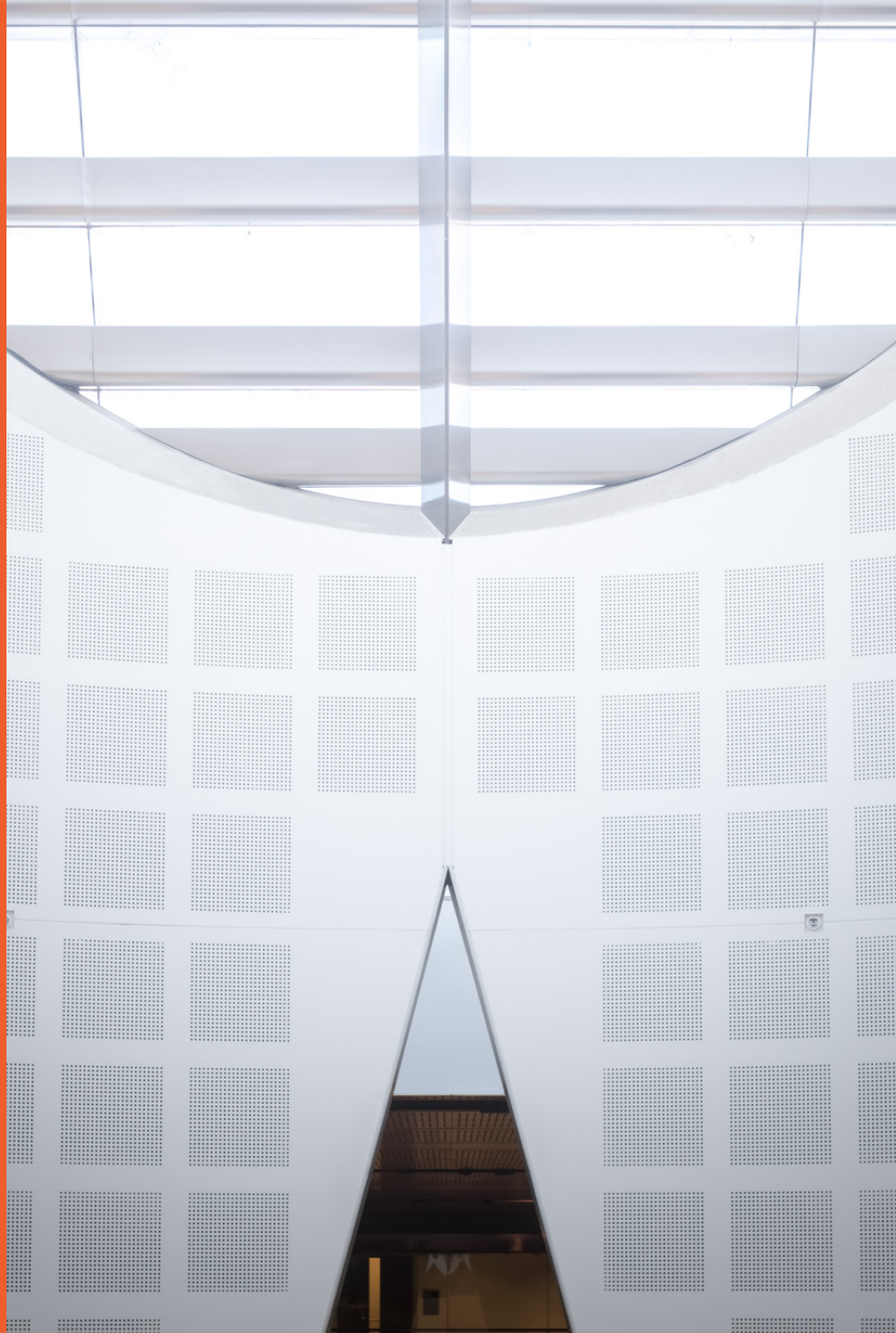
Recurrent Neural Networks

Dr Chang Xu

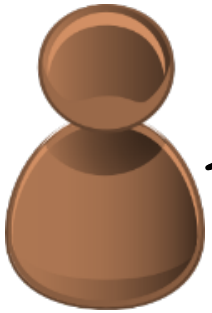
School of Computer Science

THE UNIVERSITY OF SYDNEY

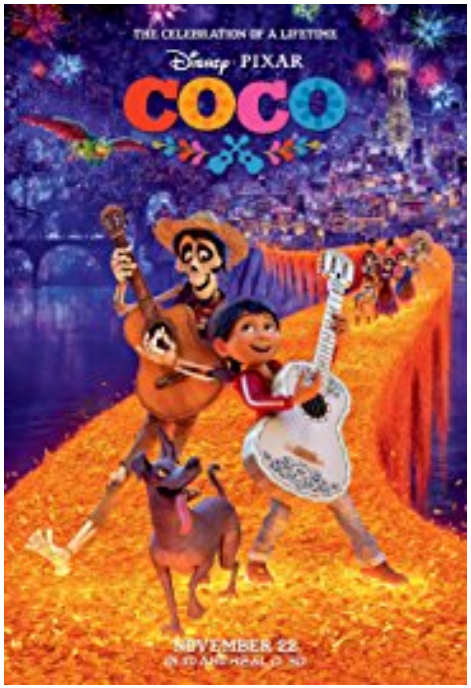


Example Application

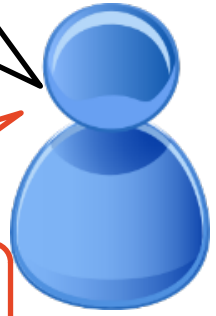
- Sentiment analysis



How about the COCO ?



Complicated? Yes, and a little slow, too.



Rank 3/5

But the animation is as colorful as the story and emotions do eventually reach the desired level.

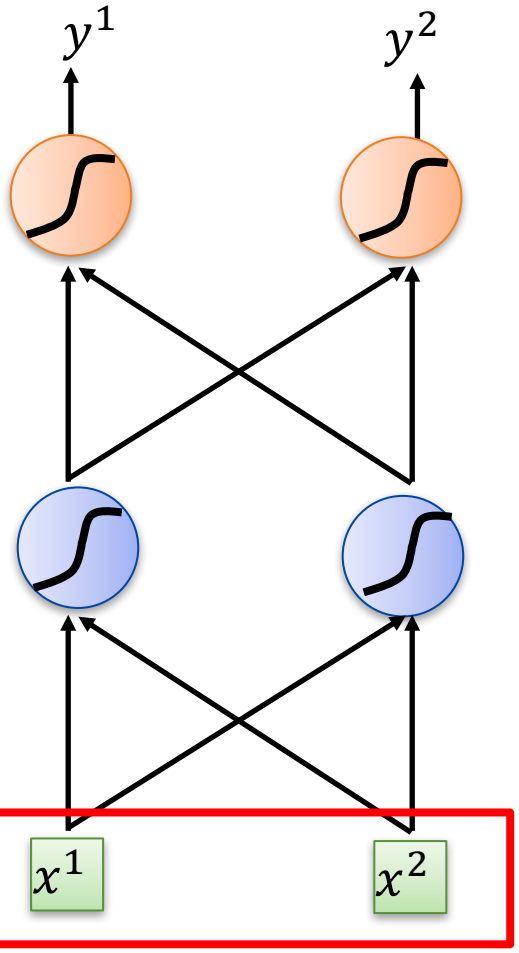
Image credit to https://www.rottentomatoes.com/m/coco_2017/reviews/

Example Application

Image credit to https://twitter.com/SamsungMobile/status/967807667463958531

Recurrent Neural Network

- Input
  - Word sequence
  - Each word is represented by vector
- Methods
  - One hot encoding
  - Word hash
  - Word embedding



Complicated

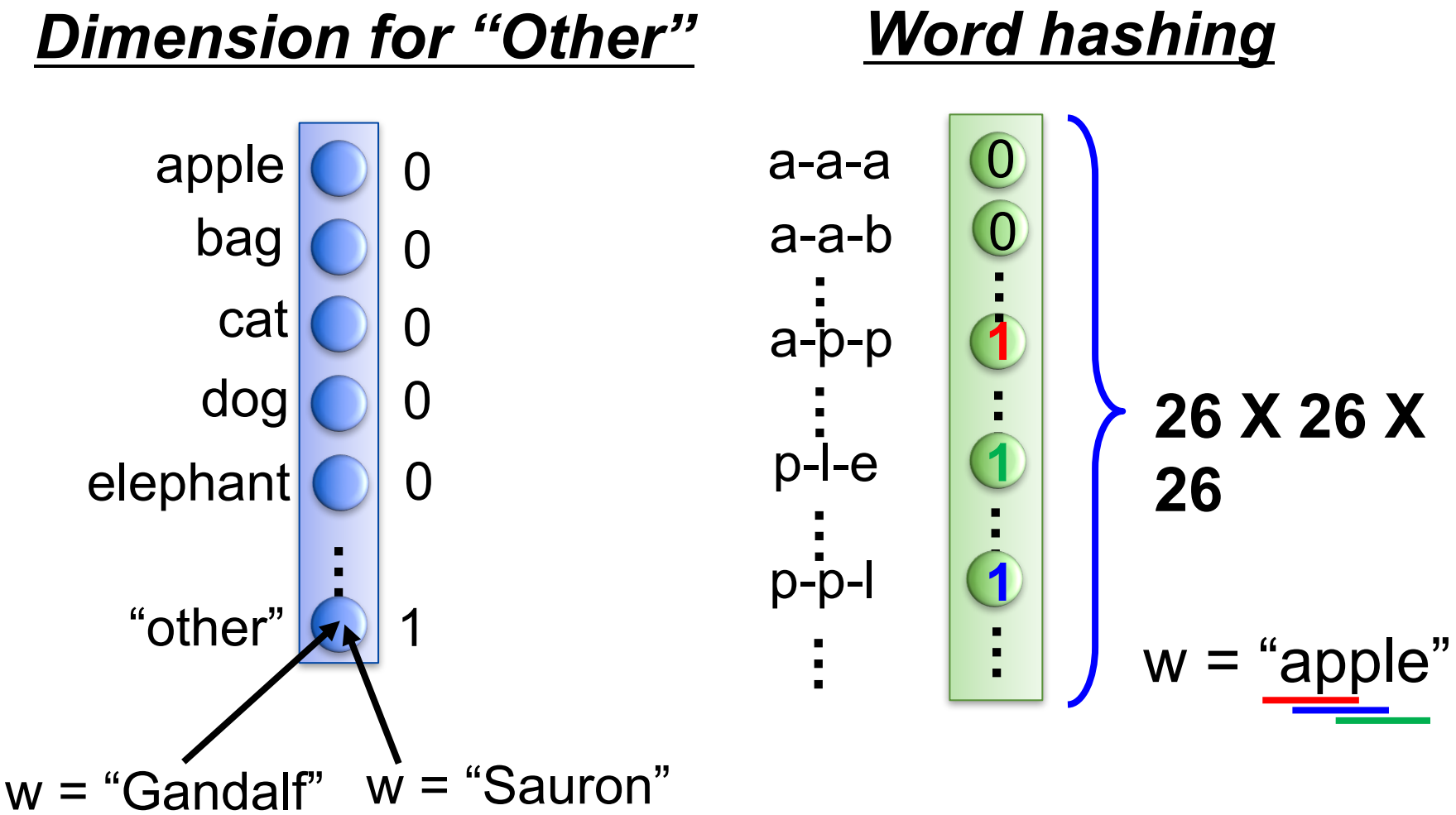
Recurrent Neural Network

- One hot encoding
  - The length of vector is lexicon size
  - Each dimension corresponds to a word in the lexicon
  - The dimension for the word is 1, and others are 0

lexicon = {apple, bag, cat, dog, elephant}

Word	D_1	D_2	D_3	D_4	D_5
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

Recurrent Neural Network



Recurrent Neural Network

- Output
  - Probability distribution that the sentence belongs to negative or positive.

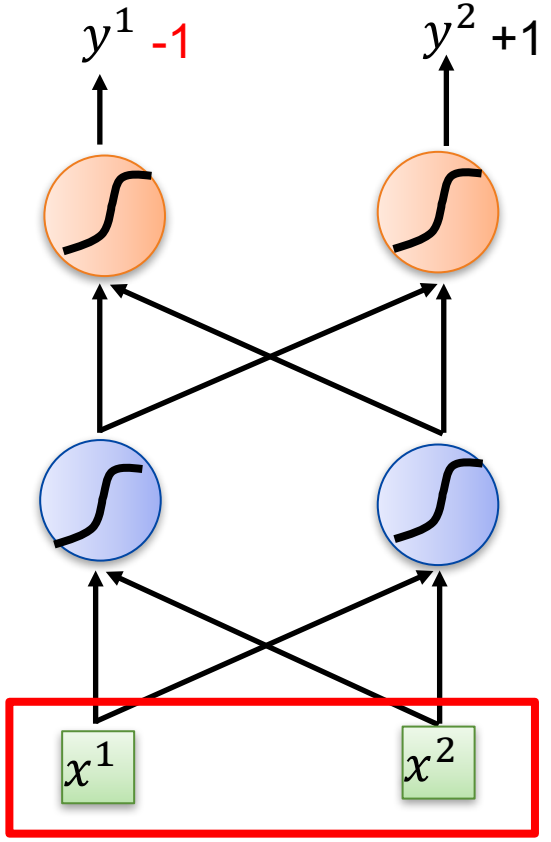
Complicated? Yes, and a little slow, too.

