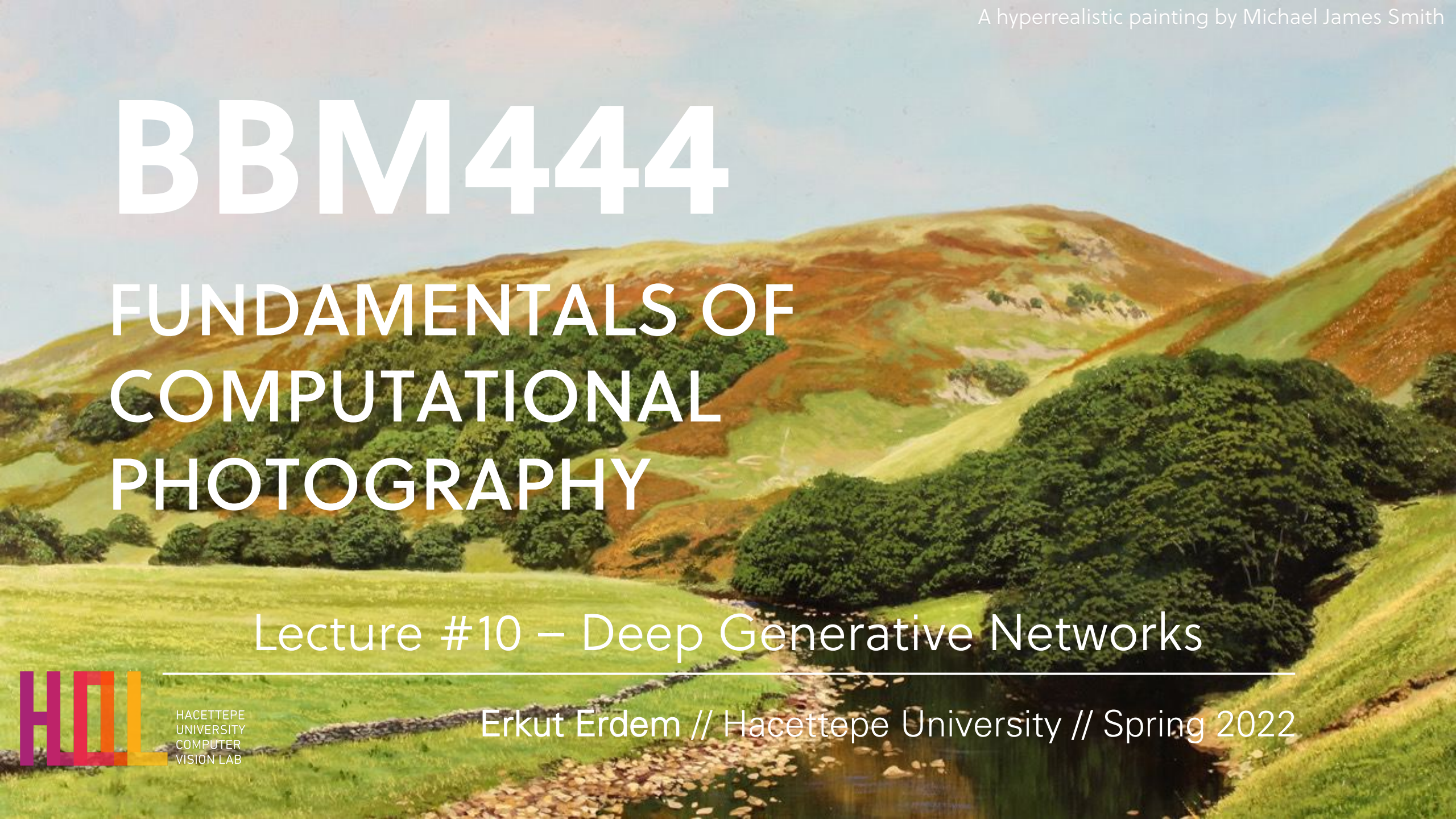


BBM444

FUNDAMENTALS OF COMPUTATIONAL PHOTOGRAPHY

Lecture #10 – Deep Generative Networks

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Spring 2022

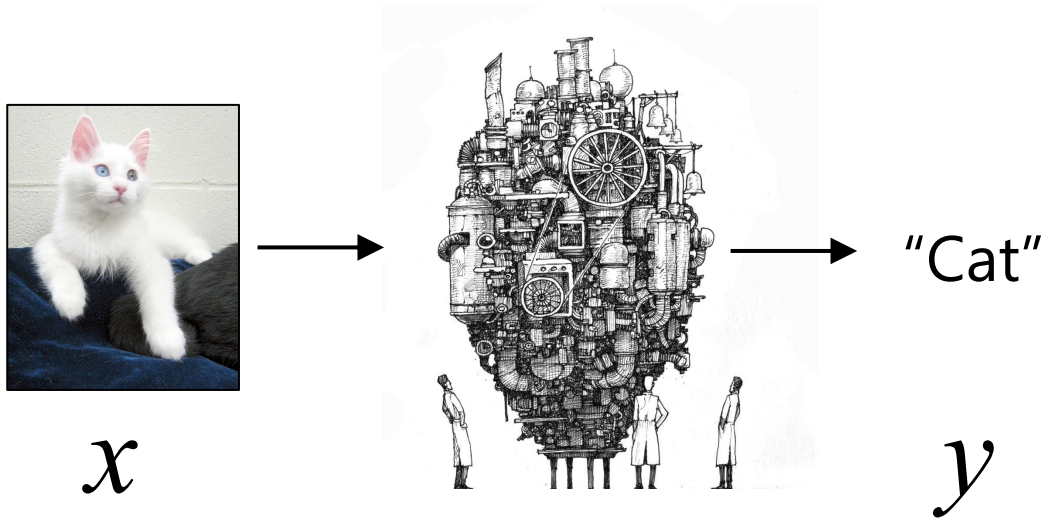
Today's Lecture

- Discriminative vs. generative models
- Unsupervised learning
  - Basic building blocks – sparse coding and autoencoders
  - Autoregressive models
  - Variational autoencoders
  - Generative adversarial networks
- Applications in computational photography

Discriminative vs. generative models

Discriminative vs. Generative Models

Discriminative models

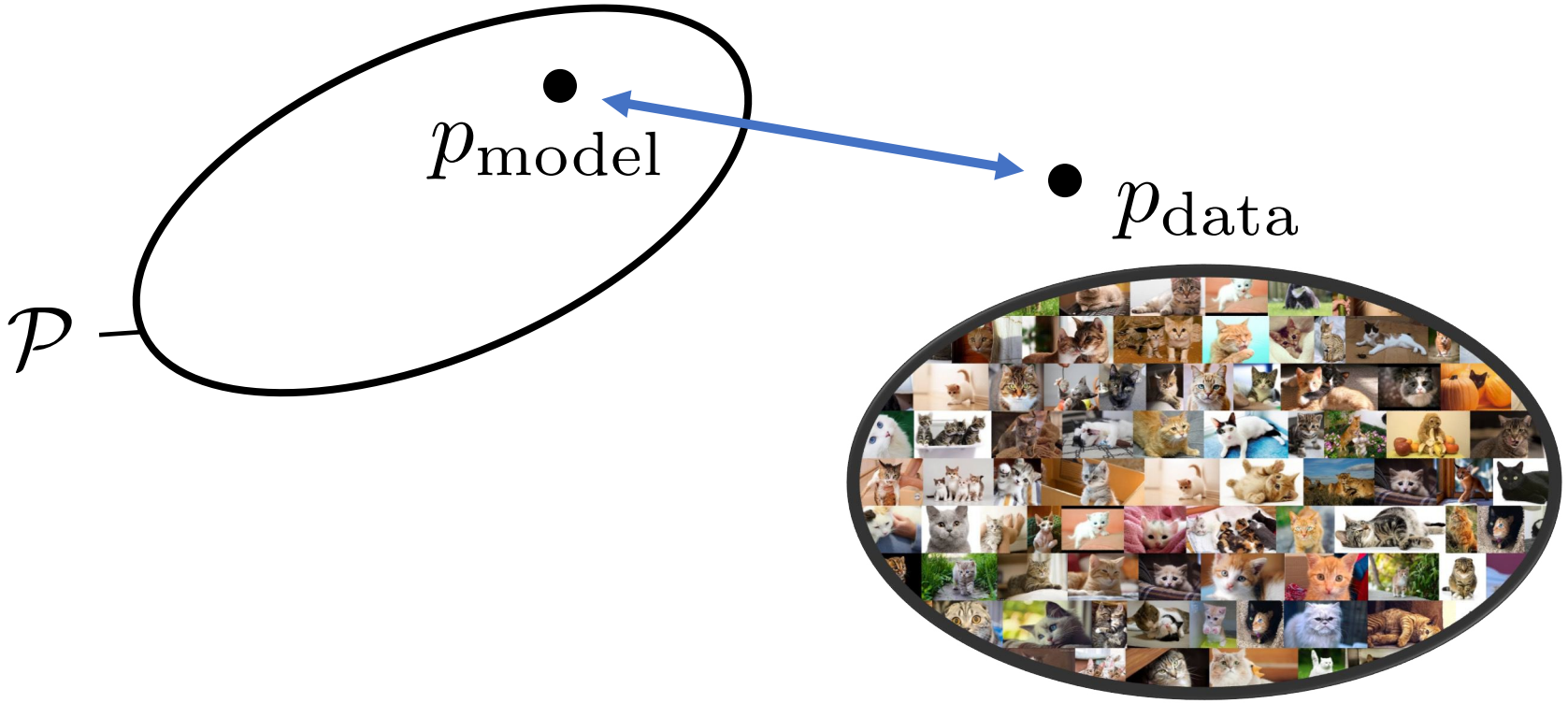


Goal: Learn a function to map x -> y

Generative models

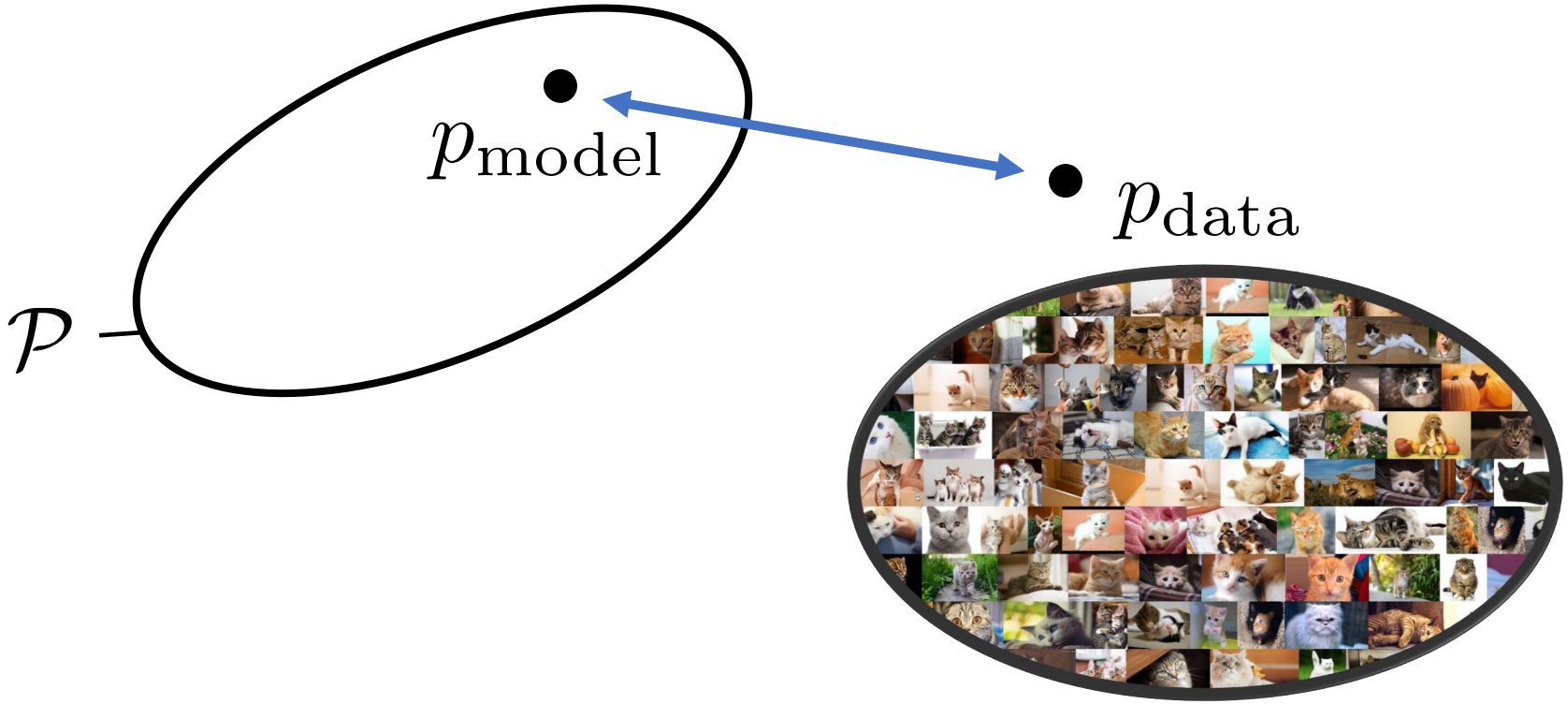
Goal: Learn some underlying hidden structure of the training samples to generate novel samples from same data distribution

Generative Modeling



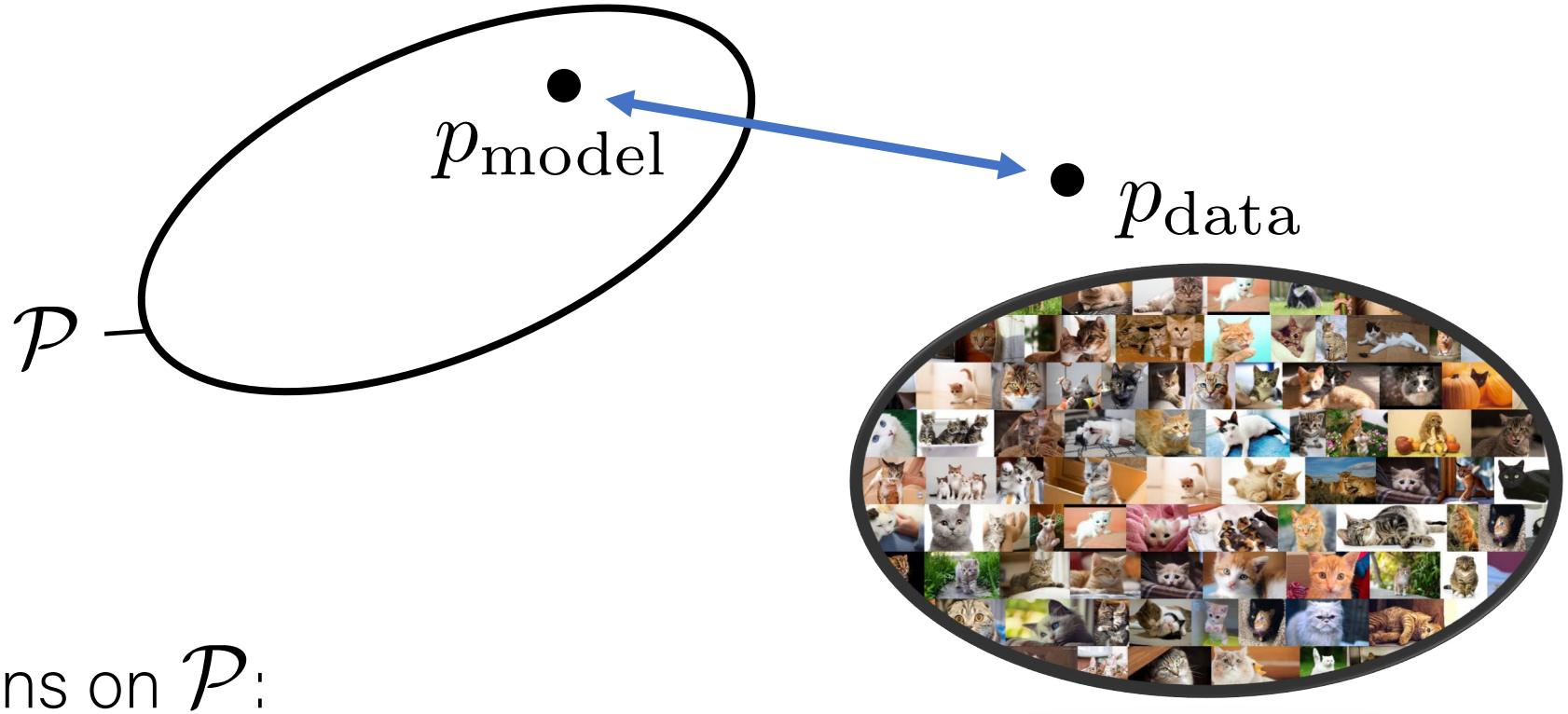
Slide adapted from Sebastian Nowozin

Generative Modeling



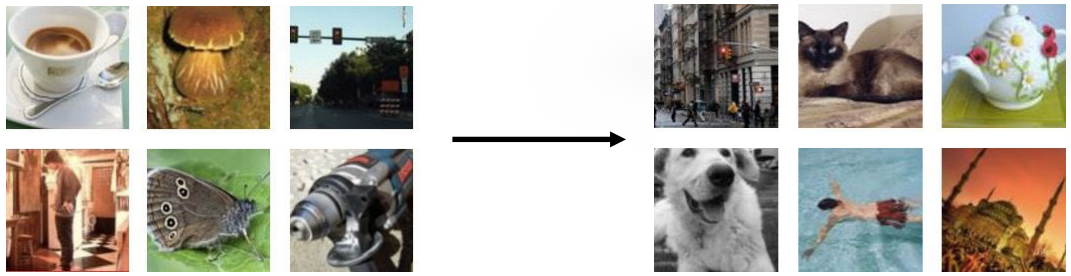
- **Goal:** Learn some underlying hidden structure of the training samples to generate novel samples from same data distribution

Generative Modeling



Assumptions on $\mathcal{P}$:

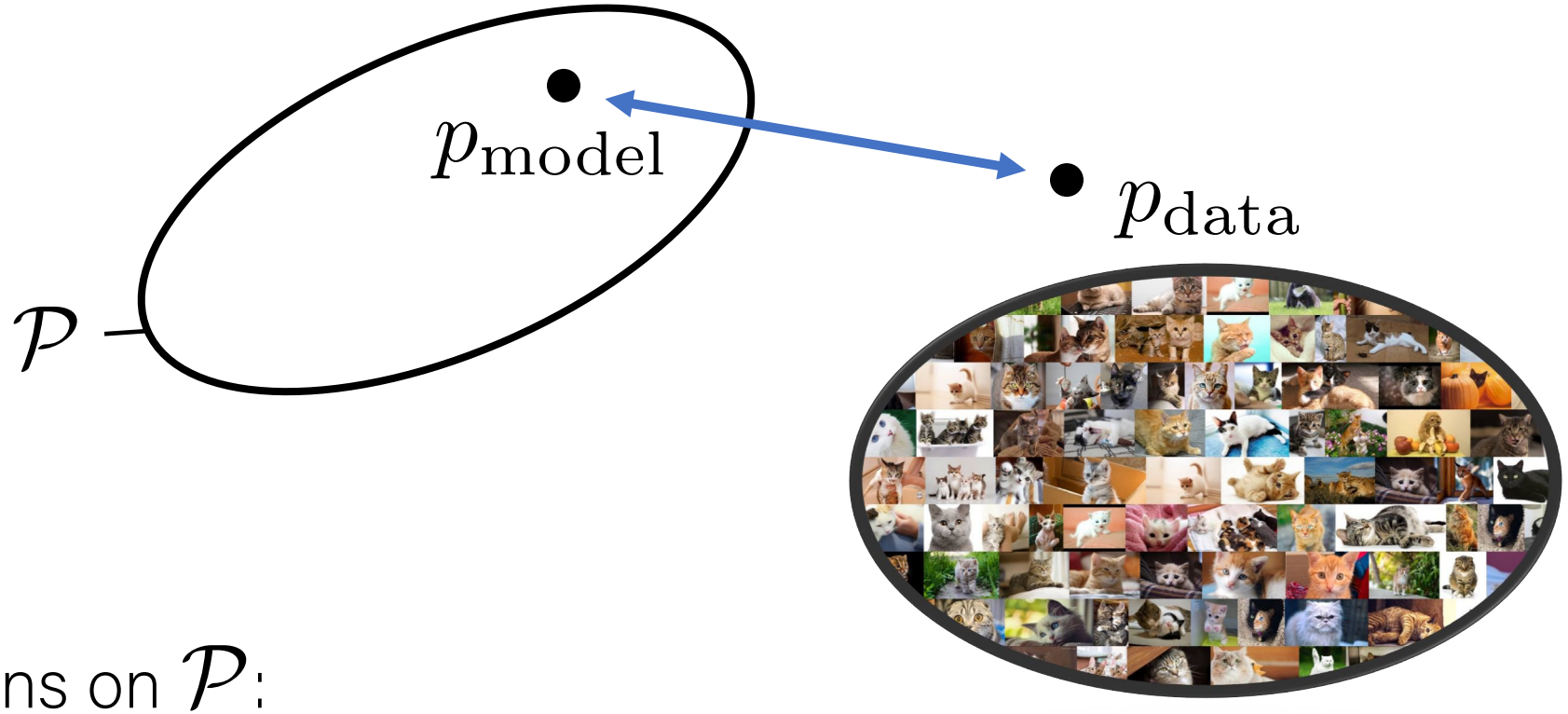
- tractable sampling



Training examples Model samples

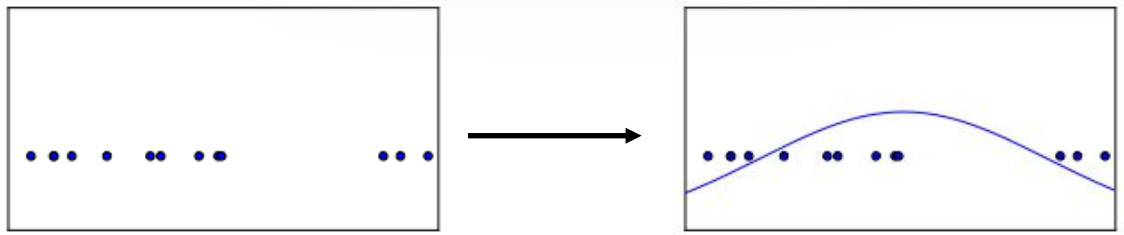
Slide adapted from Sebastian Nowozin

Generative Modeling



Assumptions on $\mathcal{P}$:

- tractable sampling
- tractable likelihood function

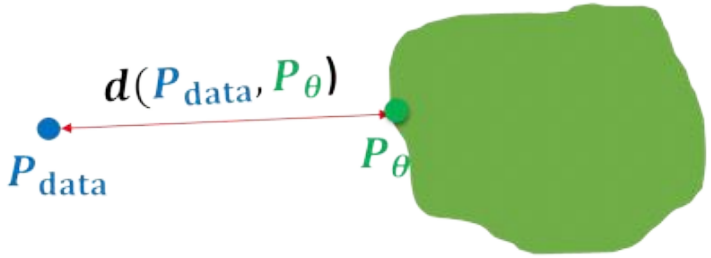


Slide adapted from Sebastian Nowozin

Learning a generative model

- We are given a training set of examples, e.g., images of dogs

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



Model family

- We want to learn a probability distribution p(x) over images x s.t.
  - **Generation**: If we sample $x_{new} \sim p(x)$, x_{new} should look like a dog (sampling)
  - **Density estimation:** p(x) should be high if x looks like a dog, and low otherwise (anomaly detection)
  - **Unsupervised representation learning**: We should be able to learn what these images have in common, e.g., ears, tail, etc. (features)

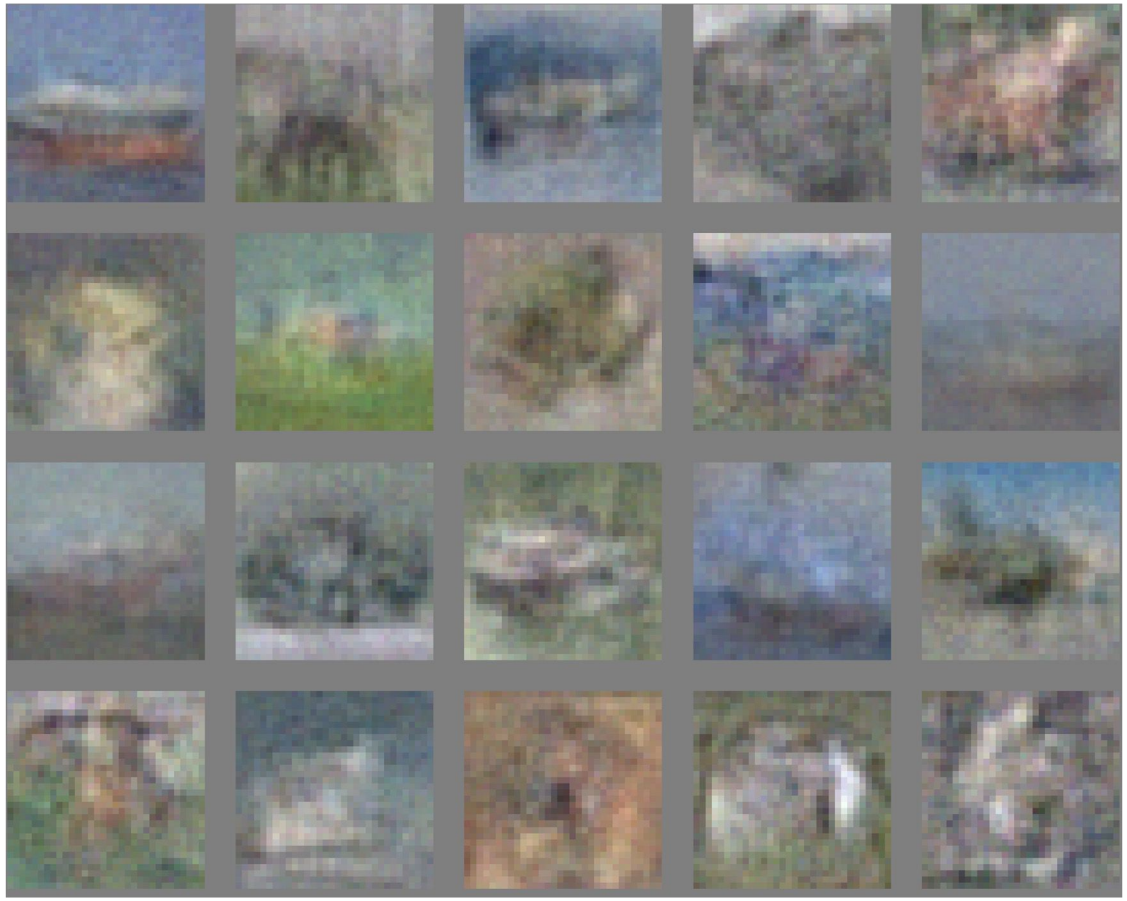
Generate Images

[Deep Belief Nets, Hinton, Osindero, Teh, 2006]

Generate Images

[VAE, Kingma and Welling, 2013]

Generate Images



[GAN, Goodfellow et al. 2014]

Generate Images

[DCGAN, Radford, Metz, Chintala 2015]

Generate Images

[DCGAN, Radford, Metz, Chintala 2015]

Generate Images

[CycleGAN: Zhu, Park, Isola & Efros, 2017]

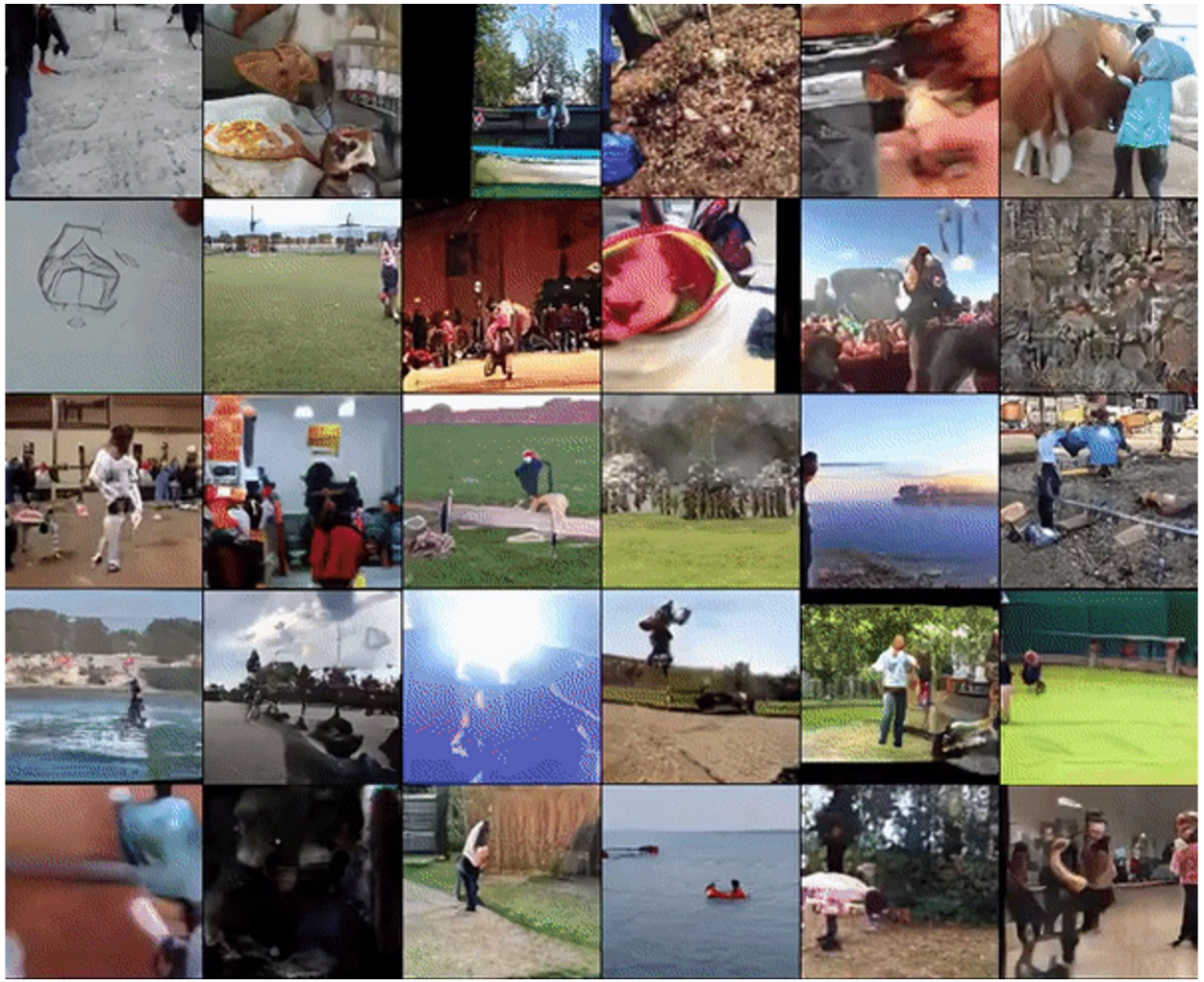
Generate Images

[BigGAN, Brock, Donahue, Simonyan, 2018]

Generate Images

[StyleGAN, Karras, Laine, Aila, 2018]

Generate Video



DVD-GAN: Adversarial Video Generation on Complex Datasets, Clark, Donahue, Simonyan, 2019

Why Unsupervised Learning?

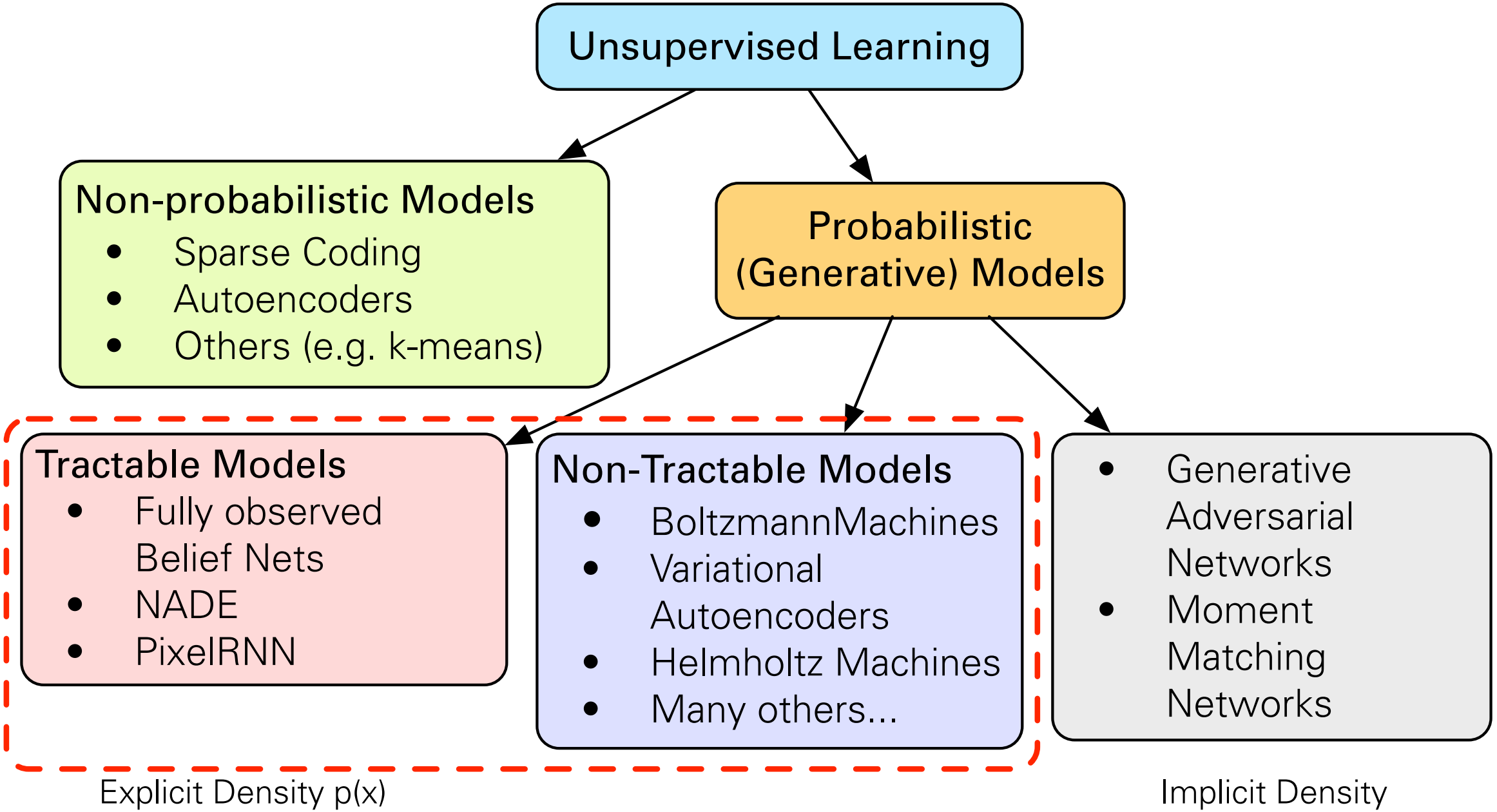
- Given high-dimensional data $X = (x_1, \ldots, x_n)$ we want to find a low-dimensional model characterizing the population.

- Recent progress mostly in supervised DL

- Real challenges for unsupervised DL

- Potential benefits:
  - **Exploit tons of unlabeled data**
  - Answer new questions about the variables observed
  - Regularizer – transfer learning – domain adaptation
  - Easier optimization (divide and conquer)
  - Joint (structured) outputs

Unsupervised Learning


Unsupervised Learning
Non-probabilistic Models
• Sparse Coding
• Autoencoders
• Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
• Fully observed Belief Nets
• NADE
• PixelRNN
Non-Tractable Models
• BoltzmannMachines
• Variational Autoencoders
• Helmholtz Machines
• Many others...
• Generative Adversarial Networks
• Moment Matching Networks
Explicit Density p(x)
Implicit Density

Basic Building Blocks

Sparse Coding

- Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).
- **Objective:** Given a set of input data vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, learn a dictionary of bases, such that:

$$\mathbf{x}_n = \sum_{k=1}^{K} a_{nk}\phi_k$$

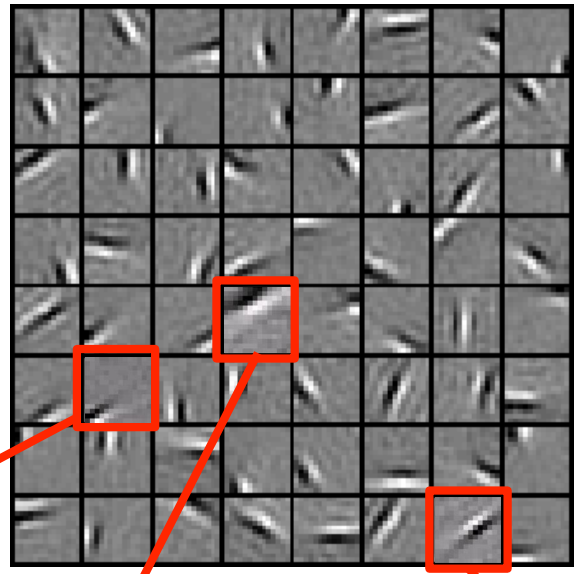
Sparse: mostly zeros

- Each data vector is represented as a sparse linear combination of bases.

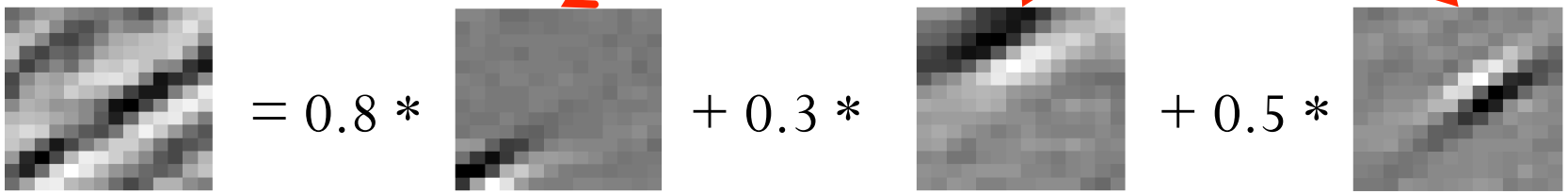
Sparse Coding

Natural Images

Learned bases: "Edges"



New example



$$= 0.8 * \quad + 0.3 * \quad + 0.5 *$$

$$x = 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65}$$

[0.0, 0.0, ... **0.8**, ..., **0.3**, ..., **0.5**, ...] = coefficients (feature representation)

Slide Credit: Honglak Lee

Sparse Coding: Training

- Input image patches: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \in \mathbb{R}^D$
- Learn dictionary of bases: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K \in \mathbb{R}^D$

$$\min_{\mathbf{a}, \boldsymbol{\phi}} \sum_{n=1}^{N} \underbrace{\left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk} \boldsymbol{\phi}_k \right\|_2^2}_{\text{Reconstruction error}} + \lambda \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} |a_{nk}|}_{\text{Sparsity penalty}}$$

- Alternating Optimization:
  1. Fix dictionary of bases and solve for activations **a** (a standard Lasso problem).
  2. Fix activations **a**, optimize the dictionary of bases (convex QP problem).

Sparse Coding: Testing Time

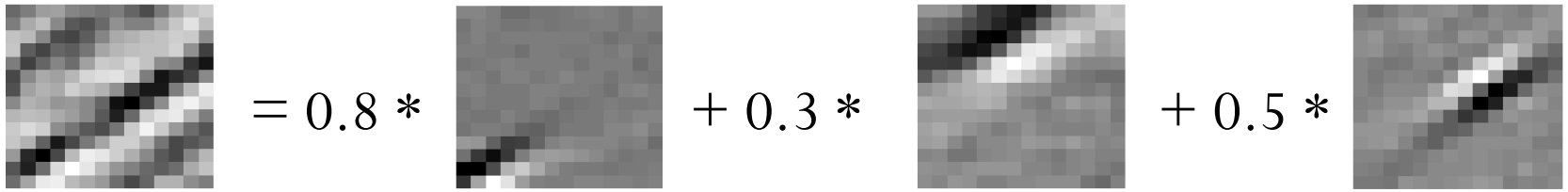
- **Input:** a new image patch x* , and K learned bases $\phi_1, \phi_2, ..., \phi_K$
- **Output:** sparse representation **a** of an image patch x*.

$$\min_{\mathbf{a}} \left|\left| \mathbf{x}^* - \sum_{k=1}^{K} a_k \boldsymbol{\phi}_k \right|\right|_2^2 + \lambda \sum_{k=1}^{K} |a_k|$$

Sparse Coding: Testing Time

- **Input:** a new image patch x* , and K learned bases $\phi_1, \phi_2, ..., \phi_K$
- **Output:** sparse representation **a** of an image patch x*.

$$\min_{\mathbf{a}} \left\| \mathbf{x}^* - \sum_{k=1}^{K} a_k \phi_k \right\|_2^2 + \lambda \sum_{k=1}^{K} |a_k|$$

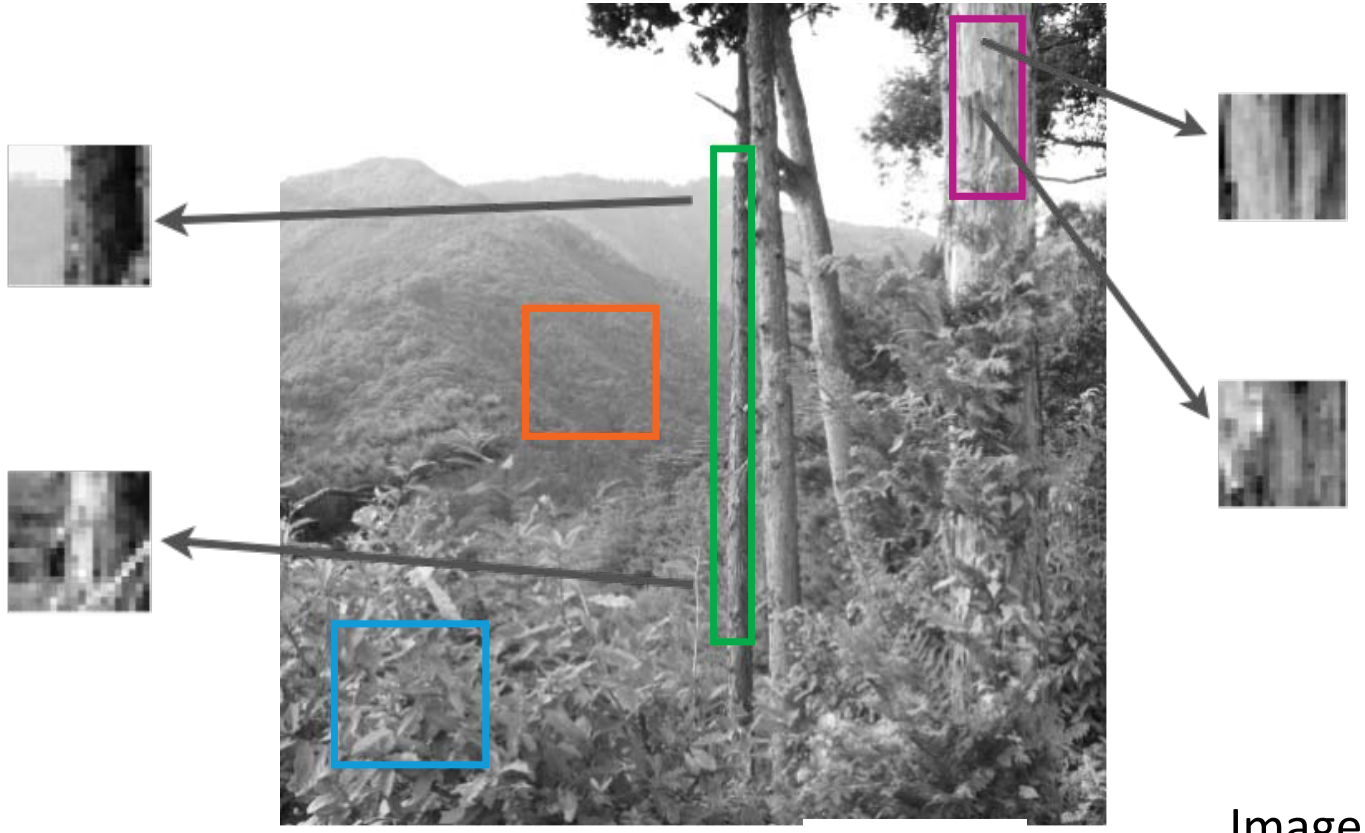


x^* $= 0.8 *$ ϕ_{36} $+ \ 0.3 *$ ϕ_{42} $+ 0.5 *$ ϕ_{65}

[0.0, 0.0, ... **0.8**, ..., **0.3**, ..., **0.5**, ...] = coefficients (feature representation)

Modeling Image Patches

- Natural image patches:
  - small **image regions** extracted from an image of nature (forest, grass, …)

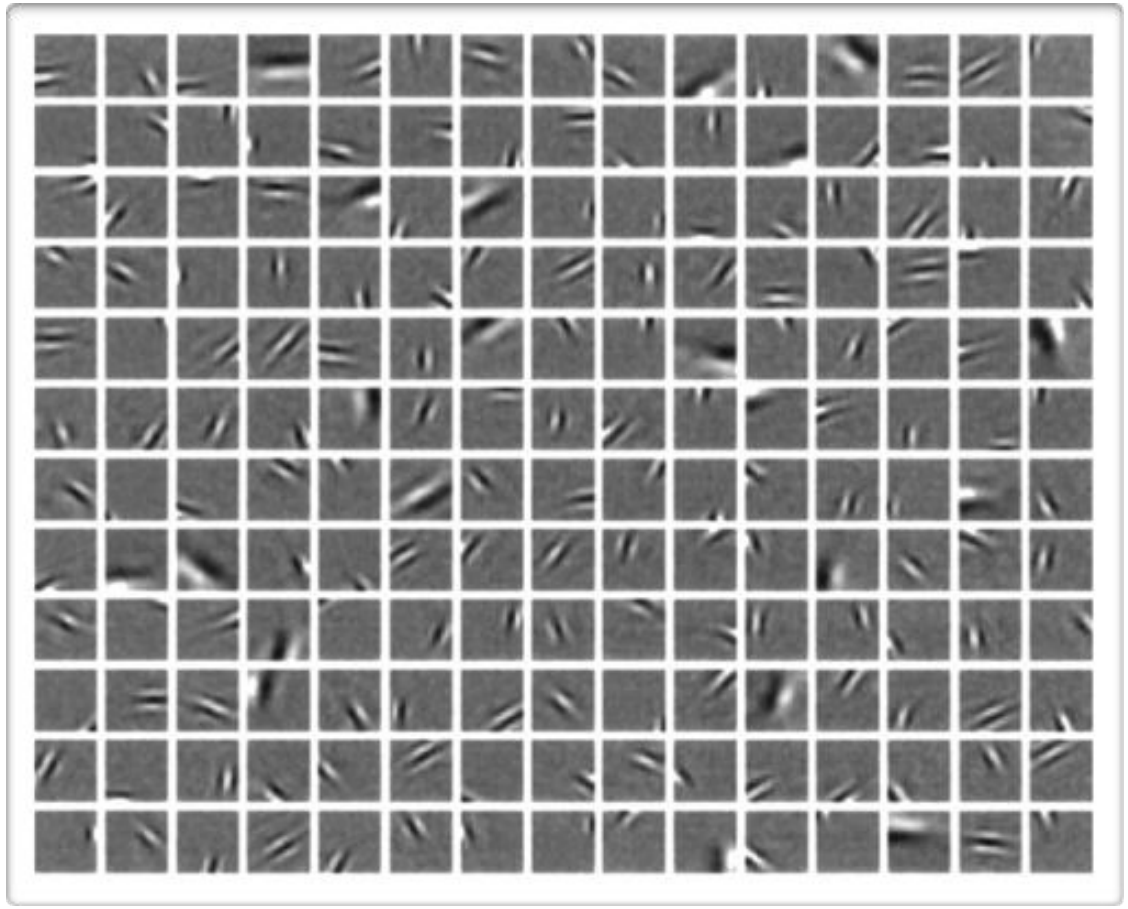


Image

Image taken from: Emergence of complex cell properties by learning to generalize in natural scenes. Karklin and Lewicki, 2009

Relationship to V1

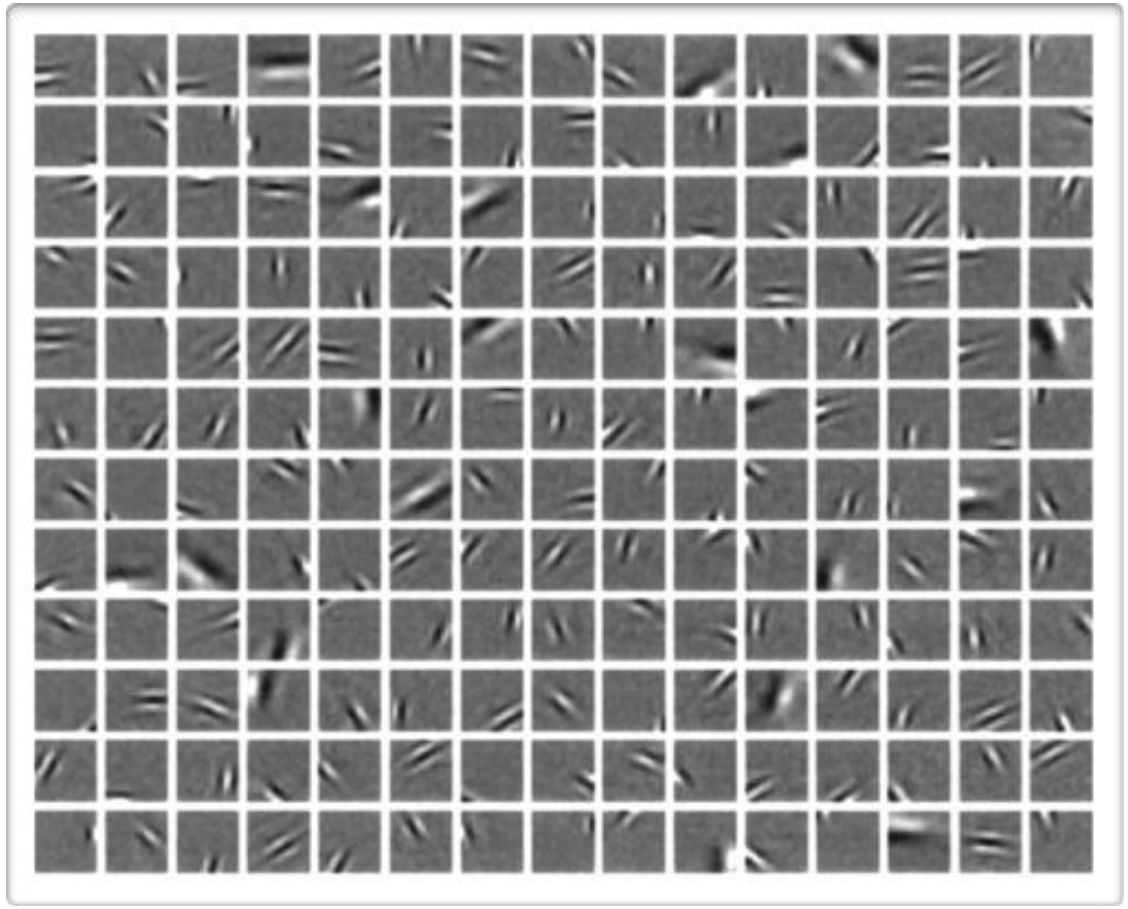
- When trained on natural image patches
  - the dictionary columns (‘‘atoms’’) look like **edge detectors**
  - each atom is tuned to a particular **position**, **orientation** and **spatial frequency**
  - V1 neurons in the mammalian brain have a similar behavior



Emergence of simple-cell receptive field properties by learning a sparse code of natural images. Olshausen and Field, 1996

Relationship to V1

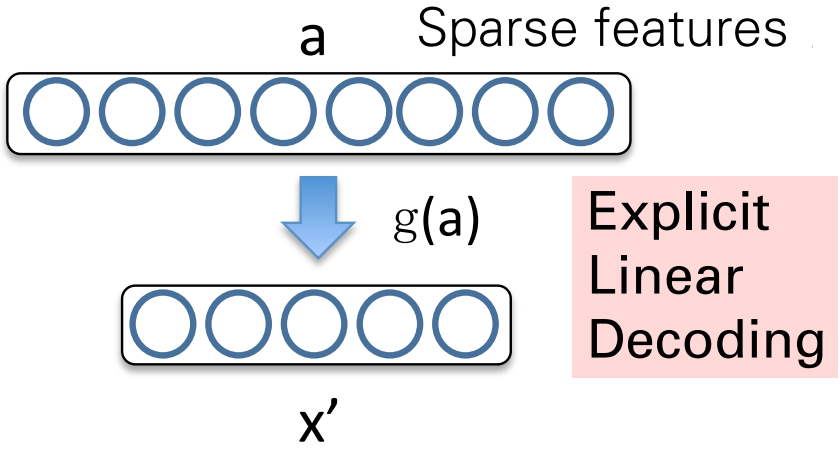
- Suggests that the brain might be learning a sparse code of visual stimulus
  - Since then, many other models have been shown to learn similar features
  - they usually all incorporate a notion of sparsity



Emergence of simple-cell receptive field properties by learning a sparse code of natural images. Olshausen and Field, 1996

Interpreting Sparse Coding

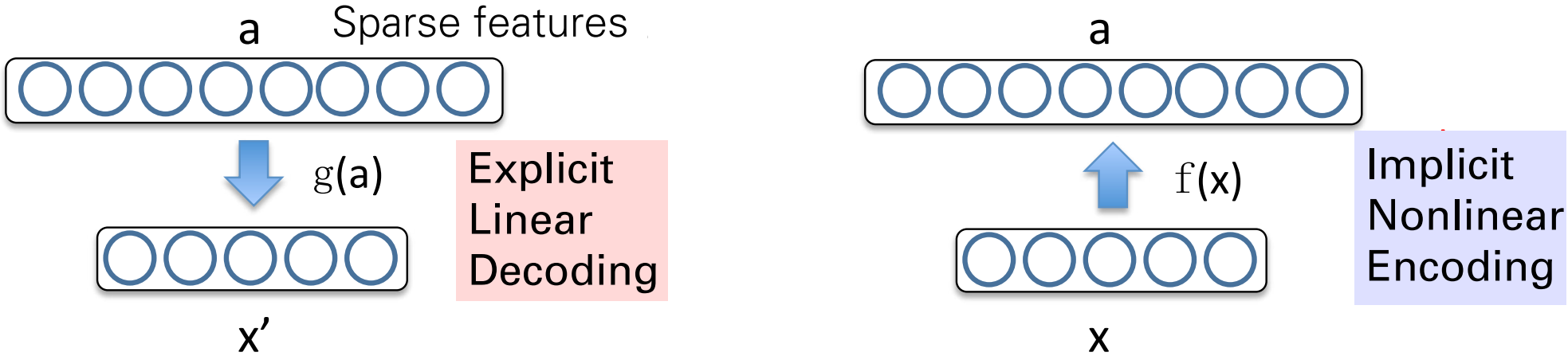
$$\min_{\mathbf{a},\boldsymbol{\phi}} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk}\boldsymbol{\phi}_k \right\|_2^2 + \lambda \sum_{n=1}^{N}\sum_{k=1}^{K} |a_{nk}|$$



Explicit
Linear
Decoding

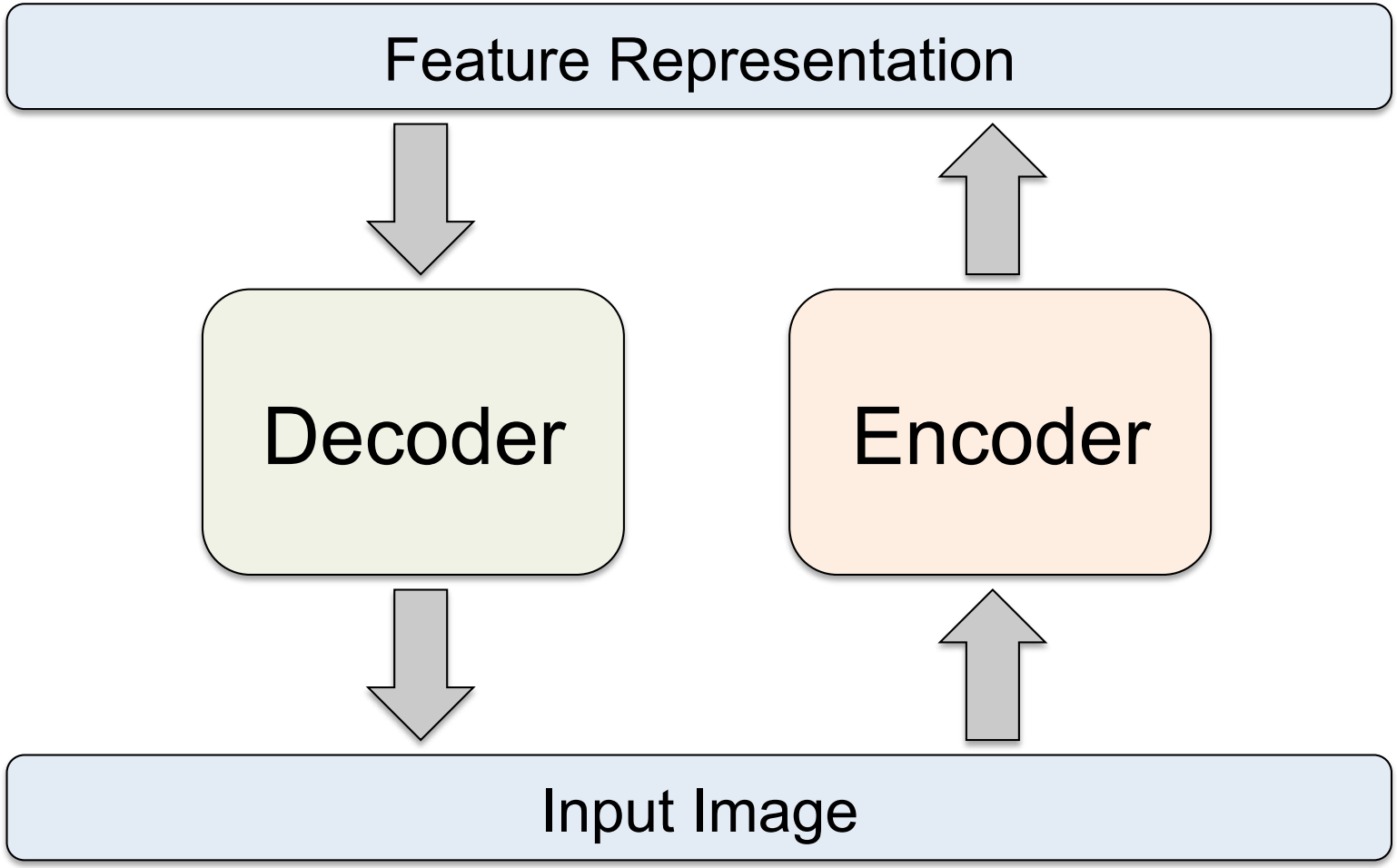
Interpreting Sparse Coding

$$\min_{\mathbf{a},\boldsymbol{\phi}} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk}\boldsymbol{\phi}_k \right\|_2^2 + \lambda \sum_{n=1}^{N}\sum_{k=1}^{K} |a_{nk}|$$

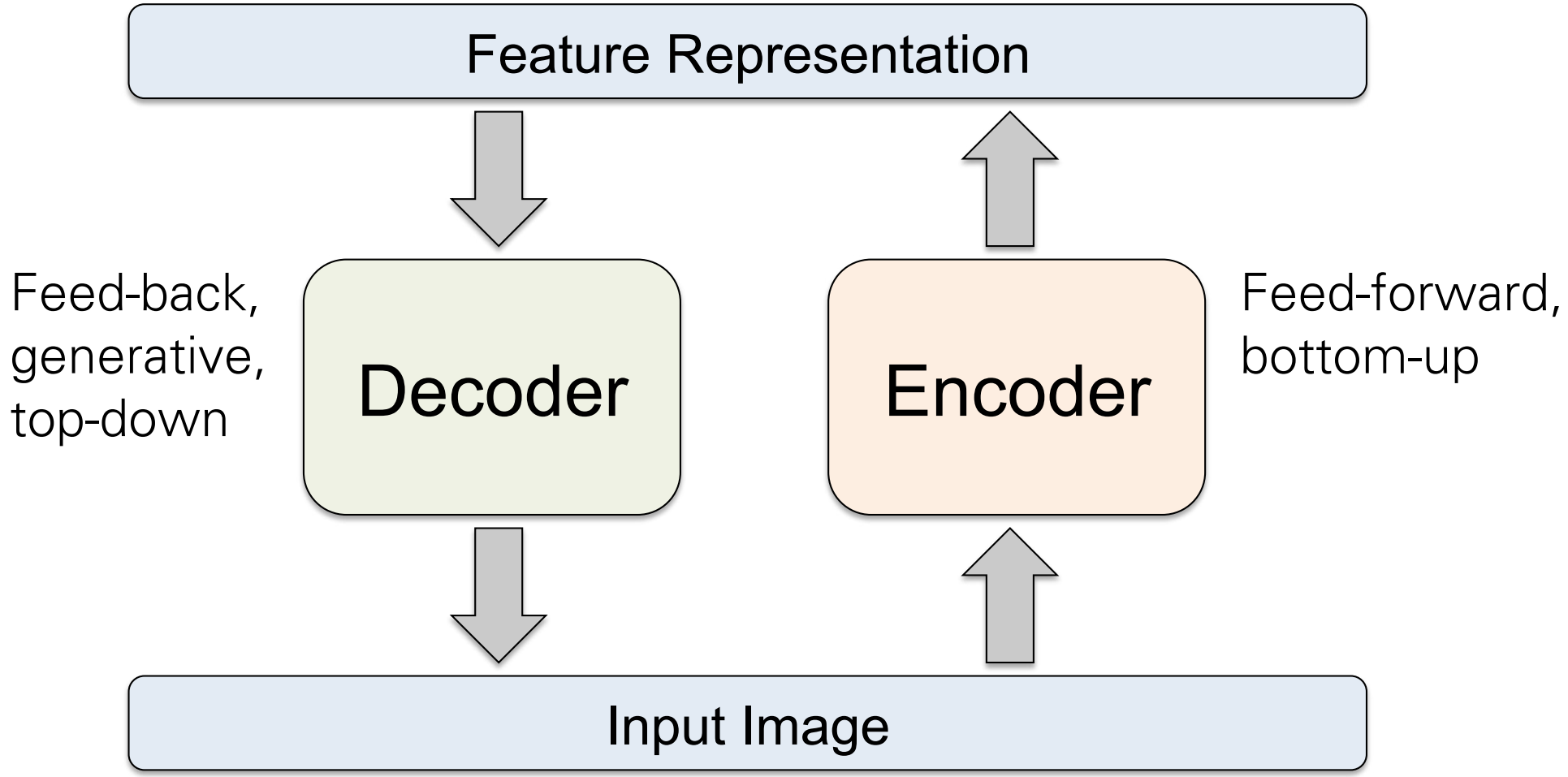


- Sparse, over-complete representation **a**.
- Encoding **a** = f(**x**) is implicit and nonlinear function of **x**.
- Reconstruction (or decoding) **x'** = g(**a**) is linear and explicit.

