




# **The Introduction To Artificial Intelligence**

**Yuni Zeng [yunizeng@zstu.edu.cn](mailto:yunizeng@zstu.edu.cn)  
2024-2025-1**

# The Introduction to Artificial Intelligence

- Part I Brief Introduction to AI & Different AI tribes
- Part II Knowledge Representation & Reasoning
- Part III AI GAMES and Searching
- Part IV Model Evaluation and Selection
- Part V Machine Learning
-  Part VI Neural Networks

# Neural Networks

- Brief review
- Sequence Learning



# Sequence Learning

## □ A Sequence Recognizing Example

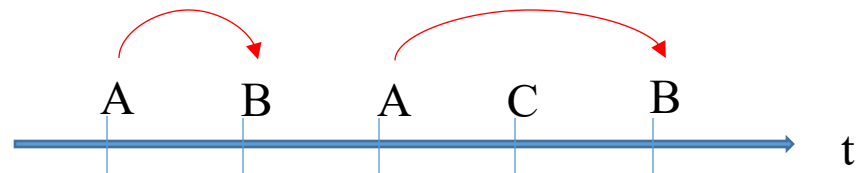
*Recognize A followed by B Problem*

The task is to recognize A followed by B.

Generated Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC

.....



# Sequence Learning

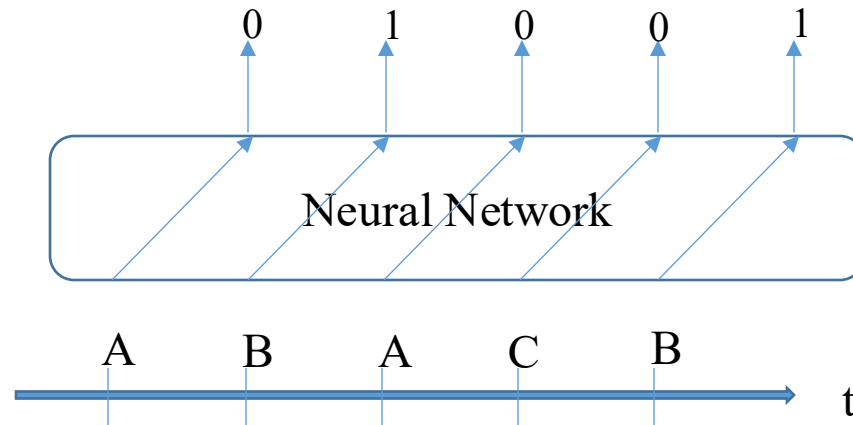
## □ A Sequence Recognizing Example

### *Recognize A followed by B Problem*

The task is to recognize A followed by B.

Generated Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC
- .....



**Problem: Can we use neural network to solve this problem?**

# Sequence Learning

## □ A Sequence Recognizing Example

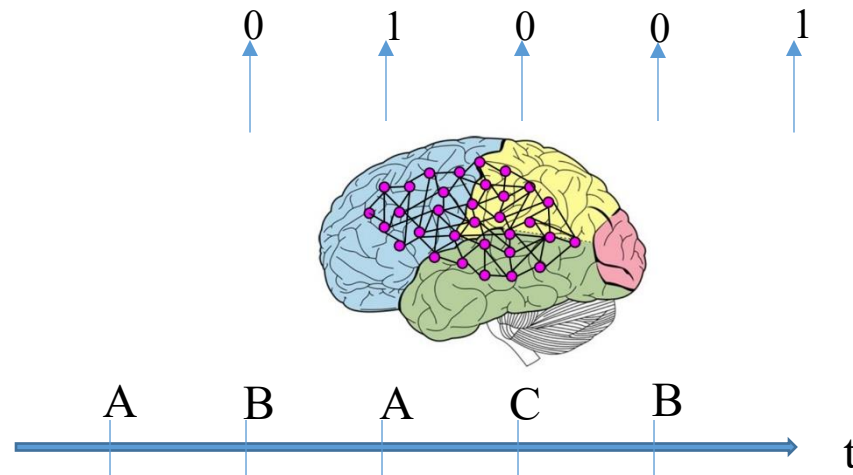
### *Recognize A followed by B Problem*

The task is to recognize A followed by B.

The brain can solve this problem simply by memorizing the last A.

#### Generated Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC
- .....



**Problem:** Can we use neural network to solve this problem?

# Sequence Learning

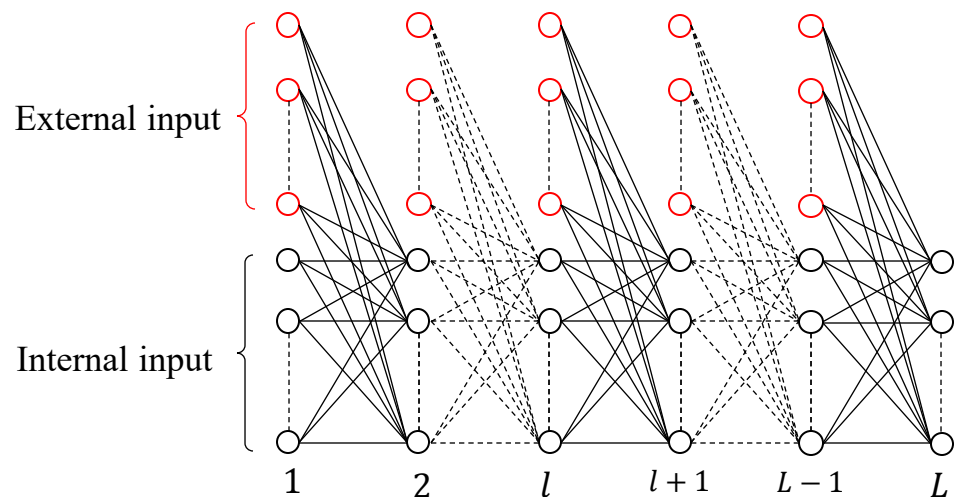
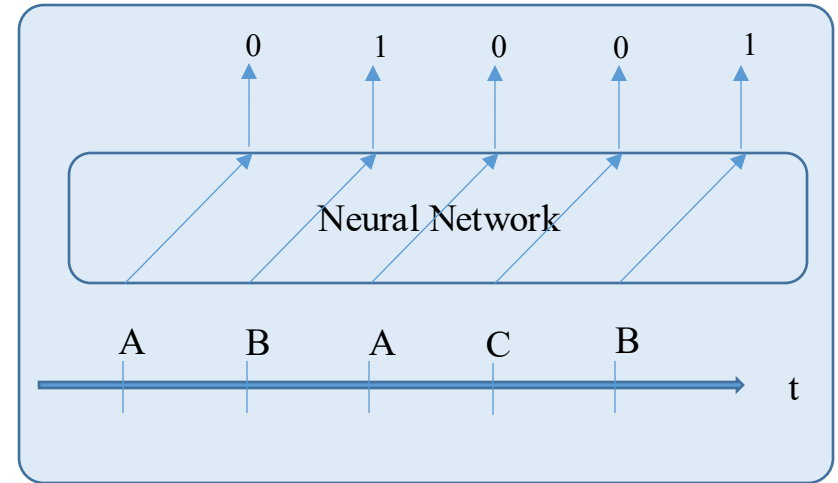
## □ A Sequence Recognizing Example

### *Recognize A followed by B Problem*

The task is to recognize A followed by B.

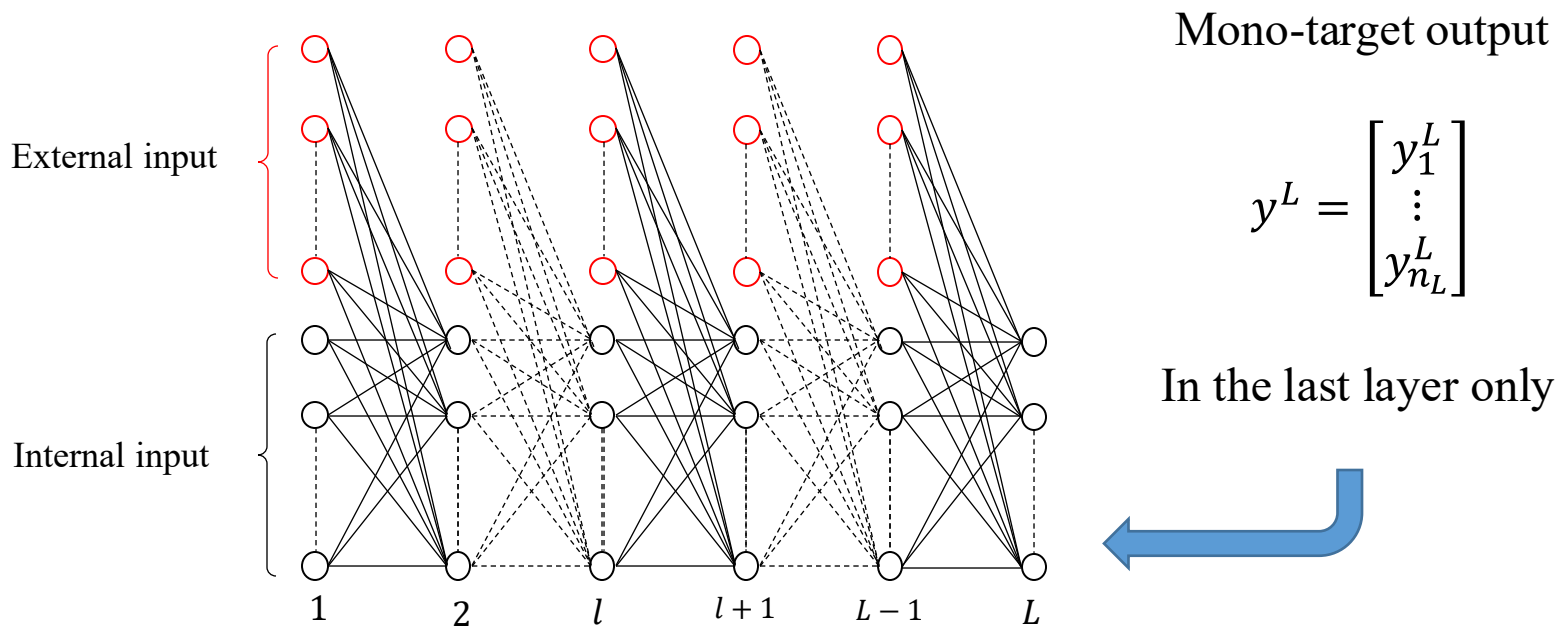
Generated Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC
- .....



# Sequence Learning

## □ A Sequence Recognizing Example



Mono-target output network cannot solve the sequence recognizing problem.

# Sequence Learning

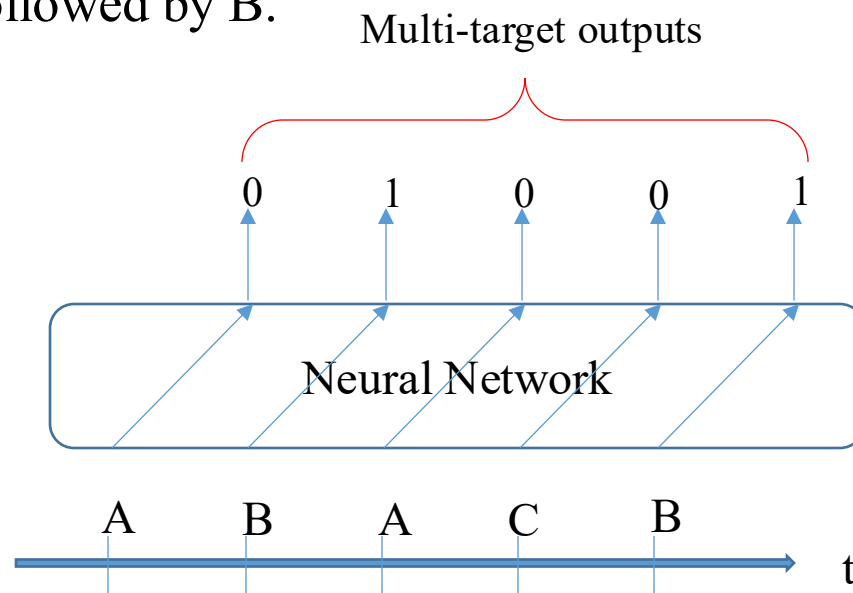
## □ A Sequence Recognizing Example

### *Recognize A followed by B Problem*

The task is to recognize A followed by B.

#### Generated Sequences

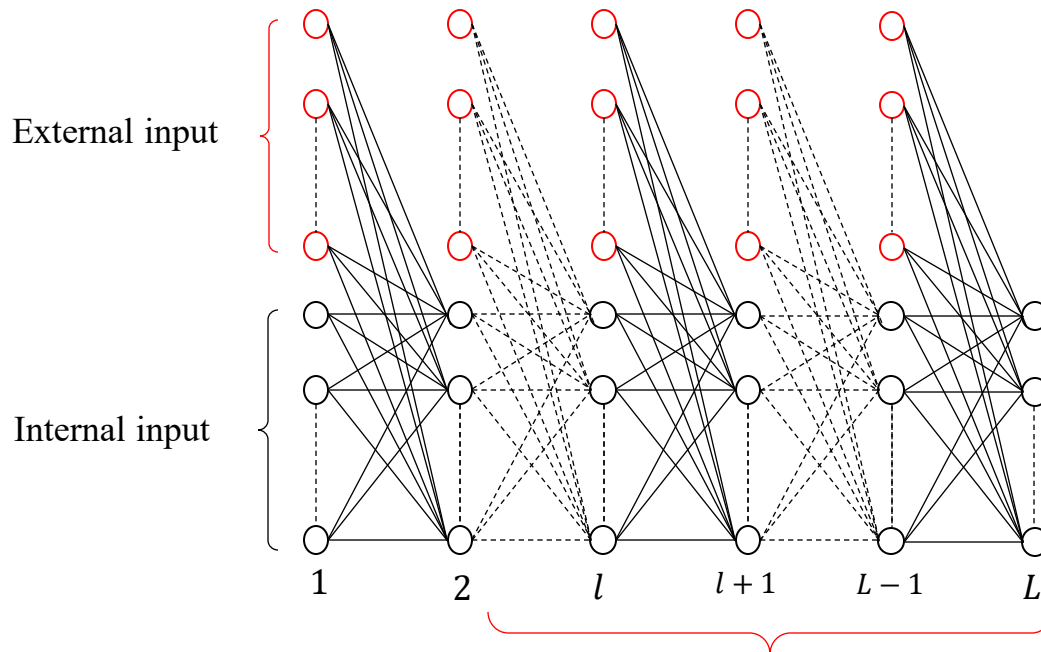
1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC
- .....



