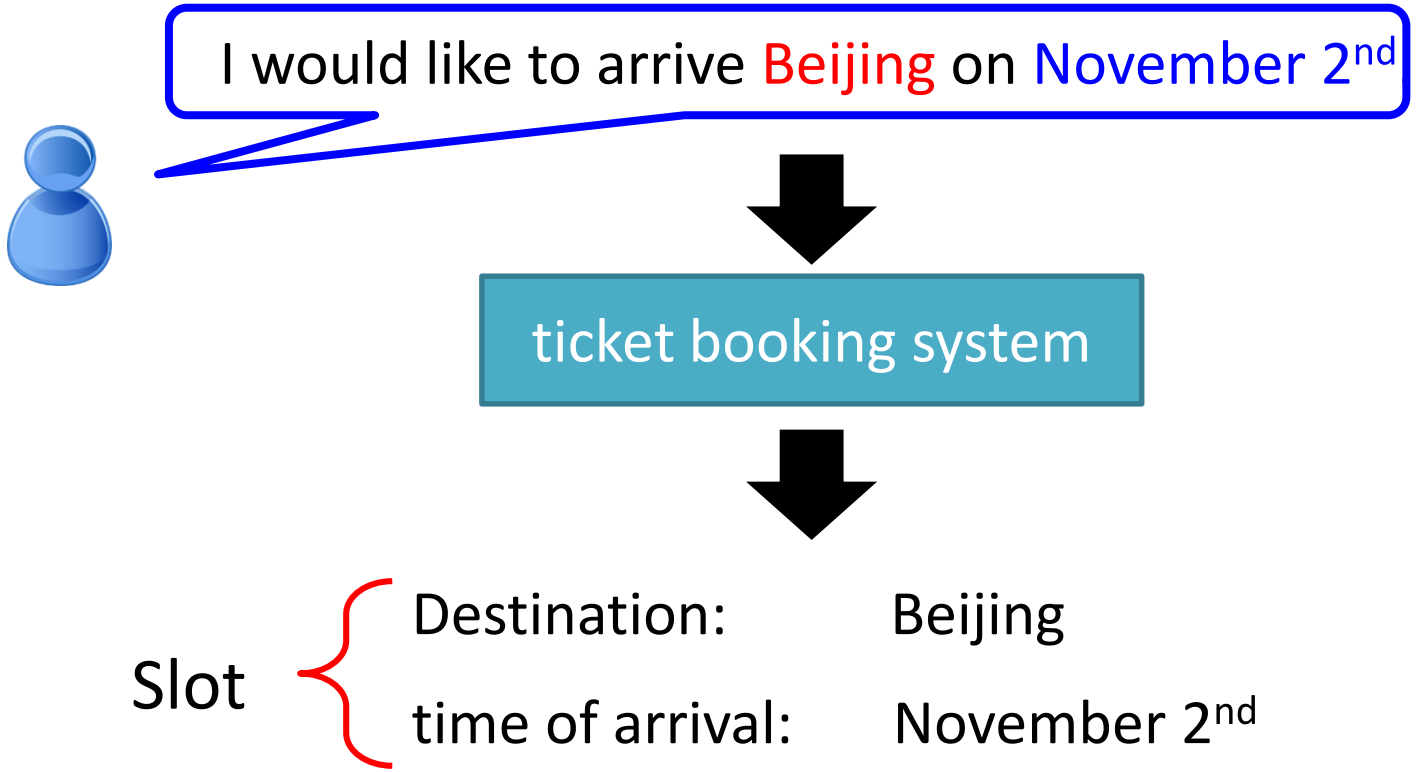
Recurrent Neural Network & Attention

本节课所使用的课件大部分源自台湾大学李宏毅老师的机器学习课件，

感谢李老师团队在课程建设方面所做的工作!

Example Application

- Slot Filling

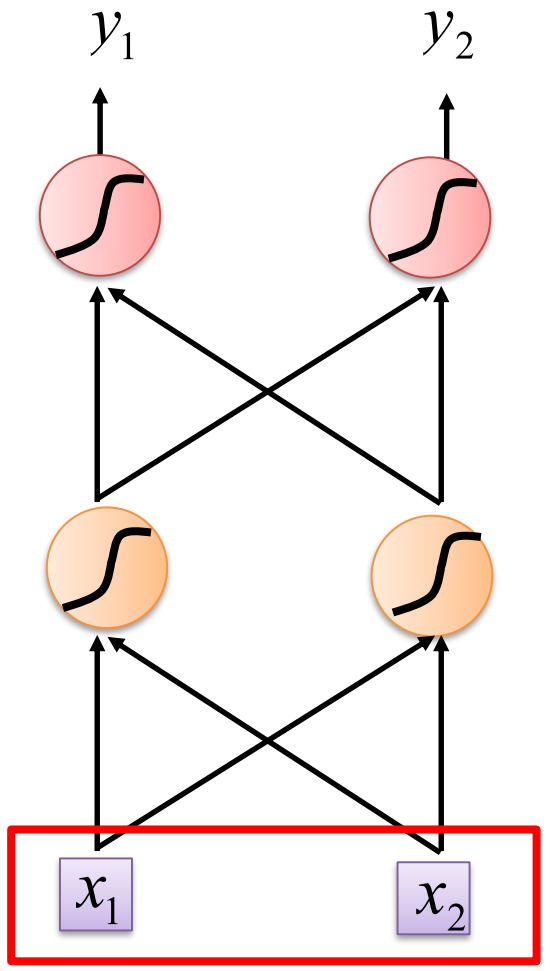


Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)



1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [1 0 0 0 0]
bag = [0 1 0 0 0]
cat = [0 0 1 0 0]
dog = [0 0 0 1 0]
elephant = [0 0 0 0 1]

Example Application

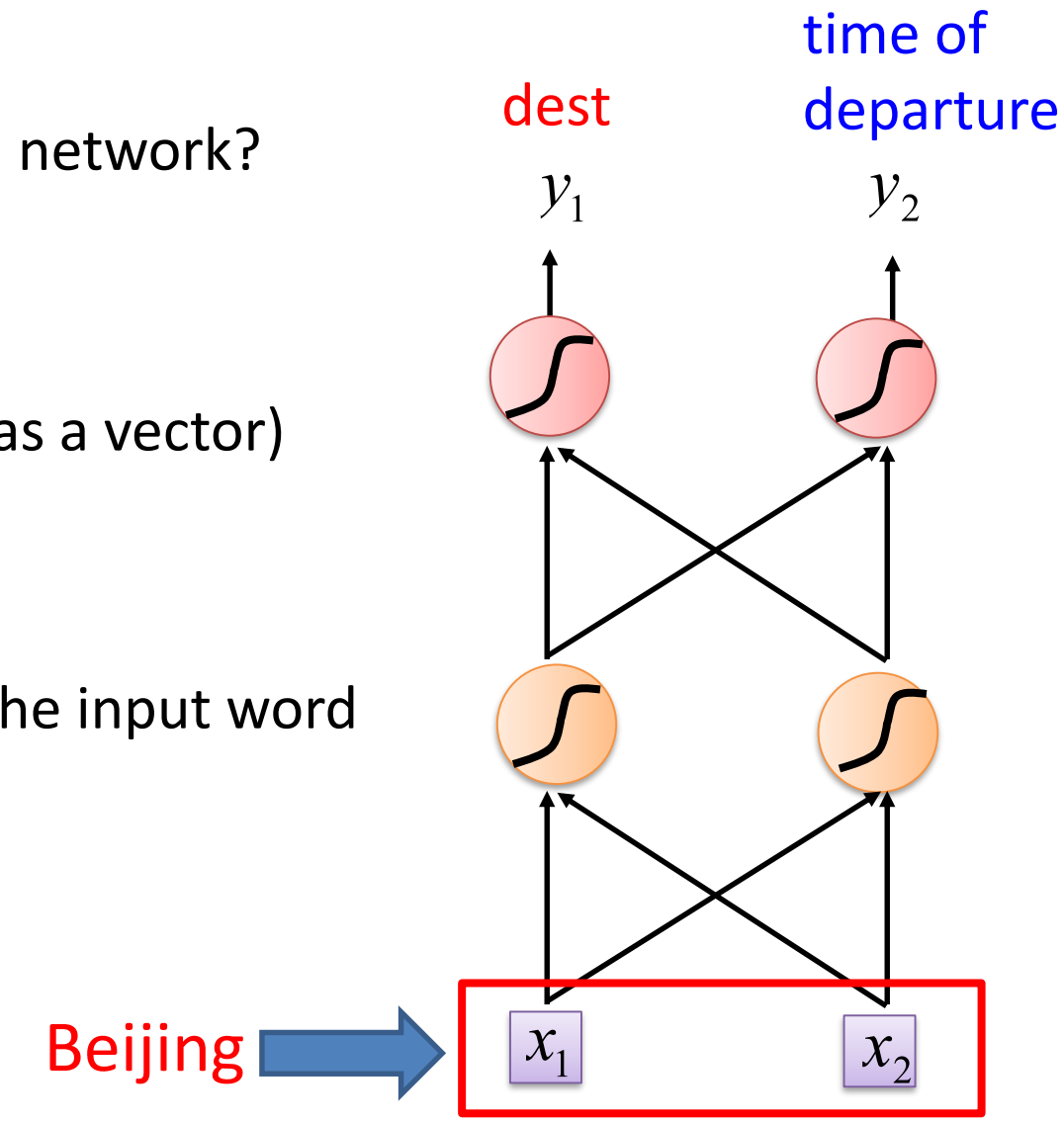
Solving slot filling by Feedforward network?

Input: a word

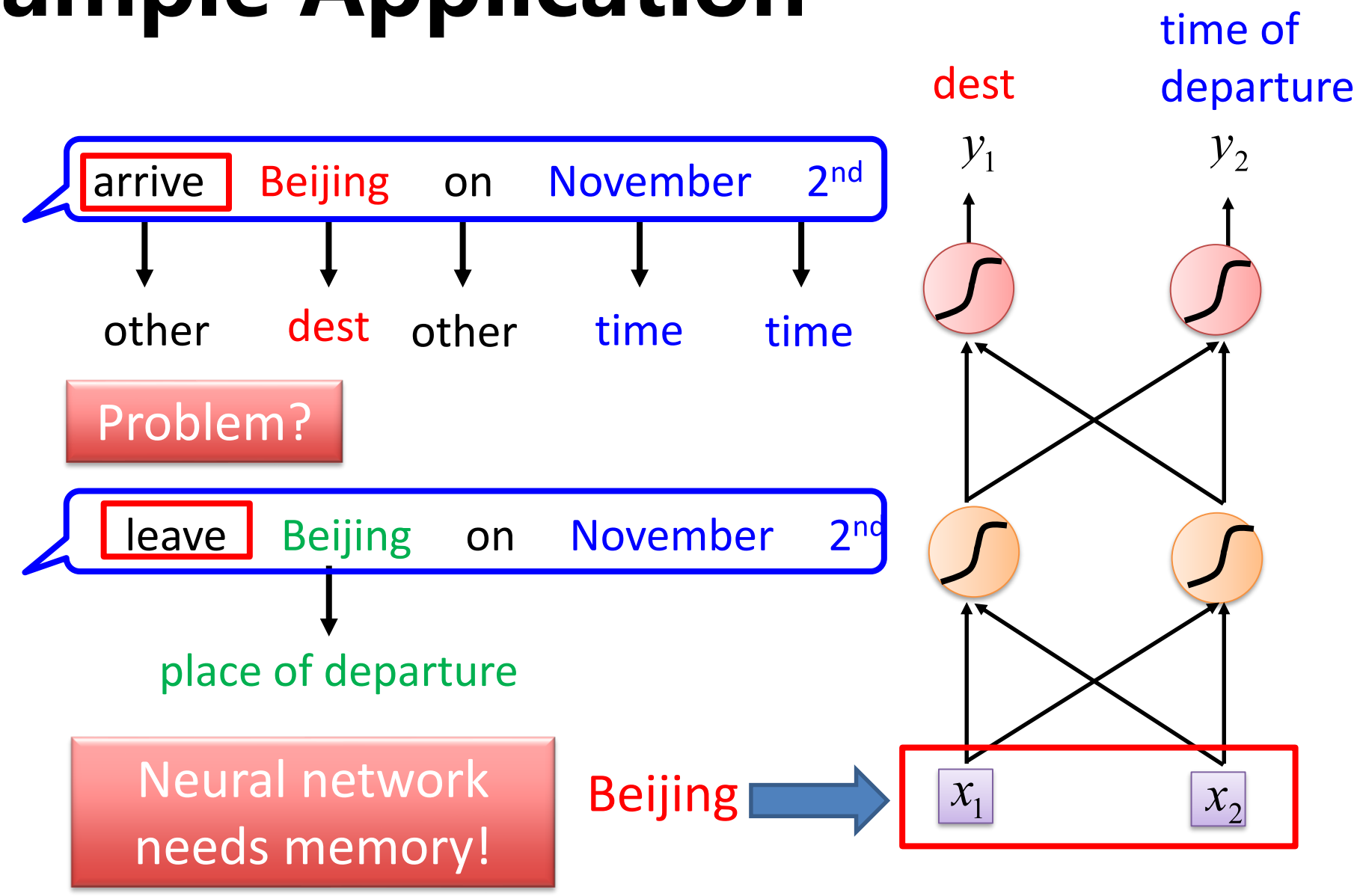
(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots

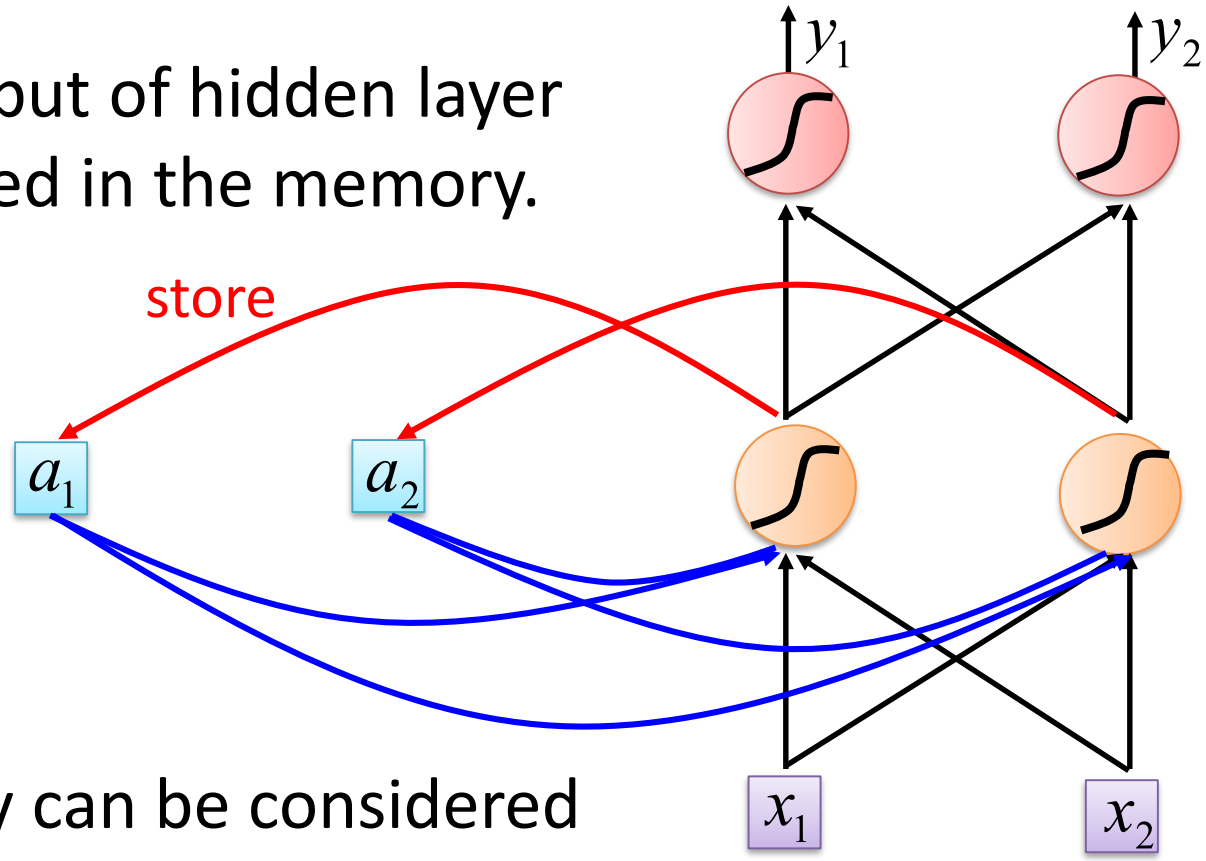


Example Application



Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.

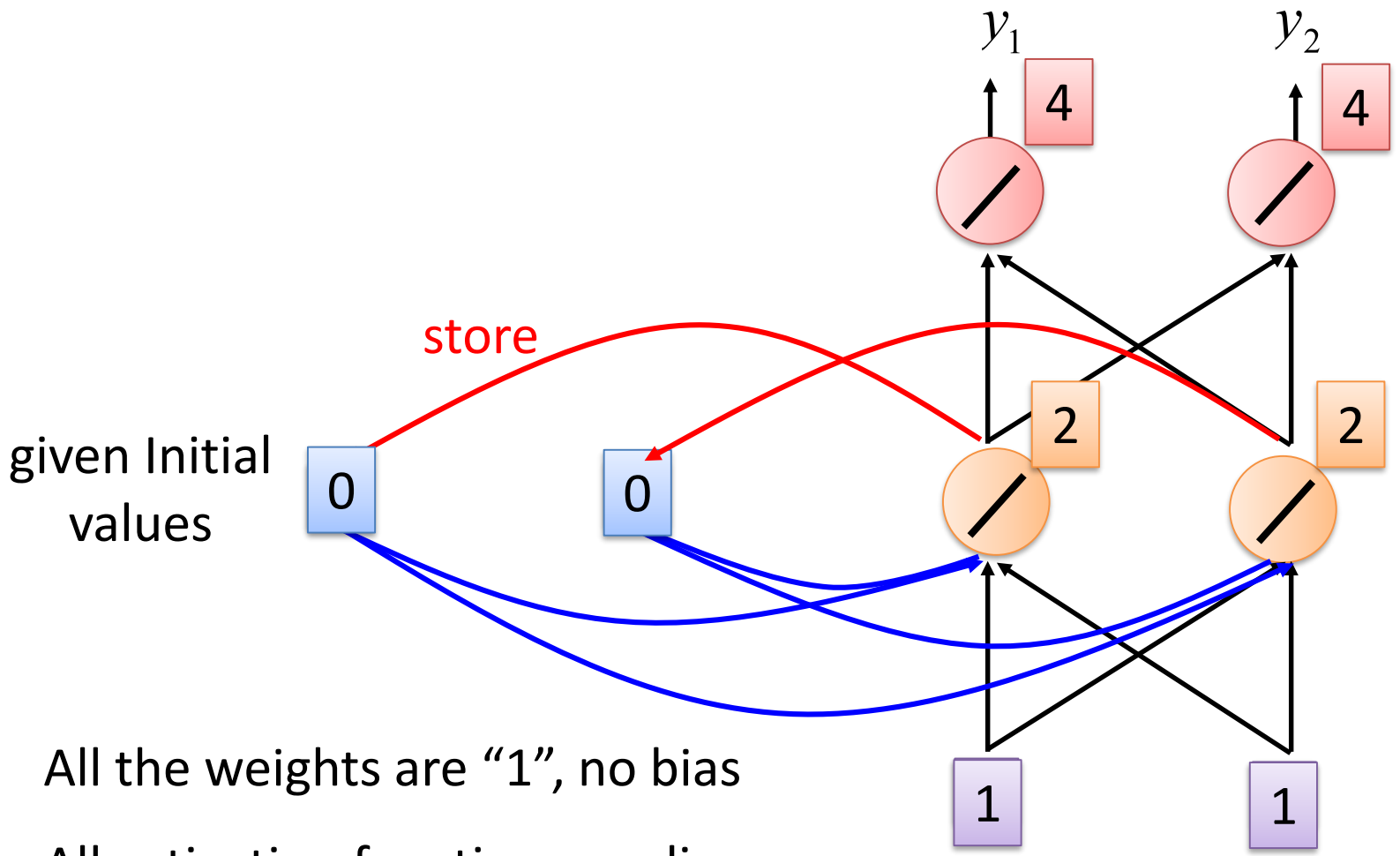


Memory can be considered as another input.

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$



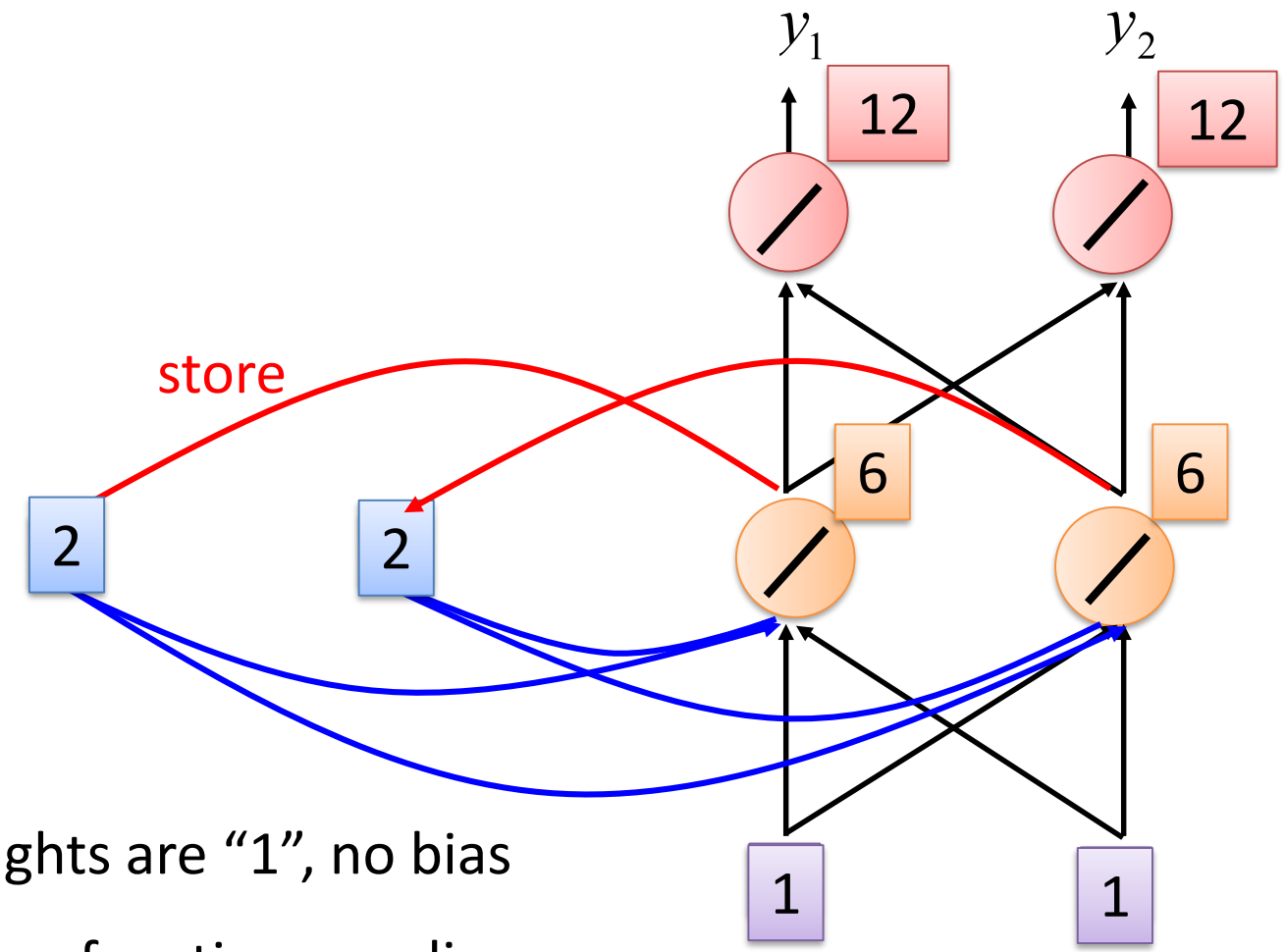
All the weights are “1”, no bias

All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}2\\2\end{bmatrix}\ \cdots\cdots$

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}$



All the weights are “1”, no bias

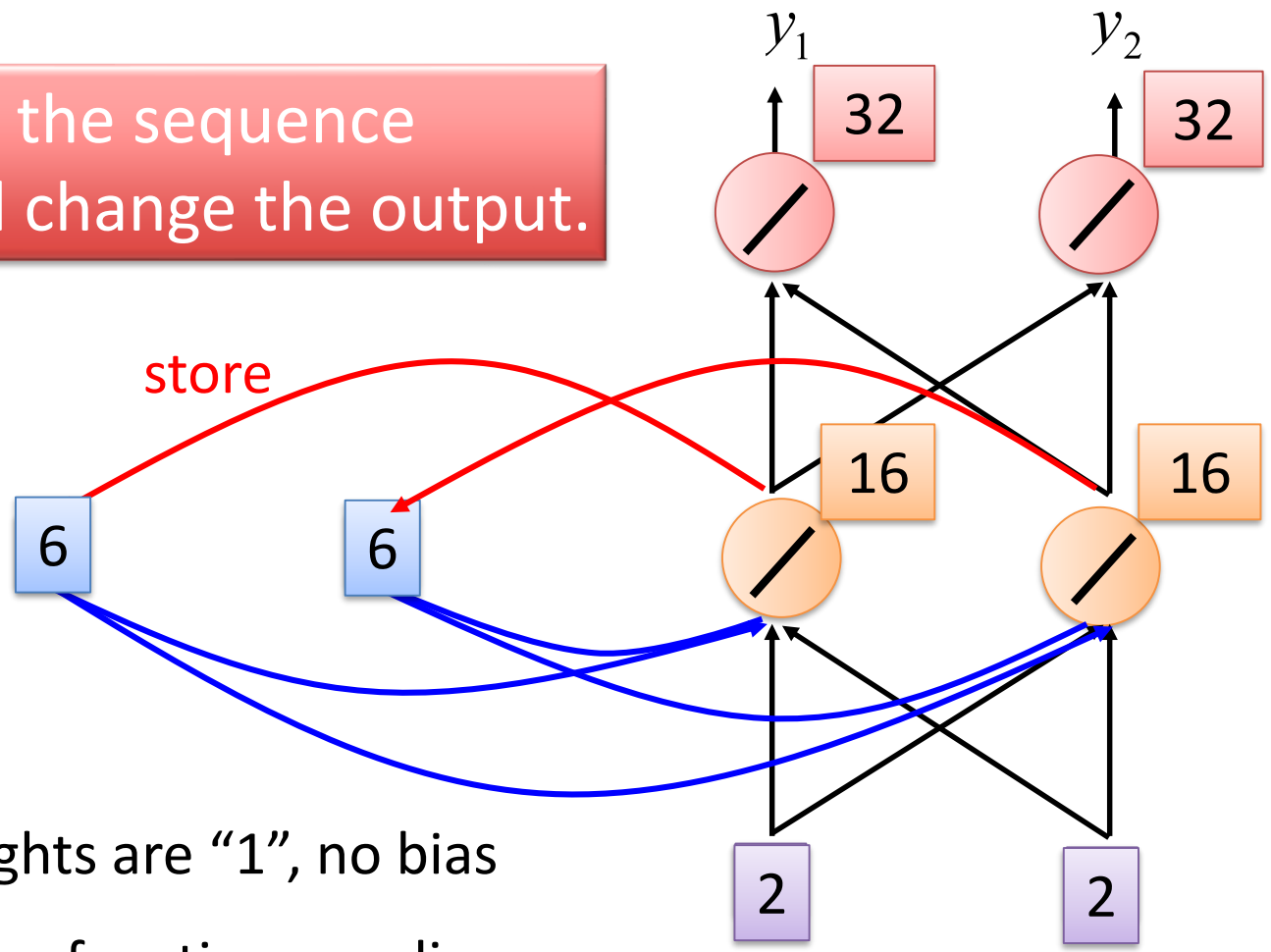
All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}2\\2\end{bmatrix}\ \cdots\cdots$

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$

Changing the sequence order will change the output.



