# Recurrent Neural Networks

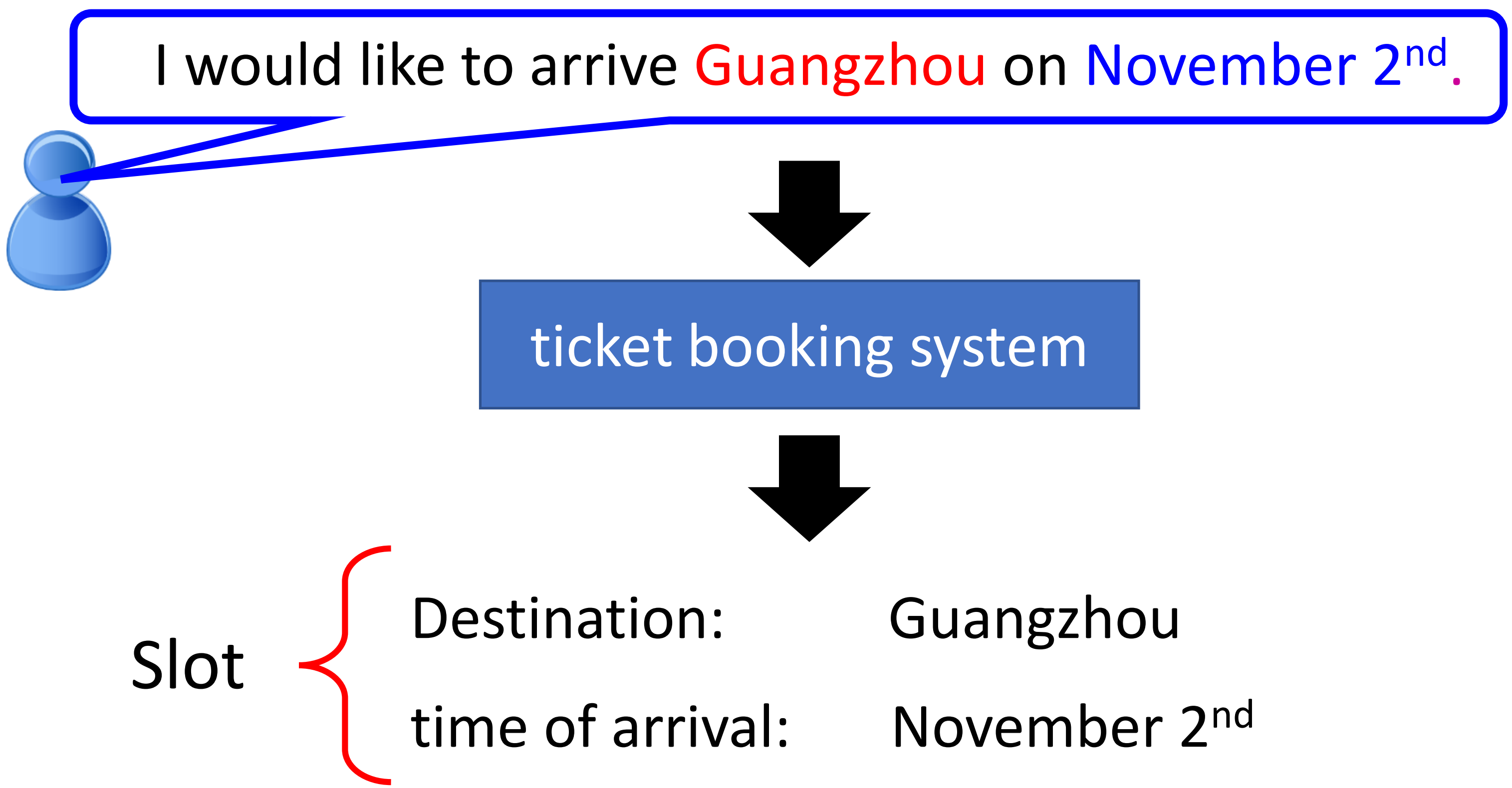
Hankui Zhuo

May 22, 2019

http://xplan-lab.org/IAA-Course

# Example Application

- Slot Filling

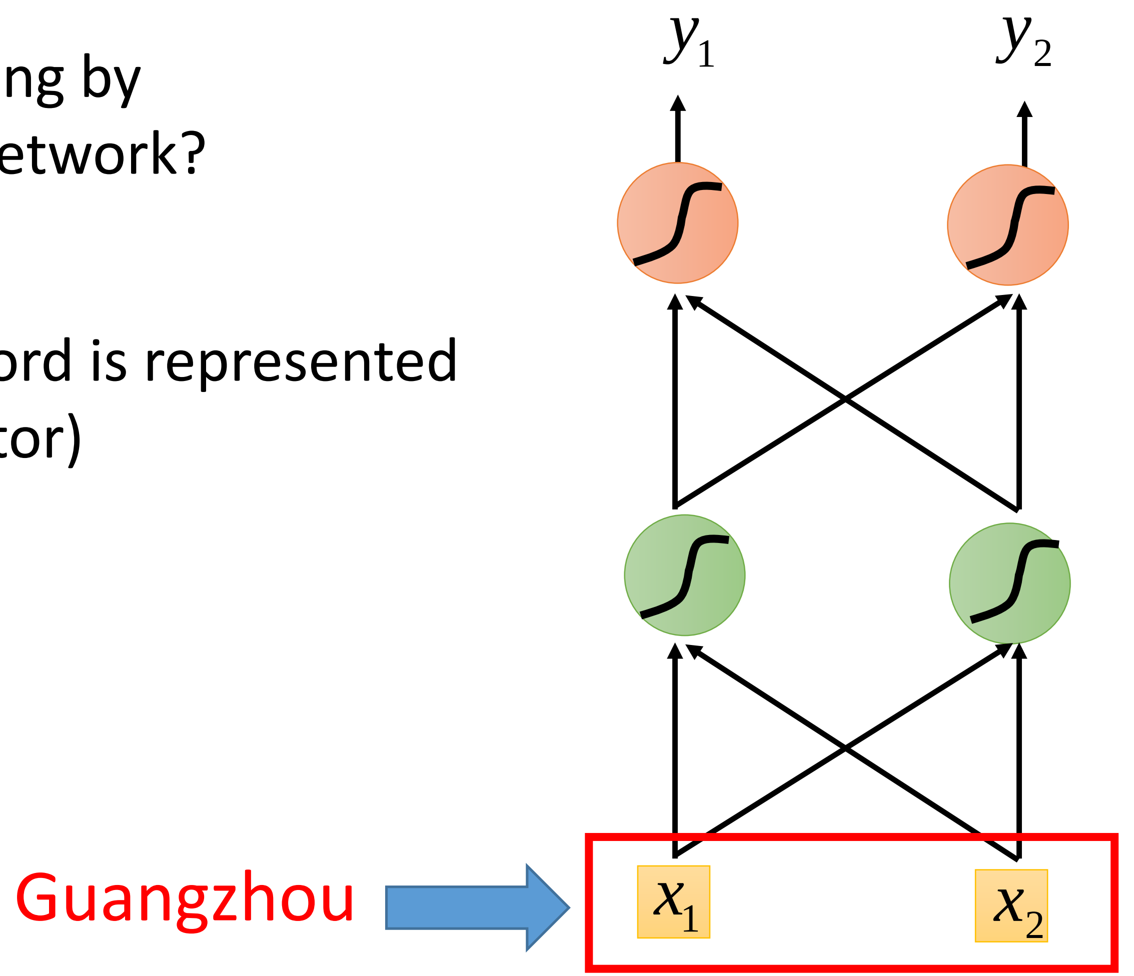


# Example Application

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]

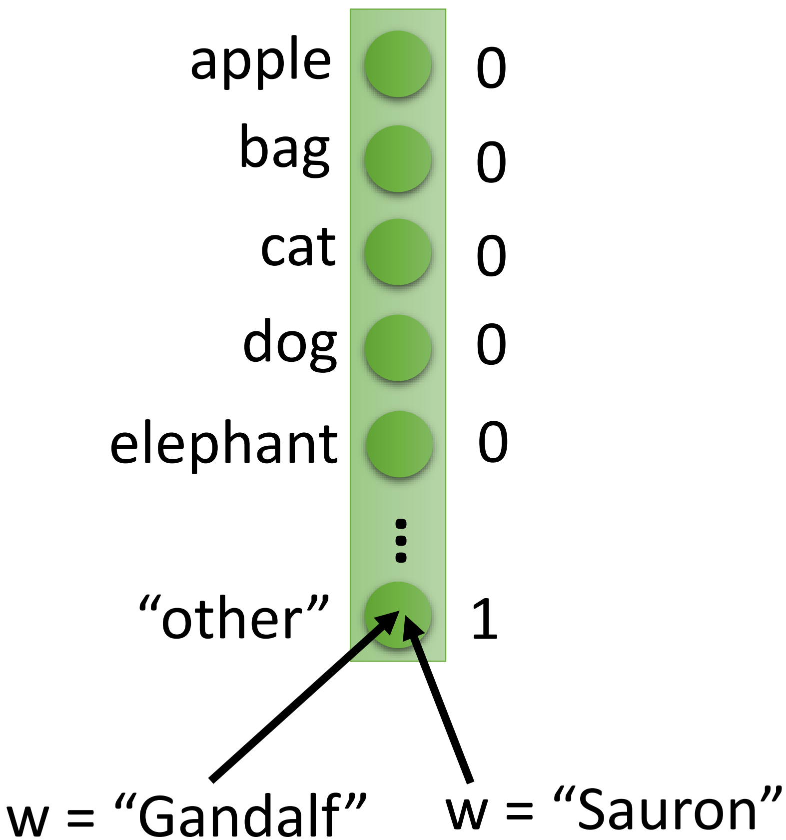
cat = [ 0 0 1 0 0]

dog = [ 0 0 0 1 0]

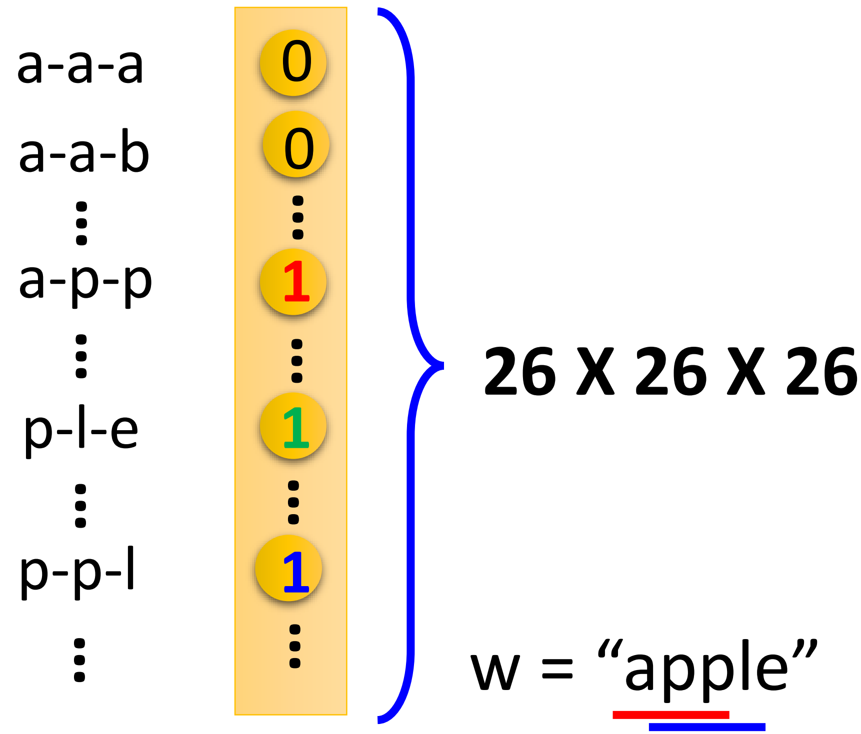
elephant = [ 0 0 0 0 1]

# Beyond 1-of-N encoding

***Dimension for "Other"***




***Word hashing***



# Example Application

Solving slot filling by Feedforward network?

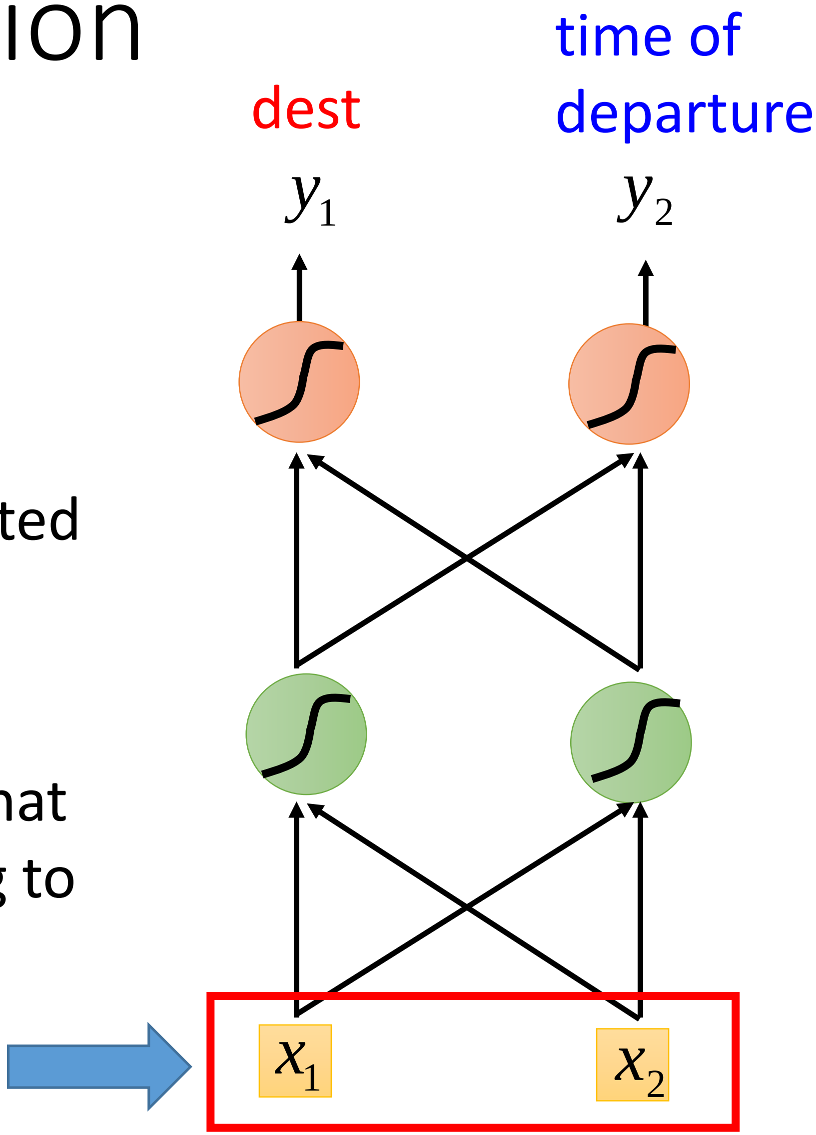
Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots

Guangzhou



# Example Application

arrive GZ on November $2^{nd}$

other dest other time time

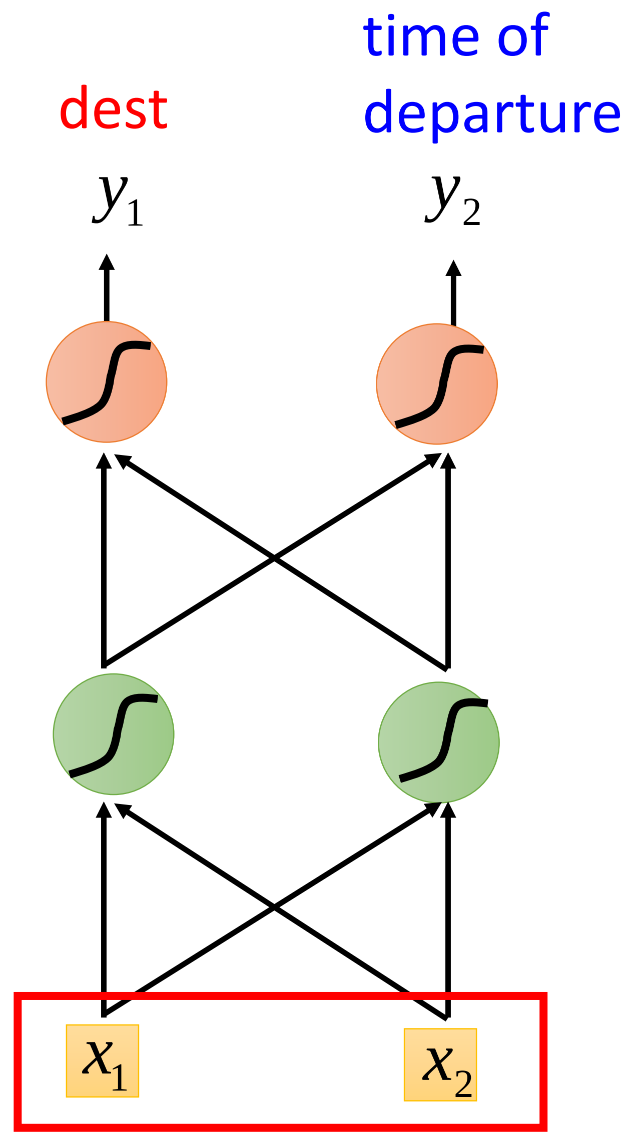
Problem?

leave GZ on November $2^{nd}$

place of departure

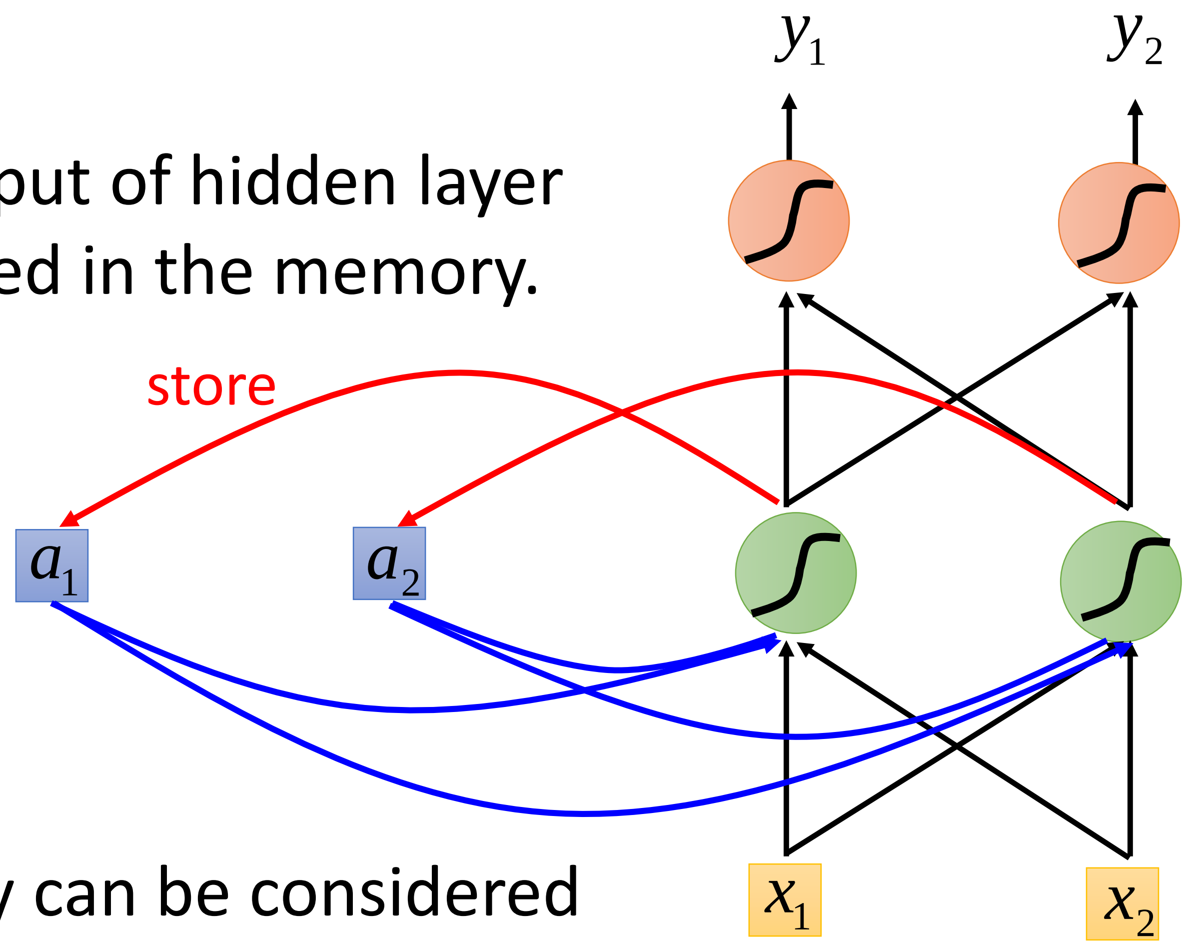
Neural network needs memory!

angzhou



# Recurrent Neural Networks (RNNs)

The output of hidden layer are stored in the memory.

