

Speech and Language Processing

Lecture 5 Neural networks

Information and Communications Engineering Course

Takahiro Shinozaki

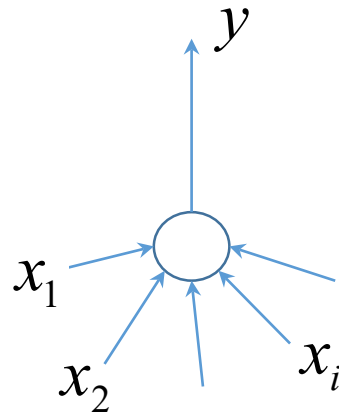
Lecture Plan (Shinozaki's part)

I gives the first 6 lectures about speech recognition. Through these lectures, the backbone of the latest speech recognition techniques is explained.

1. 10/4 (remote)
Introduction and Preparation
2. 10/4 (remote)
Probability Distributions, Markov Models, Samplings
3. 10/6 (remote)
Maximum Likelihood Estimation and EM Algorithm
4. 10/6 (remote)
Bayesian Networks and Bayesian Inference
5. 10/7 (remote)
Neural networks
6. 10/7 (remote)
Reinforcement Learning

Multi Layer Perceptron (MLP)

- Unit of MLP



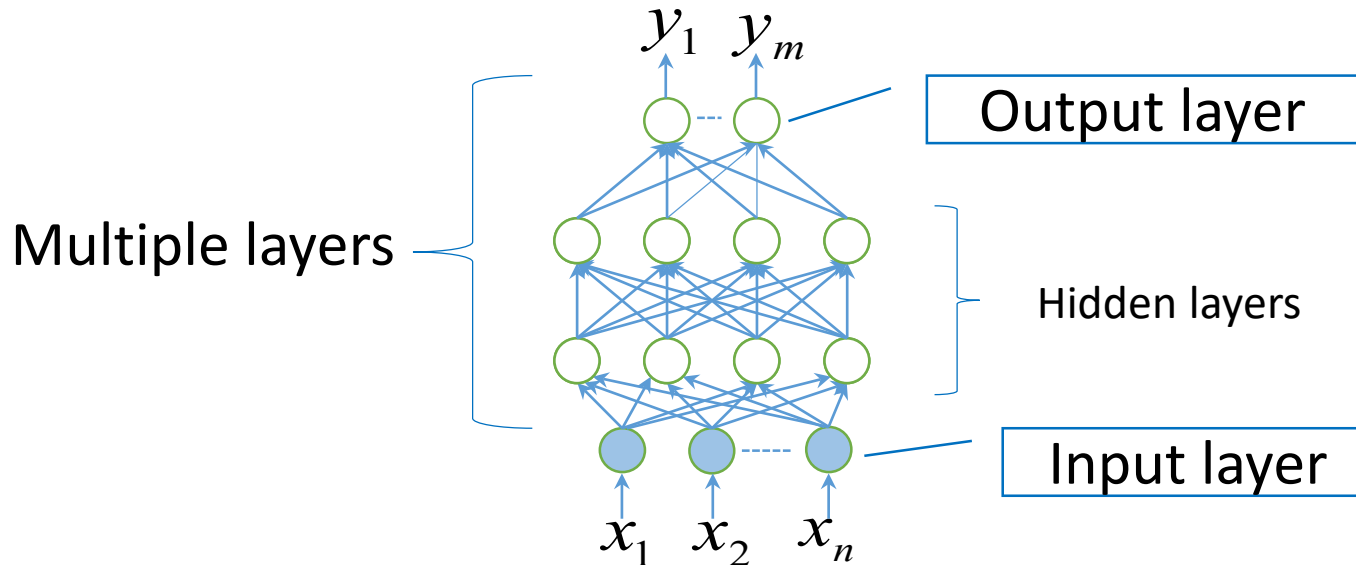
$$y = h\left(\sum_i w_i x_i + b\right)$$

h : activation function

w : weight

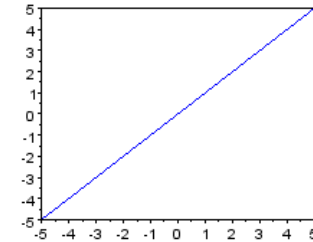
b : bias

- MLP consists of multiple layers of the units

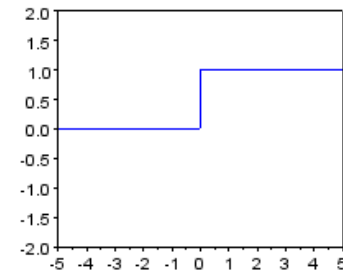


Activation Functions

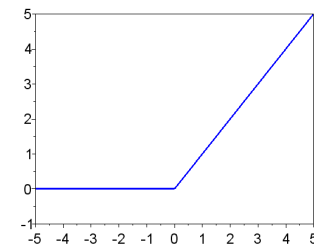
- Linear function $h(x) = x$



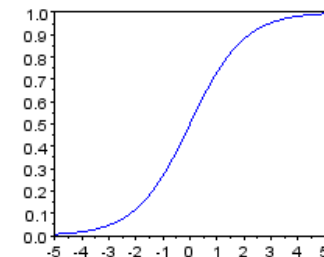
- Unit step function $h(x) = \begin{cases} 1 & \text{if } 0 \leq x \\ 0 & \text{otherwise} \end{cases}$



- hinge function $h(x) = \max\{0, x\}$



- Sigmoid function $h(x) = \frac{1}{1 + \exp(-x)}$



Softmax Function

- For N variables z_i , softmax function is:

$$h(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- Properties of softmax

- Positive $0 < h(z_i)$
- Sum is one $\sum_{i=1}^N h(z_i) = 1.0$



Expresses a probability distribution

- Example

$$Z = \langle z_1, z_2, z_3 \rangle = \langle -1, 2, 1 \rangle \rightarrow h(Z) = \langle h(z_1), h(z_2), h(z_3) \rangle = \langle 0.0351, 0.7054, 0.2595 \rangle$$

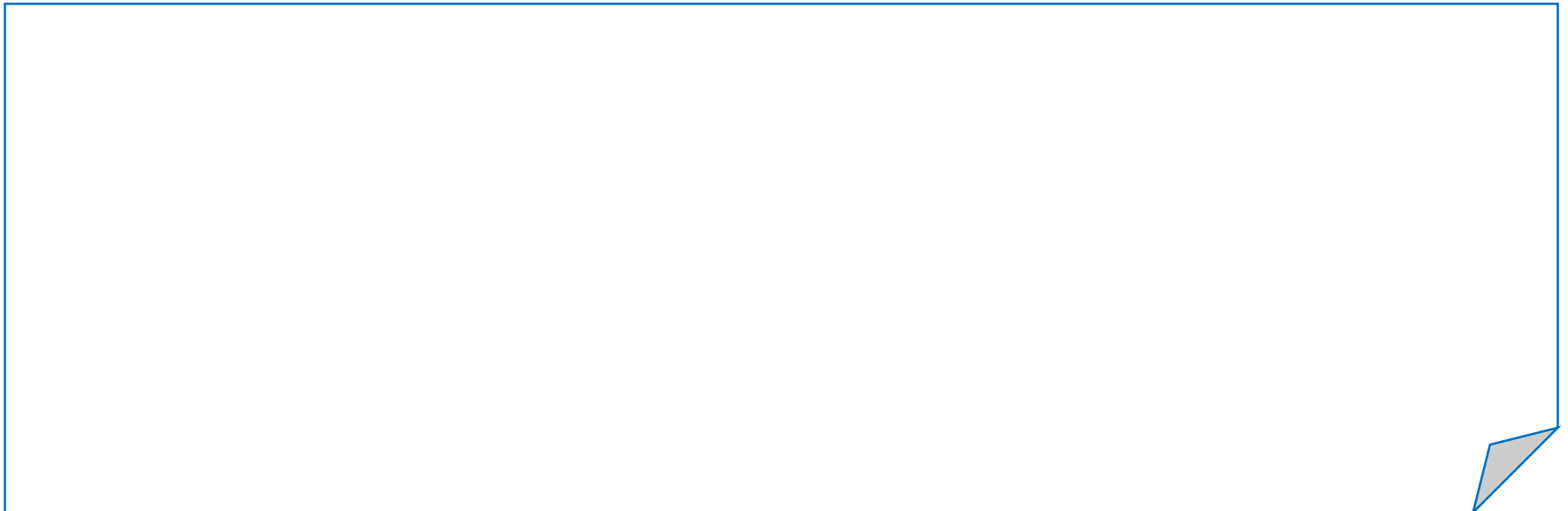
$$Z = \langle z_1, z_2, z_3 \rangle = \langle 16, 8, 12 \rangle \rightarrow h(Z) = \langle h(z_1), h(z_2), h(z_3) \rangle = \langle 0.9817, 0.0003, 0.0180 \rangle$$

Exercise 5.1

- Let h be a softmax function having inputs z_1, z_2, \dots, z_N .