**Problem: Can we use neural network to solve this problem?**

# Sequence Learning

## □ A Sequence Recognizing Example



Multi-target outputs

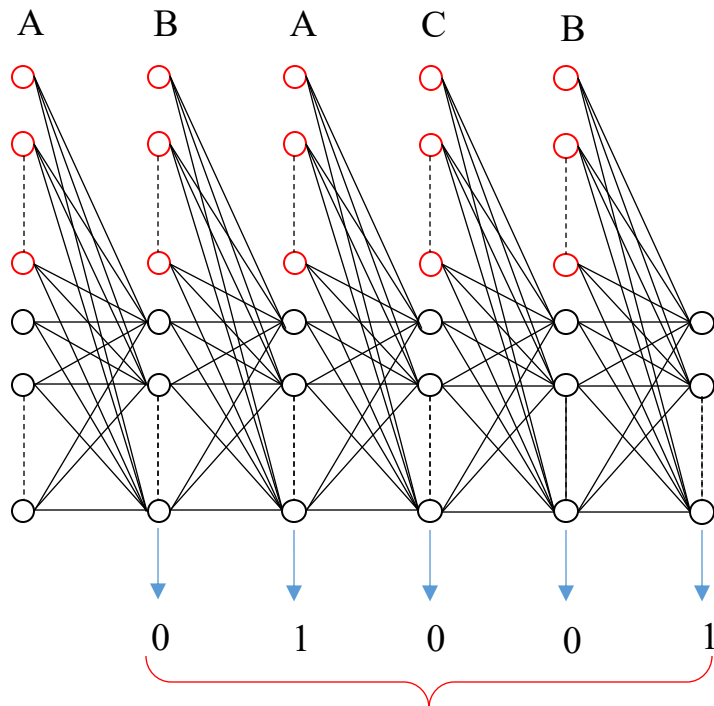
$$y^l = \begin{bmatrix} y_1^l \\ \vdots \\ y_{n_L}^l \end{bmatrix}$$

$$(l = 2, \dots, L)$$

Multi-target outputs

# Sequence Learning

## □ A Sequence Recognizing Example



Multi-target outputs

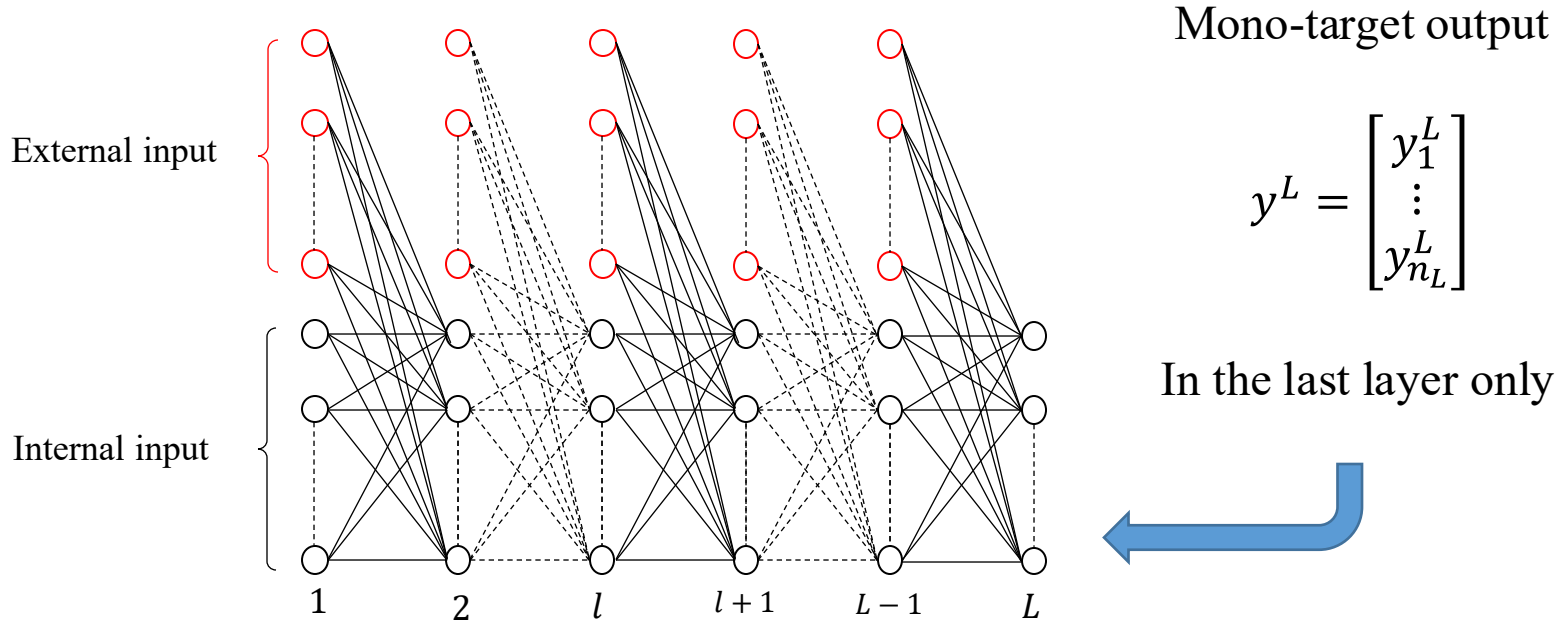
$$y^l = \begin{bmatrix} y_1^l \\ \vdots \\ y_{n_L}^l \end{bmatrix}$$

$$(l = 2, \dots, L)$$

**Problem:**  
How to develop algorithm to train the network?

# Sequence Learning

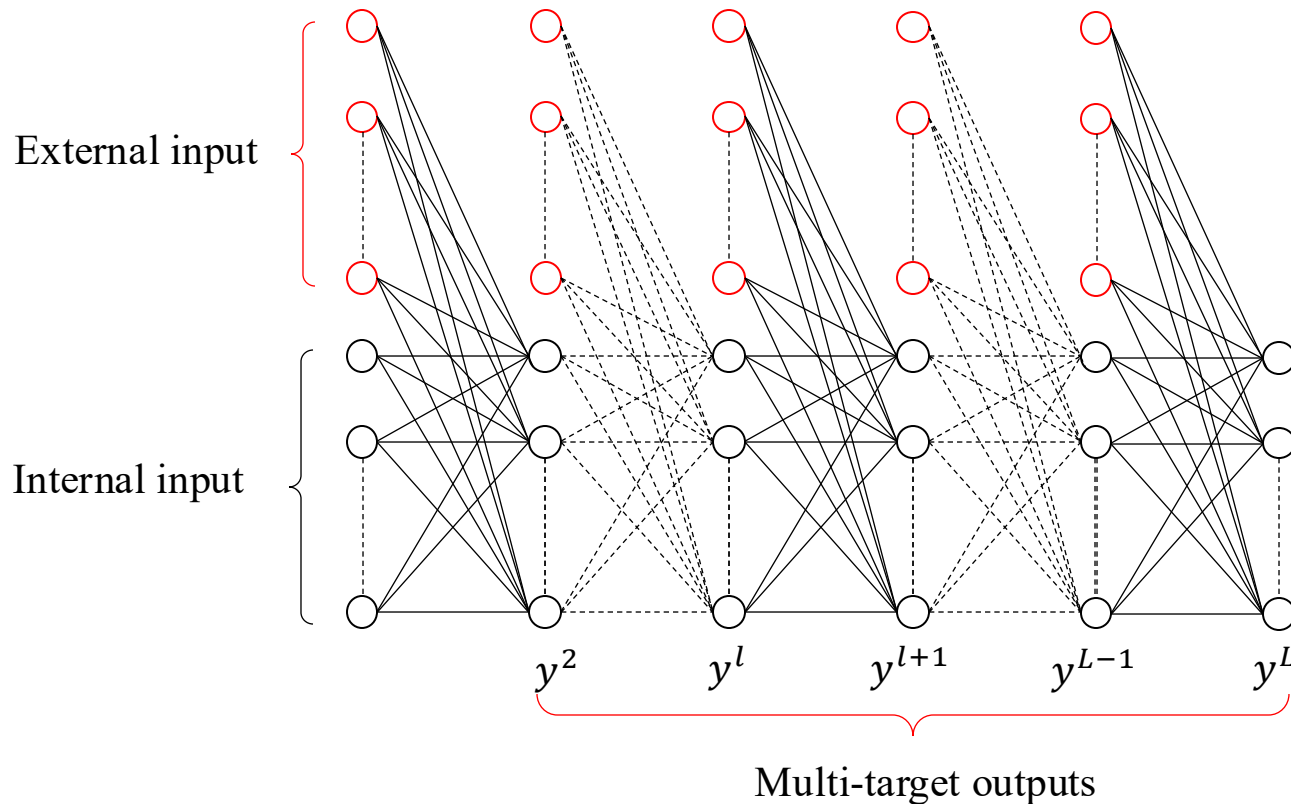
## □ A Sequence Recognizing Example



We already developed BP Algorithm for mono-target output.

# Sequence Learning

## □ A Sequence Recognizing Example



*Problem:*

Can we develop learning algorithms similar to BP for multi-target output?

Multi-target outputs

$$y^l = \begin{bmatrix} y_1^l \\ \vdots \\ y_{n_L}^l \end{bmatrix}$$

$$(l = 2, \dots, L)$$

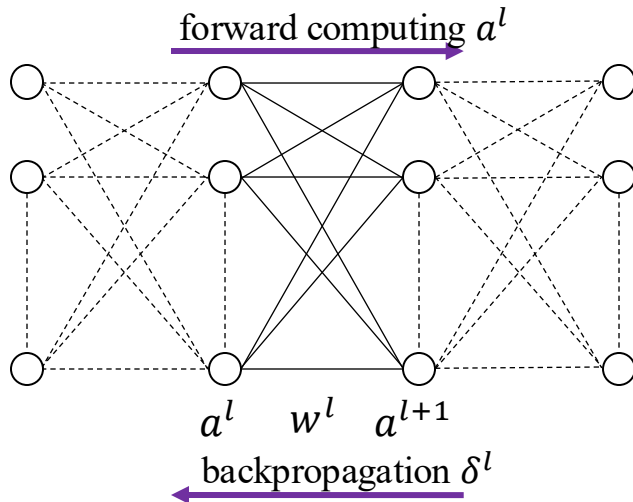
# Sequence Learning

## □ Review of BP algorithm for mono-output NNs

Cost function:  $J(w^1, \dots, w^{L-1})$

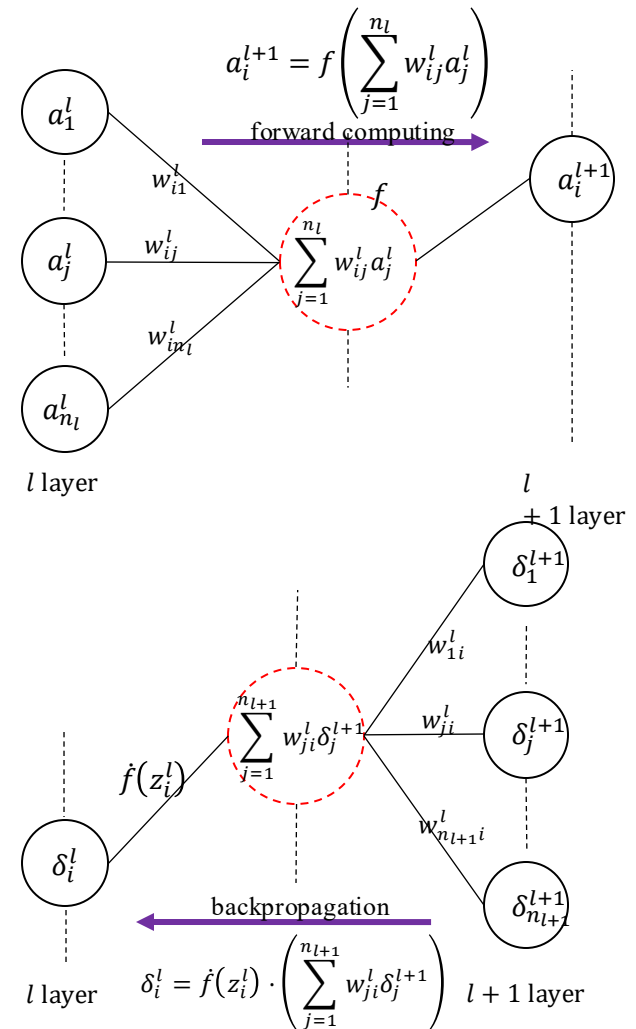
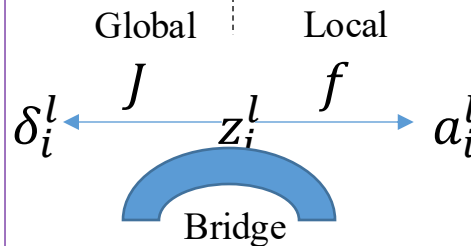
Updating rule:  $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

Relationship:  $\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$



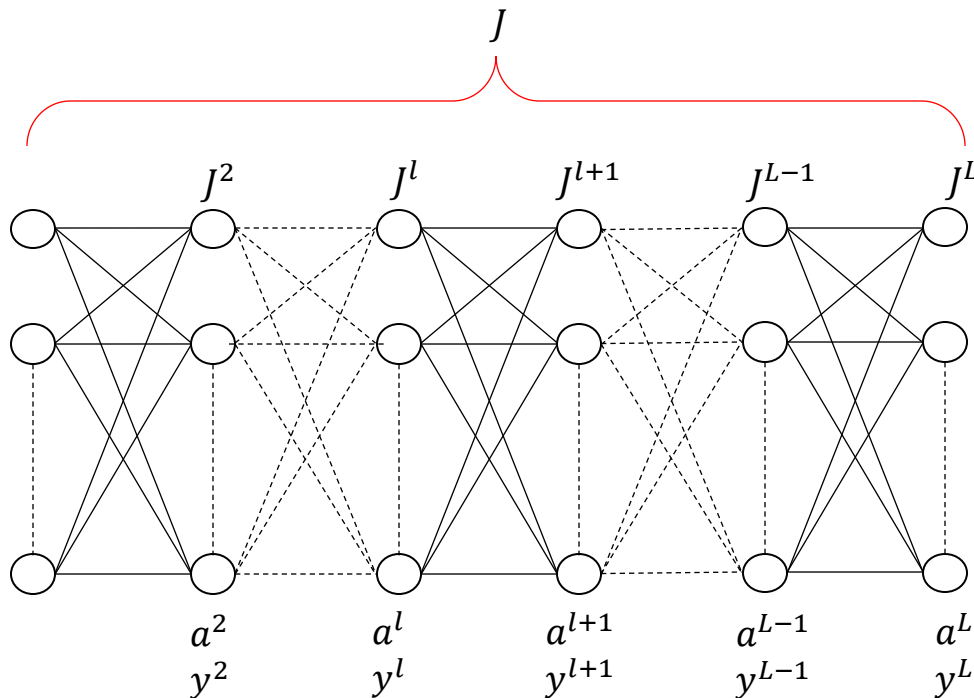
$l$  layer  $i^{th}$  neuron

$$\frac{a_i^l = f(z_i^l)}{\delta_i^l = \frac{\partial J}{\partial z_i^l}}$$



# Sequence Learning

## □ Review of BP algorithm for mono-output NNs



Cost function

$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

Network Outputs    Multi-target outputs

$$a^l = \begin{bmatrix} a_1^l \\ \vdots \\ a_{n_l}^l \end{bmatrix}$$

$$(l = 2, \dots, L)$$

$$y^l = \begin{bmatrix} y_1^l \\ \vdots \\ y_{n_l}^l \end{bmatrix}$$

$$(l = 2, \dots, L)$$

# Sequence Learning

## □ Review of BP algorithm for mono-output NNs

Steepest Descent Method

$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

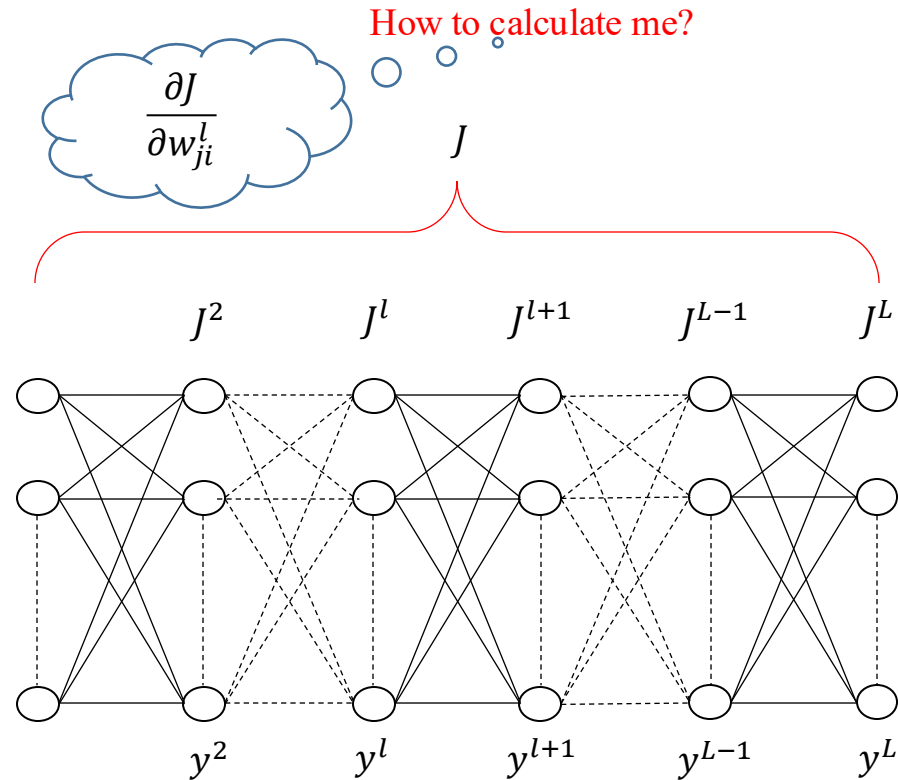
$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

