

AI504: Programming for Artificial Intelligence

Week 5: Variational Autoencoder

Edward Choi

Grad School of AI

edwardchoi@kaist.ac.kr

Today's Topic

- Generative Autoencoder
- Variational Inference
- Variational Autoencoder
- Training VAE
 - Reparametrization Trick

Generative Autoencoder

Autoencoder

- Consists of Encoder and Decoder



Encoding (Compression)



-1.2
3.1
0.2
-0.9



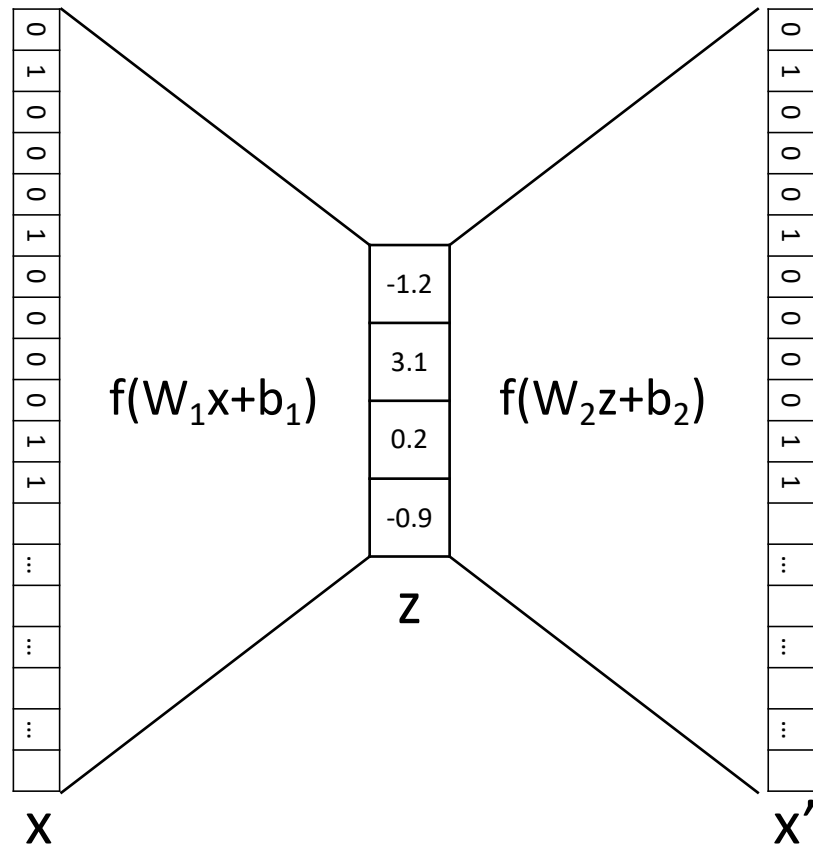
Decoding (Decompression)



Code , Latent Variables, Latent Representation

Autoencoder

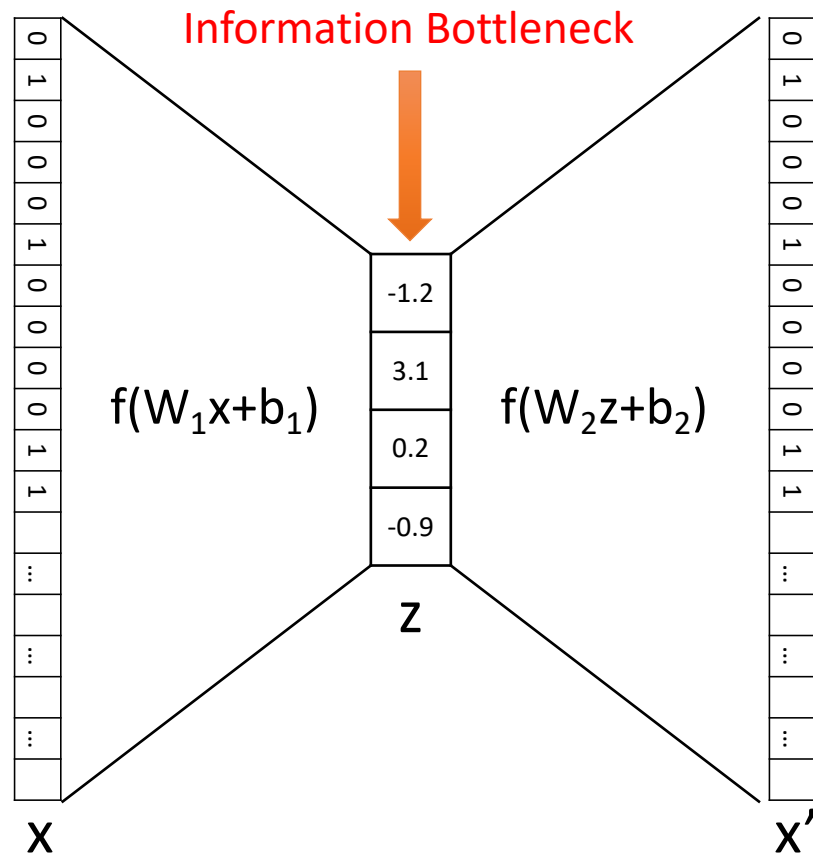
- Mean Squared Error (MSE) loss



- Encoding
 - $z = f(W_1x+b_1)$
- Decoding
 - $x' = f(W_2z+b_2)$
- Loss
 - $\mathcal{L}(x, x') = \|x - x'\|_2^2$
(Squared Error)