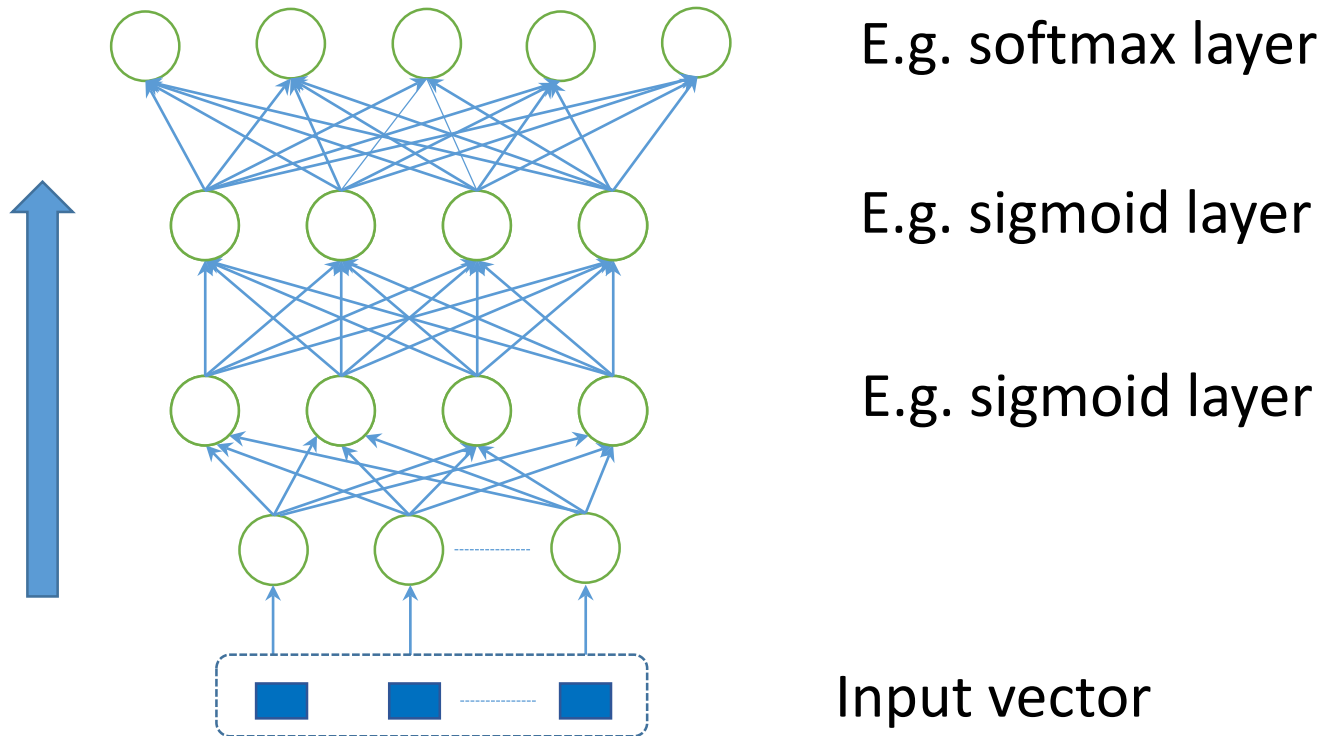
$$h(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- Prove that $\sum_{i=1}^N h(z_i) = 1.0$



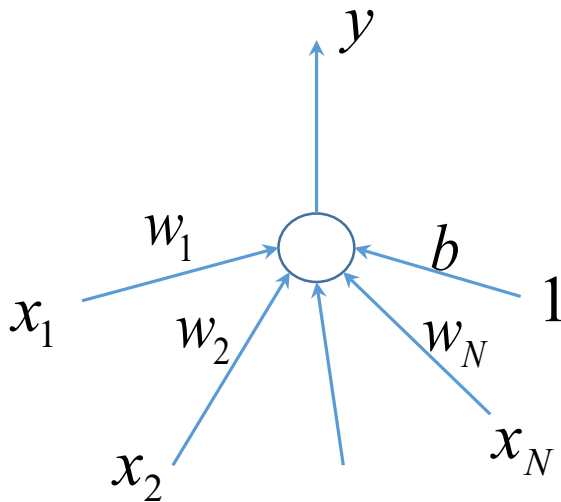
Forward Propagation

- Compute the output of MLP step by step from the input layer to the output layer



Parameters of Neural Network

- The weights and a bias of each unit need training before the network is used



$$y = h\left(\sum_i w_i x_i + b\right)$$
$$= h(\mathbf{w} \cdot \mathbf{x})$$

h : activation function

w : weight vector

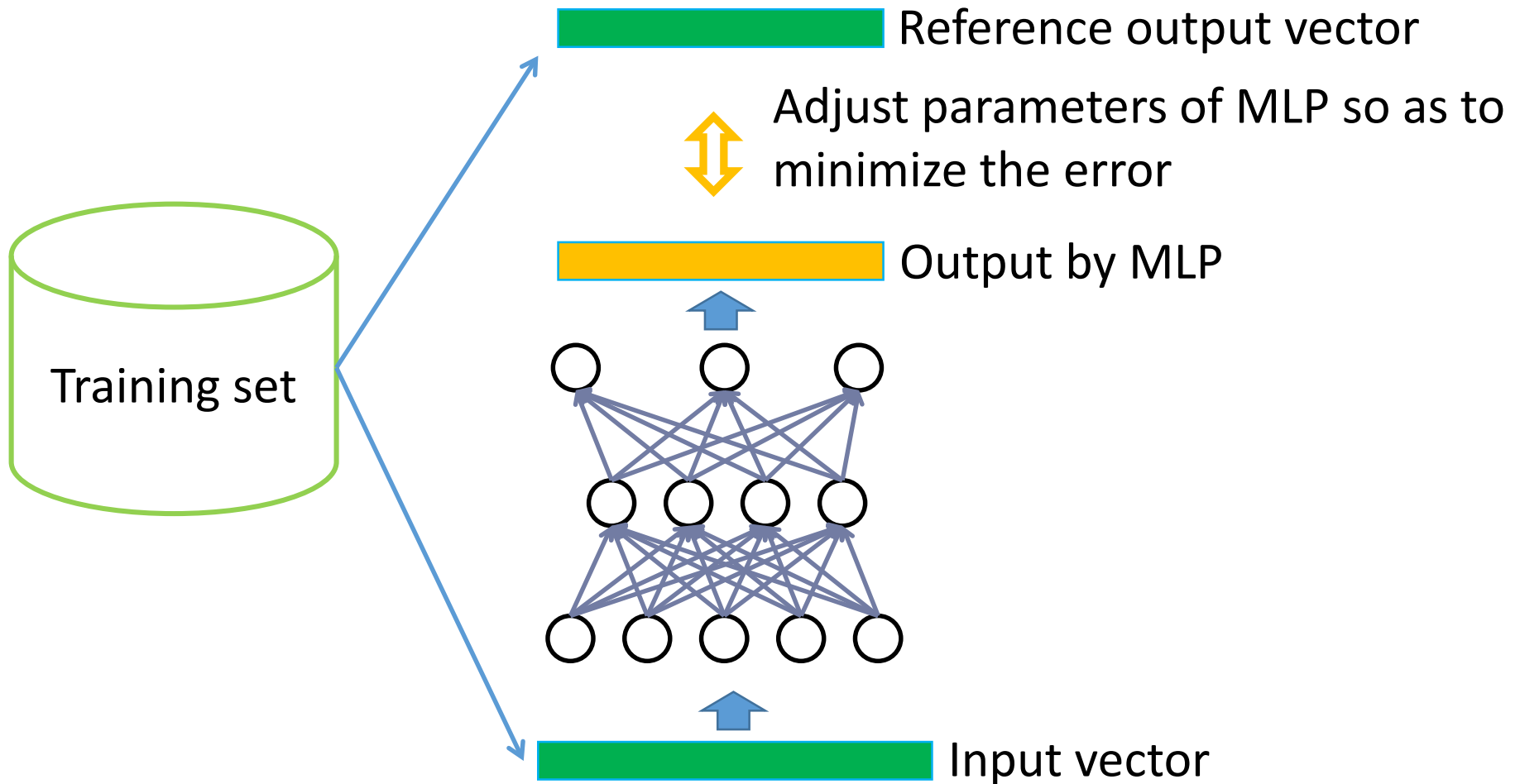
$$w = (w_1, w_2, \dots, w_N, b)$$

X : input vector

$$x = (x_1, x_2, \dots, x_N, 1)$$

The bias b can be regarded as one of the weights whose input takes a constant value 1.0

Principle of NN Training



Definitions of Errors

- Sum of square error
 - Used when output layer uses linear functions

$$E(W) = \frac{1}{2} \sum_n \|y(X_n, W) - t_n\|^2$$

W : Set of weights in MLP
 X_n : Vector of a training sample (input)
 t_n : Vector of a training sample (output)
 n : Index of training samples

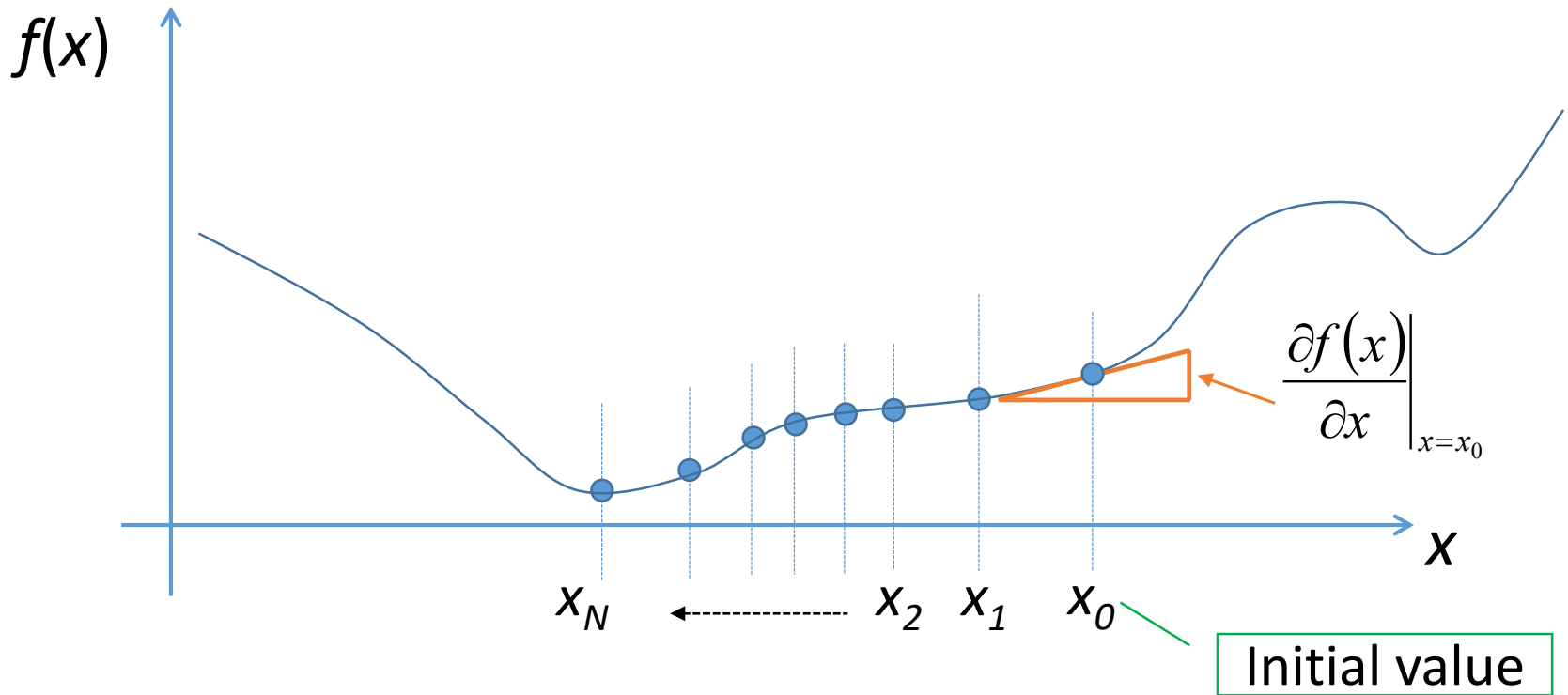
- Cross-entropy
 - Used when the output layer is a softmax

$$E(W) = - \sum_n \sum_k \{t_{nk} \ln y_{nk}(X_n, W)\}$$

t_{nk} : Reference output (Takes 1 if unit k corresponds to correct output, 0 otherwise)
 k : Index of output unit

