From Autoencoder to Variational Autoencoder

From Autoencoder to Variational Autoencoder

Feature Representation

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder

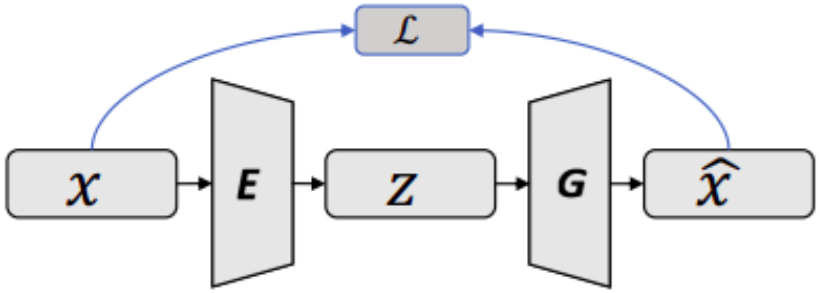
Distribution Representation

- Stacked Autoencoder
- Variational Autoencoder (VAE)

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)

Vanilla Autoencoder

- **What is it?**

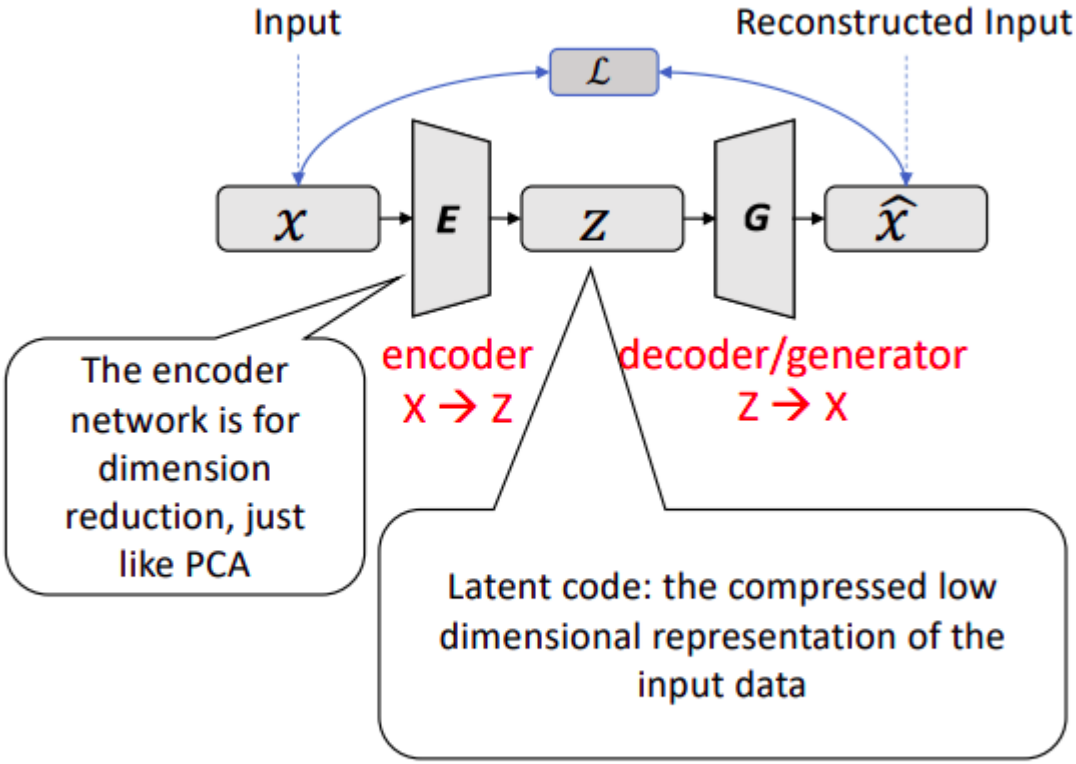


Reconstruct high-dimensional data using a neural network model with a narrow bottleneck layer.

The bottleneck layer captures the compressed latent coding, so the nice by-product is dimension reduction.

The low-dimensional representation can be used as the representation of the data in various applications, e.g., image retrieval, data compression …

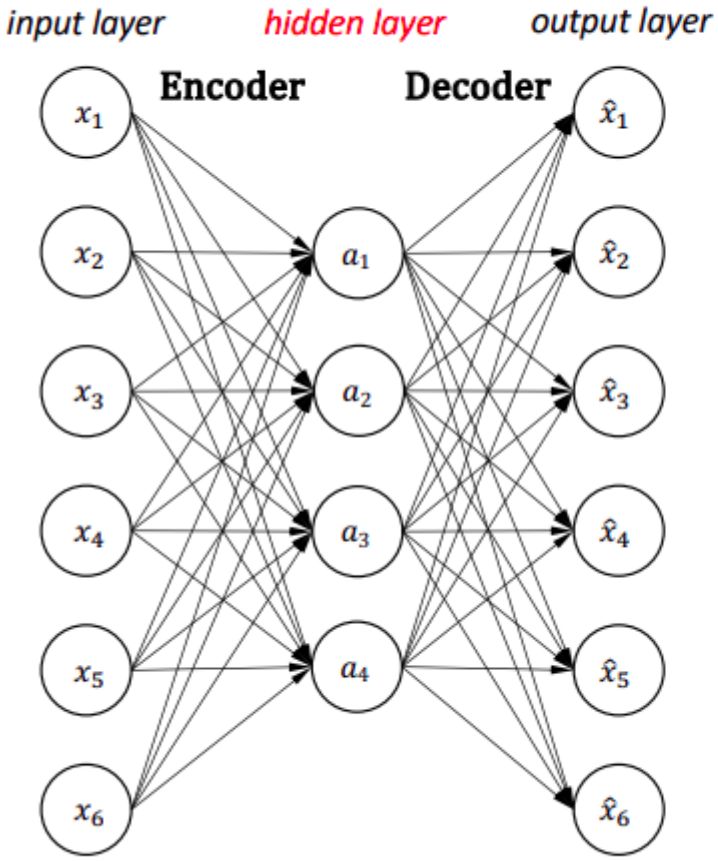
- **How it works?**



Ideally the input and reconstruction are identical

Vanilla Autoencoder

- **Training**



- The hidden units are usually less than the number of inputs
- Dimension reduction --- Representation learning

The distance between two data can be measure by Mean Squared Error (MSE):

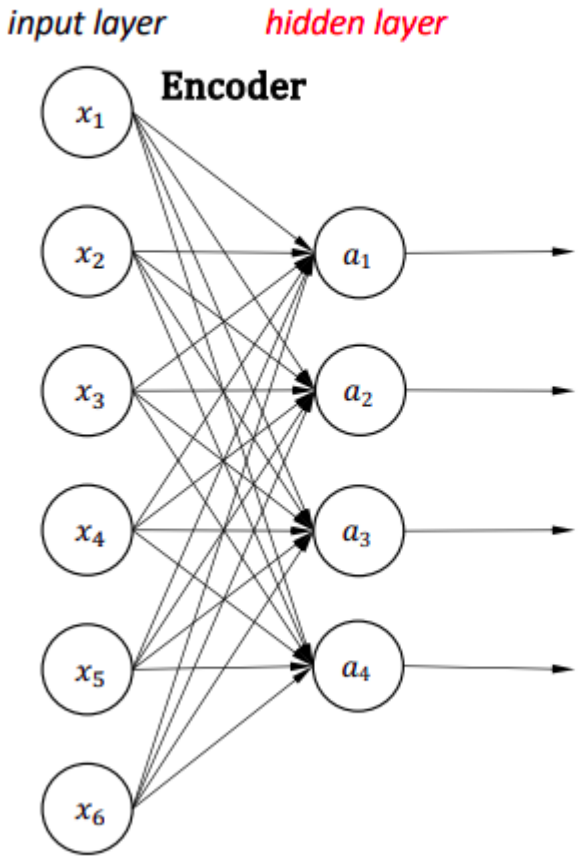
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(x^i - G(E(x^i))^2$$

where n is the number of variables

- It is trying to learn an approximation to the identity function so that the input is "compress" to the "compressed" features, discovering interesting structure about the data.

Vanilla Autoencoder

- **Testing/Inferencing**

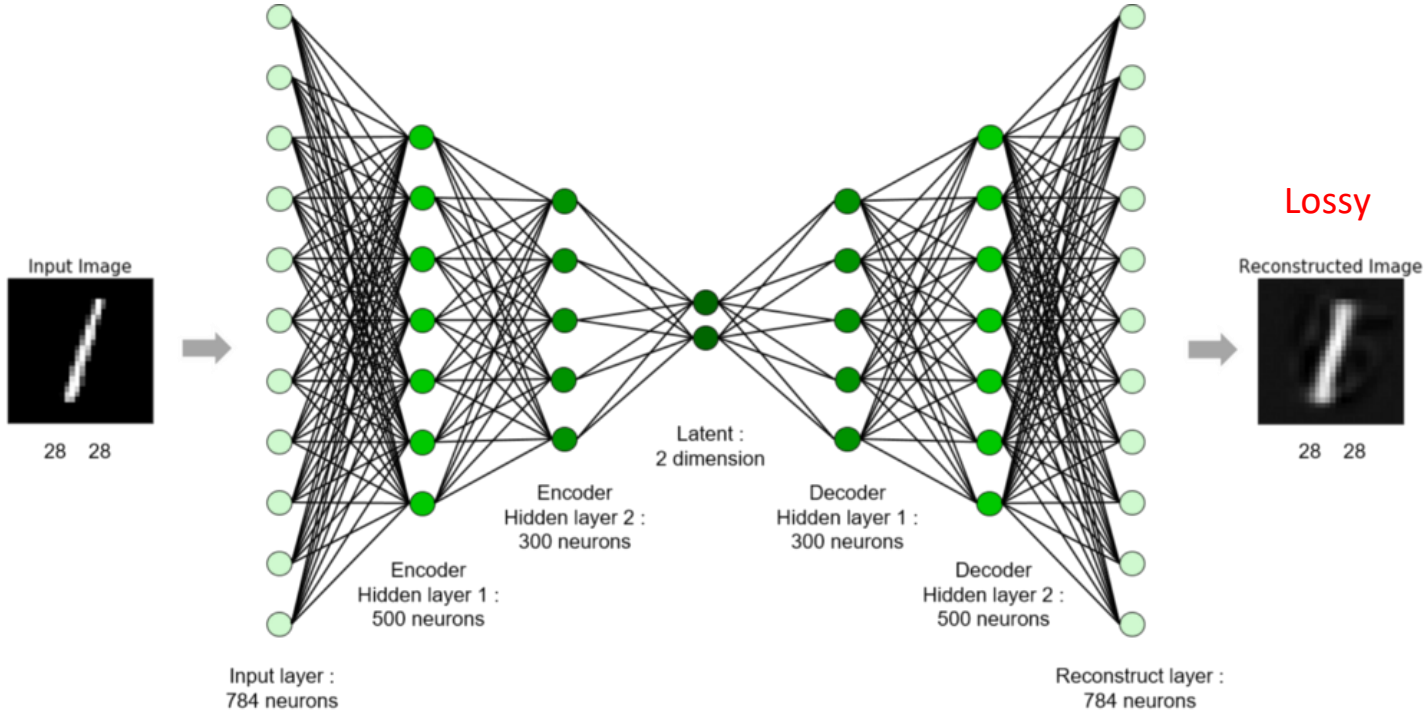


- Autoencoder is an unsupervised learning method if we considered the latent code as the "output".
- Autoencoder is also a self-supervised (self-taught) learning method which is a type of supervised learning where the training labels are determined by the input data.
- Word2Vec (from RNN lecture) is another unsupervised, self-taught learning example.

Autoencoder for MNIST dataset (28×28×1, 784 pixels)

Vanilla Autoencoder

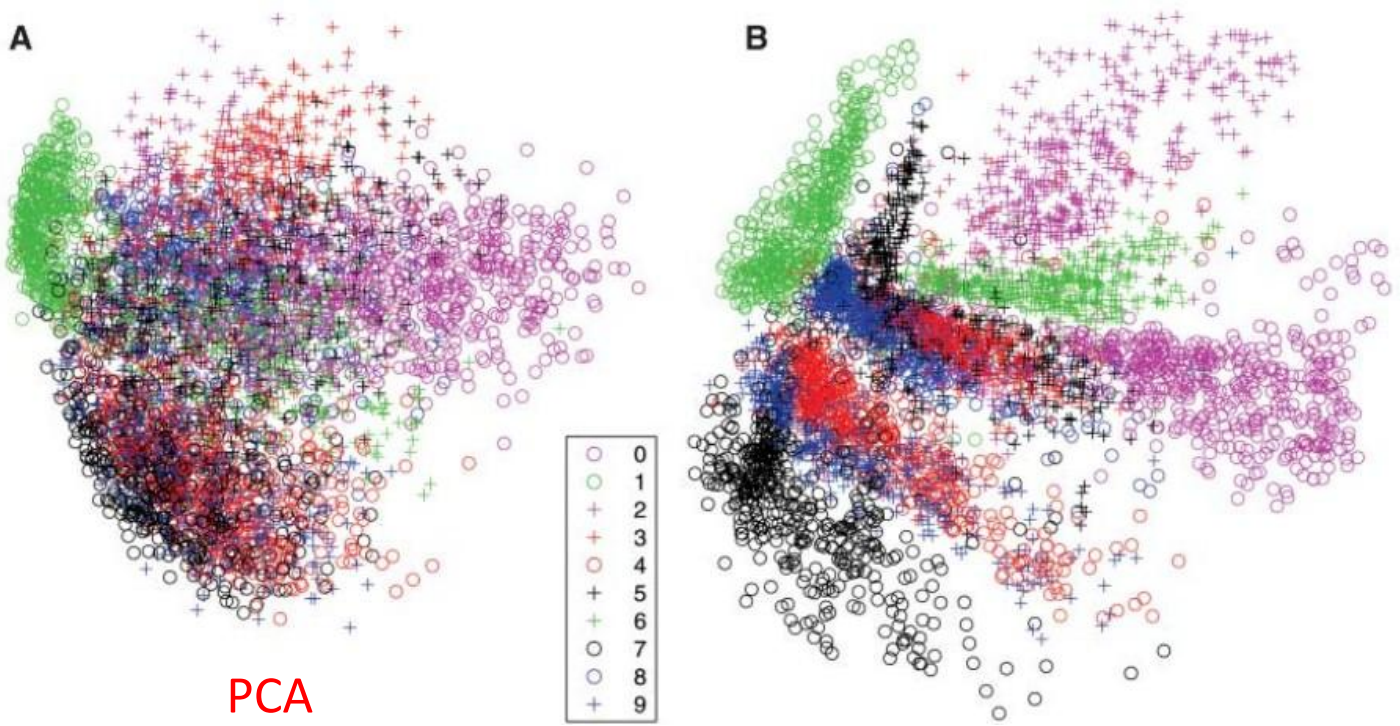
- **Example:**
  - Compress MNIST (28x28x1) to the latent code with only 2 variables



Vanilla Autoencoder

- **Power of Latent Representation**
  - t-SNE visualization on MNIST: PCA vs. Autoencoder

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



PCA

Autoencoder (Winner)

2006 Science paper by Hinton and Salakhutdinov

Vanilla Autoencoder

- **Discussion**
  - Hidden layer is overcomplete if greater than the input layer

Vanilla Autoencoder

- **Discussion**
  - Hidden layer is overcomplete if greater than the input layer
    - No compression
    - No guarantee that the hidden units extract meaningful feature

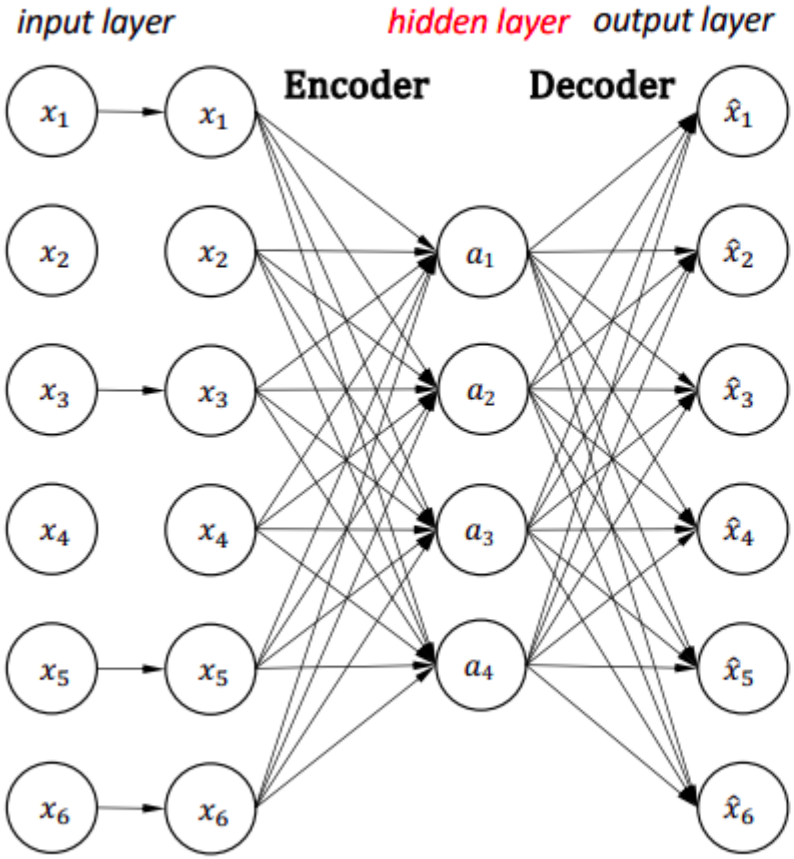
- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)

Denoising Autoencoder (DAE)