All the weights are “1”, no bias

All activation functions are linear

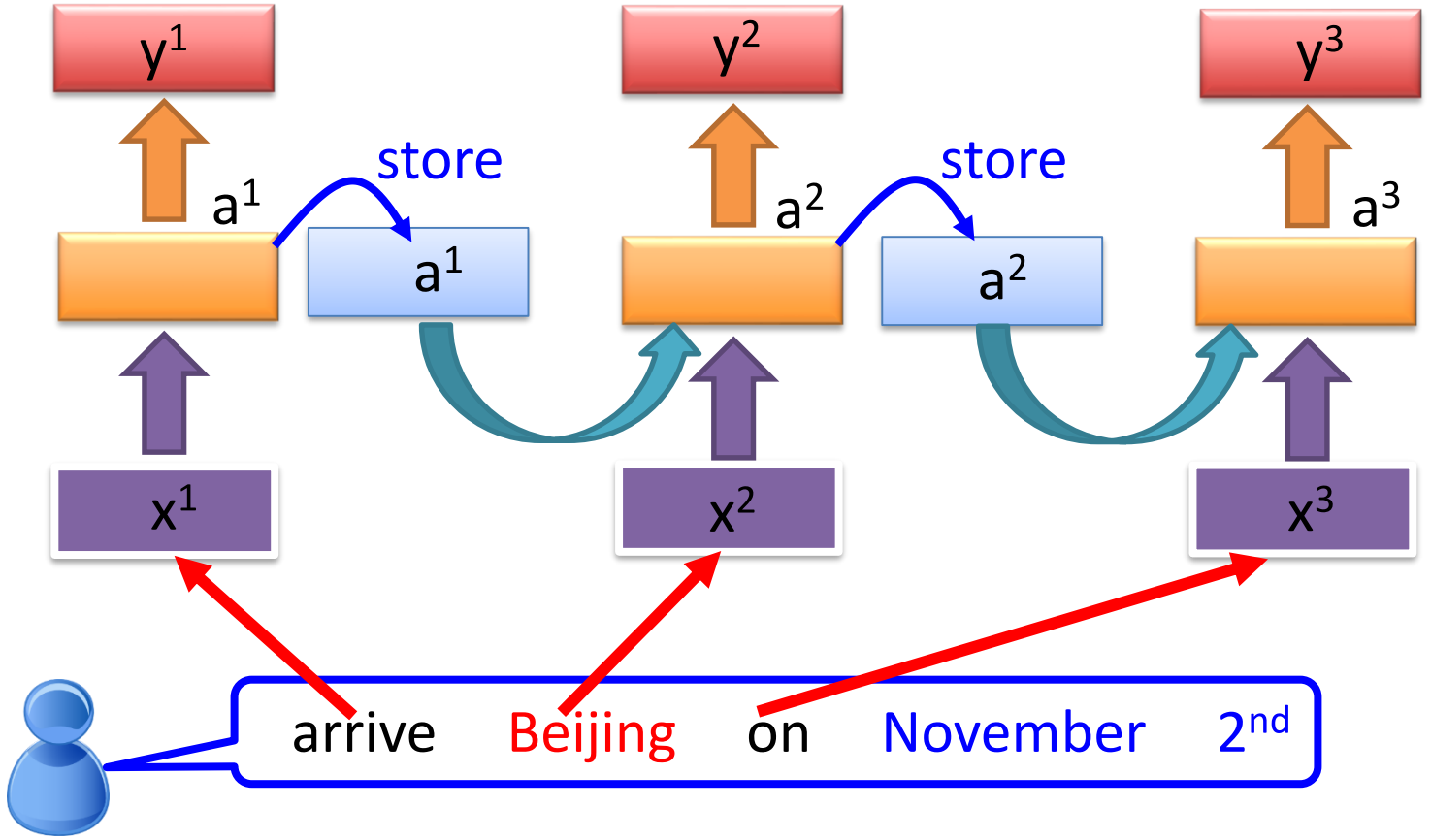
RNN

The same network is used again and again.

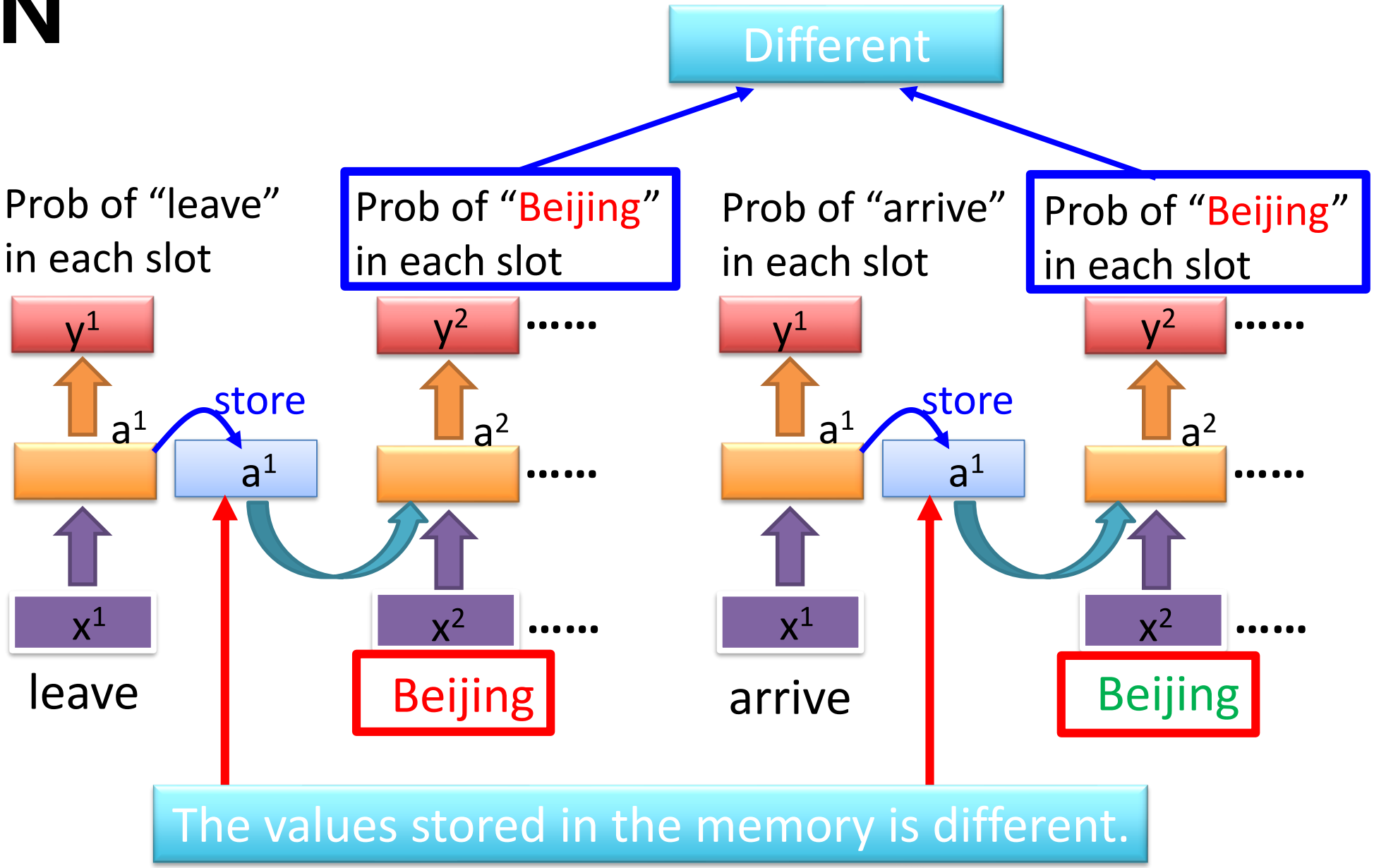
Probability of "arrive" in each slot

Probability of "Beijing" in each slot

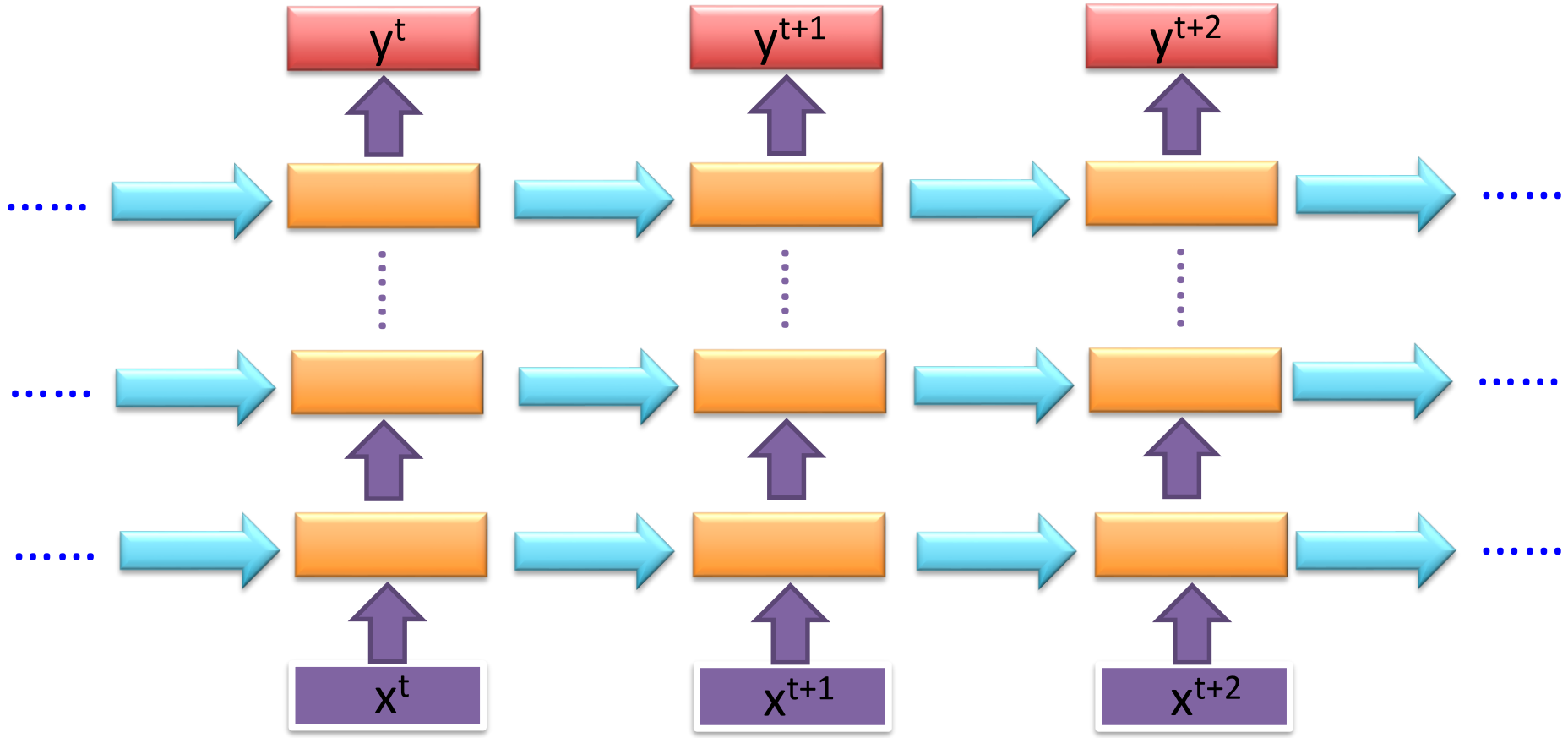
Probability of "on" in each slot



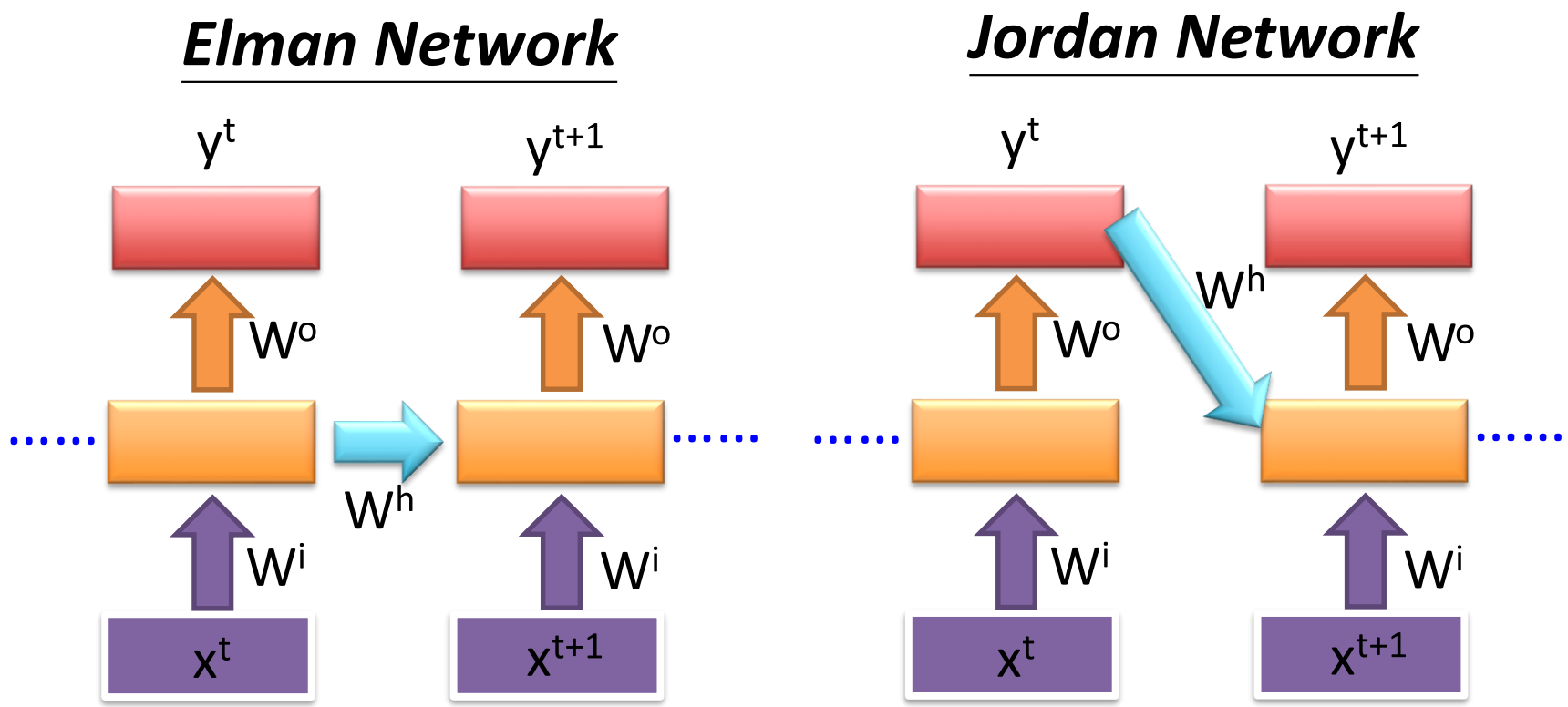
RNN


Different
Prob of "leave" in each slot
Prob of "Beijing" in each slot
Prob of "arrive" in each slot
Prob of "Beijing" in each slot
store
store
leave
Beijing
arrive
Beijing
The values stored in the memory is different.