Gradient Descent

- An iterative optimization method



$$x_{t+1} = x_t - \varepsilon \frac{\partial f(x)}{\partial x_t}$$

ε : Learning rate
(small positive value)

MLP Training by Gradient Descent

- Define an error measure $E(W)$ for training samples

$$\text{e.g. } E(W) = \frac{1}{2} \sum_n \|y(X_n, W) - t_n\|^2$$

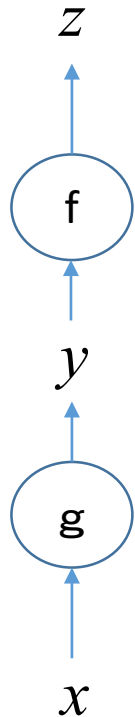
- Initialize parameters $W = \{w_1, w_2, \dots, w_M\}$
- Repeatedly update the parameter set using gradient descent

$$w_i(t+1) = w_i(t) - \varepsilon \left. \frac{\partial E(W)}{\partial w_i} \right|_{w_i = w_i(t)}$$

Chain Rule of Differentiation

$$z = f(y)$$

$$y = g(x)$$



- When x, y, z are scalars:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

- When x, y, z are vectors:

$$x = \langle x_1, x_2, x_3 \rangle, y = \langle y_1, y_2 \rangle, z = \langle z_1, z_2 \rangle$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \quad \text{The same rule holds using Jacobian matrix}$$

$$\begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \end{pmatrix}$$

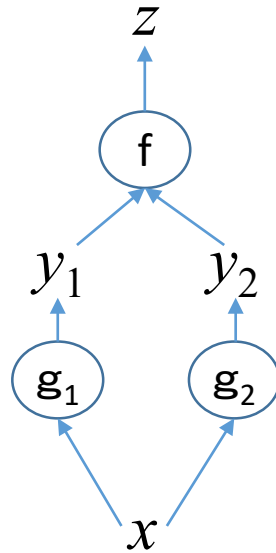
Jacobian matrix

When There Are Branches

$$z = f(y_1, y_2)$$

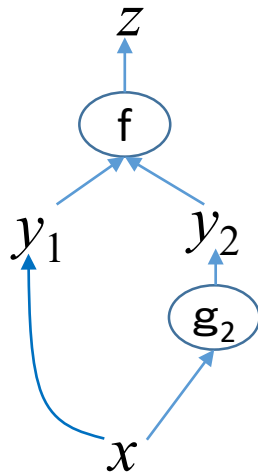
$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$



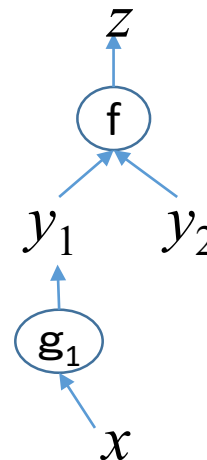
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$

Variations:



$$g_1(x) = x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$

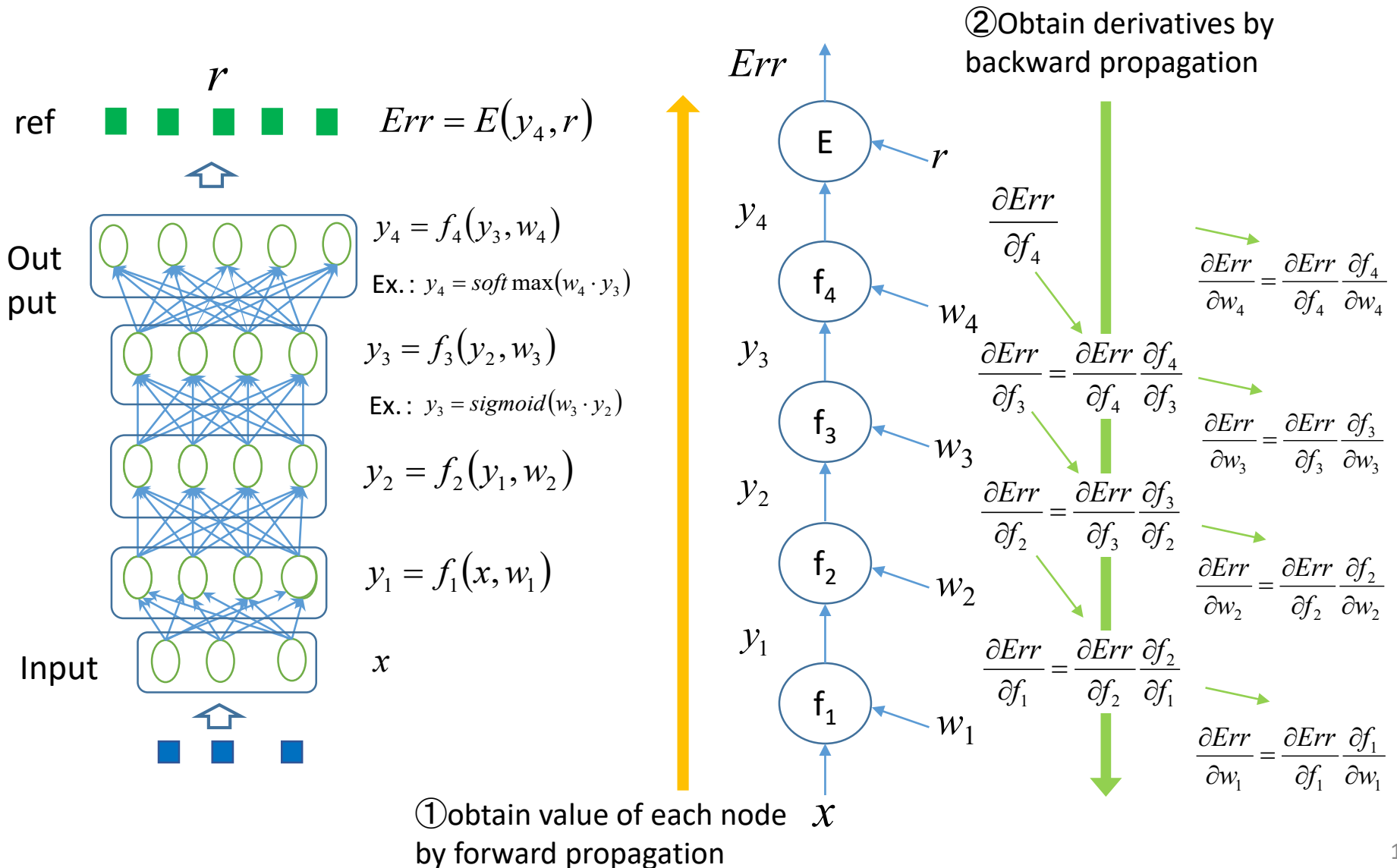


$$g_2(x) = C$$

(independent of x)

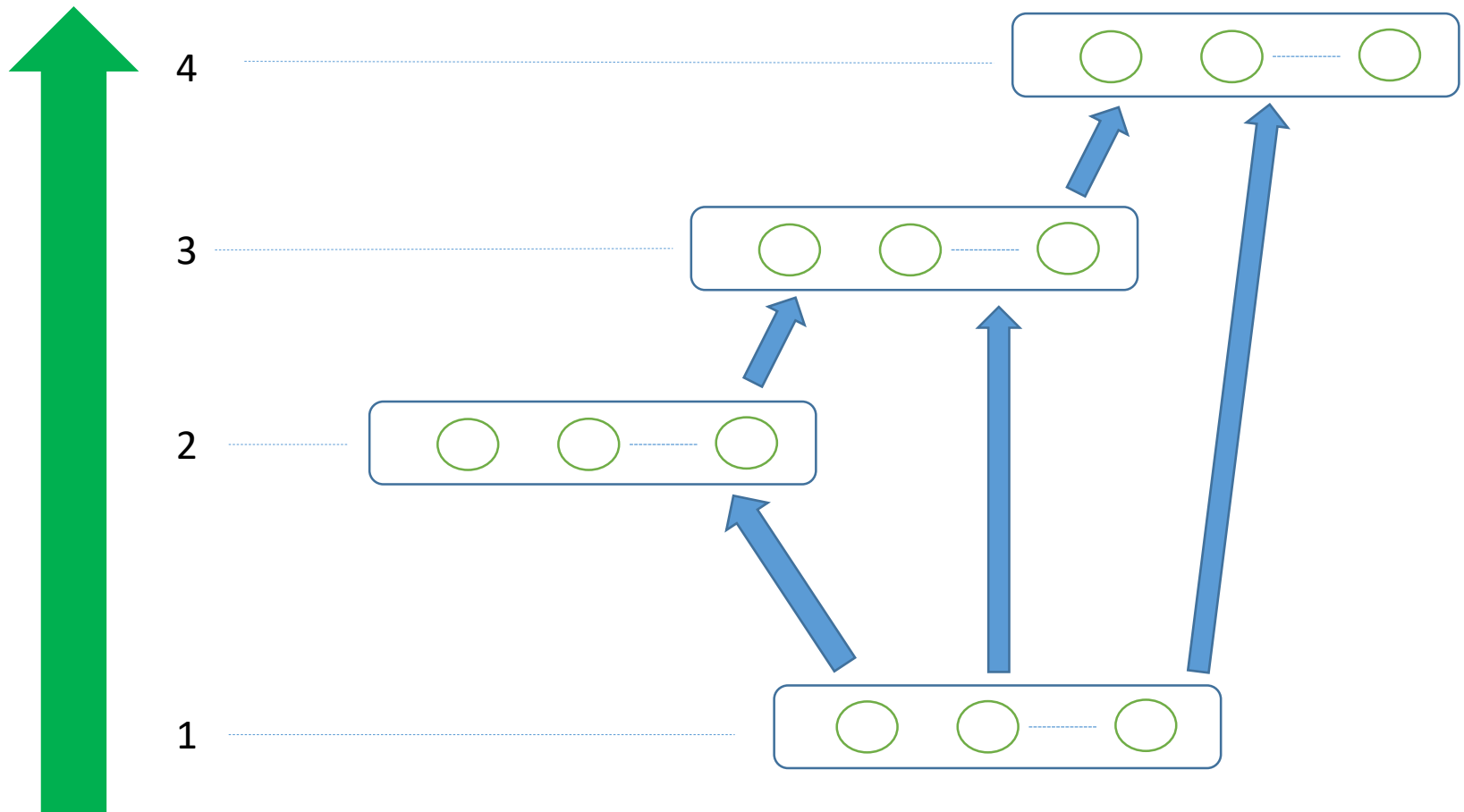
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x}$$

