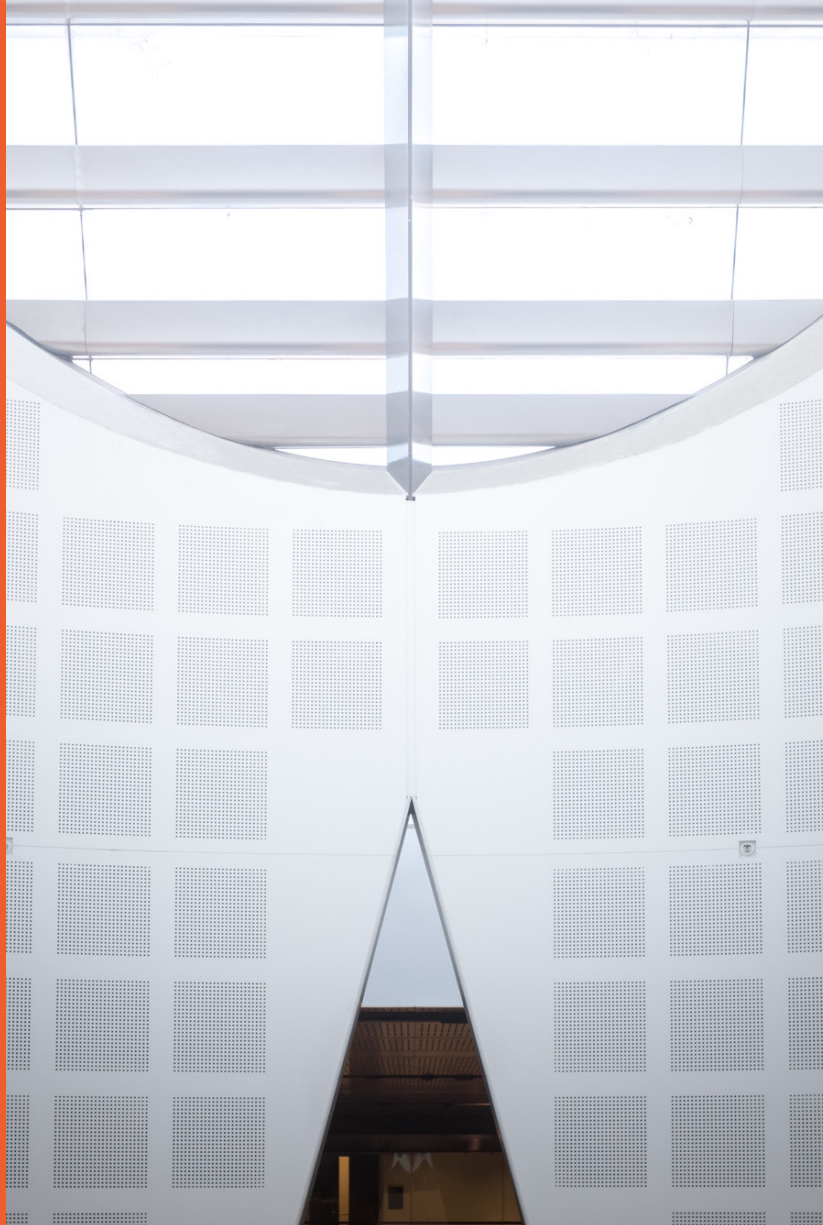
Recurrent Neural Networks

Dr Chang Xu

School of Computer Science

THE UNIVERSITY OF SYDNEY



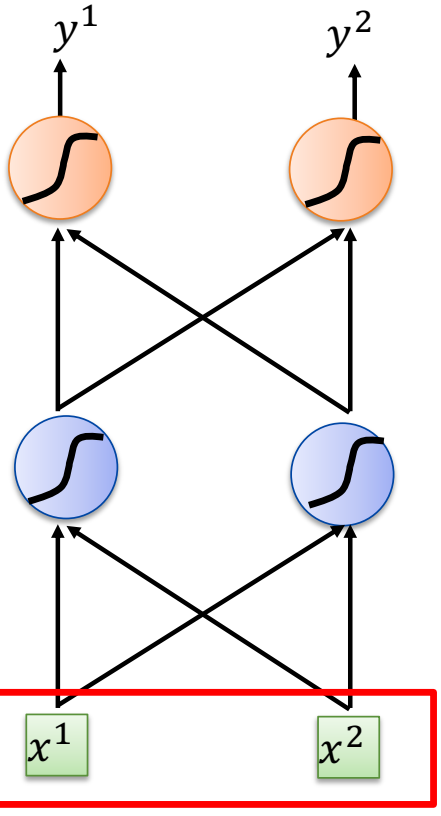
Example Application

sequential analysis

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

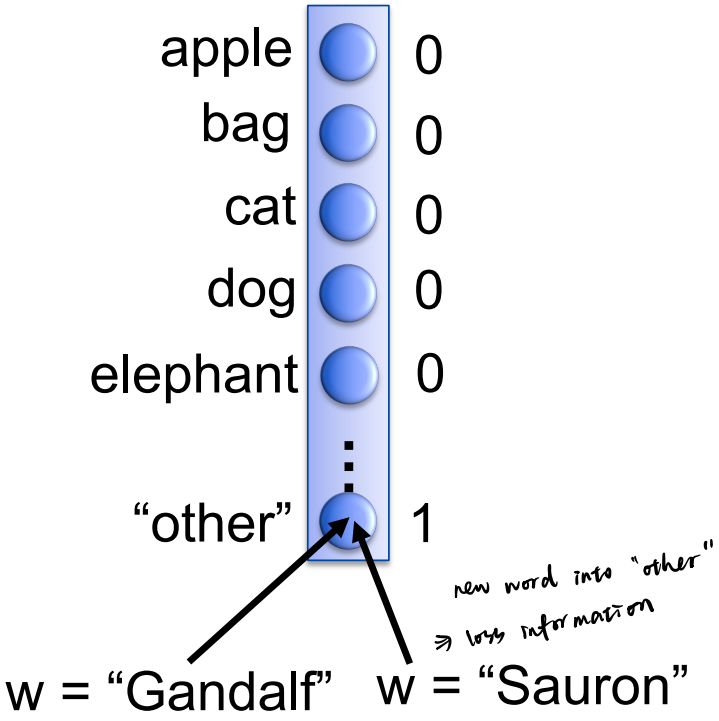
- One hot encoding
  - The length of vector is lexicon size
  - Each dimension corresponds to a word in the lexicon
  - The dimension for the word is 1, and others are 0

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

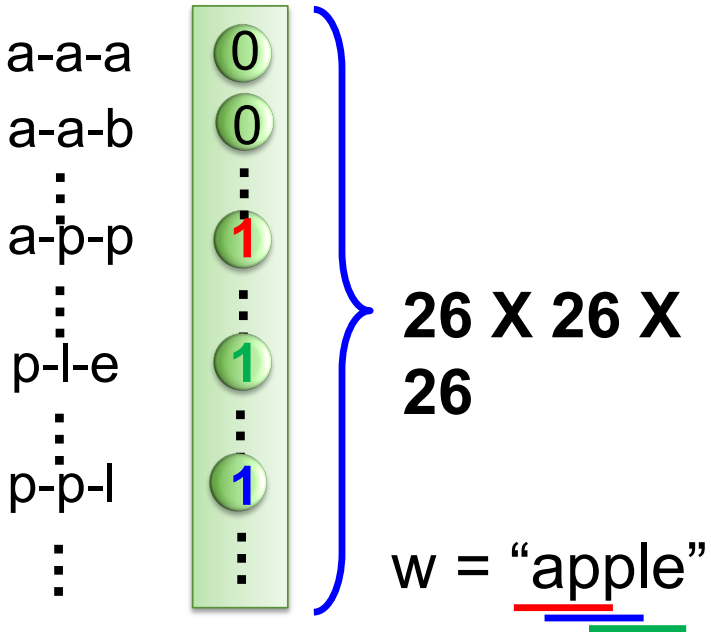
Word	D_1	D_2	D_3	D_4	D_5
Apple	1	0	0	0	0
Bag	0	1	0	0	0
Cat	0	0	1	0	0
Dog	0	0	0	1	0
Elephant	0	0	0	0	1

Recurrent Neural Network

Dimension for "Other"

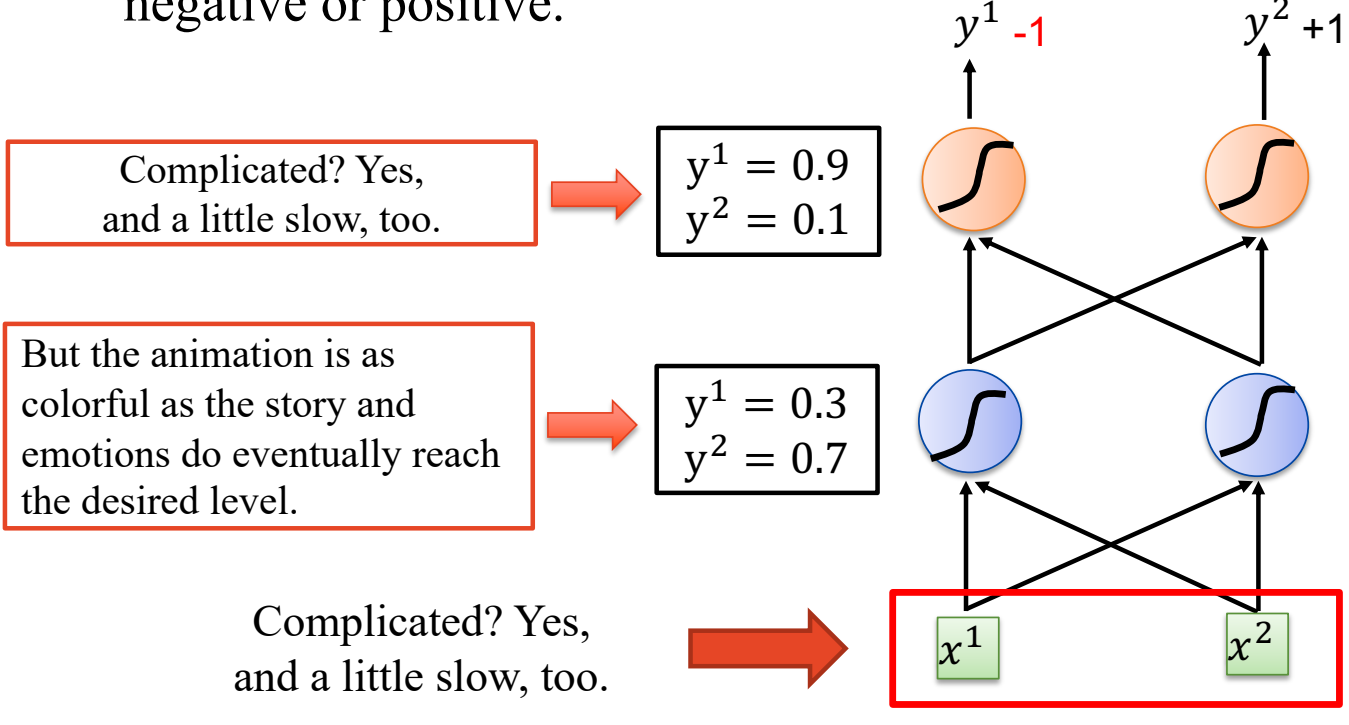


Word hashing

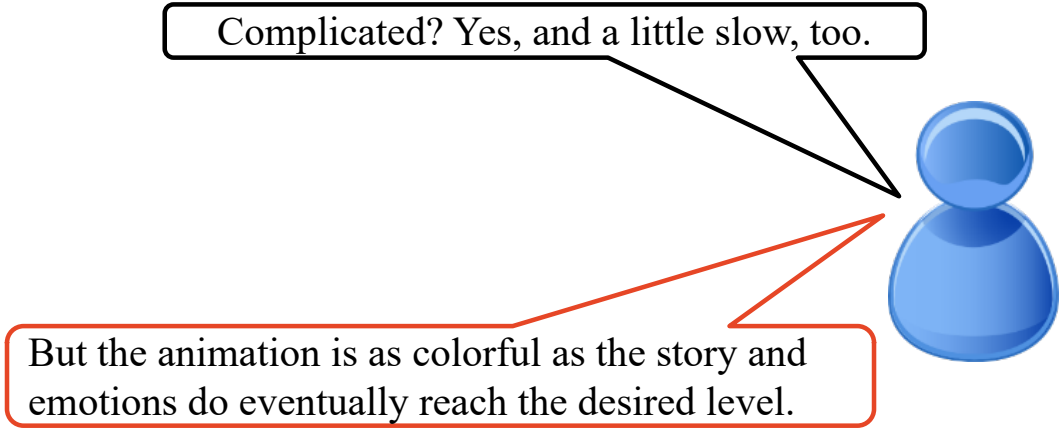


Recurrent Neural Network

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



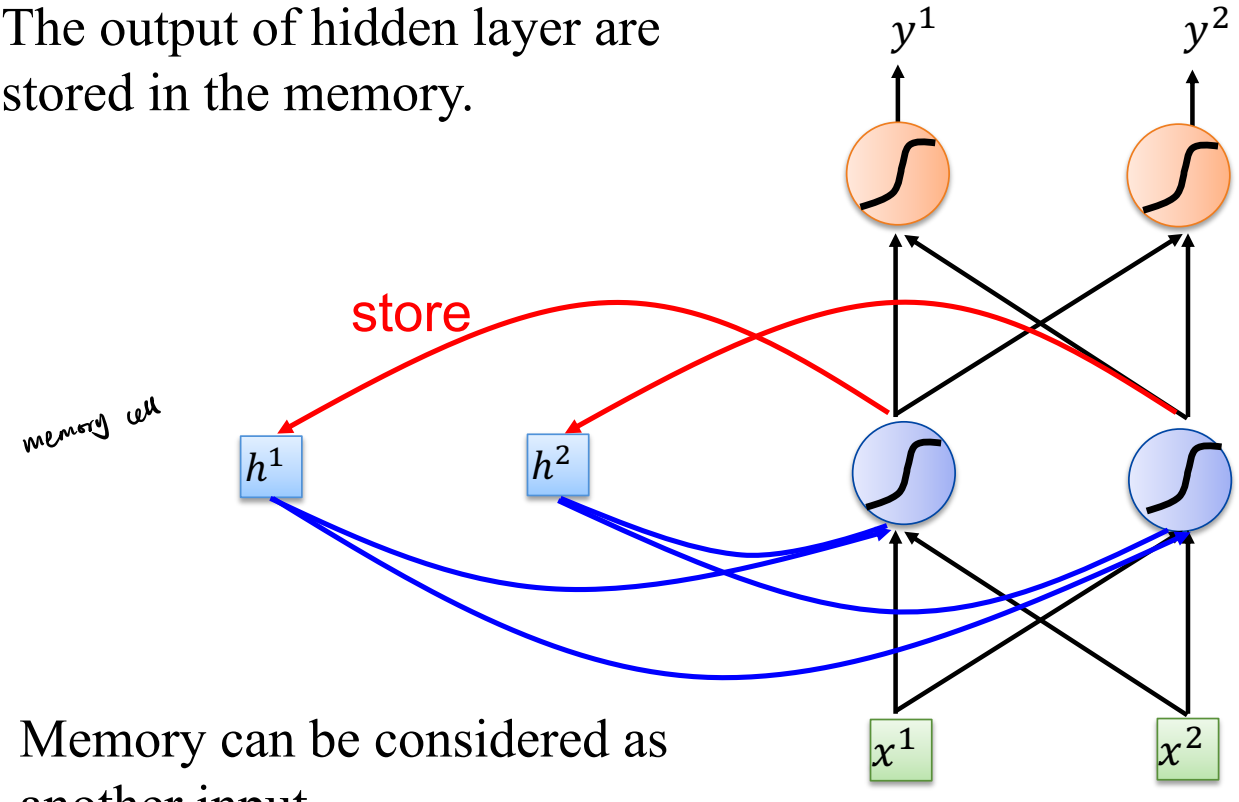
Recurrent Neural Network



Negative OR Positive → Need Memory

Recurrent Neural Network

The output of hidden layer are stored in the memory.

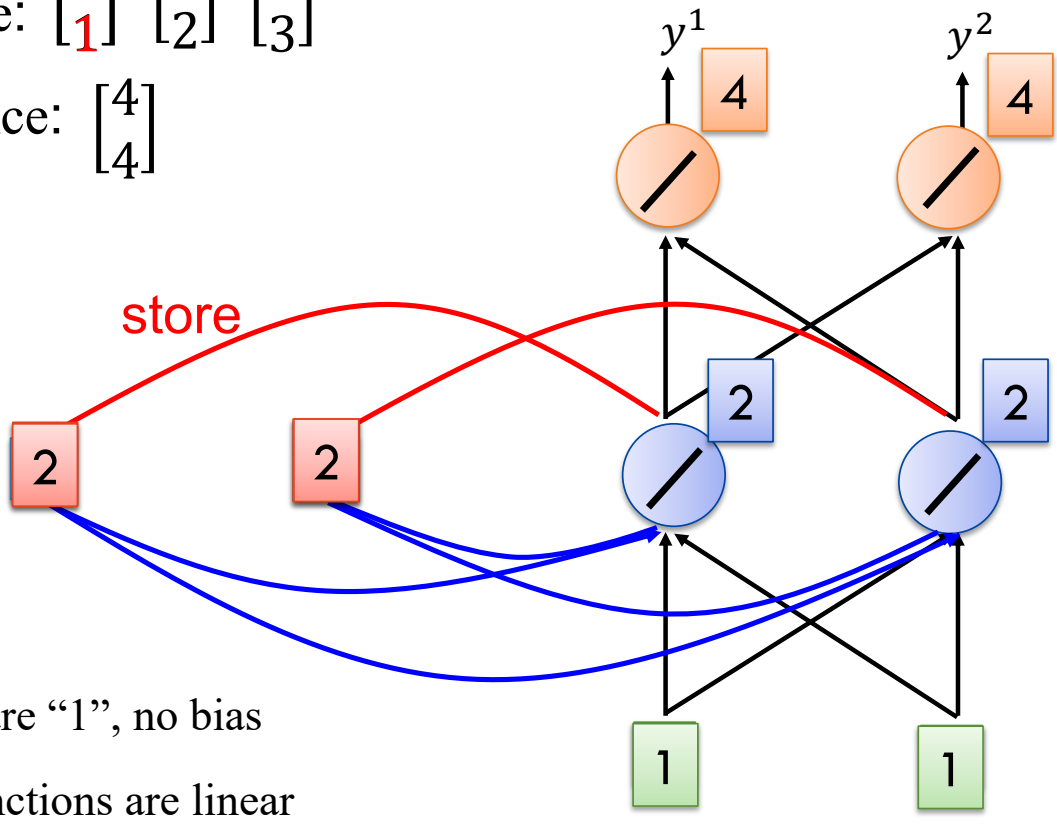


Memory can be considered as another input.

