RECURRENT NEURAL NETWORK

Narges Norouzi

VARIANTS OF NEURAL NETWORKS

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

MOTIVATION

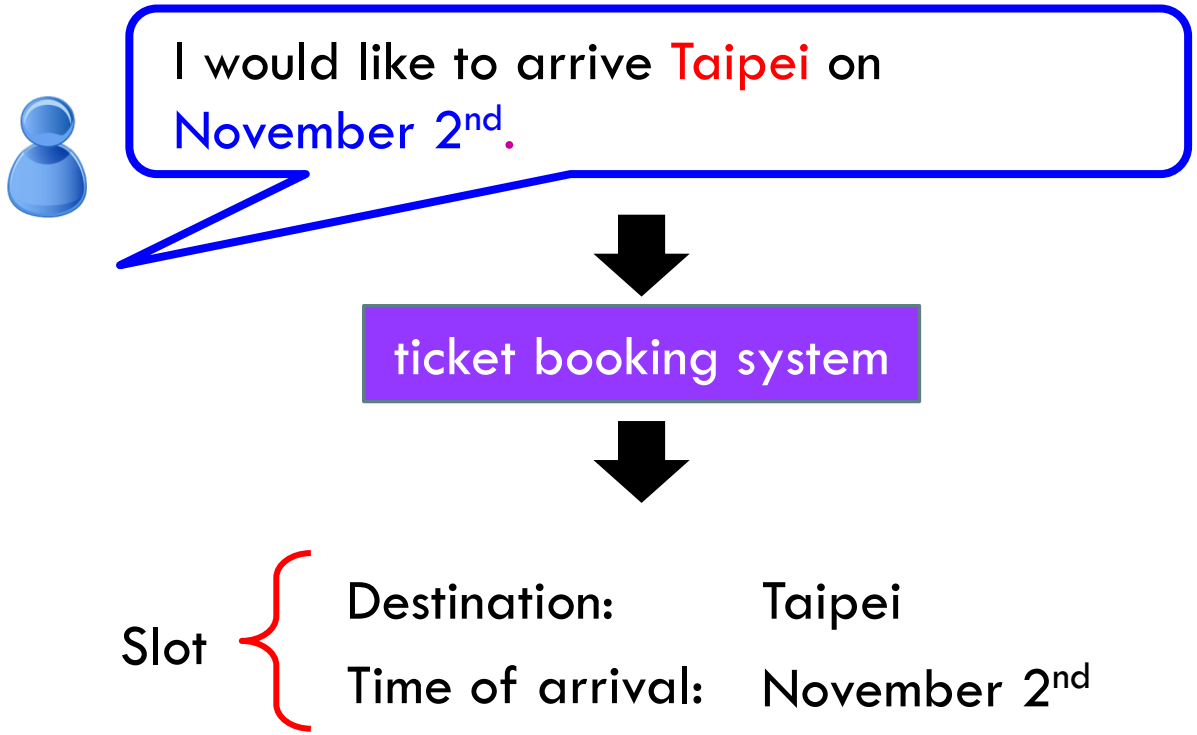
- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output

- Fixed amount of computational steps

- Recurrent networks allow us to operate over *sequences* of vectors

EXAMPLE APPLICATION

- Slot Filling

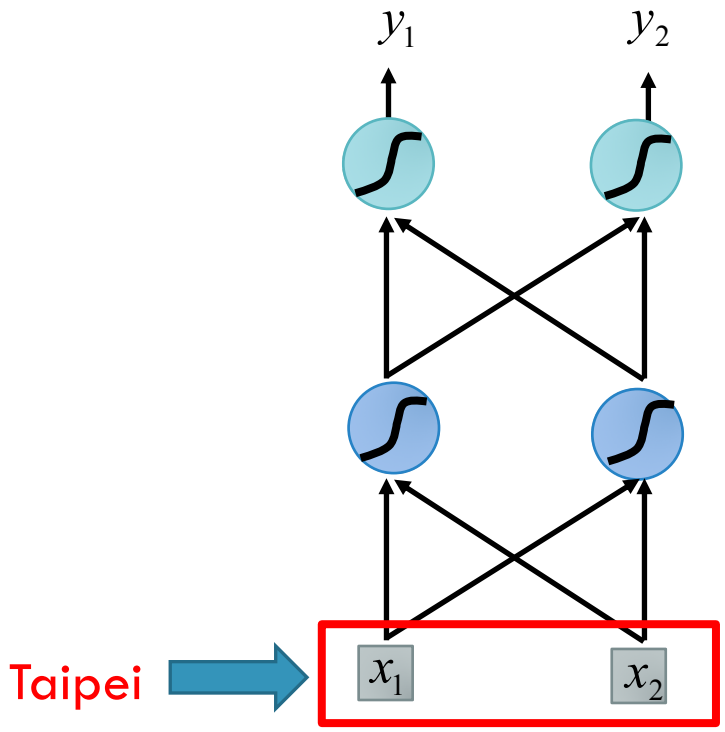


EXAMPLE APPLICATION

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]

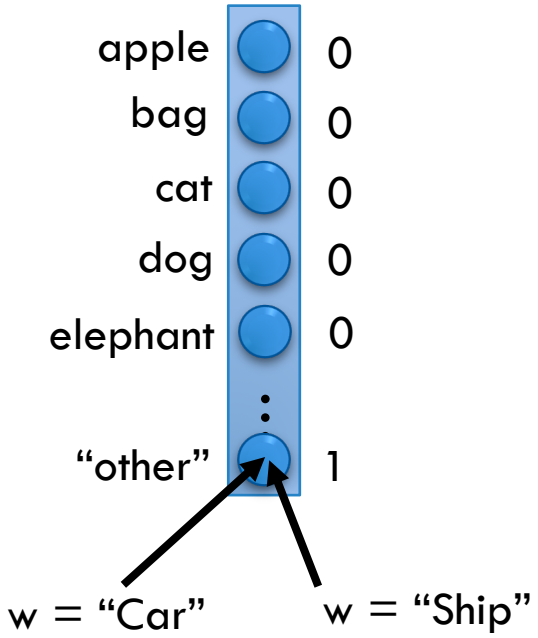
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

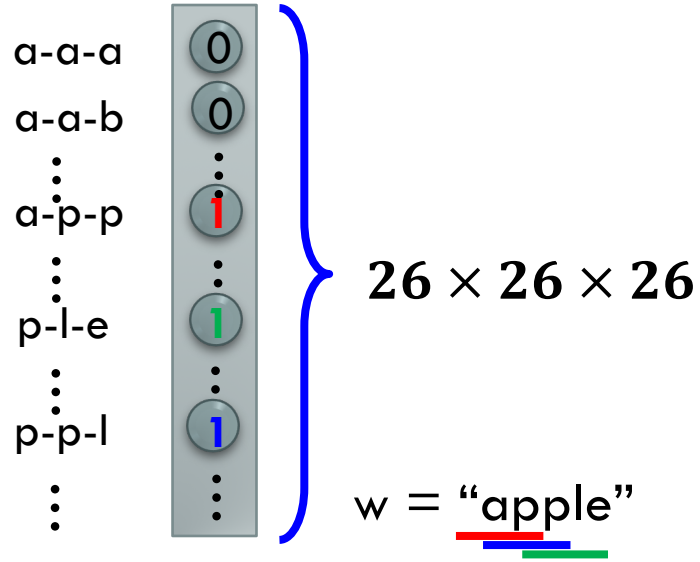
elephant = [0 0 0 0 1]

BEYOND 1-OF-N ENCODING

Dimension for "Other"