- **Why?**
  - **Avoid overfitting**
  - **Learn robust representations**

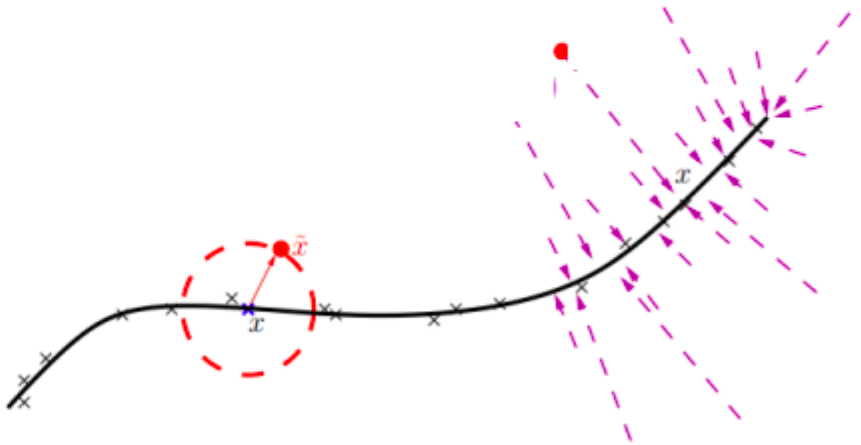
Denoising Autoencoder

- **Architecture**



Applying dropout between the input and the first hidden layer

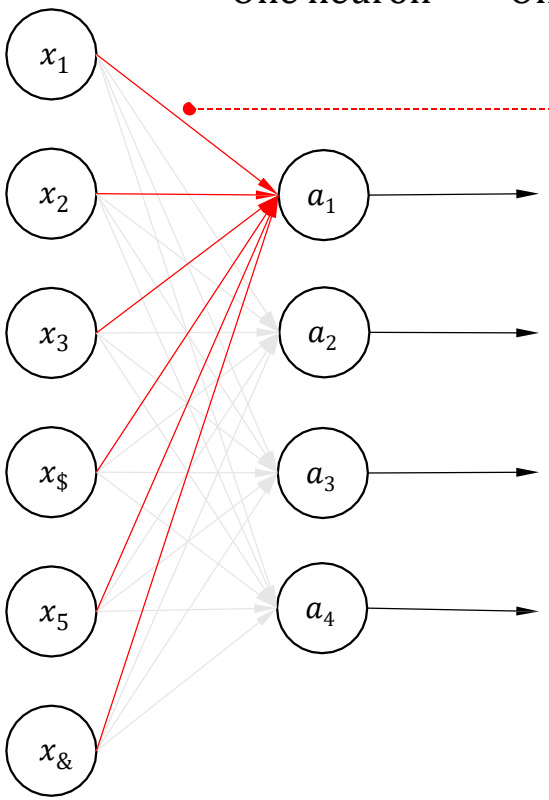
- Improve the robustness



Denoising Autoencoder

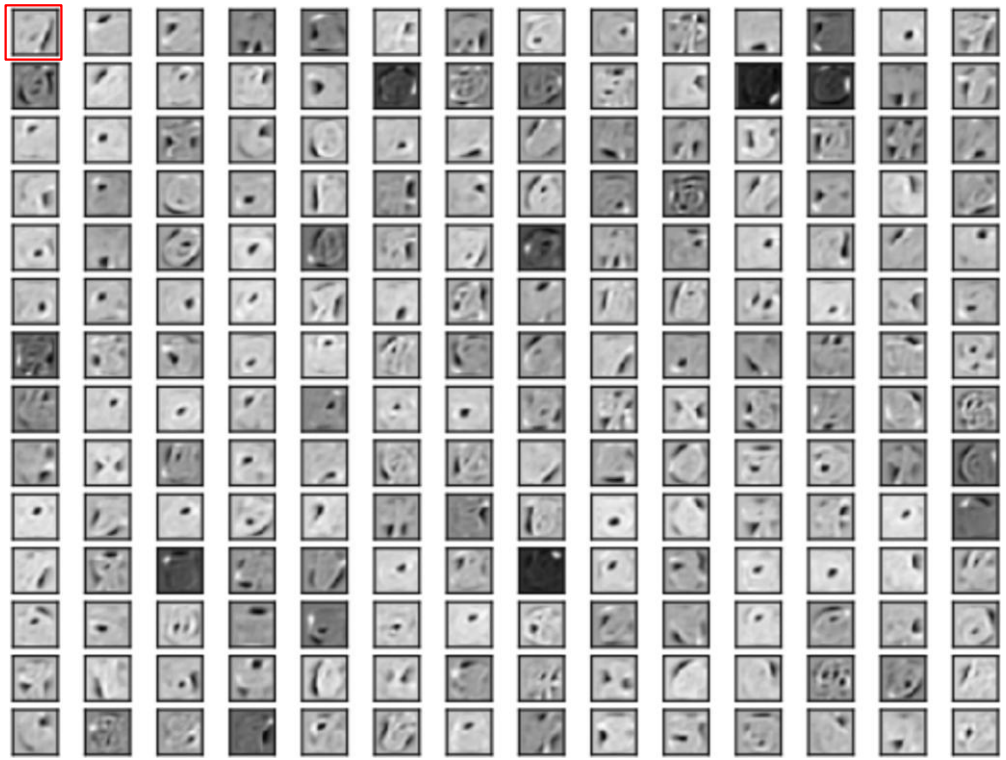
- **Feature Visualization**

One neuron == One feature extractor



reshape

Visualizing the learned features



Denoising Autoencoder

- **Denoising Autoencoder & Dropout**

Denoising autoencoder was proposed in 2008, 4 years before the dropout paper (Hinton, et al. 2012).

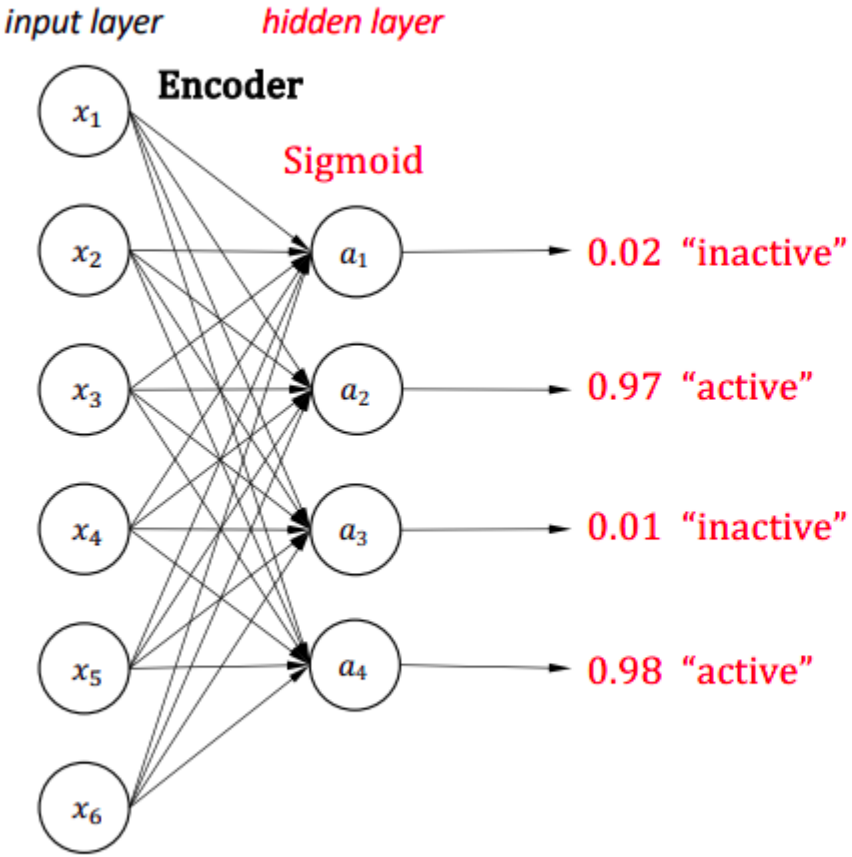
Denoising autoencoder can be seem as applying dropout between the input and the first layer.

Denoising autoencoder can be seem as one type of data augmentation on the input.

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)

Sparse Autoencoder

- Why?



- Even when the number of hidden units is large (perhaps even greater than the number of input pixels), we can still discover interesting structure, by imposing other constraints on the network.

- In particular, if we impose a "'sparsity"' constraint on the hidden units, then the autoencoder will still discover interesting structure in the data, even if the number of hidden units is large.

Sparse Autoencoder

- Recap: KL Divergence

$$KL\Big(p(x)\Big\|q(x)\Big) = \int p(x)\ln\frac{p(x)}{q(x)}dx = \mathbb{E}_{x\sim p(x)}\left[\ln\frac{p(x)}{q(x)}\right]$$

Smaller == Closer

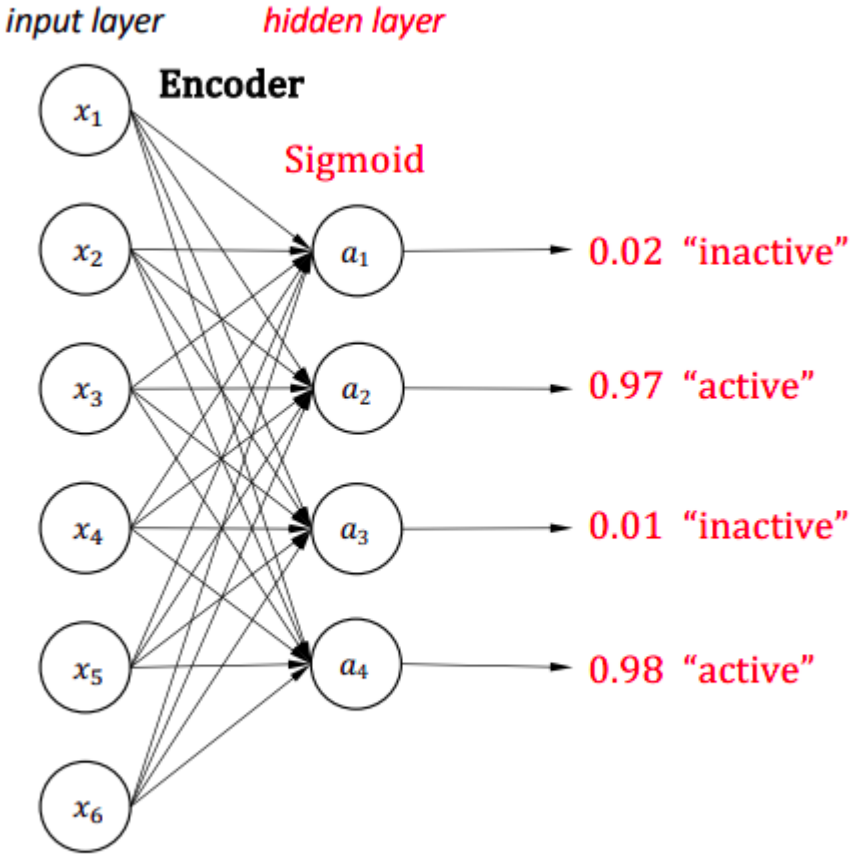
$$KL\Big(p(x)\Big\|q(x)\Big) = 0 \Leftrightarrow p(x) = q(x)$$

$$KL\Big(p(x)\Big\|q(x)\Big) = 0 \Leftrightarrow p(x) = q(x)$$

$$D_B\Big(p(x), q(x)\Big) = -\ln\int\sqrt{p(x)q(x)}dx$$

Sparse Autoencoder

- Sparsity Regularization



Given M data samples (batch size) and Sigmoid activation function, the active ratio of a neuron a_j:

$$\hat{\rho}_j = \frac{1}{M}\sum_{m=1}^{M} a_j$$

To make the output "sparse", we would like to enforce the following constraint, where ρ is a "sparsity parameter", such as 0.2 (20% of the neurons)

$$\hat{\rho}_j = \rho$$

The penalty term is as follow, where s is the number of activation outputs.

$$\mathcal{L}_\rho = \sum_{j=1}^{s} KL(\rho||\hat{\rho}_j)$$
$$= \sum_{j=1}^{s}(\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j})$$

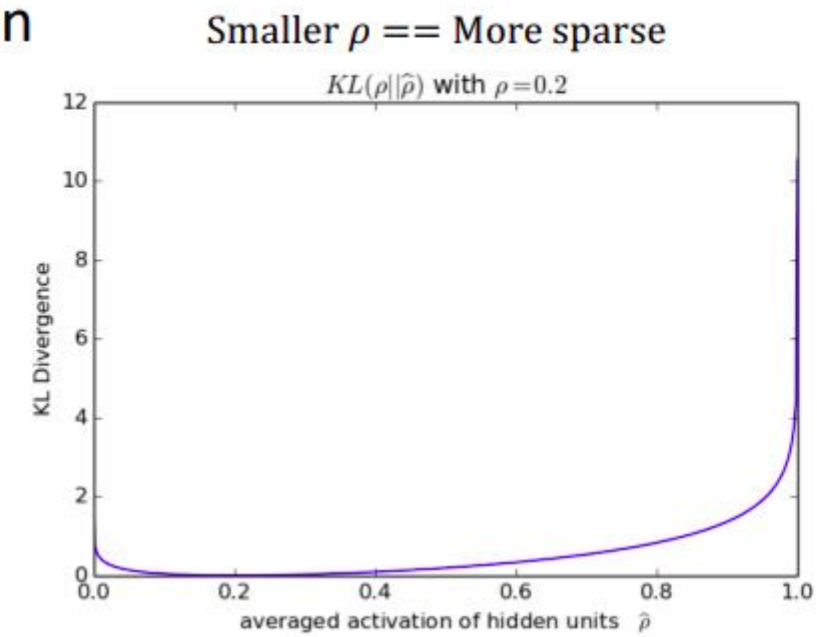
The total loss:

$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \lambda\mathcal{L}_\rho$$

The number of hidden units can be greater than the number of input variables.

Sparse Autoencoder

- Sparsity Regularization



Autoencoders for MNIST dataset

Sparse Autoencoder

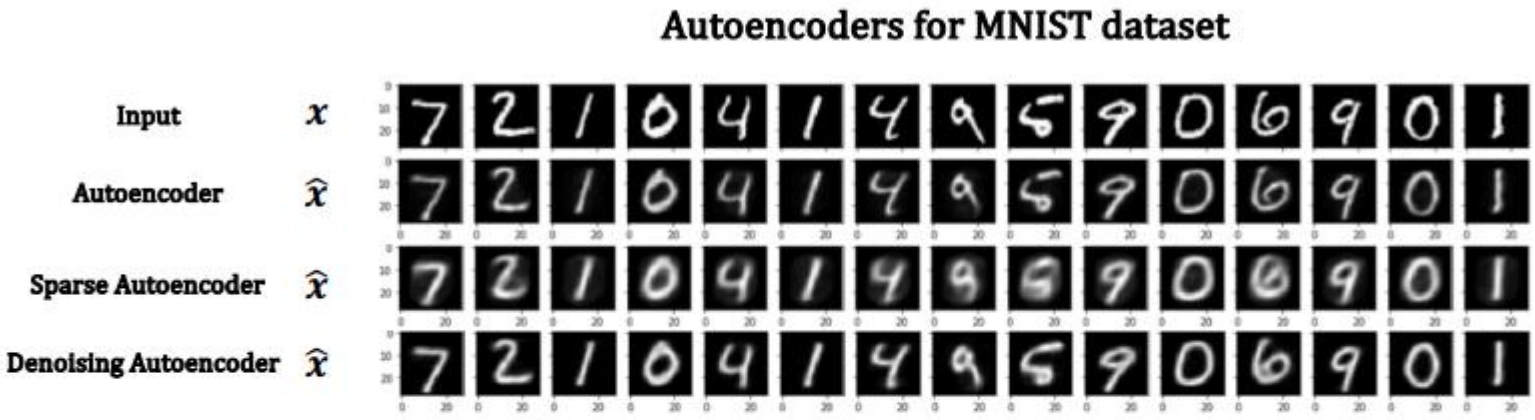
- **Different regularization loss**

Method	Hidden Activation	Reconstruction Activation	Loss Function
Method 1	Sigmoid	Sigmoid	$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \mathcal{L}_{\rho}$
Method 2	ReLU	Softplus	$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \|\boldsymbol{a}\|$

$\mathcal{L}_1$ on the hidden activation output

Sparse Autoencoder

- **Autoencoder vs. Denoising Autoencoder vs. Sparse Autoencoder**



- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)

Contractive Autoencoder

- **Why?**
  - Denoising Autoencoder and Sparse Autoencoder overcome the overcomplete problem via the input and hidden layers.
  - Could we add an explicit term in the loss to avoid uninteresting features?

    We wish the features that ONLY reflect variations observed in the training set

Contractive Autoencoder

- **How**
  - Penalize the representation being too sensitive to the input
  - Improve the robustness to small perturbations
  - Measure the sensitivity by the Frobenius norm of the Jacobian matrix of the encoder activations

Contractive Autoencoder

- **Recap: Jocobian Matrix**

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_1 + z_2 \\ 2z_1 \end{bmatrix} = f\left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right)$$

$$x = f(z) \quad z = f^{-1}(x) \qquad \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_2/2 \\ x_1 - x_2/2 \end{bmatrix} = f^{-1}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

input

$$J_f = \begin{bmatrix} \partial x_1/\partial z_1 & \partial x_1/\partial z_2 \\ \partial x_2/\partial z_1 & \partial x_2/\partial z_2 \end{bmatrix}$$ output

$$J_f = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$$

$$J_{f^{-1}} = \begin{bmatrix} \partial z_1/\partial x_1 & \partial z_1/\partial x_2 \\ \partial z_2/\partial x_1 & \partial z_2/\partial x_2 \end{bmatrix}$$

$$J_{f^{-1}} = \begin{bmatrix} 0 & 1/2 \\ 1 & -1/2 \end{bmatrix}$$

$$J_f J_{f^{-1}} = I$$

Contractive Autoencoder

- **Jocobian Matrix**

$$y = \begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \\ \vdots \\ f_n(x) \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3, \cdots x_n) \\ f_2(x_1, x_2, x_3, \cdots x_n) \\ f_3(x_1, x_2, x_3, \cdots x_n) \\ \vdots \\ f_n(x_1, x_2, x_3, \cdots x_n) \end{bmatrix}$$

$$J(x_1, x_2, x_3, \cdots x_n) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$y(x) \approx y(p) + J \bullet (x - p)$$

Contractive Autoencoder

- **New Loss**

$$\|J_f(x)\|_F^2 = \sum_{ij}\left(\frac{\partial h_j(x)}{\partial x_i}\right)^2$$

$$\mathcal{J}_{CAE}(\theta) = \sum_{x \in D_n}\left(L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2\right)$$

reconstruction new regularization

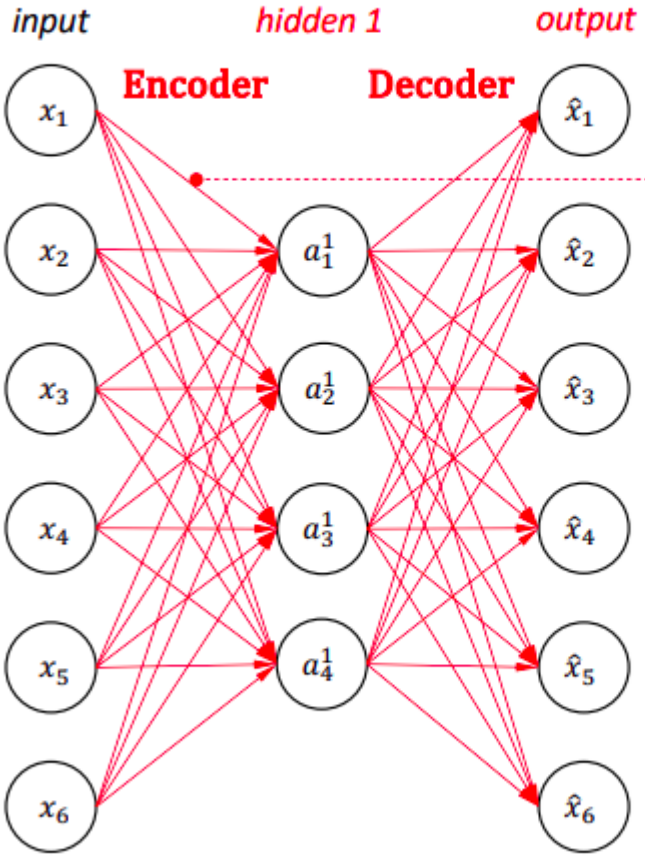
Contractive Autoencoder

- **vs. Denoising Autoencoder**
  - Advantages
    - CAE can better model the distribution of raw data
  - Disadvantages
    - DAE is easier to implement
    - CAE needs second-order optimization (conjugate gradient, LBFGS)

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)

Stacked Autoencoder

- **Start from Autoencoder: Learn Feature From Input**



Unsupervised

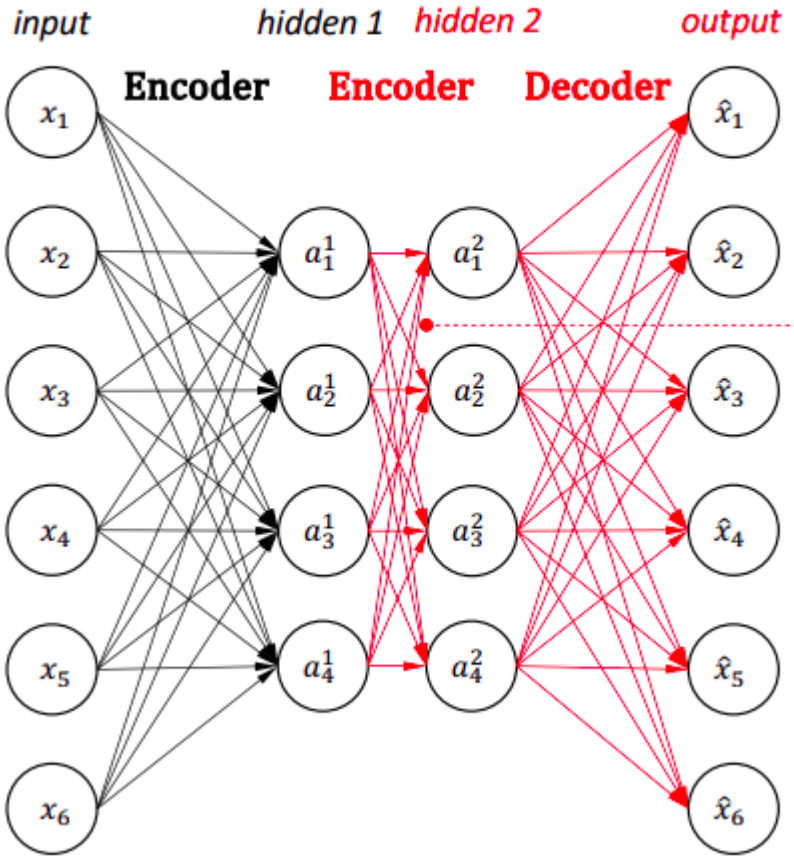
- The feature extractor for the input data