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



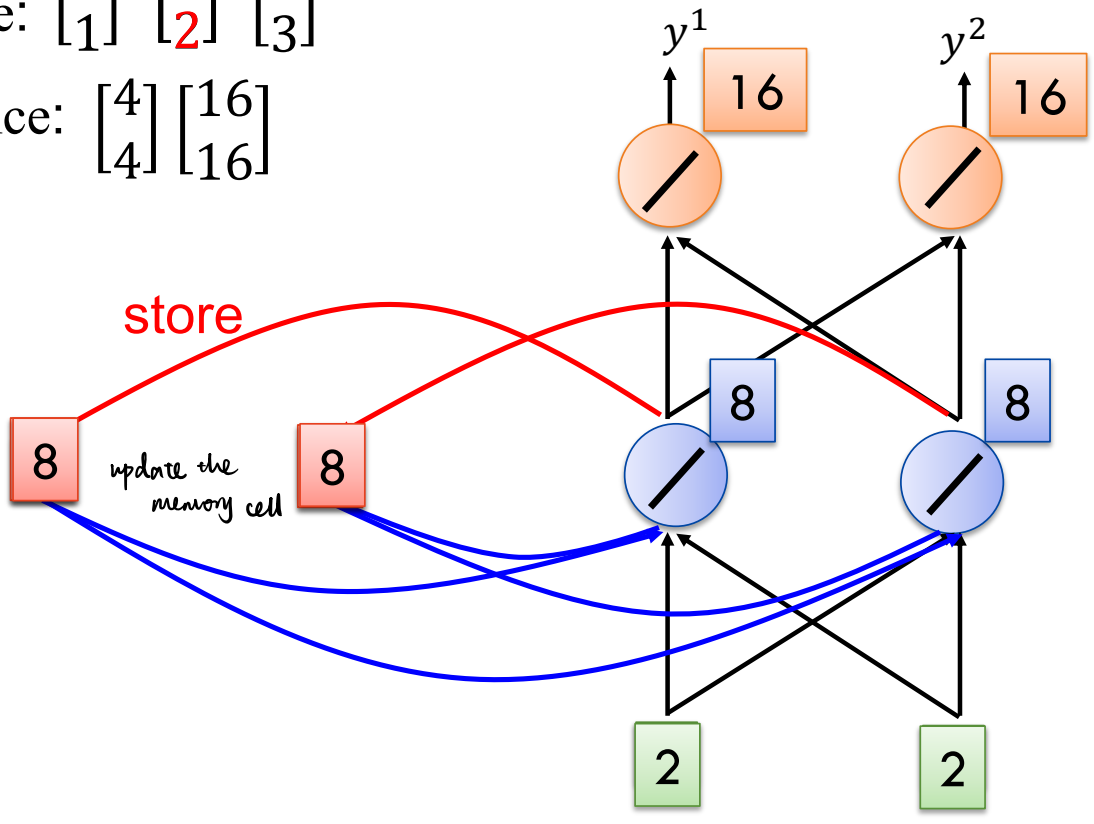
All the weights are "1", no bias

All activation functions are linear

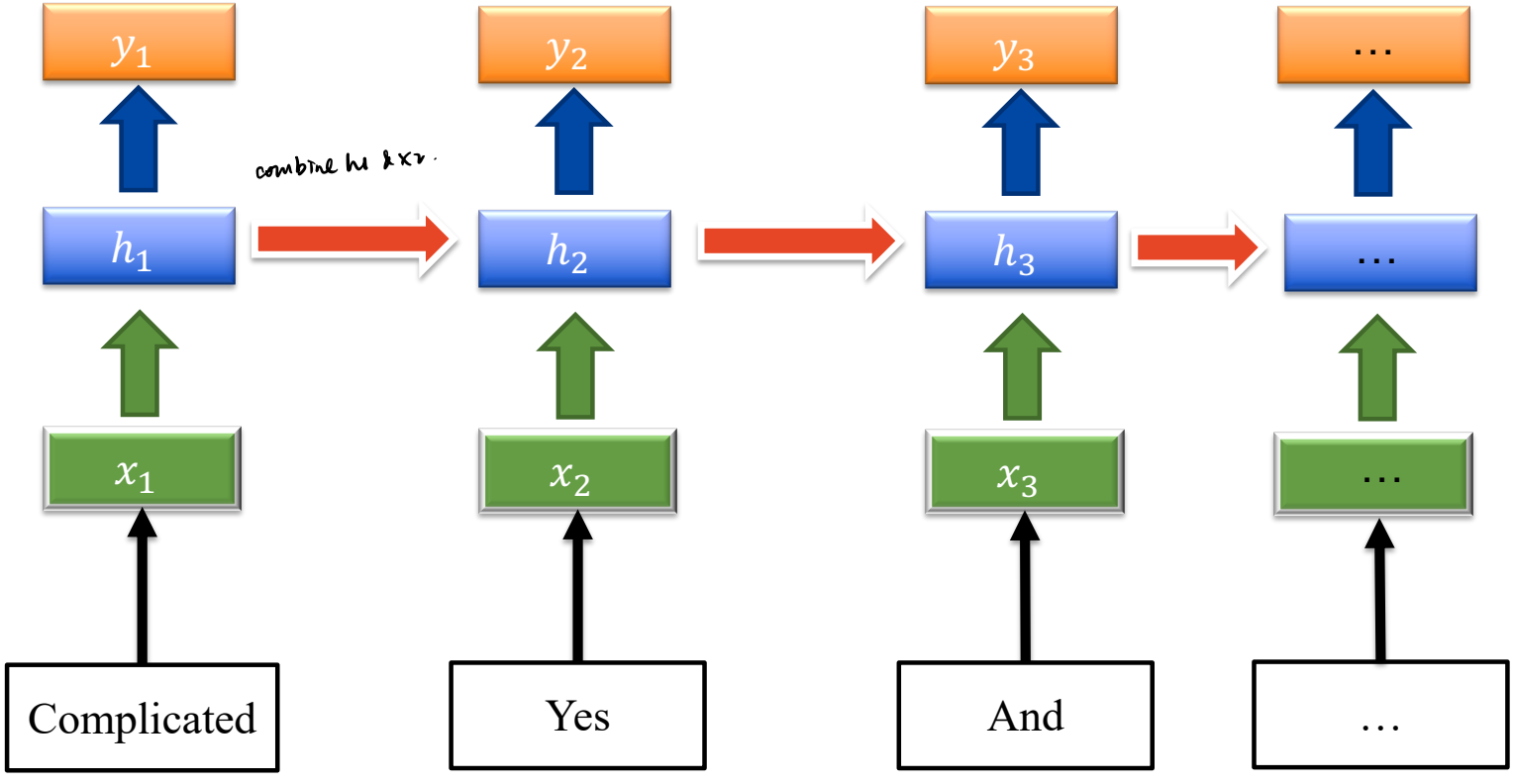
Recurrent Neural Network

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

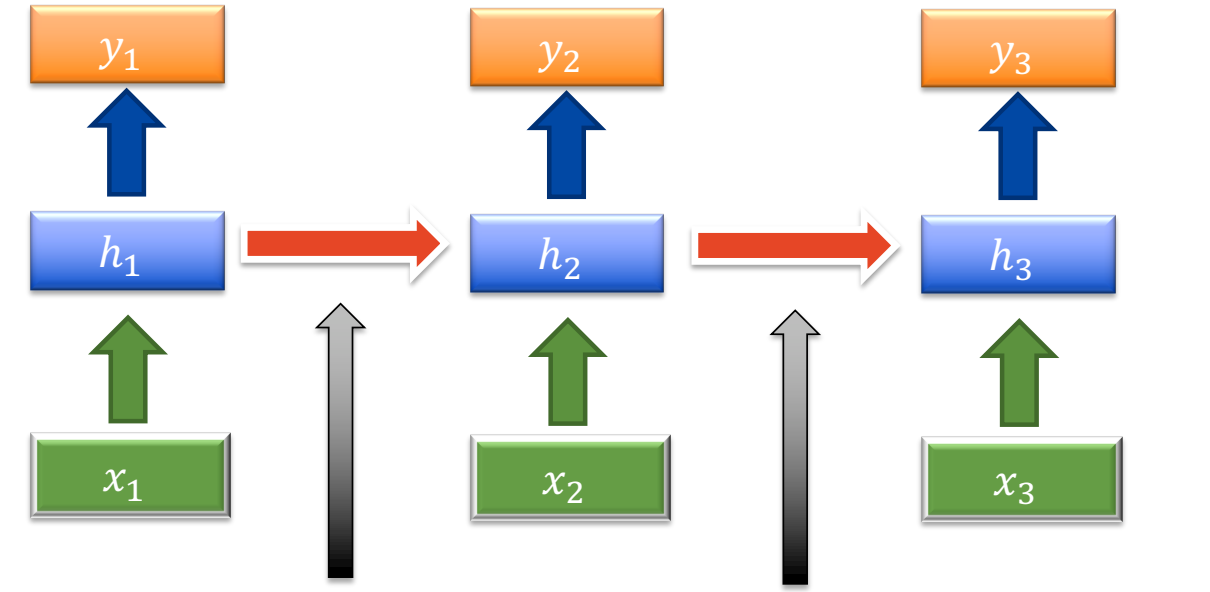
Output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$ $\begin{bmatrix}16\\16\end{bmatrix}$



Recurrent Neural Network



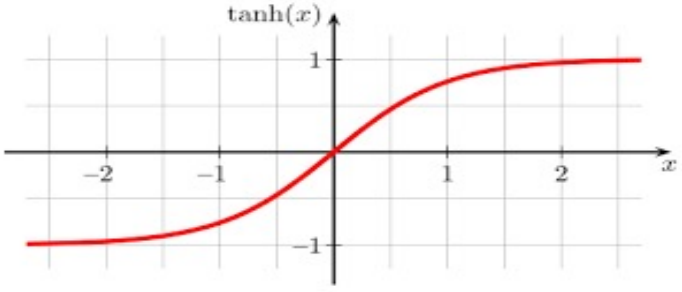
Recurrent Neural Network



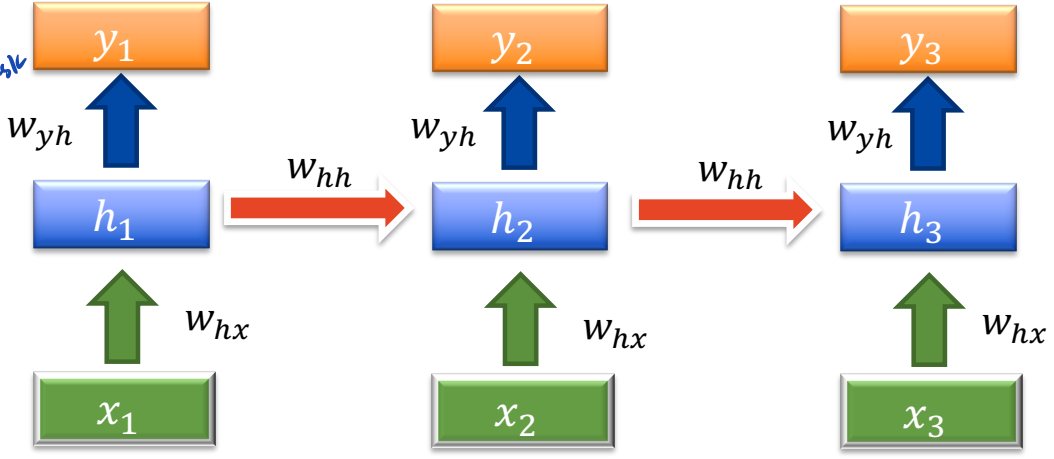
Memory can be considered as another input.

Recurrent Neural Network

- Formally
  - x_t is the input
  - h_t is the hidden state
  - y_t is the output
  - $h_0 = \vec{0}$ ← initialize memory value.
  - $h_t = \tanh(w_{hh} h_{t-1} + w_{hx} x_t + b_h)$ (previous, current)
  - $y_t = w_{yh} h_t$