Autoencoder

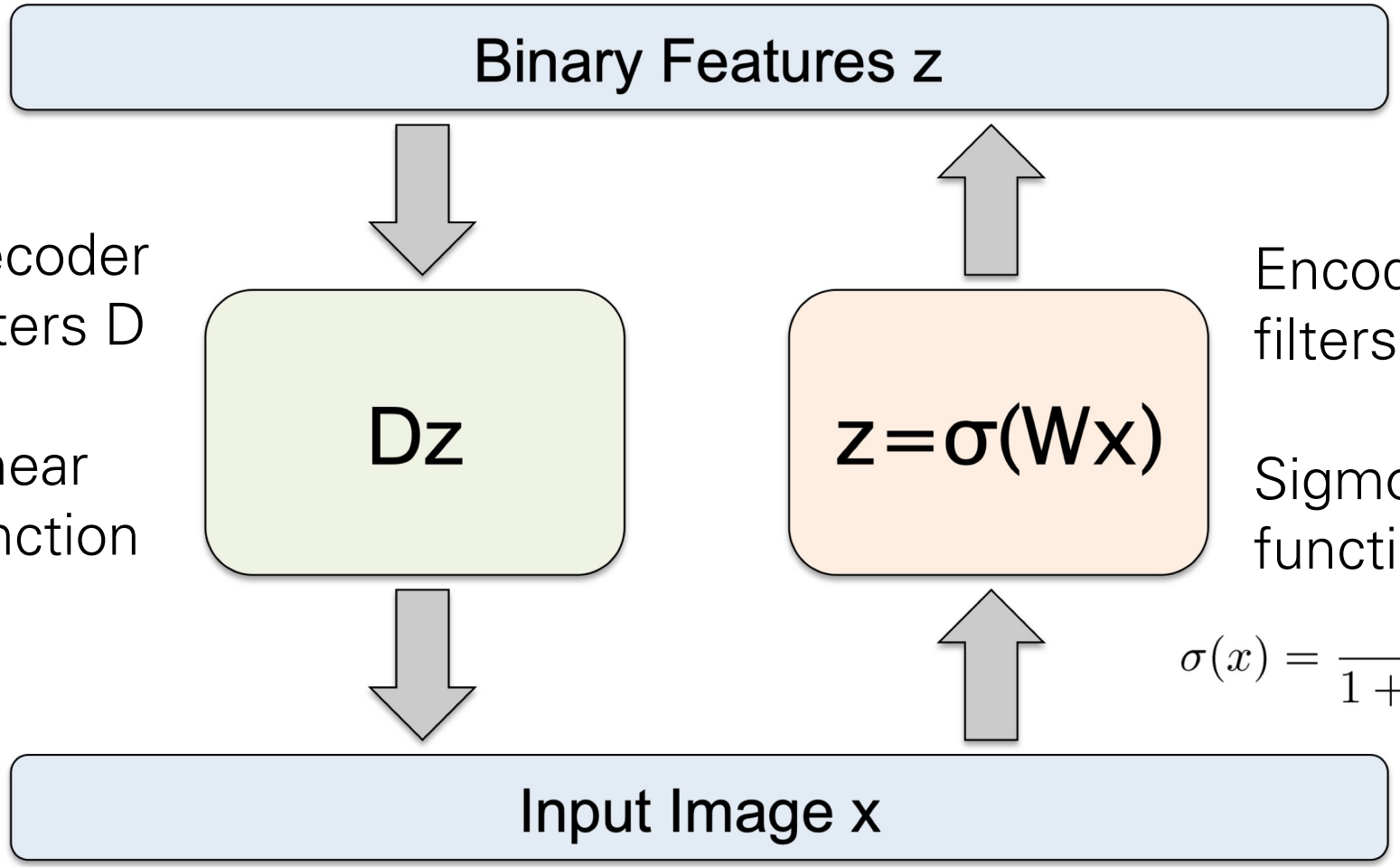


Autoencoder



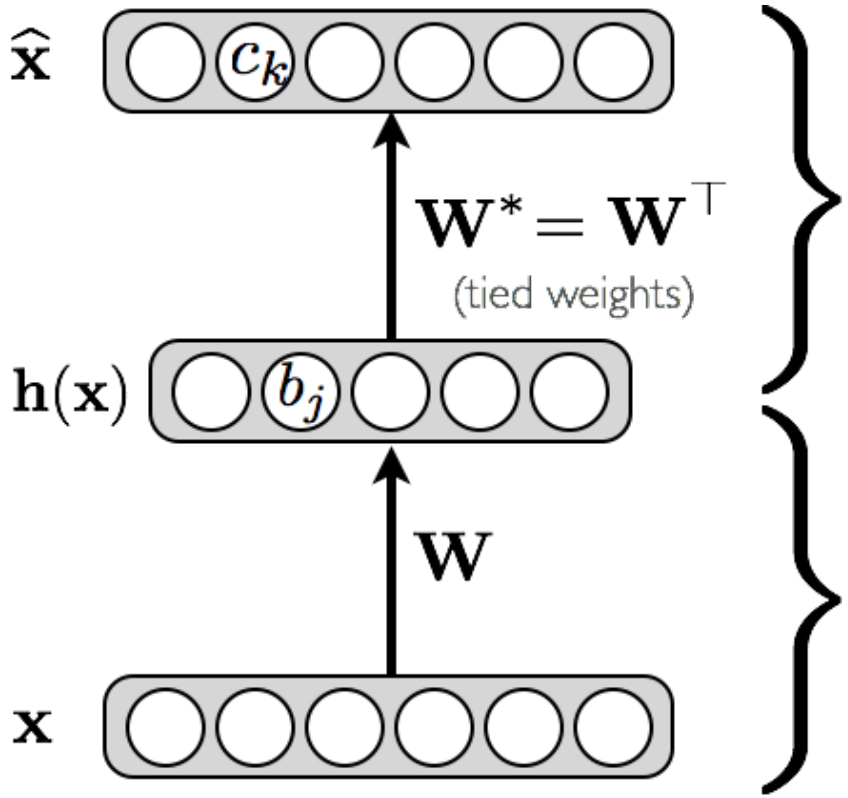
- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder



Autoencoder

- Feed-forward neural network trained to reproduce its input at the output layer



Decoder

$$\begin{aligned}\hat{\mathbf{x}} &= o(\hat{\mathbf{a}}(\mathbf{x})) \\ &= \underbrace{\text{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))}_{\text{for binary units}}\end{aligned}$$

Encoder

$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\ &= \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$

Loss Function

- **Loss function for binary inputs**

$$l(f(\mathbf{x})) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k) \log(1 - \widehat{x}_k)\right)$$

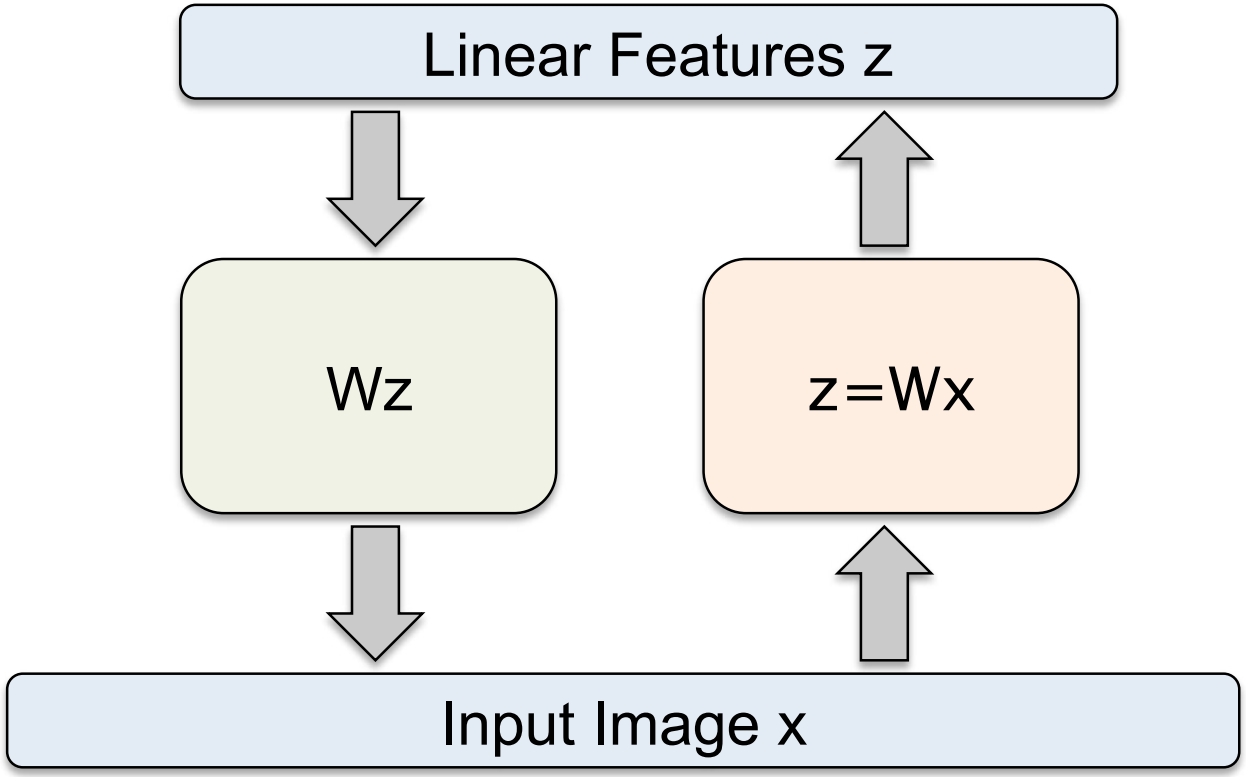
  - Cross-entropy error function (reconstruction loss) $f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$

- **Loss function for real-valued inputs**

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\widehat{x}_k - x_k)^2$$

  - sum of squared differences (reconstruction loss)
  - we use a linear activation function at the output

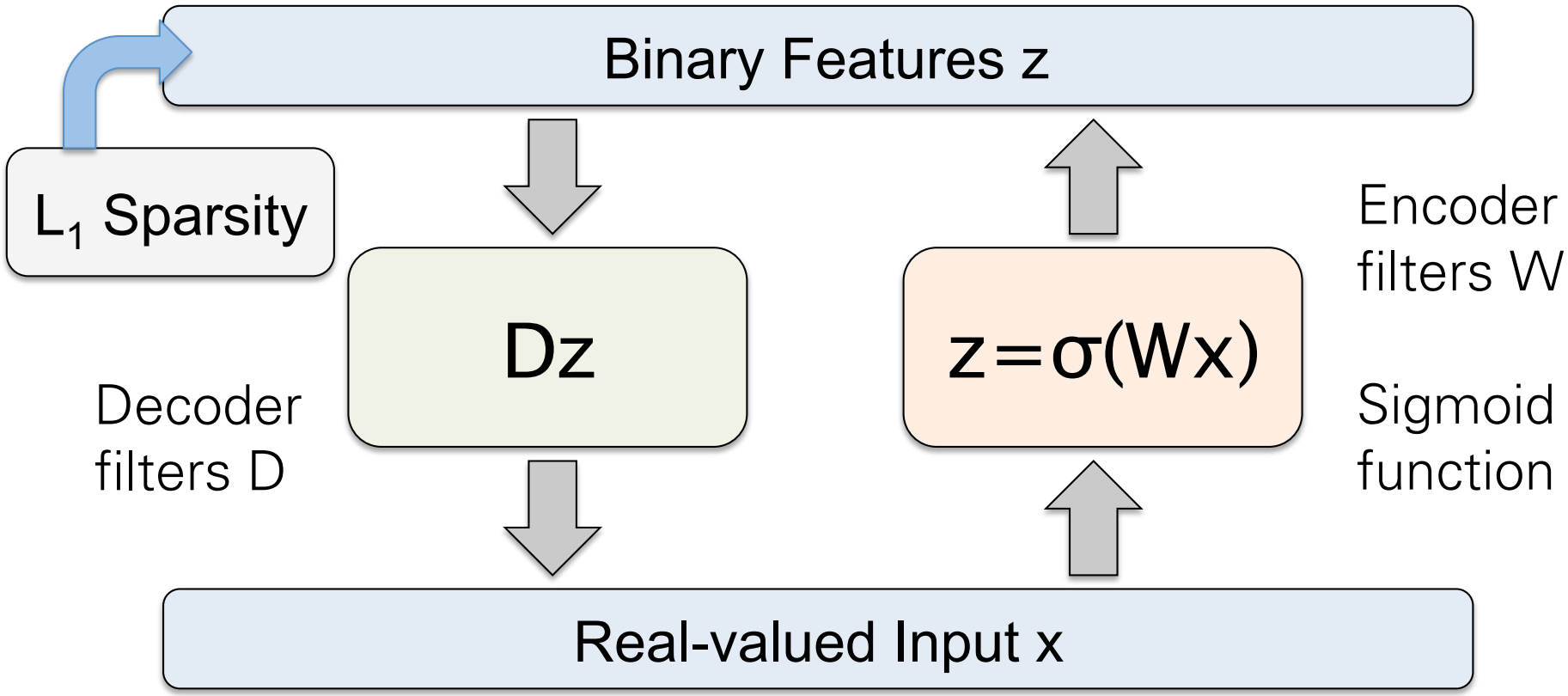
Autoencoder



- If the **hidden and output layers are linear**, it will learn hidden units that are a linear function of the data and minimize the squared error.
- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

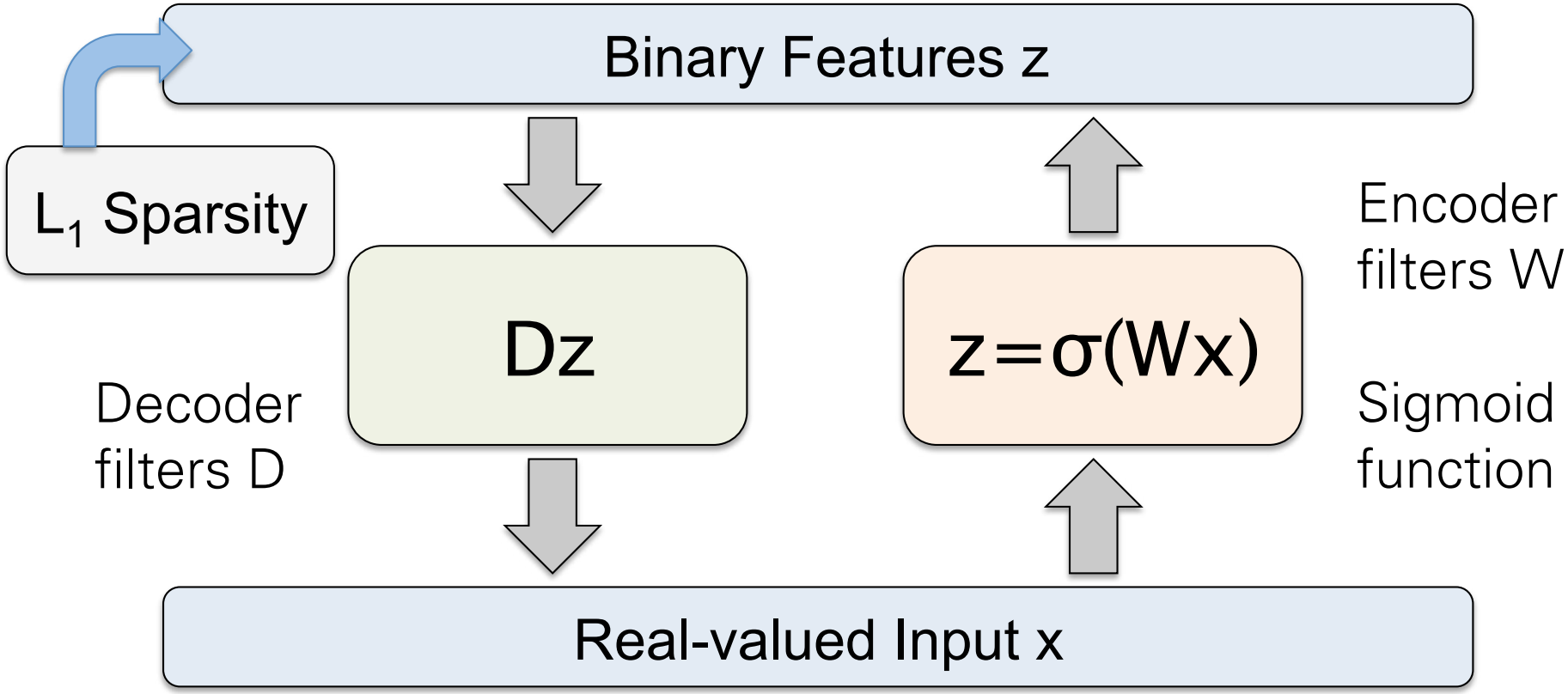
- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Predictive Sparse Decomposition



(Kavukcuoglu, Ranzato, Fergus, LeCun, 2009)

Predictive Sparse Decomposition

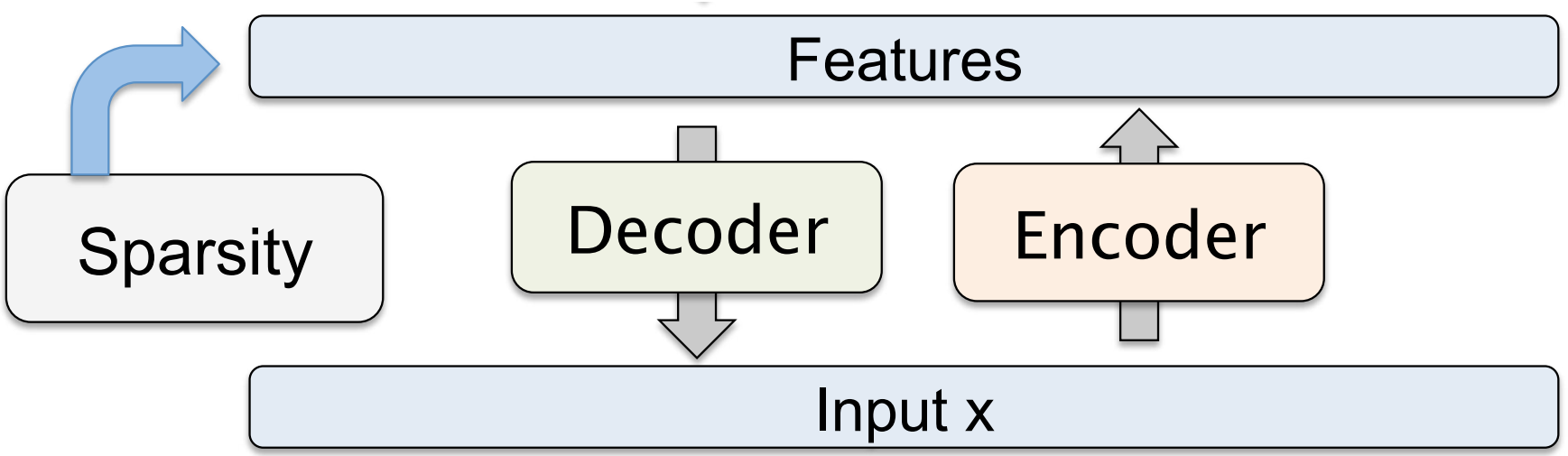


At training time

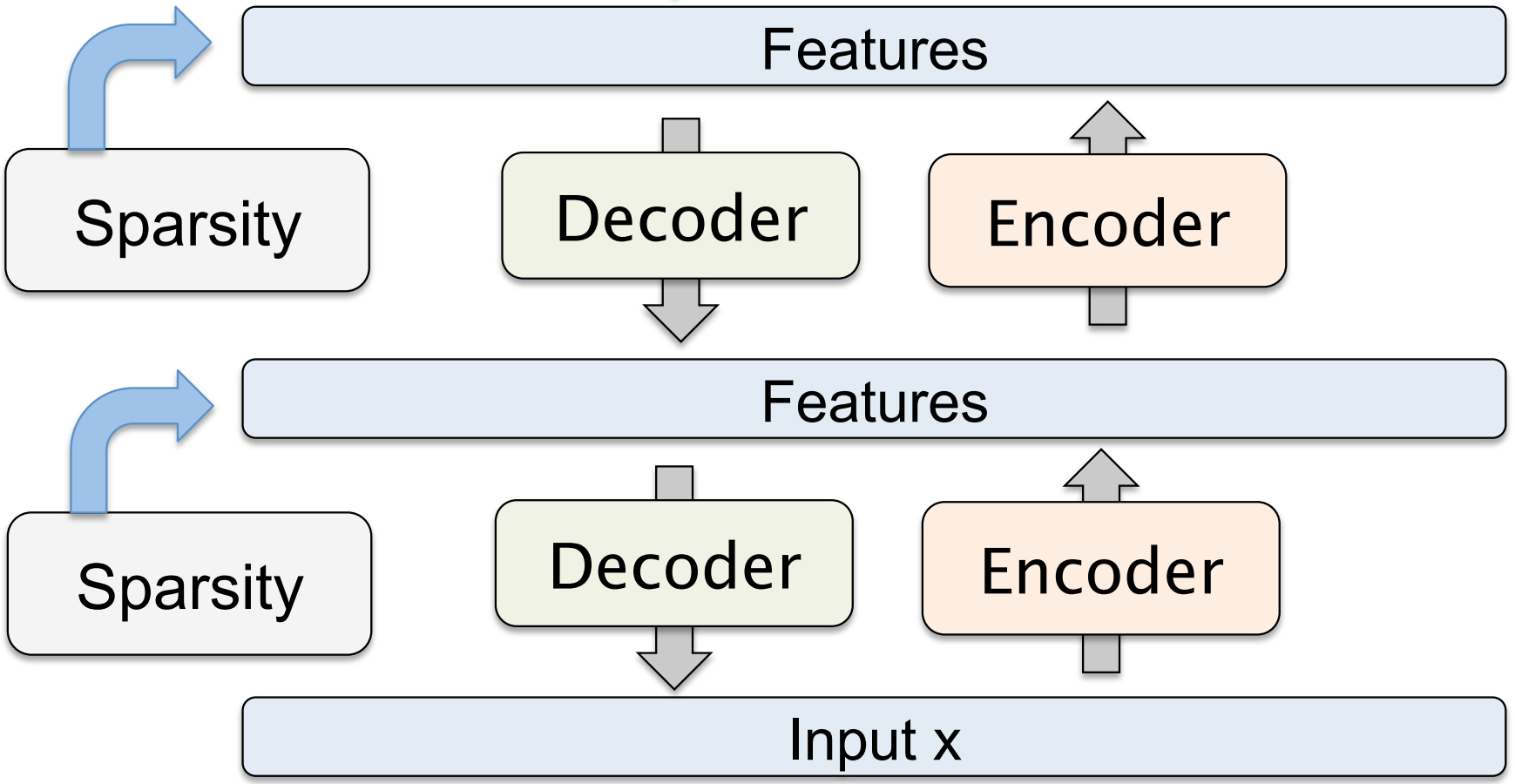
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z}-\mathbf{x}||_2^2+\lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x})-\mathbf{z}||_2^2}_{\text{Encoder}}$$

(Kavukcuoglu, Ranzato, Fergus, LeCun, 2009)

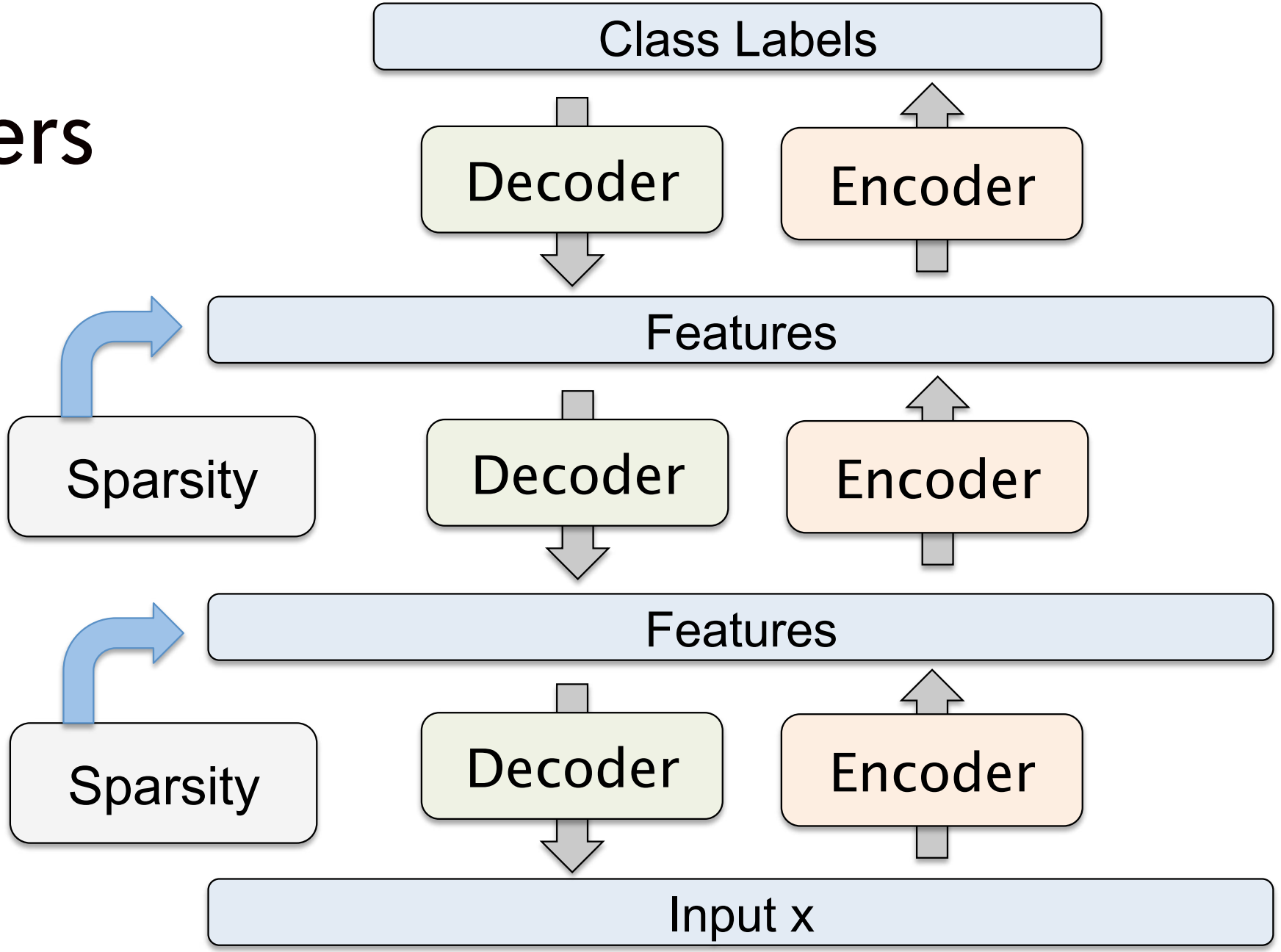
Stacked Autoencoders



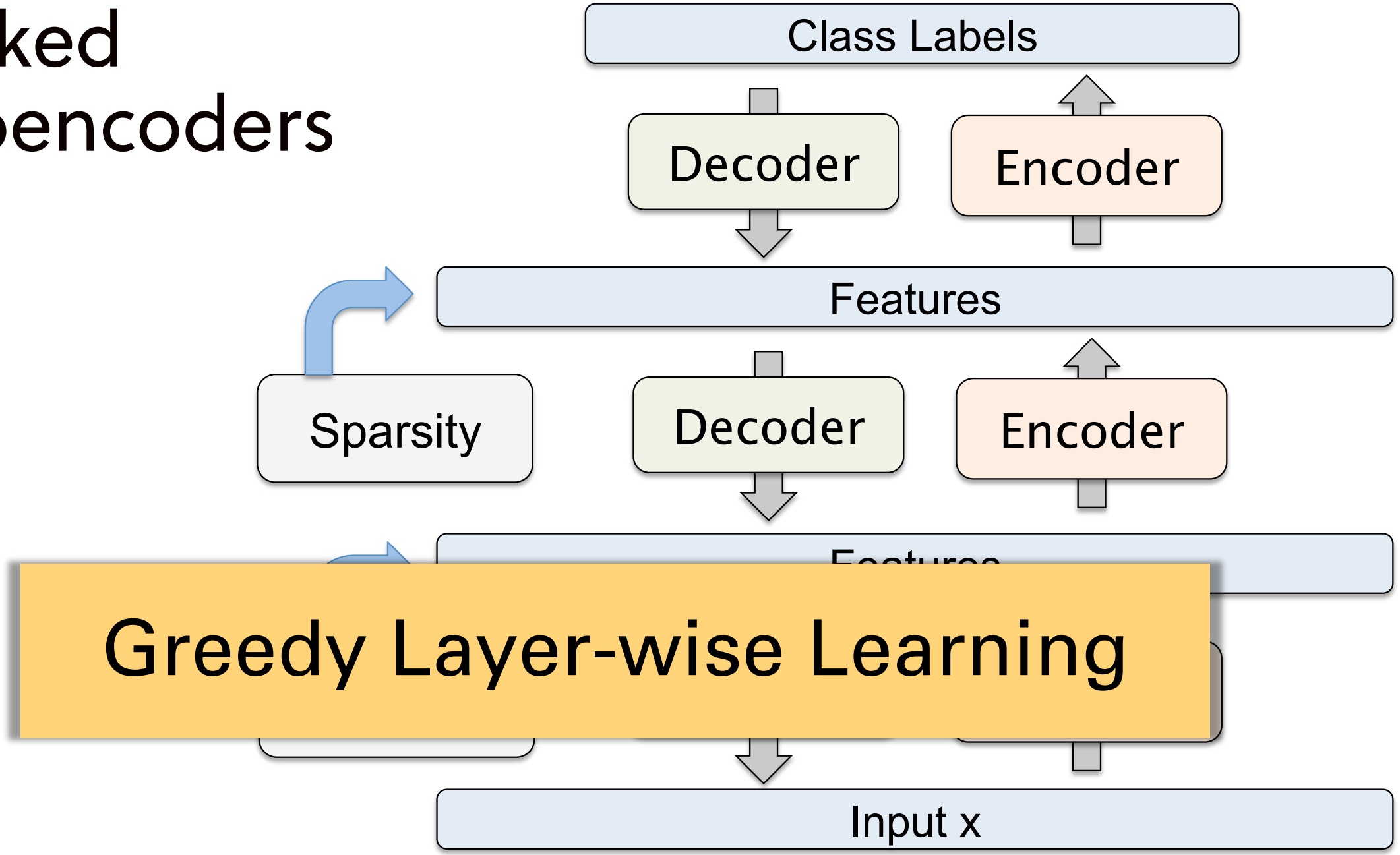
Stacked Autoencoders



Stacked Autoencoders



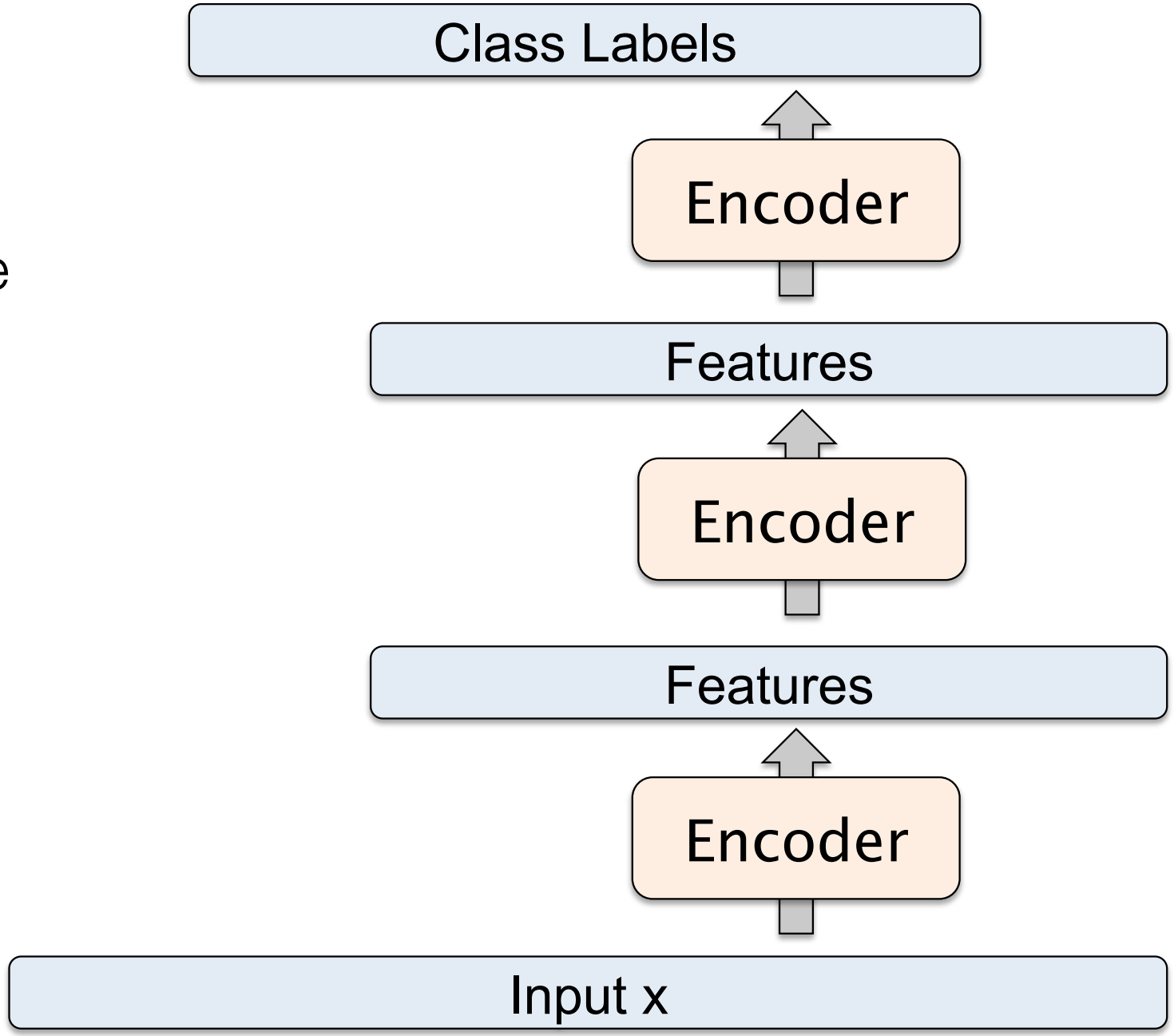
Stacked Autoencoders



Greedy Layer-wise Learning

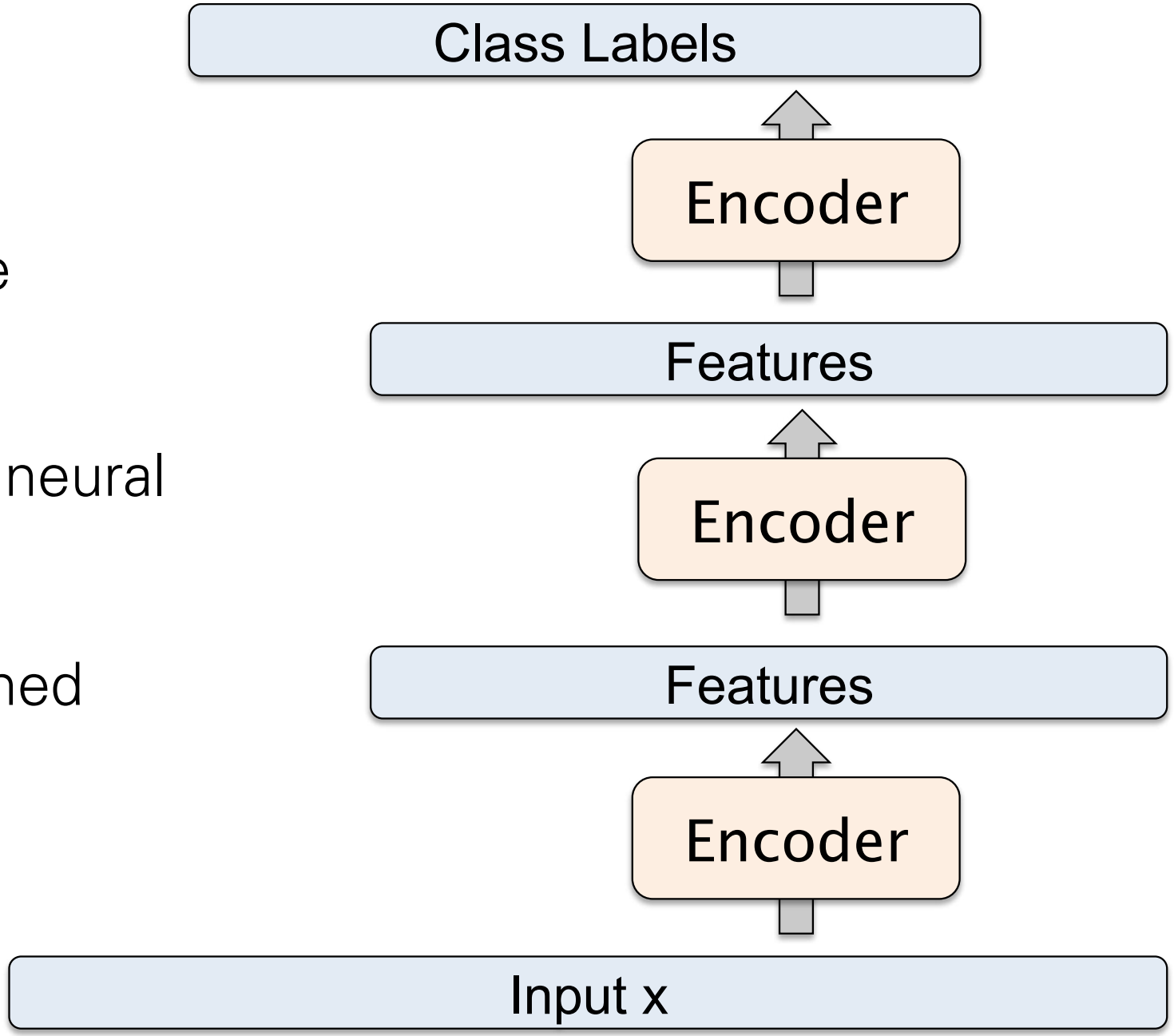
Stacked Autoencoders

- Remove decoders and use feed-forward part.

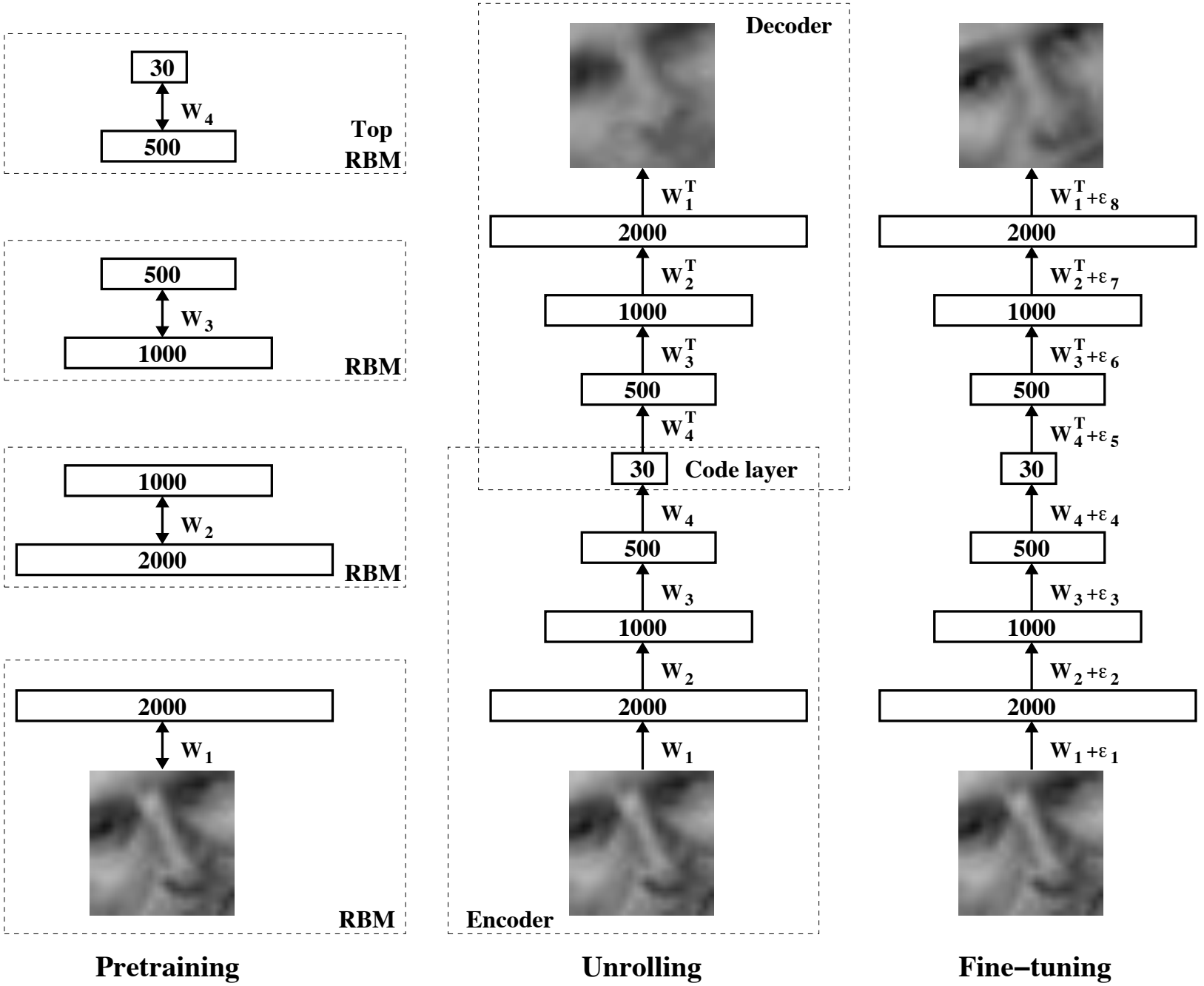


Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.

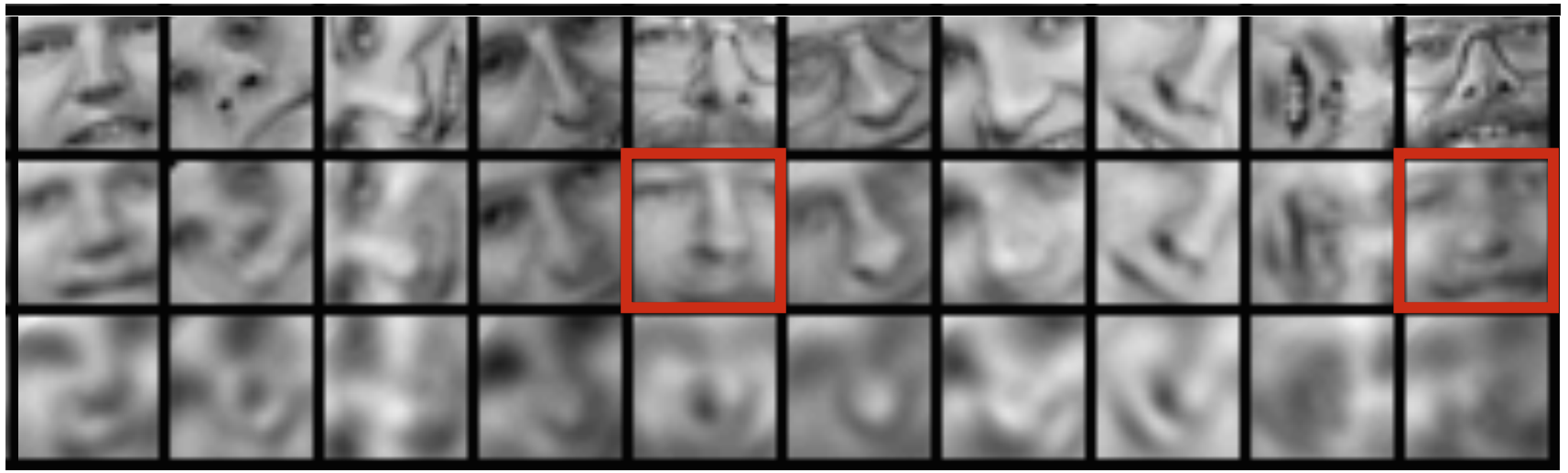


Deep Autoencoder



Deep Autoencoders

- 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Oliver face patches.



- **Top:** Random samples from the test dataset.
- **Middle:** Reconstructions by the 30-dimensional deep autoencoder.
- **Bottom:** Reconstructions by the 30-dimensional PCA.

(Hinton and Salakhutdinov, Science 2006)

Autoregressive Models

Learning the Distribution of Natural Data

$$p(\mathbf{x}) = \prod_i p(x_i|\mathbf{x}_<)$$

1D sequences such as text or sound

$$p(\mathbf{x}) = \prod_j \prod_i p(x_{i,j}|\mathbf{x}_<)$$

2D tensors such as images

$$p(\mathbf{x}) = \prod_k \prod_j \prod_i p(x_{i,j,k}|\mathbf{x}_<)$$

3D tensors such as videos

- Fully visible belief networks — [Frey et al.,1996] [Frey, 1998]
- NADE/MADE — [Larochelle and Murray, 2011] [Germain et al., 2015]
- PixelRNN/PixelCNN (Images) — [van den Oord, Kalchbrenner, Kavukcuoglu, 2016] [van den Oord, Kalchbrenner, Vinyals, et al., 2016]
- Video Pixel Nets (Videos) — [Kalchbrenner, van den Oord, Simonyan, et al., 2016]
- ByteNet (Language/seq2seq) — [Kalchbrenner, Espeholt, Simonyan, et al., 2016]
- WaveNet (Audio) — [van den Oord, Dieleman, Zen, et al., 2016]

Slide adapted from Nal Kalchbrenner

PixelCNN

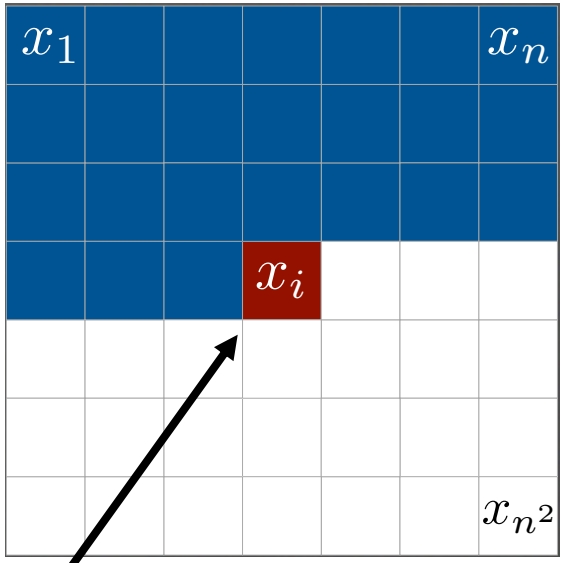
$P($  $)$

- approach the generation process as sequence modeling problem
- an explicit density model

Slide adapted from Nal Kalchbrenner

PixelCNN

$P($  $)$

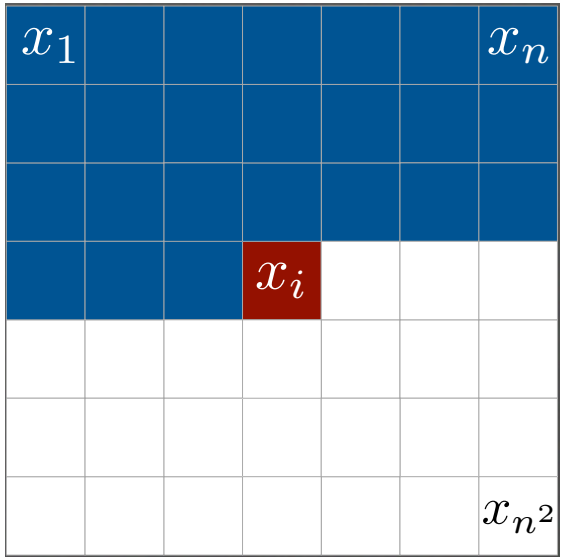


$$p(x_i|\mathbf{x}_<)$$

Slide adapted from Nal Kalchbrenner

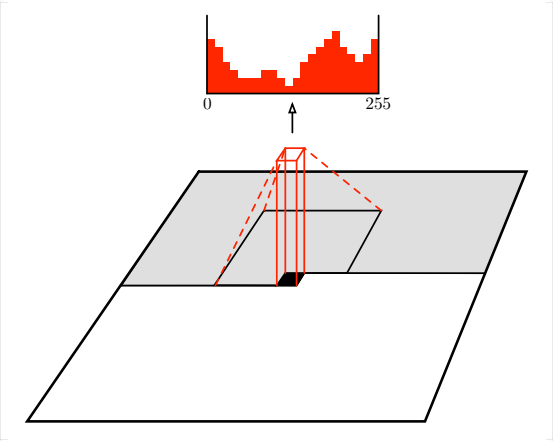
PixelCNN

$P($  $)$



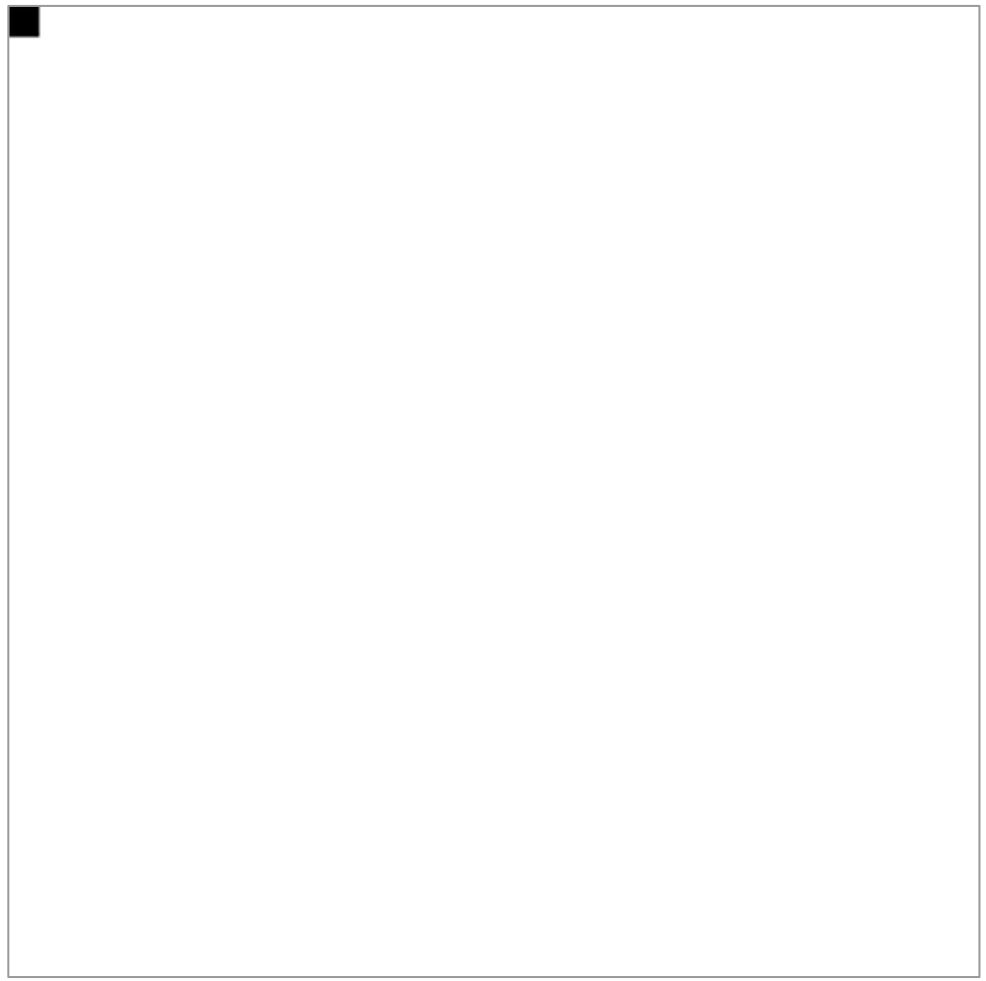
By chain rule and using pixels as variables,

$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$



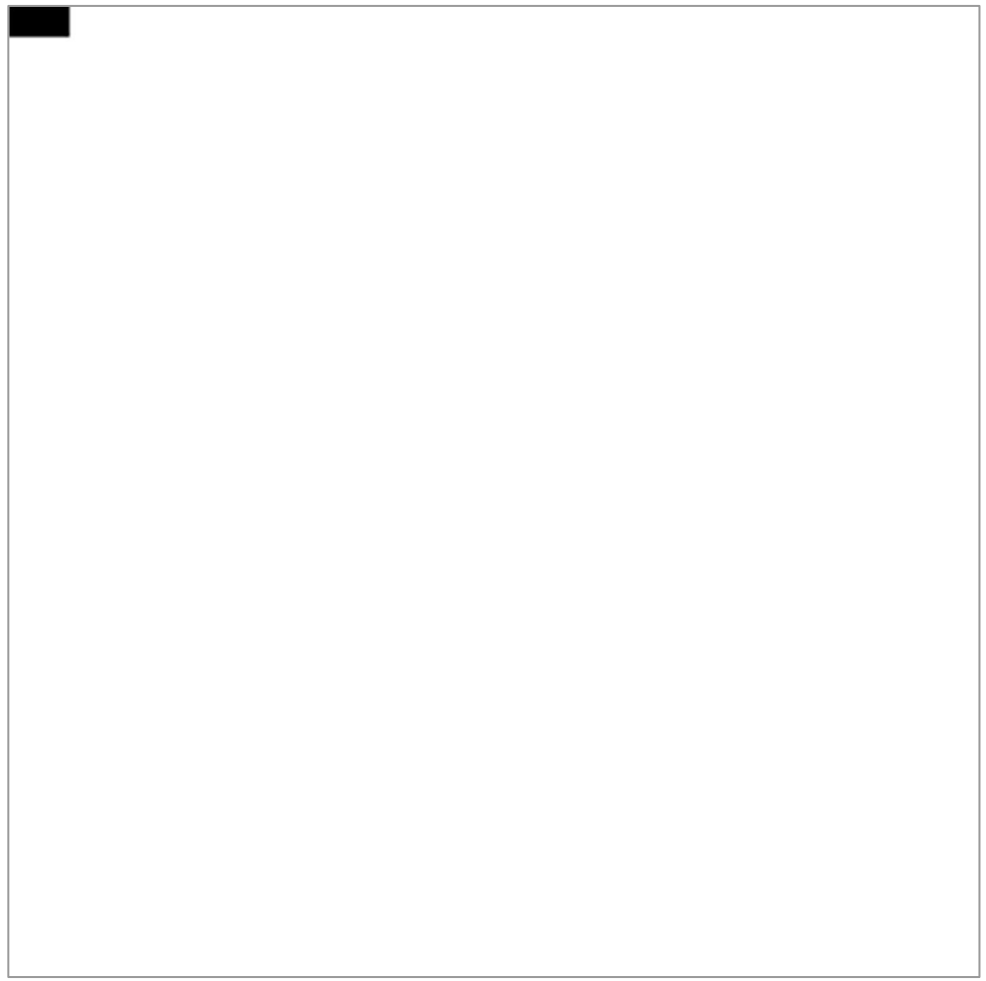
Slide adapted from Nal Kalchbrenner

PixelCNN



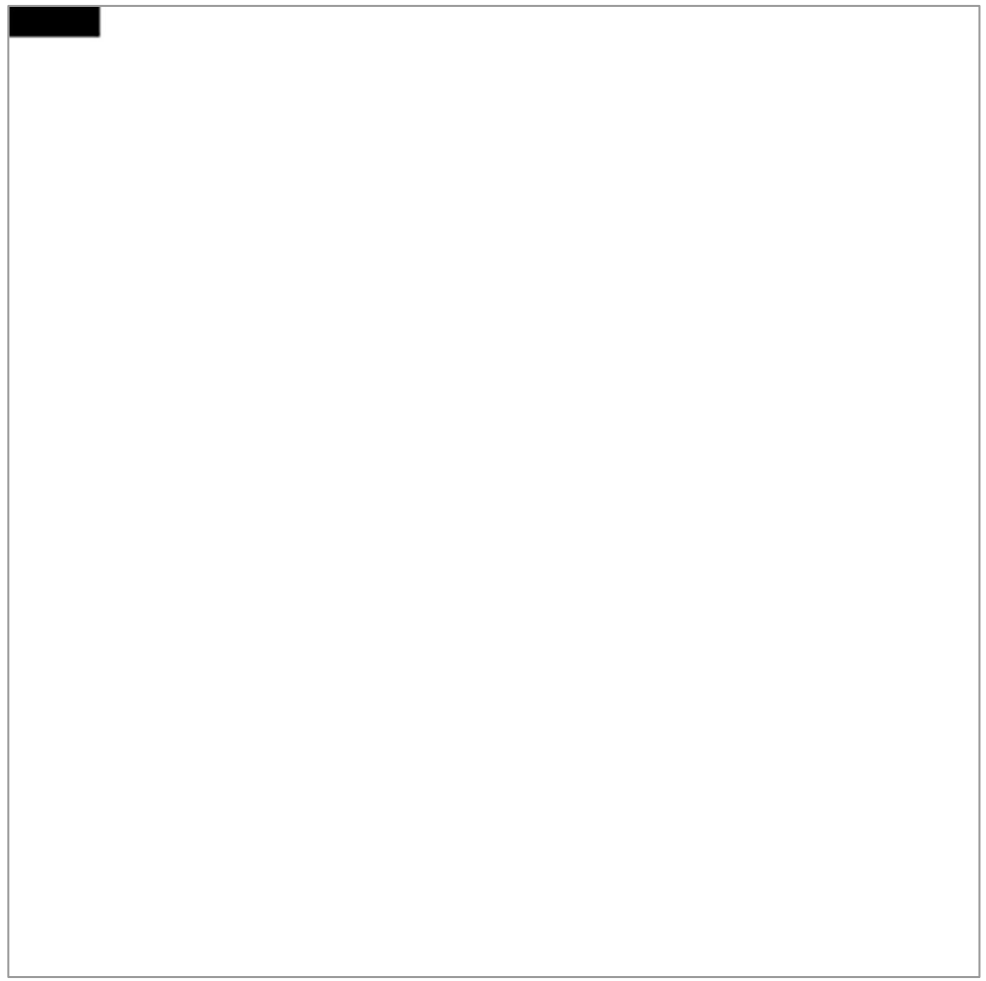
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



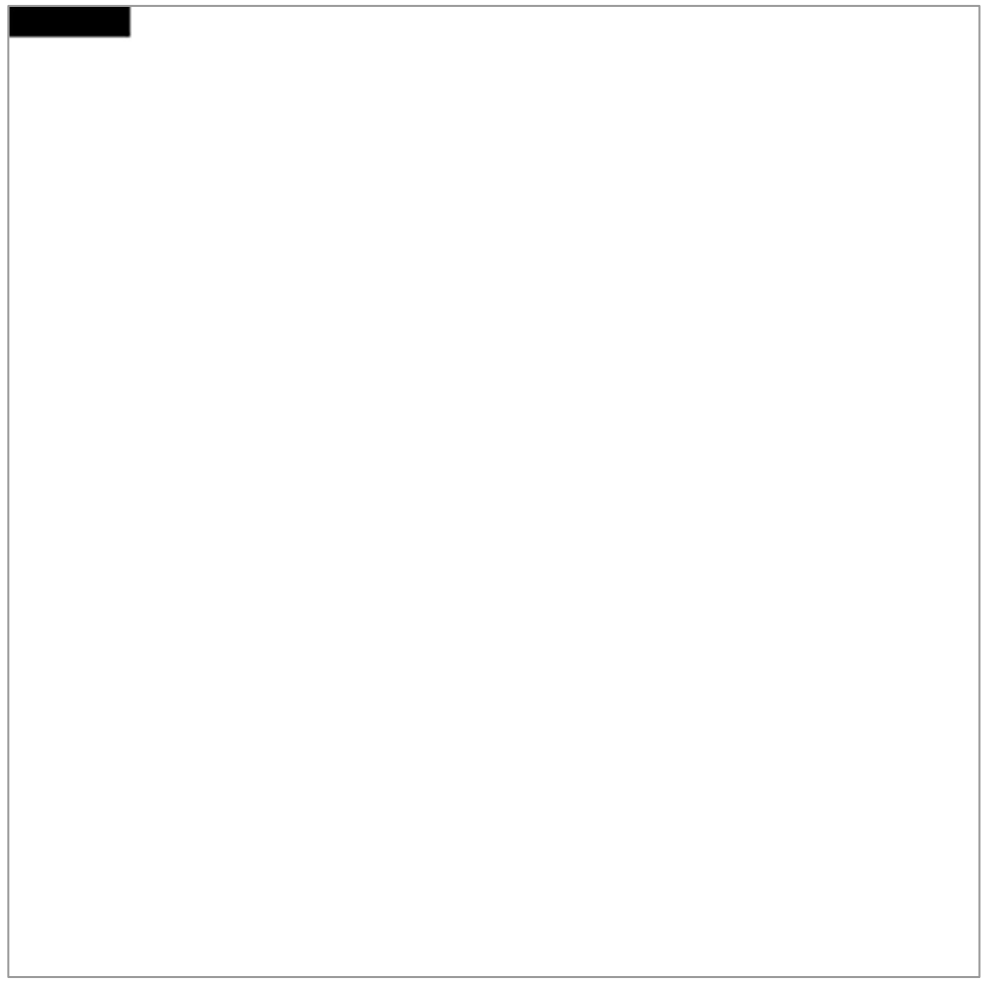
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



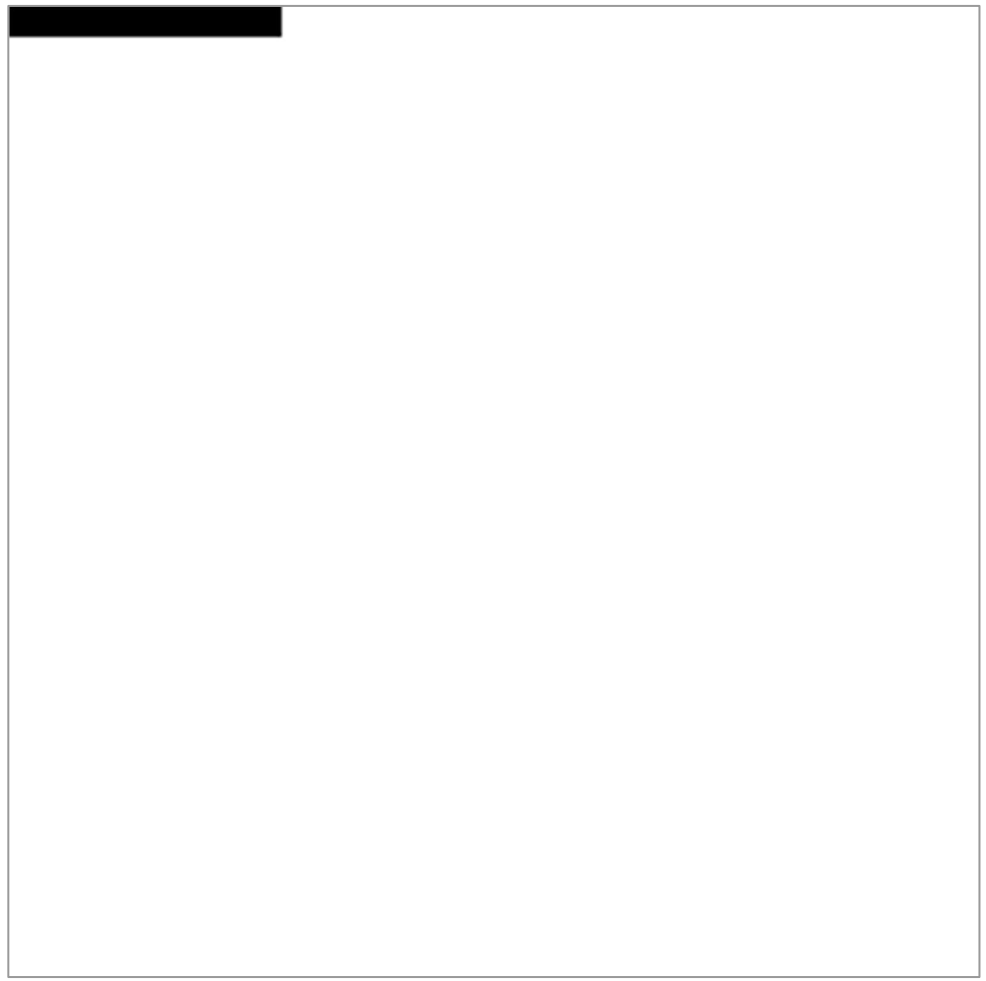
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



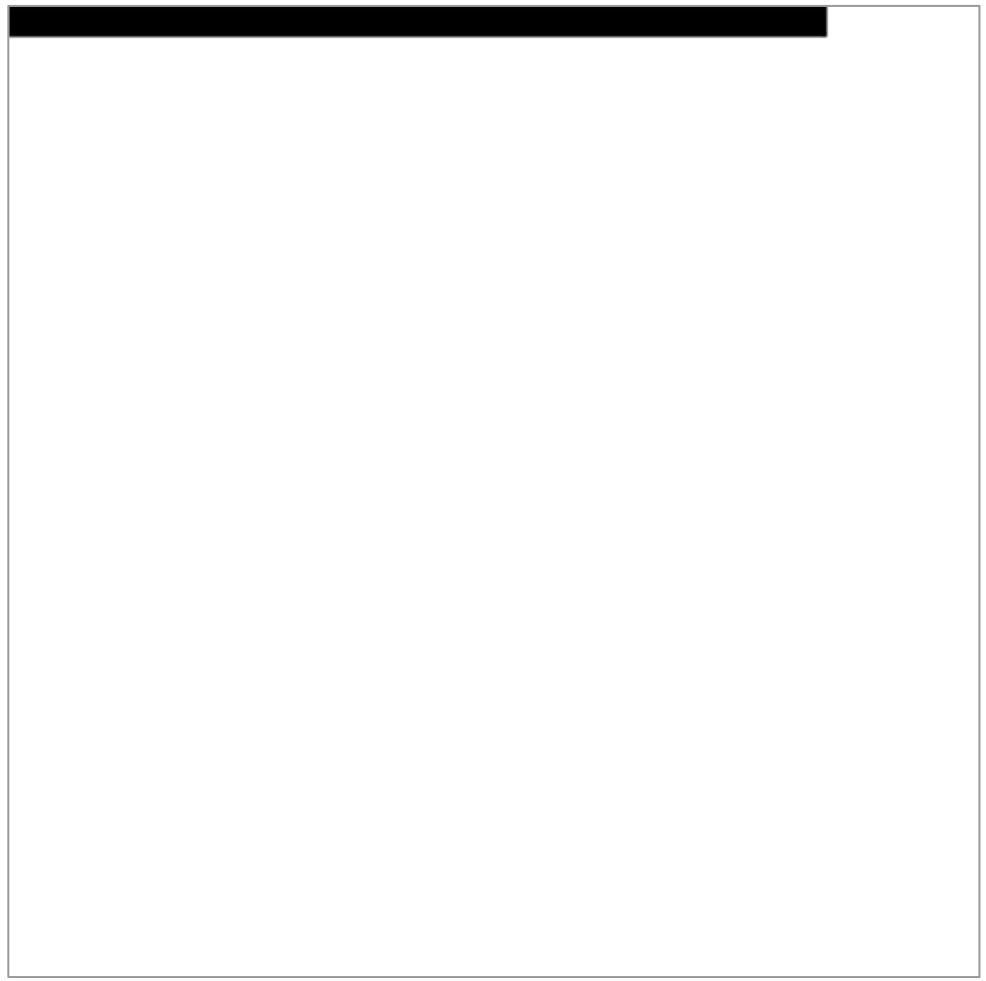
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN

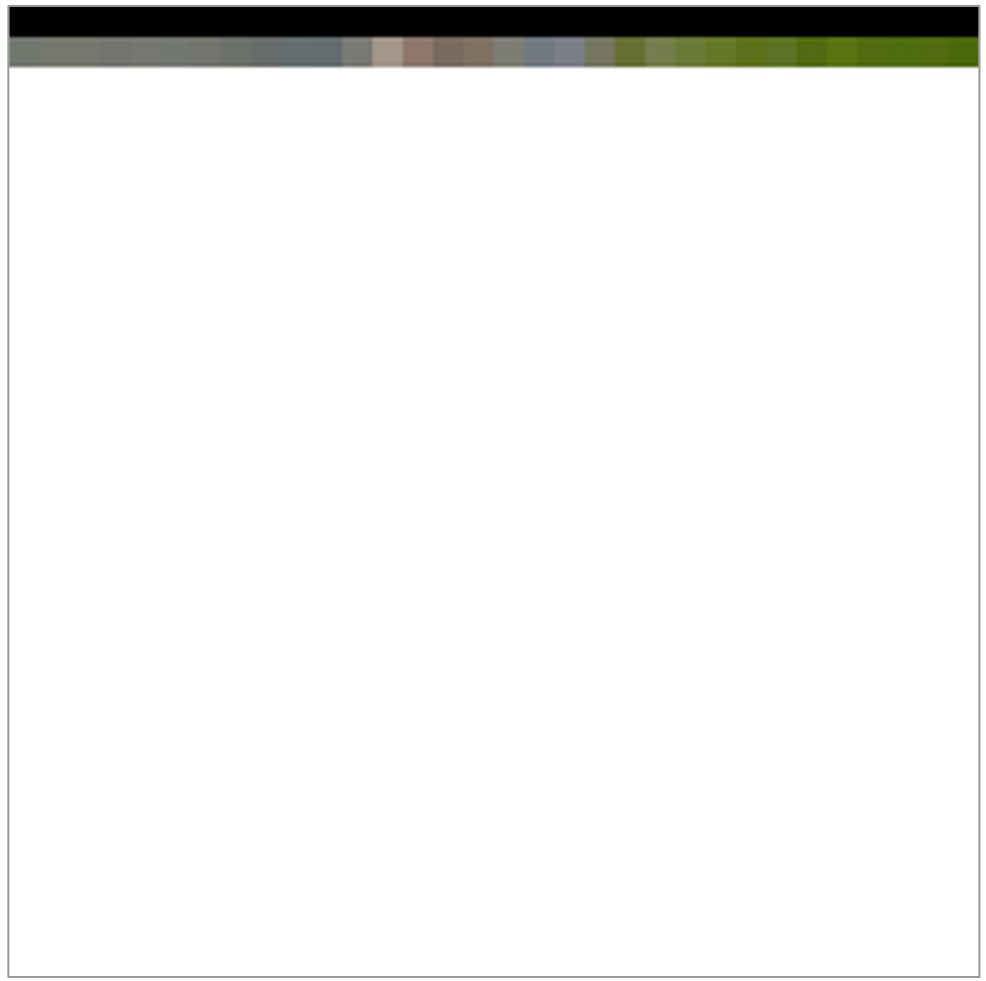


Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN

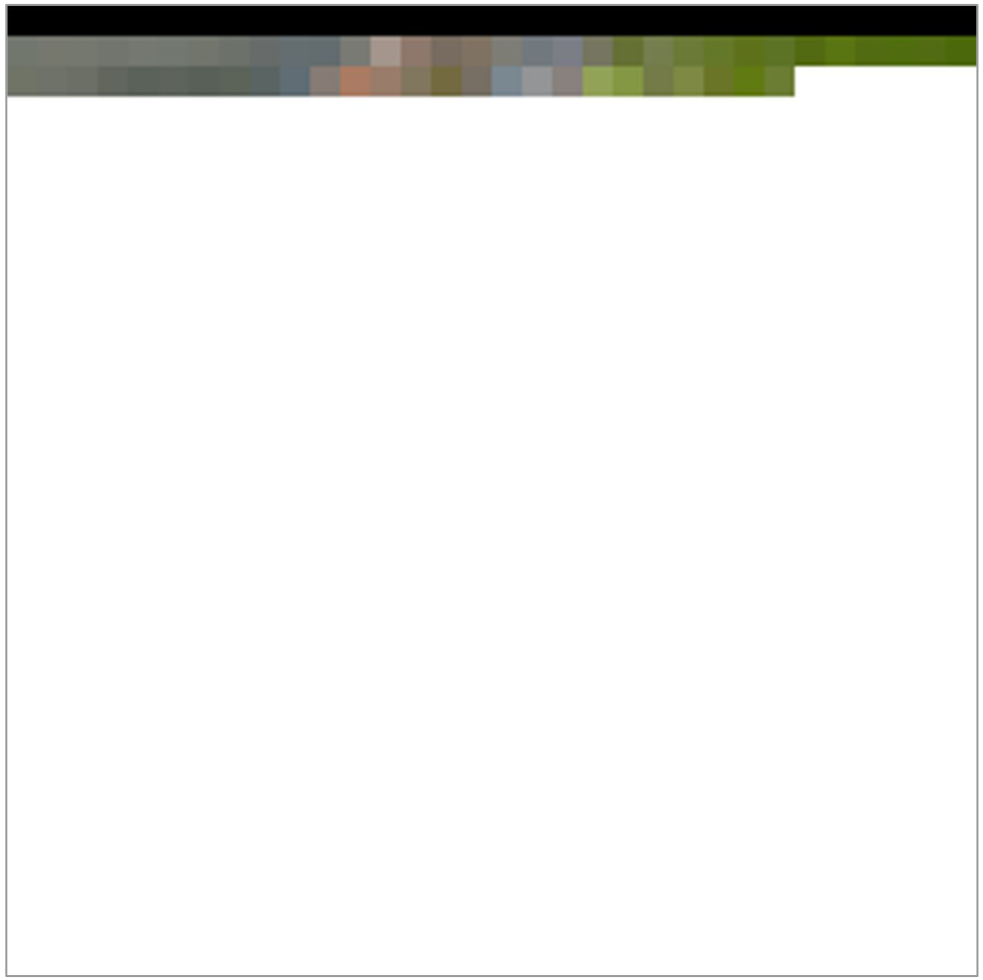
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