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

Stacked Autoencoder

- **2nd Stage: Learn 2nd Level Feature From 1st Level Feature**



Unsupervised

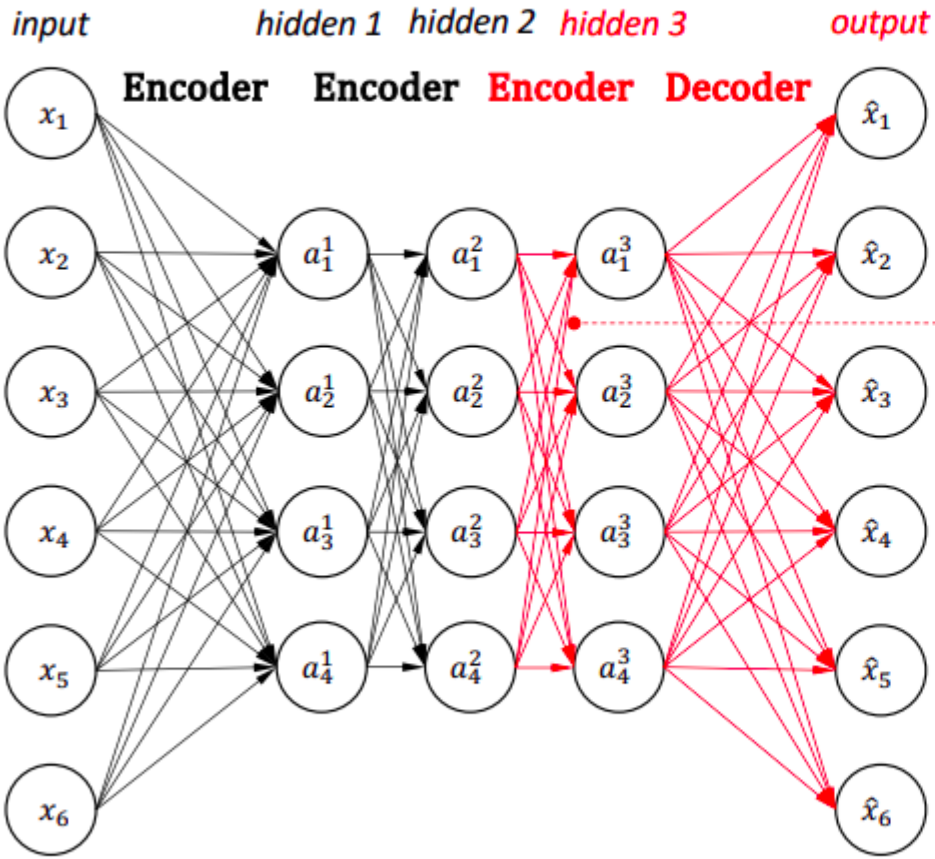
- The feature extractor for the first feature extractor

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

Stacked Autoencoder

- **3rd Stage: Learn 3rd Level Feature From 2nd Level Feature**



Unsupervised

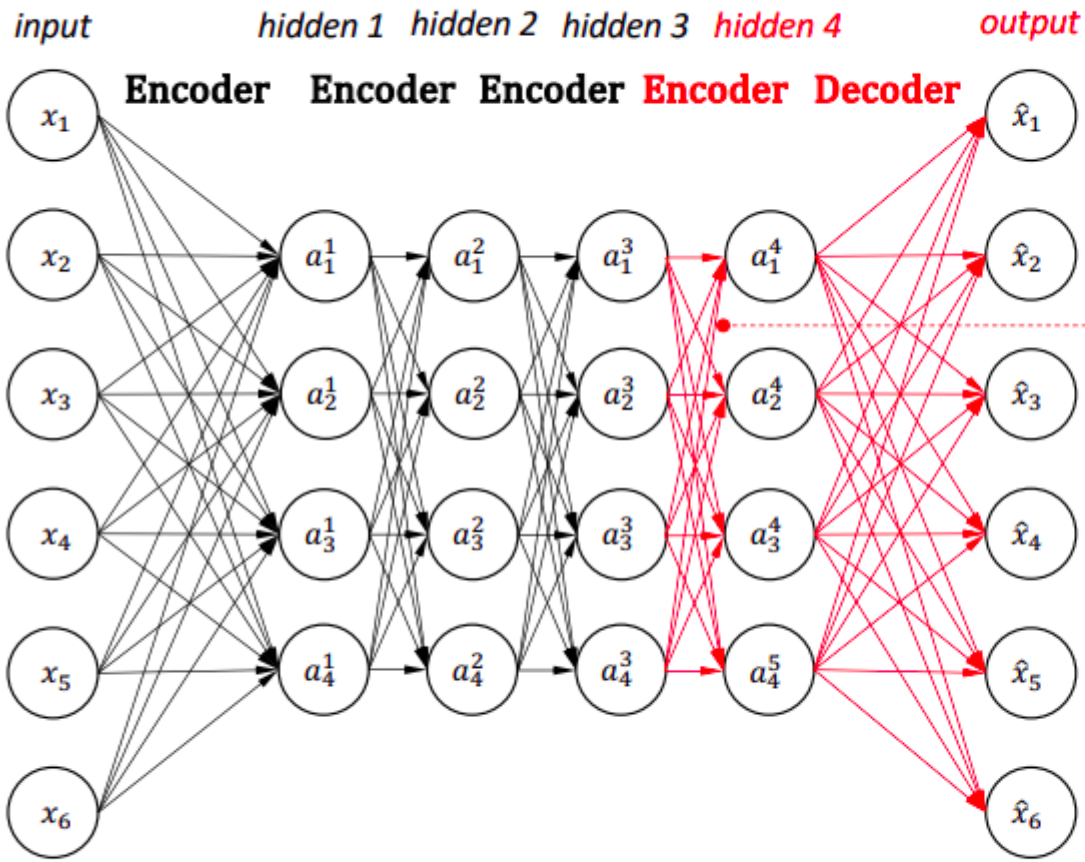
- The feature extractor for the second feature extractor

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

Stacked Autoencoder

- **4th Stage: Learn 4th Level Feature From 3rd Level Feature**



Unsupervised

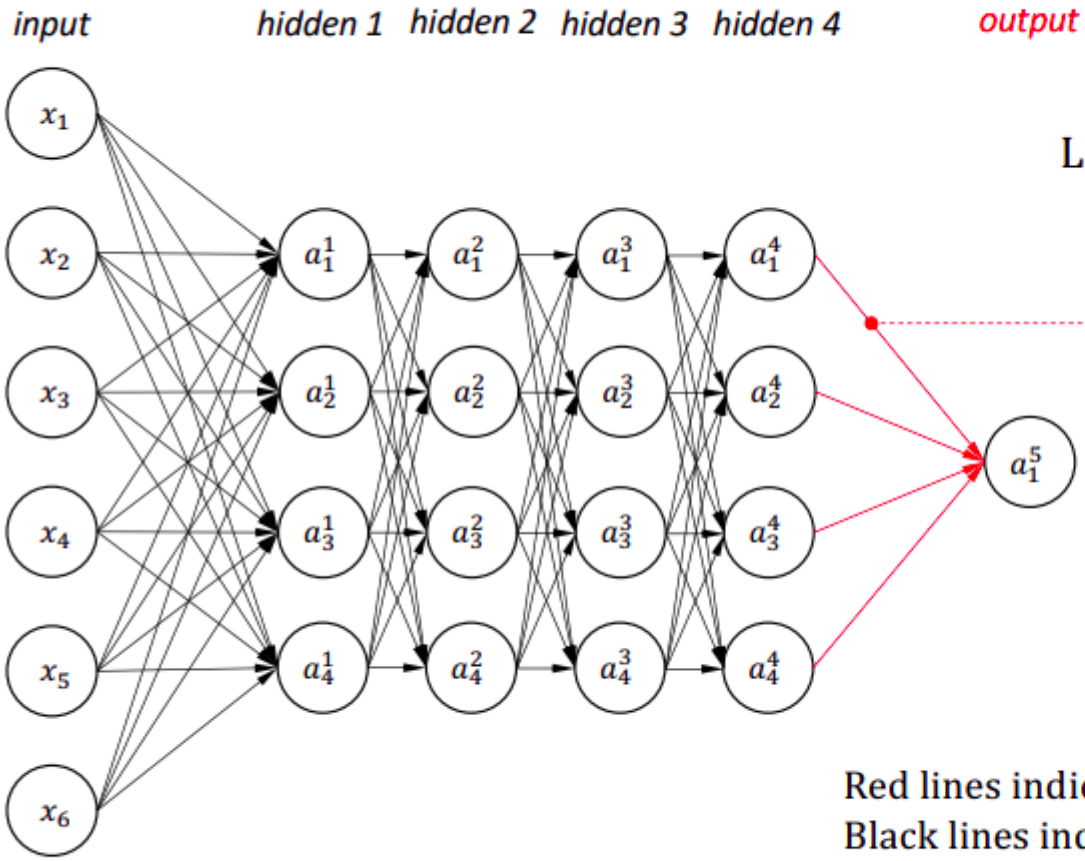
- The feature extractor for the third feature extracto

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

Stacked Autoencoder

- **Use the Learned Feature Extractor for Downstream Tasks**



Supervised

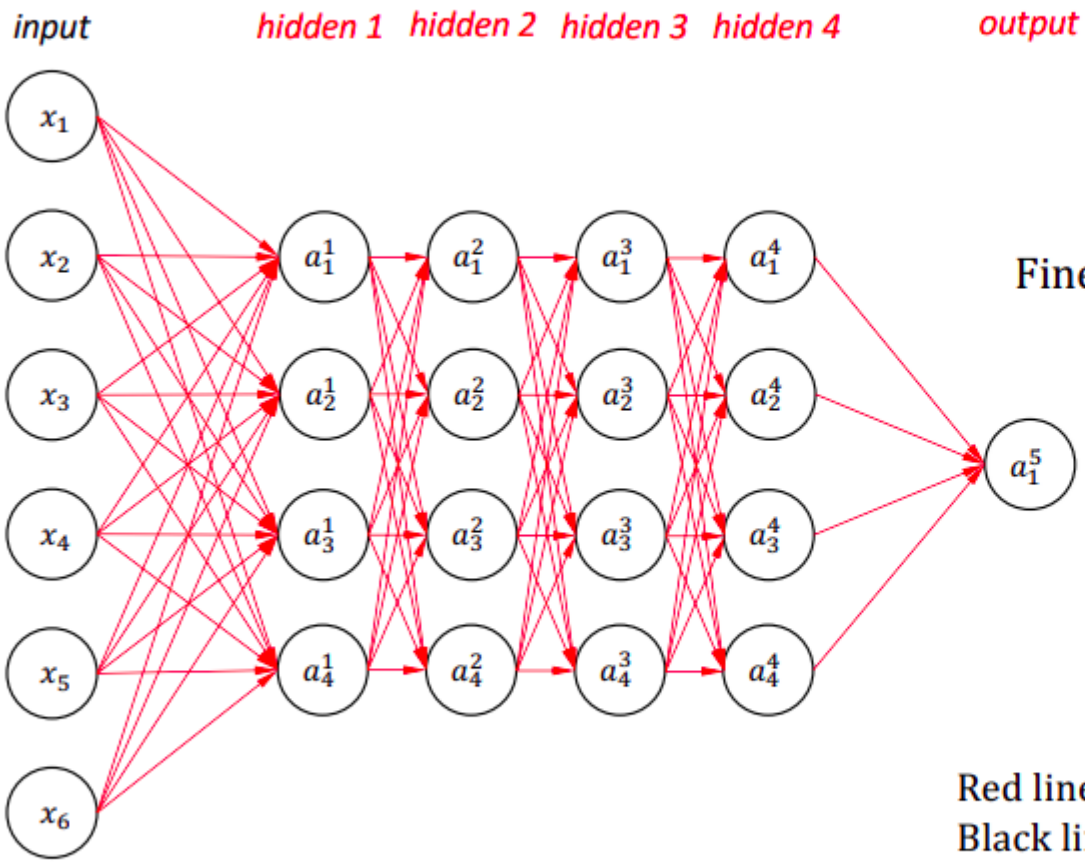
Learn to classify the input data by using the labels and high-level features

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

Stacked Autoencoder

- **Fine-tuning**



Supervised

Fine-tune the entire model for classification

Red color indicates the trainable weights

Red lines indicate the trainable weights
Black lines indicate the fixed/nontrainable weights

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)
  - From Neural Network Perspective
  - From Probability Model Perspective

Advantages of Stacked Autoencoder:

1. **Feature Learning Hierarchies:** Stacked autoencoders can learn hierarchical representations of the input data. Each layer in the stack learns increasingly abstract features, capturing complex patterns in the data. This hierarchical feature learning can lead to better performance in tasks such as classification and generation.

2. **Unsupervised Pre-training**: Stacked autoencoders can be pre-trained layer by layer in an unsupervised manner. This pre-training initializes the network parameters in a way that helps avoid the problem of vanishing gradients and speeds up convergence during supervised fine-tuning. It also helps in learning useful representations even with limited labeled data.

Disadvantages of Stacked Autoencoder:

1. **Model Complexity and Training Time**: Stacked autoencoders with multiple layers can be computationally expensive to train, especially on large datasets. Training deep architectures requires significant computational resources and time. Additionally, deep models are prone to overfitting, requiring careful regularization and tuning to prevent.

2. **Difficulty in Interpretability**: As the number of layers increases in a stacked autoencoder, the interpretability of learned features becomes more challenging. Understanding the representations learned by intermediate layers can be complex, making it difficult to interpret how the model makes decisions. This lack of interpretability can be a drawback in applications where transparency and explainability are important.

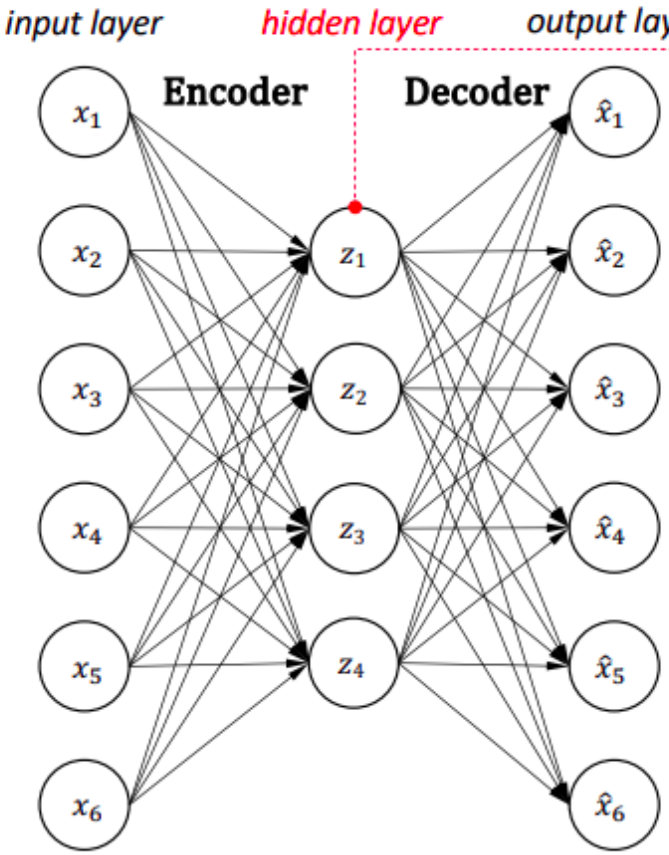
Before we start

- **Question?**
  - Are the previous Autoencoders generative model?
  - Recap: We want to learn a probability distribution $p(x)$ over x
  - **Generation (sampling):** $\mathbf{x}_{new} \sim p(\mathrm{x})$
    (NO, The compressed latent codes of autoencoders are not prior distributions, autoencoder cannot learn to represent the data distribution)
  - **Density Estimation:** $p(\mathrm{x})$ high if $\mathbf{x}$ looks like a real data
    NO
  - **Unsupervised Representation Learning:**
    Discovering the underlying structure from the data distribution (e.g., ears, nose, eyes …)
    (YES, Autoencoders learn the feature representation)

- Vanilla Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder
- Stacked Autoencoder
- Variational Autoencoder (VAE)
  - From Neural Network Perspective
  - From Probability Model Perspective

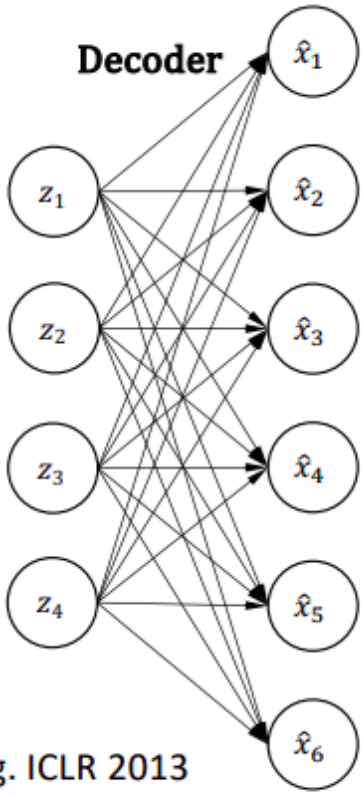
Variational Autoencoder

- **How to perform generation (sampling)?**



Can the hidden output be a prior distribution, e.g., Normal distribution?