$y^1 = 0.9$
$y^2 = 0.1$

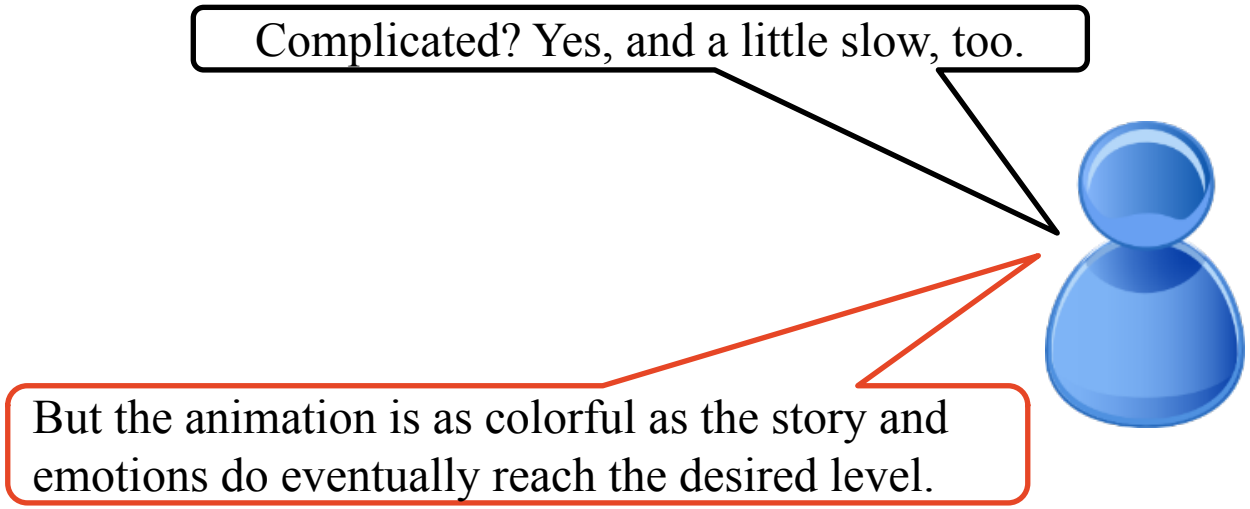
But the animation is as colorful as the story and emotions do eventually reach the desired level.

$y^1 = 0.3$
$y^2 = 0.7$

Complicated? Yes, and a little slow, too.



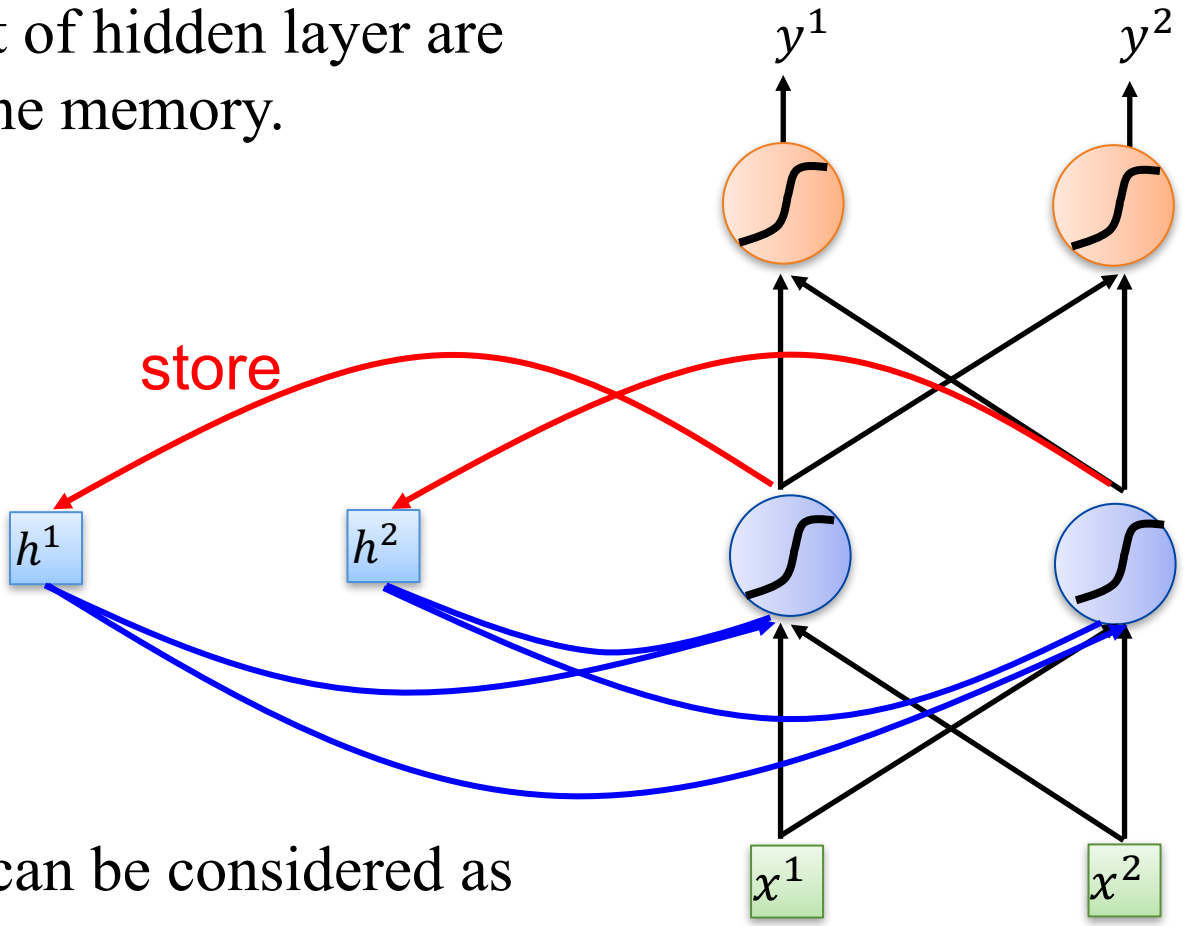
Recurrent Neural Network



Negative OR Positive → Need Memory

Recurrent Neural Network

The output of hidden layer are stored in the memory.

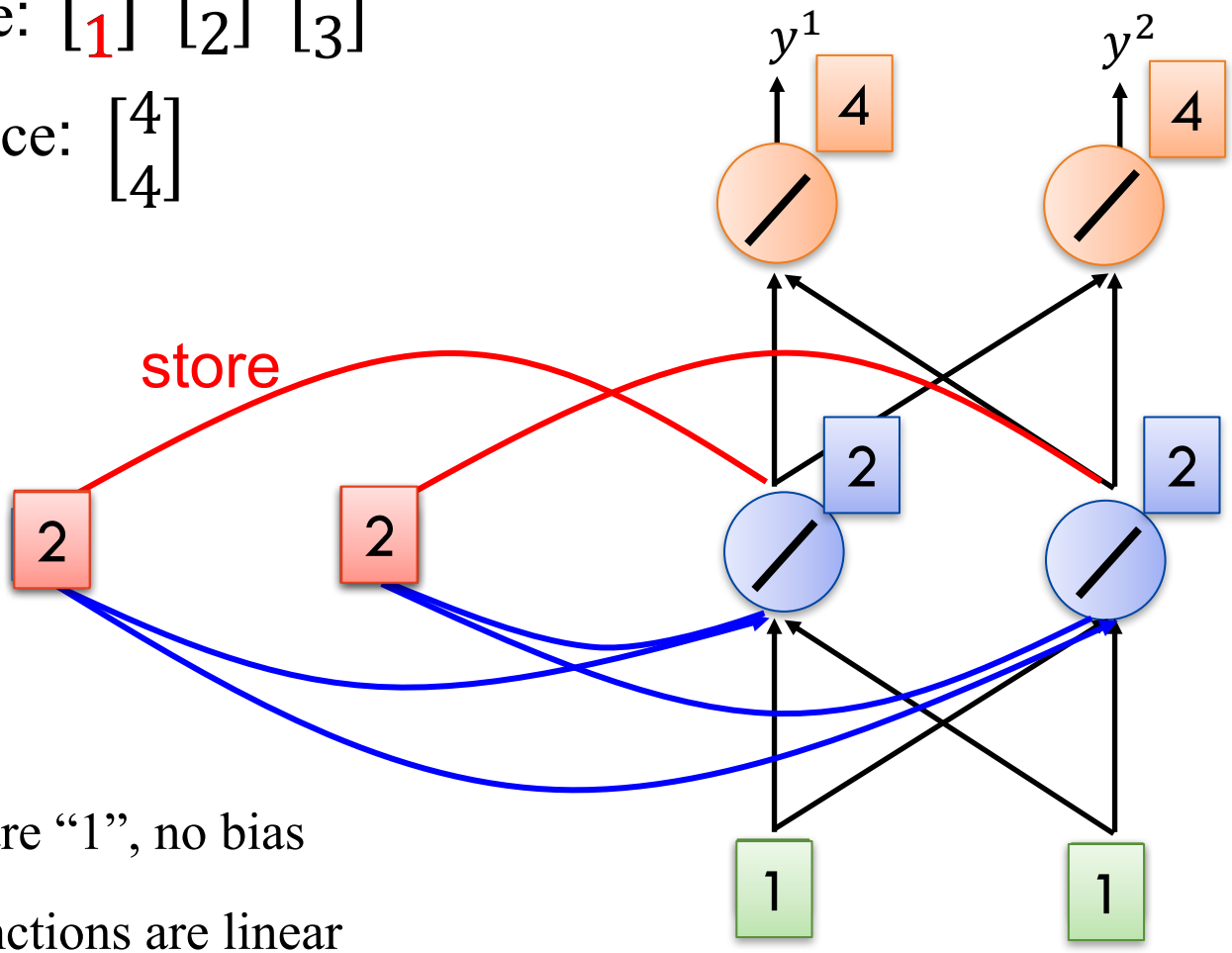


Memory can be considered as another input.

Recurrent Neural Network

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ $\begin{bmatrix}3\\3\end{bmatrix}$

