Automatic Diagnosis of Parkinson's disease using Deep Learning

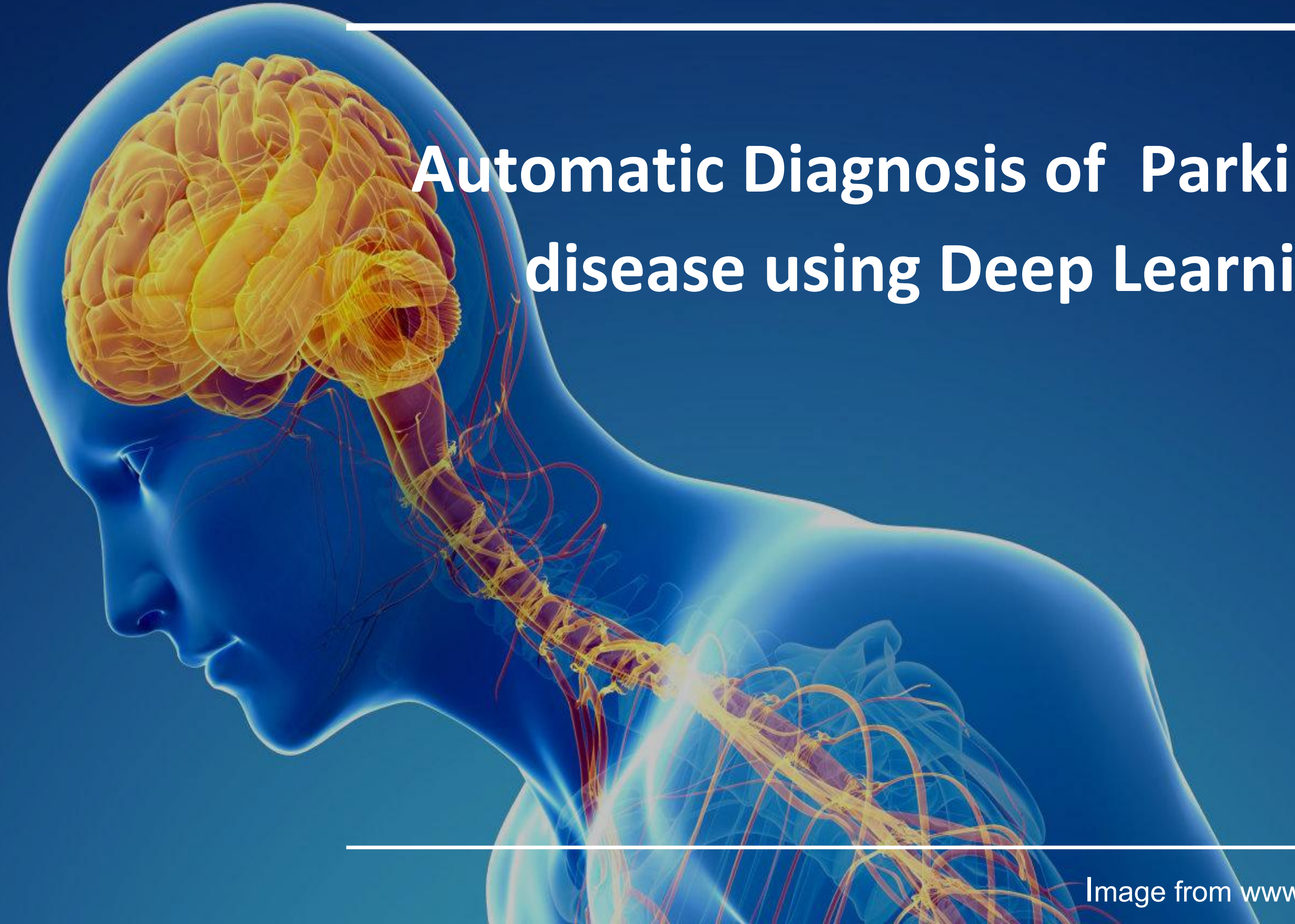


Image from www.express.co.uk

Outline:

- Problem setup
- Data processing - Efrén
- Classification - Denny
- Future Work - Yonghua

Study Team

Initially Sponsored By Dr. Benzi Kluger Of UCH-Neurology

Keeran Maharaj - Research

Part Of Larger Study To Improve Parkinson’s Treatment

Recently Joined By Debashis Ghosh Of UCH Bioinformatics

Efren Cruz Cortés - Machine Learning

Yonghua Zhuang - Machine learning - Biostatistics PhD student

Dennis Graham - Data Science - Programmer

Introduction

Goal Is To Develop An Inexpensive And Reliable Parkinson’s Disease Diagnostic

Parkinson’s Difficult To Diagnose Because Of Imposters

- Mild Cognitive Impairment MCI
- Essential Tremor ET
- Parkinson’s Dementia PD+
- Parkinson’s PD

What is MEG?

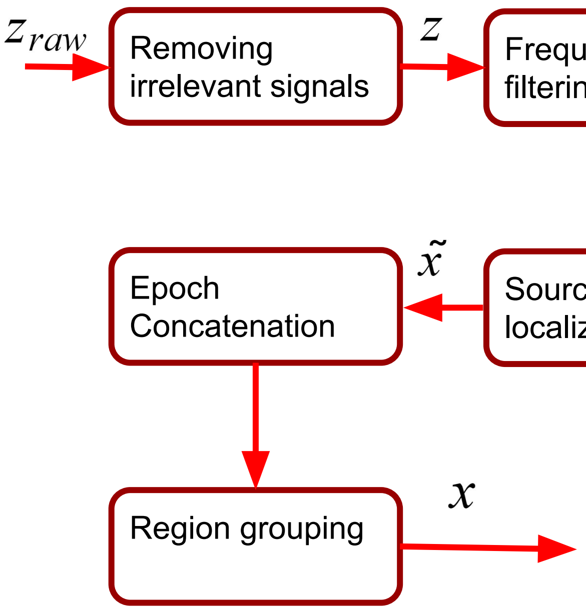
Measures magnetic fields of the brain

Array of magnetometers outside skull

Similar to EEG

Image from Wikimedia

Preprocessing flowchart



Removing Irrelevant Signals That May Be Relevant

- Independent Component Analysis
- Eyeblink and heartbeat signatures are examples but ‘temporal muscle activity’ is included.
- Software: EEGLAB

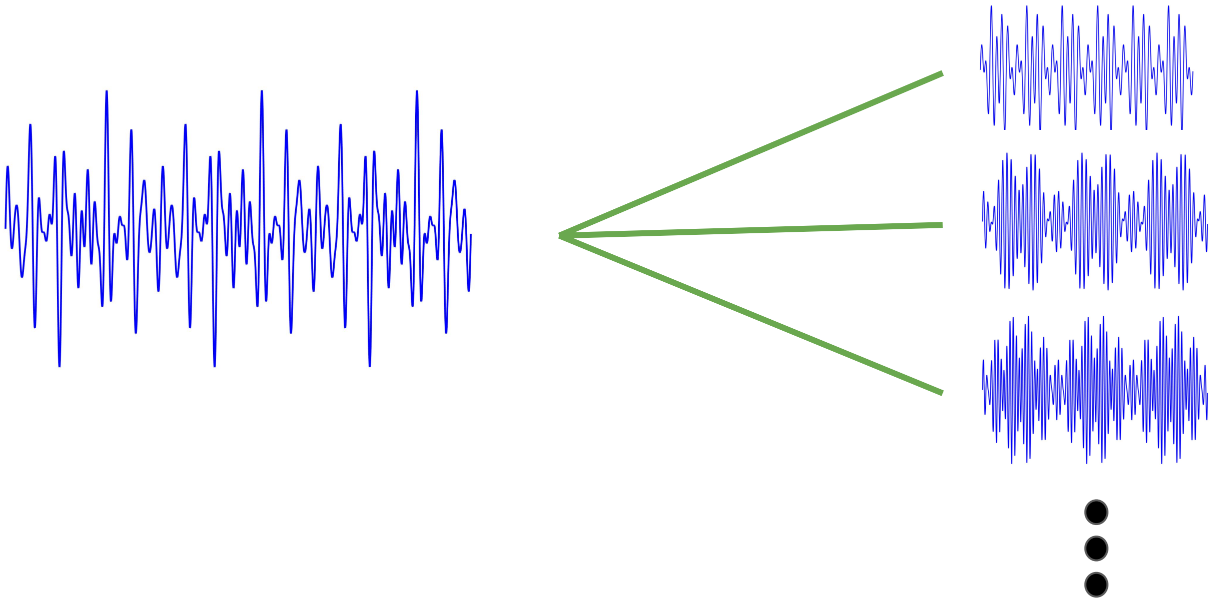
Comments:

Eyeblink and temporal muscle activity patterns will differ from PD to non-PD patients. Make sure relevant signal is not removed or do not use.

From EEGLAB documentation: The quality of the data is critical for obtaining a good ICA decomposition. ICA can separate out certain types of artifacts -- only those associated with fixed scalp-amp projections. These include eye movements and eye blinks, temporal muscle activity and line noise.

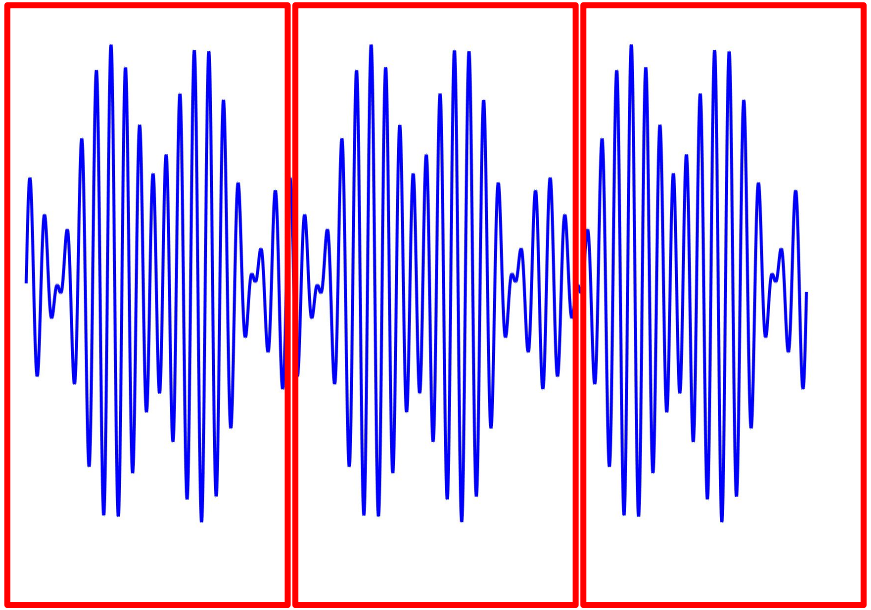
Frequency Filtering

- Bandpass filter over 5 frequency bands:
- [4,8], [8,10], [10-13], [13-30], [30-56] (“brain waves”- delta, alpha, etc.)



