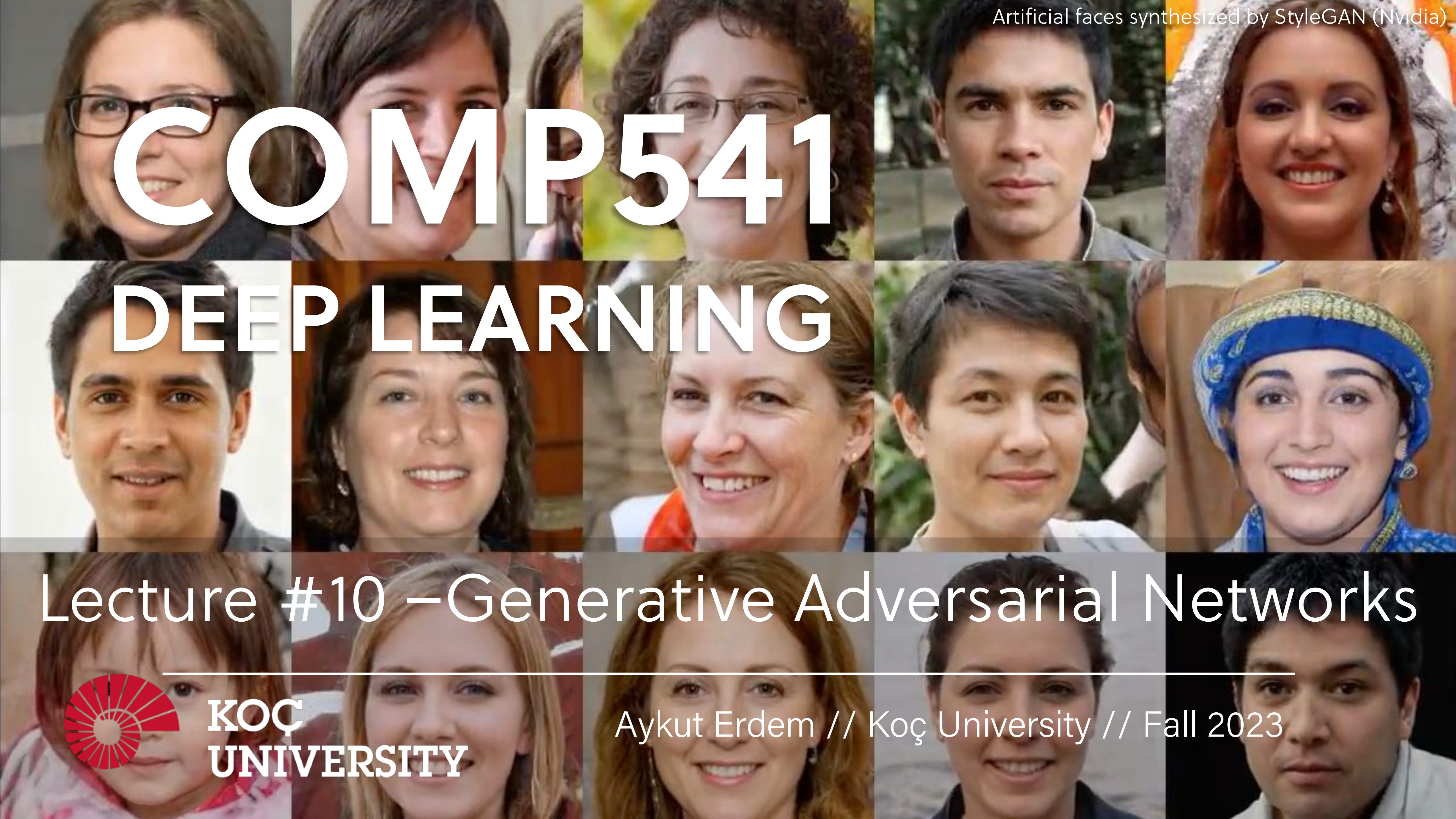


Artificial faces synthesized by StyleGAN (Nvidia)

COMP541
DEEP LEARNING

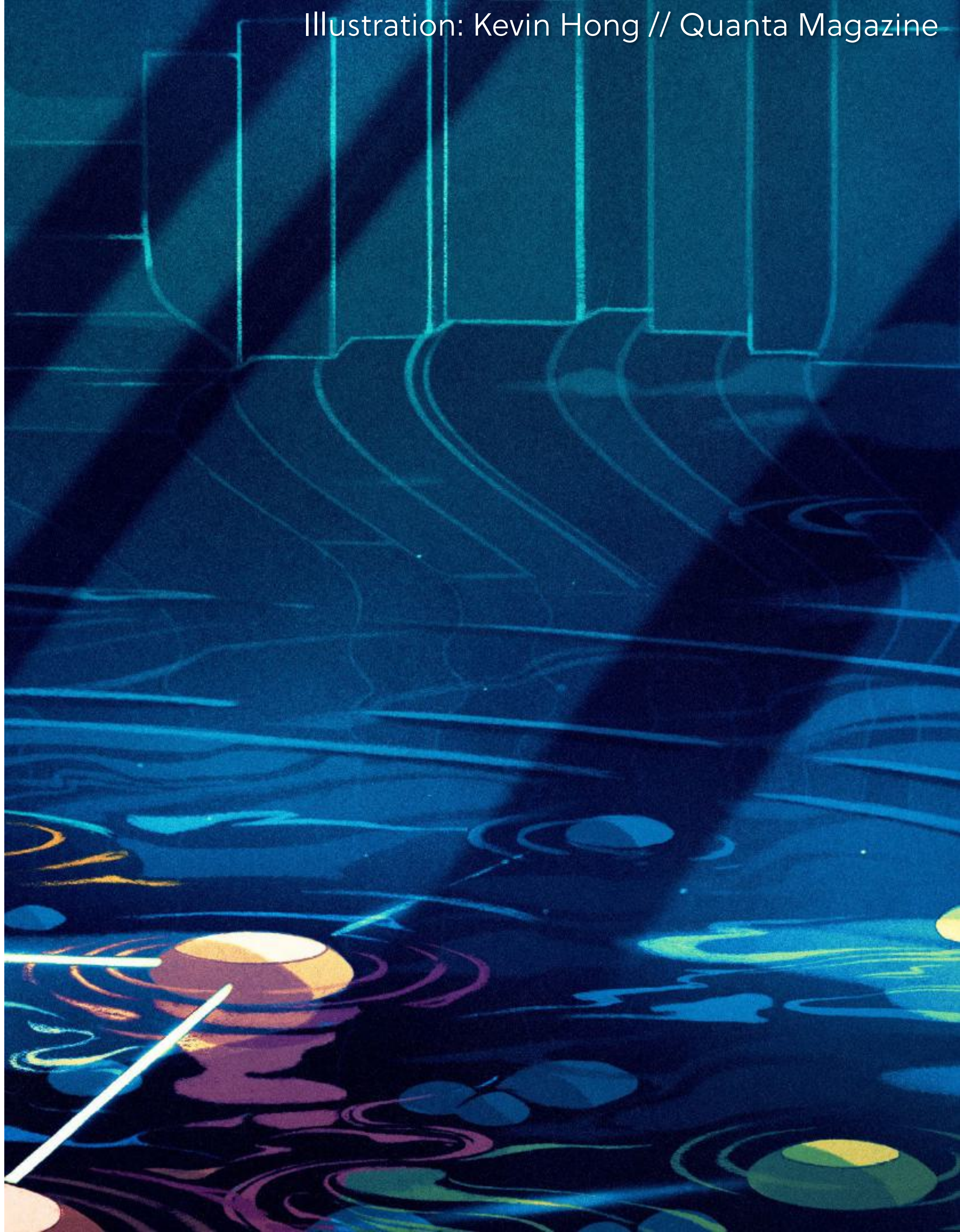
Lecture #10 – Generative Adversarial Networks

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Fall 2023

Previously on COMP541

- graph structured data
- graph neural nets (GNNs)
- GNNs for "classical" network problems



Lecture overview

- supervised vs unsupervised learning
- generative modeling
- basic foundations
  - sparse coding
  - autoencoders
- generative adversarial networks (GANs)

Disclaimer: Some of the material and slides for this lecture were borrowed from

—Justin Johnson's EECS 498/598 class

—Ruslan Salakhutdinov's talk titled "Unsupervised Learning: Learning Deep Generative Models"

—Ian Goodfellow's tutorial on "Generative Adversarial Networks"

—Aaron Courville's IFT6135 class

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Classification

Cat

Supervised vs Unsupervised Learning

Supervised Learning

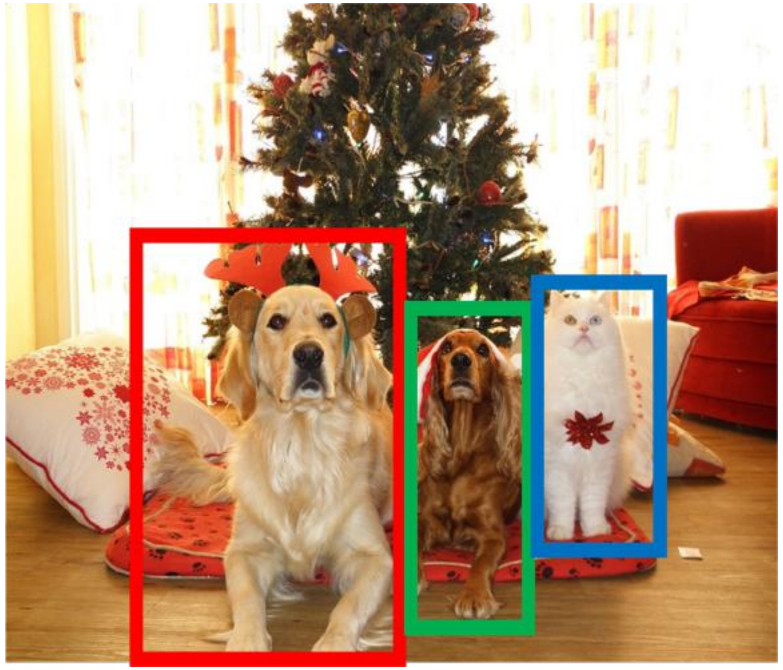
Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Object Detection



DOG, DOG, CAT

Supervised vs Unsupervised Learning

Supervised Learning

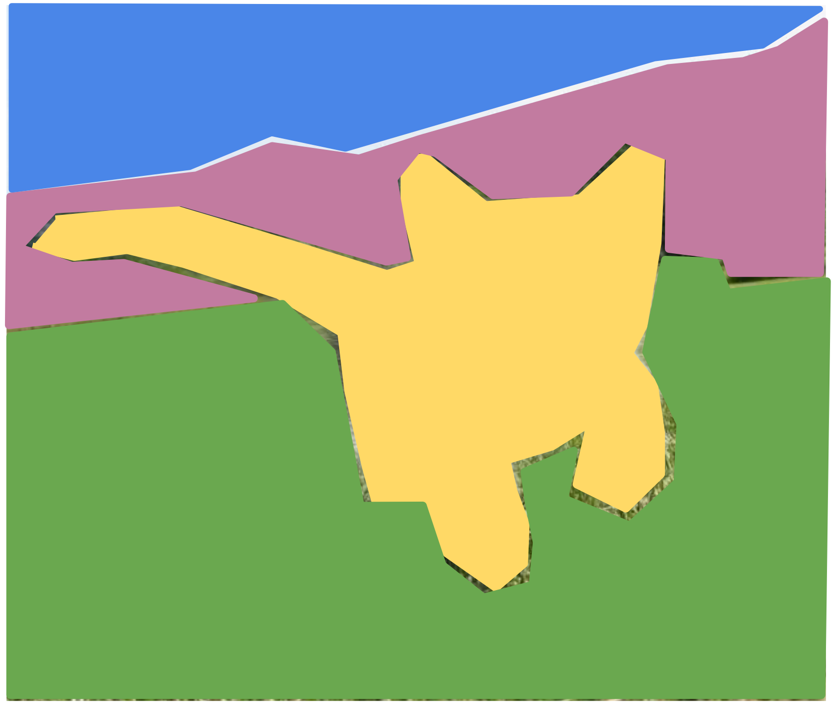
Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Semantic Segmentation



GRASS, CAT, TREE, SKY

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Image captioning

A cat sitting on a suitcase on the floor

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Sentiment Analysis

```
“This Movie is amazing.
It has a great plot and
talented actors, and
the supporting cast is
really good as well.“
```

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Unsupervised Learning

Data: x

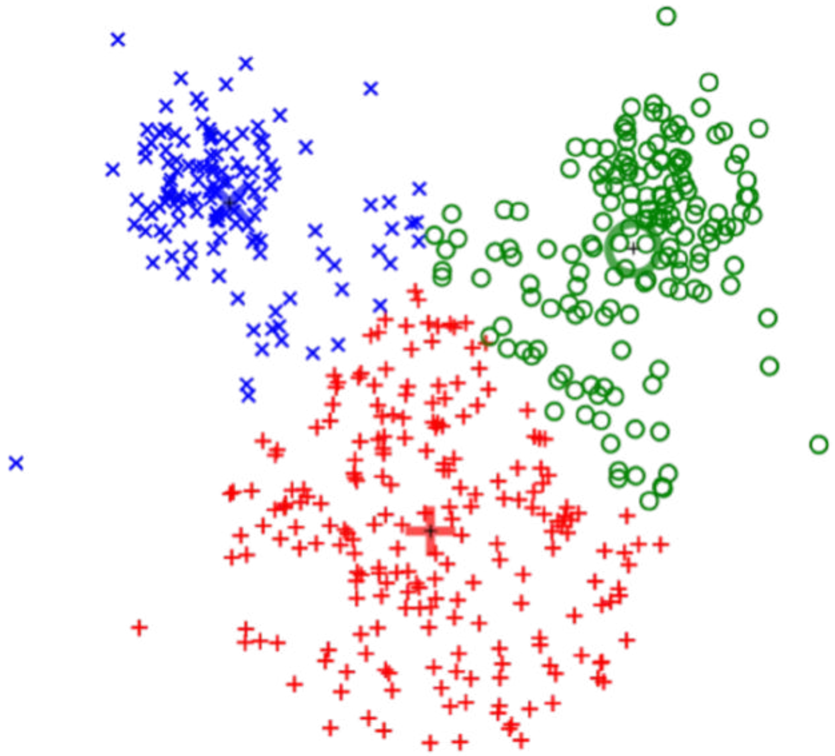
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Clustering
(e.g. K-Means)



Unsupervised Learning

Data: x

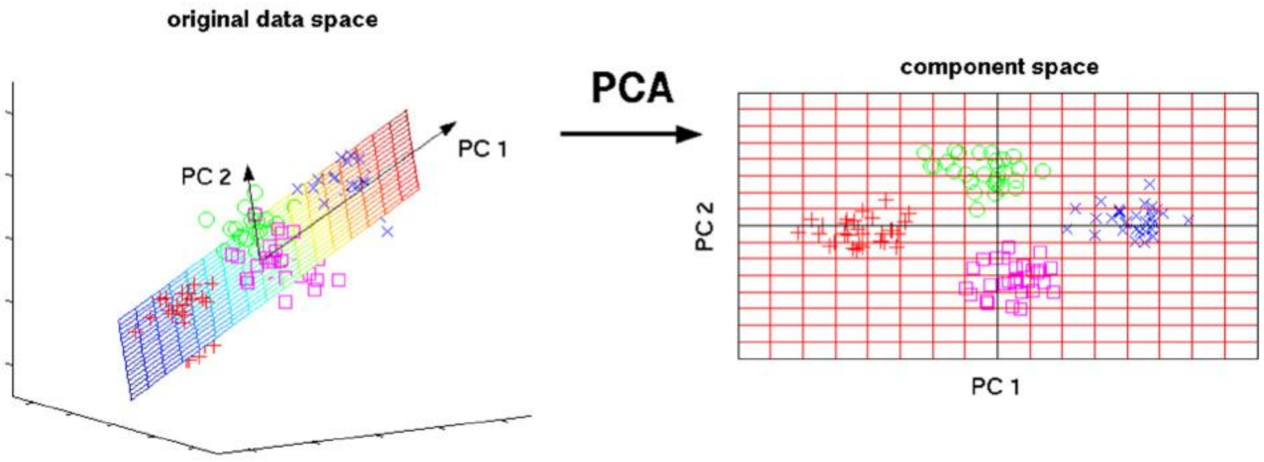
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Dimensionality Reduction
(e.g. Principal Components Analysis)



3D → 2D

Unsupervised Learning

Data: x

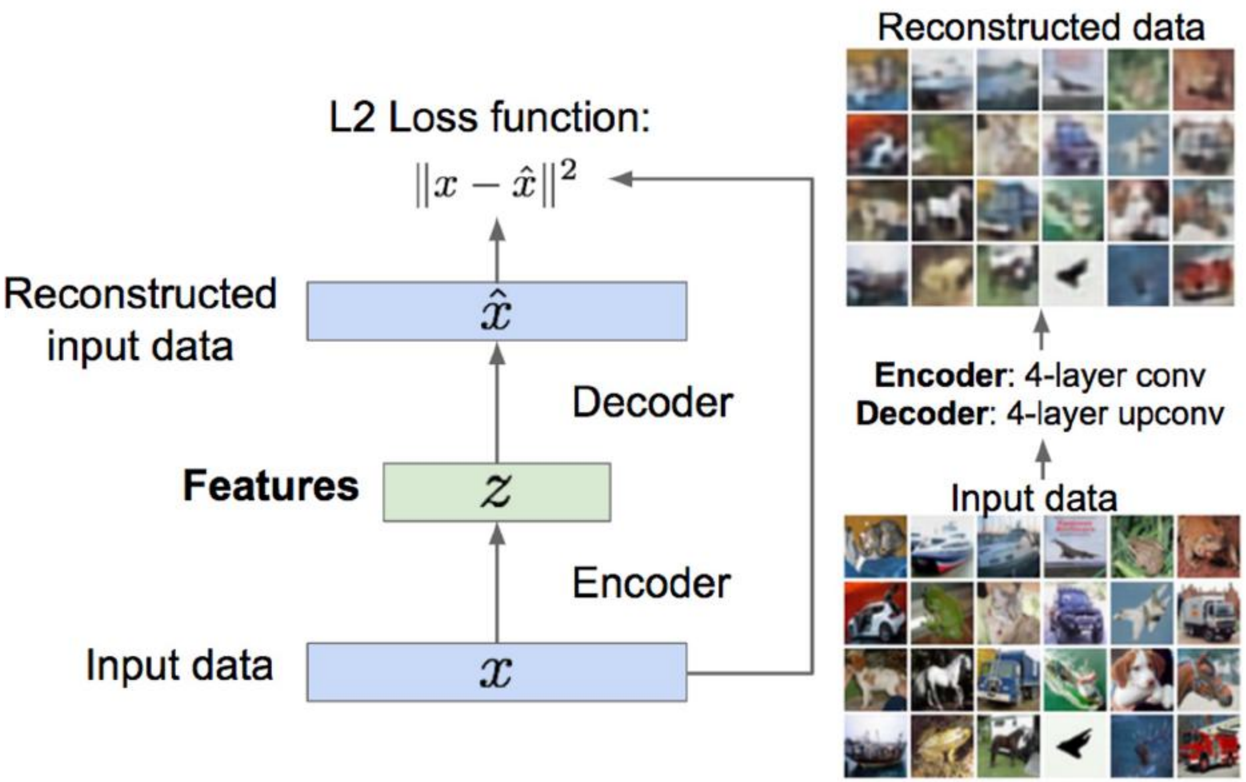
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Feature Learning
(e.g. autoencoders)



Unsupervised Learning

Data: x

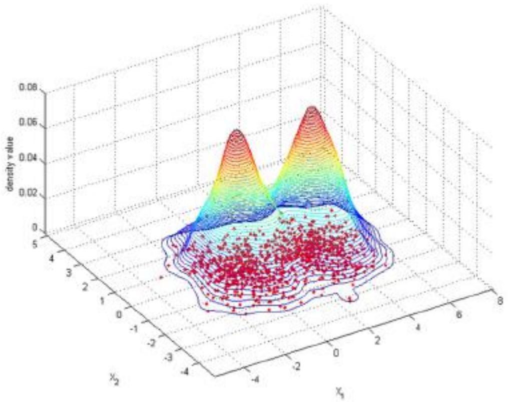
Just data, no labels!

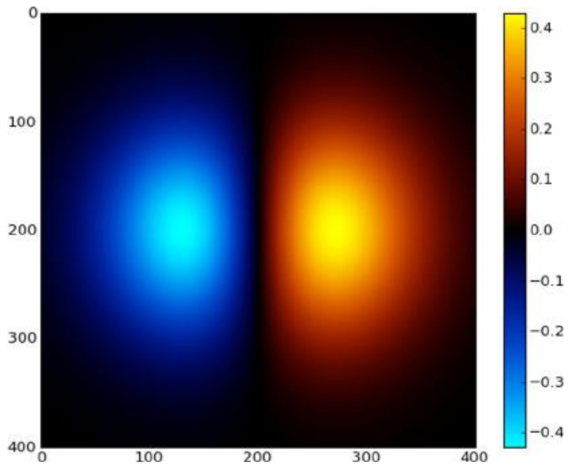
Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Density Estimation





Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a function to map x → y

Examples: classification, regression, object detection, semantic segmentation, image captioning, sentiment analysis, etc.

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

Data: x

Label: y

Cat

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

Data: x

Label: y
Cat

Probability Recap:

Density Function
p(x) assigns a positive number to each possible x; higher numbers mean x is more likely

Density functions are **normalized**:

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

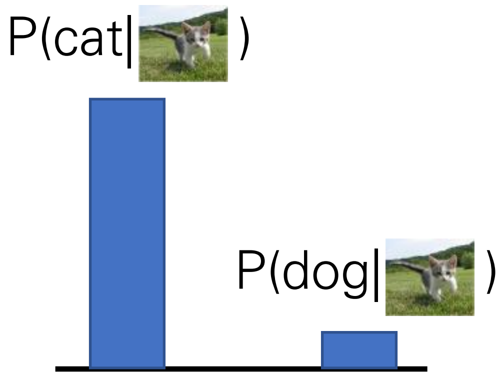
Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

Data: x



Density Function
p(x) assigns a positive number to each possible x; higher numbers mean x is more likely

Density functions are **normalized**:

$$\int_X p(x)dx = 1$$

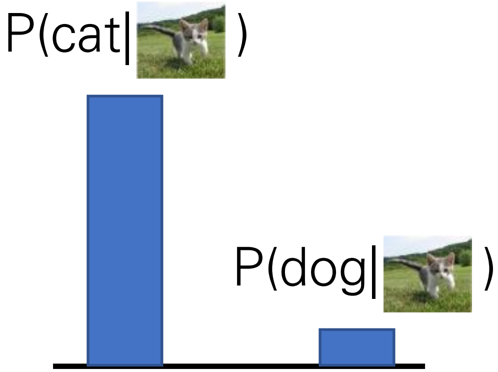
Different values of x **compete** for density

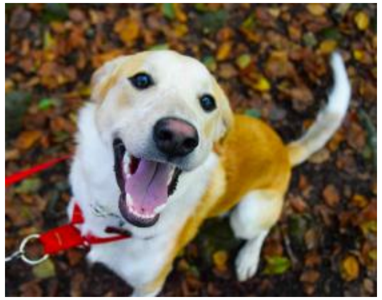
Discriminative vs Generative Models

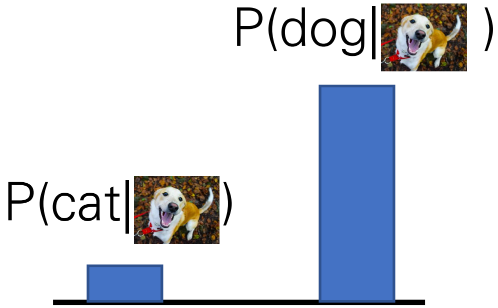
Discriminative Model:
Learn a probability distribution p(y|x)

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)







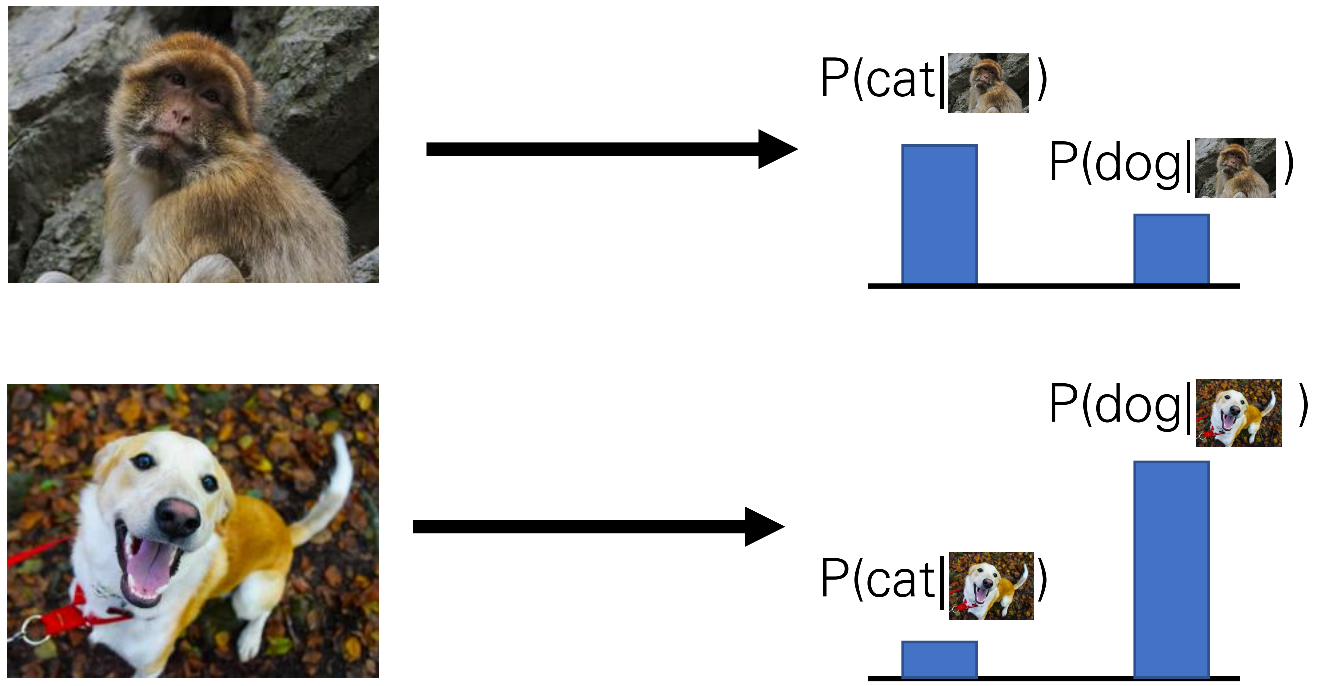
Discriminative model: the possible labels for each input "compete" for probability mass. But no competition between **images**

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)



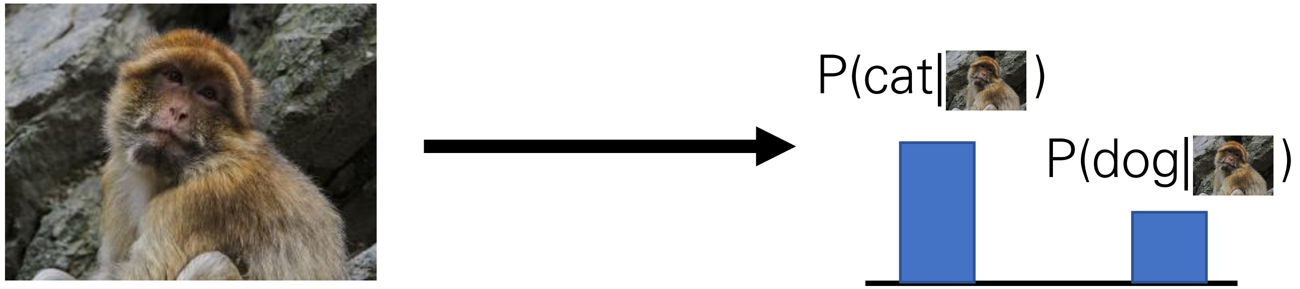
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

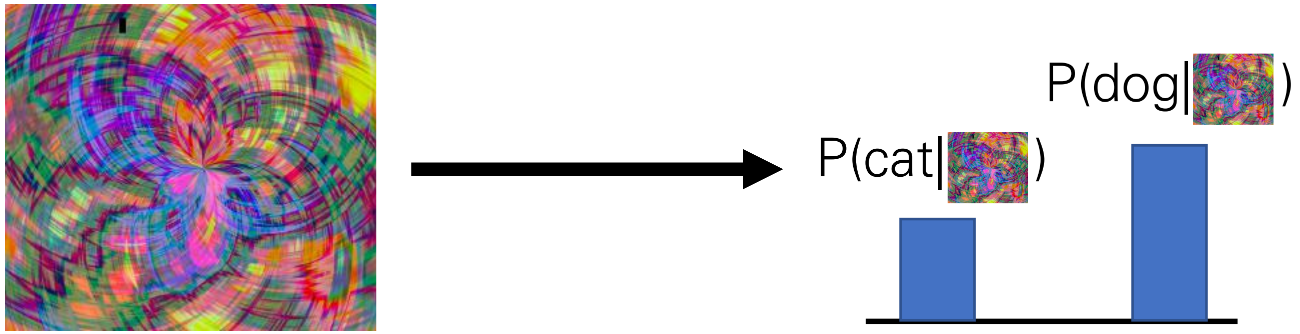
Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)





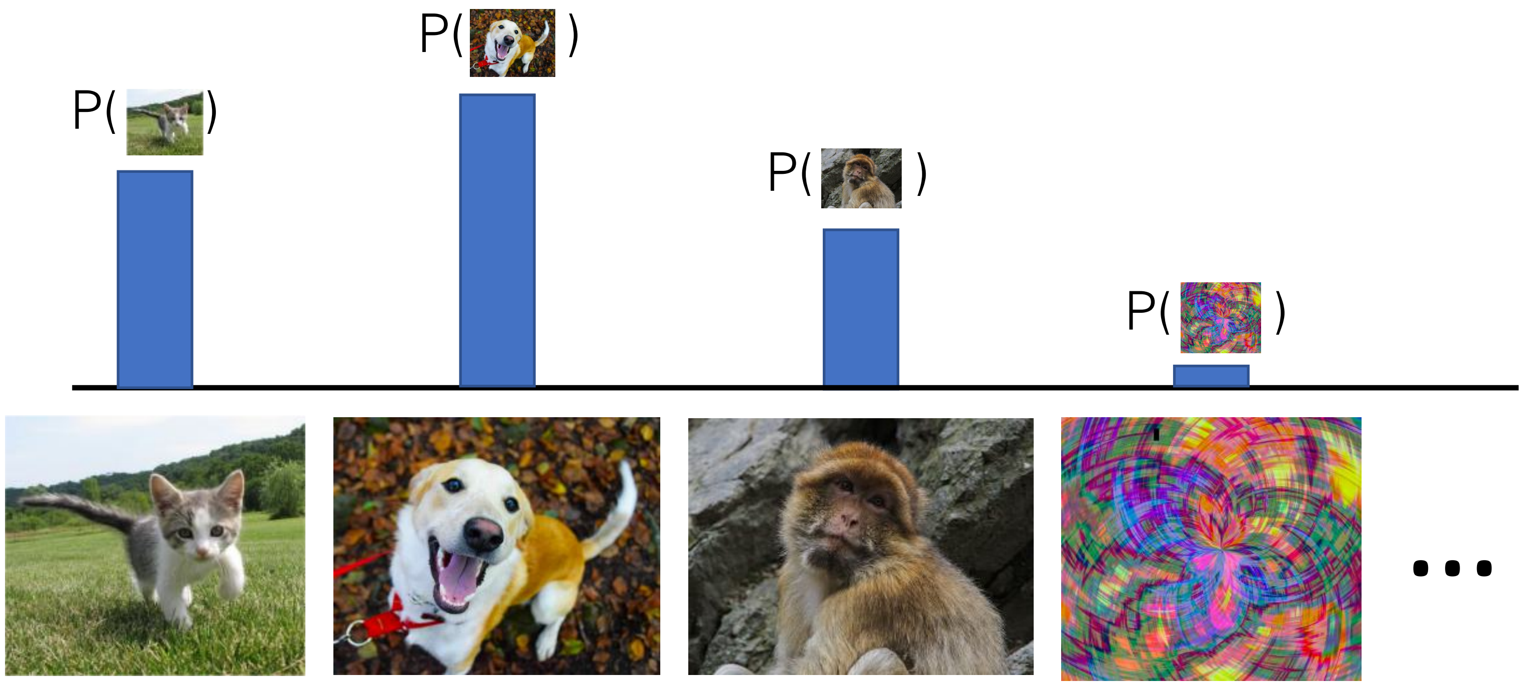
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)



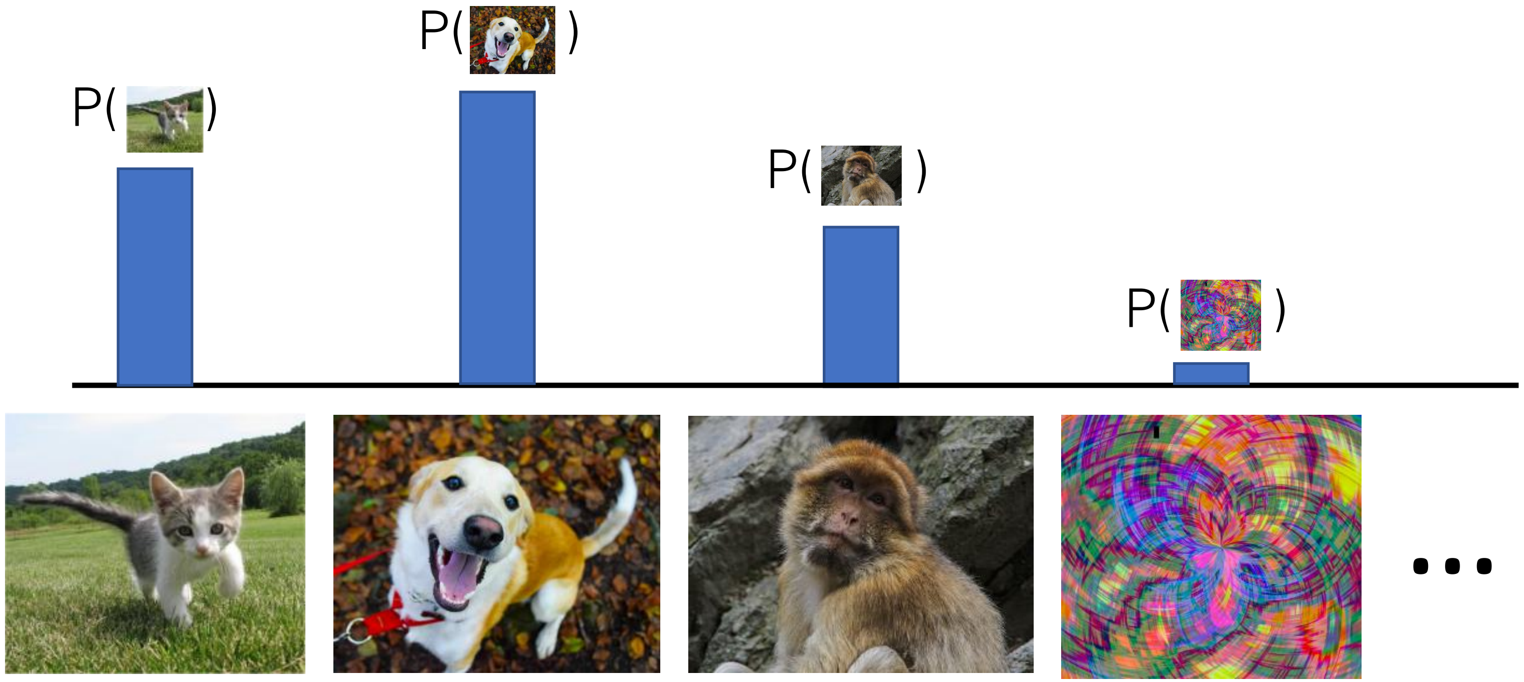
Generative model: All possible images compete with each other for probability mass

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)



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

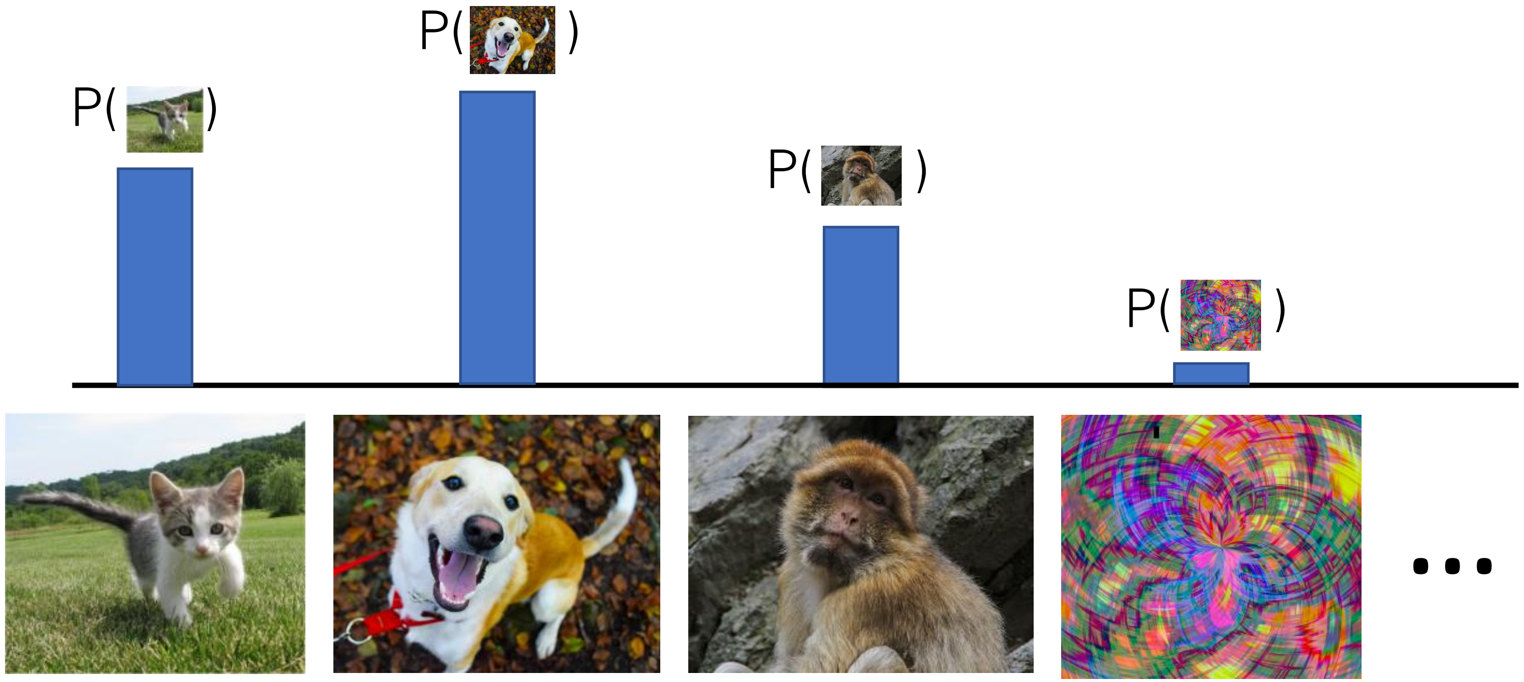
Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)



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

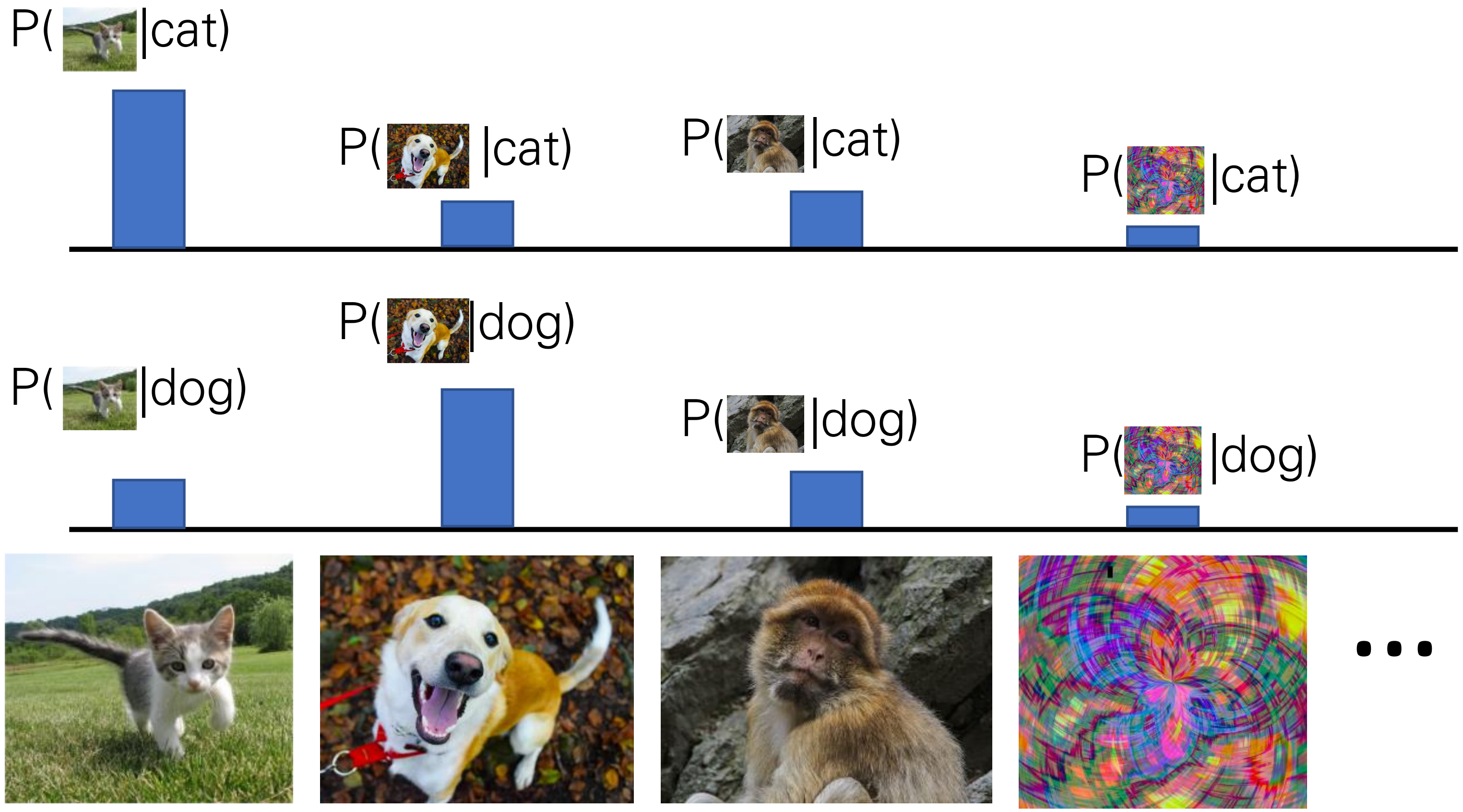
Model can "reject" unreasonable inputs by assigning them small values

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)



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

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

Recall **Bayes' Rule:**

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

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

Recall **Bayes' Rule:**

Discriminative Model

(Unconditional) Generative Model

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

Conditional Generative Model

Prior over labels

We can build a conditional generative model from other components!

What can we do with a discriminative model?

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model:
Learn p(x|y)

What can we do with a discriminative model?

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

→ Assign labels to data
Feature learning (with labels)

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

→ Detect outliers
Feature learning (without labels)
Sample to **generate** new data

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

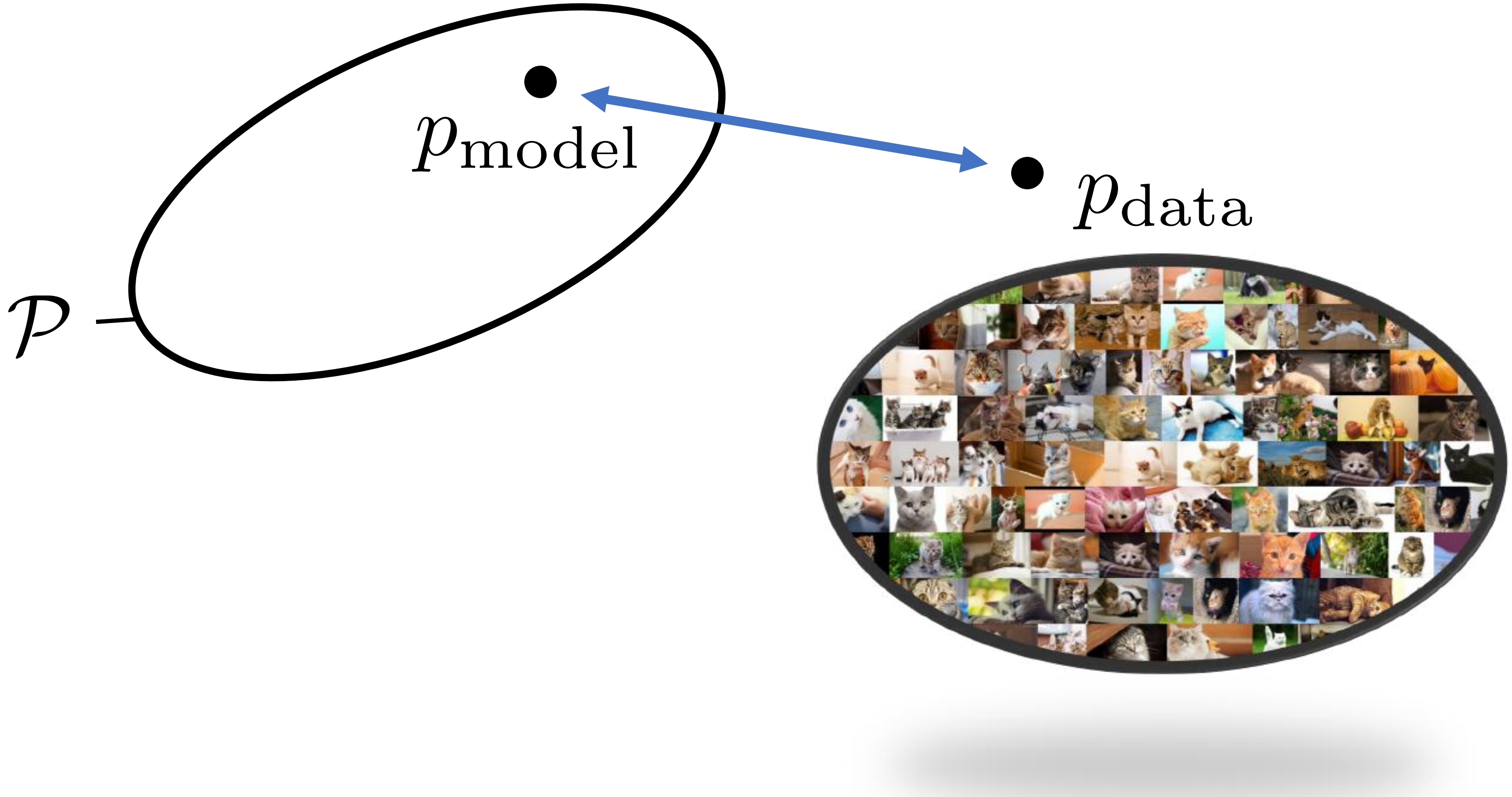
What can we do with a discriminative model?

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

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

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

Generative Modeling

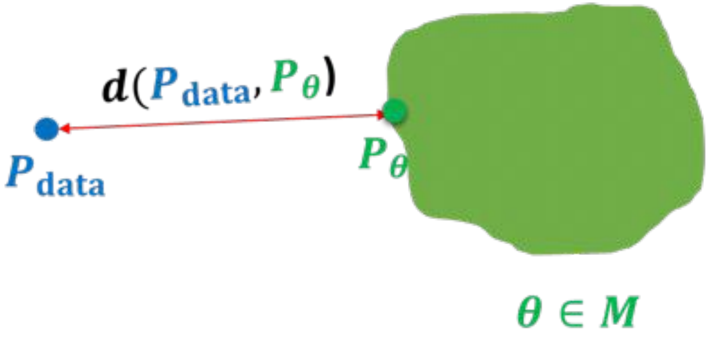


- **Goal:** Learn some underlying hidden structure of the training samples to generate novel samples from same data distribution

Learning a generative model

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

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



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)

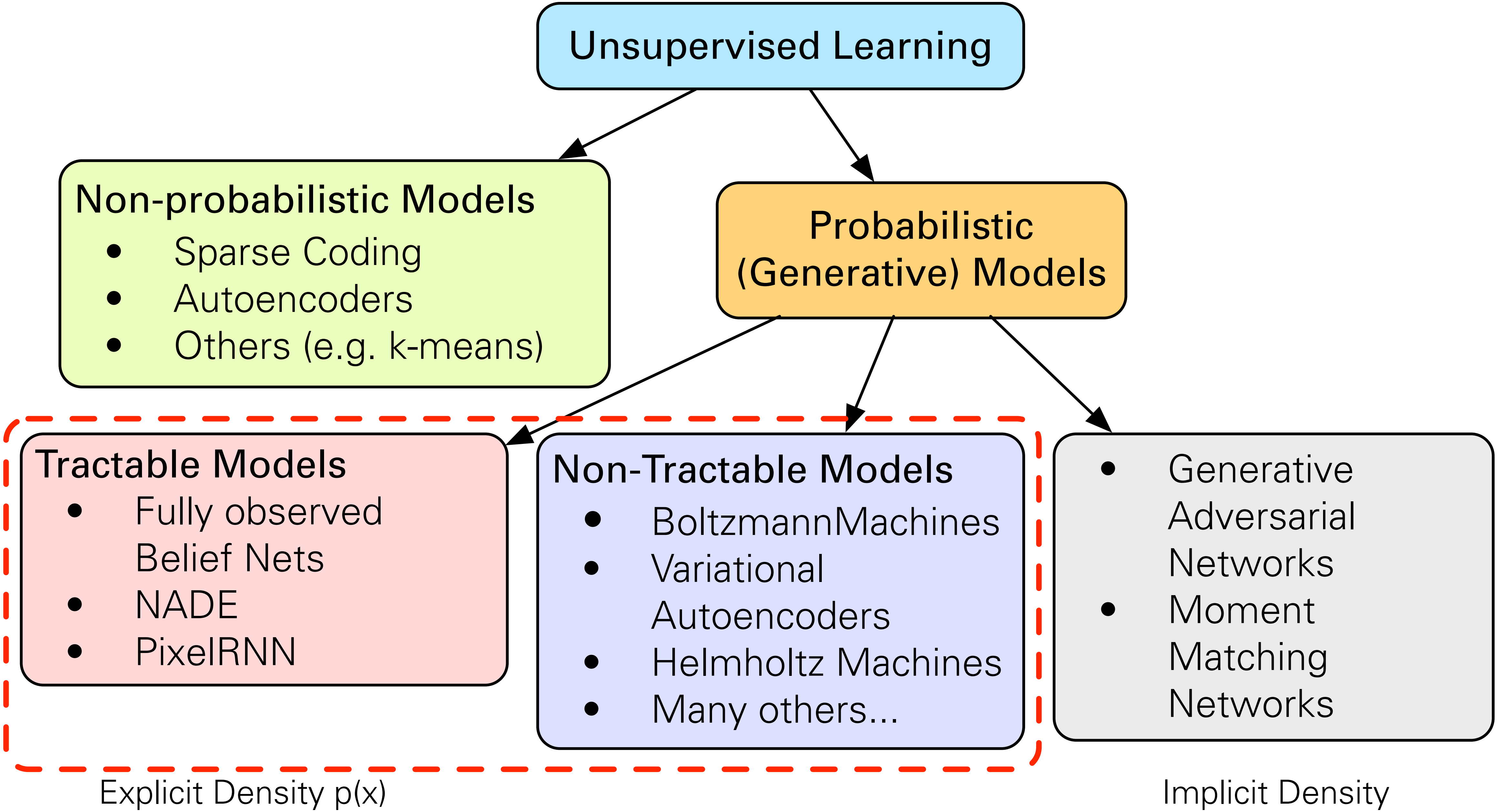
Why Unsupervised Learning?

