

10707: Deep Learning

Russ Salakhutdinov

Machine Learning Department

rsalakhu@cs.cmu.edu

Generative Adversarial Networks

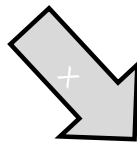
Statistical Generative Models



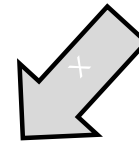
+

Model family, loss function,
optimization algorithm, etc.

Data



Learning



Prior Knowledge

Image x 

A probability
distribution
 $p(x)$

probability $p(x)$

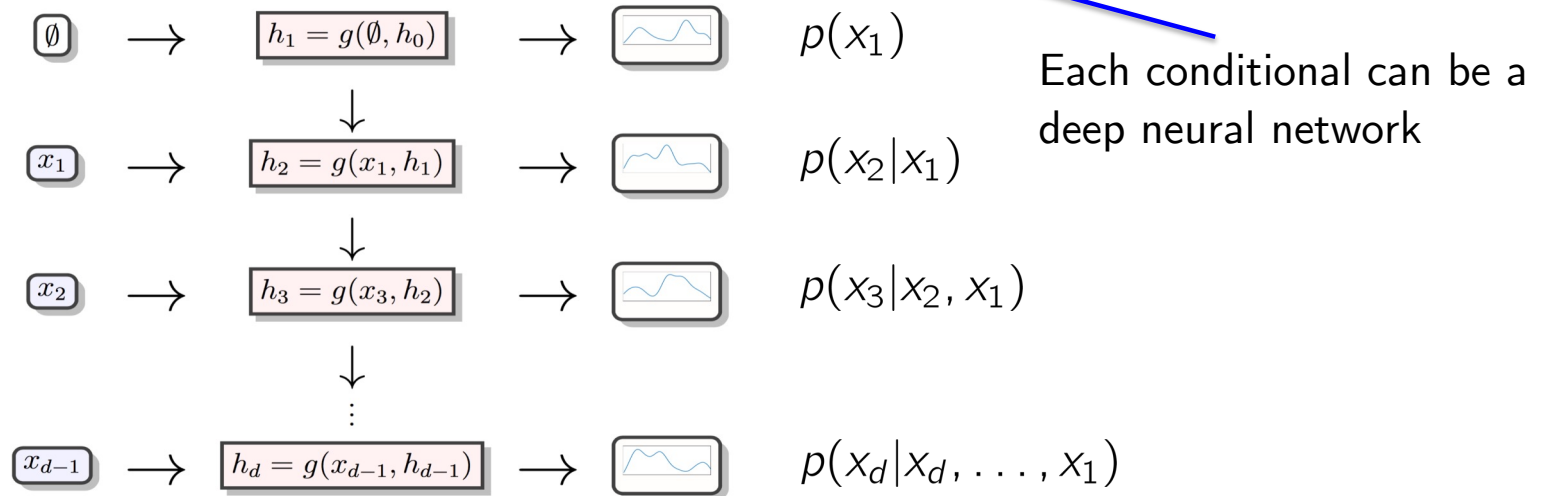
Sampling from $p(x)$ **generates** new images:



Fully Observed Models

► Density Estimation by Autoregression

$$p(x_1, \dots, x_d) = \prod_{i=1}^d p(x_i | x_{i-1}, \dots, x_1) \approx \prod_{i=1}^d p(x_i | g(x_{i-1}, \dots, x_1))$$

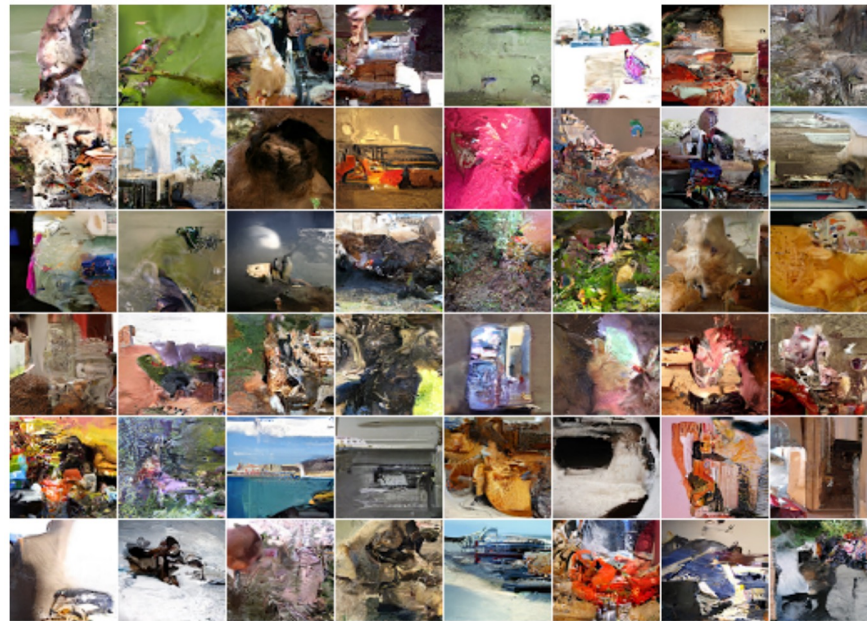


► Ordering of variables is crucial

NADE (Uribe 2013), MADE (Germain 2017), MAF (Papamakarios 2017), PixelCNN (van den Oord, et al, 2016)

Fully Observed Models

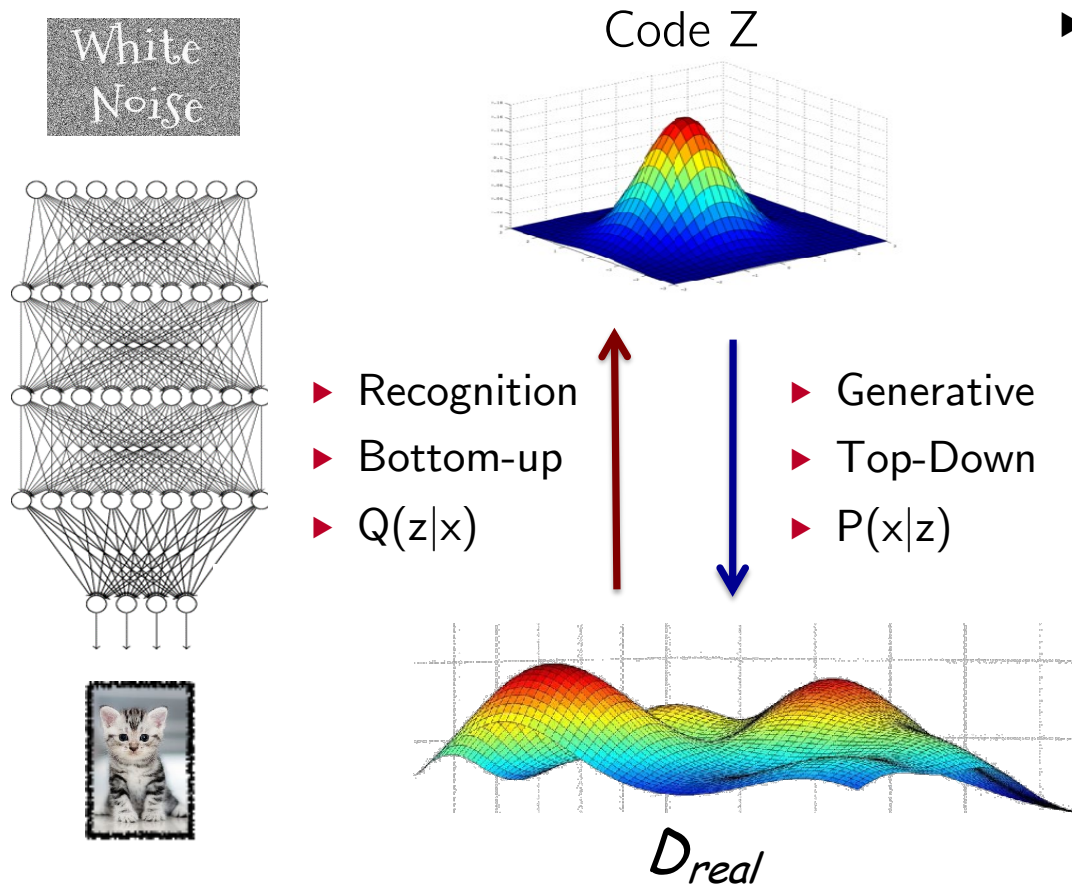
► Density Estimation by Autoregression



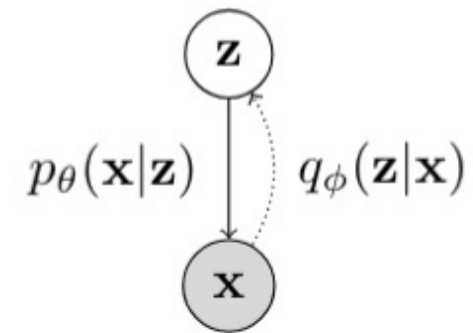
PixelCNN (van den Oord, et al, 2016)

