CS286 AI for Science and Engineering

Lecture 10: Autoencoders

Jie Zheng (郑杰)

PhD, Associate Professor

School of Information Science and Technology (SIST), ShanghaiTech University

Fall, 2020

Outline

- Efficient Data Representations
- Autoencoder and PCA
- Stacked Autoencoders
- Denoising Autoencoders
- Sparse Autoencoders
- Variational Autoencoders

Efficient Data Representations

Efficient Data Representations

Question: Which sequence is easier to remember ?

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68 → The shorter sequence seems easier ?
- 50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20 ← What if there are some rules that we can follow ?

Efficient Data Representations

Question: Which sequence is easier to remember ?

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
- 50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20

If you can remember the start of sequence 50, the length of sequence and two rules, then it is easy to reconstruct the whole sequence

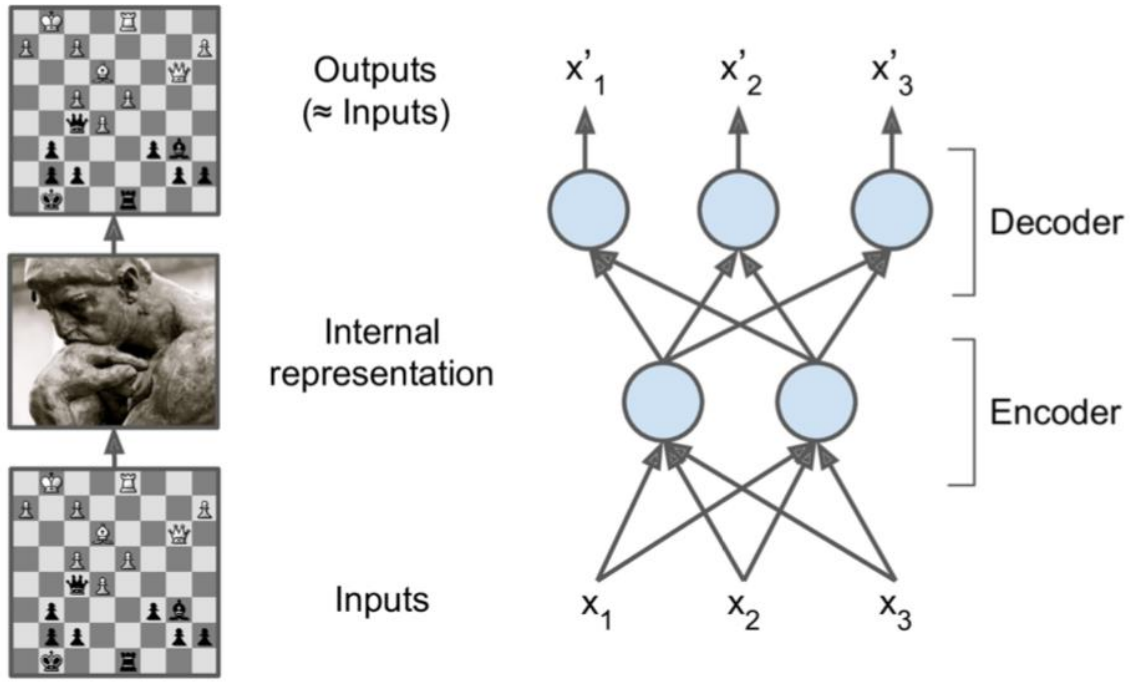
Chess memory experiment

Professional chess players can memorize positions of all pieces on a chessboard in 5 seconds. How can they do that?

Step 1: Look at the chessboard

Step 2: Analyze (encode) the chessboard

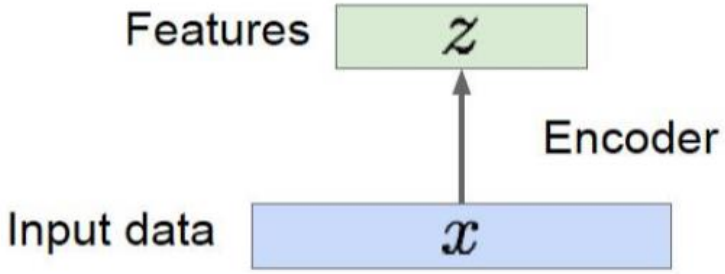
Step 3: Recover (decode) the chessboard



Internal representation has a lower dimensionality than the input data

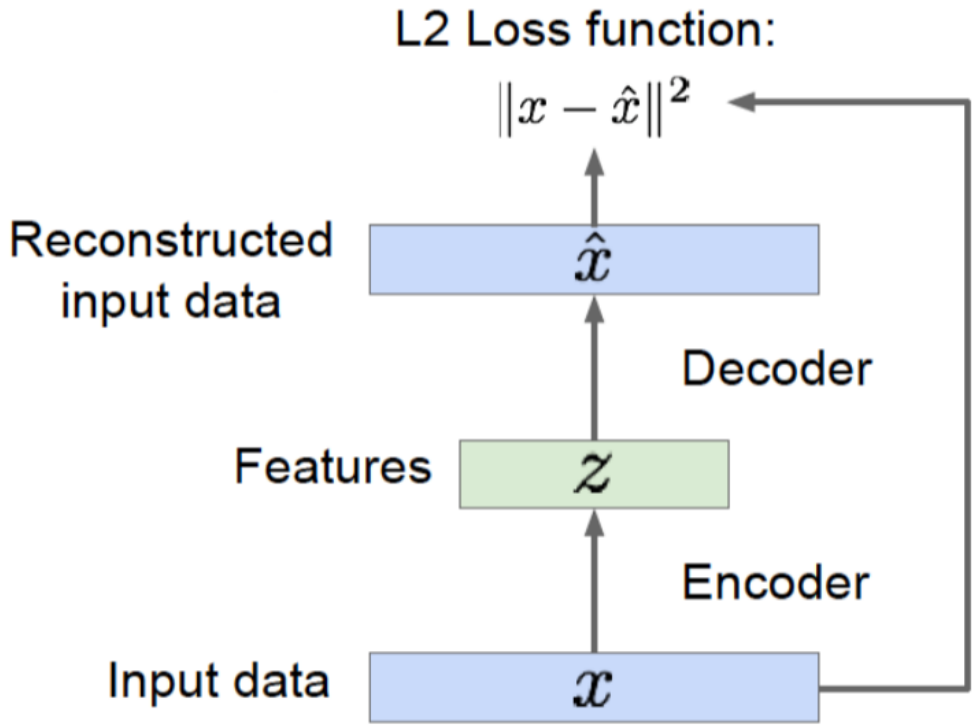
Autoencoders

- The above two examples show that, by finding internal representations with lower dimensionality than the input data, we can find important patterns in the data
- **Autoencoder (AE)** is an unsupervised neural network for learning a lower-dimensional feature representation from unlabeled training data
  - It tries to reconstruct the input with lower dimensions (less neurons)



Training an Autoencoder

- How to train an autoencoder ?
  - x : input data
  - z : internal representation / feature
  - $\hat{x}$: reconstructed data
  - $\|x - \hat{x}\|^2$: reconstruction error (or reconstruction loss)
- Then, an optimizer (e.g. Adam) can be used to minimize the reconstruction loss



