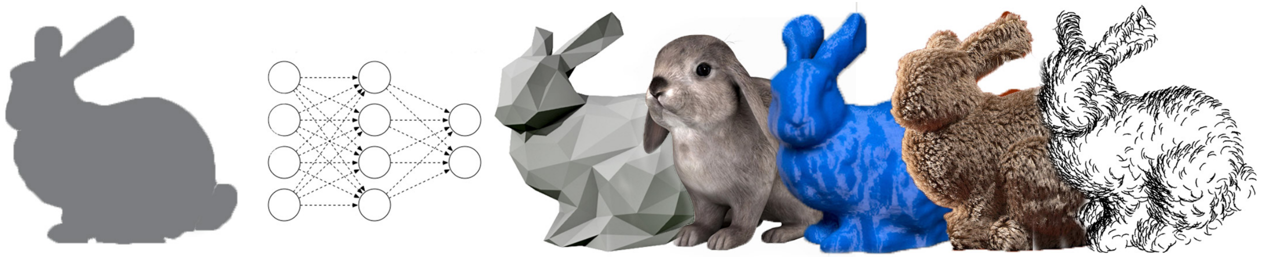
thrive SIGGRAPH2019
LOS ANGELES • 28 JULY – 1 AUGUST



CreativeAI: Deep Learning for Computer Graphics

Unsupervised Learning in CG

Niloy Mitra	Iasonas Kokkinos	Paul Guerrero	Vladimir Kim	Nils Thuerey	Leonidas Guibas
UCL/Adobe	UCL/Ariel AI	UCL/Adobe	Adobe	TUM	Stanford/FAIR

Timetable

		Niloy	Iasonas	Paul	Nils	Leonidas
Introduction	9:00	X				
Neural Network Basics	~9:15		X			
Supervised Learning in CG	~9:50	X				
Unsupervised Learning in CG	~10:20			X		
Learning on Unstructured Data	~10:55					X
Learning for Simulation/Animation	~11:35				X	
Discussion	12:05	X	X	X	X	X

Unsupervised Learning

There is no direct ground truth for the quantity of interest

Focus on **generative** models:

- Variational Autoencoders (VAEs)
- Normalizing Flows
- Autoregressive Models (slides only)
- Generative Adversarial Networks (GANs)

Generative Models

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset

Generated

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks,* Karras et al.

Generative Models

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset

Generated

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks*, Karras et al.

Generative Models

...

$$p_{\text{data}}(x) \approx \boxed{p_\theta(x)}$$

Generative model
with parameters θ

...

How do we measure the similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?

Which model?

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks*, Karras et al.

Generative Models

How do we measure the similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?

1) Likelihood of data samples in $p_\theta(x)$

Variational Autoencoders (VAEs)

Normalizing Flows

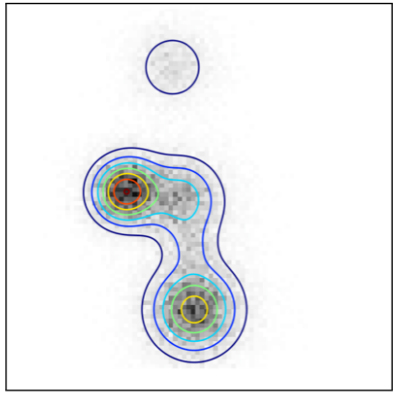
Autoregressive Models

2) Adversarial game

Generative Adversarial Networks (GANs)

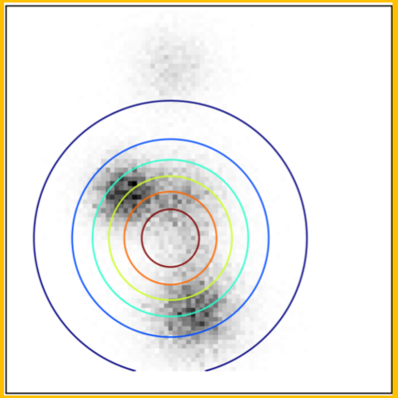
Generative Models

How do we measure the similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?



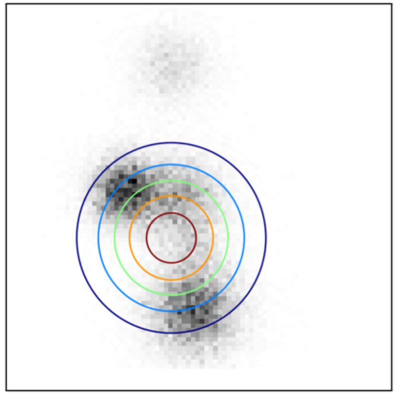
$p_{\text{data}}(x)$

1) Likelihood of data samples in $p_\theta(x)$



$\approx KL(p_{\text{data}} \,||\, p_\theta)$

2) Adversarial game



$\approx JS(p_{\text{data}} \,||\, p_\theta)$

Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár

Likelihood-Based Models: Two Goals

Sample?
Evaluate?

1) Sample from the generative model

$x \sim p_\theta$

Generative model
with parameters θ

2) Evaluate the likelihood of a given sample in the model

$p_\theta(x)$

Generative model
with parameters θ

x

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks*, Karras et al.

The Feature Space

Data distribution in 2D feature space (colors are class labels)

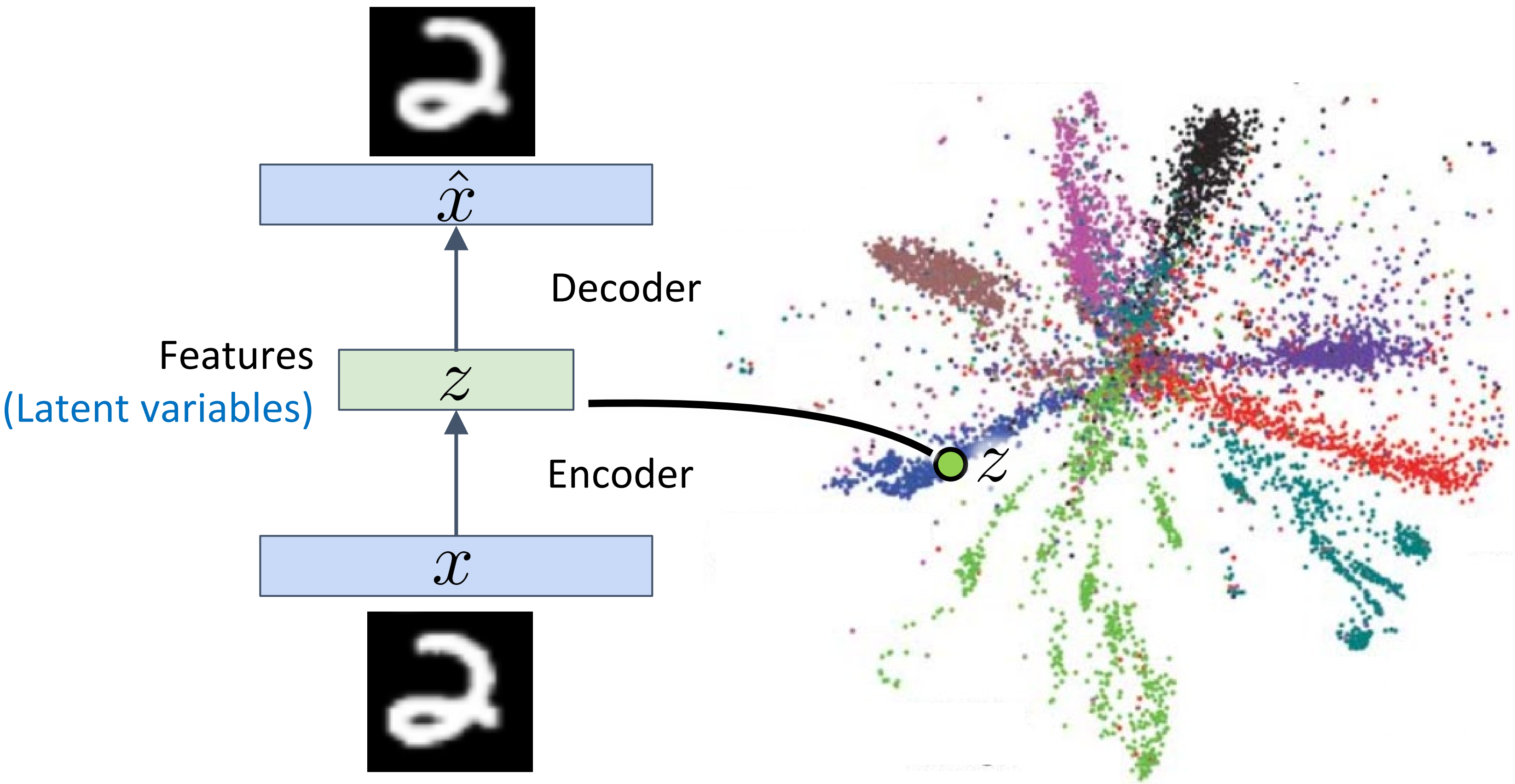
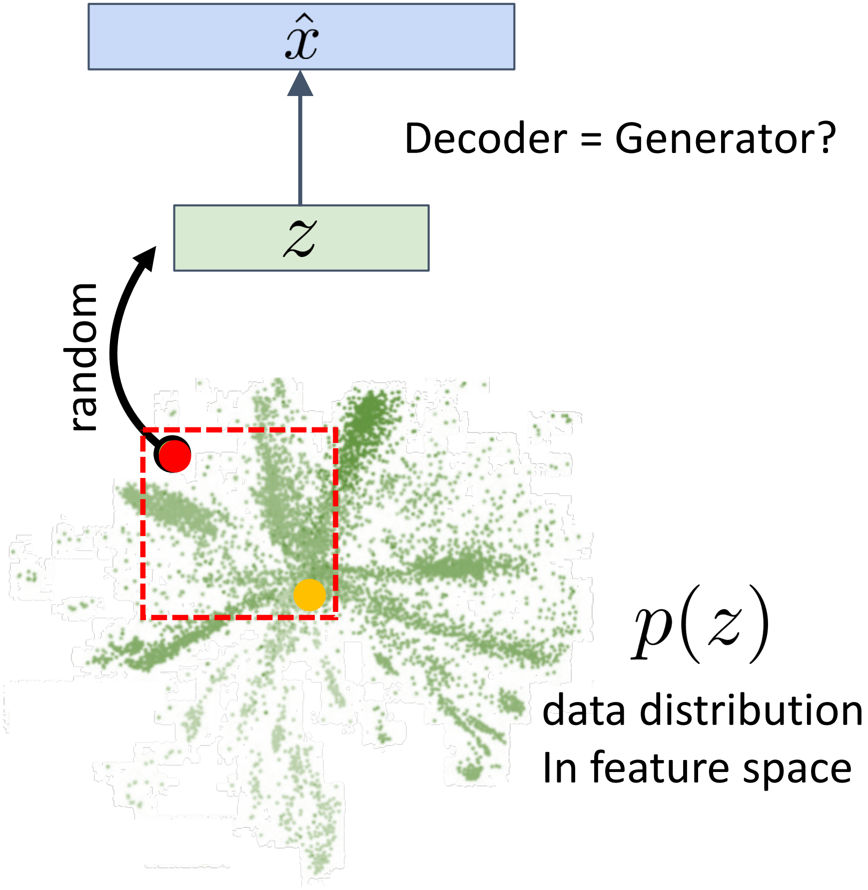


Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

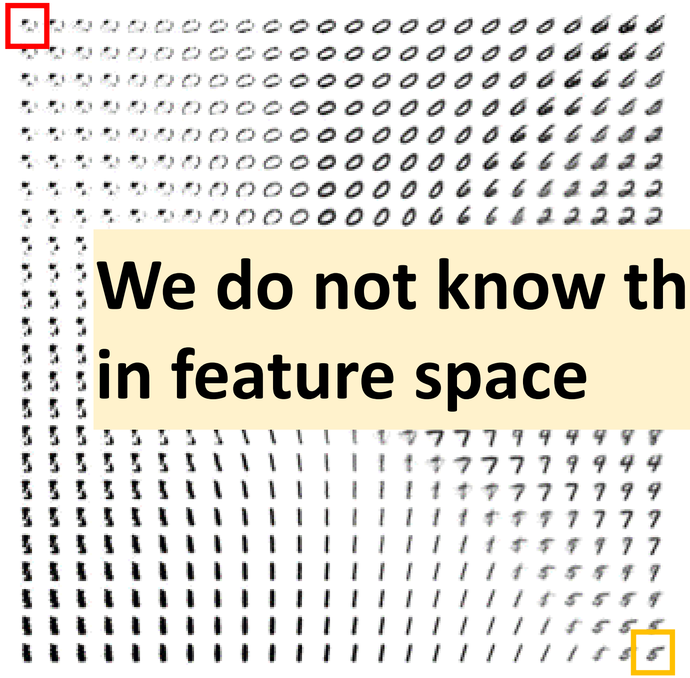
Autoencoders as Generative Models?

Sample?
Evaluate?

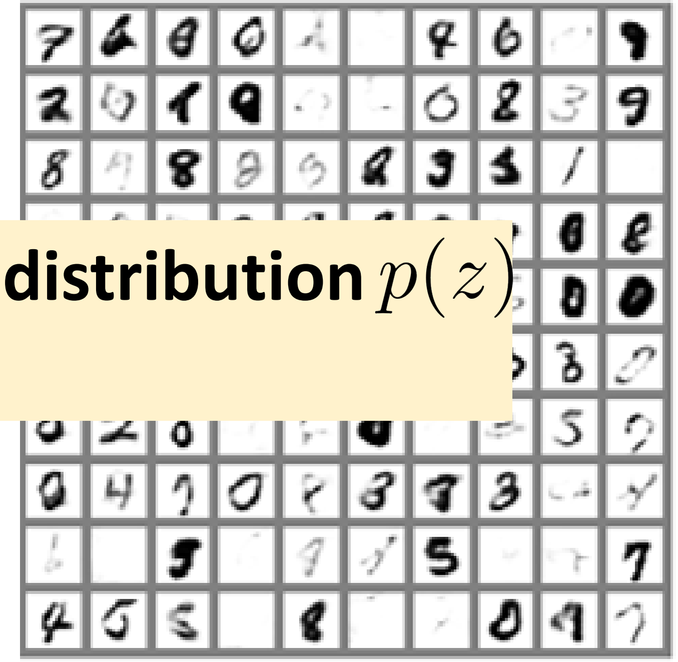


- Is a trained decoder a generative model?
- Can we generate a new sample $x \sim p_\theta$?

sample grid



random samples

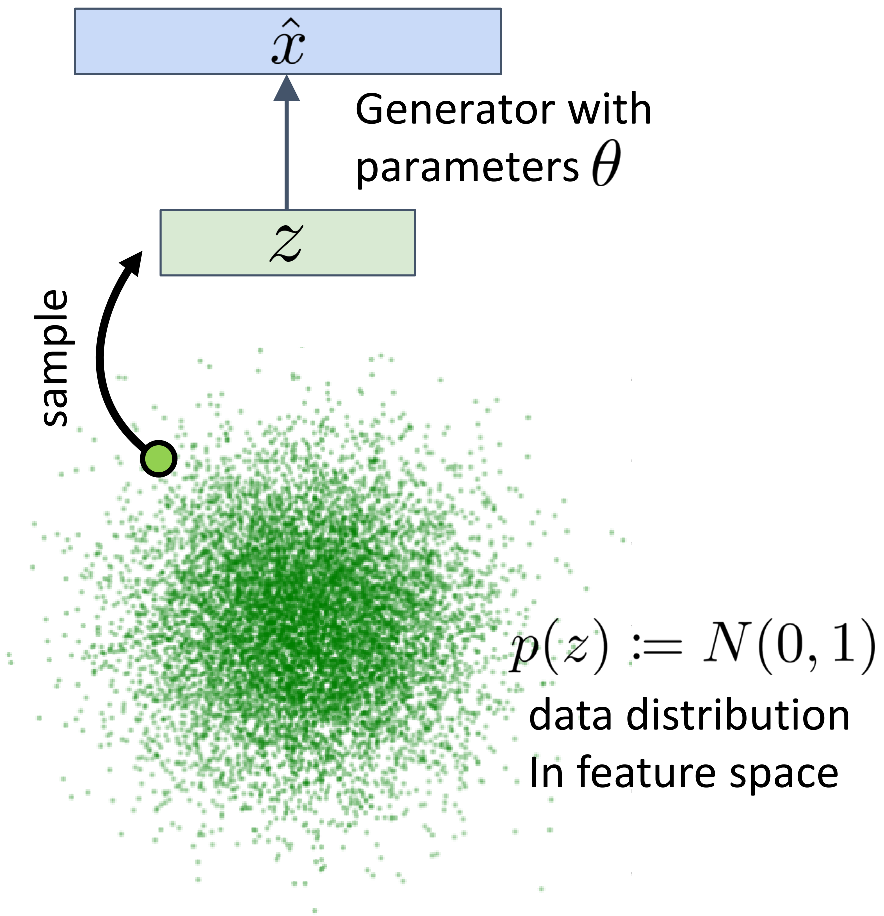


We do not know the distribution $p(z)$ in feature space

Image Credit: *Reducing the Dimensionality of Data with Neural Networks*, Hinton and Salakhutdinov

Latent Variable Model

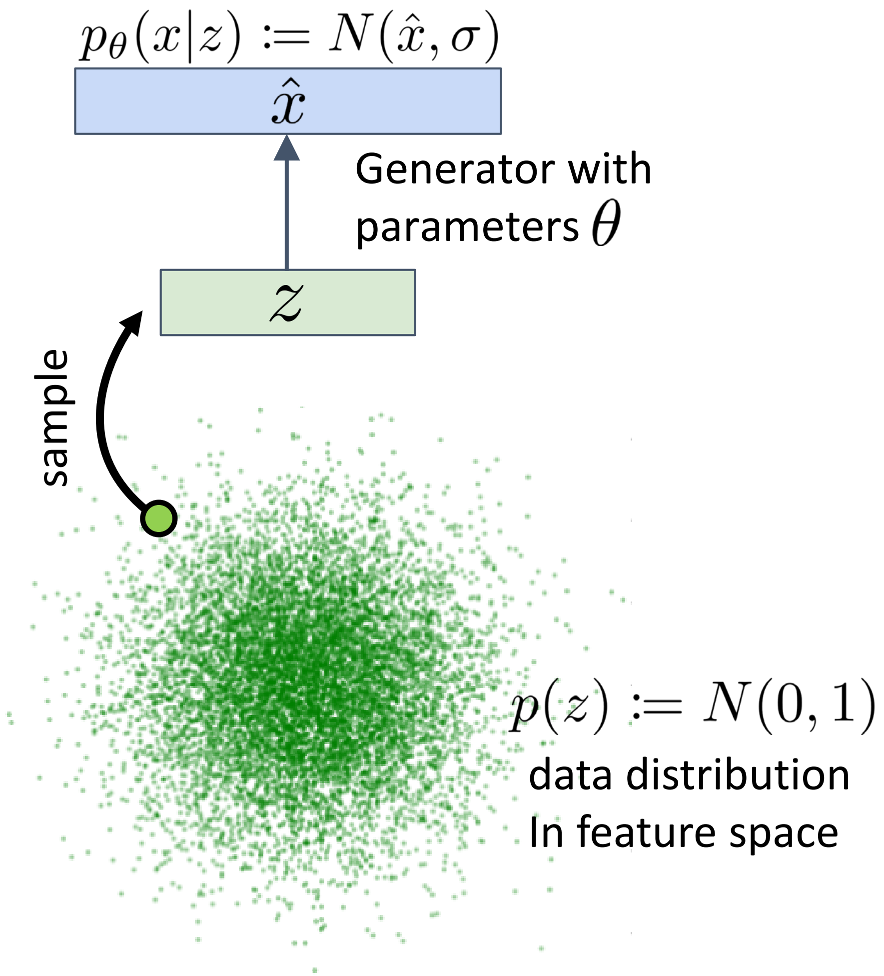
Sample? Evaluate?



- Define $p(z)$ as a known distribution

Latent Variable Model

Sample?
Evaluate?



(Negative log-likelihood)

- Train generator with NLL of data as loss

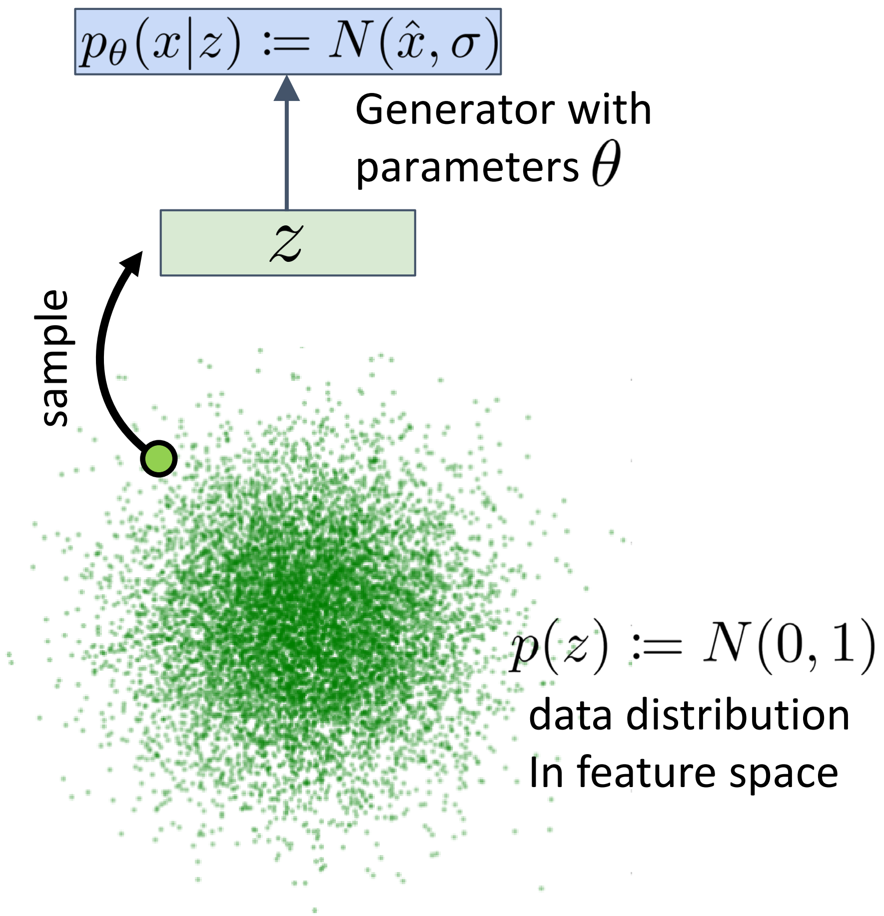
$$-\log \sum_{x_i \in \text{data}} p_\theta(x_i)$$

- Can we compute the likelihood $p_\theta(x)$?