1. Computing

$$\frac{\partial J}{\partial w_{ji}^l}$$

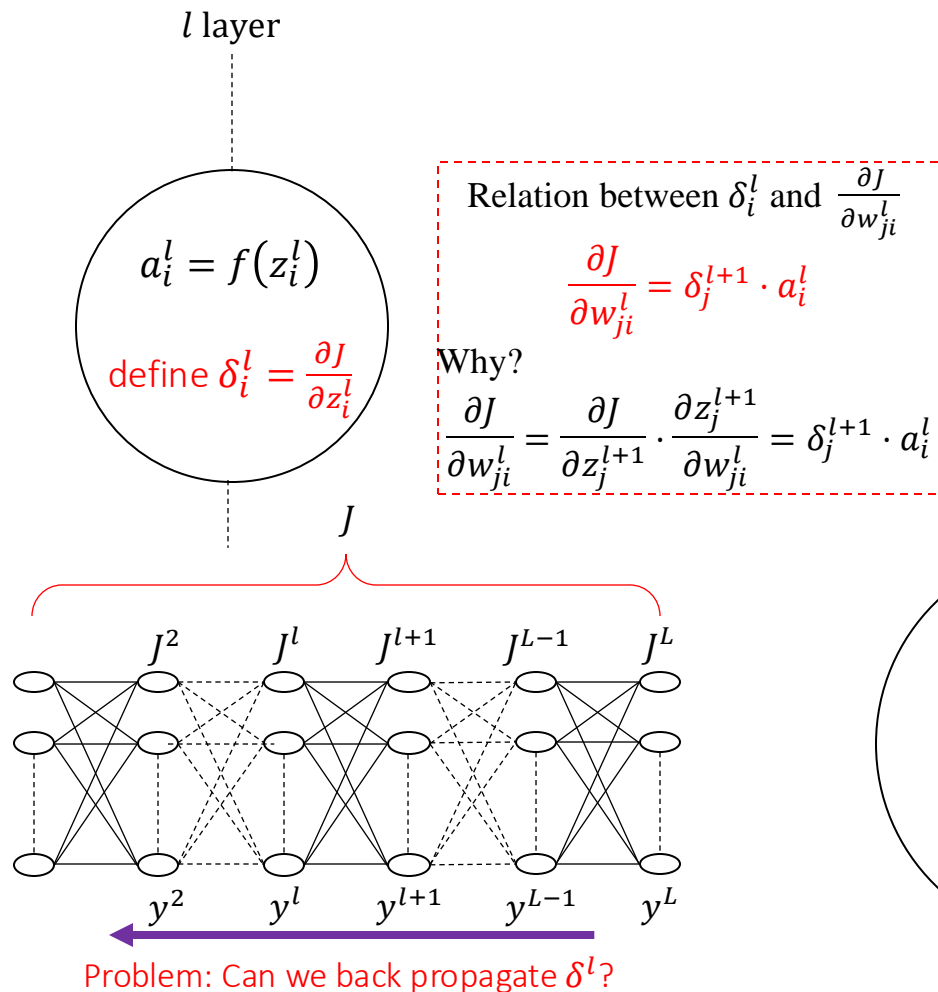
2. Iterating

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$



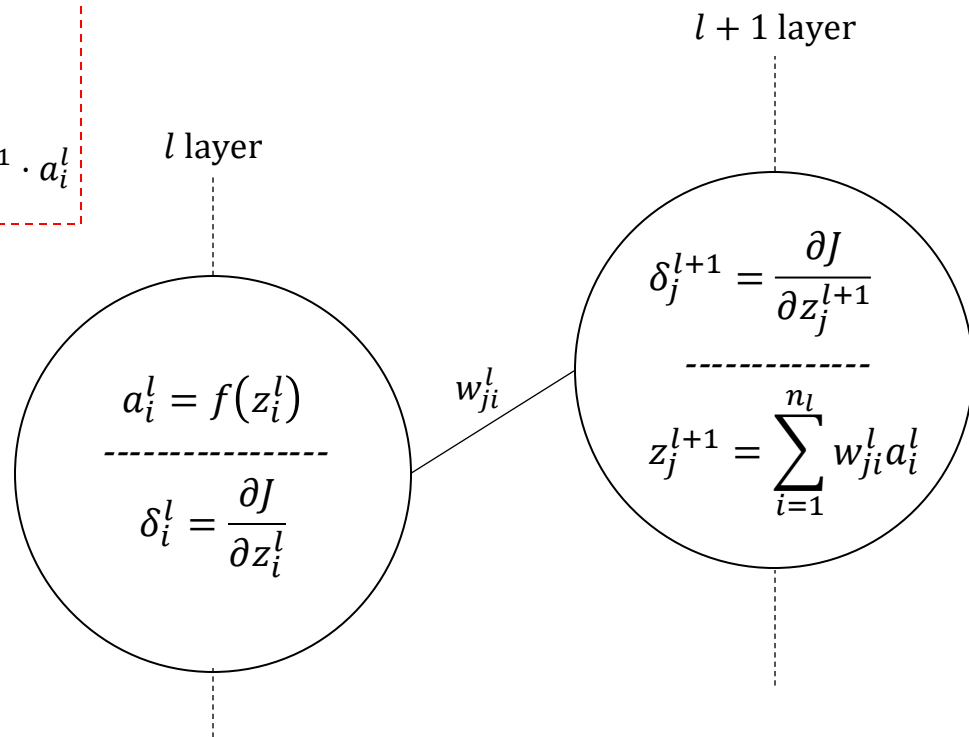
# Sequence Learning

## □ Review of BP algorithm for mono-output NNs



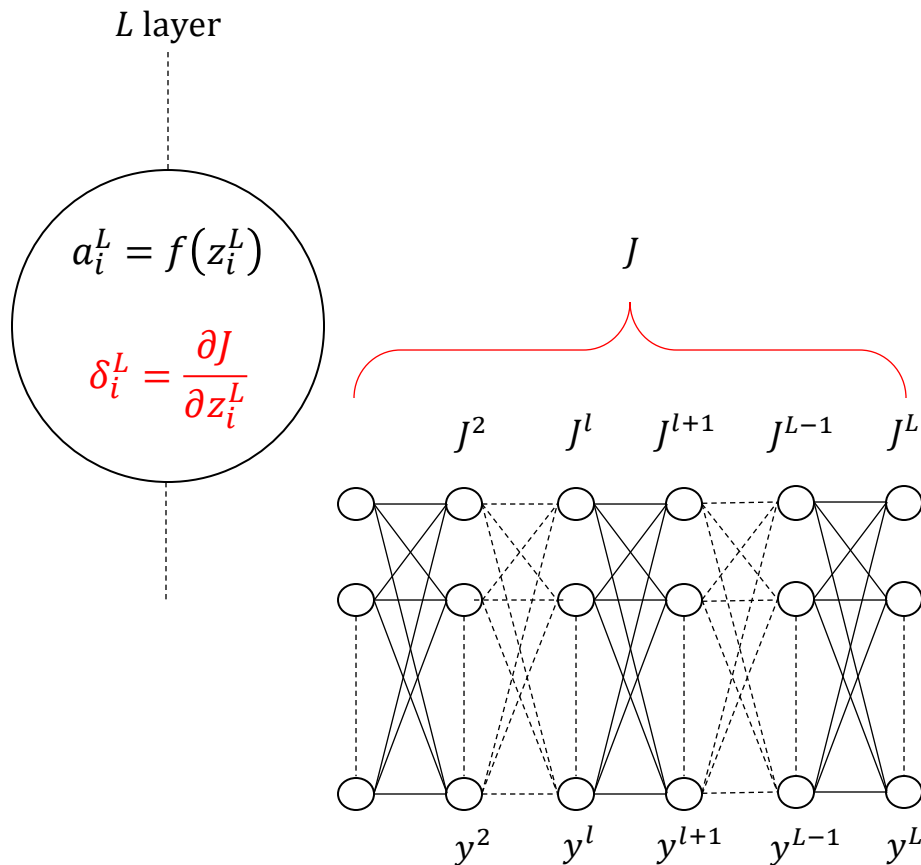
$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$



# Sequence Learning

## Step 1: Calculating $\delta^L$ in Last Layer



$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

It holds that,

$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \frac{\partial J^L}{\partial z_i^L} = (a_i^L - y_i^L) \cdot \frac{\partial a_j^L}{\partial z_i^L} = (a_i^L - y_i^L) \cdot f'(z_i^L)$$

# Sequence Learning

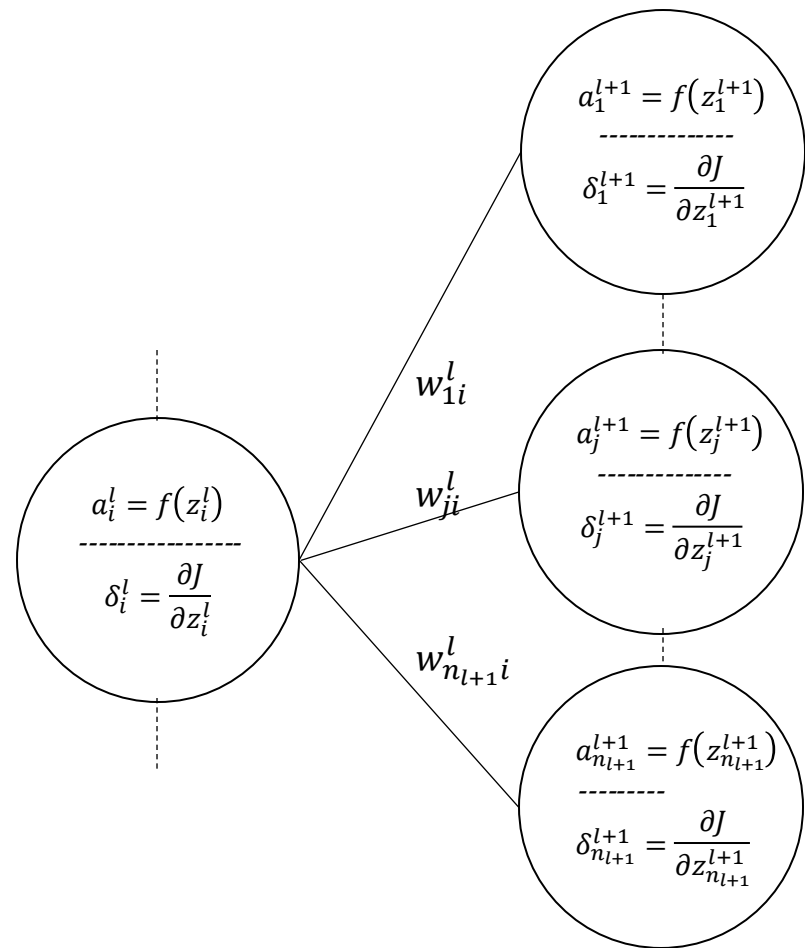
## □ Step 2: Relation Between $\delta^l$ and $\delta^{l+1}$

$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

$J$  may have an explicit dependence on  $z_i^l$ , it may also have an implicit dependence on  $z_i^l$  through later output values. To avoid ambiguity in interpreting partial derivatives, define  $z_i^l(*) = z_i^l$ .

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \frac{\partial J}{\partial z_i^l(*)} \cdot \frac{\partial z_i^l(*)}{\partial z_i^l} + \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l}$$



# Sequence Learning

## □ Step 2: Relation Between $\delta^l$ and $\delta^{l+1}$

$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

$J$  may have an explicit dependence on  $z_i^l$ , it may also have an implicit dependence on  $z_i^l$  through later output values. To avoid ambiguity in interpreting partial derivatives, define  $z_i^l(*) = z_i^l$ .

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \frac{\partial J}{\partial z_i^l(*)} \cdot \frac{\partial z_i^l(*)}{\partial z_i^l} + \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l}$$

An Illustrate Example

$$J = x + y, y = \exp(x)$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial x}$$

$$x^* = x$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial x^*} \cdot \frac{\partial x^*}{\partial x} + \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial x}$$

# Sequence Learning

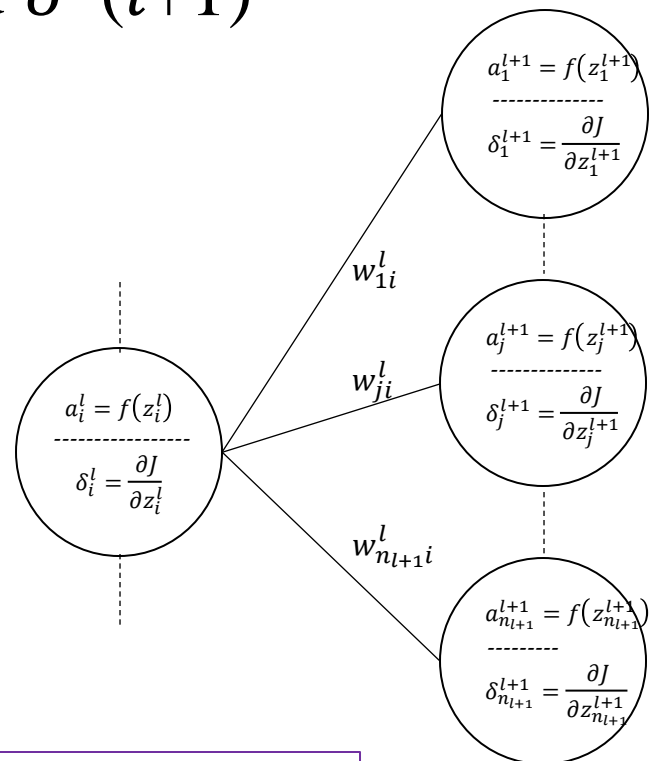
## □ Step 2: Relation Between $\delta^l$ and $\delta^{l+1}$

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \frac{\partial J}{\partial z_i^l(*)} \cdot \frac{\partial z_i^l(*)}{\partial z_i^l} + \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l}$$

$$\frac{\partial J}{\partial z_i^l(*)} \cdot \frac{\partial z_i^l(*)}{\partial z_i^l} = \frac{\partial J^l}{\partial z_i^l} = (a_i^l - y_i^l) \cdot f'(z_i^l)$$

$$\sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = f'(z_i^l) \cdot \left( \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$

$$\delta_i^l = f'(z_i^l) \cdot \left[ (a_i^l - y_i^l) + \left( \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right) \right]$$

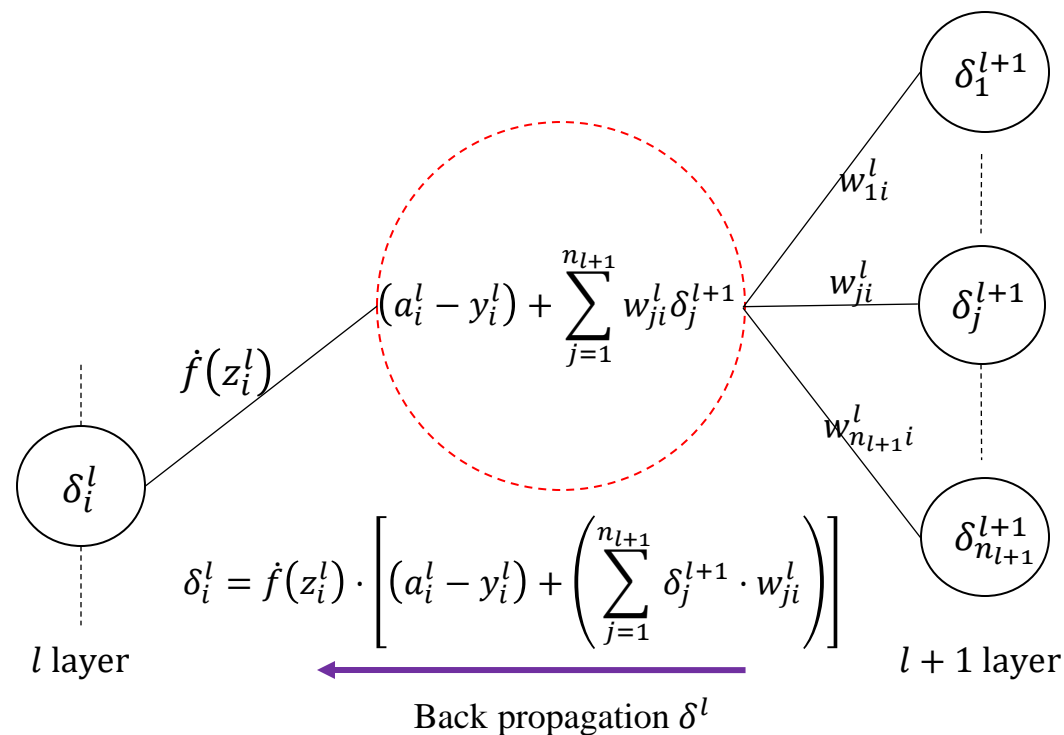


$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$

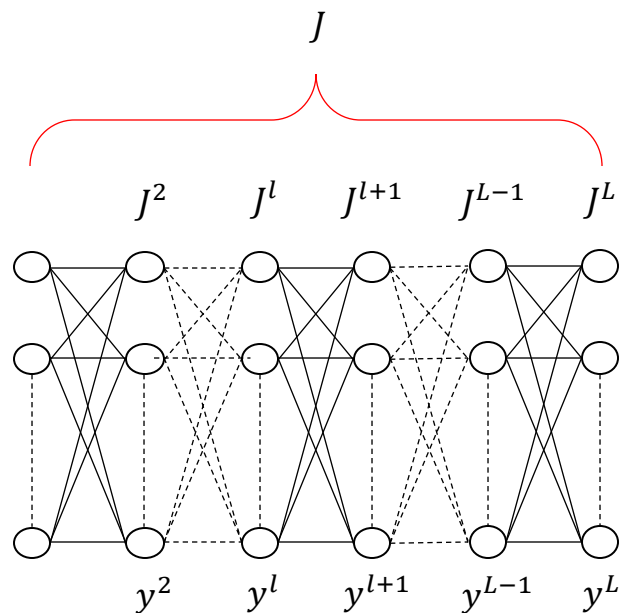
# Sequence Learning

## Step 3: Backpropagation $\delta$



$$J^l = \frac{1}{2} \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2, (l = 2, \dots, L)$$

$$J = \sum_{l=2}^L J^l = \frac{1}{2} \sum_{l=2}^L \sum_{i=1}^{n_l} (a_i^l - y_i^l)^2$$



# Sequence Learning

## □ The BP Algorithm

Step 1. Input the training data set  $D = \{(x, y)\}$

Step 2. Initial each  $w_{ij}^l$ , and choose a learning rate  $\alpha$ .