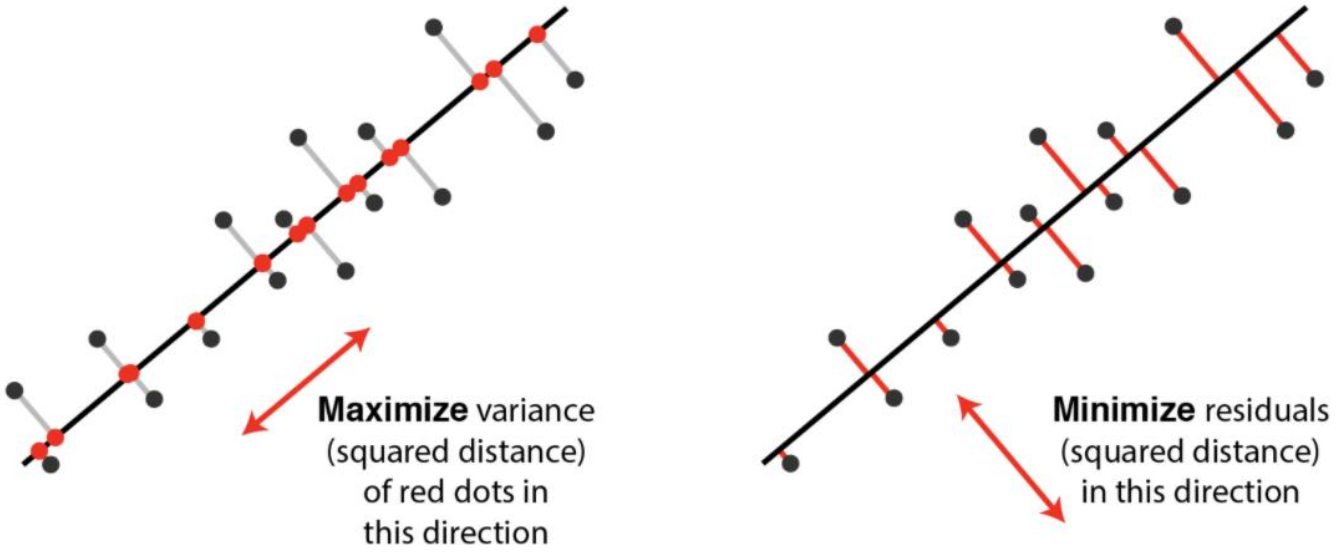
Autoencoder and PCA

PCA

- What is Principal Components Analysis (PCA)?
  - Reduce the data dimensionality by linear transformation without losing much information
- Formulation :
  - Suppose there are n samples in an m-dimension space $X = \{x_1, x_2, \dots, x_n\}$
  - Project the sample points into k dimension ($k < m$):
    - the projection matrix is $W = \{w_1, w_2, \dots, w_n\}$, and W is orthogonal
    - the projected points is $Z = WX$

PCA

- What does “without losing much information” mean ?
- There are two views :
  - A : Maximize variance
  - B : Minimize residuals



Objectives of PCA

- Objective of Maximize variance :

$$\max_{W} \ tr(X^T W^T W X) \qquad s.t. W^T W = I$$

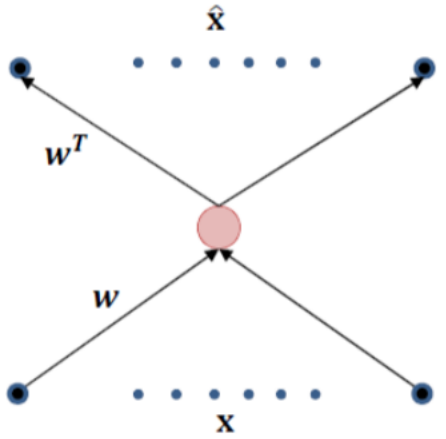
- Objective of Minimize residuals:

$$\min_{W} \ \left\| X - W^T W X \right\|_F^2 \qquad s.t. \ \ W^T W = I$$

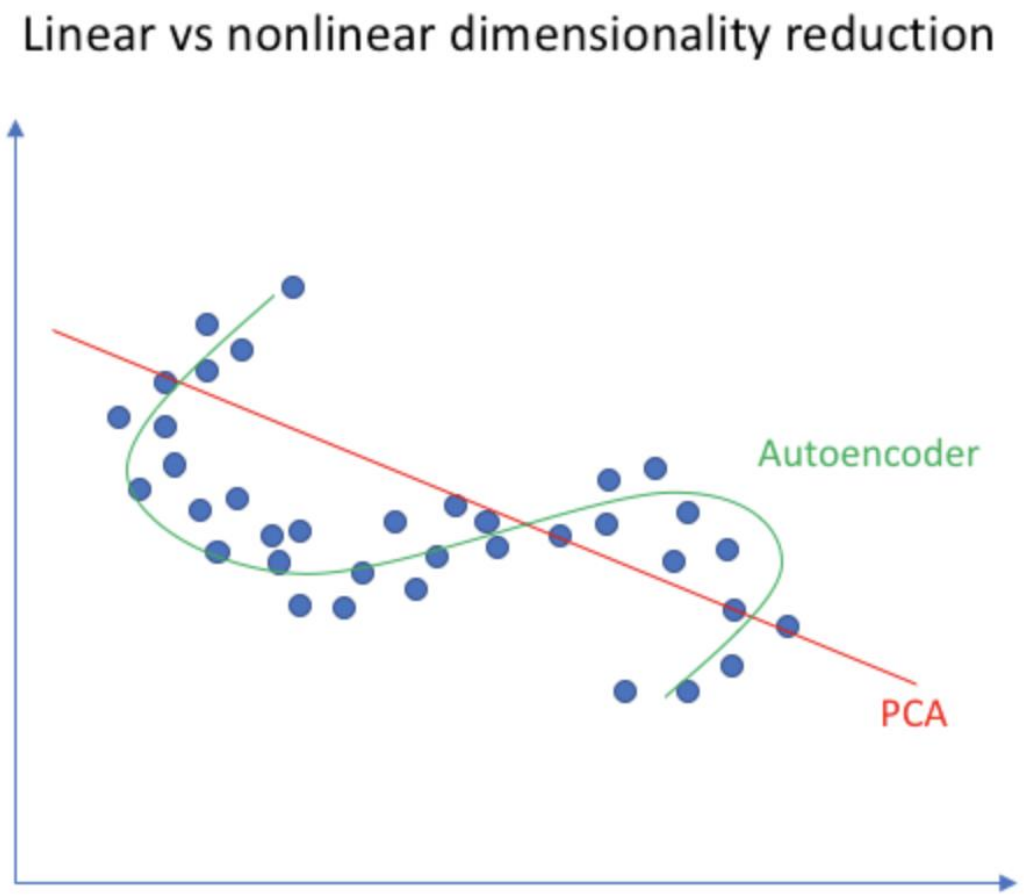
Results of the above two objectives are equivalent

Relation between AE and PCA

- What if the encoding of AE is linear ?
  - Encoding process becomes a linear projection $W \in \mathbb{R}^{n \times m}$ from a high-dimensional space $x \in \mathbb{R}^n$ to a low-dimensional space $z = Wx$
  - Decoding process becomes $\hat{x} = W^T W x$
  - Reconstruction error is $\|x - W^T W x\|^2$, the same as PCA
  - Therefore, PCA is a special (linear) form of AE

