AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Introduction to Deep Learning

Sequential Models

Recurrent Neural Network (RNN)

Courtesy: Mitesh Kapra NPTEL, analytics vidya,fast.ai, coursera: AndrewNG

“working love learning we on deep”,

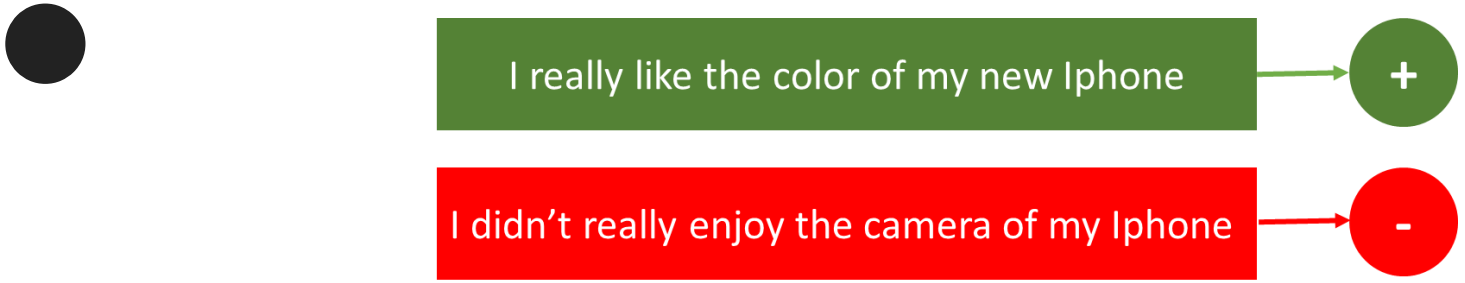
did this make any sense to you?

“We love working on deep learning”

Made perfect sense!

what are kind of tasks ?

- **Sentiment Classification** – This can be a task of simply classifying tweets into positive and negative sentiment. So here the **input would be a tweet of varying lengths**, while **output is of a fixed type** and **size.**



what are kind of tasks ?

•**Image Captioning** – Here, let's say we have an image for which we need a textual description. So we have a single input – the image, and a series or sequence of words as output. Here the image might be of a fixed size, but the output is a description of varying lengths

A person riding a motorcycle on a dirt road.

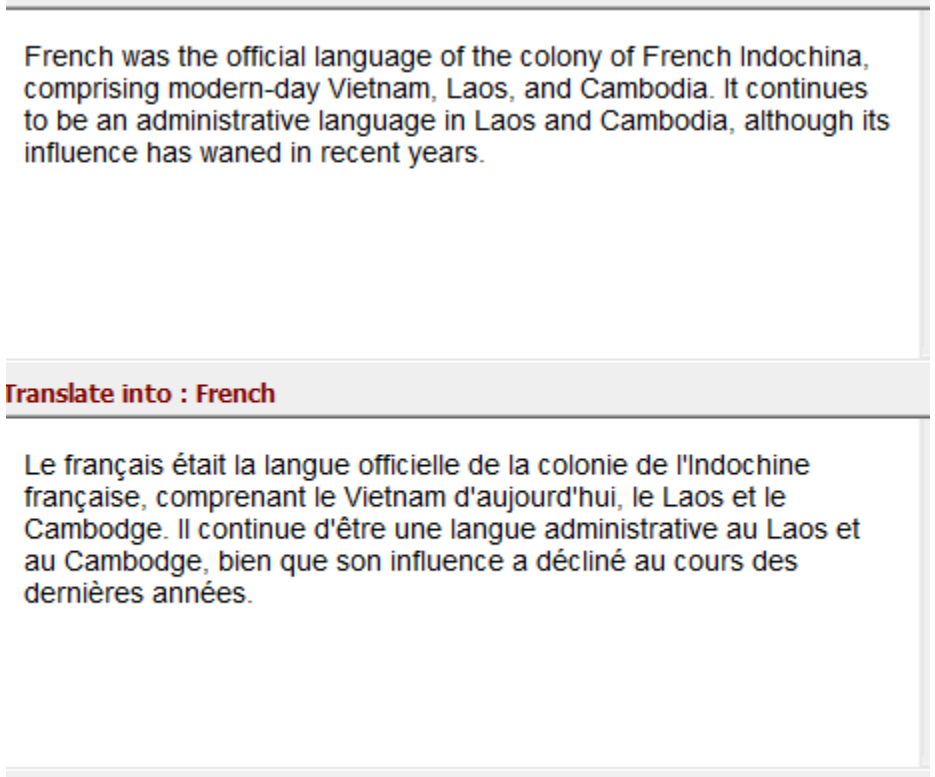
Two dogs play in the grass.

A group of young people playing a game of frisbee.

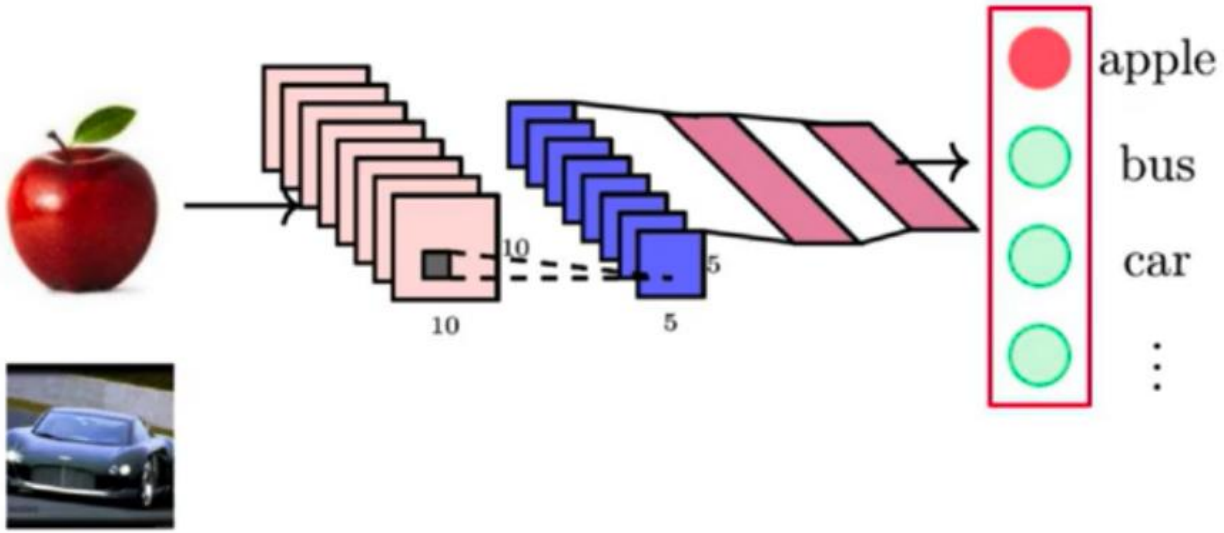
Two hockey players are fighting over the puck.

