CS 4803 / 7643: Deep Learning

Topics:

- Variational Auto-Encoders (VAEs)
  - Reparameterization trick
- Generative Adversarial Networks (GANs)

Dhruv Batra

Georgia Tech

Administrativia

- Project submission instructions released
  - Due: 12/04, 11:55pm
  - Last deliverable in the class
  - Can’t use late days
  - https://piazza.com/class/jkujs03pgu75cd?cid=225

Recap from last time

Variational Autoencoders (VAE)

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, ..., x_{i-1})$$

$$p(\vec{x})$$

$$p(\vec{x}, z) = p(\vec{x}|z)\, p(z)$$

VAEs define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

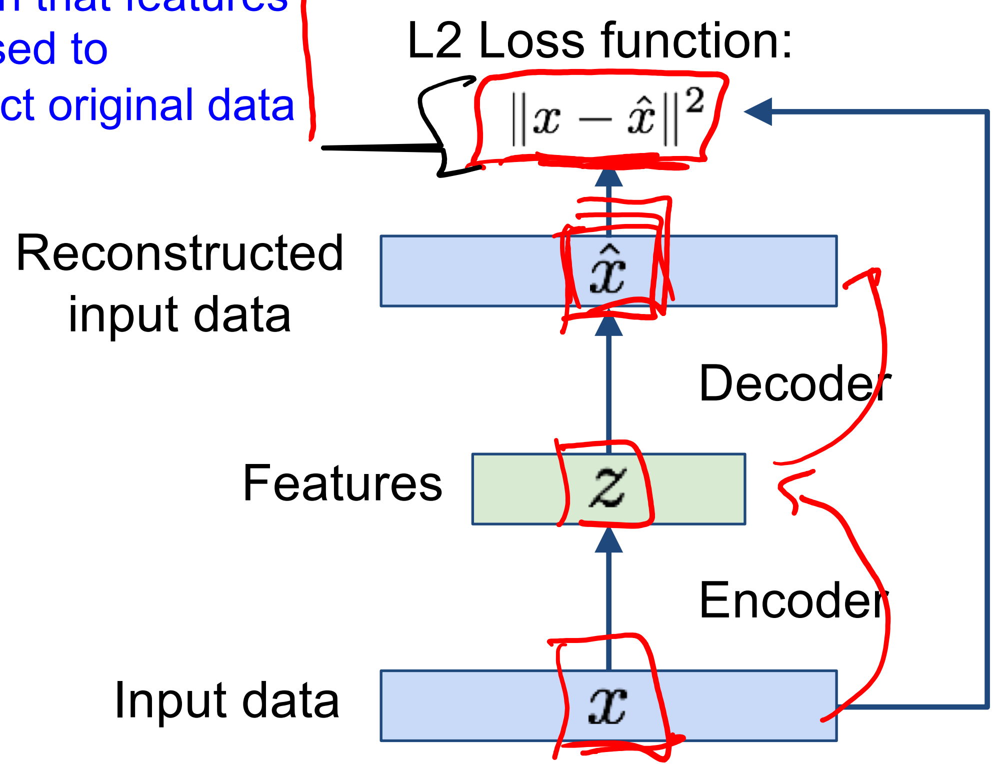
Variational Auto Encoders

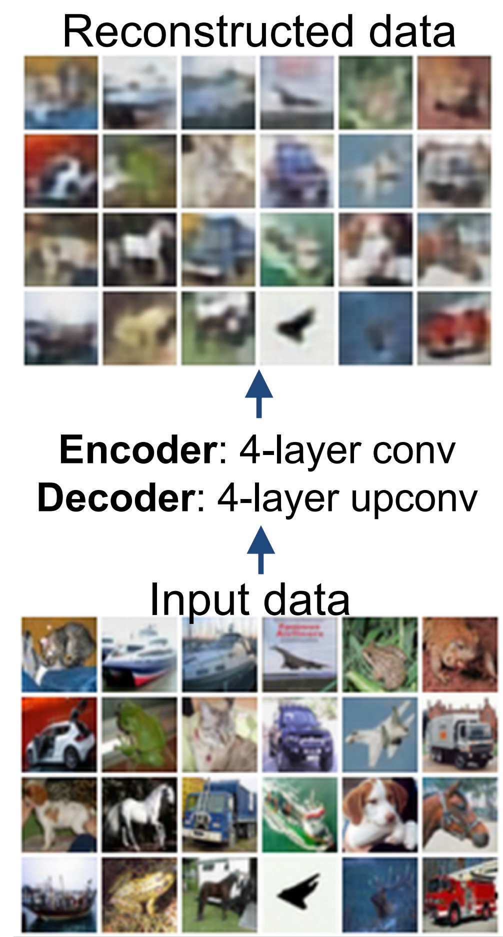
VAEs are a combination of the following ideas:

1. Auto Encoders
2. Variational Approximation
   - Variational Lower Bound / ELBO
3. Amortized Inference Neural Networks
4. "Reparameterization" Trick

Autoencoders

Train such that features can be used to reconstruct original data





Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

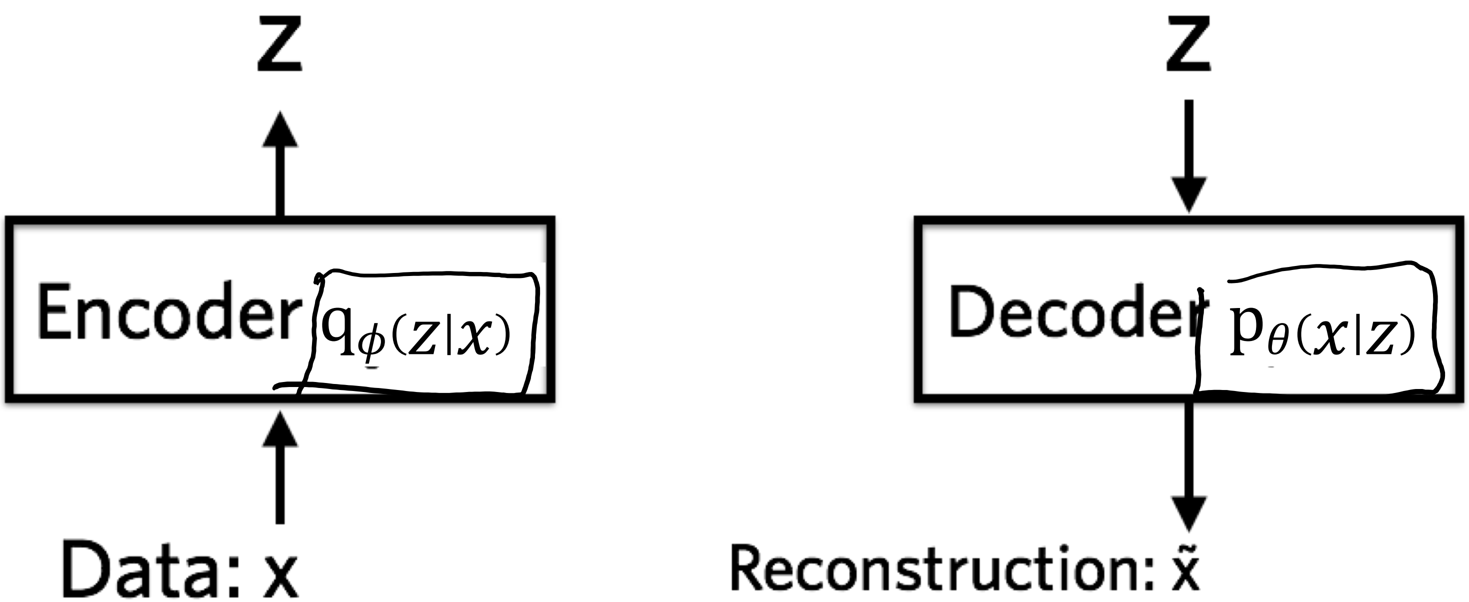


Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   - Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick

Key problem

- $\boxed{P(z|x)} = \dfrac{P(z,x)}{P(x)} = \dfrac{P(x|z)\,P(z)}{\sum_z P(x|z)\,P(z)}$

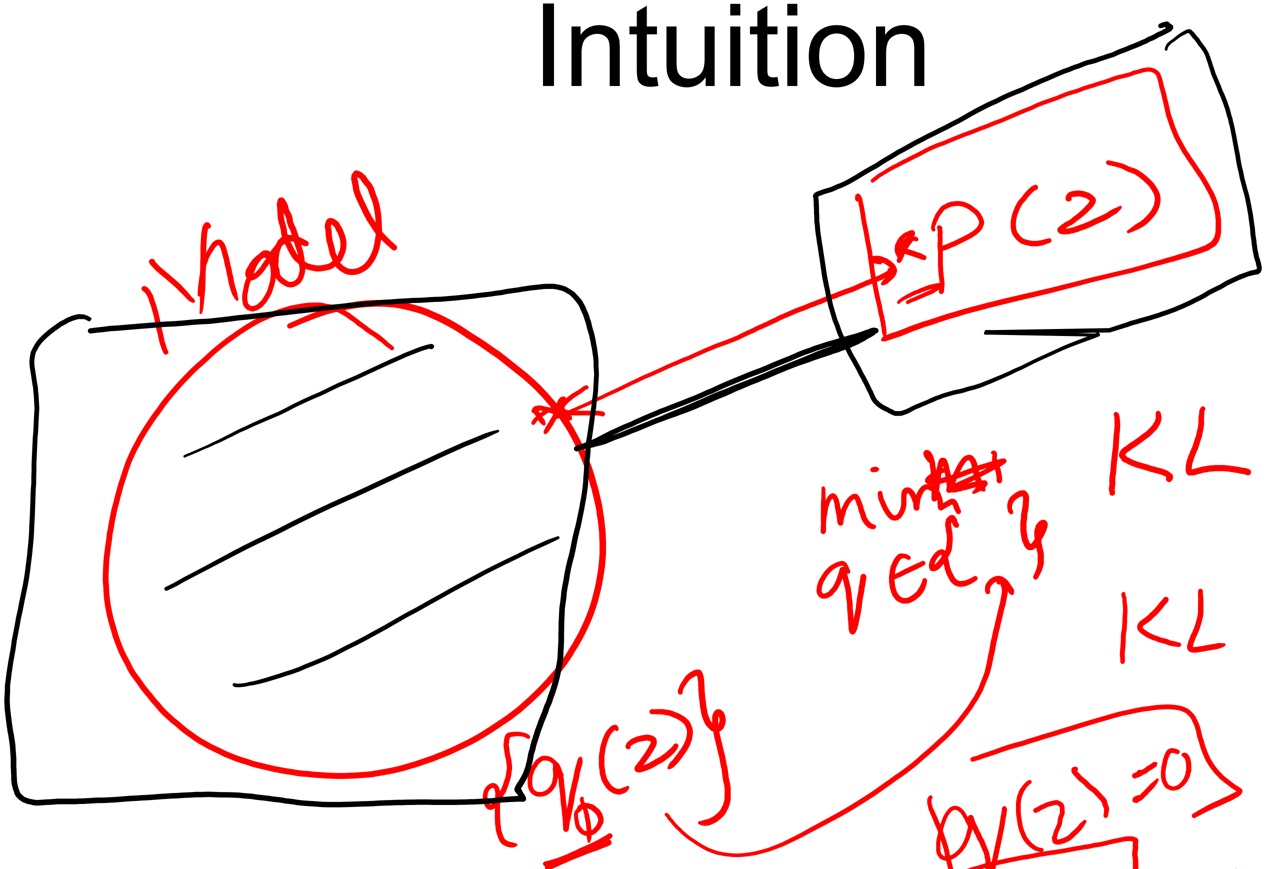
$\downarrow$

$q(z)$

What is Variational Inference?

- Key idea
  - Reality is complex
  - Can we approximate it with something "simple"?
  - Just need to make sure the simple thing is "close" to the complex thing.

Intuition



$$KL(p(z) \| q(z))$$

$$KL(q(z) \| p(z))$$

$q(z) = 0$

error

$$KL(p(z) \| q(z)) = \sum_z \boxed{p(z)} \log \left\{ \frac{p(z)}{q(z)} \right\}$$

$$-\sum_{\text{sample}} \log q(z)$$

The general learning problem with missing data

- Marginal likelihood – **x** is observed, **z** is missing:

$$
\begin{aligned}
ll(\theta : \mathcal{D}) &= \log \prod_{i=1}^{N} P(\mathbf{x}_i \mid \theta) \\
&= \sum_{i=1}^{N} \log P(\mathbf{x}_i \mid \theta) \\
&= \sum_{i=1}^{N} \log \sum_{\mathbf{z}} P(\mathbf{x}_i, \mathbf{z} \mid \theta)
\end{aligned}
$$

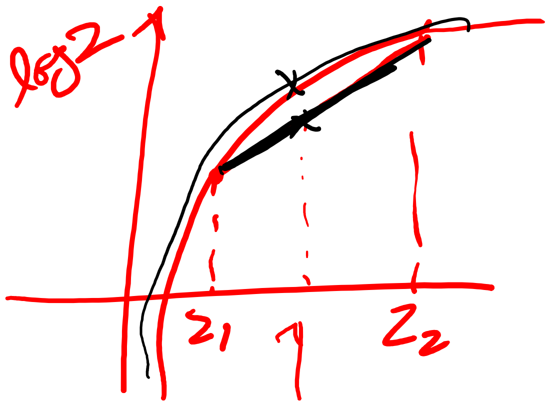
$P(\vec{x}_i, z)$

$= \log \sum_z P(\vec{x}_i \mid \theta) P(z \mid \vec{x}_i, \theta)$

$= \log E_{P(z \mid x_i, \theta)} [P(\vec{x}_i \mid \theta)]$

Applying Jensen's inequality

- Use: $\log \sum_{\mathbf{z}} P(\mathbf{z})\, g(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log g(\mathbf{z})$



$\lambda z_1 + \lambda_2 z_2$

$\lambda_1, \lambda_2 \geq 0$

$\lambda_1 + \lambda_2 = 1$

$$f(\lambda_1 z_1 + \lambda_2 z_2) \geq \lambda_1 f(z_1) + \lambda_2 f(z_2)$$

$$f\left(\sum_{i=1}^{K} \boxed{\lambda_i} z_i\right) \geq \sum_{i=1}^{K} \boxed{\lambda_i} f(z_i)$$

$$f(E[z]) \geq E[f(z)]$$

$z \to g(z)$

$$f(E[g(z)]) \geq E[f(g(z))]$$

Applying Jensen's inequality

- Use: $\log \sum_{\mathbf{z}} P(\mathbf{z})\, g(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log g(\mathbf{z})$

$$\log \sum_{z} \underbrace{P(\vec{x}_i, z \mid \theta) \cdot \frac{Q_i(z)}{Q_i(z)}}$$

$$\geq \sum_{z} \underbrace{Q_i(z)} \cancel{\log}\,\log \underbrace{\frac{P(\vec{x}_i, z \mid \theta)}{Q_i(z)}}$$

$$\boxed{F(\theta, Q_i)}$$

"Free Energy"

