Lecture 16: Deep Generative Models II: AE & VAE

Lan Xu

SIST, ShanghaiTech

Fall, 2021

Outline

- Representation learning
  - AutoEncoder
- Variational Autoencoders (VAEs)
  - VAE objective
  - Reparametrization trick
  - Connection to Auto-Encoders

Acknowledgement: Feifei Li et al's cs231n notes

Recall EM GMM

- MLE: maximizing the log-likelihood

$$\ell(\theta) = \sum_{i=1}^{n} \log p(x^{(i)};\theta)$$

- ELBO: Evidence Lower Bound

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int_z p(\mathbf{x}, z) \\
&= \log \int_z p(\mathbf{x}, z) \frac{q(\mathbf{z})}{q(\mathbf{z})} \\
&= \log(E_q[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}]) \\
&\geq E_q[\log p(\mathbf{x}, \mathbf{z})] - E_q[\log q(\mathbf{z})] \\
&= ELBO
\end{aligned}$$

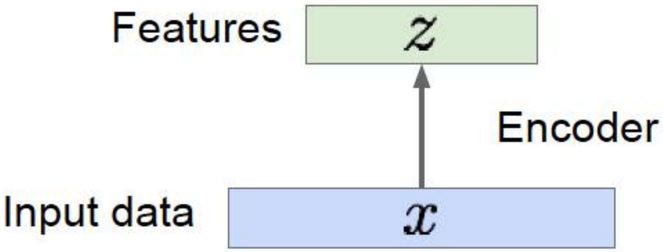
$$\begin{aligned}
KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= E_q[\log q(\mathbf{z})] - E_q[\log p(\mathbf{z}|\mathbf{x})] \\
&= E_q[\log q(\mathbf{z})] - E_q[\log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x}) \\
&= \log p(\mathbf{x}) - (E_q[\log p(\mathbf{z}, \mathbf{x})] - E_q[\log q(\mathbf{z})]) \\
&= \log p(\mathbf{x}) - ELBO
\end{aligned}$$

$$\begin{aligned}
EBLO &= E_q[\log p(\mathbf{x}, \mathbf{z})] - E_q[\log q(\mathbf{z})] \\
&= E_q[\log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z})}] - E_q[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}] \\
&= E_q[\log p(\mathbf{x}|\mathbf{z})] - KL(q(\mathbf{z})||p(\mathbf{z}))
\end{aligned}$$

Autoencoder

- Feature representation learning

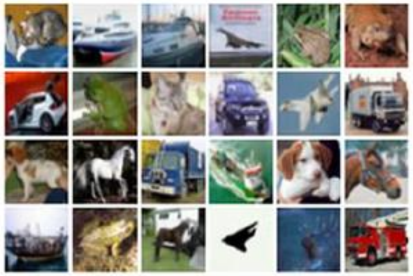
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



Features z

Encoder

Input data x

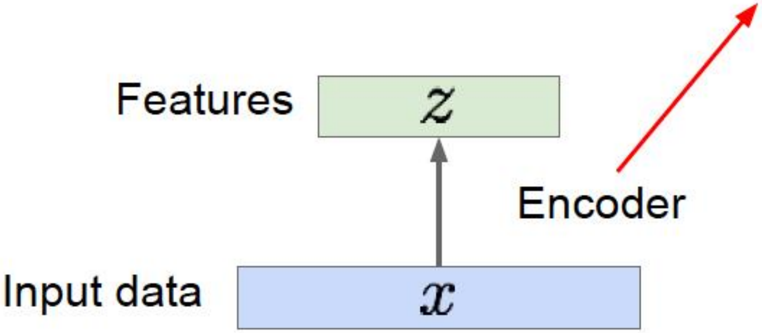


Autoencoder

- Feature representation learning

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Features z

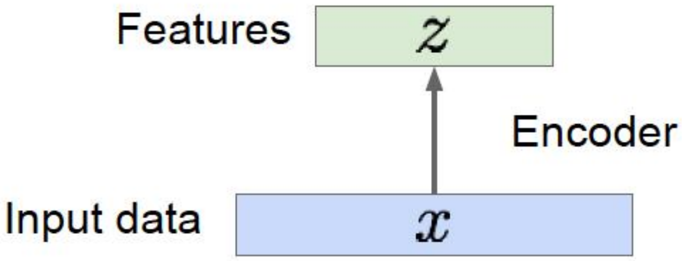
Encoder

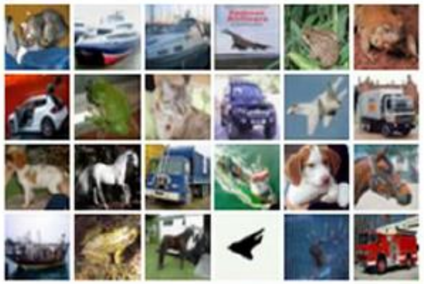
Input data x

Autoencoder

- Feature representation learning

How to learn this feature representation?

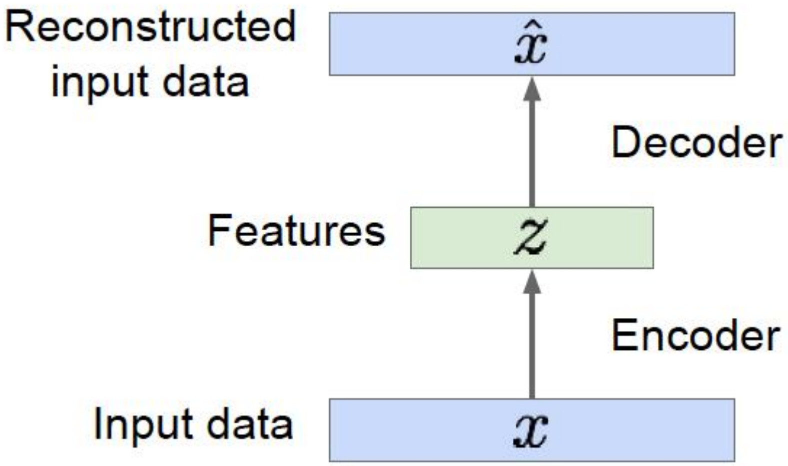




Autoencoder

- Feature representation learning

How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself



Reconstructed input data

$\hat{x}$

Decoder

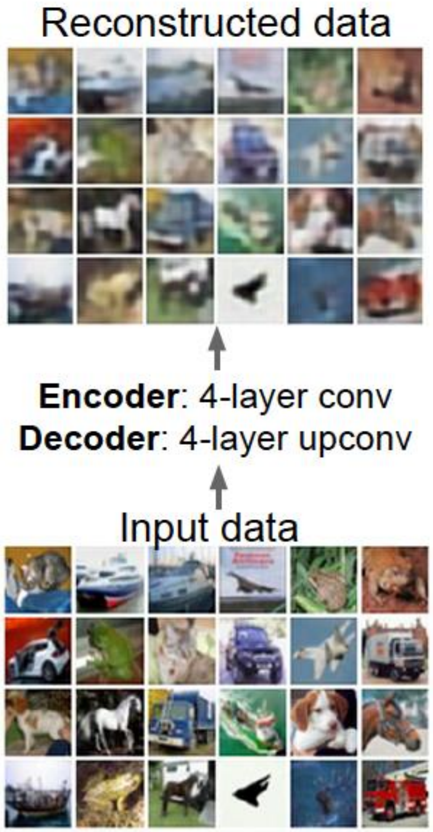
Features

z

Encoder

Input data

x



Reconstructed data

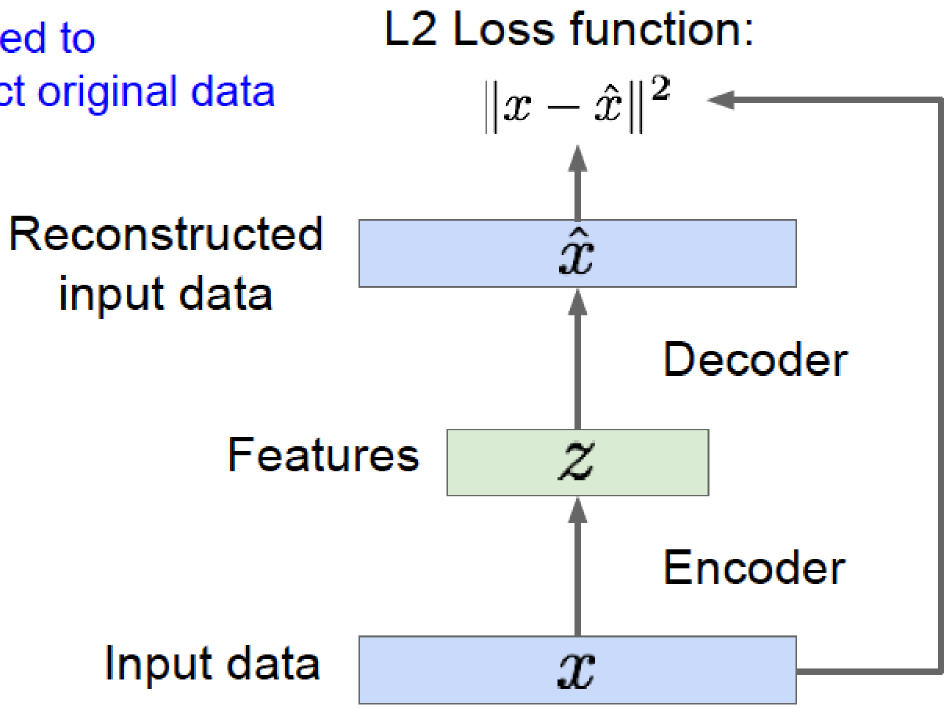
Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Autoencoder