Autoencoder

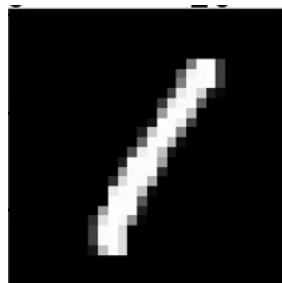
- Compression



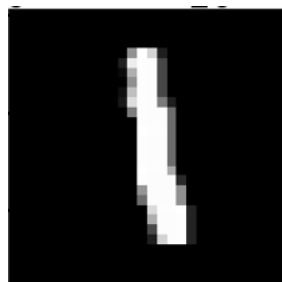
Need to pack all information in 4-D

➔ Need to learn some useful hidden features

Latent Representations



0.5
-2.1
-1.8
1.7



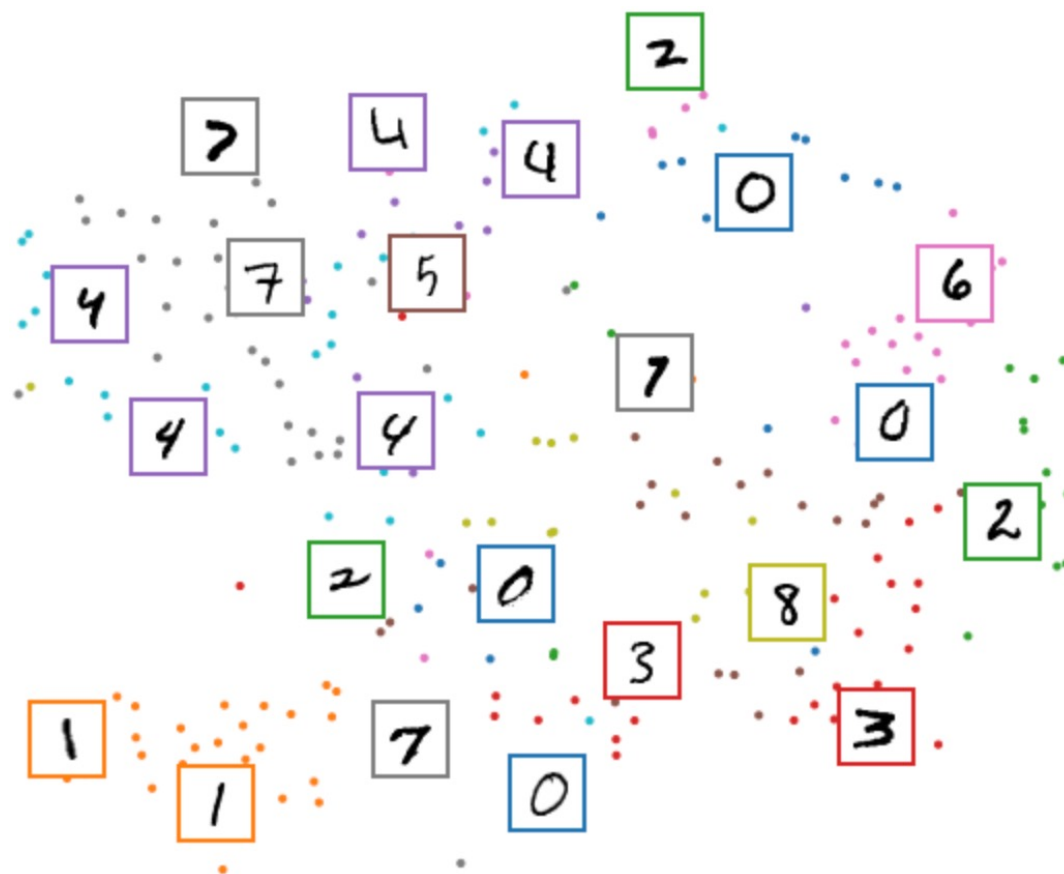
0.9
-1.9
-2.6
1.9



-1.2
3.1
0.2
-0.9

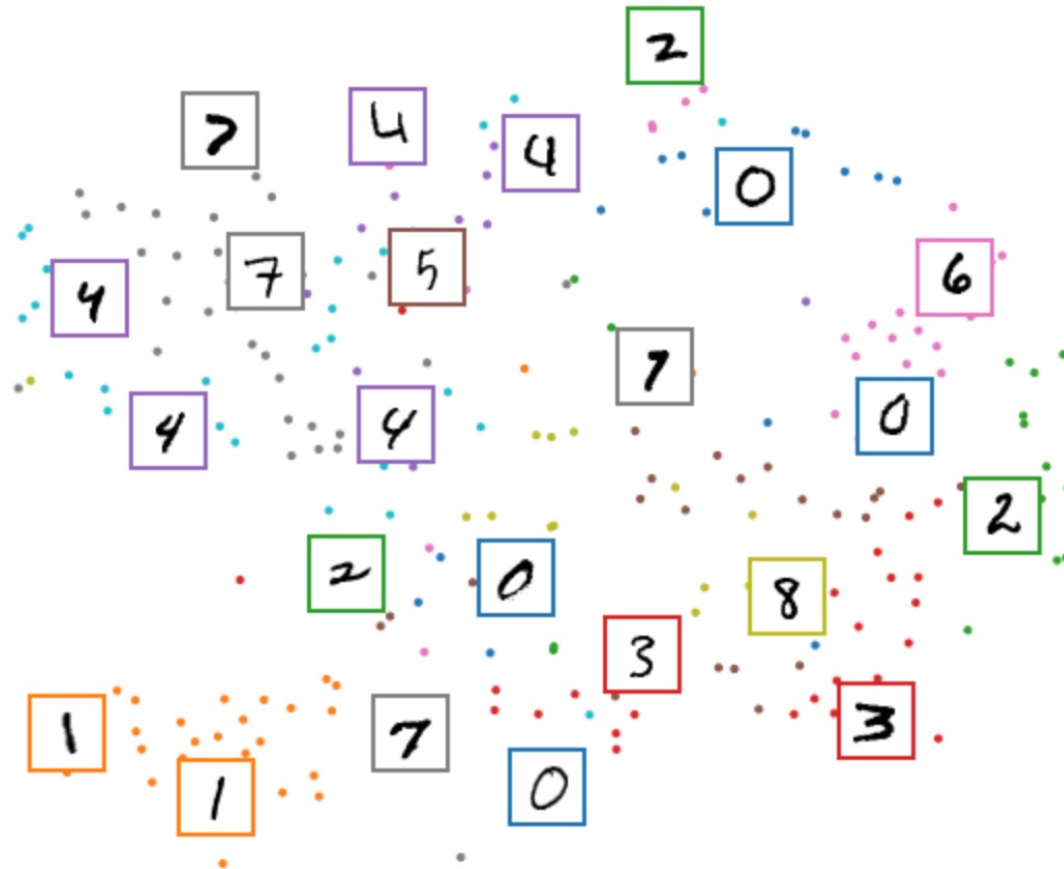
Latent Representation

- Digits are compressed to a 4-D space.



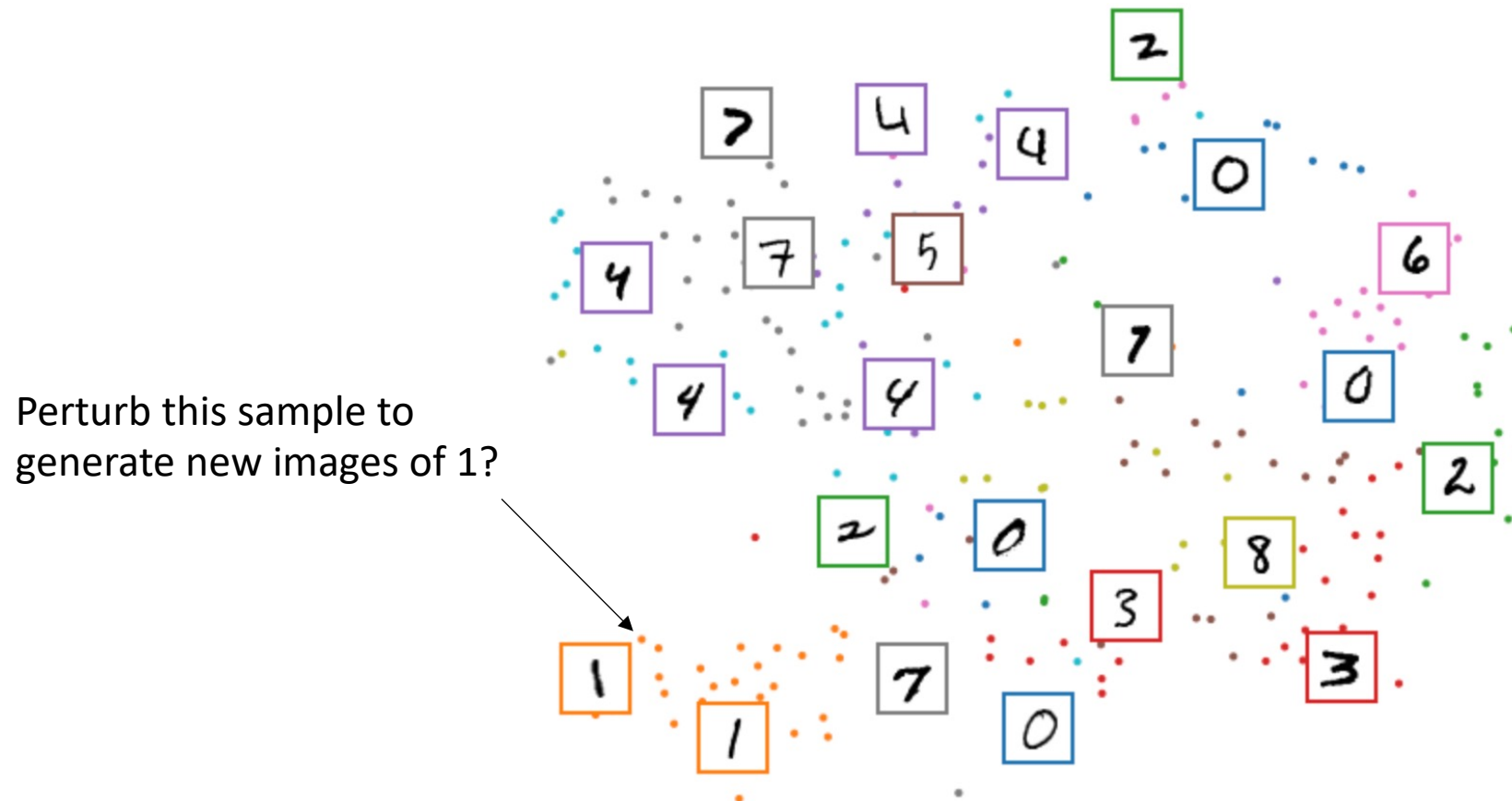
Generating Samples

- How can we generate new samples?



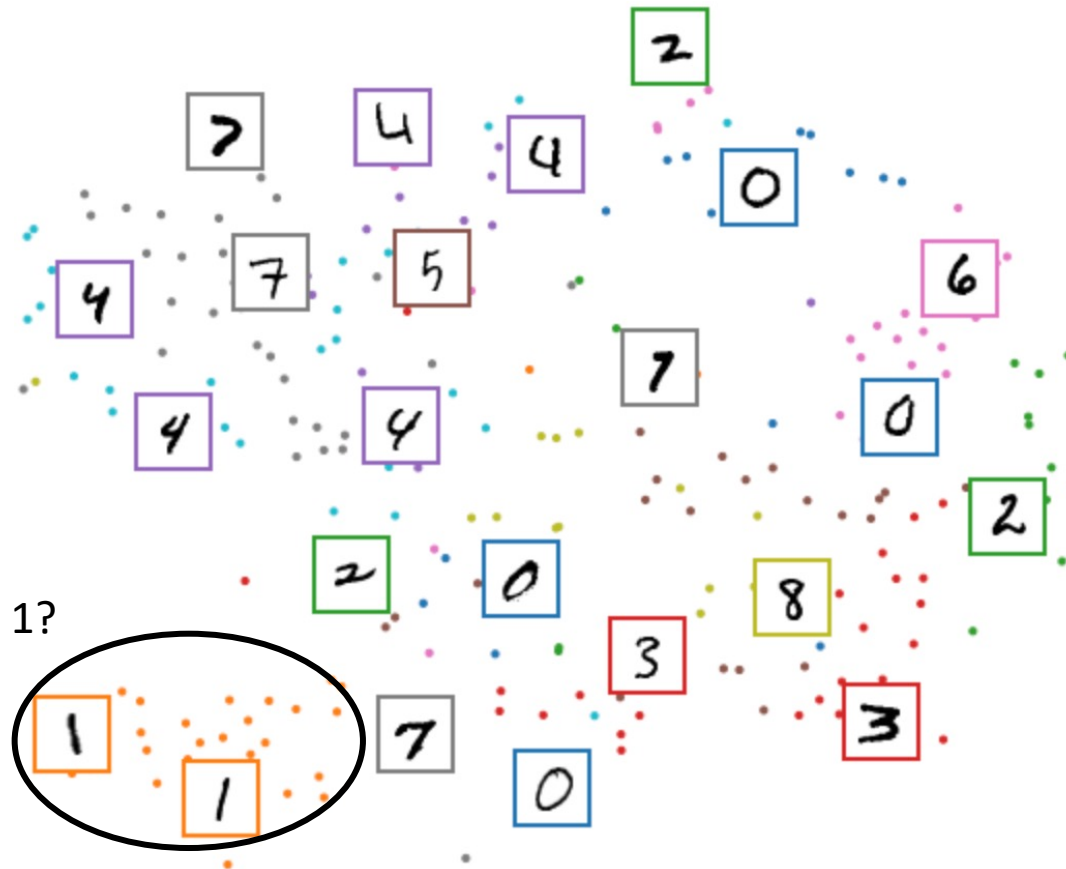
Generating New Samples

- Perturb an existing \mathbf{z} ?