NADE (Uribe 2013), MADE (Germain 2017), MAF (Papamakarios 2017), PixelCNN (van den Oord, et al, 2016)

Deep Directed Generative Models



▶ Latent Variable Models

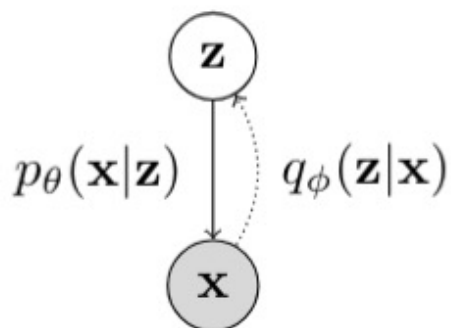


$$\log p_{\theta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}, z) dz$$

- ▶ Conditional distributions are parameterized by deep neural networks

Directed Deep Generative Models

- ▶ Directed Latent Variable Models with Inference Network



- ▶ Maximum log-likelihood objective

$$\max_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \log p_{\theta}(\mathbf{x})$$

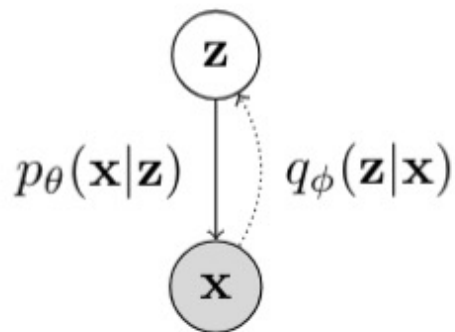
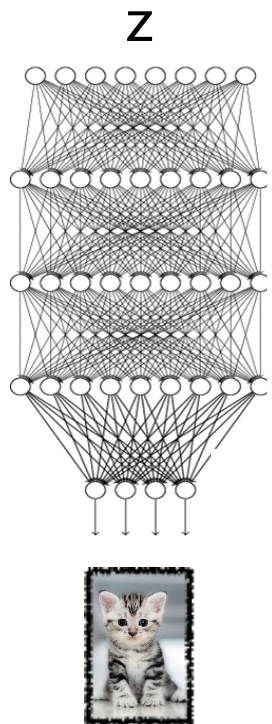
- ▶ Marginal log-likelihood is **intractable**:

$$\log p_{\theta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- ▶ **Key idea**: Approximate true posterior $p(\mathbf{z}|\mathbf{x})$ with a simple, tractable distribution $q(\mathbf{z}|\mathbf{x})$ (inference/recognition network).

Variational Autoencoders (VAEs)

- ▶ Single stochastic (Gaussian) layer, followed by many deterministic layers



$$p(\mathbf{z}) = \mathcal{N}(0, I)$$

$$p_{\theta}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mu(\mathbf{z}, \theta), \Sigma(\mathbf{z}, \theta))$$

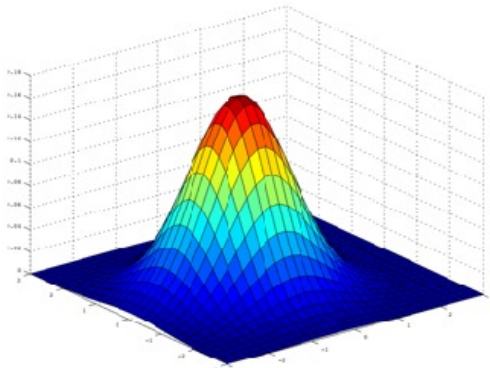
Deep neural network parameterized by θ .
(Can use different noise models)

$$q_{\phi}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}, \phi), \Sigma(\mathbf{x}, \phi))$$

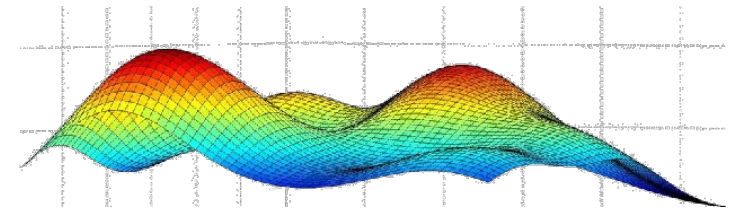
Deep neural network parameterized by ϕ .

Generative Adversarial Networks (GAN)

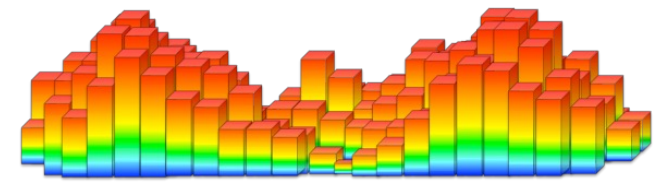
- ▶ Implicit generative model for an unknown target density $p(x)$
- ▶ Converts sample from a known noise density $p_Z(z)$ to the target $p(x)$



Noise density $p_Z(z)$ over space Z



Unknown target density $p(x)$ of data over domain \mathcal{X} , e.g. $\mathbb{R}^{32 \times 32}$



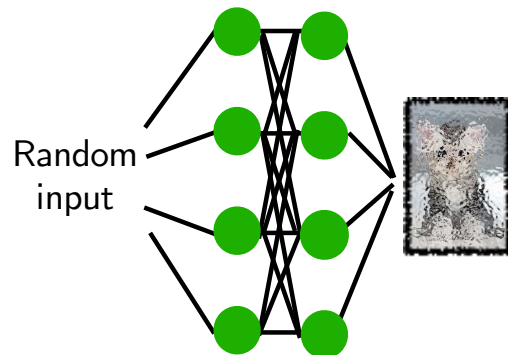
Distribution of generated samples should follow target density $p(x)$

GAN Formulation

- ▶ GAN consists of two components

Generator

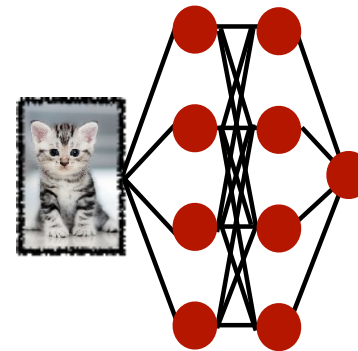
$$G : \mathcal{Z} \rightarrow \mathcal{X}$$



Goal: Produce samples indistinguishable from true data

Discriminator

$$D : \mathcal{X} \rightarrow \mathbb{R}$$



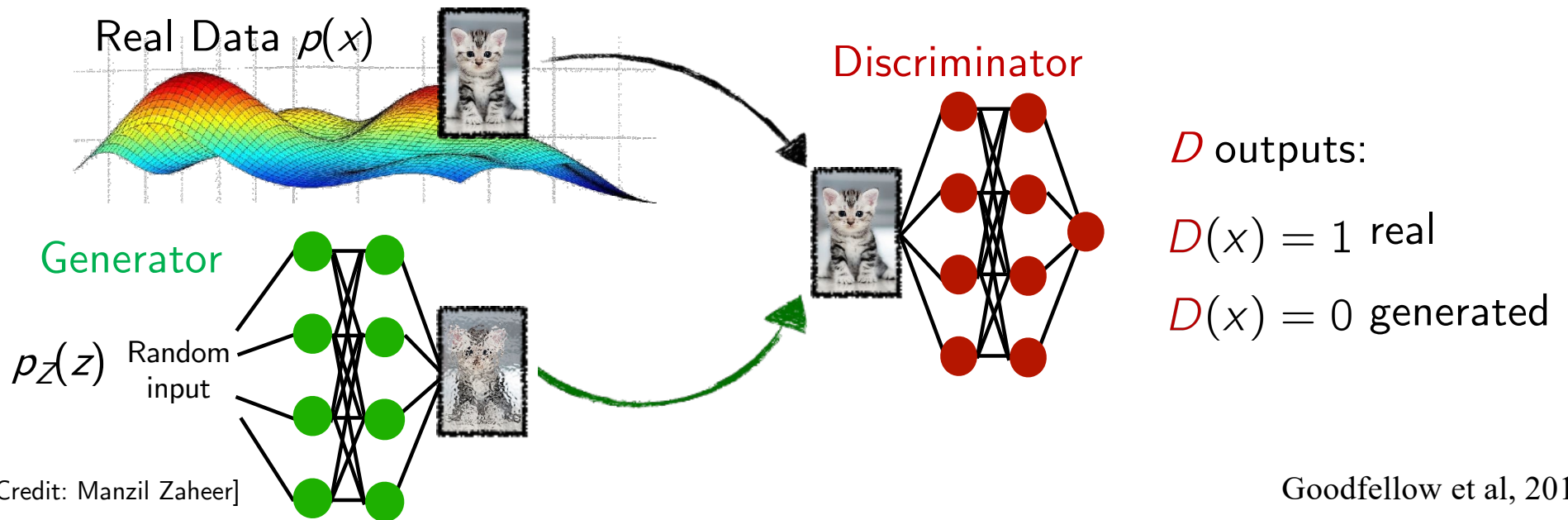
Goal: Distinguish true and generated data apart

