Generative Models

Fatemeh Saleh

Australian
National
University

Australian National University

Outline

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

Generative Models

- Informally:
  - A model that can generate new data after learning from the dataset.
- More formally:
  - A generative model models the joint distribution $P(X, Y)$ of the observation X and the target Y.
  - A discriminative model models the conditional distribution $P(Y|X)$.

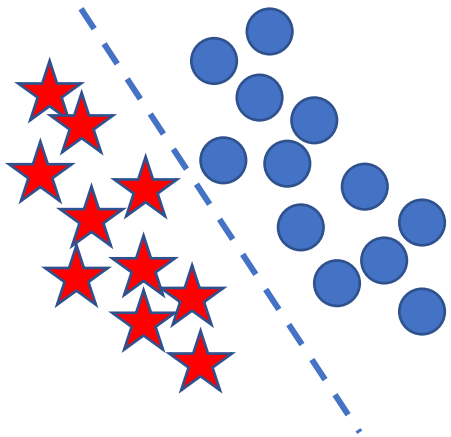
Discriminative versus Generative

- Discriminative Model
  - Tries to learn the discriminative information from the data
  - Example: Classify C1 vs C2 vs C3
    - Finds a good decision boundary by directly modeling conditional distribution $P(Y|X)$
    - Learns mappings from inputs to classes

- Generative Model
  - Tries to learn the distribution of the data
  - Models the distribution of inputs characteristic of the class
  - For classification, builds a model of $P(X|Y)$ and then applies Bayes Rule

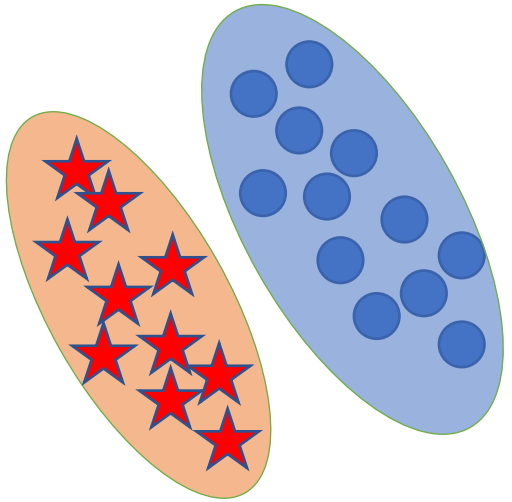
Discriminative versus Generative

Discriminative Model



Learn $P(Y|X)$ directly
Logistic regression use a sigmoid function to estimate this directly

Generative Model



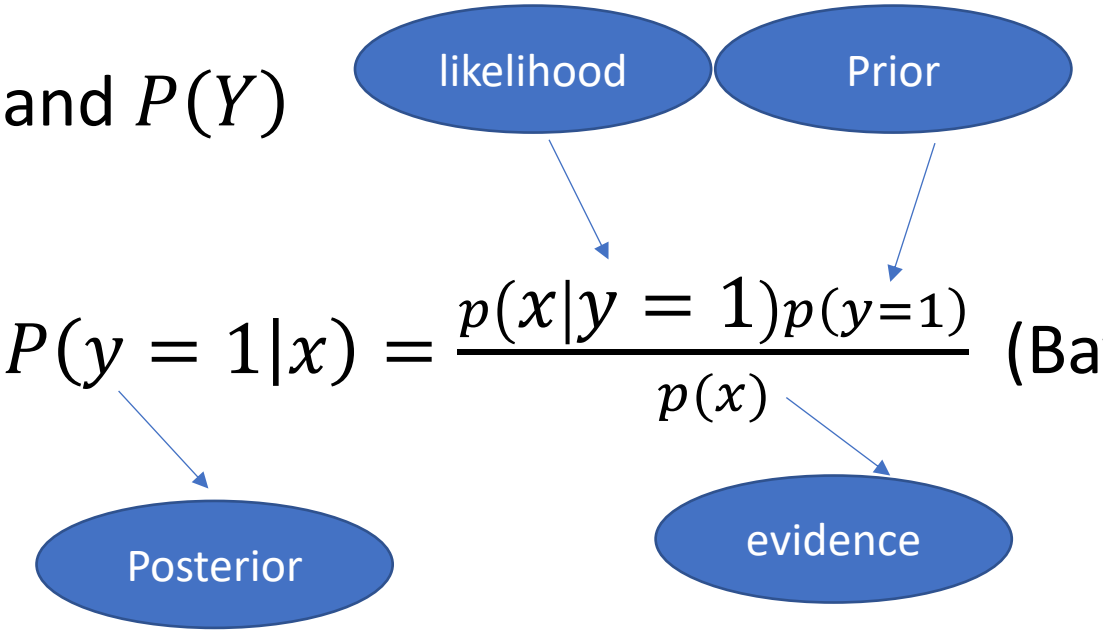
Learn $P(X|Y)$ and $P(Y)$

Learning What do “stars” look like?

The class Prior
p(y=0) p(y=1)

Discriminative versus Generative

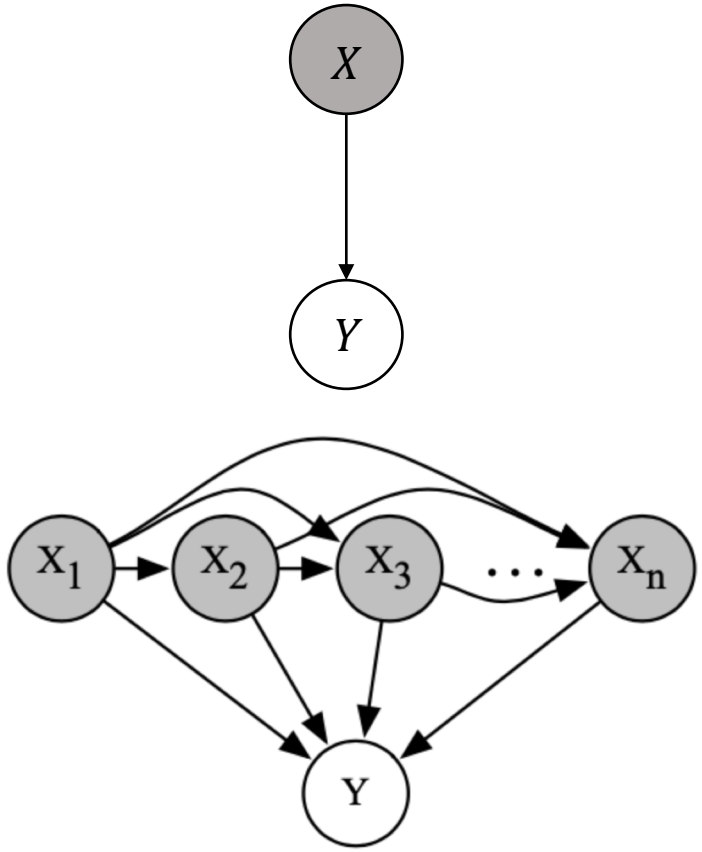
- Suppose that: $P(X|Y)$ and $P(Y)$
- New example x
- We can then compute $P(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)}$ (Bayes Rule)



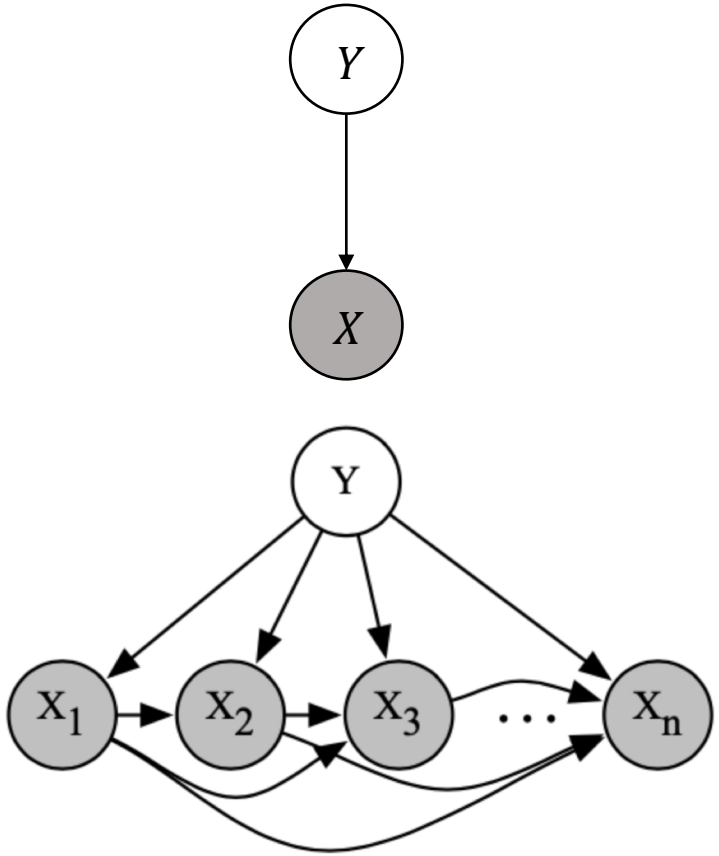
- where $p(x) = \sum_y p(x,y) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$

Discriminative versus Generative

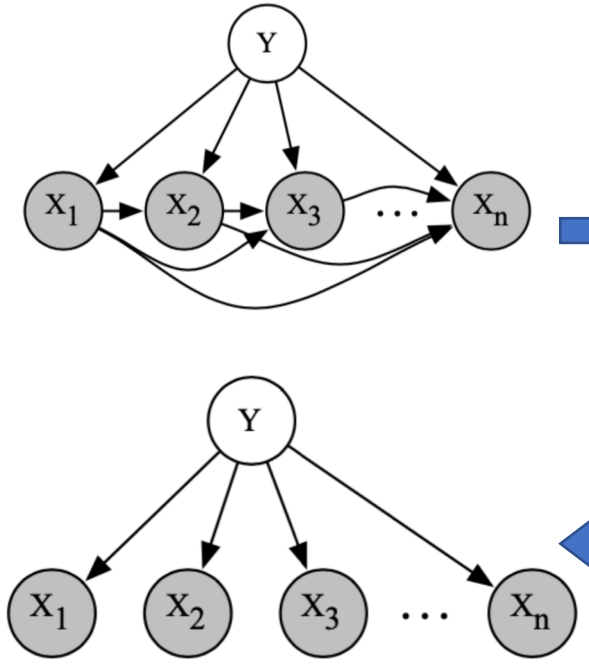
Discriminative Model



Generative Model



Generative Model (naive Bayes)



$$P(y, x_1, x_2, \dots, x_n) = P(y)P(x_1|y)P(x_2|y, x_1) \dots P(x_n|y, x_1, \dots, x_{n-1})$$

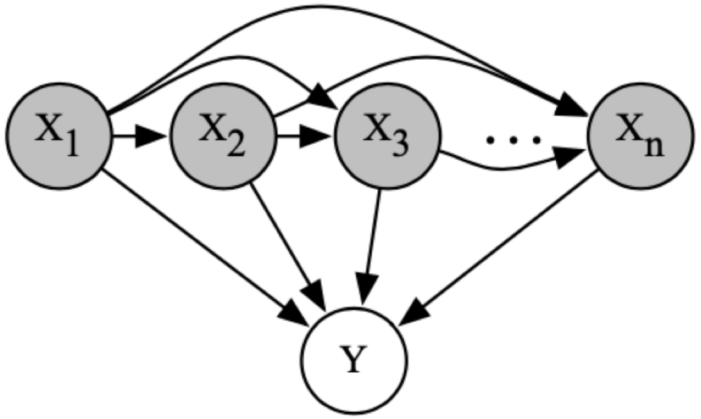
$$x_1 \perp x_2 \perp \cdots \perp x_n \mid y$$

$$P(y, x_1, x_2, \dots, x_n) = P(y)P(x_1|y)P(x_2|y) \dots P(x_n|y) = P(y)\prod_{i=1}^{n} P(x_i|y)$$

Estimate parameters from training data. Predict with Bayes rule:

$$P(Y = 1|x_1, x_2, \dots, x_n) = \frac{P(Y = 1)\prod_{i=1}^{n} P(x_i|Y = 1)}{\sum_{y=\{0,1\}} P(Y = y)\prod_{i=1}^{n} P(x_i|Y = y)}$$

Discriminative Model (logistic regression)



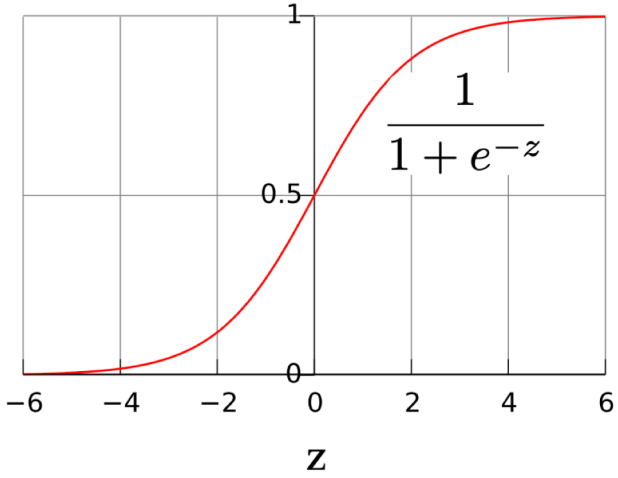
$$P(Y = 1|X; \alpha) = f(X, \alpha)$$

$$z(\alpha, \mathrm{X}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$$

$$P(Y = 1|X; \alpha) = \sigma(z(X, \alpha))$$

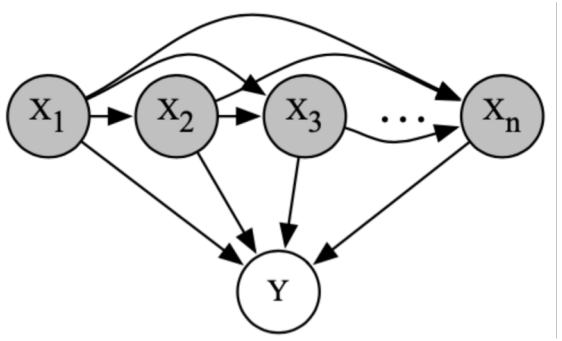
where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