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

Latent Variable Model

Sample? Evaluate?



(Negative log-likelihood)

- Train generator with NLL of data as loss

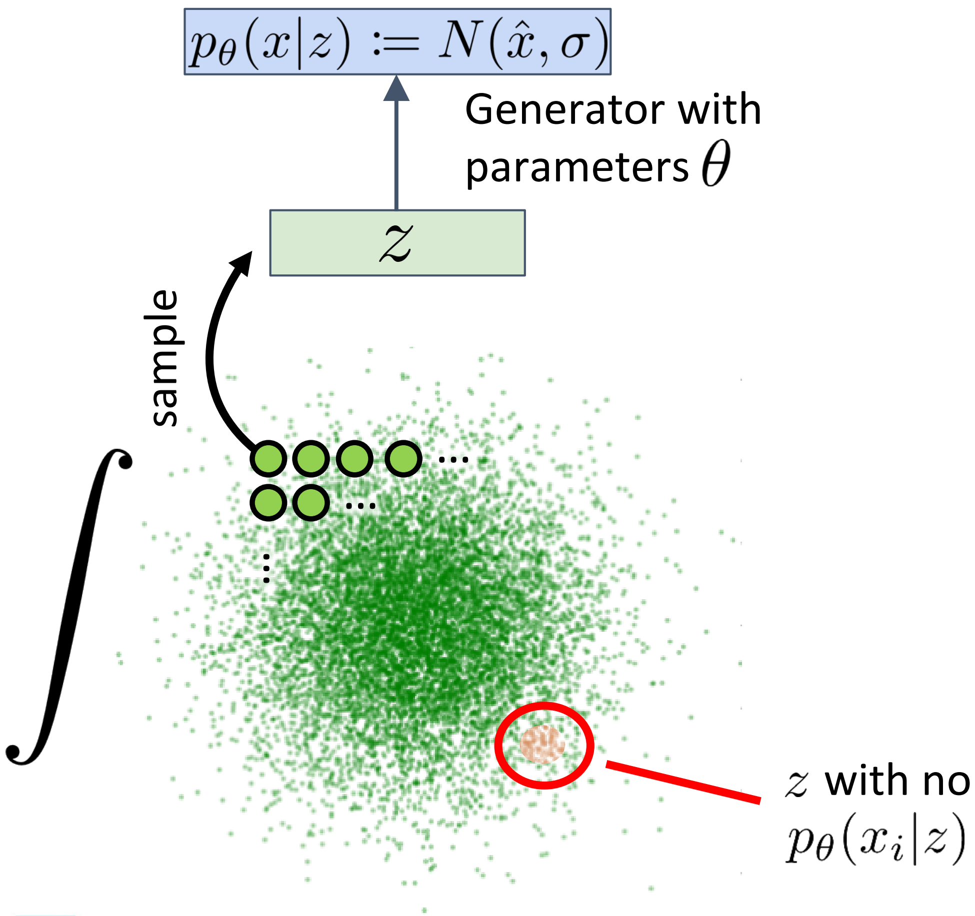
$$-\log \sum_{x_i \in \text{data}} p_\theta(x_i)$$

- Can we compute the likelihood $p_\theta(x)$?

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

Latent Variable Model: Monte-Carlo

Sample? Evaluate?



(Negative log-likelihood)

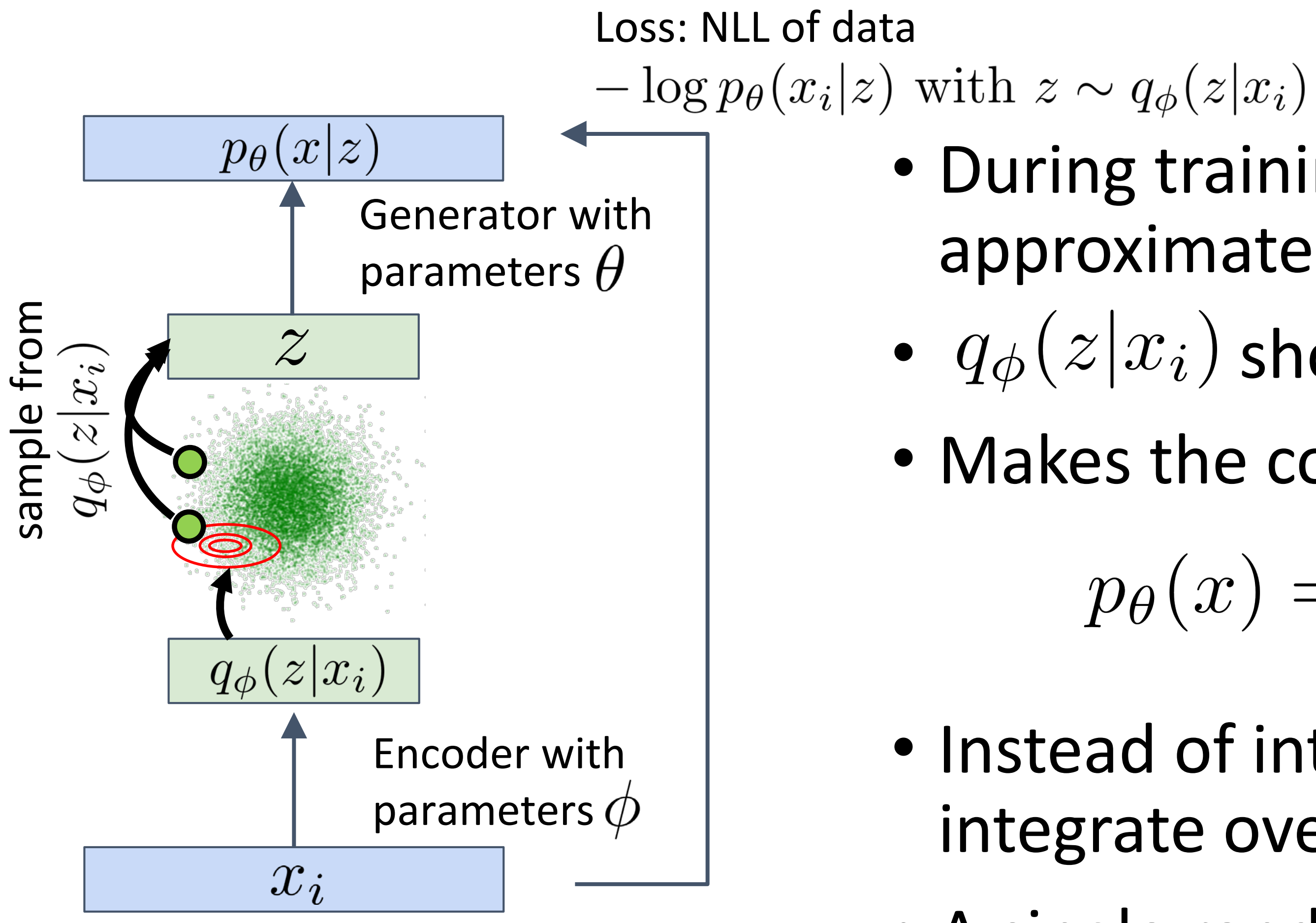
- Can we compute the likelihood $p_\theta(x)$?

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

- Monte-Carlo integration to solve integral for each data sample
- Very expensive, or very inaccurate (depending on sample count)

Variational Autoencoders (VAEs): The Encoder

Sample? ✓
Evaluate?



- During training, another network can learn to approximate the distribution $p(z|x_i)$
- $q_\phi(z|x_i)$ should be much smaller than $p(z)$
- Makes the computing the integral tractable

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

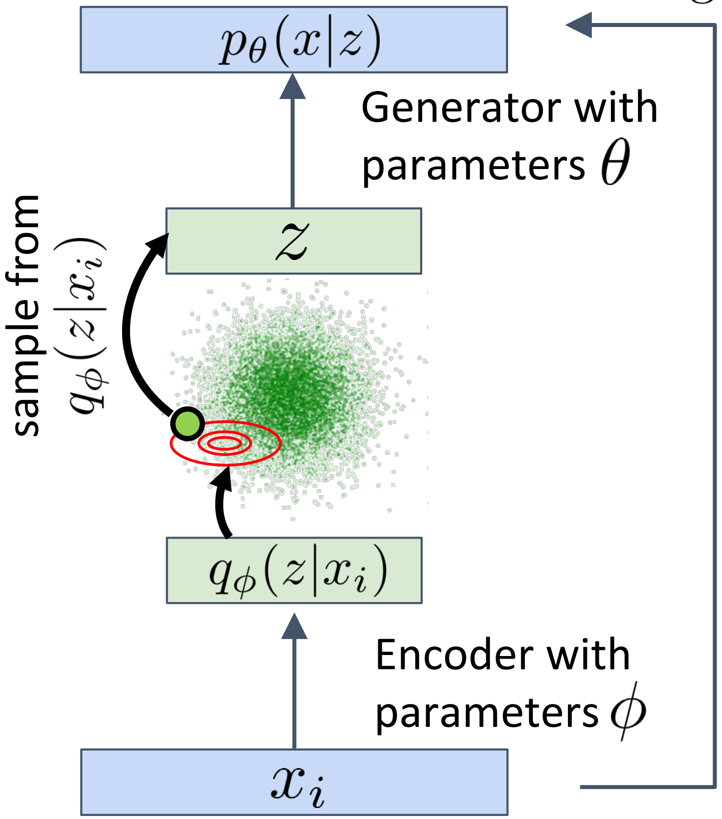
- Instead of integrating over all $p(z)$, integrate over $q_\phi(z|x_i)$ only
- A single random sample from $q_\phi(z|x_i)$ per iteration is usually enough

Sample? ✓
Evaluate? $\approx$

Variational Autoencoders (VAEs): Regularization

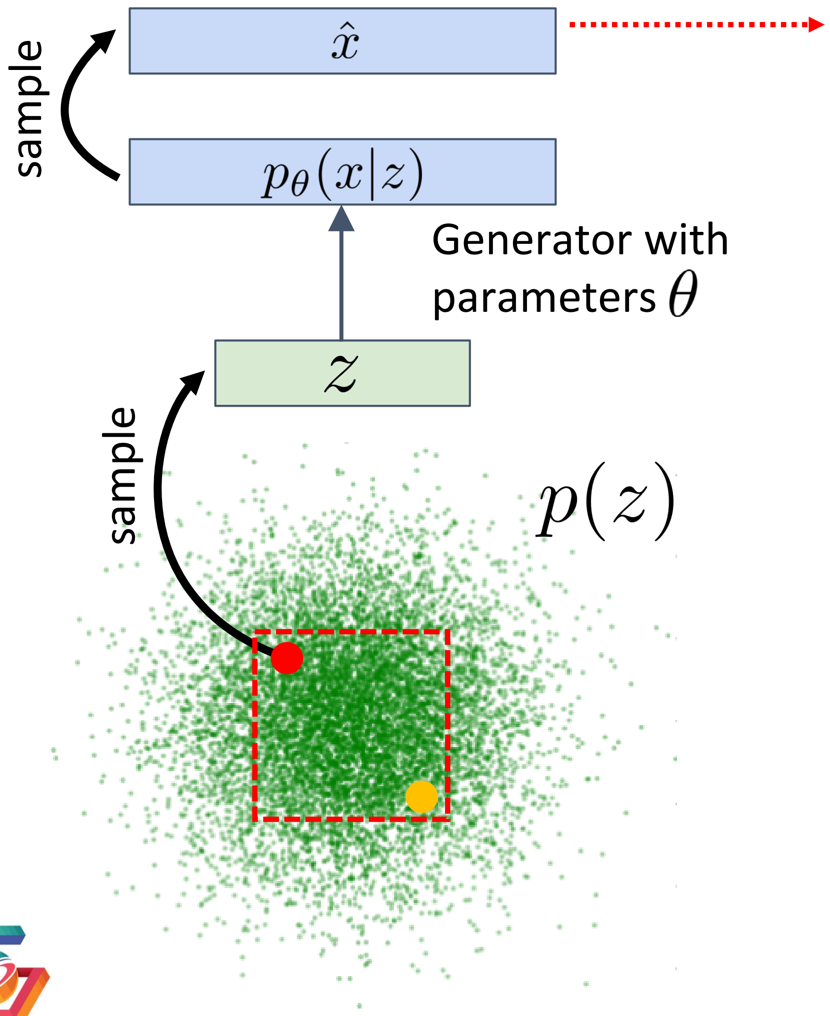
NLL of data as loss

$$-\log p_\theta(x_i|z) \text{ with } z \sim q_\phi(z|x_i) + KL(\ q_\phi(z|x_i) \parallel p(z)\ )$$



- The network can choose $q_\phi(z|x_i)$ freely
- But it is **regularized** it to approximate $p(z)$
- Neg. loss is a lower bound for the data's likelihood in the generated distribution

Generating Data



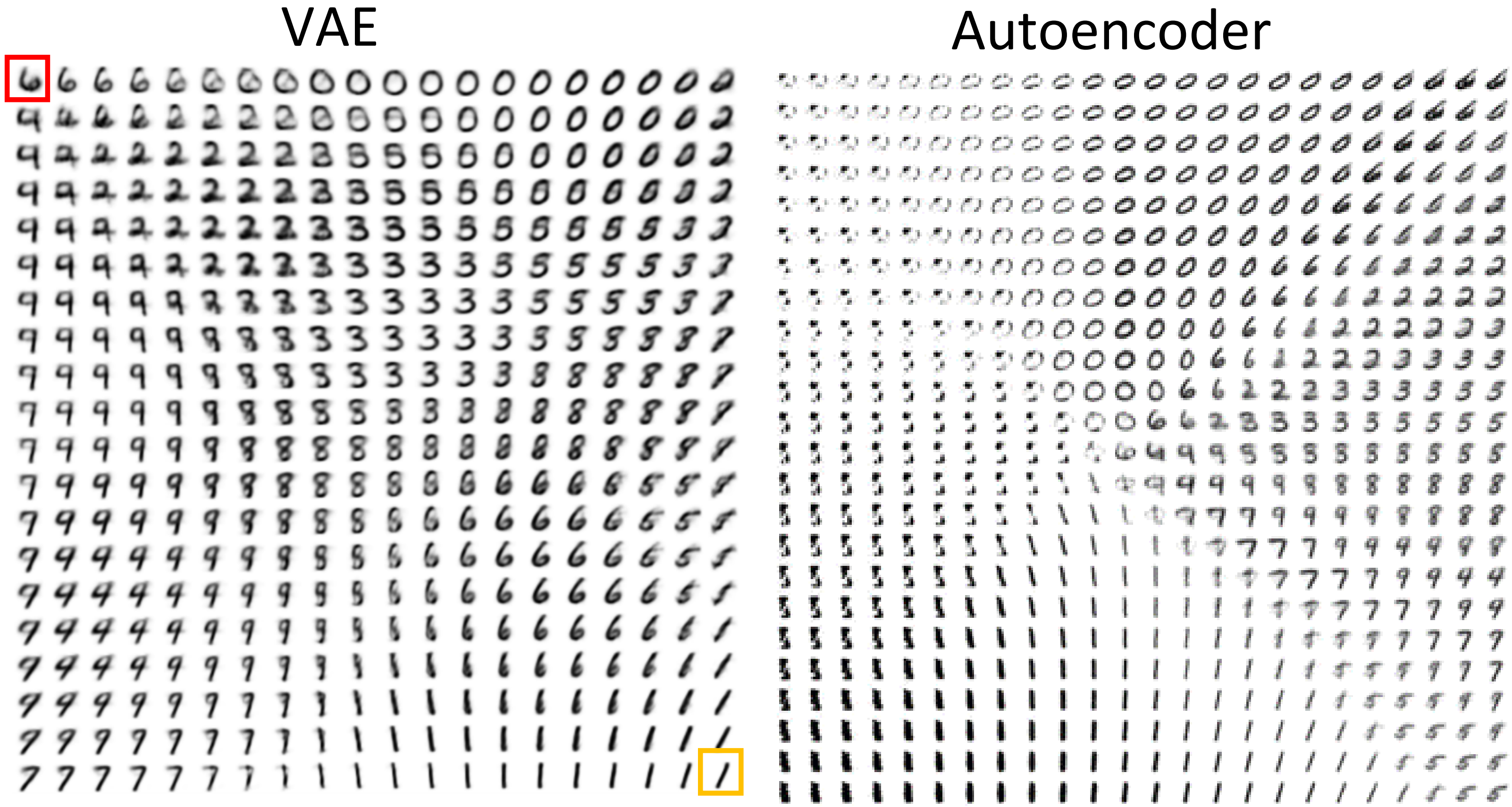
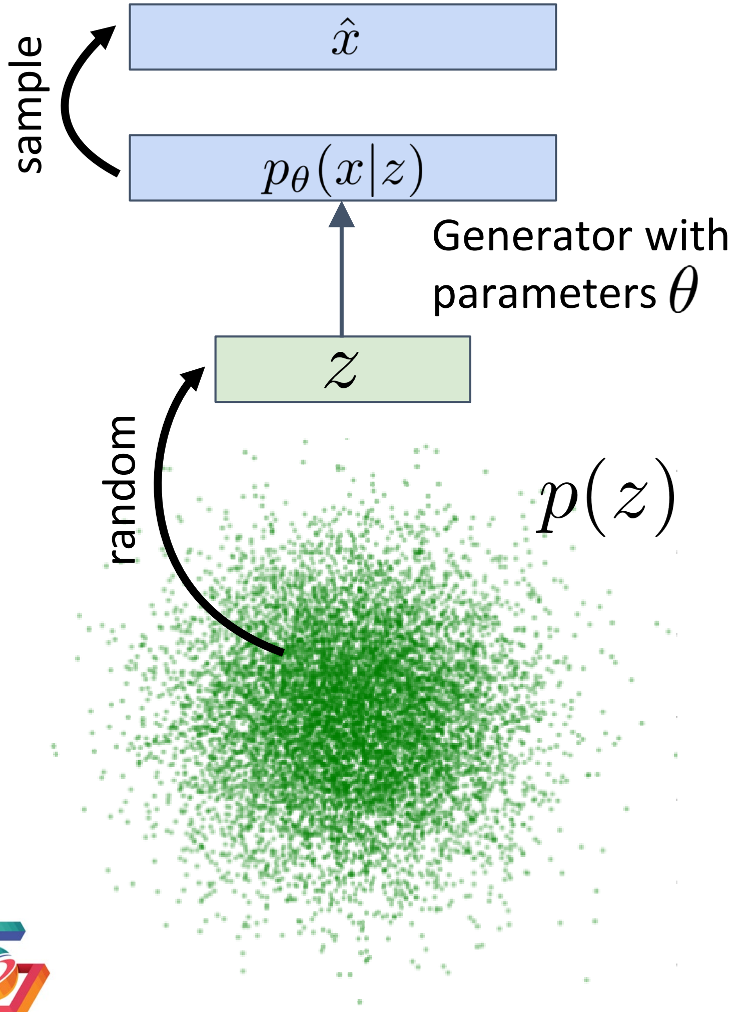


Image Credit: *Auto-Encoding Variational Bayes*, Kingma and Welling

Generating Data

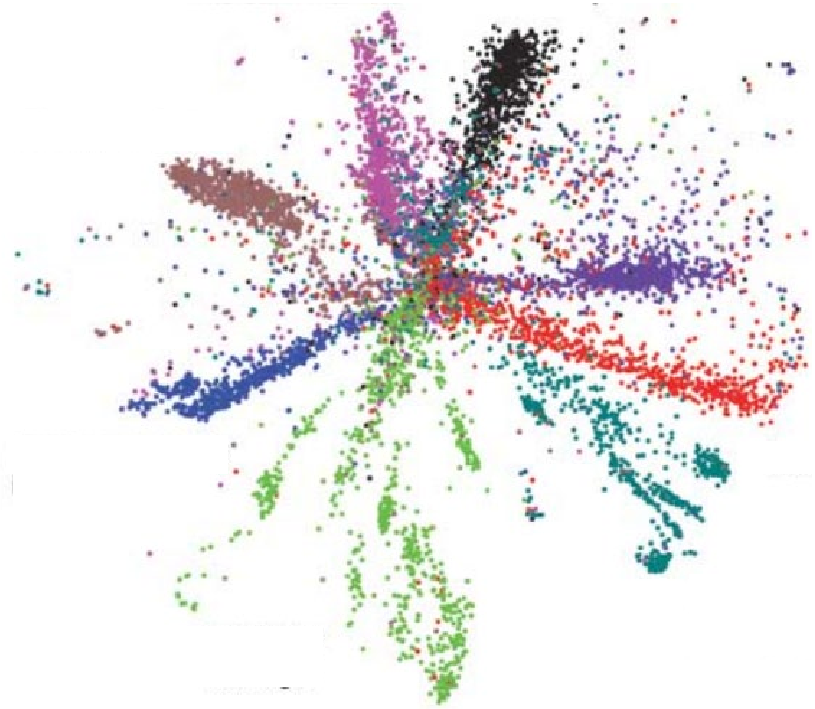


VAE

Autoencoder

Feature Space of Autoencoders vs. VAEs

Autoencoder



VAE

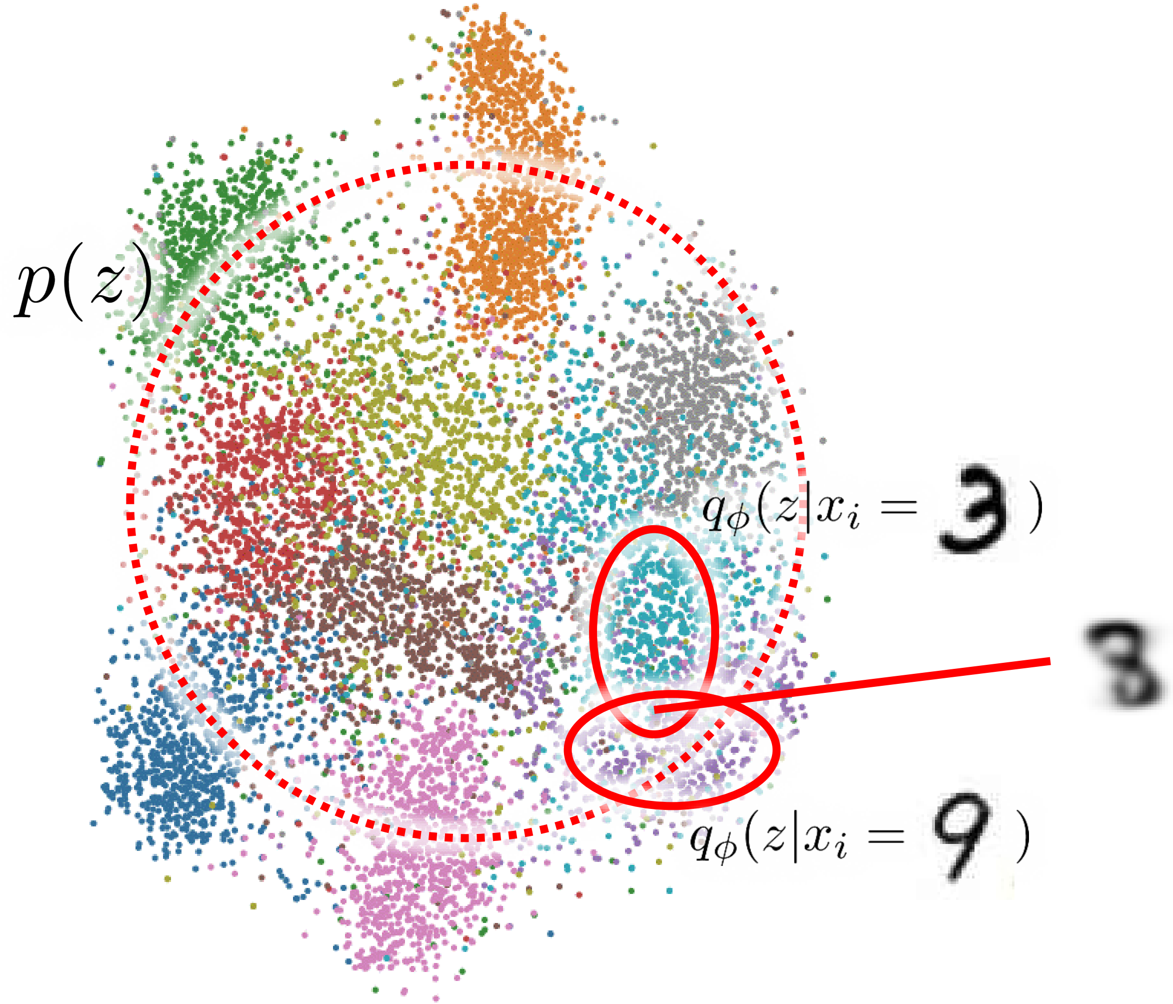


Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

Summary: Variational Autoencoders (VAEs)