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



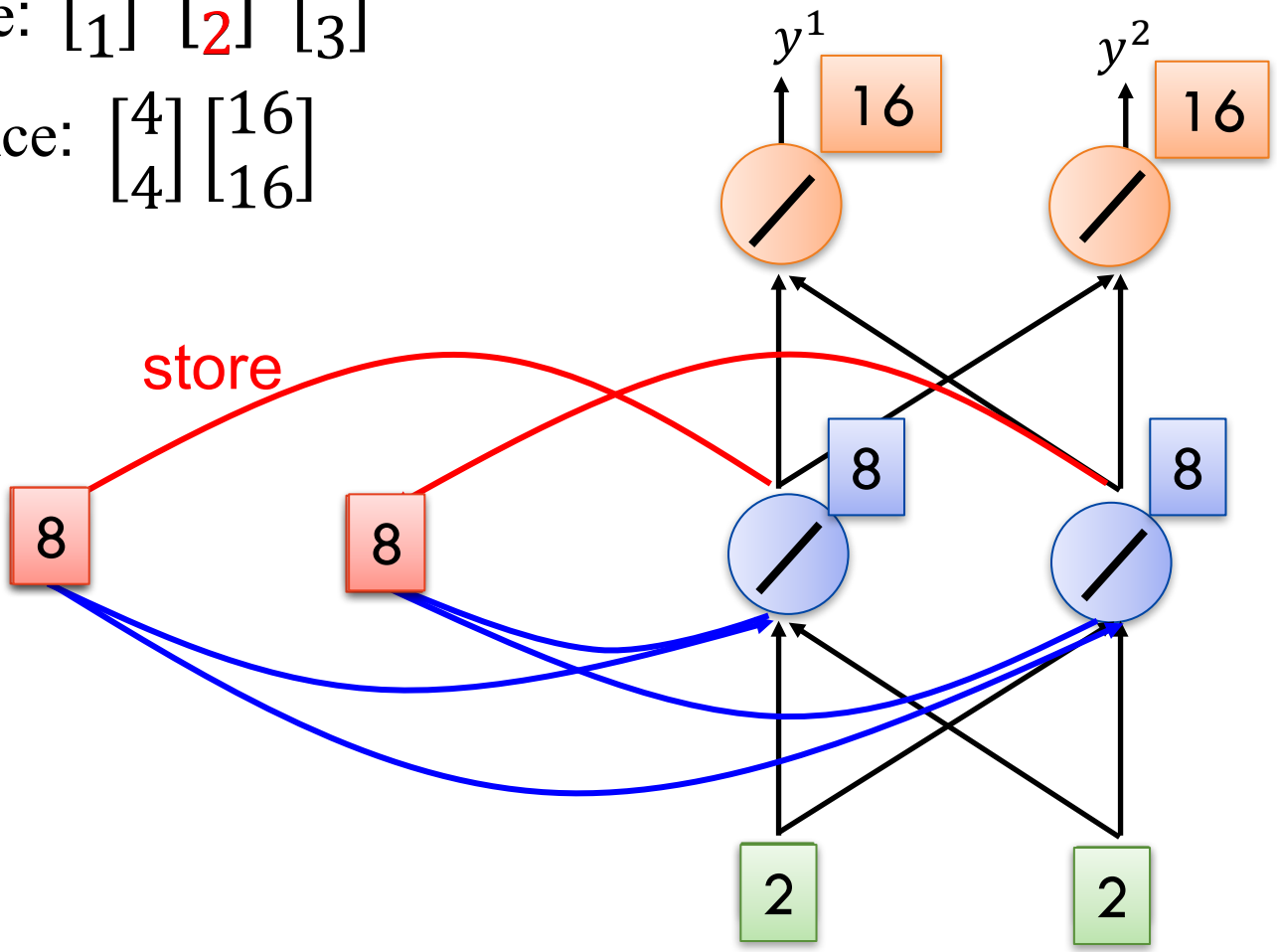
All the weights are "1", no bias

All activation functions are linear

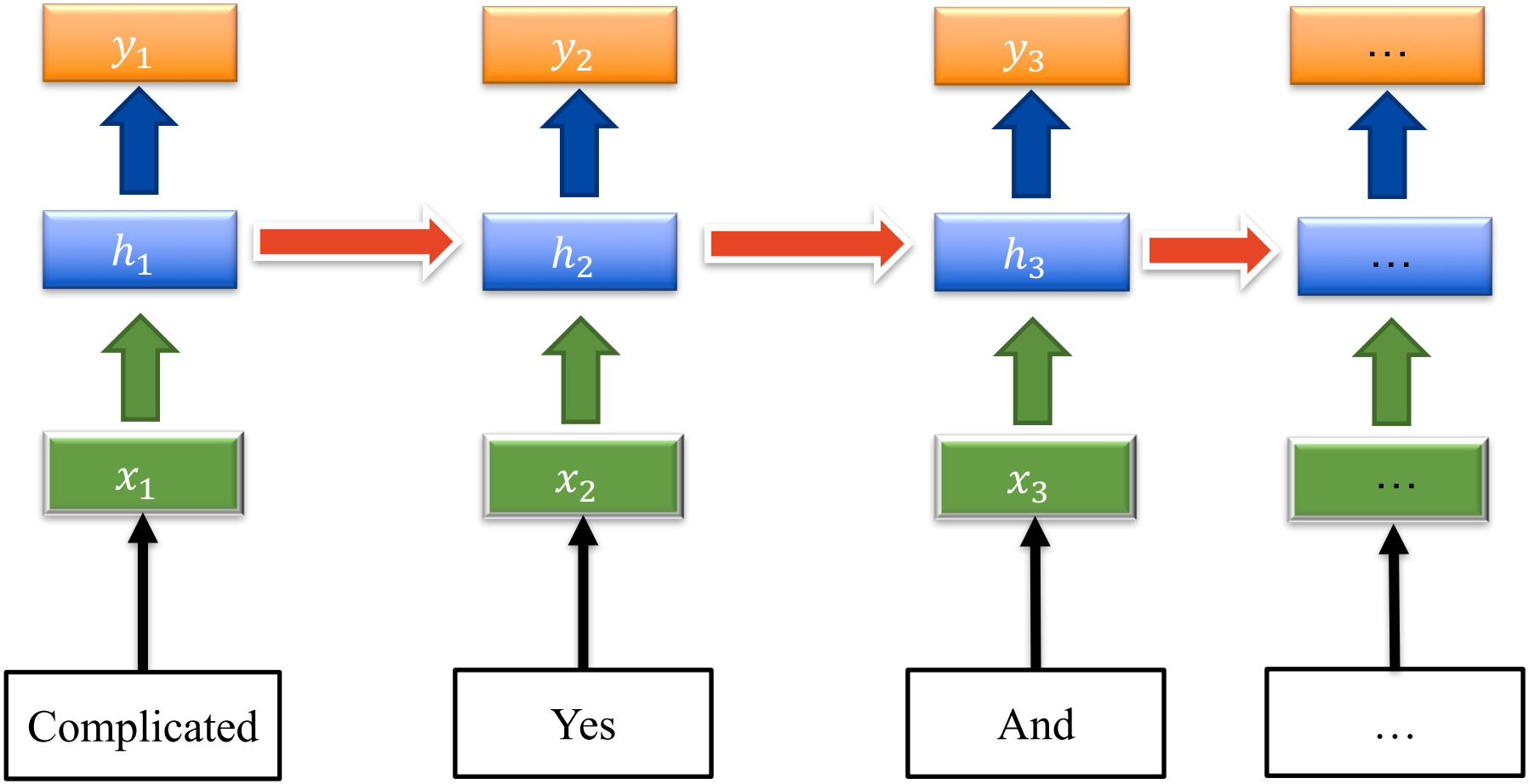
Recurrent Neural Network

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ $\begin{bmatrix}3\\3\end{bmatrix}$

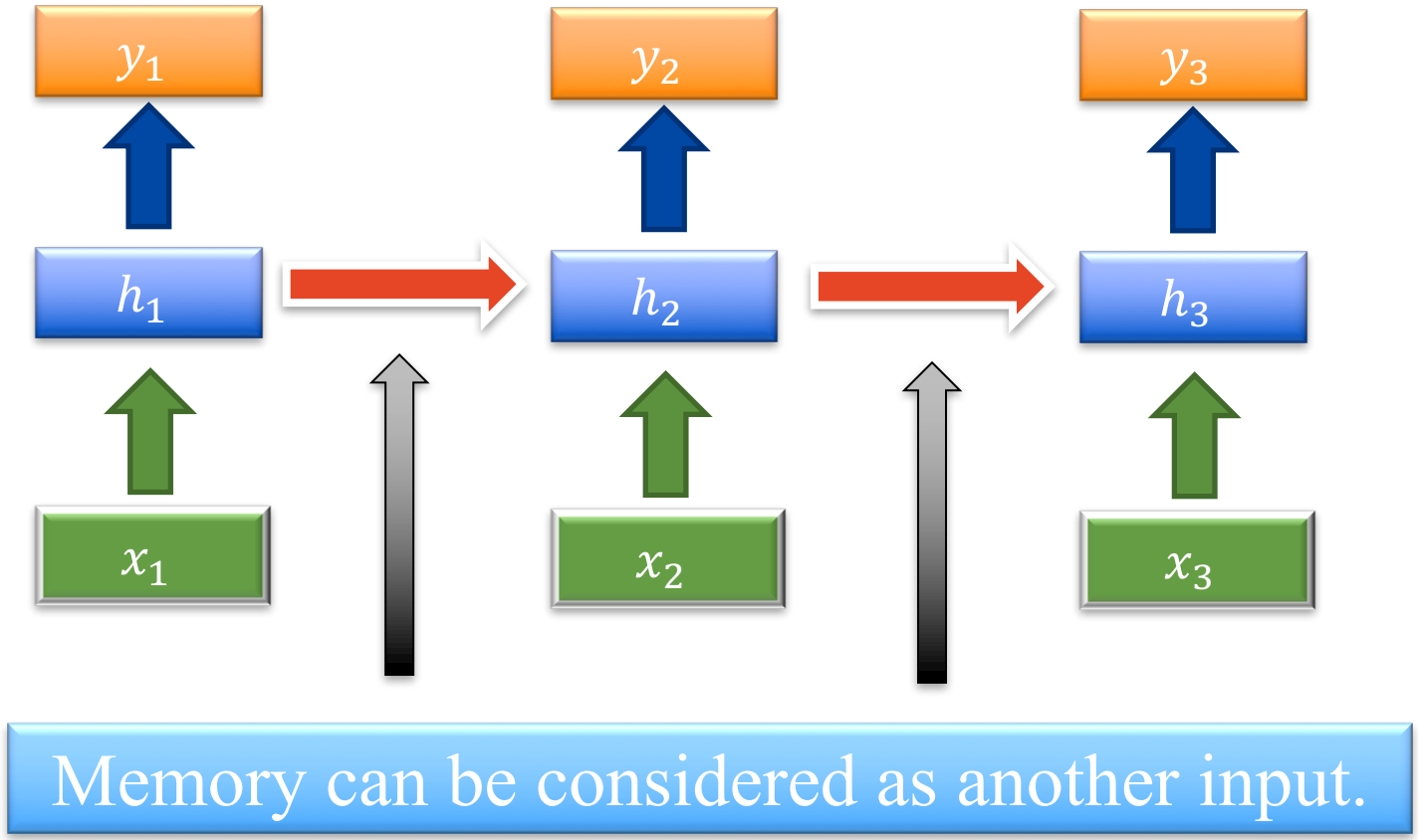
Output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$ $\begin{bmatrix}16\\16\end{bmatrix}$



Recurrent Neural Network

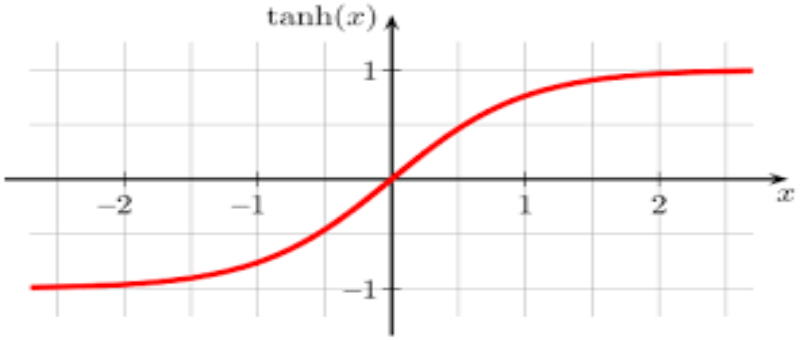


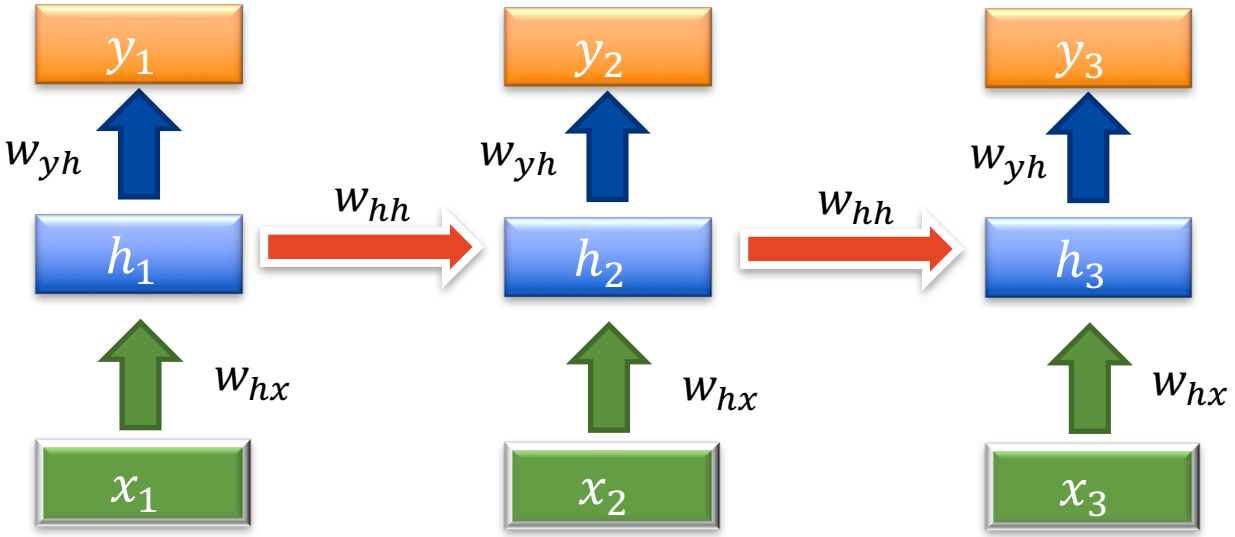
Recurrent Neural Network



Recurrent Neural Network

- Formally
  - x_t is the input
  - h_t is the hidden state
  - y_t is the output
  - $h_0 = \vec{0}$
  - $h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
  - $y_t = w_{yh}h_t$