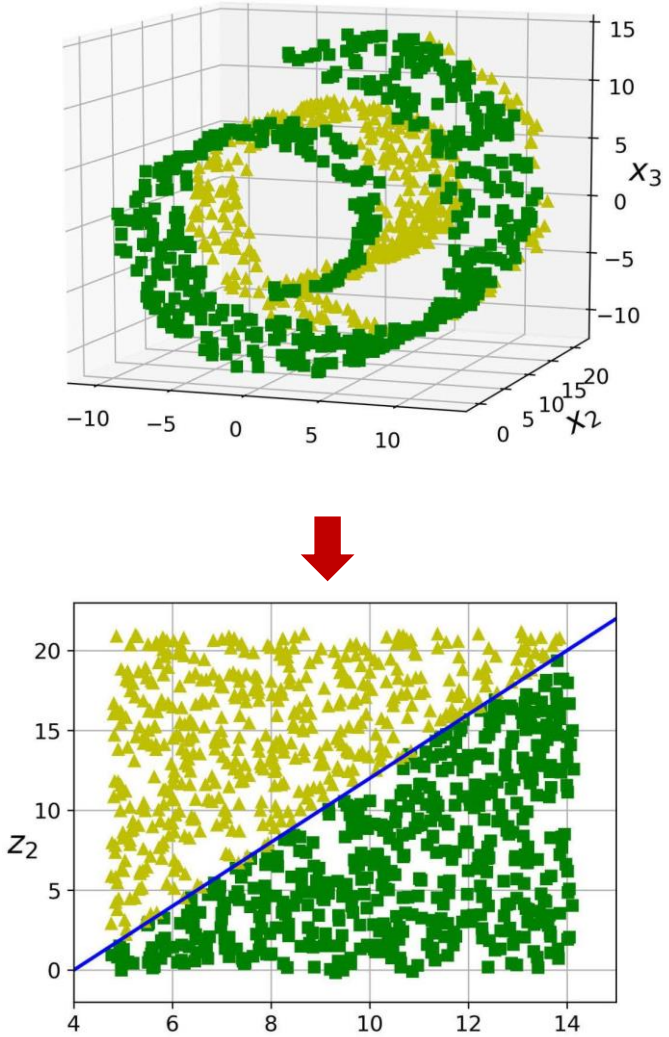


Relation between AE and PCA



Manifold learning

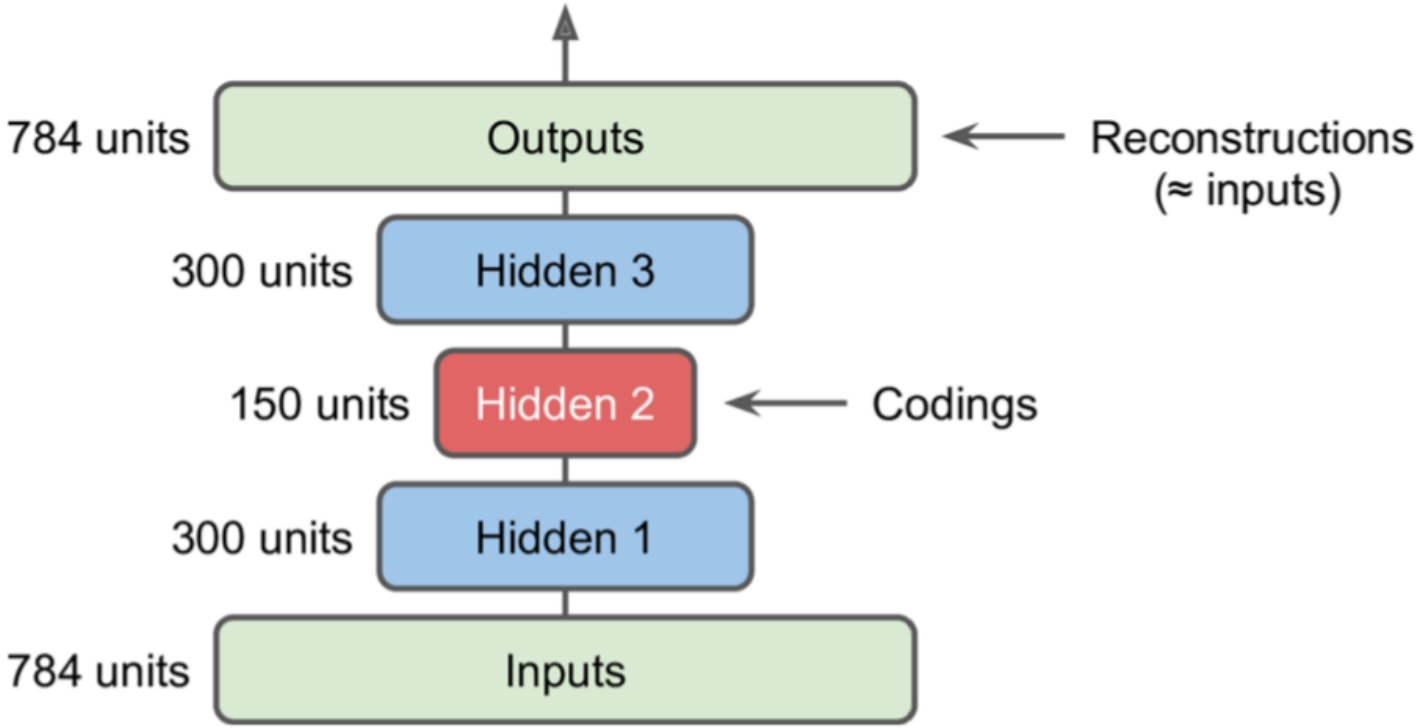
- **Manifold learning** is to discover the "hidden" structure (i.e. manifold) in a high-dimensional space through nonlinear dimensionality reduction
- Many approaches:
  - Self-organizing map (Kohonen map/netweork)
  - Autoencoders
  - Principles curves & manifold: Extension of PCA
  - Kernel PCA, Nonlinear PCA
  - Curvilinear Component Analysis
  - Isomap: Floyd-Marshall + Multidimensional scaling
  - Data-driven high-dimensional scaling
  - Locally-linear embedding
  - …



Stacked Autoencoder

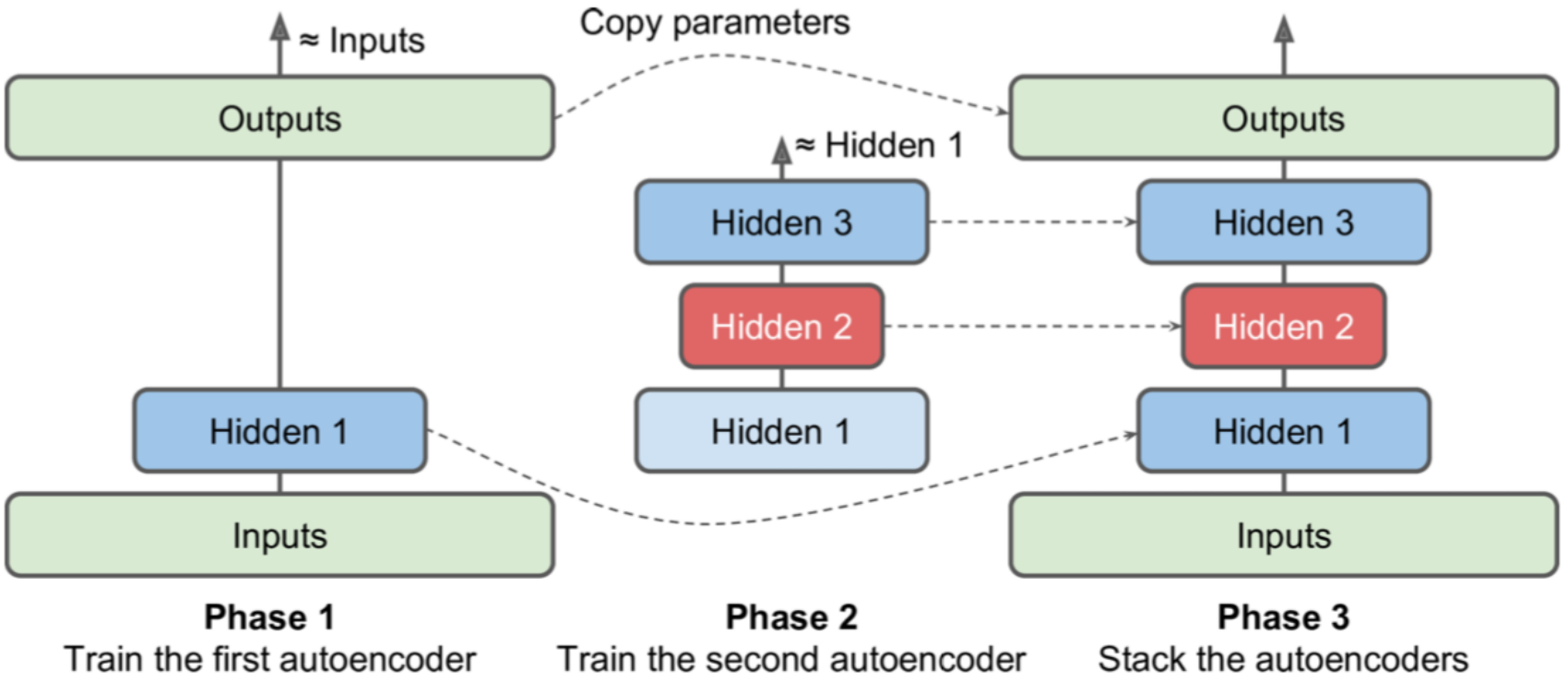
What is a stacked (or deep) autoencoder?

