Generative Models

Michal Drozdzal
mdrozdzal@fb.com

The gif from the first slide is based on: https://youtu.be/X0xxPcy5Gr4

When preparing the slides, I used original papers and the following sources:

1. Ian Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks
   https://arxiv.org/pdf/1701.00160.pdf
2. Francois Fleuret : Deep Learning Course, https://fleuret.org/dlc/
3. Jakob Verbeek: Deep learning for generative modeling,
   http://lear.inrialpes.fr/~verbeek/tmp/DL.jjv.pdf
4. OpenAI blog on generative models: https://blog.openai.com/generative-models/
5. Shakir Mohamed: Building Machines that Imagine and Reason
   https://drive.google.com/file/d/0B_wzP_JlVFcKMnFMNnAtYTVkV28/view
6. Aaron Courville: Generative Models II
   https://drive.google.com/file/d/0B_wzP_JlVFcKQ21udGpTSkh0aVk/view
7. Fei-Fei Li & Justin Johnson & Serena Yeung: Generative Models
   http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Index

Image generation

volcanos

monasteries

ants

Generator

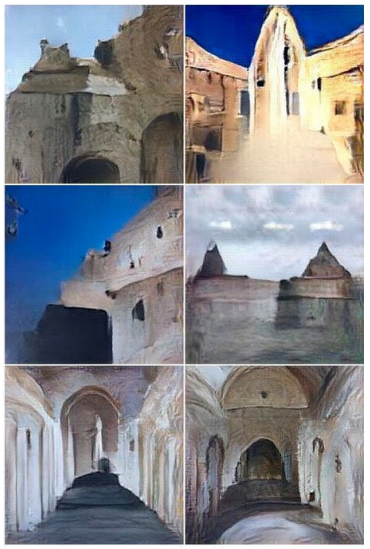
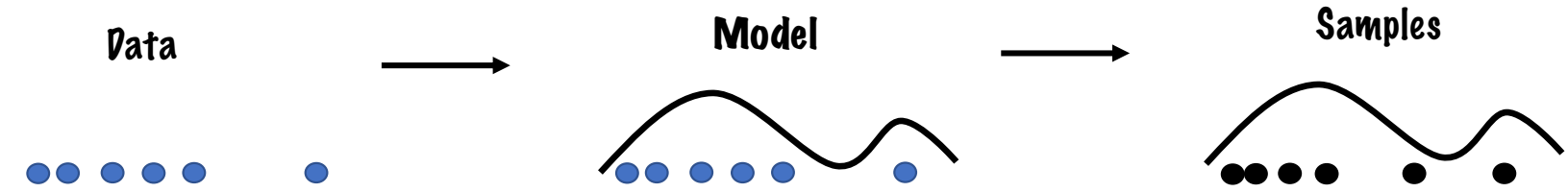


Image generation - density fitting



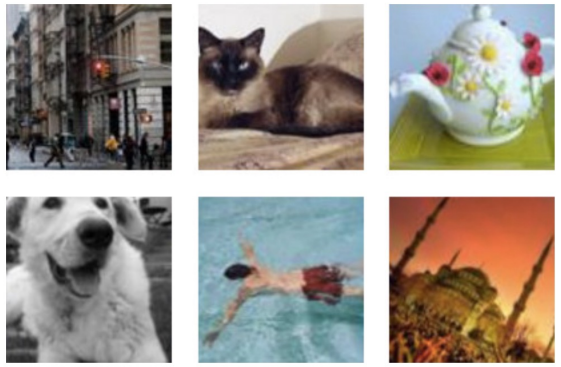


Image generation

- Image generation can be both class conditional and unconditional.
- Image generation is generally seen as unsupervised/self-supervised.
- It can be seen as one-to-many (one class many images) or none-to-many (unconditional setup).
  - Comparing to other CV tasks:
    - Image classification - many-to-one mapping (many images map to a single class).
    - Image reconstruction - one-to-one mapping (one image maps to a single image).
- We will focus on the unconditional case.

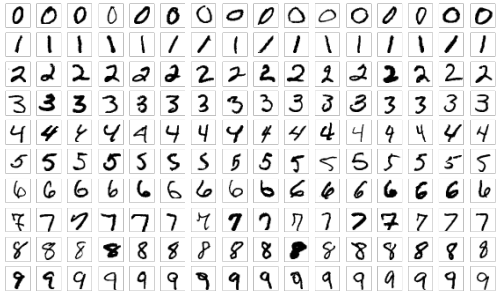
What are the applications for generative models?

In this class, we will focus on image generation.
Image generation has many applications:

Compression
Denoising
Inpainting
Texture synthesis
Semi-supervised learning
Unsupervised feature learning
Creativity
...

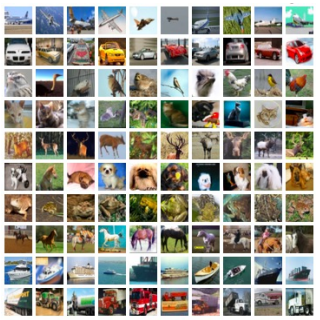
Datasets/Benchmarks

MNIST



CelebA

CIFAR



SVHN

LSUN Bedrooms

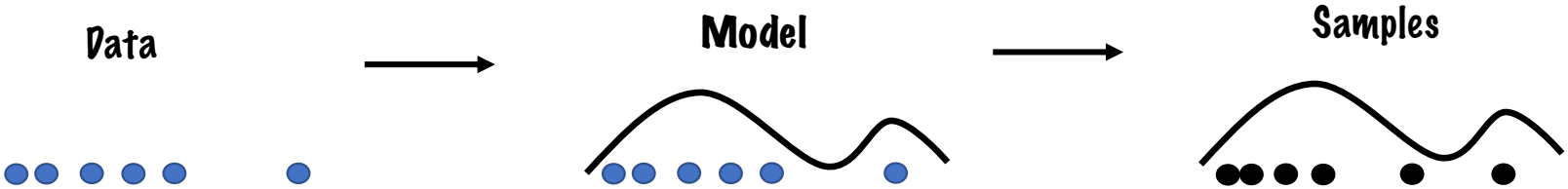
Imagenet

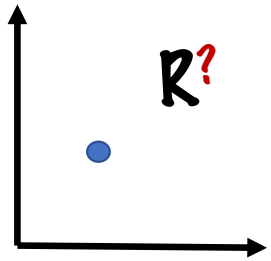
Evaluation

It's complicated...

Let's come back to it at the end of the class...

What makes image generation hard? Data dimensionality



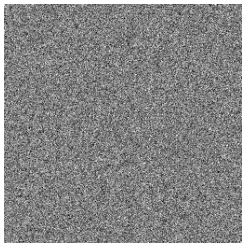


256 x 256 pixels RGB images

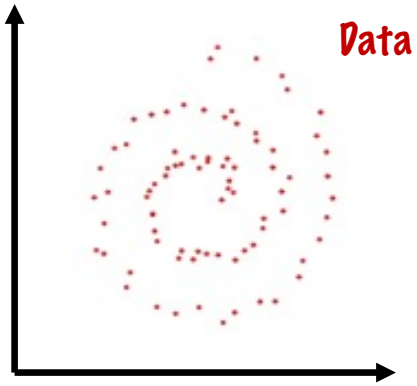
Each image is in R^{196608}

Density estimation in high dimensional spaces is hard....

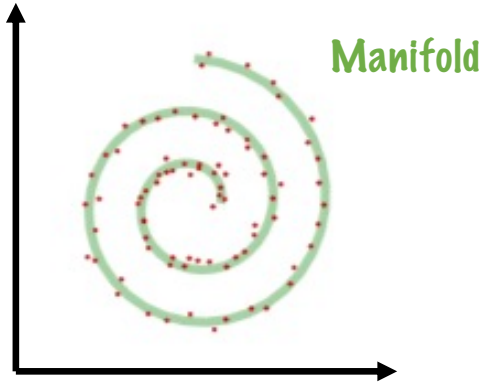
Manifold hypothesis



A sample from original space



Original space



Original space

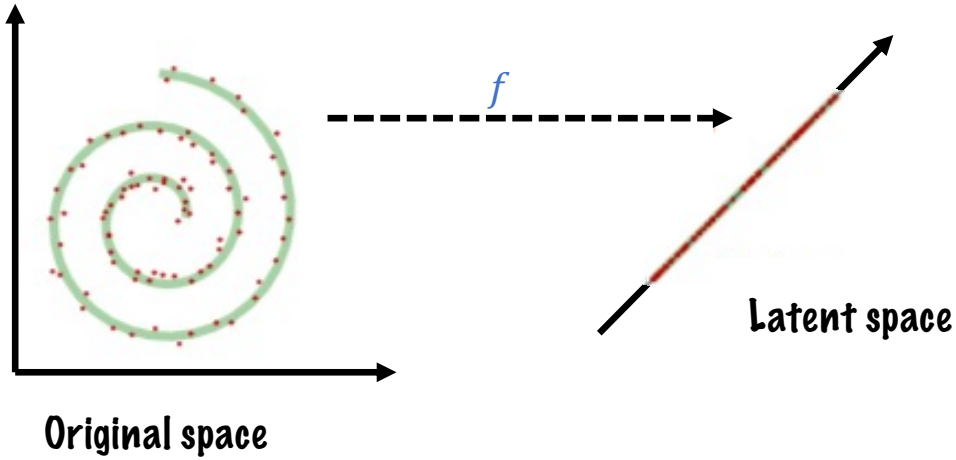
A sample from manifold

Francois Fleuret : Deep Learning Course, https://fleuret.org/dlc/

Latent space

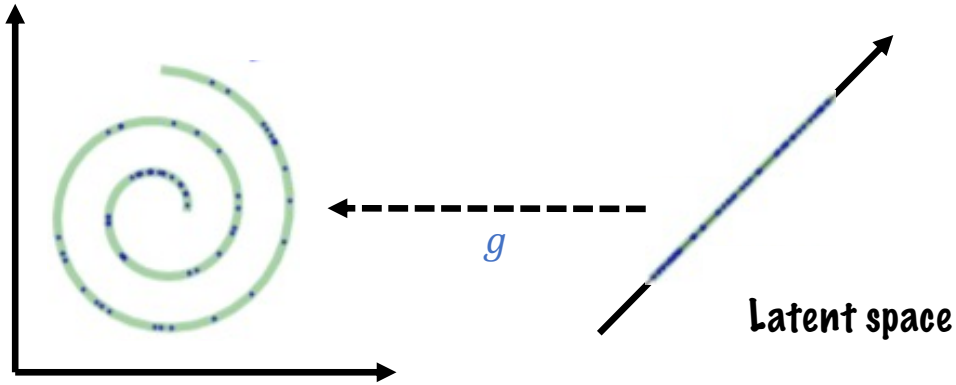
f is a neural network

- encoder (network)
- inference (network)
- recognition (network)



Francois Fleuret : Deep Learning Course, https://fleuret.org/dlc/

Latent space

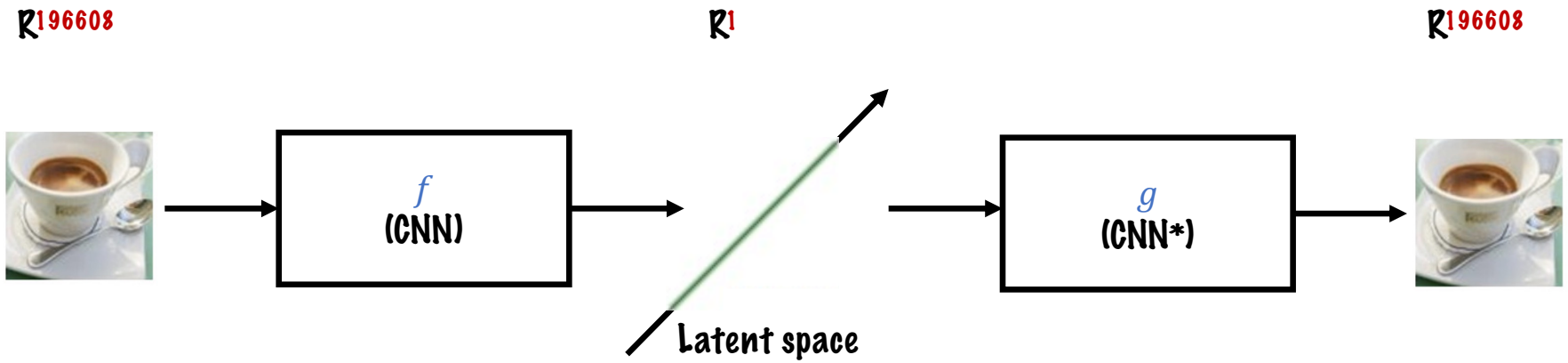


g is a neural network

- decoder (network)
- generative (network)

Francois Fleuret : Deep Learning Course, https://fleuret.org/dlc/

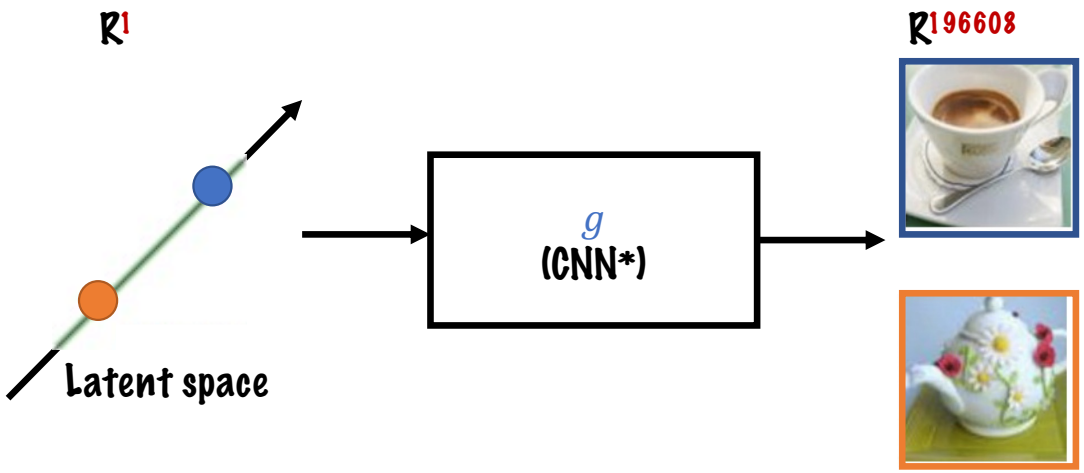
Encoder and Decoder networks (e.g. Autoencoder)



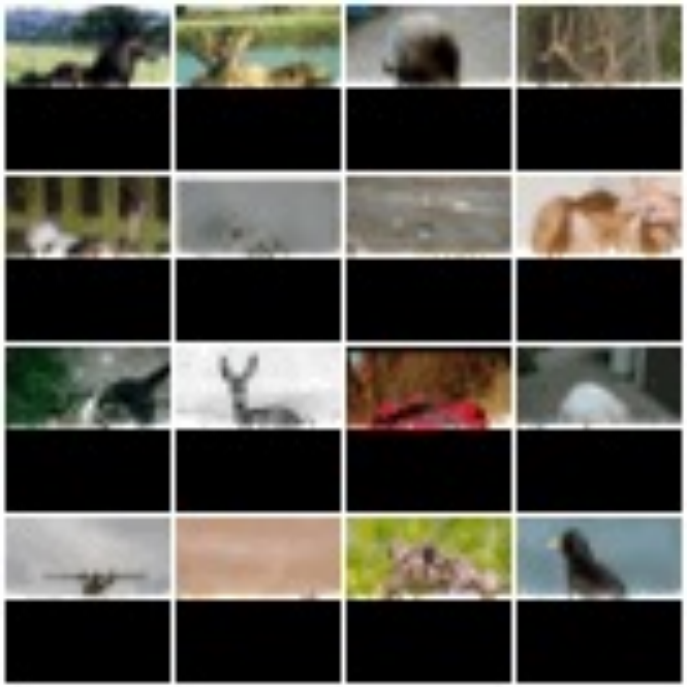
- Convolutional layers
- Poolings
- Non-linearities
- Fully connected layers
- Maybe some extras...