$N(0,1)$



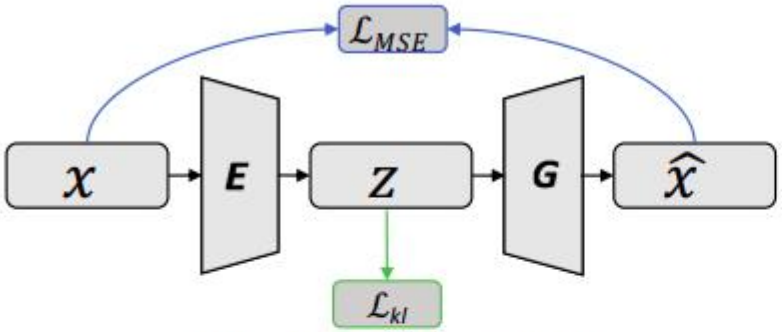
$$p(X) = \sum_Z p(X|Z)p(Z)$$

Decoder(Generator) maps $N(0,1)$ to data space

Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

Variational Autoencoder

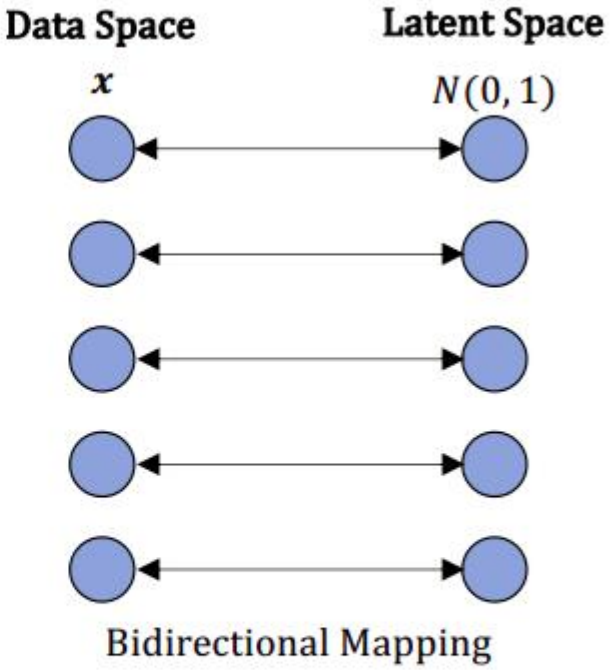
- **Quick Overview**



$q(z|x)$
Inference
(encoder)

$p(x|z)$
generation
(decode)

$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \mathcal{L}_{kl}$$



Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

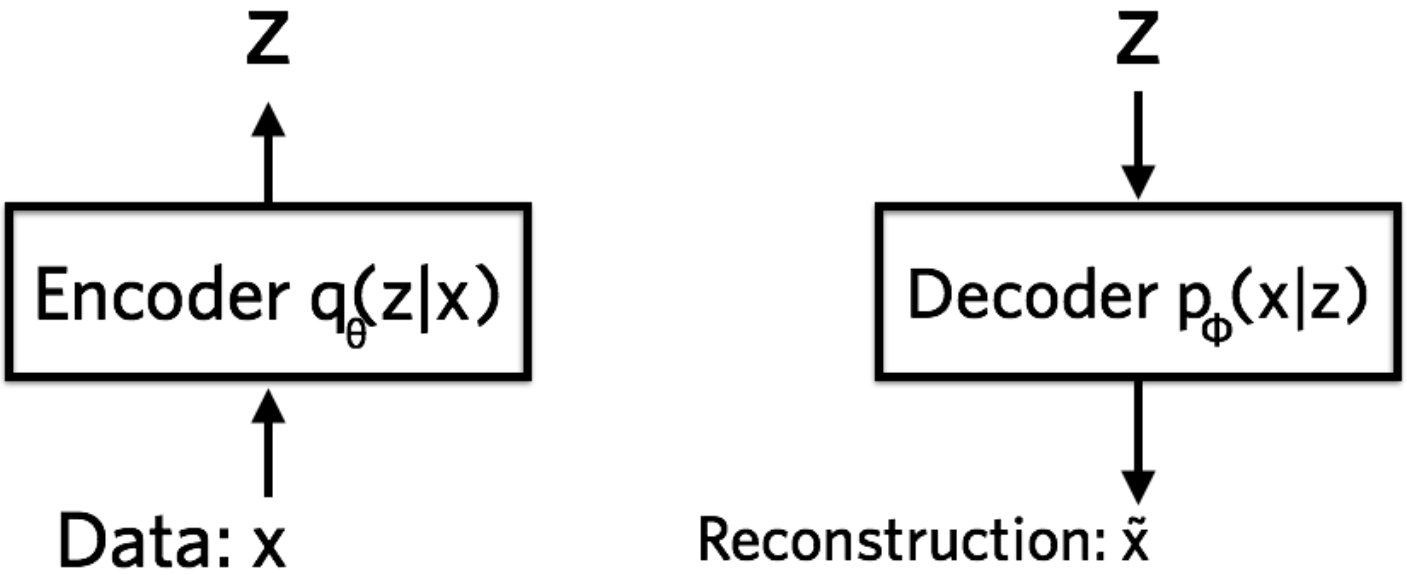
Variational Autoencoder

- **The neural net perspective**
  - A variational autoencoder consists of an encoder, a decoder, and a loss function

Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

Variational Autoencoder

- **Encoder, Decoder**



Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

Variational Autoencoder

- **Loss function**

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

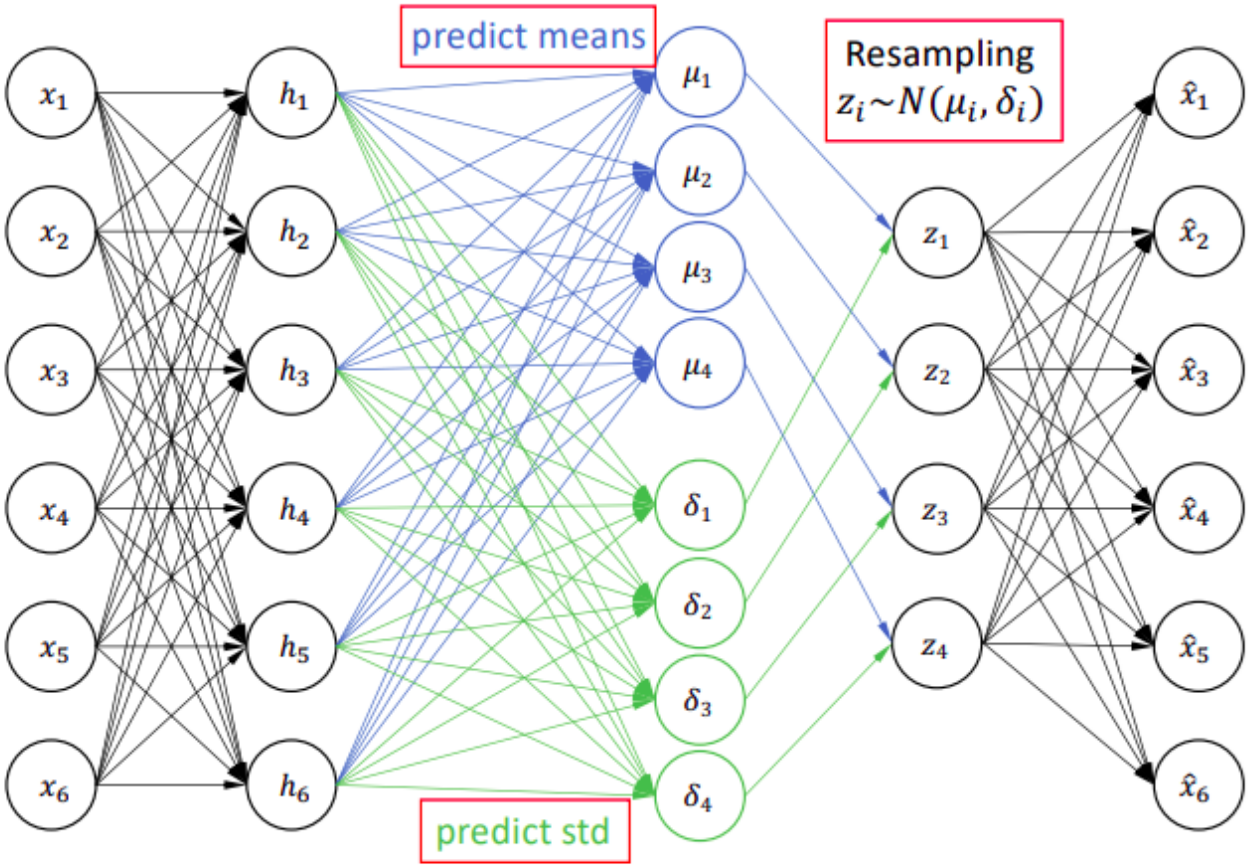
Can be represented by MSE

regularization

Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

Variational Autoencoder

- **Reparameterization Trick**



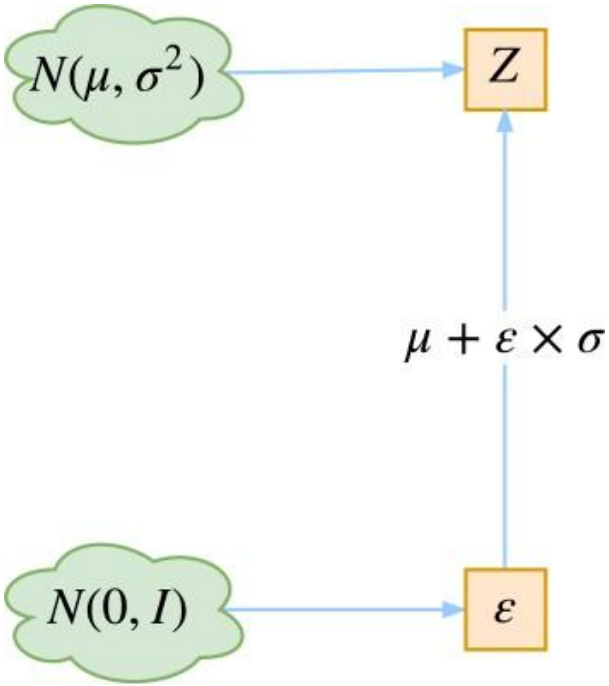
Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

1. Encode the input
2. Predict means
3. Predict standard derivations
4. Use the predicted means and standard derivations to sample new latent variables individually
5. Reconstruct the input

Variational Autoencoder

- **Reparameterization Trick**
  - z ~ N(μ, σ) is not differentiable
  - To make sampling z differentiable
  - z = μ + σ * ϵ ϵ ∼ *N(0, 1)*

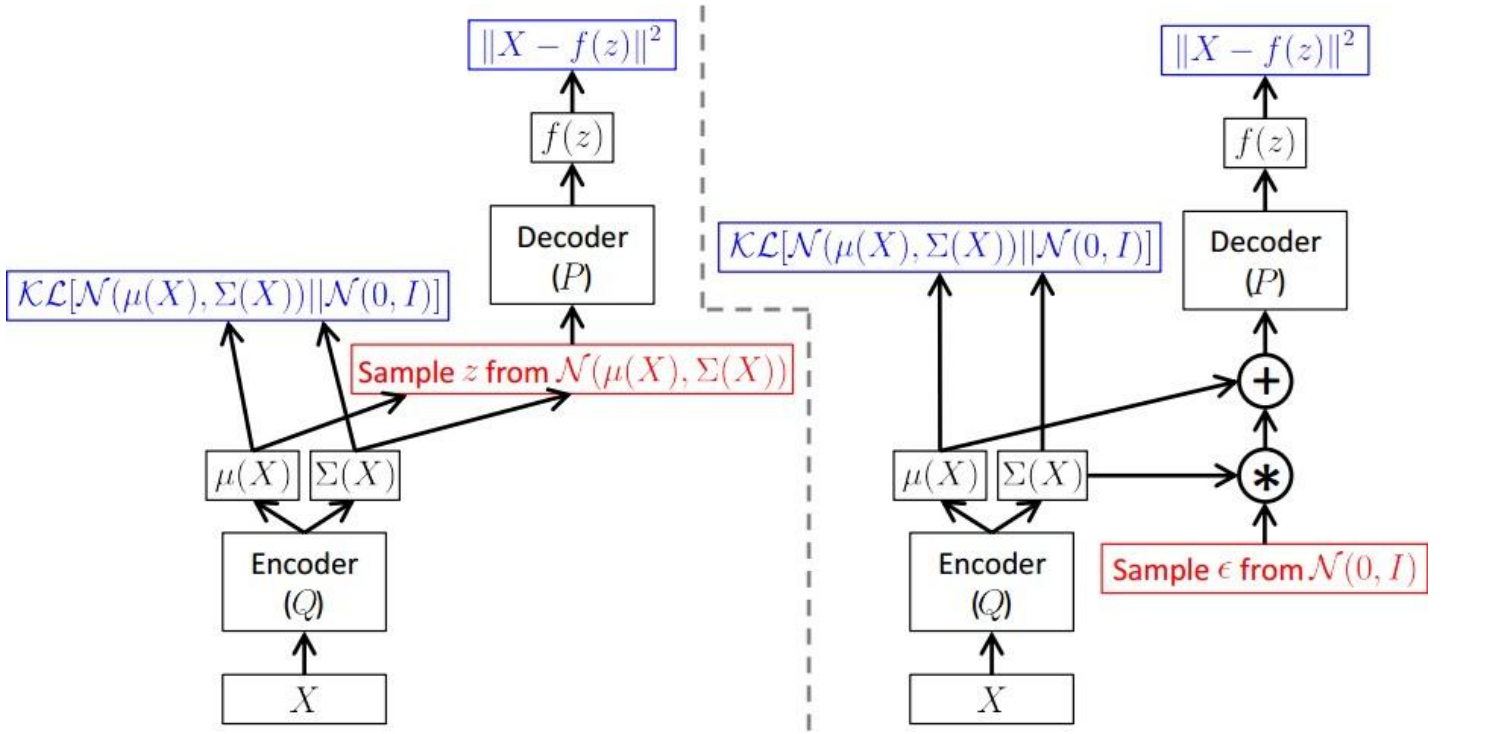
$$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)dz$$
$$=\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\right]d\left(\frac{z-\mu}{\sigma}\right)$$



Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

Variational Autoencoder

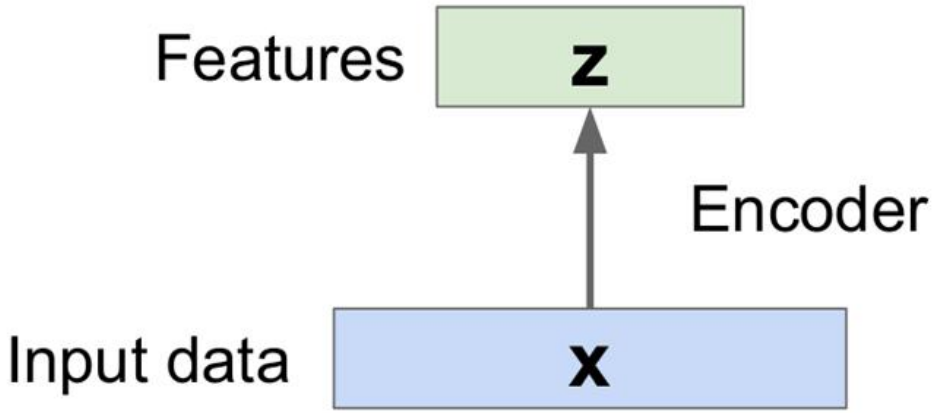
- **Reparameterization Trick**



Auto-Encoding Variational Bayes. Diederik P. Kingma, Max Welling. ICLR 2013

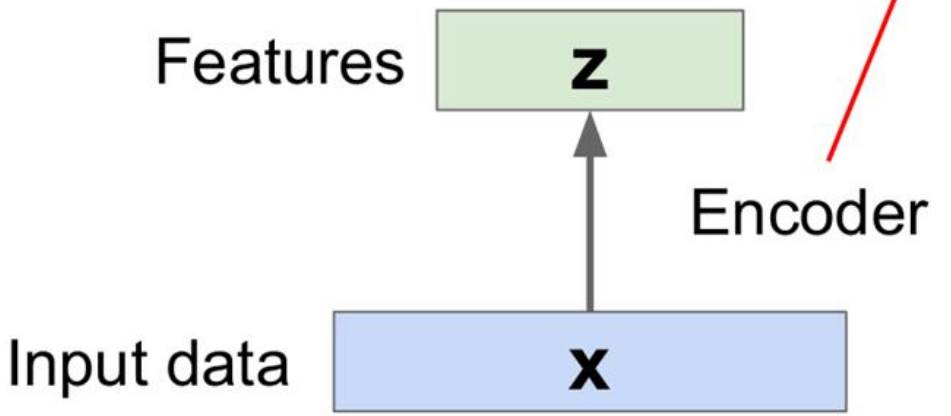
MORE MATHS ON VAE

Autoencoders



Autoencoders

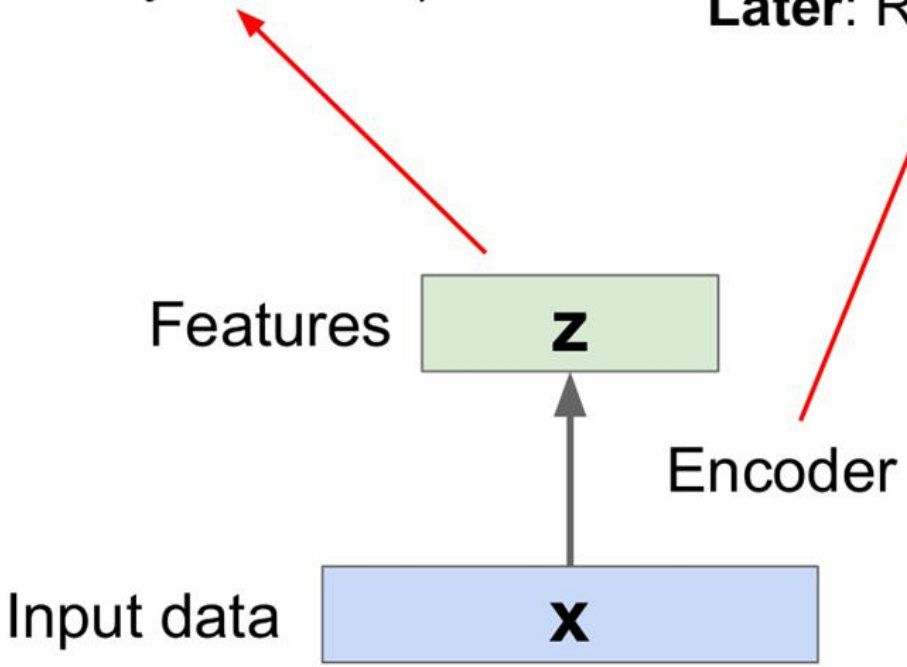
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

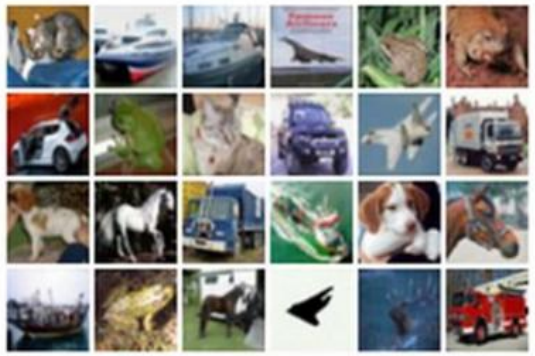


Autoencoders

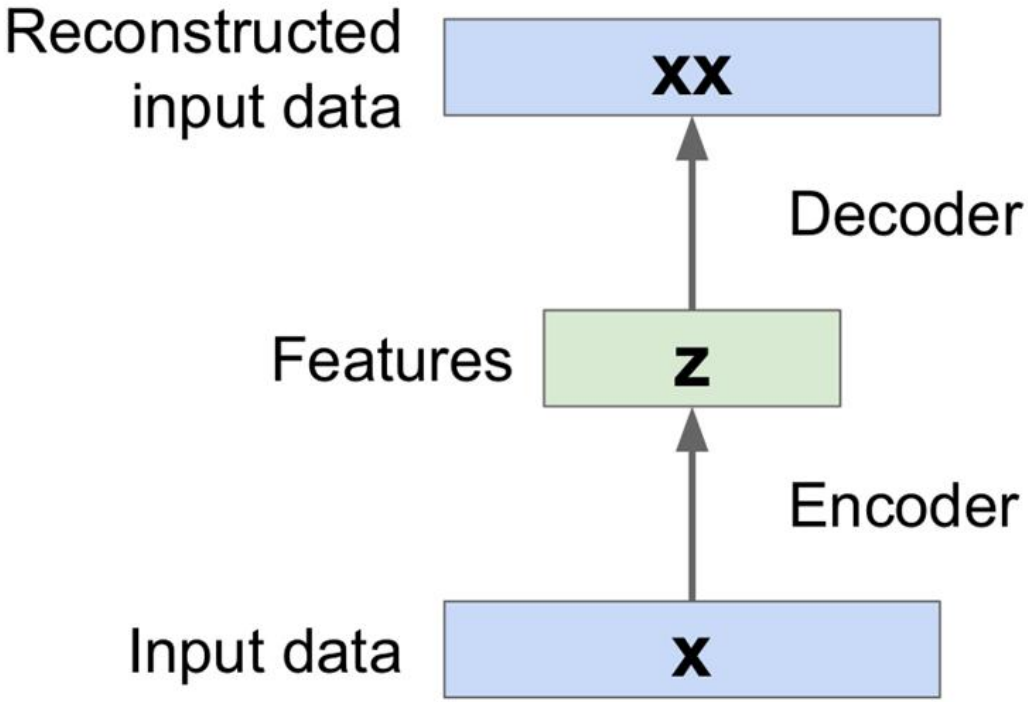
z usually smaller than **x**
(dimensionality reduction)