- More hidden layers than standard Autoencoders



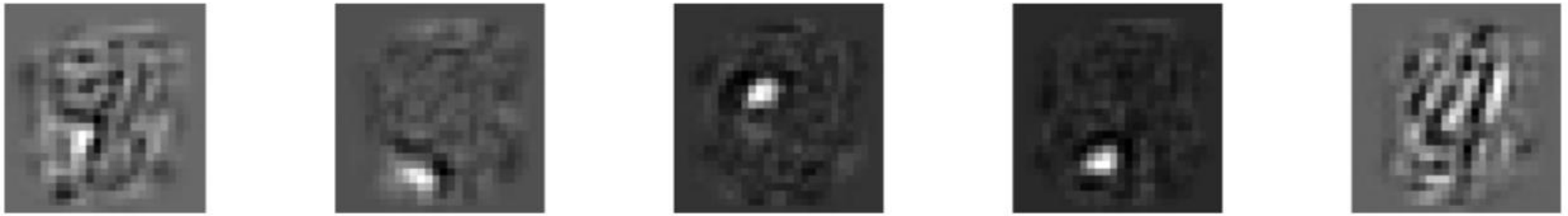
How to train a stacked AE?

- Train one AE at a time
- Outer layer’s output is used as the inner layer’s input



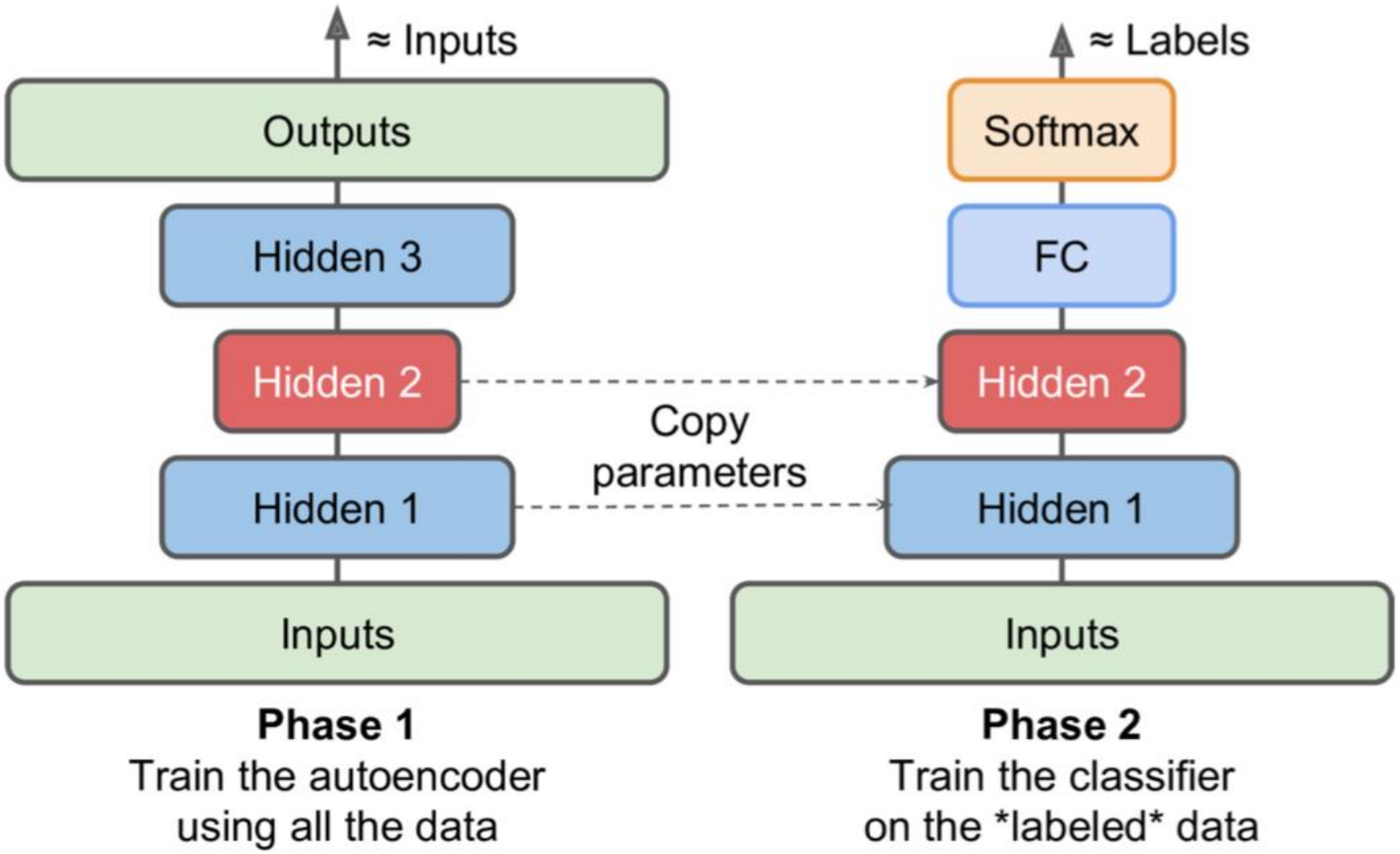
Advantages of stacked AEs

- More powerful for feature representation
  - Previous layers can extract low-level features
  - Deep (inner) layers can learn high-level features