- Convolutional layers
- ~~Poolings~~
- Non-linearities
- Fully connected layers
- Maybe some extras...

Generator (with latent space)



Generator (without latent space)



Autoregressive approaches use previously generated pixels to obtain new ones.

Why this would be a good idea?
Formulate image generation as multiple generations in R^256 (number of values in each channel in RGB images).

Image generation vs retrieval

Generation:

- Incorporates learning
- Use a large dataset to learn a density model
- Can generate novel images (not in the dataset)
- Model data density

Retrieval:

- Might incorporate learning (e.g., embedding)
- Use a large dataset to retrieve images
- Cannot produce images that are not in the dataset
- Search based (e.g., knn)

Intro: summary

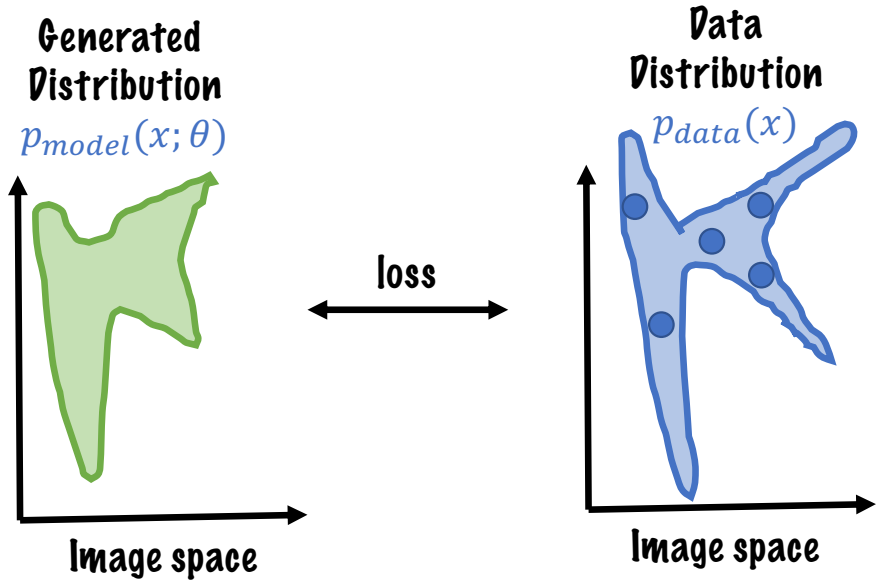
- Image generation is about systems that produce nice images
- Image generation approach the problem by learning data distribution
- These systems can create novel images
- The main challenge when building such systems is image dimensionality
- We will cover two modeling approaches that can handle image dimensionality: latent variable models and autoregressive models

Index

Notation

$x^{(i)}$	Data point (e. g. single image)
$X = [x^{(1)}, x^{(2)}, \cdots, x^{(N)}]^T$	Dataset
$p(x)$	Probability distribution
$p(x\|y)$	Conditional probability distribution
$p(x, y)$	Joint probability distribution
$p(x; \theta)$	Probability distribution parametrized by θ
$\mathbb{E}[Y]$	Expected value

Generative models with maximum likelihood Intuition



Adjust model parameters (θ) to match the model distribution $p_{model}(x;\theta)$ with the data distribution $p_{data}(x)$.

Image adapted from OpenAI blog on generative models: https://blog.openai.com/generative-models/

Maximum Likelihood

Data

$$X = [x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}, x^{(6)}]^T$$

$$x^{(i)} \sim p_{data}$$

Model

$$p_{model}(X;\theta)$$

Goal of learning (maximum likelihood):

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, p_{model}(X;\theta)$$

We can rewrite it as:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_X \log p_{model}(x^{(i)};\theta)$$

Derivation:

$$p_{model}(X;\theta) = \prod_{i=1}^{N} p_{model}\left(x^{(i)};\theta\right)$$

Log likelihood:

$$\log p_{model}(X;\theta) = \sum_{i=1}^{N} \log p_{model}\left(x^{(i)};\theta\right)$$

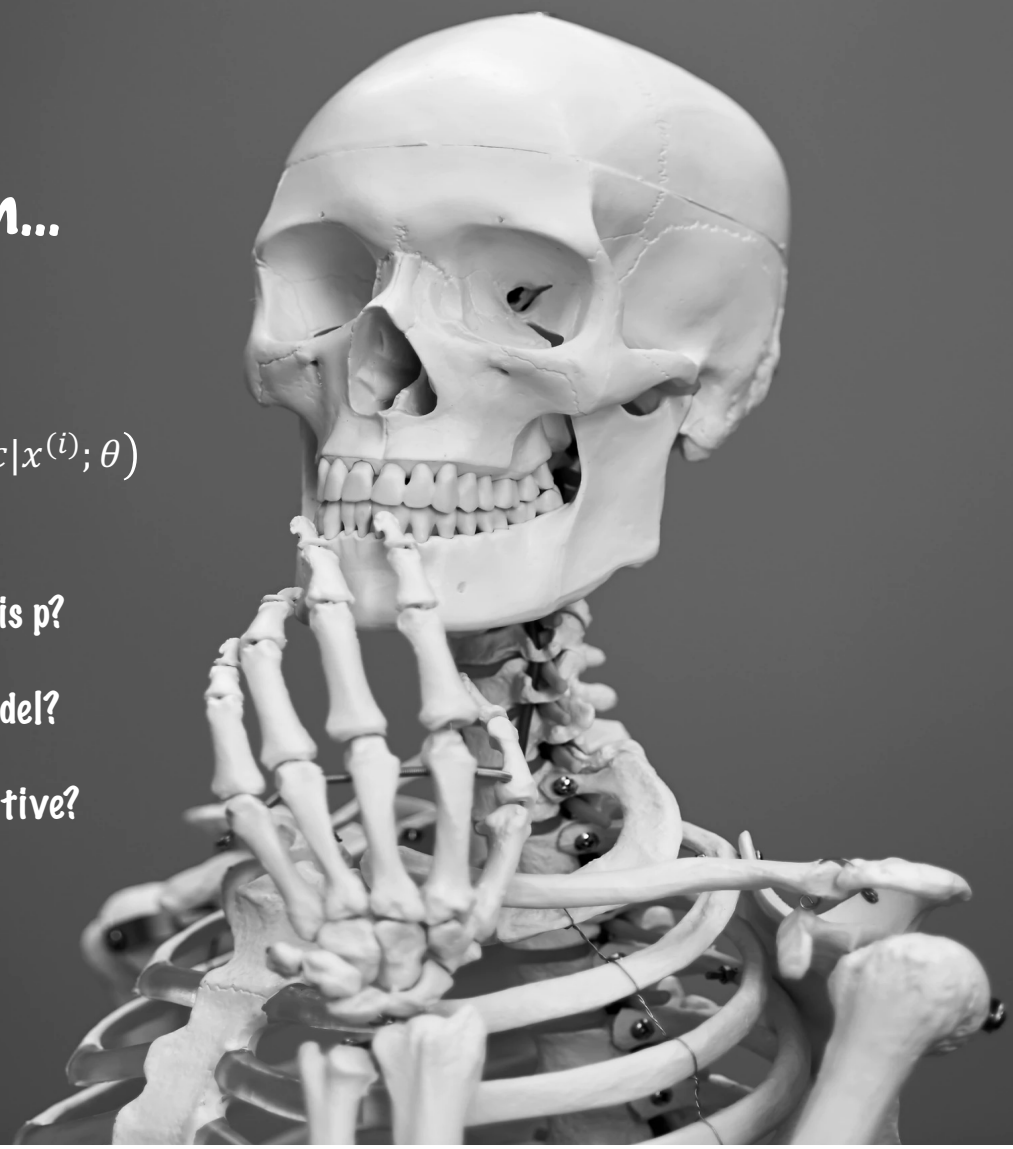
Small review
Image classification...
P(c | x)

$$\log p_{model}(C|X;\theta) = \sum_{i=1}^{N} \log p_{model}\left(c|x^{(i)};\theta\right)$$

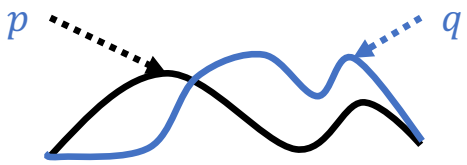
What kind of distribution is p?

What is the structure of model?

What is the training principle/objective?



KL divergence



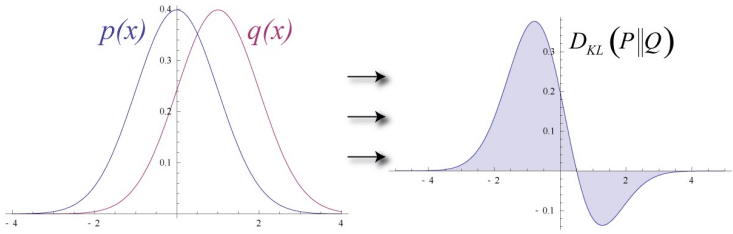
Kullback-Leibler (KL) divergence:

$$D_{KL}(p||q) = \mathbb{E}_{x\sim p}\left[\log\frac{p(x)}{q(x)}\right]$$

Kullback-Leibler (KL) divergence is asymmetric:

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

If $D_{KL}(p||q) = 0$ **then** $p = q$



Original Gaussian PDF's KL Area to be Integrated

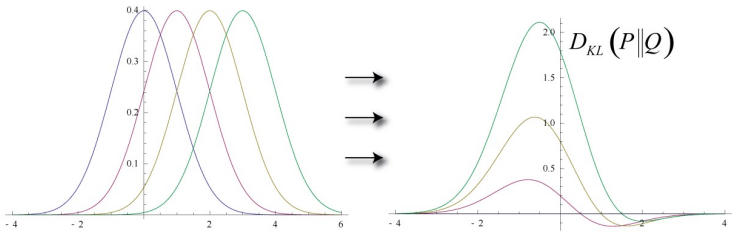
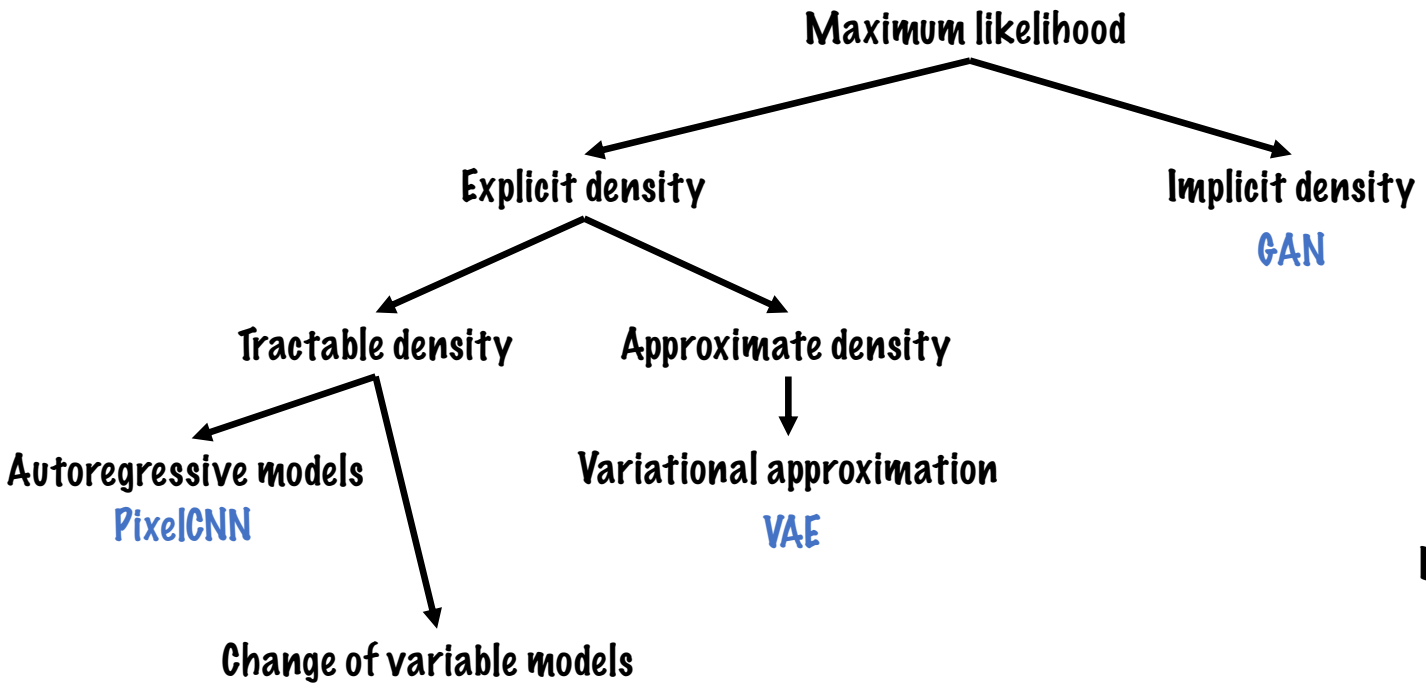


Image from Wikipedia

Maximizing likelihood is equivalent to minimization of the KL divergence between data generating distribution $p_{data}(x)$ **and the model** $p_{model}(x)$**.**

Maximum likelihood generative models



Latent variable

VAE

GAN

Figure adapted from https://arxiv.org/pdf/1701.00160.pdf /

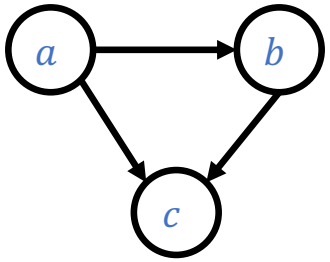
Index

Graphical model

Use a graph to express structure between random variables.

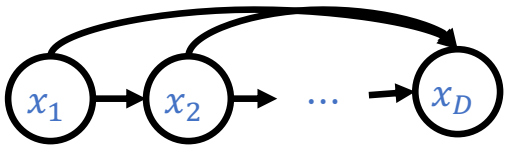
Example: Given 3 random variables a, b, c draw a graphical model of $p(a, b, c)$

Using product rule of probability, we can write $p(a, b, c) = p(c|a, b)p(a, b) = p(c|a, b)p(b|a)p(a)$



Autoregressive models

$$p_{model}(X;\theta) = \prod_{i=1}^{N} p_{model}\left(x^{(i)};\theta\right)$$

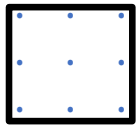


$$p_{model}\left(x_{1:D}^{(i)};\theta\right) = p_{model}\left(x_1^{(i)};\theta\right)\prod_{j=2}^{D} p_{model}\left(x_j^{(i)}\middle|x_{<j}^{(i)};\ \theta\right)$$

$$x_{<i} = x_1, \cdots, x_{i-1}$$

Notation:
x_i - dimension in x
D - dimension of x

How do we apply this to images? Impose some kind of ordering.



3 x 3 image

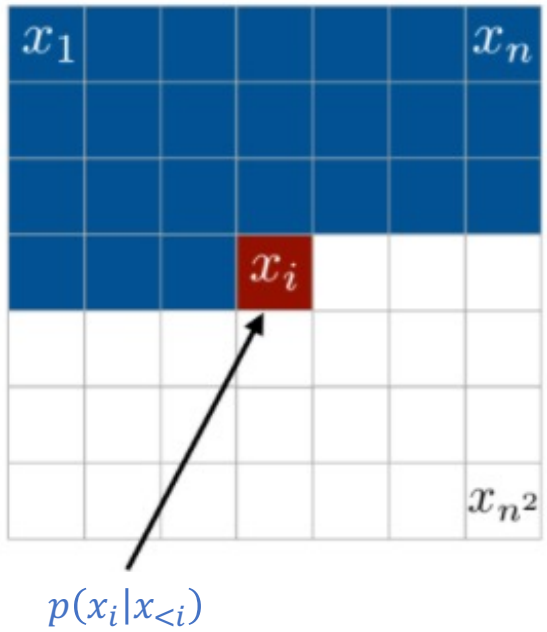
x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

3 x 3 image with an ordering

Autoregressive models

Context:

$x_{<i} = x_1, \cdots, x_{i-1}$

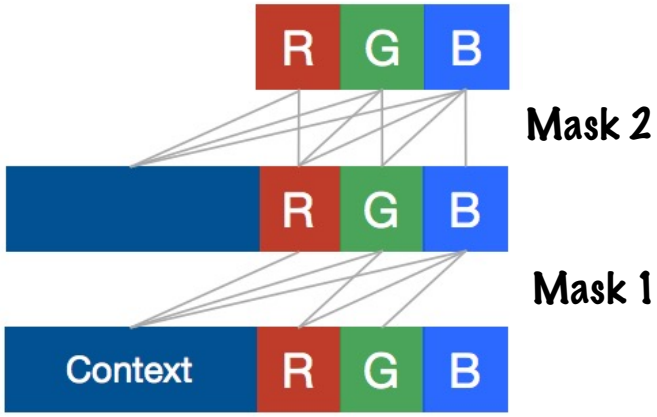


One more problem, images are RGB.

