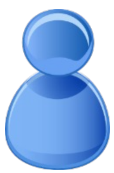
05
Neural Networks
Recurrent Neural Networks (RNNs)

Example Application

- Slot Filling



I would like to arrive Changsha on November 2nd.

ticket booking system

Slot

Destination:	Changsha
time of arrival:	November 2nd

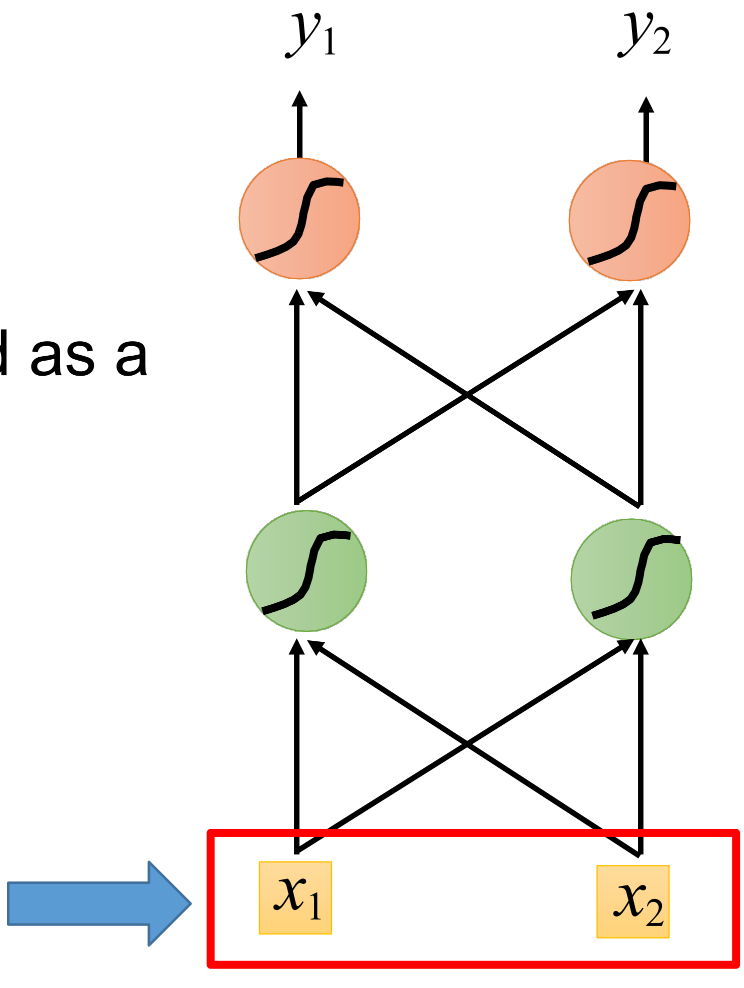
Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Changsha



1-of-N encoding

How to represent each word as a vector?

<u>1-of-N Encoding</u> 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]

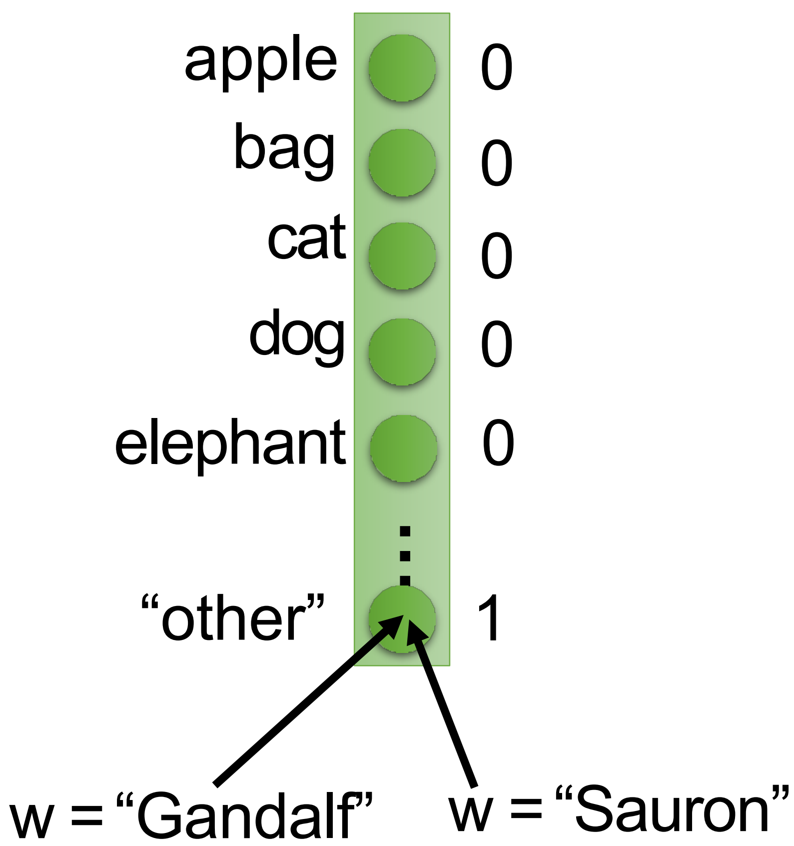
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

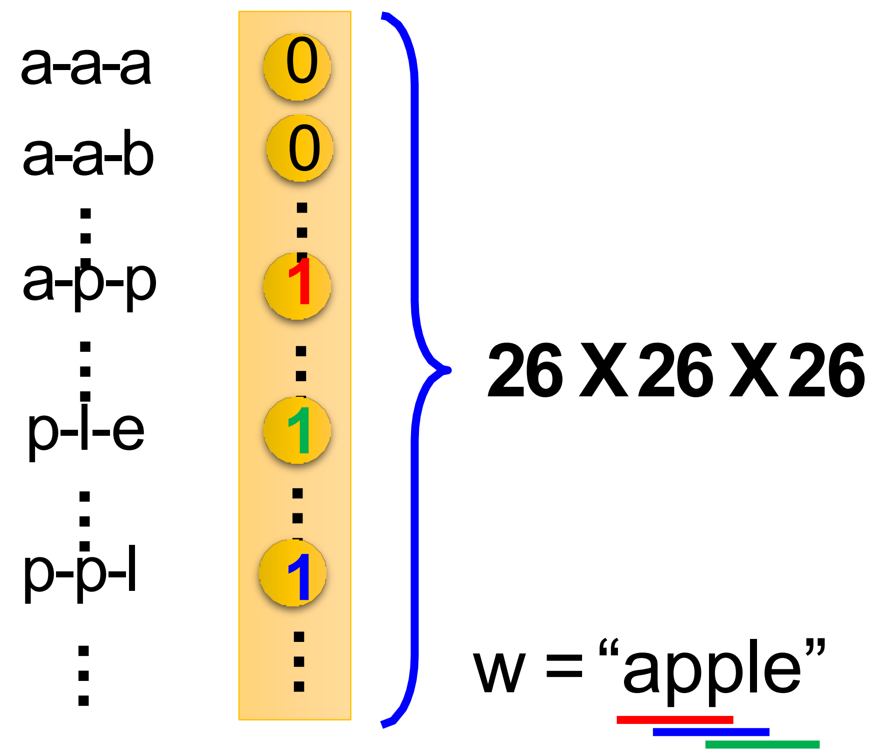
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

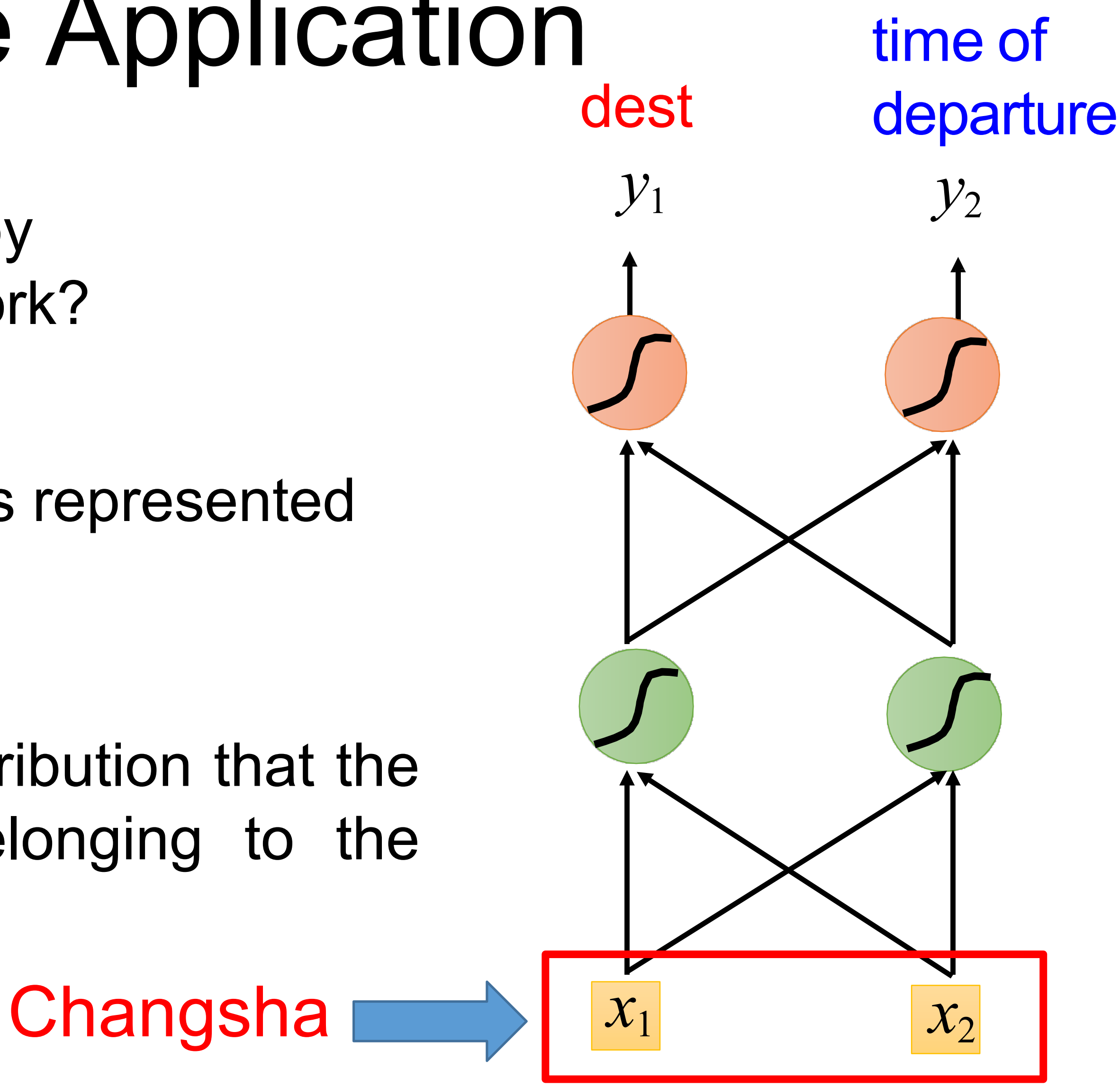
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



Example Application

arrive Changsha on November 2nd

other dest other time time

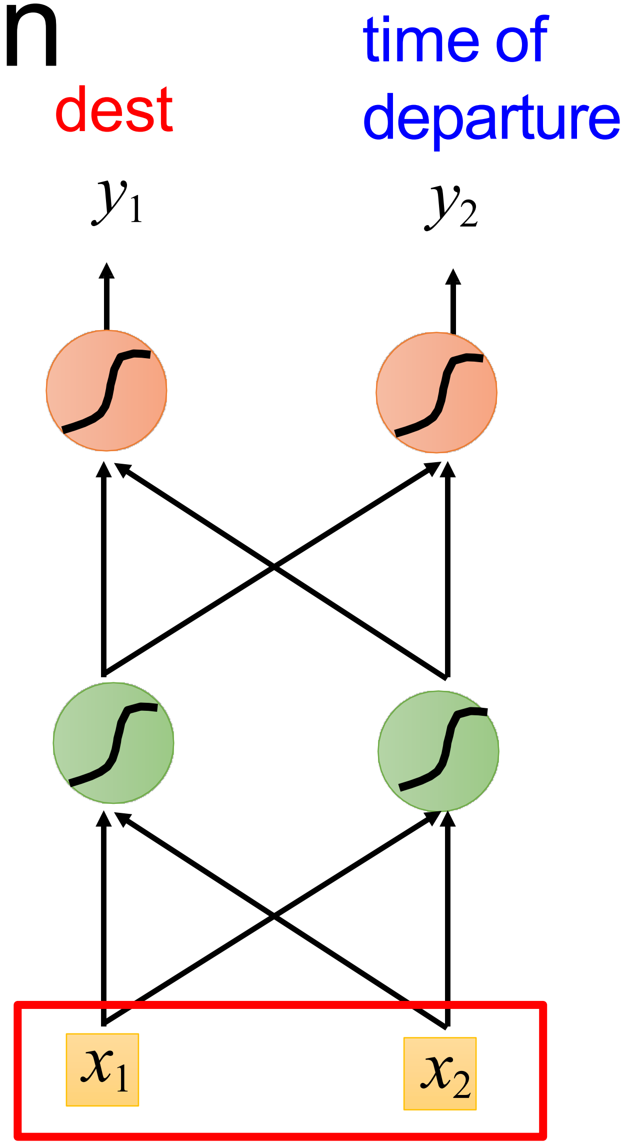
Problem?

leave Changsha on November 2nd

place of departure

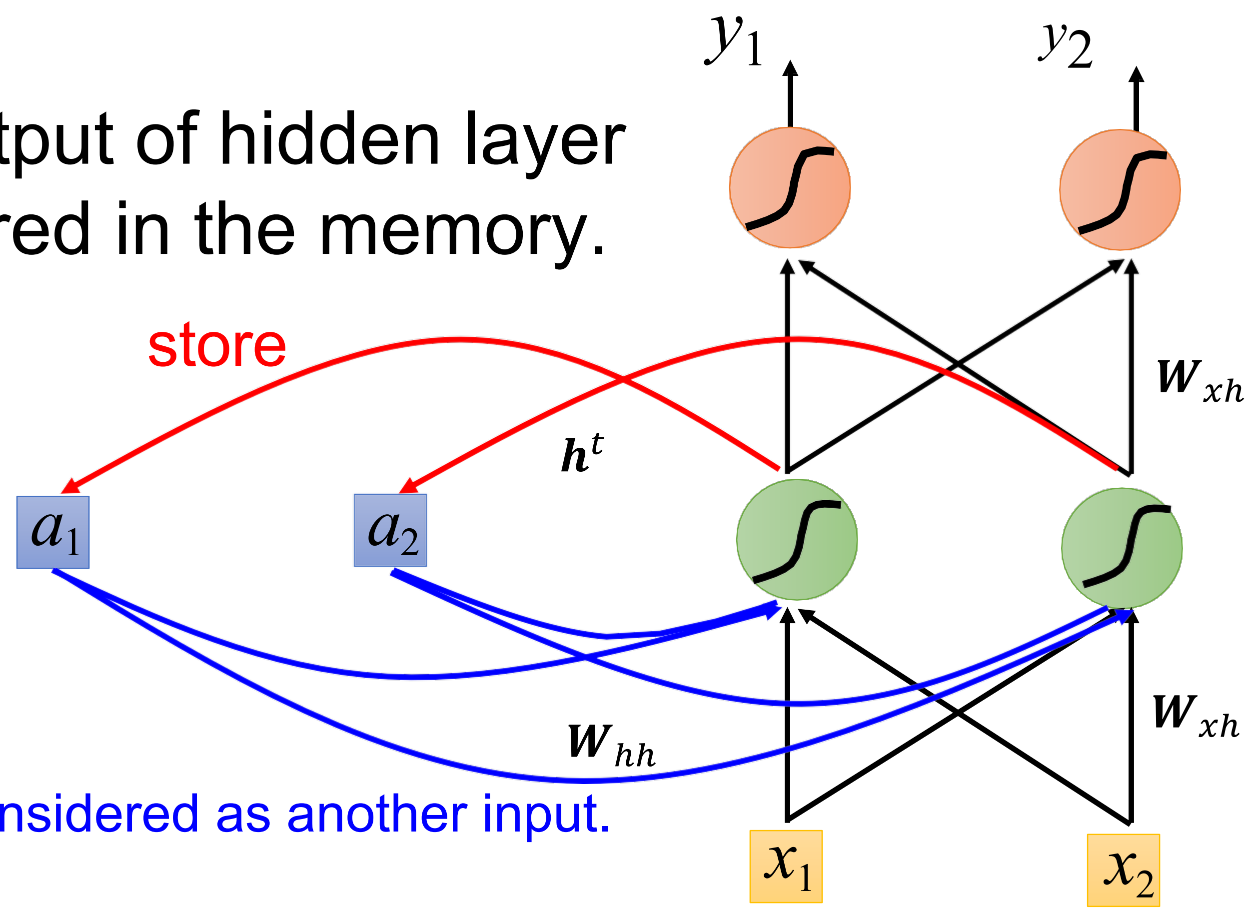
Neural network needs memory!

Changsha



Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



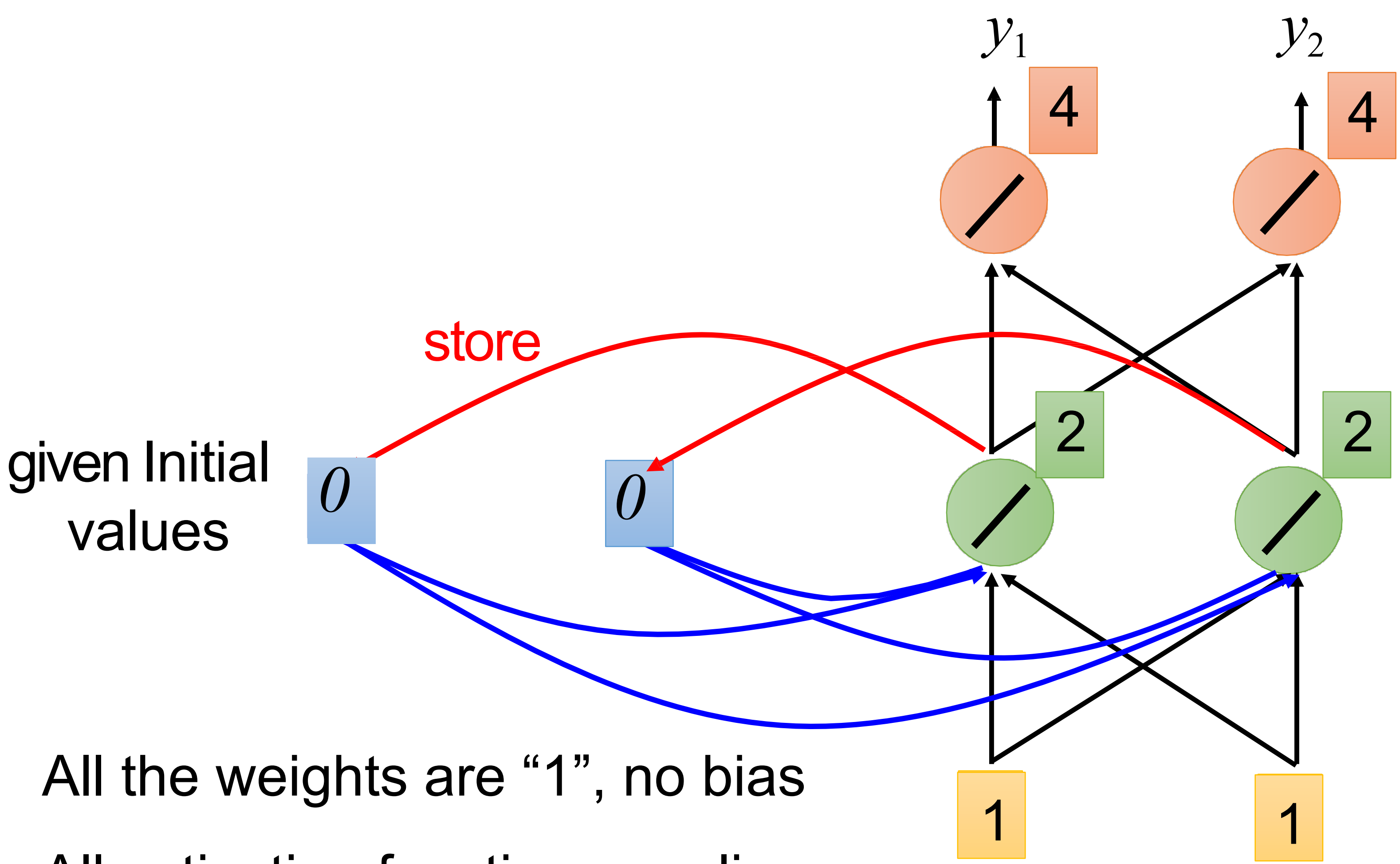
Memory can be considered as another input.

$$\begin{cases} \boldsymbol{h}^t = f(\boldsymbol{W}_{xh}\boldsymbol{x}^t + \boldsymbol{W}_{hh}\boldsymbol{h}^{t-1} + \boldsymbol{b}_h) \\ \boldsymbol{y}^t = f(\boldsymbol{W}_{hy}\boldsymbol{h}^t + \boldsymbol{b}_y) \end{cases}$$