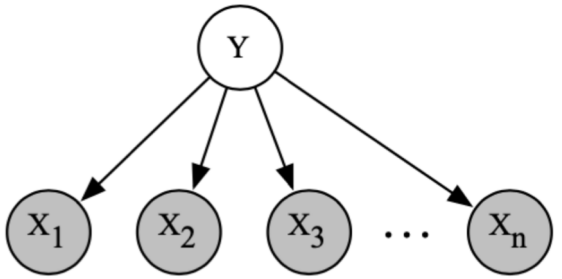


Discriminative versus Generative

Discriminative (logistic regression)



Generative (Naïve Bayes)



Discriminative models are powerful

- Logistic model does not assume $x_i \perp x_{i-1} | y$, unlike naïve Bayes
- This can make a big difference in many applications
  - For example, in spam classification, let $x_1 = 1[\text{“}bank\text{”} \; in \; email]$ and $x_2 = 1[\text{“}account\text{”} \; in \; email]$
  - Regardless of whether spam, these always appear together, i.e. $x_1 = x_2$
  - Learning in naïve Bayes results in $p(x_1|y) = p(x_2|y)$, thus naïve Bayes double counts the evidence
  - Learning with logistic regression sets $\alpha_1 = 0$ or $\alpha_2 = 0$, in effect ignoring it

Generative models are still very useful

- Using a conditional model is only possible when X is always observed

Credit: https://deepgenerativemodels.github.io/

Generative Models

- Given a training set of examples, e.g., images of cats:

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

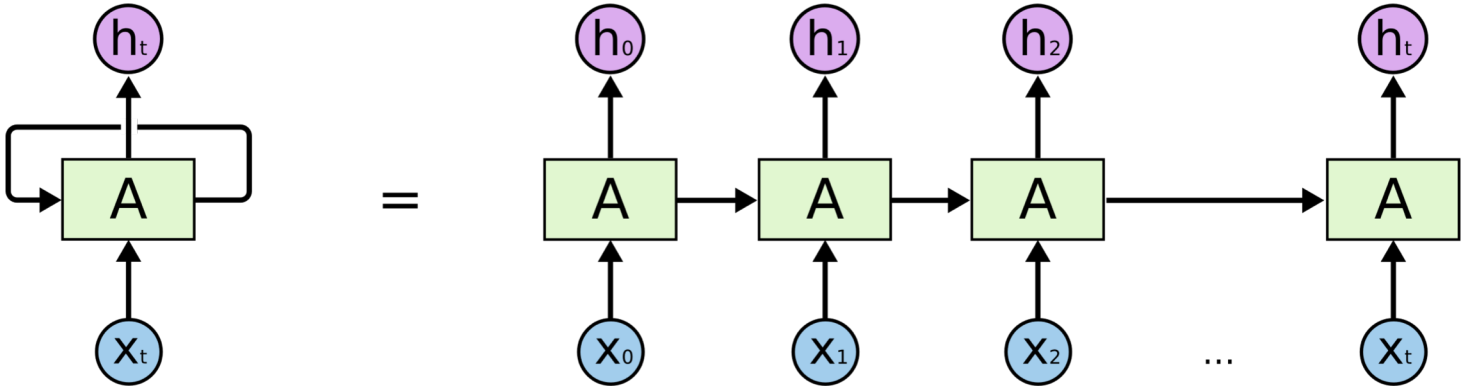
- We want to learn a probability distribution $p(x)$ over images x for
  - **Generation:** If we sample $x_{new} \sim p(x)$, x_{new} should look like a cat (sampling)
  - **Density estimation:** $p(x)$ should be high if x looks like a cat, and low otherwise
  - **Unsupervised representation learning:** The model should be able to learn what these images have in common, e.g., ears, tail, etc. (features)

Autoregressive Models

Autoregressive Models

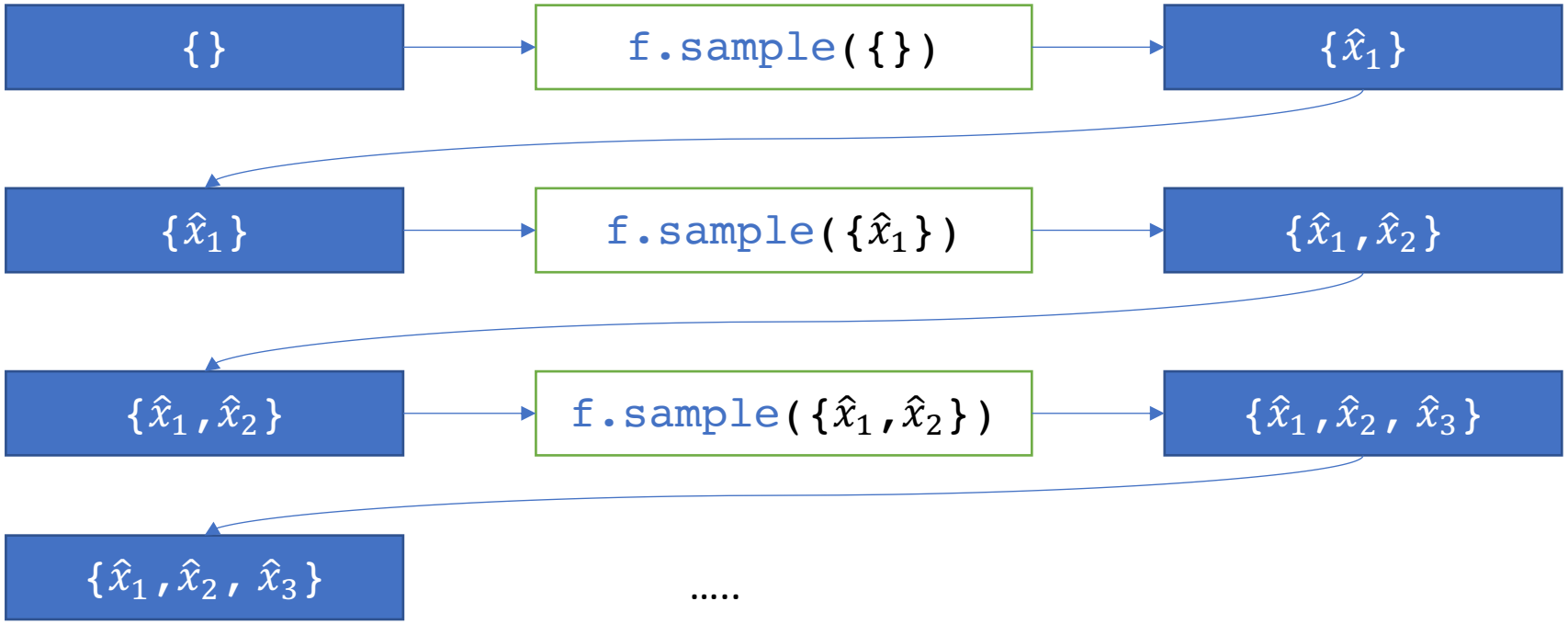
- Definition:
  - Given the observation as sequences $x_1, x_2, \ldots, x_T$, we decompose the likelihood into a product of conditional distributions:

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$



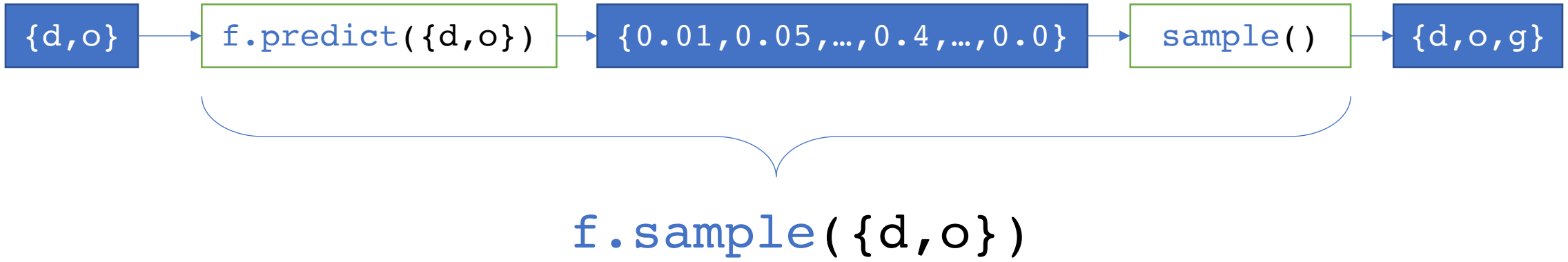
Autoregressive Models

- Given such a model `f` and a sampling function `sample`, the generative process for a full sequence is:

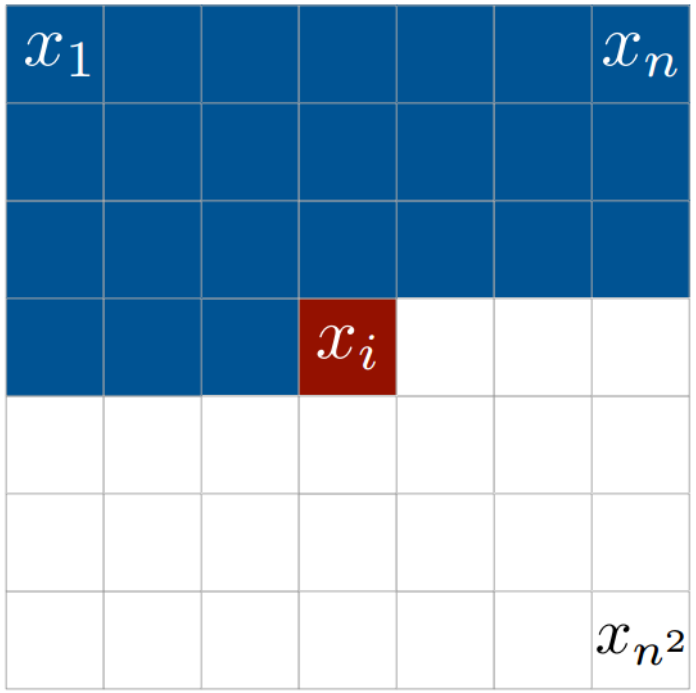


Autoregressive Models

- How `f.sample(.)` works?
  - Consider the example of character generation, where we only have C possible options to choose from.
  - The model maps the d dimensional data to C dimensional probability distribution and then samples from the estimated distribution.