We need to order channels too:

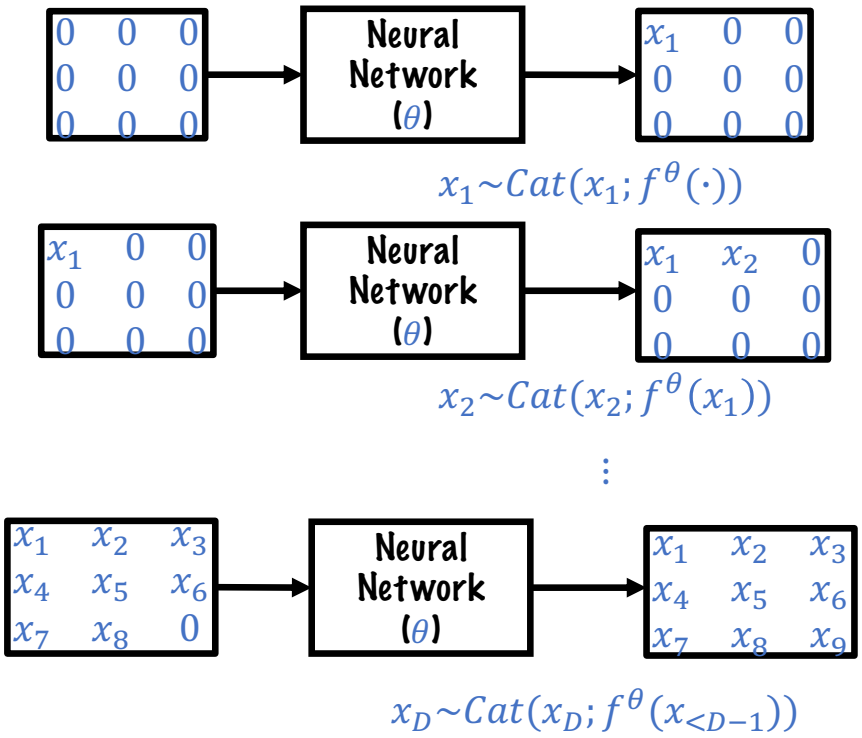
$$p(x_i|x_{<i}) = p(x_{i,R}|x_{<i})p(x_{i,G}|x_{i,R}, x_{<i})p(x_{i,B}|x_{i,G}x_{i,R}, x_{<i})$$

The order on channels is imposed with masks

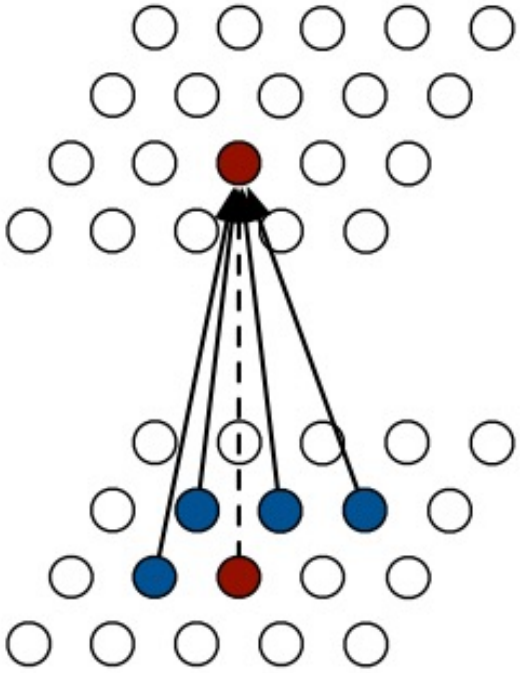


Autoregressive models

Image generation (sampling):



Autoregressive models Architectures



PixelCNN

LSTM

BiLSTM

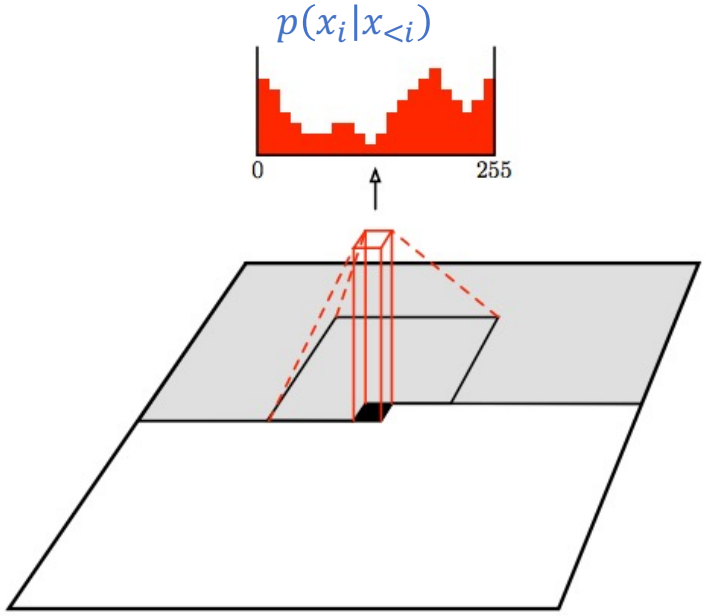
In general, RNNs

PixelCNN

Use masked receptive field as context -> Use masked convolution kernels

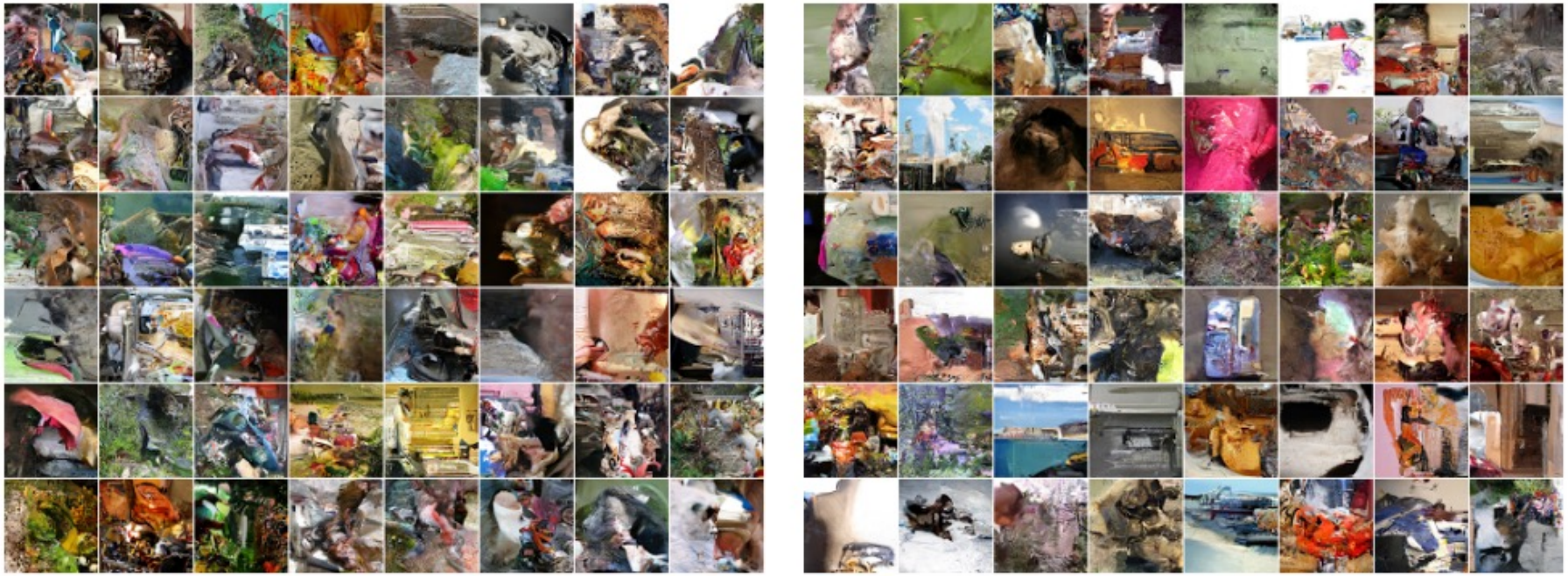
5 x 5 masked convolution kernel

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



PixelCNN Results

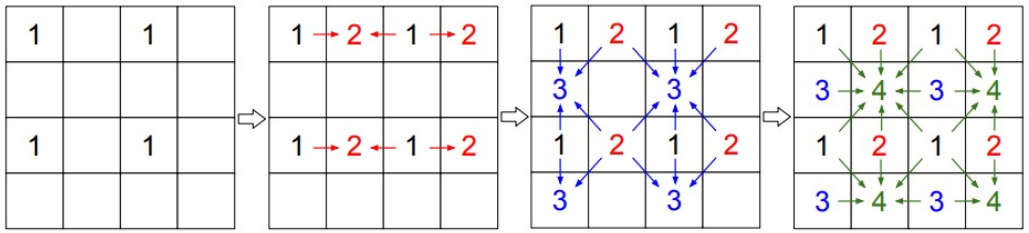
Samples from the model.

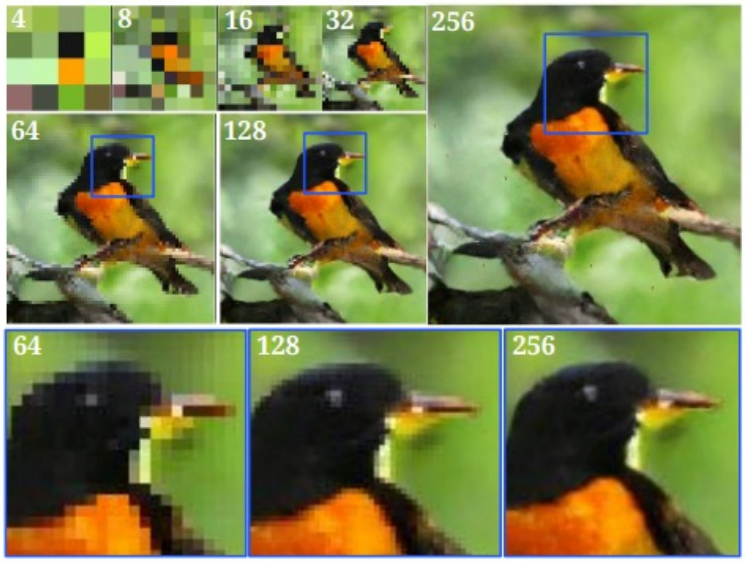


The model captures well local structure, not very well global structure.

Speed up the image generation

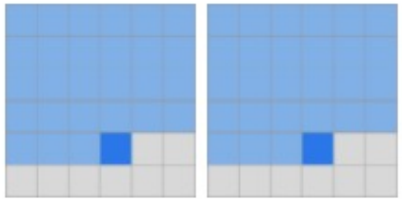
Multiscale generation!



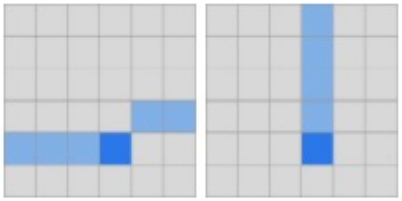


Sparse Transformers

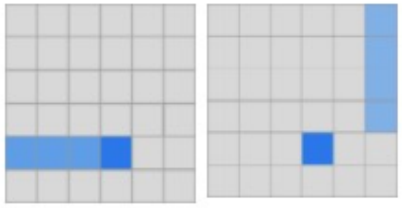
Masked Attention: a good way to implement neighborhood information



Normal transformer

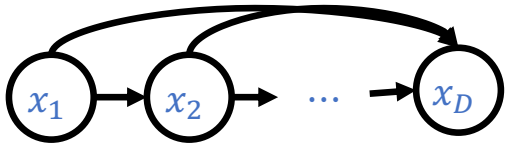


Strided attention



Fixed attention

PixelCNN conclusions



\+

Can explicitly compute likelihood
Nice looking samples

\-

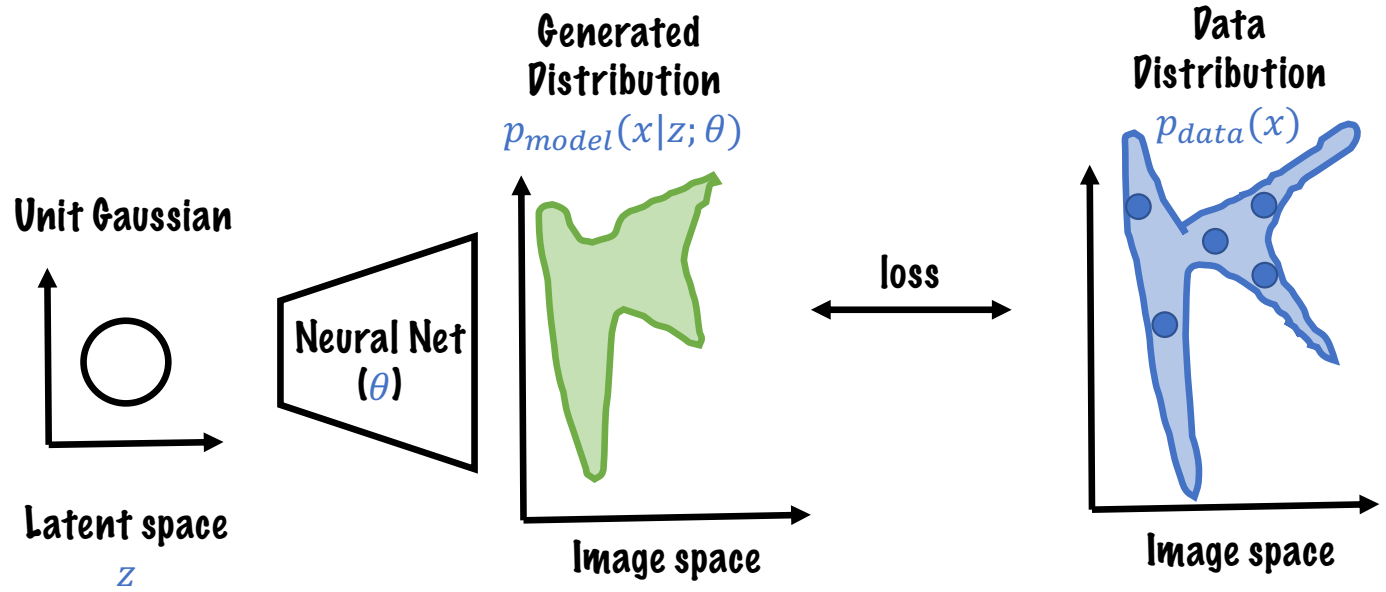
Sequential generation (slow)
Order sensitive

?

What is the best way of imposing order...

Index

Generative models with latent variables Intuition



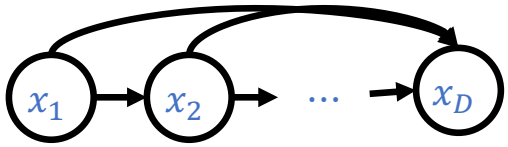
Adjust model parameters (θ) to match the model distribution $p_{model}(x|z;\theta)$ with the data distribution $p_{data}(x)$.