Back Propagation(BP)



Feed-Forward Neural Network

- When the network structure is a DAG, it is called feed-forward network
- The nodes are ordered in a line so that all connections have the same direction
- The forward/backward propagation can be efficiently applied



Exercise 5.2

When $h(y)$ and $y(x)$ are given as follows, obtain $\frac{\partial h}{\partial x}$

$$h(y) = \frac{1}{1 + \exp(-y)}$$

$$y = ax + b$$

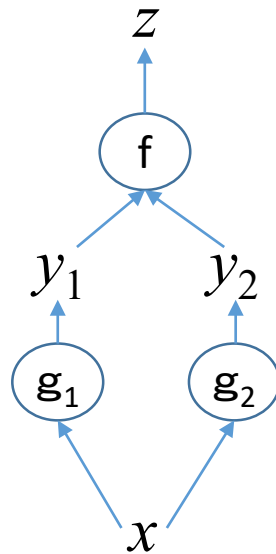
Gradient Evaluation with Branches

These are special cases of the full connection network

$$z = f(y_1, y_2)$$

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

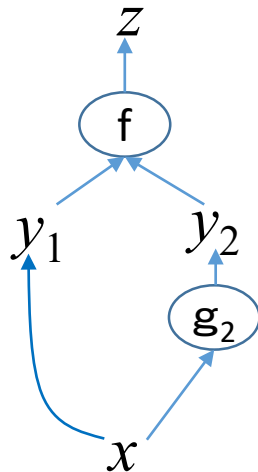


$$\frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial z}{\partial y_1} & \frac{\partial z}{\partial y_2} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \end{bmatrix}$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$

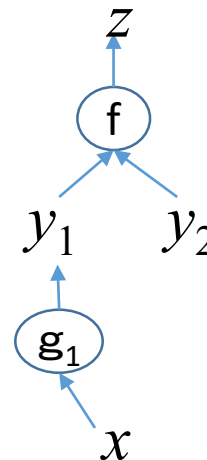
Corresponds to considering block matrices

Variations:



$$y_1 = g_1(x) = x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$

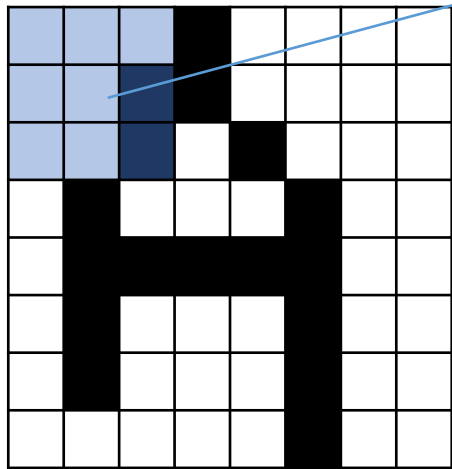


$$y_2 = g_2(x) = C$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x}$$

Convolutional Neural Network (CNN)

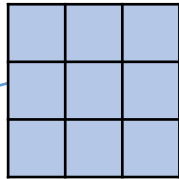
A filter is shifted and applied at different positions



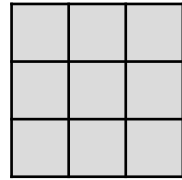
Input

A type of feed-forward neural network with parameter sharing and connection constraint

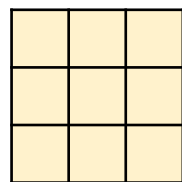
Filter (1)



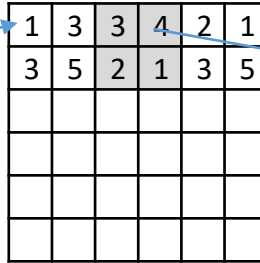
Filter (2)



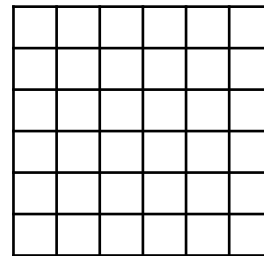
Filter (3)



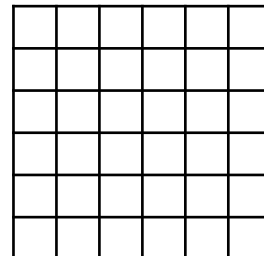
Activation map (1)



Activation map (2)

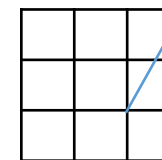
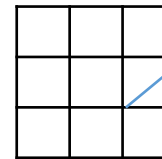
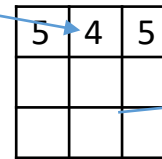


Activation map (N)