Example

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

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



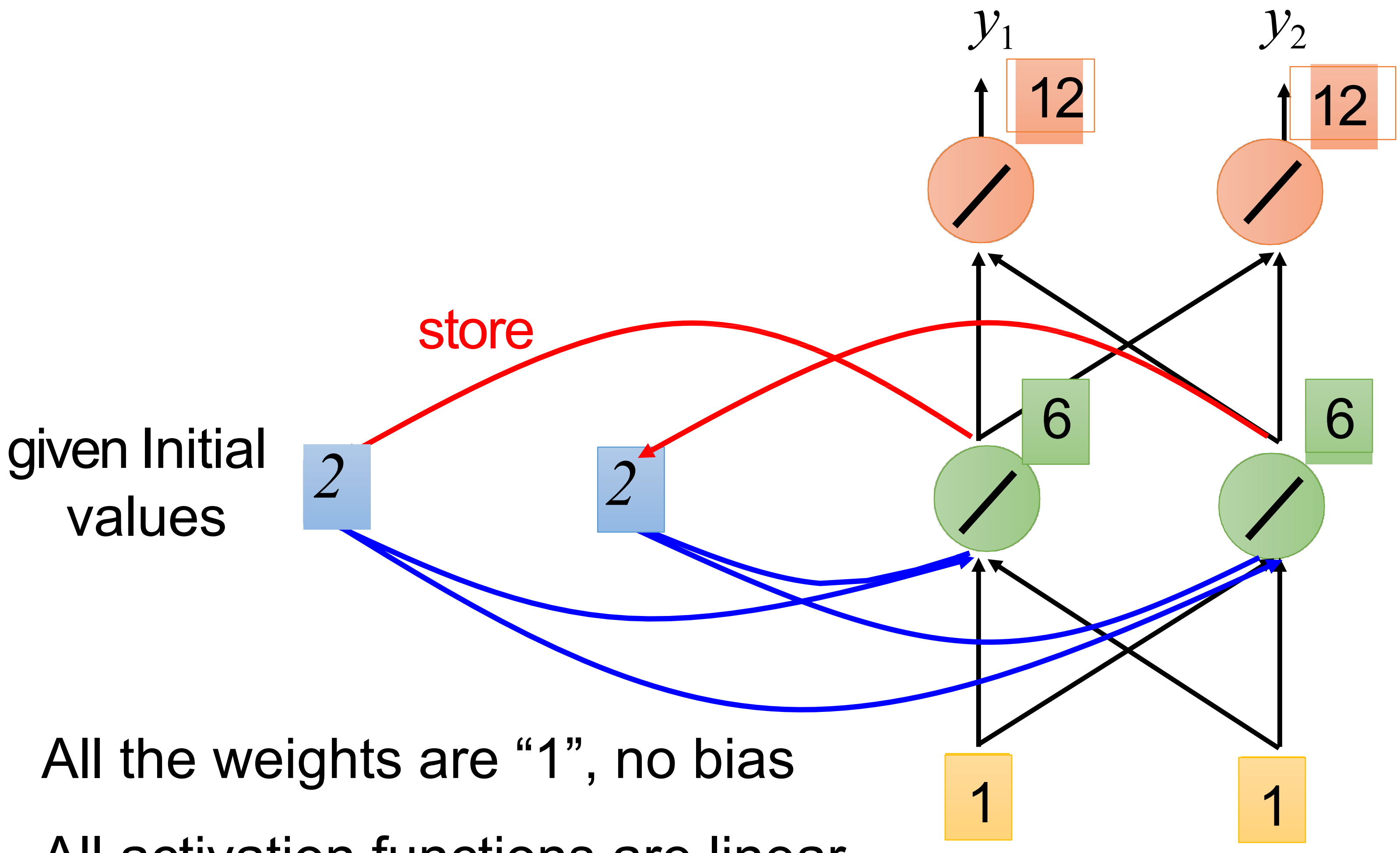
All the weights are "1", no bias

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

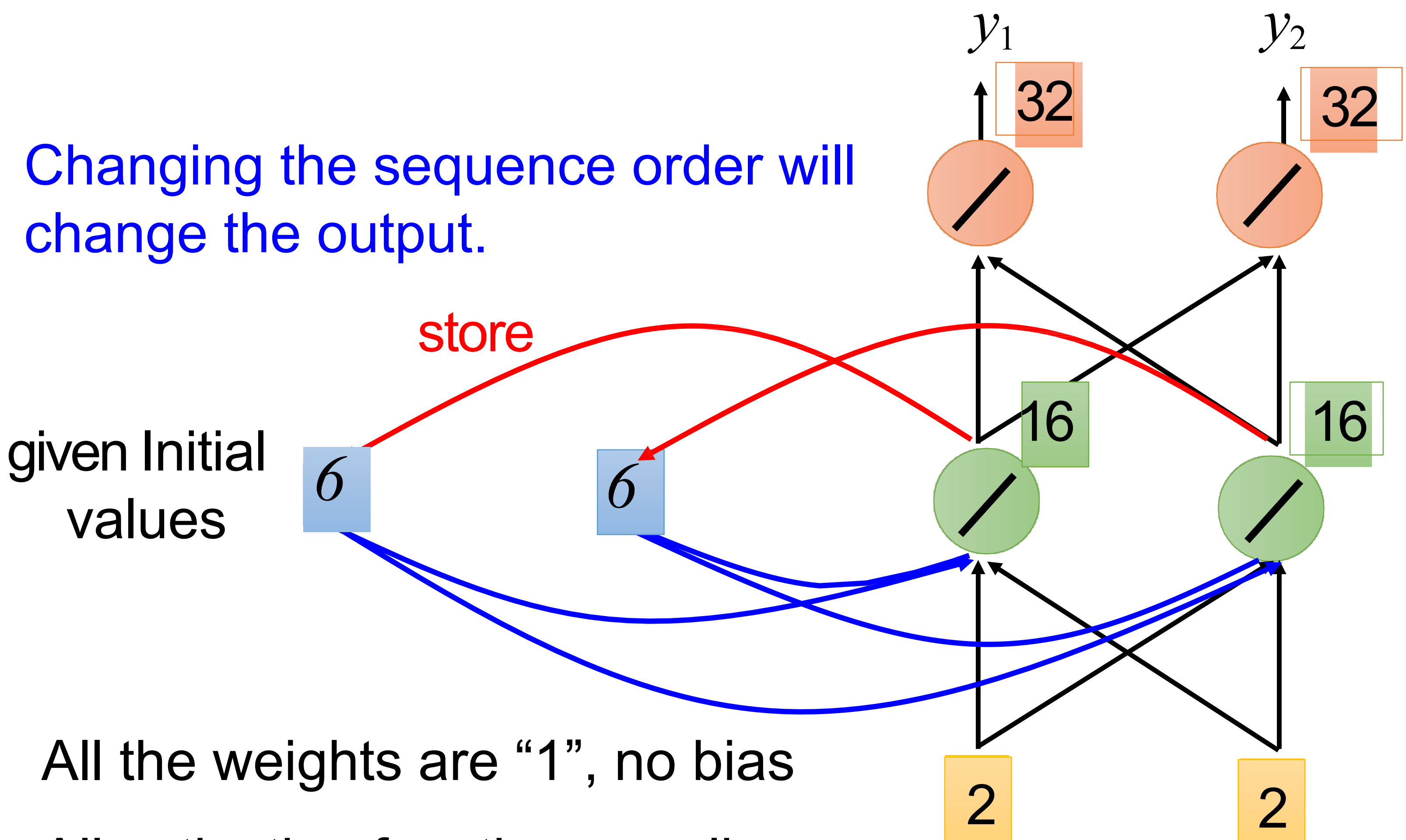
All activation functions are linear

Example

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

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.

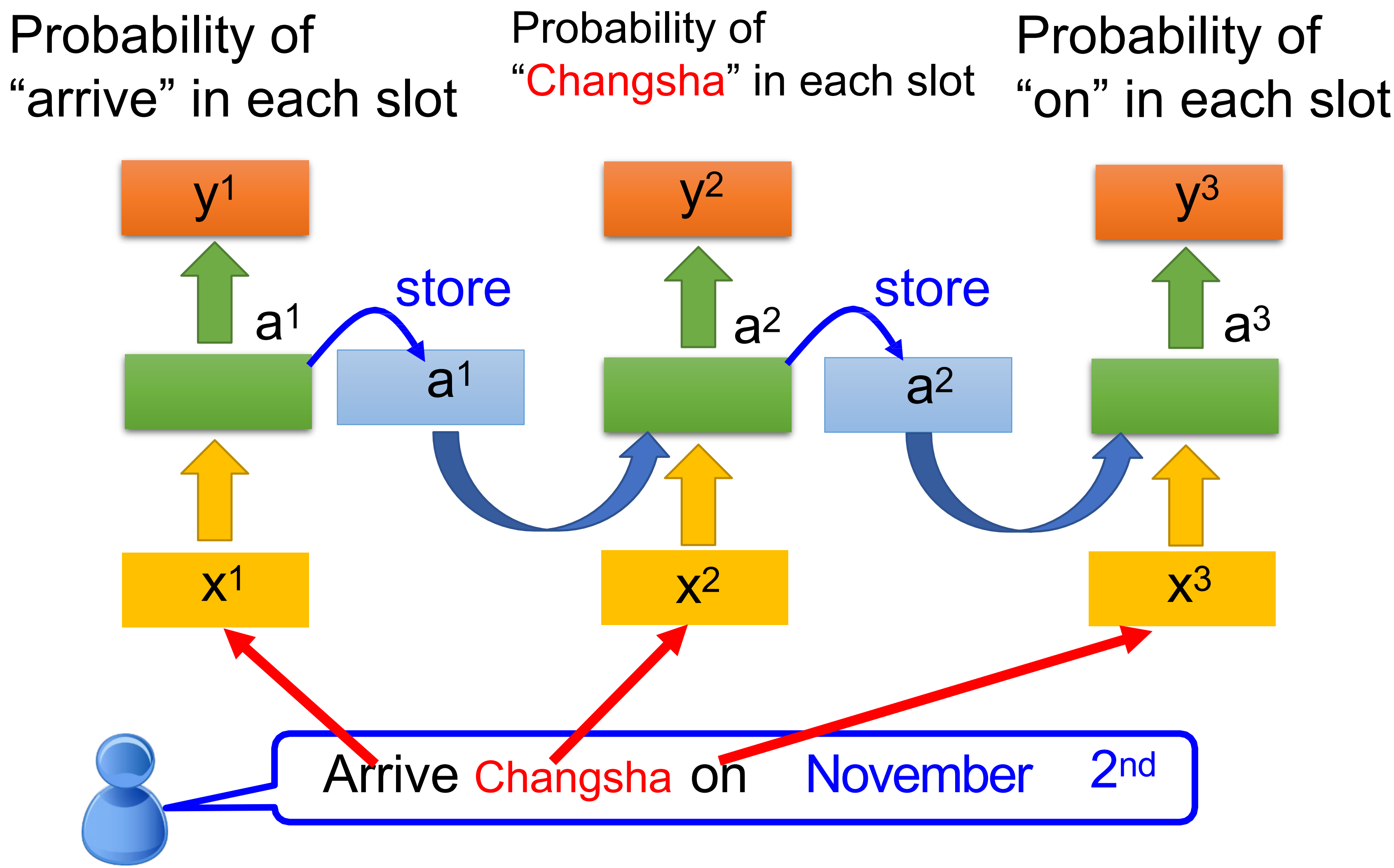


All the weights are "1", no bias

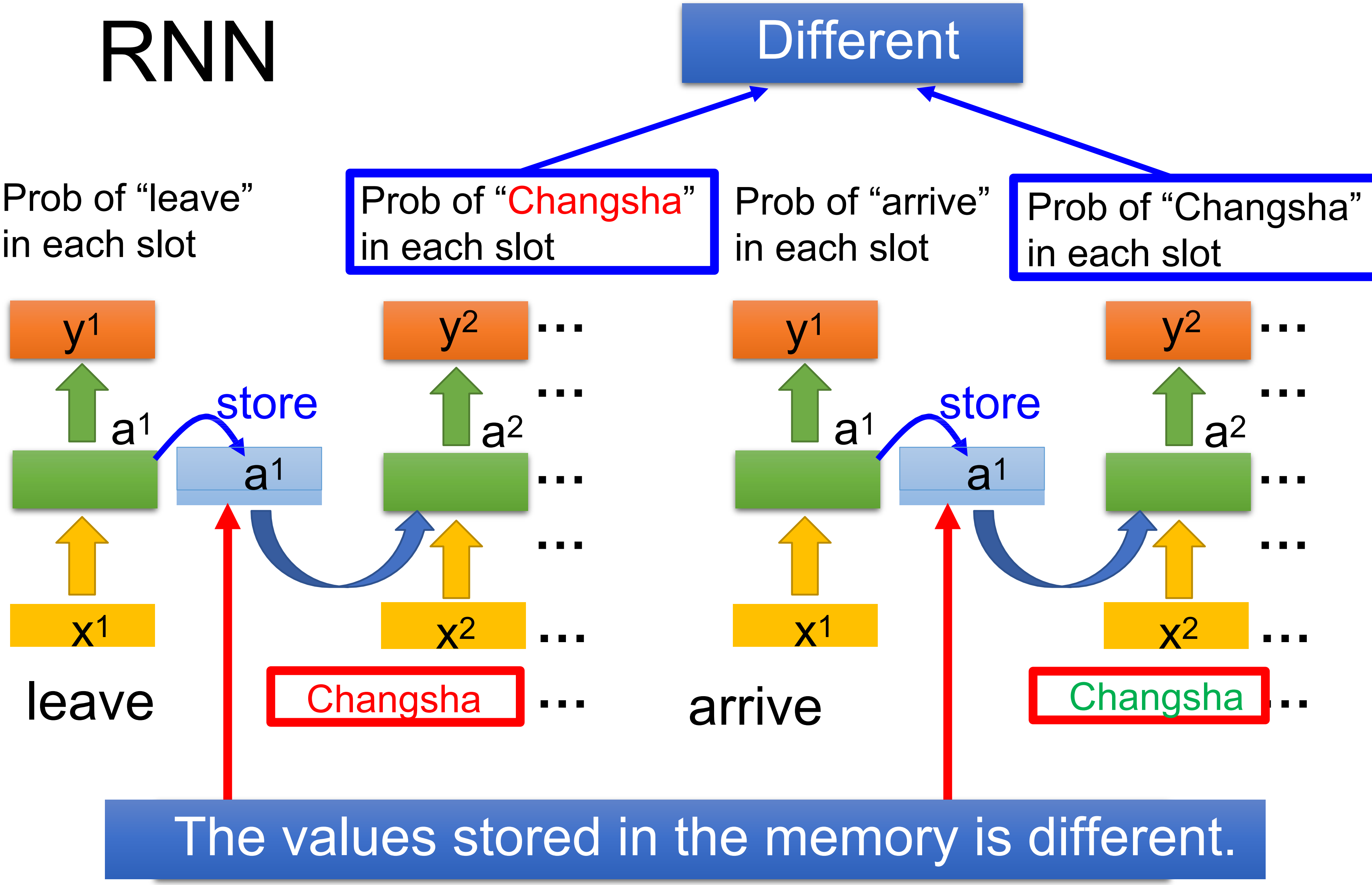
All activation functions are linear

RNN

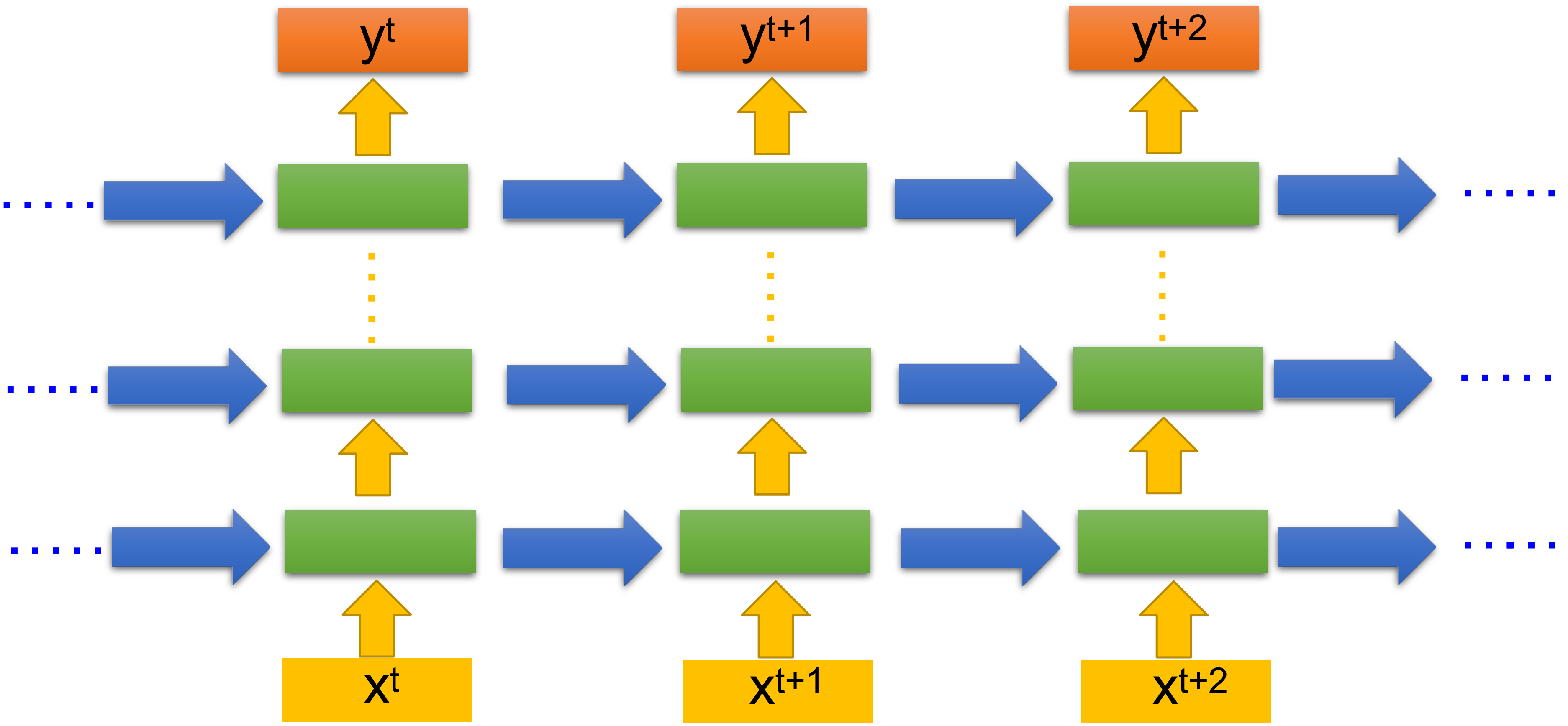
The same network is used again and again.



