# Deep Learning

## 19 Variational Autoencoder

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2023

# Autoencdoers

1. Designed to reproduce input, especially reproduce the input from a learned encoding

# Autoencdoers



1. Designed to reproduce input, especially reproduce the input from a learned encoding
2. We attempted to project the data into the latent space and model it via a probability distribution

# Autoencdoers

1. Designed to reproduce input, especially reproduce the input from a learned encoding
2. We attempted to project the data into the latent space and model it via a probability distribution
3. This wasn't satisfying

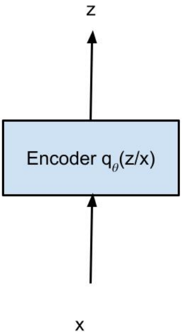
# Variational Autoencoders



1. 'Regularized' autoencoder to enforce latent space 'organization'

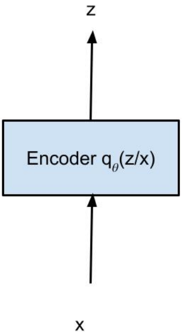
# Variational Autoencoders

1. Key idea is to make both Encoder and Decoder stochastic
   - instead of encoding an i/p as a single point, we encode it as a distribution over the latent space



# Variational Autoencoders

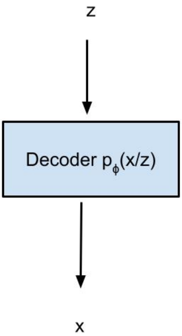
1. Key idea is to make both Encoder and Decoder stochastic
   - instead of encoding an i/p as a single point, we encode it as a distribution over the latent space
2. Latent variable z is drawn from a probability distribution for the given input x



# Variational Autoencoders



1. Then, the reconstruction is chosen probabilistically from the sampled z



# VAE Encoder

1. Takes i/p and returns the parameters of a probability density (e.g. Gaussian, mean and covariance matrix)

# VAE Encoder



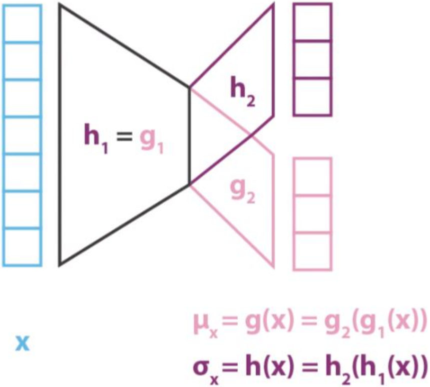
1. Takes i/p and returns the parameters of a probability density (e.g. Gaussian, mean and covariance matrix)
2. We can sample this to get random values of the latent variable z

# VAE Encoder



1. Takes i/p and returns the parameters of a probability density (e.g. Gaussian, mean and covariance matrix)
2. We can sample this to get random values of the latent variable z
3. NN implementation of the encoder gives (for every input x) a vector mean and a diagonal covariance

# VAE Encoder



# VAE Decoder



1. Decoder takes the latent vector z and returns the parameters for a distribution

# VAE Decoder

1. Decoder takes the latent vector z and returns the parameters for a distribution
2. $p_\phi(x/z)$ gives mean and variance for each pixel in the output

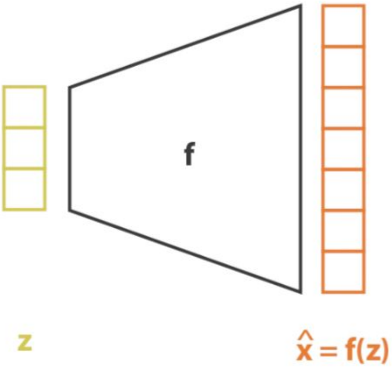
# VAE Decoder



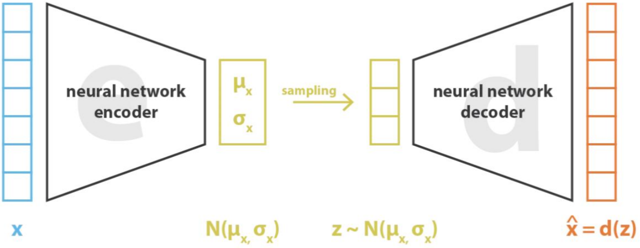
1. Decoder takes the latent vector z and returns the parameters for a distribution
2. $p_\phi(x/z)$ gives mean and variance for each pixel in the output
3. Reconstruction of x is via sampling (with some assumptions, the data sample can be output)

# VAE Decoder





# VAE Forward pass



# VAE loss function

1. Loss for AE: $l_2$ distance between the input and its reconstruction

# VAE loss function



1. Loss for AE: $l_2$ distance between the input and its reconstruction
2. In case of VAE: we need to learn parameters of two probability distributions

