CreativeAI: Deep Learning for Graphics

Alternatives to Direct Supervision

Niloy Mitra UCL

Iasonas Kokkinos UCL

Paul Guerrero UCL

Nils Thuerey TUM

Tobias Ritschel UCL

Technische Universität München

Timetable

			Niloy	Paul	Nils
Theory and Basics	Introduction	2:15 pm	X	X	X
	Machine Learning Basics	~ 2:25 pm	X		
	Neural Network Basics	~ 2:55 pm			X
	Feature Visualization	~ 3:25 pm		X	
	Alternatives to Direct Supervision	~ 3:35 pm		X	
	15 min. break				
State of the Art	Image Domains	4:15 pm		X	
	3D Domains	~ 4:45 pm	X		
	Motion and Physics	~ 5:15 pm			X
	Discussion	~ 5:45 pm	X	X	X

Unsupervised Learning

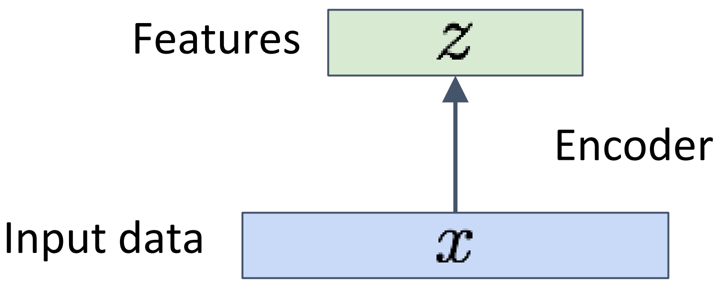
- There is no direct ground truth for the quantity of interest

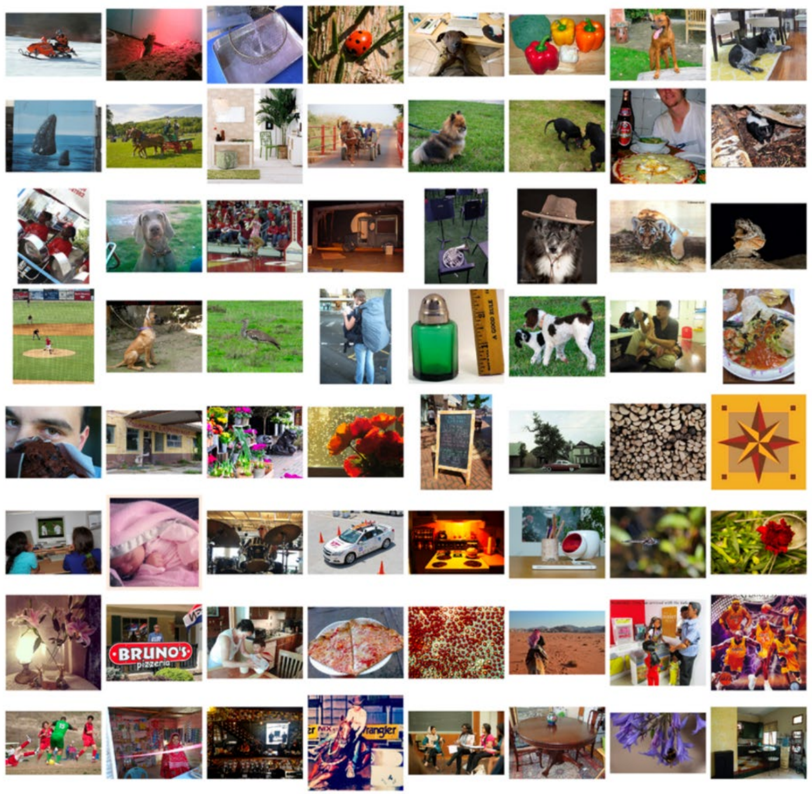
- Autoencoders
- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)

Autoencoders

Goal: Meaningful features that capture the main factors of variation in the dataset

- These are good for classification, clustering, exploration, generation, …
- We have no ground truth for them



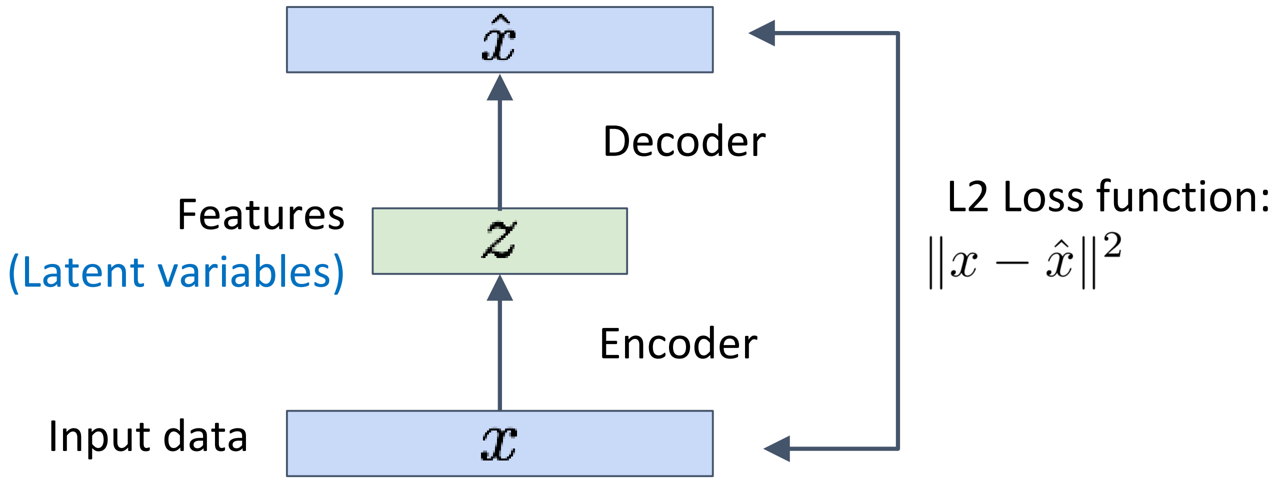


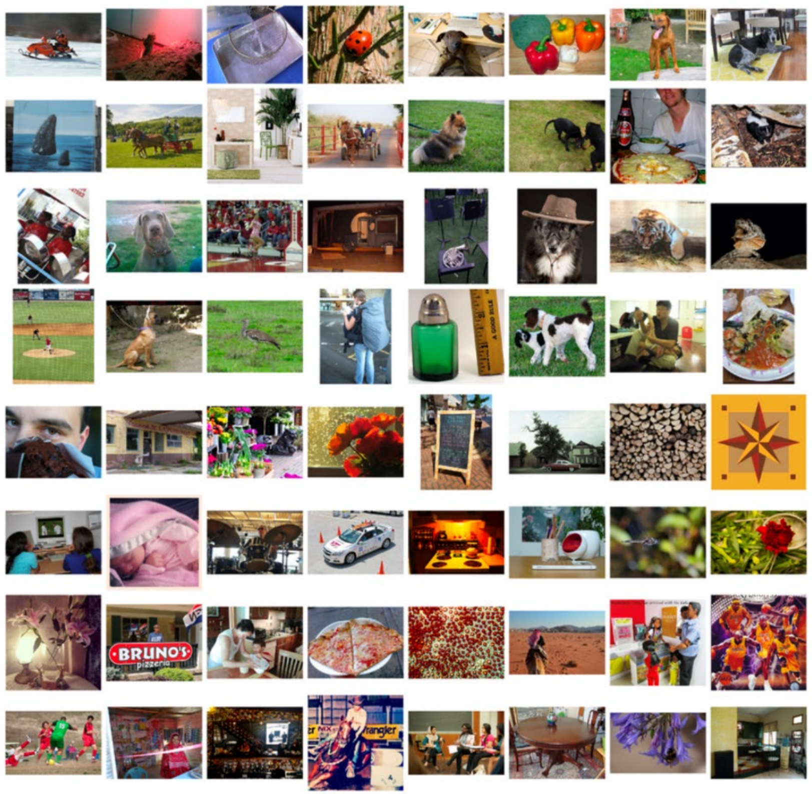
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

Goal: Meaningful features that capture the main factors of variation

Features that can be used to reconstruct the image



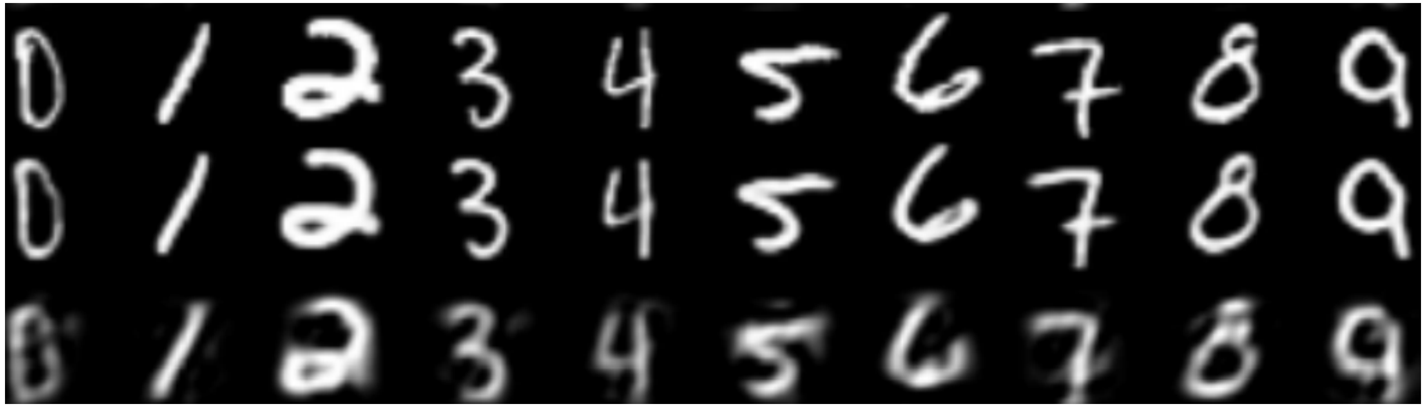


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders

Linear Transformation for Encoder and Decoder give result close to PCA

Deeper networks give better reconstructions, since basis can be non-linear



Original

Autoencoder

PCA

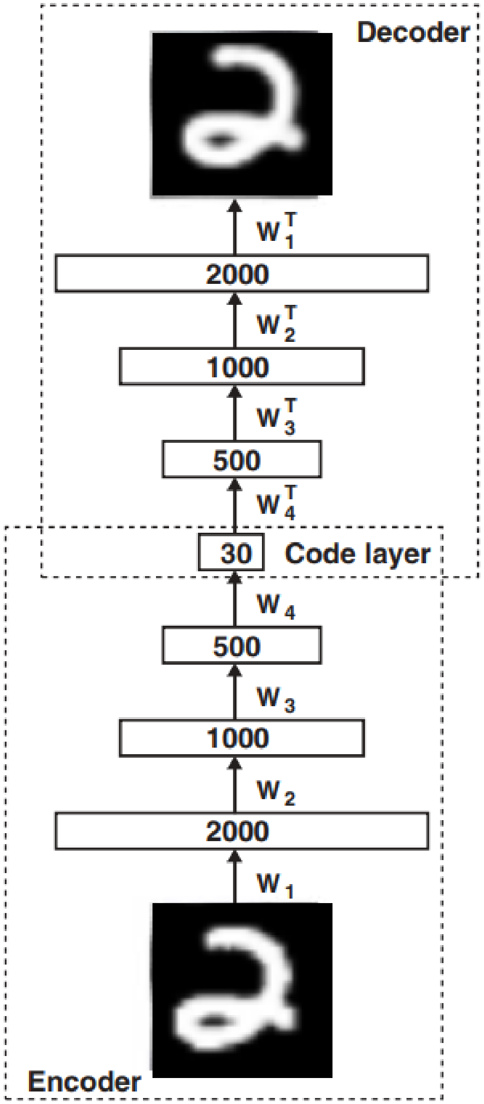
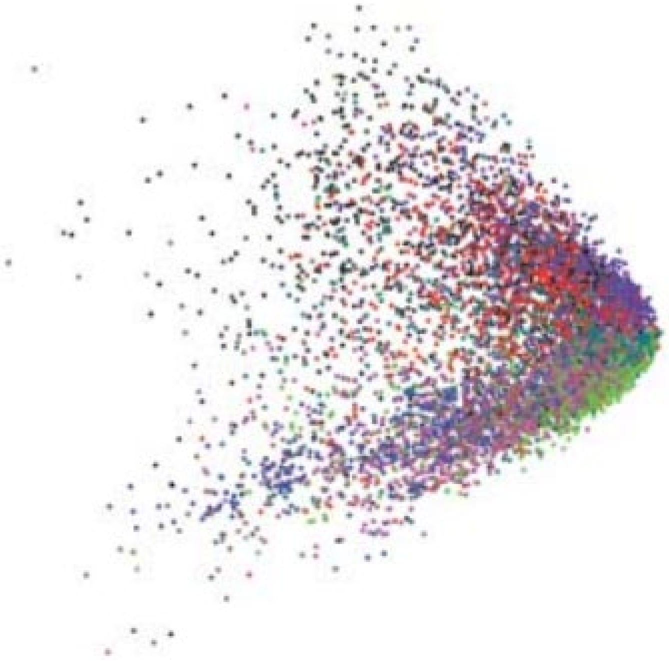


Image Credit: Reducing the Dimensionality of Data with Neural Networks, . Hinton and Salakhutdinov