Epoch segmentation

- Segment into 5 second epochs
- Allows for removal of artifacts

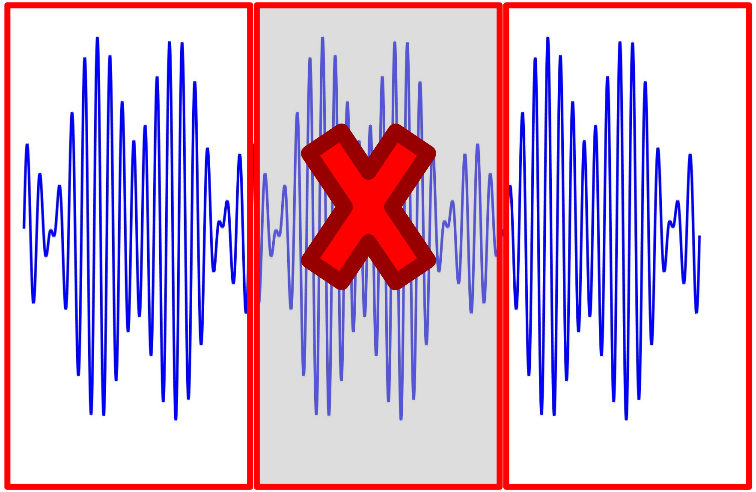


Gross artifact rejection

- Delete epochs where gross artifacts (e.g. head bump) are identified

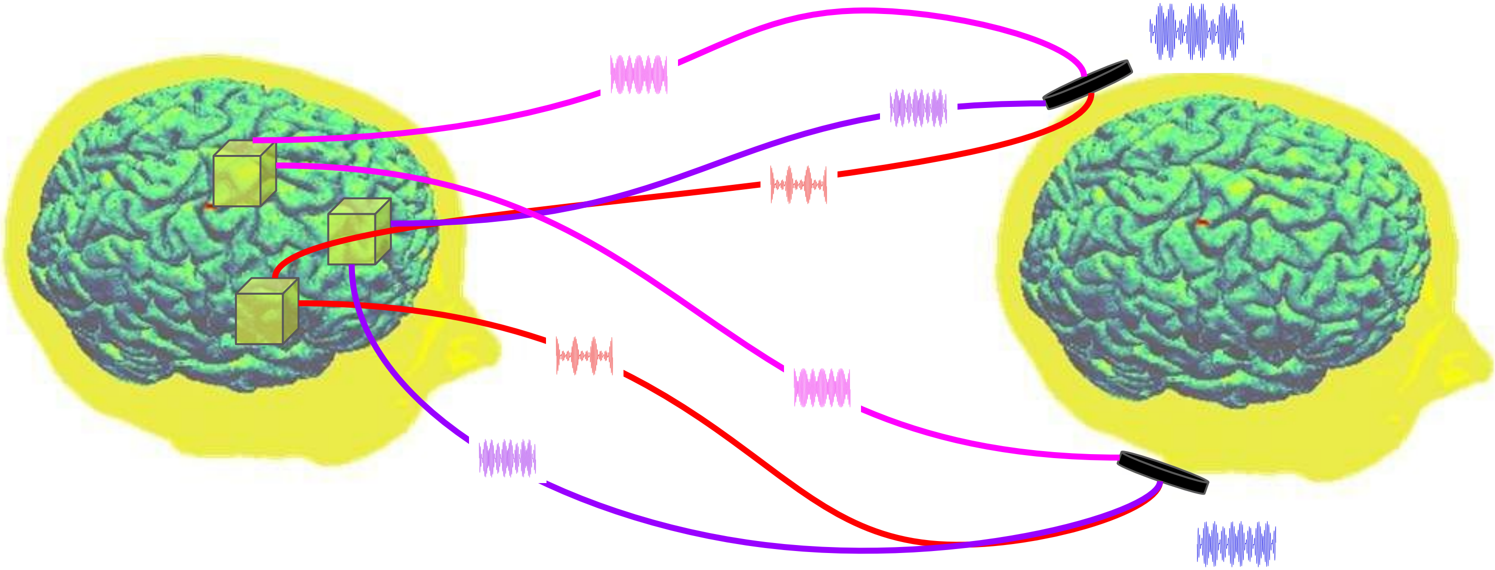
Questions:

- Deleted epochs are replaced by ?



Source localization

- Blind Source Separation Problem (similar to ICA)
- Recover brain region signals from sensor signals



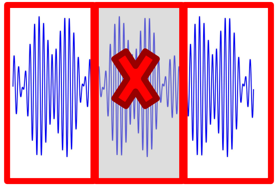
Brain Image Courtesy of Luis Gomez, Duke University

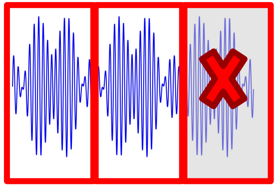
Epoch concatenation

- Concatenate surviving epochs

Questions:

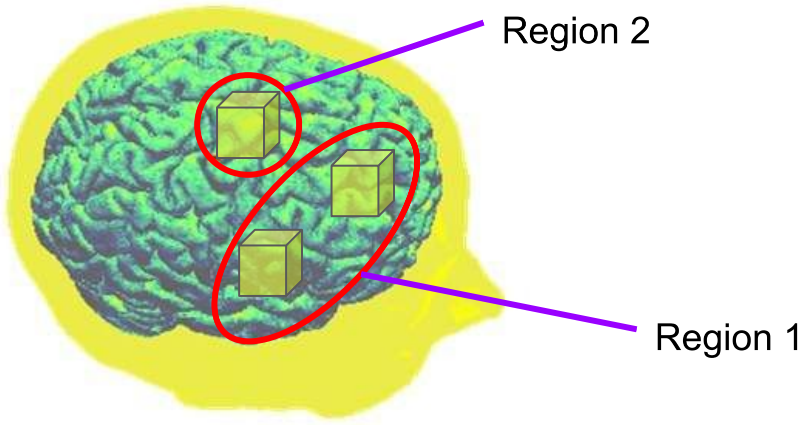
- How are patients commensurate after asynchronous concatenation?





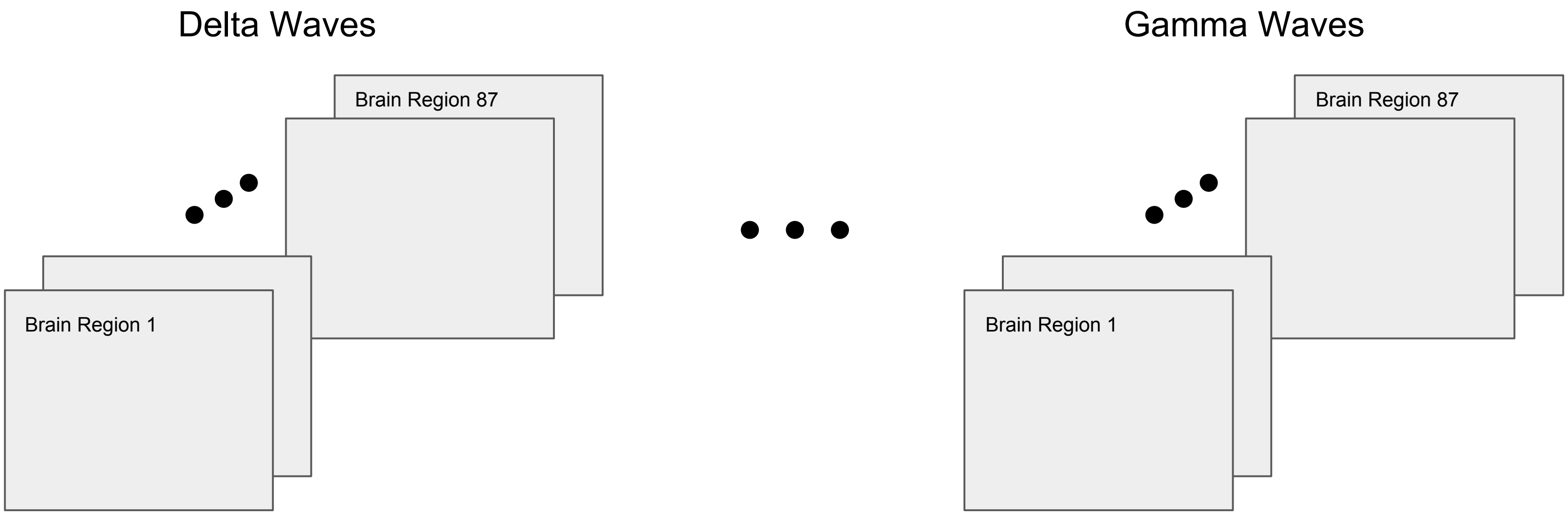
Region grouping

- From source localization obtain MNI coordinates
- From MNI coordinates obtain brain region (e.g. premotor gyrus)
- 87 neural regions



Preprocessed data - final form

For each patient, 5 sets of 87 matrices, each corresponding to a neural region, with time signal of points in that region. Each set is a frequency band.