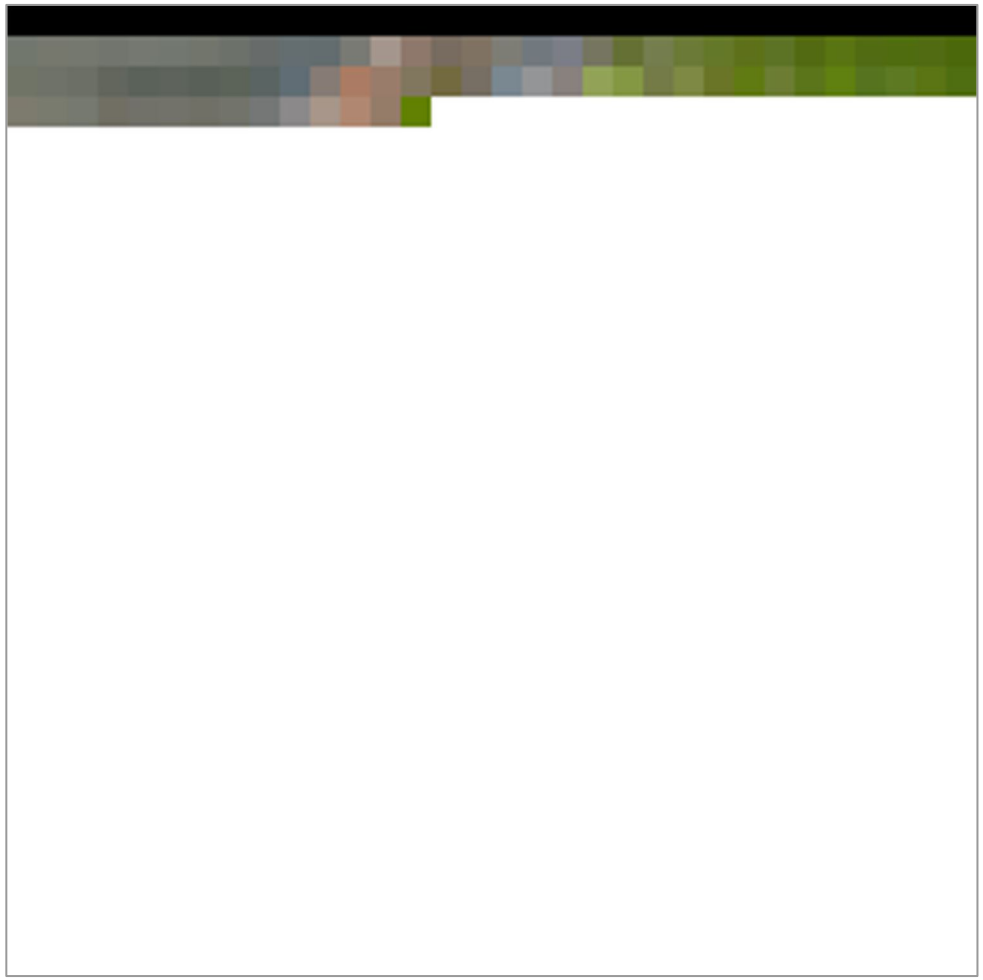
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



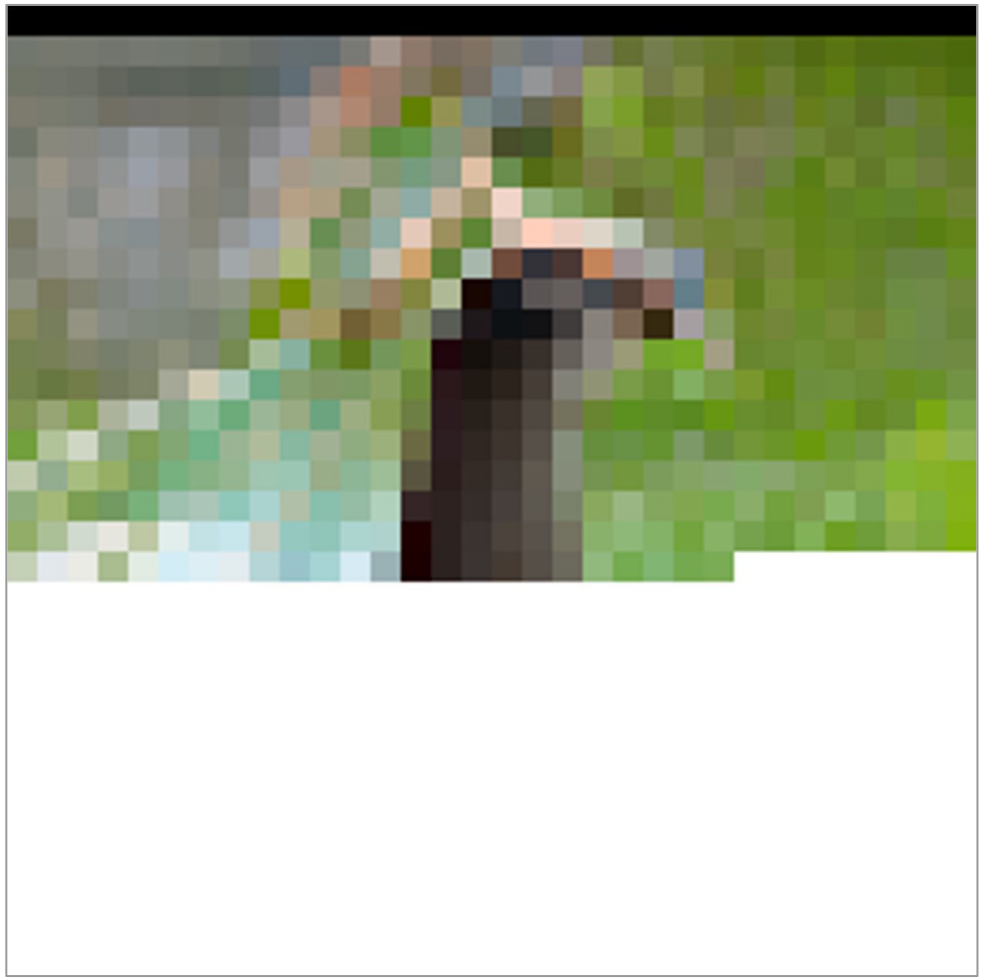
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN



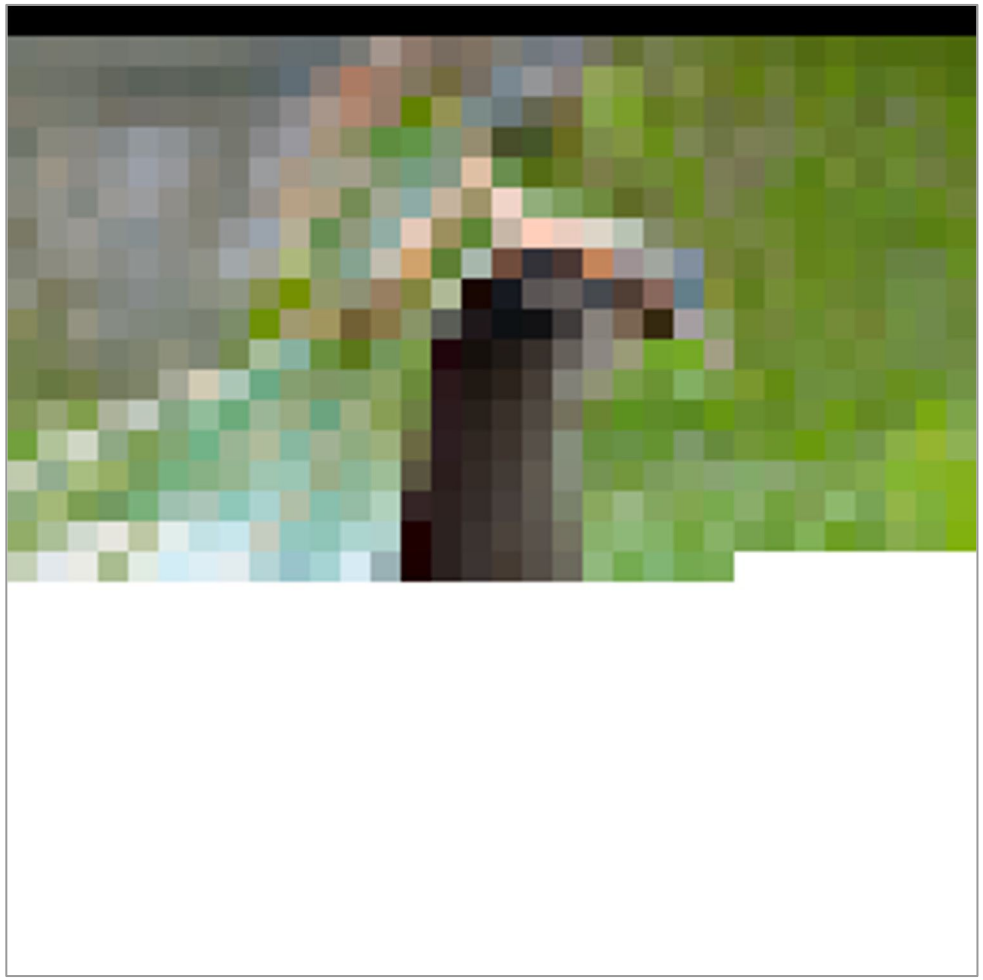
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

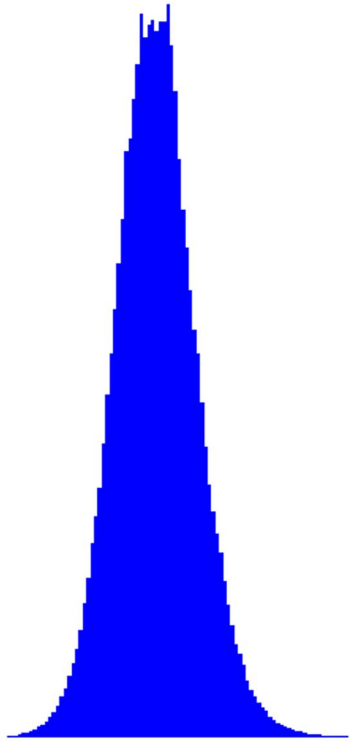
PixelCNN



Slide adapted from
Oriol Vinyals and Navdeep Jaitly

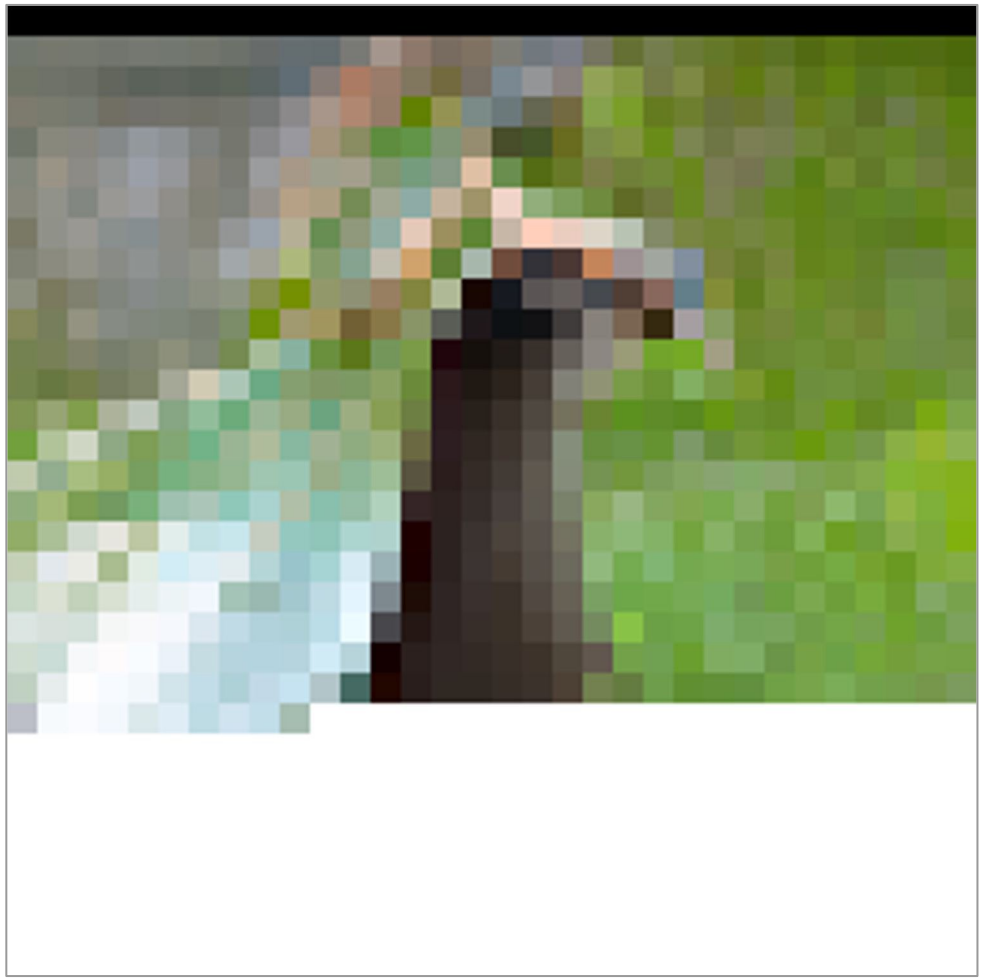
PixelCNN – Softmax Sampling

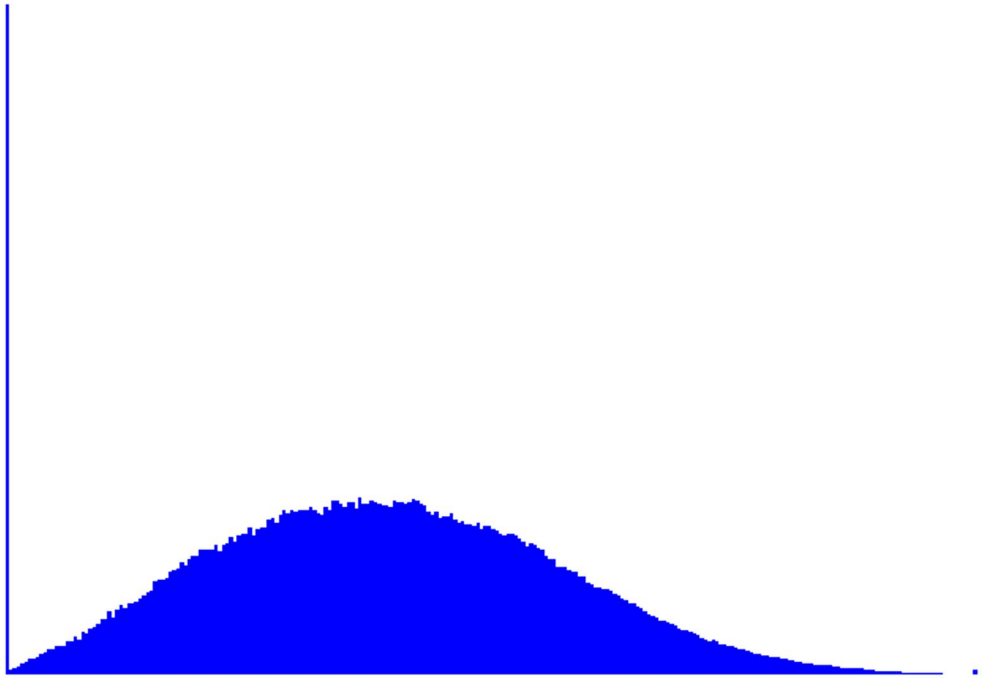




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

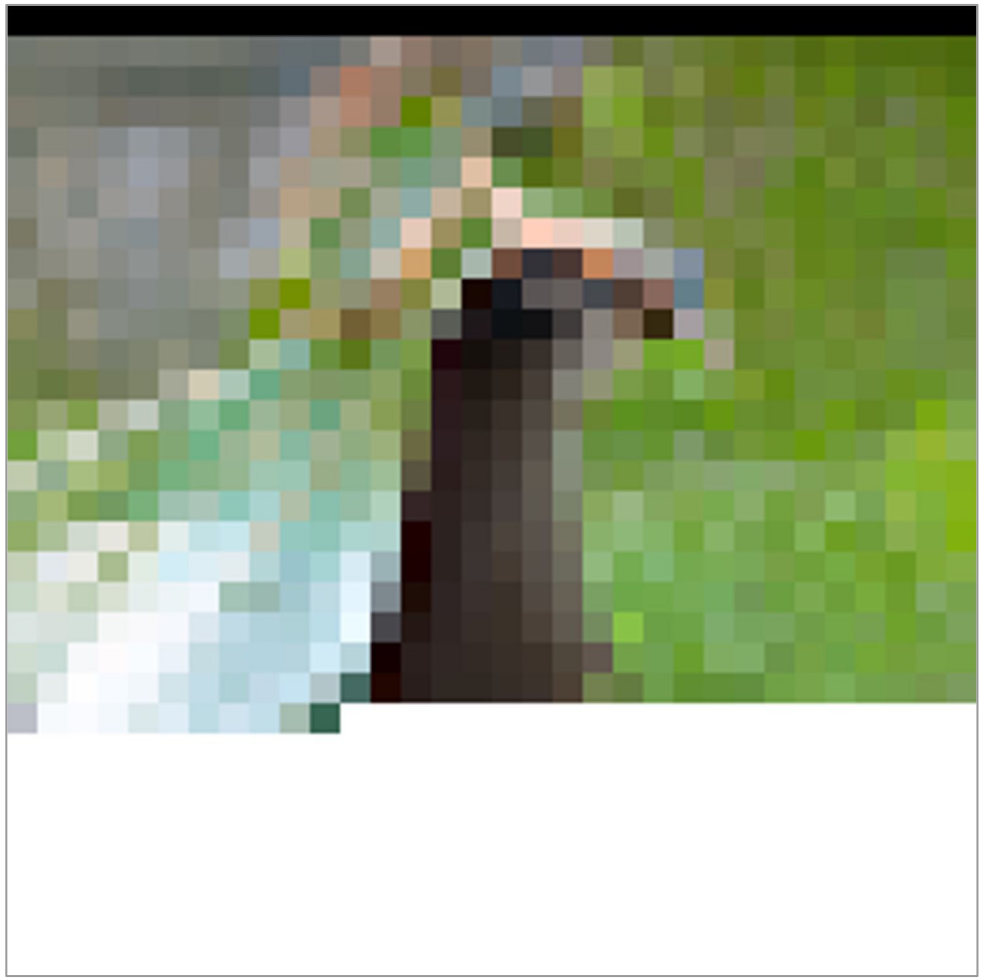
PixelCNN – Softmax Sampling

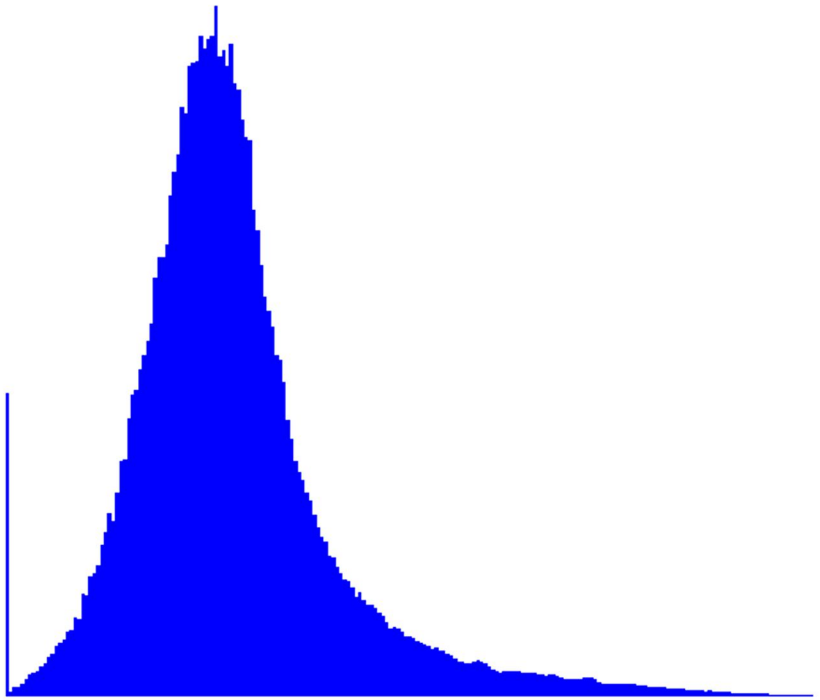




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

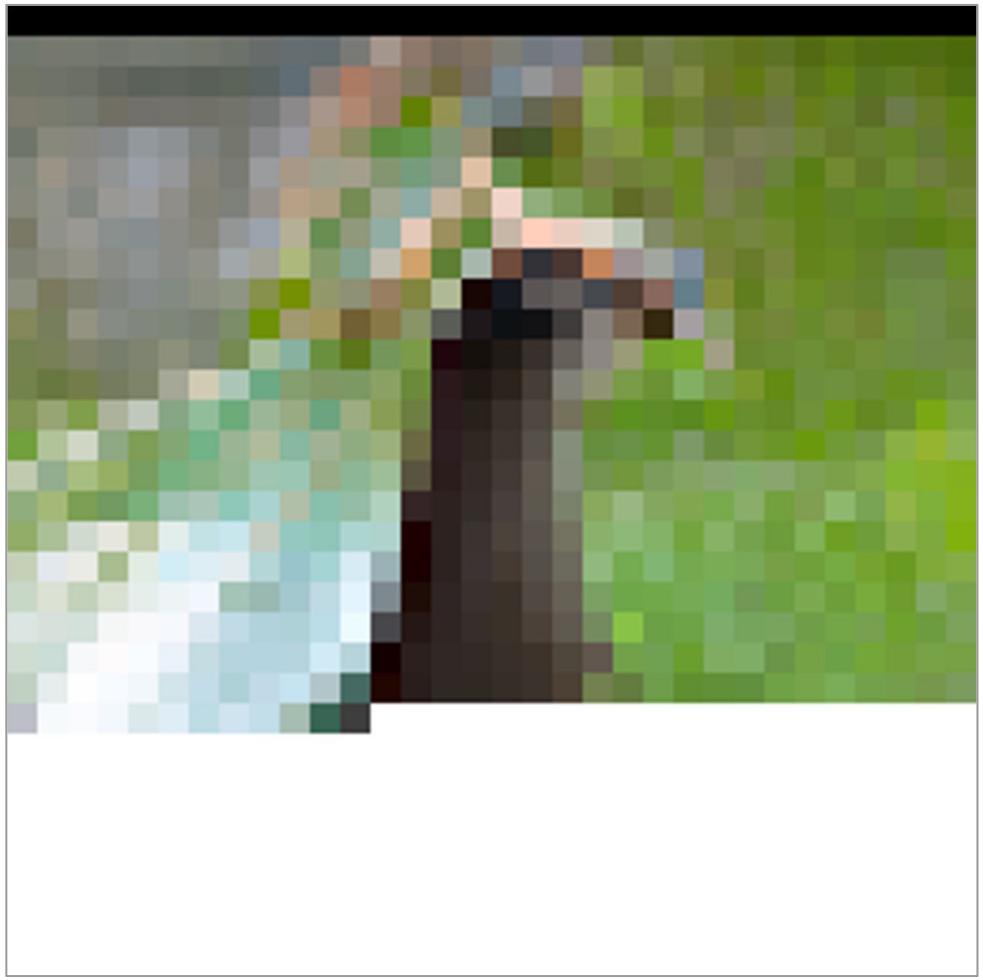
PixelCNN – Softmax Sampling

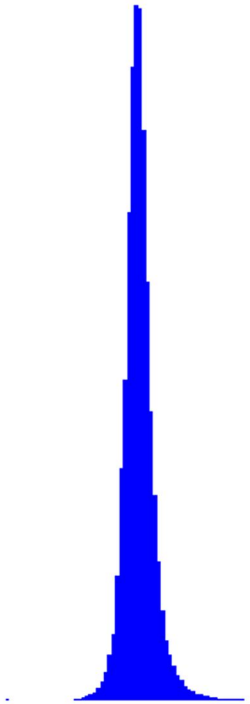




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

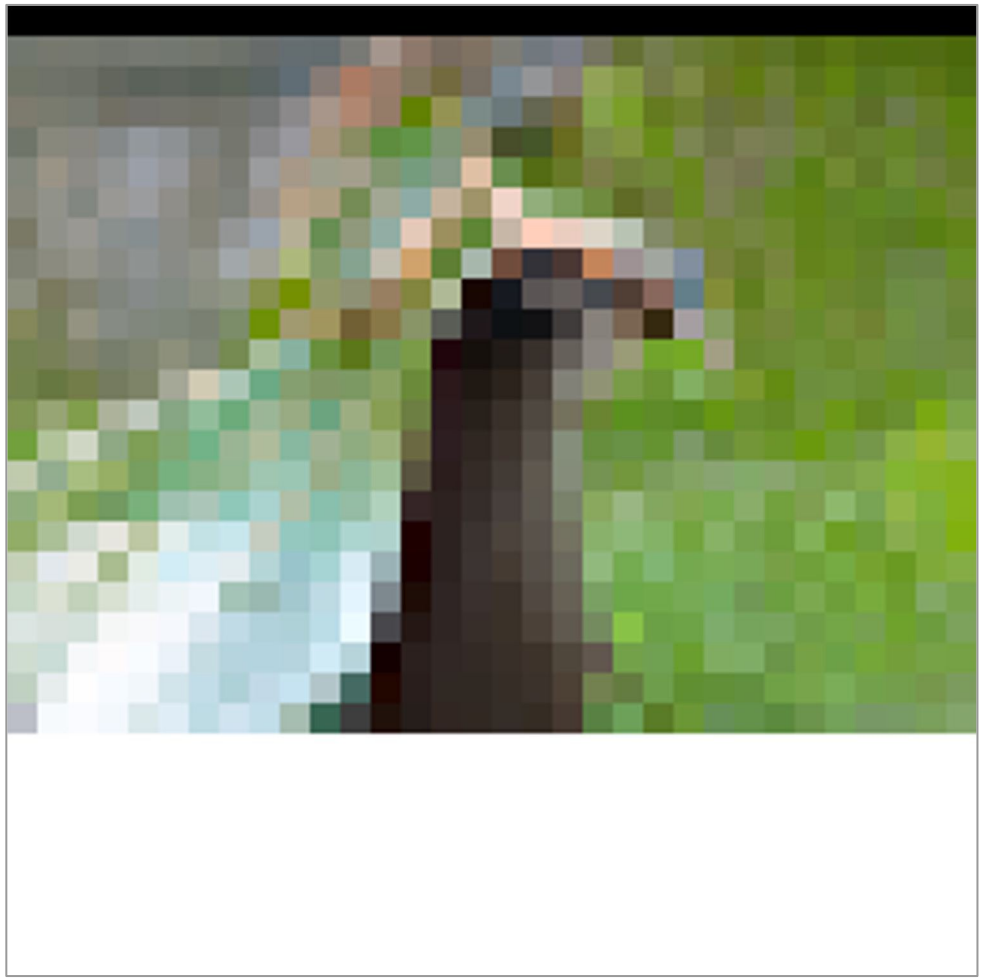
PixelCNN – Softmax Sampling

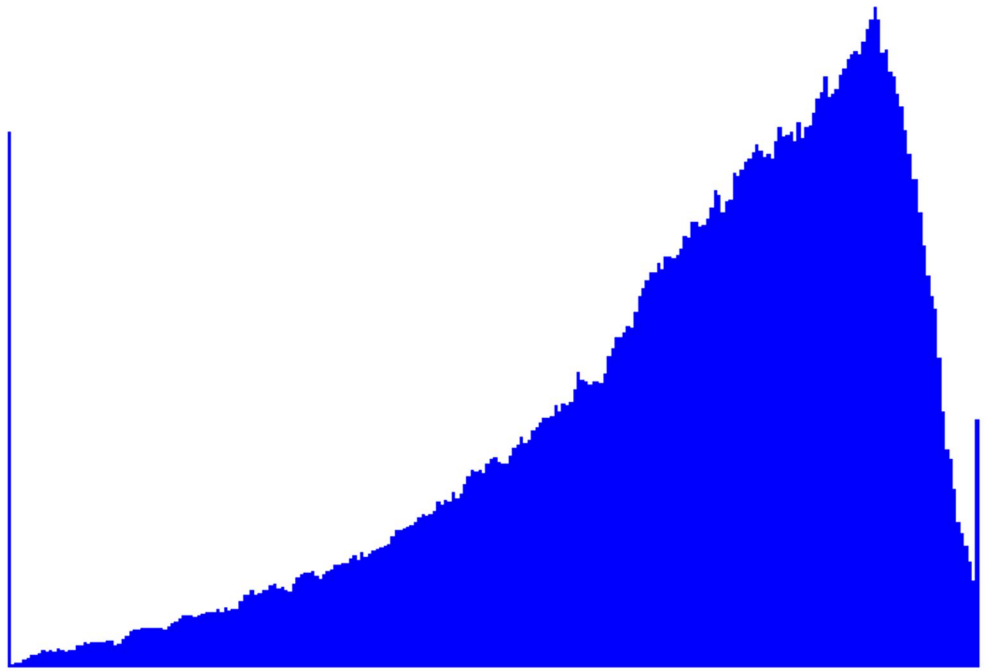




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

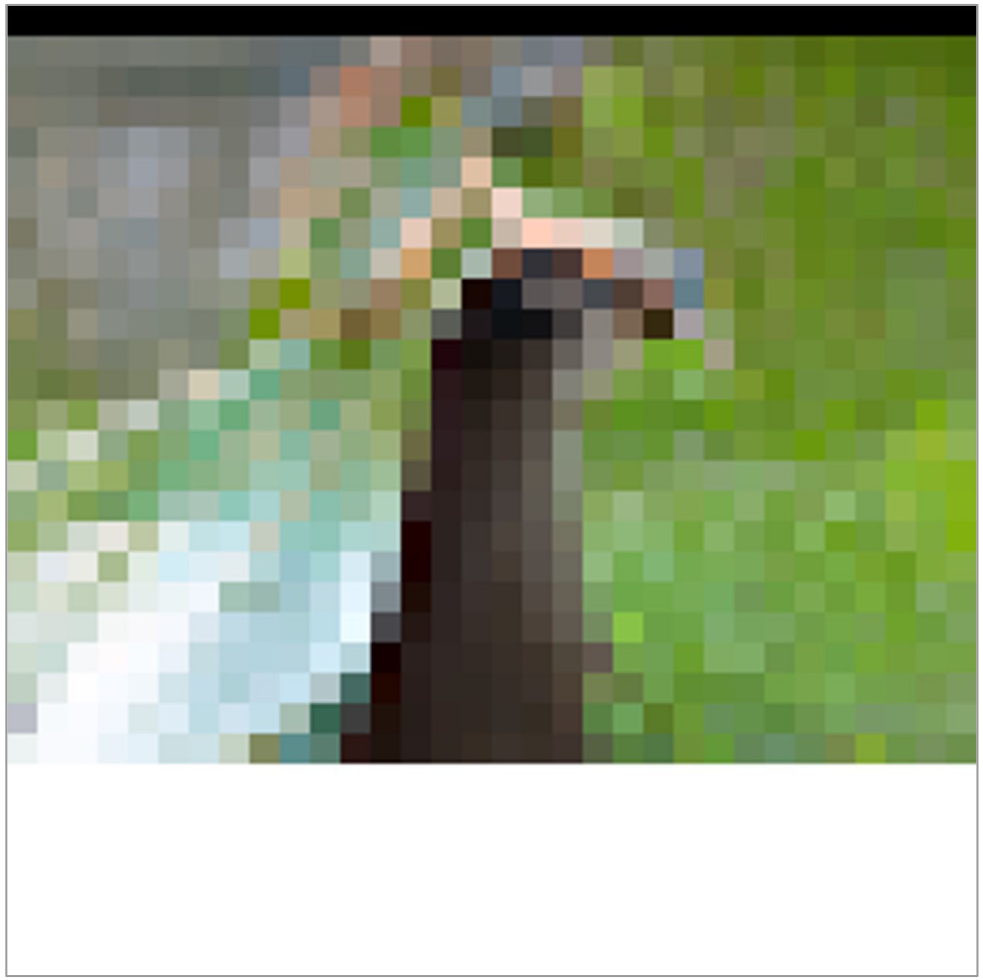
PixelCNN – Softmax Sampling

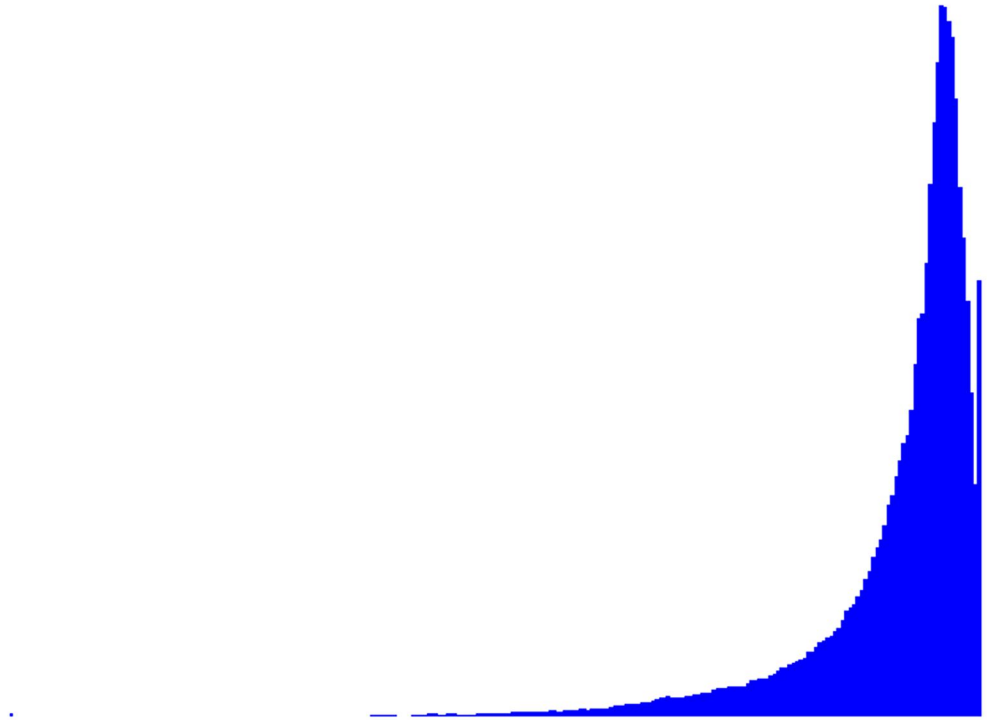




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

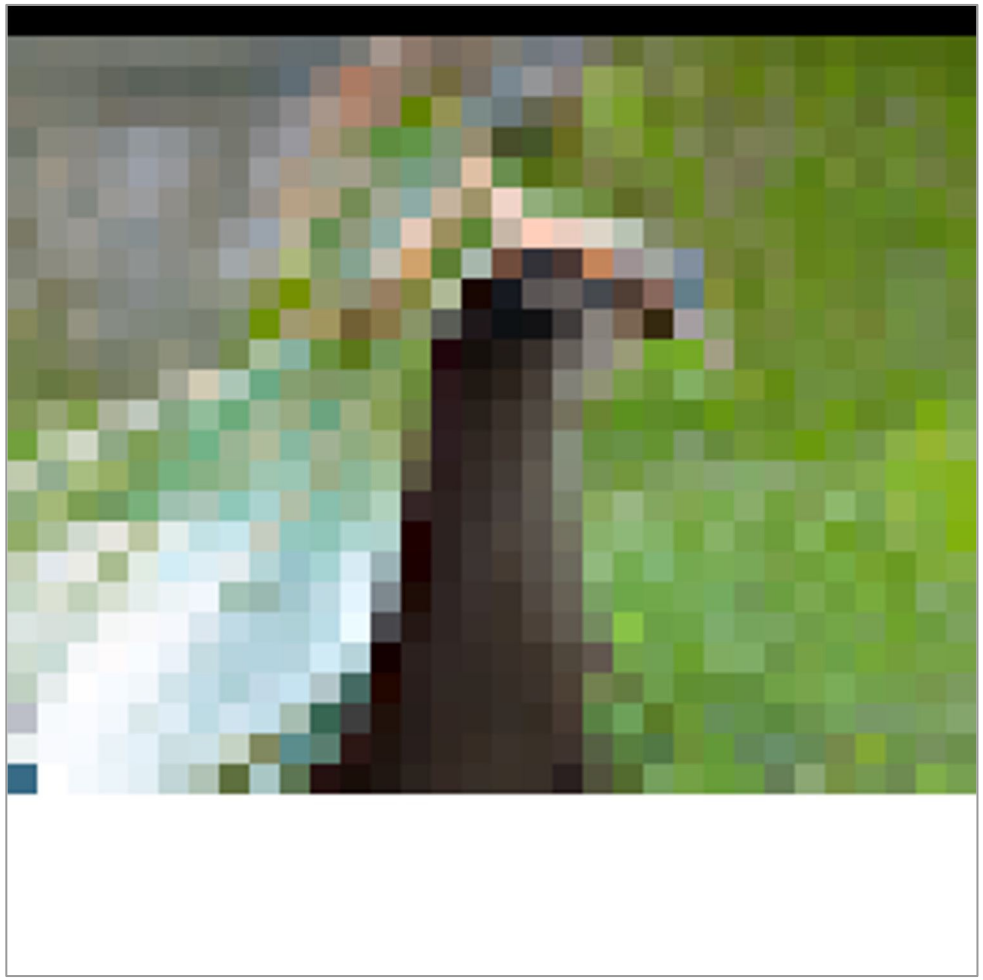
PixelCNN – Softmax Sampling

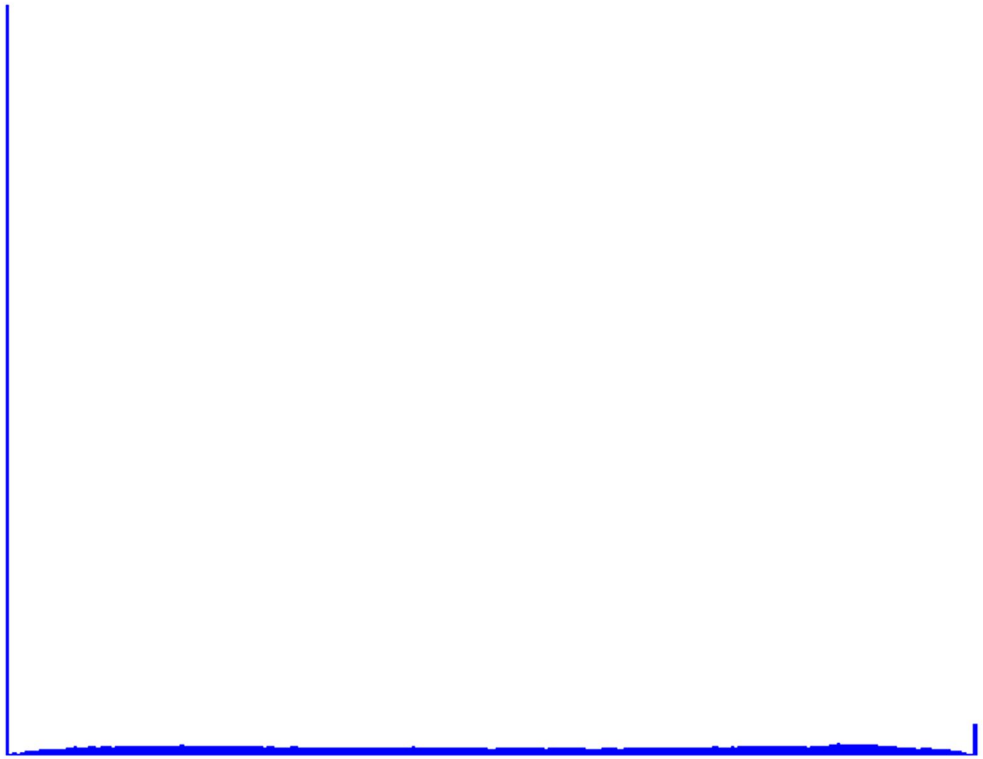




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

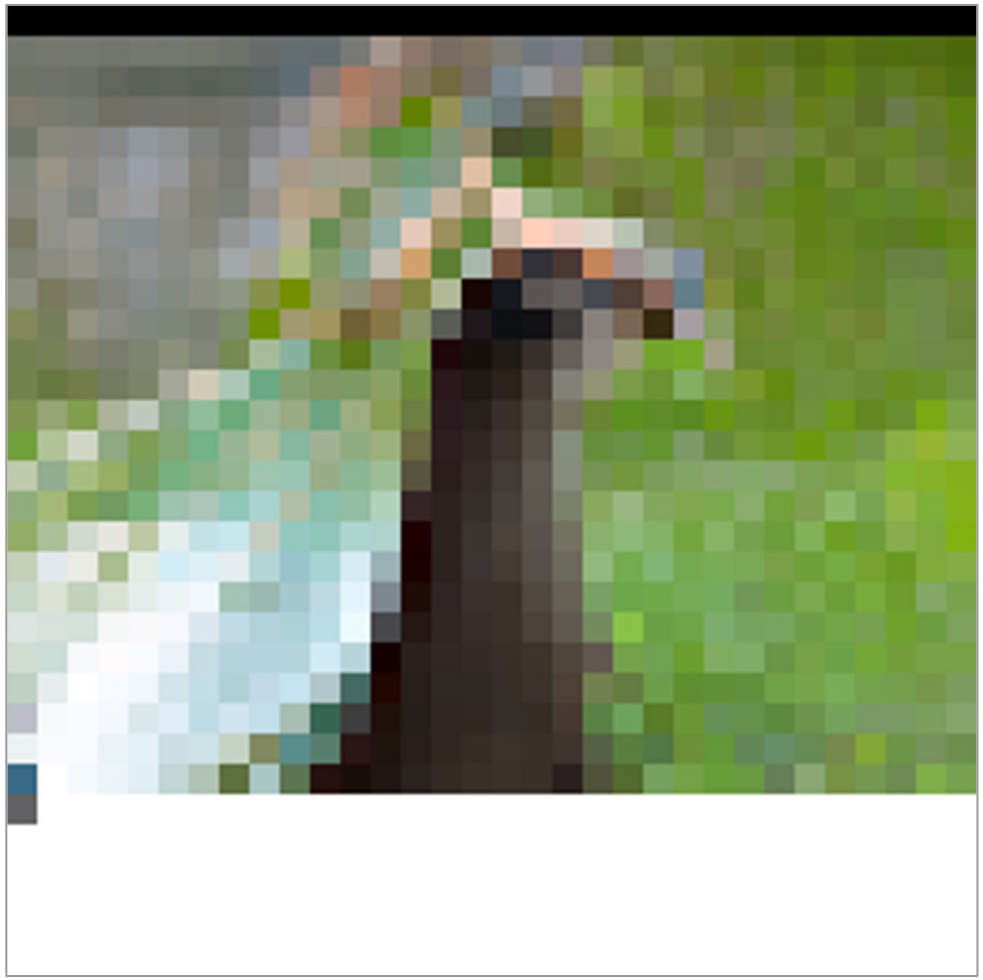
PixelCNN – Softmax Sampling

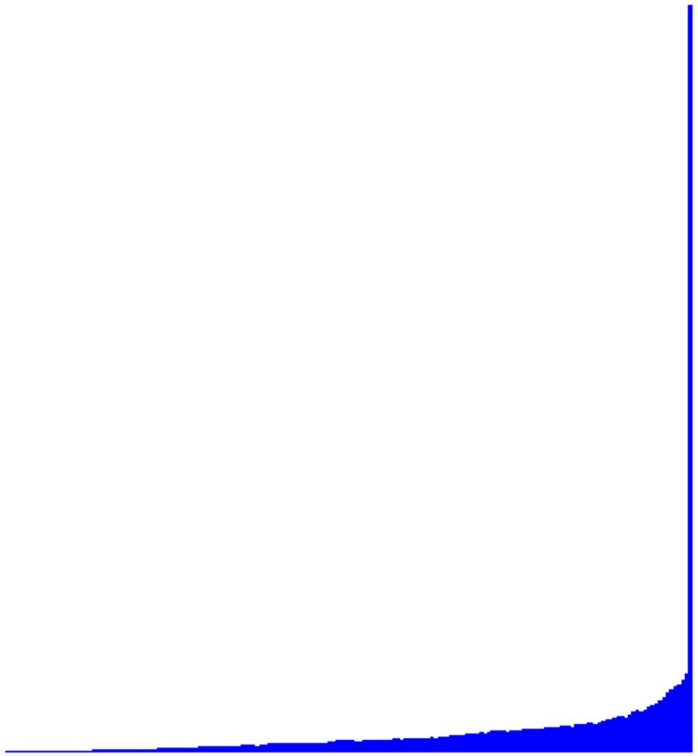




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

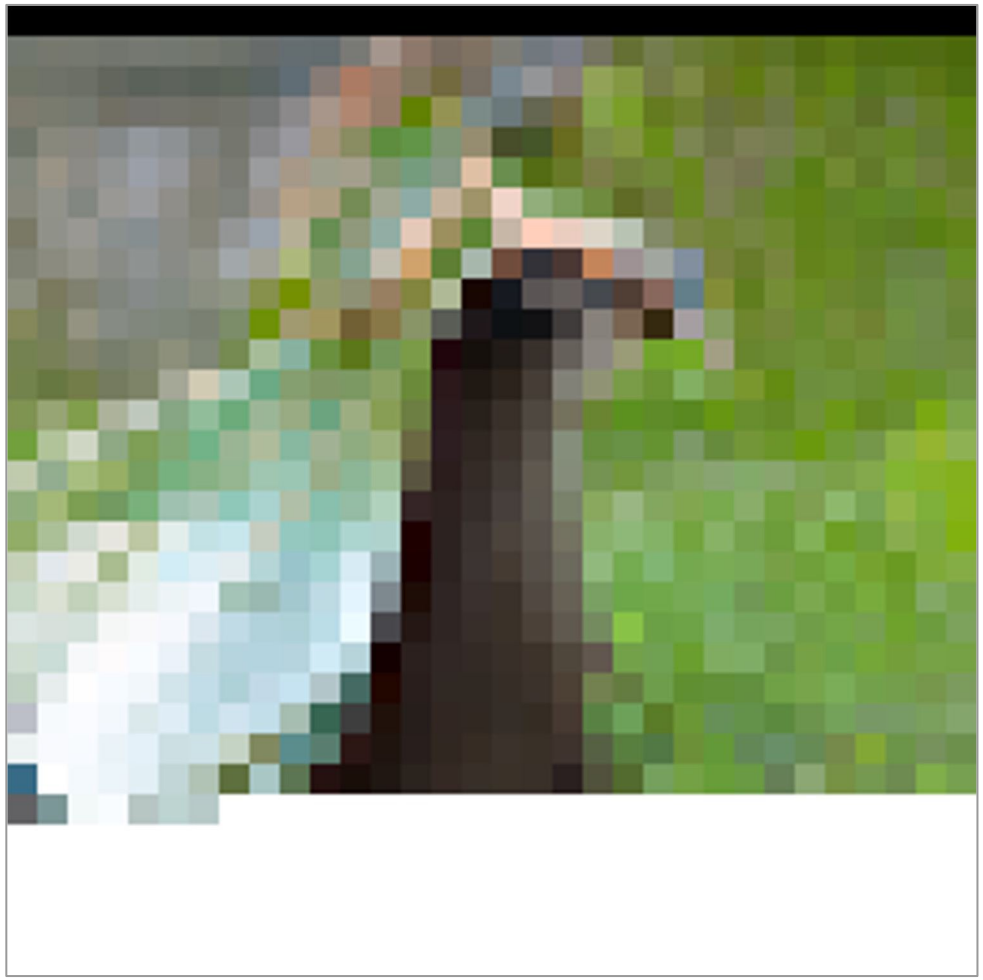
PixelCNN – Softmax Sampling

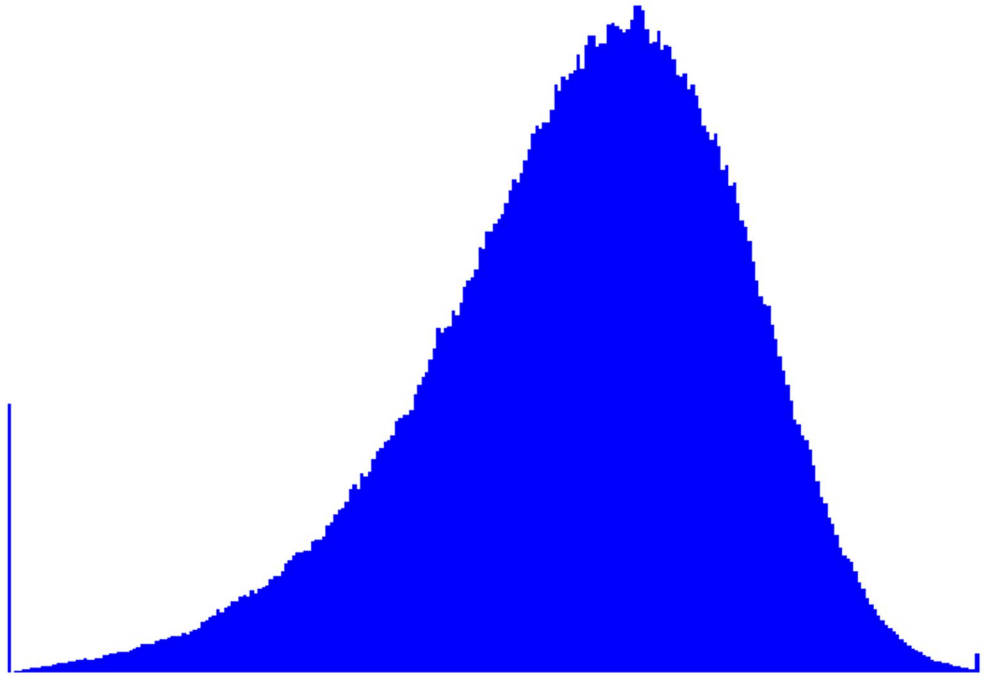




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

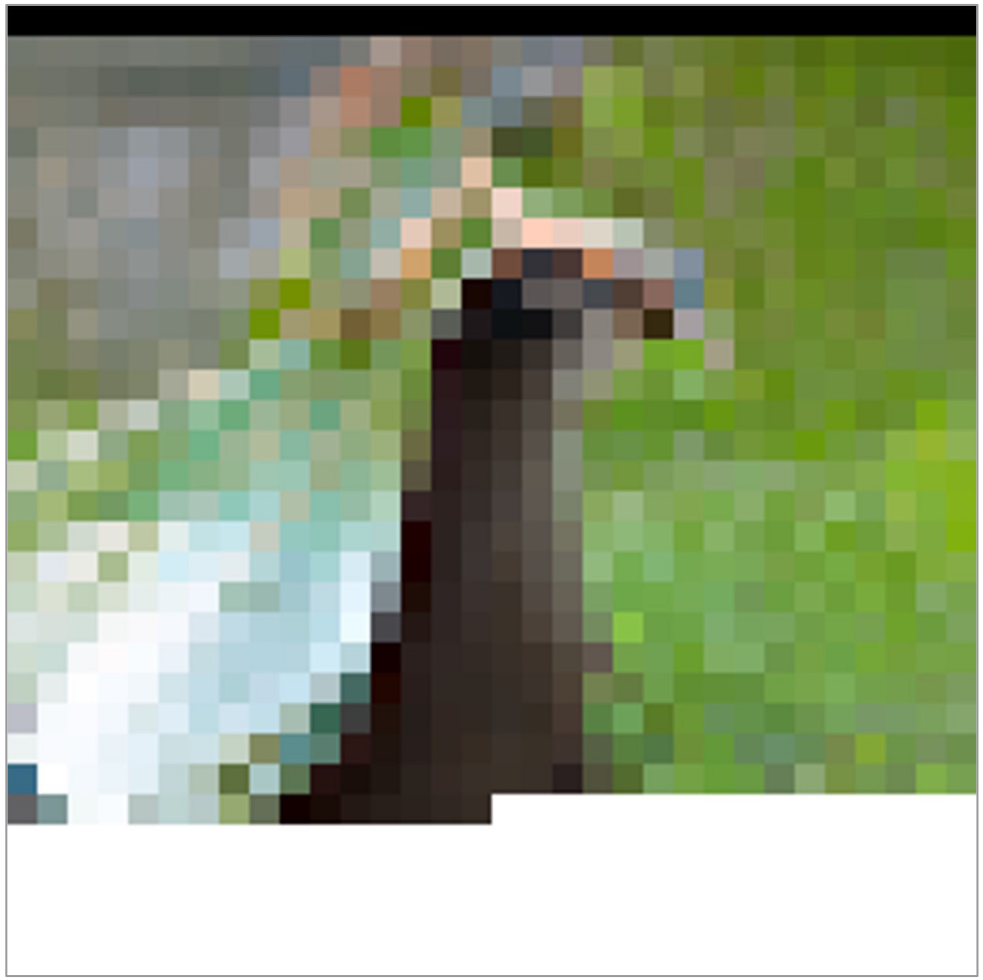
PixelCNN – Softmax Sampling

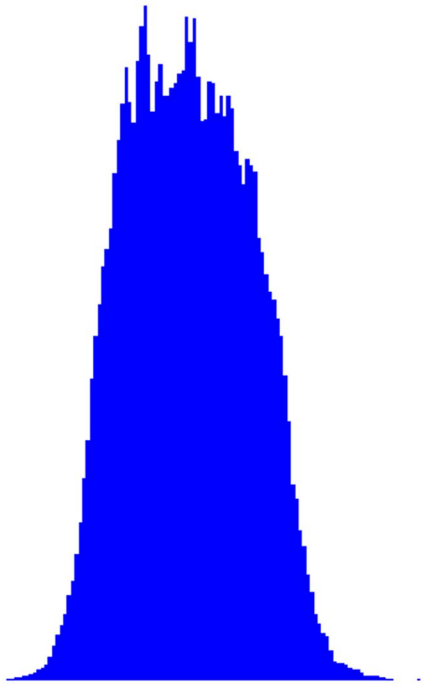




Slide adapted from
Oriol Vinyals and Navdeep Jaitly

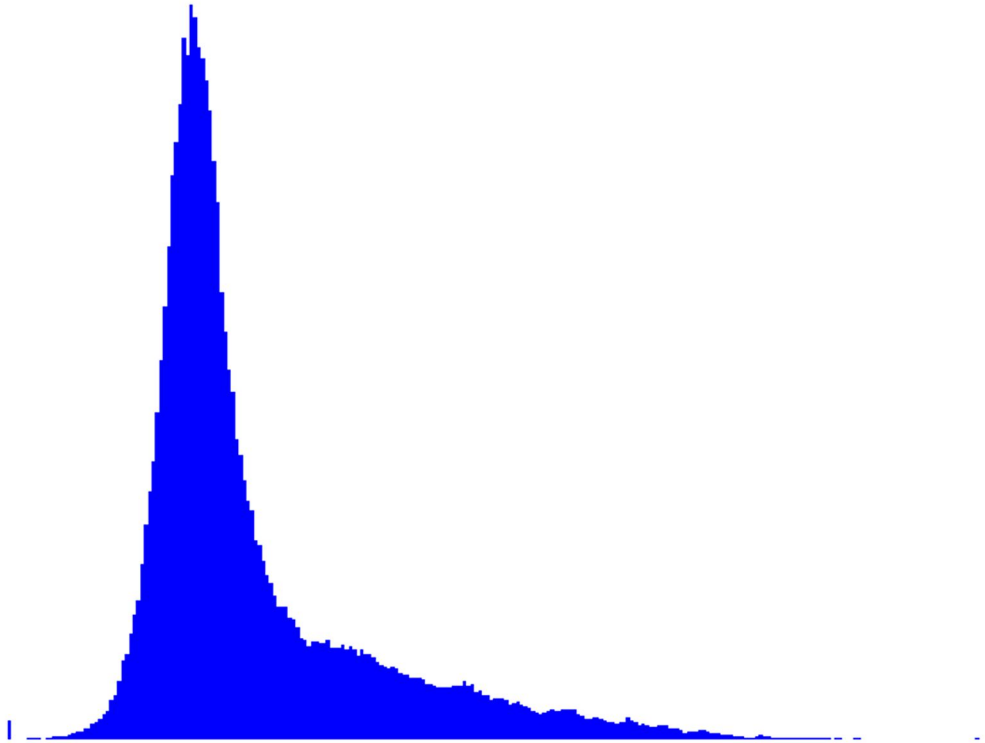
PixelCNN – Softmax Sampling





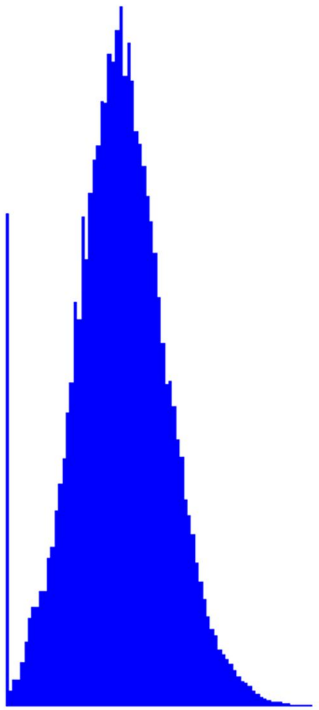
Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN – Softmax Sampling



Slide adapted from
Oriol Vinyals and Navdeep Jaitly

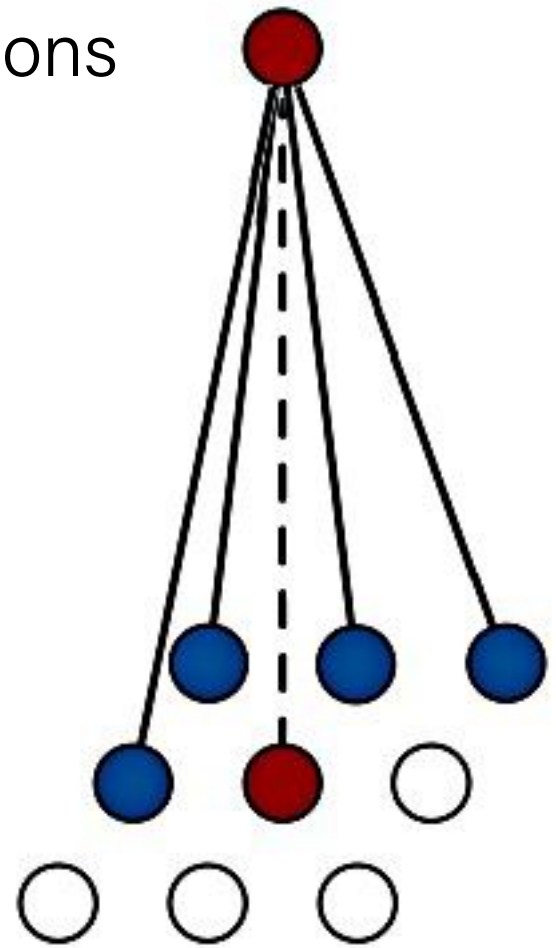
PixelCNN – Softmax Sampling

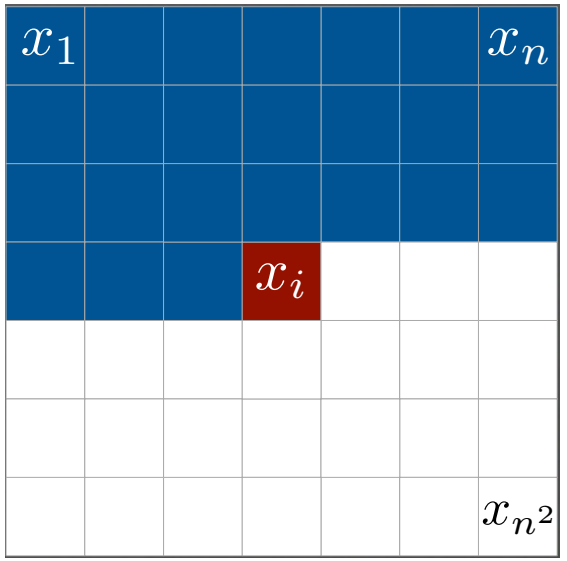


Slide adapted from
Oriol Vinyals and Navdeep Jaitly

PixelCNN

use masked convolutions to enforce the autoregressive relationship



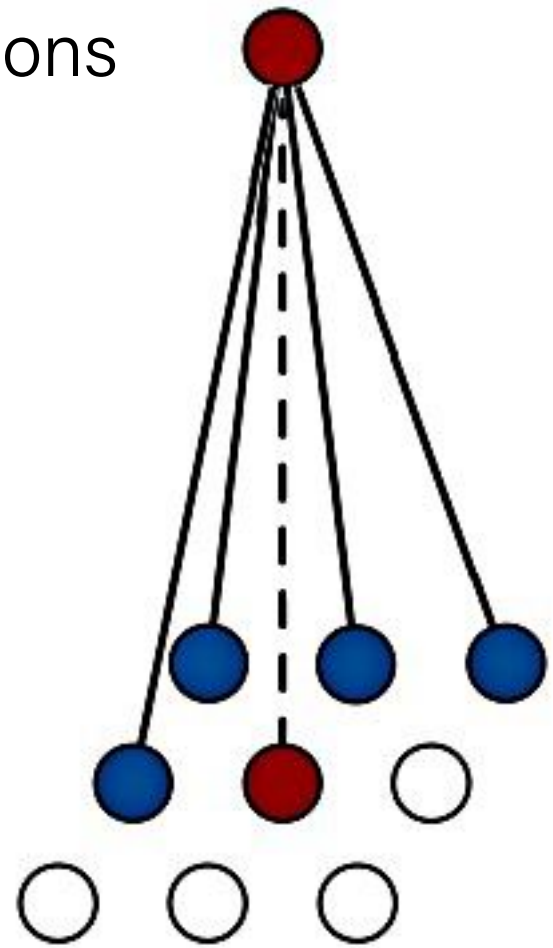


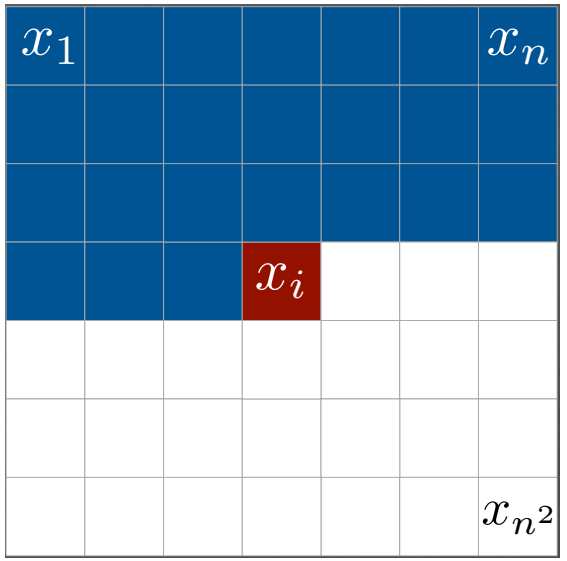
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

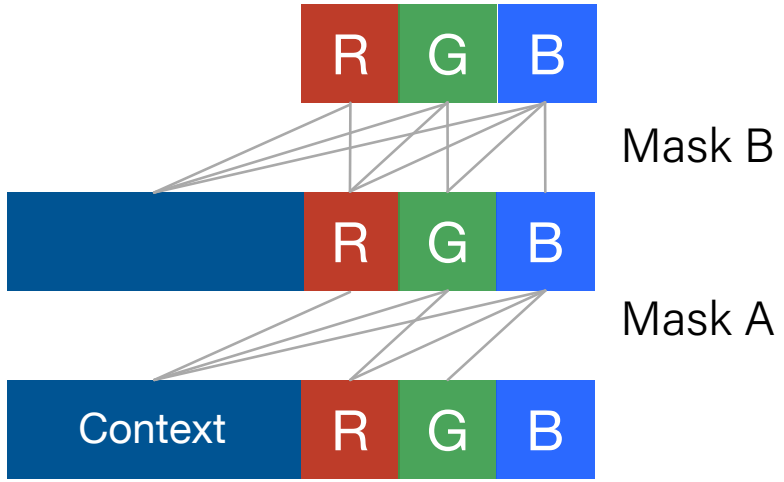
Slide adapted from Nal Kalchbrenner

PixelCNN

use masked convolutions to enforce the autoregressive relationship







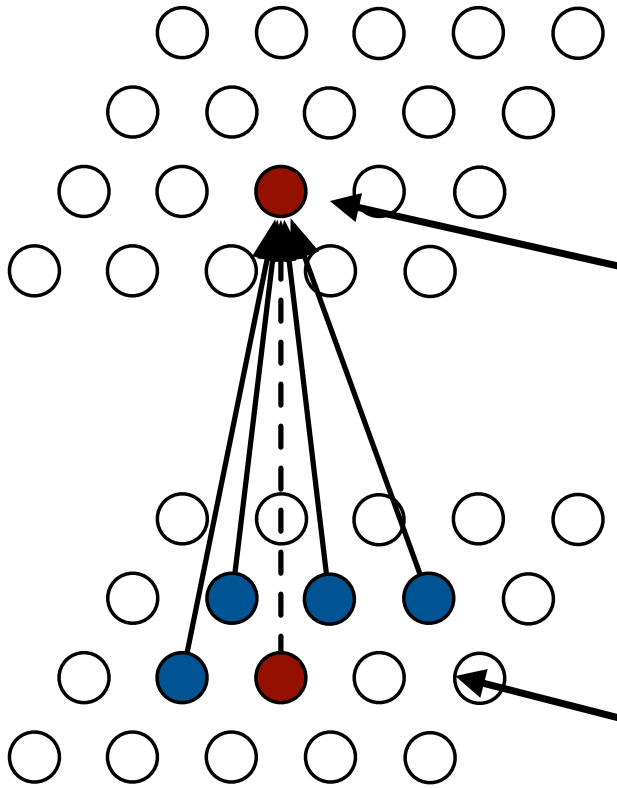
autoregressive over color channels

$$p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i})$$

Slide adapted from Nal Kalchbrenner

PixelCNN

Multiple layers of masked convolutions



composing multiple layers increases the context size

only depends on pixel above and to the left

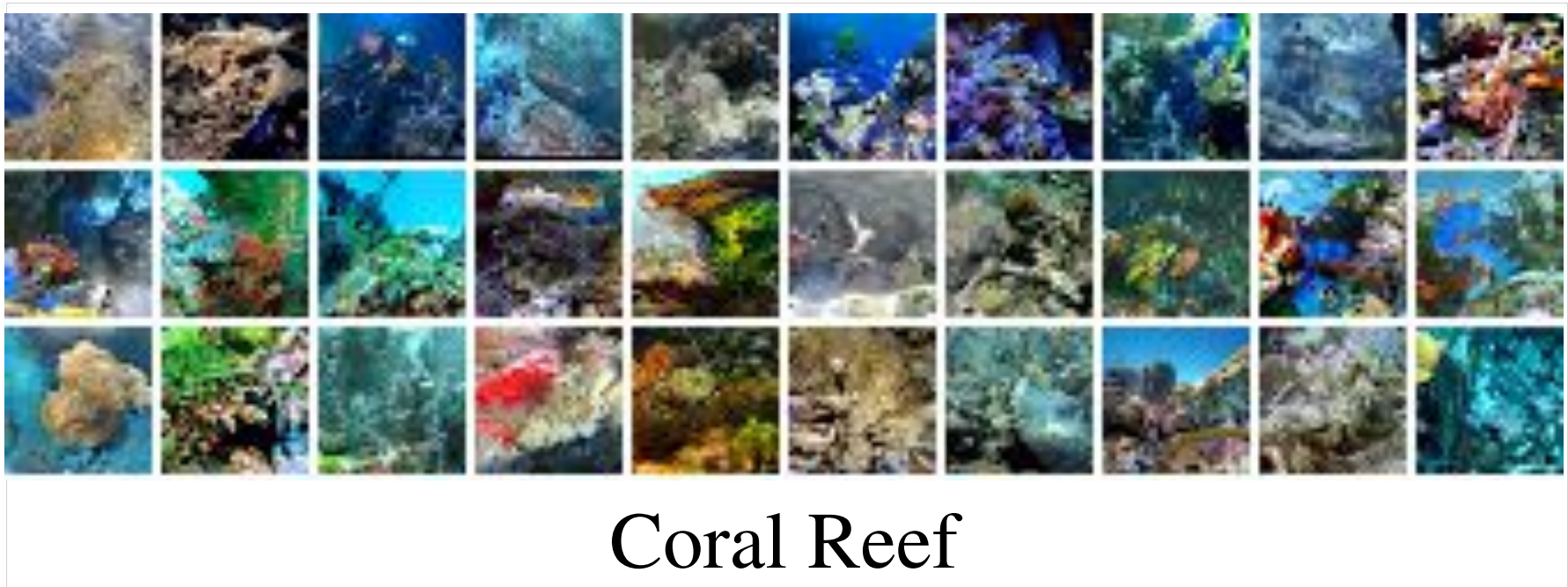
masked convolution

Slide adapted from Aaron Courville

Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN



Slide credit: Nal Kalchbrenner

Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN

Sorrel horse

Slide credit: Nal Kalchbrenner

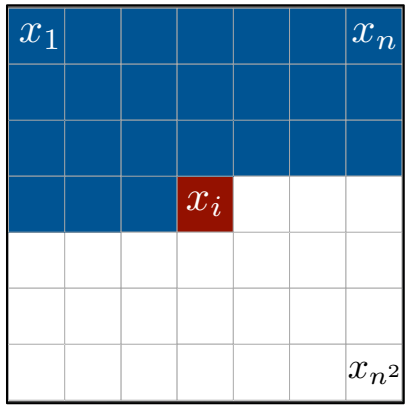
Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN

Slide credit: Nal Kalchbrenner

Autoregressive Models



- Explicitly model conditional probabilities:

$$p_{\text{model}}(\boldsymbol{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

Each conditional can be a complicated neural net

Advantages:

- $p_{\text{model}}(x)$ is tractable (easy to train and sampl

Disadvantages:

- Generation can be too costly
- Generation can not be controlled by a latent code

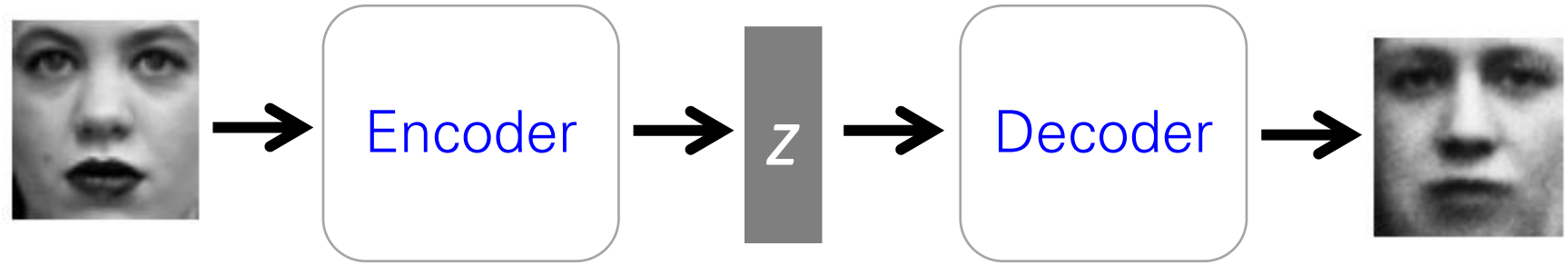
PixelCNN elephants
(van den Ord et al. 2016)

Slide adapted from Ian Goodfellow

Variational Autoencoders

Traditional Autoencoders

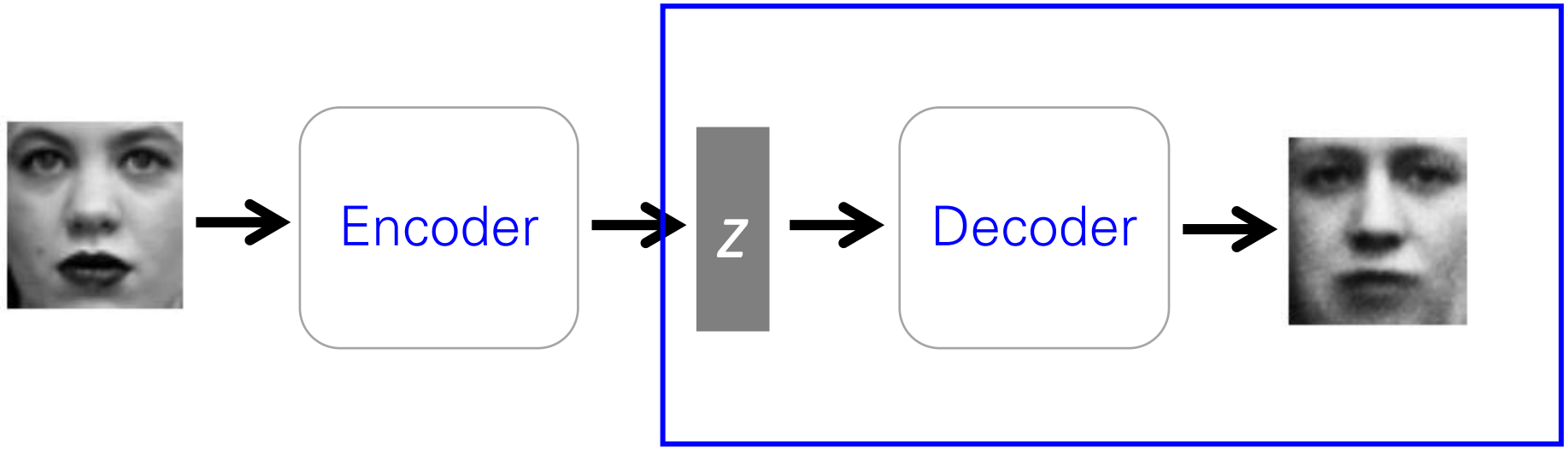
- In traditional autoencoders, we can think of encoder and decoders as some function mapping.



$$z = h(x) \qquad \hat{x} = f(z)$$

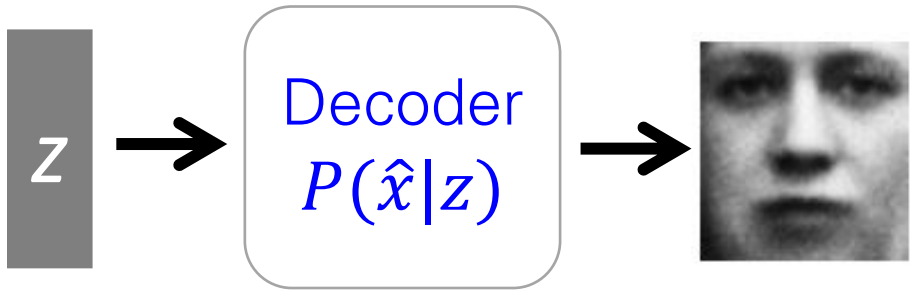
Variational Autoencoders

- To go to variational autoencoders, we need to first add some **stochasticity** and think of it as a probabilistic modeling.