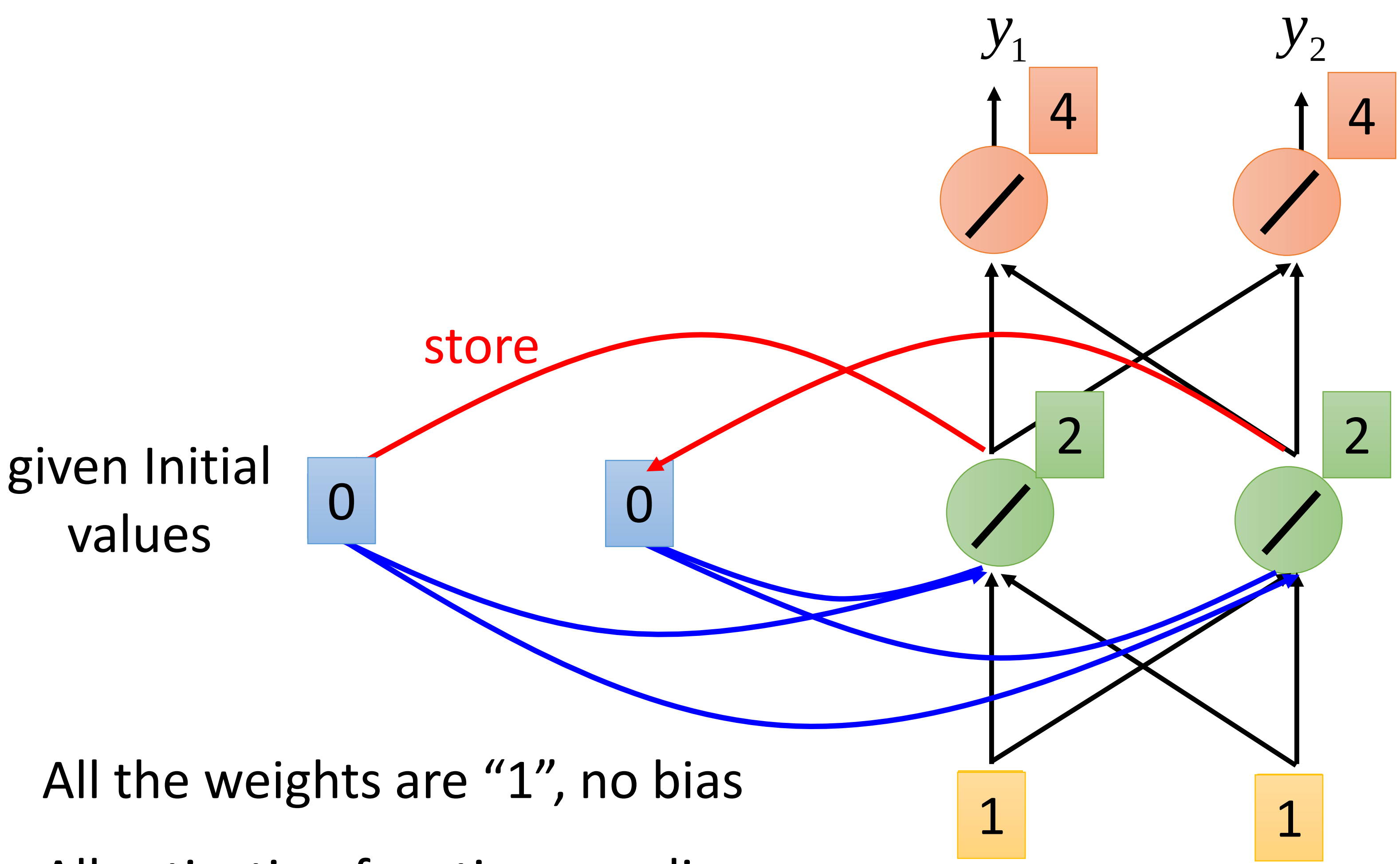



Memory can be considered as another input.

# Example

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

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



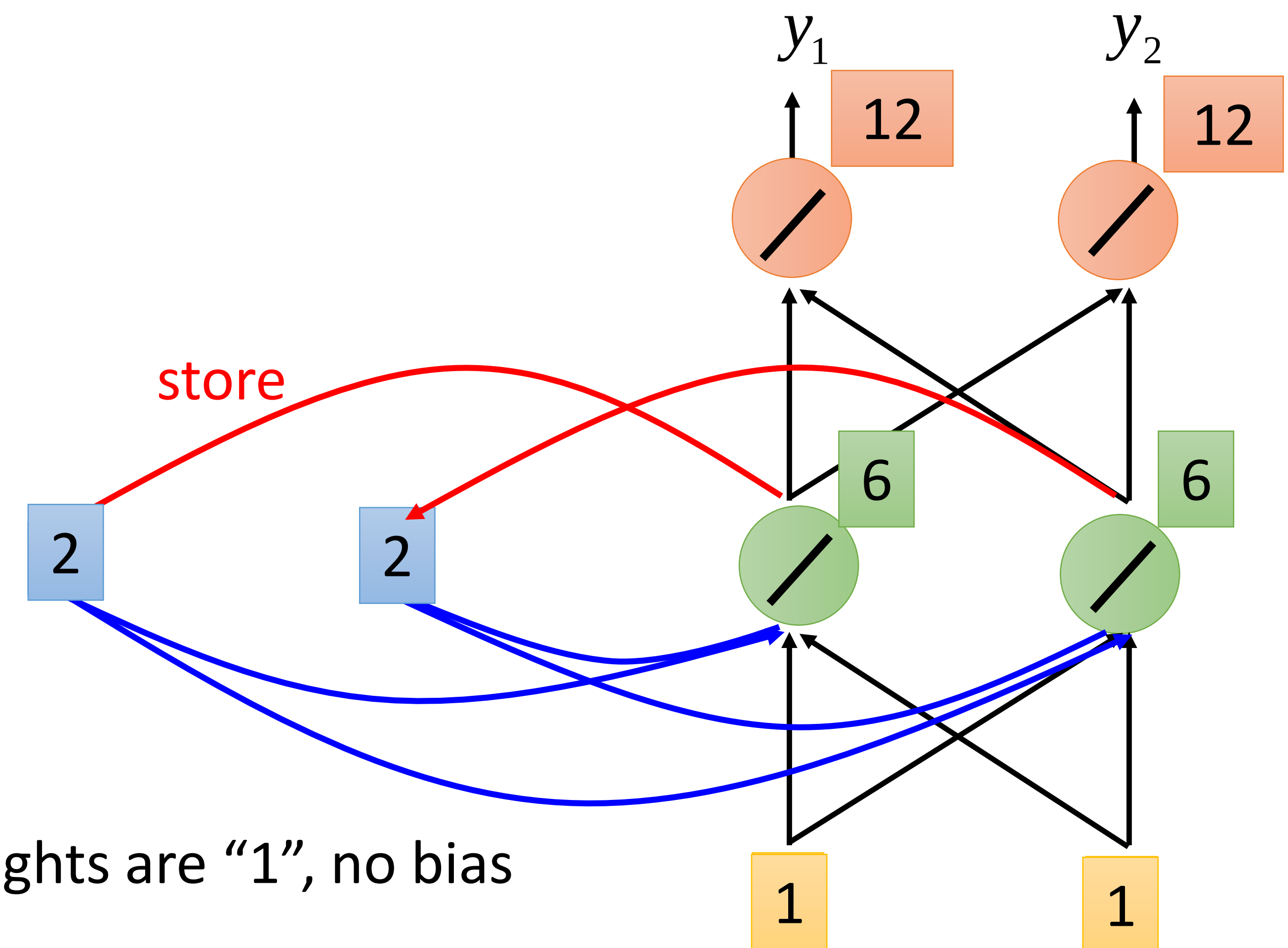

All the weights are "1", no bias

All activation functions are linear

# Example

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

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




All the weights are "1", no bias

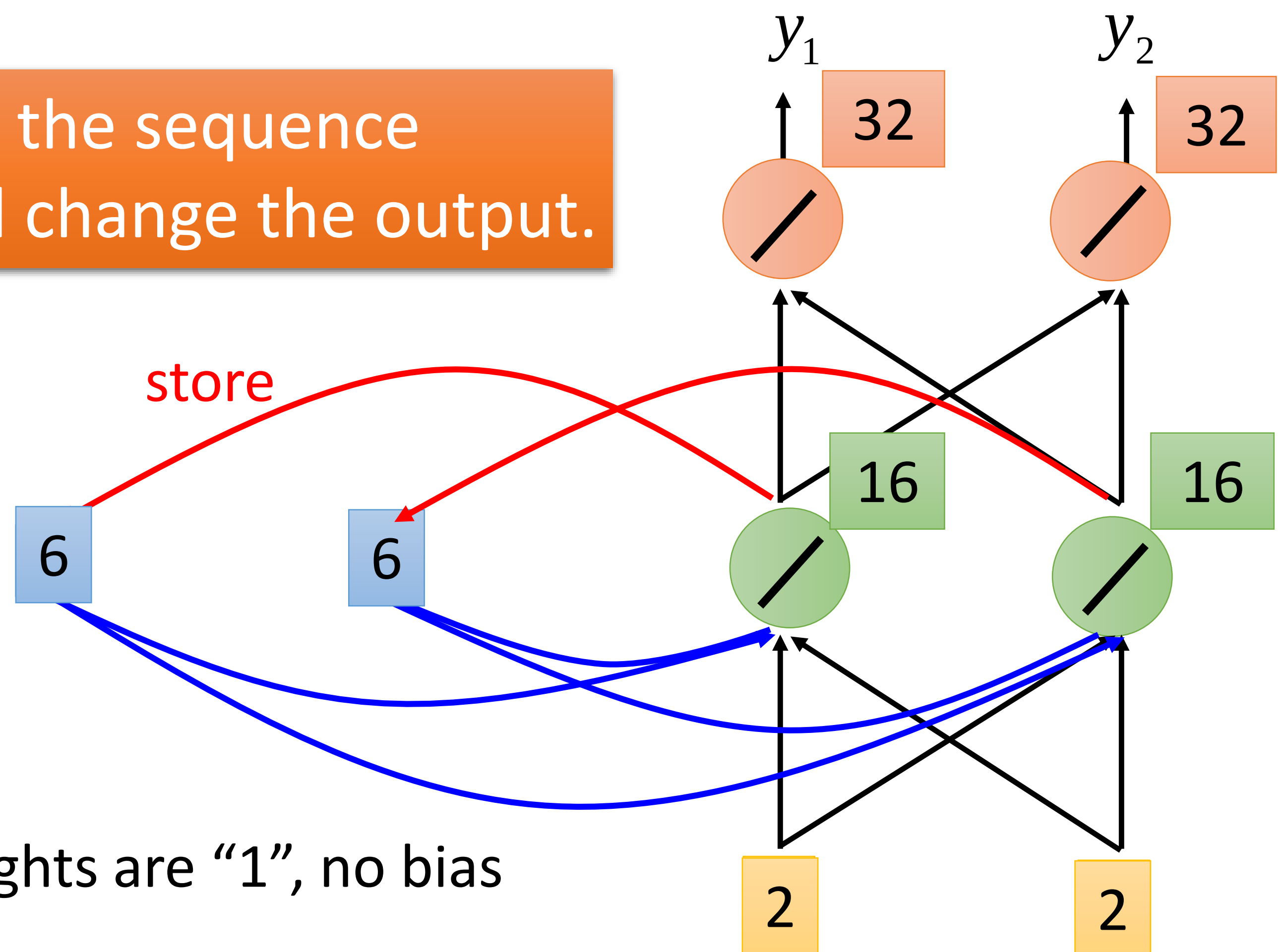
All activation functions are linear

# Example

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

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.