Example: Document Word Prob. → 2D Code

PCA-based



Autoencoder

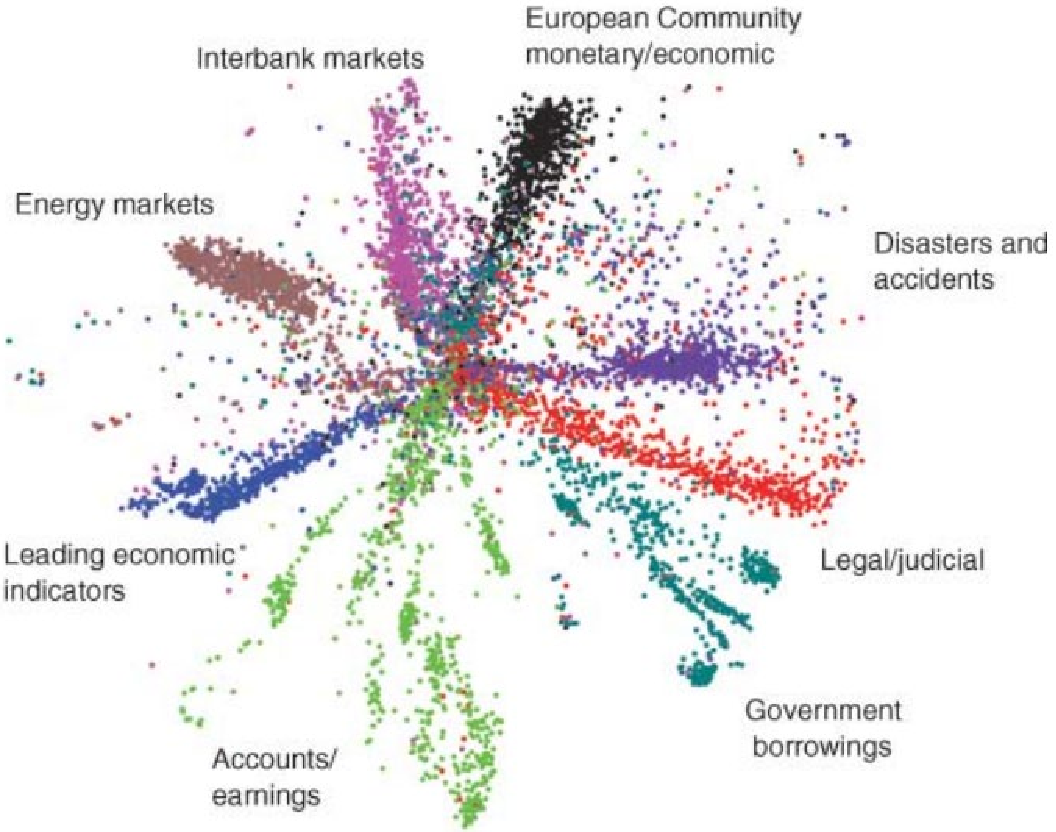
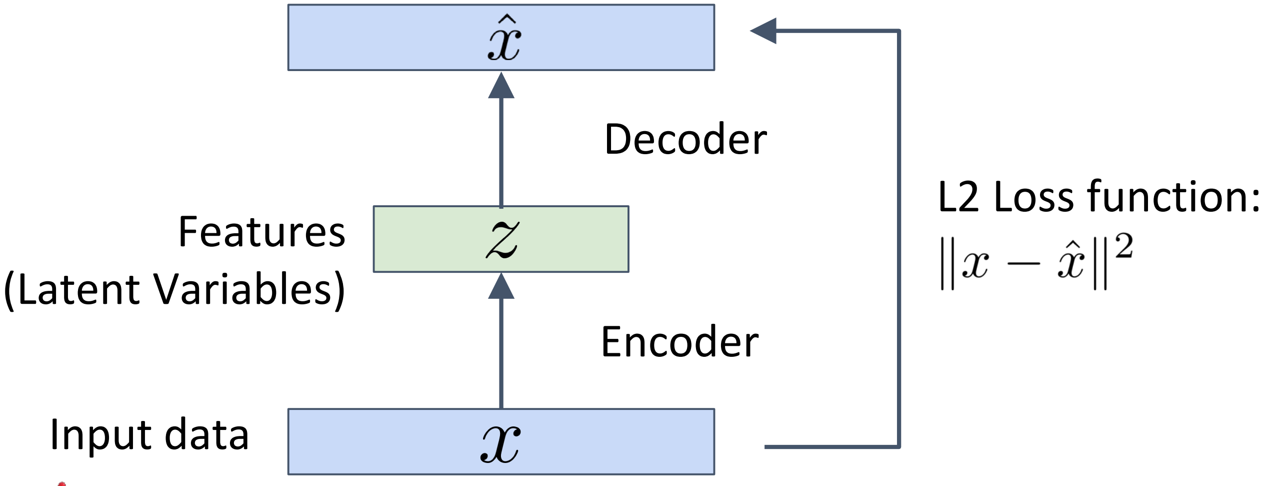


Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

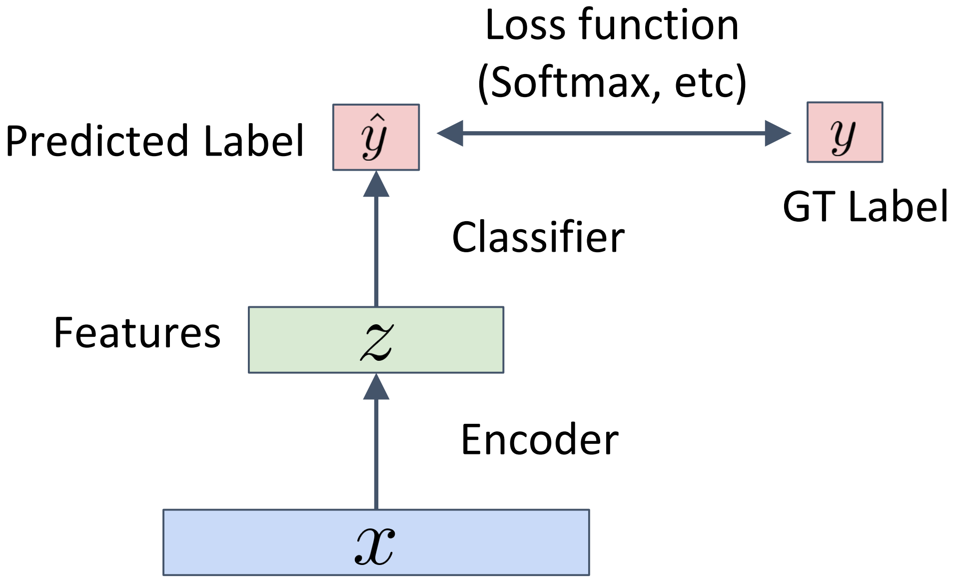
Example: Semi-Supervised Classification

- Many images, but few ground truth labels

start unsupervised
train autoencoder on many images



supervised fine-tuning
train classification network on labeled images



Code example

Autoencoder

geometry.cs.ucl.ac.uk/creativeai

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: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* 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

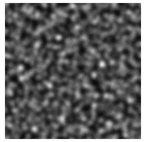
Generated

Image credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

Generative Models

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

Generator with
parameters θ



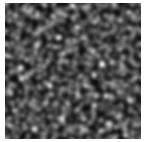
$p(z)$

known and
easy to sample from

Generative Models

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

Generator with parameters θ



$p(z)$

known and easy to sample from

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

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

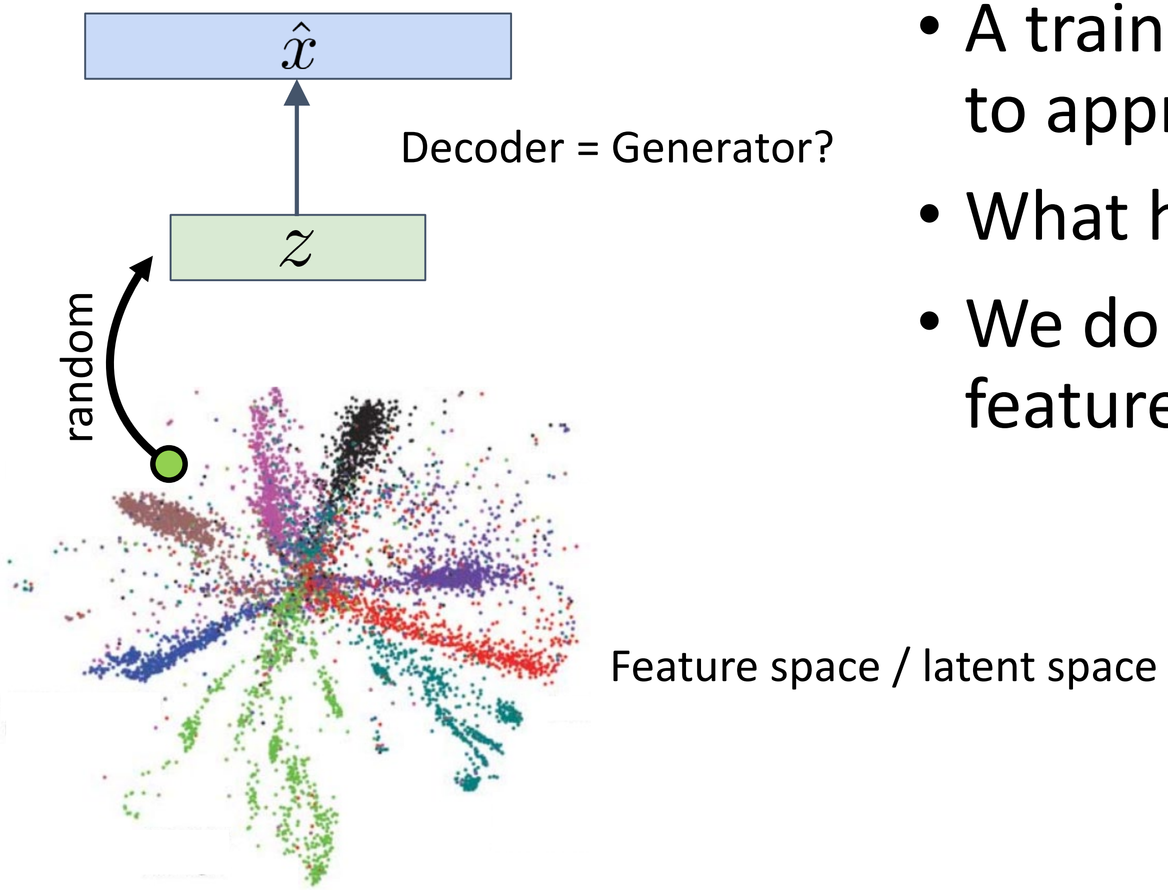
Variational Autoencoders (VAEs)

2) Adversarial game:

Discriminator distinguishes $p_\theta(x)$ and $p_{\text{data}}(x)$ vs *Generator* makes it hard to distinguish

Generative Adversarial Networks (GANs)

Autoencoders as Generative Models?

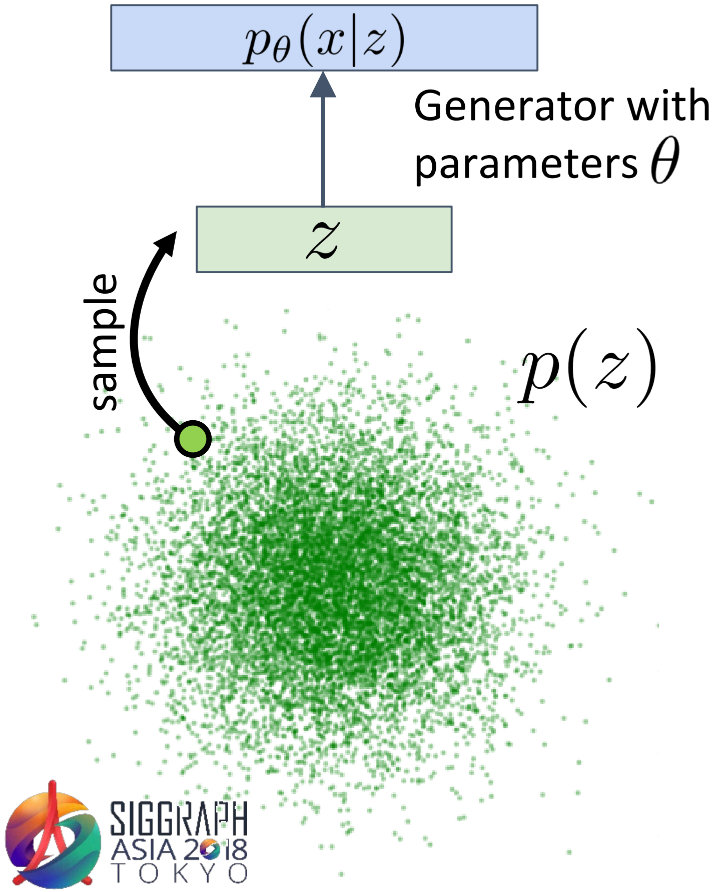


- A trained decoder transforms some features z to approximate samples from $p_{\text{data}}(x)$
- What happens if we pick a random z?
- We do not know the distribution $p(z)$ of features that decode to likely samples

SIGGRAPH ASIA 2018 TOKYO

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

Variational Autoencoders (VAEs)



- Pick a parametric distribution $p(z)$ for features
- The generator maps $p(z)$ to an image distribution $p_\theta(x)$ (where θ are parameters)