Goodfellow et al, 2014

GAN Formulation: Discriminator

- **Discriminator's** objective: Tell real and generated data apart like a classifier

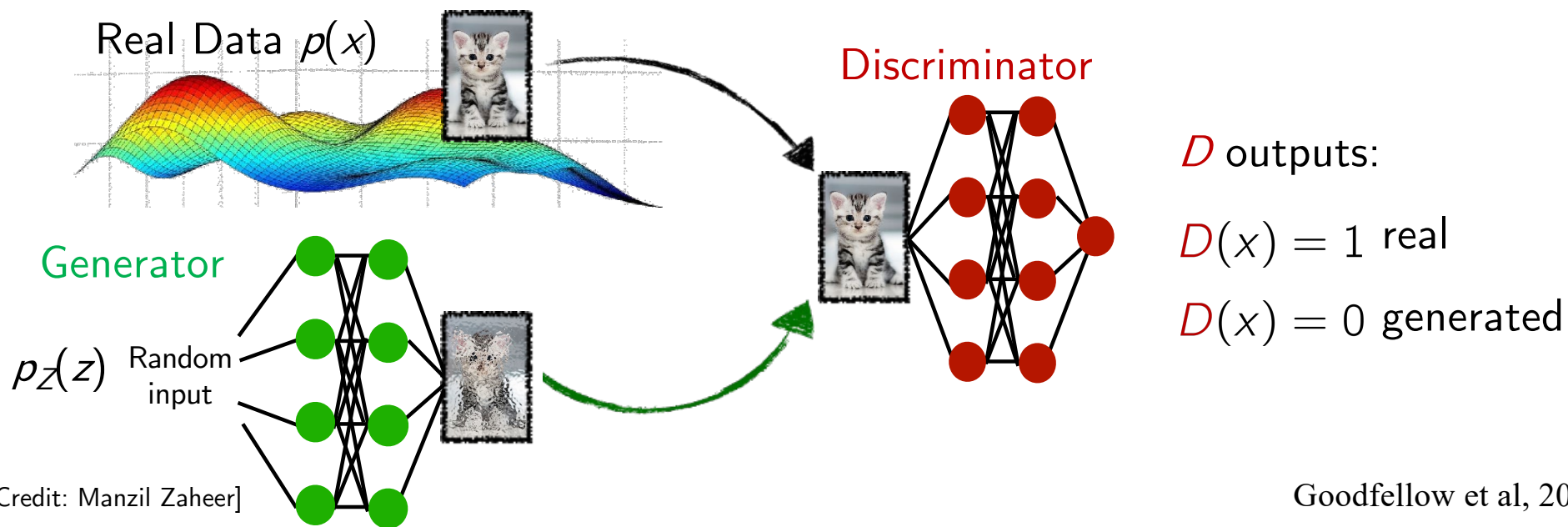
$$\max_D \mathbb{E}_{x \sim p} [\log D(x)] + \mathbb{E}_{z \sim p_Z} [\log (1 - D(G(z)))]$$



GAN Formulation: Generator

- ▶ **Generator's** objective: Fool the best **discriminator**

$$\min_G \max_D \mathbb{E}_{x \sim p} [\log D(x)] + \mathbb{E}_{z \sim p_Z} [\log (1 - D(G(z)))]$$



GAN Formulation: Optimization

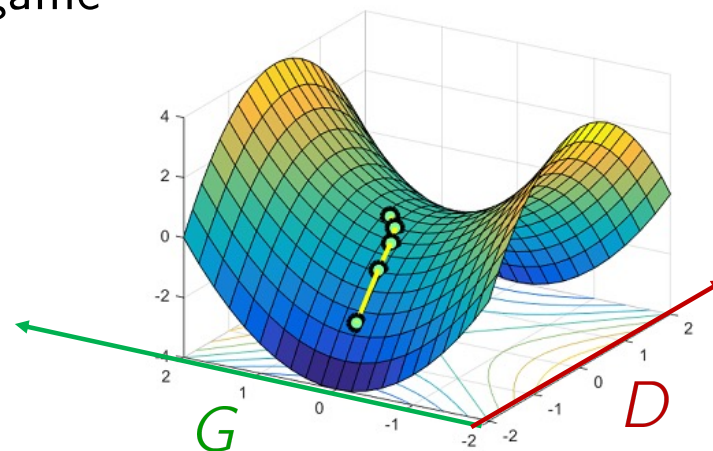
- ▶ Overall GAN optimization

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

- ▶ The generator-discriminator are iteratively updated using SGD to find “equilibrium” of a “min-max objective” like a game

$$G \leftarrow G - \eta_G \nabla_G V(G, D)$$

$$D \leftarrow D - \eta_D \nabla_D V(G, D)$$



Distributional perspective - Discriminator

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

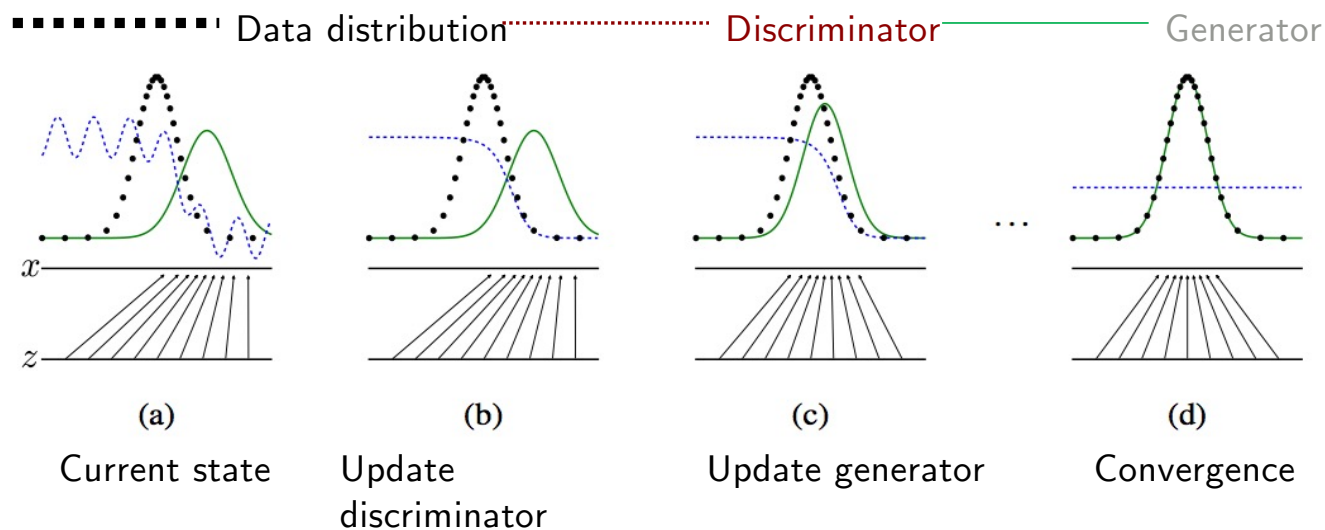
- ▶ For a fixed generator, discriminator is maximizing negative cross entropy
- ▶ Optimal discriminator is given by:

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

A minimax learning objective

- During learning, generator and discriminator are updated alternatively

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$$



Evaluation

- ▶ Likelihoods may not be defined or tractable
- ▶ Directed model permits ancestral sampling
 - ▶ For labelled datasets, metrics such as inception scores quantify sample diversity and quality using pretrained classifiers

Mode Collapse

- ▶ In practice, GANs suffer from mode collapse



Arjovsky et al., 2017

Wasserstein GAN

- ▶ WGAN optimization

$$\min_G \max_D W(G, D) = \mathbb{E}_{x \sim p} [D(x)] - \mathbb{E}_{z \sim p_Z} [D(G(z))]$$

- ▶ Difference in expected output on real vs. generated images
 - ▶ Generator attempts to drive objective ≈ 0
- ▶ More stable optimization

Compare to training DBMs

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}} [\mathbf{v} \mathbf{h}^1{}^\top] - \mathbb{E}_{P_\theta} [\mathbf{v} \mathbf{h}^1{}^\top]$$