Recurrent Neural Network

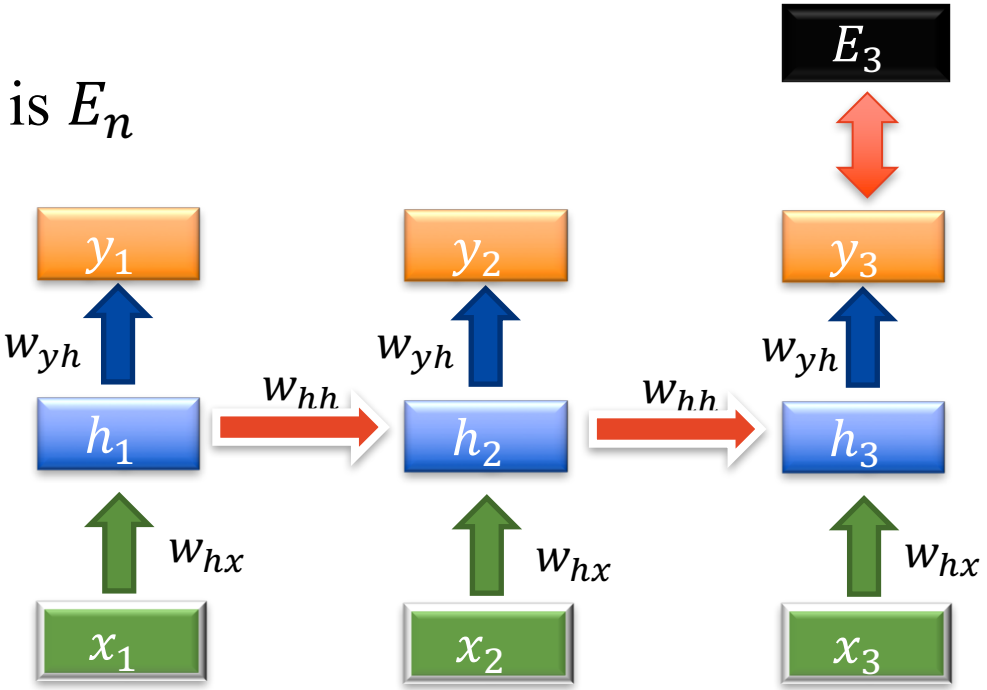
- Vanishing gradient problem
  - $h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
  - $y_t = w_{yh}h_t$
  - For every step, its loss is E_n

$$\frac{\partial E_3}{\partial w_{hh}} = \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial h_3}\frac{\partial h_3}{\partial w_{hh}}$$

$$h_3 = \tanh(w_{hh}h_2 + w_{hx}x_3 + b_h)$$

$$\frac{\partial h_3}{\partial w_{hh}} = (\tanh)'\left(h_2 + w_{hh}\frac{\partial h_2}{\partial w_{hh}}\right)$$

$$\frac{\partial h_2}{\partial w_{hh}} = (\tanh)'\left(h_1 + w_{hh}\frac{\partial h_1}{\partial w_{hh}}\right)$$



$$(\tanh)'w_{hh}\ (\tanh)'w_{hh}\ (\tanh)'w_{hh}\ \cdots$$

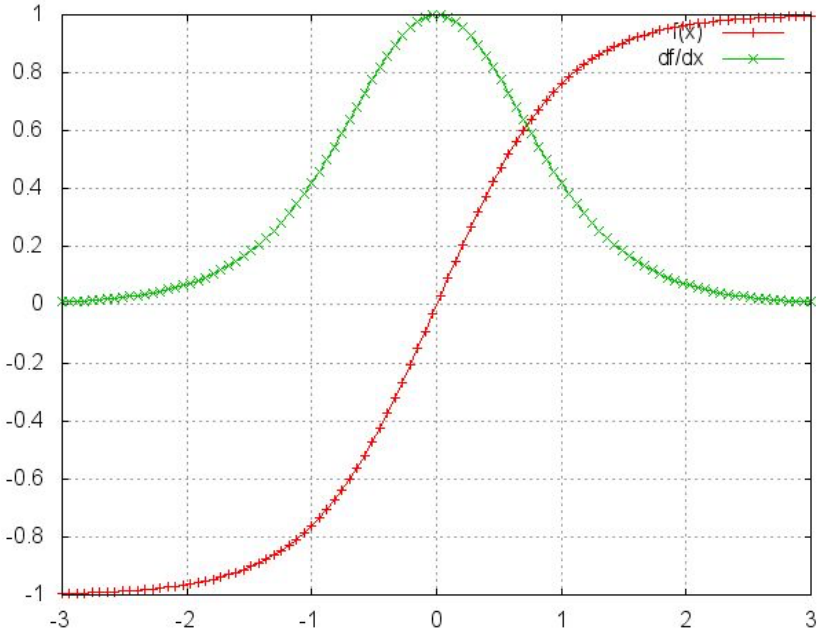
Recurrent Neural Network

- Vanishing gradient problem
  - $h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
  - $y_t = w_{yh}h_t$
  - For every step, its loss is E_n

$$0.9^{1000} \approx 0$$
$$1.01^{1000} \approx 21000$$

$$\frac{\partial E_3}{\partial w_{hh}} = \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial h_3}\frac{\partial h_3}{\partial w_{hh}}$$

$$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots$$



$(\tanh)' \leq 1$

If $0 < w_{hh} < 1$, Vanishing Gradients

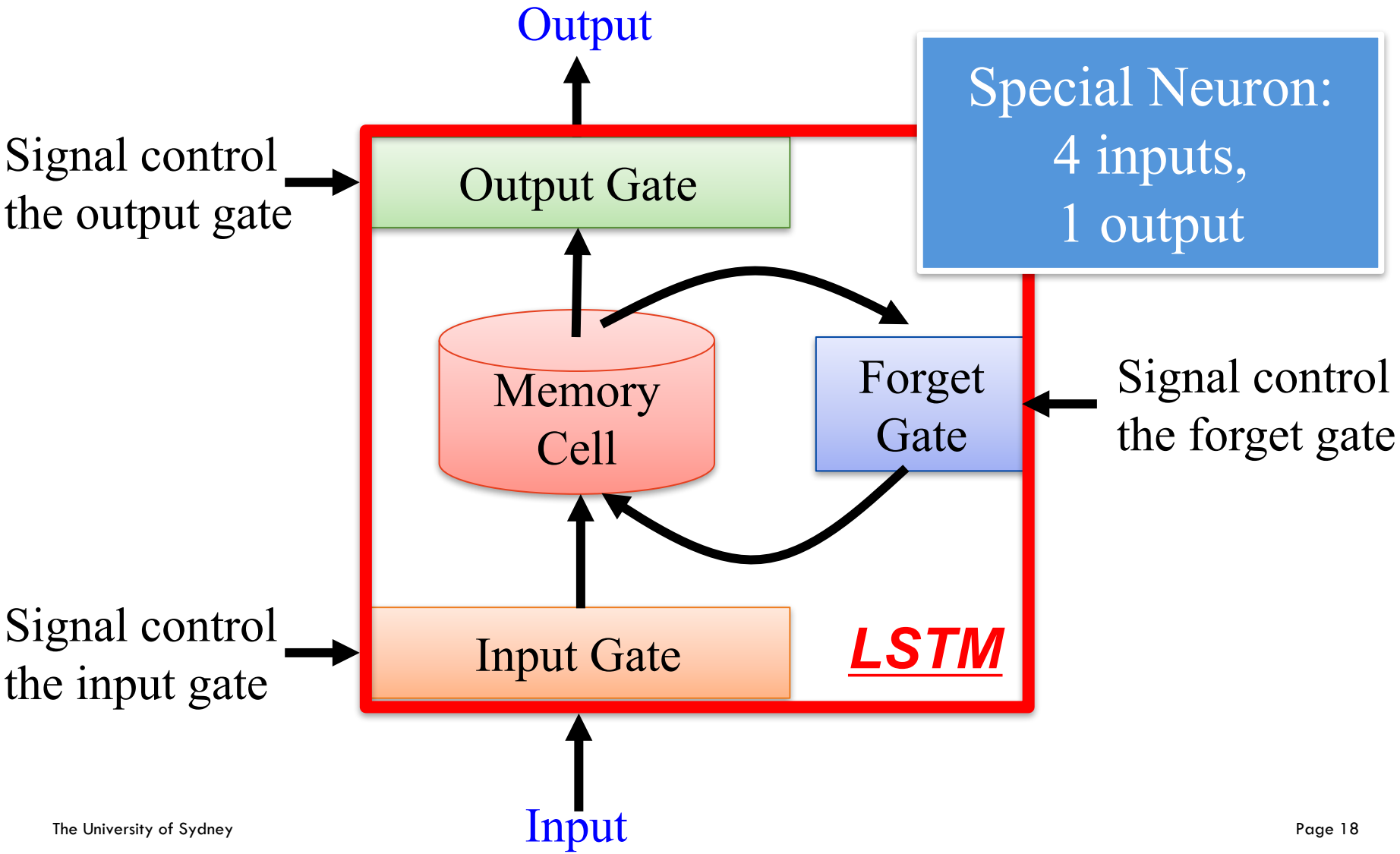
$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to 0$

If w_{hh} is larger, Exploding Gradients

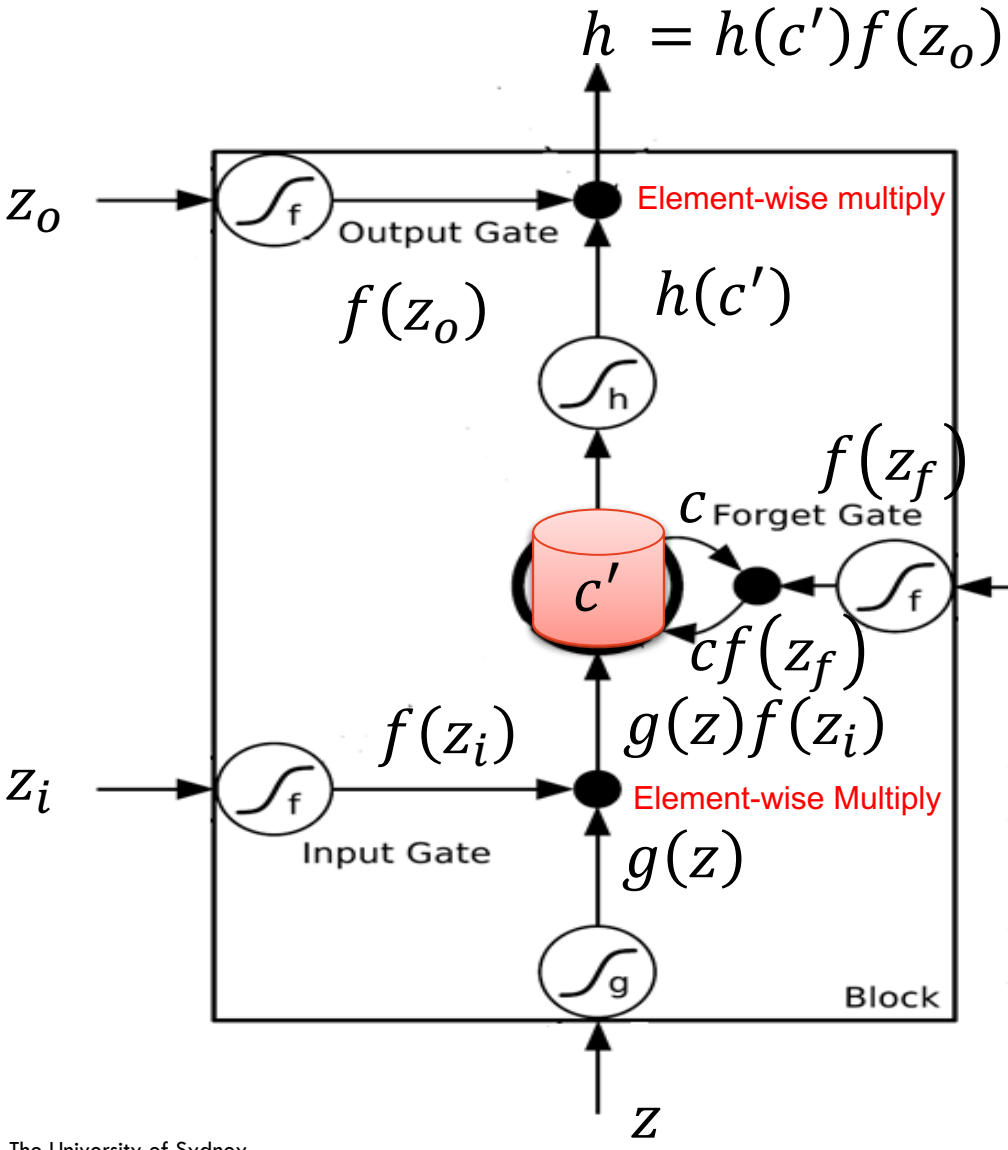
$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to \infty$