Step 3. For each mini-batch sample  $D_m \subseteq D$

```

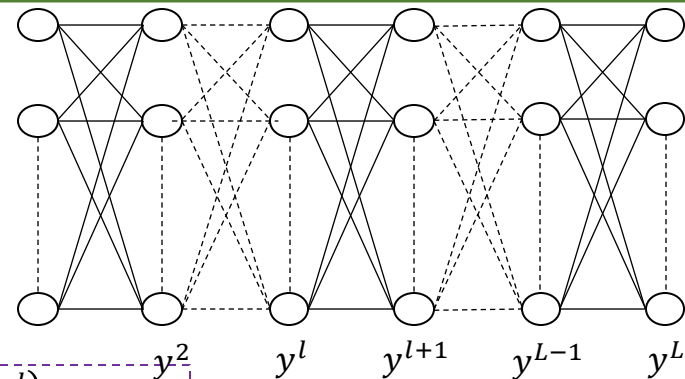
 $a^1 \leftarrow x \in D_m;$ 
  for  $l = 1:L - 1$ 
     $fc(w^l, a^l);$ 
  end
   $\delta^L = \frac{\partial J}{\partial z^L};$ 
  for  $l = L - 1:2$ 
     $bc(w^l, \delta^{l+1});$ 
  end
   $\frac{\partial J}{\partial w_{ji}^l} \leftarrow \frac{\partial J}{\partial w_{ji}^l} + \delta_j^{l+1} \cdot a_i^l;$ 

```

Step 4. Updating

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l};$$

Step 5. Return to Step 3 until each  $w^l$  converge.



function  $fc(w^l, a^l)$

for  $i = 1:n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f(z_i^{l+1})$$

end

Relationship:

$$\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$$

function  $bc(w^l, \delta^{l+1})$

for  $i = 1:n_l$

$$\delta_i^l = f'(z_i^l) \cdot \left[ (a_i^l - y_i^l) + \left( \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right) \right]$$

end

# Sequence Learning

## □ Illustrative Example

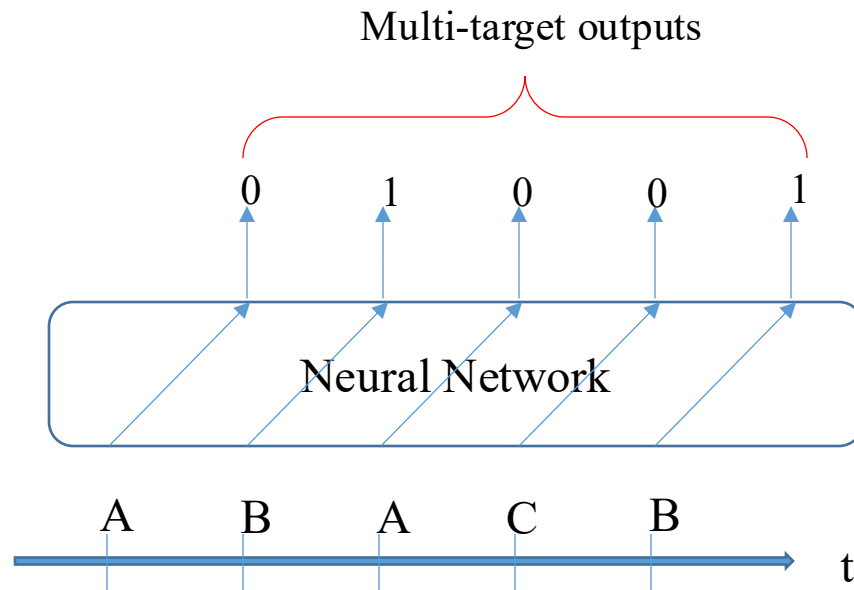
### *Recognize A followed by B Problem*

The task is to recognize A followed by B.

#### Generated Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC

.....



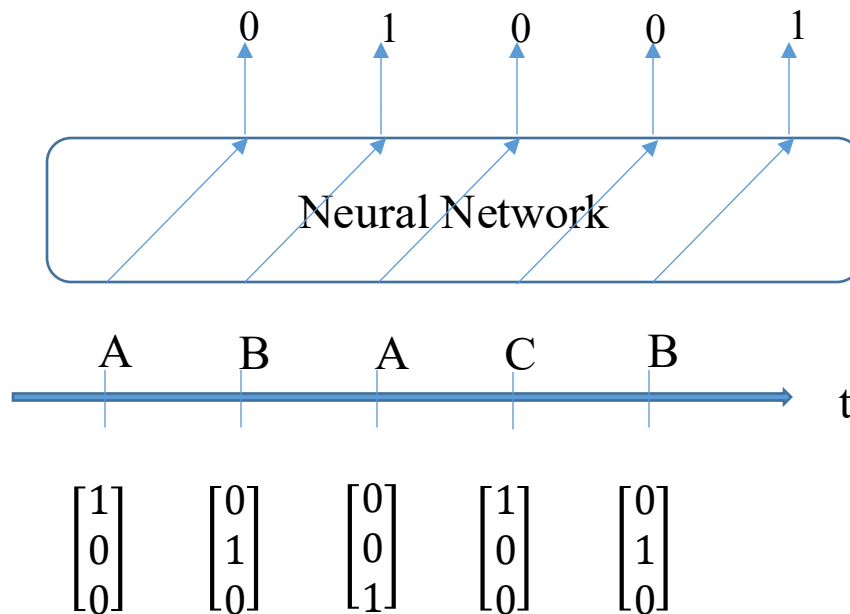
# Sequence Learning

## □ Illustrative Example

$$A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

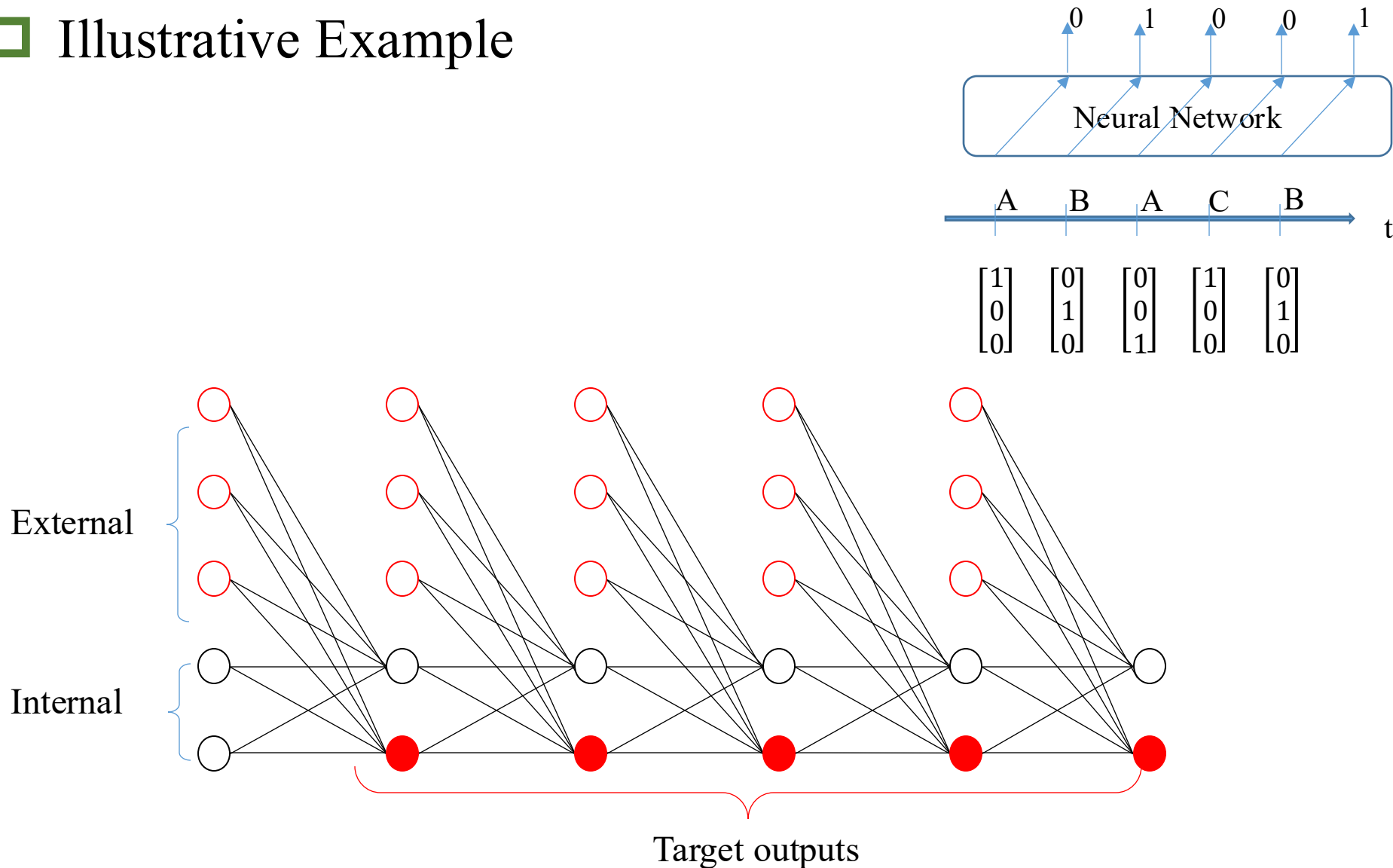
Generated Sequences

1.    A   B   C   A   B  
       $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$
2.    C   A   C   C   B  
       $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$



# Sequence Learning

## ■ Illustrative Example



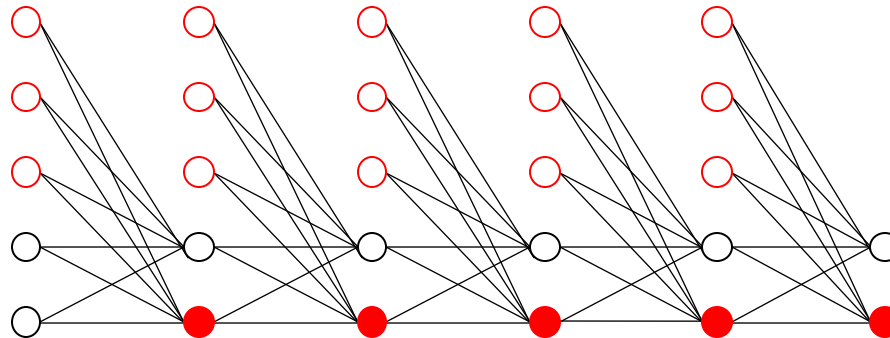
# Sequence Learning

## ■ Illustrative Example

### Generated Training Sequences

1. ABCAB
2. CCBBA
3. CACCB
4. ACCCB
5. CACBC
6. AAACB
7. BAACB
8. CCBAB
9. BCCAB
10. CABAC

.....



### Generated Testing Sequences

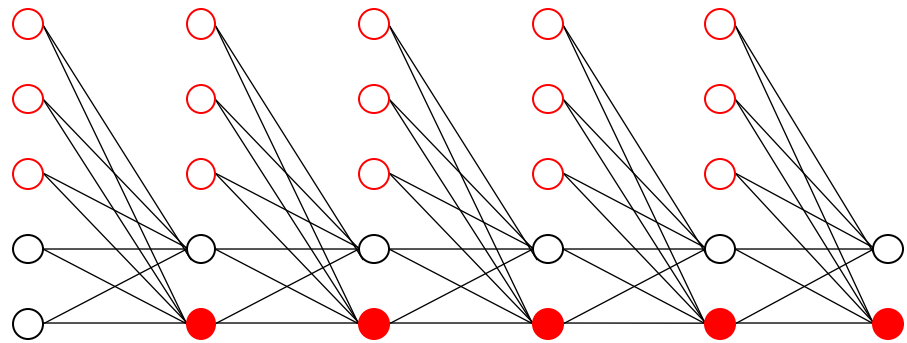
1. CBCAC
2. ACBBA
3. BACCB
4. ACBCB
5. AACBC
6. BAACB
7. AAACB
8. CCBAB
9. BBCAB
10. AABAC

.....

# Sequence Learning

## □ Illustrative Example

$$A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Example outputs:

CAAC <b>B</b>	→	0.0184	0.0001	0.0211	0.0801	<b>0.9928</b>
A <b>B</b> BCA	→	0.0179	<b>0.9375</b>	0.0267	0.0012	0.0000
AAC <b>B</b> A	→	0.0179	0.0336	0.0286	<b>0.8722</b>	0.0000
CAC <b>B</b> B	→	0.0184	0.0001	0.0170	<b>0.8494</b>	0.0013
BCAAA	→	0.0182	0.0001	0.0001	0.0622	0.0018

# Sequence Learning

## □ Another Example: Image Caption

### *Image Caption:*

The task is to describe the content of an image using properly formed English sentence.



image



English sentence

a cute panda lies on a tree

# Sequence Learning

## □ Another Example: Image Caption

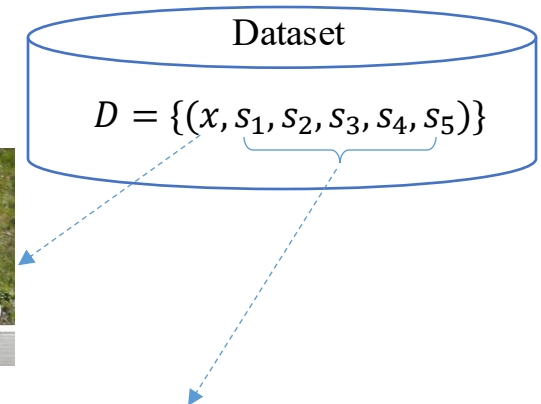
Dataset: COCO

COCO is a new image recognition, segmentation, and captioning dataset sponsored by Microsoft.

