Lecture 14: Deep Generative Models III: VAE & GAN

Lan Xu

SIST, ShanghaiTech

Fall, 2023

Outline

- VAEs
  - Inverse graphics network
  - Attribute2Image
- Generative Adversarial Networks
  - Implicit generative models
  - Adversarial learning

Acknowledgement: Feifei Li et al's cs231n notes

The Variational Autencoder: overview

- Learning:
  - Given a large dataset of observations $\mathbf{X} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\}$
  - Estimating the parameters in Deep LVM

$$p(x) = \int p(x, z)\, dz \quad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = \boxed{p(\boldsymbol{x} \mid \boldsymbol{z})} p(\boldsymbol{z})$$

  - Based on Maximum Likelihood

$$\max \sum_{i=1}^{N} \log p(x^{(i)})$$

  - Direct optimization is challenging: use EM learning strategy
  - Jointly learning inference model with the deep latent variable model

VAE objective

- Recall lower bound of the data log likelihood

$$
\begin{aligned}
\log p_\theta(x) &= \log \int_z p_\theta(x,z)dz \\
&= \log \int_z q_\phi(z|x)\frac{p_\theta(x,z)}{q_\phi(z|x)}dz \\
&\geq \int_z q_\phi(z|x)\log \frac{p_\theta(x,z)}{q_\phi(z|x)}dz \quad \text{(Jensen's Inequality)} \\
&= \mathbf{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] = \mathcal{L}(x;\theta,\phi)
\end{aligned}
$$

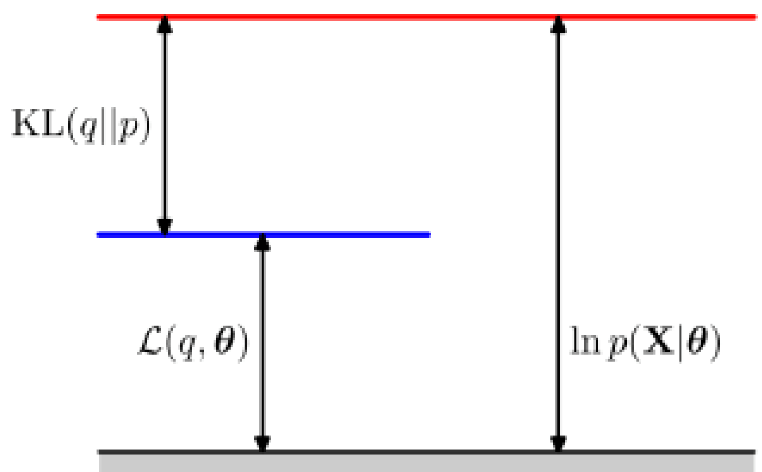
$$
\log p_\theta(x) = \boxed{\mathcal{L}(x;\theta,\phi)} + D_{KL}(q_\phi(z|x)||p_\theta(z|x))
$$

- Learning: maximize the lower bound of data likelihood
- The evidence lower bound (ELBO)

Review: VAEs

- Main ideas:
  - Introduce a parametric model $q_\phi(z \mid x)$ to approximate the true posterior $p_\theta(z \mid x)$
  - Jointly learn approximate posterior with the deep latent variable model
  - Variational EM: lower-bound of the Maximum Likelihood

$$\log p_\theta(x) = \boxed{\mathcal{L}(x;\theta,\phi)} + D_{KL}(q_\phi(z|x)||p_\theta(z|x))$$



Review: VAEs

- Main ideas:
  - Introduce a parametric model $q_\phi(z \mid x)$ to approximate the true posterior $p_\theta(z \mid x)$
  - Jointly learn approximate posterior with the deep latent variable model
  - Variational EM: lower-bound of the Maximum Likelihood

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\ &= -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]\end{aligned}$$

***regularization* term** ***reconstruction* term**

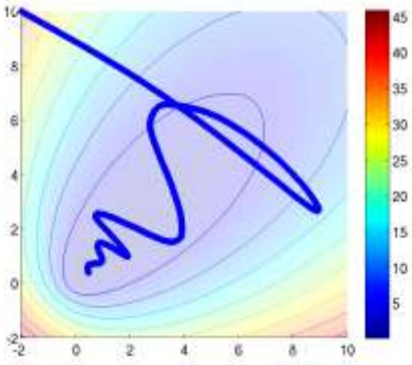
VAE learning

- EM perspective
  - Expectation Maximization alternately optimizes the ELBO, $\mathcal{L}(q, \theta)$, with respect to q (the E step) and θ (the M step)
  - Initialize $\theta^{(0)}$
  - At each iteration $t = 1, \ldots$
    - **E step:** Hold $\theta^{(t-1)}$ fixed, find $q^{(t)}$ which maximizes $\mathcal{L}\left(q, \theta^{(t-1)}\right)$
    - **M step:** Hold $q^{(t)}$ fixed, find $\theta^{(t)}$ which maximizes $\mathcal{L}\left(q^{(t)}, \theta\right)$

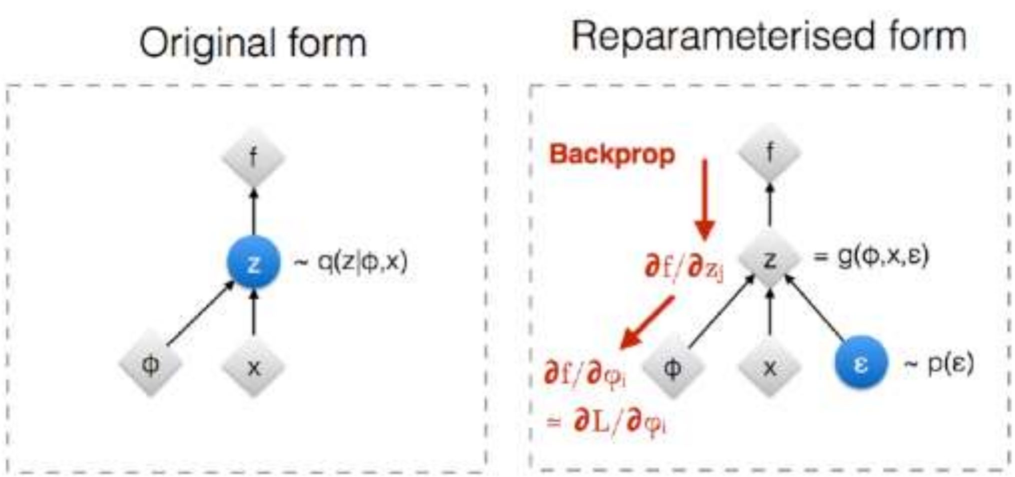
$$\mathcal{L}(x, \phi, \theta) = E_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)]$$

Two views of Learning VAE

- Optimization interpretation
  - Stochastic gradient-based



- Network interpretation
  - Backpropagation



Optimization interpretation

- Recall VAE objective

$$\mathcal{L}(x,\phi,\theta) = E_{q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)]$$

  - Or rewrite as $\mathcal{L}(x,\phi,\theta) = E_{q_\phi(z|x)}[f_{\phi,\theta}(x,z)]$

- Often no analytic solution to exact gradient

$$\nabla_{\phi,\theta}\mathcal{L}(x,\phi,\theta)$$

  - Solution: stochastic gradient ascent
  - Requires unbiased estimates of gradient
  - Can use small minibatches or single point of data

$$\nabla_\phi \mathcal{L}(x,\phi,\theta) \approx \nabla_\phi f_{\phi,\theta}(x, z^{(i)}), \quad \boxed{z^{(i)} \sim q_\phi(z|x)}$$

High variance for gradient estimation

Reparameterization trick

- Reparameterize $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x})$ using a differentiable transformation of an auxiliary noise variable ϵ

$$\mathbf{z} = g_\phi(\epsilon, \mathbf{x}) \quad \text{with} \quad \epsilon \sim q(\epsilon)$$

  - Then we can write the ELBO as

$$\mathcal{L}(x, \phi, \theta) = E_{q_\phi(z|x)}[f_{\phi,\theta}(x, z)] = E_{q(\epsilon)}[f_{\phi,\theta}(x, g_\phi(\epsilon, \mathbf{x}))]$$

  - And its gradient estimation with L samples

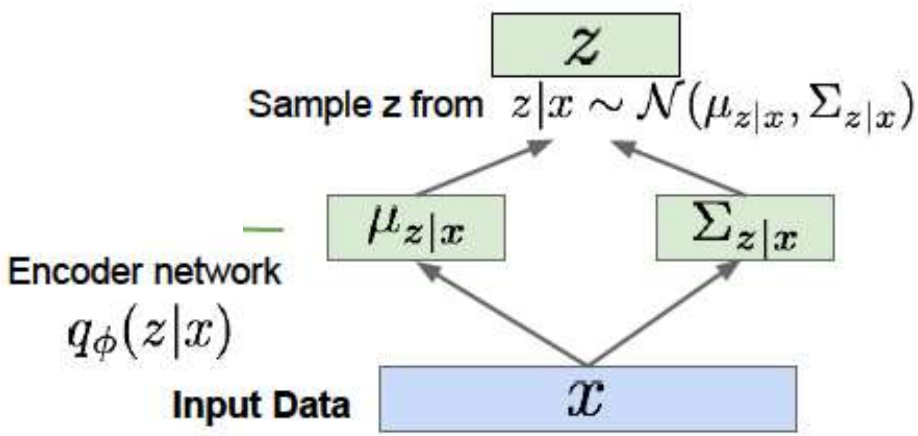
$$\nabla_\phi \mathcal{L}(x, \phi, \theta) = E_{q(\epsilon)}[\nabla_\phi f_{\phi,\theta}(x, z)] \approx \frac{1}{L} \sum_{i=1}^{L} \nabla_\phi f_{\phi,\theta}(x, g_\phi(\epsilon^{(i)}, x)), \quad \boxed{\epsilon^{(i)} \sim q(\epsilon)}$$

VAE Example

- Univariate Gaussian $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$

$$z = \mu + \sigma\epsilon \qquad \epsilon \sim \mathcal{N}(0,1)$$

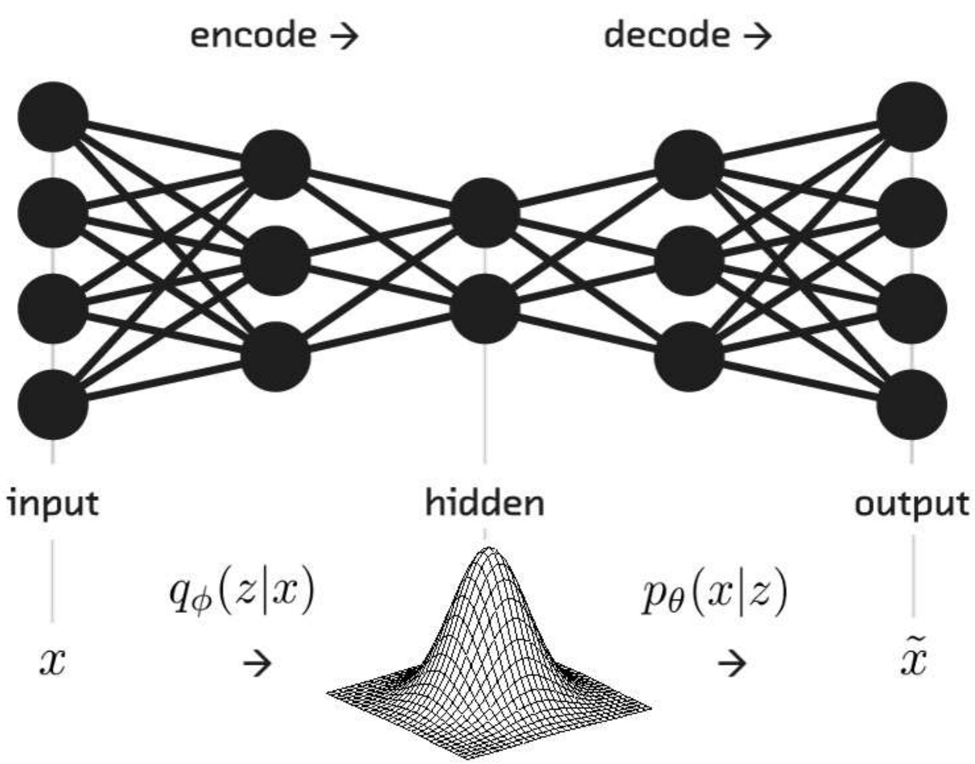
$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}\left[f(z)\right] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}\left[f(\mu+\sigma\epsilon)\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mu+\sigma\epsilon^{(l)})$$



Autoencoder Interpretation

- Objective $\mathcal{L}(x,\phi,\theta) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$