Features learned by 5 neurons from the first hidden layer

Unsupervised pretraining using stacked AEs

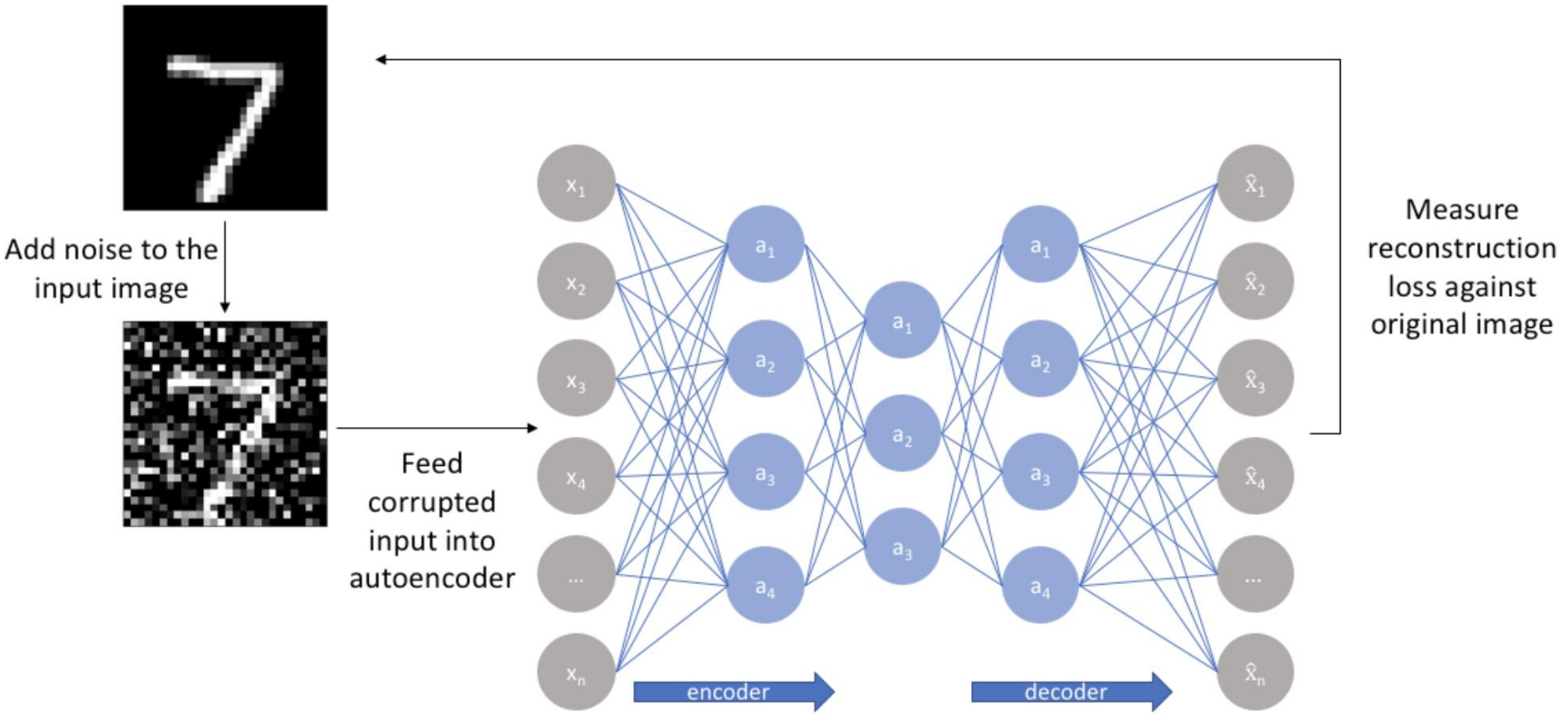


Denoising Autoencoders

Denoising Autoencoder (DAE)

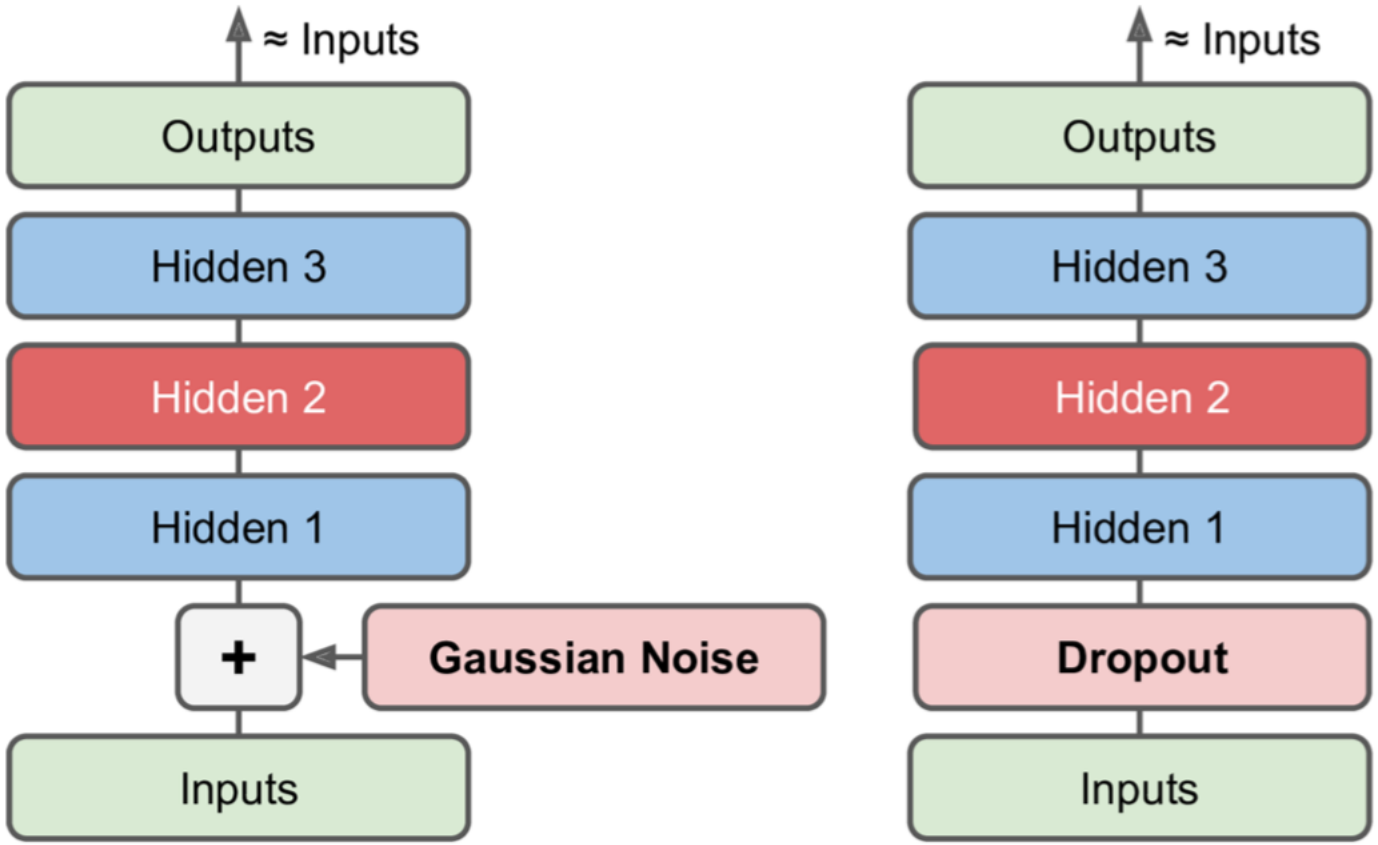
- What is DAE ?
  - Randomly corrupt some of the inputs
  - Train the autoencoder to reconstruct the input from a corrupted version of it
  - The autoencoder is forced to predict the original inputs
  - This requires to capture the joint distribution between a set of variables
- **Rationale**: To increase the difficulty of reconstructing the output by adding noise