<http://mscoco.org/dataset/#download>

There are:

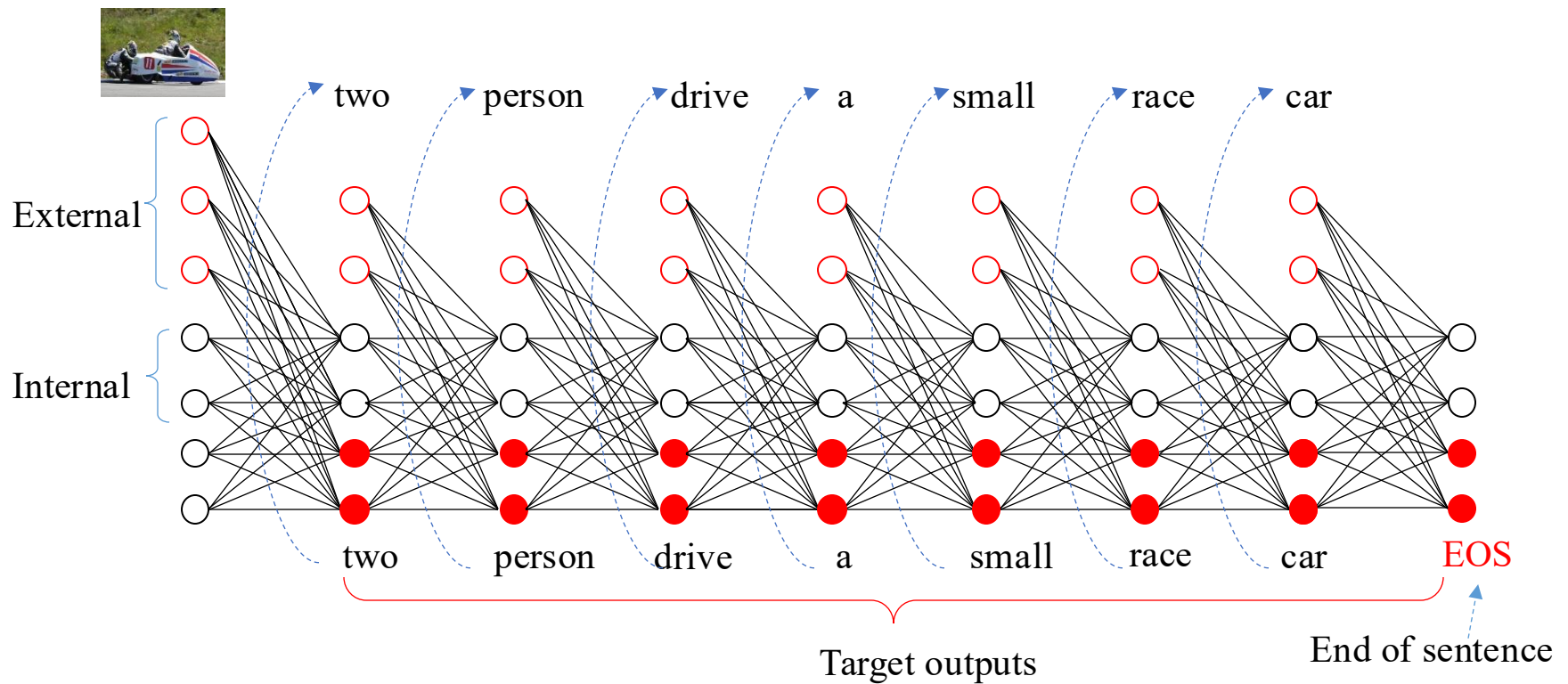
- 80,000 training samples
- 40,000 validation samples
- 40,000 test samples



1. Two person drive a small race car .
2. Two racer drive a white bike down a road .
3. Two motorist be ride along on their vehicle that be oddly design and color .
4. Two person be in a small race car drive by a green hill .
5. Two person in race uniform in a street car .

# Sequence Learning

## □ Another Example: Image Caption

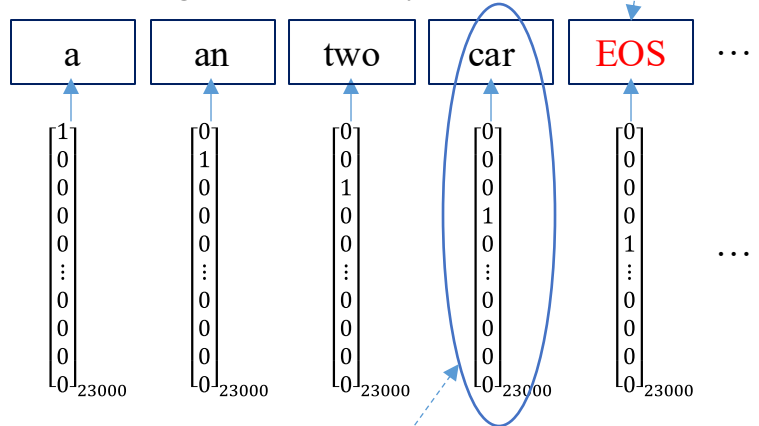


# Sequence Learning

## □ Another Example: Image Caption

Coding the Inputs:

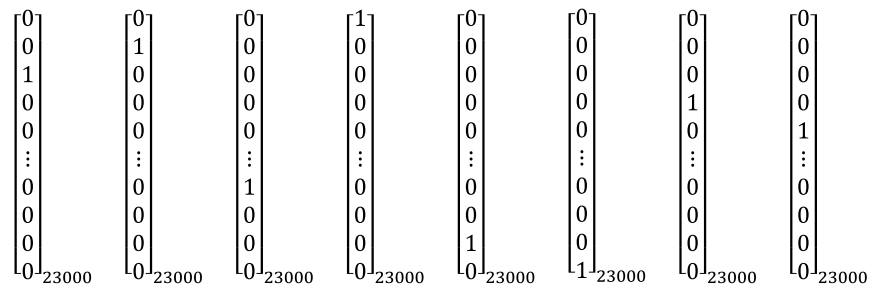
1. building the vocabulary



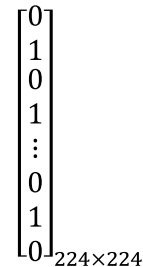
One-hot word vector

2. coding the sentence

$s =$  two person drive a small race car EOS

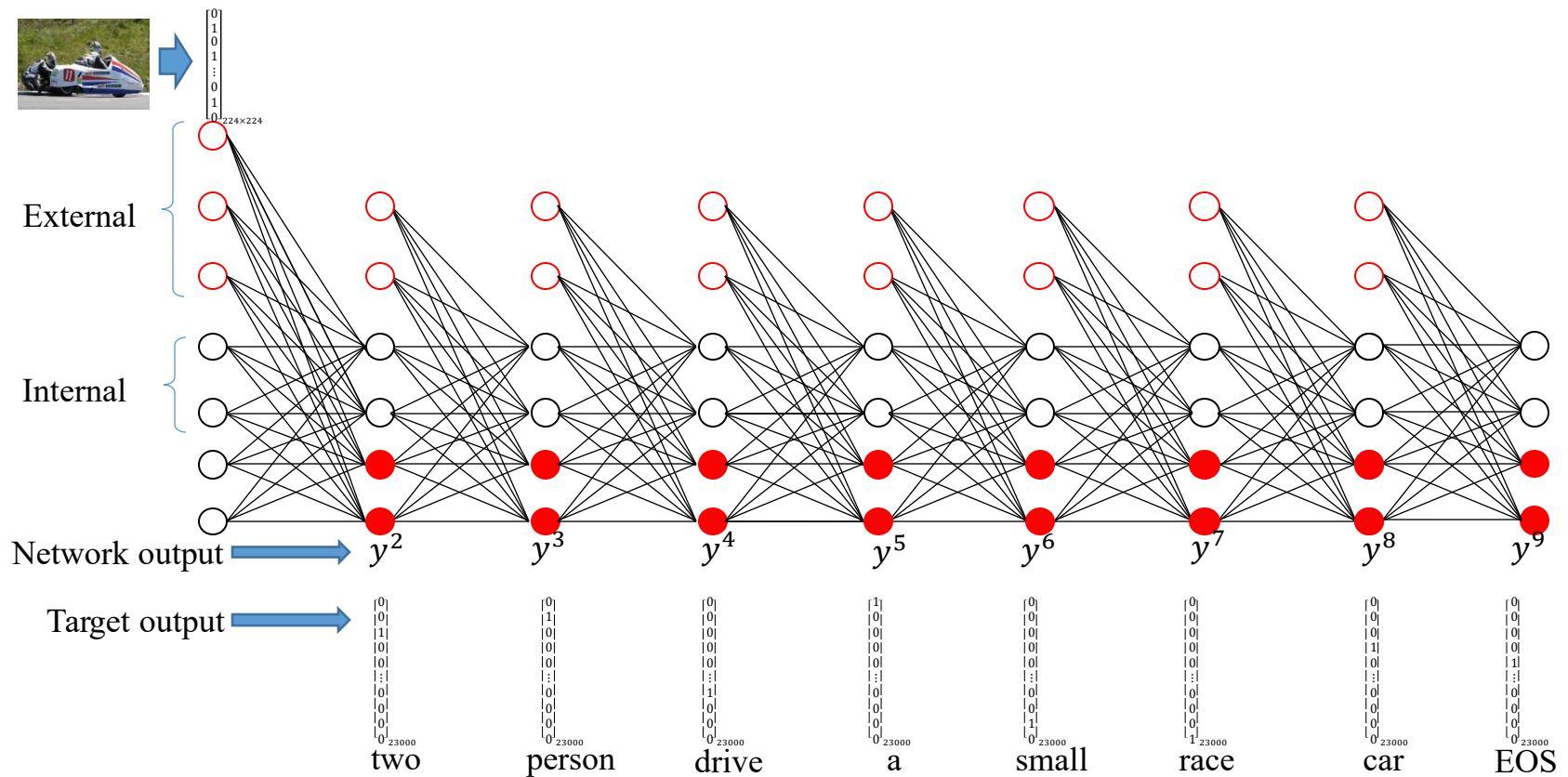


3. digitizing the image



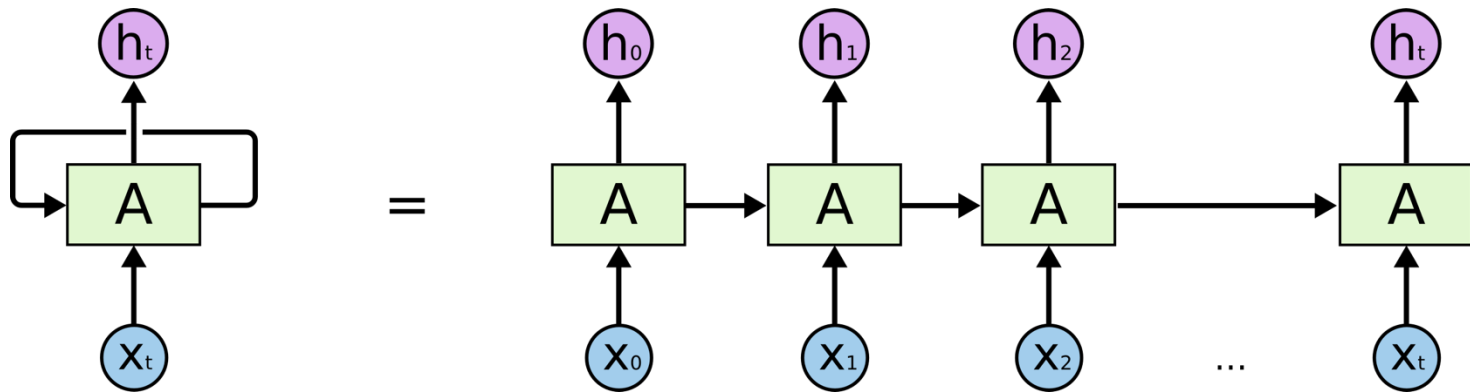
# Sequence Learning

## □ Another Example: Image Caption



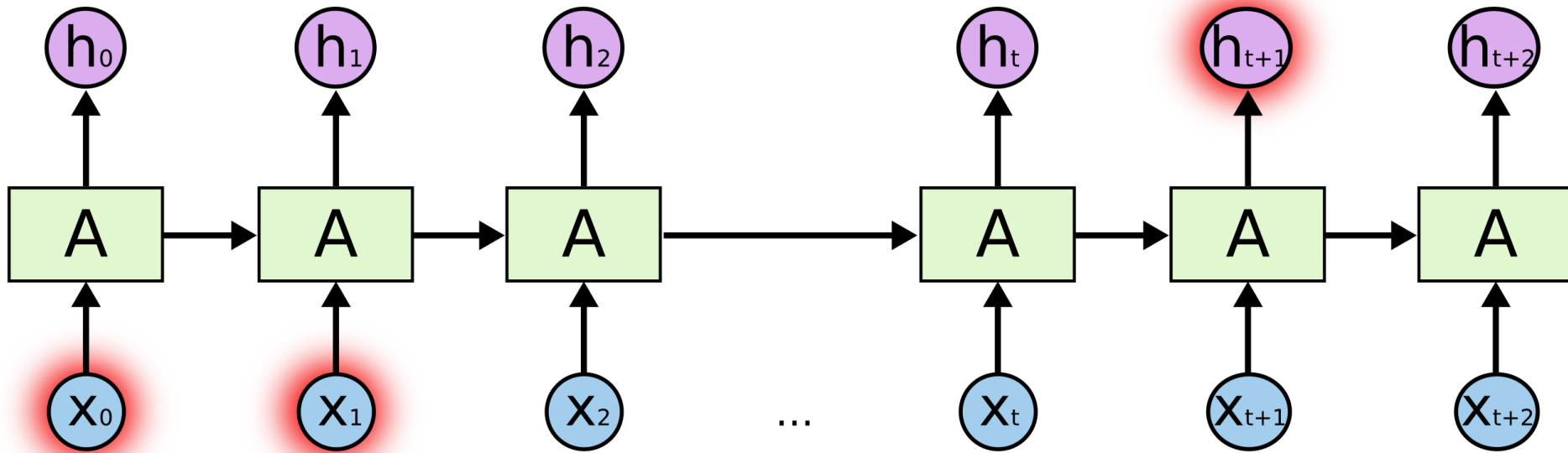
# Sequence Learning

## □ Other RNN



# Sequence Learning

## Other RNN



长期依赖(Long Term Dependencies)

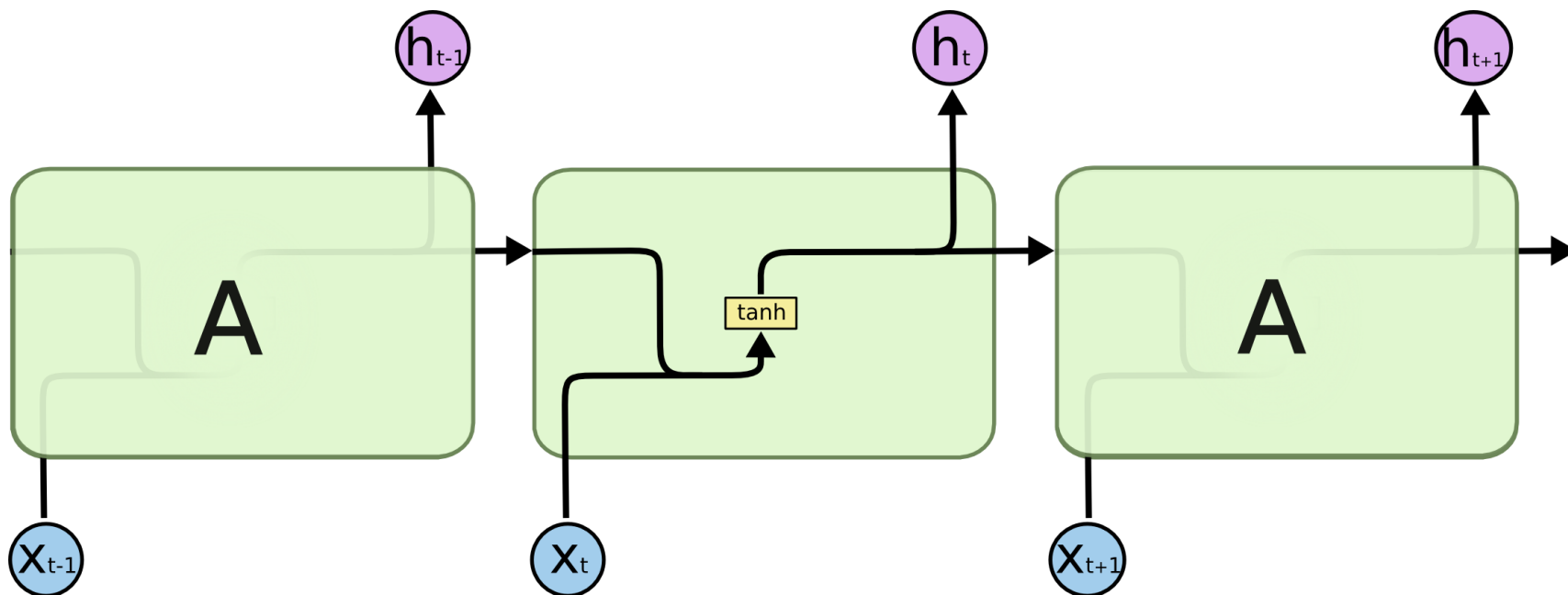
eg1: The **cat**, which already ate a bunch of food, **was** full.

eg2: The **cats**, which already ate a bunch of food, **were** full.