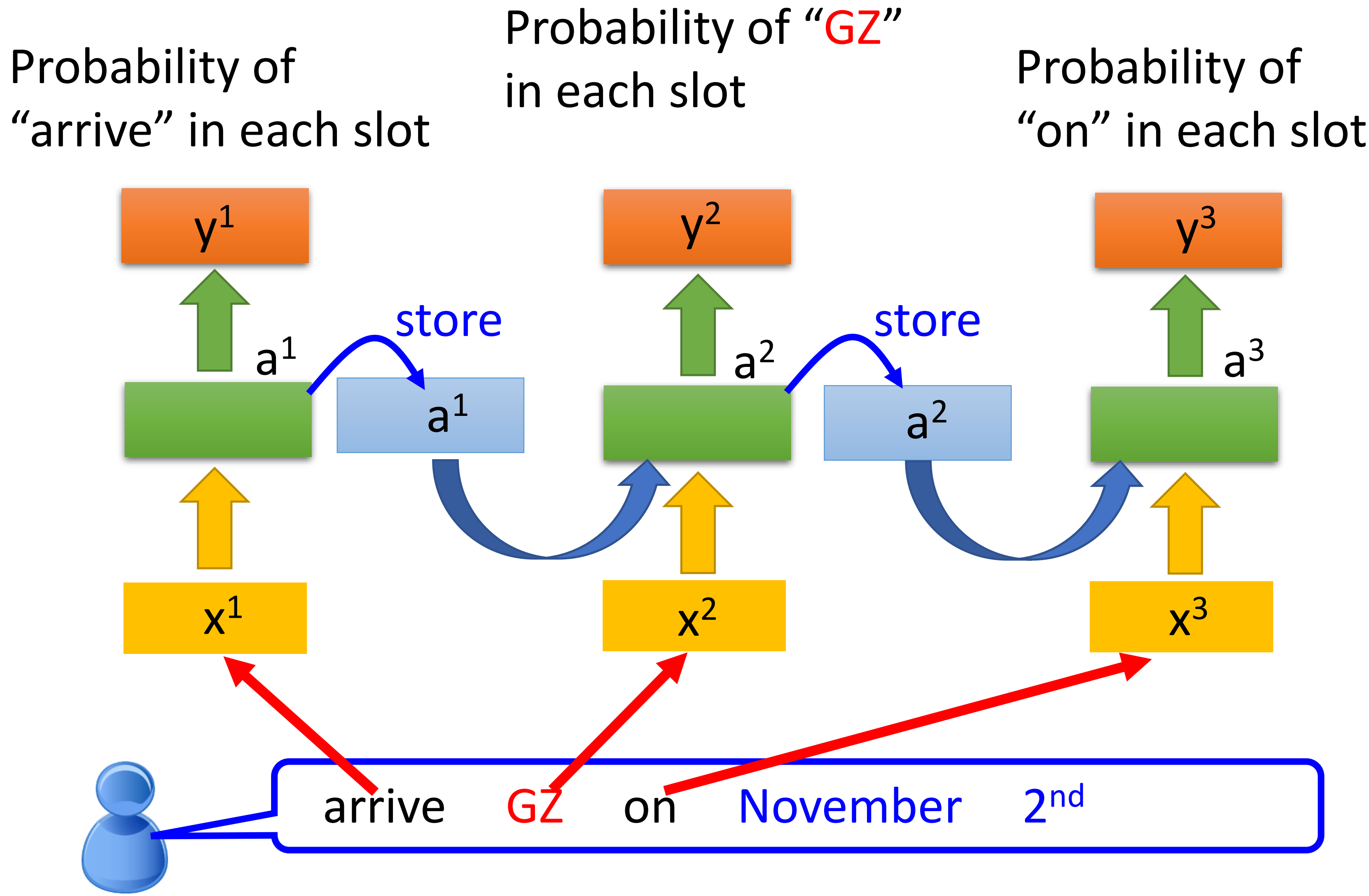



All the weights are "1", no bias

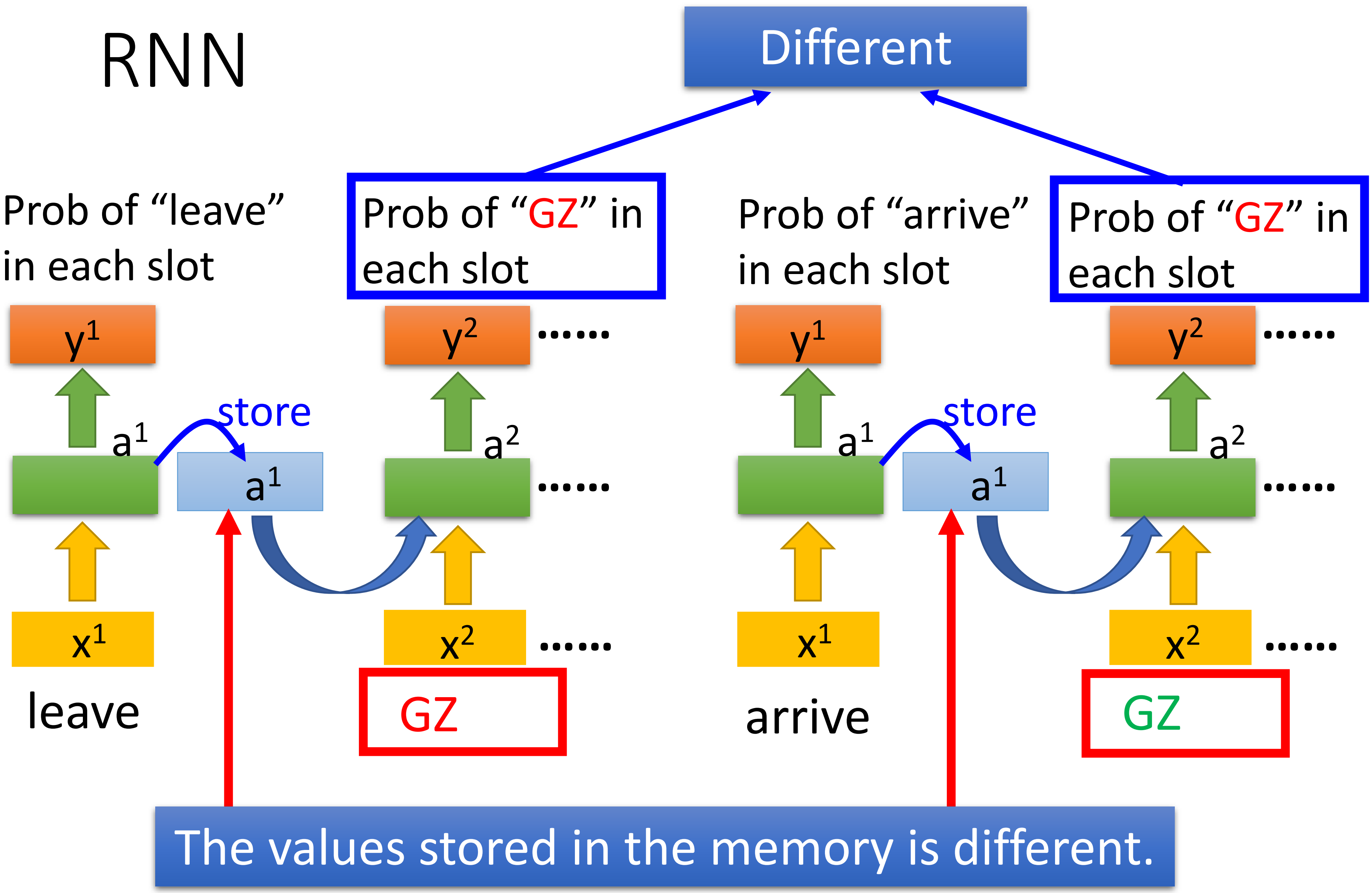
All activation functions are linear

# RNN

The same network is used again and again.

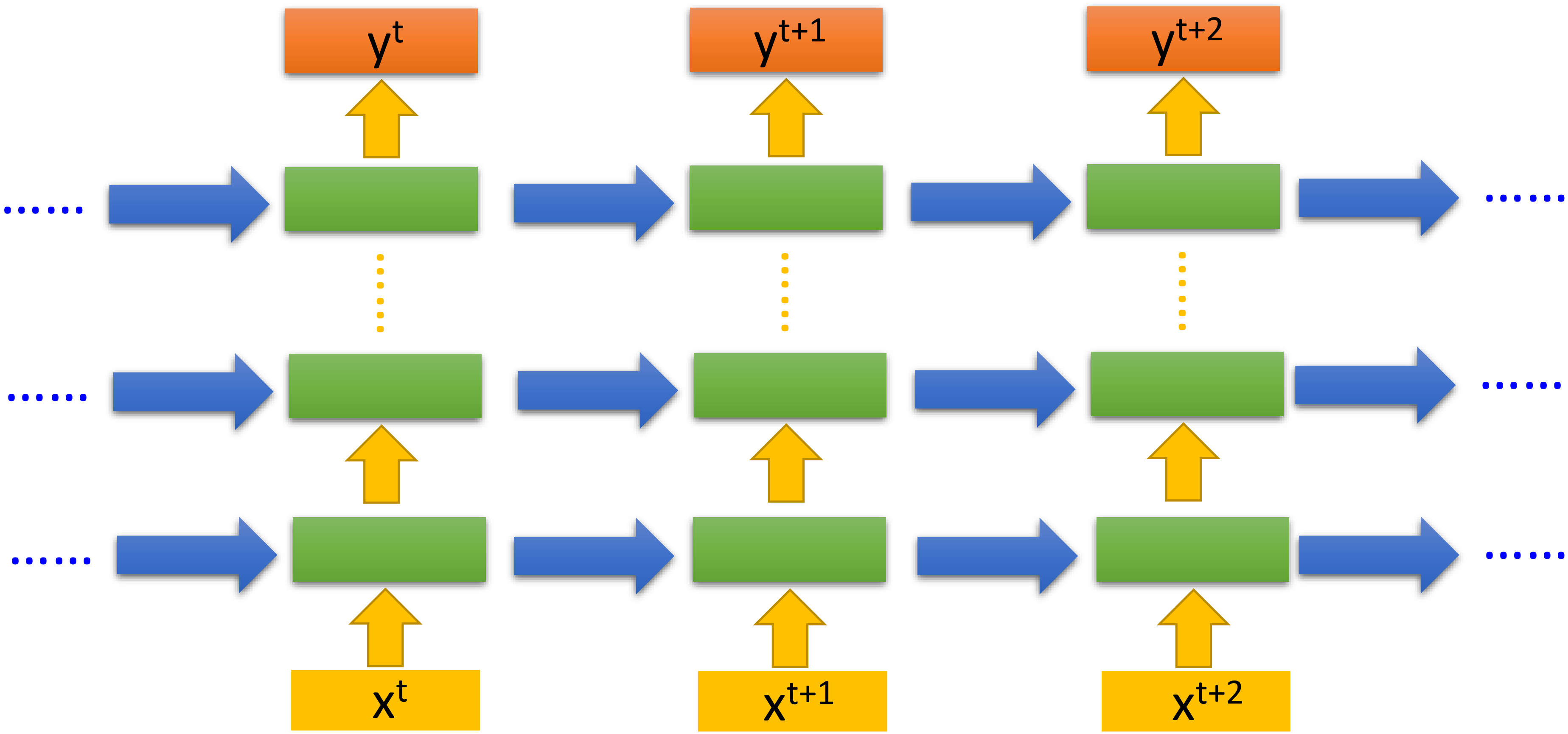


# RNN

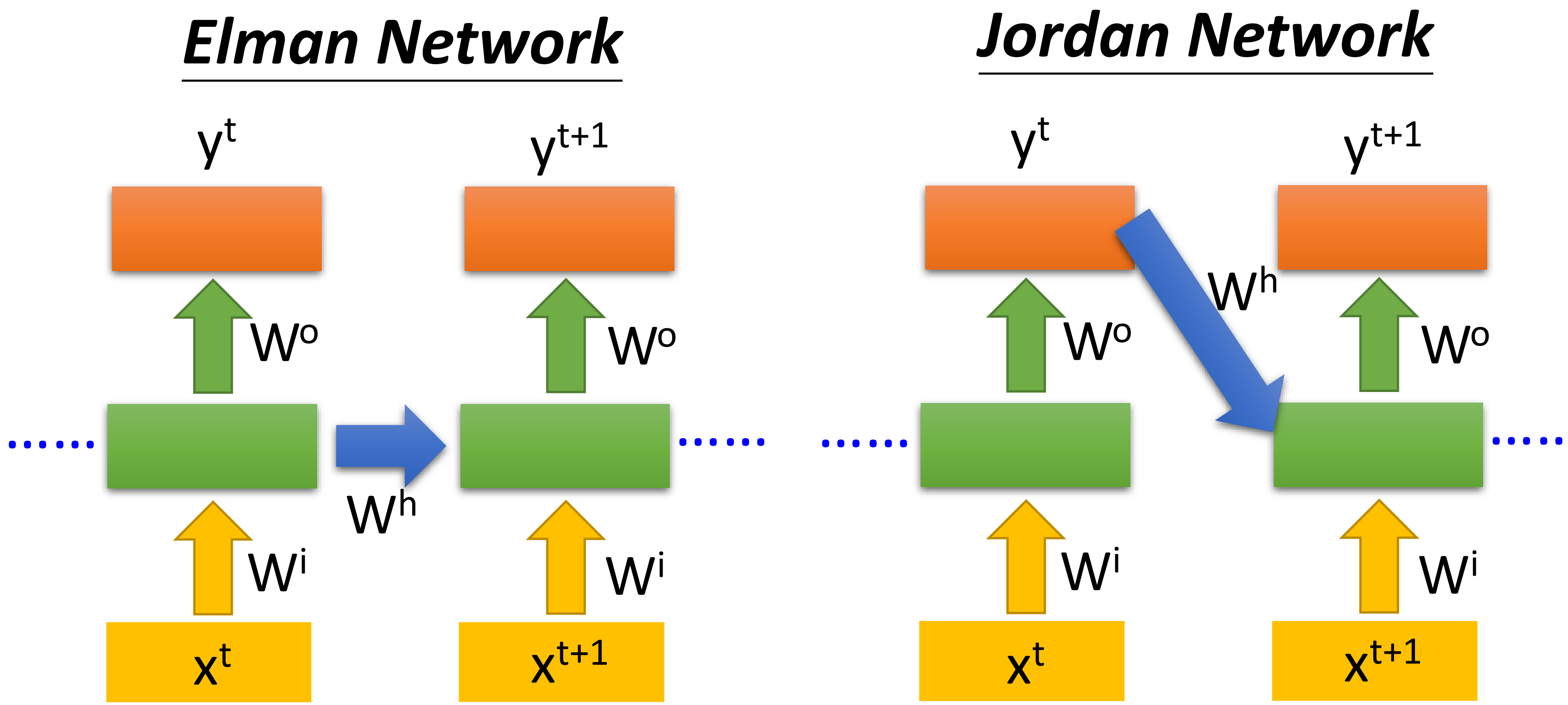


Different
Prob of "leave" in each slot
Prob of "GZ" in each slot
Prob of "arrive" in each slot
Prob of "GZ" in each slot
y1
y2
......
y1
y2
......
a1
store
a1
a2
......
a1
store
a1
a2
......
x1
x2
......
x1
x2
......
leave
GZ
arrive
GZ
The values stored in the memory is different.