# VAE loss function



1. Loss for AE: $l_2$ distance between the input and its reconstruction
2. In case of VAE: we need to learn parameters of two probability distributions
3. For each input $x_i$ we maximize expected value of returning $x_i$ (or, minimize the NLL)

$$-\mathbb{E}_{z \sim q_\theta(z/x_i)}[log\, p_\phi(x_i/z)]$$

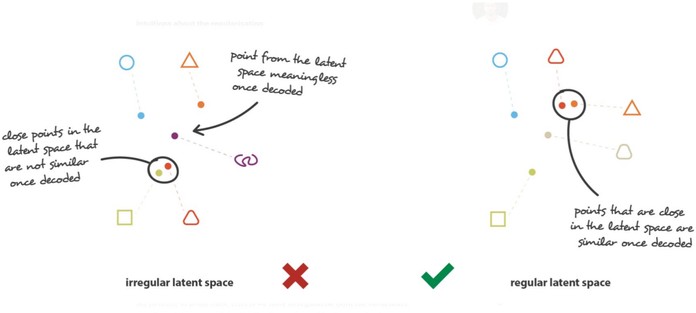
# VAE loss function

$$-\mathbb{E}_{z\sim q_\theta(z/x_i)}[log\, p_\phi(x_i/z)]$$

1. Problem: Input images may be memorized in the latent space
    - $\rightarrow$ similar inputs may get different representations in z space
    - $\rightarrow$ close points in the latent space should not give two completely different contents once decoded

# VAE loss function





# VAE loss function



$$-\mathbb{E}_{z \sim q_\theta(z/x_i)}[log\, p_\phi(x_i/z)]$$

1. Continuity and Completeness: We prefer continuous latent representations to give meaningful parameterization (e.g. smooth transition between i/ps)

# VAE loss function



$$-\mathbb{E}_{z \sim q_\theta(z/x_i)}[log\, p_\phi(x_i/z)]$$

1. Continuity and Completeness: We prefer continuous latent representations to give meaningful parameterization (e.g. smooth transition between i/ps)
2. Solution: Force $q_\theta(z/x_i)$ to be close to a standard distribution (e.g. Gaussian)

# VAE loss function



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

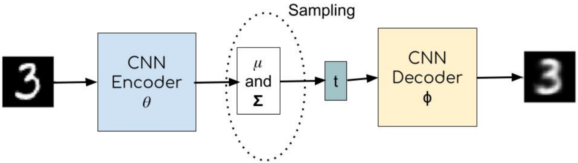
1. First term promotes recovery, sencond term keeps encoding continuous (beats memorization)

# VAE loss function



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

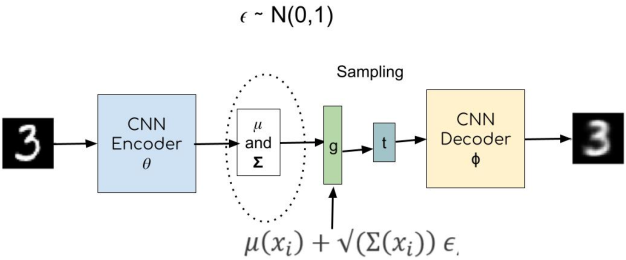
1. Problem: Differentiating over $\theta$ and $\phi$



# VAE loss function

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

1. Reparameterization: Draw samples from N(0,1) → doesn't depend on parameters



# Generation with VAE



- Sample z from the prior p(z)

# Generation with VAE

- Sample z from the prior p(z)
- Run z through the decoder ($\phi$) $\rightarrow$ distribution over data

# Generation with VAE

- Sample z from the prior p(z)
- Run z through the decoder ($\phi$) $\rightarrow$ distribution over data
- Sample from that distribution to generate the sample x

# Generation with VAE

Figure credits: Wojceich

# Generation with VAE



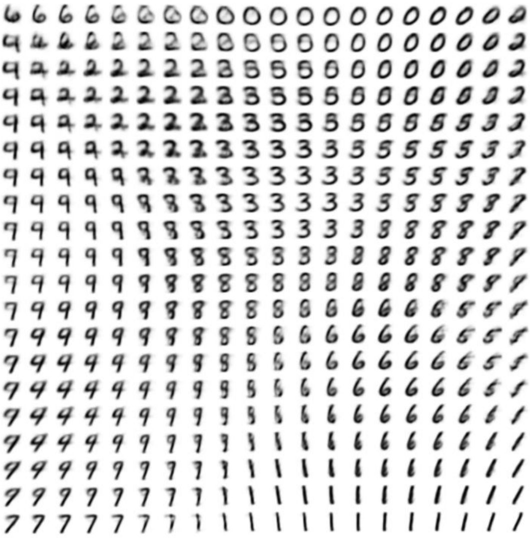


Figure credits: Kingma et al.

# Edit/Manipulate samples with VAE