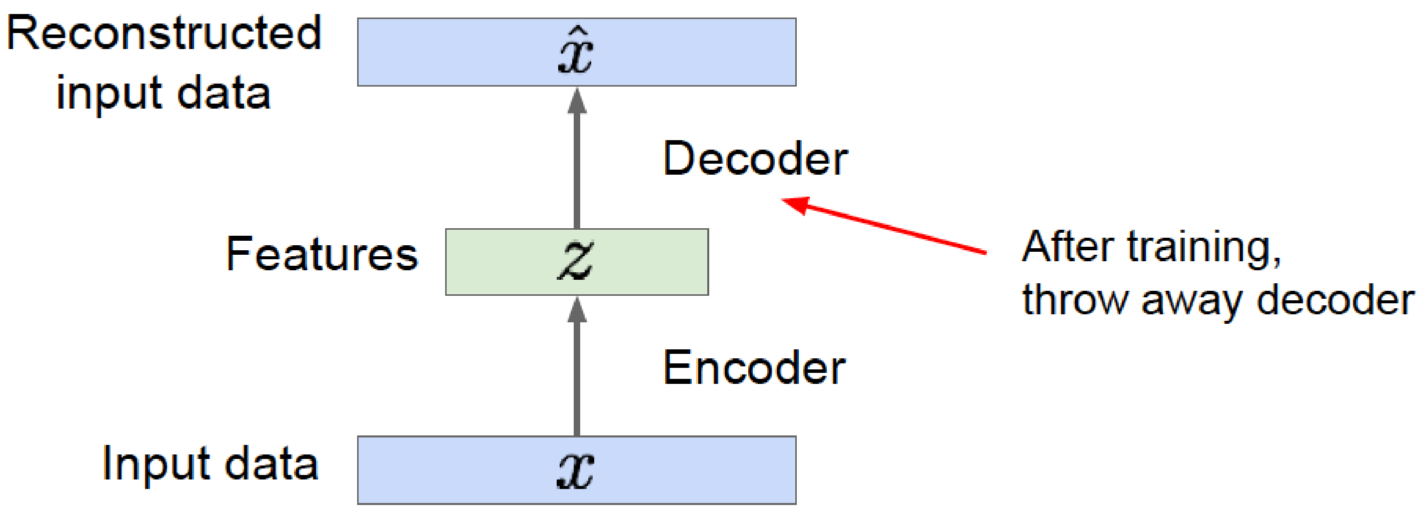
- Feature representation learning

Train such that features can be used to reconstruct original data



Autoencoder

- Feature representation learning



Reconstructed input data

$\hat{x}$

Decoder

Features

z

Encoder

Input data

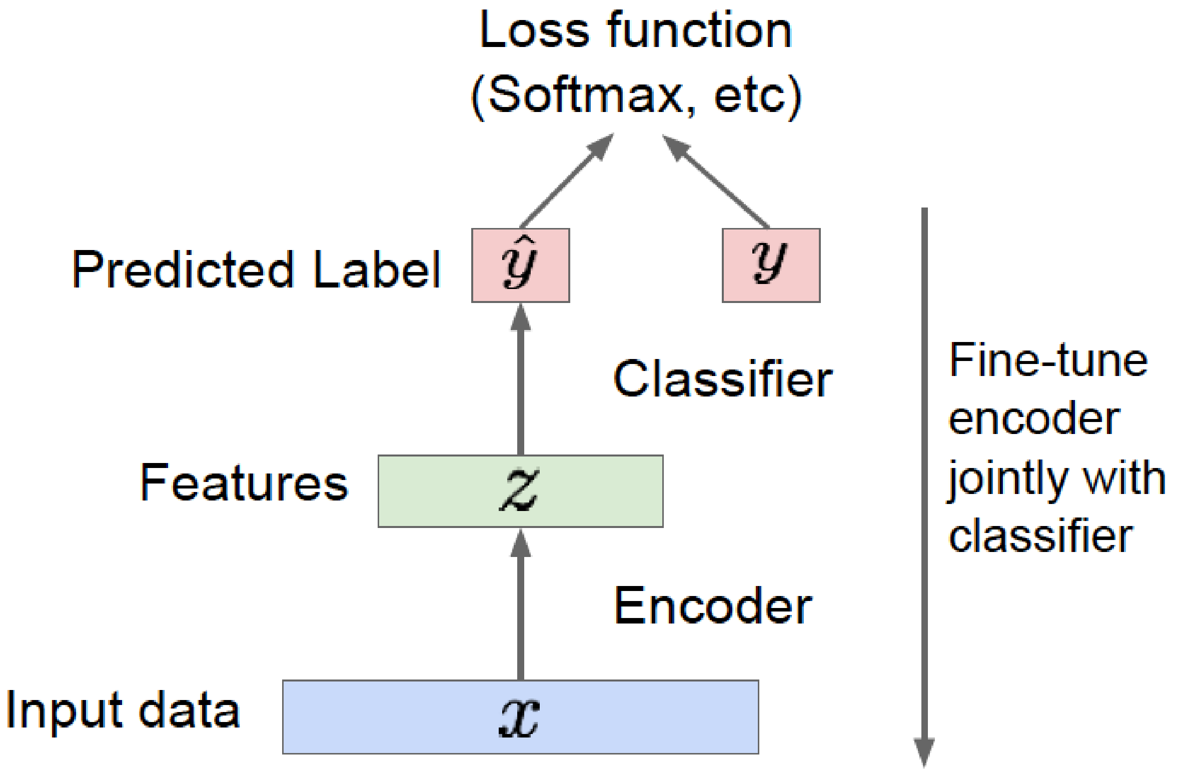
x

After training, throw away decoder

Autoencoder

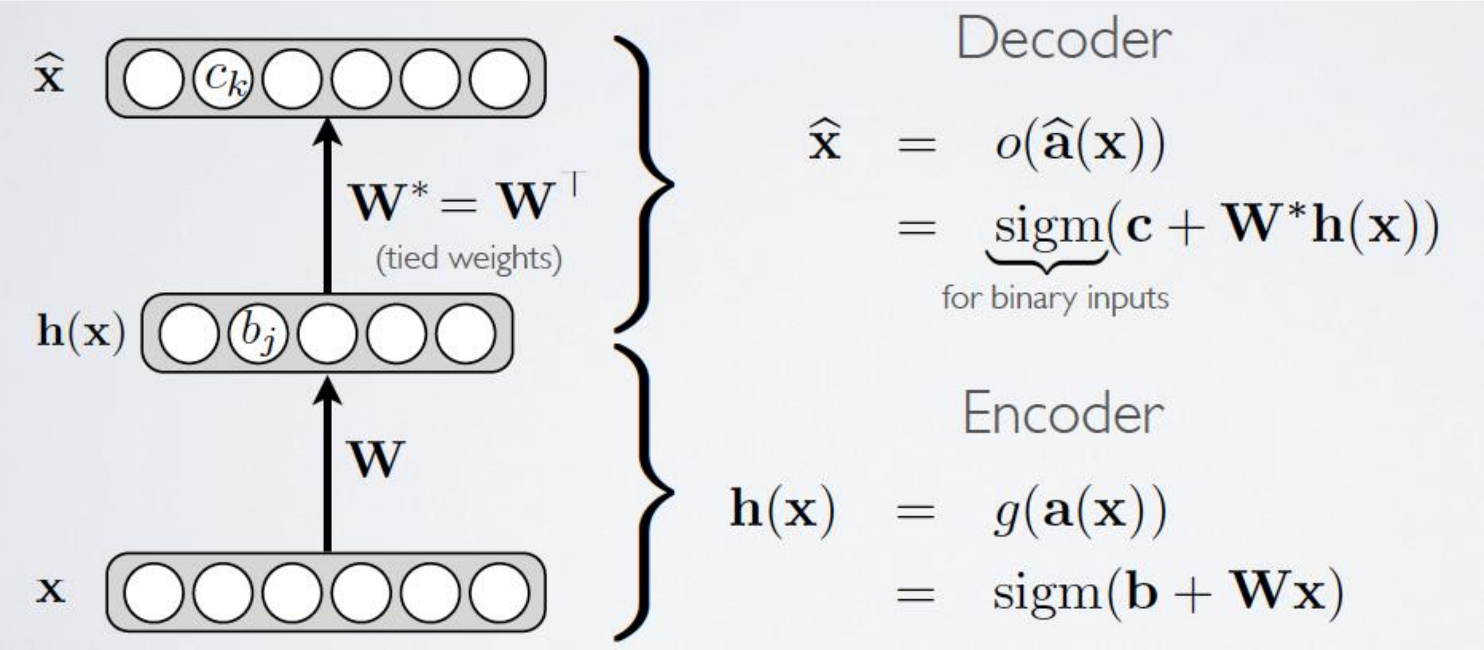
- Feature representation learning

Encoder can be used to initialize a **supervised** model



Autoencoder

- Binary input example

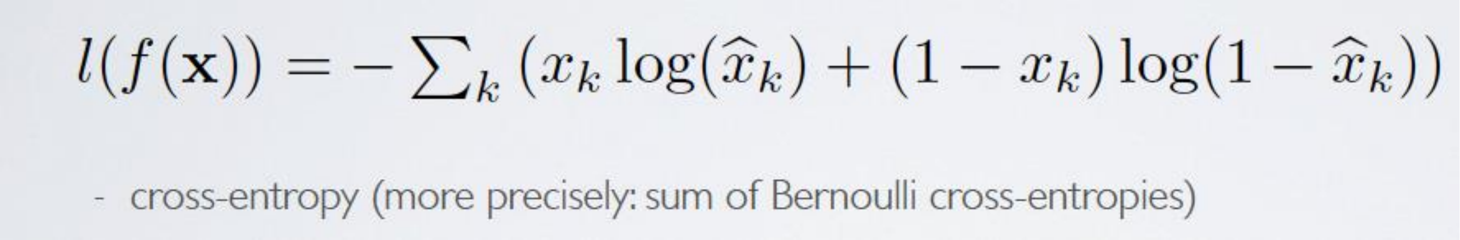


Decoder

$$\begin{aligned}\widehat{\mathbf{x}} &= o(\widehat{\mathbf{a}}(\mathbf{x})) \\ &= \underbrace{\mathrm{sigm}}_{\text{for binary inputs}}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))\end{aligned}$$

Encoder

$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\ &= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$



$$l(f(\mathbf{x})) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k)\log(1 - \widehat{x}_k)\right)$$

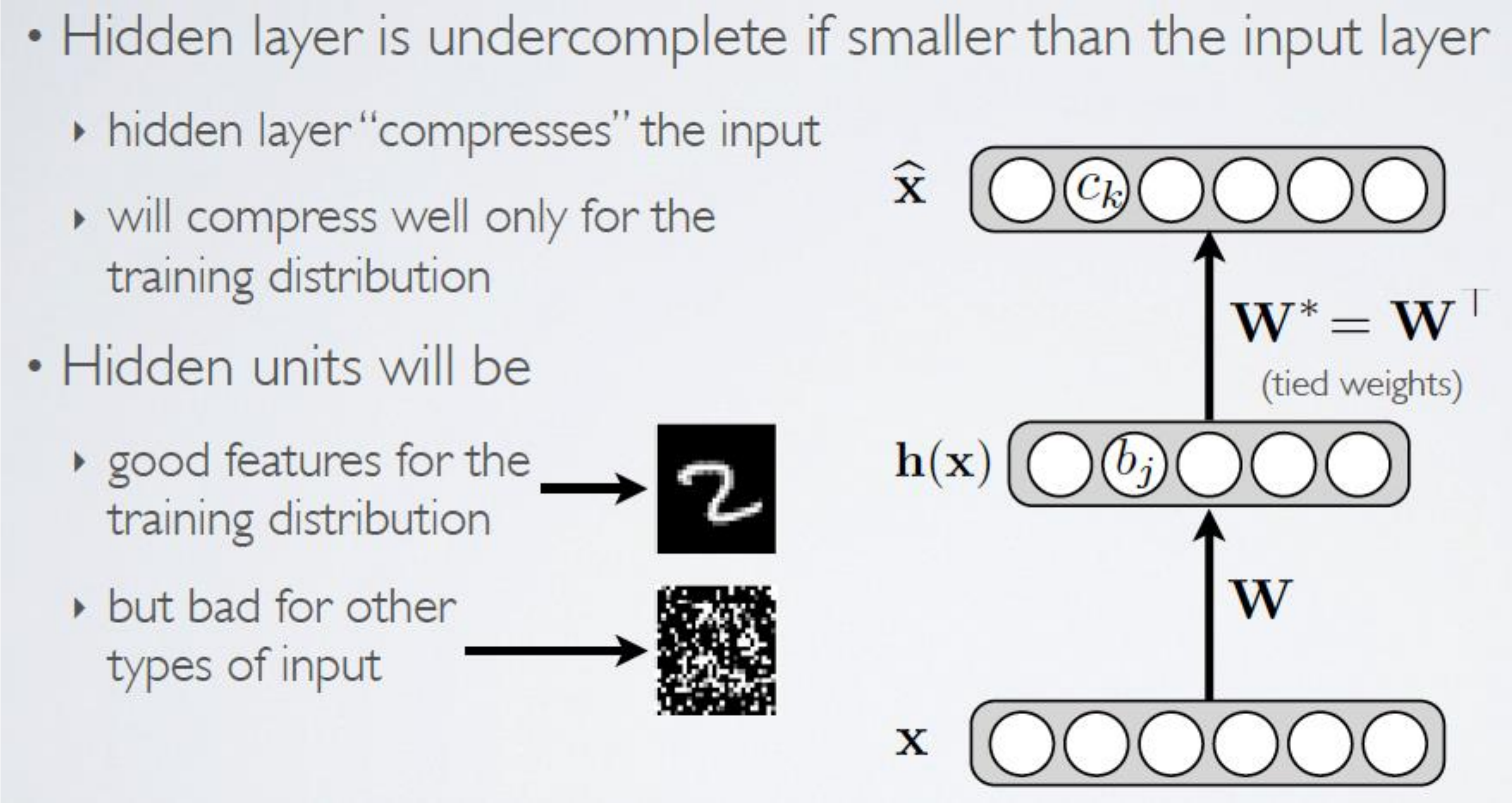
- cross-entropy (more precisely: sum of Bernoulli cross-entropies)

Regularization

- Regularized autoencoders: add regularization term that encourages the model to have other properties
  - Sparsity of the representation (sparse autoencoder)
  - Robustness to noise or to the missing inputs (denoising autoencoder)
  - Smallness of the derivative of the representation (contracitve autoencoder)

Regularization

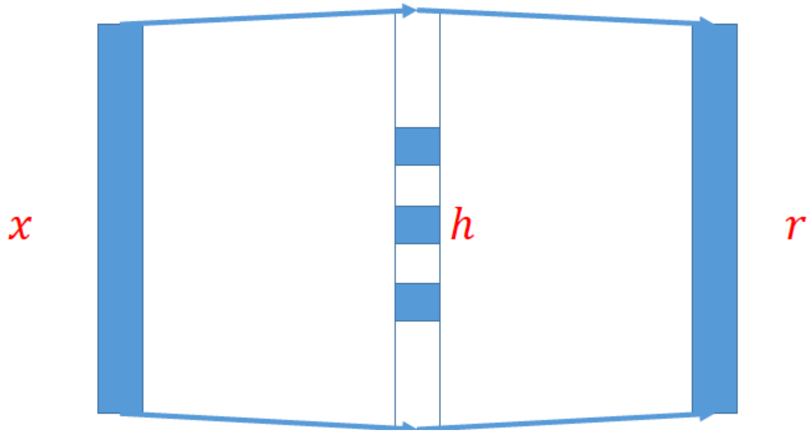
- Undercomplete representation



- Hidden layer is undercomplete if smaller than the input layer
  - hidden layer "compresses" the input
  - will compress well only for the training distribution
- Hidden units will be
  - good features for the training distribution →
  - but bad for other types of input →