- 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
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

Unsupervised Learning

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Autoregressive Generative Models
- Generative Adversarial Networks
- Variational Autoencoders
- Normalizing Flow Models

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

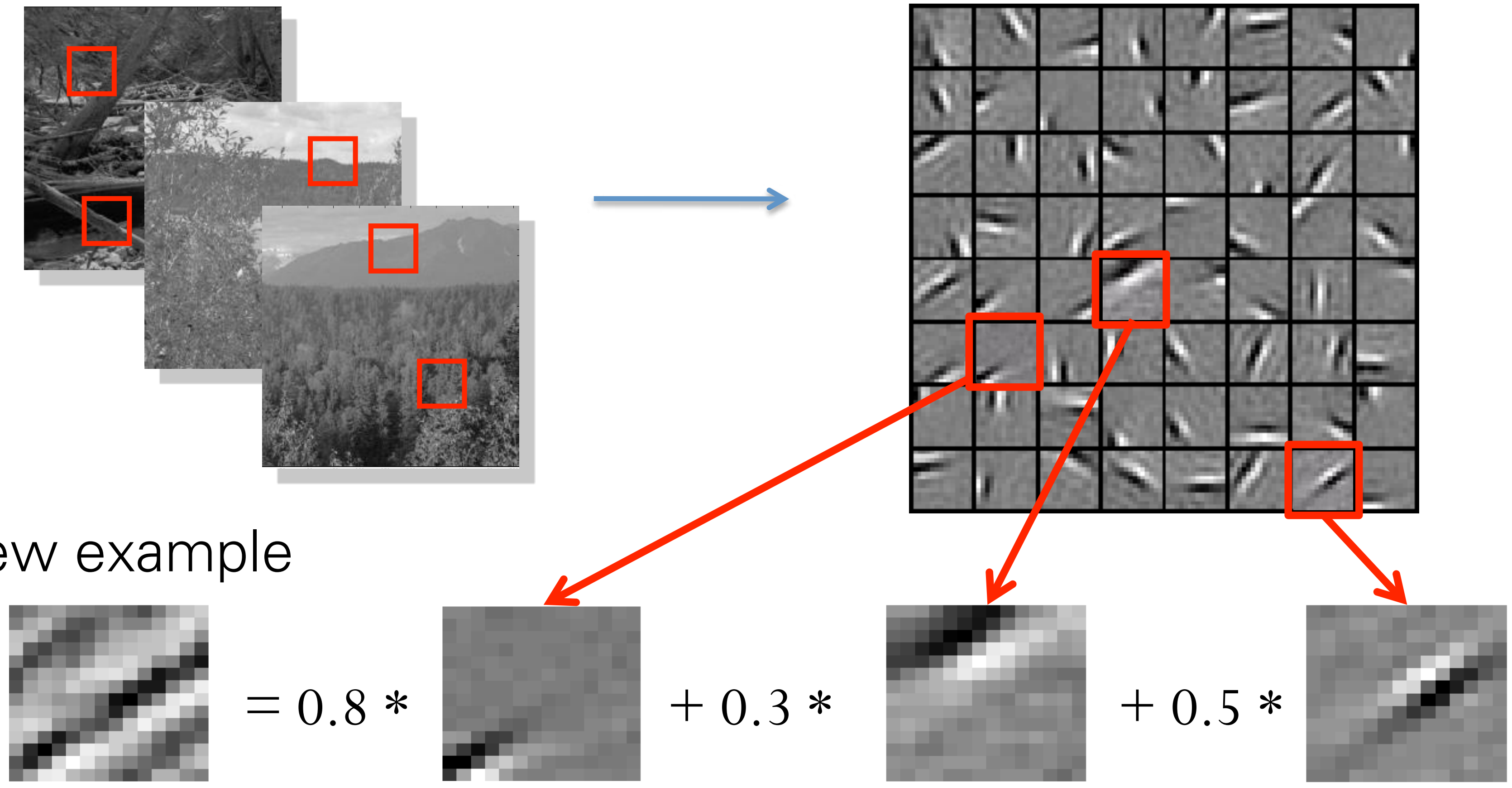
- Each data vector is represented as a sparse linear combination of bases.

Sparse Coding

Natural Images

Learned bases: "Edges"

New example



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

[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: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K \in \mathbb{R}^D$

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

- 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

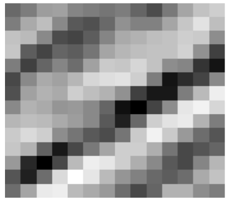
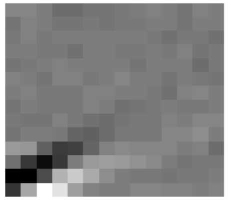
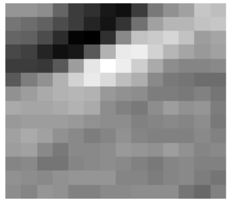
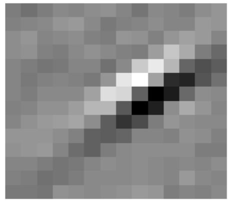
- **Input:** a new image patch x* , and K learned bases $\phi_1, \phi_2, ..., \phi_K$
- **Output:** sparse representation **a** of an image patch x*.

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

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*.

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

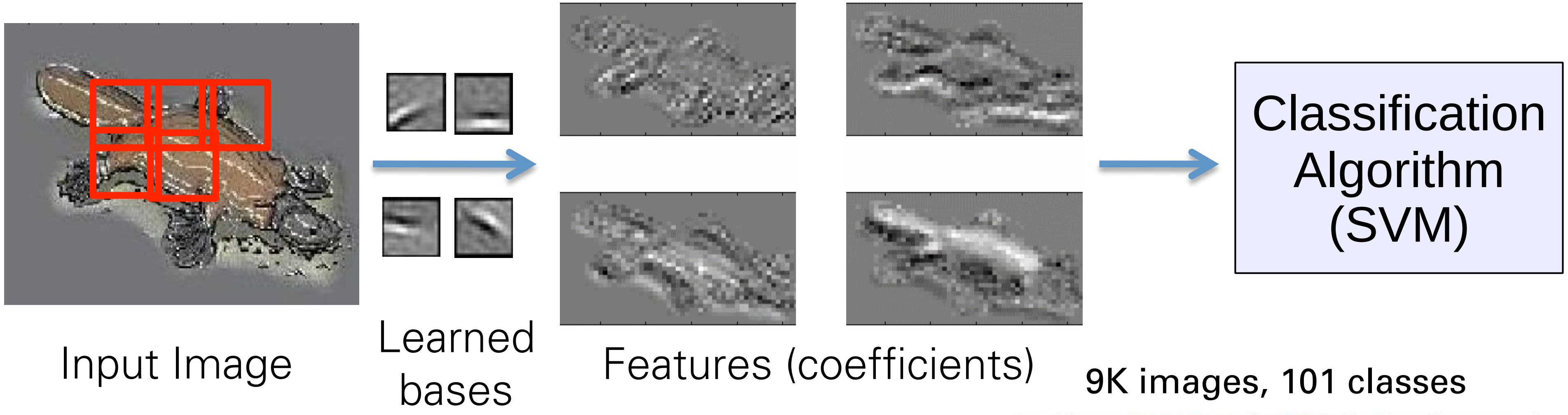
 = 0.8 *  + 0.3 *  + 0.5 * 

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

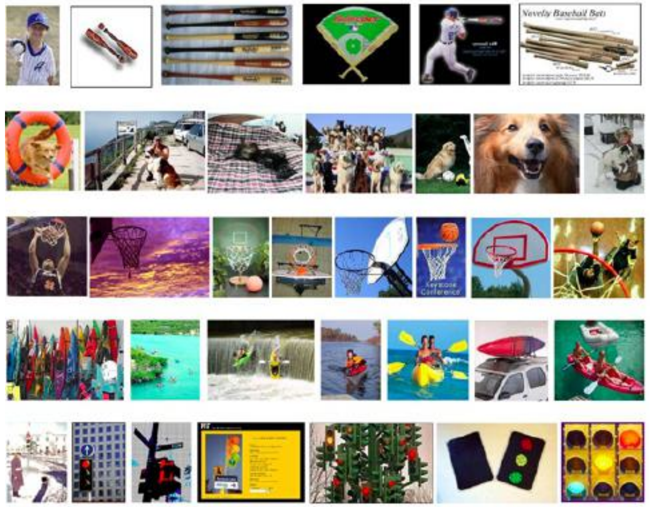
[0.0, 0.0, ... **0.8**, ..., **0.3**, ..., **0.5**, ...] = coefficients (feature representation)

Image Classification

- Evaluated on Caltech101 object category dataset.



9K images, 101 classes



Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
Sparse Coding	**47%**

Slide Credit: Honglak Lee

(Lee, Battle, Raina, Ng, NIPS 2007)

Modeling Image Patches

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

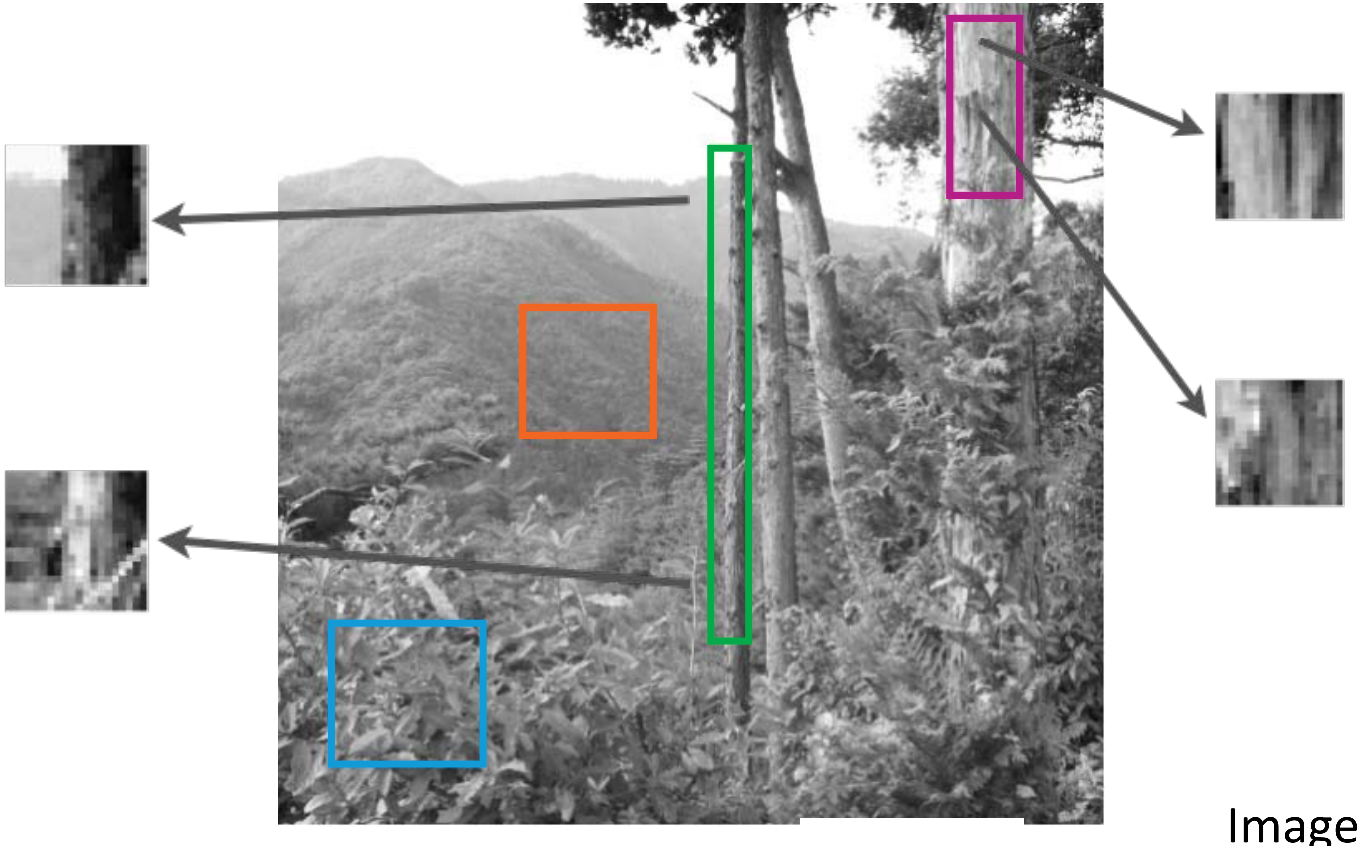
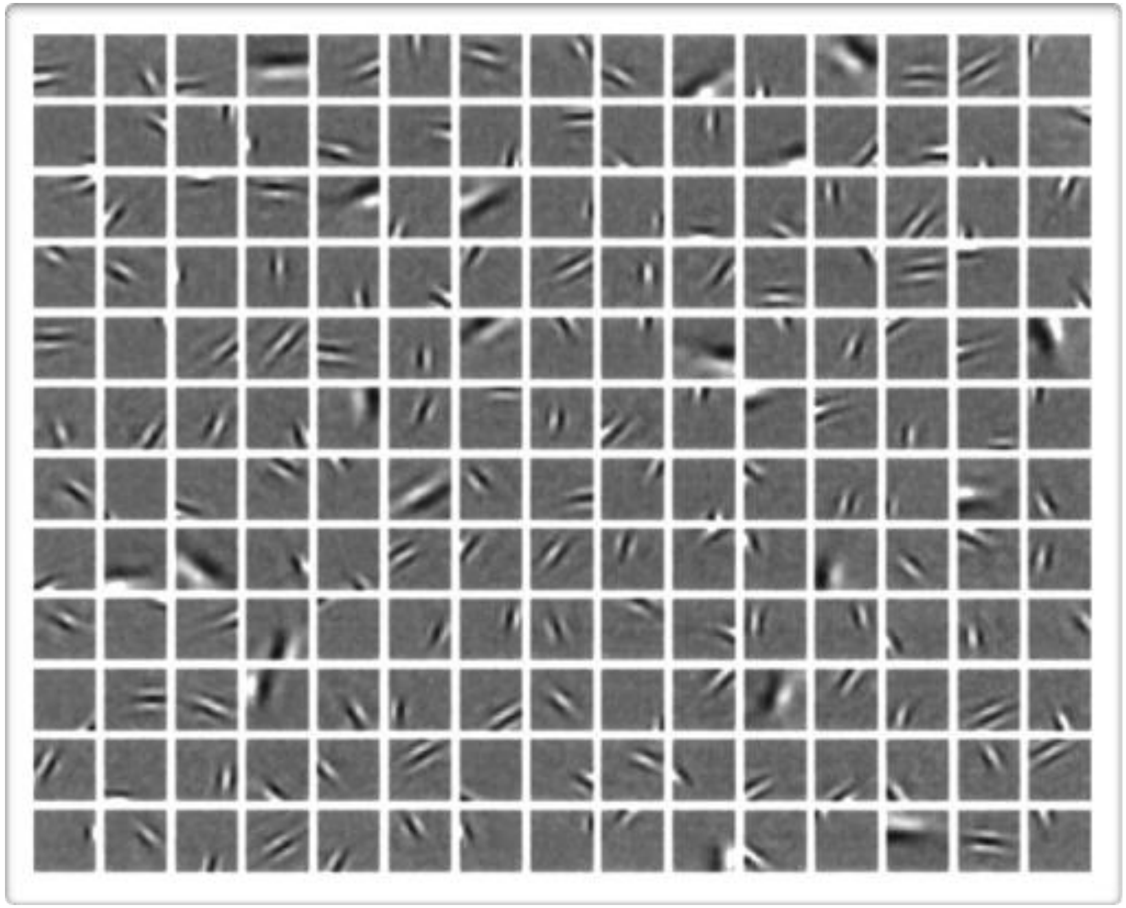


Image taken from: Emergence of complex cell properties by learning to generalize in natural scenes. Karklin and Lewicki, 2009

Relationship to V1

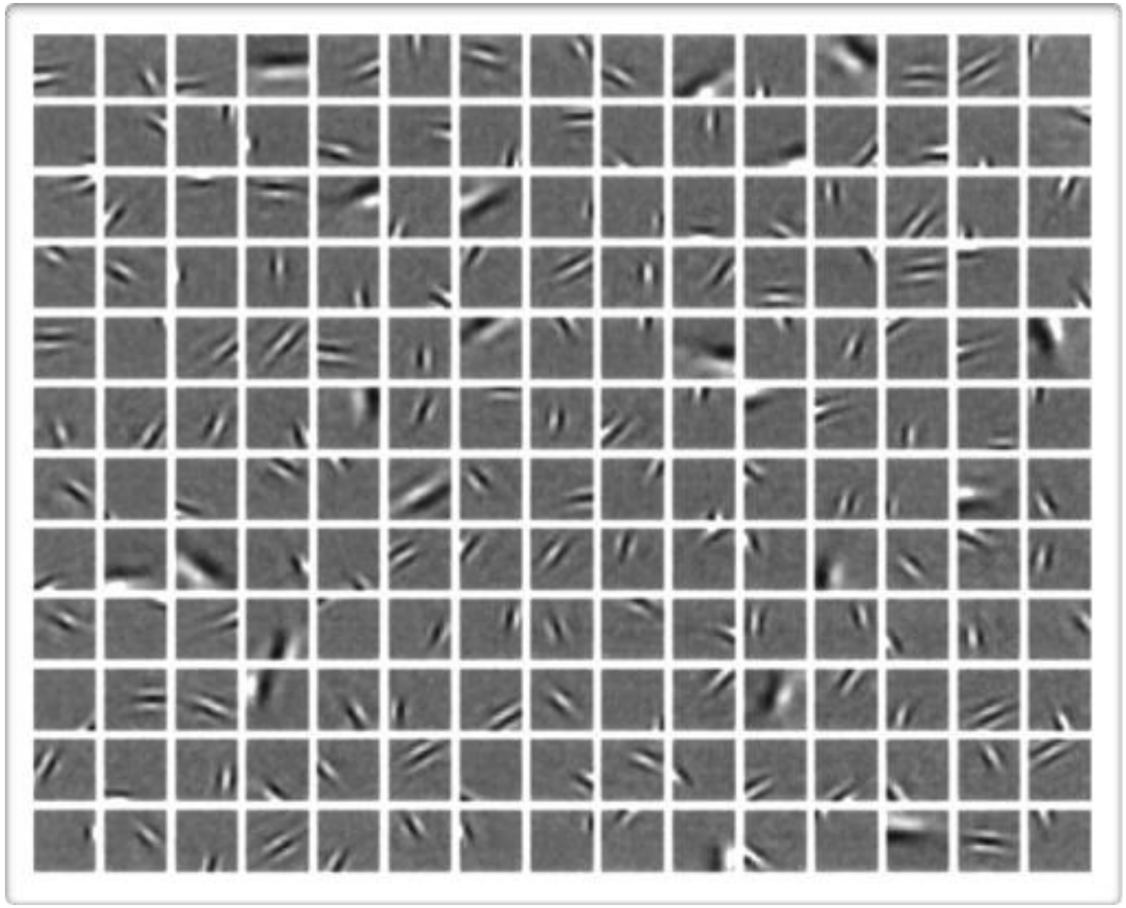
- When trained on natural image patches
  - the dictionary columns (''atoms'') look like **edge detectors**
  - each atom is tuned to a particular **position, orientation** and **spatial frequency**
  - V1 neurons in the mammalian brain have a similar behavior



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

Relationship to V1

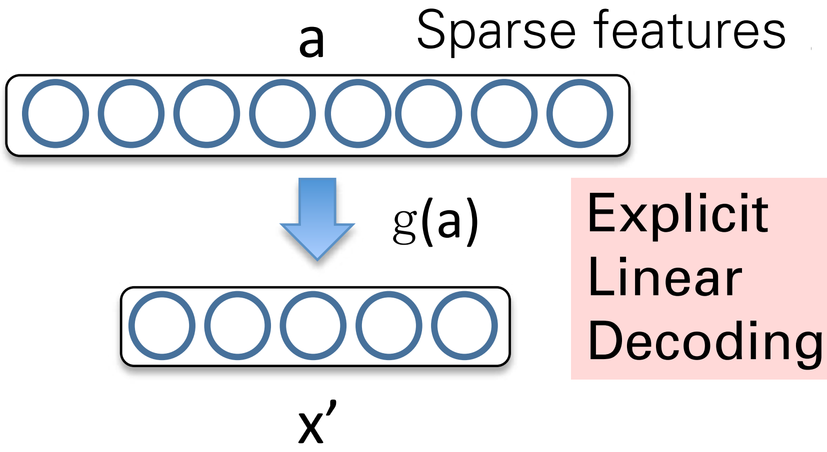
- Suggests that the brain might be learning a sparse code of visual stimulus
  - Since then, many other models have been shown to learn similar features
  - they usually all incorporate a notion of sparsity



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

Interpreting Sparse Coding

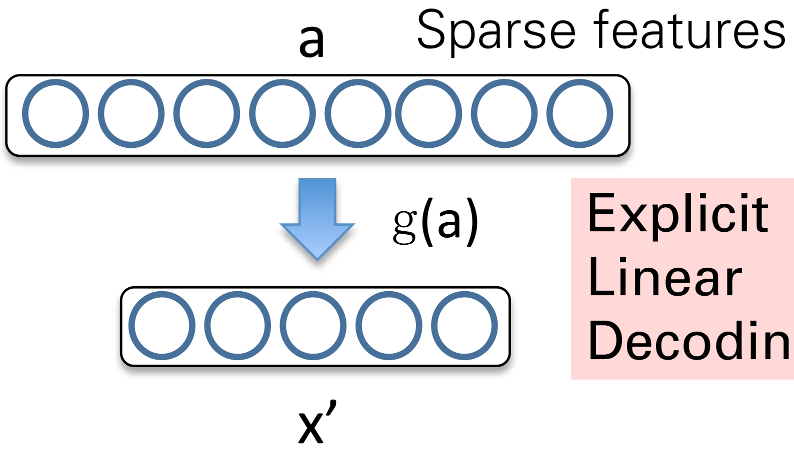
$$\min_{\mathbf{a},\phi} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk} \phi_k \right\|_2^2 + \lambda \sum_{n=1}^{N} \sum_{k=1}^{K} |a_{nk}|$$



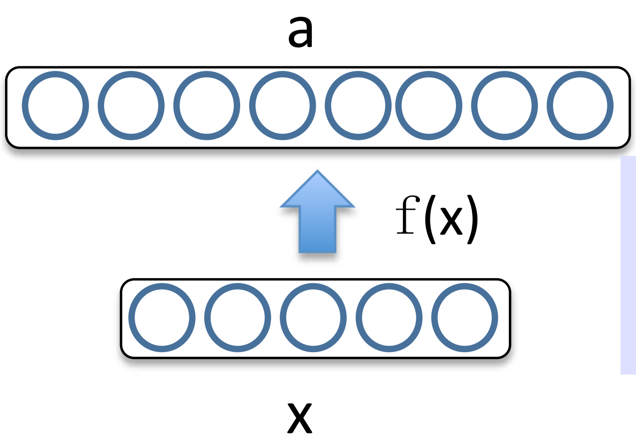
Explicit Linear Decoding

Interpreting Sparse Coding

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



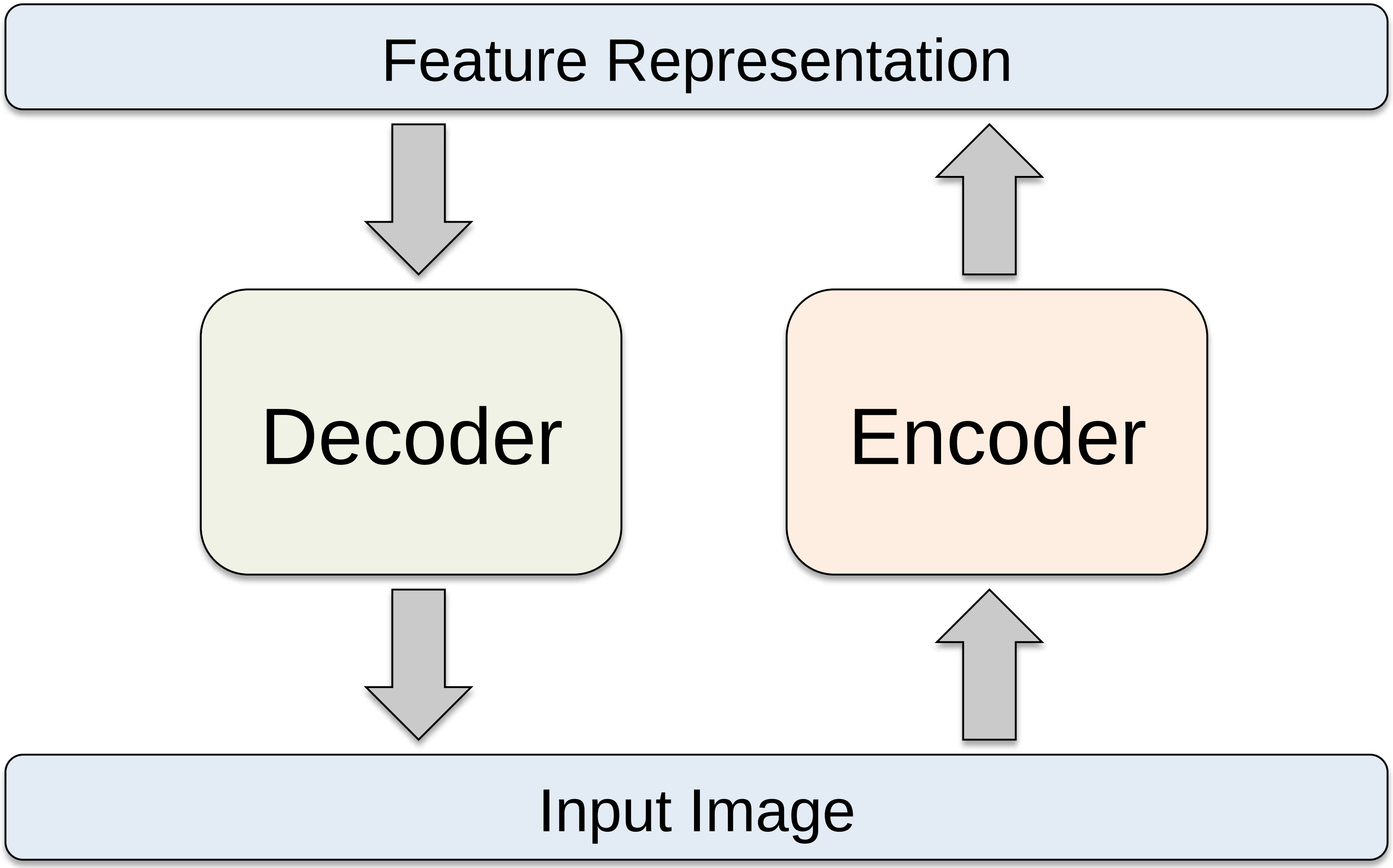
Explicit Linear Decoding



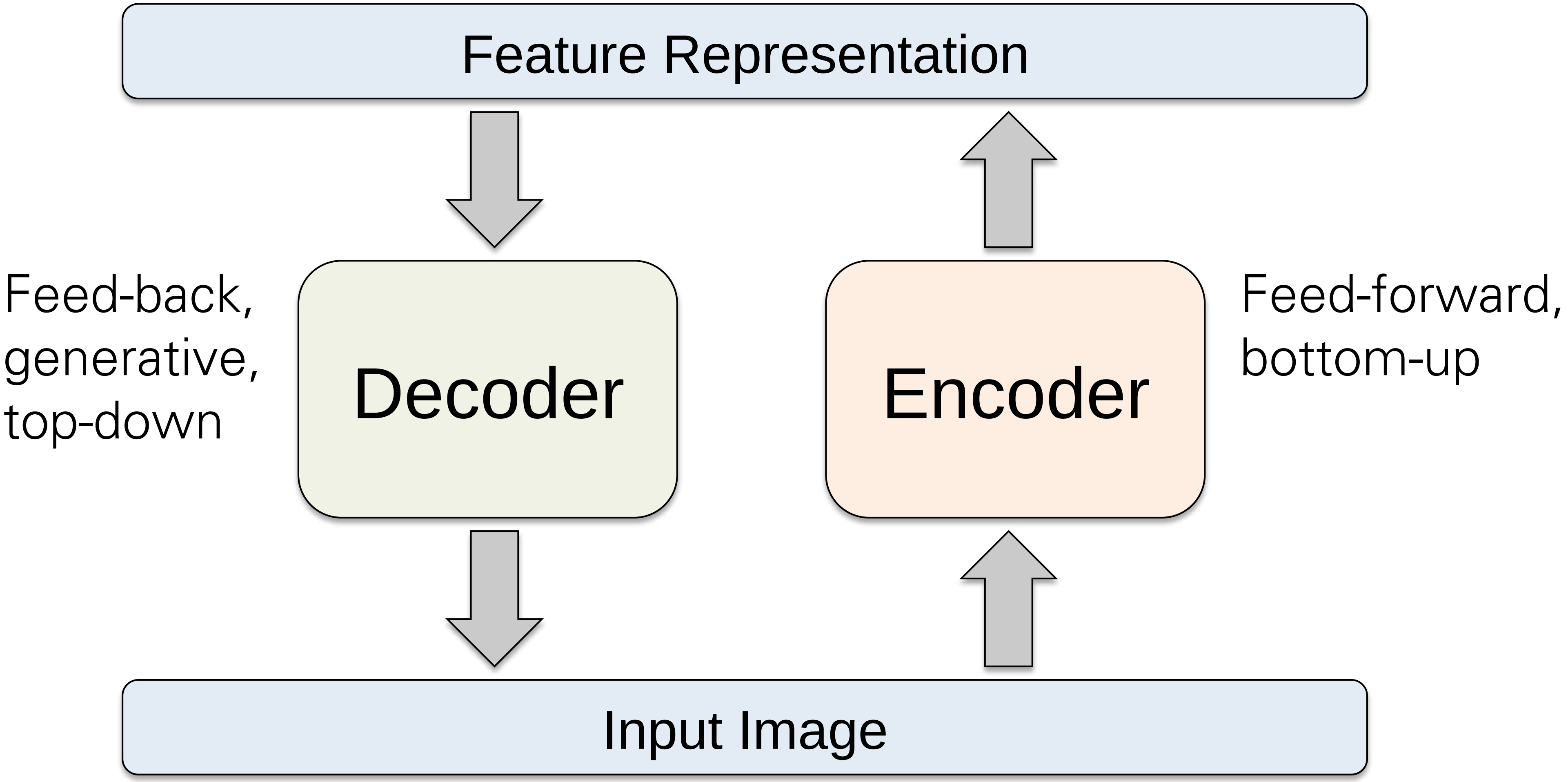
Implicit Nonlinear Encoding

- 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

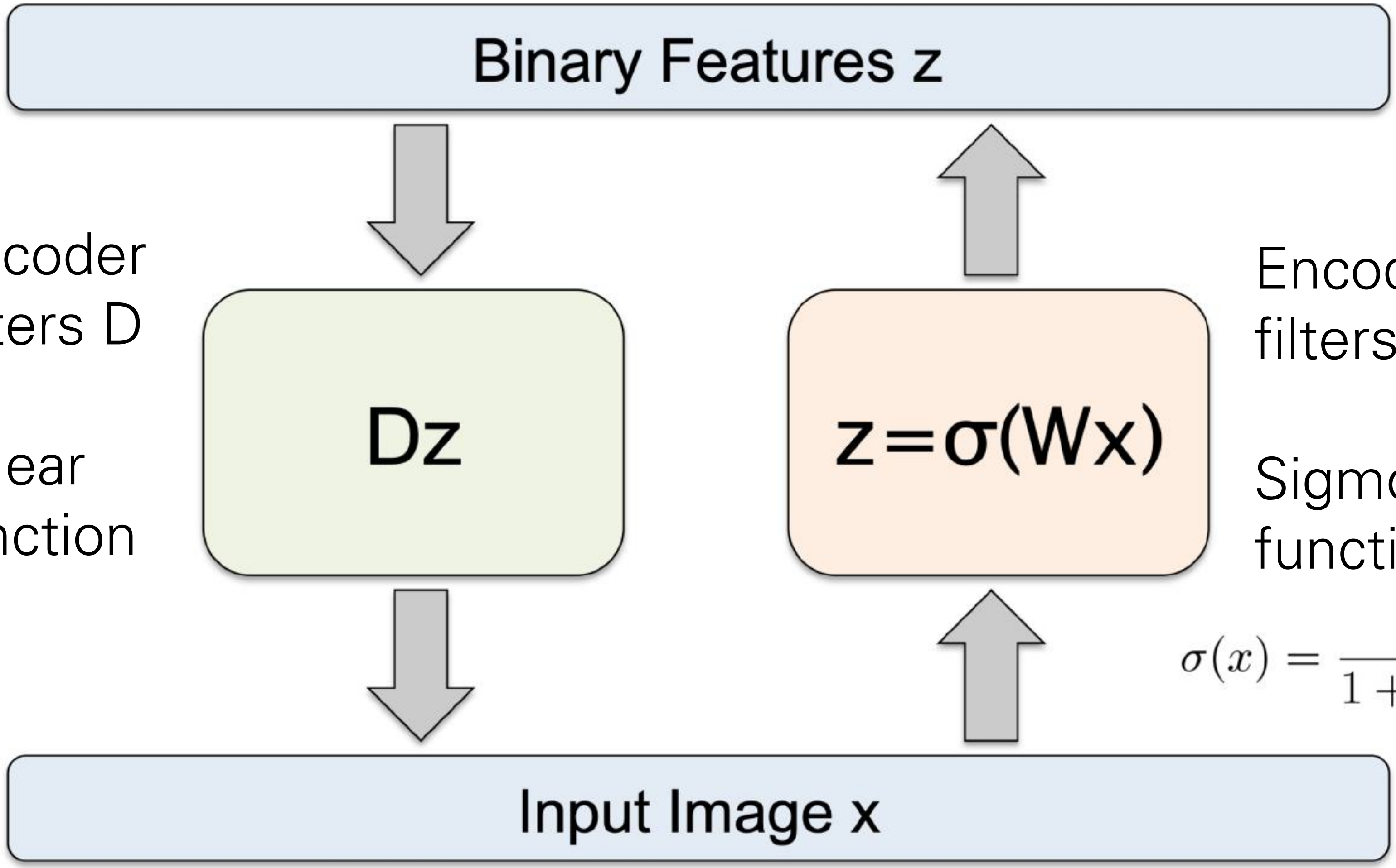


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder

Decoder
Filters D

Linear
function

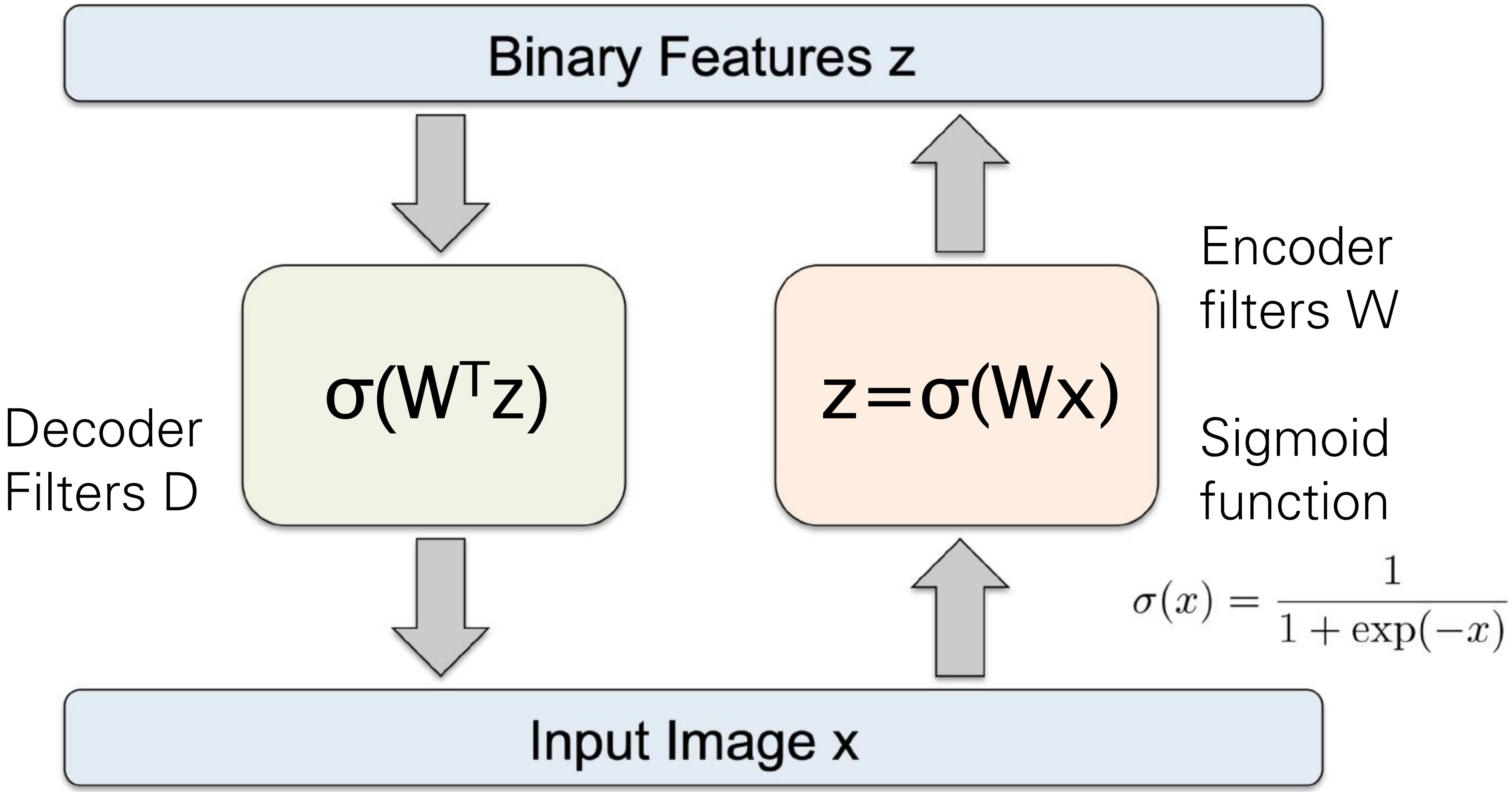


Encoder
filters W

Sigmoid
function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

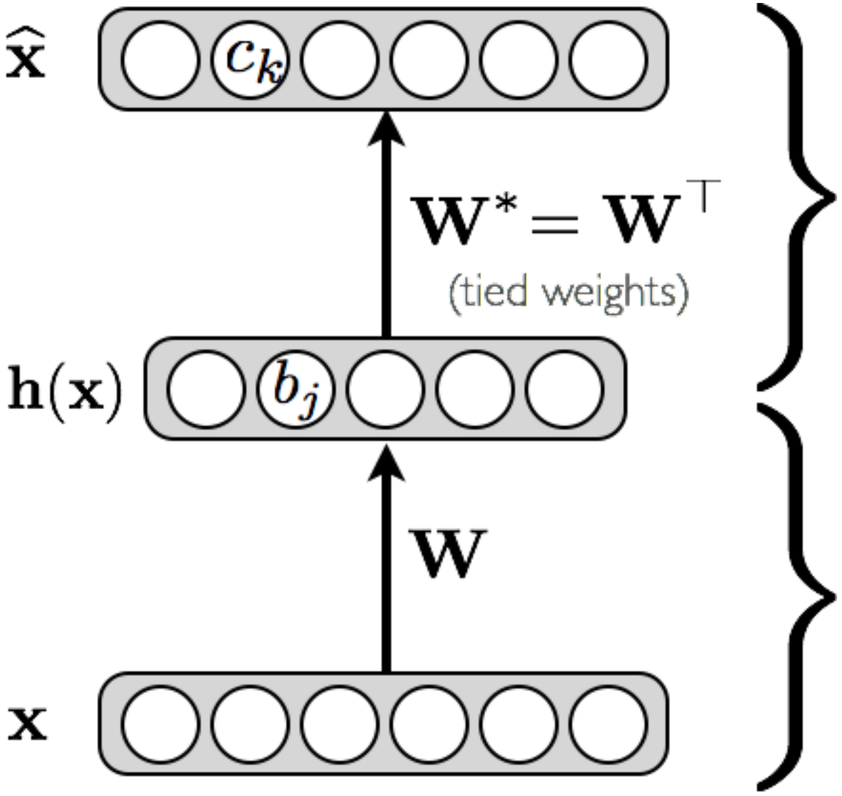
Autoencoder



- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines (later).

Autoencoder

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



Decoder

$$\widehat{\mathbf{x}} = o(\widehat{\mathbf{a}}(\mathbf{x})) = \underbrace{\mathrm{sigm}}_{\text{for binary units}}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))$$

Encoder

$$\mathbf{h}(\mathbf{x}) = g(\mathbf{a}(\mathbf{x})) = \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$$

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)

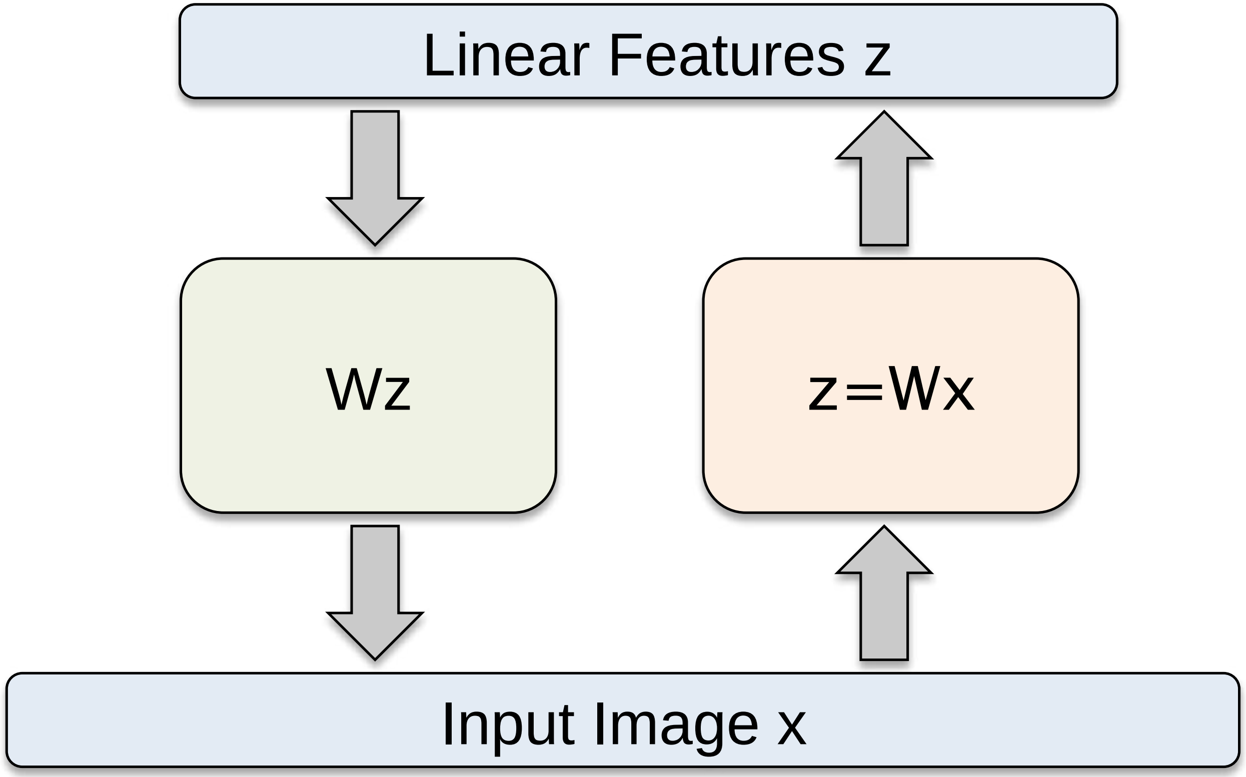
$$f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$$

- Loss function for real-valued inputs

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

  - onstruction loss)
  - we use a linear activation function at the output

Autoencoder

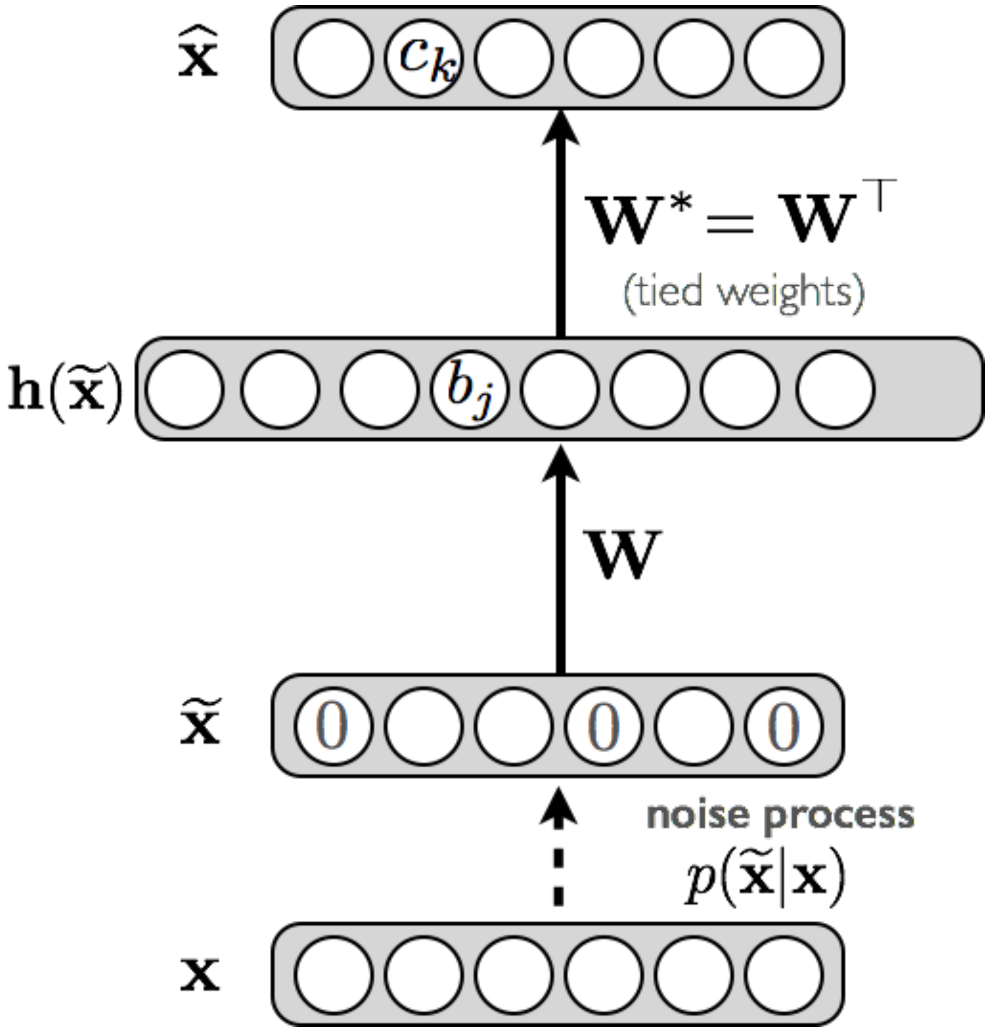


- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.
- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

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

Denoising Autoencoder

- Idea: Representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability ν
  - Similar to dropouts on the input layer
  - Gaussian additive noise

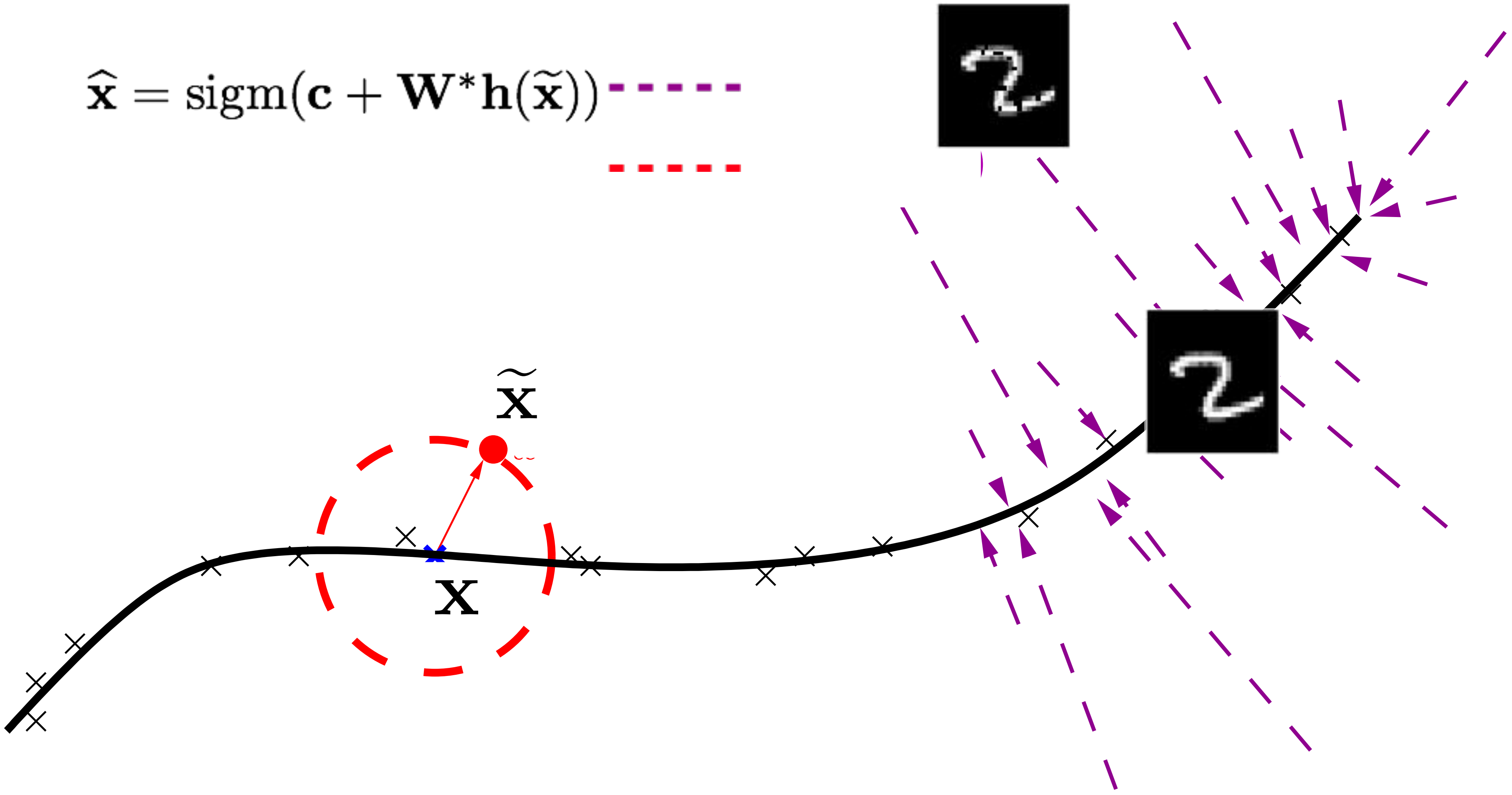


- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$
- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$

(Vincent et al., ICML 2008)