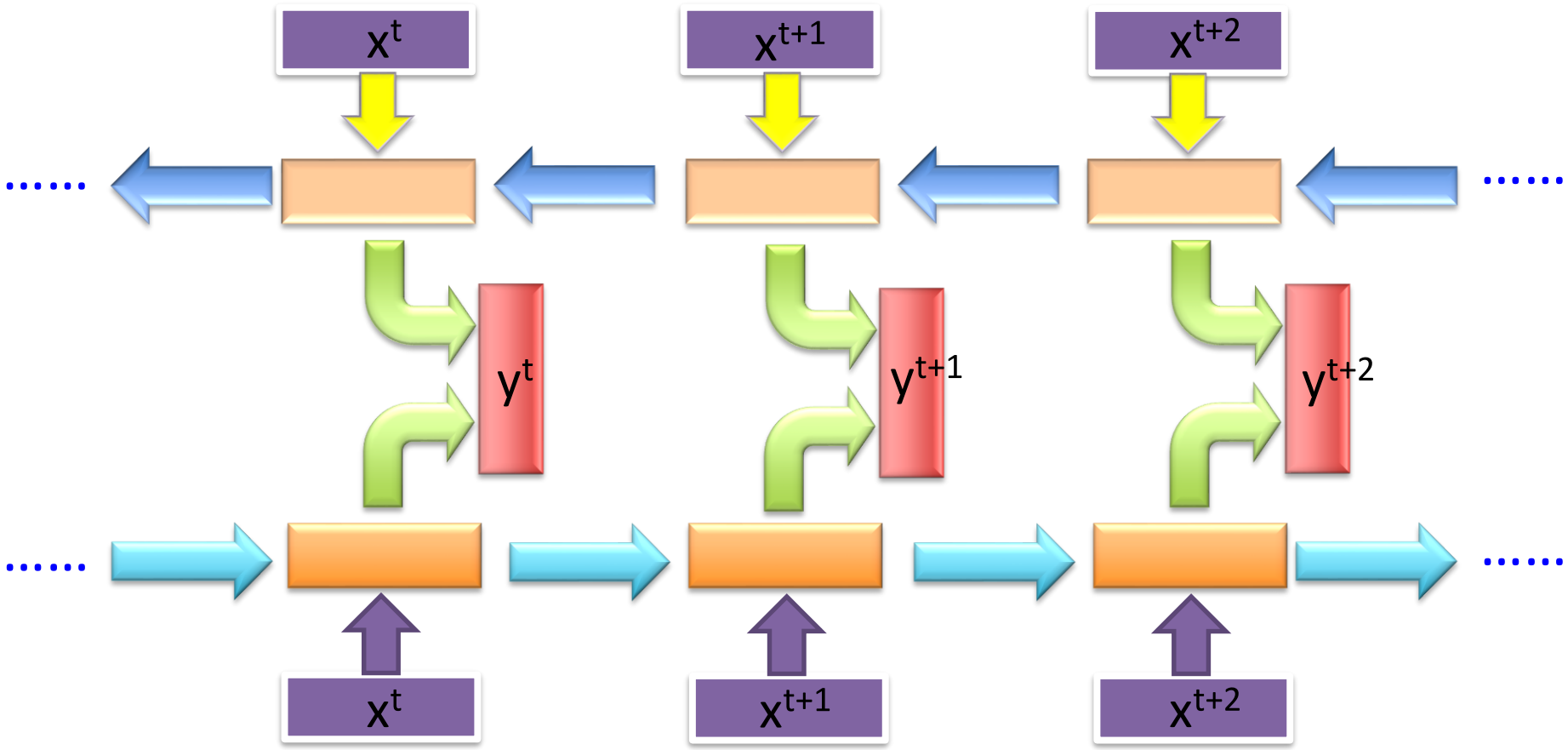
Of course it can be deep …



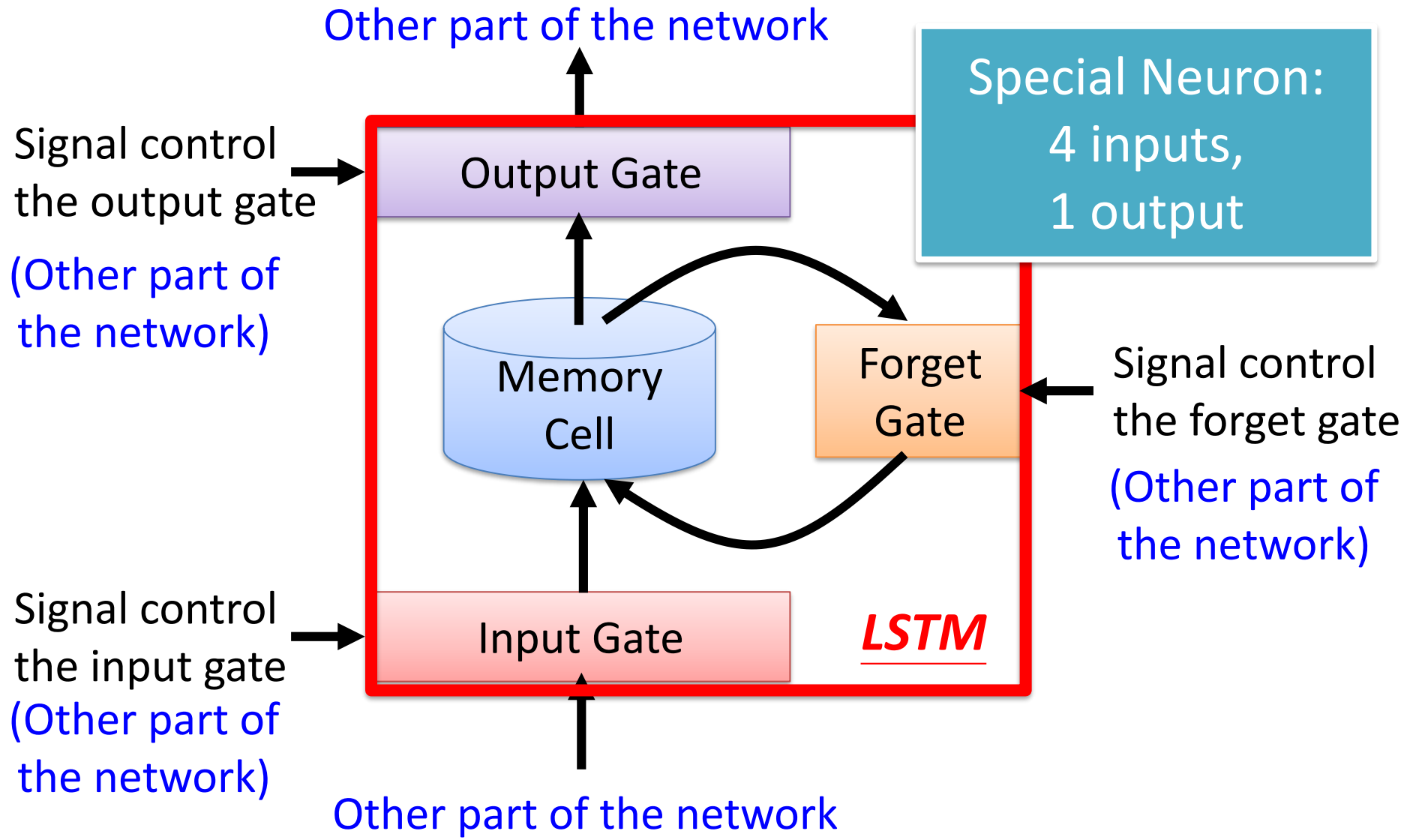
Elman Network & Jordan Network

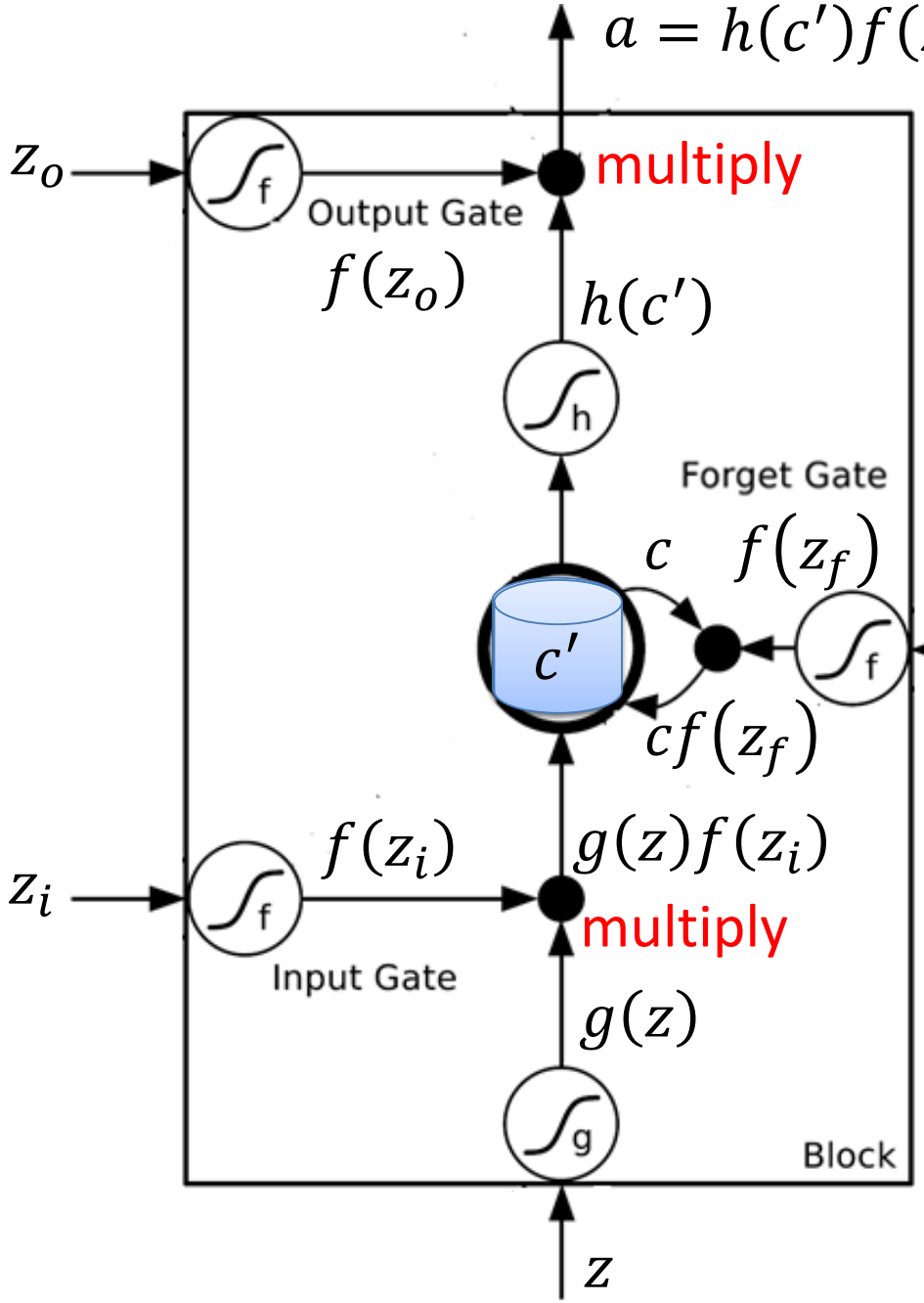


Bidirectional RNN



Long Short-term Memory (LSTM)





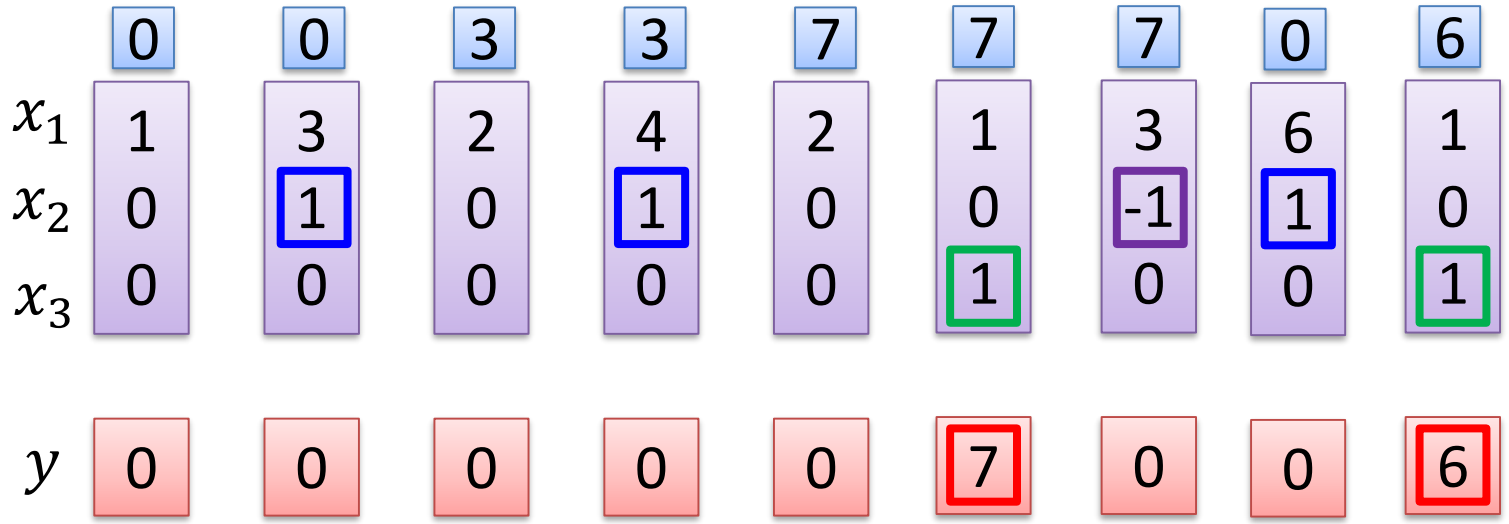
Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$

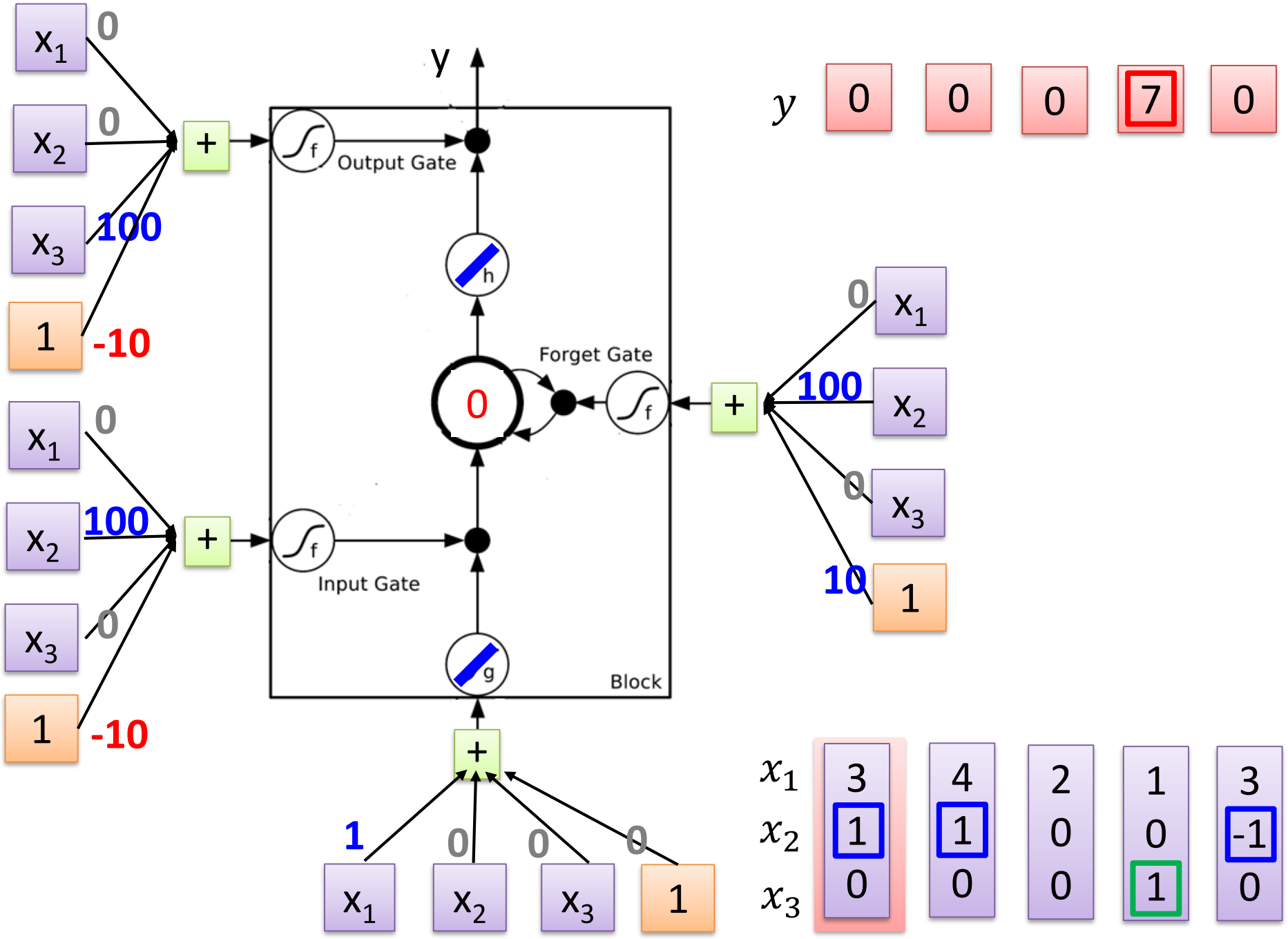
LSTM - Example



When x_2 = 1, add the numbers of x_1 into the memory

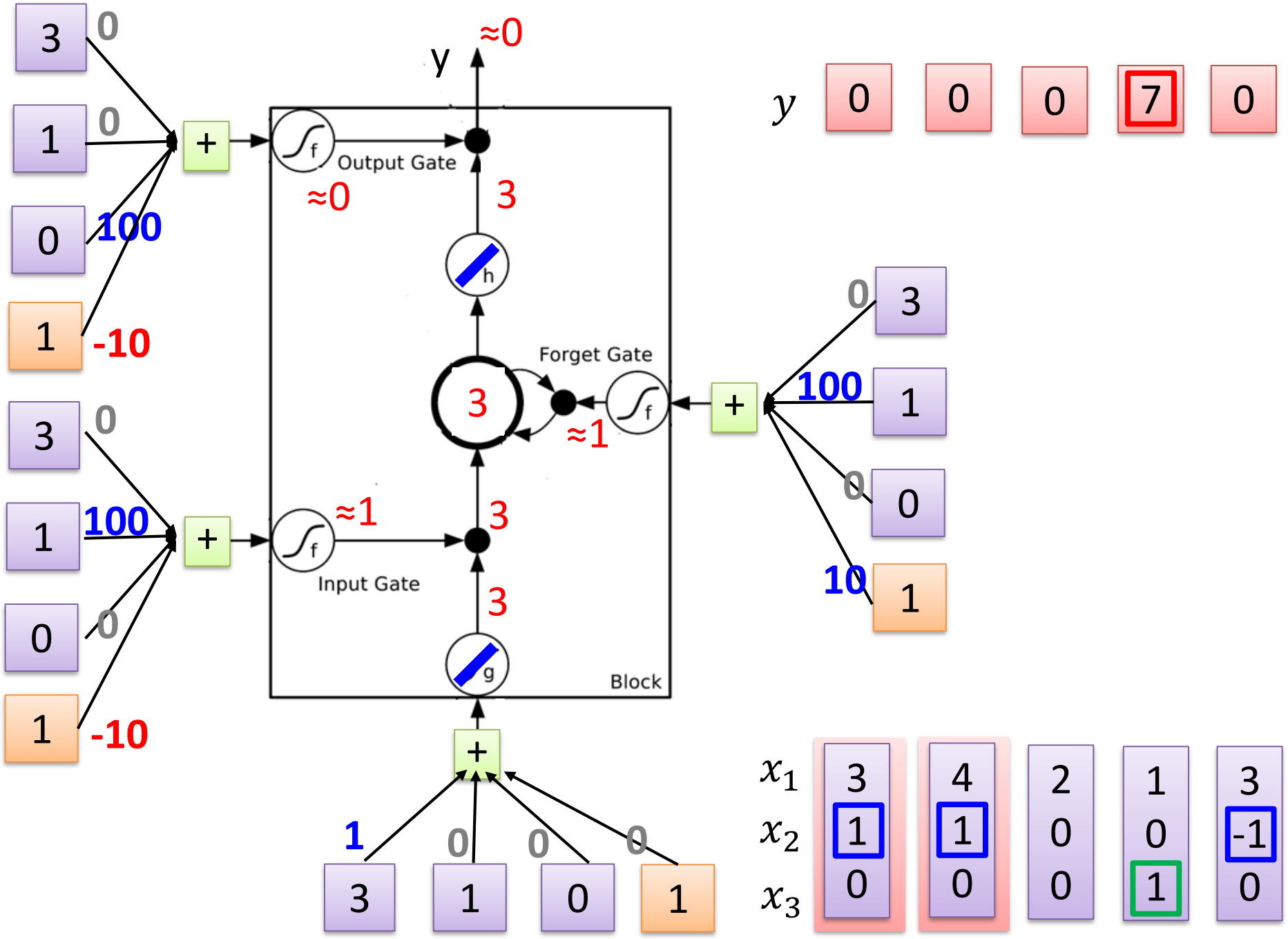
When x_2 = -1, reset the memory

When x_3 = 1, output the number in the memory.