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

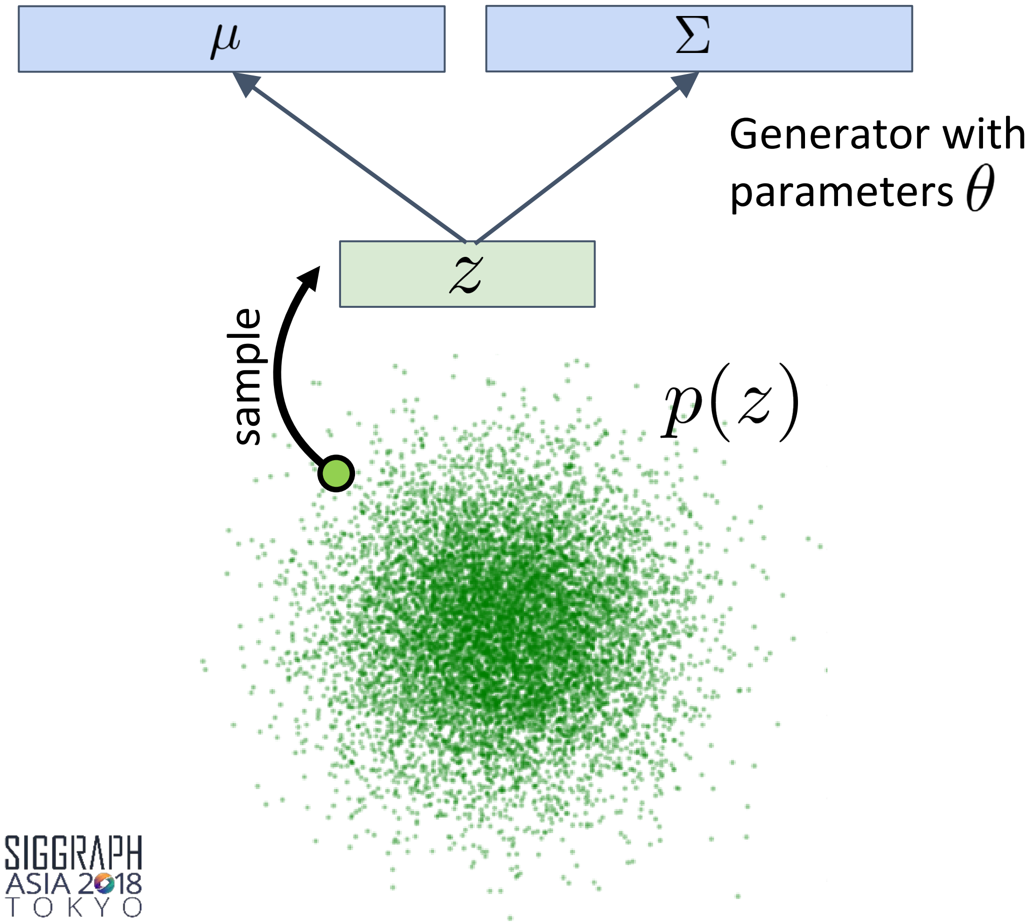
- Train the generator to maximize the likelihood of the data in $p_\theta(x)$:

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

Outputting a Distribution

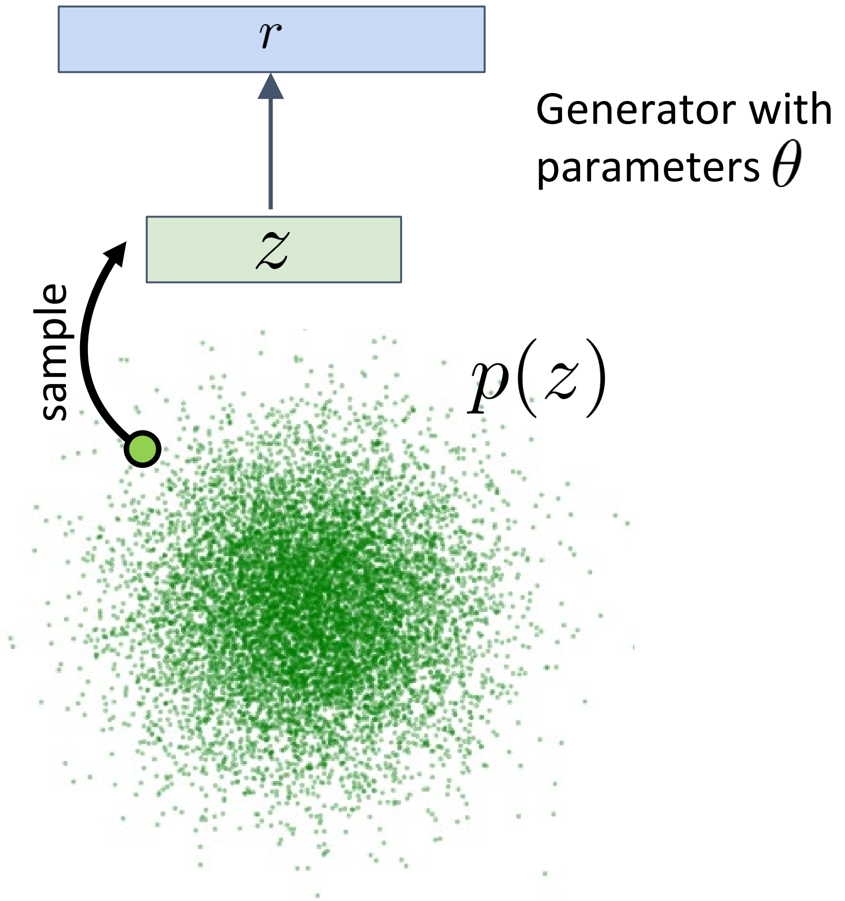
Normal distribution

$$p_\theta(x|z) = N(x; \mu(z), \Sigma(z))$$

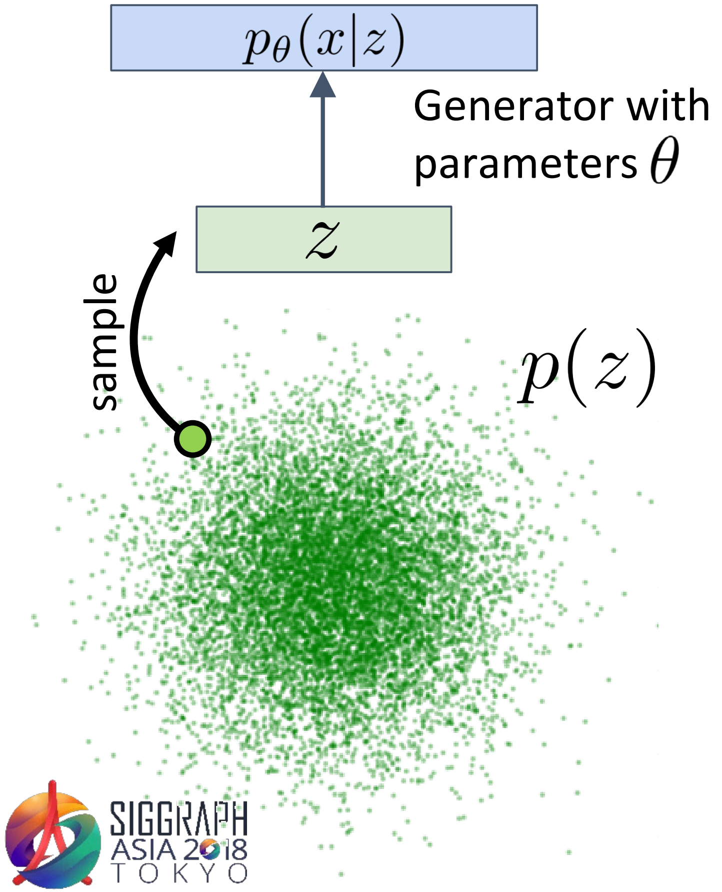


Bernoulli distribution

$$p_\theta(x|z) = Bern(x; r(z))$$



Variational Autoencoders (VAEs)



- Pick a parametric distribution $p(z)$ for features
- The generator maps $p(z)$ to an image distribution $p_\theta(x)$ (where θ are parameters)

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

- Train the generator to maximize the likelihood of the data in $p_\theta(x)$:

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

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)

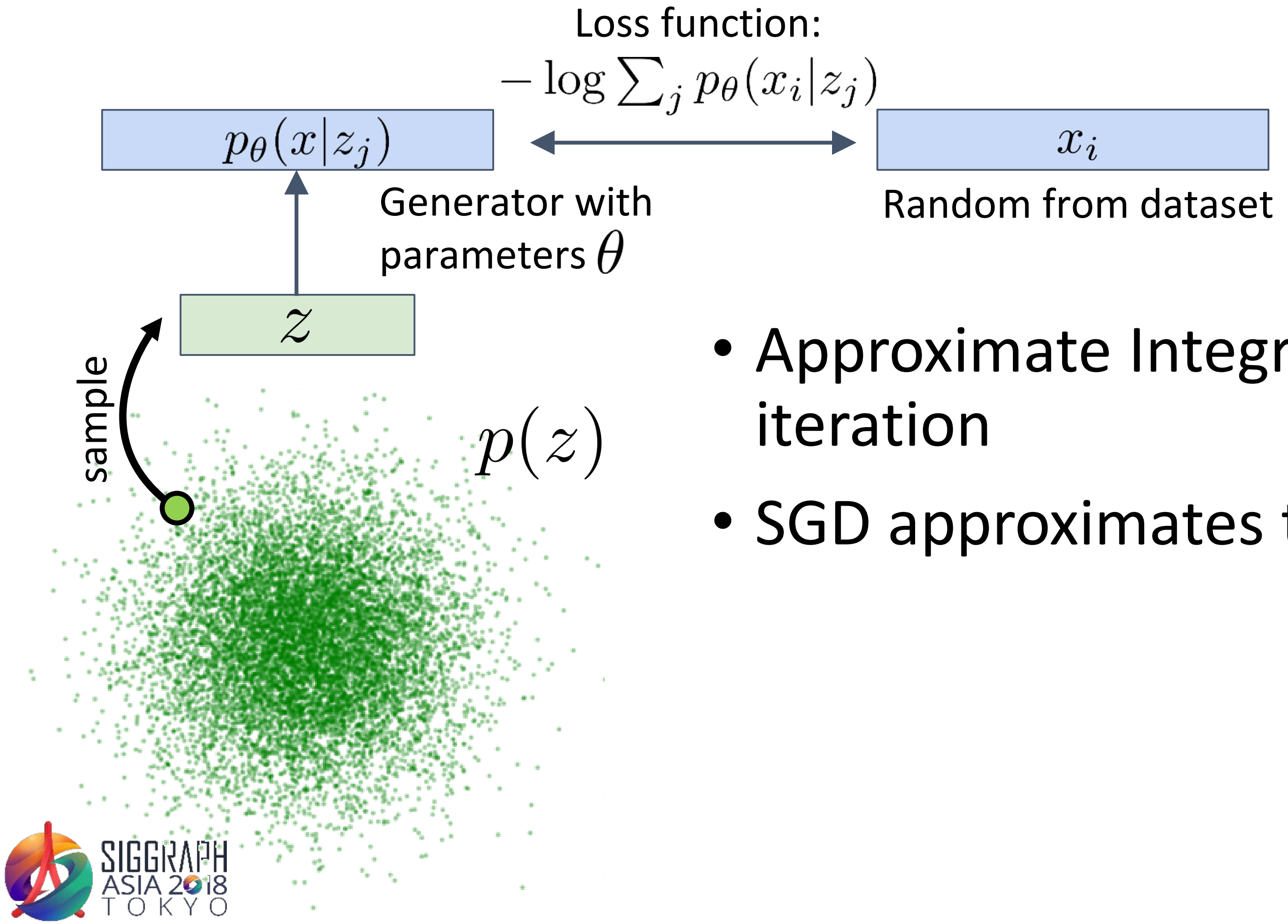
Maximum likelihood of data in generated distribution:

$$\theta^* = \arg\max_\theta \sum_{x_i \in \text{data}} \log \int p_\theta(x_i|z)\ p(z)\ dz$$

$$\theta^* \approx \arg\max_\theta \mathbb{E}_{x_i \sim p_{\text{data}}(x)} \log \mathbb{E}_{z \sim p(z)}\ p_\theta(x_i|z)$$

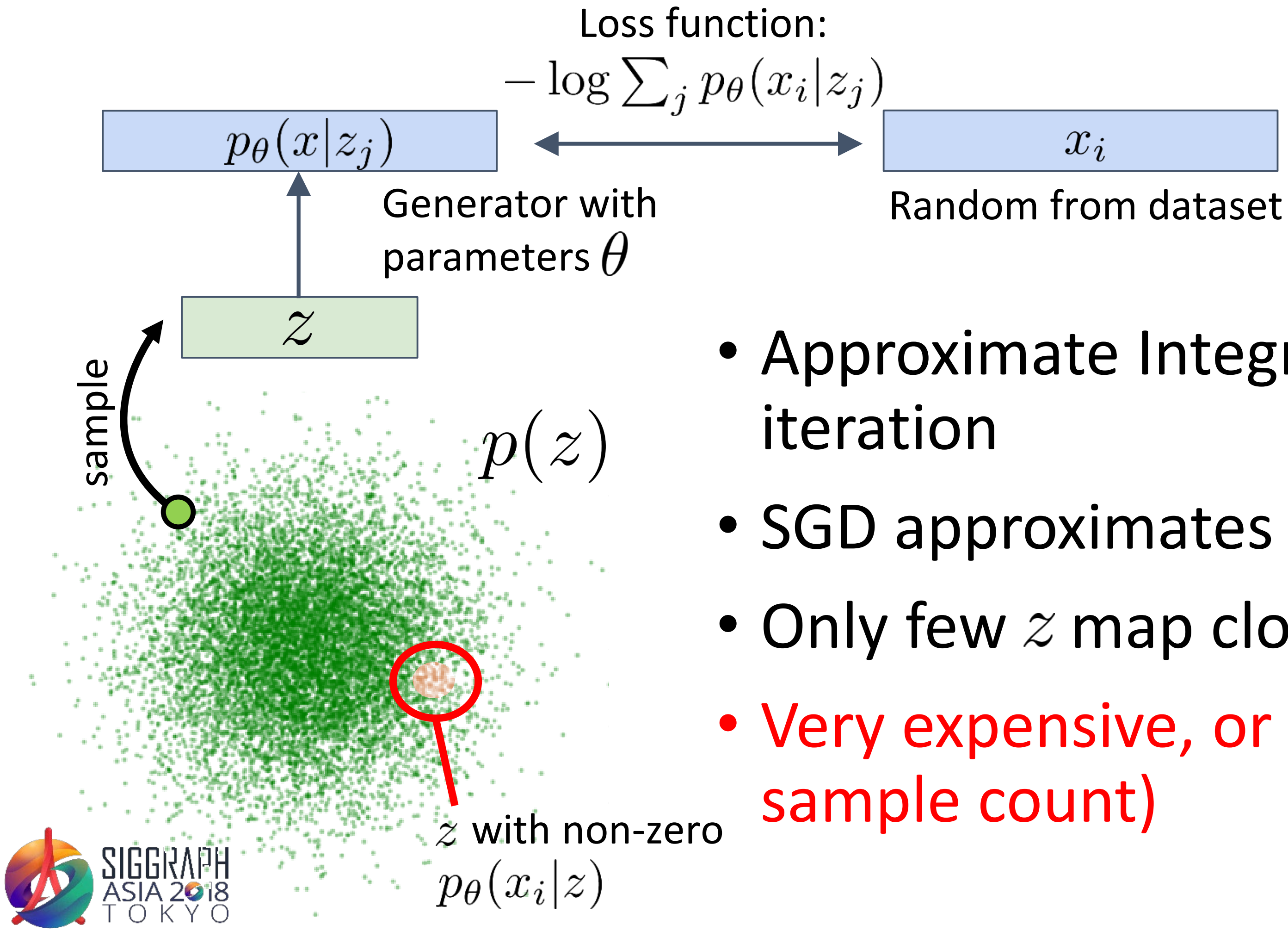
- Approximate Integral with Monte-Carlo in each iteration
- SGD approximates the sum over data