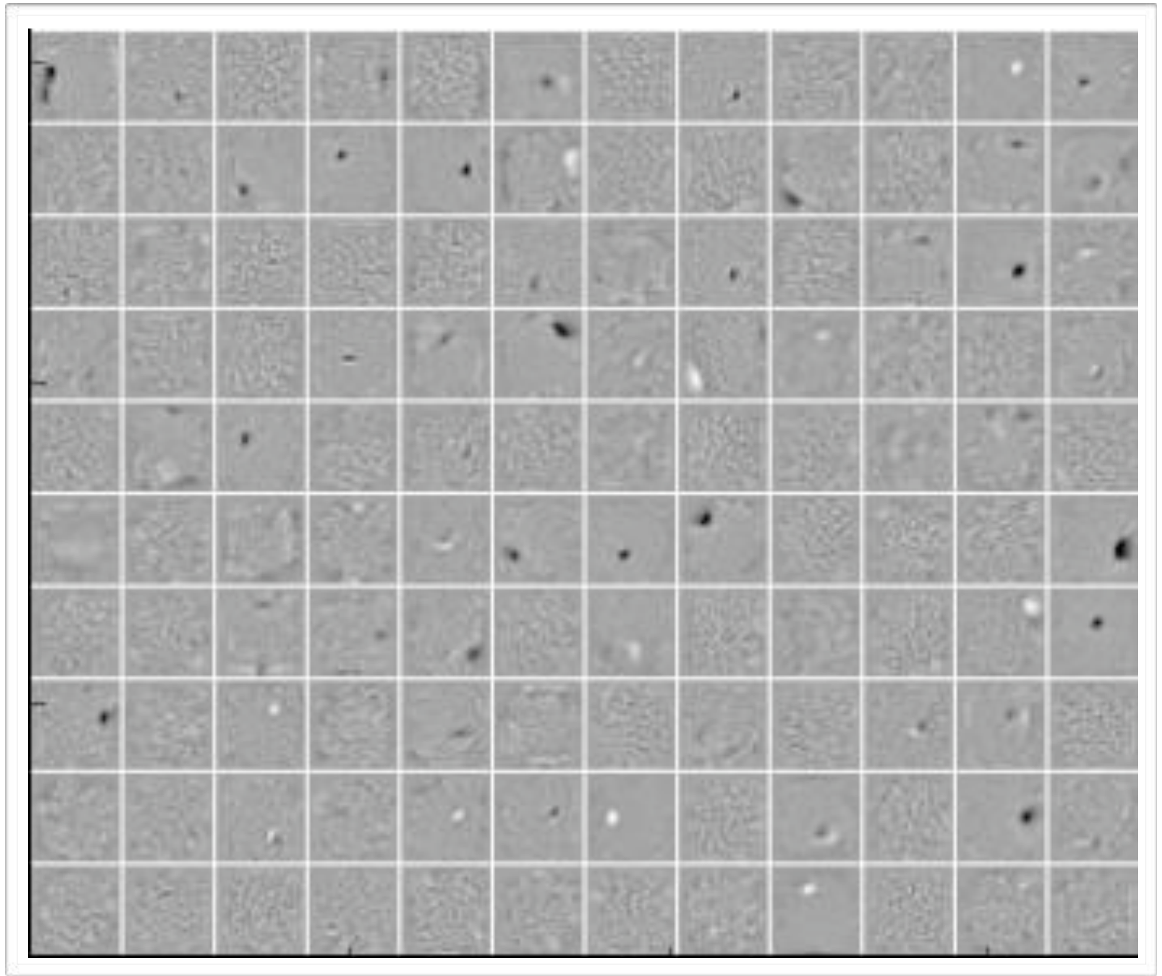
Denoising Autoencoder

$$\widehat{\mathbf{x}} = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\widetilde{\mathbf{x}}))$$

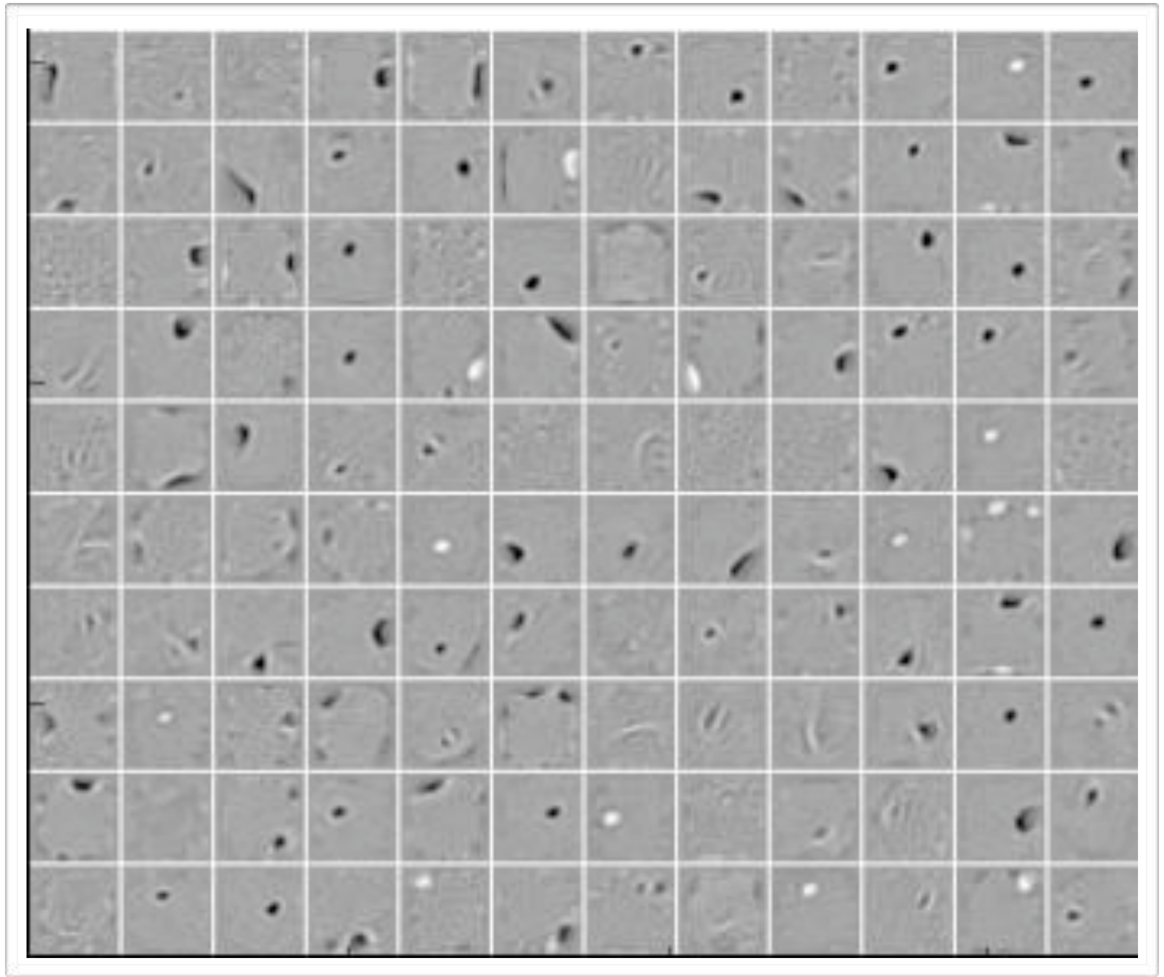


Learned Filters

Non-corrupted



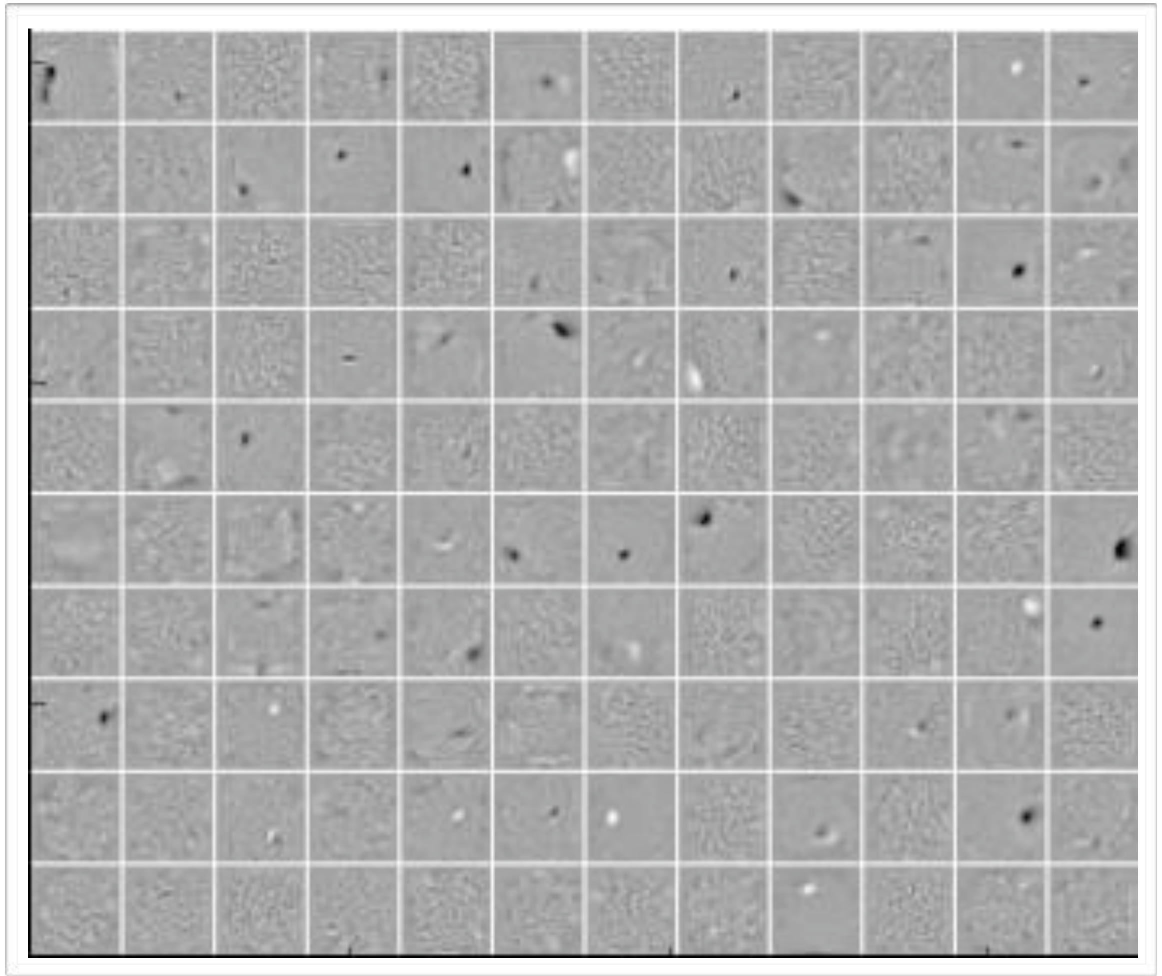
25% corrupted input



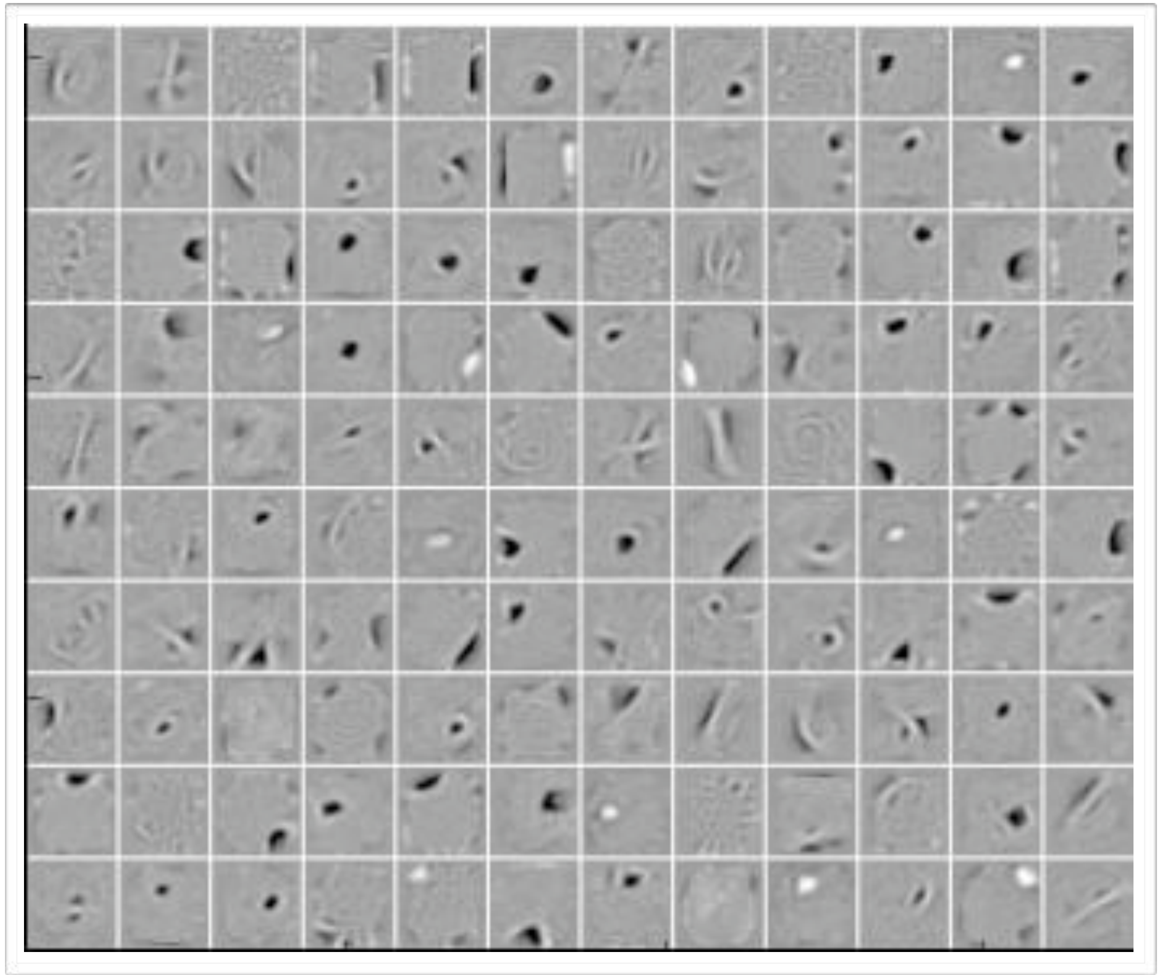
(Vincent et al., ICML 2008)

Learned Filters

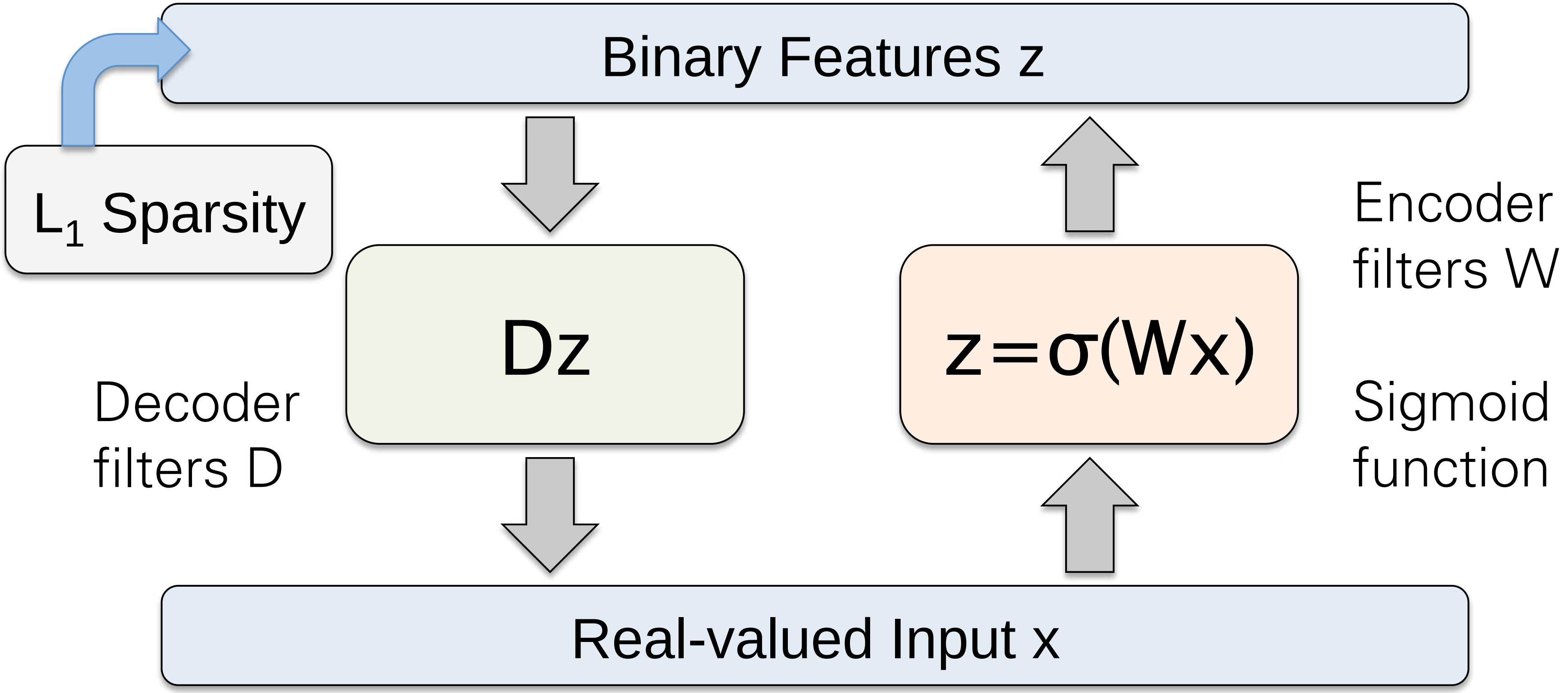
Non-corrupted



50% corrupted input

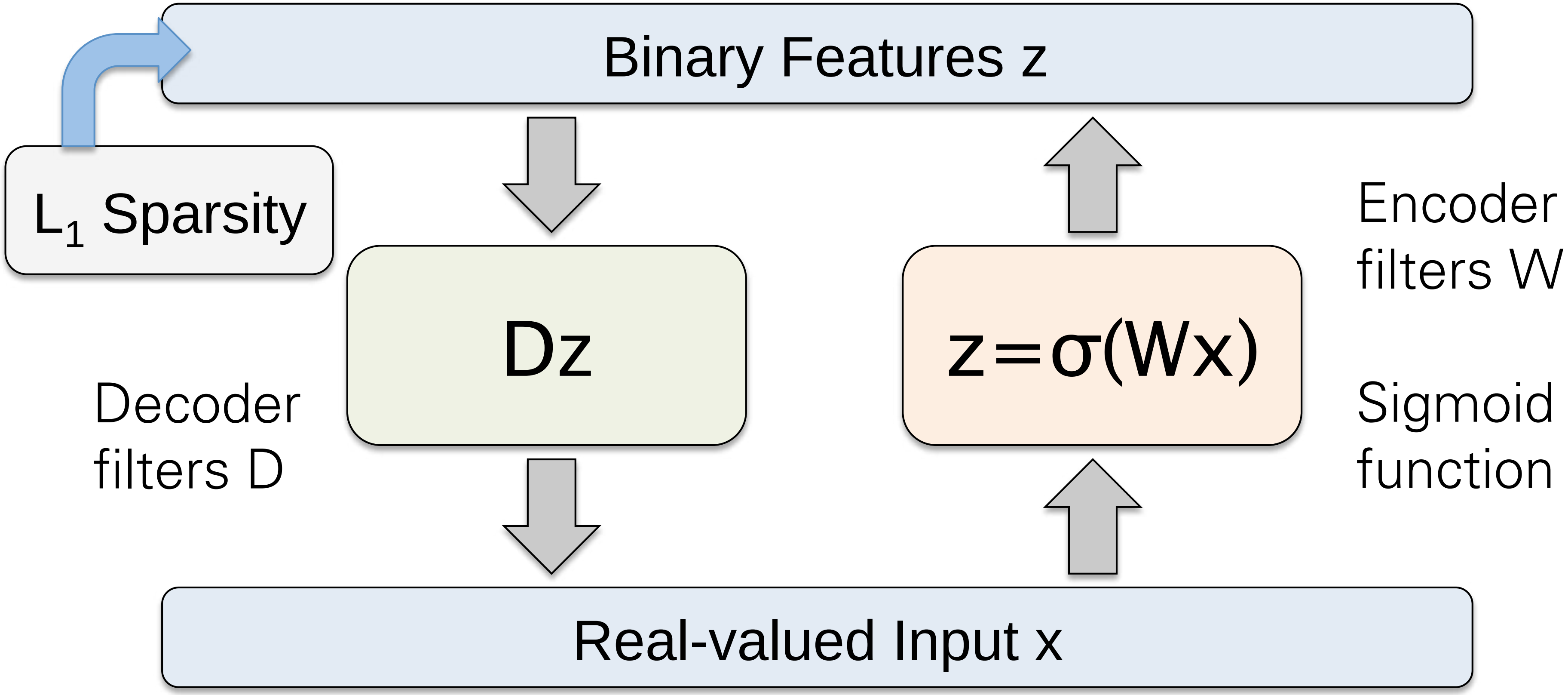


Predictive Sparse Decomposition



(Kavukcuoglu, Ranzato, Fergus, LeCun, 2009)

Predictive Sparse Decomposition

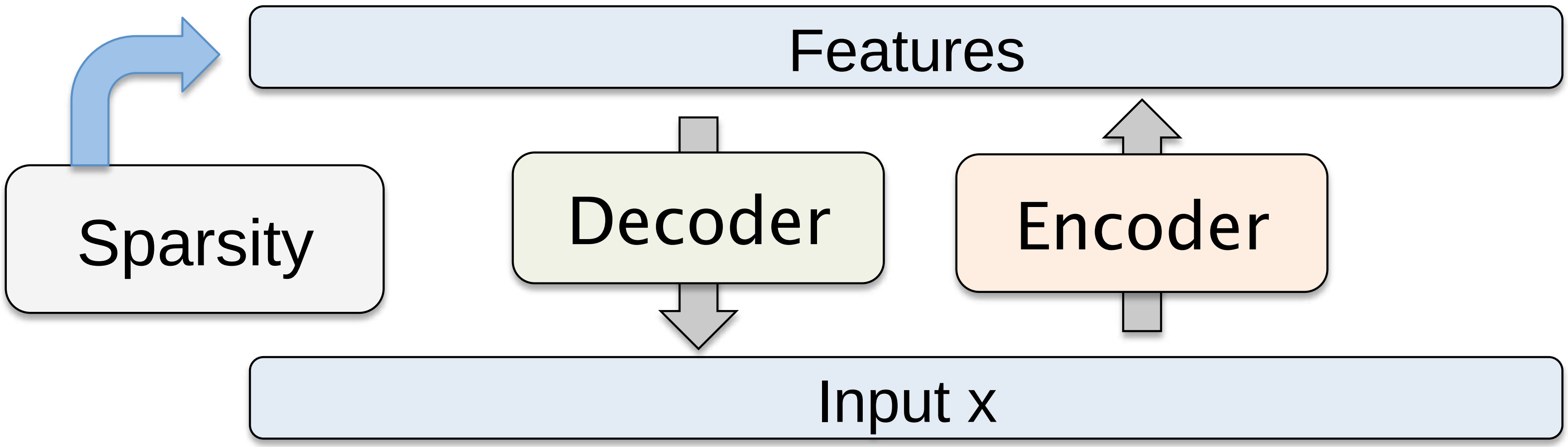


At training time

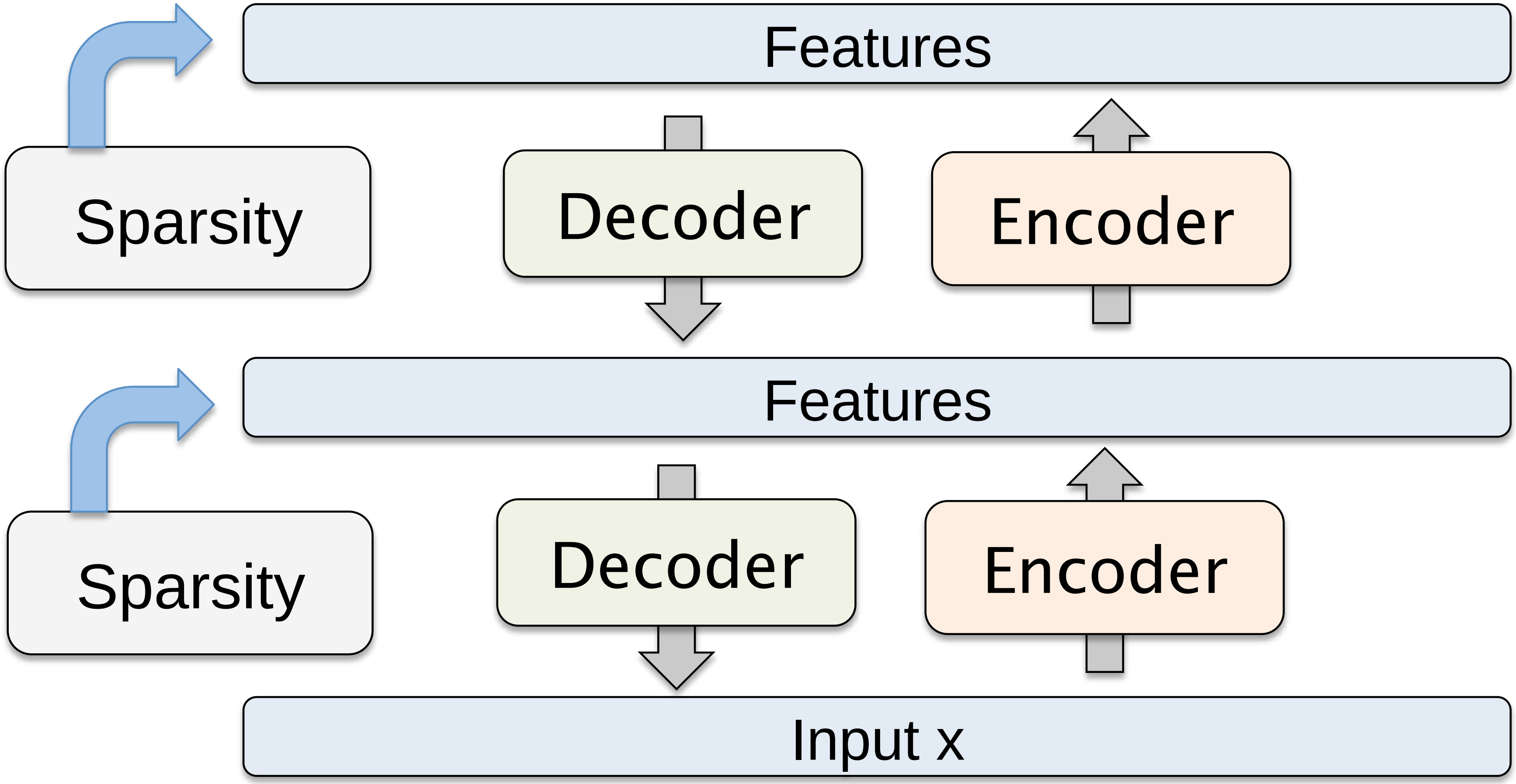
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z} - \mathbf{x}||_2^2 + \lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x}) - \mathbf{z}||_2^2}_{\text{Encoder}}$$

(Kavukcuoglu, Ranzato, Fergus, LeCun, 2009)

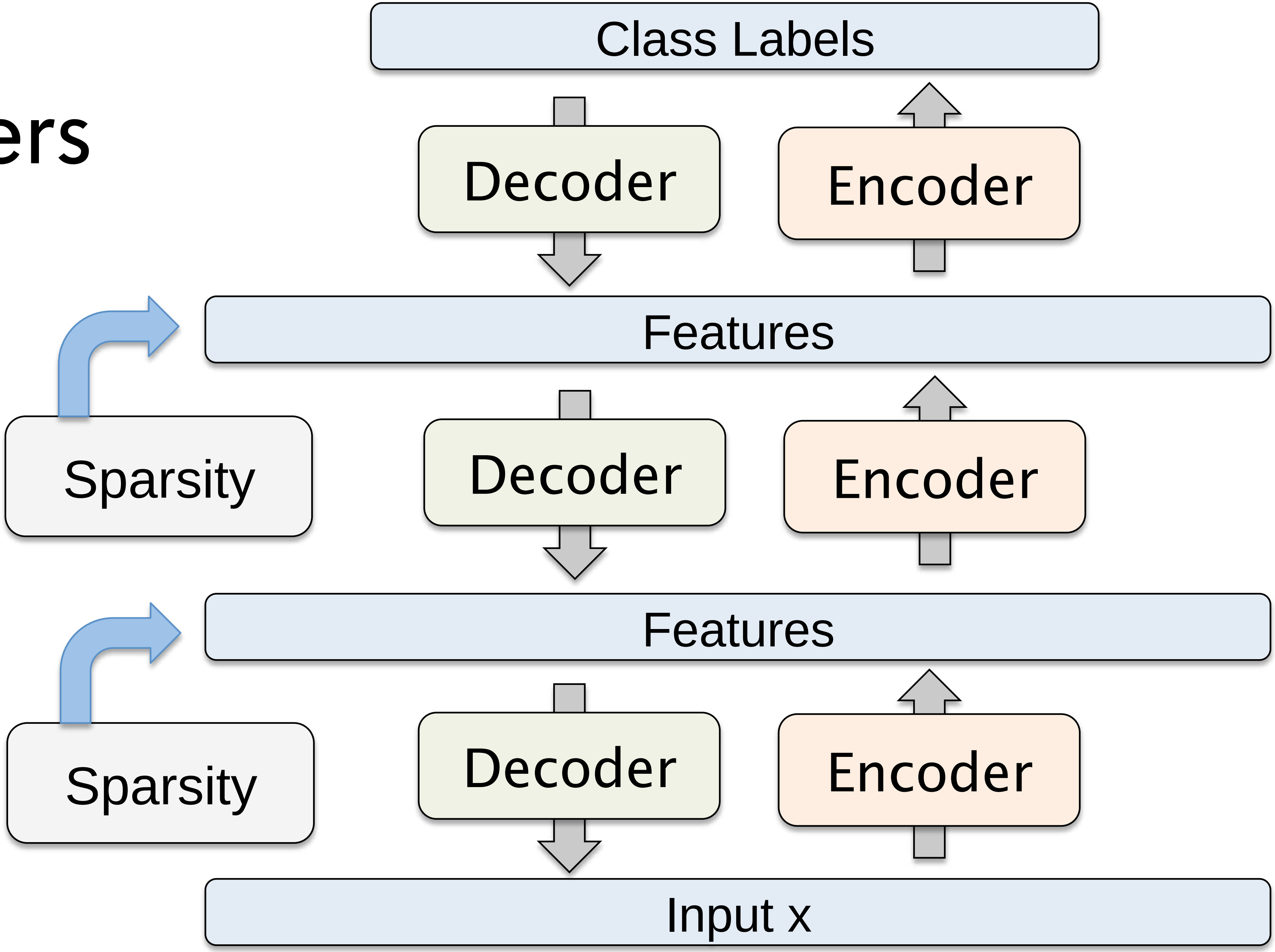
Stacked Autoencoders



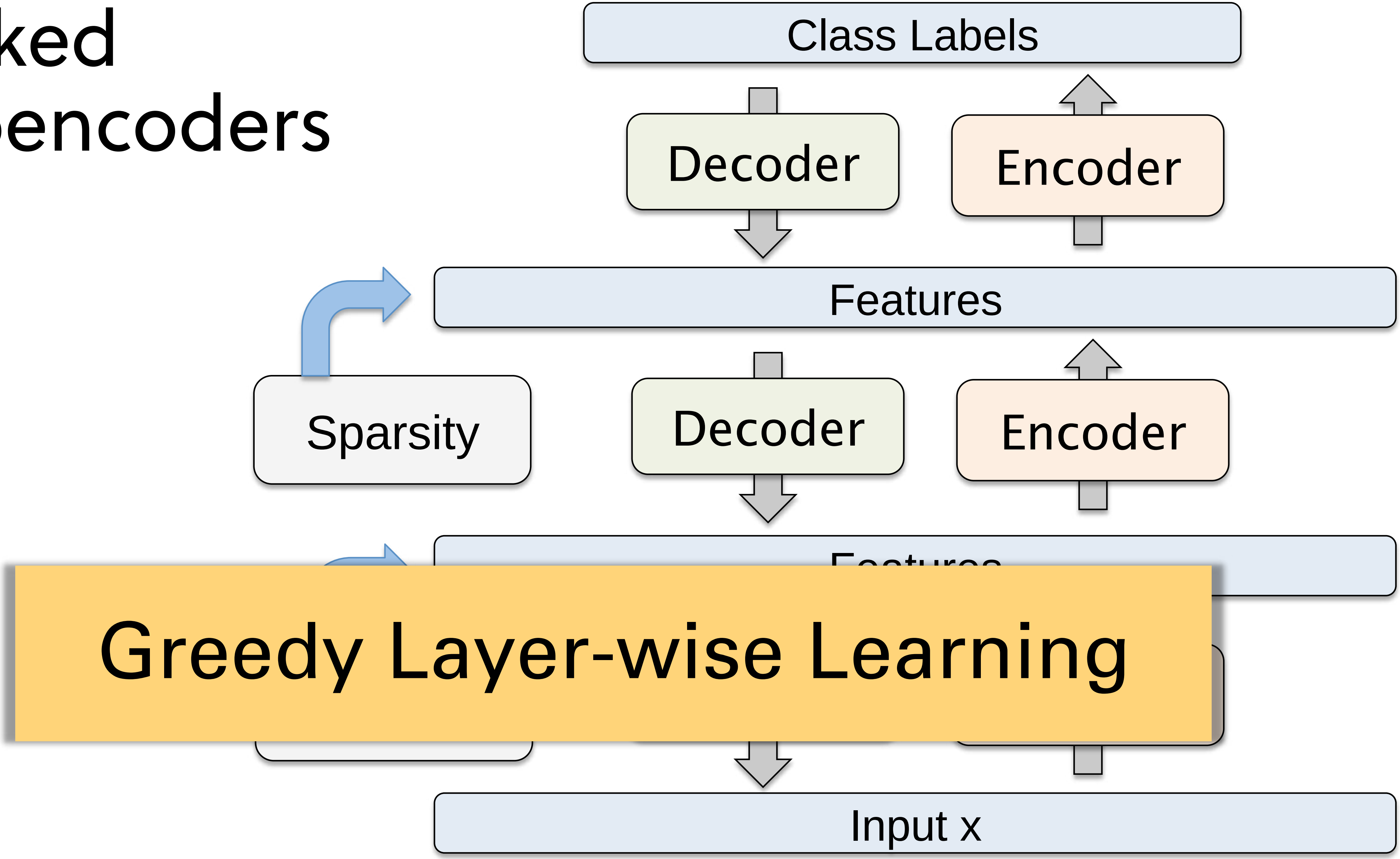
Stacked Autoencoders



Stacked Autoencoders

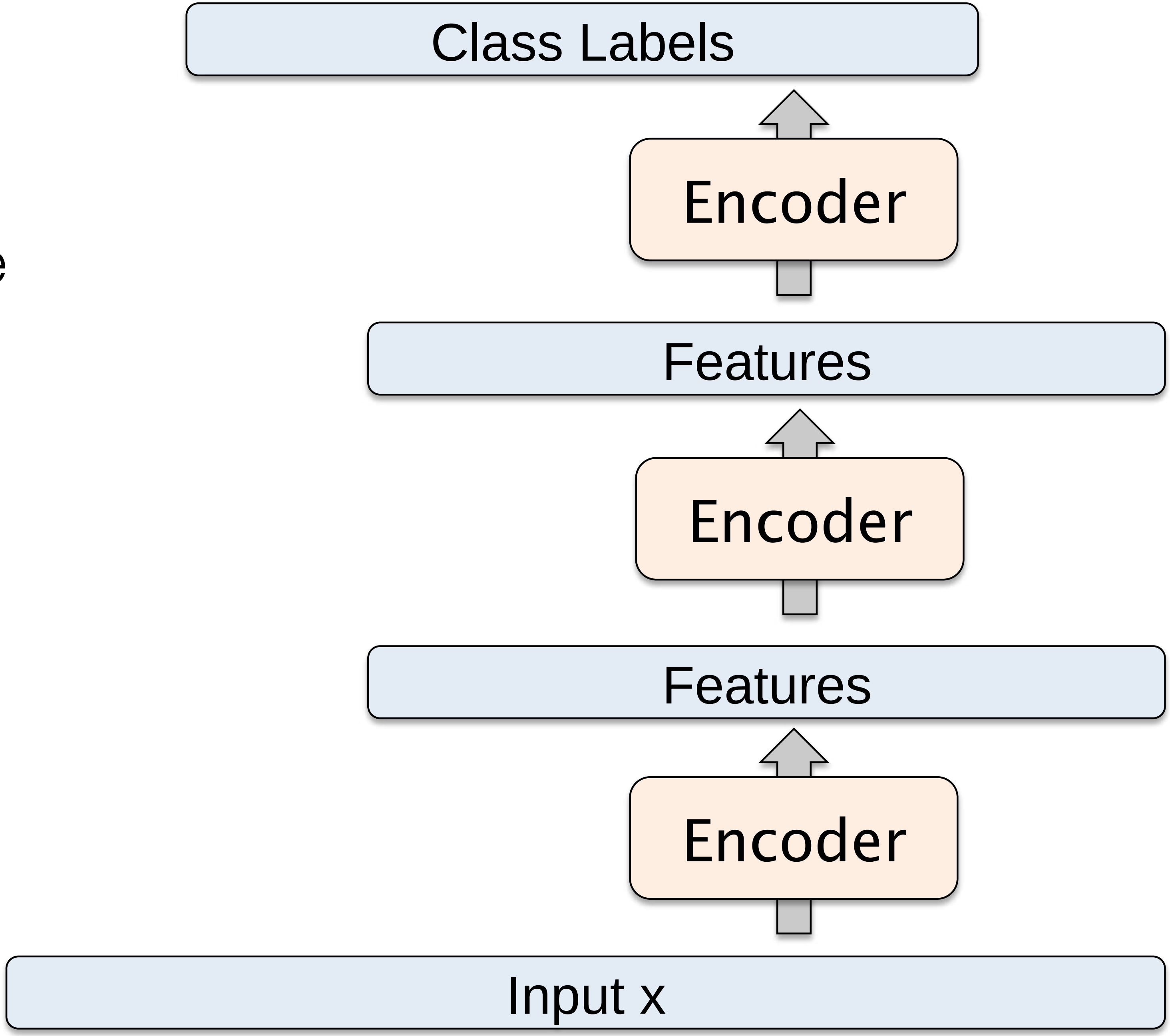


Stacked Autoencoders



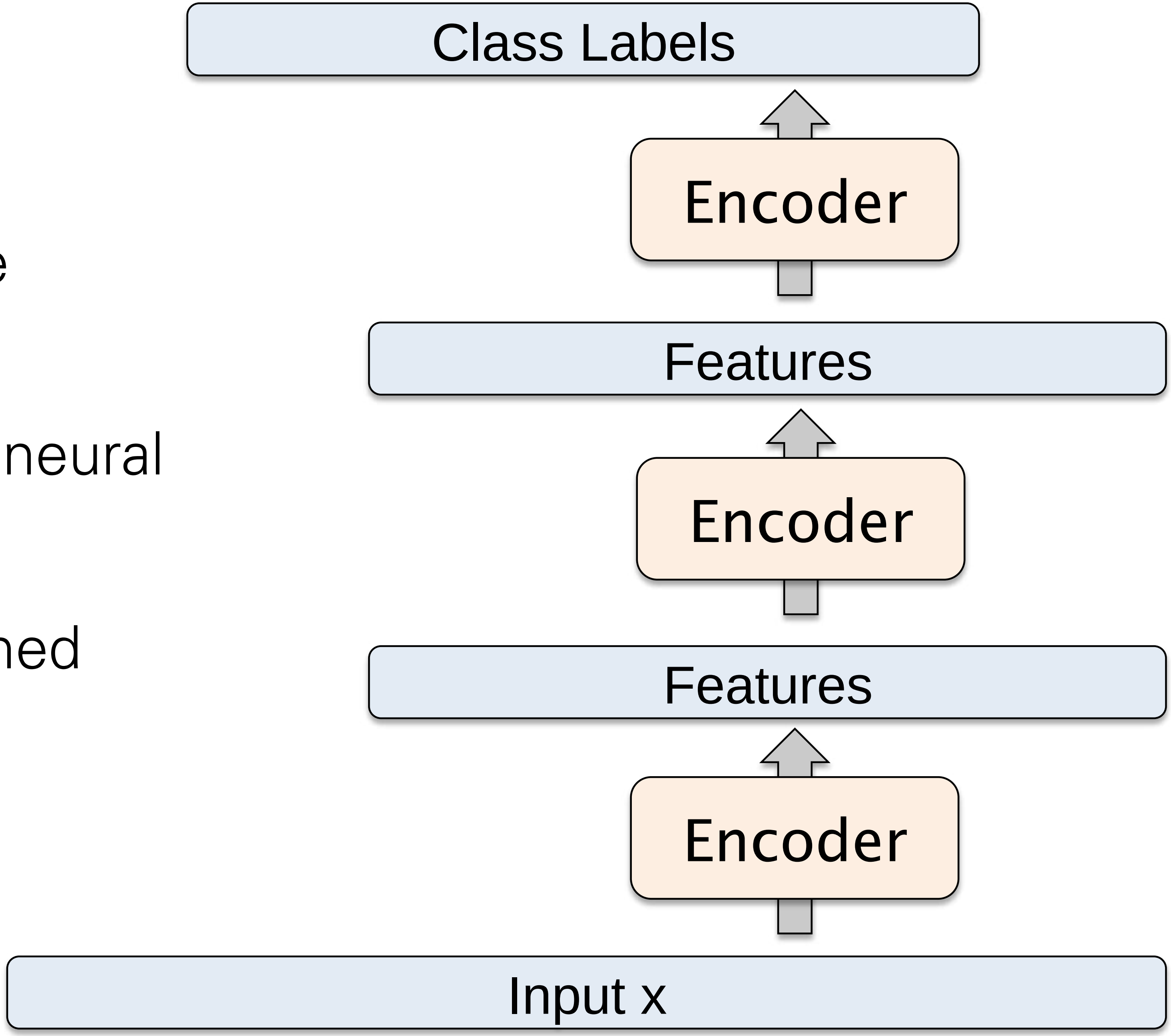
Stacked Autoencoders

- Remove decoders and use feed-forward part.

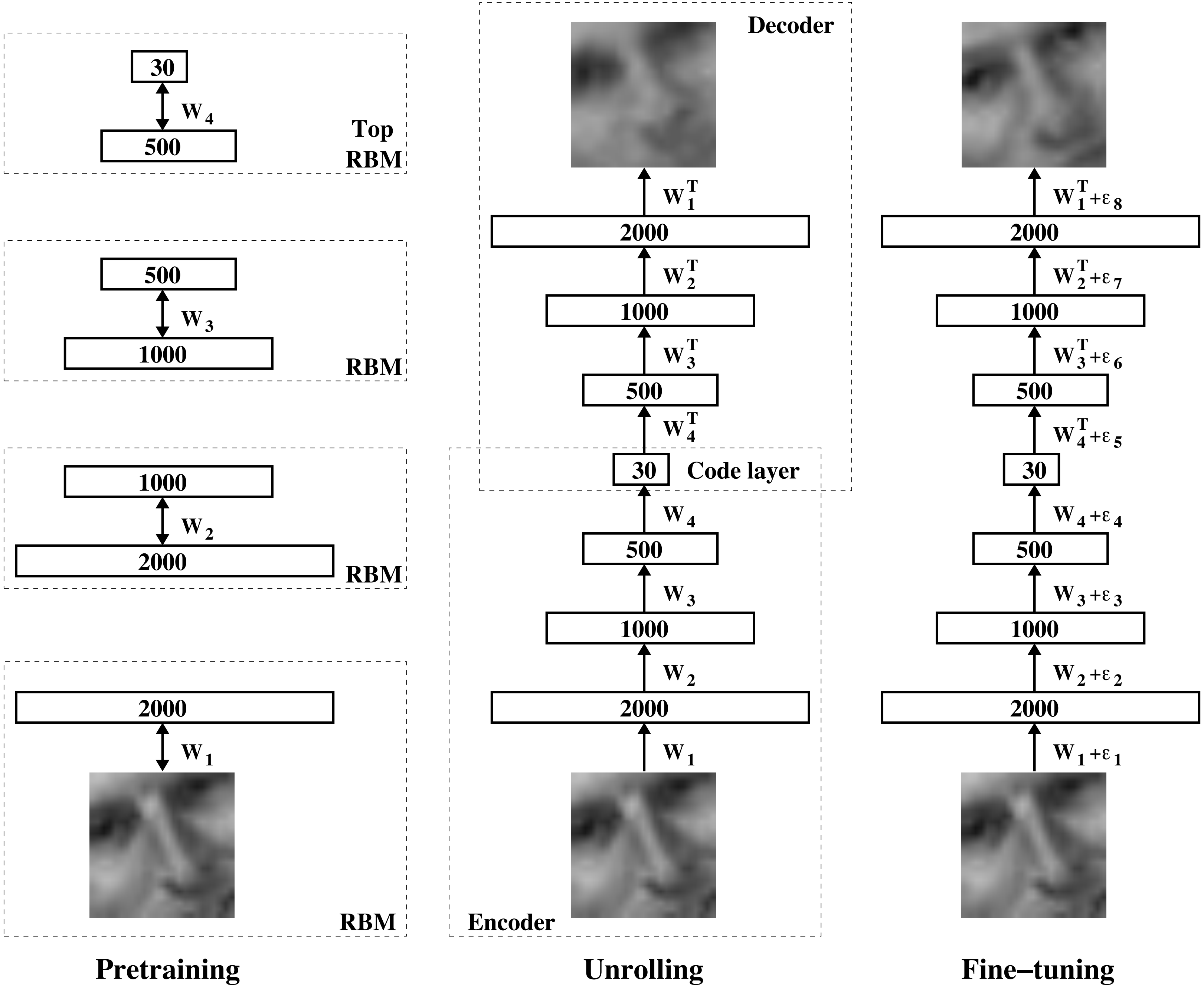


Stacked Autoencoders

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

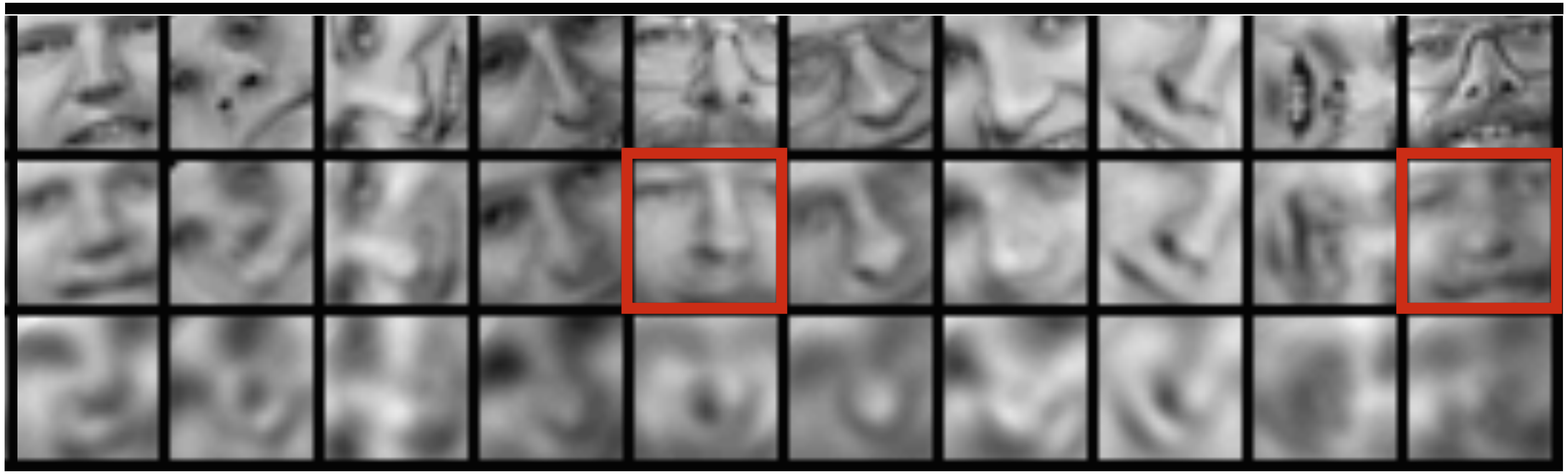


Deep Autoencoder



Deep Autoencoders

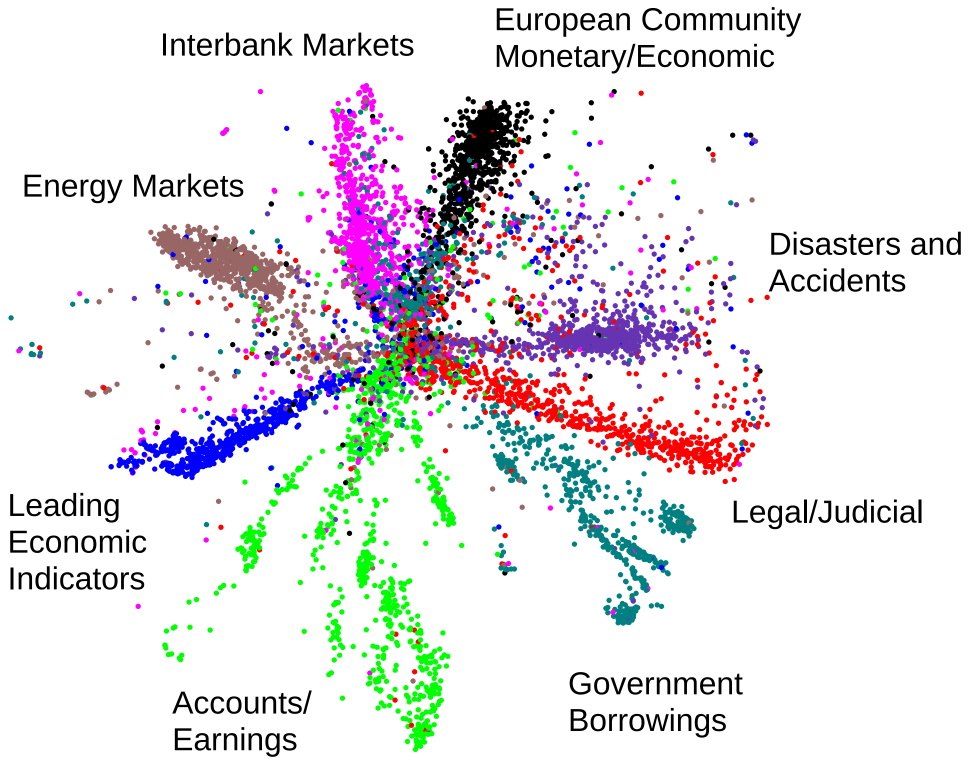
- 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Oliver face patches.

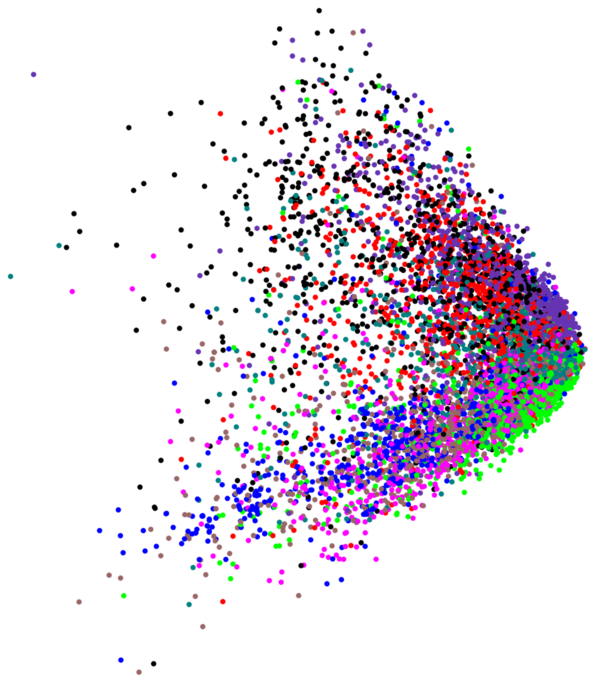


- **Top:** Random samples from the test dataset.
- **Middle:** Reconstructions by the 30-dimensional deep autoencoder.
- **Bottom:** Reconstructions by the 30-dimensional PCA.

