeoff

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[|| \epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t) ||^2 \right]$$

Reverse denoising process (generative)

Data

x_0 x_1 x_2 x_3 x_4 ... x_T

Noise

The denoising model is specialized for generating the high-frequency content (i.e., low-level details)

The denoising model is specialized for generating the low-frequency content (i.e., coarse content)

The weighting of the training objective for different timesteps is important!

DALL·E 2

OpenAI

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

1kx1k Text-to-image generation.

Outperform DALL-E (autoregressive transformer).

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

CLIP guidance

What is a CLIP model?

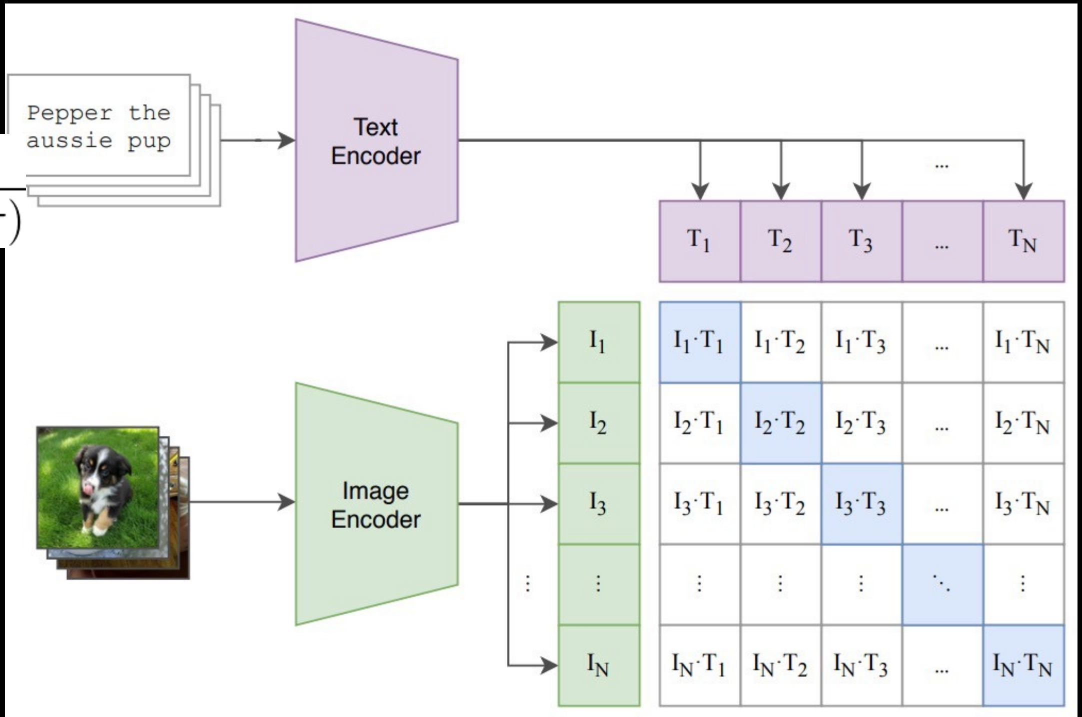
Text and image embedding for the same pair should be similar and for different pairs should be dissimilar.

- Trained by contrastive cross-entropy loss:

$$-\log \frac{\exp(f(\mathbf{x}_i) \cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_i) \cdot g(\mathbf{c}_k)/\tau)} - \log \frac{\exp(f(\mathbf{x}_i) \cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_k) \cdot g(\mathbf{c}_j)/\tau)}$$