Regularization term Reconstruction term



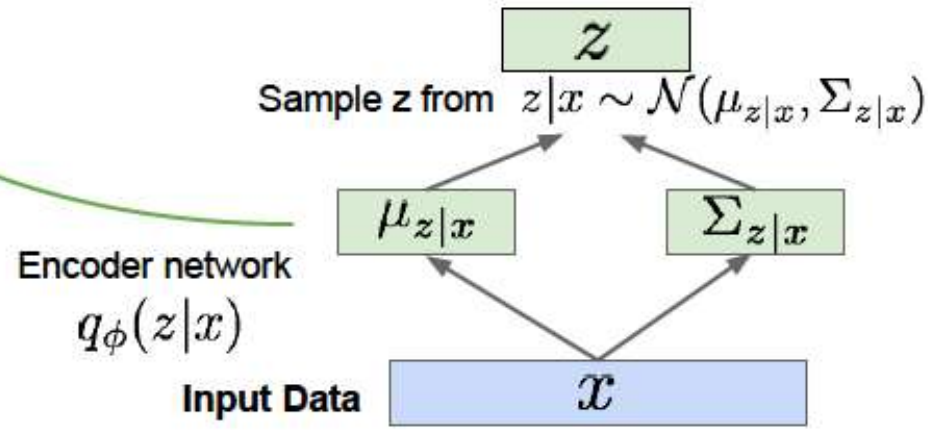
VAE Example

- Learning objective

Putting it all together: maximizing the likelihood lower bound

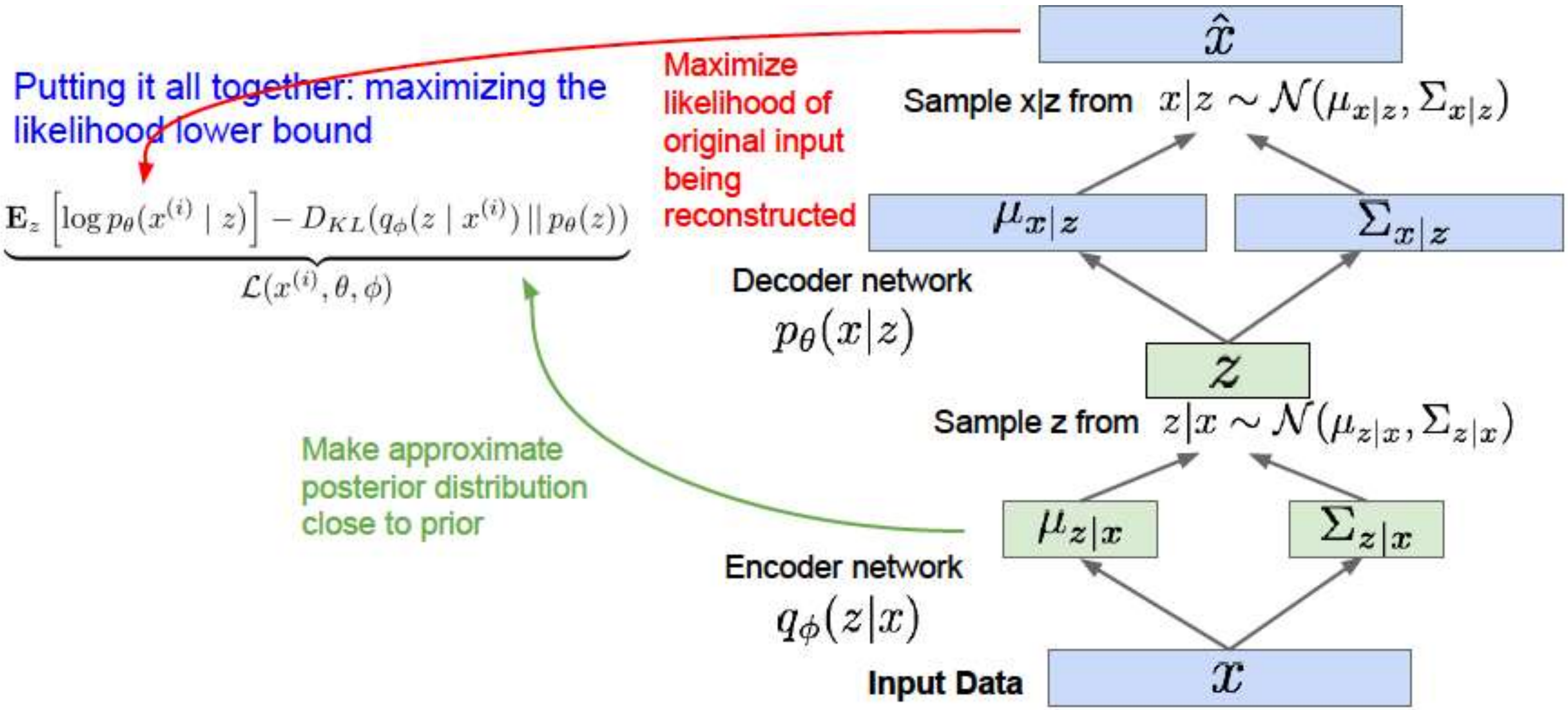
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Make approximate posterior distribution close to prior



VAE Example

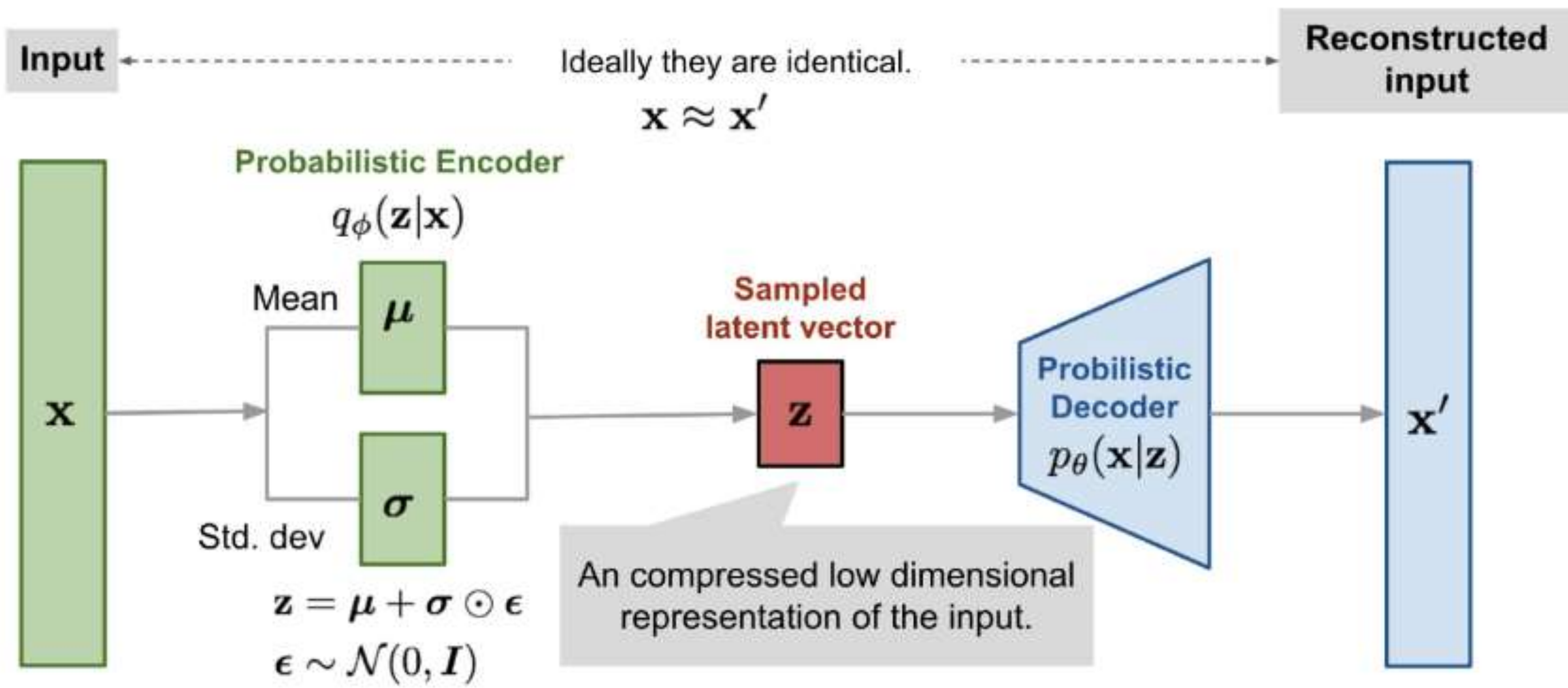
- Learning objective



Autoencoder Interpretation

- Objective $\mathcal{L}(x,\phi,\theta) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$

  Regularization term   Reconstruction term

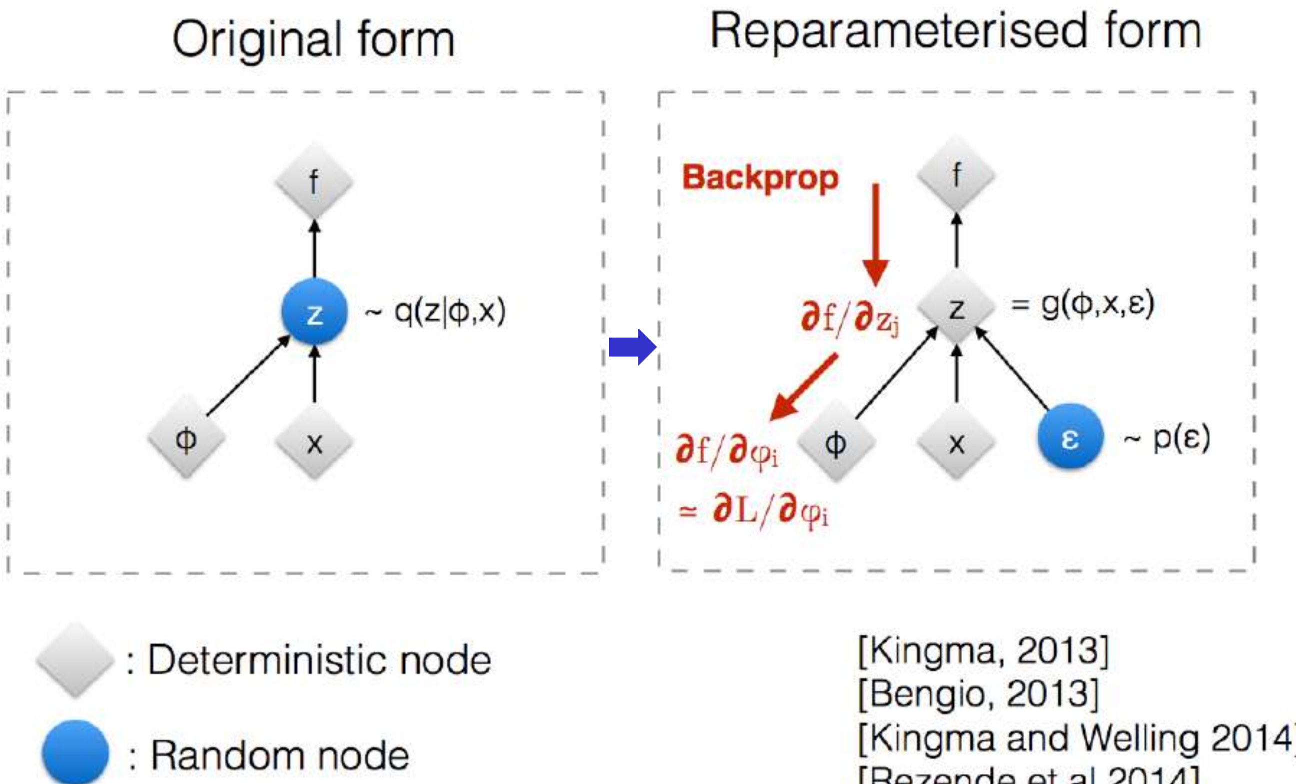


  - The objective function can be represented as an Autoencoder-like computation graph.

Network interpretation

$$\mathcal{L}(x,\phi,\theta) = E_{q_\phi(z|x)}[f_{\phi,\theta}(x,z)]$$

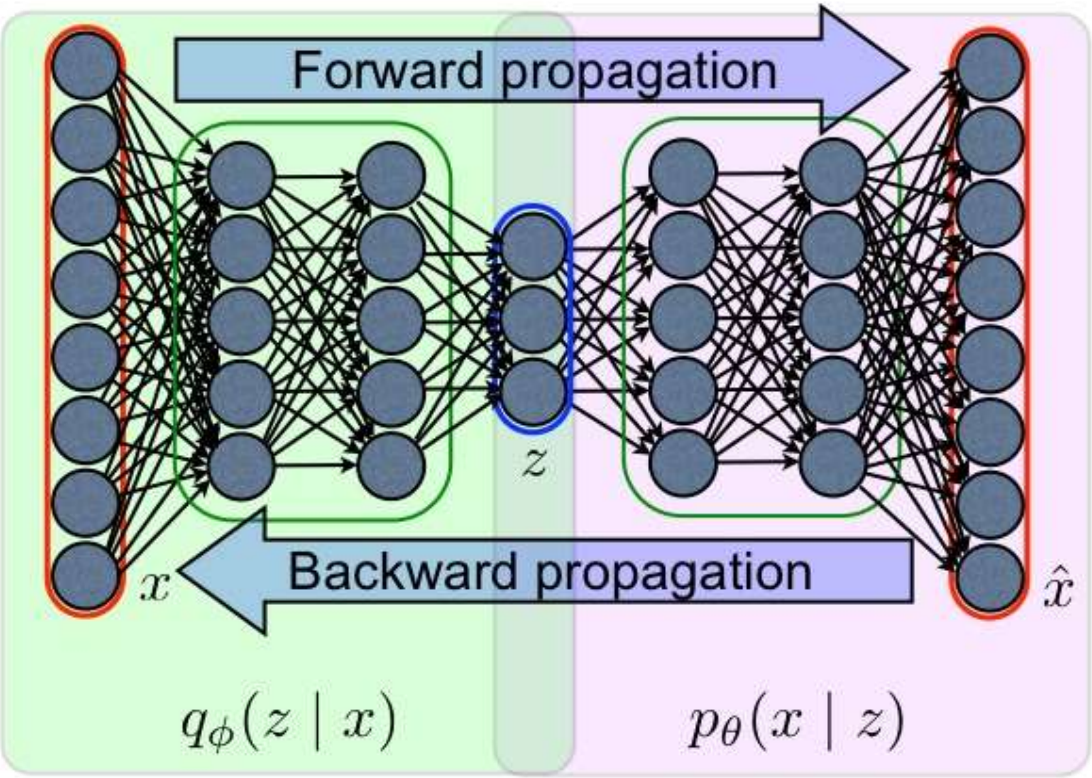
$$\mathcal{L}(x,\phi,\theta) = E_{q(\epsilon)}[f_{\phi,\theta}(x,z)] \approx \frac{1}{L}\sum_{i=1}^{L} f_{\phi,\theta}(x, g_\phi(\epsilon^{(i)}, x)), \quad \epsilon^{(i)} \sim q(\epsilon)$$



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Training with Backpropagation

- Due to reparametrization trick, we can simultaneously train both the generative model and the inference model by optimizing the variational bound using the gradient backpropagation.