Regularization

$$L_R = L(x, g(f(x))) + R(h)$$

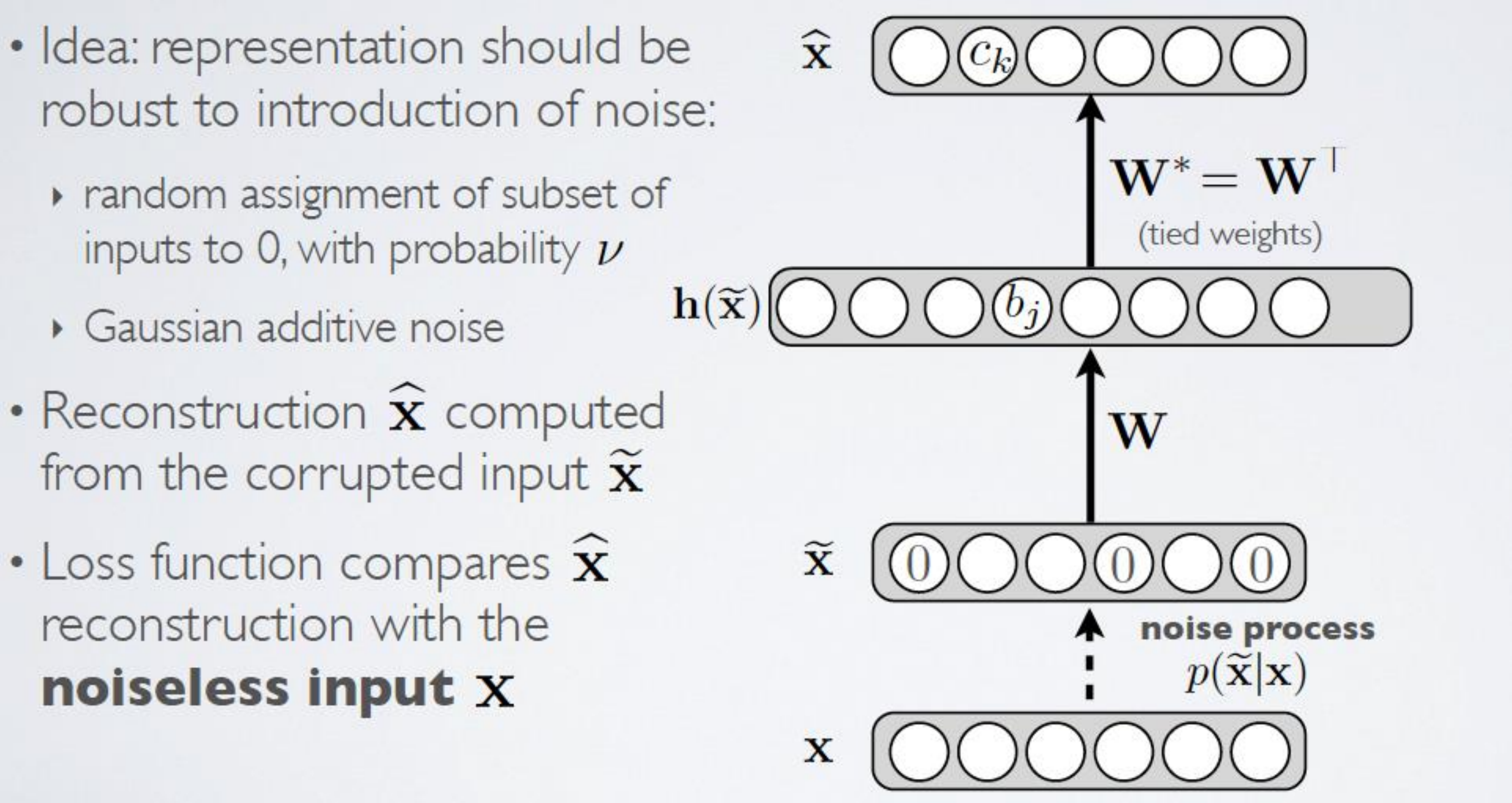


- Sparse autoencoder
  - □ Constrain the code to have sparsity
  - □ Training: minimize a loss function

$$L_R = L(x, g(f(x))) + \lambda |h|_1$$

Regularization

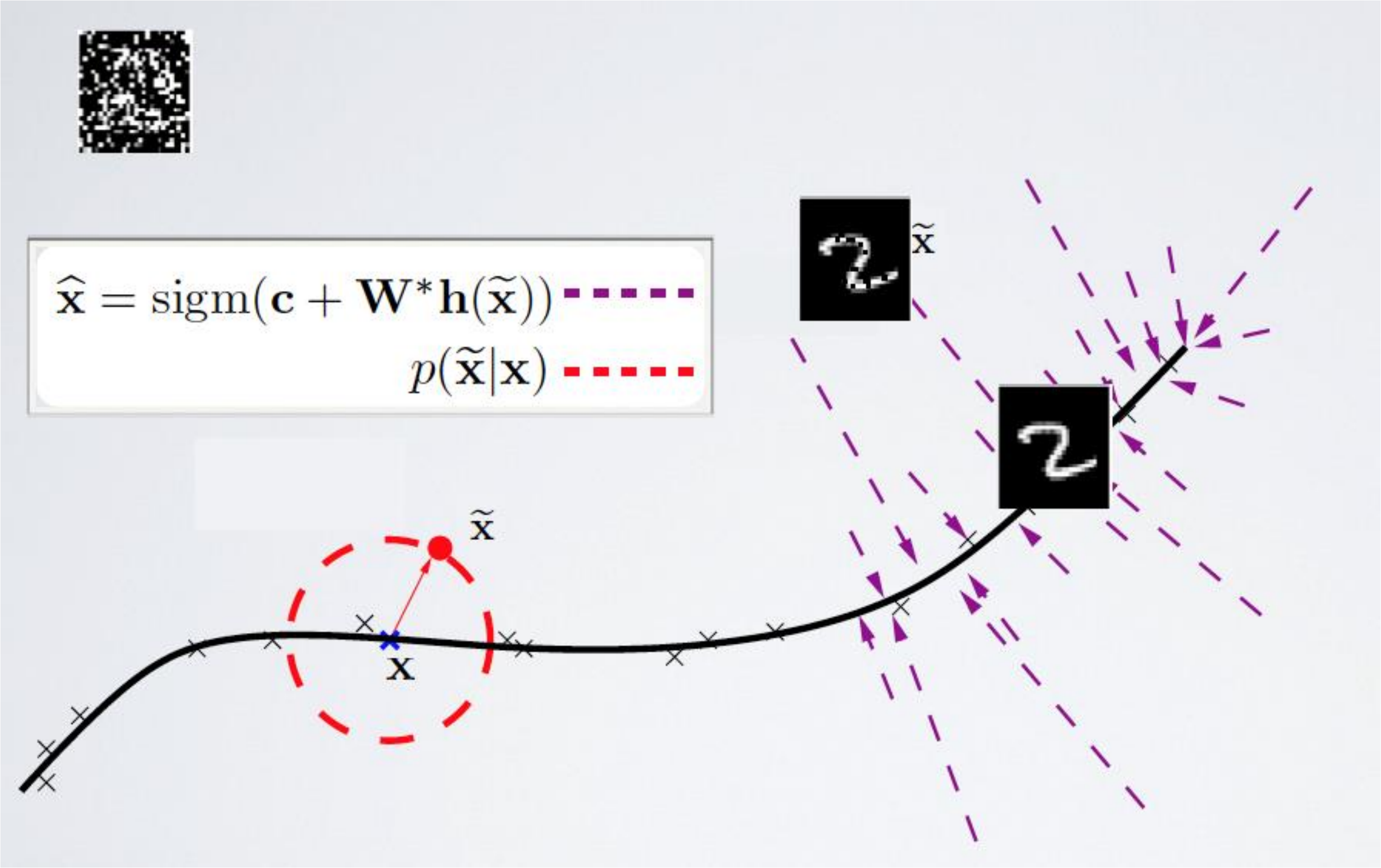
- Denoising autoencoder



- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability ν
  - Gaussian additive noise
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$
- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the **noiseless input** $\mathbf{x}$

Regularization

- Denoising autoencoder



$\widehat{\mathbf{x}} = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^{*}\mathbf{h}(\widetilde{\mathbf{x}}))$
$p(\widetilde{\mathbf{x}}|\mathbf{x})$

Outline

- Representation learning
  - AutoEncoder
- Variational Autoencoders (VAEs)
  - VAE objective
  - Reparametrization trick
  - Connection to Auto-Encoders

Acknowledgement: Feifei Li et al’s cs231n notes

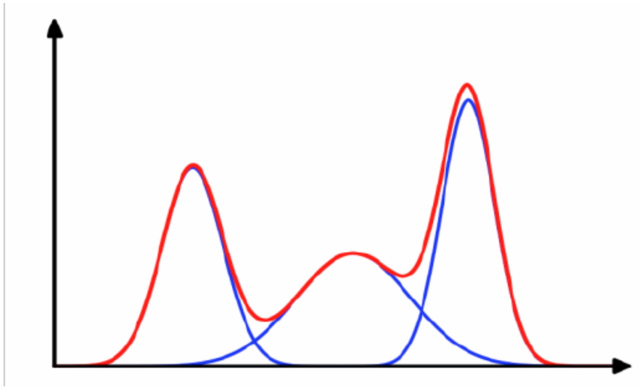
Latent variable model

- Data generation process
  - □ Latent variable $\boldsymbol{z}$ $\quad p(\boldsymbol{z}) = \text{something simple}$
  - □ A mapping from the latent space to observation $\boldsymbol{x}$