Variational

VLB / $\boxed{\text{ELBO}}$

Applying Jensen's inequality

- Use: log $\sum_{\mathbf{z}}$ P(**z**) g(**z**) ≥ $\sum_{\mathbf{z}}$ P(**z**) log g(**z**)

$P(z \mid \theta) P(x_i \mid z, \theta)$

$$F \qquad ll(\theta : \mathcal{D}) = \cancel{\sum_{i=1}^{N}} \log \sum_{\mathbf{z}} Q_i(\mathbf{z}) \frac{P(\mathbf{x}_i, \mathbf{z} \mid \theta)}{Q_i(\mathbf{z})}$$

$P(x_i \mid \theta) P(z \mid x_i, \theta)$

ELBO: Factorization #1

$$ll(\theta : \mathcal{D}) \geq F(\theta, Q_i) = \cancel{\sum_{i=1}^{N}} \sum_{\mathbf{z}} Q_i(\mathbf{z}) \log \frac{P(\mathbf{x}_i, \mathbf{z} \mid \theta)}{Q_i(\mathbf{z})}$$

$\rightarrow P(\vec{x}_i \mid \theta) P(z \mid \vec{x}_i, \theta)$

$$\sum_z Q_i(z) \log \left(\frac{\boxed{P(\vec{x}_i \mid \theta)} P(z \mid \vec{x}_i, \theta)}{Q_i(z)} \right)$$

$$= \underbrace{\sum_z Q_i(z)}_{1} \underbrace{\log P(\vec{x}_i \mid \theta)} + \left[\sum_z Q_i(z) \log \frac{P(z \mid \vec{x}_i, \theta)}{Q_i(z)} \right]$$

$$F(\theta, Q_i) = \log P(\vec{x}_i \mid \theta) - \left[\underbrace{KL\left(\boxed{Q_i(z)} \,\|\, \boxed{P(z \mid x_i, \theta)} \right)}_{0} \right]$$

$$\max_{\theta, Q_i} \boxed{F} = \underline{ll} - \underline{KL}$$

ELBO: Factorization #~~1~~ 2

$P(z|\theta)P(x_i|z,\theta)$

$$\max_{\theta, Q_i} F(\theta, Q_i)$$

$$\boxed{ll(\theta : \mathcal{D})} \geq F(\theta, Q_i) = \cancel{\sum_{i=1}^{N}} \sum_{\mathbf{z}} Q_i(\mathbf{z}) \log \frac{P(\mathbf{x}_i, \mathbf{z} \mid \theta)}{Q_i(\mathbf{z})}$$

$$= \sum_z Q_i(z) \log \frac{P(z|\theta)P(\vec{x}_i|z,\theta)}{Q_i(z)}$$

$$= \sum_z Q_i(z) \log P(\vec{x}_i|z,\theta) + \left[\sum_z Q_i(z) \log \frac{P(z|\theta)}{Q_i(z)}\right]$$

decoder

$$F = \underbrace{E_{Q_i(z)}\left[\log P(\vec{x}_i|z,\theta)\right]}_{\text{Explain the data}} - \underbrace{KL\left(Q_i(z) \,\|\, P(z|\theta)\right)}_{\text{Regularizer}}$$

encoder

Variational Auto Encoders

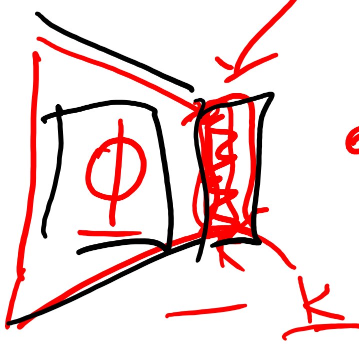
VAEs are a combination of the following ideas:

1. Auto Encoders
2. Variational Approximation
   - Variational Lower Bound / ELBO
3. Amortized Inference Neural Networks
4. "Reparameterization" Trick