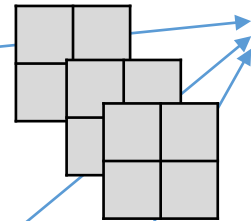


Convolution Layer

Pooling



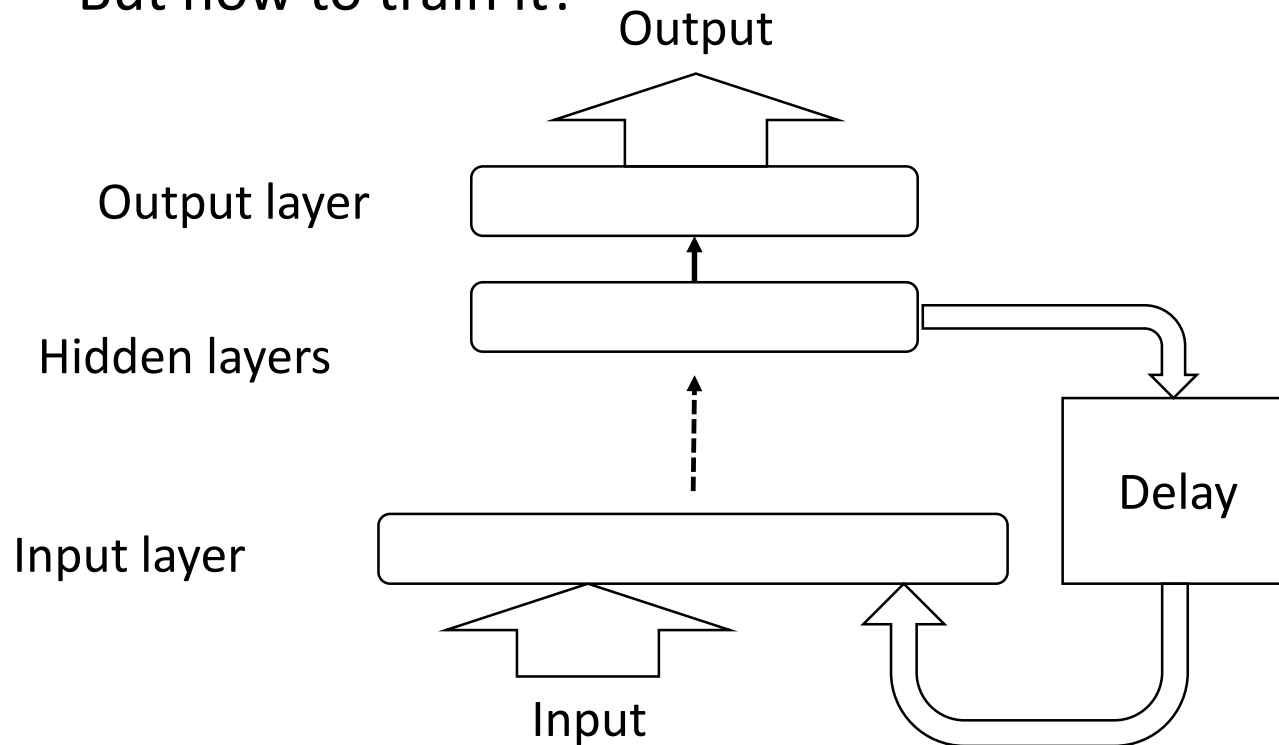
Pooling Layer



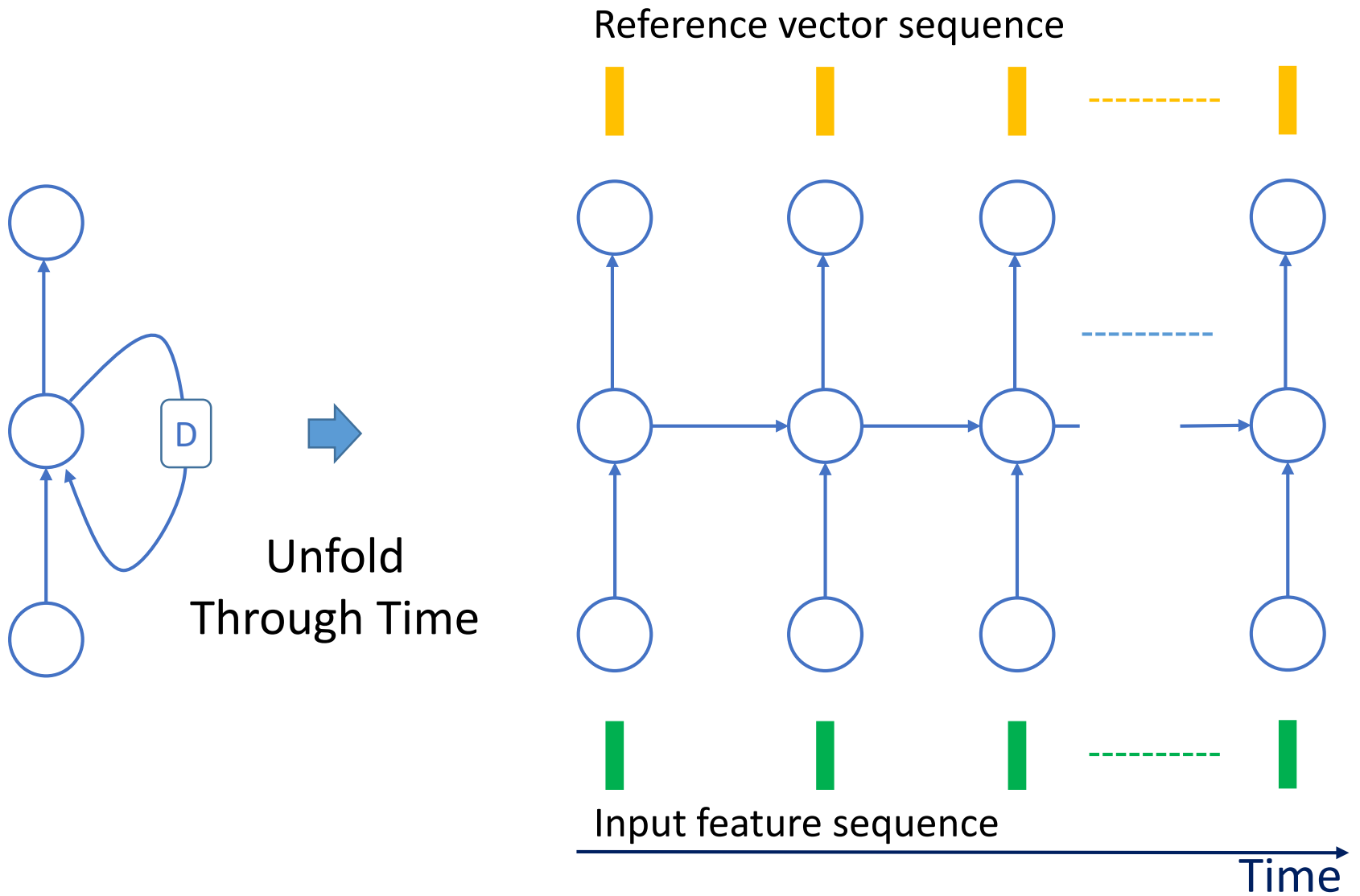
Next convolution layer etc.

Recurrent Neural Network (RNN)

- Neural networks having directed cycles
 - Have more modeling flexibility than feed-forward NN.
 - But how to train it?

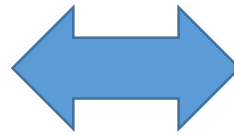
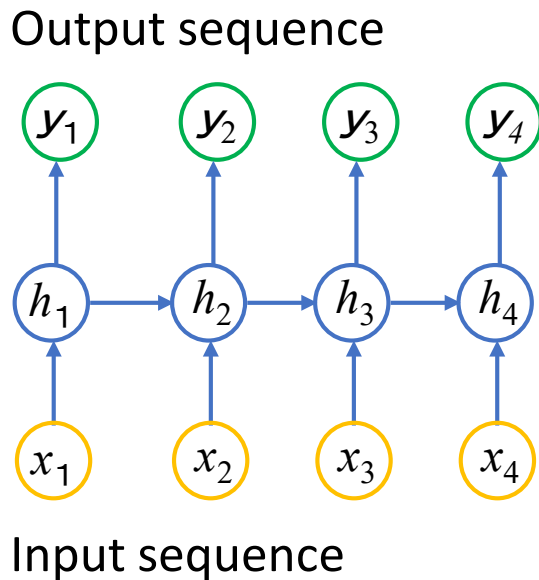


Unfolding of RNN to Time Axis

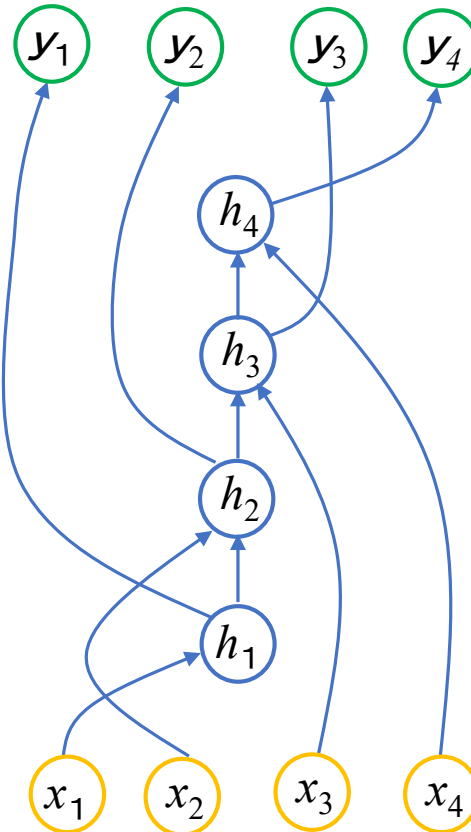


Training of RNN by BP Through Time (BPTT)

Apply BP to the unfolded network



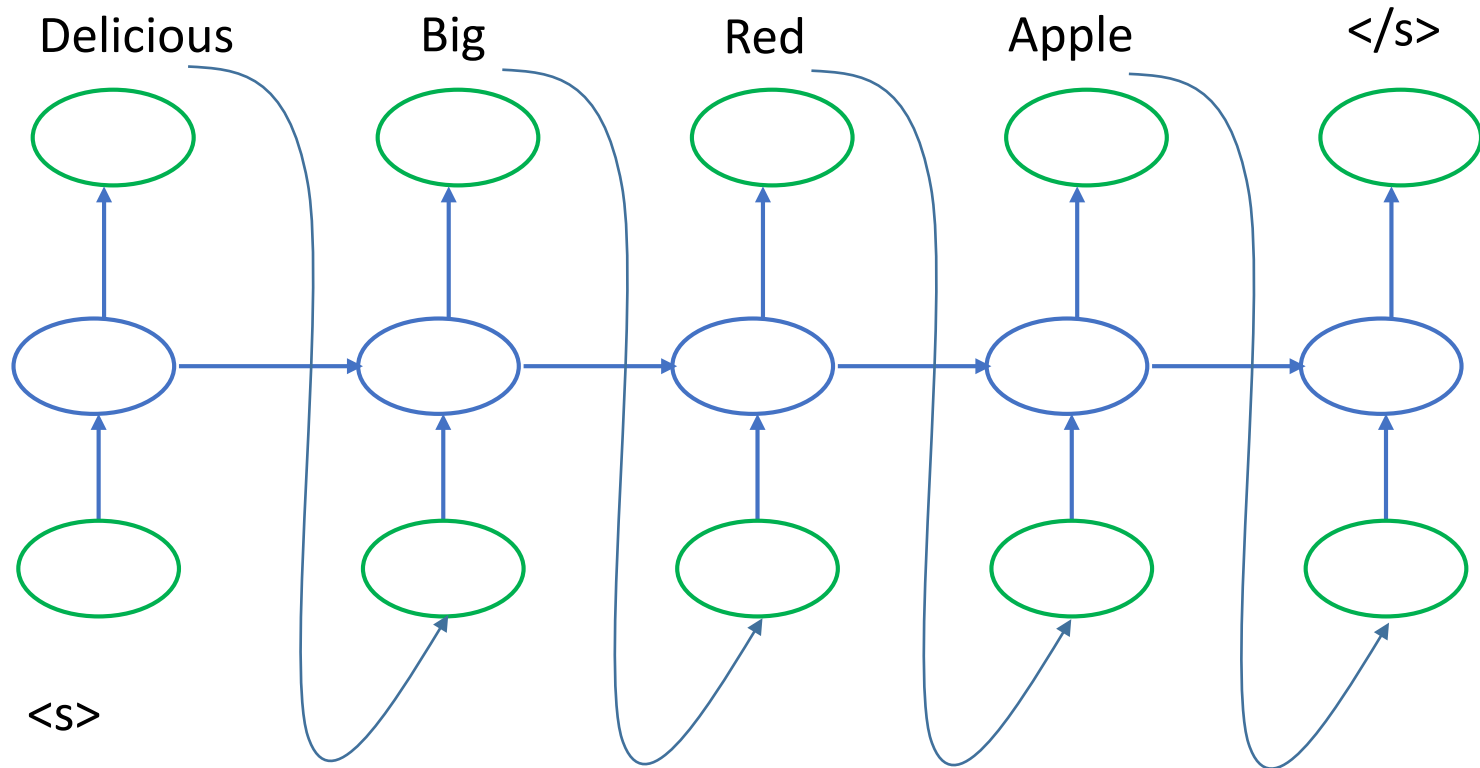
Output
(Regard the output sequence as an output)



Input
(Regard the input sequence as an input)