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

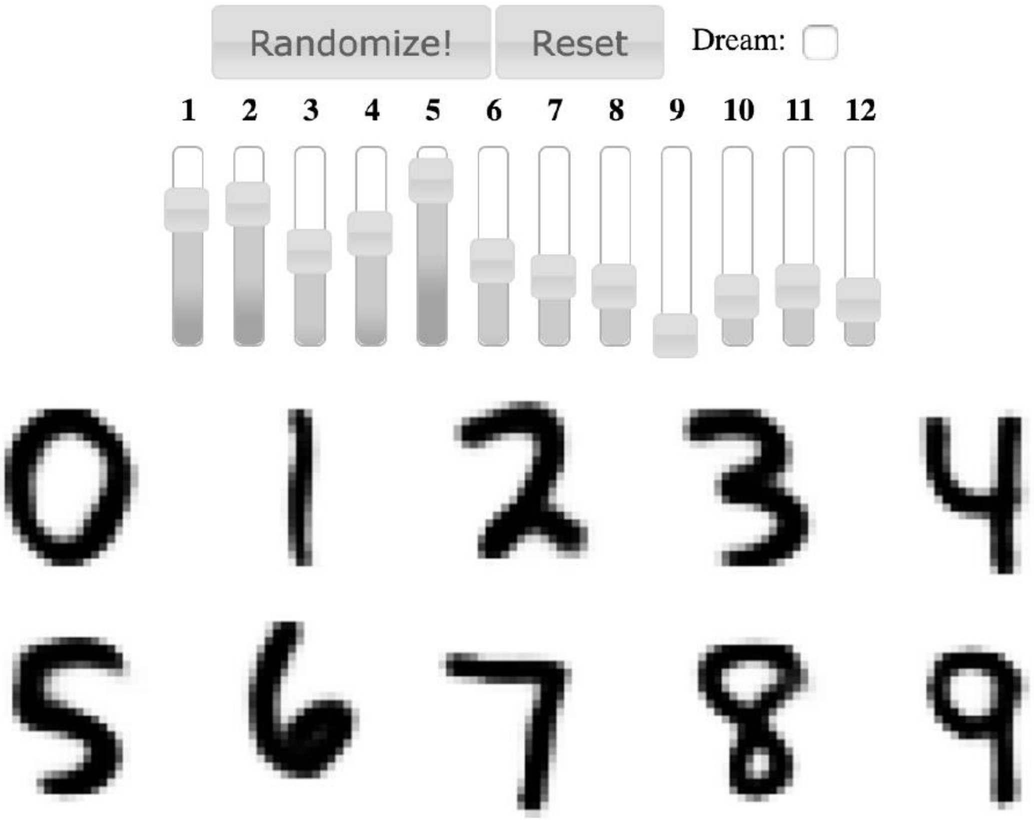
For example, a Gaussian mixture model

$$p_\theta(x) = \sum_{k=1}^{K} p_\theta(z = k) p_\theta(x|z = k)$$



An example

- Generating hand-written digits



(Latent space) **z**

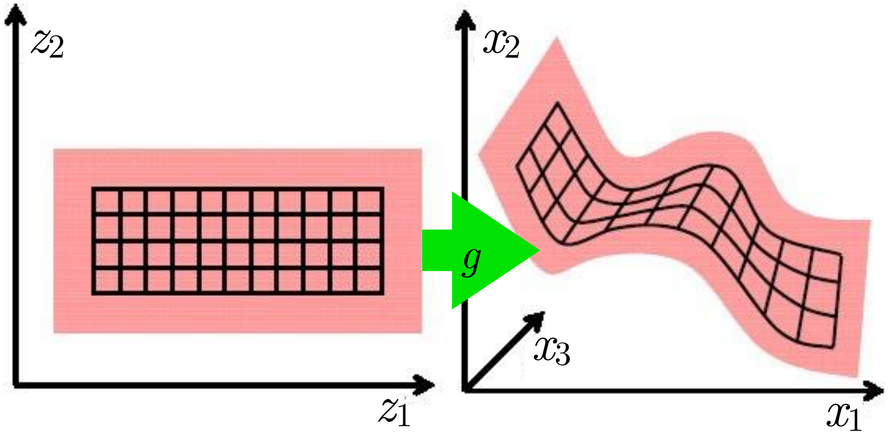
(Observed space) **x**

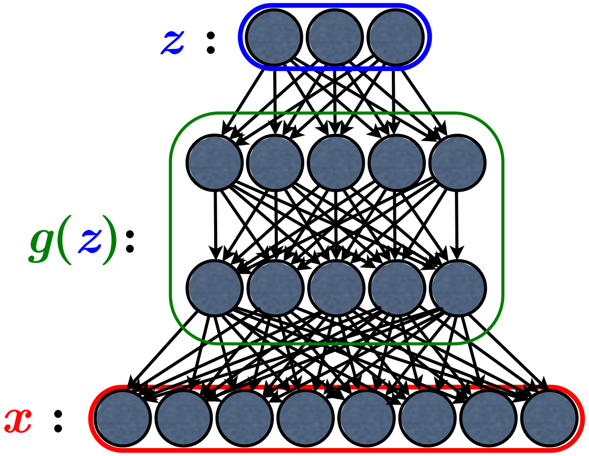
- □ http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Deep Latent Variable Models

- Leverage neural networks in a latent variable model

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z}) \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = g(\boldsymbol{z})$$





- Can represent complicated data distribution and conditional dependencies

An example

- $p(z) = N(0, I)$

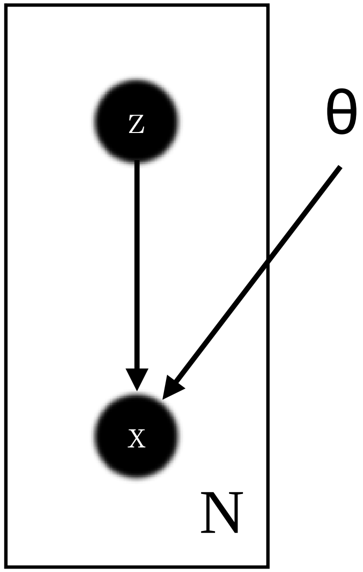
- $P_\theta(x|z) = N(\mu, \sigma^2)$

  $\mu = f_\theta(z) = \text{multilayer neural net}$

- With flexible neural net,

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

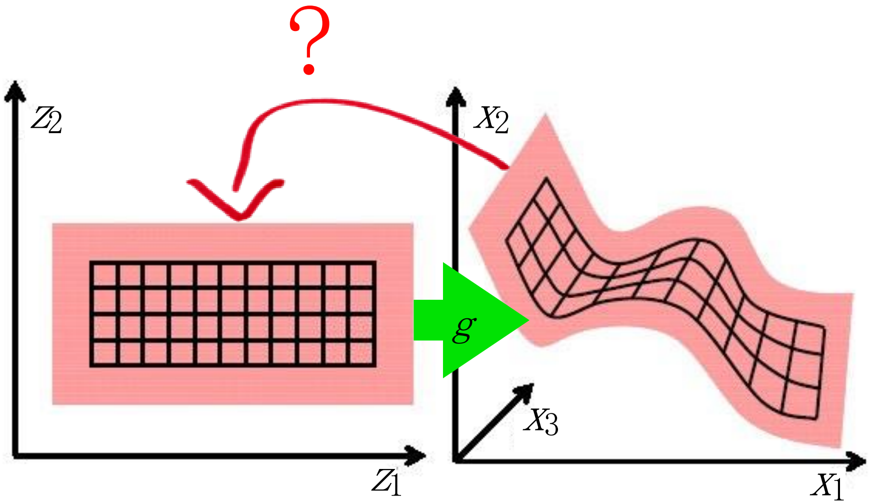
  - □ Can be arbitrarily complicated
  - □ The multilayer network can capture complex dependencies in the data generation process

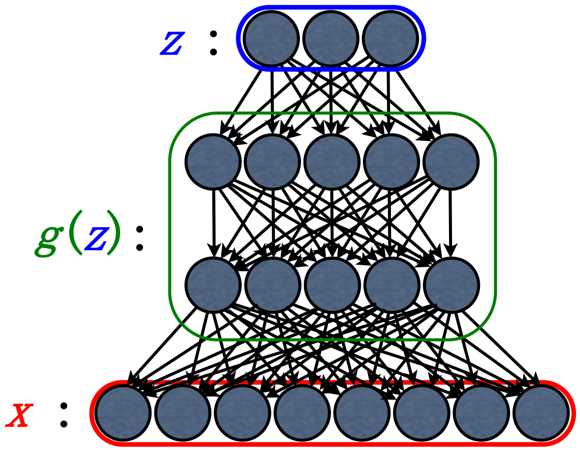


Challenges in Deep LVM

- Inference:
  - □ Given an observation x , what is the probable z ?
  - □ Computing the posterior $p(\mathbf{z}|\mathbf{x})$ (intractable)

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





The Variational Autencoder: overview

- Inference:
  - Introduce a parametric model $q_\phi(z \mid x)$ to approximate the true posterior $p_\theta(z \mid x)$
  - Variational inference or Amortized inference

$$\forall x_i \quad \arg\min_{q_i} D_{KL}(q_i(z) || p_\theta(z|x_i))$$

$$\Rightarrow \forall x_i \quad \arg\min_{\phi} D_{KL}(q_\phi(z|x_i) || p_\theta(z|x_i))$$

  - Replacing datum-wise posterior distribution by a parametric family of conditional densities.

The Variational Autencoder: overview

- Inference:
  - □ Introduce a parametric model $q_\phi(z \mid x)$ to approximate the true posterior $p_\theta(z \mid x)$
  - □ Variational inference or Amortized inference

- Learning:
  - □ 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

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

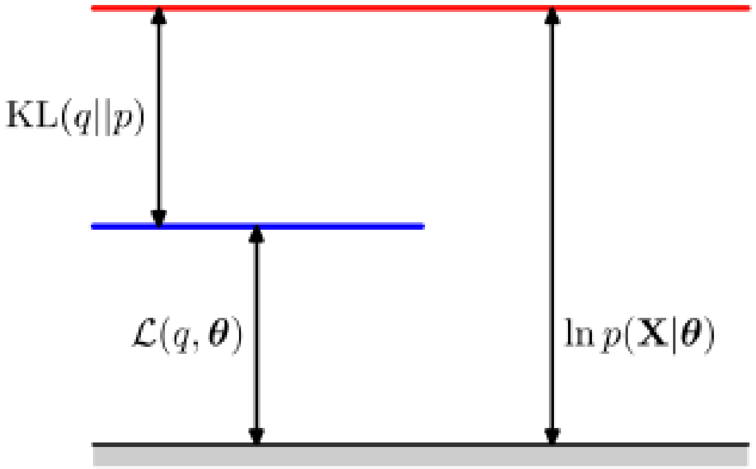
$$\log p_\theta(x) = \boxed{\mathcal{L}(x;\theta,\phi)} + D_{KL}(q_\phi(z|x)||p_\theta(z|x))$$

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

VAE objective

- Visualizing ELBO

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



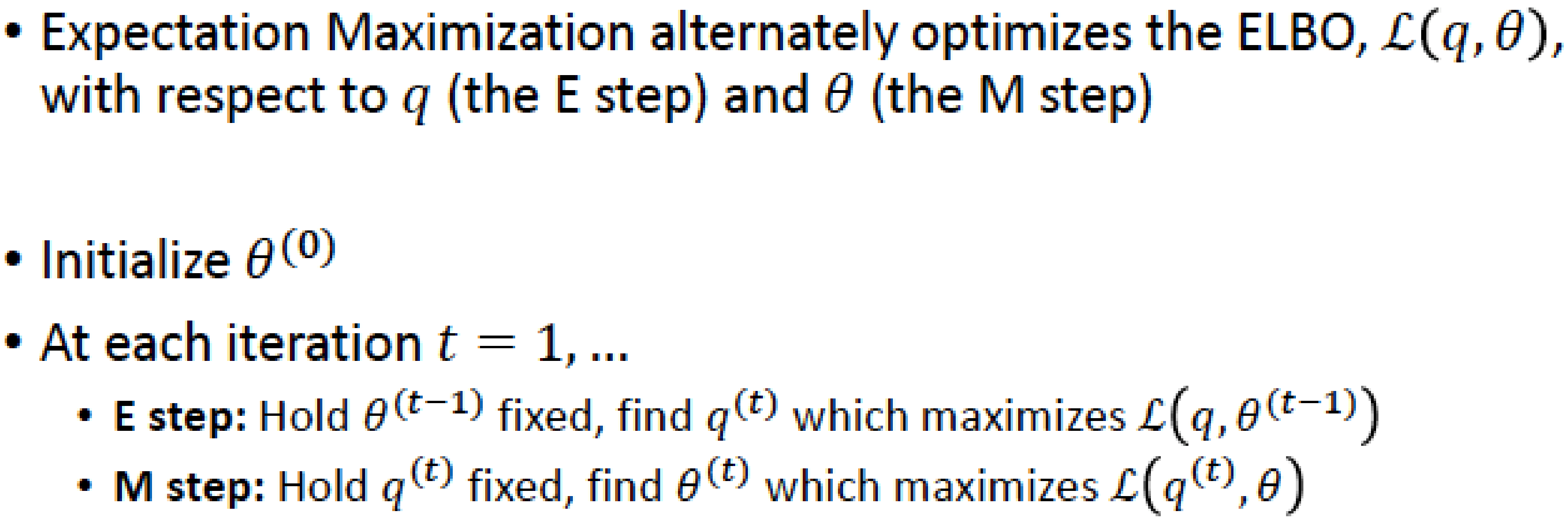
$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

Bishop – Pattern Recognition and Machine Learning

- Note: all we have done so far is decompose the log probability of the data, we still have exact equality
- This holds for any distribution q

VAE learning

- EM perspective

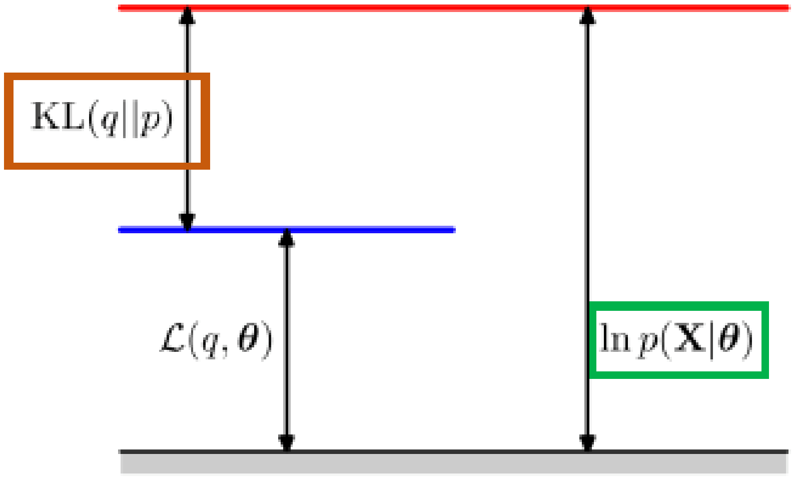


- Expectation Maximization alternately optimizes the ELBO, $\mathcal{L}(q, \theta)$, with respect to q (the E step) and θ (the M step)

- 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 $\mathcal{L}\left(q^{(t)}, \theta\right)$