Amortized Inference Neural Networks

softmax

x_i



$q_\phi(z|x_i)$ Encoder

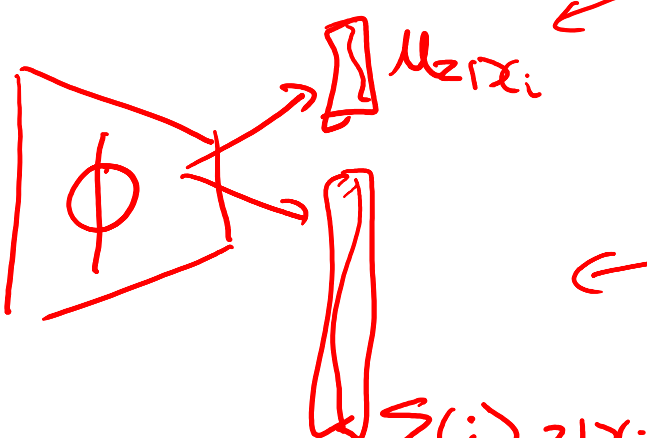
$z \sim Cat(\_)$

$\boxed{z \in \mathbb{R}^k}$

$z|x_i$ $z \sim \mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i})$

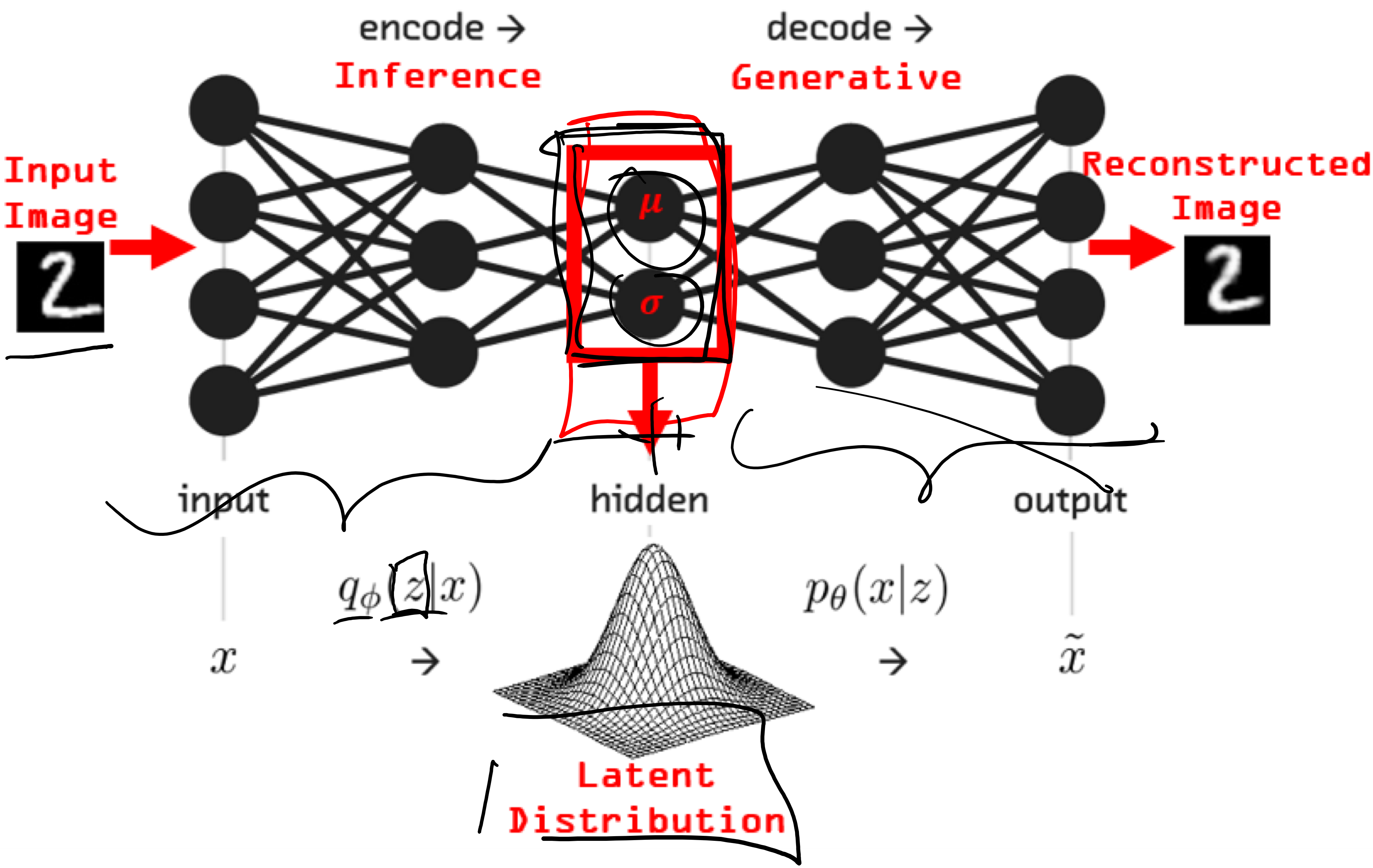
$\boxed{\text{mean}}$

$x|z \sim \mathcal{N}(.)$ x_i



$\boxed{\text{covariance}}$

VAEs



encode →
Inference
decode →
Generative
Input
Image
Reconstructed
Image
μ
σ
input
hidden
output
$q_\phi(z|x)$
$p_\theta(x|z)$
x
→
→
$\tilde{x}$
Latent
Distribution

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

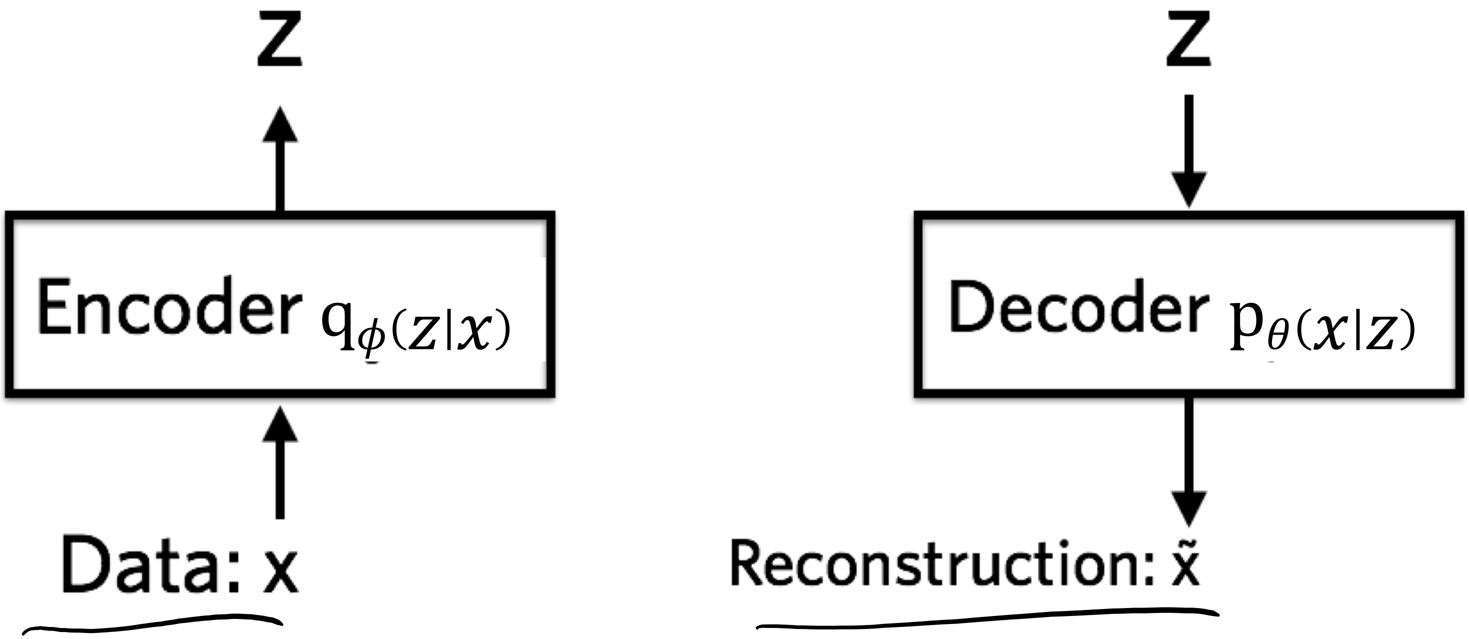
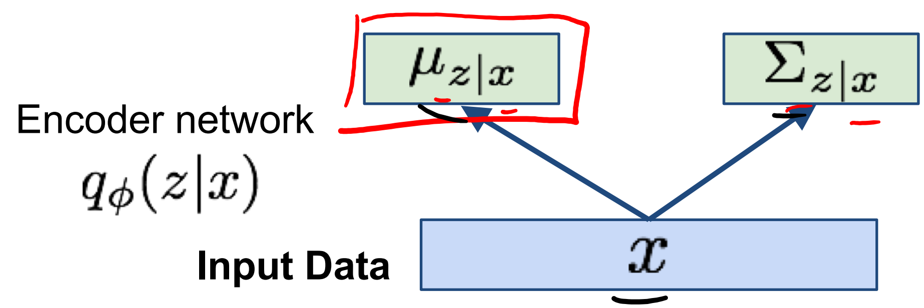


Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

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

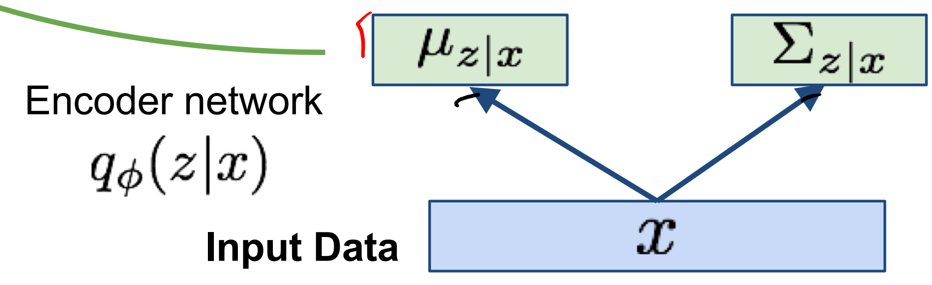


Variational Auto Encoders

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

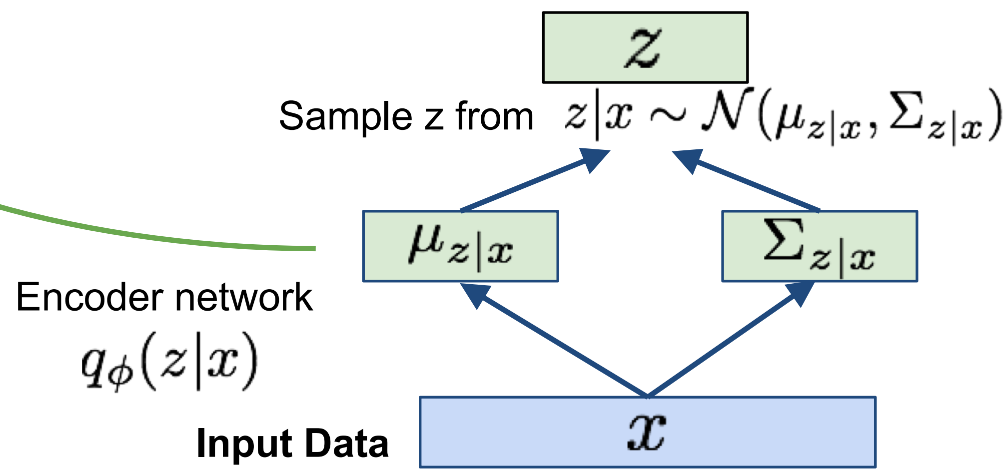


Variational Auto Encoders

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

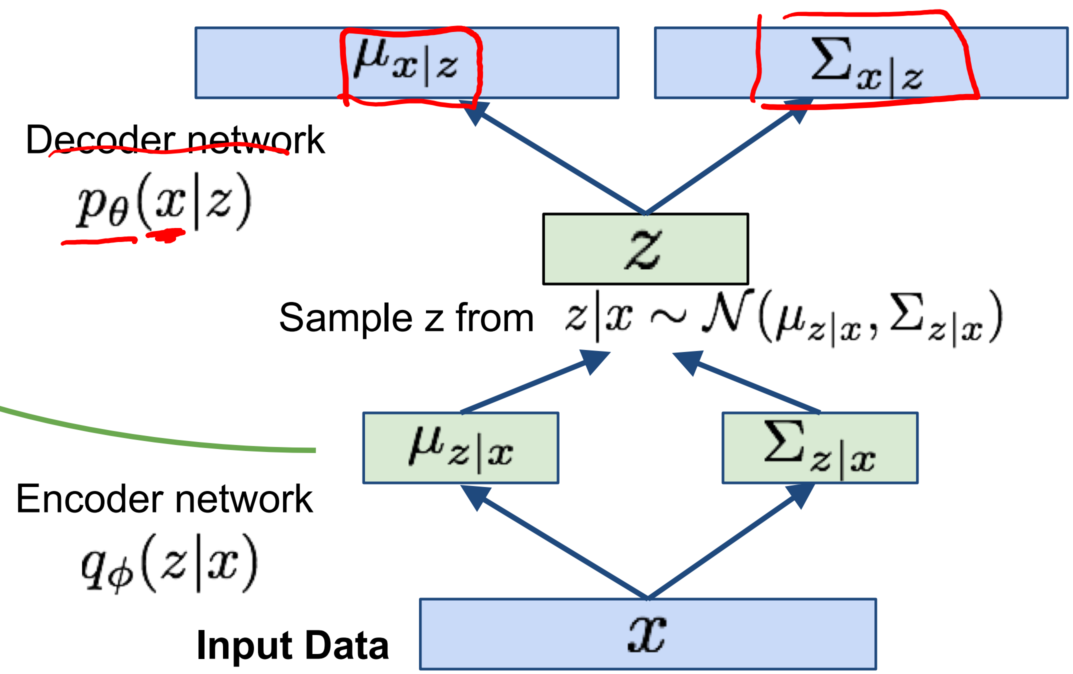


Variational Auto Encoders

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



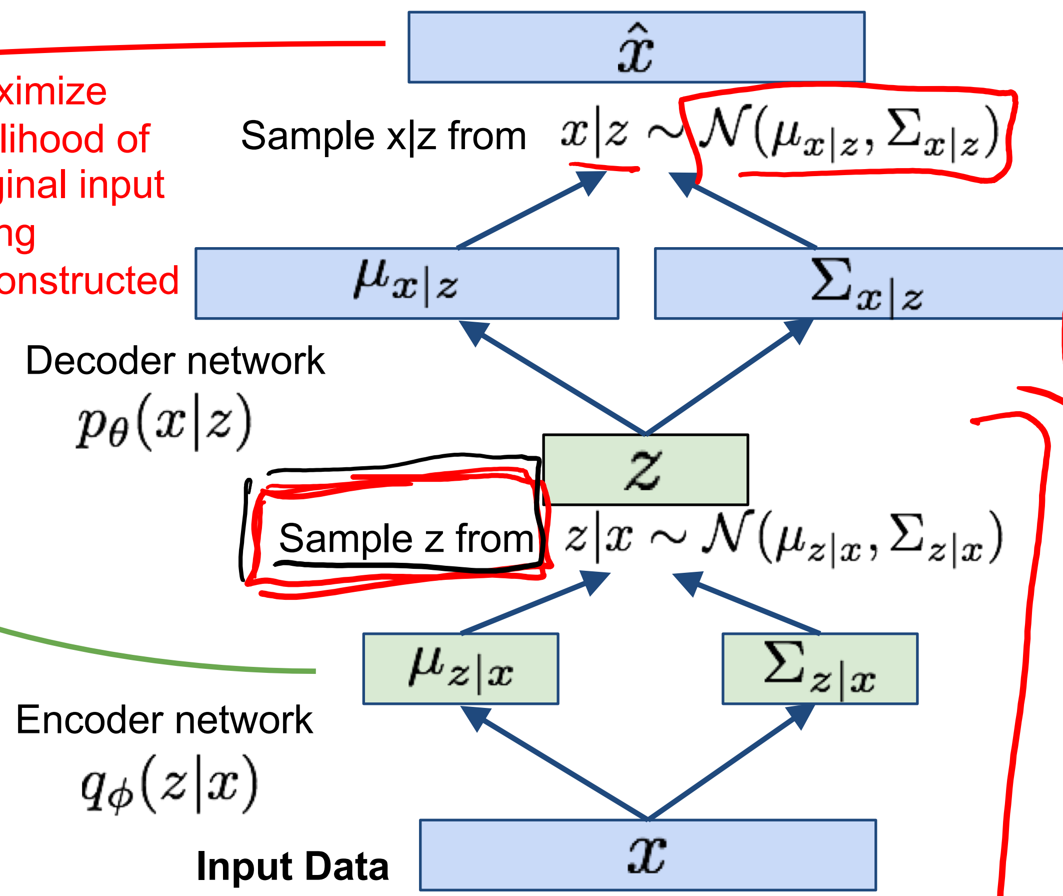
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

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

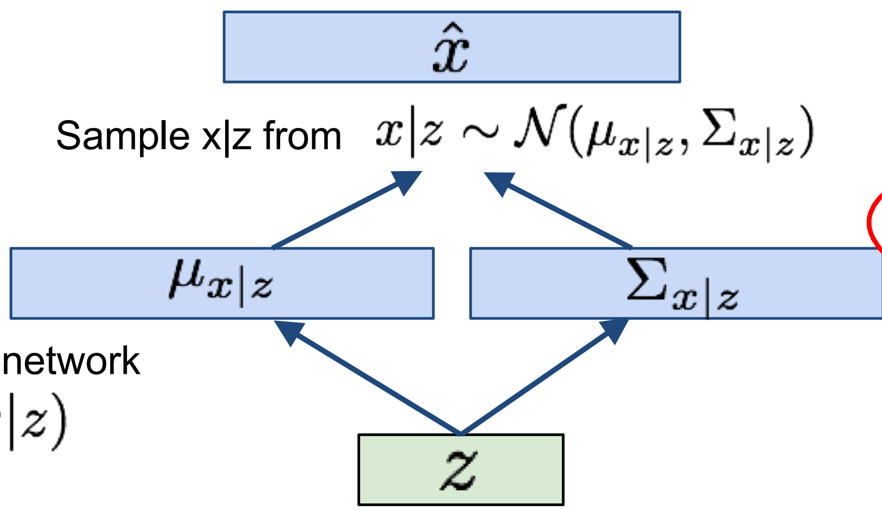
Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior



Variational Auto Encoders: Generating Data

Use decoder network. Now sample z from prior!