Variational Autoencoders

Sample from g(z)
e.g. Standard Gaussian



$$z \sim g(z) \qquad \hat{x} = f(z) \qquad \hat{x} \sim P(x|z)$$

Variational Autoencoders

Encoder

z

Traditional AE

Encoder

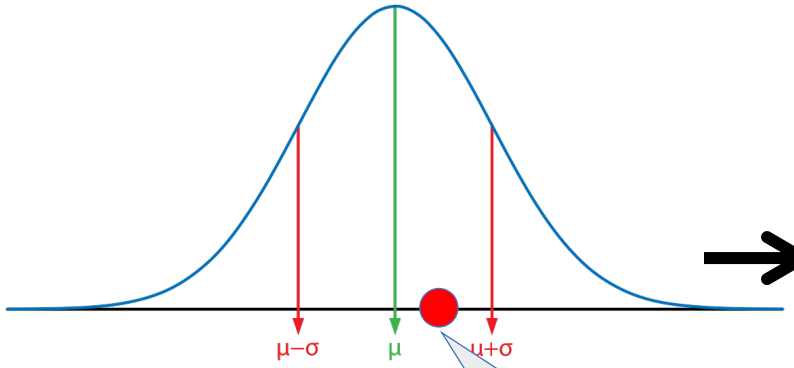
z_1

z_2

Variational AE

Consider this to be the mean of a normal μ

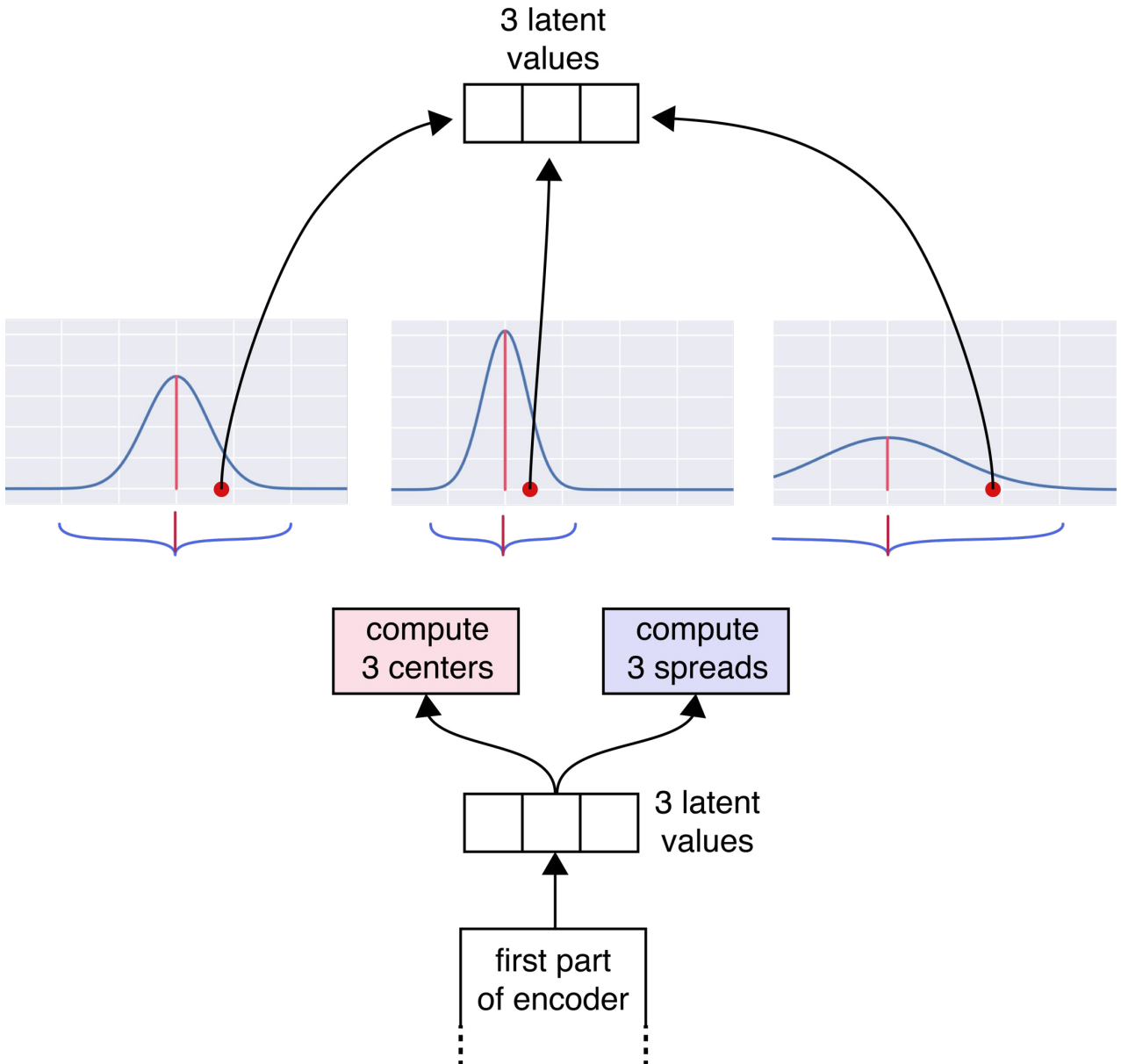
Consider this to be the std of a normal σ



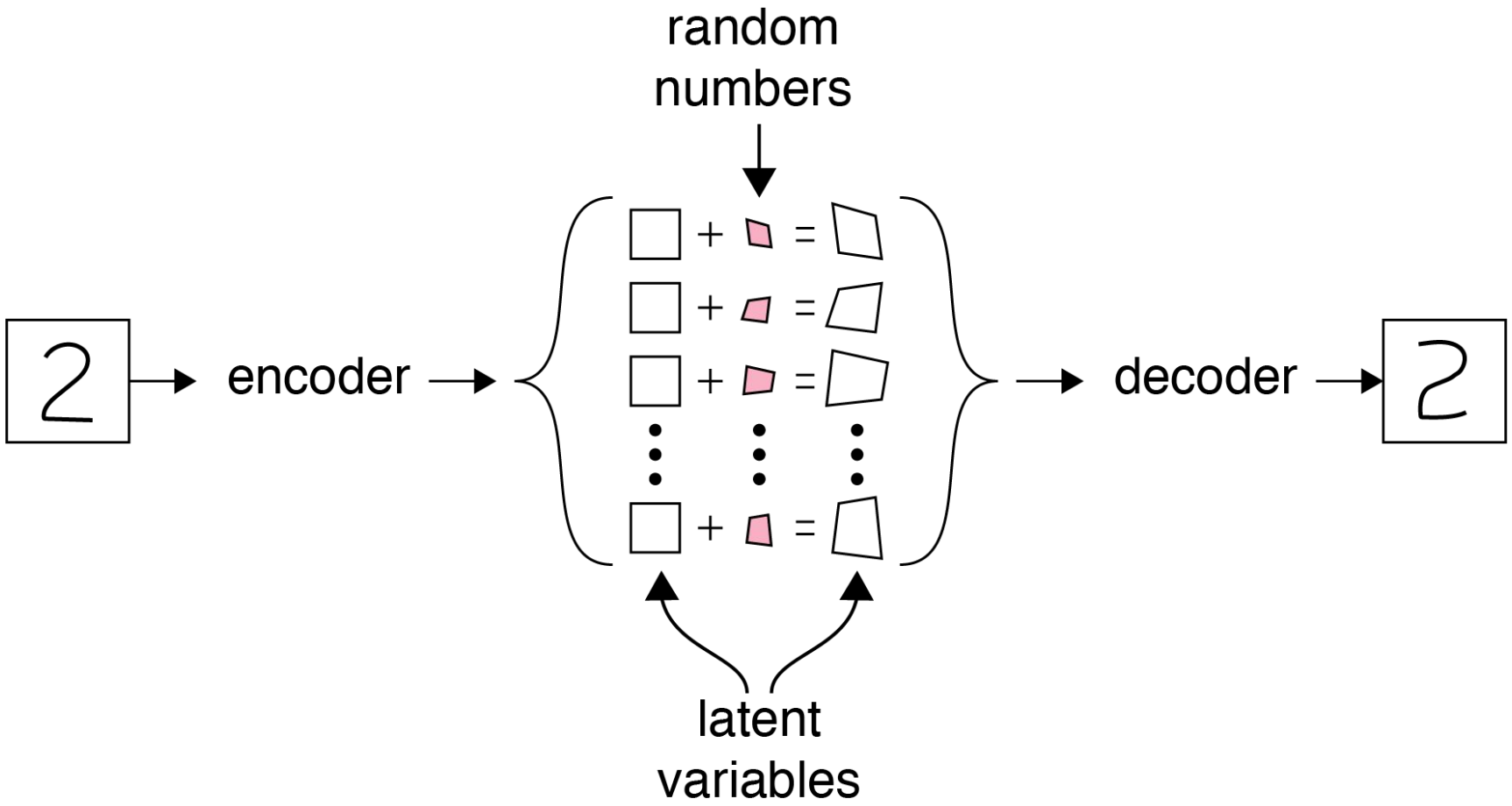
Decode

Randomly chosen value Latent value, z

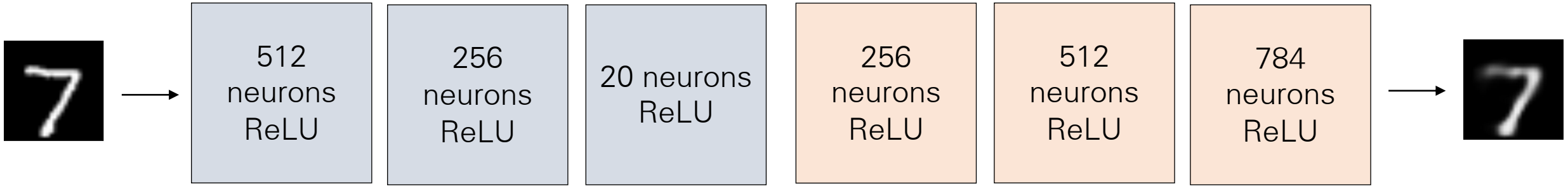
Variational Autoencoders

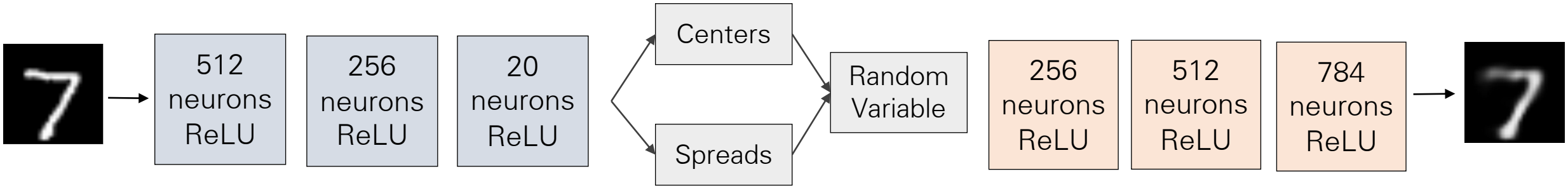


Variational Autoencoders



Variational Autoencoders





Latent Variable Models

- Variational Autoencoder (VAE) model [Kingma and Welling, 2014] [Rezende et al., 2014]

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

z: latent variable

x: observed data

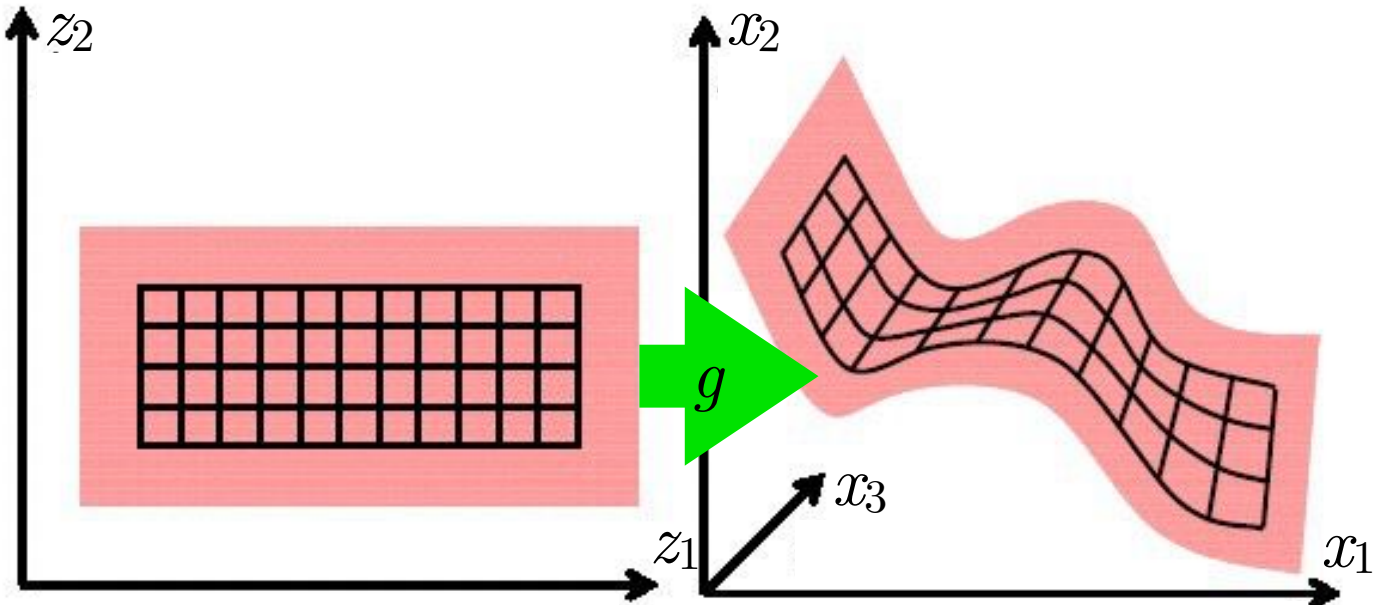
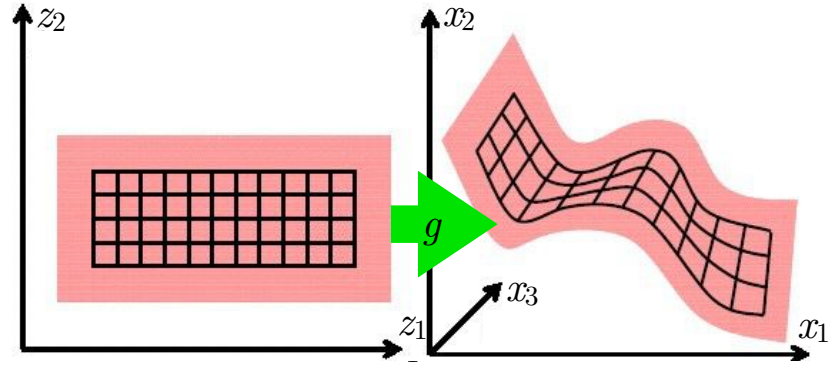


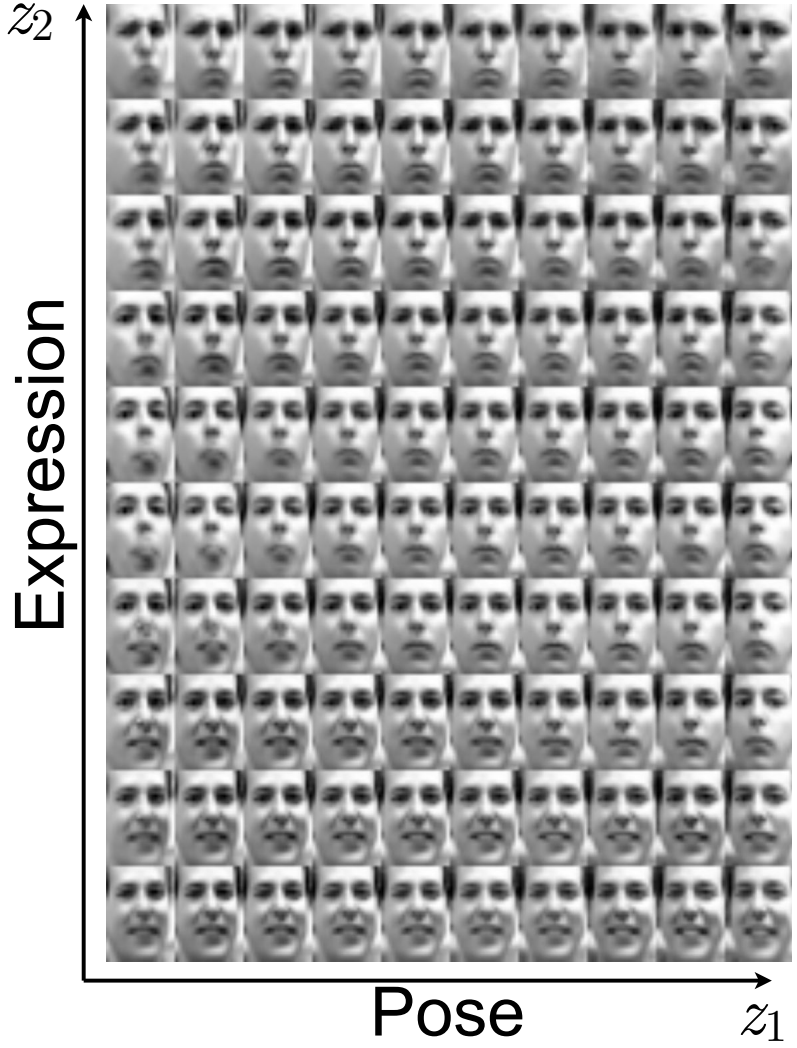
Image credit: Ward and Hamarneh

Slide credit: Aaron Courville

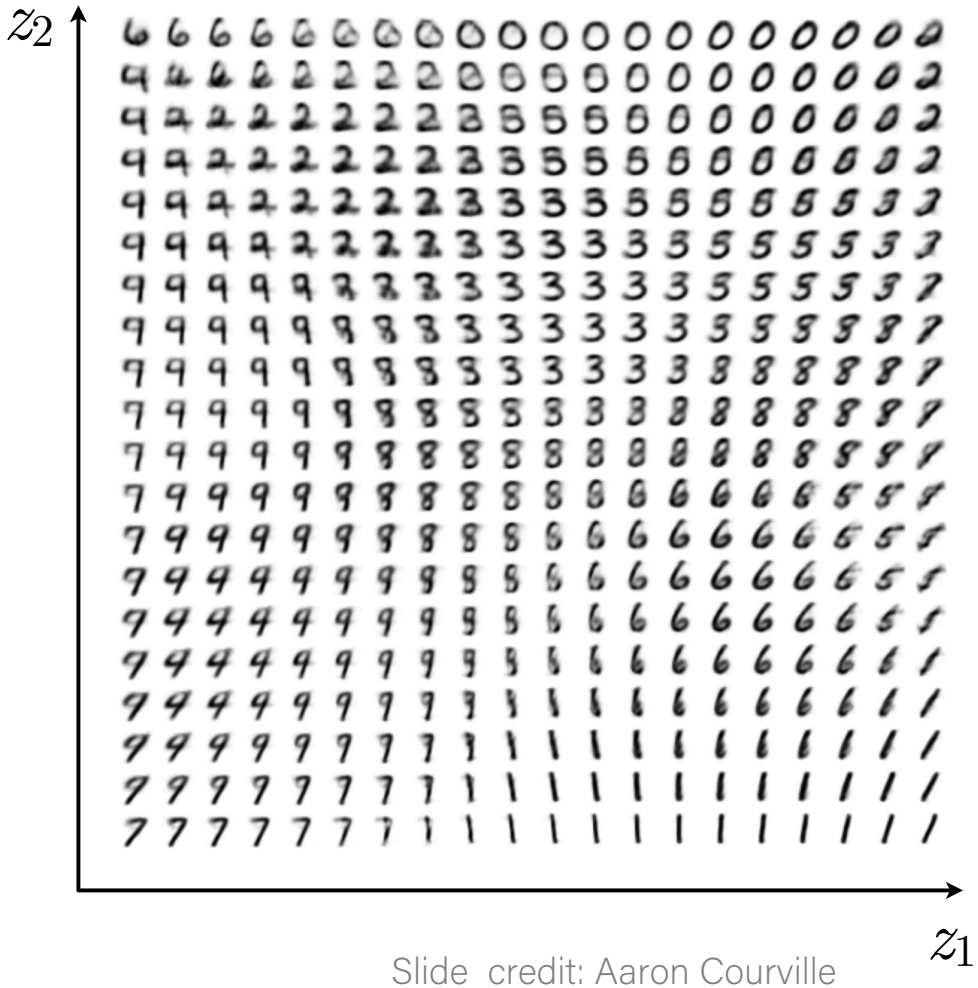
Latent Variable Models



Frey Face dataset:



MNIST:



Slide credit: Aaron Courville

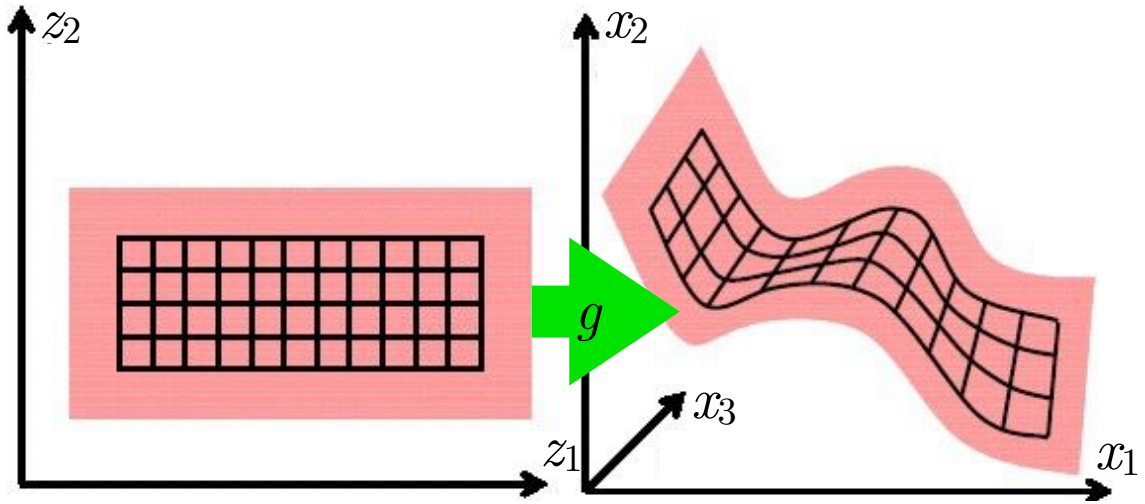
Latent Variable Models

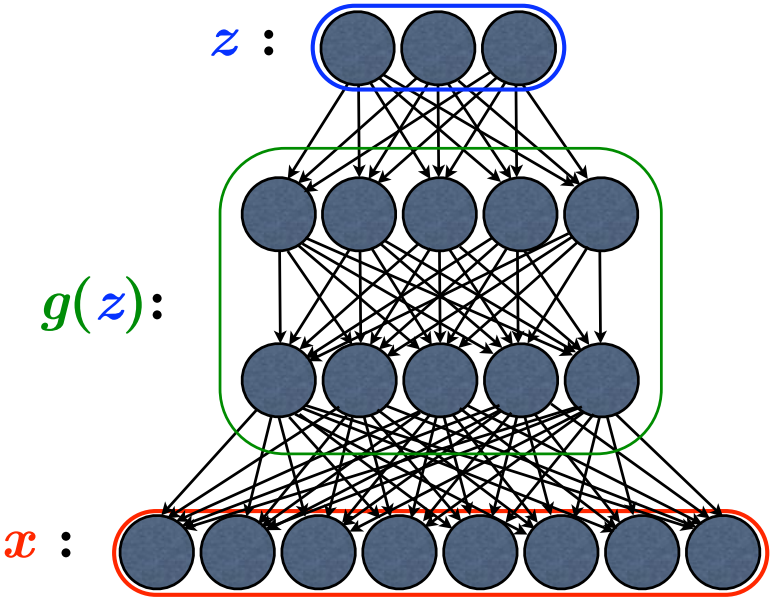
- learn a mapping from some latent variable z to a complicated distribution on x.

$$p(x) = \int p(x, z)\, dz \qquad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = g(\boldsymbol{z})$$

- Can we learn to decouple the true explanatory factors underlying the data distribution? e.g. separate identity and expression in face images



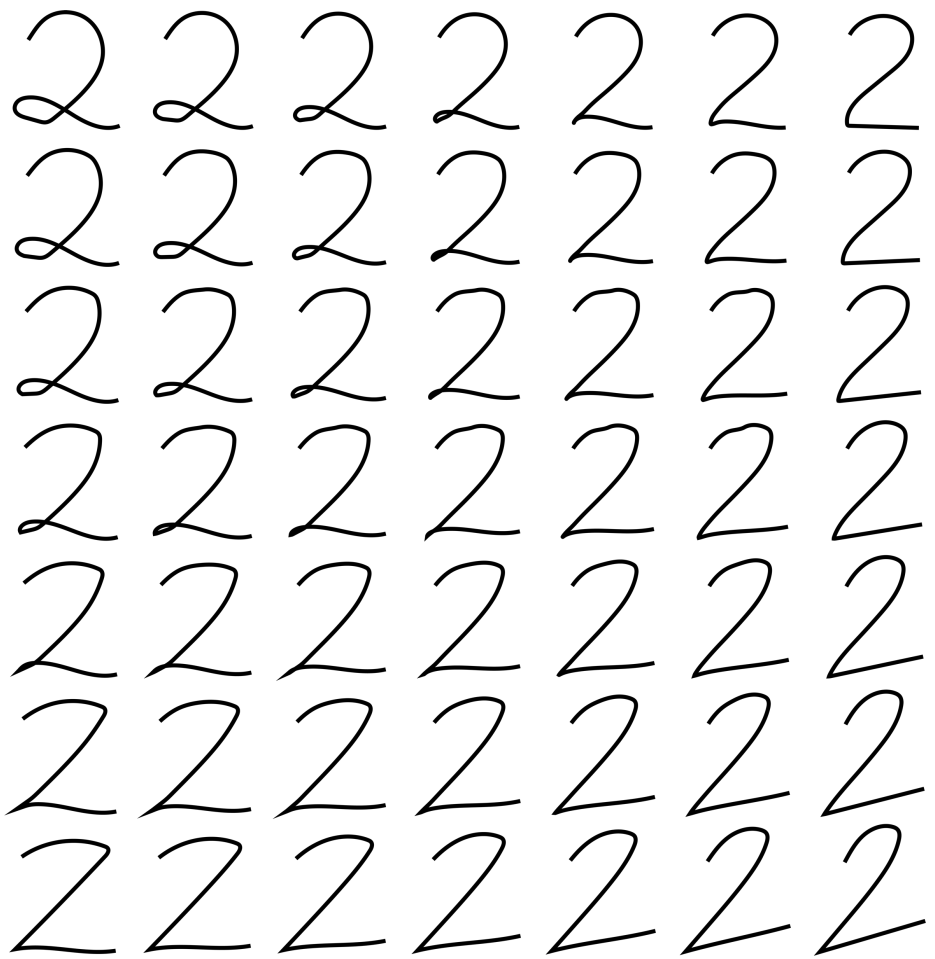


Slide credit:
Aaron Courville

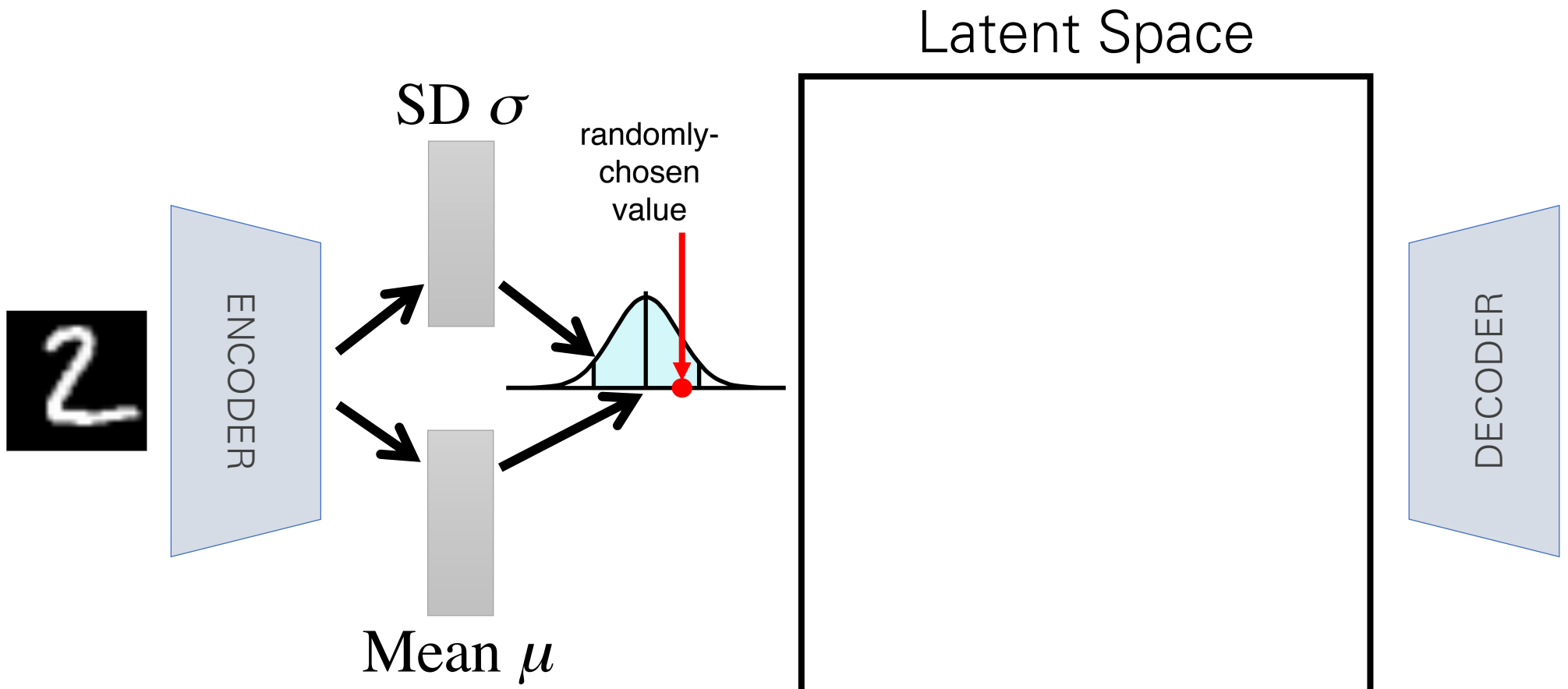
Image credit:
Ward and Hamarneh

Separability in Variational Autoencoders

- Separability is not only between classes but we also want similar items in the same class to be near each other.
- For example, there are different ways of writing “2”, we want similar styles to end up near each other.
- Let’s examine VAE, there is something magic happening once we add stochasticity in the latent space.

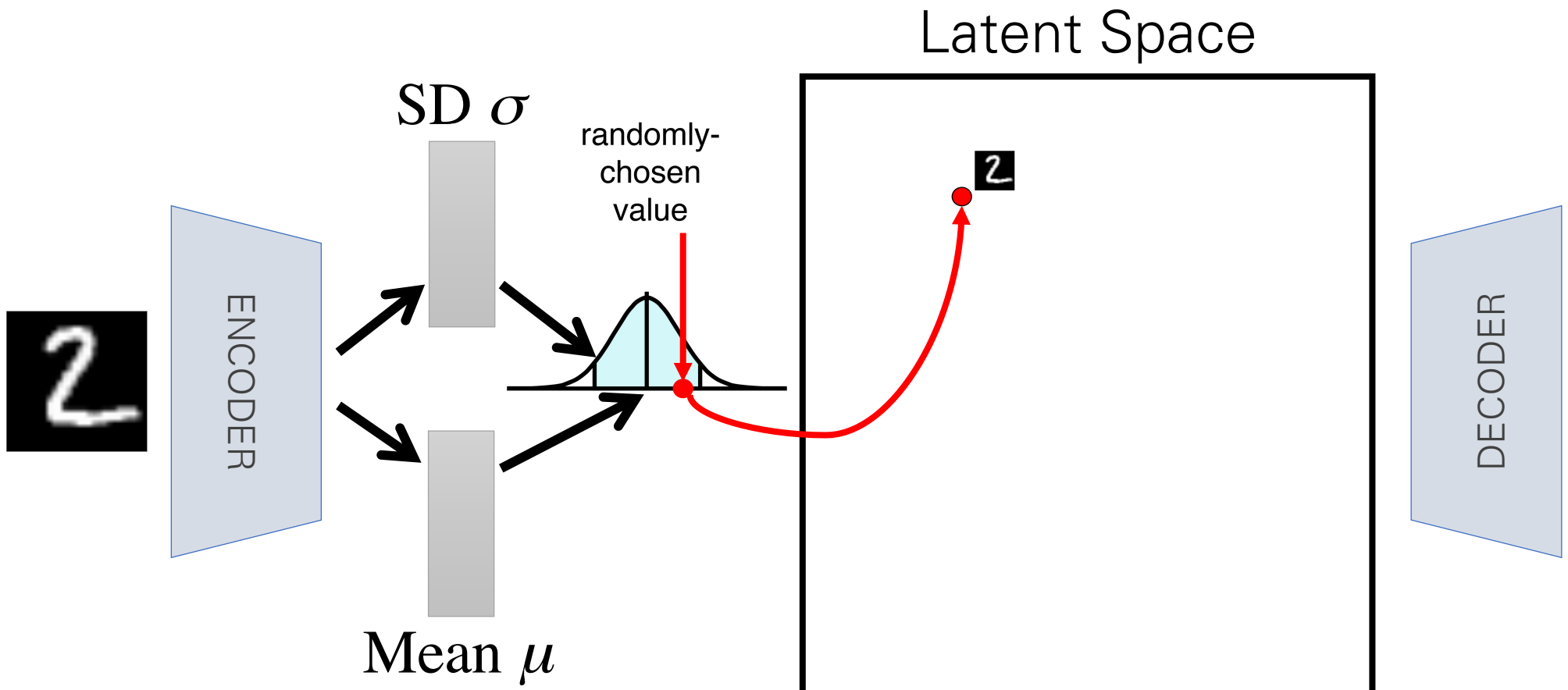


Separability in Variational Autoencoders



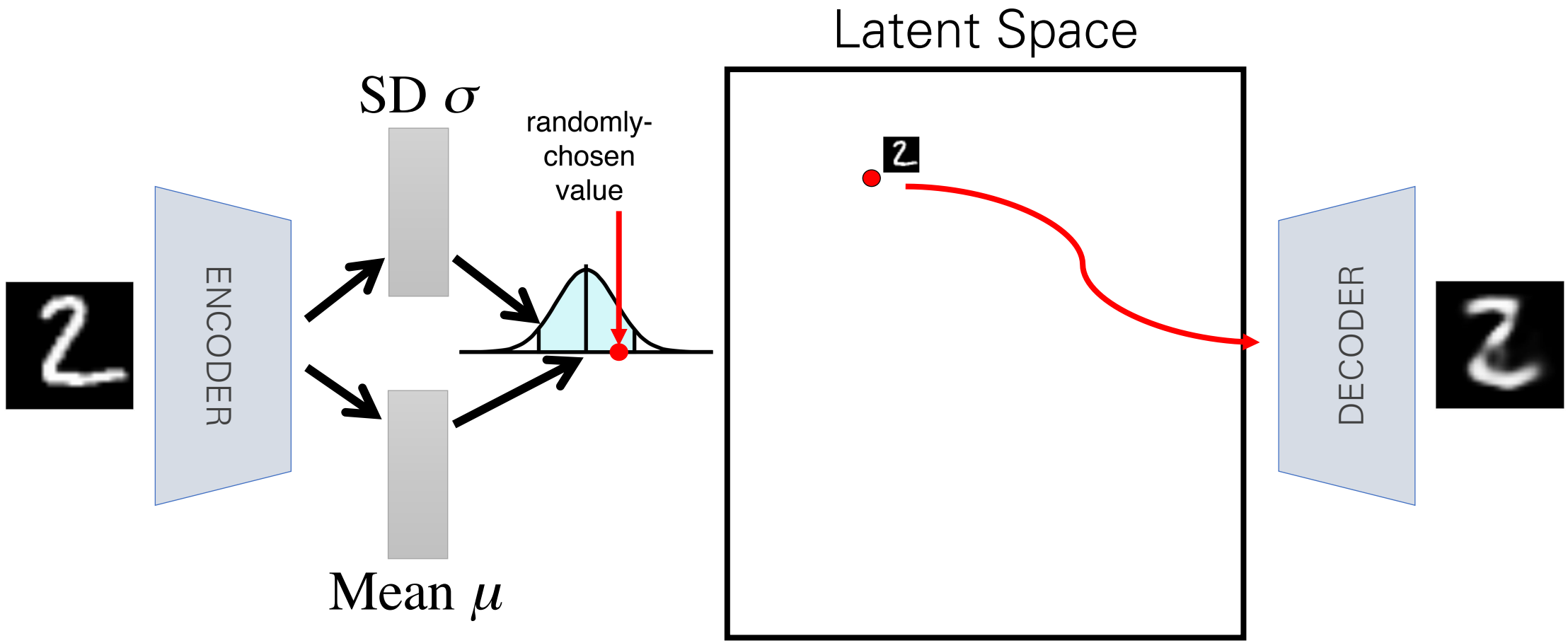
Encode the first sample (a "2") and find μ_1, σ_1

Separability in Variational Autoencoders



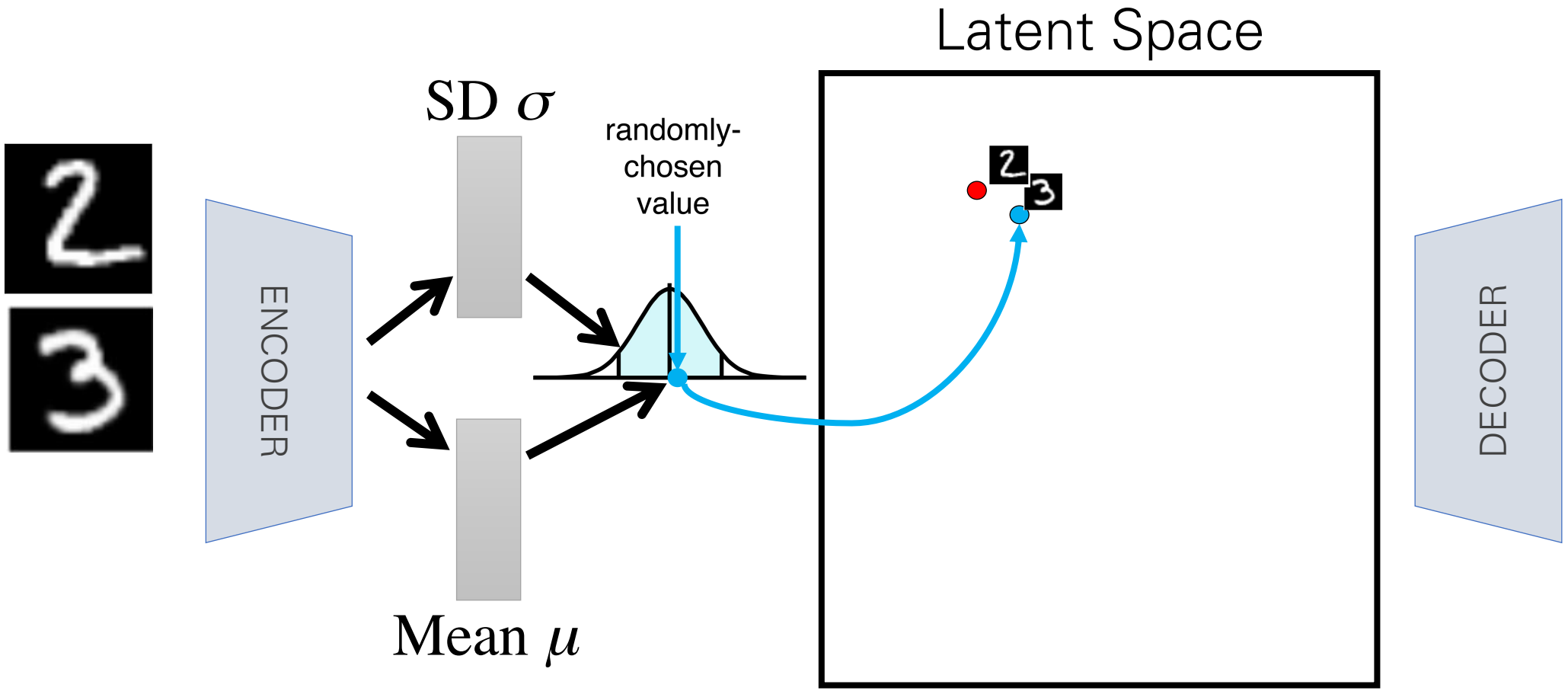
Sample $z_1 \sim N(\mu_1, \sigma_1)$

Blending Latent Variables



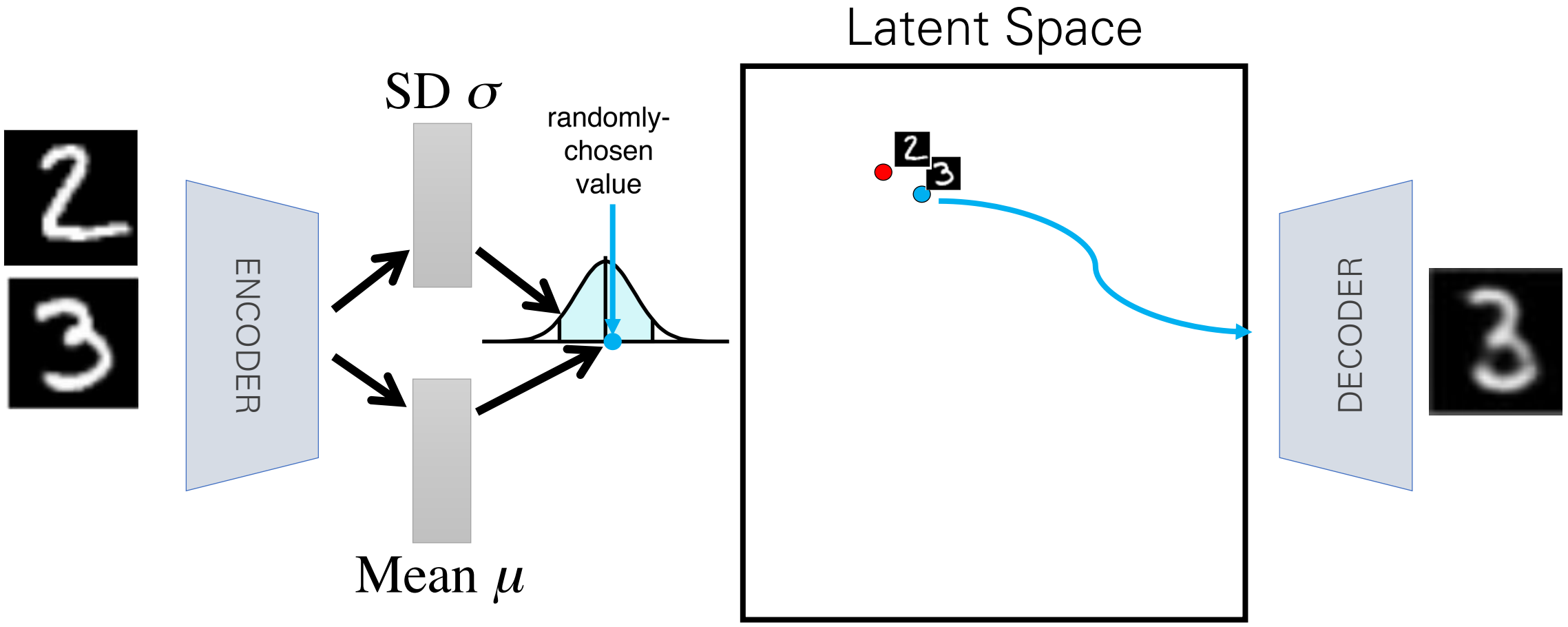
Decode to $\hat{x}_1$

Separability in Variational Autoencoders



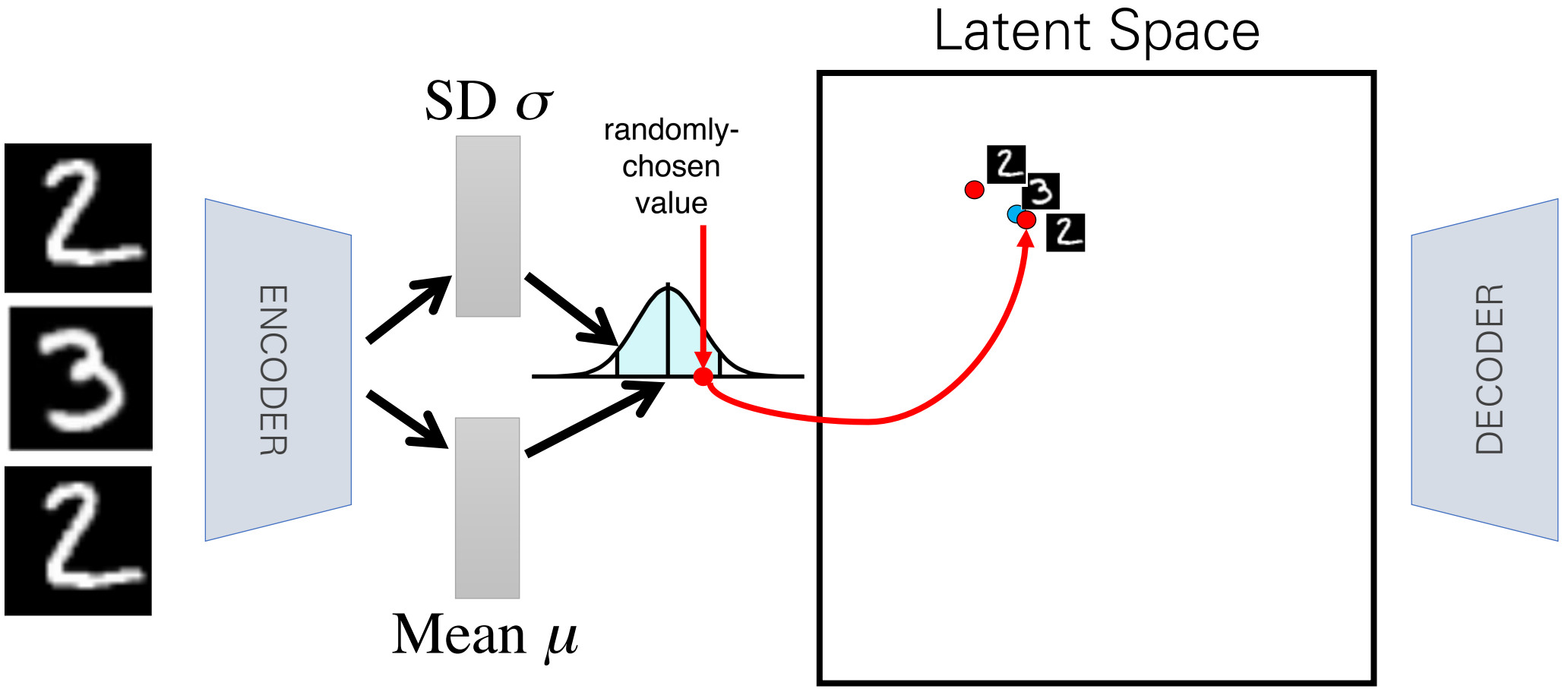
Encode the second sample (a "3") find μ_2, σ_2. Sample $z_2 \sim N(\mu_2, \sigma_2)$

Separability in Variational Autoencoders



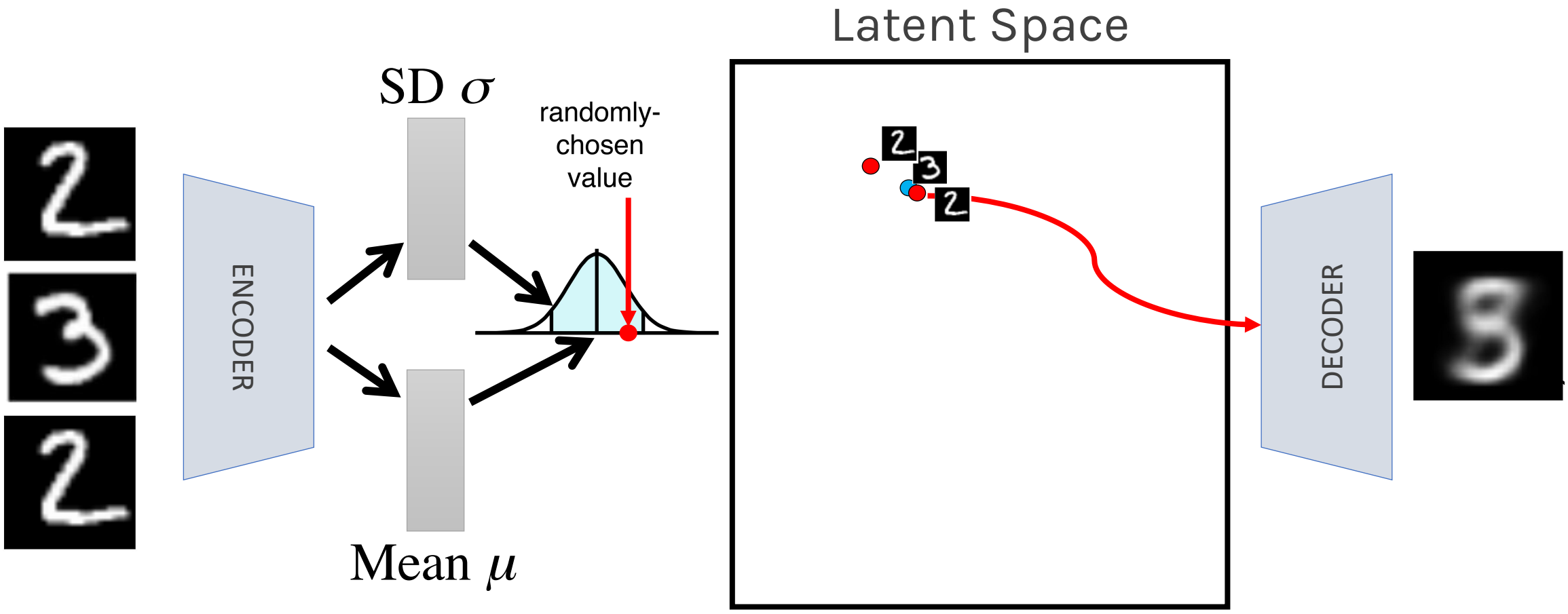
Decode to $\hat{x}_2$

Separability in Variational Autoencoders



Train with the first sample (a "2") again and find μ_1, σ_1. However $\mathbf{z}_1 \sim N(\mu_1, \sigma_1)$ **will not be the same**. It can happen to be close to the "3" in latent space.

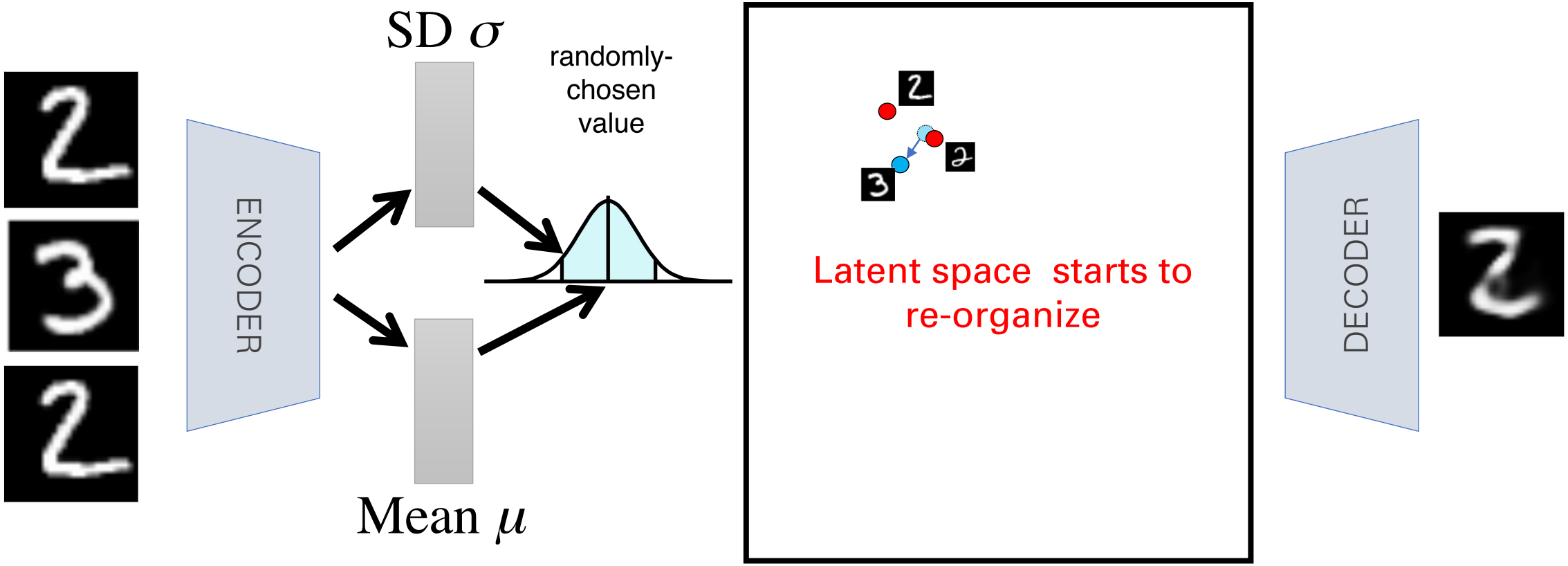
Separability in Variational Autoencoders



Decode to $\hat{x}_1$. Since the decoder only knows how to map from latent space to $\hat{x}$ space, it will return a "3".