Word hashing

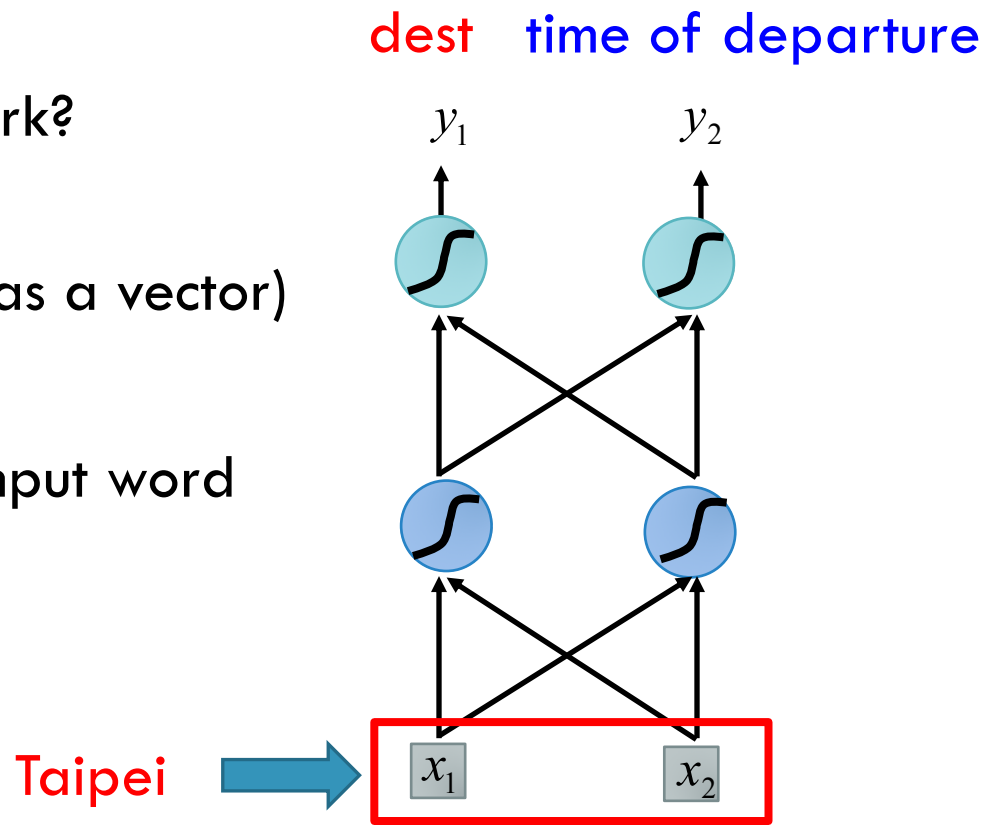


EXAMPLE APPLICATION

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



EXAMPLE APPLICATION

arrive Taipei on November 2nd

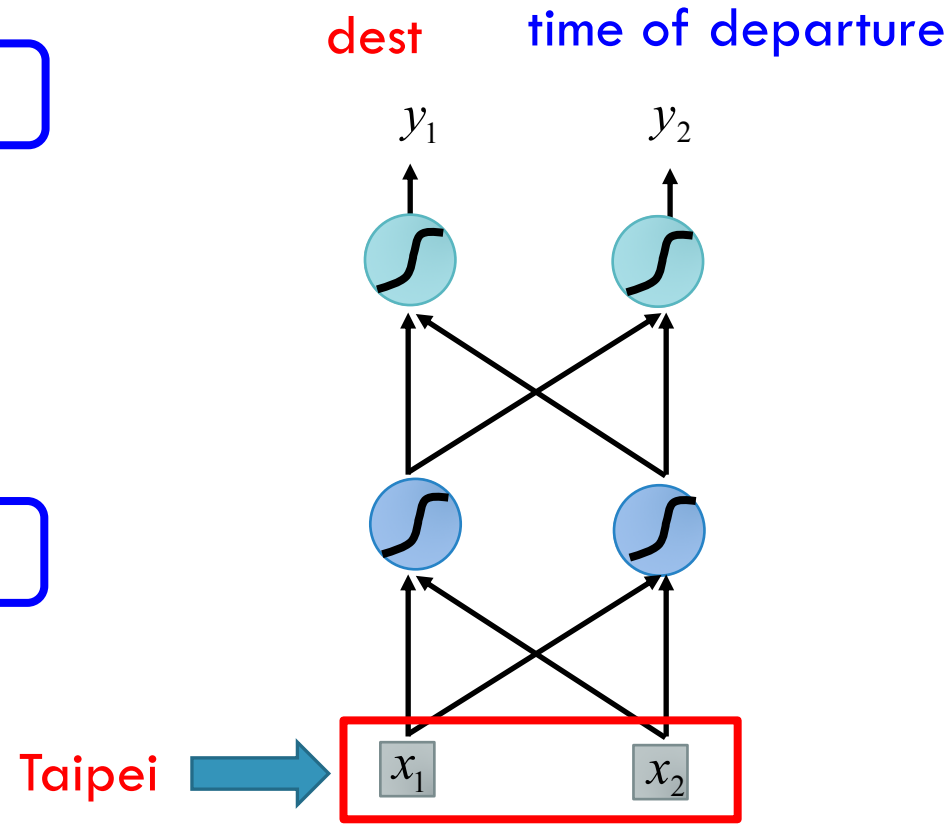
other dest other time time

Problem?

leave Taipei on November 2nd

place of departure

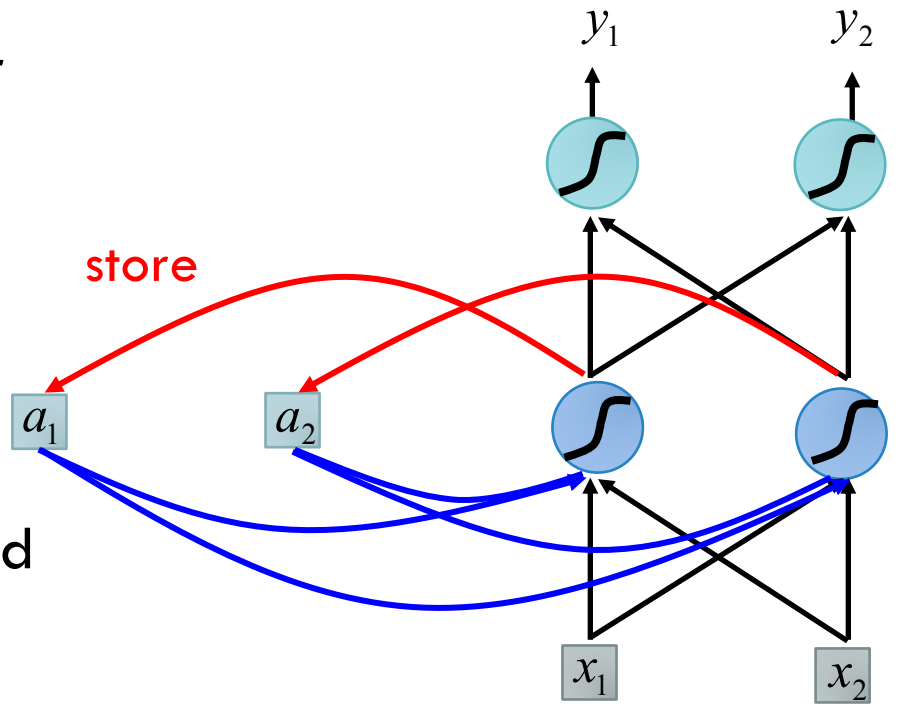
Neural network needs memory!

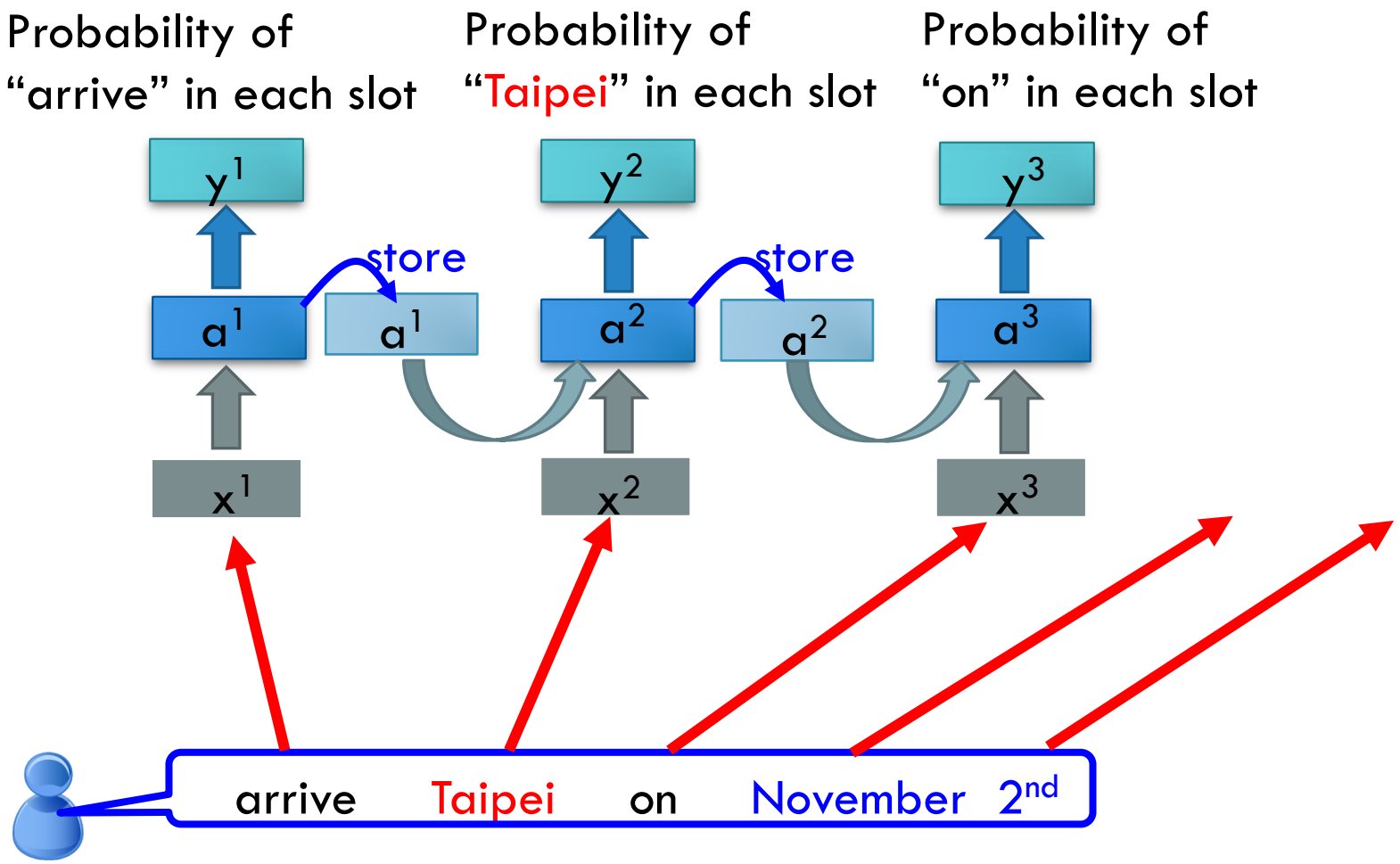


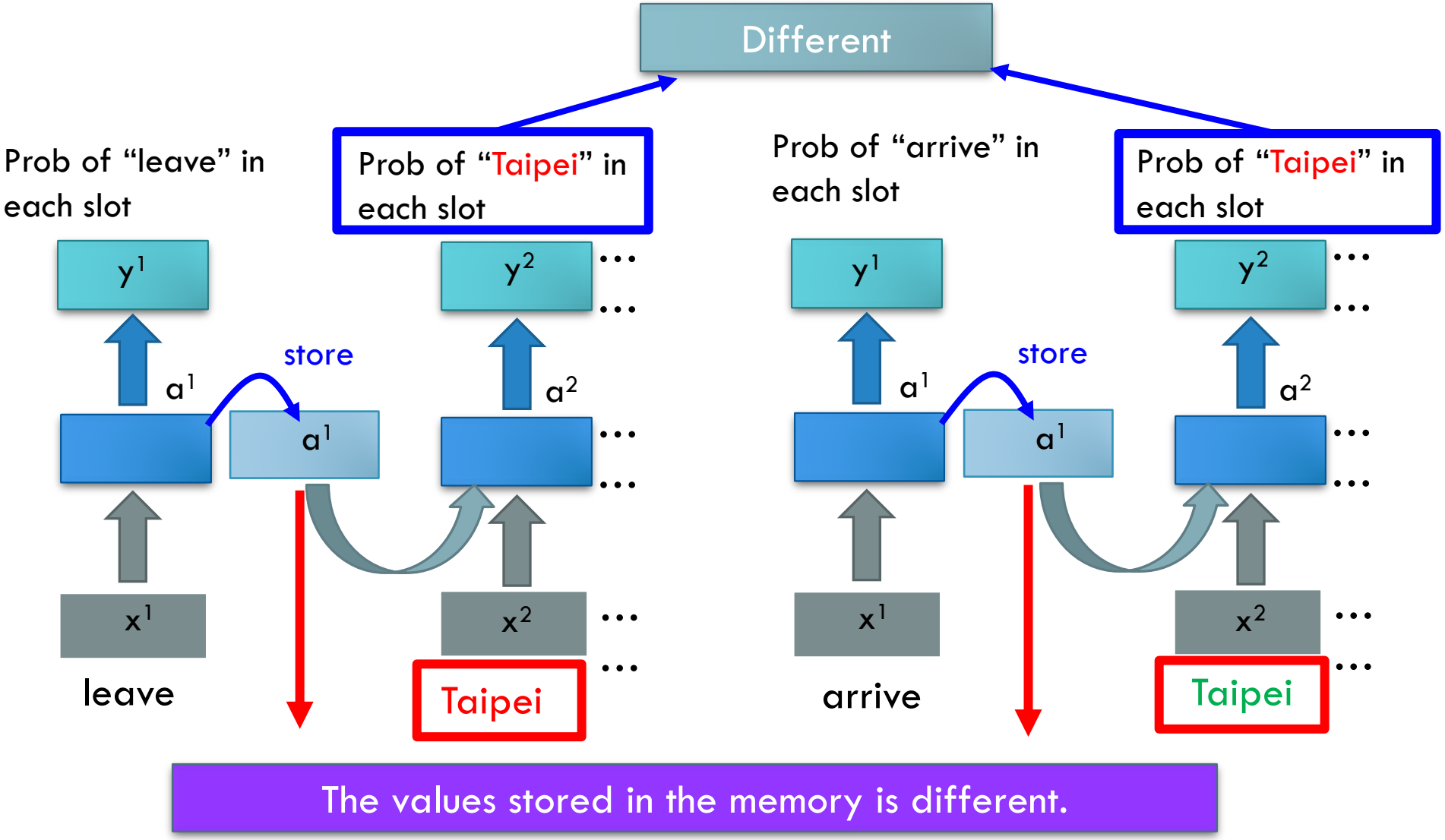
RECURRENT NEURAL NETWORK (RNN)

The output of hidden layer are stored in the memory.