# Sequence Learning

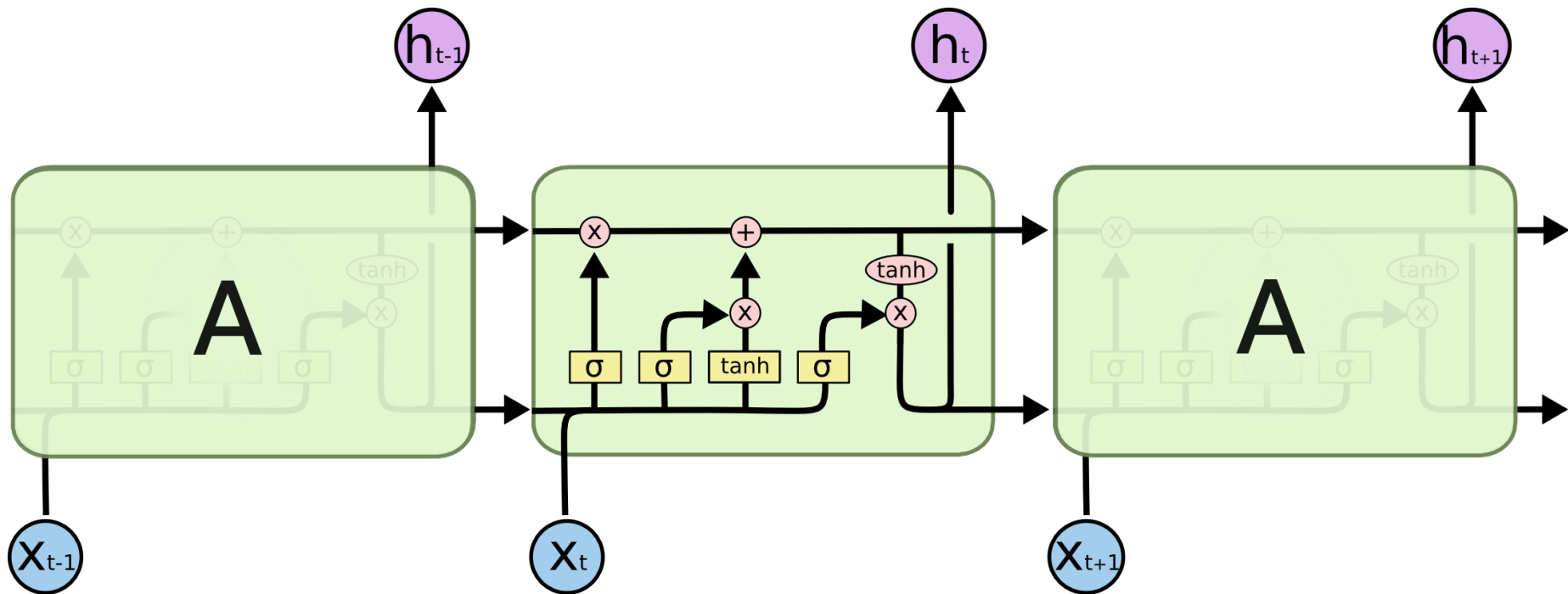
## □ Other RNN - LSTM

LSTM: Long Short Term Memory, 顾名思义, 它具有记忆长短  
期信息的能力的神经网络



# Sequence Learning

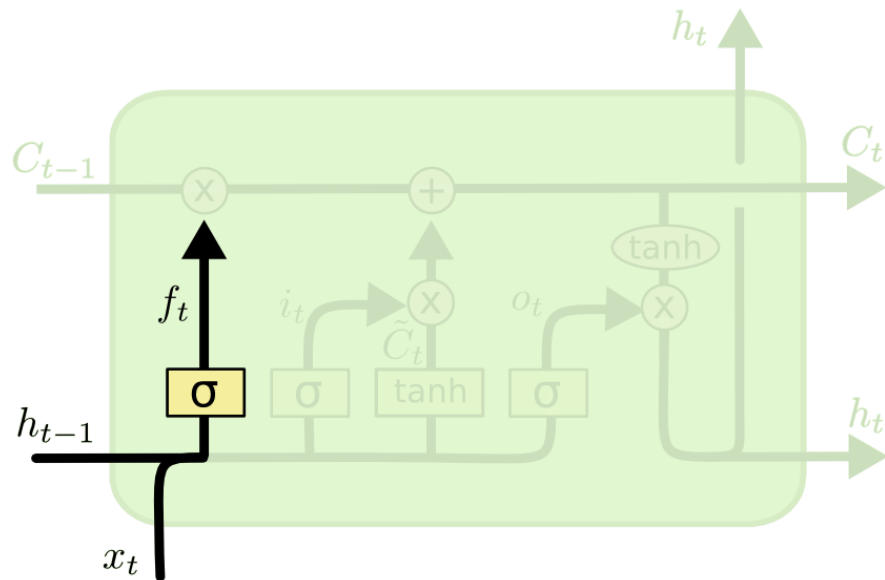
## Other RNN - LSTM



LSTM解决RNN的长期依赖问题:LSTM引入了门 (gate) 机制用于控制特征的流通和损失

# Sequence Learning

## □ Other RNN - LSTM

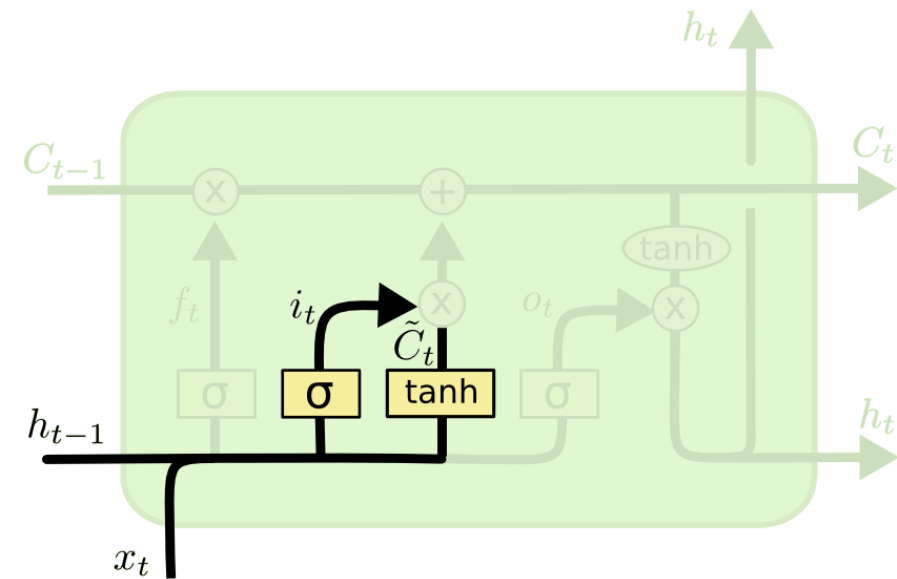


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

$f_t$  遗忘门

# Sequence Learning

## Other RNN - LSTM

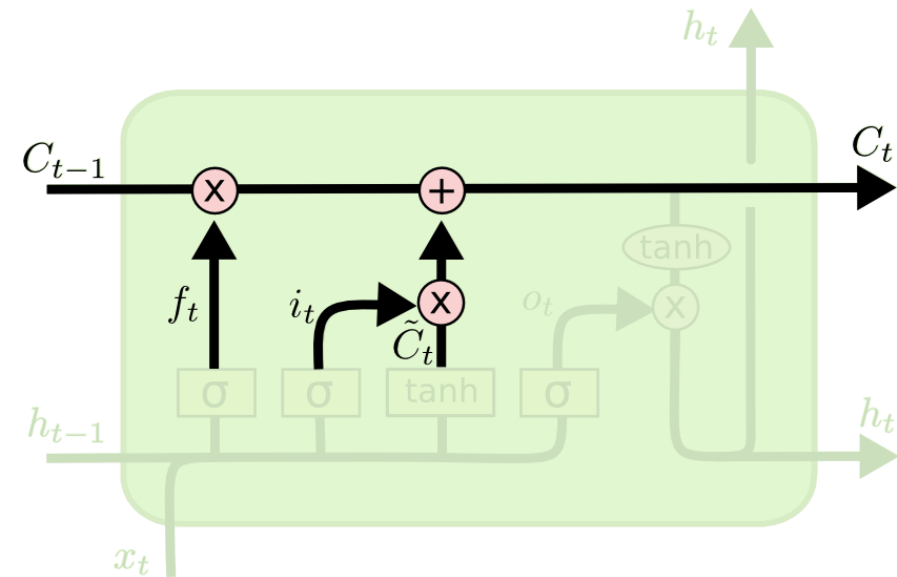


$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

# Sequence Learning

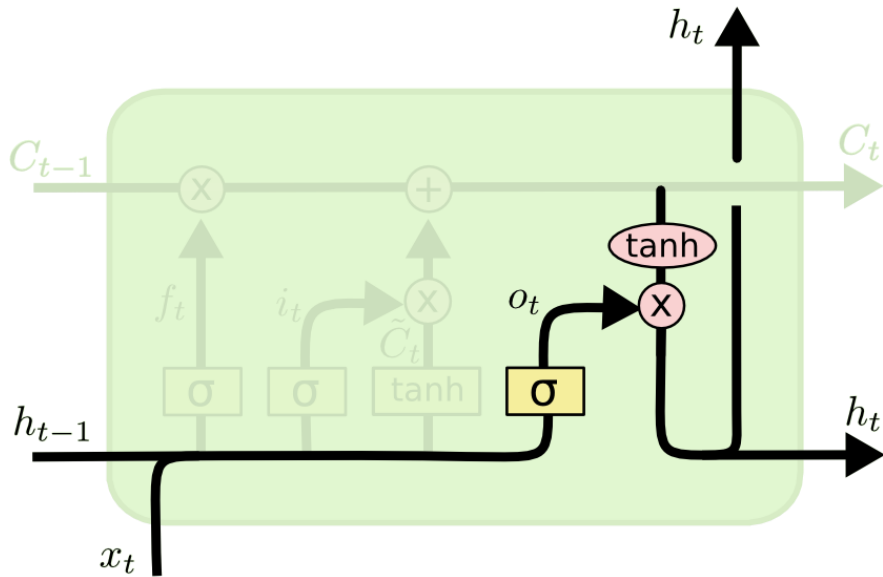
## Other RNN - LSTM



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# Sequence Learning

## □ Other RNN - LSTM

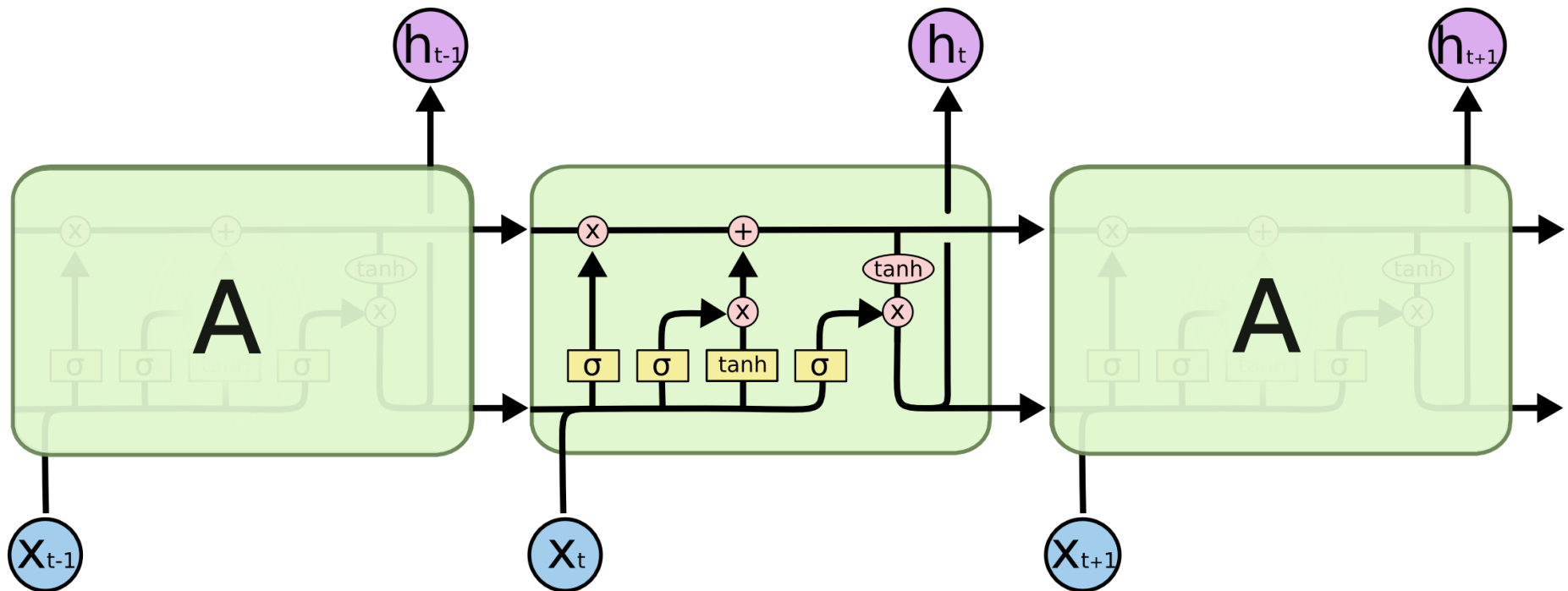


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

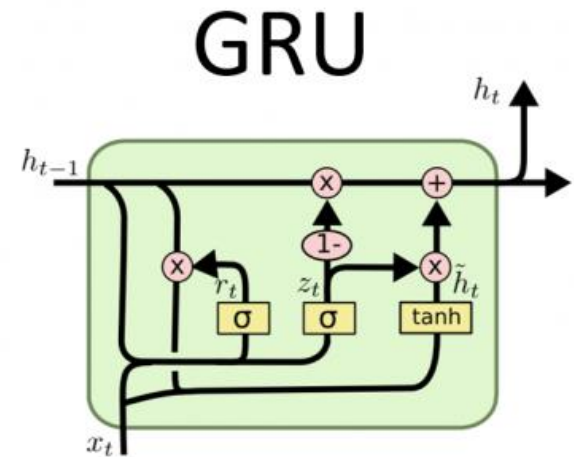
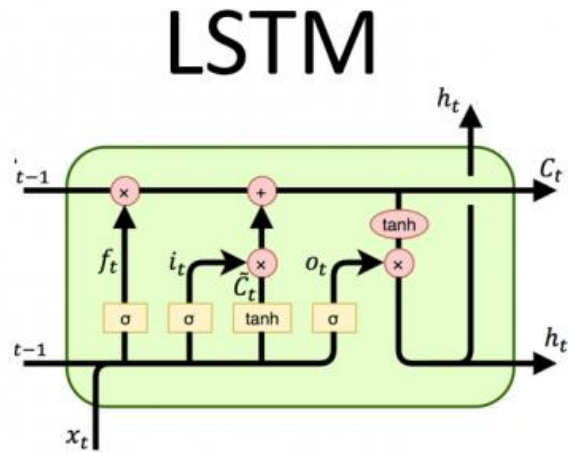
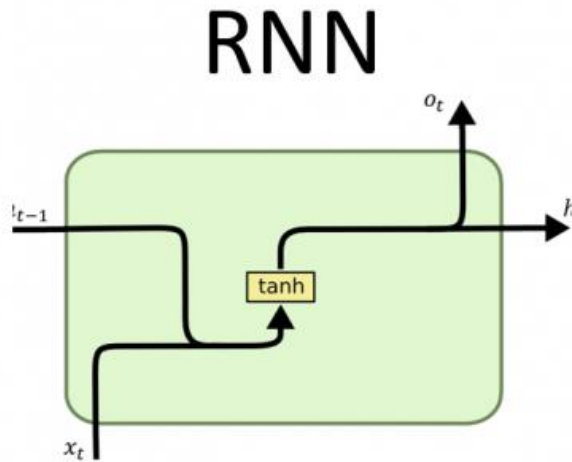
# Sequence Learning