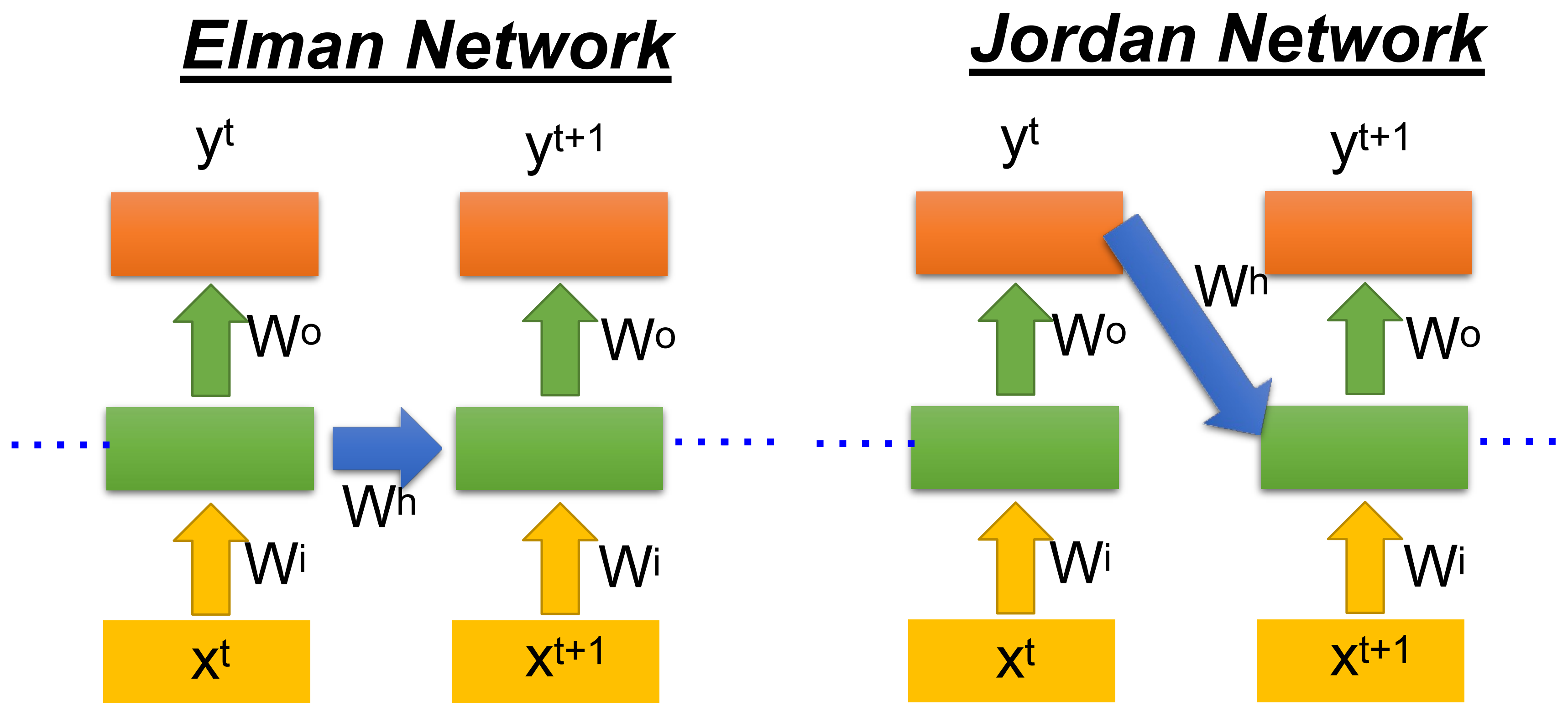
RNN


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

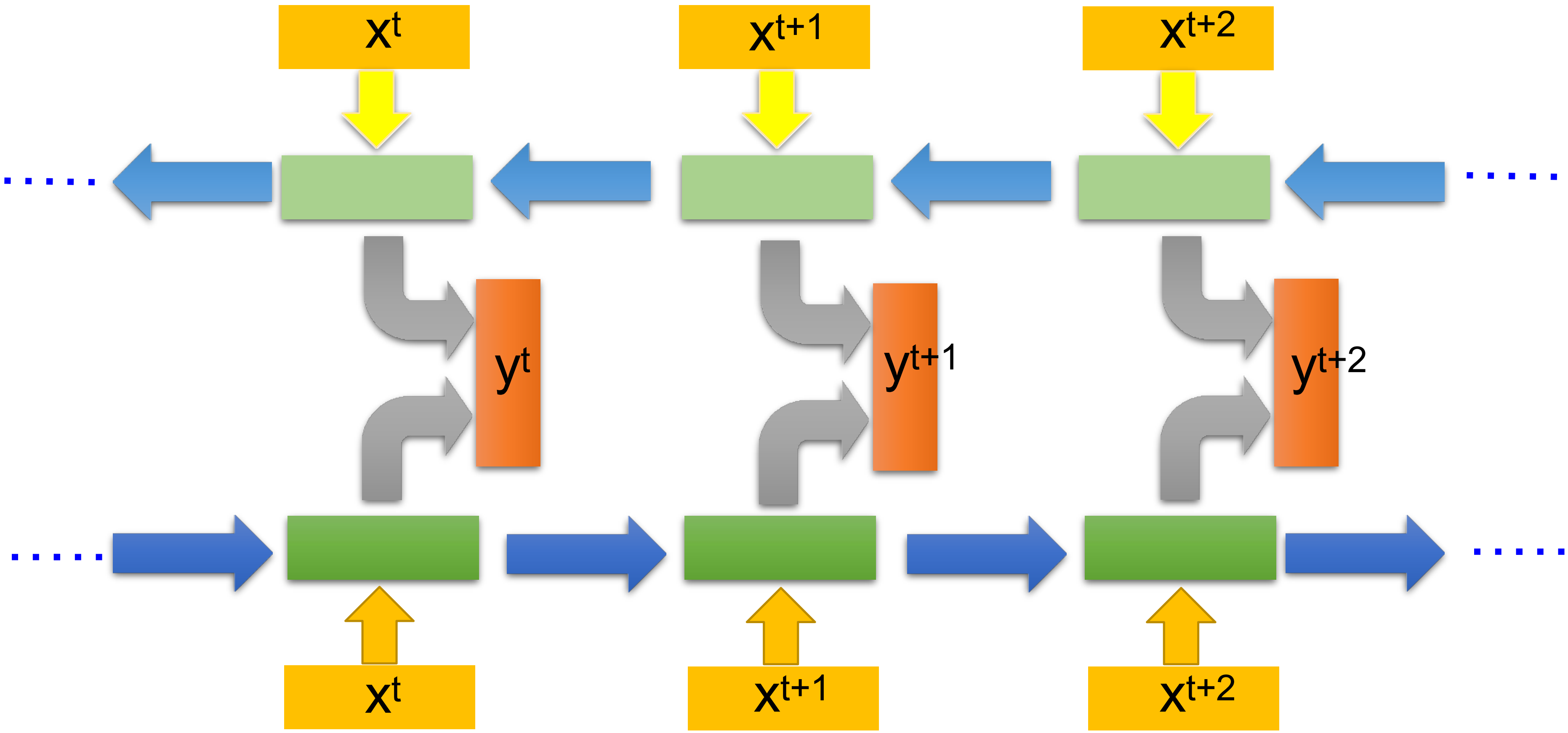
Of course it can be deep …



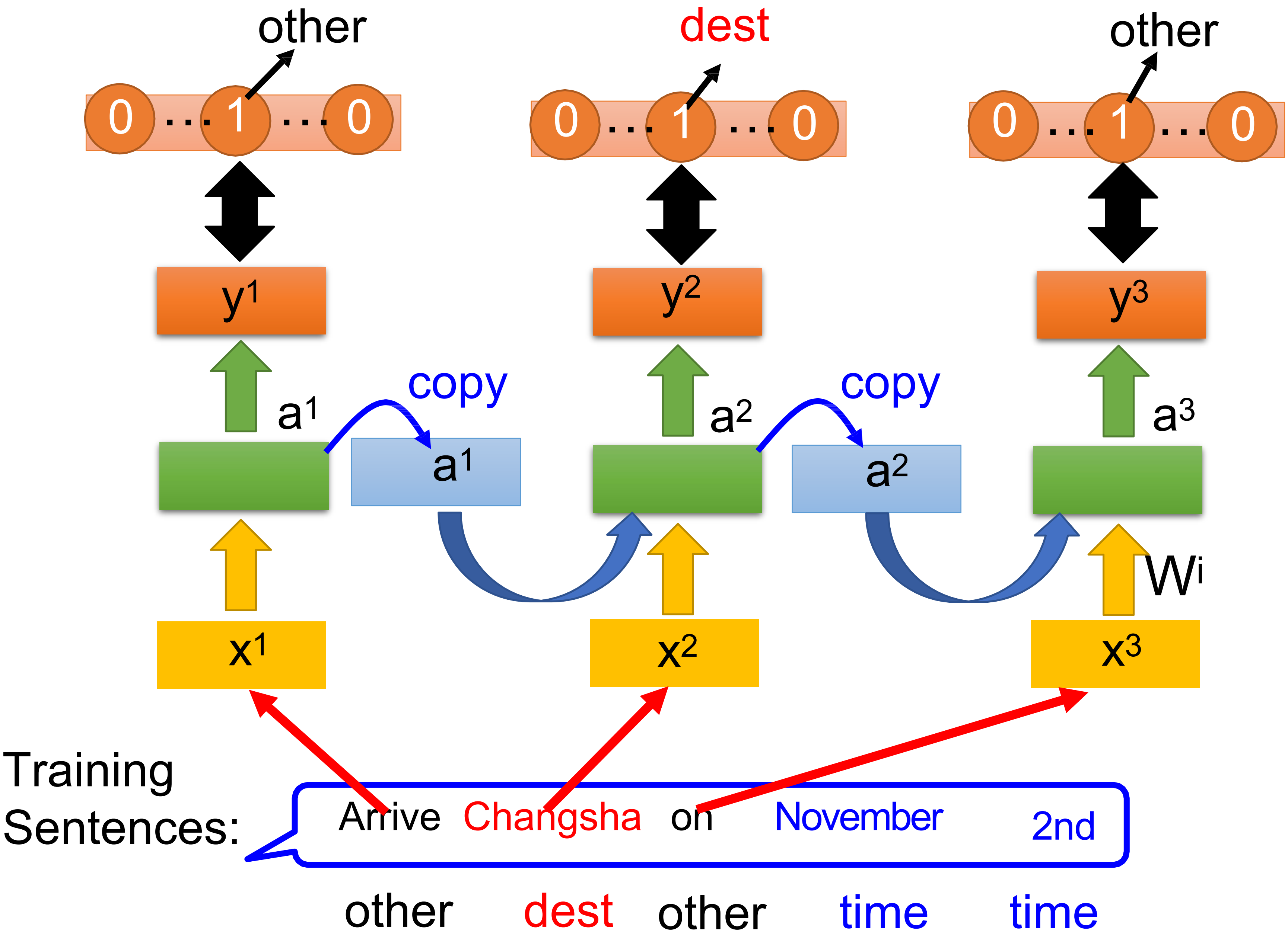
Elman Network & Jordan Network



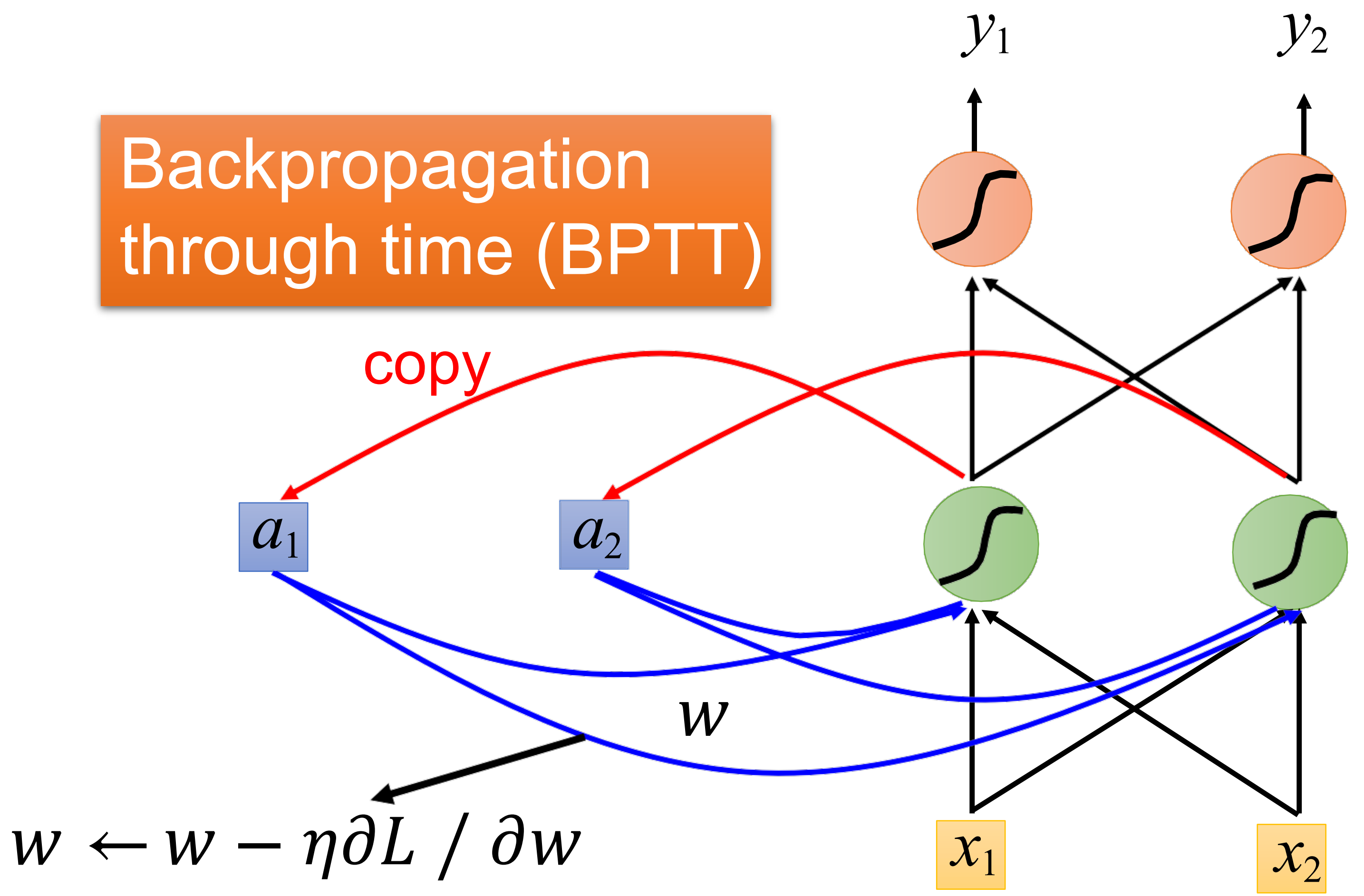
Bidirectional RNN



Learning Target

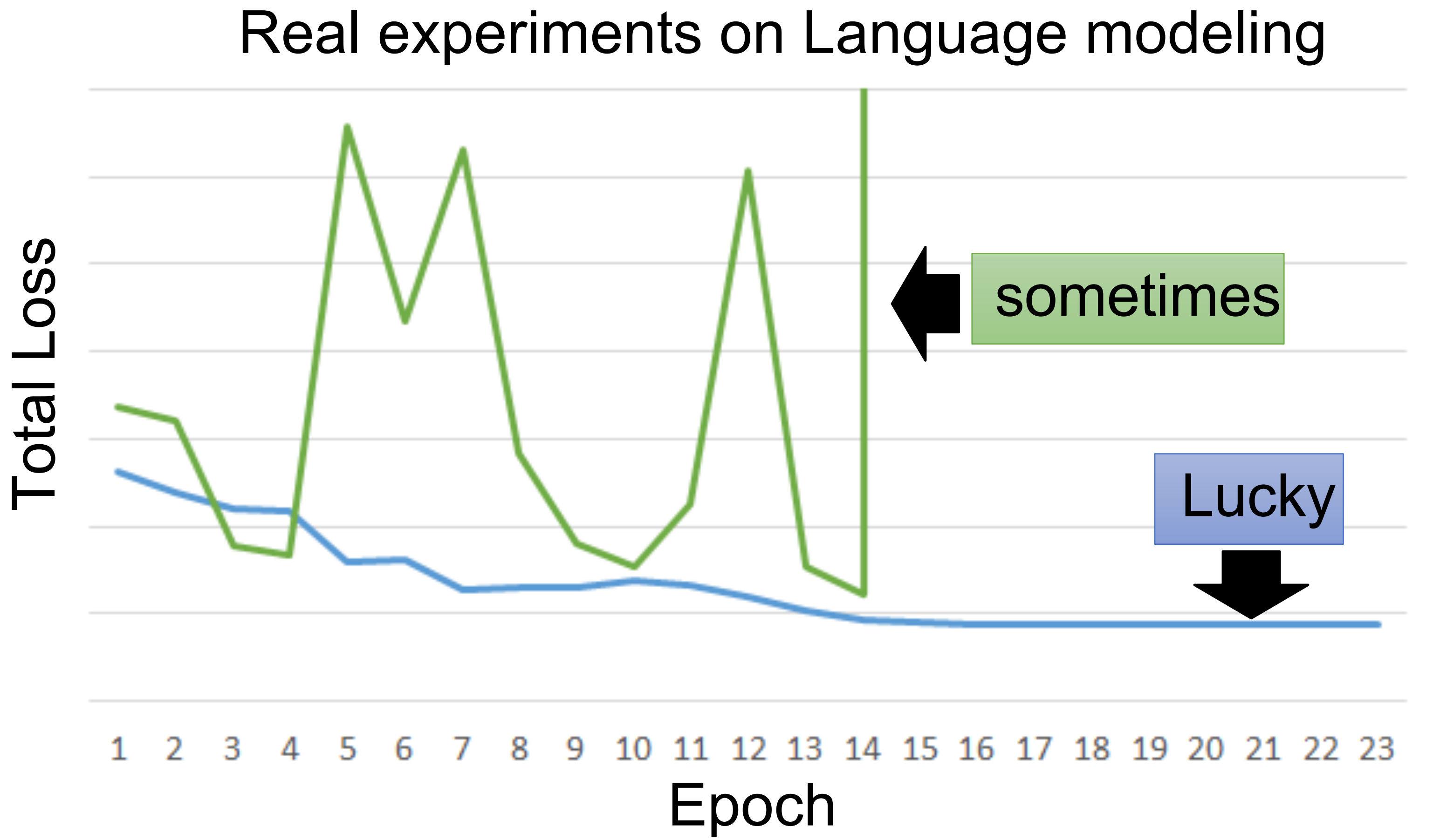


Learning

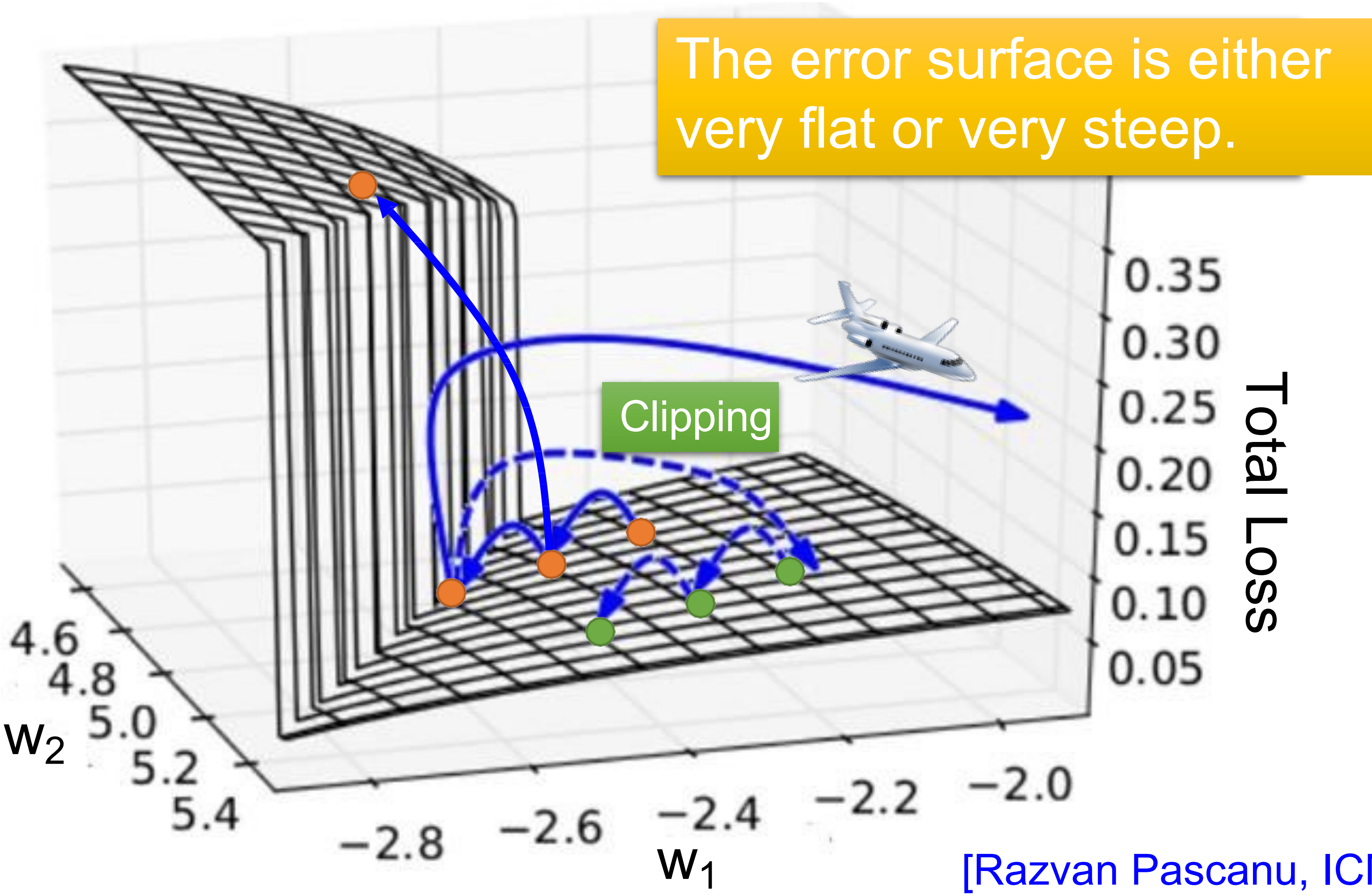

Backpropagation through time (BPTT)
copy
y_1
y_2
a_1
a_2
w
x_1
x_2
$w \leftarrow w - \eta \partial L / \partial w$

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?

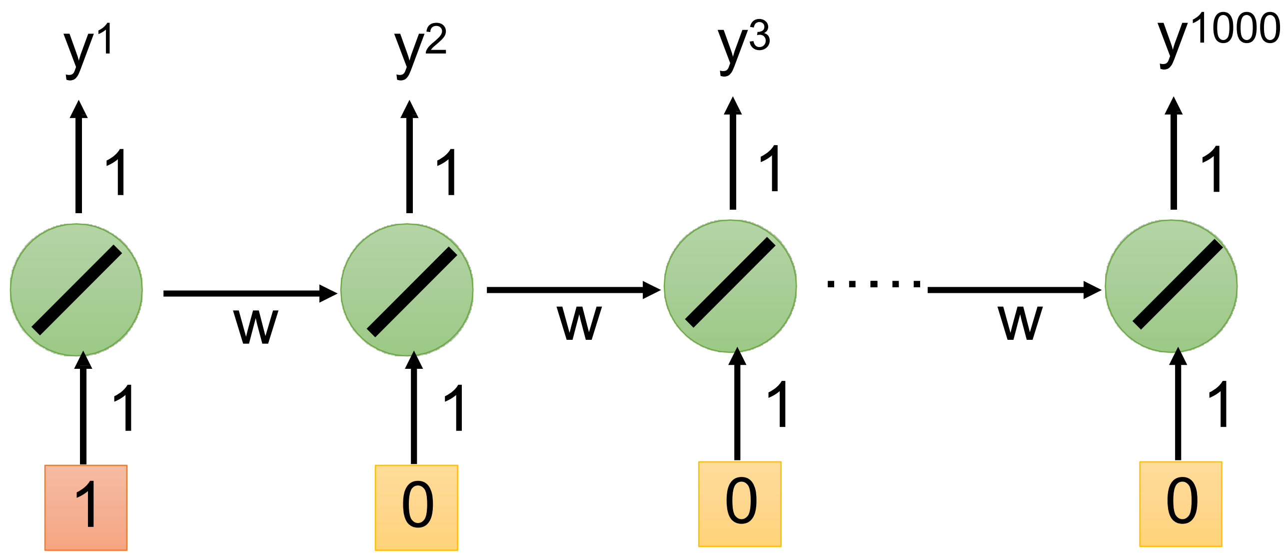
$w = 0.99 \Rightarrow y^{1000} \approx 0$

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

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

$=w^{999}$

Toy Example



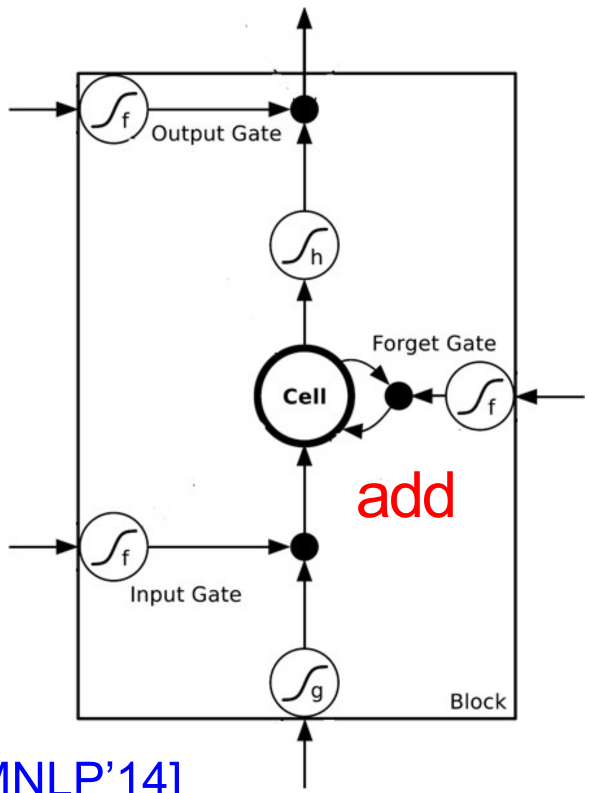
Helpful Techniques

- 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

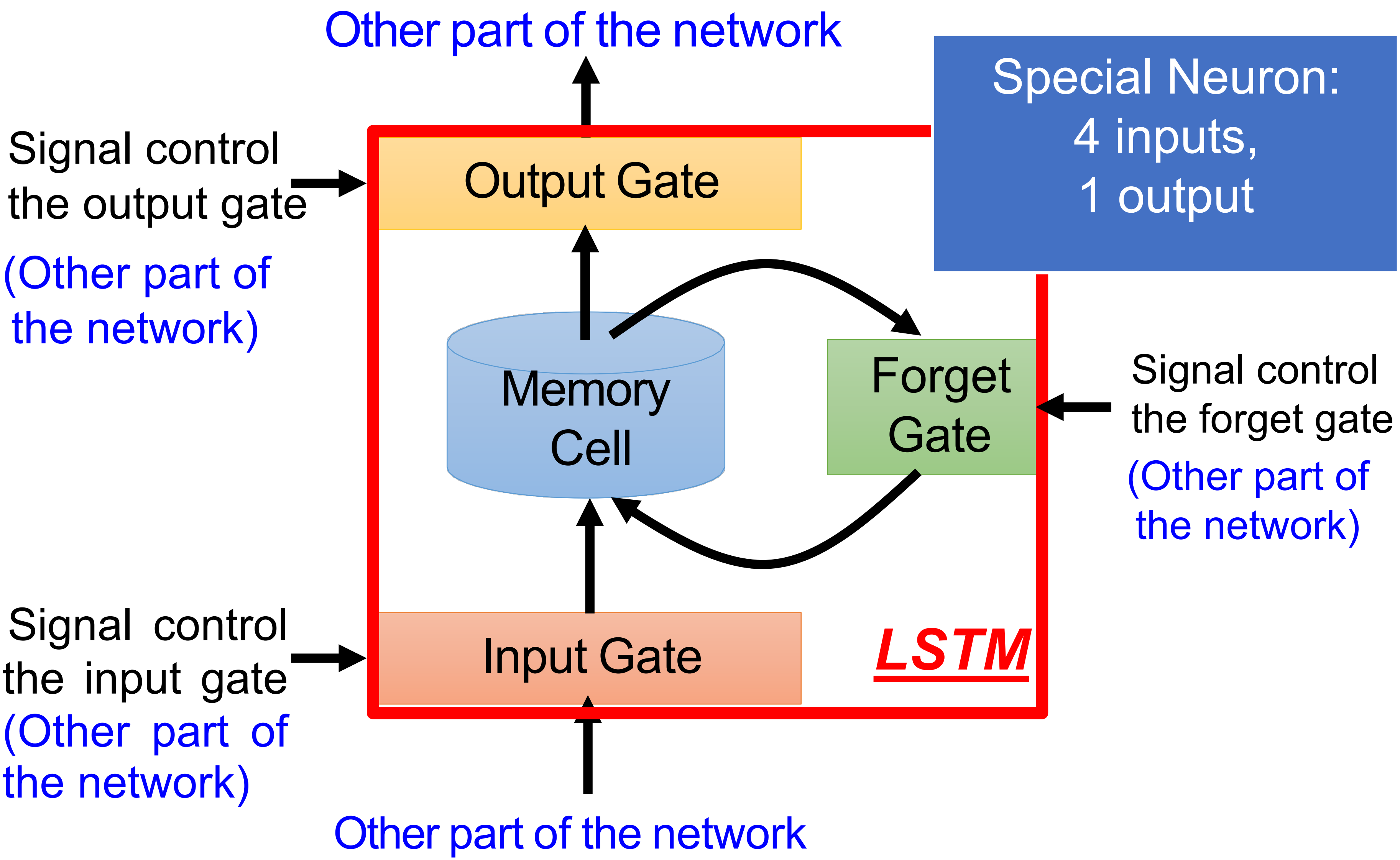
➡ No Gradient vanishing (If forget gate is opened.)