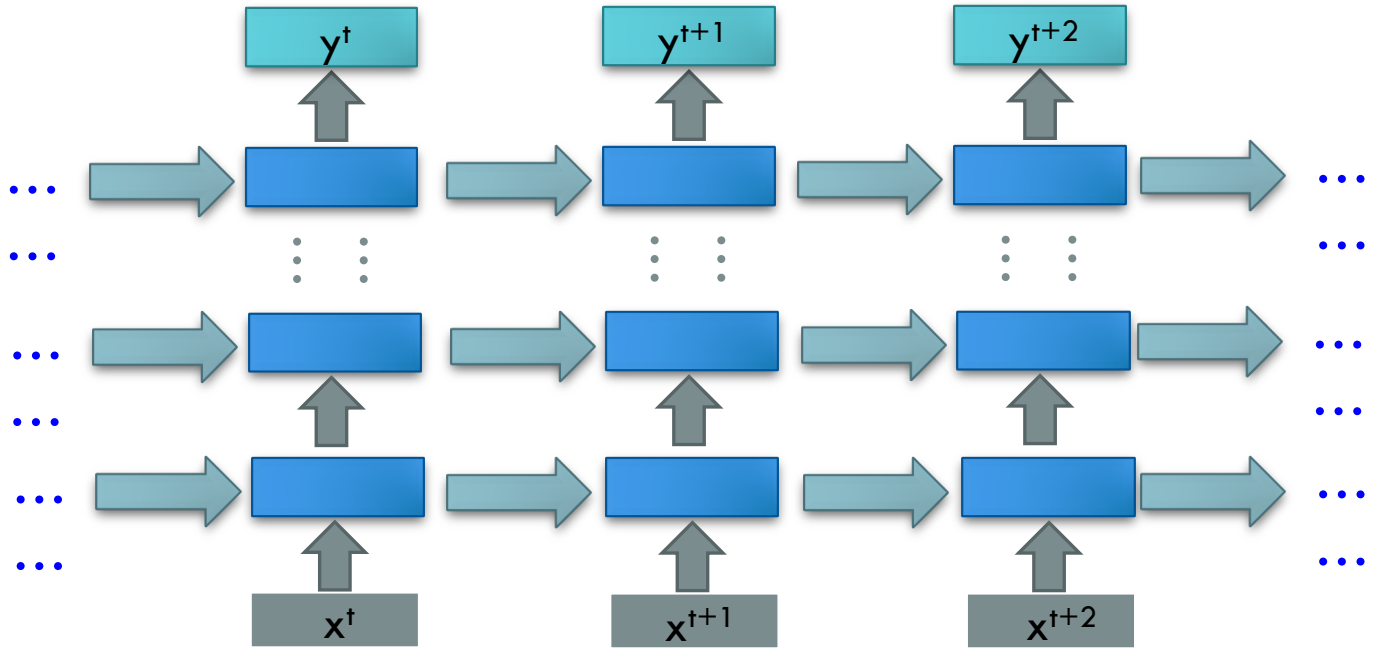
Memory can be considered as another input.



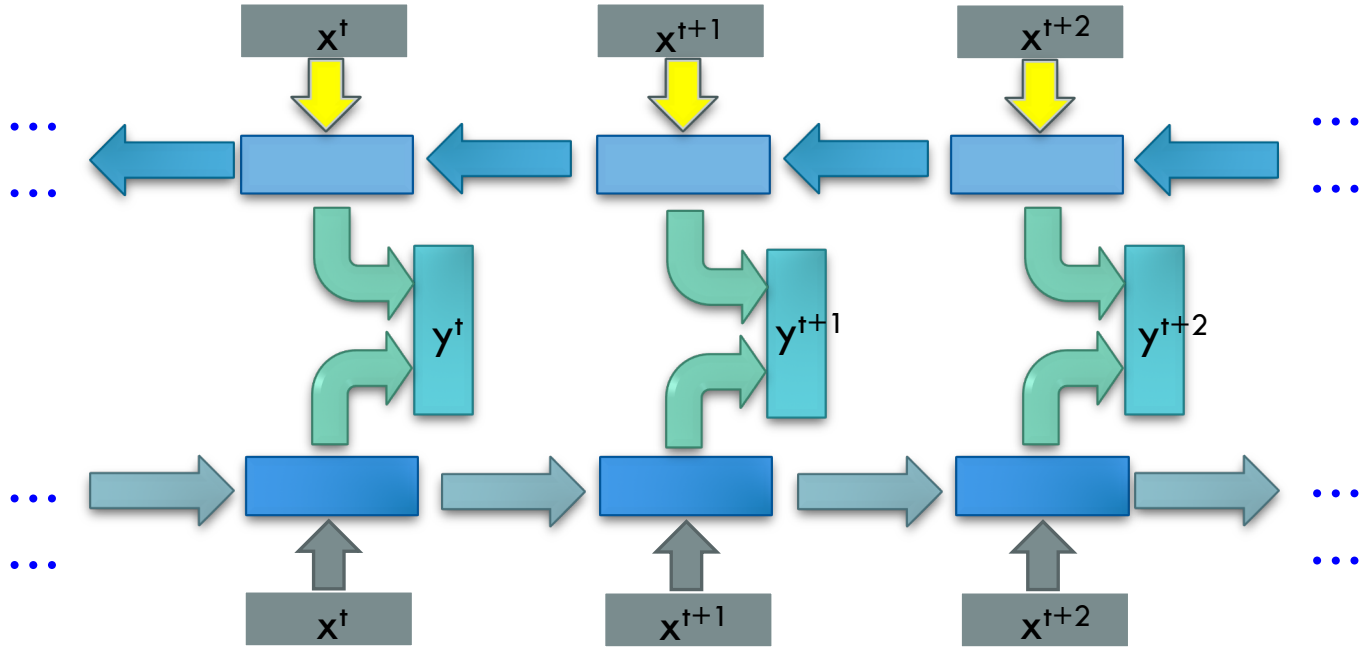

Probability of "arrive" in each slot
Probability of "Taipei" in each slot
Probability of "on" in each slot
y1
y2
y3
store
store
a1
a1
a2
a2
a3
x1
x2
x3
arrive
Taipei
on
November 2nd


Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
store
store
leave
arrive
Taipei
Taipei
The values stored in the memory is different.

OF COURSE IT CAN BE DEEP . . .

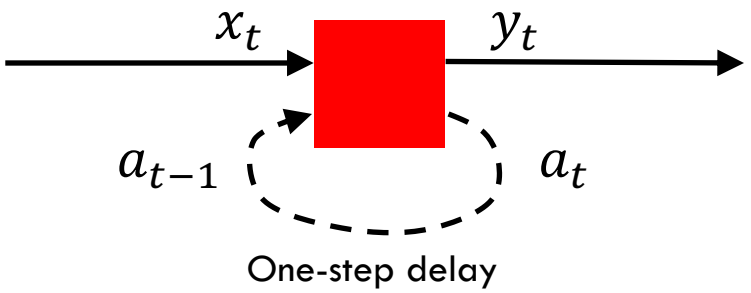


BIDIRECTIONAL RNN



RECURRENT NEURAL NETWORKS (RNN)

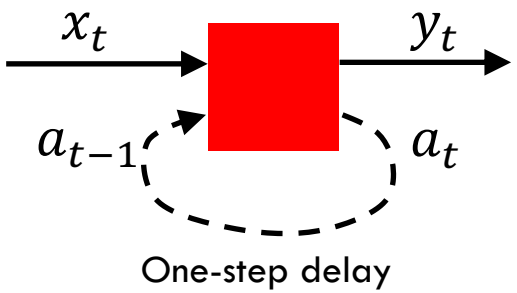
- Recurrent networks introduce the notion of cycle and time.

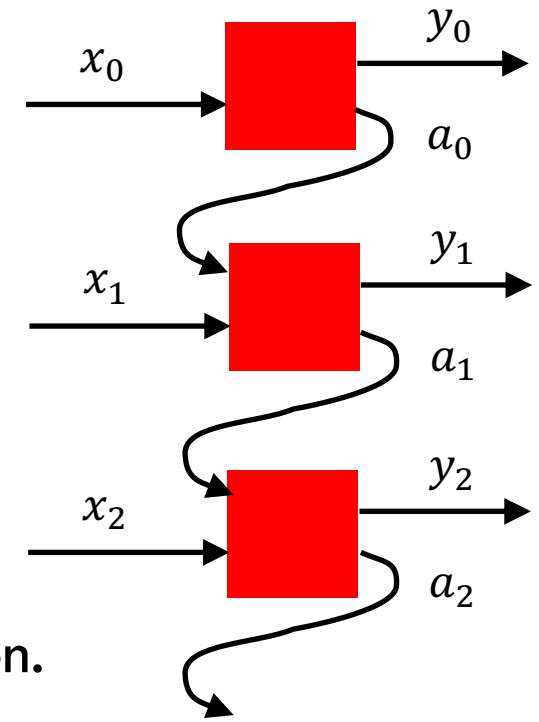


- They are designed to process sequences of data $x_1, \ldots, x_n$ and can produce sequences of outputs $y_1, \ldots, y_m$.