y					
	0	0	0	7	0

x_1	3	4	2	1	3
x_2	1	1	0	0	-1
x_3	0	0	0	1	0

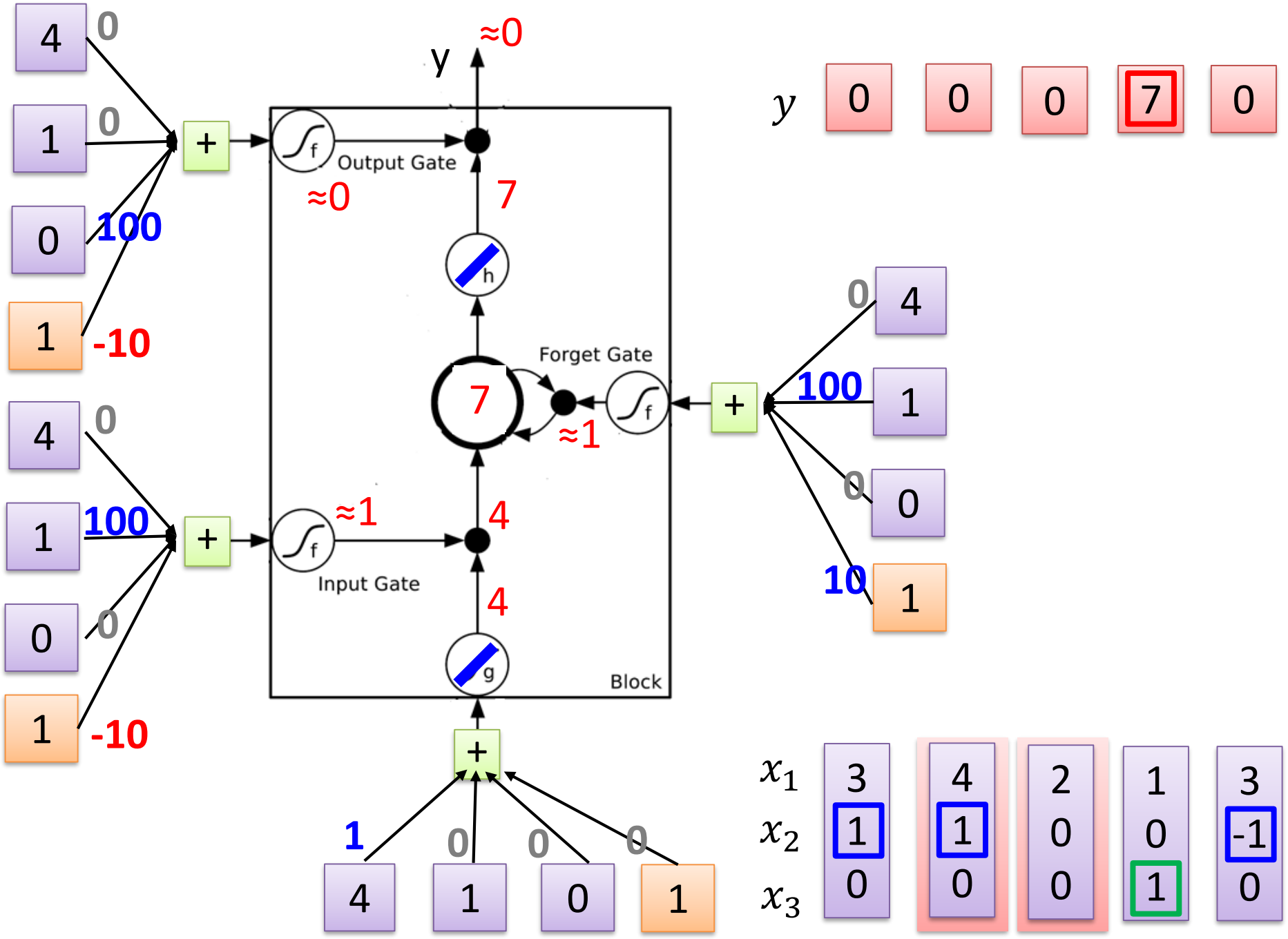


y 0 0 0 7 0

x_1 3 4 2 1 3

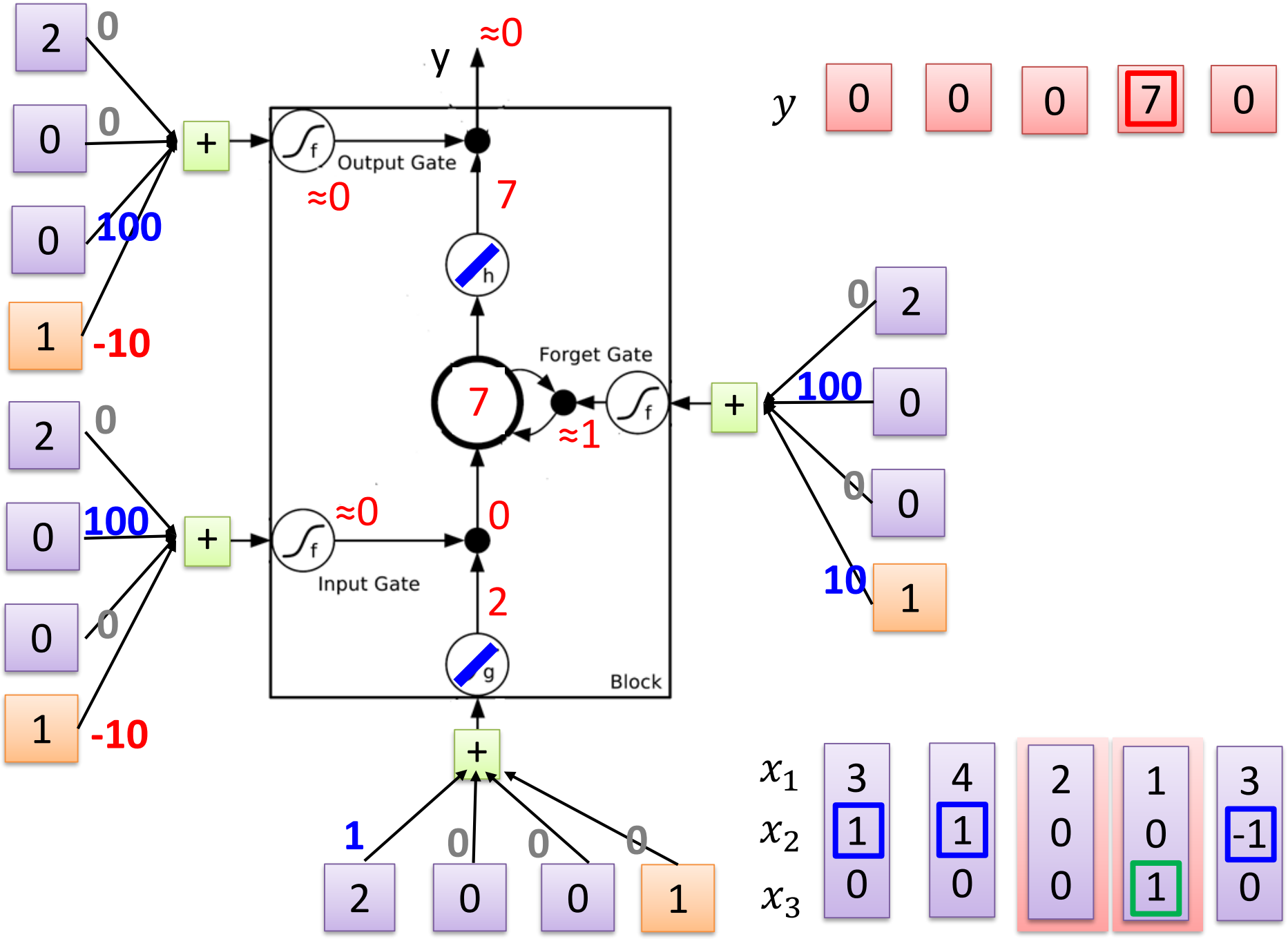
x_2 1 1 0 0 -1

x_3 0 0 0 1 0



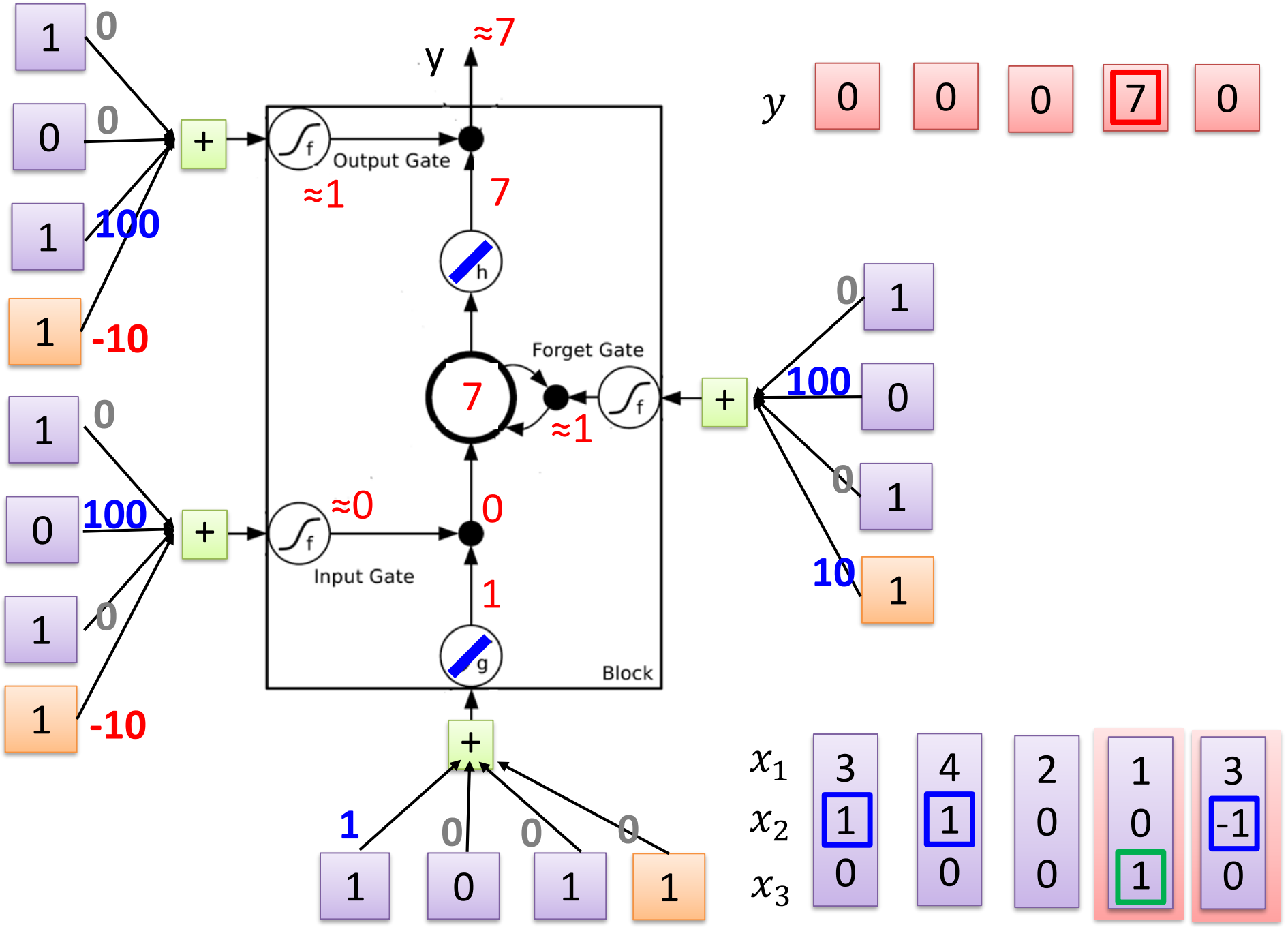
y					
	0	0	0	7	0

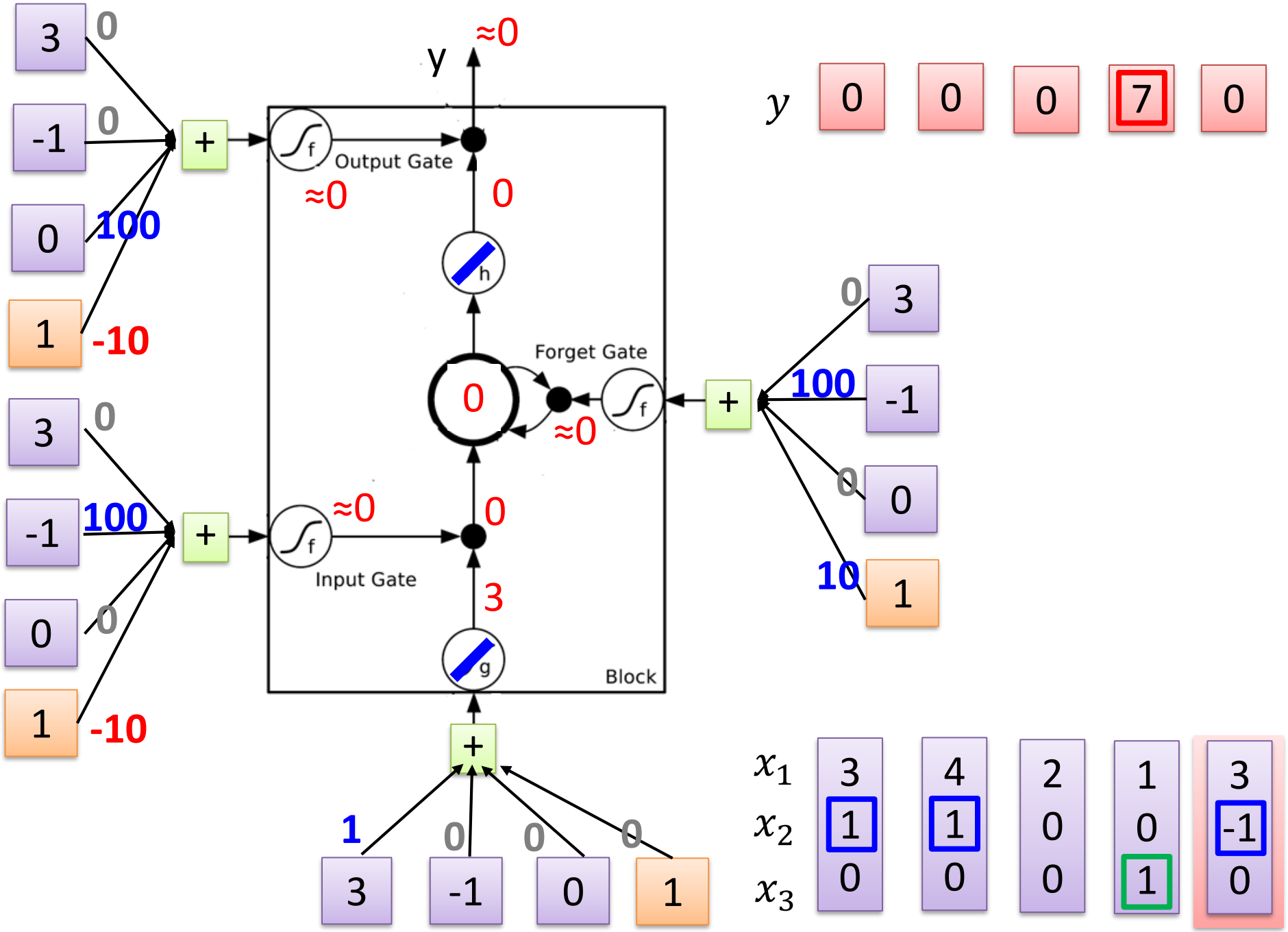
x_1	3	4	2	1	3
x_2	1	1	0	0	-1
x_3	0	0	0	1	0



y 0 0 0 7 0

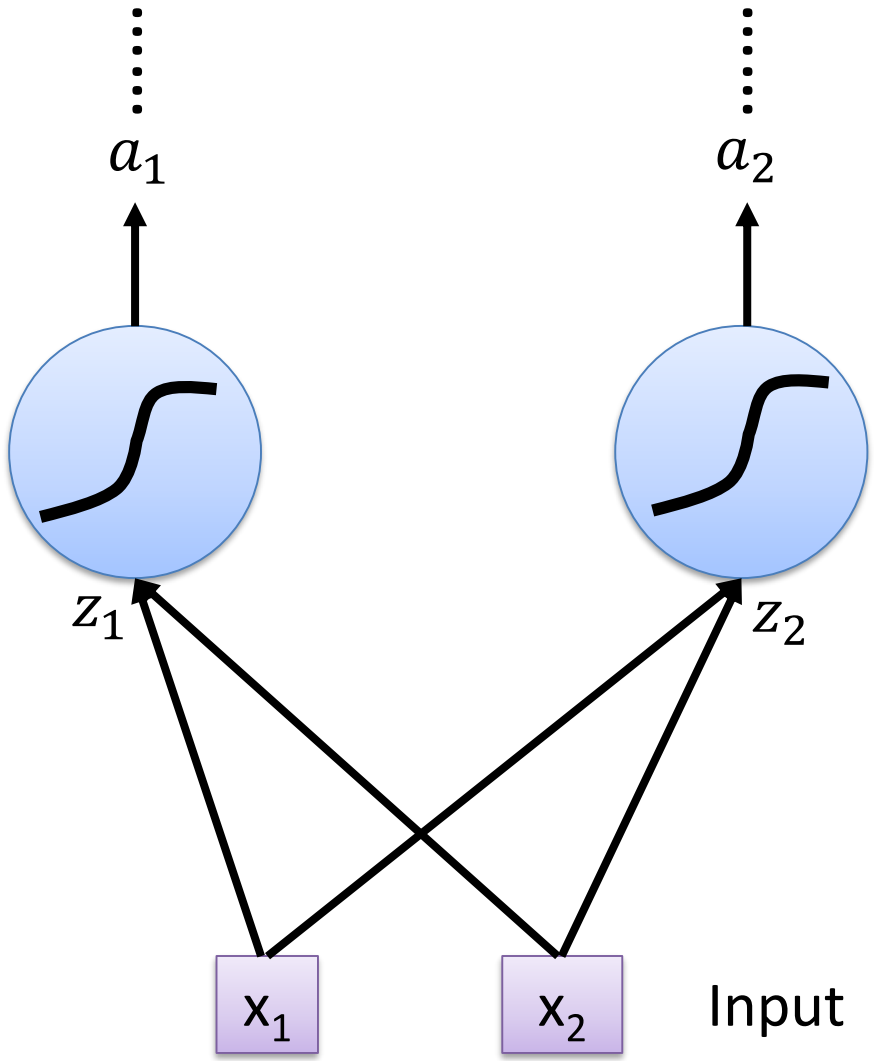
x_1	3	4	2	1	3
x_2	1	1	0	0	-1
x_3	0	0	0	1	0

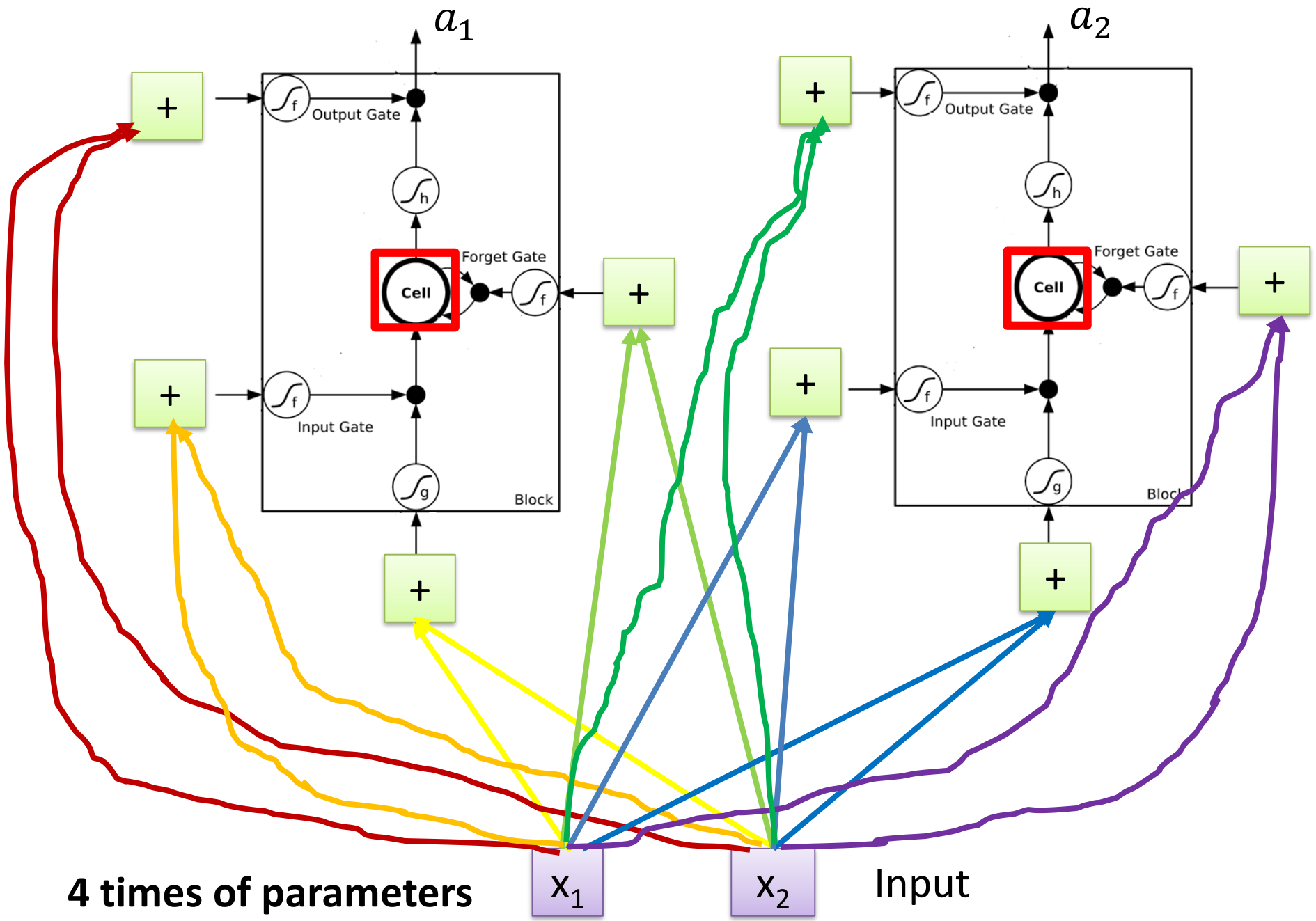

y
≈7
Output Gate
≈1
7
Forget Gate
7
≈1
Input Gate
≈0
0
1
Block
100
-10
100
-10
100
10
1
0
0
0
1
0
0
0
y 0 0 0 7 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0


3
-1
0
1
0
0
100
-10
+
f
Output Gate
≈0
y
≈0
0
h
Forget Gate
0
≈0
3
-1
0
1
0
100
0
10
+
f
3
-1
0
1
0
100
0
-10
+
f
≈0
Input Gate
0
3
g
Block
+
1
0
0
0
3
-1
0
1
y
0
0
0
7
0
x_1
x_2
x_3
3
1
0
4
1
0
2
0
0
1
0
1
3
-1
0

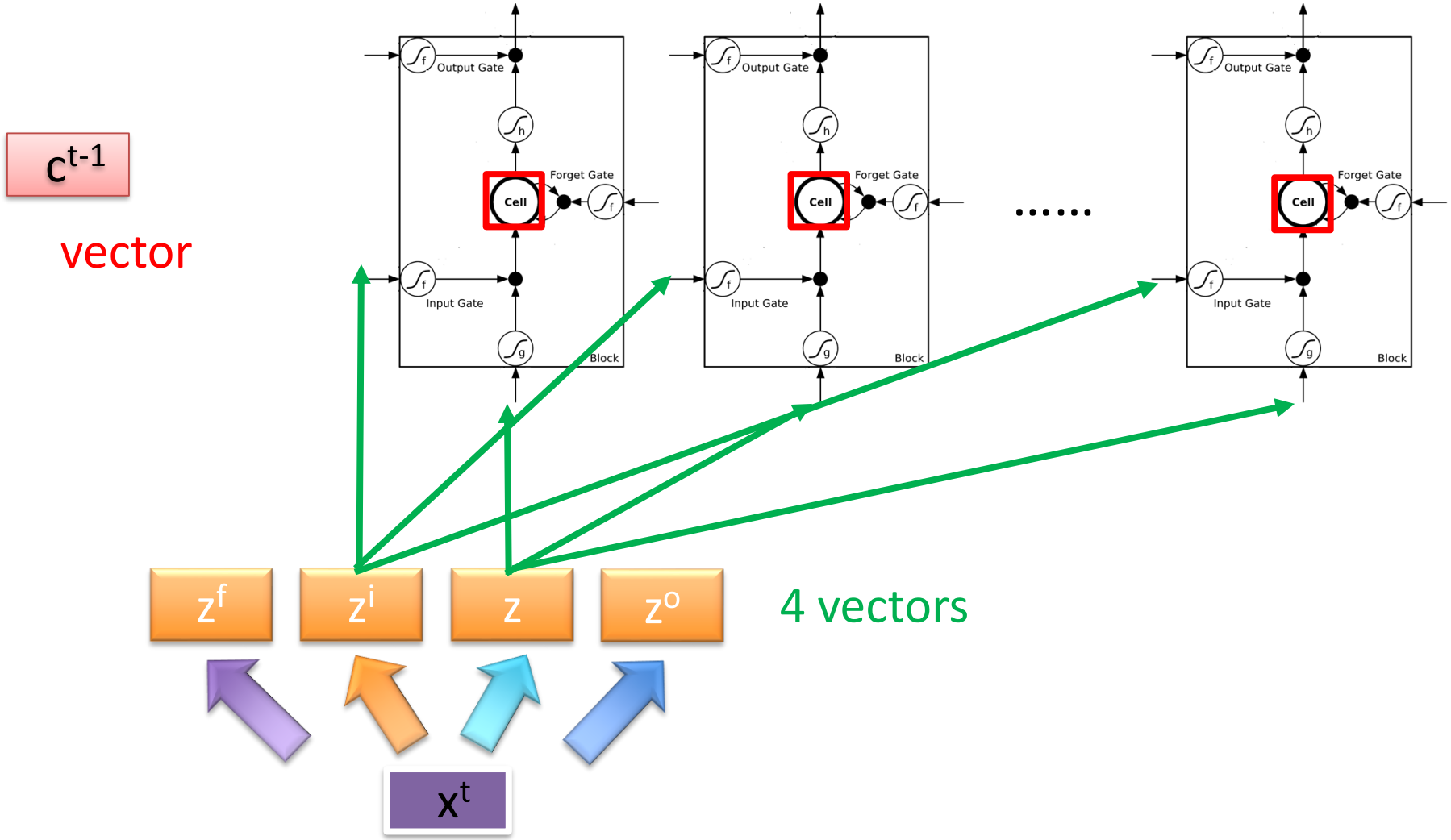
Original Network:

- Simply replace the neurons with LSTM

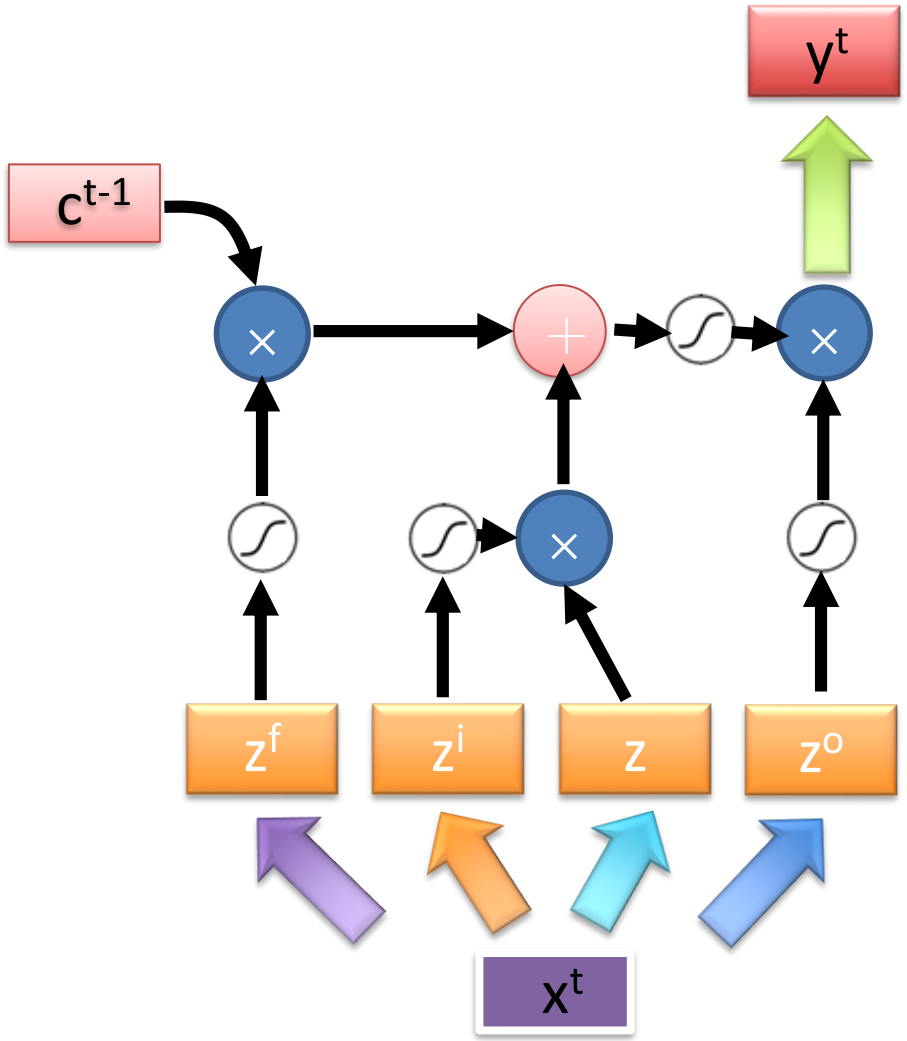


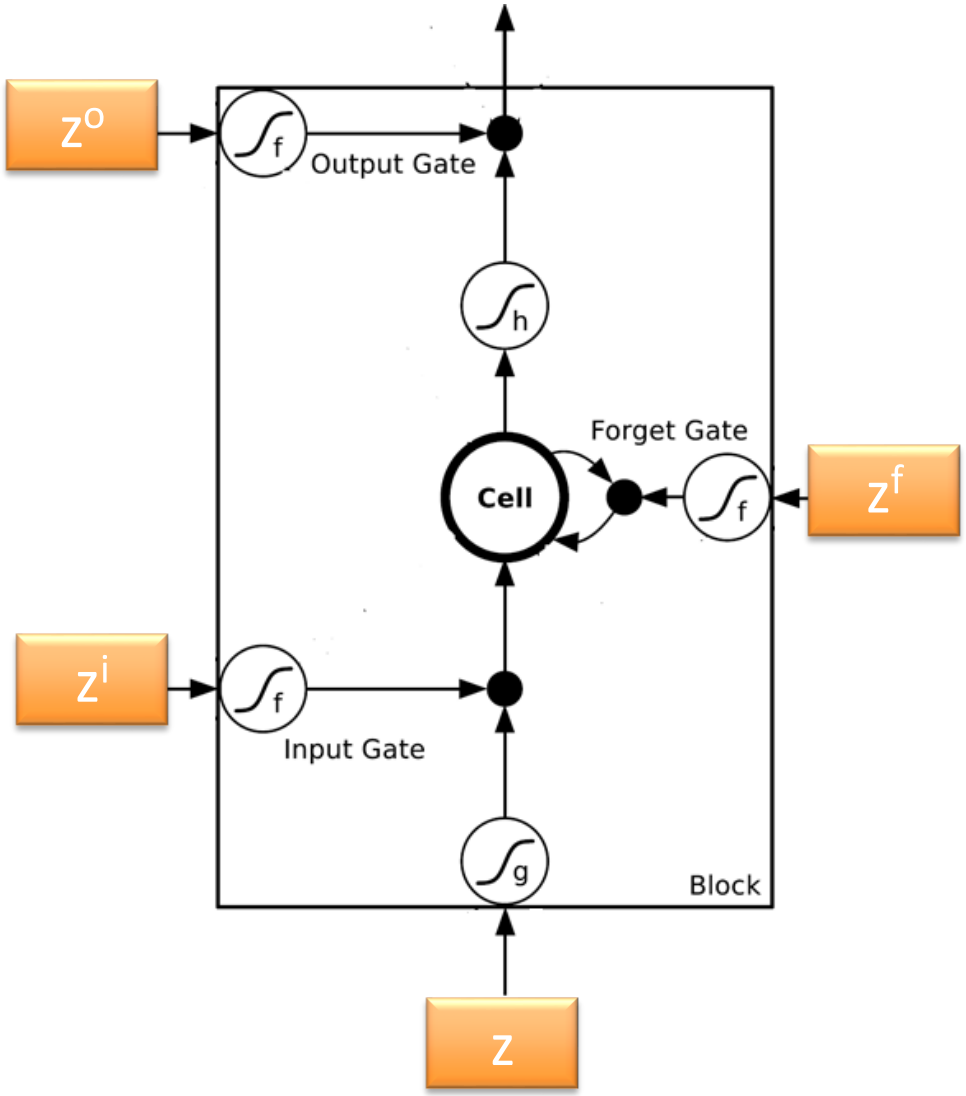

a_1
a_2
Output Gate
Forget Gate
Input Gate
Cell
Block
f
h
g
+
4 times of parameters
x_1
x_2
Input

LSTM


c^{t-1}
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
.....
z^f
z^i
z
z^o
4 vectors
x^t