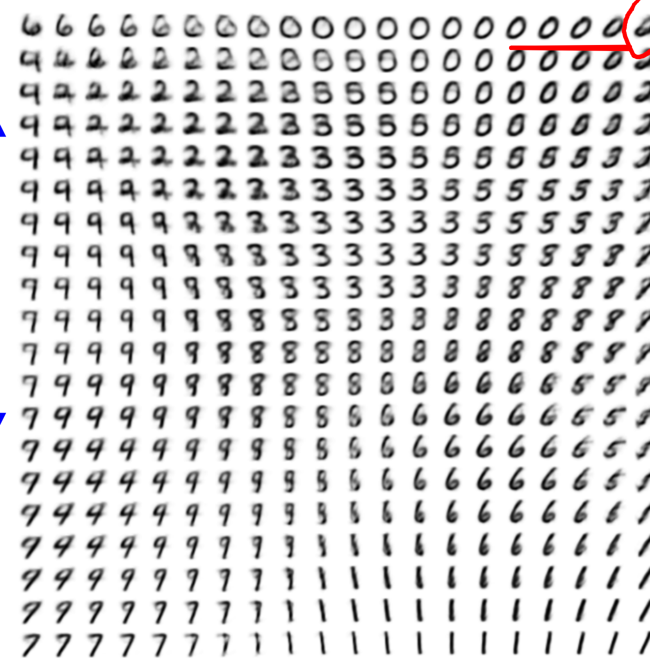
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

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**



Vary **z_1**

Vary **z_2**

Variational Auto Encoders: Generating Data

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Degree of smile

Vary z_1

Vary z_2

Head pose

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

Plan for Today

- VAEs
  - Reparameterization trick

- Generative Adversarial Networks (GANs)

Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   - Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick

Basic Problem

- Goal

$$\min_{\boxed{\theta}} \boxed{\mathbb{E}_{z\sim p_\theta(z)}[f(z)]}$$

$$\theta^{(t)} \leftarrow \theta^{(t-1)} - \nabla_\theta$$

Basic Problem

- Goal $$\underline{\min_{\theta}}\ \underline{\mathbb{E}_{z \sim p_\theta(z)}}[\underline{f(z)}]$$

- Need to compute: $$\underline{\nabla_\theta}\ \mathbb{E}_{\boxed{z \sim p_\theta(z)}}[\underline{f(z)}]$$

$$\nabla_\theta \int f_\theta(z)\, p(z)\, dz$$

$$\int \nabla_\theta f_\theta(z) \cdot p(z)\, dz$$

$$E_z \left[\nabla_\theta f_\theta(z) \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f_\theta(z_i)$$

$$\nabla_\theta \int f(z)\, p_\theta(z)\, dz$$

$$\int \nabla_\theta\ \underline{f(z)}\ \underline{p_\theta(z)}\ dz$$

$$\int \cancel{\nabla_\theta} f(z)\, \nabla_\theta p_\theta(z)\, dz$$

Does this happen in supervised learning?

- Goal

$$\min_\theta \mathbb{E}_{z\sim p_\theta(z)}[f(z)]$$

$$\min_\theta \mathbb{E}_{x,y\sim P_{data}}\left[\ell(y, \hat{y}(x,\theta))\right]$$

$$= \mathbb{E}_{x,y\sim P_{data}}\left[\nabla_\theta \ell(\ldots\theta)\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\theta \ell(y_i, \hat{y}(x_i,\theta))\right]$$

Example

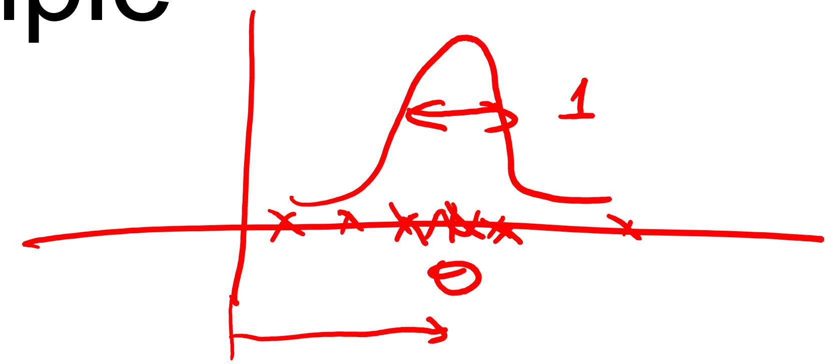
$z \sim N(\theta, 1)$

$f(z) = z^2$

$$\min_{\theta} \; E_z[f(z)]$$

$$\min_{\theta} \nabla_\theta \int z^2 \, p(z) \, dz$$

$$\int z^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{(z-\theta)^2}{2}} \, dz$$



$$Var(z) = E[(z-\theta)^2]$$

$$= E[z^2] - \theta^2$$

$$E[z^2] = \theta^2 + \boxed{Var(z)}$$

Example

Two Options

- Score Function based Gradient Estimator
  aka REINFORCE (and variants)

$$\nabla_\theta \mathbb{E}_z \left[f(z) \right] = \mathbb{E}_z \left[f(z) \nabla_\theta \log p_\theta(z) \right]$$

- Path Derivative Gradient Estimator
  aka "reparameterization trick"

$$\frac{\partial}{\partial\theta} \mathbb{E}_{z \sim p_\theta} \left[f(z) \right] = \frac{\partial}{\partial\theta} \mathbb{E}_\epsilon \left[f(g(\theta, \epsilon)) \right] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right]$$

Option 1

- Score Function based Gradient Estimator
  aka REINFORCE (and variants)

$$\nabla_\theta \mathbb{E}_z\left[f(z)\right] = \mathbb{E}_z\left[f(z)\nabla_\theta \log p_\theta(z)\right]$$

$$\nabla_\theta \int f(z)\, p_\theta(z)\, dz$$

$$= \int f(z)\, \nabla_\theta p_\theta(z)\, dz \cdot \frac{p_\theta(z)}{p_\theta(z)}$$

$$= \int f(z)\, \boxed{\nabla_\theta \log p_\theta(z)} \cdot \boxed{p_\theta(z)} \cdot dz$$

$$= \mathbb{E}_{z\sim p_\theta(z)}\left[f(z)\, \nabla_\theta \log p_\theta(z)\right] \approx \frac{1}{N}\sum$$

Example

$$P_\theta(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(z-\theta)^2}{2}}$$

$$\frac{\partial}{\partial\theta} \log P_\theta(z) = -\frac{(z-\theta)^2}{2} - \cancel{\frac{1}{2}\log 2\pi}$$

$$= -\frac{2(z-\theta)\cdot(-1)}{2} = (z-\theta)$$

$$\nabla_\theta = E\left[z^2(z-\theta)\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}(z_i^2)(z_i-\theta)$$

Two Options

- Score Function based Gradient Estimator aka REINFORCE (and variants)

$$\nabla_\theta \mathbb{E}_z \left[f(z) \right] = \mathbb{E}_z \left[f(z) \nabla_\theta \log p_\theta(z) \right]$$

- Path Derivative Gradient Estimator aka “reparameterization trick”

$$\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} \left[f(z) \right] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon \left[f(g(\theta, \epsilon)) \right] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right]$$