Positives

- Creates a feature space
- Relatively stable to train

Negatives

- Likelihood evaluation can only be approximated
- Projection of sample into feature space can only be approximated
- Regularization makes the results a bit blurry

Normalizing Flows

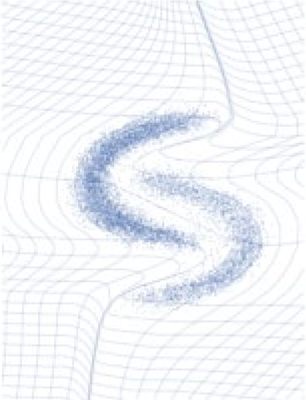
Sample?

Evaluate?

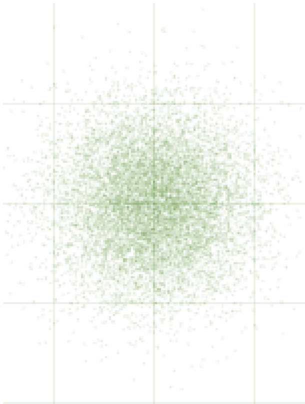
$x \sim p_\theta(x)$

f_θ Generator with parameters θ

$z \sim p_z(z)$



$$x = f_\theta(z)$$



$p_z(z)$
(known)

Variational Autoencoders (VAEs):

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

Image Credit: DENSITY ESTIMATION USING REAL NVP:, Dinh et al.

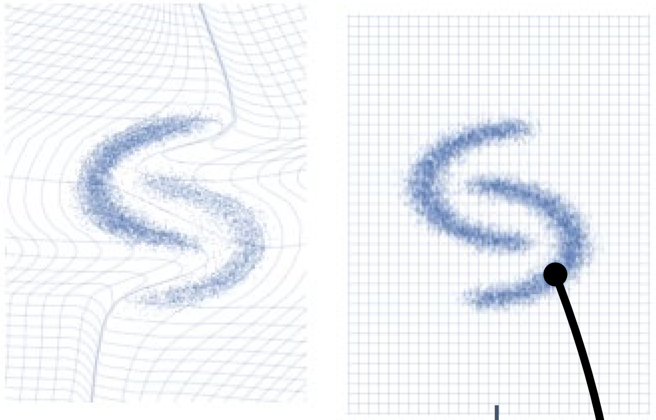
Normalizing Flows

Sample? ✓
Evaluate? ✓

$x \sim p_\theta(x)$

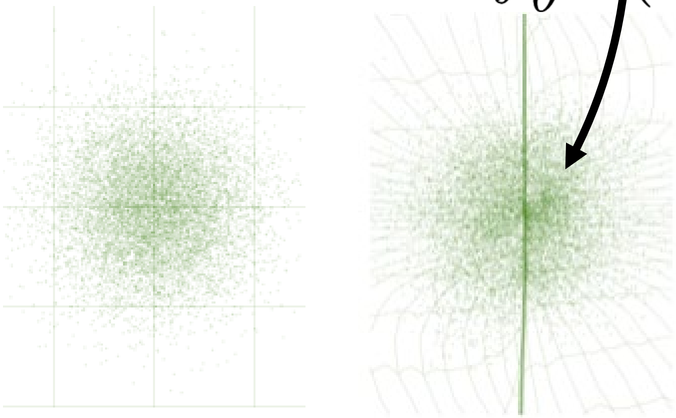
f_θ Generator with parameters θ

$z \sim p_z(z)$



$x = f_\theta(z)$

$z = f_\theta^{-1}(x)$



$p_z(z)$
(known)

Normalizing Flows:

$$p_\theta(x) = p_z\left(f_\theta^{-1}(x)\right)$$

… times the local density change caused by f_θ

$$\left|\det\left(\frac{\partial f_\theta^{-1}(x)}{\partial x}\right)\right|$$

Image Credit: DENSITY ESTIMATION USING REAL NVP:, Dinh et al.

Normalizing Flows: Chaining

Sample?
Evaluate?

$$
\begin{tikzcd}
x_3 \sim p_{x_3}(x_3) \\
x_2 \sim p_{x_2}(x_2) \arrow[u, "f_3"] \\
x_1 \sim p_{x_1}(x_1) \arrow[u, "f_2"] \\
z \sim p_z(z) \arrow[u, "f_1"]
\end{tikzcd}
$$

$$p_3(x_3) = p_z\left(f_1^{-1} \circ f_2^{-1} \circ f_3^{-1}(x_3)\right)$$

… times the local density change
caused by the chain of transformations

$$\prod_{i=1}^{3} \left| \det\left(\frac{\partial f_i^{-1}}{\partial x_i}\right) \right|$$

Example: Glow

Invertible functions:

- Linear (1x1 conv) layer with weight matrix parameterized by its LU decomposition
- Affine coupling layer to propagate information between pixels

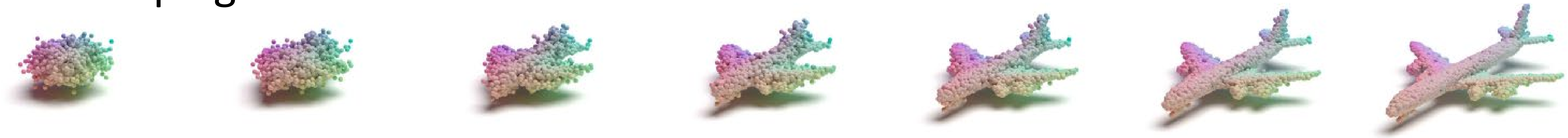
Training: 40 GPUs, 2 weeks

Image Credit: Glow: Generative Flow with Invertible 1x1 Convolutions, Kingma and Dhariwal

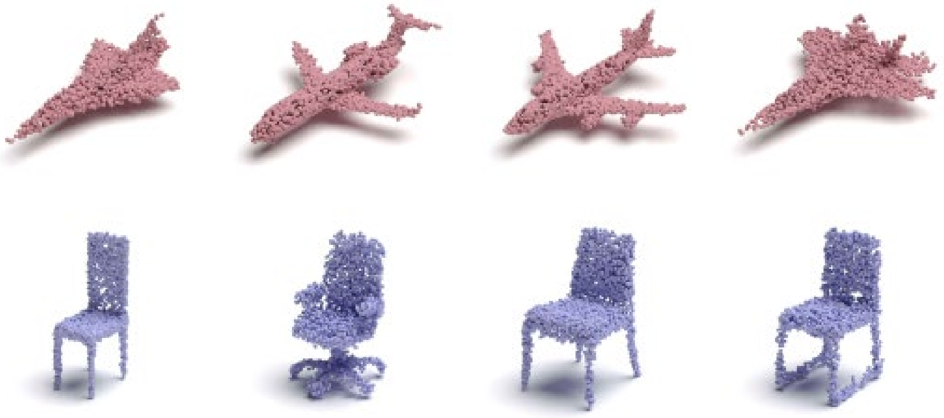
Example: PointFlow

Two flows: One to create the distribution of shape feature vectors, one for the distribution of points on a shape

Shape generation flow



Free shape generation



Shape interpolation

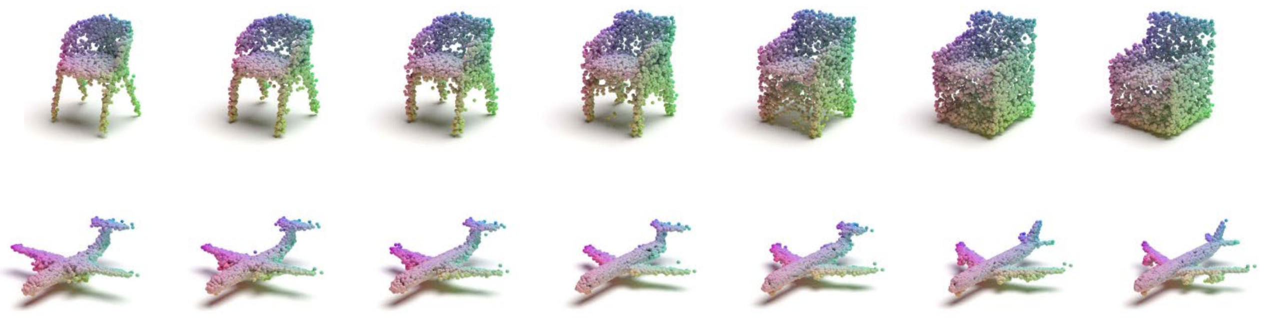


Image Credit: PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows, Yang et al.

Summary: Normalizing Flows

Positives

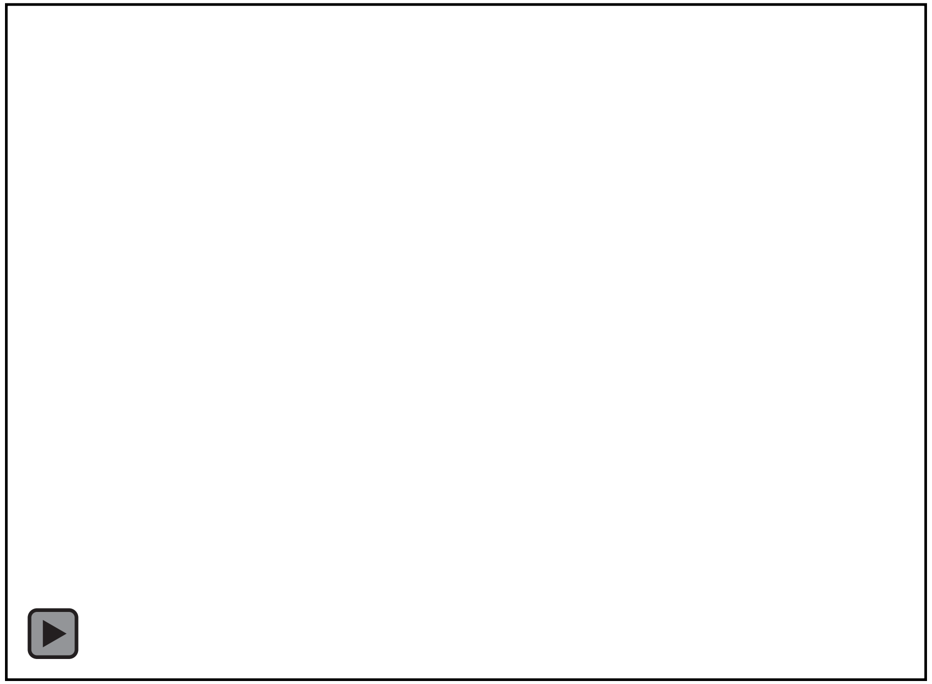
- Creates a feature space
- Exact projection to feature space
- Exact likelihood evaluation

Negatives

- Only a limited set of functions (invertible and Jacobian easy to compute)
- Currently takes longer and needs more parameters than GANs for same quality

Autoregressive Models

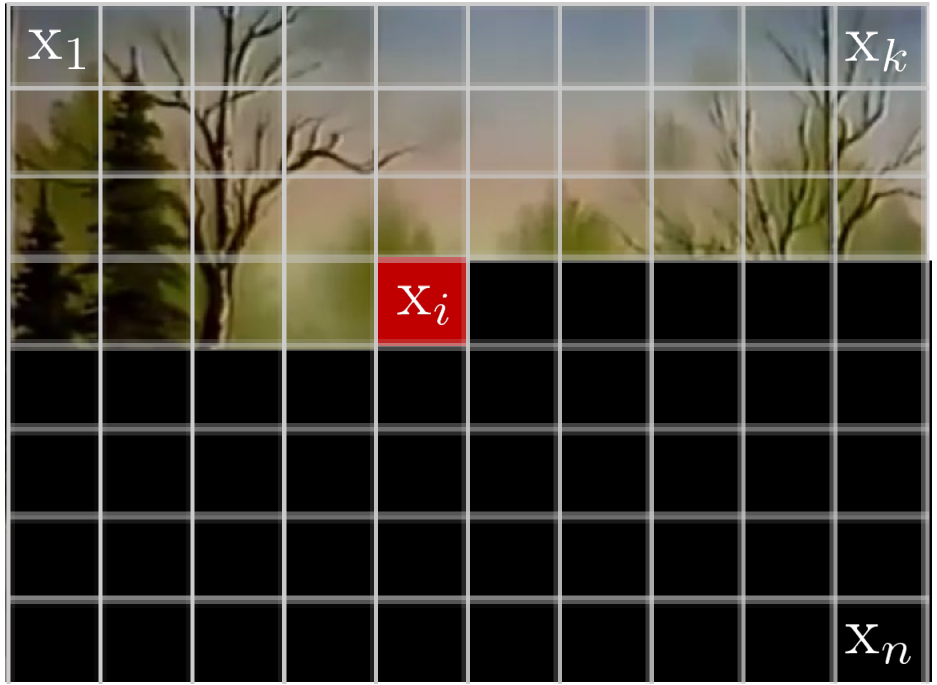
- Create output step-by-step
- Each step depends on the output of all previous steps



Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

Autoregressive Models

- Create output step-by-step
- Each step depends on the output of all previous steps



$$x = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$$

$$p_\theta(x) = p_\theta(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$$

Chain rule of probability:

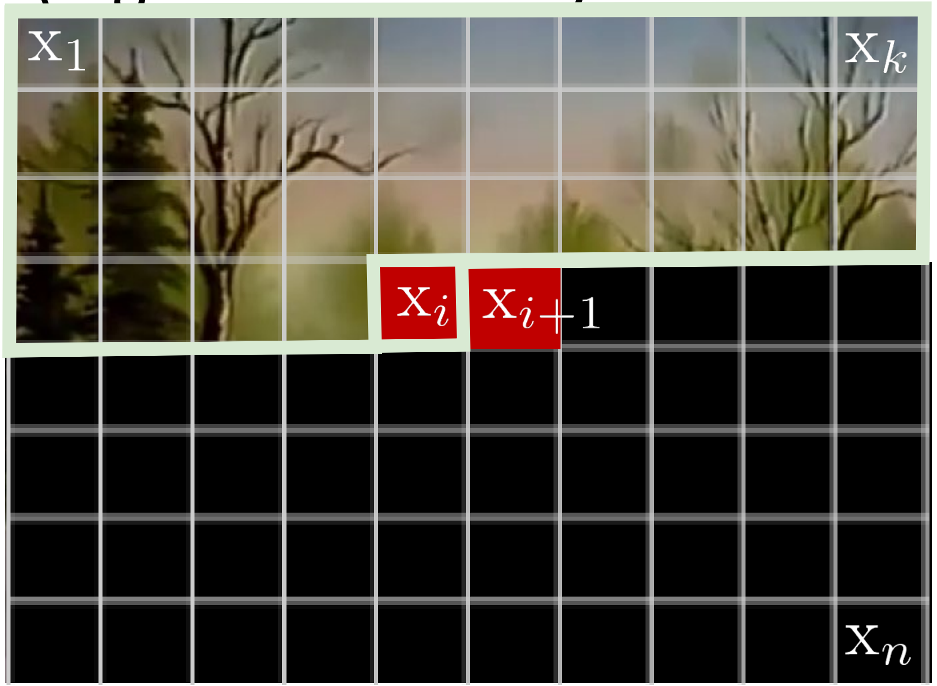
$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(\mathrm{x}_i | \mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{i-1})$$

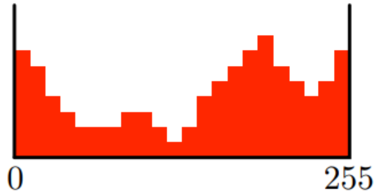
Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

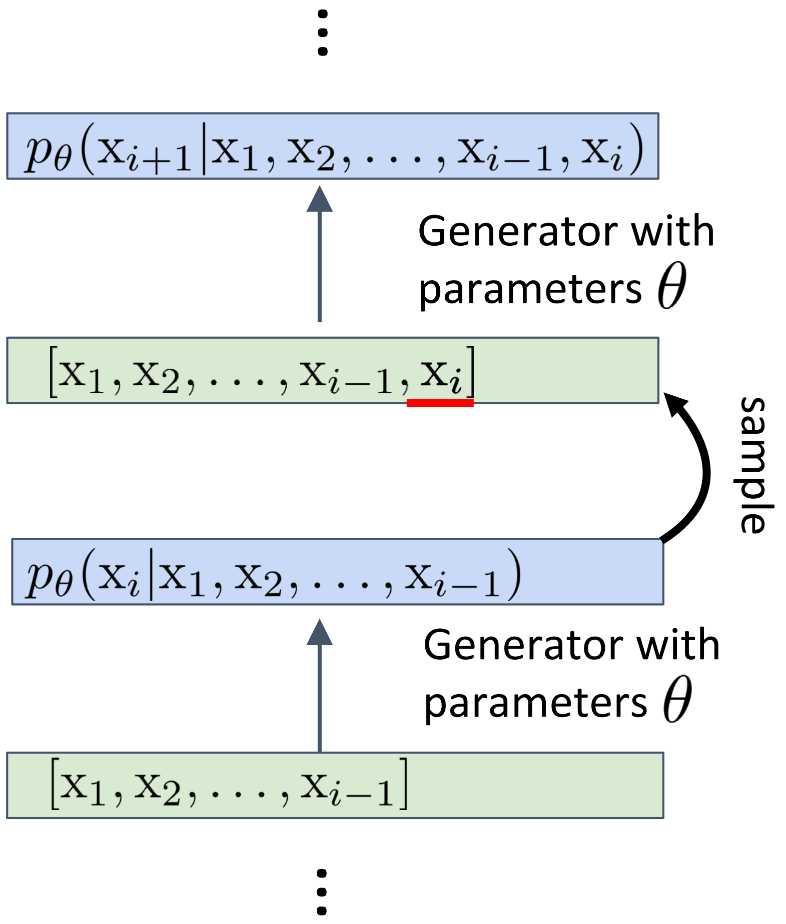
Autoregressive Models

Sample? ✓
Evaluate?