Generating New Samples

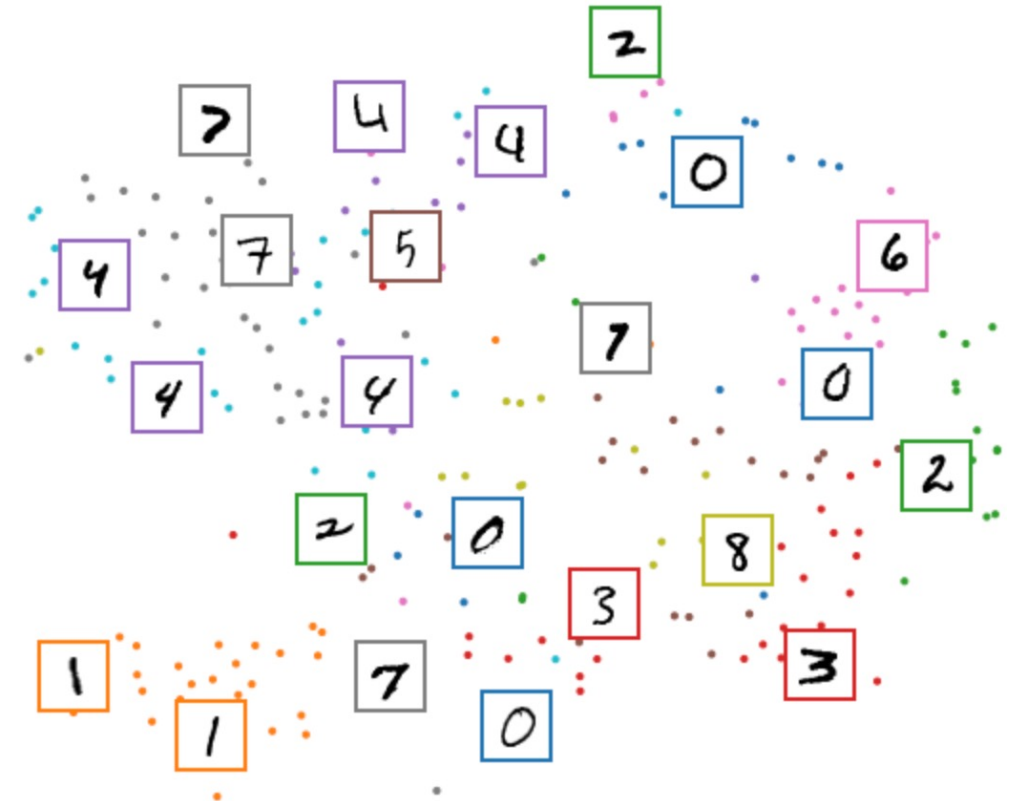
- Sample from a region?



Sample random points from
this area to generate images of 1?

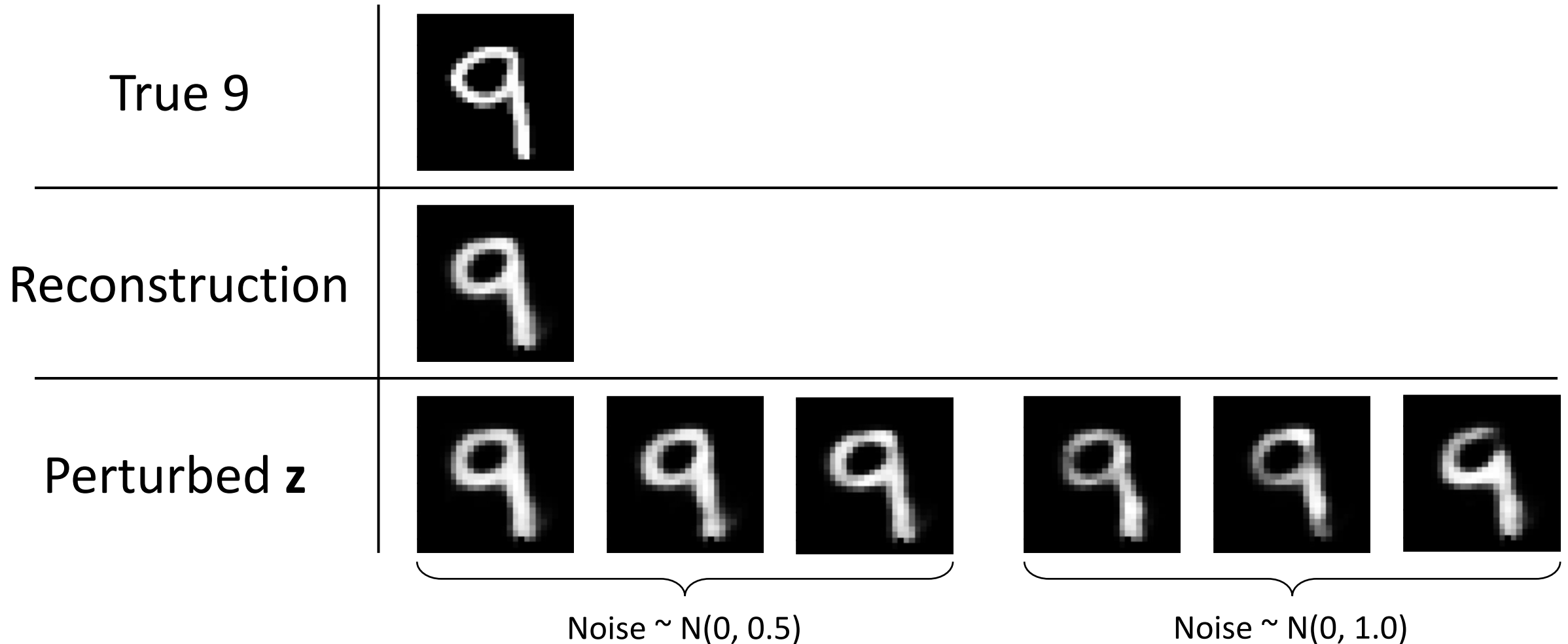
Generating New Samples

- Generating new samples
 - Perturb a known z
 - Sample from a region
 - Which region do you sample?
- Both are not guaranteed to work



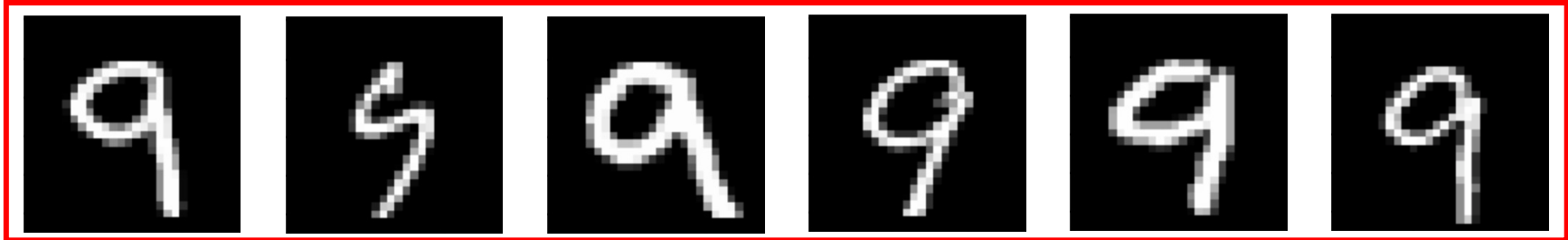
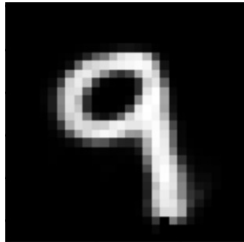
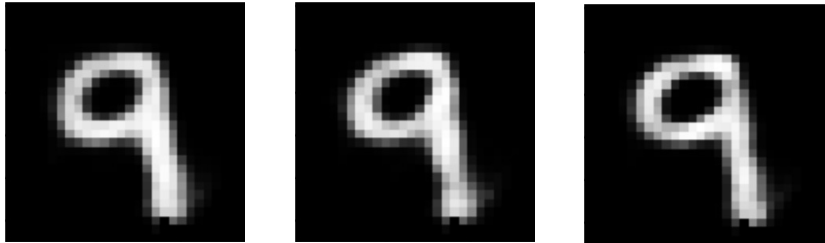
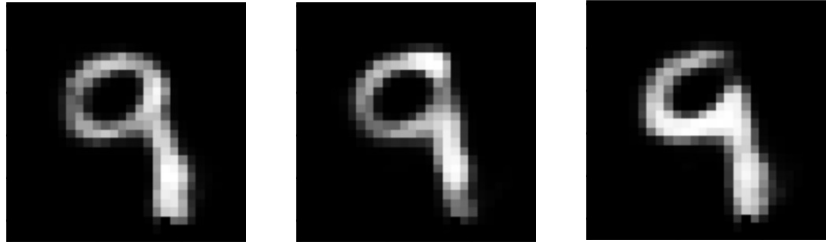
Example: Perturbing z

- Using an AE trained on MNIST for 50 epochs.