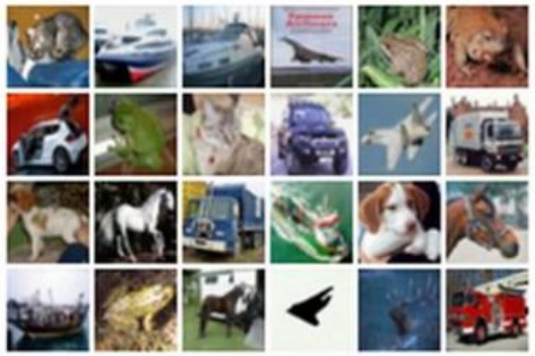
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN





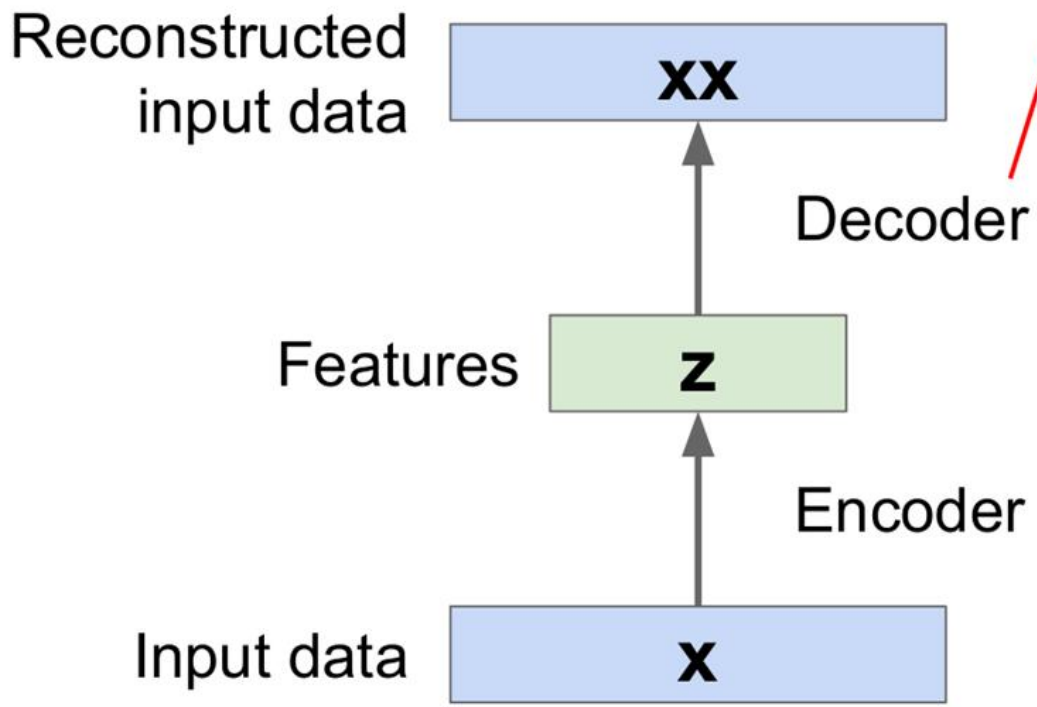
Autoencoders



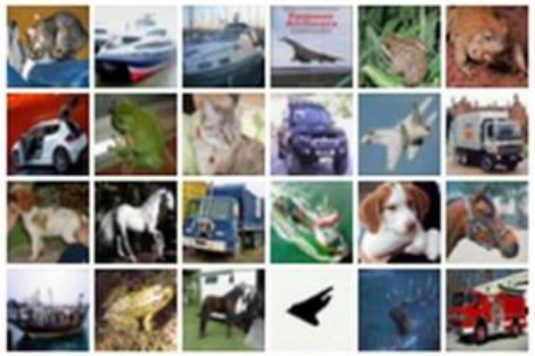


Autoencoders

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



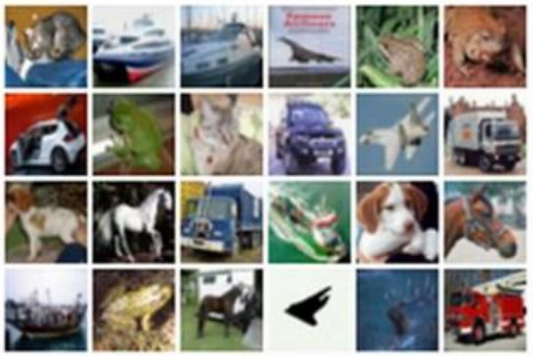
Encoder: 4-layer conv
Decoder: 4-layer upconv

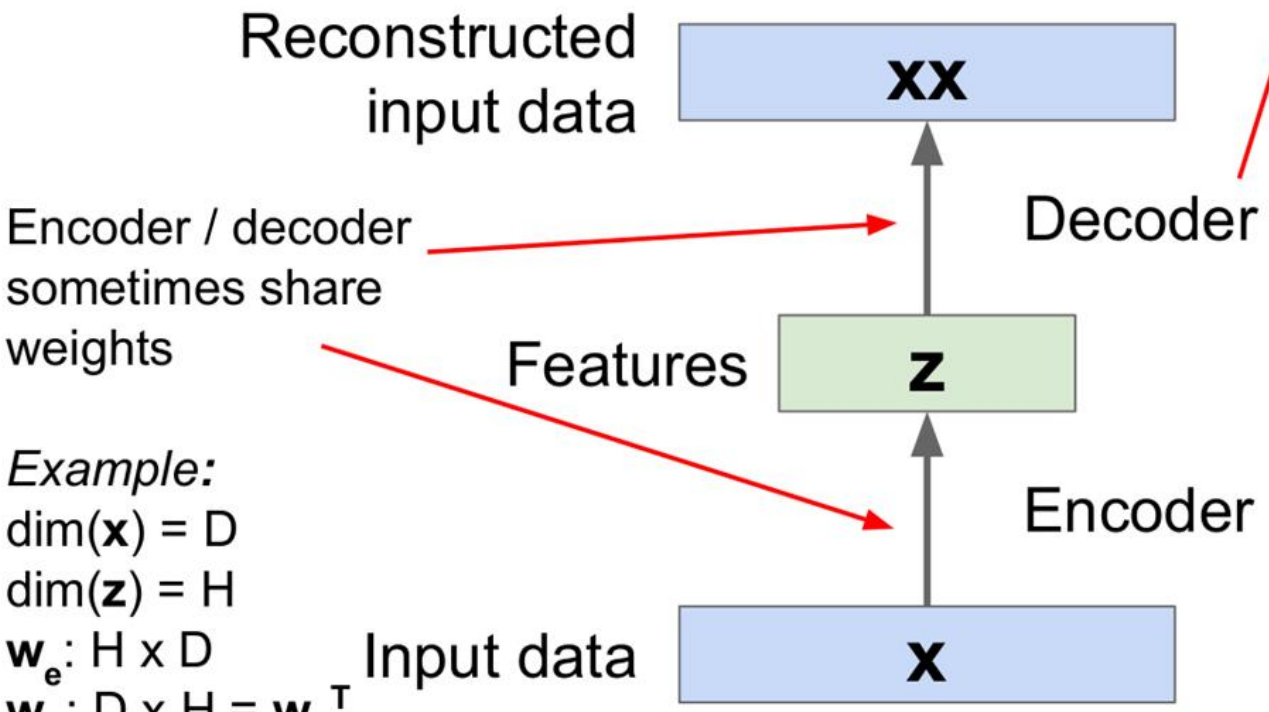


Autoencoders

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

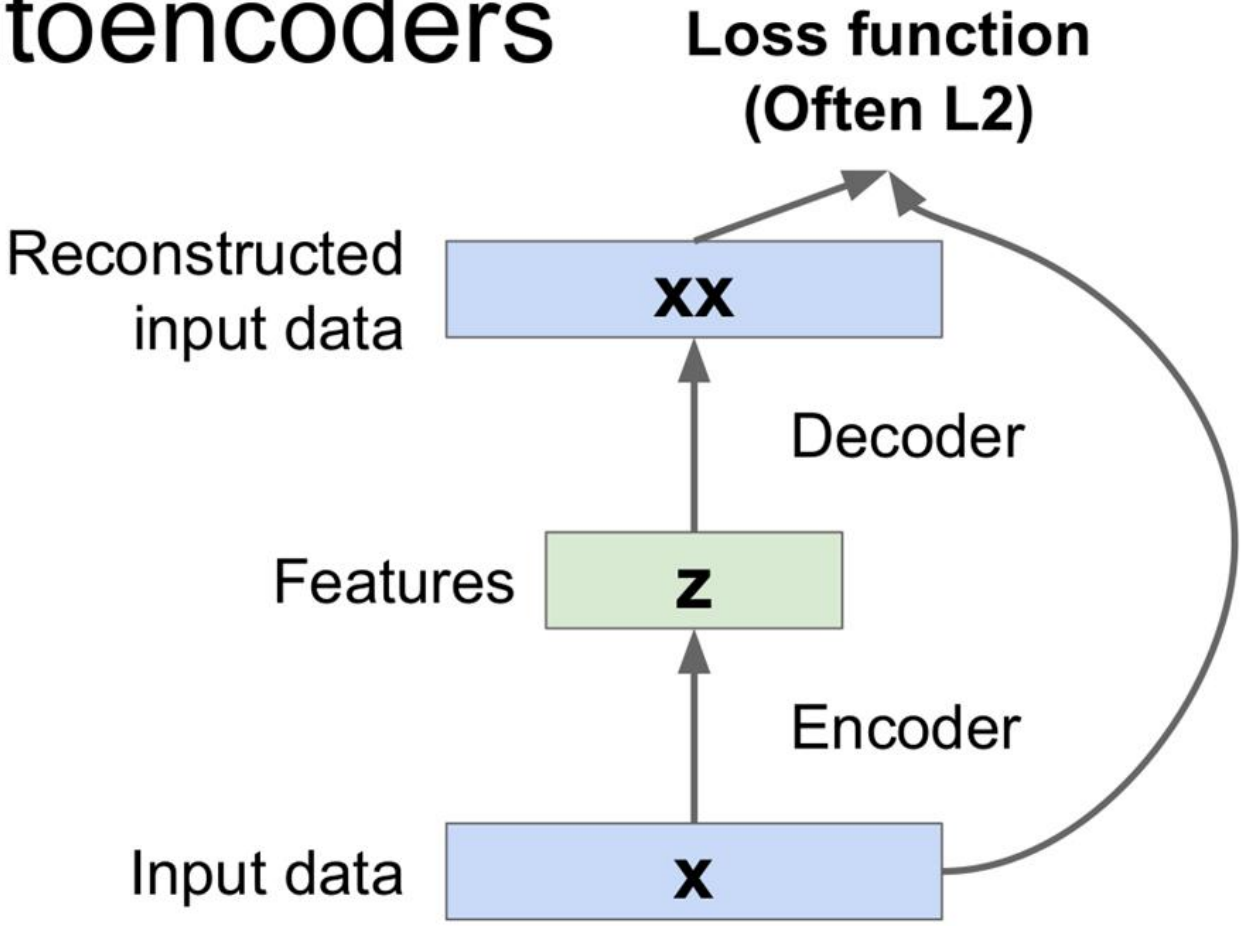
Train for reconstruction with no labels!



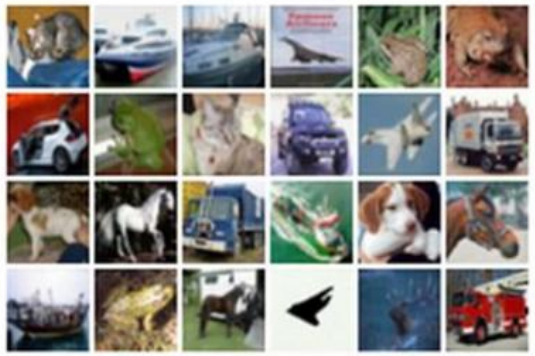


Example:
dim(**x**) = D
dim(**z**) = H
$\mathbf{w}_e$: H x D
$\mathbf{w}_d$: D x H = $\mathbf{w}_e^T$

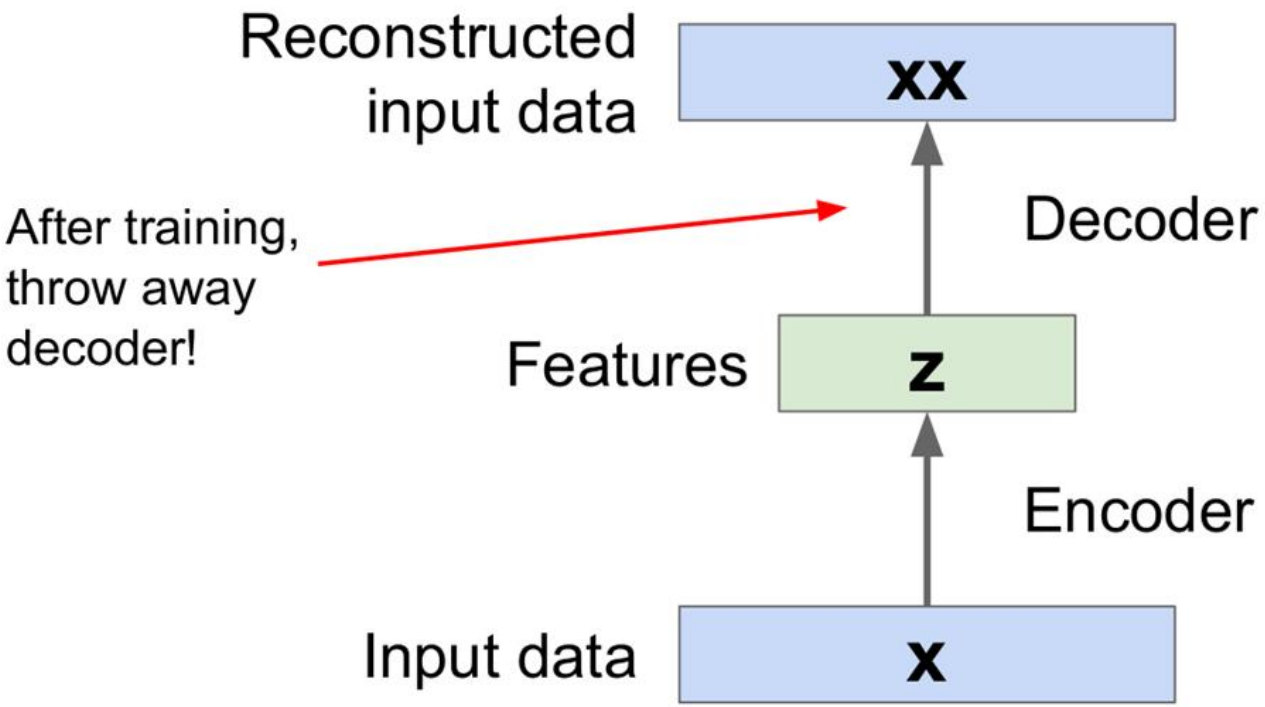
Autoencoders

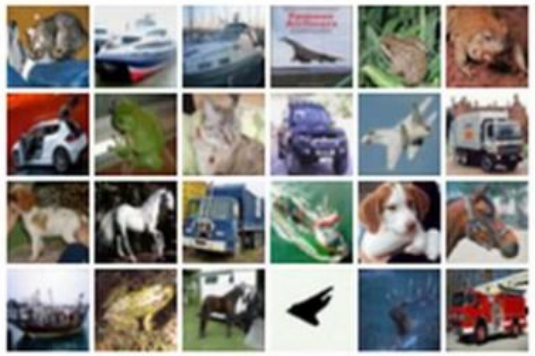


Train for
reconstruction
with no labels!

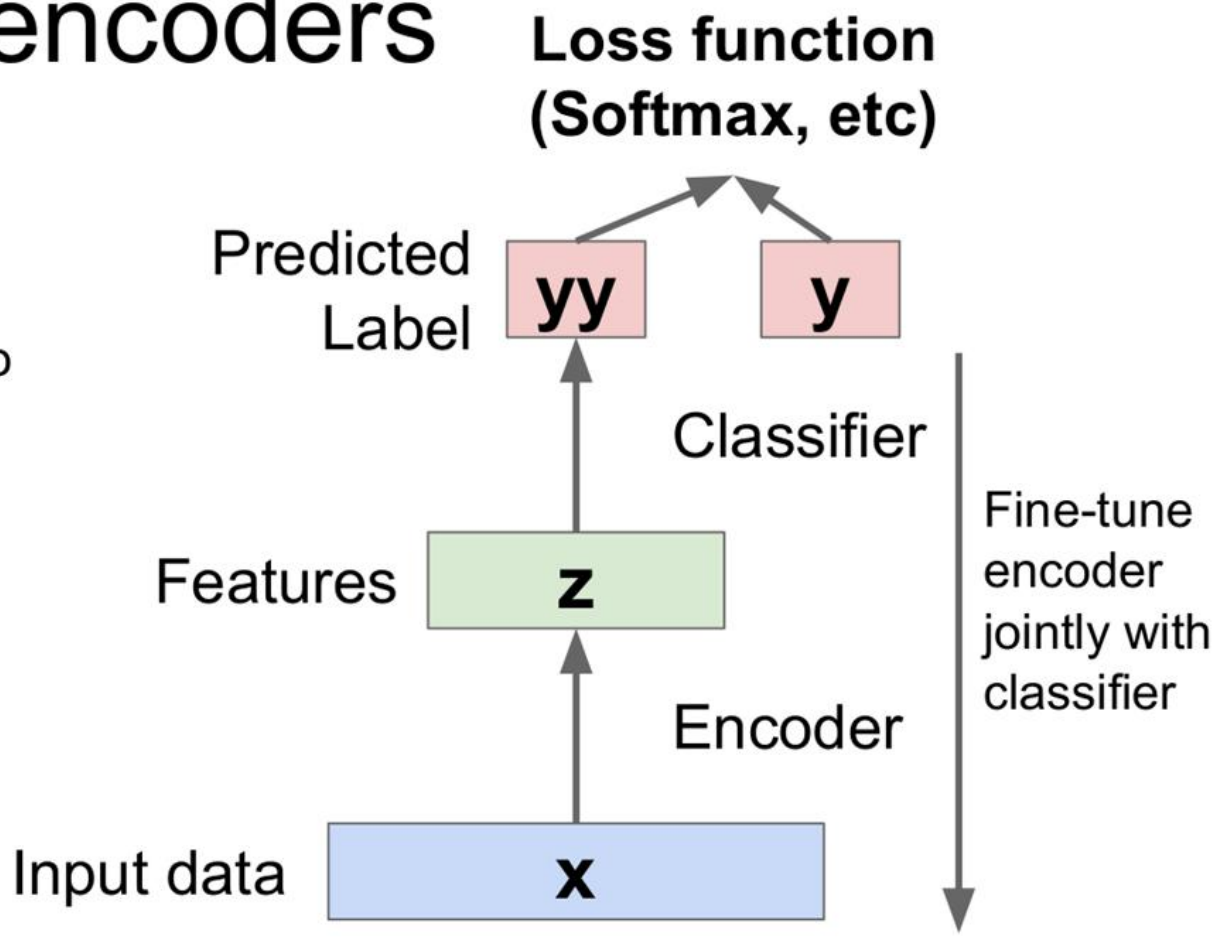


Autoencoders





Autoencoders

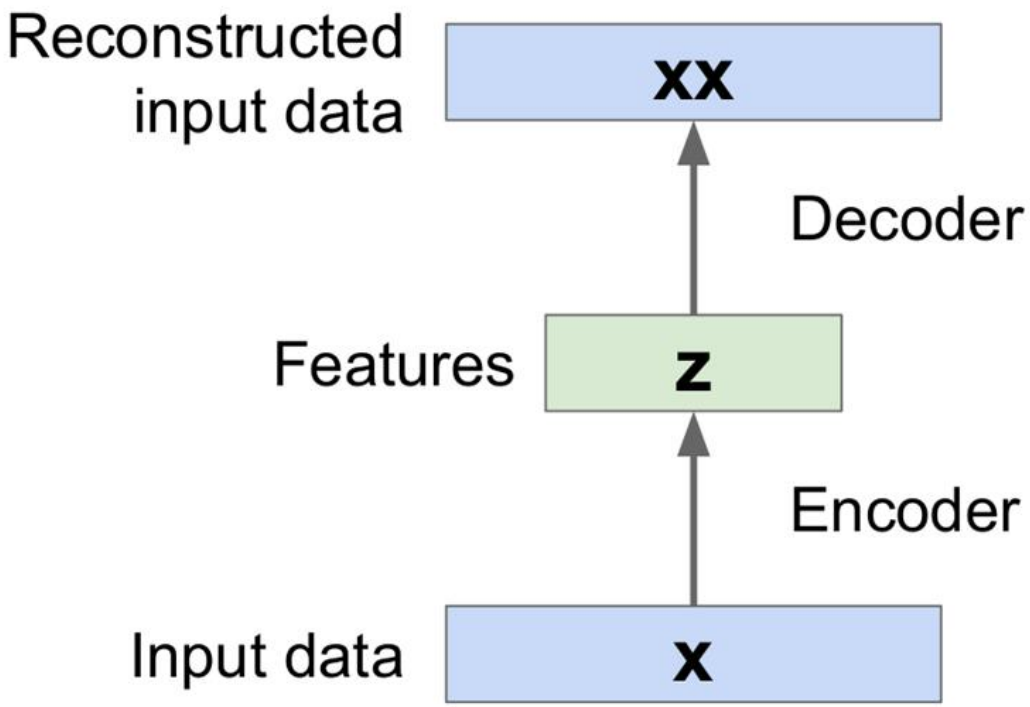


Use encoder to initialize a **supervised** model

bird plane
dog deer truck

Train for final task (sometimes with small data)