Long Short-Term Memory

Long Short-Term Memory

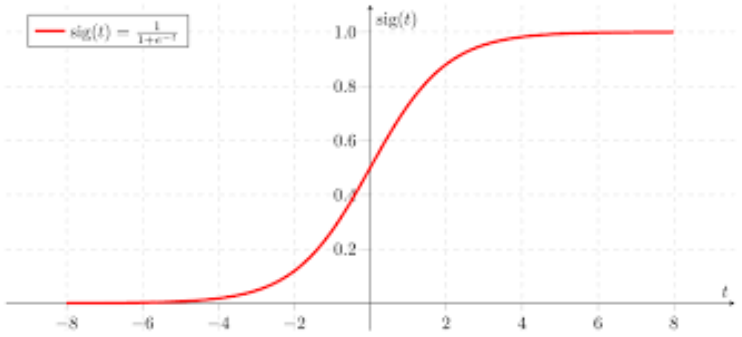


Long Short-Term Memory

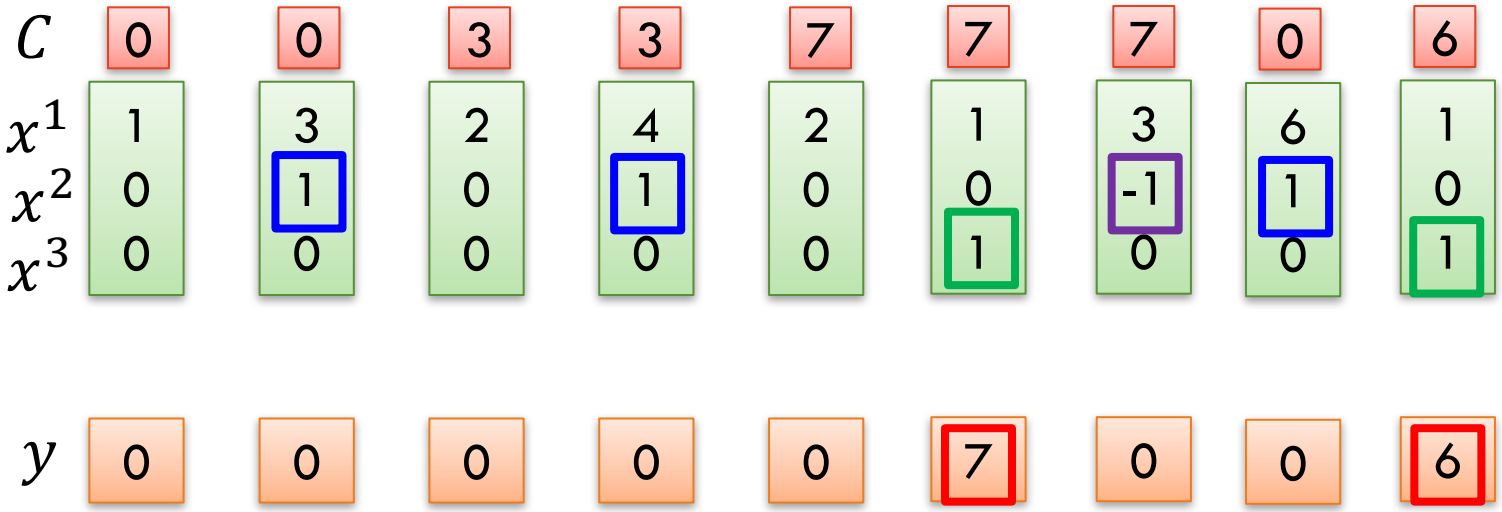


f is usually a sigmoid function
Value From 0 to 1
Mimic open and close gate

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



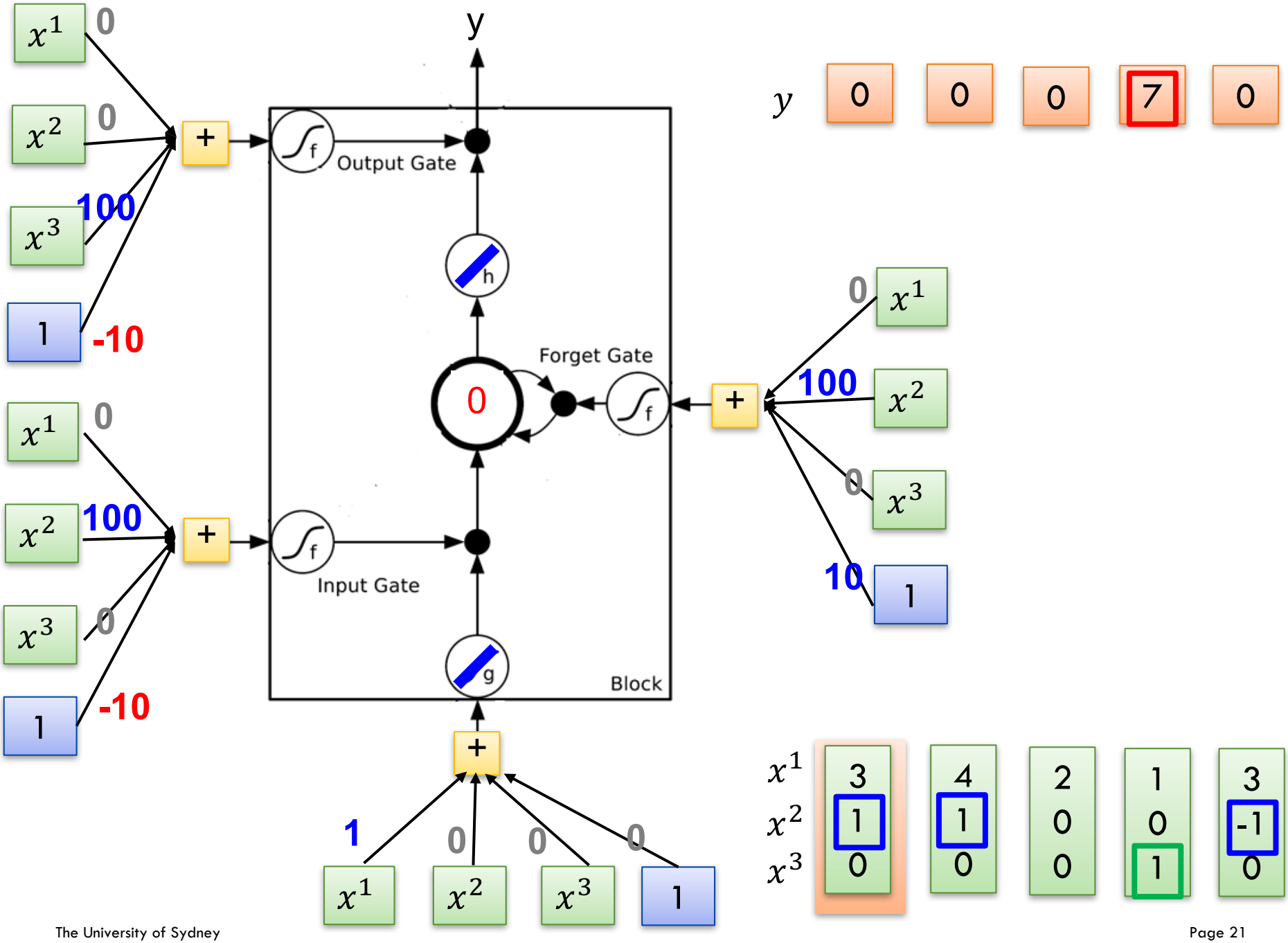
Long Short-Term Memory

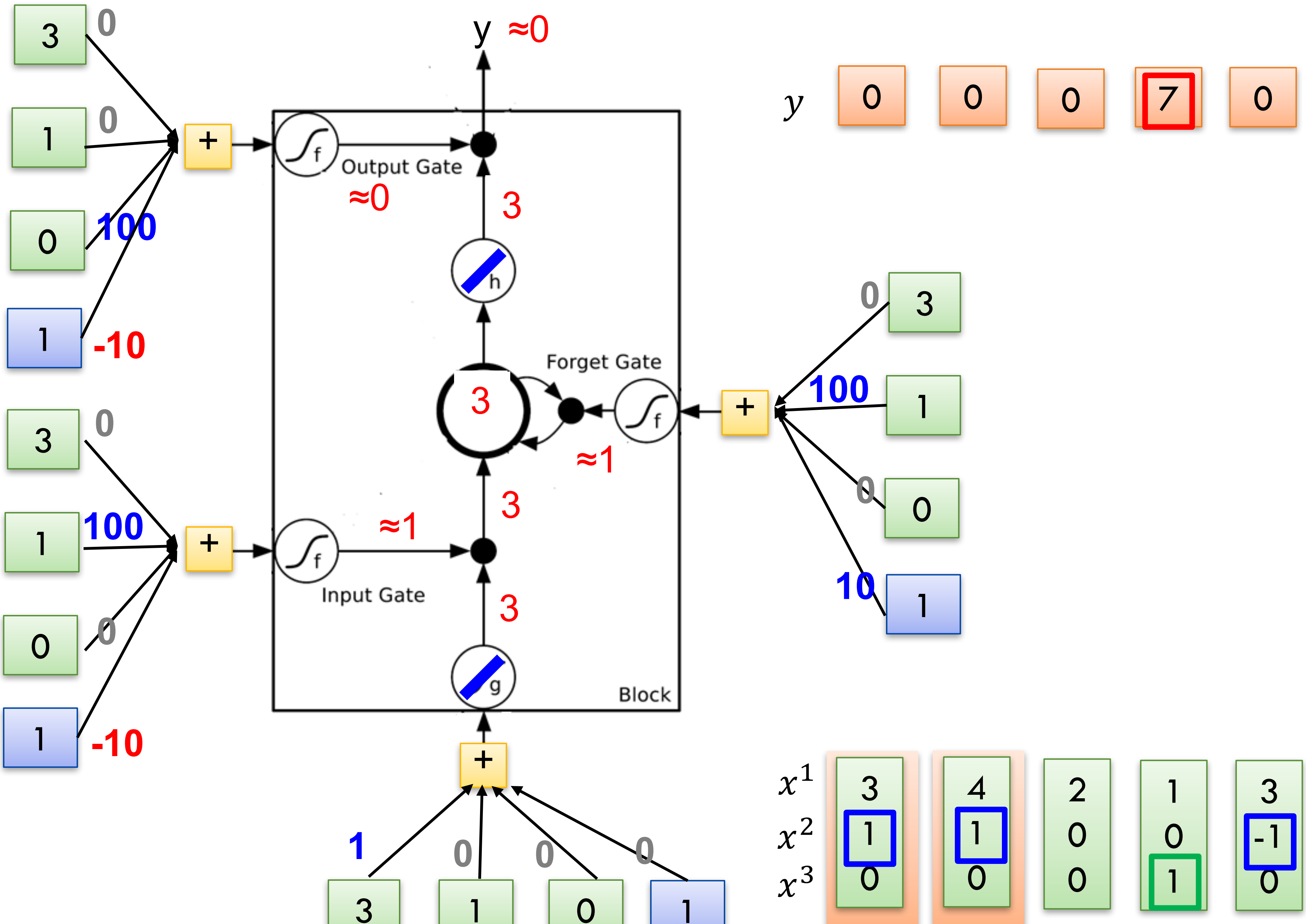


When $x^2 = 1$, add the numbers of x_1 into the memory

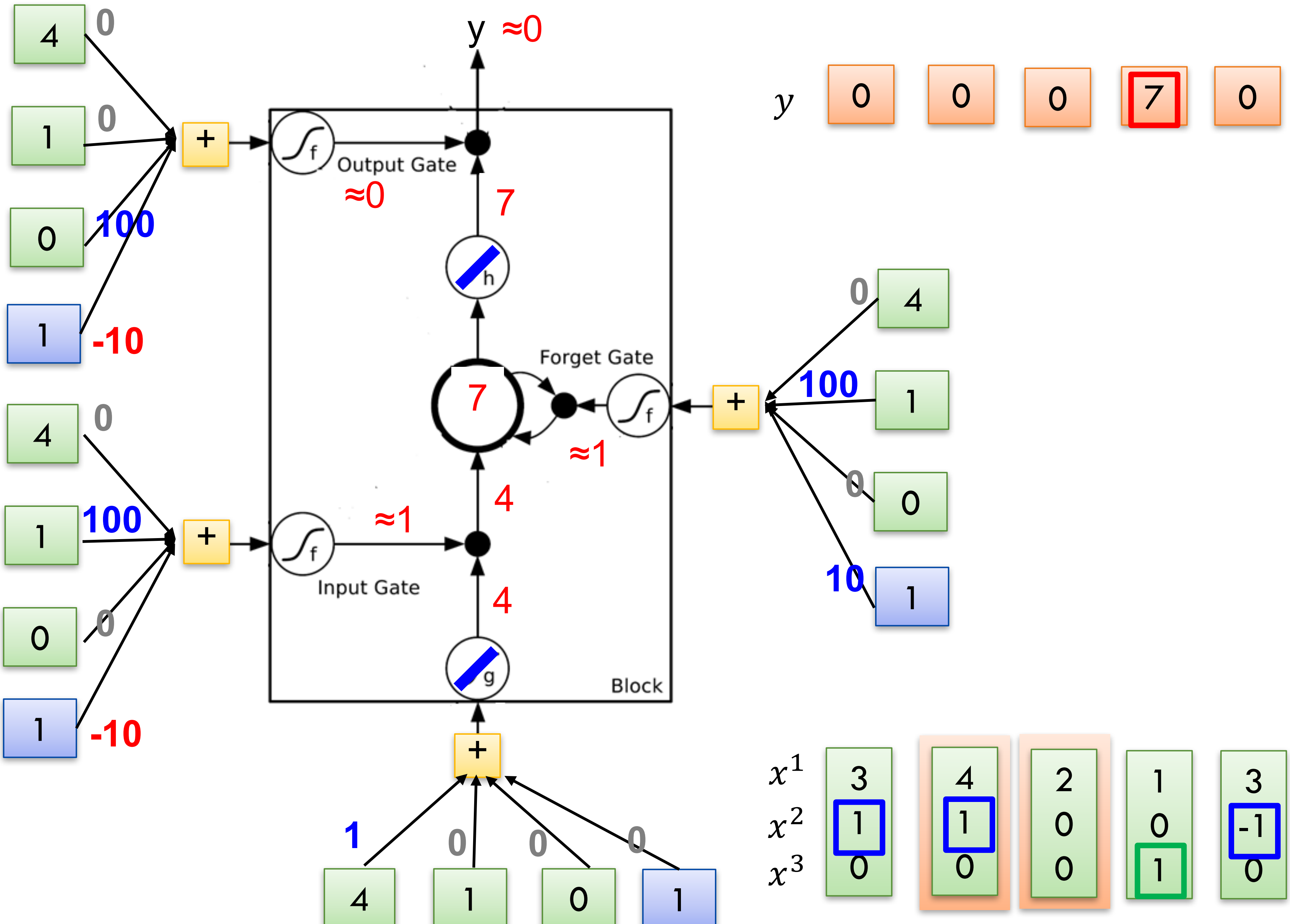
When $x^2 = -1$, reset the memory

When $x^3 = 1$, output the number in the memory.