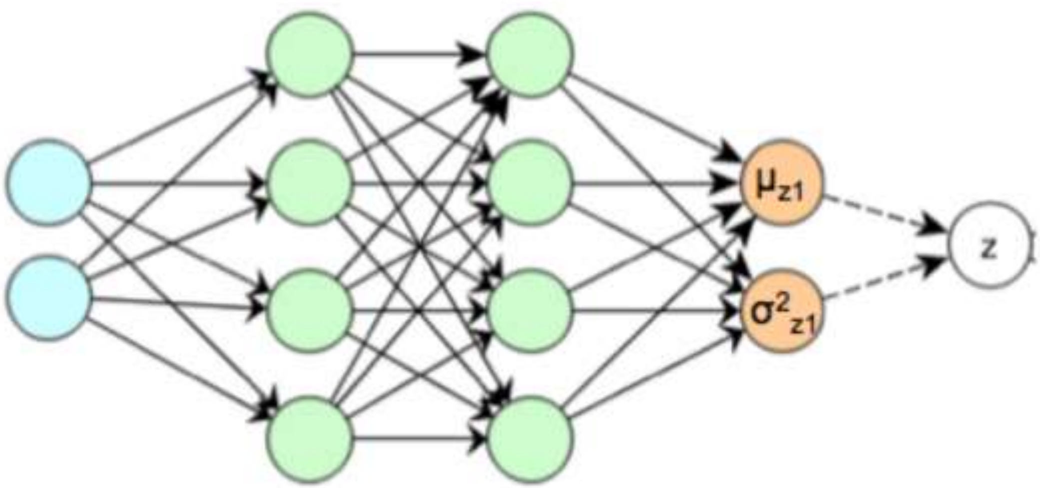
1D Gaussian Case

- We can compute the KL regularization in close form

Use N(0,1) as prior for p(z)

$q(z|x^{(i)})$ is Gaussian with parameters $(\mu^{(i)},\sigma^{(i)})$ determined by NN

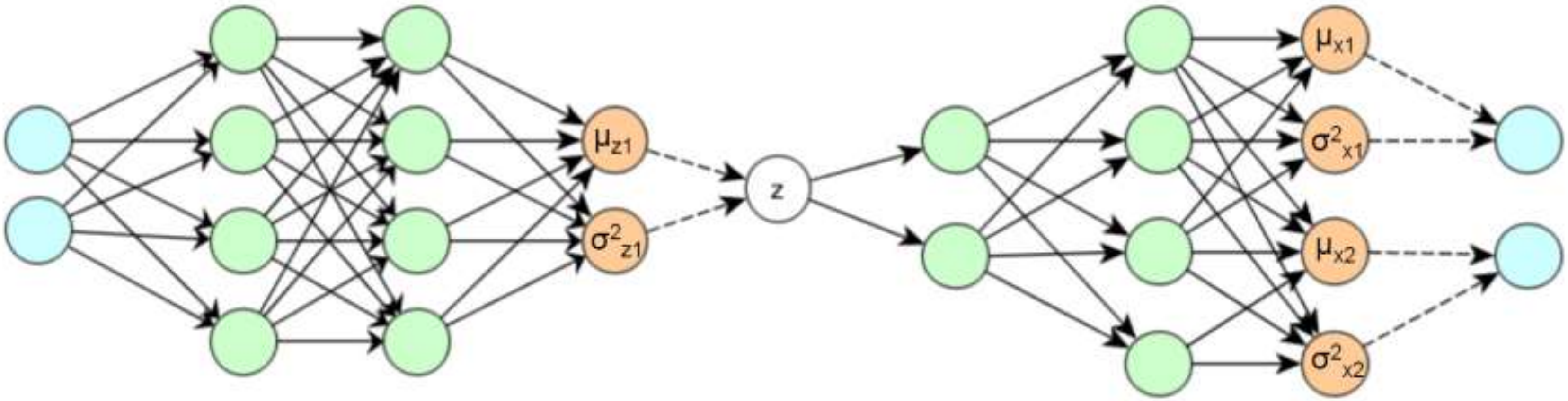
$$-D_{KL}\left(q(z|x^{(i)})\|p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$



1D Gaussian Case

- Overall loss function for BP

Prior p(z) ~ N(0,1) and p, q Gaussian, extension to dim(z) > 1 trivial



Cost: Regularisation

$$-D_{KL}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$

We use mini batch gradient decent to optimize the cost function over all $x^{(i)}$ in the mini batch

Cost: Reproduction

$$-\log\left(p(x^{(i)}|z^{(i)})\right) = \sum_{j=1}^{D}\frac{1}{2}\log(\sigma_{x_j}^2) + \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$

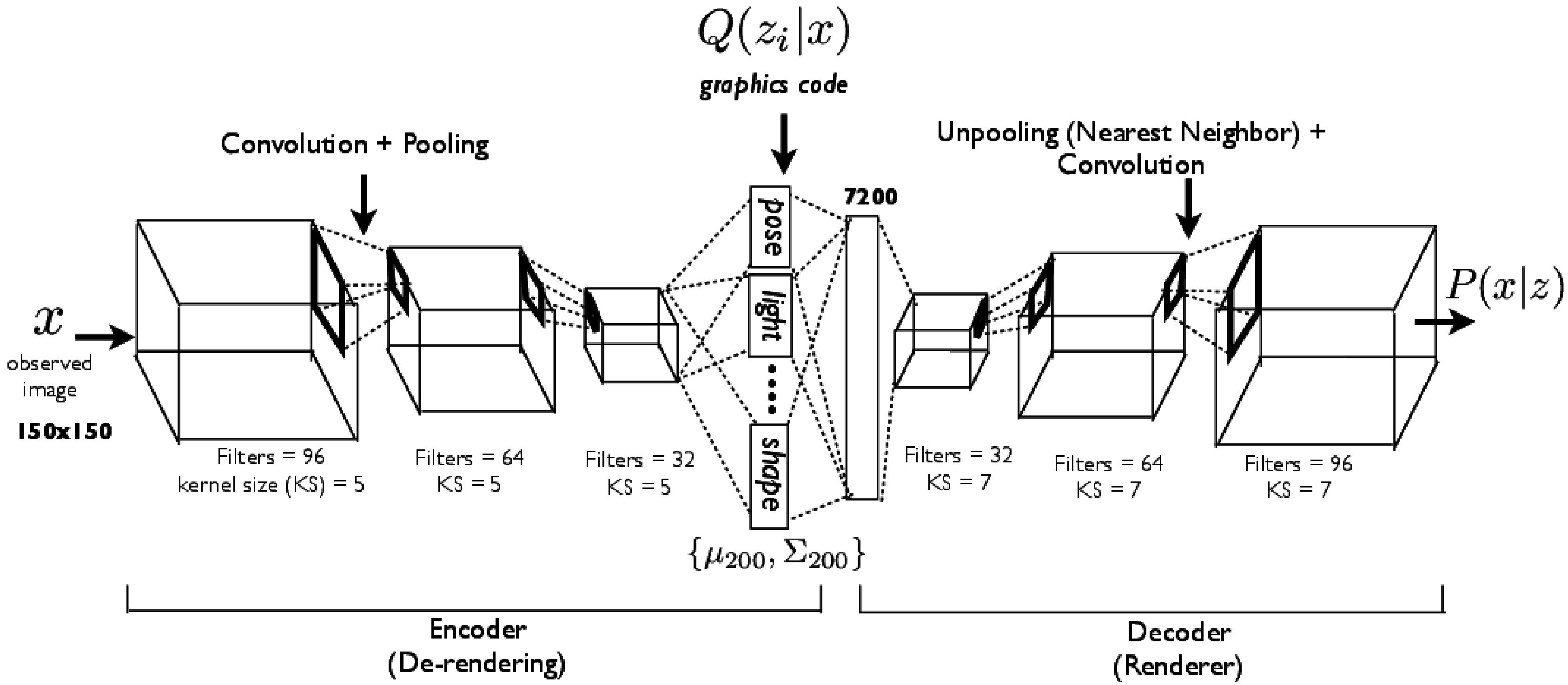
Least Square for constant variance

Interpreting the latent space

https://arxiv.org/pdf/1610.00291.pdf

Vision task I – Inverse graphics network

- Main ideas



	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_{[4,n]}$
$\mathrm{Z} =$	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_{[4,n]}$
corresponds to	ϕ	α	ϕ_L	intrinsic properties (shape, texture, etc)

Vision task I – Inverse graphics network

- Mini-batch training

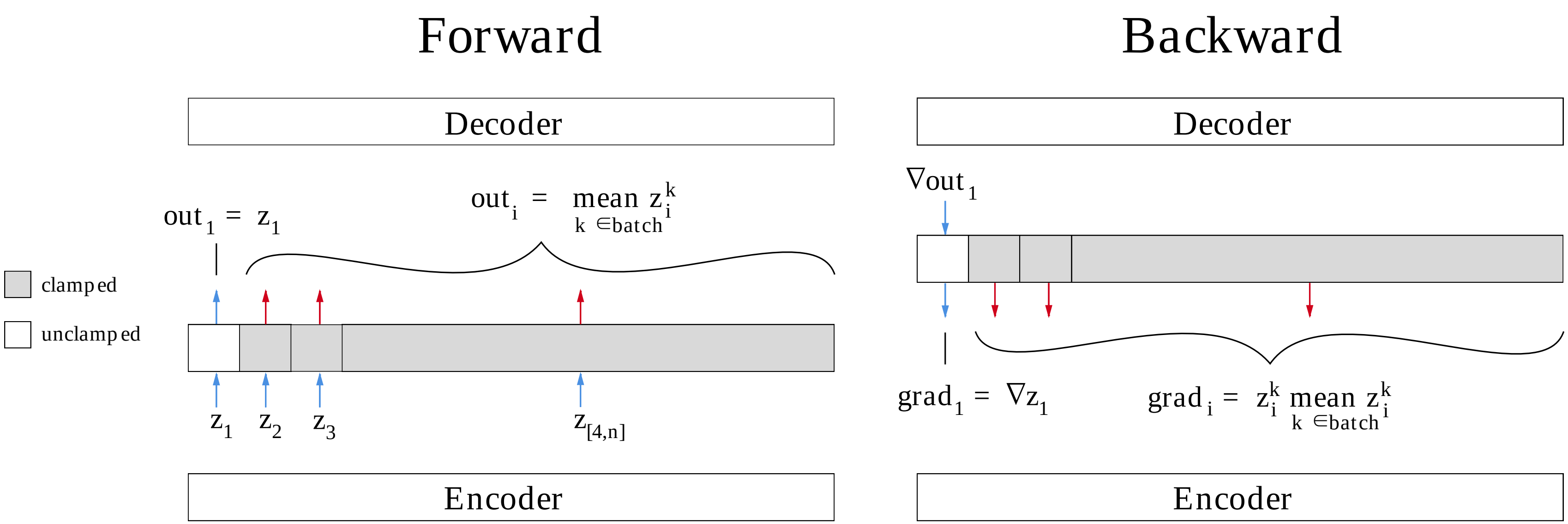
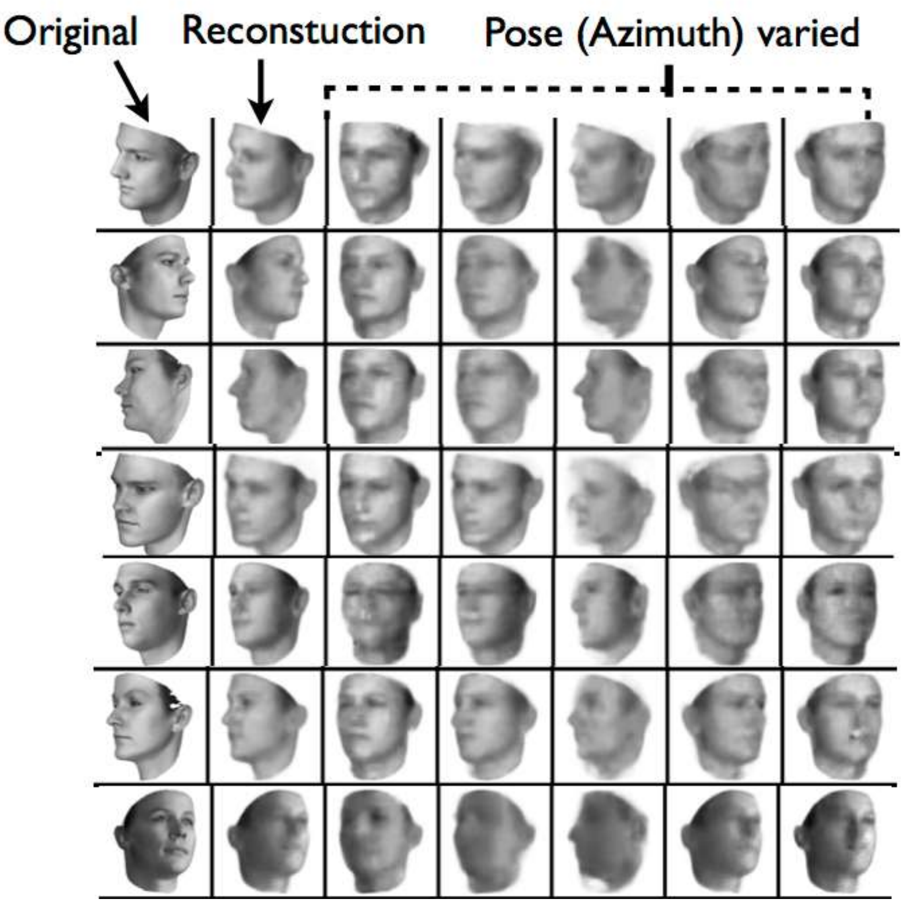
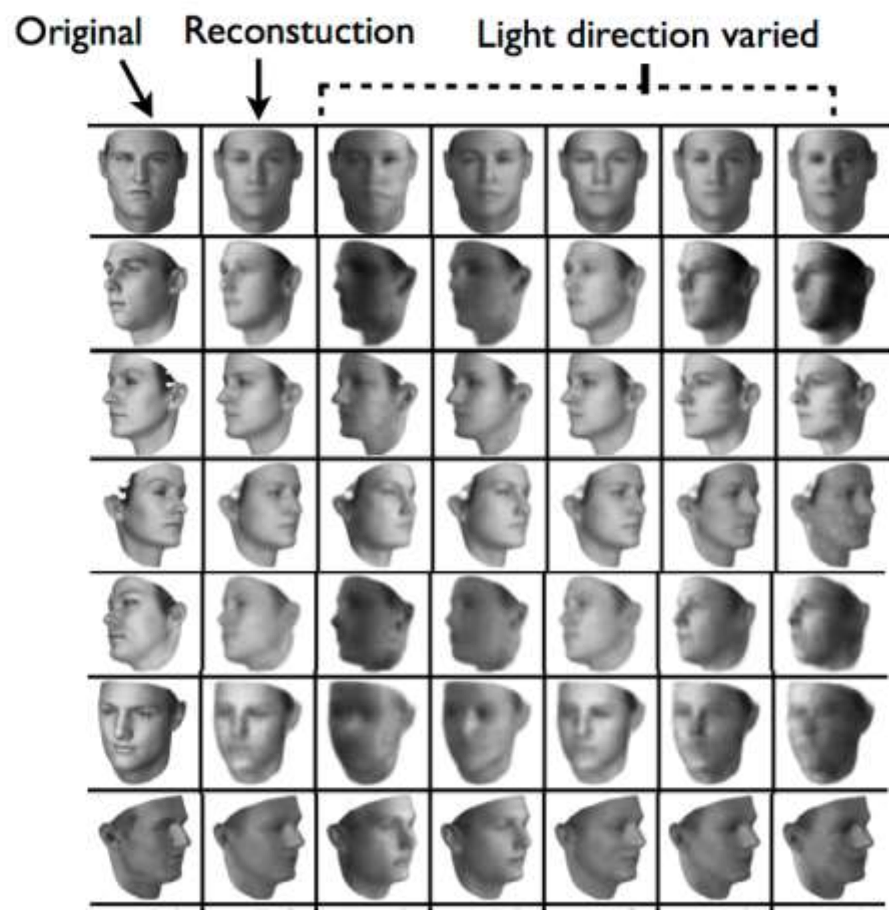


Figure 3: **Training on a minibatch in which only ϕ, the azimuth angle of the face, changes.** During the forward step, the output from each component $z_i \neq z_1$ of the encoder is altered to be the same for each sample in the batch. This reflects the fact that the generating variables of the image (e.g. the identity of the face) which correspond to the desired values of these latents are unchanged throughout the batch. By holding these outputs constant throughout the batch, the single neuron z_1 is forced to explain all the variance within the batch, i.e. the full range of changes to the image caused by changing ϕ. During the backward step z_1 is the only neuron which receives a gradient signal from the attempted reconstruction, and all $z_i \neq z_1$ receive a signal which nudges them to be closer to their respective averages over the batch. During the complete training process, after this batch, another batch is selected at random; it likewise contains variations of only one of $\phi, \alpha, \phi_L, intrinsic$; all neurons which do not correspond to the selected latent are clamped; and the training proceeds.