Example: Perturbing z

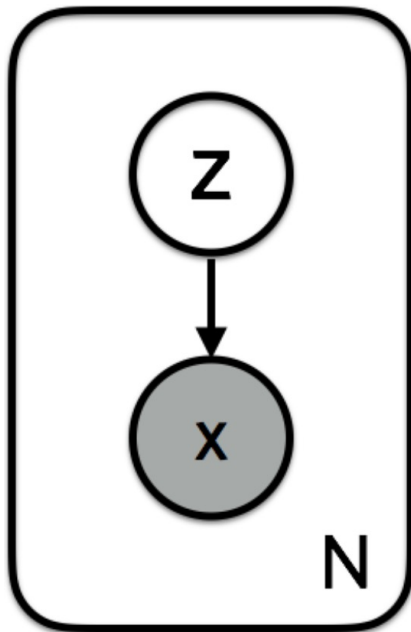
- Cannot generate diverse/novel samples

True 9						
Reconstruction						
Perturbed z						
	Noise $\sim N(0, 0.5)$			Noise $\sim N(0, 1.0)$		

Variational Inference

Posterior Distribution

- Given data \mathbf{X} and unobserved (latent/hidden) variables \mathbf{Z}
 - Assume \mathbf{Z} determines \mathbf{X}
 - We are interested in the value $P(\mathbf{Z} \mid \mathbf{X})$



- Draw latent variables $z_i \sim p(z)$
- Draw datapoint $x_i \sim p(x \mid z)$

Posterior Distribution

- Given data \mathbf{X} and unobserved (latent/hidden) variables \mathbf{Z}
 - Assume \mathbf{Z} determines \mathbf{X}
 - We are interested in the value $P(\mathbf{Z} \mid \mathbf{X})$
- Ex) Topic (\mathbf{Z}) of newspaper articles (\mathbf{X})
- Ex) Latent representations (\mathbf{Z}) of MNIST images (\mathbf{X})

Posterior Distribution

- Given data \mathbf{X} and unobserved (latent/hidden) variables \mathbf{Z}
 - Assume \mathbf{Z} determines \mathbf{X}
 - We are interested in the value $P(\mathbf{Z} \mid \mathbf{X})$
- Usually, a true posterior distribution is intractable
 - Using Bayesian principle,

$$P(\mathbf{Z} \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{P(\mathbf{X})} = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{\int_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}}$$

Posterior Distribution

- Given data \mathbf{X} and unobserved (latent/hidden) variables \mathbf{Z}
 - Assume \mathbf{Z} determines \mathbf{X}
 - We are interested in the value $P(\mathbf{Z} \mid \mathbf{X})$
- Usually, a true posterior distribution is intractable
 - Using Bayesian principle,

$$P(\mathbf{Z} \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{P(\mathbf{X})} = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{\int_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}}$$

↑
This is often combinatorially large (even infinite!)

Posterior Distribution

- Given data \mathbf{X} and unobserved (latent/hidden) variables \mathbf{Z}
 - Assume \mathbf{Z} determines \mathbf{X}
 - We are interested in the value $P(\mathbf{Z} \mid \mathbf{X})$
- Usually, a true posterior distribution is intractable
 - Using Bayesian principle,

$$P(\mathbf{Z} \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{P(\mathbf{X})} = \frac{P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Z})}{\int_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}}$$

- Instead, approximate with a simpler function Q (often Gaussian)!

$$P(\mathbf{Z} \mid \mathbf{X}) \approx Q(\mathbf{Z})$$

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} \mid \mathbf{X})$ as possible
 - Minimize *Kullback-Leibler divergence* (KL-Divergence)

$$D_{\text{KL}}(Q \parallel P) \triangleq \sum_{\mathbf{Z}} Q(\mathbf{Z}) \log \frac{Q(\mathbf{Z})}{P(\mathbf{Z} \mid \mathbf{X})} \quad (\text{Assuming } \mathbf{Z} \text{ is a discrete variable})$$

- Note that D_{KL} is not symmetric (i.e. it is not a distance!)
- This formulation is called “Reverse KL”

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} | \mathbf{X})$ as possible
 - Minimize Kullback-Leibler divergence (KL-Divergence)

$$\begin{aligned} D_{\text{KL}}(Q \parallel P) &\triangleq \sum_{\mathbf{Z}} Q(\mathbf{Z}) \log \frac{Q(\mathbf{Z})}{P(\mathbf{Z} | \mathbf{X})} \\ &= \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\log \frac{Q(\mathbf{Z})}{P(\mathbf{Z}, \mathbf{X})} + \log P(\mathbf{X}) \right] \\ &= \sum_{\mathbf{Z}} Q(\mathbf{Z}) [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \sum_{\mathbf{Z}} Q(\mathbf{Z}) [\log P(\mathbf{X})] \\ &= \sum_{\mathbf{Z}} Q(\mathbf{Z}) [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X}) \\ &= \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X}) \end{aligned}$$

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} | \mathbf{X})$ as possible
 - Minimize Kullback-Leibler divergence (KL-Divergence)

$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})$$



$$\log P(\mathbf{X}) = D_{\text{KL}}(Q \parallel P) - \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})]$$

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} | \mathbf{X})$ as possible
 - Minimize Kullback-Leibler divergence (KL-Divergence)

$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})$$



$$\log P(\mathbf{X}) = D_{\text{KL}}(Q \parallel P) - \underbrace{\mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})]}_{\mathcal{L}(Q): \text{Evidence Lower Bound (ELBO)}}$$

- Maximizing the ELBO leads to minimizing D_{KL}

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} | \mathbf{X})$ as possible
 - Minimize Kullback-Leibler divergence (KL-Divergence)

$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})$$



$$\log P(\mathbf{X}) = D_{\text{KL}}(Q \parallel P) - \underbrace{\mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})]}_{\mathcal{L}(Q): \text{Evidence Lower Bound (ELBO)}}$$

- Maximizing the ELBO leads to minimizing D_{KL}
- We converted an **inference** problem to an **optimization** problem!
 - For Gaussian Q , we can learn μ and σ .

Learning $Q(\mathbf{Z})$

- We want $Q(\mathbf{Z})$ as similar to $P(\mathbf{Z} | \mathbf{X})$ as possible
 - Minimize Kullback-Leibler divergence (KL-Divergence)

$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})$$



$$\log P(\mathbf{X}) = D_{\text{KL}}(Q \parallel P) - \underbrace{\mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})]}_{\mathcal{L}(Q): \text{Evidence Lower Bound (ELBO)}}$$

- $\mathcal{L}(Q)$ is also called “variational free energy”

$$\mathcal{L}(Q) = \mathbb{E}_{\mathbf{Z}} [\log P(\mathbf{Z}, \mathbf{X})] + H(Q) \quad (H: \text{Information Entropy})$$

Variational Autoencoder

VAE

- Objective
 - Compress \mathbf{x} to \mathbf{z} which follows $P(\mathbf{Z} | \mathbf{X})$
 - Decompress \mathbf{z} to reconstruct \mathbf{x}



Encoding (Compression)



-1.2
3.1
0.2
-0.9



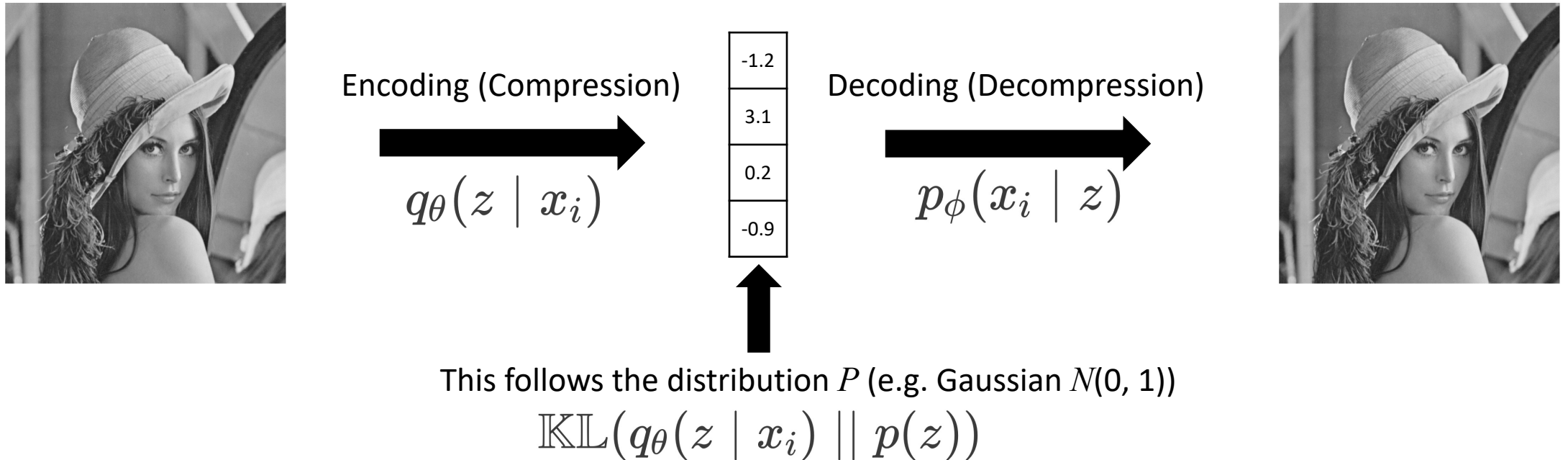
Decoding (Decompression)