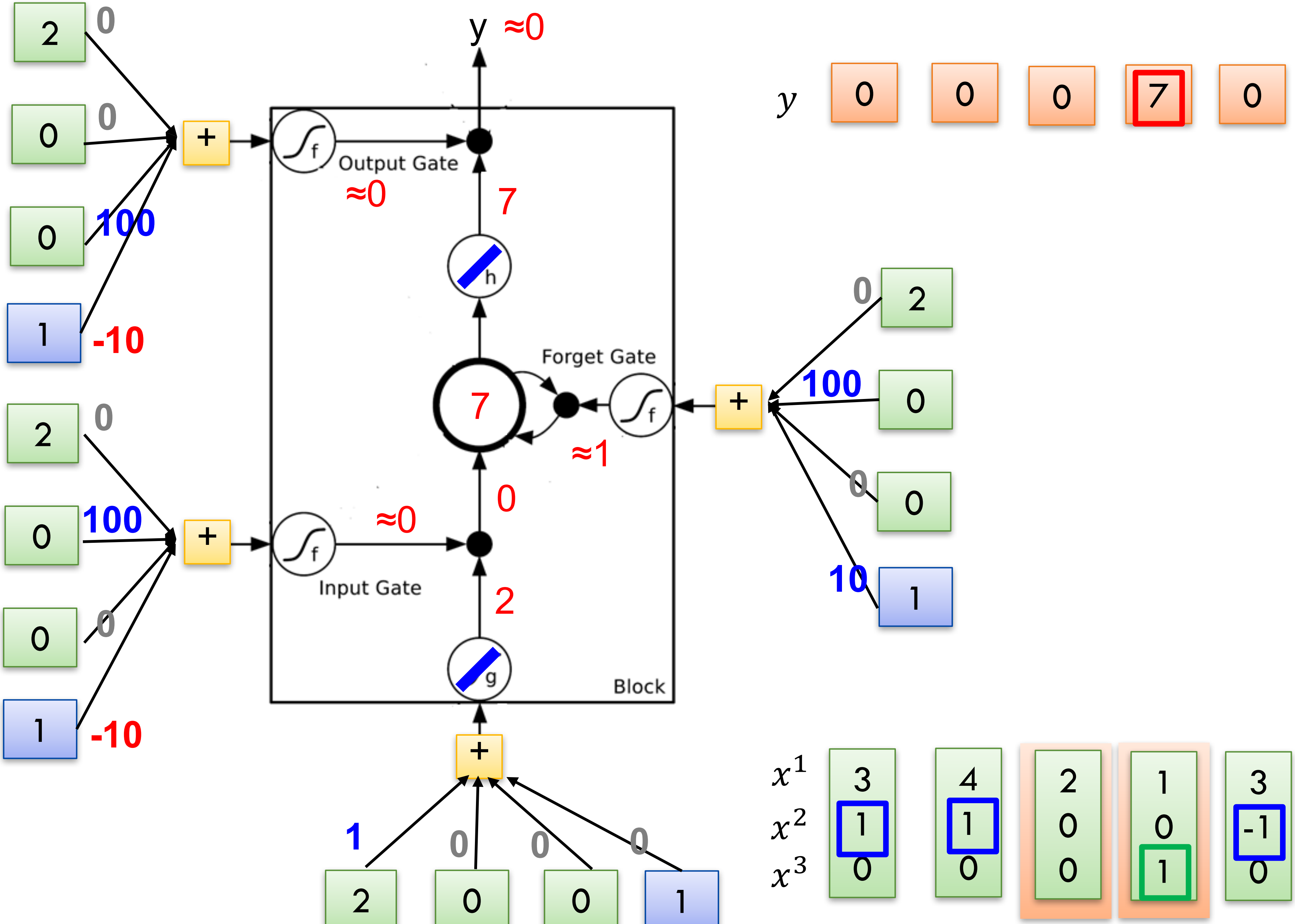

y
Output Gate
Forget Gate
Input Gate
Block
0
100
-10
1
10
x^1
x^2
x^3
y 0 0 0 7 0
x^1 3 1 0 4 1 0 2 0 0 1 0 1 3 -1 0
x^2
x^3

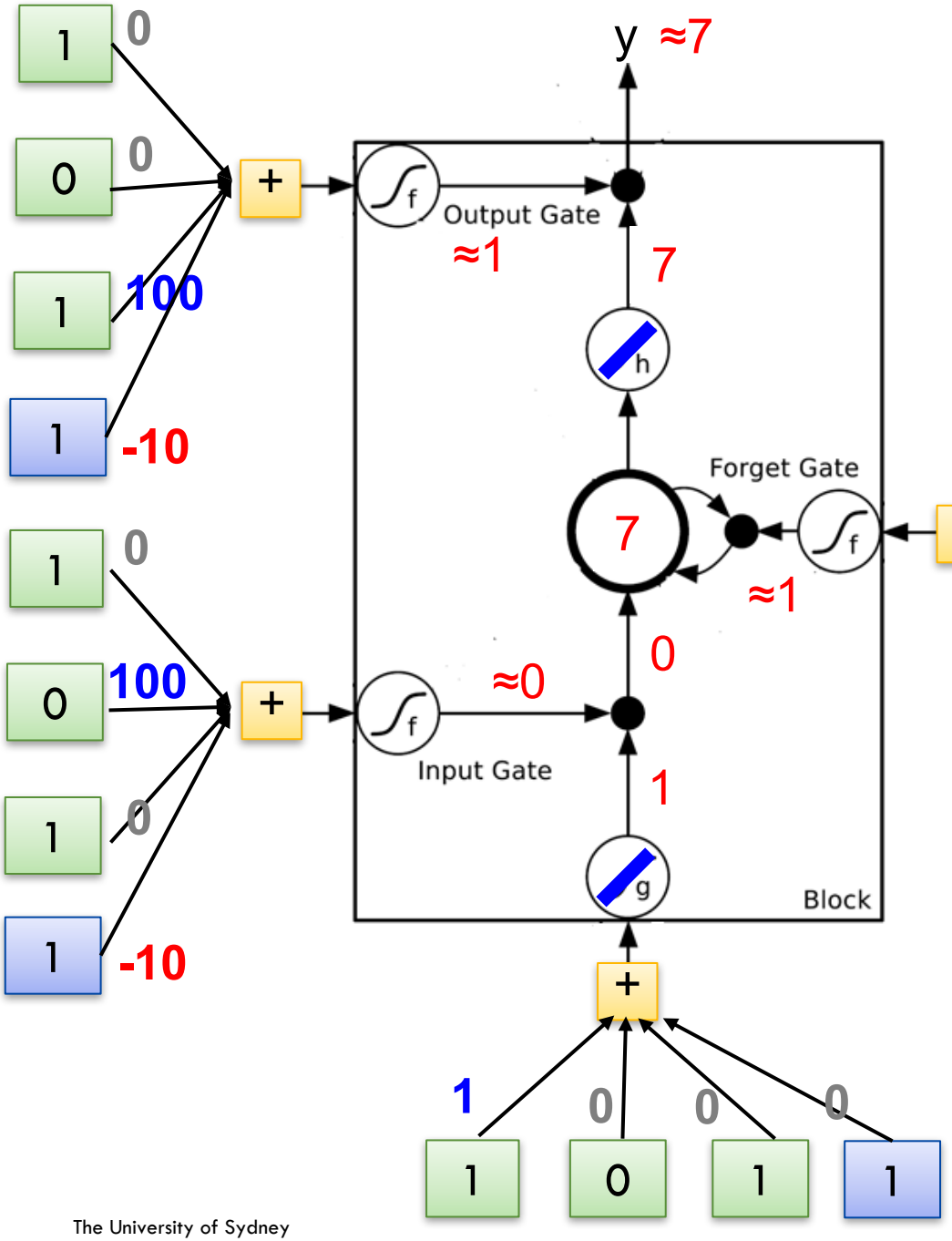


y ≈0
Output Gate
≈0
3
h
Forget Gate
3
≈1
3
Input Gate
≈1
3
g
Block
f
+
3
1
0
1
0
0
100
-10
100
0
10
1
0
0
0
y
0 0 0 7 0
x^1
x^2
x^3
3 1 0
4 1 0
2 0 0
1 0 1
3 -1 0



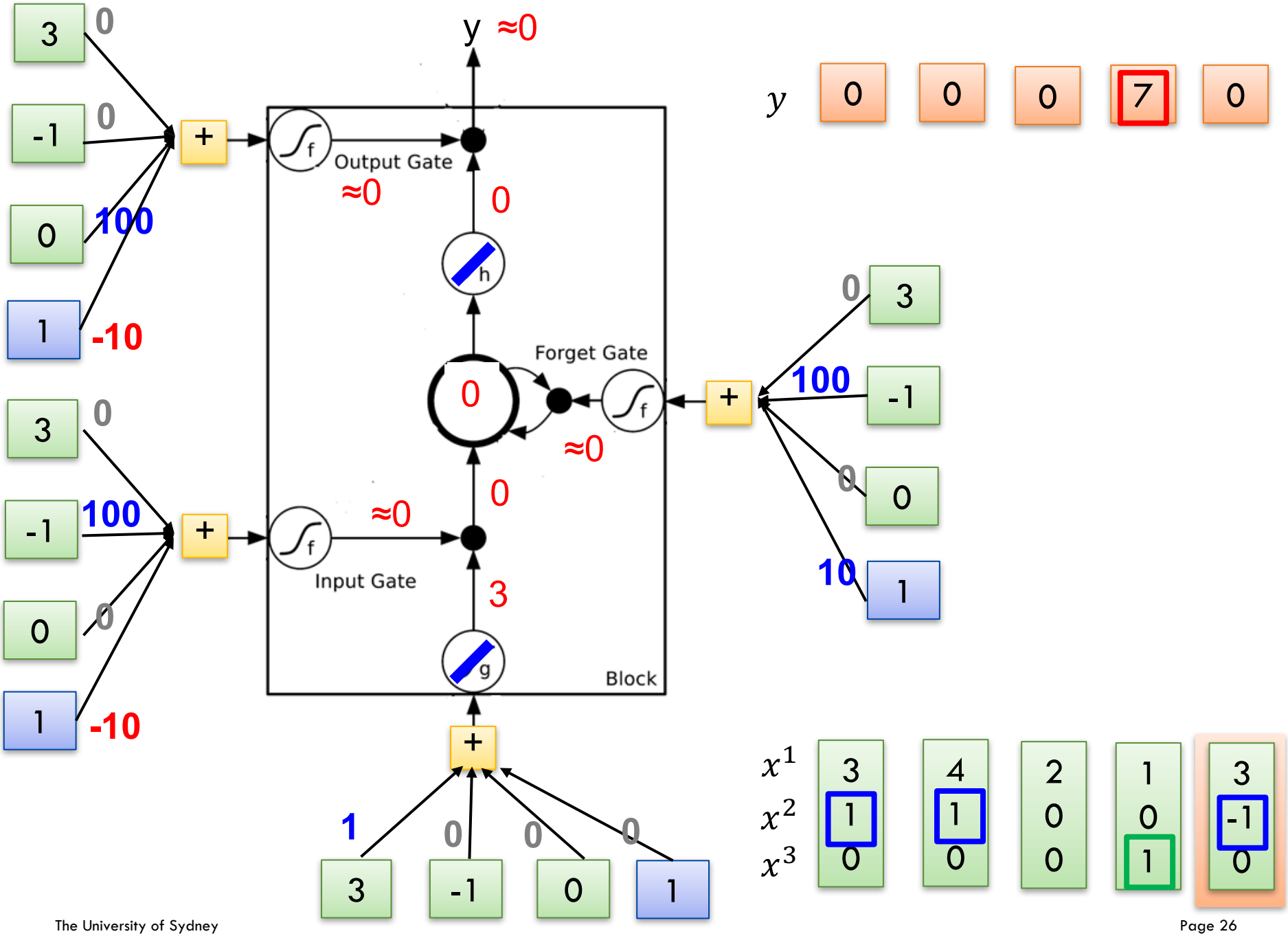
y ≈0
Output Gate
≈0
7
h
7
Forget Gate
≈1
4
Input Gate
≈1
4
g
Block
0 4
100 1
0 0
10 1
4 0
1 0
0 100
1 -10
4 0
1 100
0 0
1 -10
1 0 0 0
4 1 0 1
y
0 0 0 7 0
x^1
x^2
x^3
3 1 0
4 1 0
2 0 0
1 0 1
3 -1 0


y ≈0
Output Gate
≈0
7
h
Forget Gate
7
≈1
0
≈0
Input Gate
2
g
Block
2
0
0
0
0
100
1
-10
2
0
0
100
0
0
1
-10
0
2
100
0
0
0
10
1
1
2
0
0
0
0
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