Autoencoders



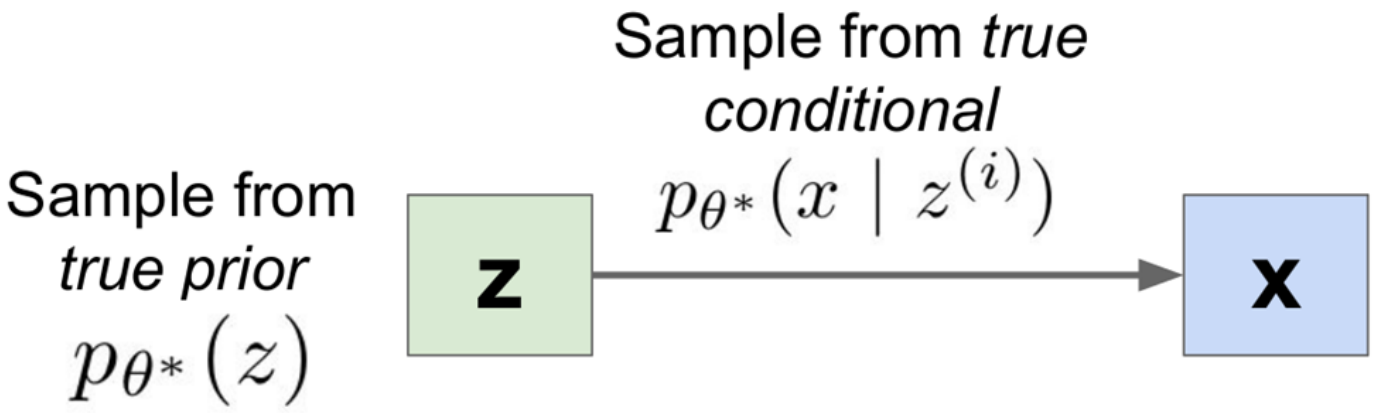
Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Can we generate images from an autoencoder?

Variational Autoencoder

A Bayesian spin on an autoencoder - lets us generate data!

Assume our data $\{x^{(i)}\}_{i=1}^{N}$ is generated like this:



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoder

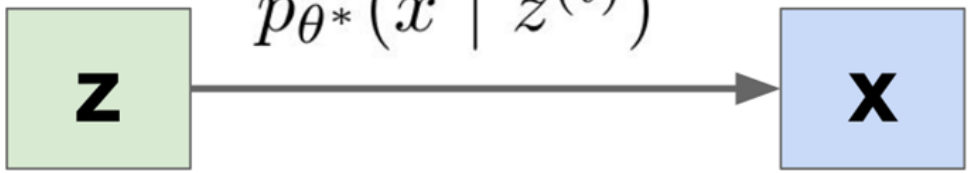
A Bayesian spin on an autoencoder!

Assume our data $\{x^{(i)}\}_{i=1}^{N}$ is generated like this:

Intuition: **x** is an image, **z** gives class, orientation, attributes, etc

Sample from *true prior* $p_{\theta^*}(z)$

Sample from *true conditional* $p_{\theta^*}(x \mid z^{(i)})$



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoder

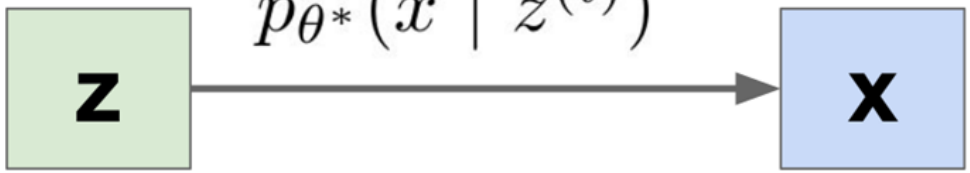
A Bayesian spin on an autoencoder!

Assume our data $\{x^{(i)}\}_{i=1}^{N}$ is generated like this:

Intuition: **x** is an image, **z** gives class, orientation, attributes, etc

Sample from *true prior* $p_{\theta^*}(z)$

Sample from *true conditional* $p_{\theta^*}(x \mid z^{(i)})$



Problem: Estimate θ without access to latent states $z^{(i)}$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoder

Prior: Assume $p_\theta(z)$ is a unit Gaussian

Kingma and Welling, ICLR 2014

Variational Autoencoder

Prior: Assume $p_\theta(z)$ is a unit Gaussian

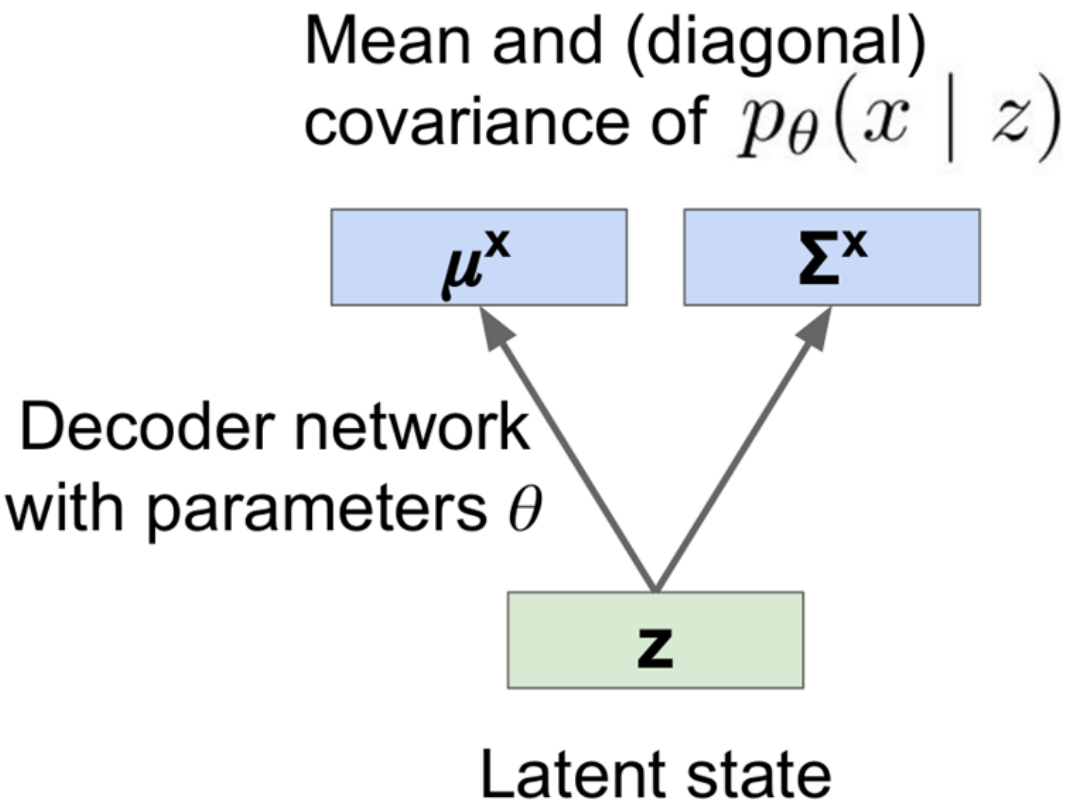
Conditional: Assume $p_\theta(x \mid z)$ is a diagonal Gaussian, predict mean and variance with neural net

Kingma and Welling, ICLR 2014

Variational Autoencoder

Prior: Assume $p_\theta(z)$ is a unit Gaussian

Conditional: Assume $p_\theta(x \mid z)$ is a diagonal Gaussian, predict mean and variance with neural net

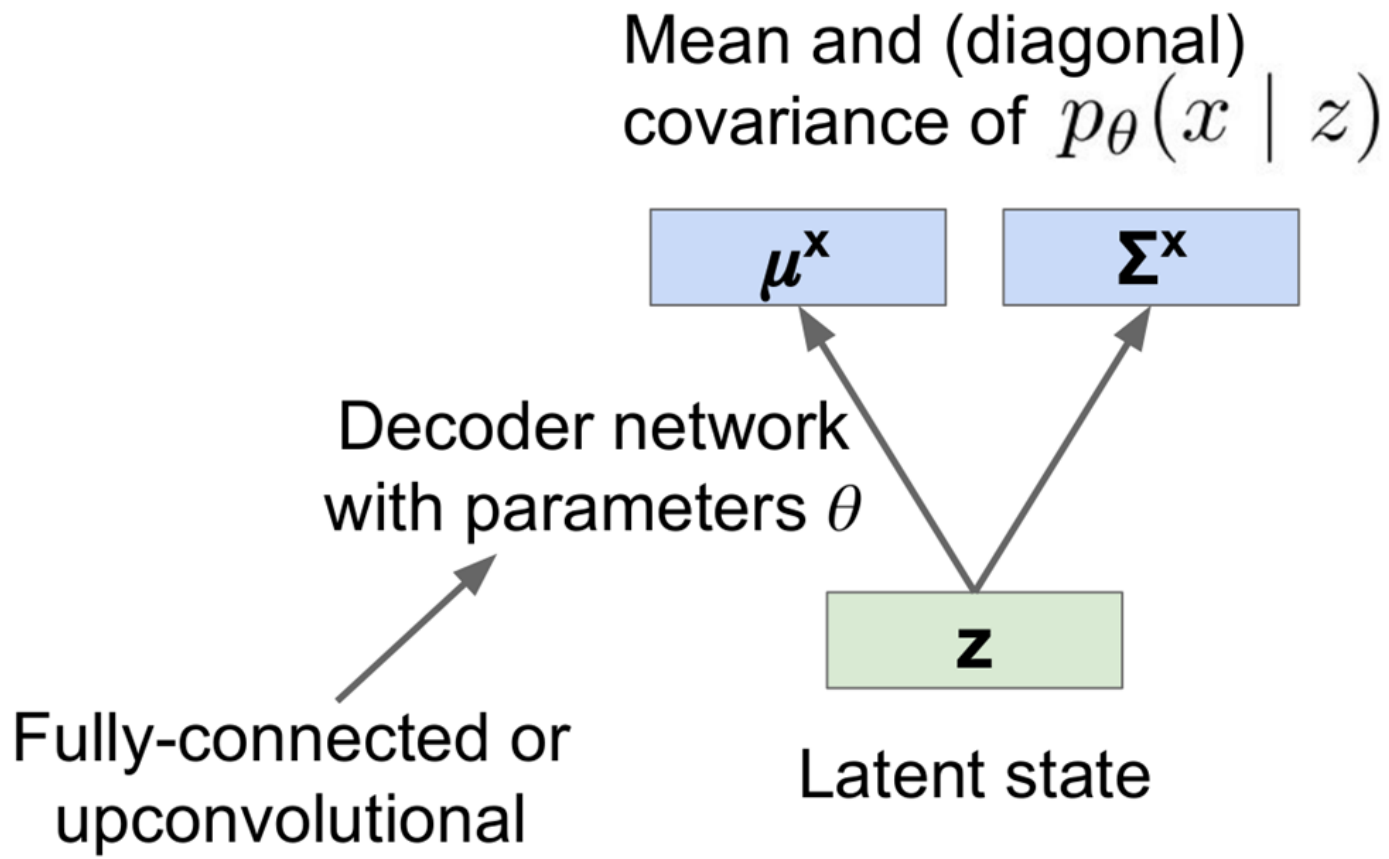


Kingma and Welling, ICLR 2014

Variational Autoencoder

Prior: Assume $p_\theta(z)$ is a unit Gaussian

Conditional: Assume $p_\theta(x \mid z)$ is a diagonal Gaussian, predict mean and variance with neural net



Kingma and Welling, ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(x)}$$

Kingma and Welling,
ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p_\theta(z)}{p_\theta(x)}$$

Use decoder network =)

Gaussian =)

Intractible integral =(

Kingma and Welling,
ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p_\theta(z)}{p_\theta(x)}$$

Use decoder network =)

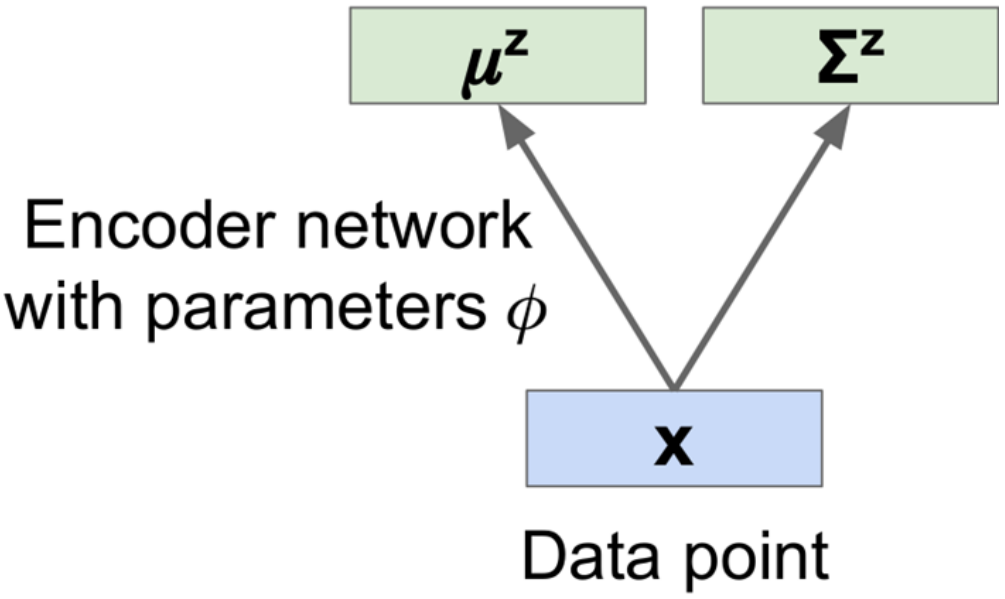
Gaussian =)

Intractible integral =(

Kingma and Welling,
ICLR 2014

Mean and (diagonal) covariance of

$$q_\phi(z \mid x)$$



Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p_\theta(z)}{p_\theta(x)}$$

Use decoder network =)

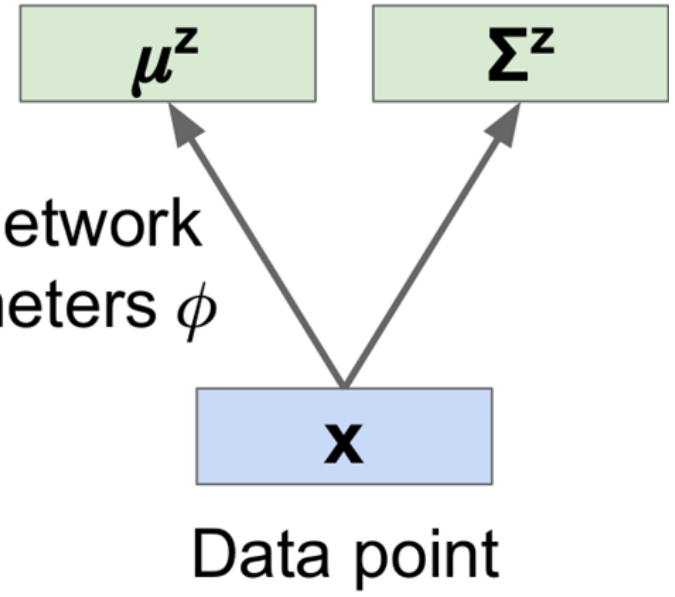
Gaussian =)

Intractible integral =(

Approximate posterior with **encoder network** $q_\phi(z \mid x)$

Kingma and Welling, ICLR 2014

Mean and (diagonal) covariance of $q_\phi(z \mid x)$



Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p_\theta(z)}{p_\theta(x)}$$

Use decoder network =)

Gaussian =)