EM perspective

- The E step


$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

Bishop – Pattern Recognition and Machine Learning

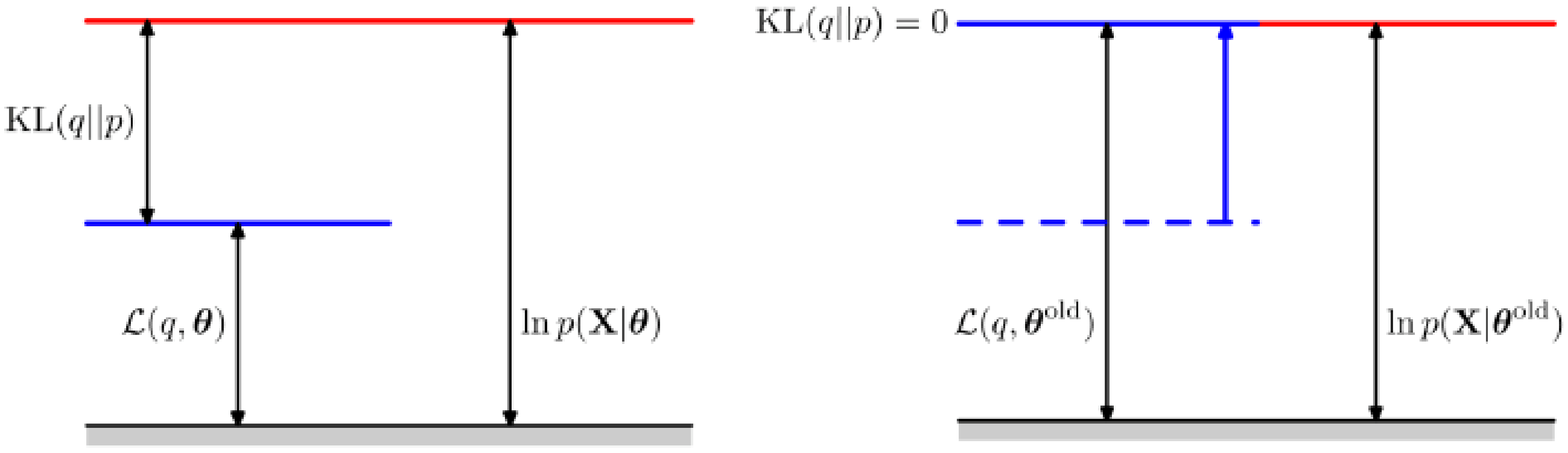
- The first term does not involve q, and we know the KL divergence must be non-negative
- The best we can do is to make the KL divergence 0
- Thus the solution is to set $\boldsymbol{q^{(t)}(z) \leftarrow p(z|x,\theta^{(t-1)})}$

- Suppose we are at iteration t of our algorithm. How do we maximize $\mathcal{L}(q,\theta^{(t-1)})$ with respect to q? We know that:

$$\mathrm{argmax}_q\, \mathcal{L}(q,\theta^{(t-1)}) = \mathrm{argmax}_q \log p(x|\theta^{(t-1)}) - \mathrm{KL}\left(q(z)\,||\,p(z|x,\theta^{(t-1)})\right)$$

EM perspective

- The E step

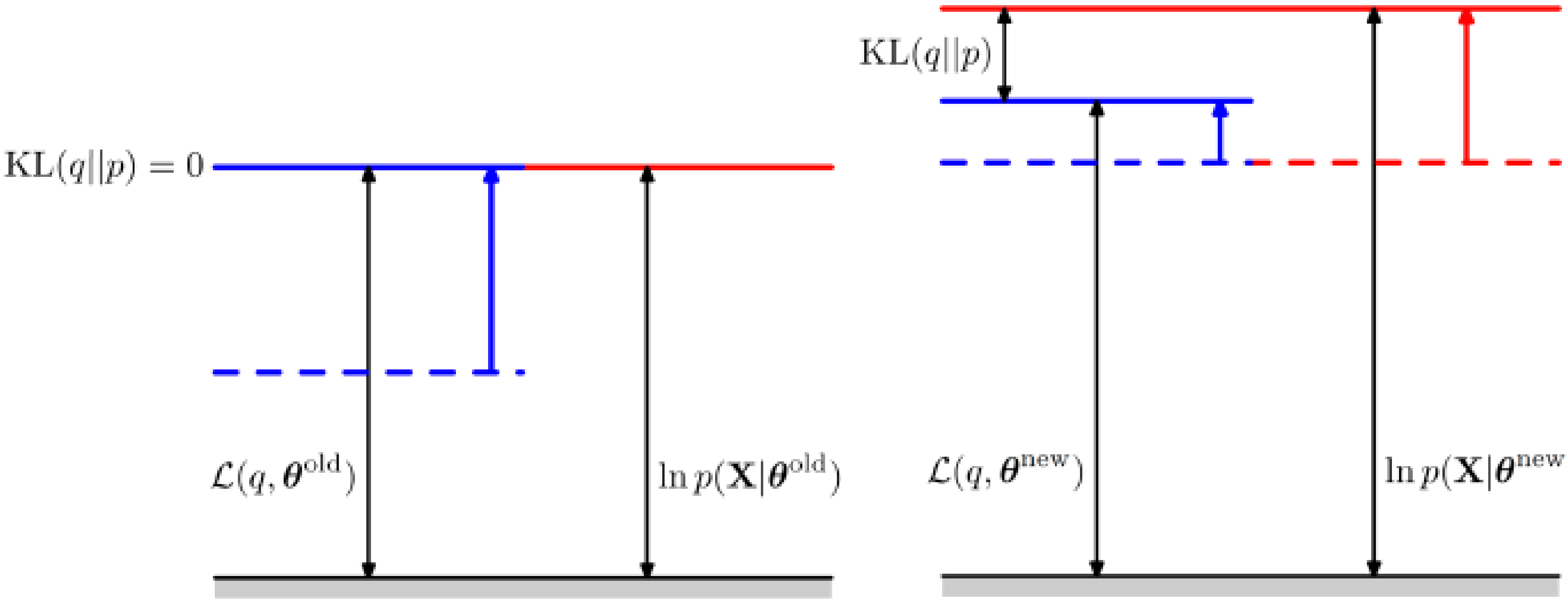

$\mathrm{KL}(q||p)$
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$
$\mathrm{KL}(q||p) = 0$
$\mathcal{L}(q,\boldsymbol{\theta}^{\mathrm{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{old}})$

Bishop – Pattern Recognition and Machine Learning

- Suppose we are at iteration t of our algorithm. How do we maximize $\mathcal{L}\left(q,\theta^{(t-1)}\right)$ with respect to q? $\boldsymbol{q}^{(\boldsymbol{t})}(\boldsymbol{z}) \leftarrow \boldsymbol{p}\left(\boldsymbol{z}|x,\boldsymbol{\theta}^{(\boldsymbol{t}-\mathbf{1})}\right)$

EM perspective

- The M step



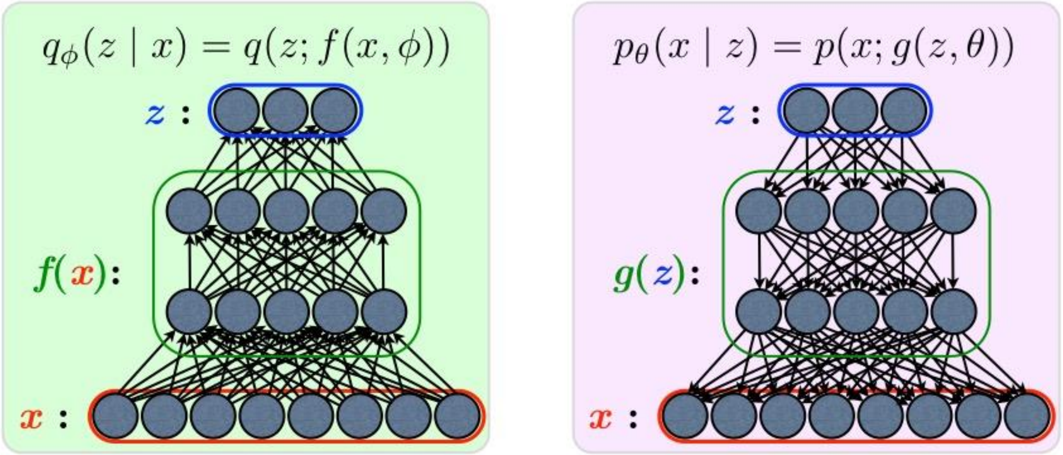
$\mathrm{KL}(q||p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{old}})$
$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{new}})$

Bishop – Pattern Recognition and Machine Learning

- After applying the E step, we increase the likelihood of the data by finding better parameters according to: $\theta^{(t)} \leftarrow \mathbf{argmax}_{\boldsymbol{\theta}}\, \mathbb{E}_{\boldsymbol{q}^{(t)}(\boldsymbol{z})}[\mathbf{log}\, \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{z}\, |\boldsymbol{\theta})]$

VAE learning

- What is $q_\phi(z|x)$?
  - □ Parametrize $q_\phi(z|x)$ with another neural network



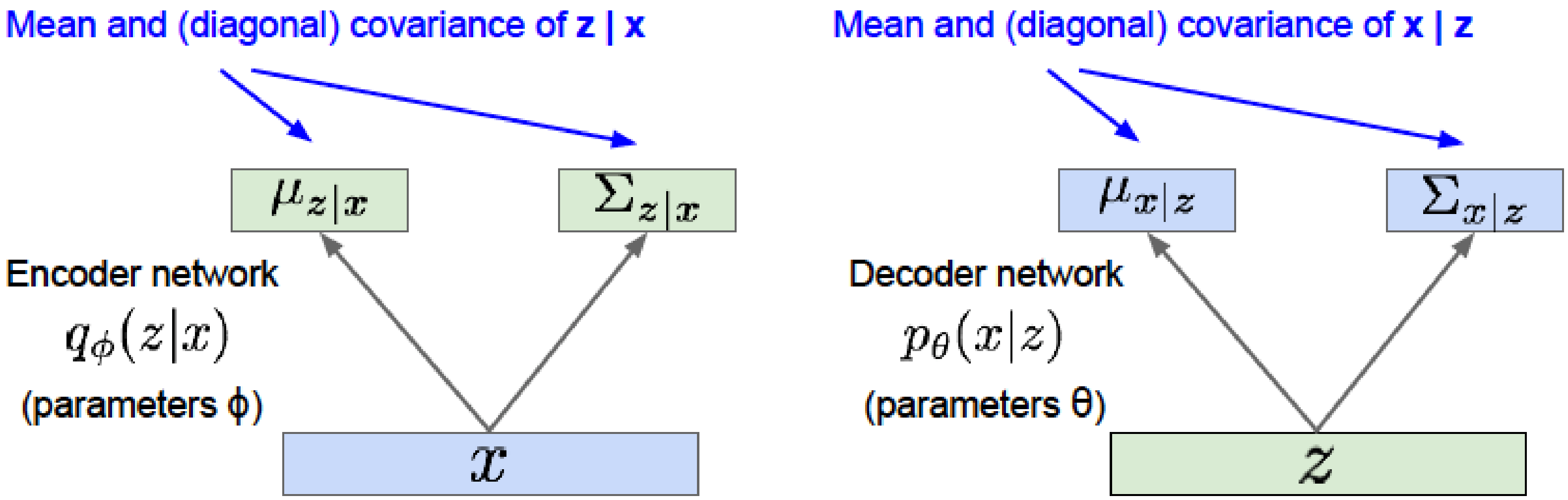
$q_\phi(z \mid x) = q(z; f(x, \phi))$
z :
$f(x)$:
x :
$p_\theta(x \mid z) = p(x; g(z, \theta))$
z :
$g(z)$:
x :