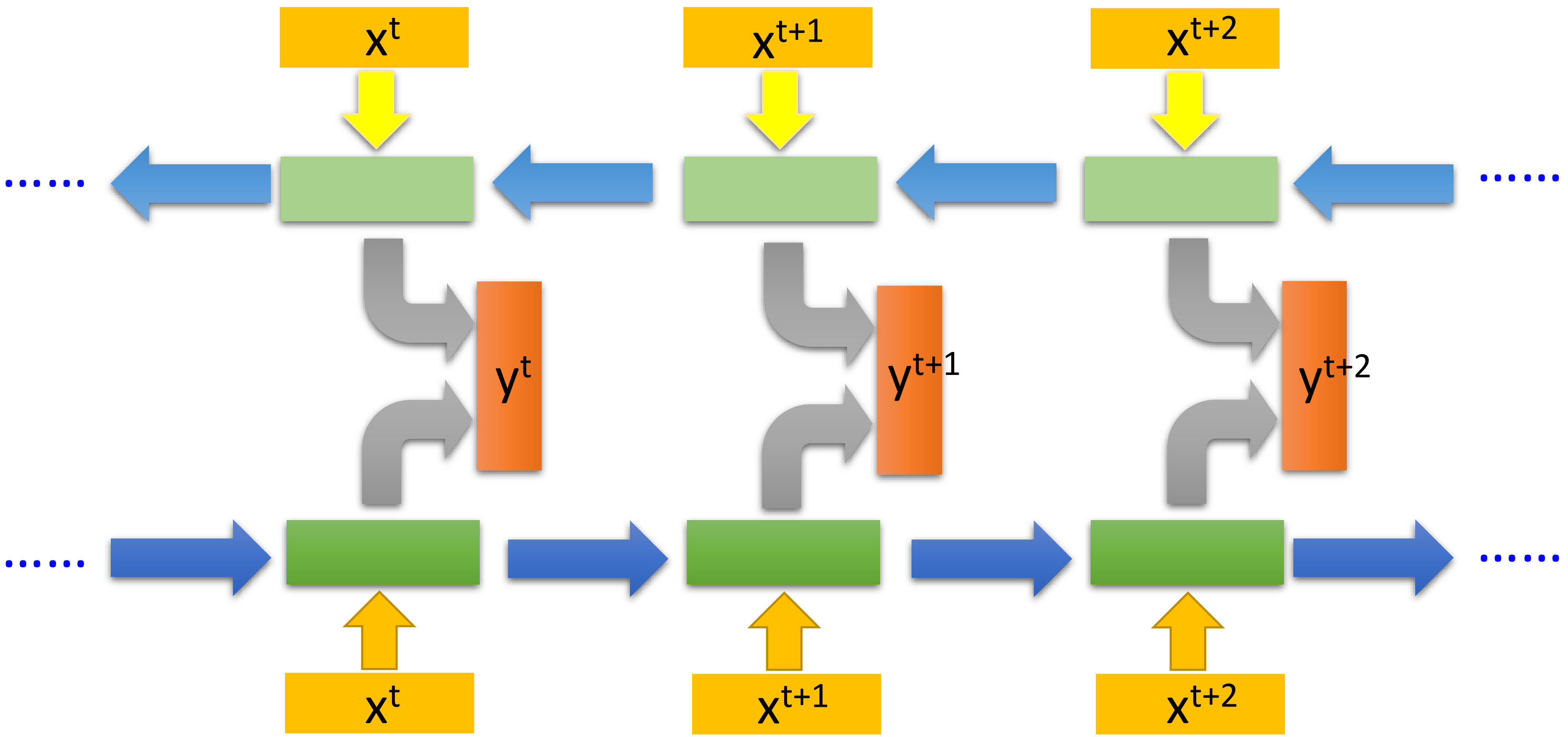
# Of course it can be deep …



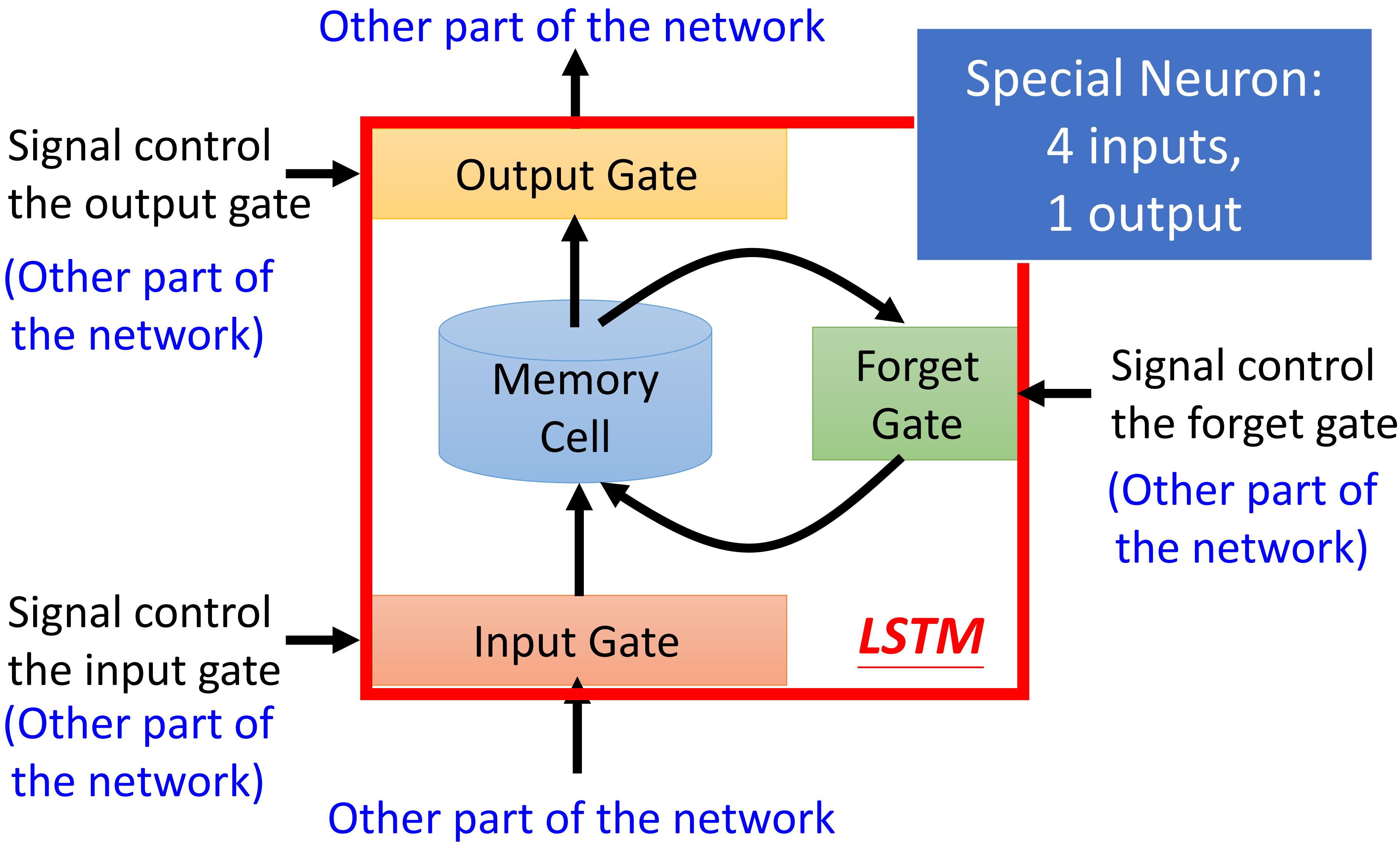
# Elman Network & Jordan Network

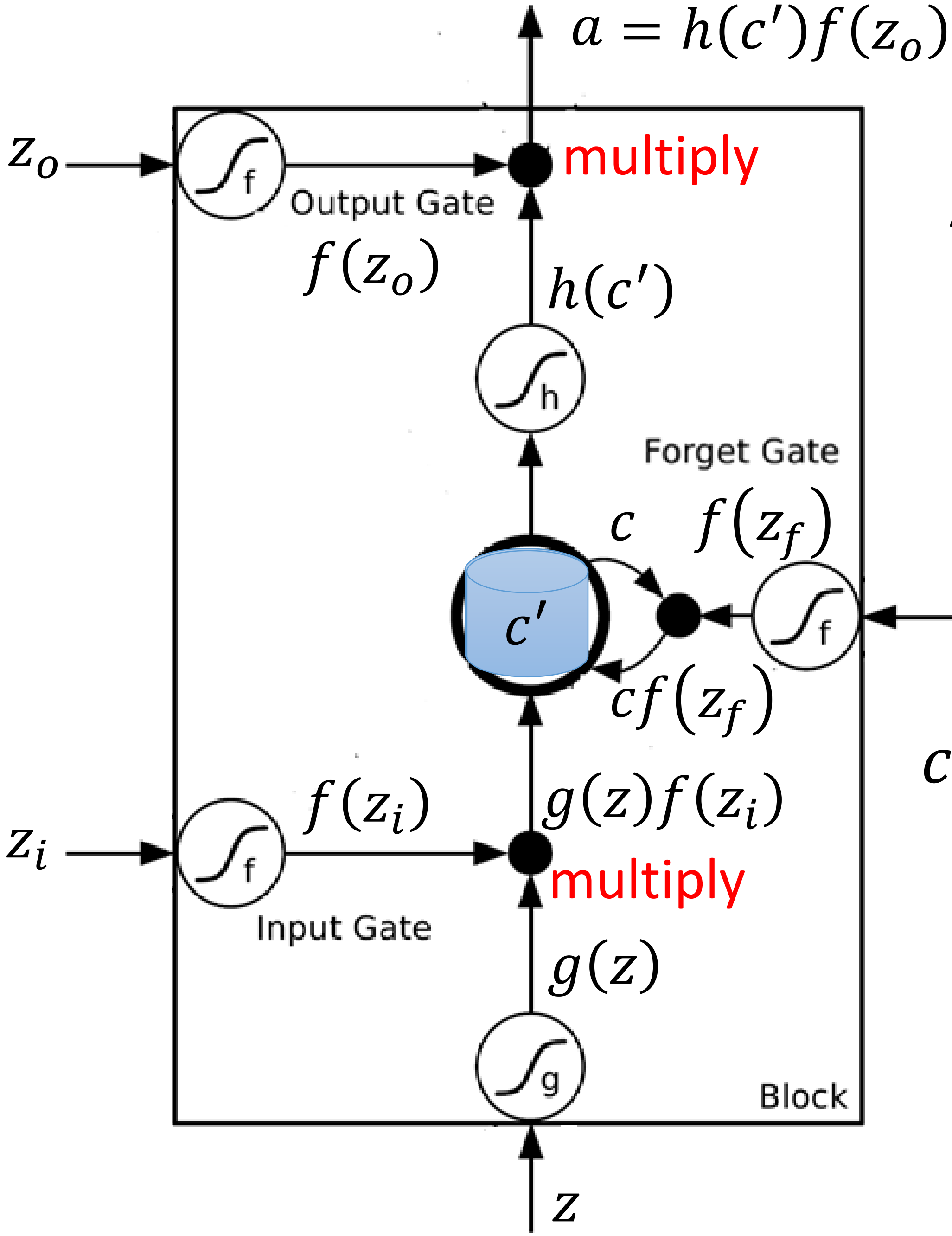


# Bidirectional RNN



# Long Short-term Memory (LSTM)






Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

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

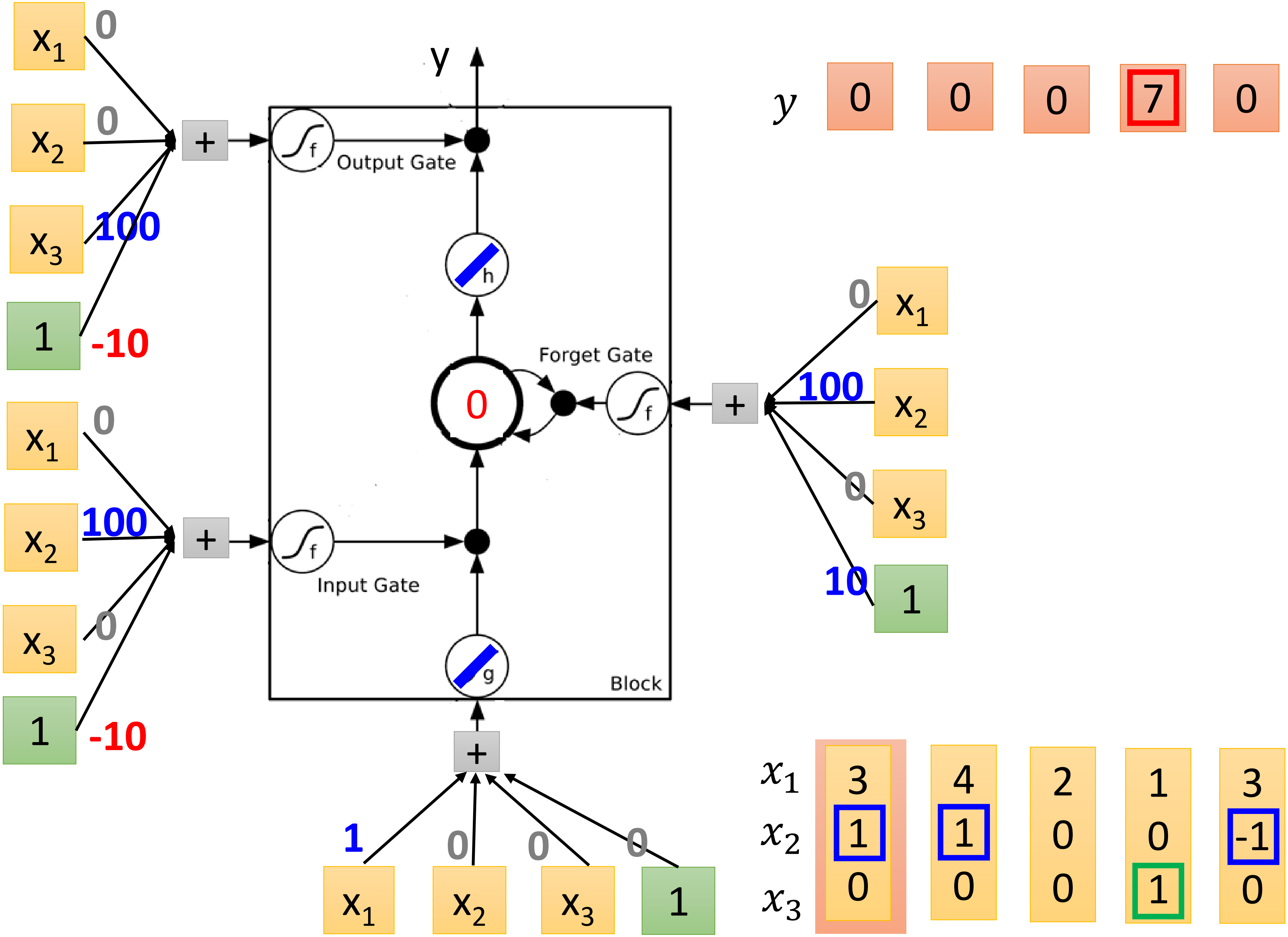
# LSTM - Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 3 | 3 | 7 | 7 | 7 | 0 | 6 |
| $x_1$ | 1 | 3 | 2 | 4 | 2 | 1 | 3 | 6 | 1 |
| $x_2$ | 0 | 1 | 0 | 1 | 0 | 0 | -1 | 1 | 0 |
| $x_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| $y$ | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 6 |

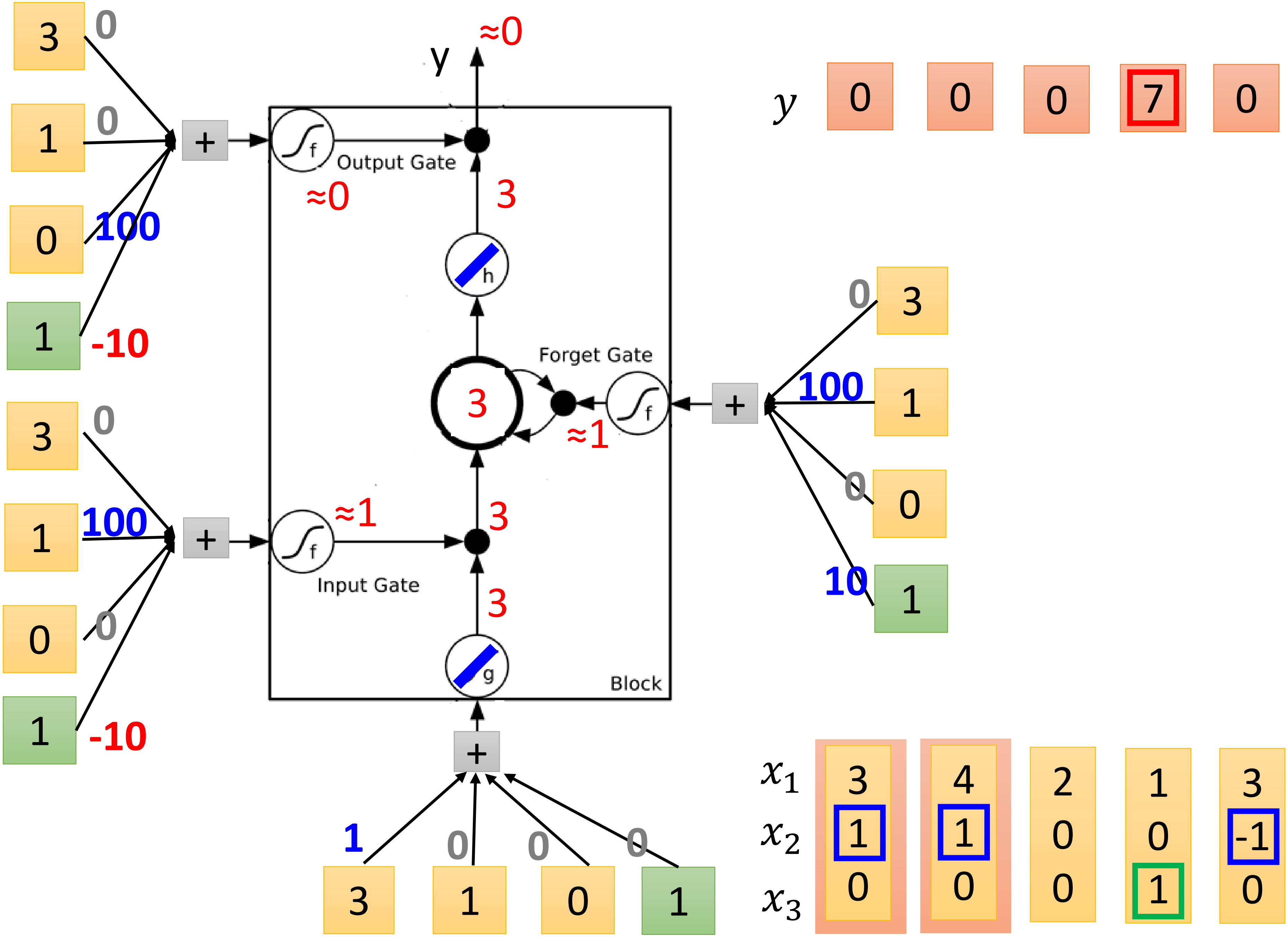
When $x_2$ = 1, add the numbers of $x_1$ into the memory

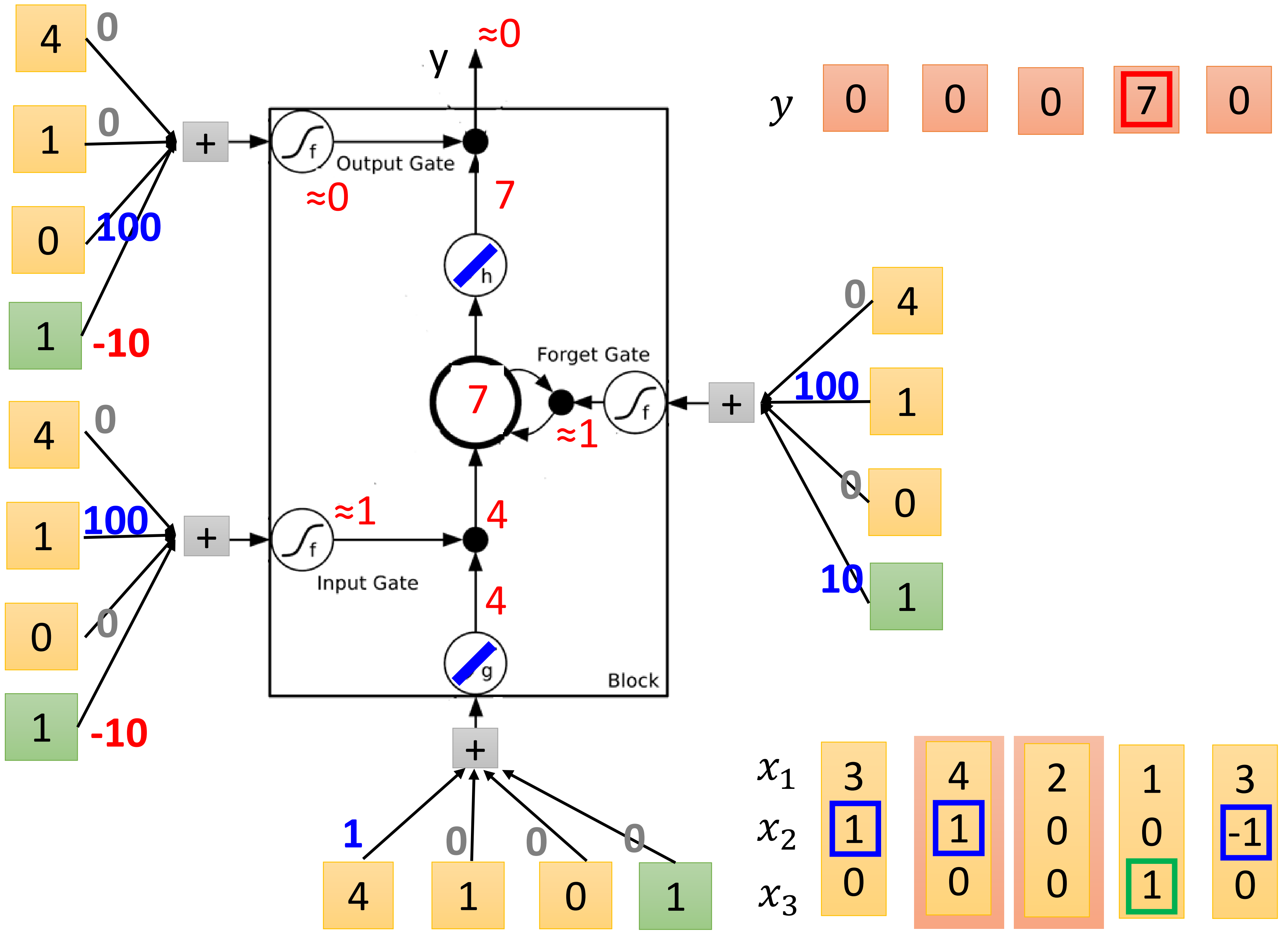
When $x_2$ = -1, reset the memory

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

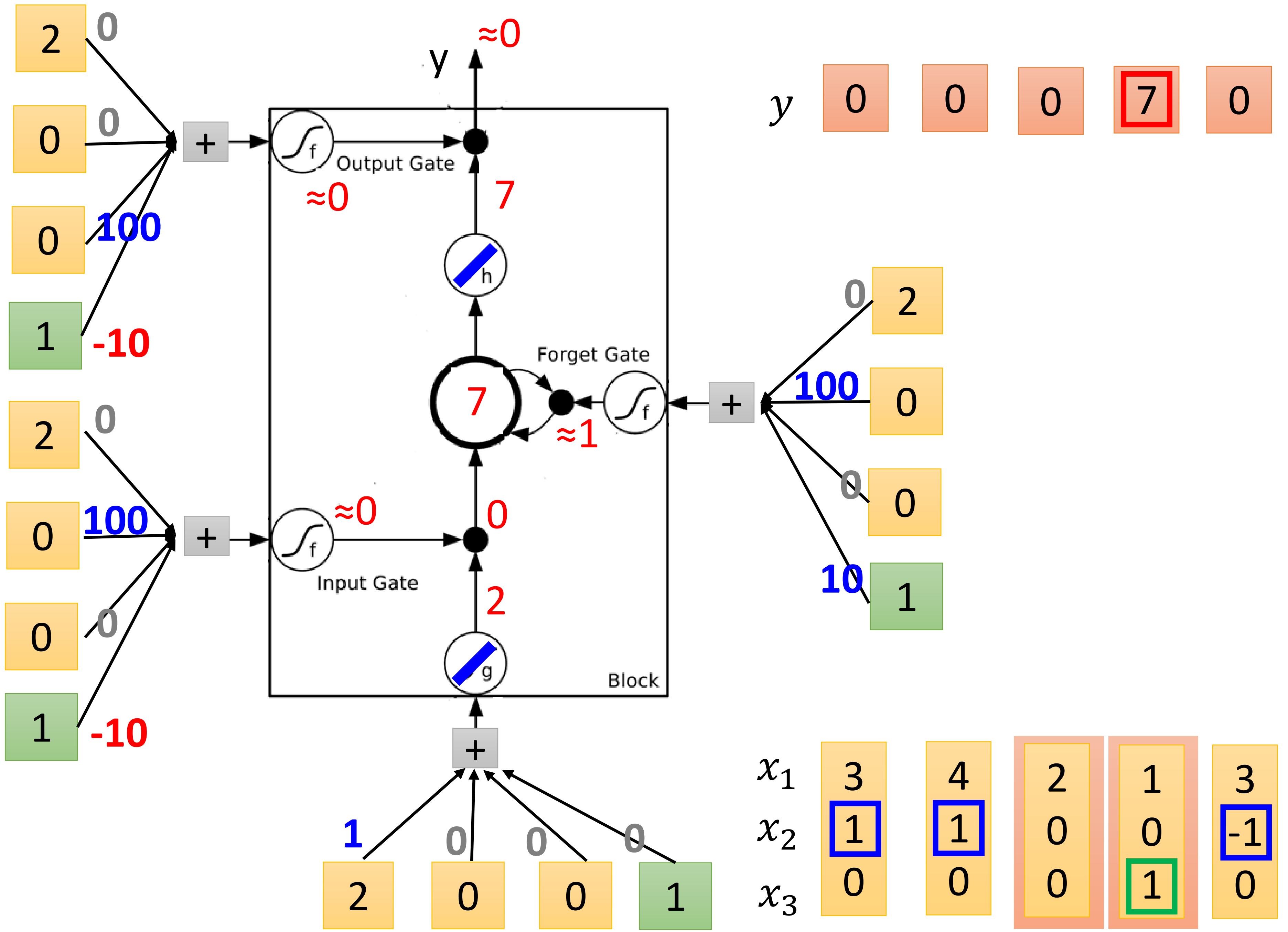


y
Output Gate
Forget Gate
Input Gate
Block
h
g
f
0
+
x1
x2
x3
1
100
-10
10
y 0 0 0 7 0
x1 3 4 2 1 3
x2 1 1 0 0 -1
x3 0 0 0 1 0