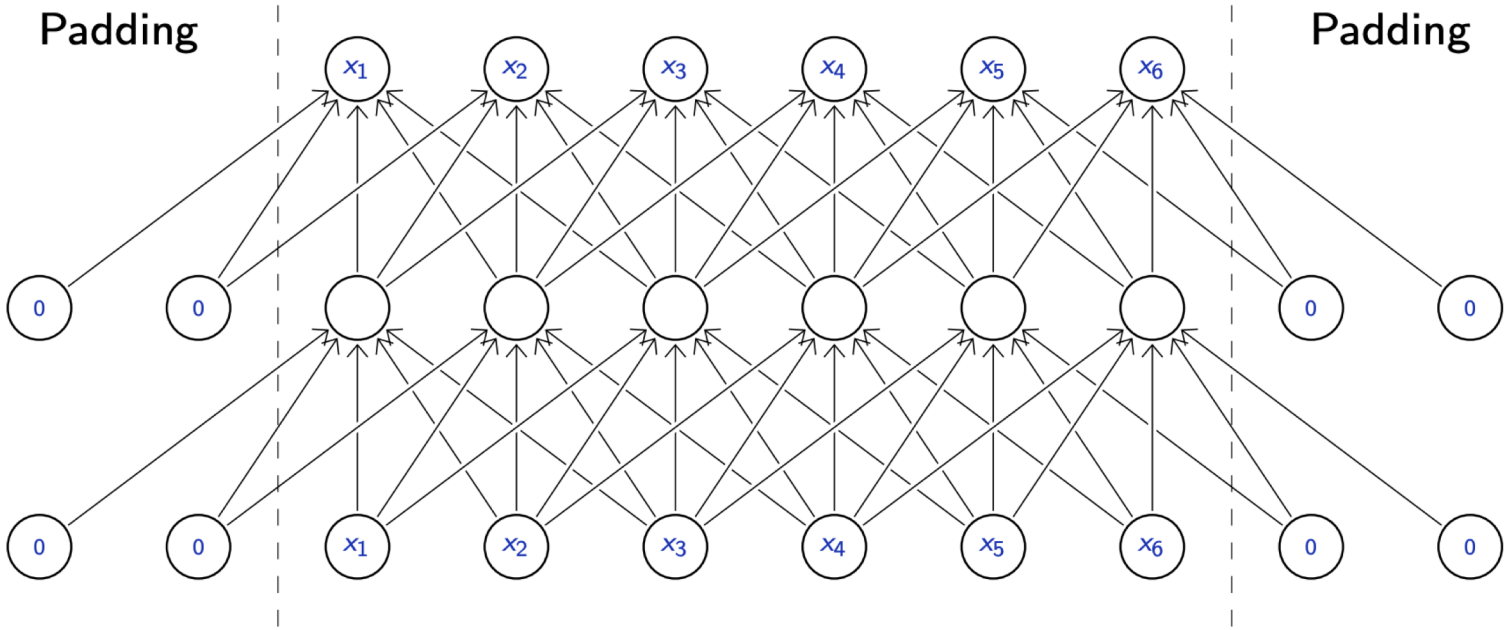


Autoregressive Models



Autoregressive Models

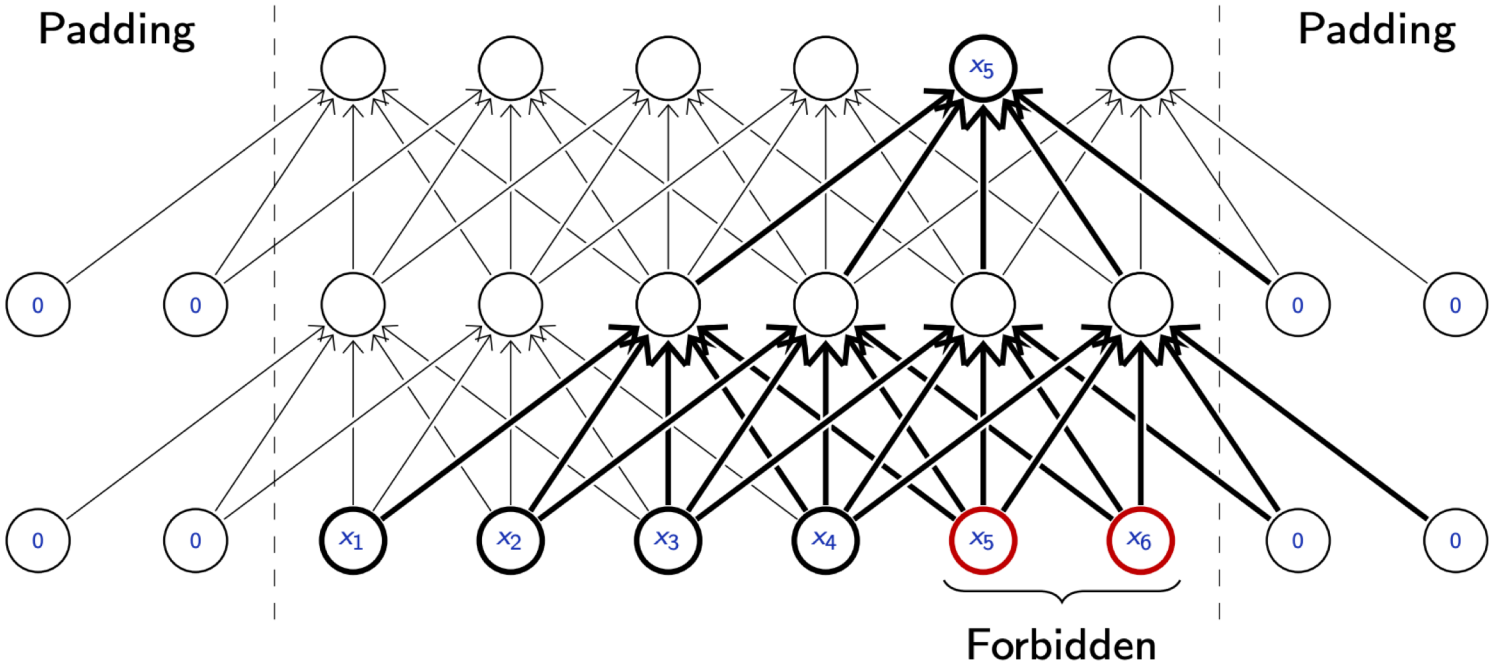
- One solution is to use *causal convolutions.*
  - How to create causal convolutions?



Credit: https://fleuret.org/ee559/

Autoregressive Models

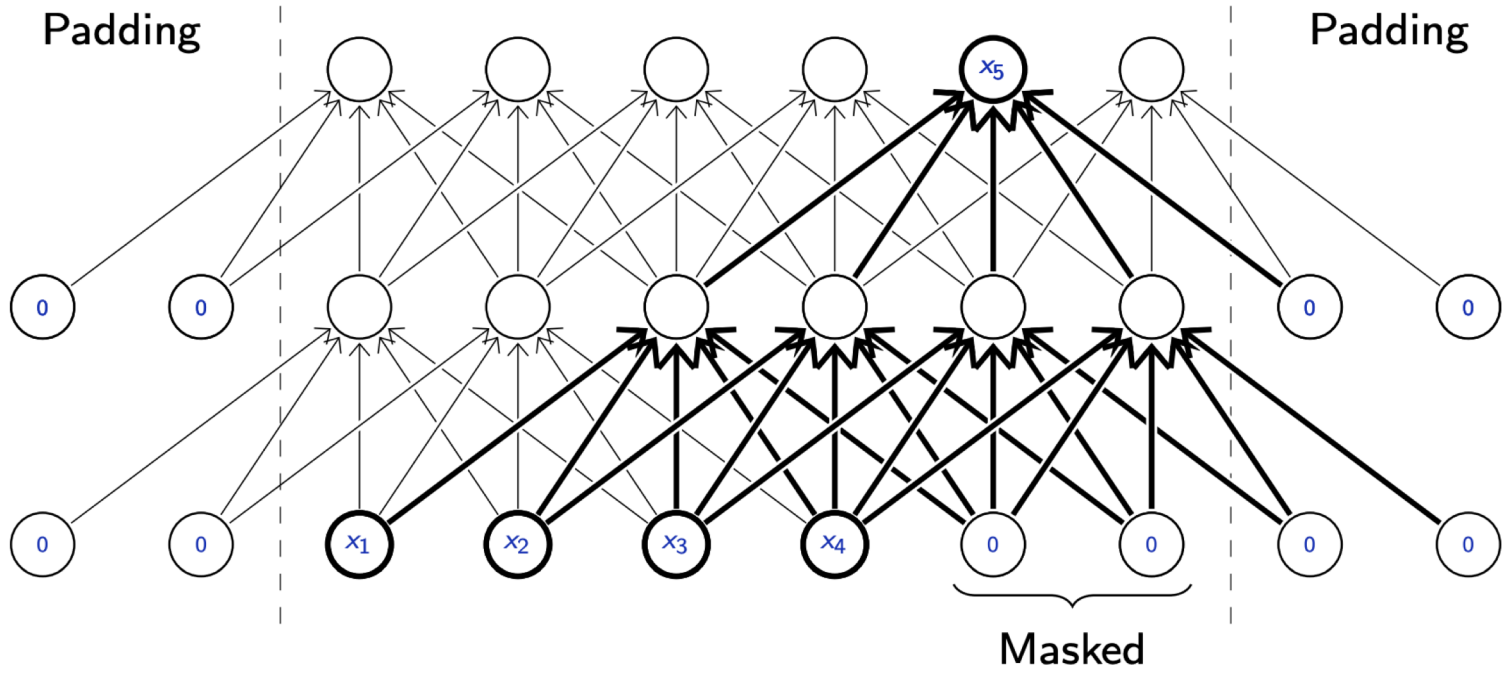
- One solution is to use *causal convolutions.*
  - How to create causal convolutions?



Credit: https://fleuret.org/ee559/

Autoregressive Models

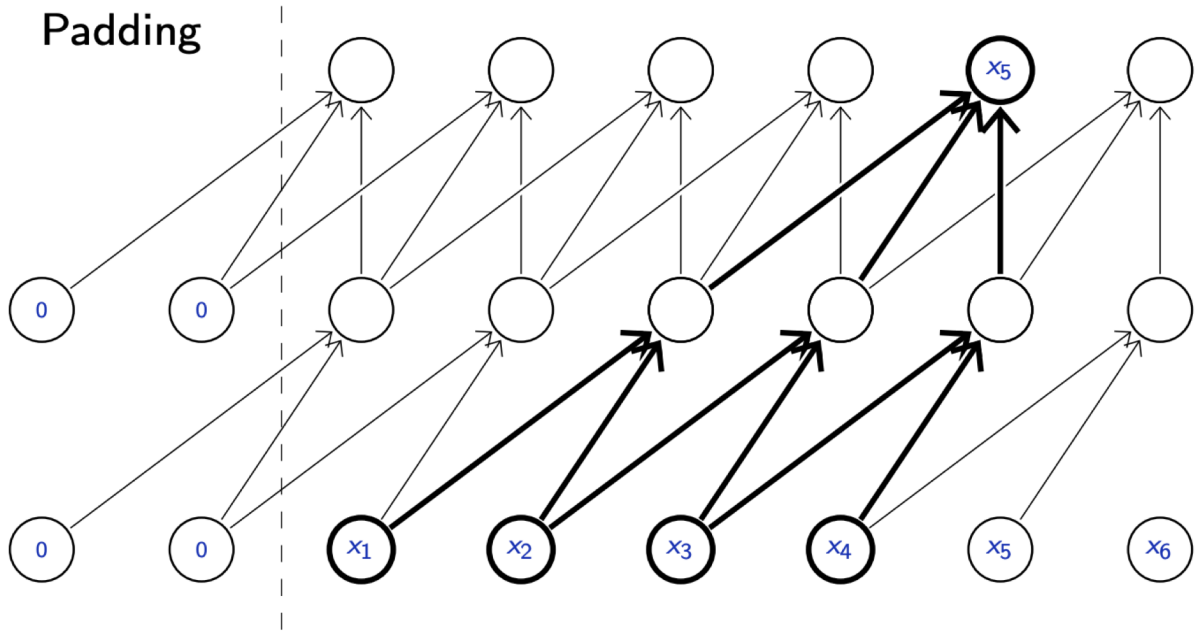
- One solution is to use *causal convolutions.*
  - How to create causal convolutions?



Credit: https://fleuret.org/ee559/

Autoregressive Models

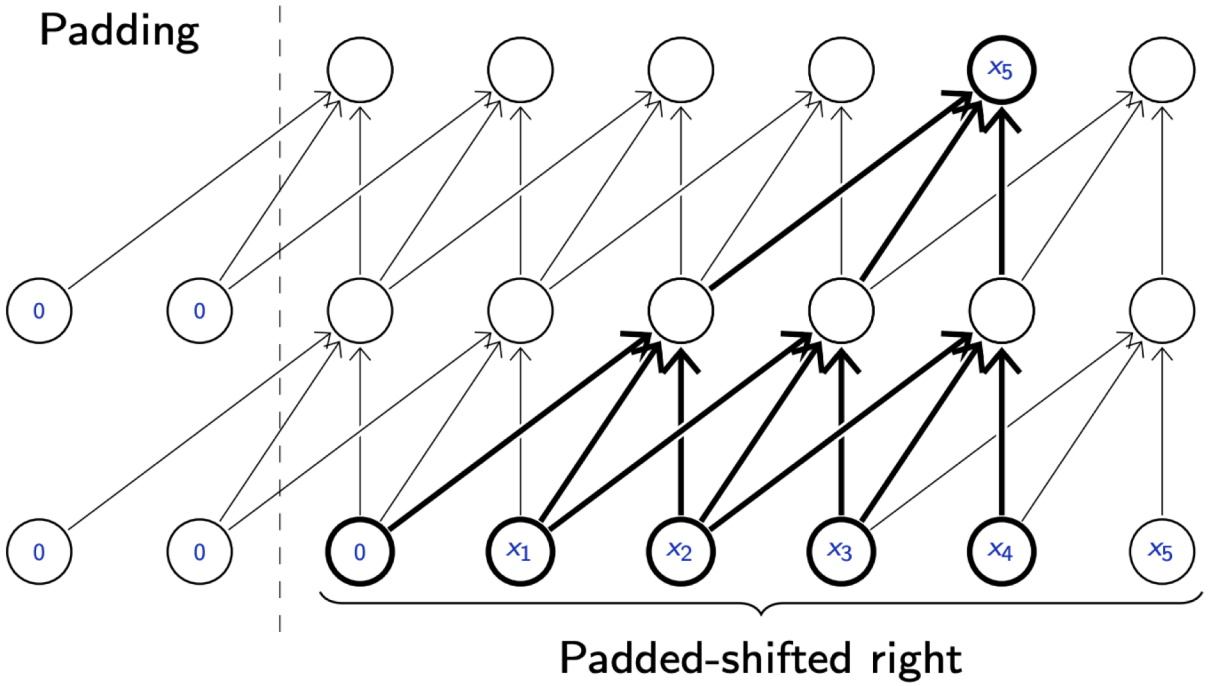
- One solution is to use *causal convolutions.*
  - How to create causal convolutions?



Credit: https://fleuret.org/ee559/

Autoregressive Models

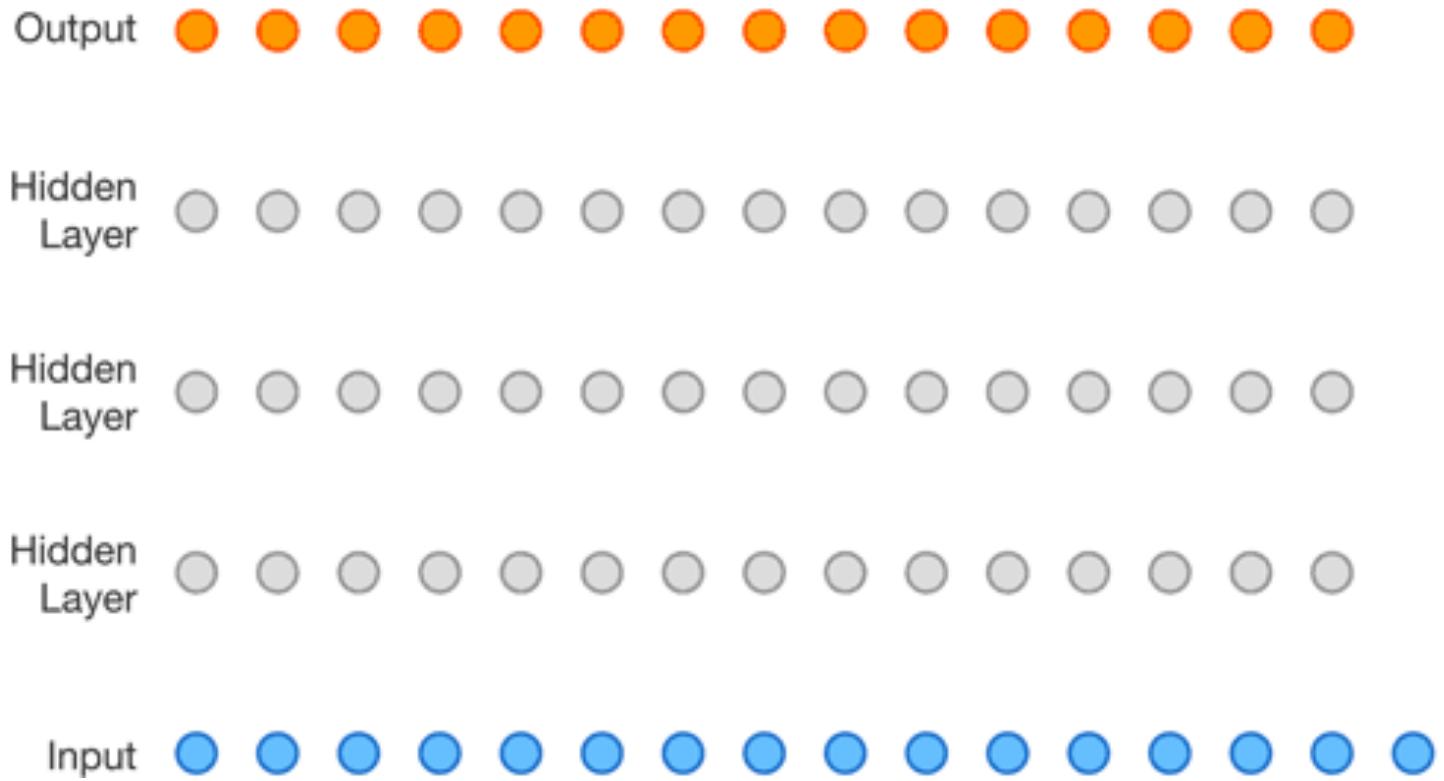
- One solution is to use *causal convolutions.*
  - How to create causal convolutions?