what are kind of tasks ?

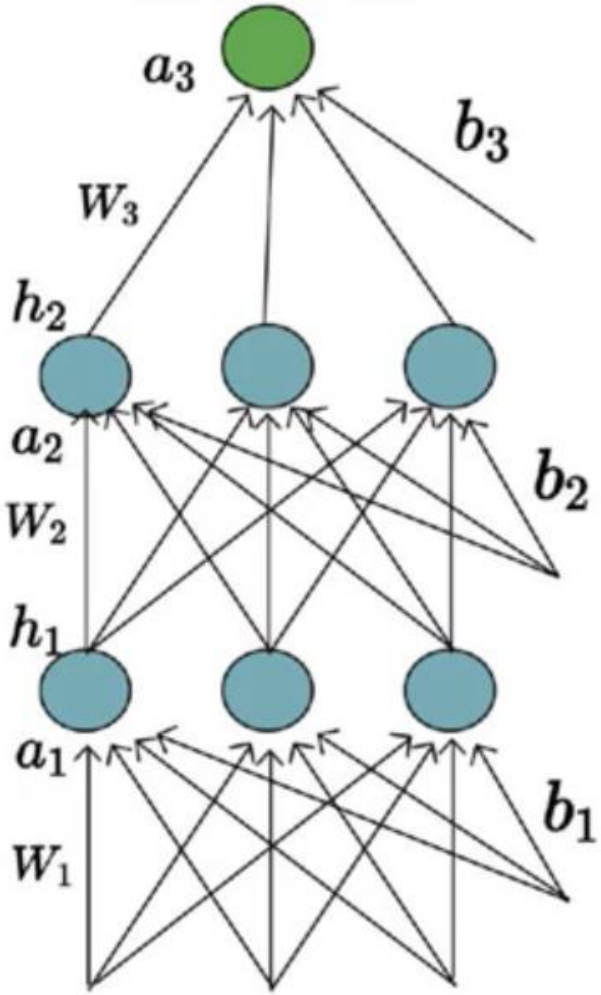
•**Language Translation** – This basically means that we have some text in a particular language let's say English, and we wish to translate it in French. Each language has it's own semantics and would have varying lengths for the same sentence. So here the inputs as well as outputs are of varying lengths.



Introduction



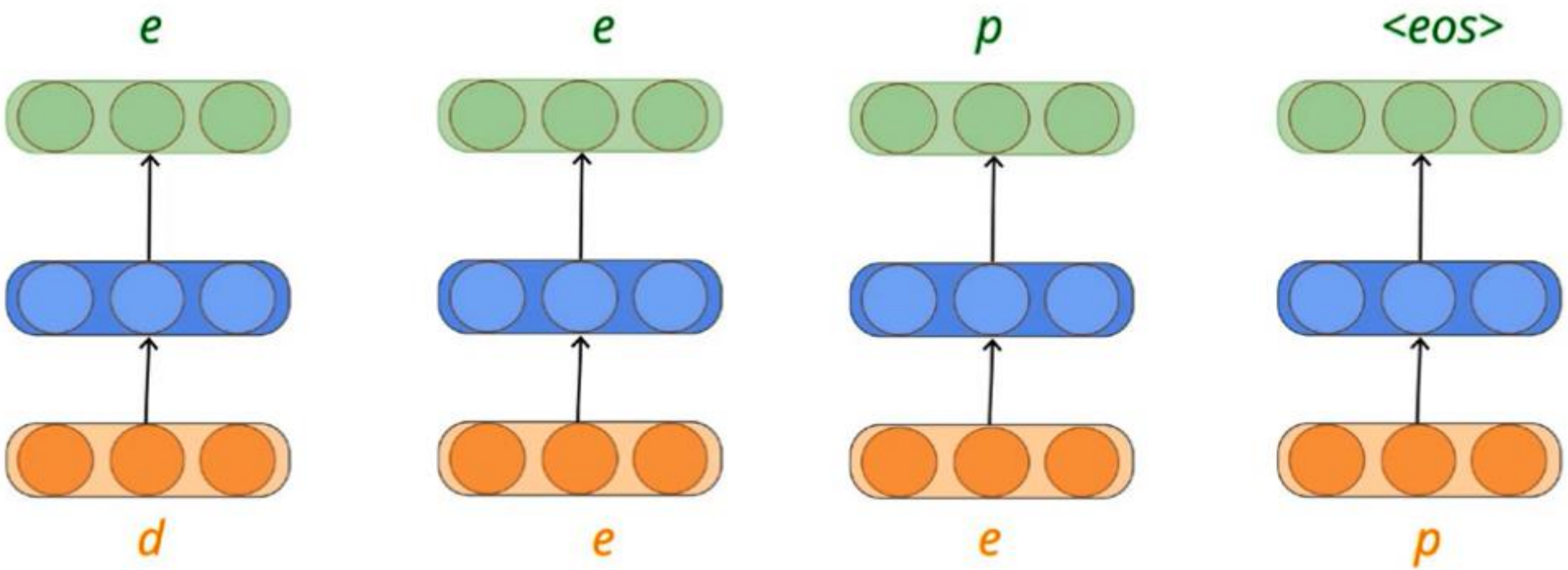
- Outputs are independent of previous inputs
- Input is of a fixed length