This follows the distribution P (e.g. Gaussian)

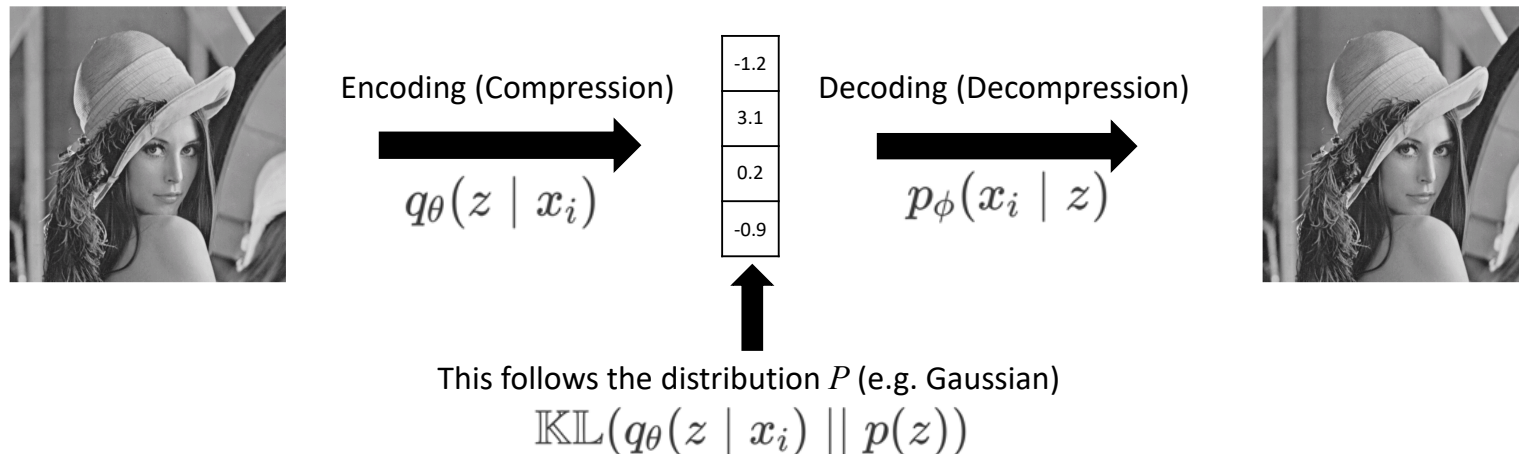
VAE

- Objective
 - Compress \mathbf{x} to \mathbf{z} which follows $P(\mathbf{Z} | \mathbf{X})$
 - Decompress \mathbf{z} to reconstruct \mathbf{x}

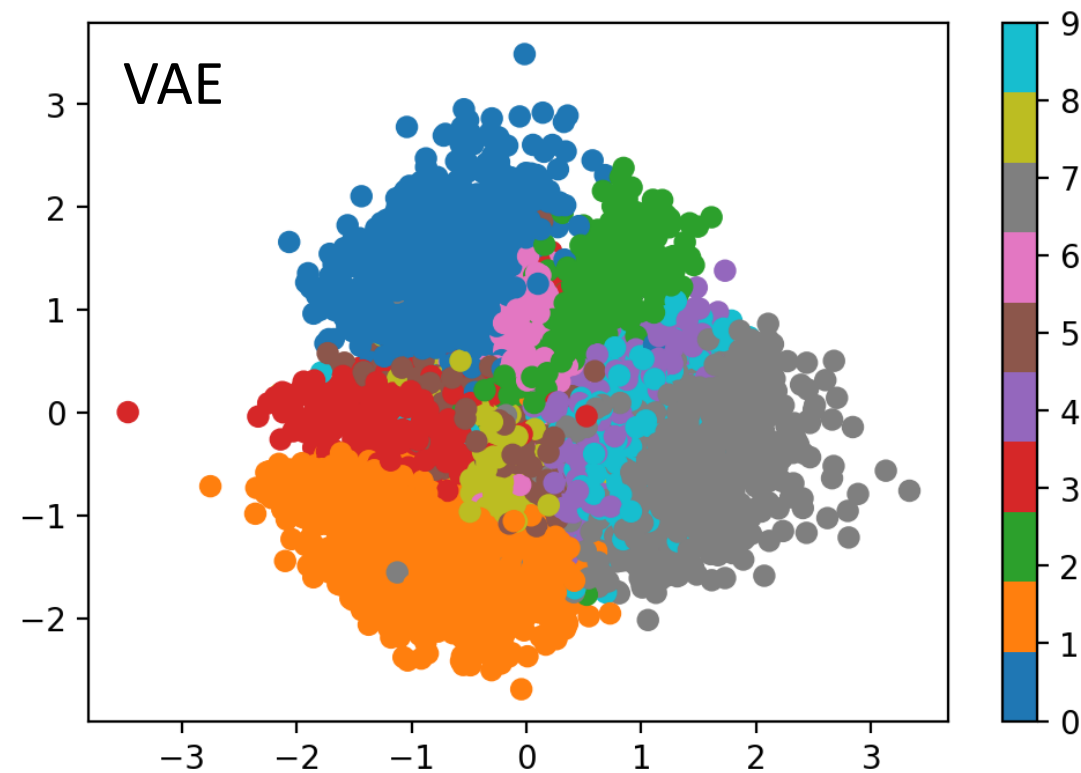
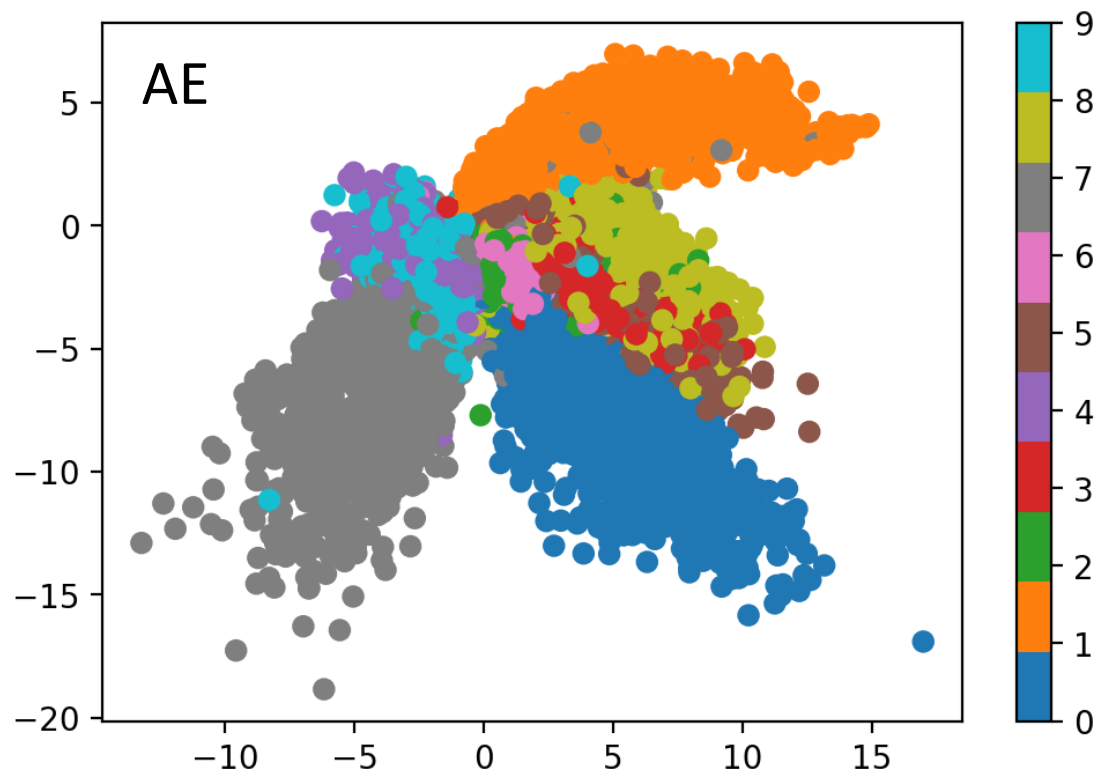


VAE

- Objective
 - Compress \mathbf{x} to \mathbf{z} which follows $P(\mathbf{Z} | \mathbf{X})$
 - Decompress \mathbf{z} to reconstruct \mathbf{x}
- This allows us to
 - Map data distribution $P(\mathbf{X})$ to a probability distribution $P(\mathbf{Z})$
 - Sample \mathbf{z} from $P(\mathbf{Z})$, which can be converted to \mathbf{x}



Latent Space: AE vs VAE



VAE Loss

- Loss consists of
 - Reconstruction loss
 - Regularization term
 - Force z to follow Gaussian distribution
- Loss for a single sample x_i

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\underbrace{\log p_\phi(x_i | z)}_{\substack{\text{Decoder network} \\ \text{parametrized by } \phi}}] + \text{KL}(\underbrace{q_\theta(z | x_i)}_{\substack{\text{Encoder network} \\ \text{parametrized by } \theta}} || \underbrace{p(z)}_{\substack{\text{Usually} \\ \mathcal{N}(0, 1)}})$$

