Deep Learning Seminar

Chapter 14. Autoencoders

Keonwoo Noh

Department of Information and Communication Engineering

DGIST

2018.01.09

Chapter 14. Autoencoders

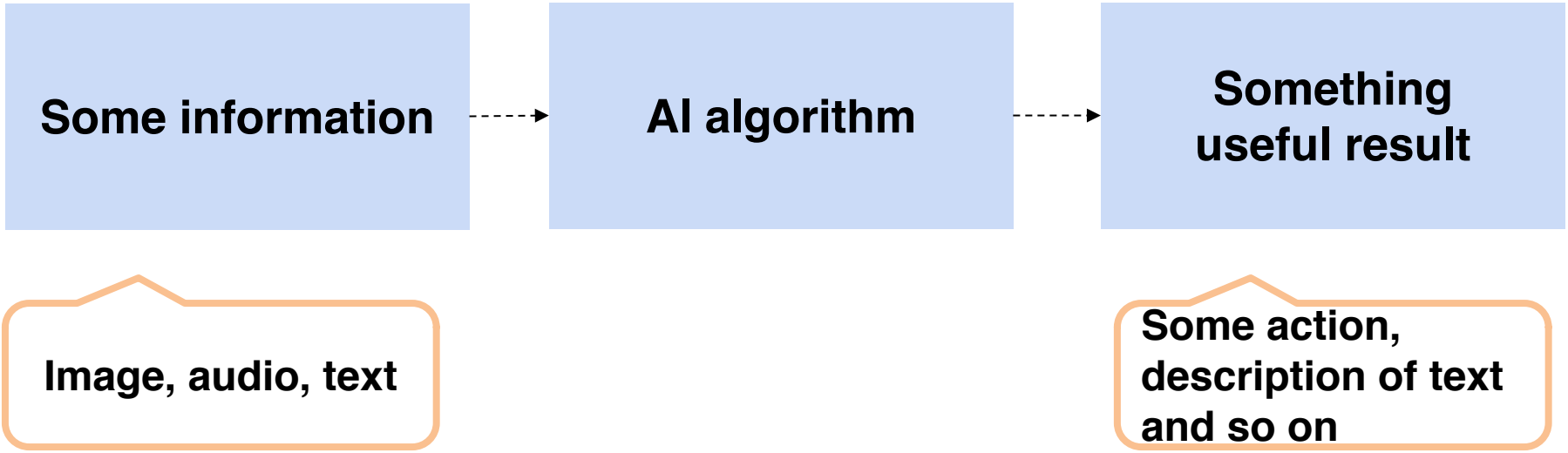
- 14.1 Undercomplete Autoencoders
- 14.2 Regularized Autoencodrs
- 14.3 Representational Power, Layer Size and Depth
- 14.4 Stochastic Encoders and Decoders
- 14.5 Denoising Autoencoders
- 14.6 Learning Manifolds with Autoencoders
- 14.7 Contractive Autoencoders
- 14.8 Predictive Sparse Decomposition
- 14.9 Applications of Autoencoders

Chapter 14. Autoencoders

- Introduction
- Stochastic Encoders and Decoders
- Regularized autoencoders
  - sparse autoencoders
  - contractive autoencoders
  - denoising autoencoders
- Representational Power, Layer Size and Depth

Introduction

Machine learning and feature



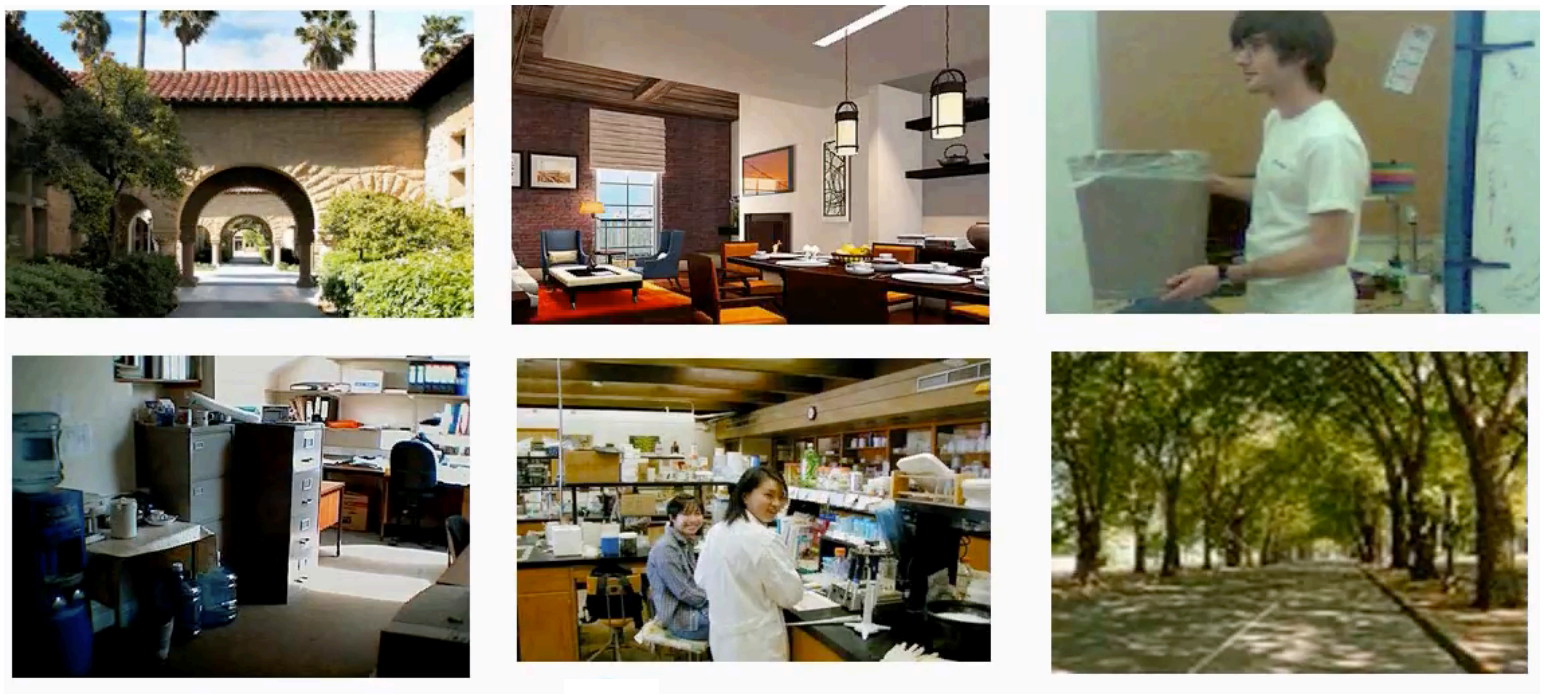
Representation of information

- Real world object can be represented as some feature in computer
- In case of human:
  - name
  - age
  - gender
  - nationality
- In case of image:
  - pixel
- In case of audio:
  - signal

image source : https://www.youtube.com/watch?v=vfnxKO2rMq4

Feature

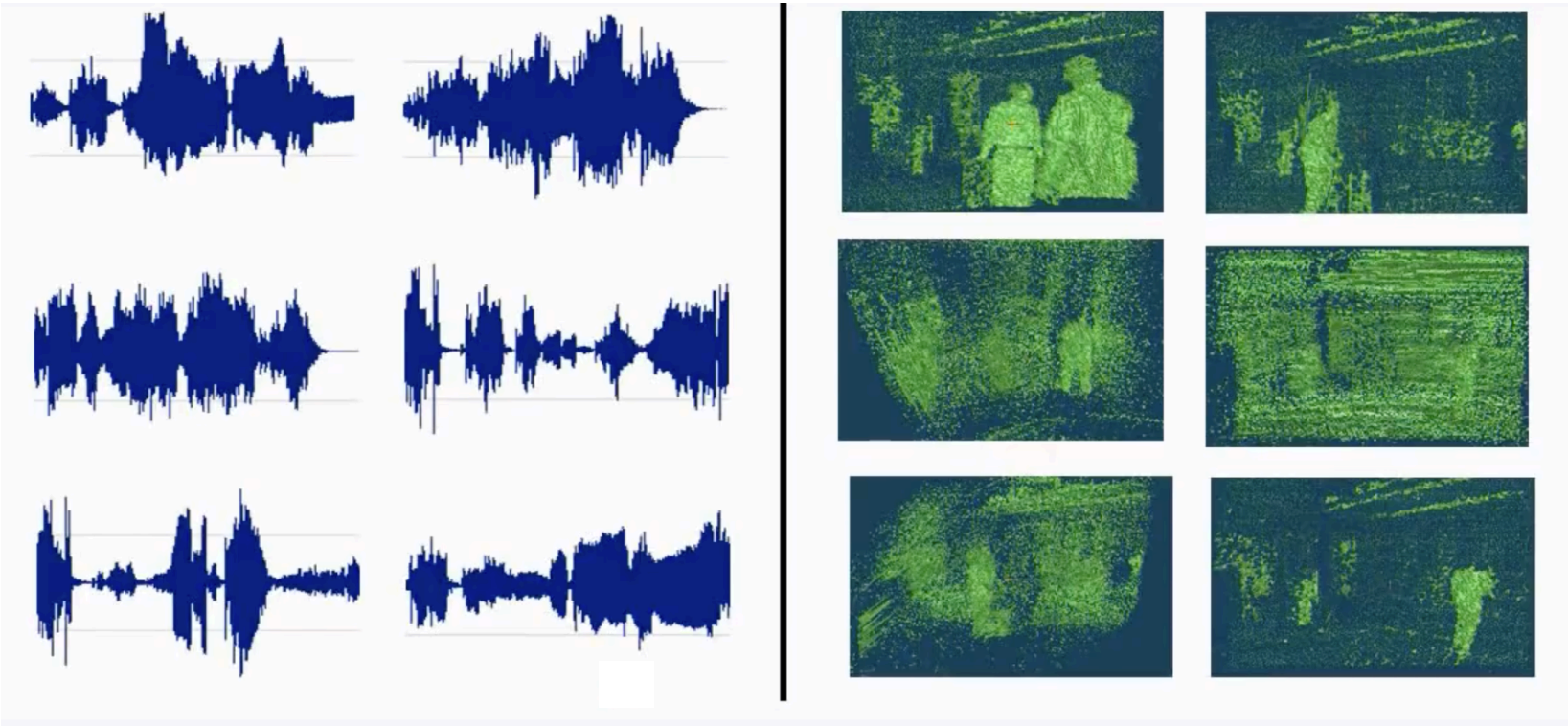
- However, we don't know which representation is better



Find a better way to represent images than pixels

Feature

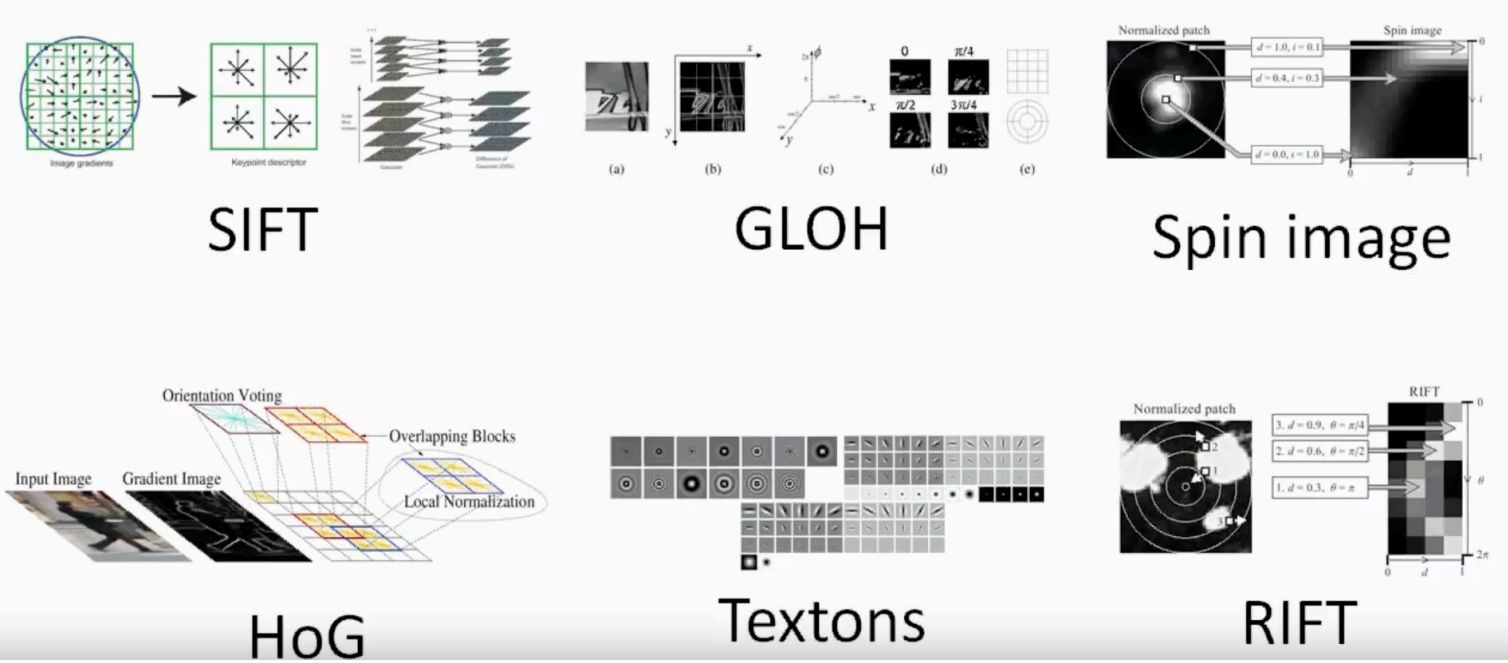
- However, we don't know which representation is better