- In each step, the model outputs a low-dimensional prob. distribution (e.g. over intensity values for one pixel)



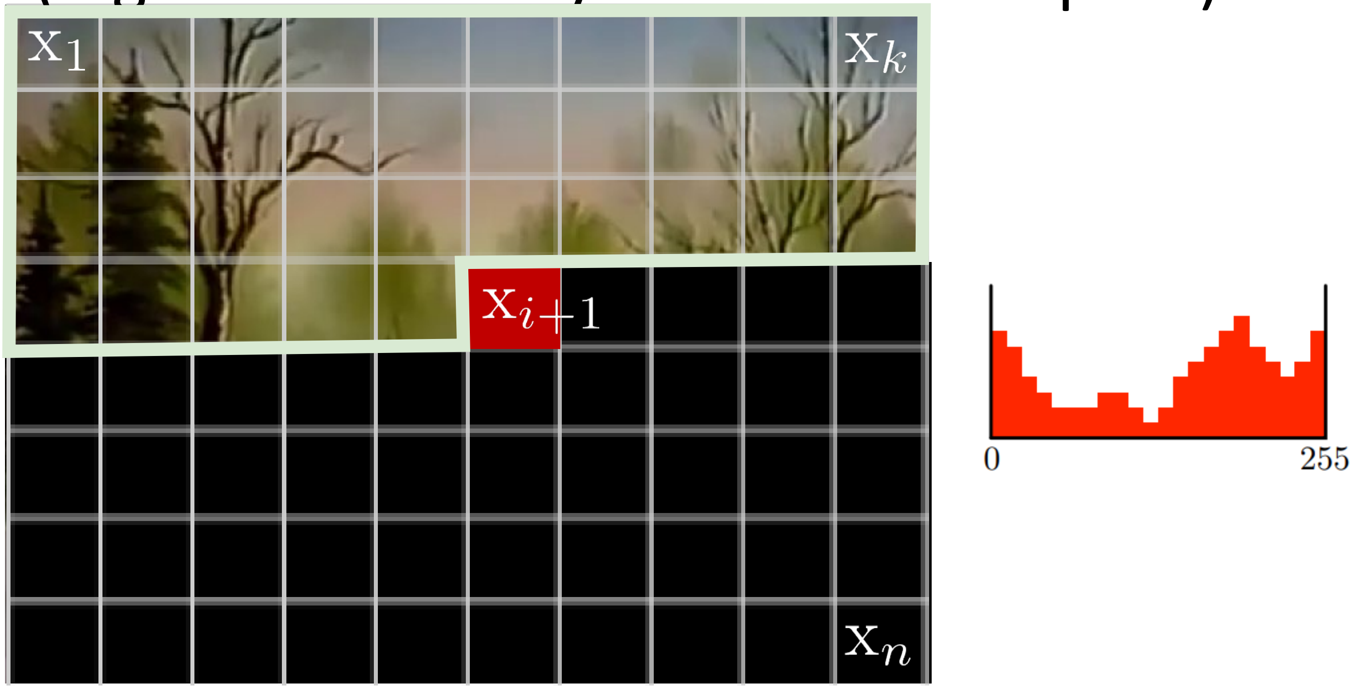




Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

Autoregressive Models

- In each step, the model outputs a low-dimensional prob. distribution (e.g. over intensity values for one pixel)



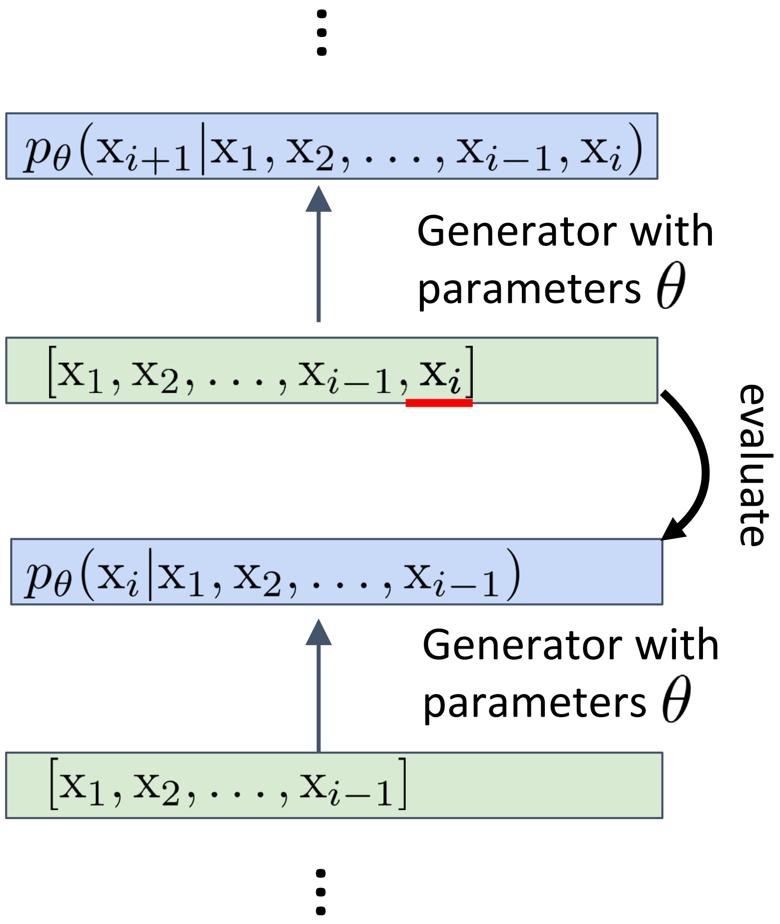
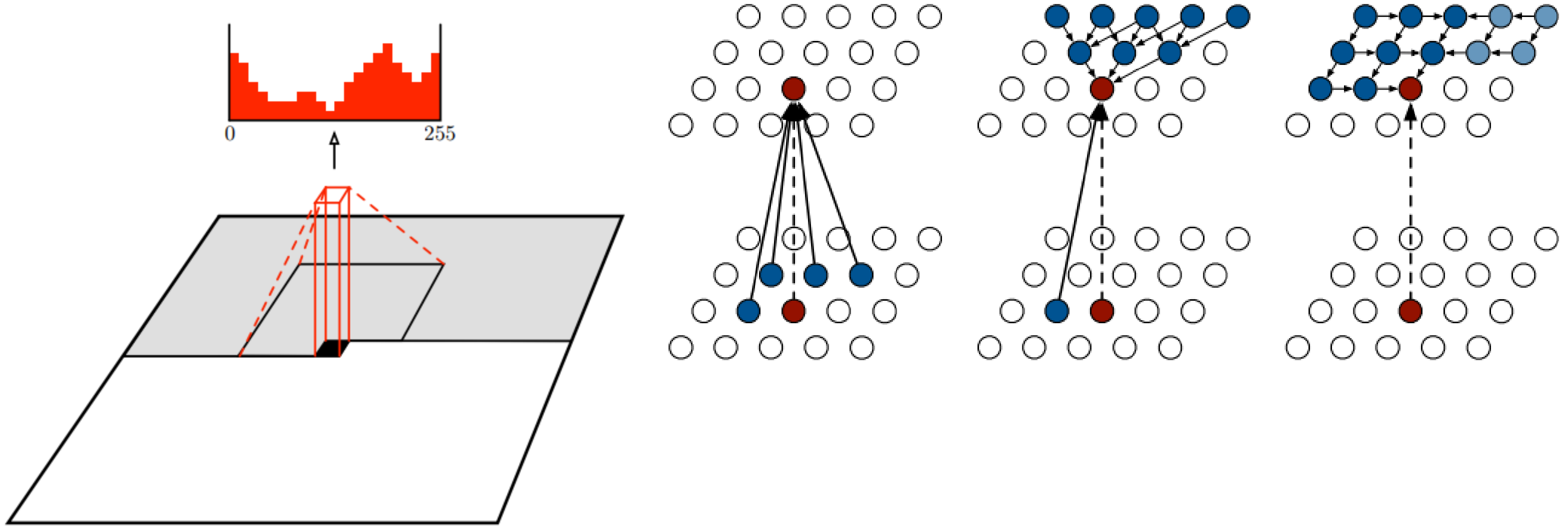


Image Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

Example: PixelRNN and PixelCNN

- Recursive network that has an **input** and a **state** (LSTM)
- Only recent steps are used as **input**, the **state** summarizes older steps



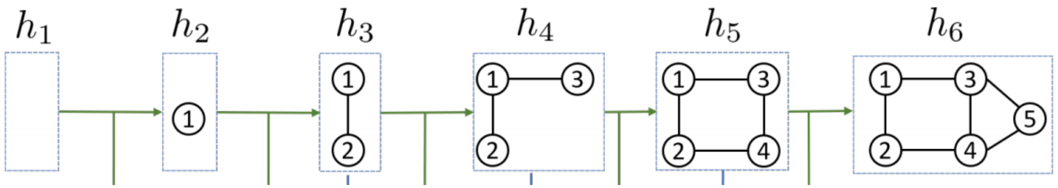
Sandbar

Lhasa Apso (dog)

Brown bear

Example: Graph Generation

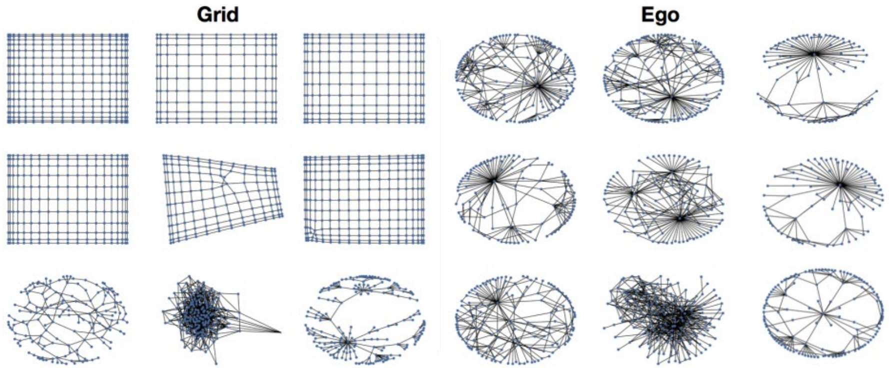
GraphRNN



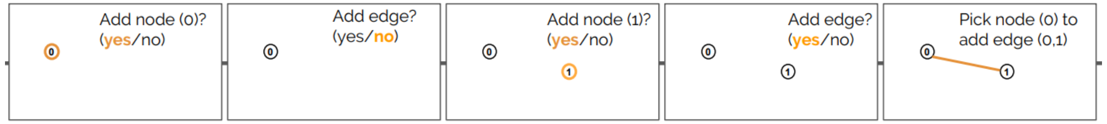
Training Set

GraphRNN

Baseline



Learning Deep Generative Models of Graphs



Training Set

New model

Baseline

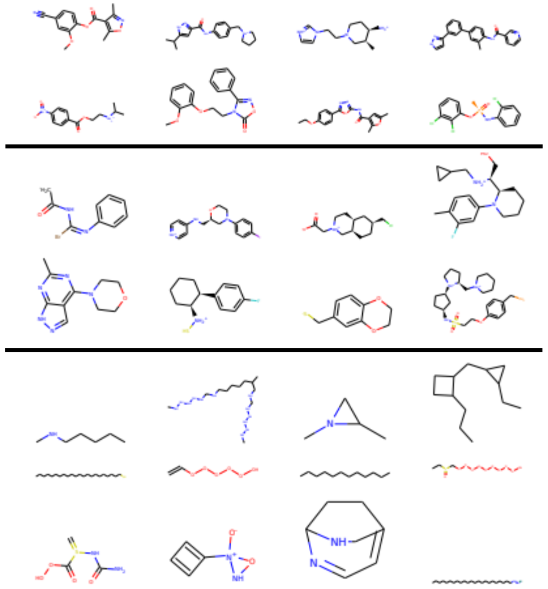


Image Credit: Conditional Image Generation with PixelCNN Decoders, Oord et al.
GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models, You et al.
Learning Deep Generative Models of Graphs , Li et al.

Summary: Autoregressive Models

Positives

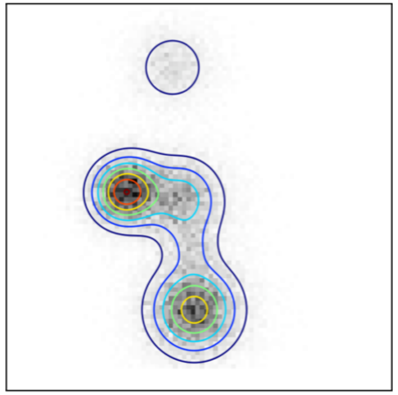
- Flexible output length
- Exact likelihood evaluation

Negatives

- No feature space
- Sequential generation (usually slow)

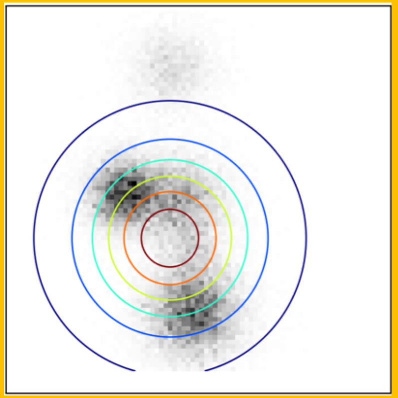
Generative Models

How do we measure the similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?



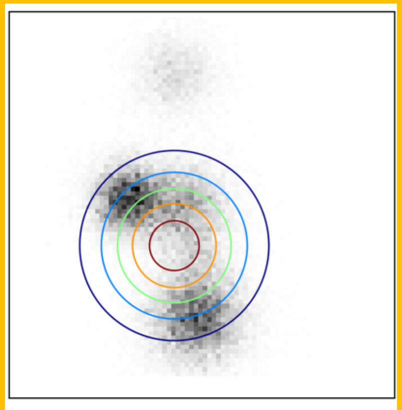
$p_{\text{data}}(x)$

1) Likelihood of data samples in $p_\theta(x)$



$\approx KL(p_{\text{data}} \,||\, p_\theta)$

2) Adversarial game



$\approx JS(p_{\text{data}} \,||\, p_\theta)$

Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár

Generative Adversarial Networks (GANs)

z → **Player 1: generator**
Scores if discriminator can't distinguish output from real image

from dataset

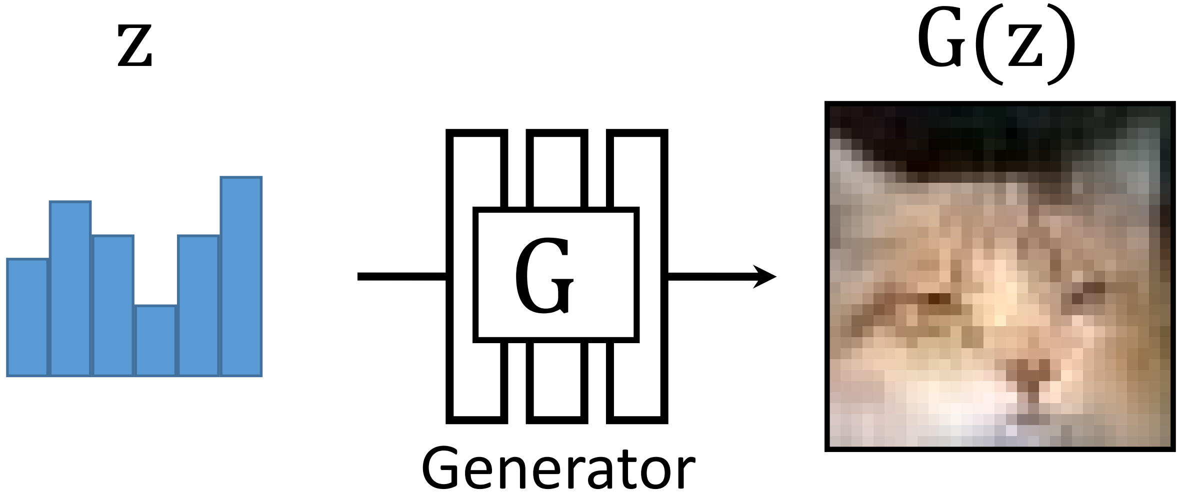
Player 2: discriminator → real/fake
Scores if it can distinguish between real and fake

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks*, Karras et al.

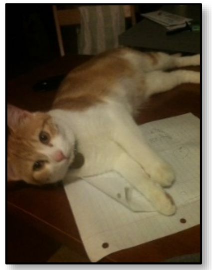
Generative Adversarial Networks (GANs)

Player 1: generator
Scores if discriminator can't distinguish output from real image

Player 2: discriminator
Scores if it can distinguish between real and fake

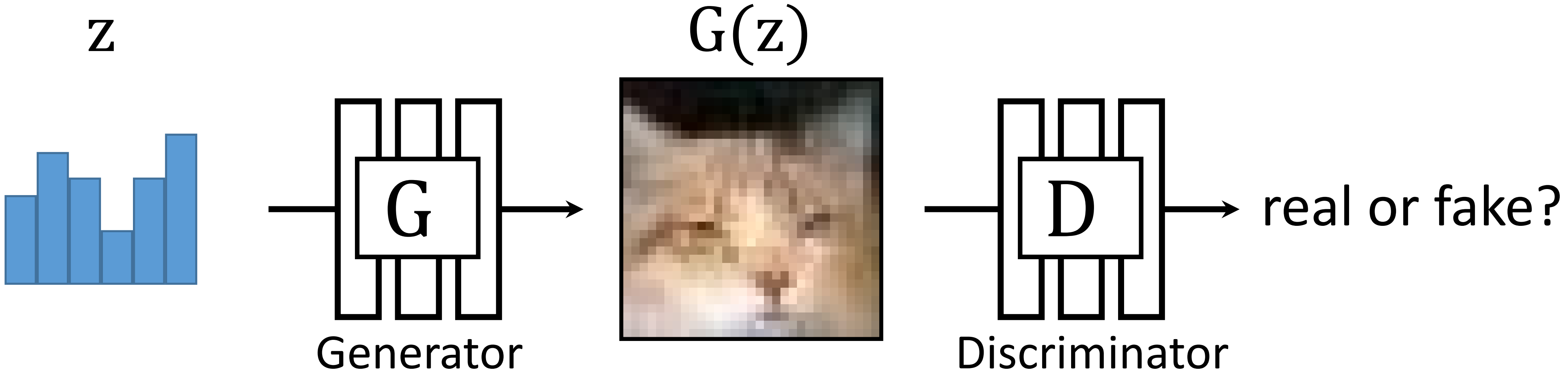


G: generate fake samples that can fool D



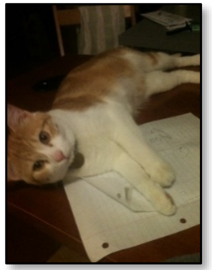
cat credit: aleju/cat-generator

slide credit: Phillip Isola & Jun-Yan Zhu



G: generate fake samples that can fool D

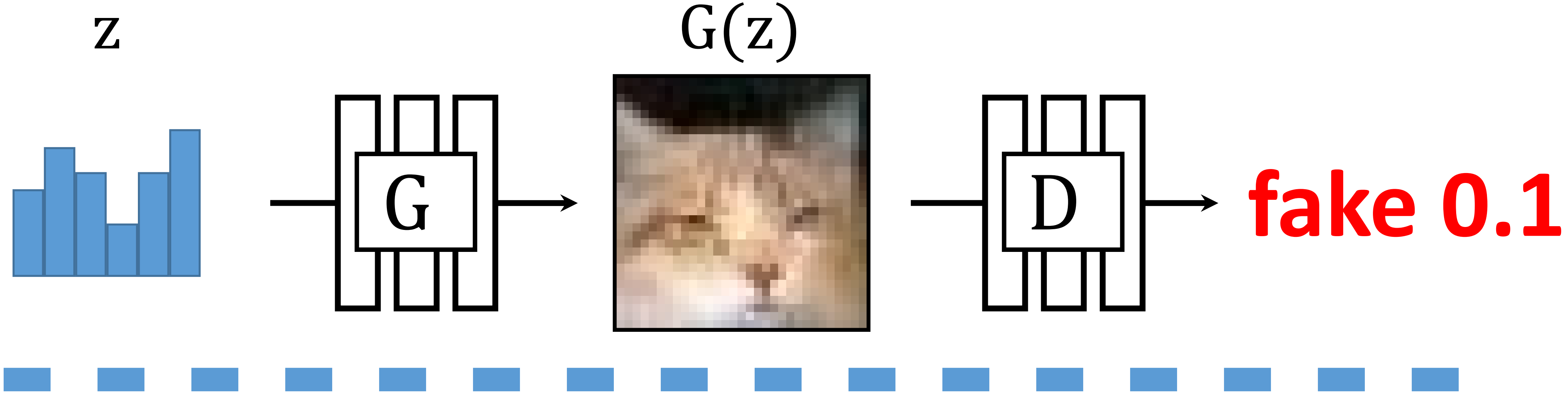
D: classify fake samples vs. real images



slide credit: Phillip Isola & Jun-Yan Zhu

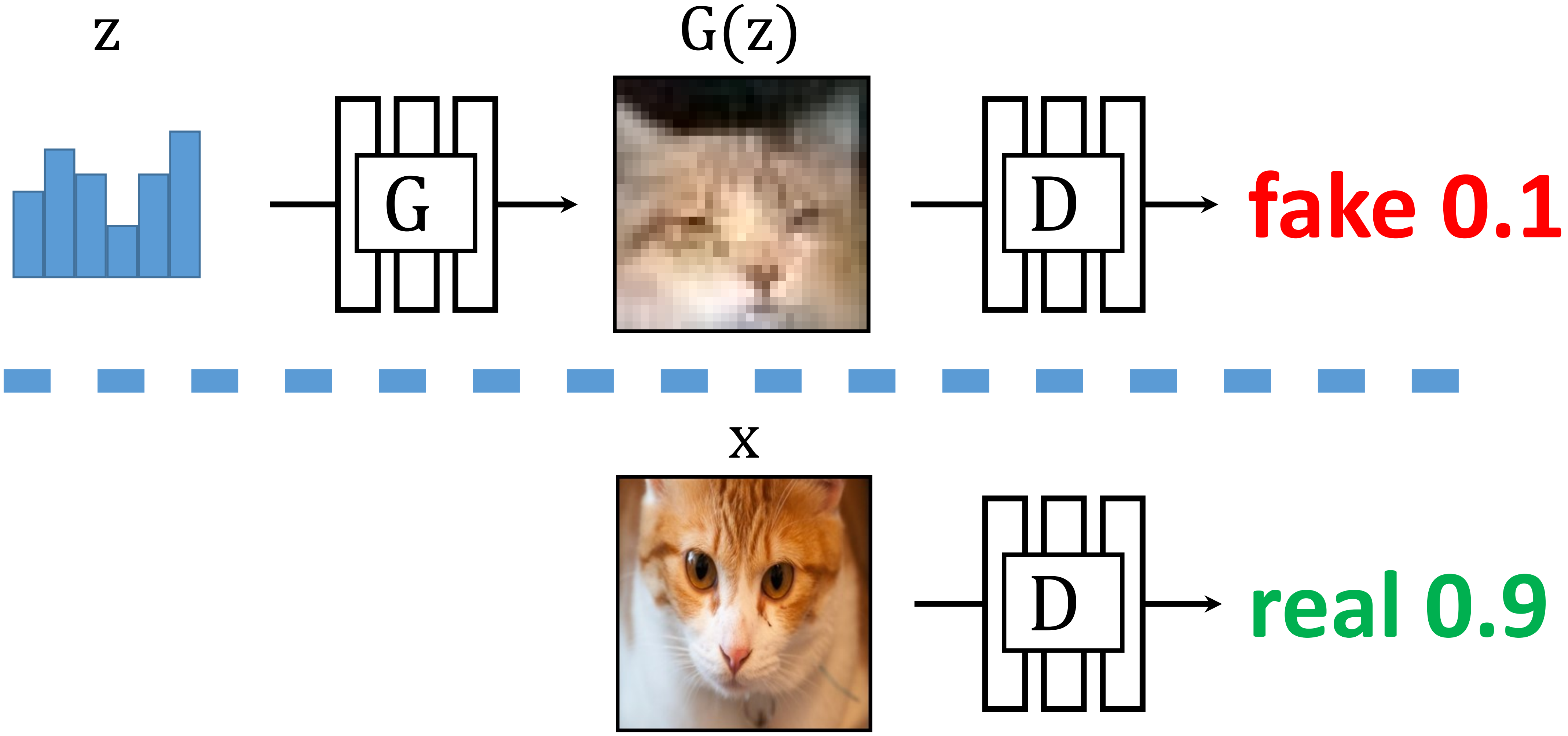
$$\min_G \max_D \mathbb{E}_{z,x}[\qquad\qquad\qquad\qquad\qquad\qquad]$$