- In autoencoders, negative log likelihood is reconstruction loss

VAE: Probability Point of View

- We want to infer $P(\mathbf{Z} \mid \mathbf{X})$
 - The posterior distribution given data \mathbf{X}

VAE Posterior Distribution

- Since $P(\mathbf{Z} | \mathbf{X})$ is intractable, use variational inference

$$\text{KL}(q_\lambda(z | x) || p(z | x)) = \mathbf{E}_q[\log q_\lambda(z | x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$



$$D_{\text{KL}}(Q || P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X}) \quad (\text{Original VI})$$

- Use $Q_\lambda(\mathbf{Z} | \mathbf{X})$ instead of $Q(\mathbf{Z})$
 - Because \mathbf{Z} is determined by \mathbf{X}
 - This will be our encoder
 - λ is (μ, σ) for Gaussian $Q \rightarrow$ Encoder generates (μ, σ)

VAE Posterior Distribution

- Since $P(\mathbf{Z} | \mathbf{X})$ is intractable, use variational inference

$$\mathbb{KL}(q_\lambda(z | x) || p(z | x)) = \mathbf{E}_q[\log q_\lambda(z | x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$



$$D_{\text{KL}}(Q || P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X}) \quad (\text{Original VI})$$

- Use $Q_\lambda(\mathbf{Z} | \mathbf{X})$ instead of $Q(\mathbf{Z})$
 - Because \mathbf{Z} is determined by \mathbf{X}
 - This will be our encoder
 - λ is (μ, σ) for Gaussian $Q \rightarrow$ Encoder generates (μ, σ)
- We want to learn an encoder $q_\lambda^*(z | x) = \arg \min_\lambda \mathbb{KL}(q_\lambda(z | x) || p(z | x))$

VAE Posterior Distribution

- Since $P(\mathbf{Z} \mid \mathbf{X})$ is intractable, use variational inference

$$\mathbb{KL}(q_\lambda(z \mid x) \parallel p(z \mid x)) = \mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$



$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X}) \quad (\text{Original VI})$$

- The new ELBO term

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$



$$\mathcal{L}(Q) = - \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] \quad (\text{Original ELBO})$$

VAE Posterior Distribution

- Since $P(\mathbf{Z} \mid \mathbf{X})$ is intractable, use variational inference

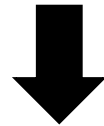
$$\mathbb{KL}(q_\lambda(z \mid x) \parallel p(z \mid x)) = \mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$



$$D_{\text{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Z}} [\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})] + \log P(\mathbf{X})$$

- The new ELBO term

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$



ELBO term for each sample x_i

$$ELBO_i(\lambda) = \mathbb{E}_{q_\lambda(z \mid x_i)}[\log p(x_i \mid z)] - \mathbb{KL}(q_\lambda(z \mid x_i) \parallel p(z))$$

ELBO Term

- ELBO term for each x_i

$$ELBO_i(\lambda) = \mathbb{E}q_\lambda(z \mid x_i)[\log p(x_i \mid z)] - \mathbb{KL}(q_\lambda(z \mid x_i) \parallel p(z))$$

- Parametrize the components
 - Approximate posterior q with θ
 - Likelihood p with ϕ
- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \parallel p(z))$$

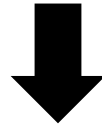
ELBO Term

- ELBO term for each x_i

$$ELBO_i(\lambda) = \mathbb{E}q_\lambda(z \mid x_i)[\log p(x_i \mid z)] - \mathbb{KL}(q_\lambda(z \mid x_i) \parallel p(z))$$

- Parametrize the components
 - Approximate posterior q with θ
 - Likelihood p with ϕ
- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \parallel p(z))$$



Remind you of something??

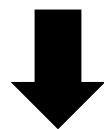
ELBO Term

- ELBO term for each x_i

$$ELBO_i(\lambda) = \mathbb{E}_{q_\lambda(z | x_i)}[\log p(x_i | z)] - \mathbb{KL}(q_\lambda(z | x_i) || p(z))$$

- Parametrize the components
 - Approximate posterior q with θ
 - Likelihood p with ϕ
- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}_{q_\theta(z | x_i)}[\log p_\phi(x_i | z)] - \mathbb{KL}(q_\theta(z | x_i) || p(z))$$



$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i | z)] + \mathbb{KL}(q_\theta(z | x_i) || p(z))$$

ELBO Term

- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}_{q_\theta(z | x_i)}[\log p_\phi(x_i | z)] - \mathbb{KL}(q_\theta(z | x_i) || p(z))$$



VAE ELBO is the negative loss!!

$$ELBO_i(\theta, \phi) = -l_i(\theta, \phi)$$

- Maximizing the ELBO is equivalent to minimizing the loss!

ELBO Term

- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}_{q_\theta(z | x_i)}[\log p_\phi(x_i | z)] - \mathbb{KL}(q_\theta(z | x_i) || p(z))$$

- Maximizing the ELBO is equivalent to minimizing the loss!
- ELBO can be maximized by both
 - Updating the encoder network (i.e. learning θ)
 - Updating the decoder network (i.e. learning ϕ)

ELBO Term

- The parametrized ELBO term

$$ELBO_i(\theta, \phi) = \mathbb{E}_{q_\theta(z | x_i)}[\log p_\phi(x_i | z)] - \mathbb{KL}(q_\theta(z | x_i) || p(z))$$

- Maximizing the ELBO is equivalent to minimizing the loss!
- ELBO can be maximized by both
 - Updating the encoder network (i.e. learning θ)
 - Updating the decoder network (i.e. learning ϕ)
 - In practice, update both params together.

Training VAE

Training VAE

- Loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \text{KL}(q_\theta(z | x_i) || p(z))$$

- How do you train your network?

Training VAE

- Loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \text{KL}(q_\theta(z | x_i) || p(z))$$

- Using a random sample x_i as an example:

- Push x_i into the encoder $q_\theta(z | x_i)$
- Obtain (μ_i, σ_i)
- Sample (many) z_i from $\mathcal{N}(\mu_i, \sigma_i)$ ➔ Monte Carlo Estimation of $\mathbb{E}_z[f(z)]$
- Insert z_i into the decoder $p_\phi(x_i | z)$
- Obtain x'_i
- Calculate squared error $\|x'_i - x_i\|^2$ and $D_{KL}(\mathcal{N}(\mu_i, \sigma_i) || \mathcal{N}(0,1))$
- Backpropagate
- Update θ and ϕ

Training VAE

- Loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \mathbb{KL}(q_\theta(z | x_i) || p(z))$$

- Using a random sample x_i as an example:

- Push x_i into the encoder $q_\theta(z | x_i)$
- Obtain (μ_i, σ_i)
- Sample (many) z_i from $\mathcal{N}(\mu_i, \sigma_i)$ ➔ Monte Carlo Estimation of $\mathbb{E}_z[f(z)]$
- Insert z_i into the decoder $p_\phi(x_i | z)$
- Obtain x'_i
- Calculate squared error $\|x'_i - x_i\|^2$ and $D_{KL}(\mathcal{N}(\mu_i, \sigma_i) || \mathcal{N}(0,1))$
- Backpropagate
- Update θ and ϕ

This has an analytical expression,

$$\frac{1}{2}(\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1)$$

which can be used to update θ

Training VAE

- Loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)} [\log p_\phi(x_i | z)] + \mathbb{KL}(q_\theta(z | x_i) || p(z))$$

- Using a random sample x_i as an example:

- Push x_i into the encoder $q_\theta(z | x_i)$
- Obtain (μ_i, σ_i)
- Sample (many) z_i from $\mathcal{N}(\mu_i, \sigma_i)$ ➔ Monte Carlo Estimation of $\mathbb{E}_z[f(z)]$
- Insert z_i into the decoder $p_\phi(x_i | z)$
- Obtain x'_i
- Calculate squared error $\|x'_i - x_i\|^2$ and $D_{KL}(\mathcal{N}(\mu_i, \sigma_i) || \mathcal{N}(0,1))$
- Backpropagate
- Update θ and ϕ

How about this?