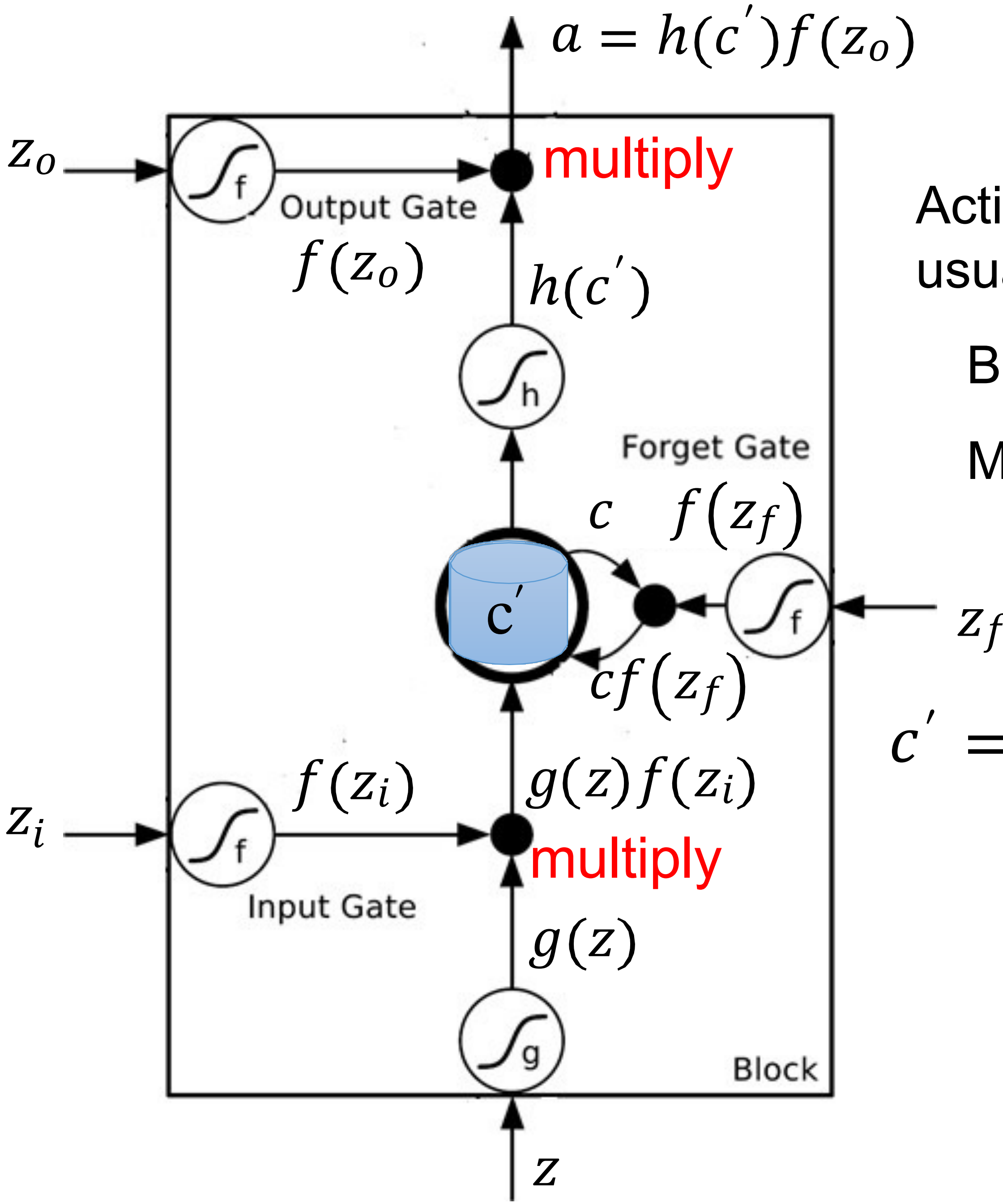


Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

Long Short-Term Memory (LSTM)





Activation function f is usually a sigmoid function

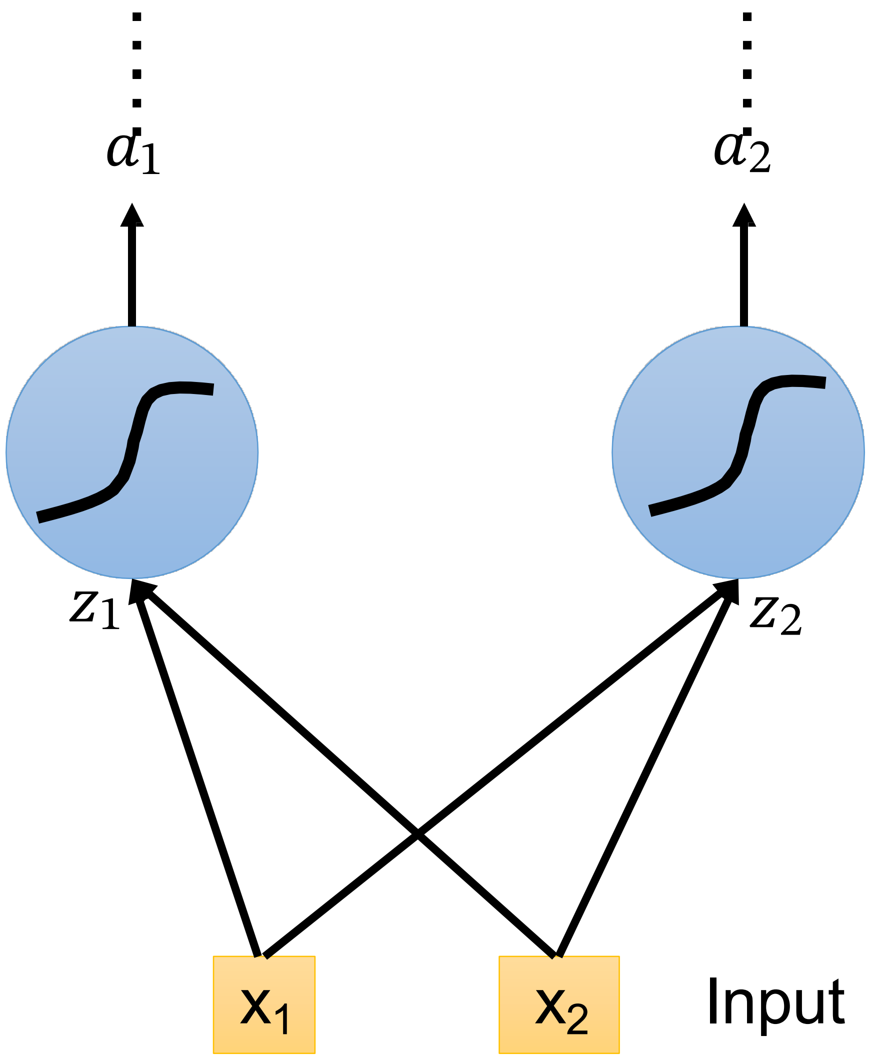
Between 0 and 1

Mimic open and close gate

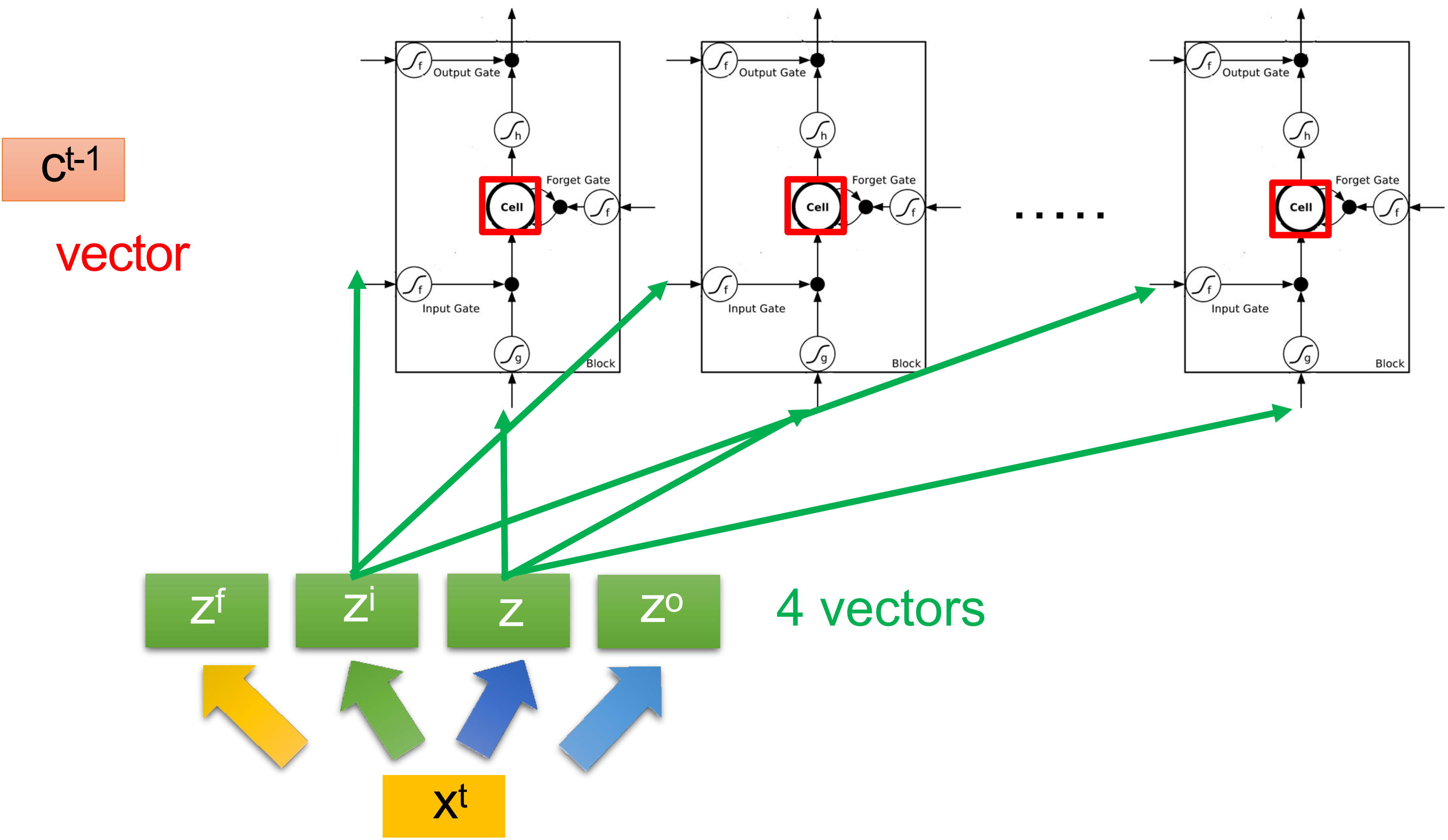
$$c' = g(z)f(z_i) + cf(z_f)$$

Original Network:

➢ Simply replace the neurons with LSTM

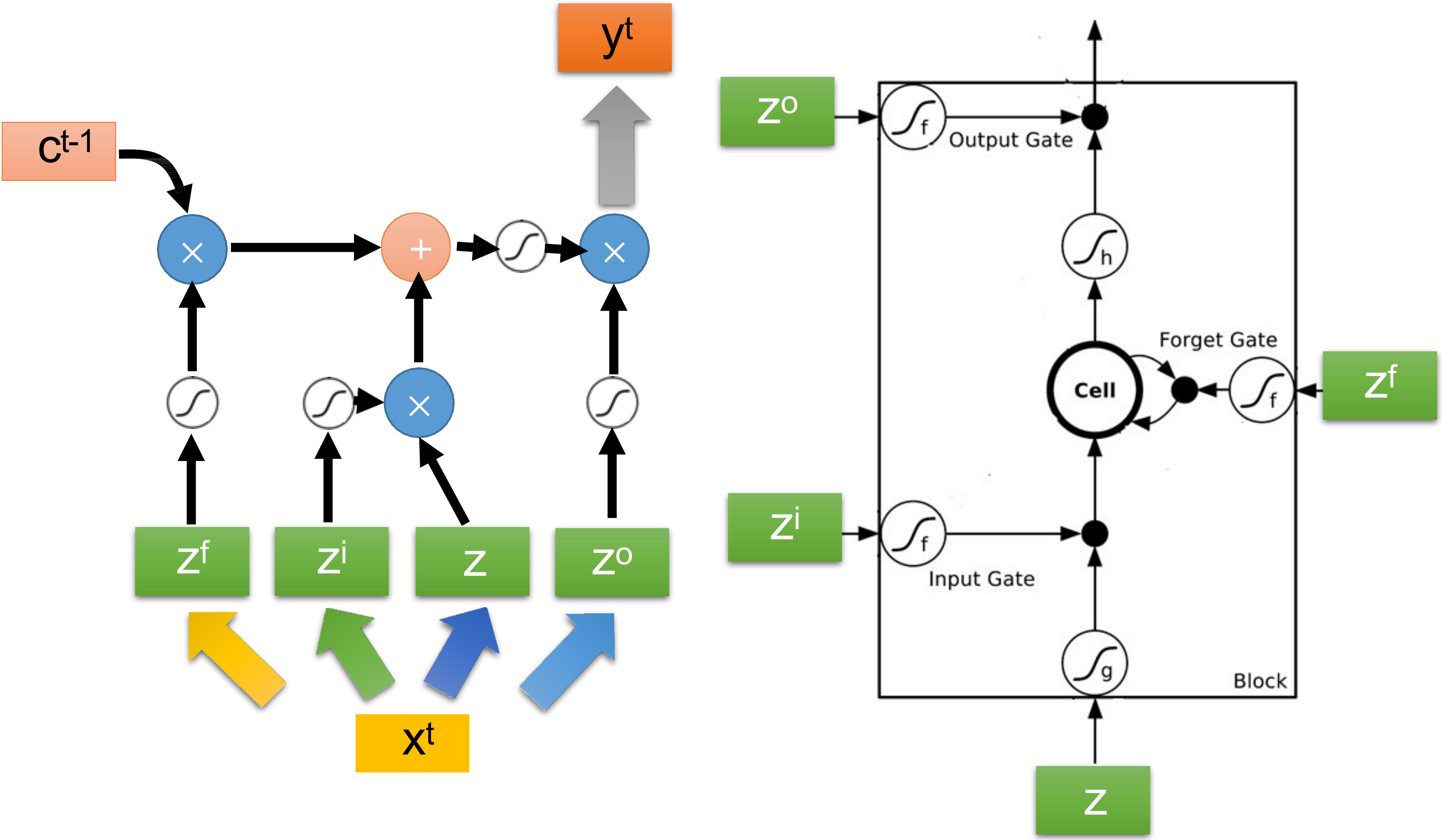


LSTM



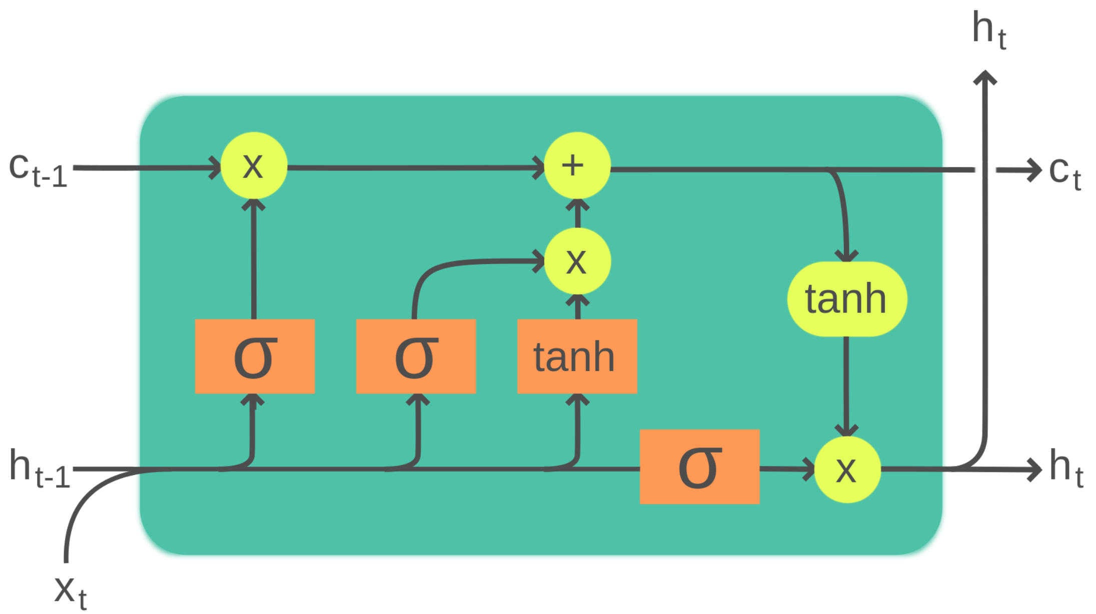
c^{t-1}
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
z^f
z^i
z
z^o
4 vectors
x^t

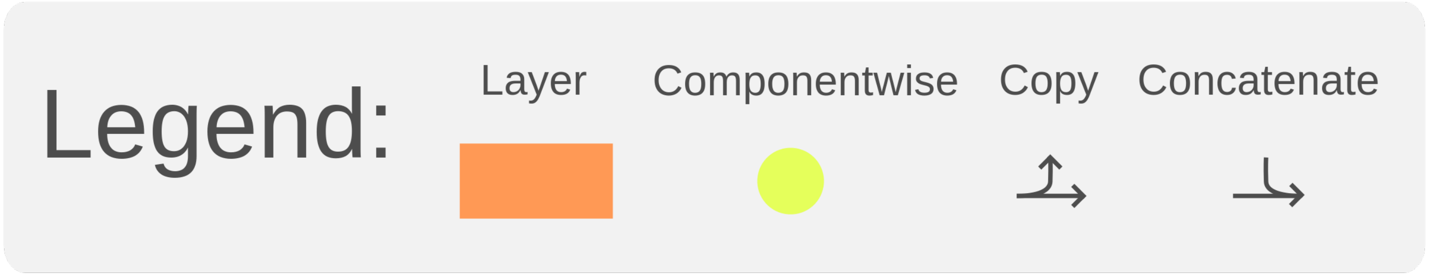
LSTM



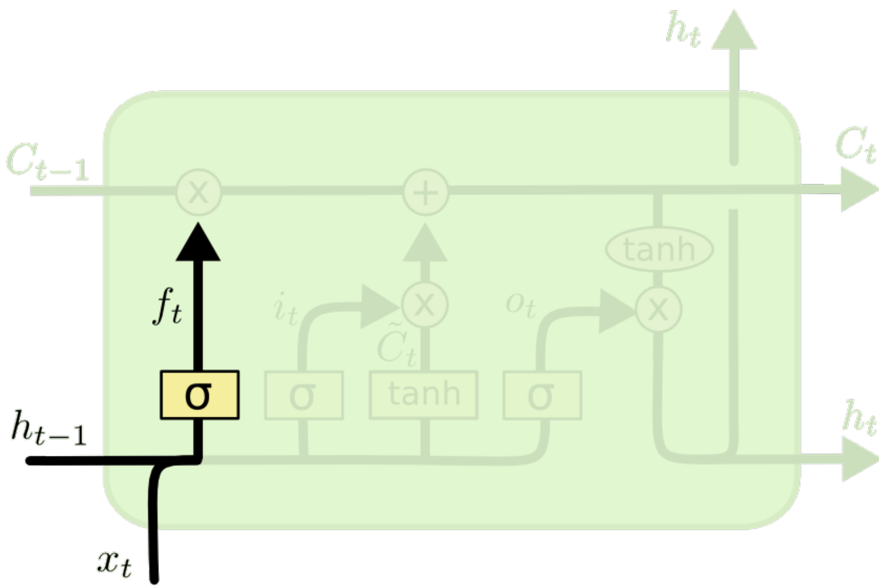
y^t
c^{t-1}
z^f
z^i
z
z^o
x^t
z^o
Output Gate
Forget Gate
Cell
z^f
z^i
Input Gate
Block
z

LSTM Detailed Structure





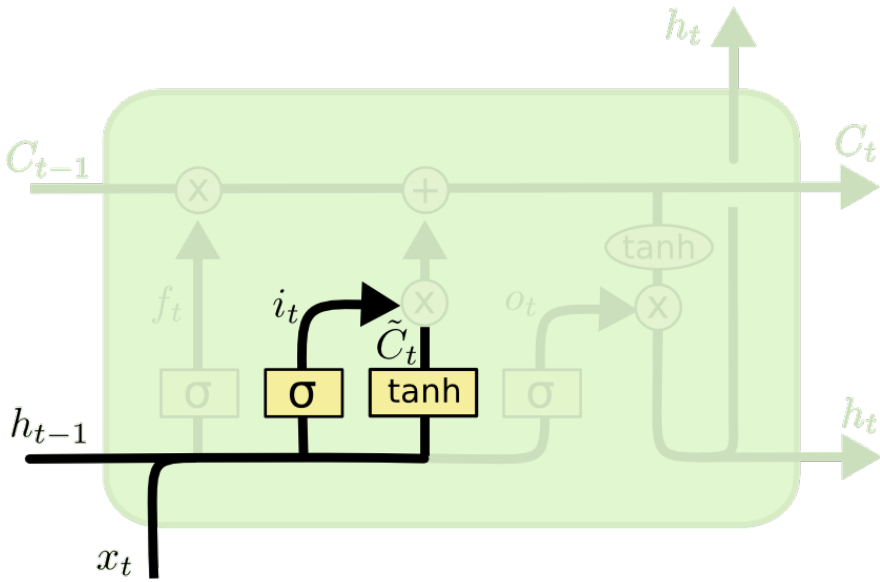
LSTM Detailed Structure



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

LSTM Detailed Structure

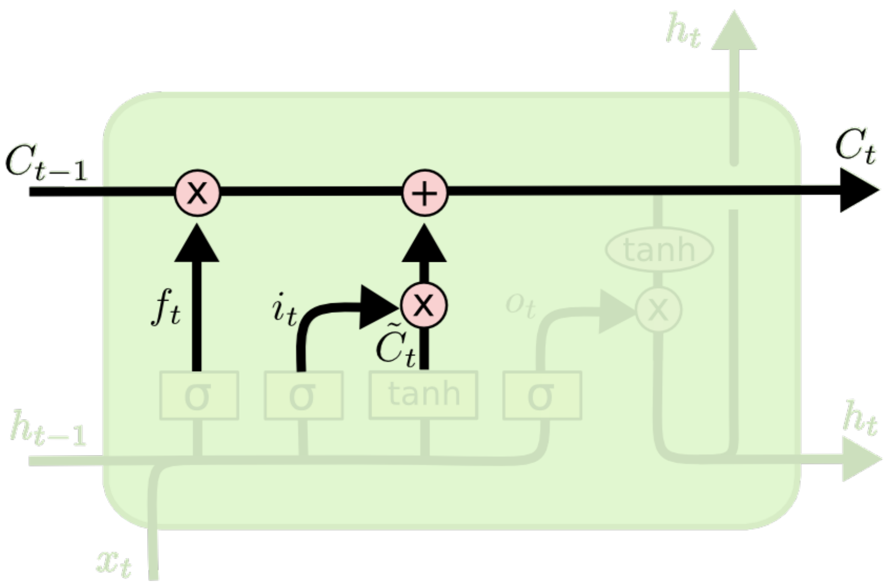


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\widetilde{C}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

LSTM Detailed Structure

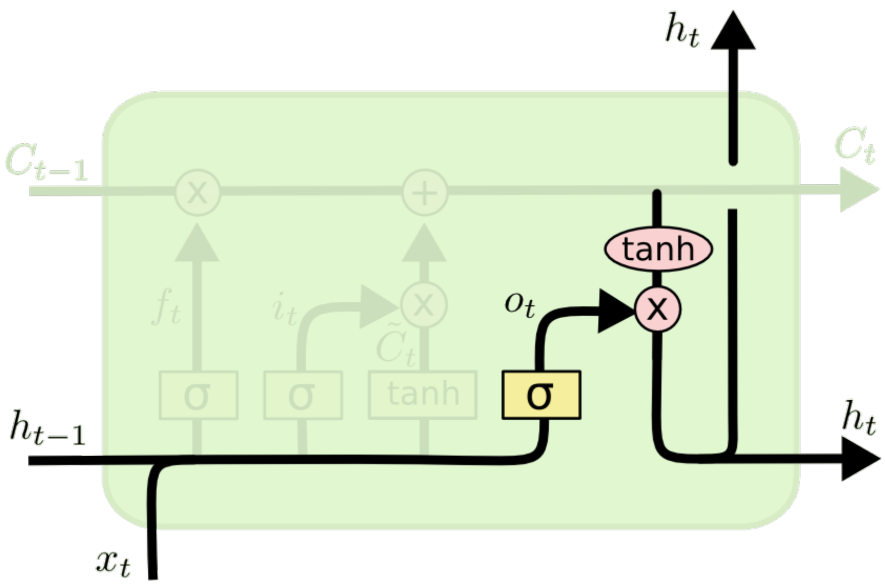


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \widetilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