Find a better way to represent audio, 3d range scans

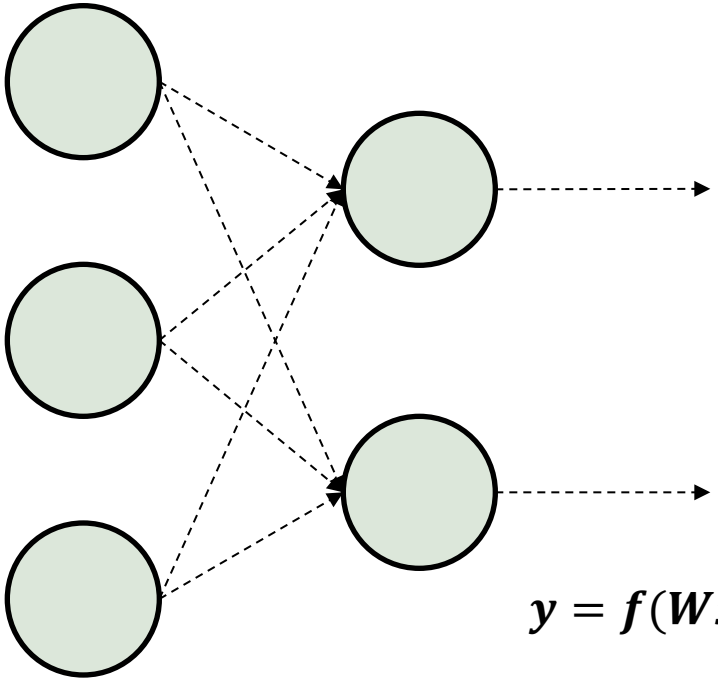
Feature

- **Thousands of Engineers have spent the years or decades to find how to represent data exists in real world**