Image adapted from OpenAI blog on generative models: https://blog.openai.com/generative-models/

Variational inference

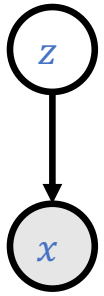
$$p_{model}(X;\theta) = \prod_{i=1}^{N} p_{model}\left(x^{(i)};\theta\right)$$

Autoregresive models



$$p_{model}\left(x_{1:D}^{(i)};\theta\right) = p_{model}\left(x_1^{(i)};\theta\right)\prod_{j=2}^{D} p_{model}\left(x_j^{(i)}\middle|x_{<j}^{(i)};\ \theta\right)$$

Latent Variable models



Hard to compute...

$$p_{model}\left(x^{(i)}\right) = \int p_{model}\left(x^{(i)},z\right)dz$$

Variational inference: use an approximation to transform the integral into an expectation over a simple, known distribution.

Evidence Lower Bound (ELBO)

$$\log p\left(x^{(i)}\right) = \log \int p\left(x^{(i)}, z\right) dz = \log \int p(x^{(i)}, z) \frac{q(z)}{q(z)} dz =$$

$$= \log \mathbb{E}_q \frac{p(x^{(i)}, z)}{q(z)} \geq \mathbb{E}_q \left[\log \frac{p\left(x^{(i)}, z\right)}{q(z)}\right] = \mathcal{L}(x^{(i)})$$

Evidence Lower Bound (ELBO)

$\log p(x^{(i)})$ - **marginal log likelihood (evidence)**

$q(z)$ - **variational distribution**

$\log \mathbb{E}[Y] \geq \mathbb{E}[\log Y]$ - **Jensen's inequality**

$p(z|x^{(i)})$ - **posterior**

We can also write:

$$\log p\left(x^{(i)}\right) \geq \log p\left(x^{(i)}\right) - D_{KL}\left(q(z)||p(z|x^{(i)})\right) = \mathcal{L}\left(x^{(i)}\right)$$

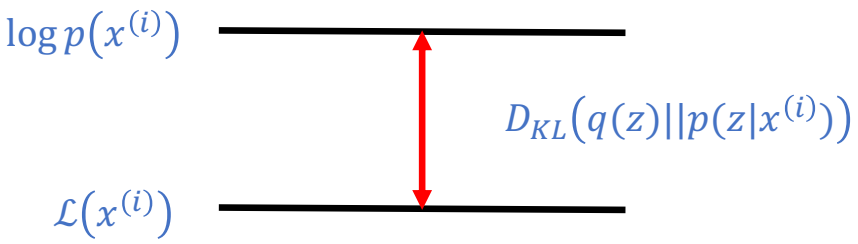
For an interested reader:

Note that: $p(z|x) = \frac{p(x,z)}{p(x)}$

$$D_{KL}\left(q(z)||p(z|x)\right) = \mathbb{E}_q \left[\log \frac{q(z)}{p(z|x)}\right] = \mathbb{E}_q \left[\log \frac{q(z)p(x)}{p(x,z)}\right] = \mathbb{E}_q \left[\log \frac{q(z)}{p(x,z)}\right] + \log p(x) = -\mathbb{E}_q \left[\log \frac{p(x,z)}{q(z)}\right] + \log p(x)$$

ELBO

$$\mathcal{L}(x^{(i)}) = \log p(x^{(i)}) - D_{KL}(q(z)||p(z|x^{(i)}))$$



The tightness of ELBO depends on how well the variational distribution approximates the true posterior.

How do we get q? (Amortized inference)

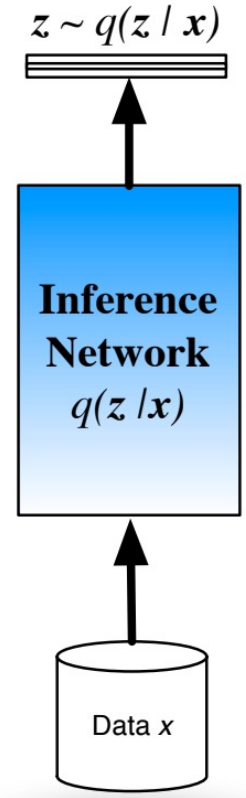
$$\mathbb{E}_q\left[\log \frac{p(x^{(i)}, z)}{q(z)}\right]$$

Vs.

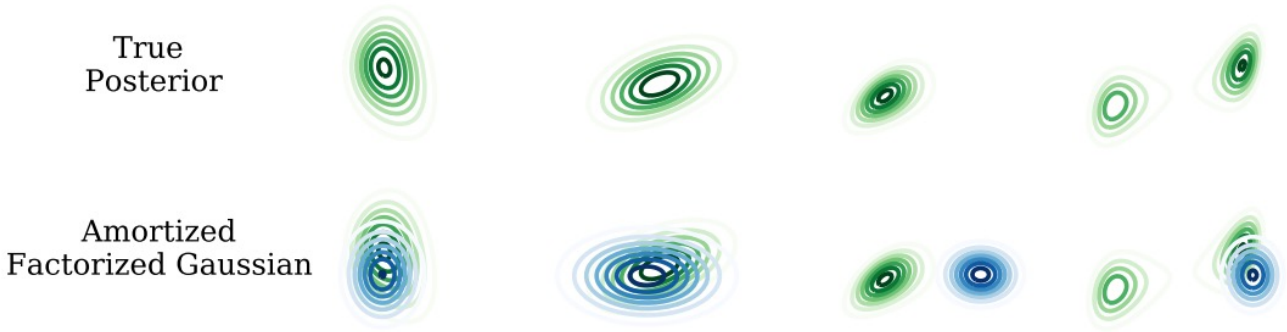
$$\mathbb{E}_{q(z|x)}\left[\log \frac{p(x, z)}{q(z|x)}\right]$$

Estimation of q is costly (e. g. EM)

Amortize -> spread the inference cost over the whole dataset

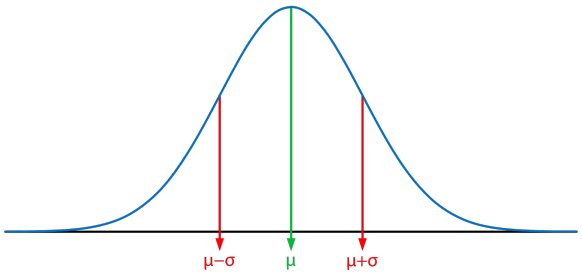


Costs of amortized inference

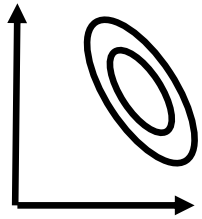


Gaussian distribution

$\mathcal{N}(\mu, \sigma)$



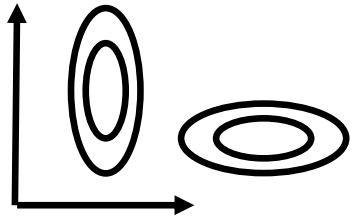
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$



$\boldsymbol{\mu} = [\mu_1, \mu_2]^T$

$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$

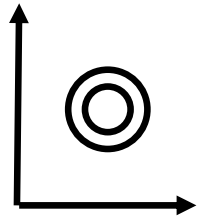
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$



$\boldsymbol{\mu} = [\mu_1, \mu_2]^T$

$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$

$\mathcal{N}(\boldsymbol{\mu}, \sigma\mathbf{1})$

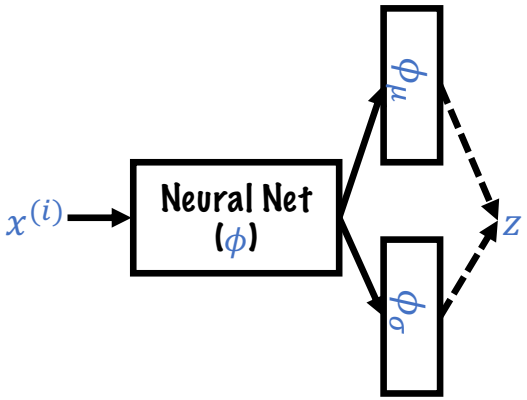


$\boldsymbol{\mu} = [\mu_1, \mu_2]^T$

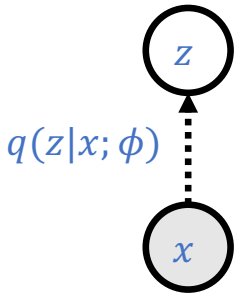
$\Sigma = \begin{bmatrix} \sigma & 0 \\ 0 & \sigma \end{bmatrix}$

$\mathbf{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

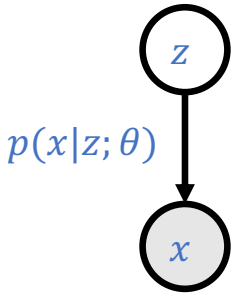
Variational Autoencoders (VAE)

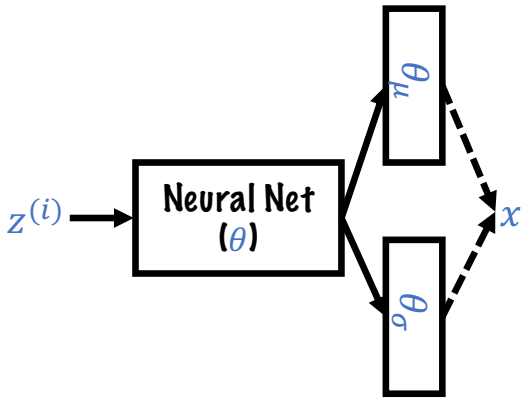


Inference model



Generative model





$$q(z|x^{(i)};\phi) = \mathcal{N}\left(z; \boldsymbol{f}_{\boldsymbol{\phi}}^{\boldsymbol{\mu}}(x^{(i)}), \boldsymbol{f}_{\boldsymbol{\phi}}^{\boldsymbol{\sigma}}(x^{(i)})\right)$$

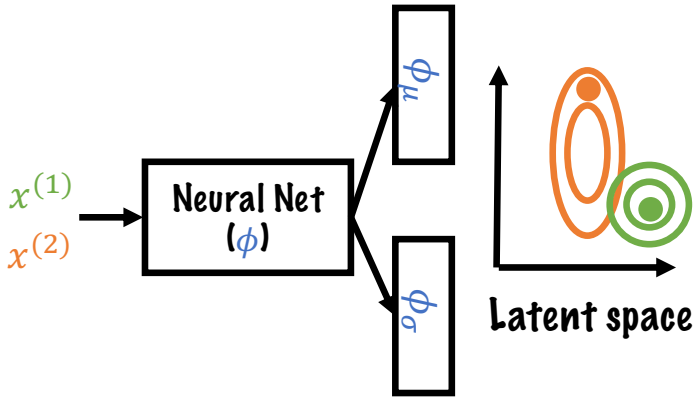
$x^{(i)}$- our data points: $x^{(i)} \sim p_{data}$

$$p\left(x|z^{(i)};\theta\right) = \mathcal{N}\left(x; \boldsymbol{g}_{\boldsymbol{\theta}}^{\boldsymbol{\mu}}(z^{(i)}), \boldsymbol{g}_{\boldsymbol{\theta}}^{\boldsymbol{\sigma}}(z^{(i)})\right)$$

$z^{(i)}$- comes from model prior $z^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ (generation)
and
$z^{(i)}$- comes from model posterior $z^{(i)} \sim q(z|x^{(i)};\phi)$ (training)

Variational Autoencoders (VAE)

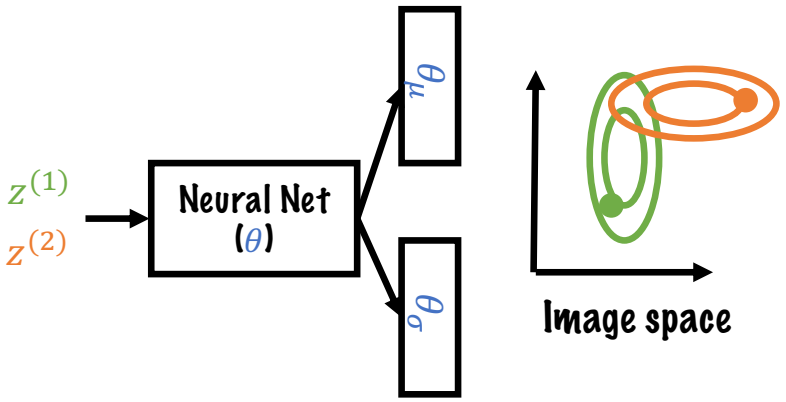
Inference model



$$q(z|x^{(i)};\phi) = \mathcal{N}\left(z; \boldsymbol{f}^{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(x^{(i)}), \boldsymbol{f}^{\boldsymbol{\sigma}}_{\boldsymbol{\phi}}(x^{(i)})\right)$$

$x^{(i)}$- our data points: $x^{(i)} \sim p_{data}$

Generative model



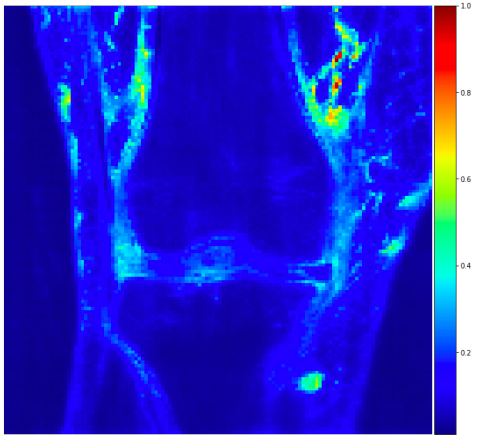
$$p(x|z^{(i)};\theta) = \mathcal{N}(x; \boldsymbol{g}^{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(z^{(i)}), \boldsymbol{g}^{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}(z^{(i)}))$$

$z^{(i)}$- comes from model prior $z^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ (generation)
and
$z^{(i)}$- comes from model posterior $z^{(i)} \sim q(z|x^{(i)};\phi)$ (training)

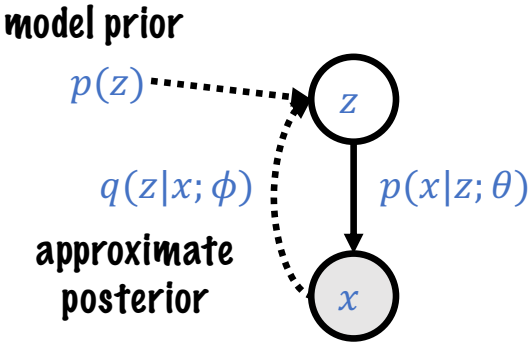
A visual example

Mean

Std.