(Hinton and Salakhutdinov, Science 2006)

Information Retrieval

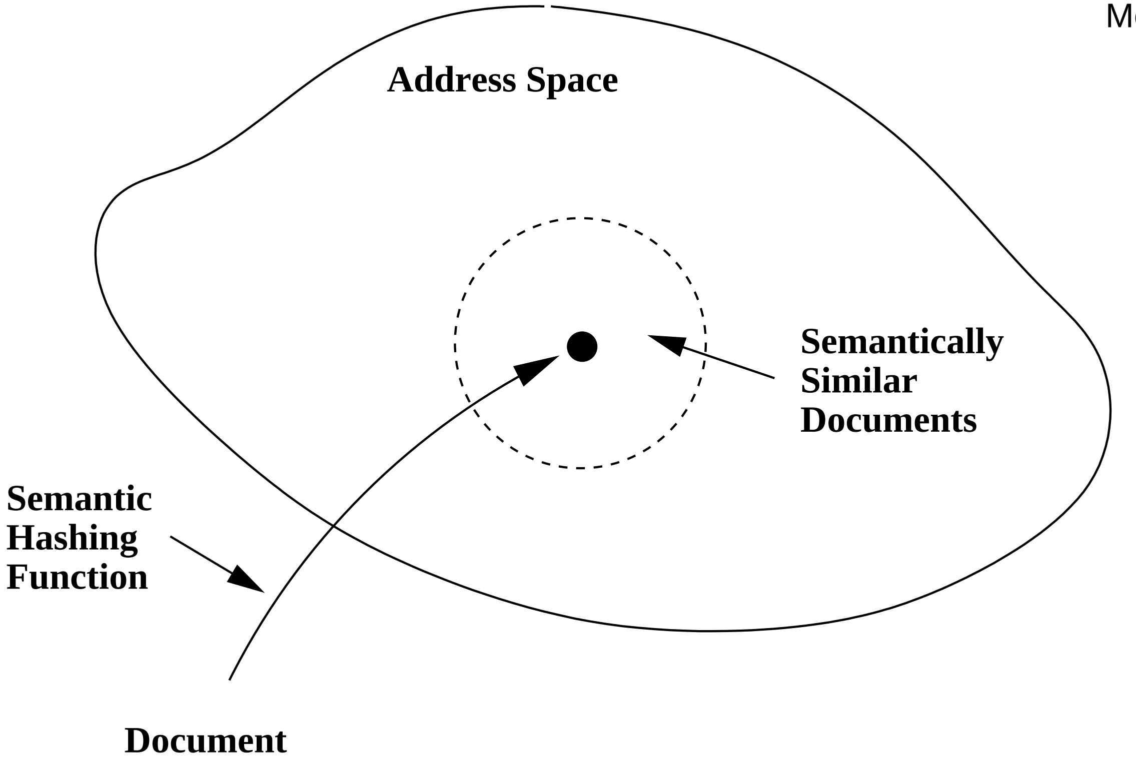


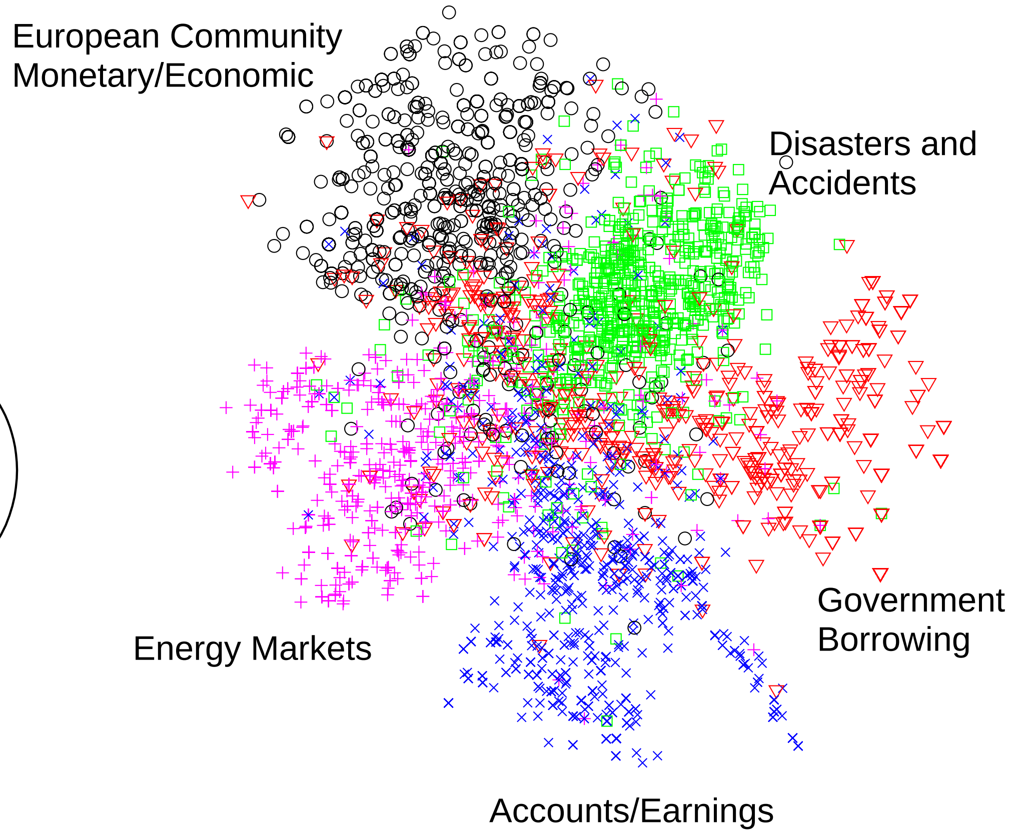


- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test**).
- "Bag-of-words" representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

(Hinton and Salakhutdinov, Science 2006)

Semantic Hashing



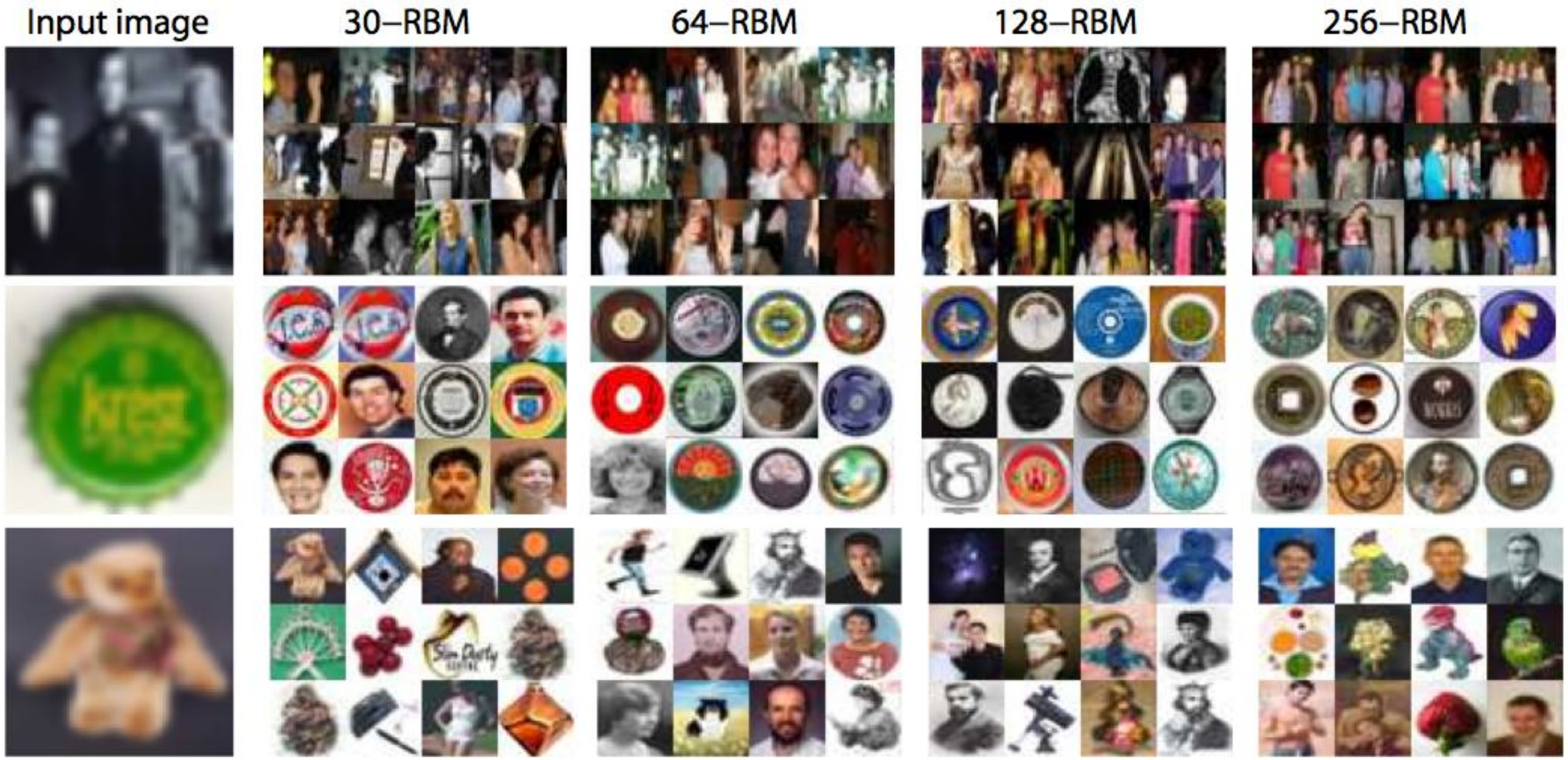


- Learn to map documents into **semantic 20-D binary codes.**
- Retrieve similar documents stored at the nearby addresses **with no search at all.**

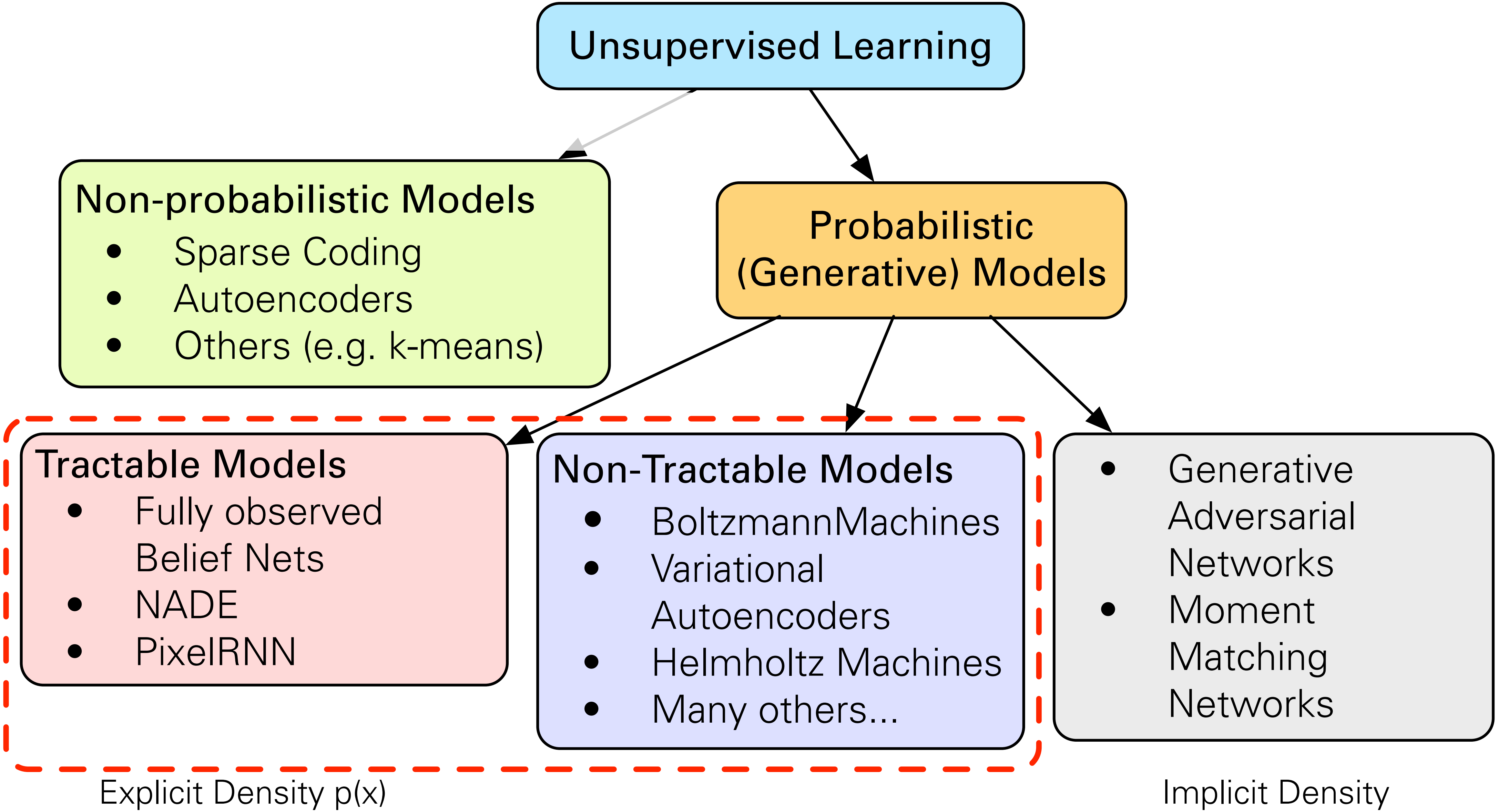
(Hinton and Salakhutdinov, Science 2006)

Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.



- Small Codes, Torralba, Fergus
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 2011
- Norouzi and Fleet, ICML 2011

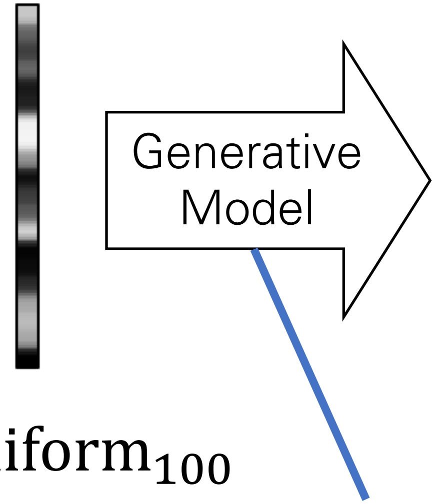

Unsupervised Learning
Non-probabilistic Models
• Sparse Coding
• Autoencoders
• Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
• Fully observed Belief Nets
• NADE
• PixelRNN
Non-Tractable Models
• BoltzmannMachines
• Variational Autoencoders
• Helmholtz Machines
• Many others...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

Generative Adversarial Networks

Genetive Adversarial Networks (GANs)

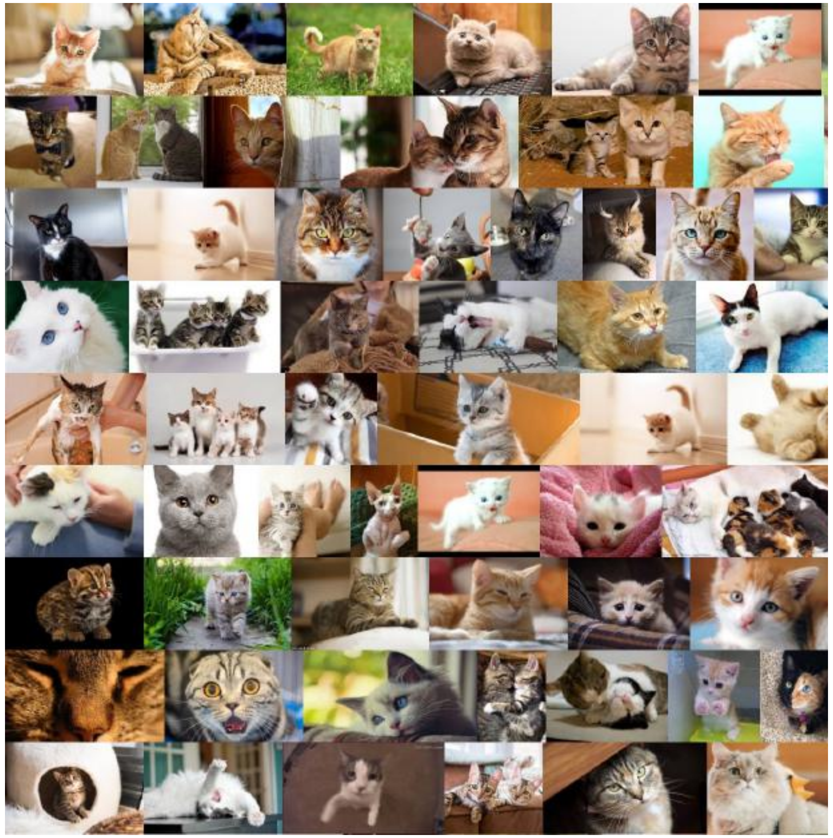
(Goodfellow et al., 2014)

Noise
(random input)



$z \sim \text{Uniform}_{100}$

think of this as
a transformation



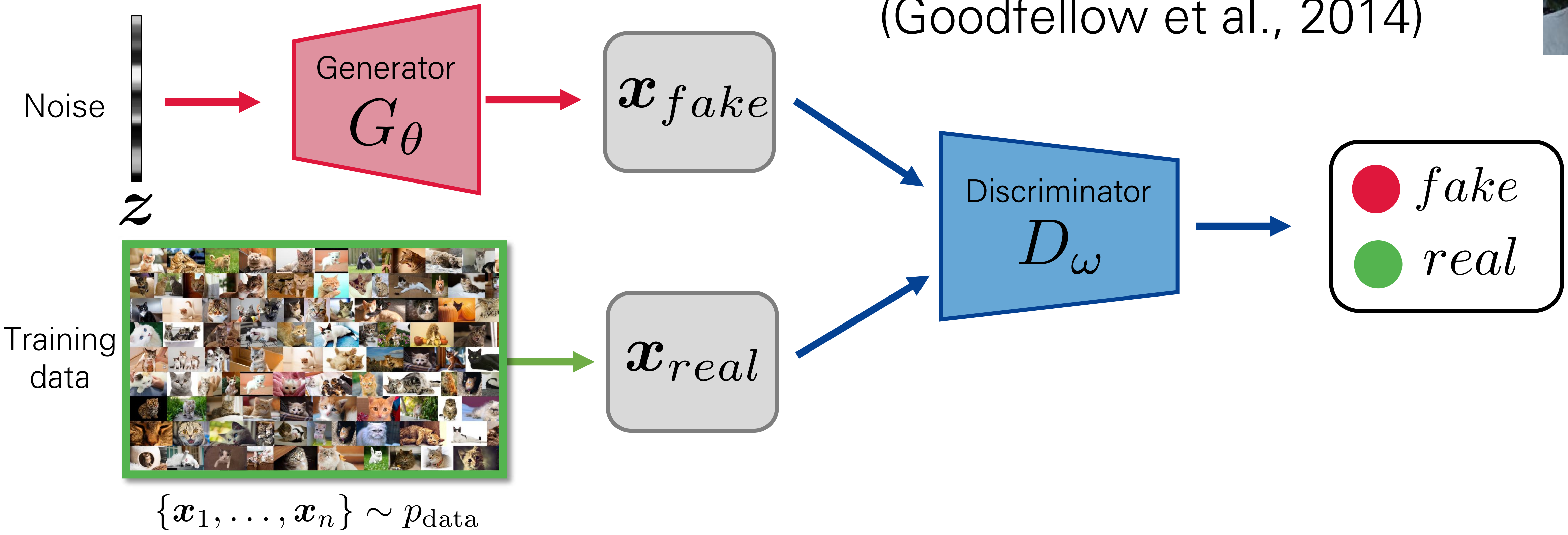
- A game-theoretic likelihood free model

Advantages:

- Uses a latent code
- No Markov chains needed
- Produces the best looking samples

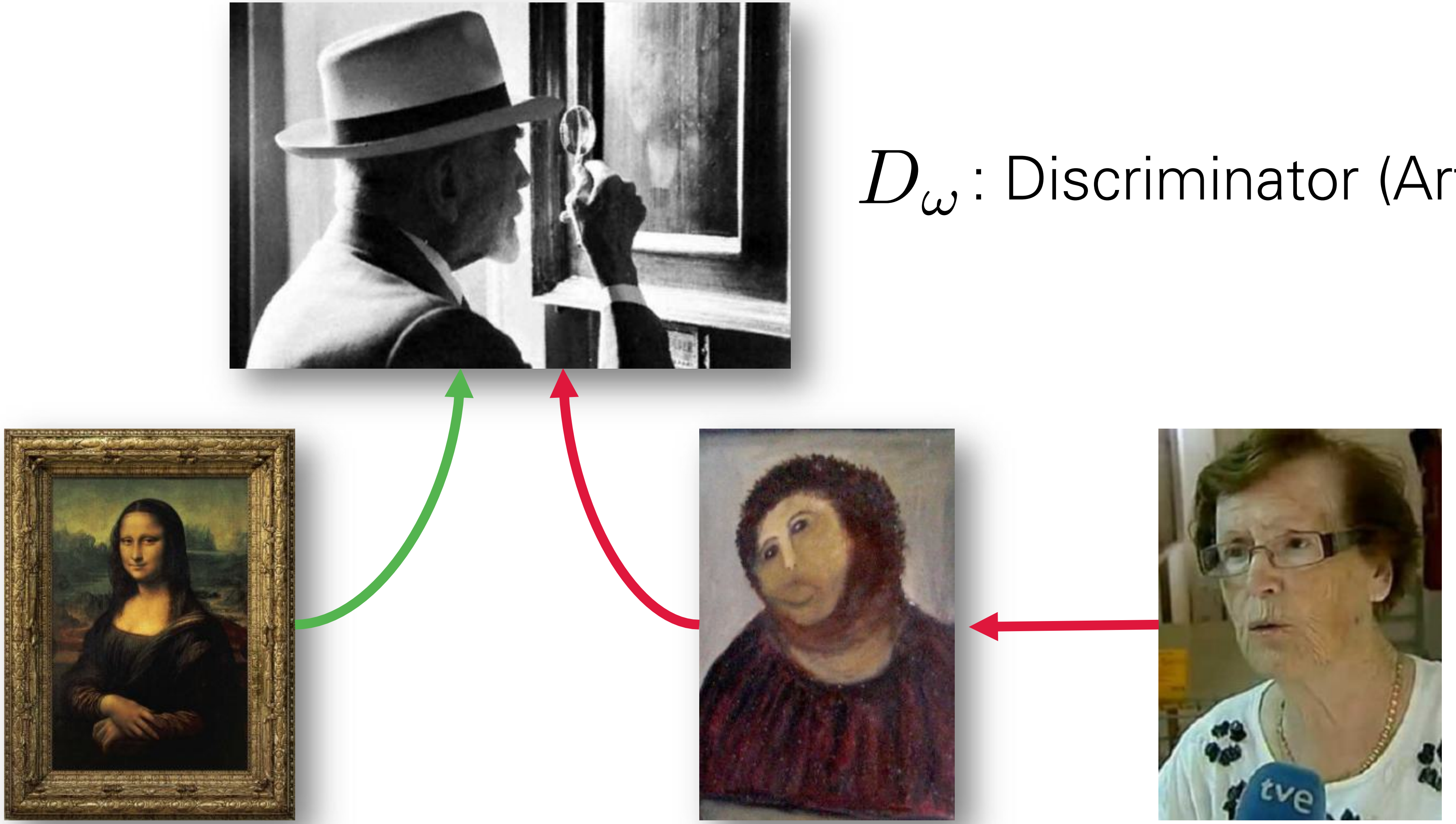
Genetive Adversarial Networks (GANs)

(Goodfellow et al., 2014)



- A game between a generator $G_\theta(\boldsymbol{z})$ and a discriminator $D_\omega(\boldsymbol{x})$
  - 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)

$\boldsymbol{x}_{real}$

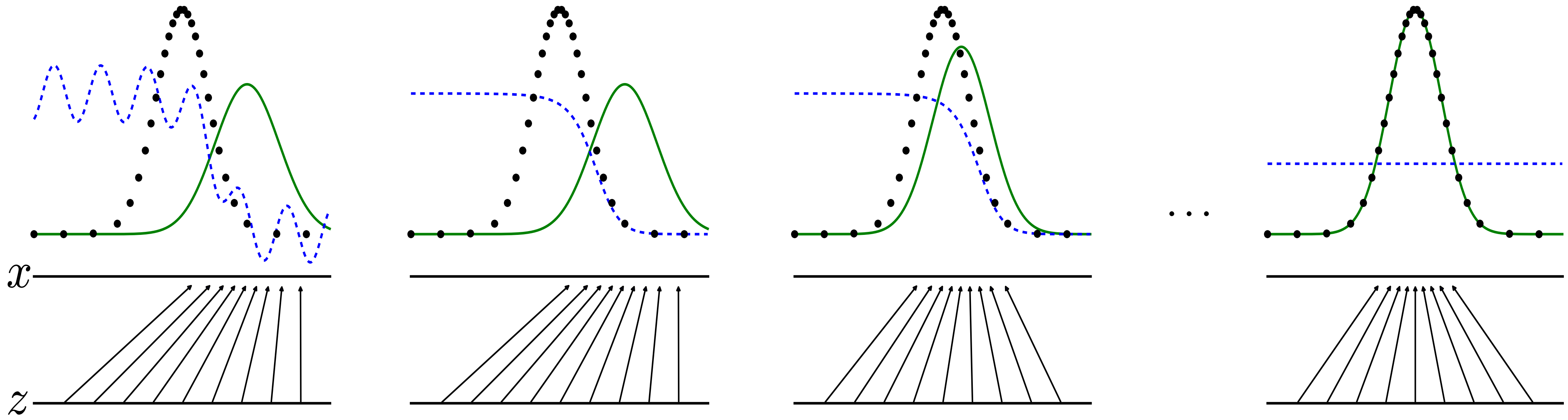
$\boldsymbol{x}_{fake}$

G_θ: Generator (Forger)

Training Procedure

(Goodfellow et al., 2014)

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



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

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

Real data

Noise vector used to generate data

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

Cross-entropy loss for binary classification

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

Generator maximizes the log-probability of the discriminator being mistaken

- Equilibrium of the game
- **Minimizes the Jensen-Shannon divergence between p_{data} and p_x**

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

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

Real data

Noise vector used

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

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

ross-entropy
ss for binary
assification

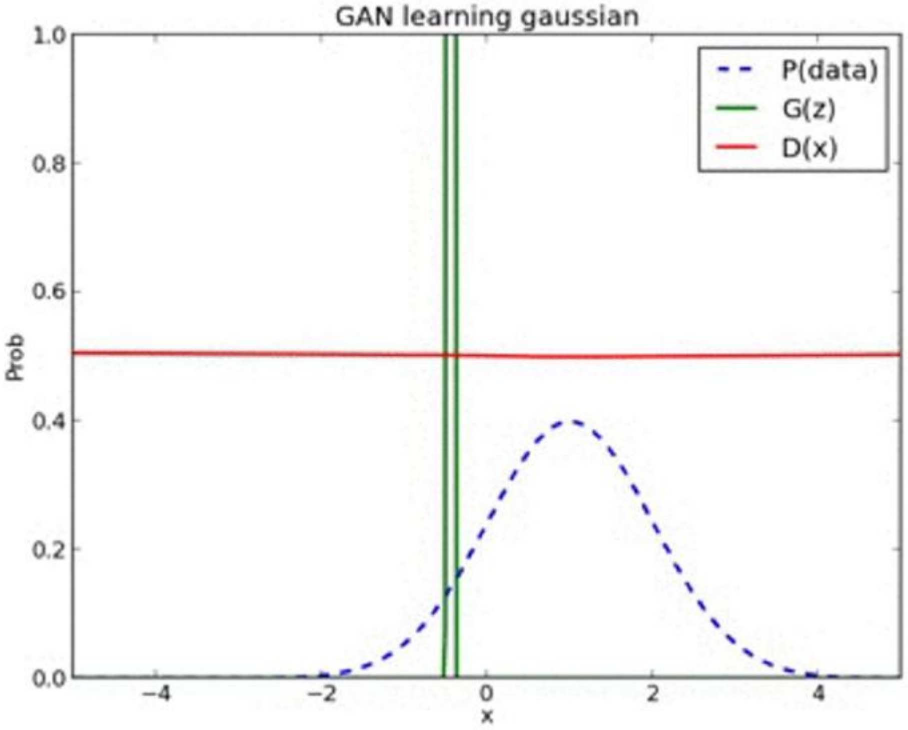
g-probability
of the discriminator being mistaken

Important question is
"Does this converge??"

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

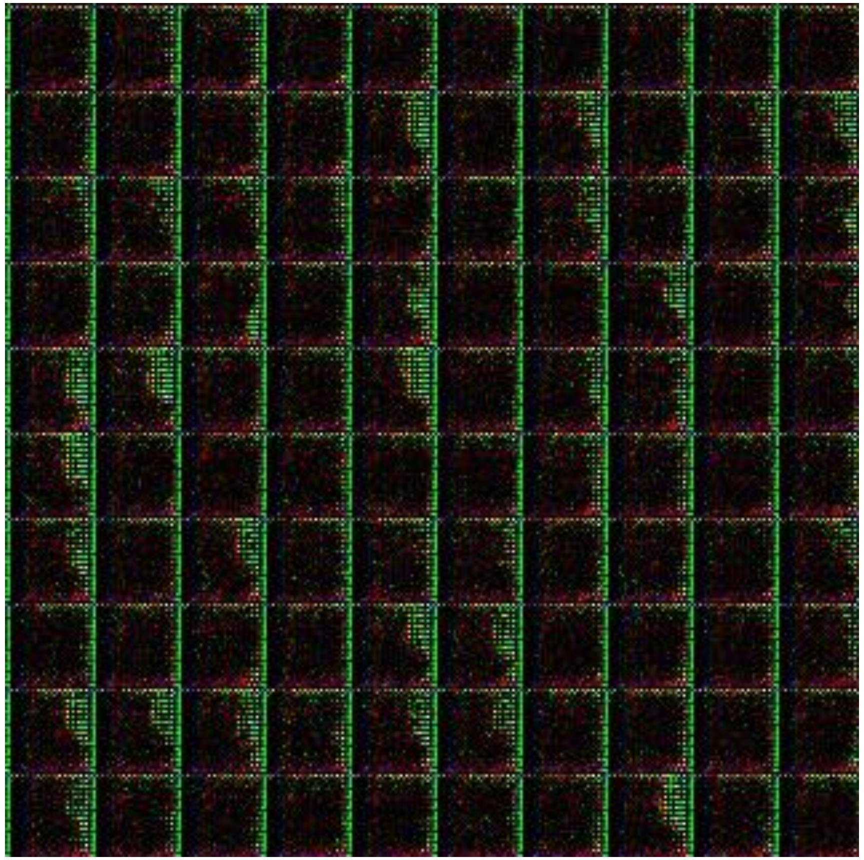
Training Procedure

(Goodfellow et al., 2014)



Source: Alec Radford

Generating 1D points



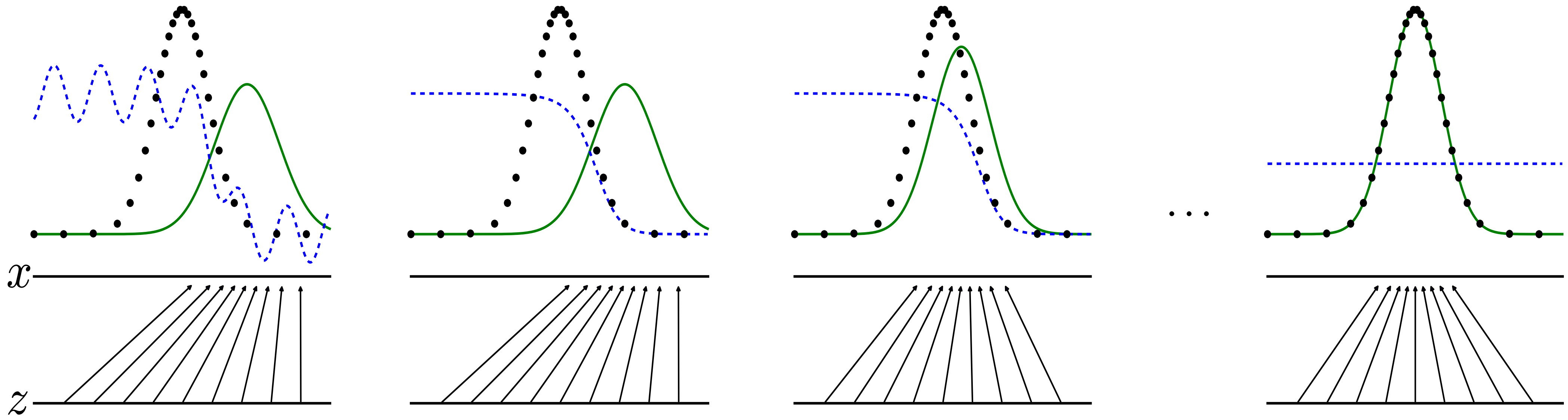
Source: OpenAI blog

Generating images

Training Procedure

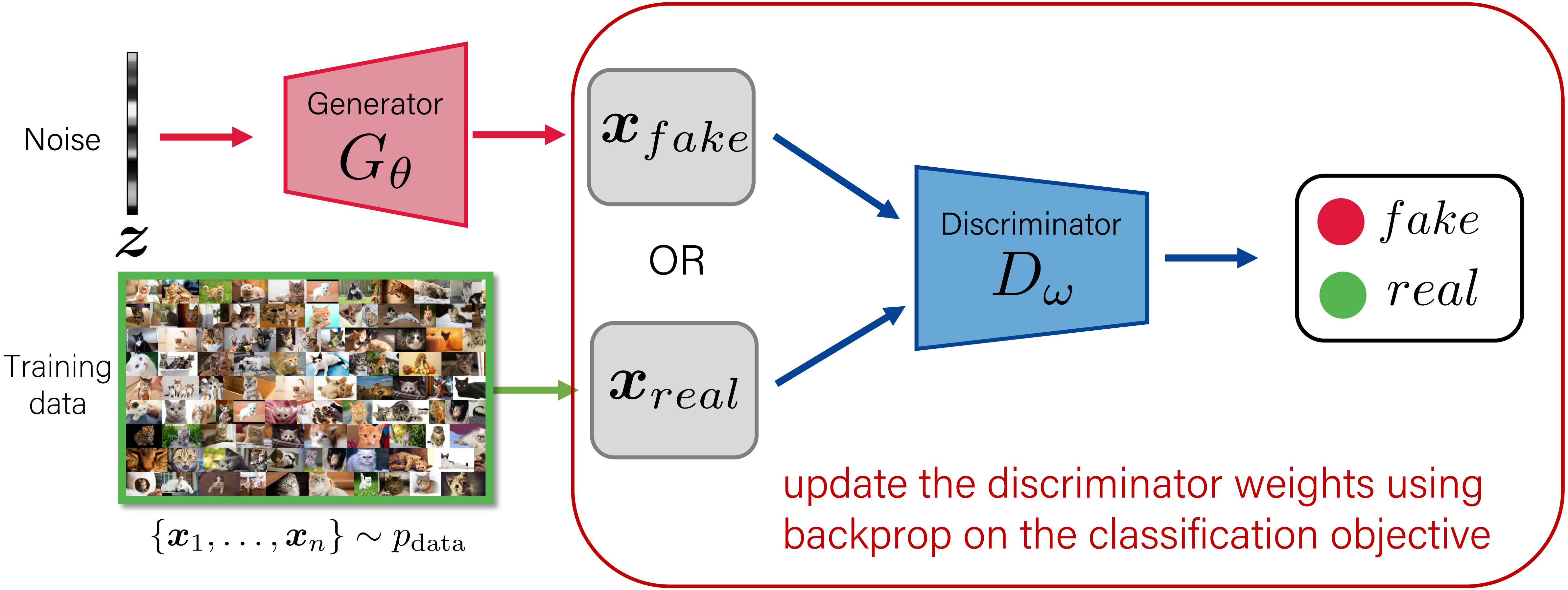
(Goodfellow et al., 2014)

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



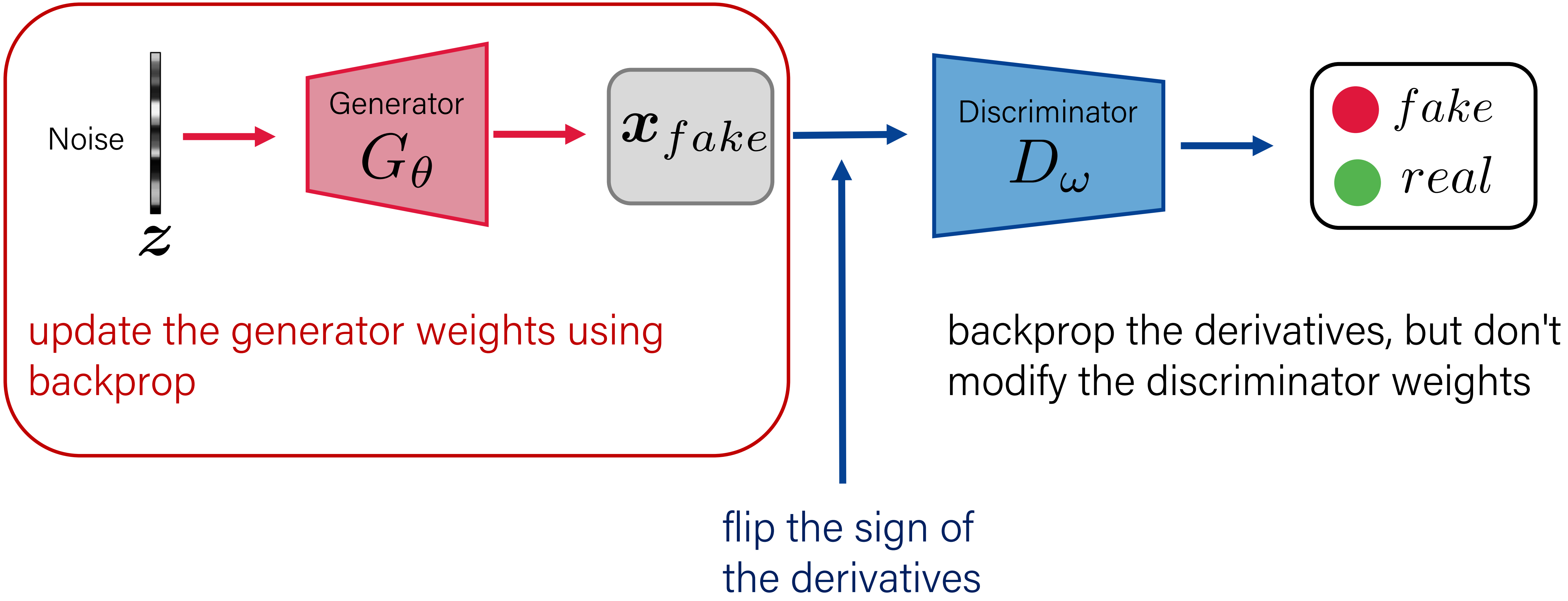
Training Procedure

- Updating the discriminator:



Training Procedure

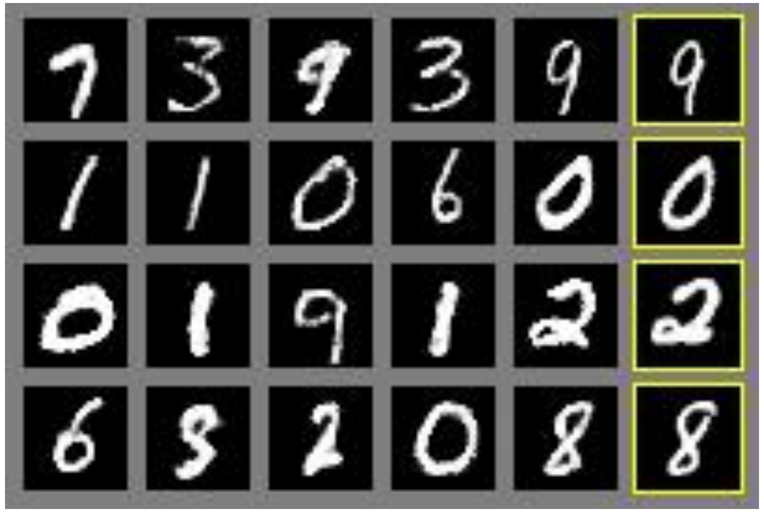
- Updating the generator:



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

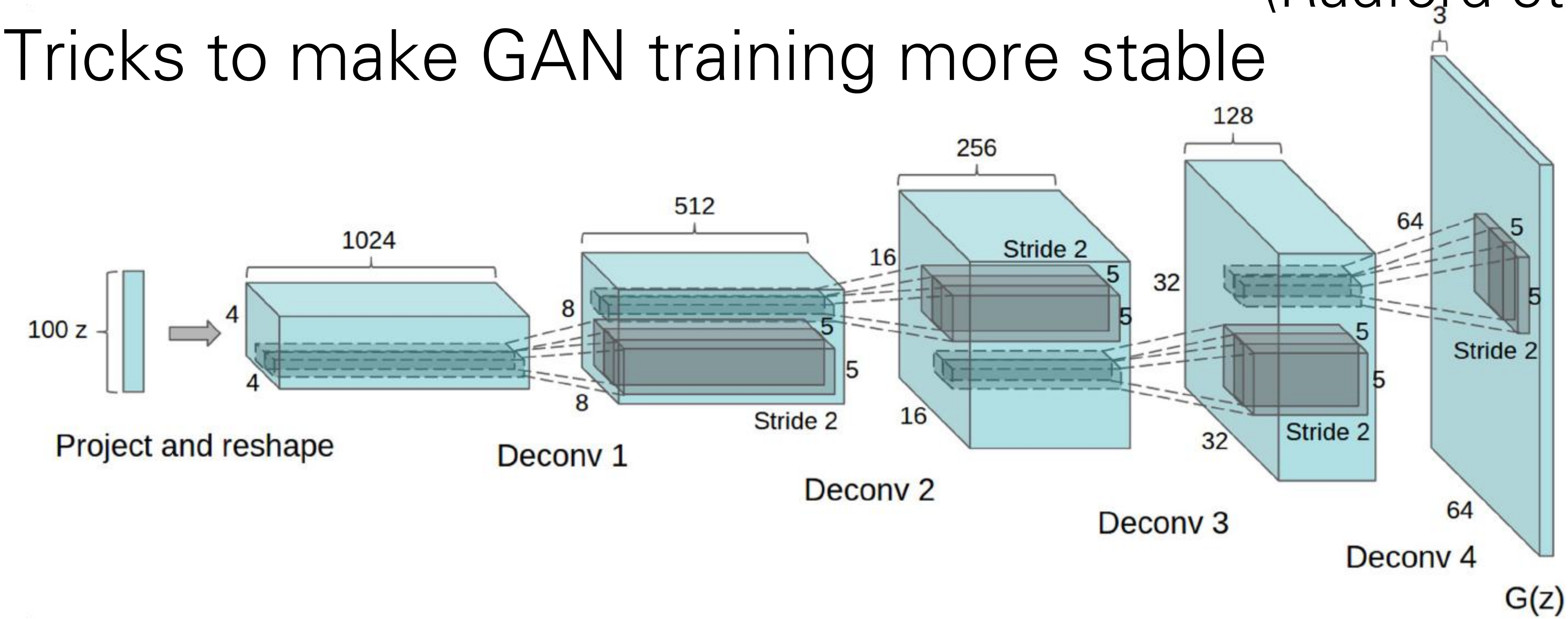
CIFAR10 samples
(fully-connected model)

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

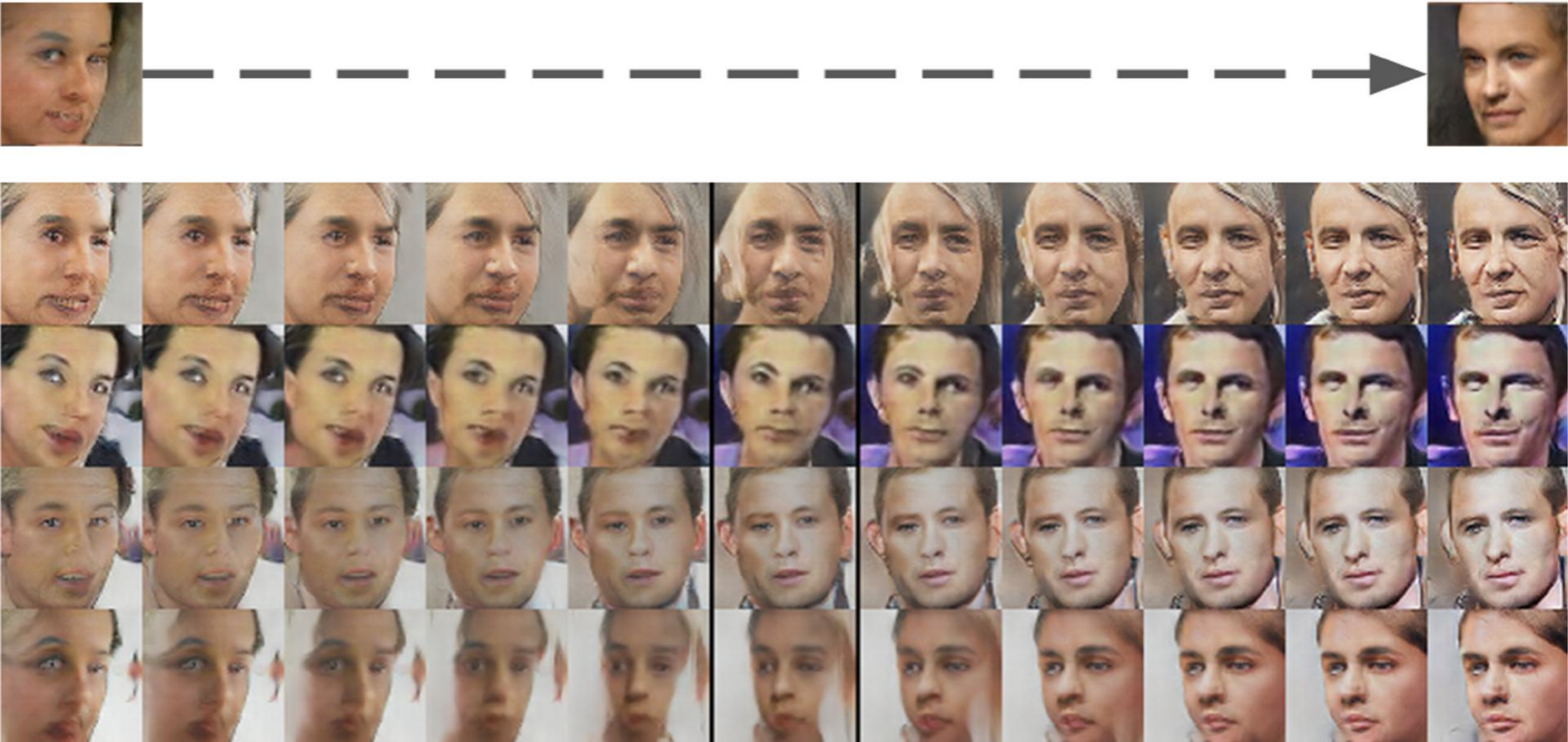
(Radford et al., 2015)

Walking over the latent space

(Radford et al., 2015)

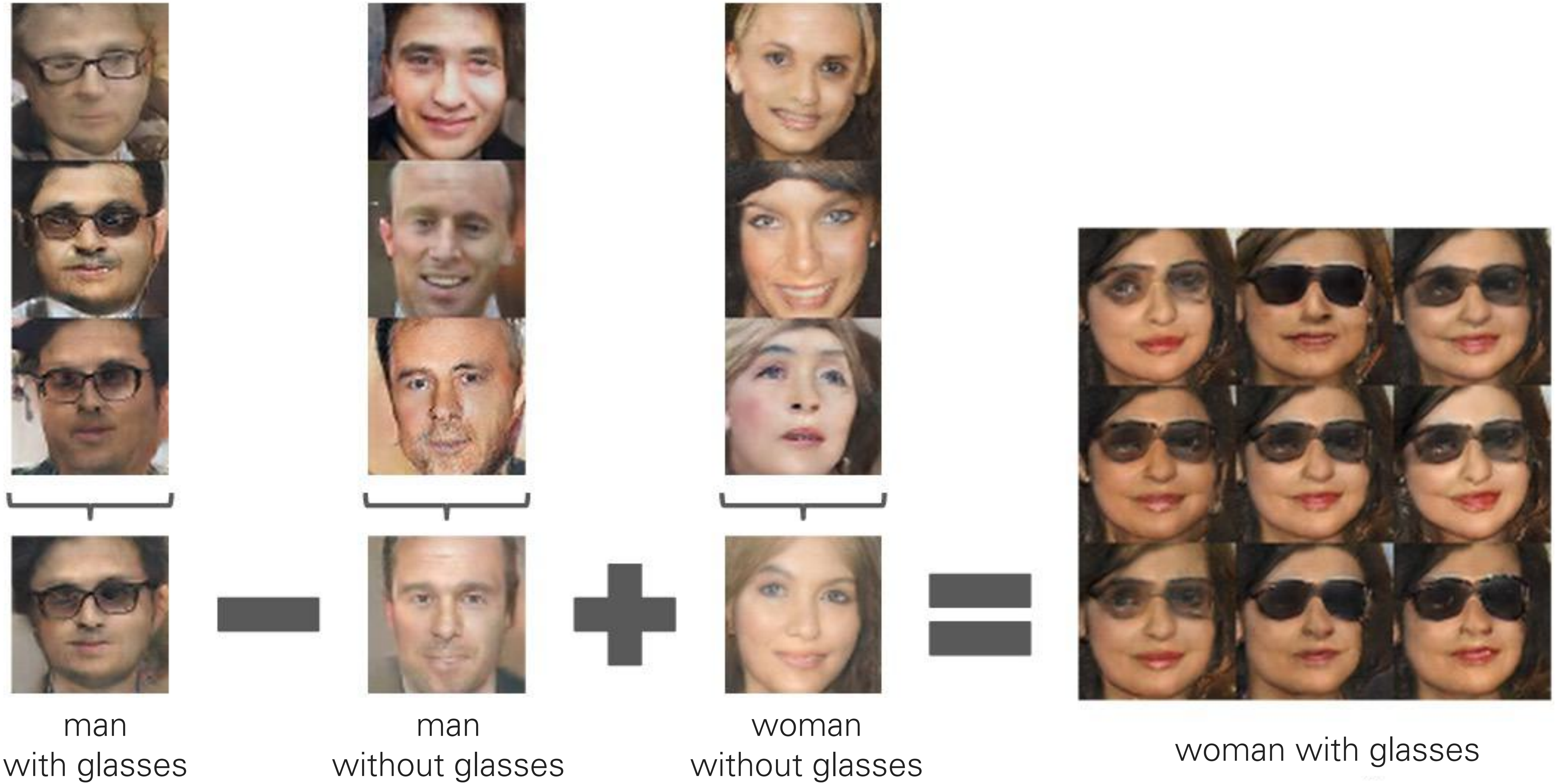
- Interpolation suggests non-overfitting behavior