Credit: https://fleuret.org/ee559/

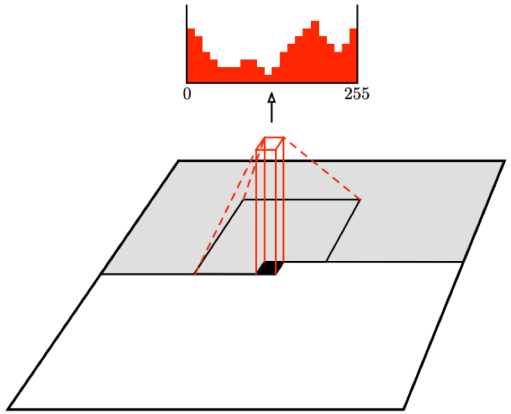
Australian National University

Autoregressive Models

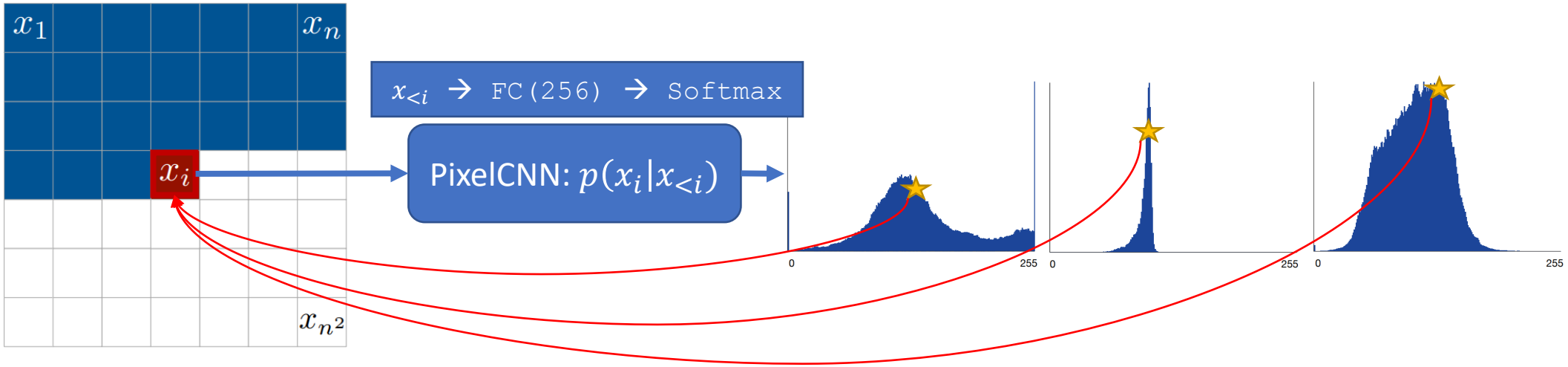


Autoregressive Models

- Example: PixelCNN



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0



Australian National University

Autoregressive models

- Pros:
  - Easy to sample from
  - Easy to compute the likelihood
  - Natural to train via maximum likelihood
- Cons:
  - Slow to generate new data
  - Does not directly learn unsupervised representations of the data.

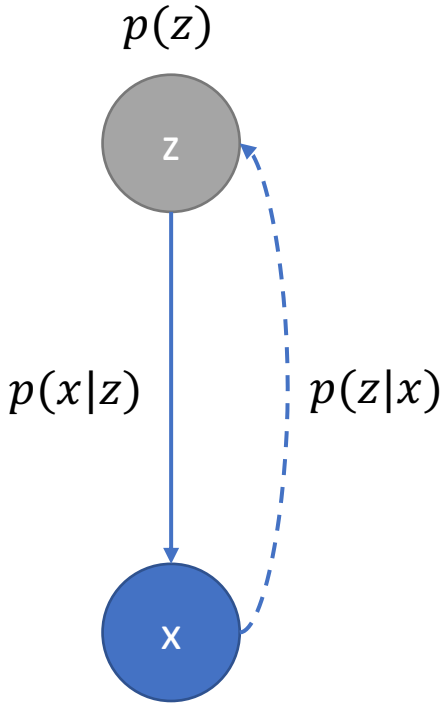
Latent Variable Models

Latent Variable Models (LVMs)

- Similar to Autoregressive models, LVMs are likelihood-based
  - Unlike Autoregressive, LVMs utilize an unobserved latent variable
  - Similar to Autoregressive, LVMs are trained with maximum likelihood
- Pros:
  - Powerful and well-understood
  - Sampling from LVMs is very efficient (no autoregressive component)
  - Well-suited for representation learning
- Cons:
  - Conceptually, more complex than fully-observed models
  - Need to use approximate inference (e.g., in VAEs) or use restricted models (e.g., in NFs)

Latent Variable Models

- LVM defines a distribution over observations x by using a latent variable z and specifying
  - The prior distribution $p(z)$ for the latent variable
  - The likelihood $p(x|z)$ that connects the latent variable to the observation



- The joint distribution

$$p(x, z) = p(x|z)p(z)$$

- We are interested in computing the marginal likelihood $p(x)$ and the posterior distribution $p(z|x)$

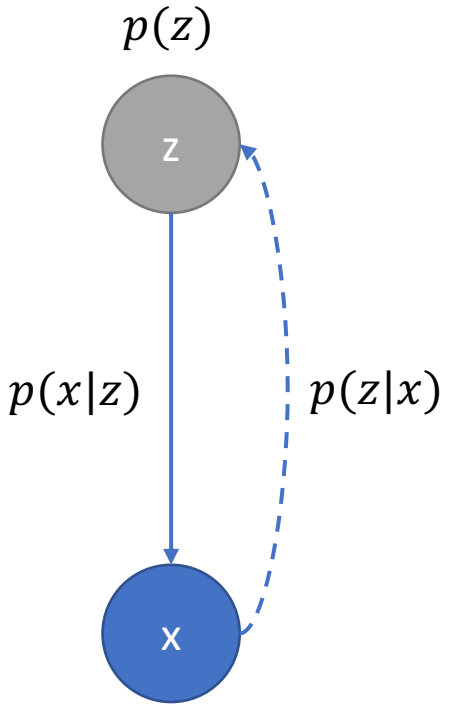
Why latent variables?

- Latent variables explains the data
- To generate an observation from such explanation

$$z \sim p(z)$$
$$x \sim p(x|z)$$

- The inverse of generation is called inference,

$$z \sim p(z|x)$$



Inference

- Inference

$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x,z)}{\int p(x,z)dz}$$

- This requires computing the marginal likelihood of the observations

$$p(x) = \int p(x,z)dz$$

Inference

- Why inference is important?
  - Explaining the observation
    - Inferring the posterior distribution for a datapoint allows us to determine what latent configurations could have plausibly generate such datapoint.
  - Learning
    - Training latent variable models requires performing the inference.

Inference

- Exact inference is hard!

$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x,z)}{\int p(x,z)dz}$$

- Computing $\int p(x,z)dz$ is intractable as it requires considering all configurations of z for a reasonable approximation of $p(x)$.

Inference

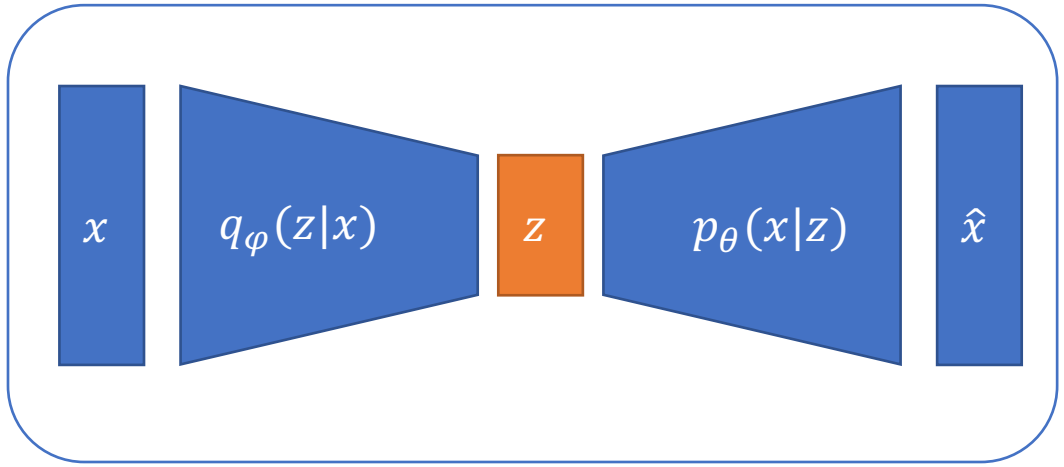
- How to avoid intractable inference?
  - Option 1: Using approximate inference
    - No exact posterior, but, allows us using more expressive models
  - Option 2: Designing models that allows tractable inference
    - Simpler training, but, in expense of less expressive models

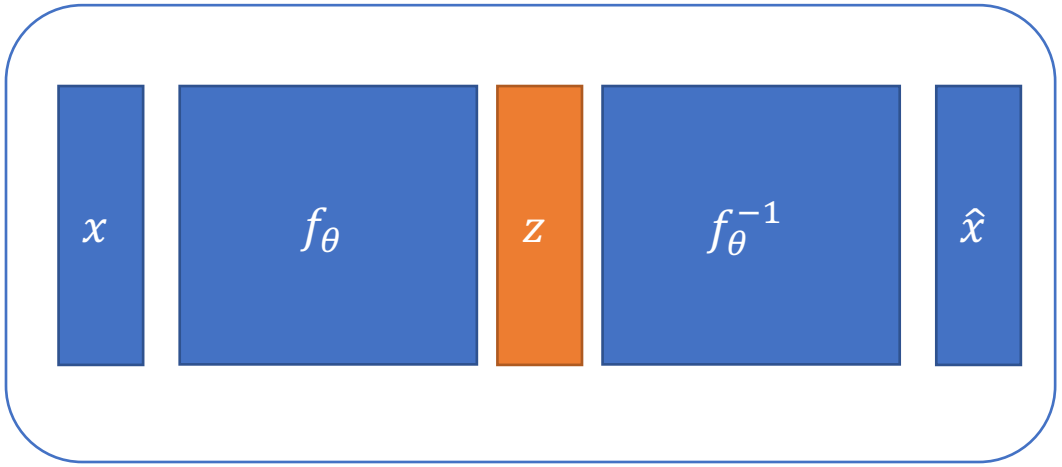
Option 1:
Variational Autoencoders (VAEs)

Option 2:
Normalizing Flows (NFs)

Inference

- How to avoid intractable inference?
  - Option 1: Using approximate inference
    - No exact posterior, but, allows us using more expressive models
  - Option 2: Designing models that allows tractable inference
    - Simpler training, but, in expense of less expressive models





Variational Autoencoders

Variational Autoencoders

- Exact inference is hard!

$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x,z)}{\int p(x,z)dz}$$

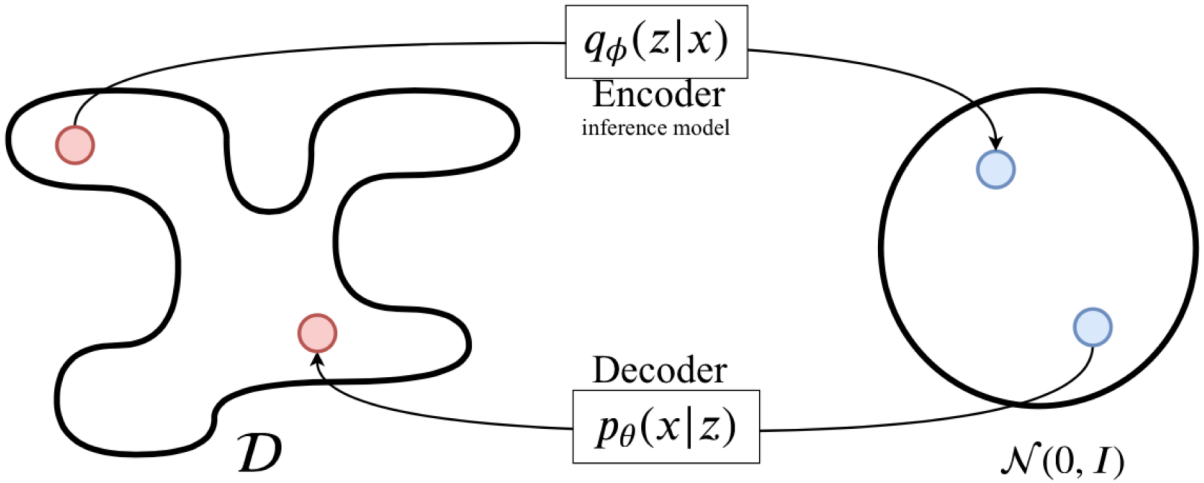
- VAEs try to approximate the true posterior $p(z|x)$ with another distribution $q_\varphi(z|x)$ such that$q_\varphi(z|x) \approx p(z|x)$.

- We call $q_\varphi(z|x)$ the approximate posterior.