Option 2

- Path Derivative Gradient Estimator
  aka “reparameterization trick”

$$\frac{\partial}{\partial\theta}\mathbb{E}_{z\sim p_\theta}\left[f(z)\right] = \frac{\partial}{\partial\theta}\mathbb{E}_{\epsilon}\left[f(g(\theta,\epsilon))\right] = \mathbb{E}_{\epsilon\sim p_\epsilon}\left[\frac{\partial f}{\partial g}\frac{\partial g}{\partial\theta}\right]$$

$z \sim p_\theta(z)$

$$\boxed{z = g(\theta, \epsilon)}$$

$\epsilon \sim U(0,1)$

$\sim N(0,1)$

$z \sim N(\mu, \sigma^2)$ $\epsilon \sim N(0,1)$

$\rightarrow z = \mu + \sigma\epsilon \leftarrow$ simple RV

constants

Option 2

- Path Derivative Gradient Estimator
  aka "reparameterization trick"

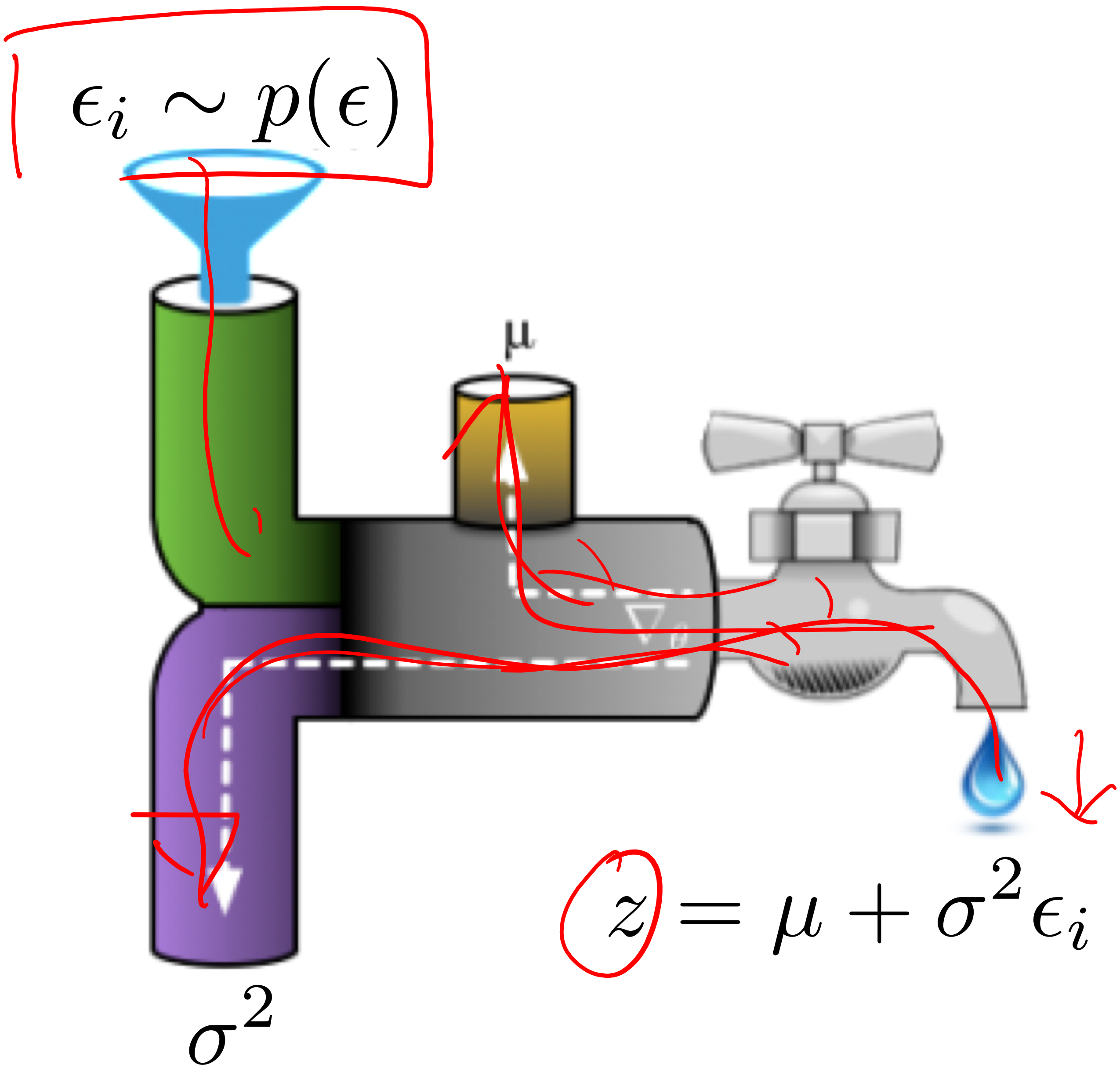
$$\frac{\partial}{\partial\theta}\mathbb{E}_{z\sim p_\theta}\left[f(z)\right] = \frac{\partial}{\partial\theta}\mathbb{E}_{\epsilon}\left[f(g(\theta,\epsilon))\right] = \boxed{\mathbb{E}_{\epsilon\sim p_\epsilon}}\left[\frac{\partial f}{\partial g}\frac{\partial g}{\partial\theta}\right] \approx \frac{1}{N}\sum\left(\ \right)$$

$$\frac{\partial}{\partial\theta}\int f(g(\theta,\varepsilon))\cdot p(\varepsilon)\cdot d\varepsilon$$

$$= \int \frac{\partial}{\partial\theta} f(g(\theta,\varepsilon))\cdot p(\varepsilon)\cdot d\varepsilon$$

$$= \int \frac{\partial f}{\partial g}\frac{\partial g}{\partial\theta}\cdot p(\varepsilon)\, d\varepsilon$$

Reparameterization Intuition



Example

$z \sim N(\theta, 1)$ $\quad z = \theta + \varepsilon \quad \boxed{\varepsilon \sim N(0,1)}$

$f(z) = (\theta + \varepsilon)^2$

$$\cancel{\min_\theta} \nabla_\theta \, E_{\varepsilon \sim p(\varepsilon)} \left[(\theta+\varepsilon)^2 \right]$$

$$= E_\varepsilon \left[\nabla_\theta (\theta+\varepsilon)^2 \right]$$

$$= E_\varepsilon \left[2(\theta+\varepsilon) \right] = \underline{2\theta} + \underbrace{\boxed{E_\varepsilon[\varepsilon]}}_{0}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} 2(\theta + \varepsilon_i)$$

Two Options

- Score Function based Gradient Estimator
  aka REINFORCE (and variants)

$$\nabla_\theta \mathbb{E}_z \left[f(z) \right] = \mathbb{E}_z \left[f(z) \nabla_\theta \log p_\theta(z) \right]$$

- Path Derivative Gradient Estimator
  aka "reparameterization trick"

$$\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} \left[f(z) \right] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon \left[f(g(\theta, \epsilon)) \right] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right]$$

Example

```
import numpy as np
N = 1000
theta = 2.0
x = np.random.randn(N) + theta
eps = np.random.randn(N)

grad1 = lambda x: np.sum(np.square(x)*(x-theta)) / x.size
grad2 = lambda eps: np.sum(2*(theta + eps)) / x.size

print grad1(x)
print grad2(eps)
```

```
4.46239612174
4.1840532024
```