x1 height weight ... . sugar bp ECG

x2 height weight ... . sugar bp ECG

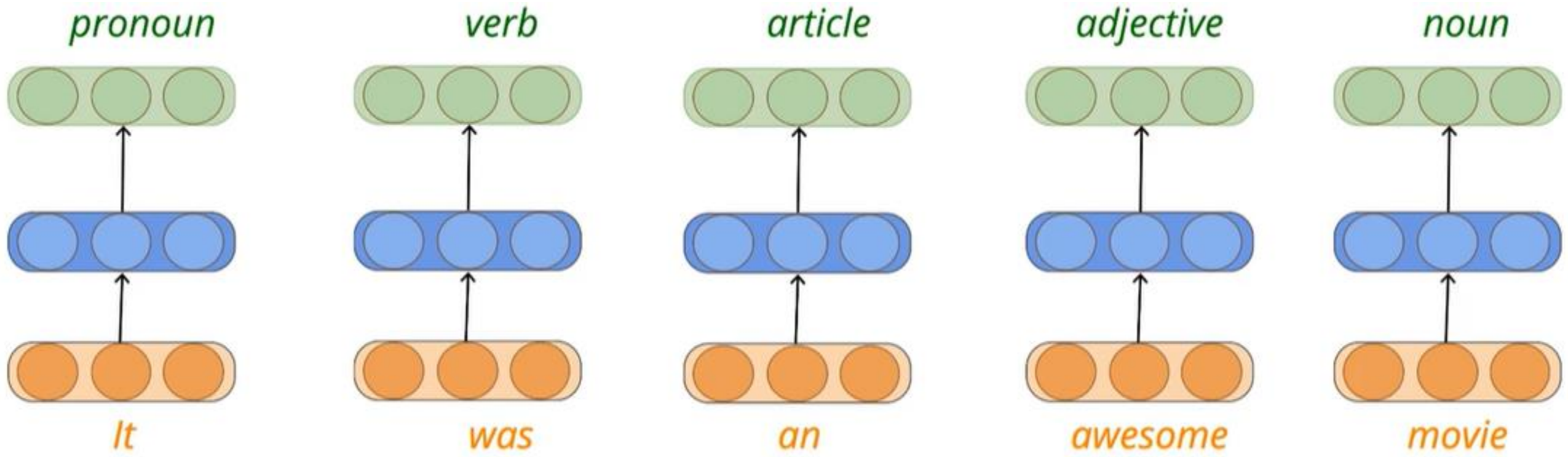
Sequence Learning Problems



- Outputs depend on previous inputs also
- The length of the input is not fixed

Sequence Learning Problems

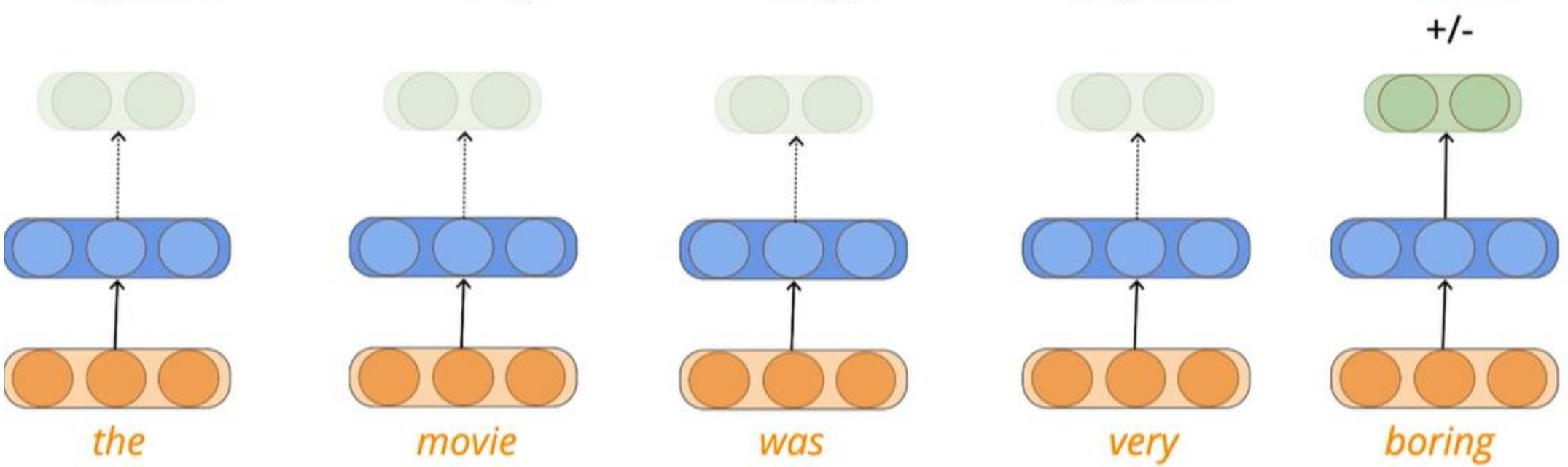
Sequence of words



1 hot encoding used to convert input to number

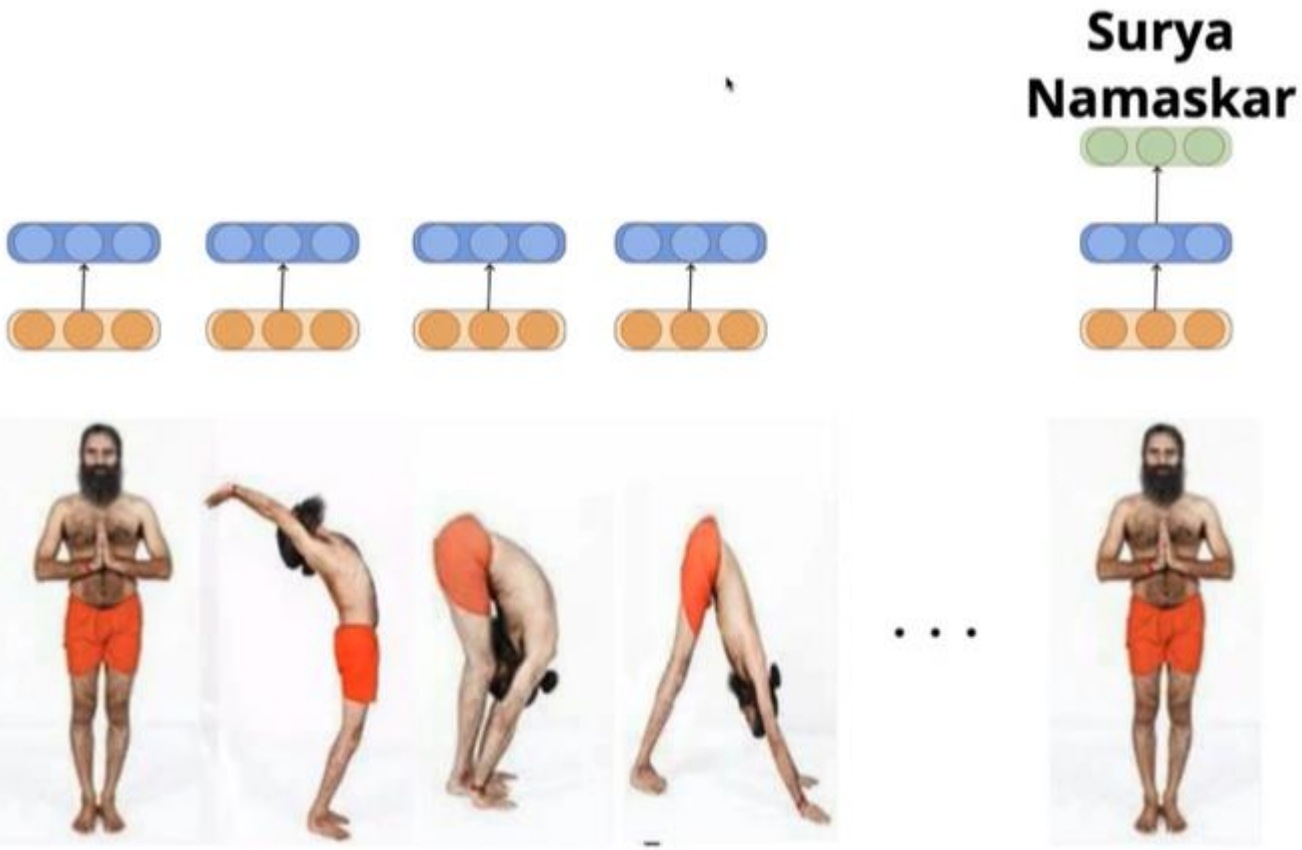
Awesome being an adjective has helped in identifying **movie** as noun
Some words like **bank**- may be used as noun or verb. So the context matters

Sequence Learning Problems- Predict the polarity of the sentence