slide credit: Phillip Isola & Jun-Yan Zhu



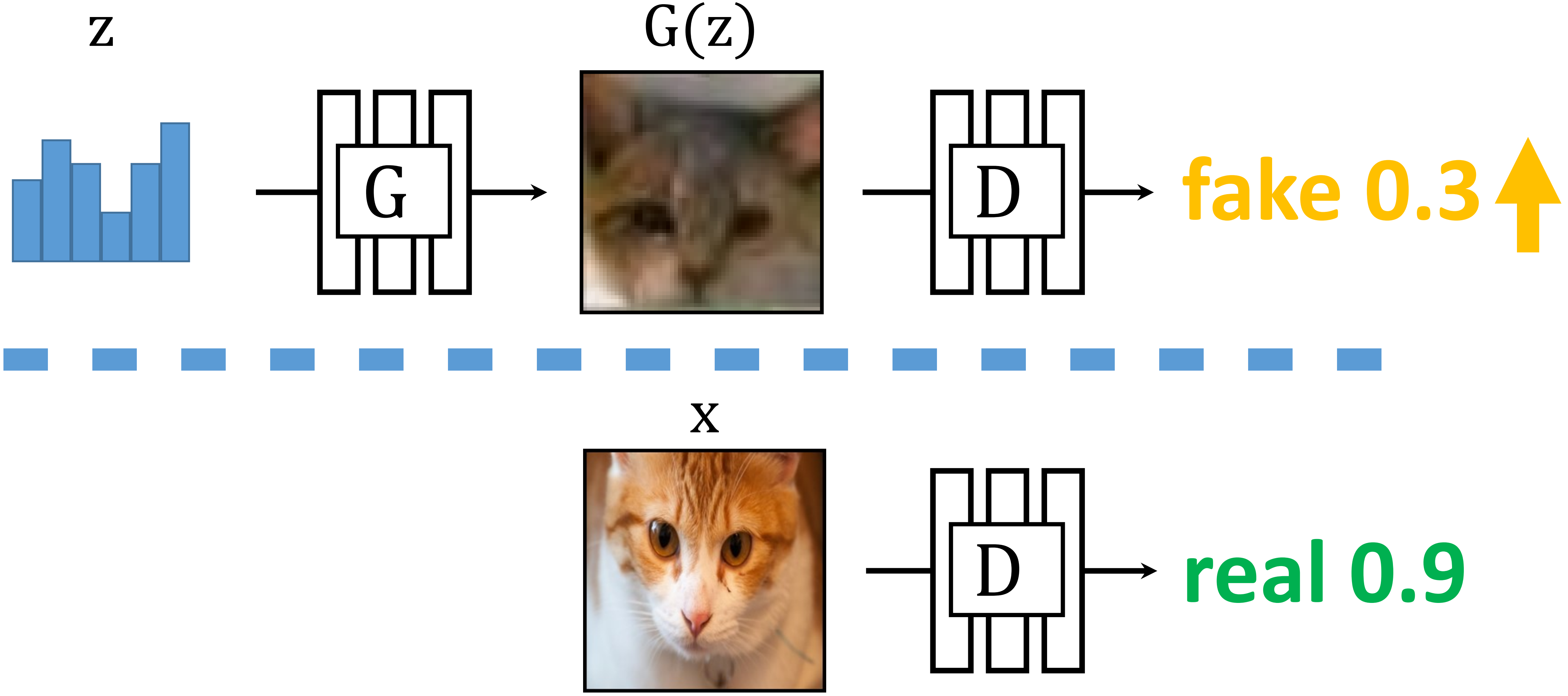
$$\min_{G}\max_{D}\mathbb{E}_{z,x}[\log \underbrace{D(G(z))}_{\text{fake}} \qquad\qquad\qquad]$$

slide credit: Phillip Isola & Jun-Yan Zhu



$$\min_G \max_D \mathbb{E}_{z,x}[\log \underbrace{D(G(z))}_{\text{fake}} + \log(1 - \underbrace{D(x)}_{\text{real}})]$$

slide credit: Phillip Isola & Jun-Yan Zhu

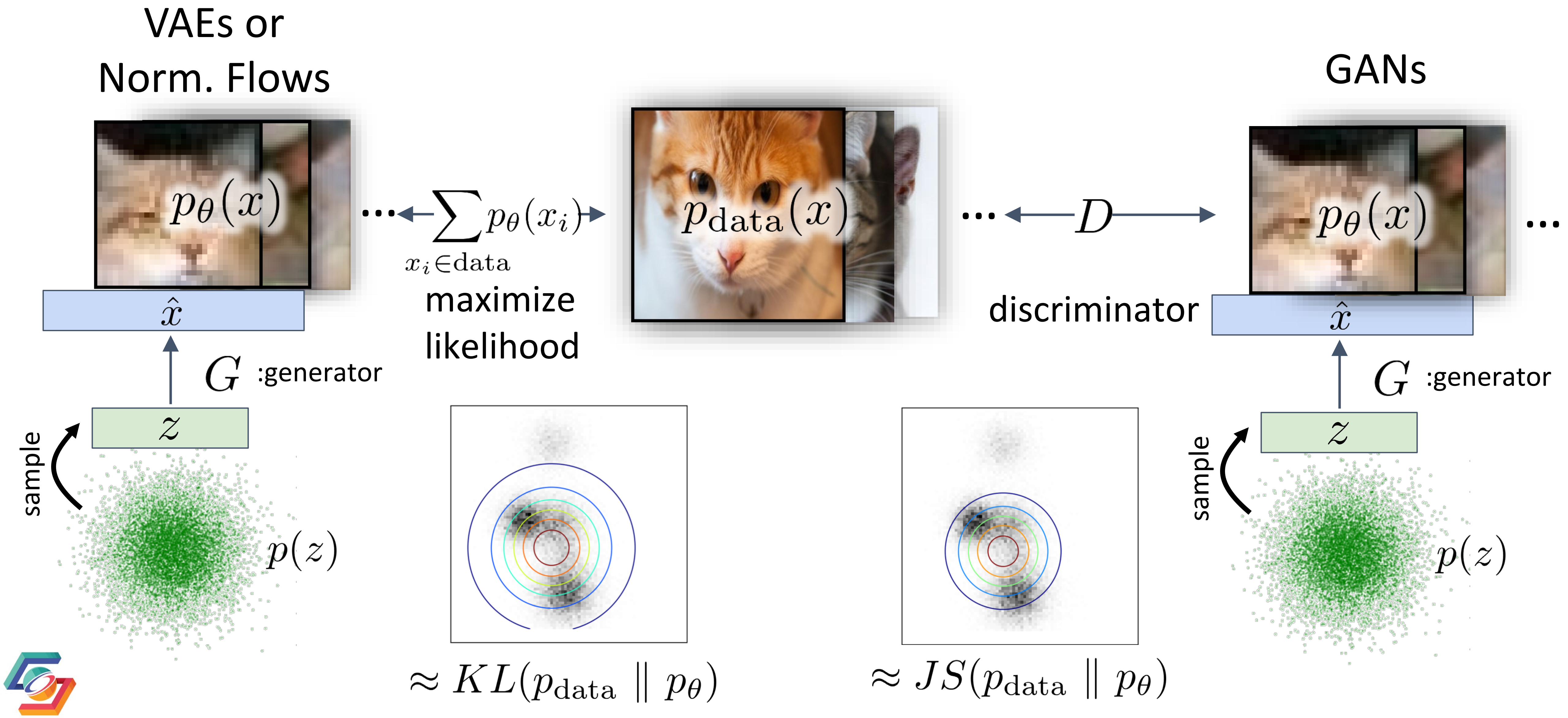


$$\min_G \max_D \mathbb{E}_{z,x}[\log \underbrace{D(G(z))}_{\text{Update G}} + \log(1 - D(x))]$$

slide credit: Phillip Isola & Jun-Yan Zhu

Generated Distributions of GANs vs ML

Sample?
Evaluate?



StyleGAN

Additional Tricks:

- Coarse-to-fine training
- Transformation of $p(z)$ to a more complex distr.
- ...

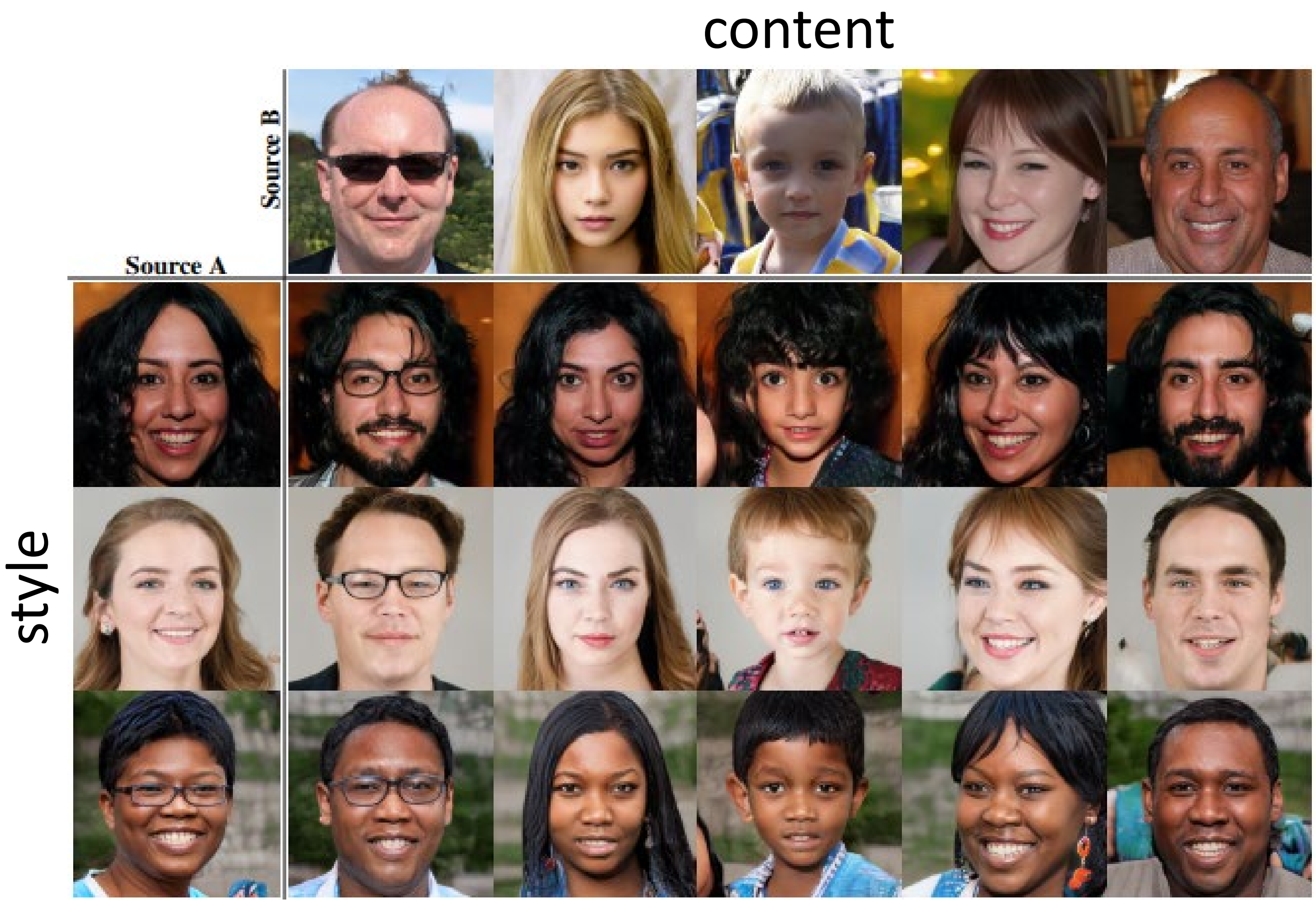


Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks,* Karras et al.

Summary: GANs

Positives

- Creates a feature space
- Currently highest-quality results

Negatives

- Can be unstable to train
- Not guaranteed to cover all of the data distribution
- Cannot evaluate likelihood

Open Problems

- More control
- Irregular data
- GAN training convergence
- Evaluating GANs

Conditional GAN: Pix2Pix

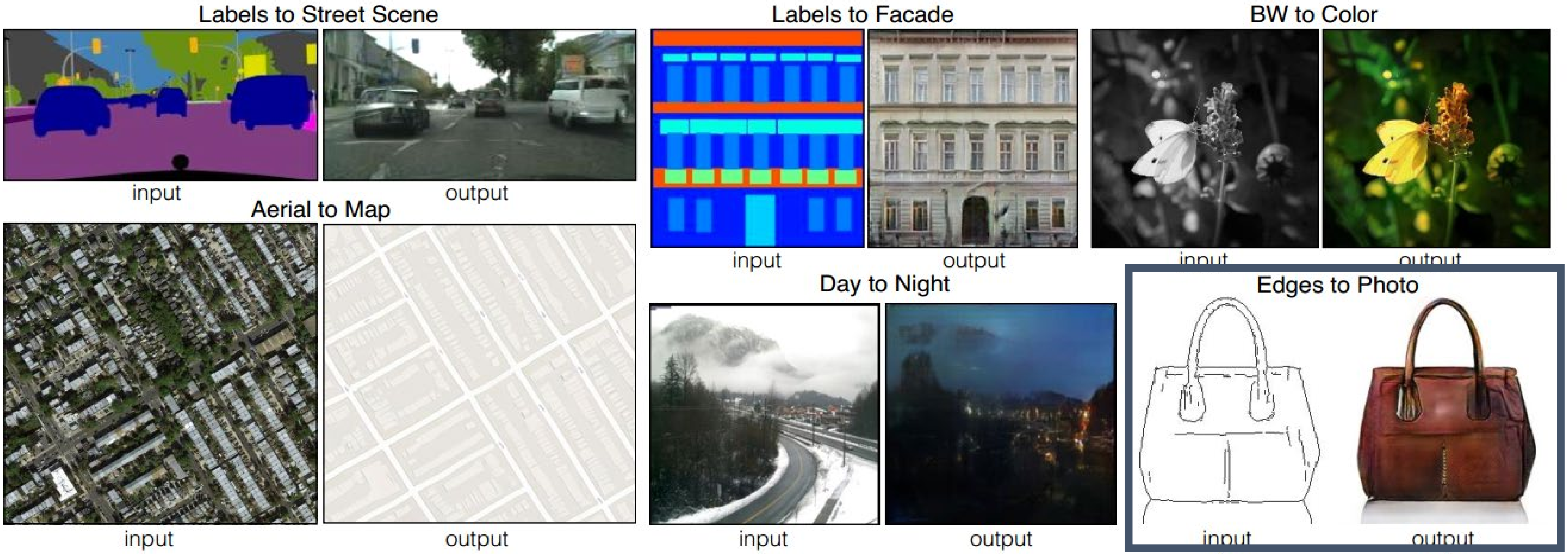
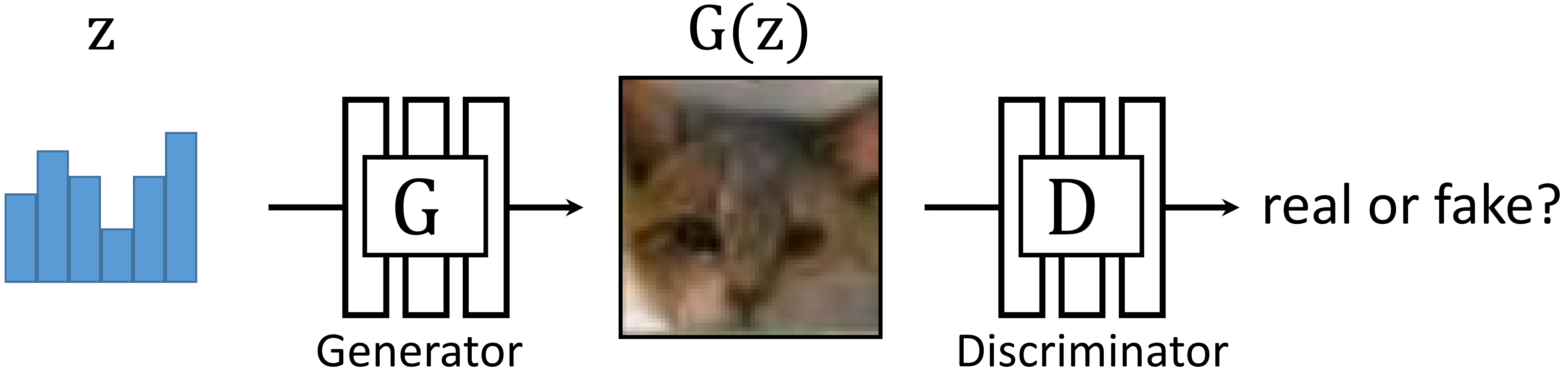


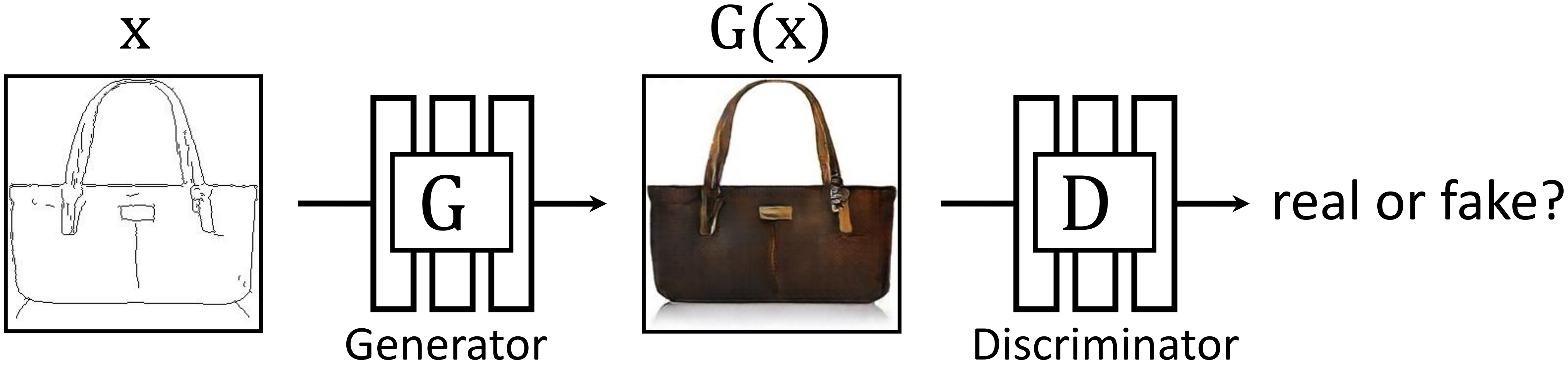
Image-to-image Translation with Conditional Adversarial Nets
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017

slide credit: Phillip Isola & Jun-Yan Zhu



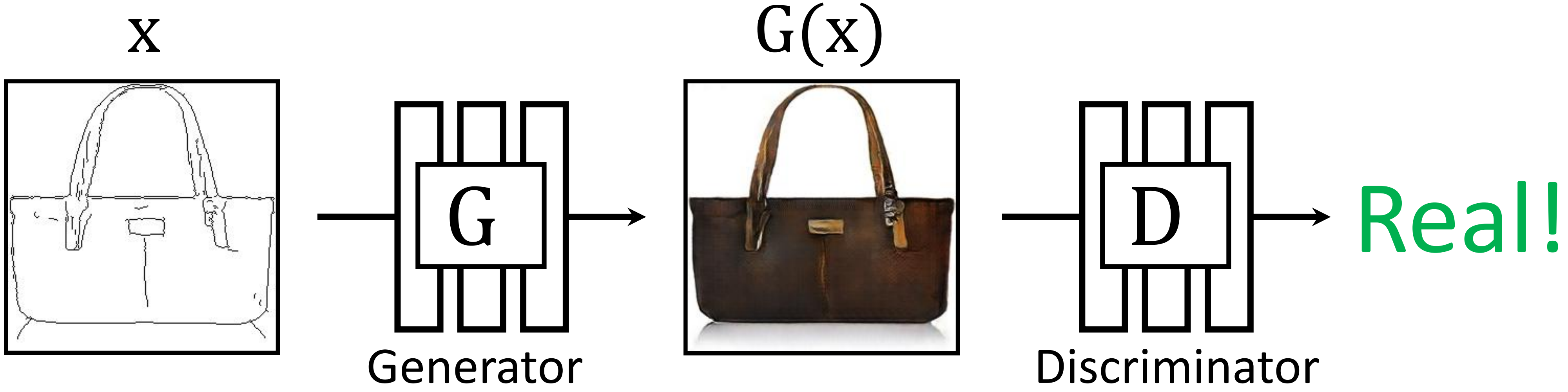
$$\min_{G}\max_{D}\mathbb{E}_{z,x}[\log D(G(z)) + \log(1 - D(x))]$$

slide credit: Phillip Isola & Jun-Yan Zhu



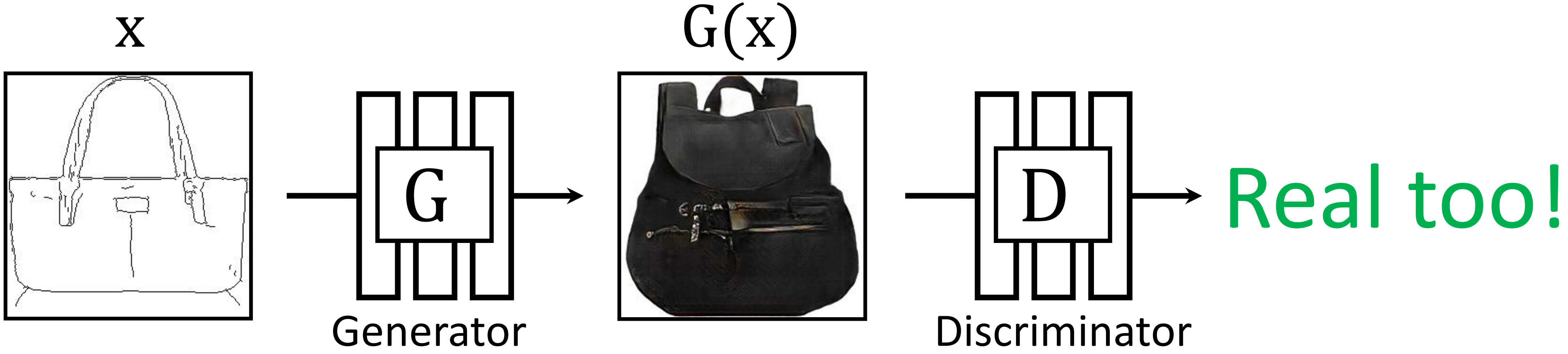
$$\min_G \max_D \mathbb{E}_{x,y}[\log D(G(x)) + \log(1 - D(y))]$$