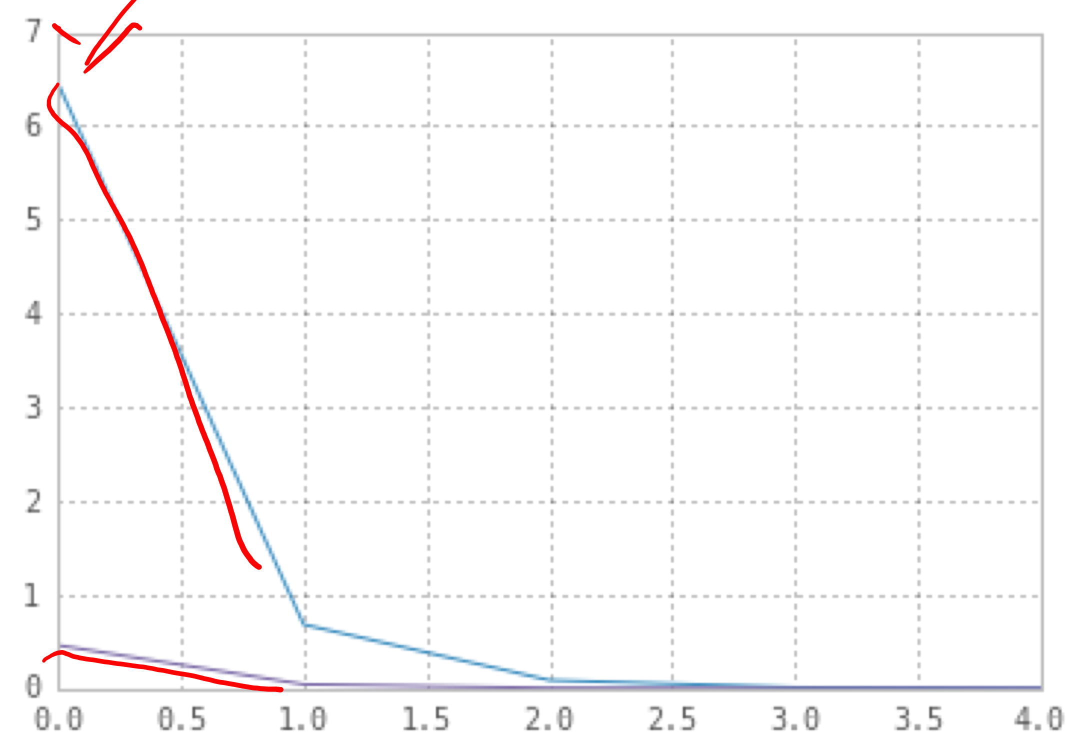
20

Example

```
Ns = [10, 100, 1000, 10000, 100000]
reps = 100

means1 = np.zeros(len(Ns))
vars1 = np.zeros(len(Ns))
means2 = np.zeros(len(Ns))
vars2 = np.zeros(len(Ns))

est1 = np.zeros(reps)
est2 = np.zeros(reps)
for i, N in enumerate(Ns):
    for r in range(reps):
        x = np.random.randn(N) + theta
        est1[r] = grad1(x)
        eps = np.random.randn(N)
        est2[r] = grad2(eps)
    means1[i] = np.mean(est1)
    means2[i] = np.mean(est2)
    vars1[i] = np.var(est1)
    vars2[i] = np.var(est2)

print means1
print means2
print
print vars1
print vars2
```

```
[ 3.8409546   3.97298803  4.03007634  3.98531095  3.99579423]
[ 3.97775271  4.00232825  3.99894536  4.00353734  3.99995899]

[  6.45307927e+00   6.80227241e-01   8.69226368e-02   1.00489791e-02
   8.62396526e-04]
[  4.59767676e-01   4.26567475e-02   3.33699503e-03   5.17148975e-04
   4.65338152e-05]
```

REINFORCE



Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
    - Variational Lower Bound / ELBO

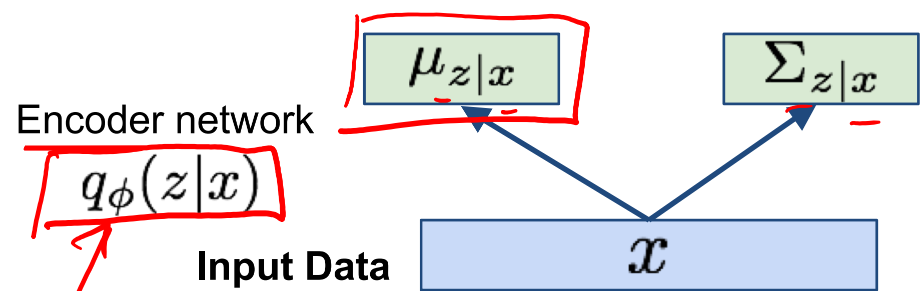
3. Amortized Inference Neural Networks

4. "Reparameterization" Trick

Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

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



ϕ

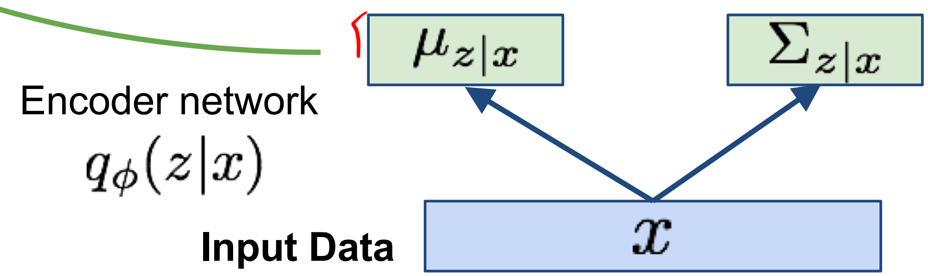
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Variational Auto Encoders

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

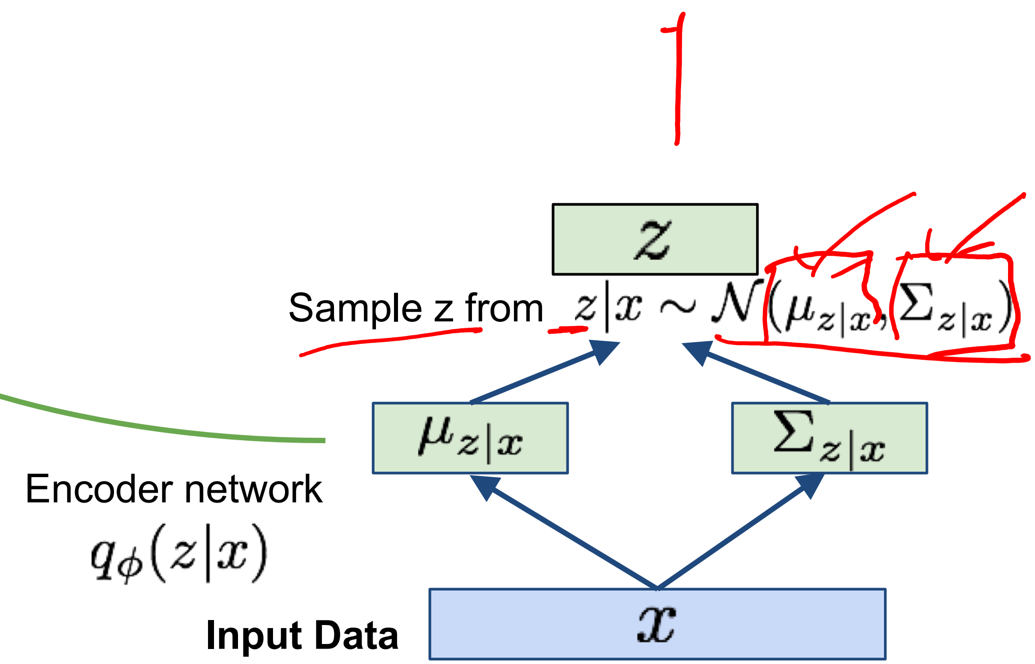


Variational Auto Encoders

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



Variational Auto Encoders

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