Variational Autoencoders (VAE)



ELBO for variational autoencoder:

$$\mathcal{L}(x,\theta,\phi) = \mathbb{E}_{q(z|x;\,\phi)}\left[\log\frac{p(x,z;\,\theta)}{q(z|x;\,\phi\,)}\right] =$$

$$\underbrace{\mathbb{E}_{q(z|x;\phi)}[\log p(x|z;\theta)]}_{\text{Reconstruction}} \underbrace{-D_{KL}(q(z|x;\phi)||p(z))}_{\text{Regularization}}$$

Reconstruction -> high quality of x from z

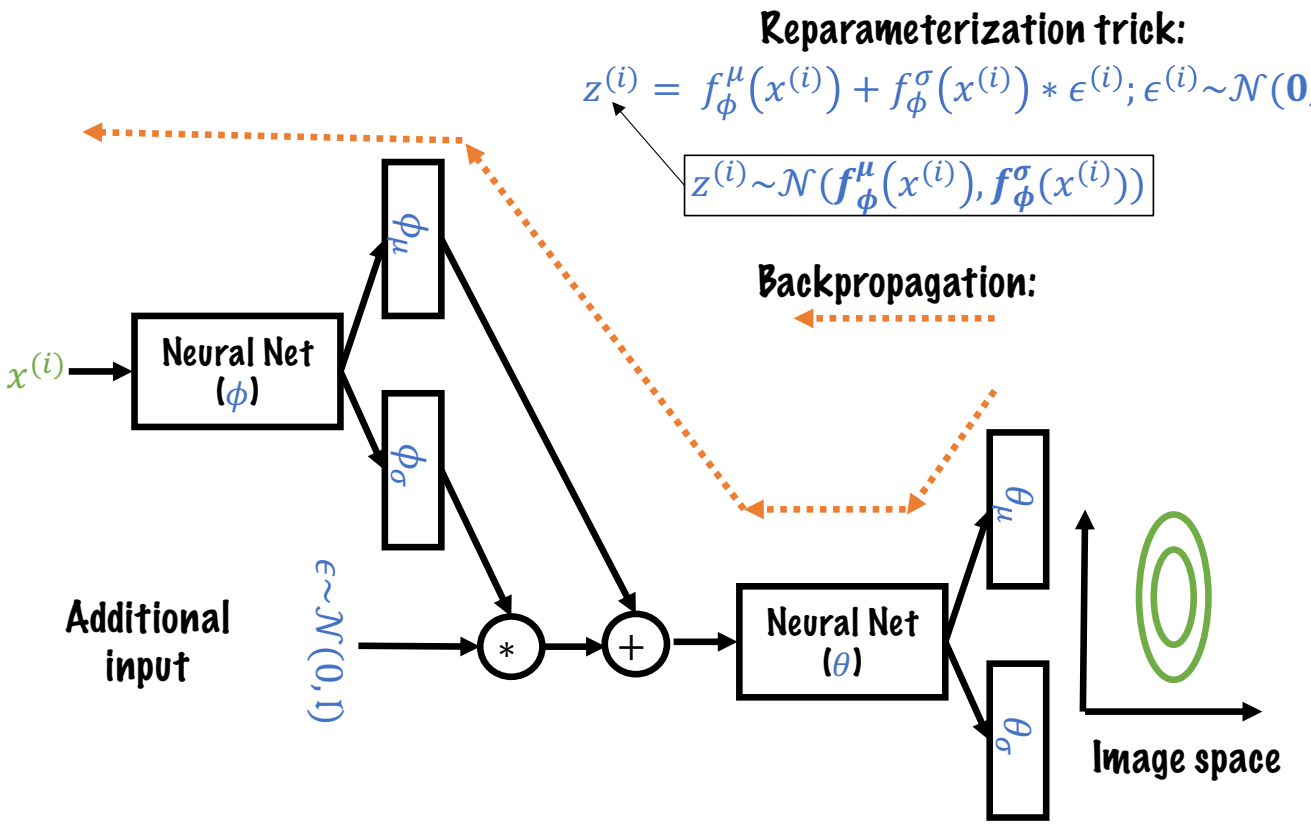
Regularization -> keeps the approximate posterior $q(z|x;\phi)$ close to the model prior $p(z)$

How to compute the expectation $\mathbb{E}_{q(z|x;\phi)}$?

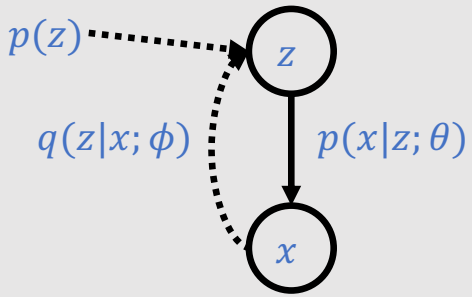
Draw samples from the approximate posterior $z^{(i)} \sim q(z|x;\phi)$ to approximate the expectation:

$$\mathbb{E}_{q(z|x;\phi)}[\log p(x|z;\theta)] \approx \frac{1}{S}\sum_{i=1}^{S}\log p(x|z^{(i)};\theta)$$

Variational Autoencoders (VAE)



Reminder:



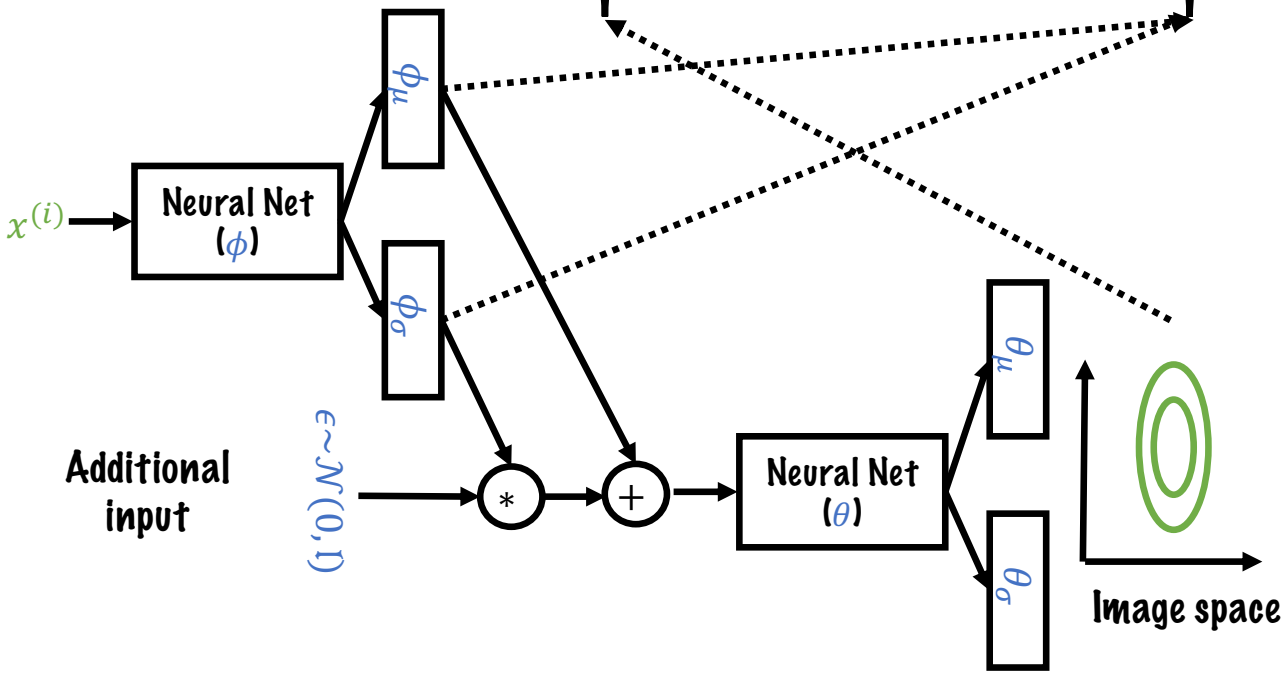
$$q(z|x^{(i)};\phi) = \mathcal{N}\left(z; f_\phi^\mu(x^{(i)}), f_\phi^\sigma(x^{(i)})\right)$$

$$p(x|z^{(i)};\theta) = \mathcal{N}(x; g_\theta^\mu(z^{(i)}), g_\theta^\sigma(z^{(i)}))$$

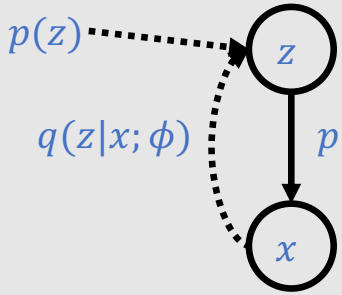
Variational Autoencoders (VAE)

Training objective:

$$\mathcal{L}(x^{(i)}, \theta, \phi) = \frac{1}{S}\sum_{j=1}^{S} \log p(x|f_\phi^\mu(x^{(i)}) + f_\phi^\sigma(x^{(i)}) * \epsilon^{(j)}; \theta) - D_{KL}(q(z|x^{(i)}; \phi)||p(z))$$



Reminder:



$$q(z|x^{(i)};\phi) = \mathcal{N}\left(z; \boldsymbol{f_\phi^\mu}(x^{(i)}), \boldsymbol{f_\phi^\sigma}(x^{(i)})\right)$$

$$p(x|z^{(i)};\theta) = \mathcal{N}(x; \boldsymbol{g_\theta^\mu}(z^{(i)}), \boldsymbol{g_\theta^\sigma}(z^{(i)}))$$

Gaussian log likelihood vs MSE

$$\log p\left(x^{(i)}|z\right) = -\frac{L}{2}\log 2\pi \; - \frac{1}{2}\sum_{l=1}^{L}\left(\log \boldsymbol{\sigma}_l - \frac{\left(x^{(i)}{}_l - \boldsymbol{\mu}_l\right)^2}{\boldsymbol{\sigma}_l}\right)$$

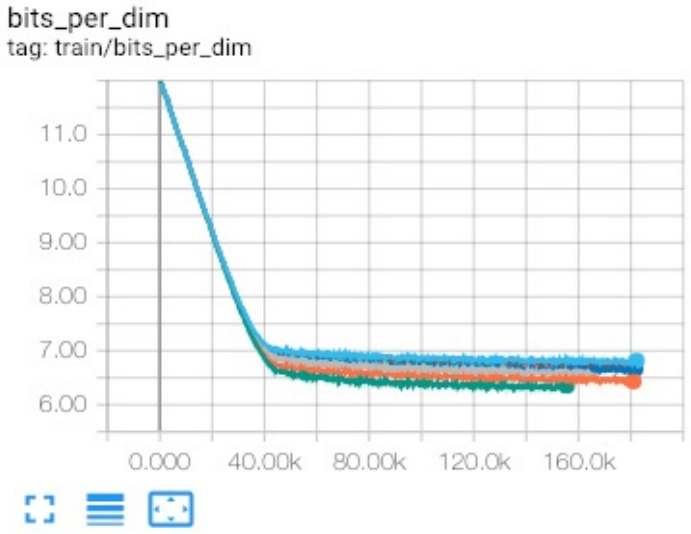
$$\text{MSE}(x^{(i)}) = \frac{1}{L}\sum_{l=1}^{L}\left(x^{(i)}{}_l - \boldsymbol{\mu}_l\right)^2$$

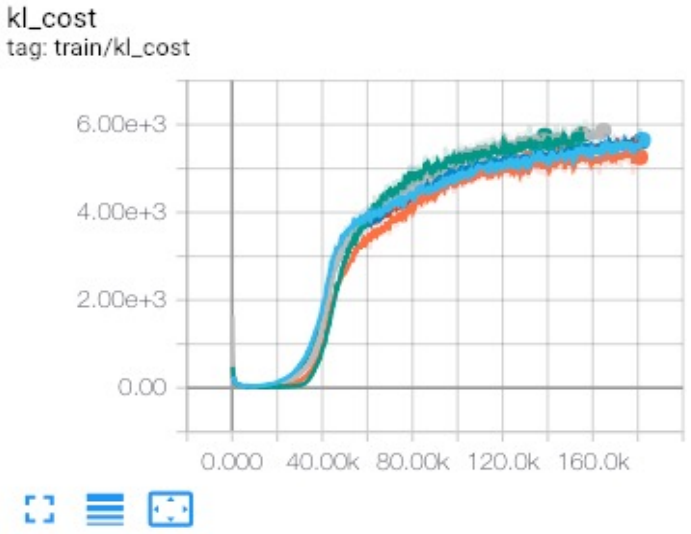
When minimizing MSE is equivalent to maximizing Gaussian LL?

Training of a VAE

$$\mathcal{L}(x, \theta, \phi) = \mathbb{E}_{q(z|x;\,\phi)}\left[\log \frac{p(x, z;\,\theta)}{q(z|x;\phi\,)}\right] =$$
$$\mathbb{E}_{\mathrm{q(z|x;\phi)}}[\log p(x|z;\theta)] - D_{KL}(q(z|x;\phi)||p(z))$$

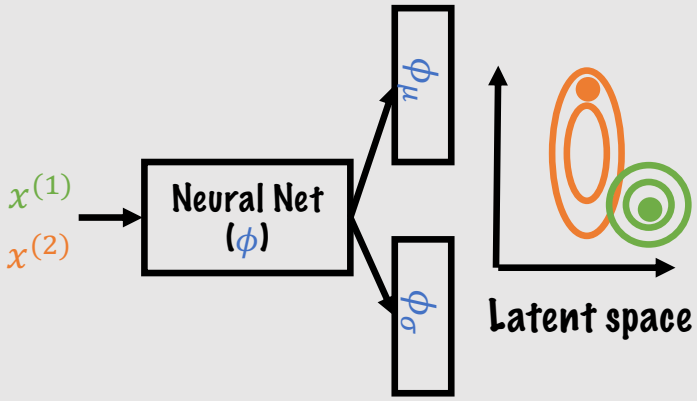
Reconstruction — Regularization





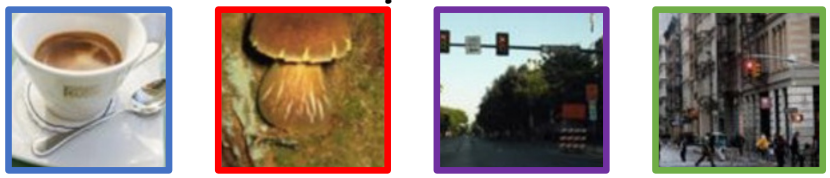
Latent variables in VAE

Inference model

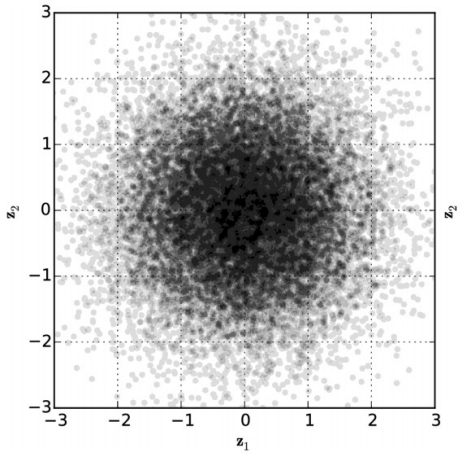


$D_{KL}\left(q(z|x^{(i)};\phi)||p(z)\right)$

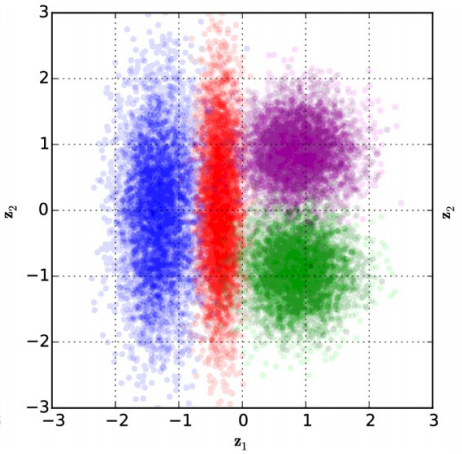
4 data points:



prior

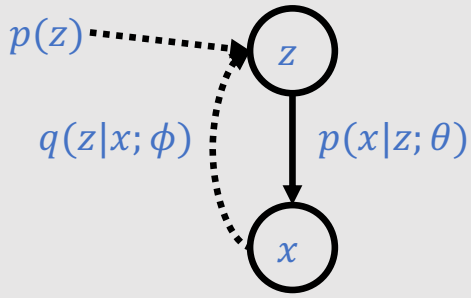


VAE posterior



Variational Autoencoders (VAE) Variants

Reminder:



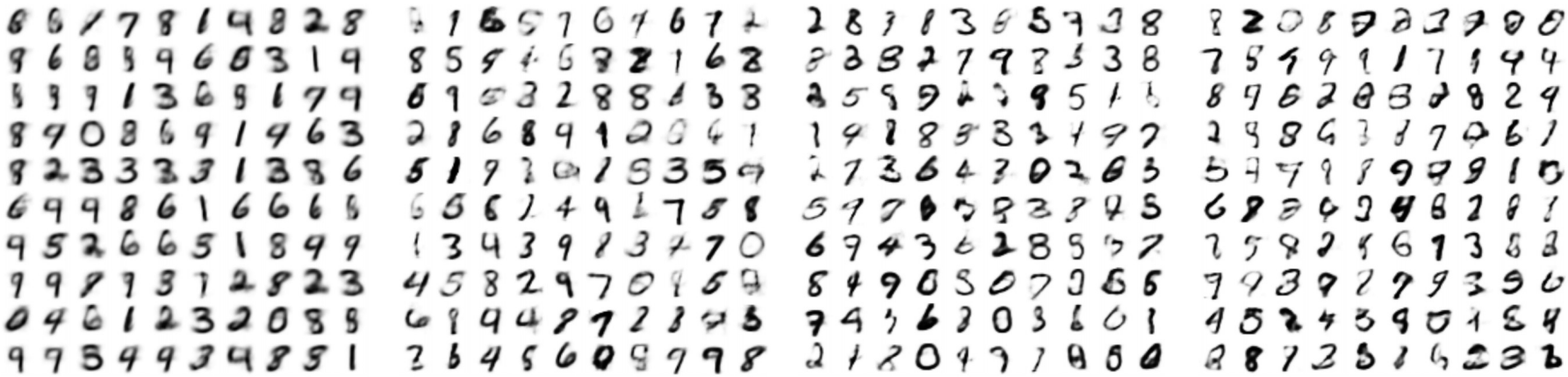
$$q(z|x^{(i)};\phi) = \mathcal{N}\left(z; \boldsymbol{f}^{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(x^{(i)}), \boldsymbol{f}^{\boldsymbol{\sigma}}_{\boldsymbol{\phi}}(x^{(i)})\right)$$

$$p(x|z^{(i)};\theta) = \mathcal{N}(x; \boldsymbol{g}^{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(z^{(i)}), \boldsymbol{g}^{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}(z^{(i)}))$$

$$\begin{cases} p(x|z^{(i)};\theta) = \mathcal{N}(x; \boldsymbol{g}^{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(z^{(i)}), g^{\sigma}_{\theta}(z^{(i)})\mathbf{1}) \\ p(x|z^{(i)};\theta) = Bernoulli\left(x; \boldsymbol{g}^{\boldsymbol{p}}_{\boldsymbol{\theta}}(z^{(i)})\right) \end{cases}$$

Results

Samples from the model

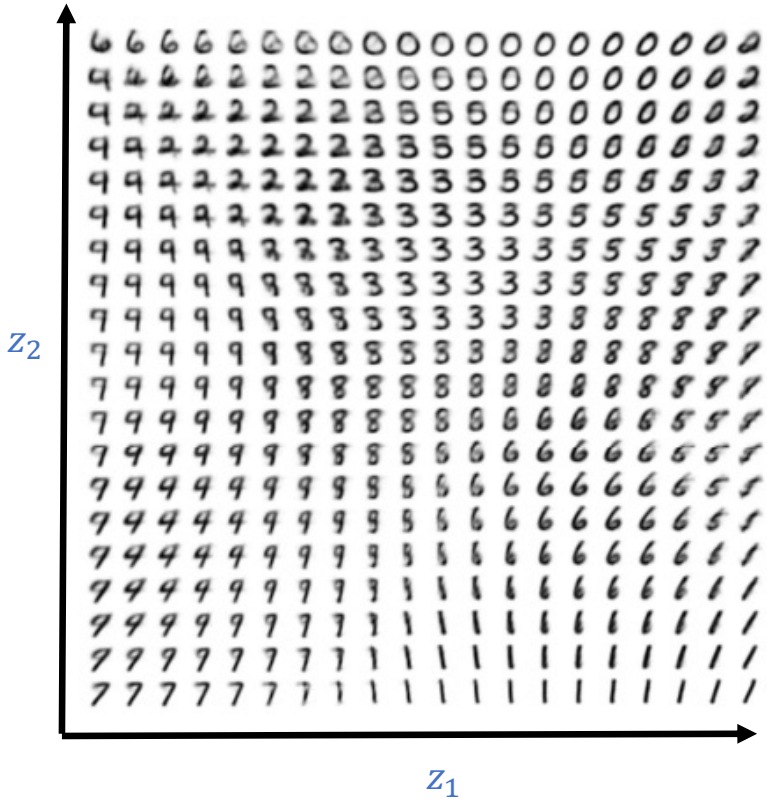


2 5 10 20

Latent space dimension

Results

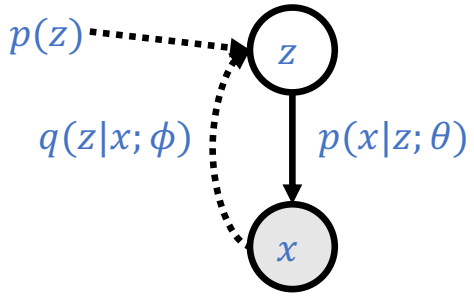
Interpolation in the latent space



Results

Current VAEs

VAE conclusions



\+

Principled approach to generative models
Allows the inference of posterior, useful in representation learning
Easy to sample

\-

Maximizes ELBO, hard to be SotA
Sample quality, still not SotA
Not easy to specify flexible posteriors

Index

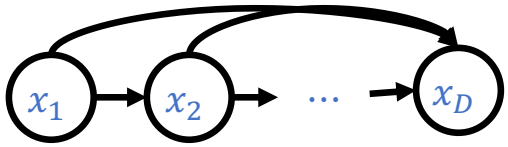
Let's play... real vs. fake

http://www.whichfaceisreal.com/index.php

Implicit model

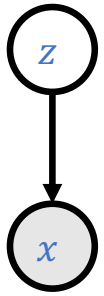
$$p_{model}(X;\theta) = \prod_{i=1}^{N} p_{model}\left(x^{(i)};\theta\right)$$

Autoregresive models



$$p_{model}\left(x_{1:D}^{(i)};\theta\right) = p_{model}\left(x_1^{(i)};\theta\right)\prod_{j=2}^{D} p_{model}\left(x_j^{(i)}\middle|x_{<j}^{(i)};\ \theta\right)$$

Latent Variable models



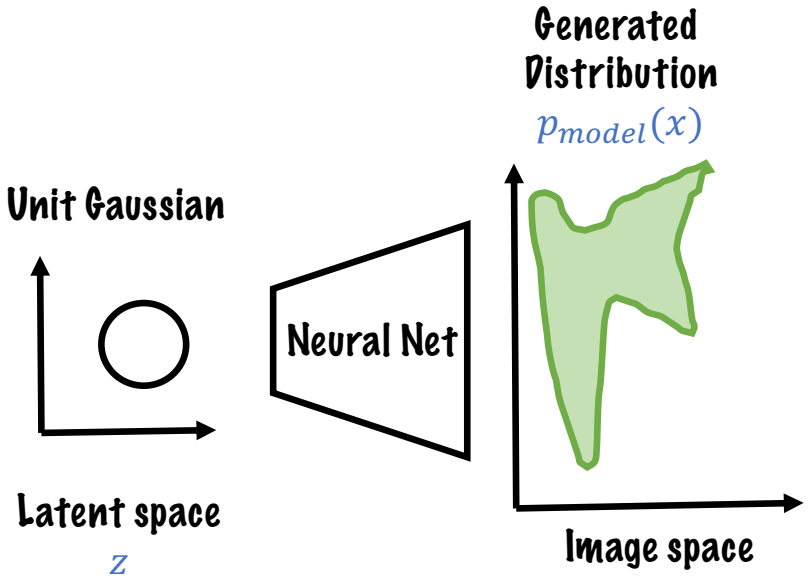
Hard to compute...

$$p_{model}\left(x^{(i)}\right) = \int p_{model}\left(x^{(i)},z\right)dz$$

Variational inference: use variational approximation

Implicit models: give up modeling the density $p_{model}\left(x^{(i)}\right)$

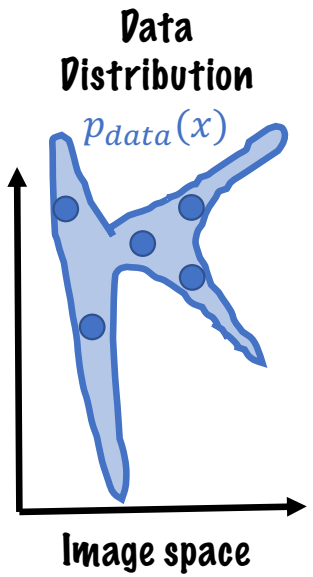
Implicit model



$$p_{model}(x) = p_{data}(x)$$

Two samples test:

$$\frac{p_{model}(x)}{p_{data}(x)} = 1$$



We do not care about the probabilities itself, we rather think how they are related.

Two sample test

Two sample test:

$$\frac{p_{model}(x)}{p_{data}(x)} = 1$$

$$p_{model}(x) = p_{data}(x)$$

We have two distributions -> 2 classes.

Let's write $p_{data}(x) = p(x|y=1)$ and $p_{model}(x) = p(x|y=-1)$.

$$\frac{p_{model}(x)}{p_{data}(x)} = \frac{p(x|y=-1)}{p(x|y=1)} = \frac{p(y=-1|x)p(x)}{p(y=-1)} \Big/ \frac{p(y=1|x)p(x)}{p(y=1)}$$

Bayes rule:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

If $p(y=-1) = p(y=1)$ e.g. we have the same # of data from $p_{model}(x)$ and $p_{data}(x)$:

$$\frac{p_{model}(x)}{p_{data}(x)} = \frac{p(y=-1|x)}{p(y=1|x)}$$ ← Classification ratio!

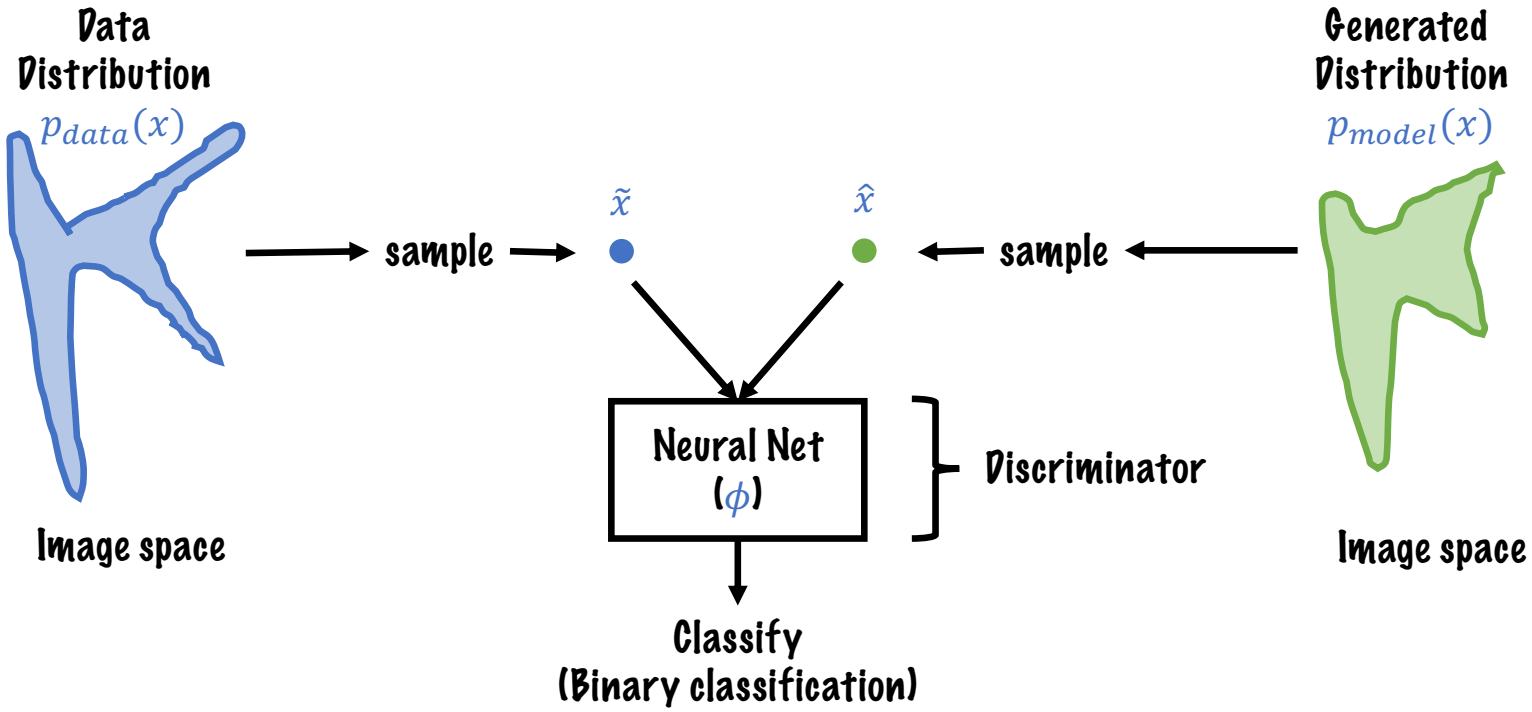
Supervised learning setup!

$$(x, y)$$

$$x \in \{\hat{x}, \tilde{x}\}, y \in \{-1, 1\}$$

Where $\hat{x}$ denotes samples form the model distribution $\hat{x} \sim p_{model}(x)$ and $\tilde{x}$ denotes samples form the data distribution $\tilde{x} \sim p_{data}(x)$.

Adversarial Loss



$$\log p(y|x;\phi) = \log p(y=1|\tilde{x};\phi) + \log p(y=-1|\hat{x};\phi) =$$
$$\log p(y=1|\tilde{x};\phi) + \log(1 - p(y=1|\hat{x};\phi))$$

Generative Adversarial Networks (GAN)

Generative Adversarial Networks

=

Adversarial Loss

+

Latent variable model

Generative Adversarial Networks (GAN) with perfect discriminator

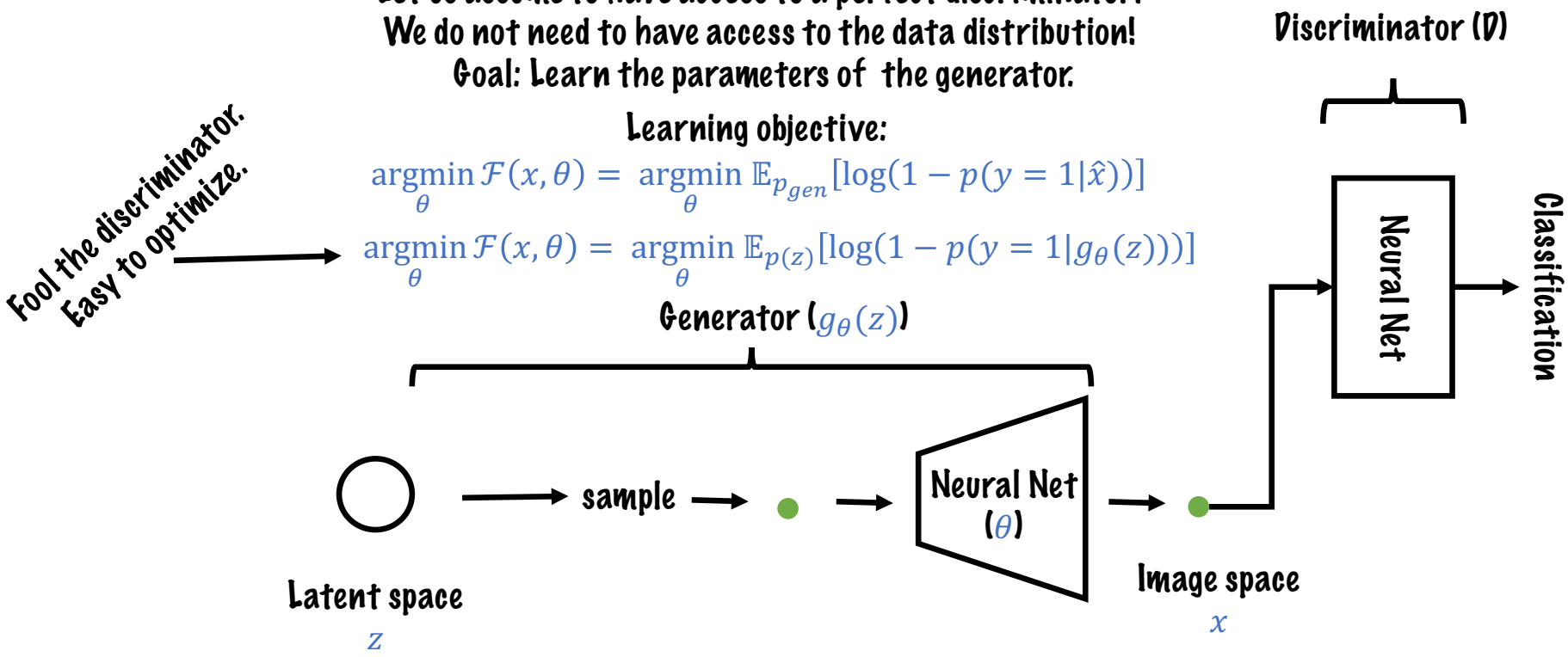
Let us assume to have access to a perfect discriminator!
We do not need to have access to the data distribution!
Goal: Learn the parameters of the generator.