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



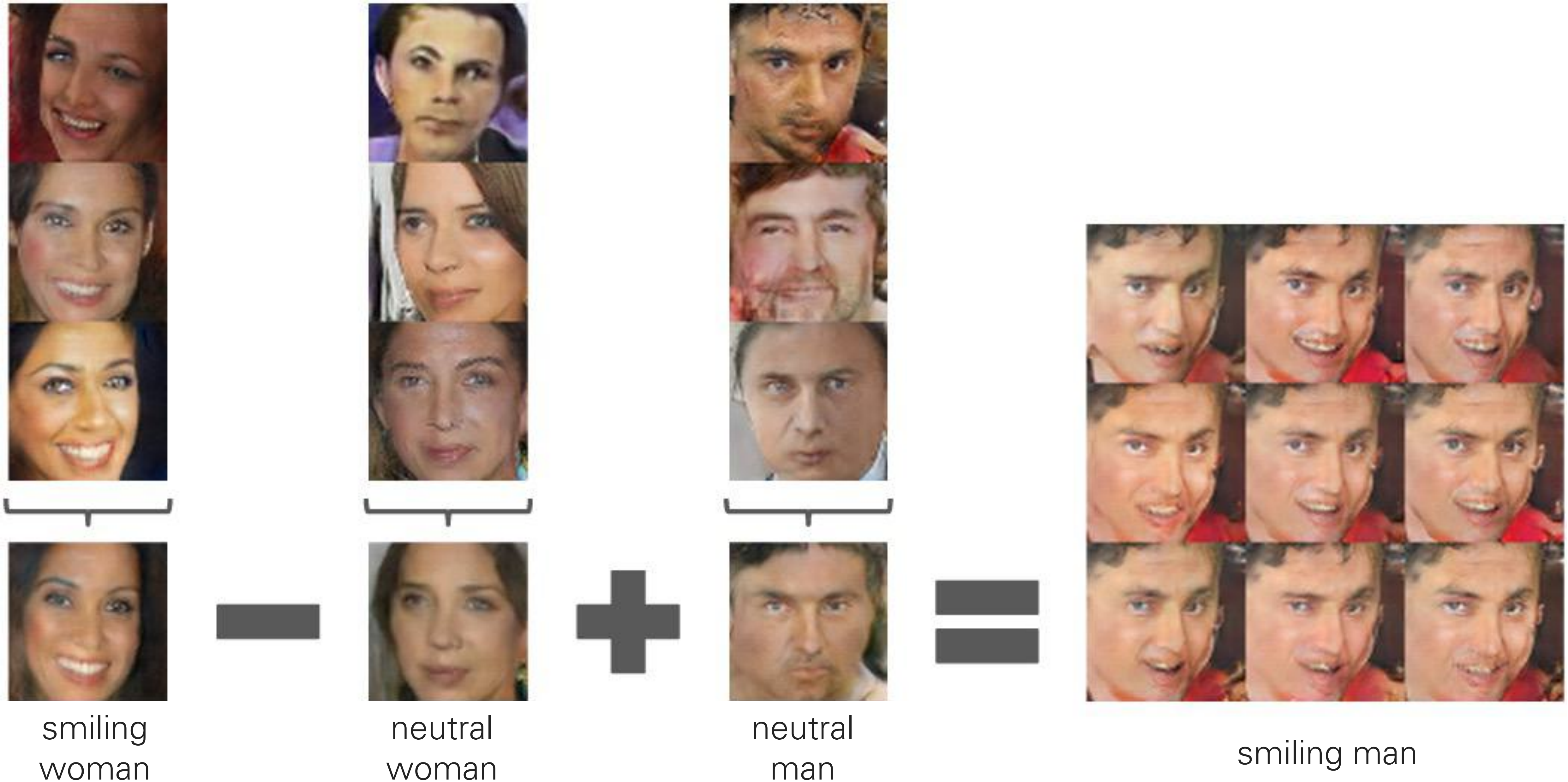
Vector Space Arithmetic

(Radford et al., 2015)

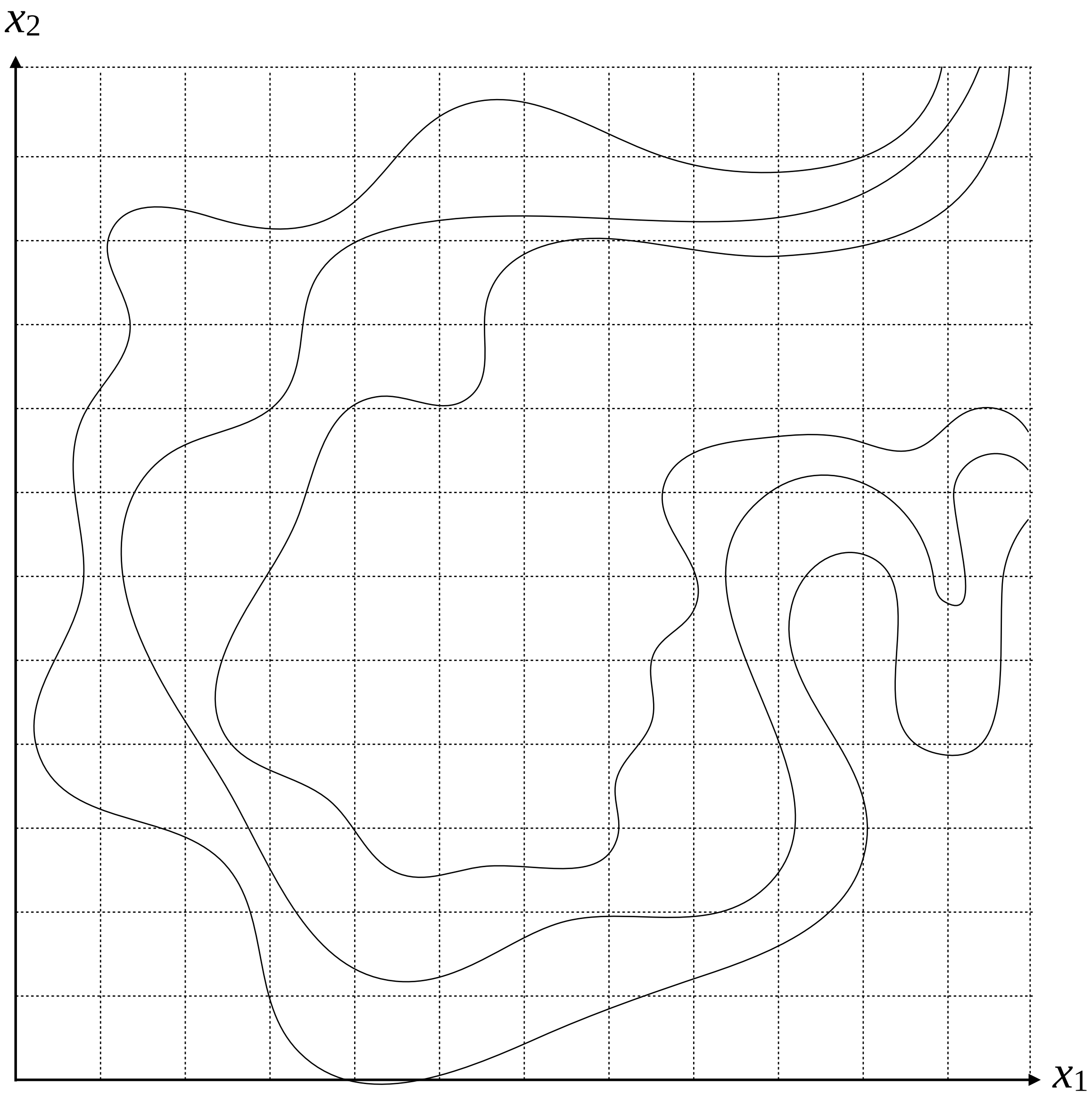


Vector Space Arithmetic

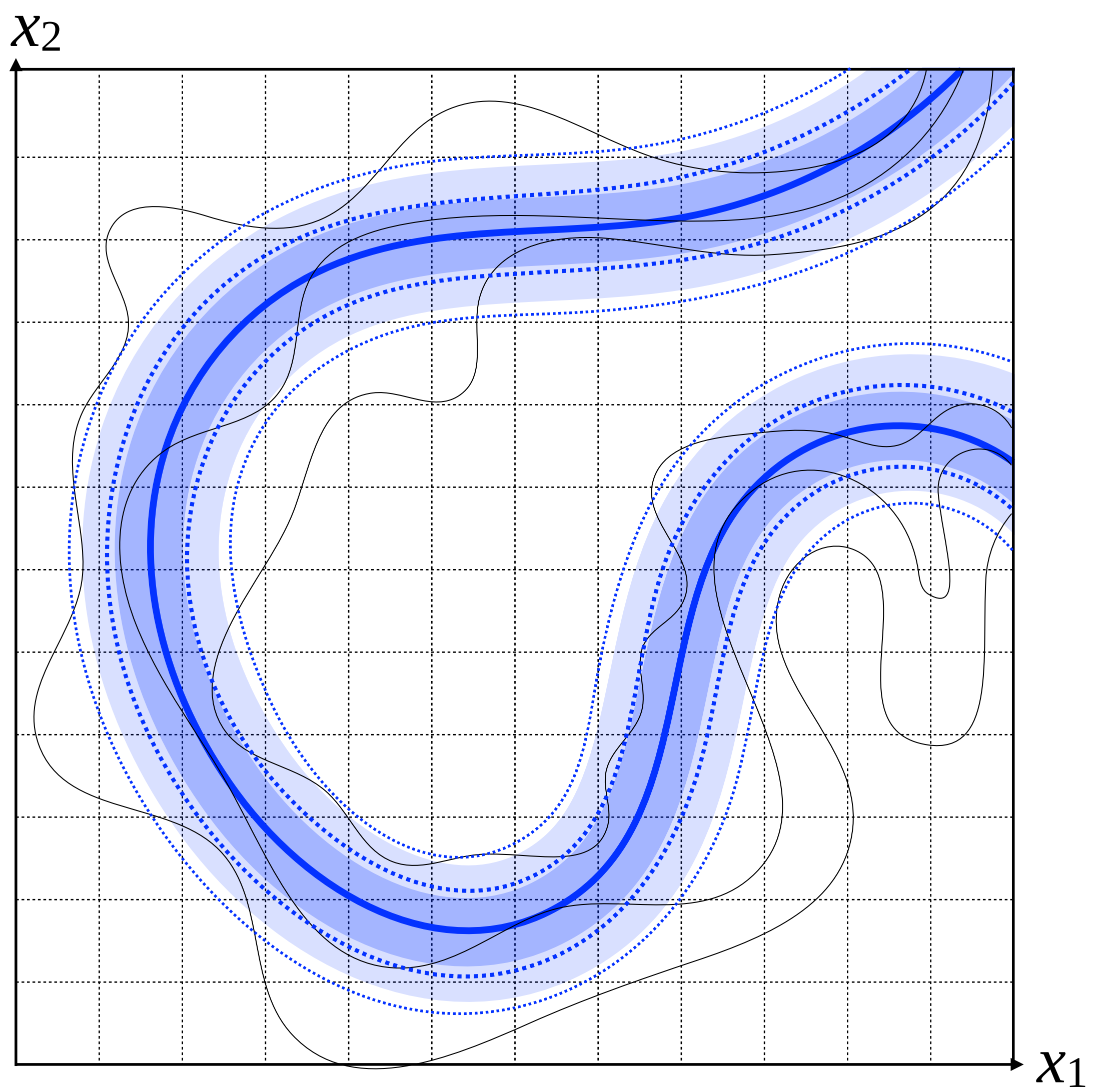
(Radford et al., 2015)



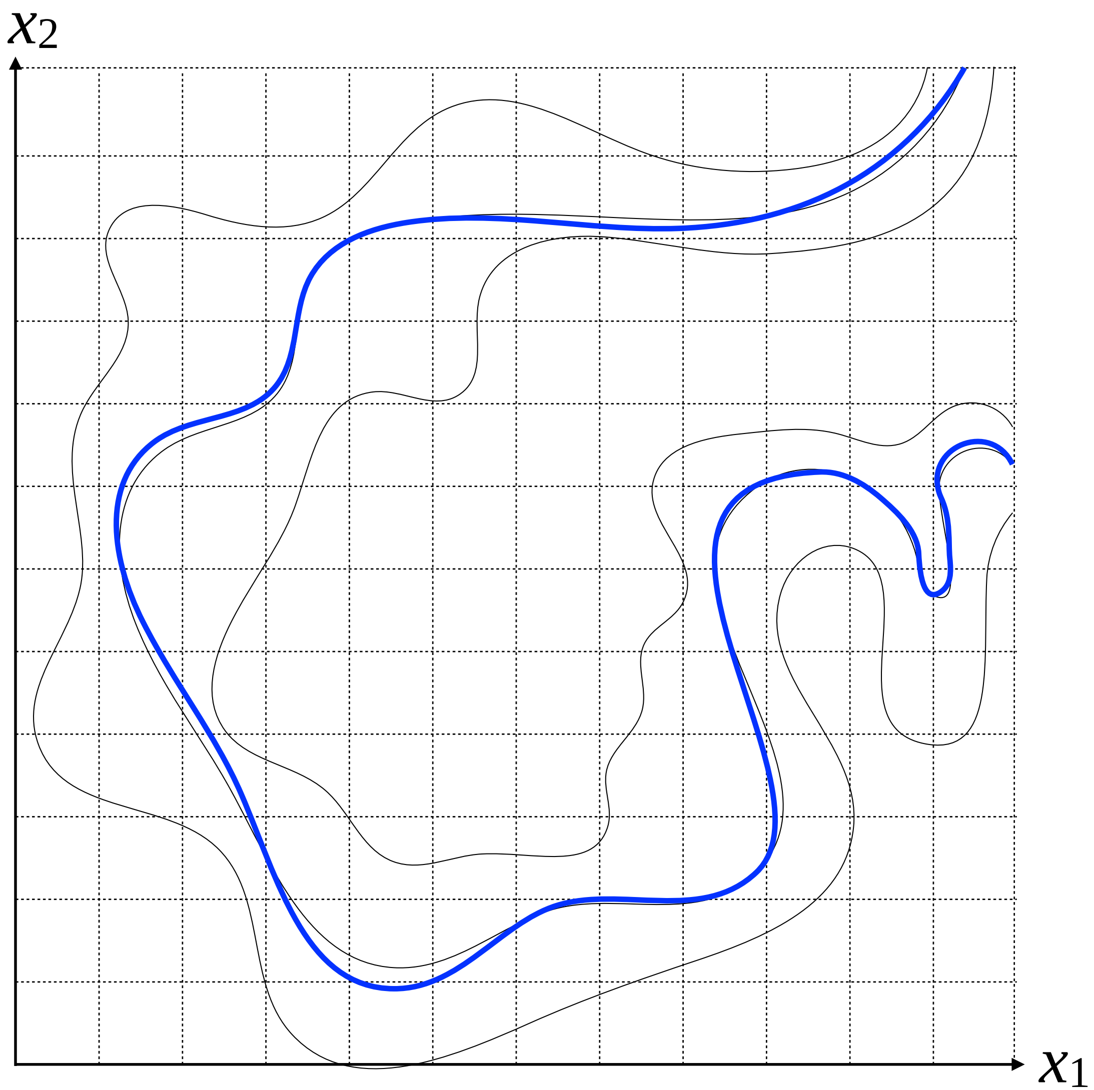
Cartoon of the Image manifold



What makes GANs special?



more traditional max-likelihood approach

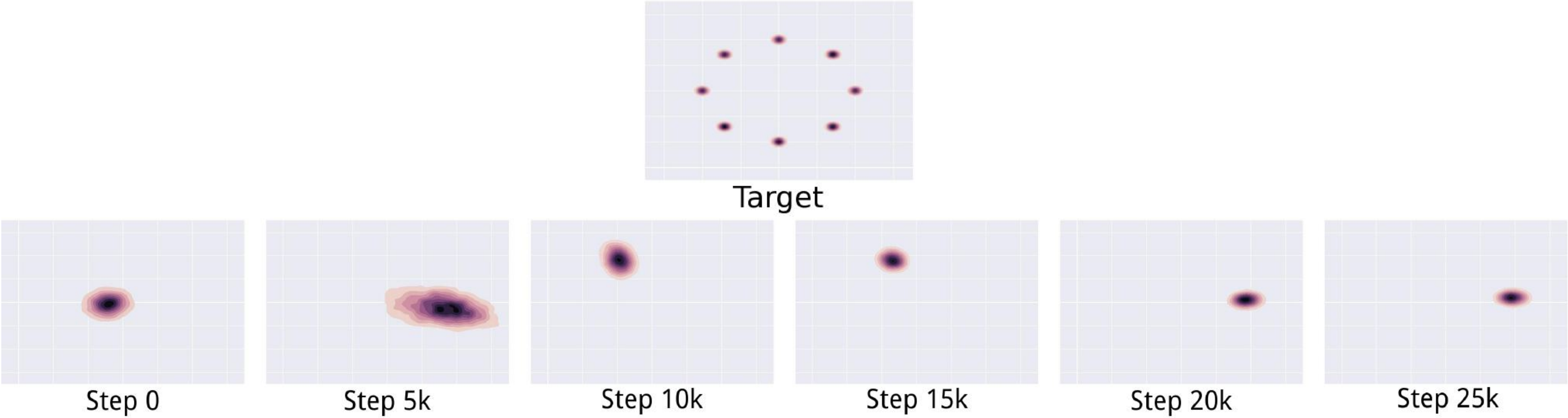


GAN

GAN Failures: Mode Collapse

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

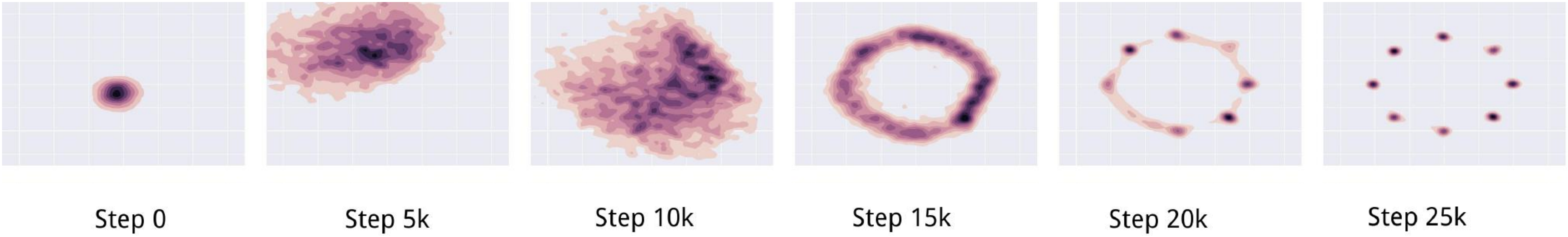
- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point



(Metz et al., 2016)

Mode Collapse: Solutions

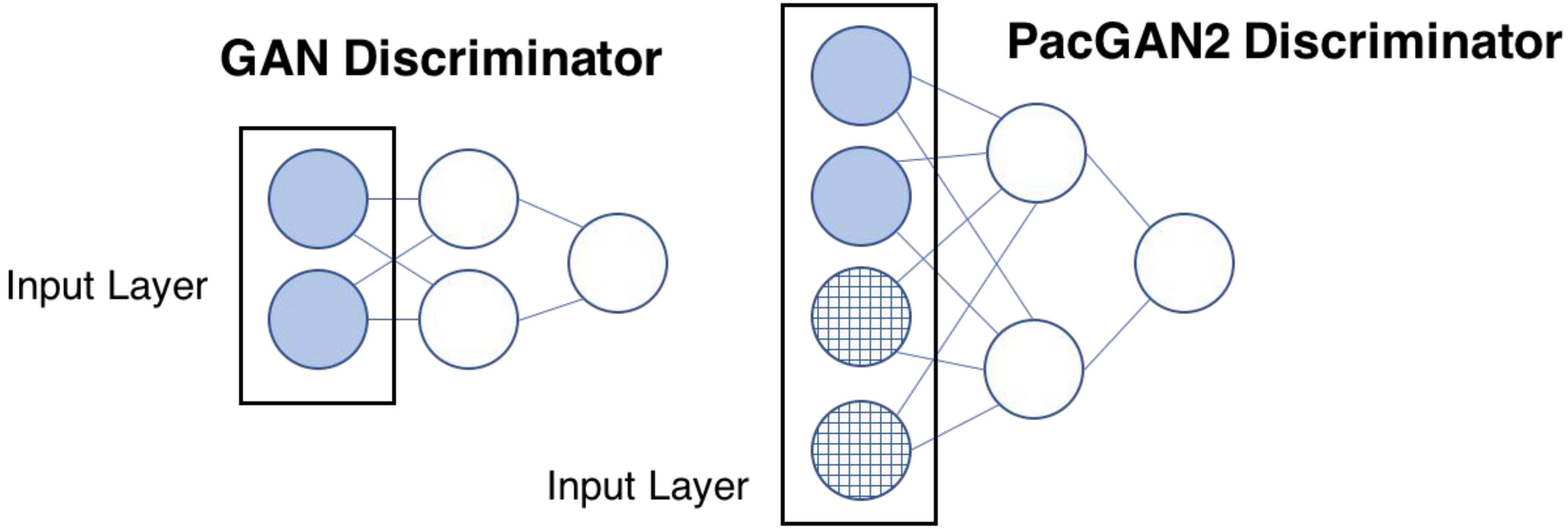
- **Unrolled GANs** (Metz et al 2016): Prevents mode collapse by backproping through a set of (k) updates of the discriminator to update generator parameters



- **VEEGAN** (Srivastava et al 2017): Introduce a reconstructor network which is learned both to map the true data distribution p(x) to a Gaussian and to approximately invert the generator network.

Mode Collapse: Solutions

- **Minibatch Discrimination** (Salimans et al 2016): Add minibatch features that classify each example by comparing it to other members of the minibatch (Salimans et al 2016)
- **PacGAN:** The power of two samples in generative adversarial networks (Lin et al 2017): Also uses multisample discrimination.



Mode Collapse: Solutions

- **PacGAN:** The power of two samples in generative adversarial networks (Lin et al 2017): Also uses multisample discrimination.

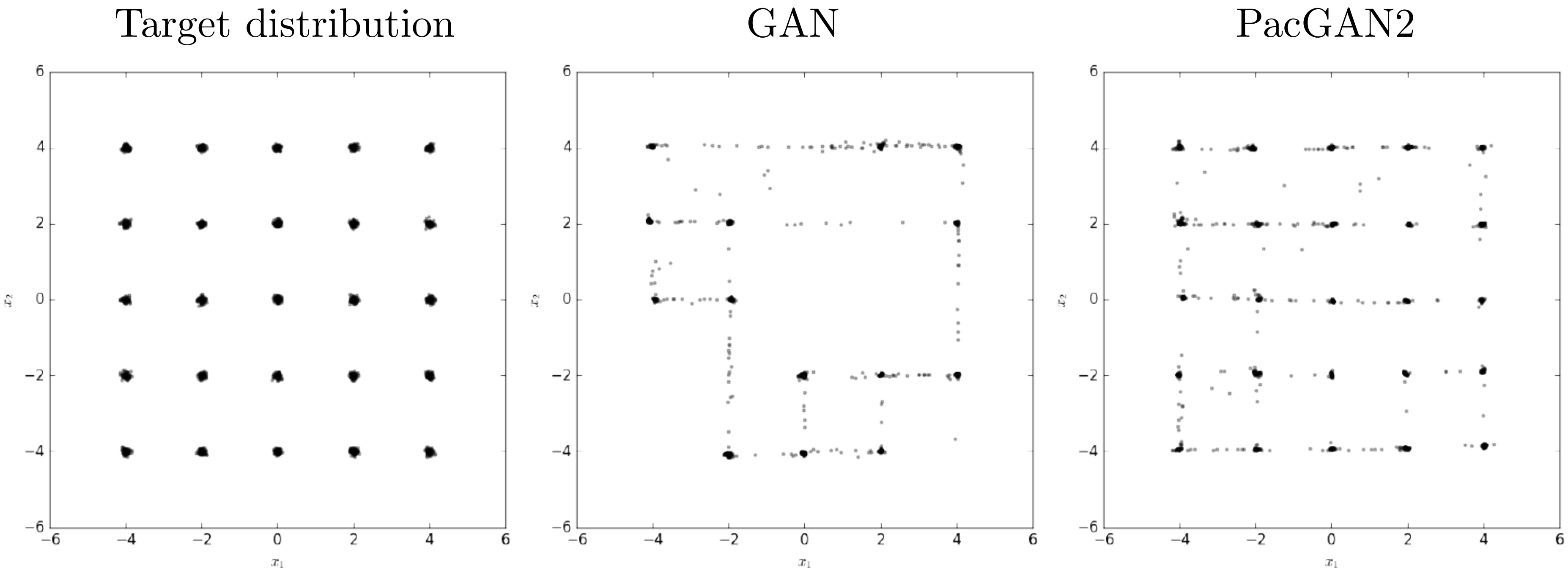


Figure 2: Scatter plot of the 2D samples from the true distribution (left) of 2D-grid and the learned generators using GAN (middle) and PacGAN2 (right). PacGAN2 captures all of the 25 modes.

GAN Evaluation

- Quantitatively evaluating GANs is not straightforward:
  - Max Likelihood is a poor indication of sample quality

- Some evaluation metrics
  - **Inception Score (IS)**:
    y = labels given gen. image. p(y|x) is from classifier - InceptionNet

$$\mathrm{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[KL(p_{\mathcal{M}}(y|\mathbf{x})||p_{\mathcal{M}}(y))]}$$

  - **Fréchet inception distance (FID):** (Currently most popular)
    Estimate mean *m* and covariance *C* from classifier output - InceptionNet

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\big(\boldsymbol{C} + \boldsymbol{C}_w - 2\big(\boldsymbol{C}\boldsymbol{C}_w\big)^{1/2}\big)$$

  - **Kernel MMD** (Maximum Mean Discrepancy):

$$\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left(\mathbb{E}_{\substack{\mathbf{x}_r,\mathbf{x}_r'\sim\mathbb{P}_r,\\ \mathbf{x}_g,\mathbf{x}_g'\sim\mathbb{P}_g}}\left[k(\mathbf{x}_r, \mathbf{x}_r') - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}_g')\right]\right)^{\frac{1}{2}}$$

Subclasses of GANs

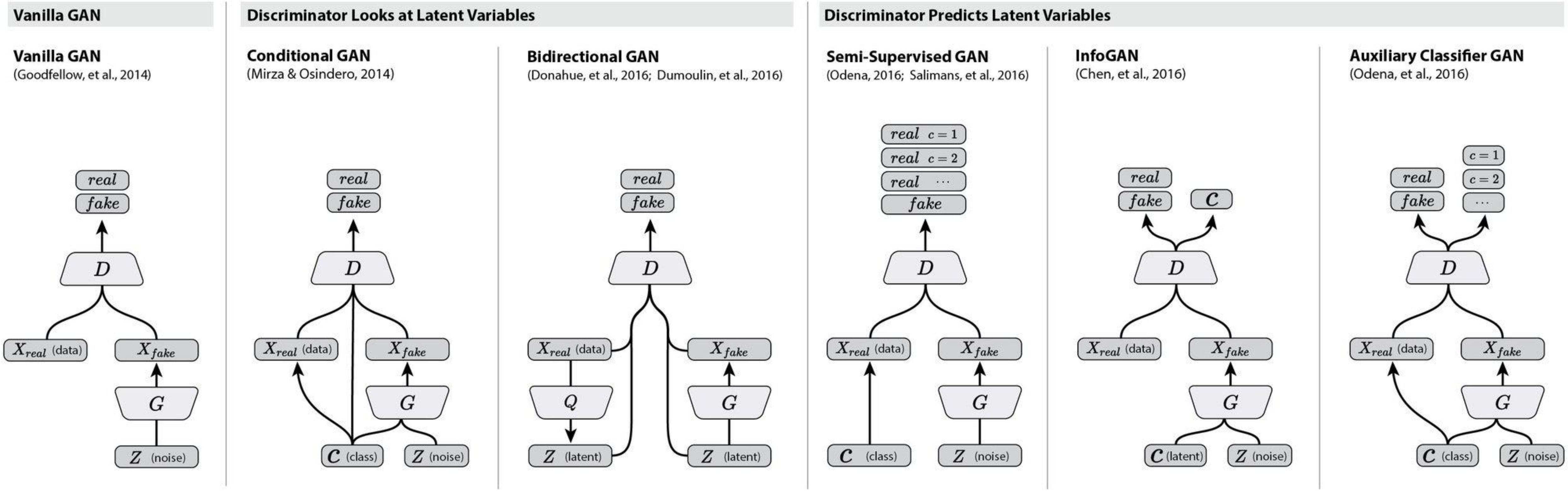
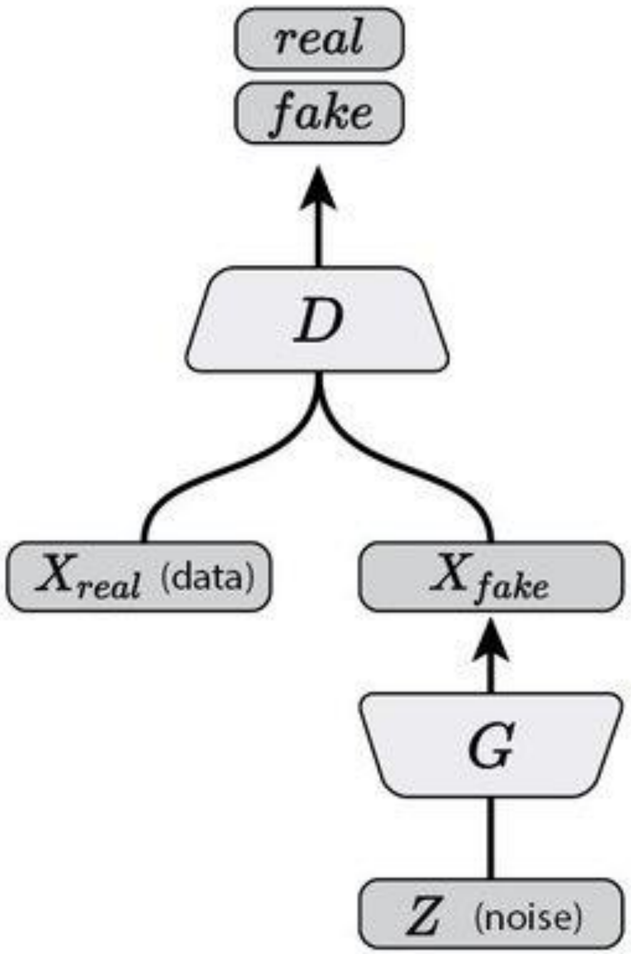
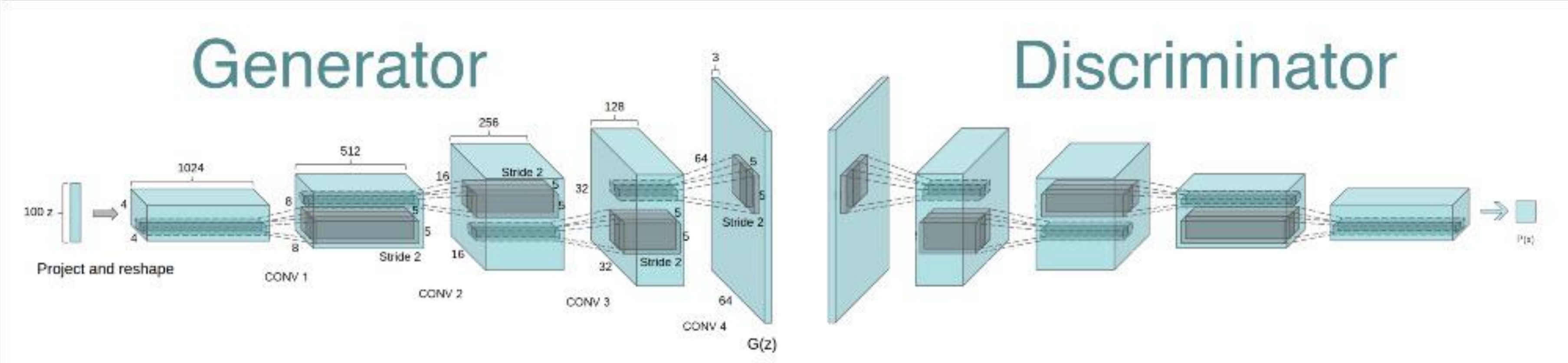


Image: Christopher Olah

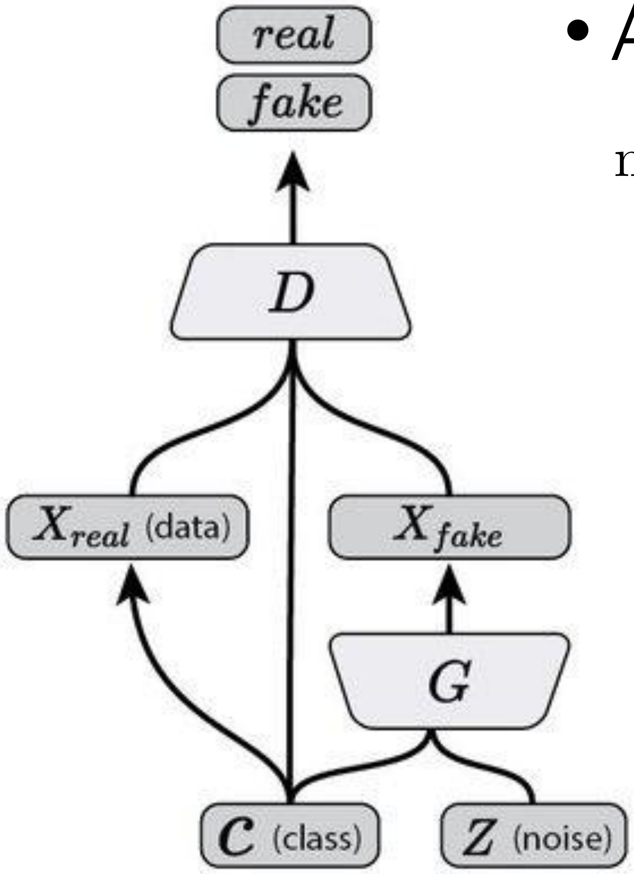
Vanilla GAN (Goodfellow et al., 2014)





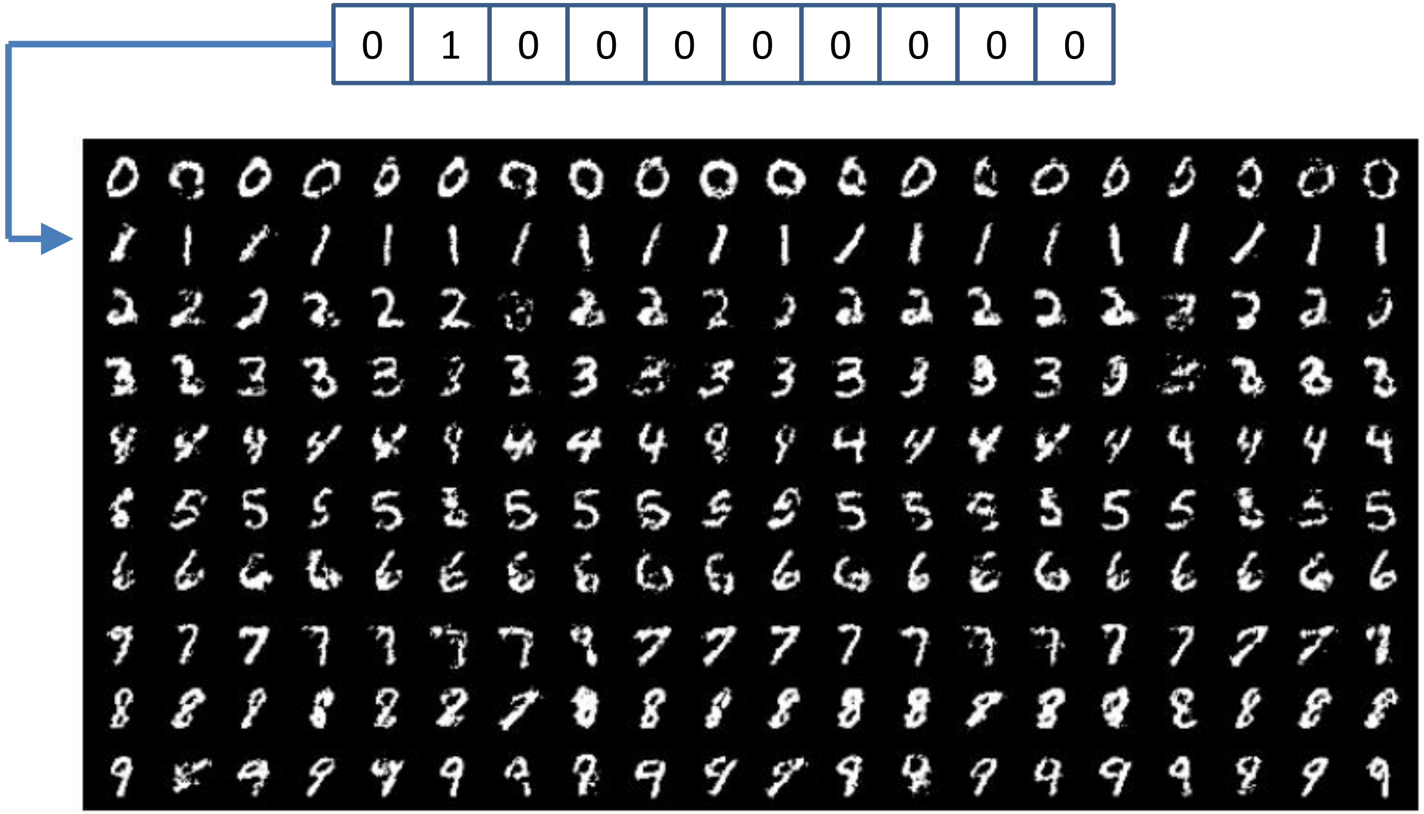
DCGAN (Radford et al., 2015)

Conditional GAN (Mirza and Osindero, 2014)

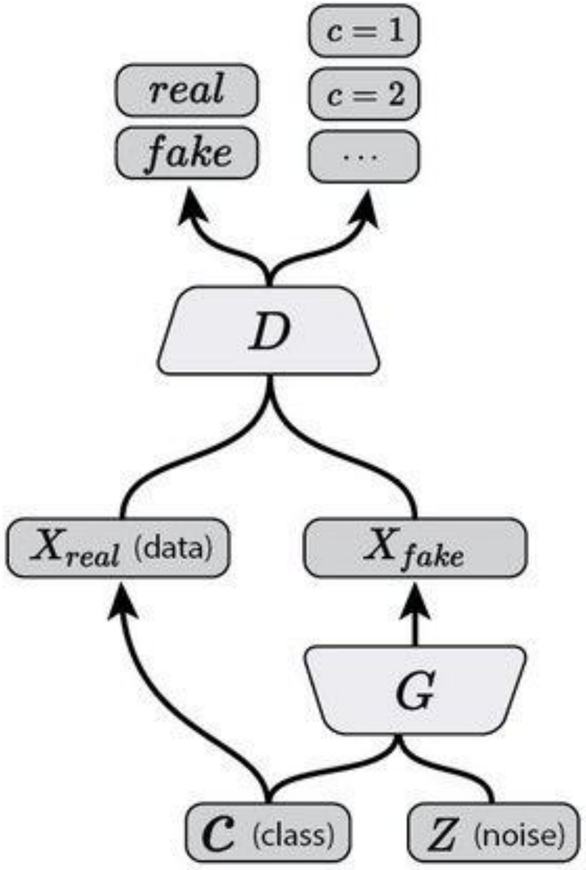


- Add conditional variables $\boldsymbol{y}$ into G and D

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



Auxiliary Classifier GAN (Odena et al., 2016)



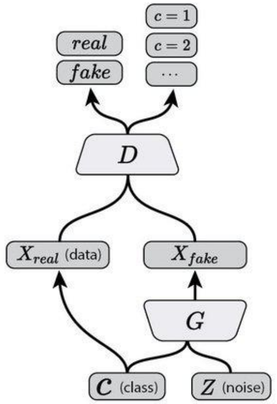
- Every generated sample has a corresponding class label

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

- 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)

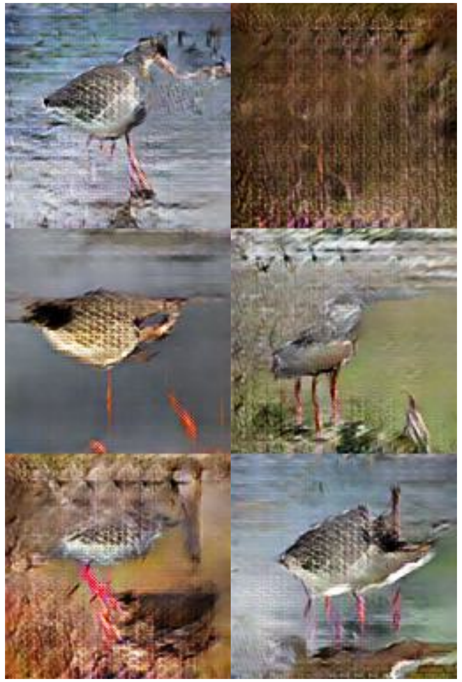


128×128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet


monarch butterfly

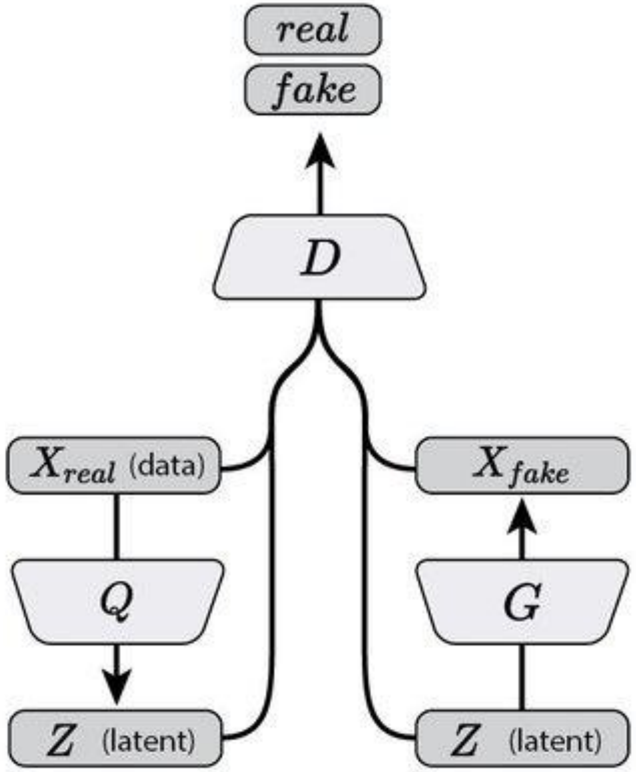

goldfinch


daisy


redshank


grey whale

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



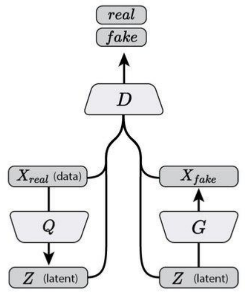
- Jointly learns a generator network and an inference network using an adversarial process.

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

CelebA reconstructions

SVNH reconstructions

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



LSUN bedrooms

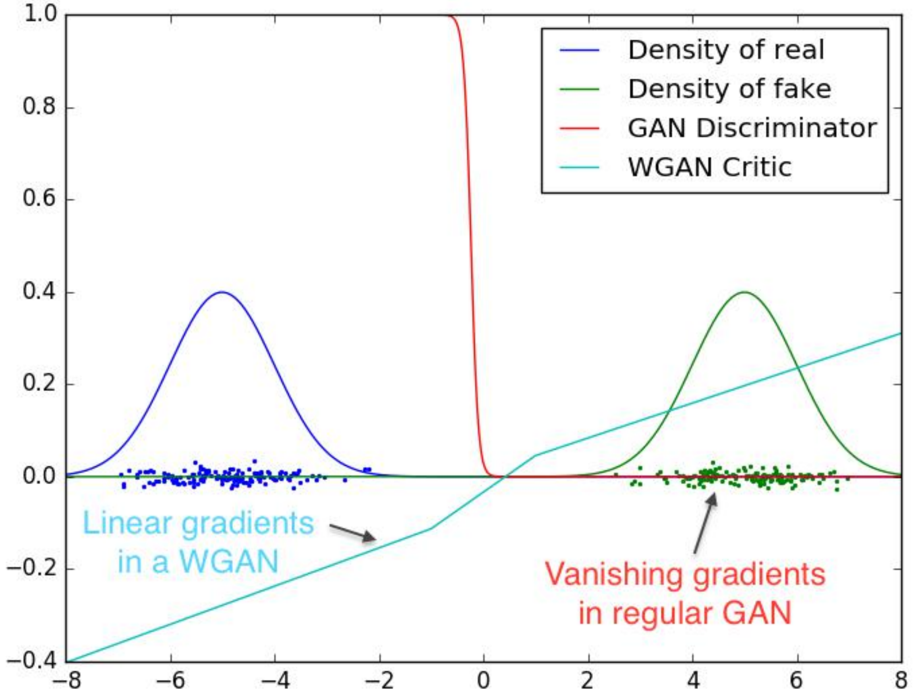
Tiny ImageNet

Wasserstein GAN (Arjovsky et al., 2016)

- Objective based on Earth-Mover or Wassertein distance:

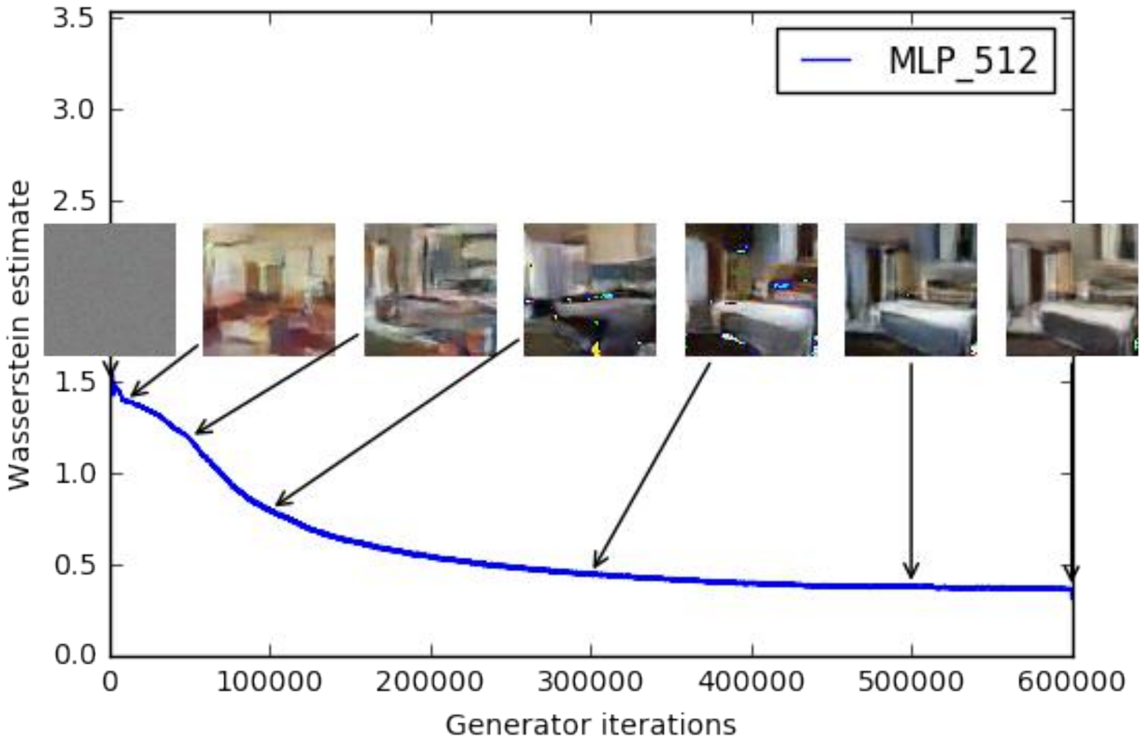
$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} [D_\omega(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} [D_\omega(G_\theta(\boldsymbol{z}))]$$

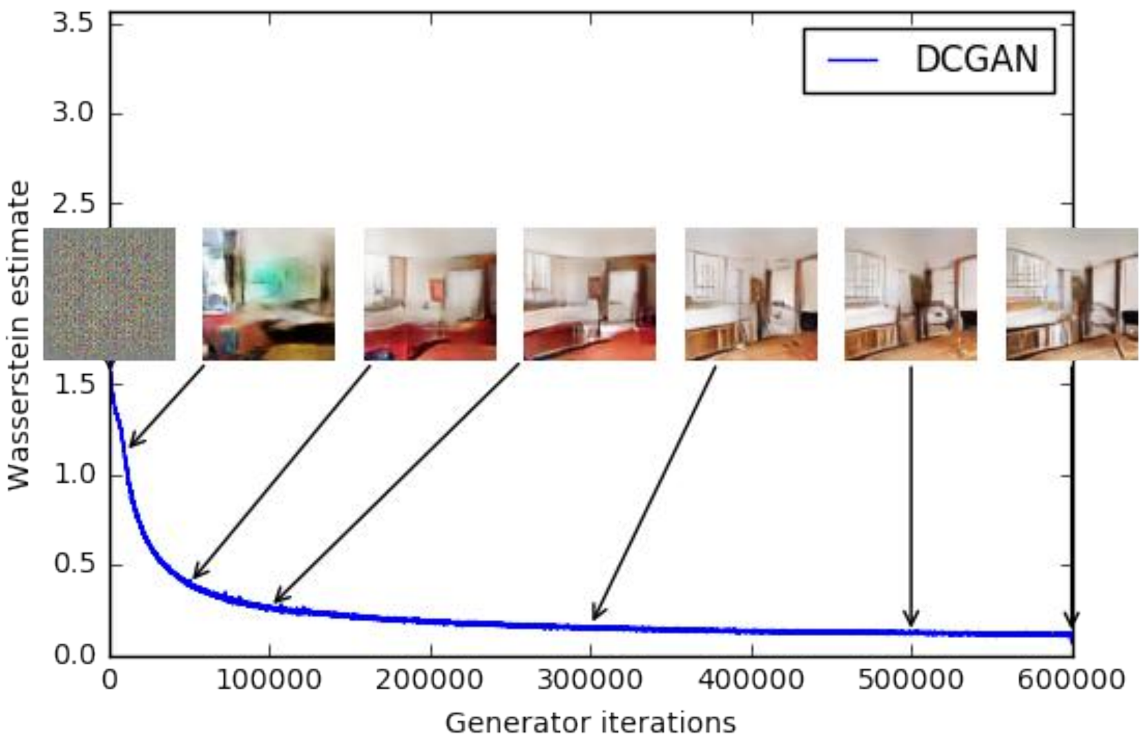
- Provides nice gradients over real and fake samples



Wasserstein GAN (Arjovsky et al., 2016)

- Wasserstein loss seems to correlate well with image quality.





