COMP0169: Machine Learning for Visual Computing

Generative Modeling

UCL

Lectures will be Recorded

Recap

MLP

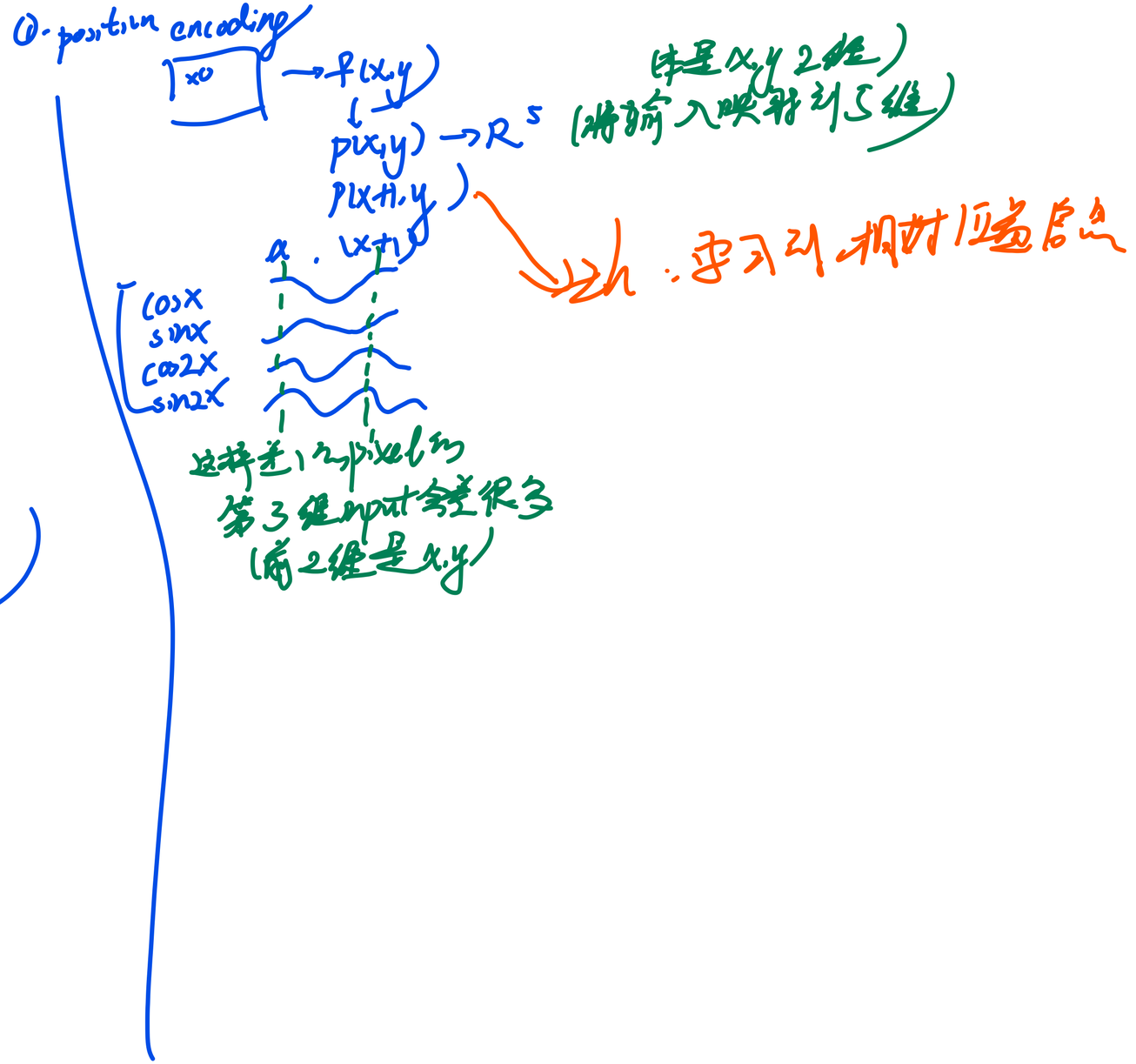
CNN

visualize

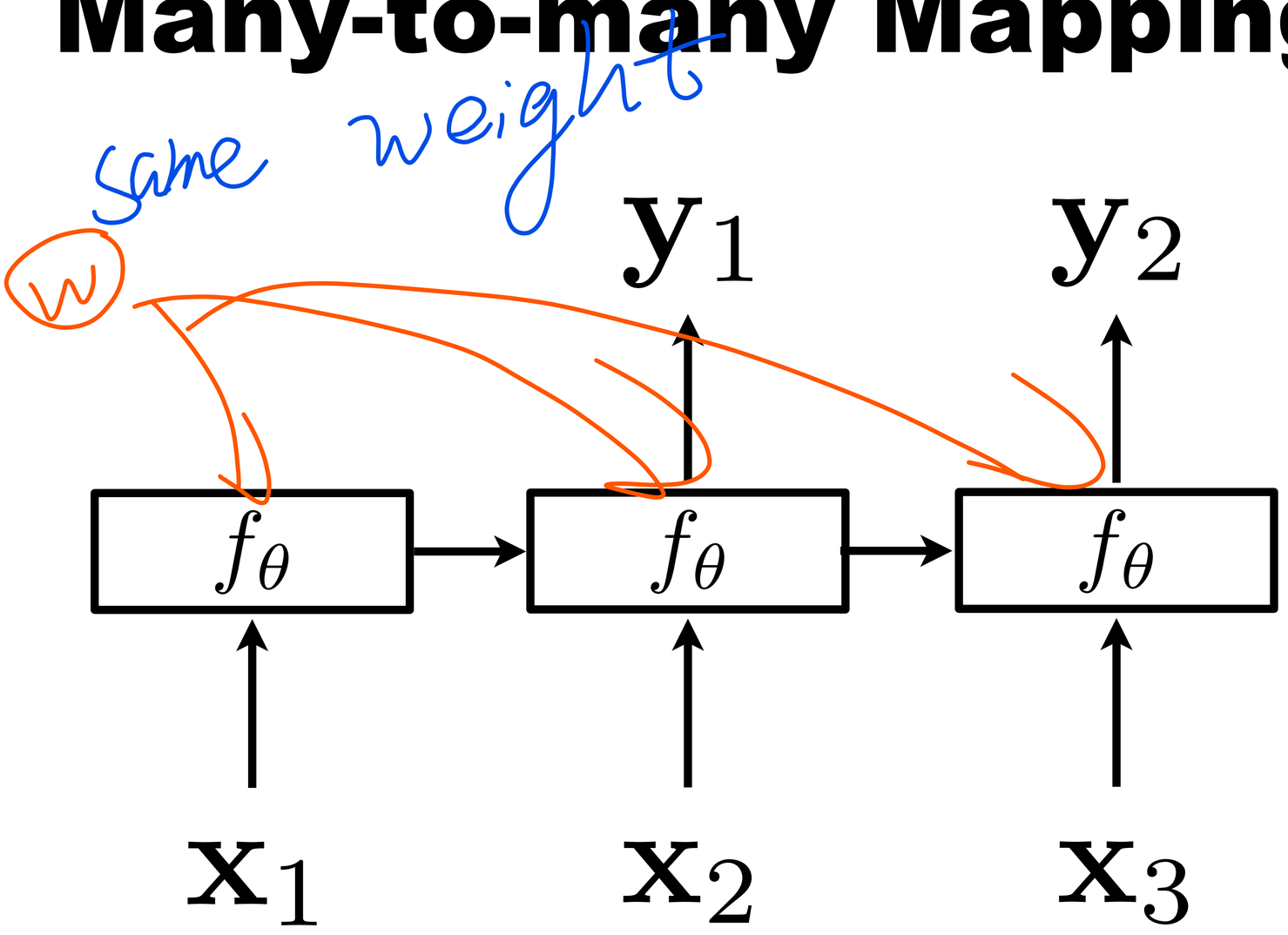
loss

sequence

CNN

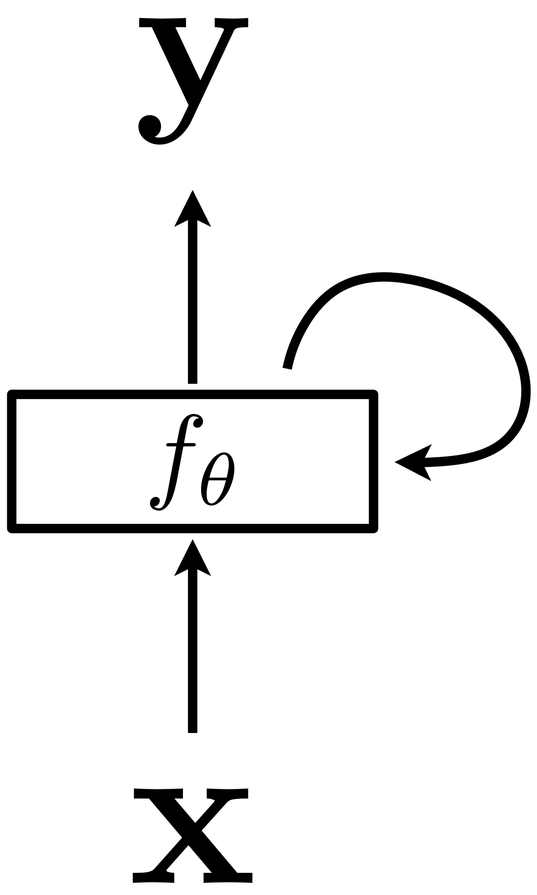


Many-to-many Mapping



- **Conditional Image Generation**
- **Machine Translation**
- **Animation Synthesis**
- **Skeleton to body animation**

Recurrent Neural Network (RNN)



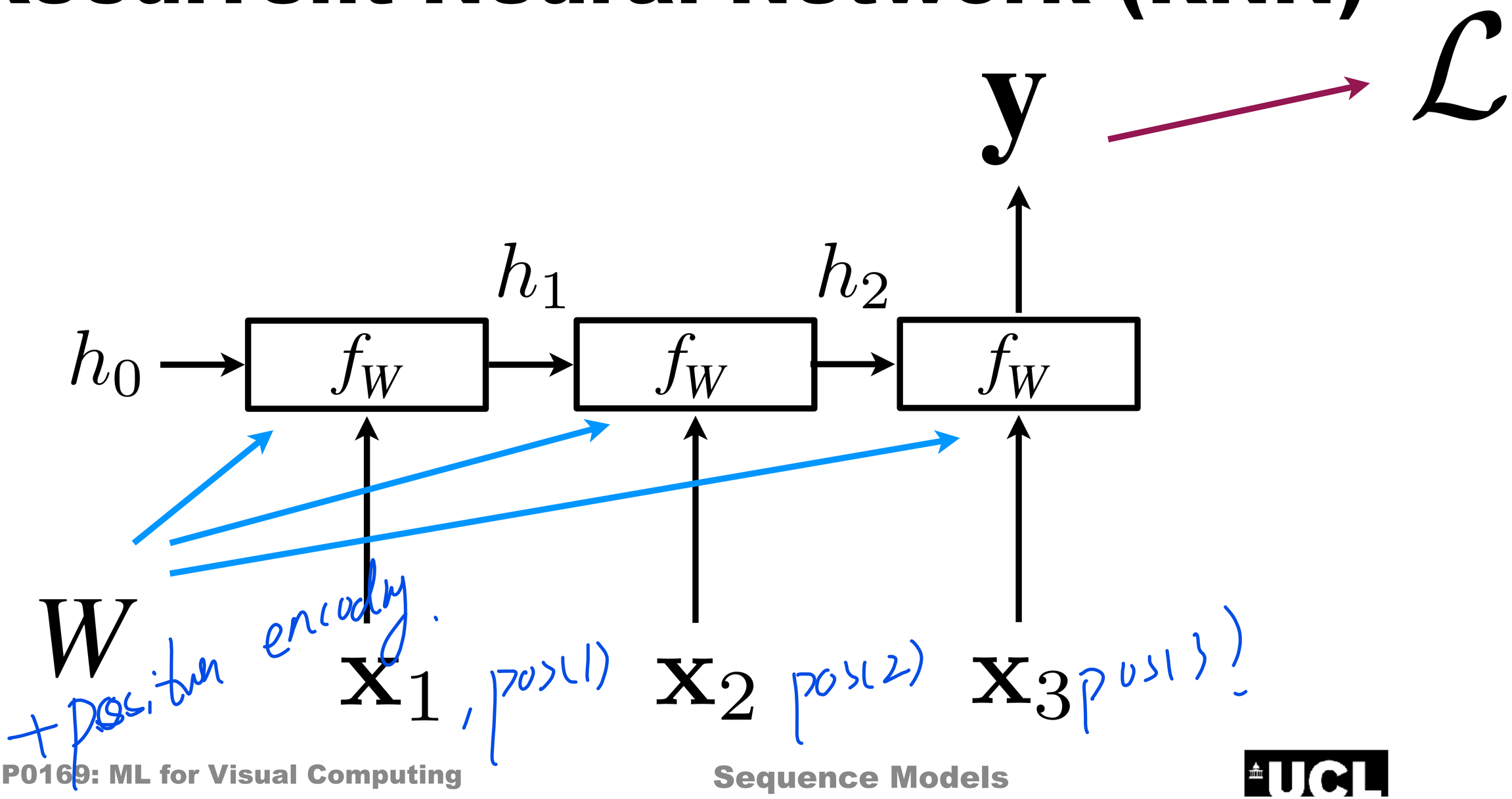
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

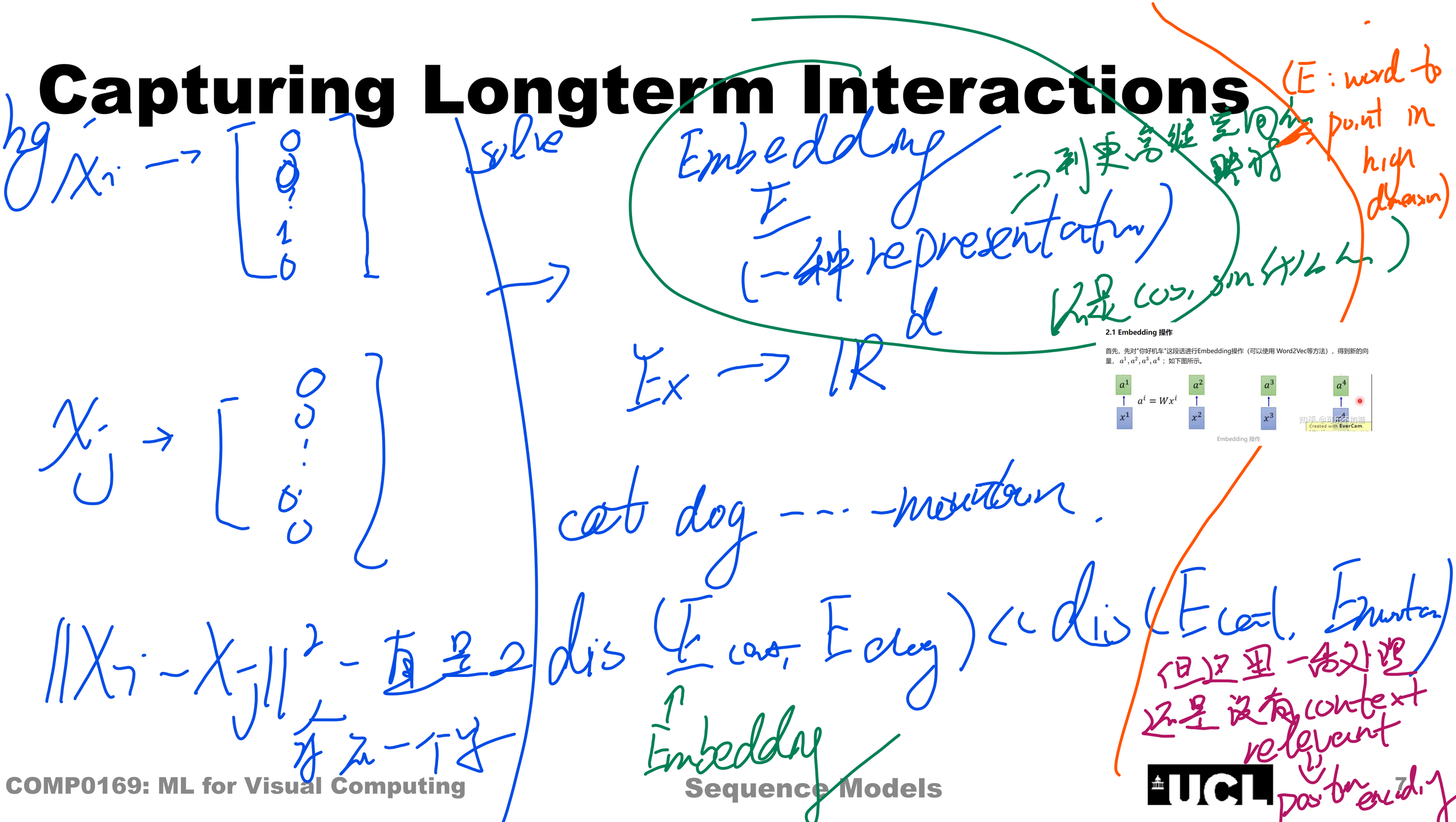
$$y_t = W_{hh}h_t$$

Further reading: https://pabloinsente.github.io/the-recurrent-net

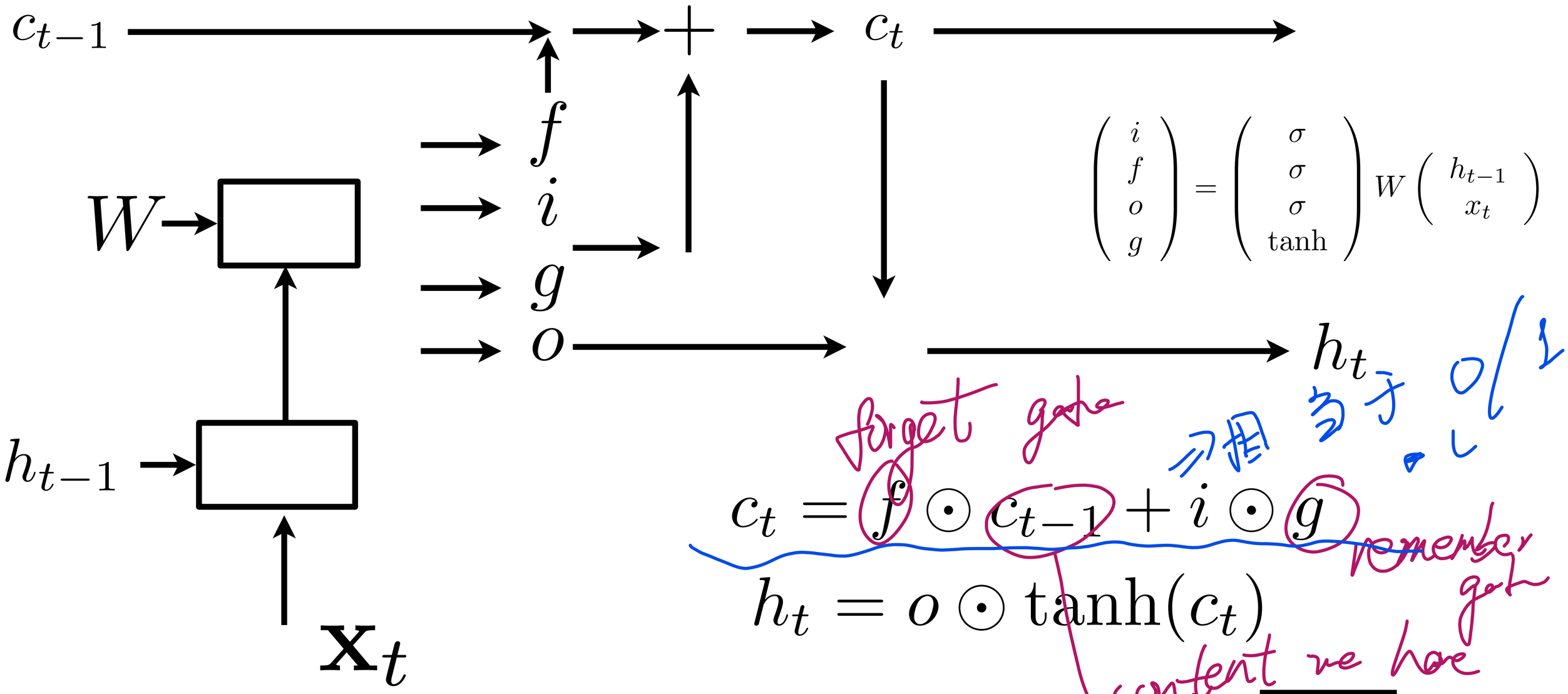
Recurrent Neural Network (RNN)