Vision task I – Inverse graphics network

- Results





Vision task I – Inverse graphics network

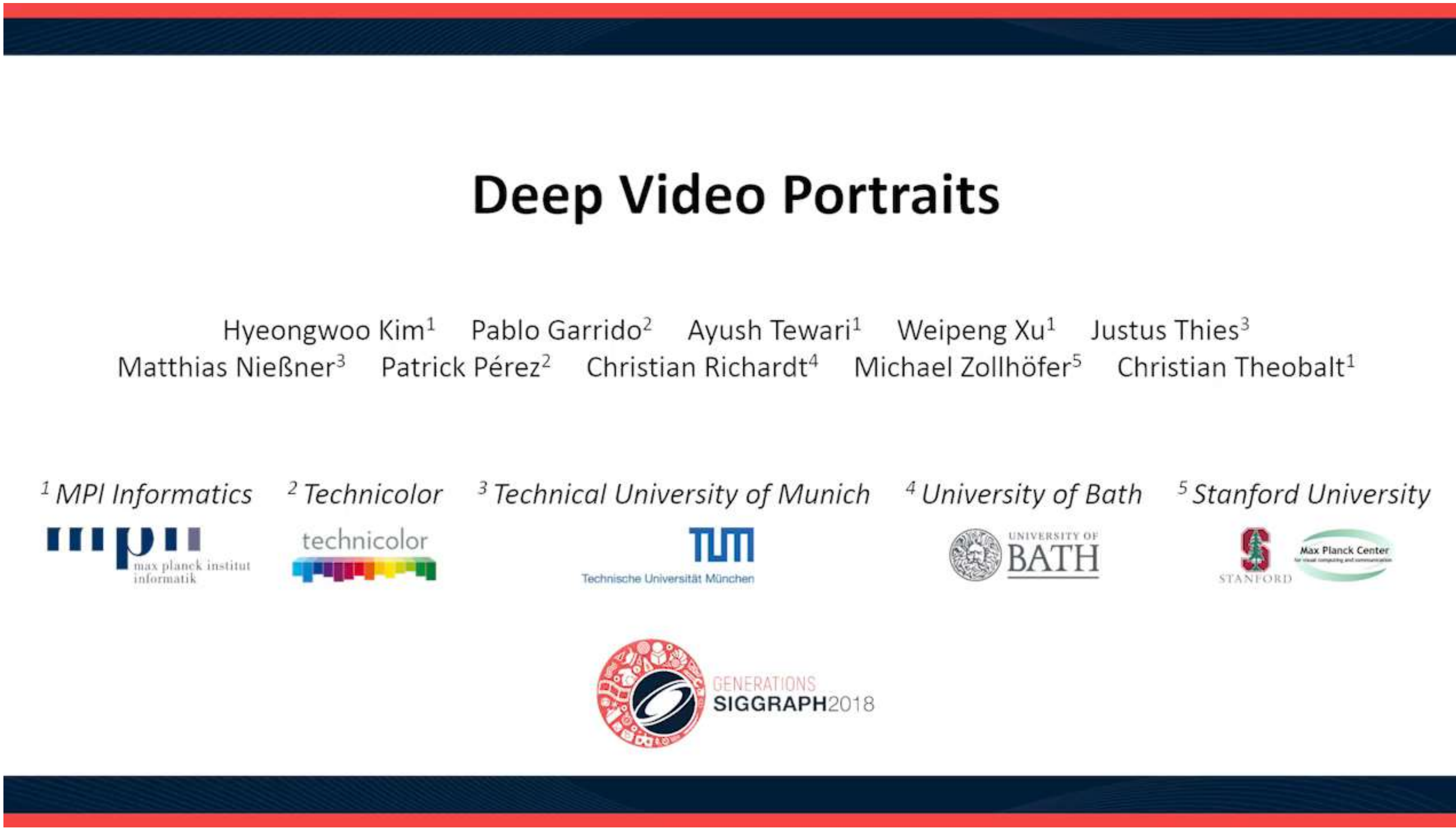
- MoFA: Model-based Face Autoencoder (ICCV2017)

Vision task I – Inverse graphics network

- FML: Face model learning from videos (CVPR2019)

Vision task I – Inverse graphics network

- DVP: Deep Video Portraits (SIGGRAPH 2018)



Vision task I – Inverse graphics network

- Live Speech Portraits (SIGGRAPHAISA 2021)

SIGGRAPH ASIA 2021 TOKYO

LIVE

Live Speech Portraits: Real-Time Photorealistic Talking-Head Animation

Yuanxun Lu[1] Jinxiang Chai[2] Xun Cao[1]

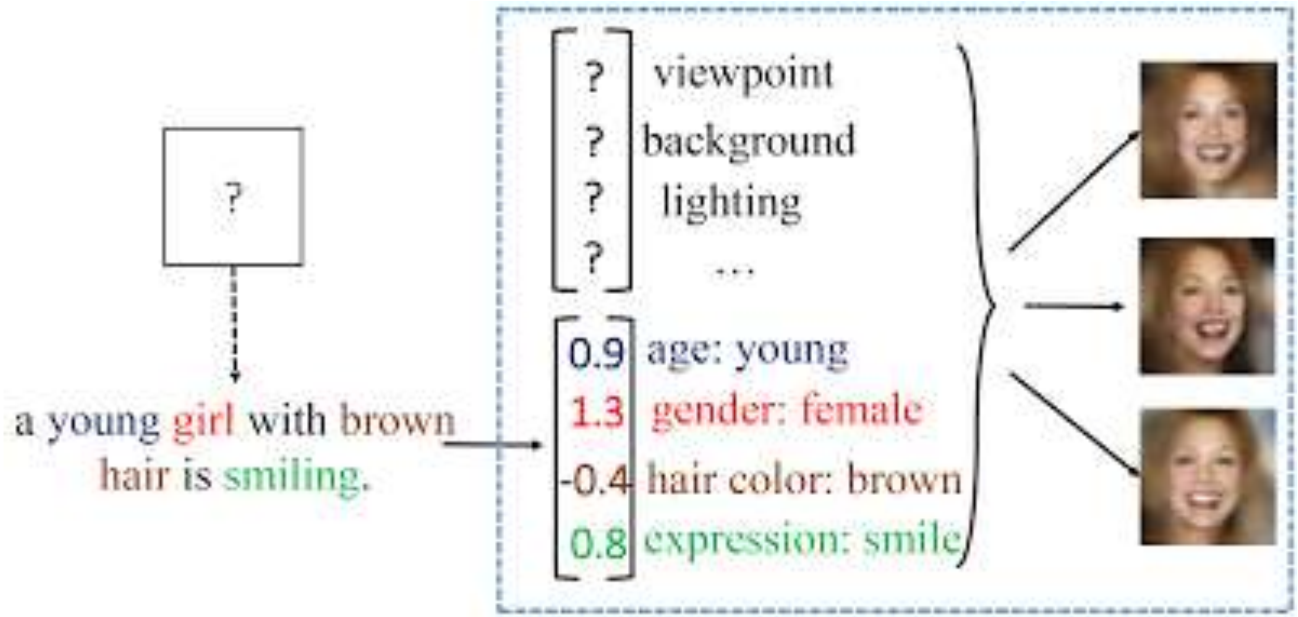
[1] Nanjing University [2] Xmov

南京大學 NANJING UNIVERSITY XMOV

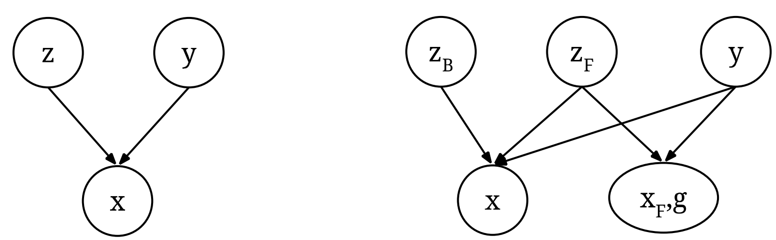
(with Audio)

Vision task II – Attribute2Image

- Main ideas



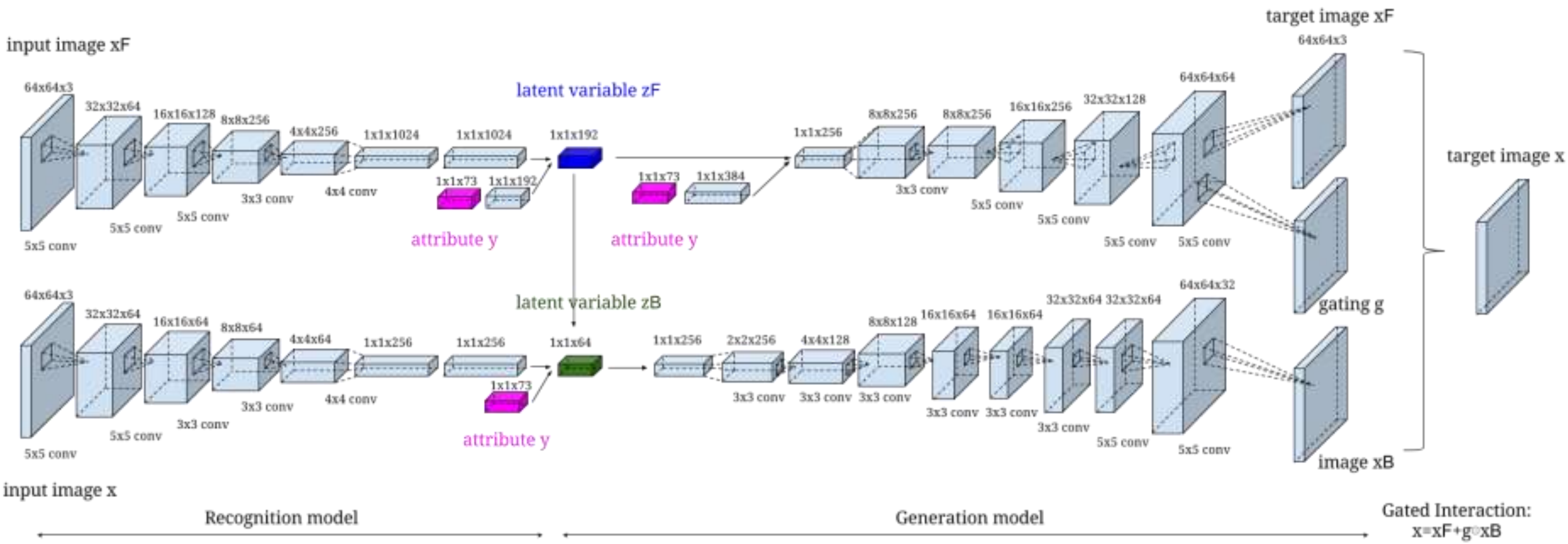
Attribute-conditioned Image Generation



(a) CVAE: $p_\theta(x|y, z)$ (b) disCVAE: $p_\theta(x, x_F, g|y, z_F, z_B)$

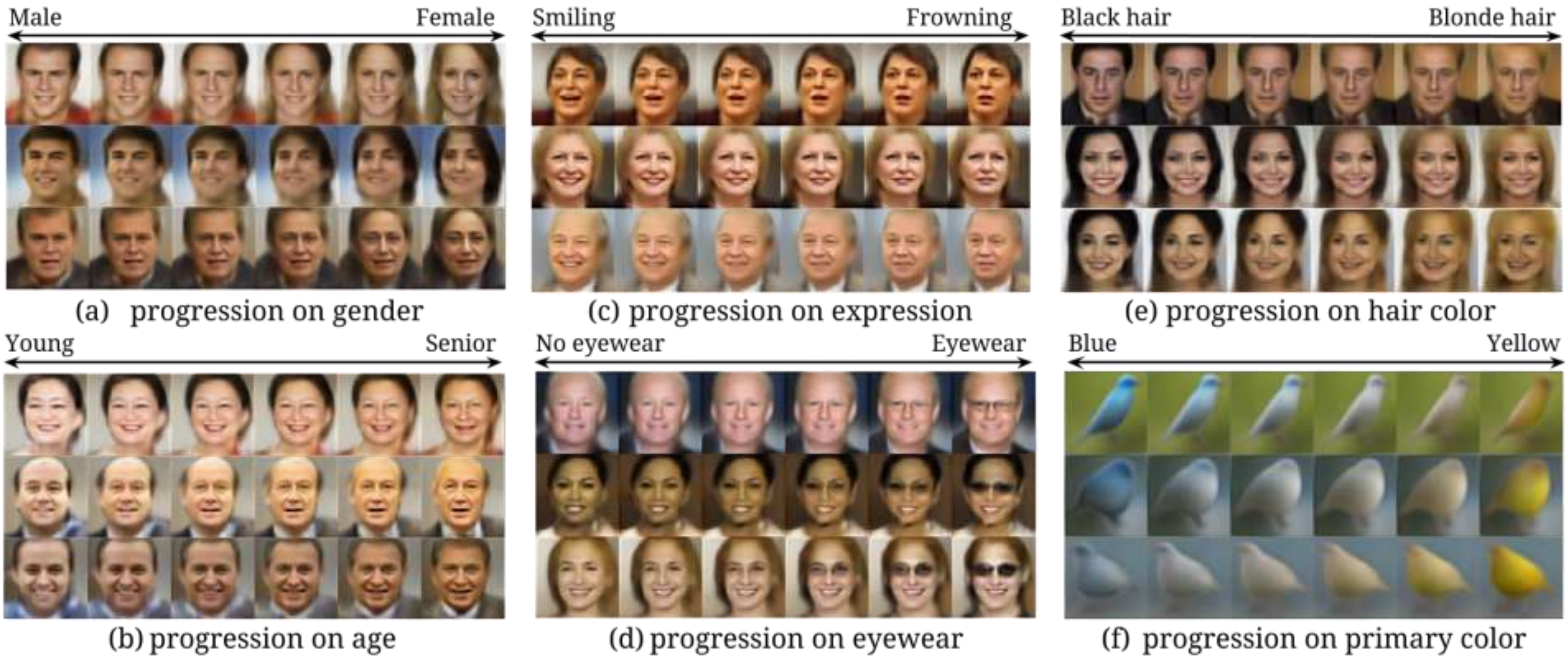
Vision task II – Attribute2Image

- Network structure



Vision task II – Attribute2Image

- Results



Problems of VAE

- Model capacity

- Note that the VAE requires 2 tractable distributions to be used:
  - The prior distribution $p(z)$ must be easy to sample from
  - The conditional likelihood $p(x|z, \theta)$ must be computable
- In practice this means that the 2 distributions of interest are often simple, for example uniform, Gaussian, or even isotropic Gaussian