Learning objective:

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{F}(x,\theta) = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{p_{gen}}[\log(1 - p(y=1|\hat{x}))]$$

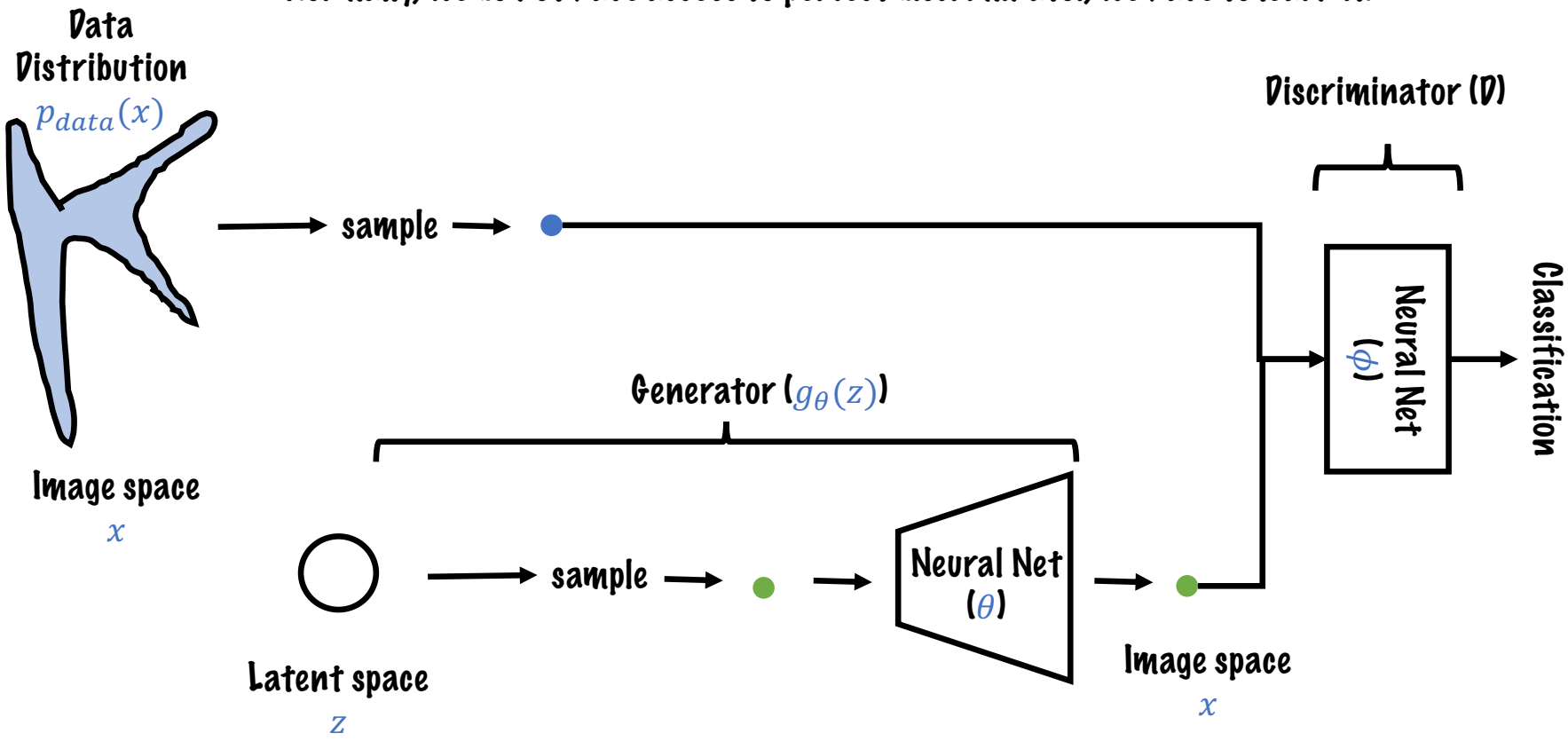
Fool the discriminator.
Easy to optimize.

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{F}(x,\theta) = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{p(z)}[\log(1 - p(y=1|g_\theta(z)))]$$



Generative Adversarial Networks (GAN)

Normally, we do not have access to perfect discriminator, we have to learn it.



Generative Adversarial Networks (GAN)

Adversarial loss have the following criterion:

$$\mathcal{F}(x, \theta, \phi) = \mathbb{E}_{p_{data}}[\log p(y = 1|\tilde{x}; \phi)] + \mathbb{E}_{p_{gen}}[\log(1 - p(y = 1|\hat{x}; \phi))]$$

We can add a generator to the criterion:

$$\mathcal{F}(x, \theta, \phi) = \mathbb{E}_{p_{data}}[\log p(y = 1|\tilde{x}; \phi)] + \mathbb{E}_{p(z)}[\log(1 - p(y = 1|g_\theta(z); \phi))]$$

The learning criteria of GAN is the following:

$$\min_{\theta} \max_{\phi} \mathcal{F}(x, \theta, \phi)$$

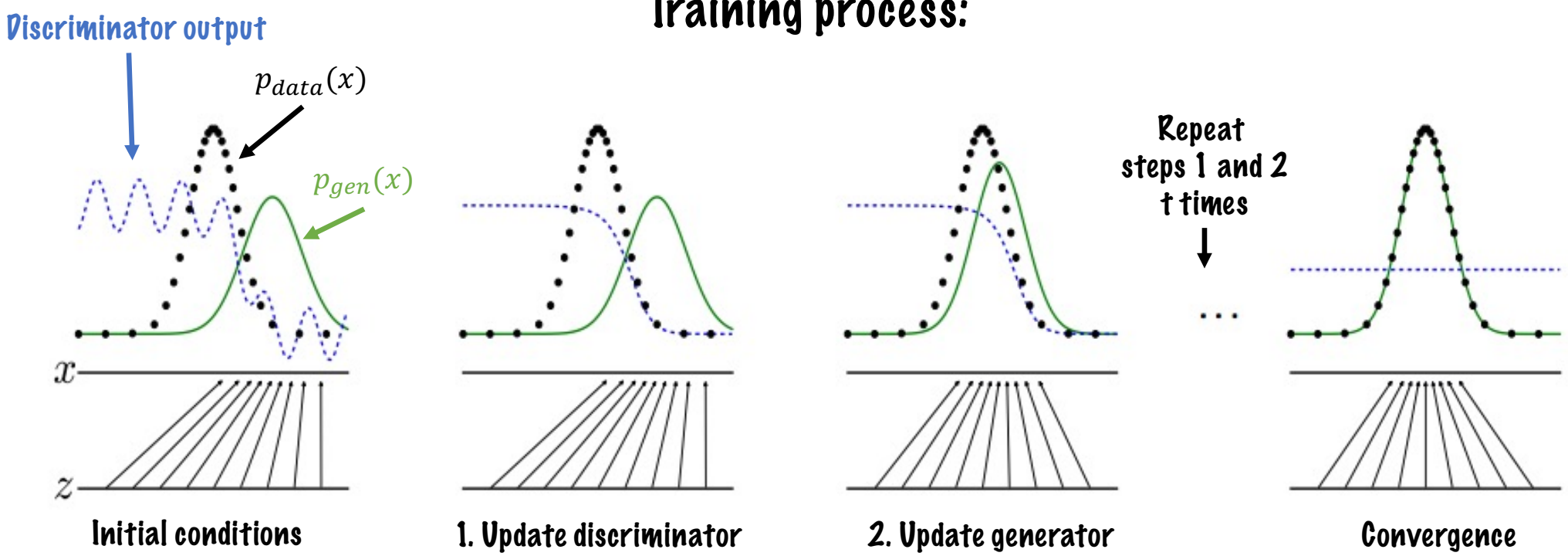
Learn a generator that fools discriminator.

Learn a good discriminator.

The learning objective of GAN is also referred to as min max game.

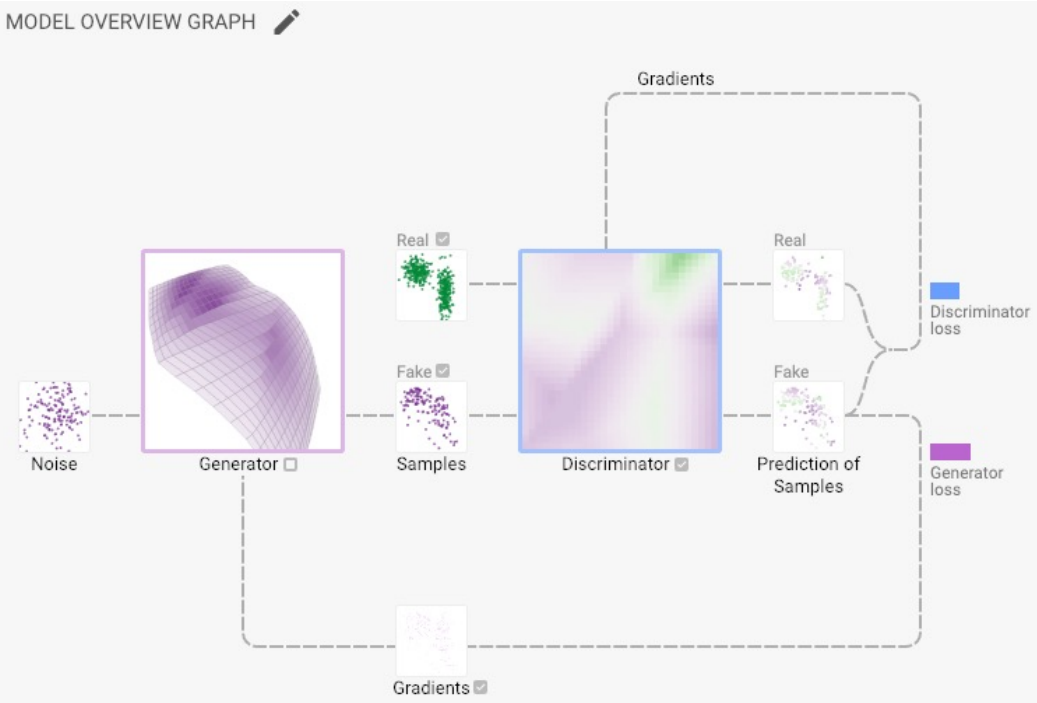
Generative Adversarial Networks (GAN)

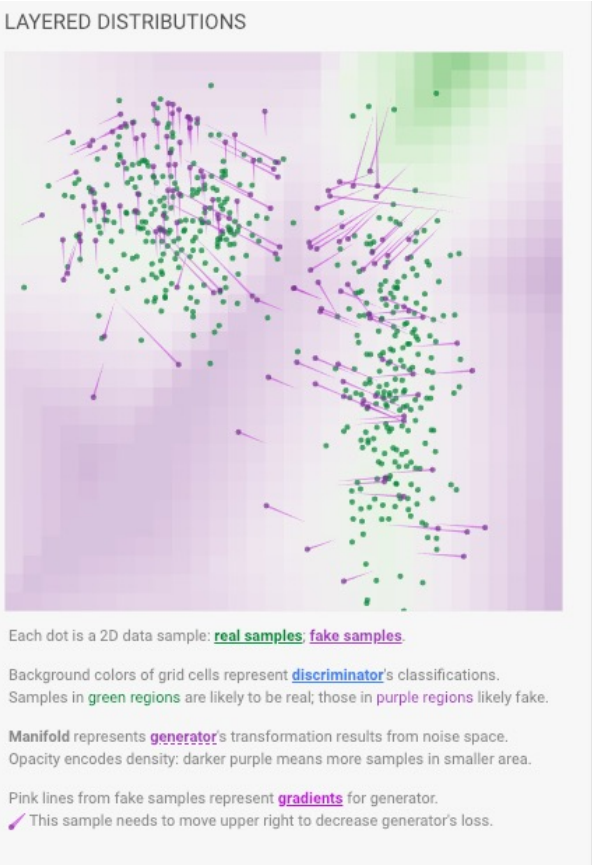
Training process:

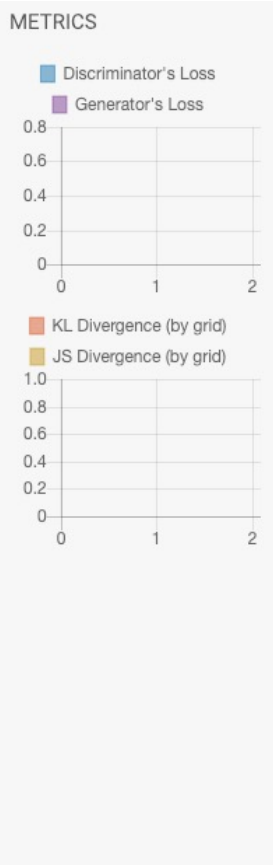


Initial conditions | 1. Update discriminator | 2. Update generator | Convergence

GAN lab







https://poloclub.github.io/ganlab/

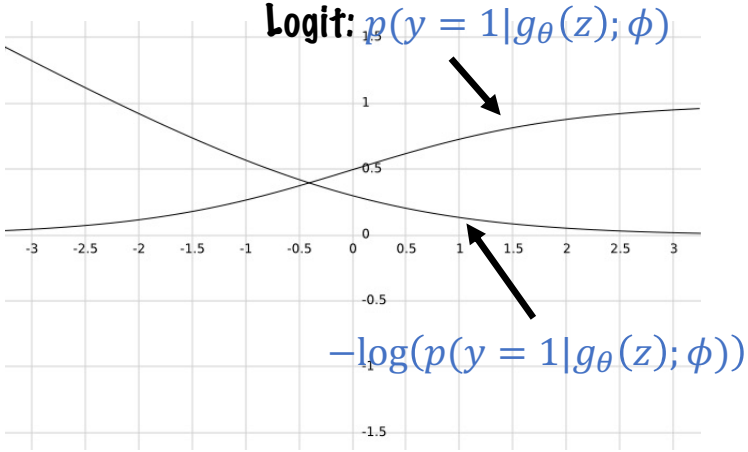
Why is it hard to train a GAN (1/2)?

Vanishing gradient in the discriminator: $\mathbb{E}_{p(z)}[\log(1 - p(y = 1|g_\theta(z); \phi))]$

Goodfellow et al suggest to use: $-\mathbb{E}_{p(z)}[\log(p(y = 1|g_\theta(z); \phi))]$

Log fixes the exponential plateau in the discriminator

Logit: $p(y = 1|g_\theta(z); \phi)$



$-\log(p(y = 1|g_\theta(z); \phi))$

The image comes from generator $g_\theta(z)$.