Capturing Longterm Interactions



LSTM



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

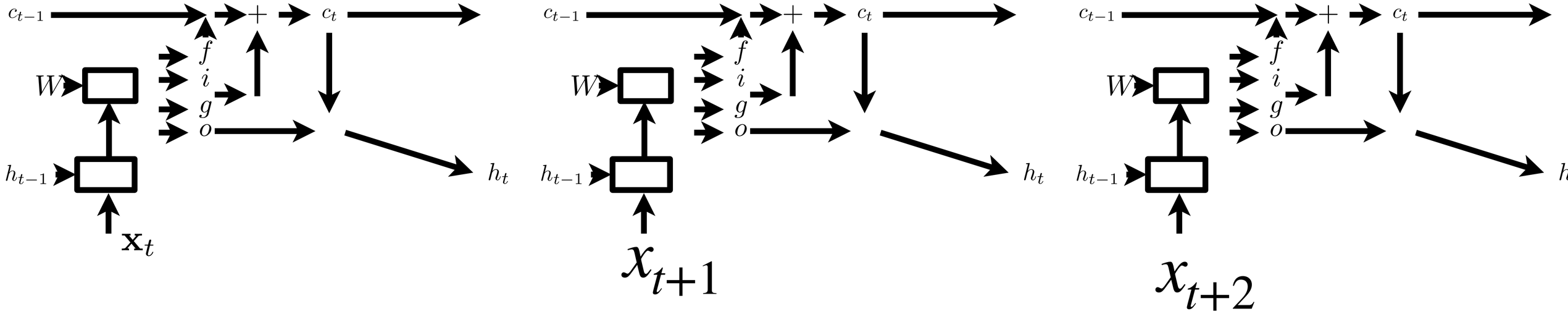
$$h_t = o \odot \tanh(c_t)$$

forget gate

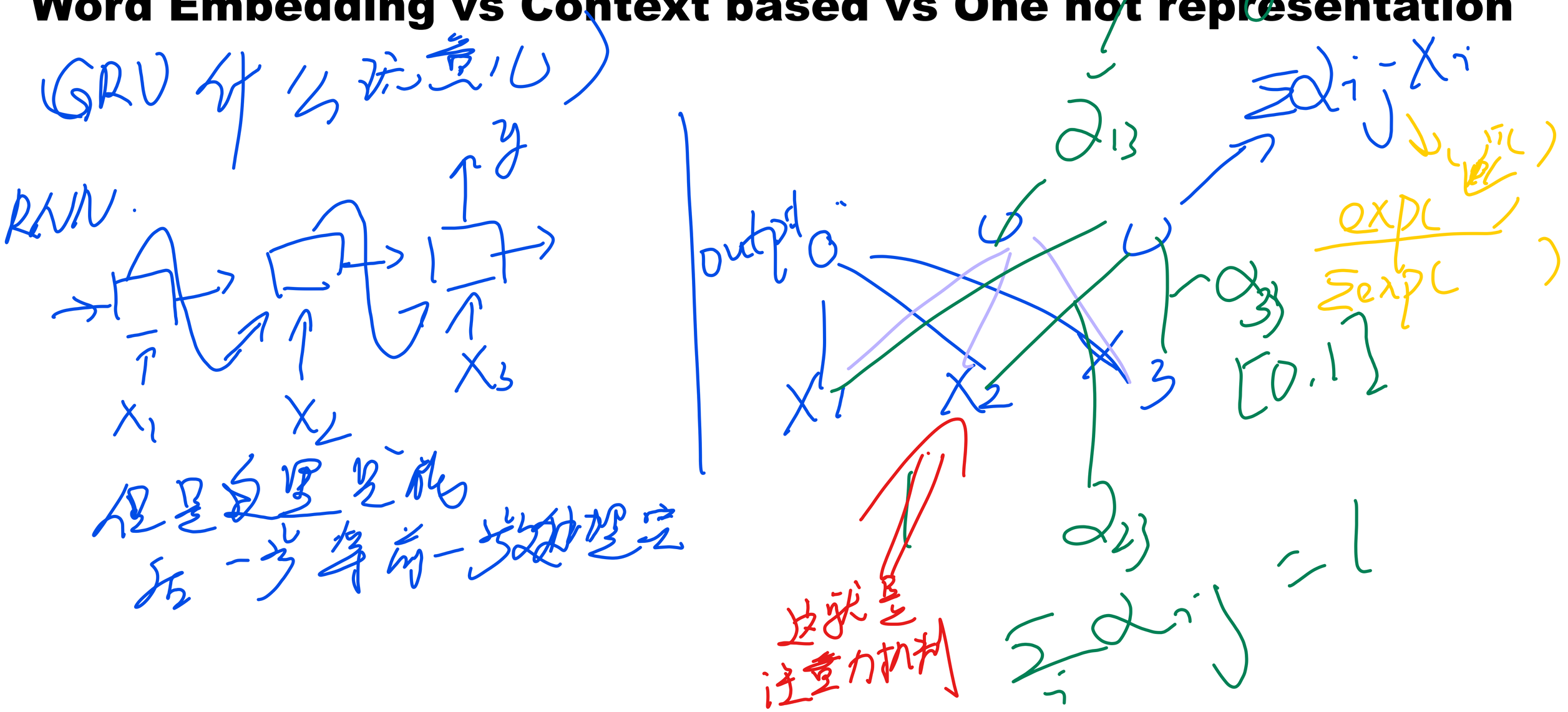
remember gate

content we have

Chaining LSTMs



Word Embedding vs Context based vs One hot representation



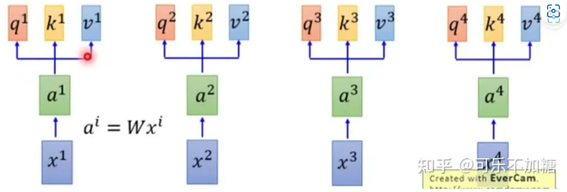
Attention / Transformer

第一步，每个 a^1, a^2, a^3, a^4 都会分别乘以三个矩阵，分别是 q, k, v；需要注意的是，矩阵 q, k, v 在整个过程中是共享的；公式如下：

$$q^i = W^q a^i$$

$$k^i = W^k a^i$$

$$v^i = W^v a^i$$

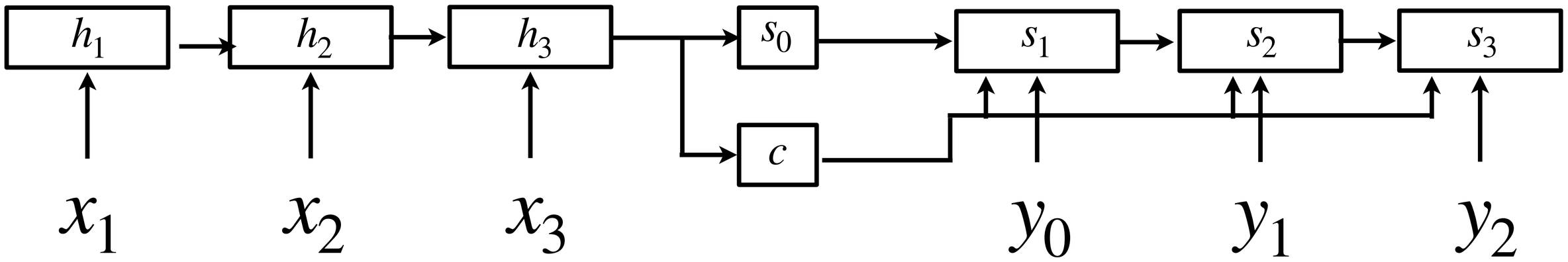


q, k, v 矩阵

其中，q (Query) 的含义一般的解释是用来和其他单词进行匹配，更准确地说是用来计算当前单词或字与其他的单词或字之间的关联或者关系；k (Key) 的含义则是被用来和 q 进行匹配，也可理解为单词或者字的关键信息。

RNN Revisited

$$s_t \leftarrow g_\phi(y_{t-1}, h_{t-1}, c)$$



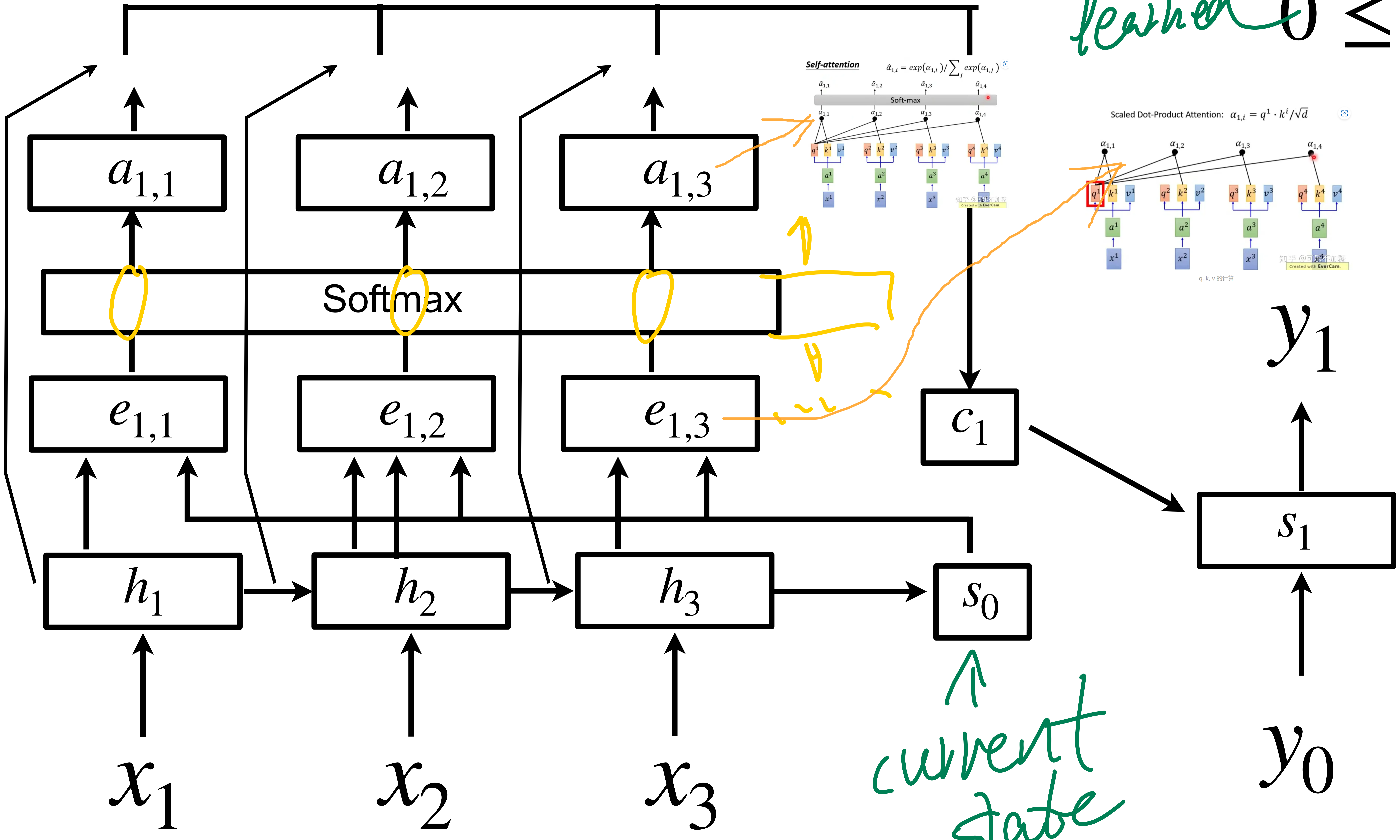
RNN with Attention

$$e_{t,i} \leftarrow g_{att}(s_{t-1}, h_i)$$

learned

$$0 \leq a_{t,i} \leq 1 \quad \sum_i a_{t,i} = 1$$

$$c_t = \sum_i a_{t,i} h_i$$



同时这里 e_{13} 也不用等 e_{11} 计算出来了

a_{13} 也不用等 a_{11}

只有softmax计算vector那步需要等所有 e.

Handling Speech

Transformer

Unsupervised Learning

There is no direct ground truth for the quantity of interest

Focus on **generative** models:

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

Generative Models

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

Dataset

Generated

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

Generative Models

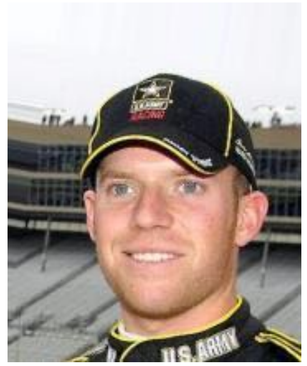
- **Assumption: the dataset are samples from an unknown distribution**
- **Goal: create a new sample from $p_{\text{data}}(\mathbf{x})$ that is not in the dataset**

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 $p_\theta(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})$?

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

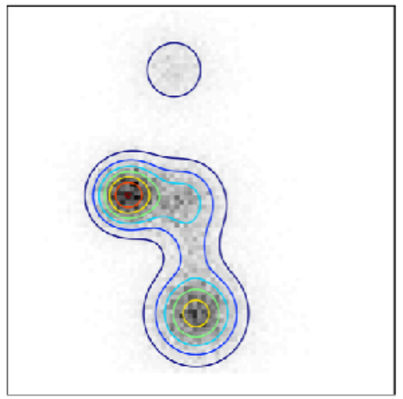
Autoregressive Models

2) Adversarial game

Generative Adversarial Networks (GANs)

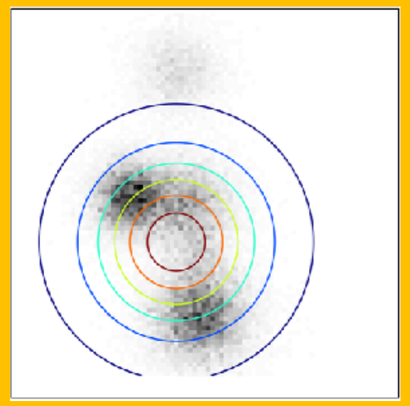
Generative Models

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



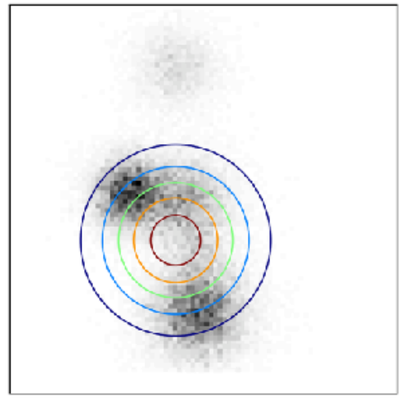
$p_{\text{data}}(x)$

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



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

2) Adversarial game



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

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

Sample? Evaluate?

Likelihood-Based Models: Two Goals

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

Feature Space

$$\min_{\theta,\phi} \sum_i \mathcal{L}(\theta, \phi, \mathbf{x}_i) = \min_{\theta,\phi} \sum_i \|D_\phi(E_\theta(\mathbf{x}_i) - \mathbf{x}_i\|^2$$

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

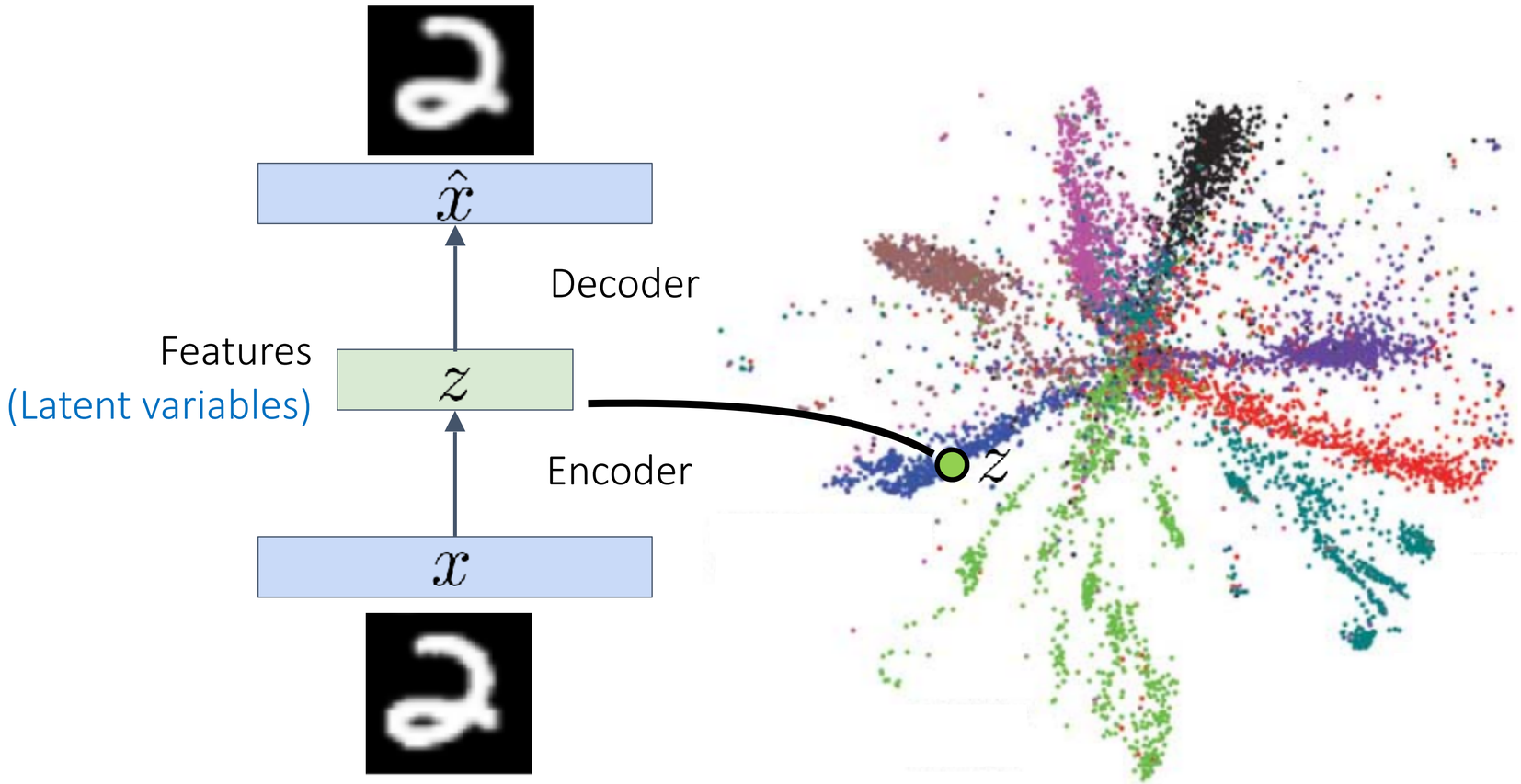
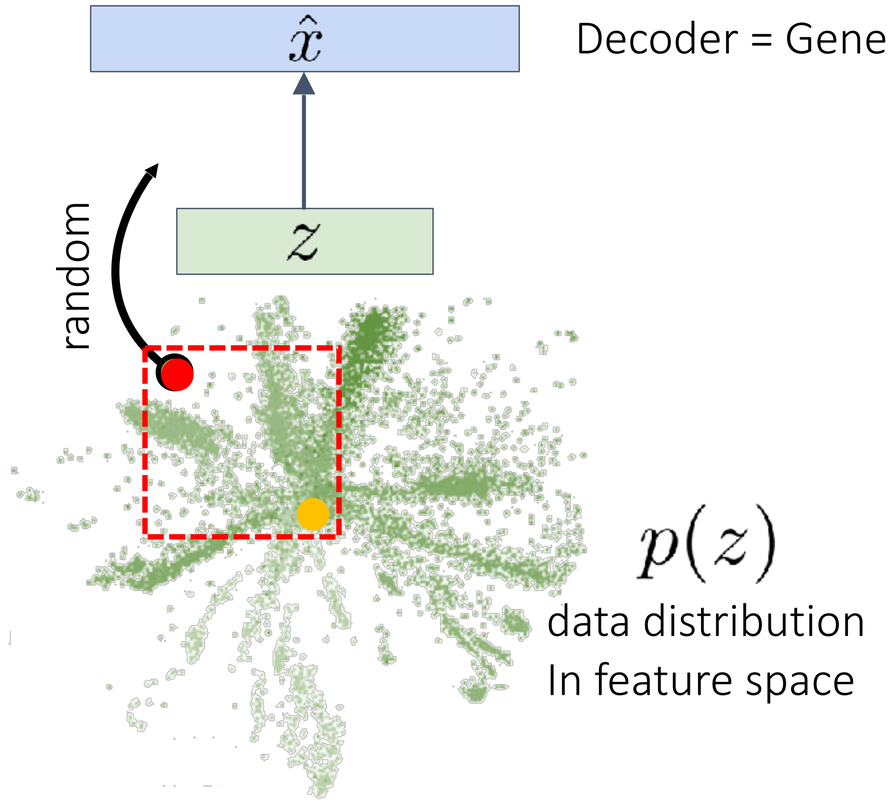


Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

Autoencoders as Generative Models?