LSTM Detailed Structure



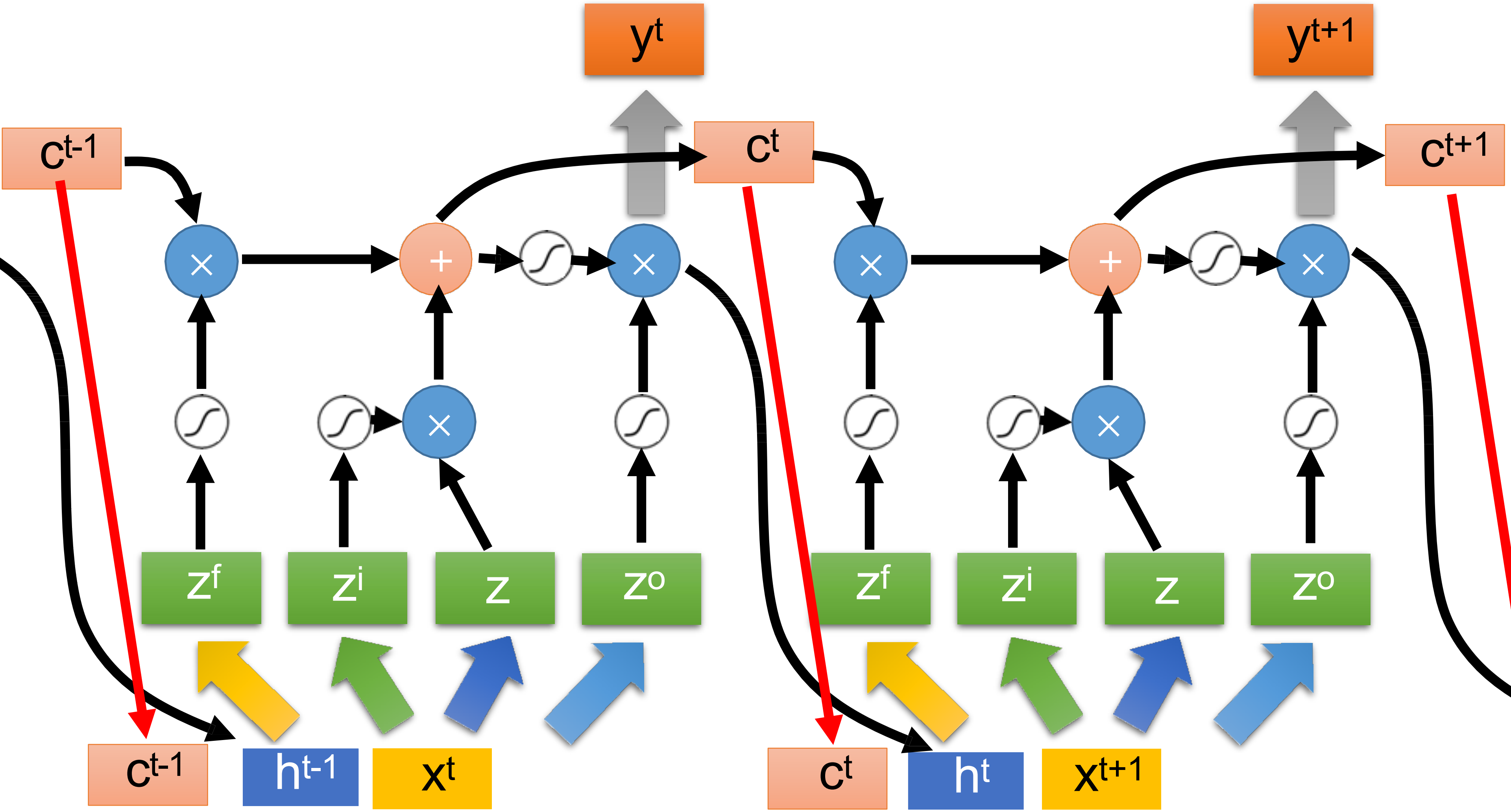
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

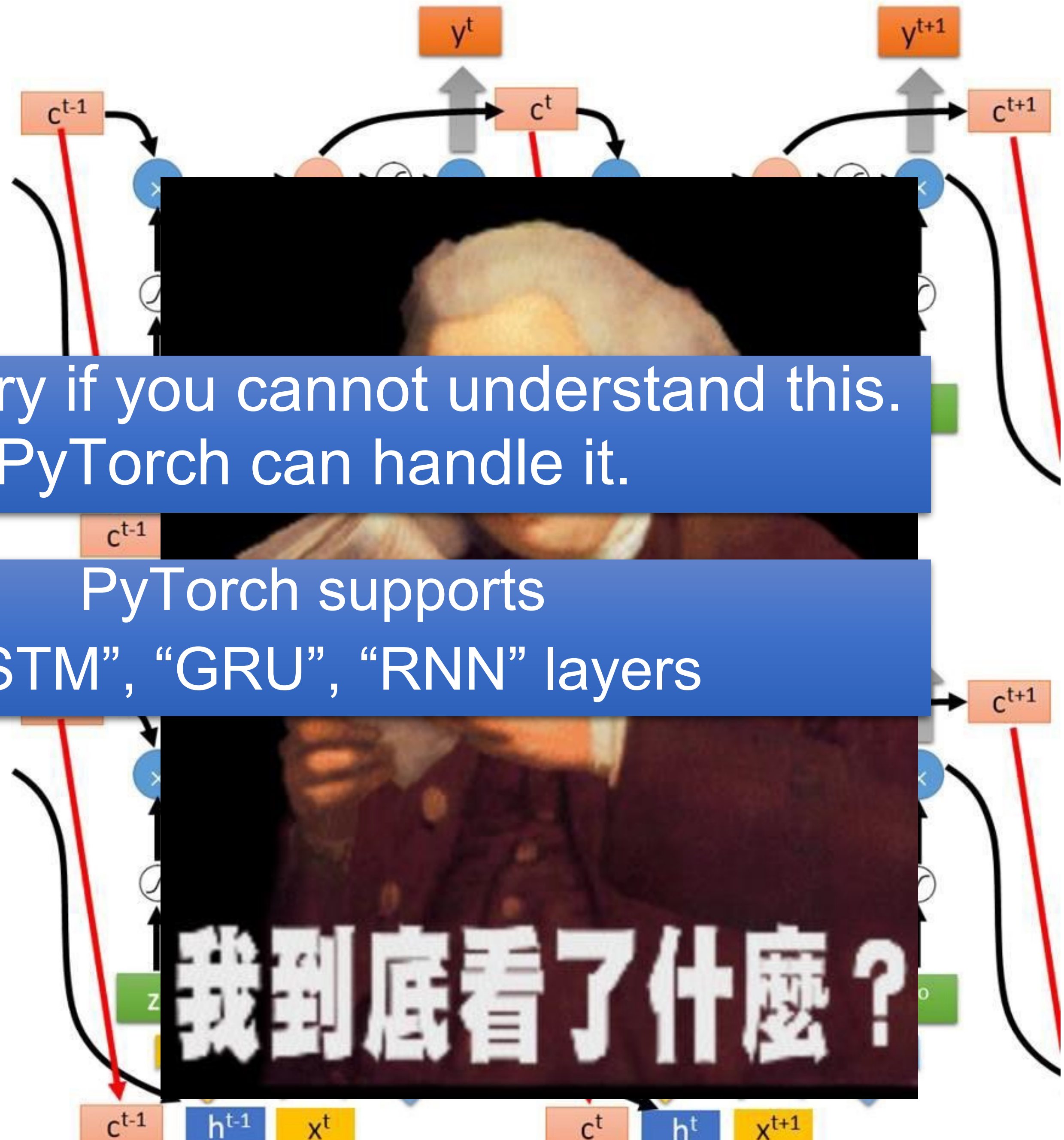
Finally, we need to decide what we're going to output. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

LSTM

Extension: “peephole”



Multiple-layer LSTM



Don’t worry if you cannot understand this.
PyTorch can handle it.

PyTorch supports
“LSTM”, “GRU”, “RNN” layers

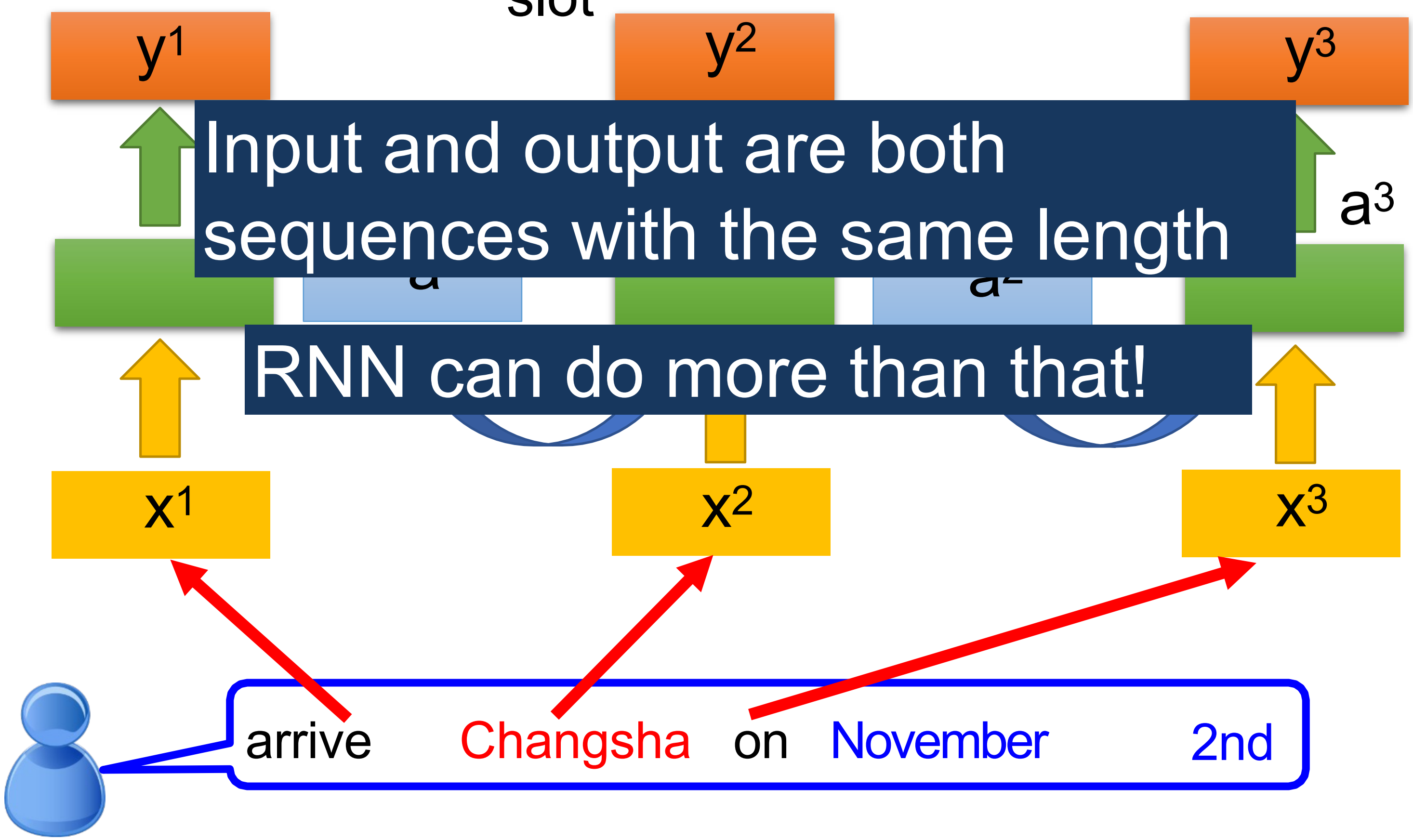
This is quite standard now.

More Applications ……

Probability of “arrive” in each slot

Probability of “Changsha” in each slot

Probability of “on” in each slot



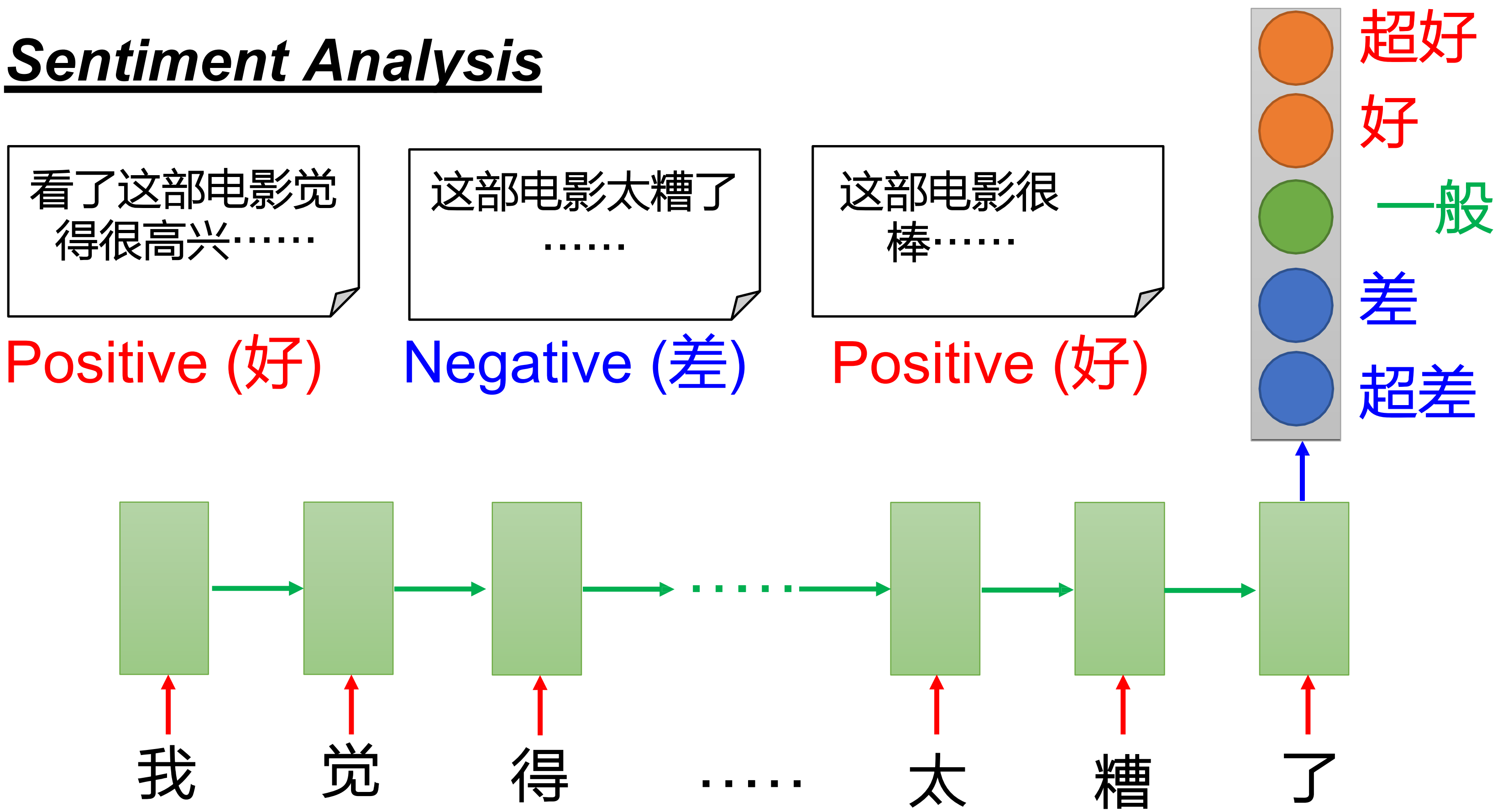
Input and output are both sequences with the same length

RNN can do more than that!

Many to one

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

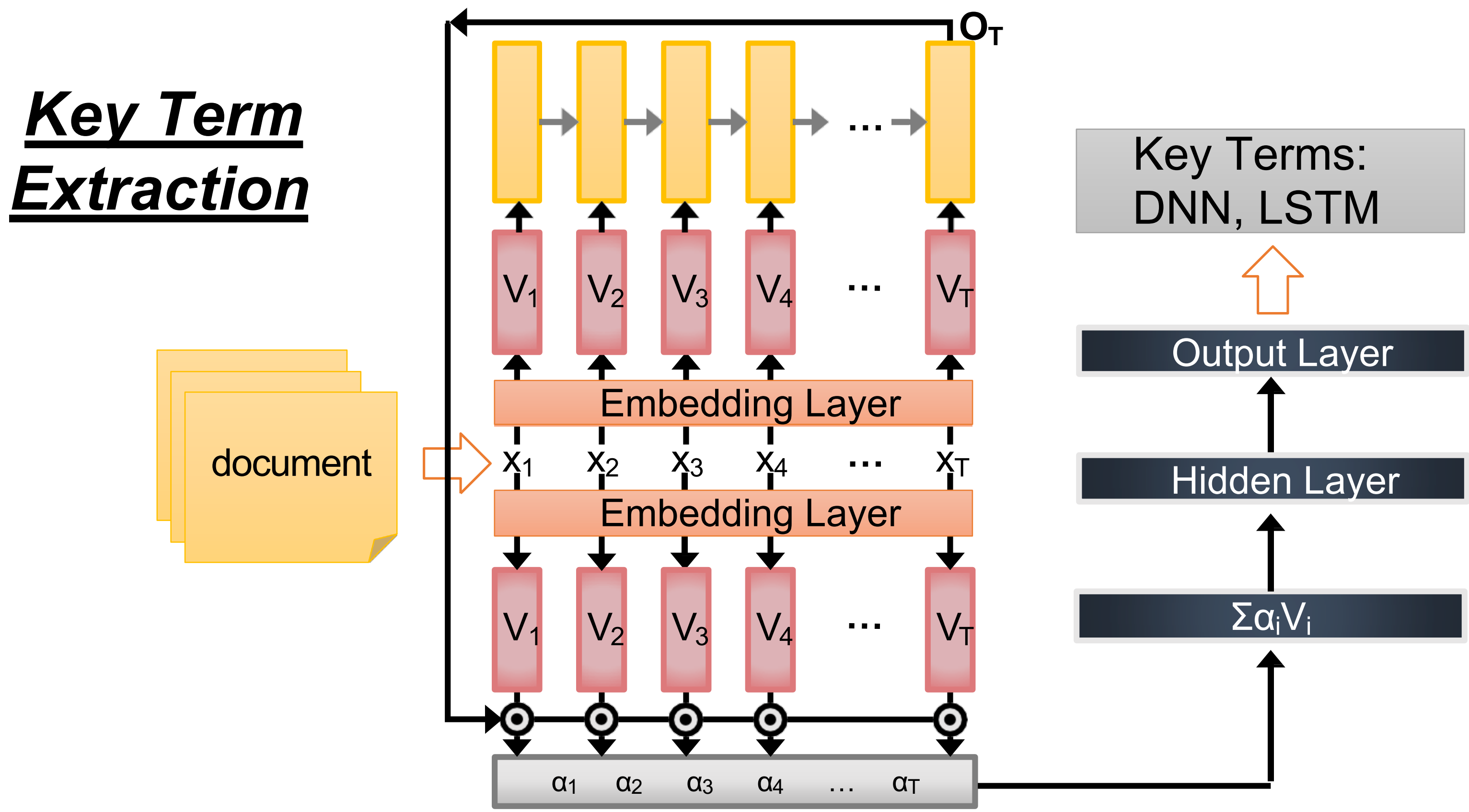
Sentiment Analysis



Many to one

[Shen & Lee, Interspeech 16]

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

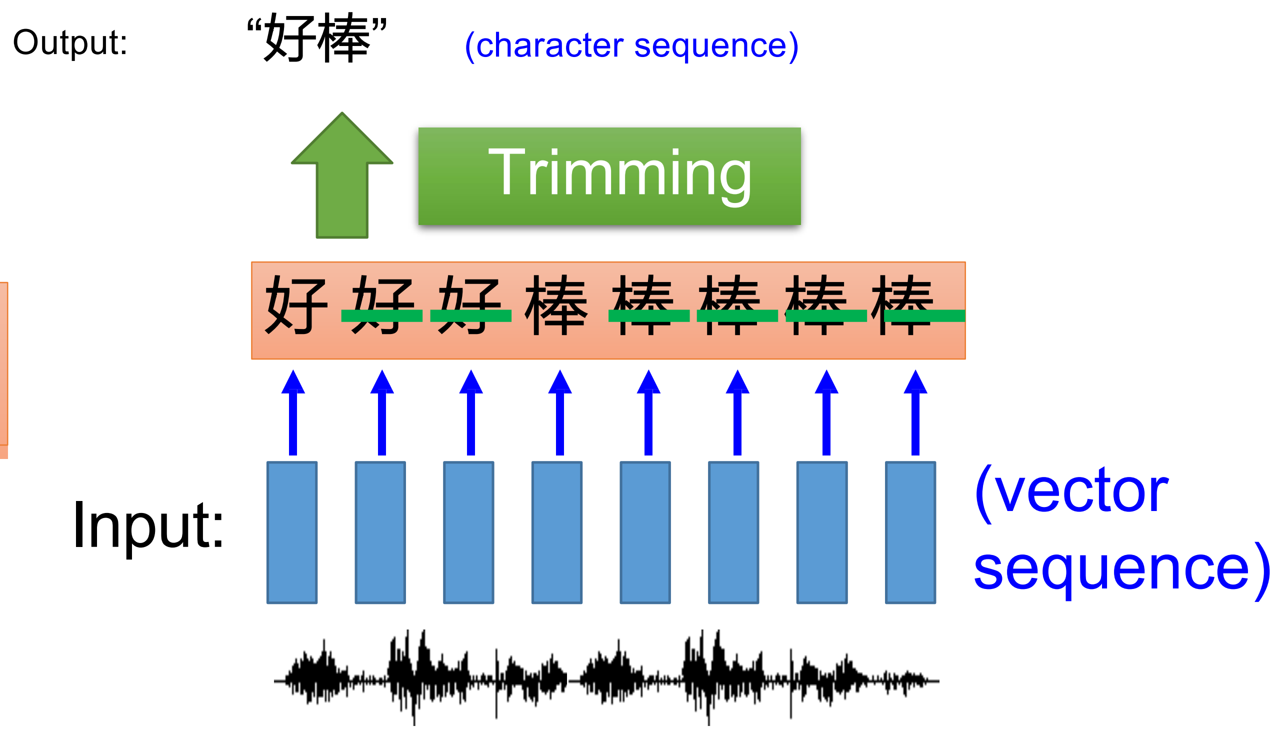


Many to Many (Output is shorter)

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

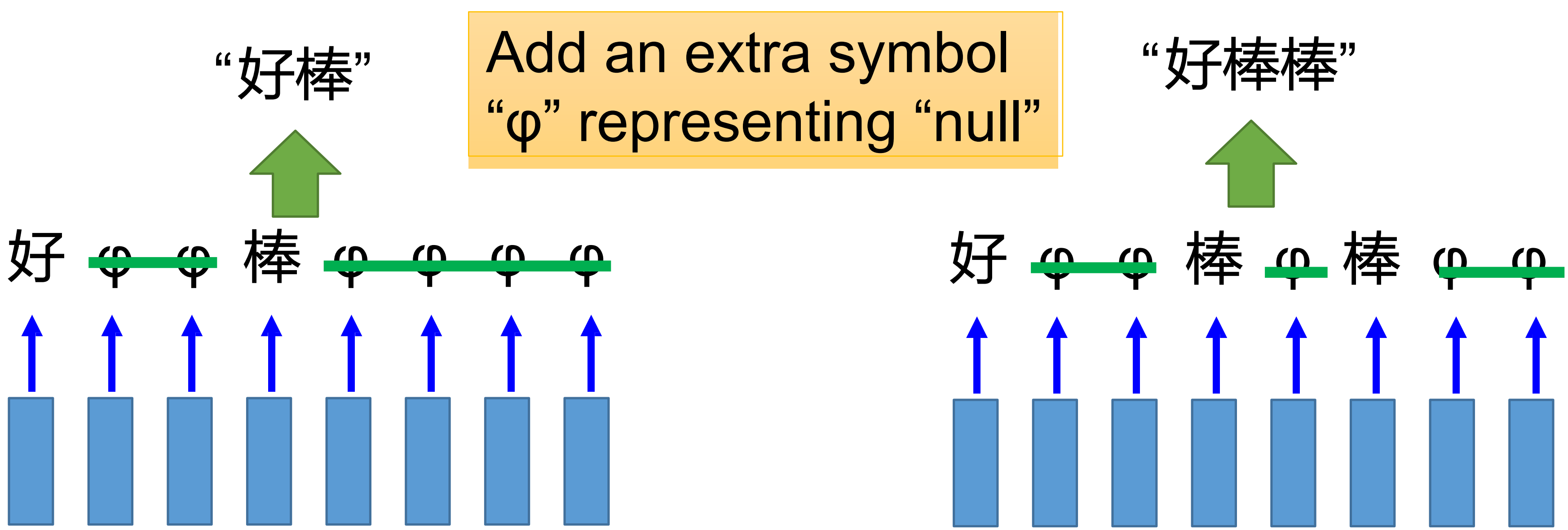
Problem?