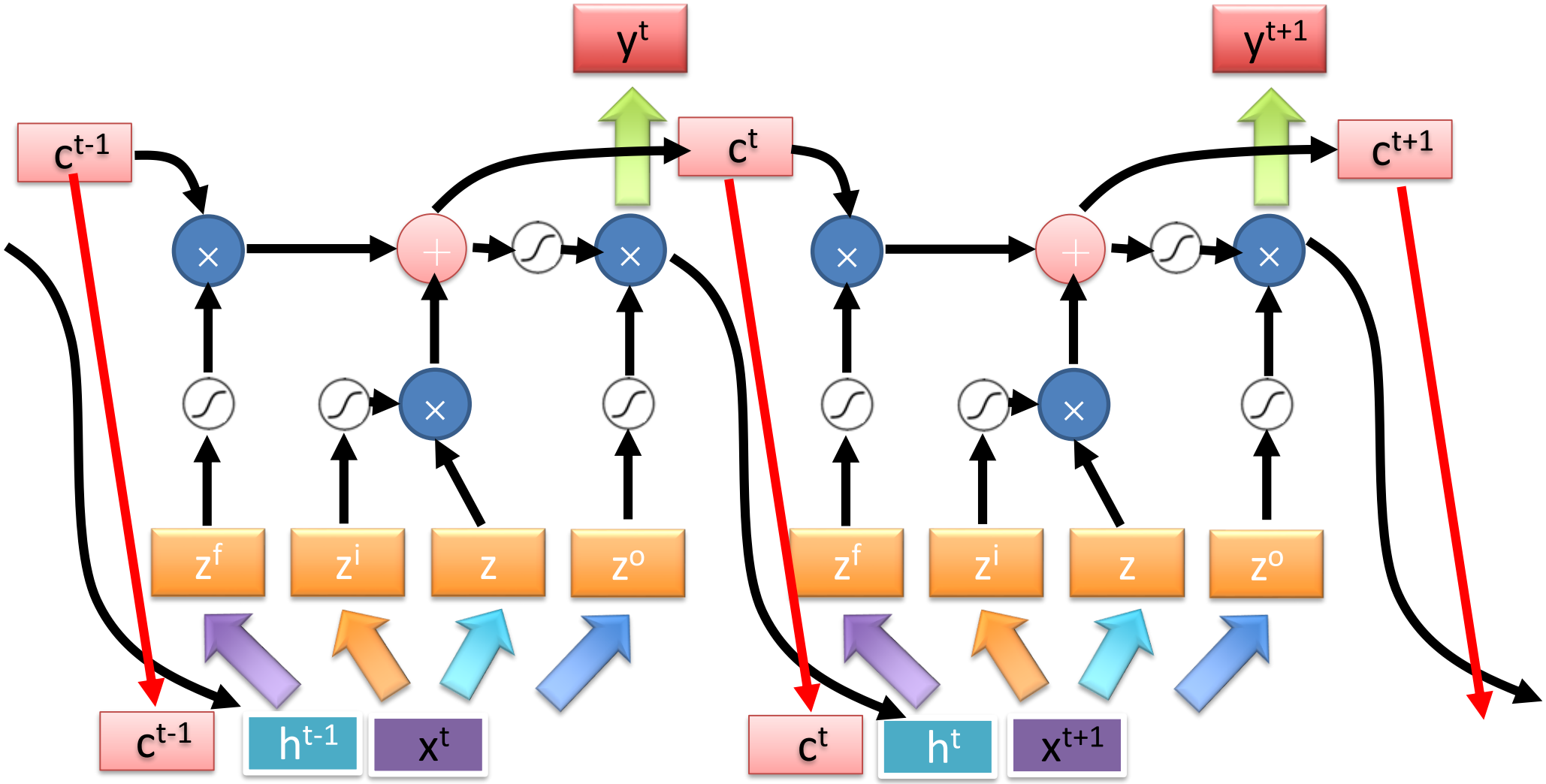
LSTM


y^t
c^{t-1}
z^f
z^i
z
z^o
x^t

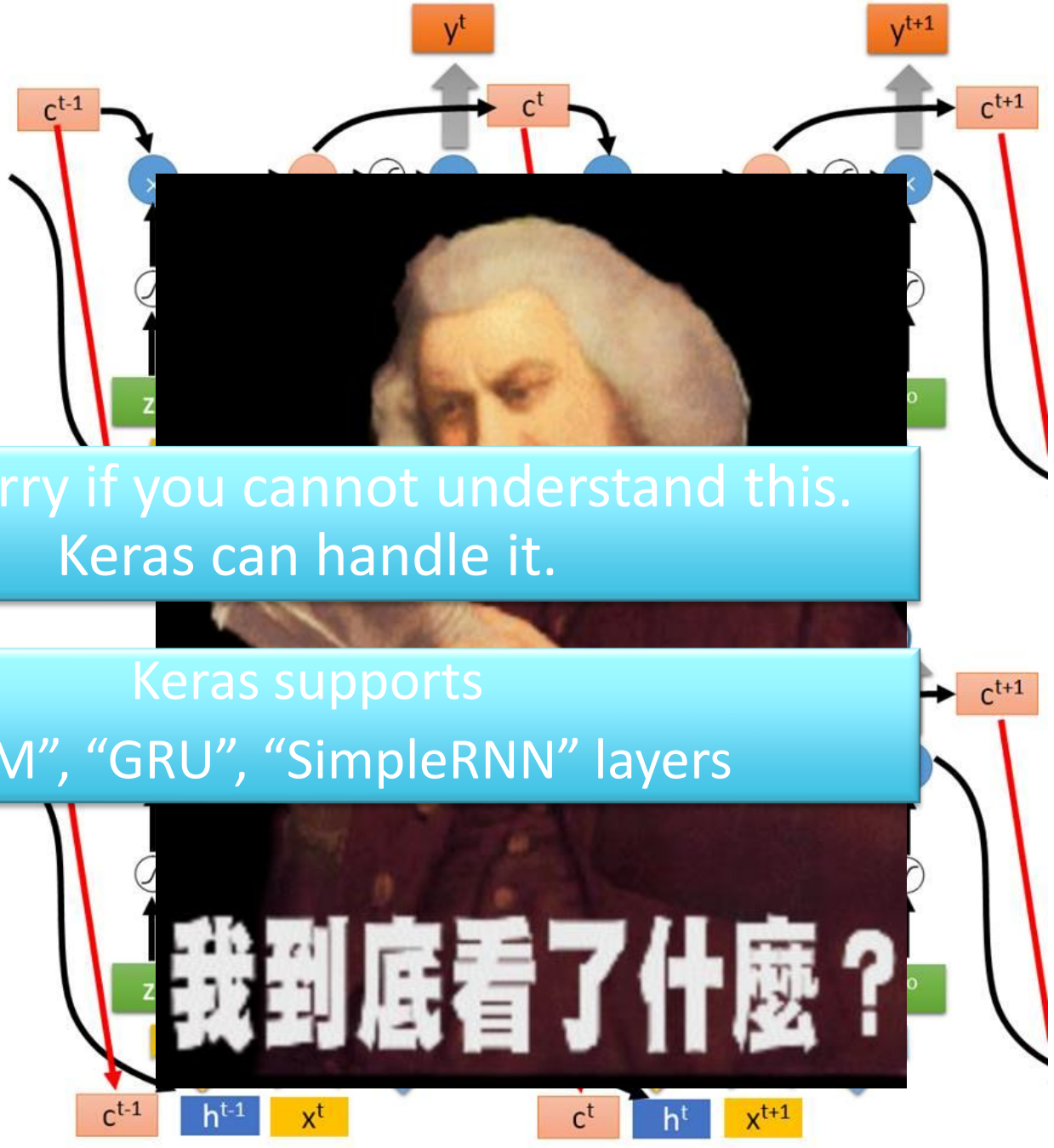

z^o
Output Gate
h
Forget Gate
Cell
z^f
z^i
Input Gate
g
Block
z

LSTM

Extension: “peephole”



Multiple-layer LSTM



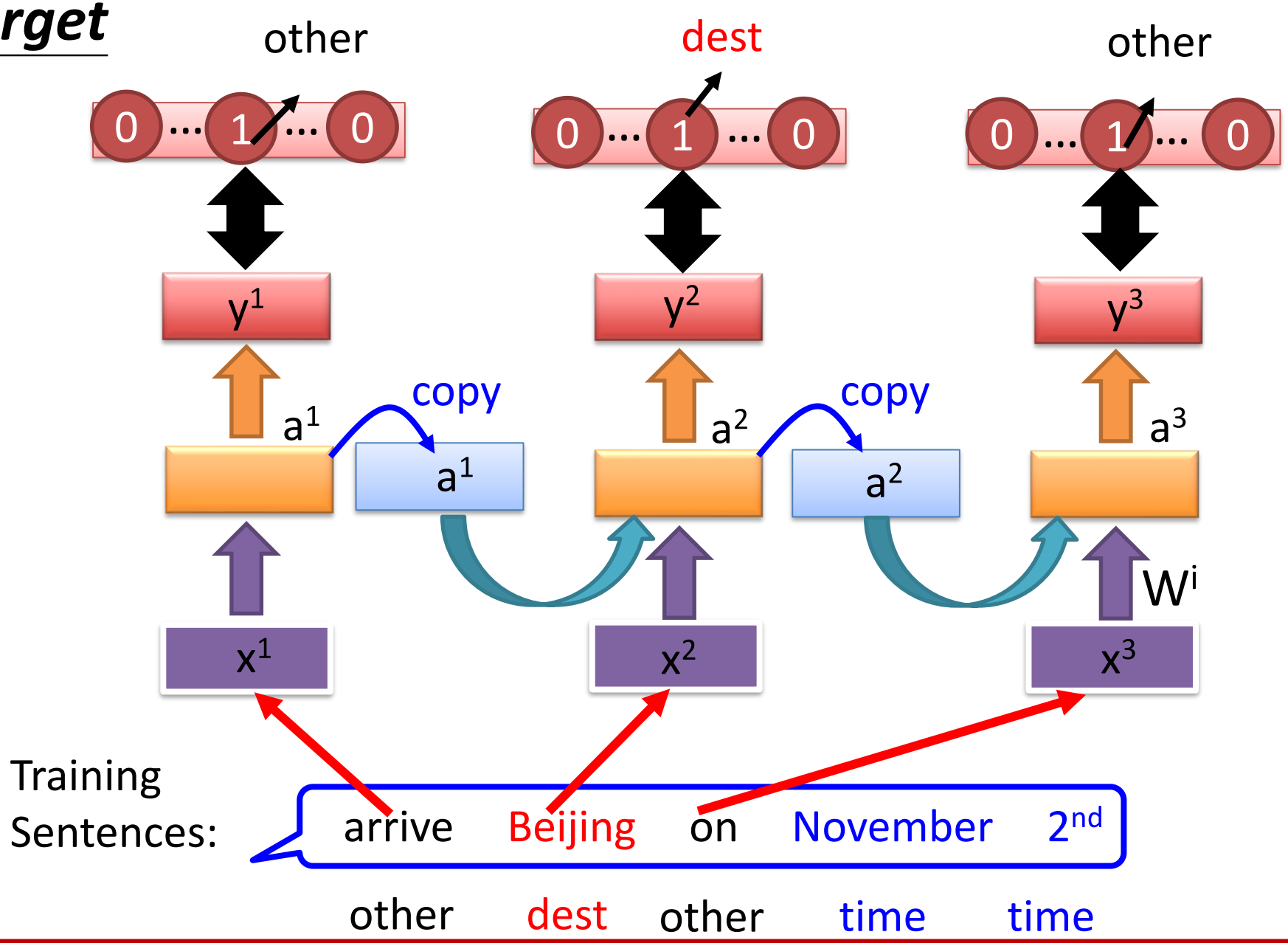
Don't worry if you cannot understand this.
Keras can handle it.

Keras supports
"LSTM", "GRU", "SimpleRNN" layers

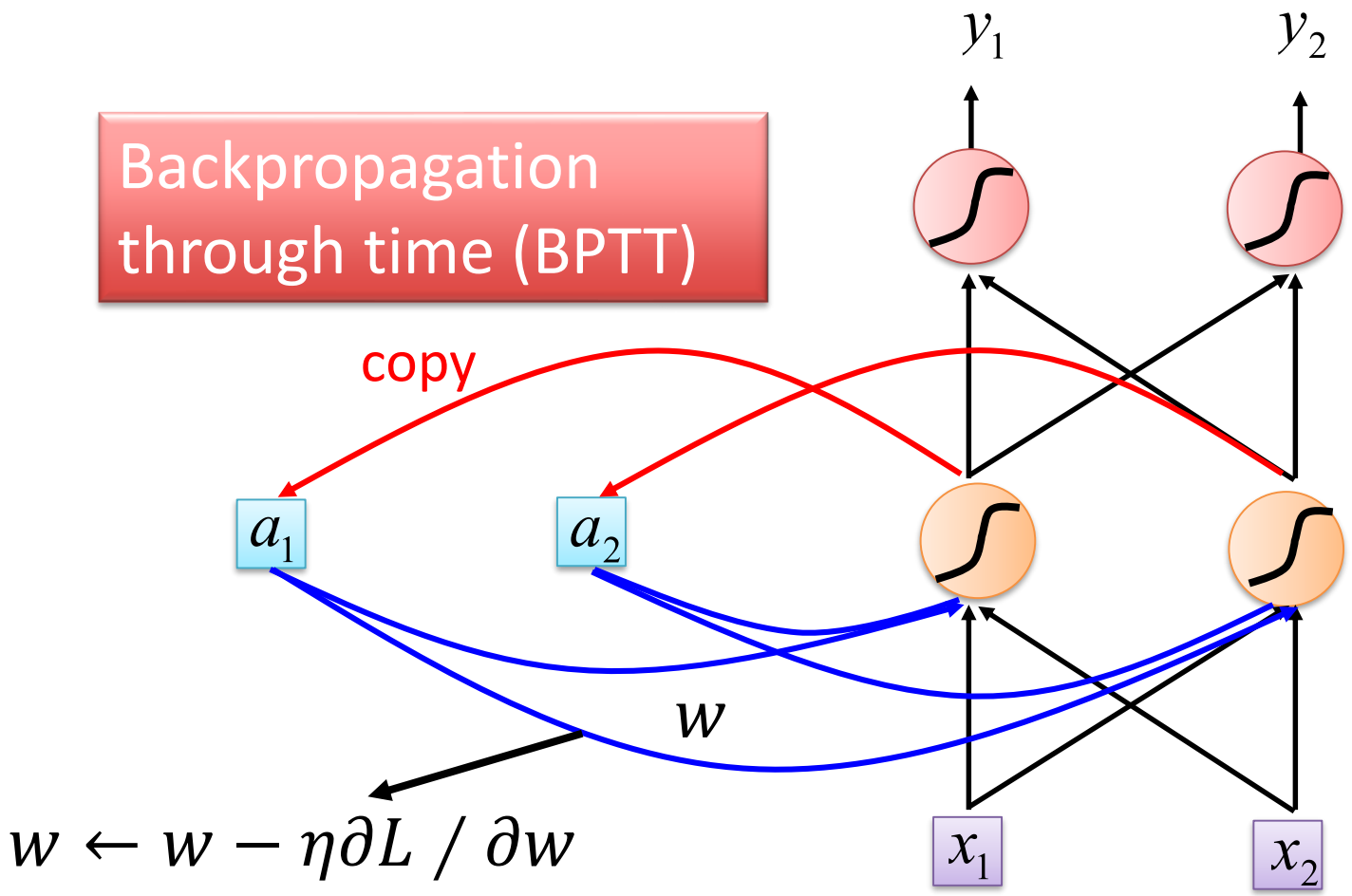
This is quite standard now.

https://img.komicolle.org/2015-09-20/src/14426967627131.gif

Learning Target

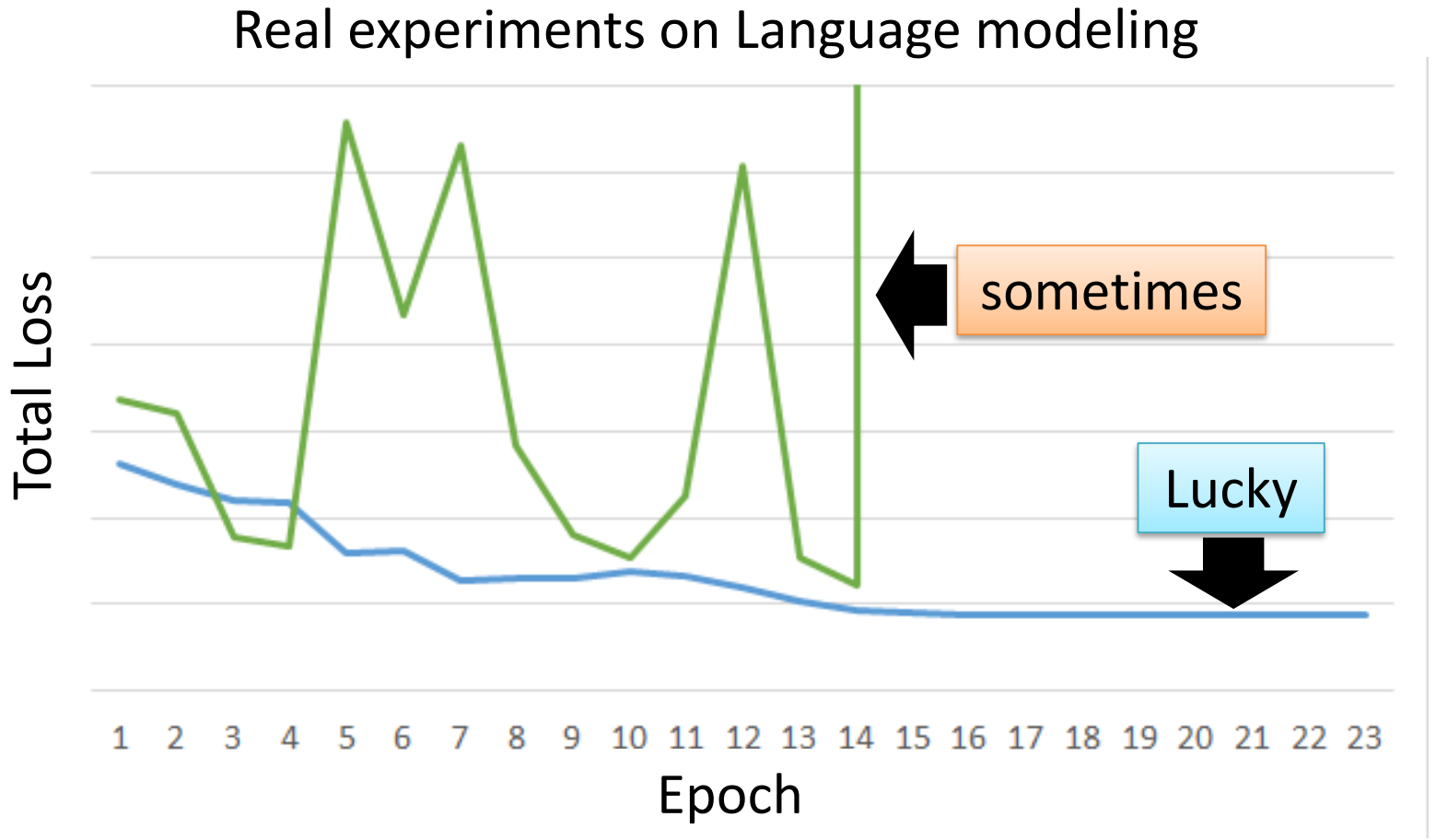


Learning

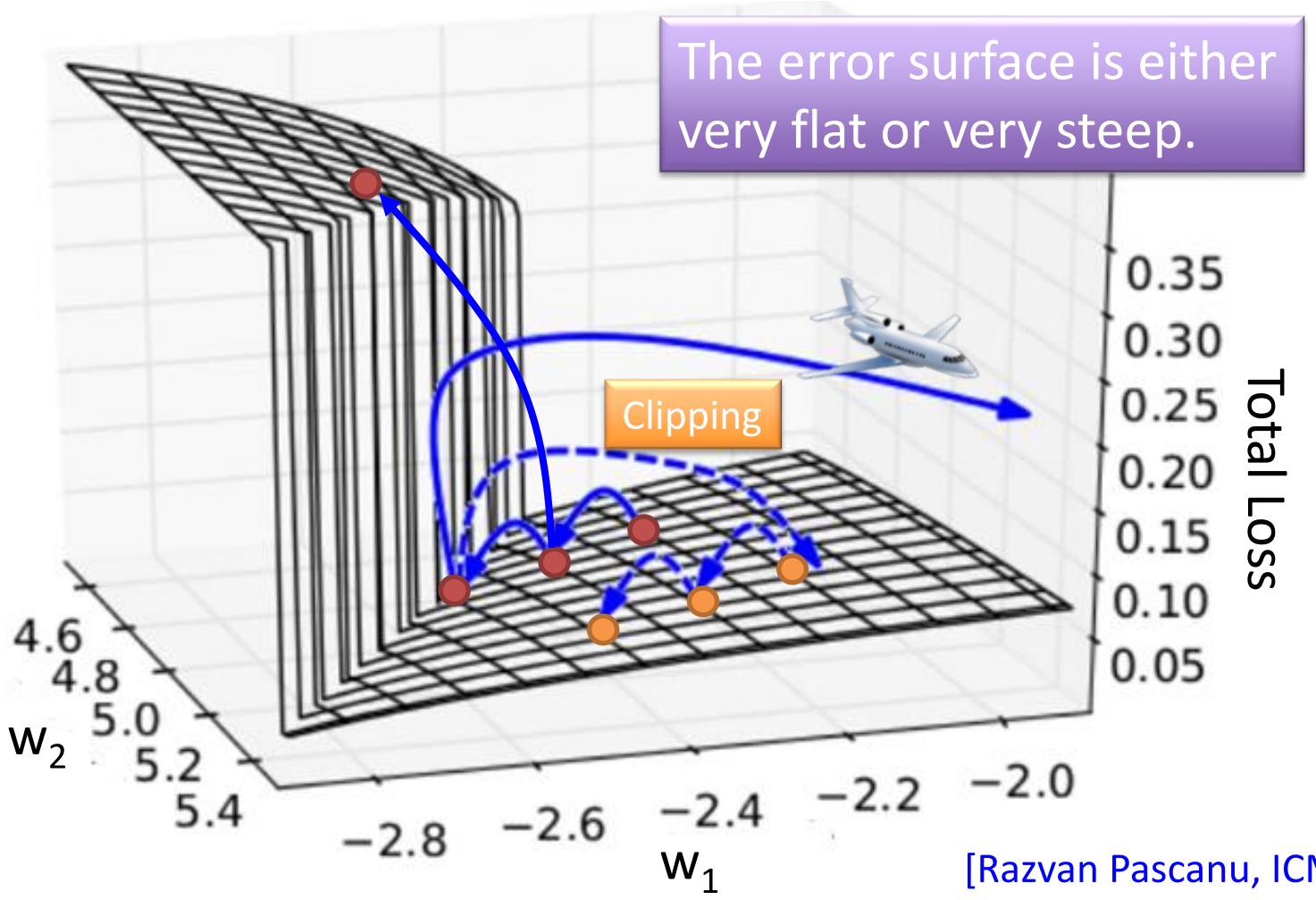

Backpropagation through time (BPTT)
copy
y_1
y_2
a_1
a_2
w
x_1
x_2
$w \leftarrow w - \eta \partial L / \partial w$

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

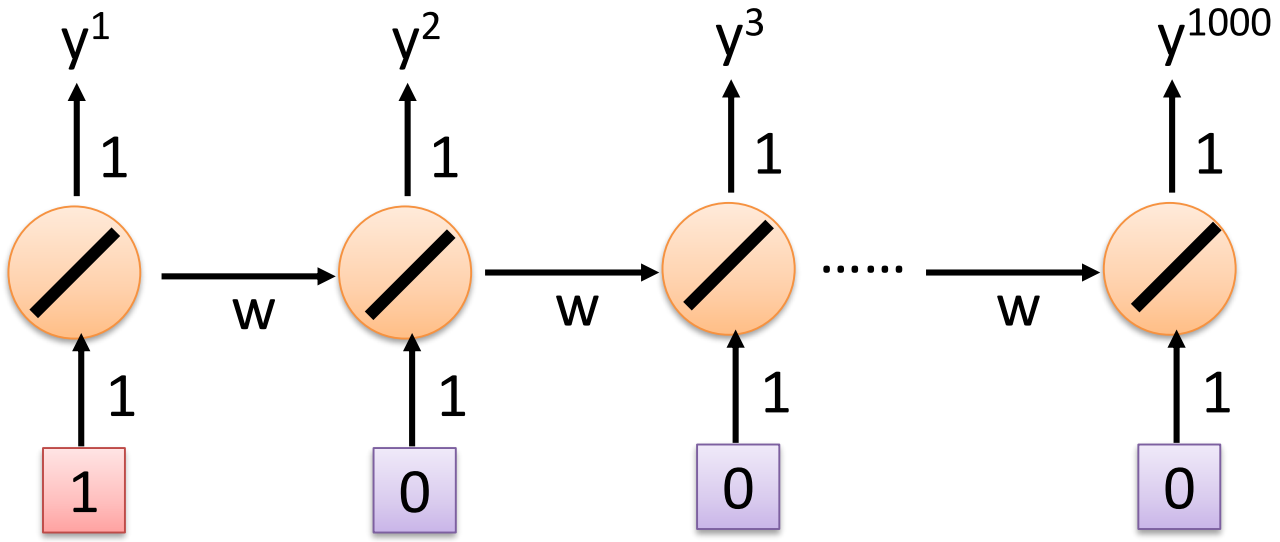
Large $\partial L/\partial w$ → Small Learning rate?

$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

small $\partial L/\partial w$ → Large Learning rate?