- Interpreting VAE objective

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

VAE Example

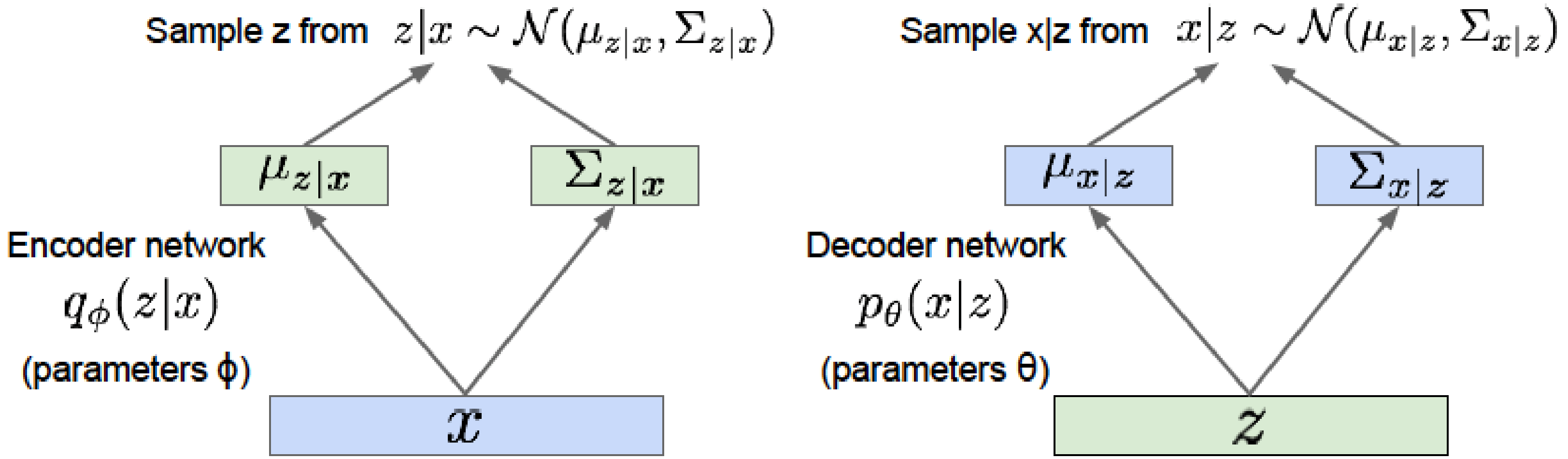
- Conditionals Gaussians



VAE Example

- Conditionals Gaussians

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

$\mu_{z|x}$ $\Sigma_{z|x}$

Encoder network

$q_\phi(z|x)$

(parameters ϕ)

x

Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

$\mu_{x|z}$ $\Sigma_{x|z}$

Decoder network

$p_\theta(x|z)$

(parameters θ)

z

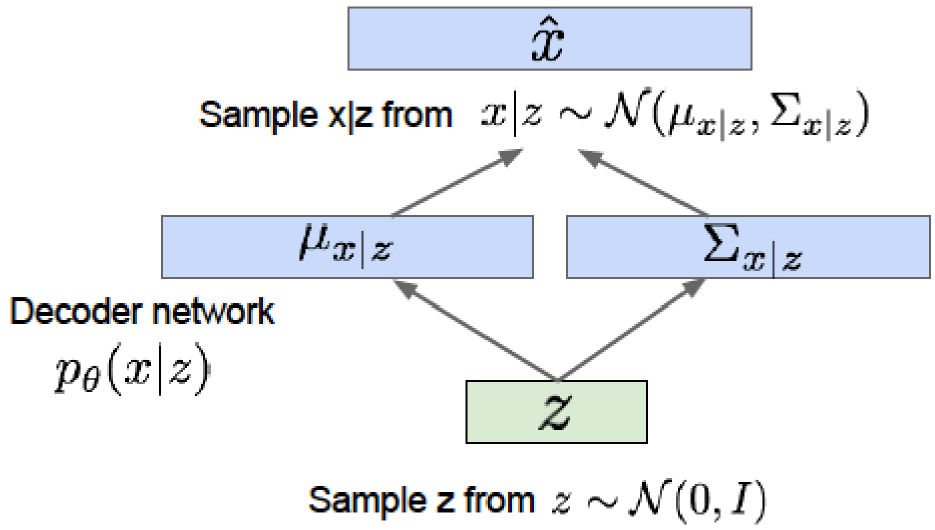
Encoder and decoder networks also called
"recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE Example

- Generating data

Use decoder network. Now sample z from prior!



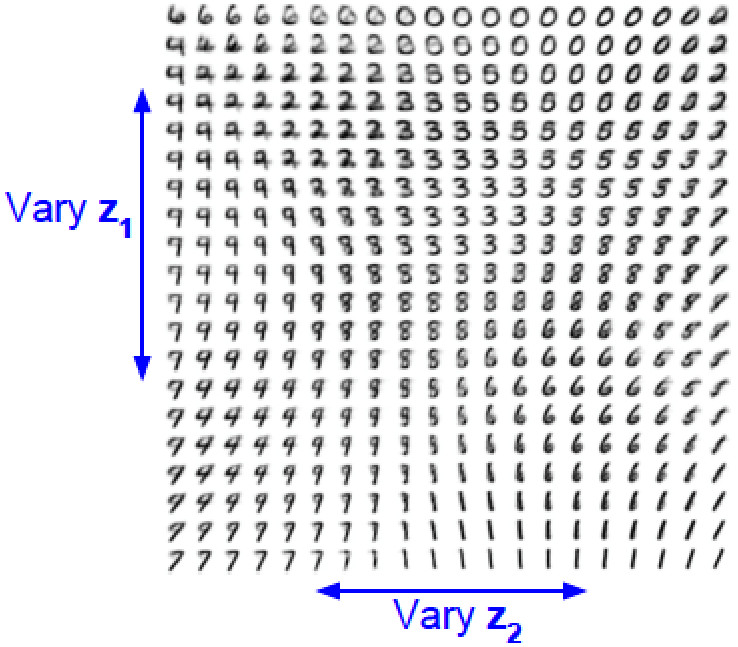
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

Decoder network $p_\theta(x|z)$

Sample z from $z \sim \mathcal{N}(0, I)$

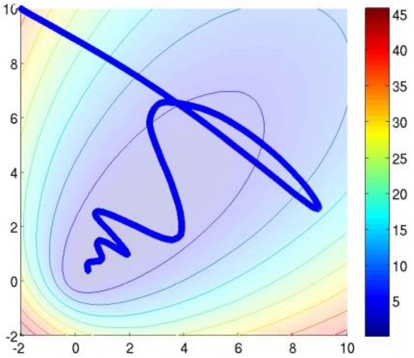
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Data manifold for 2-d **z**

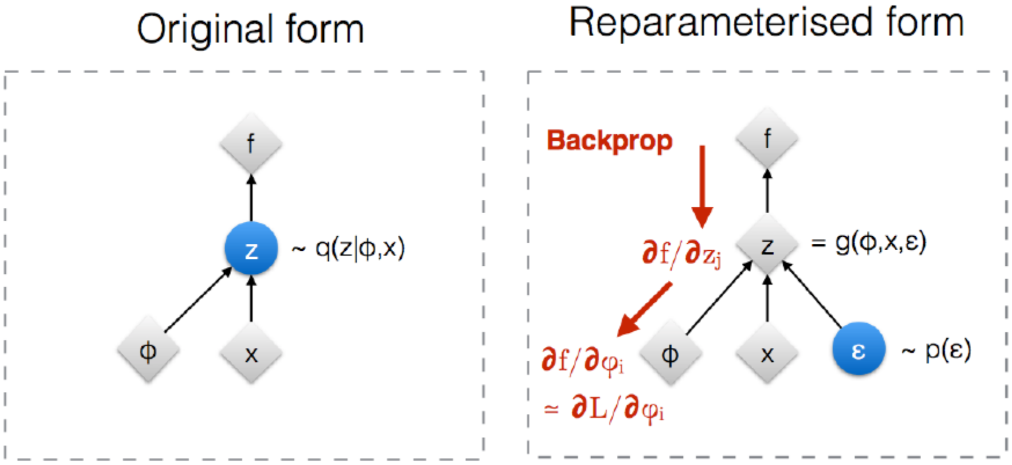


Two views of Learning VAE

- 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

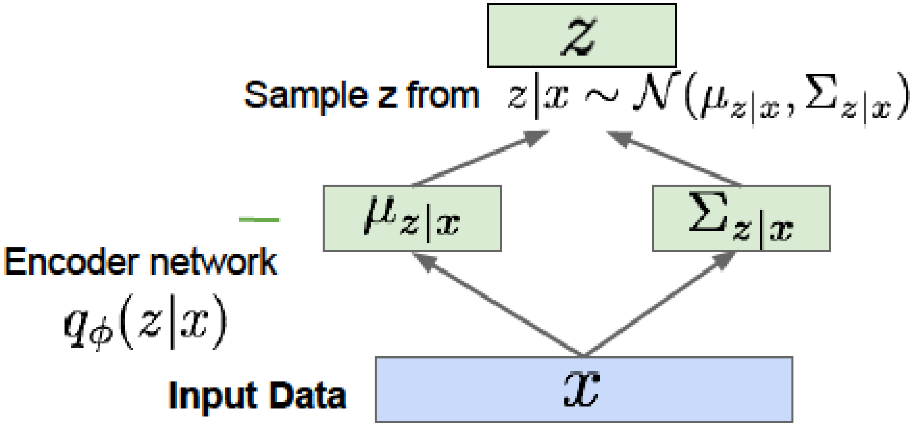
$$\nabla_\phi \mathcal{L}(x, \phi, \theta) = E_{q(\epsilon)}[\nabla_\phi f_{\phi,\theta}(x, z)] \approx \frac{1}{L} \sum_{i=1}^{L} \nabla_\phi f_{\phi,\theta}(x, g_\phi(\epsilon^{(i)}, x)), \quad \boxed{\epsilon^{(i)} \sim q(\epsilon)}$$

VAE Example

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

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

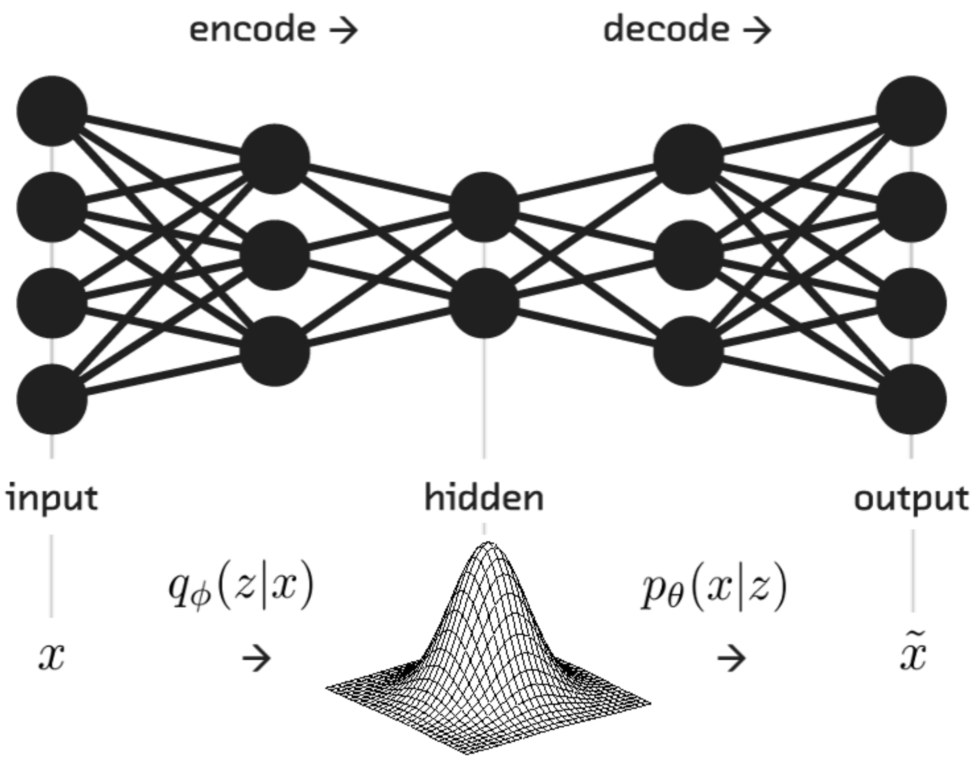
$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}\left[f(z)\right] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}\left[f(\mu+\sigma\epsilon)\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mu + \sigma\epsilon^{(l)})$$