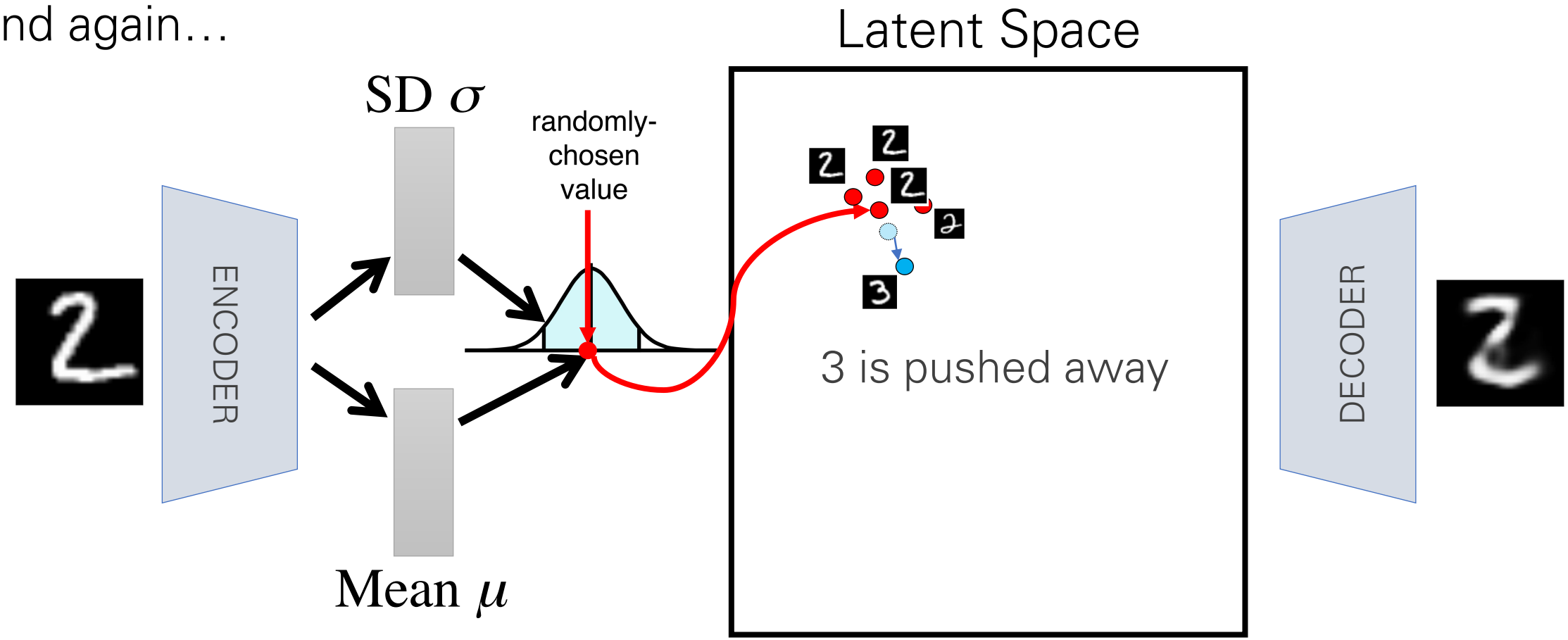
Separability in Variational Autoencoders

Train with 1st sample again



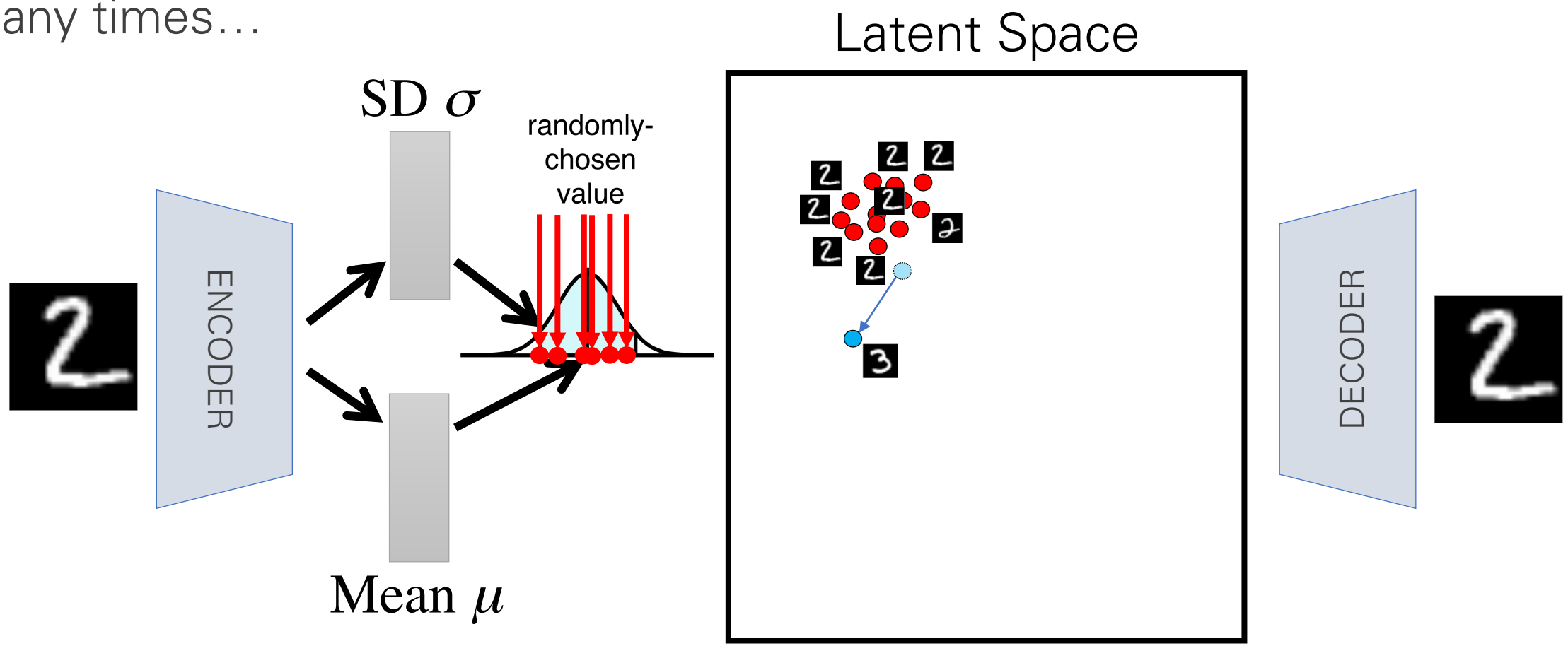
Separability in Variational Autoencoders

And again…



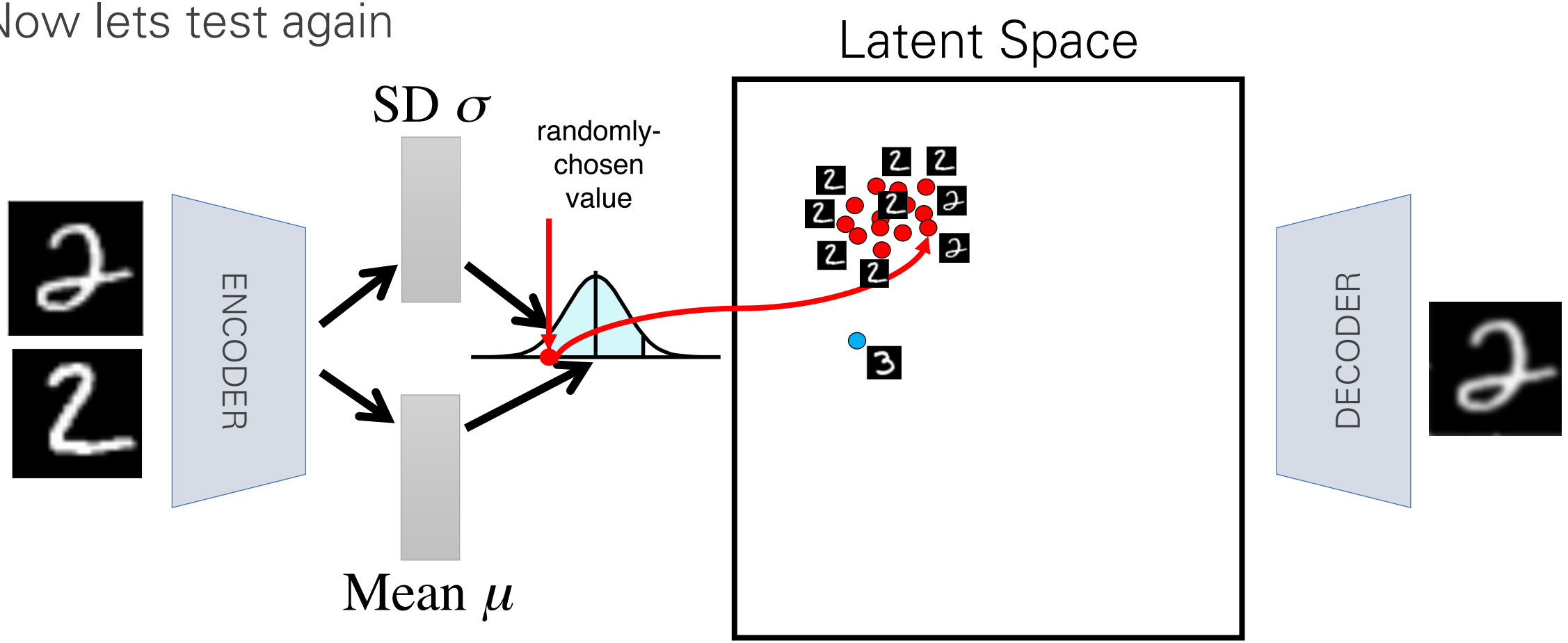
Separability in Variational Autoencoders

Many times…



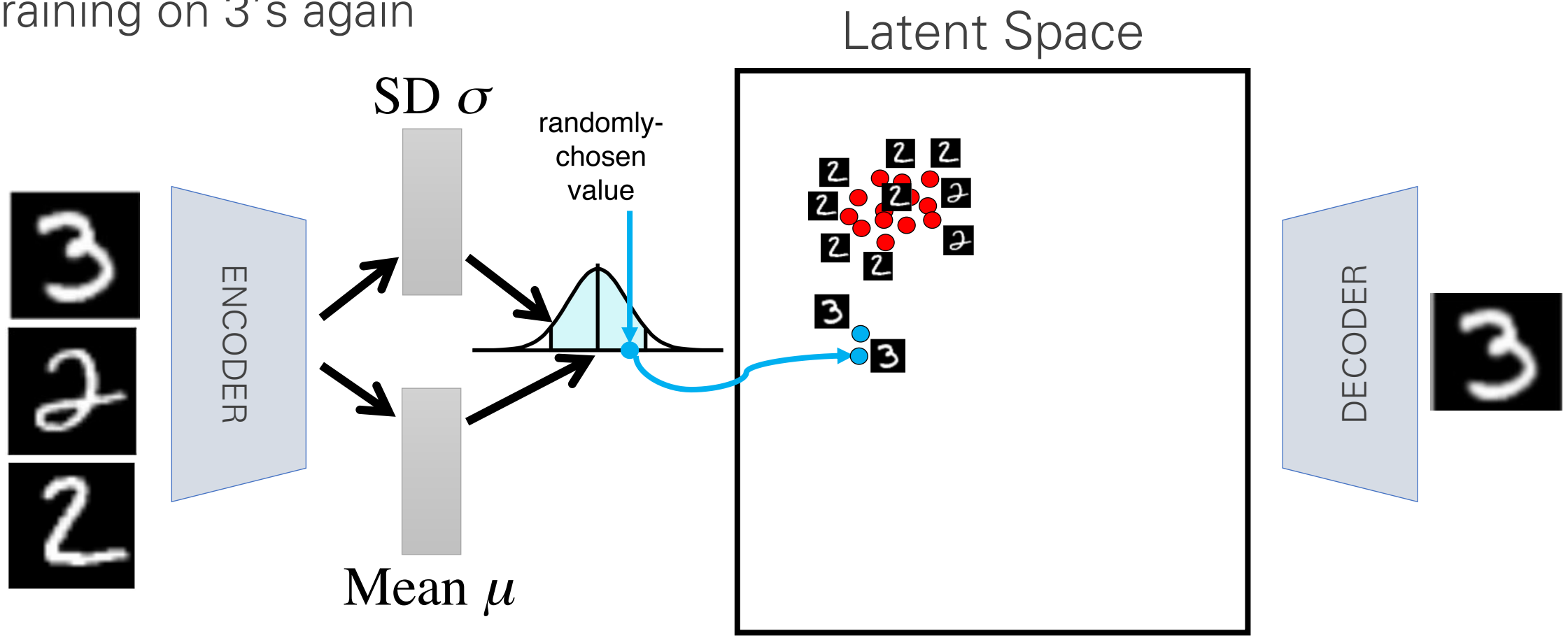
Separability in Variational Autoencoders

Now lets test again



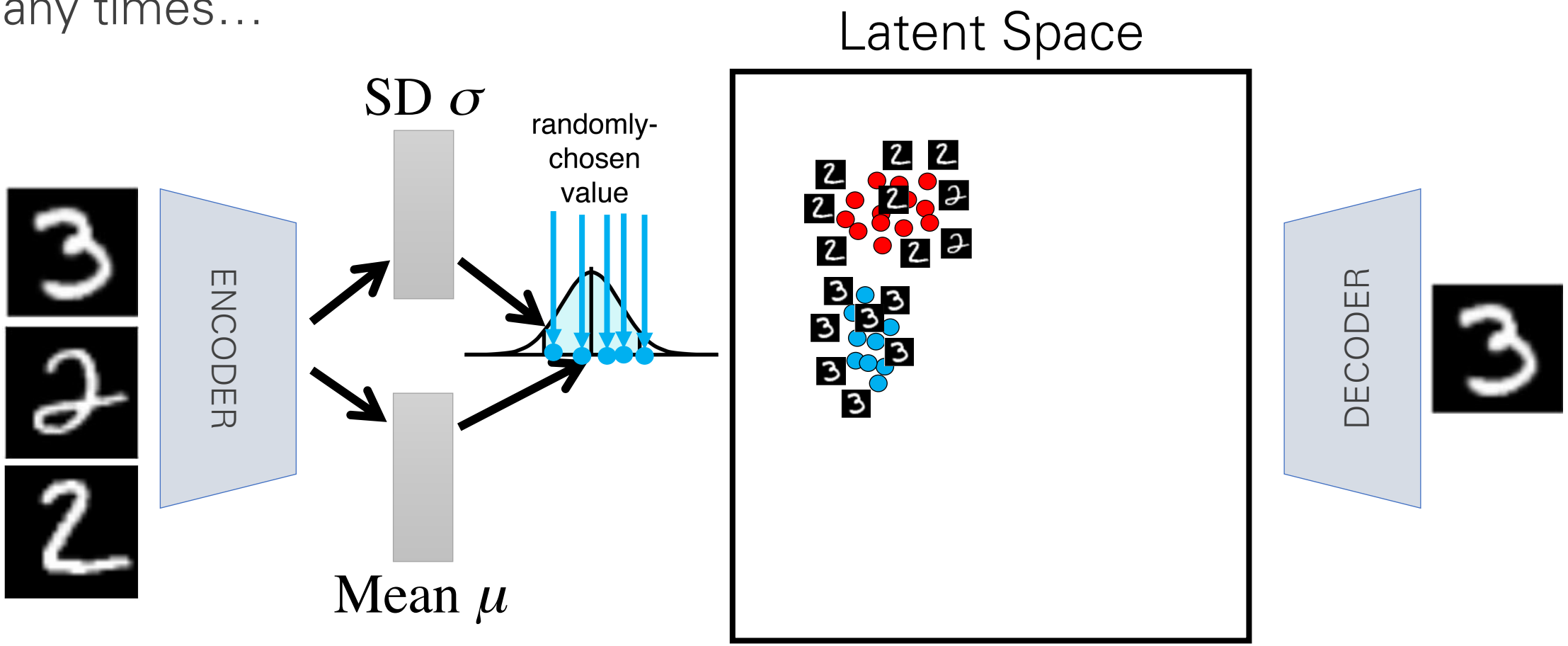
Separability in Variational Autoencoders

Training on 3's again

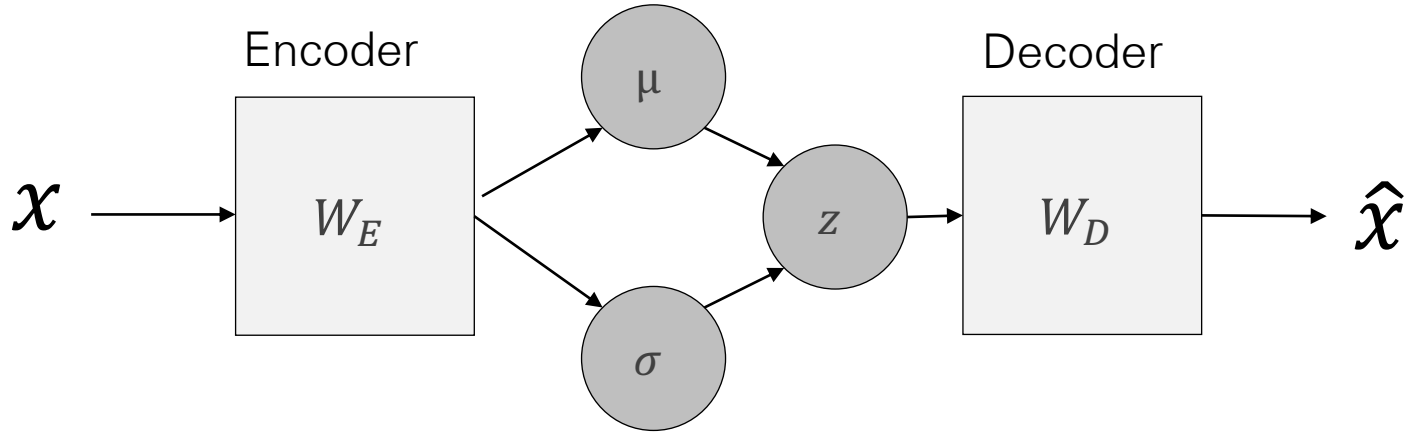


Separability in Variational Autoencoders

Many times…



Training



Training means learning W_E and W_D.

- Define a loss function $\mathcal{L}$
- Use stochastic gradient descent (or Adam) to minimize $\mathcal{L}$

The Loss function:

- Reconstruction error: $\mathcal{L}_R = \frac{1}{n}\sum_i (x_i - \hat{x}_i)^2$
- Similarity between the probability of z given x, $\mathrm{p}(z|x)$, and some predefined probability distribution $\mathrm{p}(\mathrm{z})$, which can be computed by Kullback-Leibler divergence (KL): $KL(p(z|x)||p(z))$

Gaussian VAEs 2013

Sample $z \sim \mathcal{N}(0, I)$ and compute $y_\Phi(z)$

[Alec Radford]

Vector Quantized VAEs (VQ-VAE) 2019

VQ-VAE-2, Razavi et al., NeurIPS 2019

Vector Quantized VAEs (VQ-VAE) 2019

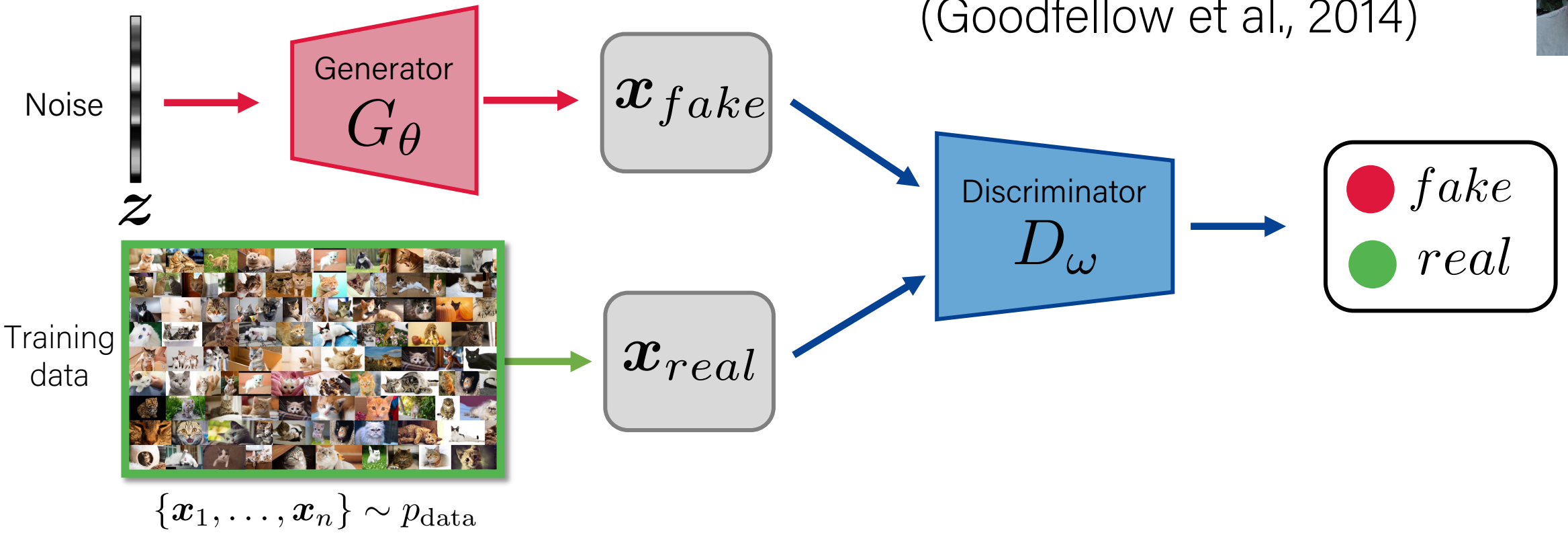
Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

VQ-VAE-2, Razavi et al., NeurIPS 2019

Generative Adversarial Networks

Generative Adversarial Networks (GANs)

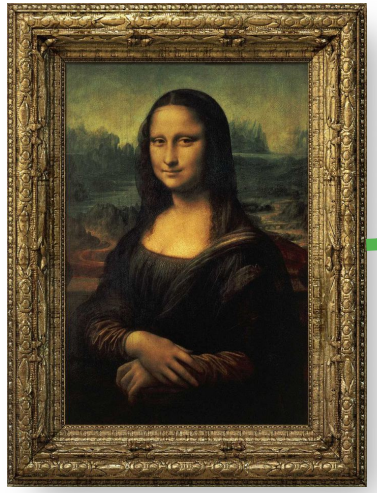
(Goodfellow et al., 2014)



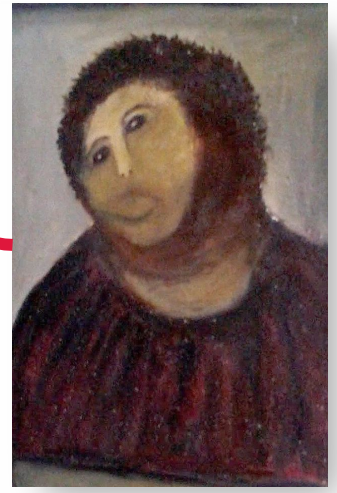
- A game between a generator $G_\theta(z)$ and a discriminator $D_\omega(x)$
  - Generator tries to fool discriminator (i.e. generate realistic samples)
  - Discriminator tries to distinguish fake from real samples

Intuition behind GANs

D_ω: Discriminator (*Art Critic*)



$\boldsymbol{x}_{real}$



$\boldsymbol{x}_{fake}$

G_θ: Generator (*Forger*)

GAN Training: Minimax Game(Goodfellow et al., 2014)

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log D_{\omega}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\log \left(1 - D_{\omega}(G_{\theta}(\boldsymbol{z}))\right)\right]$$

Real data

Noise vector used to generate data

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log \left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Cross-entropy loss for binary classification

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

Generator maximizes the log-probability of the discriminator being mistaken

- Equilibrium of the game
- Minimizes the Jensen-Shannon divergence

GAN Training: Minimax Game(Goodfellow et al., 2014)

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log D_{\omega}(\boldsymbol{x}) \right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\log \left(1 - D_{\omega}(G_{\theta}(\boldsymbol{z})) \right) \right]$$

Real data

Noise vector used to

ross-entropy
ss for binary
assification

probability
of the discriminator being mistaken

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim}$$

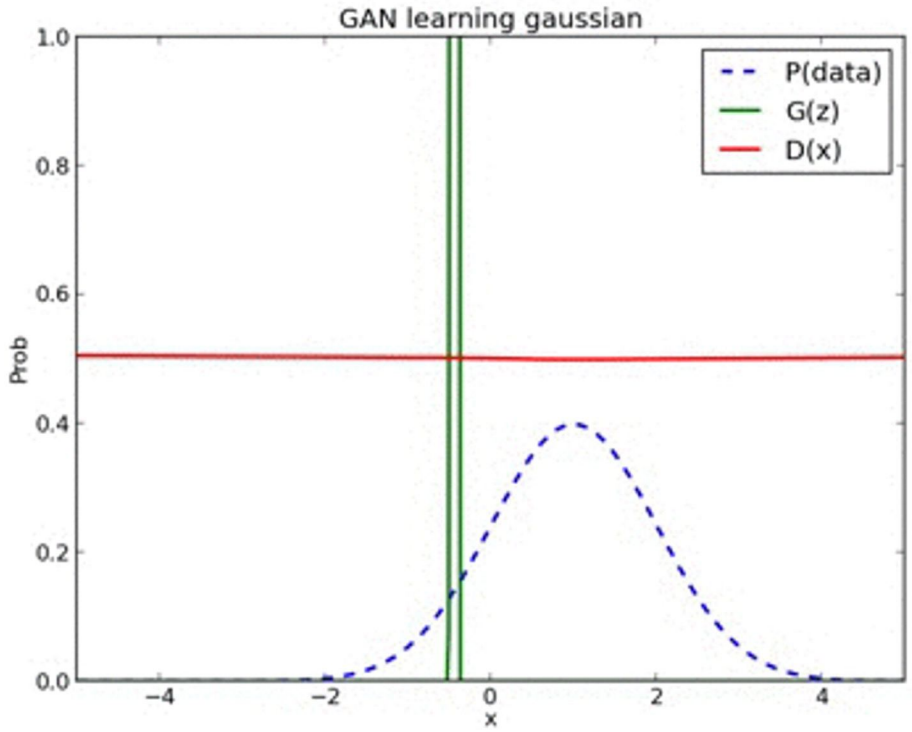
$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}}$$

Important question is "Does this converge??"

- Equilibrium of the game
- Minimizes the Jensen-Shannon divergence

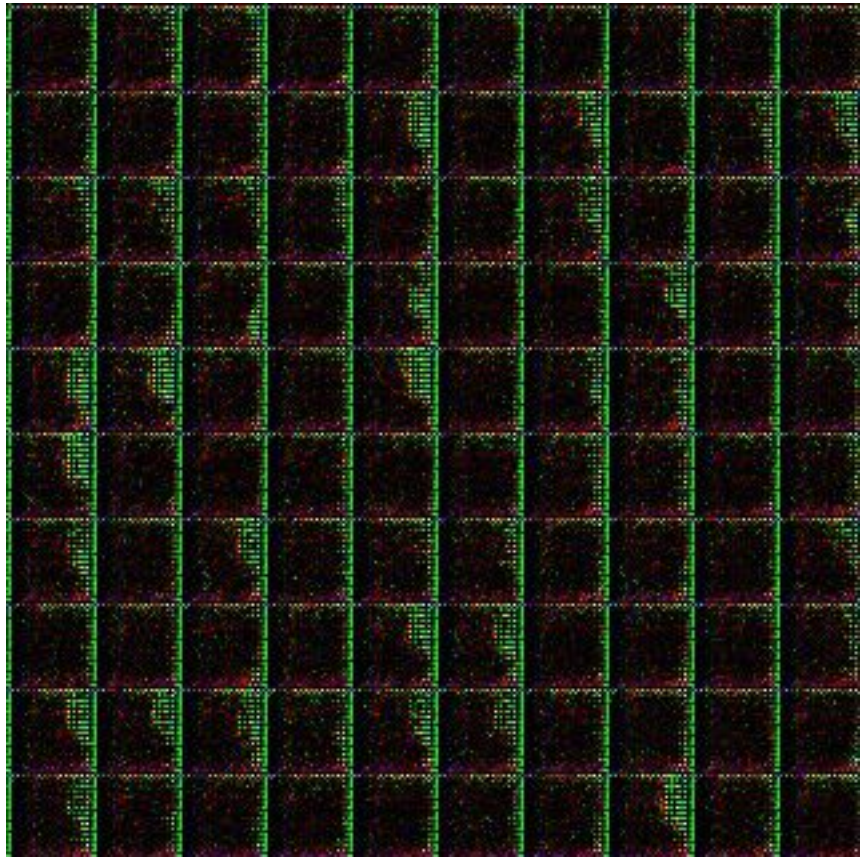
Training Procedure

(Goodfellow et al., 2014)



Source: Alec Radford

Generating 1D points



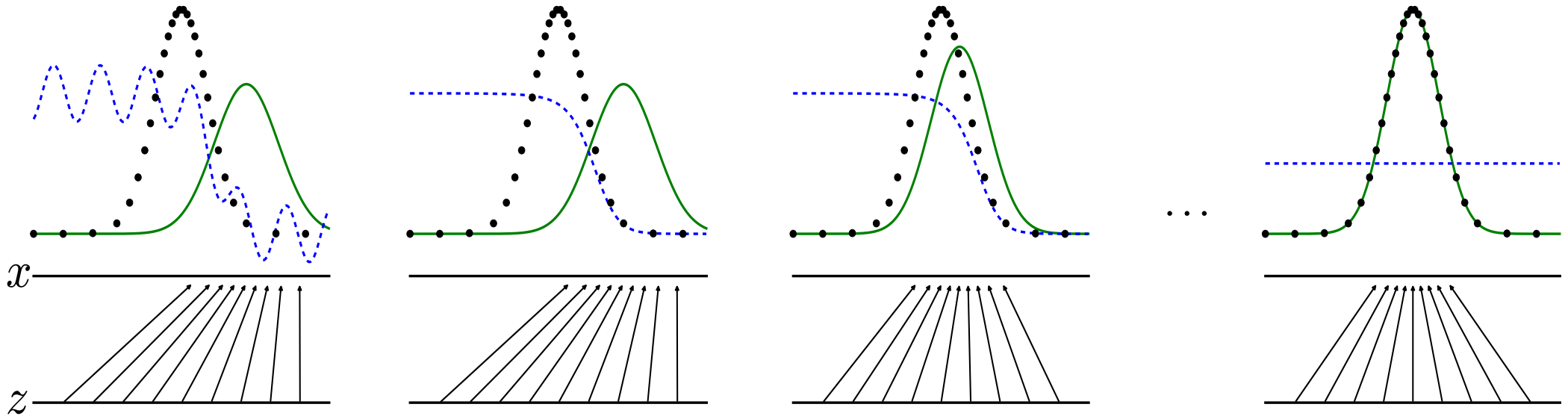
Source: OpenAI blog

Generating images

Training Procedure

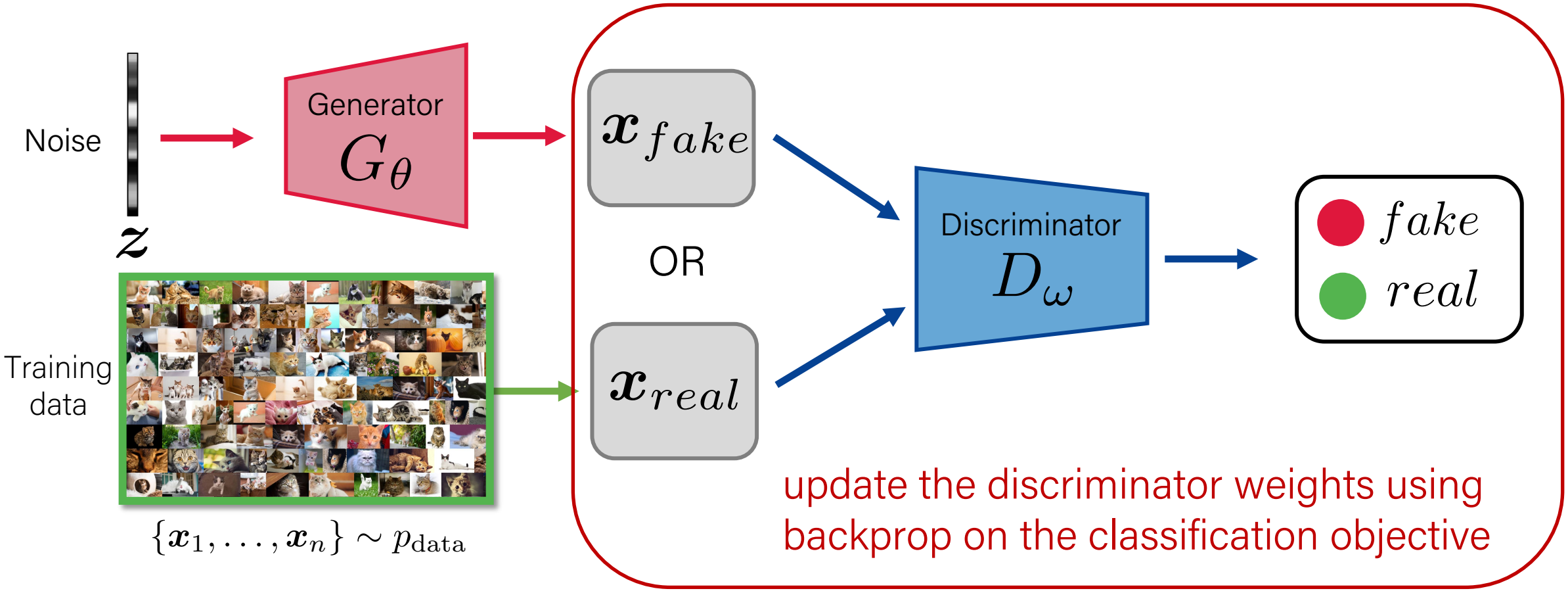
(Goodfellow et al., 2014)

- Use SGD on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples



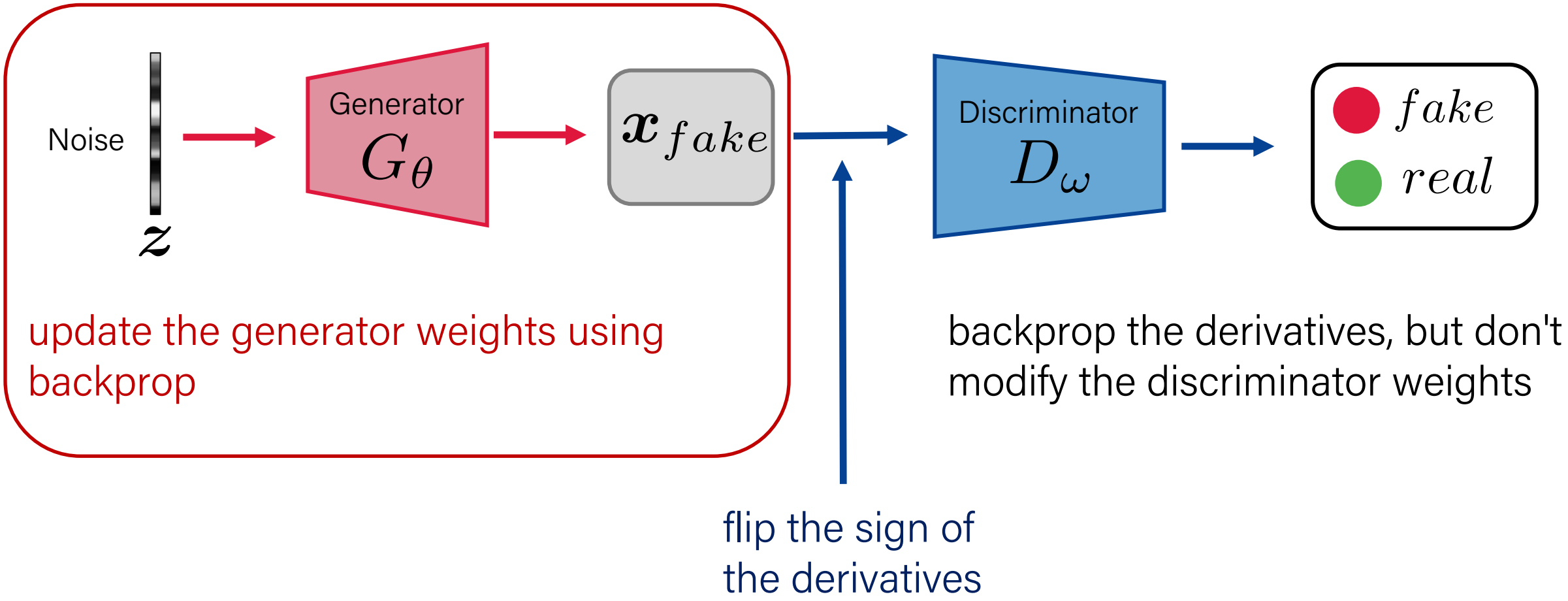
Training Procedure

- Updating the discriminator:



Training Procedure

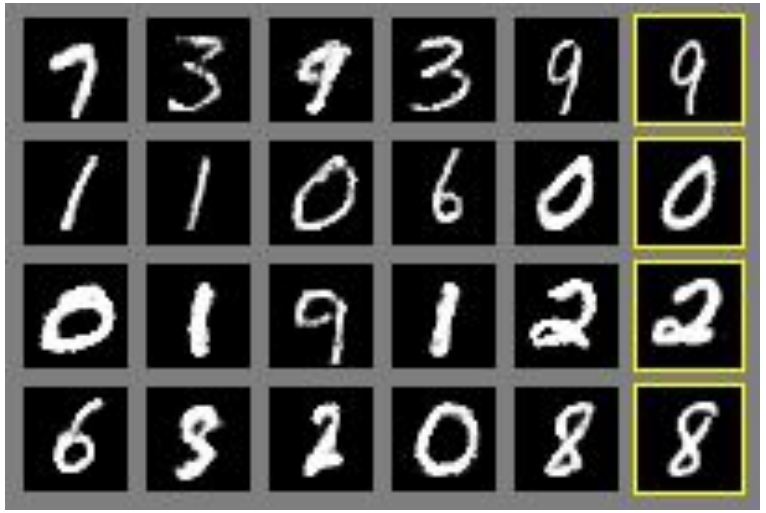
- Updating the generator:



Results

(Goodfellow et al., 2014)

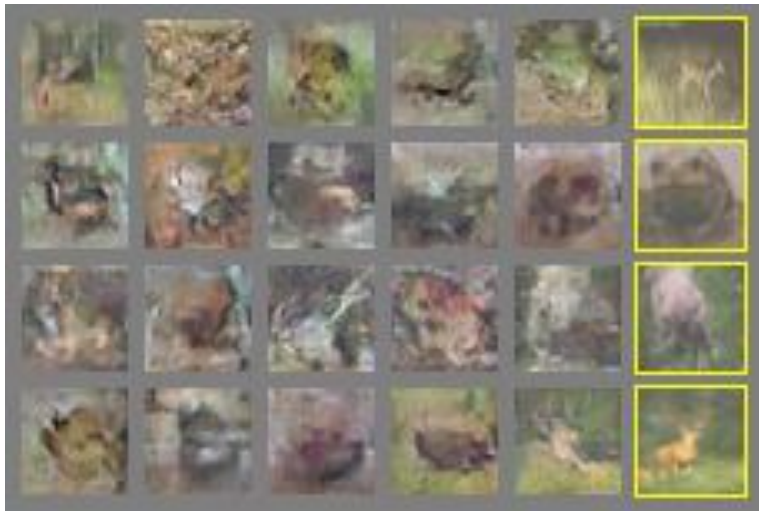
- The generator uses a mixture of rectifier linear activations and/or sigmoid activations
- The discriminator net used maxout activations.



MNIST samples

TFD samples

CIFAR10 samples
(*fully-connected model*)

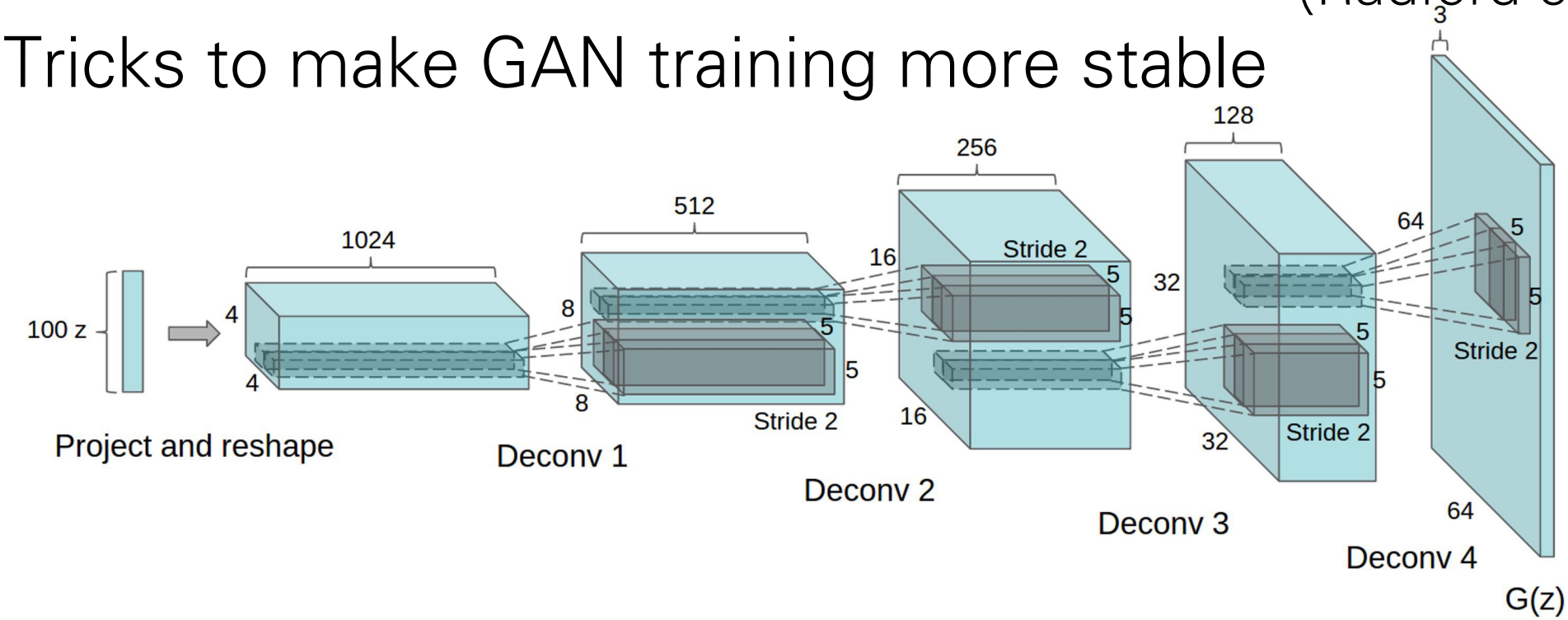


CIFAR10 samples
(*convolutional discriminator, deconvolutional generator*)

Deep Convolutional GANs (DCGAN)

(Radford et al., 2015)

- **Idea:** Tricks to make GAN training more stable



- No fully connected layers
- Batch Normalization (Ioffe and Szegedy, 2015)
- Leaky Rectifier in D
- Use Adam (Kingma and Ba, 2015)
- Tweak Adam hyperparameters a bit (lr=0.0002, b1=0.5)

DCGAN for LSUN Bedrooms

64×64 pixels
~3M images

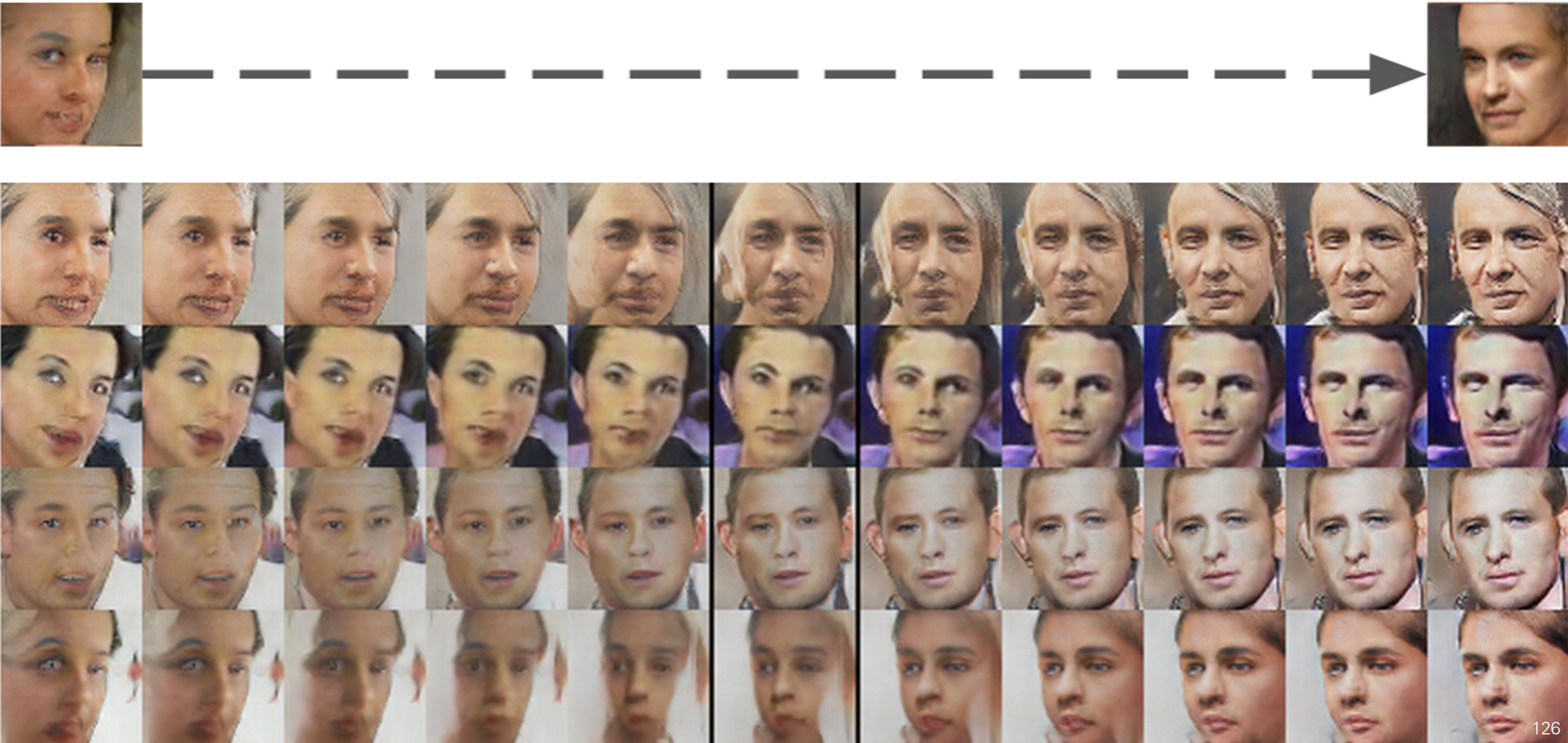
(Radford et al., 2015)

Walking over the latent space

(Radford et al., 2015)

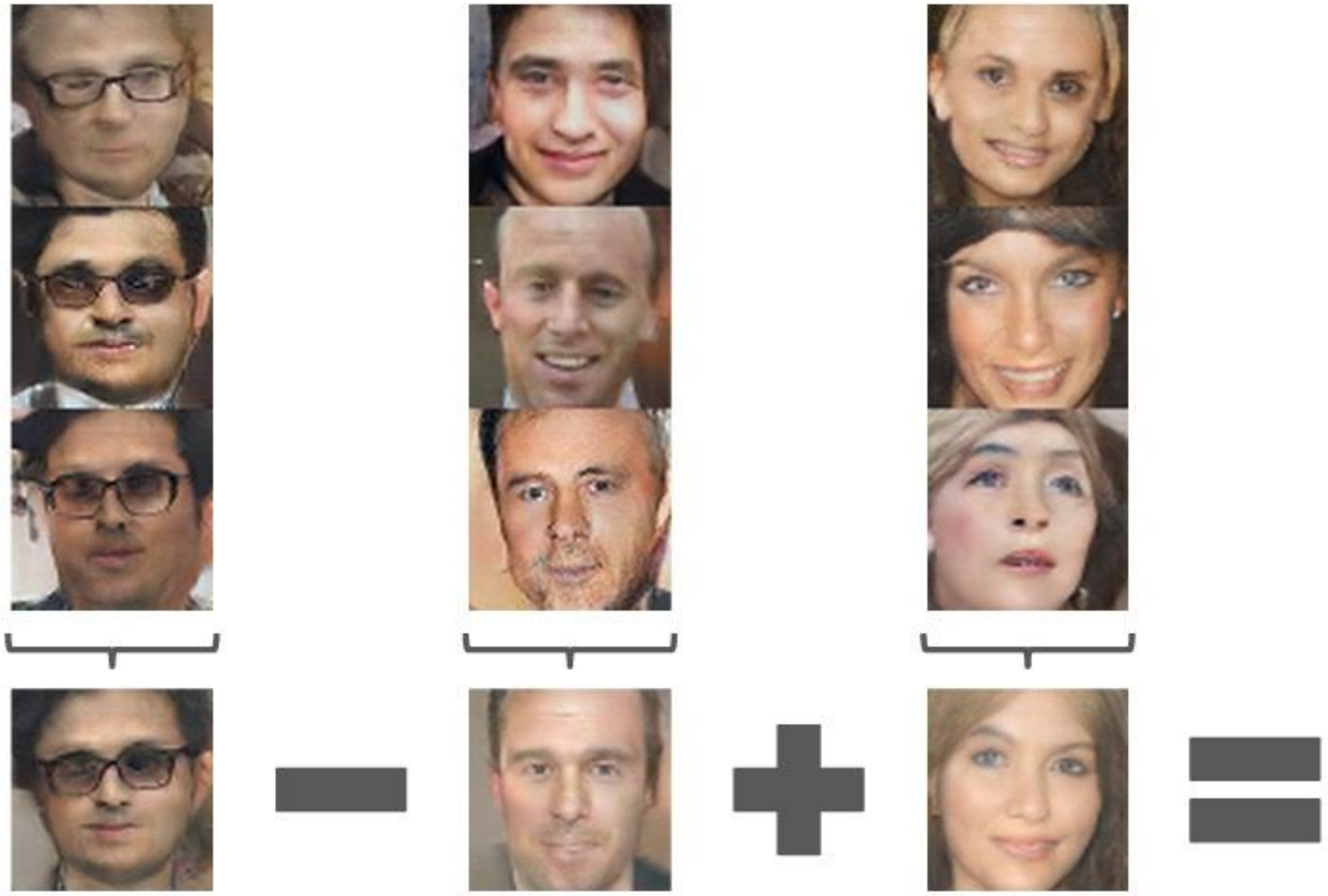
- Interpolation suggests non-overfitting behavior

Walking over the latent space (Radford et al., 2015)



Vector Space Arithmetic

(Radford et al., 2015)



man
with glasses

man
without glasses

woman
without glasses

woman with glasses

Vector Space Arithmetic

(Radford et al., 2015)

smiling woman

neutral woman

neutral man

smiling man

Subclasses of GANs

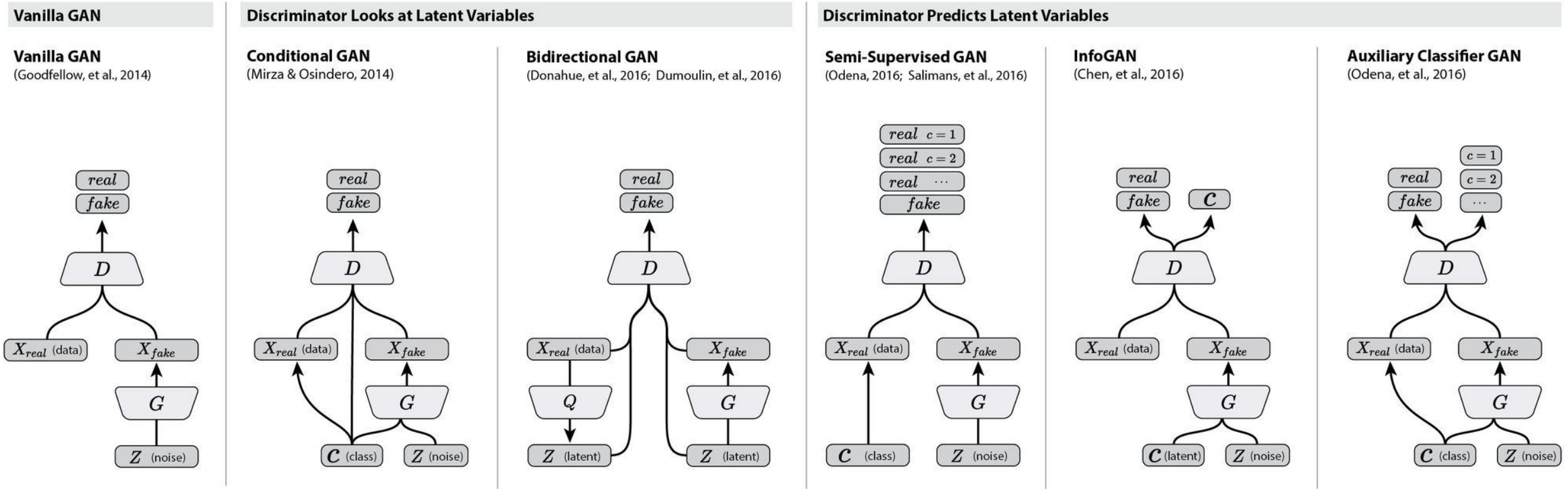
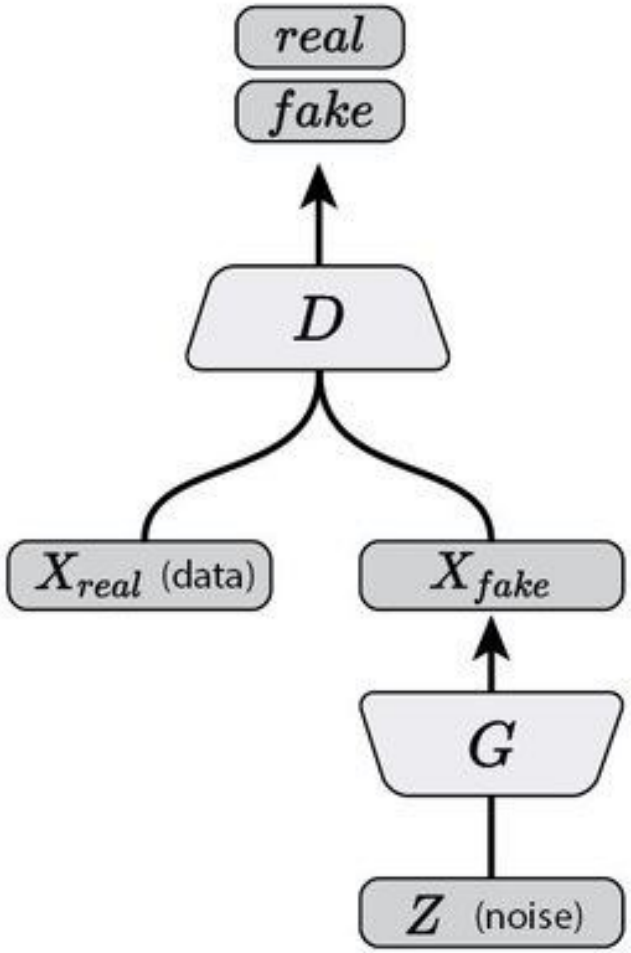
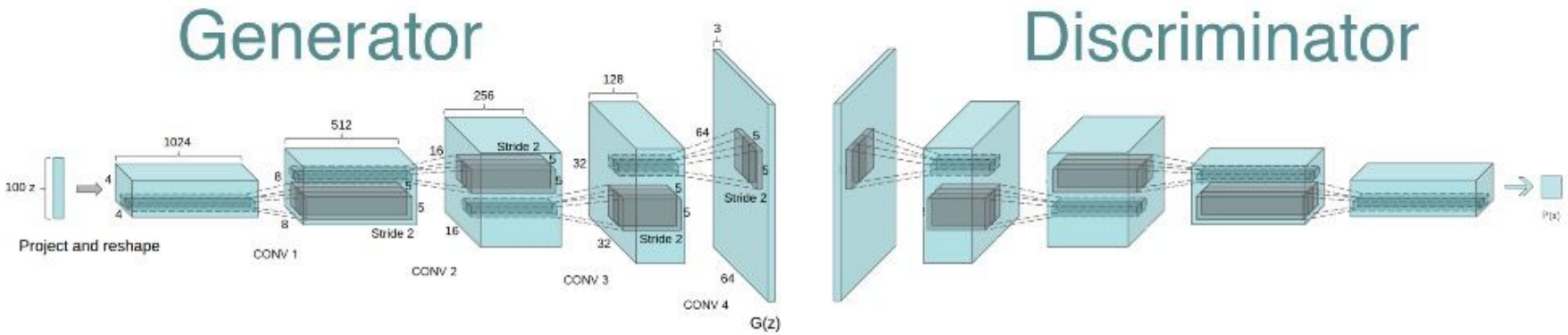


Image: Christopher Olah

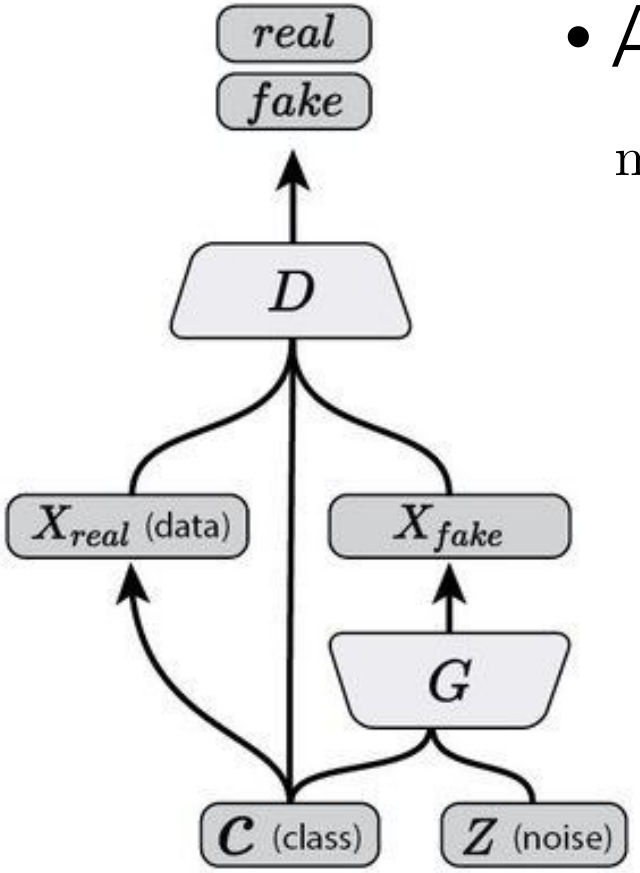
Vanilla GAN (Goodfellow et al., 2014)





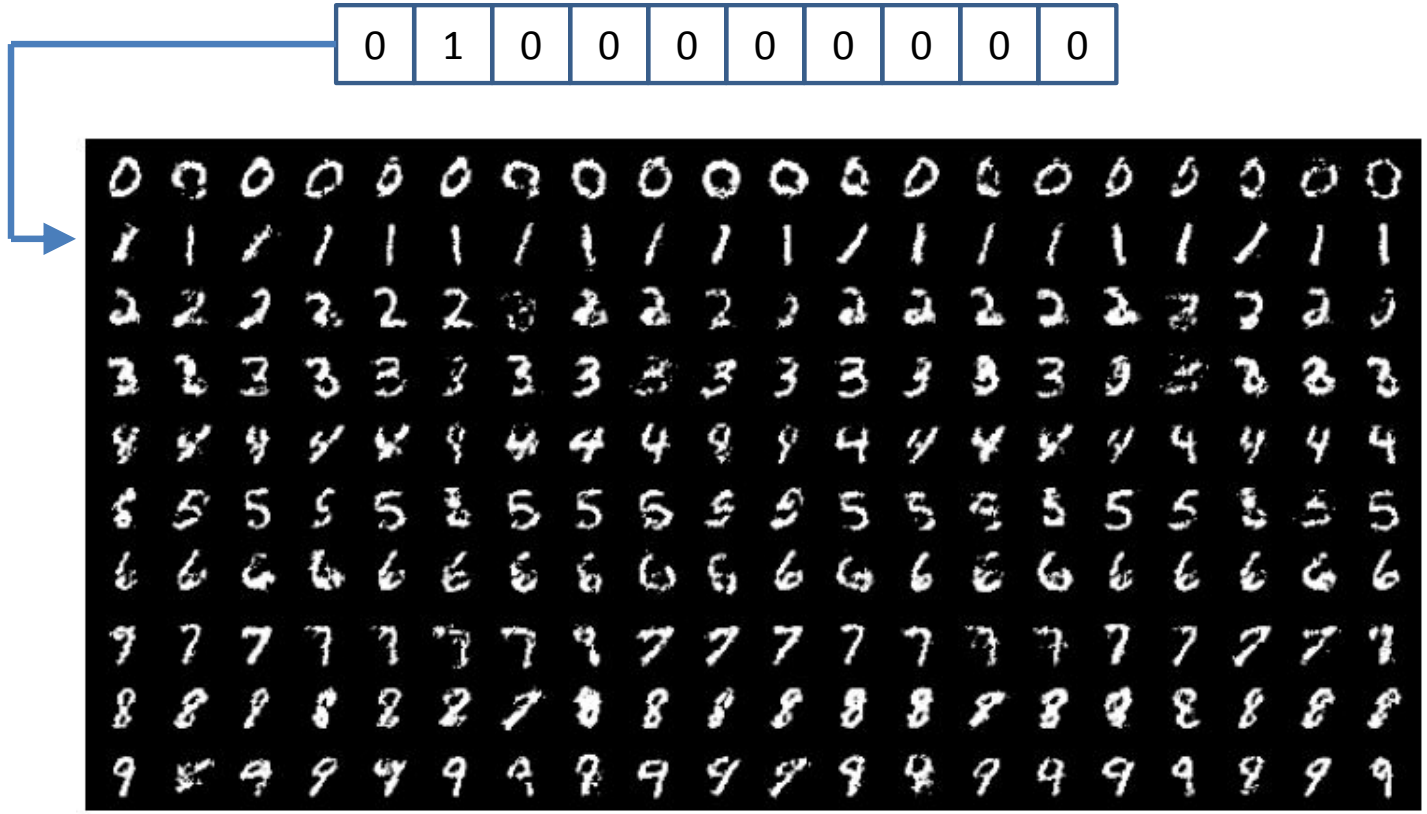
DCGAN (Radford et al., 2015)

Conditional GAN (Mirza and Osindero, 2014)

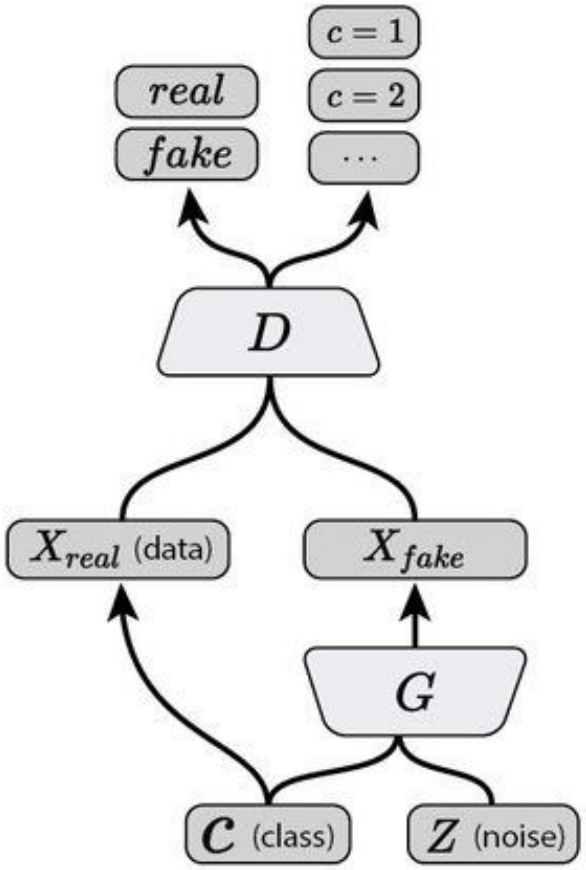


- Add conditional variables $\boldsymbol{y}$ into G and D

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



Auxiliary Classifier GAN (Odena et al., 2016)



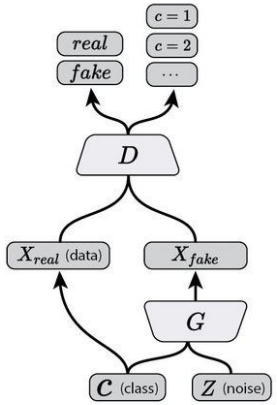
- Every generated sample has a corresponding class label

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$
$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

- D is trained to maximize $L_S + L_C$
- G is trained to maximize $L_C - L_S$

- Learns a representation for z that is independent of class label

Auxiliary Classifier GAN (Odena et al., 2016)

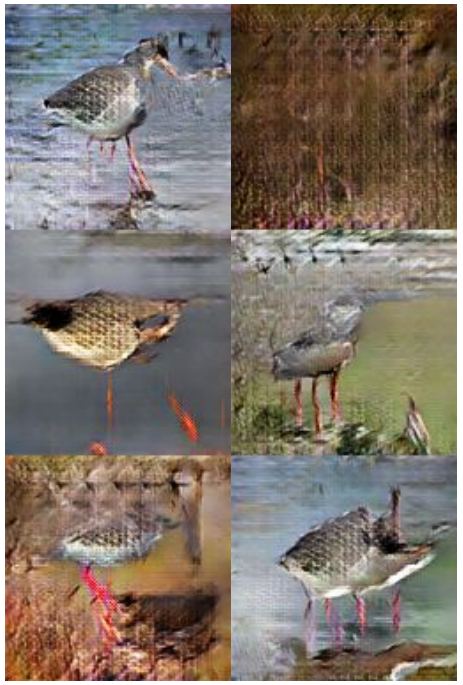


128×128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet

monarch butterfly

goldfinch

daisy

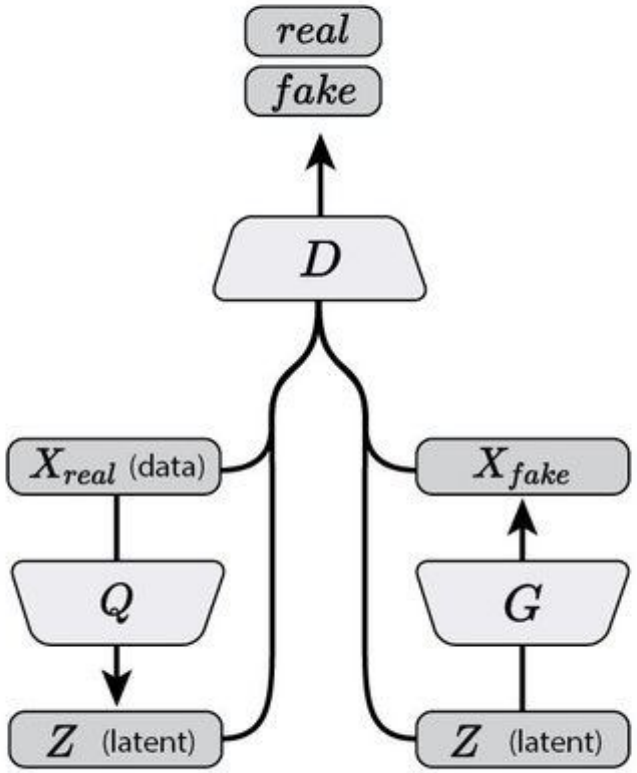


redshank

grey whale

Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)

- Jointly learns a generator network and an inference network using an adversarial process.



$$\min_G \max_D V(D,G) = \mathbb{E}_{q(\boldsymbol{x})}[\log(D(\boldsymbol{x}, G_z(\boldsymbol{x})))] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D(G_x(\boldsymbol{z}), \boldsymbol{z}))]$$

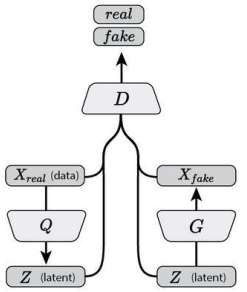
$$= \iint q(\boldsymbol{x})q(\boldsymbol{z} \mid \boldsymbol{x}) \log(D(\boldsymbol{x}, \boldsymbol{z}))d\boldsymbol{x}d\boldsymbol{z}$$

$$+ \iint p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z}) \log(1 - D(\boldsymbol{x}, \boldsymbol{z}))d\boldsymbol{x}d\boldsymbol{z}.$$

CelebA reconstructions

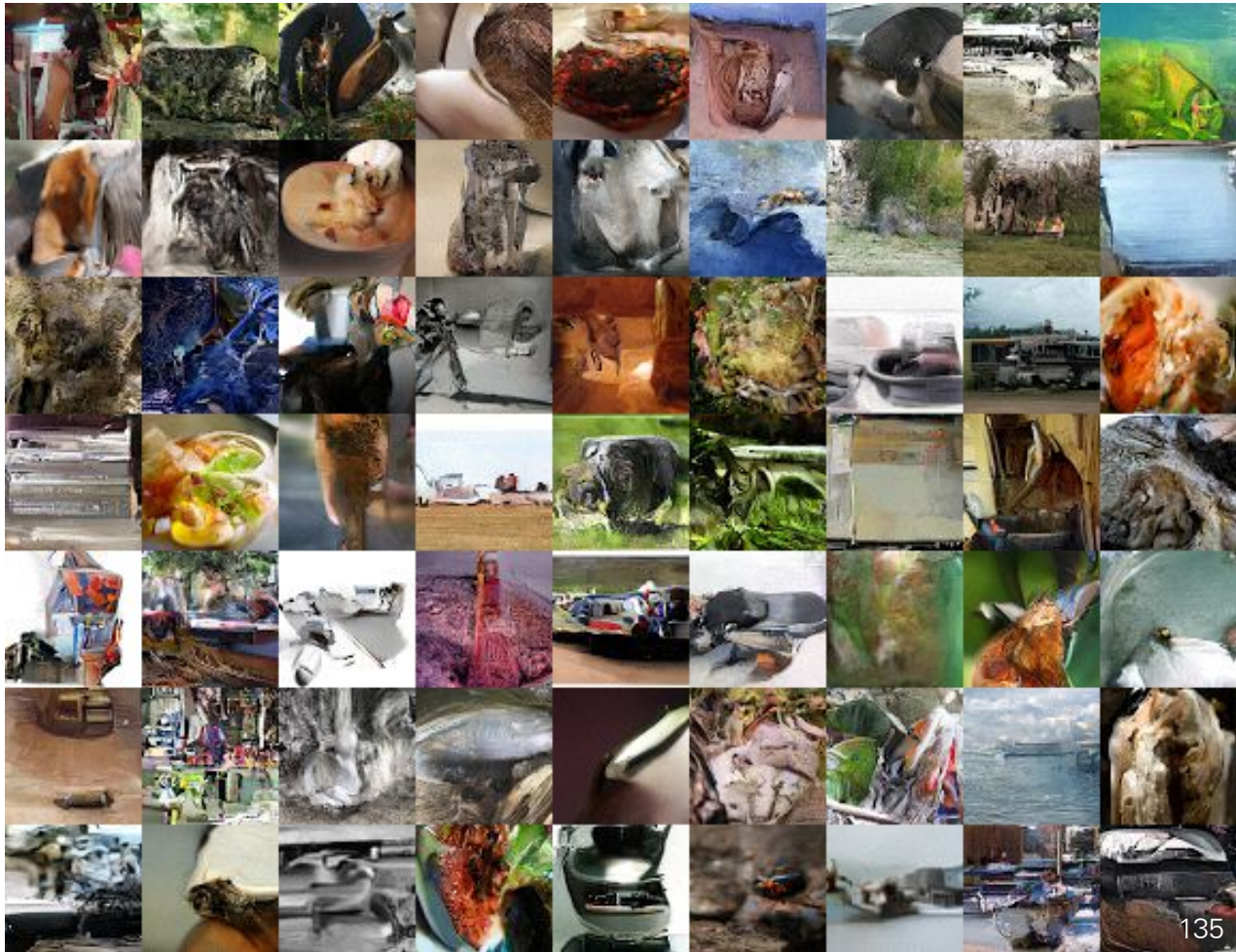
SVNH reconstructions

Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)



LSUN bedrooms

Tiny ImageNet

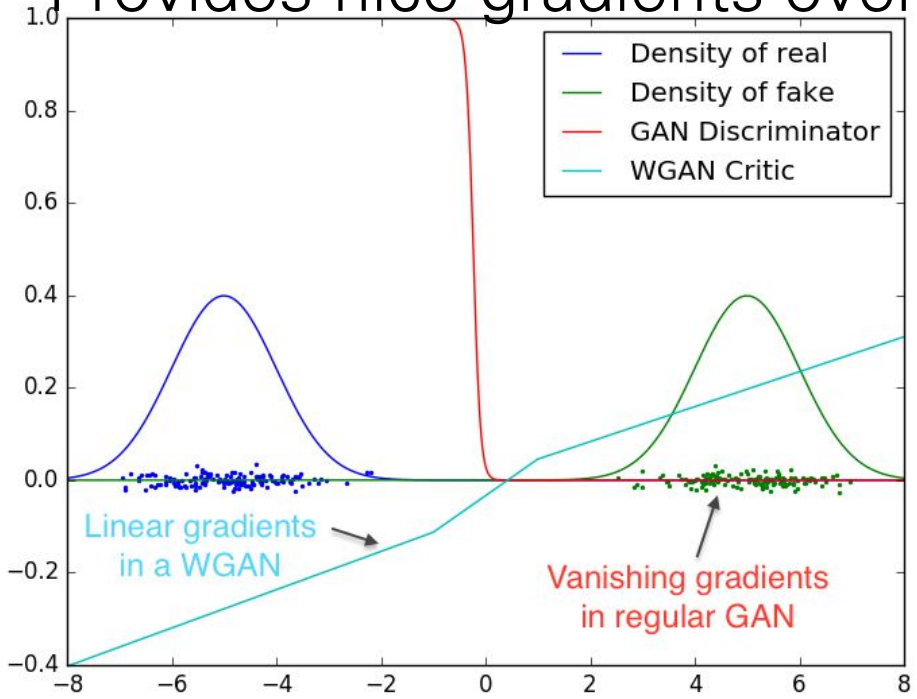


Wasserstein GAN (Arjovsky et al., 2016)

- Objective based on Earth-Mover or Wassertein distance:

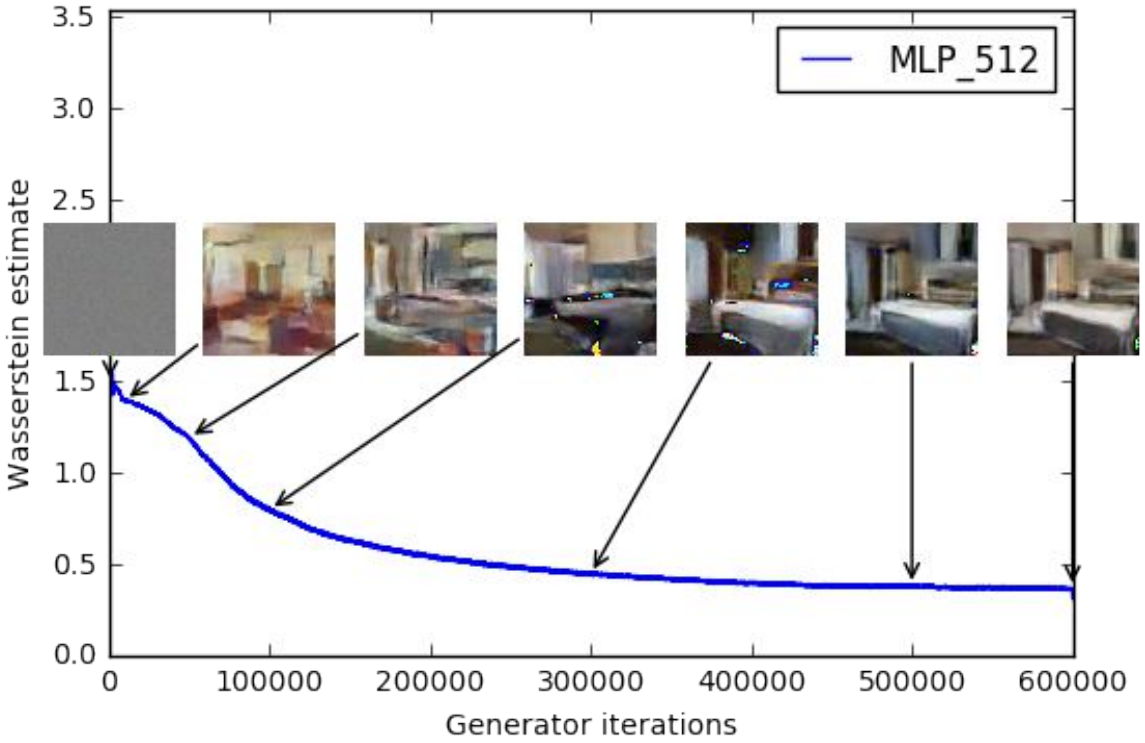
$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [D_\omega(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [D_\omega(G_\theta(\boldsymbol{z}))]$$

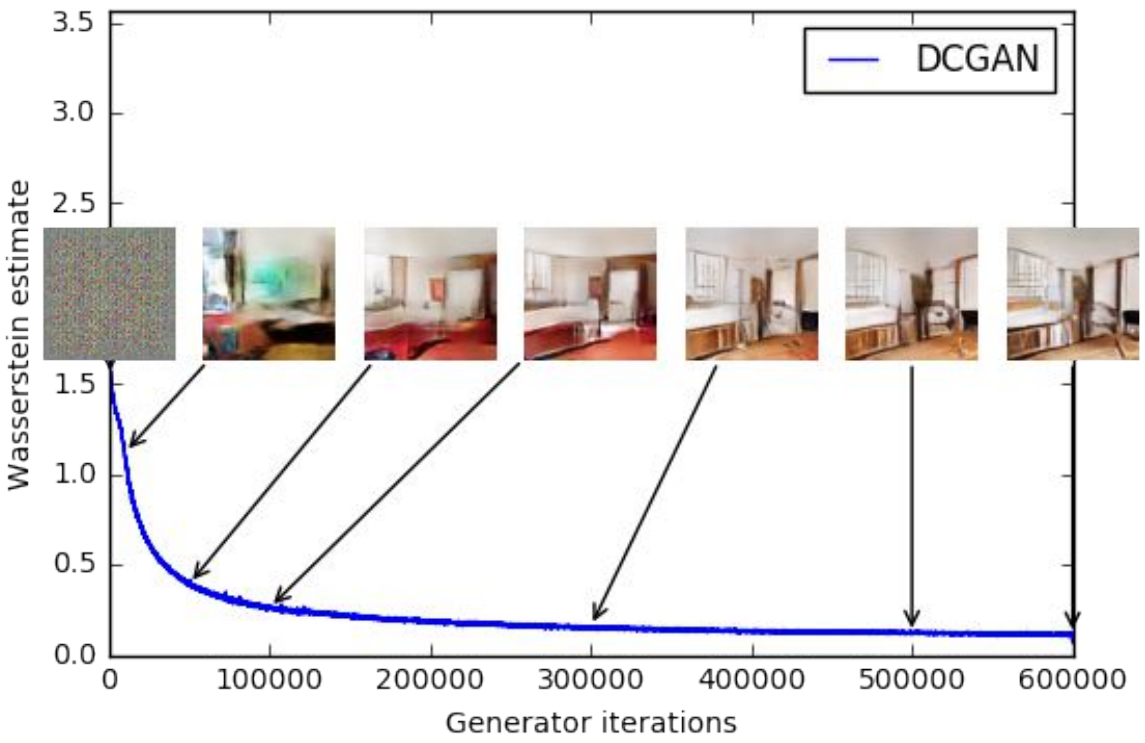
- Provides nice gradients over real and fake samples



Wasserstein GAN (Arjovsky et al., 2016)

- Wasserstein loss seems to correlate well with image quality.