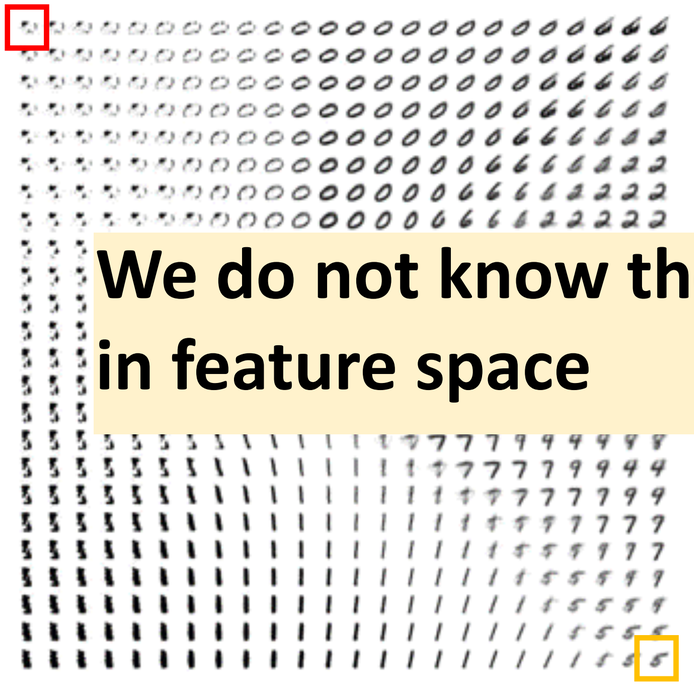
- **Is a trained decoder a generative model?**
- **Can we generate a new sample?**

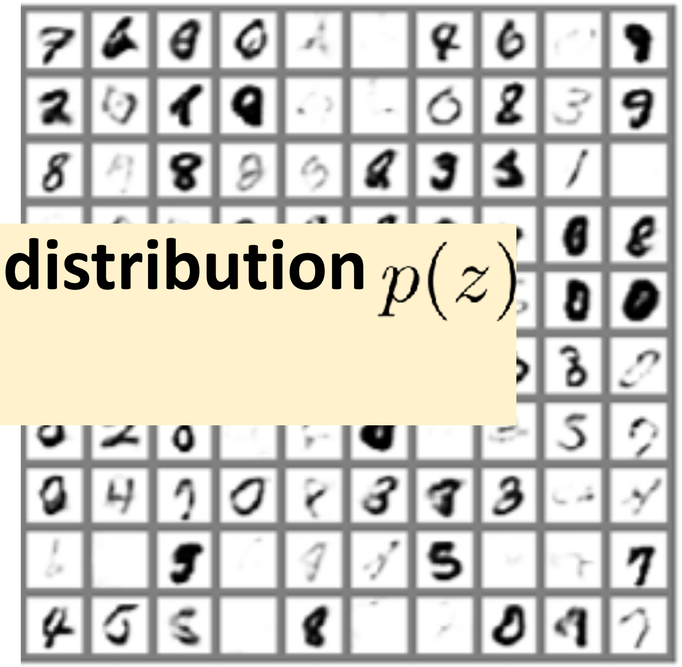


Decoder = Generator?

$p(z)$
data distribution
In feature space

$x \sim p_\theta$



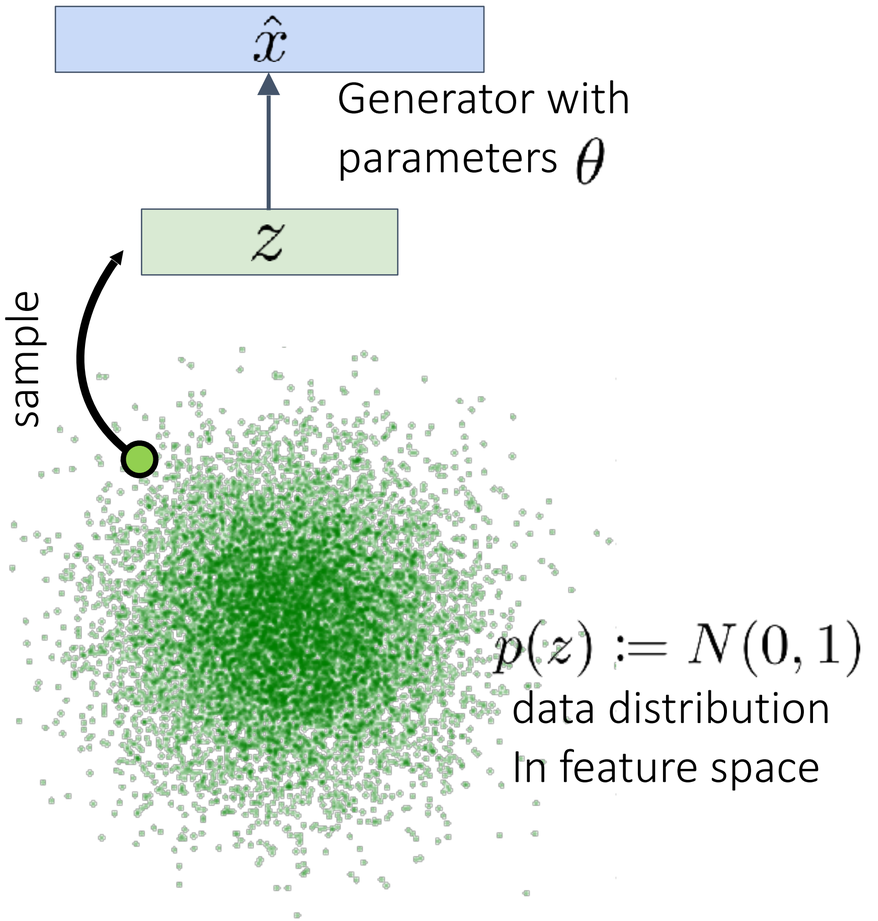


We do not know the distribution $p(z)$ in feature space

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

Latent Variable Model

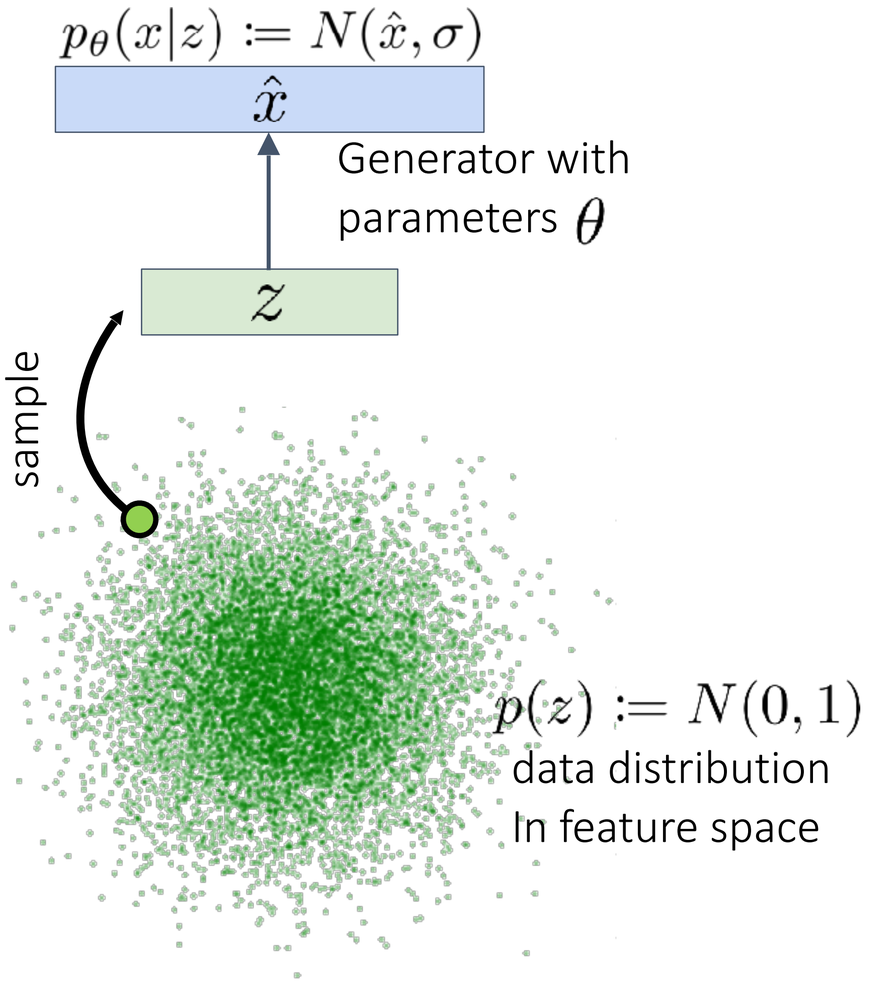
Sample?
Evaluate?



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

Latent Variable Model

Sample?
Evaluate?



(Negative log-likelihood)

- Train generator with NLL of data as loss

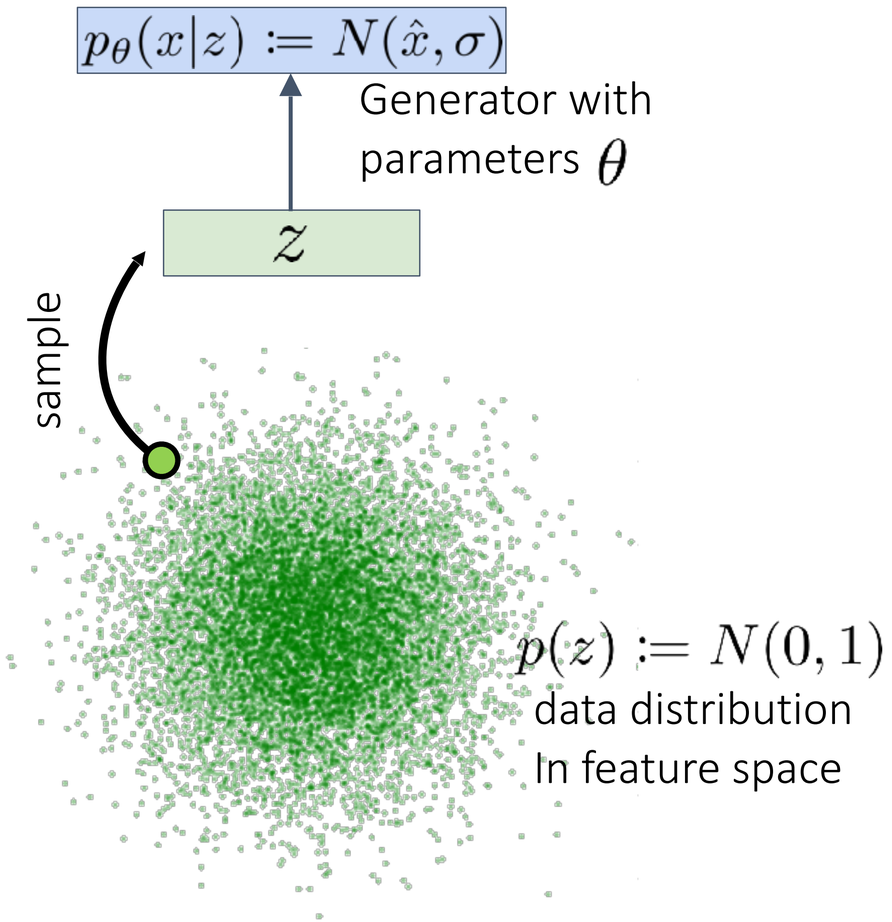
$$-\log \sum_{x_i \in \text{data}} p_\theta(x_i)$$

- 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 $p_\theta(x)$?

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

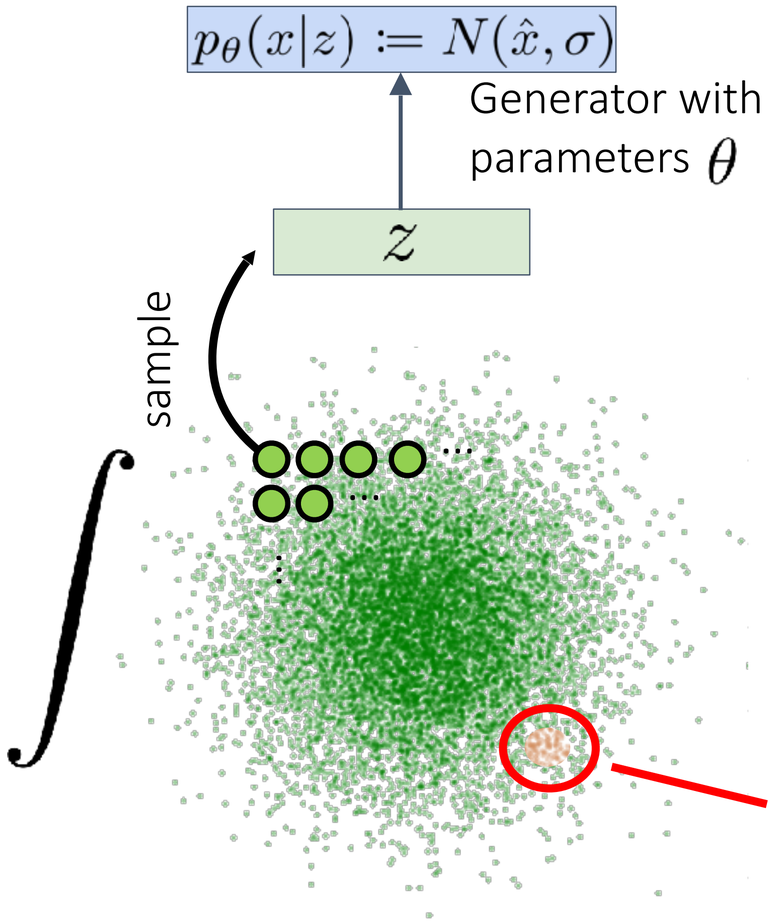
Sample?
Evaluate?

Variational Autoencoders (VAEs): The Encoder

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

Sample?
Evaluate?

Latent Variable Model: Monte-Carlo



(Negative log-likelihood)

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

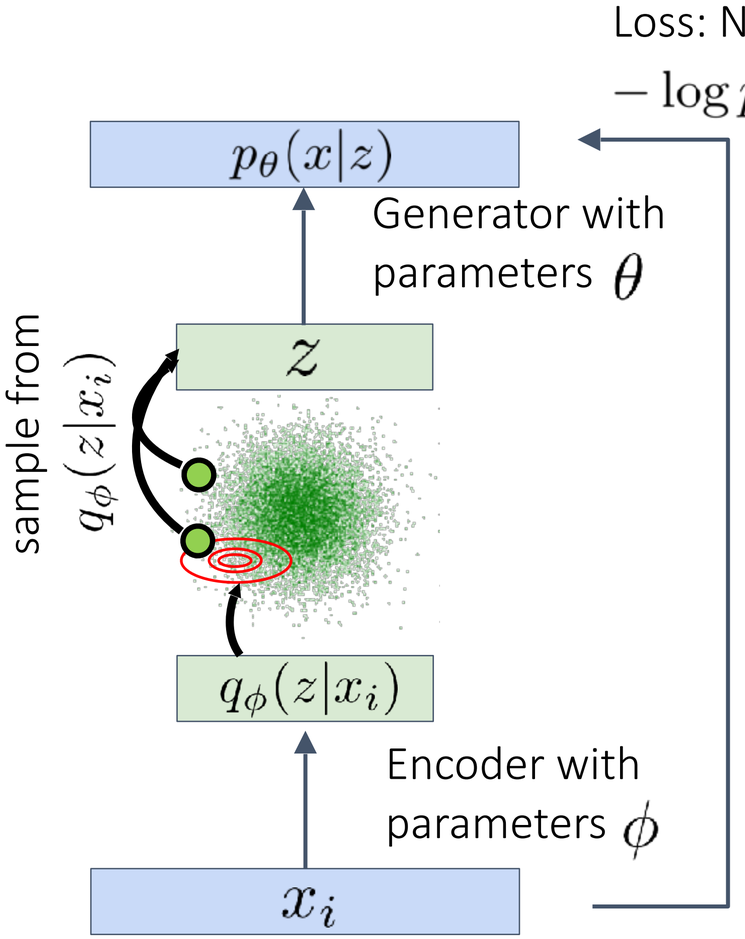
$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

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

$p(z|x_i)$

Variational Autoencoders (VAEs): Encoder

Sample?
Evaluate?

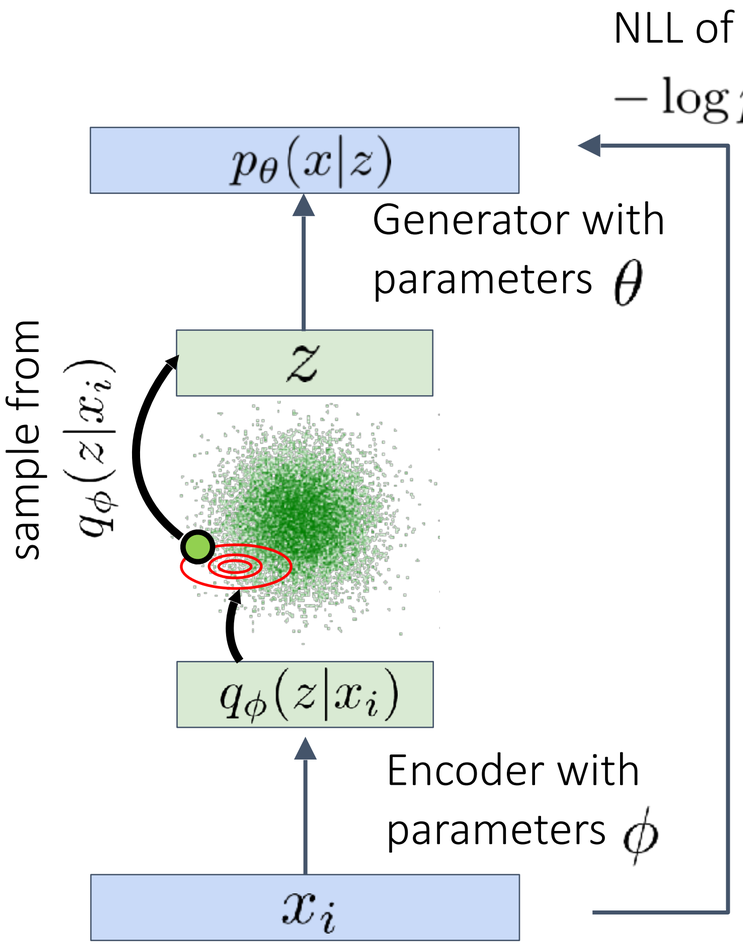


$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

- 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

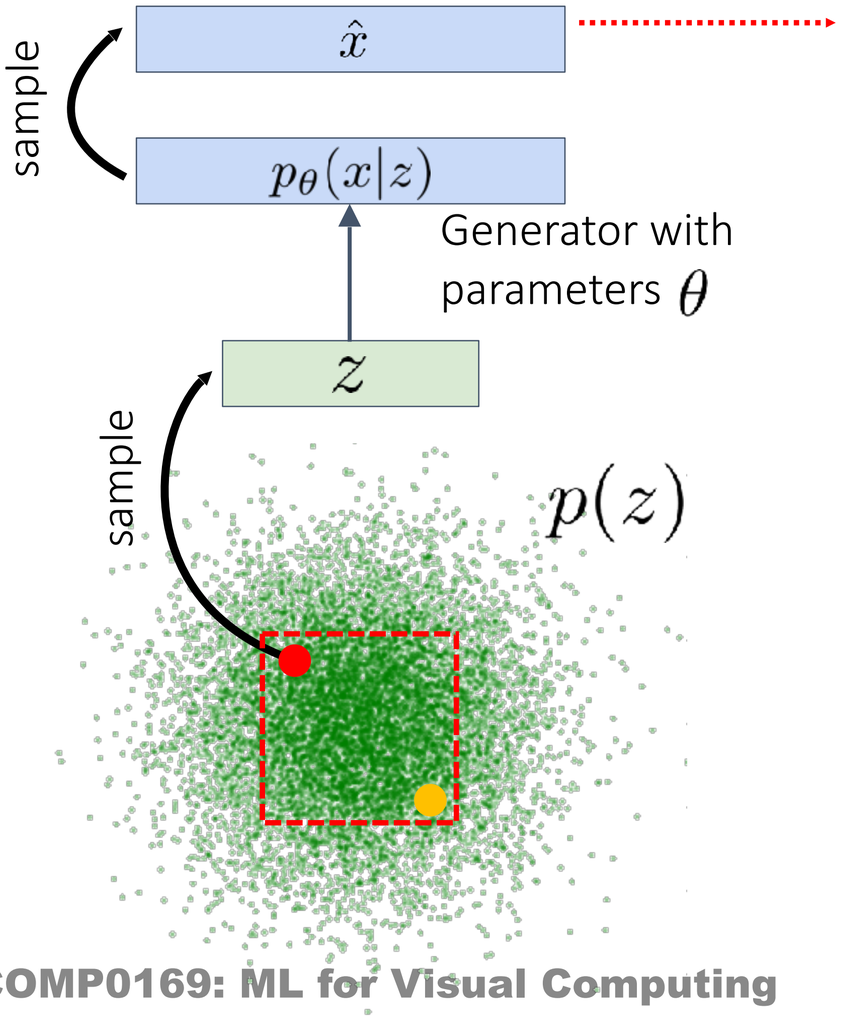
Variational Autoencoders (VAEs): Regularizer

Sample?
Evaluate?