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



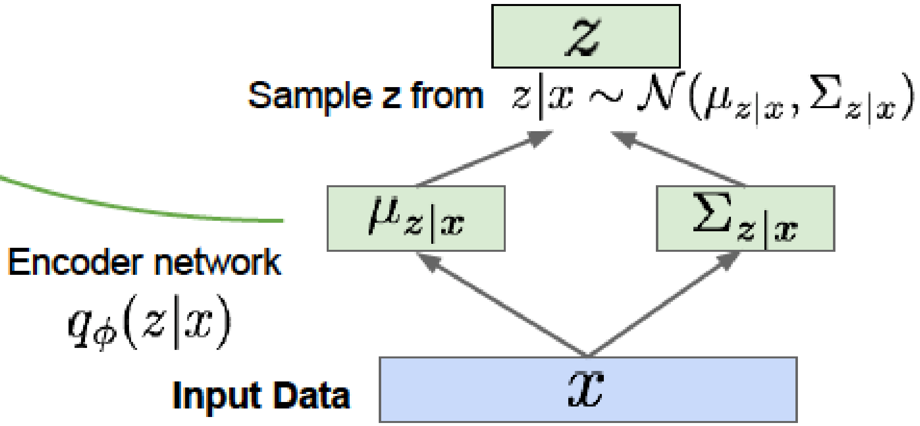
VAE Example

- Learning objective

Putting it all together: maximizing the likelihood lower bound

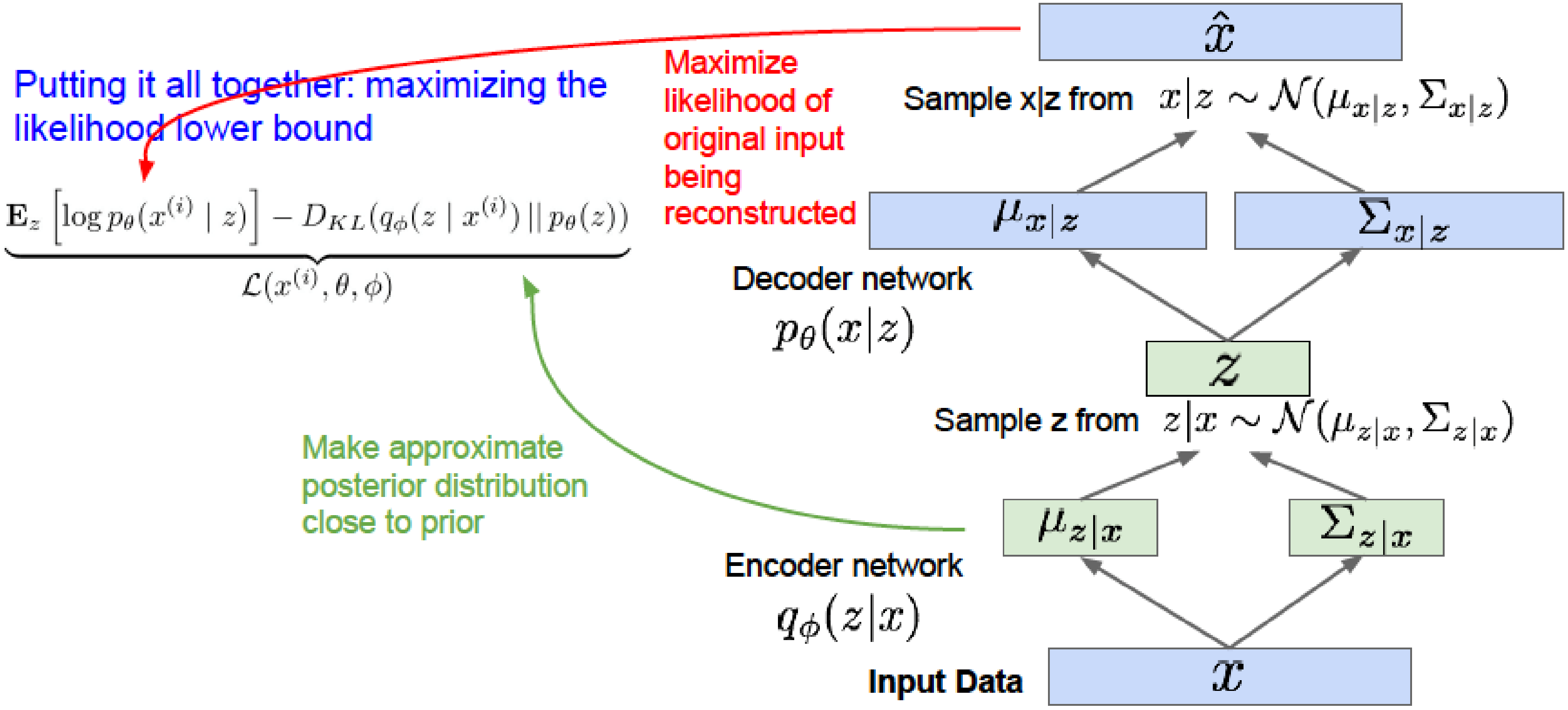
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Make approximate posterior distribution close to prior



VAE Example

- Learning objective



Putting it all together: maximizing the likelihood lower bound

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

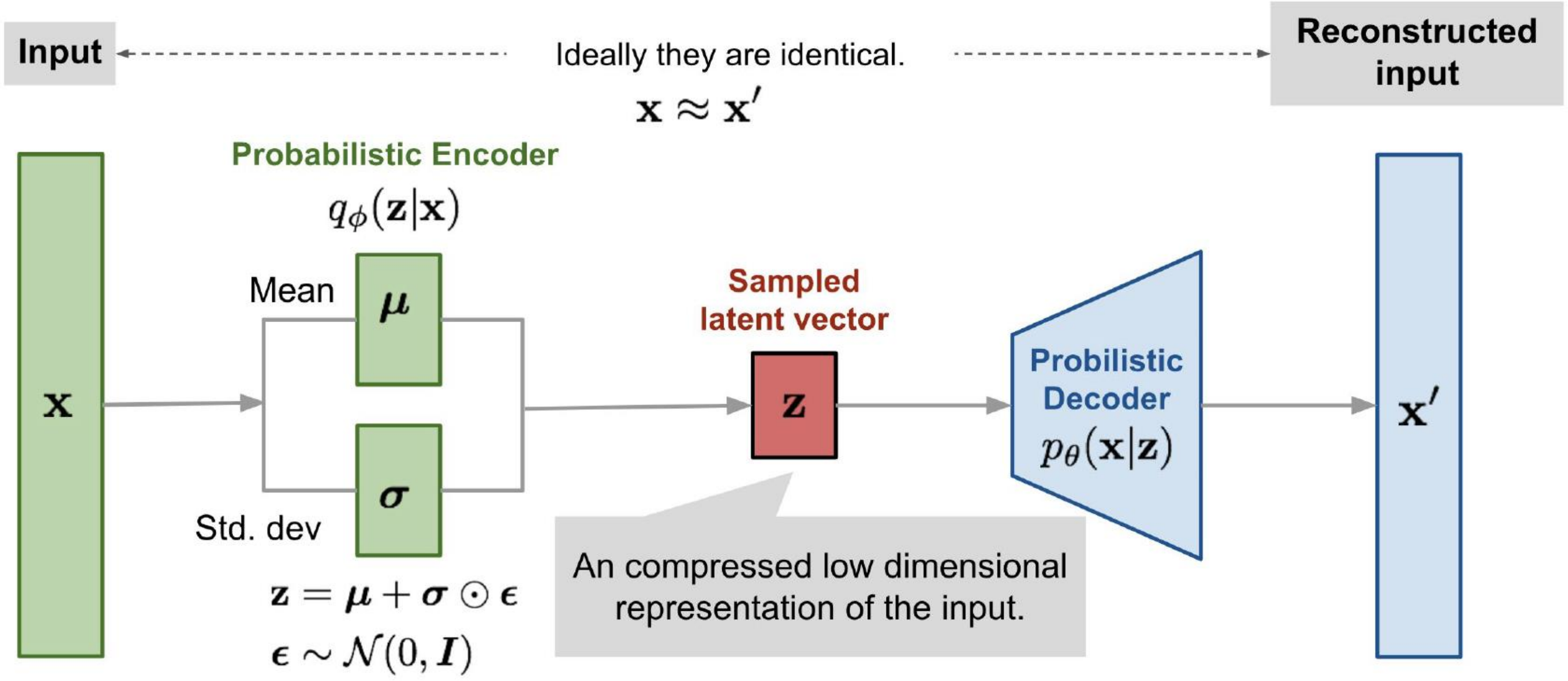
Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

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



Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\mathbf{x}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

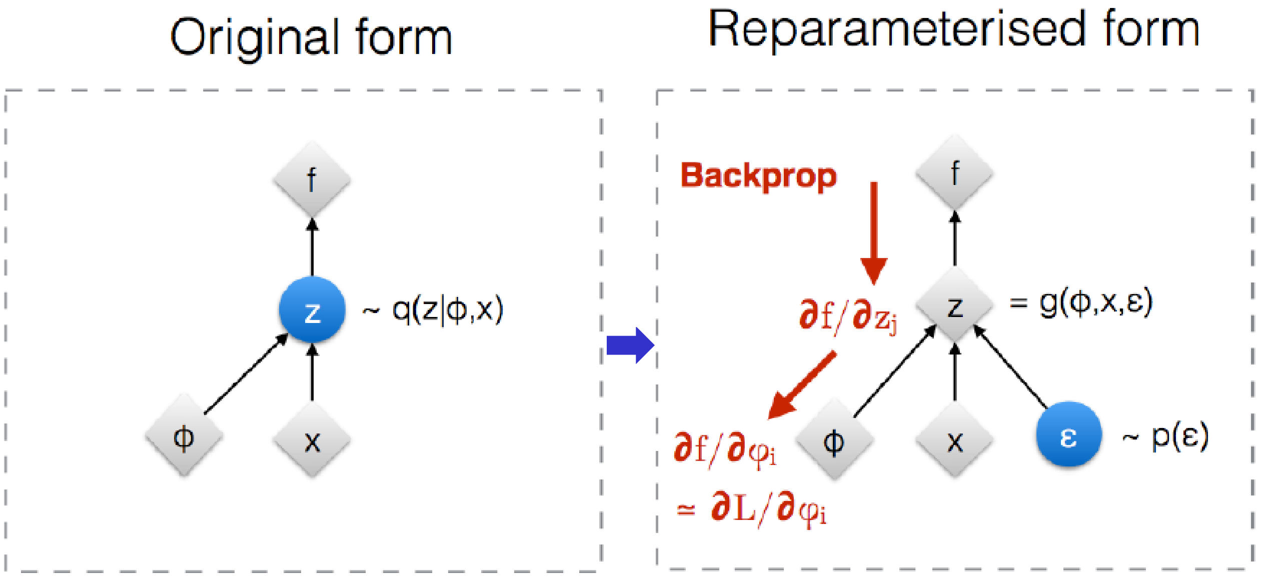
- □ The objective function can be represented as an Autoencoder-like computation graph.

Network interpretation

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

⬇

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



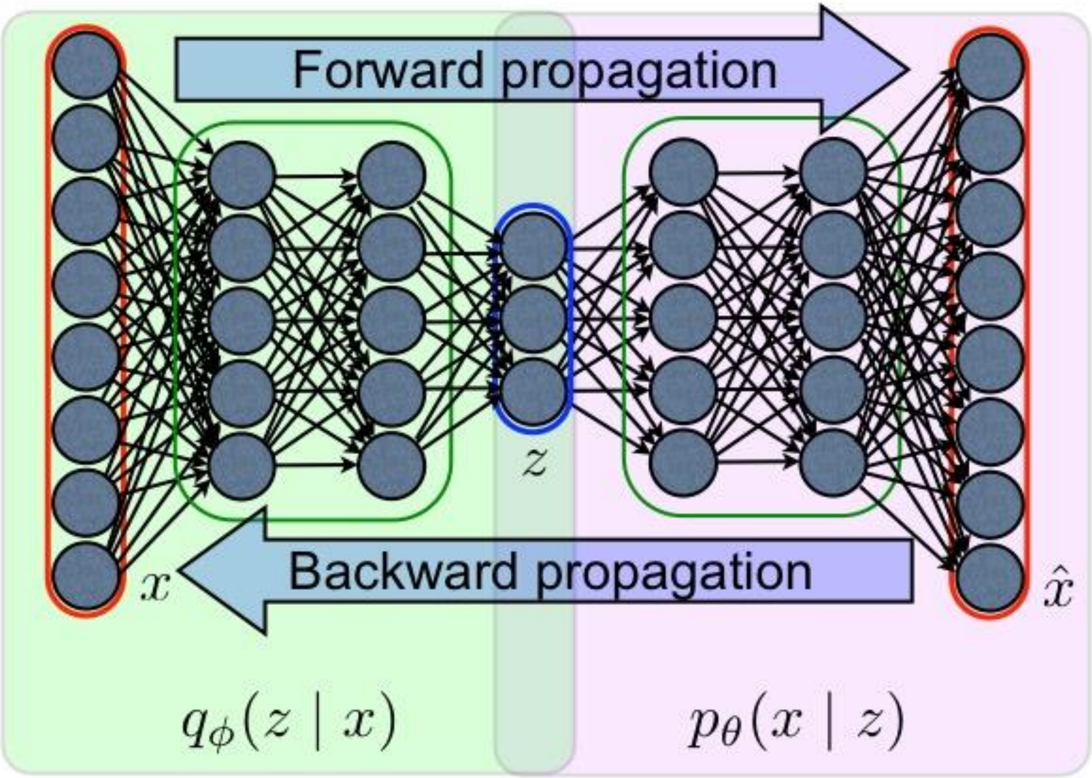
◆ : Deterministic node

● : Random node

[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.