Structure of DAE

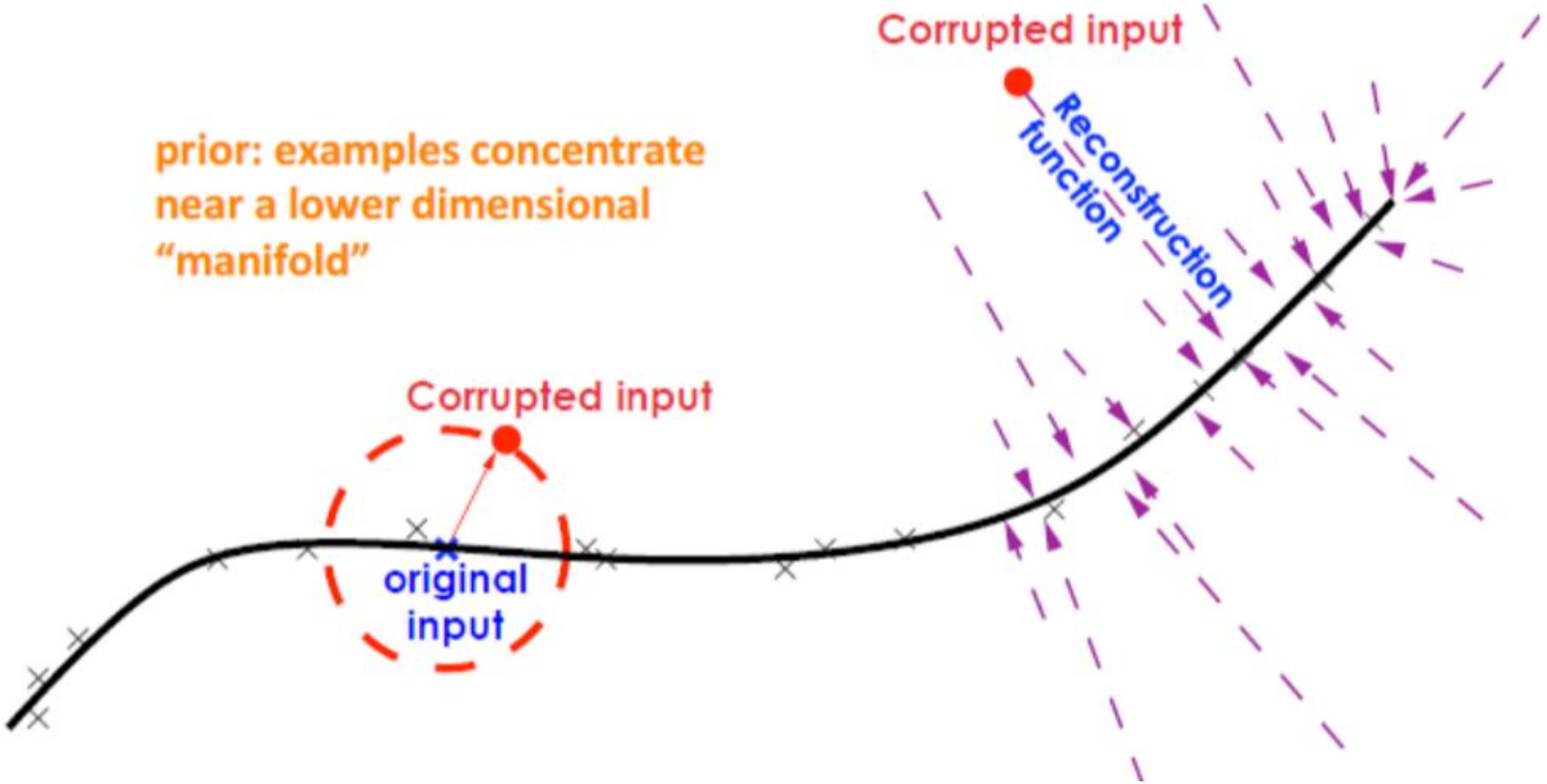


Gaussian noise vs. dropout

- Either Gaussian noise or dropout can be used to corrupt the input



Geometric view of DAE

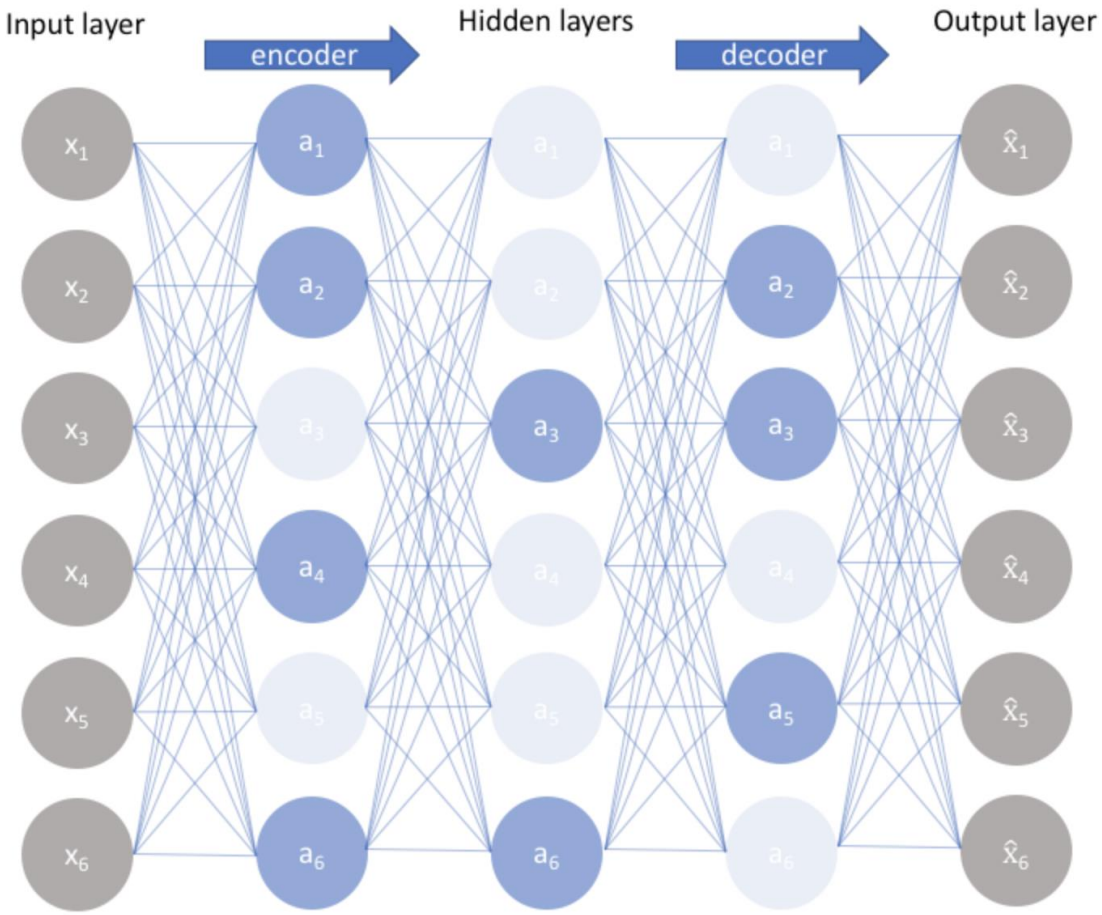


Sparse Autoencoder

Sparse Autoencoder

- Usually, m the dimension of hidden layers in AE is small

- What if the dimension of hidden layers k in AE is large ?
  - We call it over-completeness
  - This can be solved by enforcing sparsity



Sparse Autoencoder

- **Rationale**: To limit the amount of total activation for a neuron throughout training

$$\hat{p}_j = \frac{1}{n}\sum_{i=1}^{m}\left[a_j(x_i)\right]$$

  - a_j denotes the activation of hidden unit j in the autoencoder
  - $a_j(x)$ denotes the activation of the hidden unit when the network is given a specific input x
  - $\hat{p}_j$ as the average activation of hidden unit j
- We enforce a sparsity constraint that $\hat{p}_j = p_0$
  - p_0 is small, usually set as 0.05

Sparse Autoencoder (continued)

- How to enforce the sparsity?
- Use **Kullback-Leibler (KL) divergence** to measure the divergence between:
  - the target probability that a neuron will be activated and
  - the actual probability over the training batch:

$$\sum_j KL(p_0||\hat{p}_j) = \sum_j p_0 \log\frac{p_0}{\hat{p}_j} + (1-p_0)\log\frac{1-p_0}{1-\hat{p}_j}$$