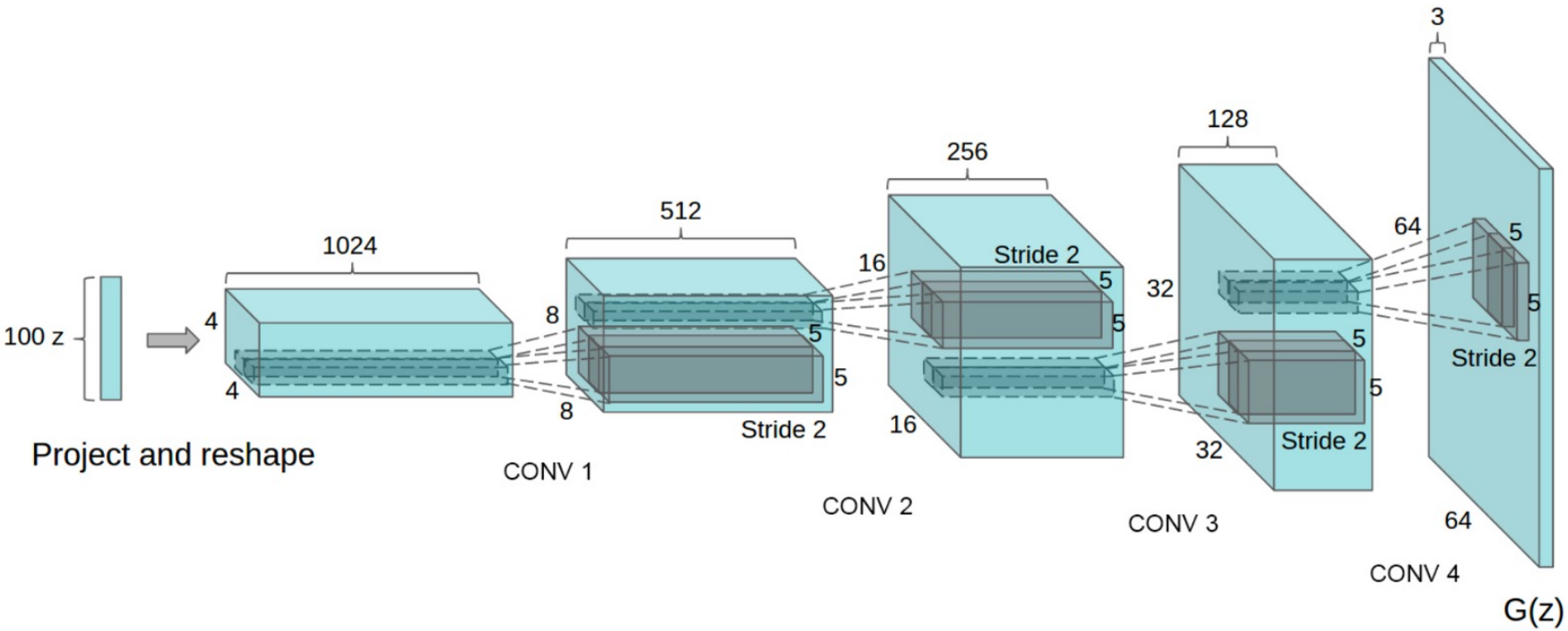
Strong discriminator

Weak discriminator

Why is it hard to train a GAN (2/2)?

- Mode collapse: generator models very well a small sub-space, concerning on a few modes.
- No convergence: there is no guarantee that min max game will actually converge.

Deep Convolutional GAN (DCGAN) architecture

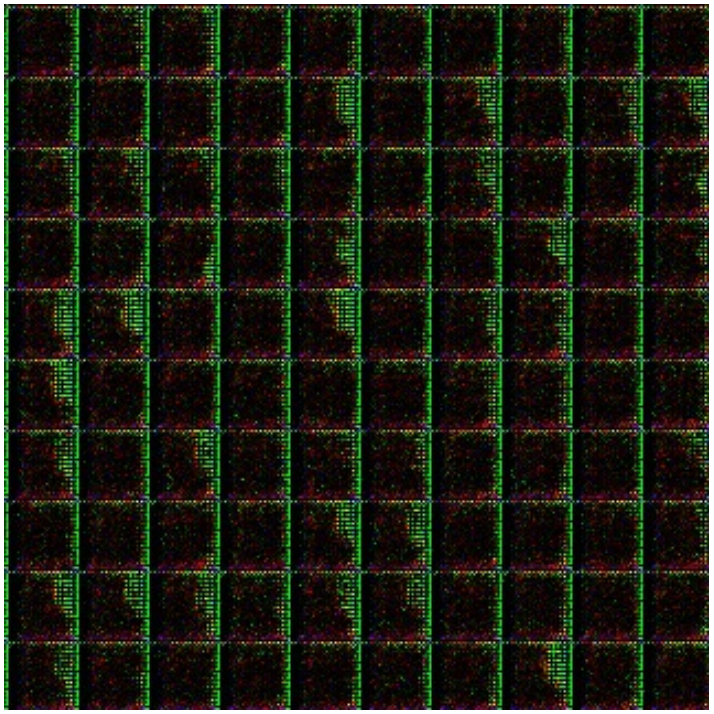


DCGAN architecture

- Use strided convolution in discriminator
- Use strided transposed convolutions in generator
- Use batchnorm in discriminator and generator
- Do not use fully connected hidden layers
- ReLu in generator (last layer tanh)
- LeakyReLu in discriminator

VAE vs GAN training

VAE



GAN

DCGAN Results

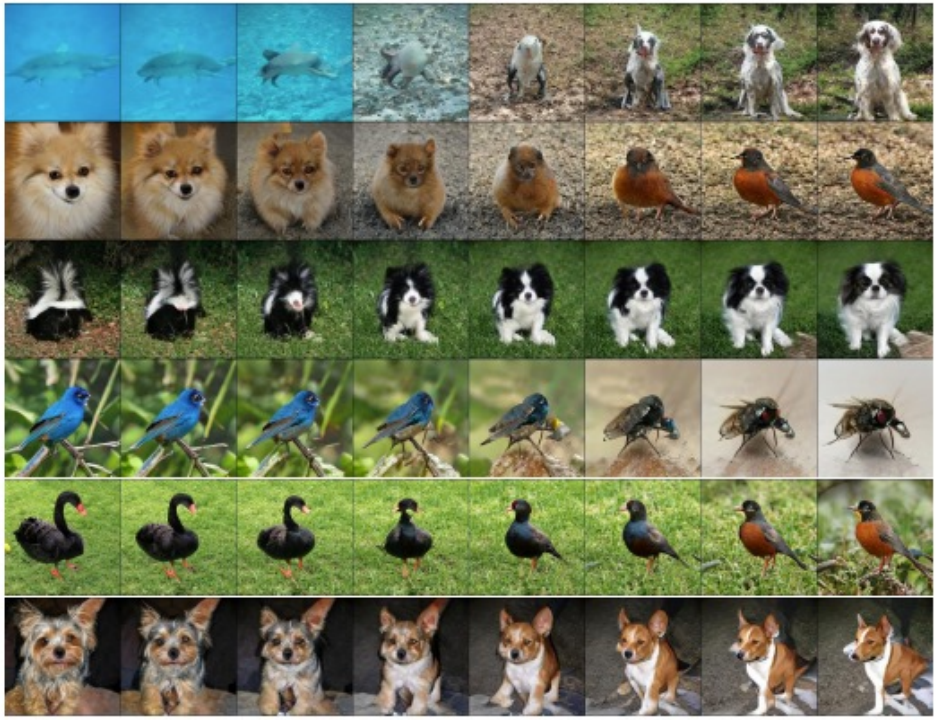
Samples

Interpolations

GANs: progress over the years

BIGGAN
Samples

BIGGAN Interpolations



To compare with likelihood based models

Interesting GAN papers not covered in the class

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

Adversarial Feature Learning

Jeff Donahue
jdonahue@cs.berkeley.edu
Computer Science Division
University of California, Berkeley

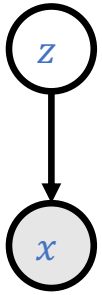
Philipp Krähenbühl
philkr@utexas.edu
Department of Computer Science
University of Texas, Austin

Trevor Darrell
trevor@eecs.berkeley.edu
Computer Science Division
University of California, Berkeley

Adversarially Learned Inference

Vincent Dumoulin[1], Ishmael Belghazi[1], Ben Poole[2]
Olivier Mastropietro[1], Alex Lamb[1], Martin Arjovsky[3]
Aaron Courville[1†]
[1] MILA, Université de Montréal, firstname.lastname@umontreal.ca.
[2] Neural Dynamics and Computation Lab, Stanford, poole@cs.stanford.edu.
[3] New York University, martinarjovsky@gmail.com.
[†]CIFAR Fellow.

GAN conclusions



\+

Easy to sample
Nice image generations

\-

Hard to train (unstable)
Don't have inference model $p(z|x^{(i)})$
We cannot compute likelihood $p_{model}(X;\theta)$

Index

Evaluation

Evaluation depends on the exact task we solve with our generative model.

Evaluation of image generation is tricky.

- Image generation metrics:
  - We can evaluate data points (e. g. test set)
    - How likely is it to observe an image under the model? (Likelihood)
  - We can evaluate samples from the model
    - Are the samples plausible? (Visual evaluation or Inception score)

Test set likelihood

We want to compute the log likelihood over test set:

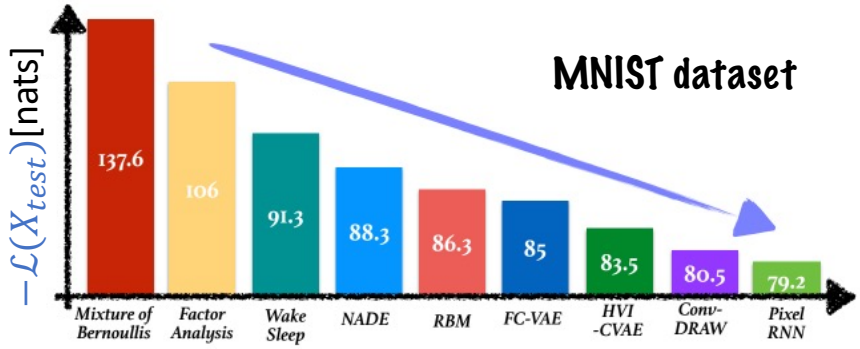
$$\mathcal{L}(X_{test}) = \log p_{model}(X_{test}\,; \theta)$$

That is expectation (mean log likelihood) over test set:

$$\mathcal{L}(X_{test}) = \mathbb{E}_{X_{test}} \log p_{model}(x^{(i)}_{test}\,; \theta)$$

In some papers people will report negative log likelihood: $-\mathcal{L}(X_{test})$

The base for the $\log$ determines units, can be either 2 (bits) or e (nats).



We cannot compute this metric for implicit models.

Figure from: https://drive.google.com/file/d/0B_wzP_JlVFcKMnFMNnAtYTVkV28/view

Visual quality of samples

Subjective metric!

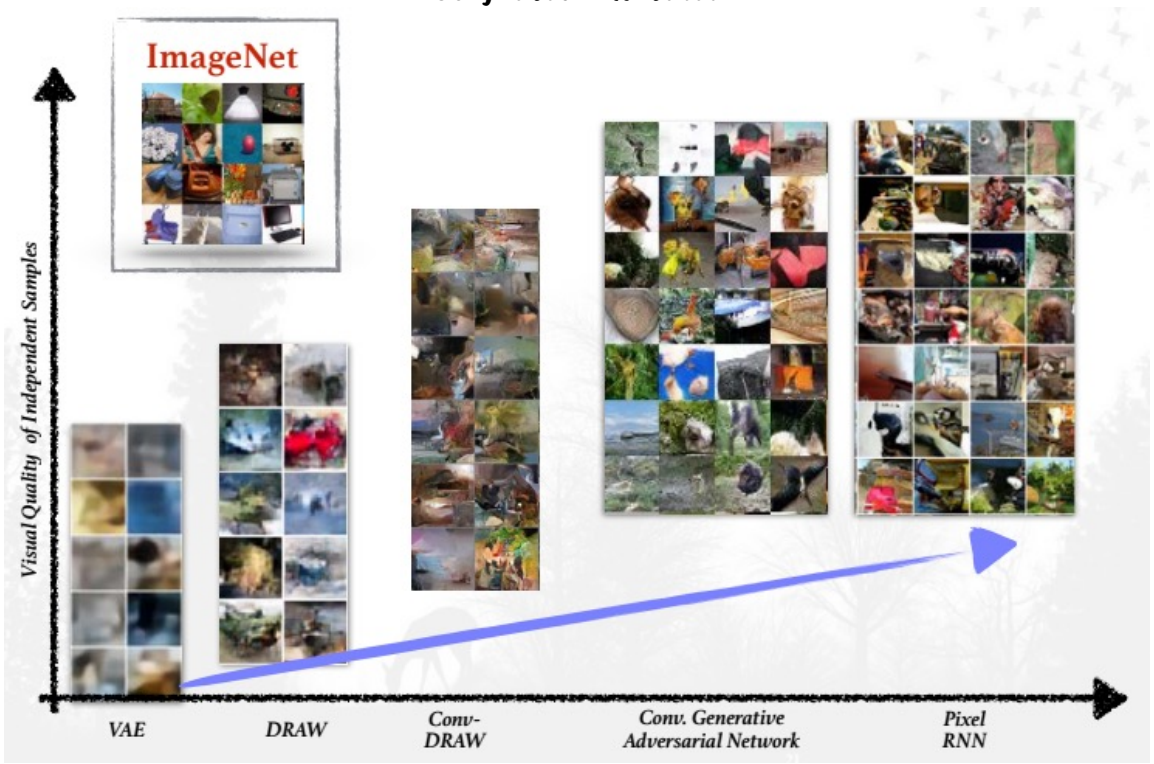
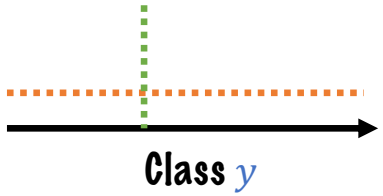


Figure from: https://drive.google.com/file/d/0B_wzP_JlVFcKMnFMNnAtYTVkV28/view

Inception score (IS)

Use inception v3 model trained on ImageNet to evaluate the samples.

The generative model should create samples that are:
Variable (all classes of imagenet are probable)
For a given data point, the inception model should be highly confident



$$IS = \exp(\mathbb{E}_{x \sim p_{model}} D_{KL}(p_{inception}(y|x)||p(y))$$

IS is not perfect.

Samples with high IS.

Fréchet Inception Distance (FID)

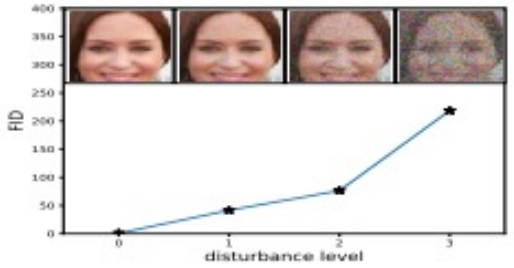
Use inception model trained on ImageNet to evaluate the samples.

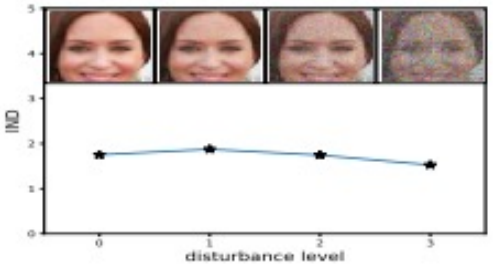
Compare the statistics of generated samples.

-> Fit a gaussian to an inception model activations at some level using real test set and generated images.

-> Compute the FID distance between the gaussains.

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}\right).$$





Index

To be added

Wrap up

Intro

Latent space

Maximum likelihood definition

Maximum likelihood models

Autoregressive models (Pixel CNN)

Compute likelihood directly

Variational approximation for likelihood computation

Variational inference (VAE)

Implicit models (GAN)

Give up computation of likelihood and use two sample test

How to compare models?

Summary

Wrap up (applications)

Generative models are important for many applications:

We have only scratched the surface...

Figure from: https://drive.google.com/file/d/0B_wzP_JlVFcKMnFMNnAtYTVkV28/view

References

Autoregressive models (Pixel CNN)

Pixel Recurrent Neural Networks; Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu

Conditional Image Generation with PixelCNN Decoders; Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu

Parallel Multiscale Autoregressive Density Estimation; Scott Reed, Aaron van den Oord, Nal Kalchbrenner, Sergio Gomez Colmenarejo, Ziyu Wang, Dan Belov, Nando de Freitas

Variational inference (VAE)

Auto-Encoding Variational Bayes; Diederik P. Kingma, Max Welling

Variational Inference with Normalizing Flows; Danilo Jimenez Rezende, Shakir Mohamed

Improved Variational Inference with Inverse Autoregressive Flow; Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, Max Welling

Implicit models (GAN)

Generative Adversarial Nets; Ian J. Goodfellow , Jean Pouget-Abadie , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS; Alec Radford, Luke Metz, Soumith Chintala

Summary

A NOTE ON THE EVALUATION OF GENERATIVE MODELS; Lucas Theis, Aaron van den Oord, Matthias Bethg

Generative Models

Michal Drozdzal
mdrozdzal@fb.com