UNROLLING RNN

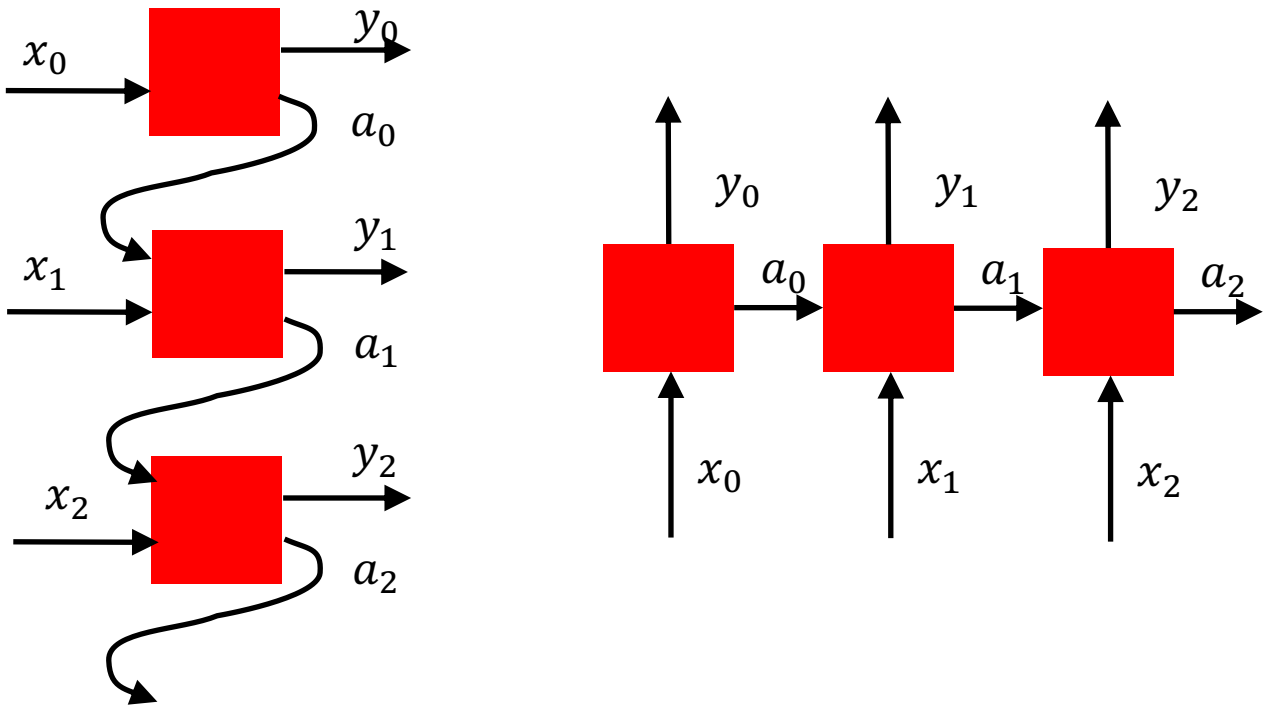
- RNNs can be unrolled across multiple time steps



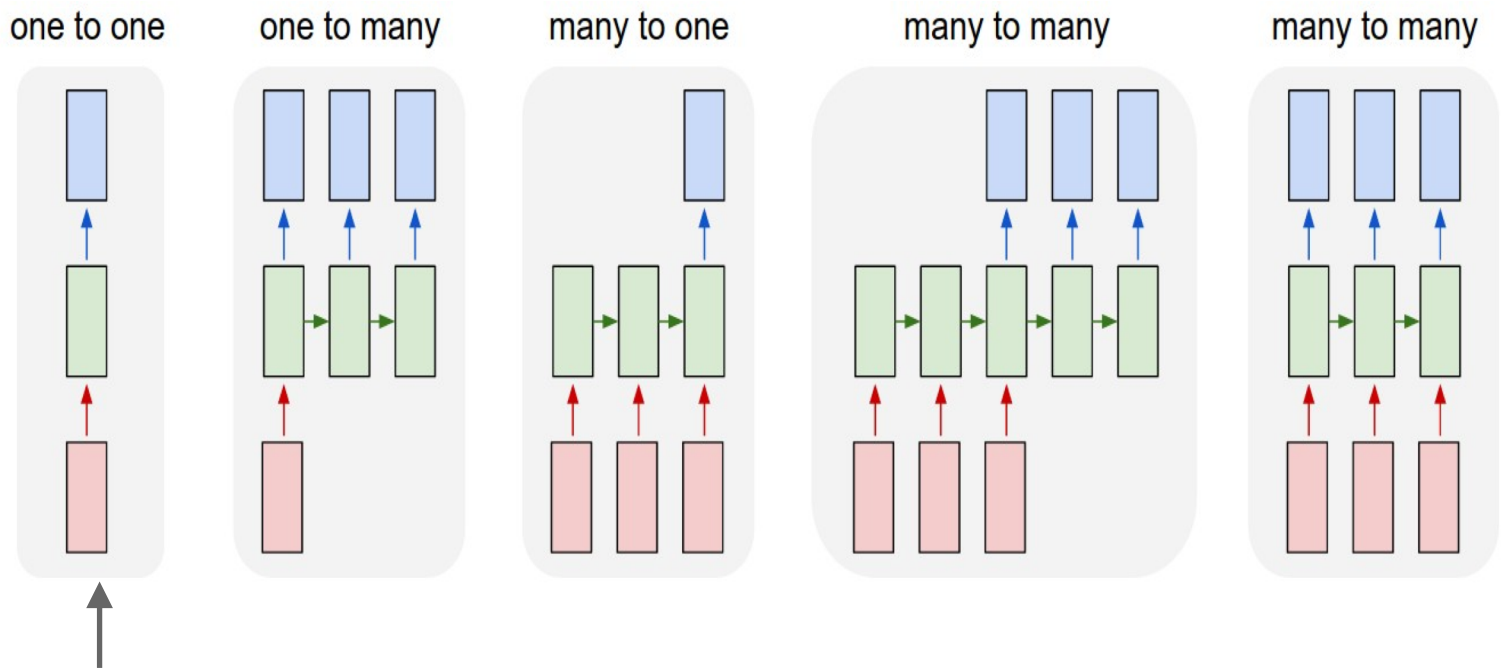


- This produces a DAG which supports backpropagation.
- But its size depends on the input sequence length.