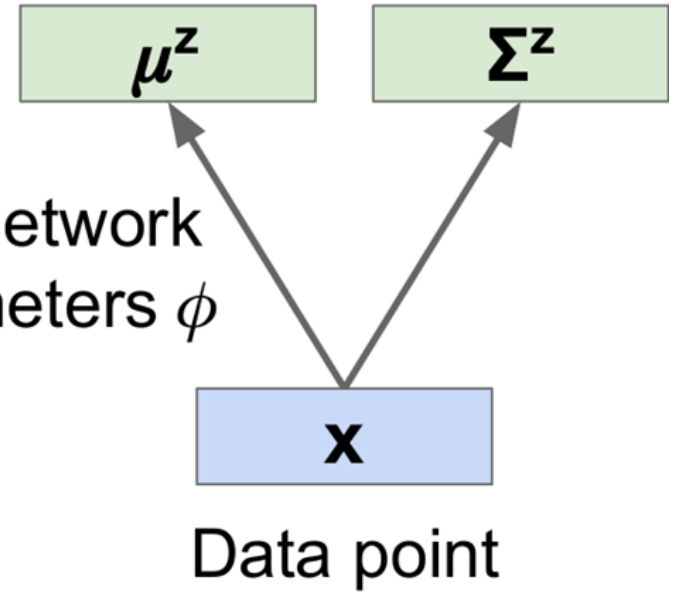
Intractible integral =(

Approximate posterior with **encoder network** $q_\phi(z \mid x)$

Kingma and Welling, ICLR 2014

Fully-connected or convolutional

Mean and (diagonal) covariance of $q_\phi(z \mid x)$

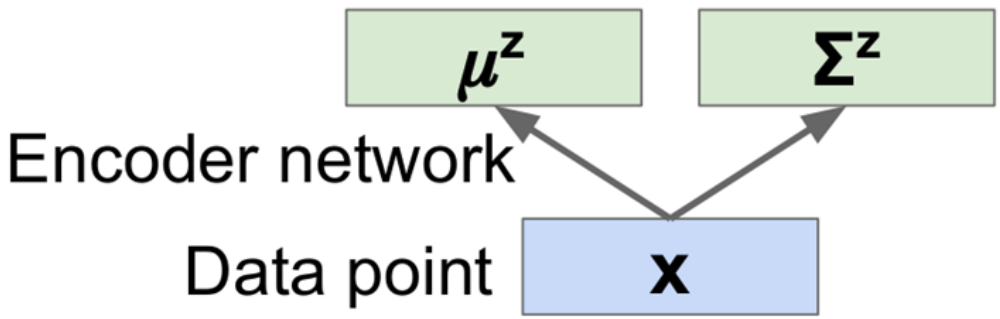


Variational Autoencoder

Data point 

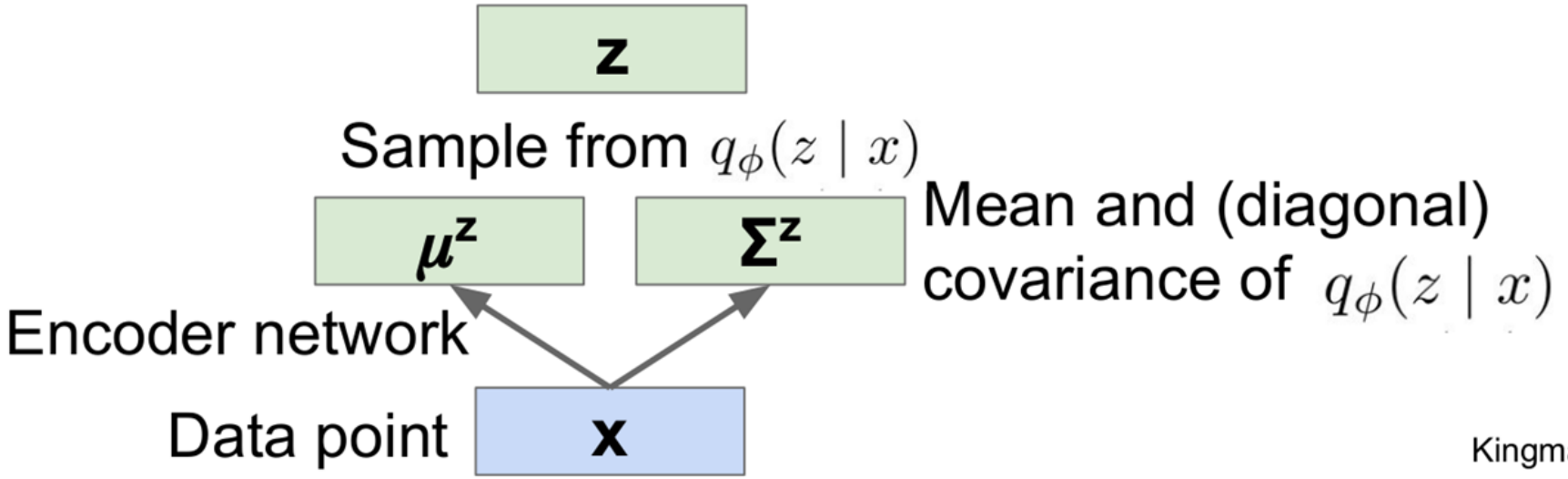
Kingma and Welling, ICLR 2014

Variational Autoencoder



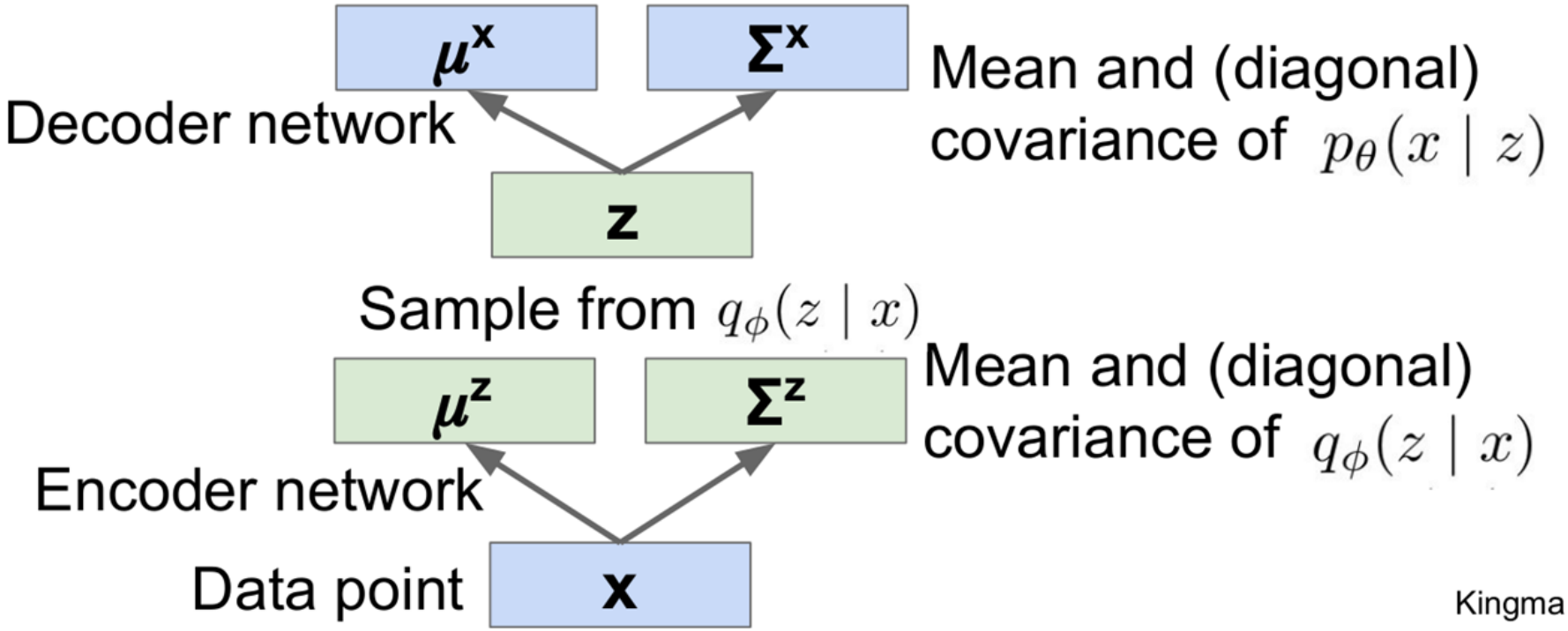
Kingma and Welling, ICLR 2014

Variational Autoencoder



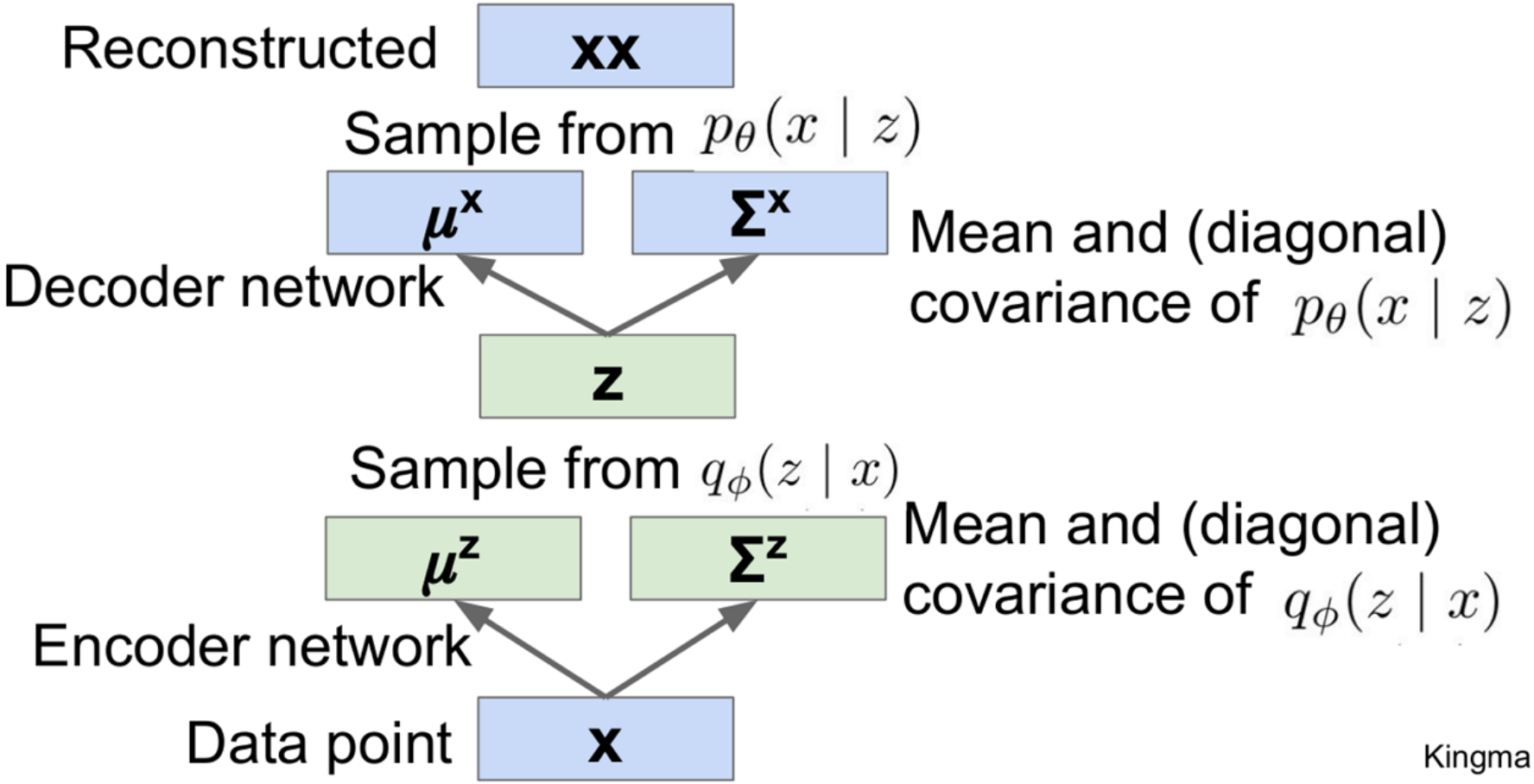
Kingma and Welling, ICLR 2014

Variational Autoencoder



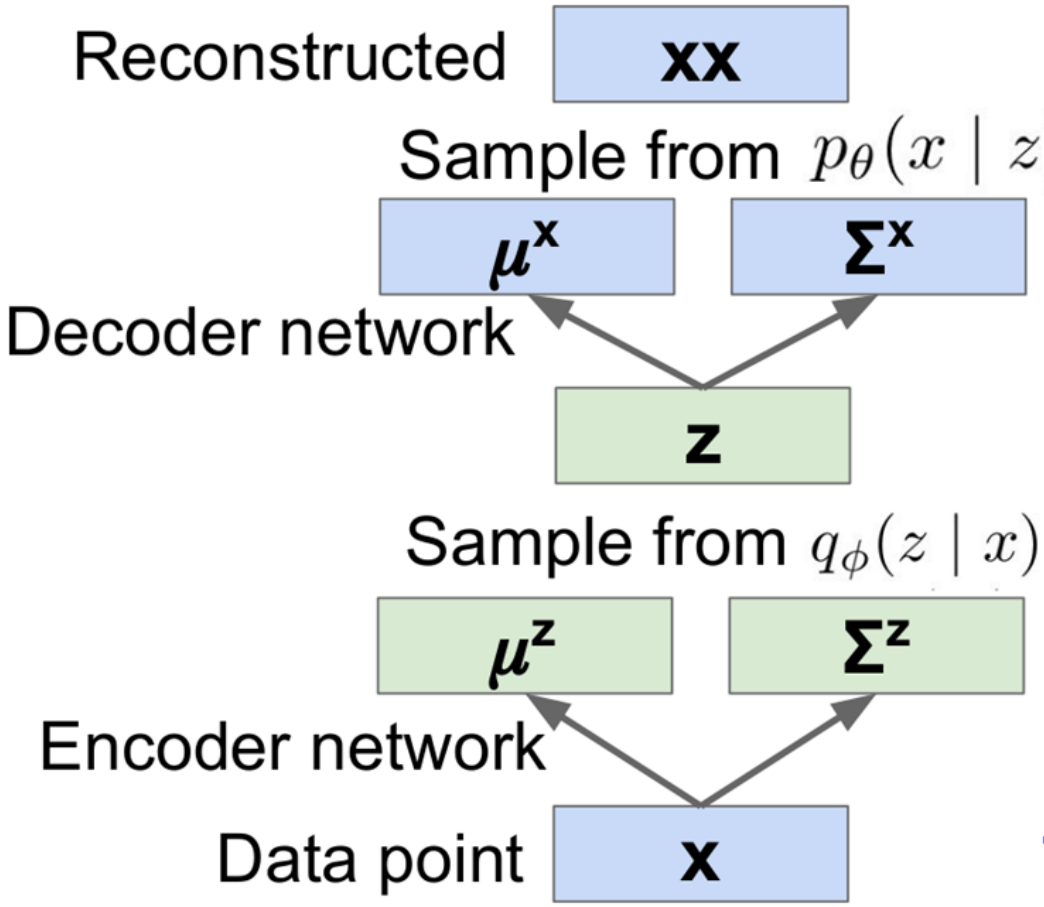
Kingma and Welling, ICLR 2014

Variational Autoencoder



Kingma and Welling, ICLR 2014

Variational Autoencoder



Training like a normal autoencoder:
reconstruction loss at the end,
regularization toward prior in middle

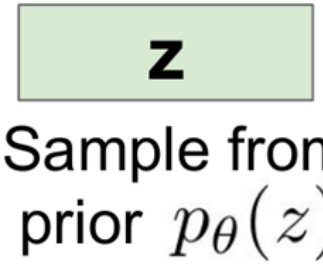
Mean and (diagonal) covariance of $p_\theta(x \mid z)$
(should be close to data x)

Mean and (diagonal) covariance of $q_\phi(z \mid x)$
(should be close to prior $p_\theta(z)$)

Kingma and Welling, ICLR 2014

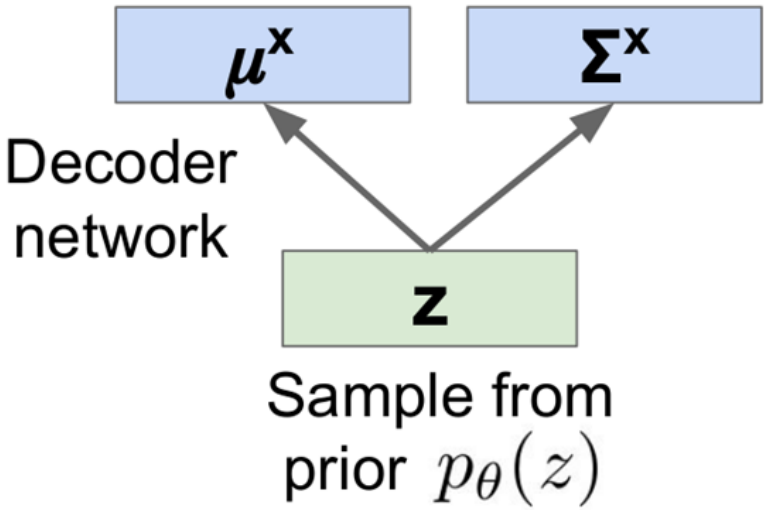
Variational Autoencoder: Generate Data!

After network is trained:



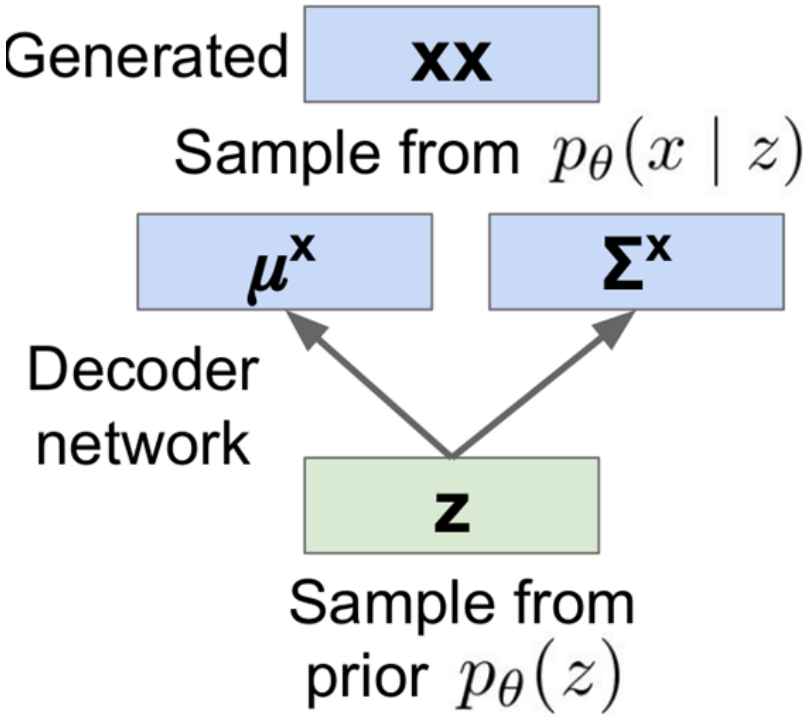
Variational Autoencoder: Generate Data!

After network is trained:



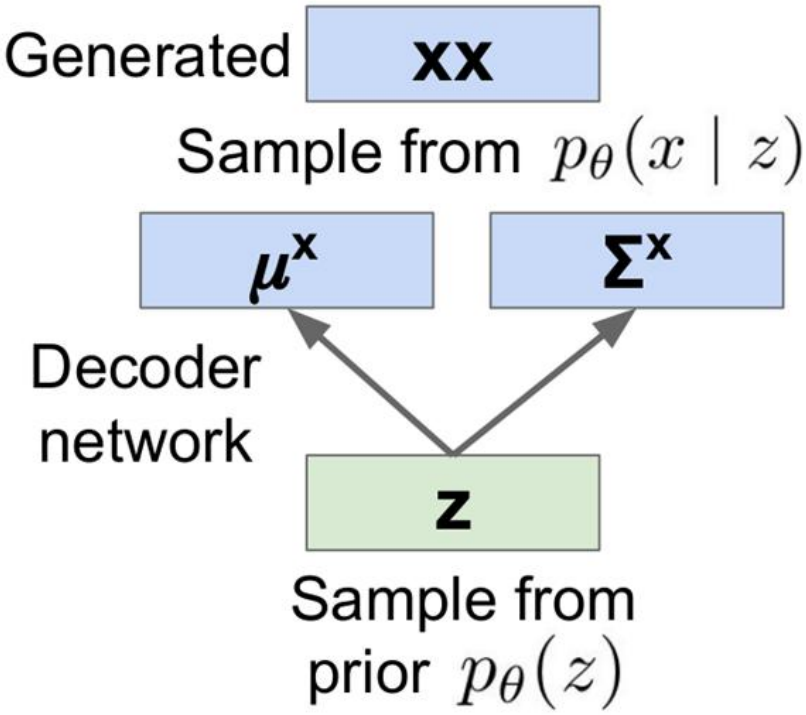
Variational Autoencoder: Generate Data!

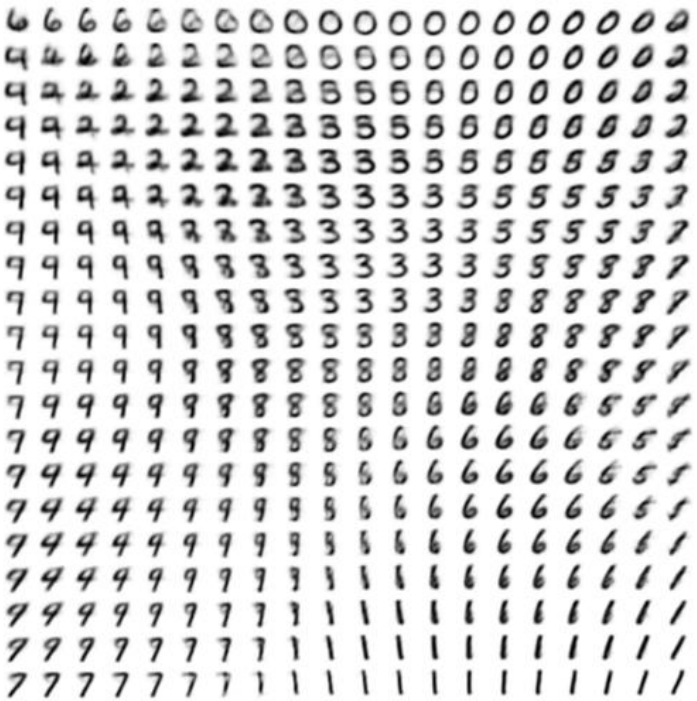
After network is trained:



Variational Autoencoder: Generate Data!