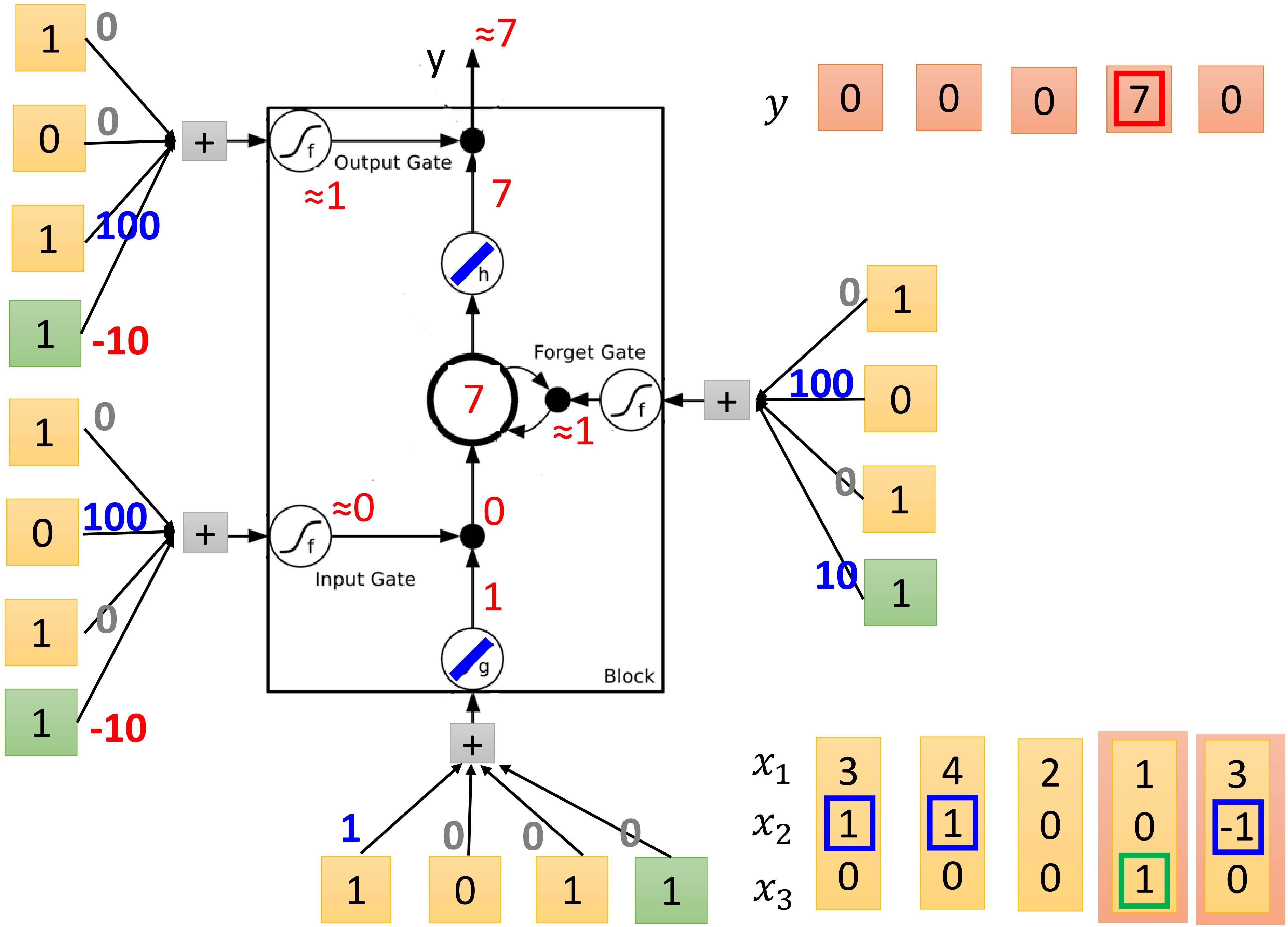

y
≈0
Output Gate
≈0
3
Forget Gate
3
≈1
≈1
3
Input Gate
3
Block
3 0
1 0
0 100
1 -10
3 0
1 100
0 0
1 -10
0 3
100 1
0 0
10 1
1 0 0 0
3 1 0 1
y 0 0 0 7 0
x1 3 4 2 1 3
x2 1 1 0 0 -1
x3 0 0 0 1 0



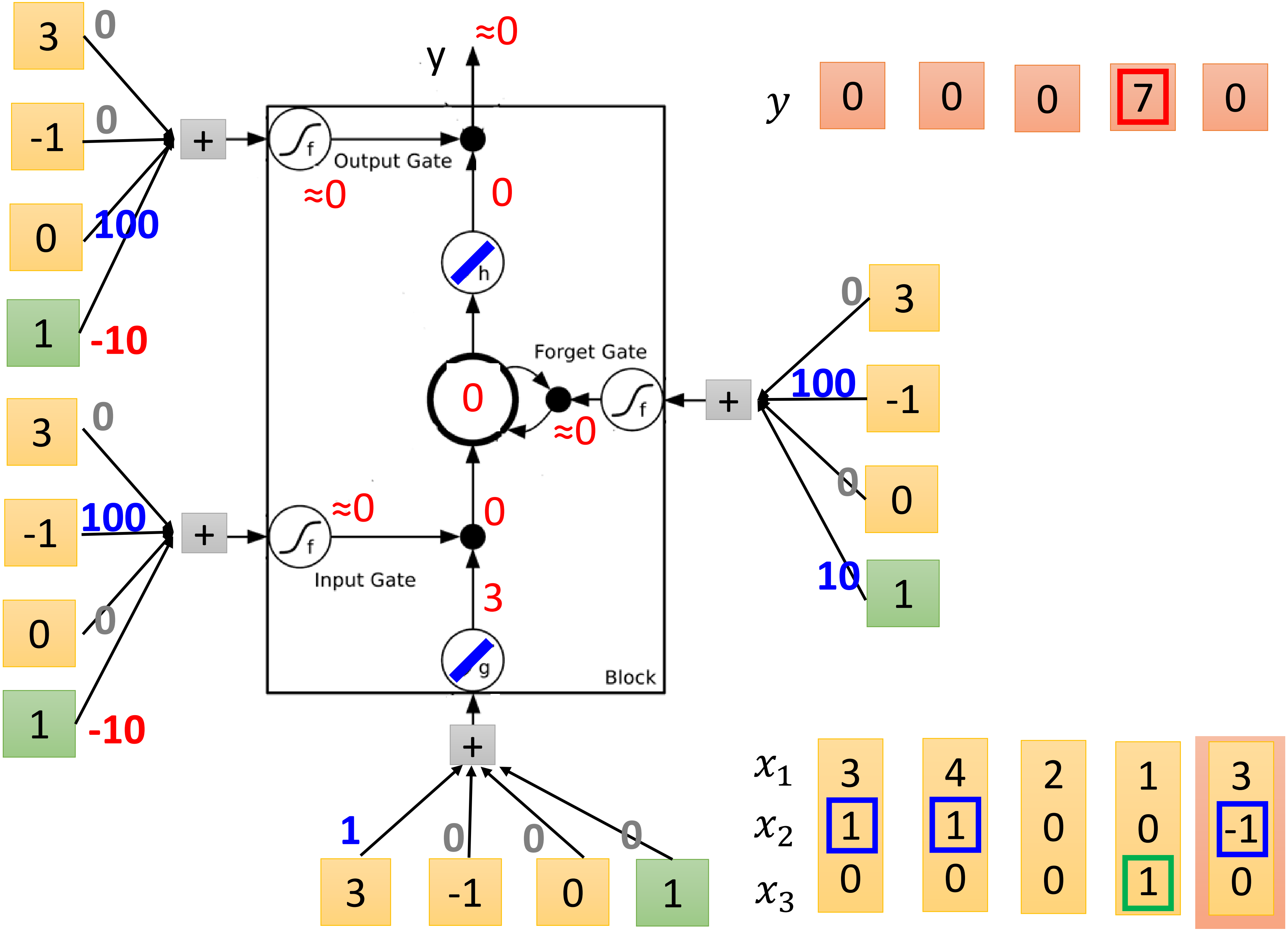
y
≈0
Output Gate
≈0
7
h
7
Forget Gate
≈1
≈1
Input Gate
4
4
g
Block
4
0
1
0
0
100
1
-10
4
0
1
100
0
0
1
-10
0
4
100
1
0
0
10
1
1
4
0
1
0
0
0
1
y
0
0
0
7
0
x1
x2
x3
3
1
0
4
1
0
2
0
0
1
0
1
3
-1
0



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



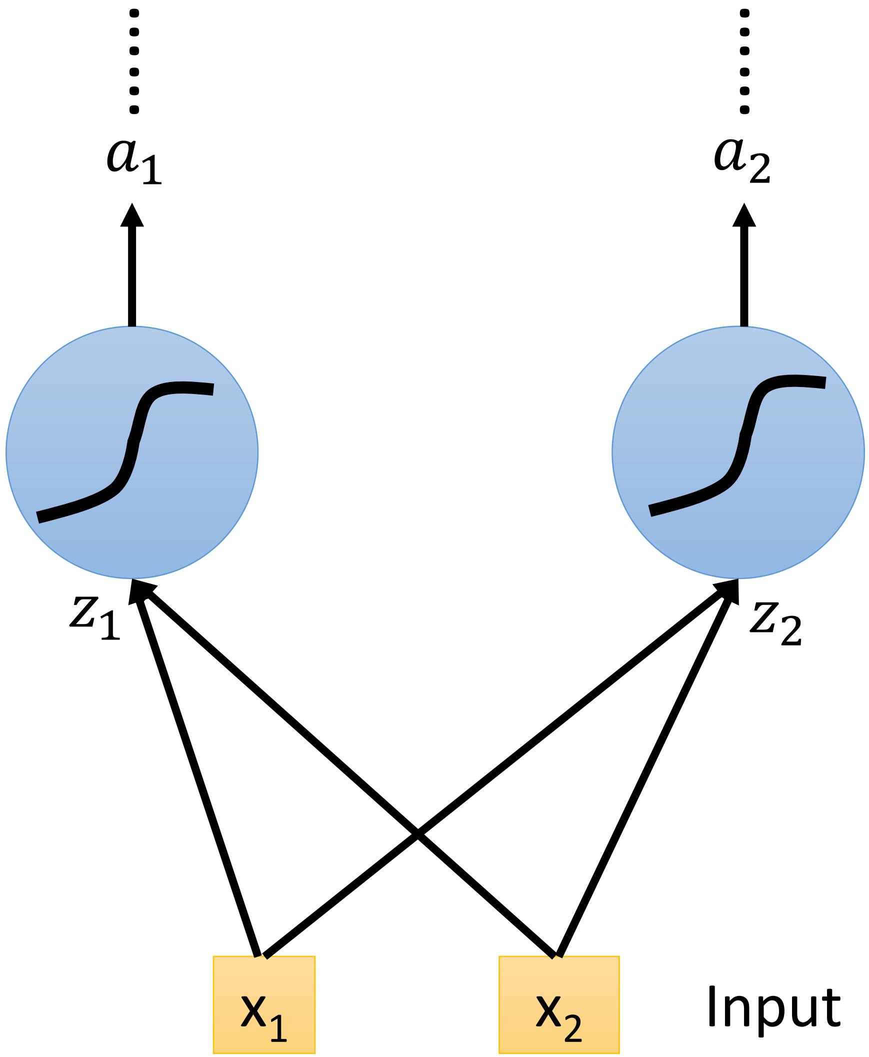
y
≈7
Output Gate
≈1
7
Forget Gate
7
≈1
≈0
0
Input Gate
1
Block
1 0
0 0
1 100
1 -10
1 0
0 100
1 0
1 -10
0 1
100 0
0 1
10 1
1 1
0 0
0 1
0 1
y 0 0 0 7 0
x1 3 4 2 1 3
x2 1 1 0 0 -1
x3 0 0 0 1 0

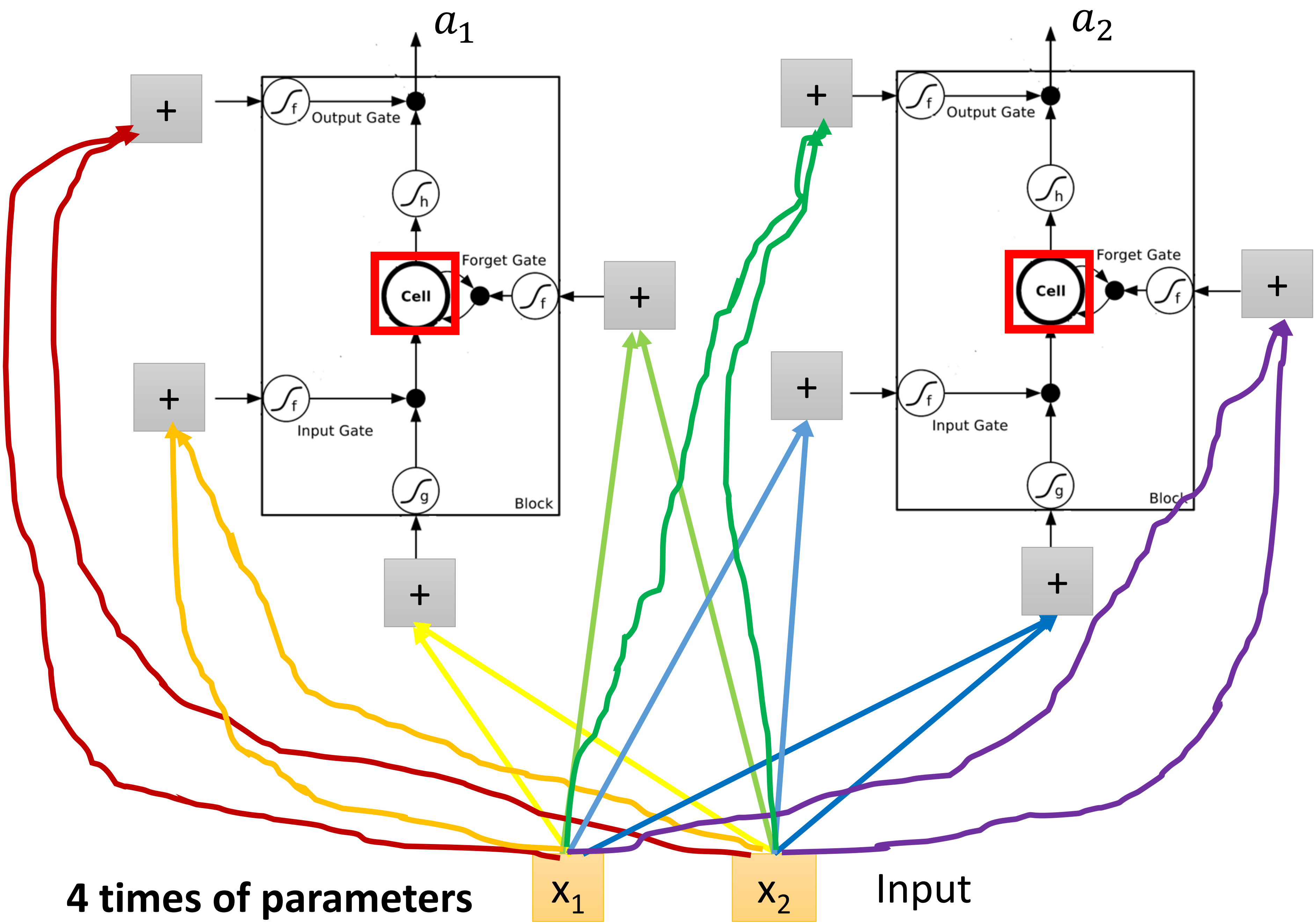


y
≈0
Output Gate
≈0
0
h
Forget Gate
0
≈0
Input Gate
≈0
0
3
g
Block
+
3
0
-1
0
0
100
1
-10
3
0
-1
100
0
0
1
-10
0
3
100
-1
0
0
10
1
1
3
0
-1
0
0
0
1
y
0
0
0
7
0
x1
x2
x3
3
1
0
4
1
0
2
0
0
1
0
1
3
-1
0

Original Network:

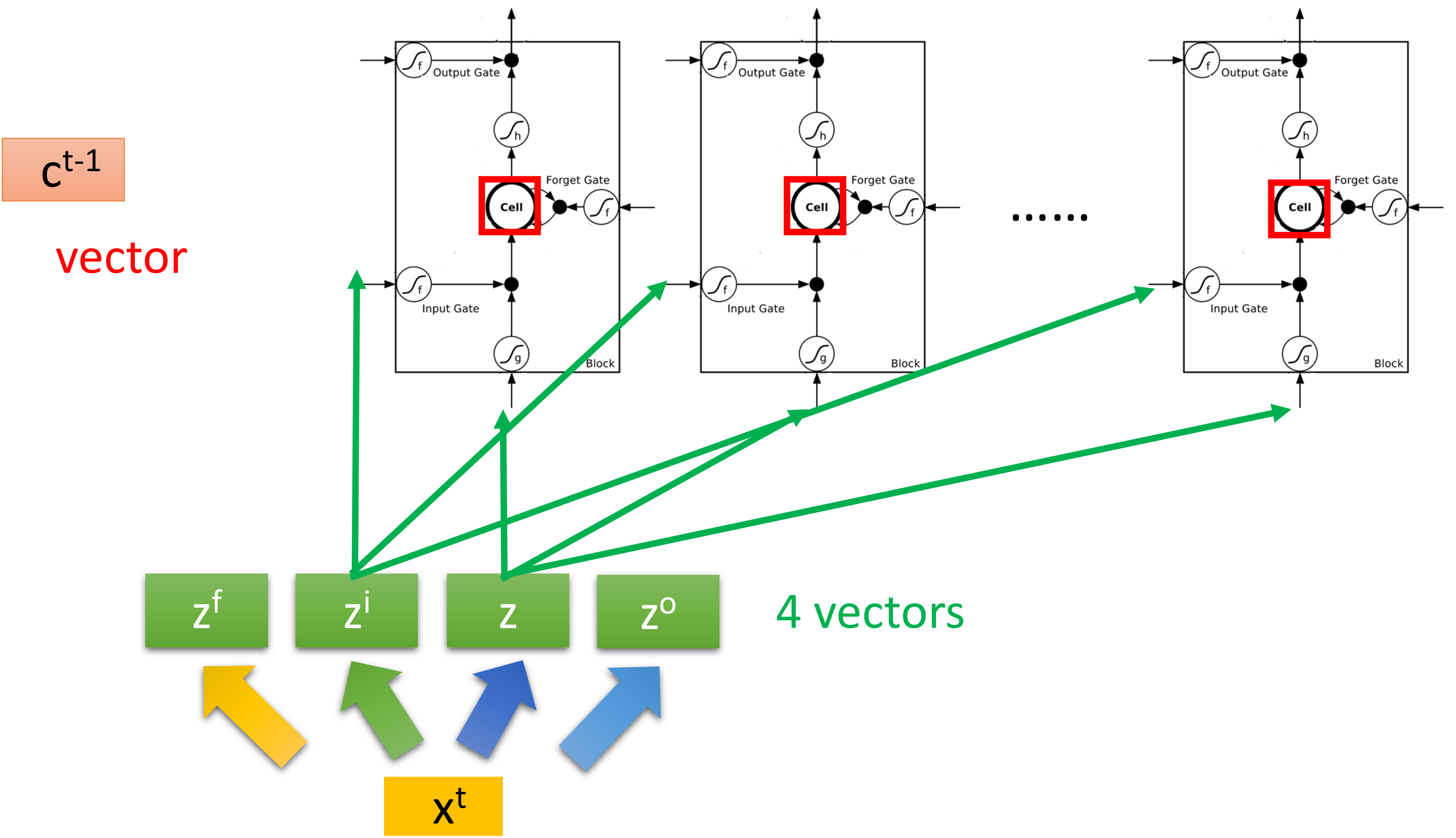
➢Simply replace the neurons with LSTM





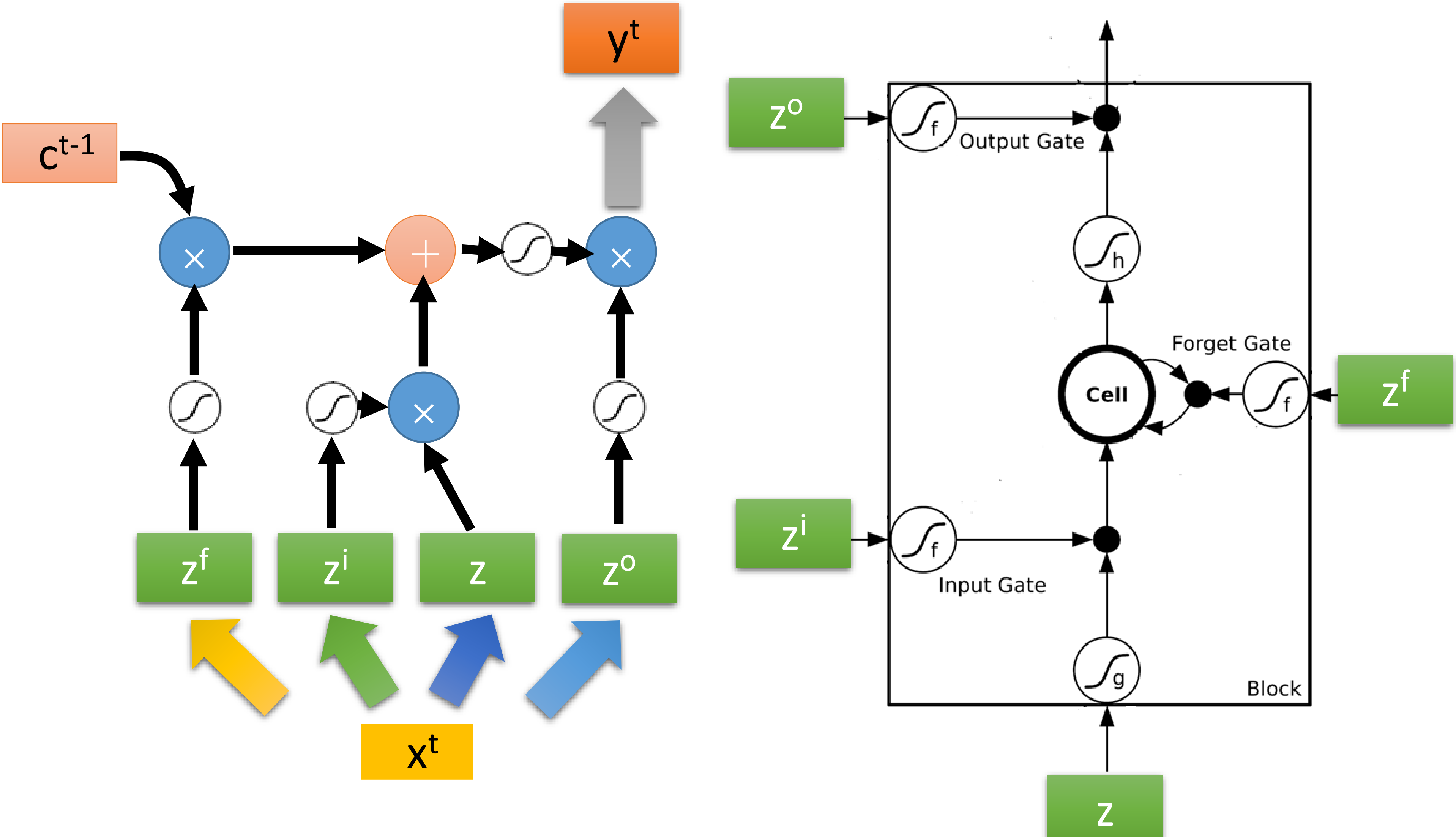

**4 times of parameters**

# LSTM



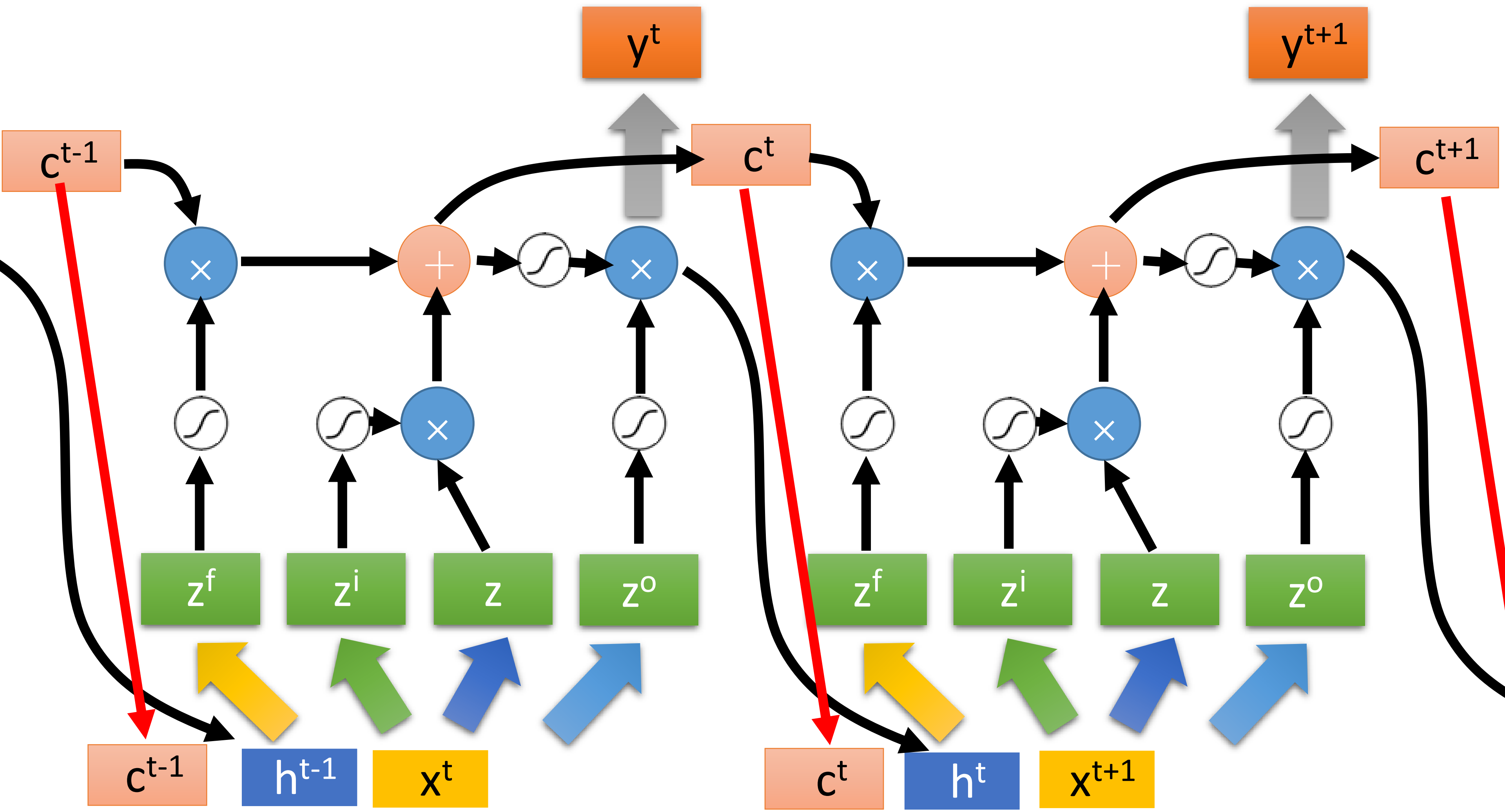
$c^{t-1}$
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
......
$z^f$
$z^i$
$z$
$z^o$
4 vectors
$x^t$

# LSTM

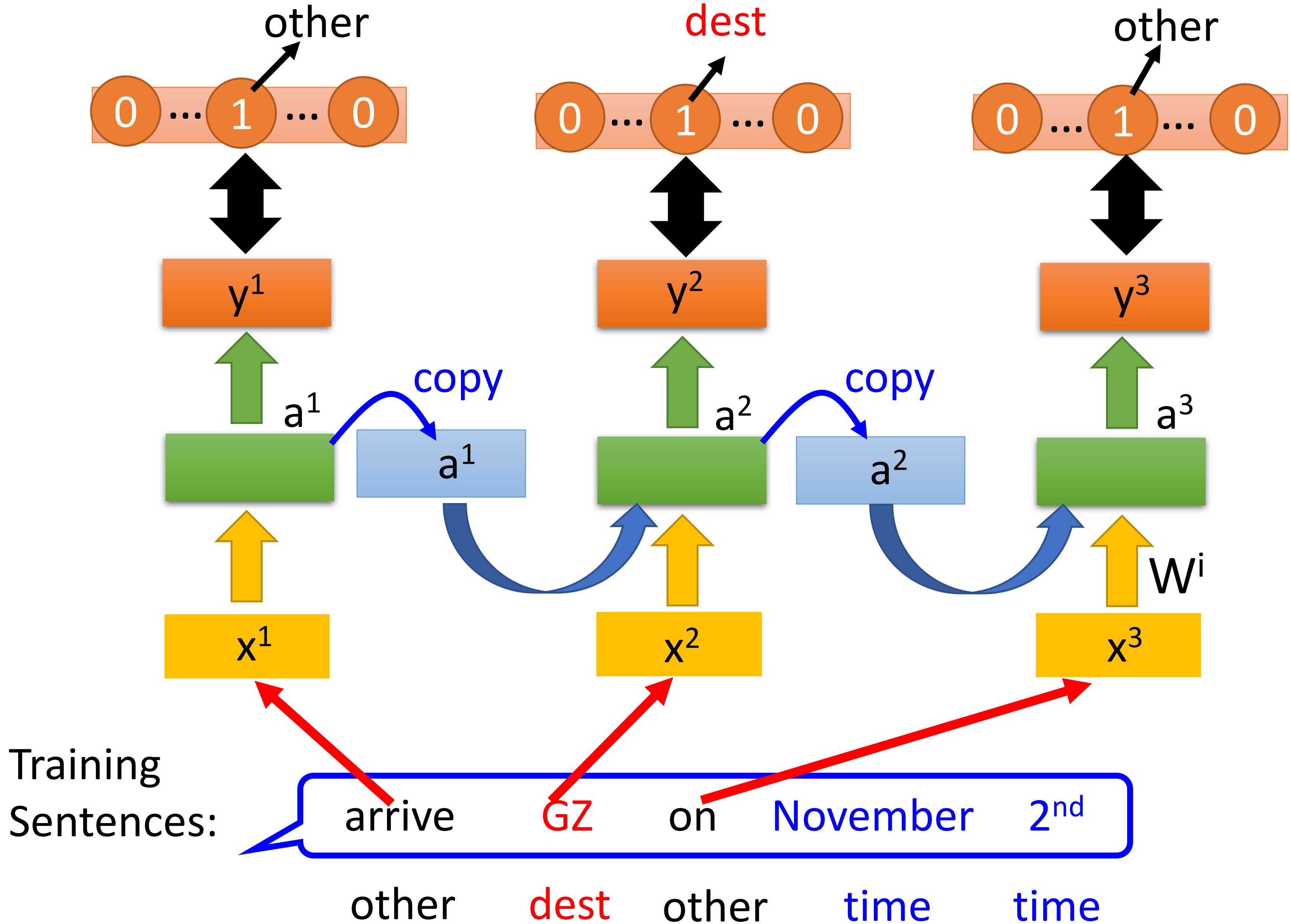

$y^t$
$c^{t-1}$
$z^f$
$z^i$
$z$
$z^o$
$x^t$
$z^o$
Output Gate
Forget Gate
Cell
$z^f$
$z^i$
Input Gate
Block
$z$

# LSTM

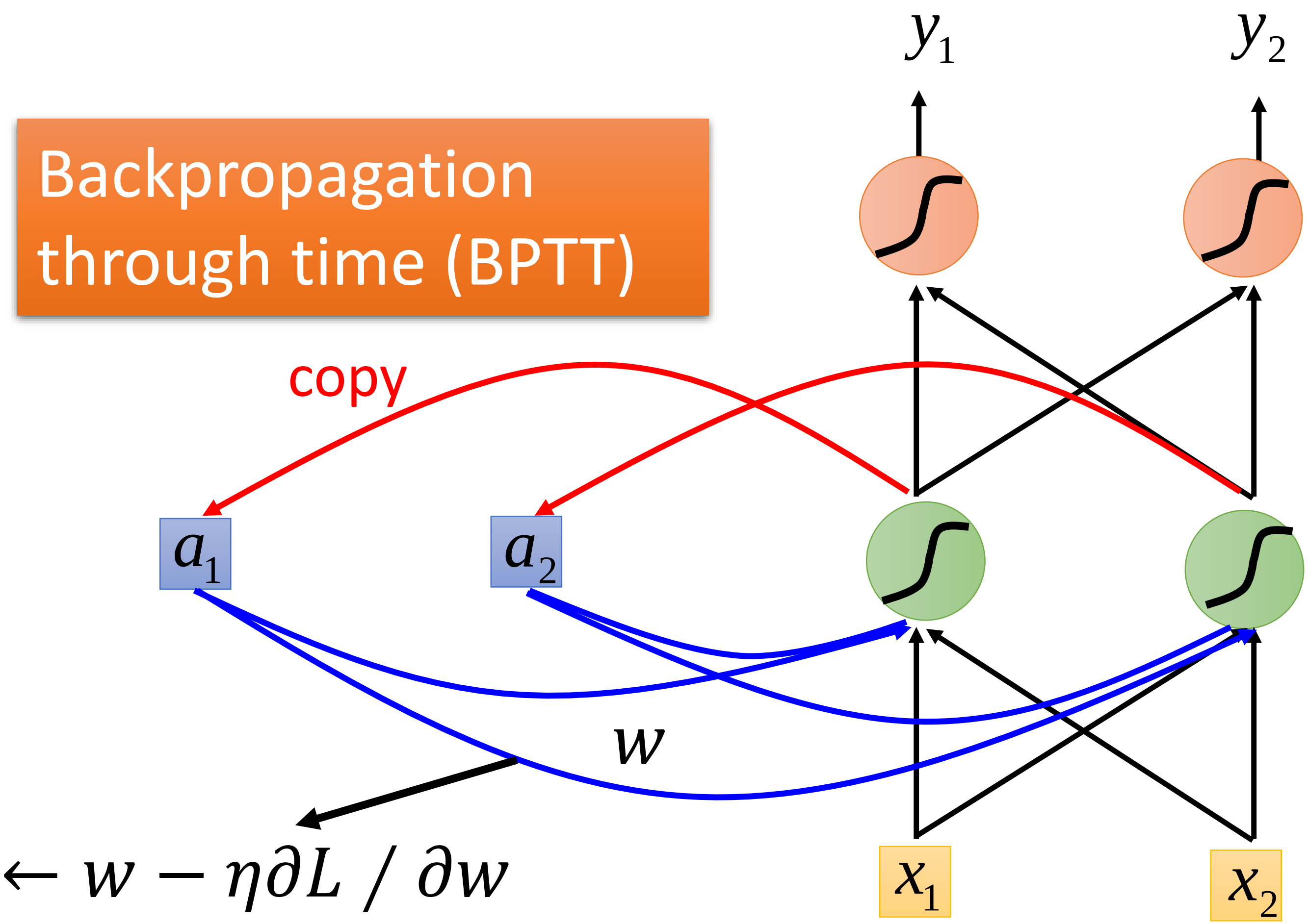
Extension: “peephole”