The no: of input and output may not be same. Here the input is a sequence of words and output is single which classifies the sentence polarity- positive or negative.

Other sequence learning problems



Classify the sequence of yoga posters as one Yoga exercise name - Input is a sequence of frames, and output is a single Yoga exercise name. challenges- Variable no of frames based on speed of action

Speech

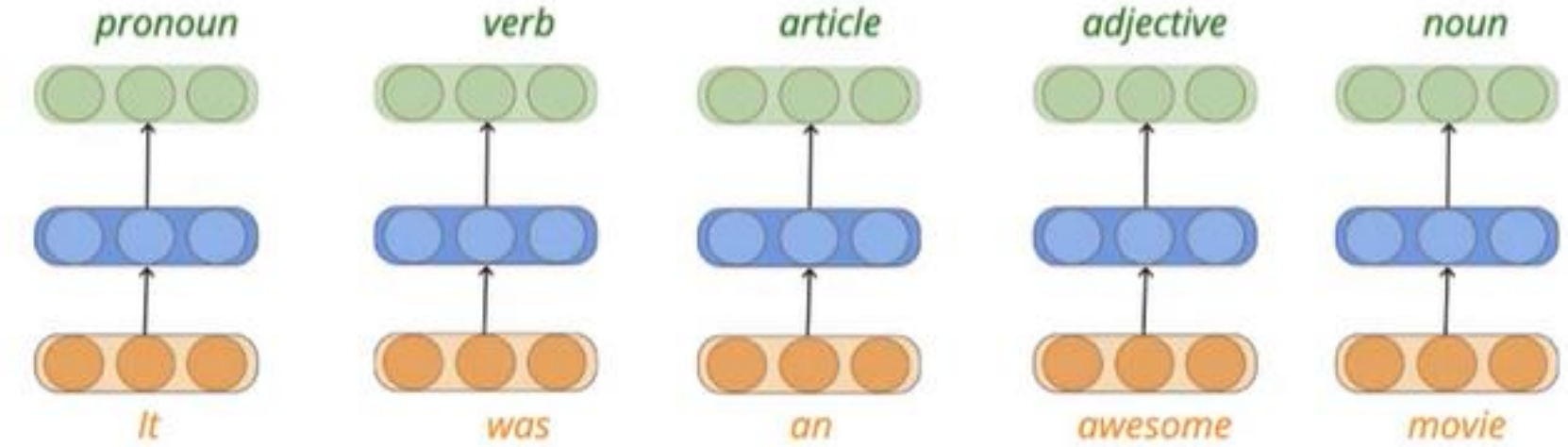
Speech Processing- another sequence learning problem. Take audio signals as input and classify each of them as phonemes