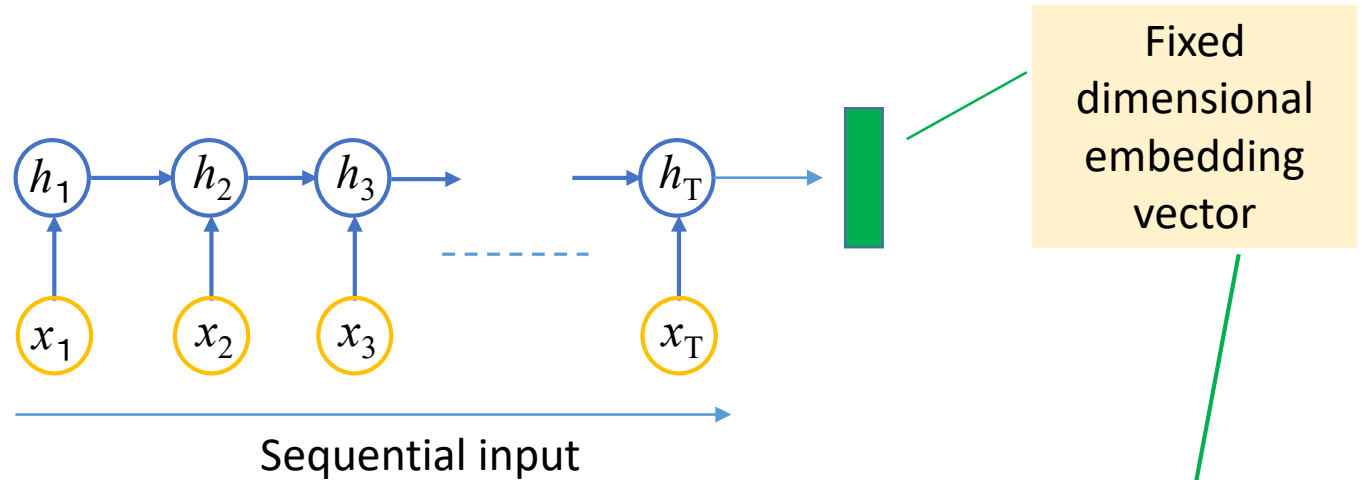
RNN Language Model

$P(\langle s \rangle, \text{Delicious}, \text{Big}, \text{Red}, \text{Apple}, \langle /s \rangle)$