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



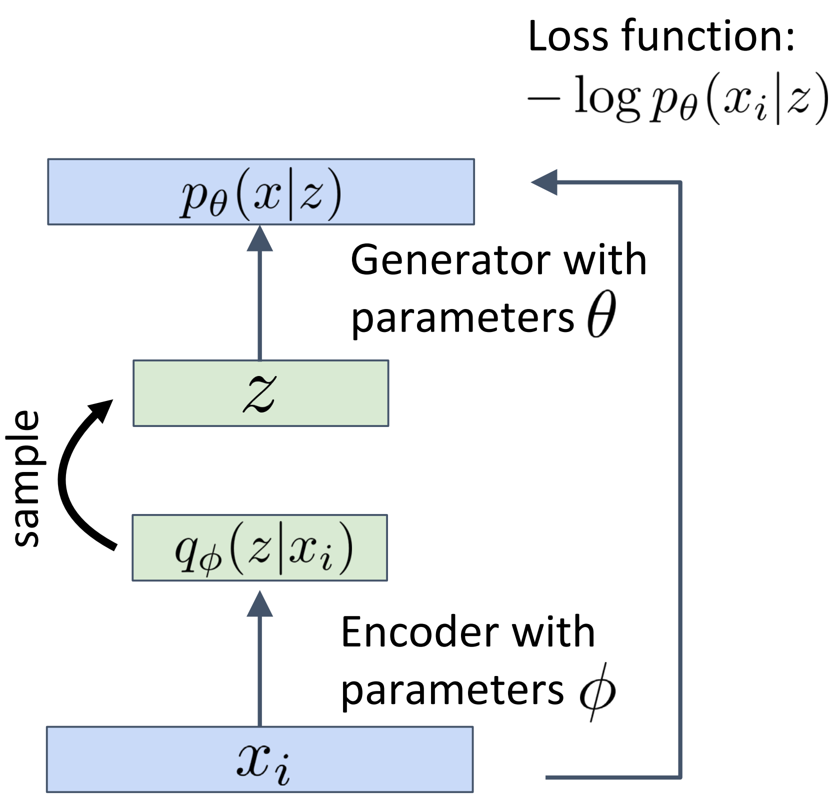
- Approximate Integral with Monte-Carlo in each iteration
- SGD approximates the expectancy over data

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



- Approximate Integral with Monte-Carlo in each iteration
- SGD approximates the expectancy over data
- Only few z map close to a given x_i
- Very expensive, or very inaccurate (depending on sample count)

Variational Autoencoders (VAEs): The Encoder



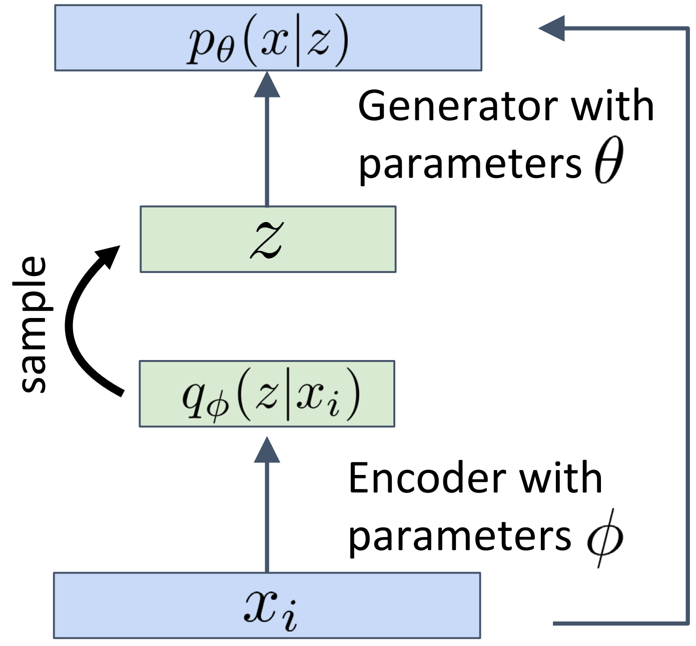
$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

- During training, another network can learn a distribution of good z for a given x_i
- $q_\phi(z|x_i)$ should be much smaller than $p(z)$
- A single sample is good enough

Variational Autoencoders (VAEs): The Encoder

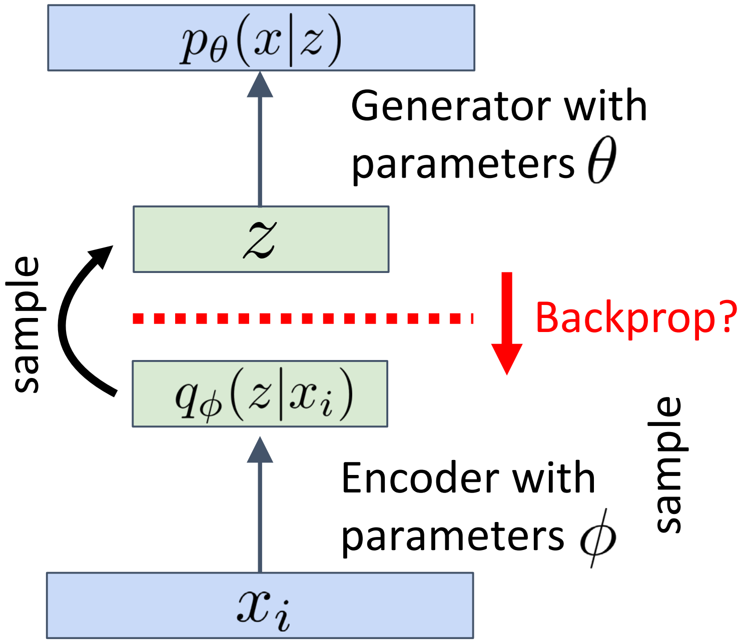
Loss function:

$$-\log p_\theta(x_i|z) + KL(\ q_\phi(z|x_i)\ \|\ p(z)\ )$$



- Can we still easily sample a new z?
- Need to make sure $q_\phi(z|x_i)$ approximates $p(z)$
- Regularize with KL-divergence
- Negative loss can be shown to be a lower bound for the likelihood, and equivalent if $q_\phi(z|x) = p_\theta(z|x)$

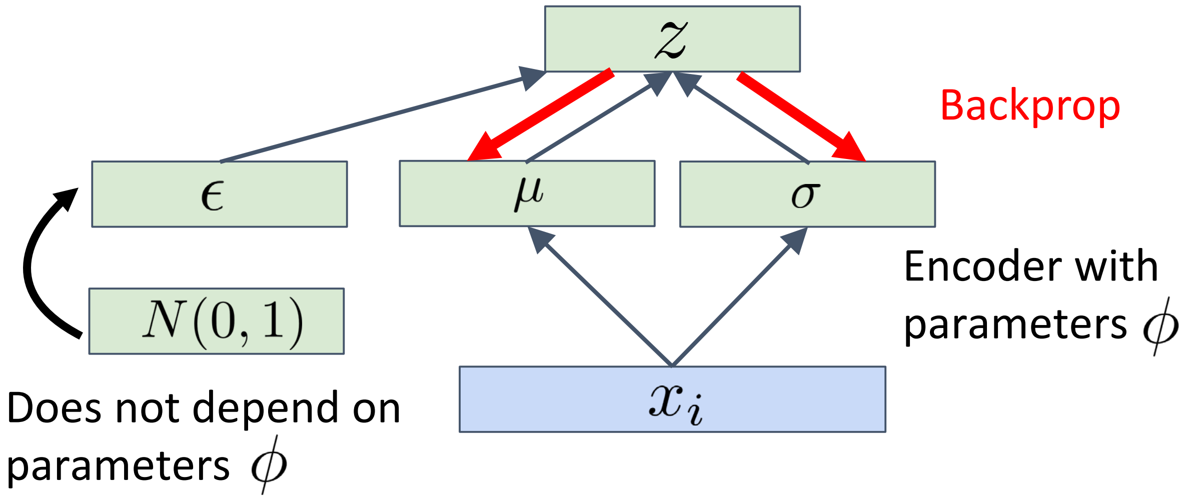
Reparameterization Trick



Example when $q_\phi(z|x_i) = N(z; \mu(x_i), \sigma(x_i))$:

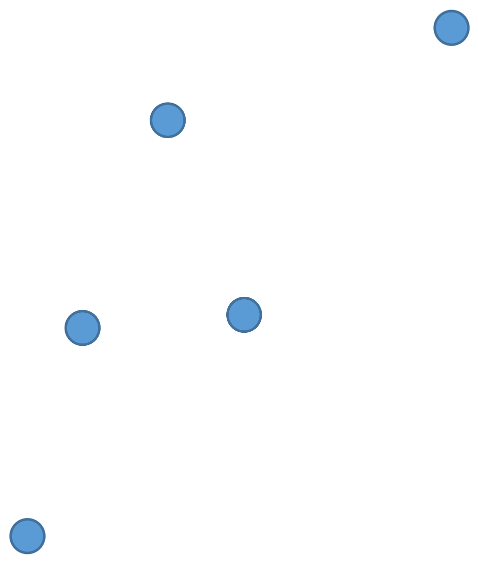
$z = \sigma + \mu \cdot \epsilon$, where $\epsilon \sim N(0,1)$

$$\frac{\partial z}{\partial \phi} = \frac{\partial \mu}{\partial \phi} + \frac{\partial \sigma}{\partial \phi} \cdot \epsilon$$