- The optimal value of $f(\mathbf{x}) \cdot g(\mathbf{c})$ is

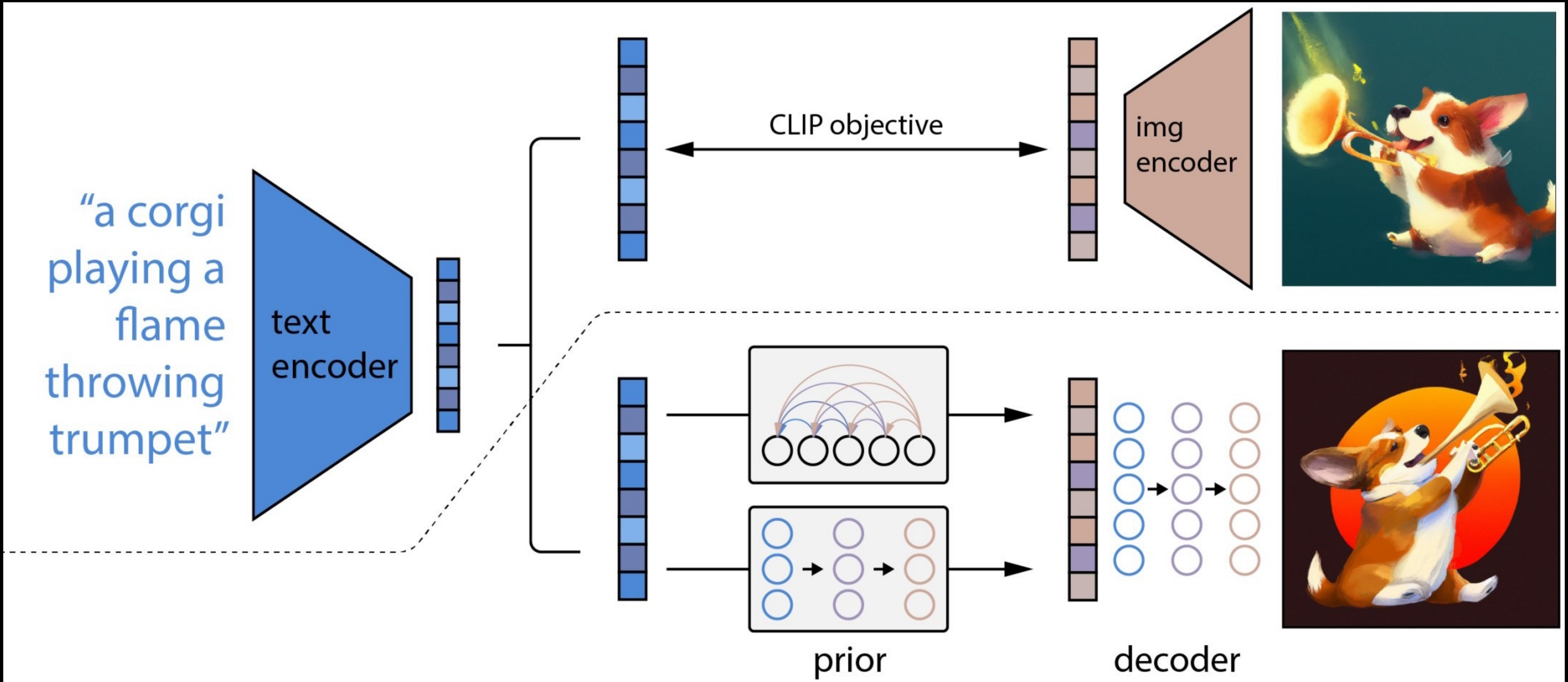
$$\log \frac{p(\mathbf{x}, \mathbf{c})}{p(\mathbf{x})p(\mathbf{c})} = \log p(\mathbf{c}|\mathbf{x}) - \log p(\mathbf{c})$$



Radford et al., "Learning Transferable Visual Models From Natural Language Supervision", 2021.
Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

DALL·E 2

Model components



Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

Slide Credits

- EECS 6322 Deep Learning for Computer Vision, Kosta Derpanis (York University)
- Diffusion Model Tutorial CVPR 2022

  https://cvpr2022-tutorial-diffusion-models.github.io/
- Many amazing research papers!