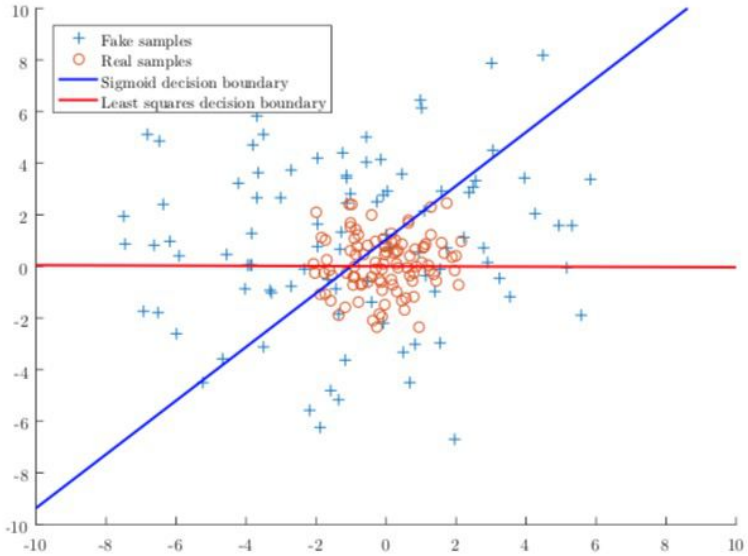


Least Squares GAN (LSGAN) (Mao et al., 2017)

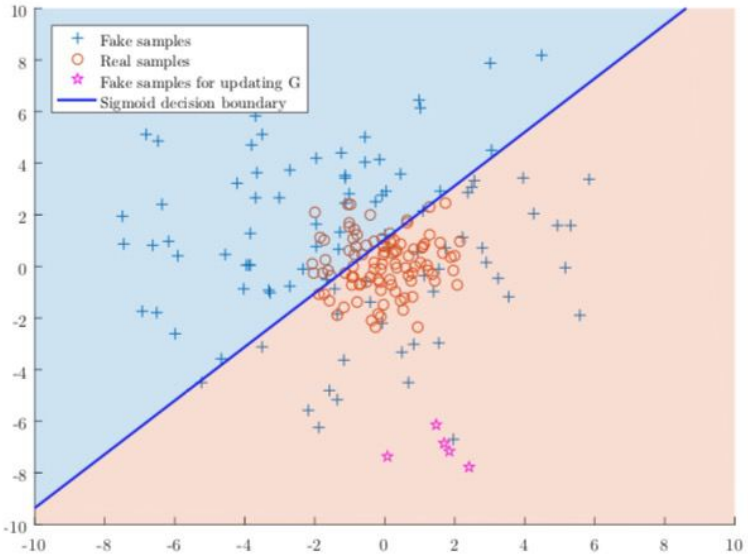
- Use a loss function that provides smooth and non-saturating gradient in discriminator D

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

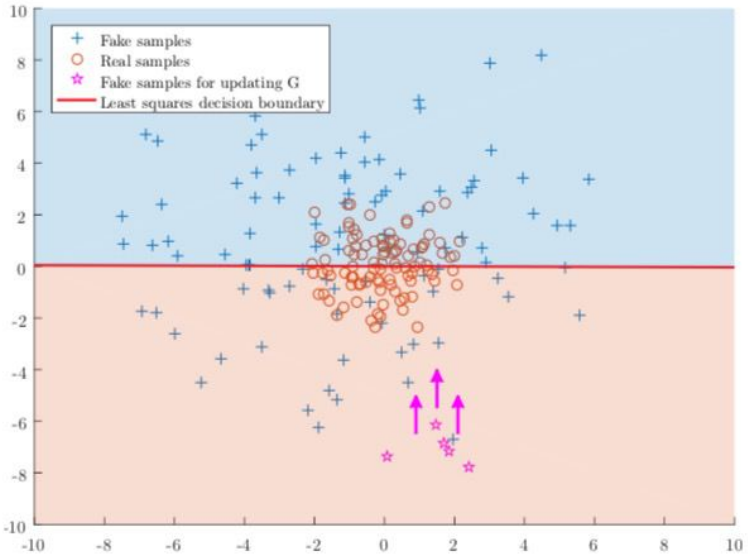
$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$



Decision boundaries of Sigmoid & Least Squares loss functions



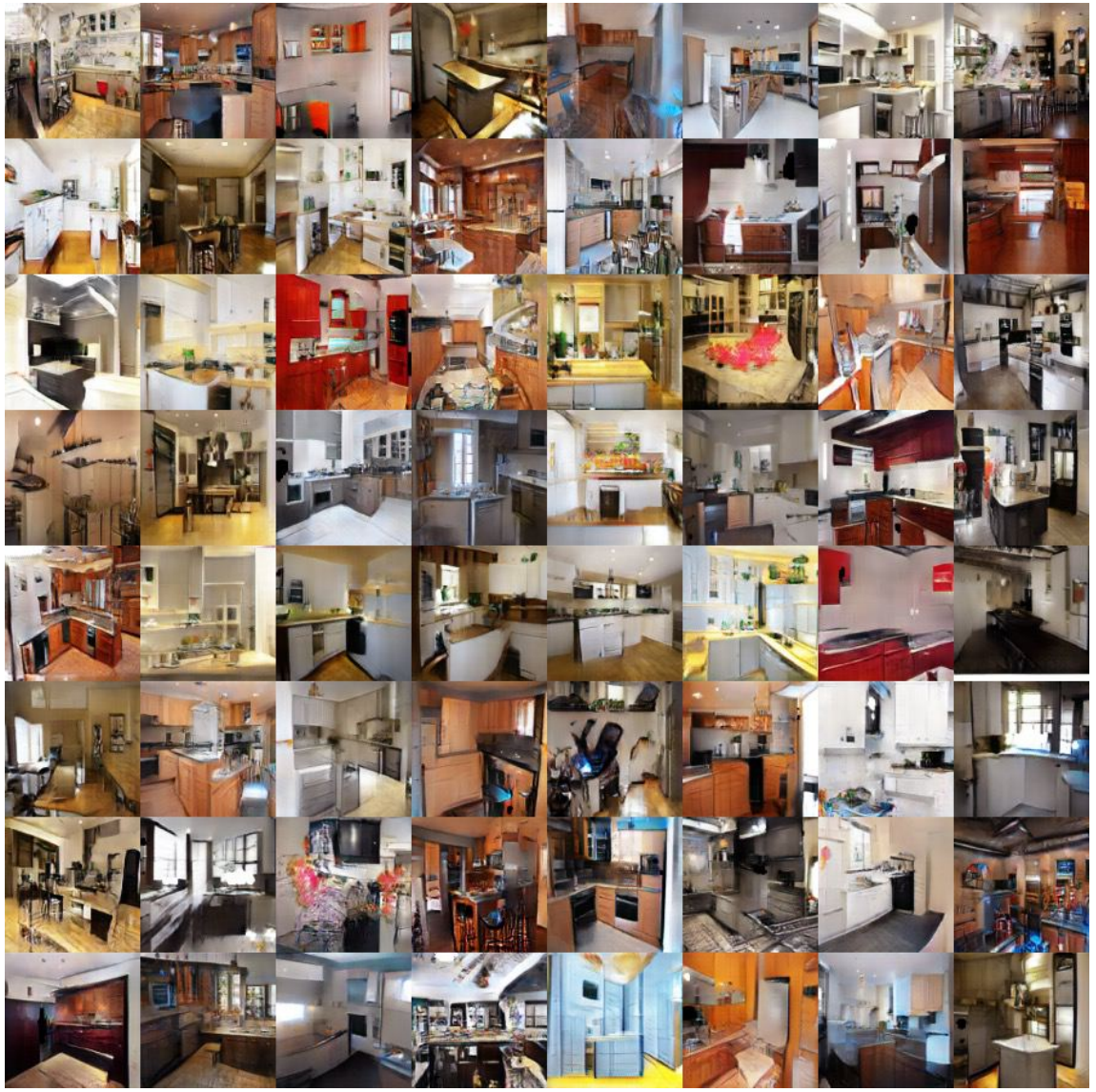
Sigmoid decision boundary



Least Squares decision boundary

Least Squares GAN (LSGAN) (Mao et al., 2017)

Church

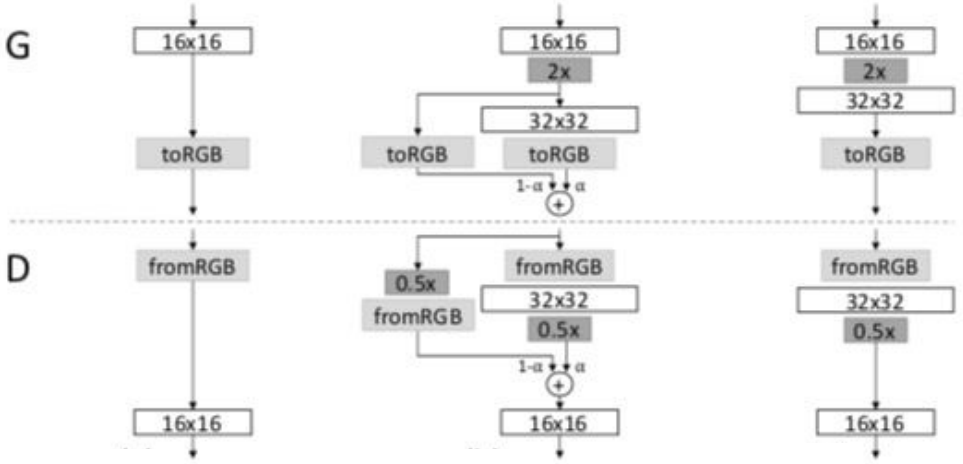


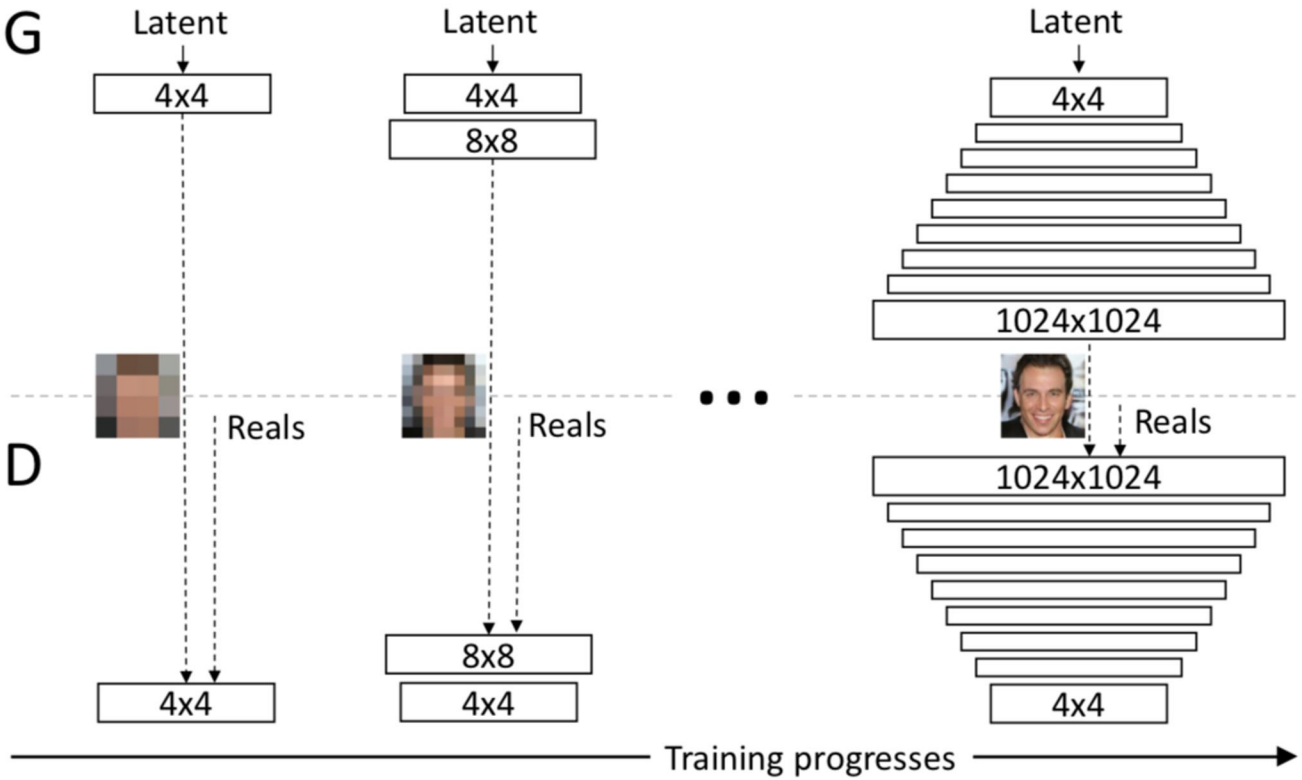
Kitchen

Progressive GANs(Karras et al., 2018)

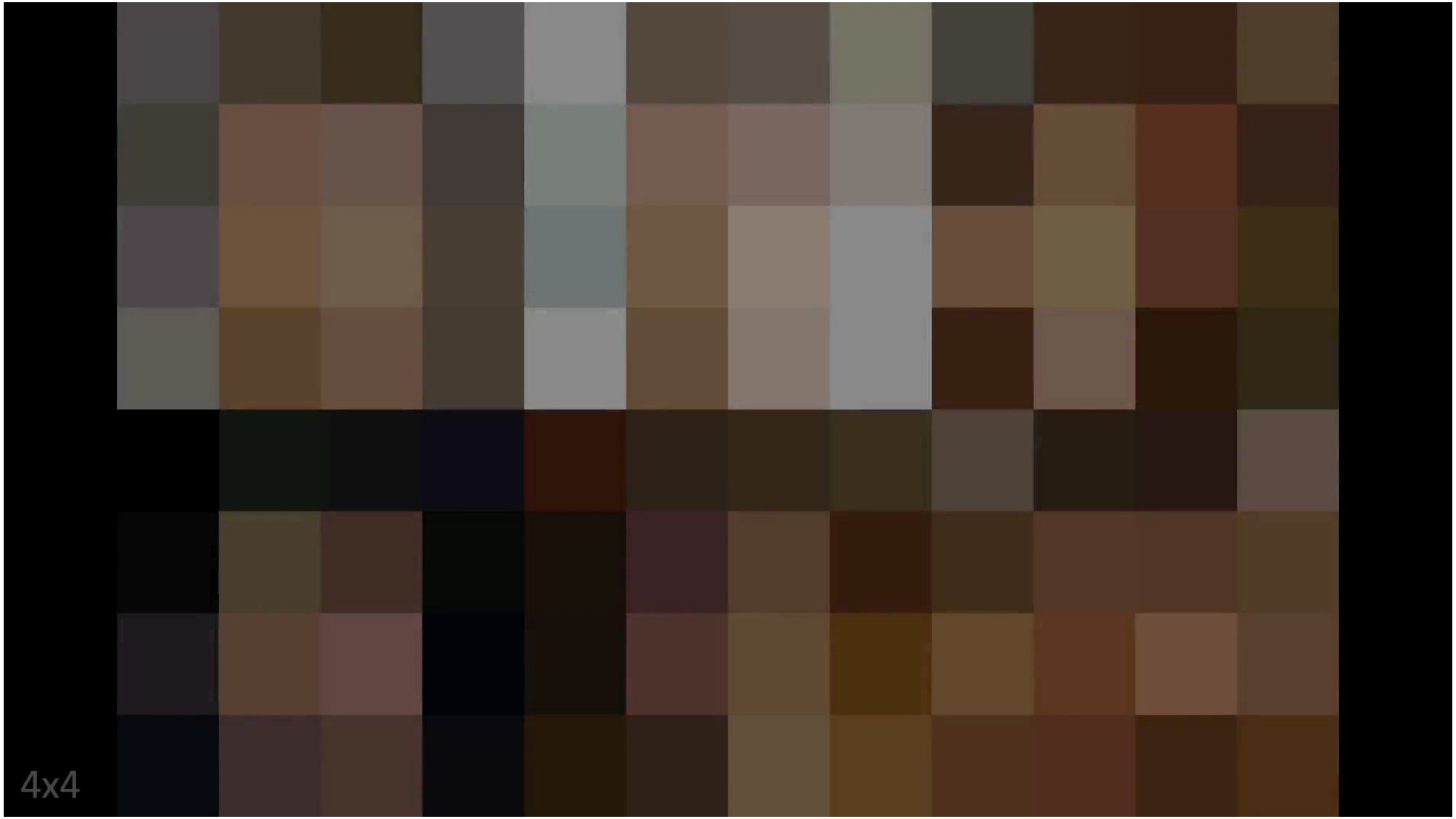
- Progressively generate high-res images

- Multi-step training from low to high resolutions





Progressive GANs(Karras et al., 2018)



- Training process

Progressive GANs (Karras et al., 2018)

CelebA-HQ
random interpolations

BigGANs (Brock et al., 2019)

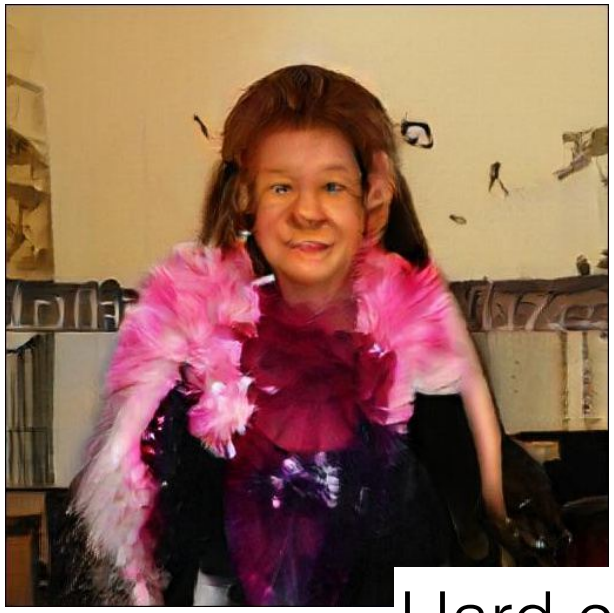
High resolution, class-conditional samples generated by the model

- BigGANs trained with 2-4x as many parameters and 8x the batch size compared to prior art.
- Uses Gaussian truncation to sample z (avoid sampling from the tail of the Gaussian distribution)
- Uses multiple other tricks including multiple regularizations including a Gradient penalty regularization and an Orthogonal Regularization:

$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2,$$

BigGANs (Brock et al., 2019)

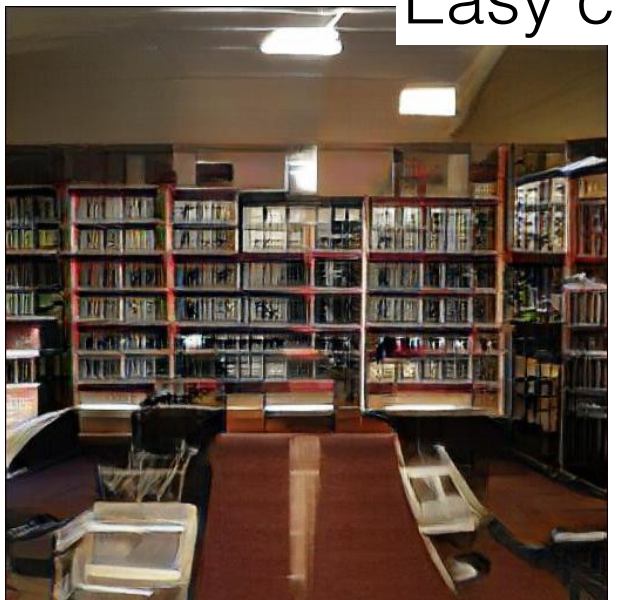





Easy classes

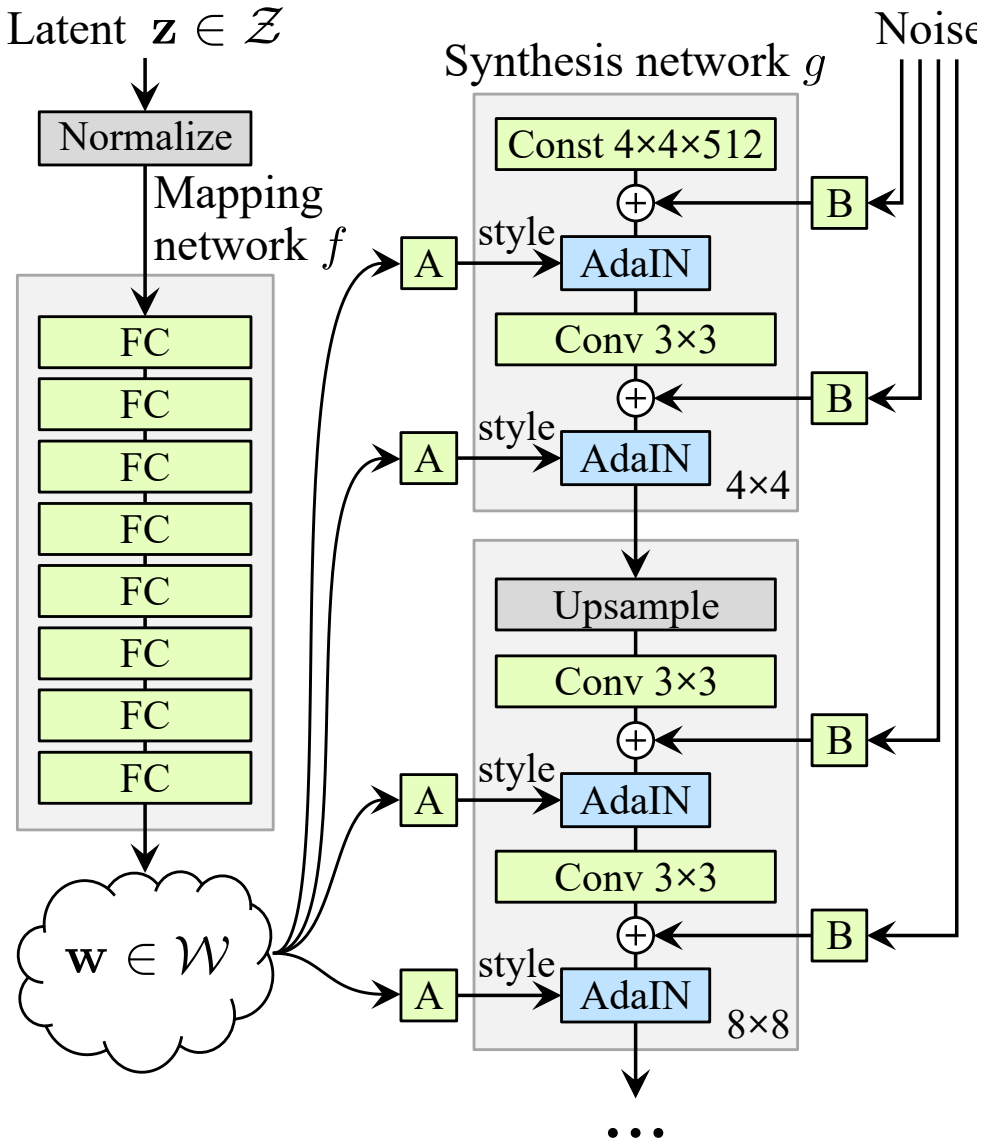
Hard classes

Resolution: 512x512

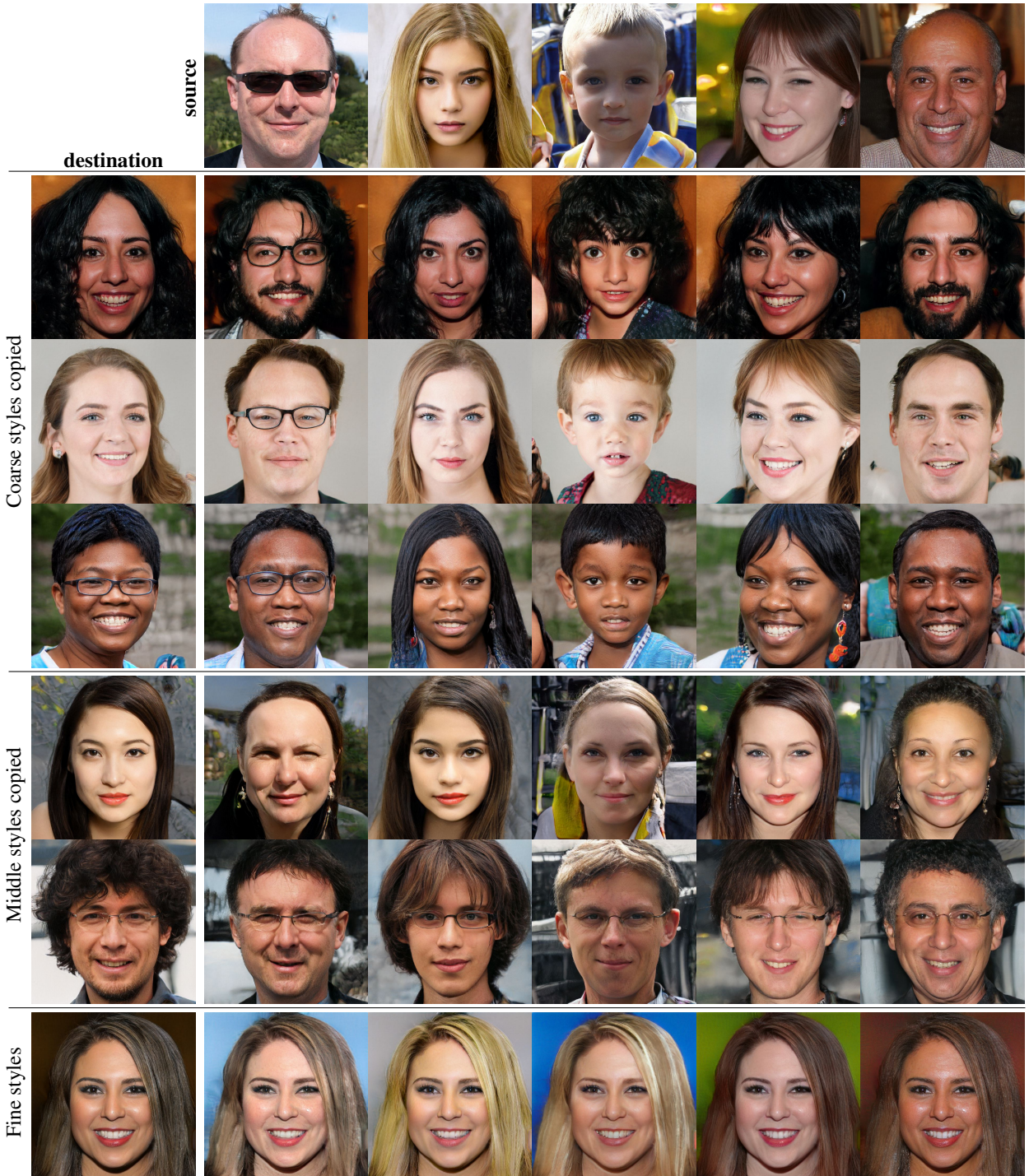



StyleGANs (Karras et al., 2019)

- A new architecture motivated by the style transfer networks
- allows unsupervised separation of high-level attributes and stochastic variation in the generated images

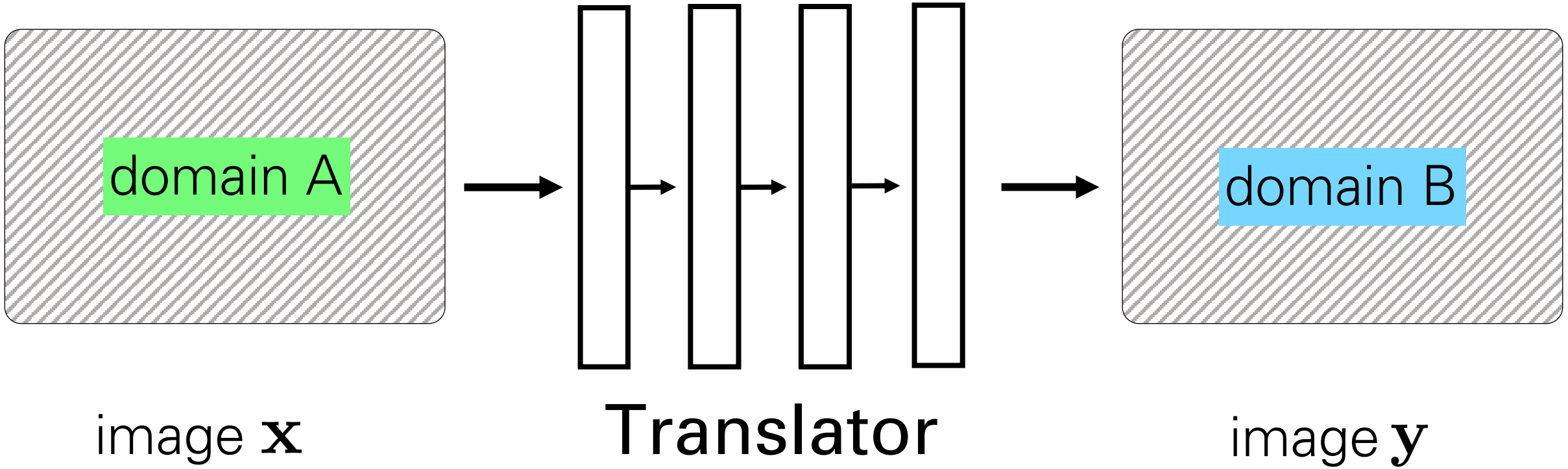


StyleGANs (Karras et al., 2018)



Applications of GANs

Image-to-Image Translation



- Learning to map images from one domain into another
- A general framework for both low and high-level image processing

Super-resolution

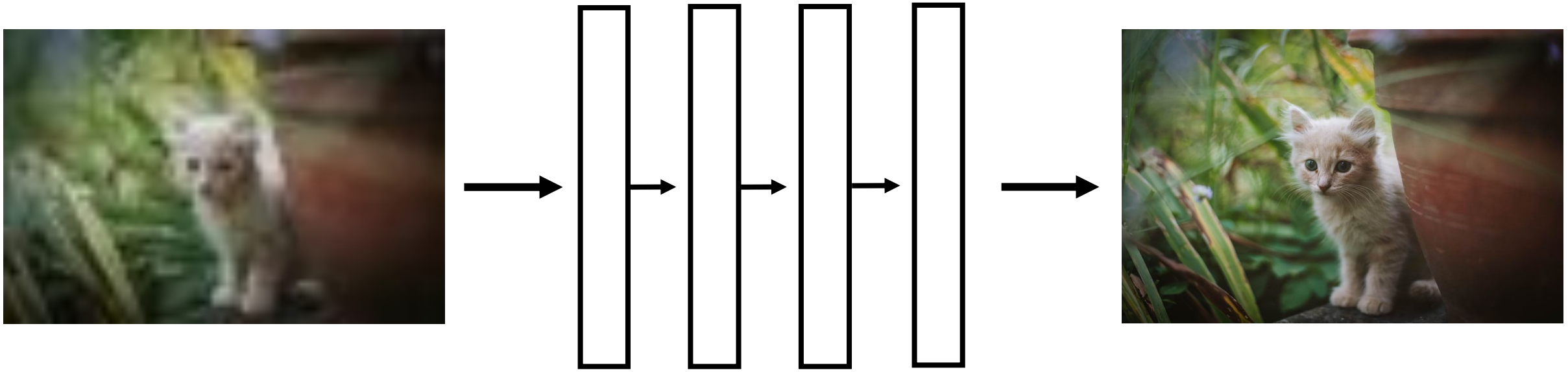


image $\mathbf{x}$

Translator

image $\mathbf{y}$

- **Input:** low-res image
  Output: high-res image

C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. CVPR 2017.

Colorization

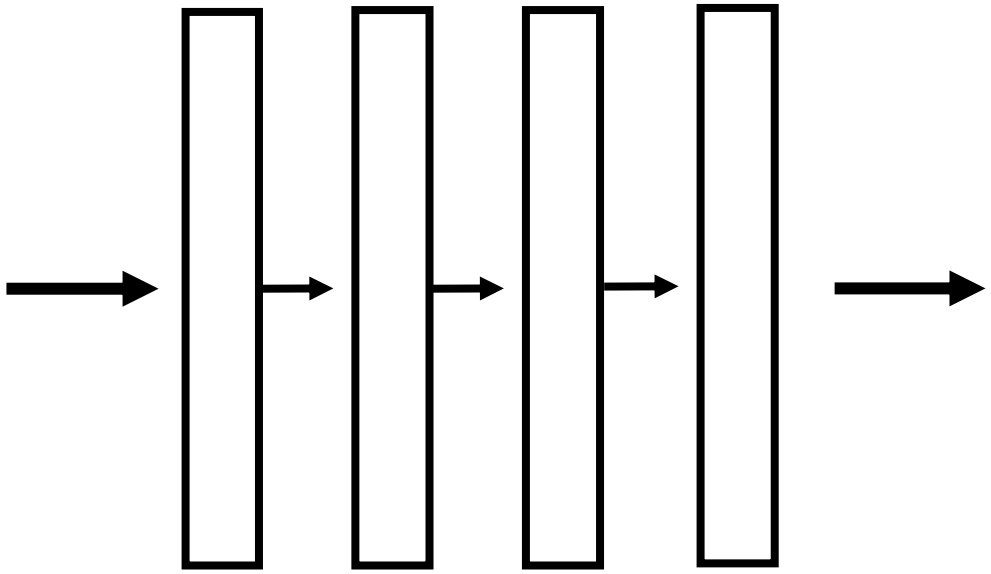


image $\mathbf{x}$ **Translator** image $\mathbf{y}$

- **Input:** black & white image
  Output: color image

Jason Antic. DeOldify: **A Deep Learning based project for colorizing and restoring old images**. 2018

Day to Night

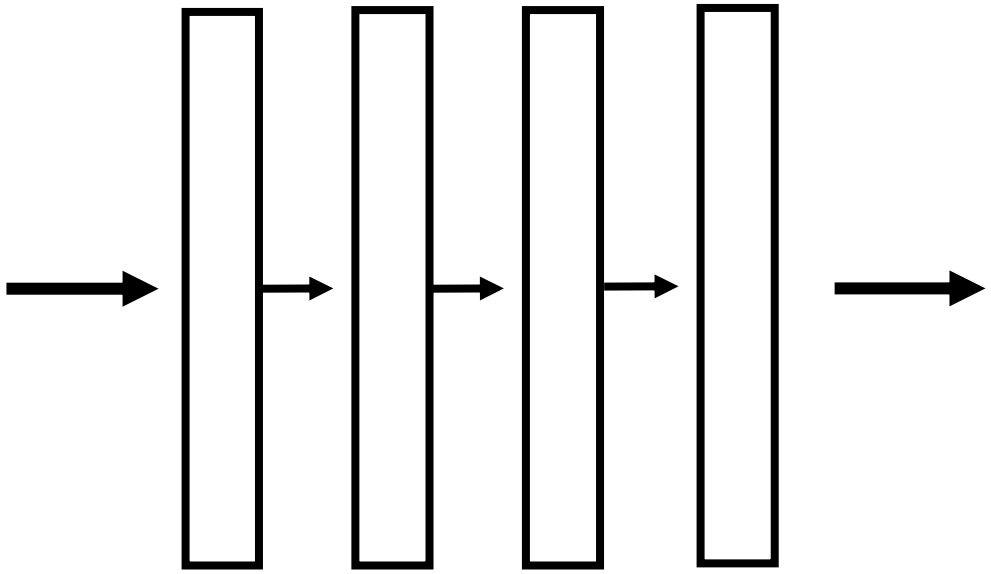


image $\mathbf{x}$

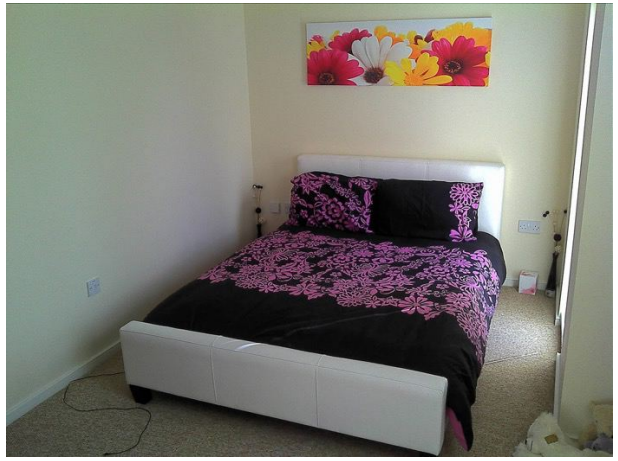
Translator

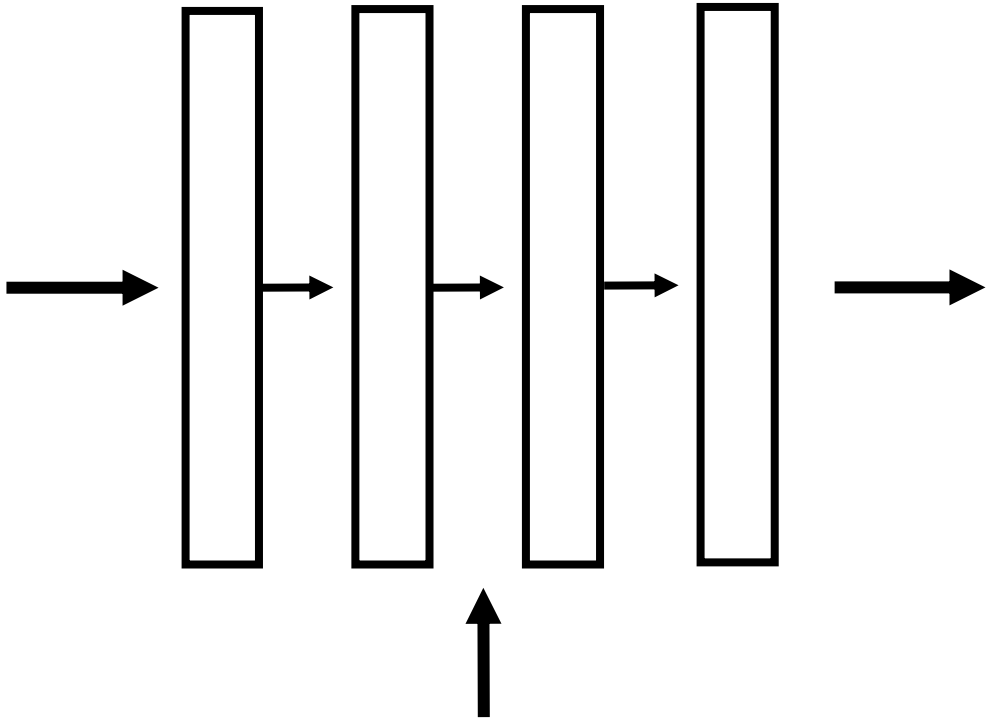
image $\mathbf{y}$

- **Input:** day image
  Output: night image

Ming-Yu Liu, Thomas Breuel, Jan Kautz. **Unsupervised Image-to-Image Translation Networks**. NIPS 2017.

Photo Style Transfer





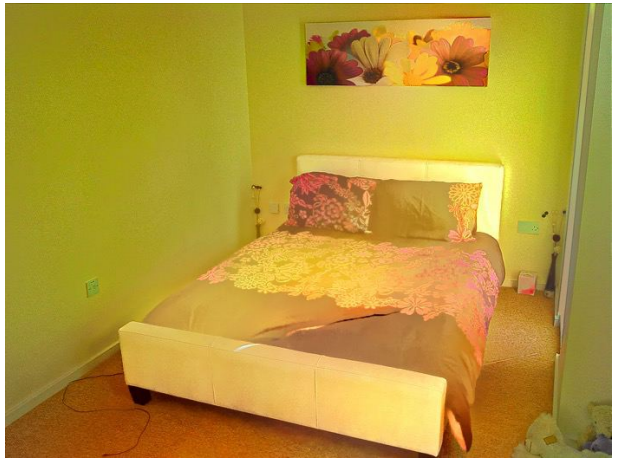


image $\mathbf{x}$

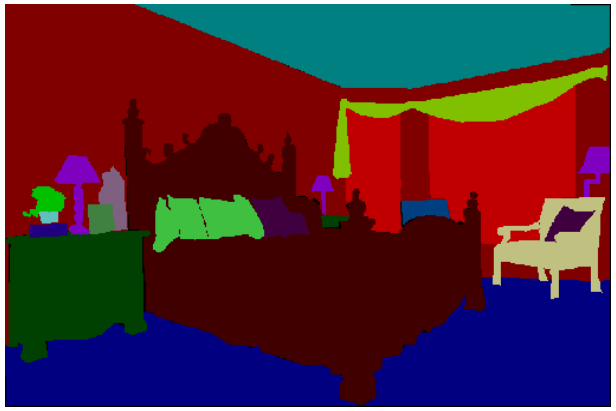
image $\mathbf{y}$

Target style image $\mathbf{t}$

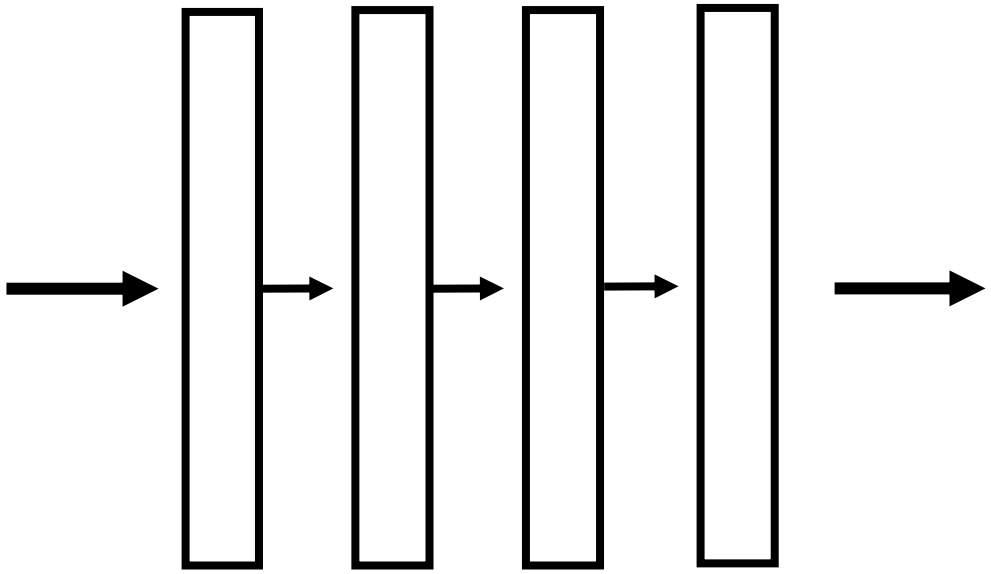
- **Input:** input image + target style image
  Output: synthesized image

Fujun Luan, Sylvain Paris, Eli Shechtman, Kavita Bala. **Deep Photo Style Transfer**. CVPR 2017.

Semantic Image Synthesis



semantic layout $\mathbf{x}$

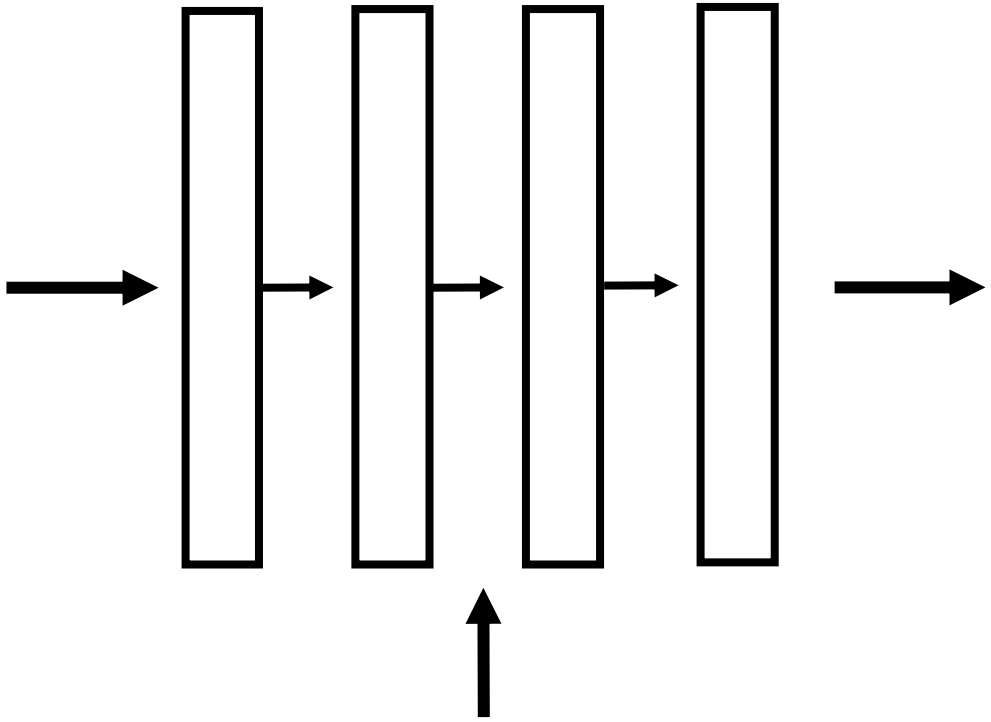


synthesized image $\mathbf{y}$

- **Input:** input layout
  Output: synthesized image

T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro. **High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs**. CVPR 2018.

Semantic Image Synthesis



semantic layout $\mathbf{x}$

synthesized image $\mathbf{y}$

Target style image $\mathbf{t}$

- **Input:** input layout + target style image
  Output: synthesized image

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu. **Semantic Image Synthesis with Spatially-Adaptive Normalization**. CVPR 2020.

Semantic Image Editing

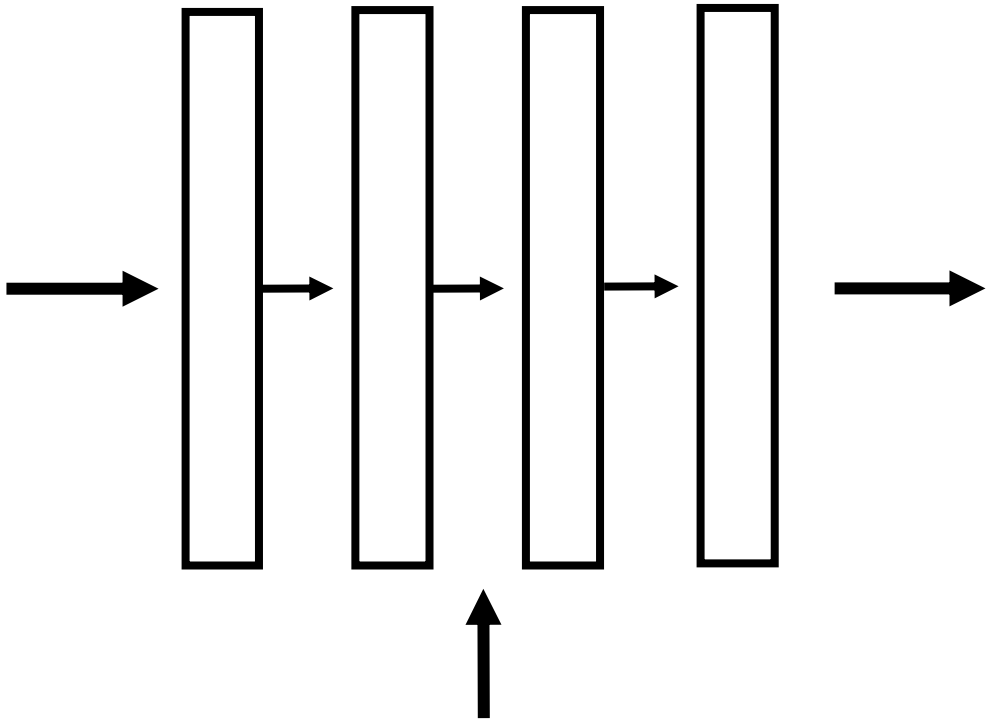


image $\mathbf{x}$

manipulated image $\mathbf{y}$

"more flowers"
"more cloudy"

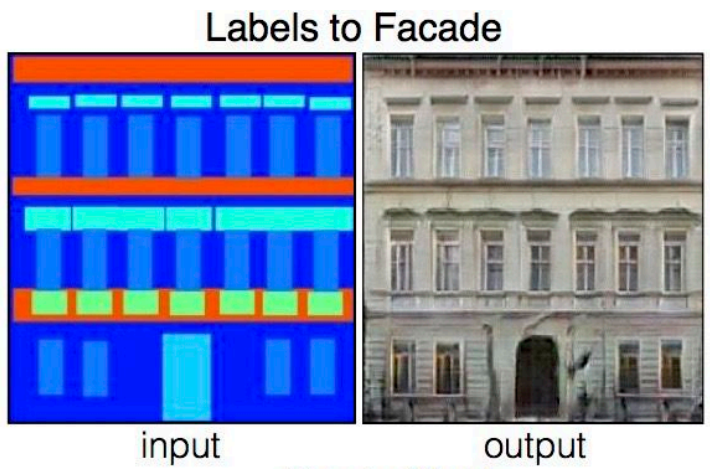
Target scene attribute(s) $\mathbf{t}$

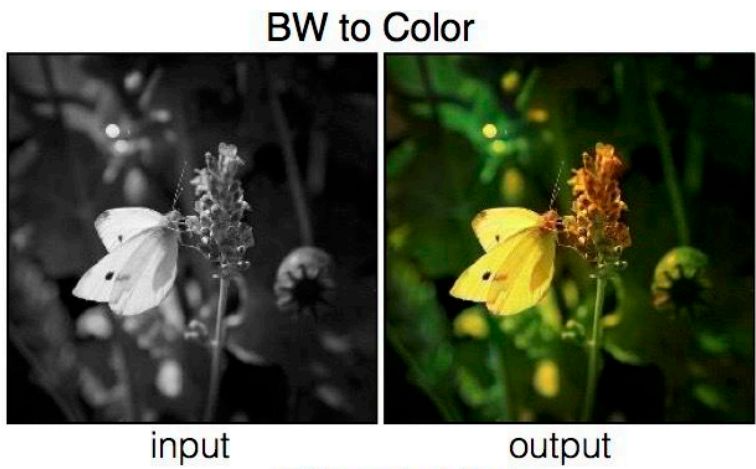
- **Input:** input image
  (+ semantic layout)
  + target attribute
  Output: manipulated image

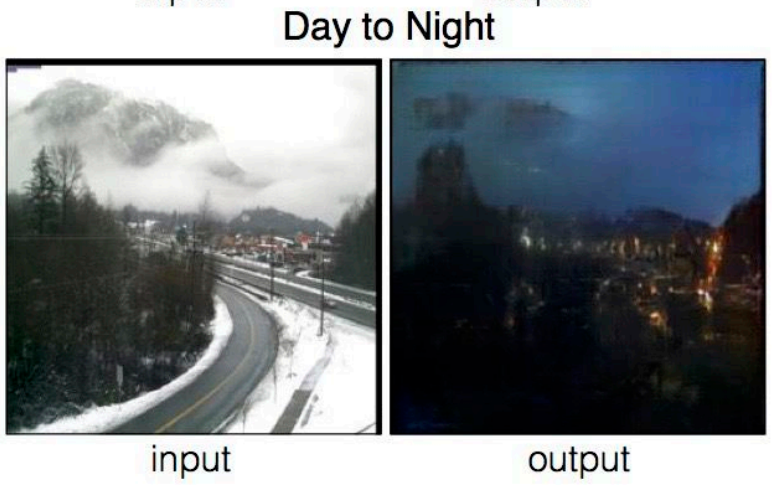
Levent Karacan, Zeynep Akata, Aykut Erdem, Erkut Erdem. **Manipulating Attributes of Natural Scenes via Hallucination**. ACM TOG 2020.

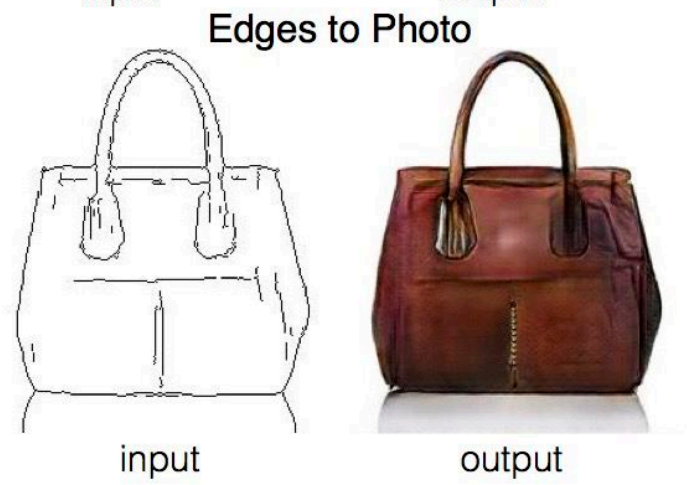
Image to Image Translation (pix2pix) (Isola et al., 2016)

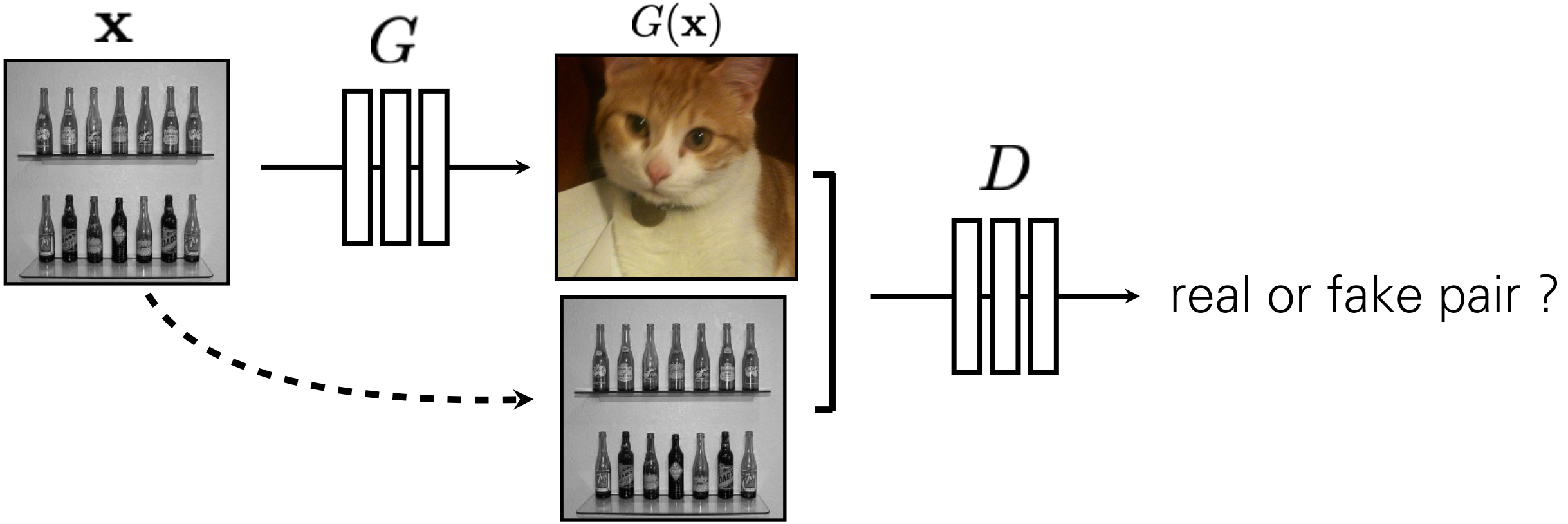
- Requires paired training data





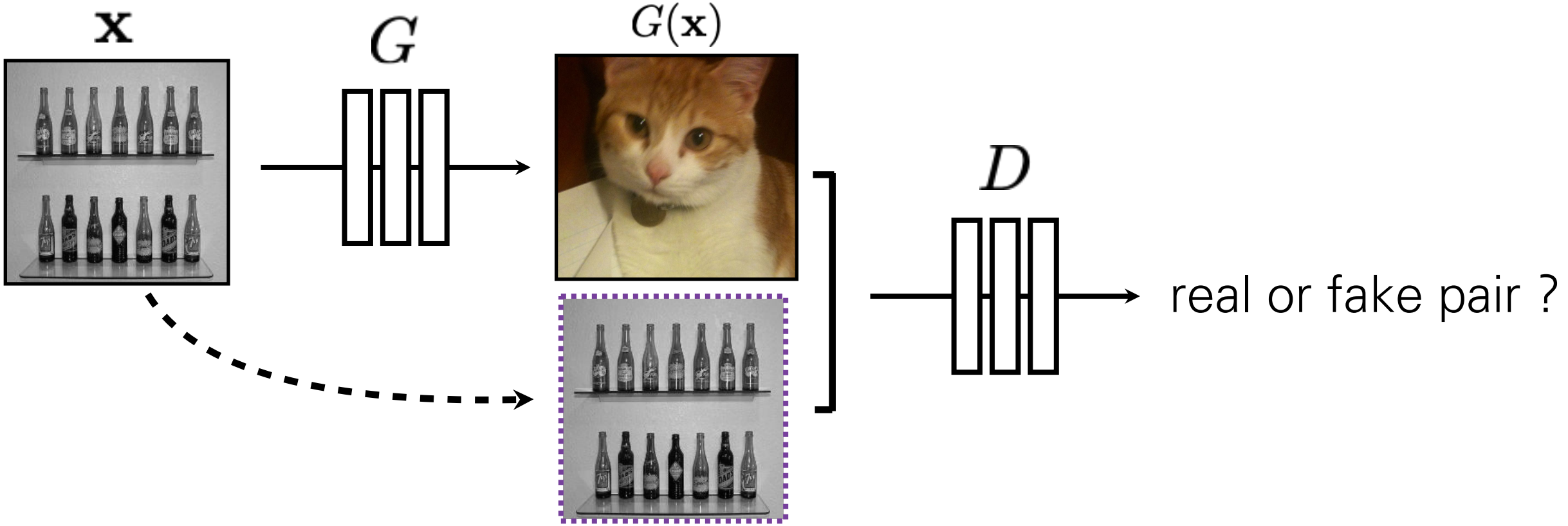






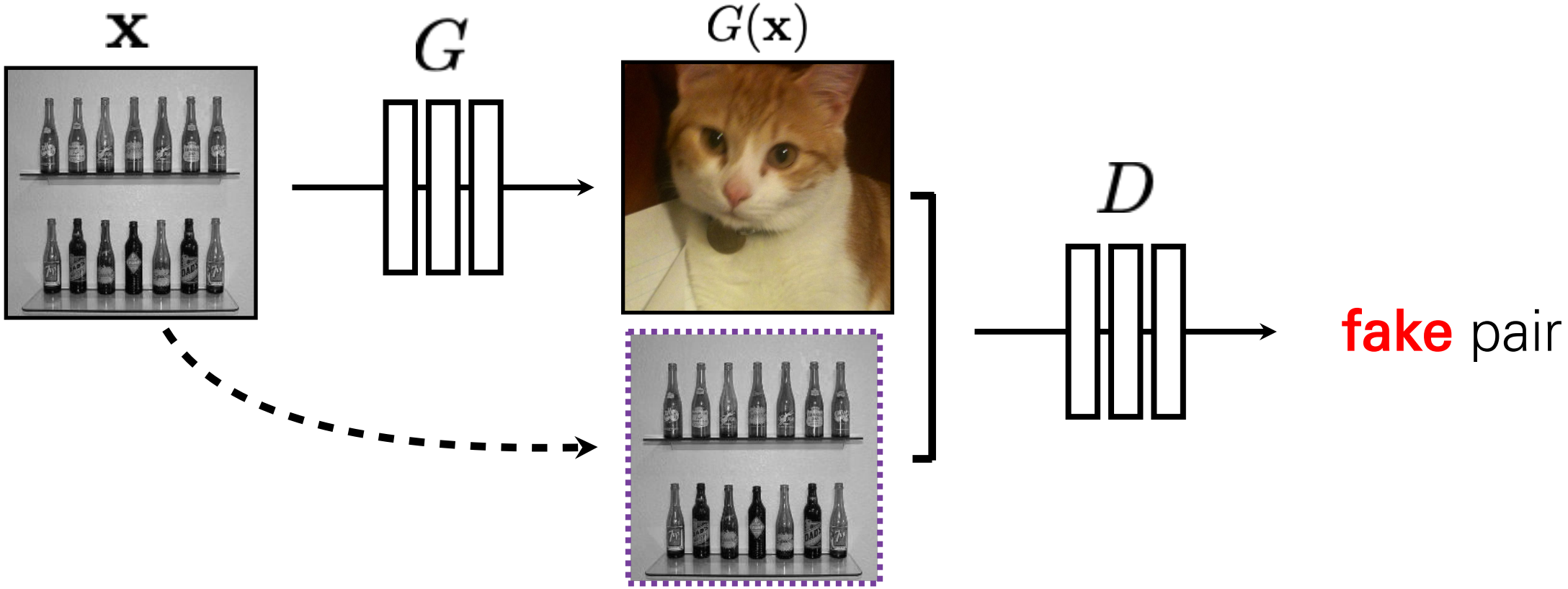
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$

Slide Credit: Philip Isola



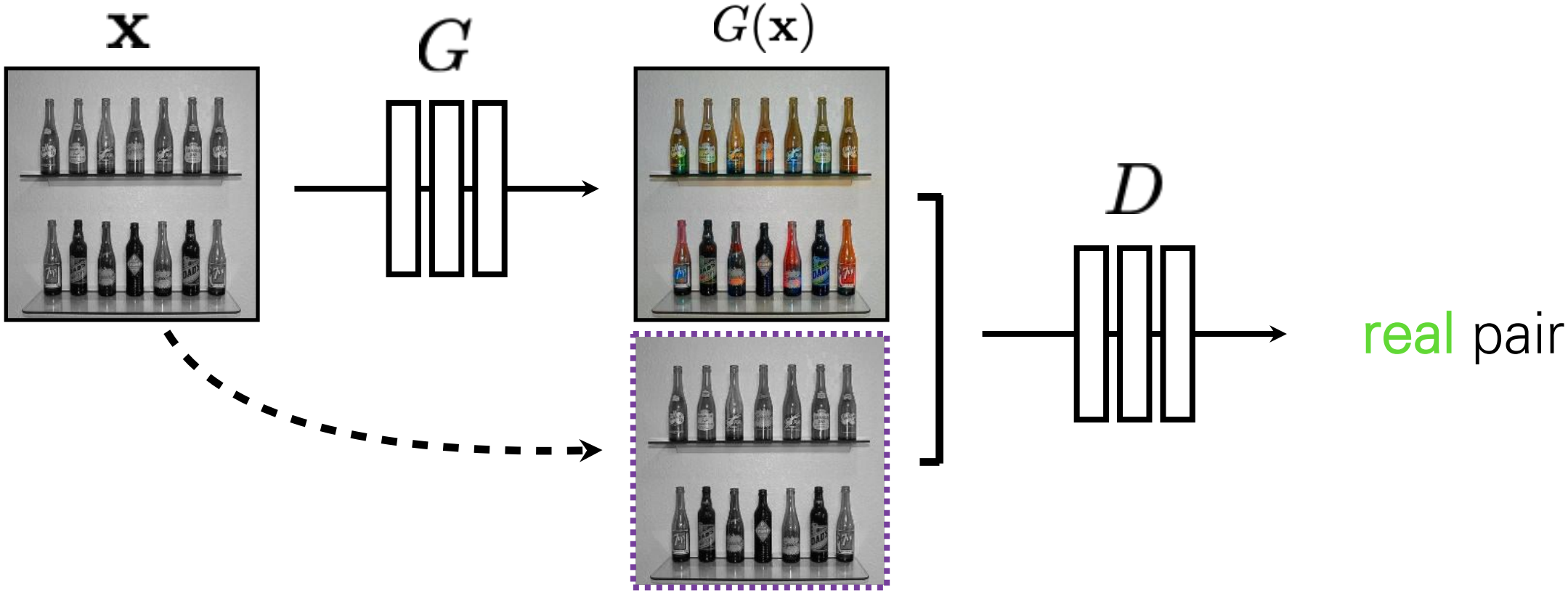
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Slide Credit: Philip Isola