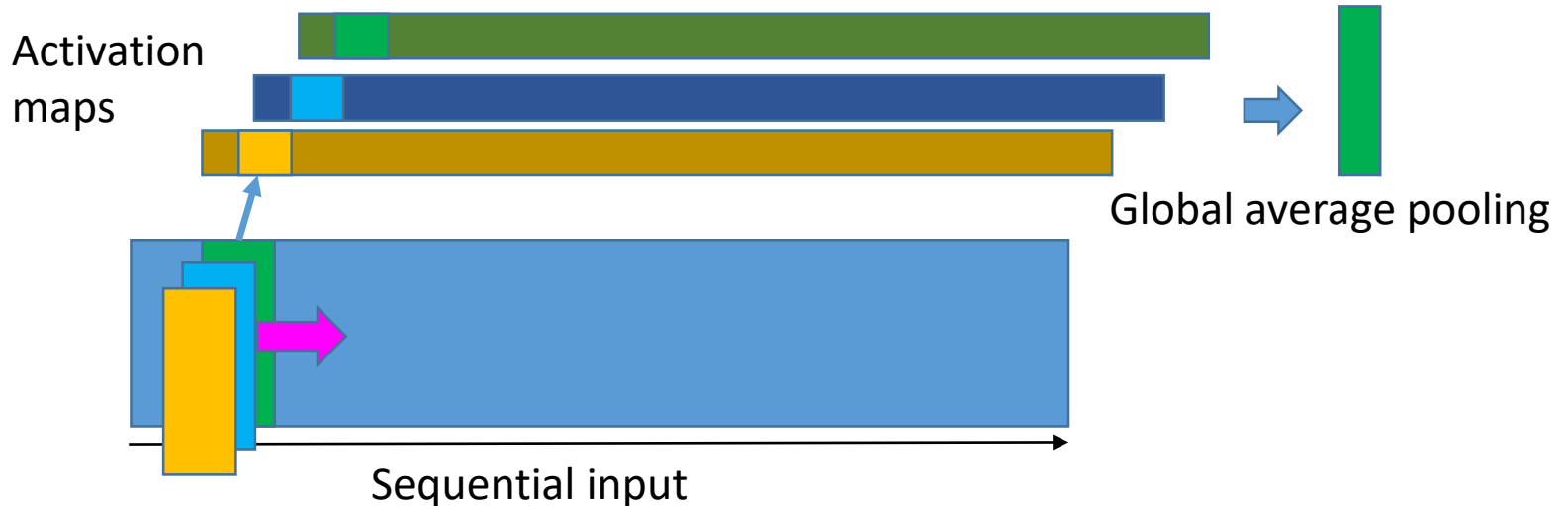


Fixed-Dimensional Embeddings of Sequences

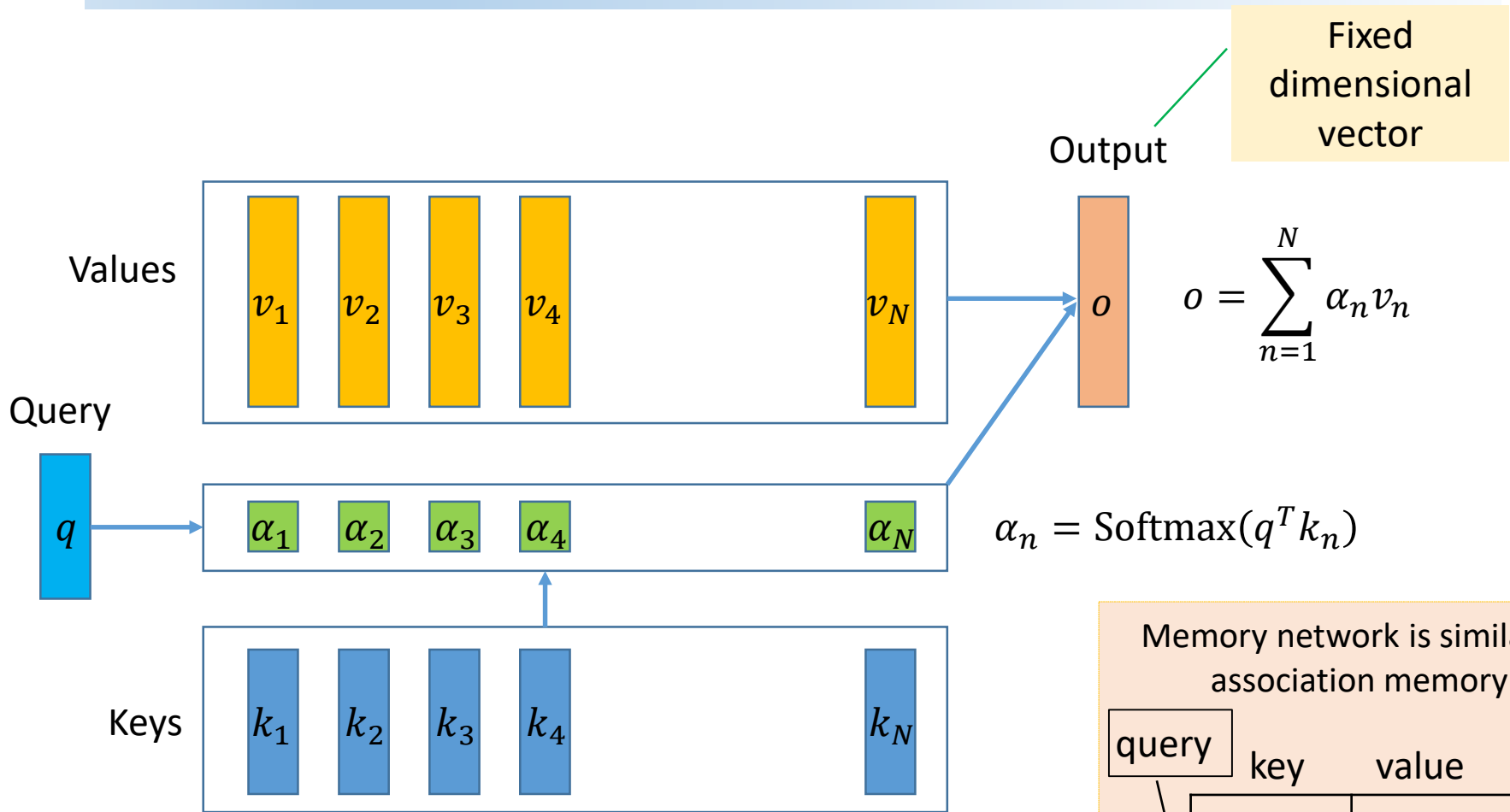
- Use RNN



- 1D CNN with Global average pooling



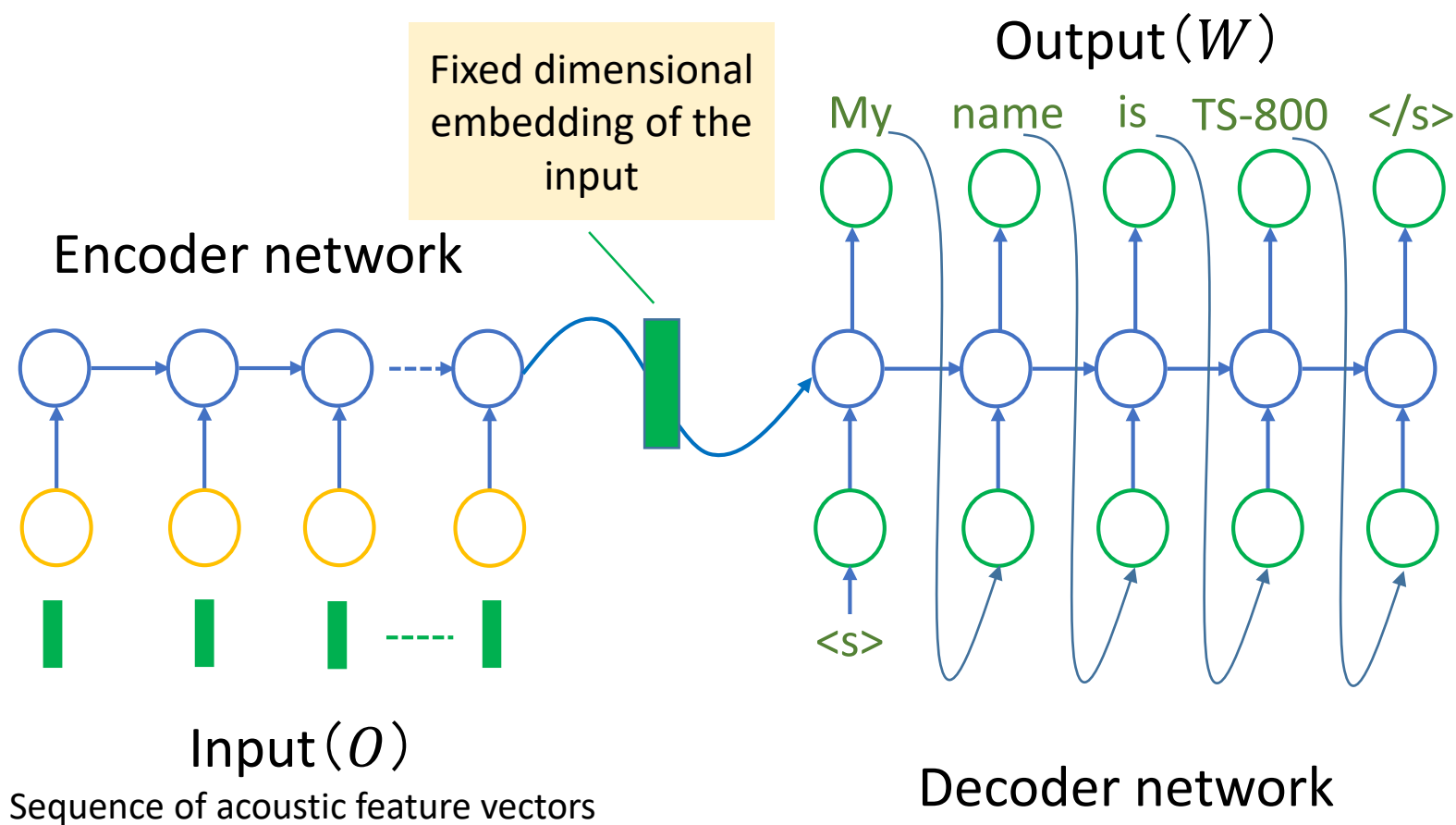
Memory Network



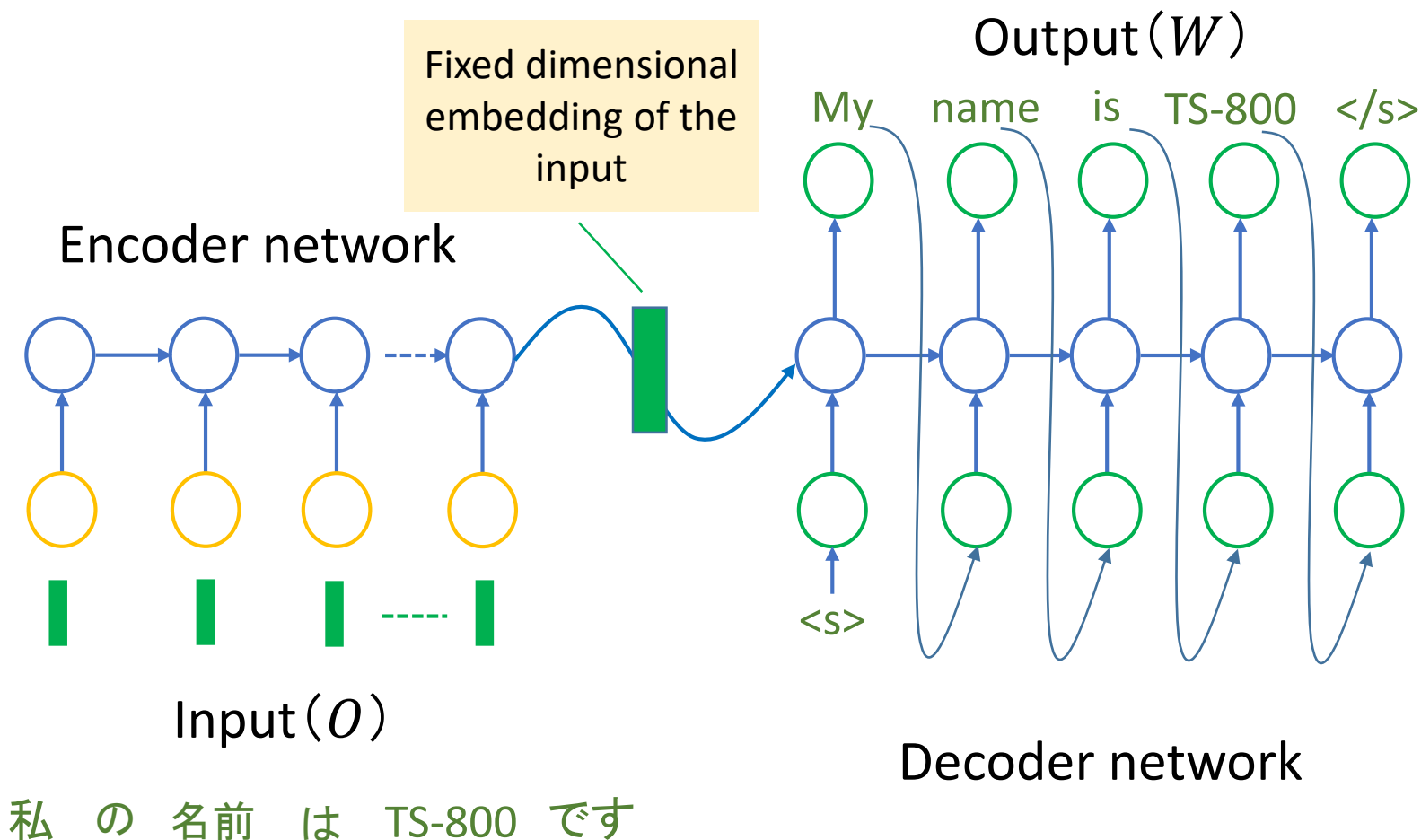
Memory network is similar to association memory

query	key	value
↓	Apple	120
	Banana	100
	Durian	5500

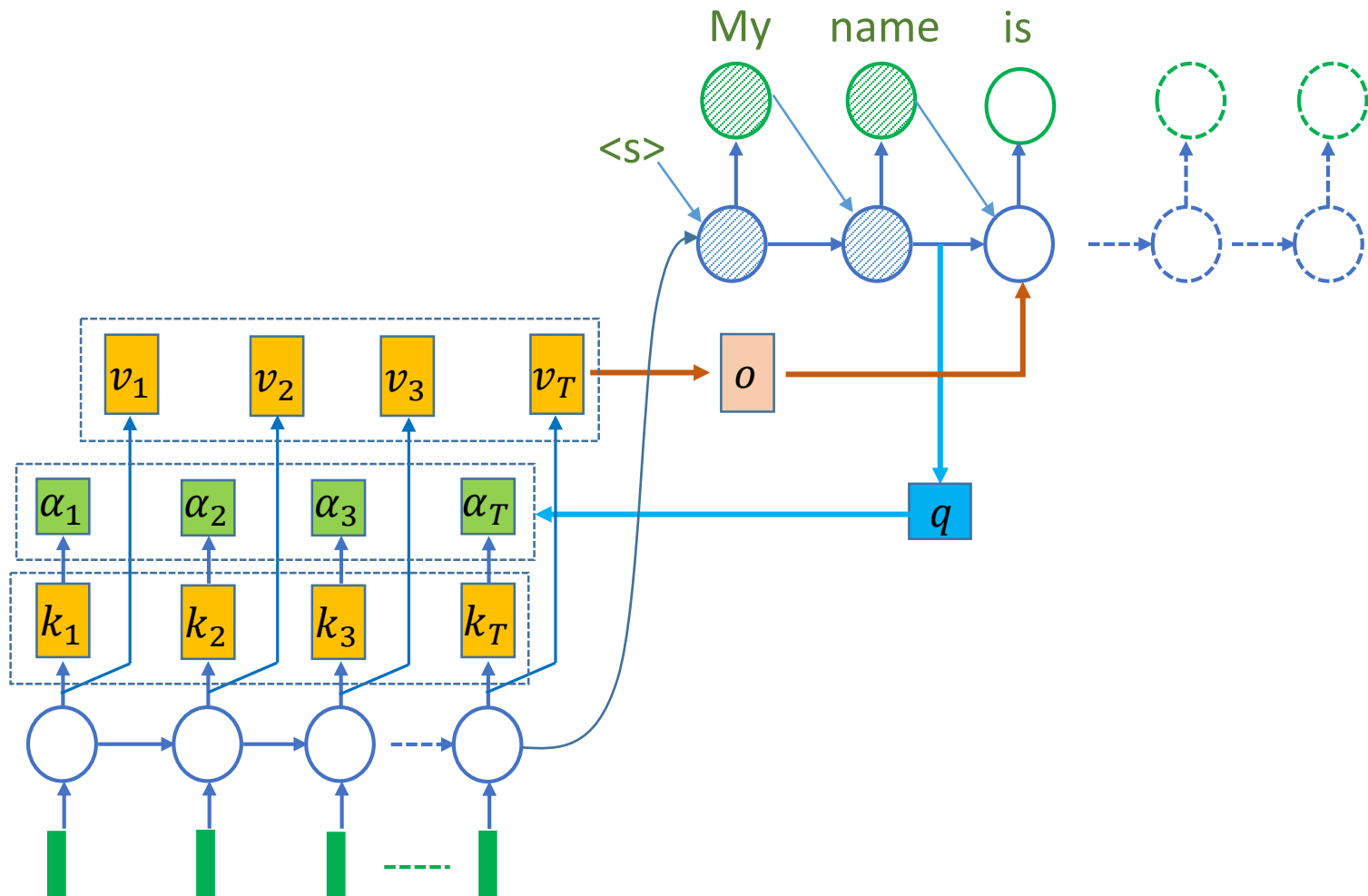
Encoder-Decoder Speech Recognition System