Variational Autoencoders

- An overview

Data space
(unknown distribution)



Prior space
(known distribution)

Variational Autoencoders

- VAE's objective function
  - $q_{\varphi}(z|x) \approx p(z|x)$

$$D_{KL}\left[q_{\phi}(z|x)||p_{\theta}(z|x)\right] = \mathbf{E}_{z\sim q_{\phi}(z|x)}\left[\log \frac{q_{\phi}(z|x)}{p_{\theta}(z|x)}\right]$$

Background on KL Divergence

- KL divergence provides a way of quantifying the difference between two distributions.
- KL divergence is defined as

$$KL(q(z)||p(z)) = \mathrm{E}_{q(z)}\left[\log \frac{q(z)}{p(z)}\right]$$

- KL divergence is
  - Non-negative
  - KL=0 if $q(z) = p(z)$
  - Non-symmetric, $KL(q(z)||p(z)) \neq KL(p(z)||q(z))$

Variational Autoencoders

- Expanding the KL divergence between the approximate and true posterior distributions

$$D_{KL}\left[q_\phi(z|x)||p_\theta(z|x)\right] = \mathbf{E}_{z\sim q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= \mathbf{E}_{z\sim q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z|x)\right]$$

True posterior

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{p_\theta(x)}$$

Variational Autoencoders

- The data distribution $p_\theta(x)$ is independent of z, so

$$D_{KL}\left[q_\phi(z|x)||p_\theta(z|x)\right] = \mathbf{E}_{z\sim q_\phi(z|x)}\Big[\log q_\phi(z|x) - \log p_\theta(x|z) - \log p(z)\Big] + \log p_\theta(x)$$

- By putting $p_\theta(x)$ to the left side

$$D_{KL}\left[q_\phi(z|x)||p_\theta(z|x)\right] - \log p_\theta(x) = \mathbf{E}_{z\sim q_\phi(z|x)}\Big[\log q_\phi(z|x) - \log p_\theta(x|z) - \log p(z)\Big]$$

$$\log p_\theta(x) - D_{KL}\left[q_\phi(z|x)||p_\theta(z|x)\right] = \mathbf{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z) - \big(\log q_\phi(z|x) - \log p(z)\big)\Big]$$

$$= \mathbf{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] - \mathbf{E}_{z\sim q_\phi(z|x)}\Big[\log q_\phi(z|x) - \log p(z)\Big]$$

Variational Autoencoders

- So, it can be written as

$$\log p_\theta(x) - D_{KL}\left[q_\phi(z|x)||p_\theta(z|x)\right] = \mathbf{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left[q_\phi(z|x)||p(z)\right]$$

Log-likelihood of the data

KL divergence between the approximate and the true posterior

not computable!

According to definition, it is positive!

Reconstruction loss

KL divergence between the approximate posterior and prior

- Thus, we can only optimize the lower bound on the data log-likelihood

$$\log p_\theta(x) \geq \mathbf{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left[q_\phi(z|x)||p(z)\right]$$

Variational Autoencoders

- Therefore, the ELBO of the VAE is

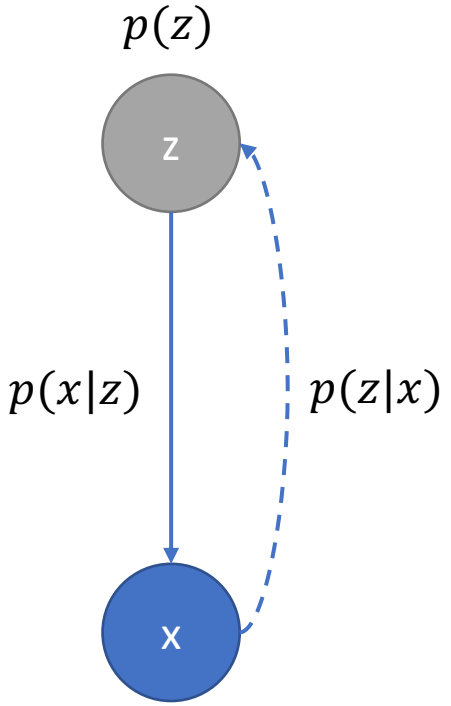
$$\log p_\theta(x) \geq \mathbf{E}_{z\sim q_\phi(z|x)}\big[\log p_\theta(x|z)\big] - D_{KL}\big[q_\phi(z|x)||p(z)\big]$$

- And, similarly, the ELBO of conditional VAE is

$$\log p_\theta(x|c) \geq \mathbf{E}_{z\sim q_\phi(z|x)}\big[\log p_\theta(x|z,c)\big] - D_{KL}\big[q_\phi(z|x,c)||p(z|c)\big]$$

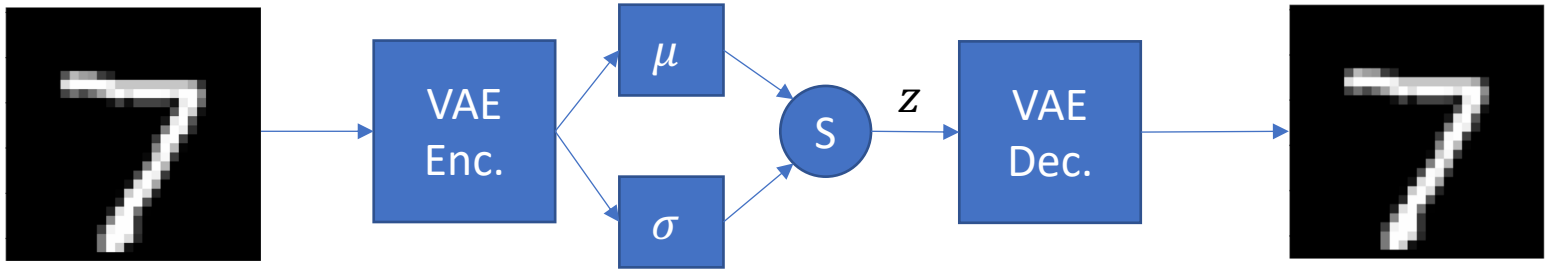
Variational Autoencoders

- In practice,
  - Prior: $p(z) = \mathcal{N}(0, I)$
  - Encoder: $q_{\varphi}(z|x) = \mathcal{N}\left(\mathrm{Net}_{\varphi}(x), \mathrm{diag}\left(\mathrm{Net}_{\varphi}(x)\right)\right)$
  - Decoder: $p_{\theta}(x|z) = \mathrm{Net}_{\theta}(z)$