Why can’t it be “好棒棒”



Many to Many (Output is shorter)

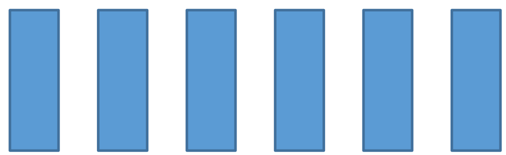
- Both input and output are both sequences, ***but the output is shorter.***
- Connectionist Temporal Classification (CTC) [Alex Graves, ICML’06][Alex Graves, ICML’14][Hasim Sak, Interspeech’15][Jie Li, Interspeech’15][Andrew Senior, ASRU’15]



Many to Many (Output is shorter)

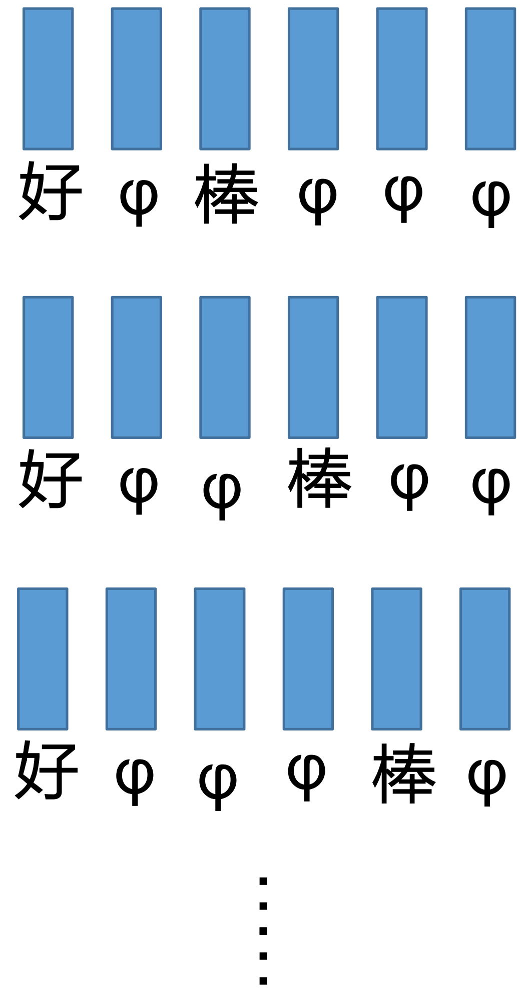
- CTC: Training

Acoustic Features:



Label: 好 棒

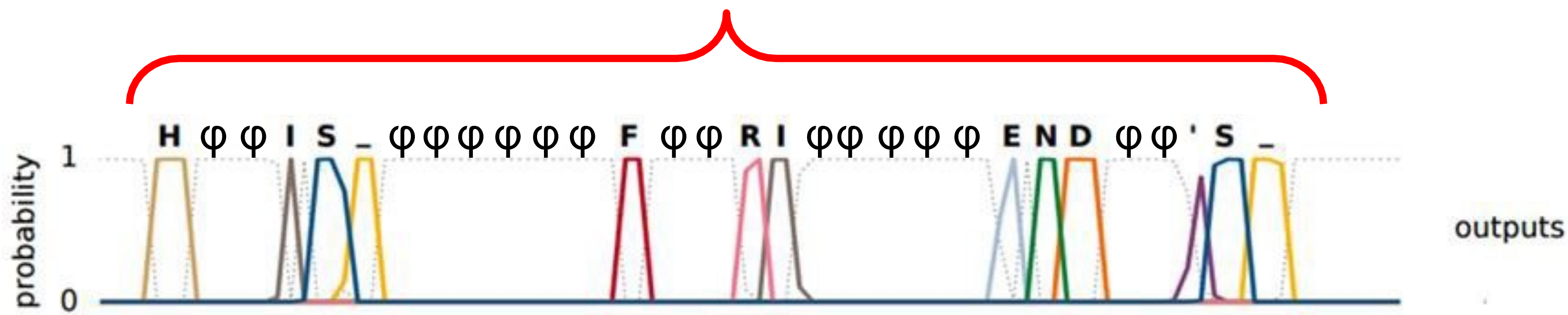
All possible alignments are considered as correct.



Many to Many (Output is shorter)

- CTC: example

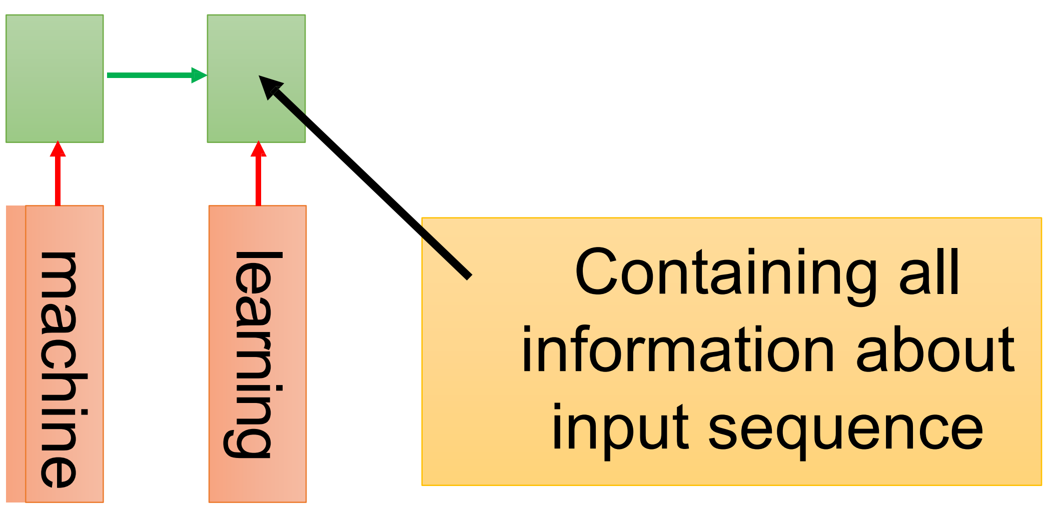
HIS FRIEND’S



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.

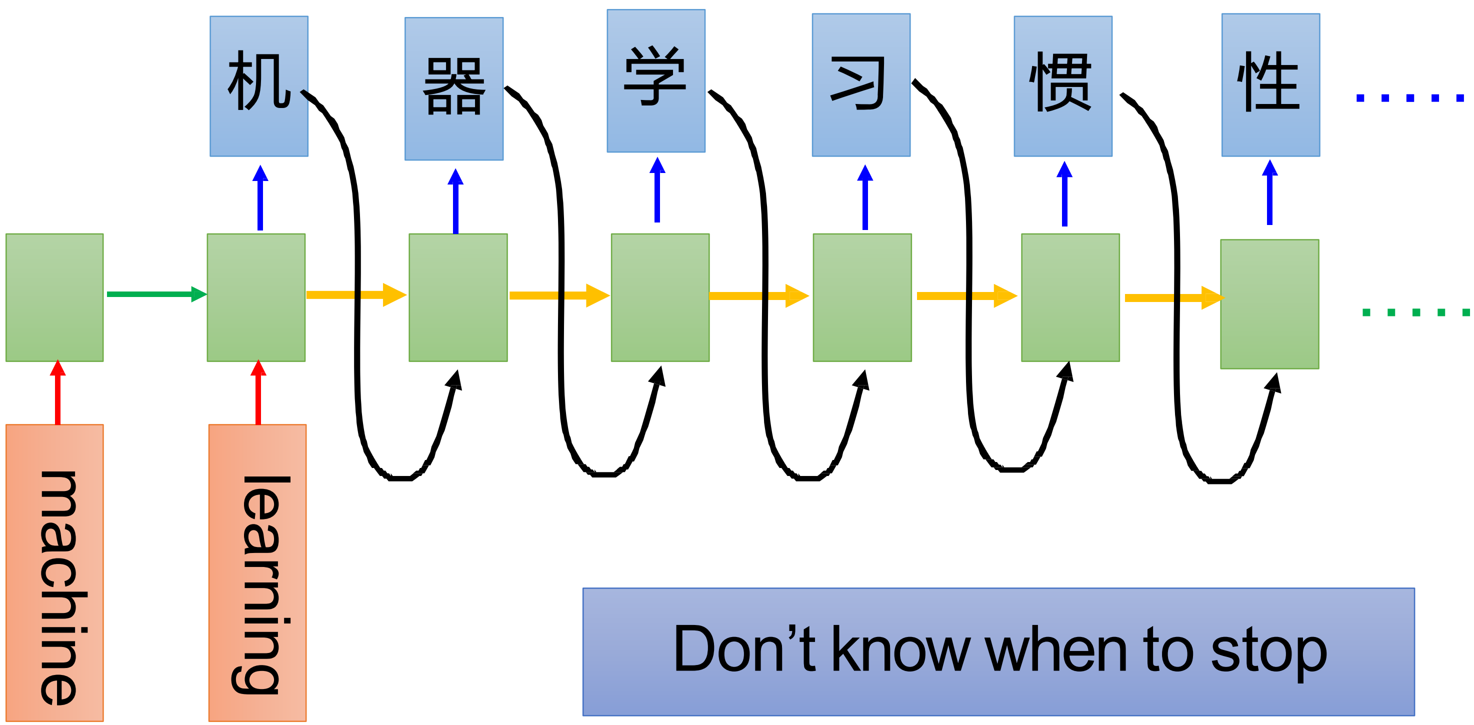
Many to Many (No Limitation)

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



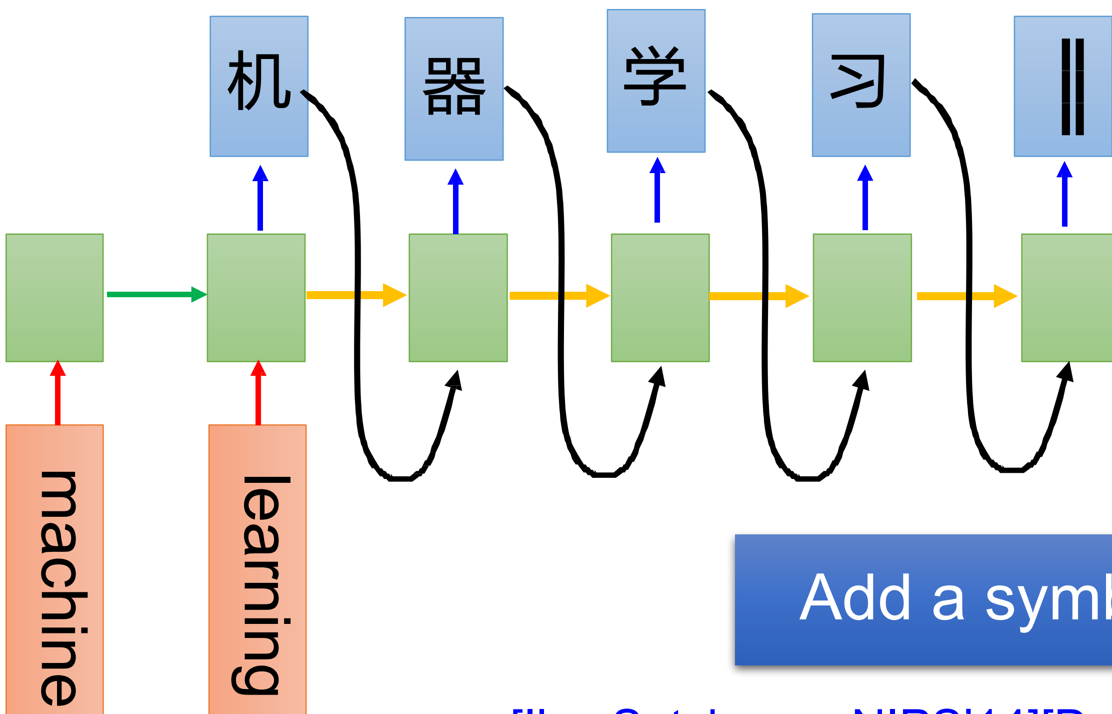
Many to Many (No Limitation)

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



Many to Many (No Limitation)

- Both input and output are both sequences ***<u>with different lengths</u>***. → ***<u>Sequence to sequence learning</u>***
    - E.g. ***<u>Machine Translation</u>*** (machine learning→机器学习)

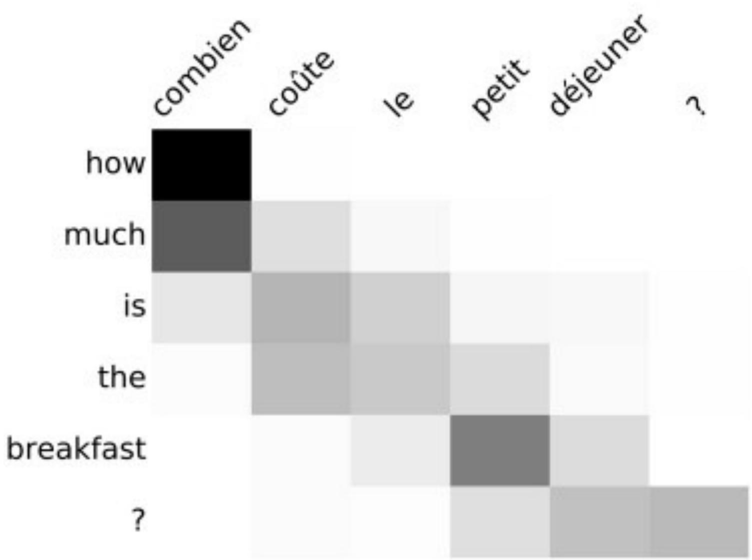


Add a symbol "===" (断)

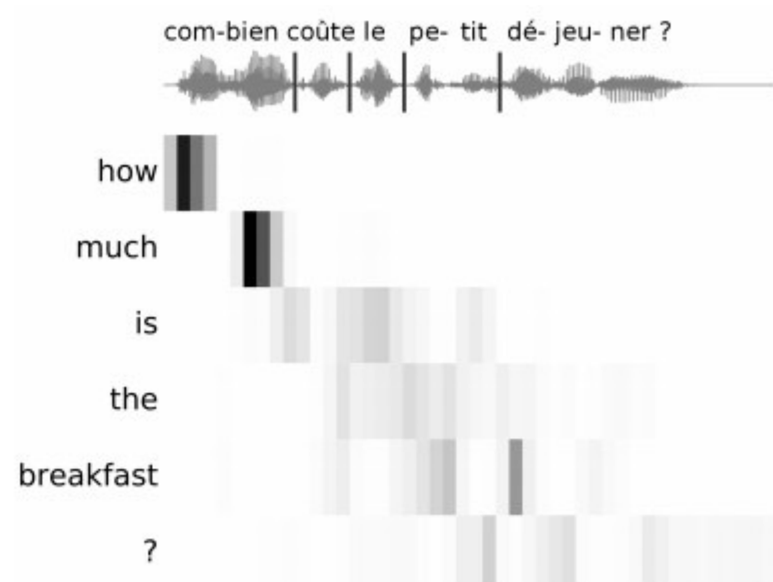
[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15] https://arxiv.org/pdf/1612.01744v1.pdf

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→机器学习)



(a) Machine translation alignment

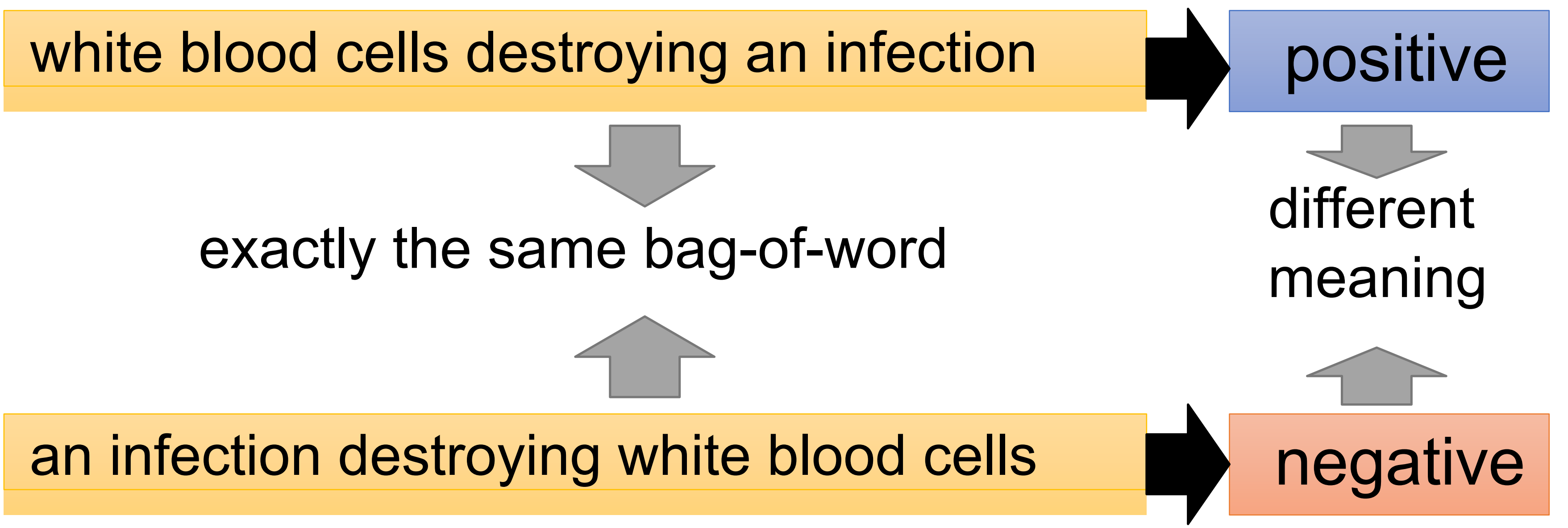


(b) Speech translation alignment

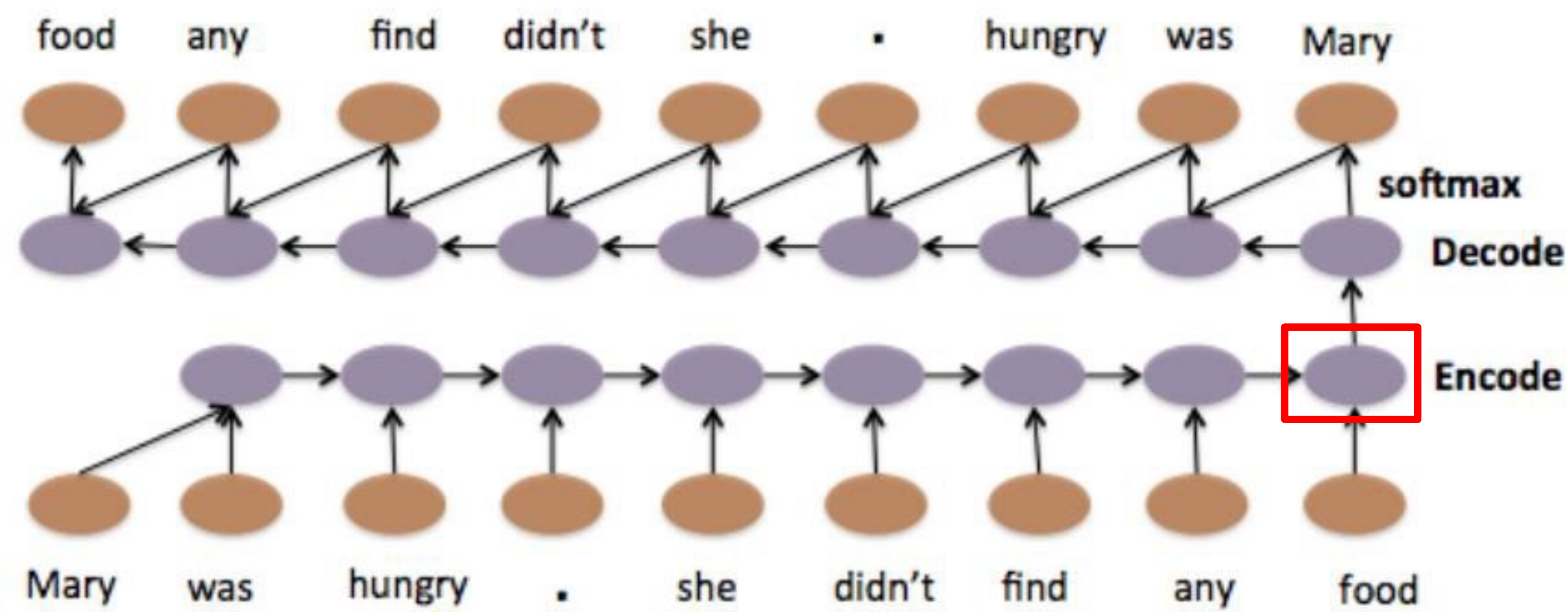
Figure 1: Alignments performed by the attention model during training

Sequence-to-sequence Auto-encoder -Text

- To understand the meaning of a word sequence, the order of the words can not be ignored.