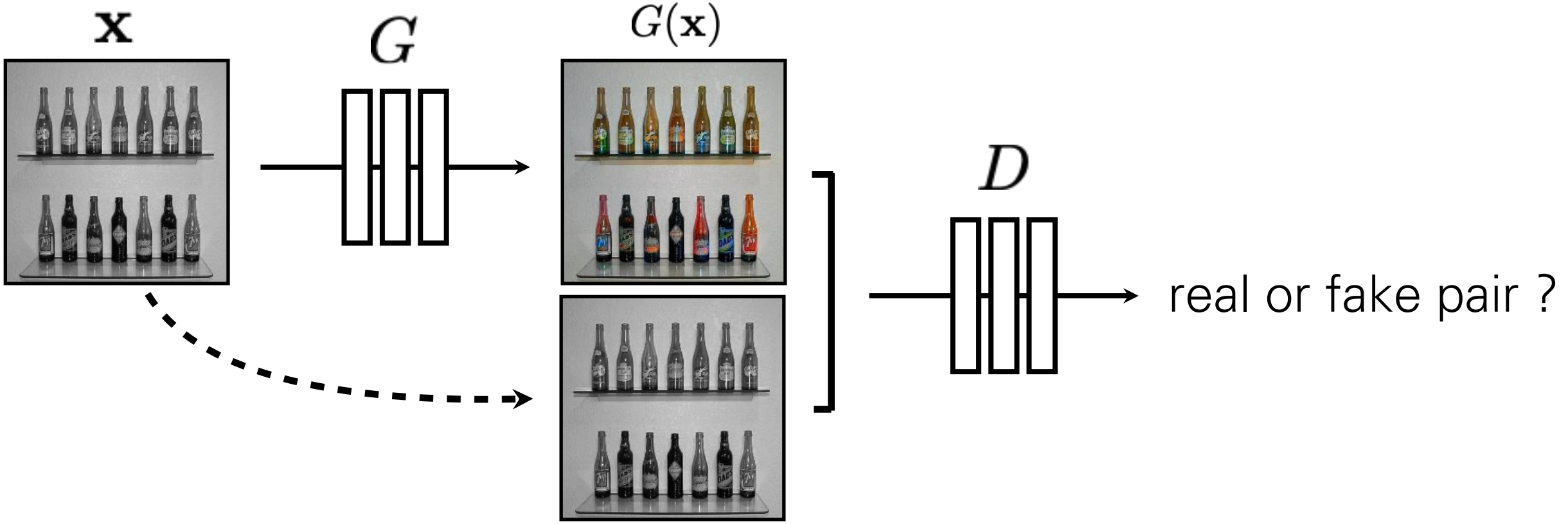
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Slide Credit: Philip Isola



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

Slide Credit: Philip Isola



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

Slide Credit: Philip Isola

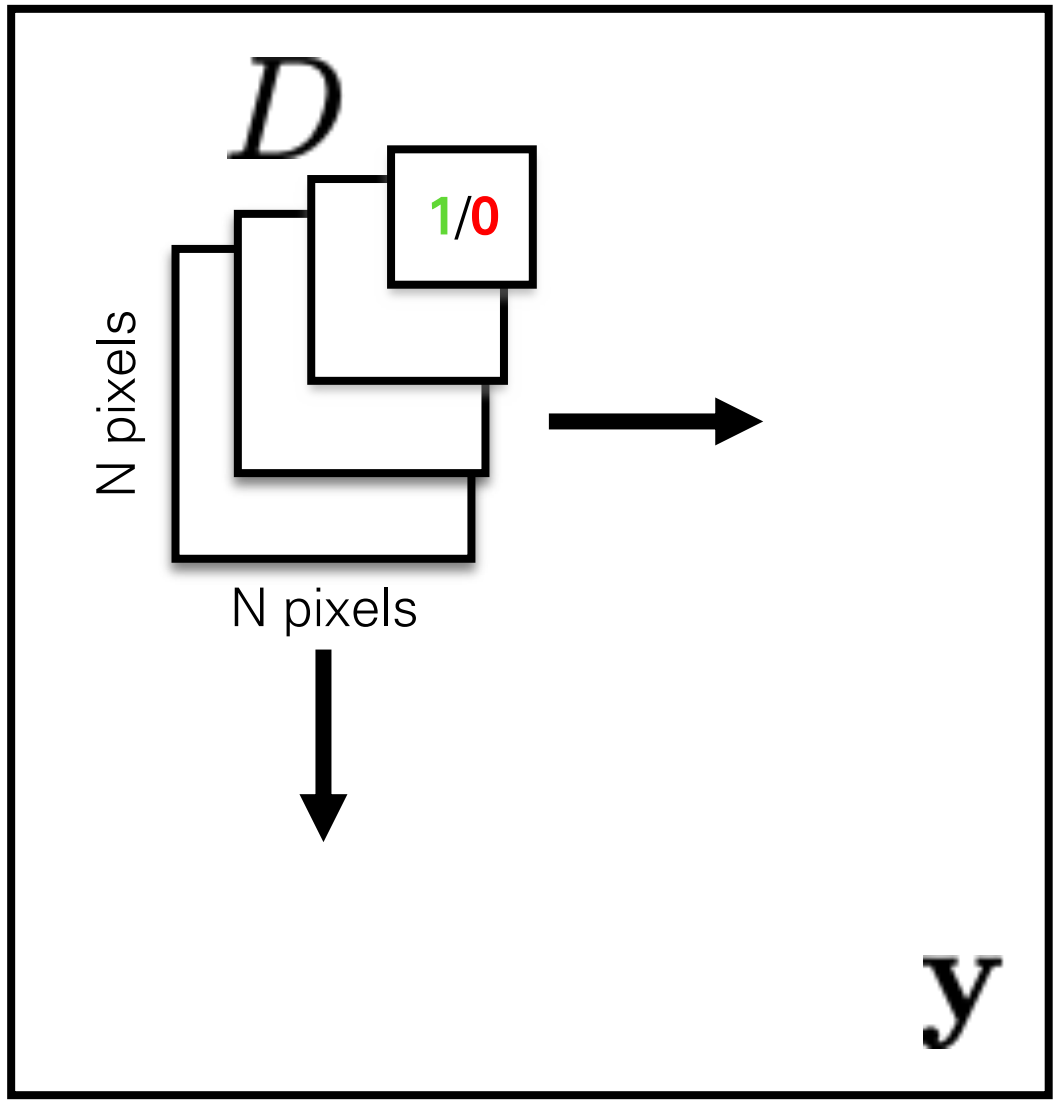
BW → Color

Input Output Input Output Input Output

Data from [Russakovsky et al. 2015]

Slide Credit: Philip Isola

Shrinking the capacity: Patch Discriminator



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

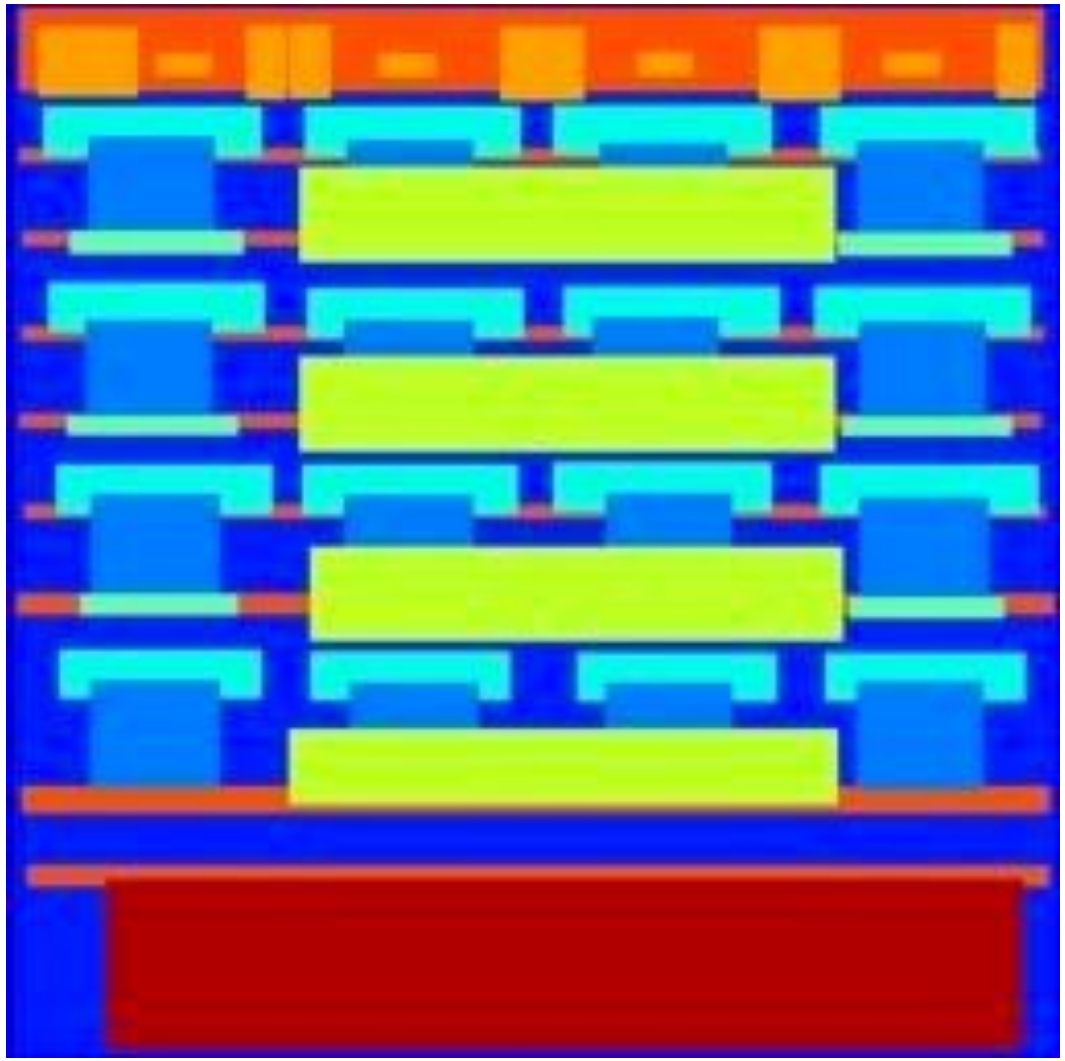
- Faster, fewer parameters
- More supervised observations
- Applies to arbitrarily large images

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

Slide Credit: Philip Isola

Labels → Facades

Input



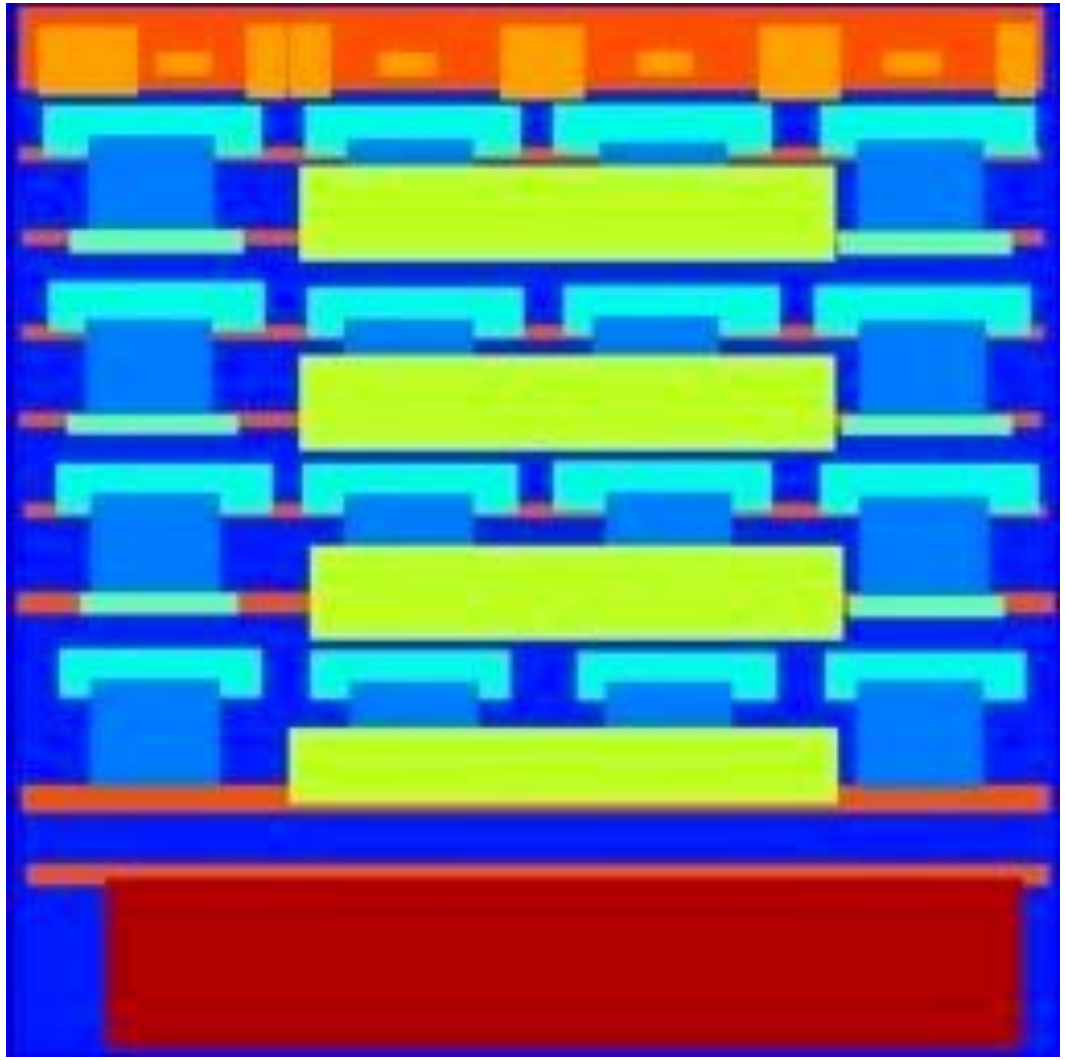
1x1 Discriminator

Data from [Tylecek, 2013]

Slide Credit: Philip Isola

Labels → Facades

Input



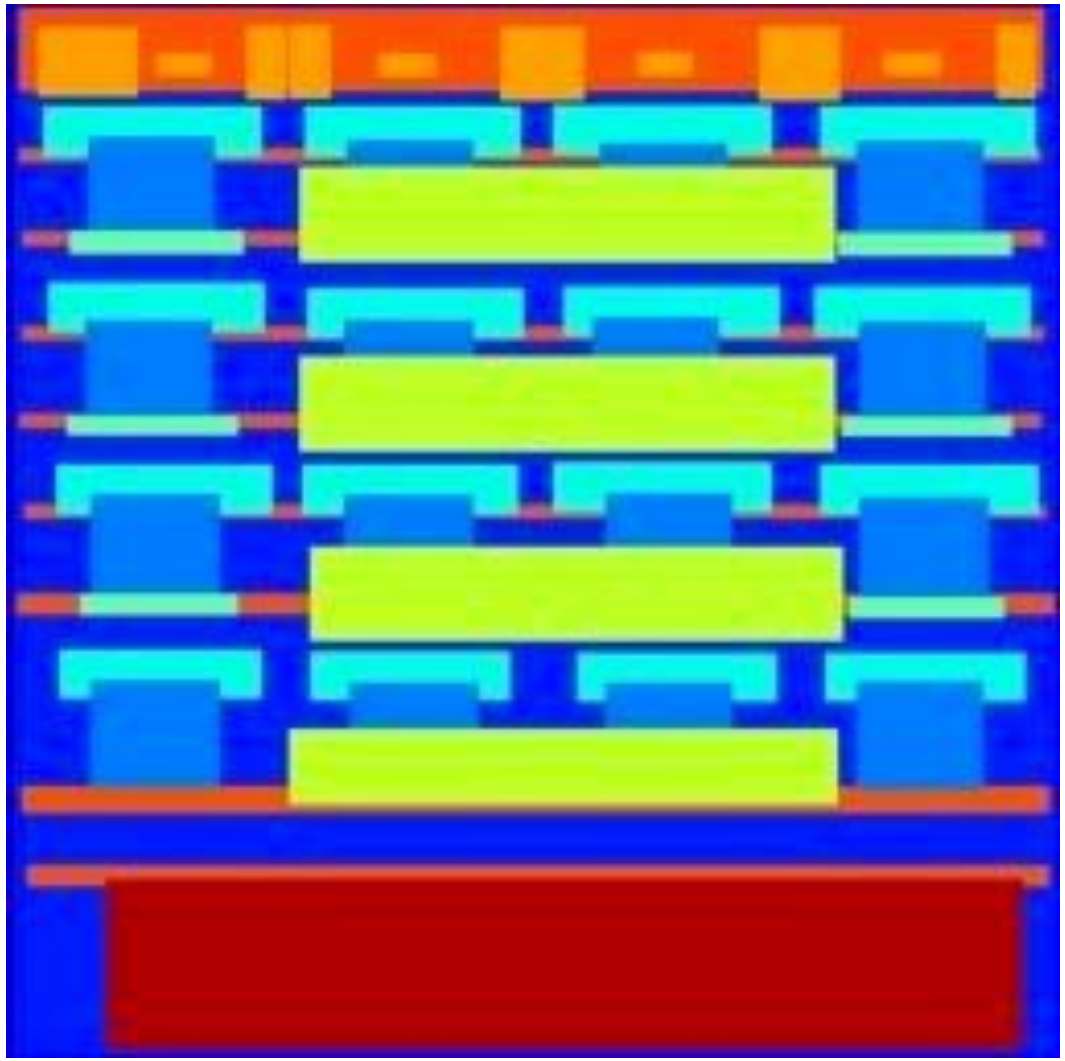
16x16 Discriminator

Data from [Tylecek, 2013]

Slide Credit: Philip Isola

Labels → Facades

Input



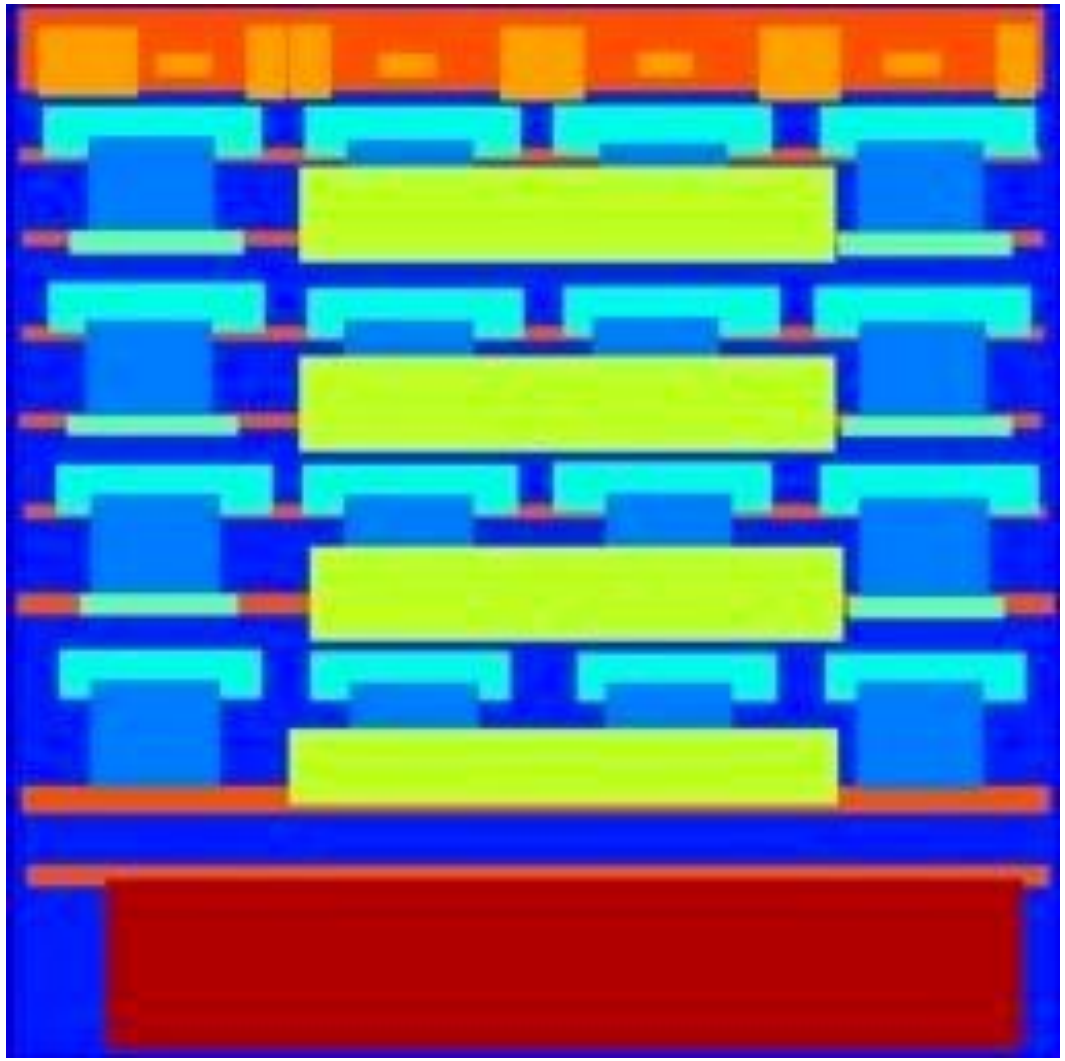
70x70 Discriminator

Data from [Tylecek, 2013]

Slide Credit: Philip Isola

Labels → Facades

Input

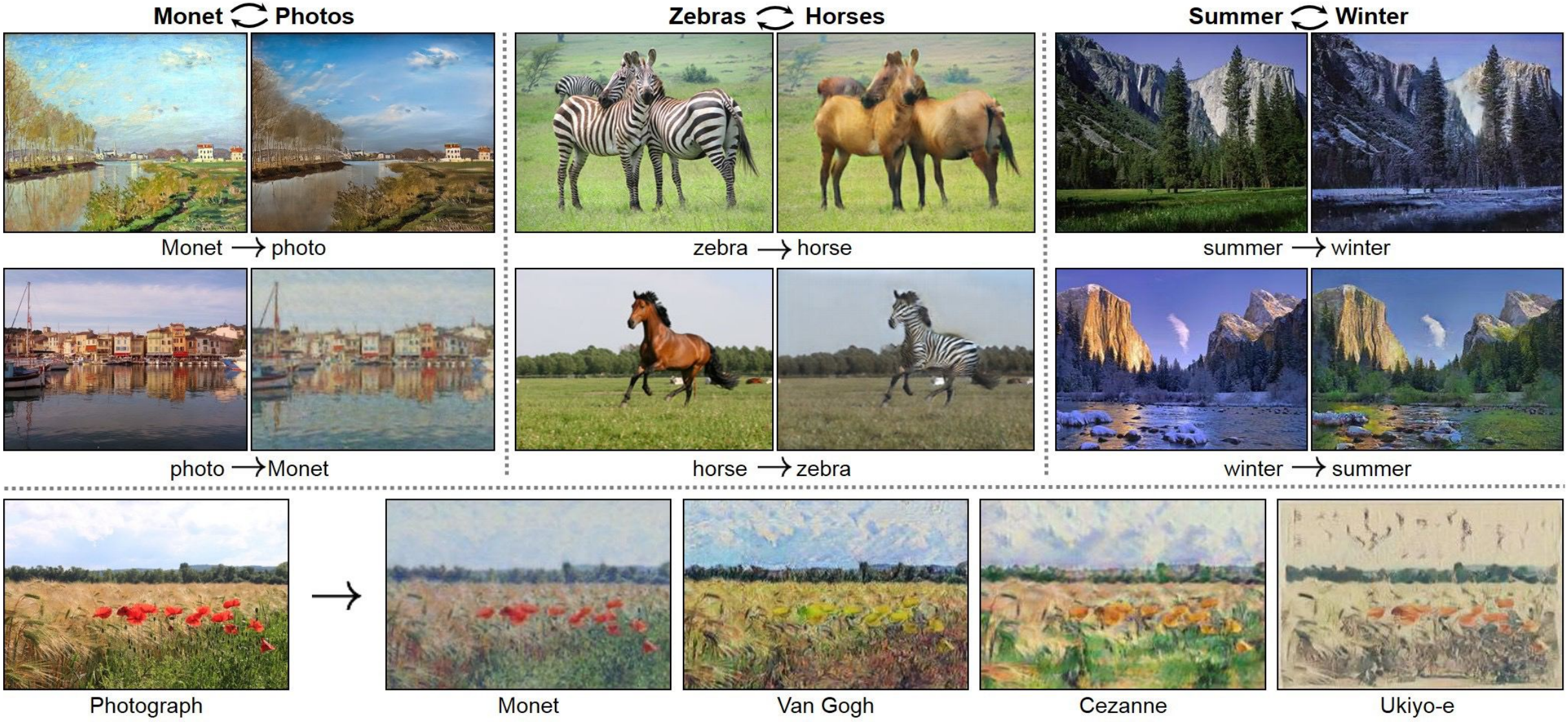


Full image Discriminator

Data from [Tylecek, 2013]

Slide Credit: Philip Isola

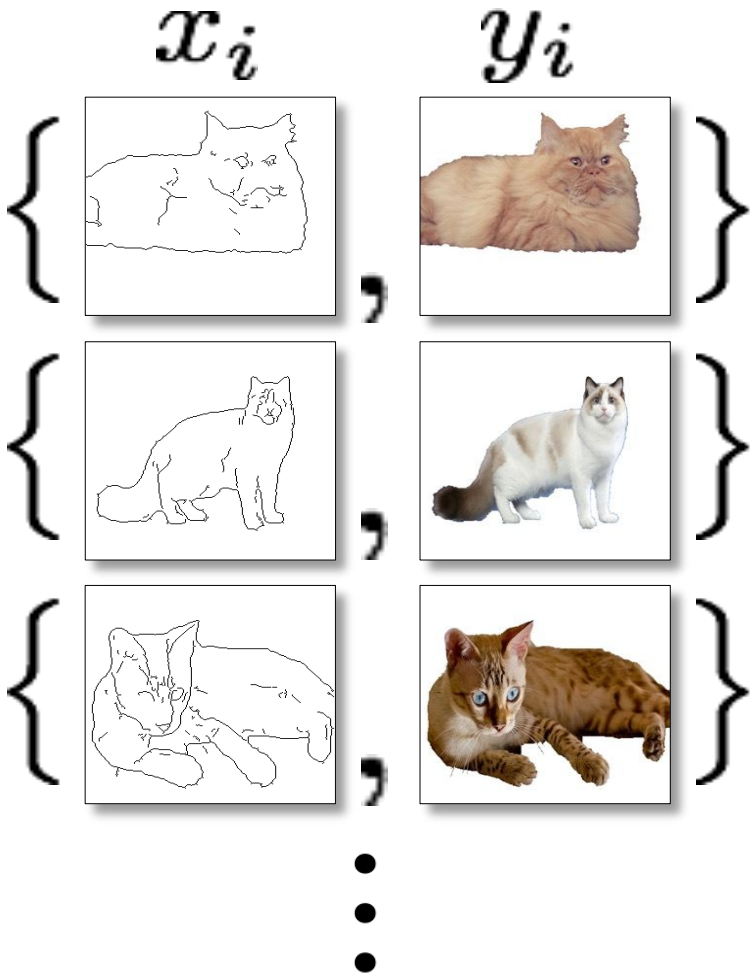
CycleGAN: Pix2Pix w/o input-output pairs



(Zhu et al. 2017)

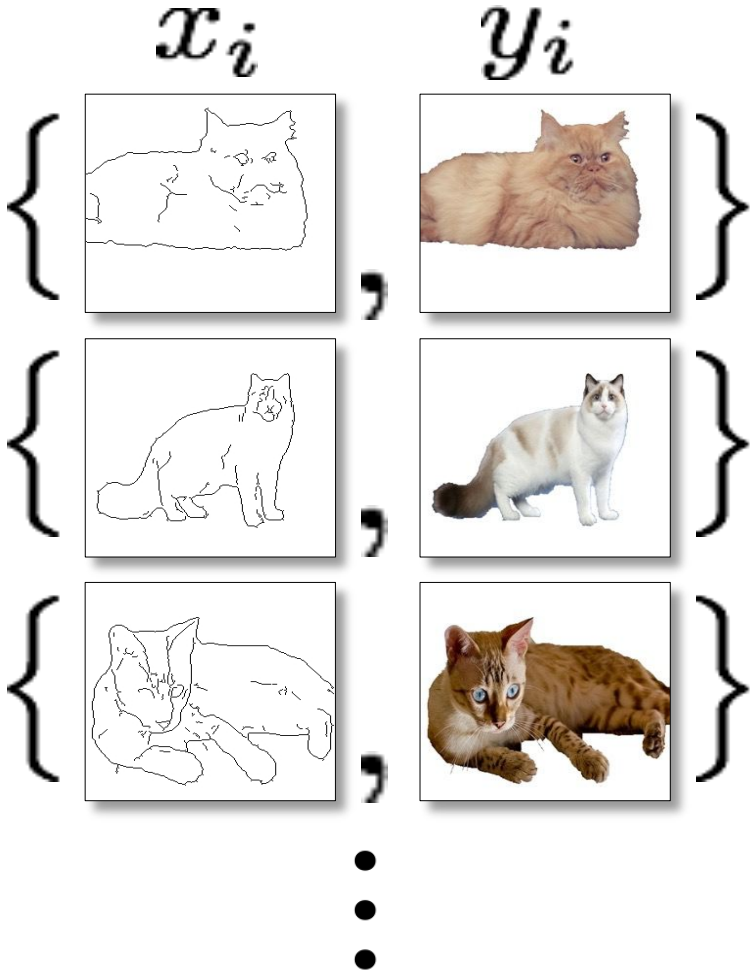
Slide Credit: Taesung Park and Jun-Yan Zhu

Paired data

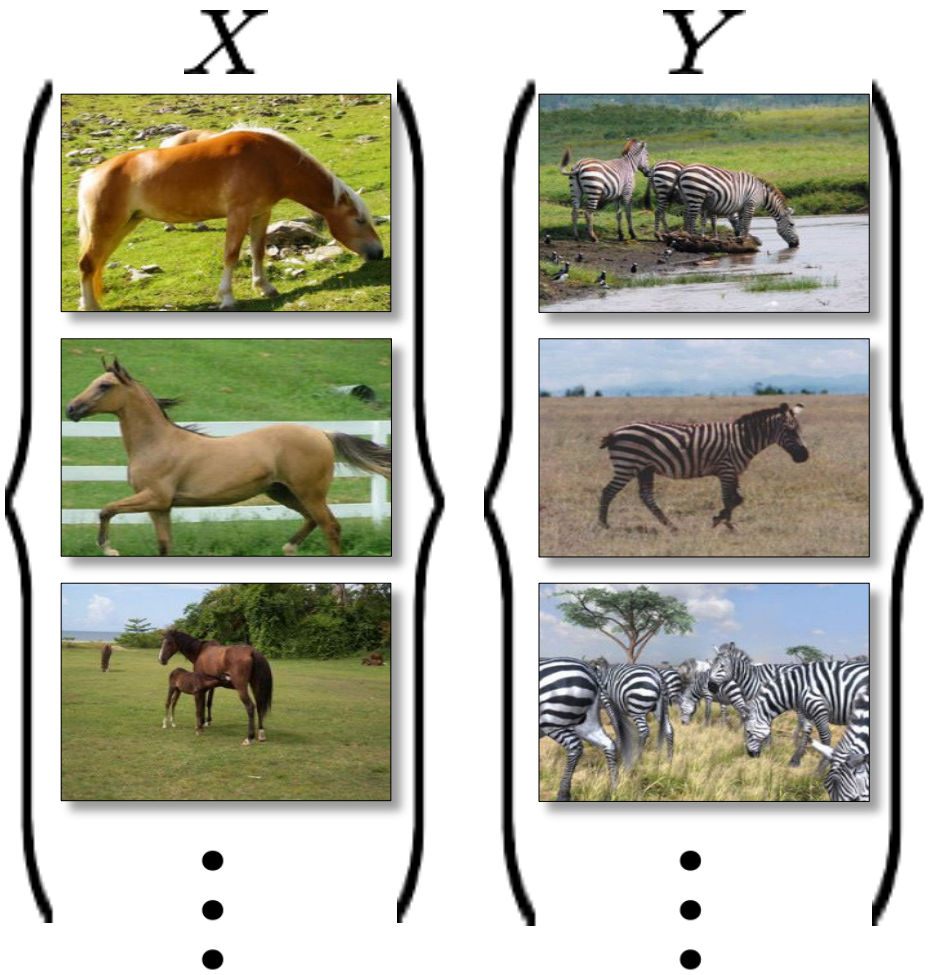


Slide Credit: Taesung Park and Jun-Yan Zhu

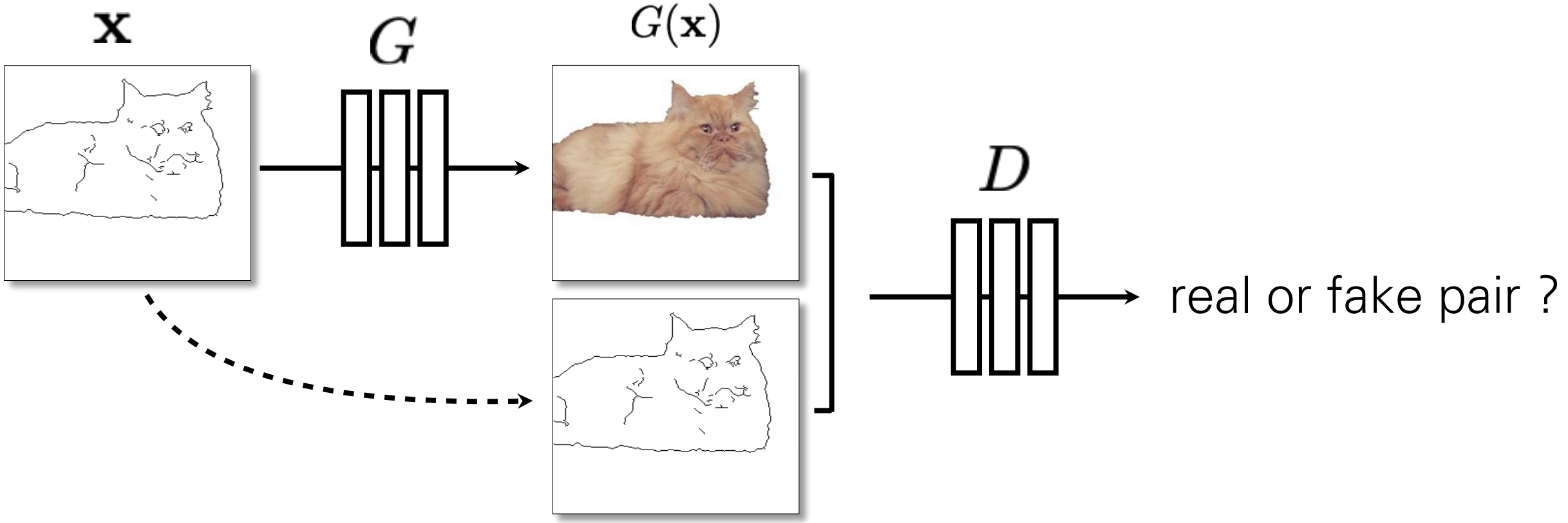
Paired data



Unpaired data

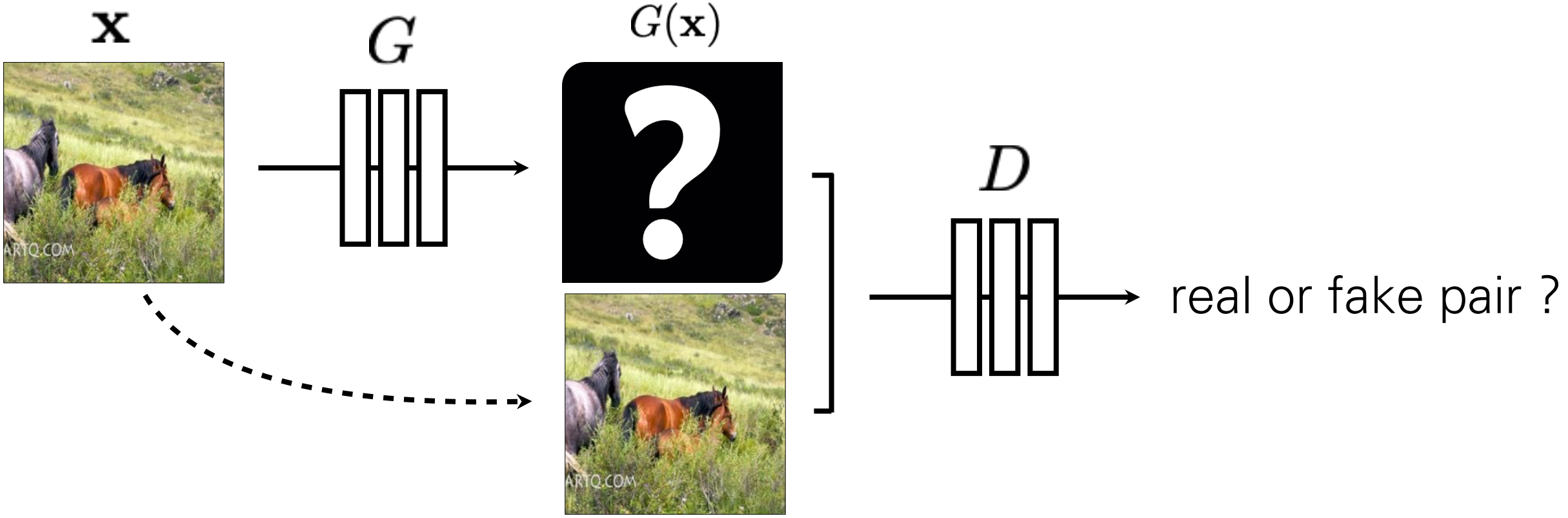


Slide Credit: Taesung Park and Jun-Yan Zhu



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

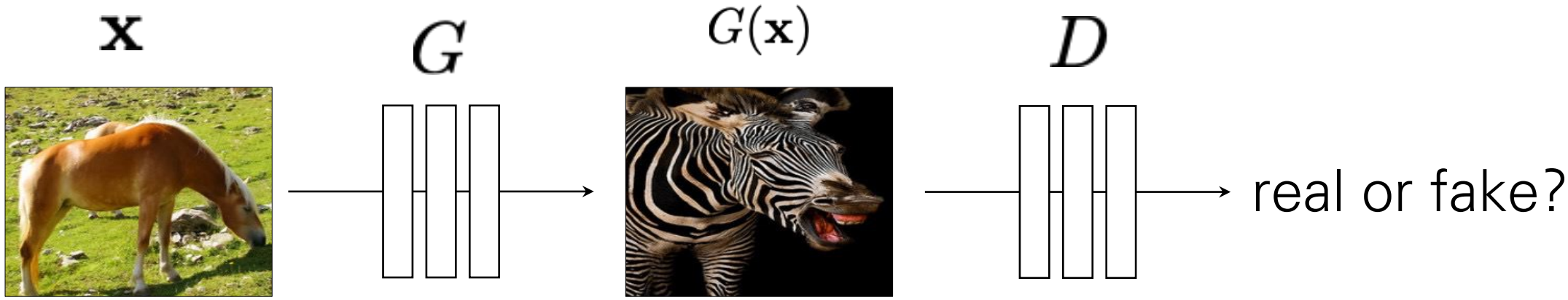
Slide Credit: Taesung Park and Jun-Yan Zhu



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

No input-output pairs!

Slide Credit: Taesung Park and Jun-Yan Zhu

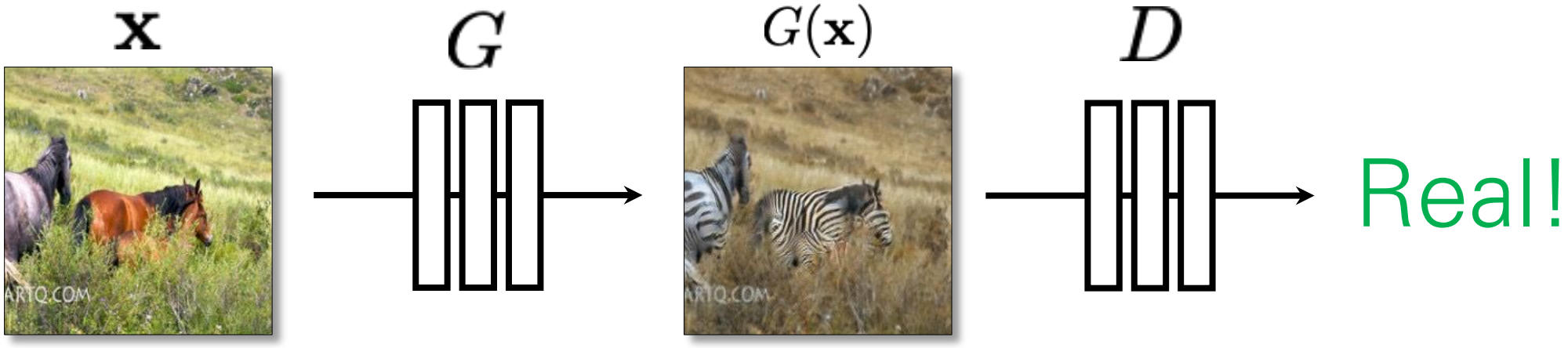


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$

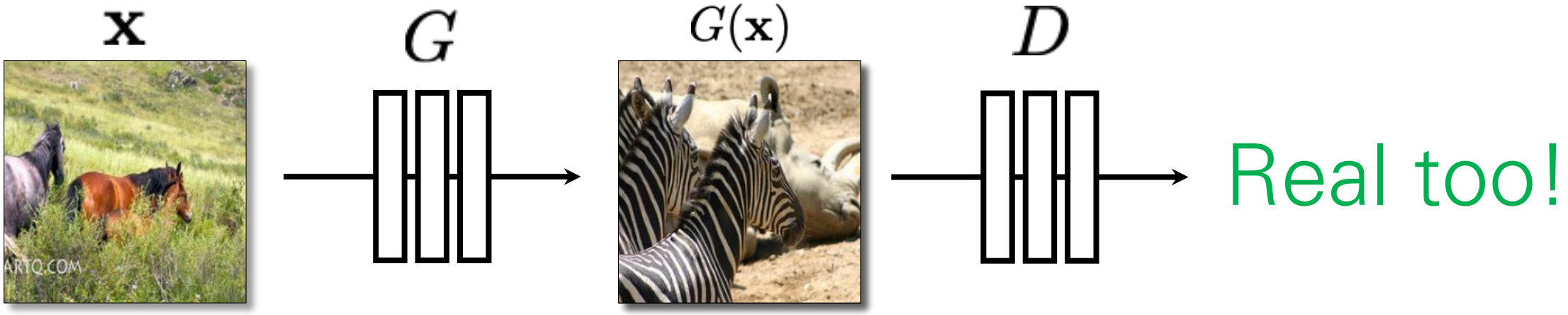
Usually loss functions check if output matches a target instance

GAN loss checks if output is part of an admissible set

Slide Credit: Taesung Park and Jun-Yan Zhu



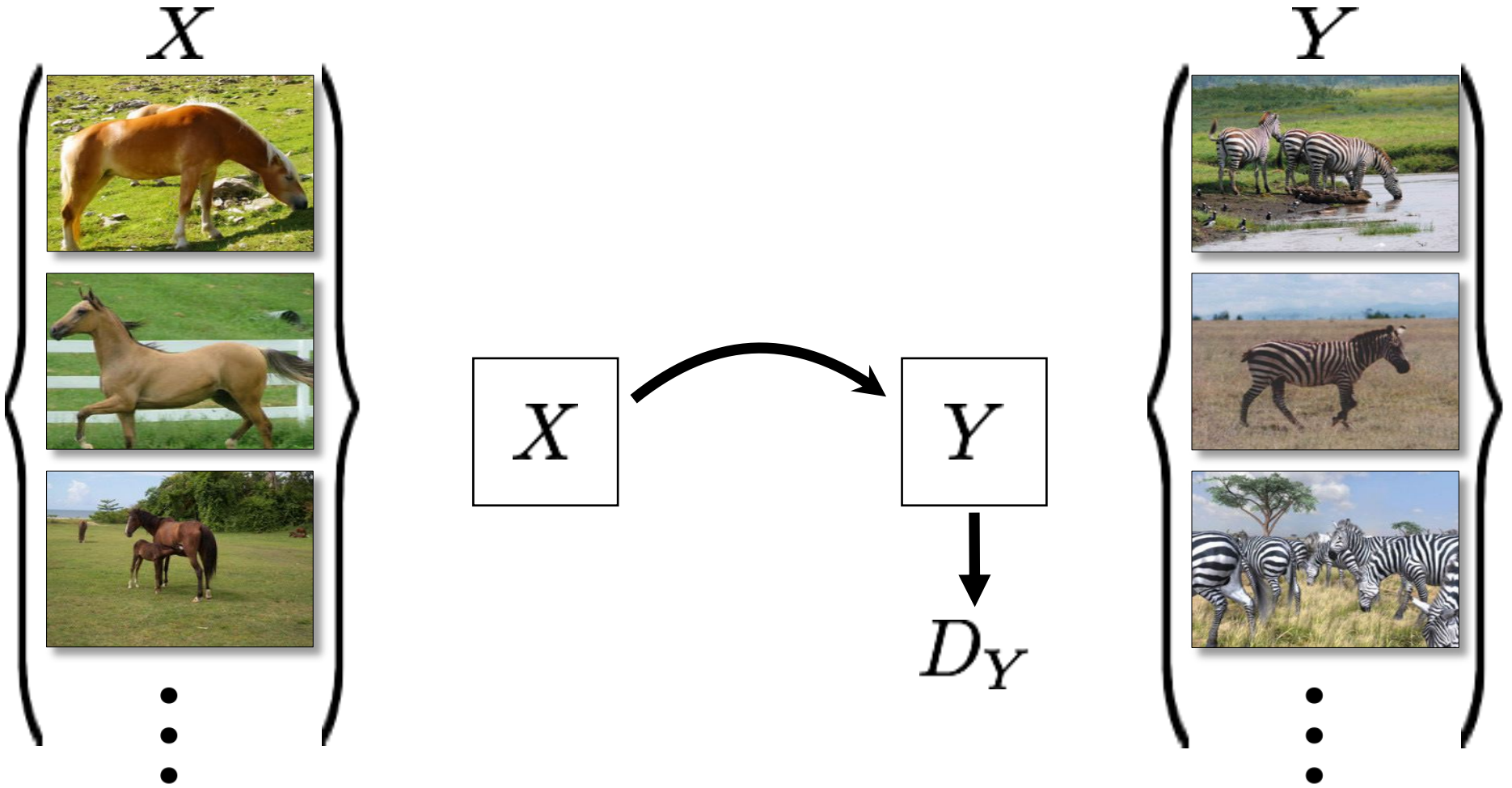
Slide Credit: Jun-Yan Zhu



Nothing to force output to correspond to input

Slide Credit: Taesung Park and Jun-Yan Zhu

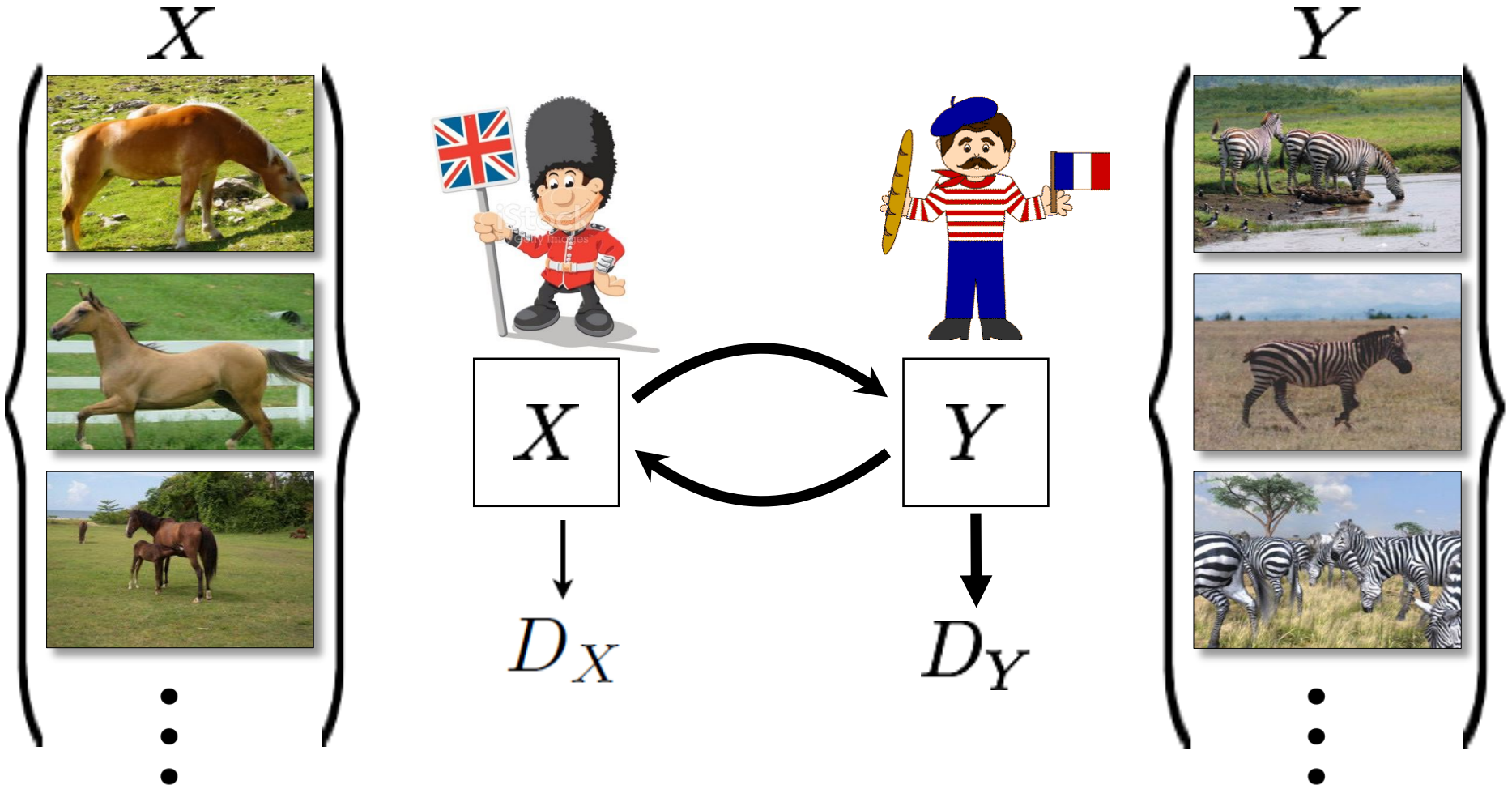
Cycle-Consistent Adversarial Networks



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

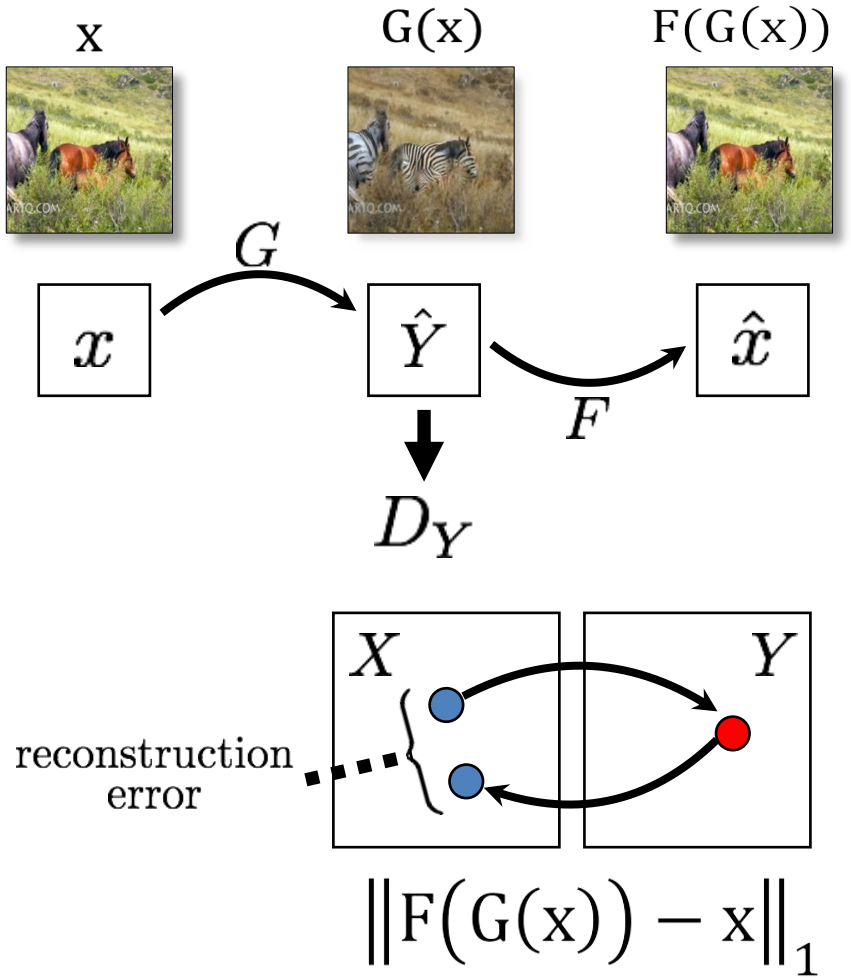
Slide Credit: Taesung Park and Jun-Yan Zhu

Cycle-Consistent Adversarial Networks



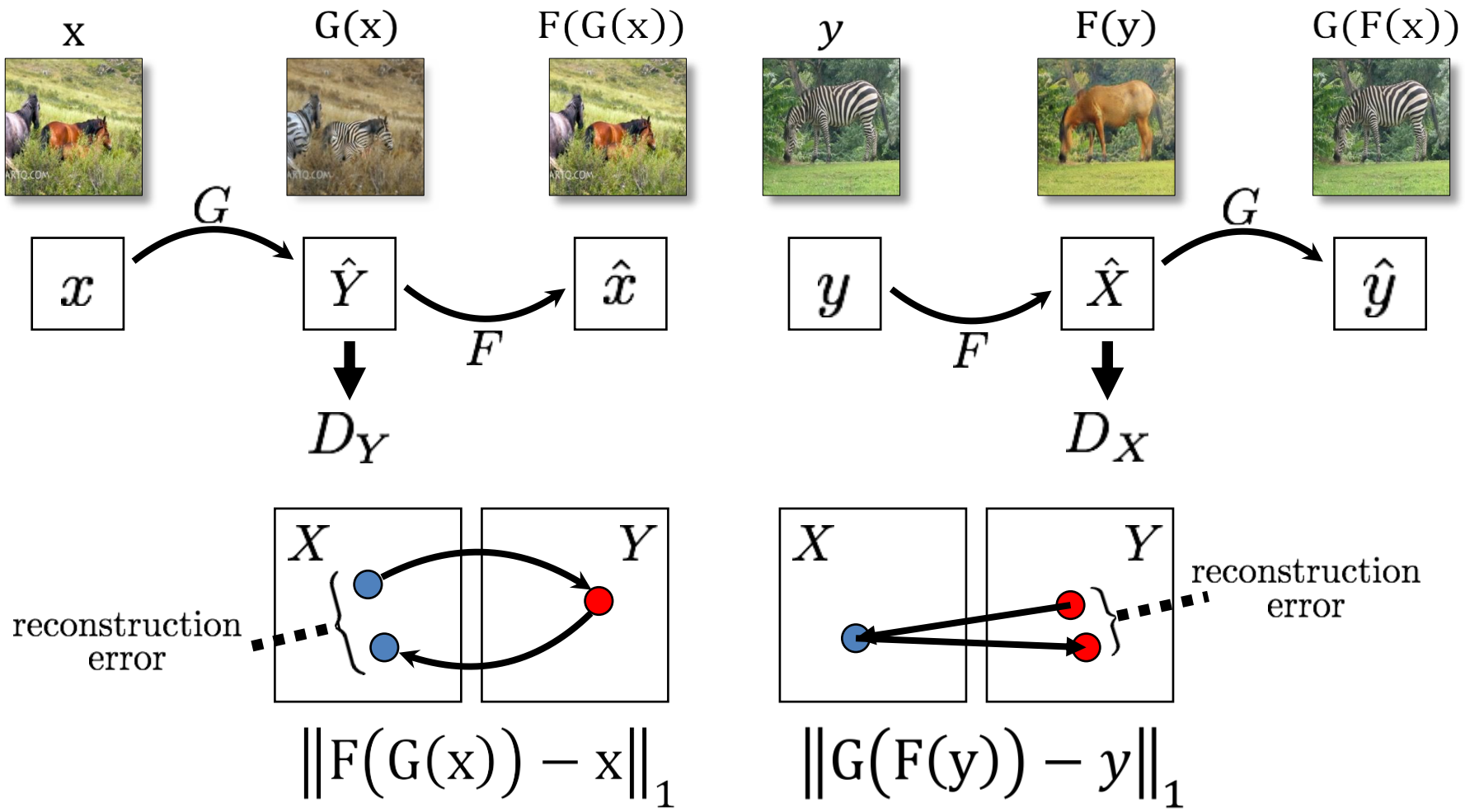
Slide Credit: Taesung Park and Jun-Yan Zhu

Cycle Consistency Loss



Slide Credit: Taesung Park and Jun-Yan Zhu

Cycle Consistency Loss



Slide Credit: Taesung Park and Jun-Yan Zhu

Slide Credit: Taesung Park and Jun-Yan Zhu

Slide Credit: Taesung Park and Jun-Yan Zhu

Monet's paintings → photos

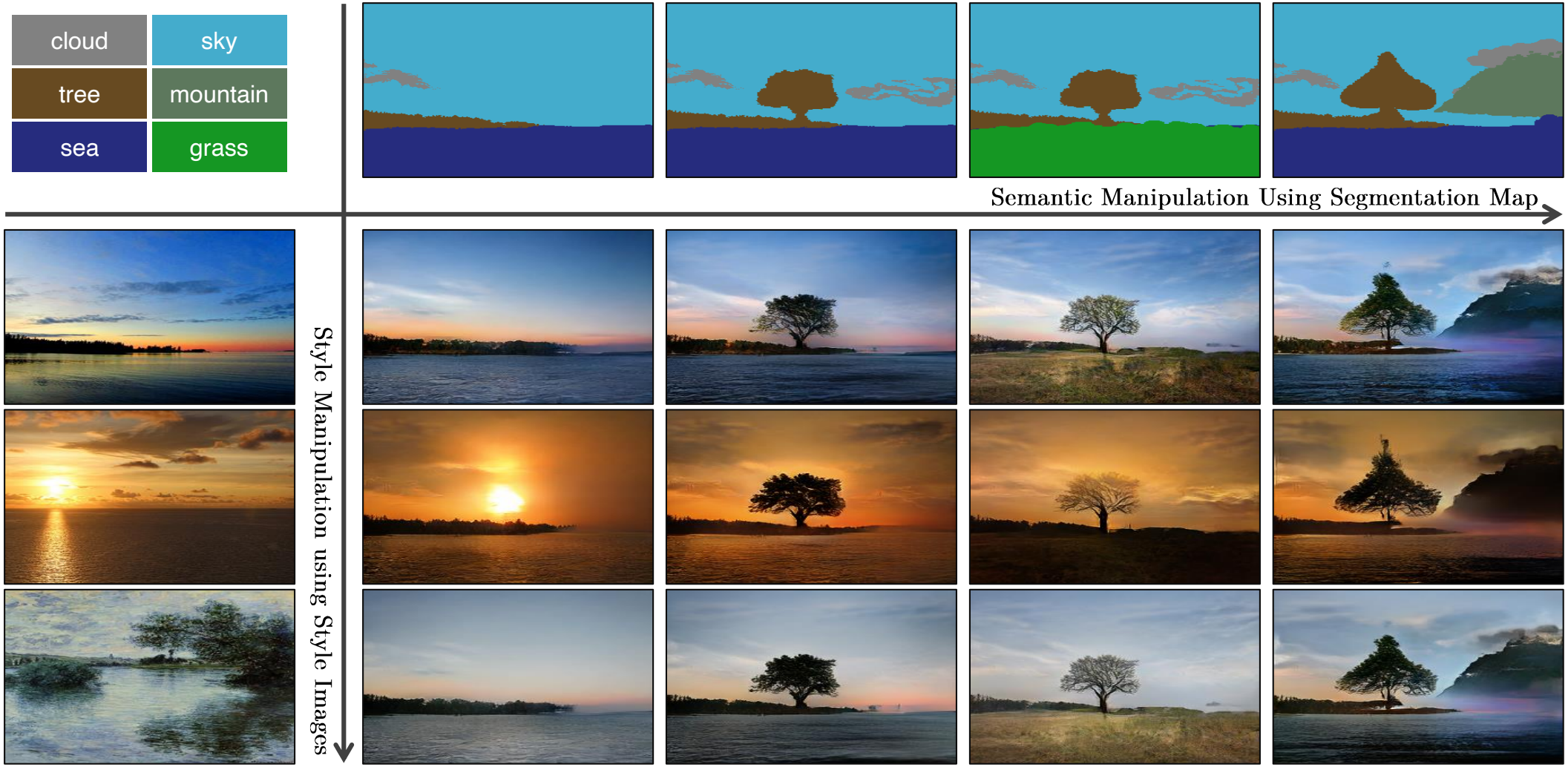
Slide Credit: Taesung Park and Jun-Yan Zhu

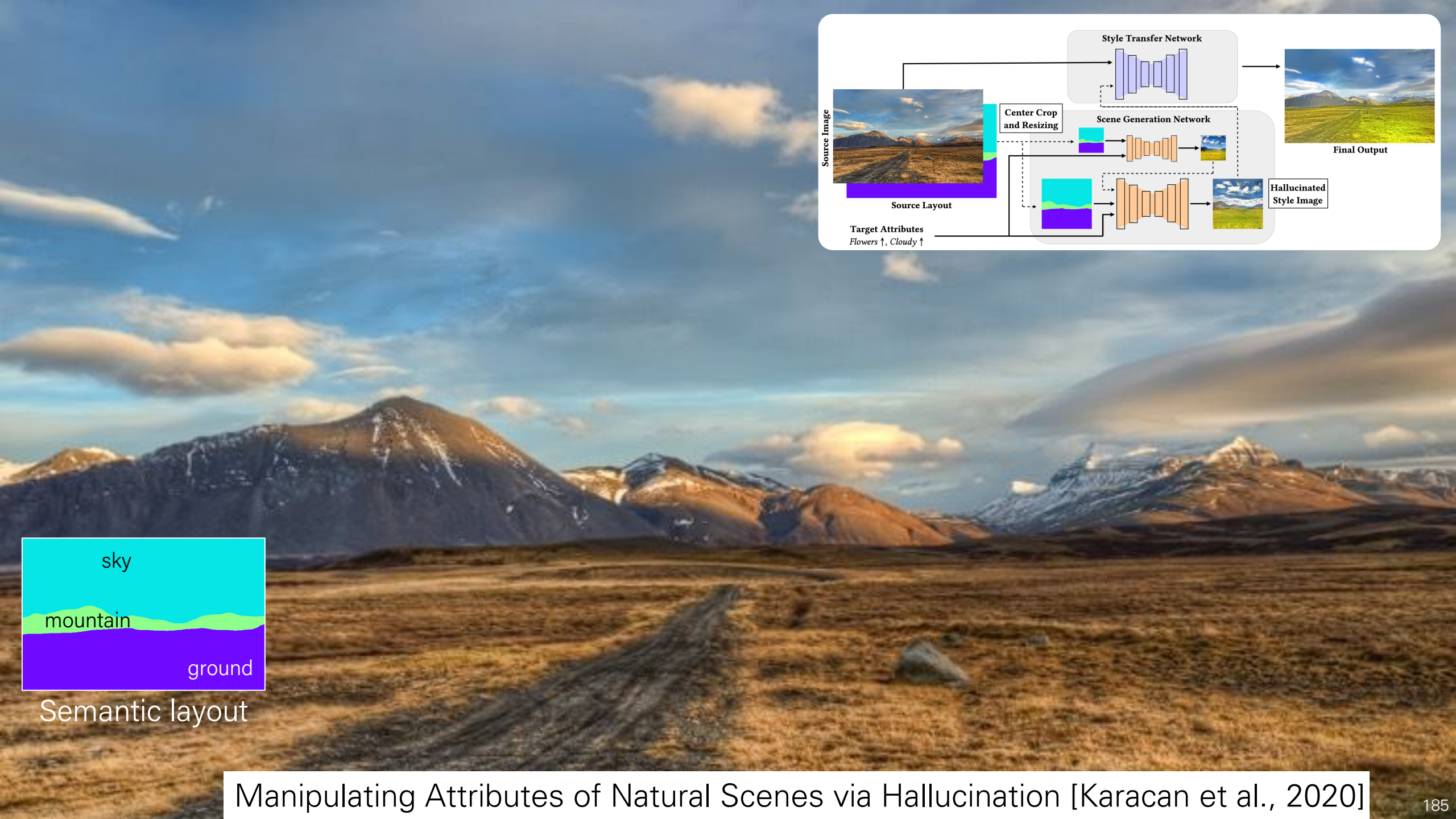
Monet's paintings → photos

Slide Credit: Taesung Park and Jun-Yan Zhu

Semantic Image Synthesis (SPADE) (Park et al., 2019)