whether to use $y_1, \ldots$ depends on the task



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

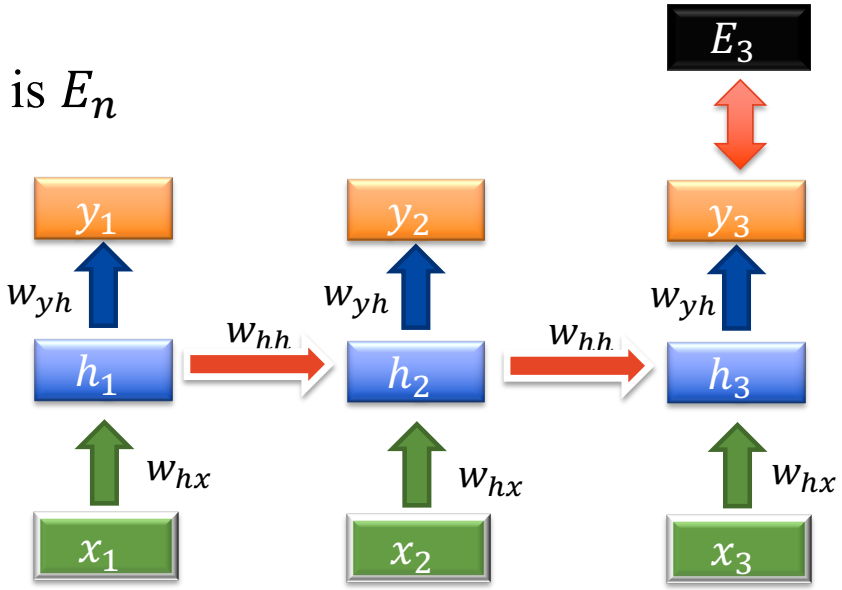
update W_{hh}

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

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

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

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



↓ substitude back to original function

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

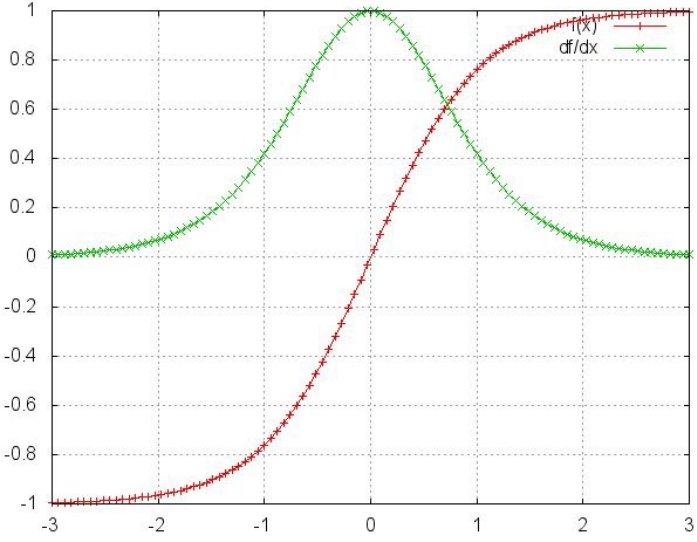
Recurrent Neural Network

- 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

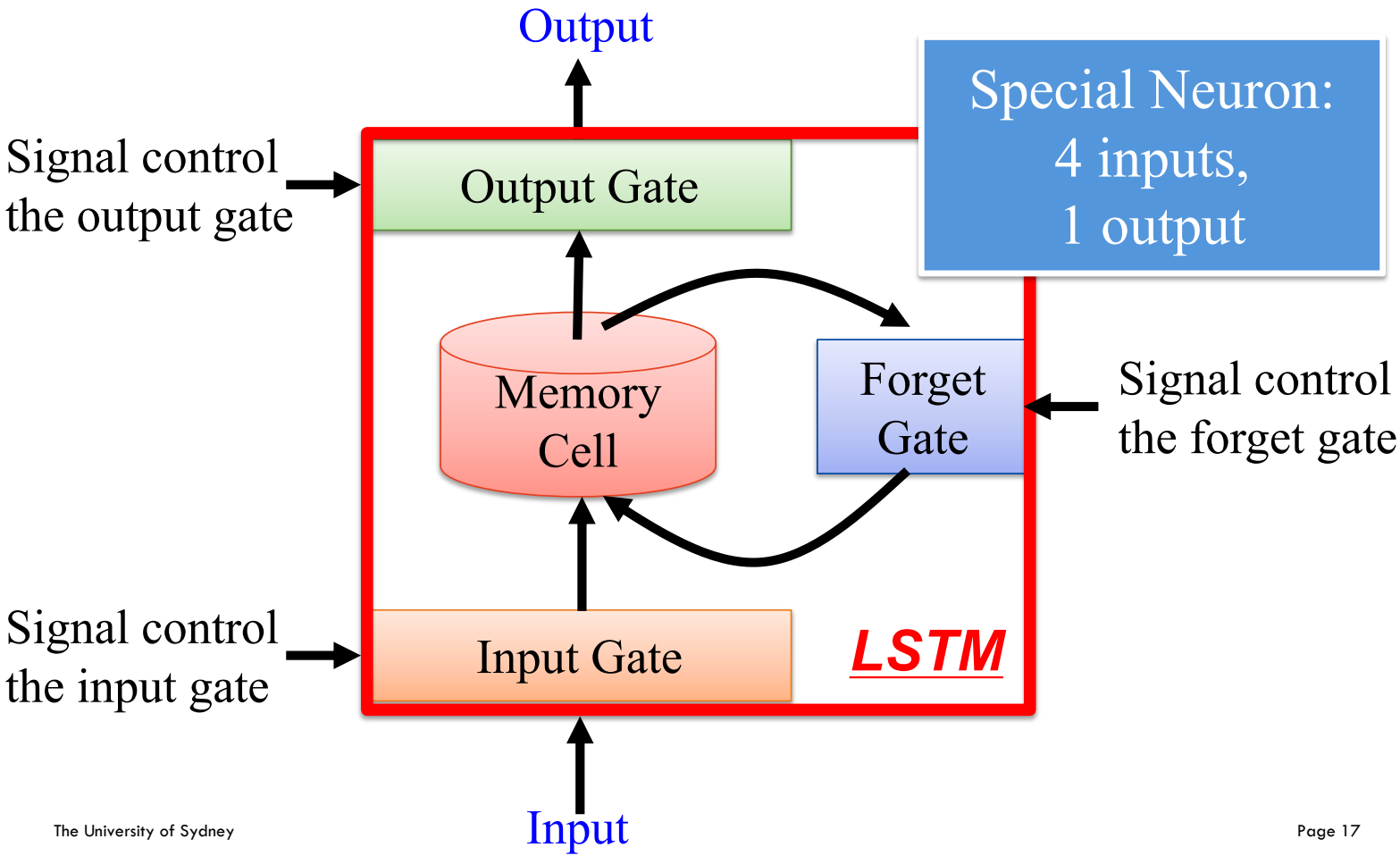
$$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to 0$$

If w_{hh} is larger, Exploding Gradients

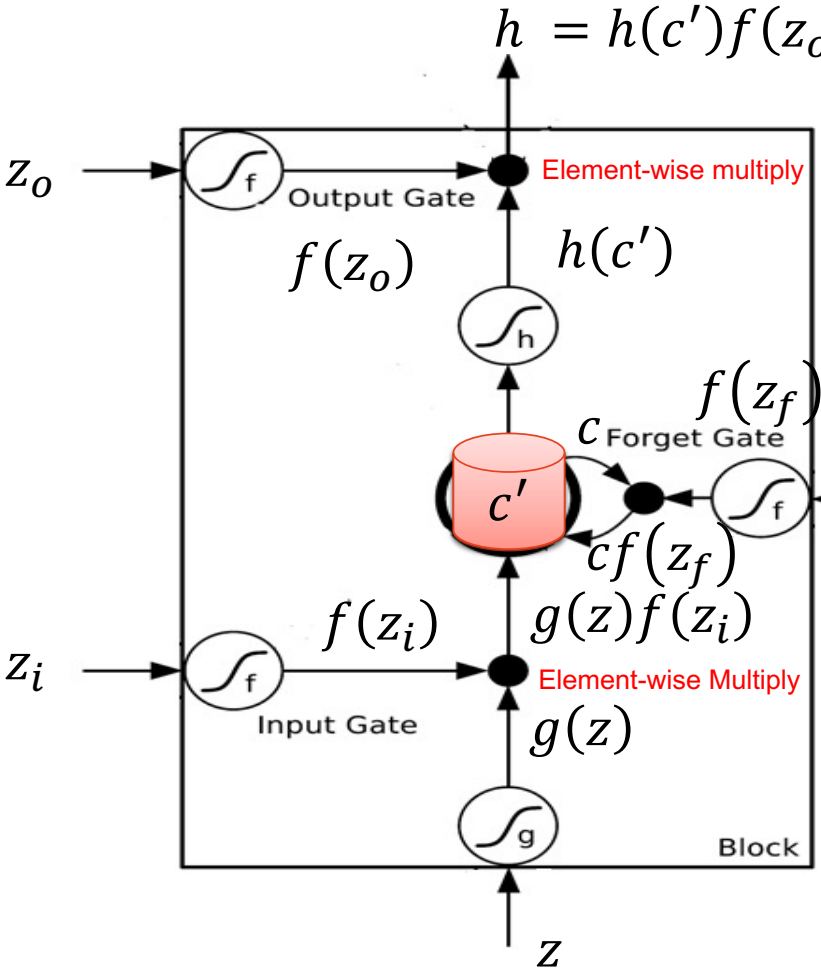
$$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to \infty$$

Long Short-Term Memory

Long Short-Term Memory



Long Short-Term Memory



f is usually a sigmoid function

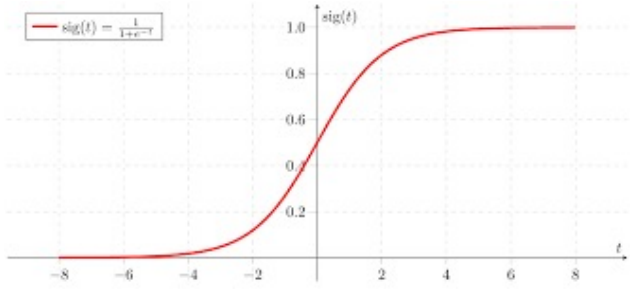
Value From 0 to 1

Mimic open and close gate

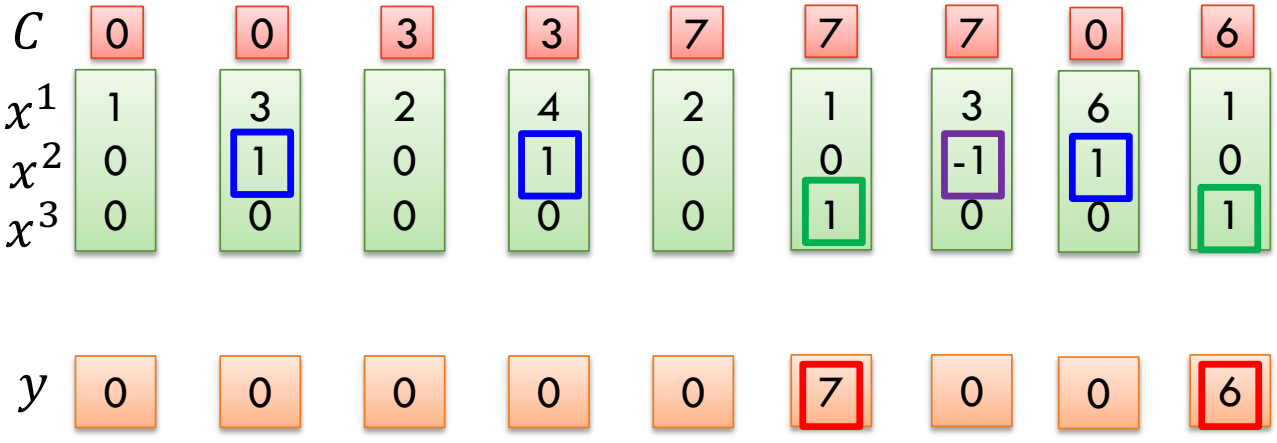
input gate.

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

new info



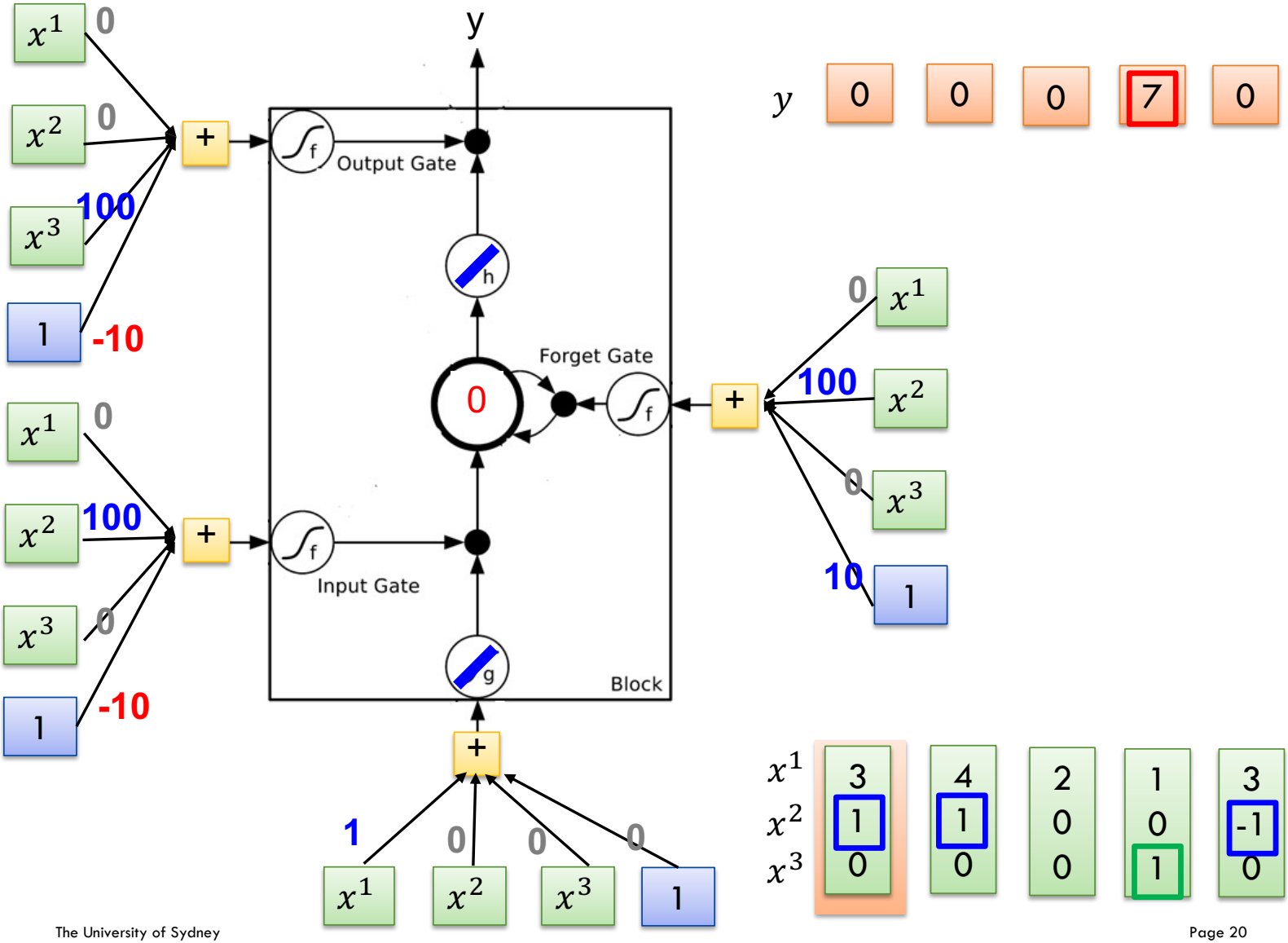
Long Short-Term Memory

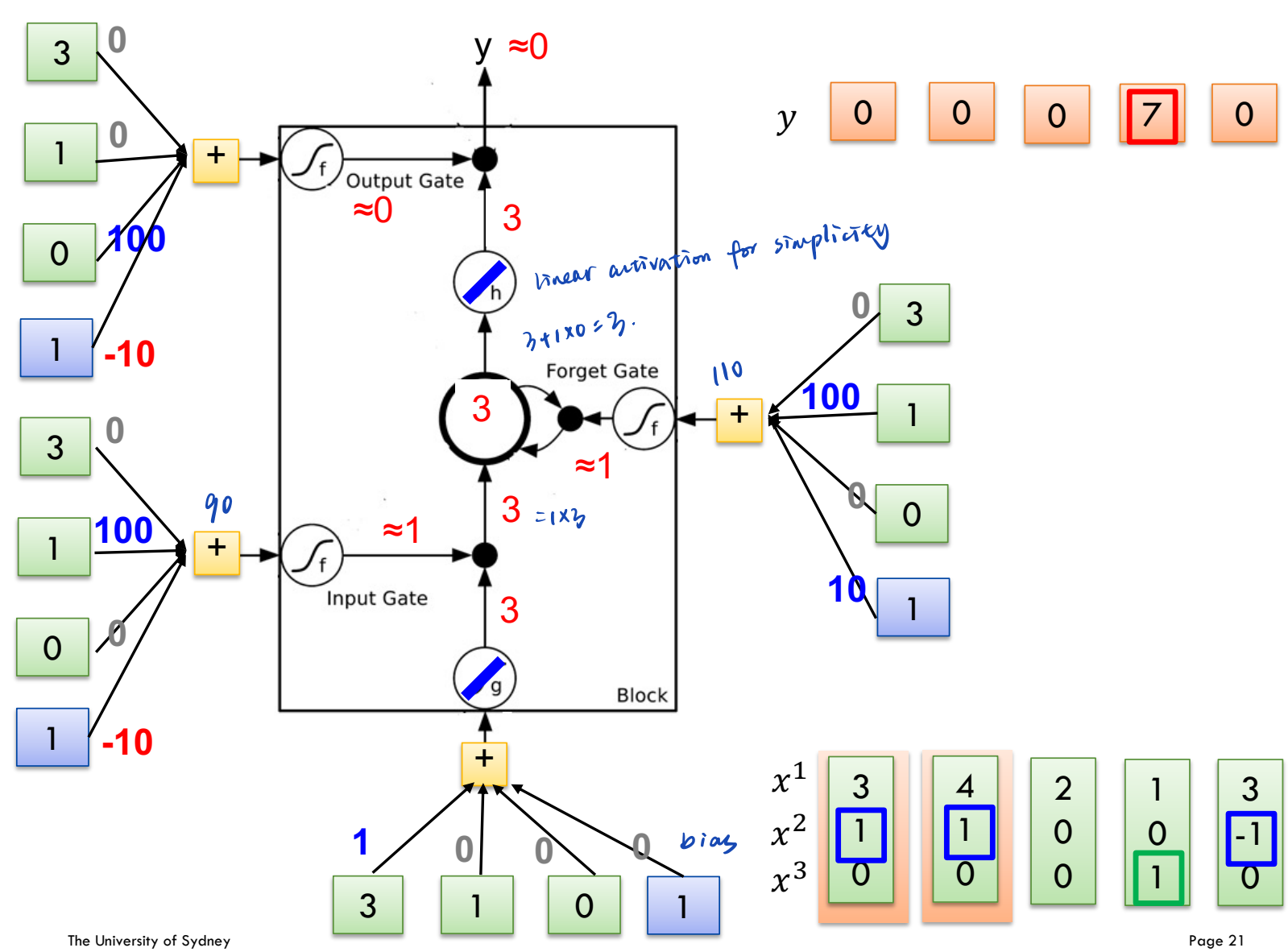


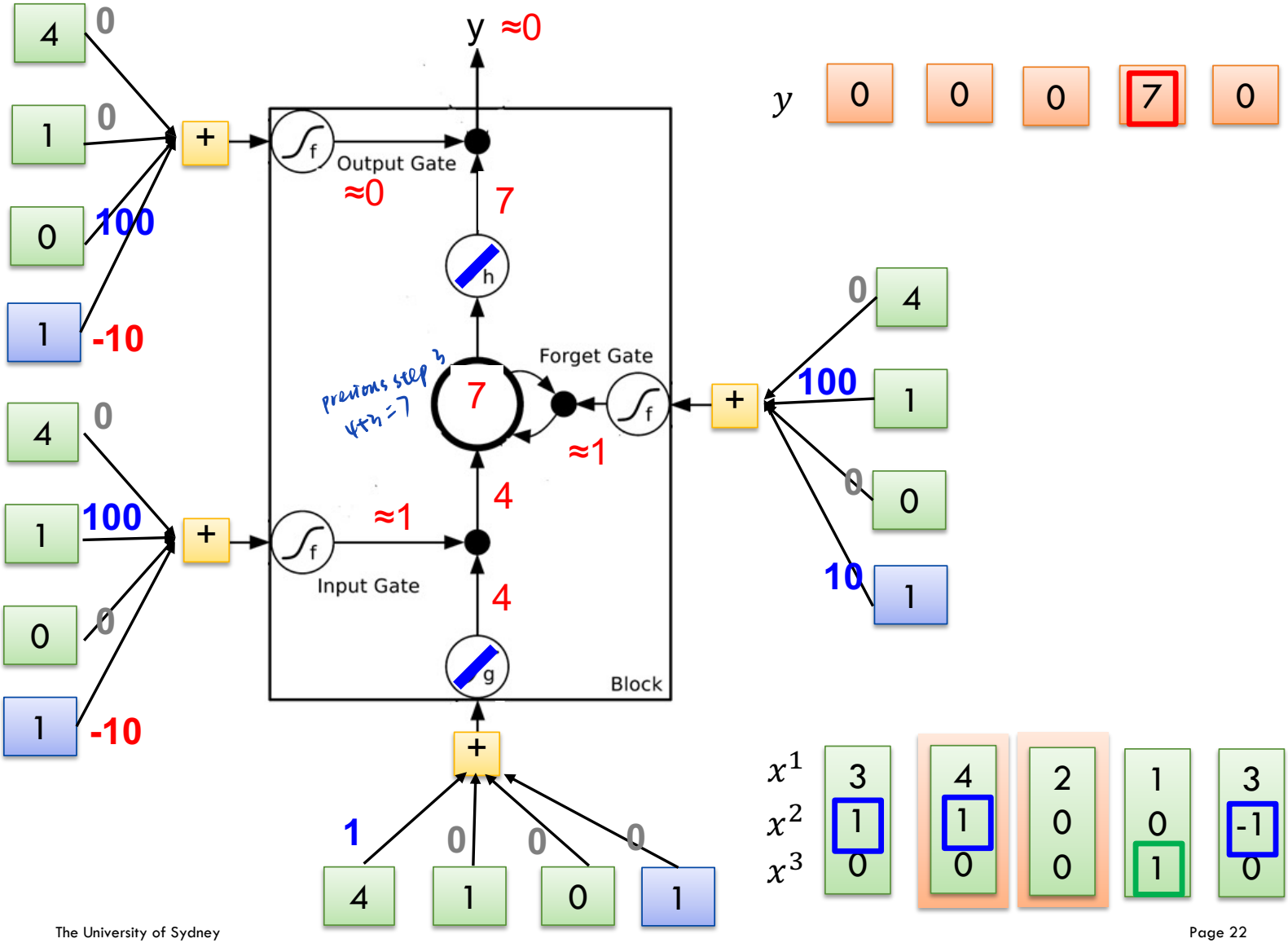
When $x^2 = 1$, add the numbers of x_1 into the memory