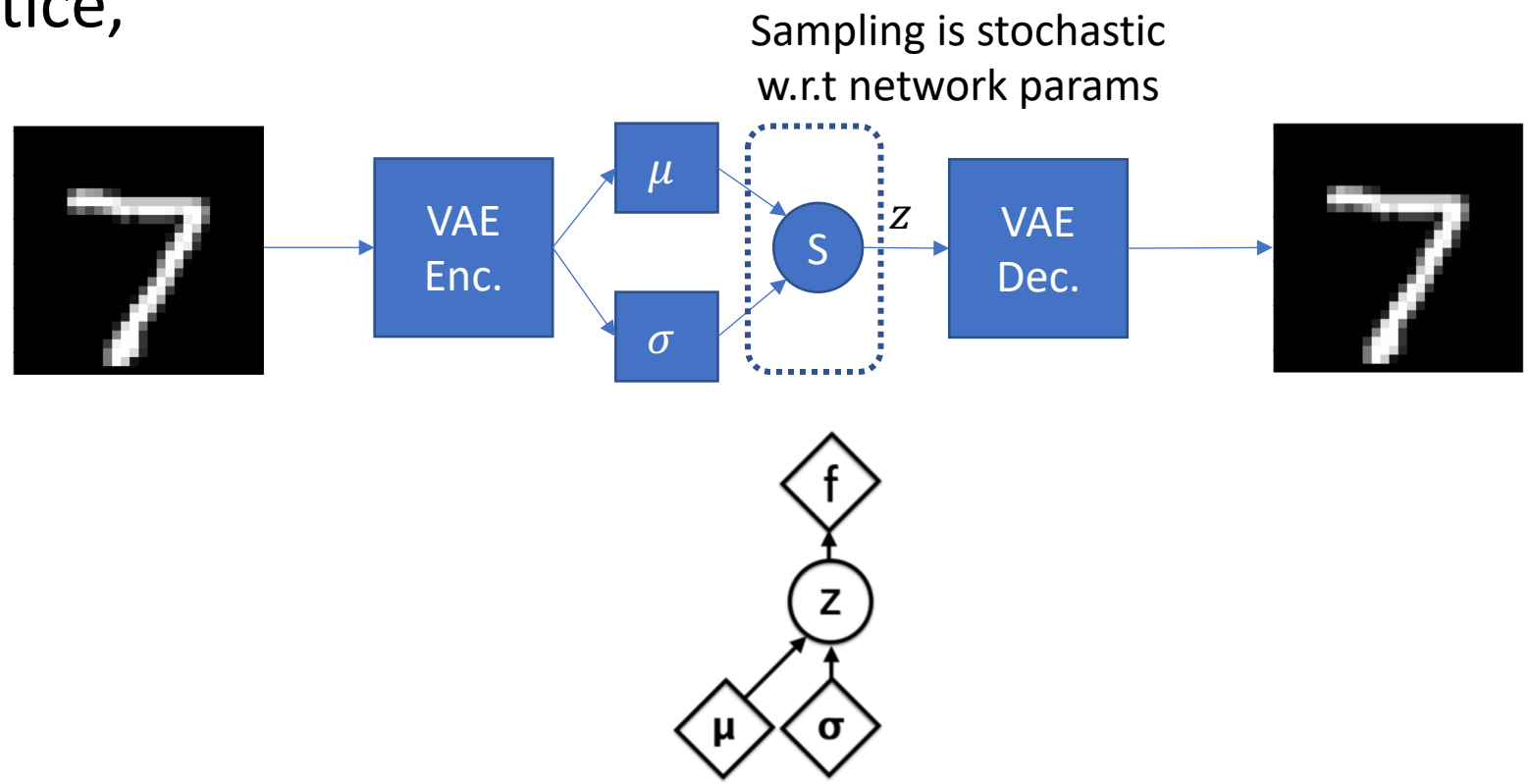
Variational Autoencoders

- In practice,



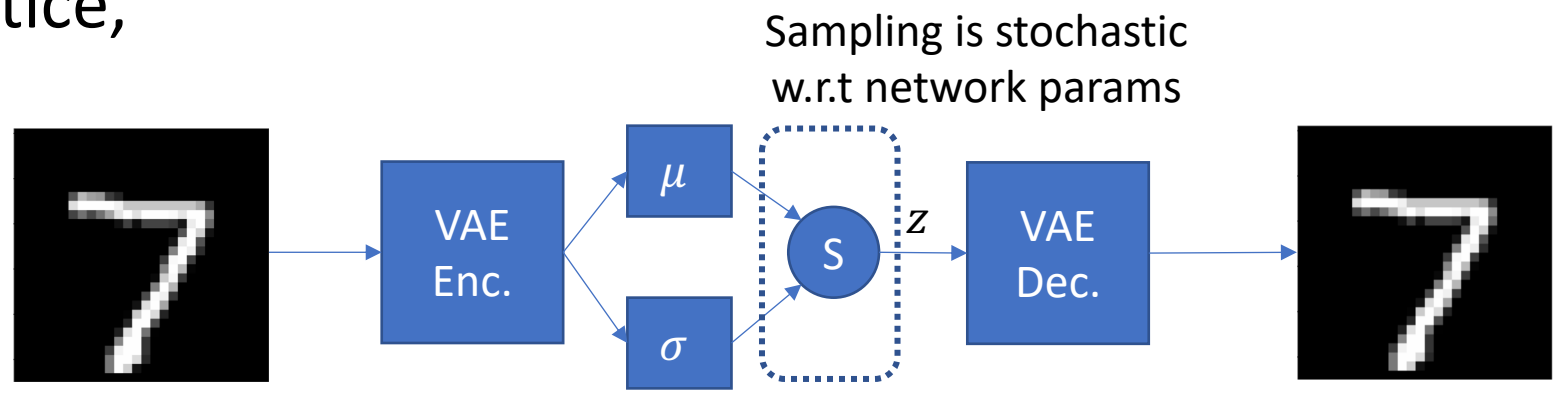
Variational Autoencoders

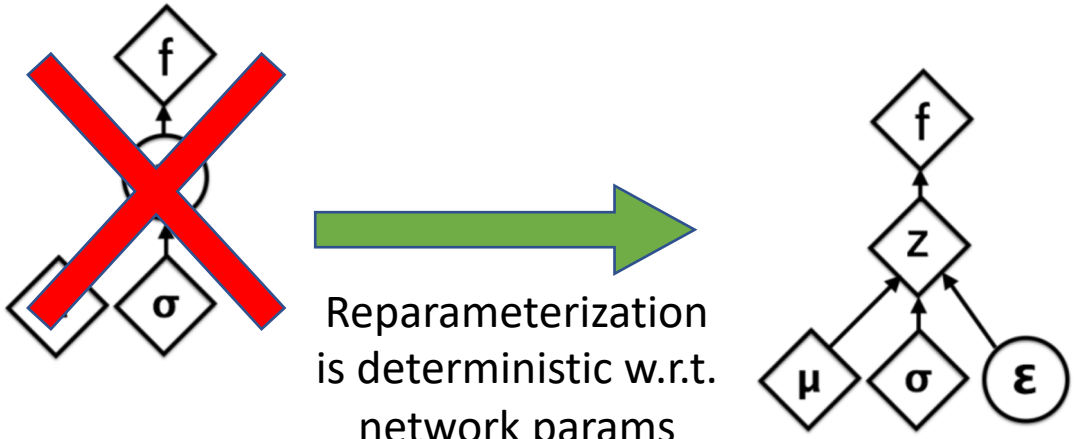
- In practice,



Variational Autoencoders

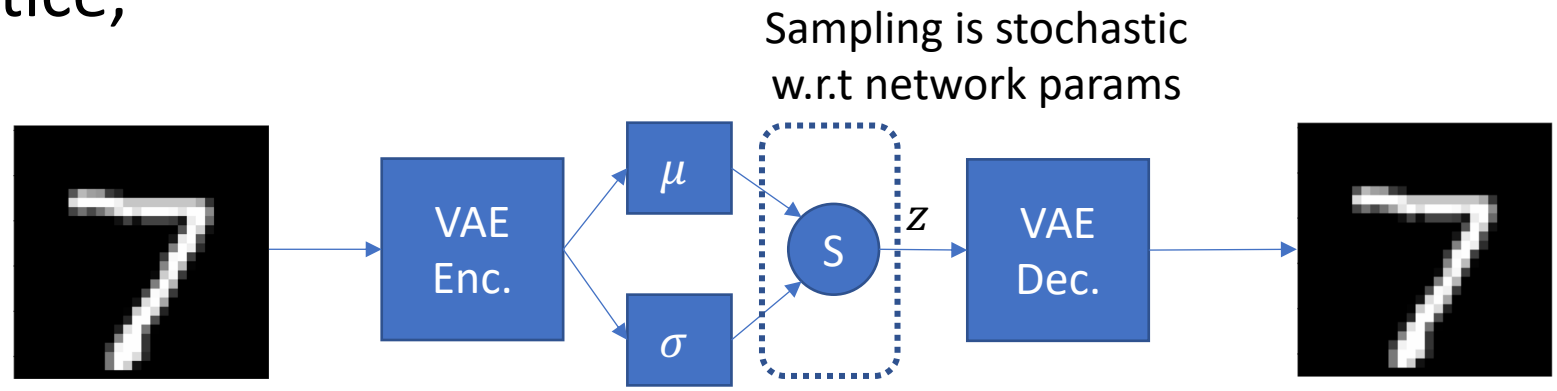
- In practice,



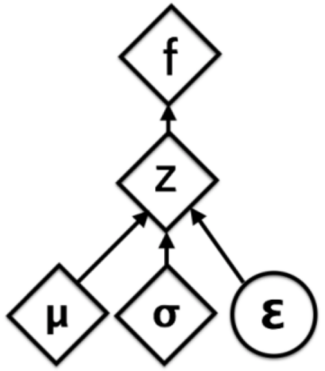


Variational Autoencoders

- In practice,

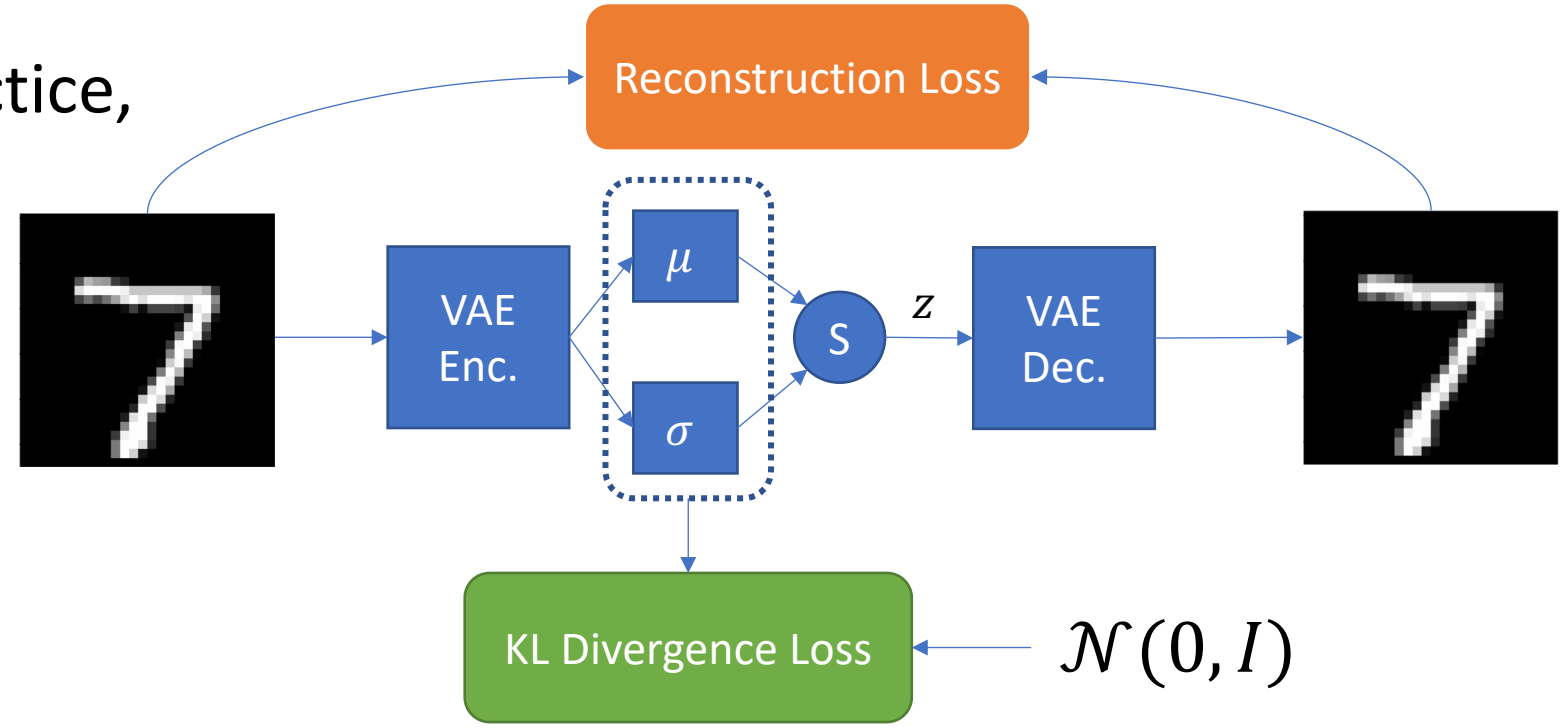


- To sample from posterior
  - Sample epsilon from $\epsilon \sim \mathcal{N}(0, I)$
  - Shift and scale ϵ given estimated posterior parameters
  - $z = \mu + \sigma \odot \epsilon$
  - This is equivalent to $z \sim \mathcal{N}(\mu, \sigma)$



Variational Autoencoders

- In practice,

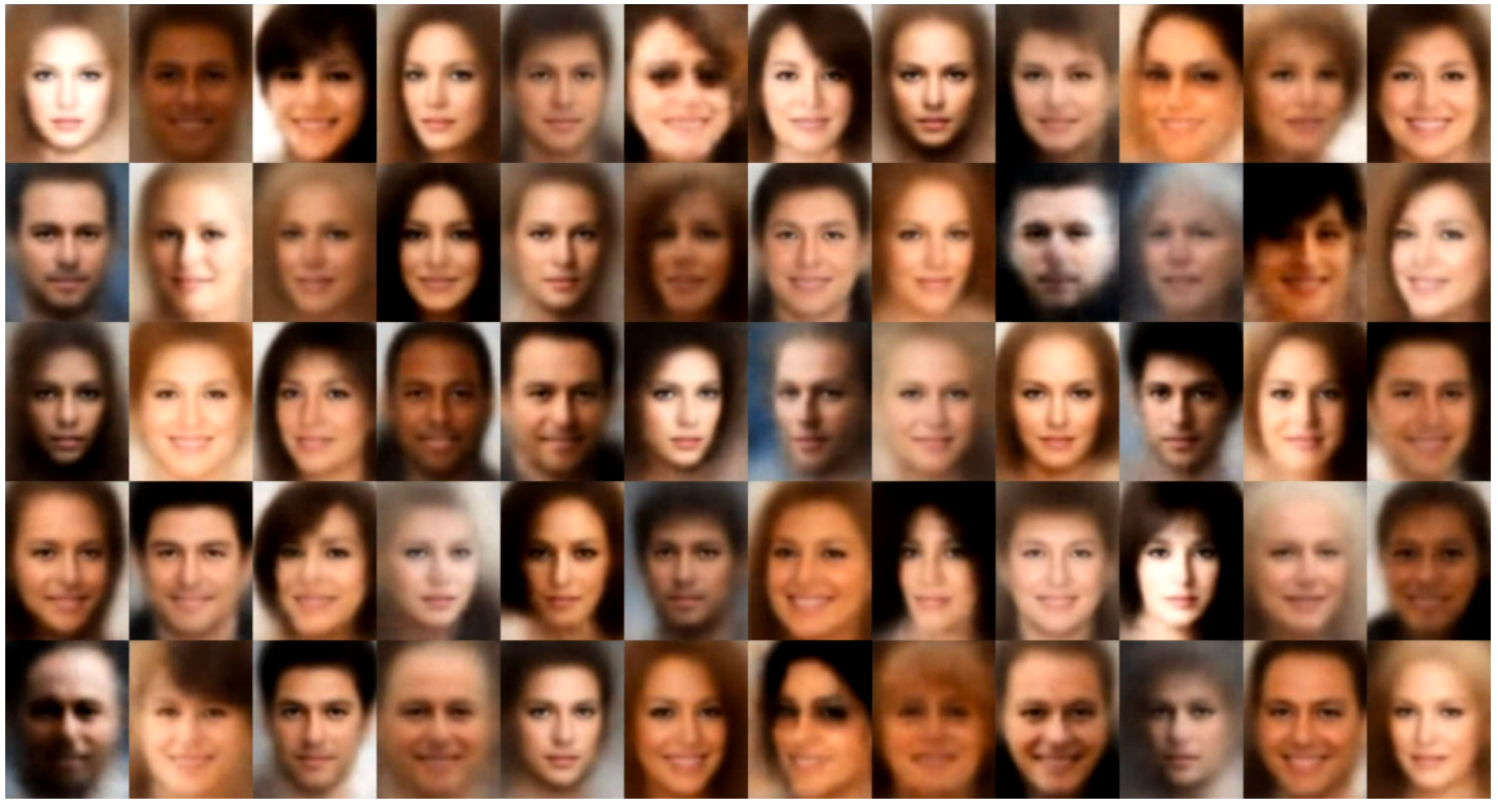


$$\log p_\theta(x) \geq \mathbf{E}_{z \sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left[q_\phi(z|x)||p(z)\right]$$

Australian National University

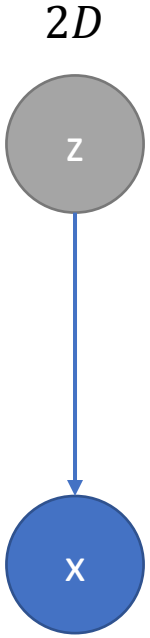
Variational Autoencoders

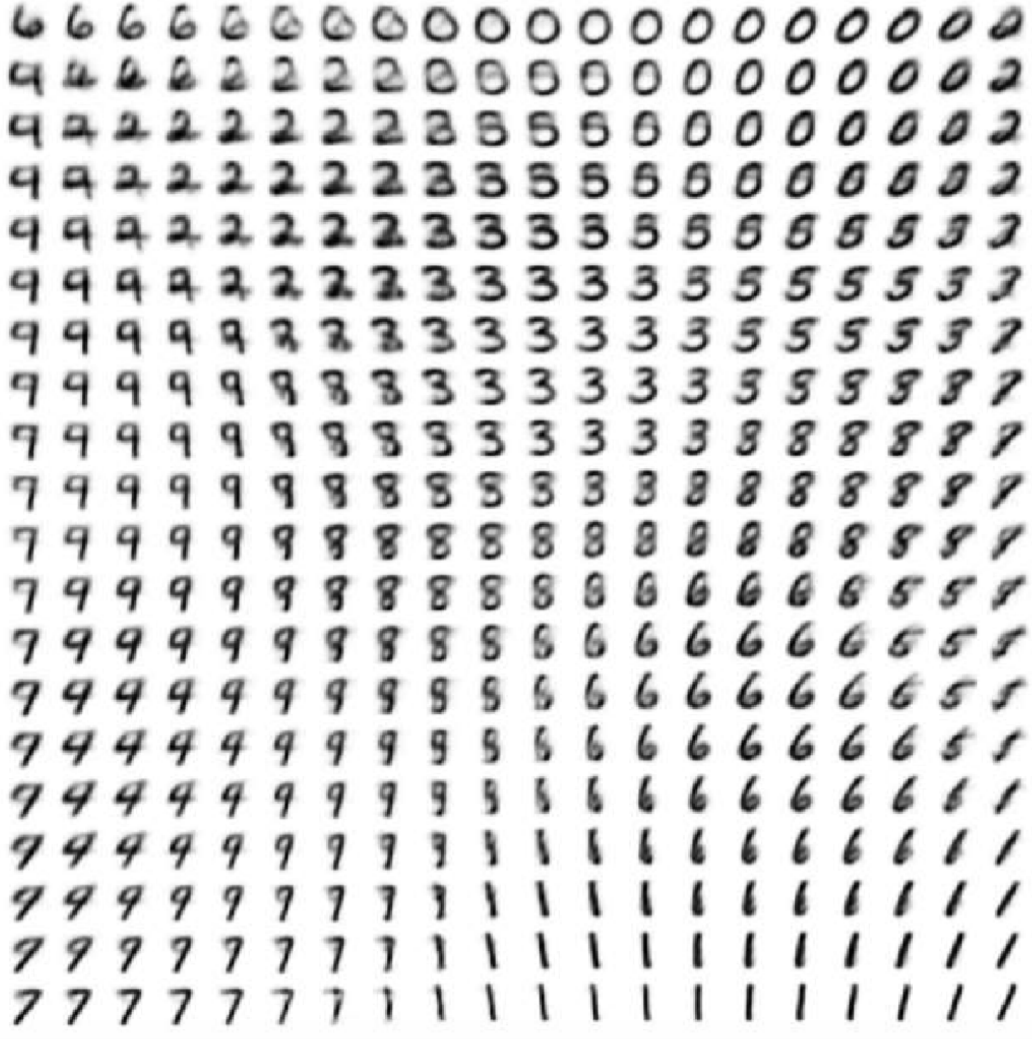
- Generating new samples



Variational Autoencoders

- Representation Learning / Visualization





Variational Autoencoders

[stat.ML] 8 Jul 2020

NVAE: A Deep Hierarchical Variational Autoencoder

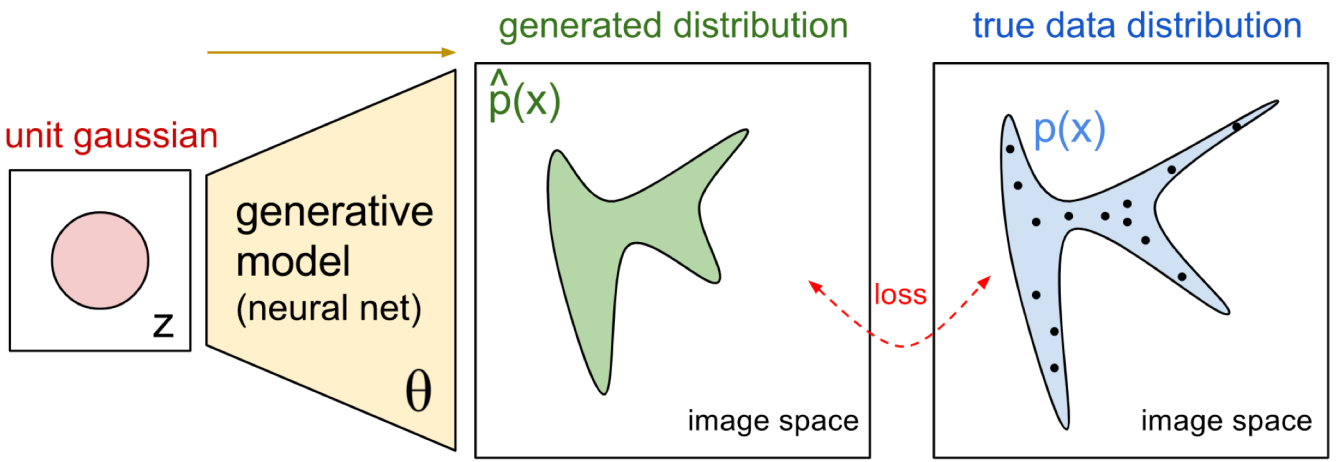
Arash Vahdat, Jan Kautz
NVIDIA
{avahdat, jkautz}@nvidia.com

Figure 1: 256×256-pixel samples generated by NVAE, trained on CelebA HQ [28].