Yoga Video classification :. Each frame in the video correspond to a pose and we want to classify each of the frames into one pose resulting in a sequence of poses

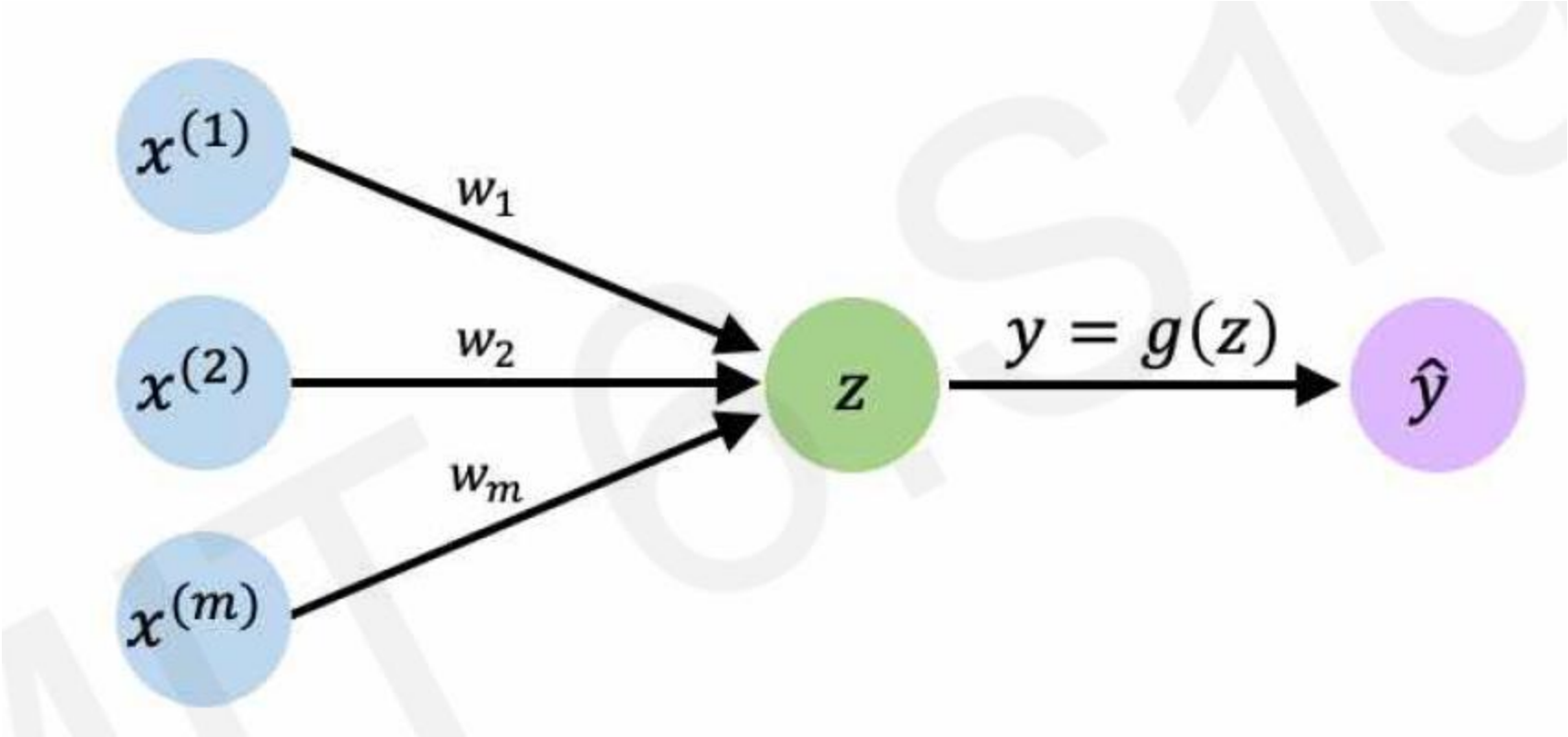
What can be a solution?