- **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



VAE

Autoencoder

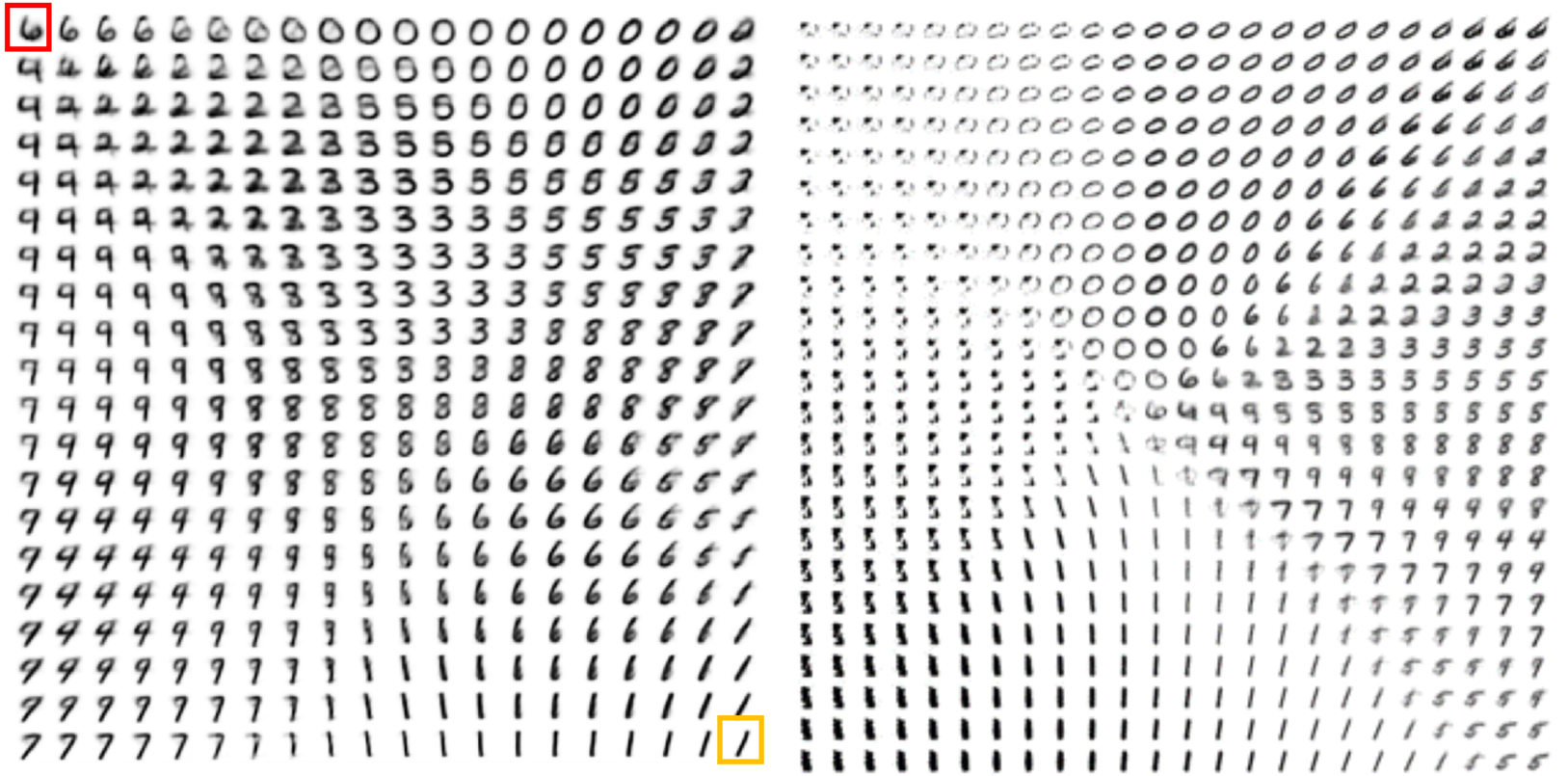
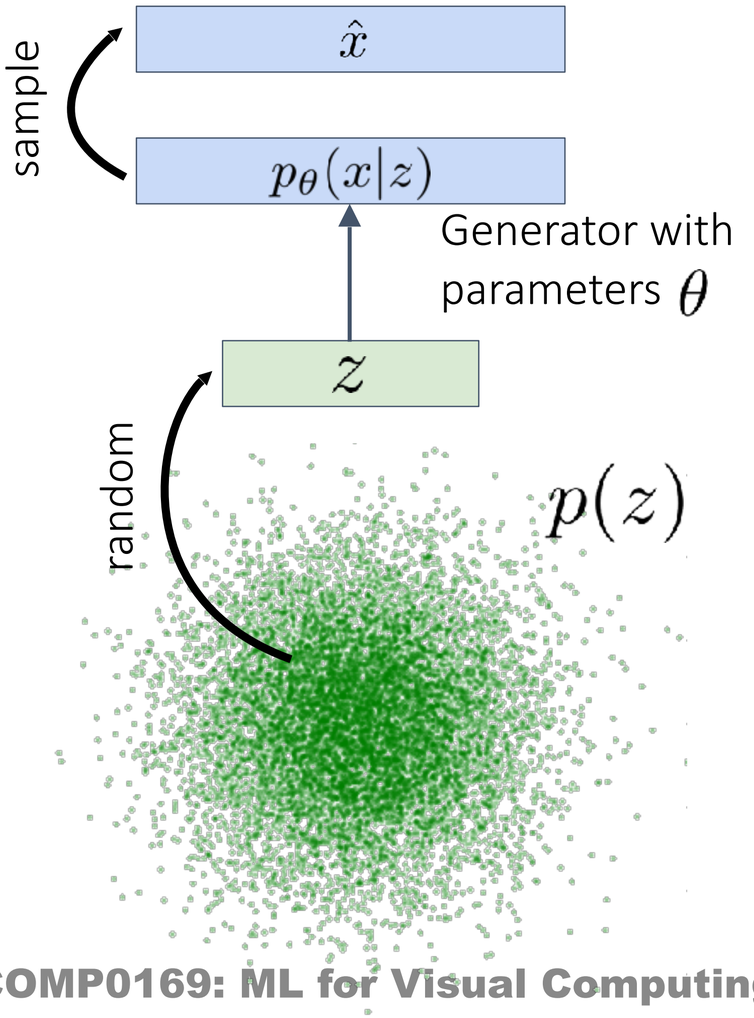


Image Credit: *Auto-Encoding Variational Bayes*, Kingma and Welling

Generating Data

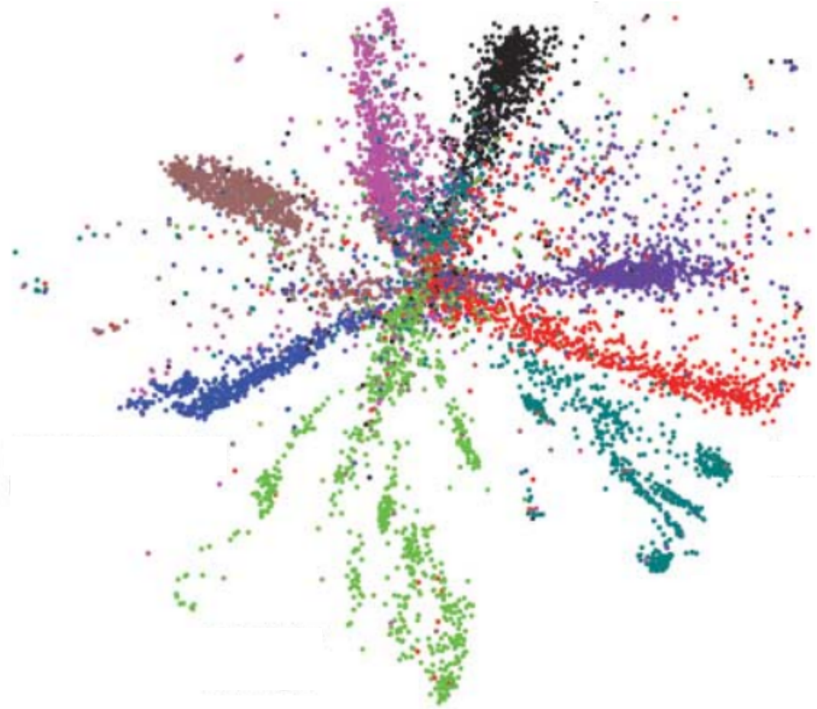


VAE

Autoencoder

Feature Space of Autoencoders vs. VAEs

Autoencoder



VAE

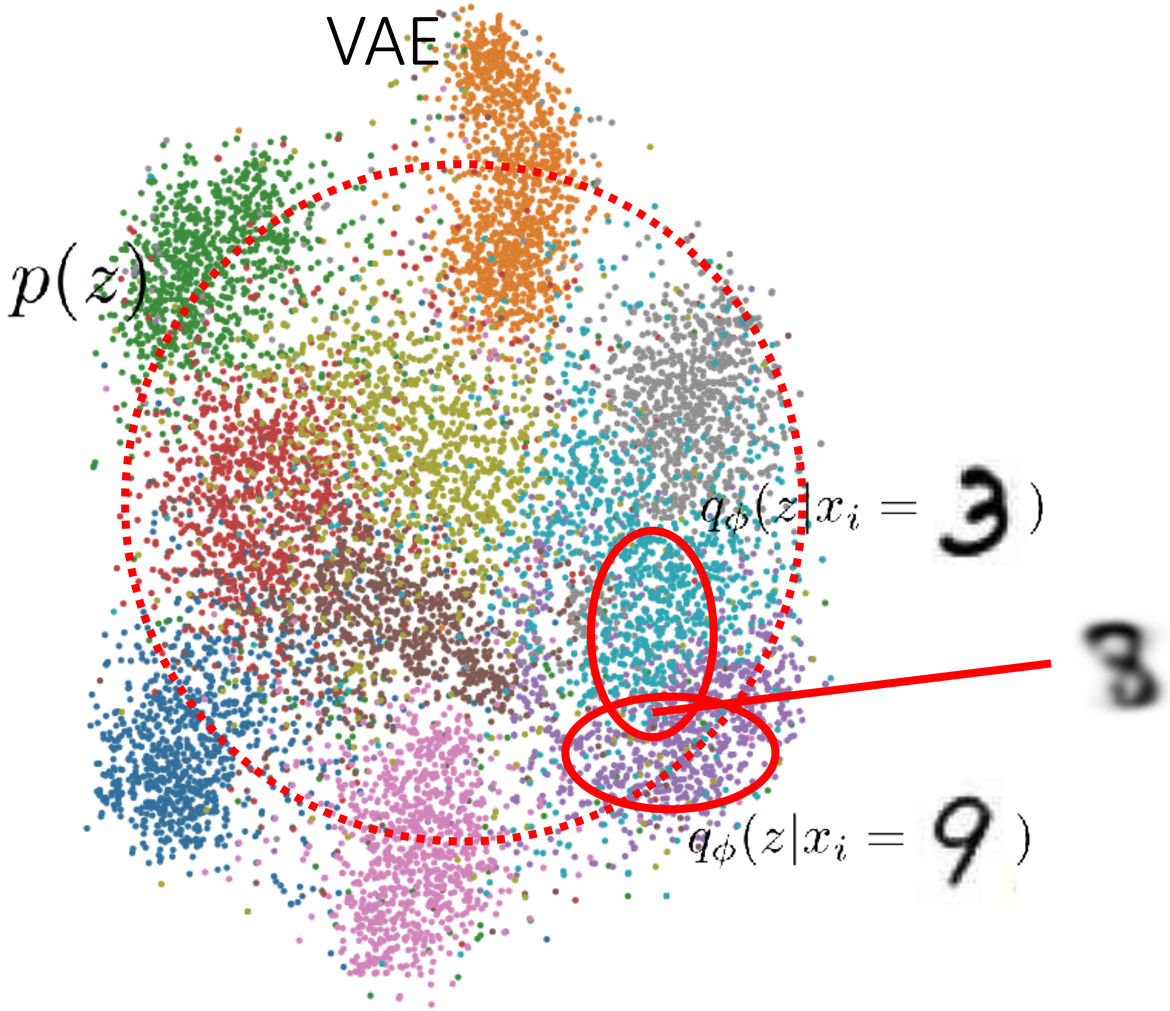


Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

Summary: Variational Autoencoders (VAEs)

Positives

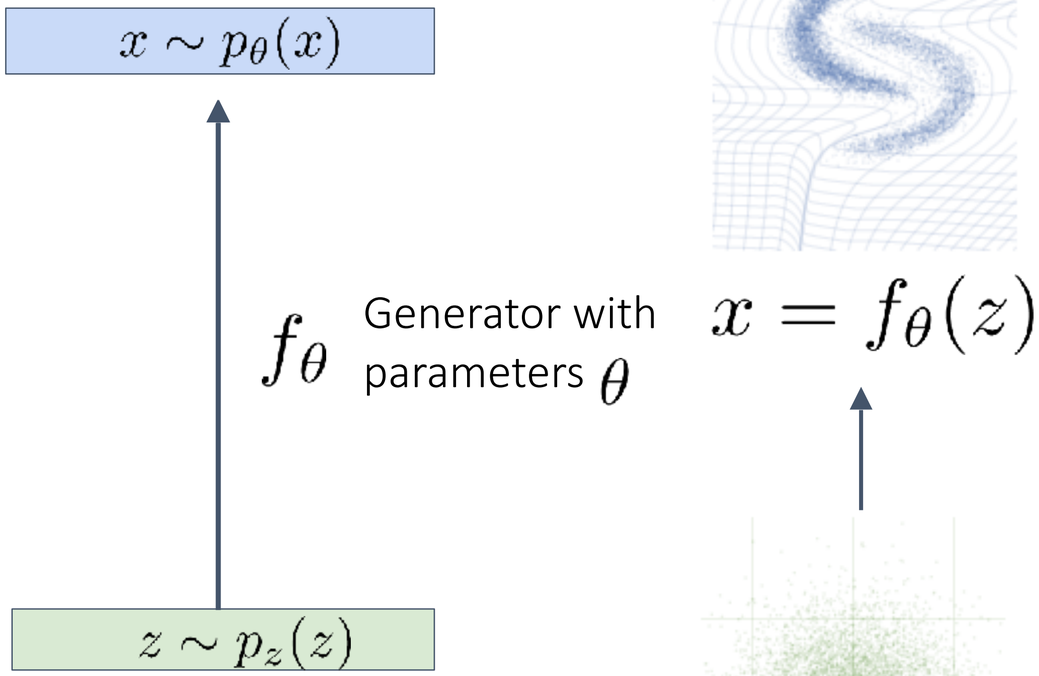
- Creates a feature space
- Relatively stable to train

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?



Variational Autoencoders (VAEs):

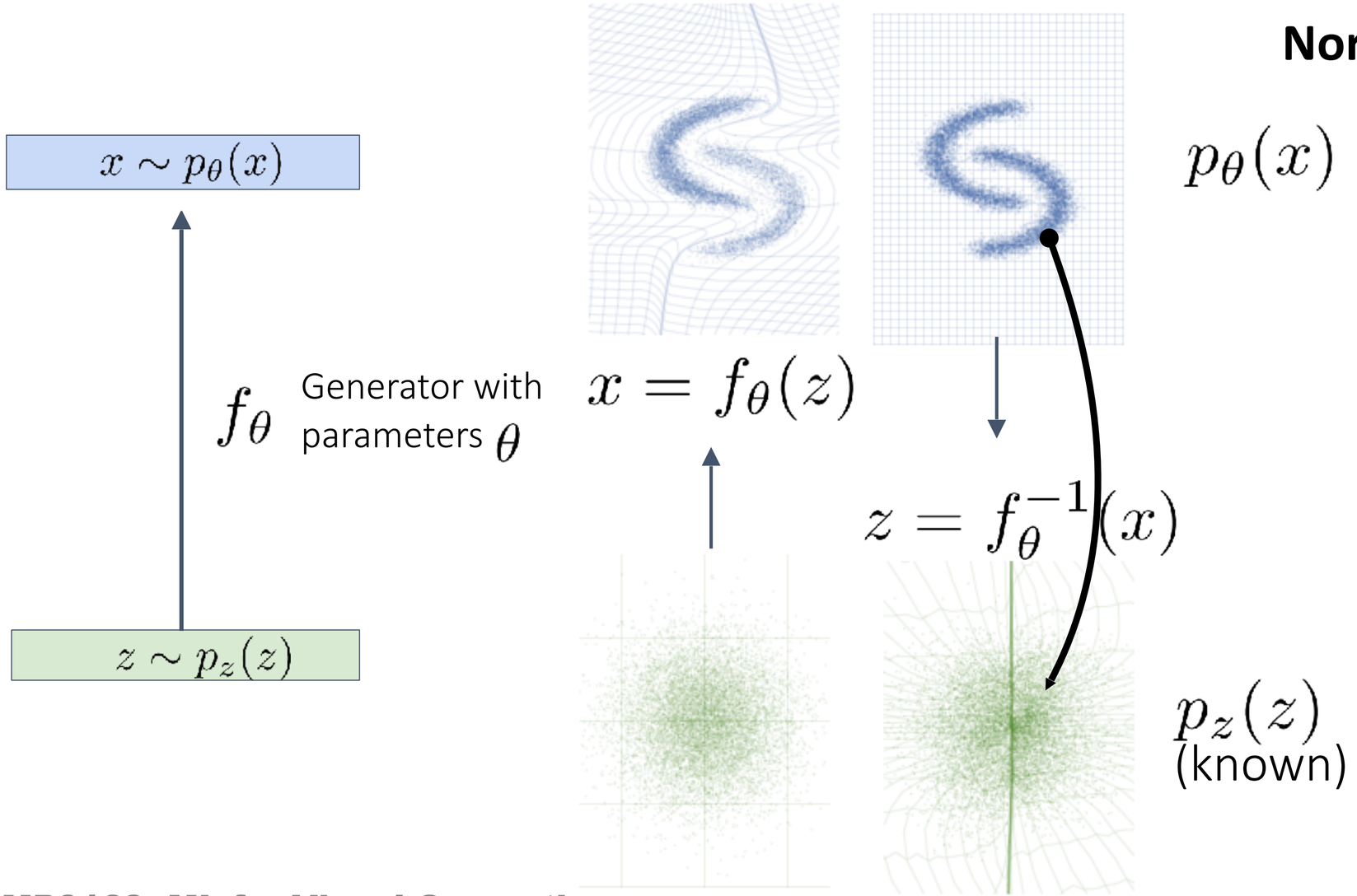
$$p_\theta(x) = \int ? p_\theta(x|z)\ p(z)\ dz$$

$p_z(z)$ (known)

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

Normalizing Flows

Sample?
Evaluate?



Normalizing Flows:

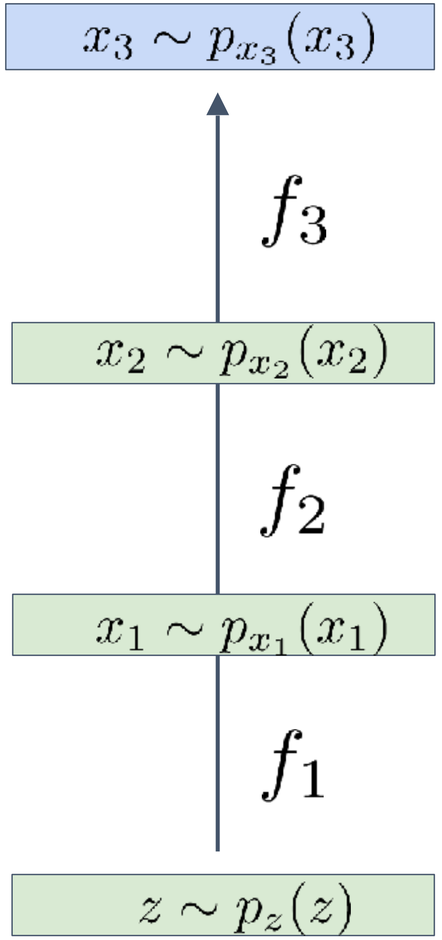
$$p_\theta(x) = p_z\left(f_\theta^{-1}(x)\right)$$

… times the local density change caused by f_θ

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

Normalizing Flows: Chaining

Sample?
Evaluate?



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

… 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**

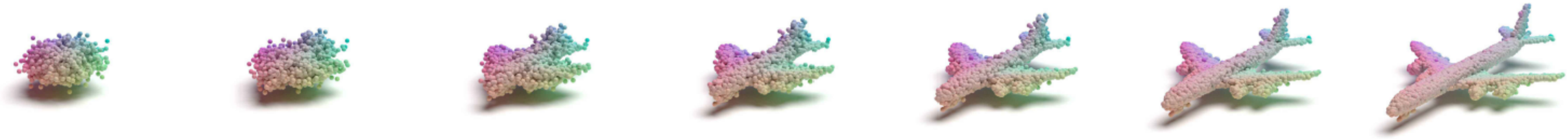
Training: 40 GPUs, 2 weeks

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

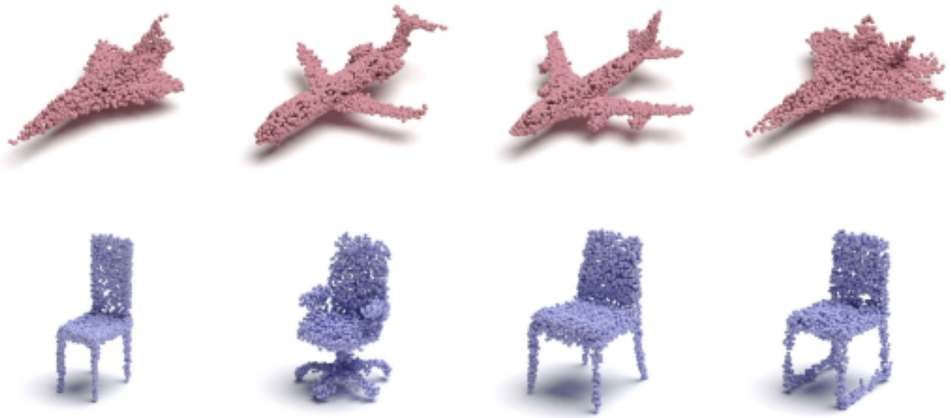
Example: PointFlow

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

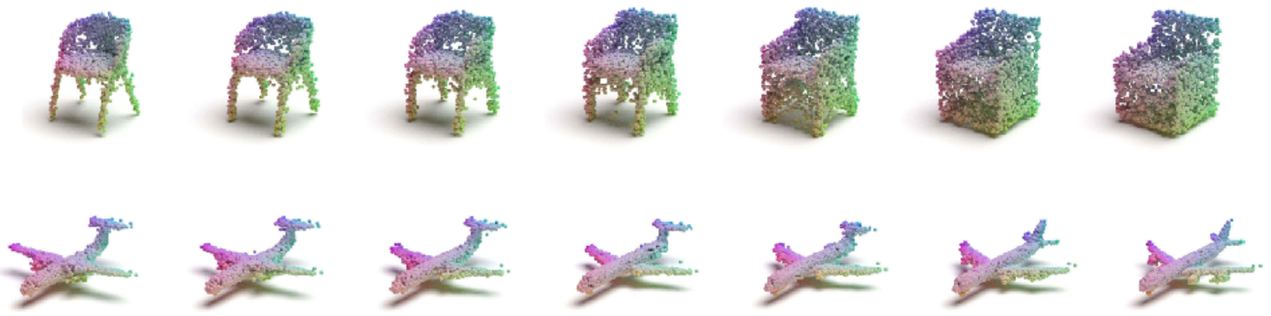


Image Credit: PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows, Yang et al.