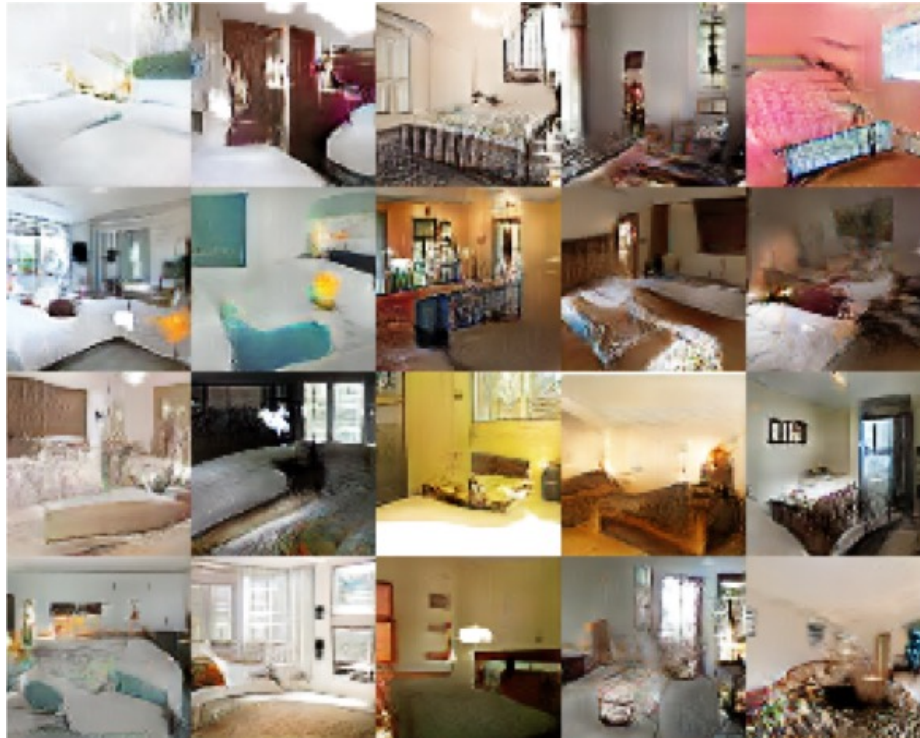
D outputs:

$$D(x) = 1 \text{ real}$$

$$D(x) = 0 \text{ generated}$$

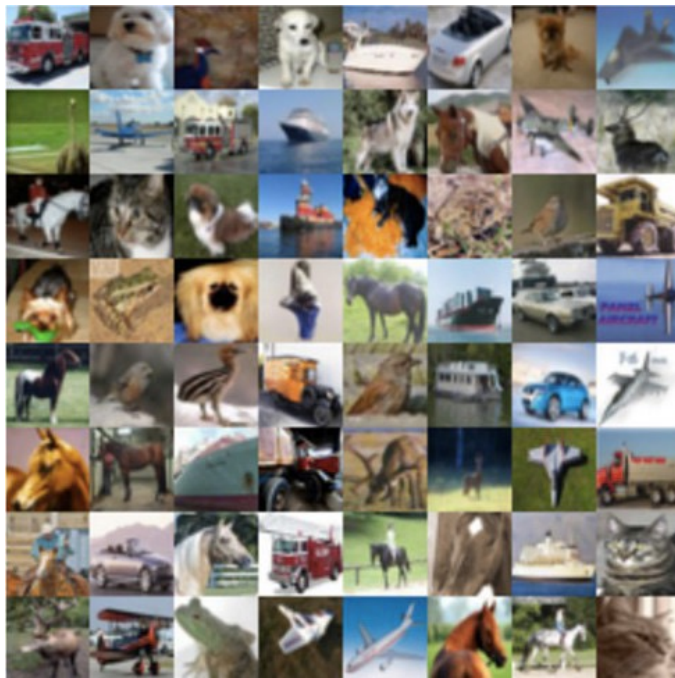
Arjovsky et al., 2017

LSUN Bedroom: Samples

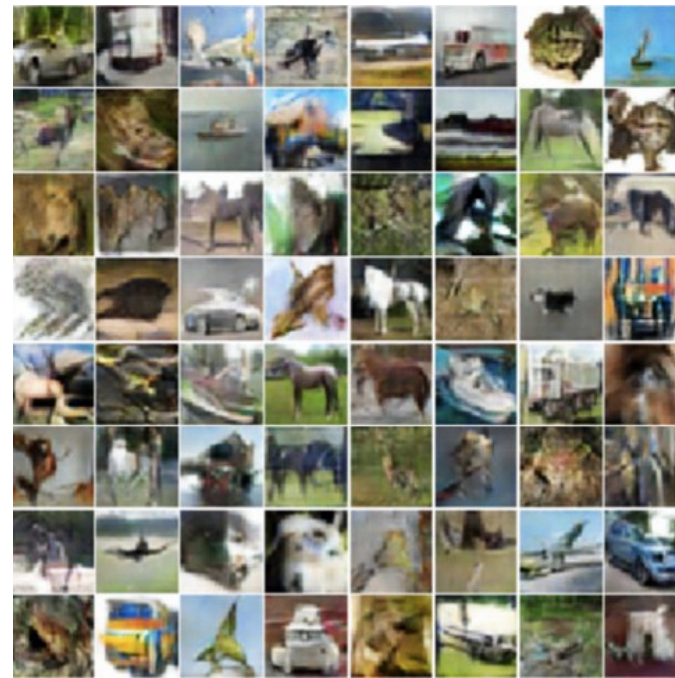


Radford et al., 2015

CIFAR Dataset



Training



Samples

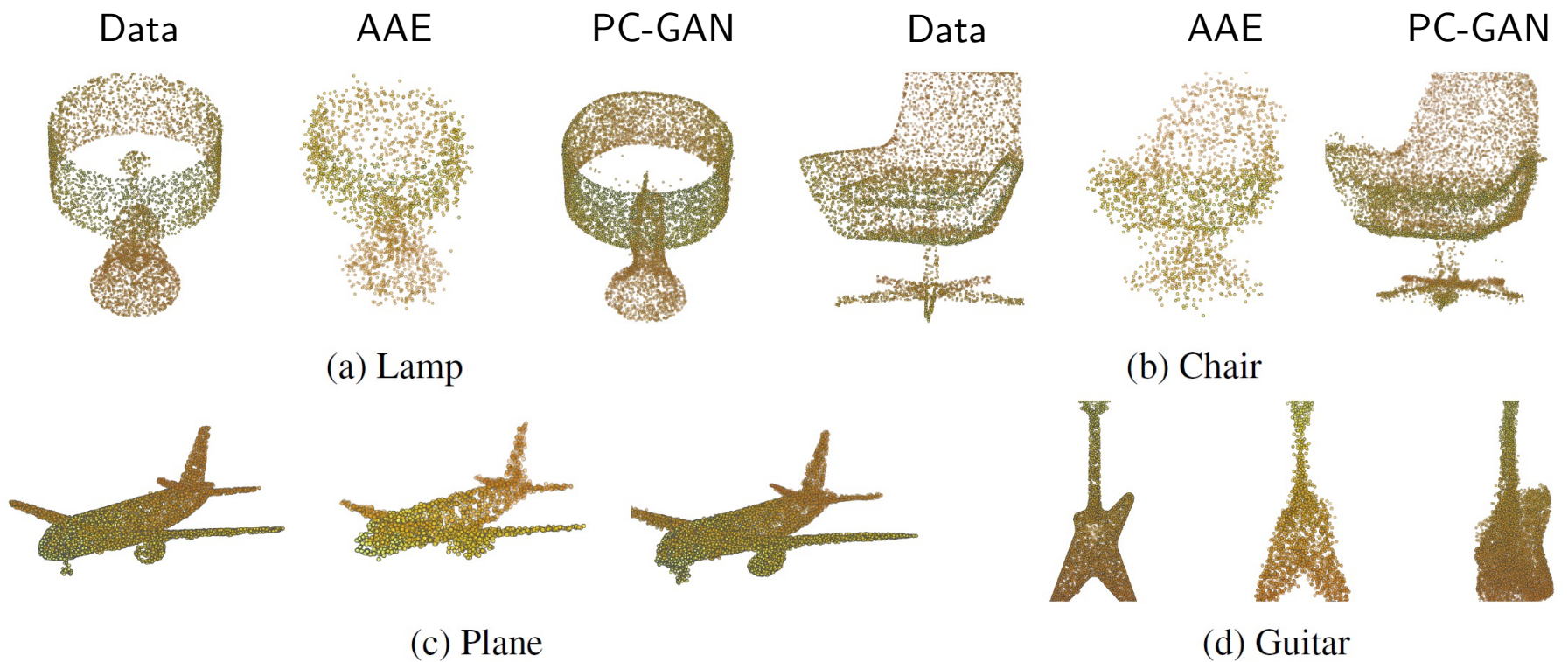
ImageNet: Cherry-Picked Samples



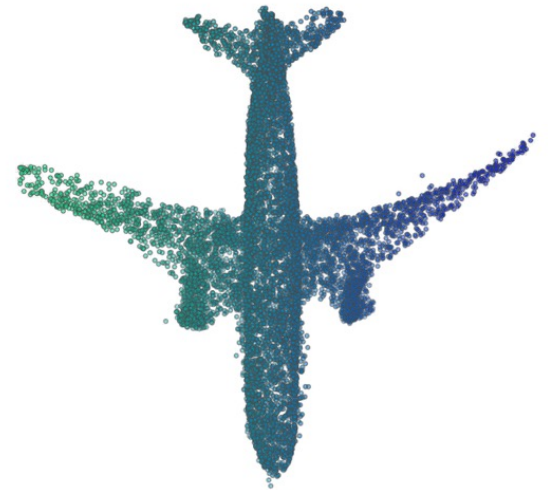
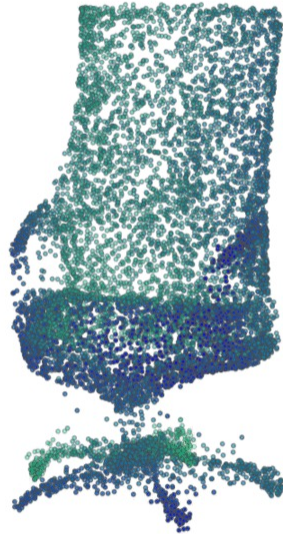
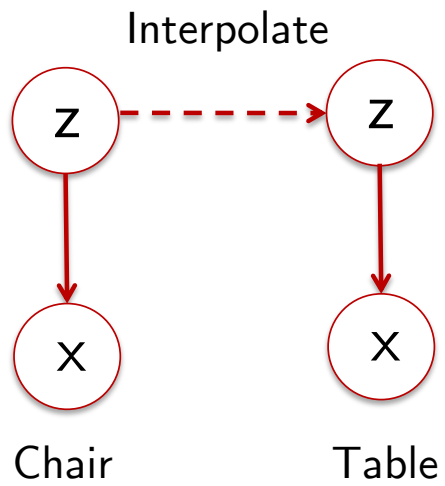
- ▶ **Open Question:** How can we quantitatively evaluate these models!

Slide Credit: Ian Goodfellow

Modelling Point Cloud Data



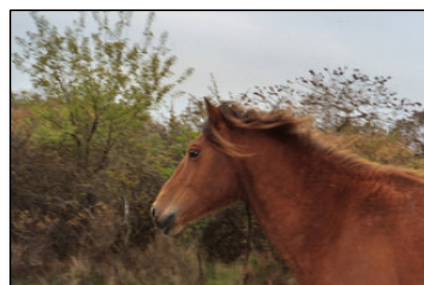
Interpolation in Latent Space



Cycle GAN



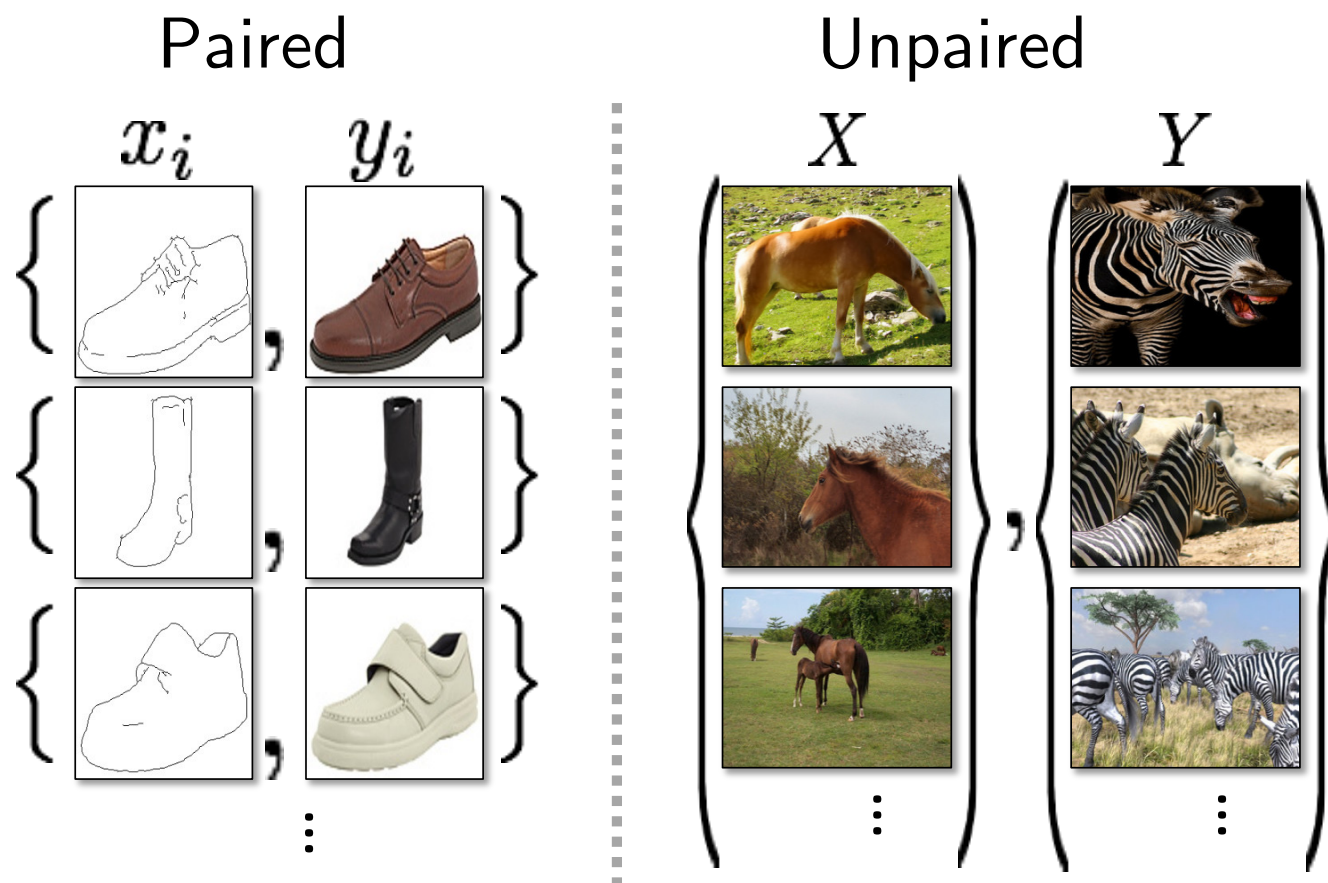
Label photo: per-pixel labeling



Horse zebra: how to get zebras?

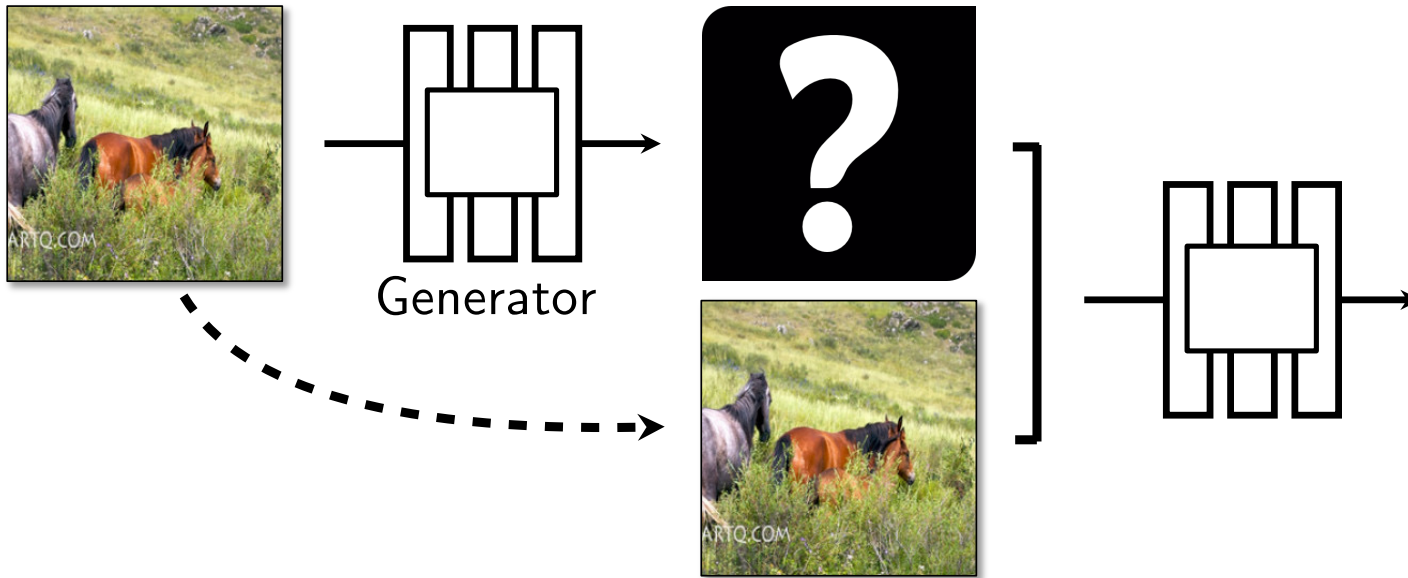
- Expensive to collect pairs.
- Impossible in many scenarios.