$$y_t = \hat{f}(x_1, x_2, ..., x_t)$$

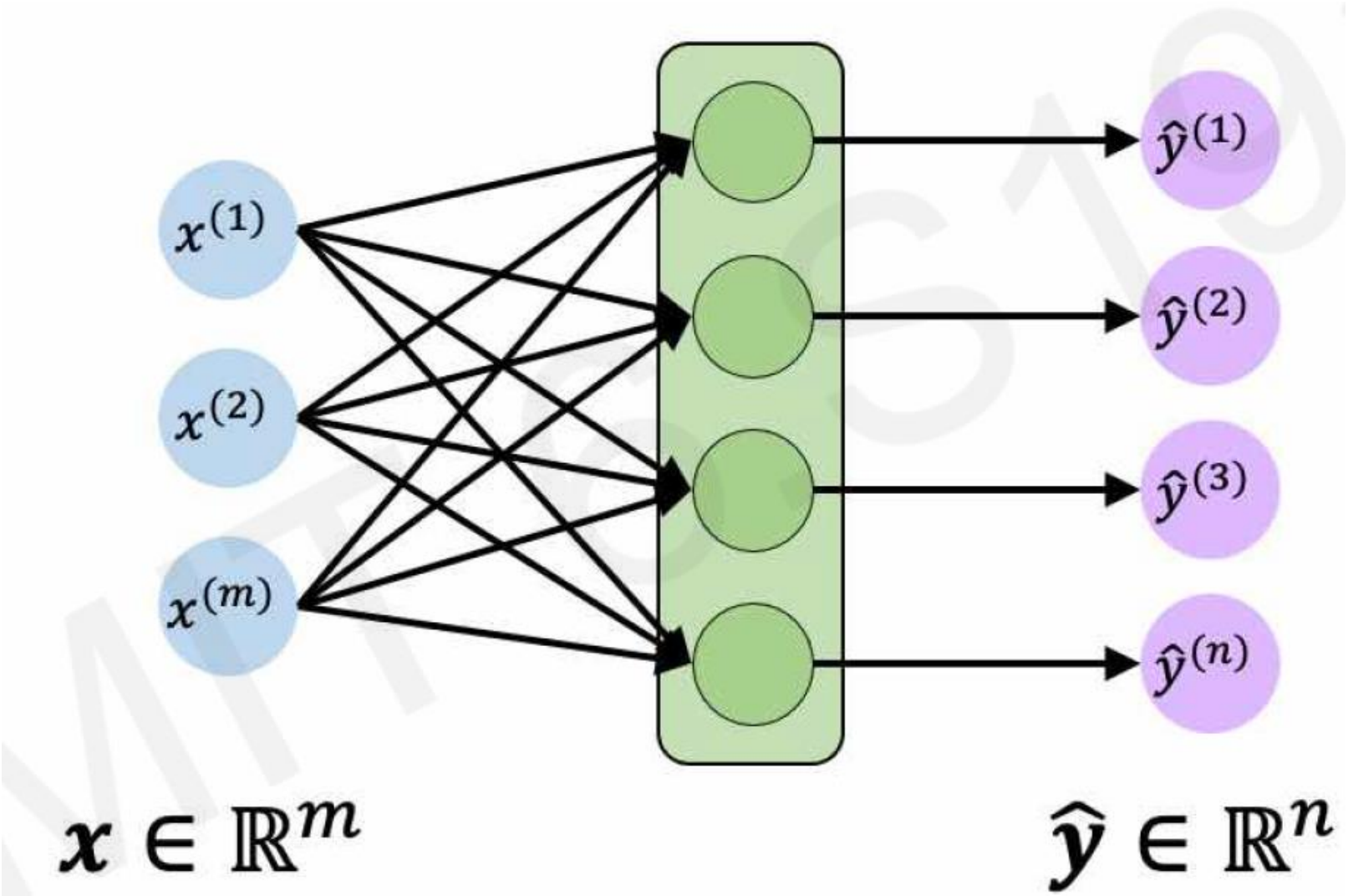
- Ensure that y_t is dependent on previous inputs also
- Ensure that the function can deal with variable number of inputs
- **Ensure that the function executed at each time step is the same**