# Learning Target



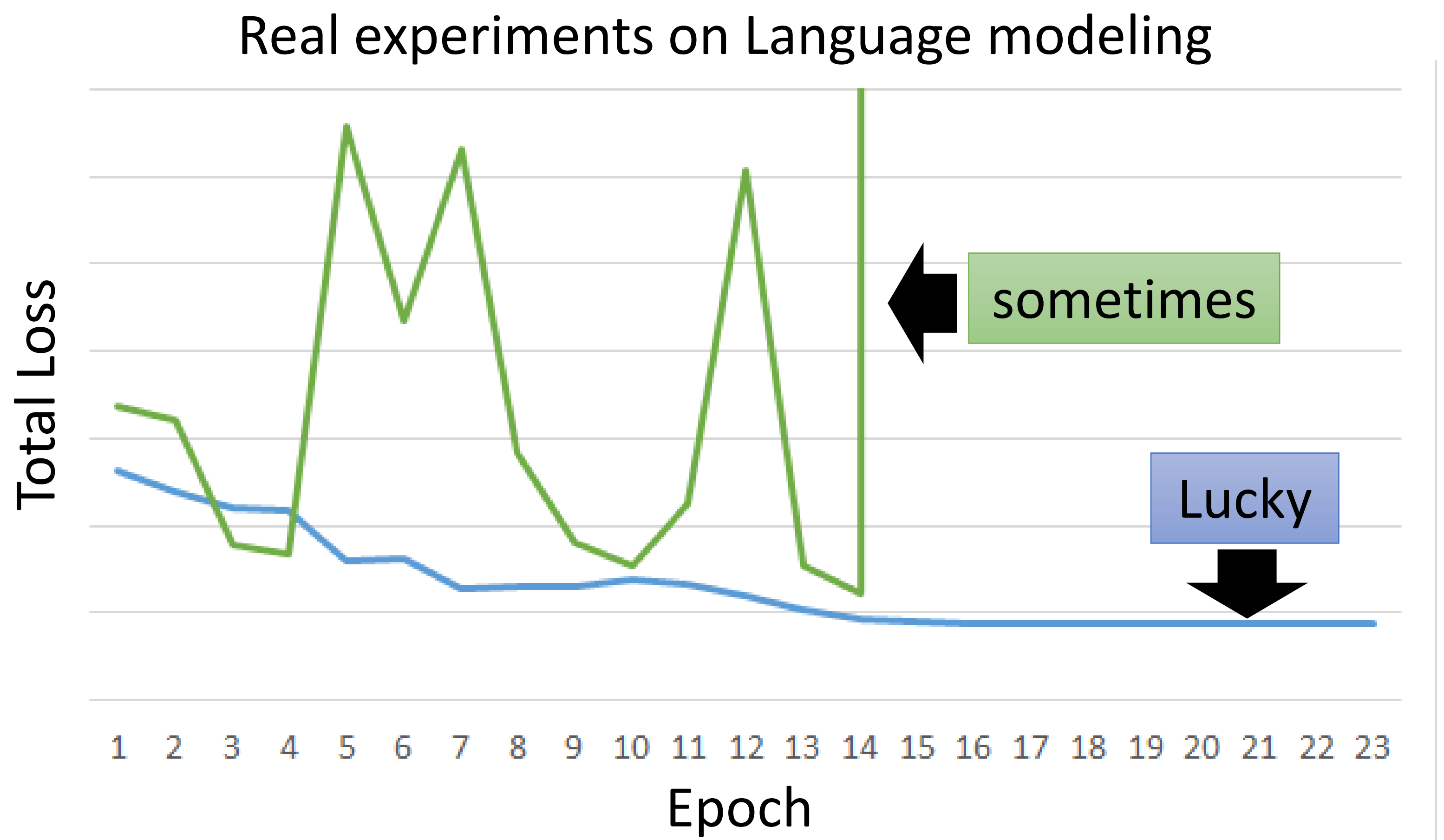
# Learning



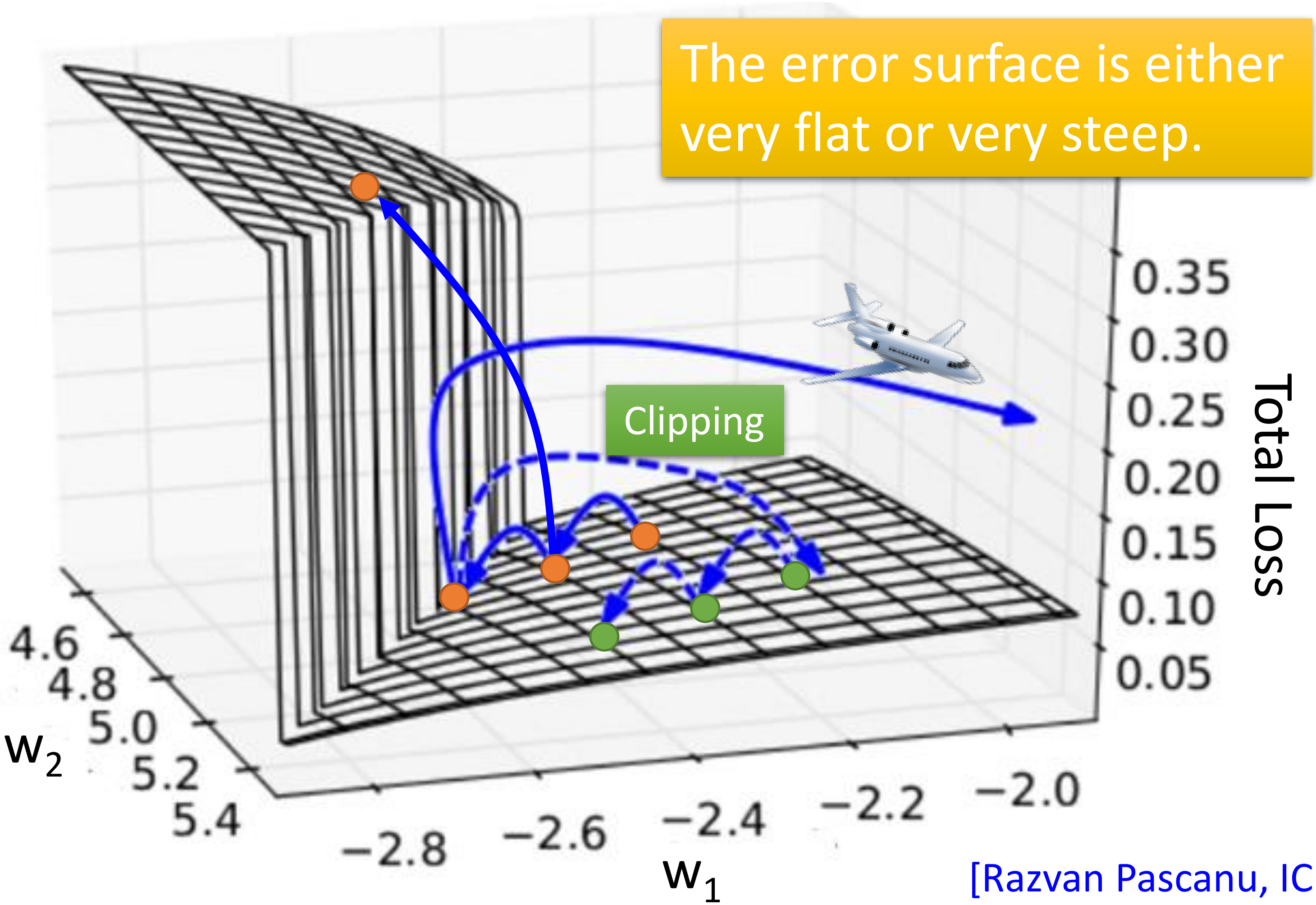

$$w \leftarrow w - \eta \partial L / \partial w$$

# Unfortunately ……

- RNN-based network is not always easy to learn



# The error surface is rough.




[Razvan Pascanu, ICML'13]

# Why?

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

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

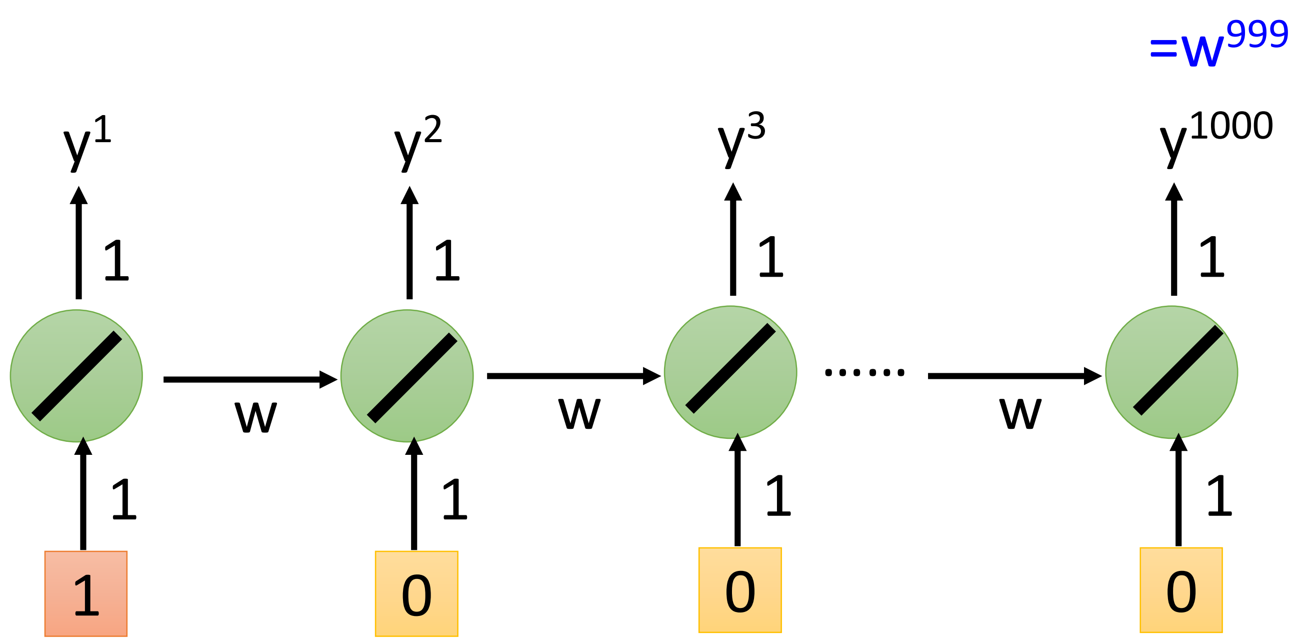
Large $\partial L / \partial w$ → Small Learning rate?

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

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

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

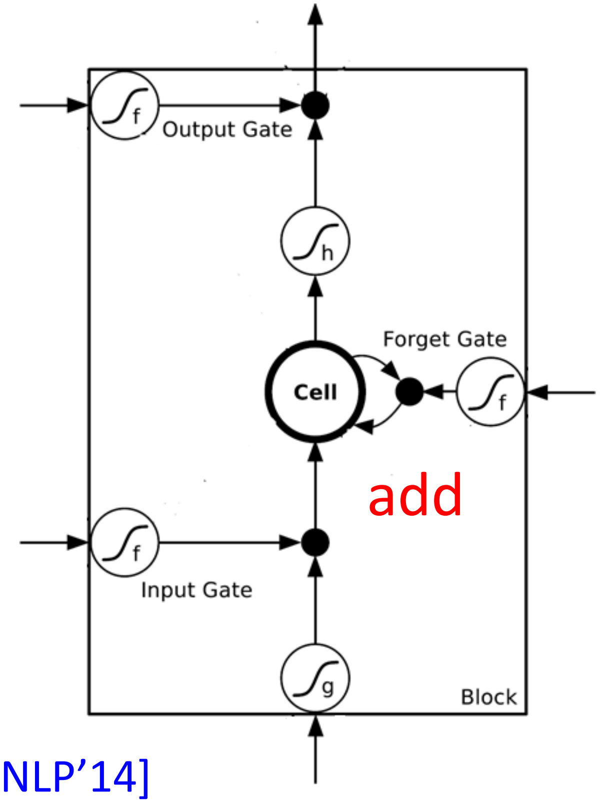
***Toy Example***



# 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.)

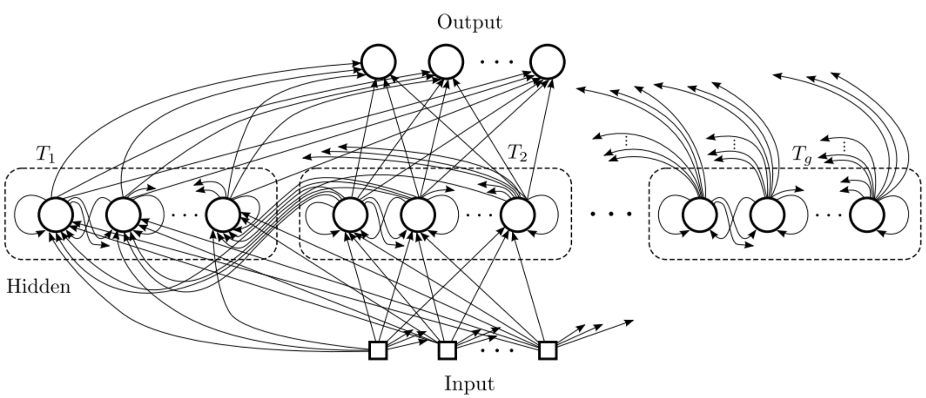



Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

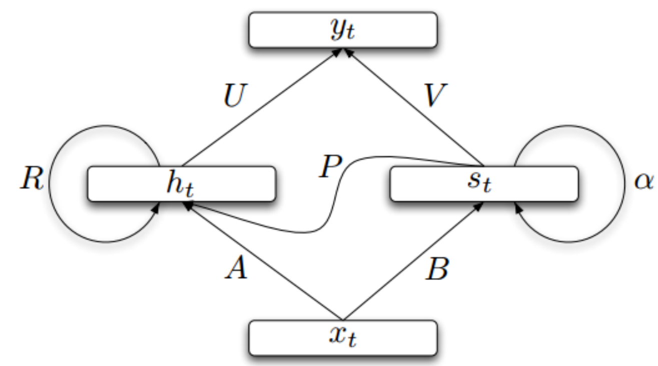
# Helpful Techniques

Clockwise RNN




[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)




[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

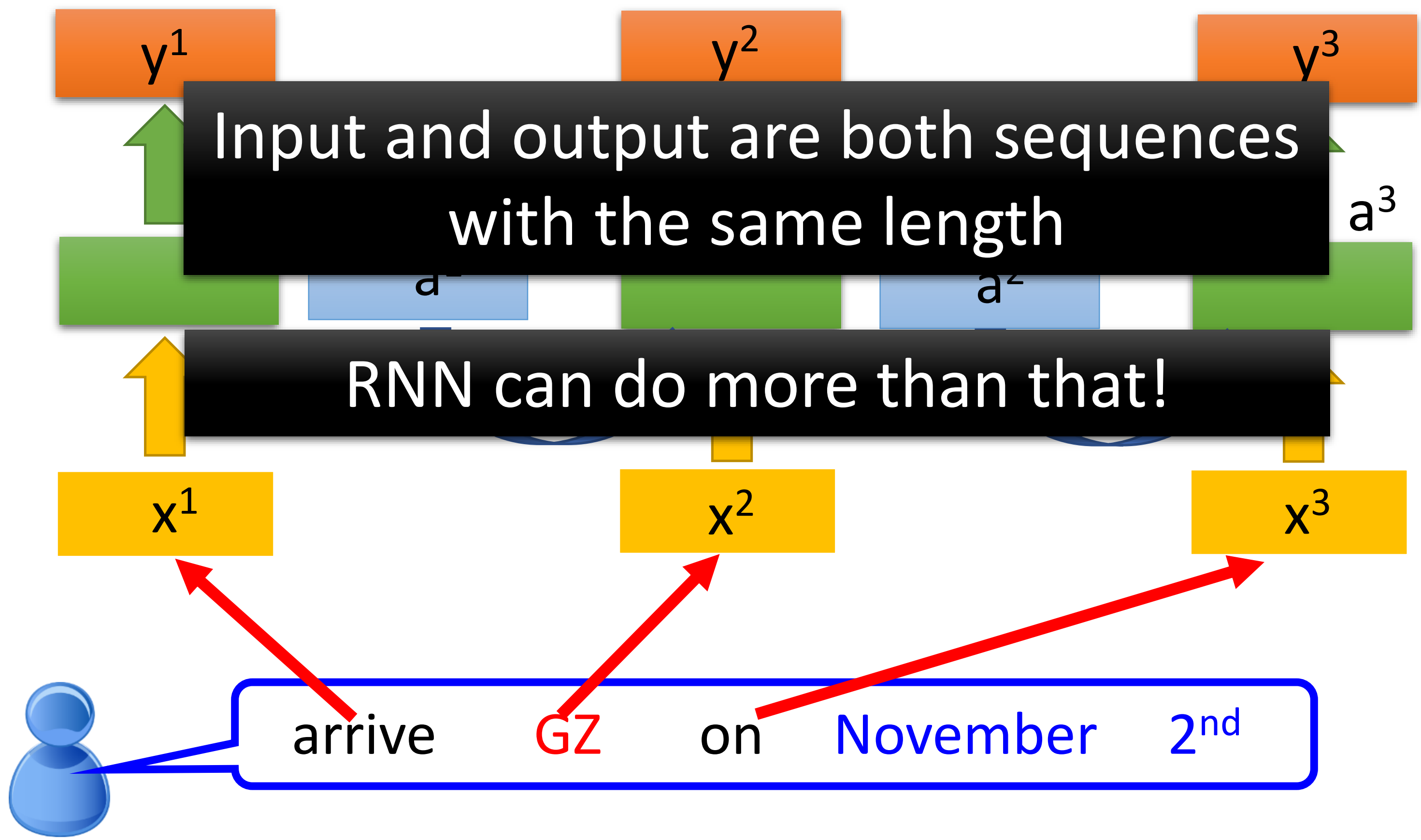
- Outperform or be comparable with LSTM in 4 different tasks