Summary: Normalizing Flows

Positives

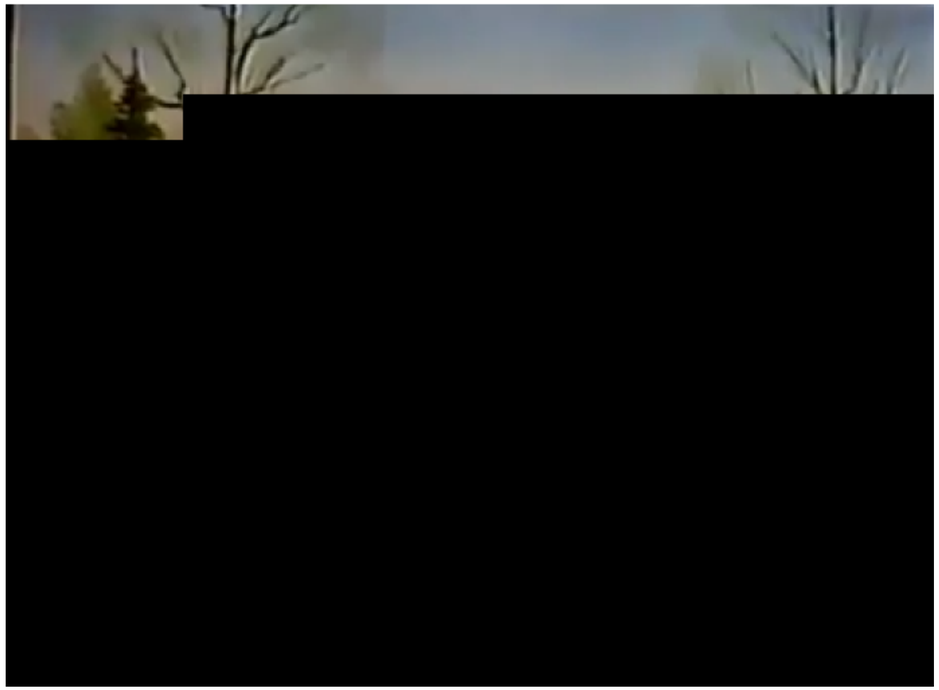
- **Creates a feature space**
- **Exact projection to feature space**
- **Exact likelihood evaluation**

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

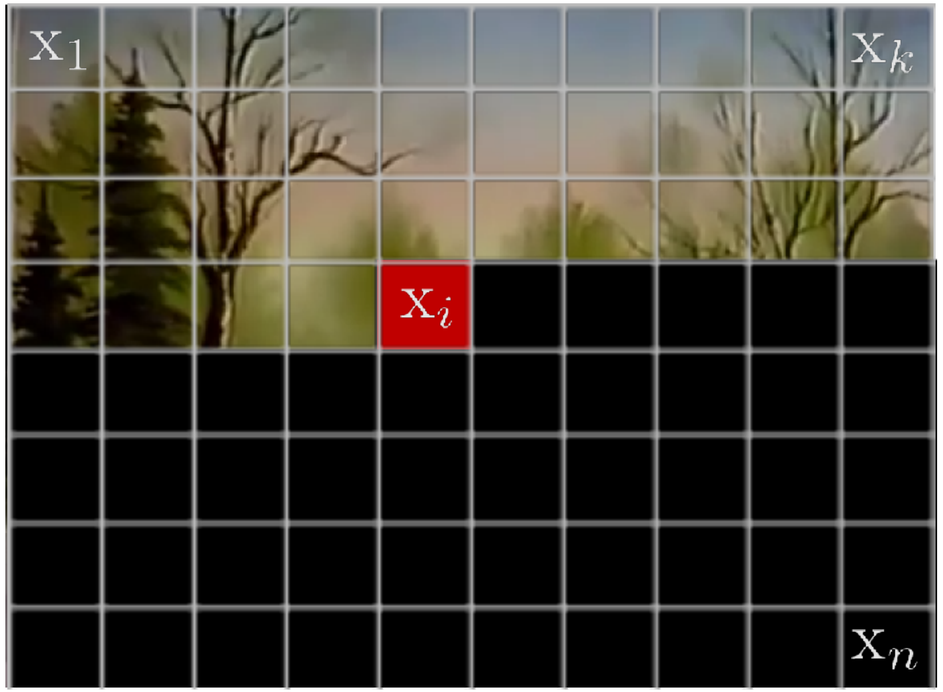
- Create output step-by-step
- Each step depends on the output of all previous steps



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**



$$x = [x_1, x_2, \ldots, x_n]$$

$$p_\theta(x) = p_\theta(x_1, x_2, \ldots, x_n)$$

Chain rule of probability:

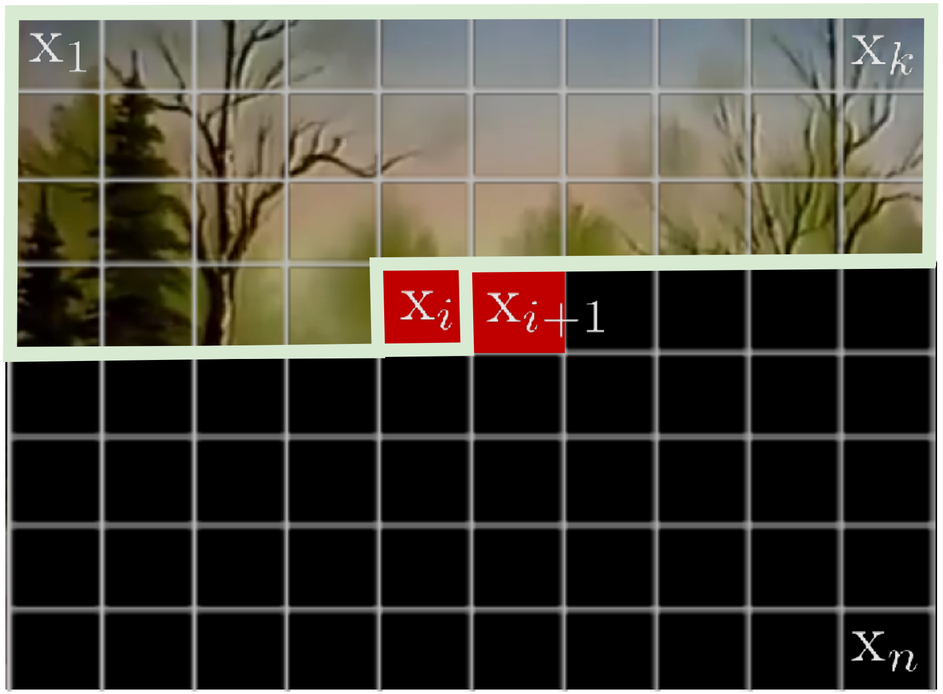
$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, x_2, \ldots, x_{i-1})$$

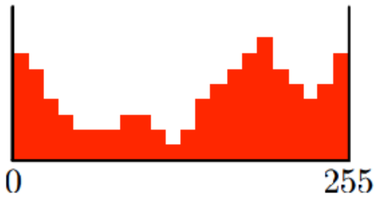
Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

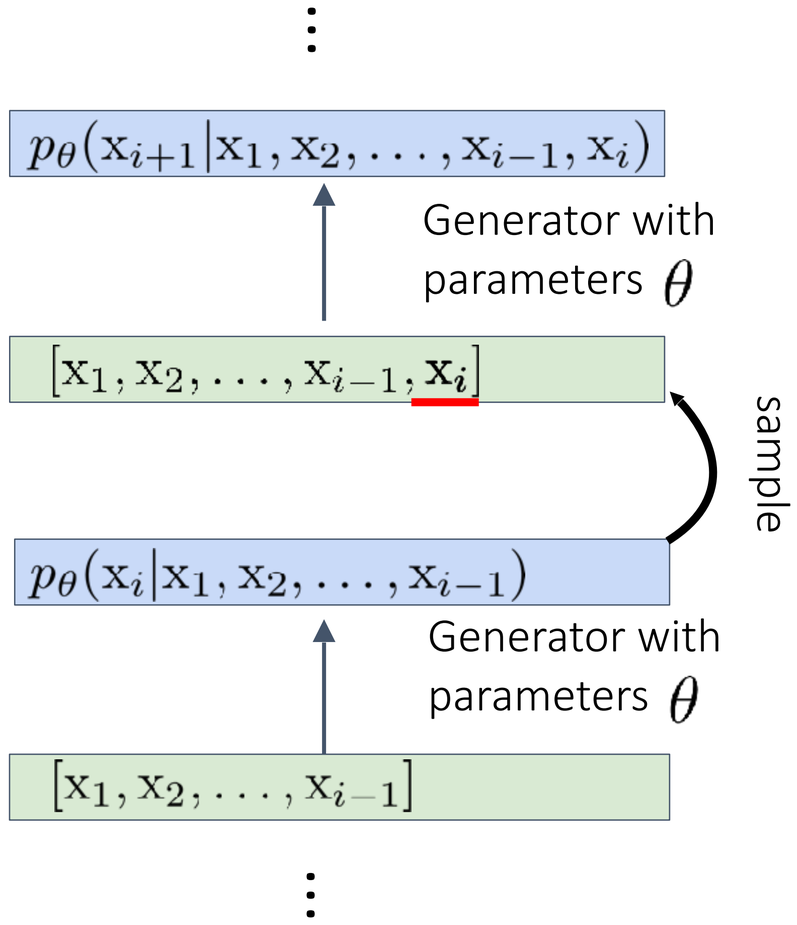
Autoregressive Models

Sample?
Evaluate?

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





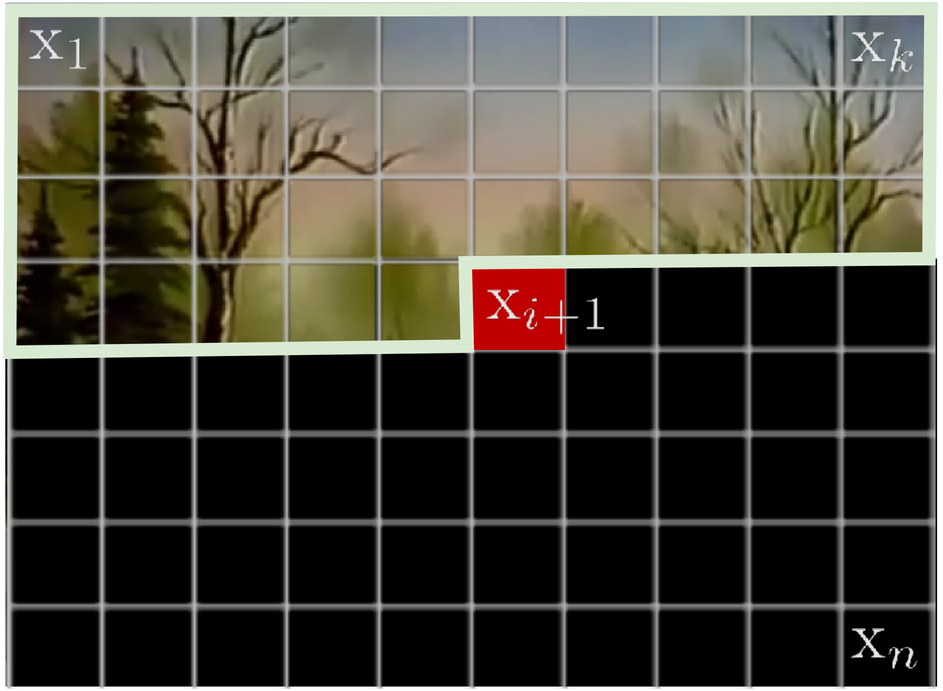


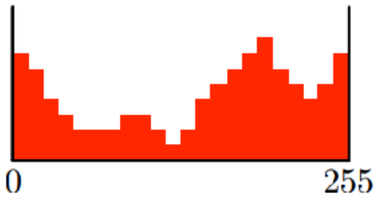
Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

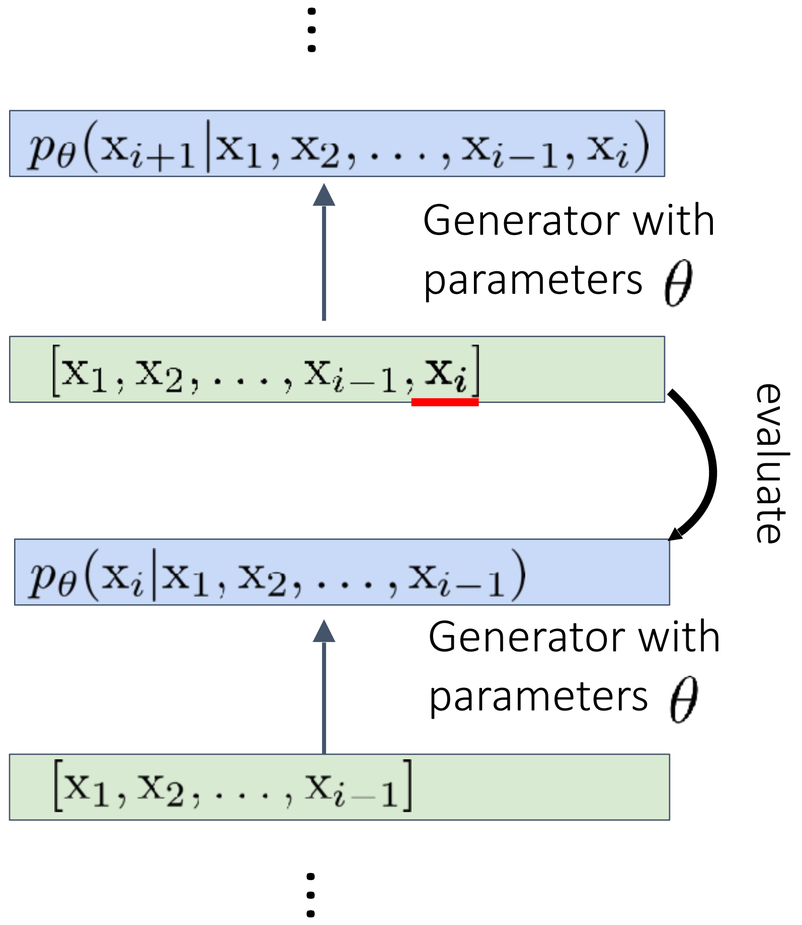
Autoregressive Models

Sample?
Evaluate?

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

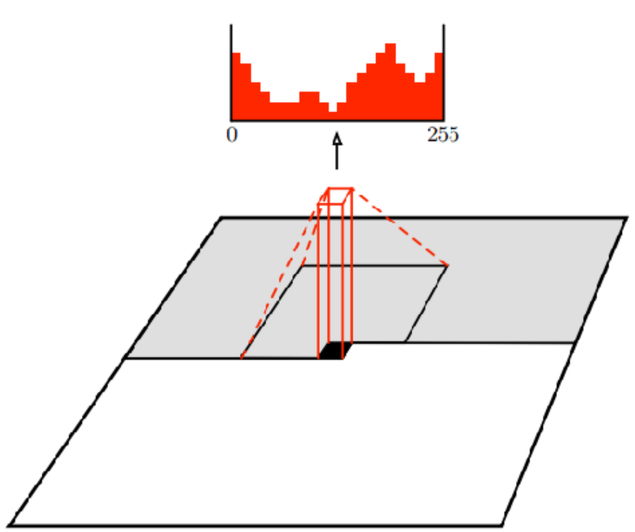


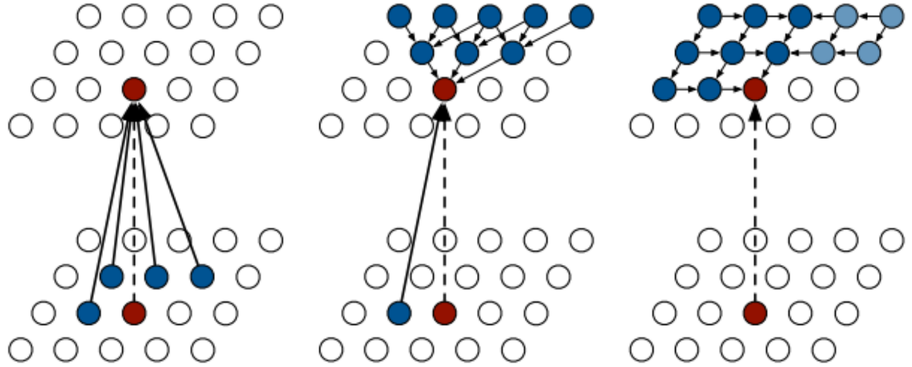




Example: PixelRNN and PixelCNN

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





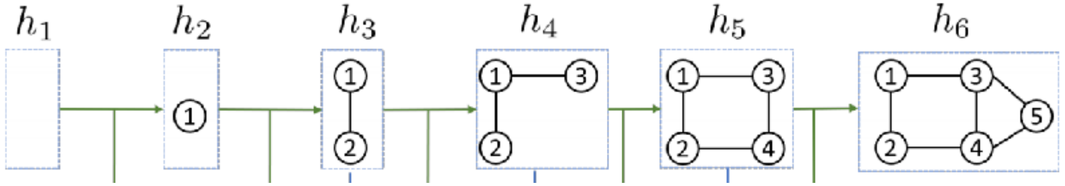
Sandbar

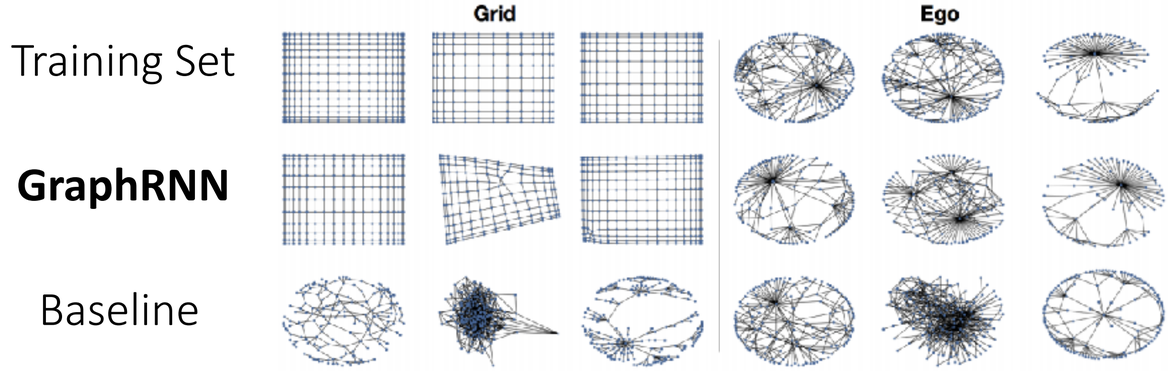
Lhasa Apso (dog)

Brown bear

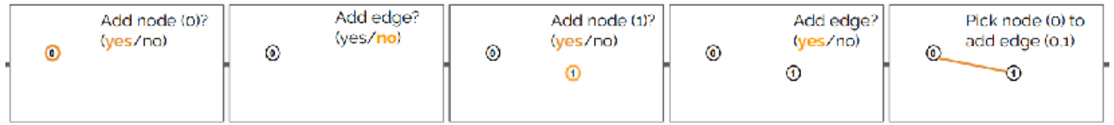
Example: Graph Generation

GraphRNN





Learning Deep Generative Models of Graphs



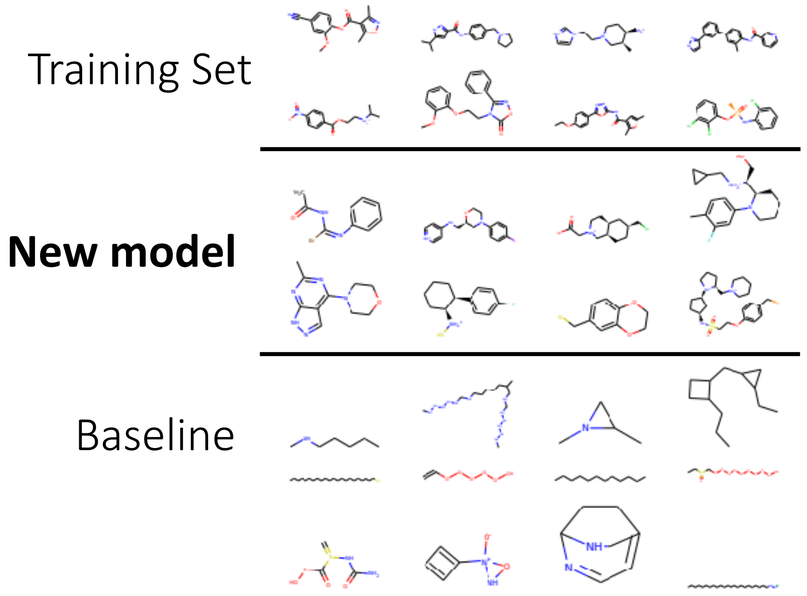


Image Credit: Conditional Image Generation with PixelCNN Decoders, Oord et al.
GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models, You et al.
Learning Deep Generative Models of Graphs , Li et al.