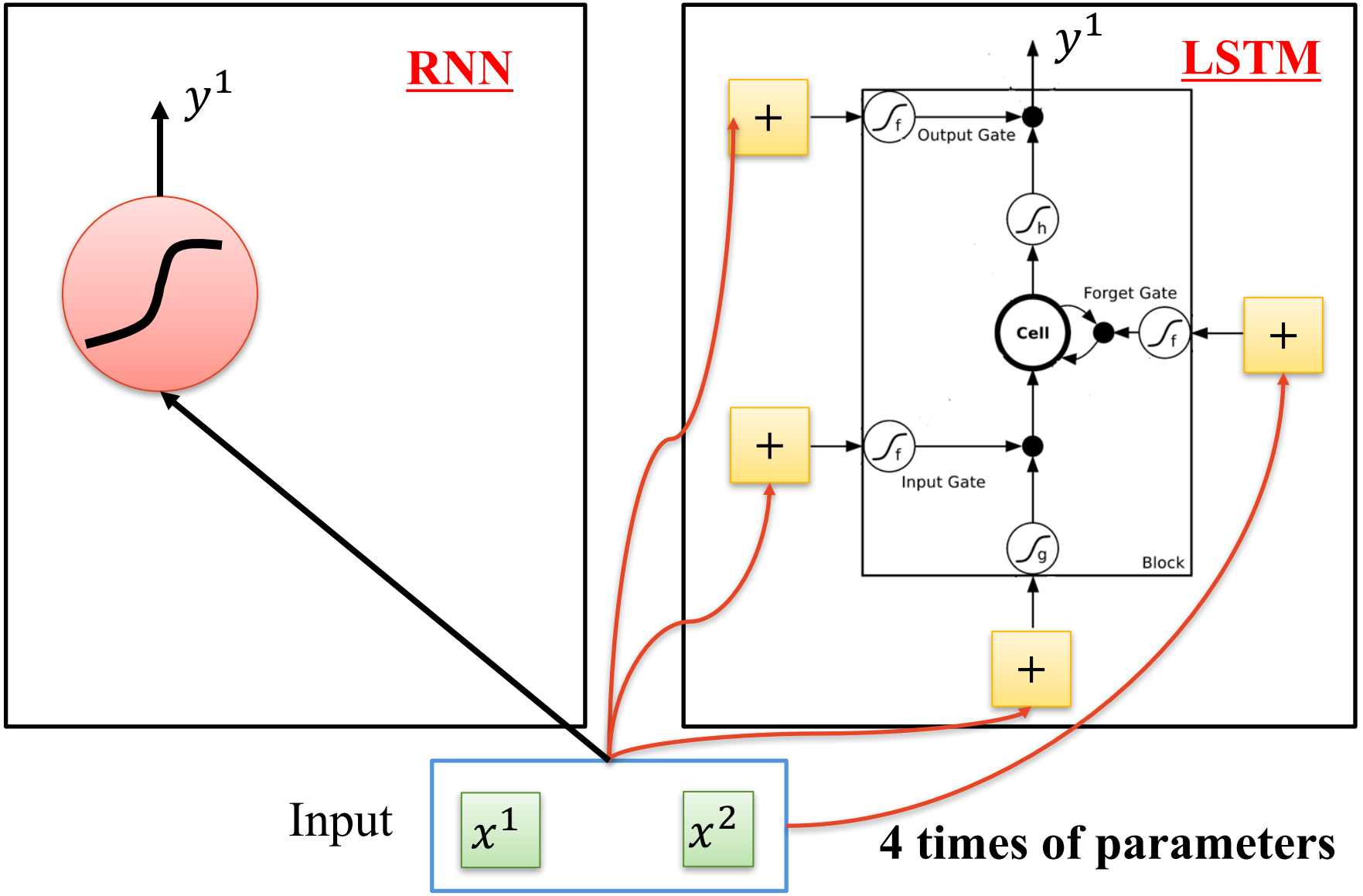


y	0	0	0	7	0

x^1	3	4	2	1	3
x^2	1	1	0	0	-1
x^3	0	0	0	1	0

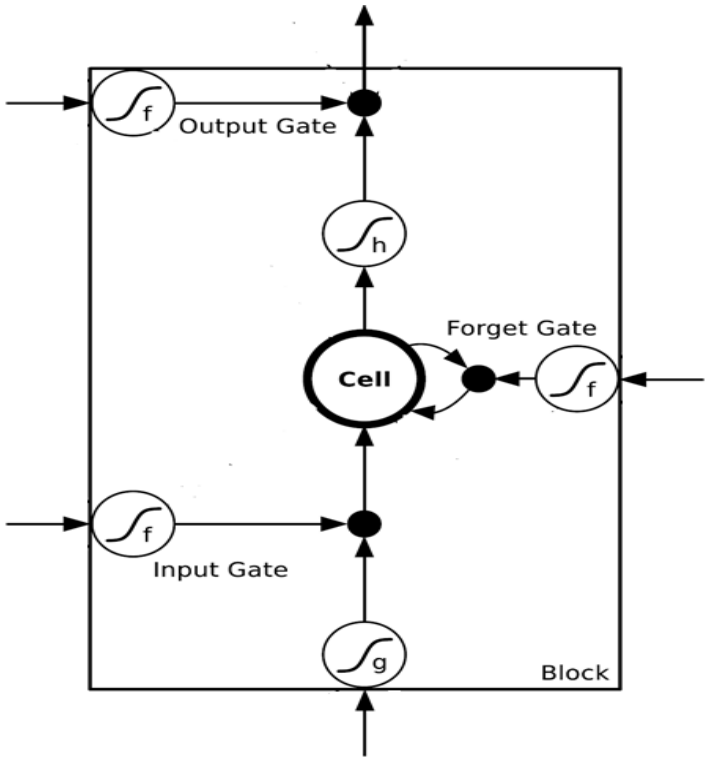

y ≈0
Output Gate
≈0
0
Forget Gate
0
≈0
Input Gate
≈0
0
3
Block
3
0
-1
100
0
0
1
-10
3
0
-1
100
0
0
1
-10
0
3
100
-1
0
0
10
1
1
0
0
0
3
-1
0
1
y 0 0 0 7 0
x^1 3 4 2 1 3
x^2 1 1 0 0 -1
x^3 0 0 0 1 0

Long Short-Term Memory



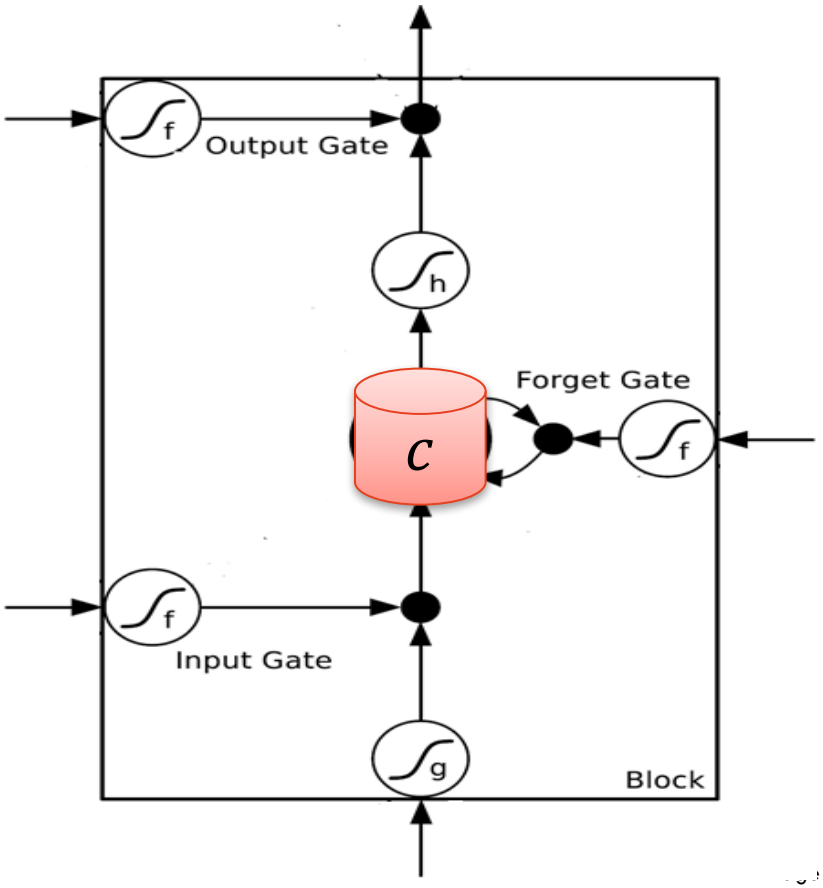
Long Short-Term Memory

- Forget gate
  - $f_t = \sigma(w_{fh}h_{t-1} + w_{fx}x_t + b_f)$
- Input gate
  - $i_t = \sigma(w_{ih}h_{t-1} + w_{ix}x_t + b_i)$
- Output gate
  - $o_t = \sigma(w_{oh}h_{t-1} + w_{ox}x_t + b_o)$



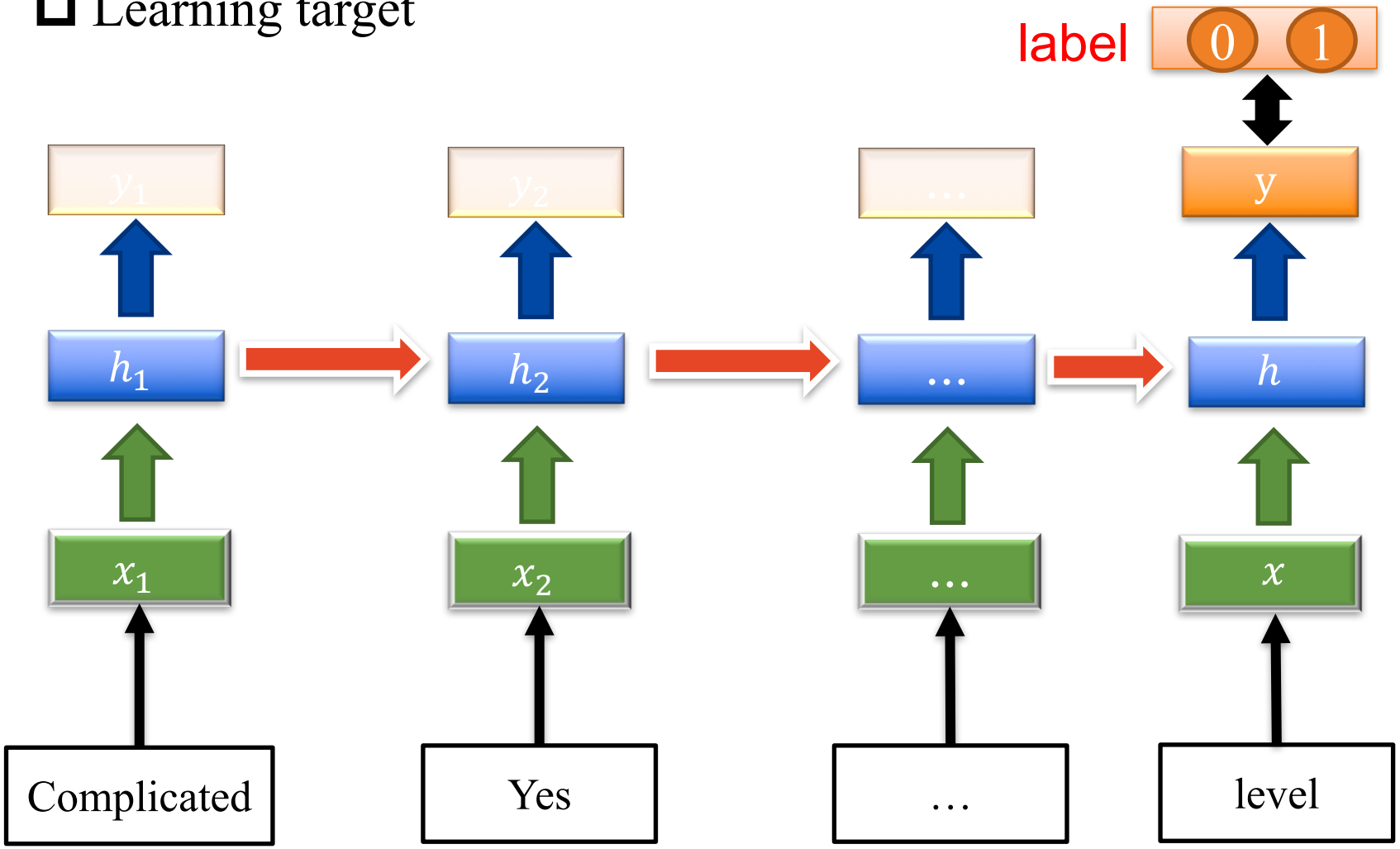
Long Short-Term Memory

- Candidate cell state
  - $\tilde{c}_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
- Update cell state
  - $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$
- Output state
  - $h_t = o_t * \tanh(c_t)$