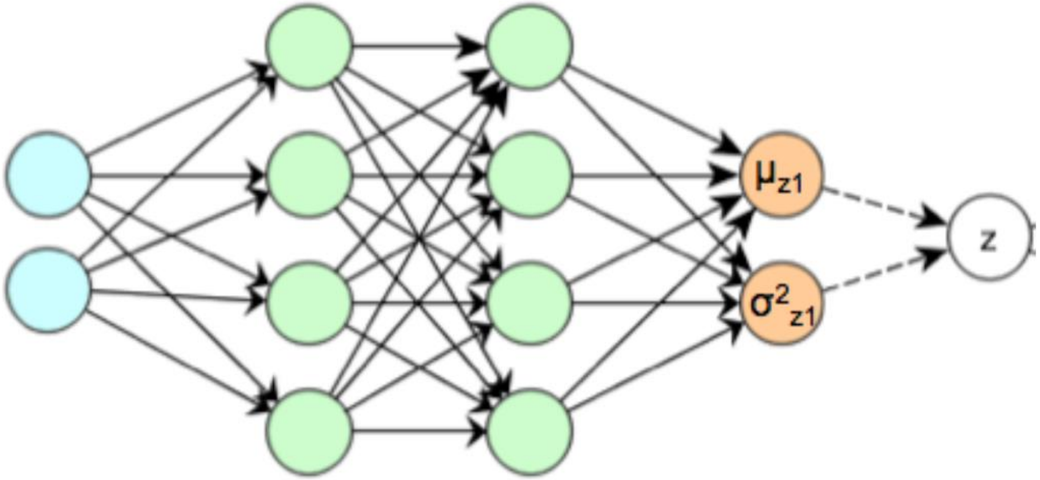
1D Gaussian Case

- 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

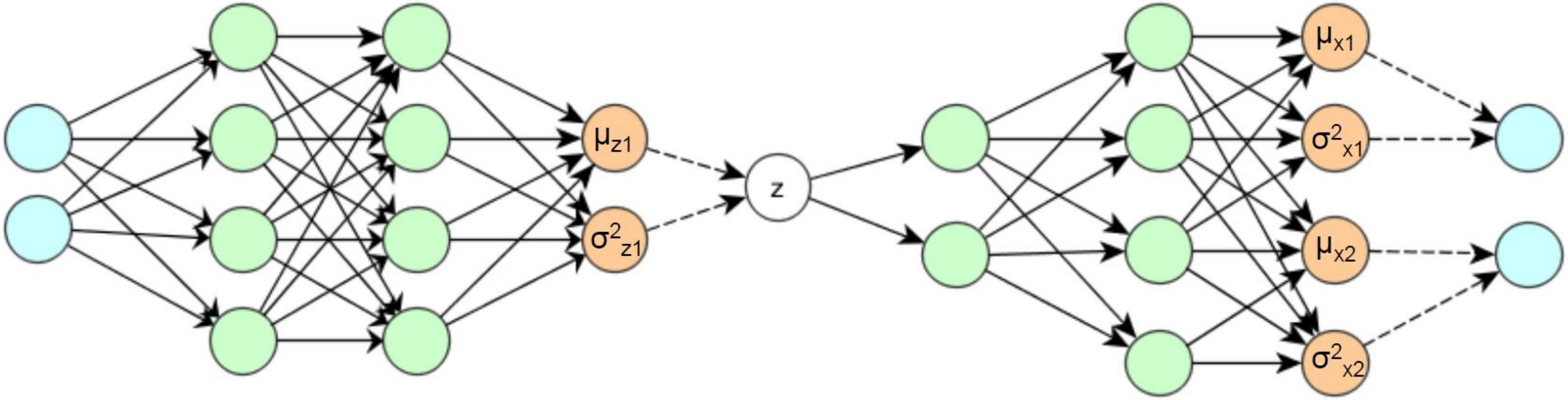
$$-D_{\mathrm{KL}}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$



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

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

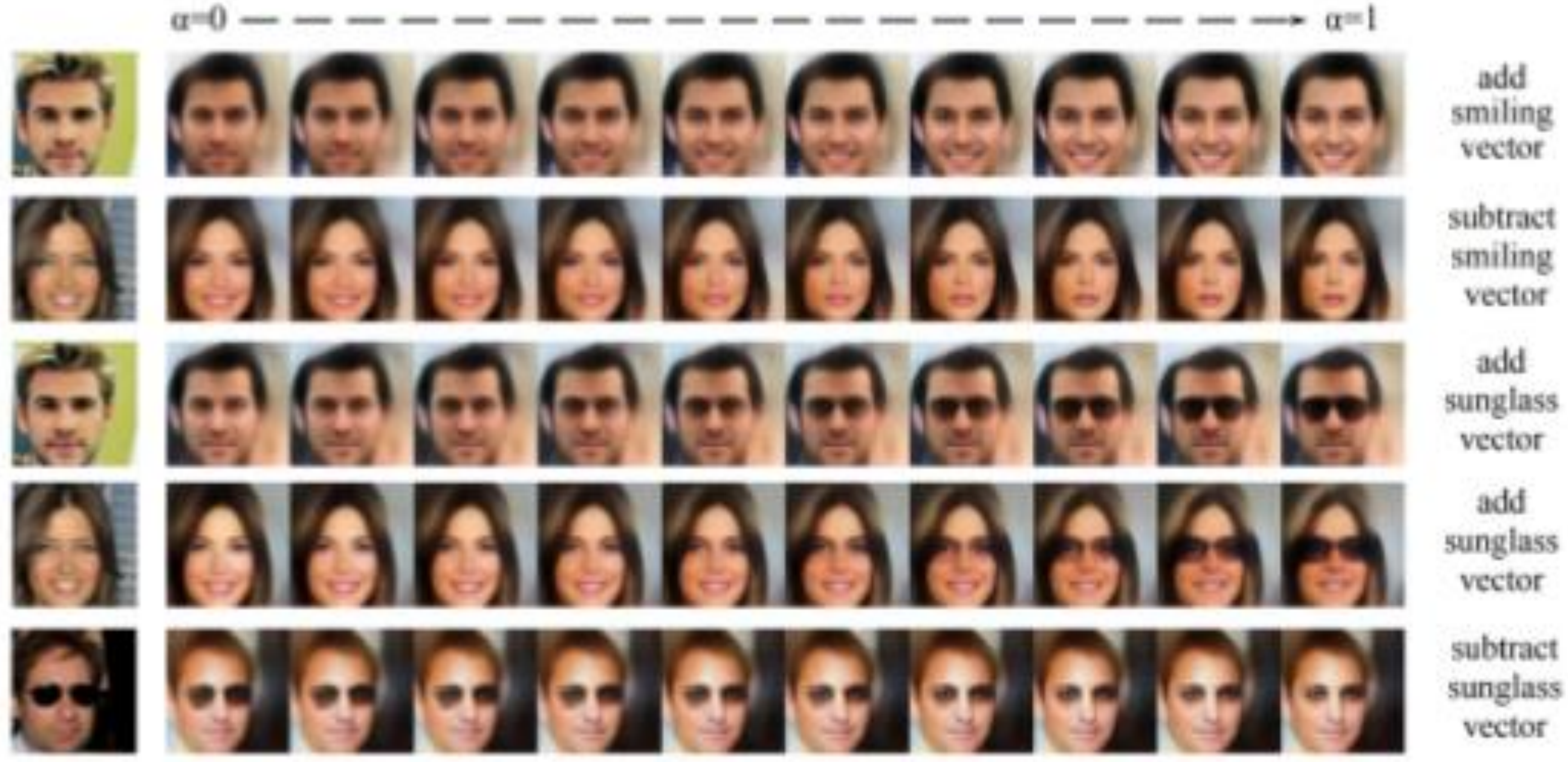
We use mini batch gradient decent to optimize the cost function over all $x^{(i)}$ in the mini batch

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

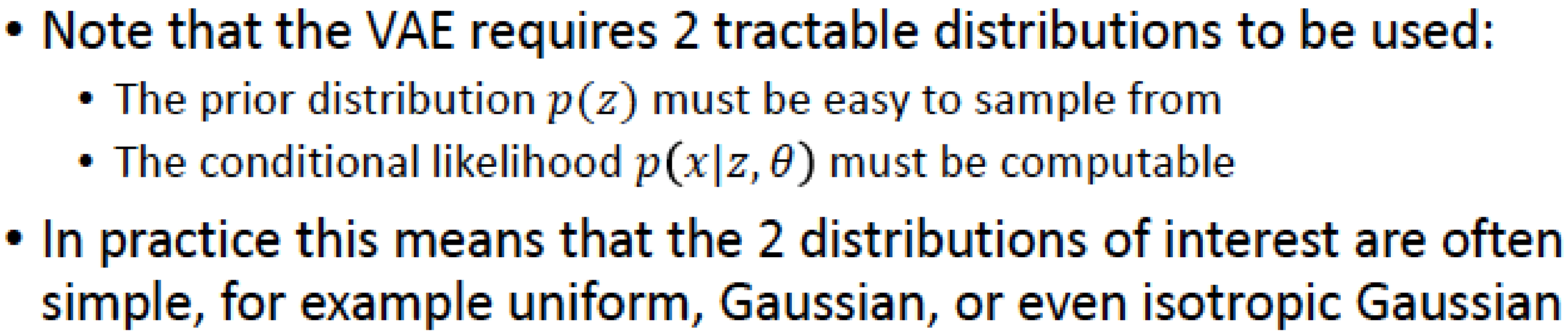
Interpreting the latent space



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

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

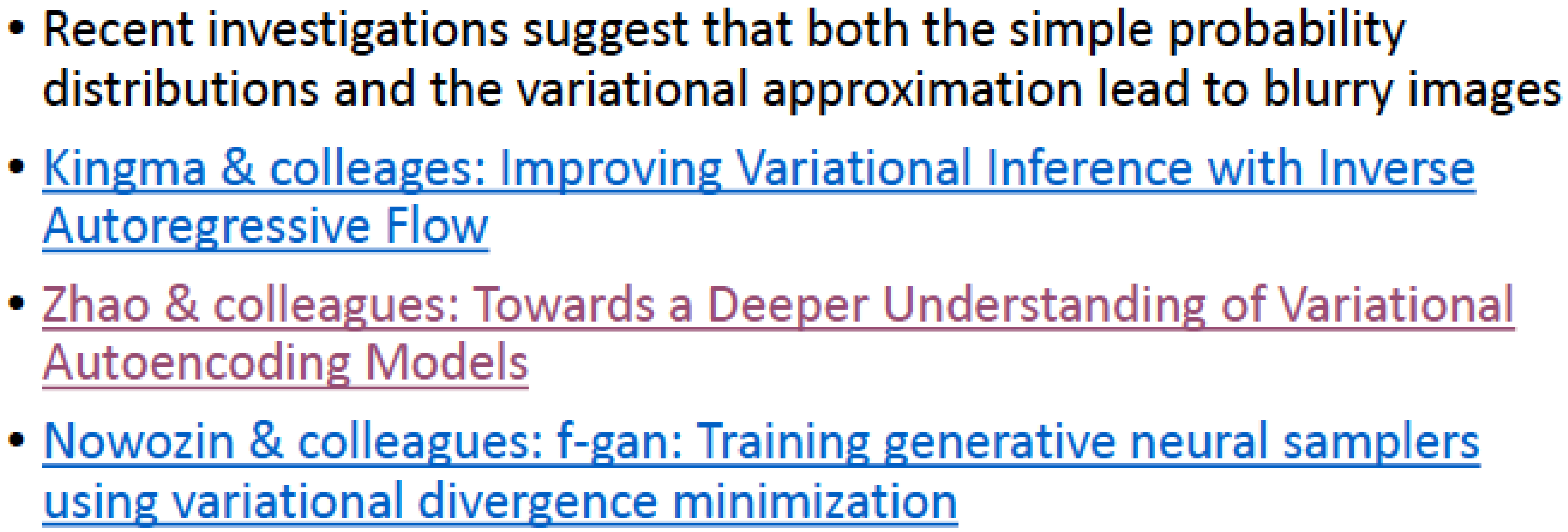
- Blurry images

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

Summary

- Autoencoders (AEs)
  - Representation learning
- Variational Autoencoders (VAEs)
  - VAE objective
  - Reparametrization trick
- Next time:
  - VAE: Vision applications
  - GAN
- Reading material
  - http://www.cs.columbia.edu/~blei/talks/Blei_VI_tutorial.pdf
  - http://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/slides/lec17.pdf
  - https://dvl.in.tum.de/slides/adl4cv-ws18/6.Bayesian&VAE.pdf
  - https://arxiv.org/pdf/1312.6114.pdf

Acknowledgement: Feifei Li et al's cs231n notes