Simple neural network model

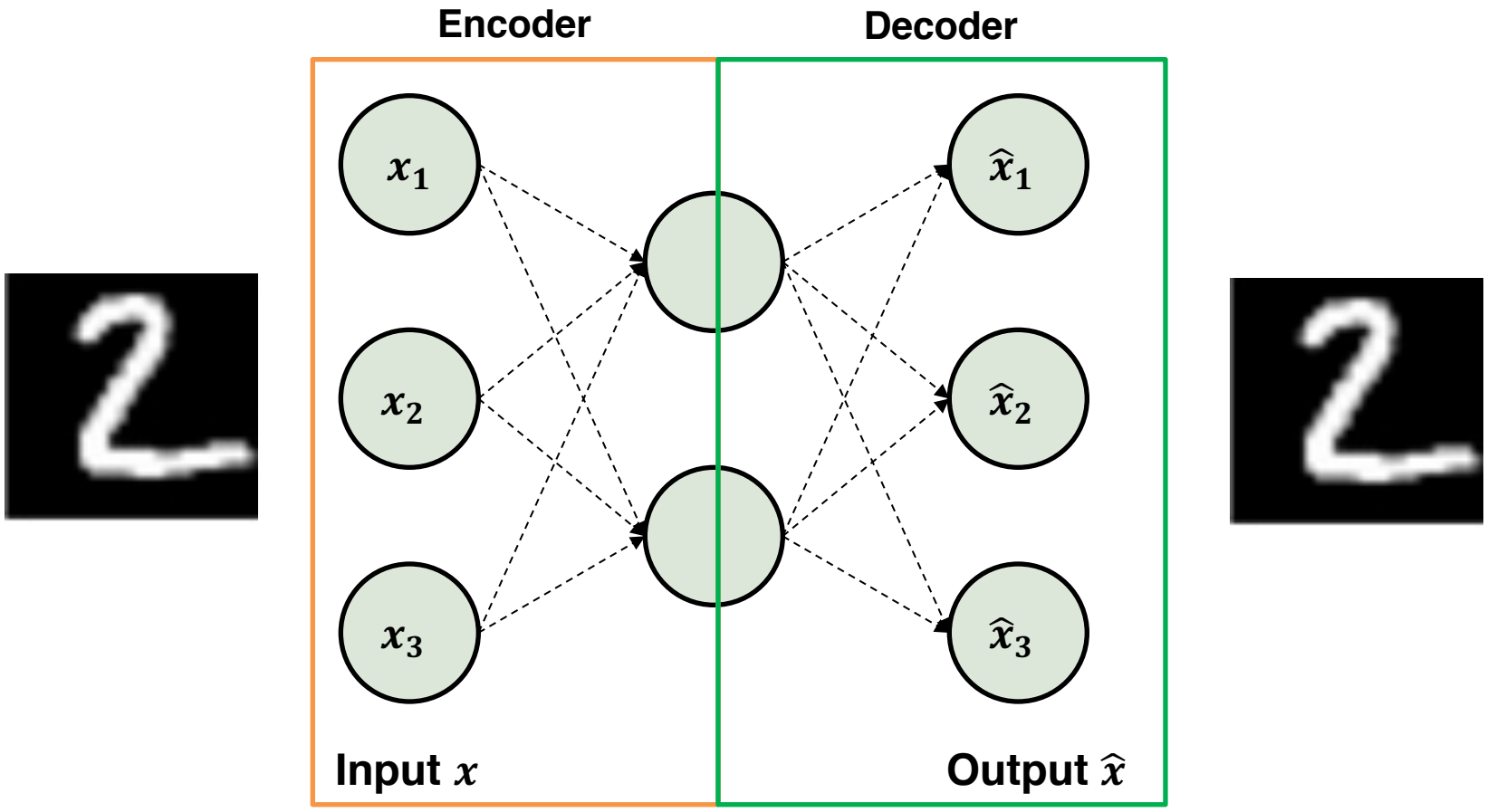
Input x



$$y = f(Wx + b)$$

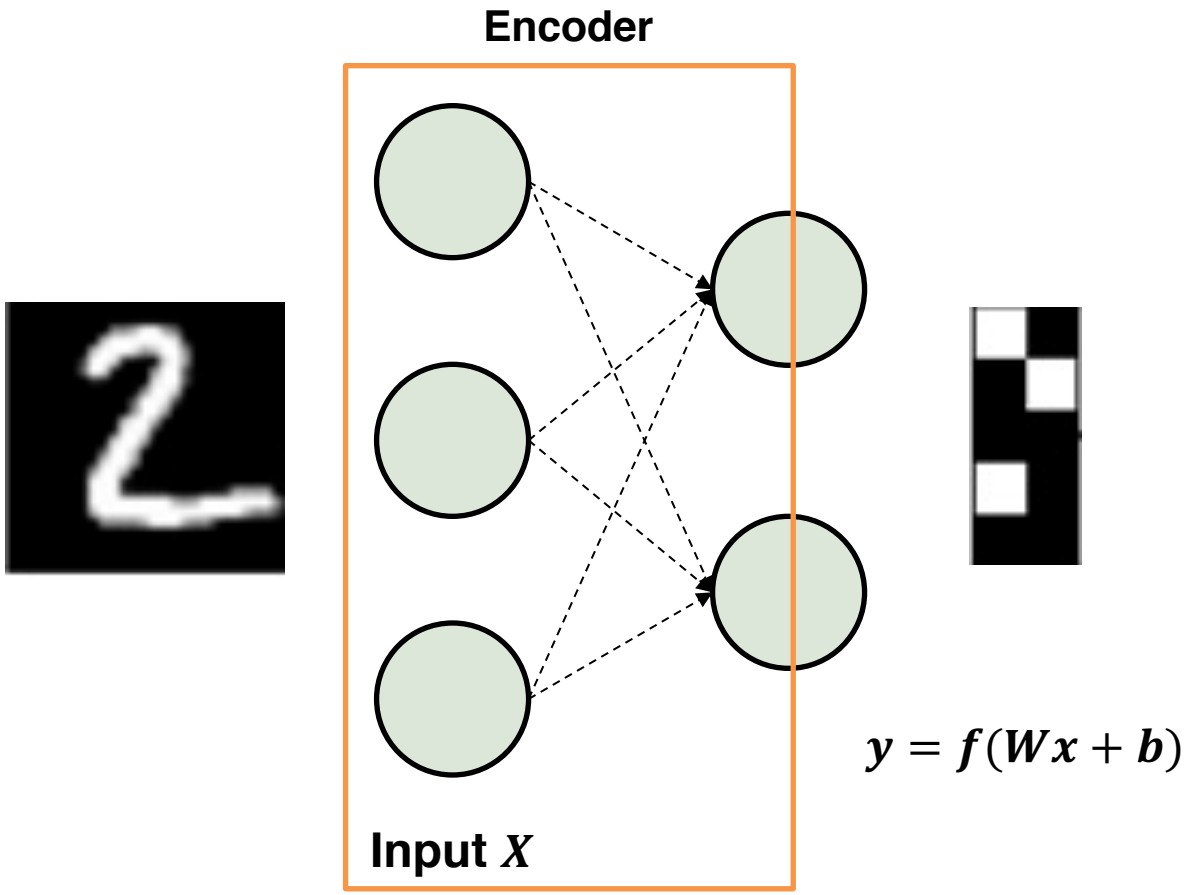
Autoencoders

- Neural net with symmetry weights



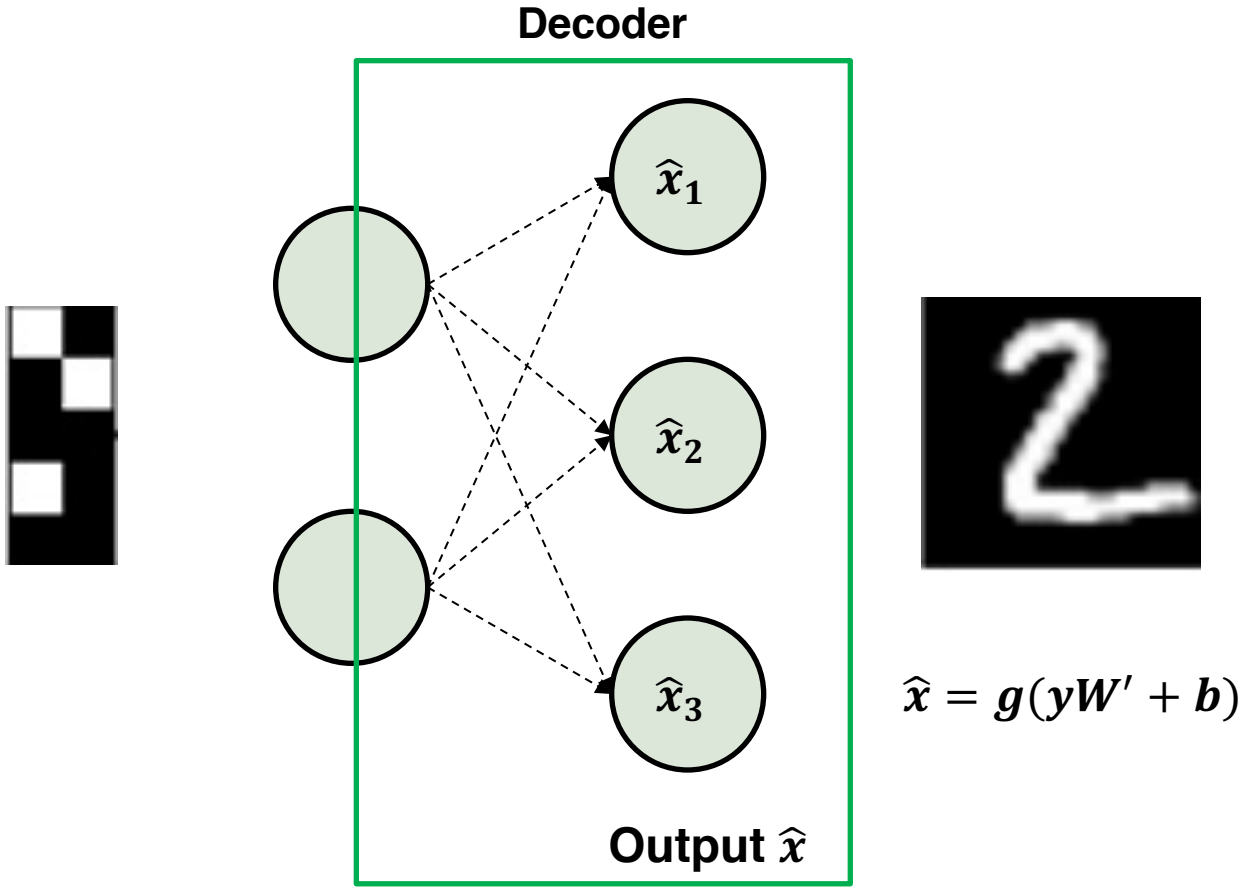
Autoencoders

- Encoder yields compression of input data



Autoencoders

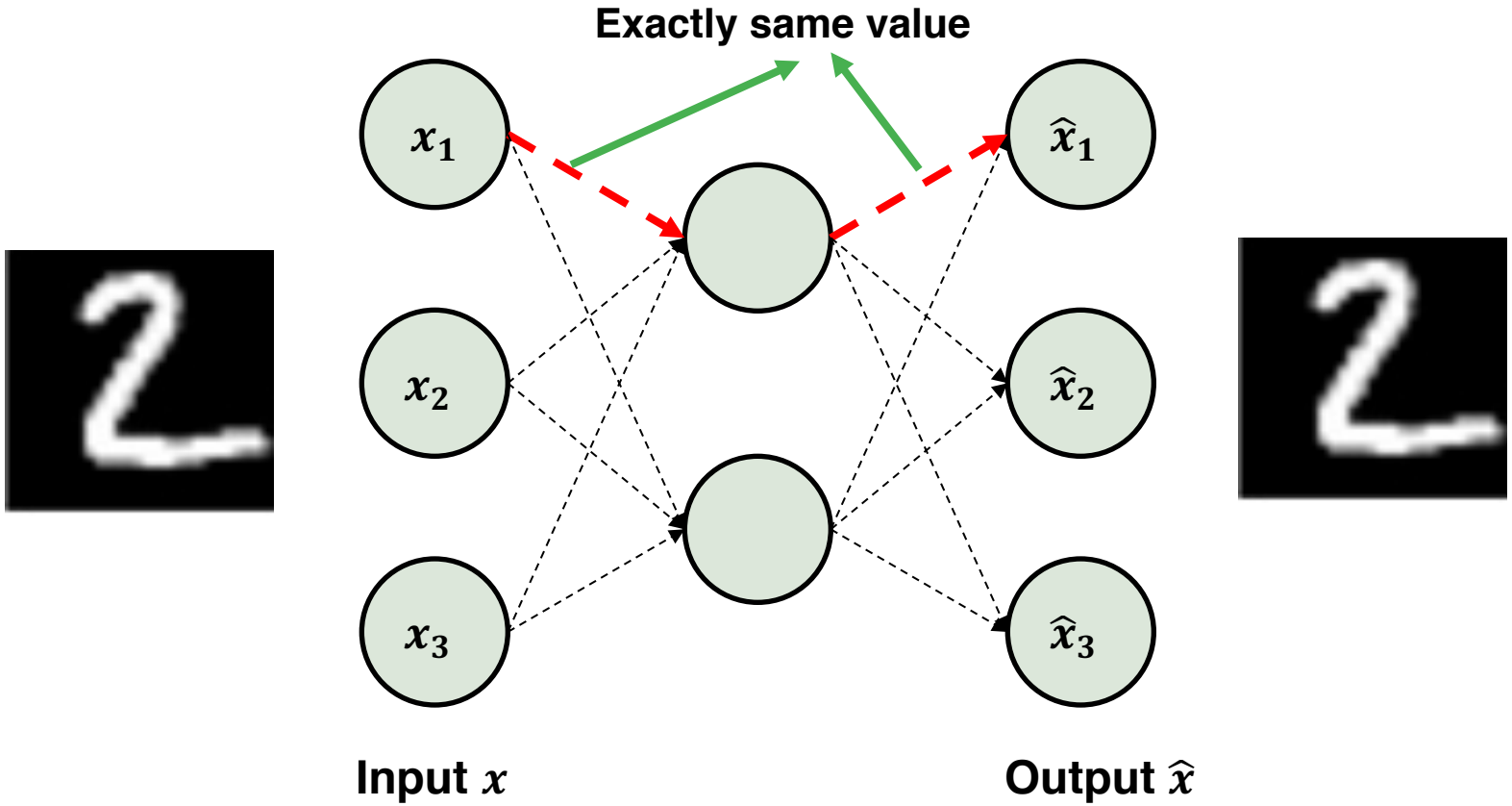
- Decoder reconstructs input data



$$\hat{x} = g(yW' + b)$$

Parameter sharing

- It is possible to design autoencoder as $W' = W^T$

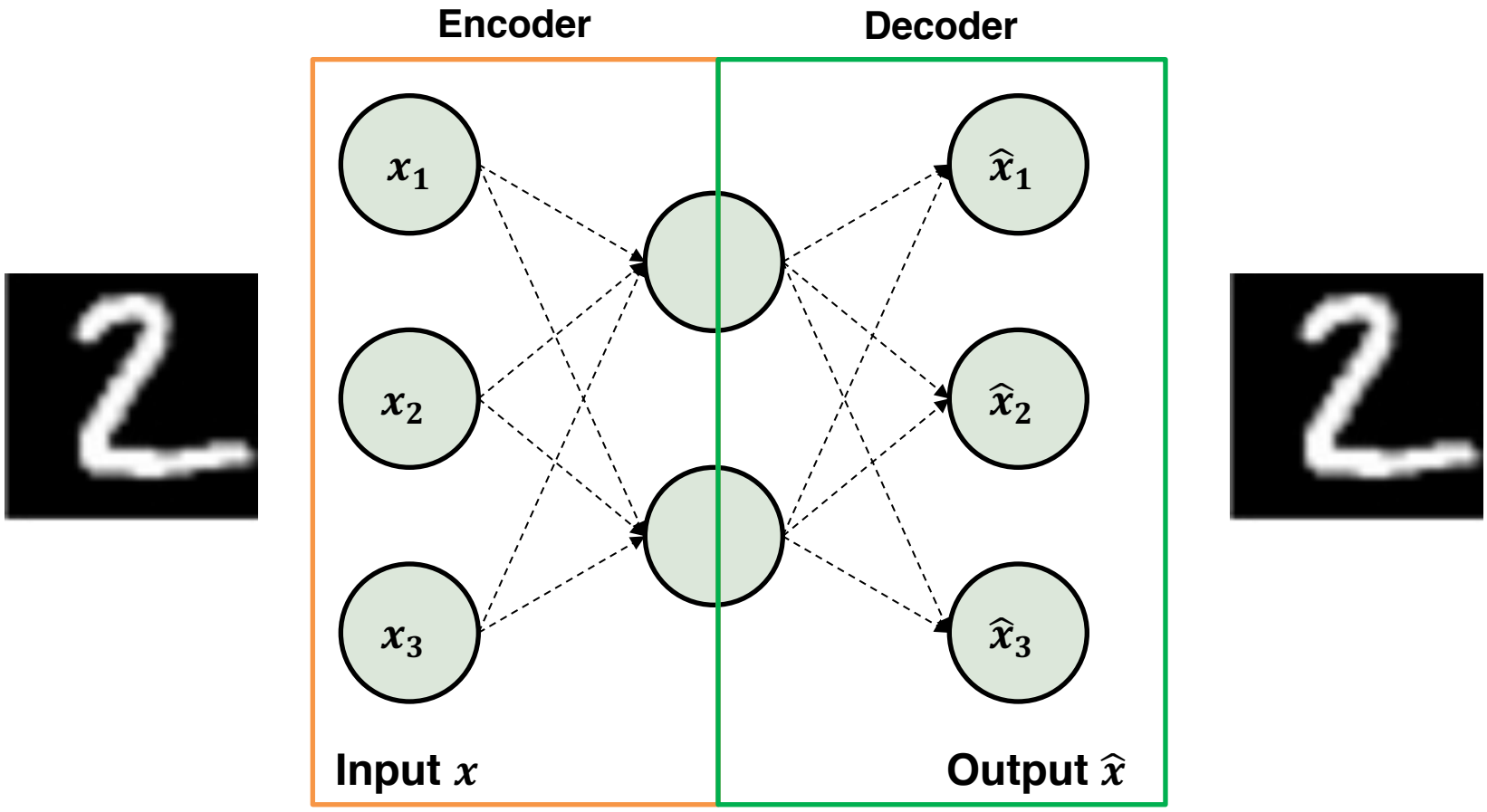


Application of autoencoders

- **Pre-training**
- **Dimensionality reduction**
- **Generative model**
  - some autoencoders are equivalent to Restricted Boltzmann Machines (RBM)
  - autoencoder for generative model will be dealt with at chap. 20

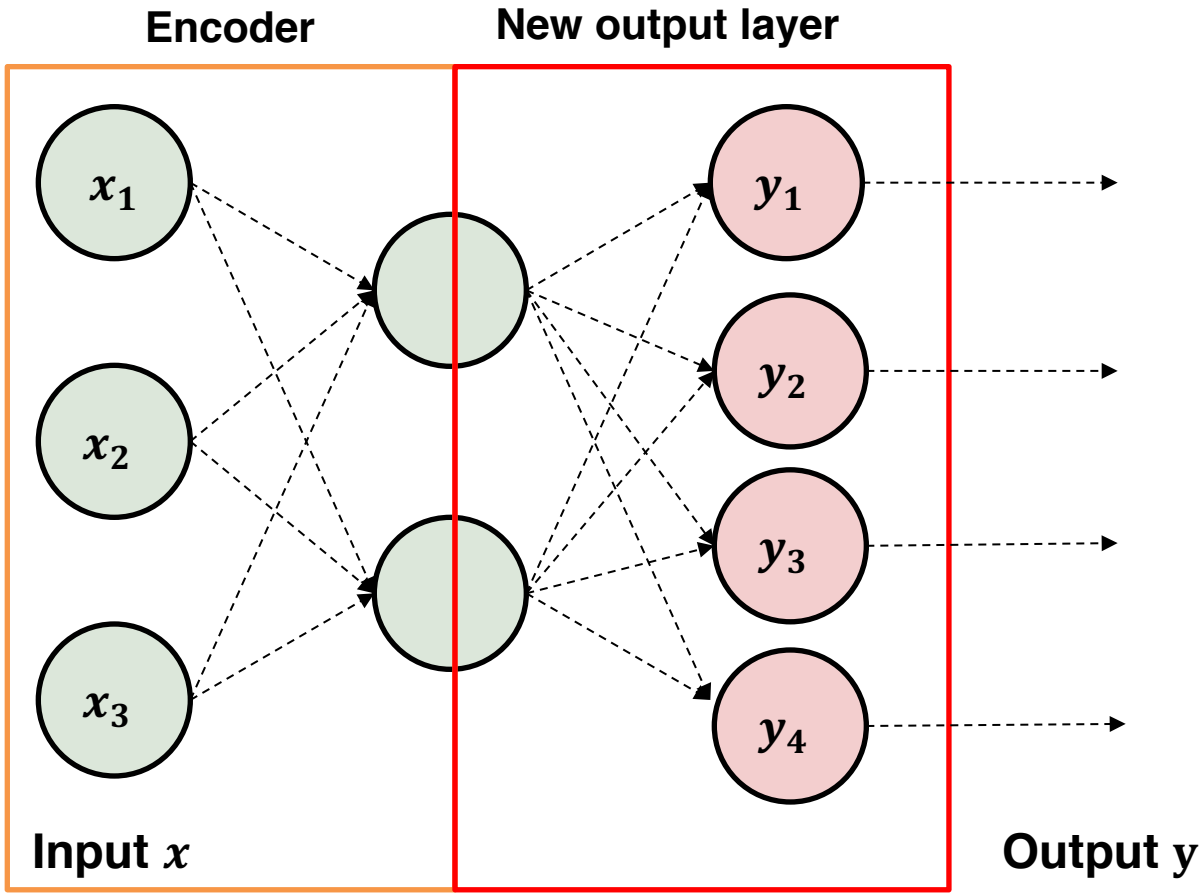
Pre-training

- Train autoencoder and replace decoder



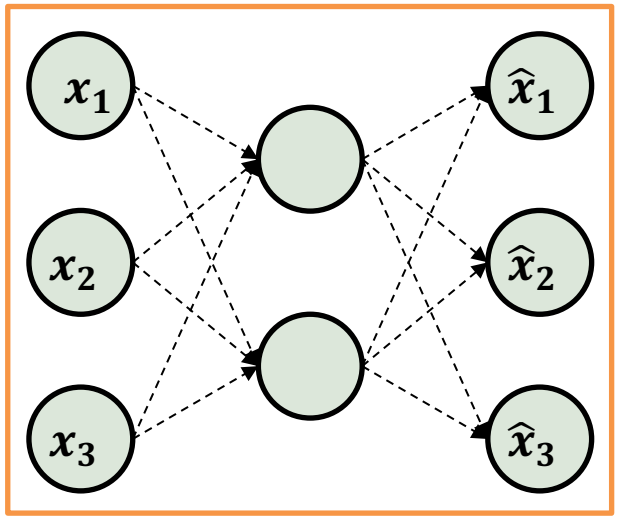
Pre-training (fine tuning)