Problems of VAE

- Blurry images

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

- The samples from the VAE look blurry
- Three plausible explanations for this
  - Maximizing the likelihood
  - Restrictions on the family of distributions
  - The lower bound approximation

Problems of VAE

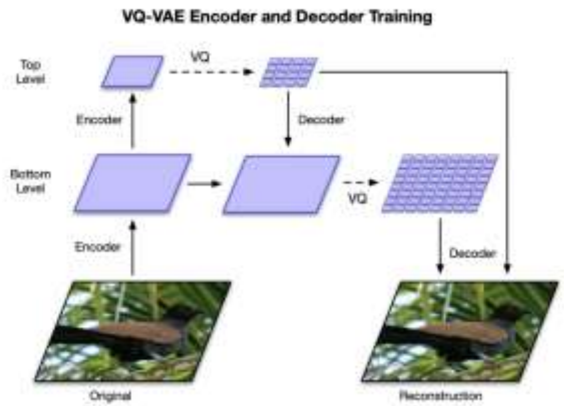
- Blurry images

  - Recent investigations suggest that both the simple probability distributions and the variational approximation lead to blurry images
  - Kingma & colleages: Improving Variational Inference with Inverse Autoregressive Flow
  - Zhao & colleagues: Towards a Deeper Understanding of Variational Autoencoding Models
  - Nowozin & colleagues: f-gan: Training generative neural samplers using variational divergence minimization

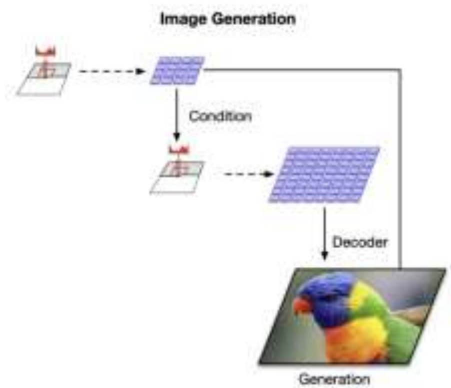
VQ-VAE

- Vector-Quantized Variational Autoencoder
- Combining VAE + Autoregressive: VQ-VAE1,2

Train a VAE-like model to generate multiscale grids of latent codes



Use a multiscale PixelCNN to sample in latent code space



Aaron van den Oord et al, “Neural Discrete Representation Learning”, NeurIPS 2017
Razavi et al, “Generating Diverse High-Fidelity Images with VQ-VAE-2”, NeurIPS 2019

VQ-VAE

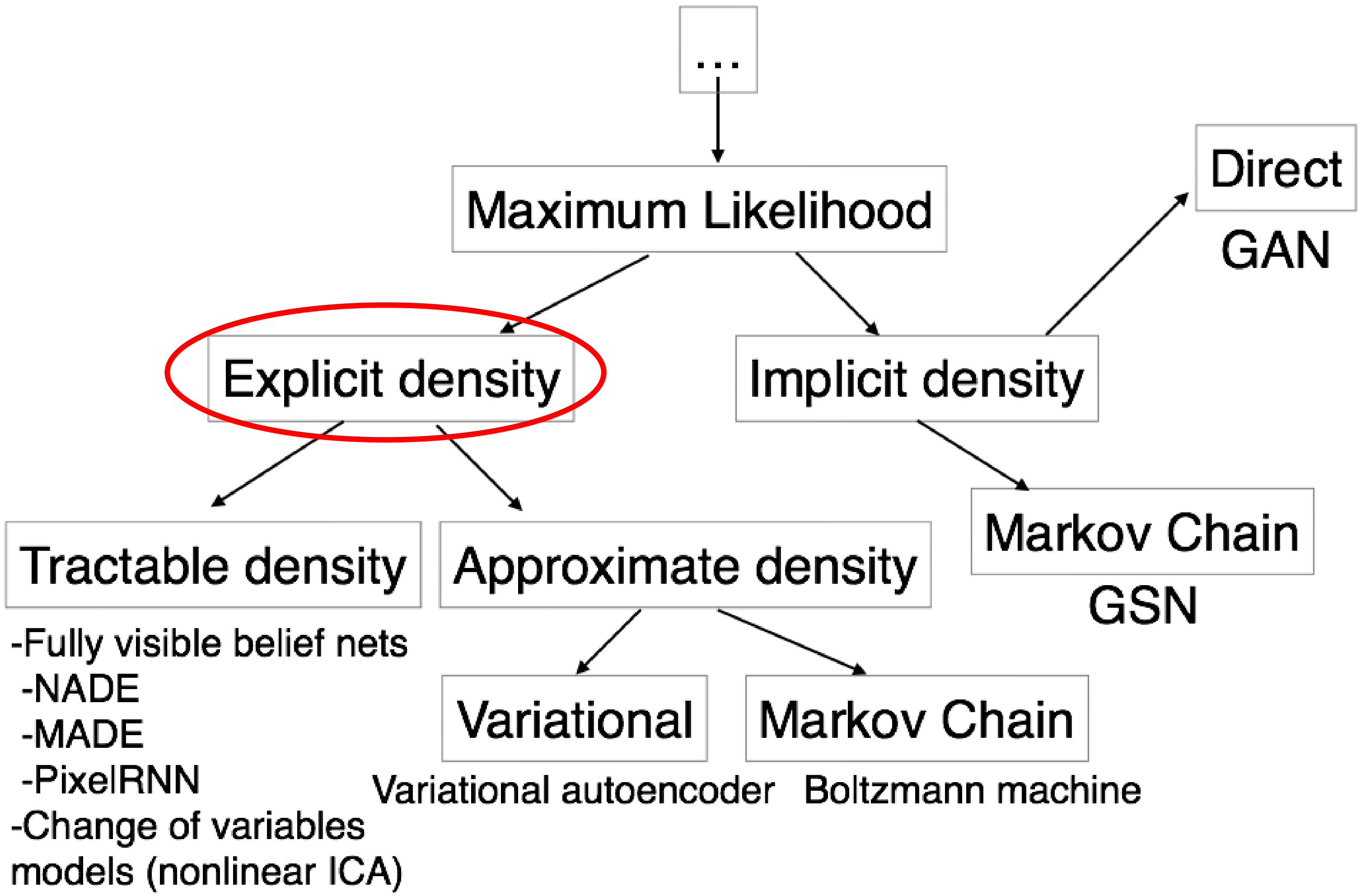
- 1024 x 1024 generated faces, trained on FFHQ

Outline

- Vision applications of VAEs
  - Inverse graphics network
  - Attribute2Image
- Generative Adversarial Networks
  - Implicit generative models
  - Adversarial learning

Acknowledgement: Feifei Li et al's cs231n notes

Taxonomy of Generative Models



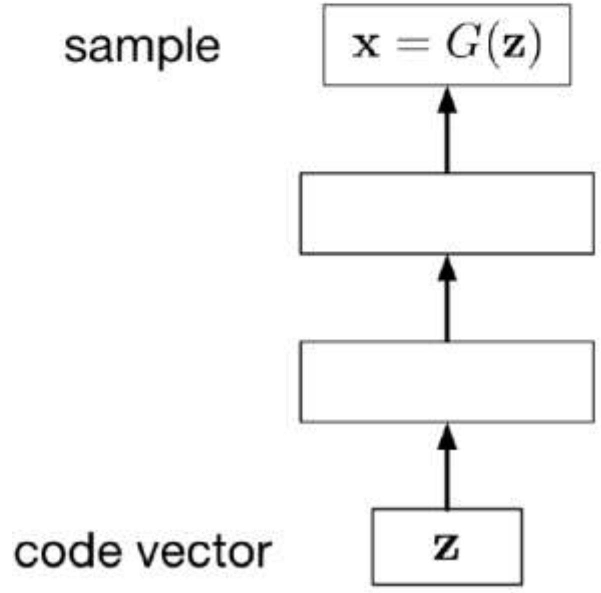
Implicit Generative Models

- Working with explicit model p(x) could be expensive
  - Variational Autoencoder (variational inference)
  - Boltzmann Machines (MCMC)

- Representation learning may not require p(x)
  - Sometimes we are more interested in taking samples from p(x) instead of p itself

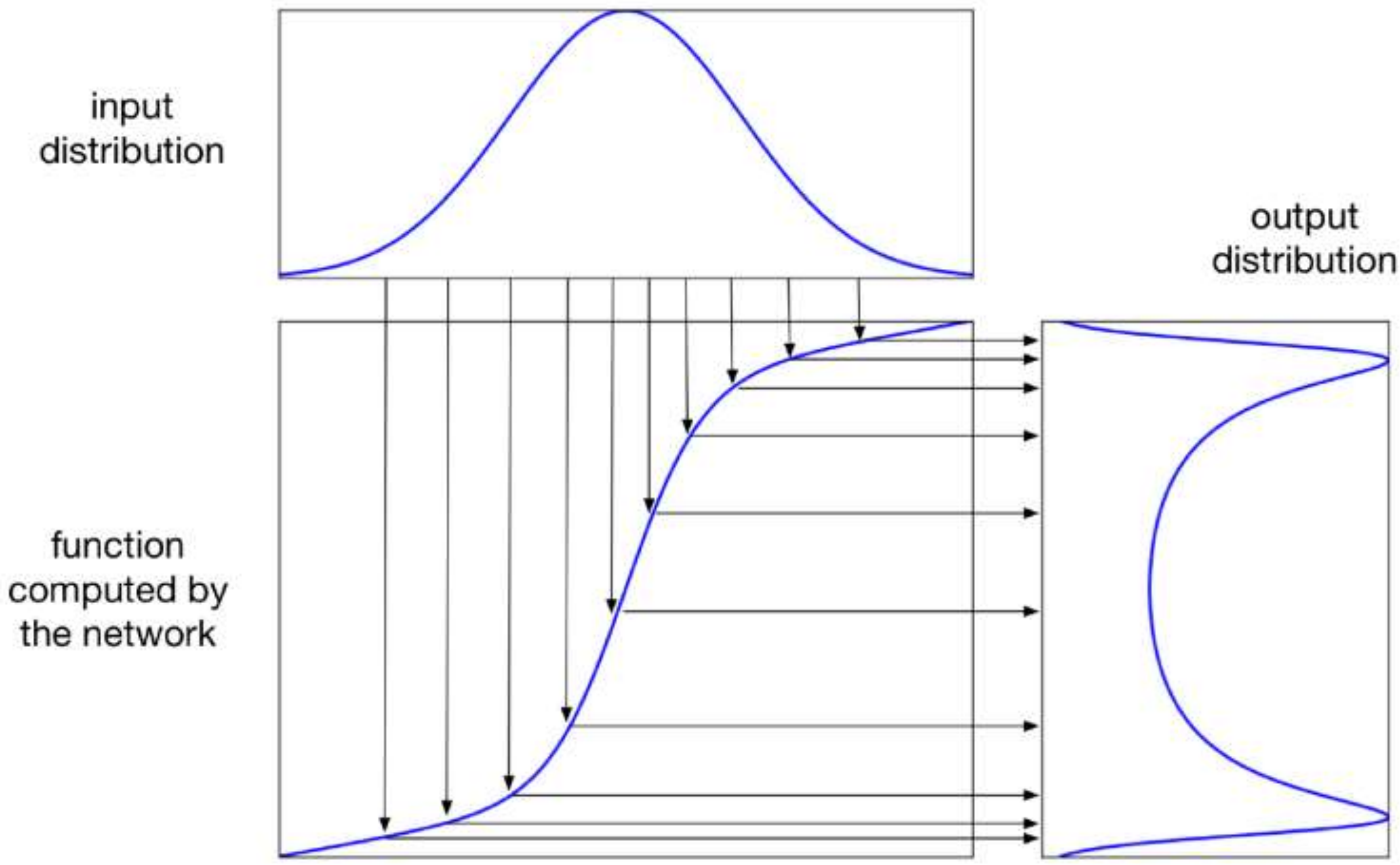
Implicit Generative Models

- Implicitly define a probability distribution
- Start by sampling the code vector z from a fixed, simple distribution
- A generator network computes a differentiable function G mapping z to an x in data space



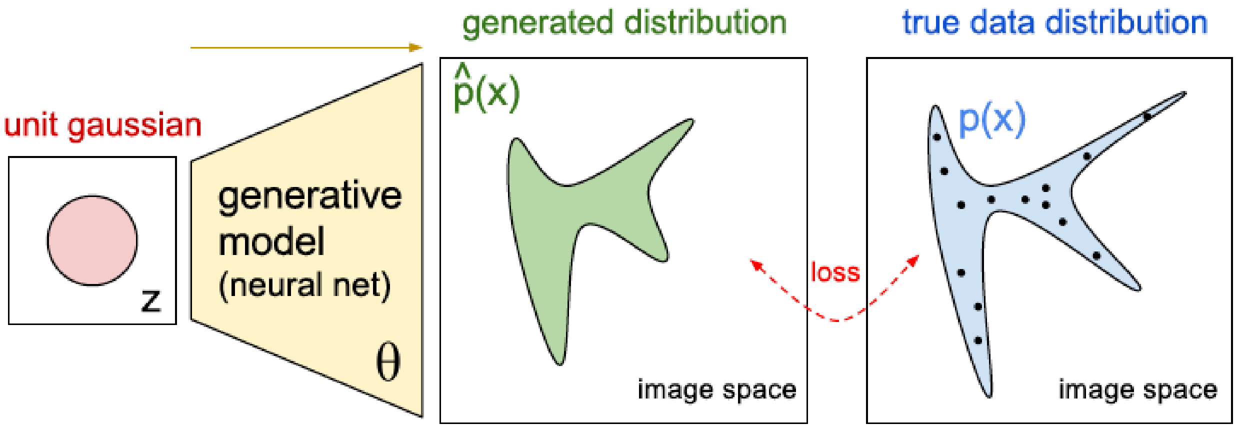
Implicit Generative Models

- Intuition: 1D example



Implicit Generative Models

- Intuition

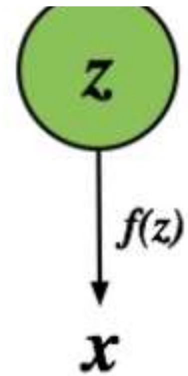


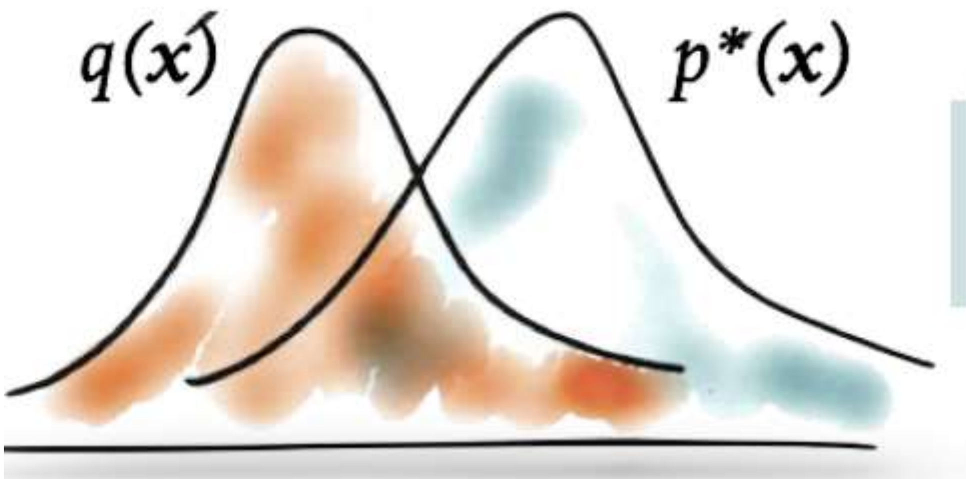
advocate/penalize samples within the blue/white region.