Generative Adversarial Networks

Implicit Generative Models

- Implicit generative models implicitly define a probability distribution
- Start by sampling the code vector z from a fixed, simple distribution
- The generator network computes a differentiable function mapping z to an x in data space

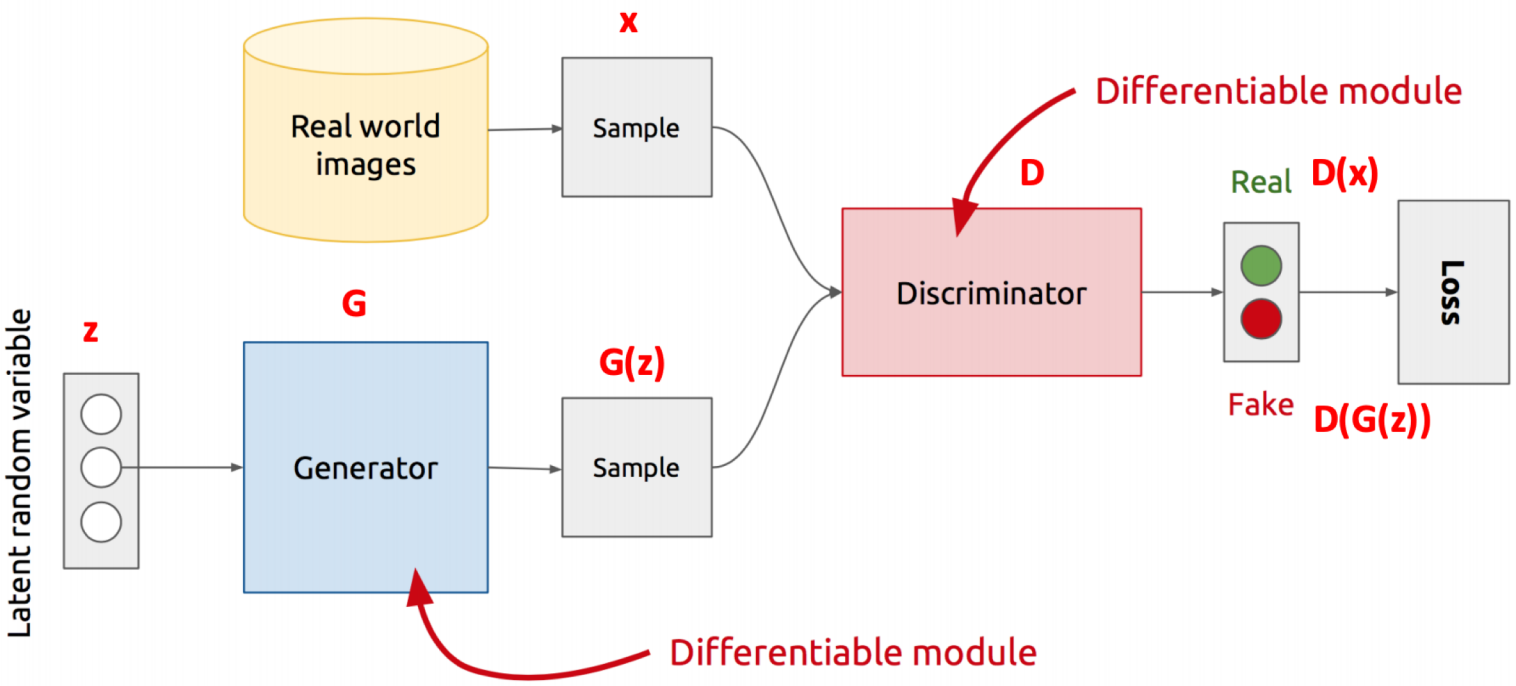


https://openai.com/blog/generative-models/

Generative Adversarial Networks (GANs)

- Generator: generate fake samples, tries to fool the Discriminator
- Discriminator: tries to distinguish between real and fake samples
- Train them against each other
- Repeat this and we get better Generator and Discriminator

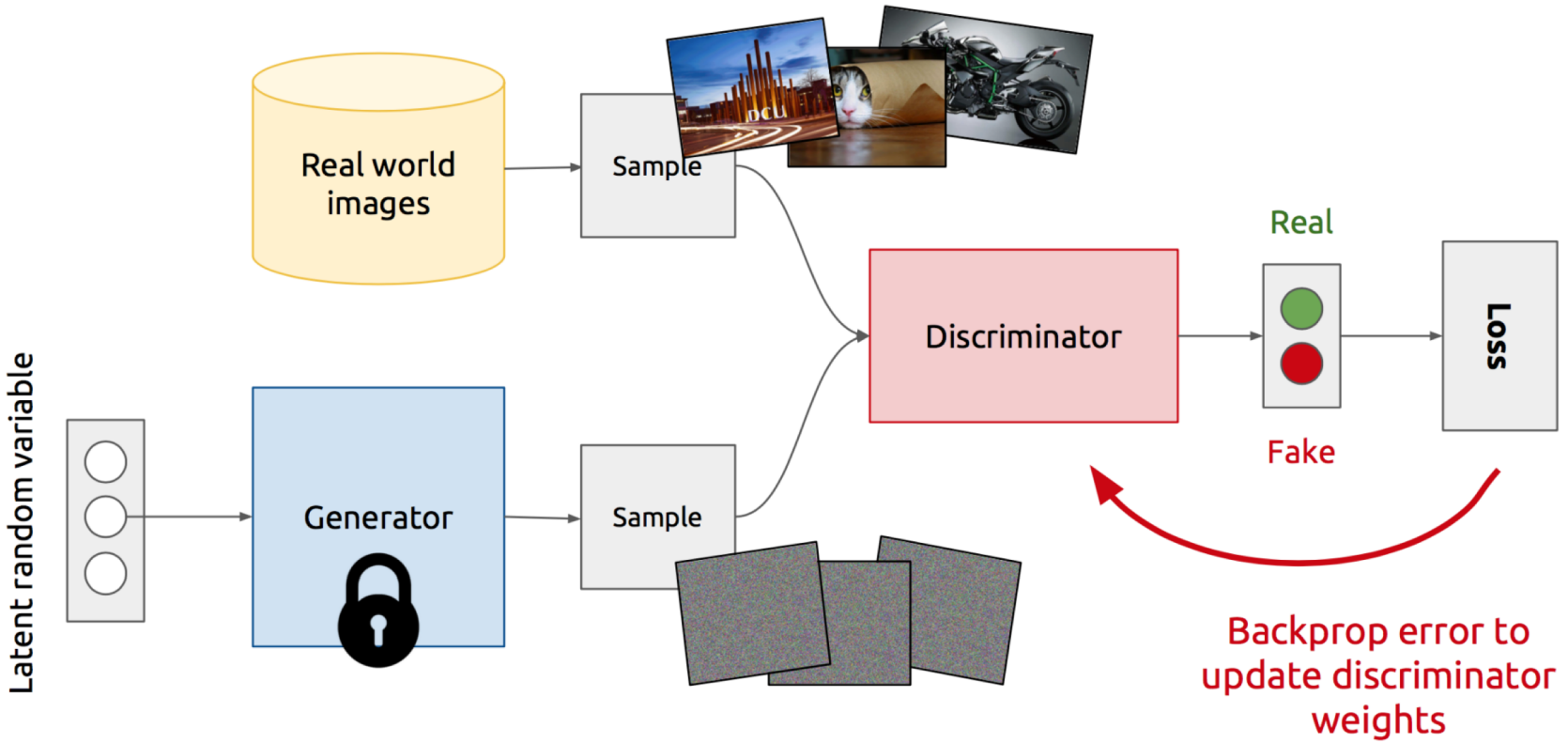
GAN's Architecture



Z is some random noise (Gaussian/Uniform).
Z can be thought as the latent representation of the image.

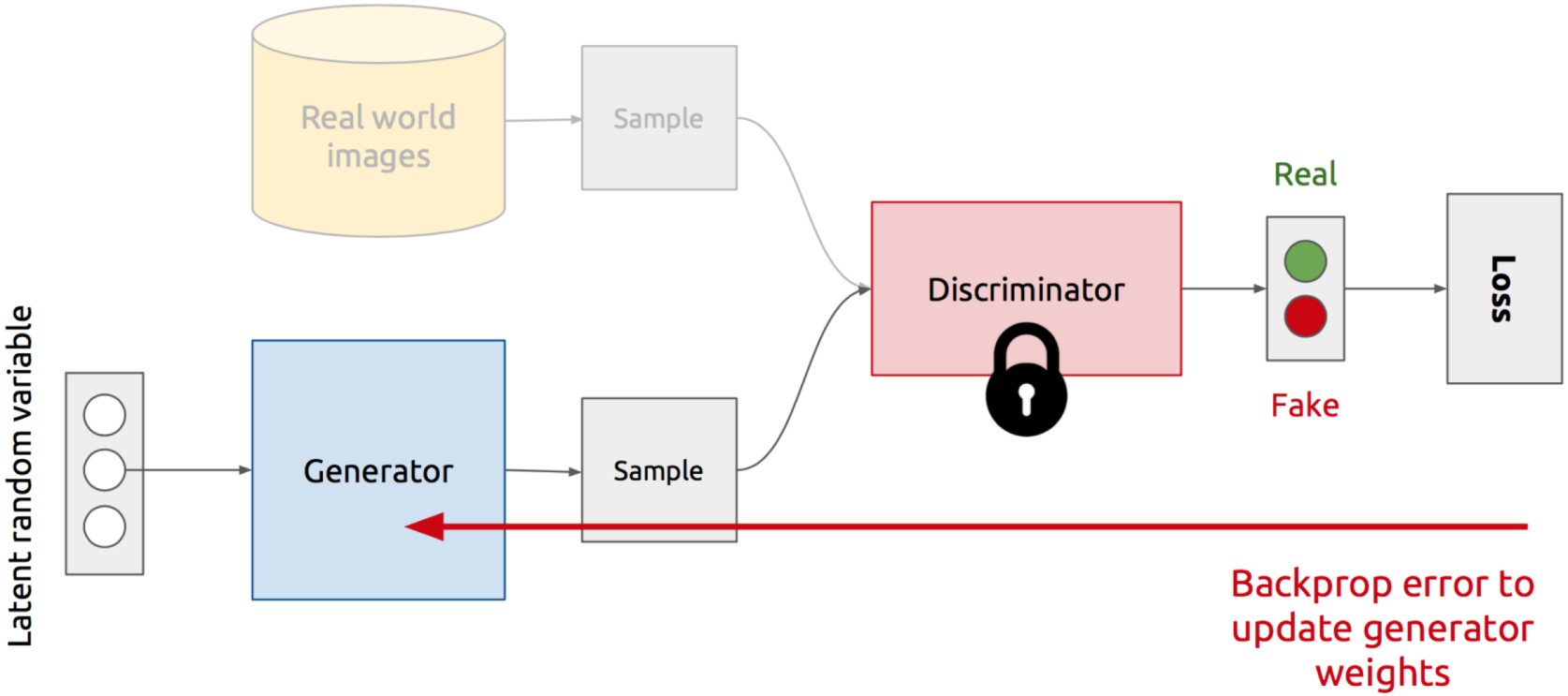
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN's formulation

- Let D denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{L}_D = \mathbb{E}_{x\sim p(x)}[-\log D(x)] + \mathbb{E}_{z\sim q(z)}[-\log\Big(1 - D\big(G(z)\big)\Big)]$$