- Train autoencoder and replace decoder

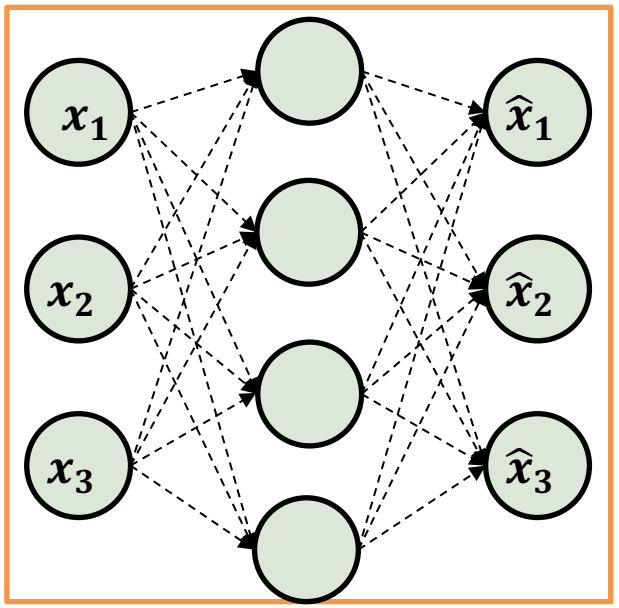


Dimensionality reduction

- **Undercomplete autoencoder**
  - output dimension of encoder is smaller than input dimension



Undercomplete



Overcomplete

Dimensionality reduction

- If autoencoder is undercomplte, encoder has effect of dimensionality reduction
- If decoder is linear, encoder has same effect of Principle Component Analysis (PCA)

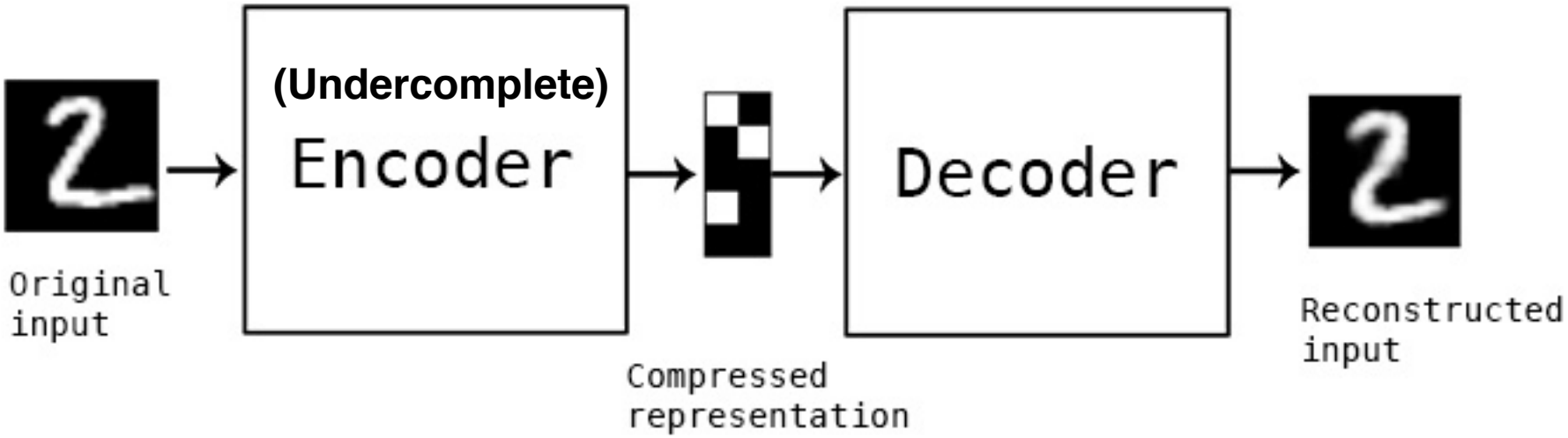
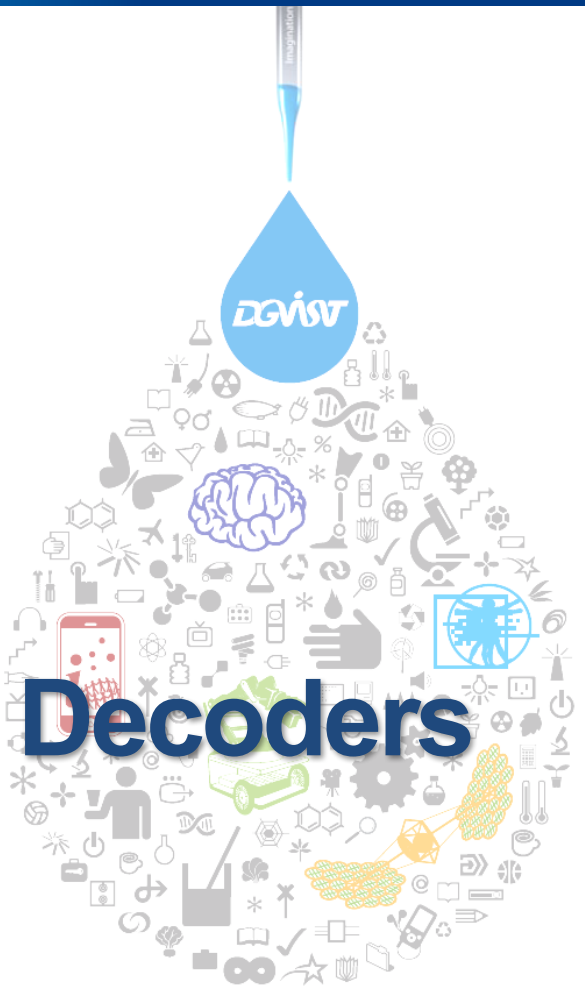


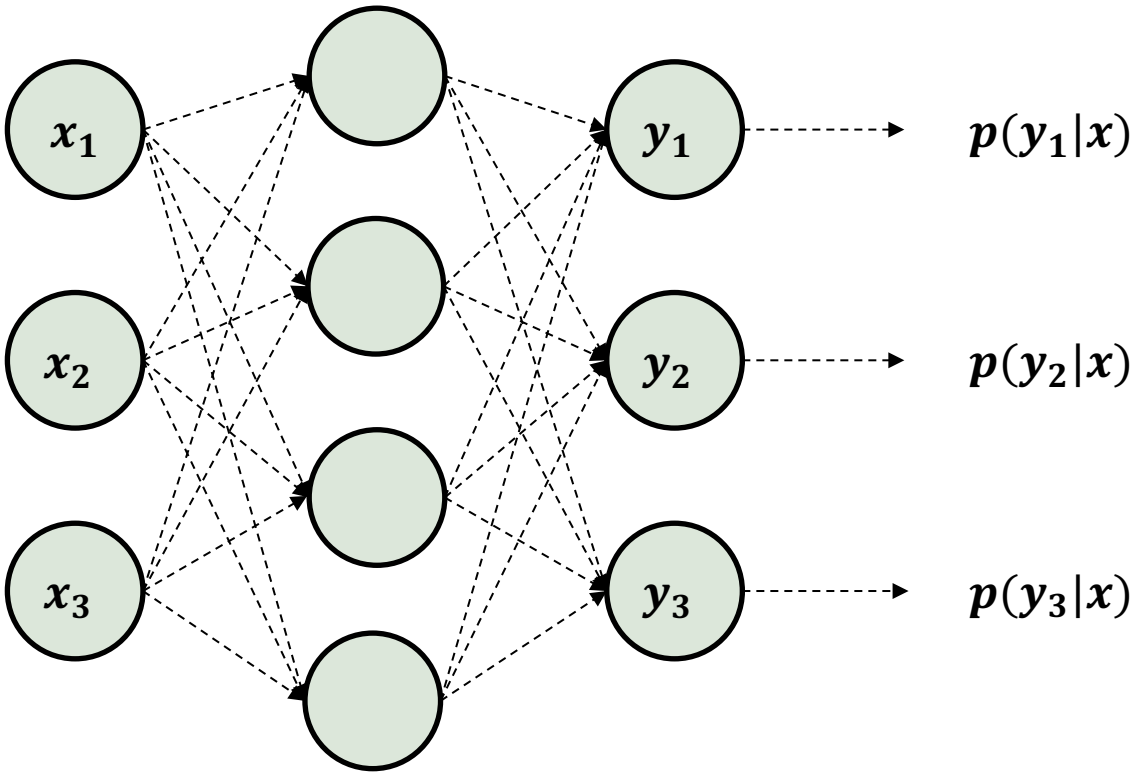
image source : https://blog.keras.io/building-autoencoders-in-keras.html

Stochastic Encoders and Decoders



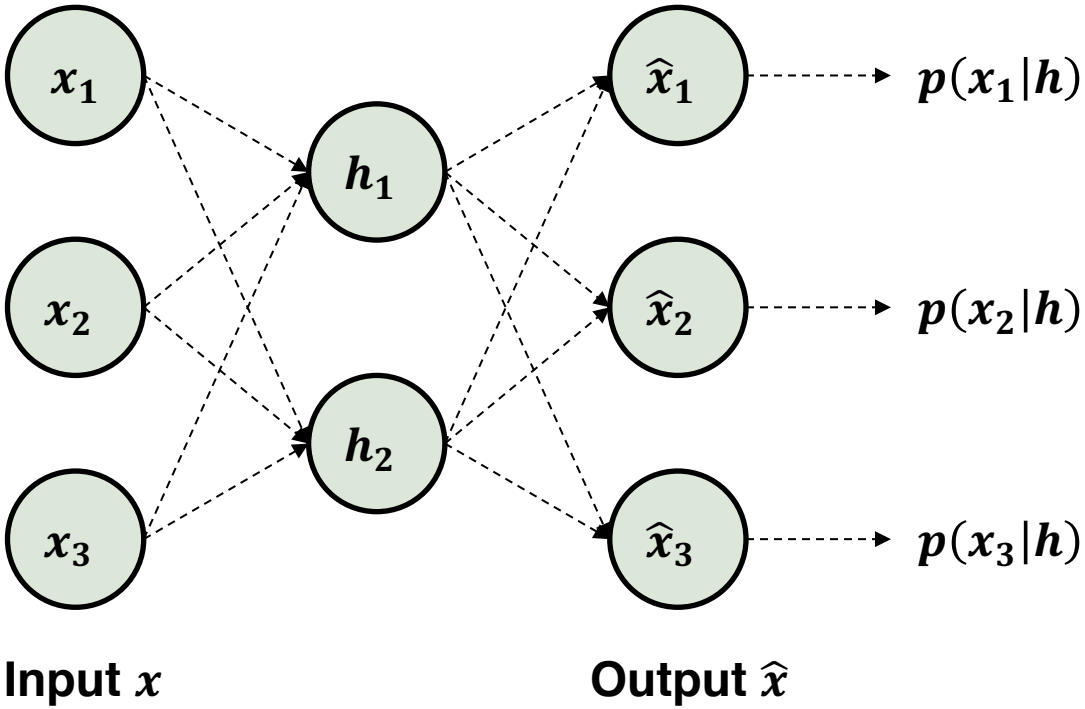
Probability of neural network

- **Normal feedforward neural network and its probability**
  - **training can be done with minimizing $-\log(p(y|x))$**