UNROLLING RNNS

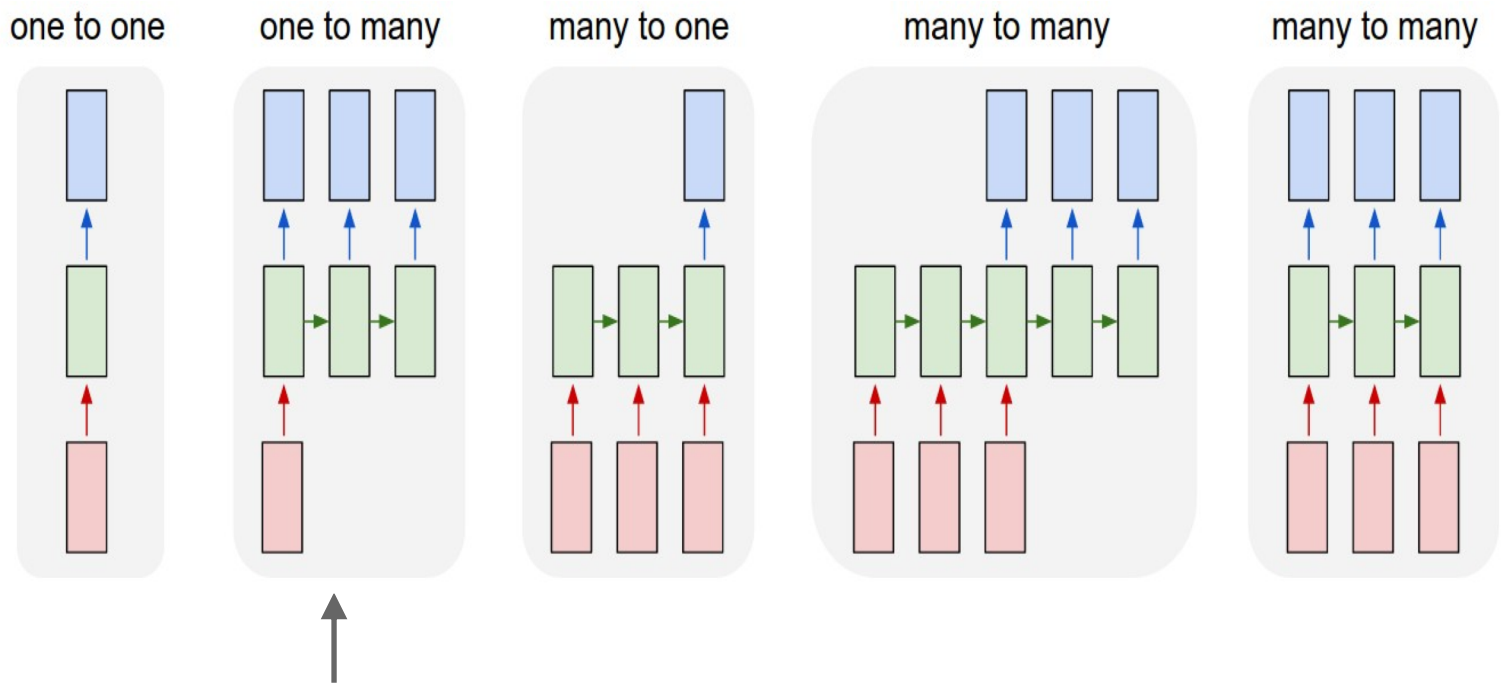


TYPES OF RNN



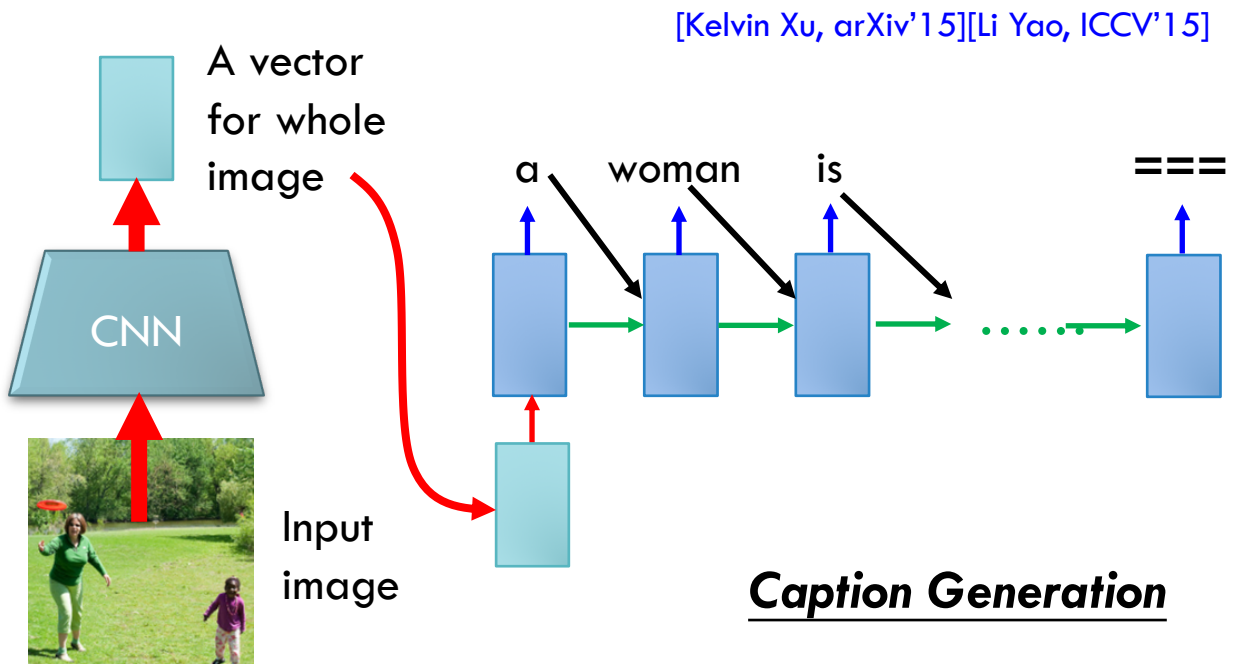
Vanilla Neural Networks

TYPES OF RNN

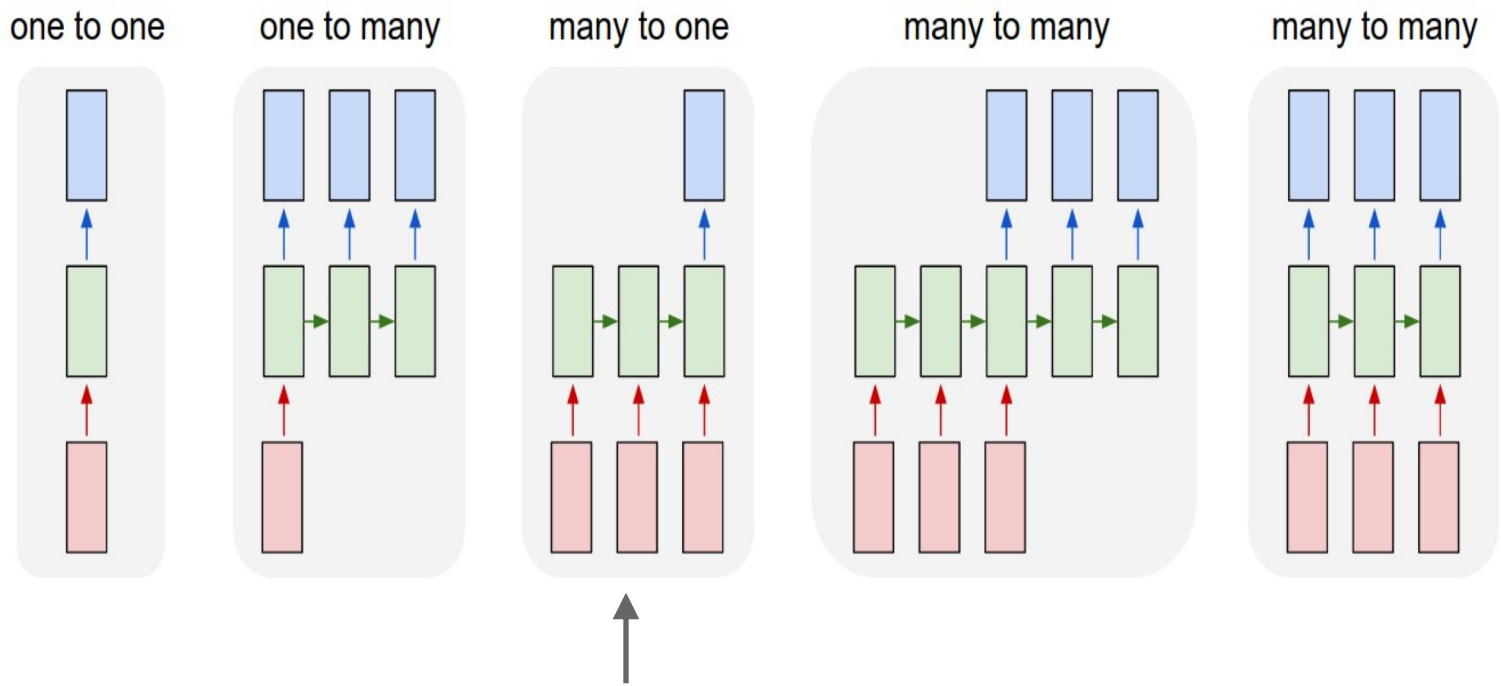


e.g. **Image Captioning**
image → sequence of words

IMAGE CAPTION GENERATION

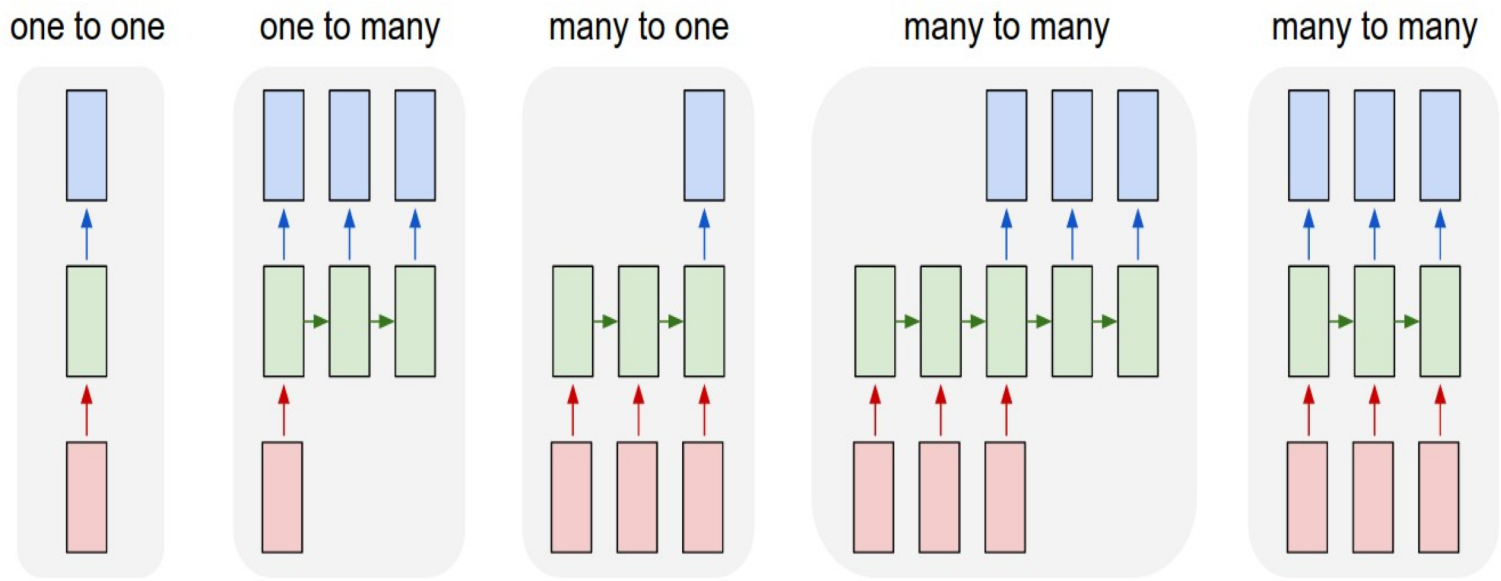


TYPES OF RNN



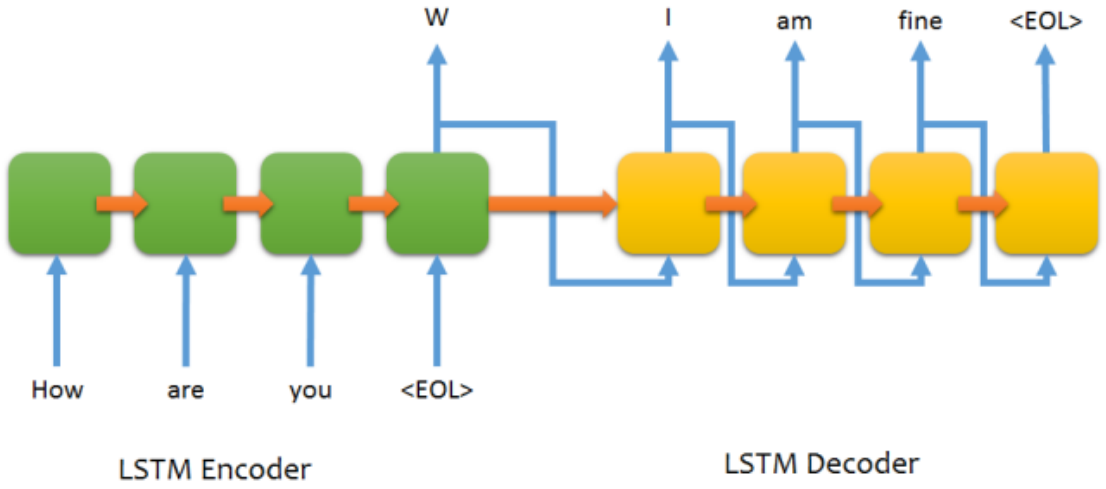
e.g. **Sentiment Classification**
sequence of words → sentiment