WGAN with gradient penalty (Gulraani et al., 2017)

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\boldsymbol{x}}) \right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} \left[D(\boldsymbol{x}) \right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}$$

- Faster convergence and higher-quality samples than WGAN with weight clipping
- Train a wide variety of GAN architectures with almost no hyperparameter tuning, including discrete models

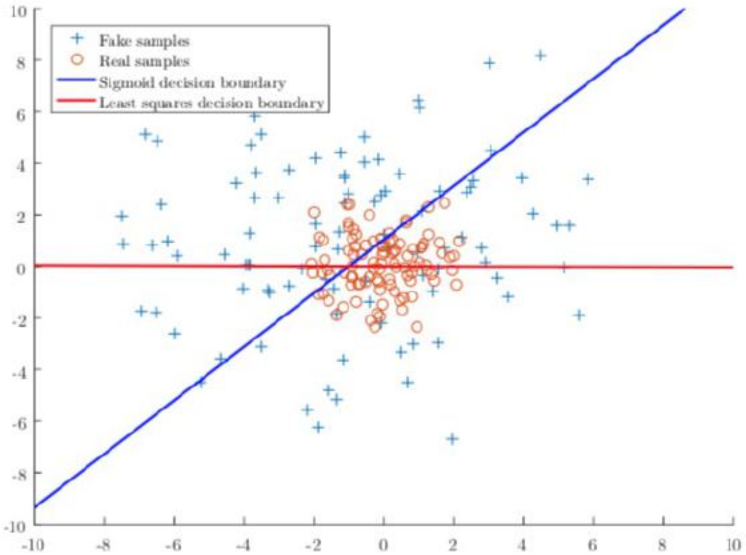
WGAN with gradient penalty	
Busino game camperate spent odea	Solice Norkedin pring in since
In the bankaway of smarling the	ThiS record (31.) UBS) and Ch
SingersMay , who kill that imvic	It was not the annuas were plogr
Keray Pents of the same Reagun D	This will be us , the ect of DAN
Manging include a tudancs shat "	These leaded as most-worsd p2 a0
His Zuith Dudget , the Denmbern	The time I paid0a South Cubry i
In during the Uitational questio	Dour Fraps higs it was these del
Divos from The ' noth ronkies of	This year out howneed allowed lo
She like Monday , of macunsuer S	Kaulna Seto consficutes to repor
The investor used ty the present	A can teal , he was schoon news
A papees are cointry congress oo	In th 200. Pesish picriers rega
A few year inom the group that s	Konney Panice rimimber the teami
He said this syenn said they wan	The new centuct cut Denester of
As a world 1 88 ,for Autouries	The near , had been one injostie
Foand , th Word people car , Il	The incestion to week to shorted
High of the upseader homing pull	The company the high product of
The guipe is worly move dogsfor	20 - The time of accomplete , wh
The 1874 incidested he could be	John WVuderenson seqiivic spends
The allo tooks to security and c	A ceetens in indestredly the Wat
Standard GAN objective	
ddddddddddddddddddddddddddddddd	ddddddddddddddddddddddddddddddd
ddddddddddddddddddddddddddddddd	ddddddddddddddddddddddddddddddd

Samples from a character-level GAN language model on Google Billion Word

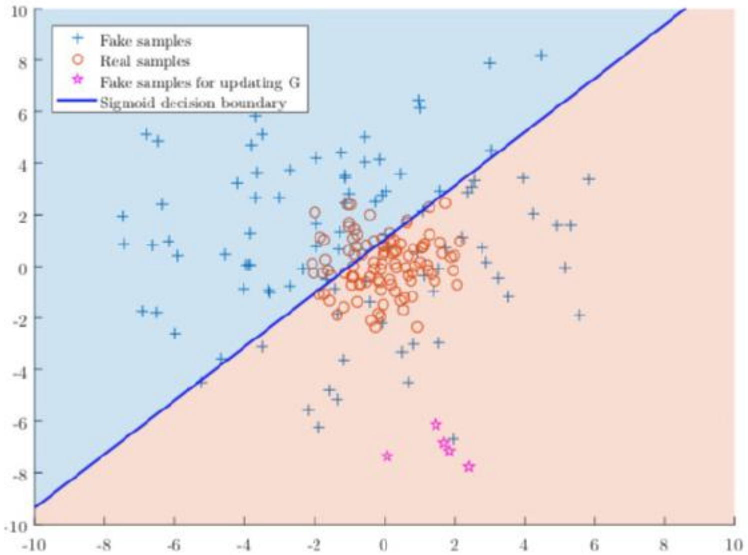
Least Squares GAN (LSGAN) (Mao et al., 2017)

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

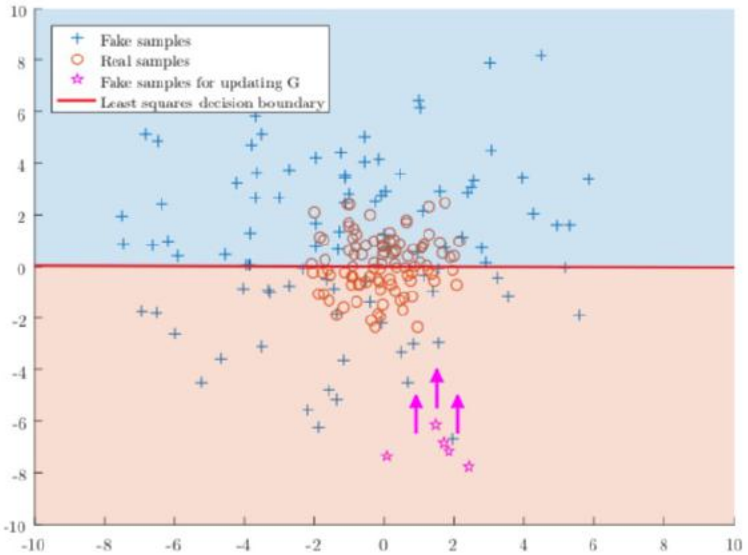
$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim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



Decision boundaries of Sigmoid & Least Squares loss functions



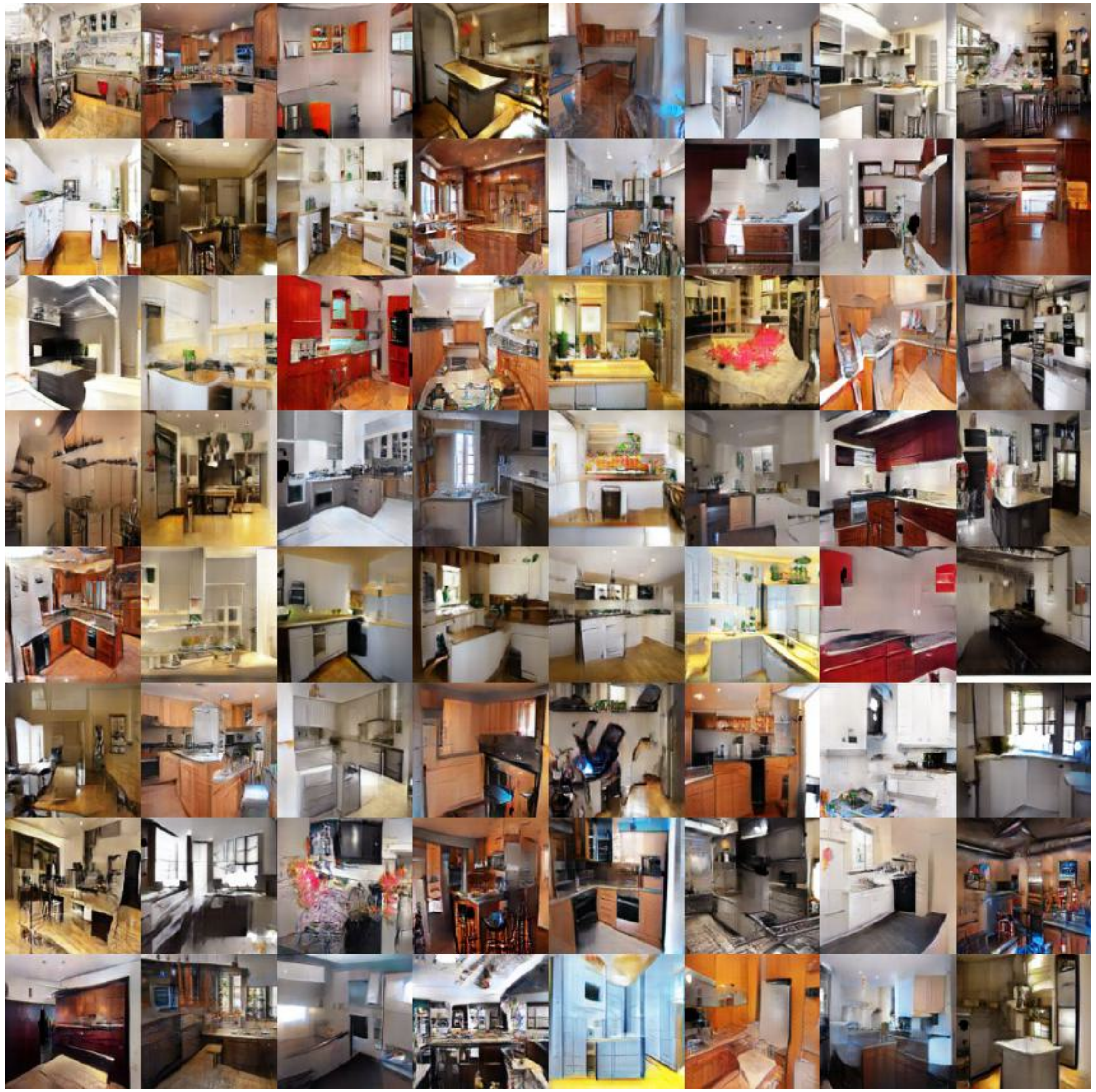
Sigmoid decision boundary



Least Squares decision boundary

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

Church



Kitchen

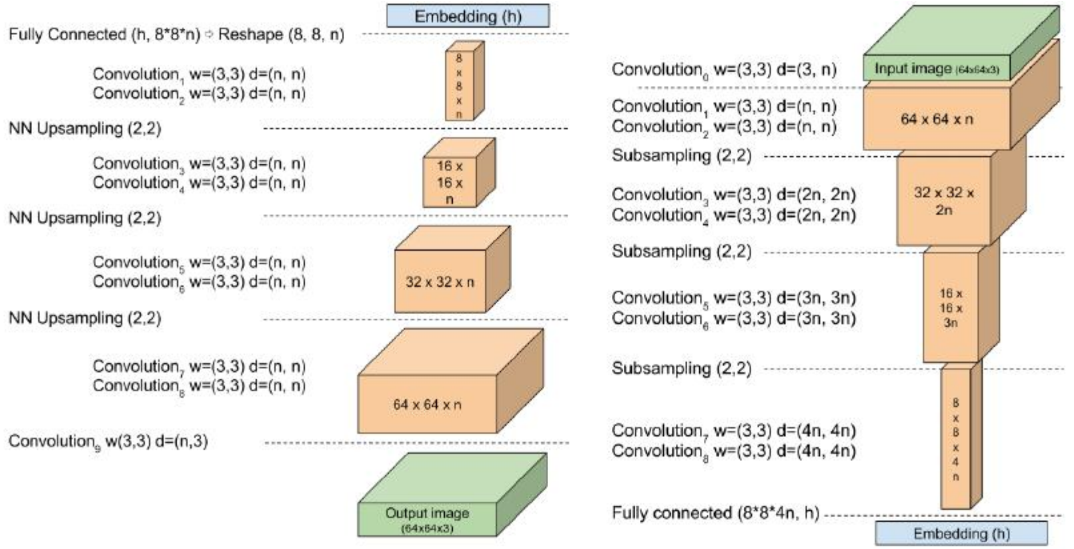
Boundary Equilibrium GAN (BEGAN)

(Berthelot et al., 2017)

- A loss derived from the Wasserstein distance for training auto-encoder based GANs

$$\mathcal{L}(v) = |v - D(v)|^{\eta} \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} & \text{is the autoencoder function.} \\ \eta \in \{1, 2\} & \text{is the target norm.} \\ v \in \mathbb{R}^{N_x} & \text{is a sample of dimension } N_x. \end{cases}$$

  - Wasserstein distance btw. the reconstruction losses of real and generated data



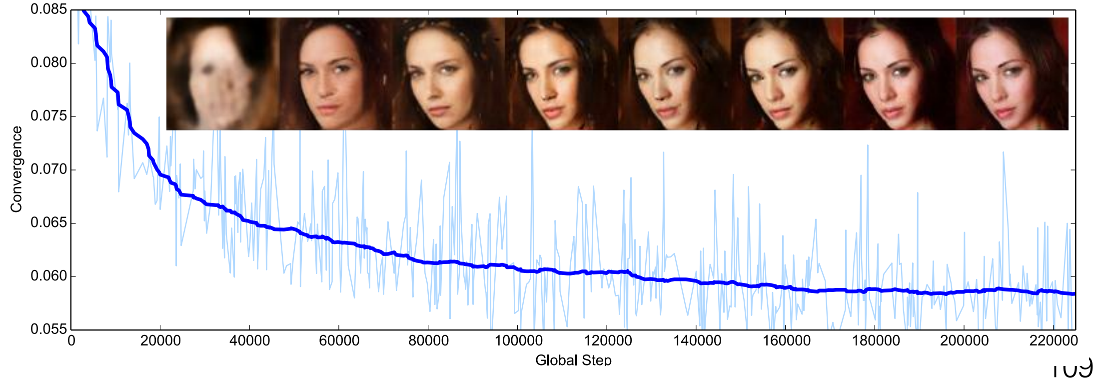
(a) Generator/Decoder (b) Encoder

- Convergence measure:

$$\mathcal{M}_{global} = \mathcal{L}(x) + |\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))|$$

- Objective:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t . \mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k (\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$



BEGANs for CelebA

360K celebrity face images
128x128 with 128 filters

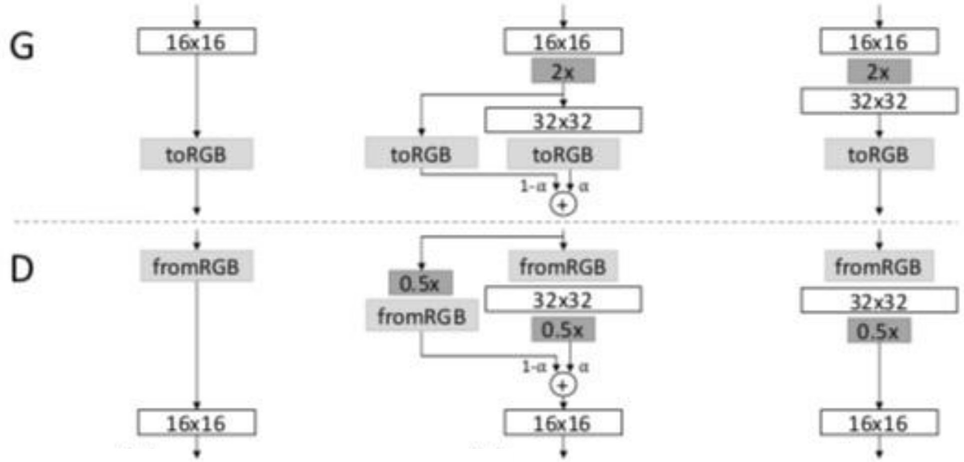
(Berthelot et al., 2017)

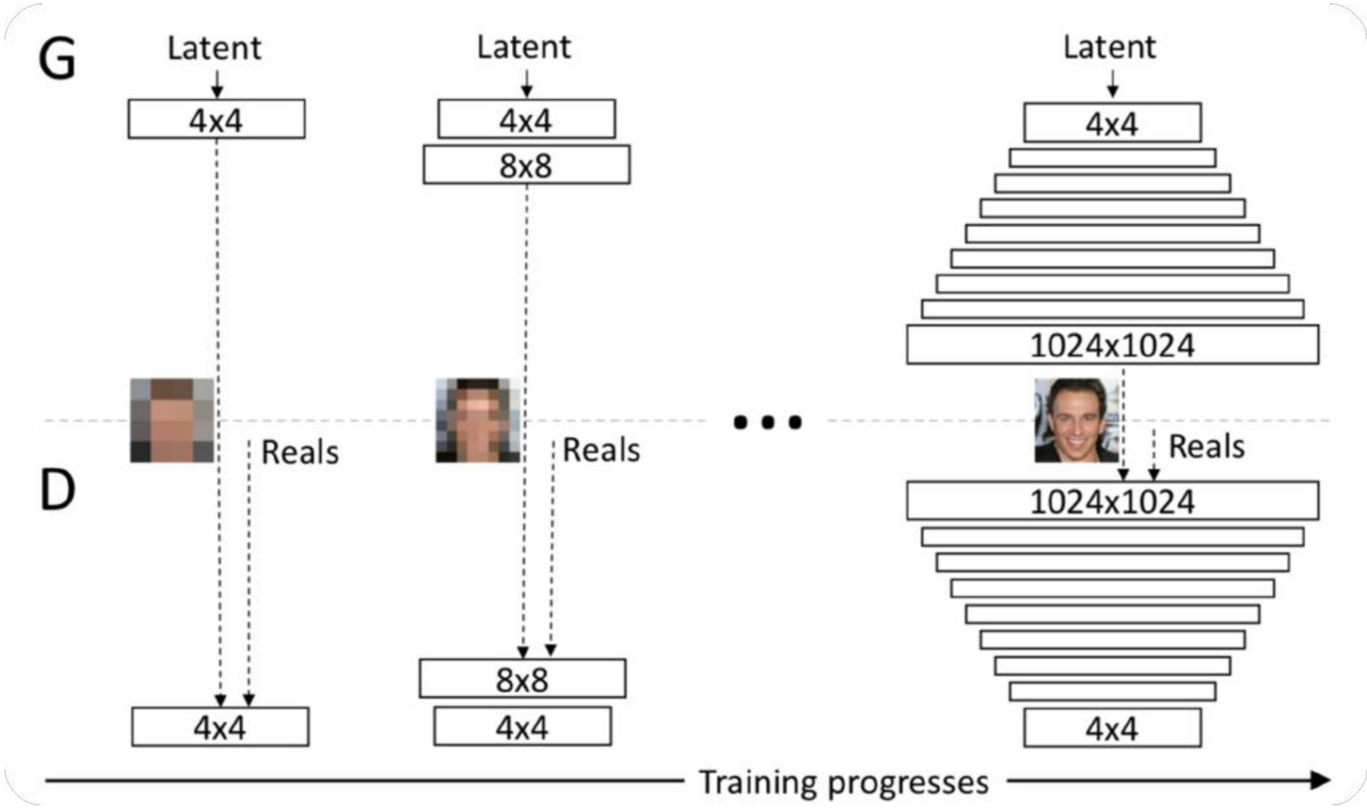
Interpolations in the latent space

Mirror interpolation example

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)

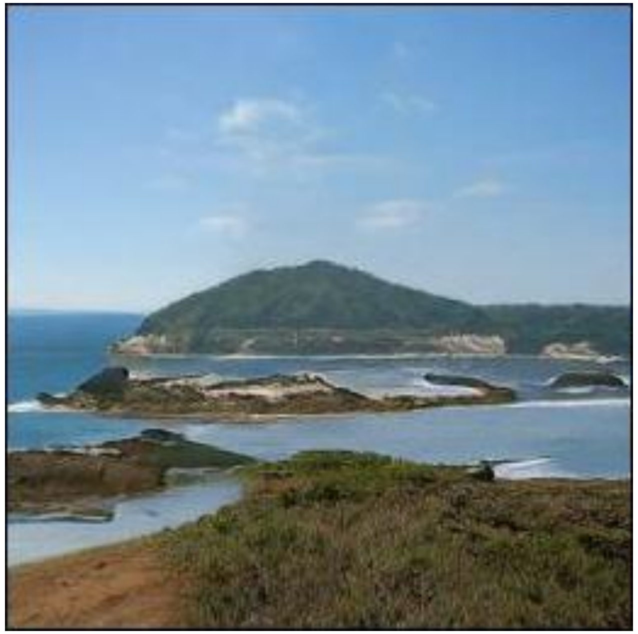
- Training process

Progressive GANs (Karras et al., 2018)

CelebA-HQ
random interpolations

BigGANs (Brock et al., 2019)

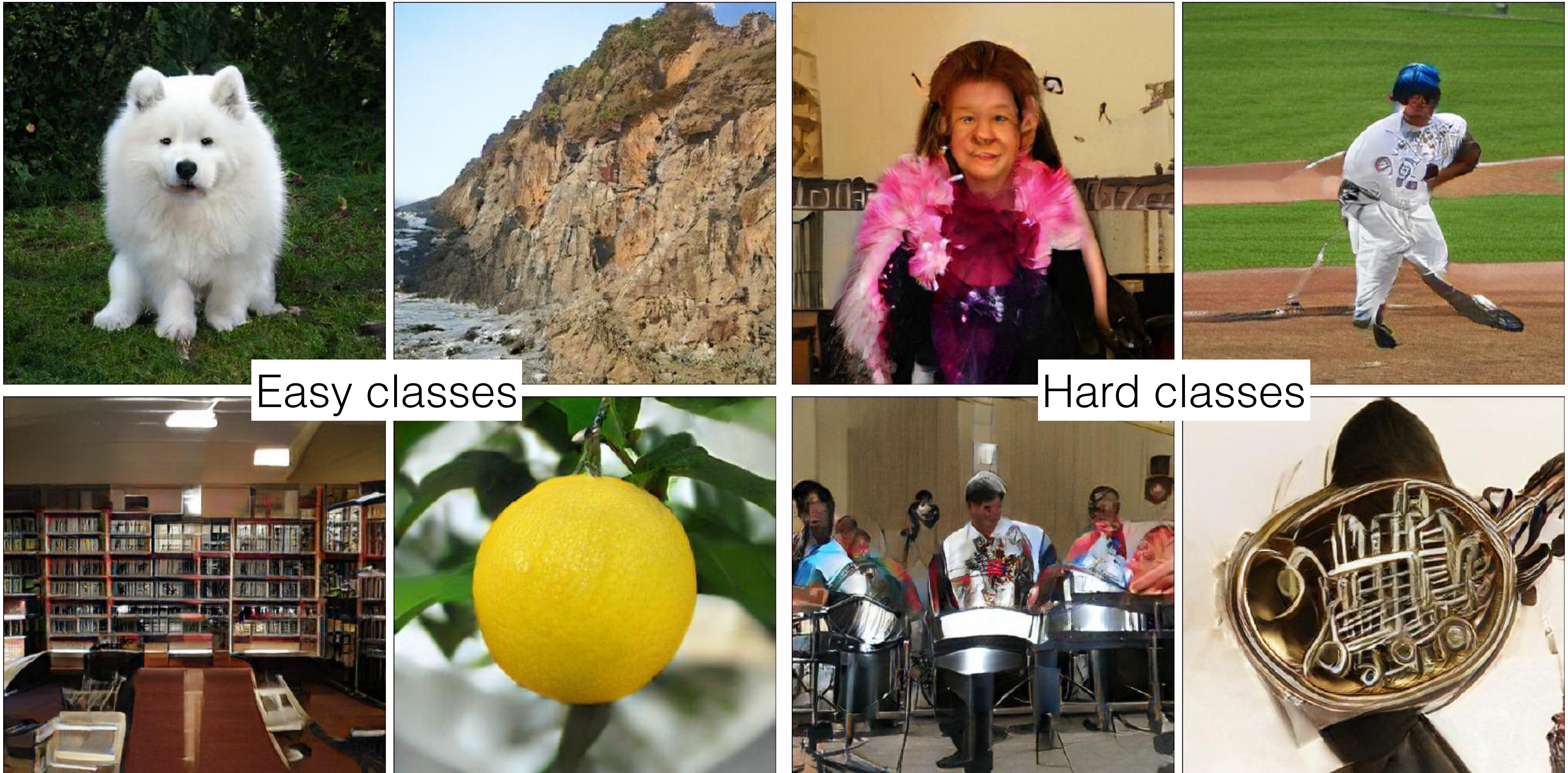
High resolution, class-conditional samples generated by the model

- 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:

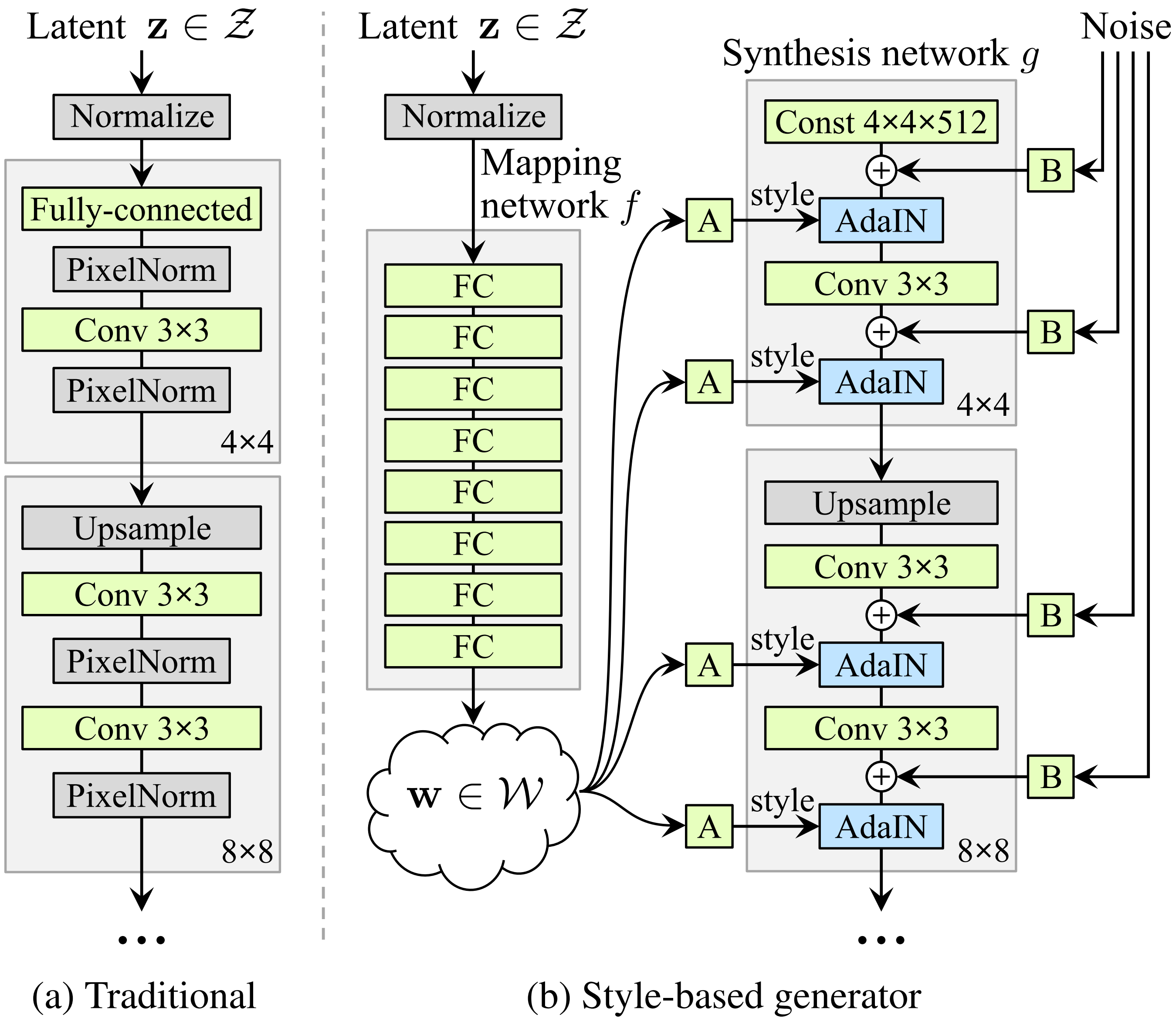
$$R_\beta(W) = \beta \| W^\top W \odot (\mathbf{1} - I) \|_{\mathrm{F}}^2,$$

BigGANs (Brock et al., 2019)



Resolution: 512x512

StyleGAN (Karras et al., 2019)



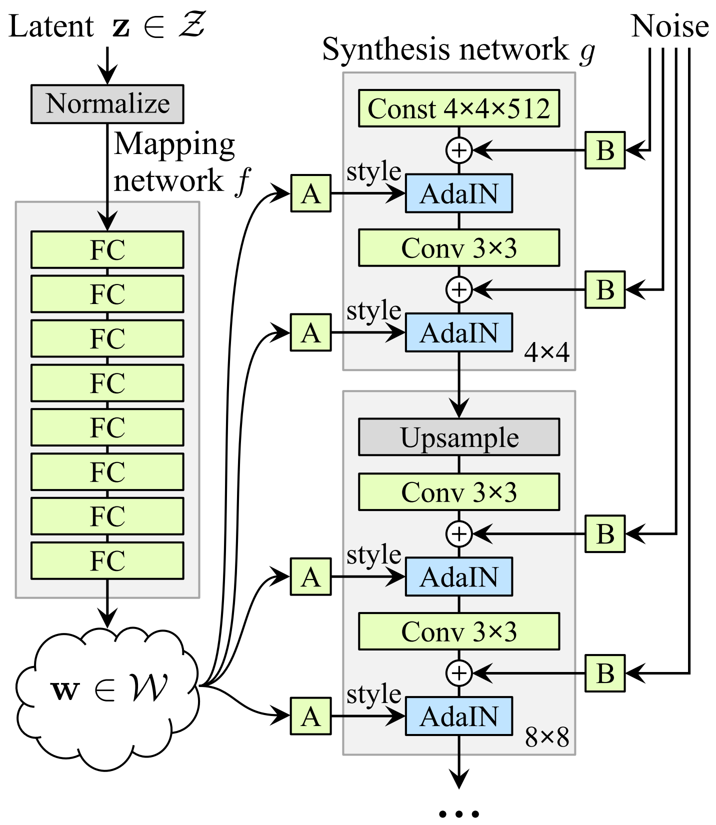
Feature map affine transformation:

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

Samples (trained on the FFHQ dataset)

StyleGAN (Karras et al., 2019)

- Swapping out the **destination** style for the **source** style



Some Applications of GANs

Semi-supervised Classification

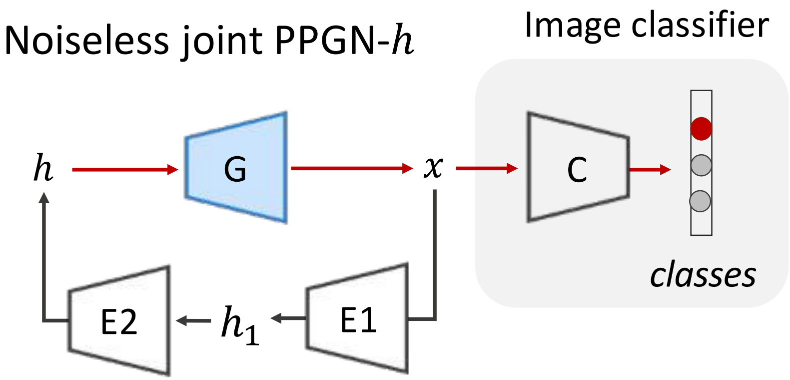
(Salimans et al., 2016; Dumoulin et al., 2016)

SVNH

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

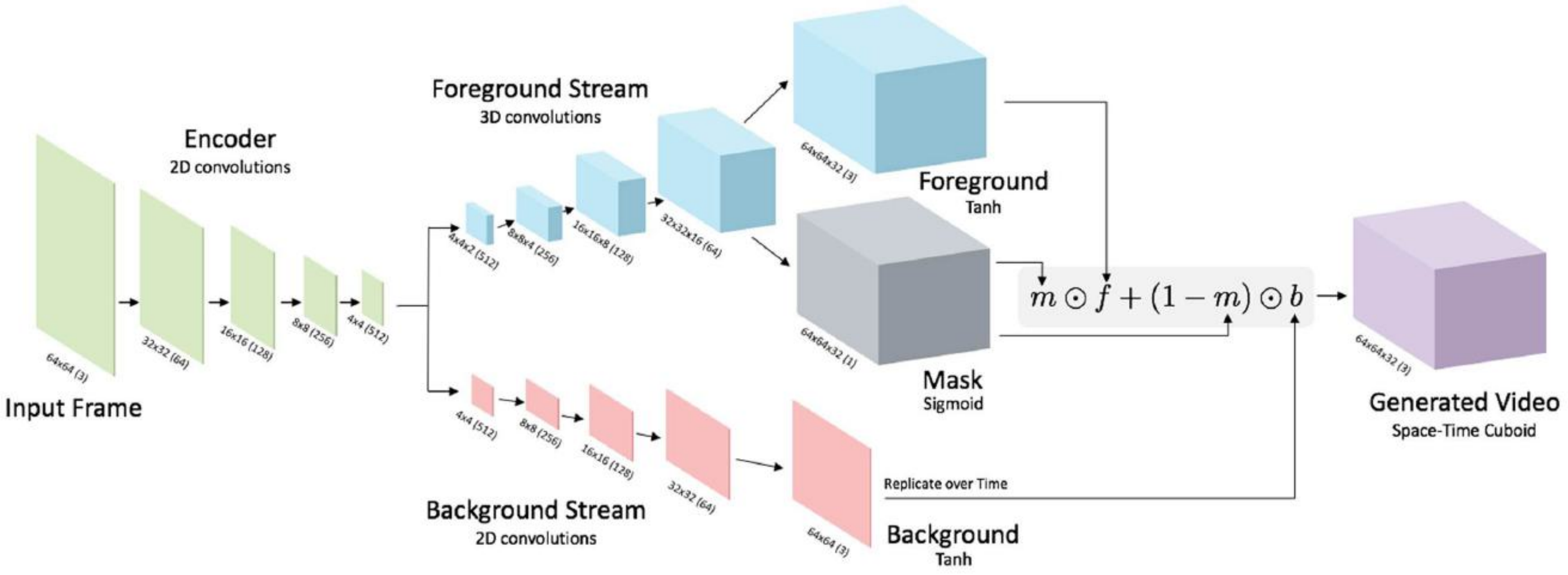
Class-specific Image Generation (Nguyen et al., 2016)

- Generates 227x227 realistic images from all ImageNet classes
- Combines adversarial training, moment matching, denoising autoencoders, and Langevin sampling

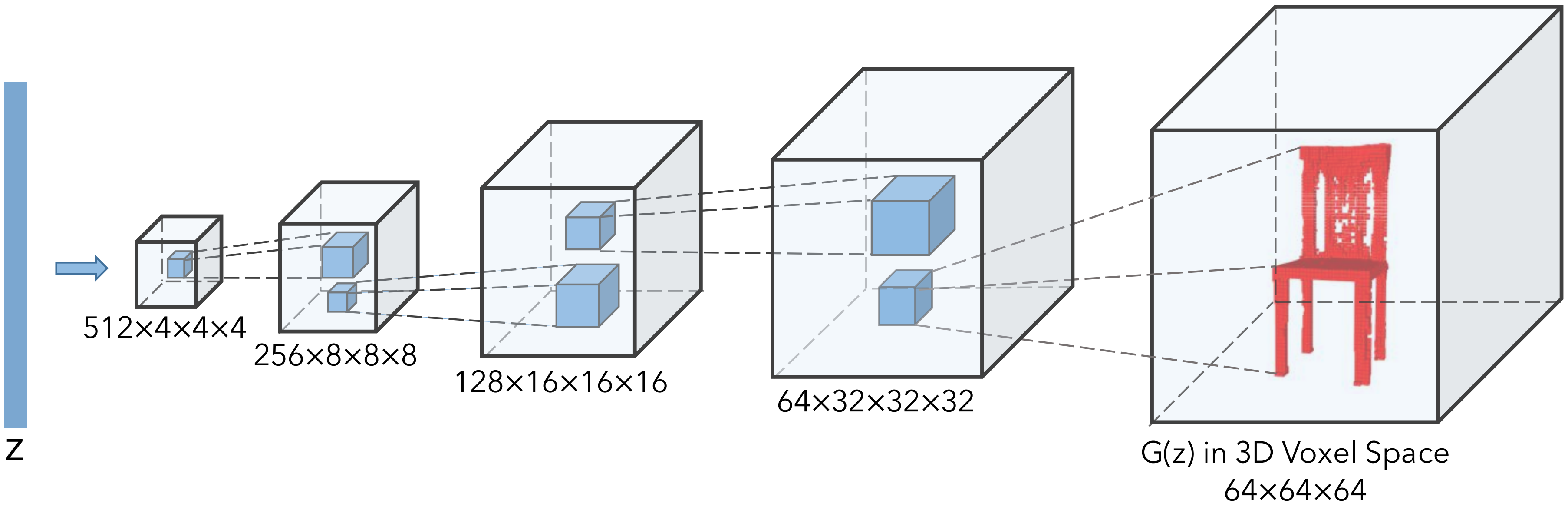


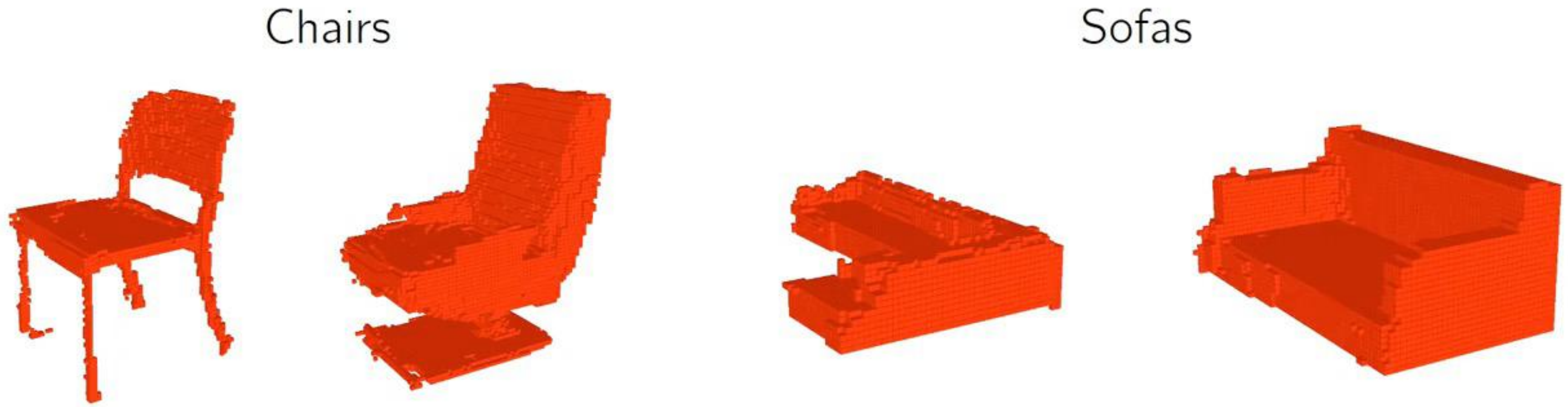
redshank ant monastery volcano

Video Generation (Vondrick et al., 2016)



Generative Shape Modeling (Wu et al., 2016)





Text-to-Image Synthesis (Zhang et al., 2016)

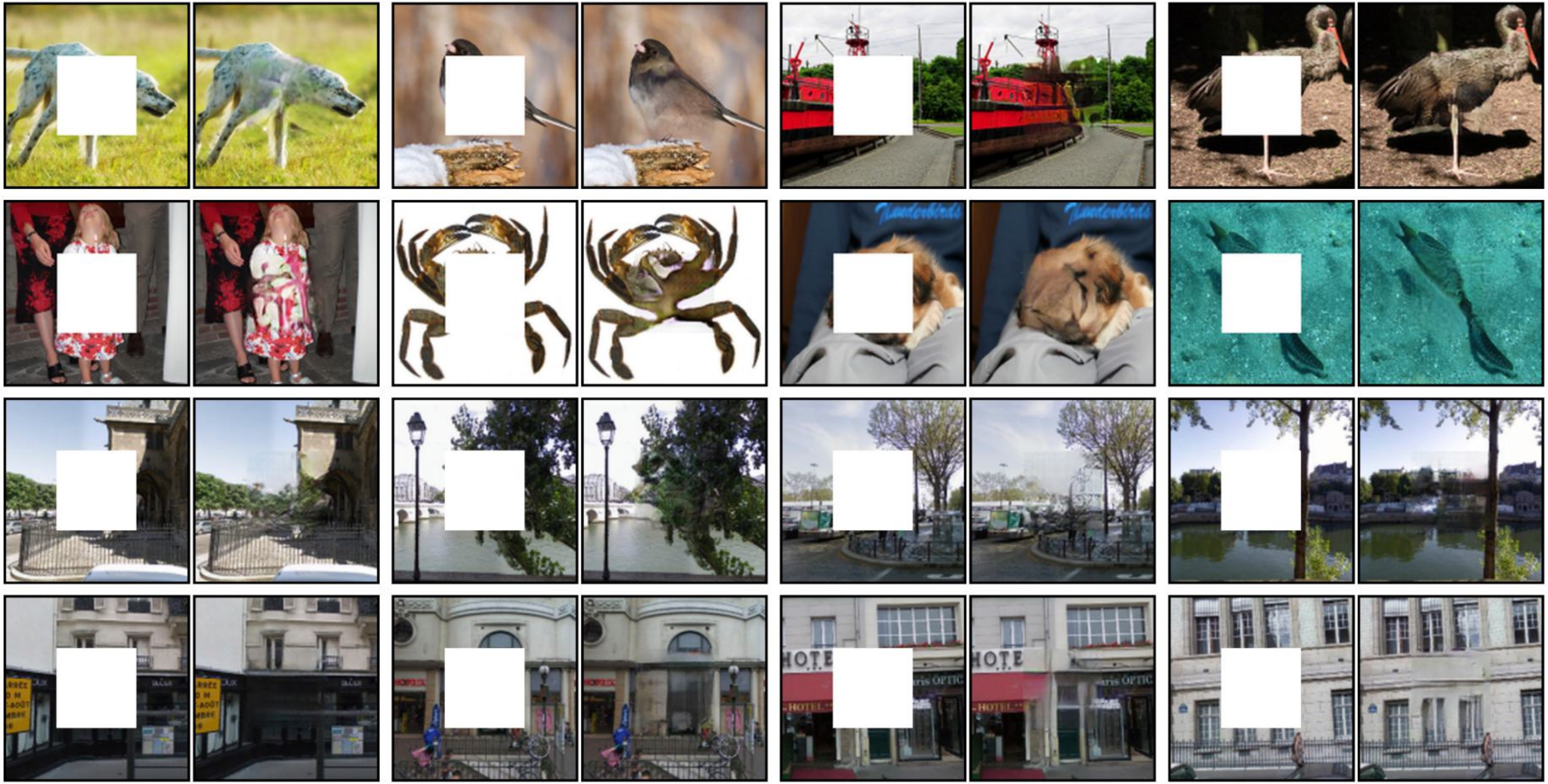
The small bird has a red head with feathers that fade from red to gray from head to tail

The petals of this flower are white with a large stigma	A unique yellow flower with no visible pistils protruding from the center	This flower is pink and yellow in color, with petals that are oddly shaped	This is a light colored flower with many different petals on a green stem	This flower is yellow and green in color, with petals that are ruffled	The flower have large petals that are pink with yellow on some of the petals	A flower that has white petals with some tones of yellow and green filaments

Single Image Super-Resolution (Ledig et al., 2016)

- Combine content loss with adversarial loss

Image Inpainting (Pathak et al., 2016)



Unsupervised Domain Adaptation (Bousmalis et al., 2016)

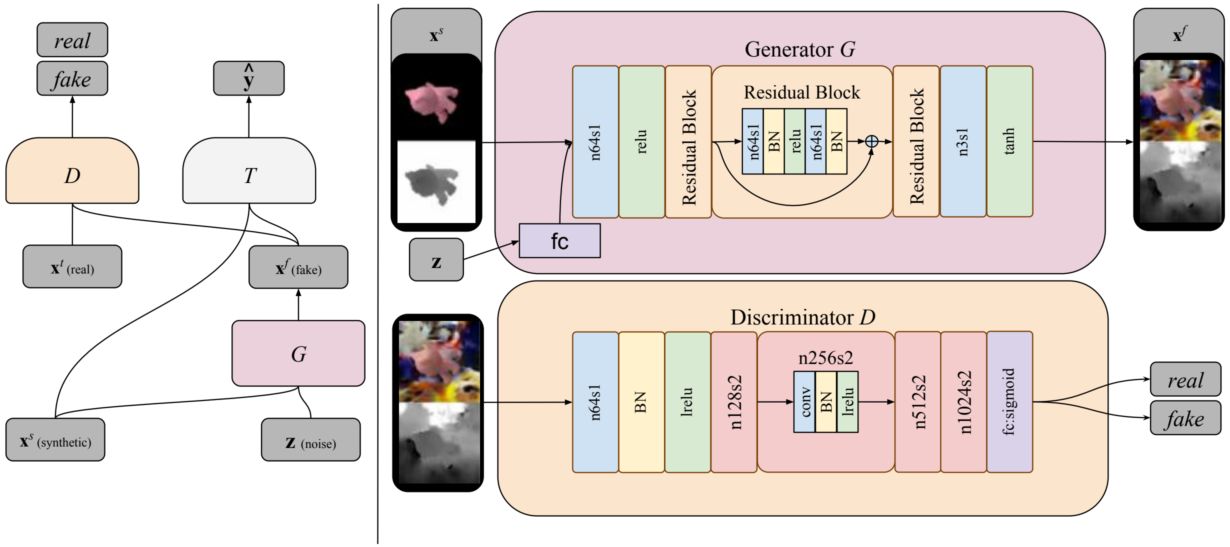
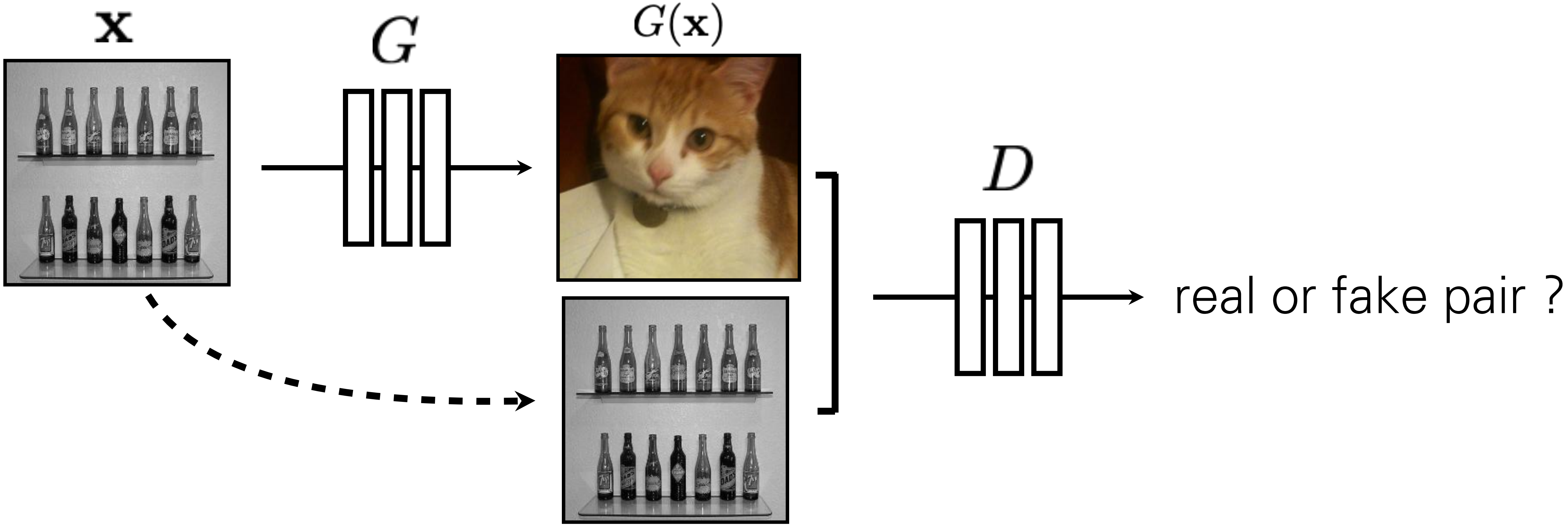


Image examples from the Linemod dataset

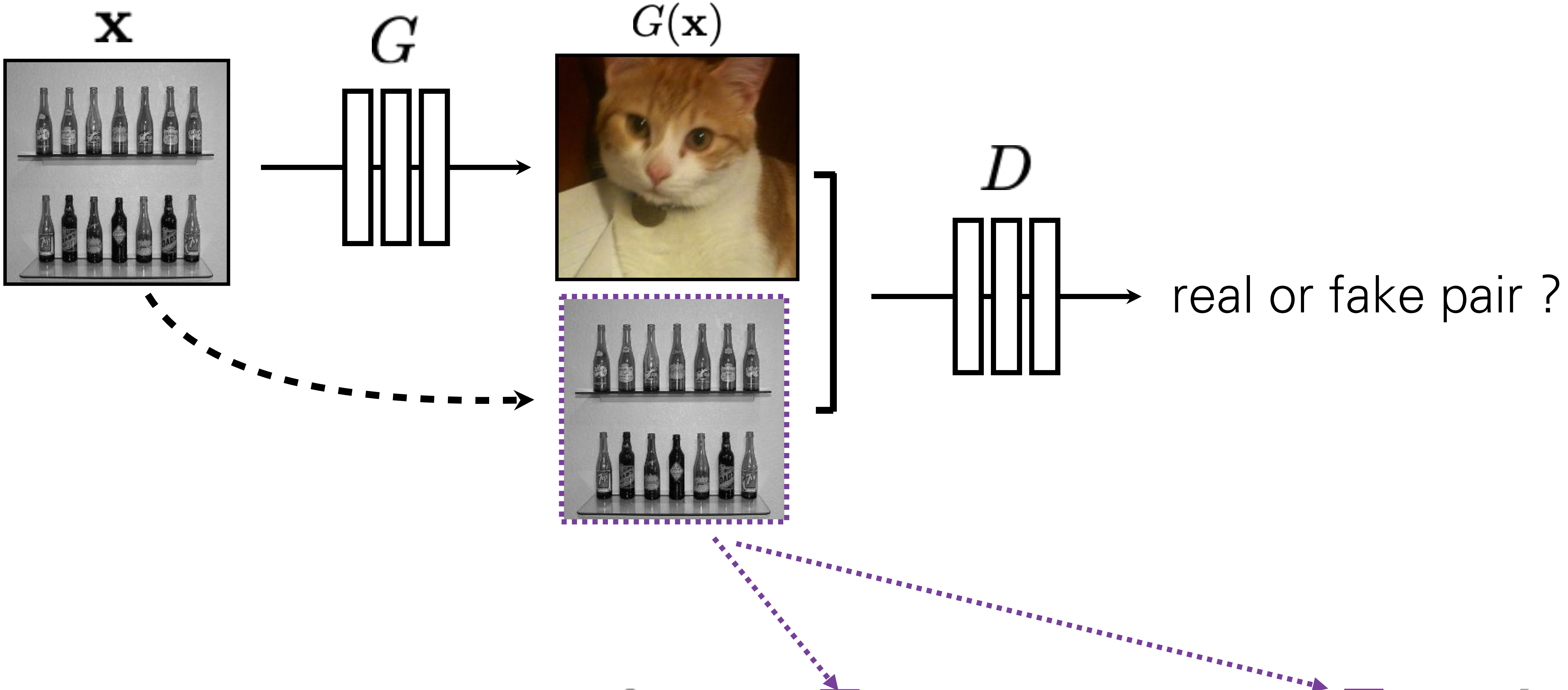
RGDB image samples
(conditioned on a synthetic image)

Image to Image Translation (Pix2Pix)