slide credit: Phillip Isola & Jun-Yan Zhu



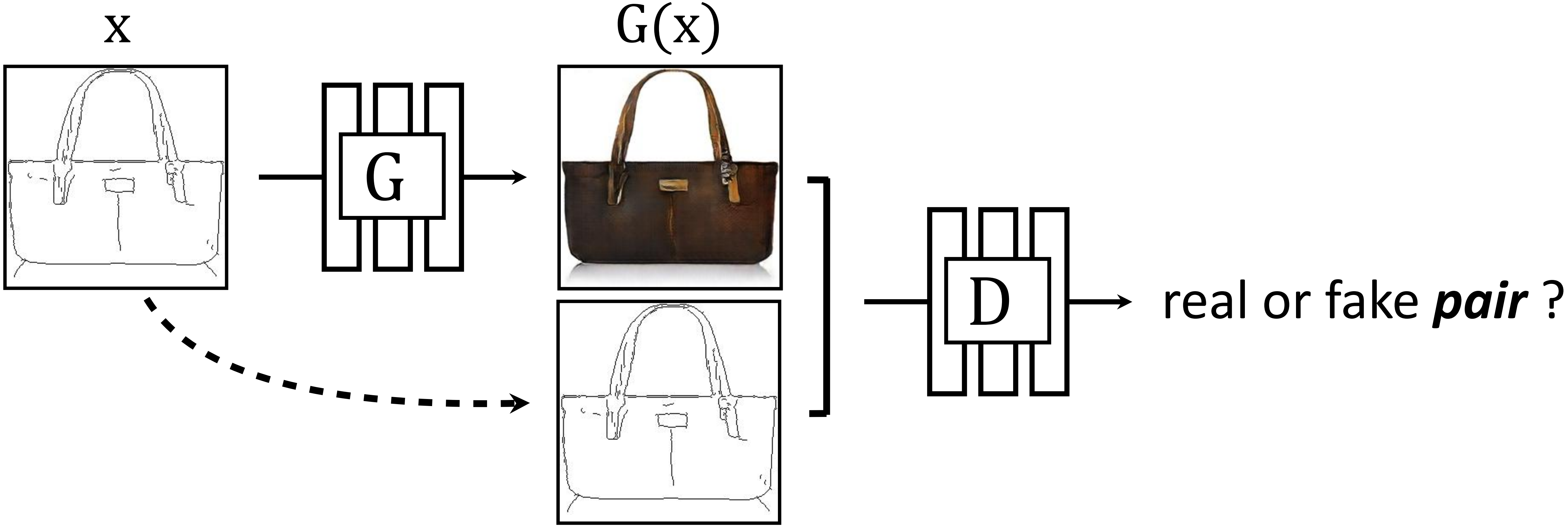
$$\min_G \max_D \mathbb{E}_{x,y}[\log D(G(x)) + \log(1 - D(y))]$$

slide credit: Phillip Isola & Jun-Yan Zhu



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

slide credit: Phillip Isola & Jun-Yan Zhu



$$\min_G \max_D \mathbb{E}_{x,y}[\log \underbrace{D(x, G(x))}_{\text{fake pair}} + \log(1 - \underbrace{D(x,y)}_{\text{real pair}})]$$

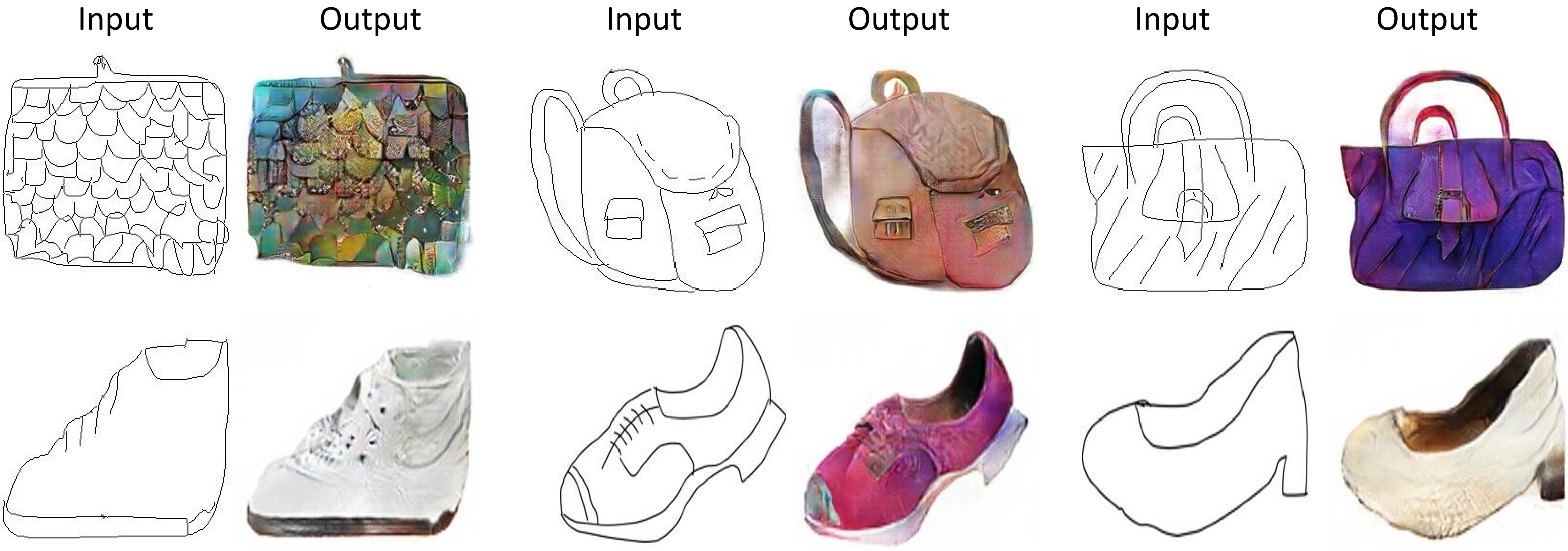
match joint distribution $p(G(x), y) \sim p(x, y)$

Edges → Images

Edges from [Xie & Tu, 2015]

slide credit: Phillip Isola & Jun-Yan Zhu

Sketches → Images



Trained on Edges → Images

Data from [Eitz, Hays, Alexa, 2012]

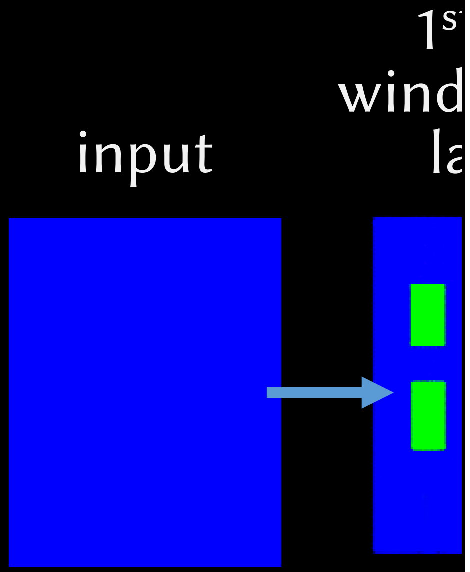
slide credit: Phillip Isola & Jun-Yan Zhu

Decomposition into Steps: FrankenGAN

input

Slide Credit: FrankenGAN, Kelly and Guerrero et al.

Decomposition into Steps: FrankenGAN

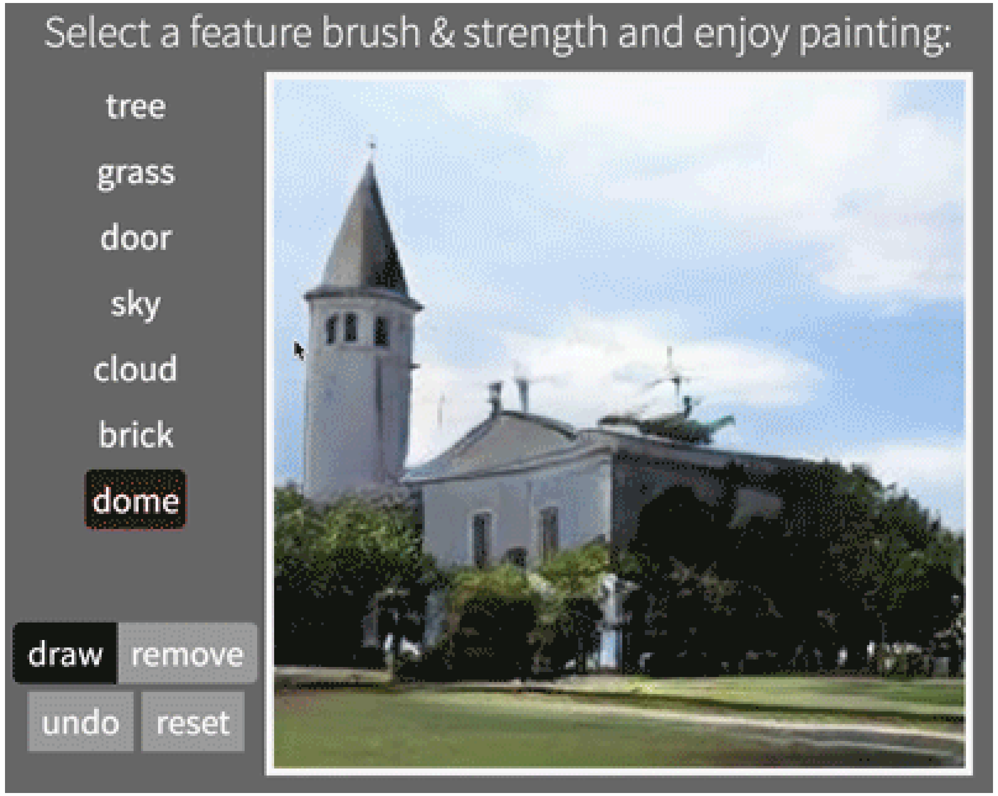


Slide Credit: FrankenGAN, Kelly and Guerrero et al.

GAN Dissection

Question: How does a GAN create an image? What do individual neurons do?

Insight: Neurons are specialized to create objects of specific types



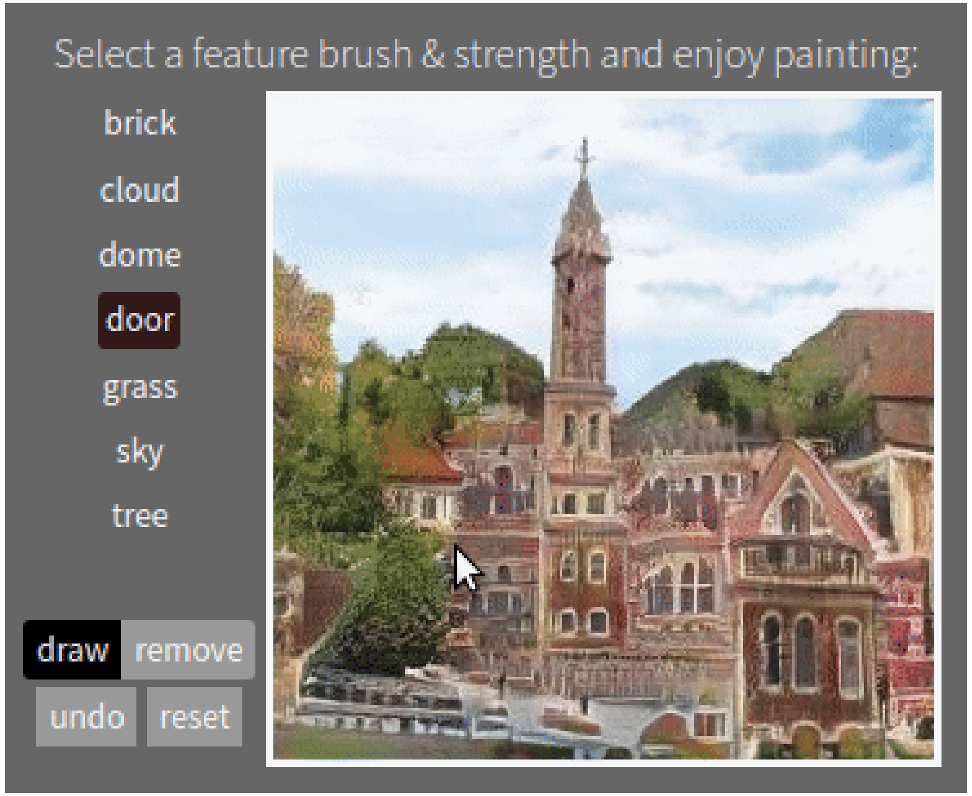


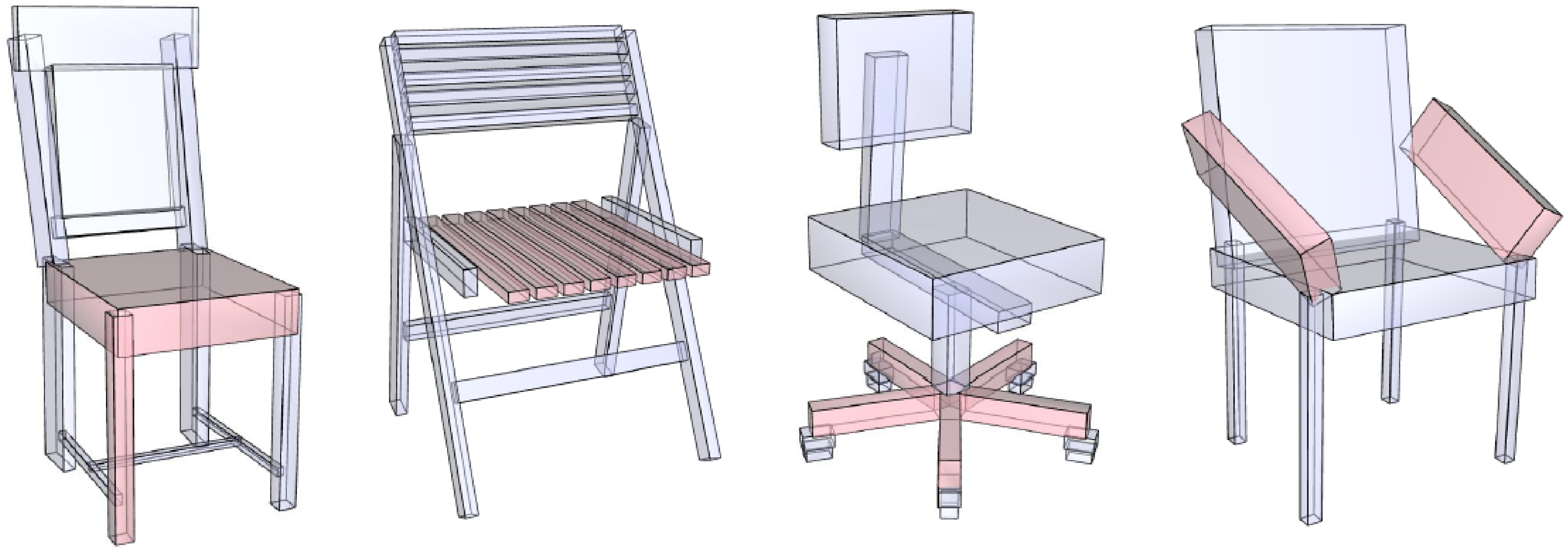
Image Credit:GAN Dissection: Visualizing and Understanding Generative Adversarial Networks, Bau et al.

Open Problems

- More control
- Irregular data
- GAN training convergence
- Evaluating GANs

GRASS

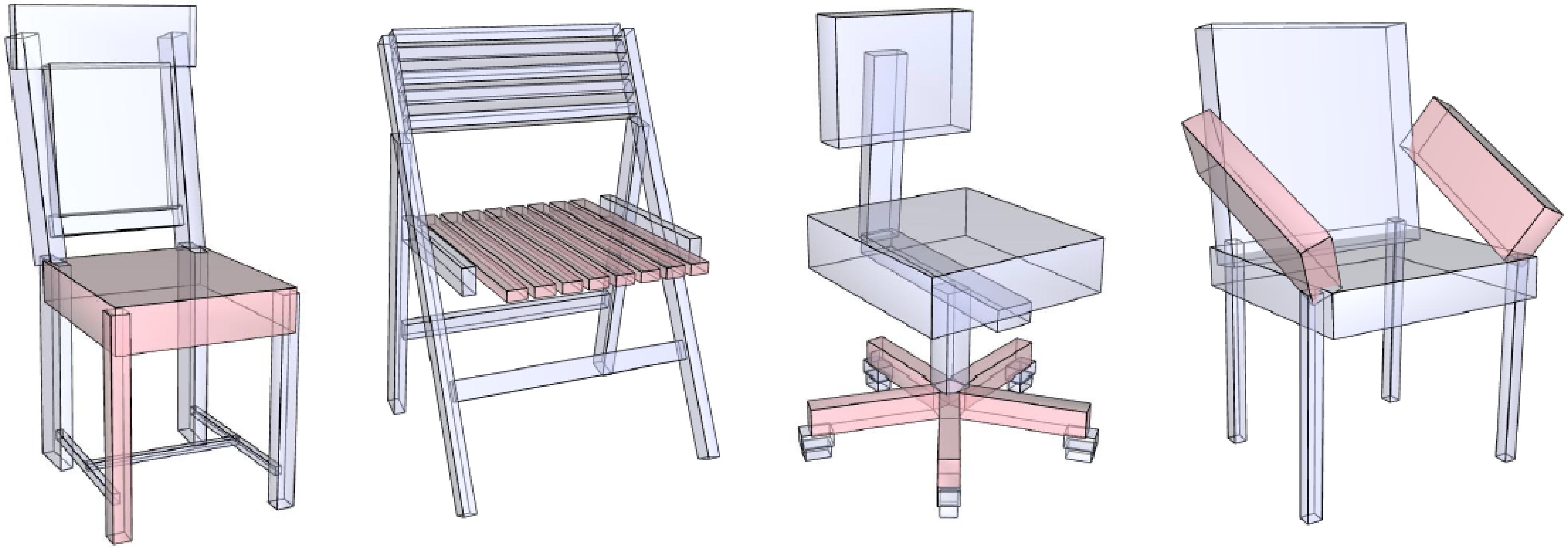
Space of chairs?



Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

GRASS

Part bounding boxes and their relationships represent a shape

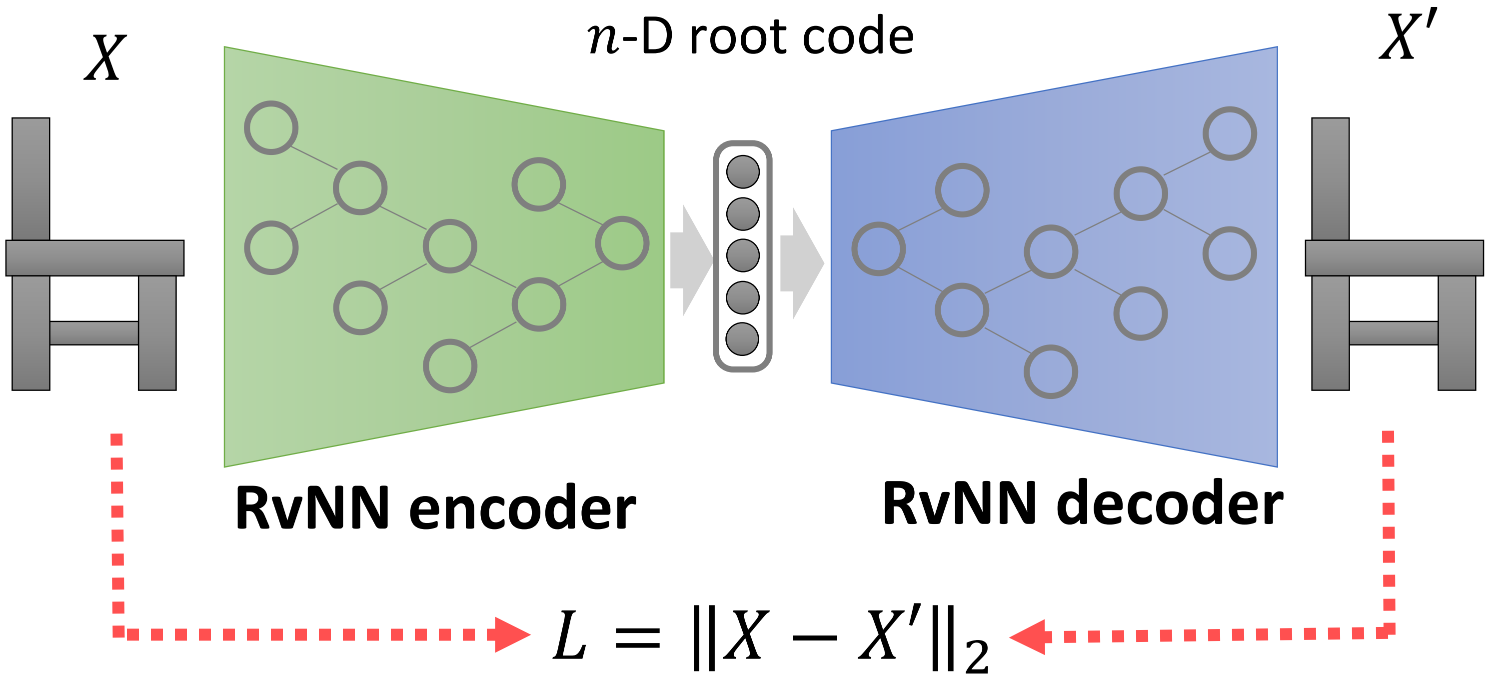


Adjacency | Translational symmetry | Rotational symmetry | Reflectional symmetry

Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

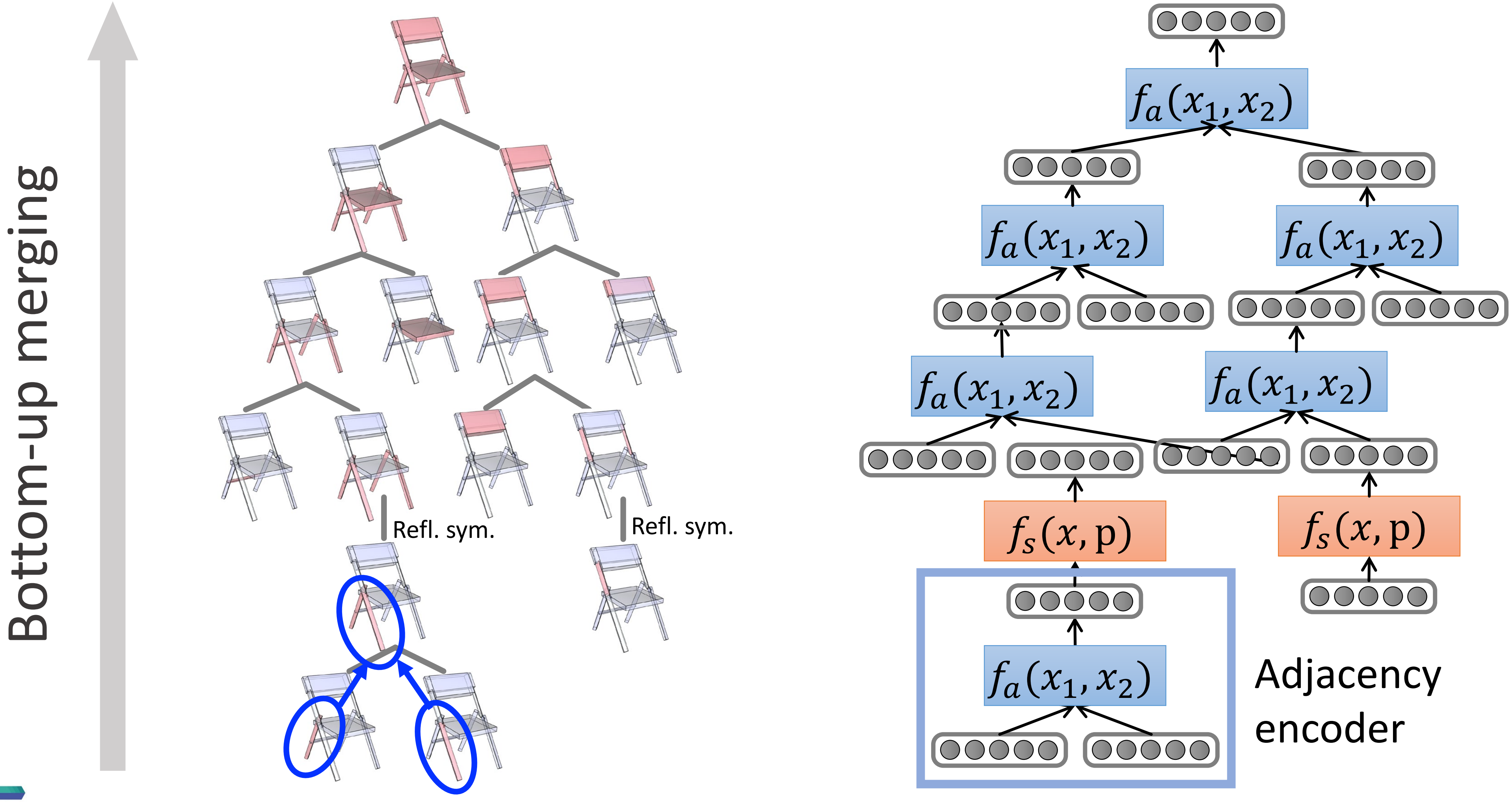
GRASS Training

VAE using a hierarchical encoder and decoder



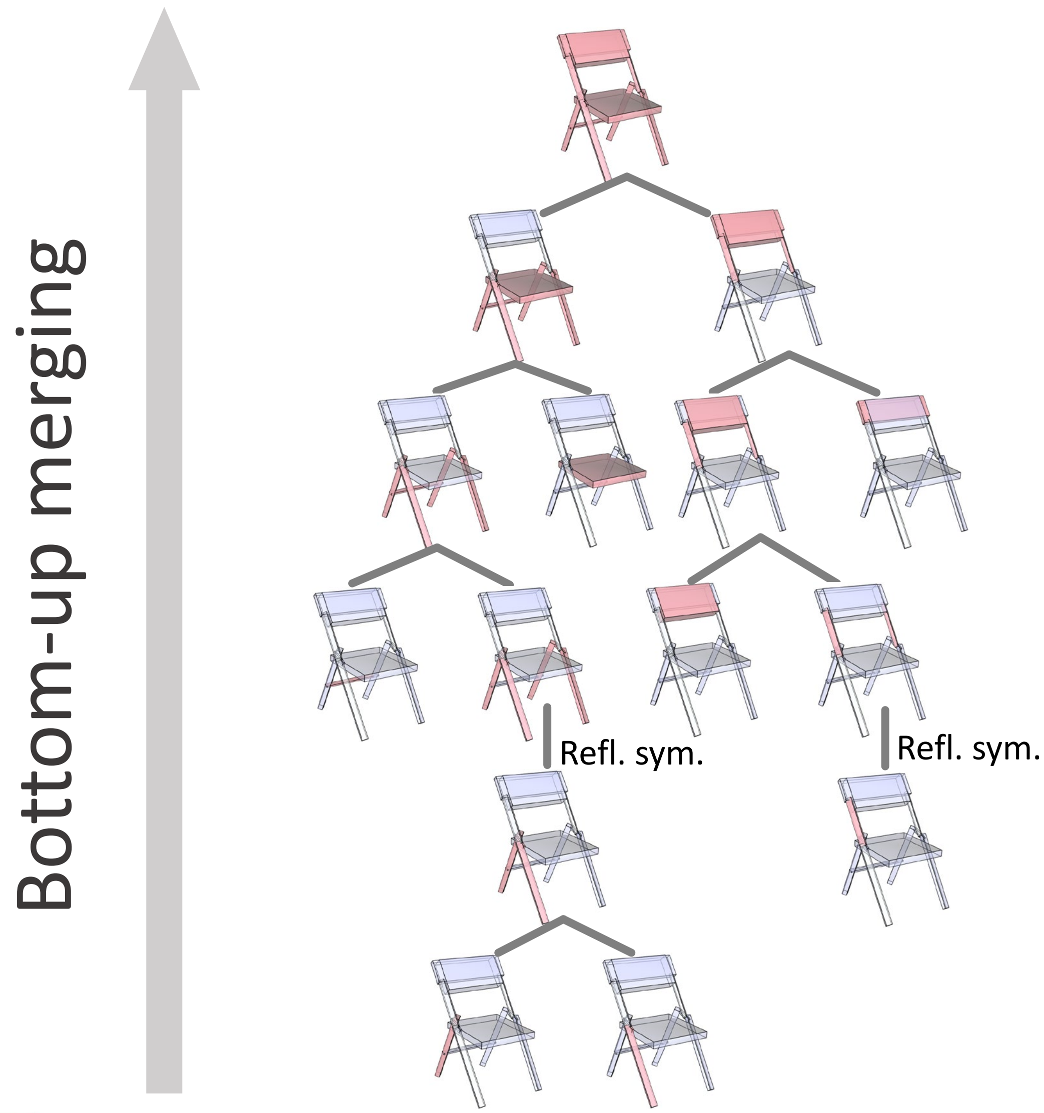
Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

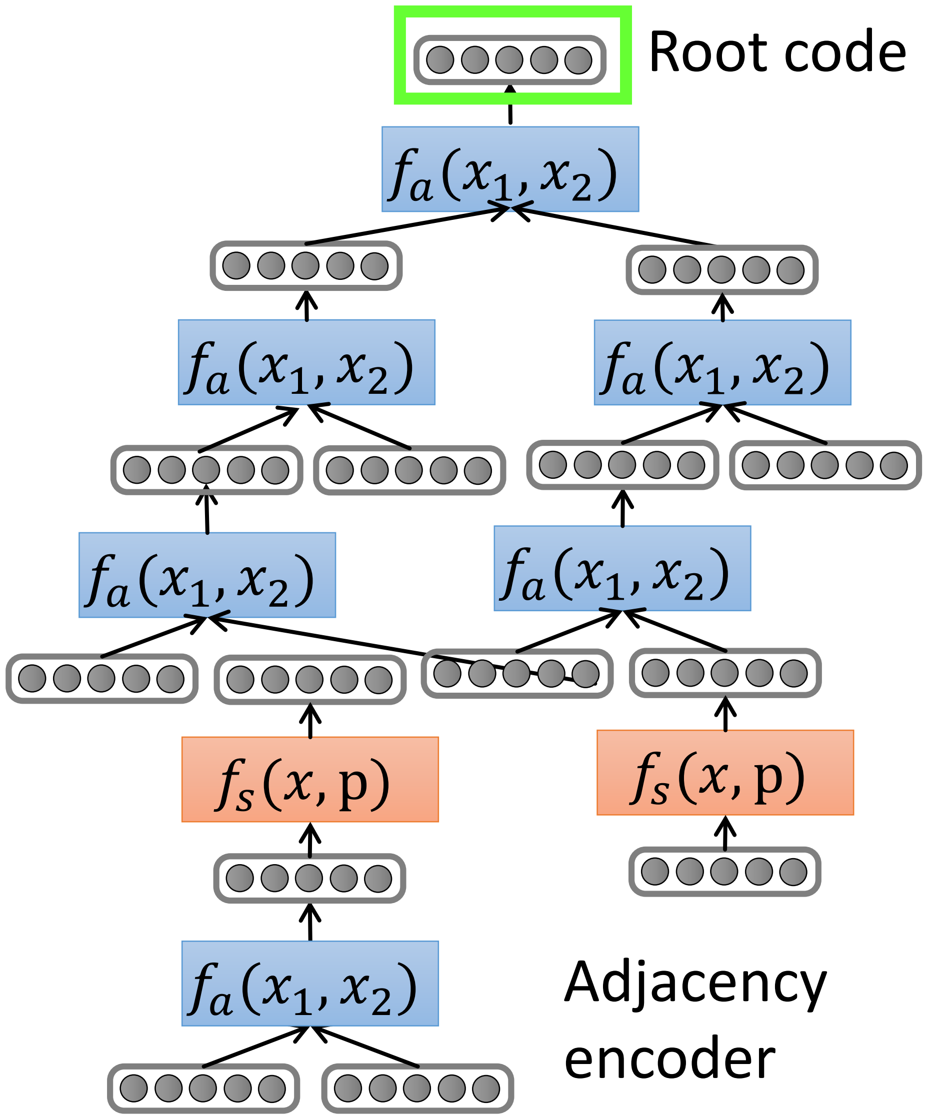
GRASS Hierarchical Encoder



Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

GRASS Hierarchical Encoder

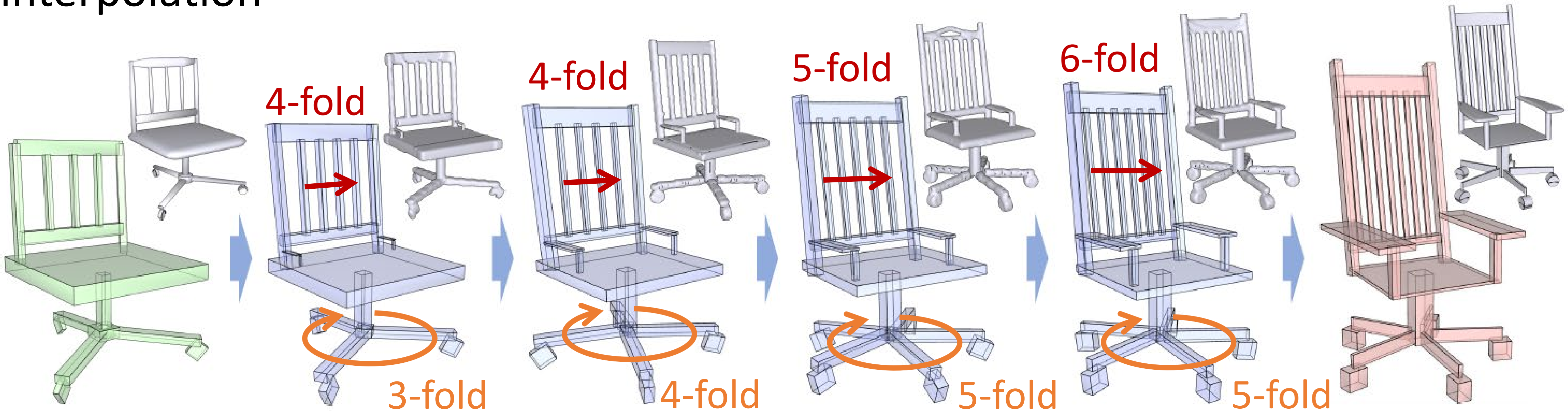




Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

GRASS Results

interpolation



free generation

Slide Credit: GRASS: Generative Recursive Autoencoders for Shape Structures, Li et al.