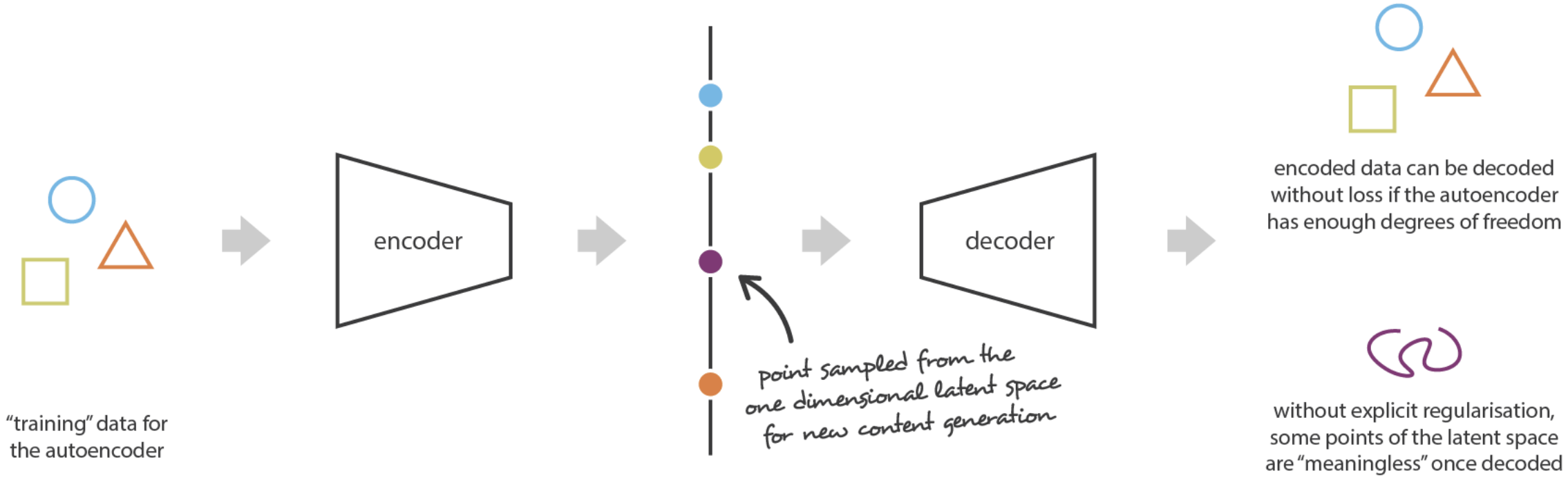
- Adding the sparsity loss, the overall loss function becomes :

$$L = \left|\left|x - \hat{x}\right|\right|^2 + \beta \sum_j KL(p_0||\hat{p}_j)$$

Variational Autoencoders

Limitation of Autoencoder

- The autoencoder is solely trained to encode and decode with as few loss as possible, regardless how the latent space is organized

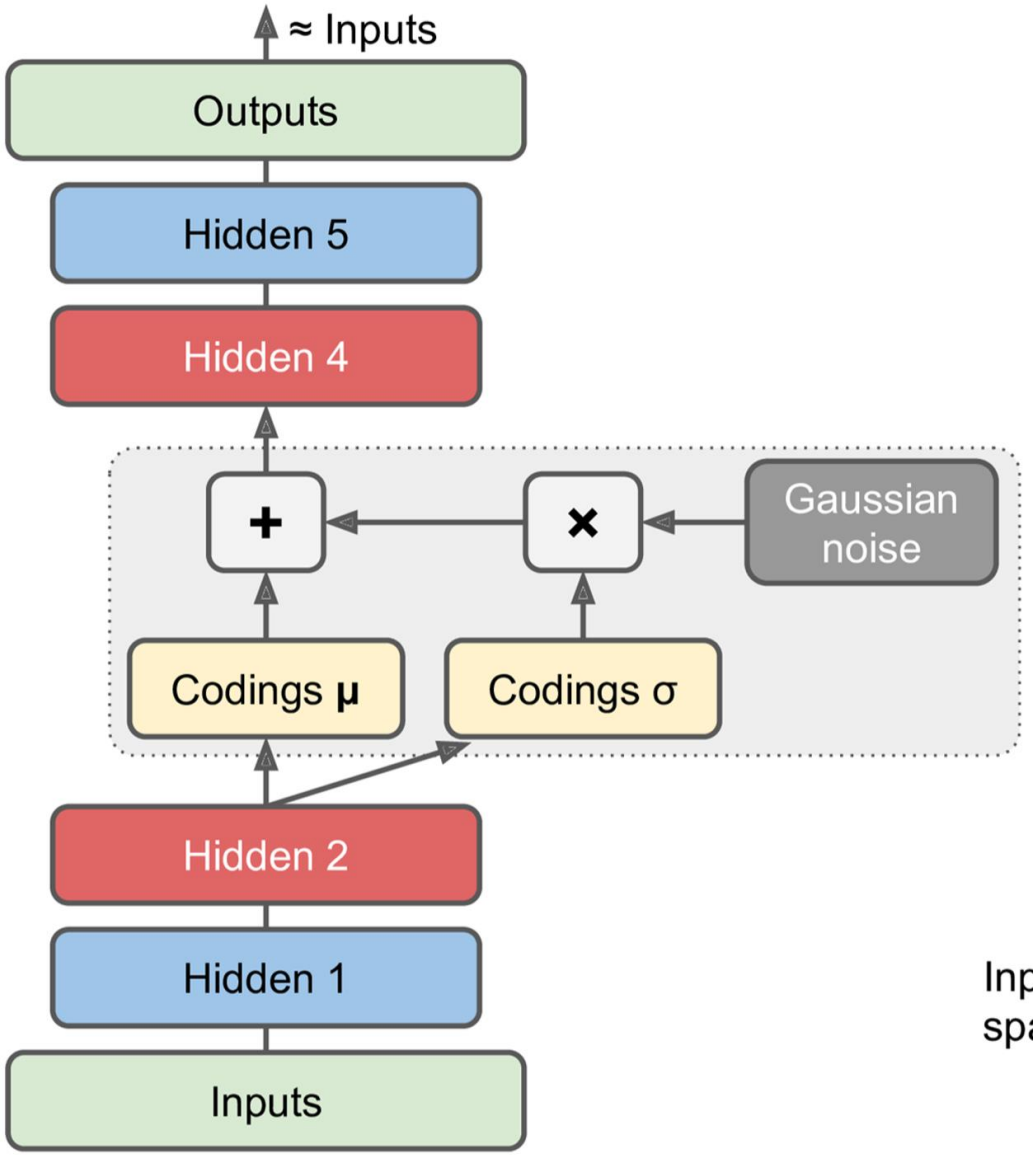


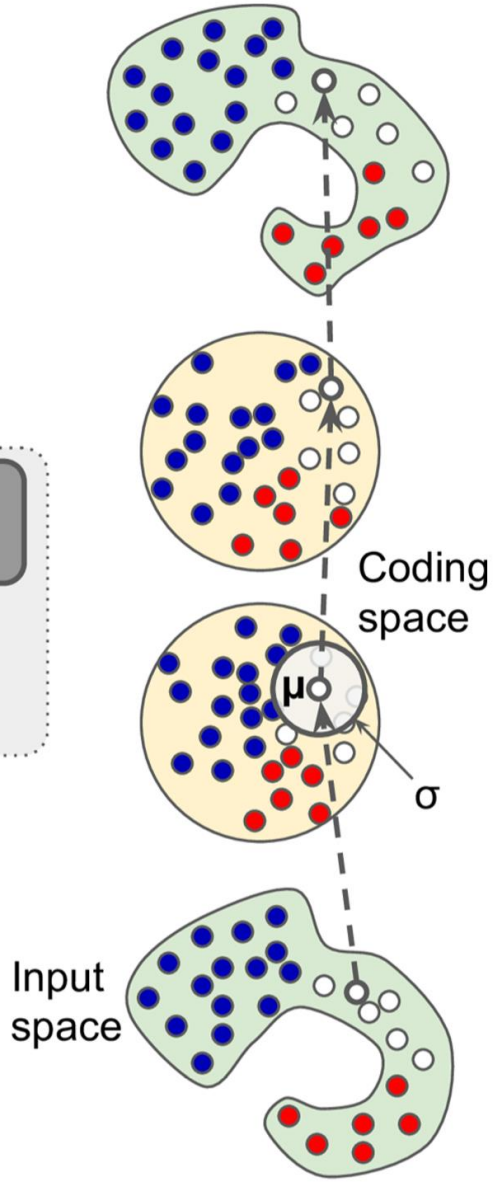
What is Variational Autoencoder (VAE)?