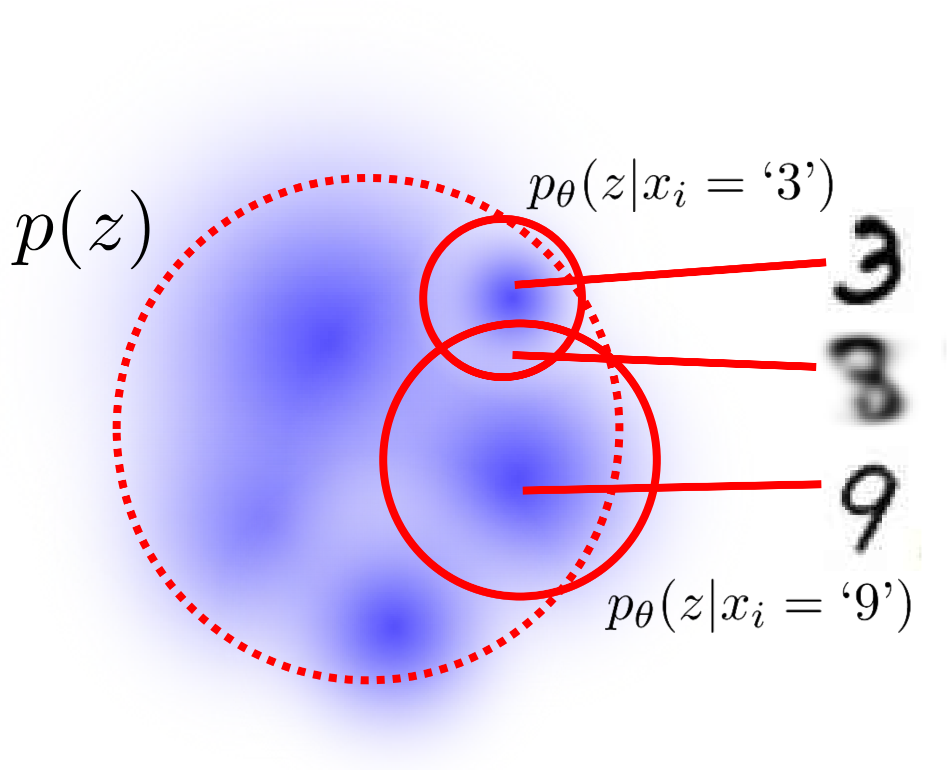


Feature Space of Autoencoders vs. VAEs

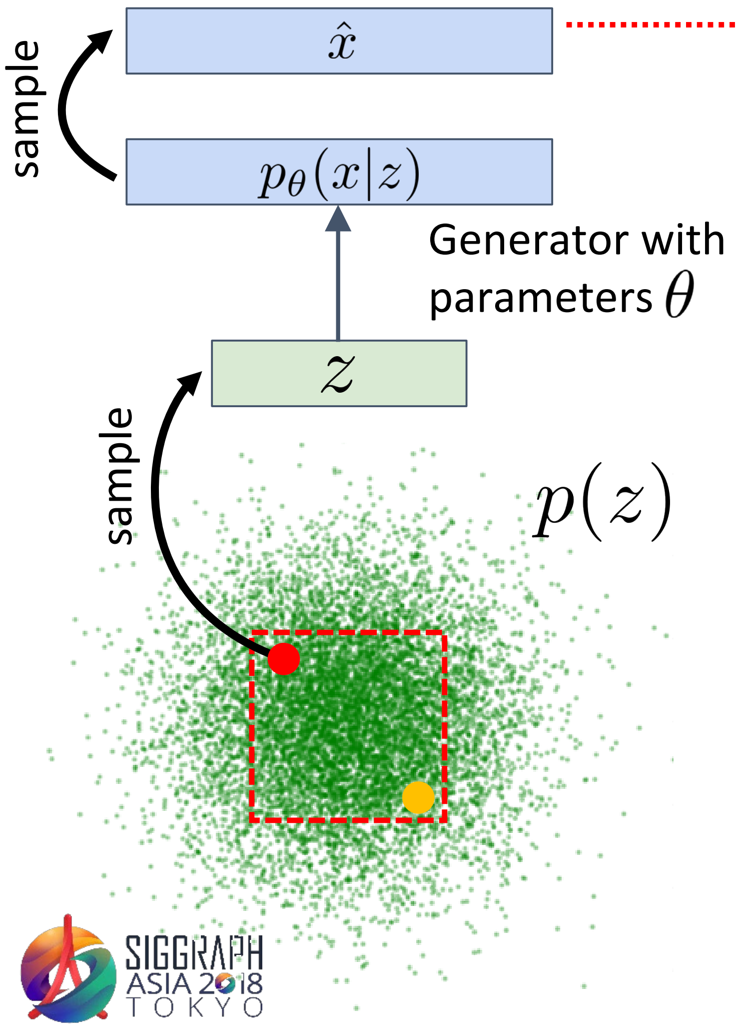
Autoencoder



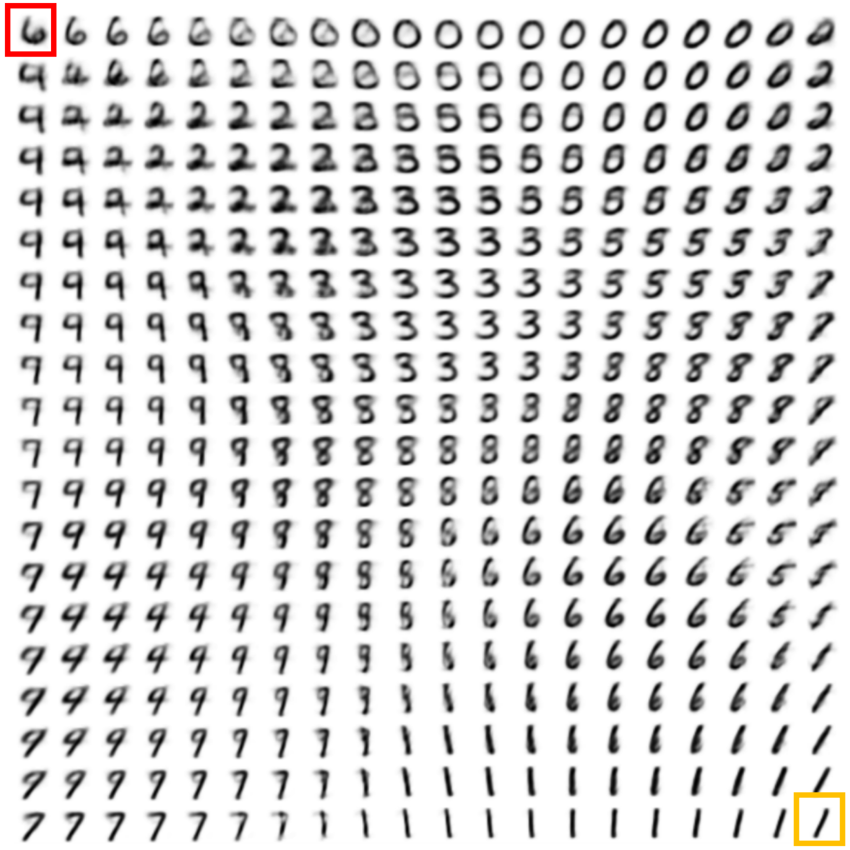
VAE



Generating Data



MNIST



Frey Faces

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

Demos

VAE on MNIST

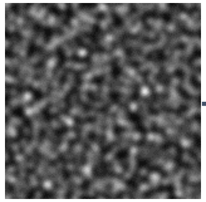
https://www.siarez.com/projects/variational-autoencoder

Code example

Variational Autoencoder

geometry.cs.ucl.ac.uk/creativeai

Generative Adversarial Networks



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

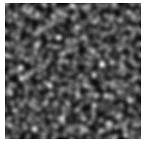
from dataset

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

Generative Models

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

Generator with parameters θ



$p(z)$

known and easy to sample from

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

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

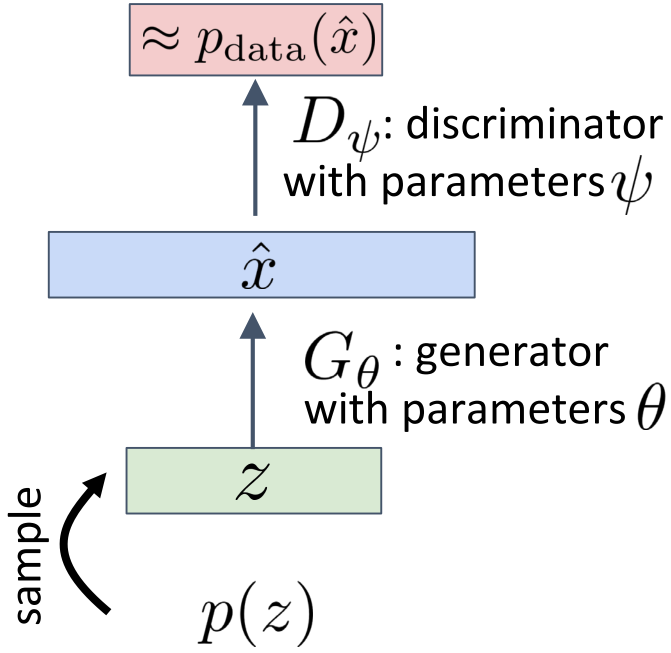
Variational Autoencoders (VAEs)

2) Adversarial game:

Discriminator distinguishes $p_\theta(x)$ and $p_{\text{data}}(x)$ vs *Generator* makes it hard to distinguish

Generative Adversarial Networks (GANs)

Why Adversarial?



- If discriminator approximates $p_{\text{data}}(x)$:
- x^* at maximum of $p_{\text{data}}(x)$ has lowest loss
- Optimal $p_\theta(x)$ has single mode at x^*, small variance

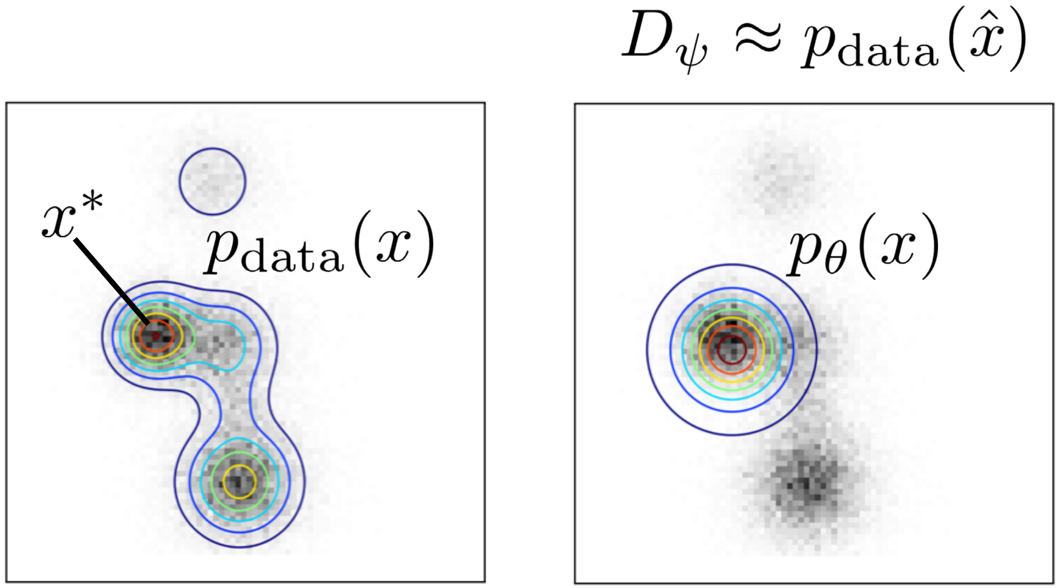
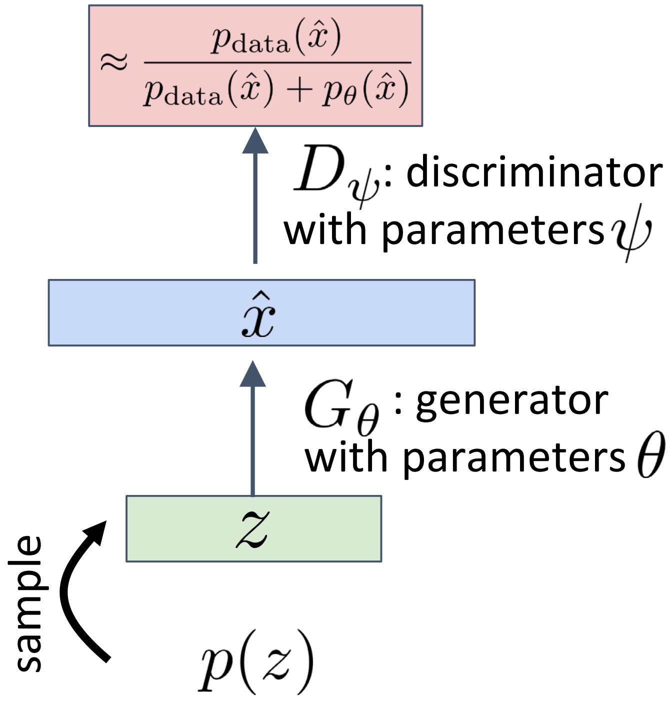


Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár

Why Adversarial?



- For GANs, the discriminator instead approximates:

$$\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$$

depends on the generator

$D_\psi \approx p_{\text{data}}(\hat{x})$ $\quad D_\psi \approx \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_\theta(x)}$

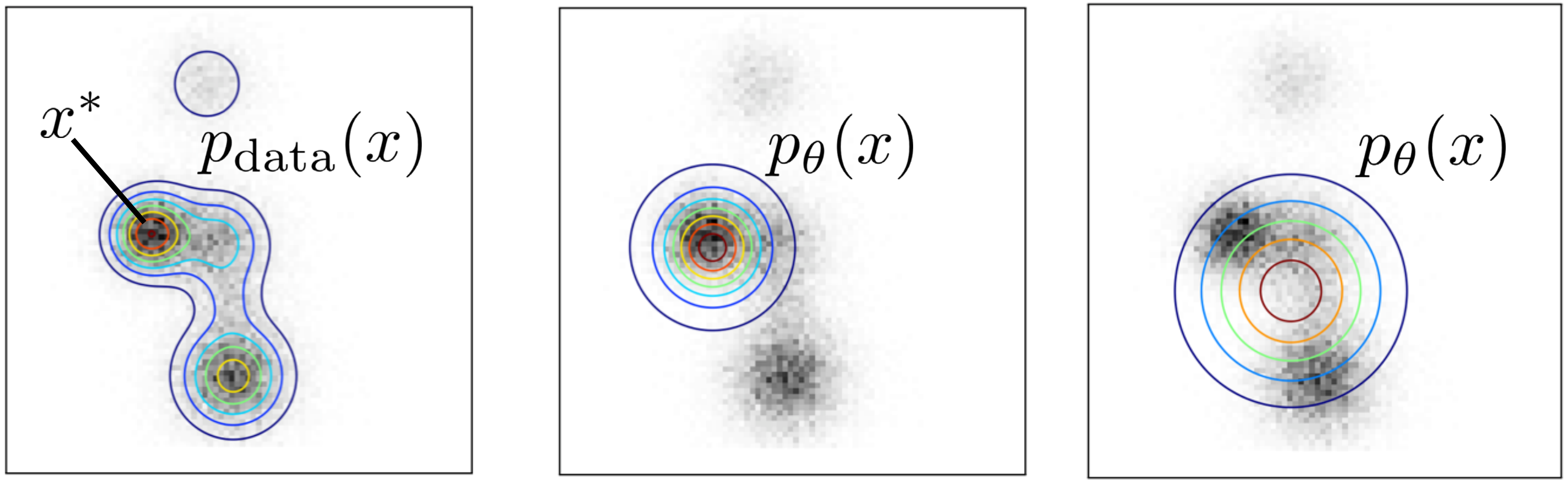
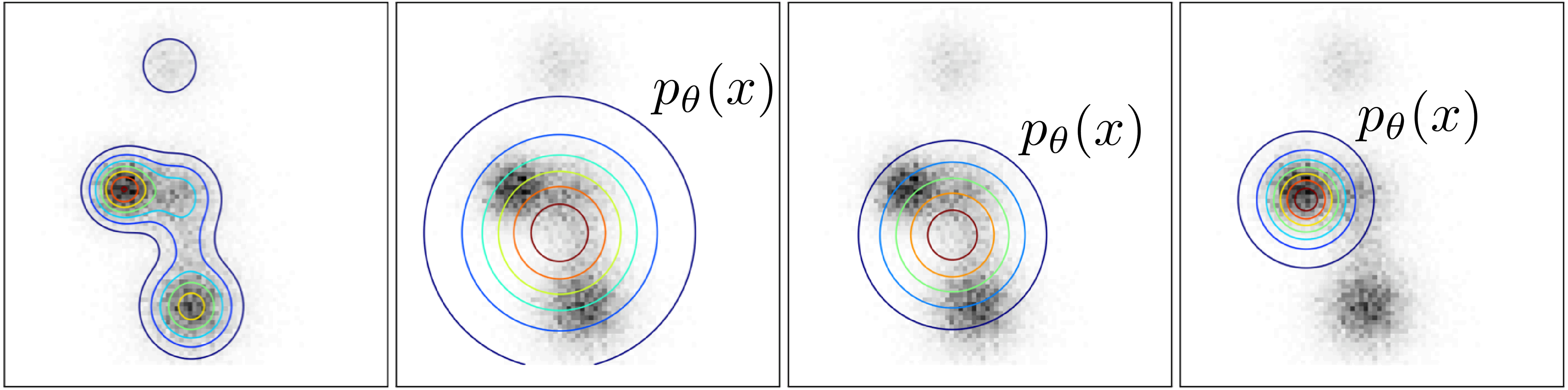


Image Credit: *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, Ferenc Huszár

Why Adversarial?