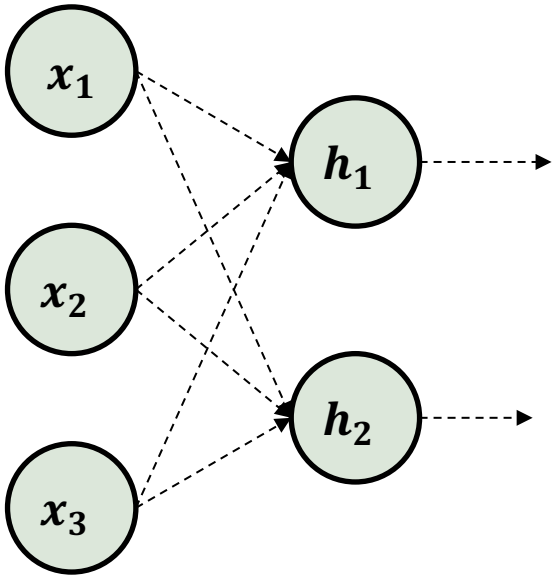
Probability of autoencoder

- **Autoencoders are just feedforward networks**
  - the same loss function and output unit types that used for feedforward neural net can be used for autoencoder



Probability of autoencoder

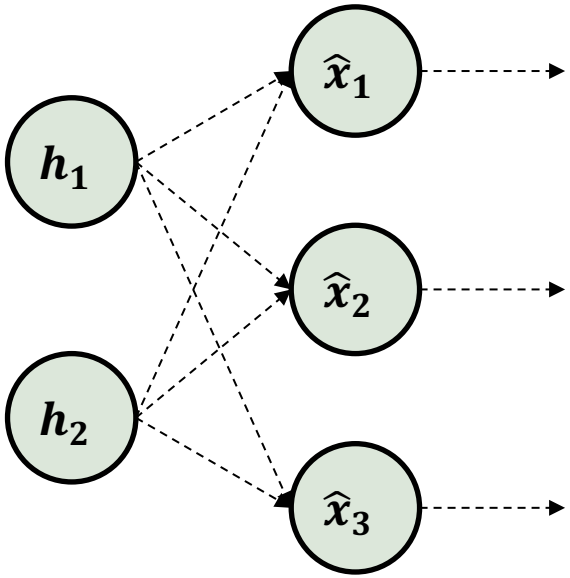
- **Encoder**



$$p_{\text{encoder}}(\boldsymbol{h} \mid \boldsymbol{x}) = p_{\text{model}}(\boldsymbol{h} \mid \boldsymbol{x})$$

Probability of autoencoder

- **Decoder**

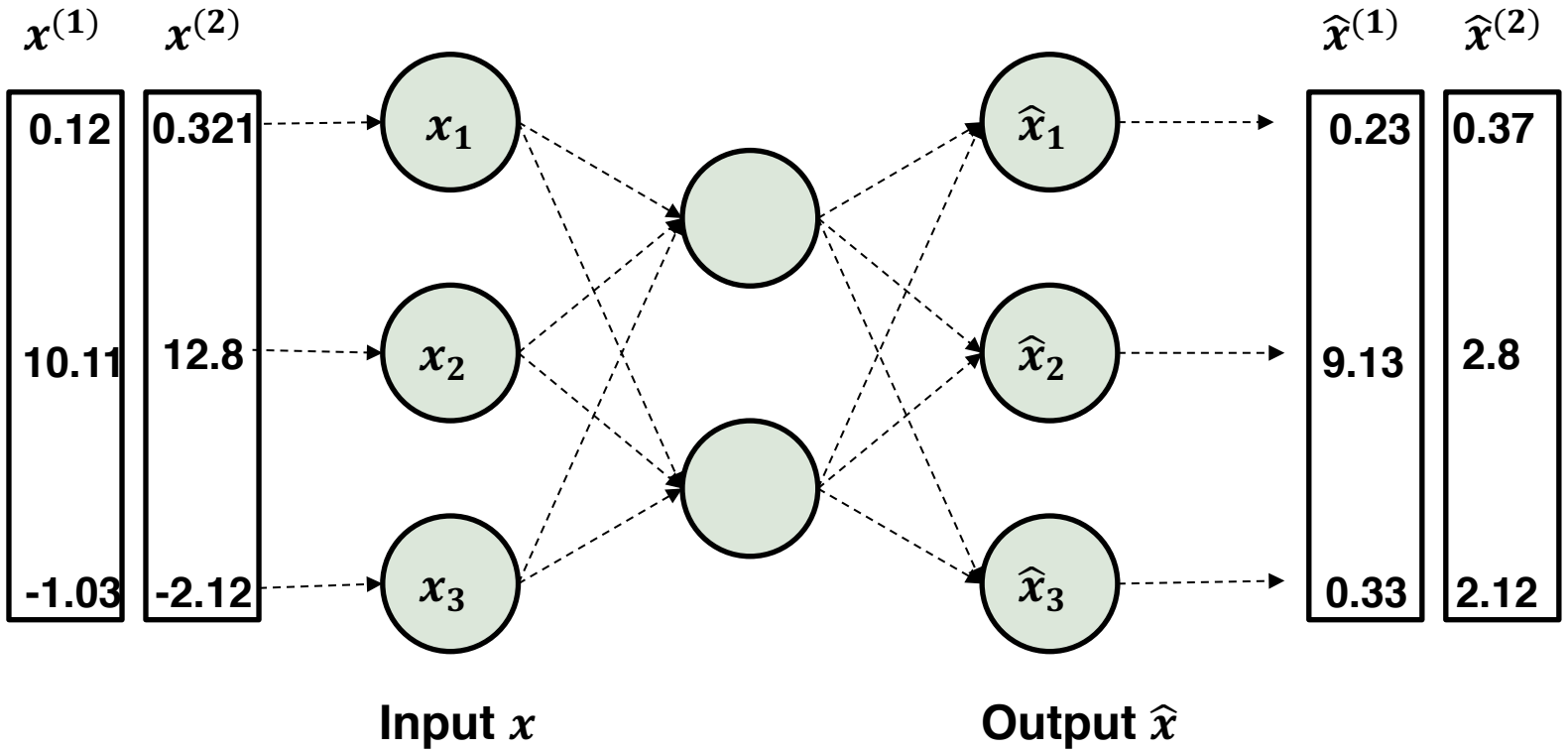


$$p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h}) = p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h}).$$

Learning of autoencoders

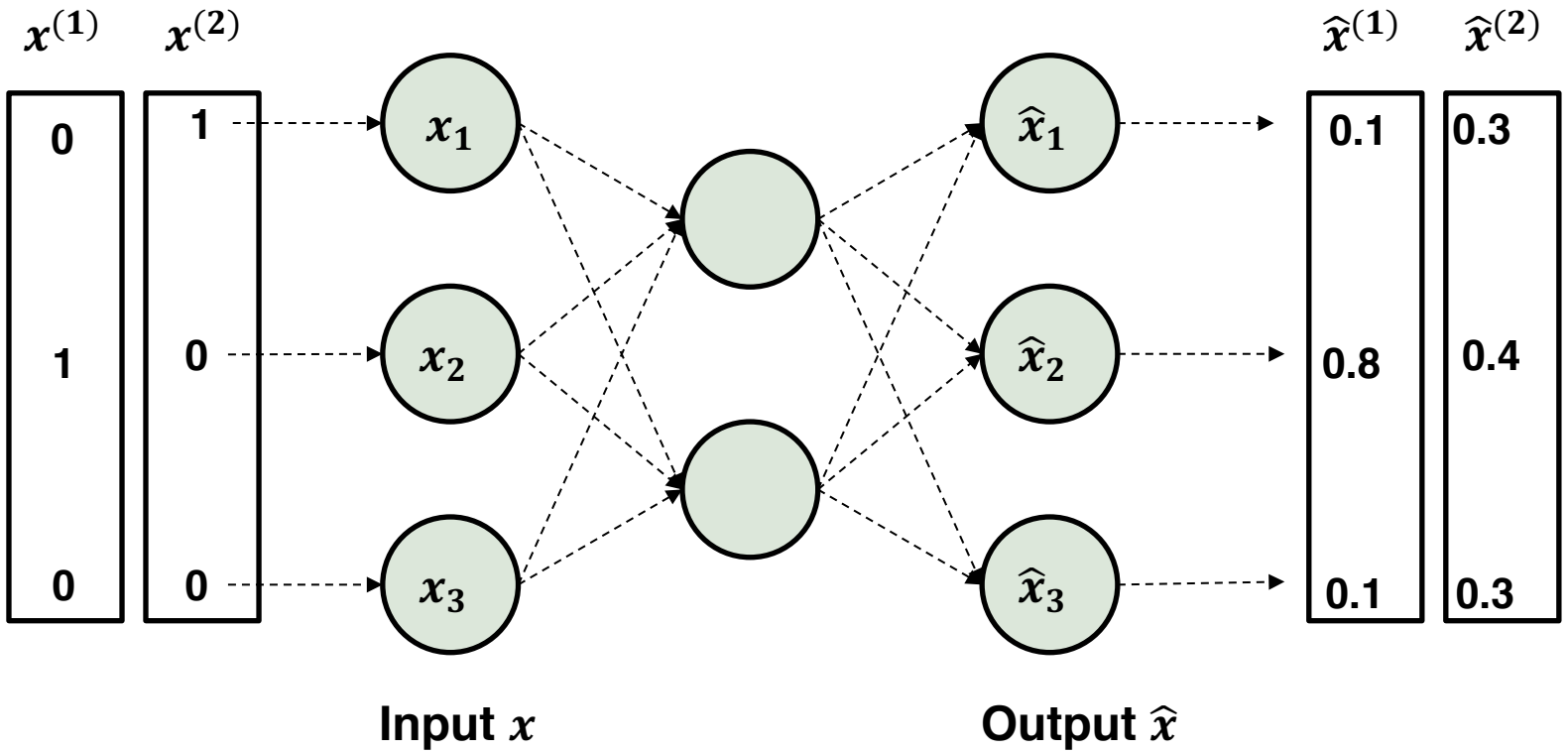
- **Autoencoder with real num**
  - use Mean Square Error (MSE)

$$MSE = \frac{1}{N}\frac{1}{M}\sum_{m}^{M}\sum_{n=1}^{N}(x_m^n - \hat{x}_m^n)^2$$

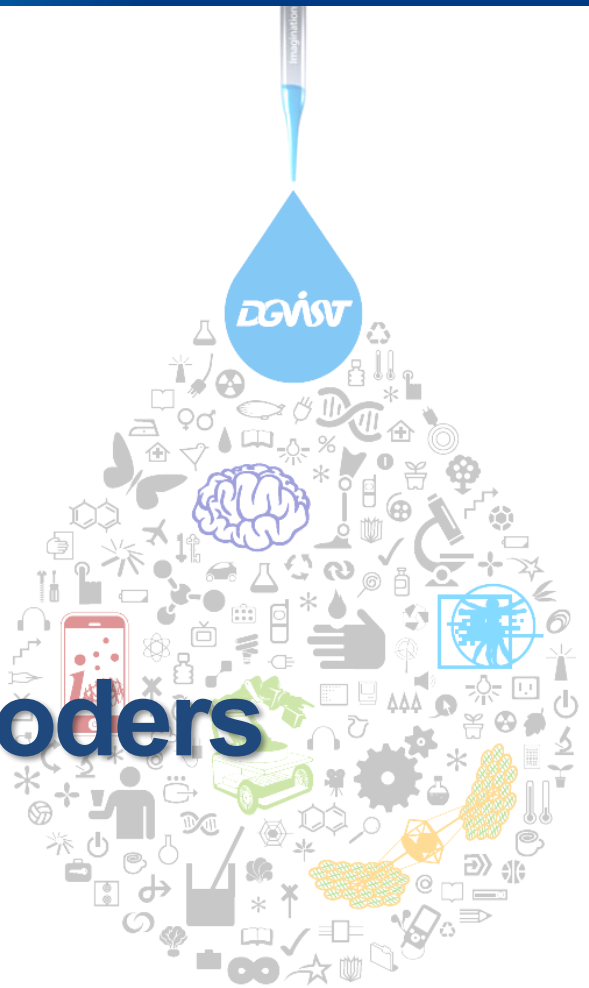


Learning of autoencoders

- **Autoencoder with binomial/multinomial value as input**
  - use softmax as output