Cycle GAN

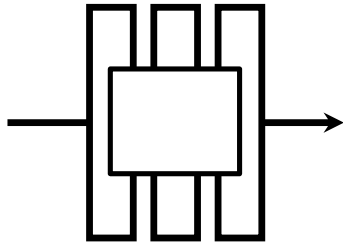


Slide credit: Jun-Yan Zhu

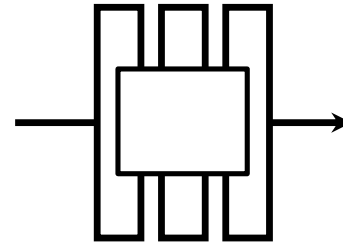
[Zhu*, Park*, Isola, and Efros, ICCV 2017]



No input-output pairs!

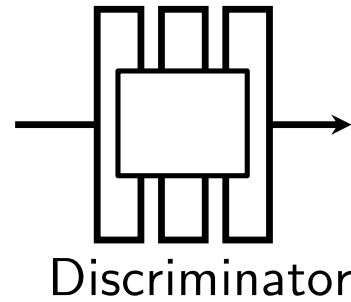
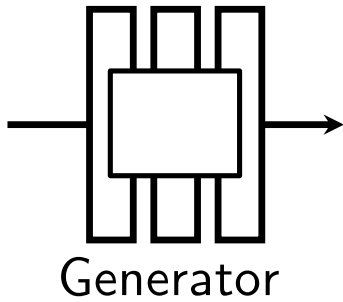


Generator



Discriminator

Real



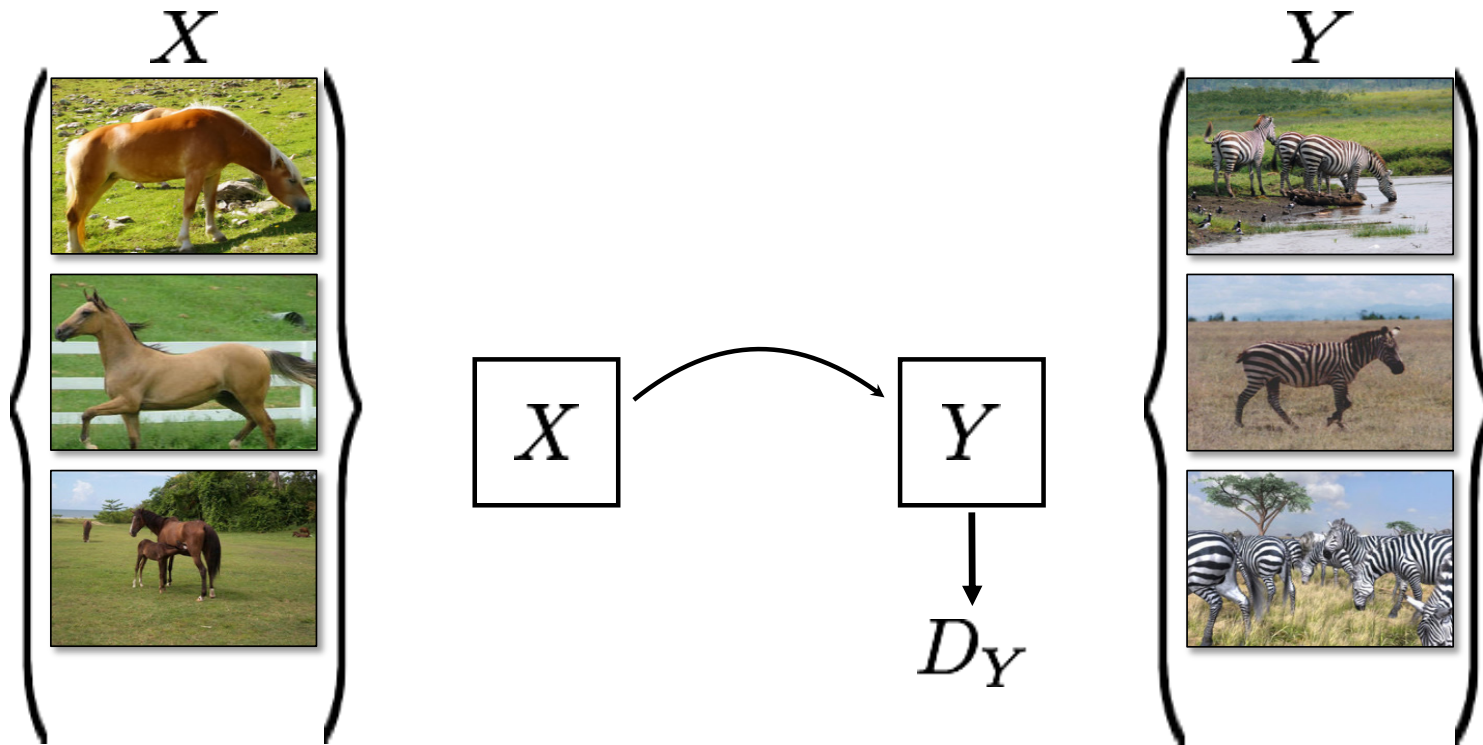
Real too

GANs doesn't force output to correspond to input



mode collapse

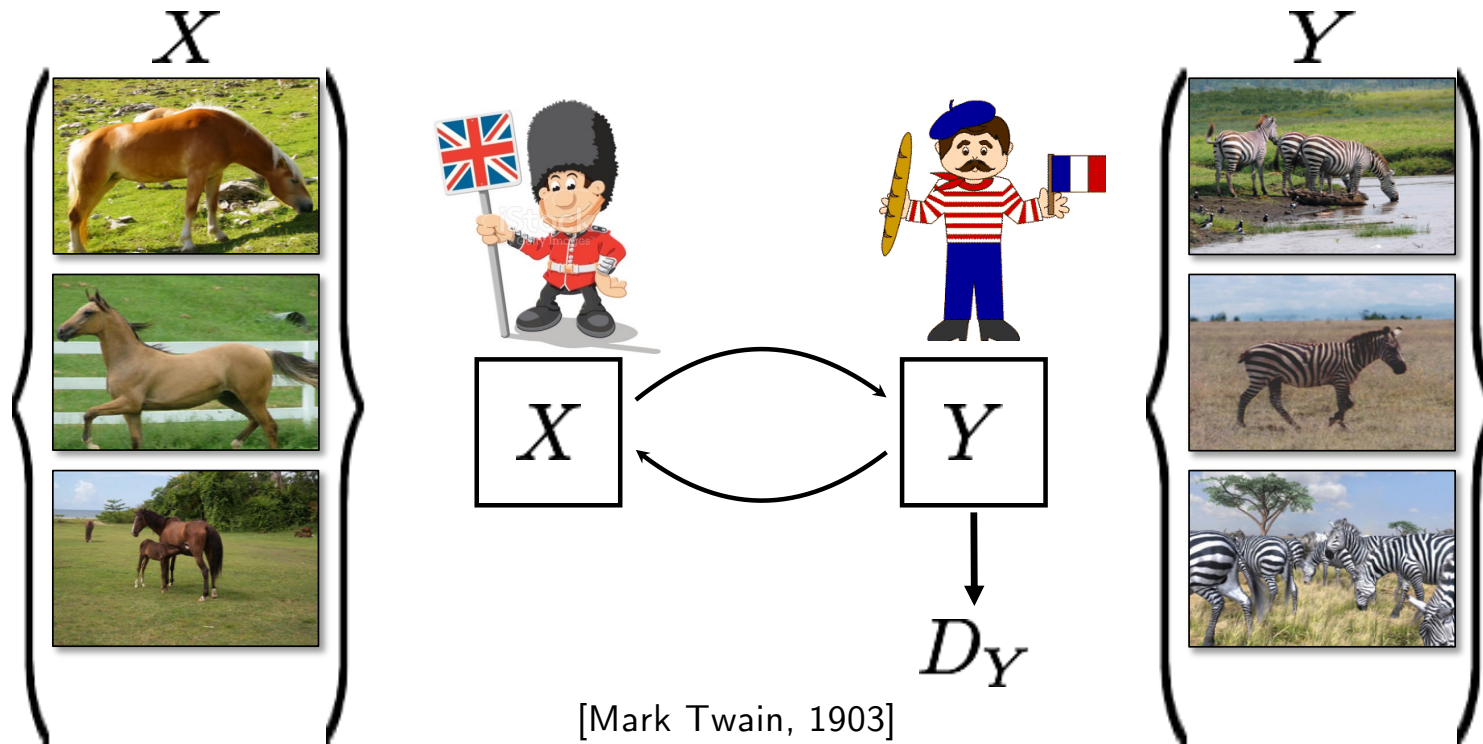
Cycle Consistent Adversarial Networks



Slide credit: Jun-Yan Zhu

[Zhu*, Park*, Isola, and Efros, ICCV 2017]