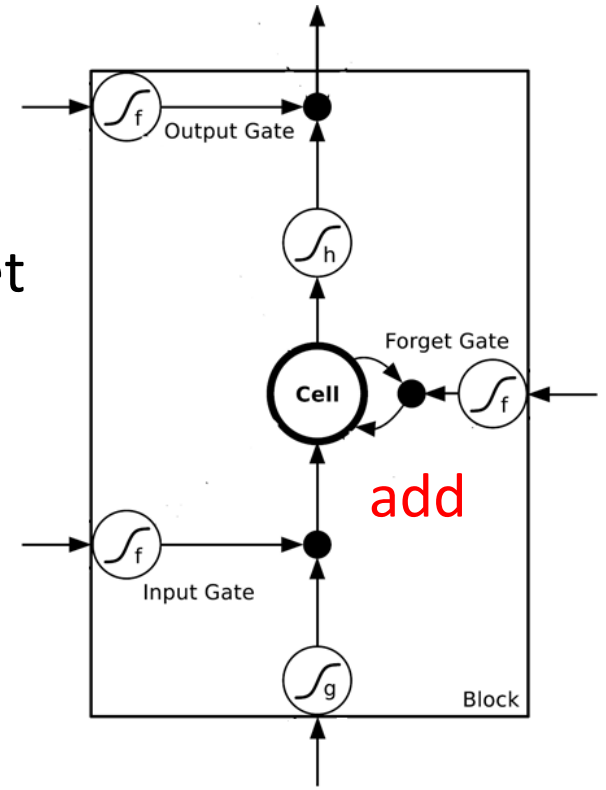
Toy Example



Helpful Techniques

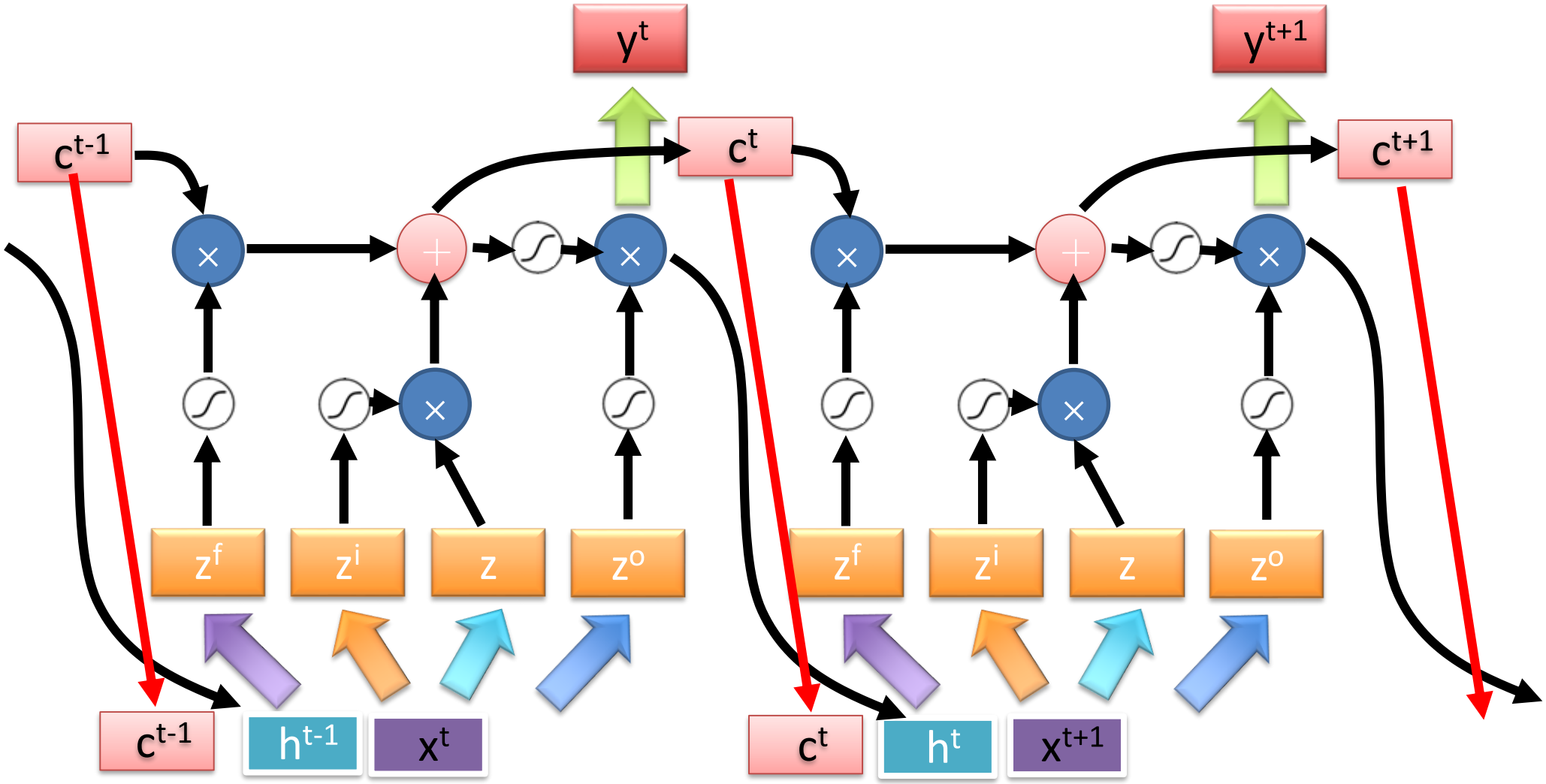
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
    - Memory and input are ***added***
    - The influence never disappears unless forget gate is closed

No Gradient vanishing
(If forget gate is opened.)



LSTM

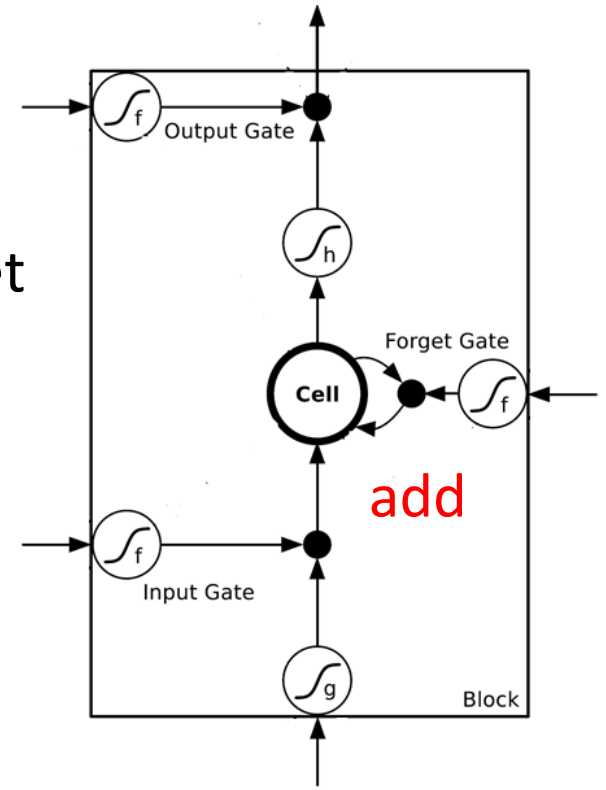
Extension: “peephole”



Helpful Techniques

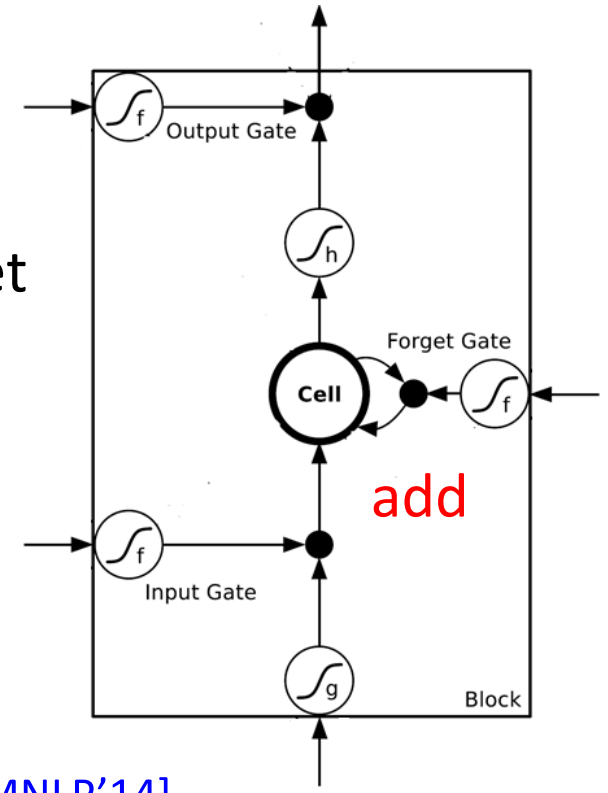
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
    - Memory and input are ***added***
    - The influence never disappears unless forget gate is closed

No Gradient vanishing
(If forget gate is opened.)



Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
    - Memory and input are ***added***
    - The influence never disappears unless forget gate is closed



[Cho, EMNLP'14]

→ No Gradient vanishing
(If forget gate is opened.)

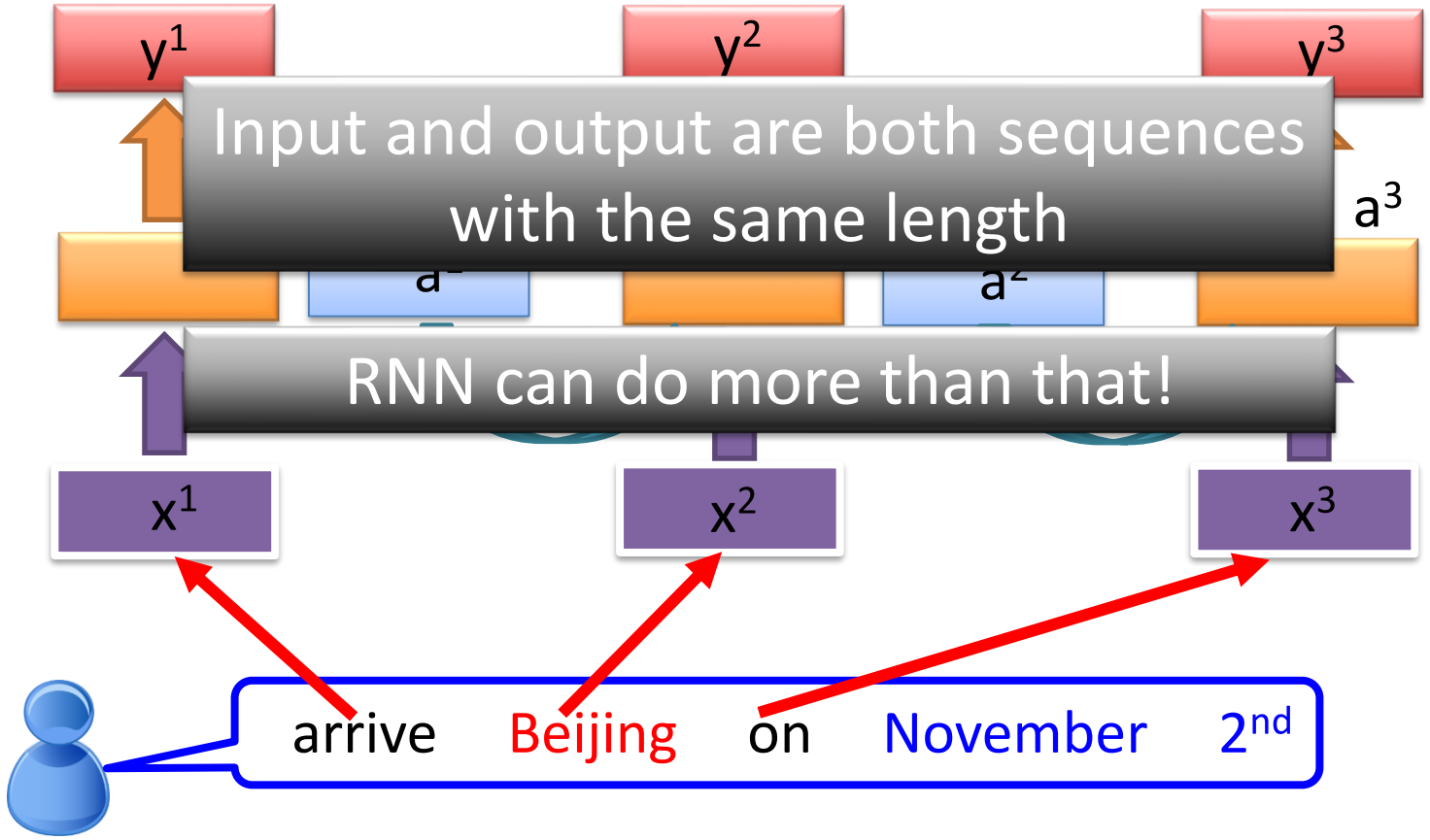
Gated Recurrent Unit (GRU): simpler than LSTM

More Applications ……

Probability of "arrive" in each slot

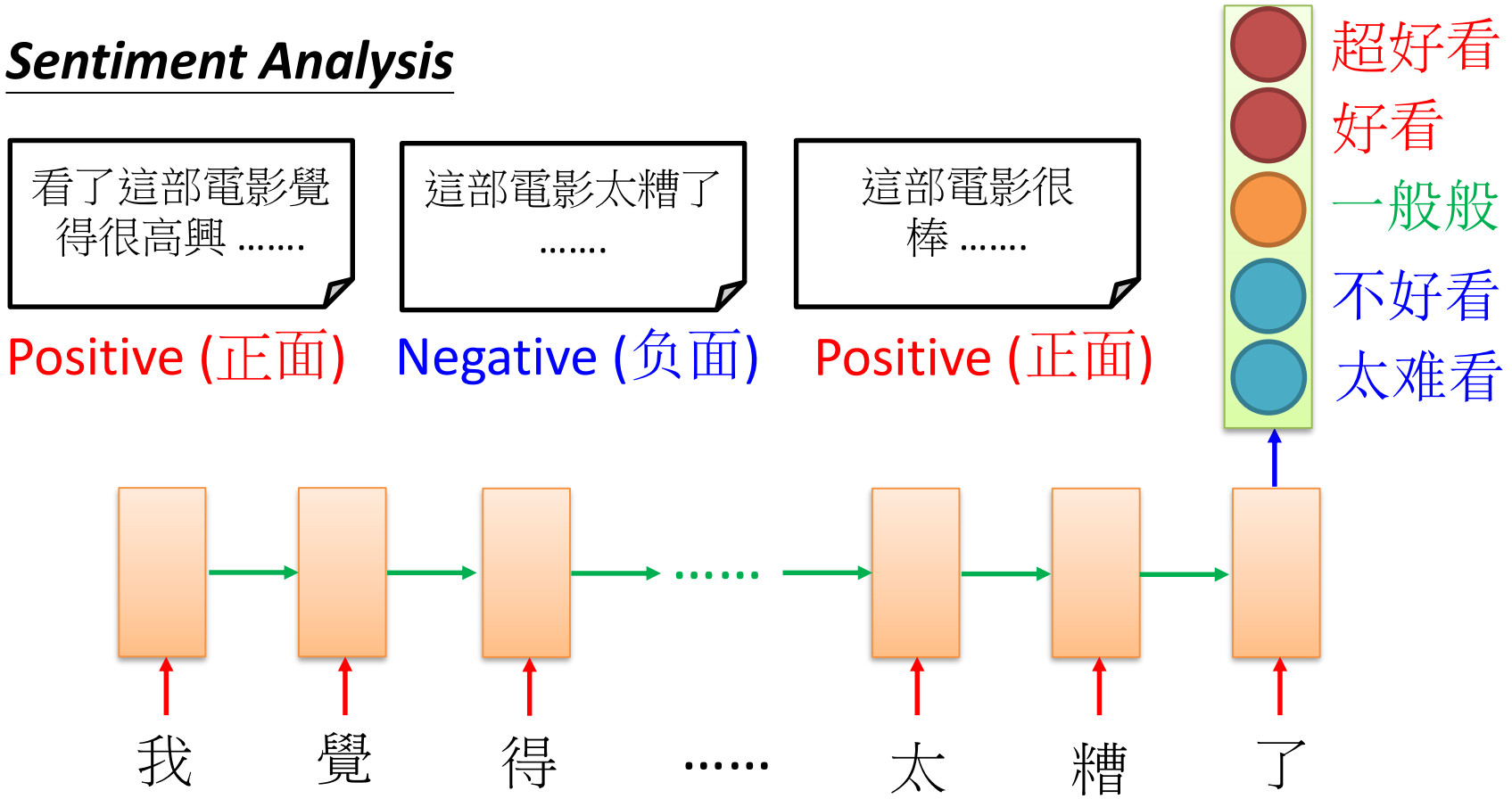
Probability of "Beijing" in each slot

Probability of "on" in each slot



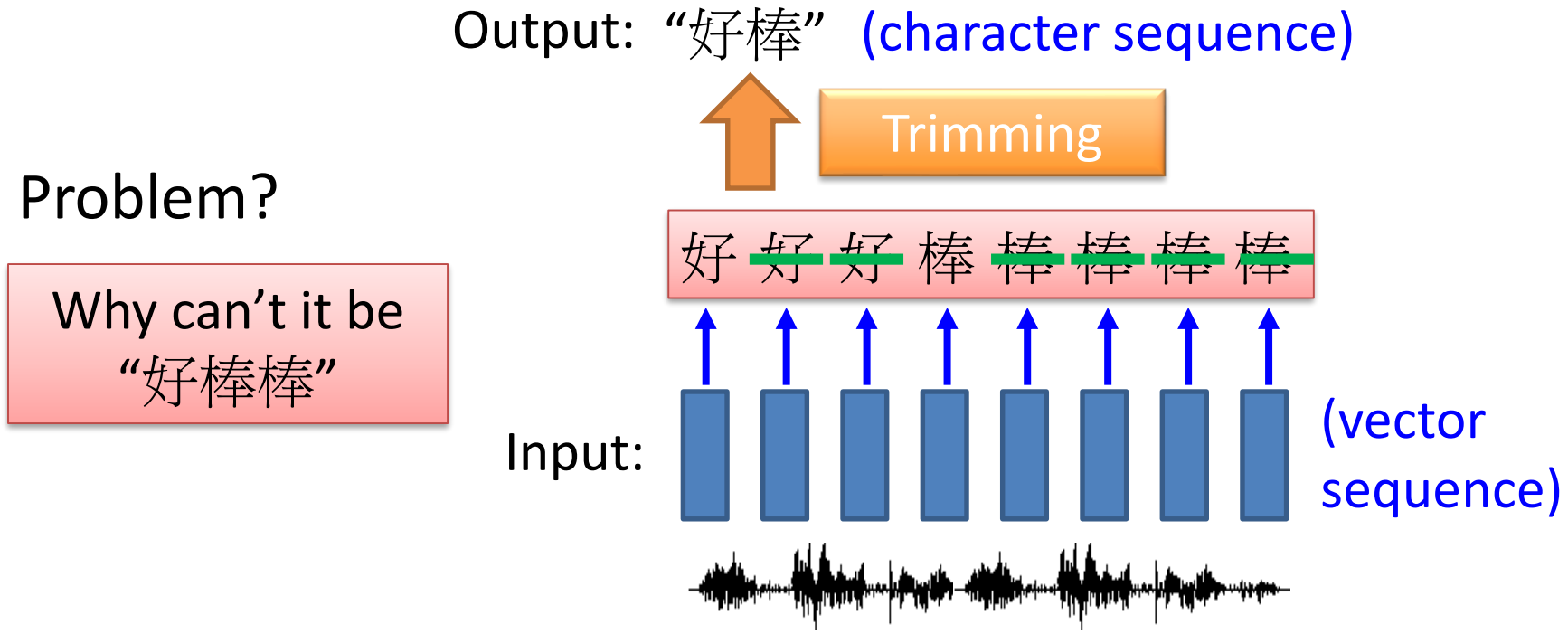
Many to one

- Input is a vector sequence, but output is only one vector



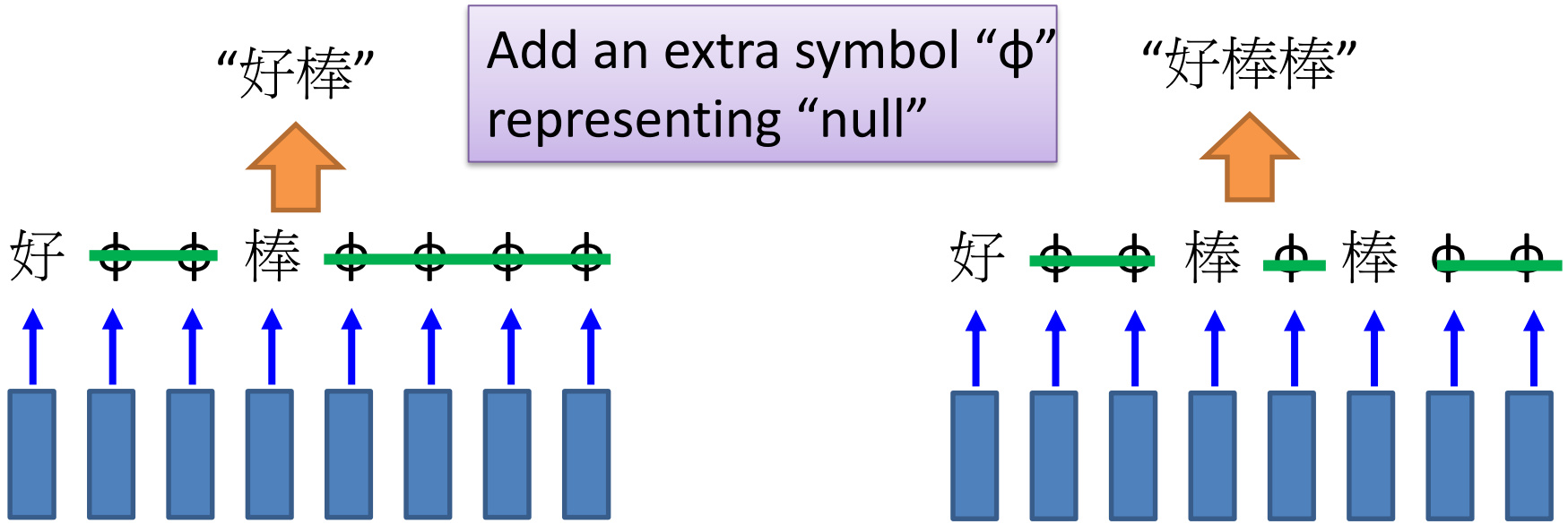
Many to Many (Output is shorter)

- Both input and output are both sequences, ***but the output is shorter.***
  - E.g. ***Speech Recognition***



Many to Many (Output is shorter)

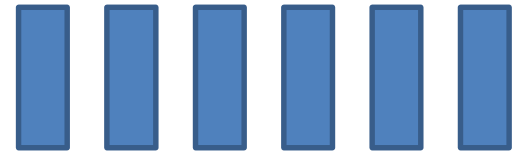
- Both input and output are both sequences, ***but the output is shorter.***
  - Connectionist Temporal Classification (CTC) [Alex Graves, ICML' 06][Alex Graves, ICML' 14][Haşim Sak, Interspeech' 15][Jie Li, Interspeech' 15][Andrew Senior, ASRU' 15]



Many to Many (Output is shorter)

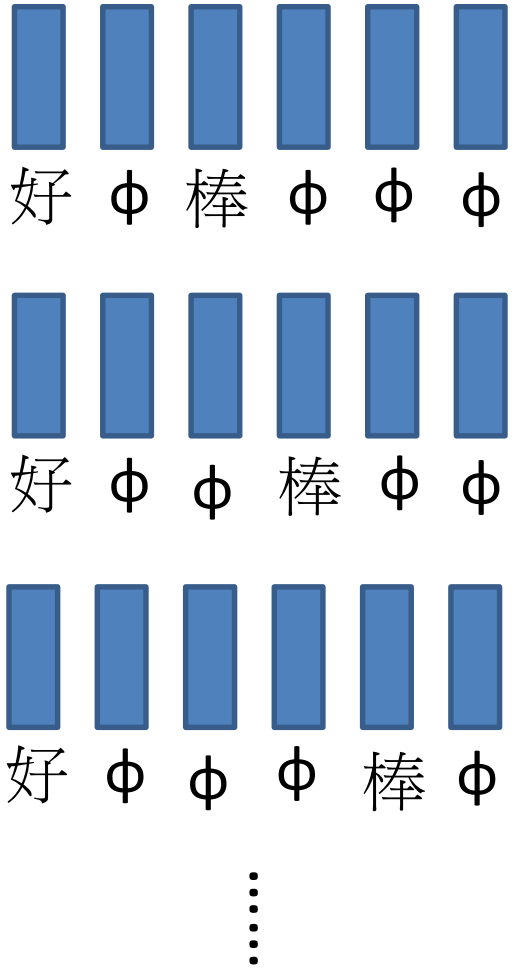
- CTC: Training

Acoustic Features:



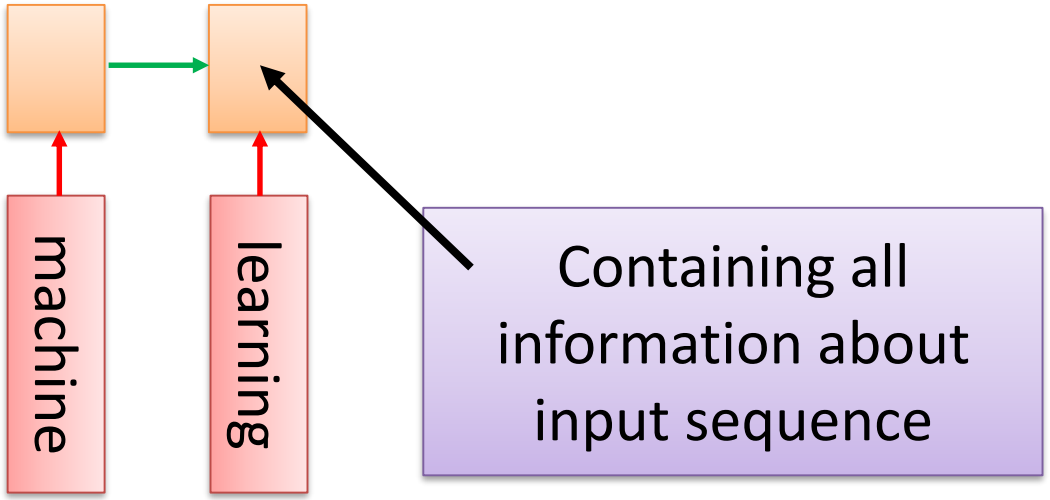
Label: 好 棒

All possible alignments are considered as correct.