Summary: Autoregressive Models

Positives

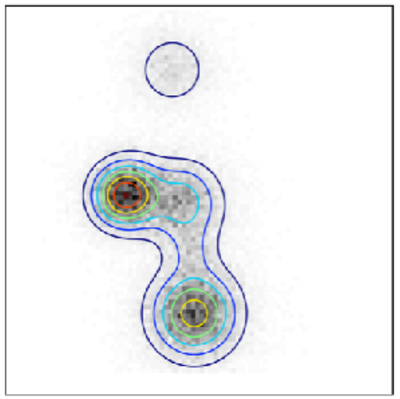
- Flexible output length
- Exact likelihood evaluation

Negatives

- No feature space
- Sequential generation (usually slow)

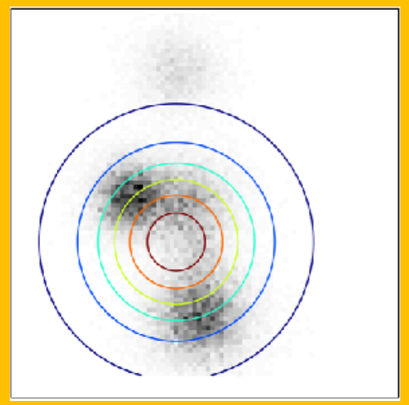
Generative Models

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



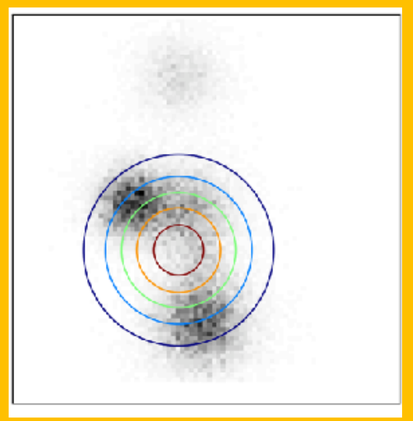
$p_{\text{data}}(x)$

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



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

2) Adversarial game



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

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

Generative Adversarial Networks (GANs)

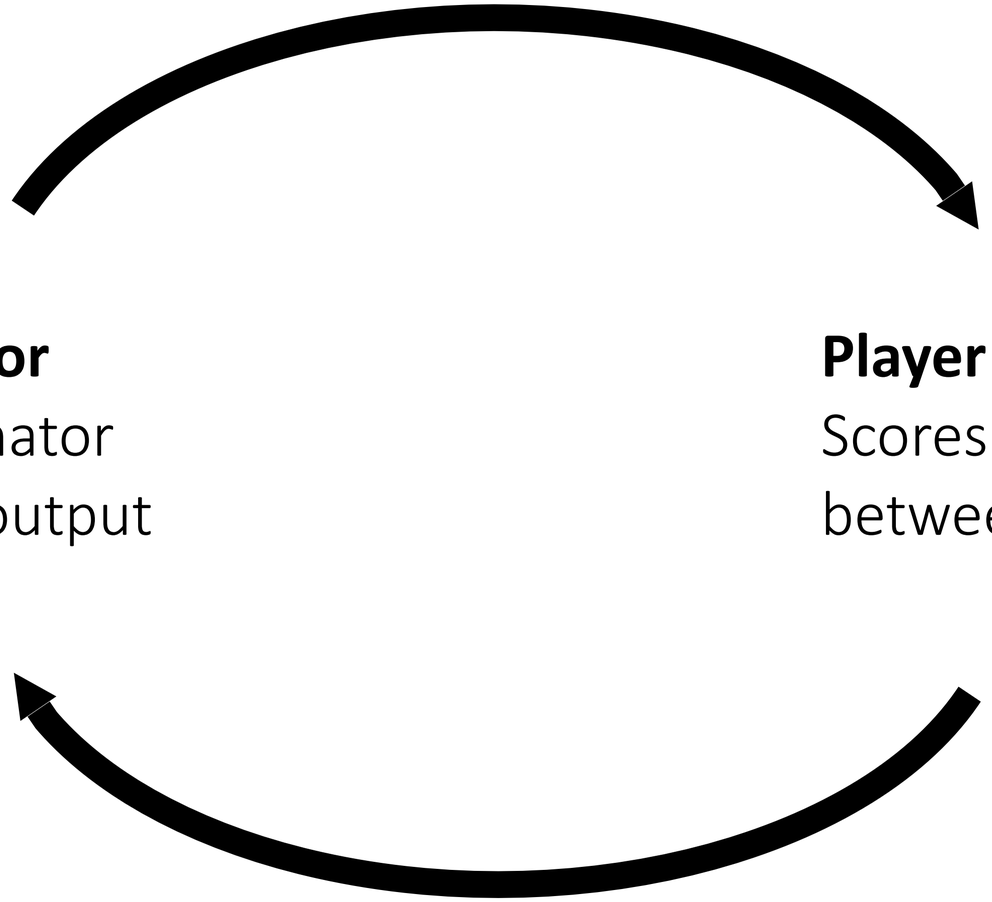
z → **Player 1: generator**
Scores if discriminator can't distinguish output from real image

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.

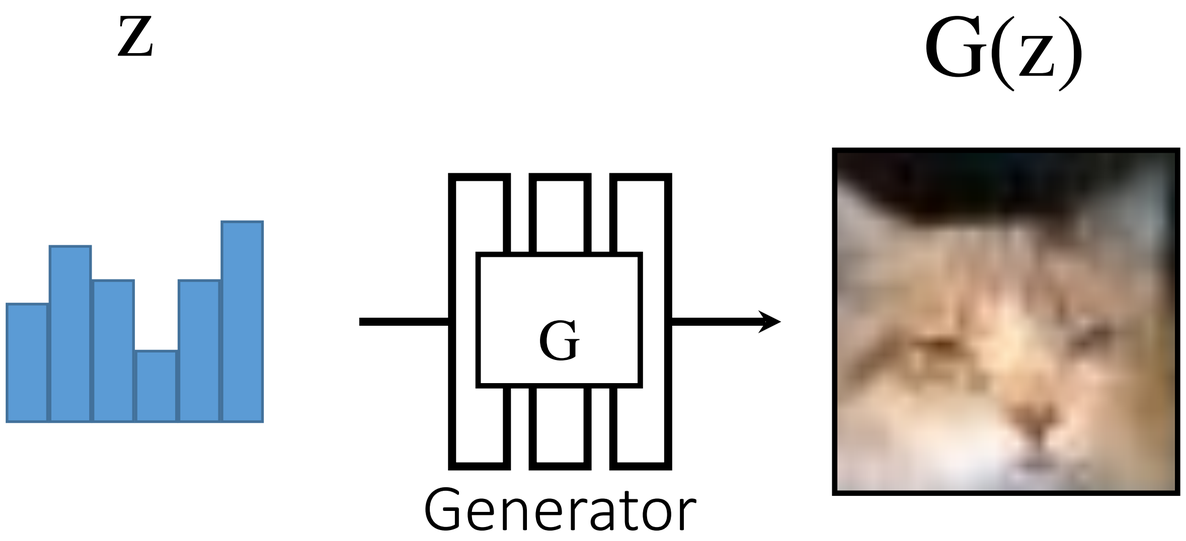
Generative Adversarial Networks (GANs)



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

Generator

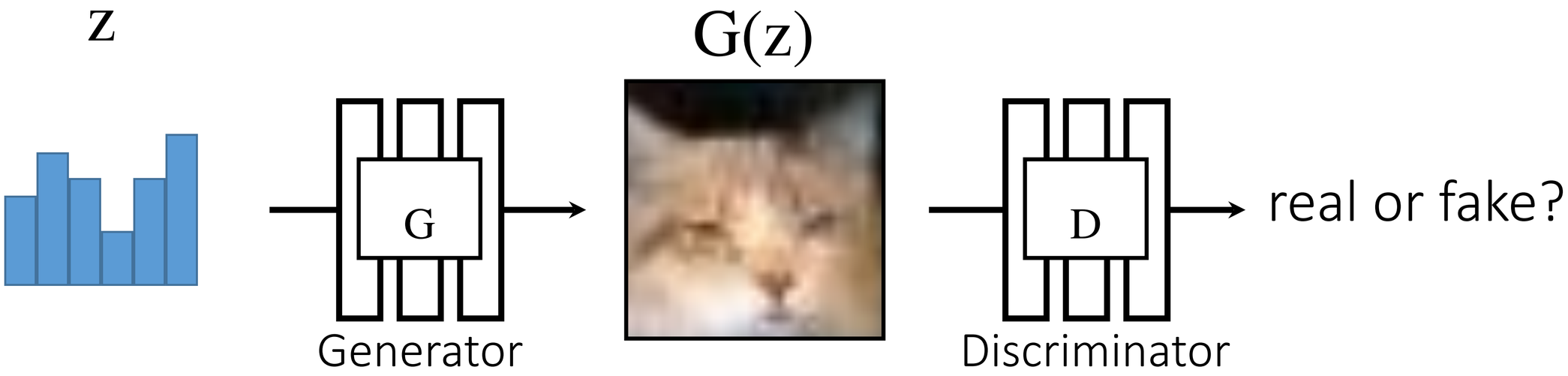


G: generate fake samples that can fool D

cat credit: aleju/cat-generator

slide credit: Phillip Isola & Jun-Yan Zhu

Generator + Discriminator



G: generate fake samples that can fool D

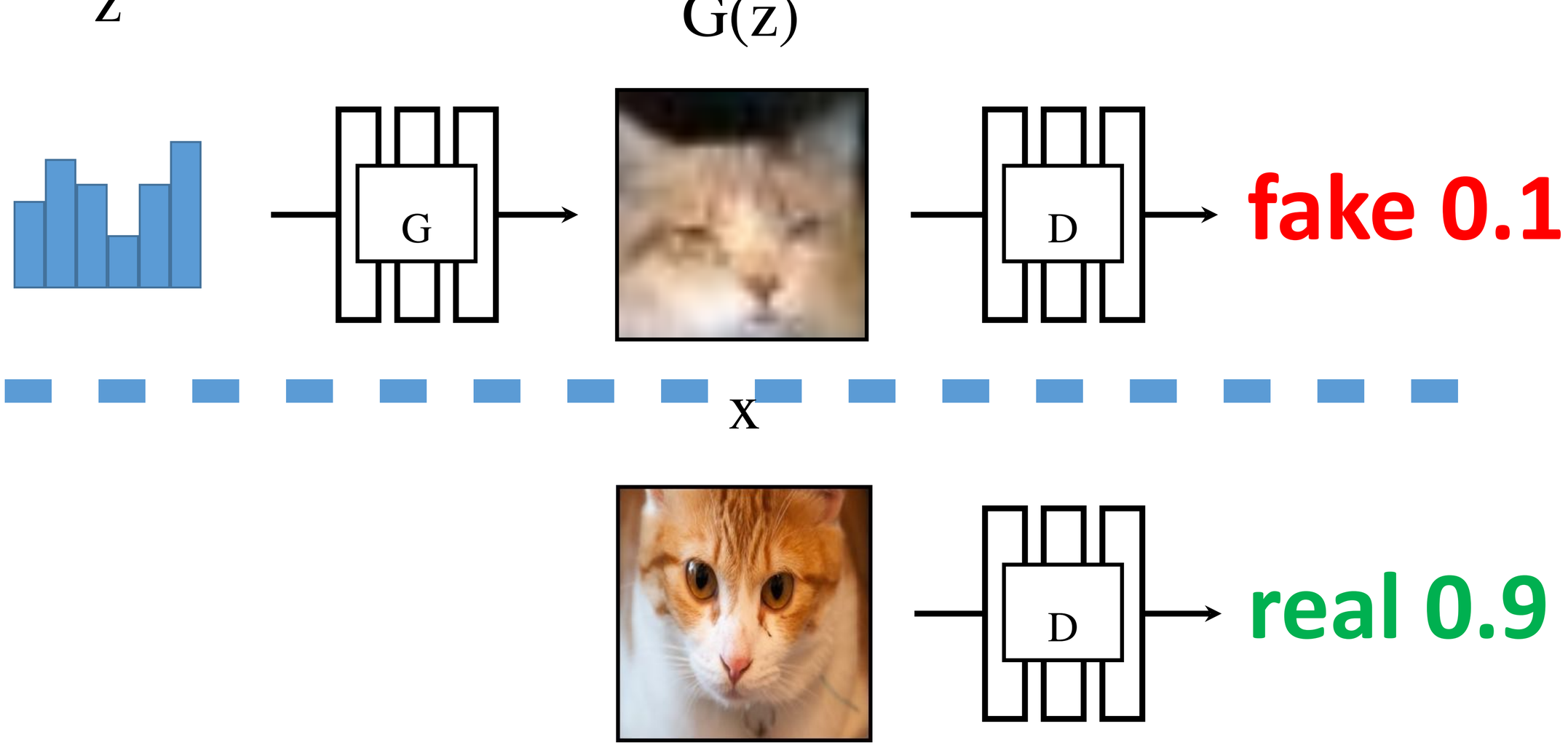
D: classify fake samples vs. real images

slide credit: Phillip Isola & Jun-Yan Zhu

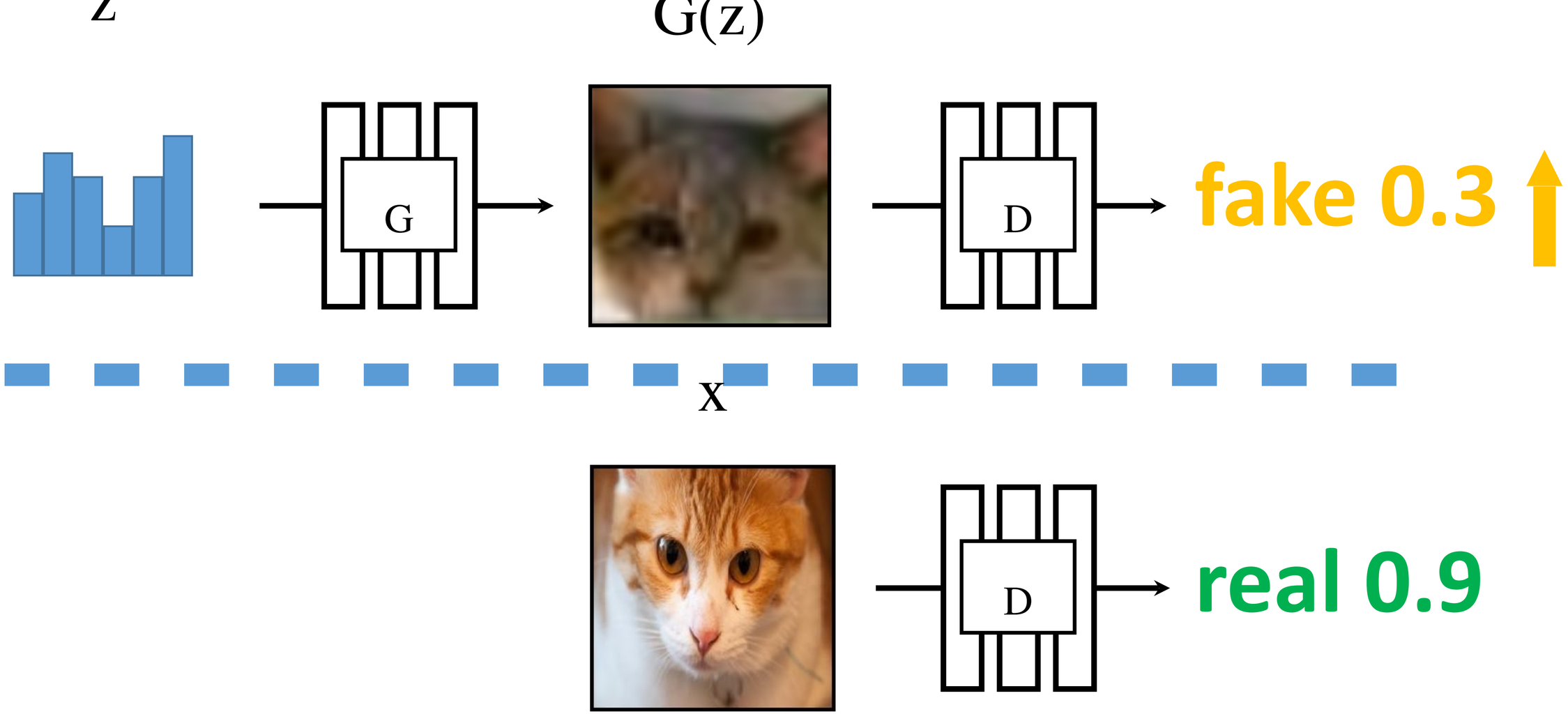
$$\min_{G}\max_{D}\mathbb{E}_{z,x}\Big[\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\Big]$$

slide credit: Phillip Isola & Jun-Yan Zhu

$$\min_G \max_D \mathbb{E}_{z,x}\left[\log \underbrace{D(G(z))}_{\text{fake}}\right]$$



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



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

Update G

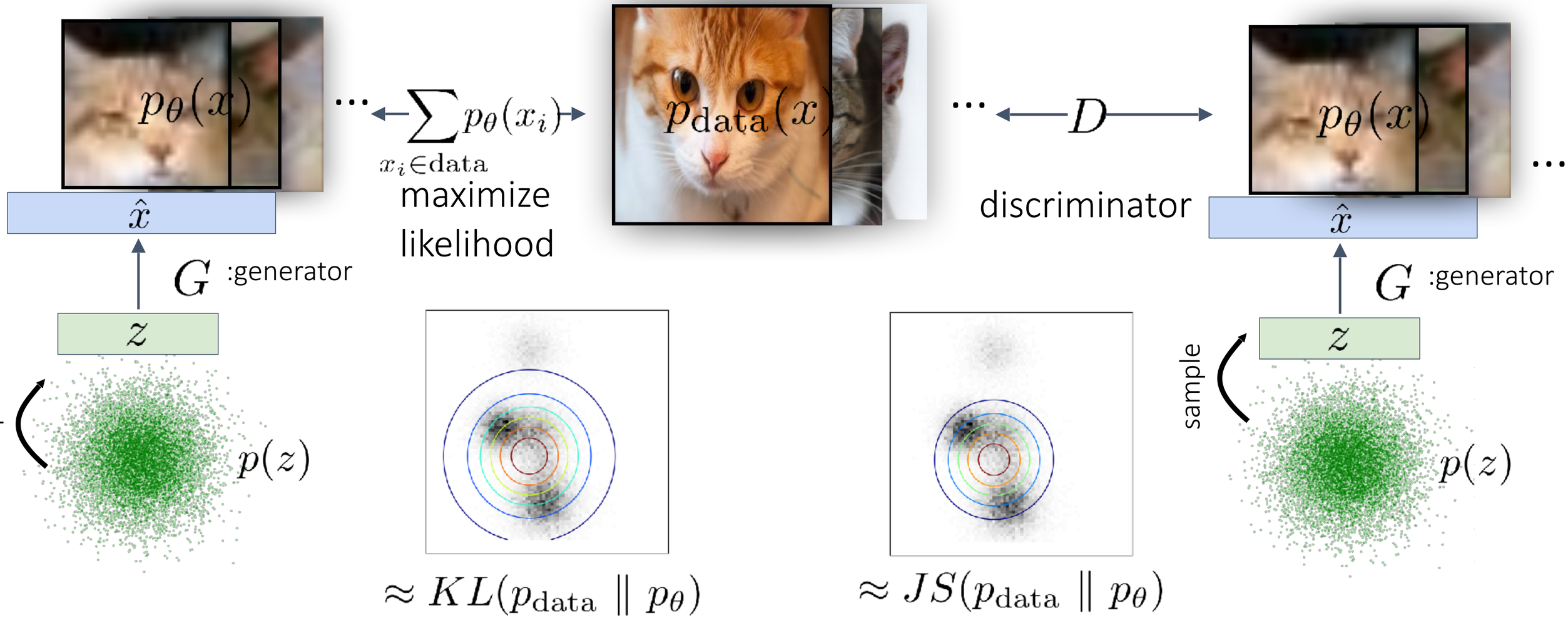
Generated Distributions of GANs vs ML

Sample?

Evaluate?