Stochasticity in the Network

- $\|x'_i - x_i\|^2$ involves a random sample z_i
 - x'_i is decoded based on $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$
- One can backpropagate through the decoder.
 - Consider z_i as an input to the decoder $p_\phi(x_i | z)$
 - We can obtain $\frac{\partial \mathcal{L}}{\partial \phi}$ via chain-rule ($\mathcal{L} = \text{MSE Loss}$)

Stochasticity in the Network

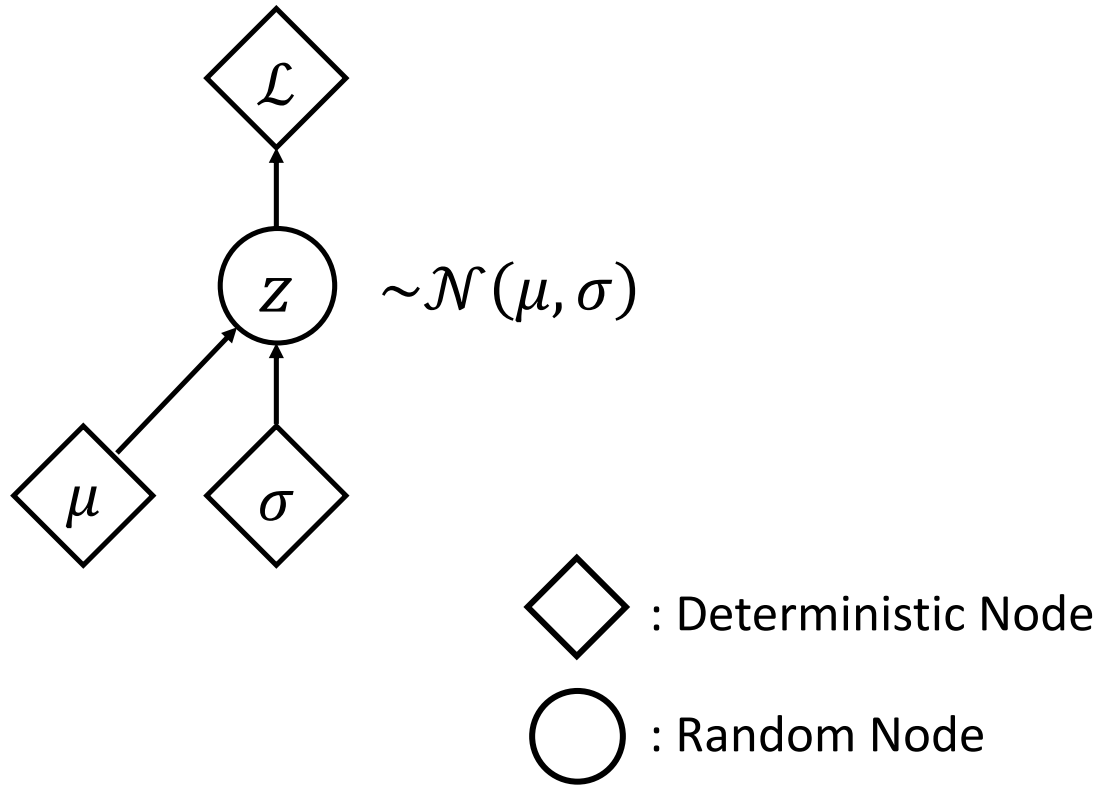
- $\|x'_i - x_i\|^2$ involves a random sample z_i
 - x'_i is decoded based on $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$
- One can backpropagate through the decoder.
 - Consider z_i as an input to the decoder $p_\phi(x_i | z)$
 - We can obtain $\frac{\partial \mathcal{L}}{\partial \phi}$ via chain-rule (\mathcal{L} = MSE Loss)
- Can you backpropagate through the encoder to obtain $\frac{\partial \mathcal{L}}{\partial \theta}$?

Stochasticity in the Network

- $\|x'_i - x_i\|^2$ involves a random sample z_i
 - x'_i is decoded based on $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$
- One can backpropagate through the decoder.
 - Consider z_i as an input to the decoder $p_\phi(x_i | z)$
 - We can obtain $\frac{\partial \mathcal{L}}{\partial \phi}$ via chain-rule (\mathcal{L} = MSE Loss)
- Can you backpropagate through the encoder to obtain $\frac{\partial \mathcal{L}}{\partial \theta}$?
 - How would you get past the randomly sampled z_i ?

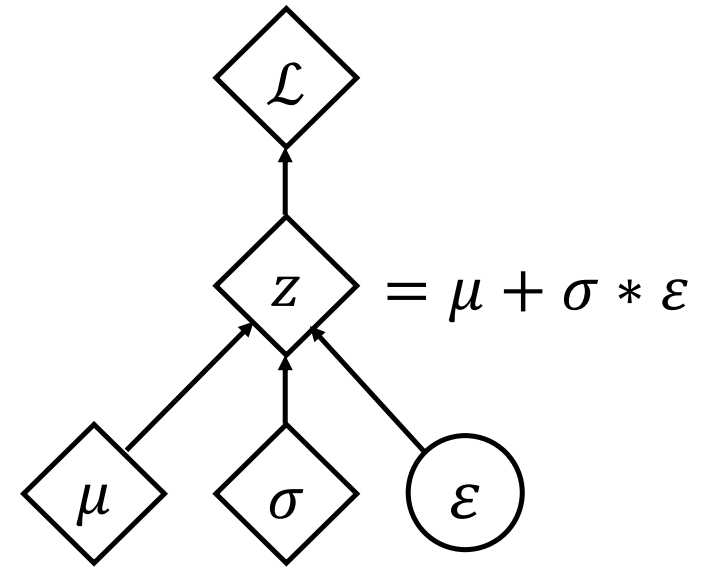
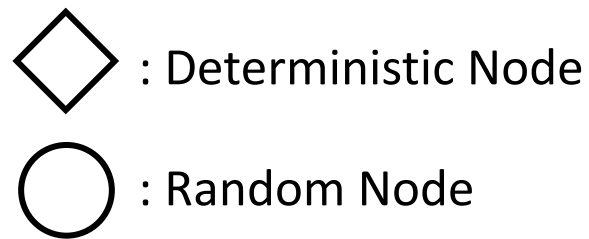
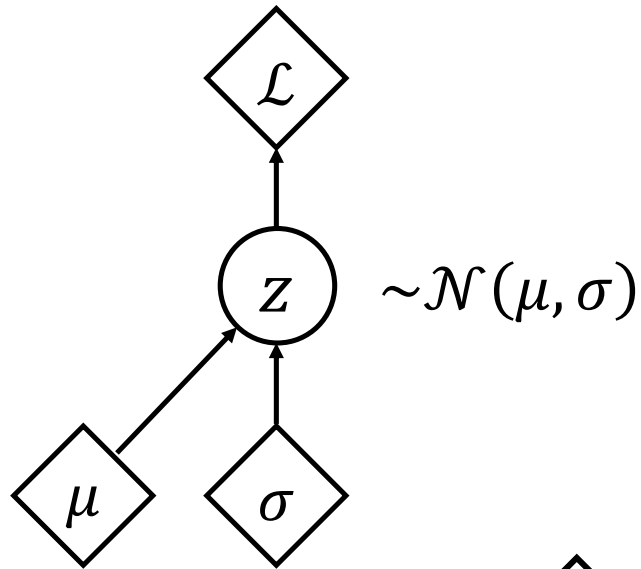
Stochasticity in the Network

- Can you backpropagate through the encoder to obtain $\frac{\partial \mathcal{L}}{\partial \theta}$?
 - How would you get past the randomly sampled z_i ?



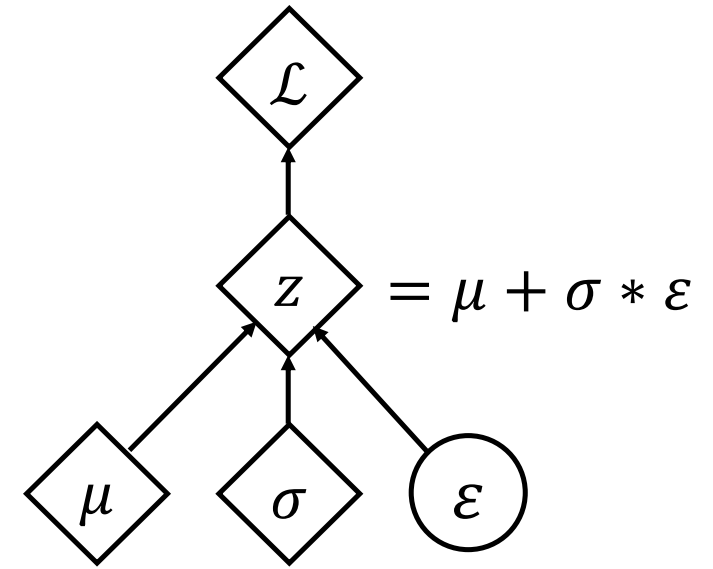
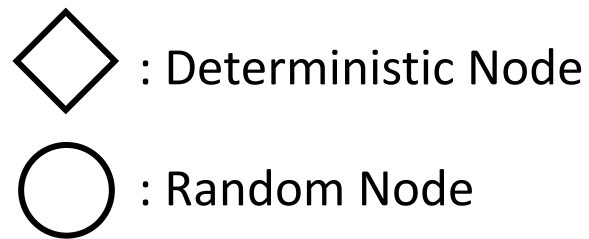
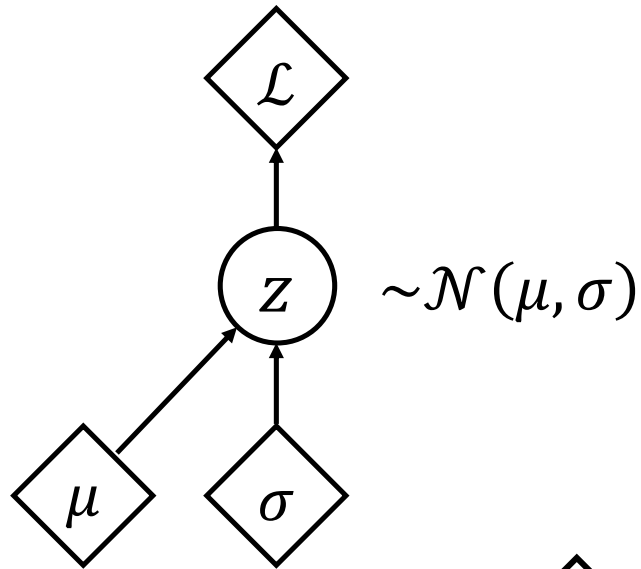
Reparametrization Trick

- Can you backpropagate through the encoder to obtain $\frac{\partial \mathcal{L}}{\partial \theta}$?
 - How would you get past the randomly sampled z_i ?