- Image generation conditioned on semantic layouts





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

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

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

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

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

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

Spring and clouds

prediction

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

Moist, rain and fog

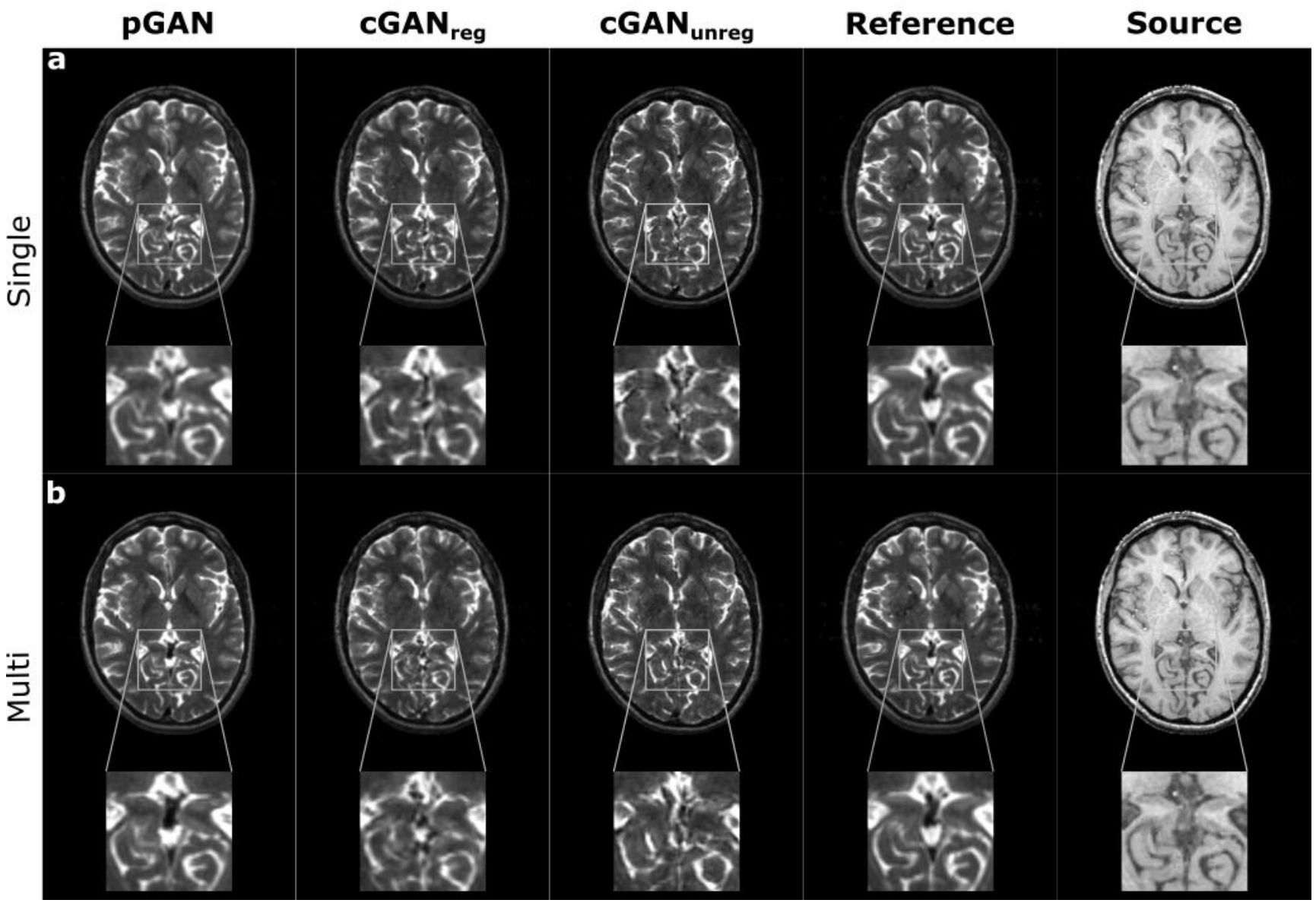
prediction

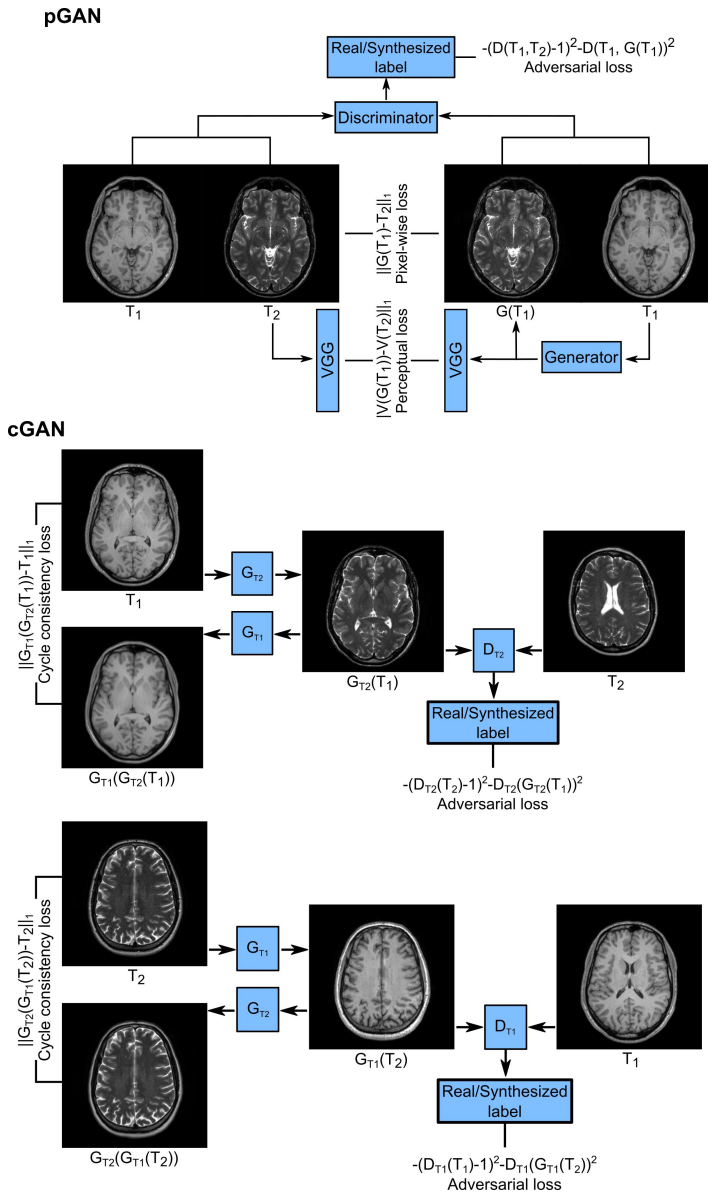
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

flowers

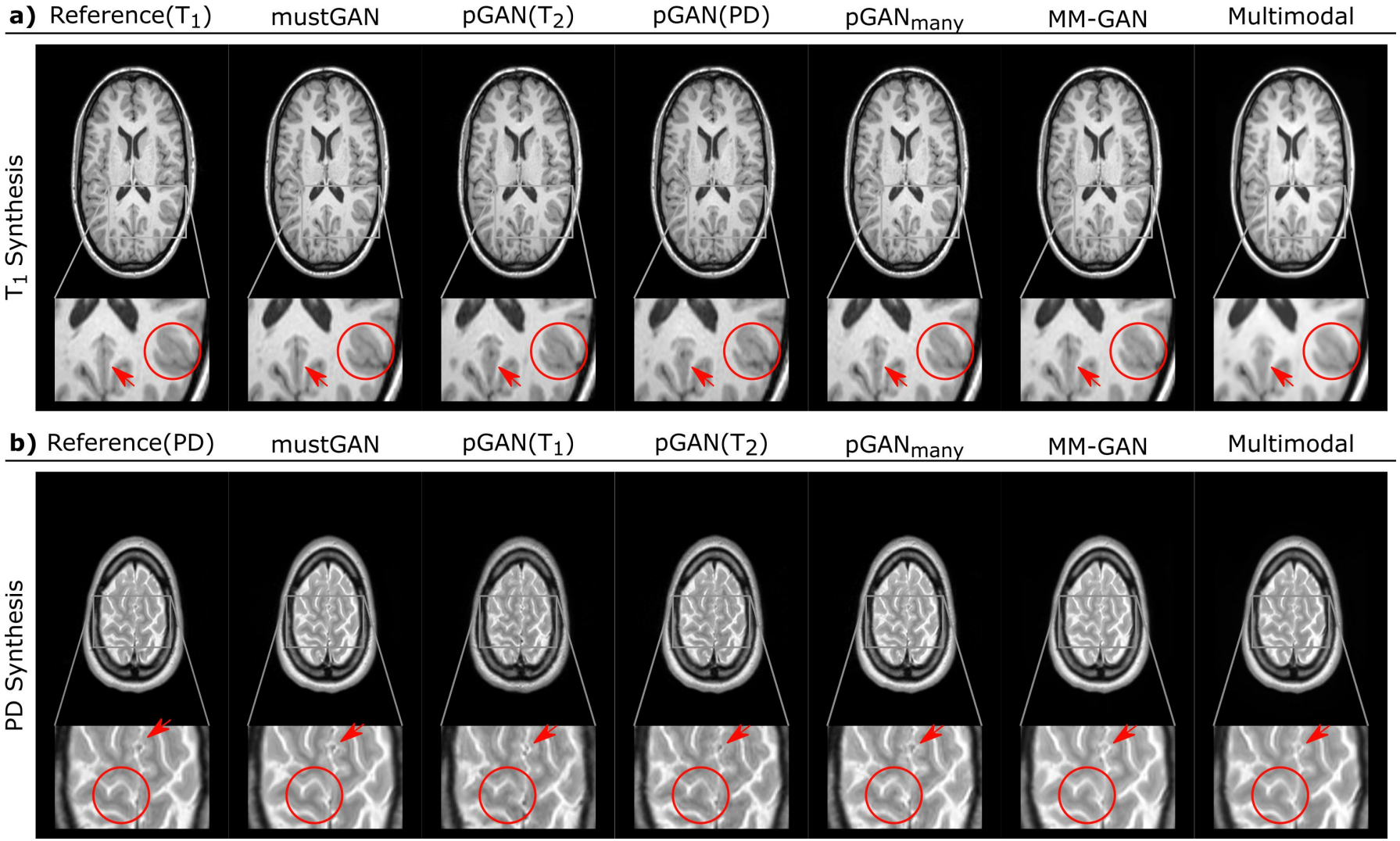
prediction

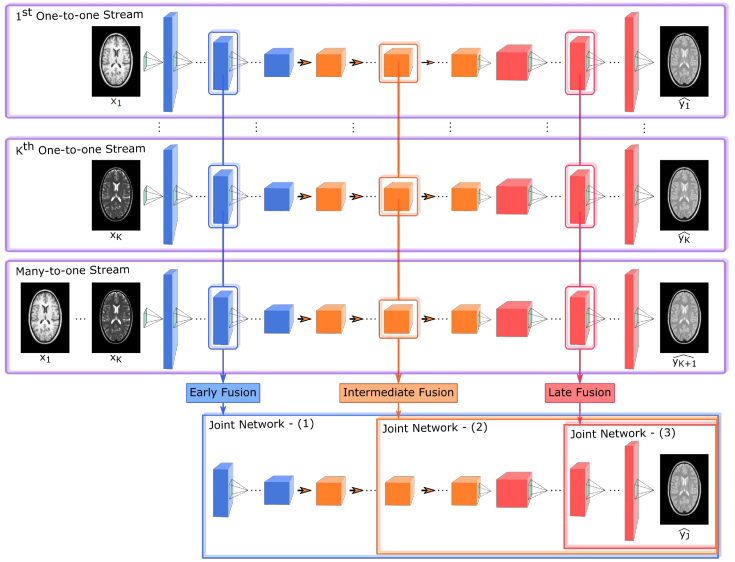
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]





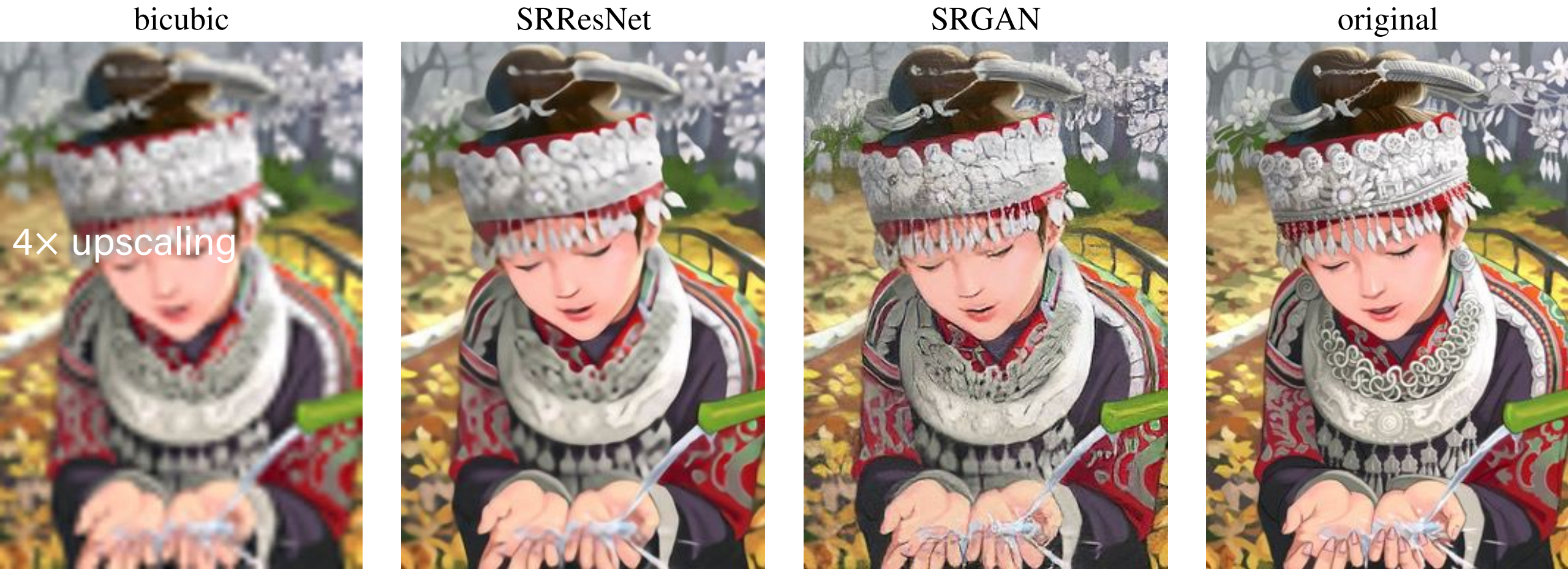
- Image Synthesis in Multi-Contrast MRI [Ul Hassan Dar et al. 2019]





- Image Synthesis in Multi-Contrast MRI [Mahmut Yurt et al. 2021]

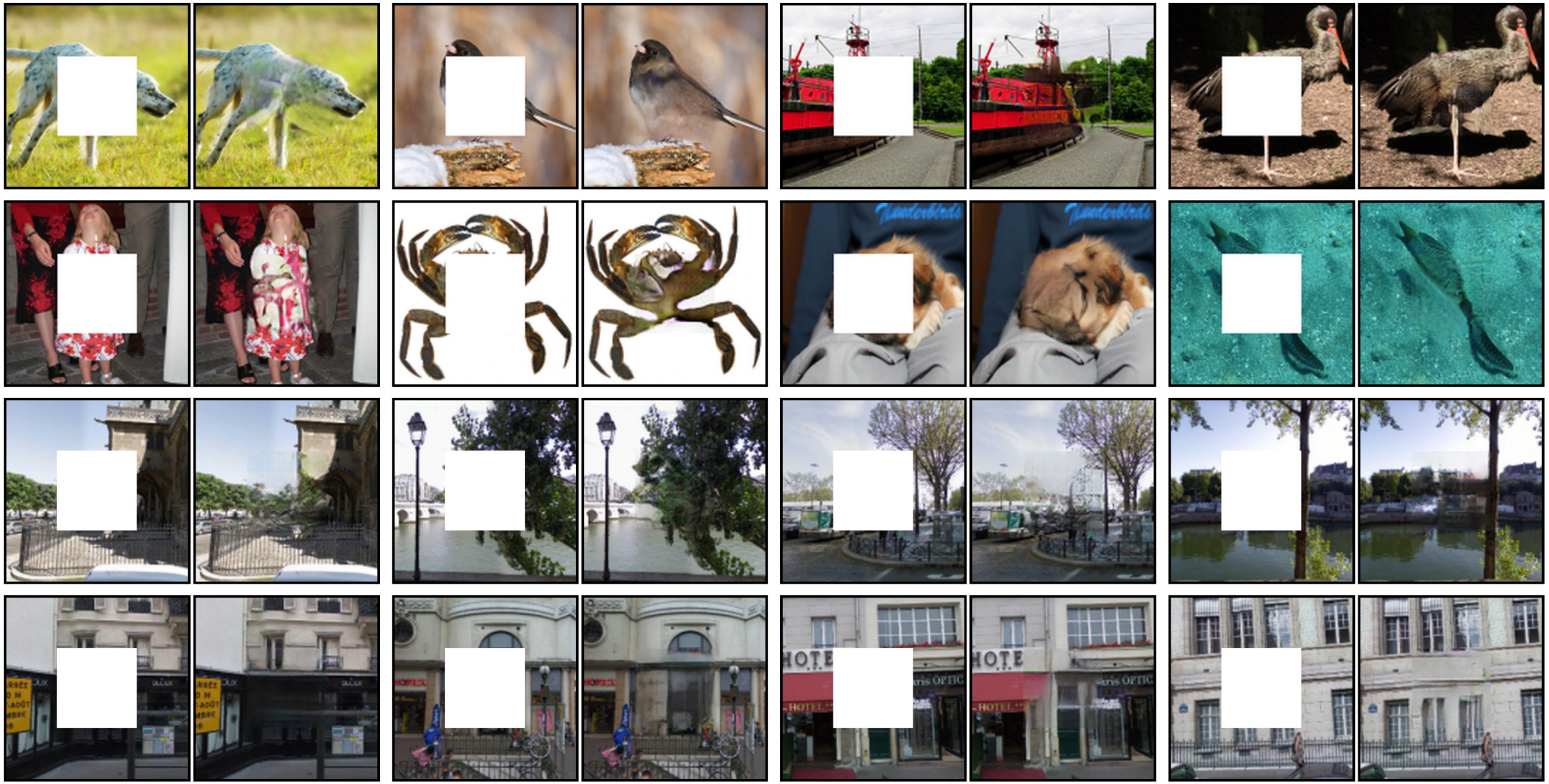
Single Image Super-Resolution



- Key idea: Combine content loss with adversarial loss

(Ledig et al., CVPR 2016)

Image Inpainting (Pathak et al., 2016)



- Key idea: Combine content loss with adversarial loss

(Ledig et al., CVPR 2016)

Image Deblurring

- non-uniform motion blur from a single blurry image.
- **Key idea:** Use multiscale CNNs to restore sharp images in an end-to-end manner

$$\mathcal{L}_{cont} = \frac{1}{2K} \sum_{k=1}^{K} \frac{1}{c_k w_k h_k} \|L_k - S_k\|^2$$

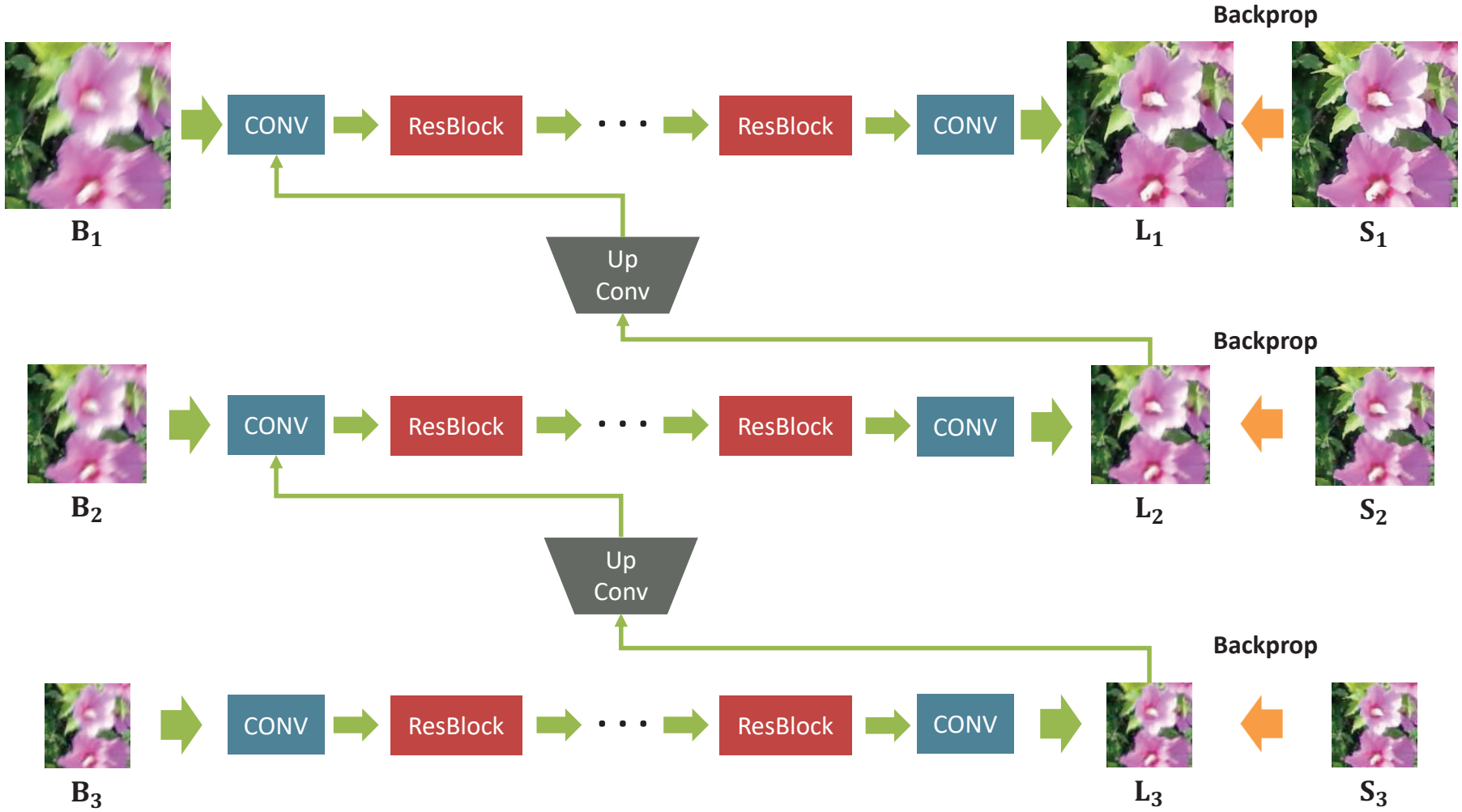
- can be interpreted as a kind of image to image translation
- An additional adversarial loss

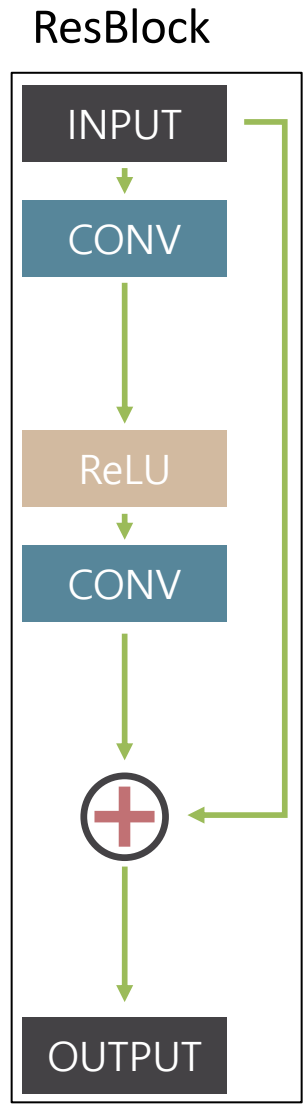
$$\mathcal{L}_{adv} = \mathop{\mathbb{E}}_{S \sim p_{sharp}(S)} [\log D(S)] + \mathop{\mathbb{E}}_{B \sim p_{blurry}(B)} [\log(1 - D(G(B)))]$$

(Nah et al., CVPR 2017)

198

Image Deblurring

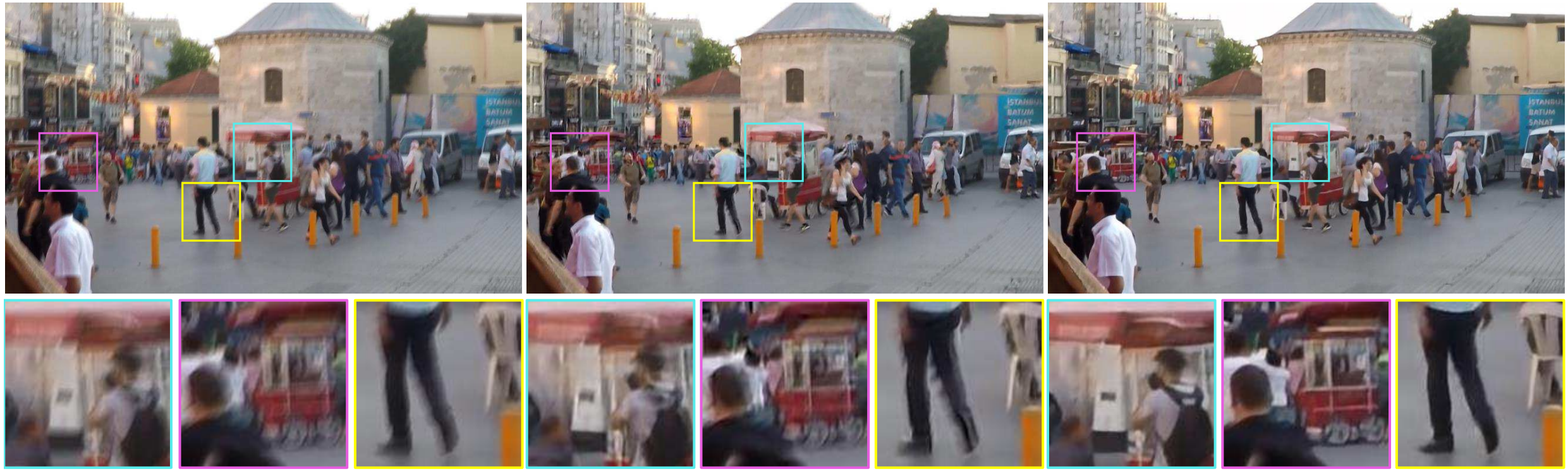




- Coarser scale features aid finer scale image deblurring

(Nah et al., CVPR 2017)

Image Deblurring



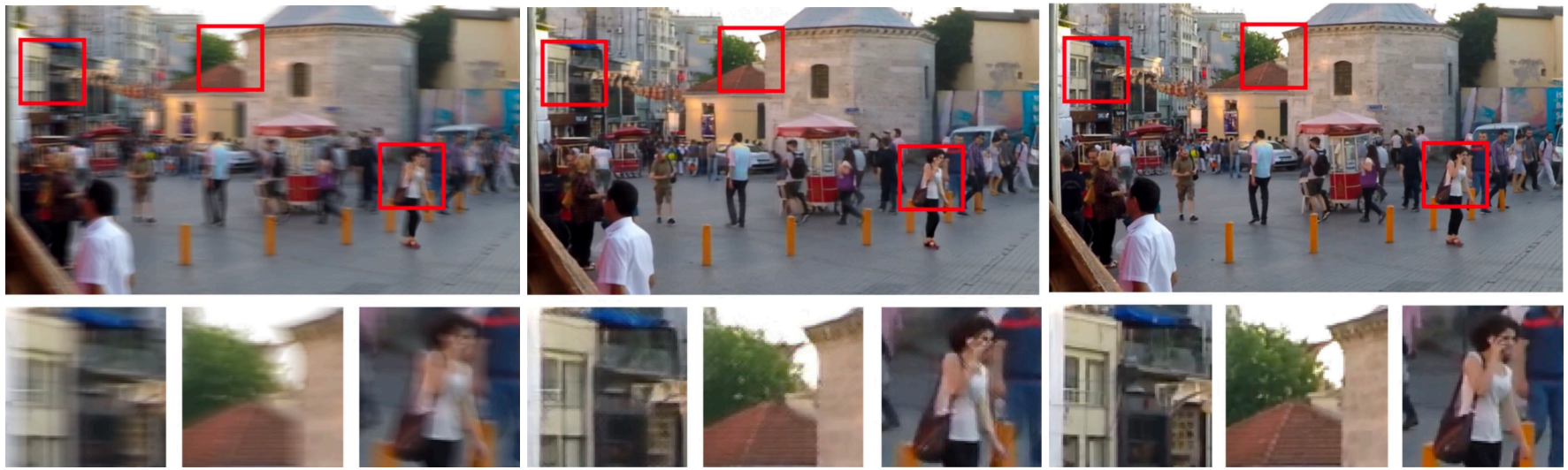
Blurred images | Sun et al., CVPR 2015 | Nah et al., CVPR 2017

(Nah et al., CVPR 2017)

Image Deblurring

- non-uniform motion blur from a single blurry image.
- **Key idea:** Use a conditional GAN and content loss

$$\mathcal{L} = \underbrace{\underbrace{\mathcal{L}_{GAN}}_{adv\ loss} + \underbrace{\lambda \cdot \mathcal{L}_X}_{content\ loss}}_{total\ loss} \qquad \mathcal{L}_{GAN} = \sum_{n=1}^{N} -D_{\theta_D}(G_{\theta_G}(I^B)) \qquad \mathcal{L}_X = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^S)_{x,y} - \phi_{i,j}(G_{\theta_G}(I^B))_{x,y})^2$$

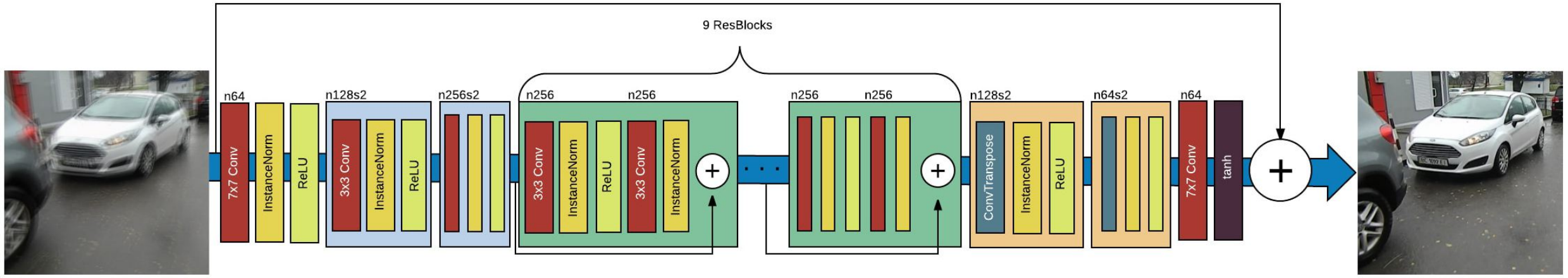


Blurred images Groundtruth Predicted

(Kupyn et al., CVPR 2018)

Image Deblurring

- Key idea: An image to image translation model that learns the residual to sharpen the blurred image



202

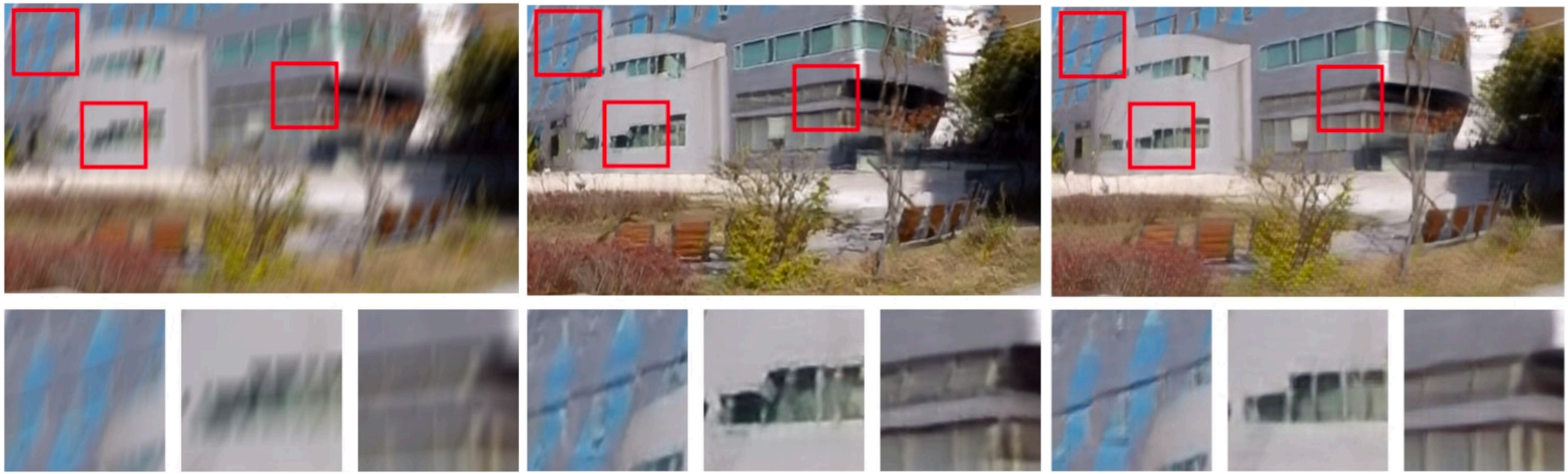
(Kupyn et al., CVPR 2018)

Image Deblurring

Blurred images | Nah et al., CVPR 2017 | Kupyn et al., CVPR 2018

(Kupyn et al., CVPR 2018)

Image Deblurring



Blurred images | Nah et al., CVPR 2017 | Kupyn et al., CVPR 2018

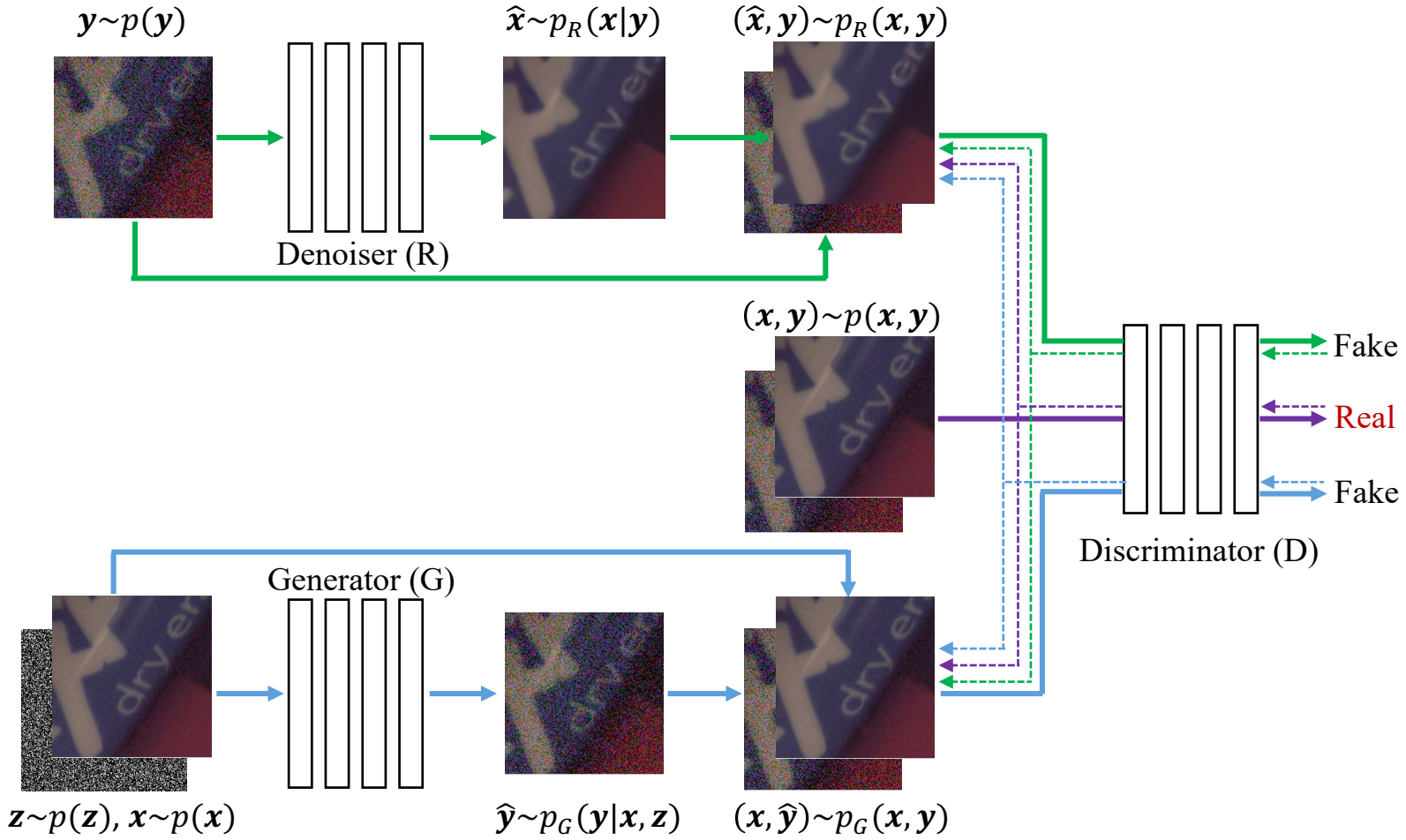
(Kupyn et al., CVPR 2018)

Image Deblurring

Blurred images | Nah et al., CVPR 2017 | Kupyn et al., CVPR 2018

(Kupyn et al., CVPR 2018)

Image Denoising

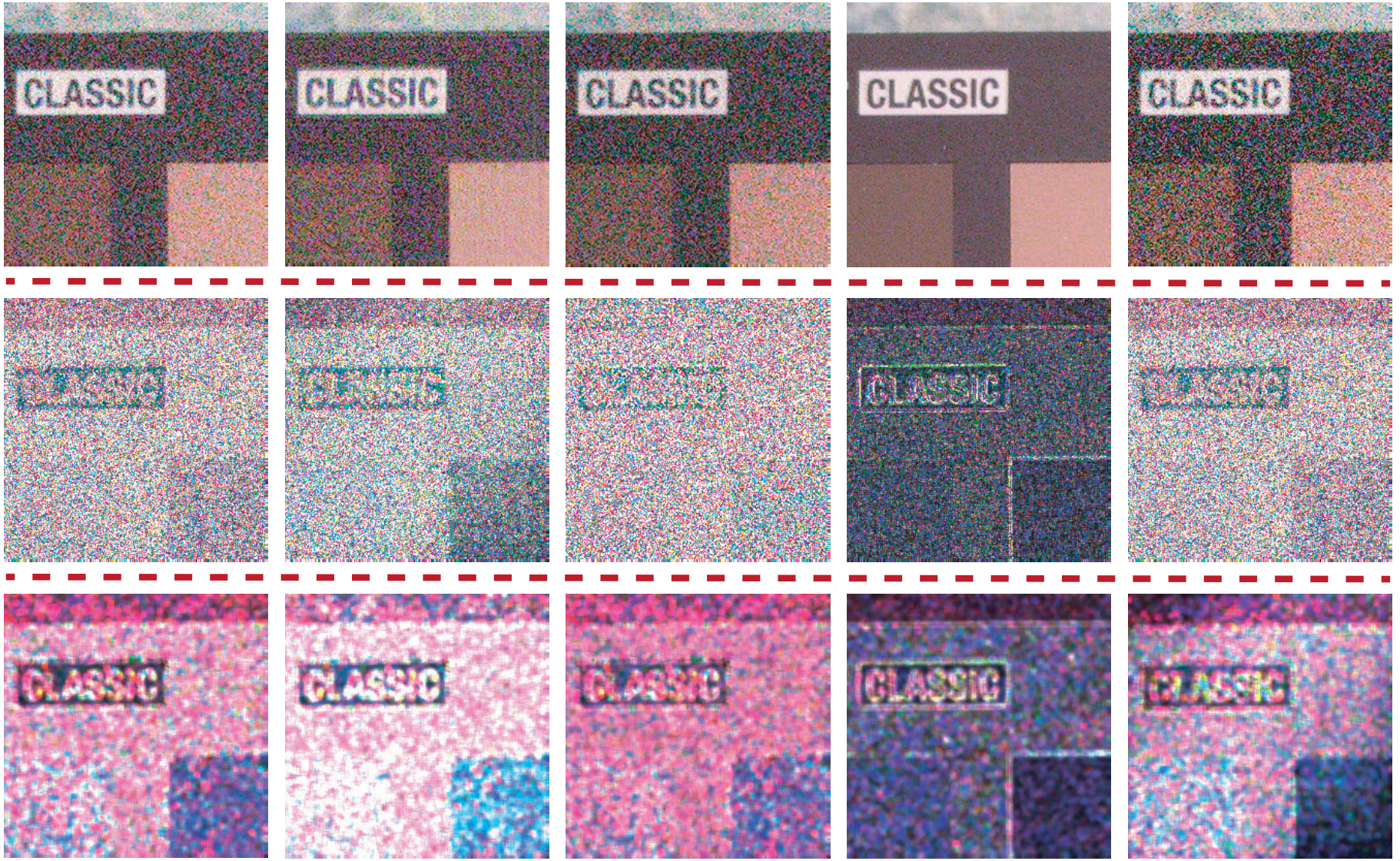


- Key idea: simultaneously deal with the noise removal and noise generation tasks.

(Yue et al., ECCV 2020)

Image Denoising

Generated Noisy Images

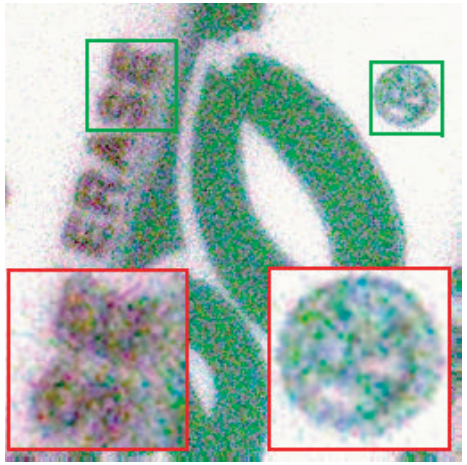


(a) Real (b) CBDNet (c) ULRD (d) GRDN (e) DANet

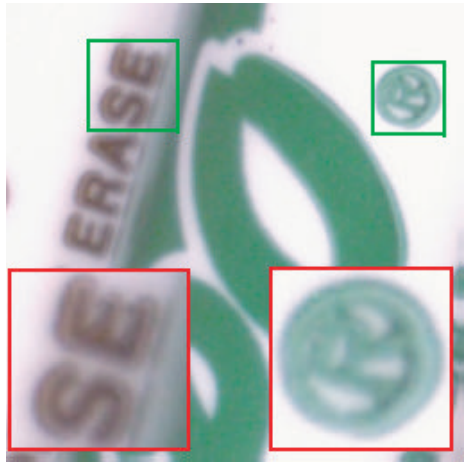
(Yue et al., ECCV 2020)

Image Denoising

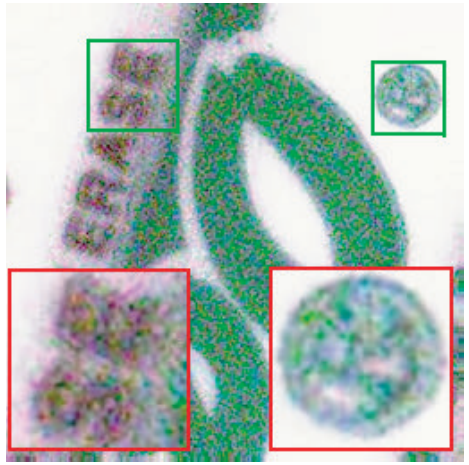
Denoising results



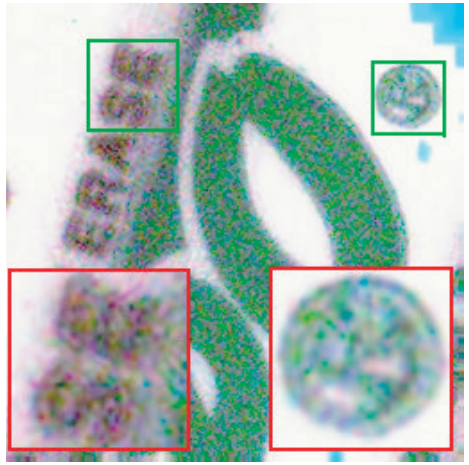
(a): Noisy



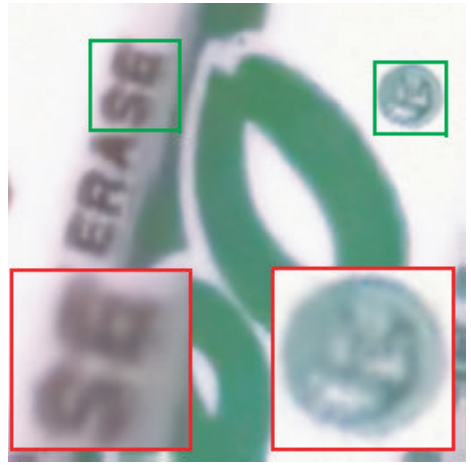
(b): GroundTruth



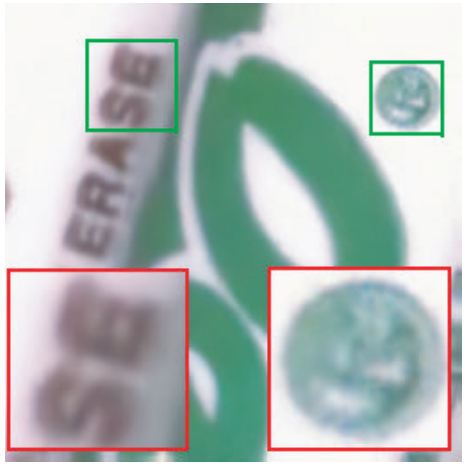
(c): BM3D



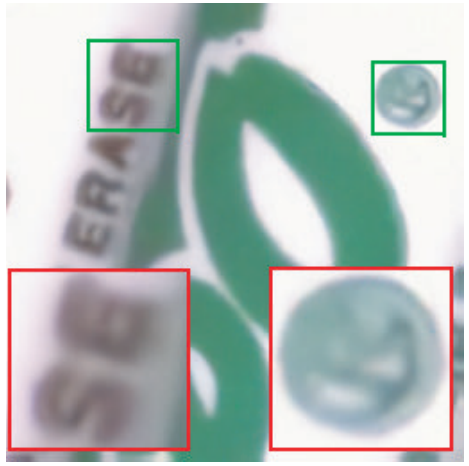
(d): WNNM



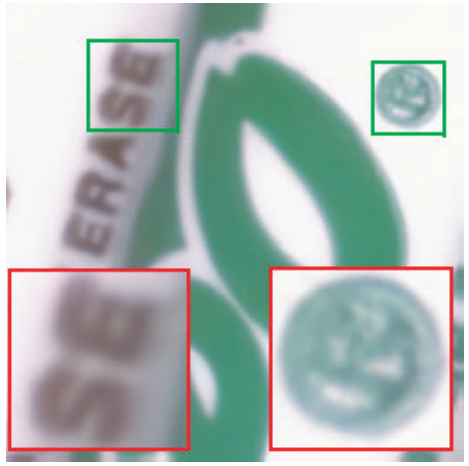
(e): DnCNN



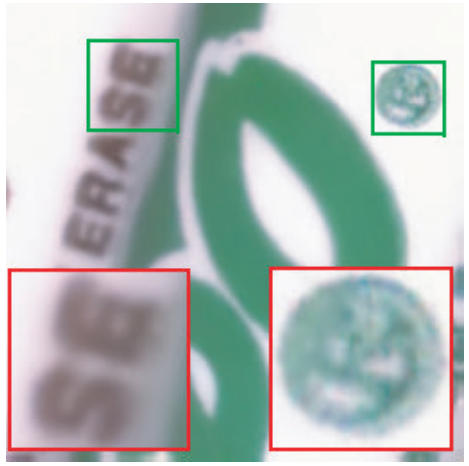
(f): CBDNet



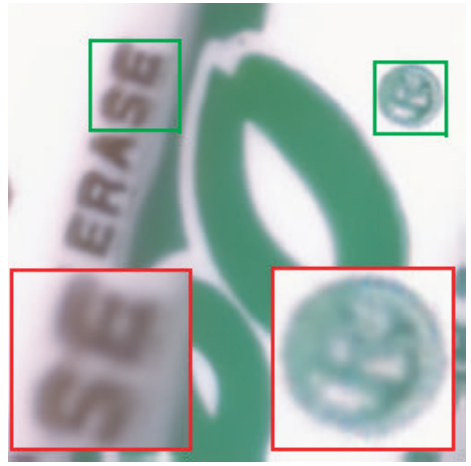
(g): RIDNet



(h): VDN



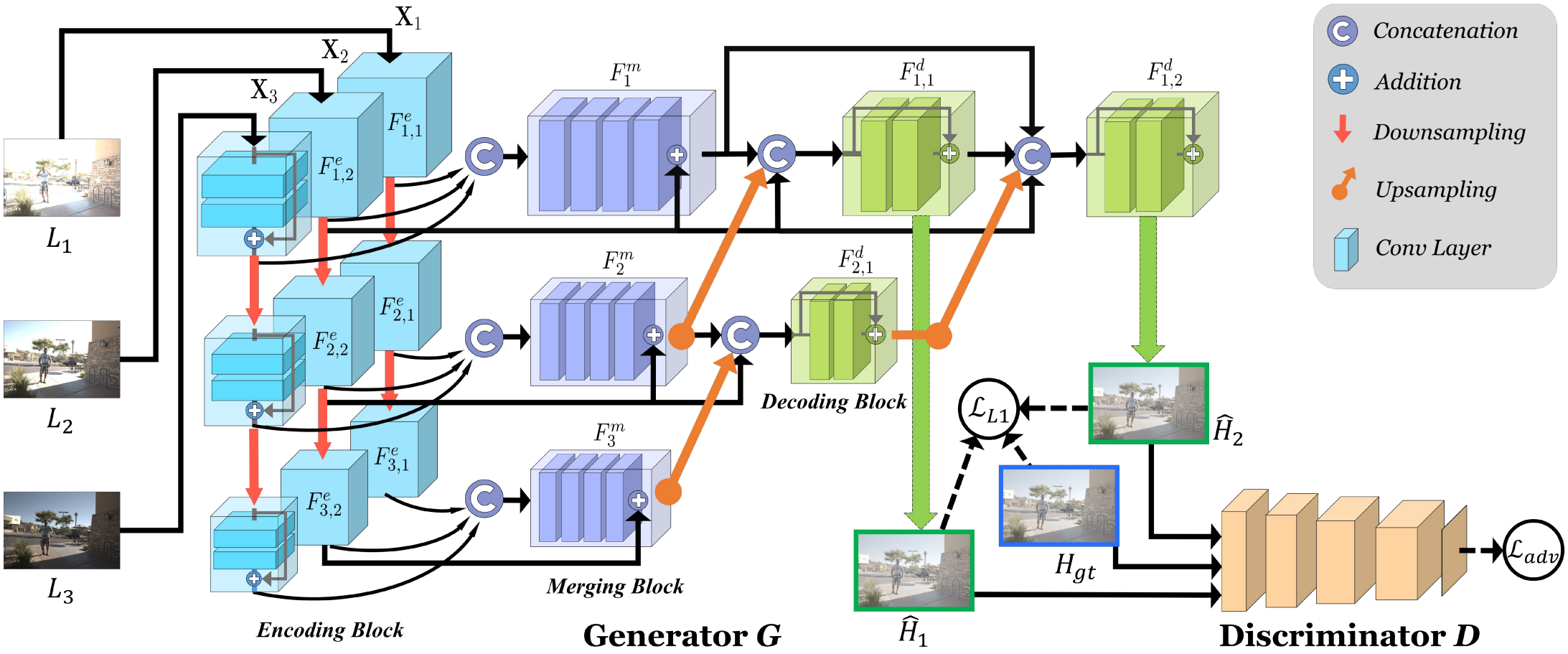
(i): DANet



(j): DANet+

(Yue et al., ECCV 2020)

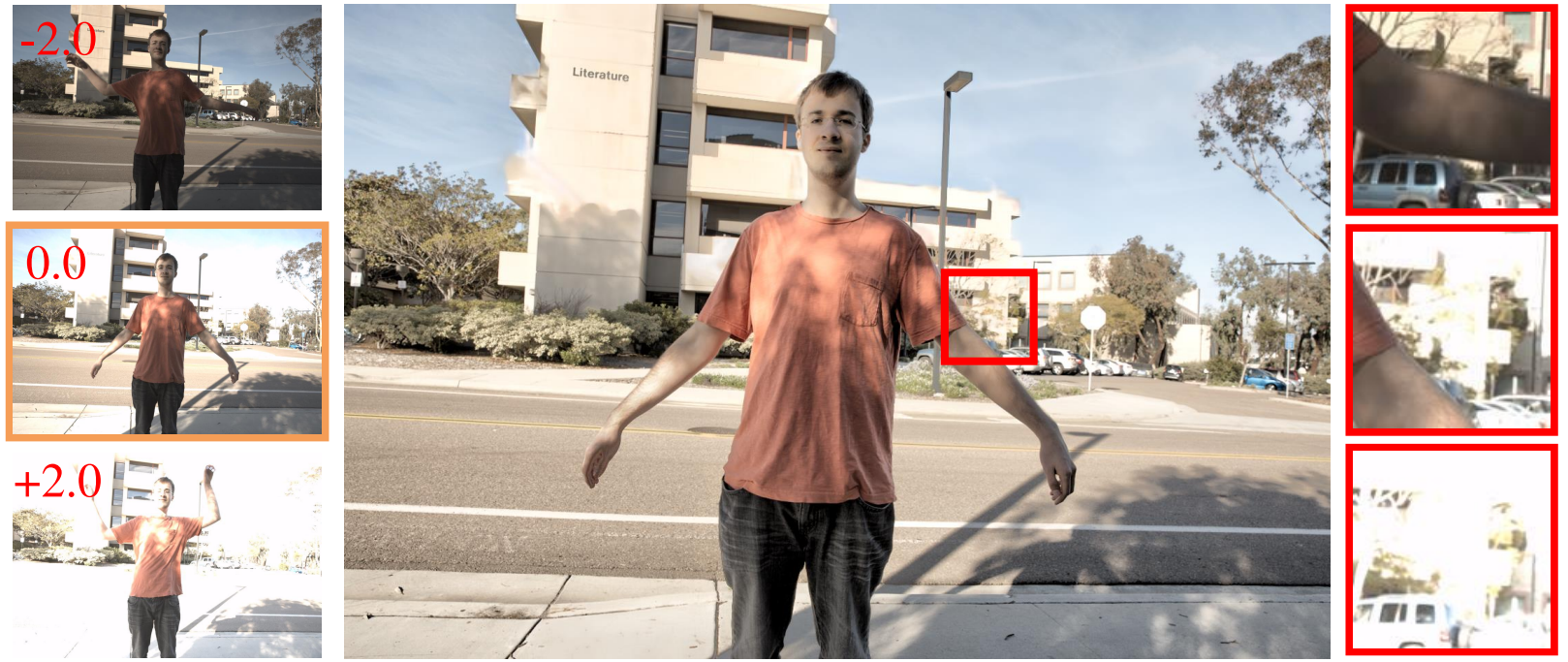
High Dynamic Range Imaging



- Key idea: Incorporate adversarial learning to be able to produce faithful information in the regions with missing content.

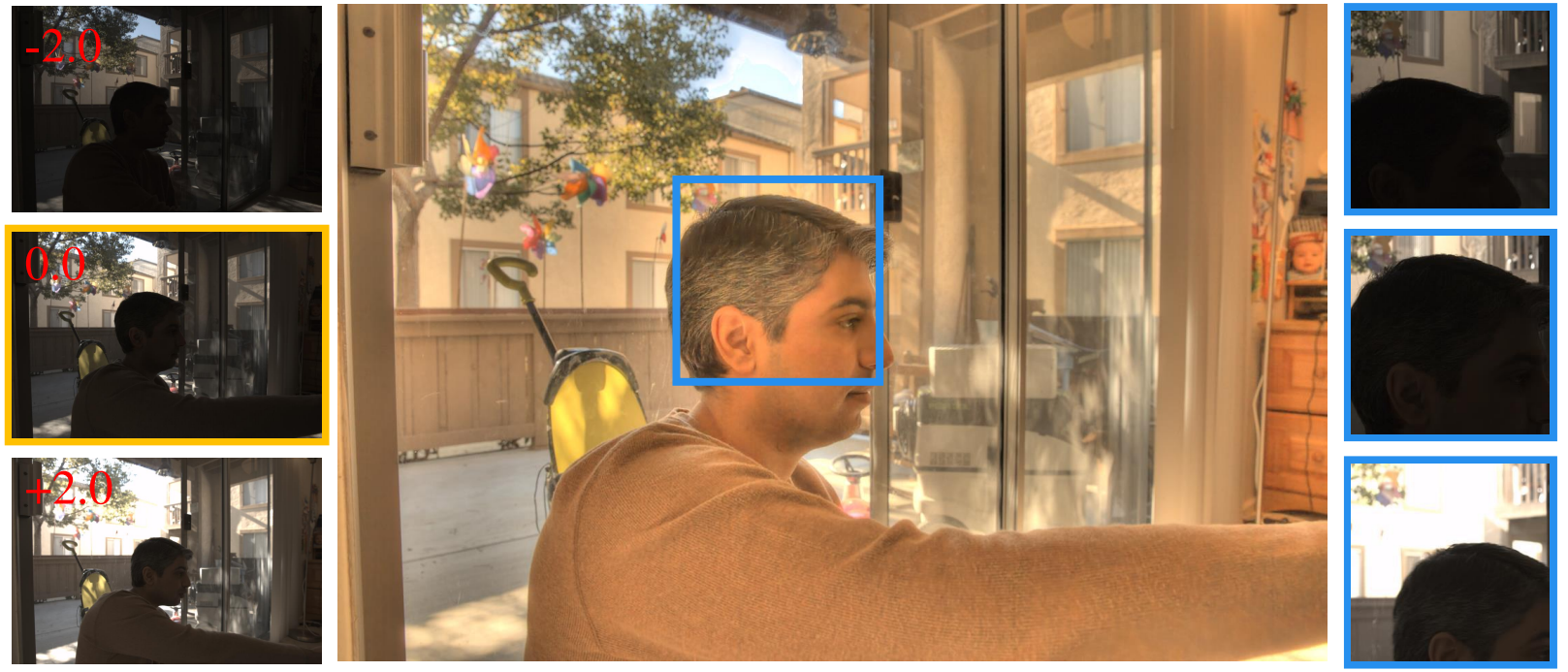
(Niu et al., IEEE TIP 2021)

High Dynamic Range Imaging



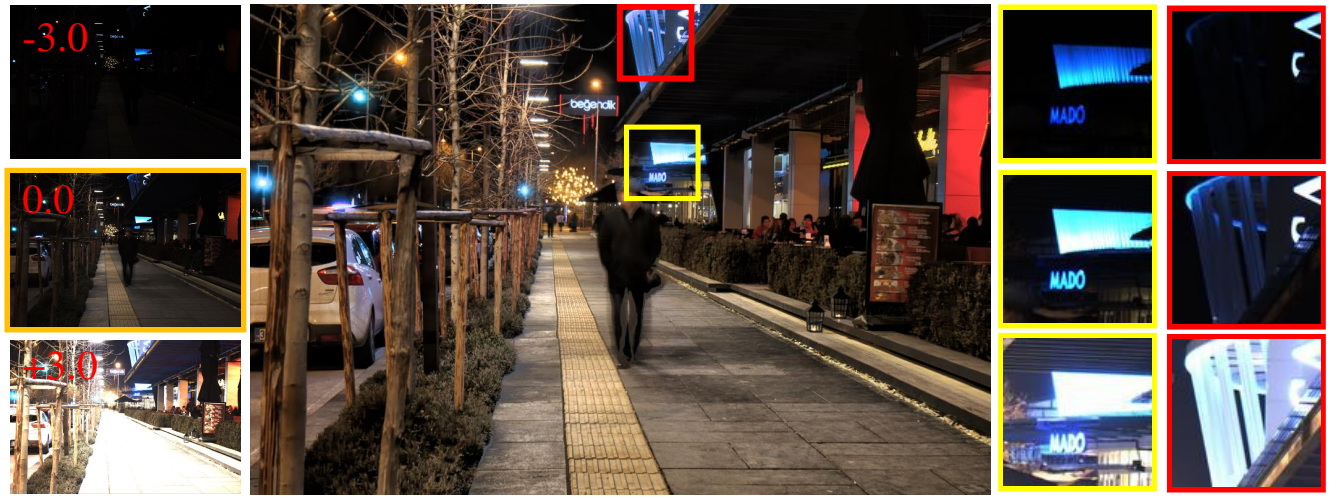
(Niu et al., IEEE TIP 2021)

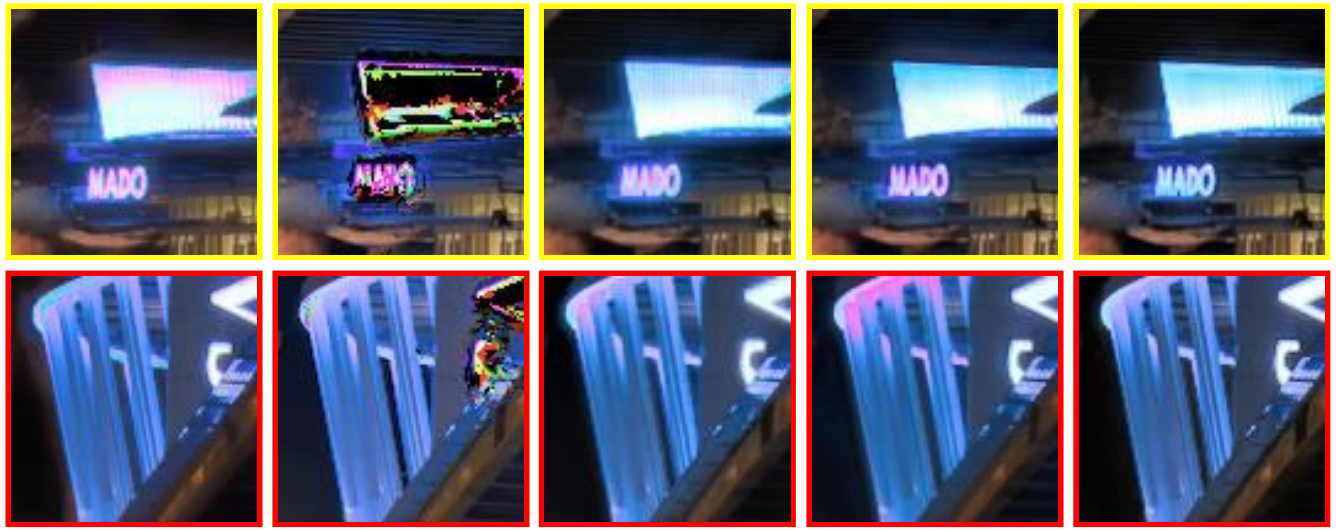
High Dynamic Range Imaging



(Niu et al., IEEE TIP 2021)

High Dynamic Range Imaging



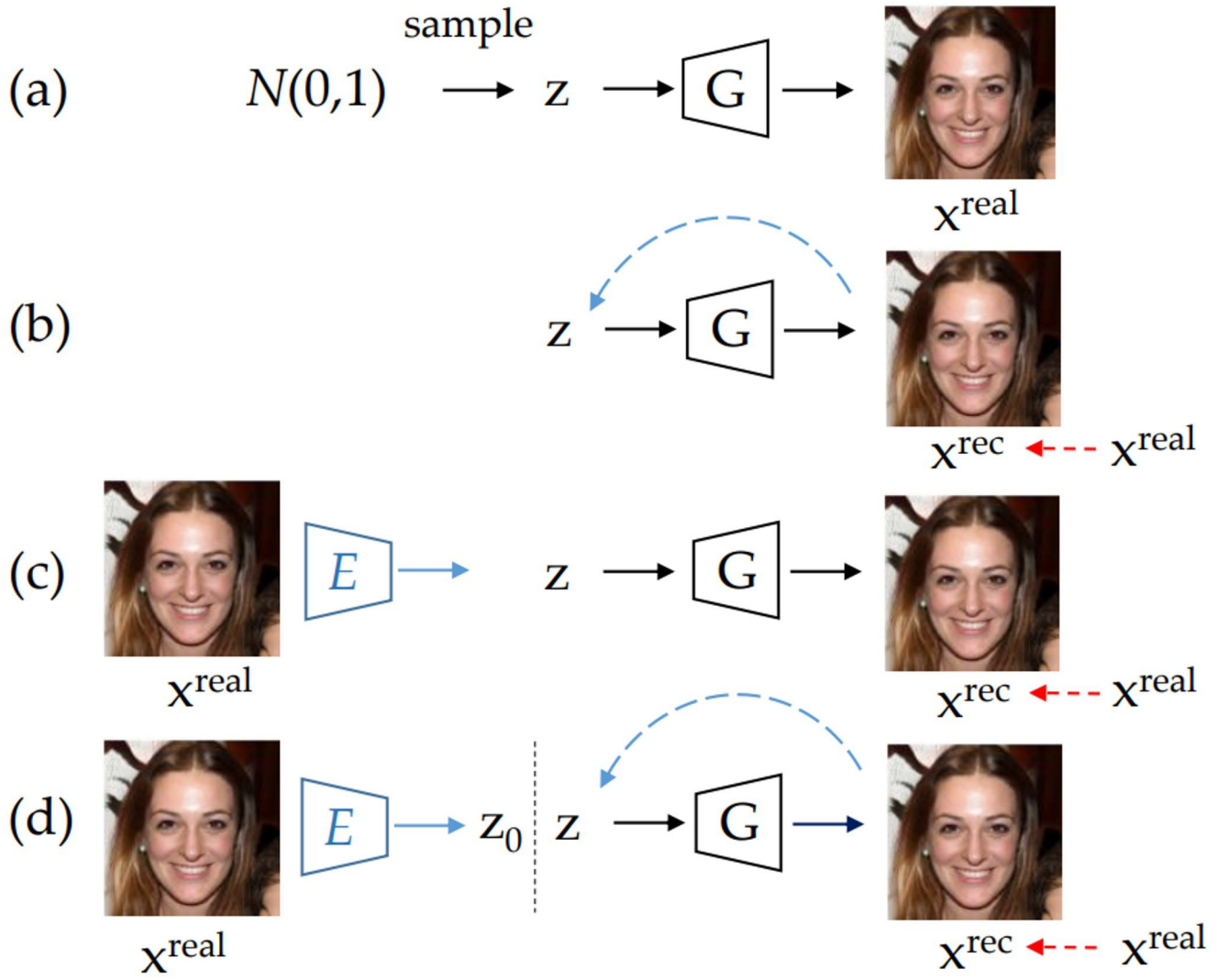


(Niu et al., IEEE TIP 2021)

GAN Inversion

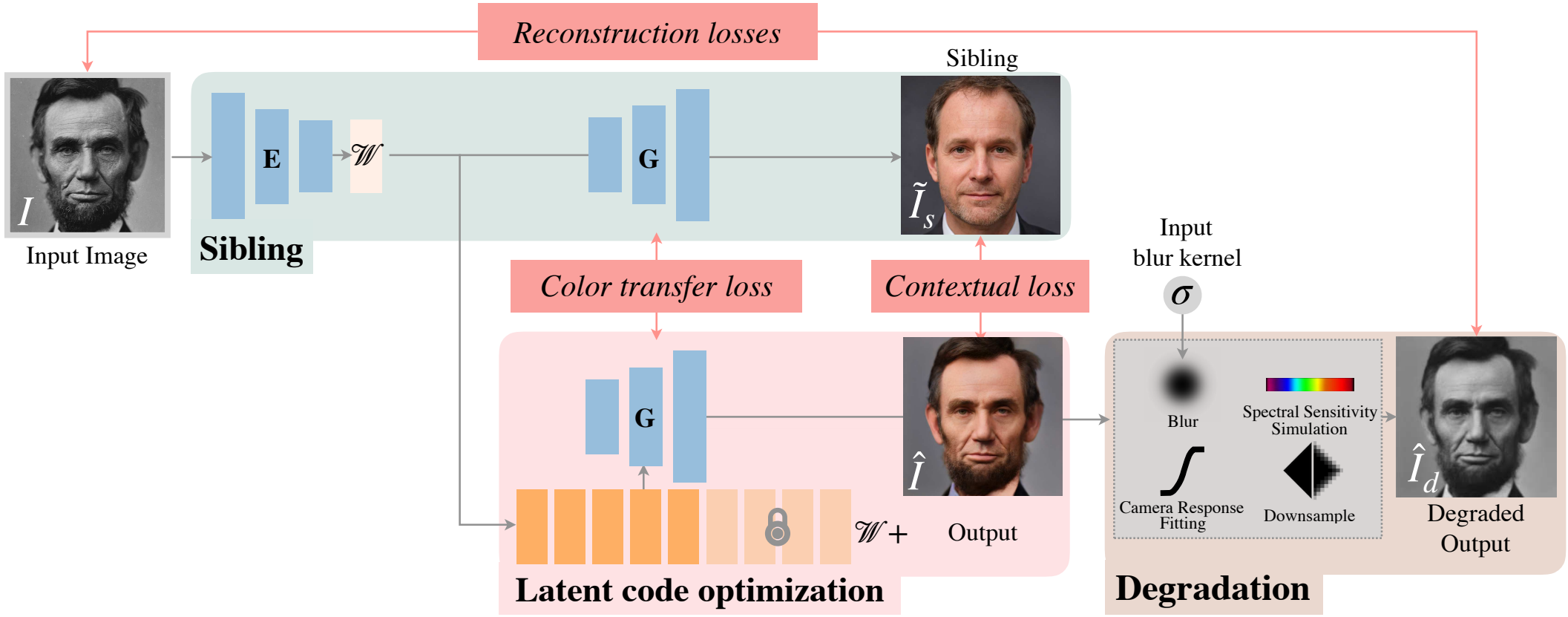
3 methods of inversion:

- Optimization-based (b)
- Learning-based (c)
- Hybrid (d)



(Xia et al., arXiv 2021)

Time-travel Rephotography



- Key idea: Use the StyleGAN2 framework to project old photos into the space of modern high-resolution photos for enhancing their quality.

(Luo et al., SIGGRAPH Asia 2021)

Time-travel Rephotography

(Luo et al., SIGGRAPH Asia 2021)

Time-travel Rephotography

(Luo et al., SIGGRAPH Asia 2021)

Next Lecture:
Visual Quality Assessment