## Other RNN - LSTM



# Sequence Learning

## Other RNNs



# Neural Networks

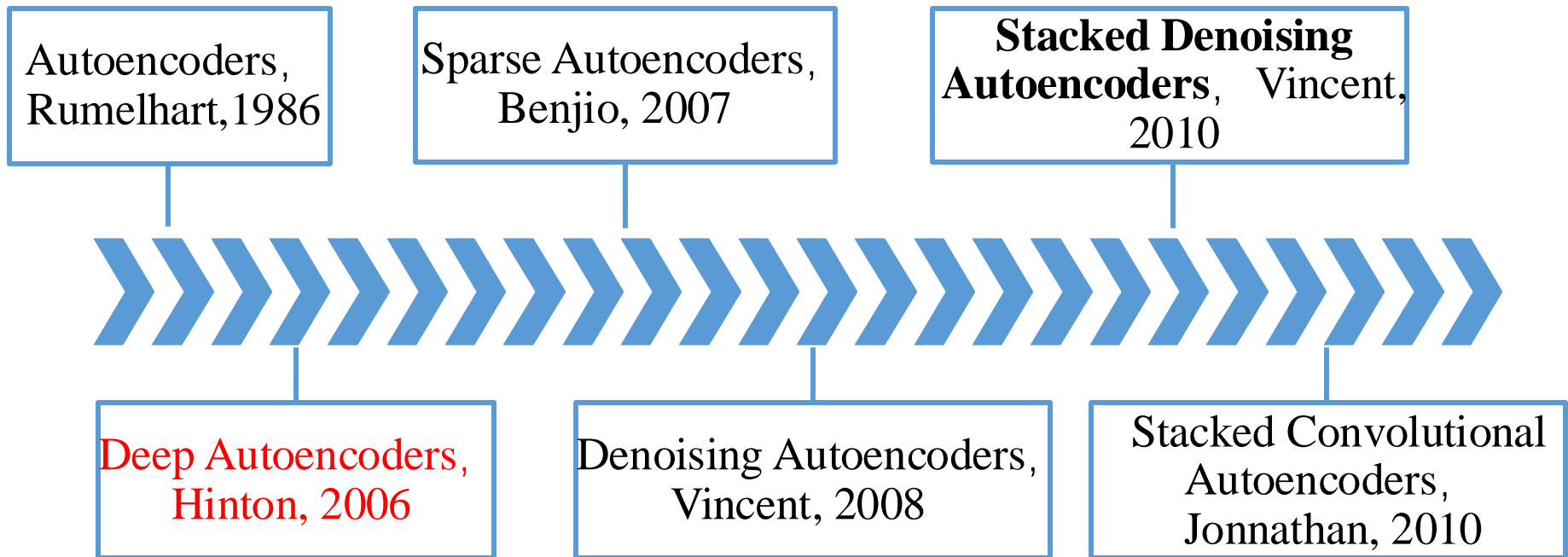
- Brief review
- Sequence Learning
- *Representation learning*



# Representation learning

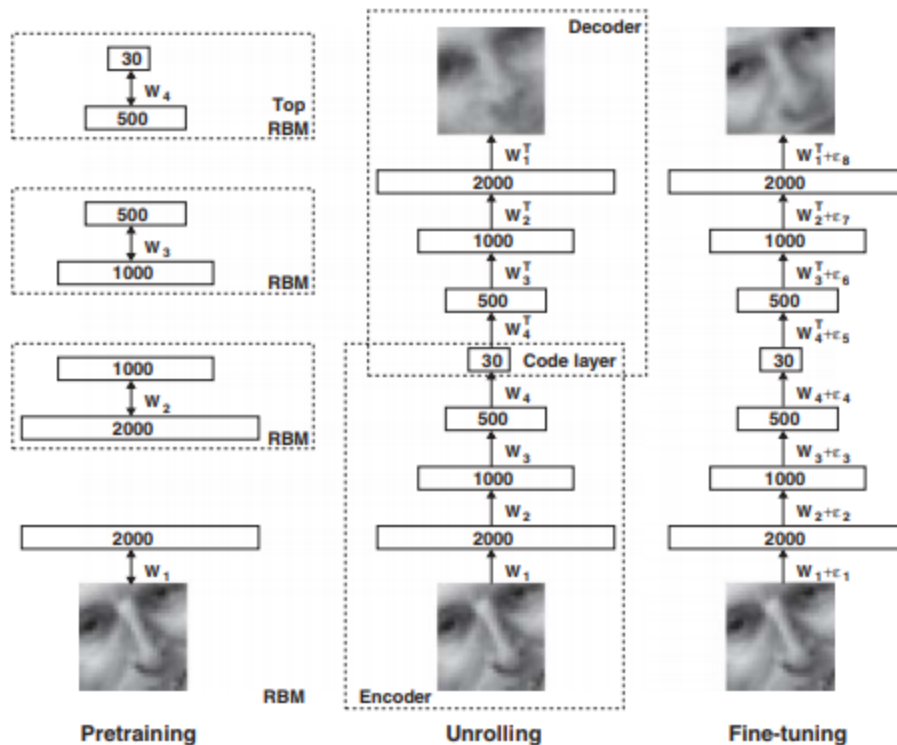
---

## □ Autoencoders



# Representation learning

## Autoencoders



**Fig. 1.** Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

## Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton\* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which

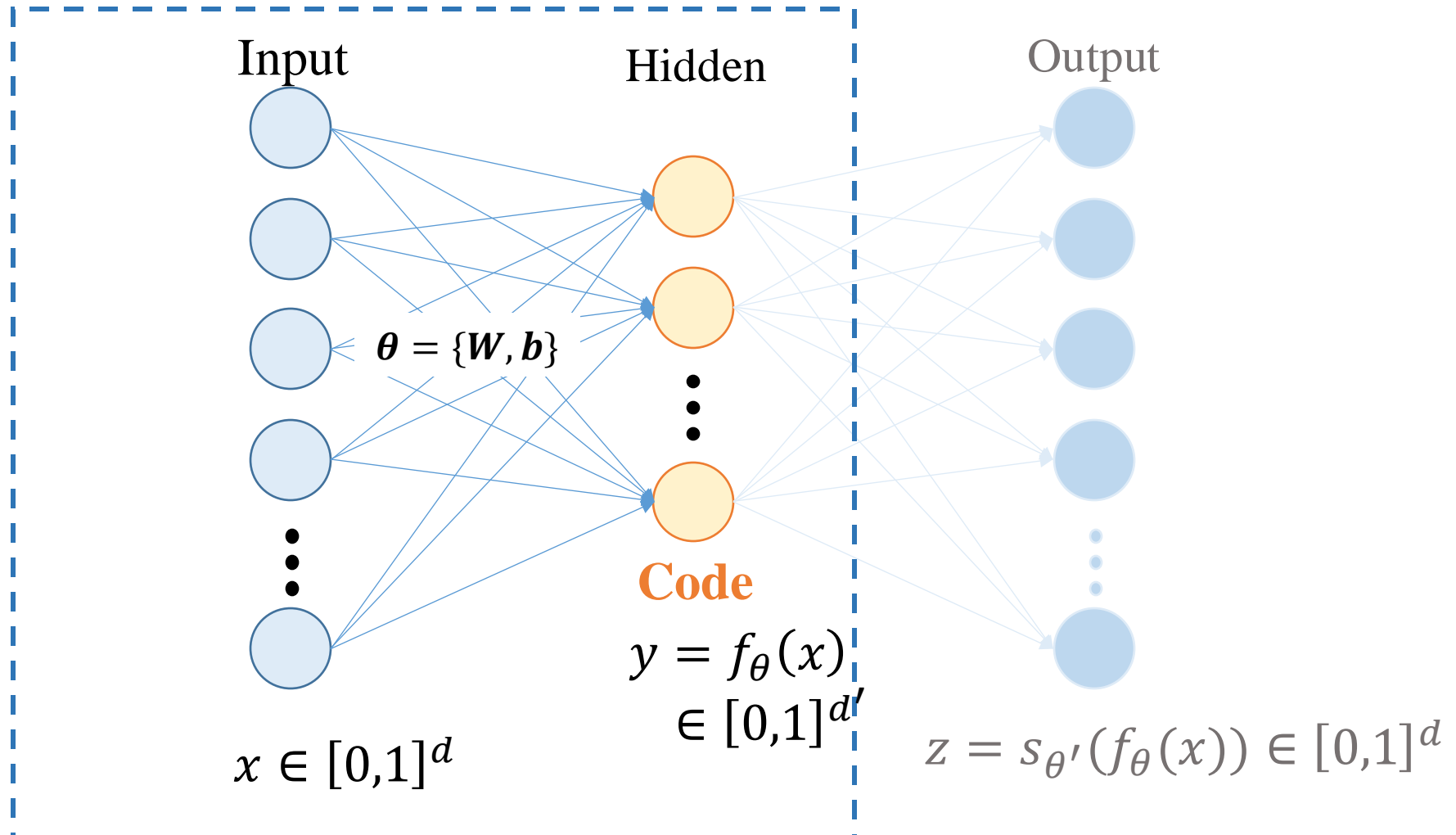
finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

# Representation learning

## □ Autoencoders

### Encoder

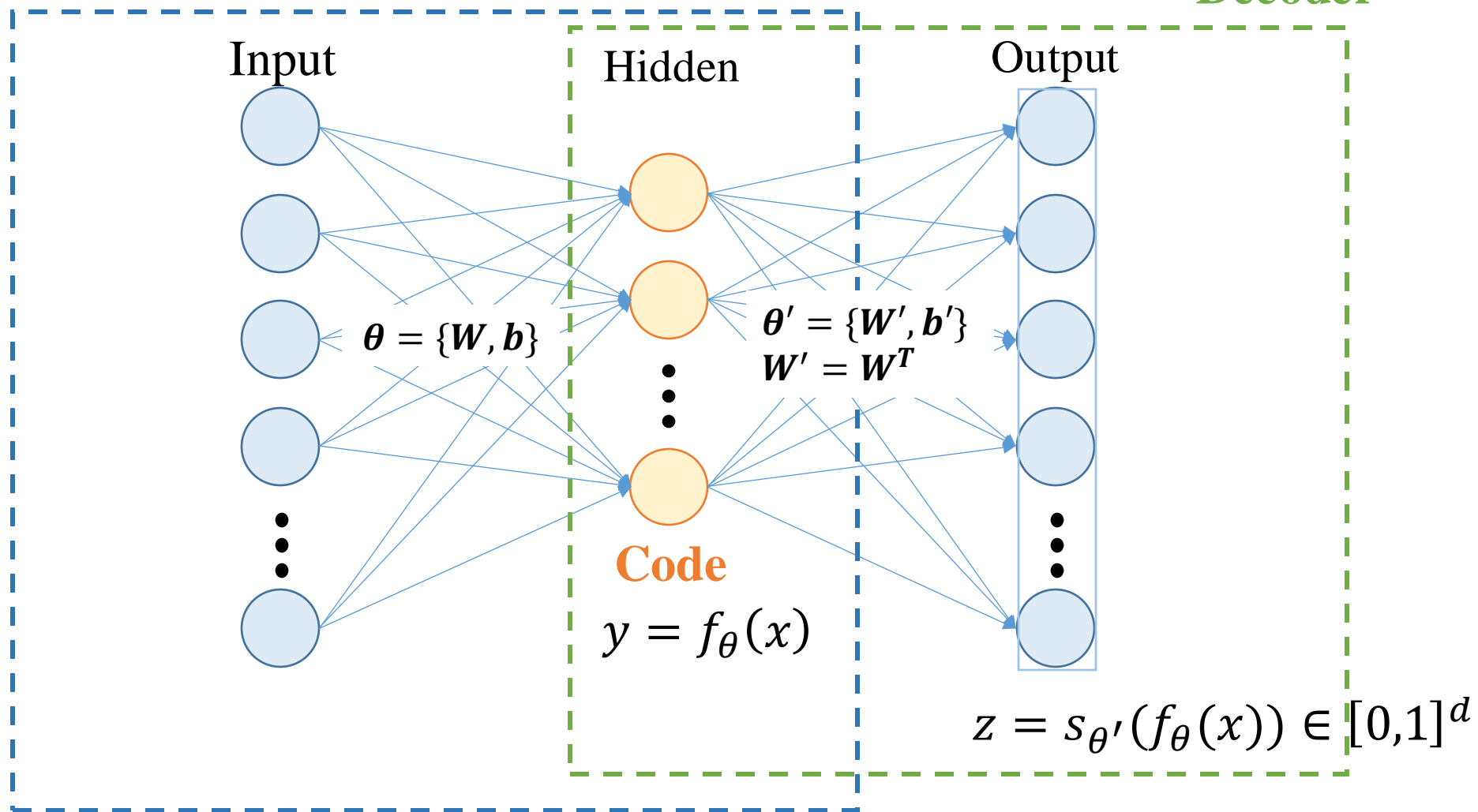
$d' < d \longrightarrow$  dimensionality reduction



# Representation learning

## Autoencoders

### Encoder

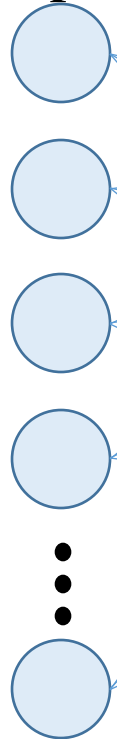


# Representation learning

## Autoencoders

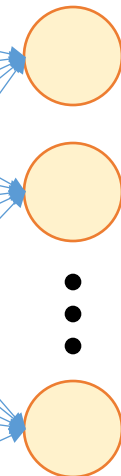
### Encoder

Input



$$\theta = \{W, b\}$$

Hidden



Code

$$y = f_{\theta}(x)$$

$$\theta' = \{W', b'\}$$
$$W' = W^T$$

Output



Decoder

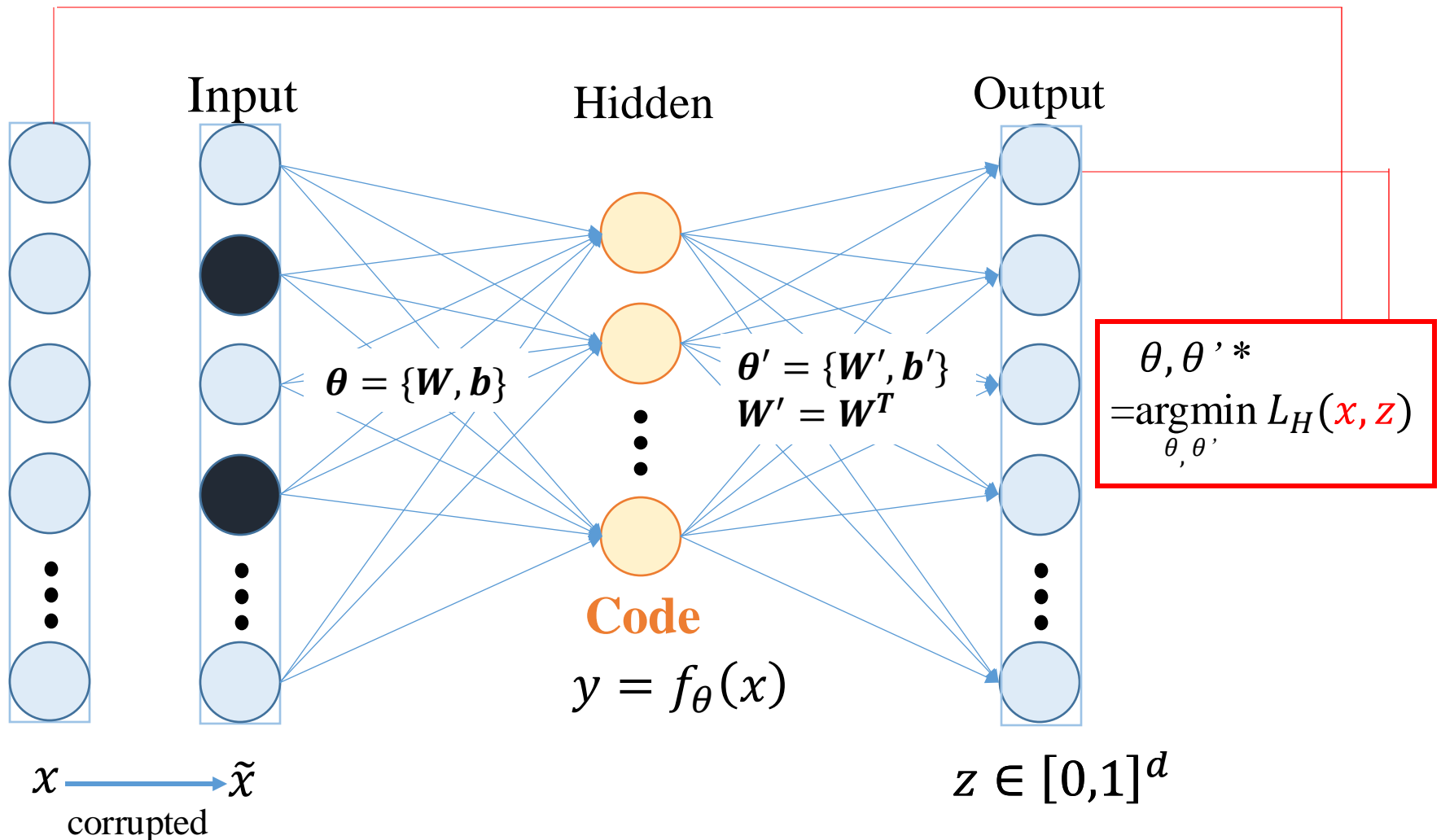
$$\theta^*, \theta'^*$$
$$= \underset{\theta, \theta'}{\operatorname{argmin}} L_H(x, z)$$