Development Process

Small Sample Of 90 Including PD,PD+,ET,MCI Plus Control Samples

Initially The MEG Data Was Processed Using EEGLAB

- No Classification Results Could Be Achieved

Decision To Apply Machine Learning

- SVM, Random Forests, Logistic Regression Failed To Classify

Decision To Apply Neural Networks

- Results Achieved With Dense ANN

Preprocessing Inhibits AI Processing

Inherent Part Of This Study But To Be Deleted In Future Study

Uses Standard Signal Processing Based On Frequency

Preprocessing Removed Data That Could Be Used By AI Techniques

AI Uses Time Dependent Events

PD IS A MOVEMENT DISORDER THAT IS ERRATIC

Expectation Should Be That Symptoms Generate Time Dependent Signals

In Different Brain Regions

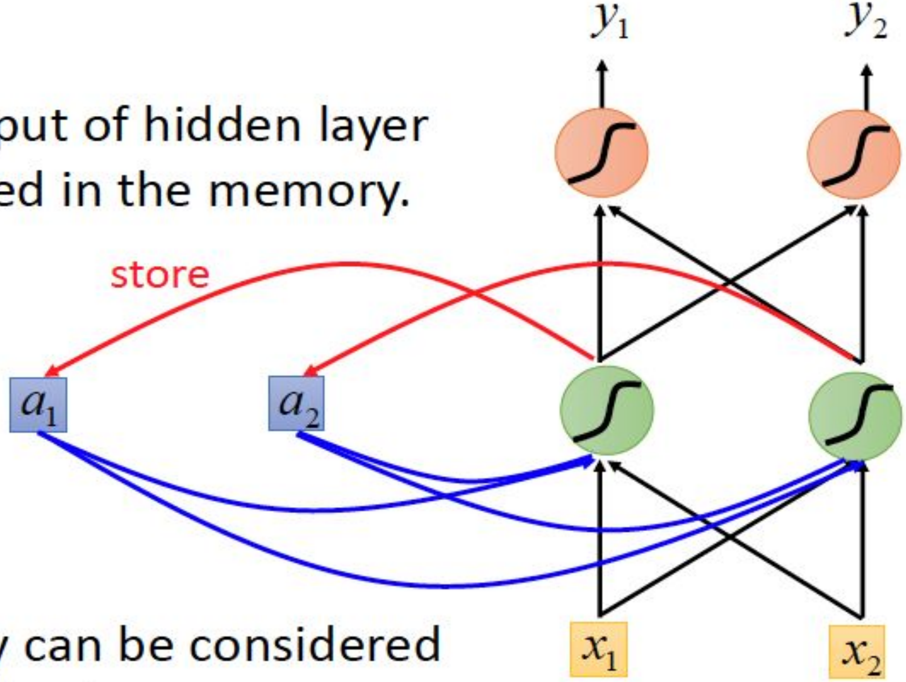
That Are Random And Appear As Irrelevant Artifacts

Future directions

- Long short-term memory Neural Network (LSTM)
- CNN with LSTM
- LSTM with Soft Attention model

Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



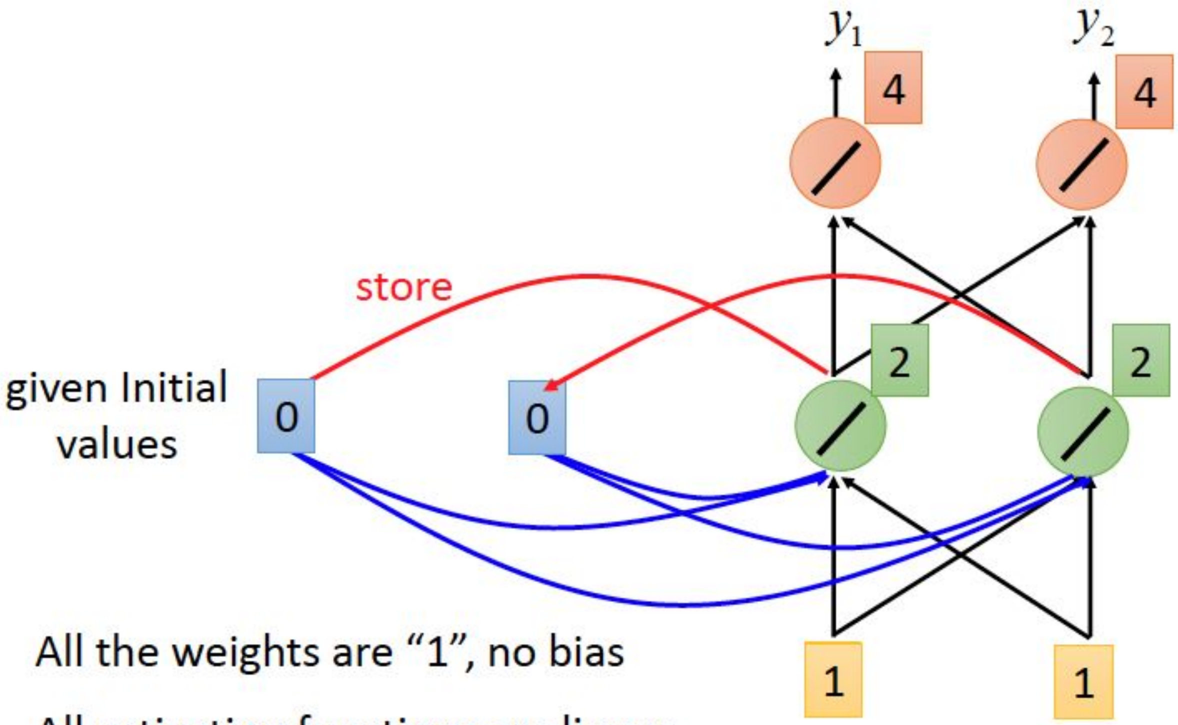
Memory can be considered as another input.

Hung-yi Le. Deep learning Tutorial. 2017.