StructureNet

- Consistent structure across the dataset

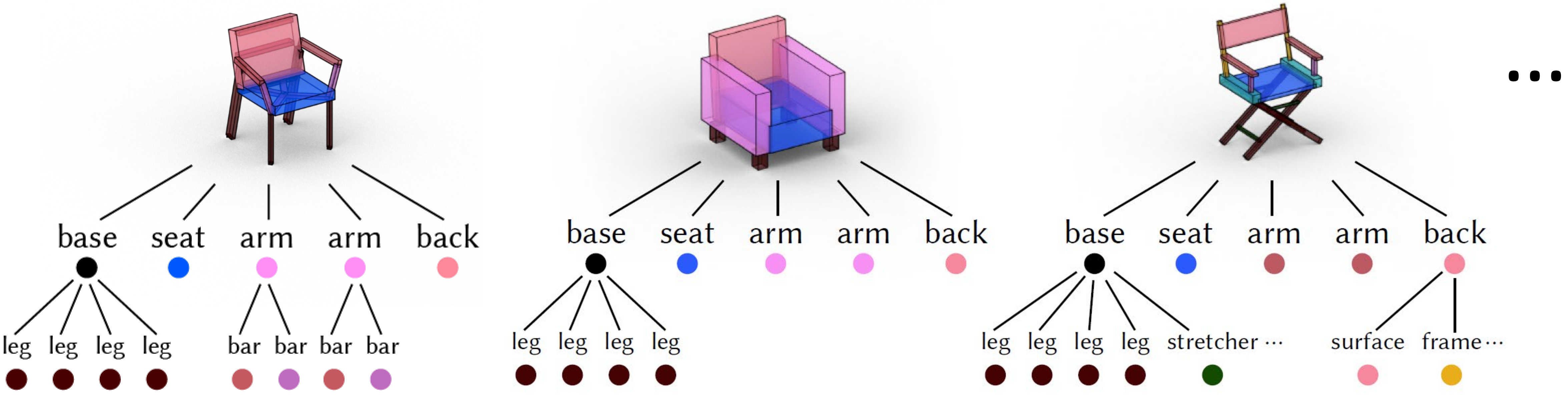


Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

StructureNet

- Consistent structure across the dataset
- Richer structure

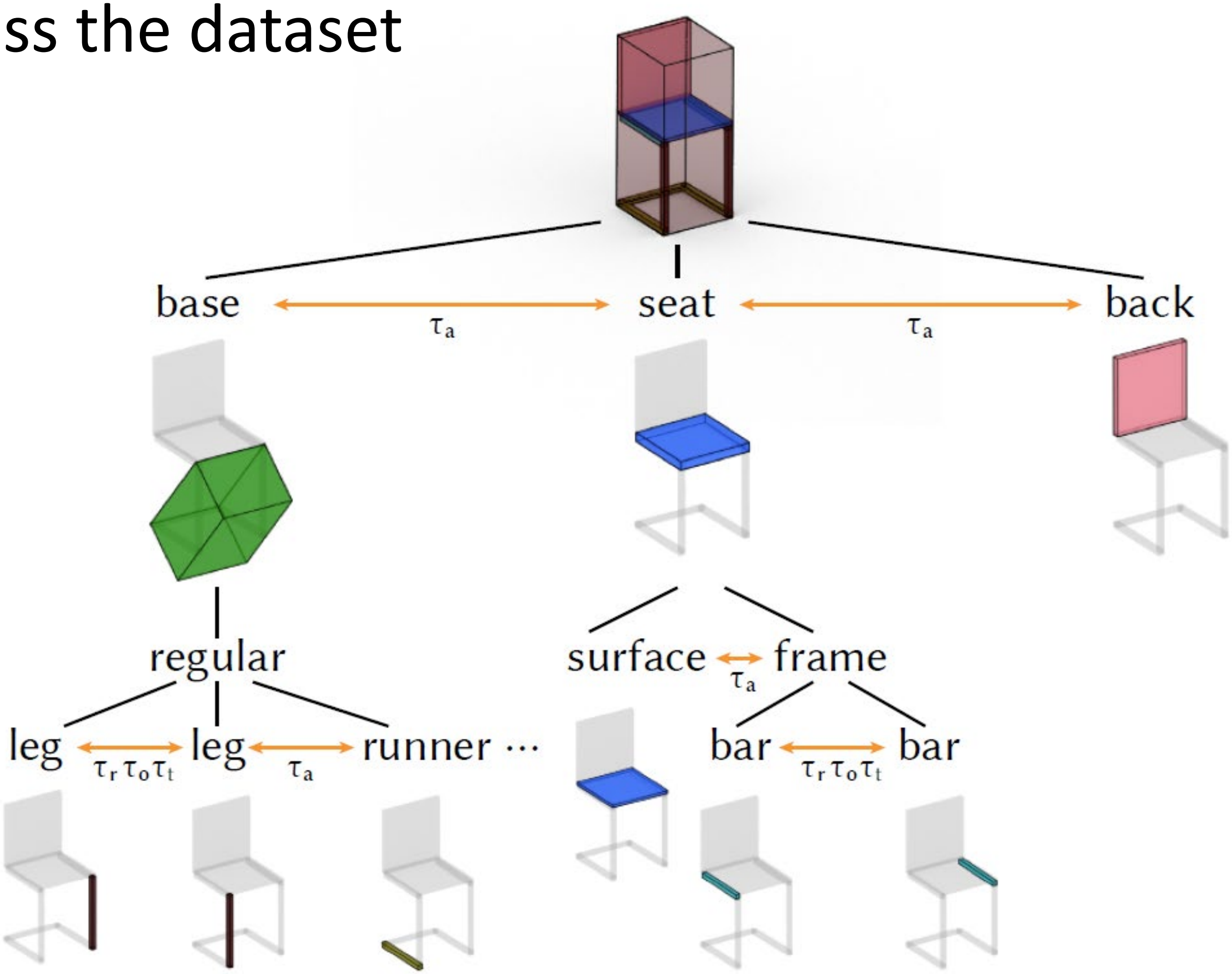


Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

Hierarchical Graph Networks as Encoder and Decoder

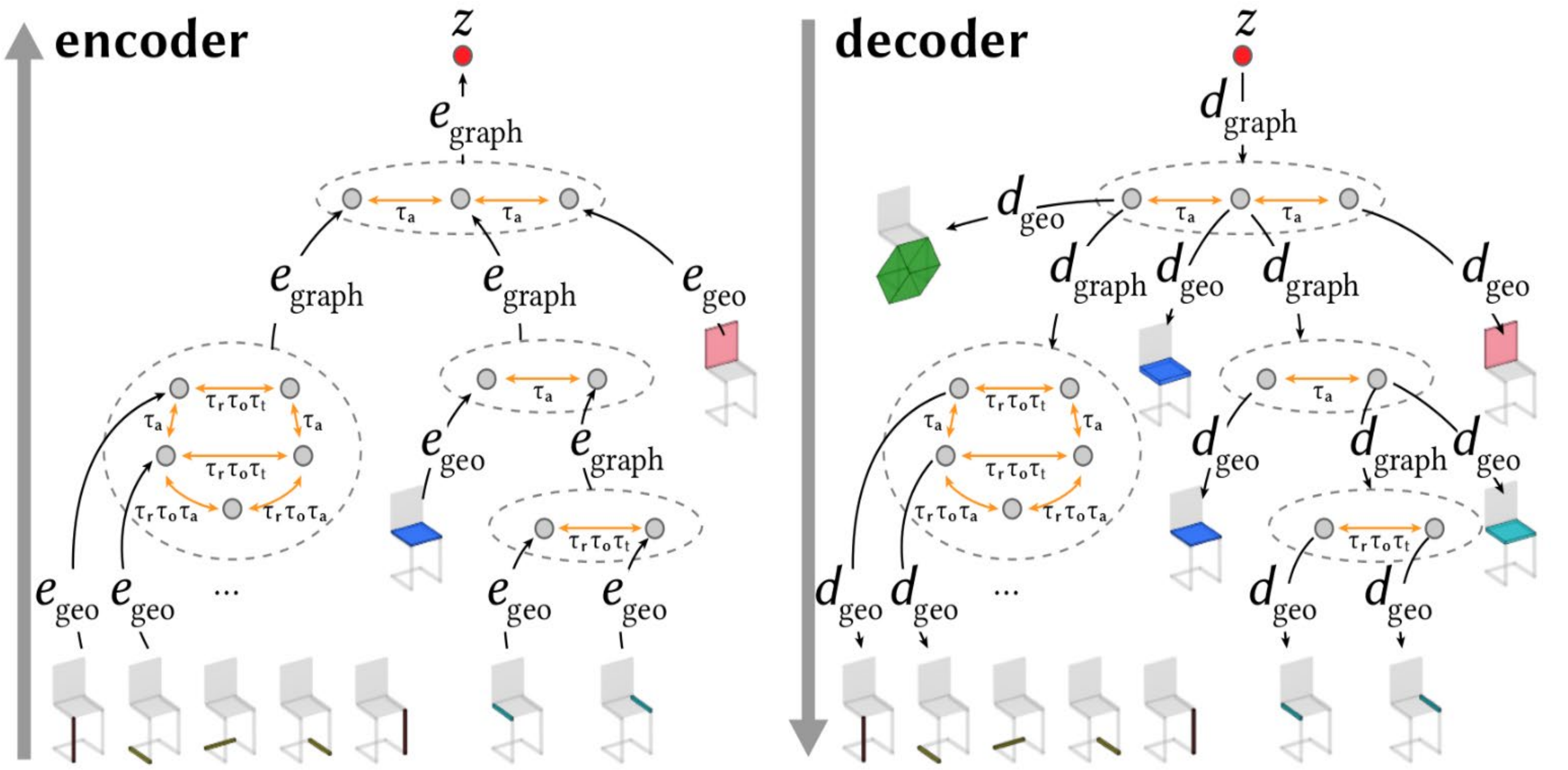
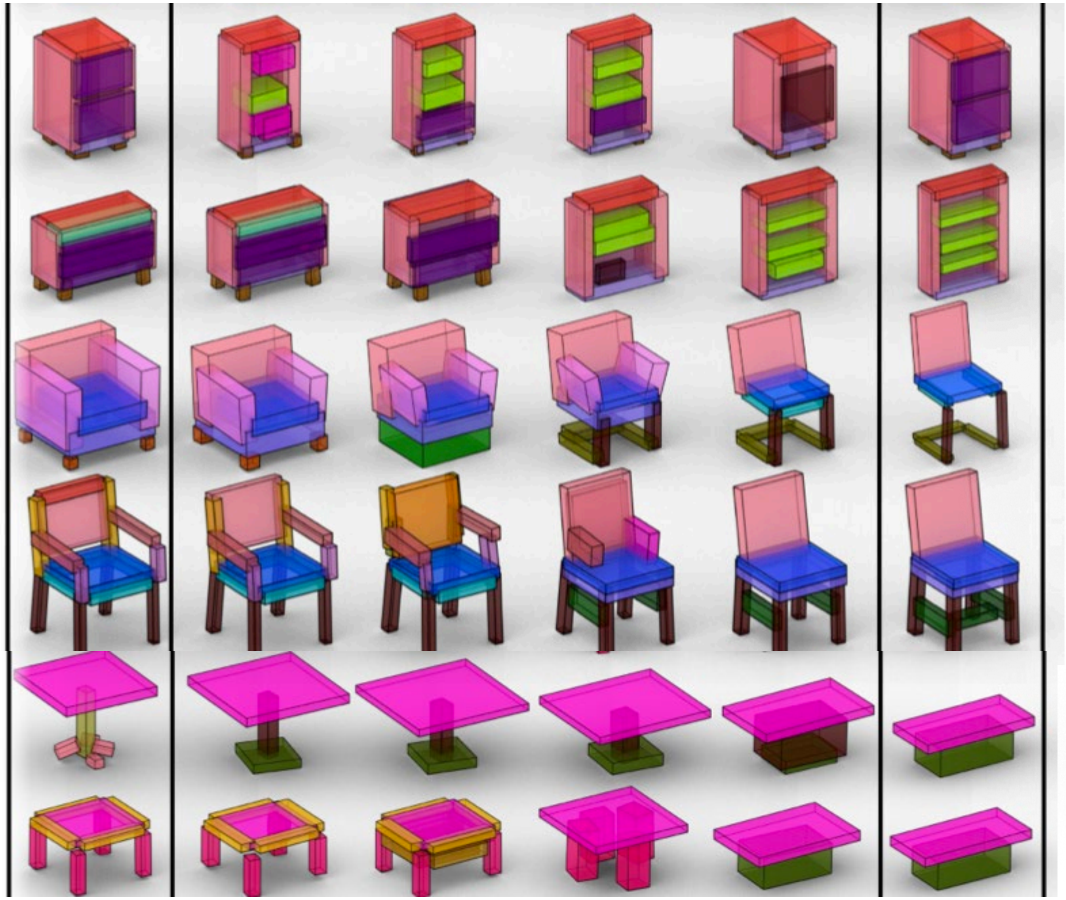


Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

StructureNet Results

interpolation



free generation

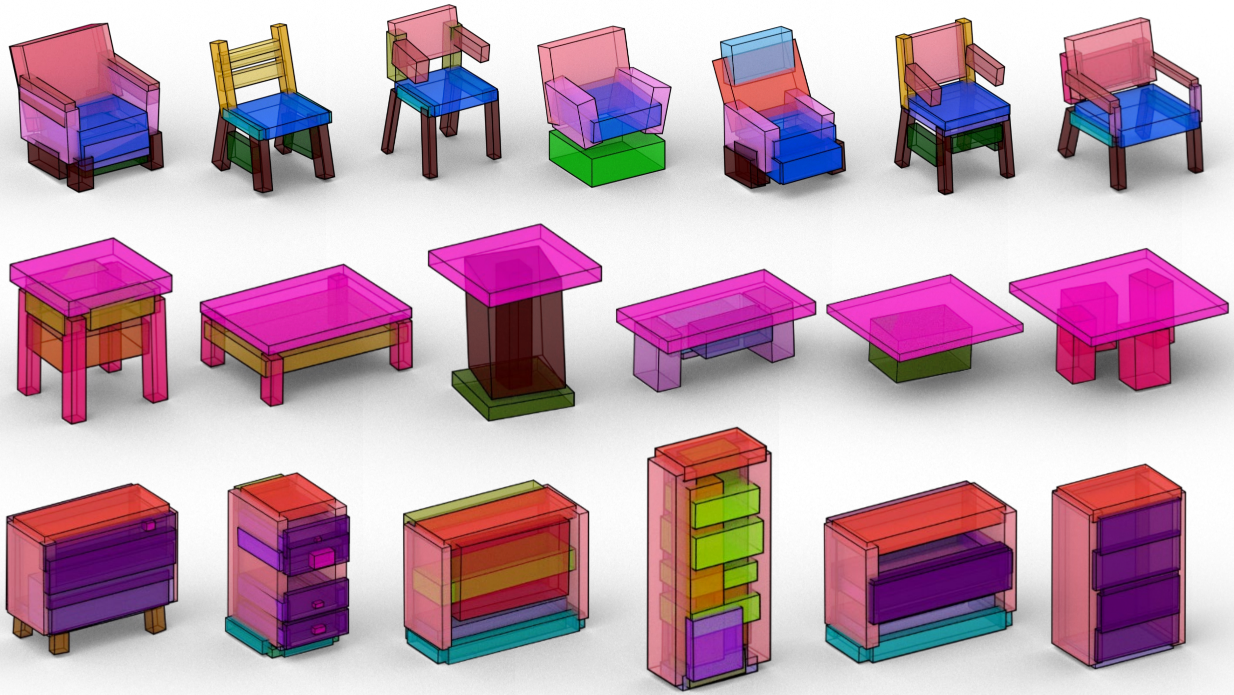


Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

Open Problems

- More control
- Irregular data
- GAN training convergence
- Evaluating GANs

GAN Training Convergence

GAN training can be unstable

- Generator and discriminator do now always converge (Nash equilibrium)
- Vanishing discriminator gradients
- Mode Collapse

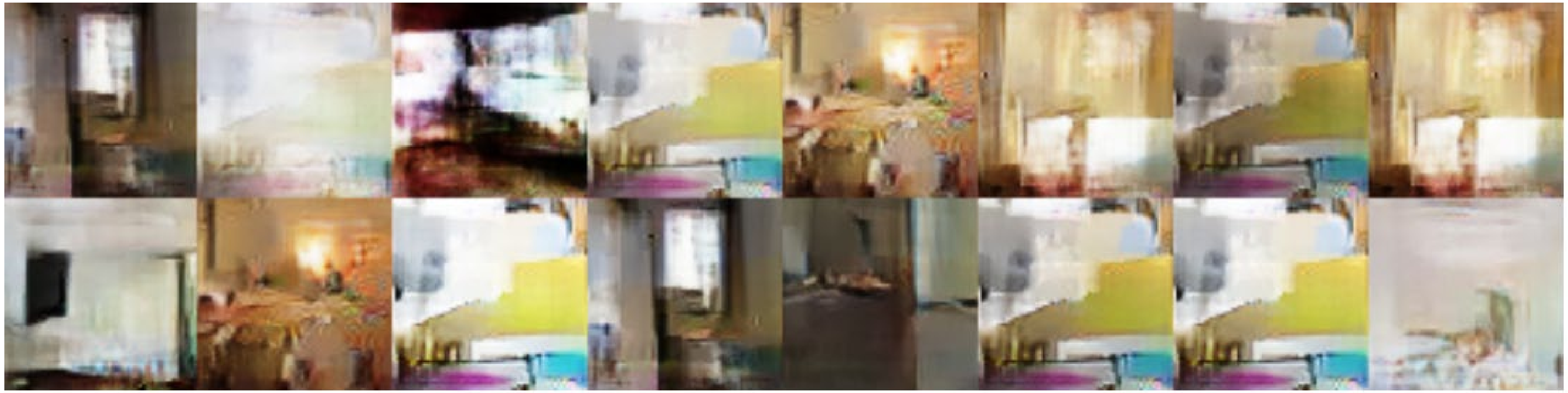
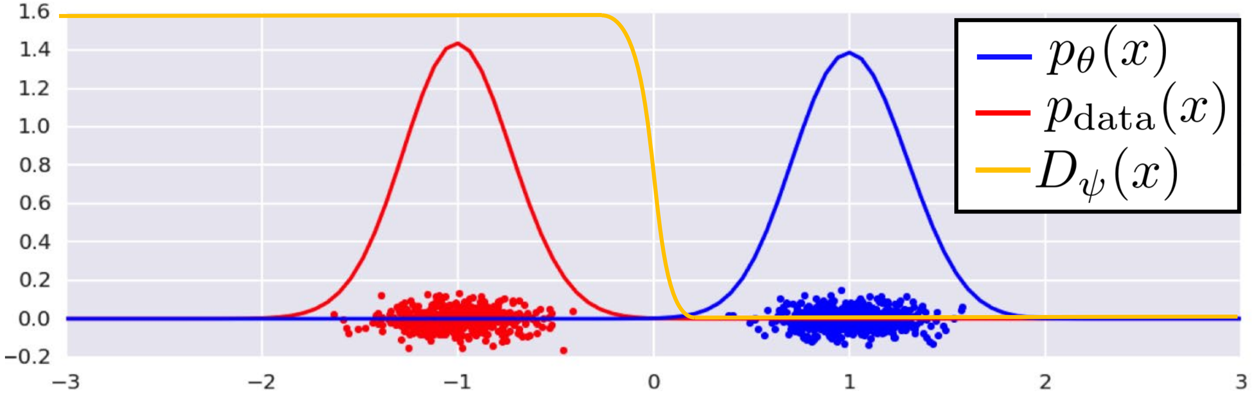


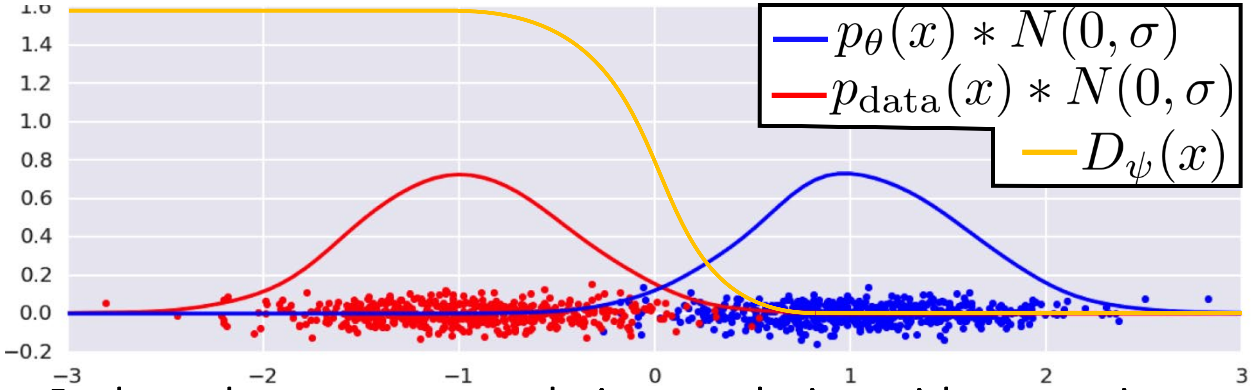
Image Credit: *Wasserstein GAN,* Arjovsky et al.

Vanishing Discriminator Gradients

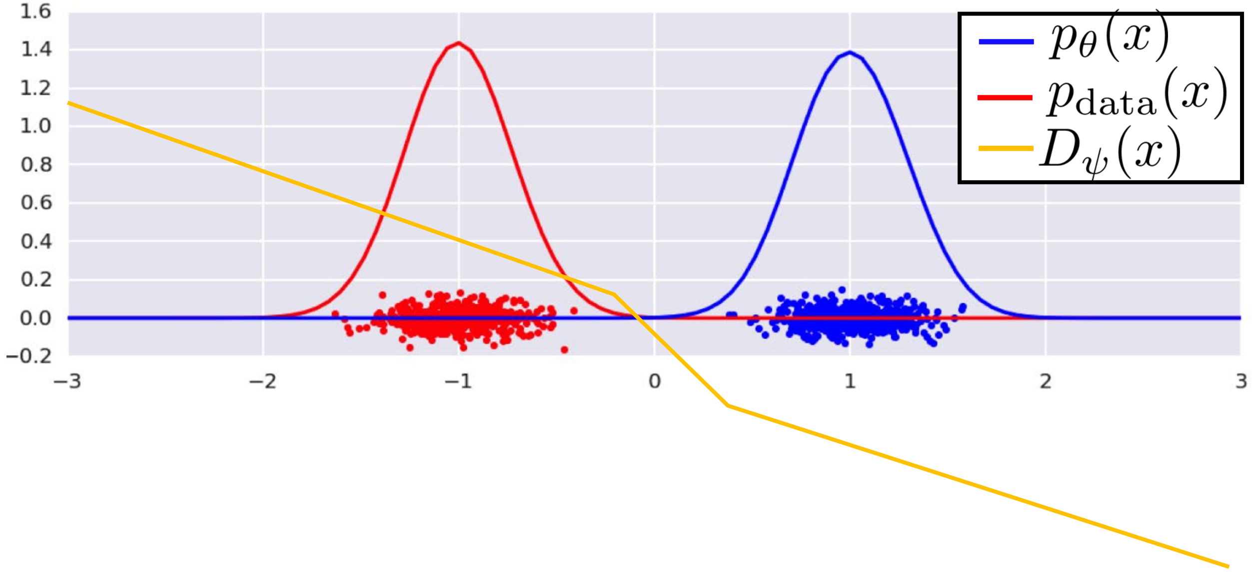
Standard



Instance noise: adding noise to generated and real images



Wasserstein GANs: EMD as distance between p_θ and p_{data}

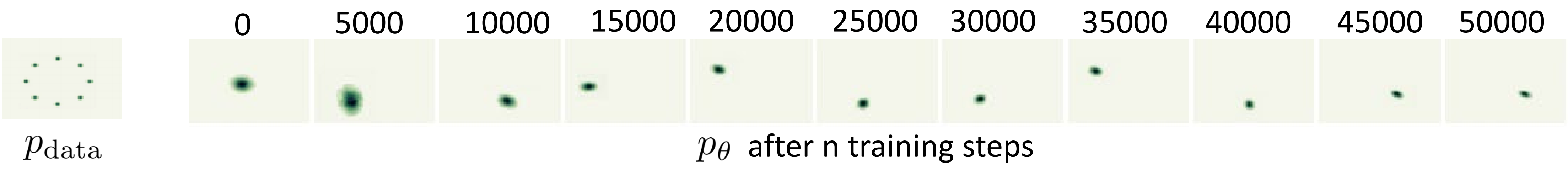


Roth et al. suggest an analytic convolution with a gaussian:
Stabilizing Training of Generative Adversarial Networks through Regularization, Roth et al. 2017

Image Credit: *Amortised MAP Inference for Image Super-resolution*, Sønderby et al.

Mode Collapse

p_θ only covers one or a few modes of p_{data}



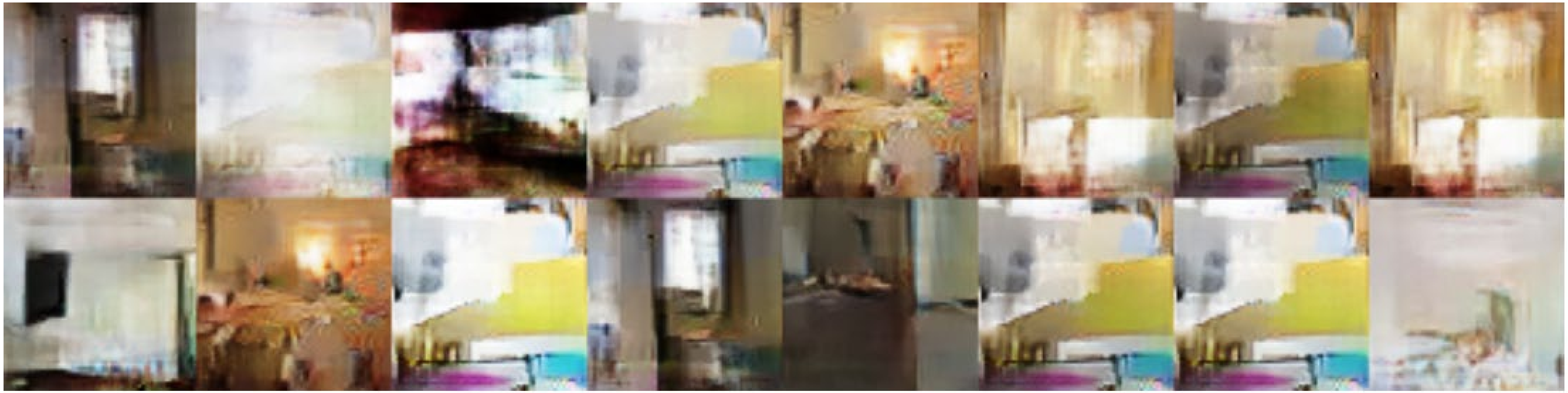
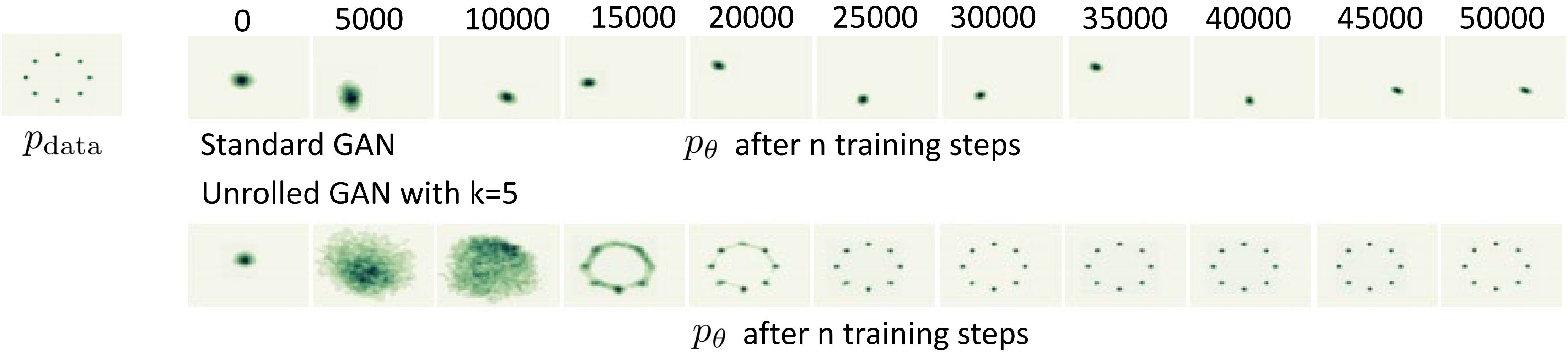


Image Credit: *Wasserstein GAN,* Arjovsky et al.
Unrolled Generative Adversarial Networks, Metz et al.

Mode Collapse

Solution attempts:

- Minibatch comparisons: Discriminator can compare instances in a minibatch (*Improved Techniques for Training GANs*, Salimans et al.)
- Unrolled GANs: Take k steps with the discriminator in each iteration, and backpropagate through all of them to update the generator

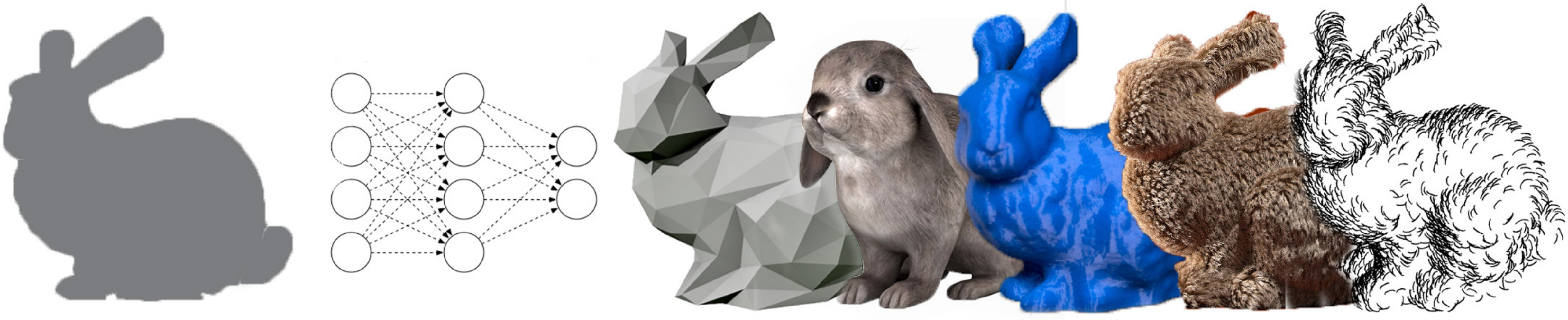


Unrolled Generative Adversarial Networks, Metz et al.

Summary

- Autoencoders
  - Can create a feature space, but bad generators
- VAEs
  - Lower quality generated samples (usually blurry)
  - Relatively stable to train
- Normalized Flows
  - Better quality generated samples
  - Invertible
  - Relatively stable, but expensive to train
- GANs
  - Can not find a latent representation for a given sample (no encoder)
  - Usually better generators than VAEs
  - Currently unstable training (active research)

Course Information (slides/code/comments)



http://geometry.cs.ucl.ac.uk/creativeai/