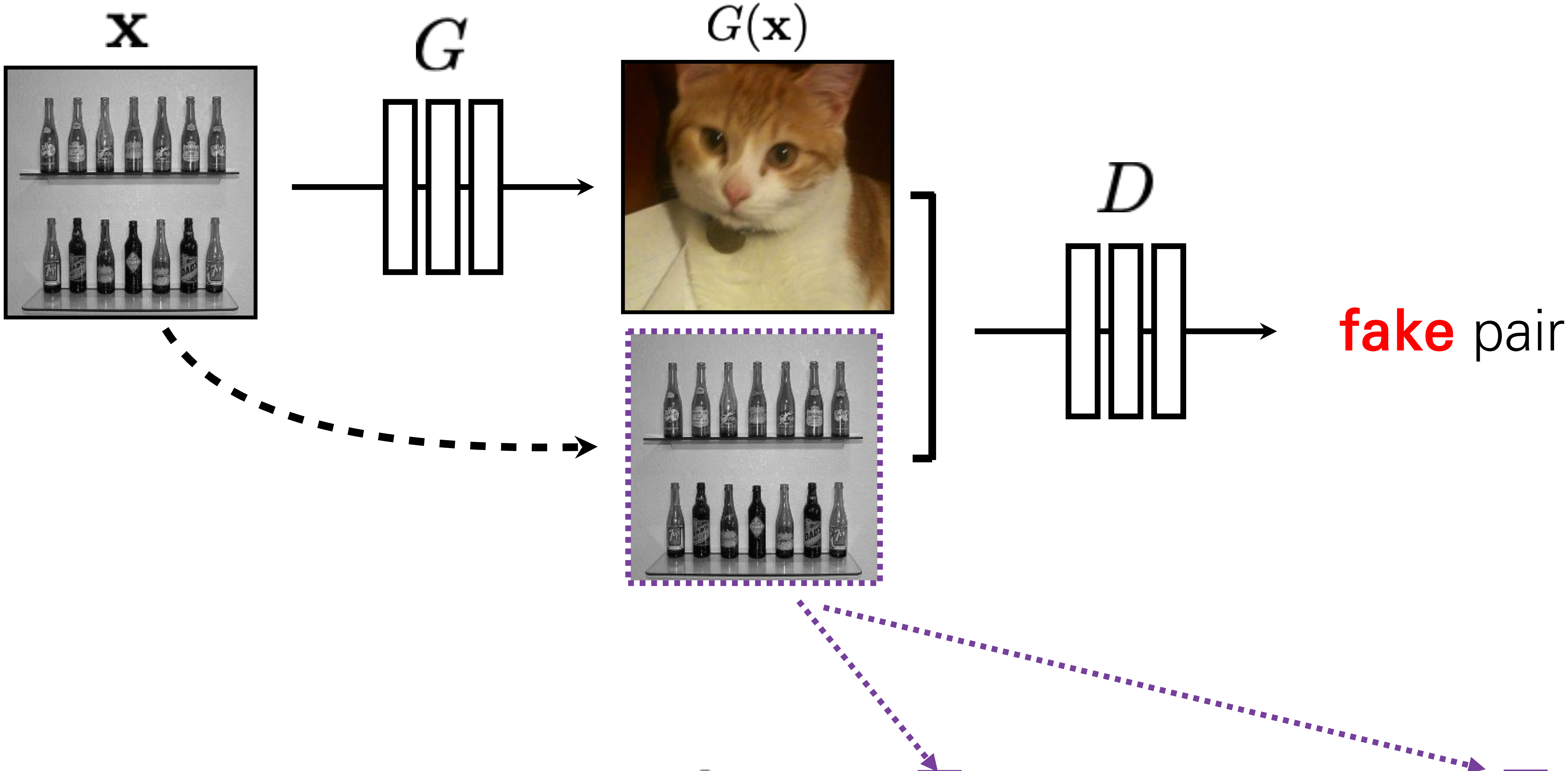
(Isola et al. 2016)



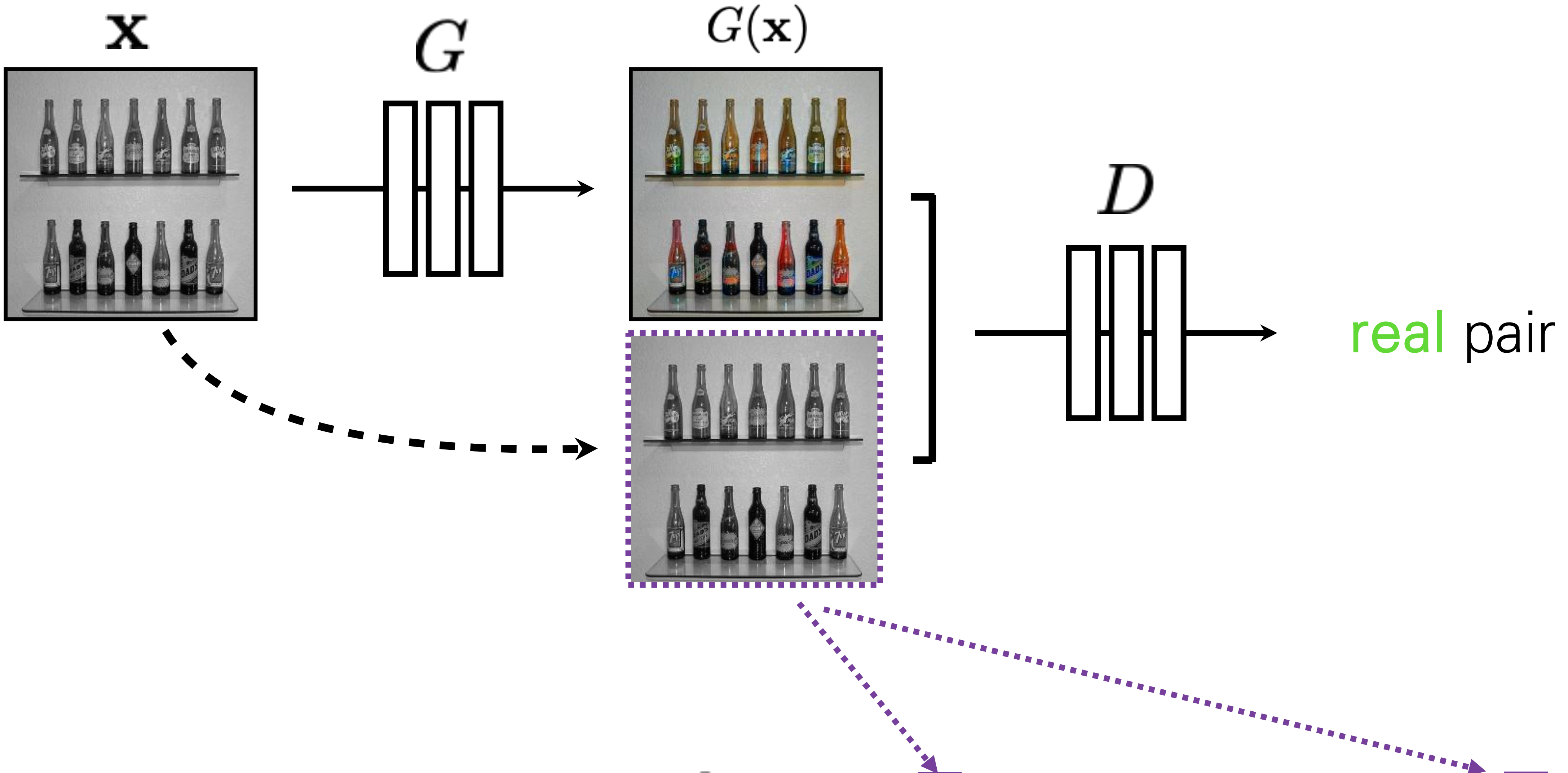
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$



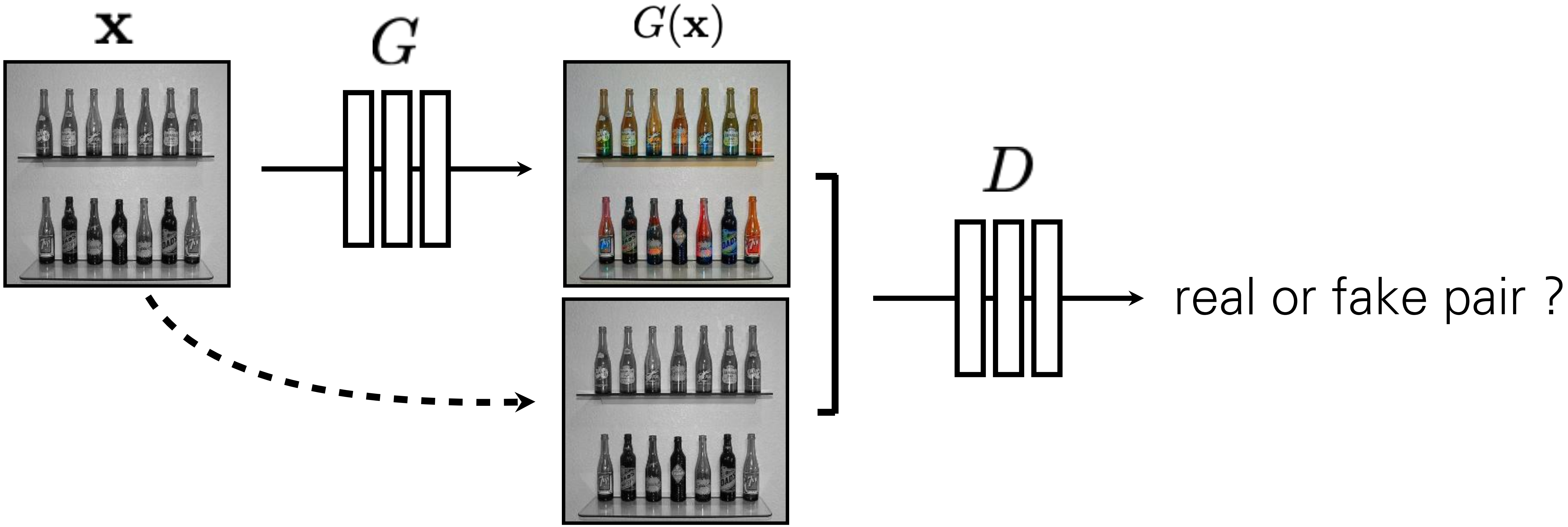
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



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



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



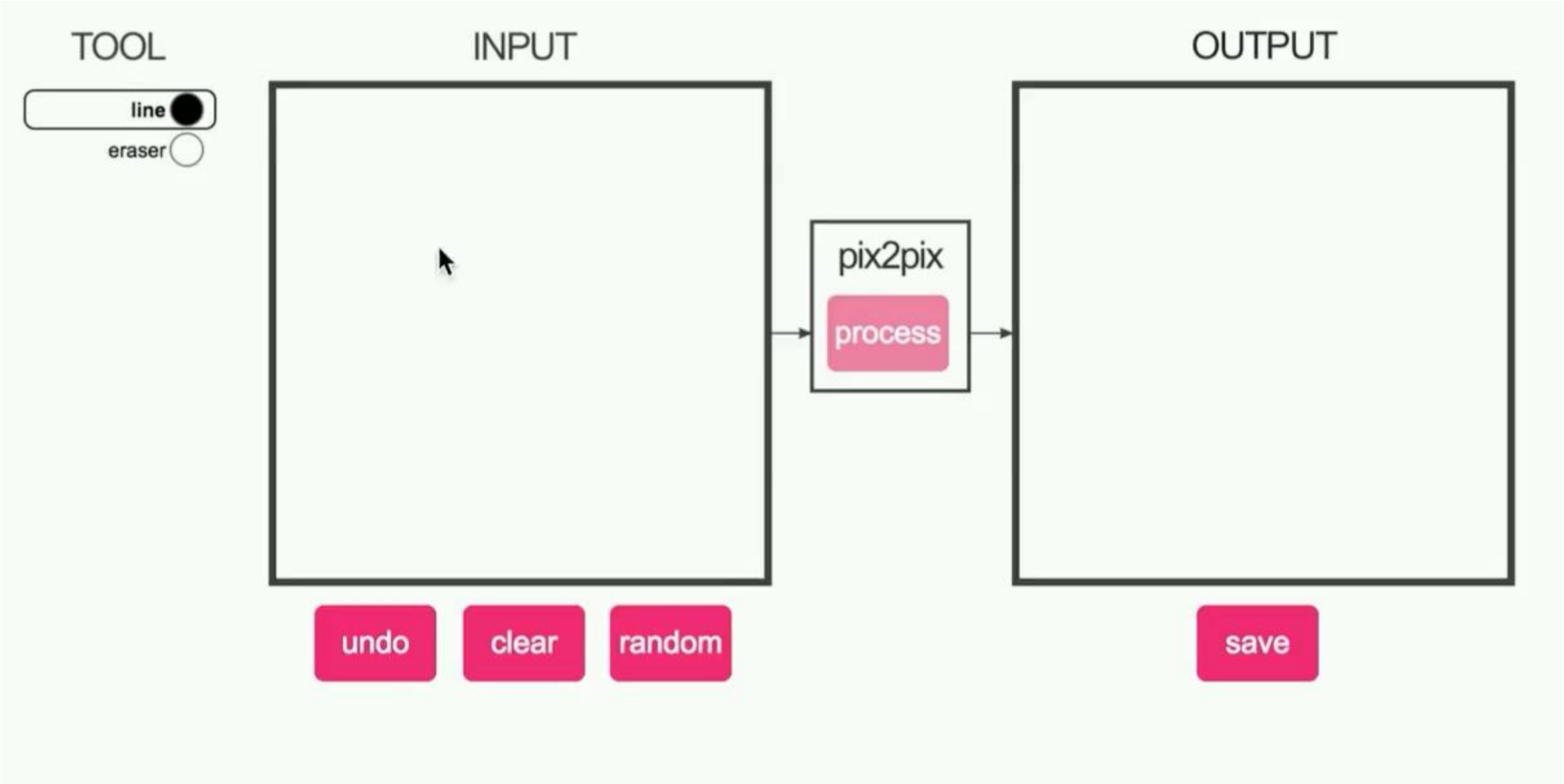
$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$

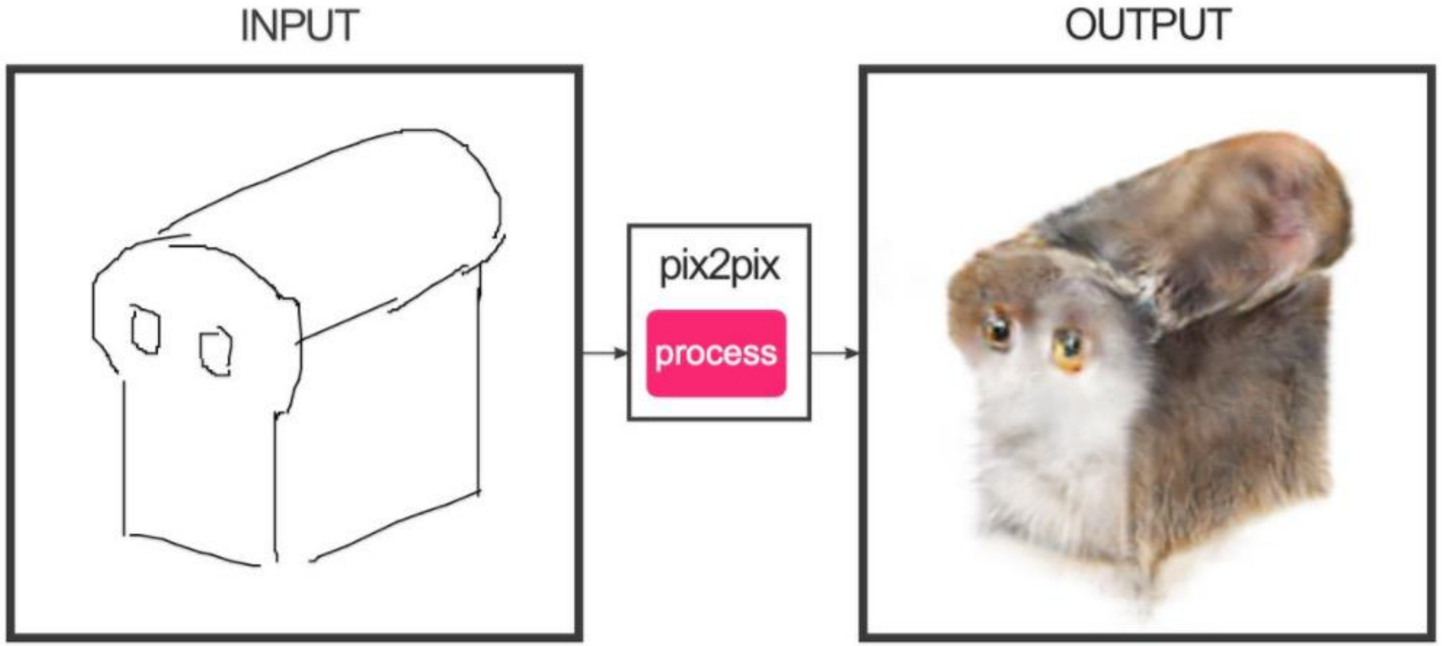
BW → Color

Input Output Input Output Input Output

Data from [Russakovsky et al. 2015]

#edges2cats [Chris Hesse]

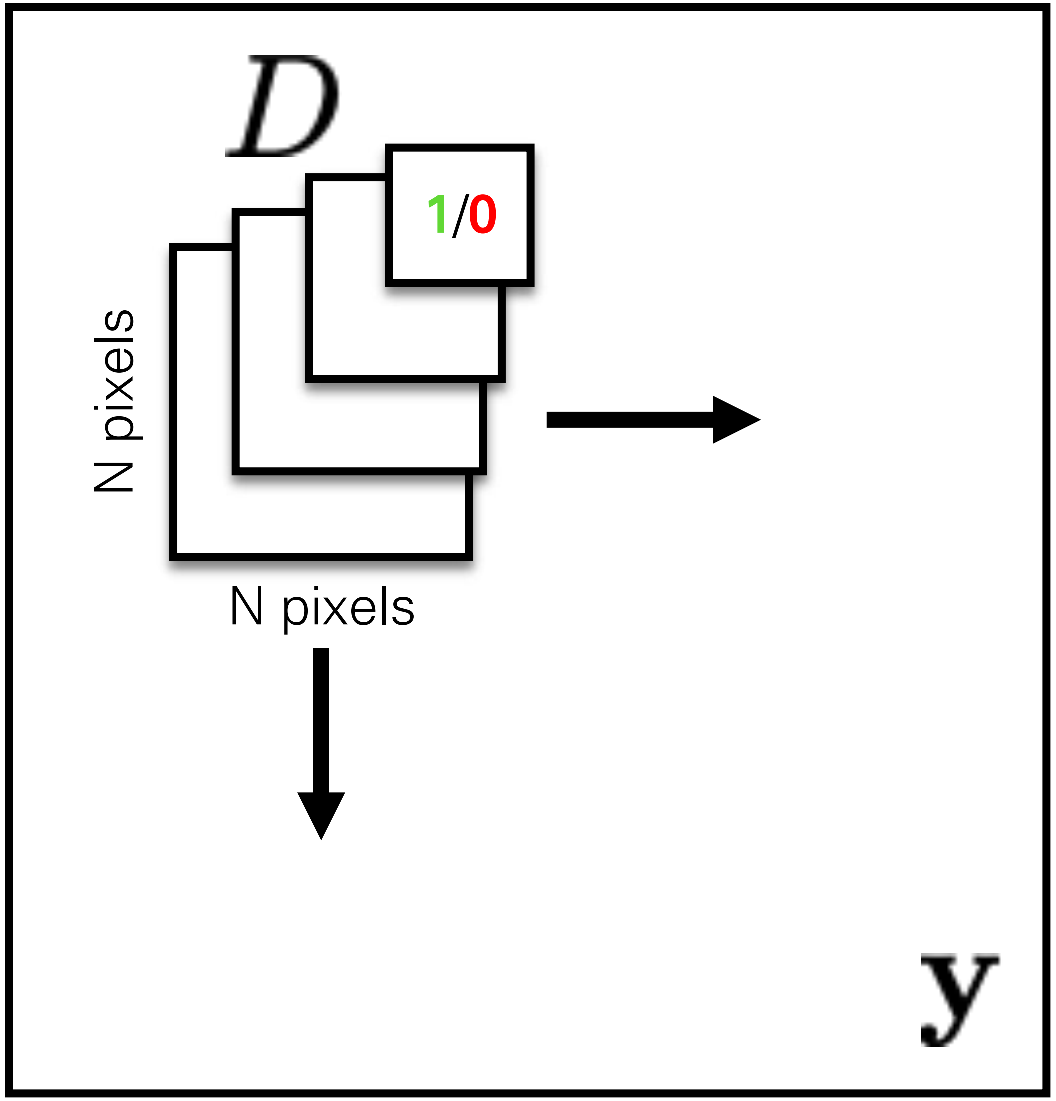




Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

Shrinking the capacity: Patch Discriminator



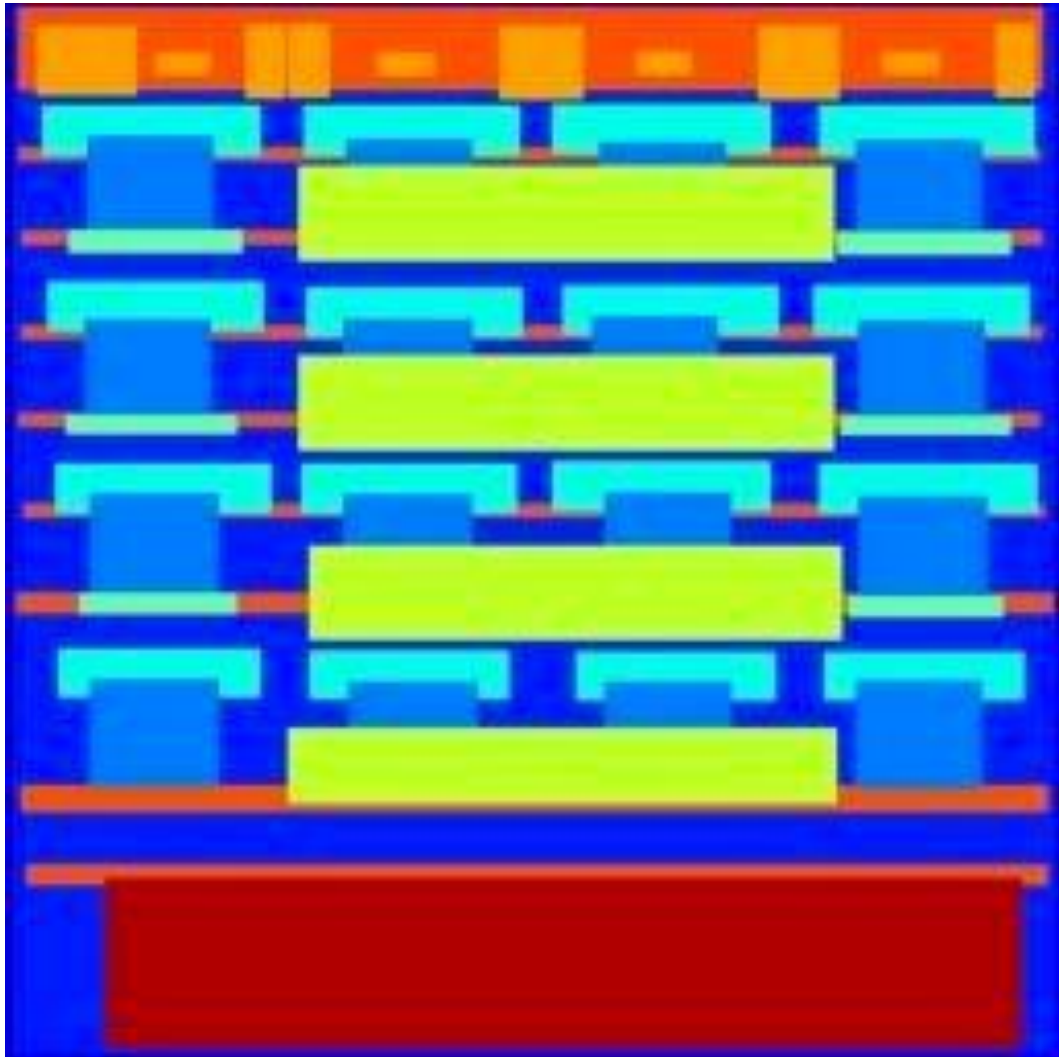
Rather than penalizing if output image looks fake, penalize if each overlapping patch in output looks fake

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

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

Labels → Facades

Input

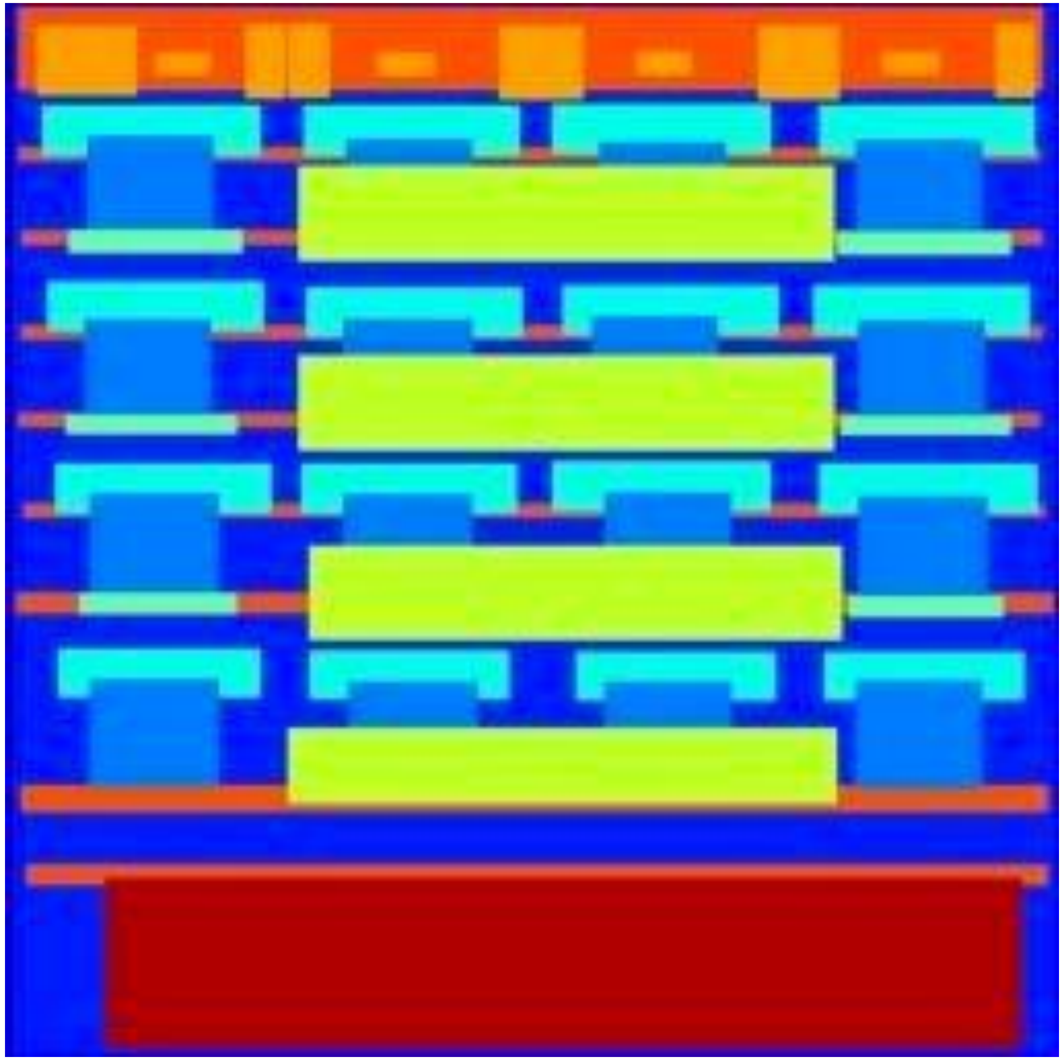


1x1 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

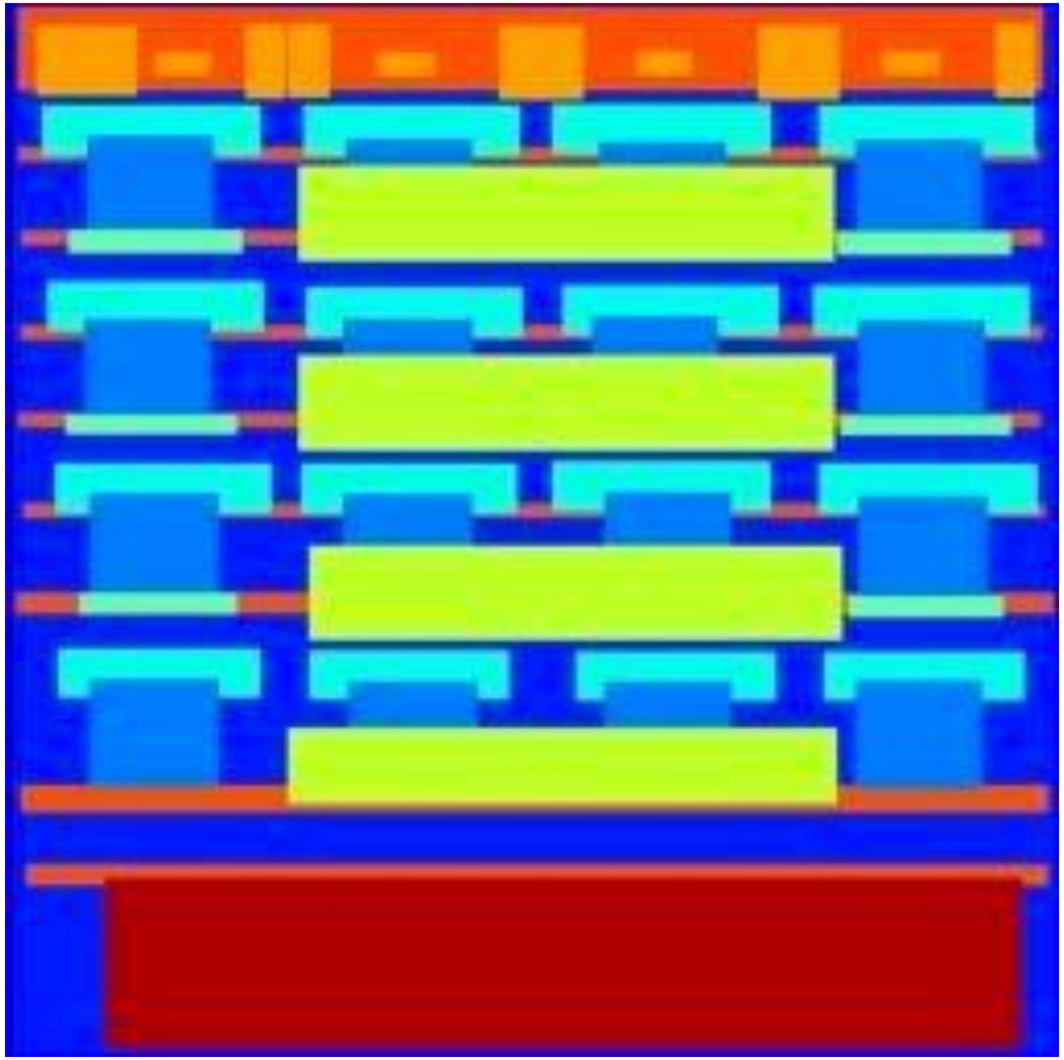


16x16 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

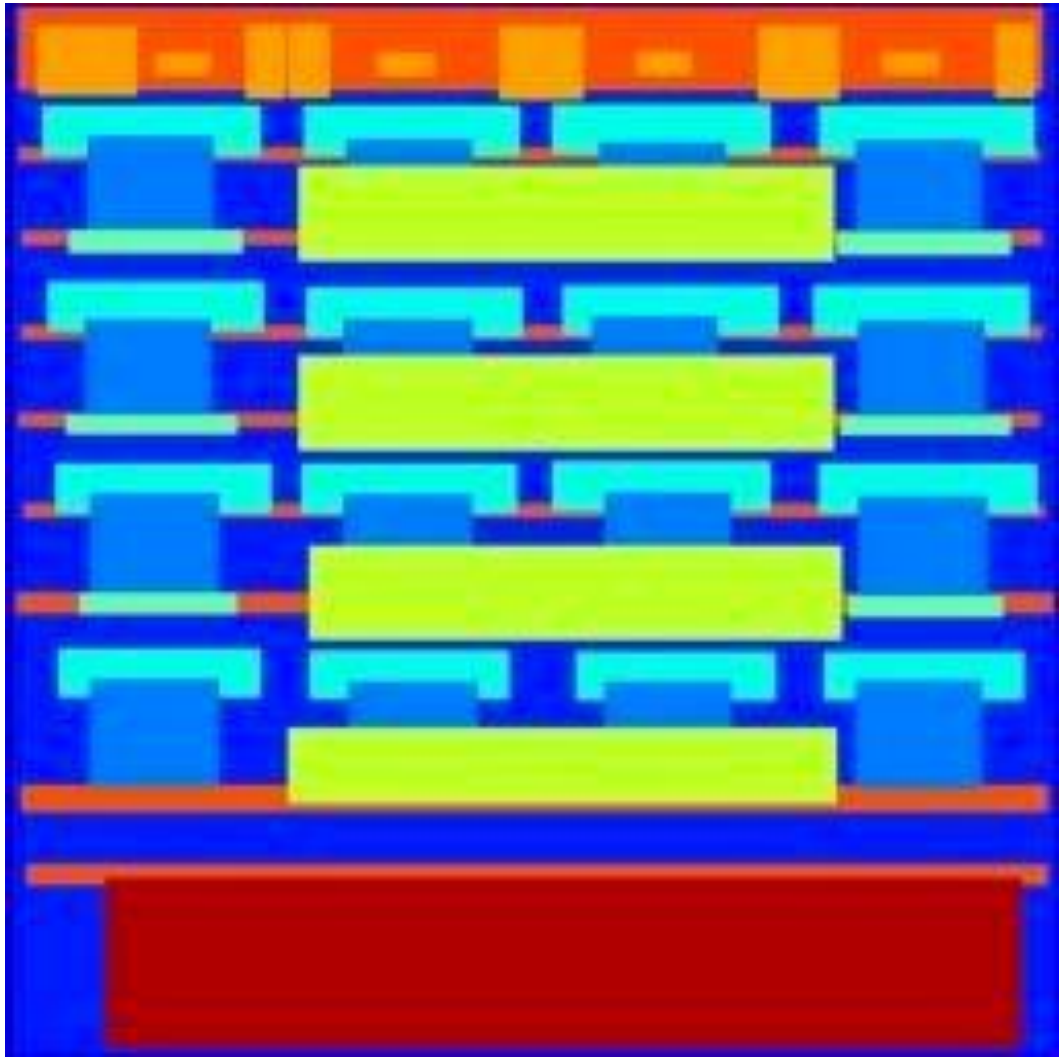


70x70 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

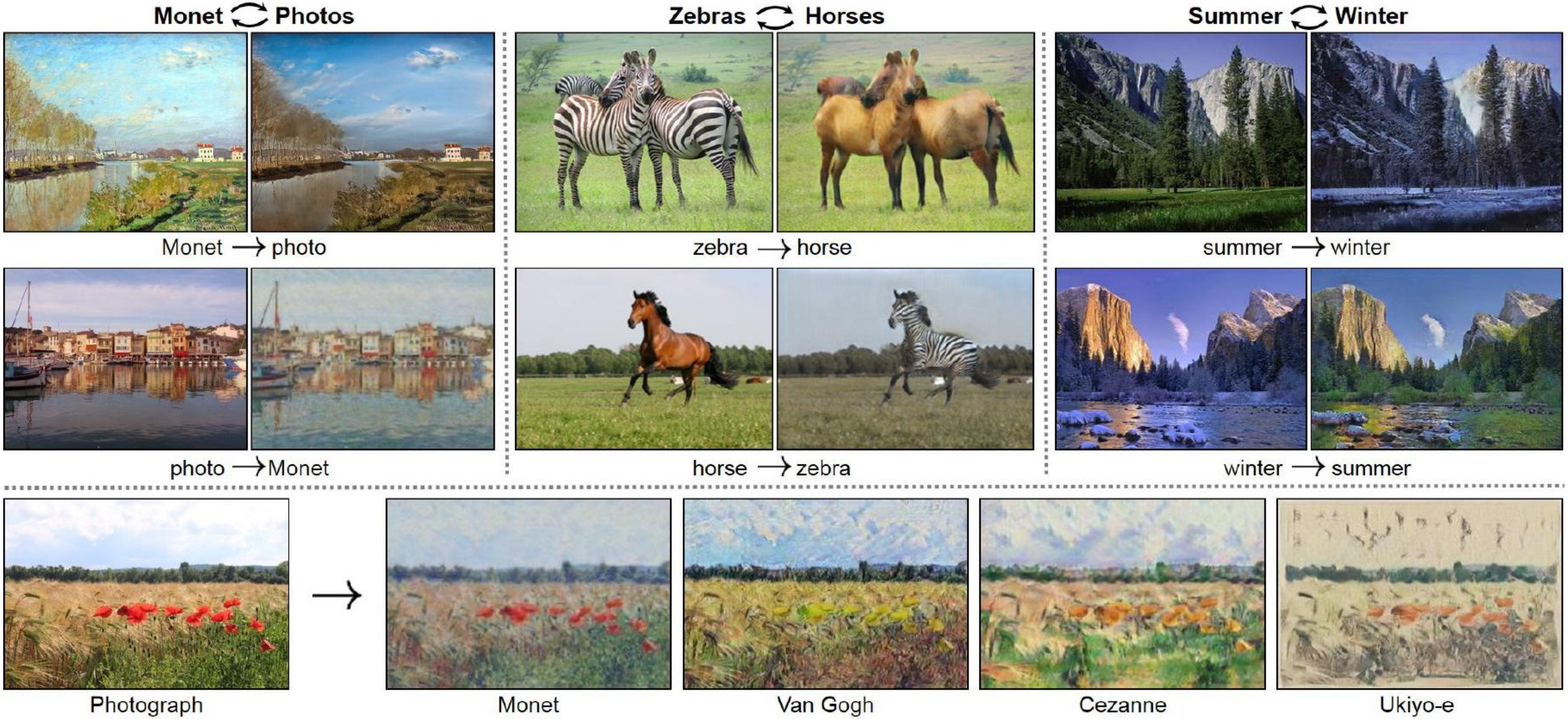
Input



Full image Discriminator

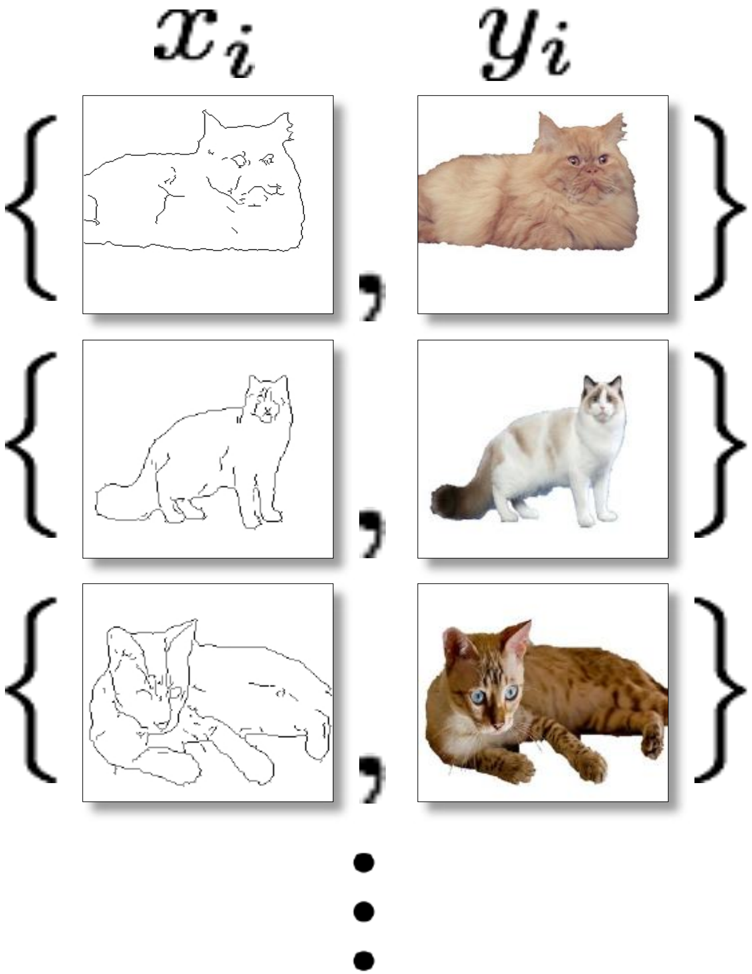
Data from [Tylecek, 2013]

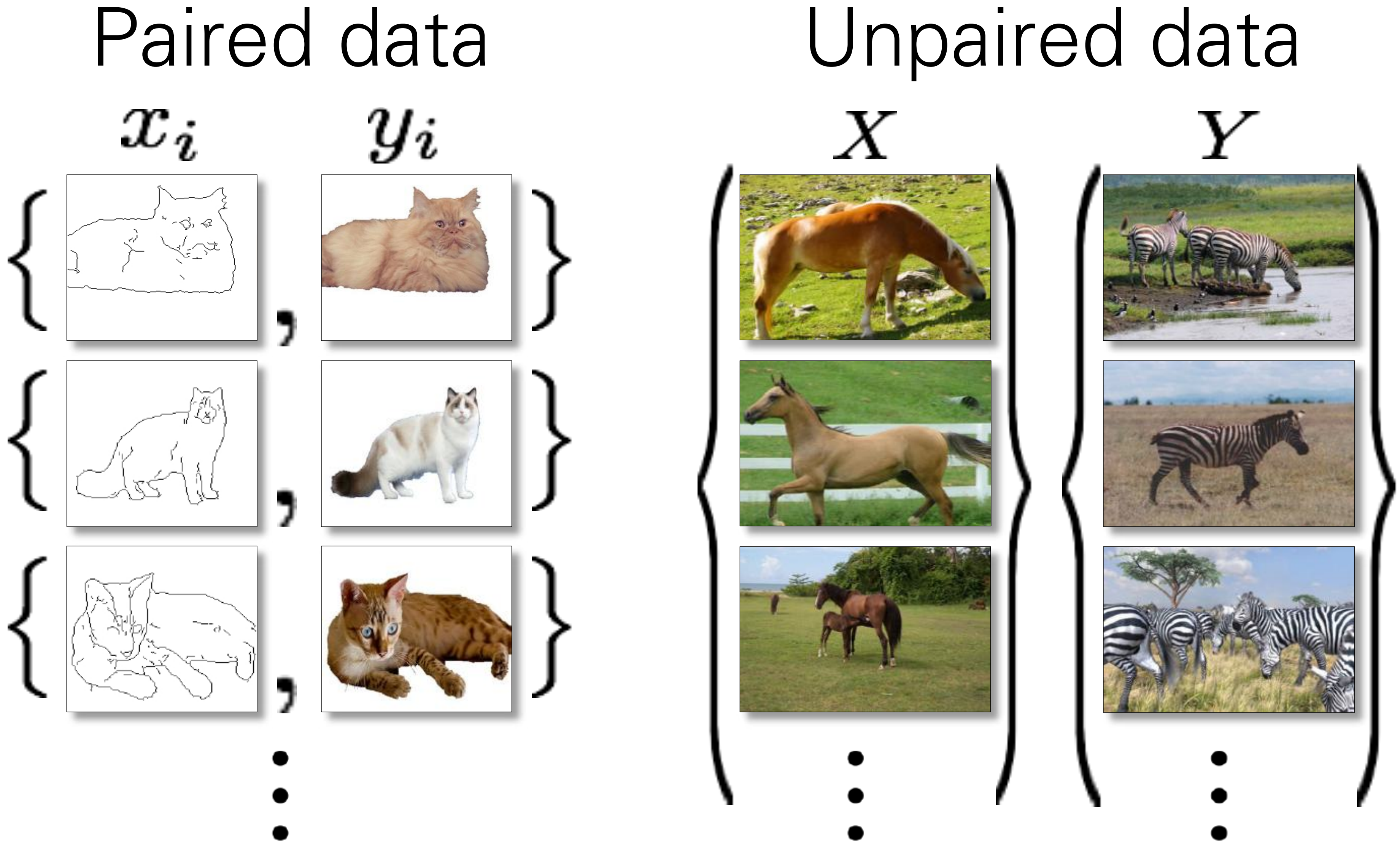
Pix2Pix w/o input-output pairs

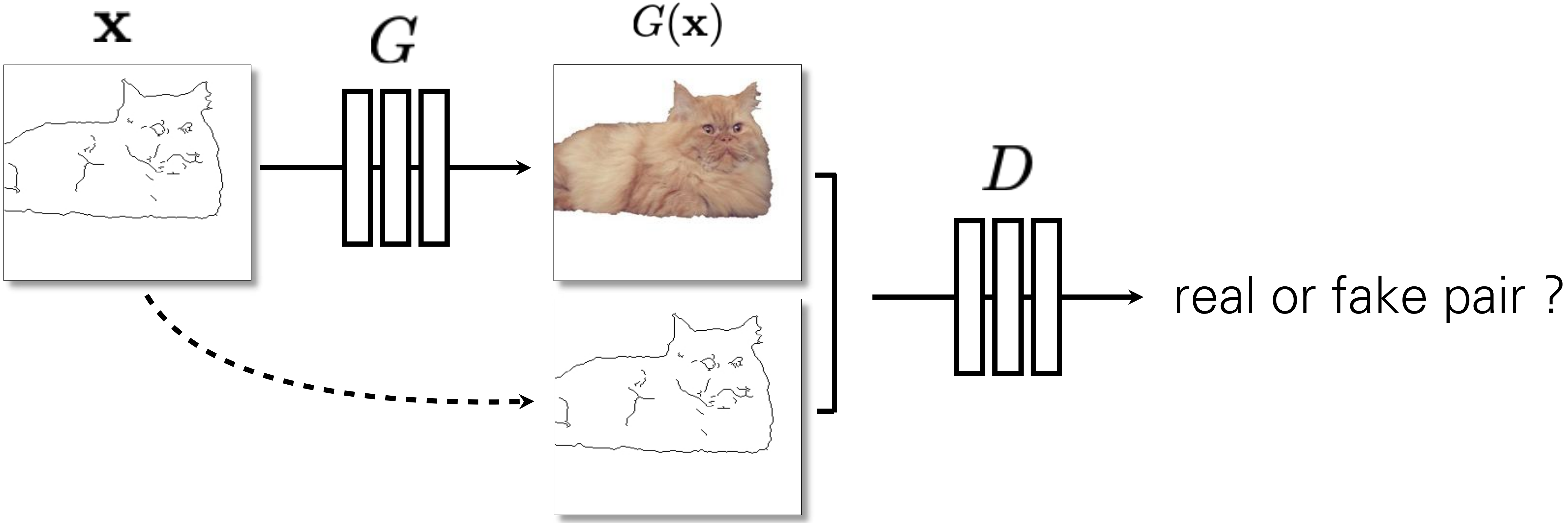


(Zhu et al. 2017)

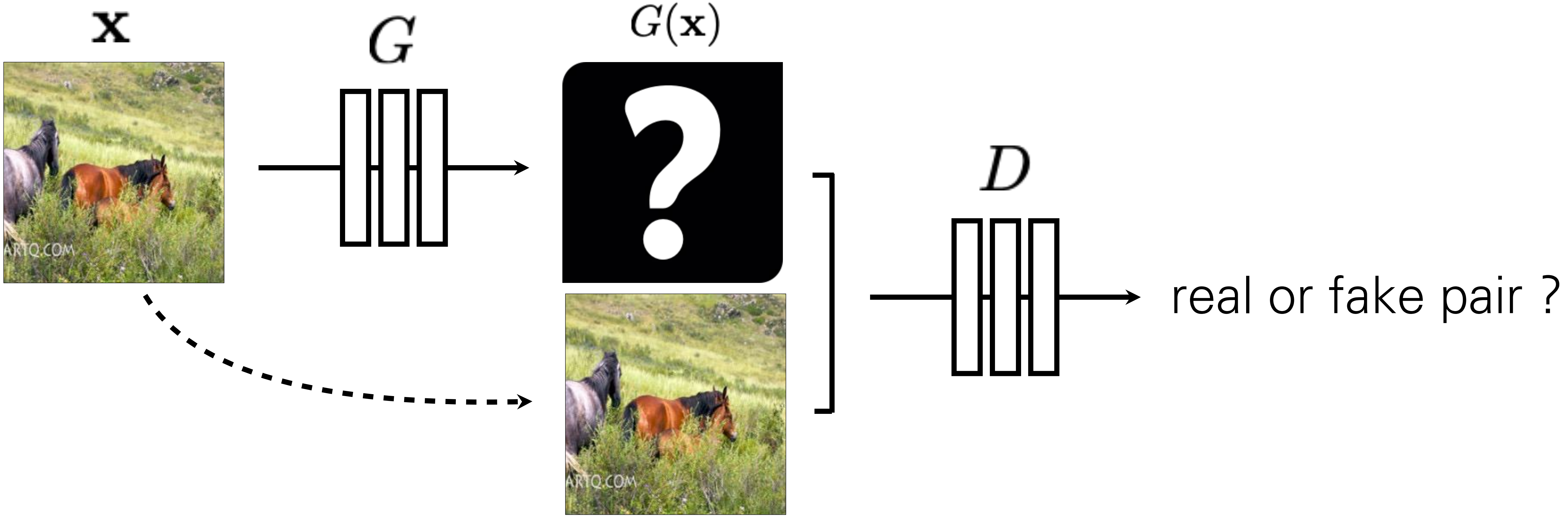
Paired data




Paired data
Unpaired data
x_i
y_i
X
Y

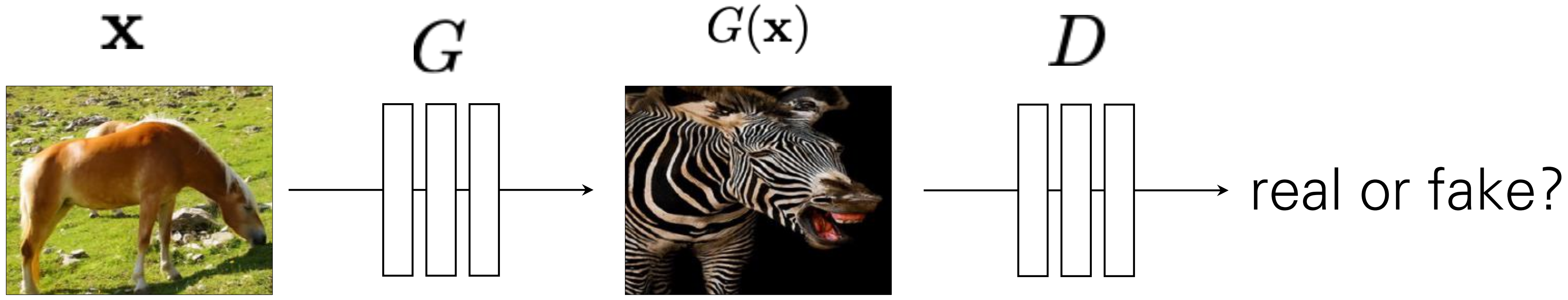


$$\arg\min_{G}\max_{D} \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



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

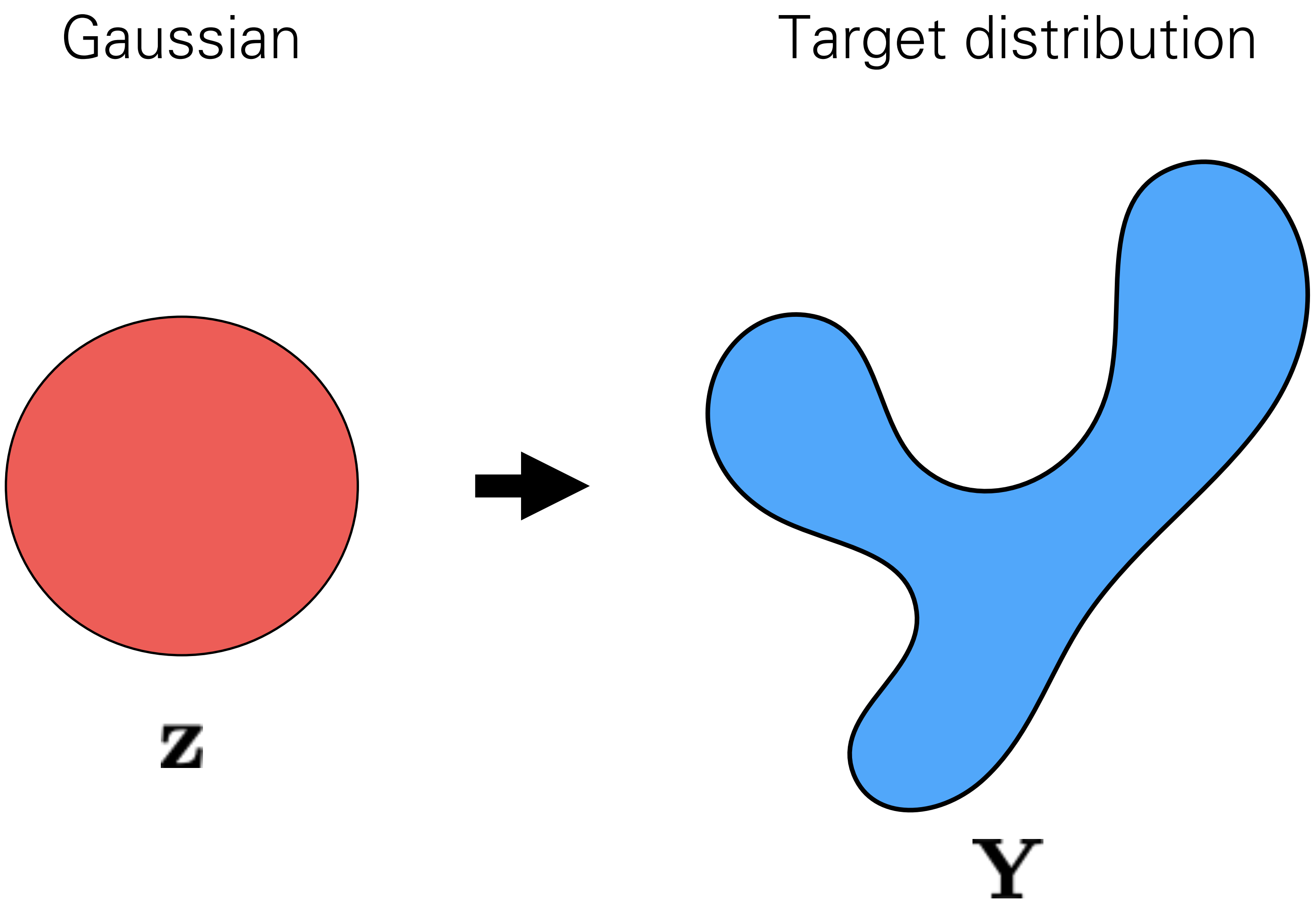
No input-output pairs!