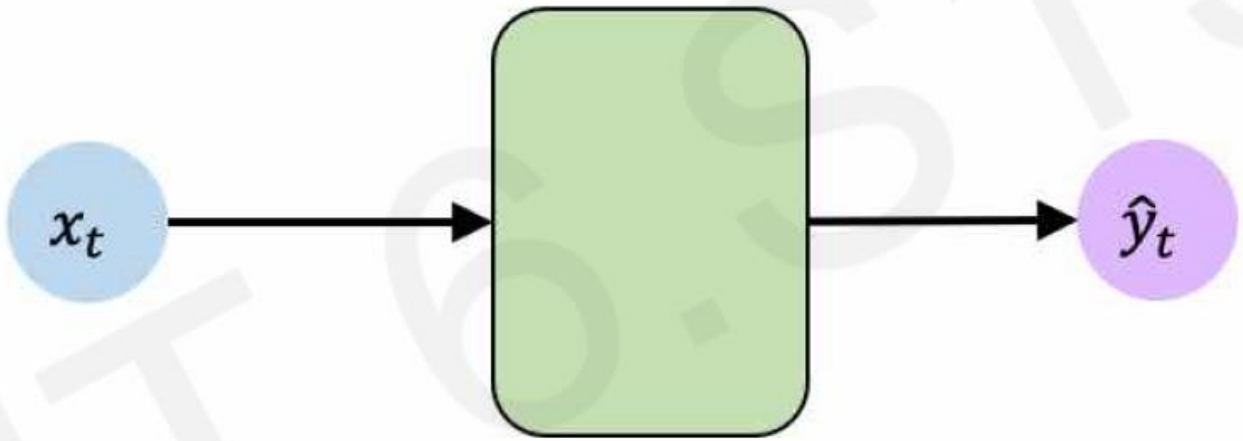
Perceptron Revisited



Feed Forward Revisited



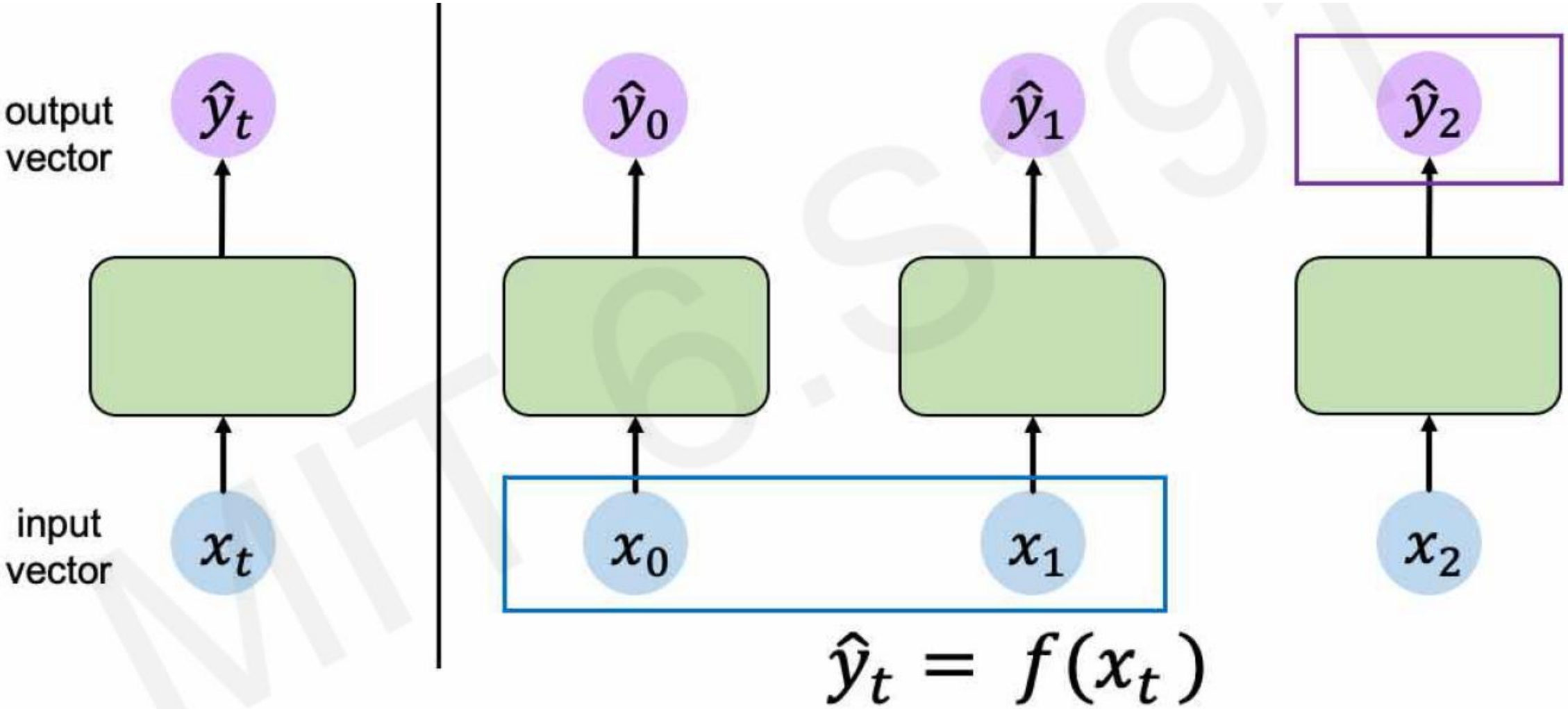
Feed Forward Revisited



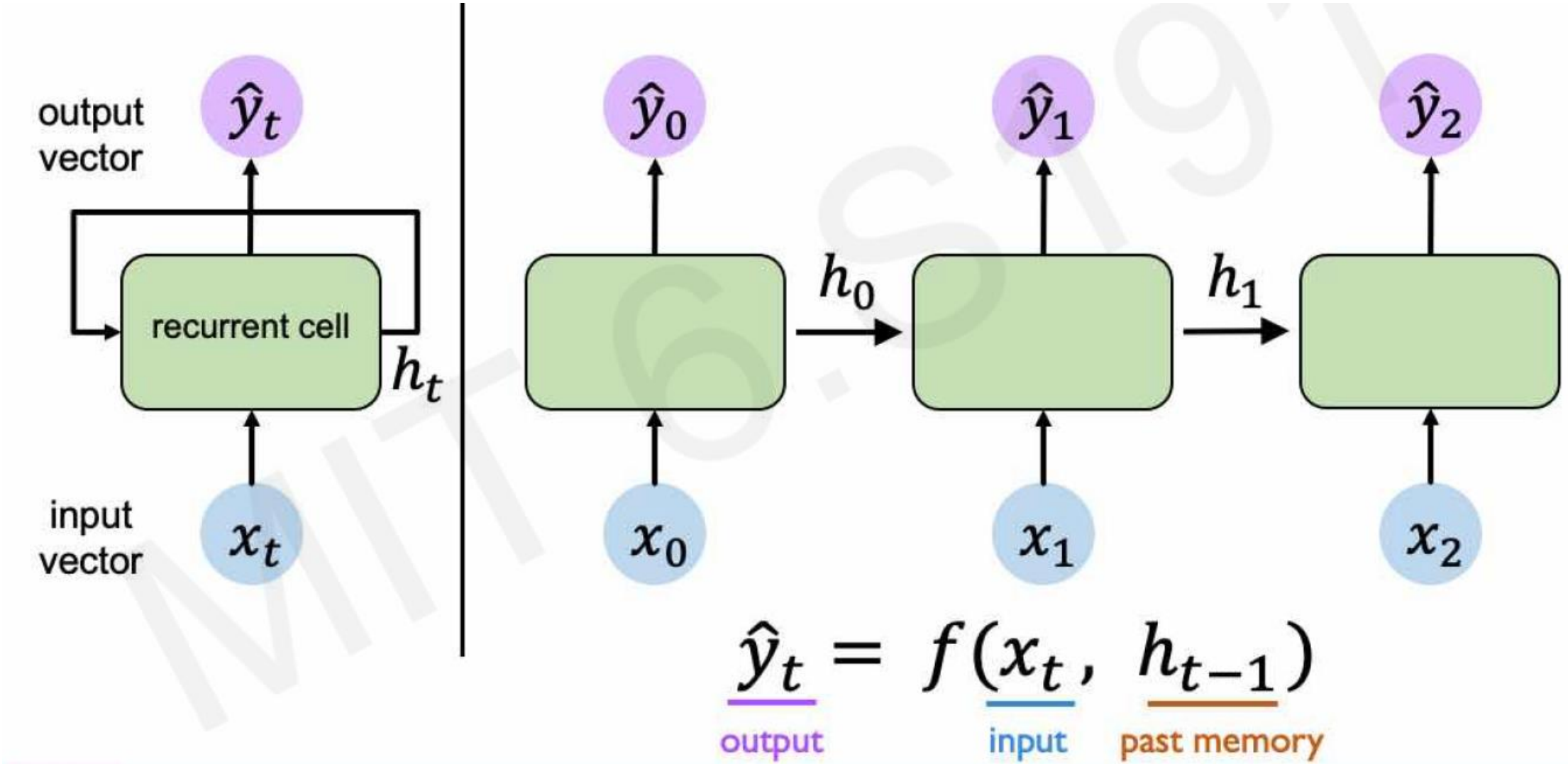
$$x_t \in \mathbb{R}^m$$

$$\hat{y}_t \in \mathbb{R}^n$$

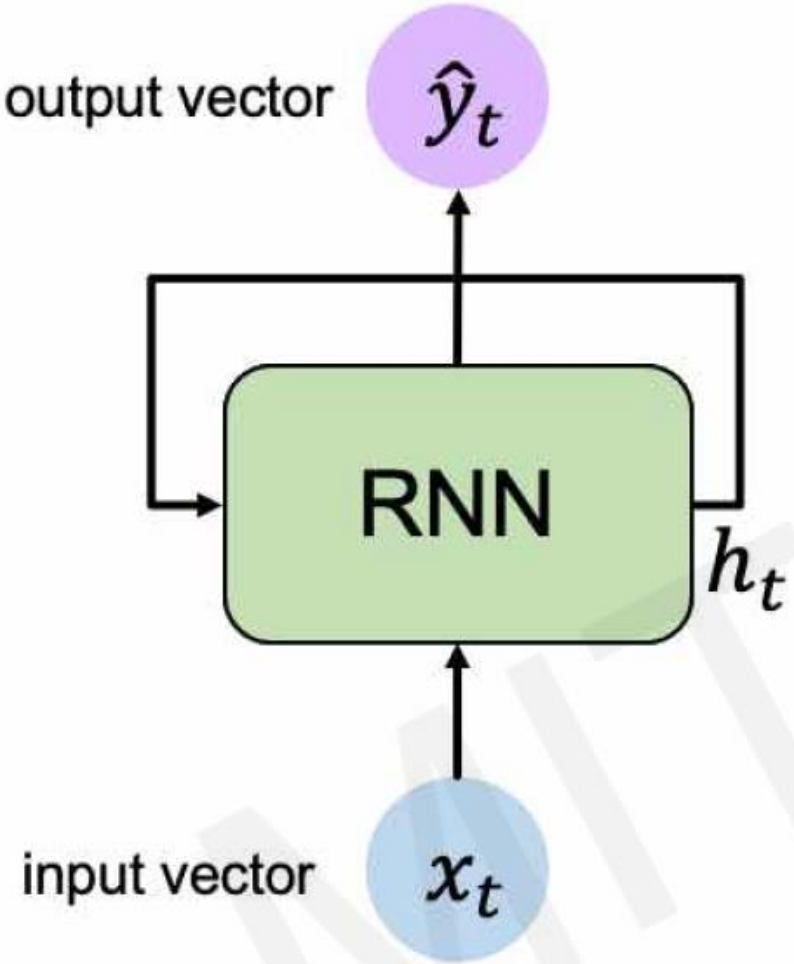
Handling individual Time steps



Neurons with Recurrence



Recurrent Neural Networks