Example

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

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



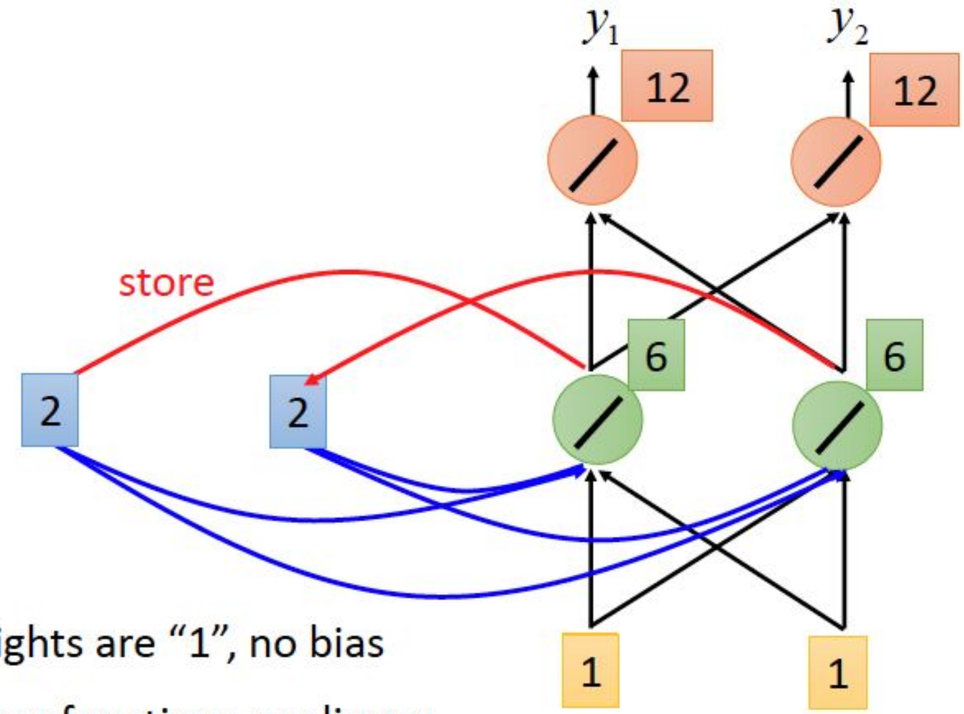
All the weights are “1”, no bias

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

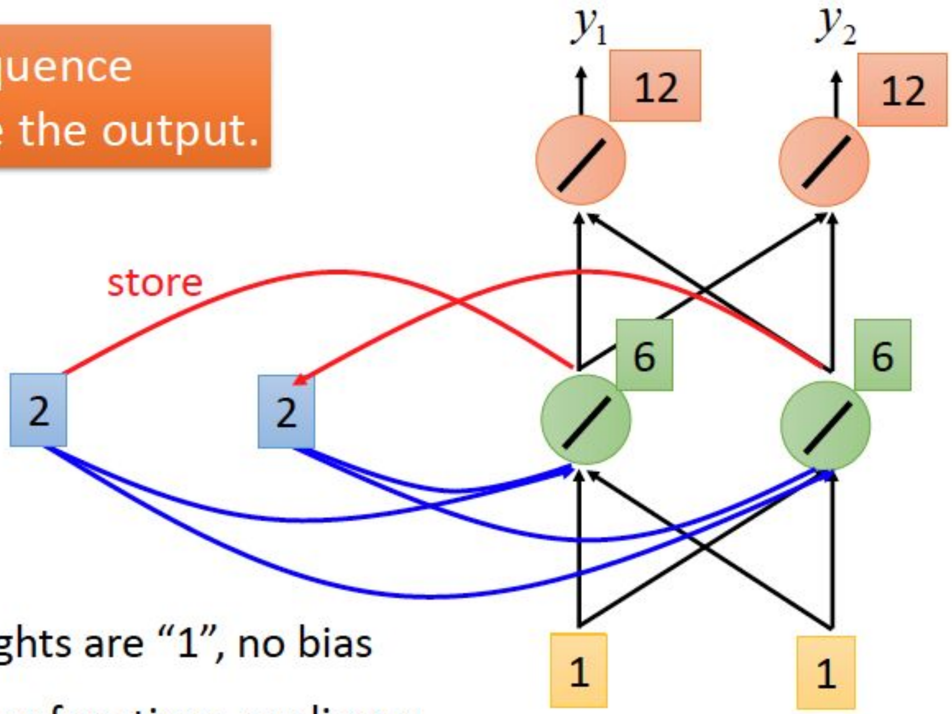
All activation functions are linear

Example

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

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

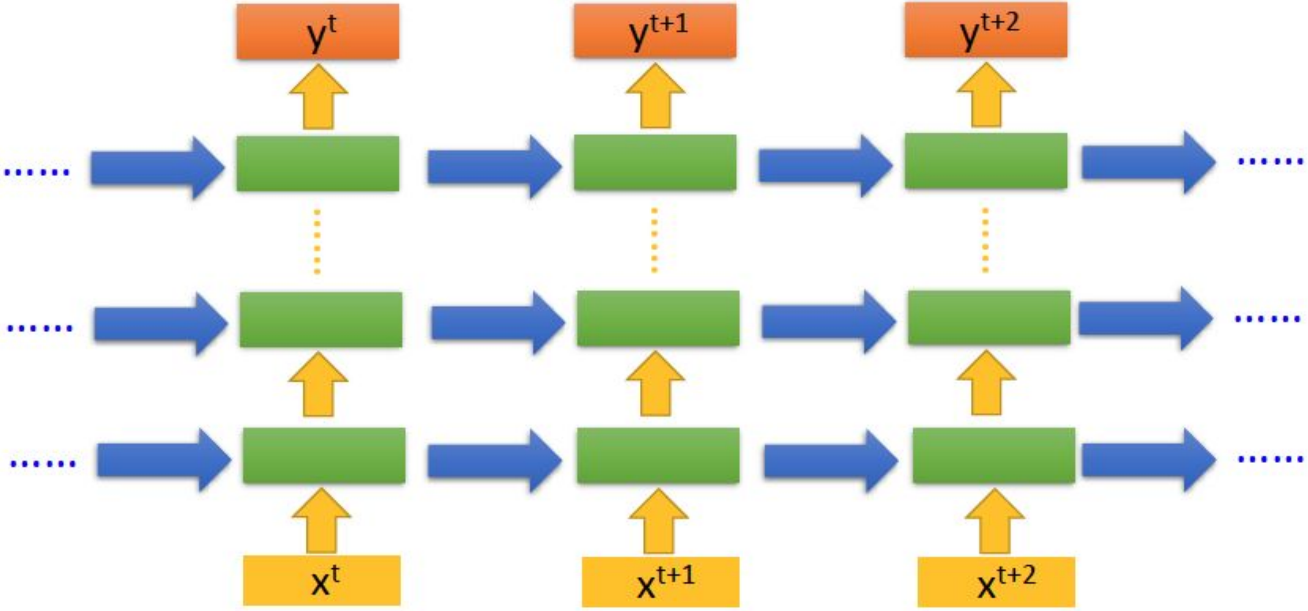
Changing the sequence order will change the output.



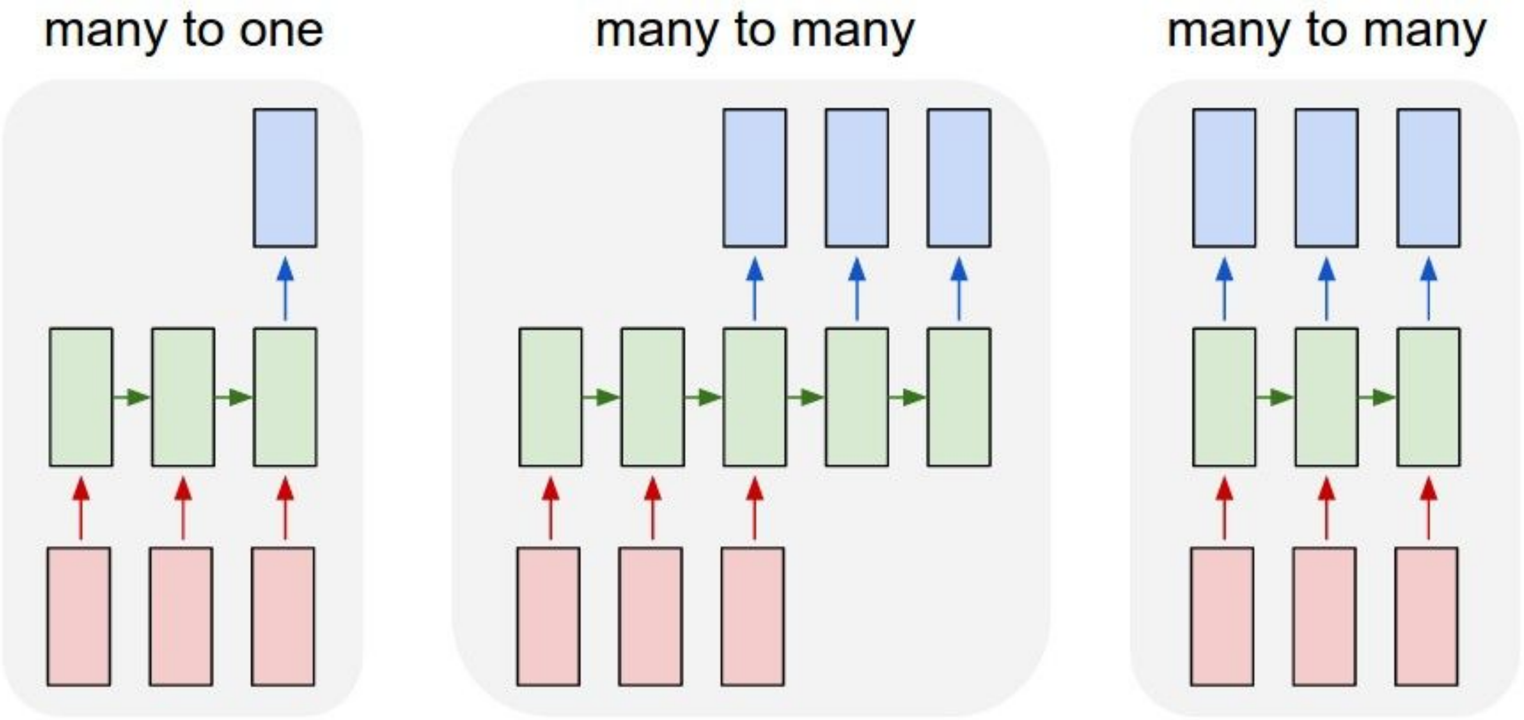
All the weights are "1", no bias

All activation functions are linear

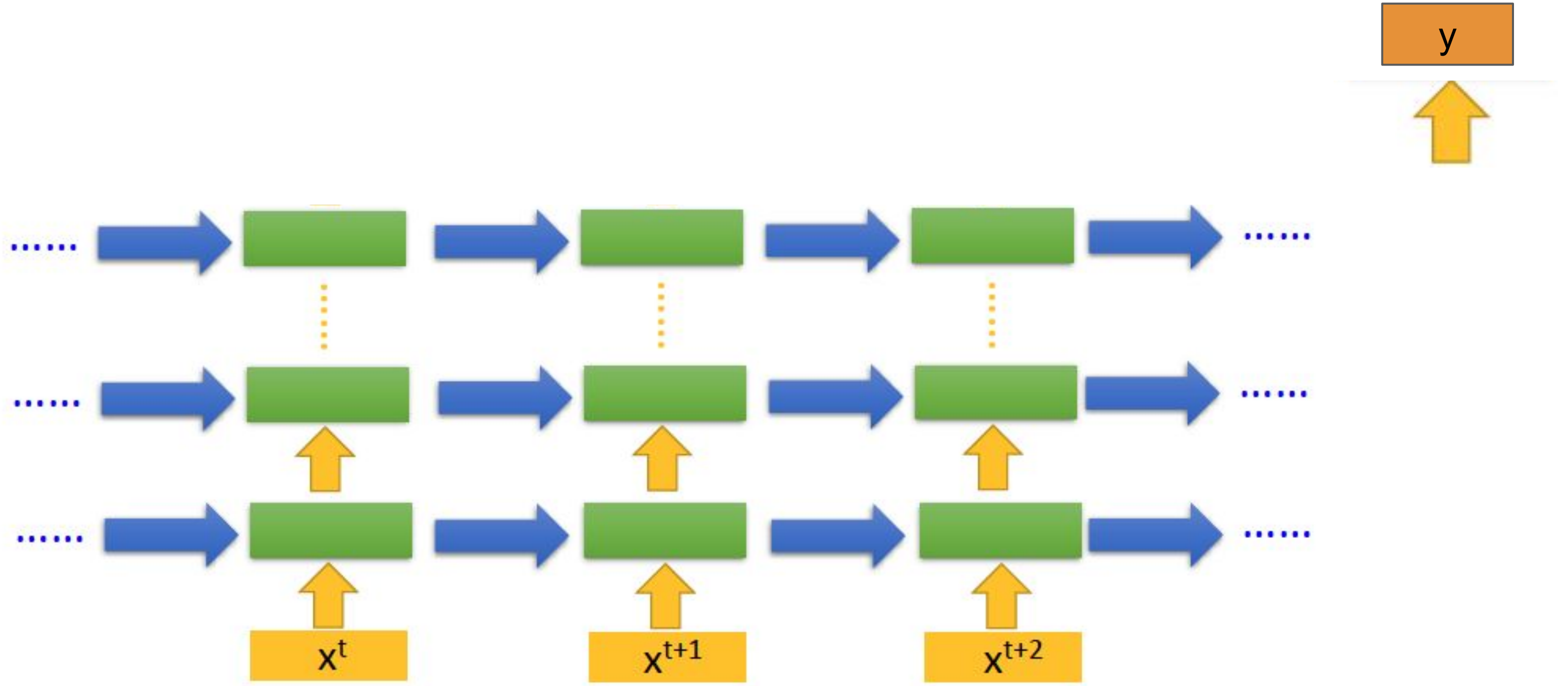
Of course it can be deep ...