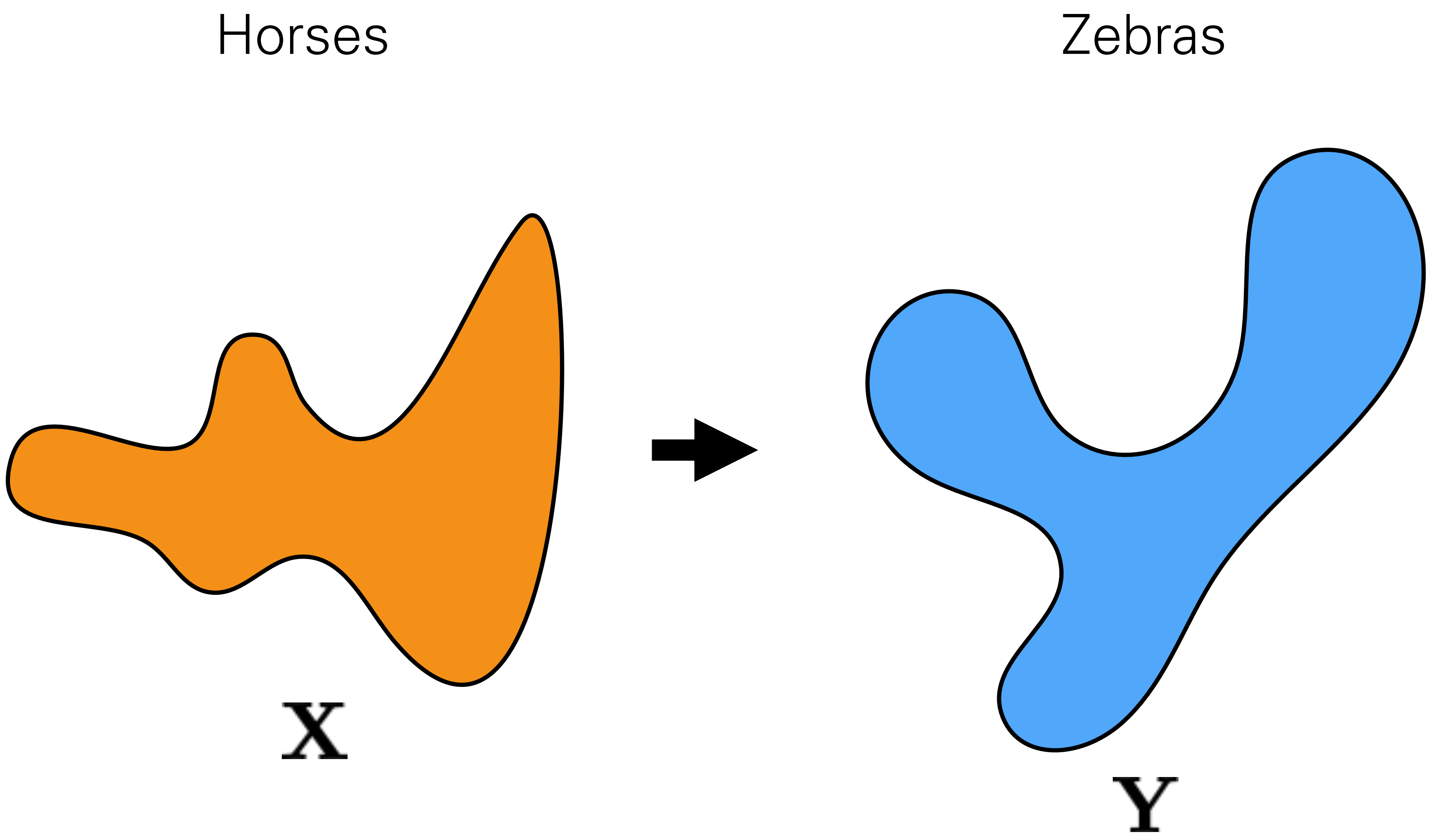


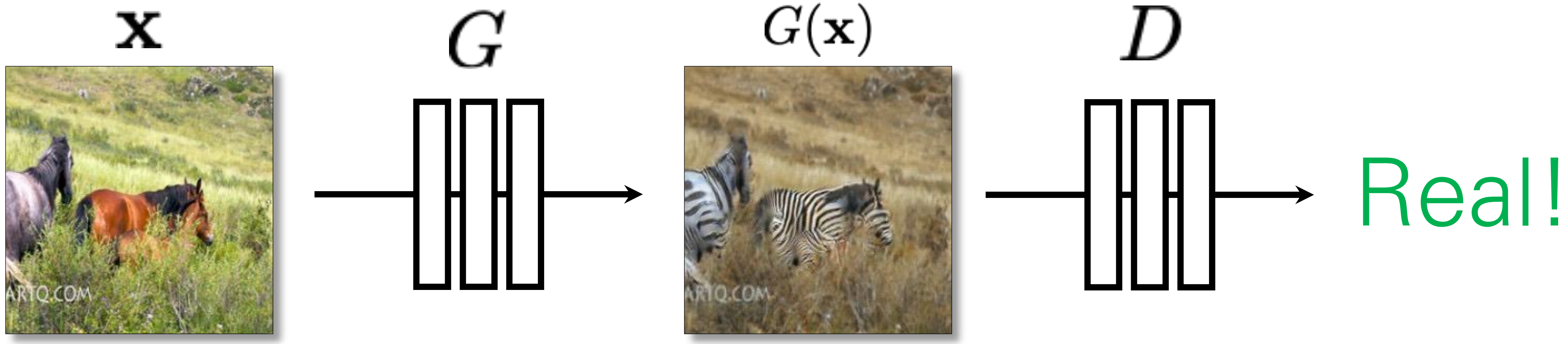
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$

Usually loss functions check if output matches a target instance

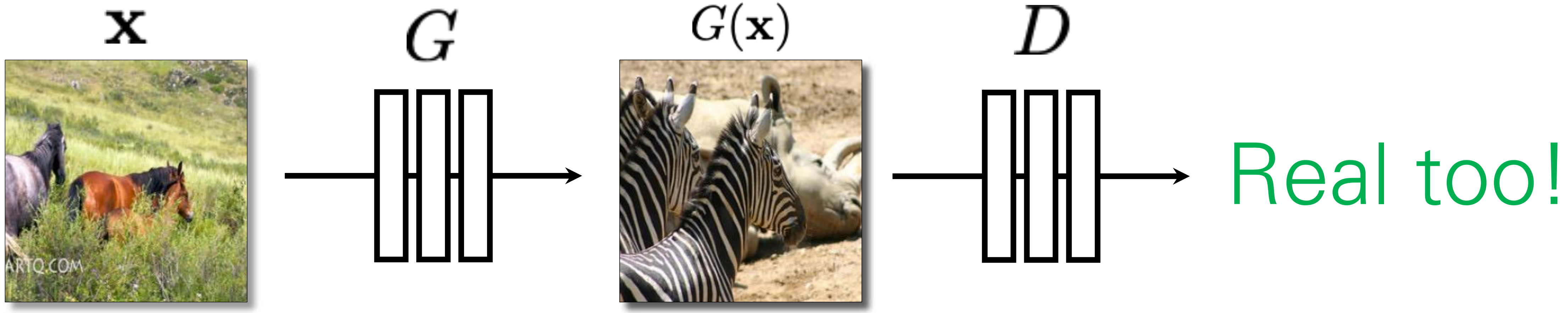
GAN loss checks if output is part of an admissible set


Gaussian
Target distribution
z
Y


Horses
Zebras
X
Y

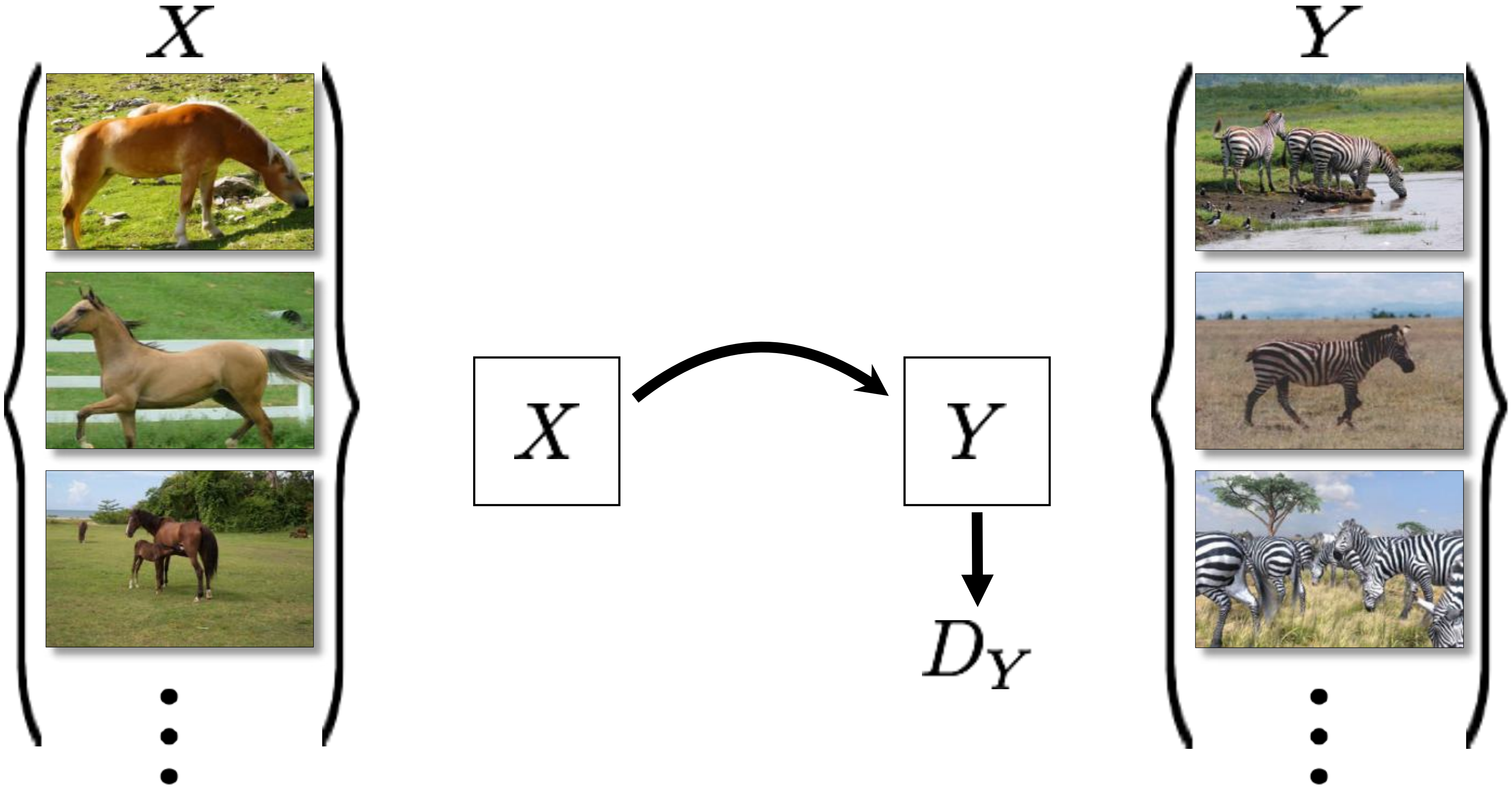


$\mathbf{x}$
G
$G(\mathbf{x})$
D
Real!



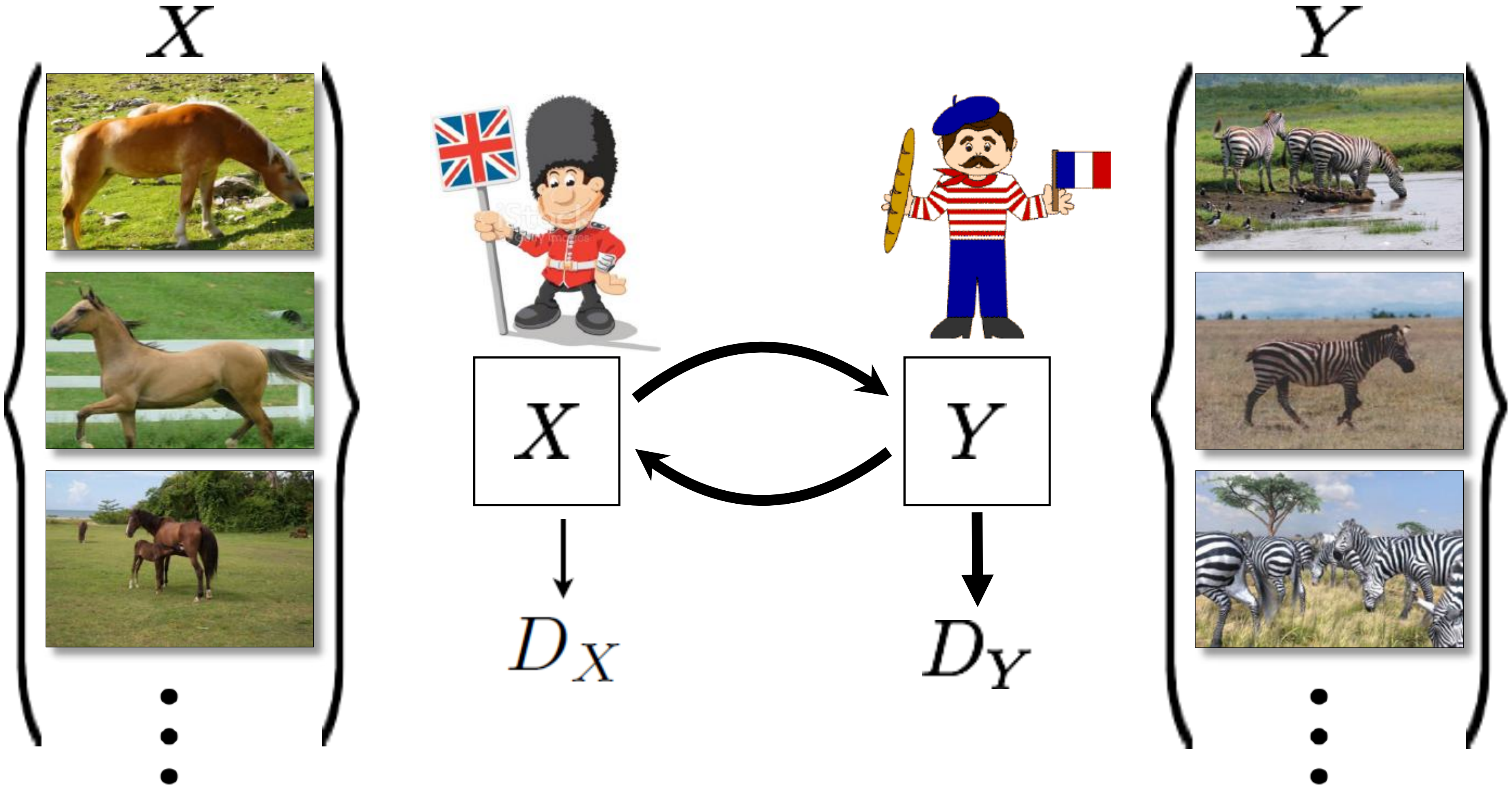
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks

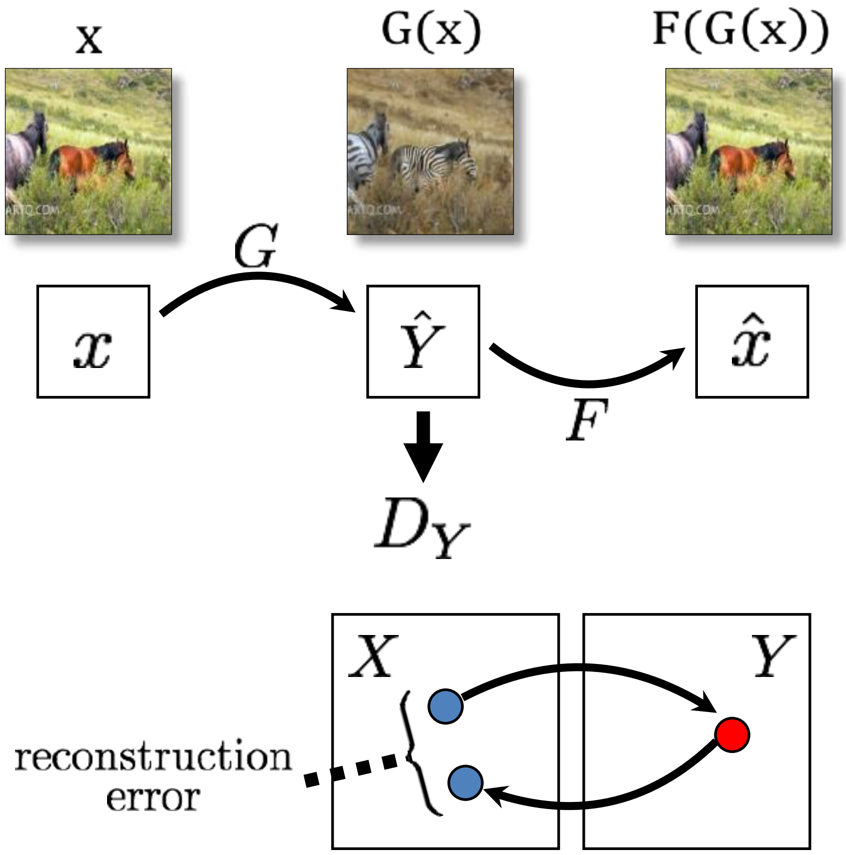


[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks

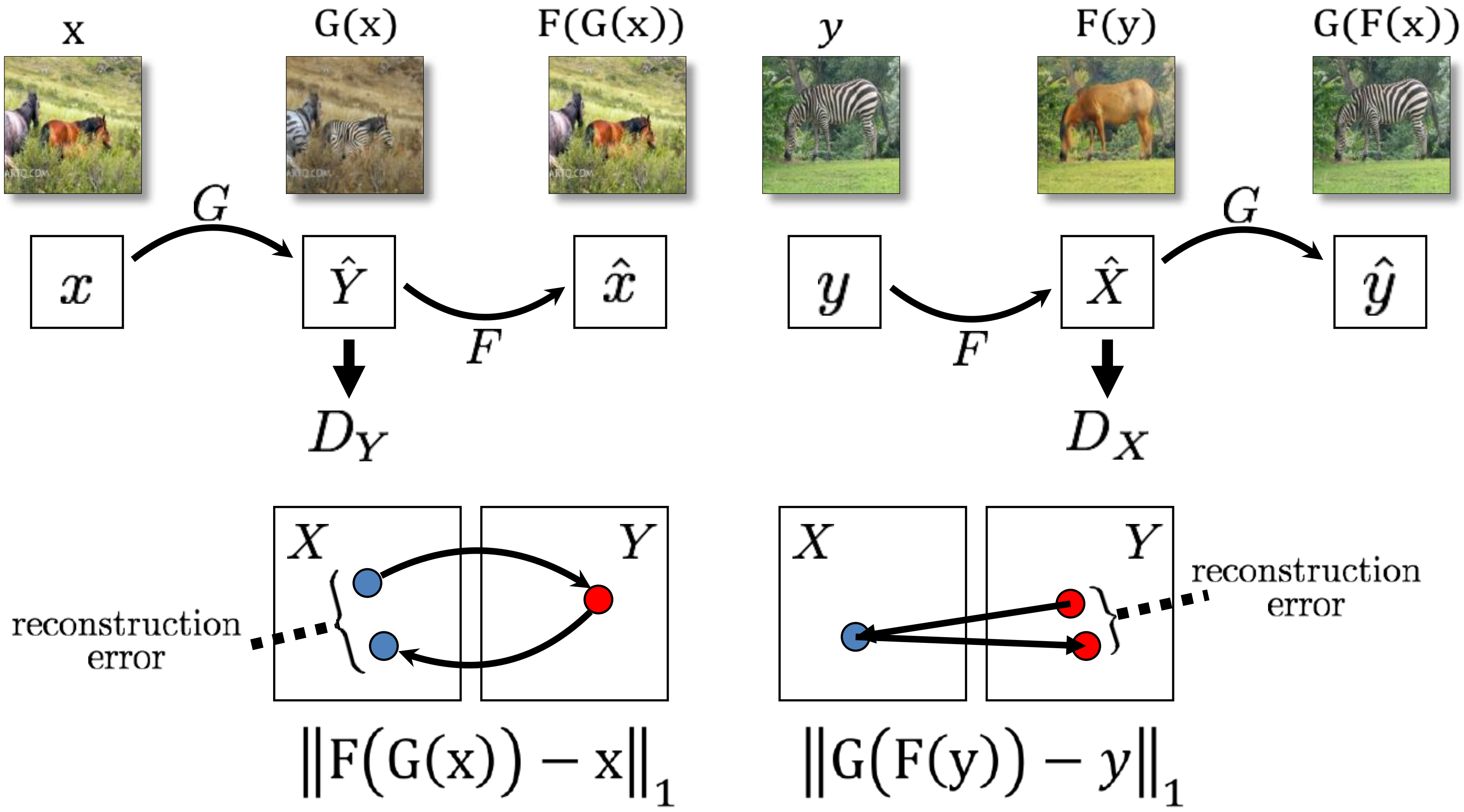


Cycle Consistency Loss



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

Cycle Consistency Loss




Collection Style Transfer

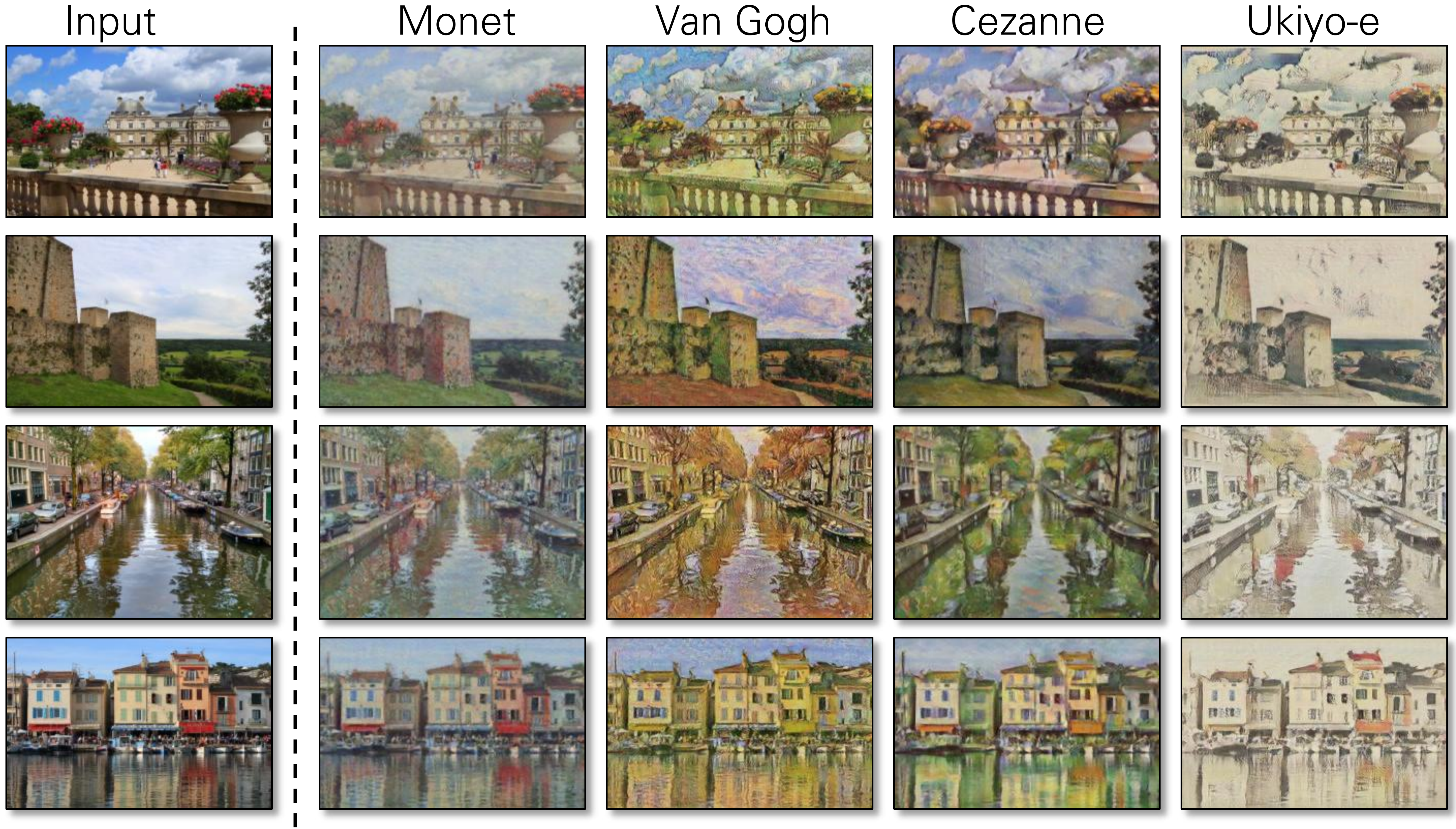
Photograph
@ Alexei Efros

Monet

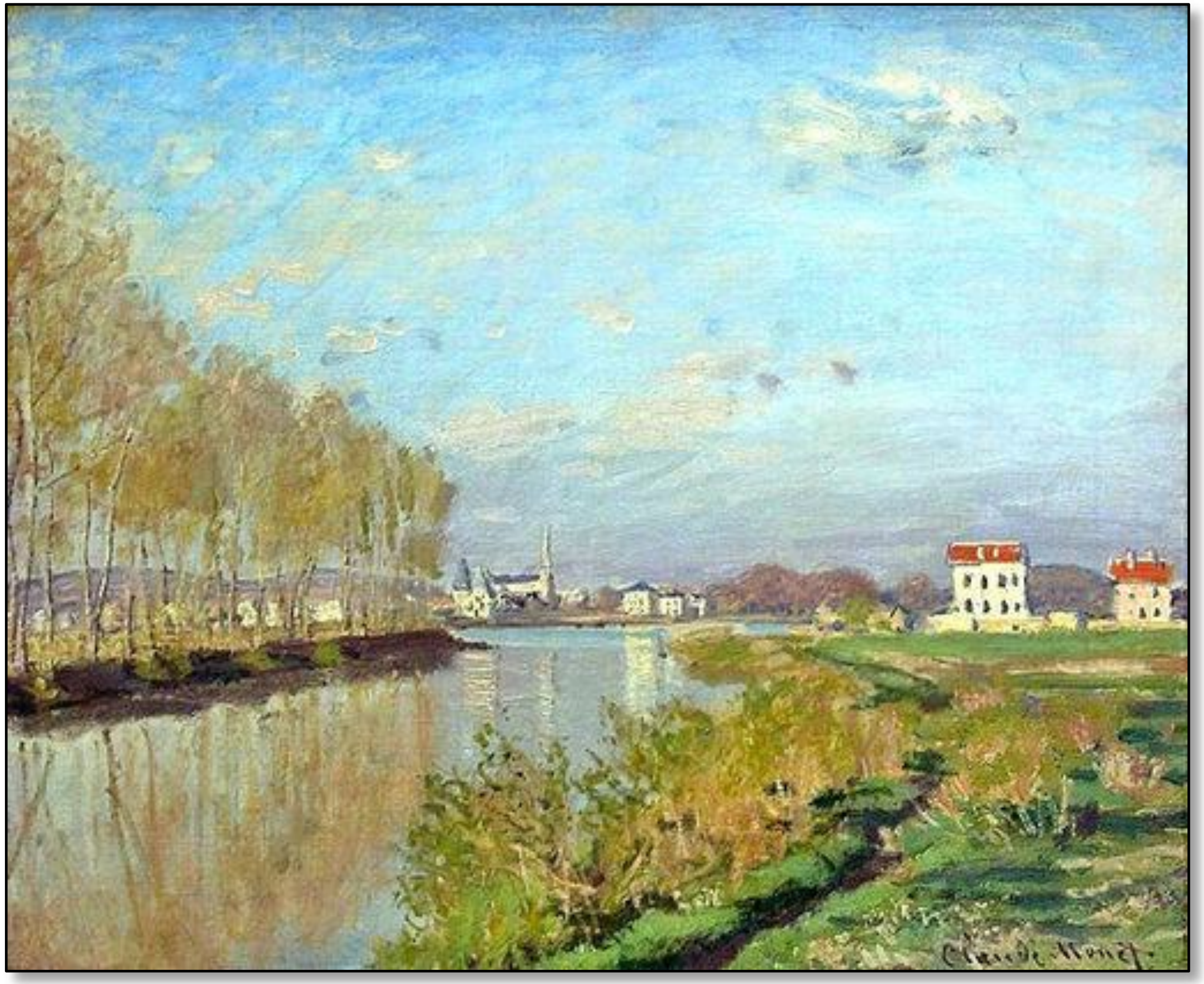
Van Gogh

Cezanne

Ukiyo-e

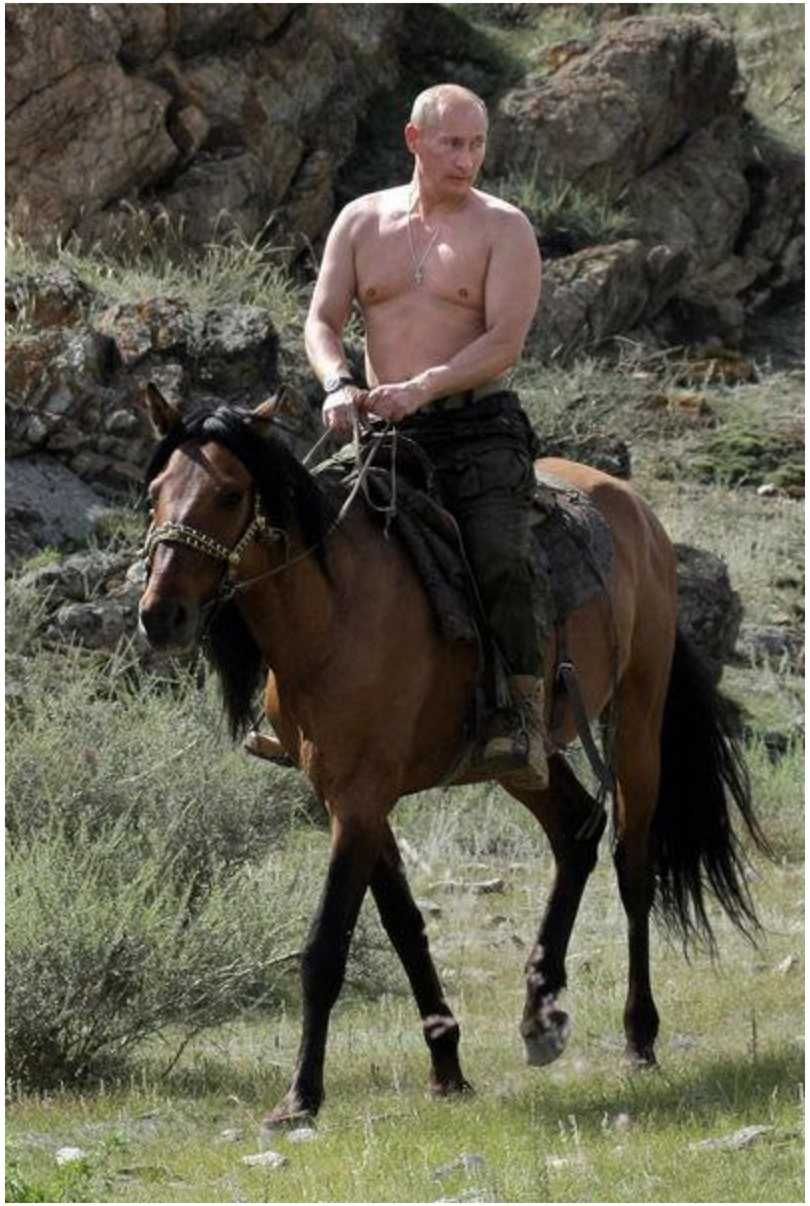

Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Monet's paintings → photos

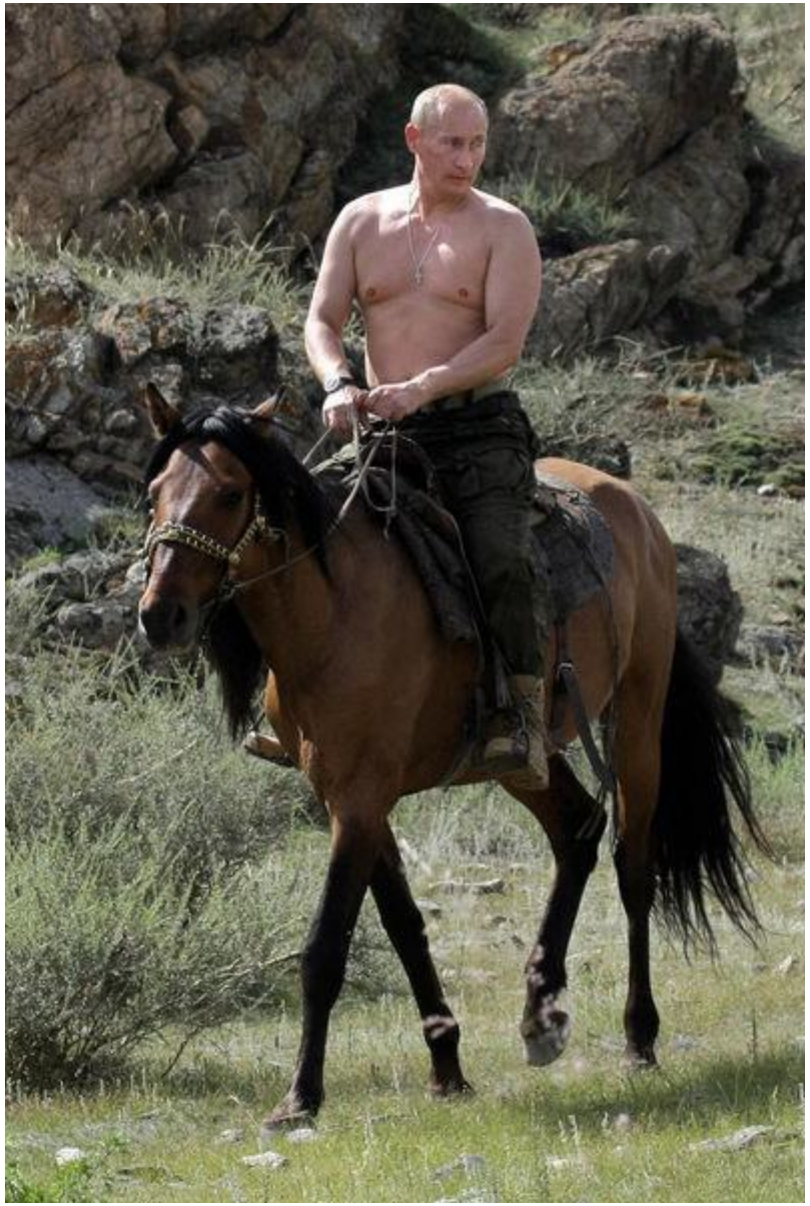


Monet's paintings → photos

Failure case

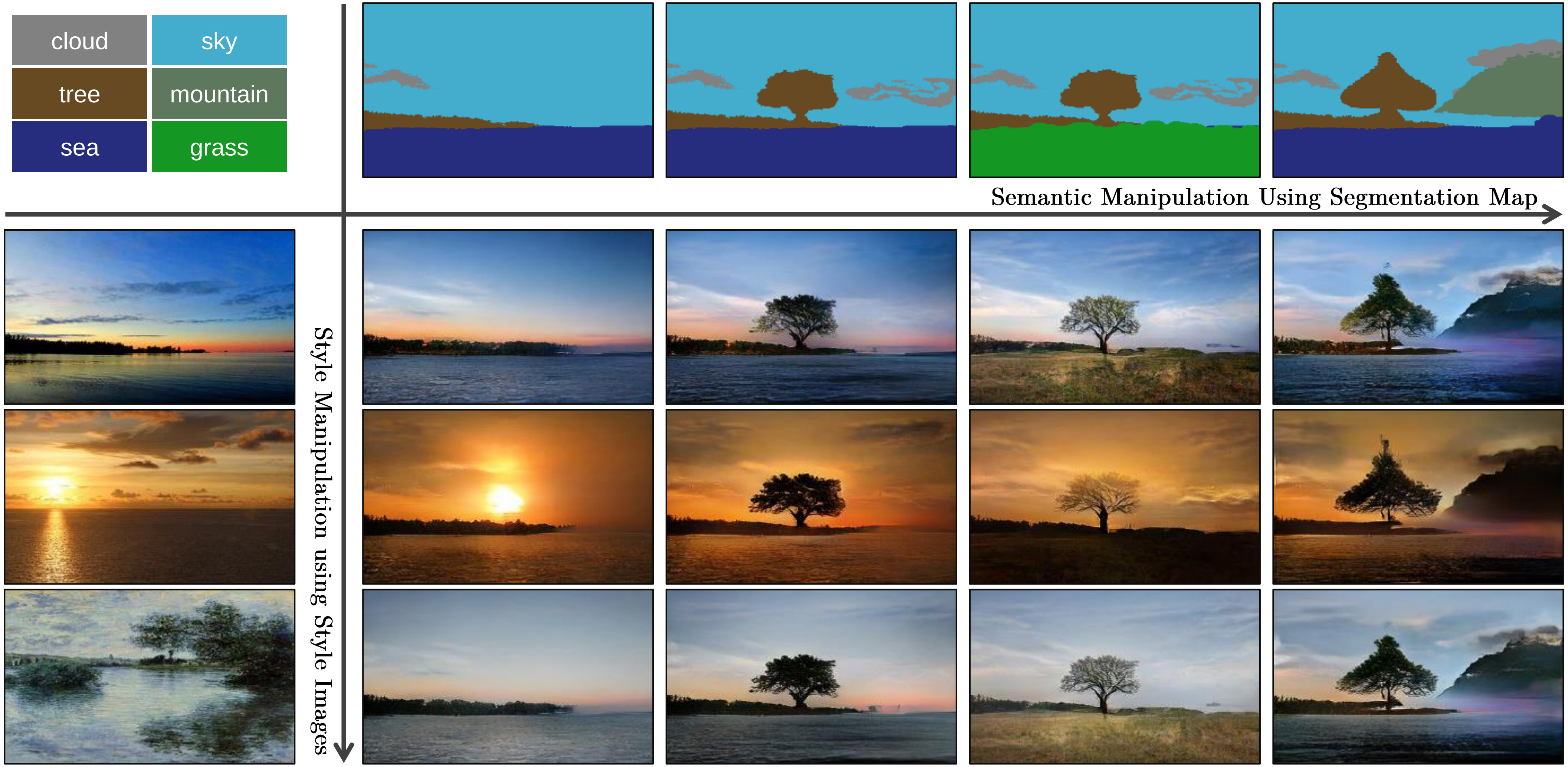


Failure case



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

- Image generation conditioned on semantic layouts



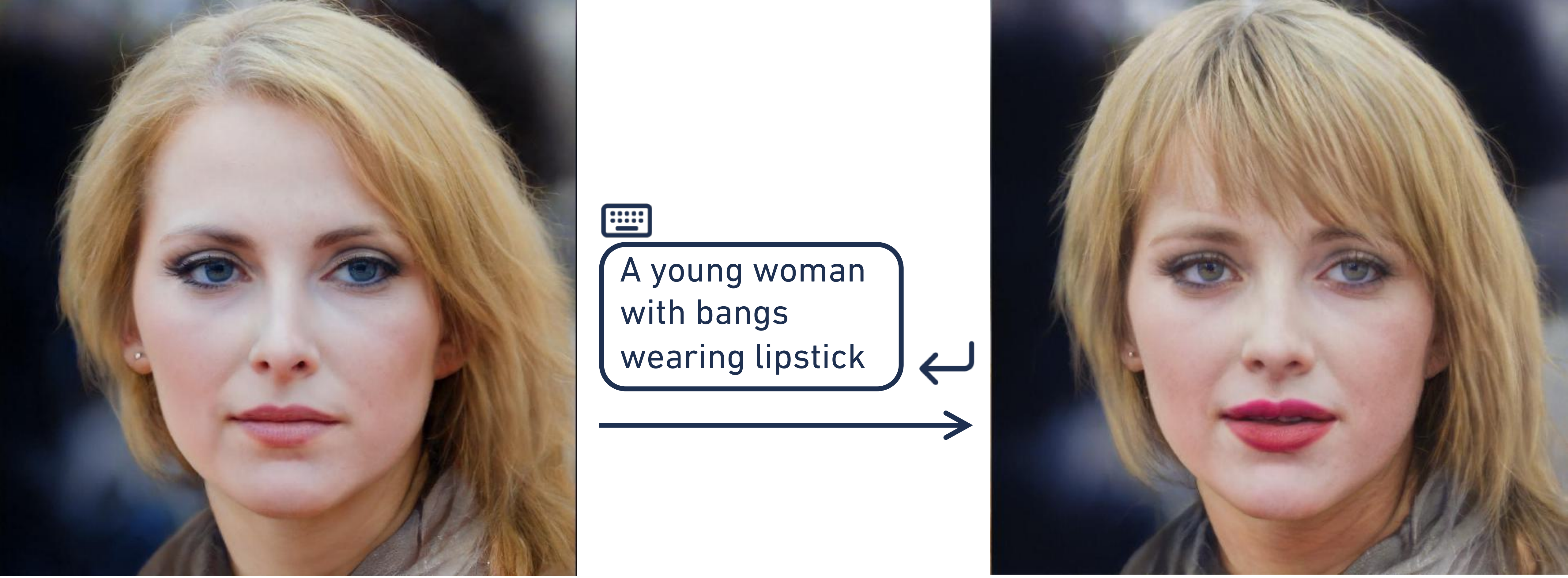
Manipulating Attributes of Natural Scenes via Hallucination.
Levent Karacan, Zeynep Akata, Aykut Erdem & Erkut Erdem.
ACM Trans. on Graphics, Vol. 39, Issue 1, Article 7, February 2020.


snow


night
prediction

Spring
+
Clouds

prediction



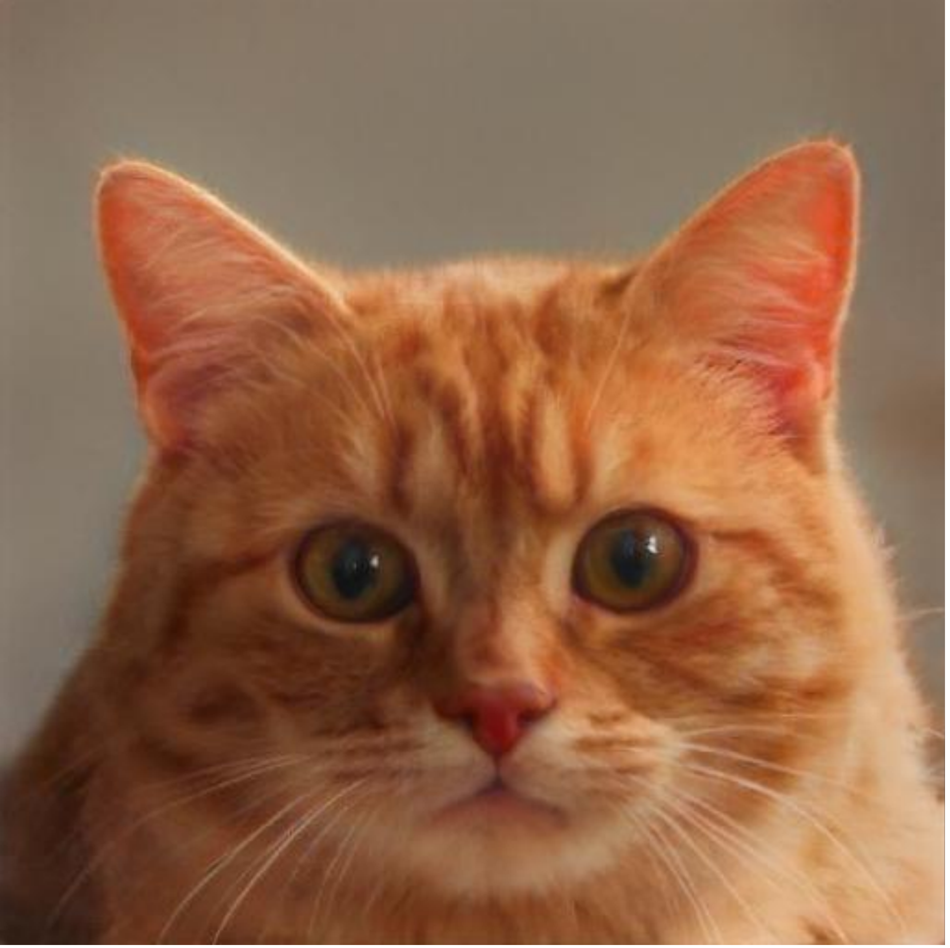
KUIS AI CENTER

Adobe Research

CLIP-Guided StyleGAN Inversion for Text-Driven Real Image Editing.

Canberk Baykal, Abdul Basit Anees, Duygu Ceylan, Aykut Erdem, Erkut Erdem, Deniz Yuret

ACM Transactions on Graphics, 2023



An old and grumpy British shorthair

Adobe Research

CLIP-Guided StyleGAN Inversion for Text-Driven Real Image Editing.

Canberk Baykal, Abdul Basit Anees, Duygu Ceylan, Aykut Erdem, Erkut Erdem, Deniz Yuret

ACM Transactions on Graphics, 2023

KUIS AI CENTER

Adobe Research

VidStyleODE: Disentangled Video Editing via StyleGAN and NeuralODE.

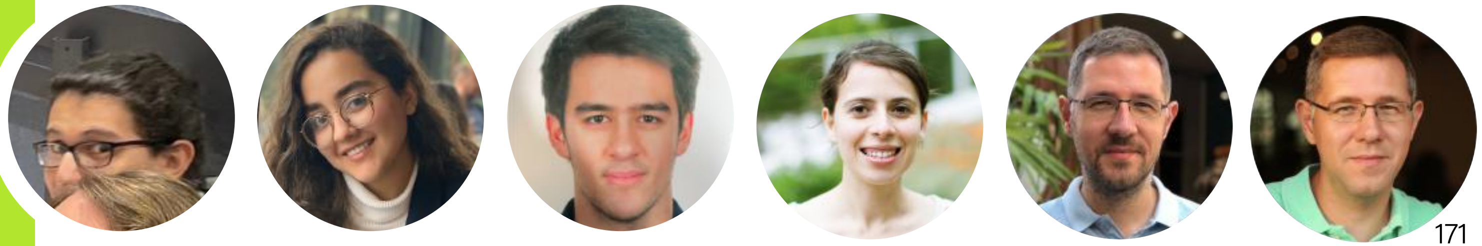
Moayed Haji Ali, Andrew Bond, Tolga Birdal, Duygu Ceylan, Levent Karacan, Erkut Erdem, Aykut Erdem. ICCV 2023

KUIS AI CENTER

Adobe Research

Audio-based Image Editing and Generation using Latent Diffusion Models

Burak Can Biner, Farrin Marouf Sofian,
Umur Berkay Karakaş, Duygu Ceylan, Erkut Erdem,
Aykut Erdem. In progress

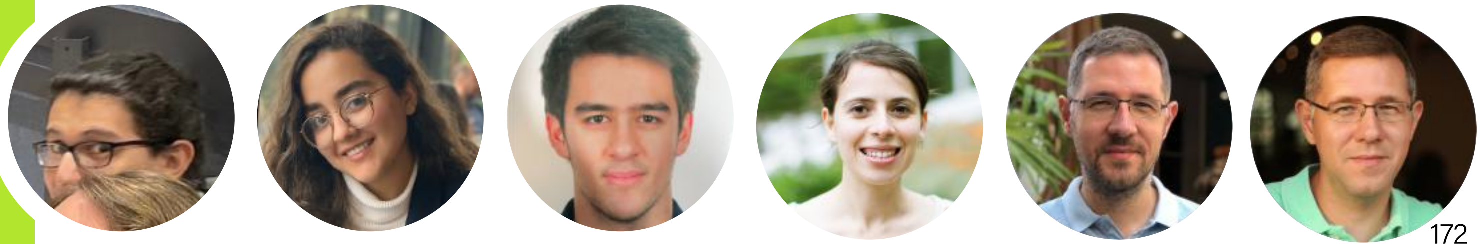


KUIS AI CENTER

Adobe Research

Audio-based Image Editing and Generation using Latent Diffusion Models

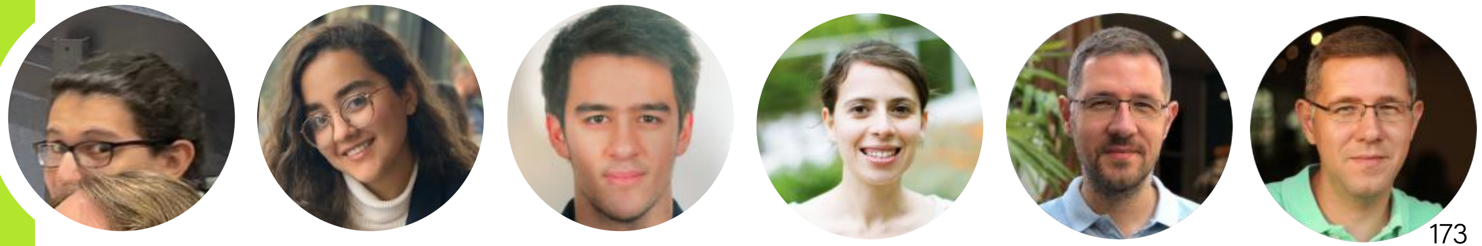
Burak Can Biner, Farrin Marouf Sofian,
Umur Berkay Karakaş, Duygu Ceylan, Erkut Erdem,
Aykut Erdem. In progress



Adobe Research

Audio-based Image Editing and Generation using Latent Diffusion Models

Burak Can Biner, Farrin Marouf Sofian,
Umur Berkay Karakaş, Duygu Ceylan, Erkut Erdem,
Aykut Erdem. In progress



Next lecture:
Autoregressive and Flow Models