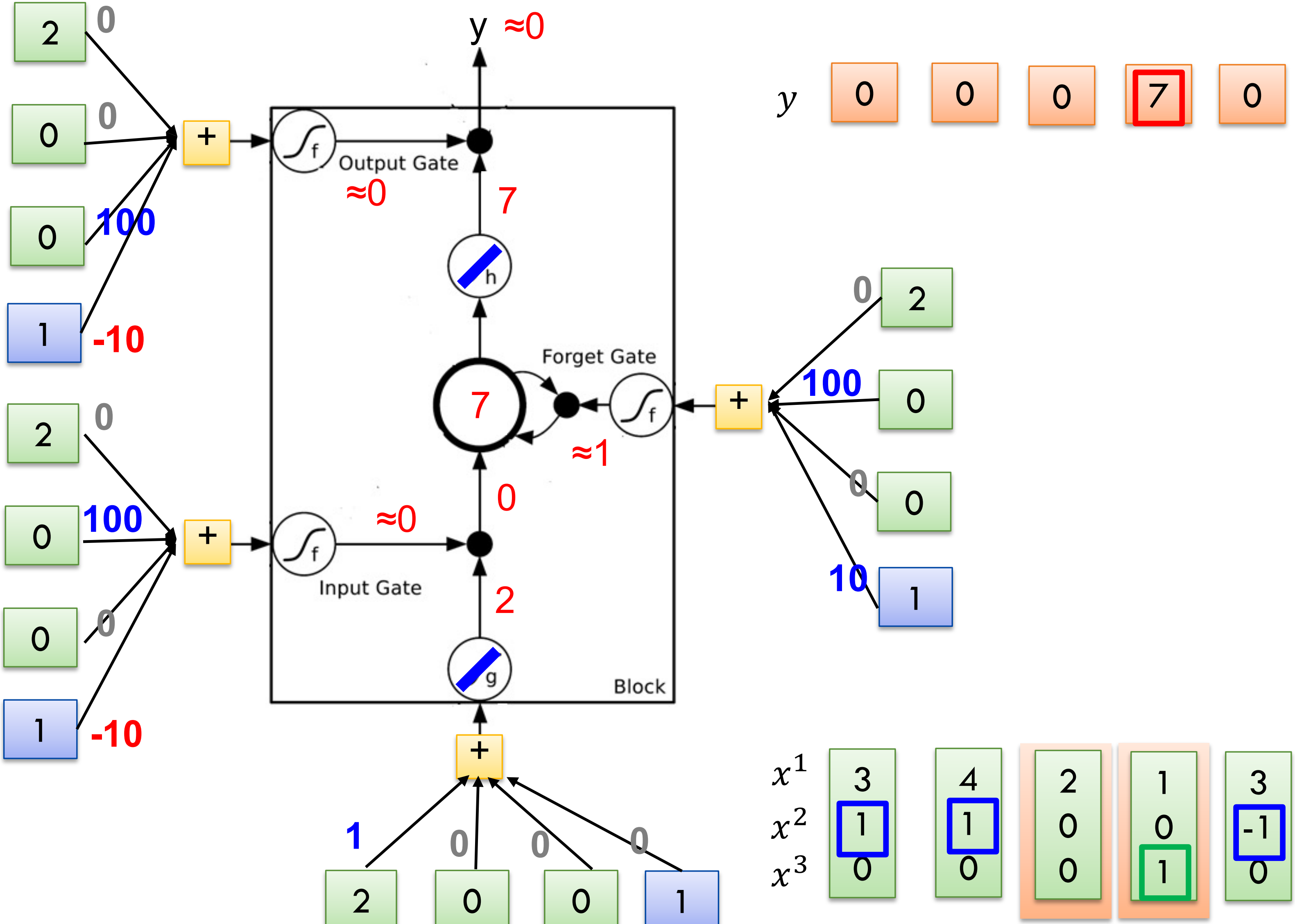
When $x^2 = -1$, reset the memory

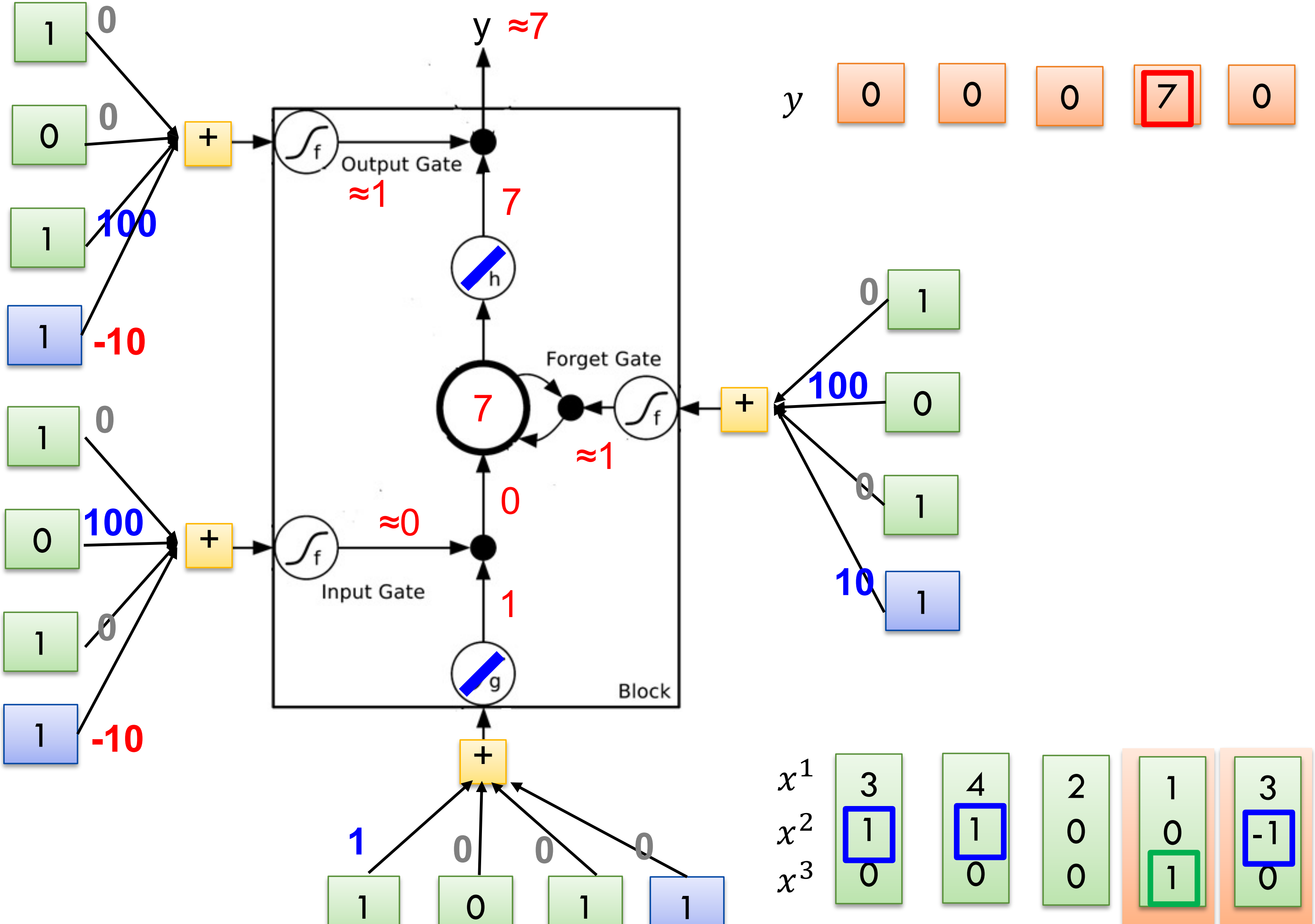
When $x^3 = 1$, output the number in the memory.

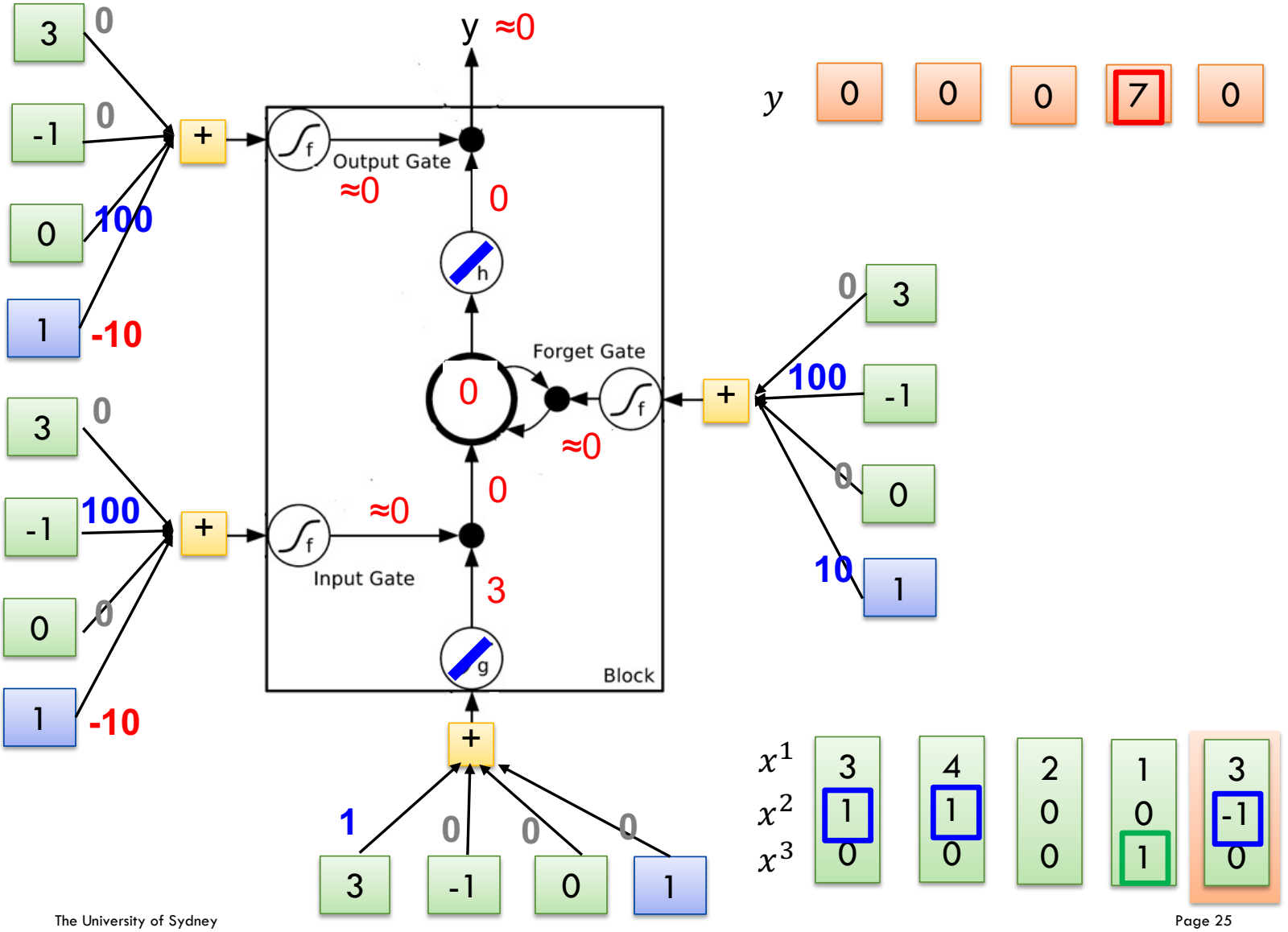

y
Output Gate
Forget Gate
Input Gate
Block
+
f
h
g
0
100
-10
10
1
x^1
x^2
x^3
y 0 0 0 7 0
x^1 3 4 2 1 3
x^2 1 1 0 0 -1
x^3 0 0 0 1 0


y ≈0
Output Gate
≈0
3
linear activation for simplicity
3+1x0=3.
Forget Gate
3
≈1
110
100
90
100
≈1
Input Gate
3 =1x3
3
Block
-10
-10
1
0
0
0
bias
y 0 0 0 7 0
x^1 3 1 0
x^2 4 1 0
x^3 2 0 0
1 0 1
3 -1 0

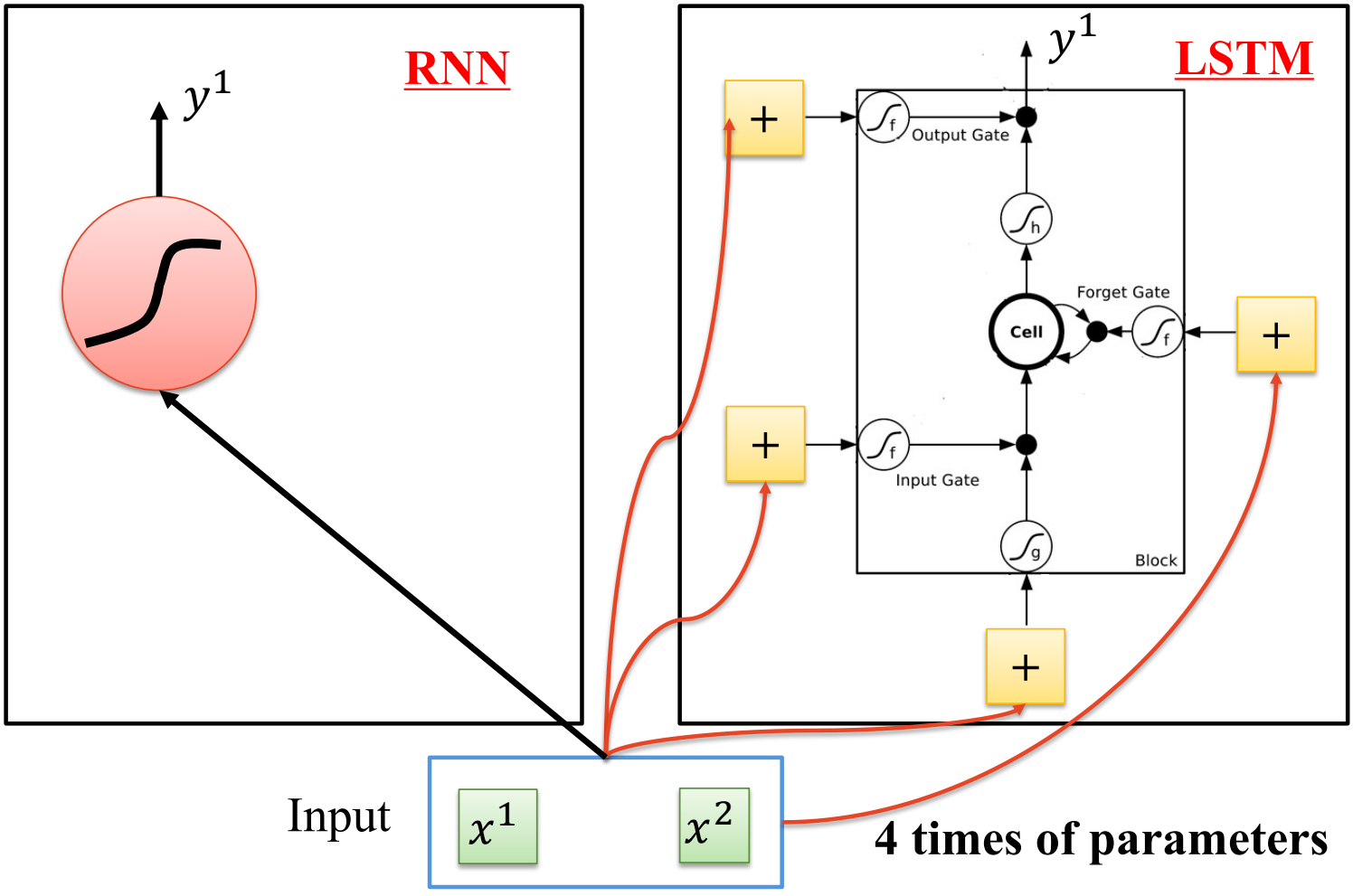

y ≈0
Output Gate
≈0
7
Forget Gate
previous step's y=7
7
≈1
4
≈1
Input Gate
4
Block
+
+
+
+
f
f
f
h
g
4
1
0
1
0
100
-10
4
100
0
-10
0
4
1
0
1
0
100
10
1
0
0
0
1
y 0 0 0 7 0
x^1 3 1 0
x^2 4 1 0
x^3 2 0 0
1 0 1
3 -1 0