$$z = s_{\theta'}(f_{\theta}(x)) \in [0, 1]^d$$

# Representation learning

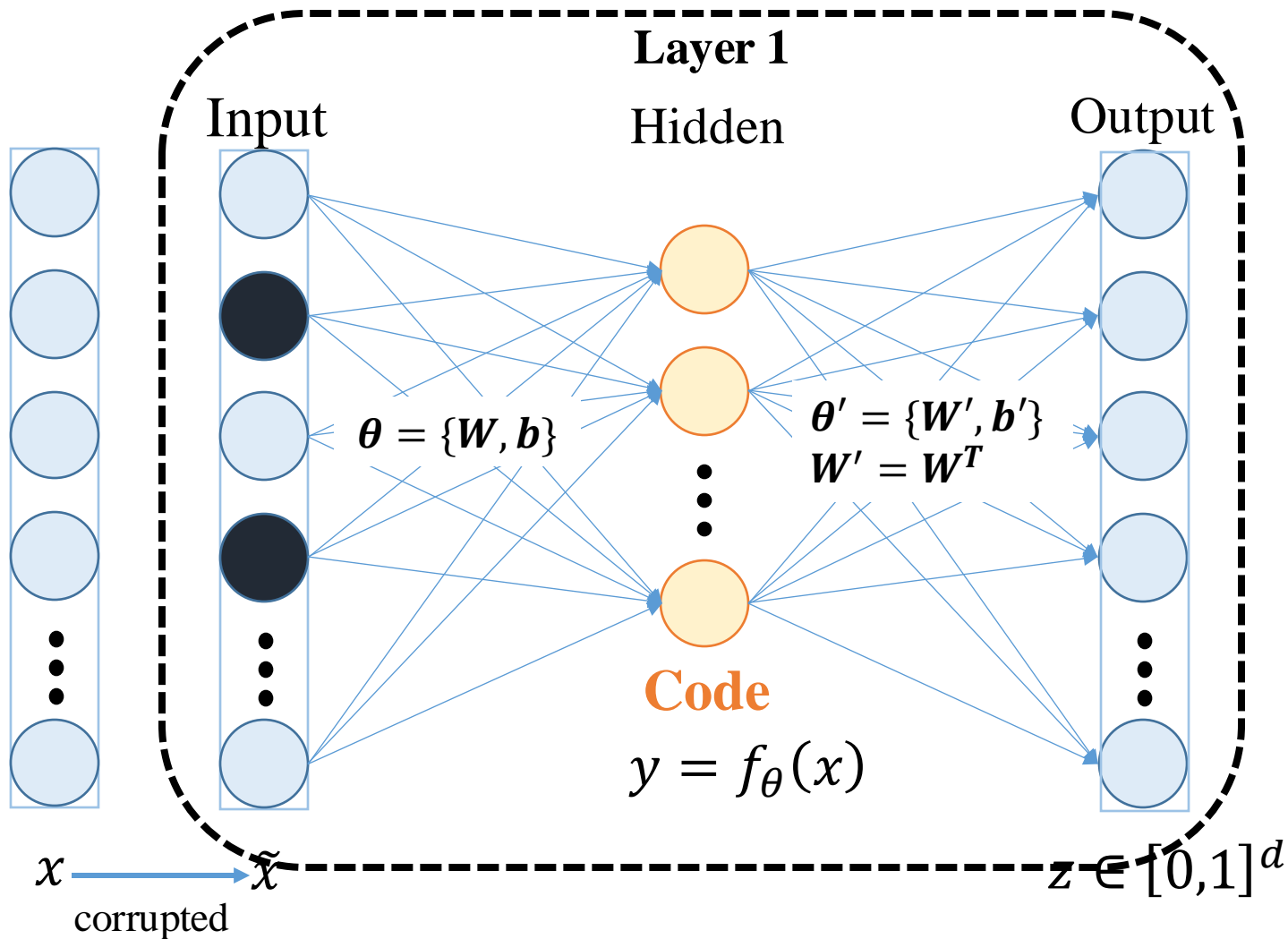
A **good** representation is one that be obtained **robustly** from a **corrupted** input.

## □ Denoising Autoencoders



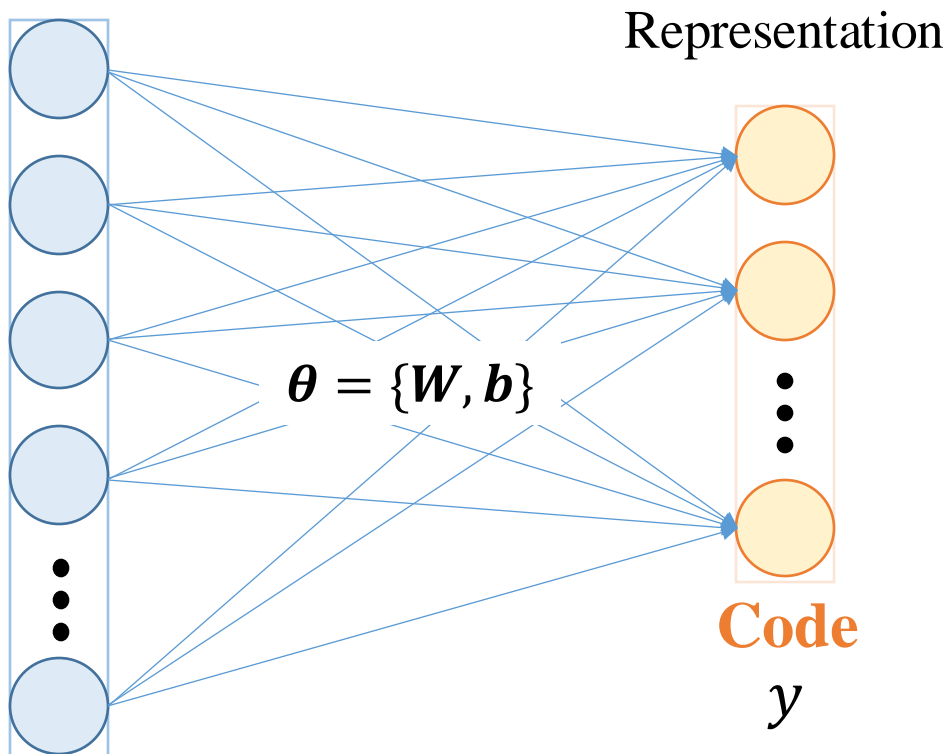
# Representation learning

## □ Denoising Autoencoders



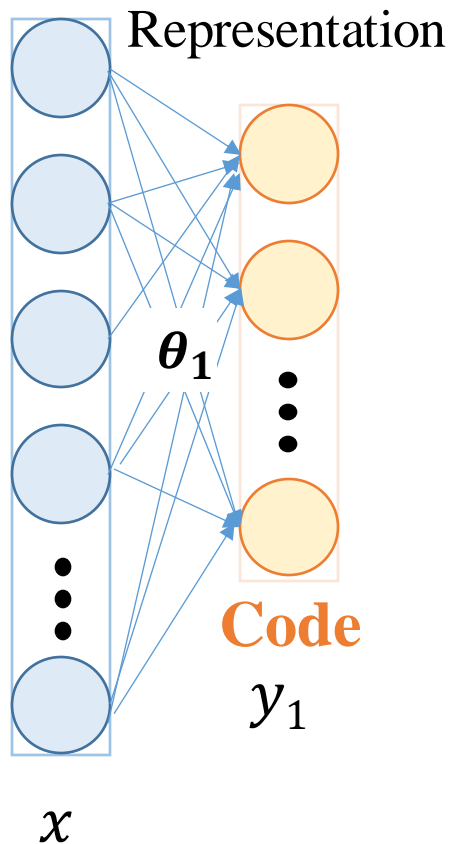
# Representation learning

## □ Denoising Autoencoders



# Representation learning

## □ Denoising Autoencoders

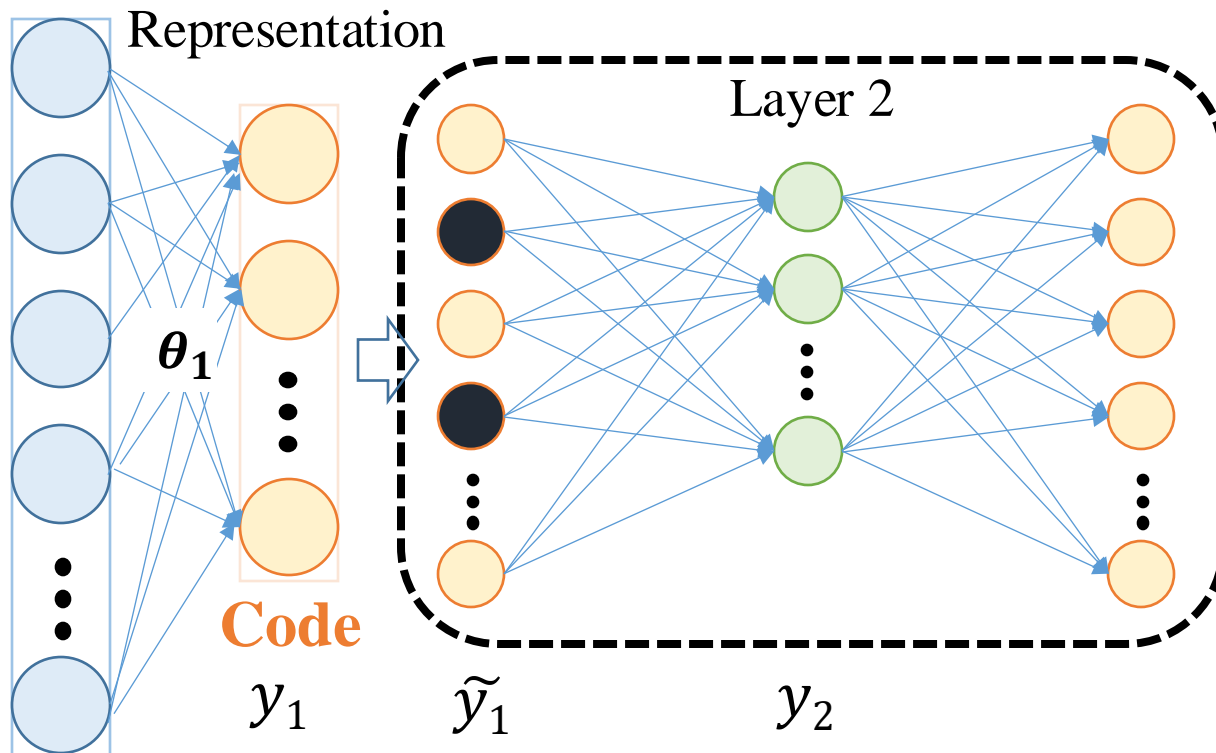


Advantages of deep architectures:

- Re-use of features
- More abstract features

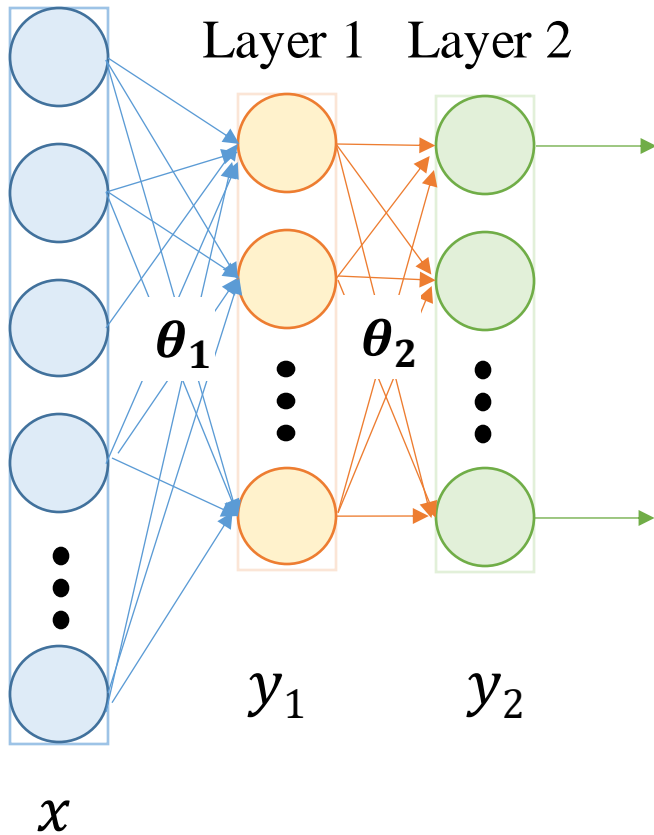
# Representation learning

## Stacked Denoising Autoencoders



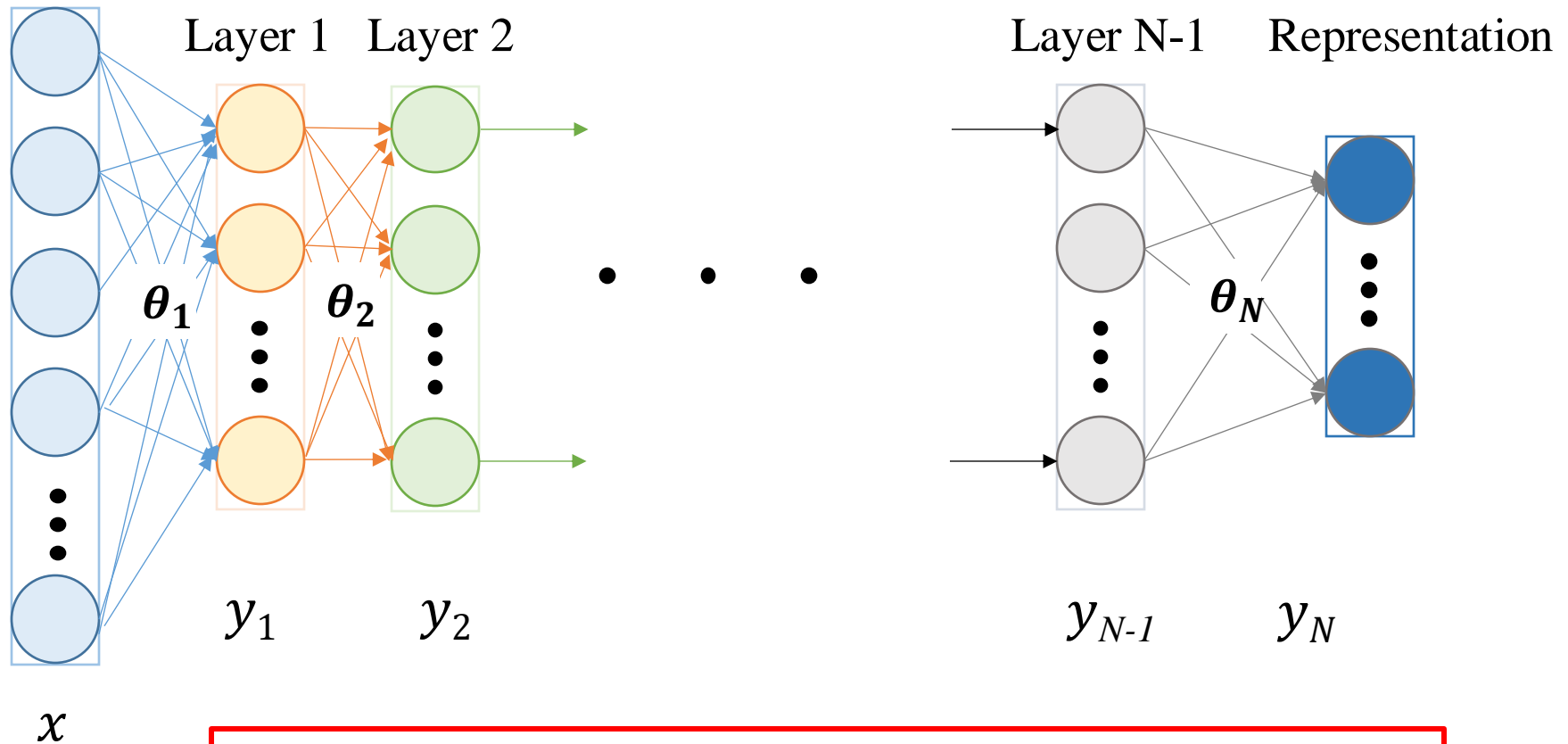
# Representation learning

## Stacked Denoising Autoencoders



# Representation learning

## Stacked Denoising Autoencoders



Each layer of the network is trained to produce a higher- level representation of the observed patterns.