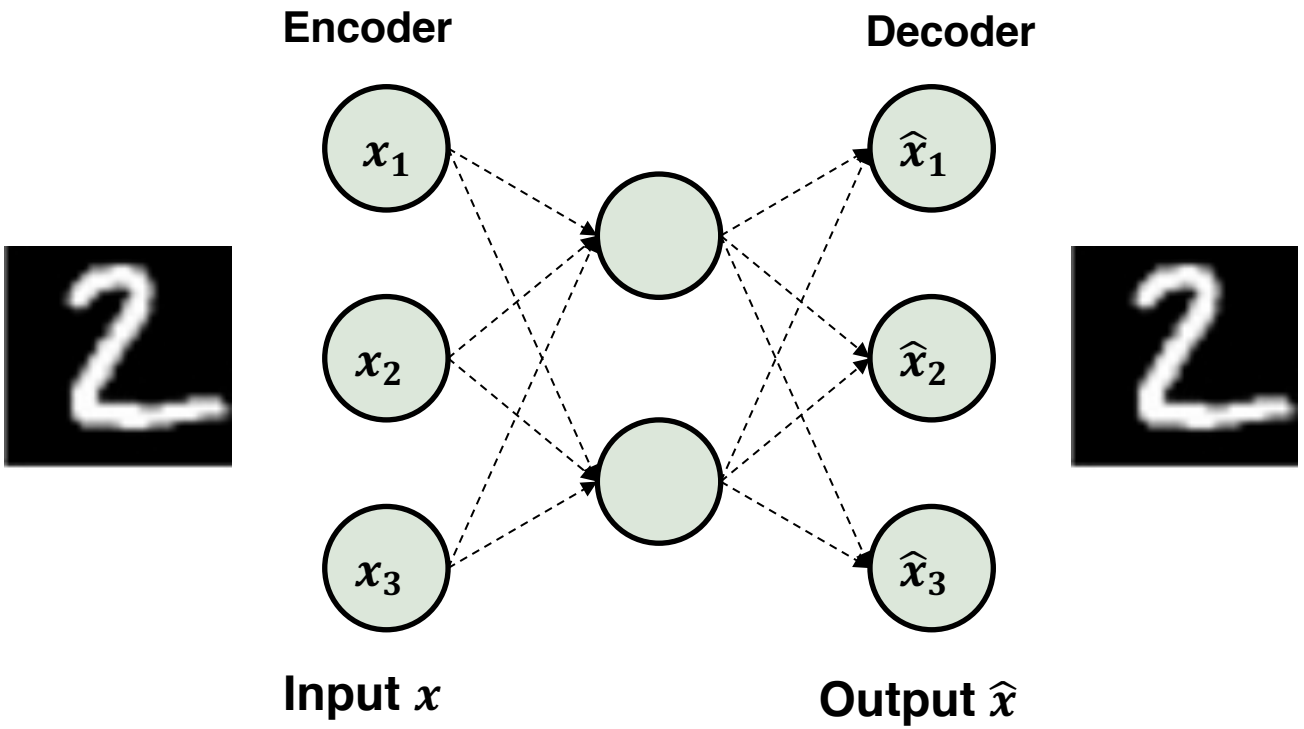


Regularized autoencoders



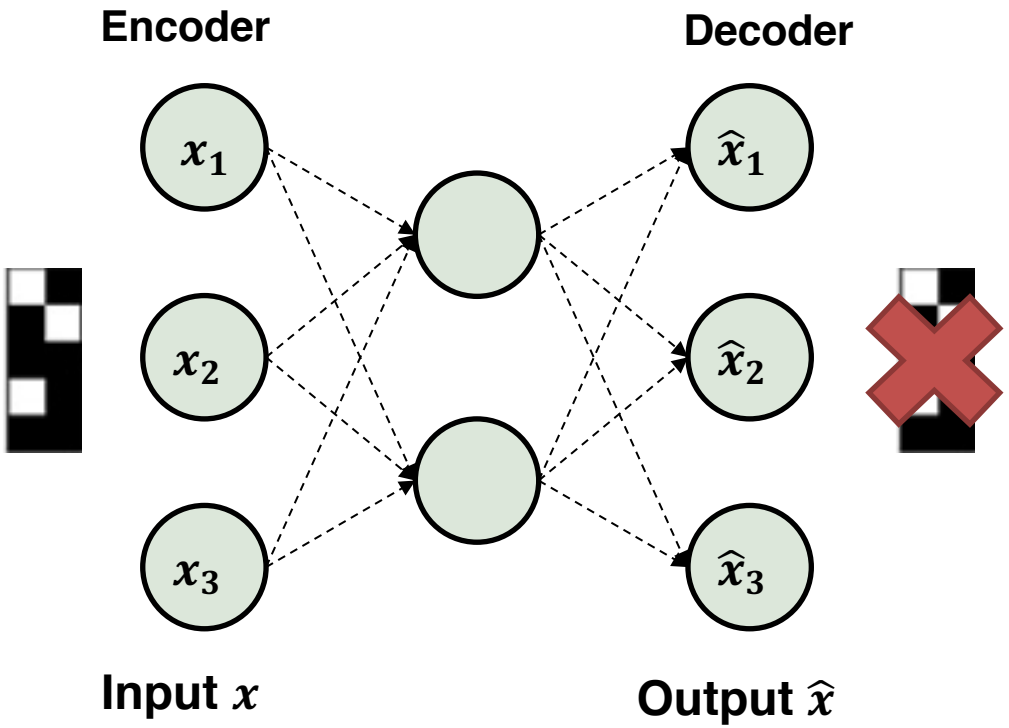
Identity function

- The identity function is simply copy input to output
- Autoencoders have not to learn identity function



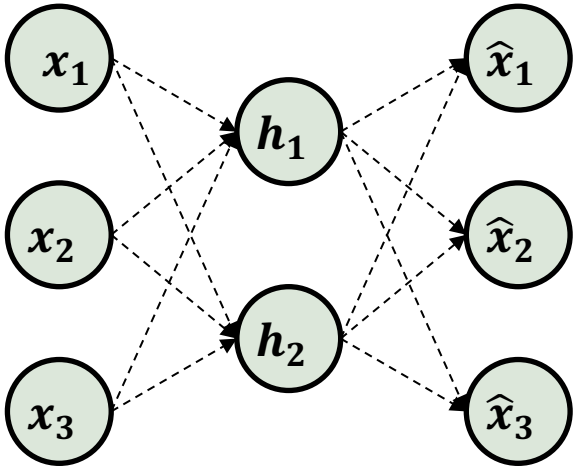
Identity function

- The identity function is simply copy input to output
- Autoencoders have not to learn identity function



Sparse autoencoders

- If the capacity of the autoencoder is too large, it can learn identity function
- Add L1 regularization term to constrain capacity



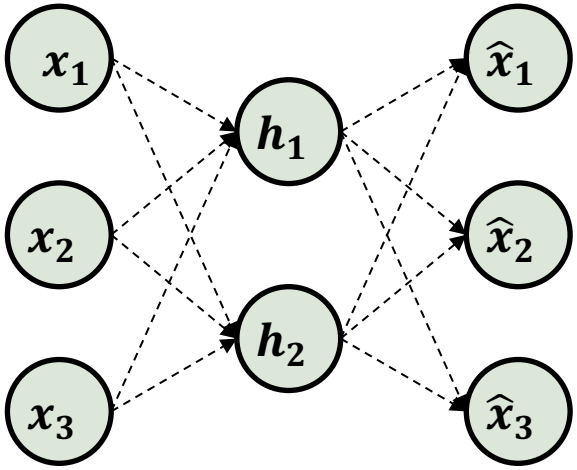
$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h})$$

$$\Omega(\boldsymbol{h}) = \lambda \sum_i |h_i|$$

Sparse autoencoders

- If the capacity of the autoencod
  learn identity function
- Add L1 regularization term to c

x : *input data*
h : *coded data*
L : *loss functoin*
f : *encoding*
g : *decoding*
Ω : penalty term
λ : importance of constrant

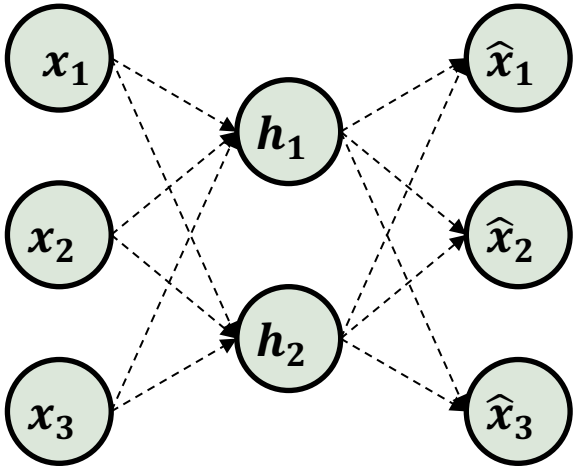


$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h})$$

$$\Omega(\boldsymbol{h}) = \lambda \sum_i |h_i|$$

Contractive autoencoder

- Spare autoencoder that replaces penalty term to differentiation of h
- This make autoencoder robust to small perturbation of input data x



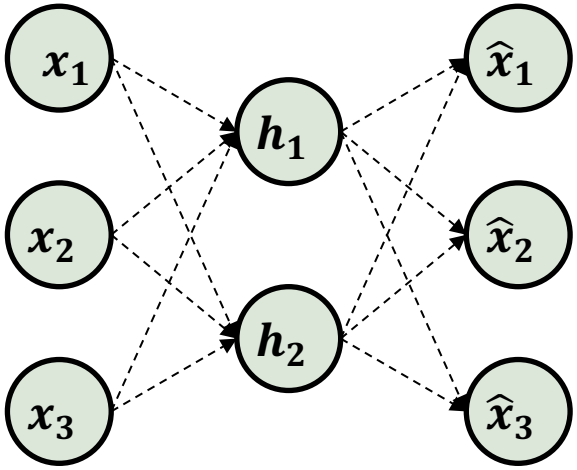
$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}, \boldsymbol{x}),$$

$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i ||\nabla_{\boldsymbol{x}} h_i||^2.$$

Contractive autoencoder

- Spare autoencoder that replace differentiation of h
- This make autoencoder robust input data x

x : input data
h : coded data
L : loss functoin
∇ : differntiation
f : encoding
g : decoding
Ω : penalty term
λ : importance of constrant



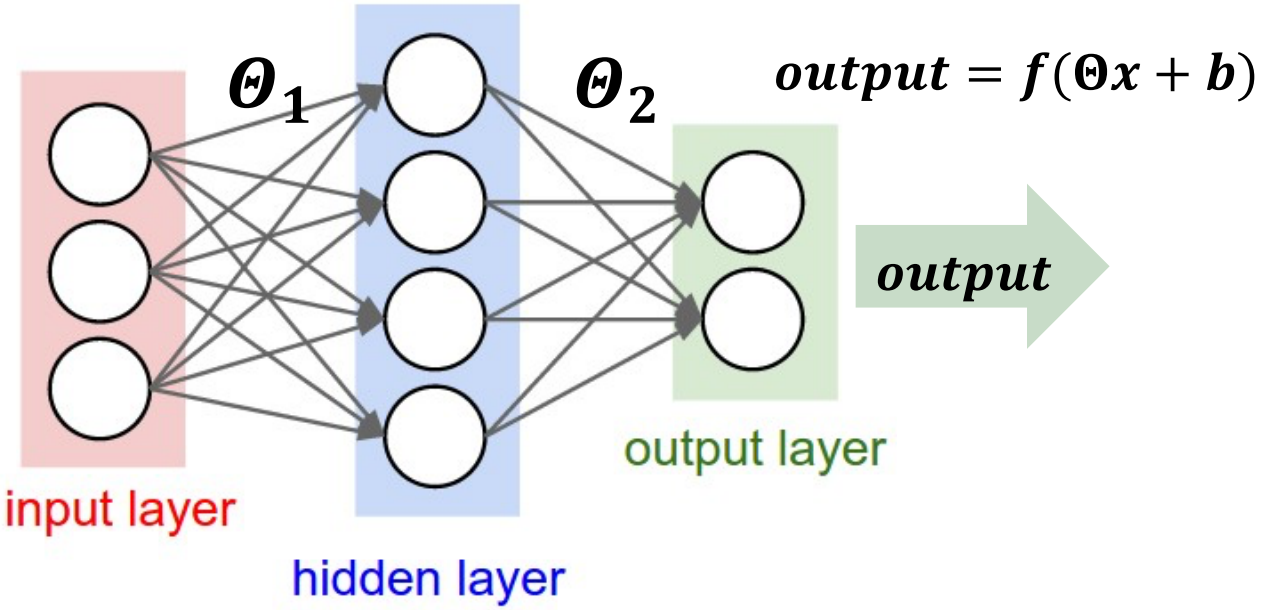
$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}, \boldsymbol{x}),$$

$$\Omega(\boldsymbol{h}, \boldsymbol{x}) = \lambda \sum_i ||\nabla_{\boldsymbol{x}} h_i||^2.$$