TYPES OF RNN

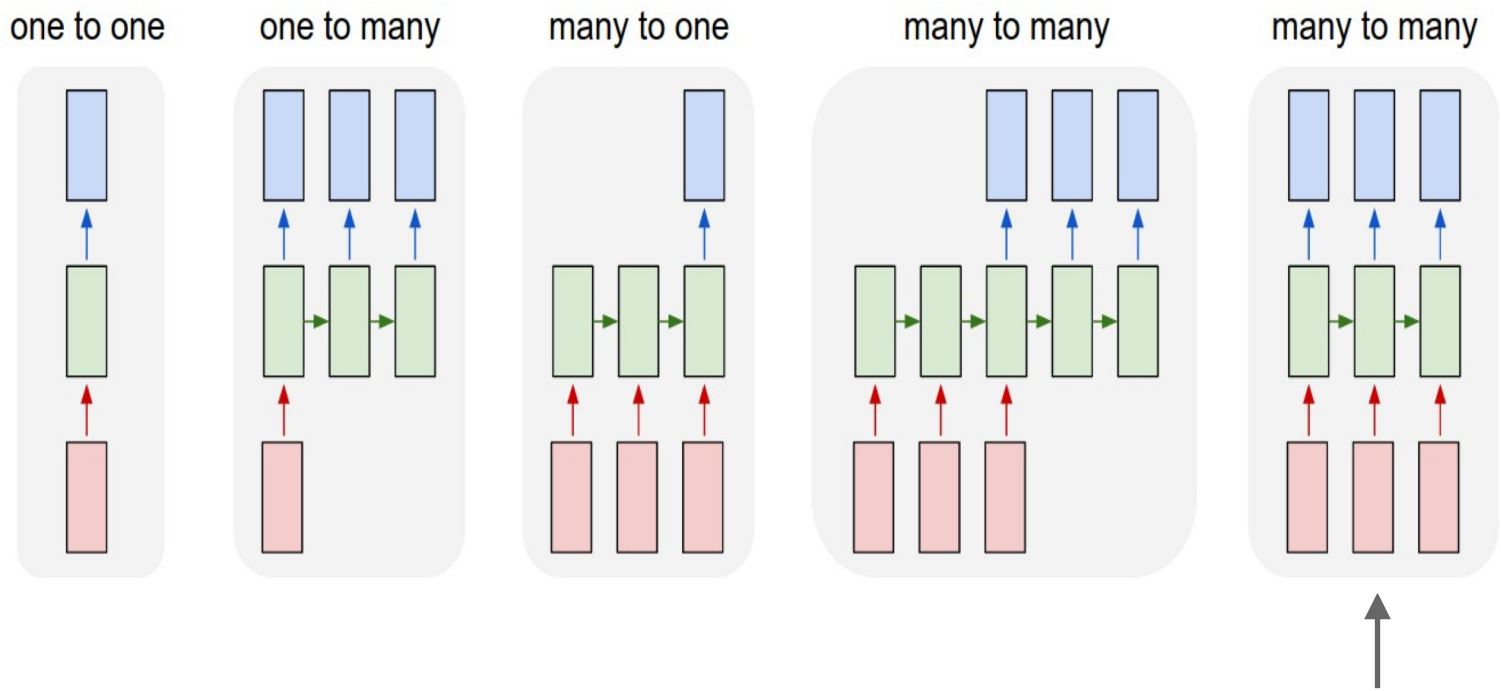


e.g. **Machine Translation**
seq of words → seq of words

CHAT-BOT

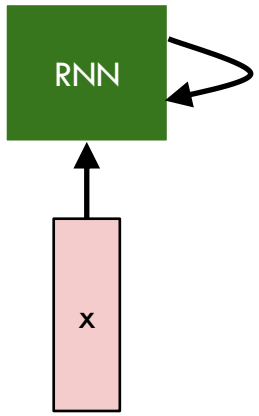


TYPES OF RNN

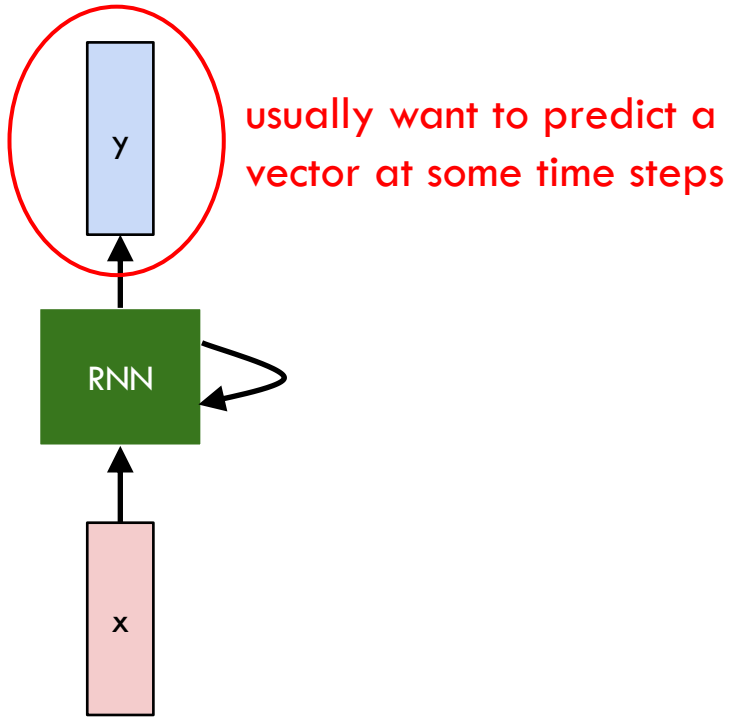


e.g. **Video classification on frame level**

RNN STRUCTURE



RNN STRUCTURE



RNN STRUCTURE

- We can process a sequence of vectors x by applying a recurrence formula at every time step:

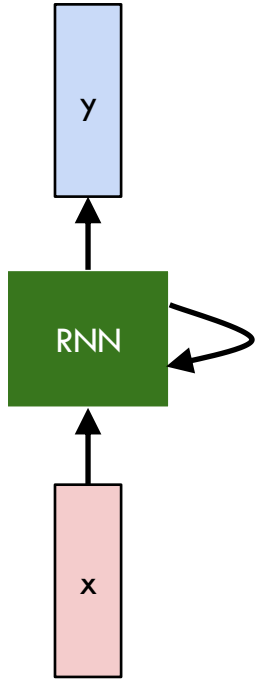
input vector at some time step

$$a_t = g_w(a_{t-1}, x_t)$$

new state

old state

some function with parameters W



RNN STRUCTURE

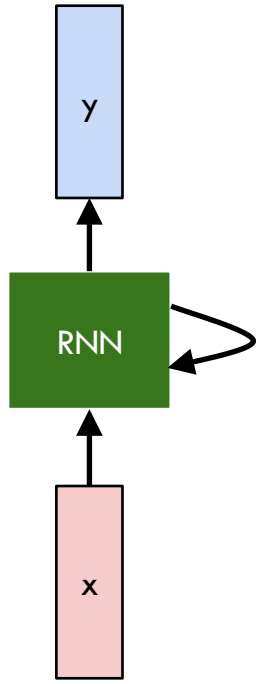
new state

input vector at some time step

$$a_t = g_W(a_{t-1}, x_t)$$

old state

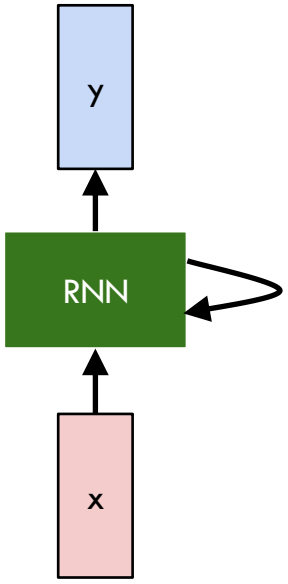
some function with parameters W



Note that the same function and the same set of parameters are used at every time step.

VANILLA RECURRENT NEURAL NETWORK

The state consists of a single *"hidden"* vector **h**:



$$a_t = g_w(a_{t-1}, x_t)$$

$$a_t = tanh(W_{aa}a_{t-1} + W_{xa}x_t) = \tanh(W_a[a_{t-1}, x_t])$$

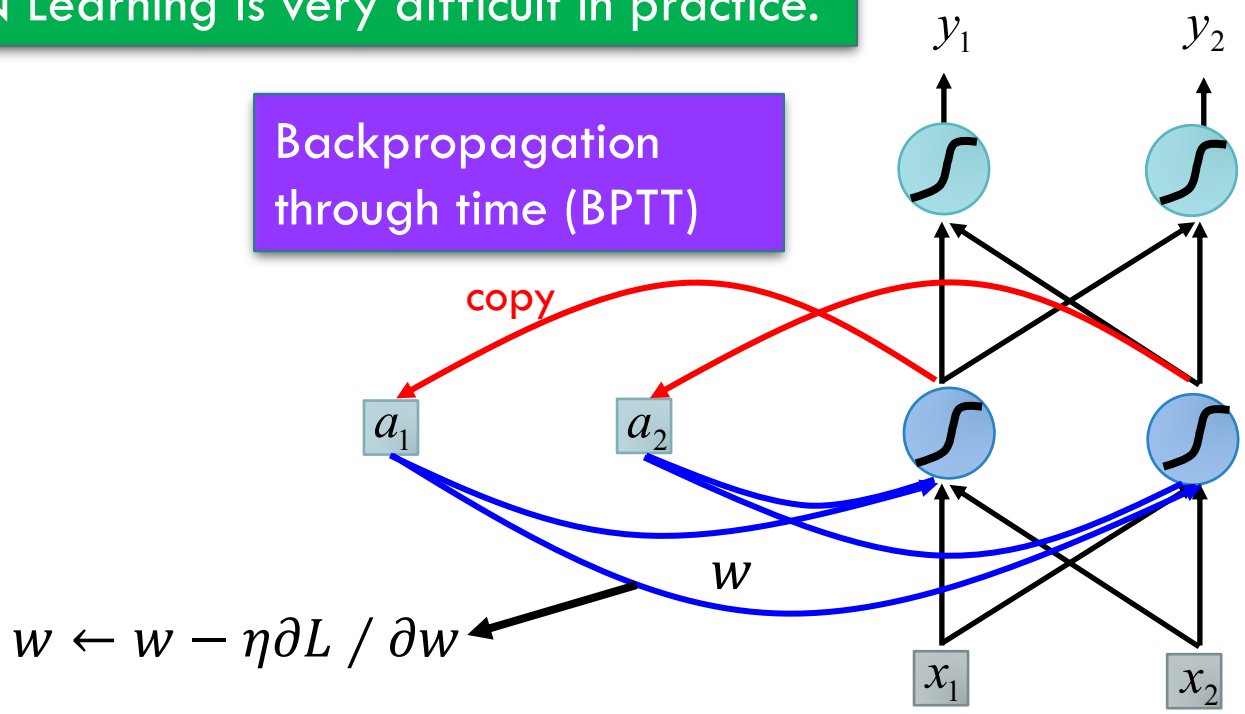
$$y_t = W_{ay}a_t$$

$$\tanh(x) = 2sigmoid(2x) - 1$$

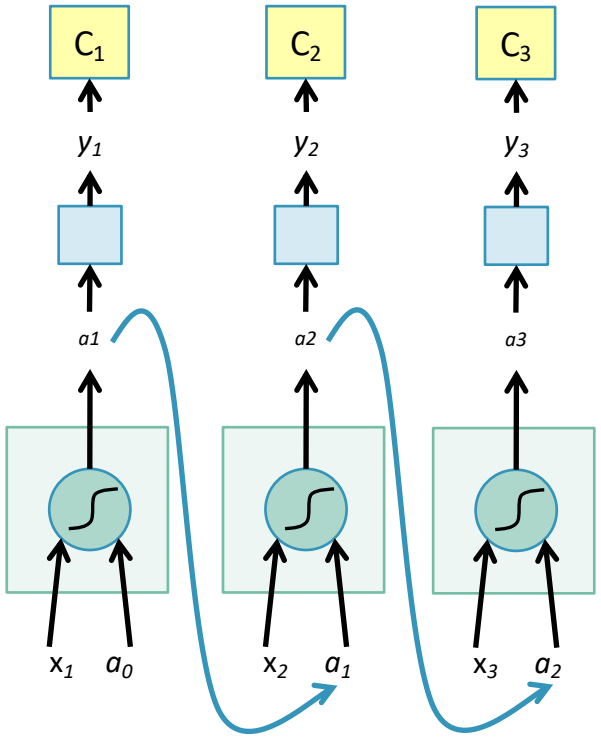
LEARNING

RNN Learning is very difficult in practice.