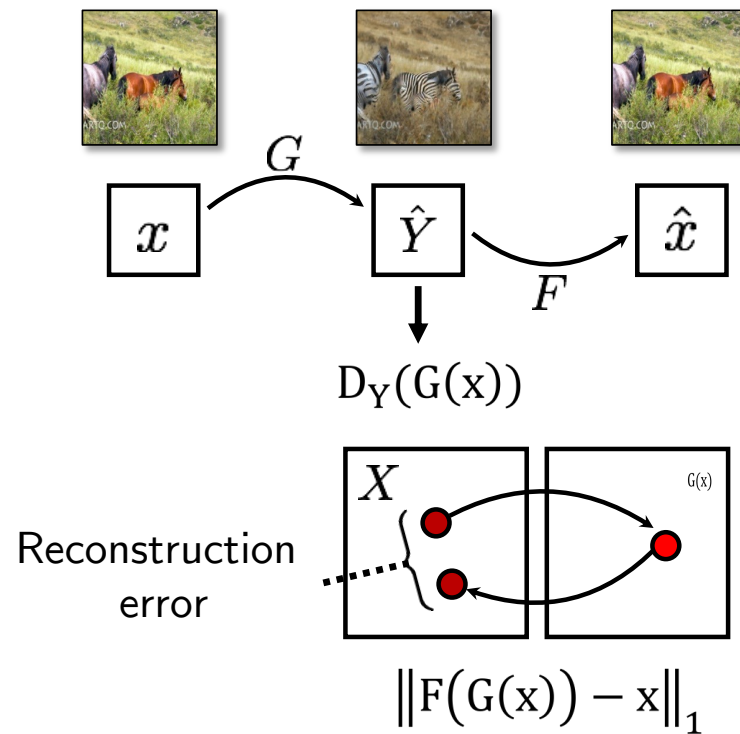
Cycle Consistent Adversarial Networks



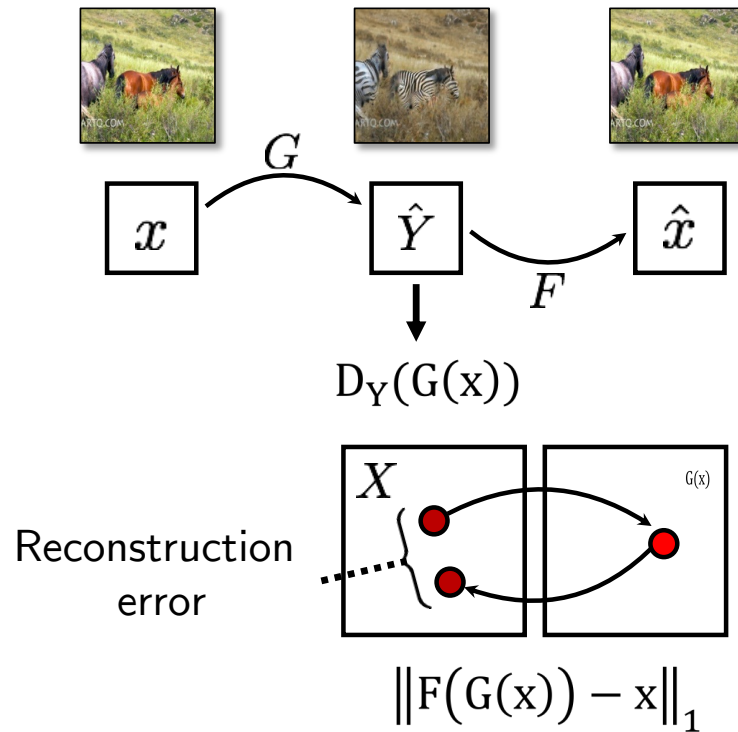
Slide credit: Jun-Yan Zhu

[Zhu*, Park*, Isola, and Efros, ICCV 2017]