Learning by comparison

- Basic idea

For some models, we only have access to an unnormalised probability, partial knowledge of the distribution, or a simulator of data.

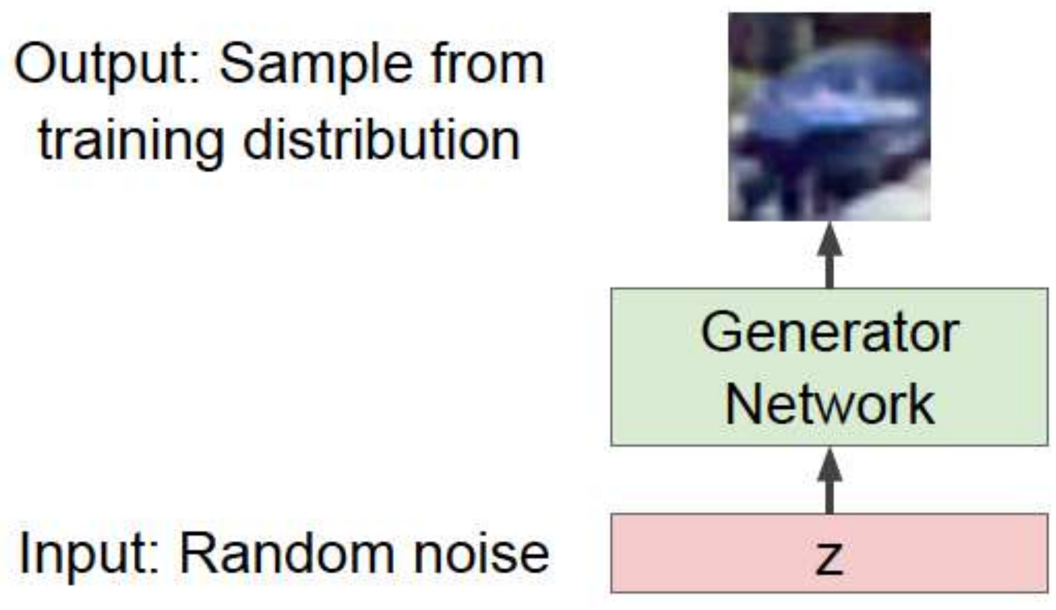




We compare the estimated distribution q(x) to the true distribution $p^*(x)$ using samples.

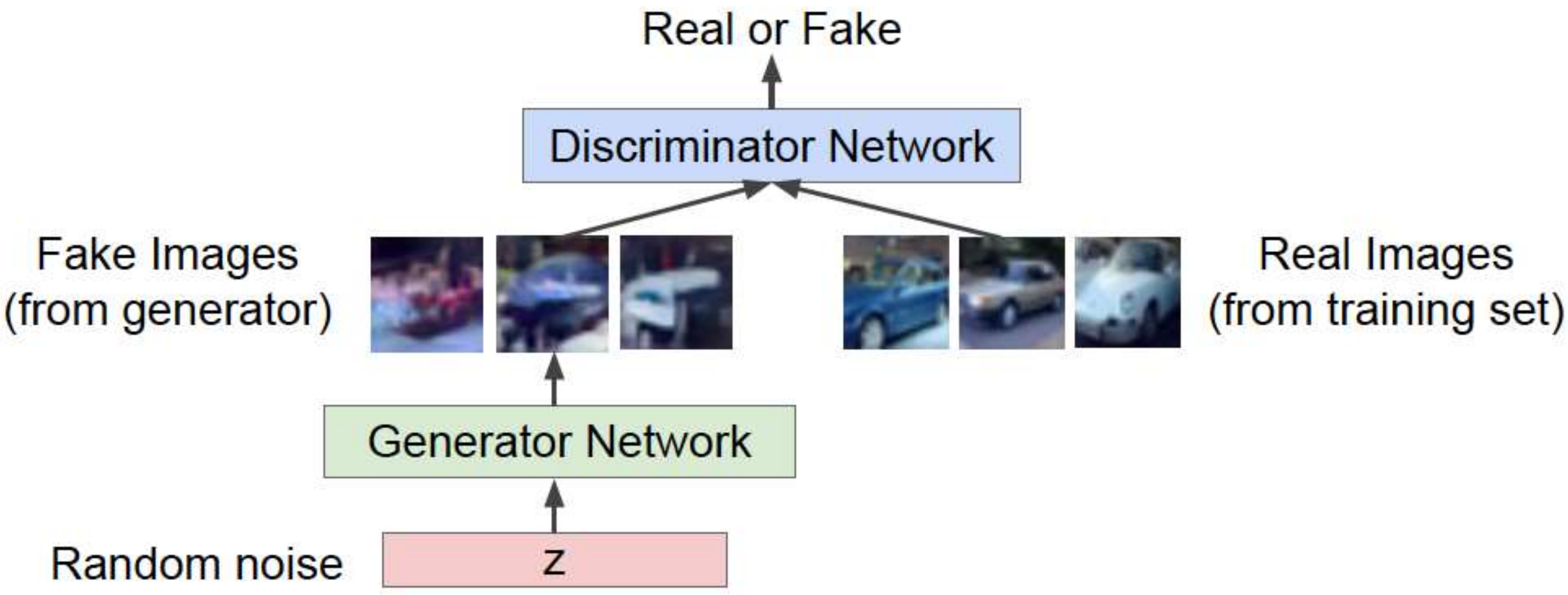
Generative Adversarial Networks

- Using a neural network to generate data



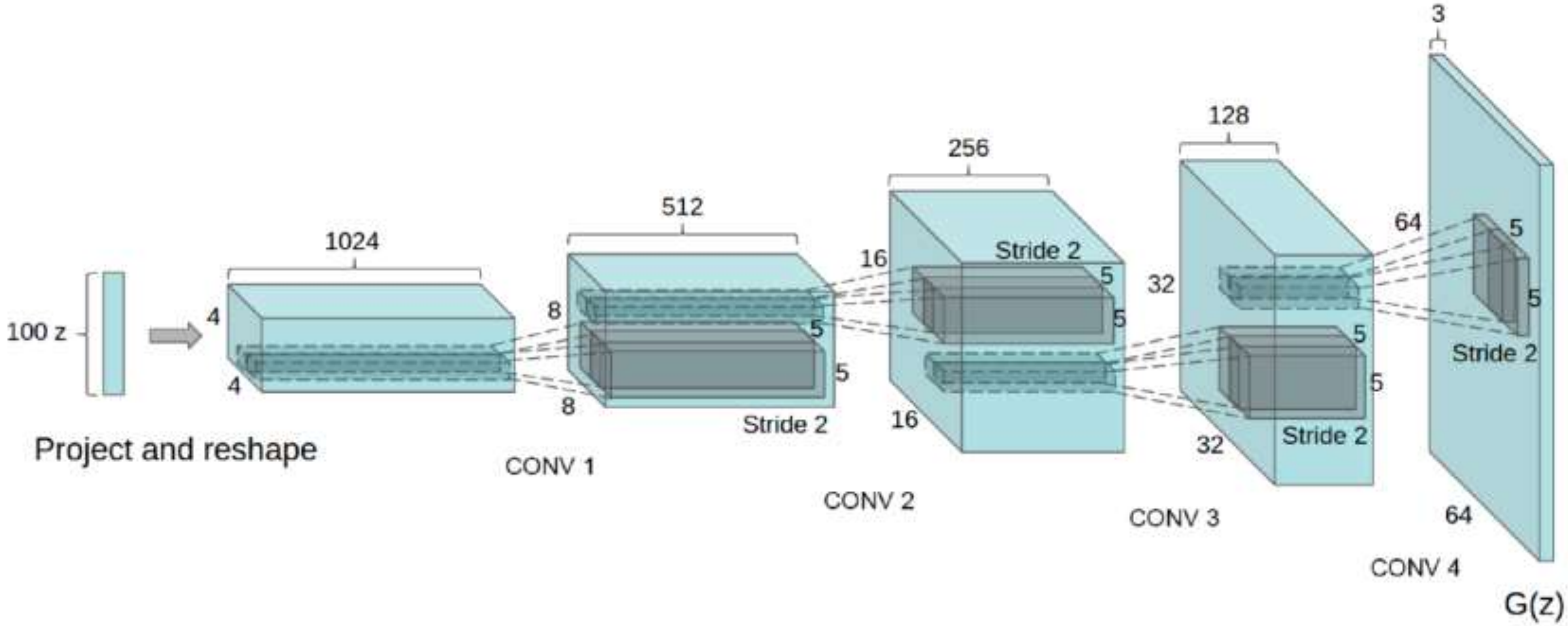
Generative Adversarial Networks

- Using another neural network to determine if the data is real or not



Typical generator architecture

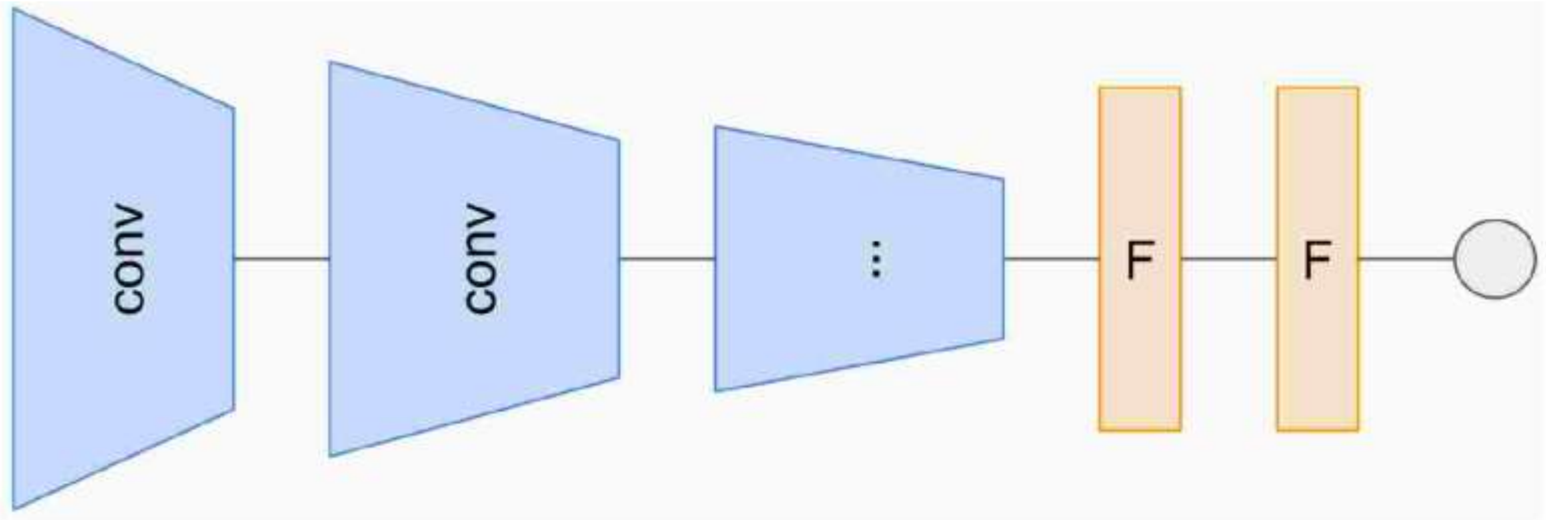
- For images



- Unit Gaussian distribution on z, typically 10-100 dim.
- Up-convolutional deep network (reverse recognition CNN)