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

VAEs:
Maximize likelihood of **data samples** in $p_\theta(x)$

GANs:
Adversarial game

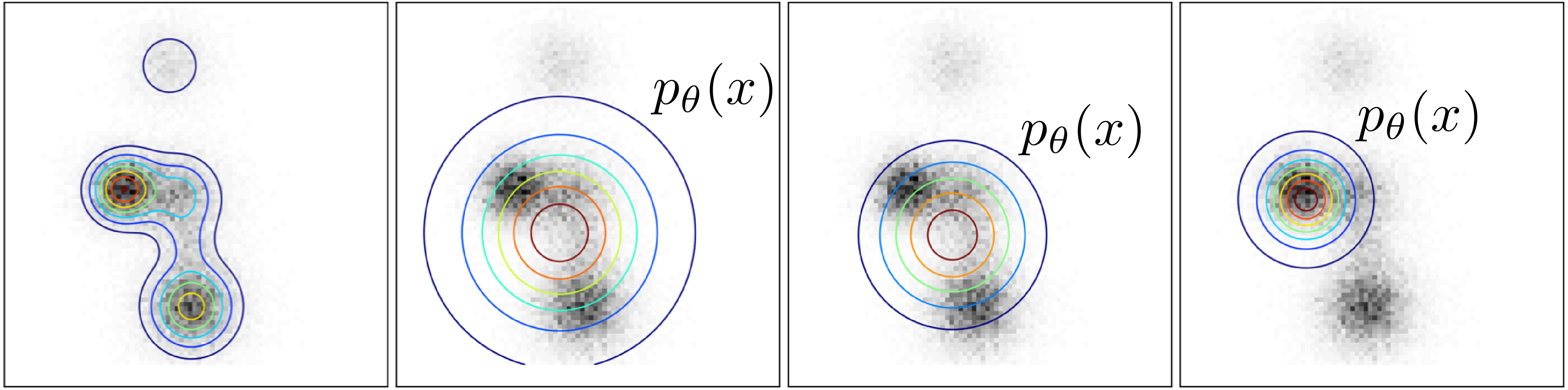
Maximize likelihood of **generator samples** in approximate $p_{\text{data}}(x)$

SIGGRAPH ASIA 2018 TOKYO

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

Why Adversarial?

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



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

VAEs:
Maximize likelihood of **data samples** in $p_\theta(x)$

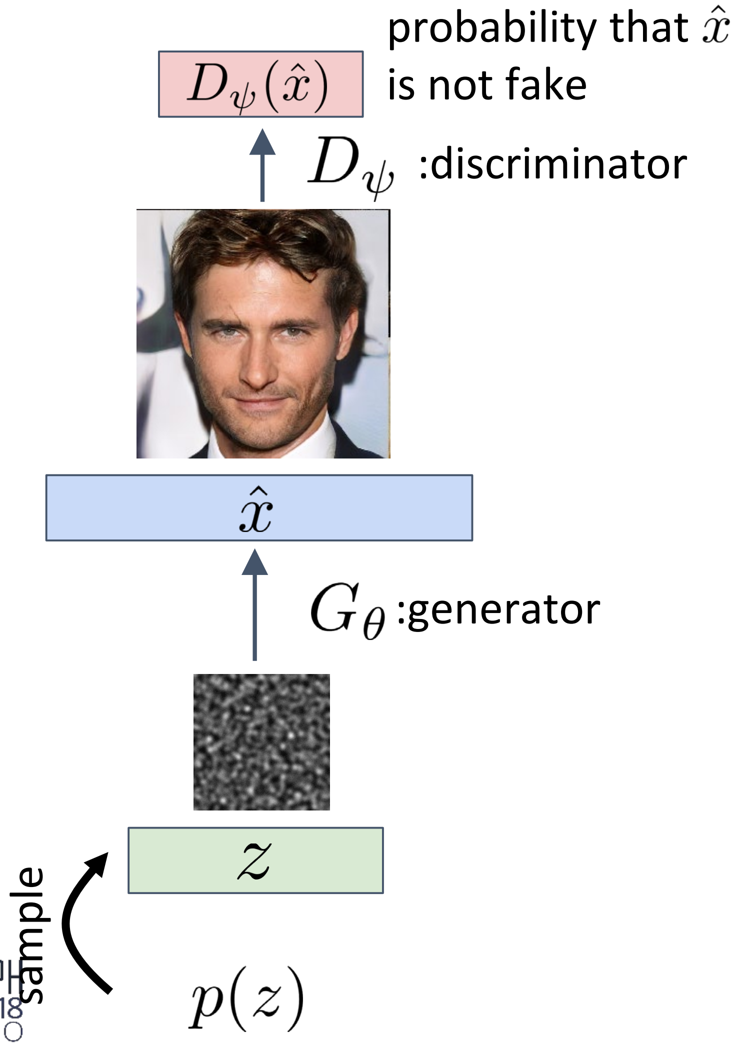
GANs:
Adversarial game

Maximize likelihood of **generator samples** in approximate $p_{\text{data}}(x)$

SIGGRAPH ASIA 2018 TOKYO

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

GAN Objective



fake/real classification loss (BCE):

$$L(\theta, \psi) = -0.5\, \mathbb{E}_{x \sim p_{\text{data}}} \log D_\psi(x) - 0.5\, \mathbb{E}_{x \sim p_\theta} \log(1 - D_\psi(x))$$

Discriminator objective:

$$\min_\psi L(\theta, \psi)$$

Generator objective:

$$\max_\theta L(\theta, \psi)$$

Non-saturating Heuristic

$$L(\theta, \psi) = -0.5\, \mathbb{E}_{x \sim p_{\text{data}}} \log D_\psi(x) - 0.5\, \mathbb{E}_{x \sim p_\theta} \log(1 - D_\psi(x))$$

Generator loss is negative binary cross-entropy:

$$L_G(\theta, \psi) = 0.5\, \mathbb{E}_{x \sim p_\theta} \log(1 - D_\psi(x))$$ poor convergence

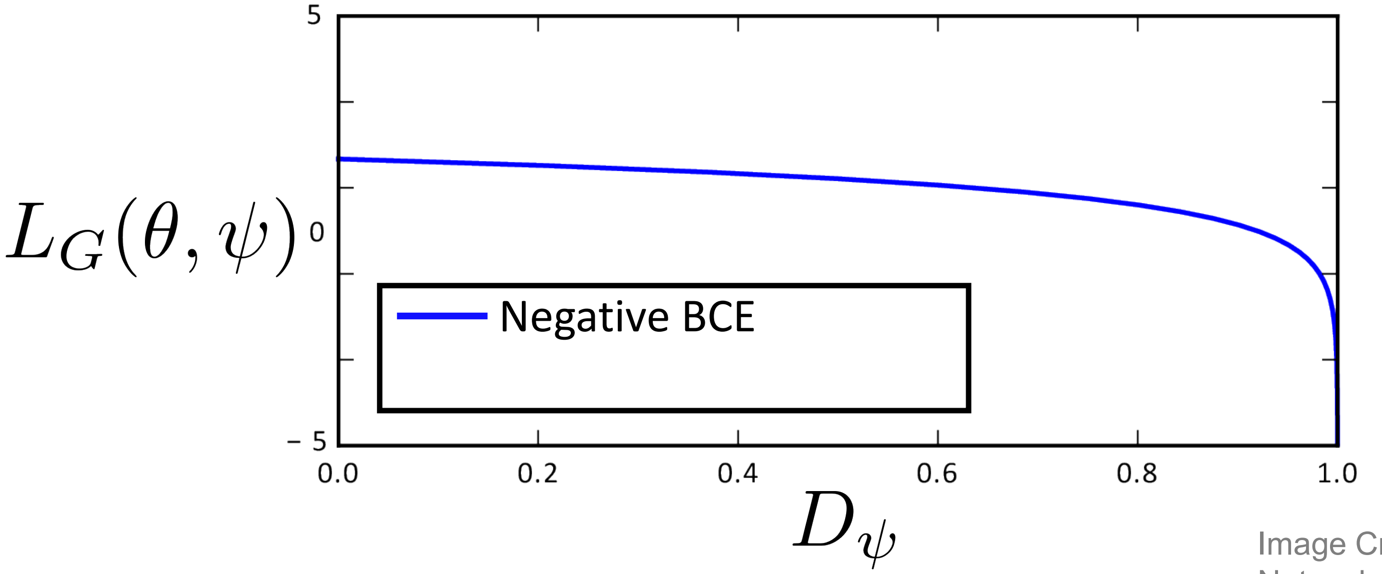


Image Credit: NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow

SIGGRAPH ASIA 2018 TOKYO

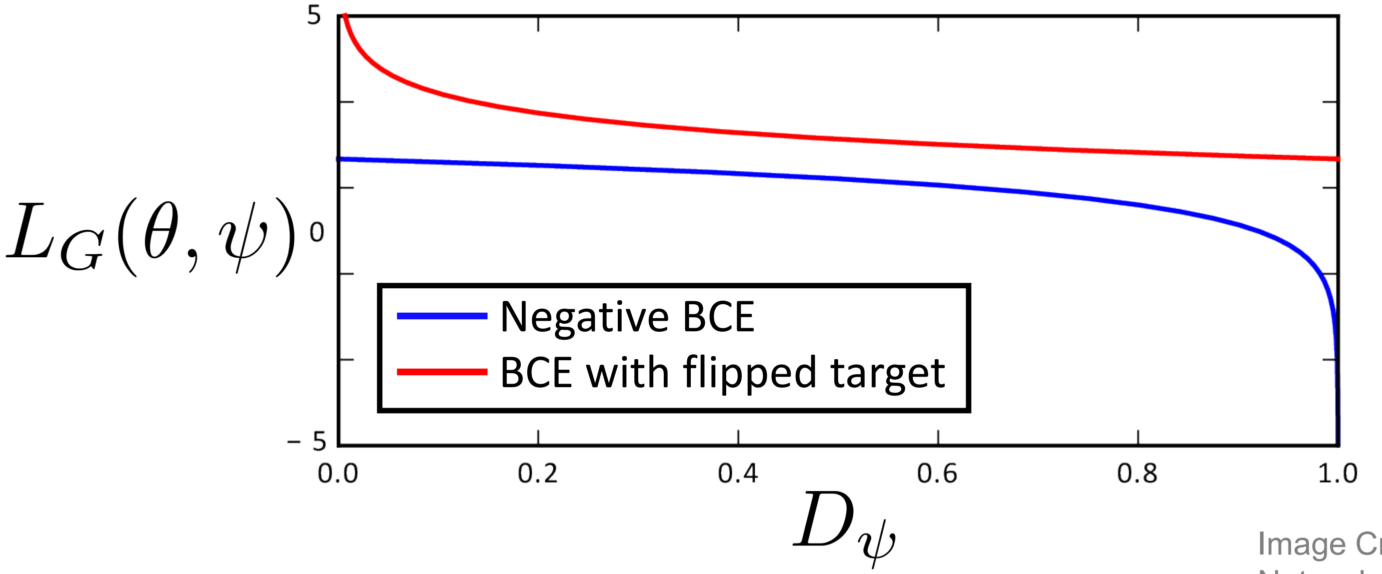
Non-saturating Heuristic

Generator loss is negative binary cross-entropy:

$$L_G(\theta, \psi) = 0.5\ \mathbb{E}_{x \sim p_\theta}\ \log(1 - D_\psi(x))$$ poor convergence

Flip target class instead of flipping the sign for generator loss:

$$L_G(\theta, \psi) = -0.5\ \mathbb{E}_{x \sim p_\theta}\ \log D_\psi(x)$$ good convergence – like BCE



SIGGRAPH ASIA 2018 TOKYO

Image Credit: NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow

GAN Training

Generator training

Loss:

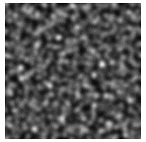
$$L_G(\theta, \psi) = -\log D_\psi(\hat{x})$$