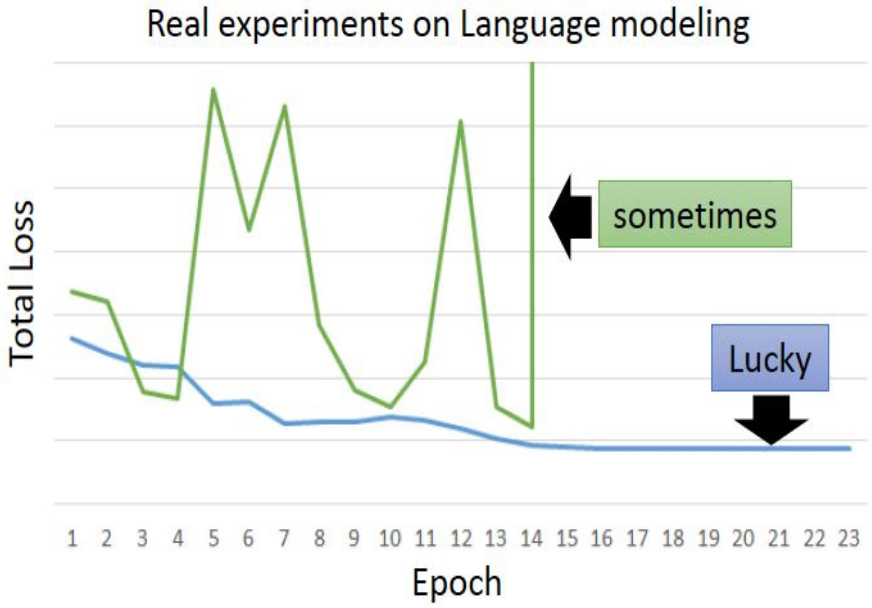
Recurrent Neural Networks



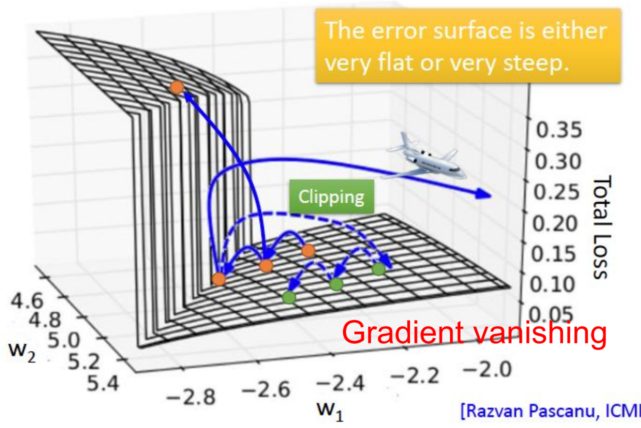
Recurrent Neural Networks (Many to one)



• RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

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

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

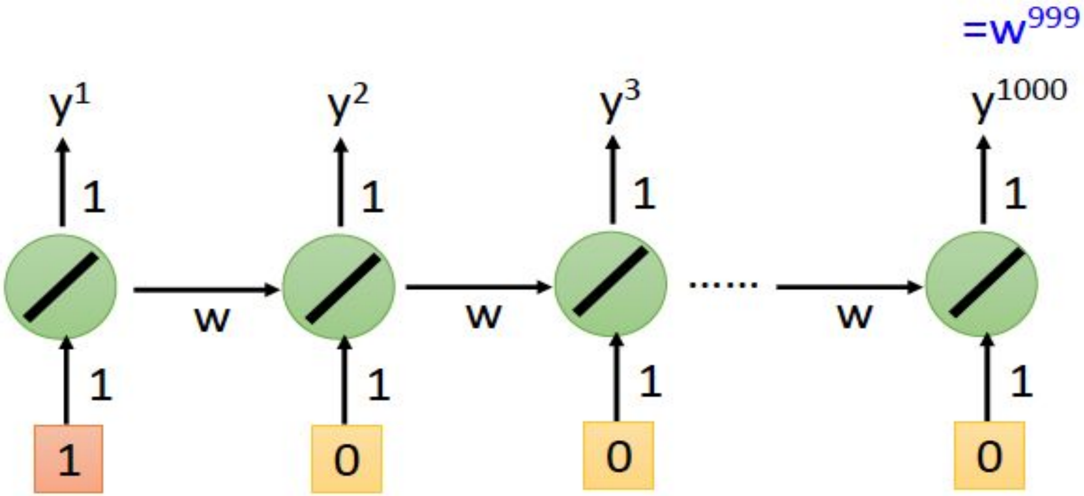
Large $\partial L / \partial w$ → Small Learning rate?

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

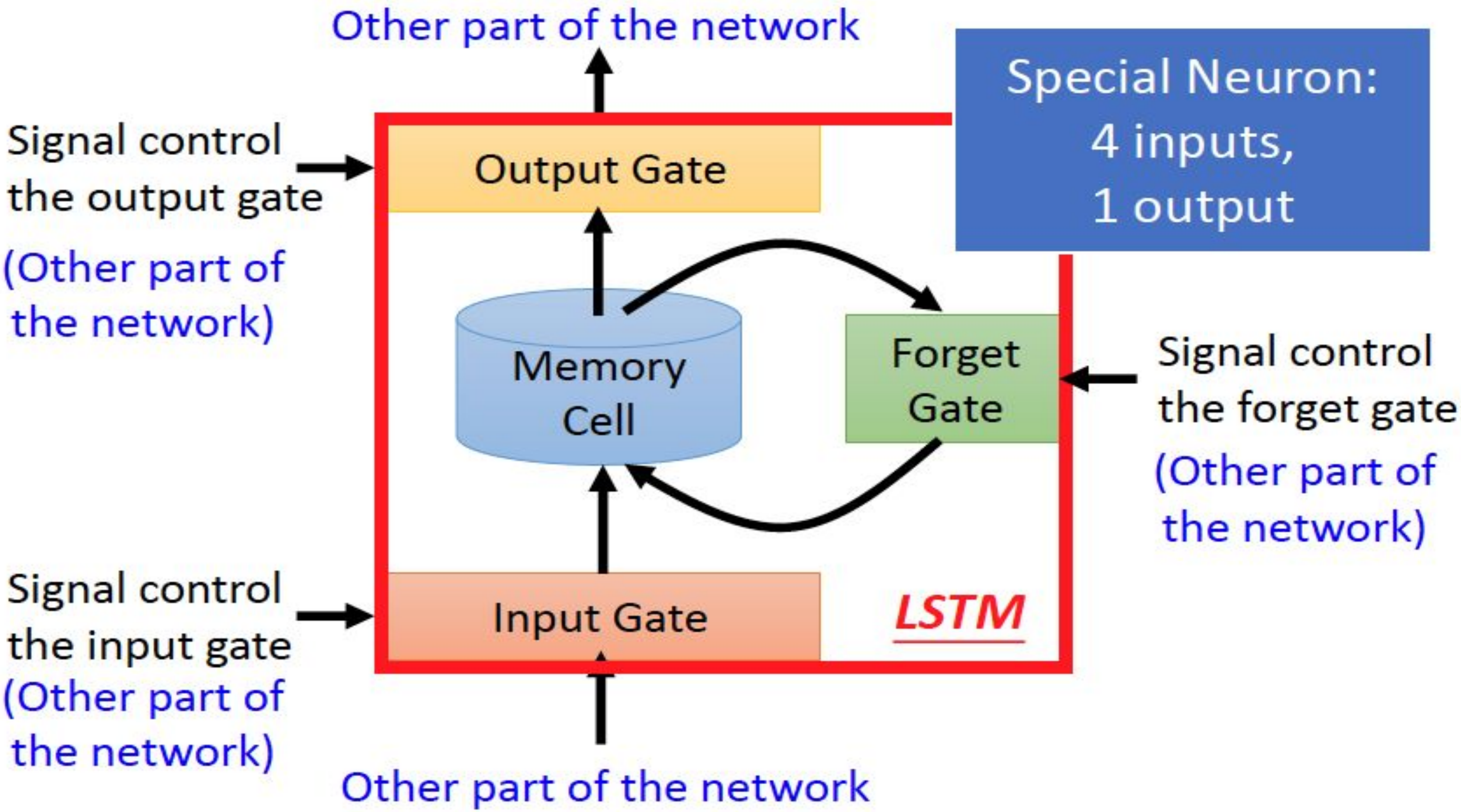
$w = 0.01 \Rightarrow y^{1000} \approx 0$

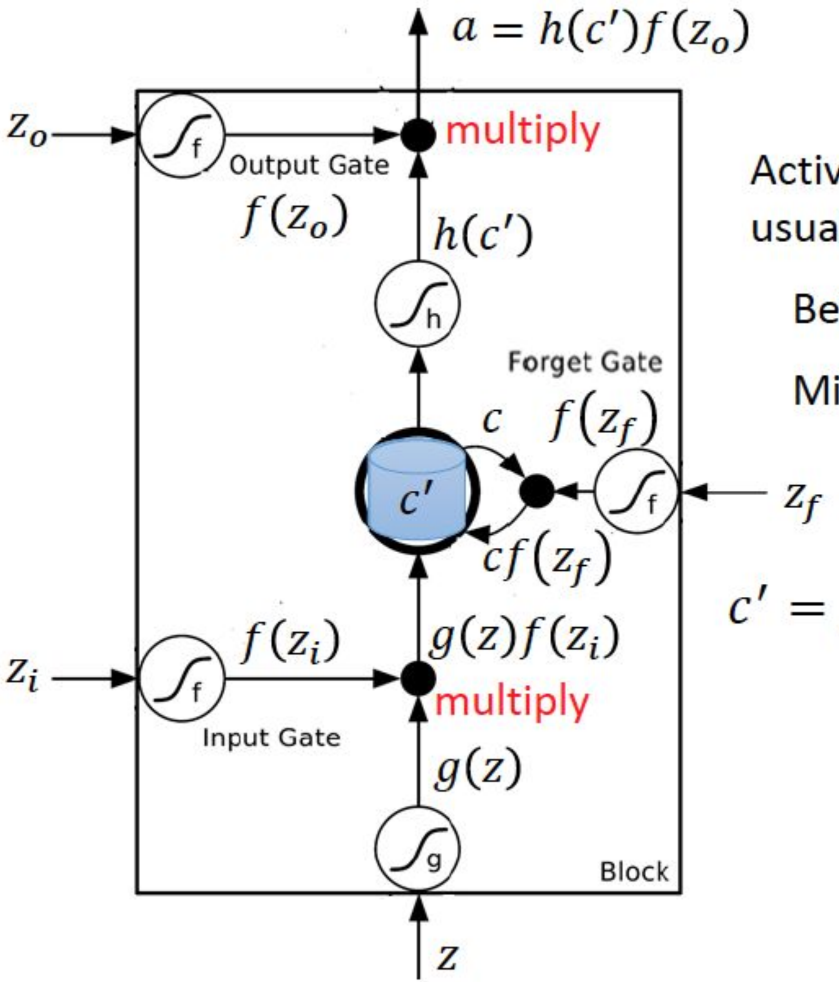
small $\partial L / \partial w$ → Large Learning rate?