- A **variational autoencoder (VAE)** can be defined as being an autoencoder whose training is regularized
  - To avoid overfitting, and
  - To ensure that the latent space has good properties that enable generative process

- Instead of encoding an input as a single point, VAE encodes it as a distribution over the latent space

What is Variational Autoencoder (VAE)?



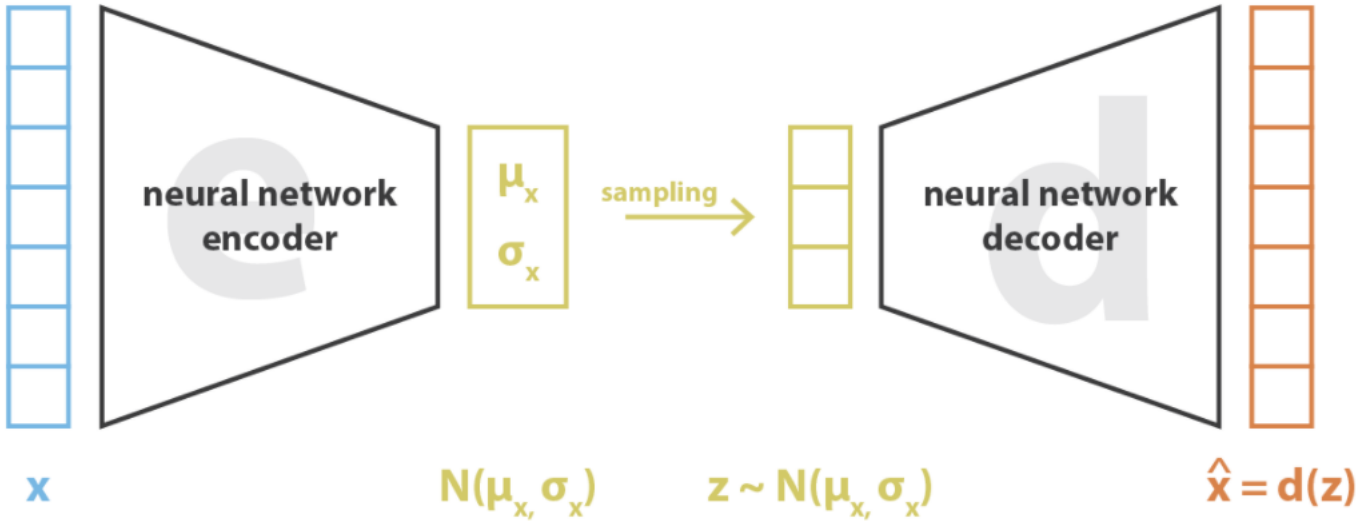


How is a VAE trained?

1. Input is encoded as distribution over the latent space
2. A point from the latent space is sampled from that distribution
3. The sampled point is decoded and the reconstruction error can be computed
4. The reconstruction error is backpropagated through the network

Distribution in the latent space

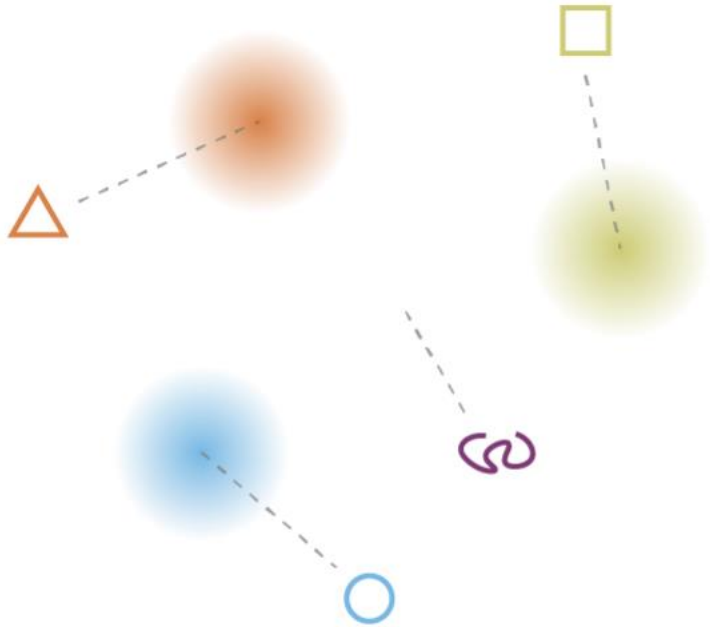
- The input is supposed to obey the Gaussian (normal) distribution
- Encoder is trained to return the mean and covariance matrix



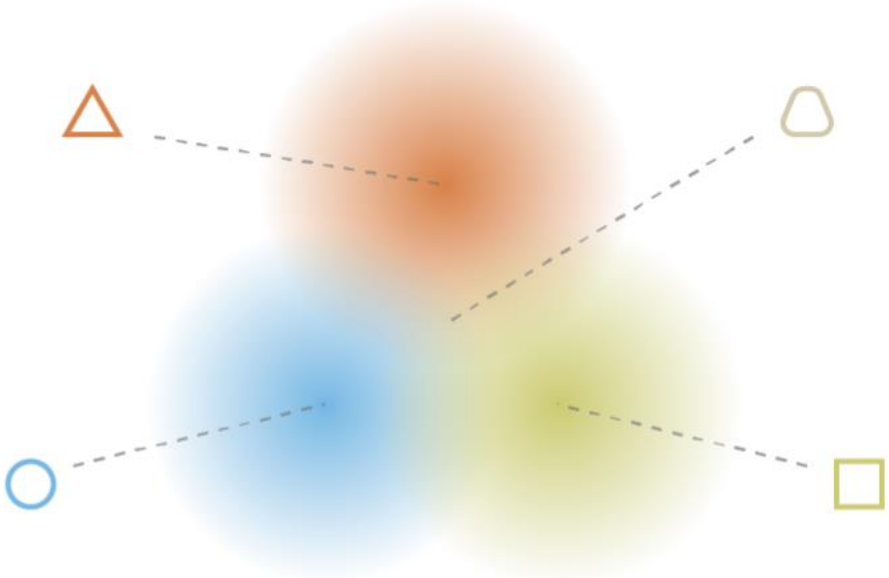
Loss function of VAE

- The loss function of VAE is composed of two terms:
  - Regularization term : To enforce the distributions to be close to a standard normal (Gaussian) distribution
    - **Continuity** : two close points in the latent space should not give two completely different contents once decoded
    - **Completeness** : for a chosen distribution, a point sampled from the latent space should give "meaningful" content once decoded
  - Reconstruction term : the same function as other AEs

Regularization view



what can happen without regularisation



what we want to obtain with regularisation

Summary

- In this lecture we have learned
  - The rationale of autoencoders with efficient data representations (i.e. to reconstruct inputs in a lower-dimensionality space)
  - Relation between PCA and autoencoders
  - Several variants of autoencoders
    - Stacked (or deep) AE
    - Denoising AE
    - Sparse AE
    - Variational AE
- References:
  - Chapter 15 of Aurélien Géron’ s book “Hands-On Machine Learning with SciKit-Learn & TensorFlow” (2017), or Chapter 17 of Edition 2 of the same book (2019)