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


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

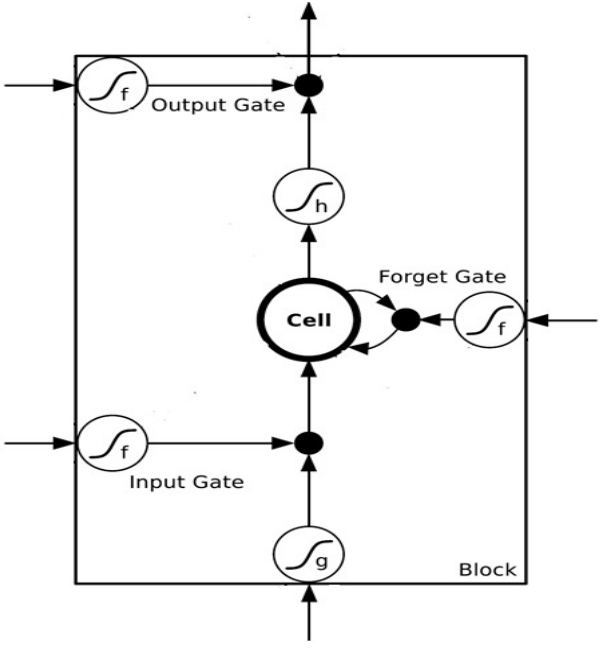

y ≈0
3
-1
0
1
0
0
100
-10
+
f
Output Gate
≈0
0
h
Forget Gate
0
f
+
≈0
0
3
100
-1
0
10
1
3
-1
100
0
1
-10
+
f
≈0
Input Gate
0
3
g
Block
+
1
0
0
0
3
-1
0
1
y 0 0 0 7 0
x^1 3 4 2 1 3
x^2 1 1 0 0 -1
x^3 0 0 0 1 0

Long Short-Term Memory



Long Short-Term Memory

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

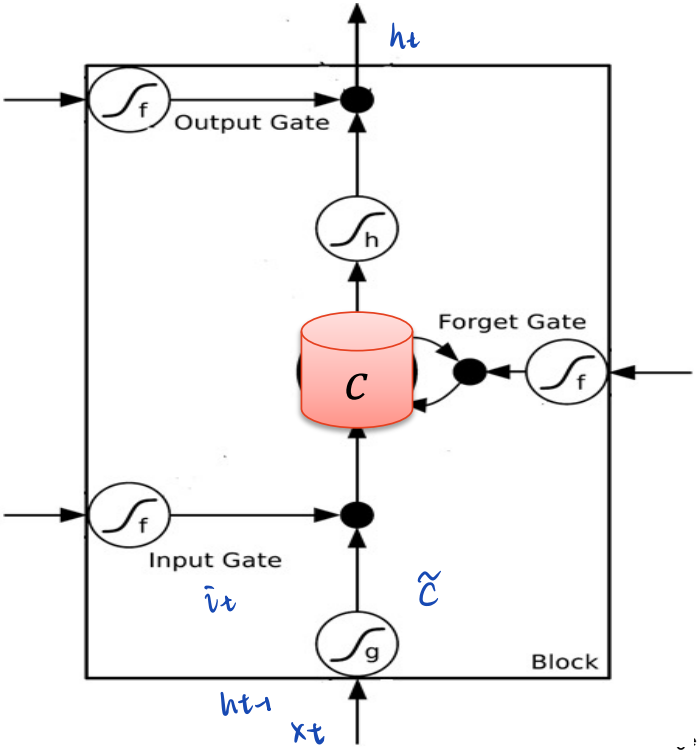


Long Short-Term Memory

- 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
  - $h_t = o_t * \tanh(c_t)$