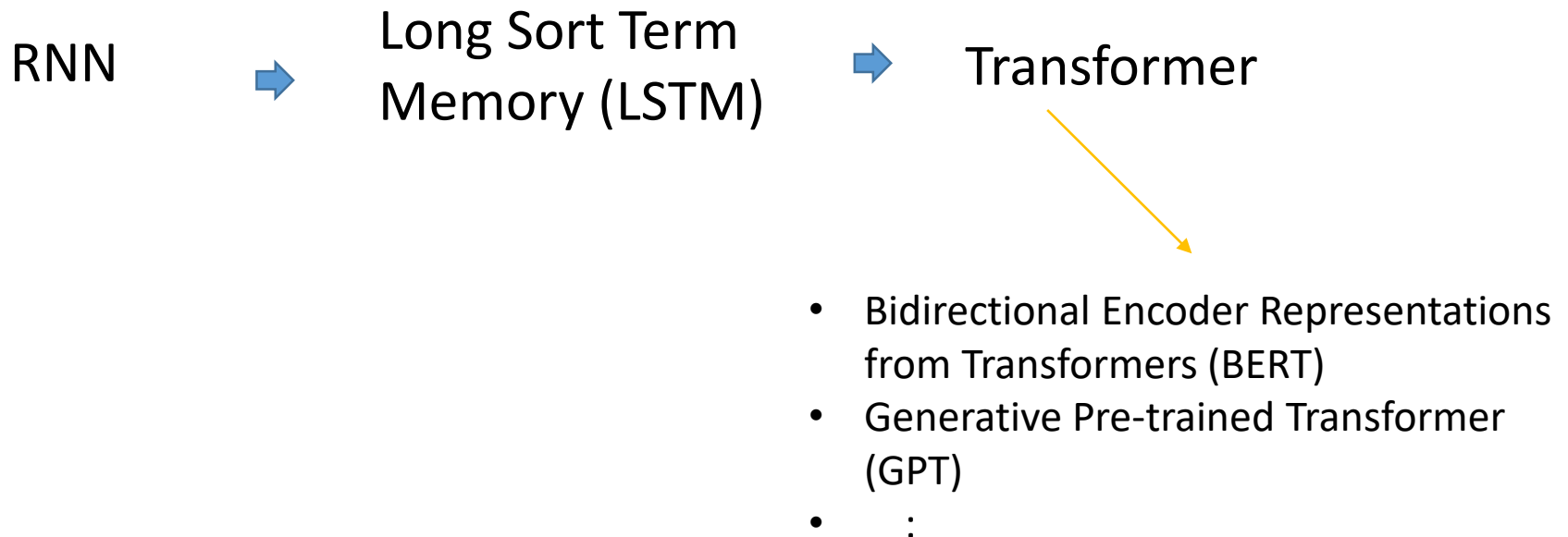
Encoder-Decoder Translation System



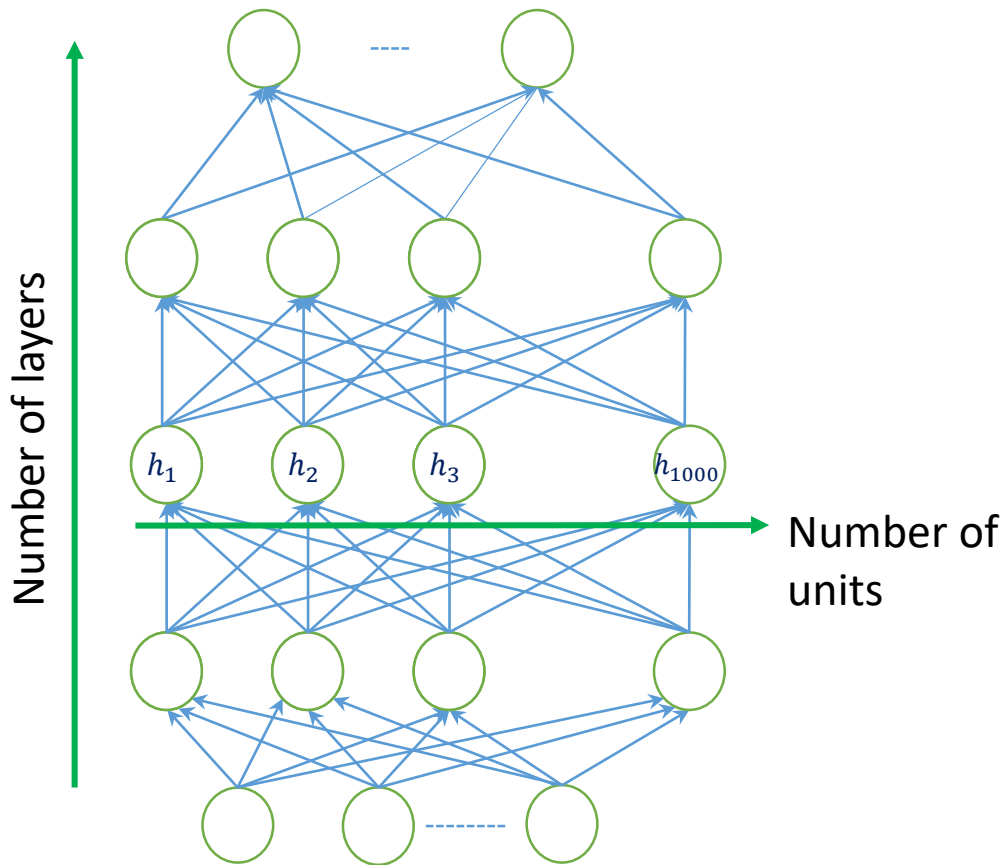
Attention Mechanism



Evolution of Sequence Modeling



Computational Cost of NN



- If the number of layers is S , the number of units per layers is T , the number of connections of units is order of ST^2 . To calculate the outputs from the K input samples, the computing order can be KST^2 .
 - E.g. when $S = 10, T = 4000, K = 100000000$
 $KST^2 = 1.6 * 10^{16}$
- In model training, many iterations of the training data are repeated



Extensive calculations are required

Performance of Compute Hardware



<https://youtu.be/KQ9spCW8ddk>



Relay Calculator

~1 FLOPS (?)

300 million years to perform 10^{16} operations

Released in 2020
GPU RTX3090

~35.7 TFLOPS (FP32),
350W

(Used in Personal
computers ~ 300,000 yen)



<https://www.nvidia.com>



https://www.flickr.com/photos/manatee_tw/3585719603/

Earth Simulator (60 billion yen)

World's top performance in 2002

~40T FLOPS ($T=10^{12}$)

250 seconds to perform 10^{16} operations

280 seconds to perform 10^{16} operations

GPU is originally made for
video games

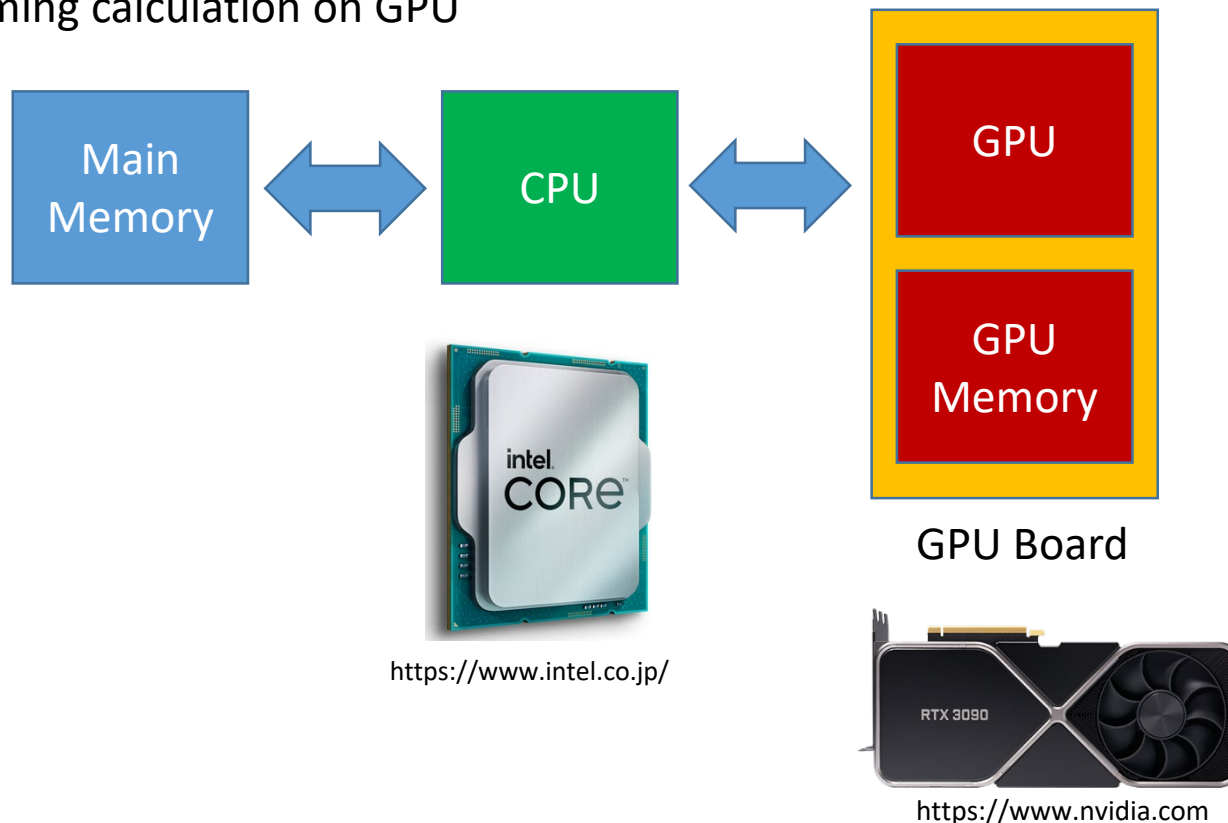
Example :

<https://www.youtube.com/watch?v=SoTVrFfSi8s>



Typical Hard Structure of Computers

- Typical hardware structures of computers that are used for NN training / evaluation
- Both CPU and GPU can be used for neural net computation, but GPU is faster
- Data (NN parameters, input/reference samples) must be in GPU memory when performing calculation on GPU



Exercise 5.3

- When you have a computer that can perform 10^9 floating point calculations per second (1G FLOPS), how many seconds does it take to perform 10^{16} floating point calculations?