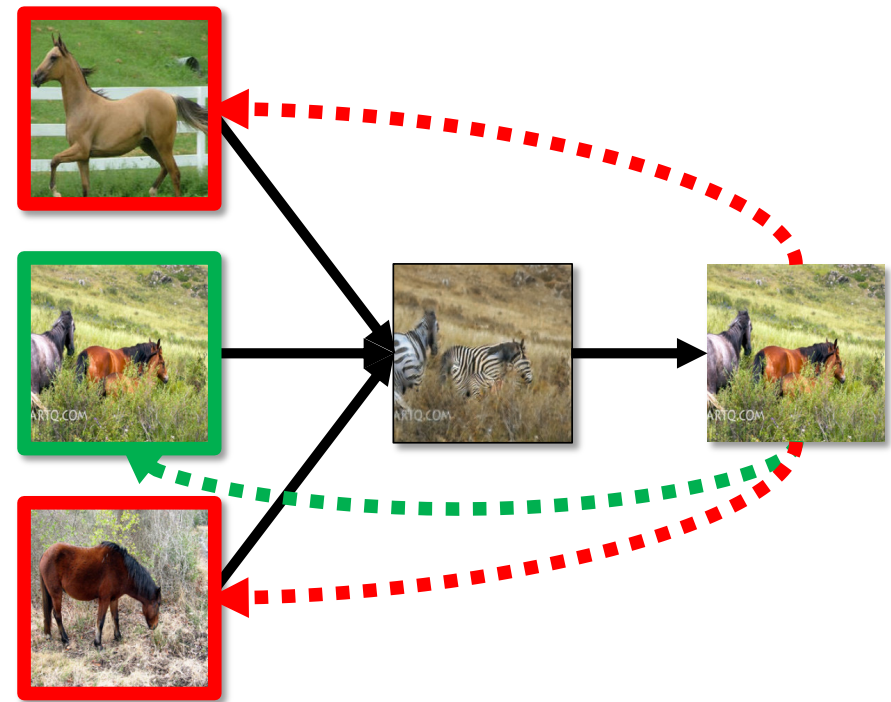
Cycle Consistency Loss



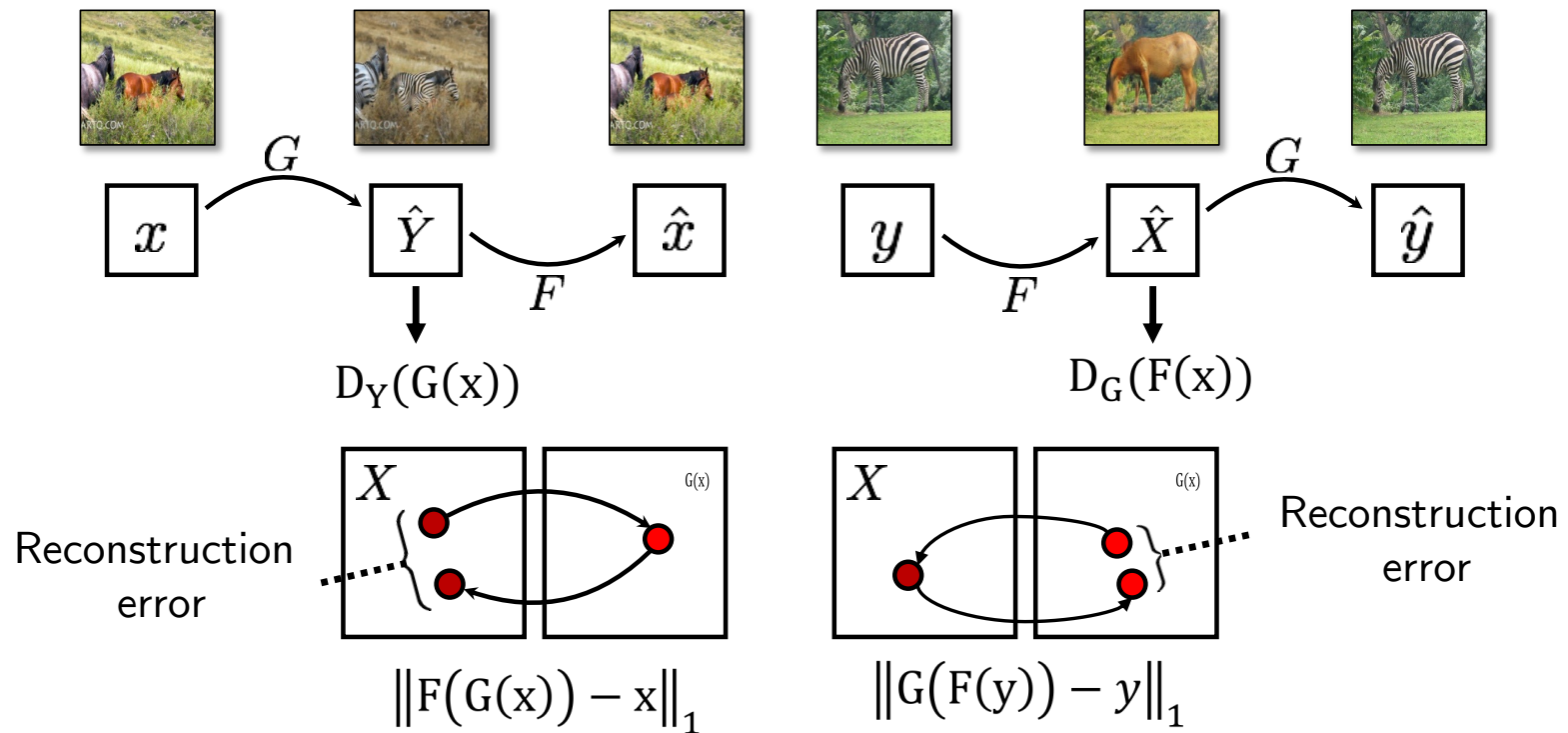
Cycle Consistency Loss



Small cycle loss
Large cycle loss



Cycle Consistency Loss



Collection Style Transfer



Ukiyo-e Cezanne

Van Gogh Monet



Input



Monet



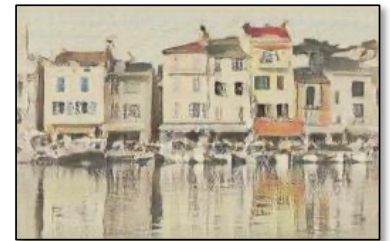
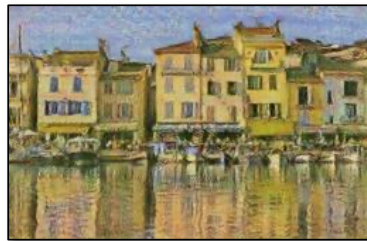
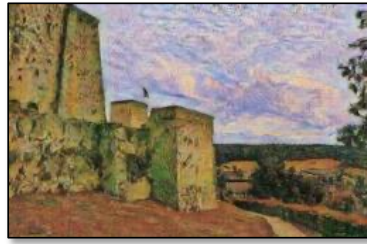
Van Gogh



Cezanne



Ukiyo-e



Conditional Generation

- ▶ Conditional generative model $P(\text{zebra images} | \text{horse images})$



- ▶ Style Transfer



Input Image



Monet

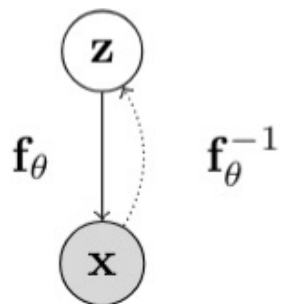


Van Gogh

Zhou et al., Cycle GAN 2017

Normalizing Flows

- ▶ Directed Latent Variable Invertible models



- ▶ The mapping between \mathbf{x} and \mathbf{z} is deterministic and invertible:

$$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$$

$$\mathbf{z} = \mathbf{f}_\theta^{-1}(\mathbf{x})$$

- ▶ Use **change-of-variables** to relate densities between \mathbf{z} and \mathbf{x}

$$p_X(\mathbf{x}; \theta) = p_Z(\mathbf{z}) \left| \det \frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{f}_\theta^{-1}(\mathbf{x})}$$

Grover and Ermon DGM Tutorial, NICE (Dinh et al. 2014),
Real NVP (Dinh et al. 2016)

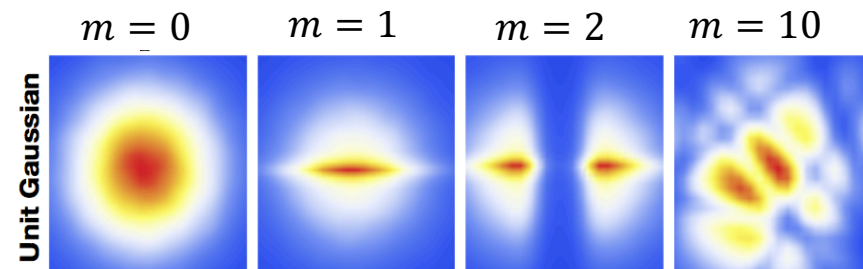
Normalizing Flows

- ▶ Invertible transformations can be composed:

$$\mathbf{x} = \mathbf{f}_\theta^M \circ \dots \circ \mathbf{f}_\theta^1(\mathbf{z}^0); \quad p_X(\mathbf{x}; \theta) = p_{Z^0}(\mathbf{z}^0) \prod_{m=1}^M \left| \det \frac{\partial(\mathbf{f}_\theta^m)^{-1}}{\partial \mathbf{Z}^m} \right|_{\mathbf{Z}^m = \mathbf{z}^m}$$

- ▶ Planar Flows

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}g(\mathbf{w}^\top \mathbf{z} + b)$$



Rezendre and Mohamed, 2016

Normalizing Flows

- ▶ Maximum log-likelihood objective

$$\max_{\theta} \log p_X(\mathcal{D}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \left(\log p_Z(\mathbf{z}) - \log \left| \det \frac{\partial(\mathbf{f}_{\theta})^{-1}}{\partial X} \right|_{X=\mathbf{x}} \right)$$

- ▶ Exact log-likelihood evaluation via inverse transformations
- ▶ Sampling from the model

$$\mathbf{z} \sim p_Z(\mathbf{z}), \quad \mathbf{x} = \mathbf{f}_{\theta}(\mathbf{z})$$

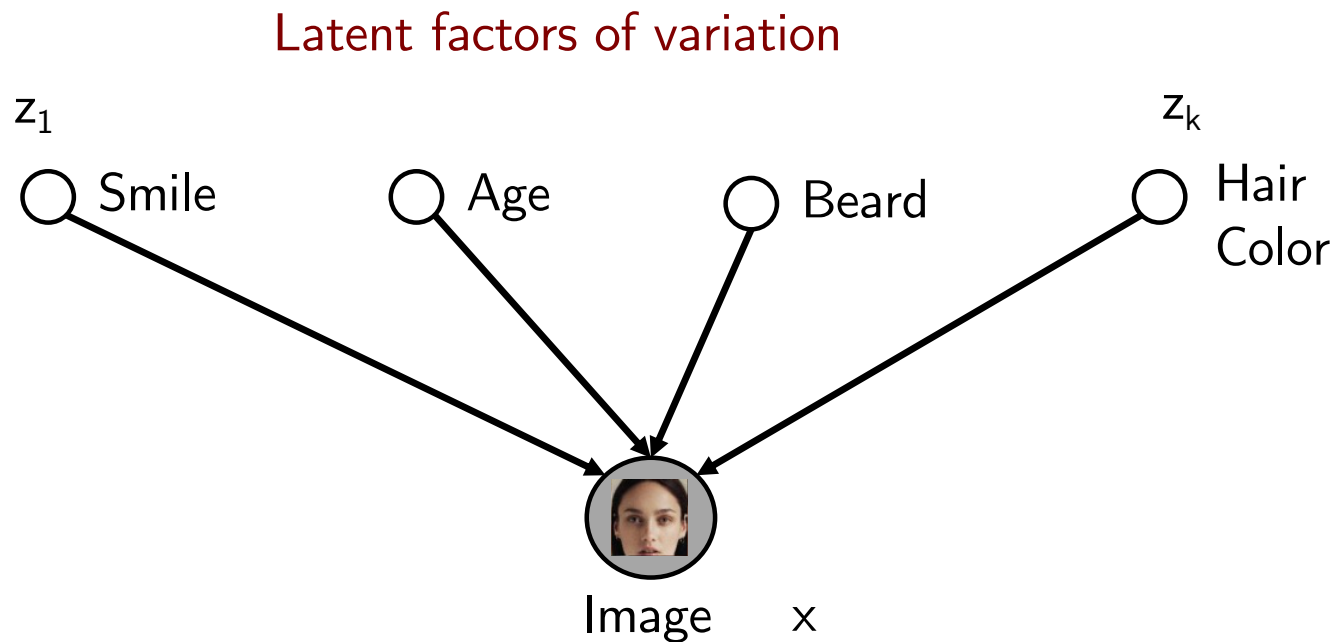
- ▶ Inference over the latent representations:

$$\mathbf{z} = \mathbf{f}_{\theta}^{-1}(\mathbf{x})$$

Example: GLOW

- ▶ Generative Flow with Invertible 1x1 Convolutions

<https://blog.openai.com/glow/>



Kingma, Dhariwal, 2018

Flow Models

- ▶ Simple prior that allows for sampling and tractable likelihood evaluation e.g., isotropic Gaussian
- ▶ Invertible transformations with tractable evaluation:
 - ▶ Likelihood evaluation requires efficient evaluation of inverse
 - ▶ Sampling requires efficient evaluation of inverse
- ▶ Tractable evaluation of determinants of Jacobian for large models
 - ▶ Computing determinants for a large matrix is prohibitive
 - ▶ Key idea: Determinant of triangular matrices is the product of the diagonal entries, i.e., an $O(n)$ operation