Toy Example



Long Short-term Memory (LSTM)





Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

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

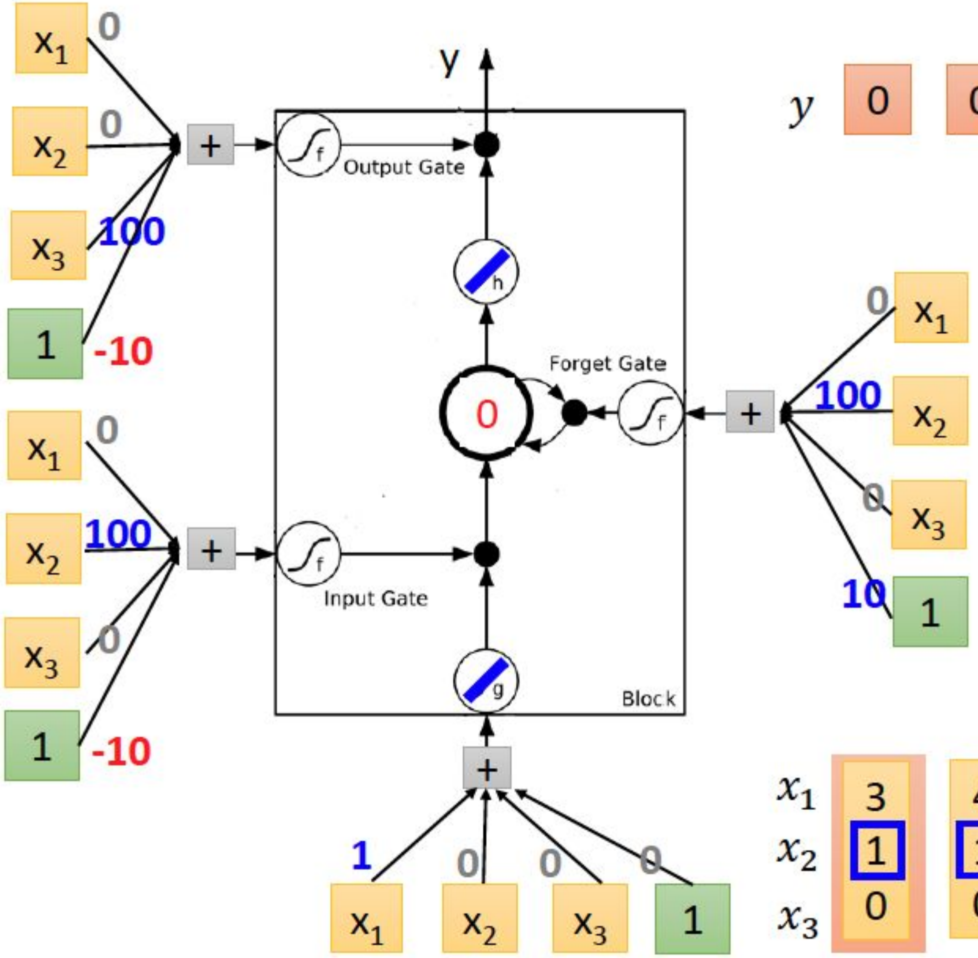
LSTM - Example

	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.

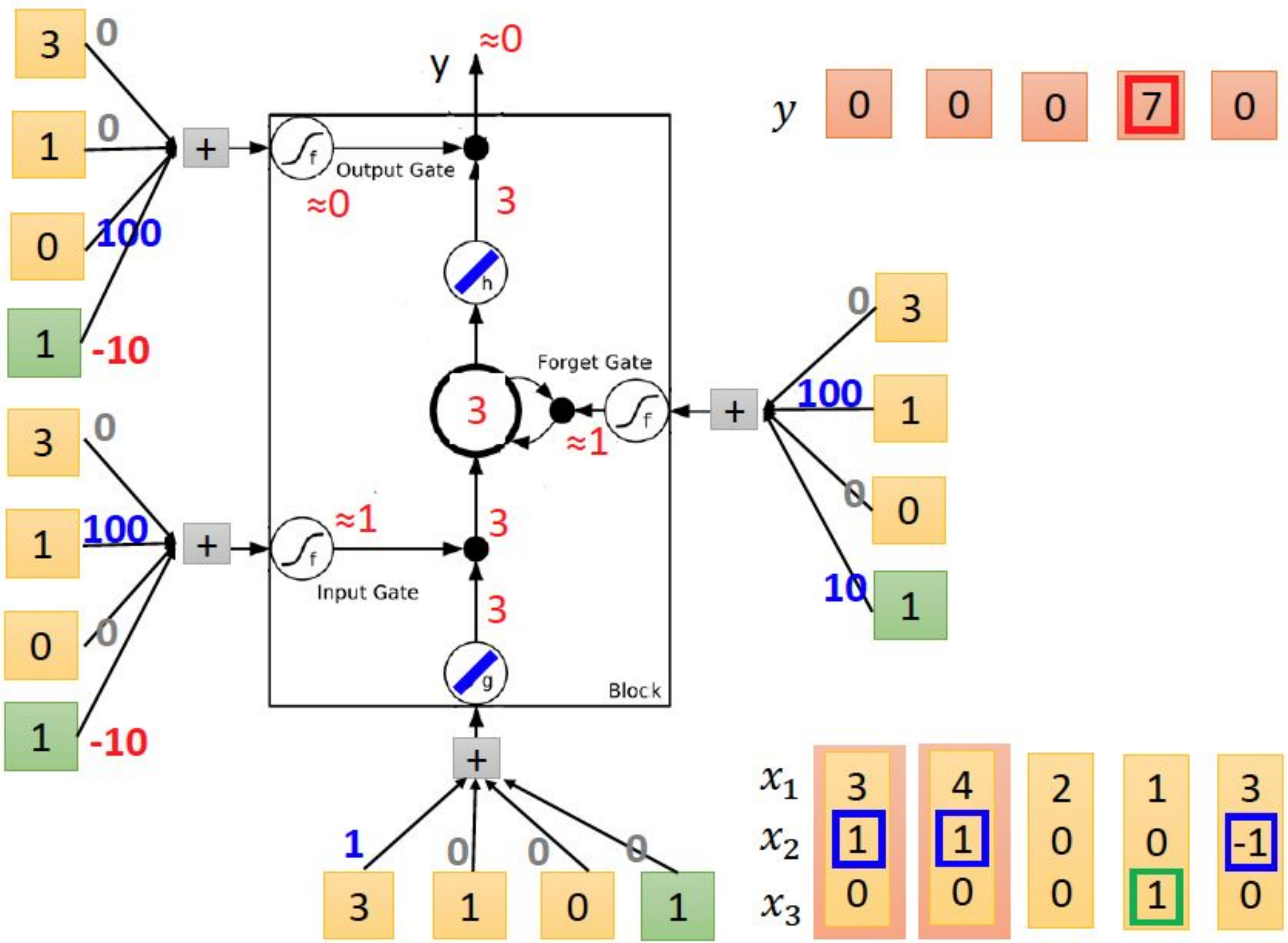


y	0	0	0	$\boxed{7}$	0
x_1	3	4	2	1	3
x_2	$\boxed{1}$	$\boxed{1}$	0	0	$\boxed{-1}$
x_3	0	0	0	$\boxed{1}$	0

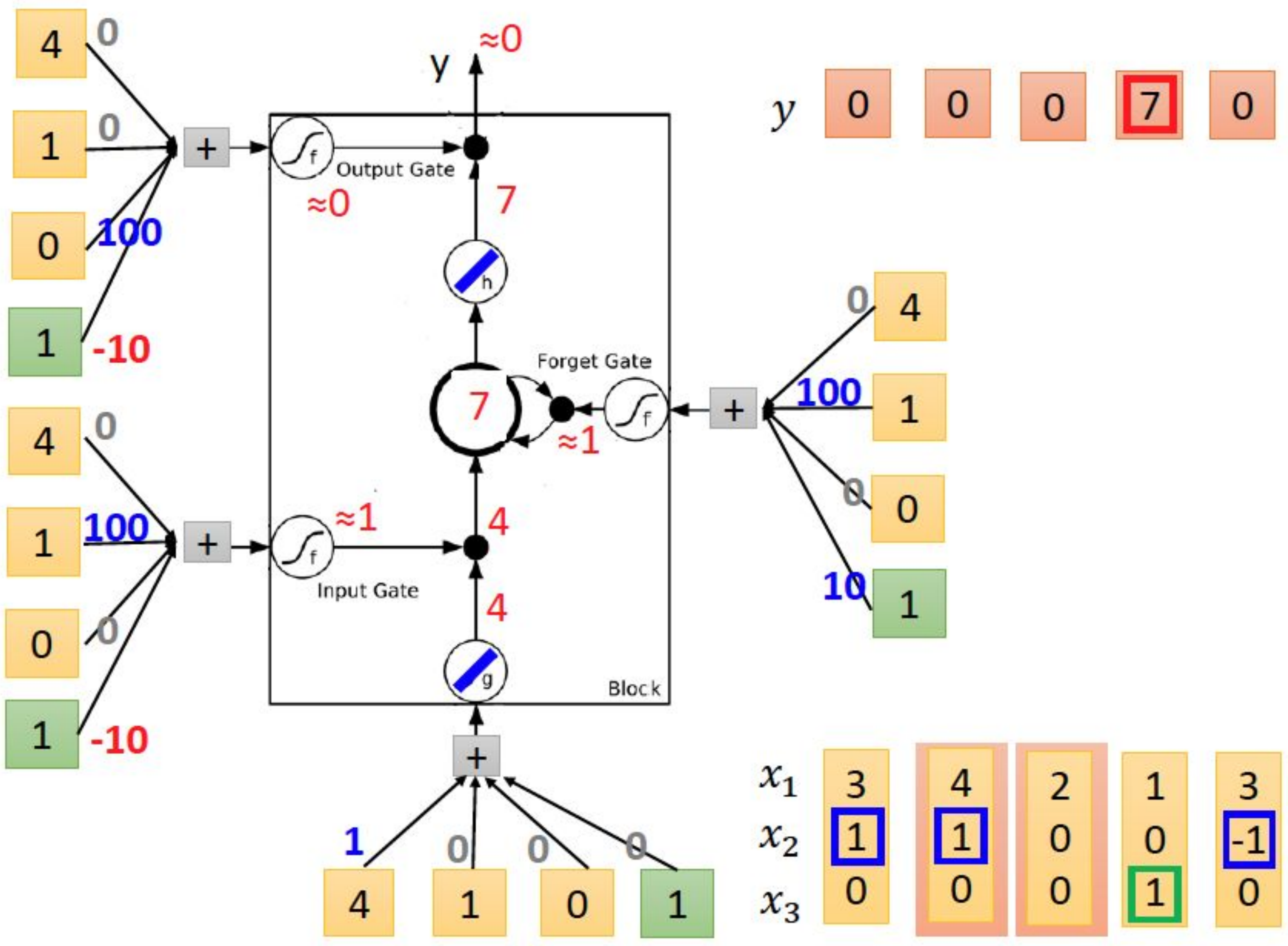
When x_2 = 1, add the numbers of x_1 into the memory