Typical discriminator architecture

- For images



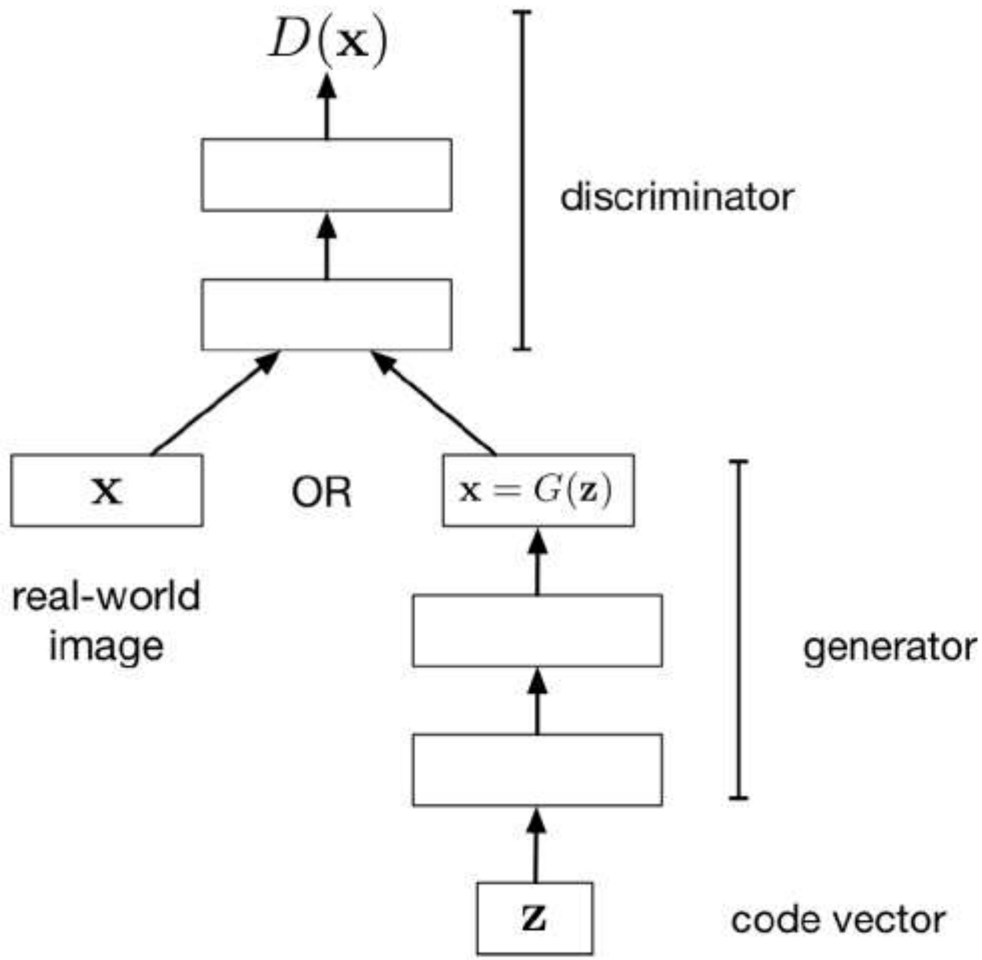
- Recognition CNN model
- Binary classification output: real / synthetic

Adversarial learning

- GAN objective for the generator is some complicated objective function defined by a neural network.
  - This means a new way of thinking about "distance".
  - We are training networks to minimize the "distance" or "divergence" between generated images and real images.
  - Instead of some hand-crafted distance metric like L1 or L2, we can make something completely new.
  - *A neural network, with the right architecture, is arguably the definition of perceptual similarity (assuming our visual system is some sort of neural network).*

Adversarial Learning

- Adversarial loss



Adversarial Learning

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

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

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

$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

Two-player game

- Minimax formulation
  - The generator and discriminator are playing a zero-sum game against each other

$$\min_{G} \max_{D} \mathcal{J}_D$$

  - Using parametric models

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Learning procedure

- Minimax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

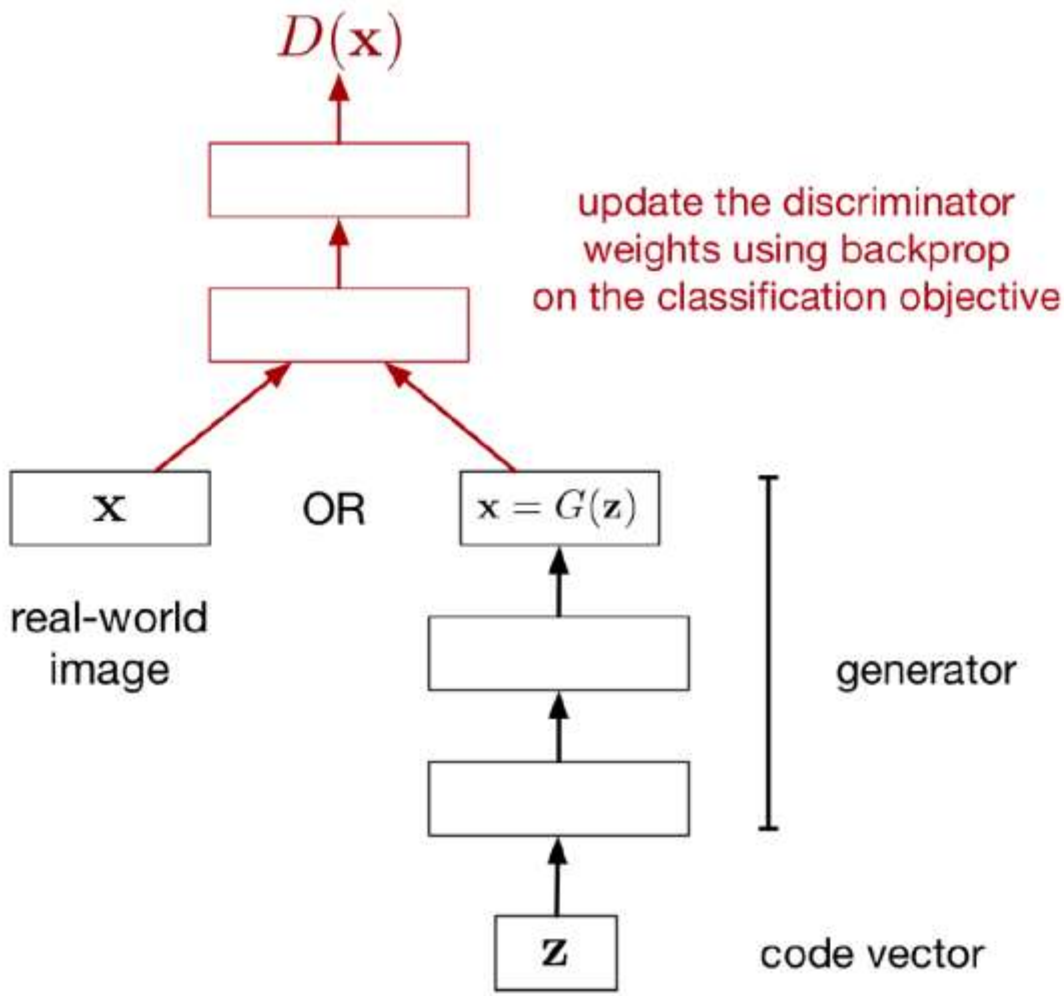
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

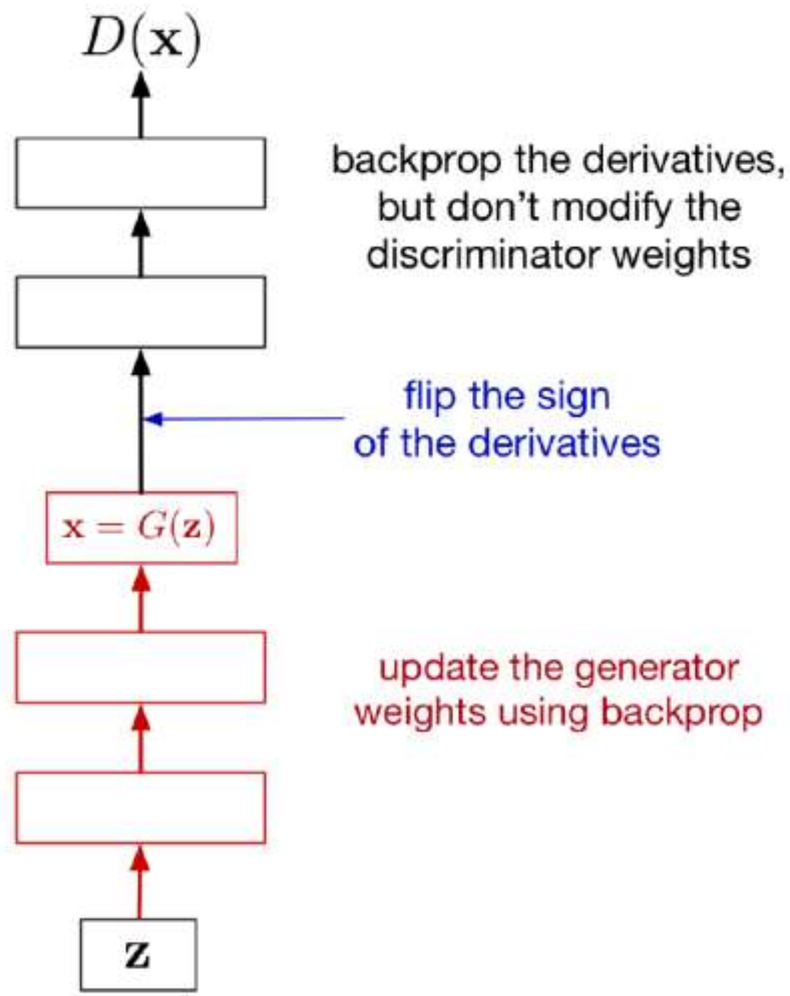
Learning procedure

- Updating the discriminator



Learning procedure

- Updating the generator

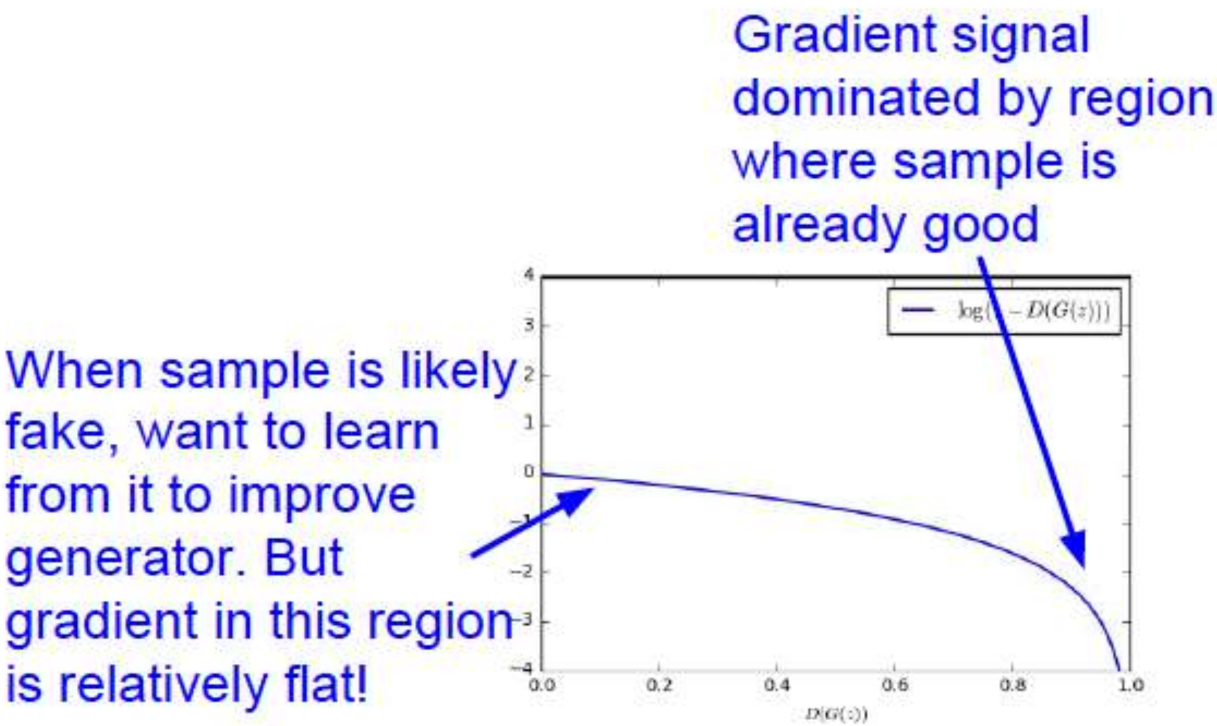


A better cost function

- The minimax cost function for the generator

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem is saturation



A better cost function

- Changing the generator cost

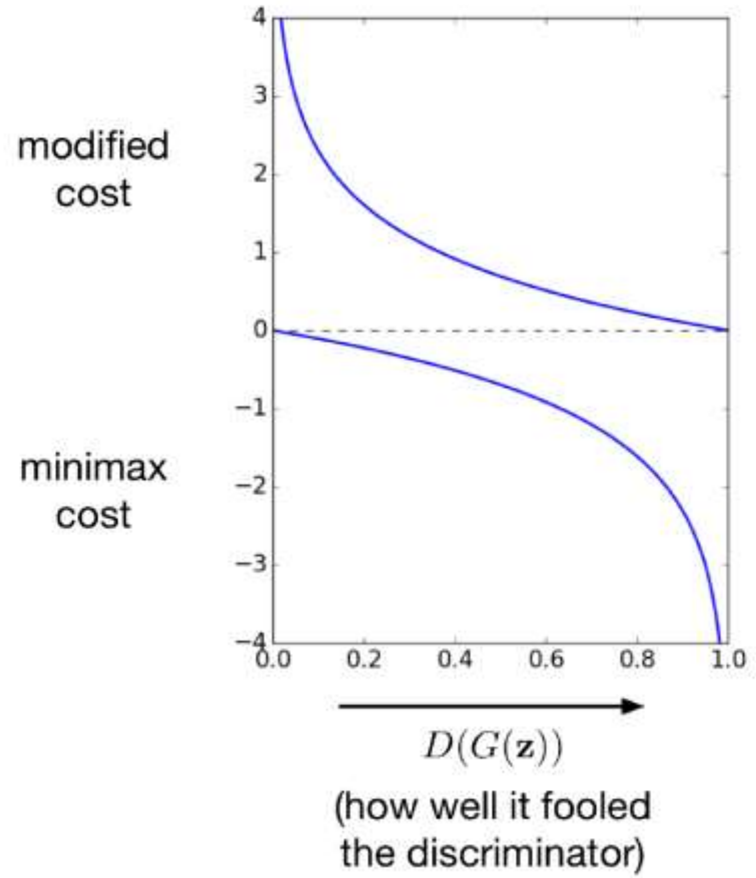
  - Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

  - Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

  - This fixes the saturation problem.



Theoretical property

- Adversarial loss

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim data}\log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \quad (1)$$

$$J^{(G)} = -J^{(D)} \quad (2)$$

- The optimal discriminator $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_{model}(x)}$.
- In this case, $J^{(G)} = 2D_{JS}(p_{data}||p_{model}) + const.$
- Jenson-Shannon divergence:
  $D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$.