Data argumentation for noise robustness

- **Injecting random noise into input data to improve robustness**
  - this was discussed at chap. 7

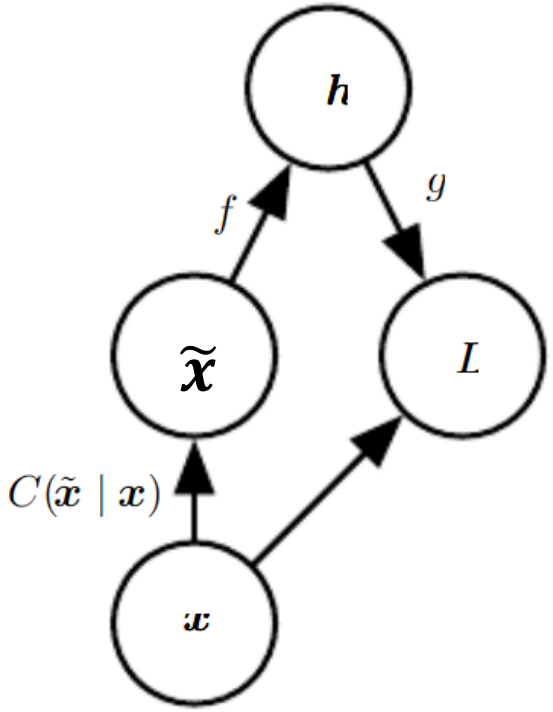
Random Noise Filter



source : http://cs231n.github.io/neural-networks-1/

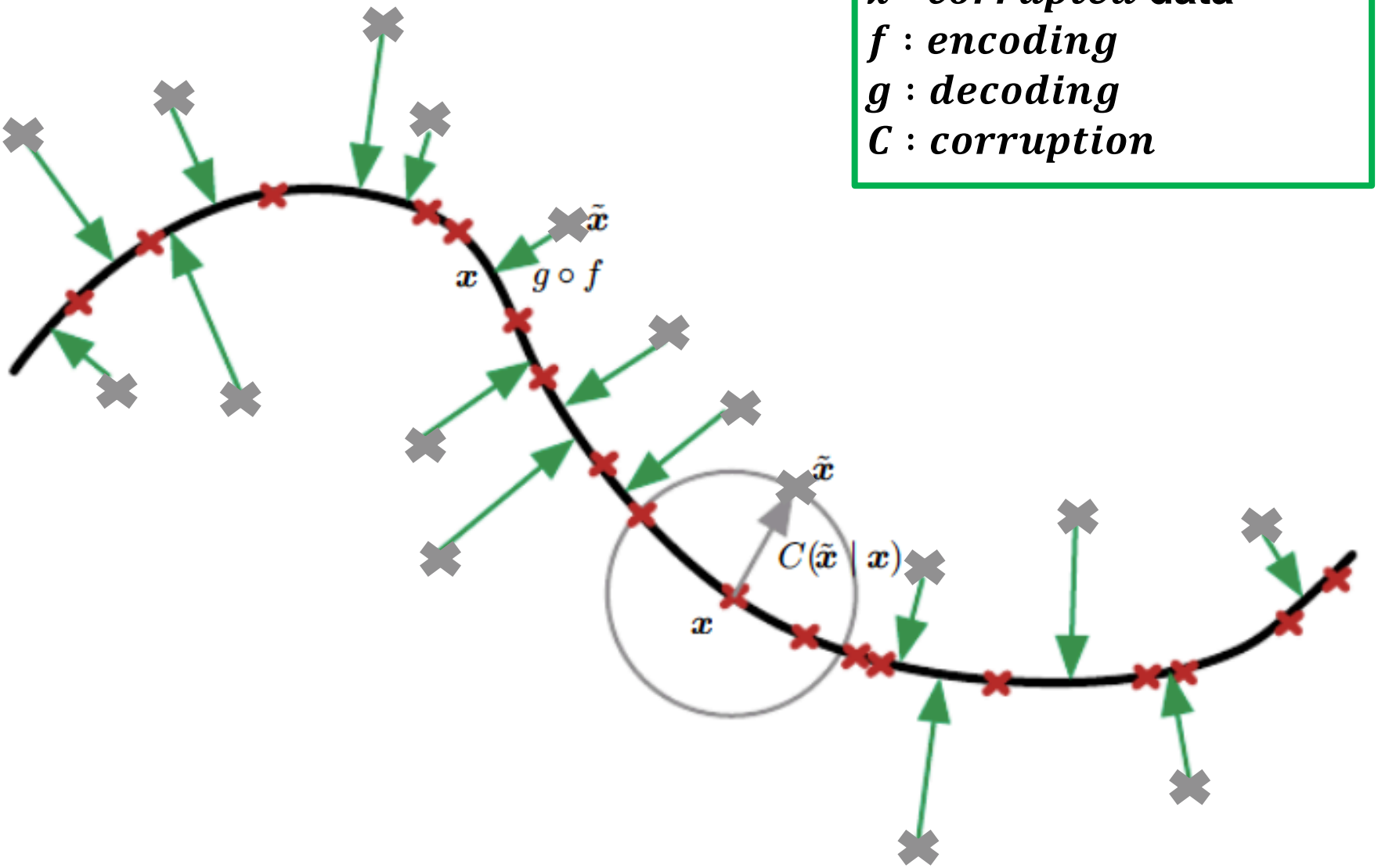
Denoising autoencoders

- **To make network robust about noise through blow procedure**
  1. **sample training example x from the training data**
  2. **sample a corrupted version $\tilde{x}$ from $C(\tilde{x}|x)$**
  3. **use $\tilde{x}$ as input data and x as label**



Denoising autoencoders

x : *original data*
$\tilde{x}$: *corrupted* data
f : *encoding*
g : *decoding*
C : *corruption*



Denoising autoencoders

- An example about denoise through denoising autoencoder

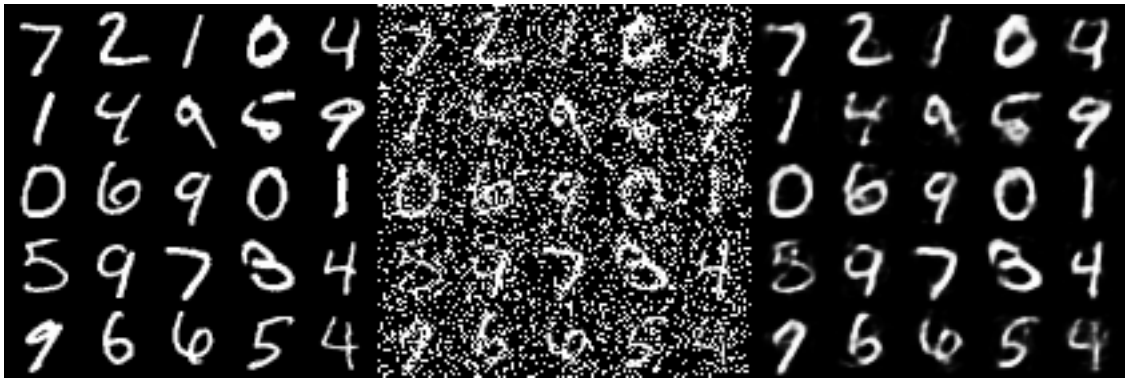


Image source : http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model

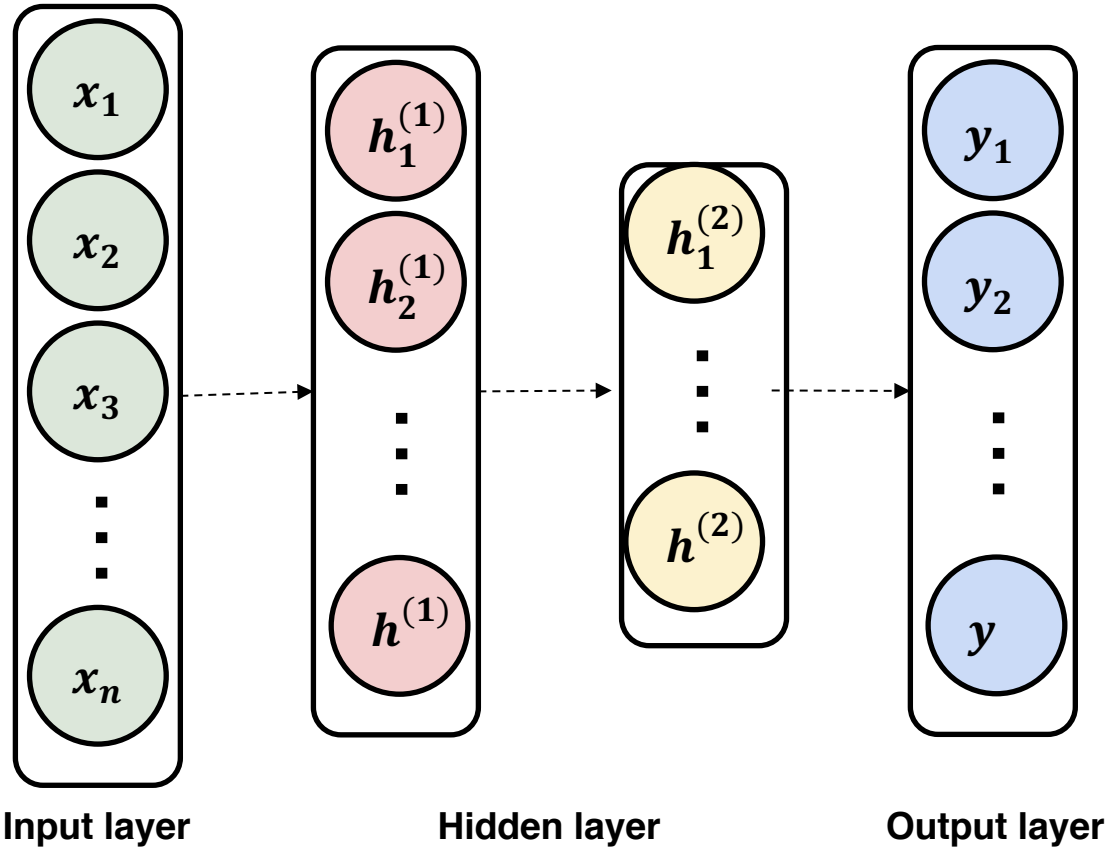
Representational Power, Layer Size and Depth

Necessity of deep autoencoder

- Deeper networks can learn more complex feature of input data
- To pre-train for deep neural network, autoencoder need to have same depth with deep neural network
- Deep autoencoders reduce the probability that the auto encoder will learn the identity function