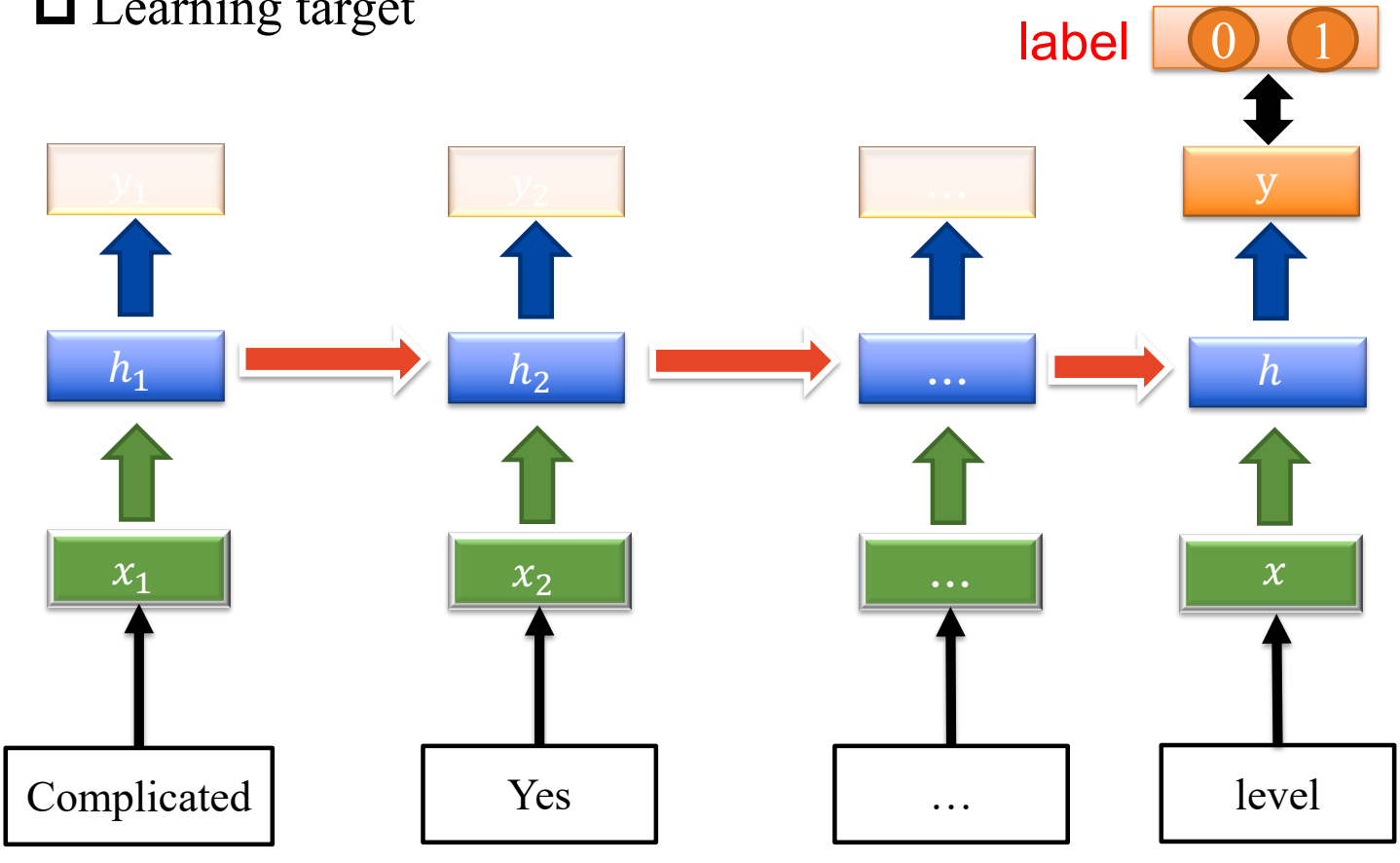
memory value from last time step

↑ memory value



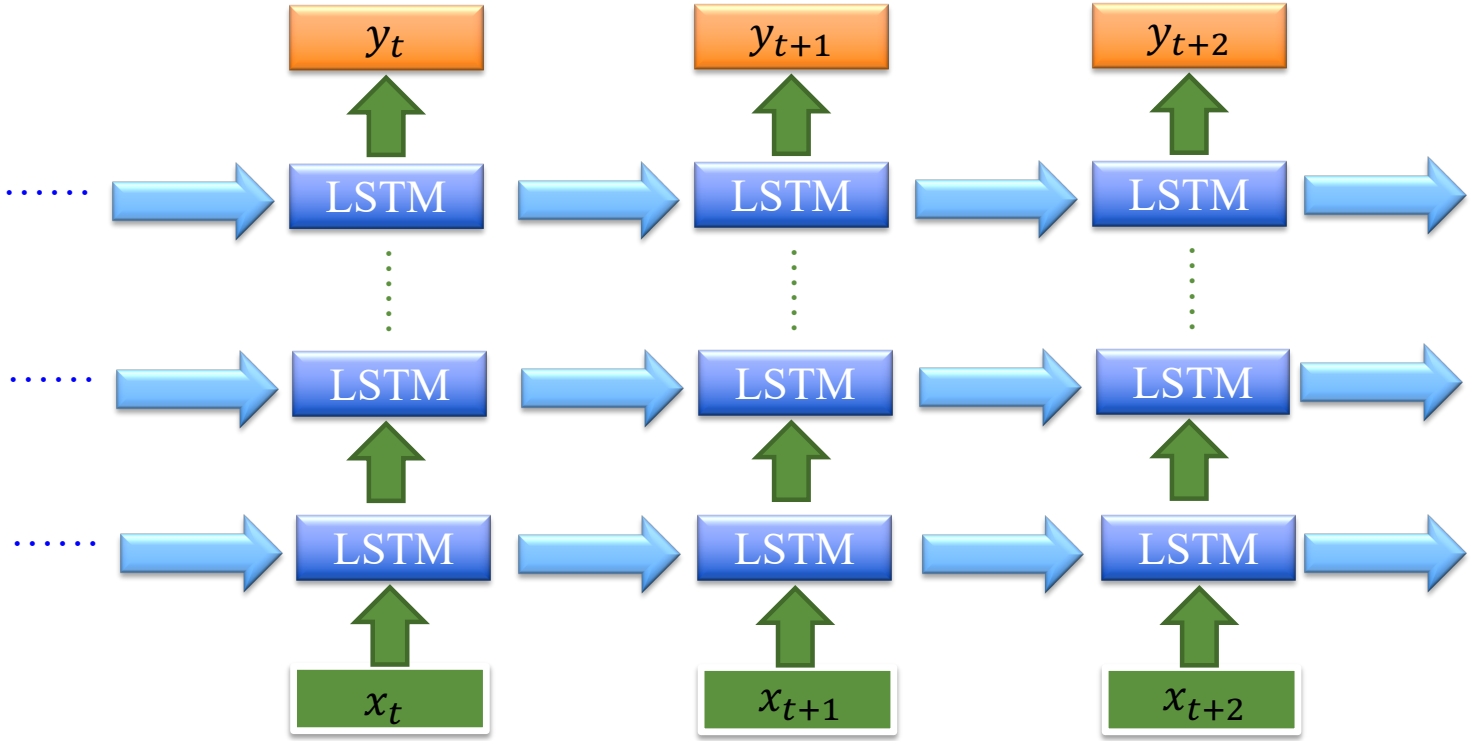
Long Short-Term Memory

- Learning target



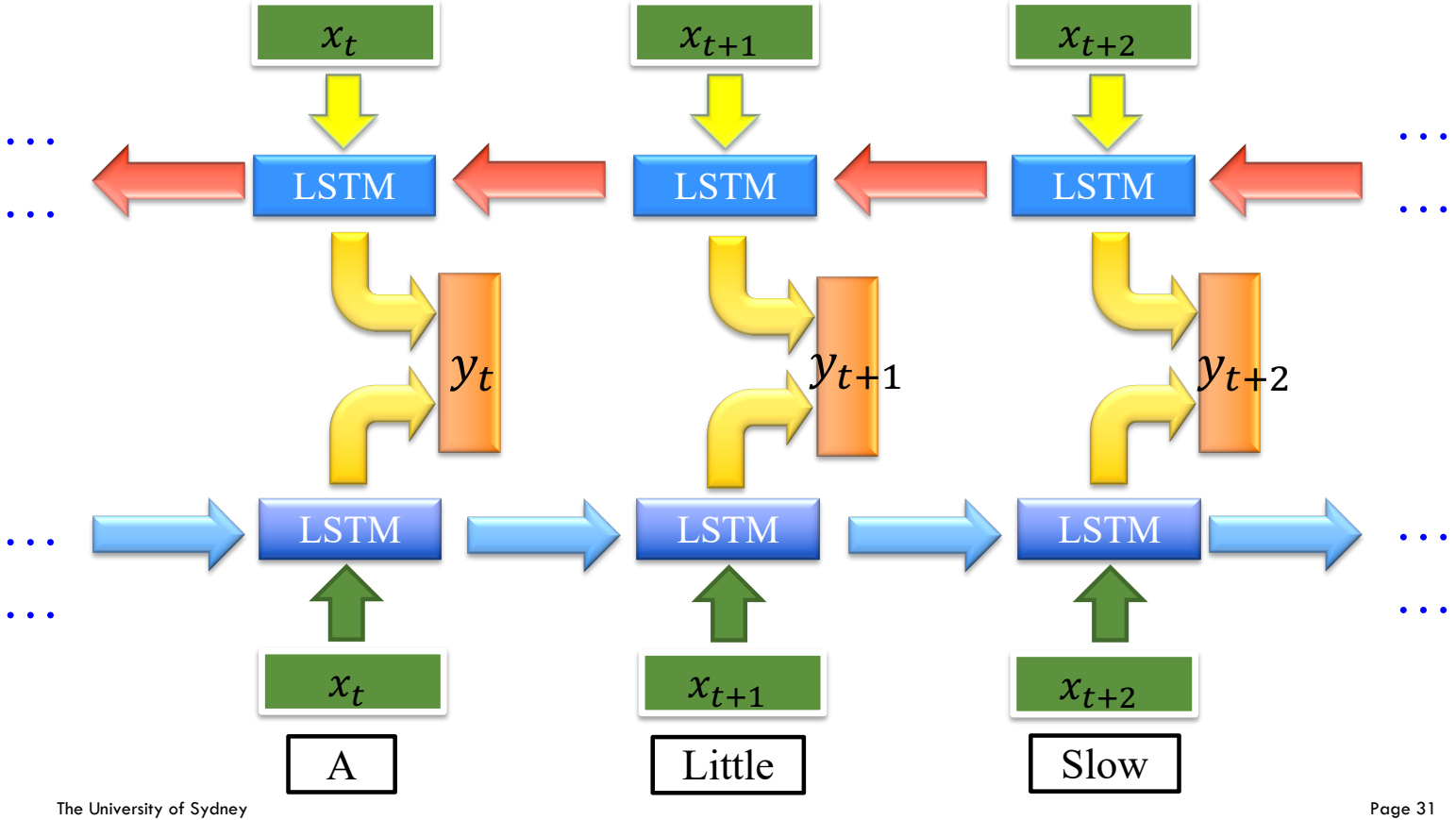
Long Short-Term Memory

- Stacked LSTM



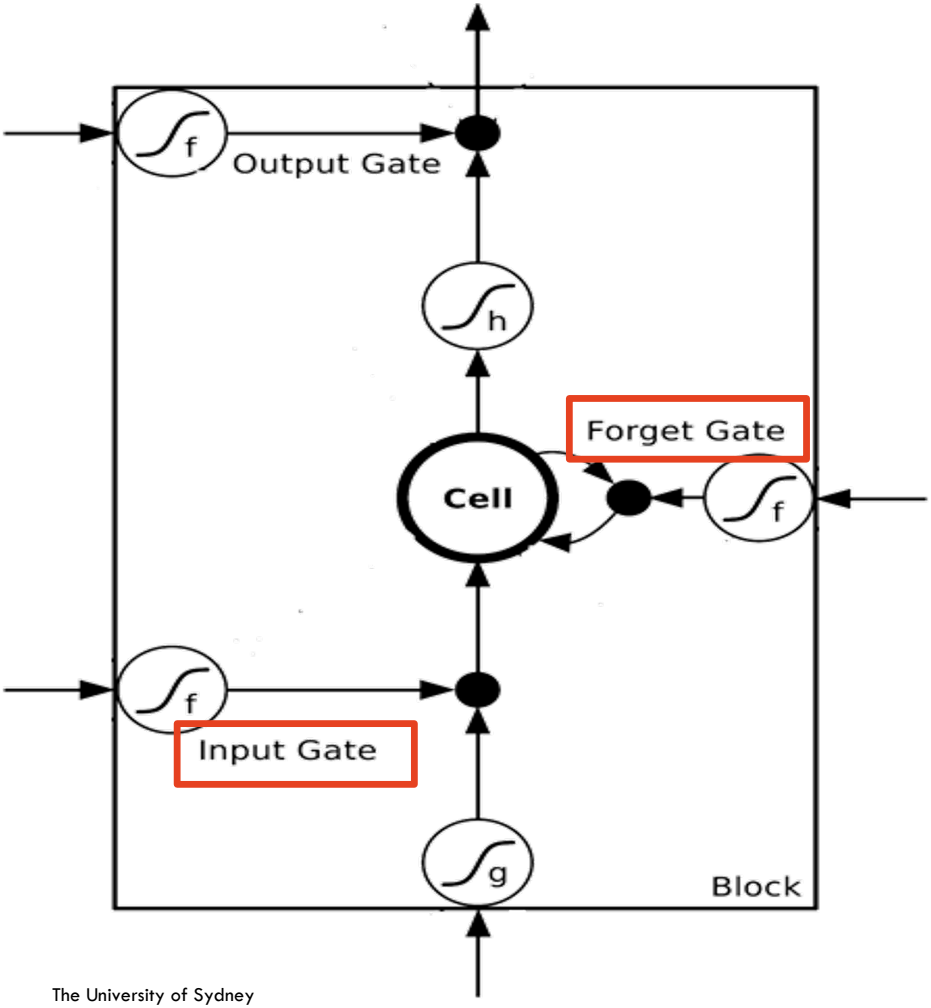
Long Short-Term Memory

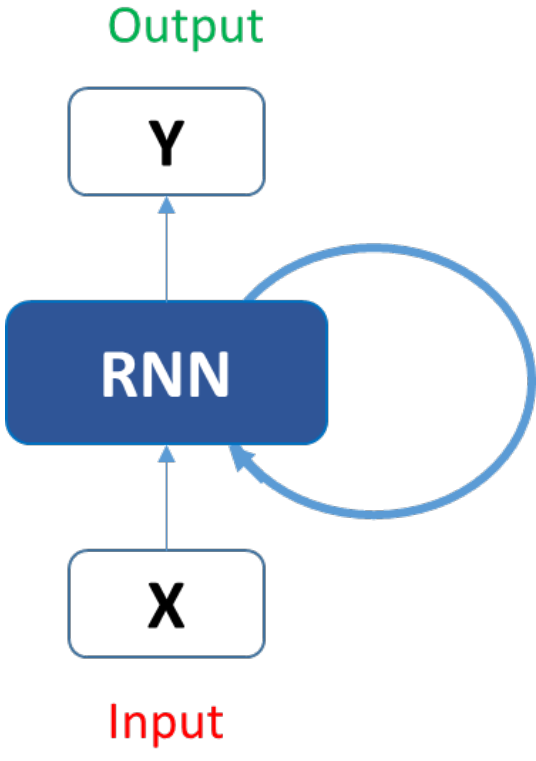
- Bidirectional RNN



LSTM 的 maintain more parameters

LSTM V.S. RNN





Prevent vanishing gradient

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

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial f_t}{\partial c_{t-1}} c_{t-1} + f_t + \frac{\partial i_t}{\partial c_{t-1}} * \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} i_t$$

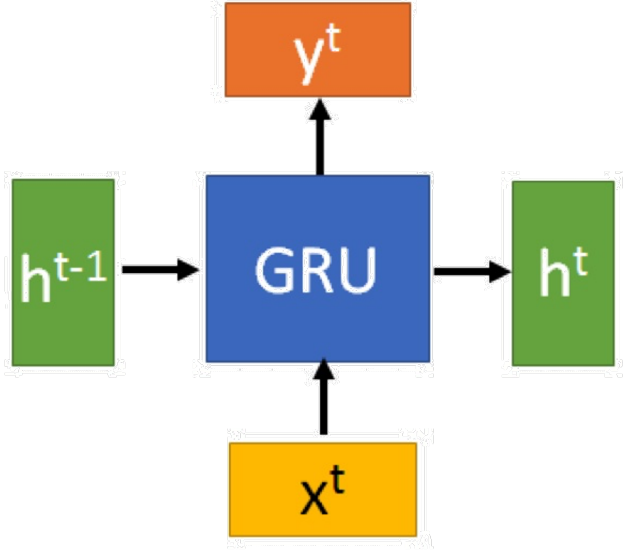
① small chance to have all terms being small values

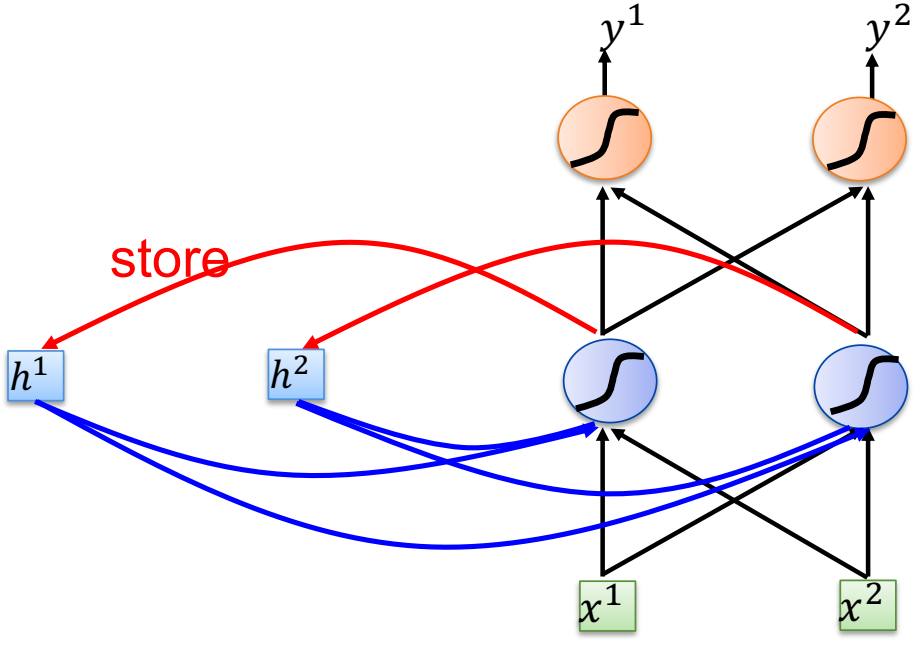
② prevent vanishing gradient problem, but may have gradient explosion problem

LSTMs solve the problem using a unique additive gradient structure that includes direct access to the forget gate's activations, enabling the network to encourage desired behaviour from the error gradient using frequent gates update on every time step of the learning process.

Gate Recurrent Unit

Gate Recurrent Unit

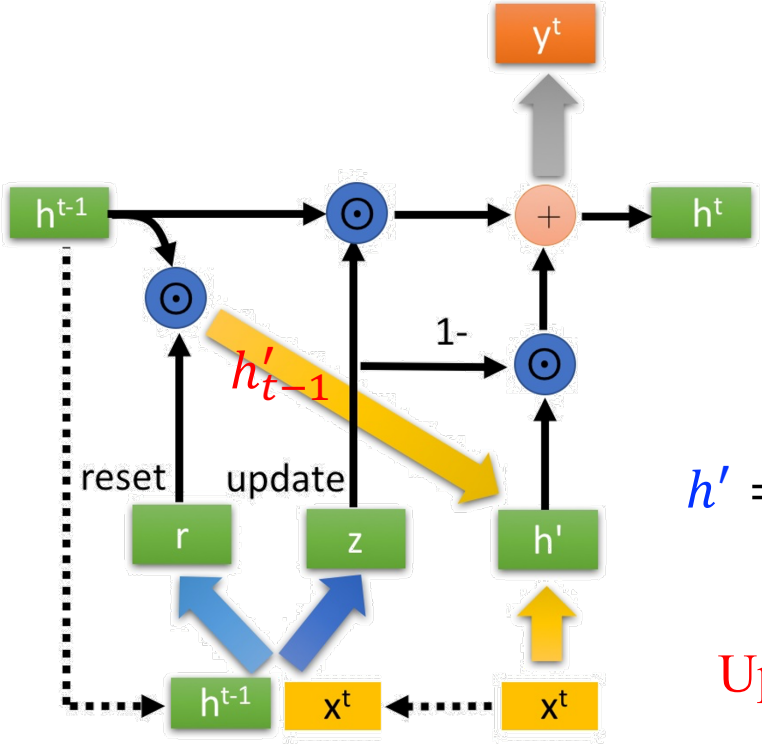




Vanilla RNN

Gate Recurrent Unit

parameter efficient



Reset Gate

$$r_t = \sigma(w_{rh}h_{t-1} + w_{rx}x_t + b_r)$$

$$h'_{t-1} = r_t * h_{t-1}$$

$$h' = \tanh(w_{hh}h'_{t-1} + w_{hx}x_t + b_h)$$

Update Gate (forget gate + input gate)

$$z_t = \sigma(w_{zh}h_{t-1} + w_{zx}x_t + b_z)$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * h'$$

memory new info

Application

- Visual Question Answer (VQA)
- Reading Comprehension
- Sequence To Sequence (Seq2seq)
  - Language Translation
  - Chat Robot

VQA

What's the mustache made of ?

AI System

Banana

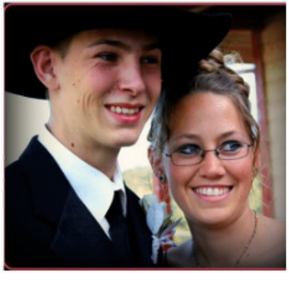
Image credit to http://www.visualqa.org/challenge.html

VQA

Who is wearing glasses?

man

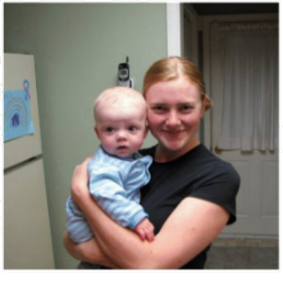
woman



Where is the child sitting?

fridge

arms



Is the umbrella upside down?

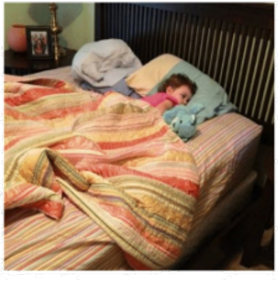
yes

no

How many children are in the bed?