Backpropagation through time (BPTT)



THE VANILLA RNN FORWARD

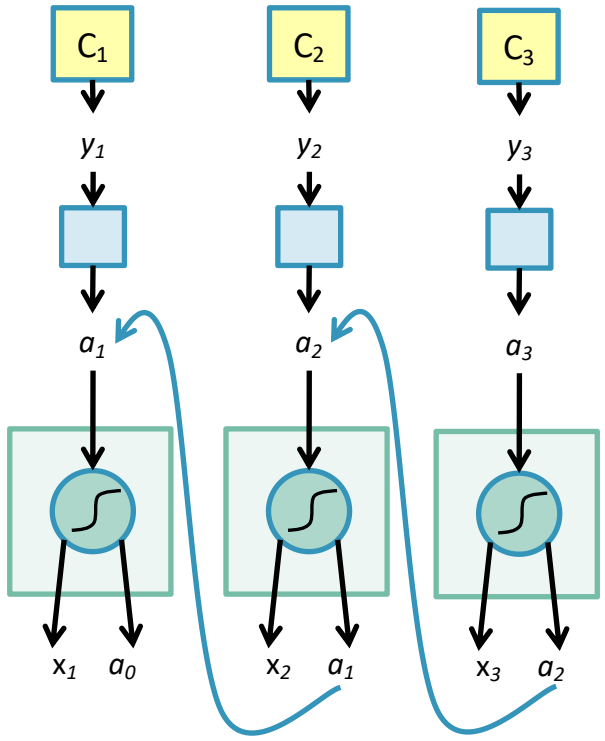


$$a_t = \tanh(W_a[a_{t-1}, x_t])$$

$$y_t = W_{ay} a_t$$

$$C_t = Loss(y_t, T_t)$$

THE VANILLA RNN BACKWARD



$$a_t = \tanh(W_a[a_{t-1}, x_t])$$

$$y_t = W_{ay} a_t$$

$$C_t = Loss(y_t, T_t)$$

$$\frac{\partial C_t}{\partial a_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial a_1}\right) = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial a_t}\right)\left(\frac{\partial a_t}{\partial a_{t-1}}\right)\ldots\left(\frac{\partial a_2}{\partial a_1}\right)$$

PROBLEMS WITH NAIVE RNN

- When dealing with a time series, it tends to forget old information. When there is a distant relationship of unknown length, we wish to have a “memory” to it.
- Vanishing and exploding gradient problem.

SOLUTION TO EXPLODING GRADIENT

- Truncated BPTT
- Clip gradient at a threshold
  - L2 Norm clipping: $gradients_{new} = \frac{gradients \times threshold}{l2_{norm(gradients)}}$
- RMSprop to adjust learning rate
  - RMSprop combines the idea of only using the sign of the gradient with the idea of adapting the step size individually for each weight

LSTM INTRODUCTION

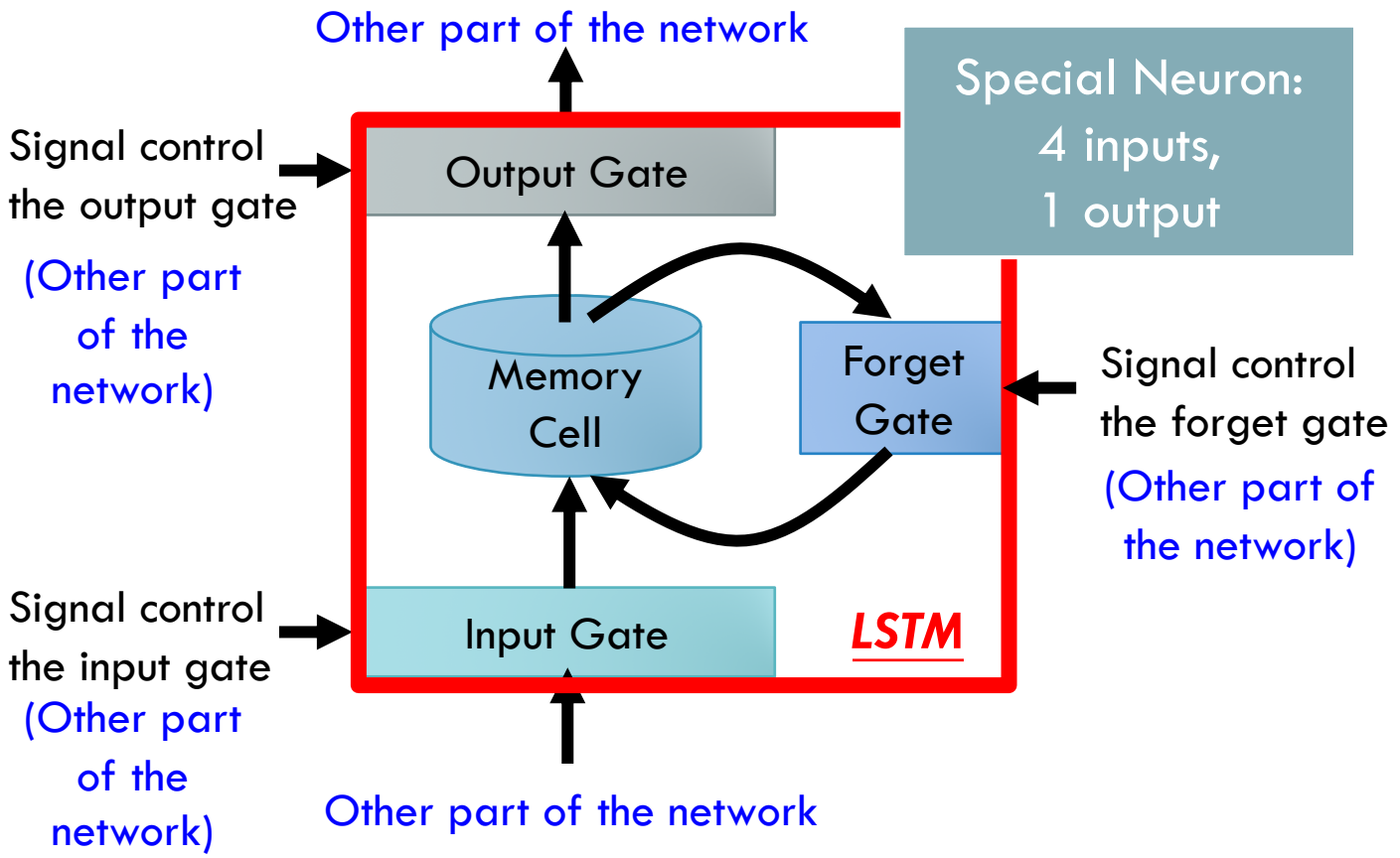
LSTM INTRODUCTION

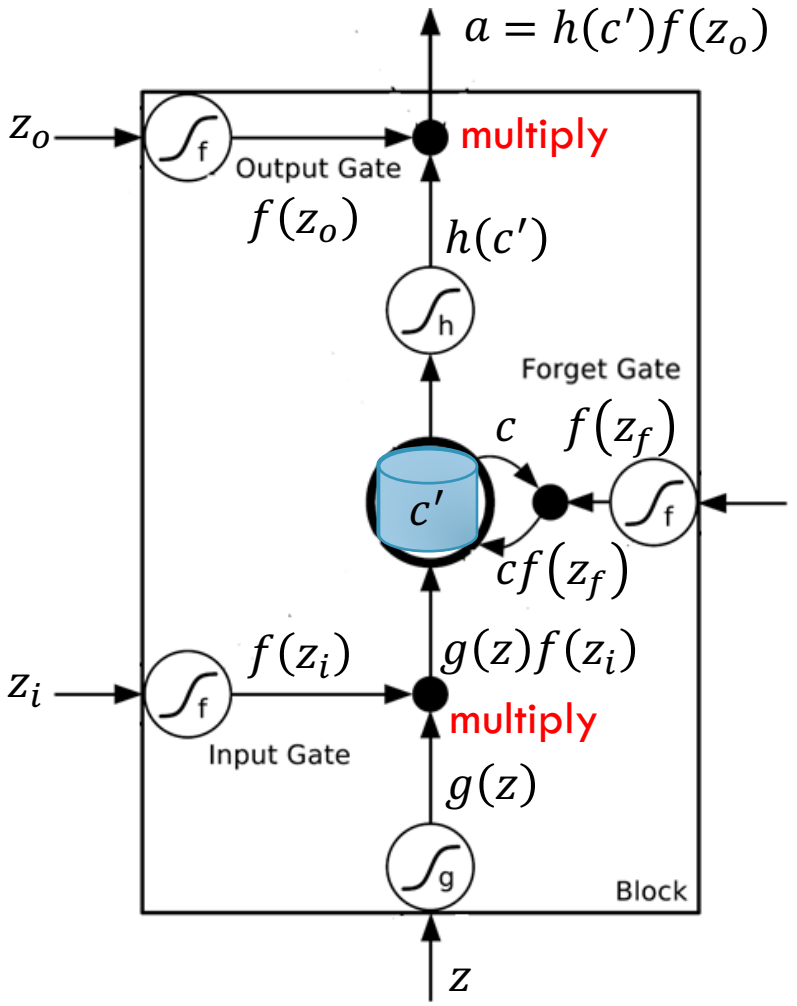
- LSTM was invented to solve the vanishing gradients problem.
- LSTM maintain a more constant error flow in the backpropogation process.
- Similar to manipulating a memory cell:
  - Forget (flush the memory)
  - Input (add to memory)
  - Output (read from memory)

MEMORY CELL IN LSTM

- LSTM networks introduce a new structure called a memory cell.
- Each memory cell contains four main elements:
  - Input gate
  - Forget gate
  - Output gate
  - Neuron with a self-recurrent
- These gates allow the cells to keep and access information over long periods of time.