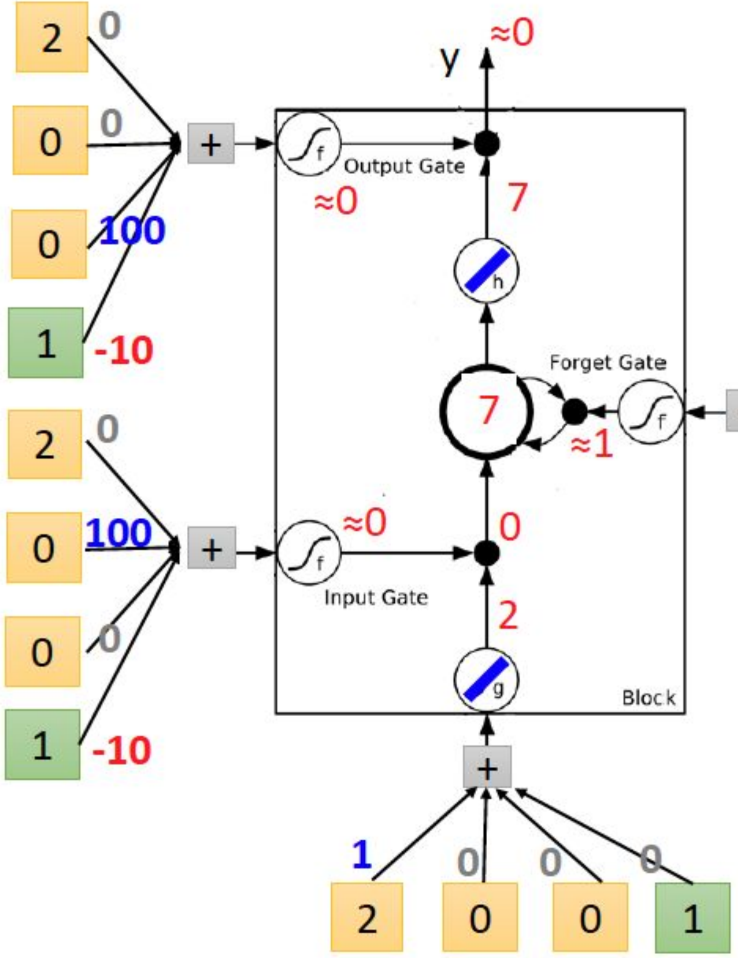
When x_2 = -1, reset the memory

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



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

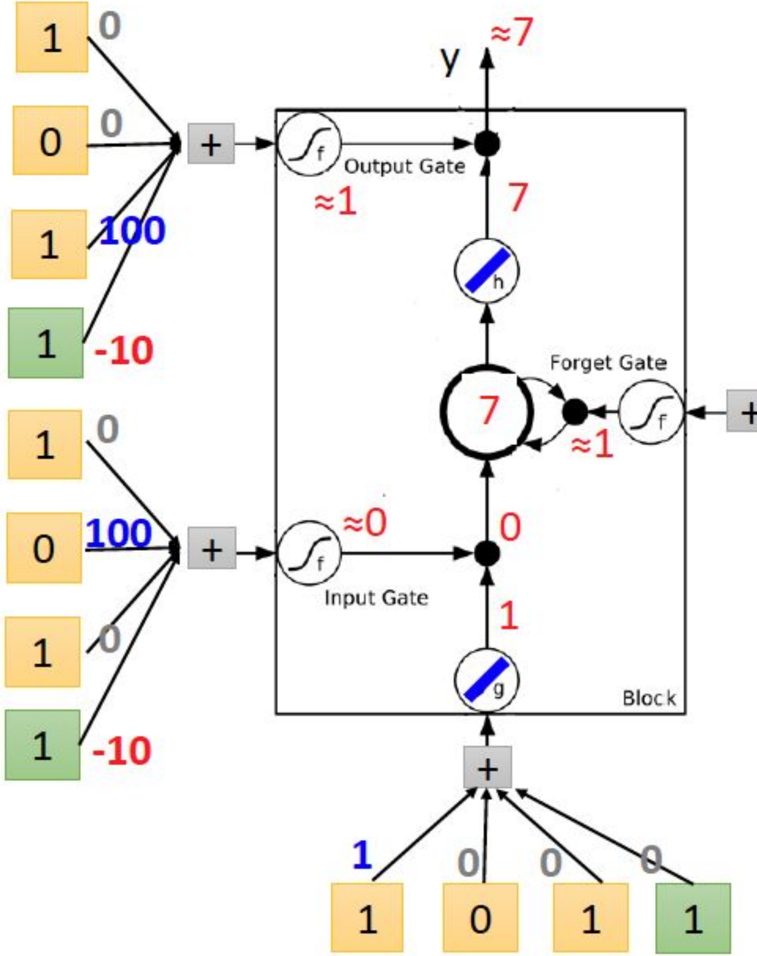


y	0	0	0	7	0
x_1	3	4	2	1	3
x_2	1	1	0	0	-1
x_3	0	0	0	1	0

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.