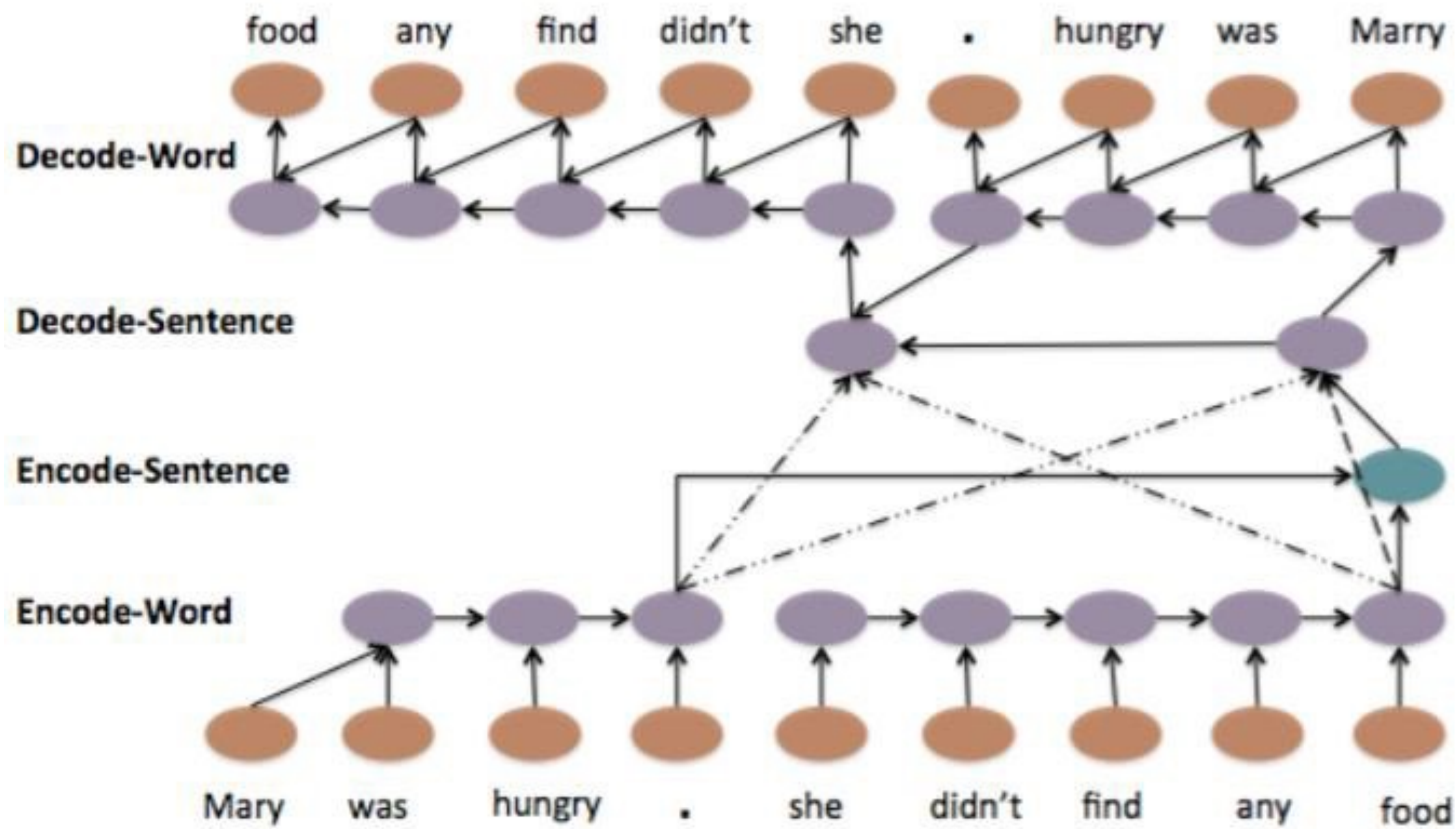


Sequence-to-sequence Auto-encoder -Text



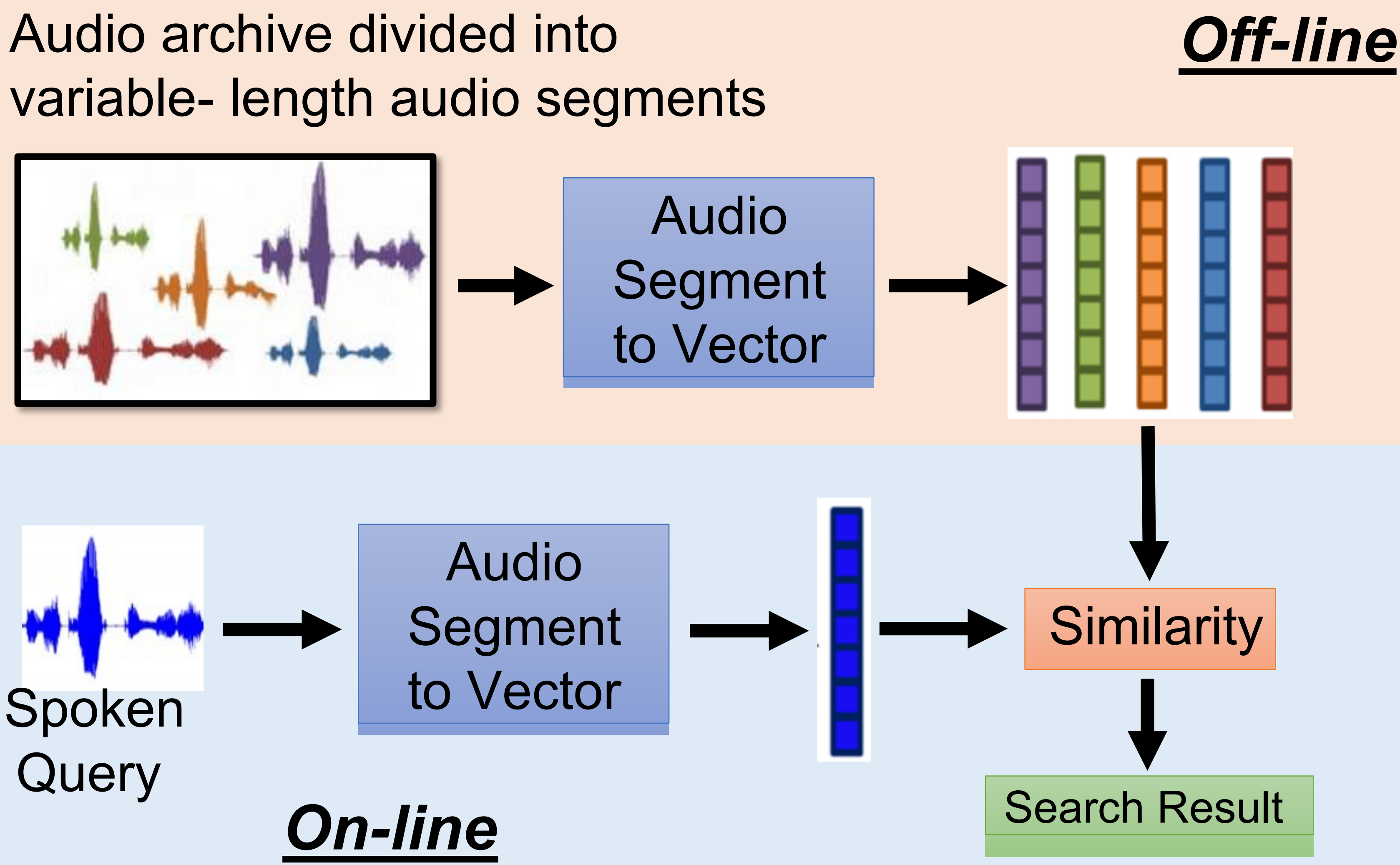
Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." *arXiv preprint arXiv:1506.01057*(2015).

Sequence-to-sequence Auto-encoder -Text

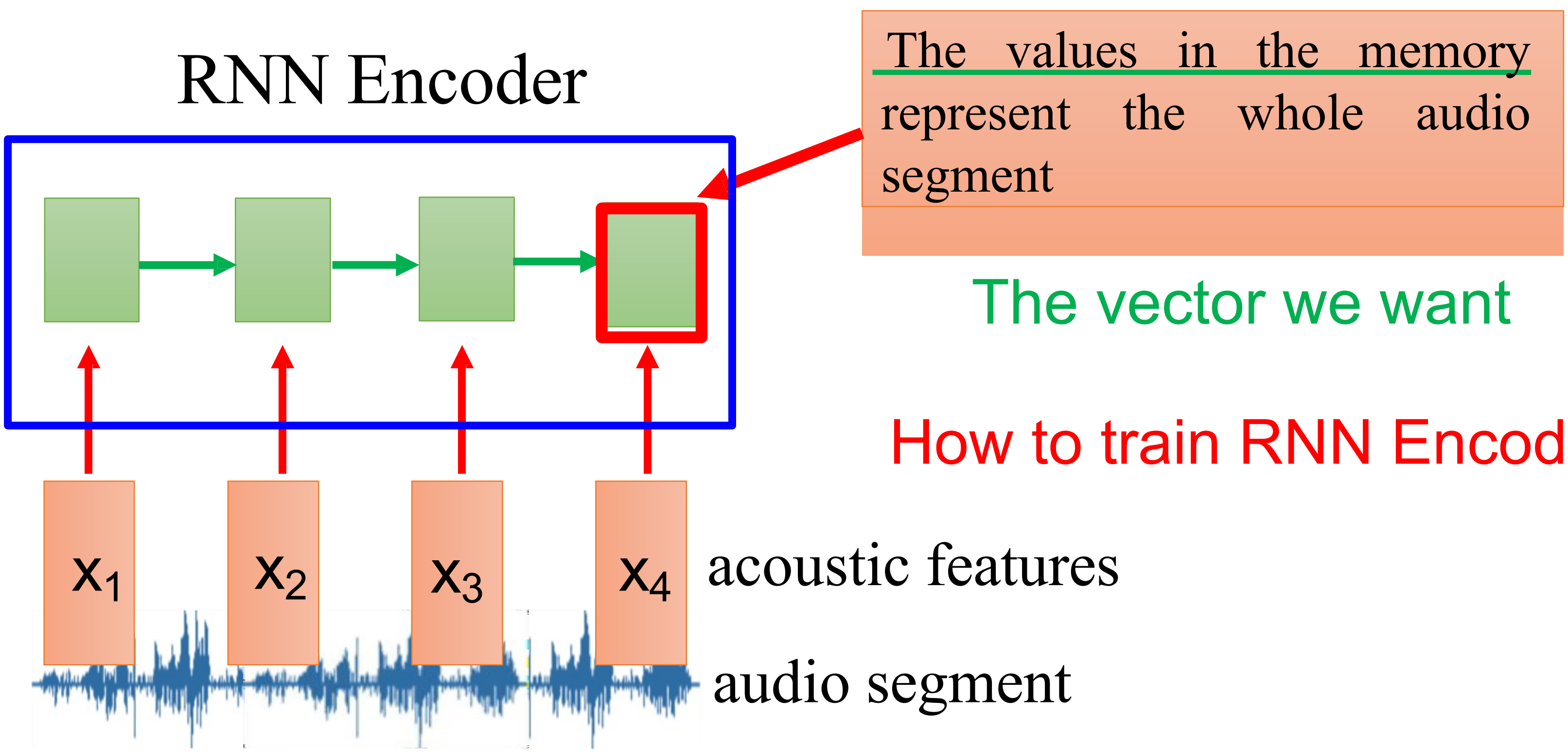


Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." *arXiv preprint arXiv:1506.01057*(2015).

Sequence-to-sequence Auto-encoder -Speech



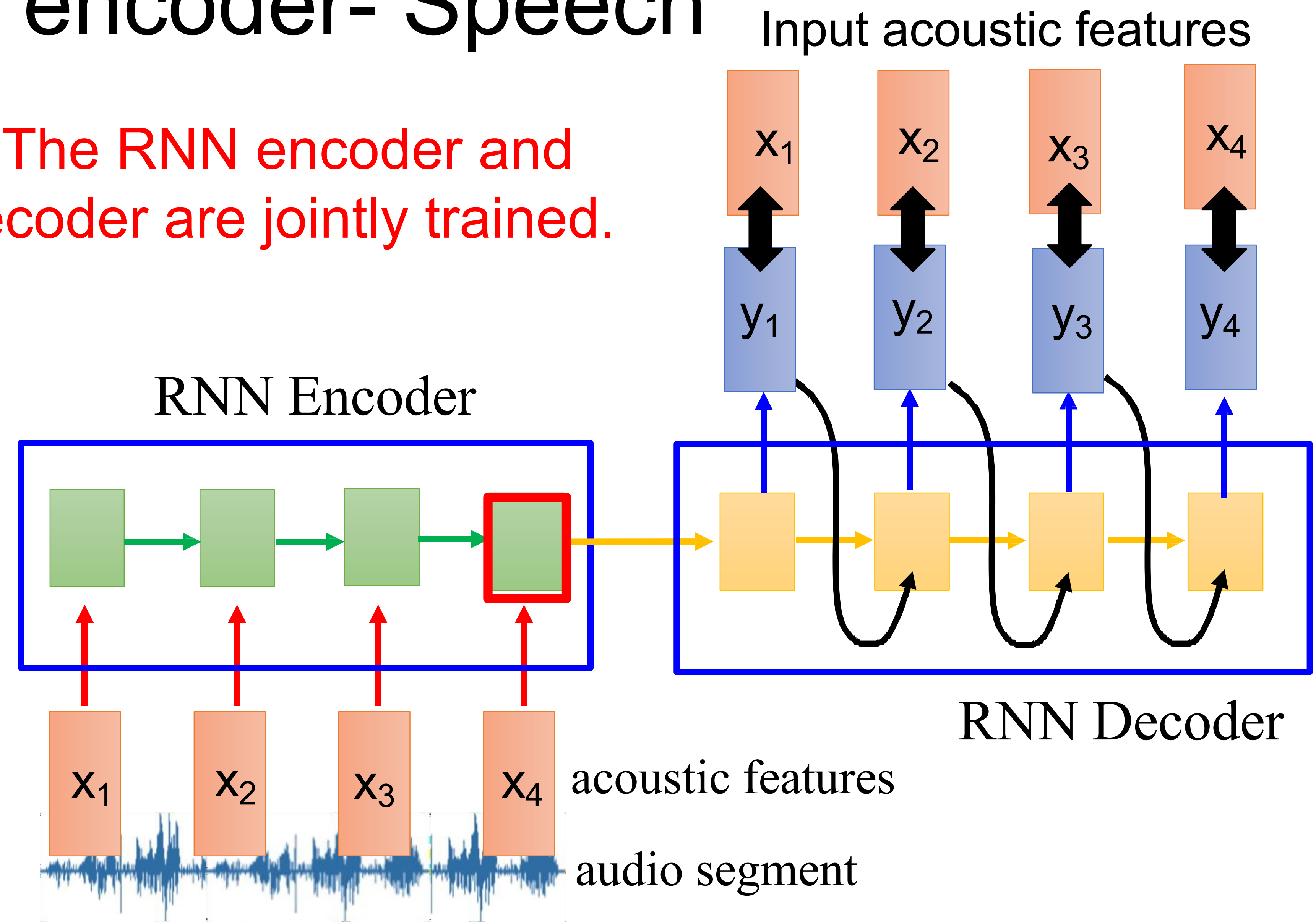
Sequence-to-sequence Auto-encoder -Speech



How to train RNN Encoder?

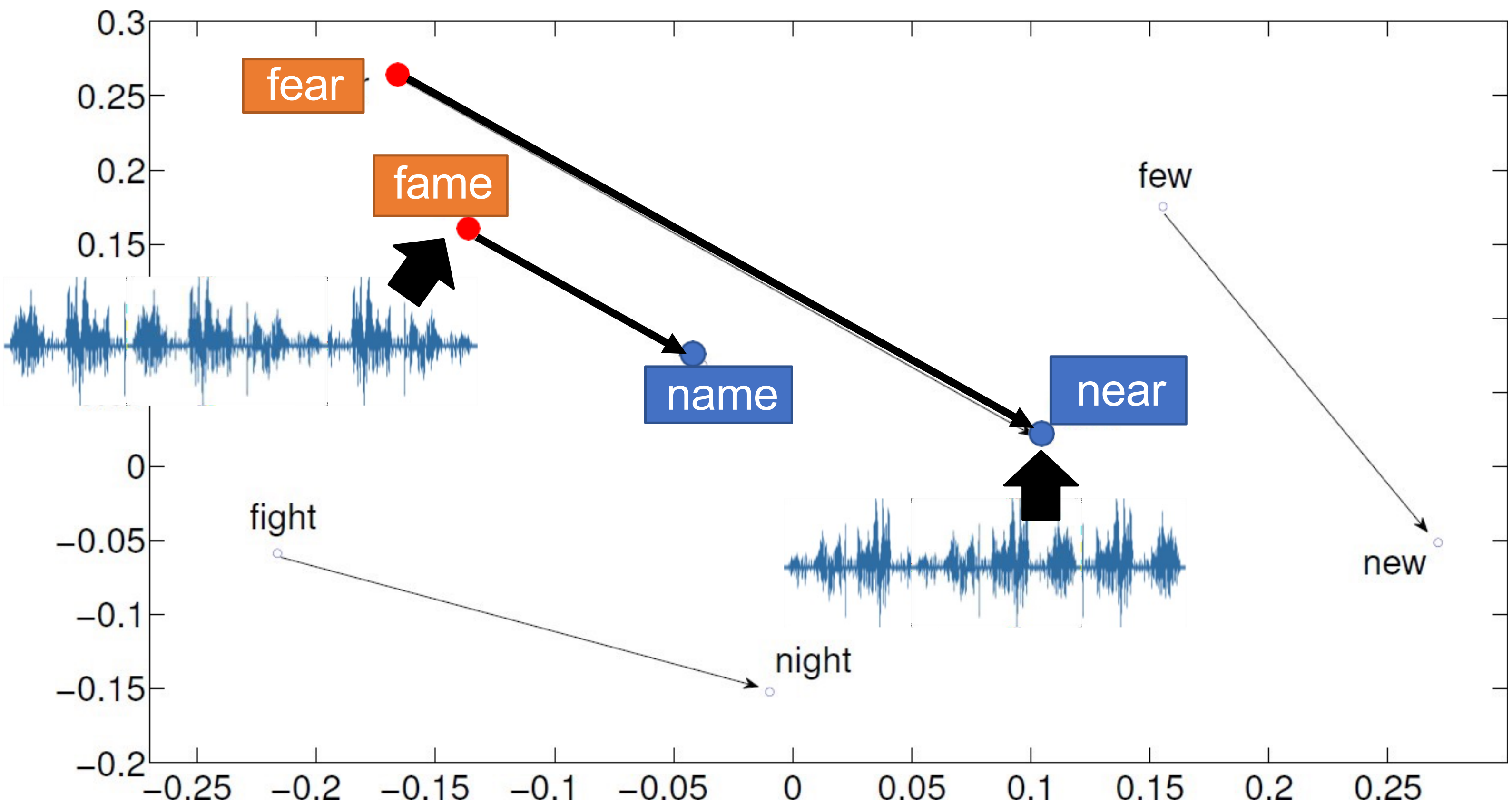
Sequence-to-sequence Auto-encoder- Speech

The RNN encoder and decoder are jointly trained.

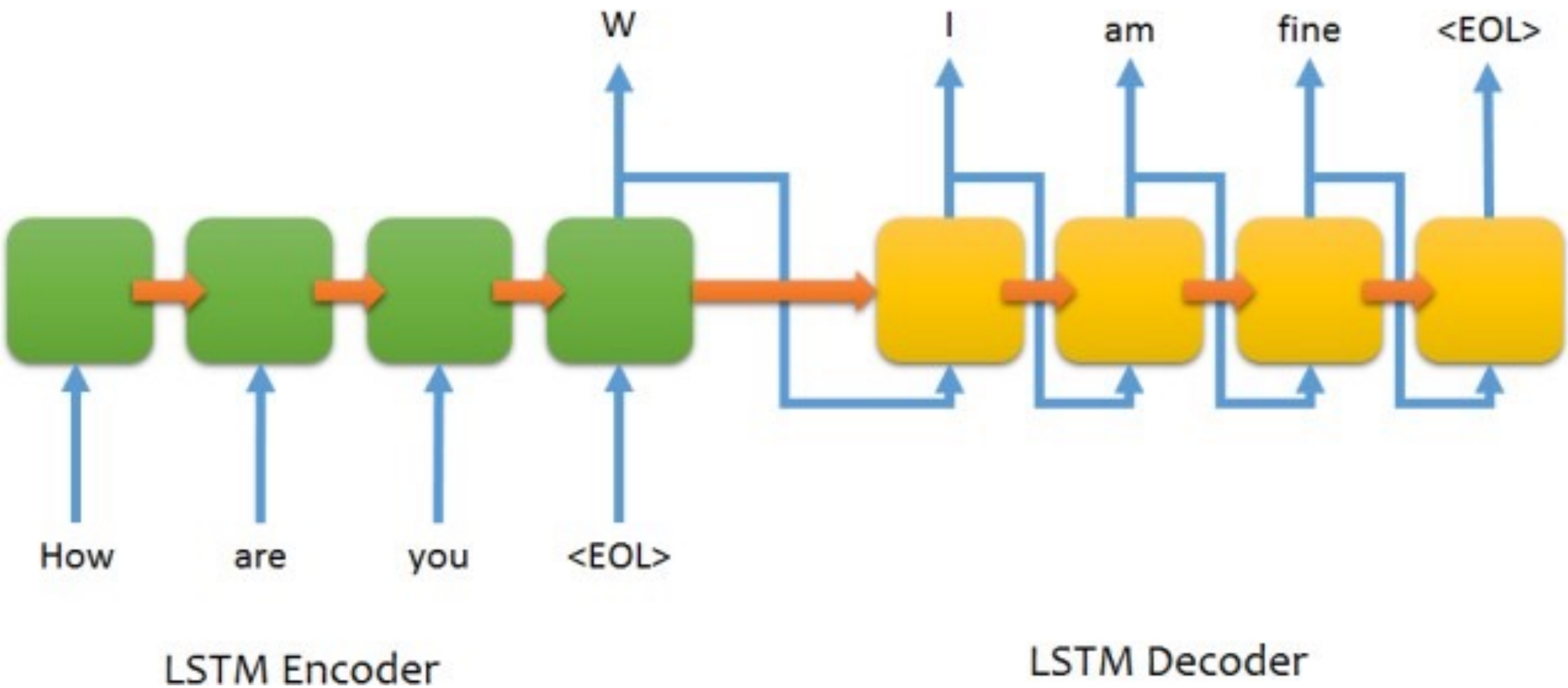


Sequence-to-sequence Auto-encoder -Speech

- Visualizing embedding vectors of the words



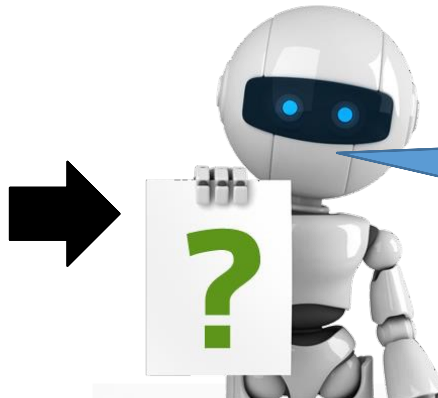
Example: Chat-bot



电视影集 (~40,000 sentences)、美国总统大选辩论

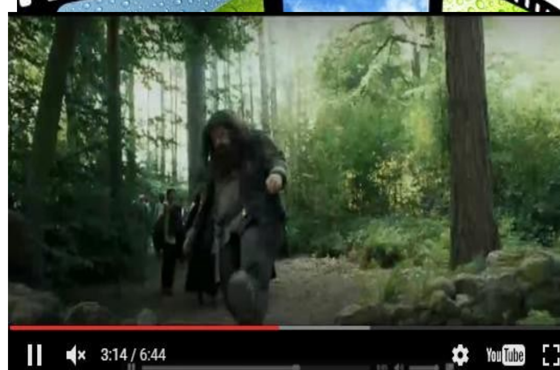
Example: Video Caption Generation

Video



A girl is running.

A group of people is knocked by a tree.

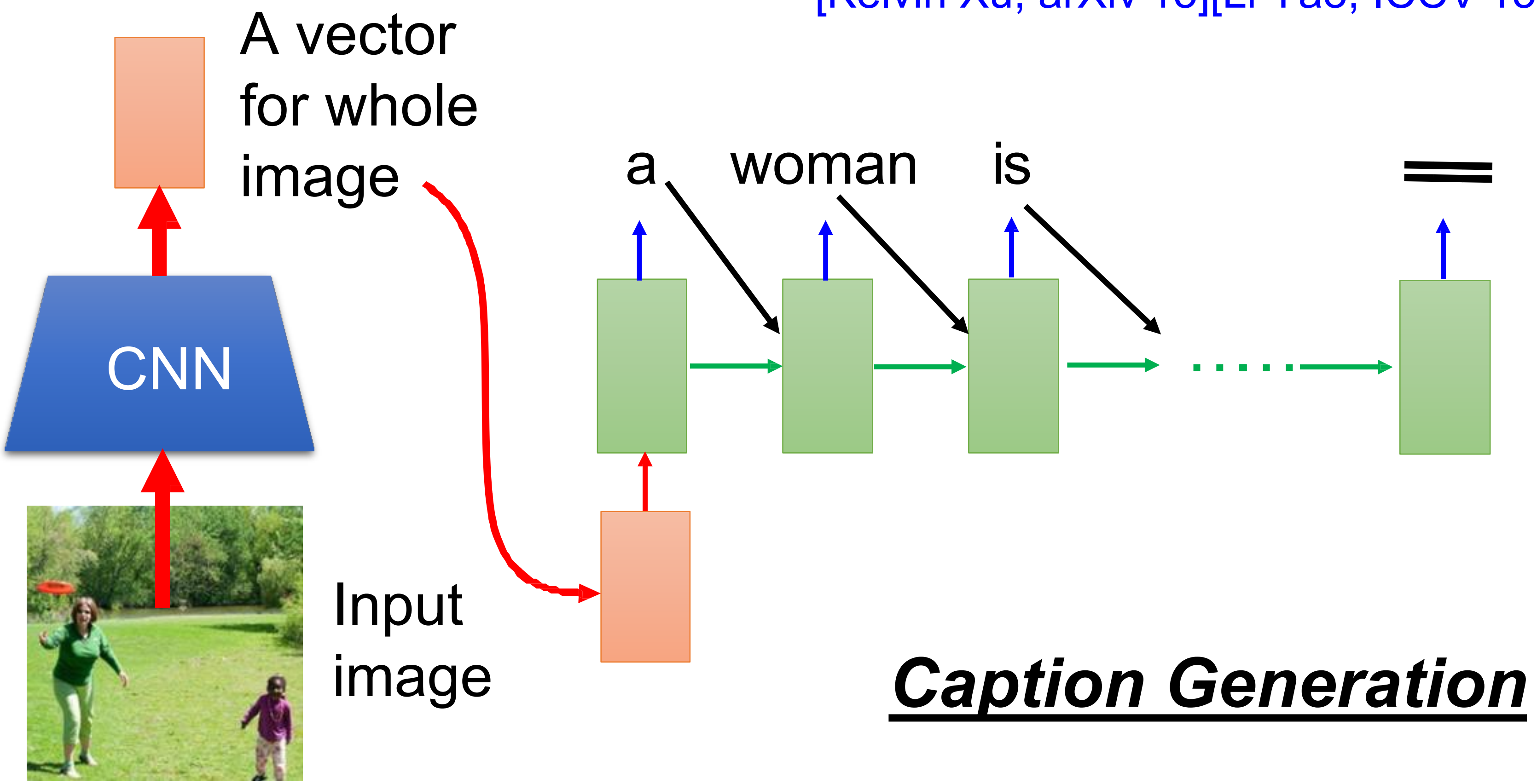


A group of people is walking in the forest.