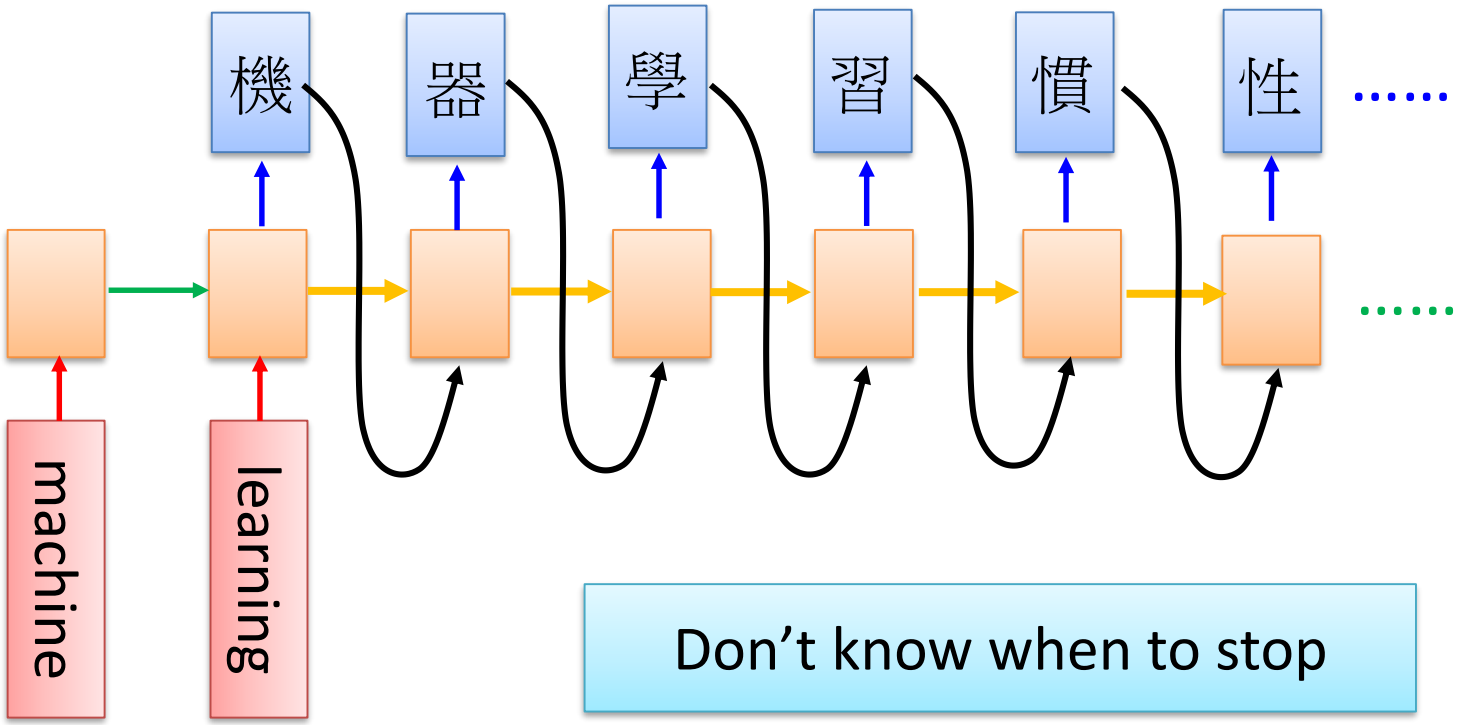
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



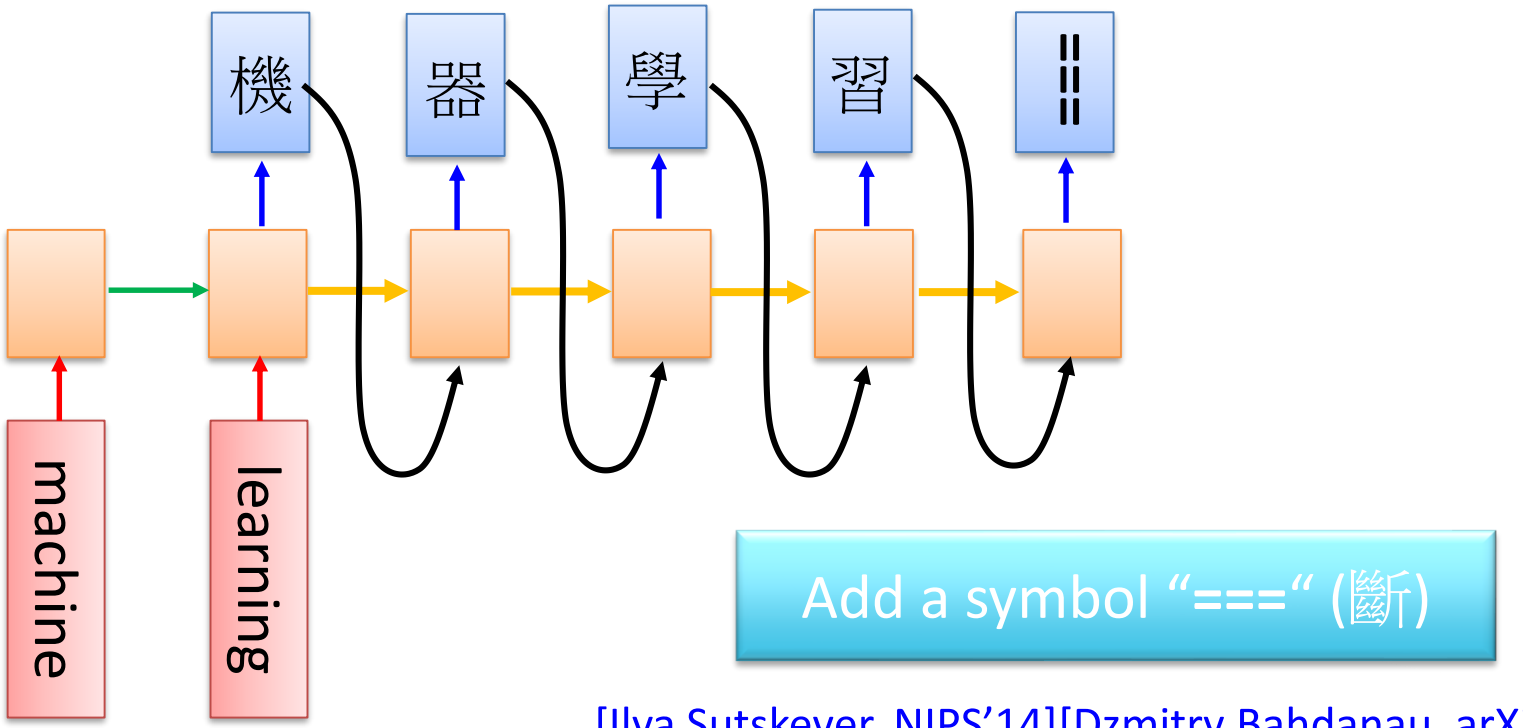
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***.
  → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



Many to Many (No Limitation)

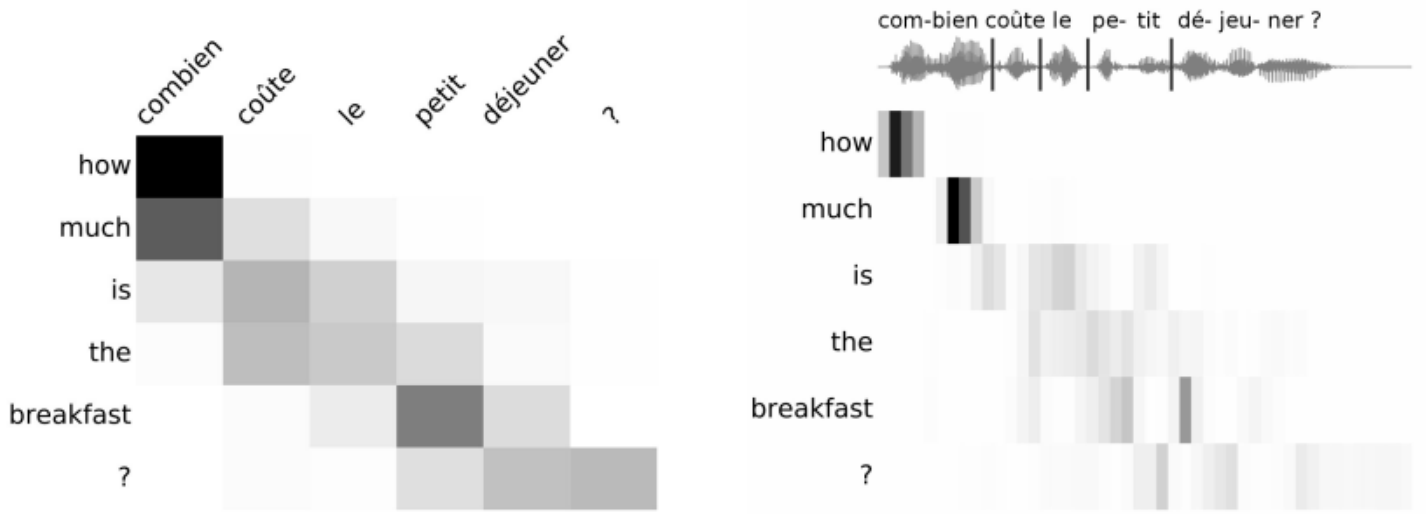
- Both input and output are both sequences ***with different lengths***.
  → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***.
  → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



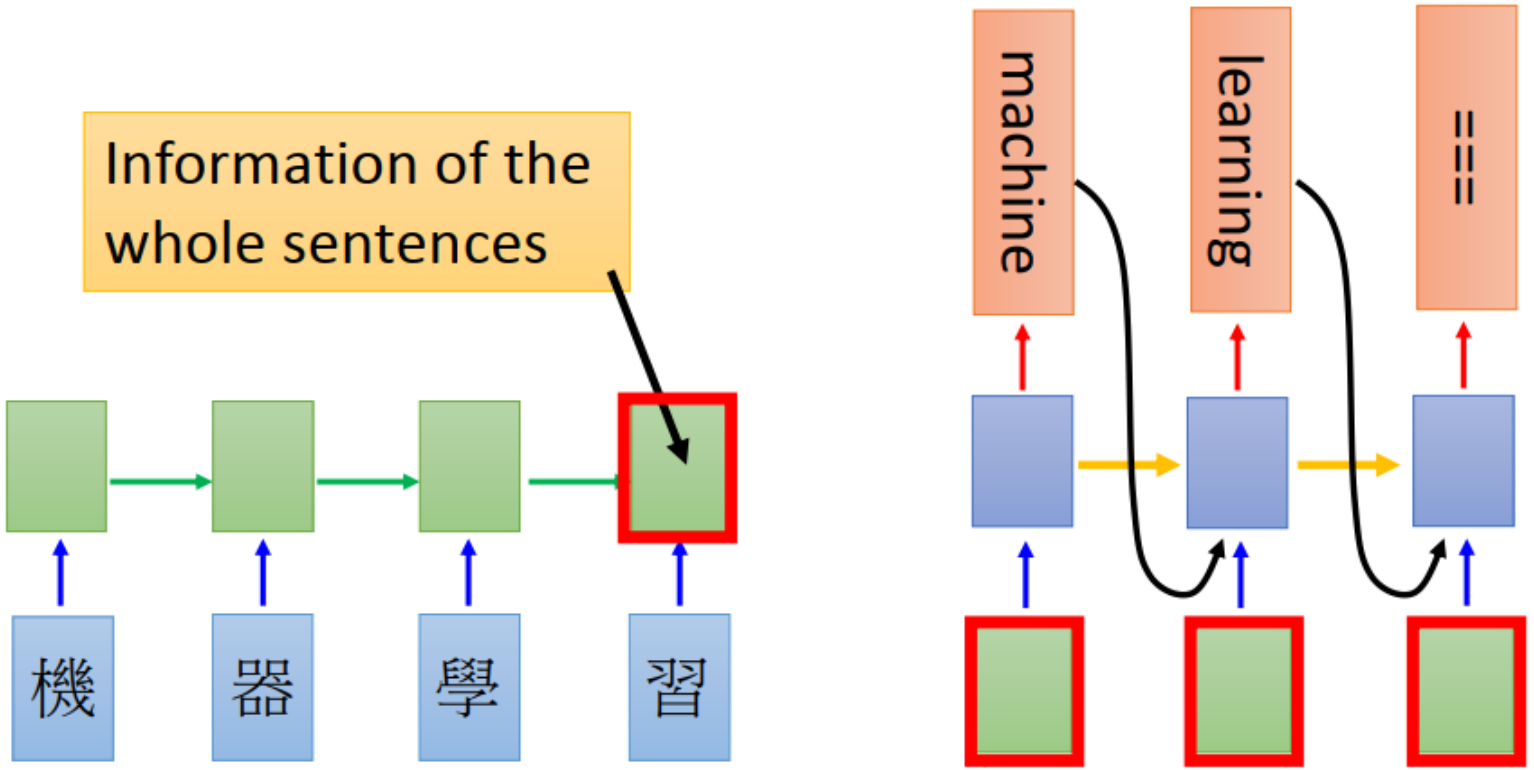
(a) Machine translation alignment (b) Speech translation alignment

Figure 1: Alignments performed by the attention model during training

https://arxiv.org/pdf/1612.01744v1.pdf

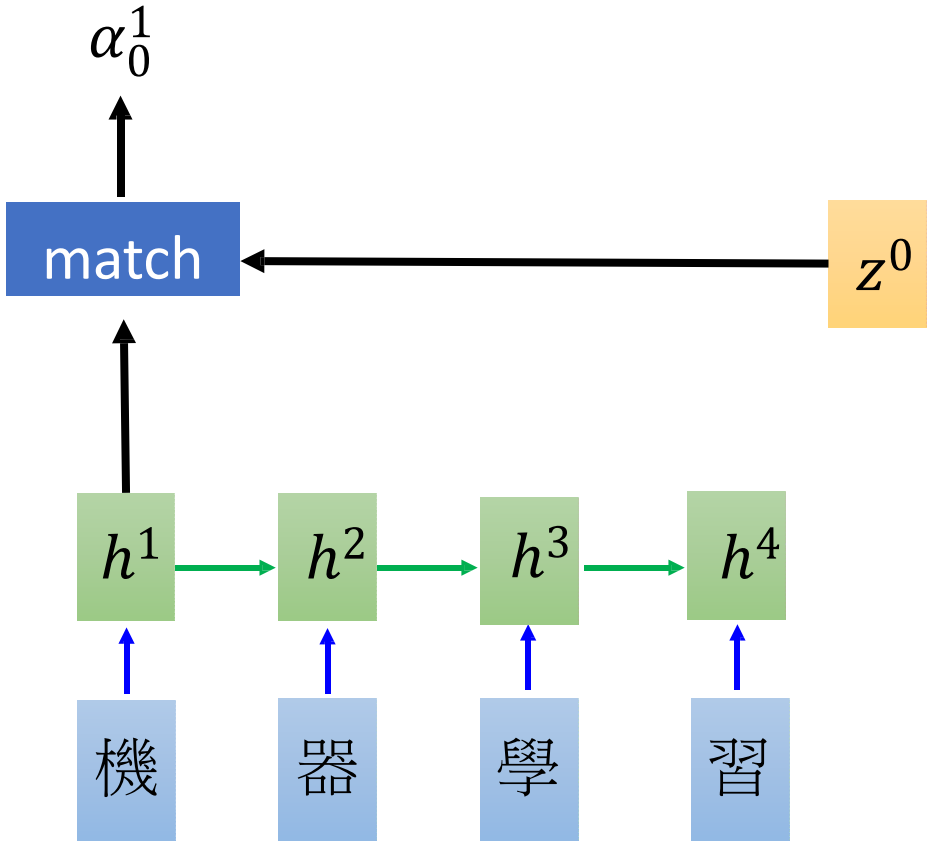
Attention

- Sequence to sequence learning: Both input and output are both sequences ***with different lengths***.
- E.g. 機器學習→machine learning

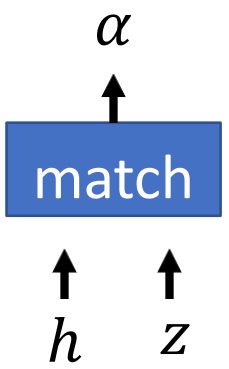


Machine Translation

- Attention-based model

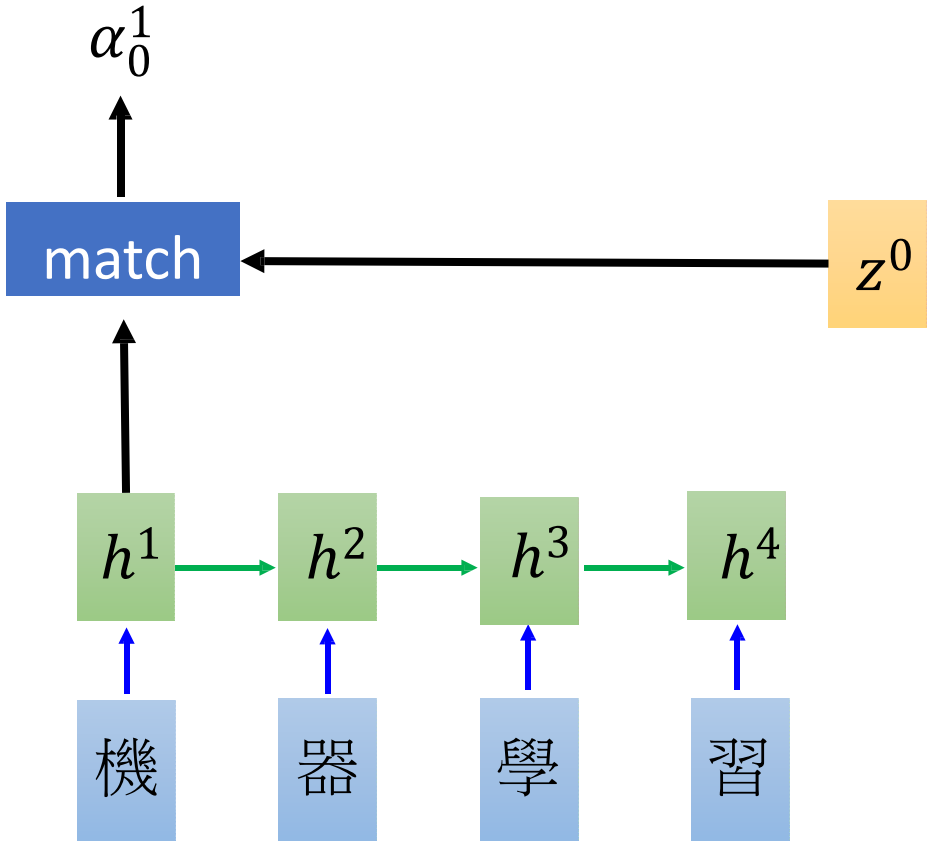


Jointly learned with other part of the network

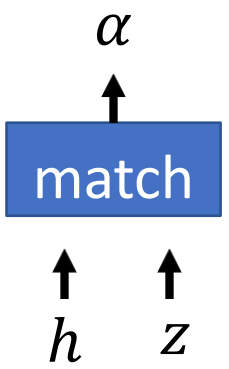


Machine Translation

- Attention-based model



Jointly learned with other part of the network



What is match ?