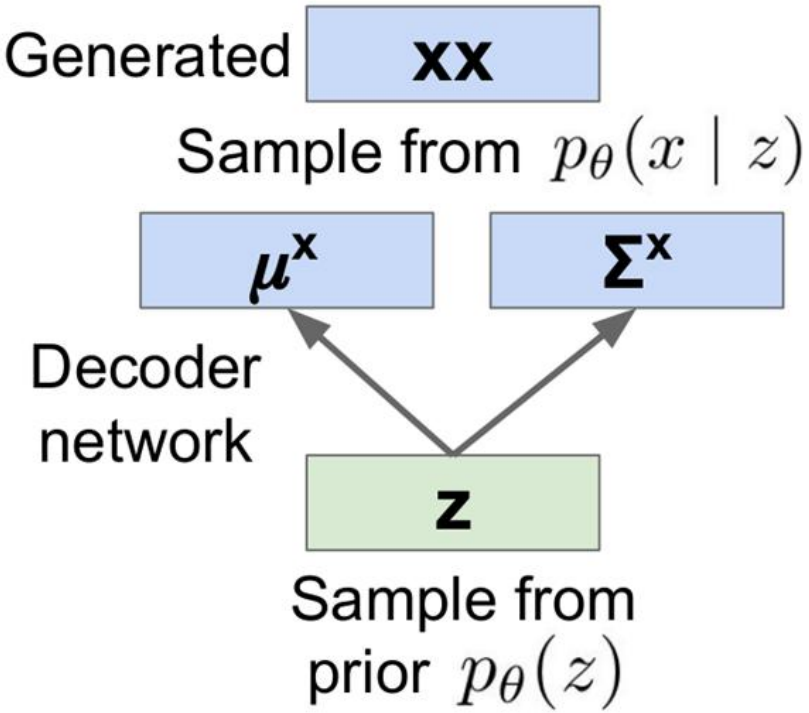
After network is trained:

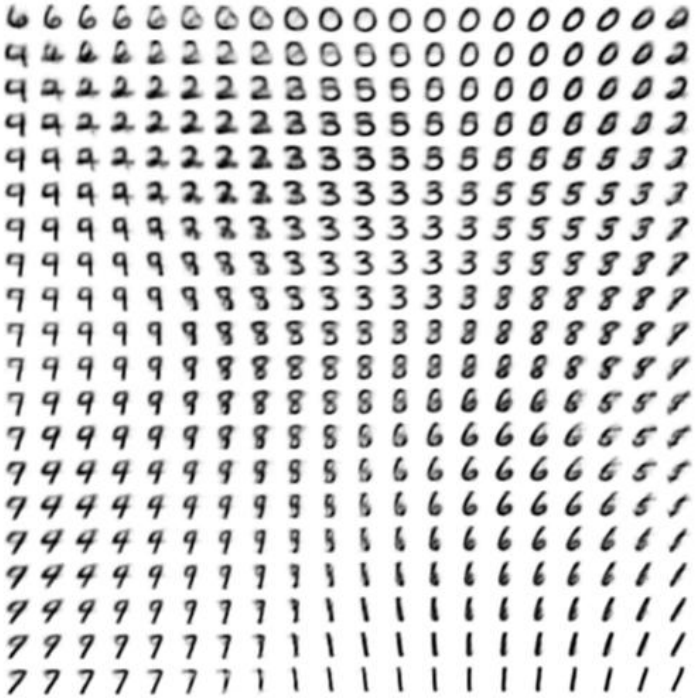




Variational Autoencoder: Generate Data!

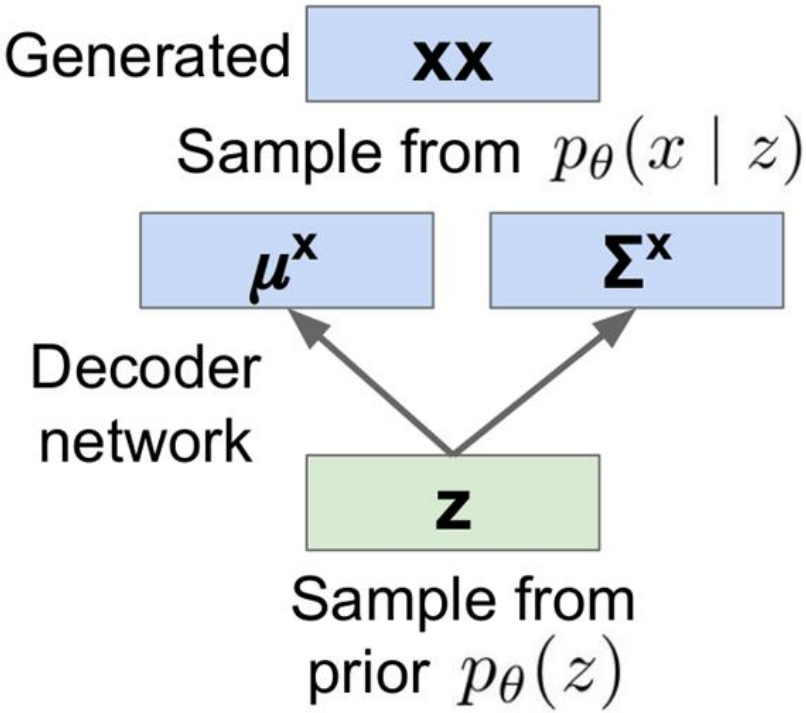
After network is trained:



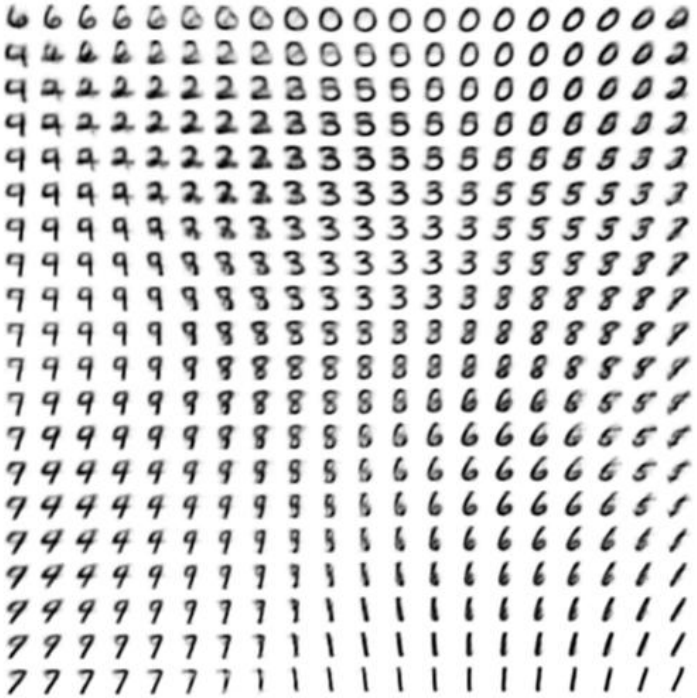


Variational Autoencoder: Generate Data!

After network is trained:



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



Variational Autoencoder: Math
Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math
Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

Maximize log-likelihood instead because sums are nicer

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math
Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$

Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^{N}$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$

Maximize log-likelihood instead because sums are nicer

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}, z) dz$$

Marginalize joint distribution

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math
Maximum Likelihood?

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta(x^{(i)})$$ Maximize likelihood of dataset $\{x^{(i)}\}_{i=1}^N$

$$= \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x^{(i)})$$ Maximize log-likelihood instead because sums are nicer

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}, z)dz = \int p_\theta(x^{(i)} \mid z)p_\theta(z)dz$$ Intractible integral =(

Variational Autoencoder: Math

$\log p_\theta(x^{(i)})$

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

"Elbow"

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Reconstruct the input data

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

Latent states should follow the prior

Reconstruct the input data

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

$\mathcal{L}(x^{(i)}, \theta, \phi)$ **"Elbow"**

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Latent states should follow the prior

Reconstruct the input data

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

Sampling with reparam. trick (see paper)

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

"Elbow"

Reconstruct the input data

Sampling with reparam. trick (see paper)

Latent states should follow the prior

Everything is Gaussian, closed form solution!

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

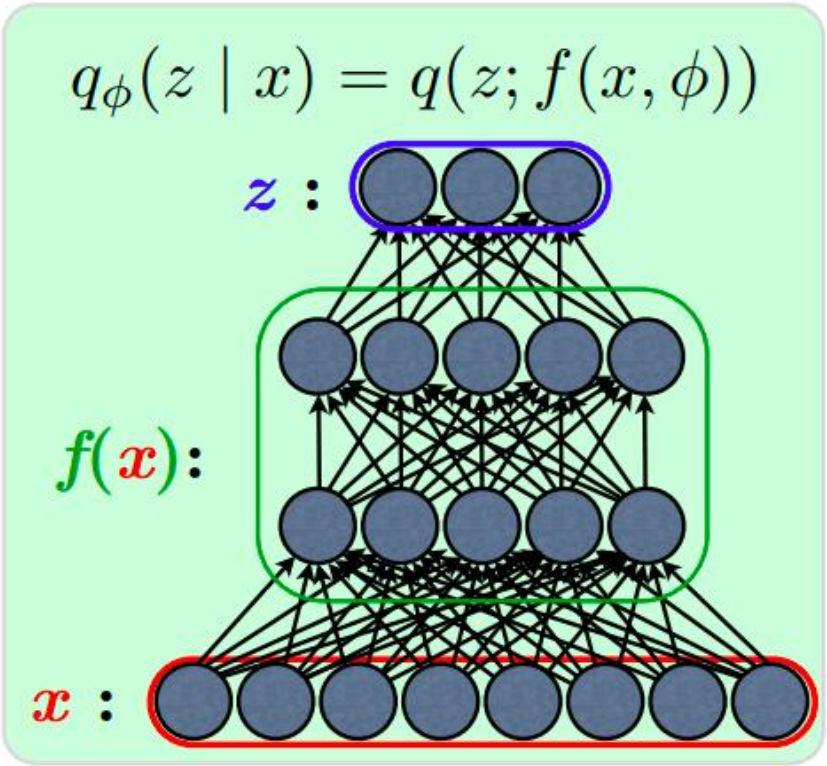
Training: Maximize lower bound

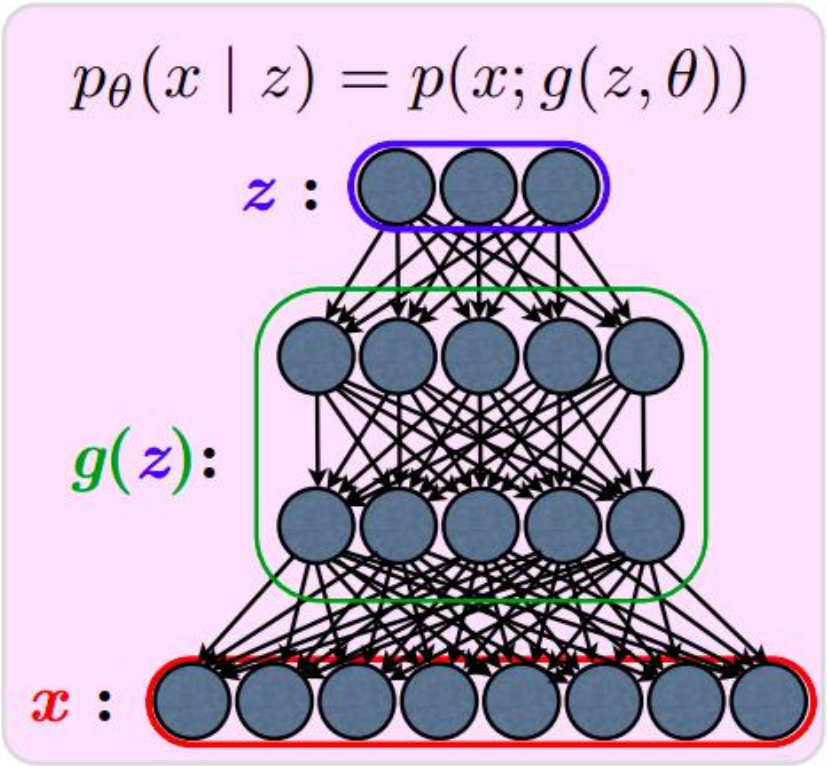
VAE Inference model

- The **VAE approach**: introduce an inference model $q_\phi(z \mid x)$ that **learns** to approximates the intractable posterior $p_\theta(z \mid x)$ by optimizing the variational lower bound:

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

- We parameterize $q_\phi(z \mid x)$ with another neural network:





Variational Autoencoder: How to train?

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

- $z \sim q_\phi(z|x)$: need to differentiate through the sampling process; how to update ϕ? (encoder is probabilistic)

Variational Autoencoder: How to train?

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\right]$$
$$= -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

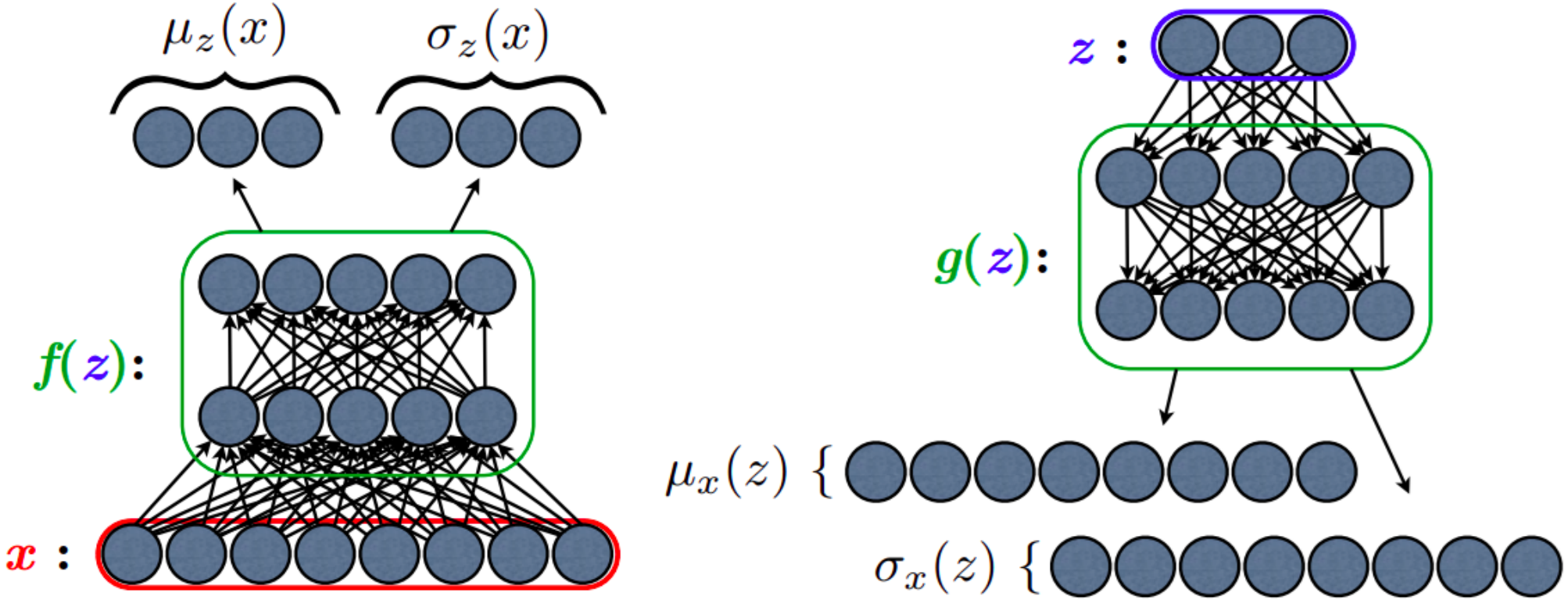
- $z \sim q_\phi(z|x)$: need to differentiate through the sampling process; how to update ϕ? (encoder is probabilistic)

Solution: Make the randomness independent of the encoder output, thus making the encoder deterministic.

How?

Reparametrization trick

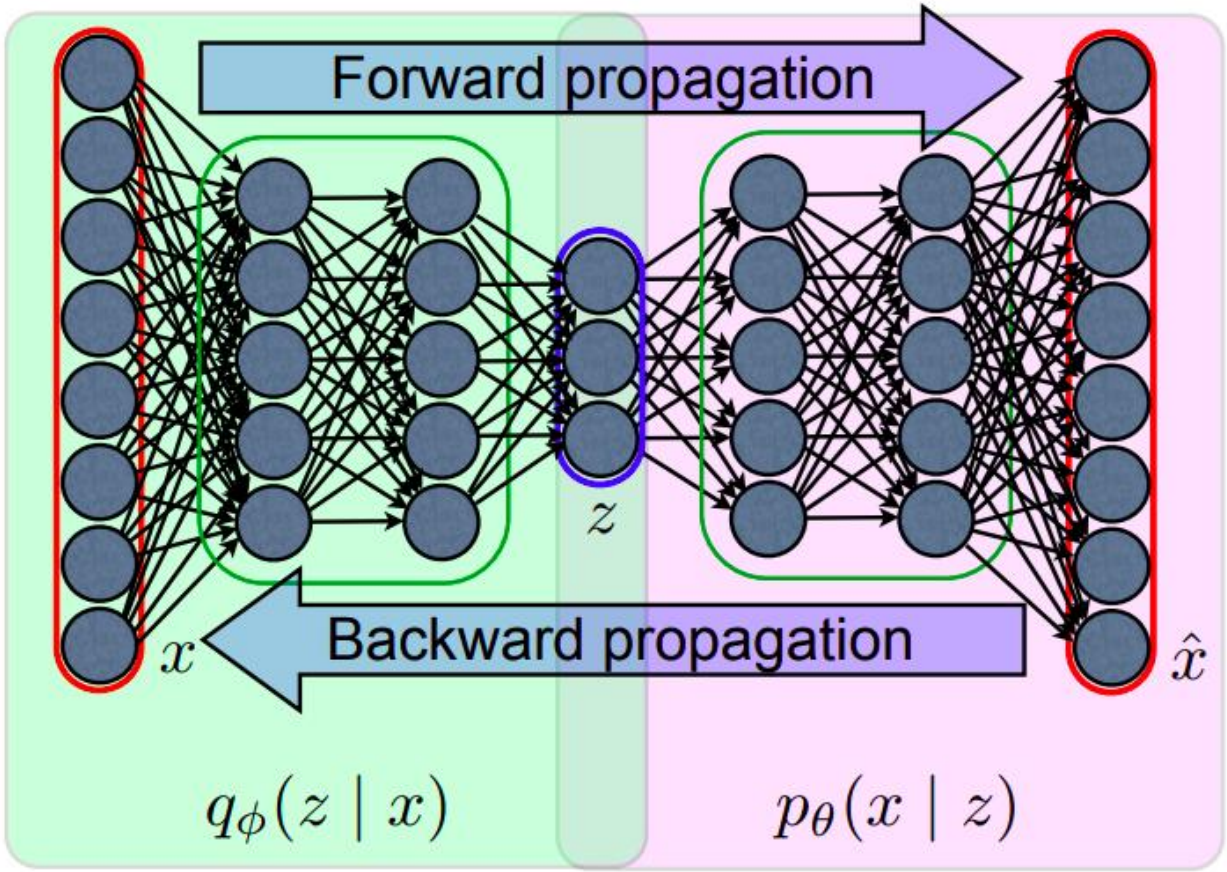
- Adding a few details + one really important trick
- Let's consider z to be real and $q_\phi(z \mid x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x))$
- Parametrize z as $z = \mu_z(x) + \sigma_z(x)\epsilon_z$ where $\epsilon_z = \mathcal{N}(0, 1)$
- (optional) Parametrize x a $x = \mu_x(z) + \sigma_x(z)\epsilon_x$ where $\epsilon_x = \mathcal{N}(0, 1)$



Training with backpropagation!

- Due to a **reparametrization** trick, we can simultaneously train both the **generative model** $p_\theta(x \mid z)$ and the **inference model** $q_\phi(z \mid x)$ by optimizing the variational bound using gradient **backpropagation**.

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



Autoencoder Overview

- Traditional Autoencoders
  - Try to reconstruct input
  - Used to learn features, initialize supervised model
  - Not used much anymore
- Variational Autoencoders
  - Bayesian meets deep learning
  - Sample from model to generate images