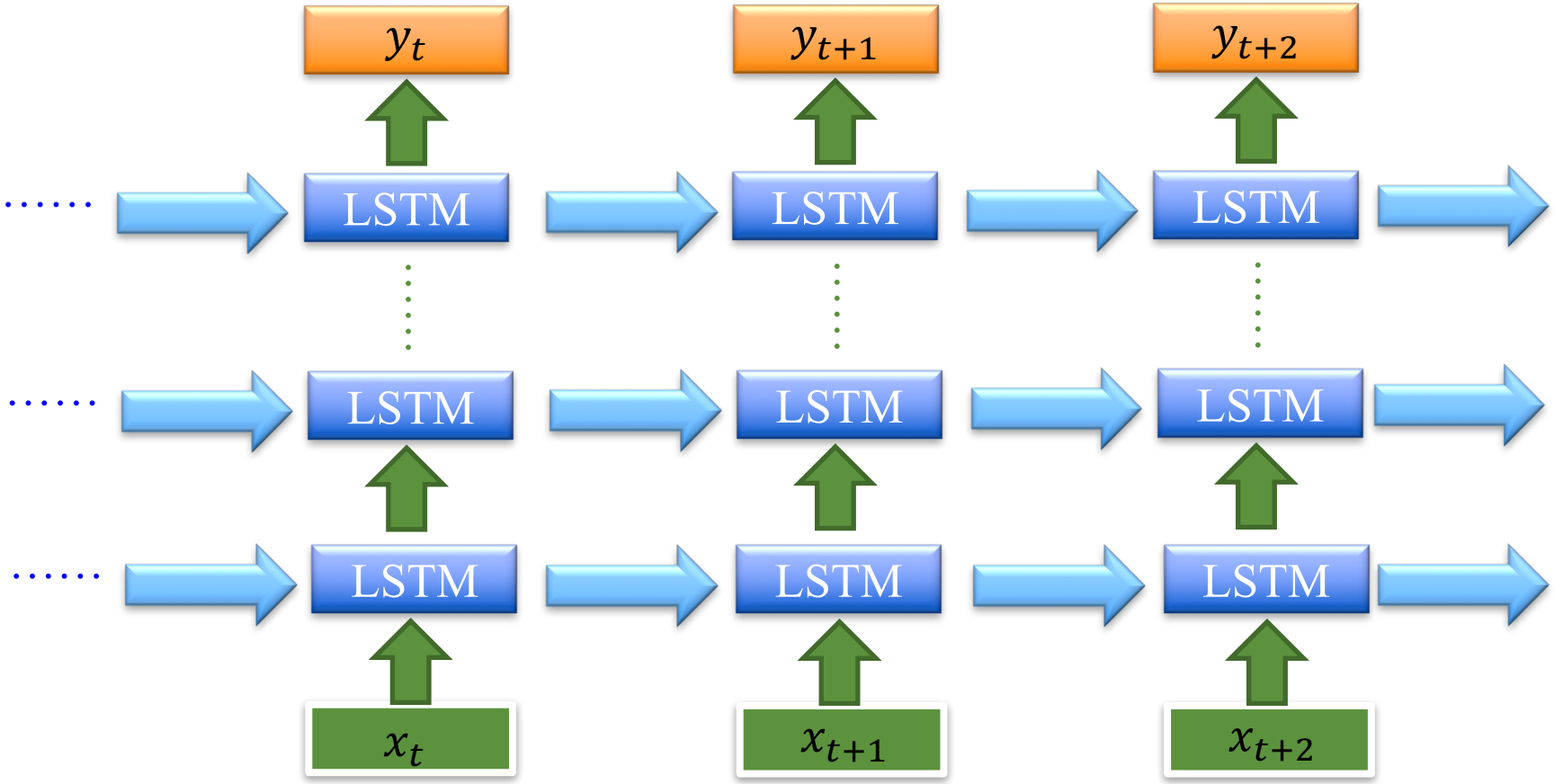
Long Short-Term Memory

- Learning target



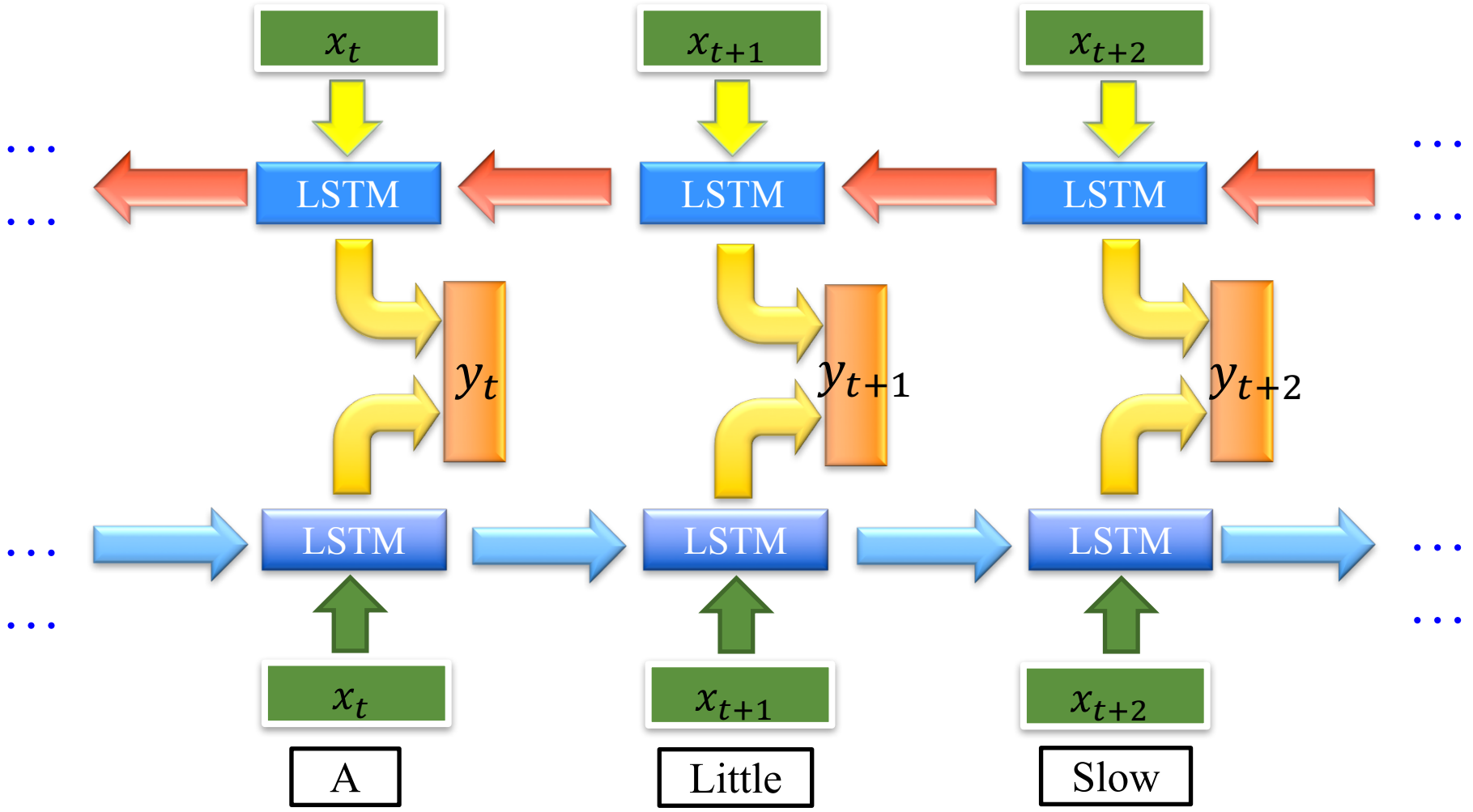
Long Short-Term Memory

- Stacked LSTM

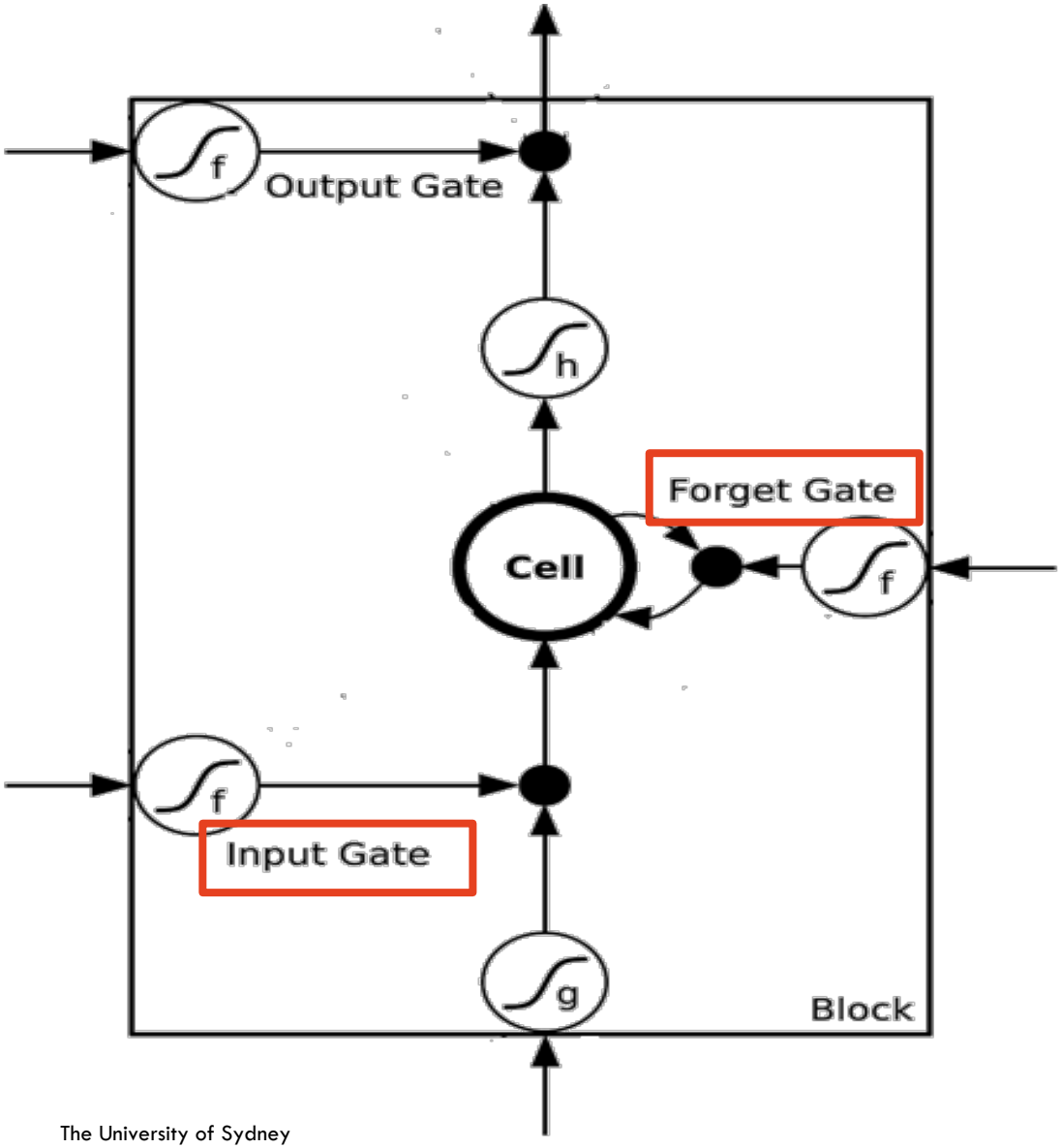


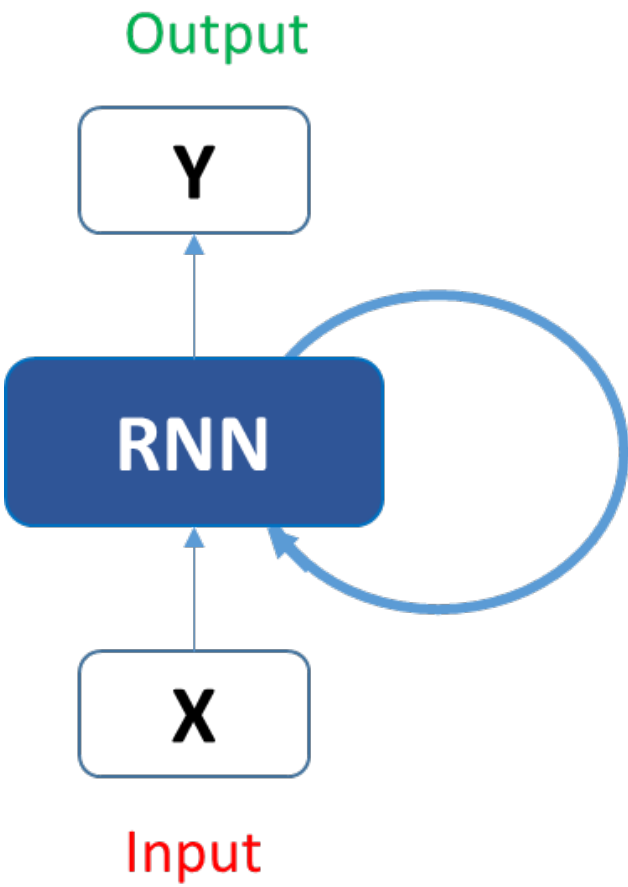
Long Short-Term Memory

- Bidirectional RNN



LSTM V.S. RNN

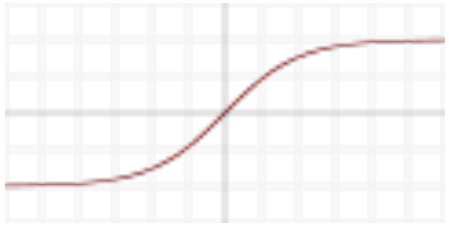

f
Output Gate
h
Forget Gate
Cell
f
f
Input Gate
g
Block


Output
Y
RNN
X
Input

Prevent vanishing gradient

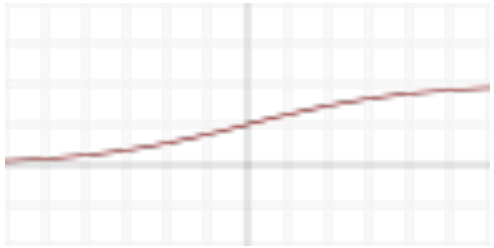
- RNN
  - $h_t = \tanh(w_{hh} h_{t-1} + w_{hx} x_t + b_h)$
- LSTM
  - $c_t = f_t * c_{t-1} + i_t * \widetilde{c}_t$

Back propagation from c_t to c_{t-1} only involves elementwise multiplication with f_t, no matrix multiplication with w_{hh} or tanh.



$f(x) = tanh(x)$

$f'(x) = 1 - f(x) \leq 1$



$f(x) = \sigma(x)$

$f'(x) = f(x)\big(1 - f(x)\big).$
≤ 0.25

Thank you !