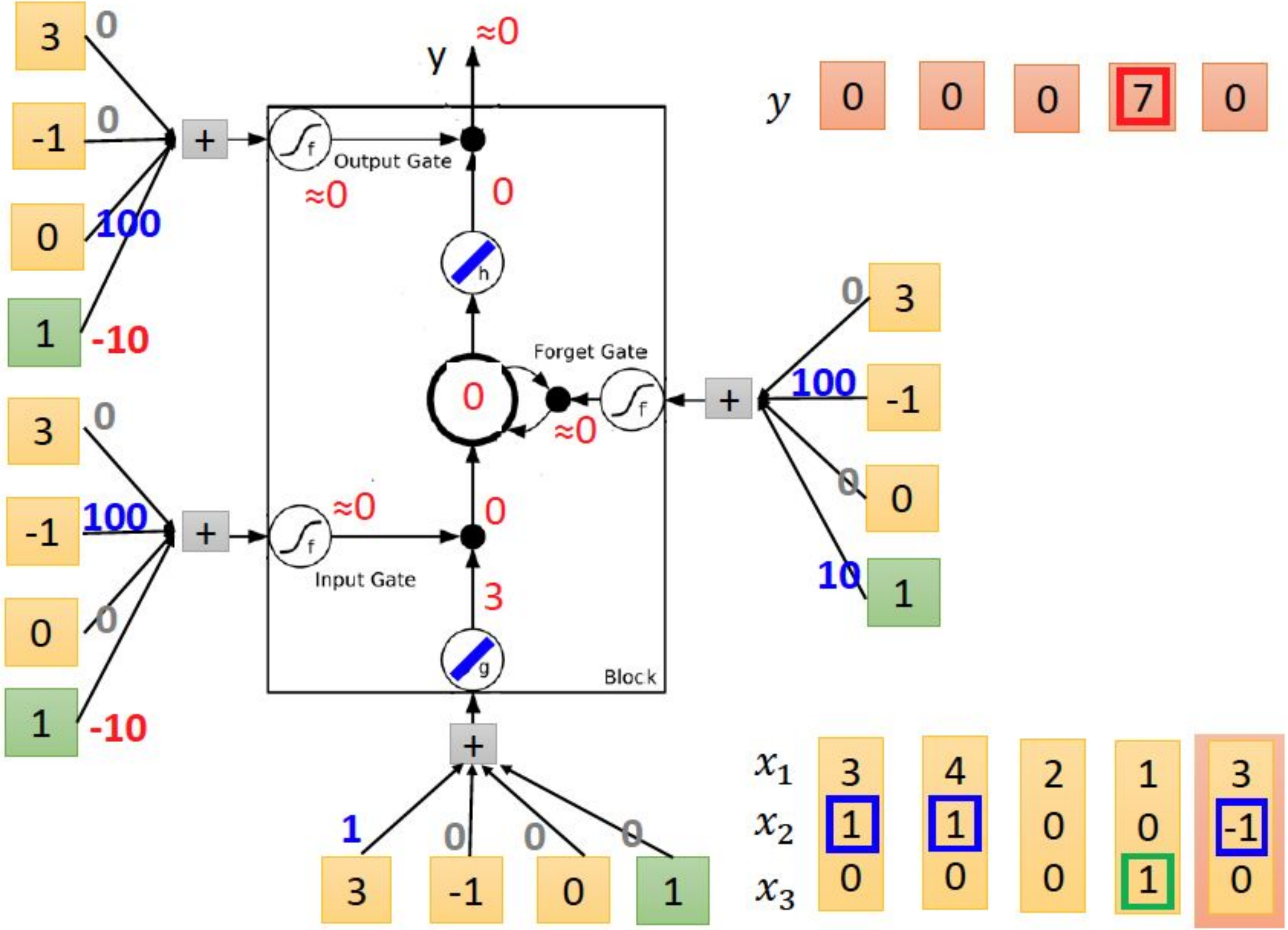
y	0	0	0	7	0

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

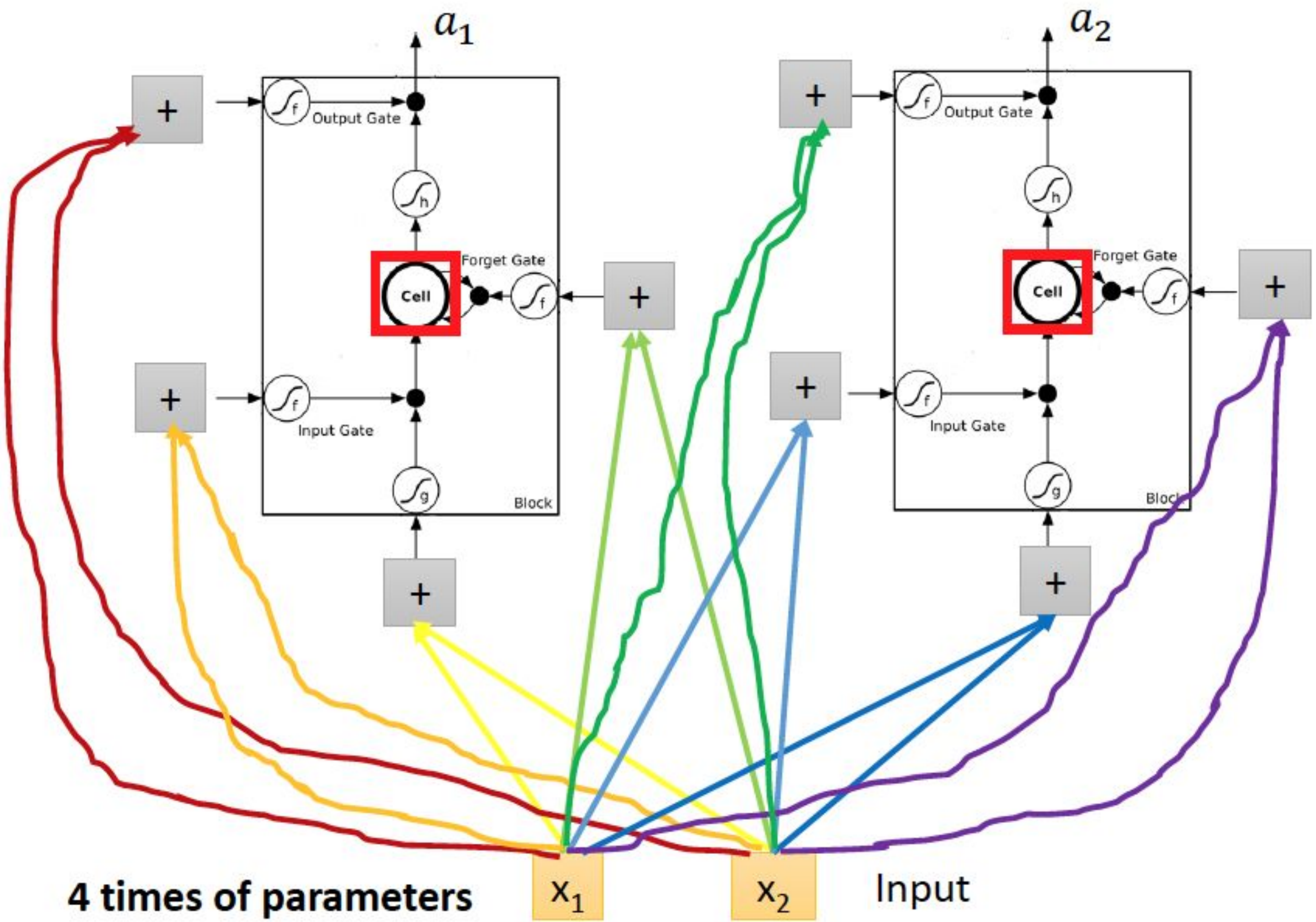
When $x_2 = -1$, reset the memory

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

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
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



4 times of parameters

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

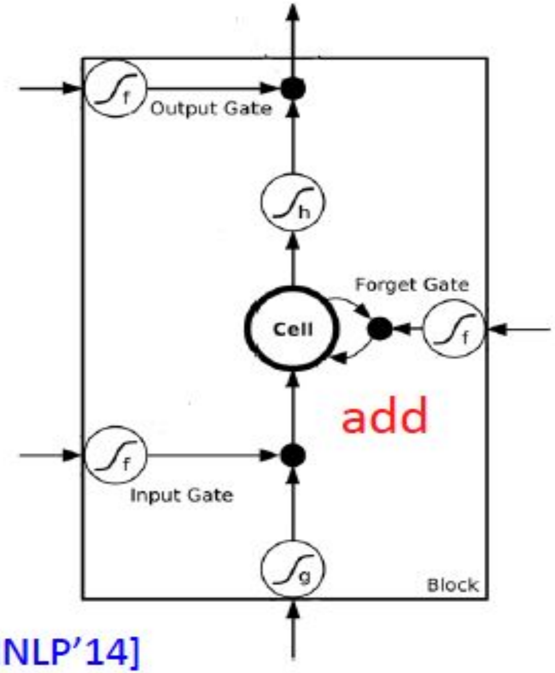
➢ Memory and input are ***added***

➢ The influence never disappears unless forget gate is closed

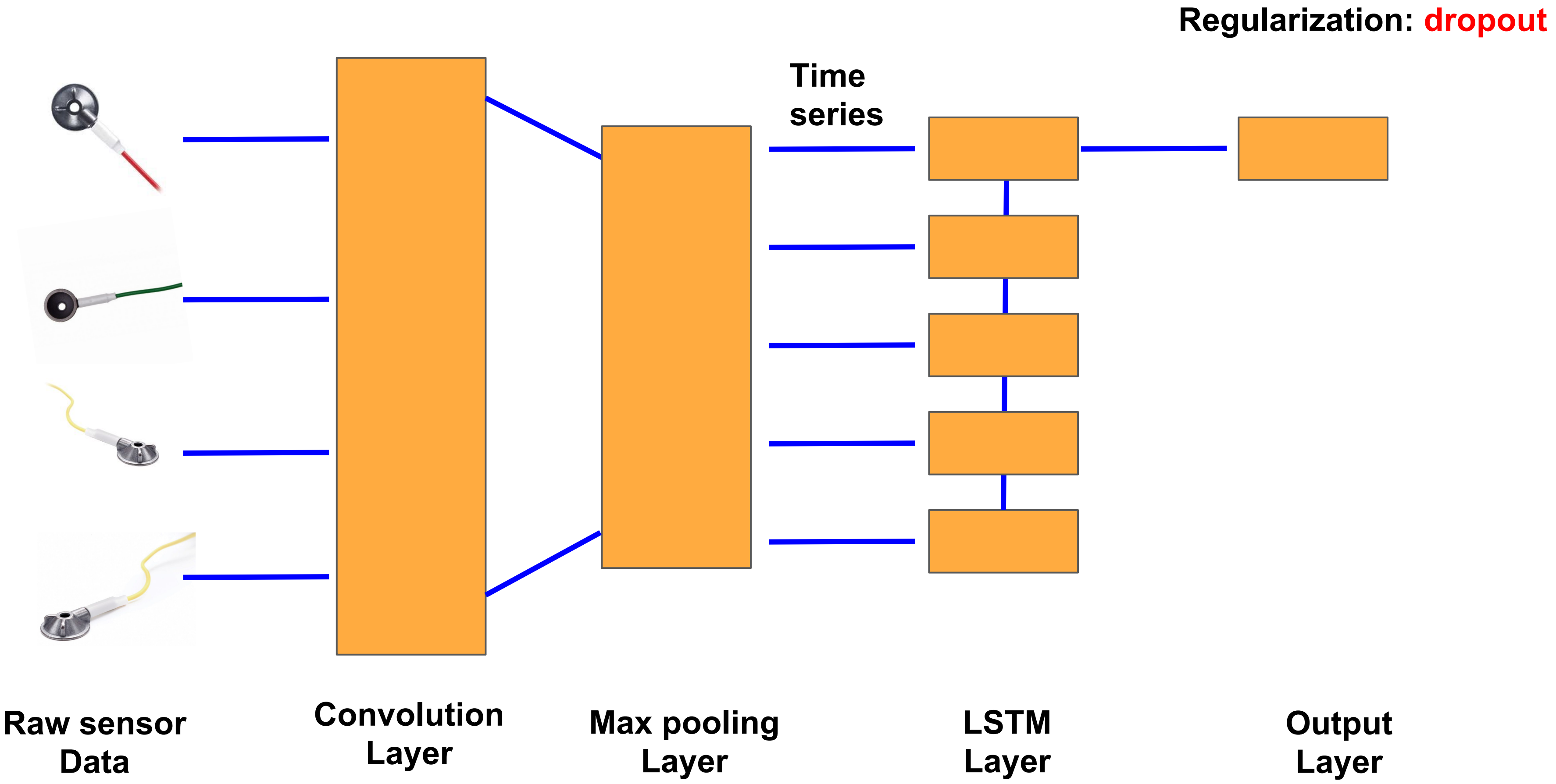
➔ No Gradient vanishing (If forget gate is opened.)

Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]



CNN with LSTM



A few words about attention model

- **Attention involves focus of certain parts of input;**
- **Types of attention: soft/hard attention;**
- **Models:**

  LSTM with soft attention (add extra attention layer);

  Simple attention model.