Theoretical property

- Adversarial loss for the optimality

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{x \sim p_G} [\log(1 - D(x))] \right)$$

$$= \min_{G} \int_{X} \max_{D} \left(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)) \right) dx$$

Theoretical property

- Adversarial loss for the optimality

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

$$= \min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))]\right)$$

$$= \min_G \int_X \max_D \left(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))\right) dx$$

$$f(y) = a \log y + b \log(1 - y) \qquad f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b} \text{ (local max)}$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

Optimal Discriminator: $\boxed{D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}}$

Theoretical property

- Adversarial loss for the optimality

$$\boxed{D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}}$$

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]\right)$$

$$= \min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))]\right)$$

$$= \min_G \int_X \left(p_{data}(x)\log D_G^*(x) + p_G(x)\log(1 - D_G^*(x))\right)dx$$

$$= \min_G \int_X \left(p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x)\log\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right)dx$$

Theoretical property

- Adversarial loss for the optimality

$$\boxed{D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}}$$

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right]\right)$$

$$= \min_G \int_X \left(p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x)\log\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right)dx$$

$$= \min_G \left(E_{x\sim p_{data}}\left[\log\frac{2}{2}\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + E_{x\sim p_G}\left[\log\frac{2}{2}\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right]\right)$$

$$= \min_G \left(E_{x\sim p_{data}}\left[\log\frac{2*p_{data}(x)}{p_{data}(x)+p_G(x)}\right] + E_{x\sim p_G}\left[\log\frac{2*p_G(x)}{p_{data}(x)+p_G(x)}\right] - \log 4\right)$$

Theoretical property

- Adversarial loss for the optimality

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$$

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

$$= \min_G \left(E_{x\sim p_{data}}\left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x\sim p_G}\left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)}\right] - \log 4\right)$$

$$= \min_G \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2}\right) + KL\left(p_G, \frac{p_{data} + p_G}{2}\right) - \log 4\right)$$

$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

Jensen-Shannon Divergence:

$$JSD(p,q) = \frac{1}{2}KL\left(p, \frac{p+q}{2}\right) + \frac{1}{2}KL\left(q, \frac{p+q}{2}\right)$$

Kullback-Leibler Divergence:

$$KL(p,q) = E_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$$

Theoretical property

- Adversarial loss for the optimality

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$
$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

- Summary: the global minimum of the minimax game happens when:

1. $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)

- Caveats:

1. G and D are neural nets → depend on the network optimization!
2. ``Theoretical'' convergence to the optimal solution,

Theoretical property

- Stationary point

There is a theoretical point in this game at which the game will be stable and both players will stop changing.

- If the generated data exactly matches the distribution of the real data, the generator should output 0.5 for all points (argmax of loss function)
- If the discriminator is outputting a constant value for all inputs, then there is no gradient that should cause the generator to update

We rarely reach a completely stable point in practice due to practical issues

Theoretical property

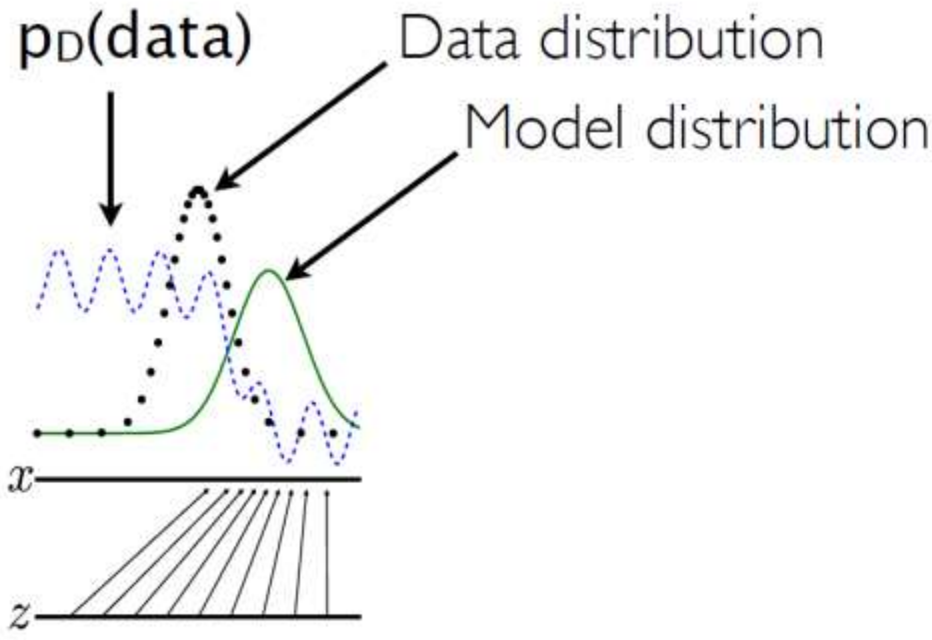
- Convergence

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

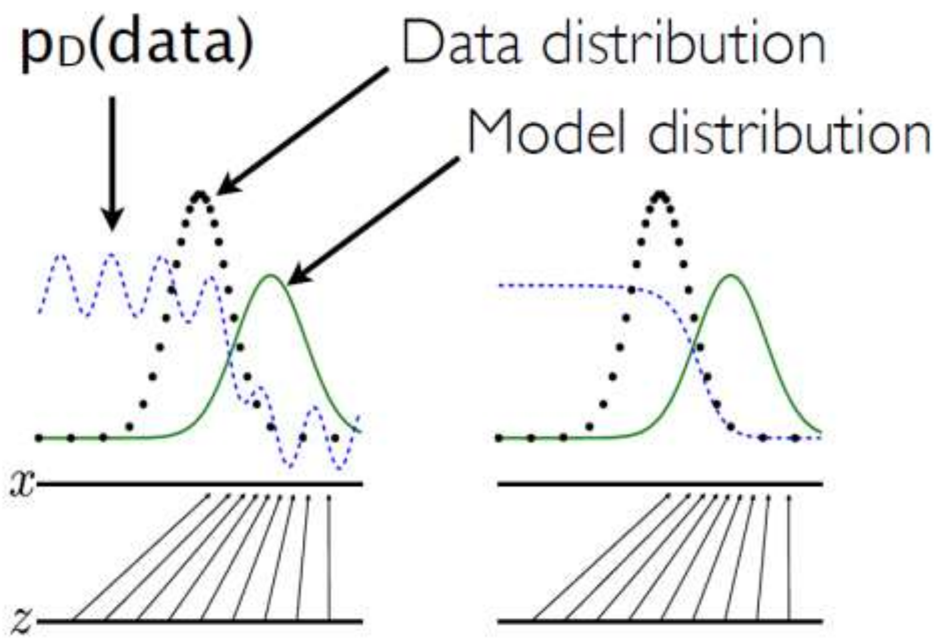
- Theoretical properties (assuming infinite data, infinite model capacity, direct updating of generator's distribution):
  - Unique global optimum.
  - Optimum corresponds to data distribution.
  - Convergence to optimum guaranteed.

If discriminator is finite and modest-sized, this message is incorrect. (regardless of training time, # samples, training objective etc..) See Sanjeev Arora, CVPR 2018 Tutorial

Training GANs

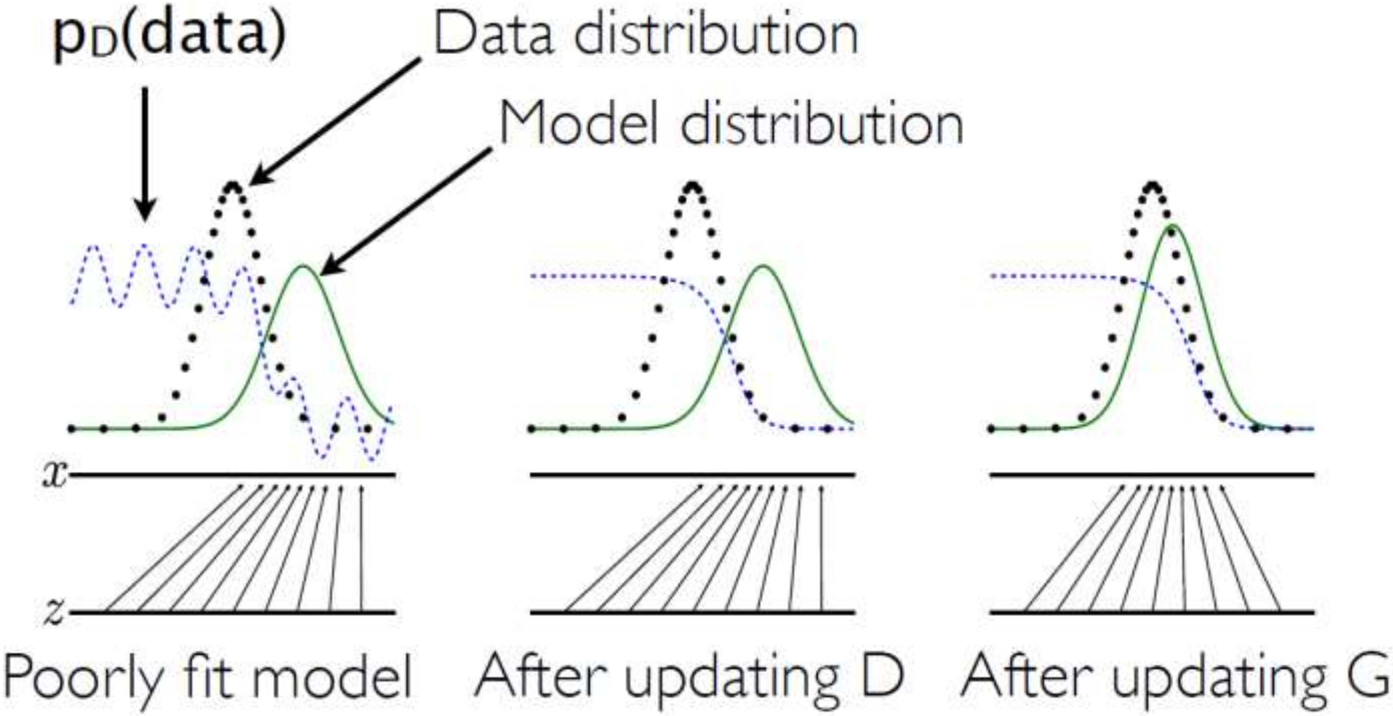


Training GANs

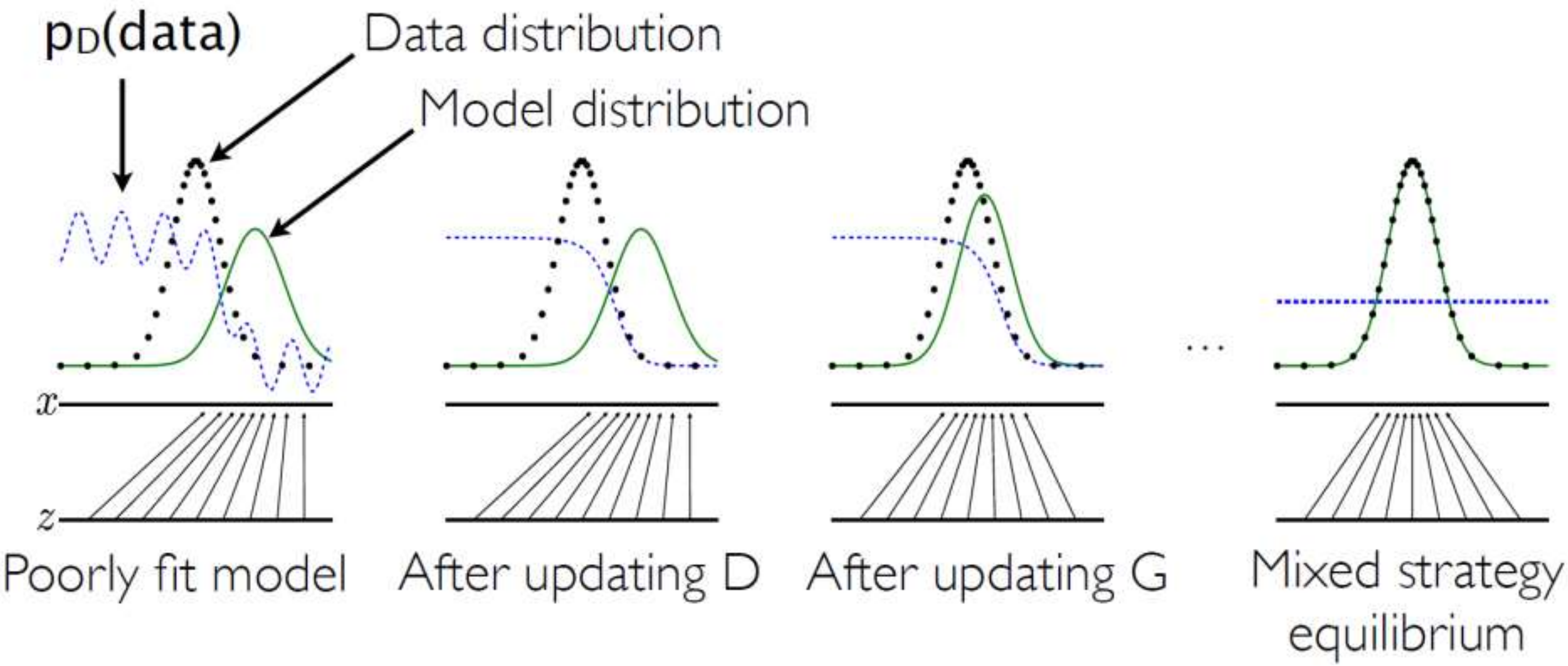


Poorly fit model   After updating D

Training GANs



Training GANs



Training GANs

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods
- GAN Zoo: https://github.com/hindupuravinash/the-gan-zoo
- In general, training a GAN is tricky and unstable
- Many tricks:
  - S. Chintala, How to train a GAN, ICCV 2017 tutorial
  - https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf

Generated Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Generated Samples

Objects:

The GAN Zoo

- https://github.com/hindupuravinash/the-gan-zoo

  - 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
  - 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
  - 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
  - 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
  - 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
  - ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
  - ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
  - AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
  - acGAN - Face Aging With Conditional Generative Adversarial Networks
  - ACGAN - Coverless Information Hiding Based on Generative adversarial networks
  - acGAN - On-line Adaptative Curriculum Learning for GANs
  - ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
  - AdaGAN - AdaGAN: Boosting Generative Models
  - Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
  - AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
  - AdvGAN - Generating adversarial examples with adversarial networks
  - AE-GAN - AE-GAN: adversarial eliminating with GAN
  - AE-OT - Latent Space Optimal Transport for Generative Models
  - AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
  - AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
  - AffGAN - Amortised MAP Inference for Image Super-resolution
  - AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization

The GAN Zoo

- Use various Divergence
- https://arxiv.org/abs/1606.00709

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert\,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1-\log(-t)$
Pearson χ^2	$\frac{1}{4}t^2+t$
Neyman χ^2	$2-2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$
Jensen-Shannon	$-\log(2-\exp(t))$
Jensen-Shannon-weighted	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1-\exp(t))$ 37

Summary

- Variational Autoencoders (VAEs)
  - VAE objective
  - VAE: Vision applications
- Generative Adversarial Networks (GAN)
  - Adversarial learning
- Next time:
  - GAN going on
- Reading material
  - https://arxiv.org/pdf/1503.03167.pdf (Deep Convolutional Inverse Graphics Network)
  - https://arxiv.org/pdf/2011.10063.pdf (Dual Contradistinctive Generative Autoencoder)
  - https://taesung.me/SwappingAutoencoder/ (Swapping Autoencoder for Deep Image Manipulation)