# More Applications ……

Probability of “arrive” in each slot

Probability of “GZ” in each slot

Probability of “on” in each slot



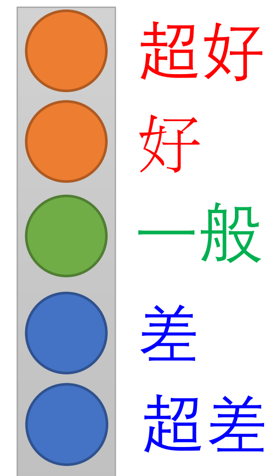

Input and output are both sequences with the same length

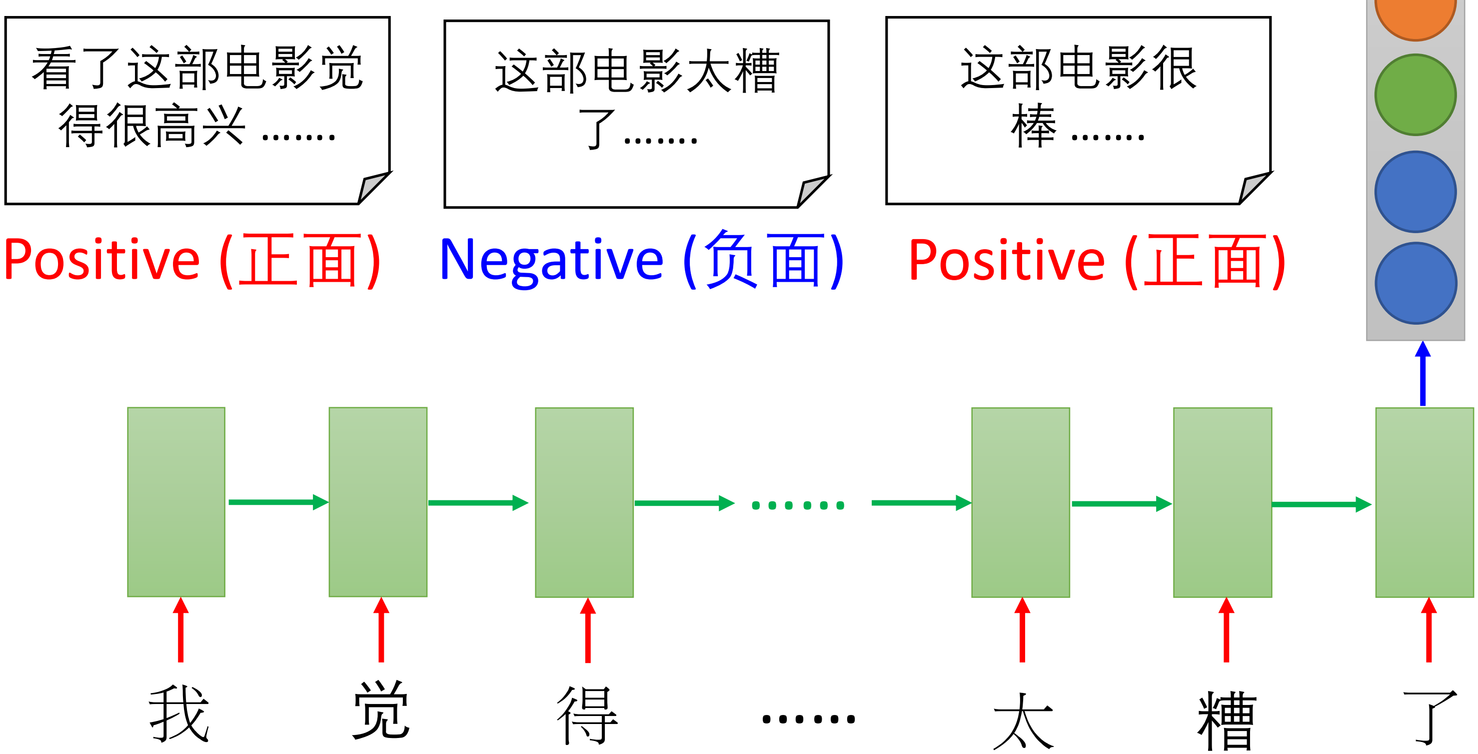
RNN can do more than that!

# Many to one

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

***Sentiment Analysis***



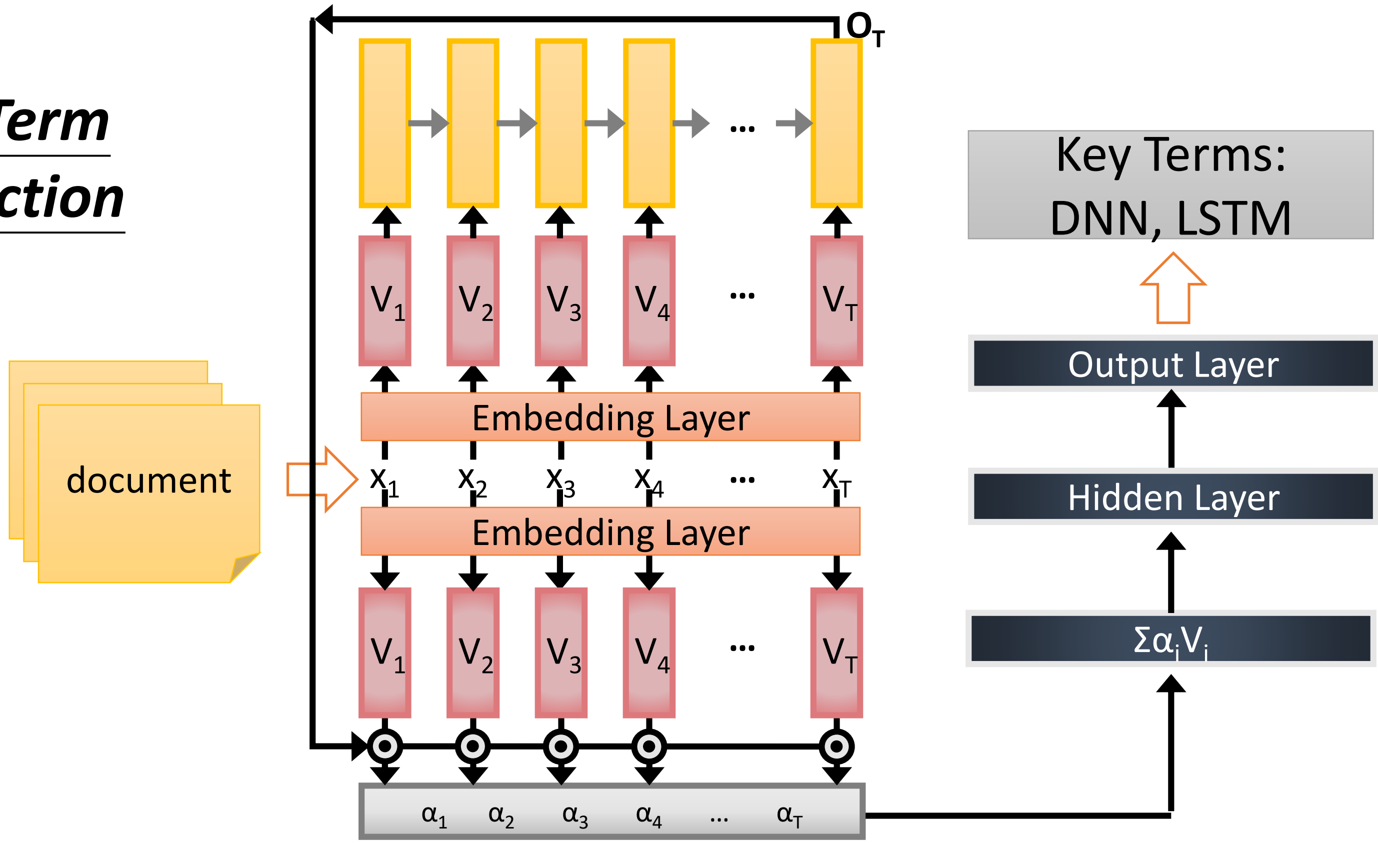



# Many to one

[Shen & Lee, Interspeech 16]

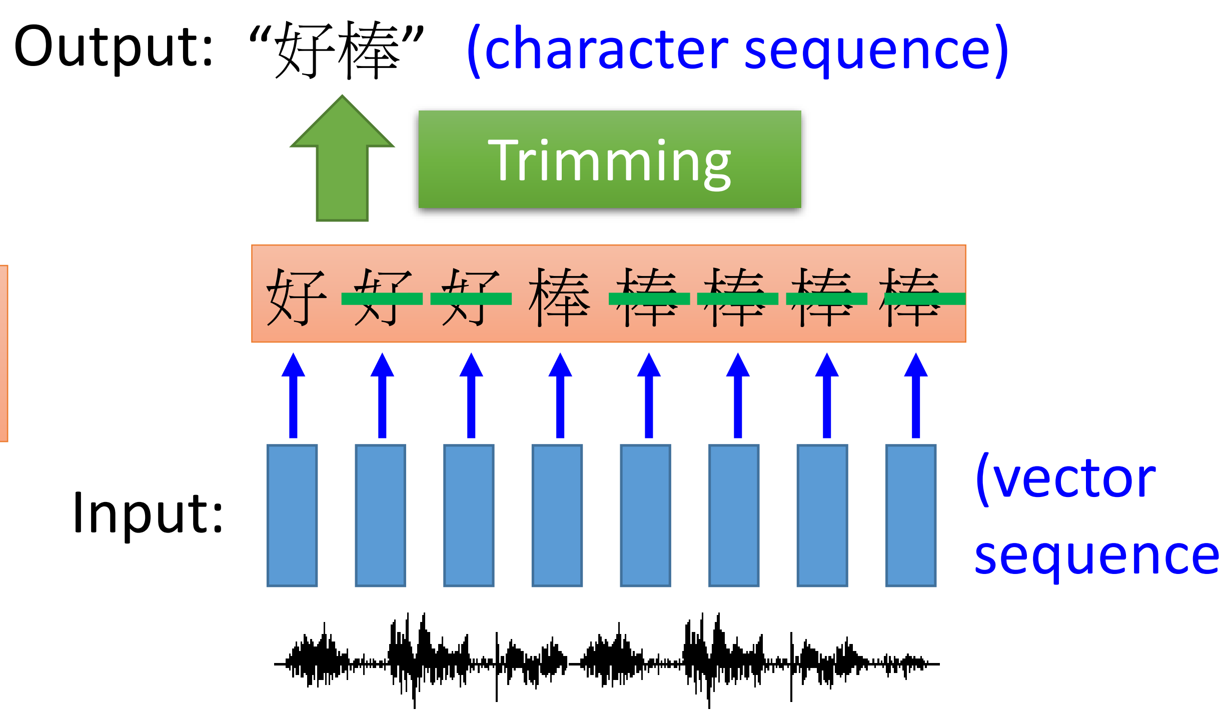
- Input is a vector sequence, but output is only one vector

***Key Term Extraction***



# Many to Many (Output is shorter)

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

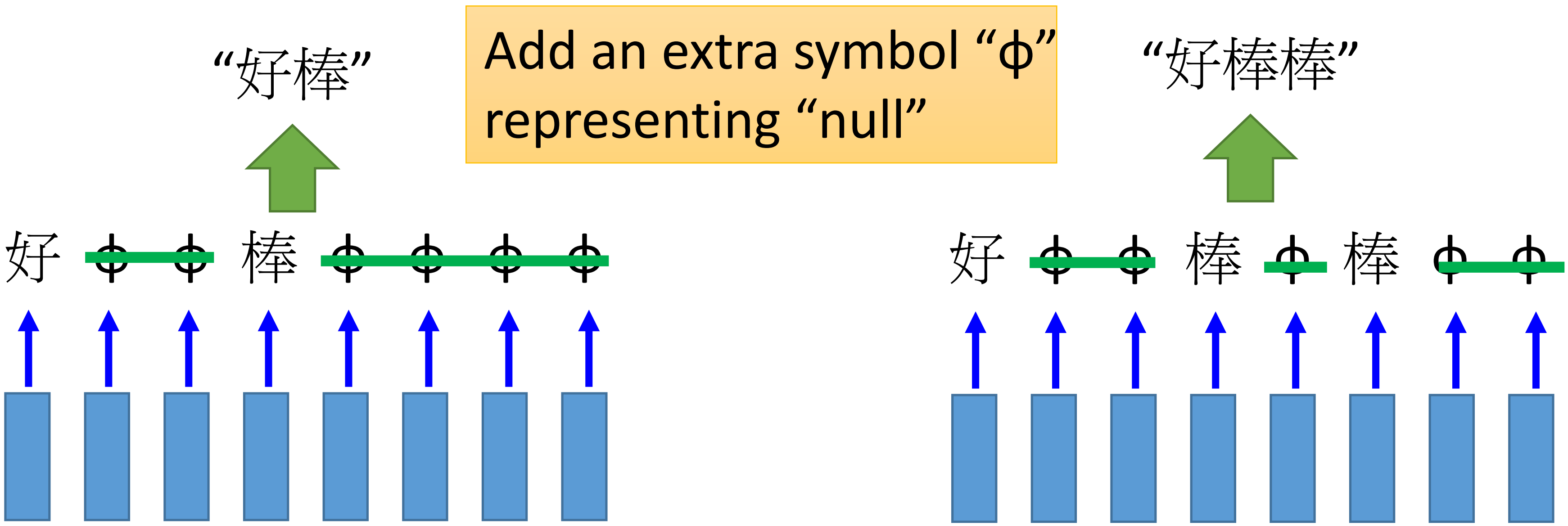



Problem?

Why can’t it be “好棒棒”

# Many to Many (Output is shorter)

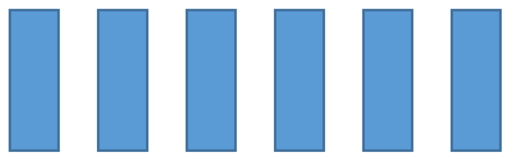
- Both input and output are both sequences, ***but the output is shorter.***
- Connectionist Temporal Classification (CTC) [Alex Graves, ICML'06][Alex Graves, ICML'14][Haşim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]



# Many to Many (Output is shorter)

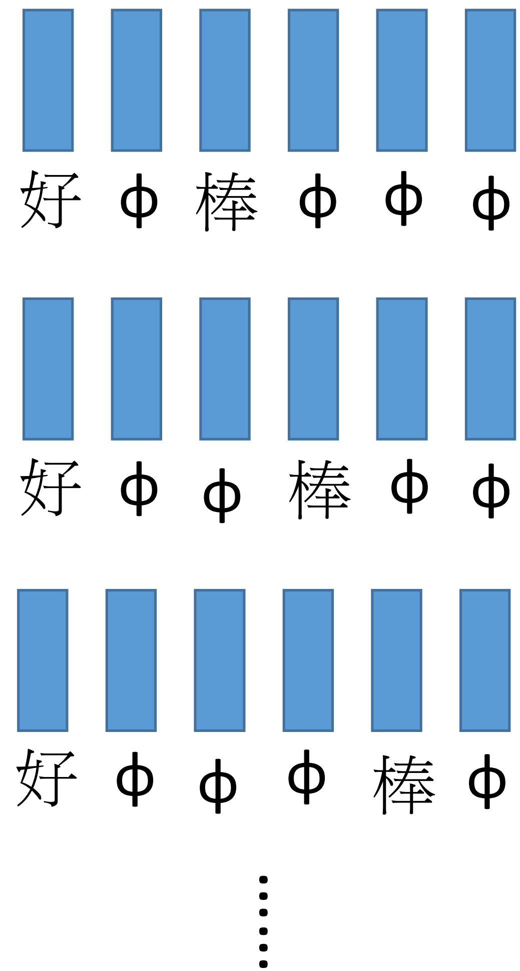
- CTC: Training

Acoustic Features:



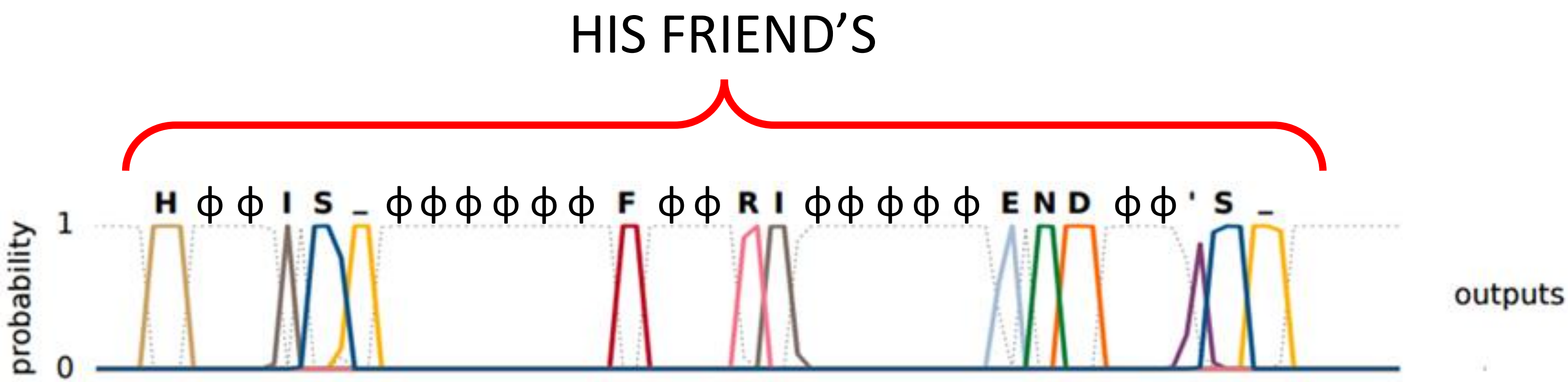
Label: 好 棒

All possible alignments are considered as correct.



# Many to Many (Output is shorter)

- CTC: example




Graves, Alex, and Navdeep Jaitly. "Towards end-to-end speech recognition with recurrent neural networks." *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014.

# The End!