Reparametrization Trick

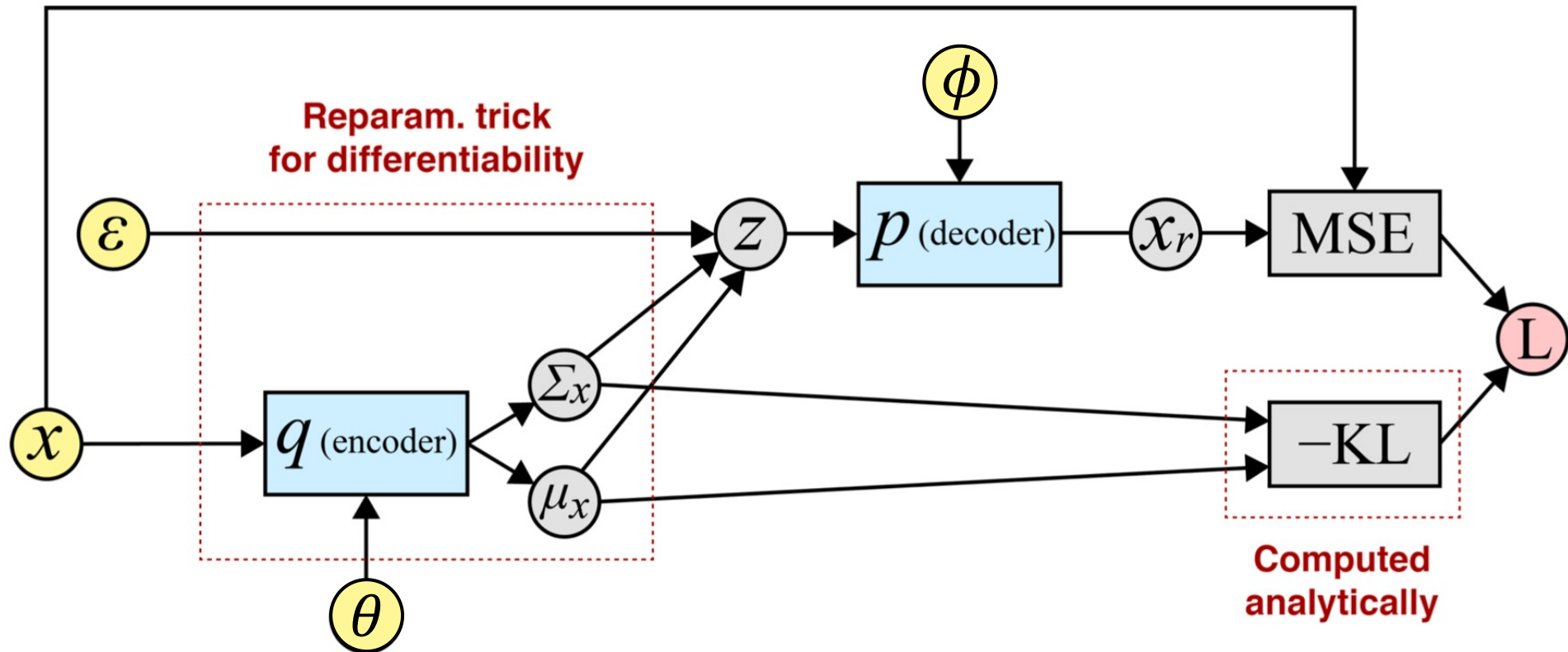
- Can you backpropagate through the encoder to obtain $\frac{\partial \mathcal{L}}{\partial \theta}$?
 - How would you get past the randomly sampled z_i ?



Reparametrization Trick:
Randomness is removed from the backprop path.

The Big Picture

- The entire flow chart



Training Process

- Training VAE on MNIST images
 - Plotting 2-dimensional z for each encoded image as training proceeds

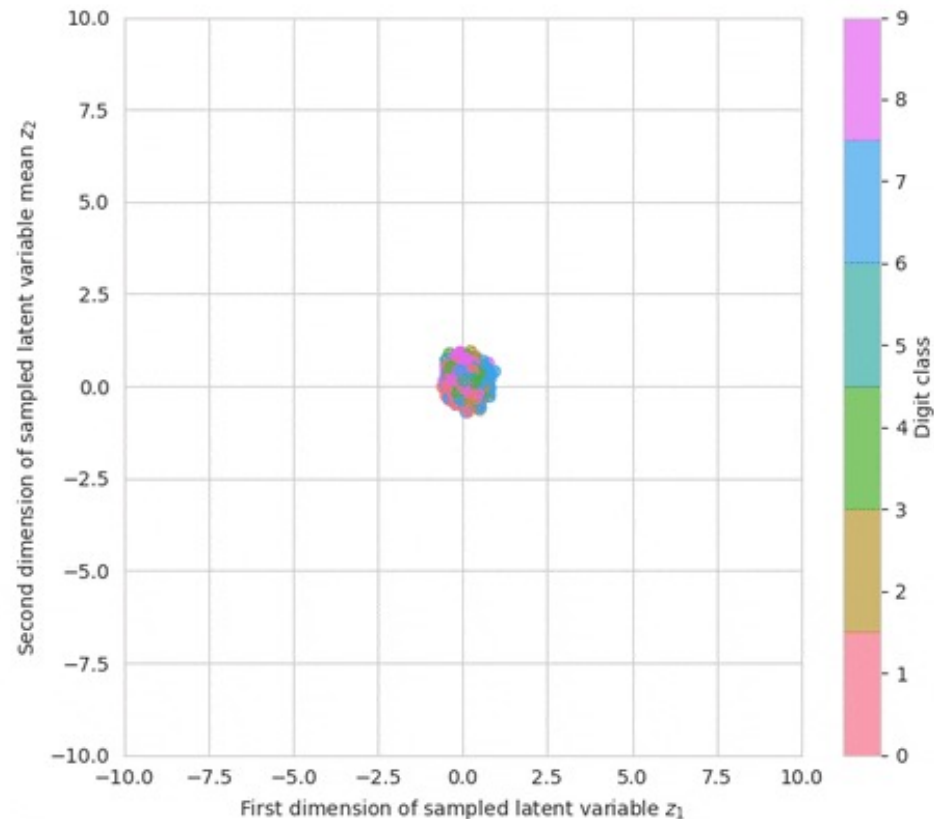
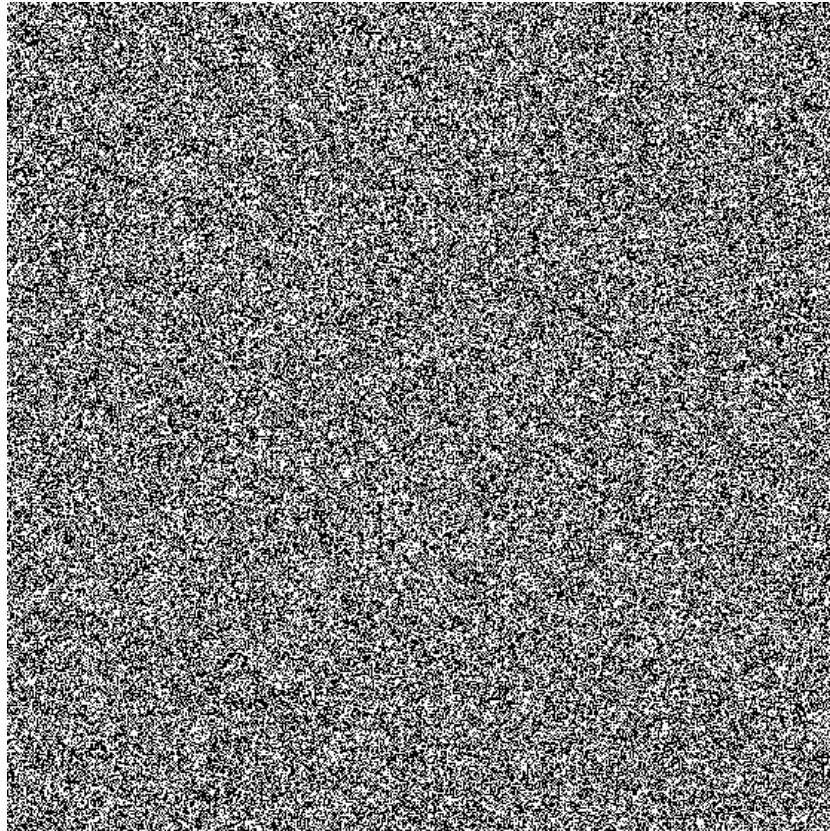


Image Generation

- Sample z from 2-dimensional space then decode.
 - Range $[-3, 3]$ for both axes.



Conclusion

- VAE is an elegant solution for approximating $P(\mathbf{Z} | \mathbf{X})$
 - It is still an approximation though
 - The two mapping functions are disconnected
 - Encoder and decoder use separate parameters
- Normalizing Flow
 - Use invertible functions to map between $p(\mathbf{X})$ and $p(\mathbf{Z})$
- β -VAE, TC-VAE
 - Disentangling the latent space
 - We want each \mathbf{z} dimension to have an interpretable role.
 - Rotation of the image, skewedness of the image, color of the image, etc.
- You can use VAE to generate sequences or graphs.

AI504: Programming for Artificial Intelligence

Week 5: Variational Autoencoder

Edward Choi

Grad School of AI

edwardchoi@kaist.ac.kr