$D_\psi(\hat{x})$

D_ψ :discriminator

$\hat{x}$

G_θ :generator



z

sample

$p(z)$

Discriminator training

Loss:

$$L_D(\theta, \psi) = -0.5 \log(1 - D_\psi(\hat{x})) - 0.5 \log D_\psi(x_i)$$

$D_\psi(\hat{x})$ $D_\psi(x_i)$

D_ψ :discriminator

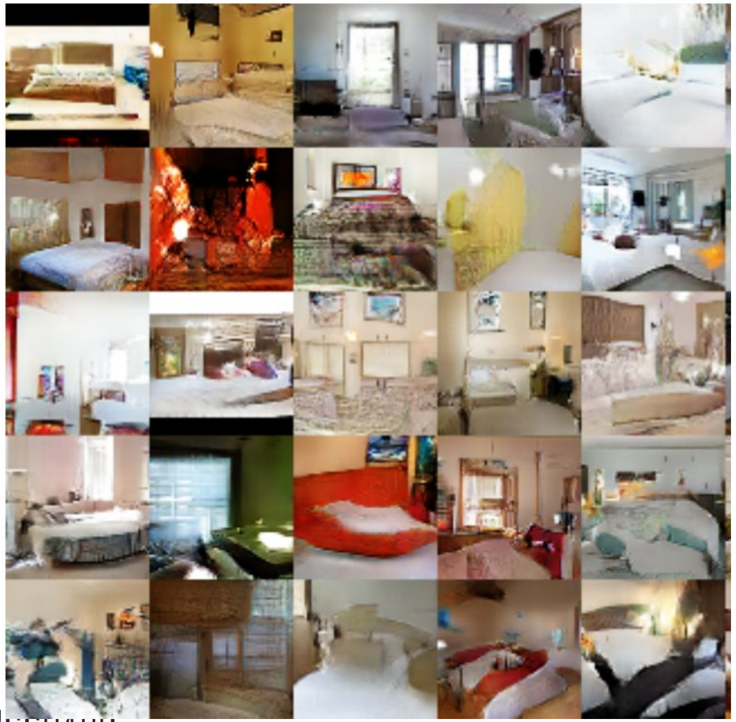
$\hat{x}$ x_i

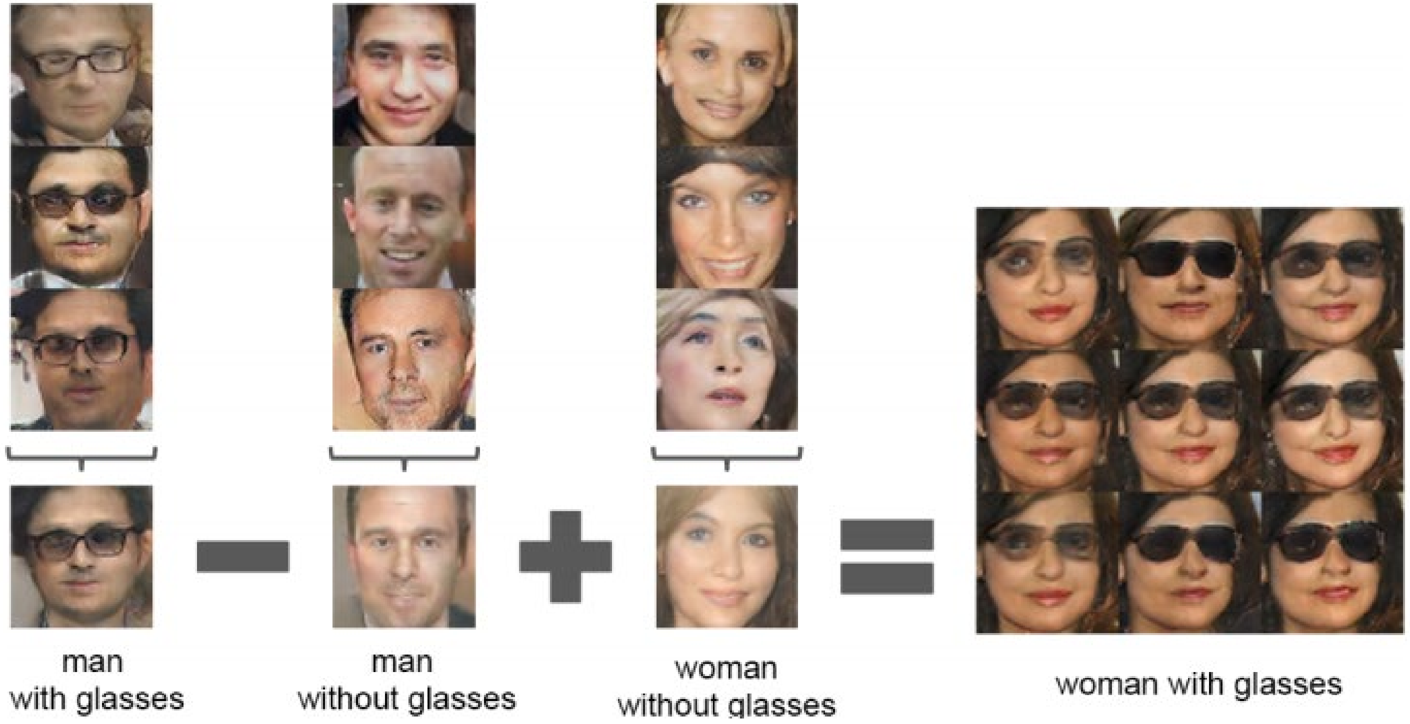
from dataset

Interleave in each training step

DCGAN

- First paper to successfully use CNNs with GANs
- Due to using novel components (at that time) like batch norm., ReLUs, etc.





SIGGRAPH ASIA 2018 TOKYO

Image Credit: *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, Radford et al.

Code example

Generative Adversarial Network

geometry.cs.ucl.ac.uk/creativeai

Conditional GANs (CGANs)

- ≈ learn a mapping between images from example pairs
- Approximate sampling from a conditional distribution $p_{\mathrm{data}}(x \mid c)$

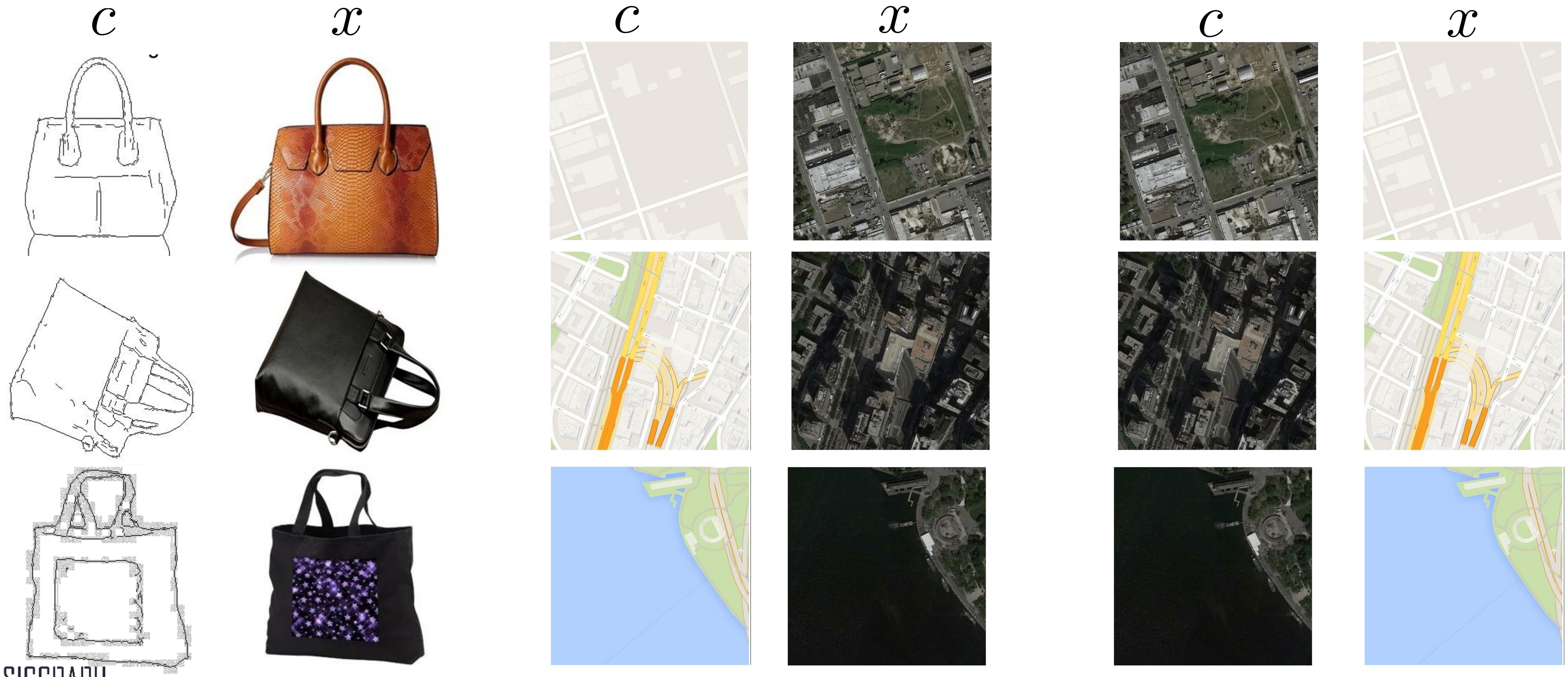


Image Credit: *Image-to-Image Translation with Conditional Adversarial Nets*, Isola et al.

Conditional GANs

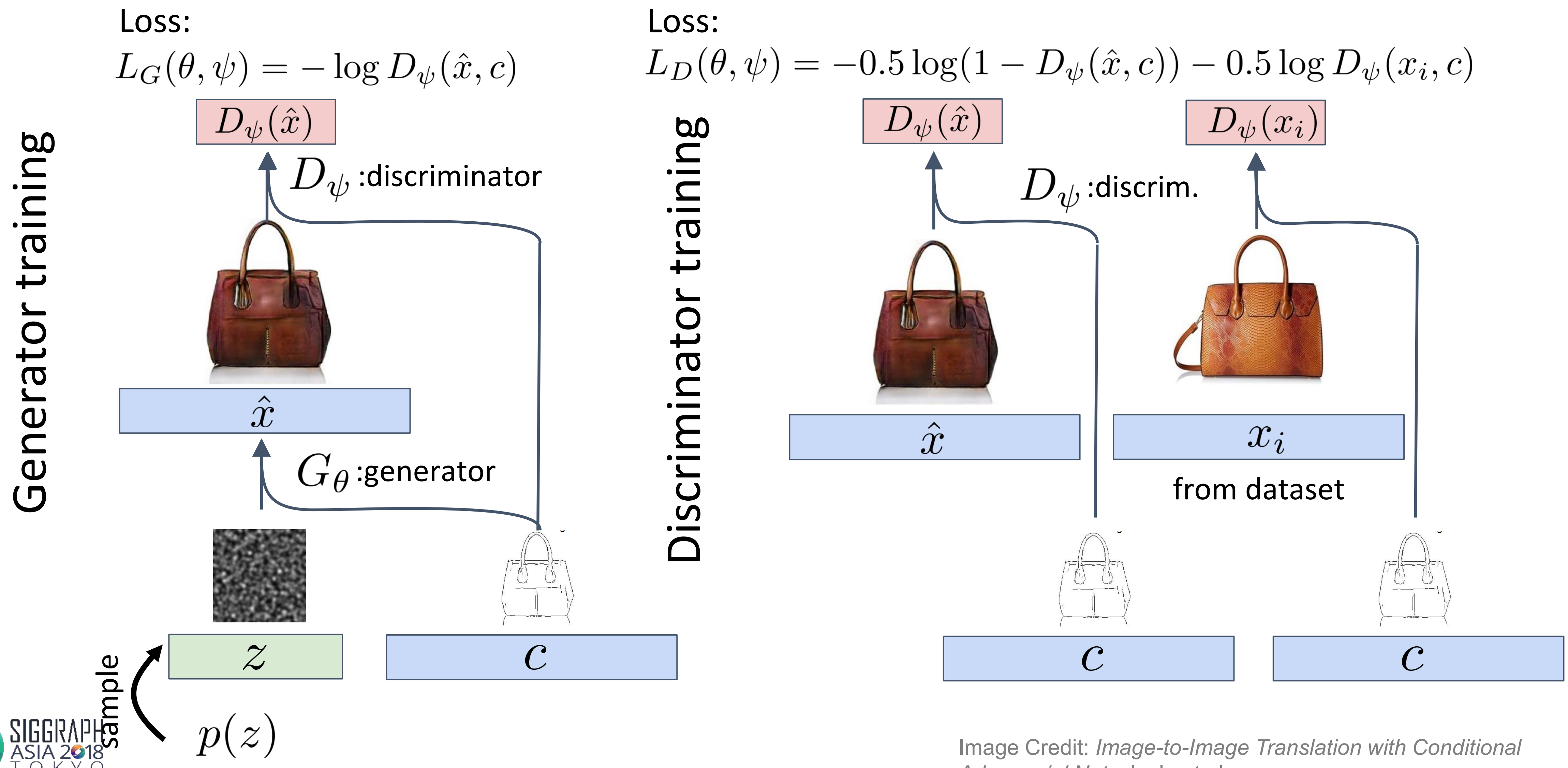


Image Credit: *Image-to-Image Translation with Conditional Adversarial Nets*, Isola et al.

Conditional GANs: Low Variation per Condition

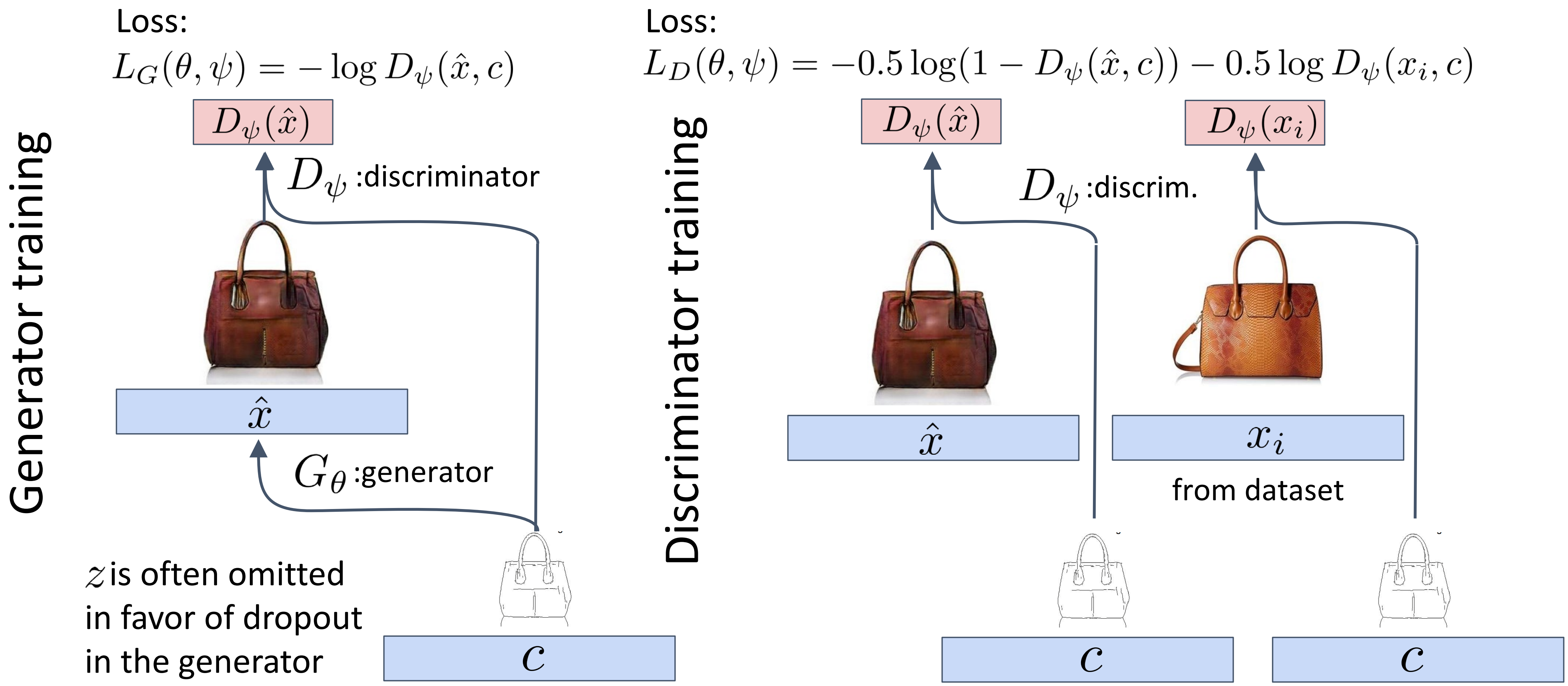


Image Credit: *Image-to-Image Translation with Conditional Adversarial Nets*, Isola et al.