LONG SHORT-TERM MEMORY (LSTM)



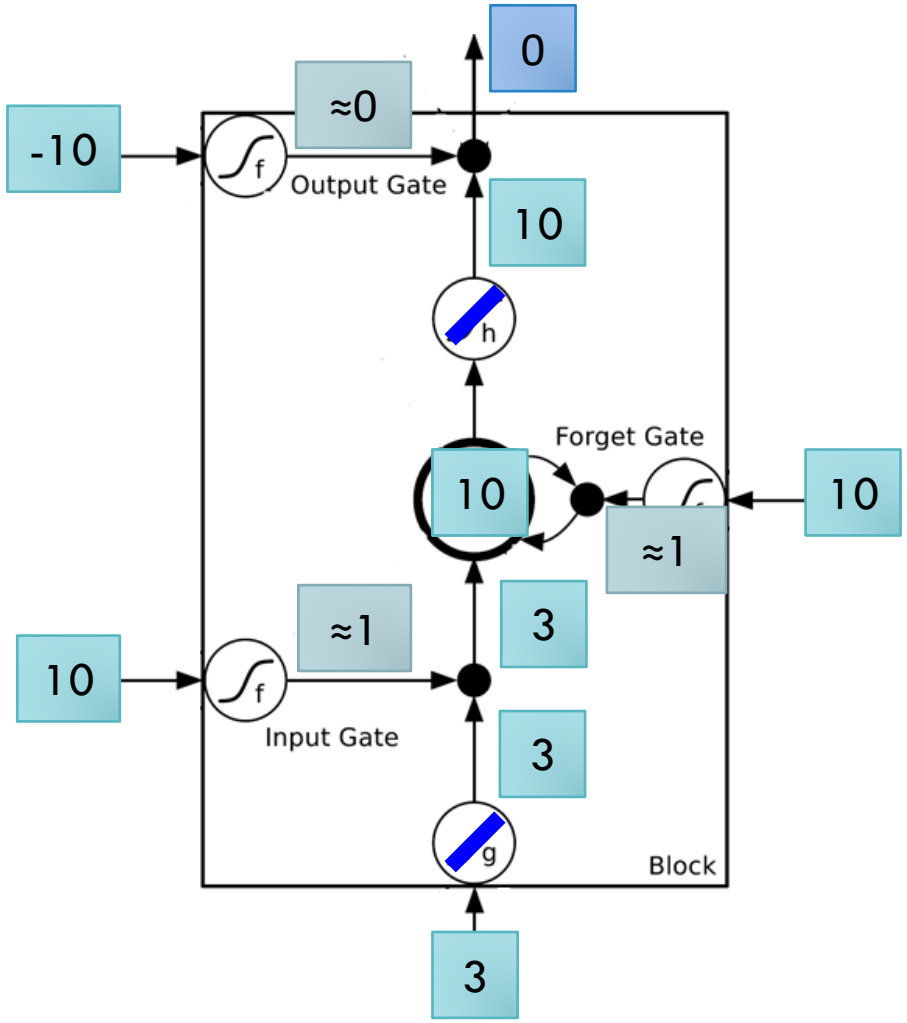


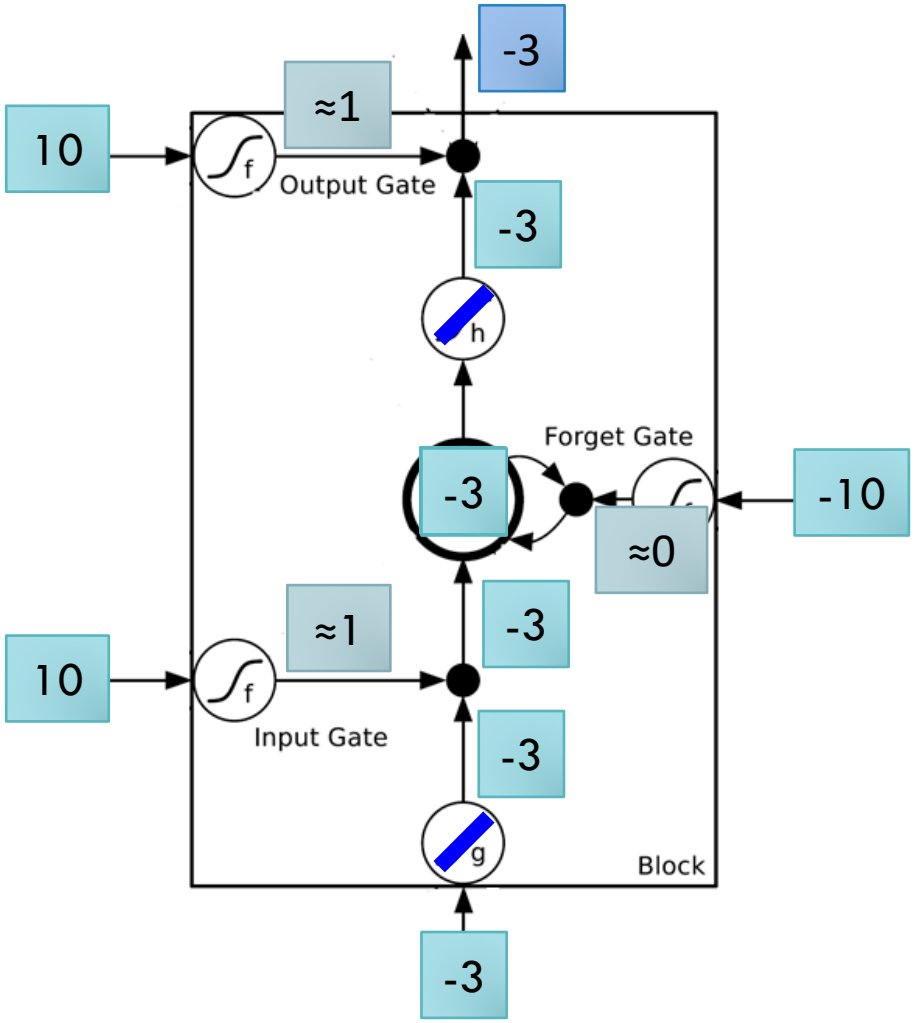
Activation function f is usually a sigmoid function

Between 0 and 1

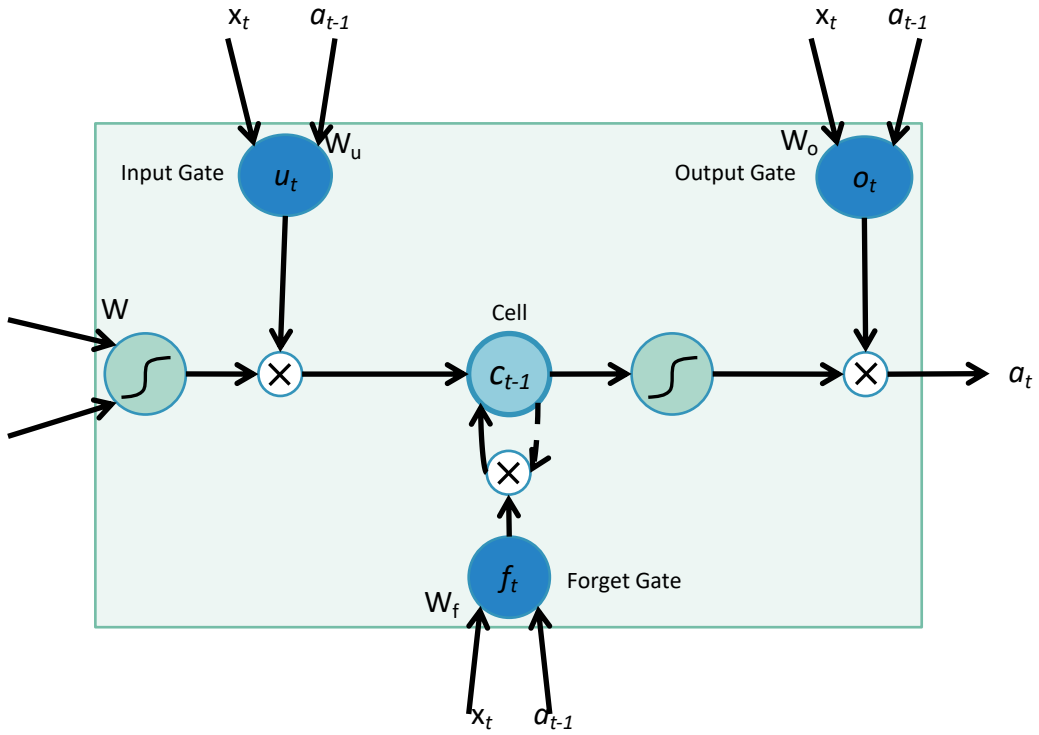
Mimic open and close gate

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


0
≈0
-10
f
Output Gate
10
h
Forget Gate
10
10
≈1
≈1
3
10
f
Input Gate
3
g
Block
3


-3
≈1
10
f
Output Gate
-3
h
Forget Gate
-3
-10
≈0
≈1
-3
10
f
Input Gate
-3
g
Block
-3

LSTM CELL EQUATIONS



$$\tilde{C}_t = \tanh(W_c[a_{t-1}, x_t])$$

$$u_t = \sigma(W_u[a_{t-1}, x_t])$$

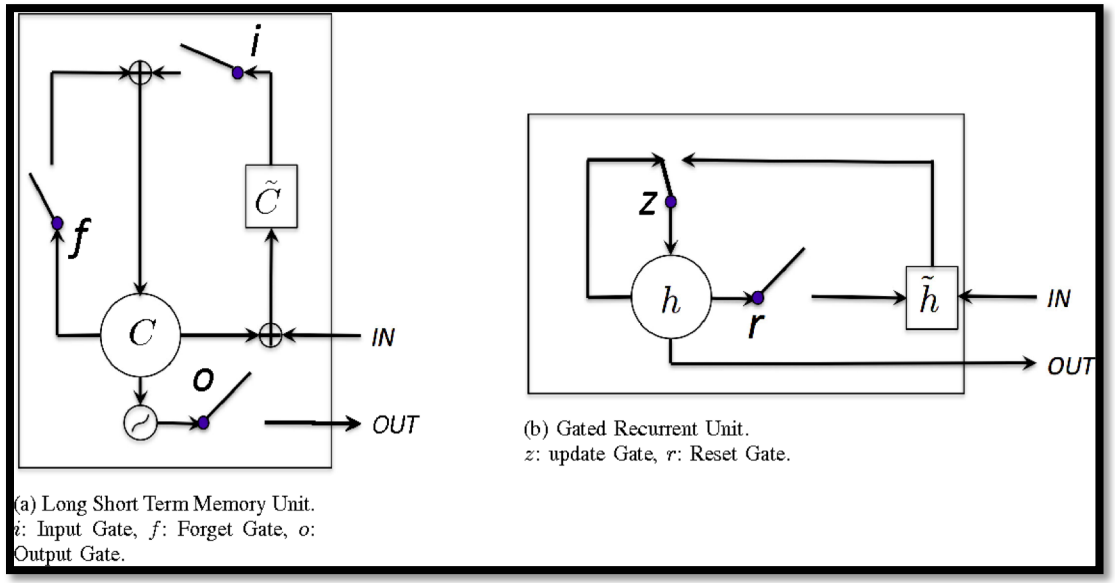
$$f_t = \sigma(W_f[a_{t-1}, x_t])$$

$$o_t = \sigma(W_o[a_{t-1}, x_t])$$

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

$$a_t = o_t . \mathrm{C_t}$$

GATED RECURRENT UNIT



(a) Long Short Term Memory Unit. i: Input Gate, f: Forget Gate, o: Output Gate.

(b) Gated Recurrent Unit. z: update Gate, r: Reset Gate.

$$\tilde{C}_t = \tanh(W_c[a_{t-1}, x_t])$$

$$u_t = \sigma(W_u[a_{t-1}, x_t])$$

$$C_t = u_t * \tilde{C}_t + (1 - u_t) * C_{t-1}$$

CLASS EXERCISE

bit.ly/ce-18