VAEs or Norm. Flows

GANs



StyleGAN

Additional Tricks:

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

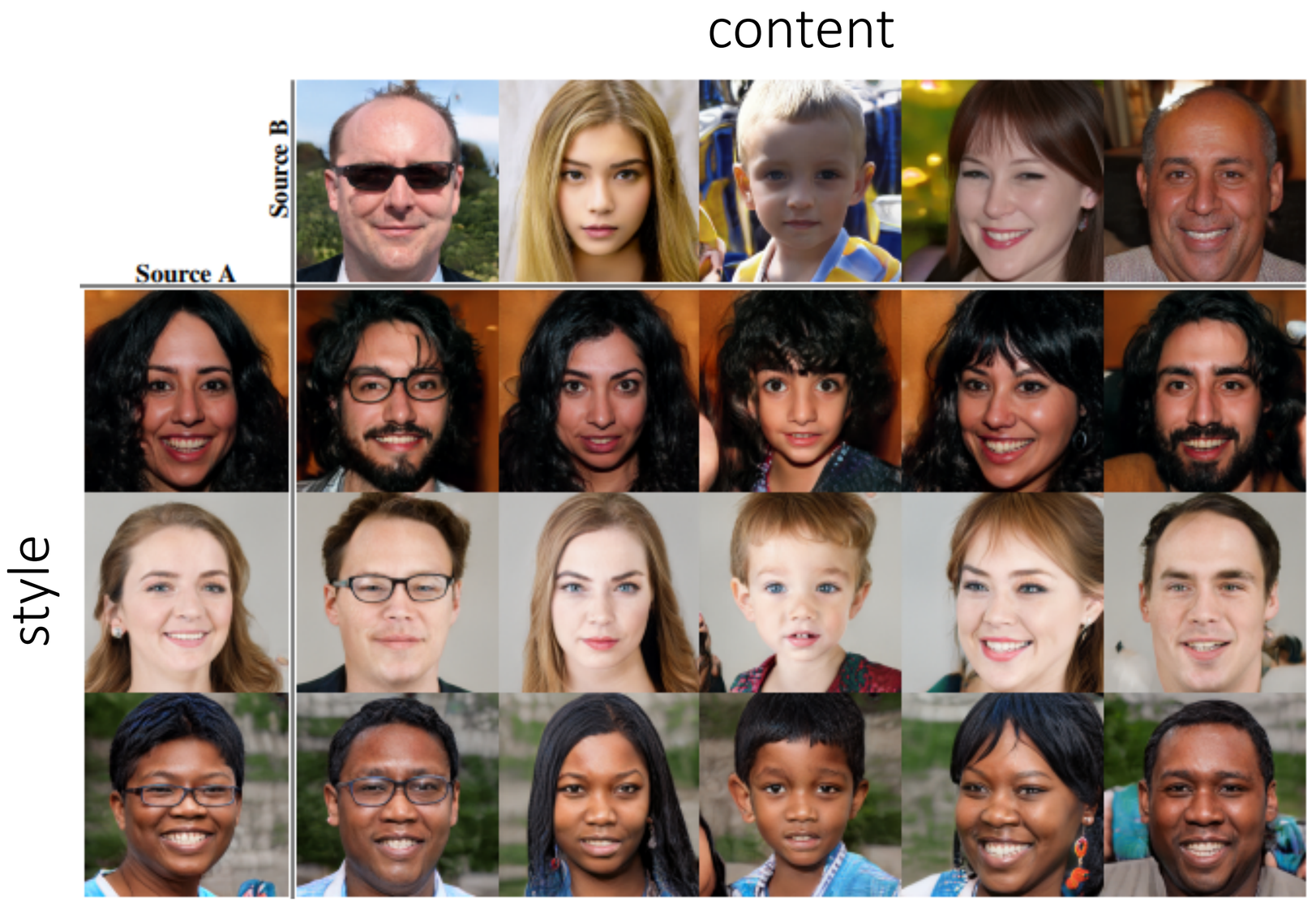


Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks,* Karras et al.

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

Conditional GAN: Pix2Pix

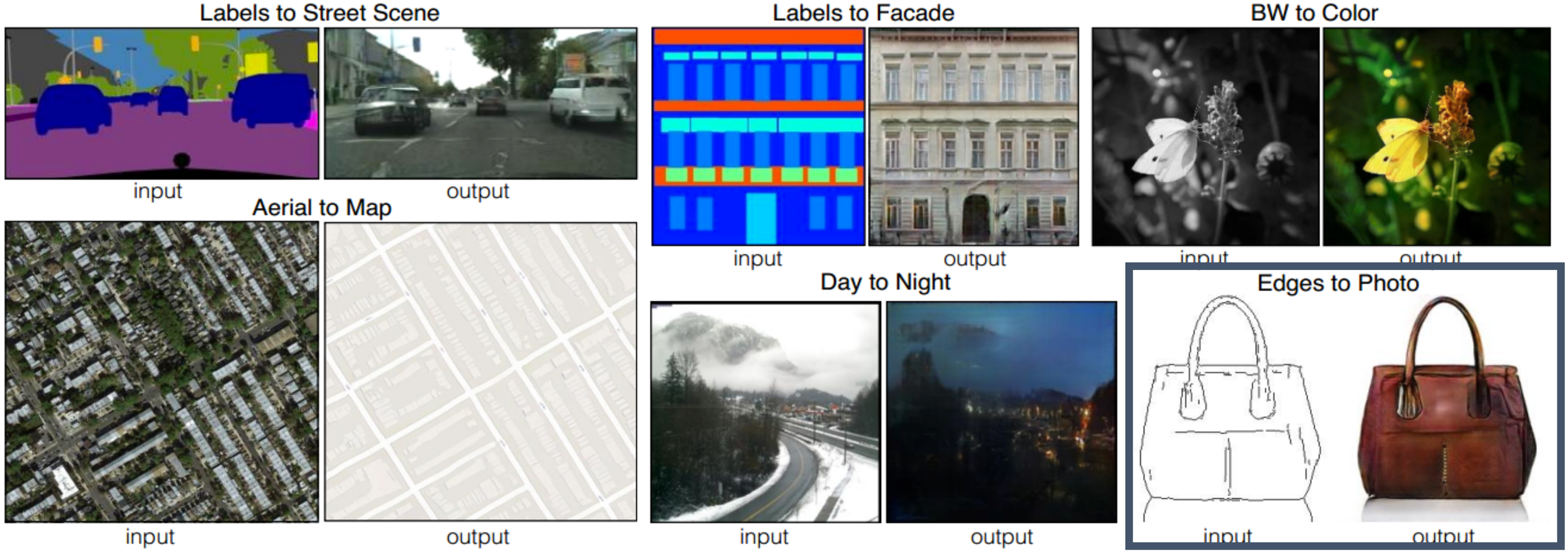
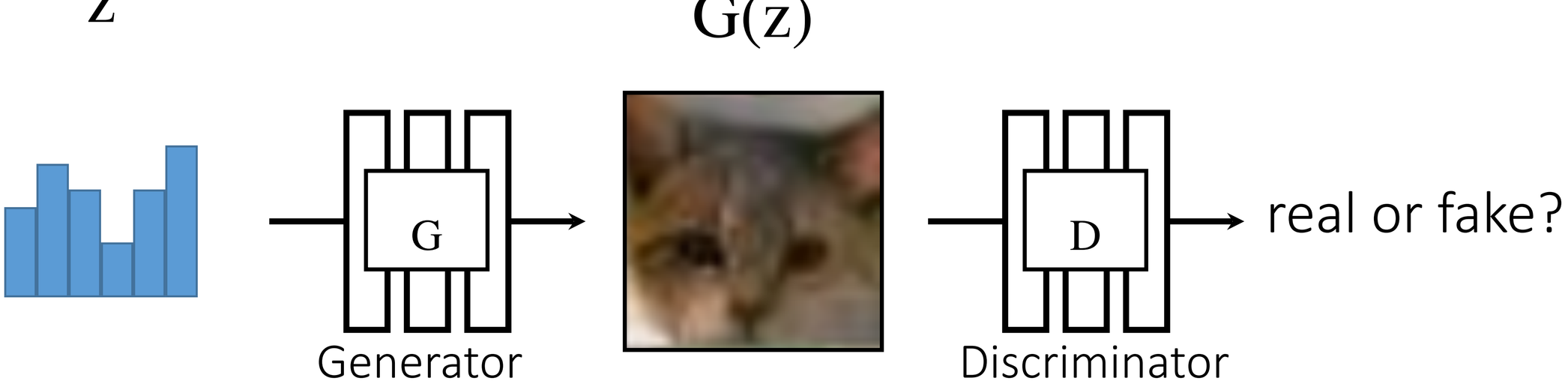


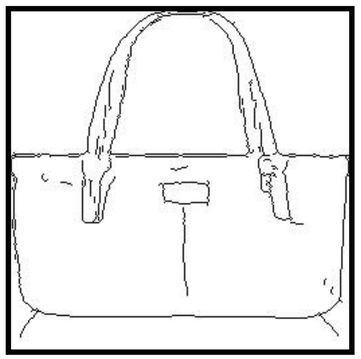
Image-to-image Translation with Conditional Adversarial Nets
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017



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

x

G(x)



G

D

real or fake?

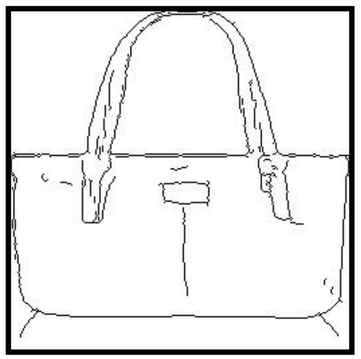
Generator

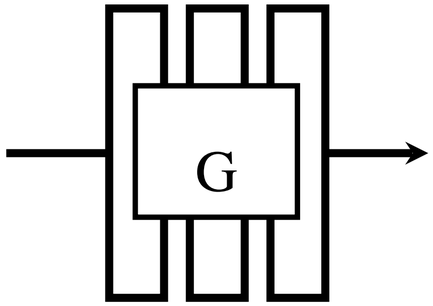
Discriminator

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

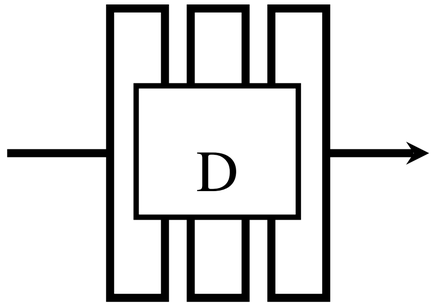
x

G(x)





Generator



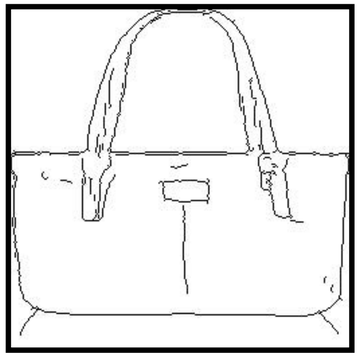
Discriminator

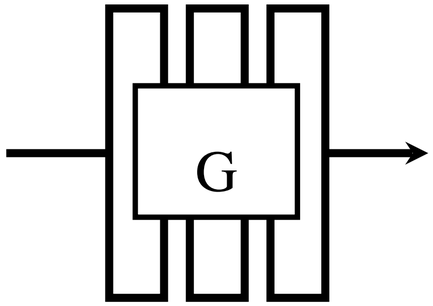
Real!

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

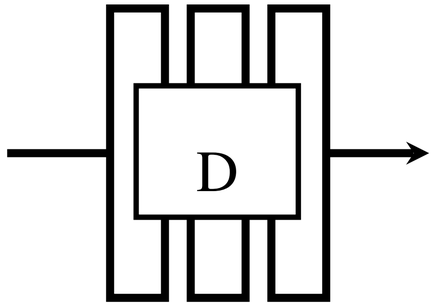
x

G(x)





Generator



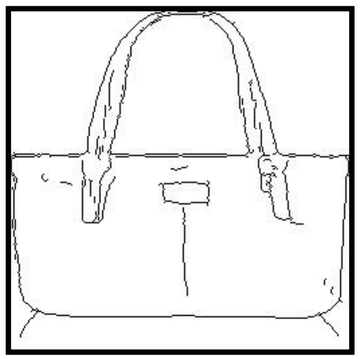
Discriminator

Real too!

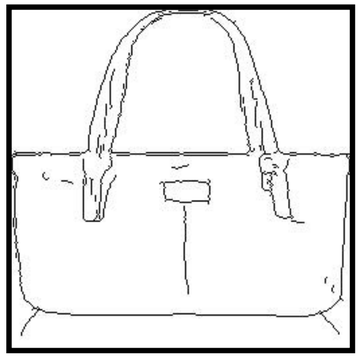
$$\min_G \max_D \mathbb{E}_{x,y}\left[\log D(G(x)) + \log(1 - D(y))\right]$$

x

G(x)



G



D

real or fake ***pair*** ?

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

fake pair

real pair

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

Edges → Images

Input Output Input Output Input Output

Edges from [Xie & Tu, 2015]

slide credit: Phillip Isola & Jun-Yan Zhu

Sketches → Images

Input Output Input Output Input Output

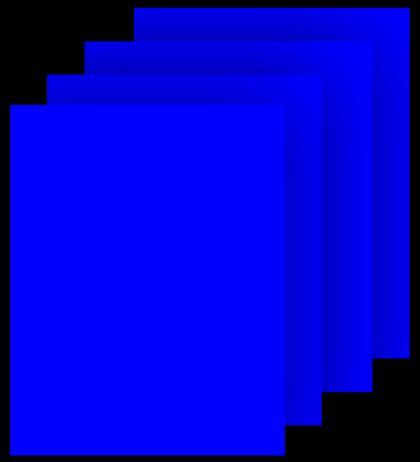
Trained on Edges → Images

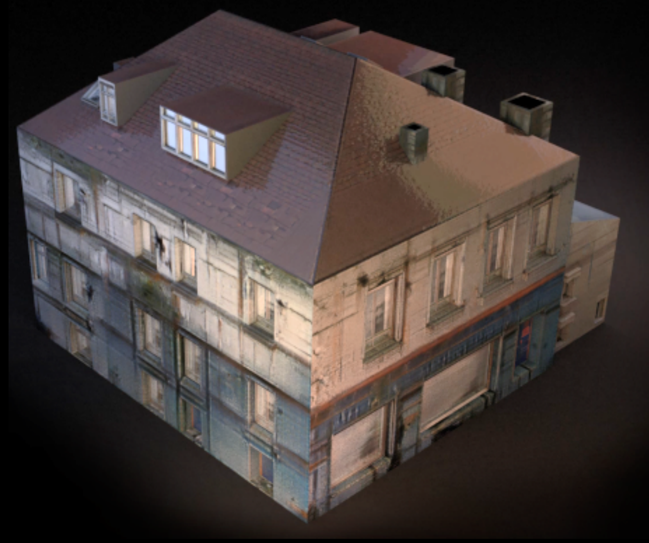
Data from [Eitz, Hays, Alexa, 2012]

slide credit: Phillip Isola & Jun-Yan Zhu

Decomposition into Steps: FrankenGAN

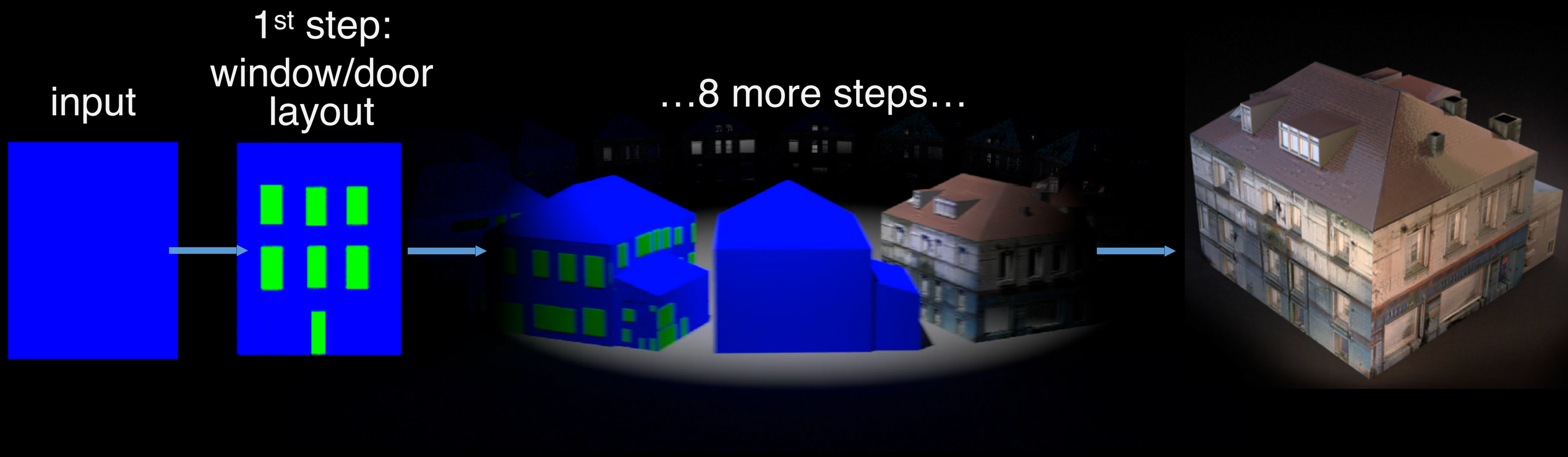
input



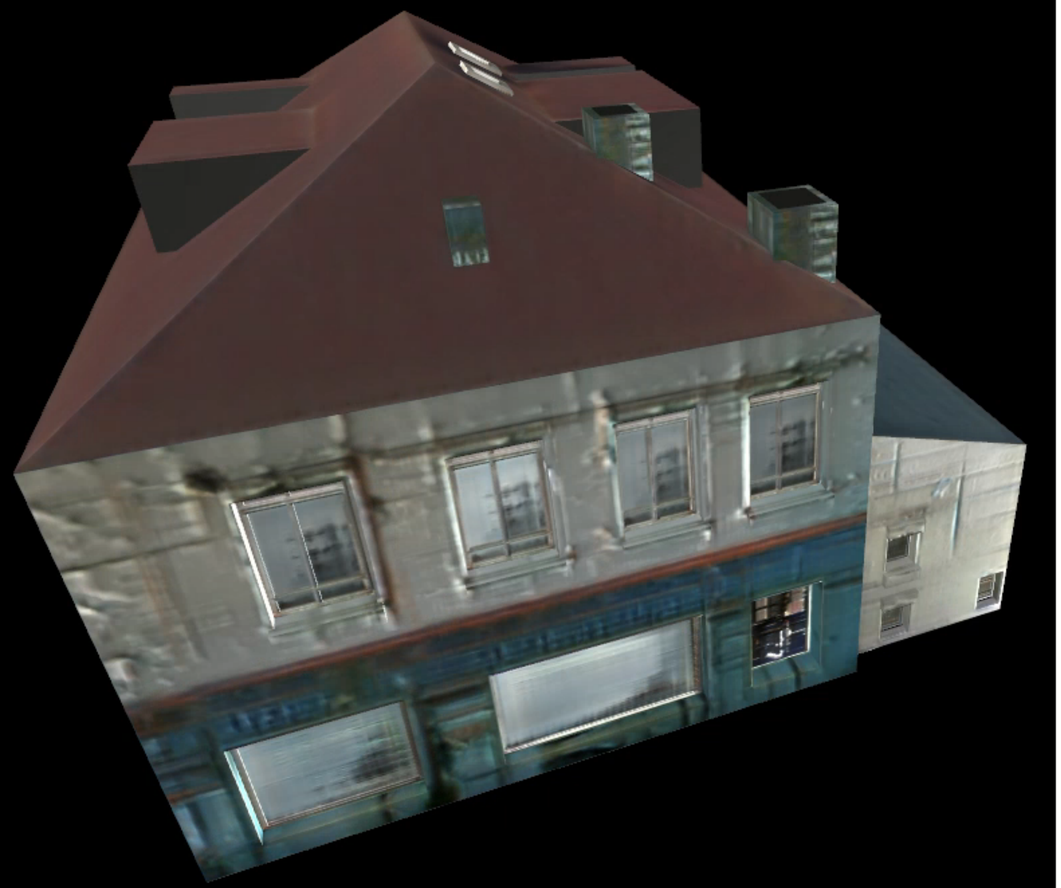


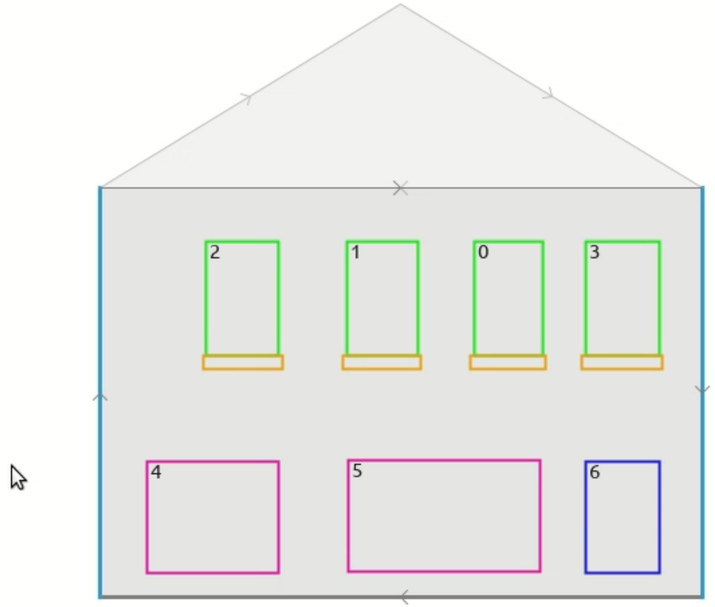
Slide Credit: FrankenGAN, Kelly and Guerrero et al.