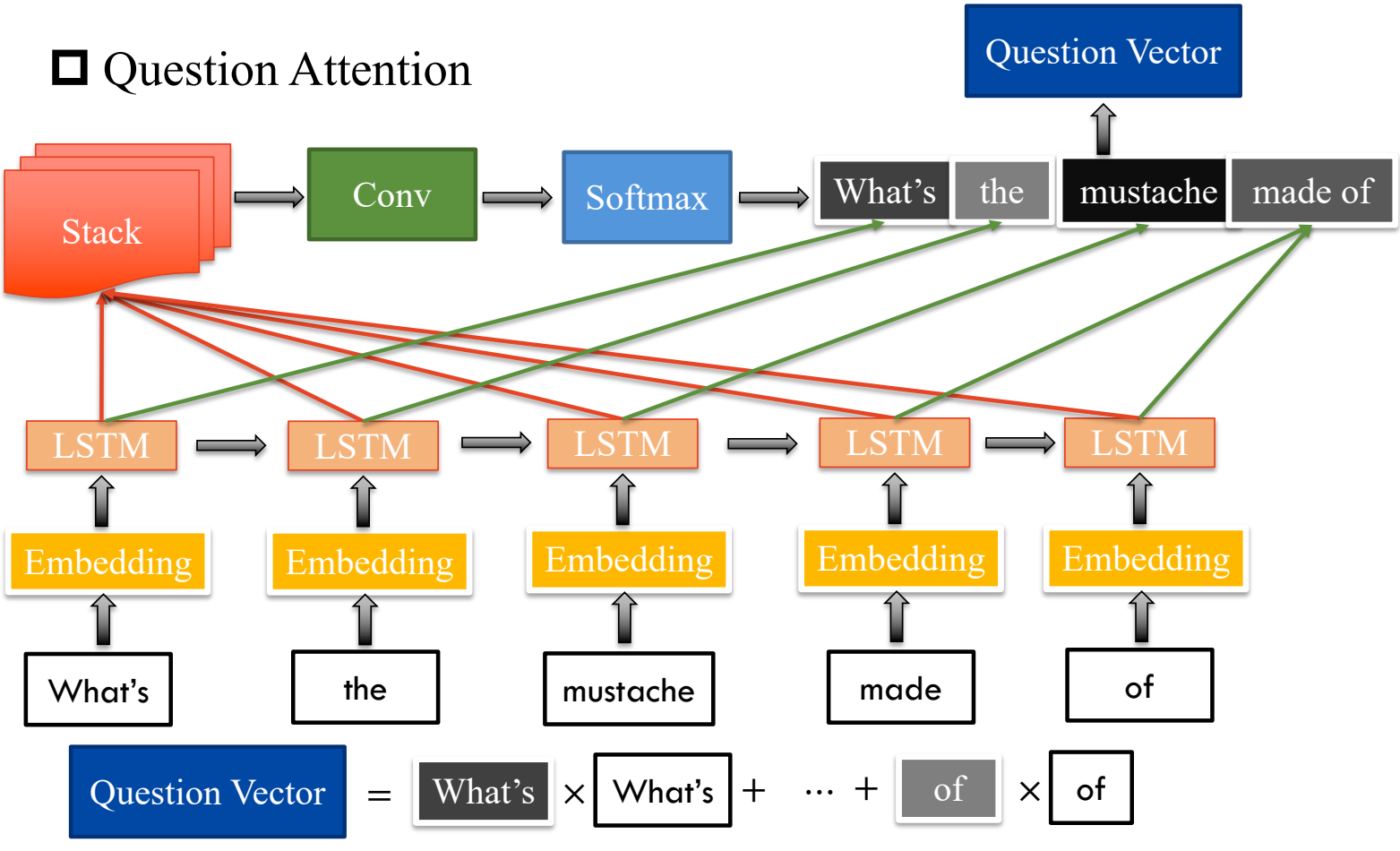
2

1



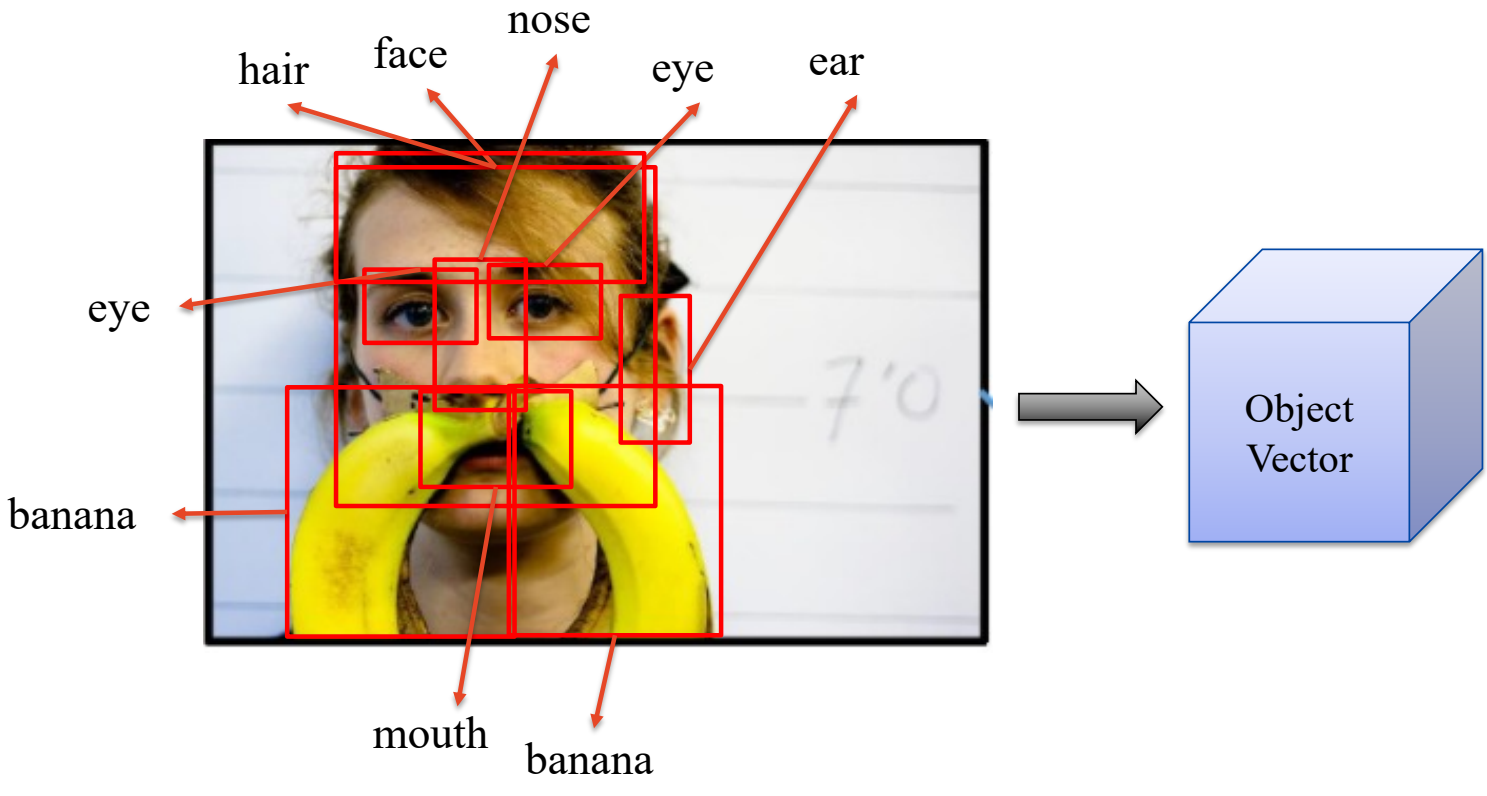
VQA

Question Attention



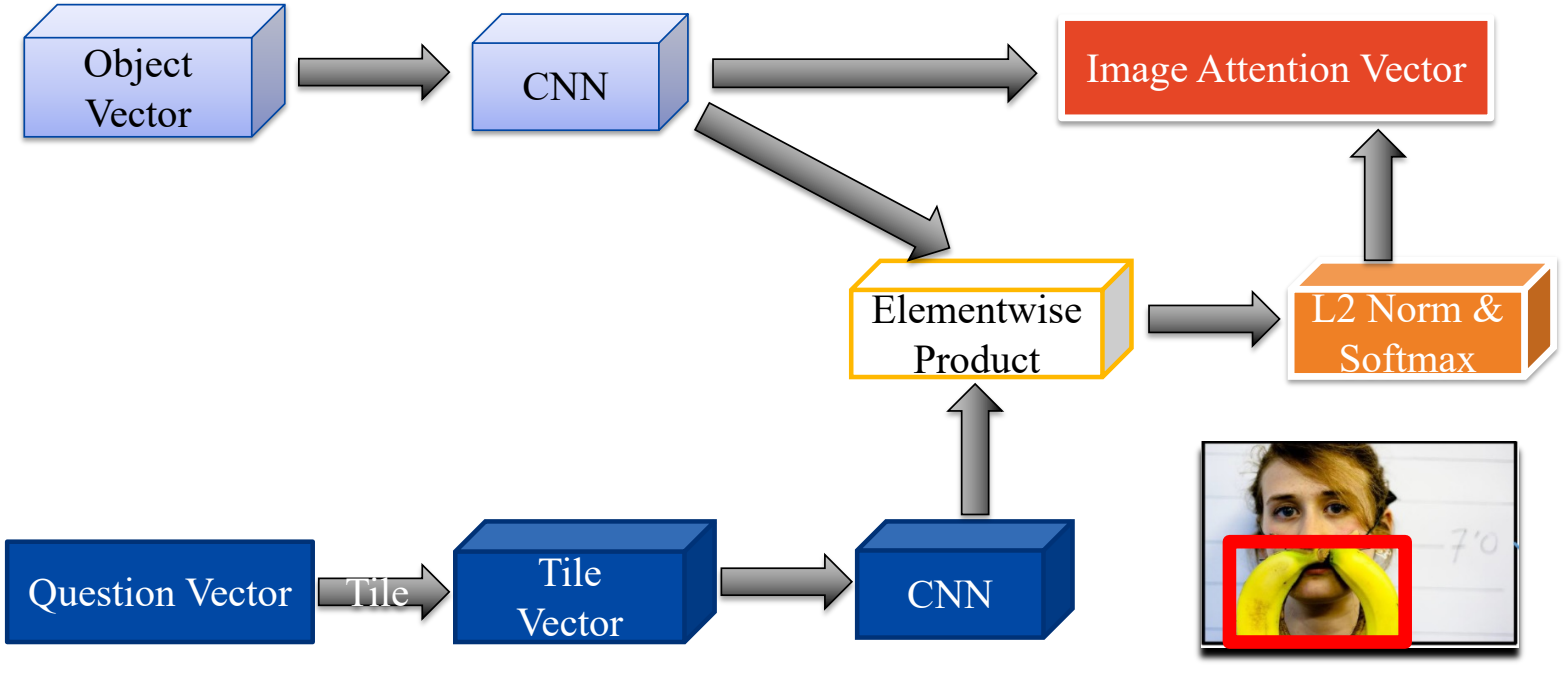
VQA

- Object Detection

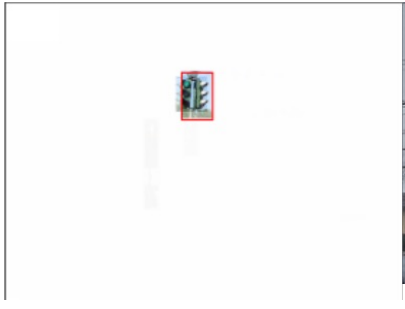


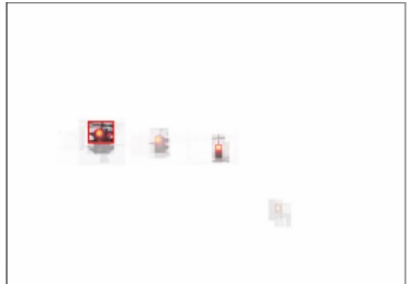
VQA

Image Attention

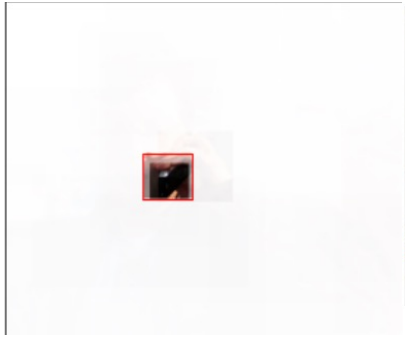


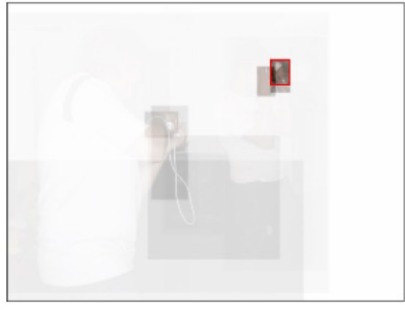
VQA



What color is illuminated on the traffic light?

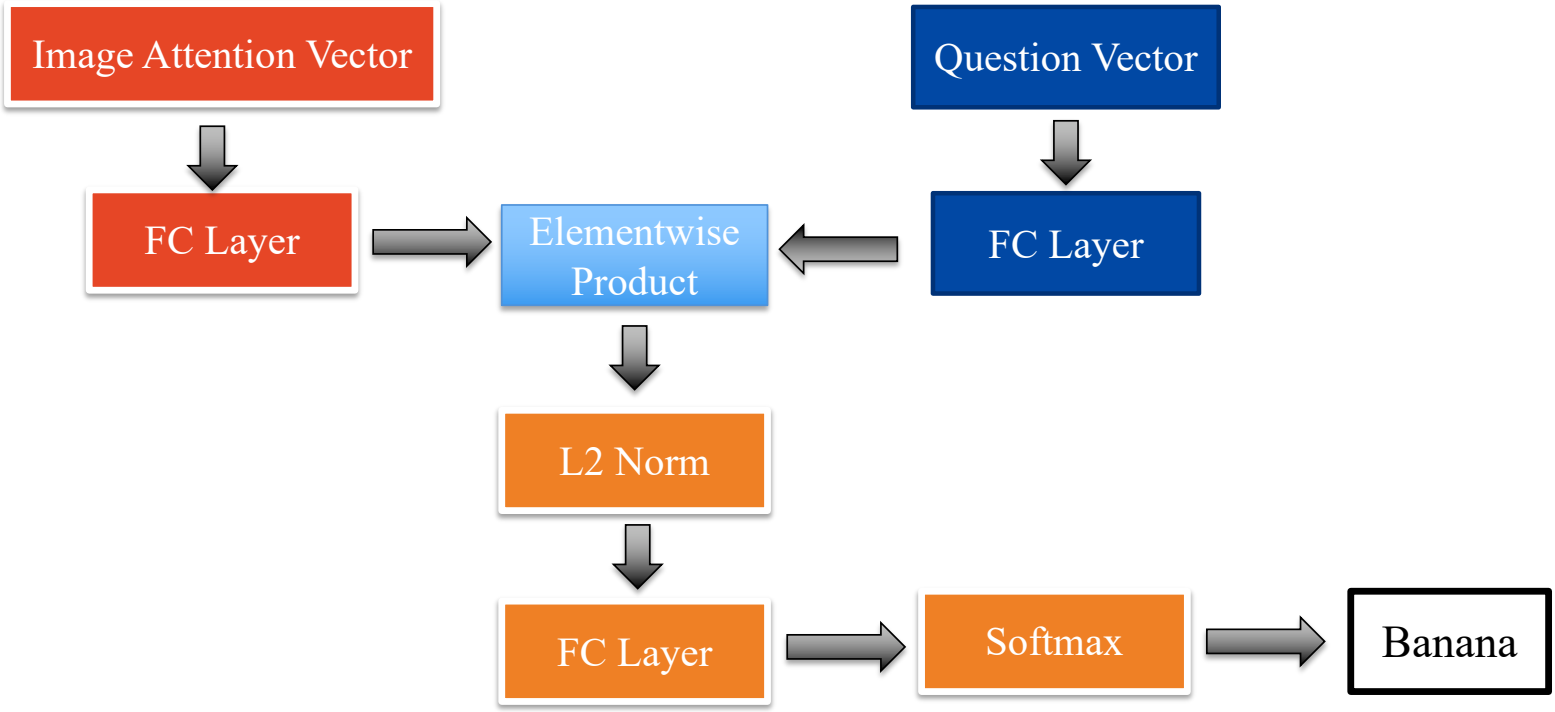


What is the man holding?

Image credit to Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering

VQA

- Question & Image Fusion



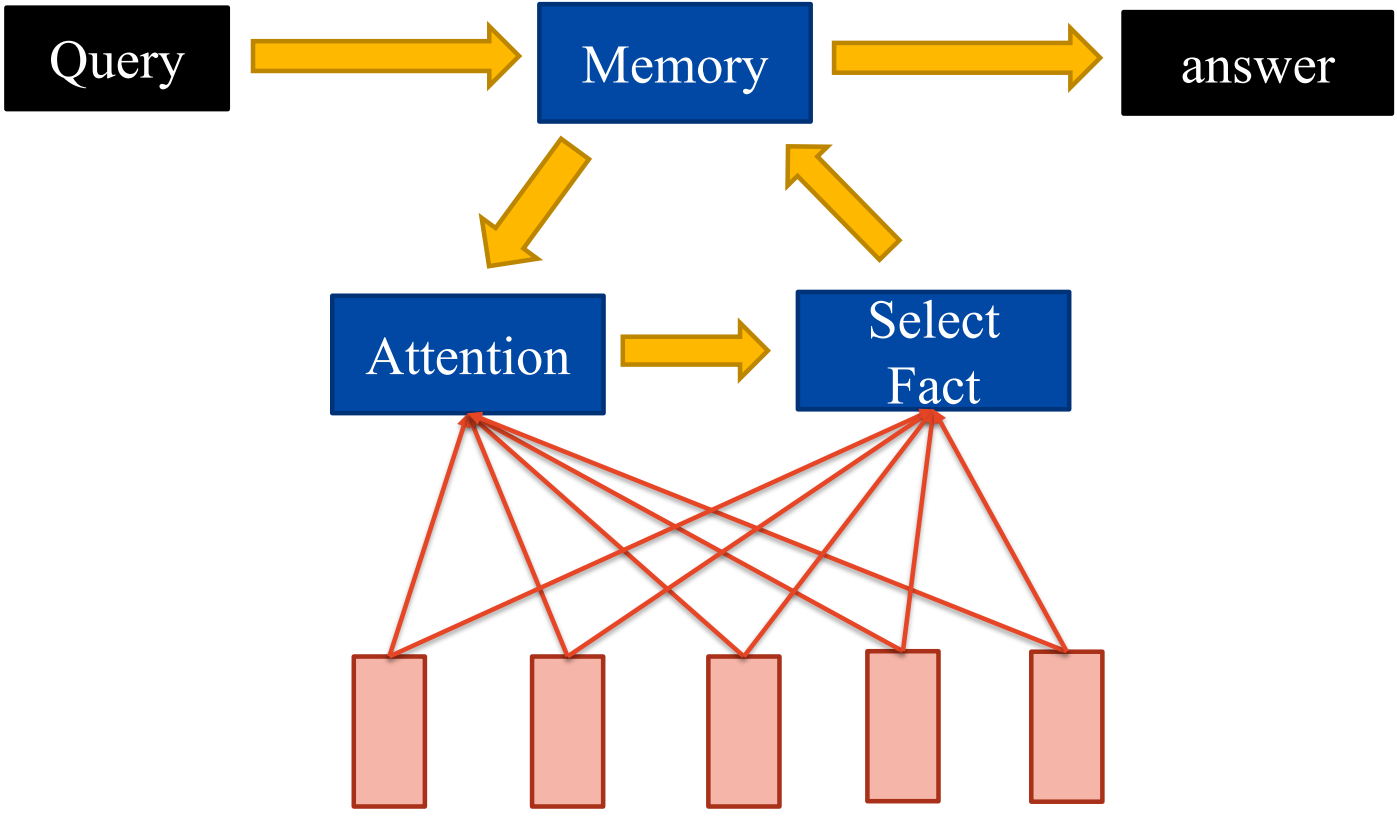
Reading Comprehension

- Answering question based on given facts
- Example

A. Brian is a frog.
B. Lily is gray.
C. Brian is yellow
D. Julius is green.
E. Greg is a frog

✓ What color is Greg?

Reading Comprehension



Each Fact becomes a vector by LSTM

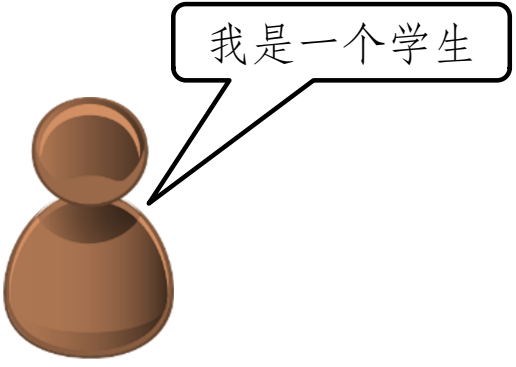
Reading Comprehension

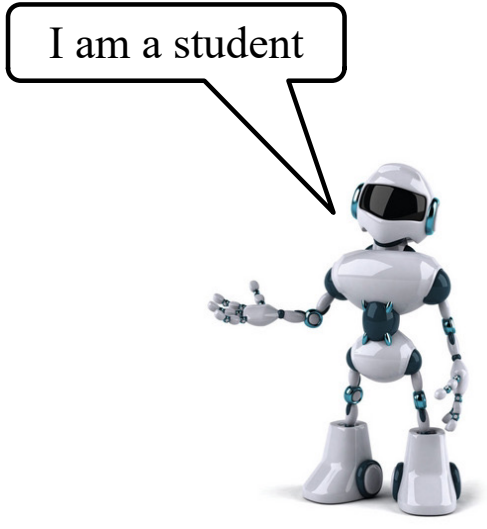
Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. NIPS, 2015.