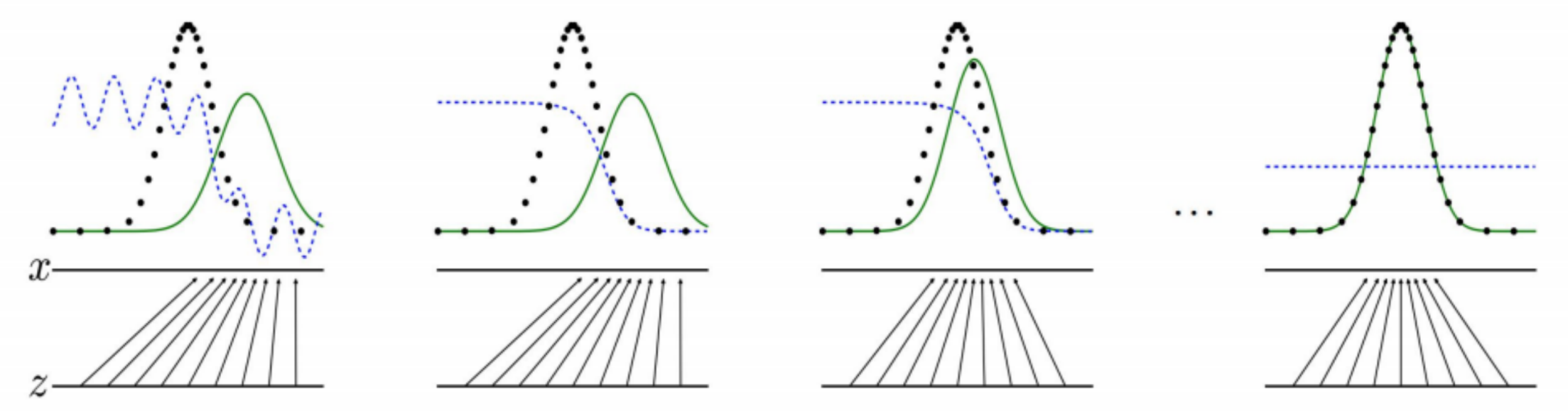
- One possible cost function for the generator: the opposite of the discriminator's

$$\mathcal{L}_G = -\mathcal{L}_D$$
$$= const + \mathbb{E}_{z\sim q(z)}[\log\Big(1 - D\big(G(z)\big)\Big)]$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

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

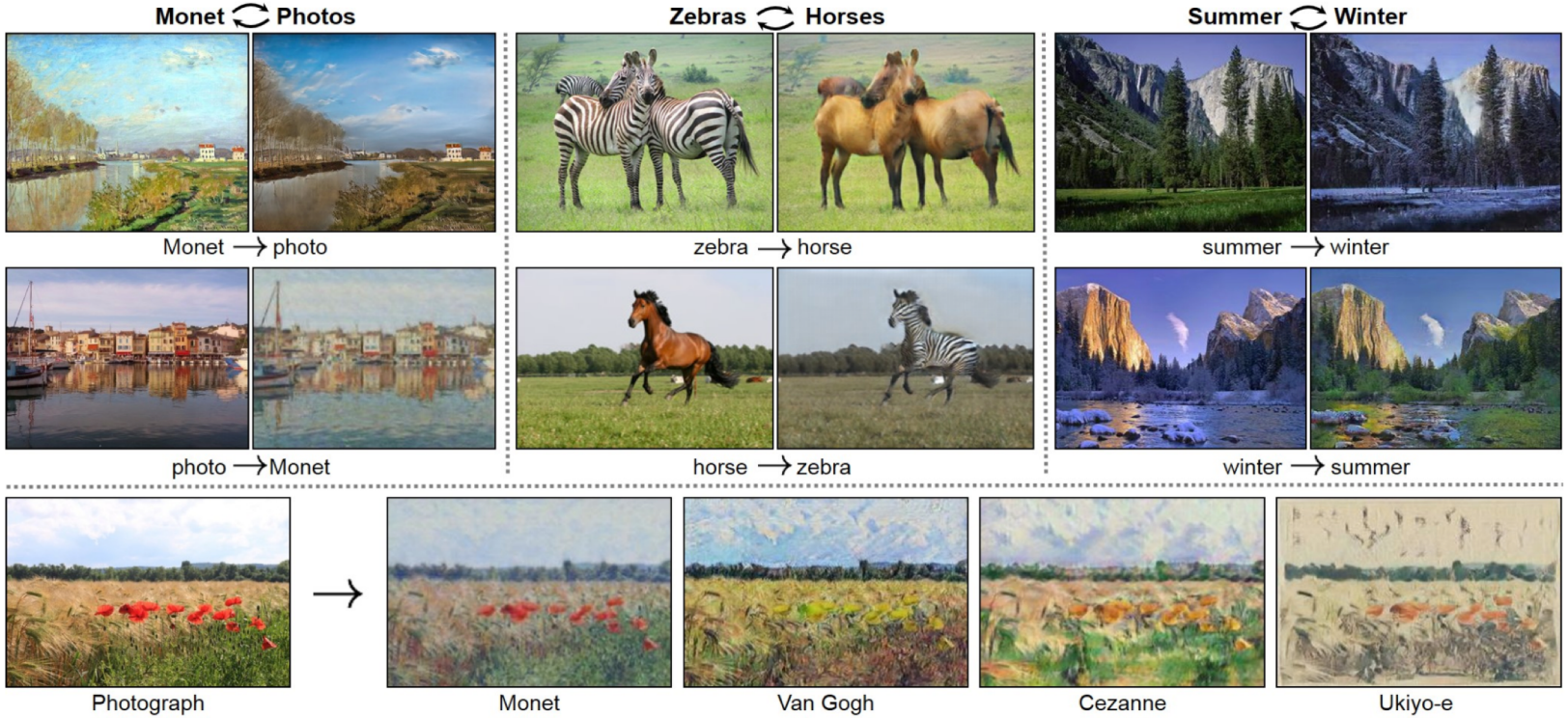
Alternating training of the Generator and Discriminator



Credit: Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014

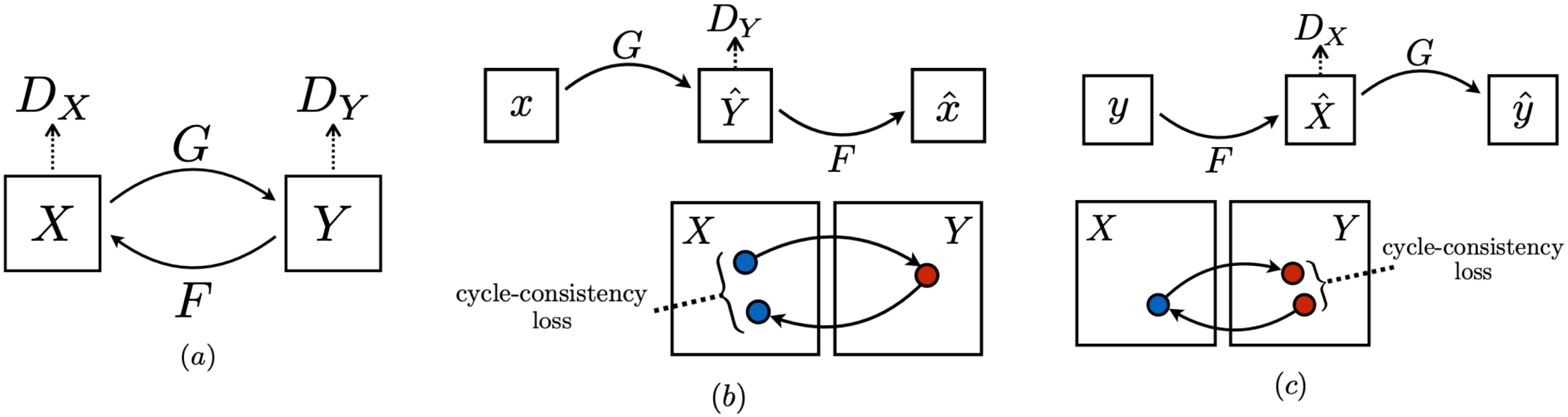
CycleGAN

- Style transfer problem: change the style of an image while preserving the content.



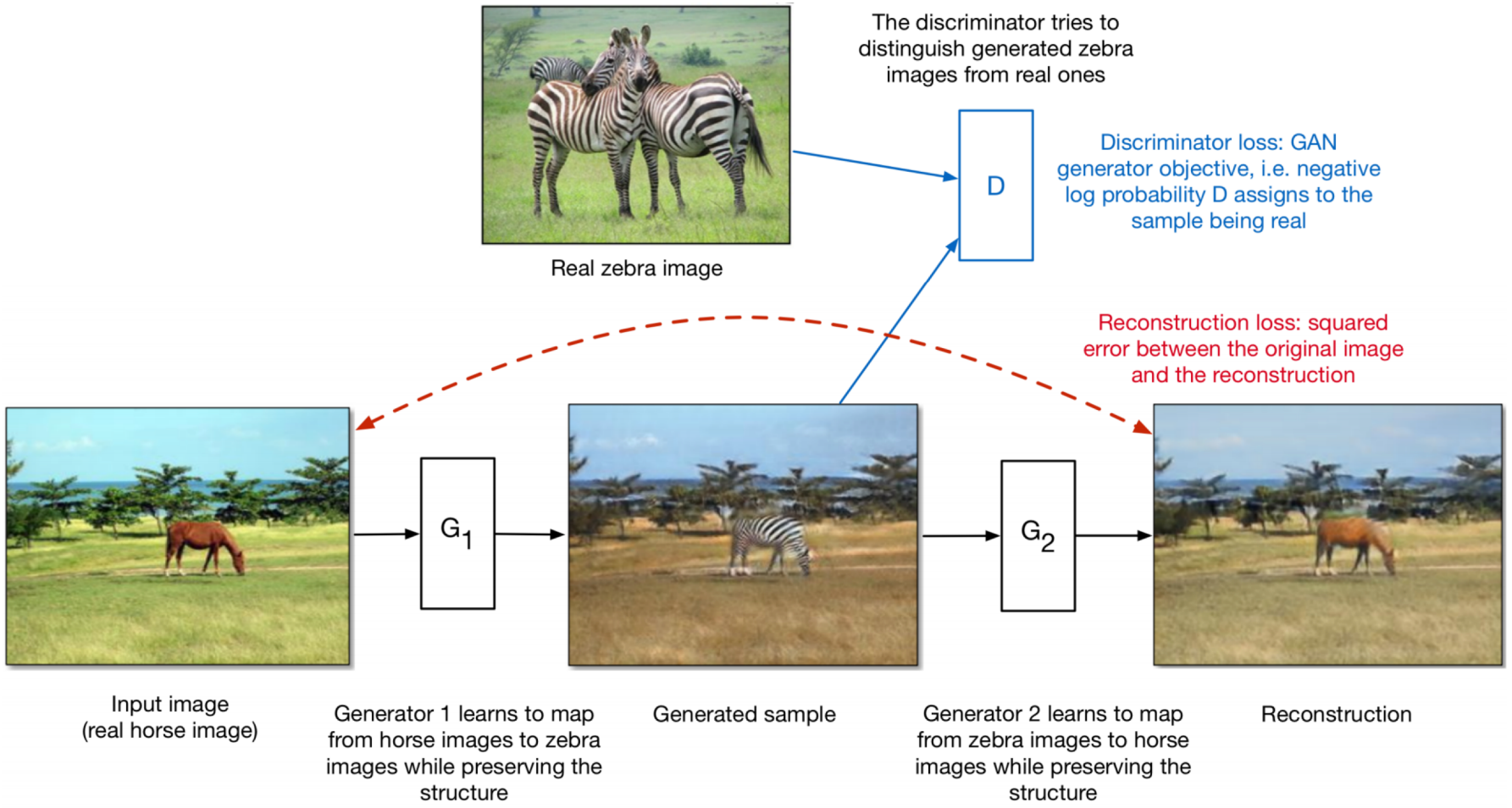
Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." ICCV, 2017.

CycleGAN



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." ICCV, 2017.

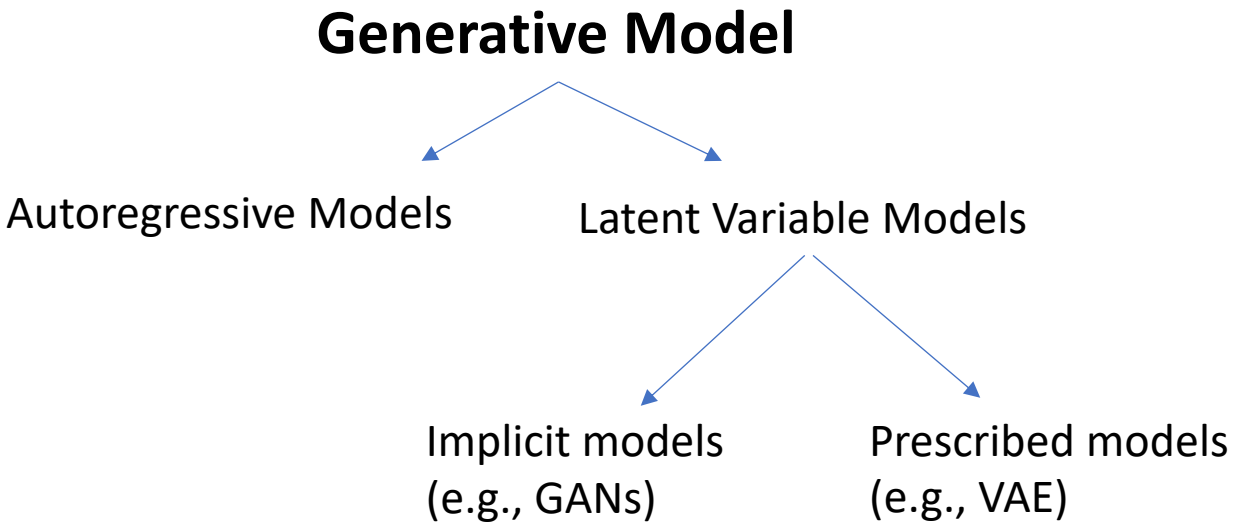
CycleGAN



Total loss = discriminator loss + reconstruction loss

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." ICCV, 2017.

Summary



Methods	Training	Likelihood	Sampling	Compression
Autoregressive models (e.g., PixelCNN)	Stable	Exact	Slow	No
Prescribed models (e.g., VAEs)	Stable	Approximate	Fast	Yes
Implicit models (e.g., GANs)	Unstable	No	Fast	No

Credit: http://docs.mlinpl.org/conference/2019/slides/jakub_tomczak_mlinpl2019.pdf

References

- EPFL course: https://fleuret.org/ee559/
- Van Den Oord, et.al. "Pixel recurrent neural networks." ICML. 2016.
- Stanford course: https://deepgenerativemodels.github.io
- Andriy Mnih, DeepMind's Deep Learning Lectures, 2020
- Kingma's PhD thesis
- Dinh et al., NICE: Non-linear Independent Components Estimation, ICLR 2015