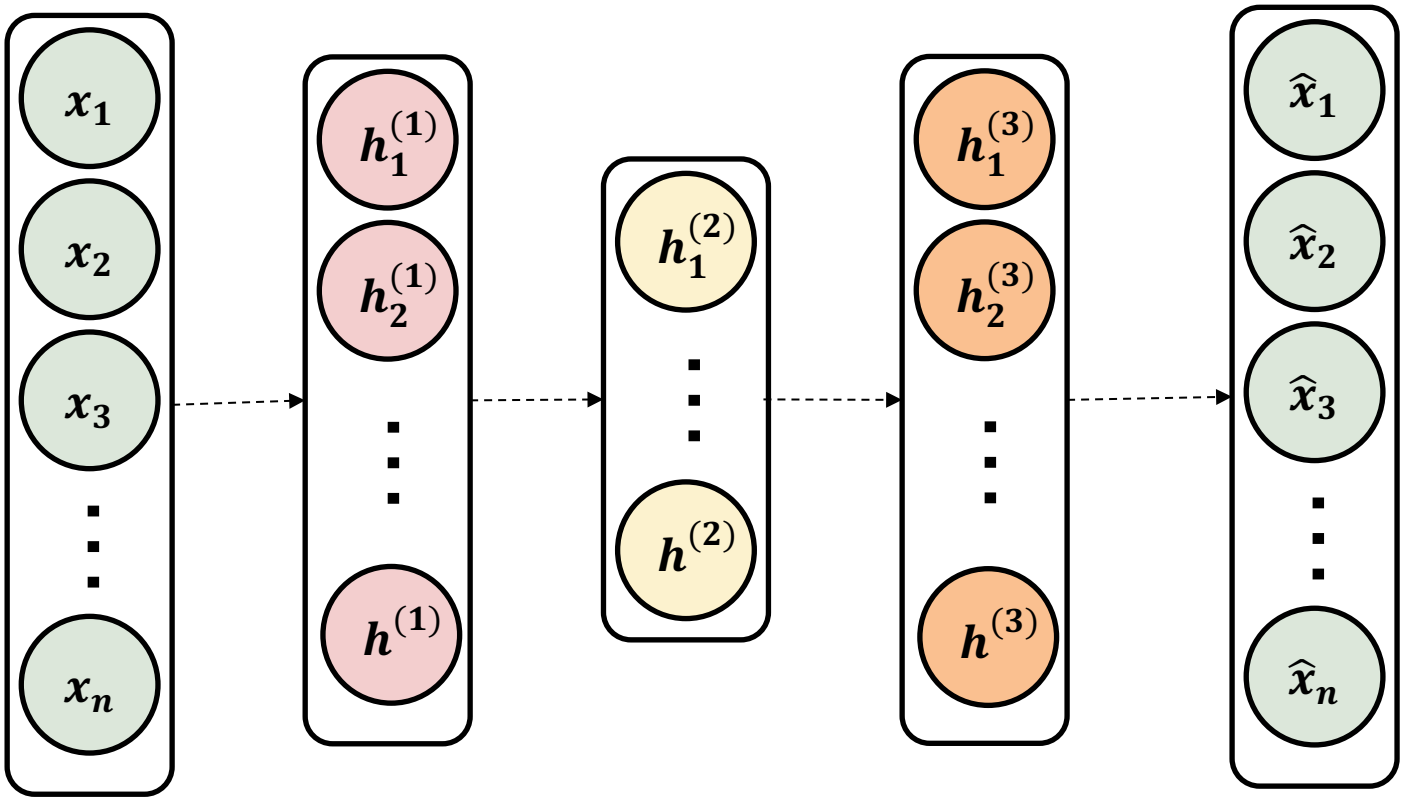
Pre-training for deep neural network example

- Deep neural network with 2 hidden layers

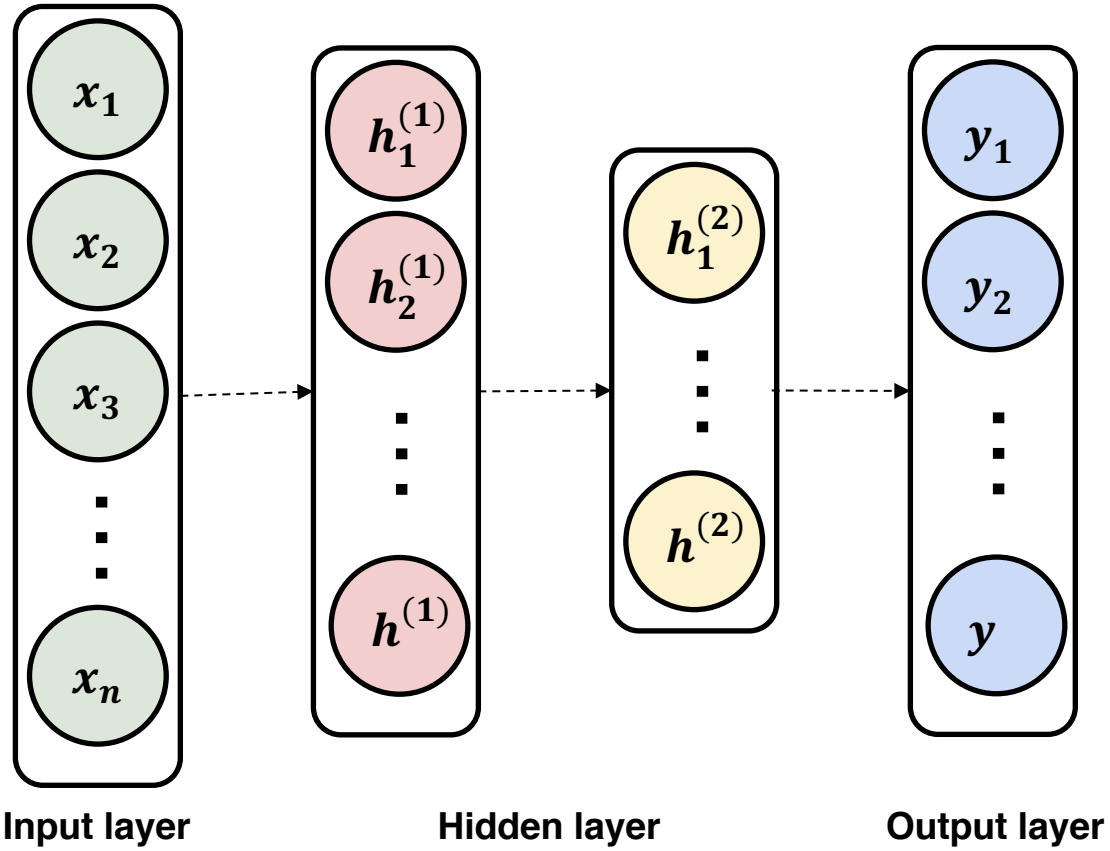


Deep autoencoder

- **Make a symmetric architecture of the original network and pre-train**

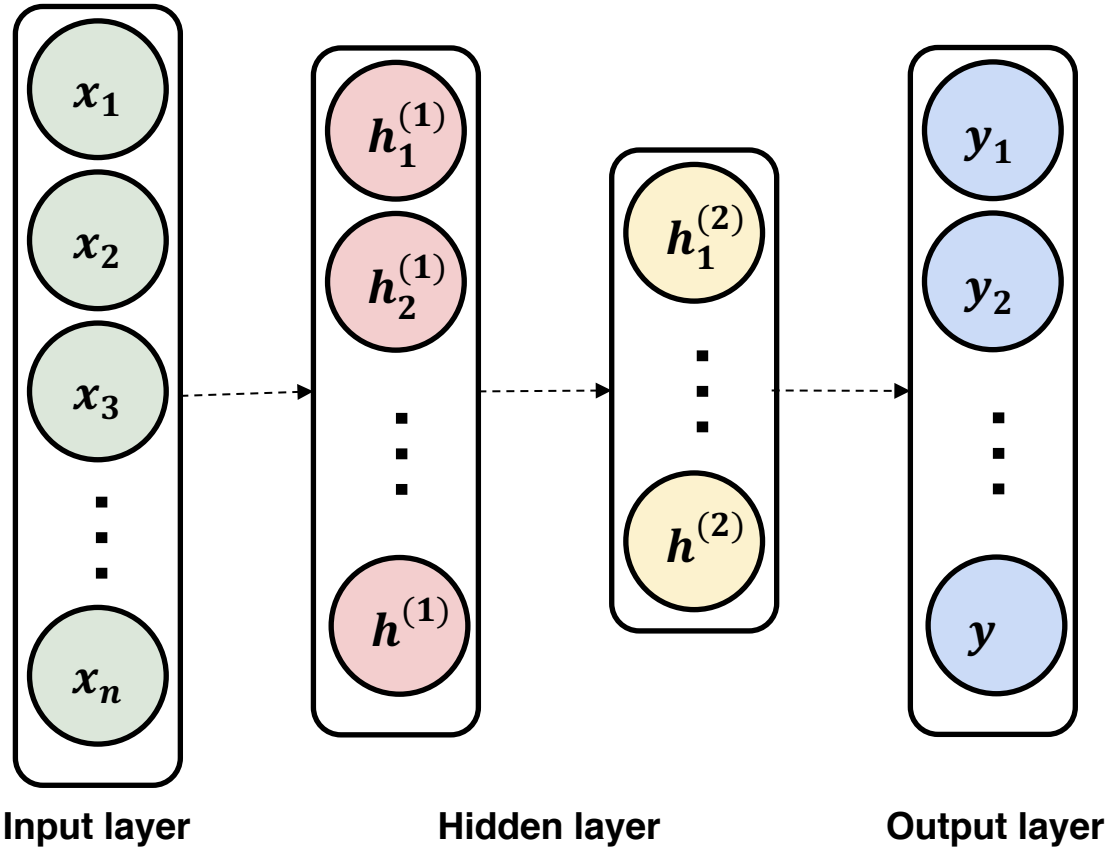


Deep autoencoder



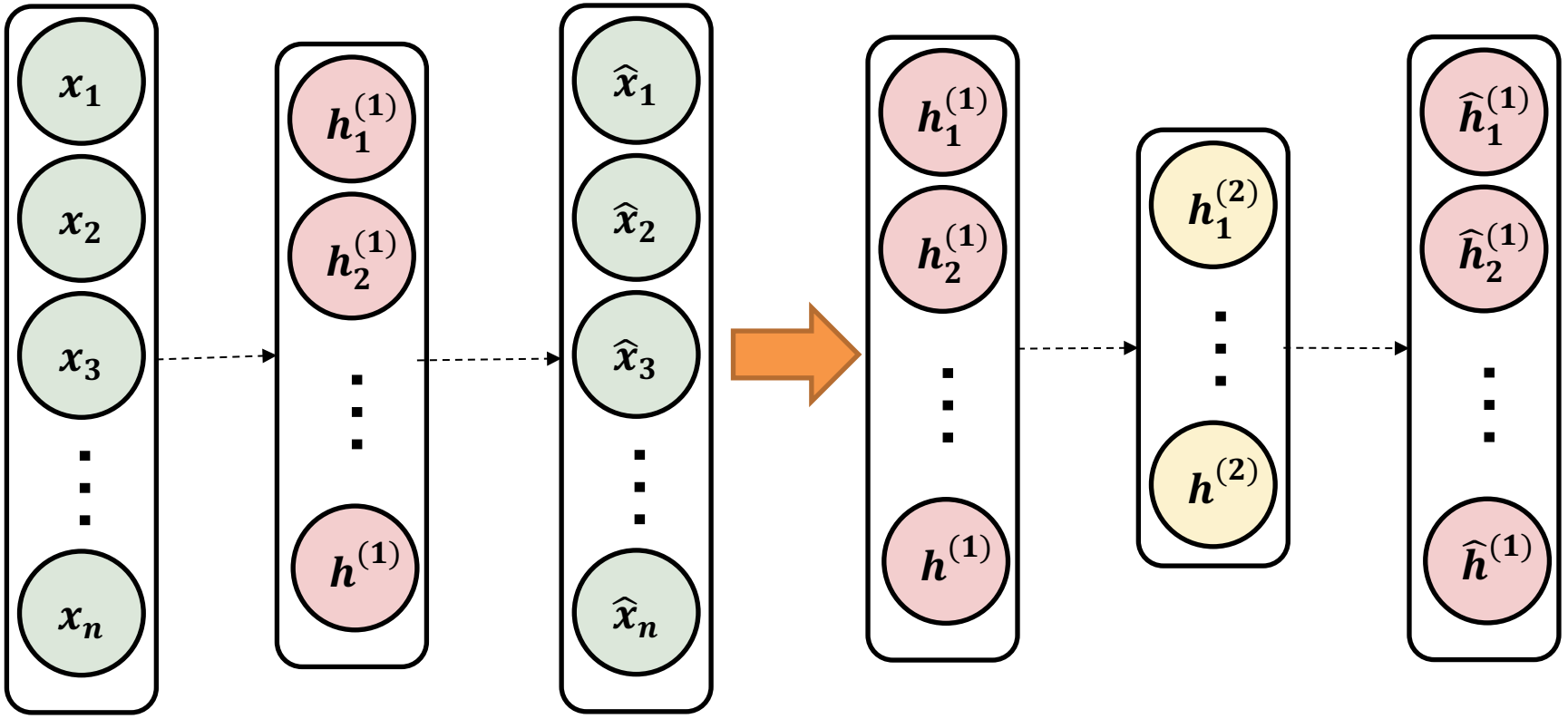
Pre-training for deep neural network example

- Deep neural network with 2 hidden layers



Stacked autoencoder

- Separate autoencoders



1st autoencoders

2nd autoencoders

Stacked autoencoder

- Attach layers after train autoencoders