Seq2seq

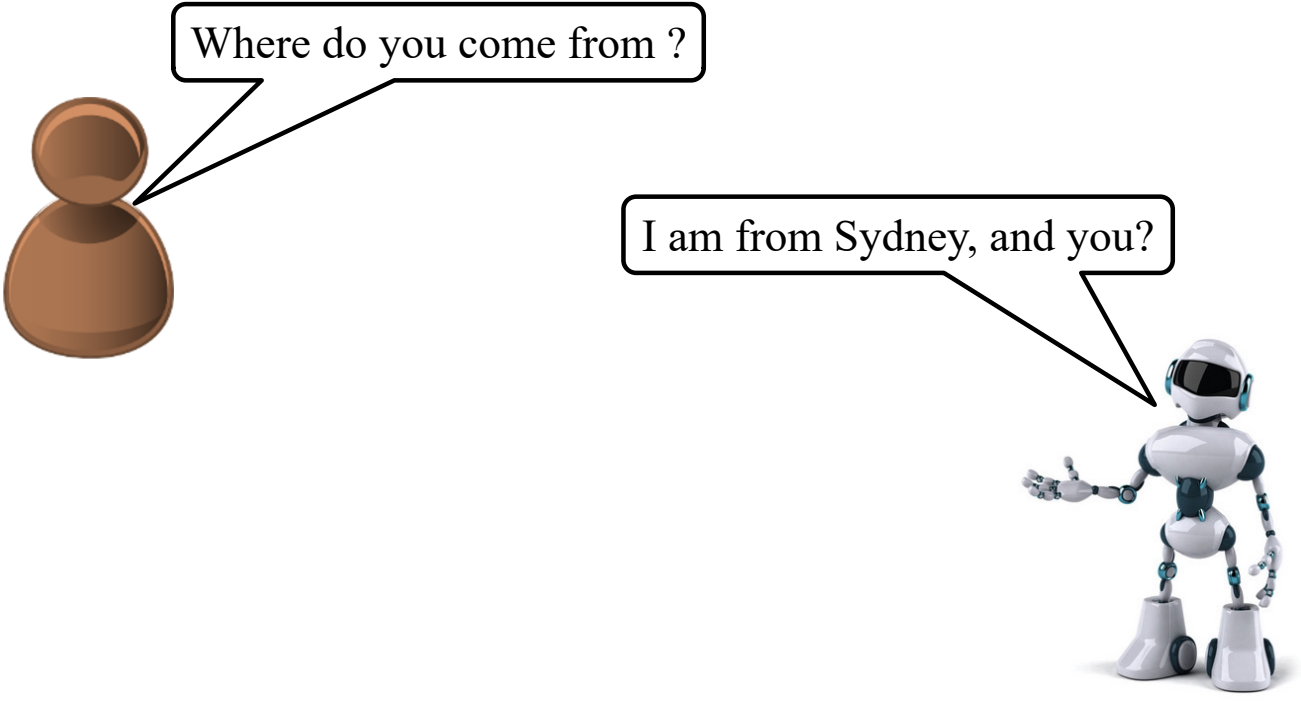
- Language Translation



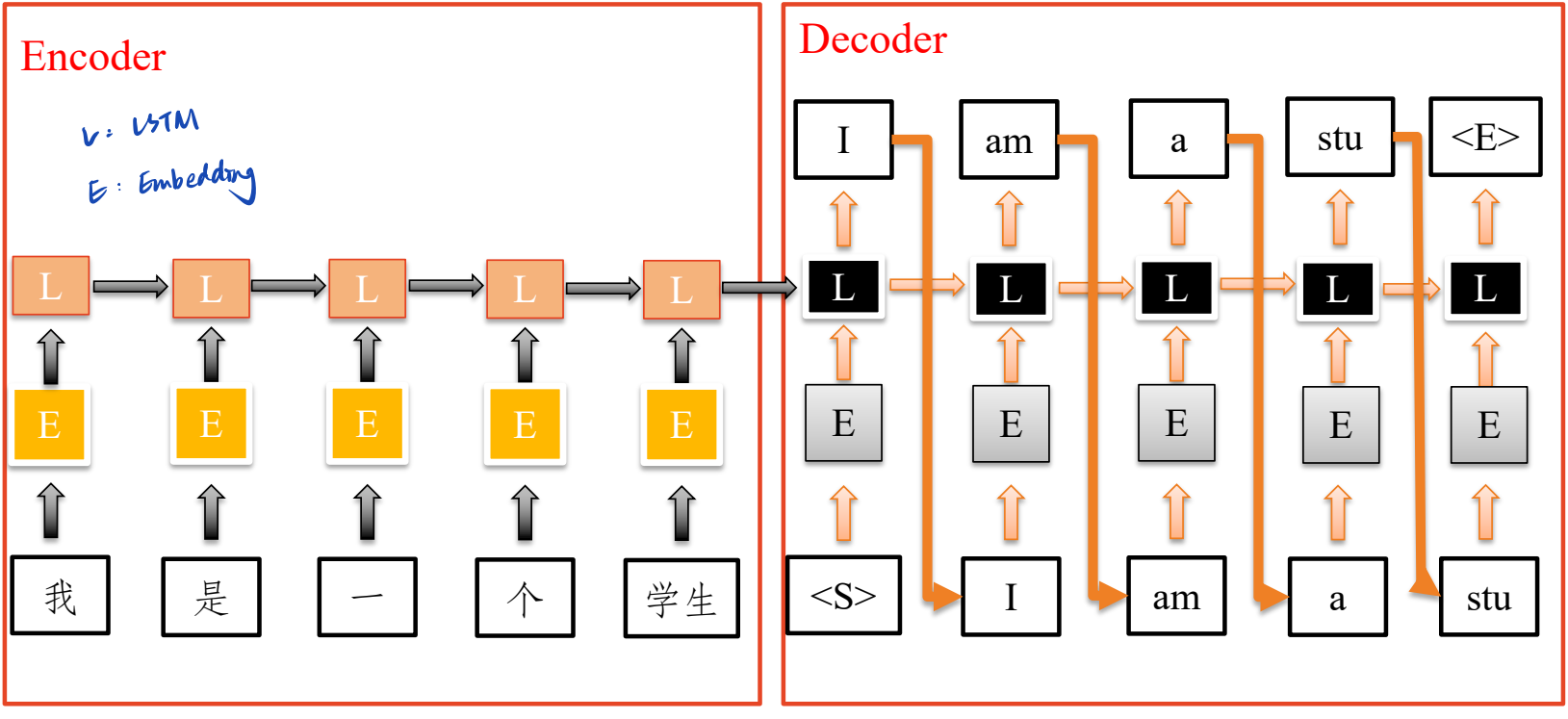


Seq2seq

- Chat Robot

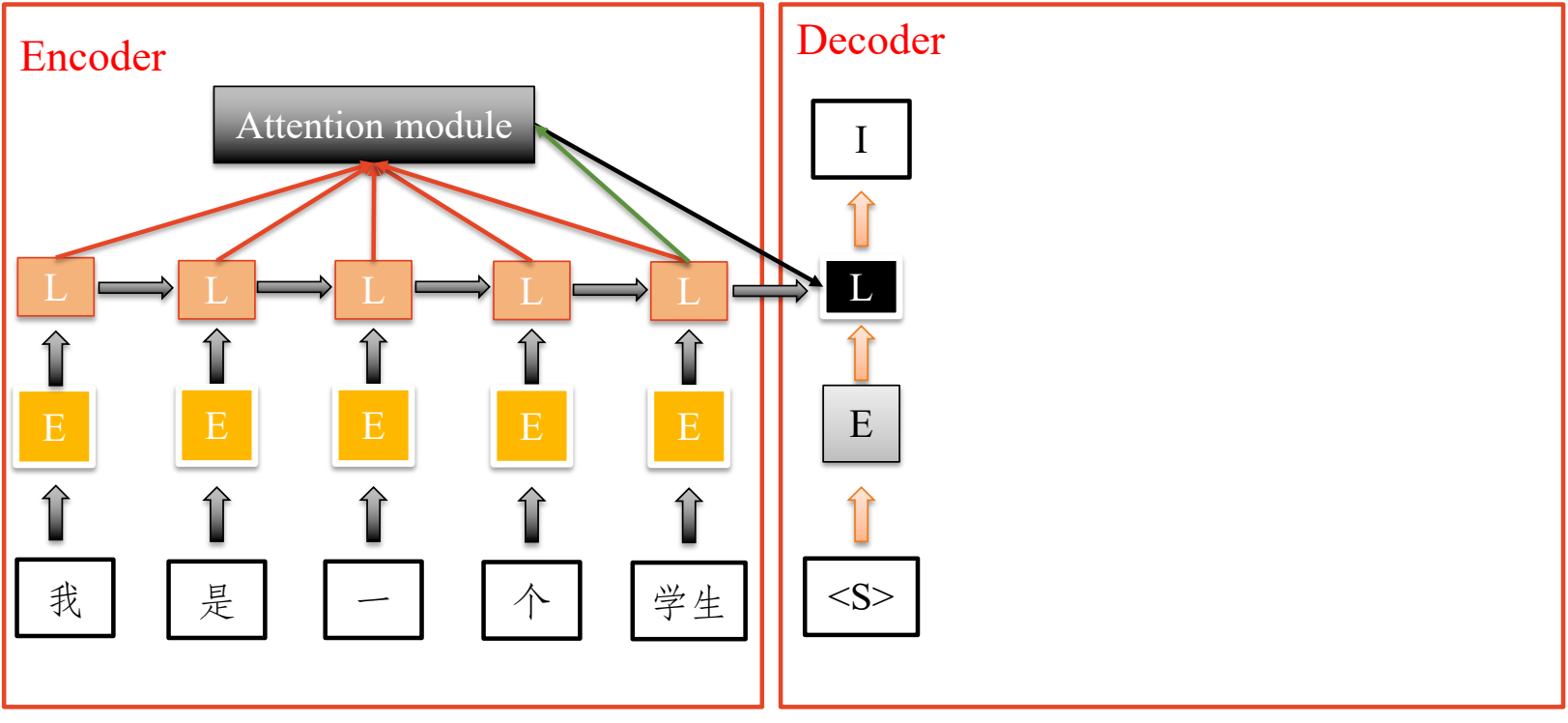


Seq2seq


Encoder
L: LSTM
E: Embedding
L
L
L
L
L
E
E
E
E
E
我
是
一
个
学生
Decoder
I
am
a
stu
<E>
L
L
L
L
L
E
E
E
E
E
<S>
I
am
a
stu

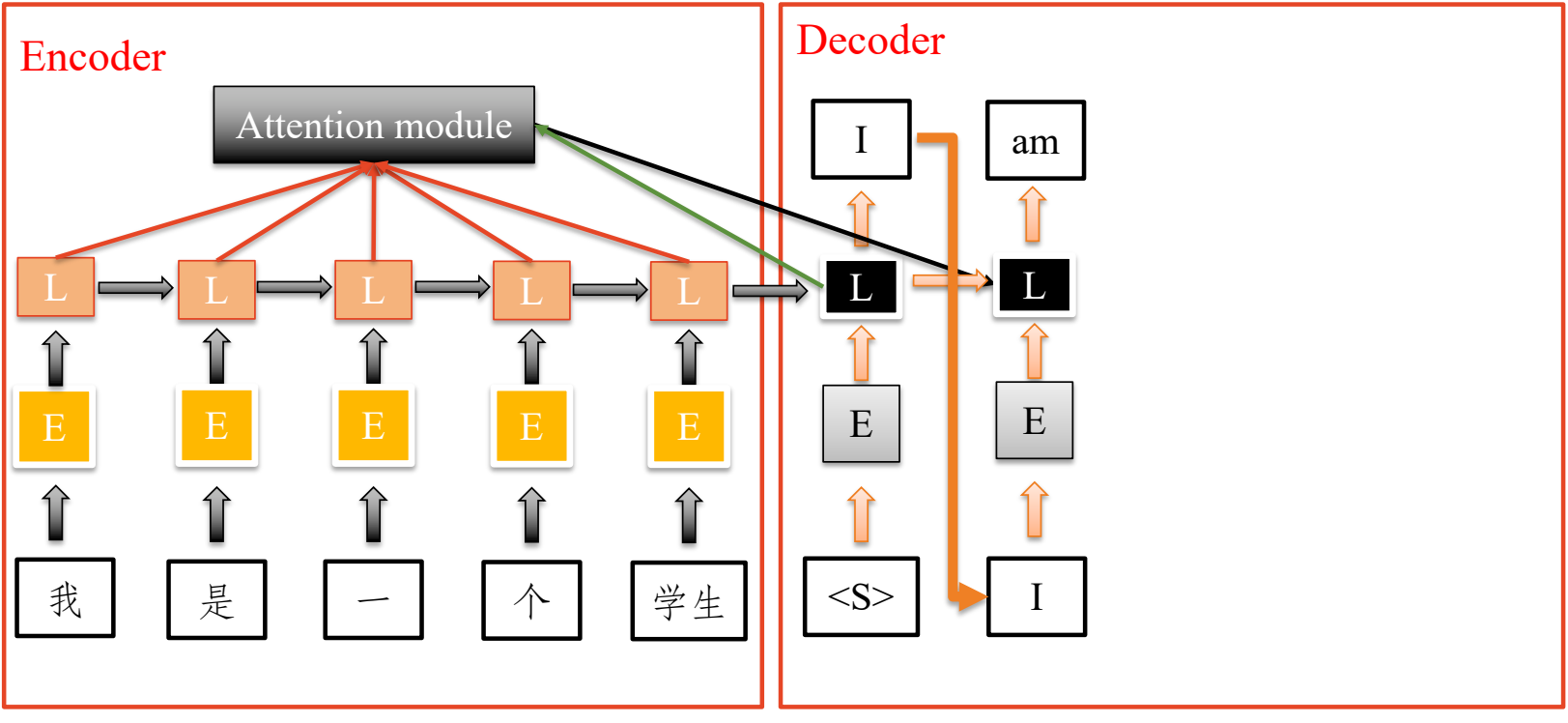
Seq2seq

- Attention



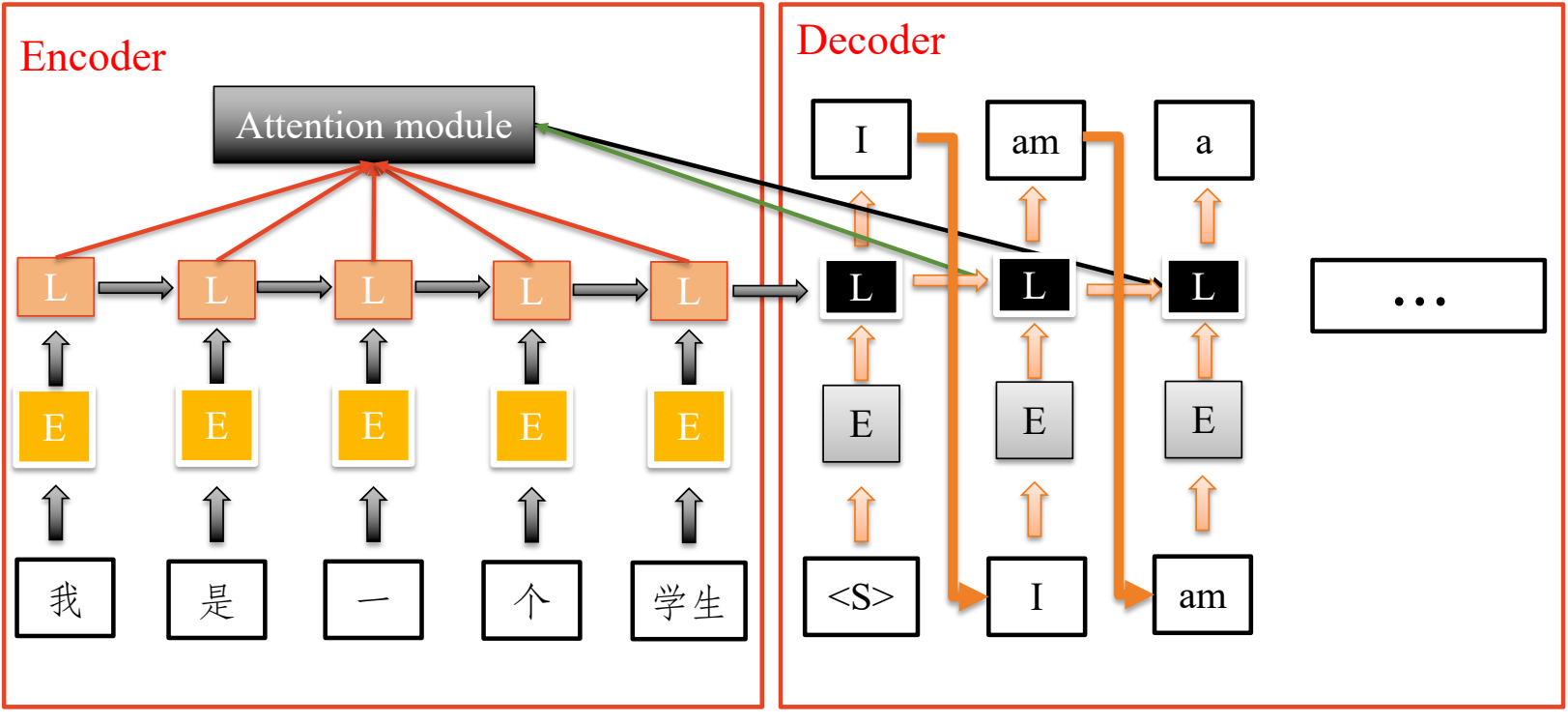
Seq2seq

- Attention



Seq2seq

- Attention



Seq2seq

- Two stage
  - Encode stage
  - Decode stage
- Attention
  - Weight for each encode symbol in decode step
  - Another input of LSTM represented as a_t
  - $f_t = \sigma(w_{fh}h_{t-1} + w_{fx}x_t + w_{fa}a_t + b_f)$
  - $i_t = \sigma(w_{ih}h_{t-1} + w_{ix}x_t + w_{ia}a_t + b_i)$
  - $o_t = \sigma(w_{oh}h_{t-1} + w_{ox}x_t + w_{oa}a_t + b_o)$
  - $\widetilde{c_t} = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + w_{ha}a_t + b_h)$

Seq2seq

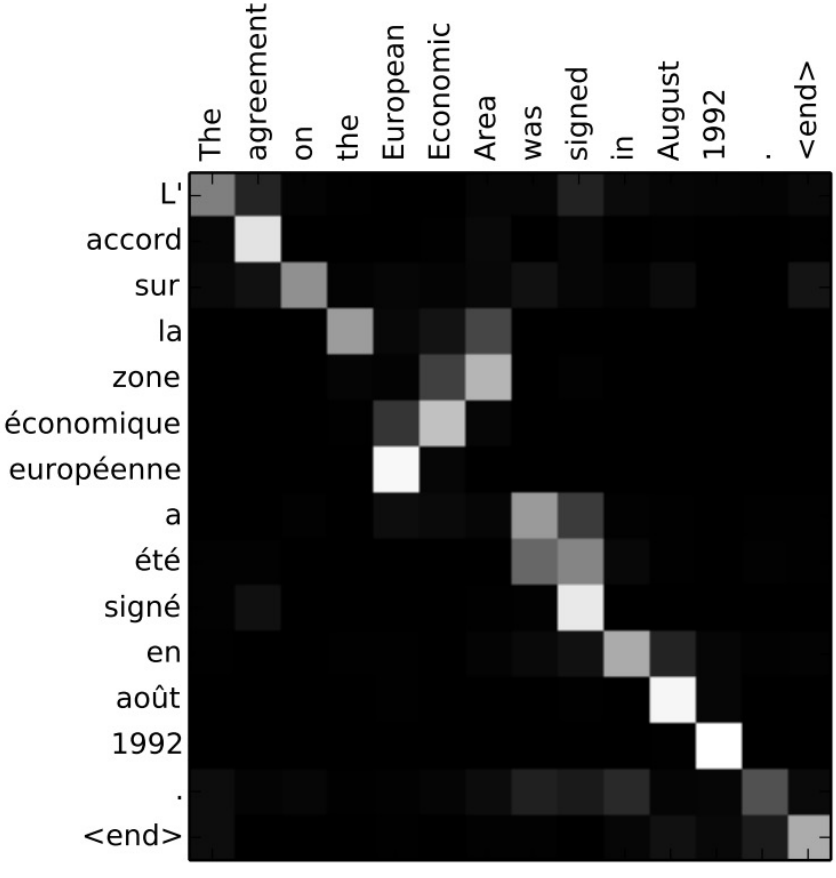


Image credit to Neural Machine Translation by Jointly Learning to Align and Translate

Thank you !