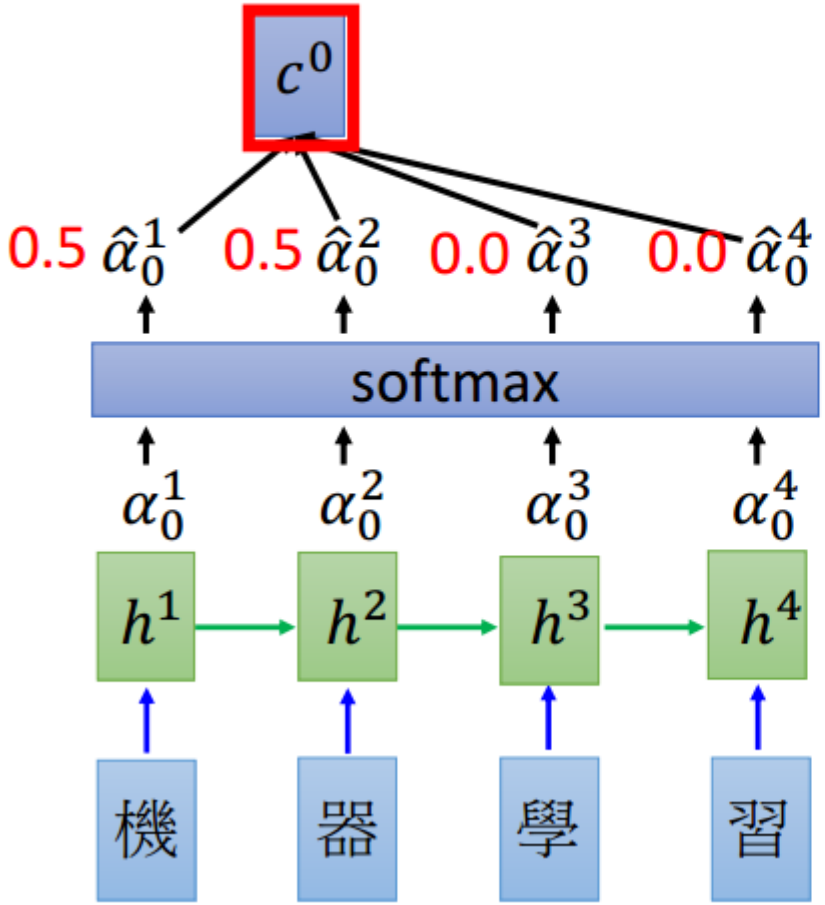
Design by yourself

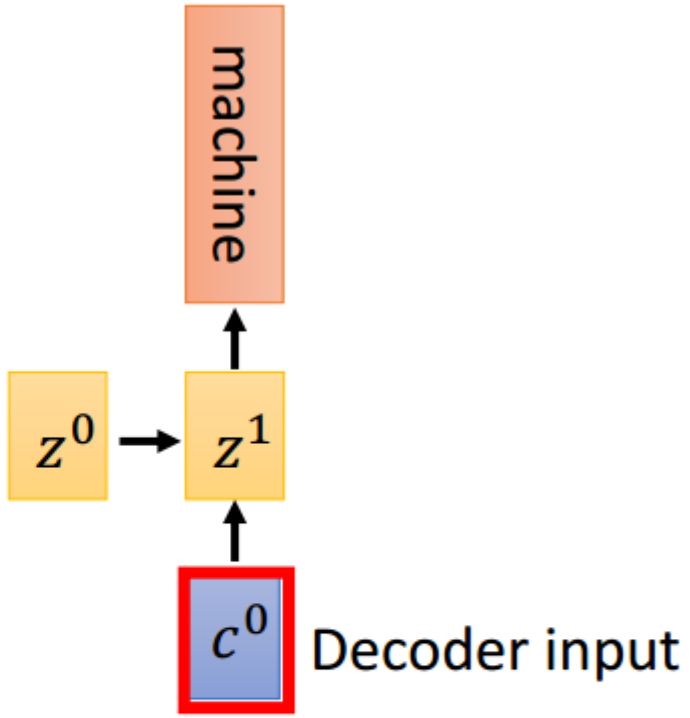
➢ Cosine similarity of z and h

➢ Small NN whose input is z and h, output a scalar

Machine Translation

- Attention-based model



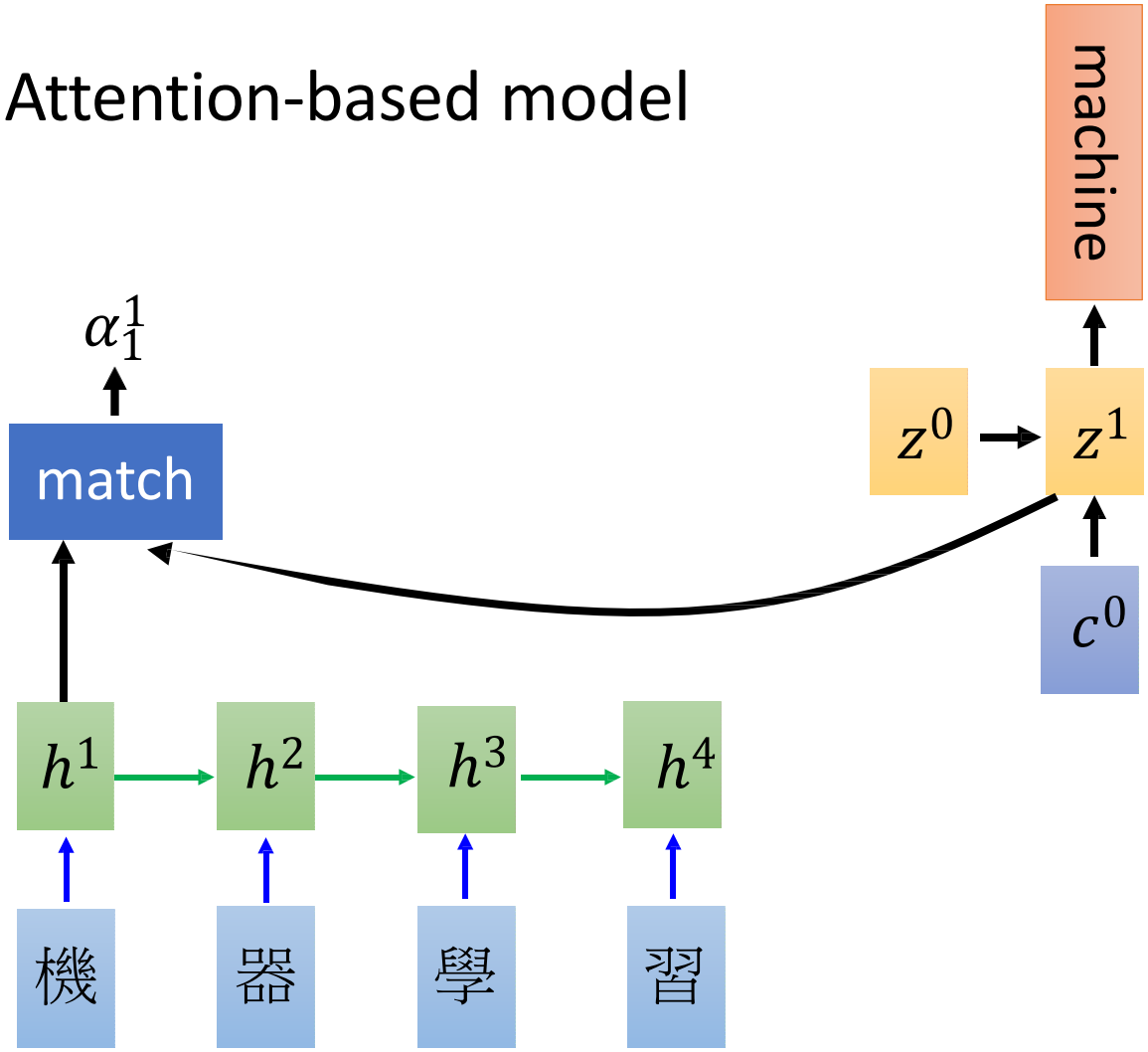


$$c^0 = \sum \hat{\alpha}_0^i h^i$$

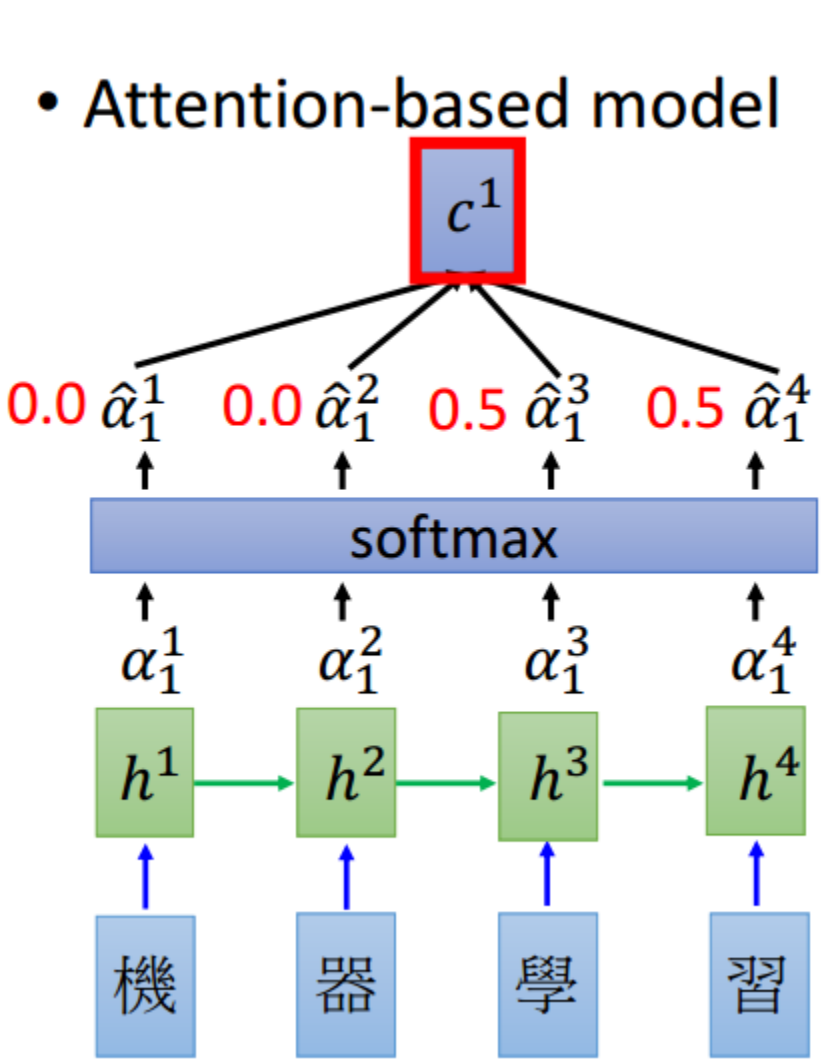
$$= 0.5h^1 + 0.5h^2$$

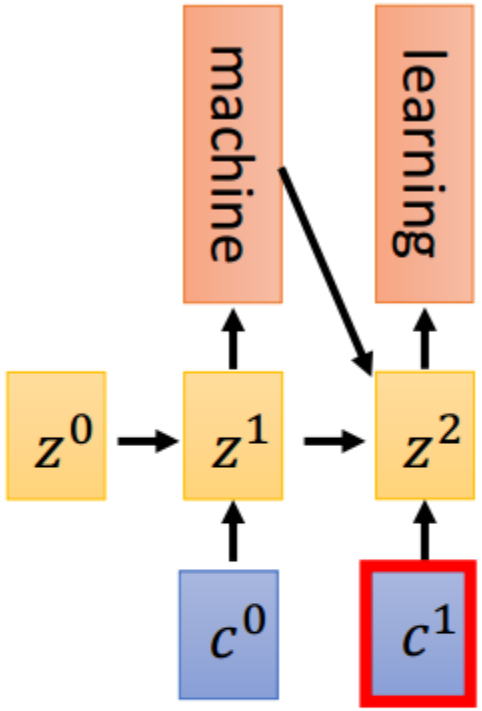
Machine Translation

- Attention-based model



Machine Translation





$$c^1 = \sum \hat{\alpha}_1^i h^i$$

$$= 0.5h^3 + 0.5h^4$$

Machine Translation

- Attention-based model

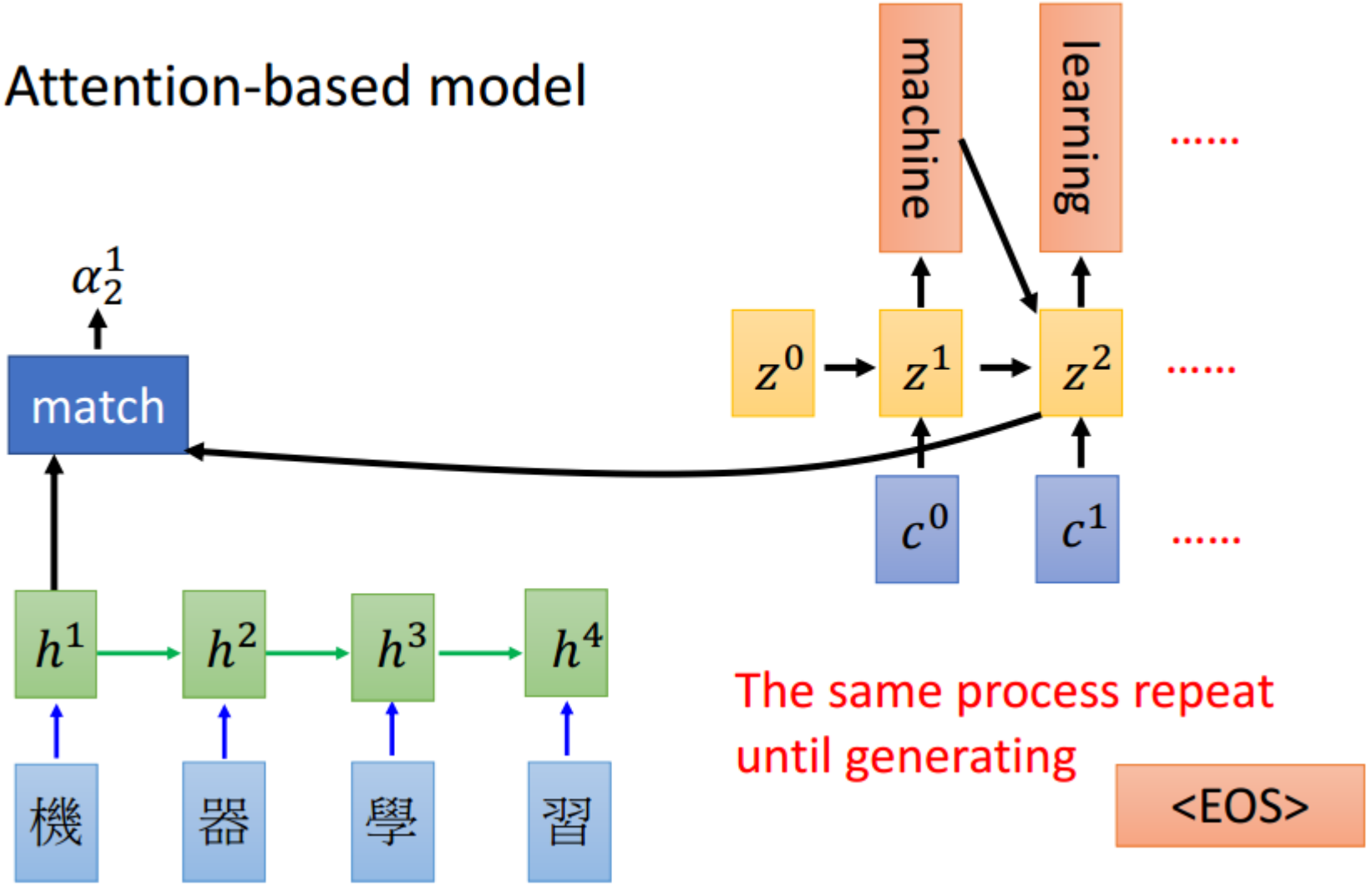


Image Caption Generation

- Input an image, but output a sequence of words

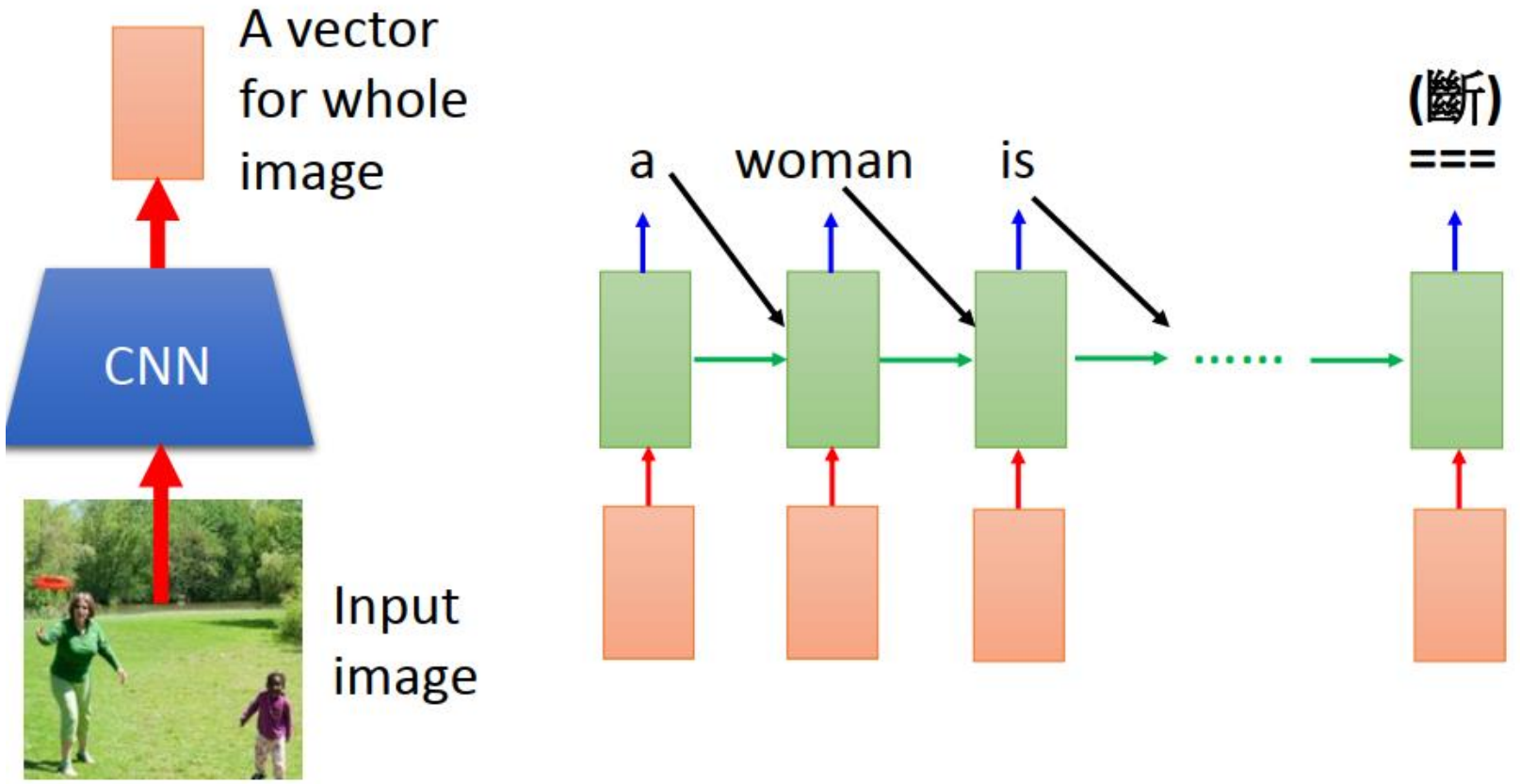


Image Caption Generation

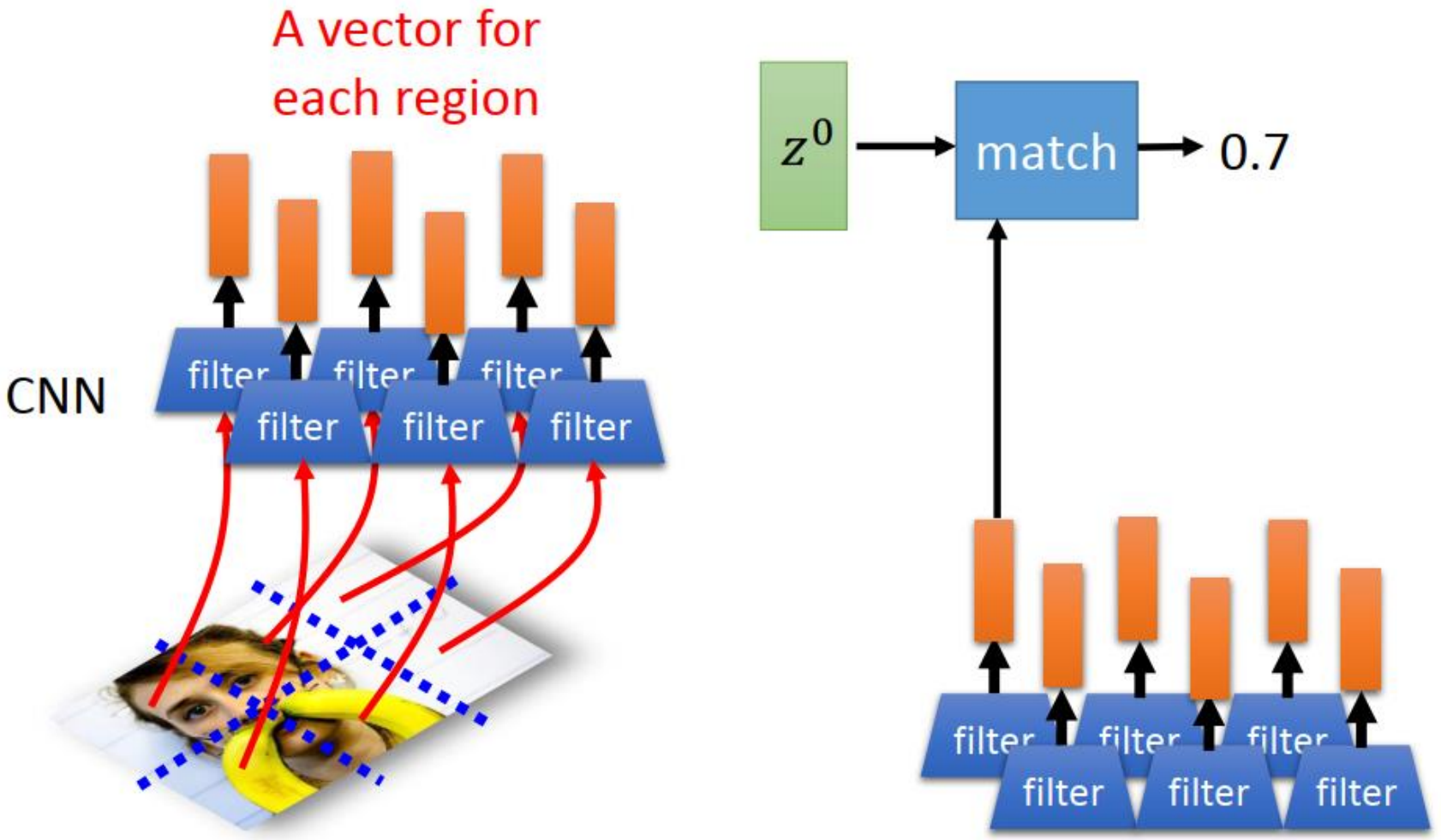


Image Caption Generation

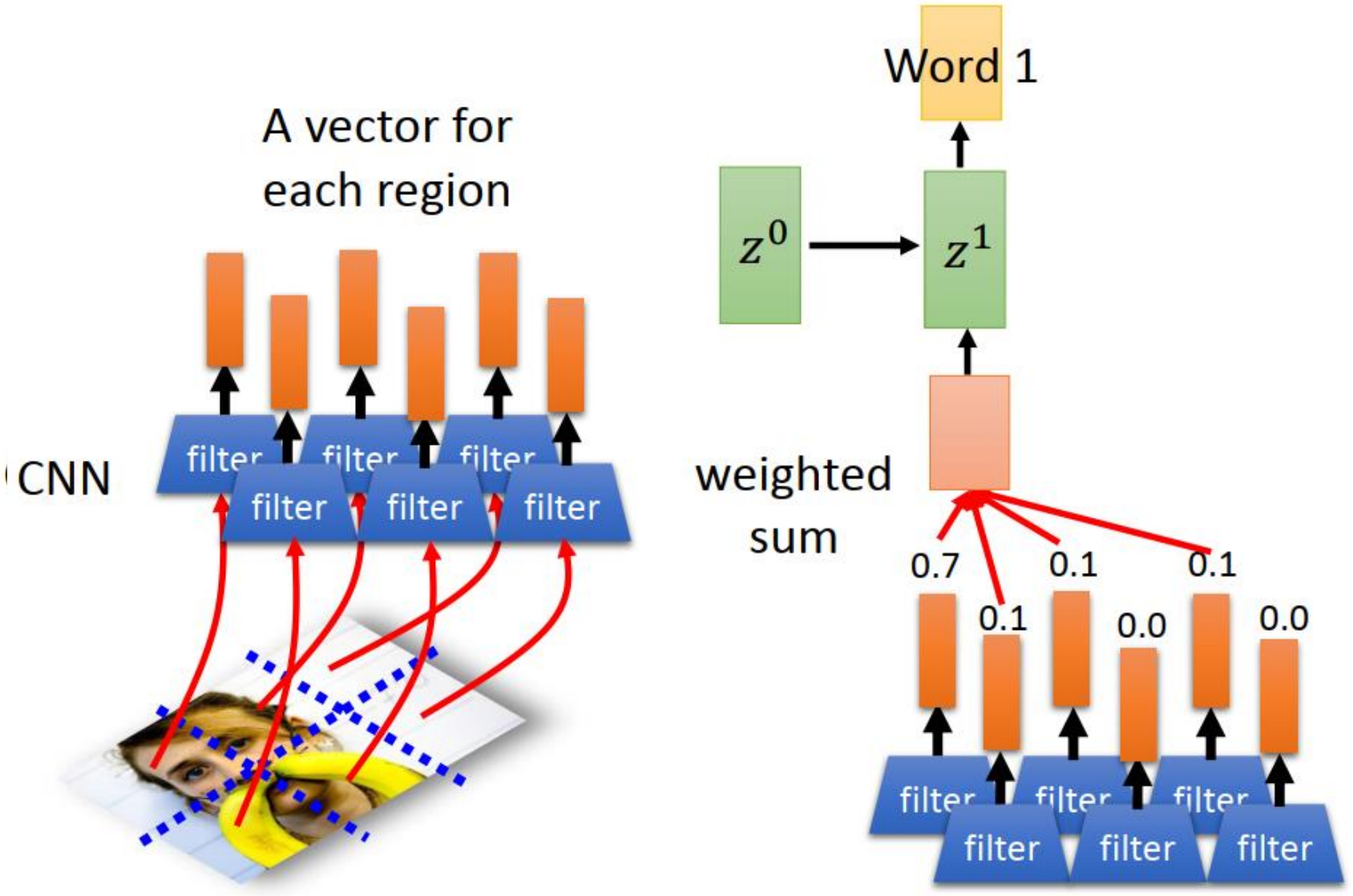
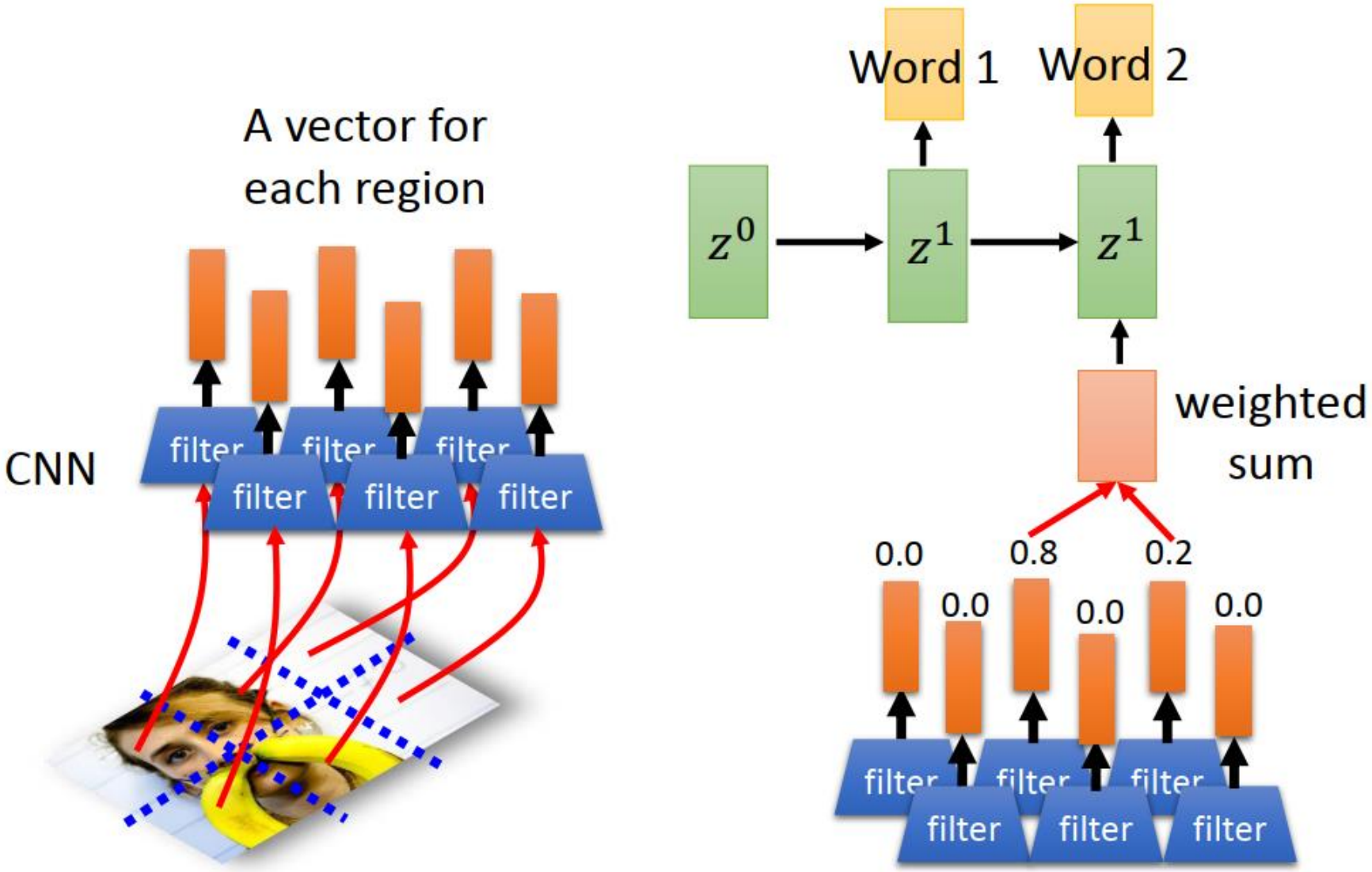


Image Caption Generation



- Good captions

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

- Bad captions

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Ref: A man and a woman ride a motorcycle

A **man** and a **woman** are **talking** on the **road**

Ref: A woman is frying food

Someone is **frying** a **fish** in a **pot**