Demos

CGAN

https://affinelayer.com/pixsrv/index.html

Unstable Training

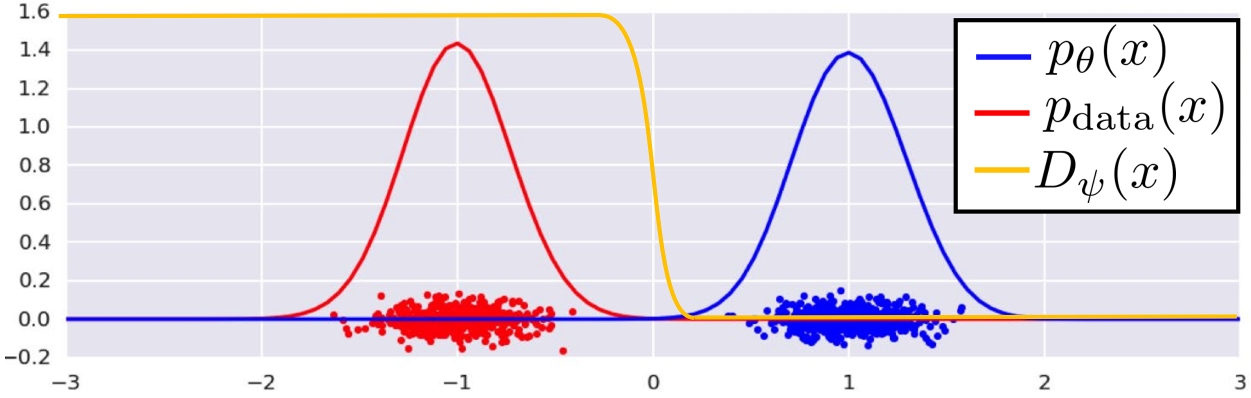
GAN training can be unstable

Three current research problems (may be related):

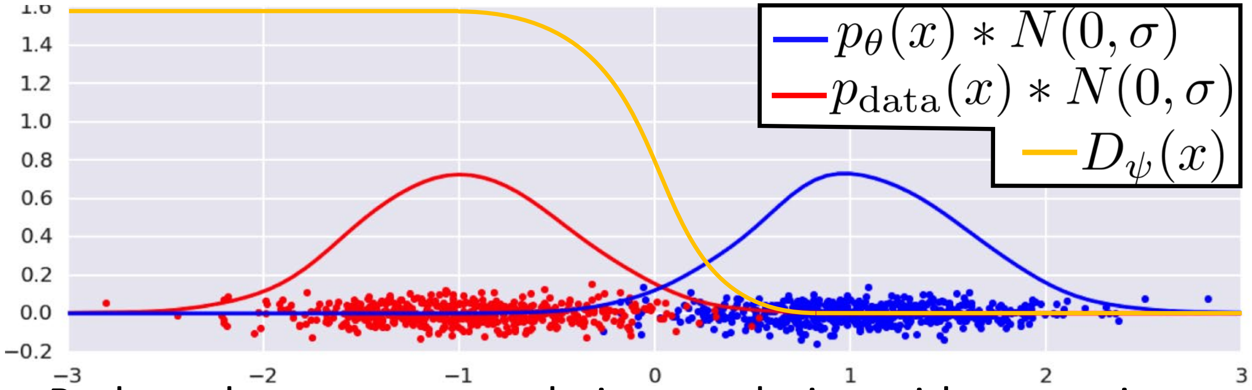
- Reaching a Nash equilibrium (the gradient for both L_G and L_D is 0)
- p_θ and p_{data} initially don't overlap
- Mode Collapse

Generator and Data Distribution Don’t Overlap

Standard



Instance noise: adding noise to generated and real images



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

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

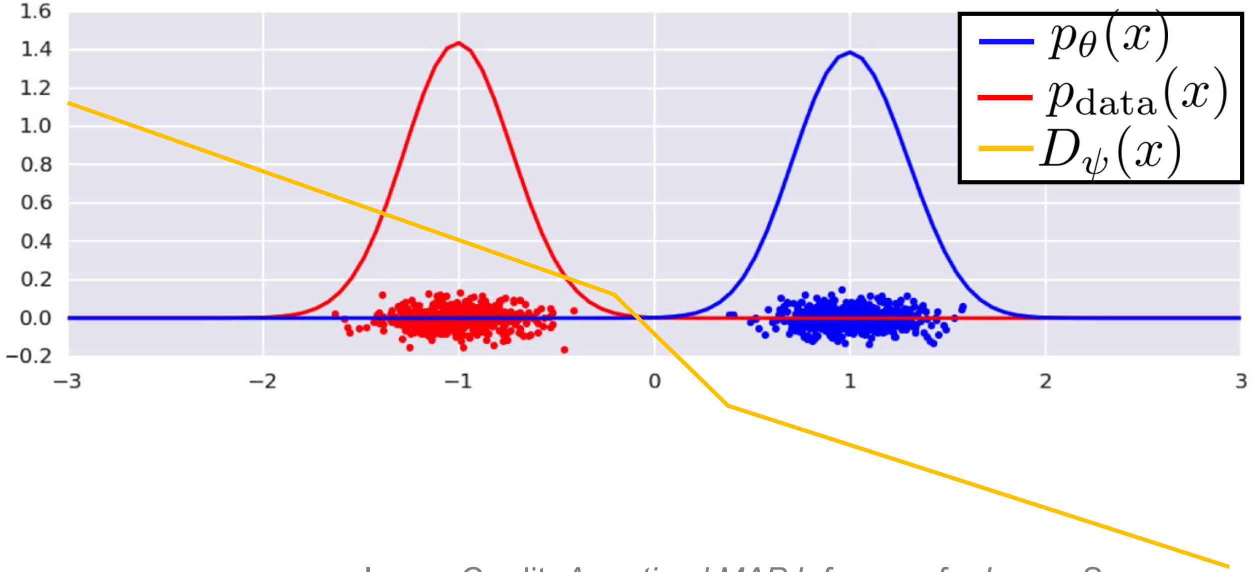


Image Credit: *Amortised MAP Inference for Image Super-resolution*, Sønderby et al.

Mode Collapse

Optimal $D_\psi(x)$: $\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$

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

0 5000 10000 15000 20000 25000 30000 35000 40000 45000 50000

p_{data} p_θ after n training steps

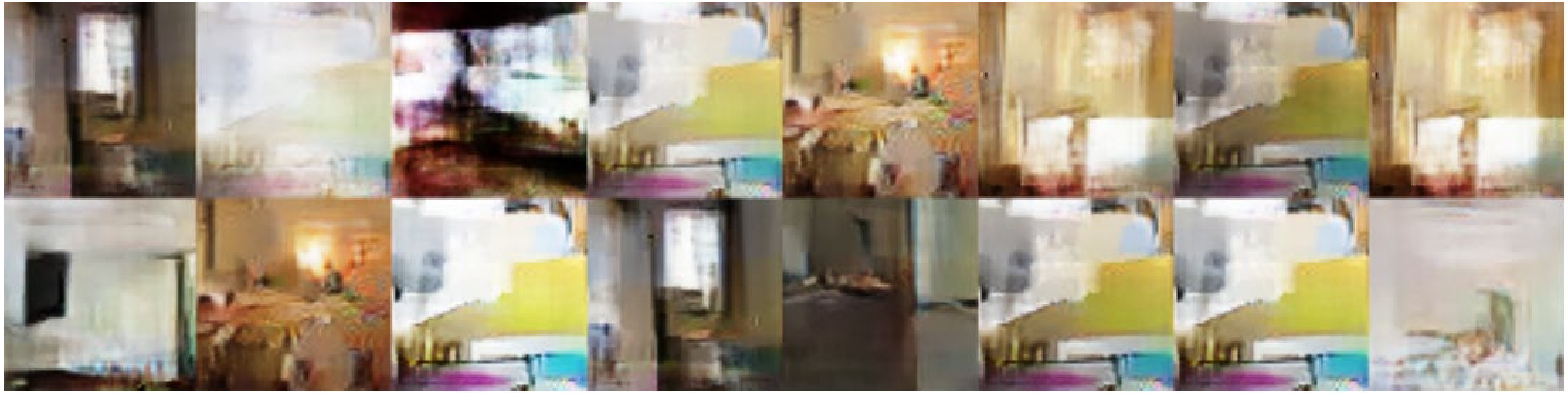


Image Credit: *Wasserstein GAN,* Arjovsky et al.
Unrolled Generative Adversarial Networks, Metz et al.

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

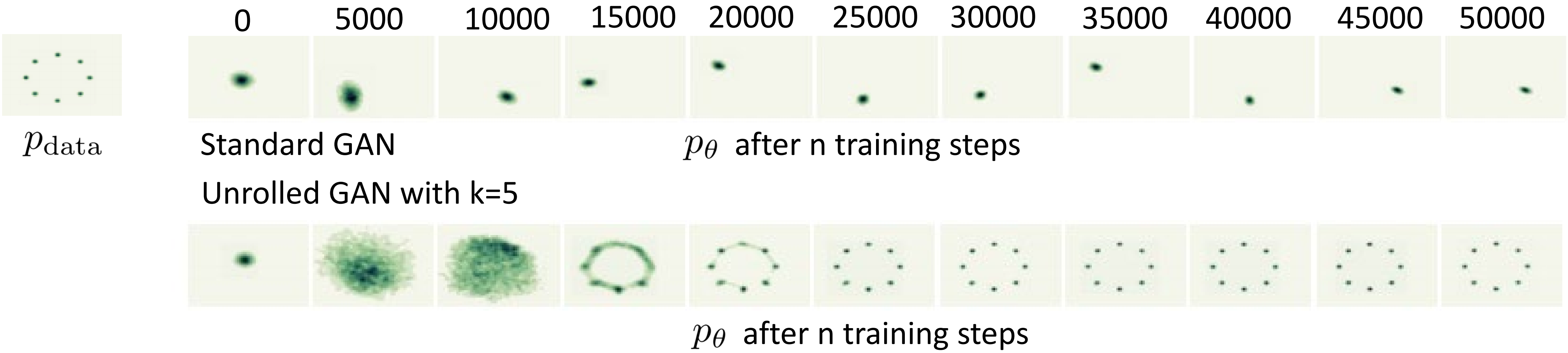


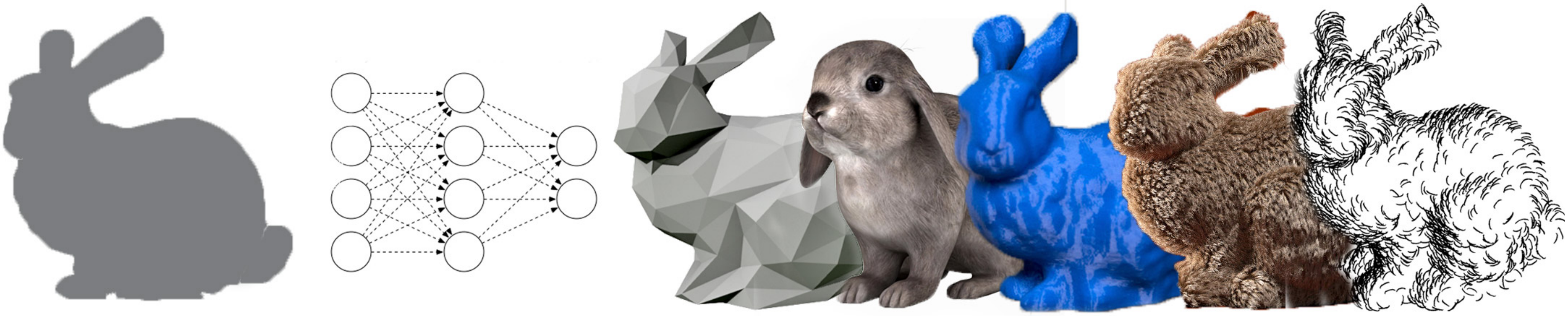
Image Credit: *Wasserstein GAN,* Arjovsky et al.
Unrolled Generative Adversarial Networks, Metz et al.

SIGGRAPH ASIA 2018 TOKYO

Summary

- Autoencoders
  - Can infer useful latent representation for a dataset
  - Bad generators
- VAEs
  - Can infer a useful latent representation for a dataset
  - Better generators due to latent space regularization
  - Lower quality reconstructions and generated samples (usually blurry)
- 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/

SIGGRAPH ASIA 2018 TOKYO