Decomposition into Steps: FrankenGAN



Manual Control



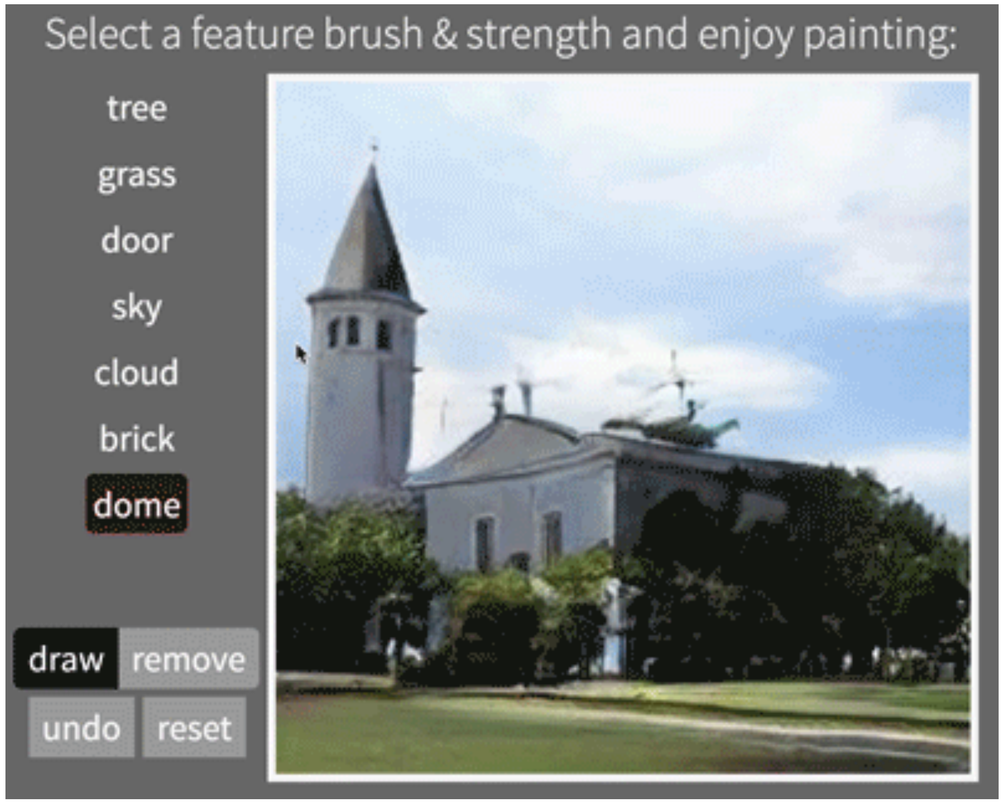


Slide Credit: FrankenGAN, Kelly and Guerrero et al.

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



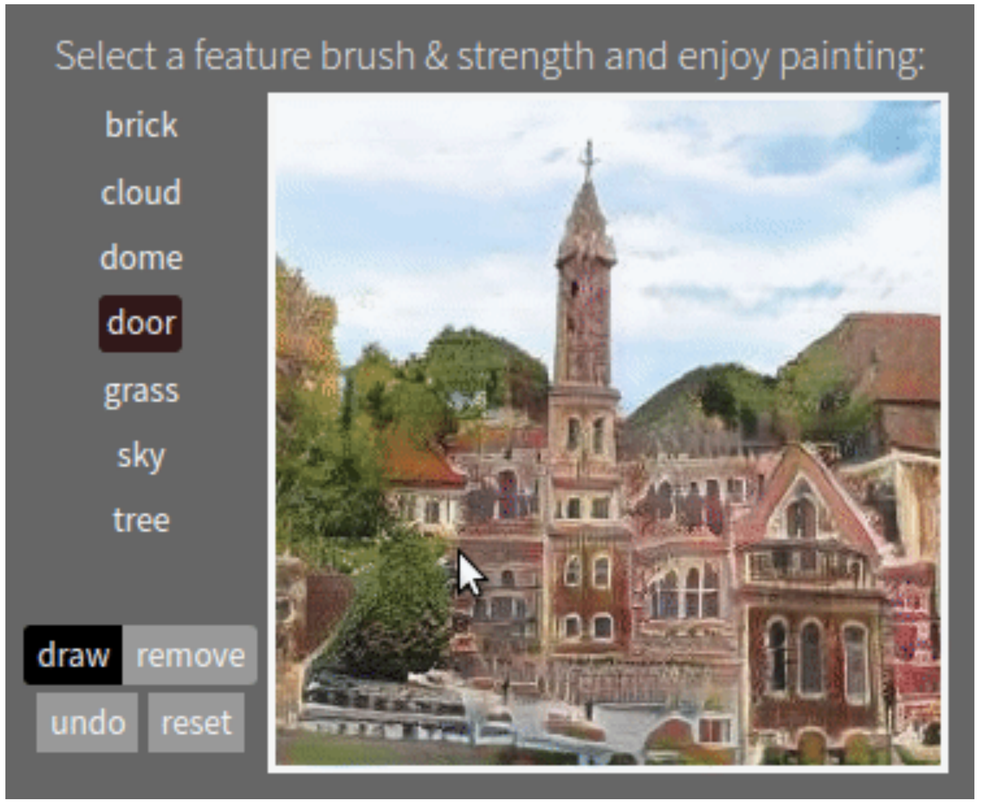


Image Credit:GAN Dissection: Visualizing and Understanding Generative Adversarial Networks, Bau et al.

StructureNet

- **Consistent structure across the dataset**

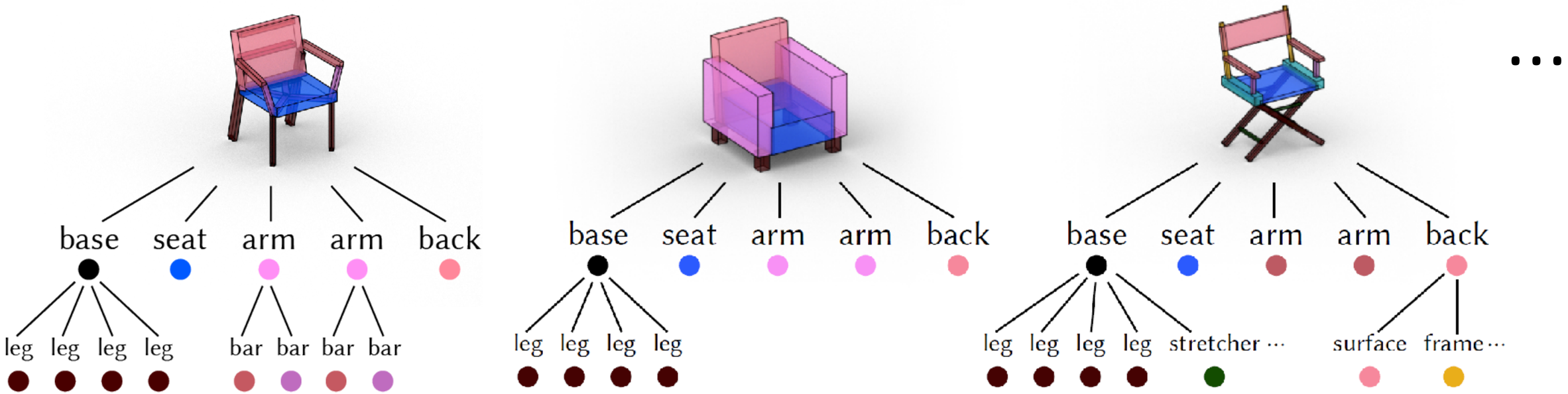
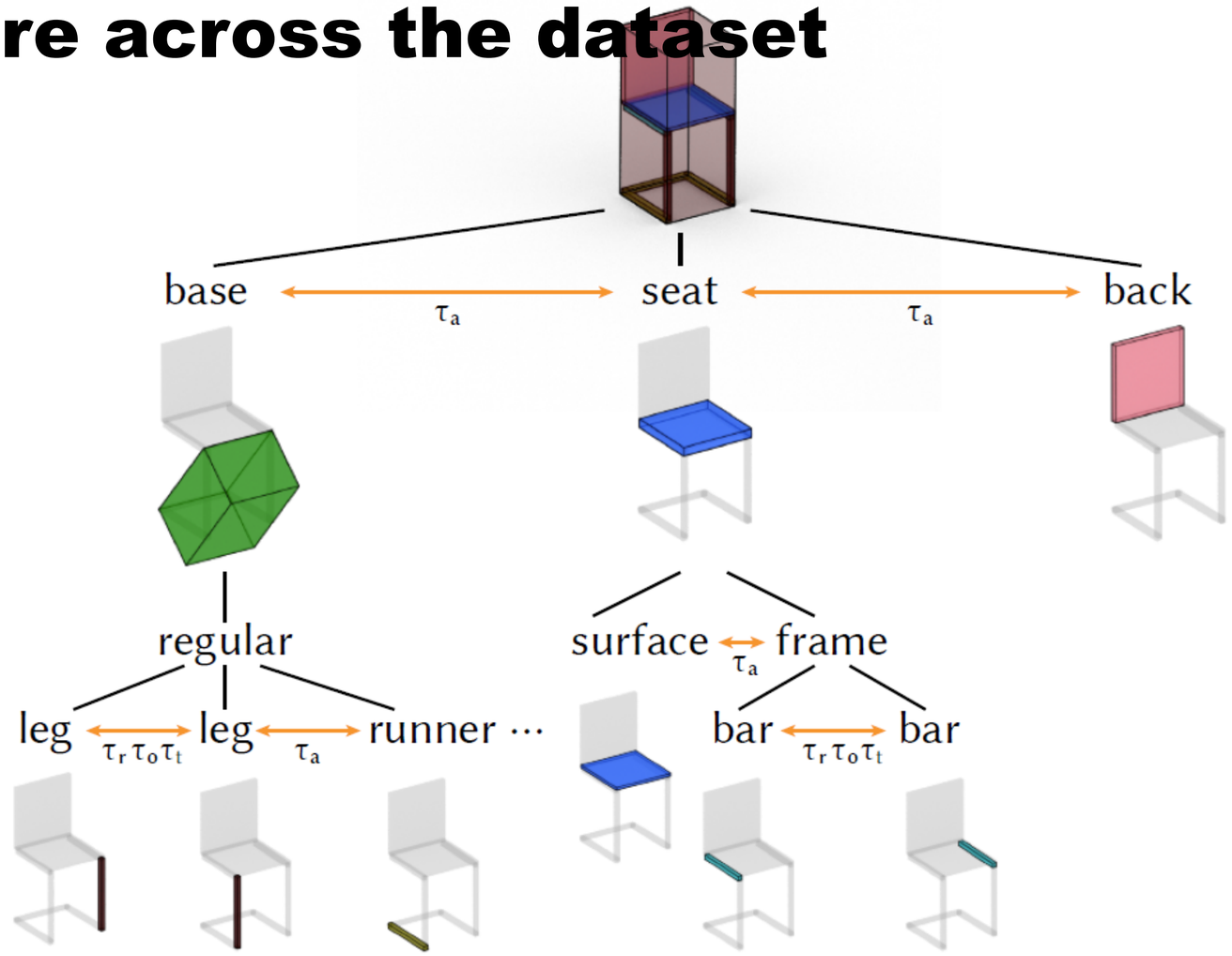


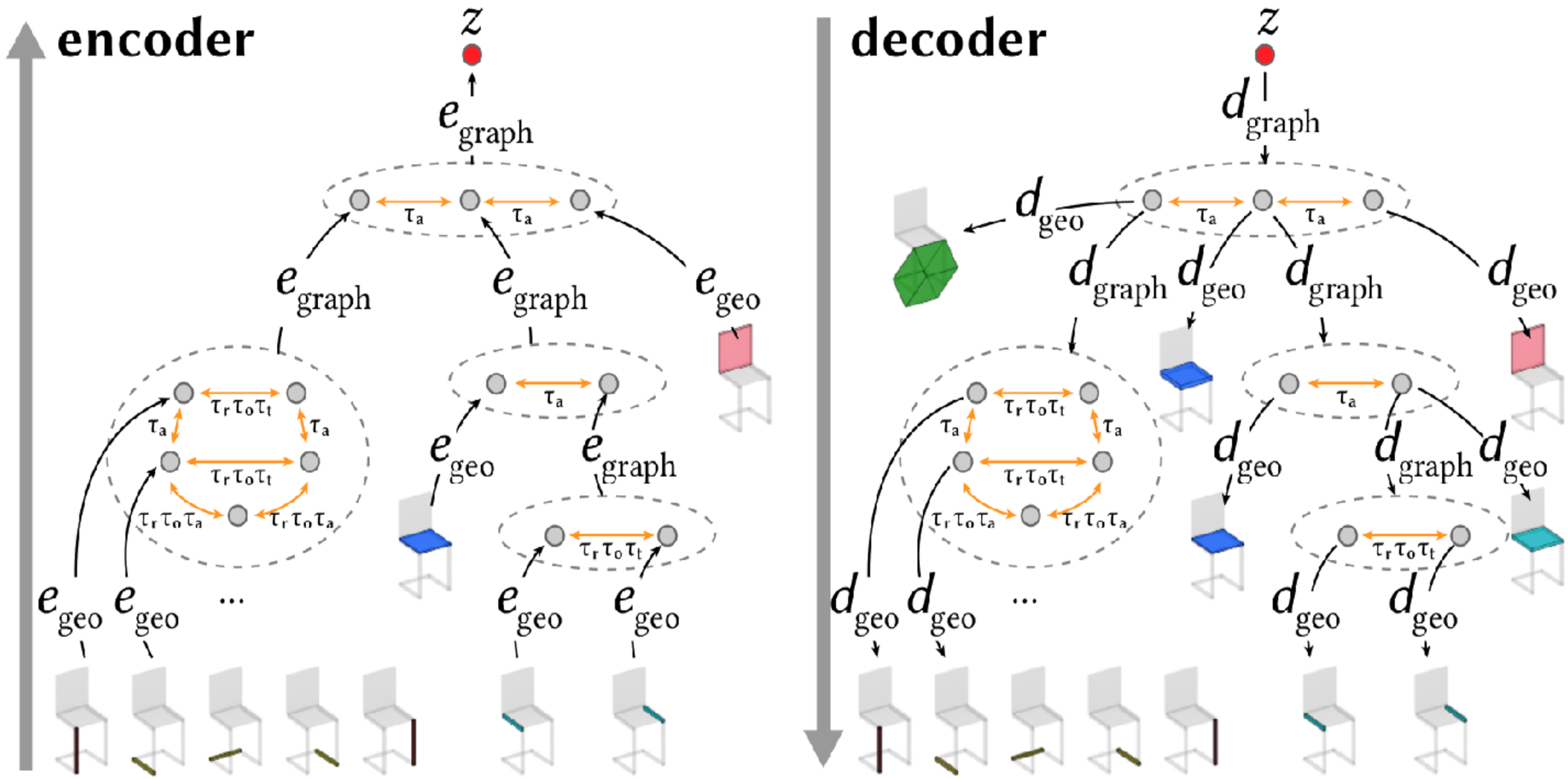
Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

StructureNet

- Consistent structure across the dataset
- Richer structure

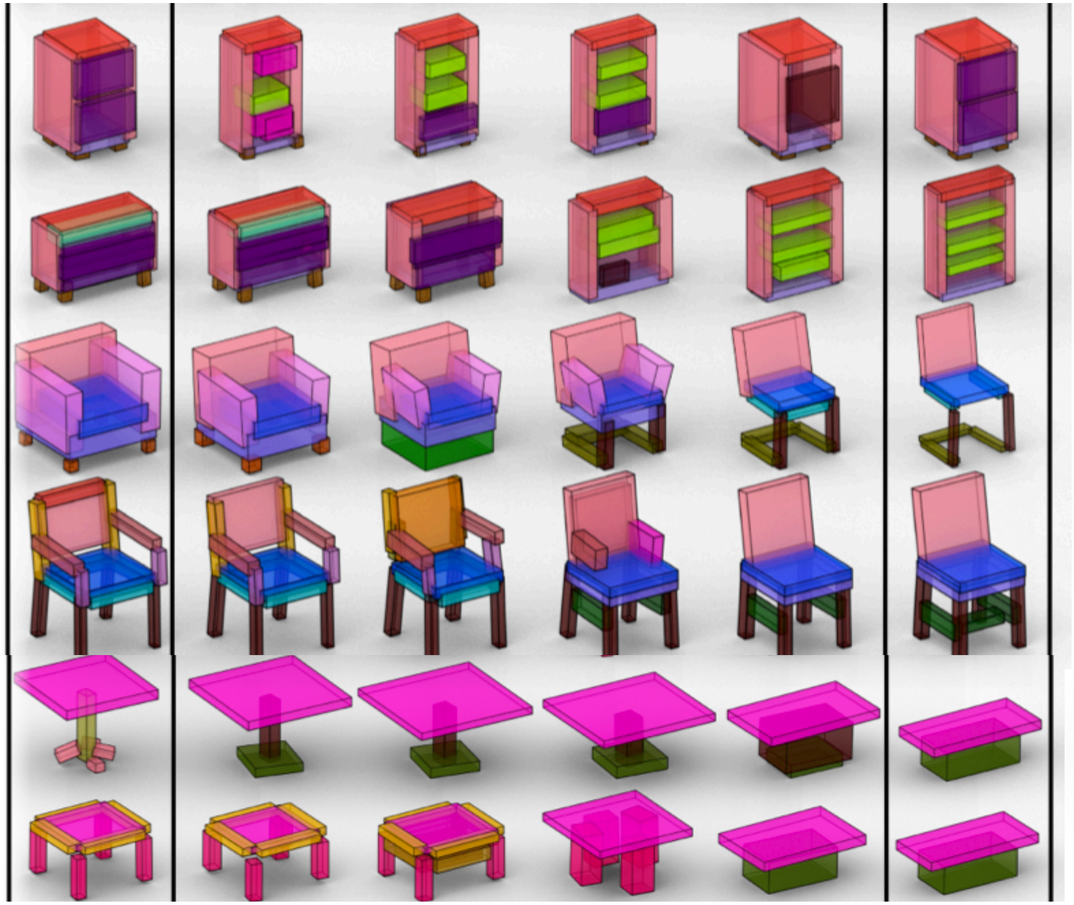


Hierarchical Graph Networks as Encoder and Decoder



StructureNet Results

interpolation



free generation

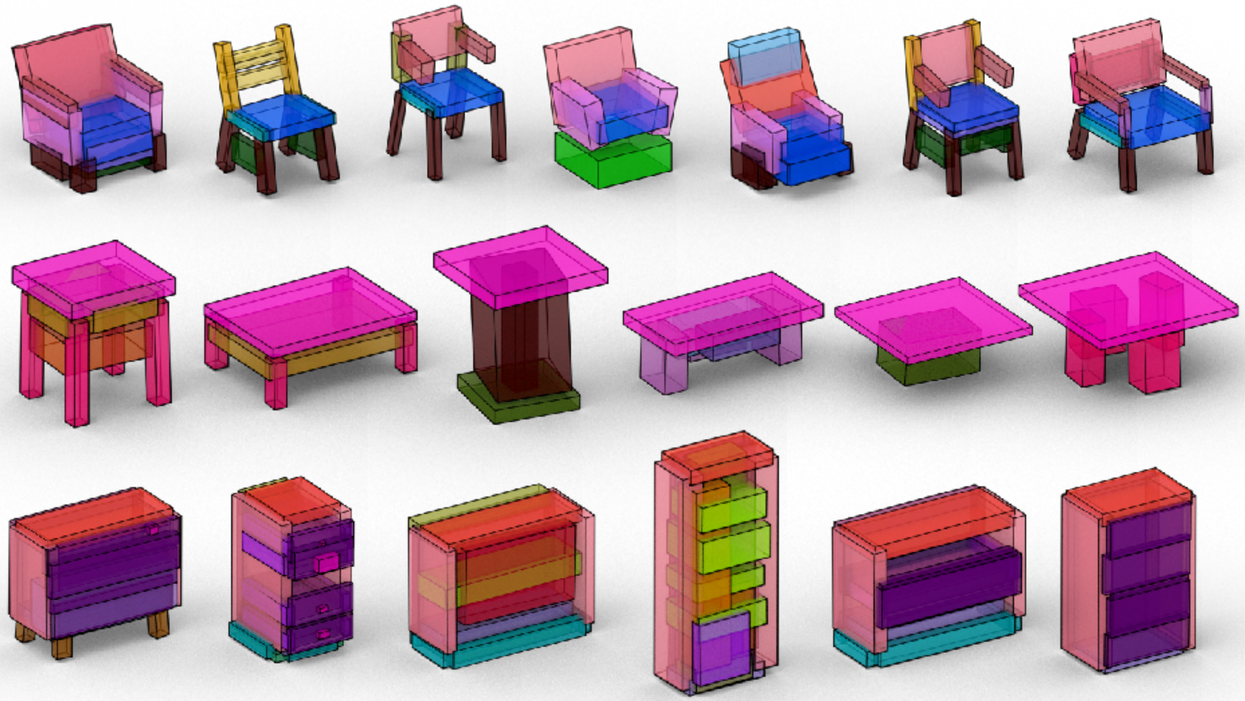


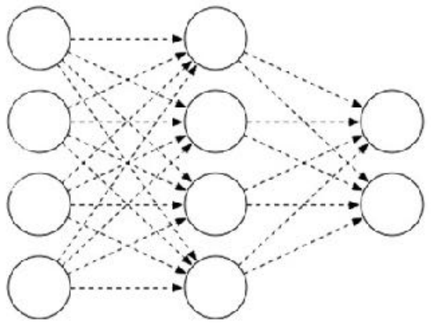
Image Credit: StructureNet: Hierarchical Graph Networks for 3D Shape Generation , Mo and Guerrero et al.

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.



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