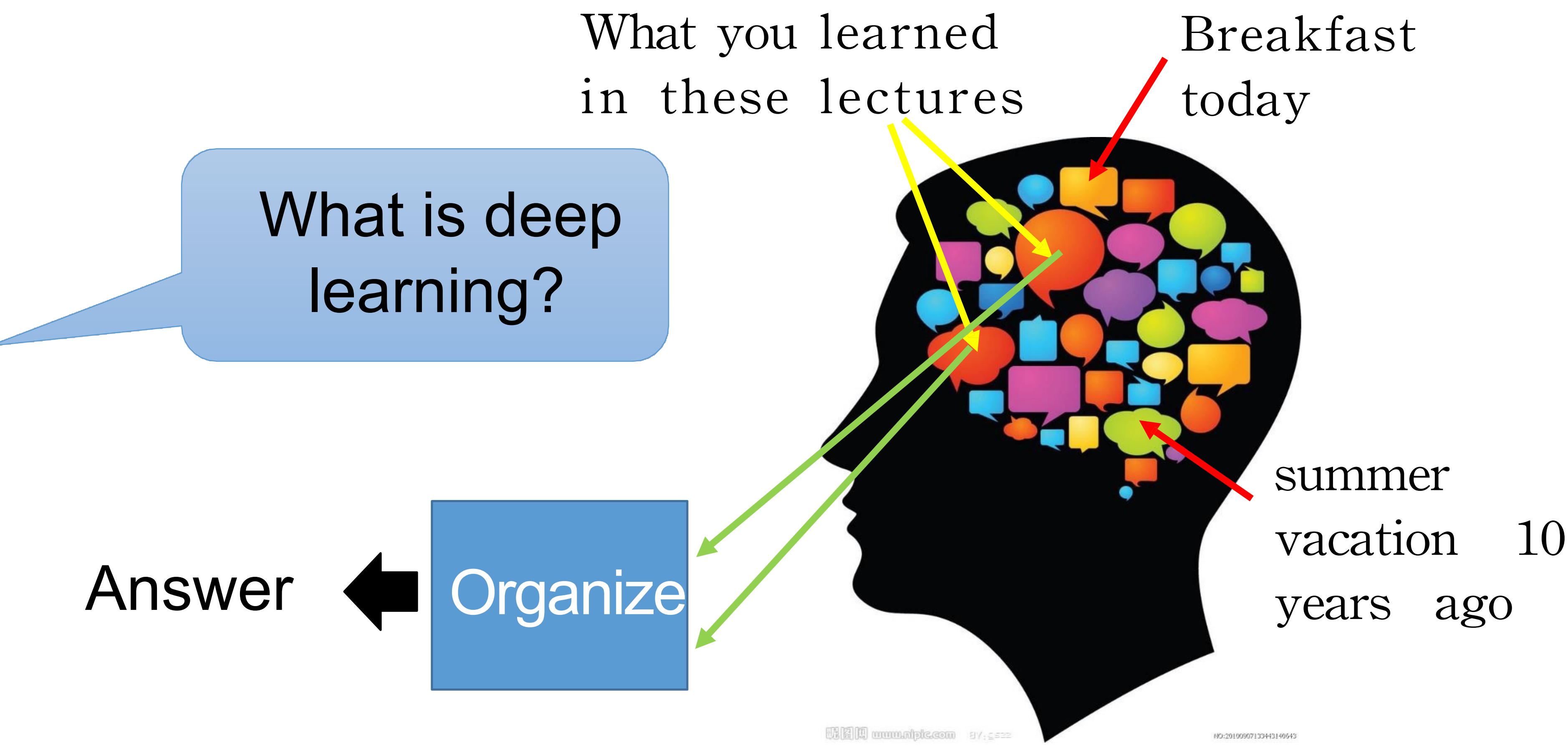
Example: Image Caption Generation

- Input an image, but output a sequence of words

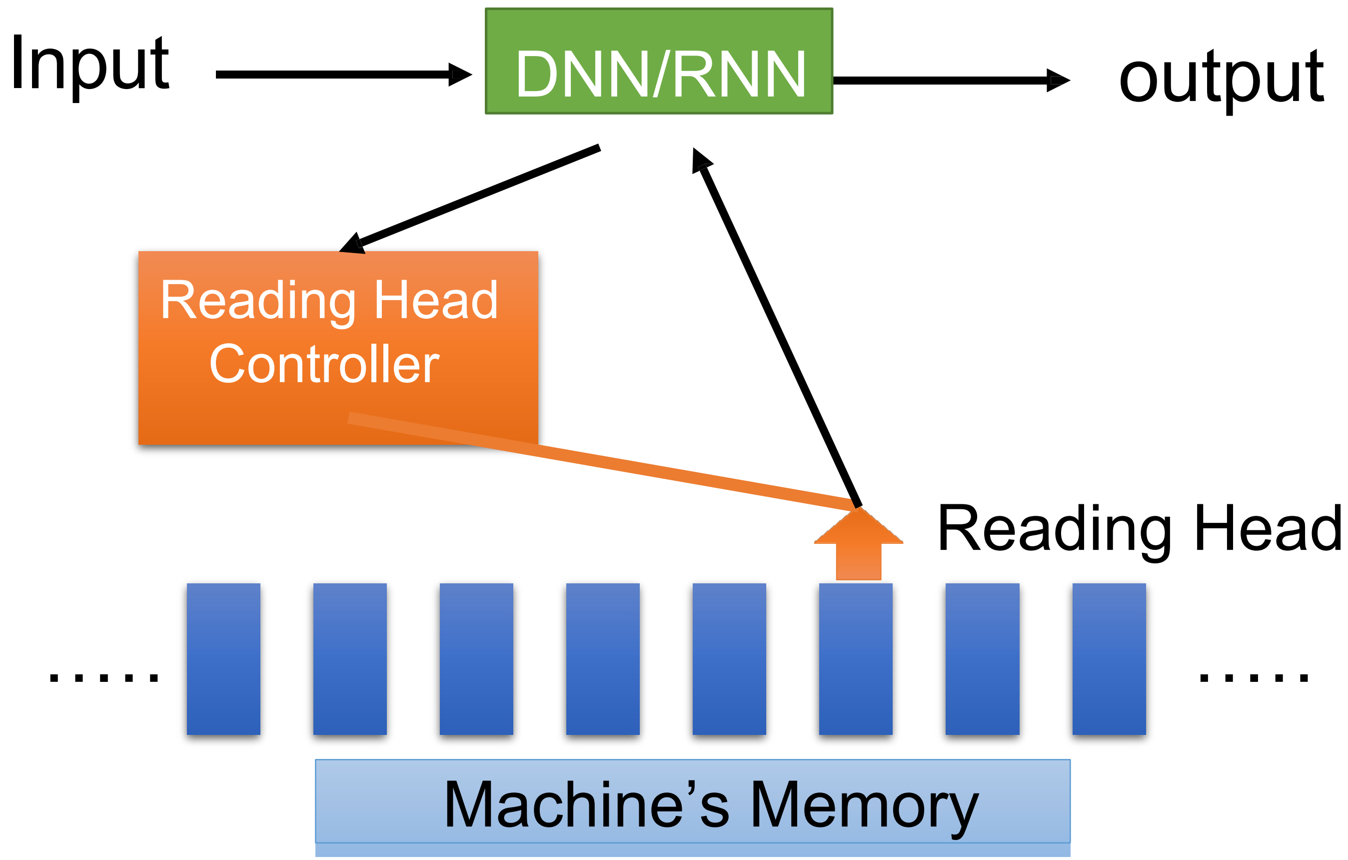
[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]



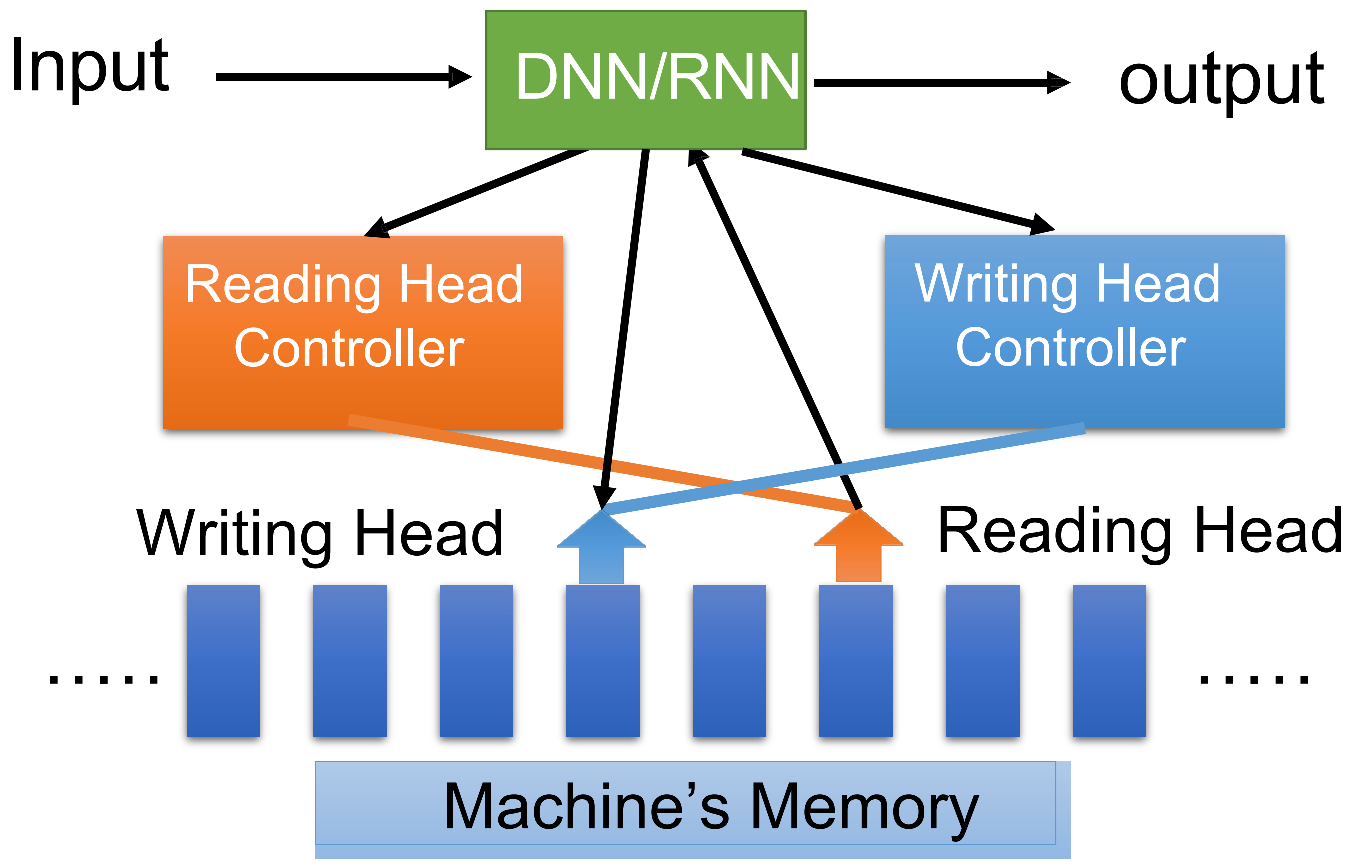
Attention-based Model



Attention-based Model

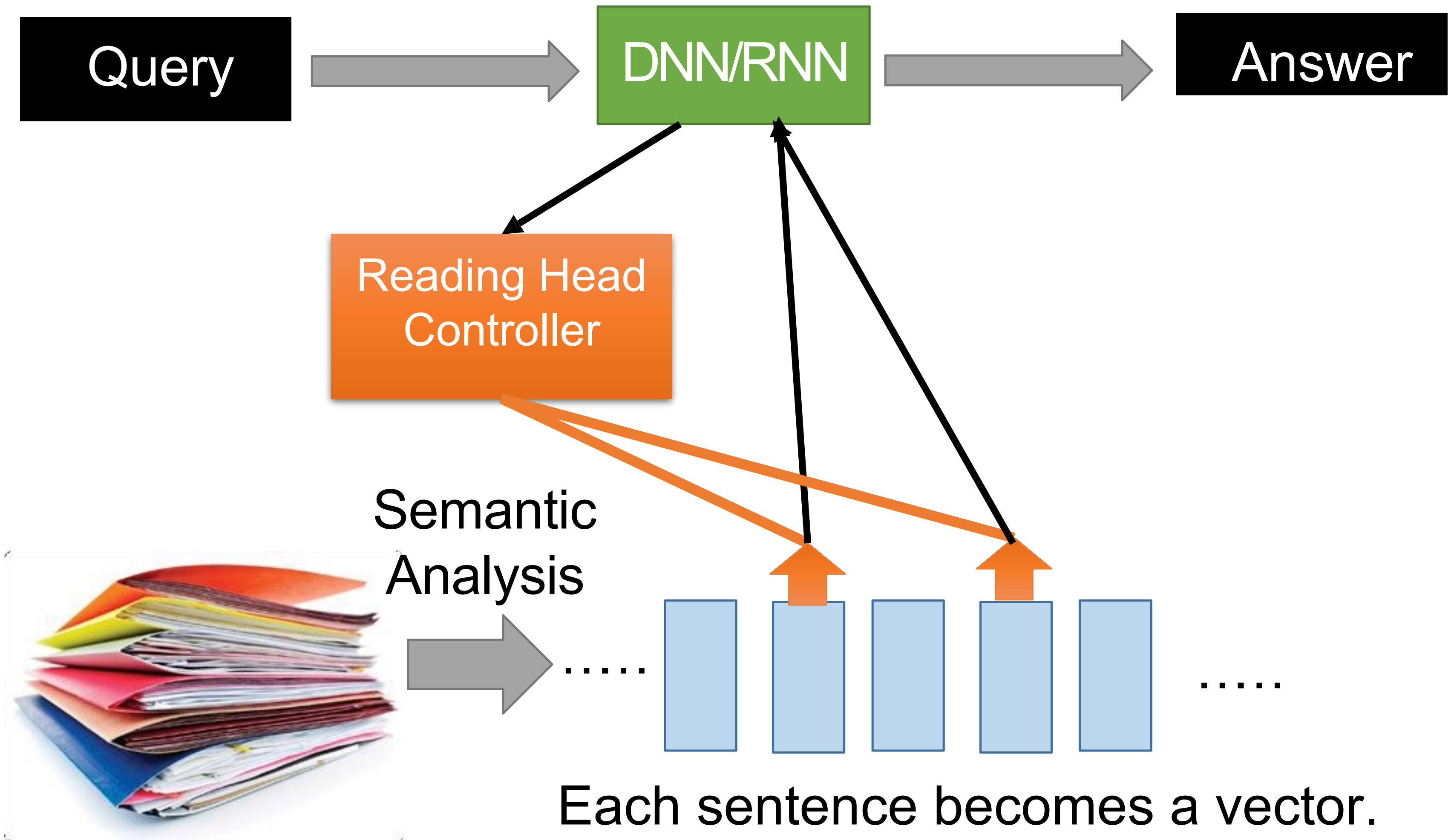


Attention-based Model v2



Neural Turing Machine

Reading Comprehension



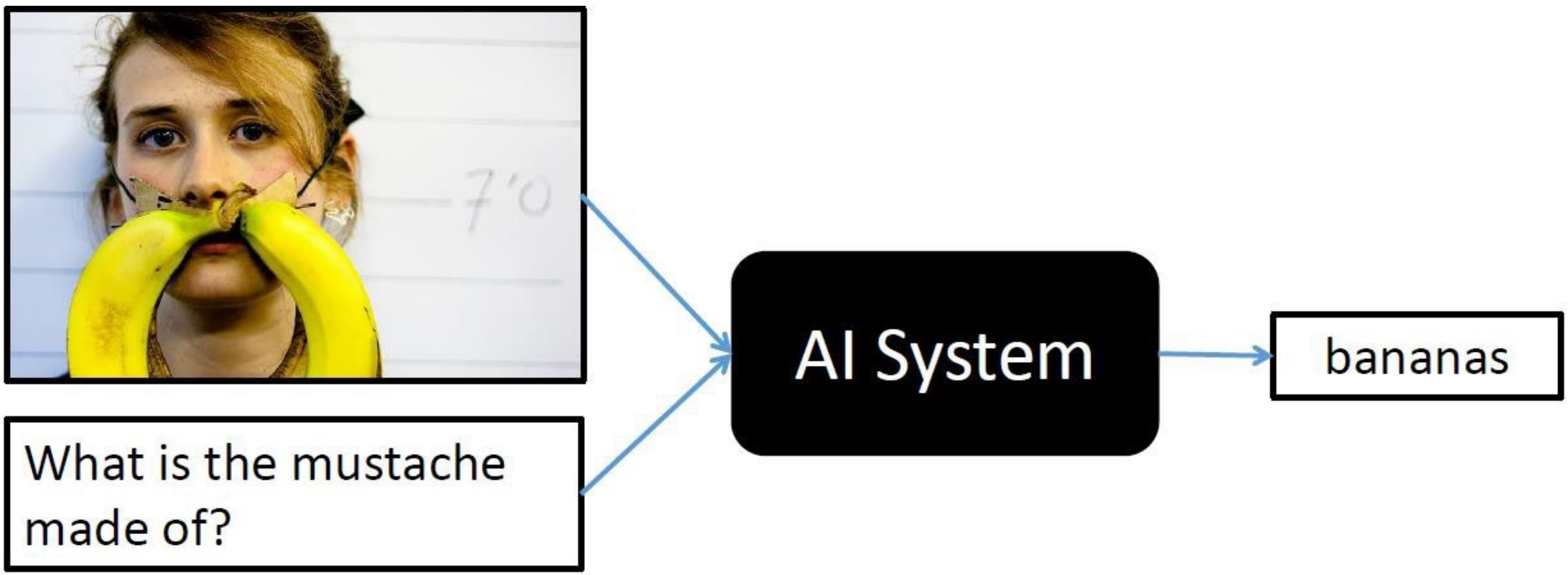
Reading Comprehension

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

The position of reading head:

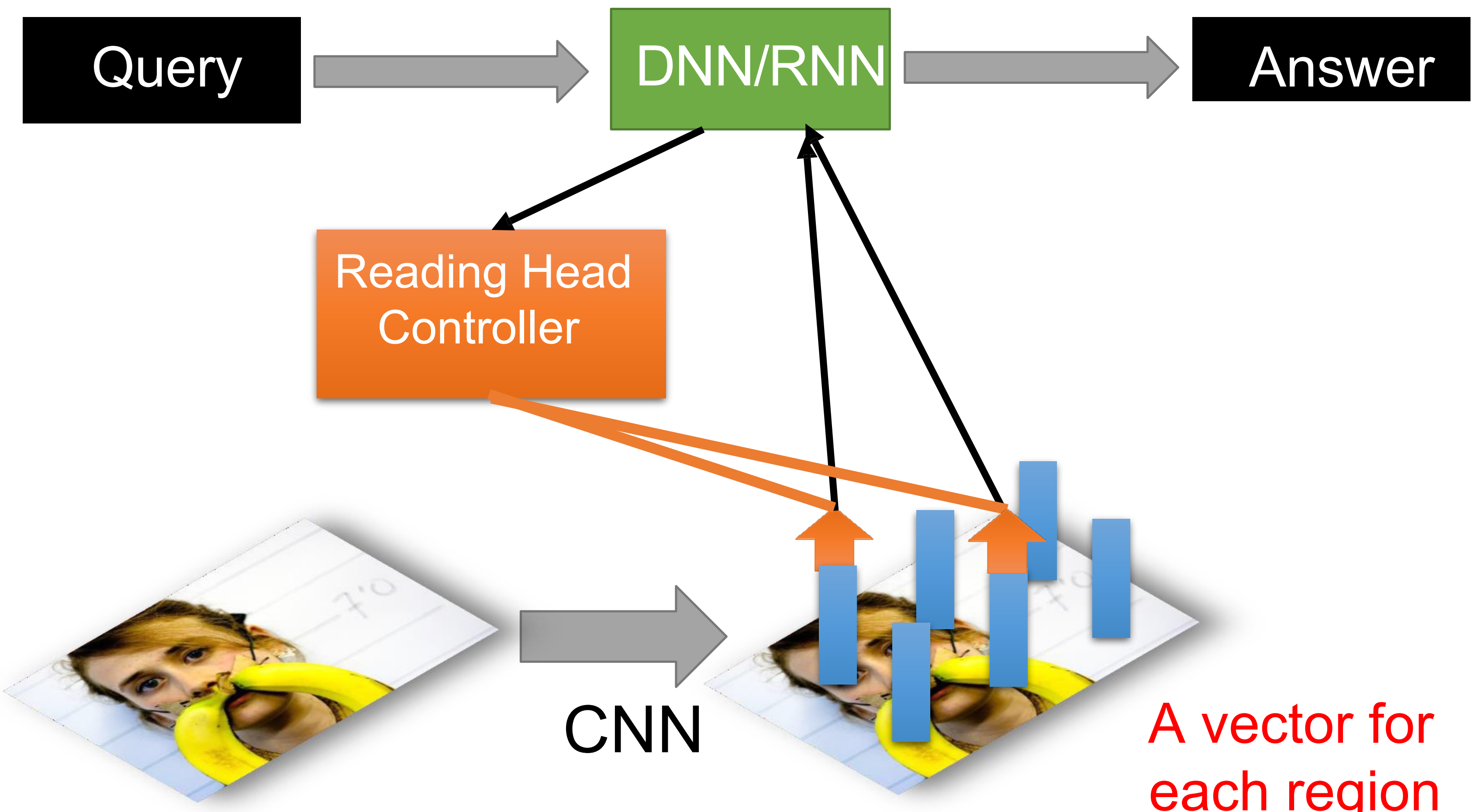
Story (16: basic induction)	**Support**	**Hop 1**	**Hop 2**	**Hop 3**
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

Visual Question Answering



source: http://visualqa.org/

Visual Question Answering



To Learn More ……

- The Unreasonable Effectiveness of Recurrent Neural Networks
  - http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Understanding LSTM Networks
  - http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Acknowledgement

Reference and thanks to:

- **National Taiwan University ML2020 Course:**

Machine Learning
https://speech.ee.ntu.edu.tw/~hylee/ml/2020-spring.php

- **Understanding LSTM Networks**

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- **LSTM Wikipedia**

https://en.wikipedia.org/wiki/Long_short-term_memory