Apply a **recurrence relation** at every time step to process a sequence:

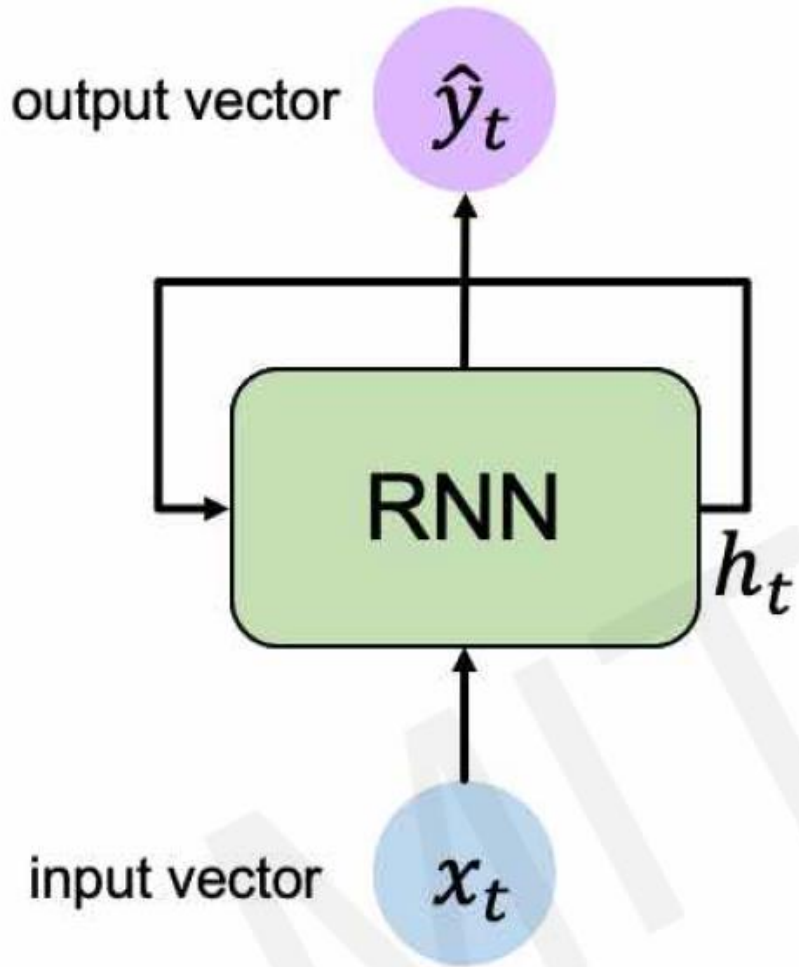
$$h_t = f_W(x_t, h_{t-1})$$

cell state | function with weights W | input | old state

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

RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

Recurrent Neural Networks



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{hy}^{T} h_t$$

Update Hidden State

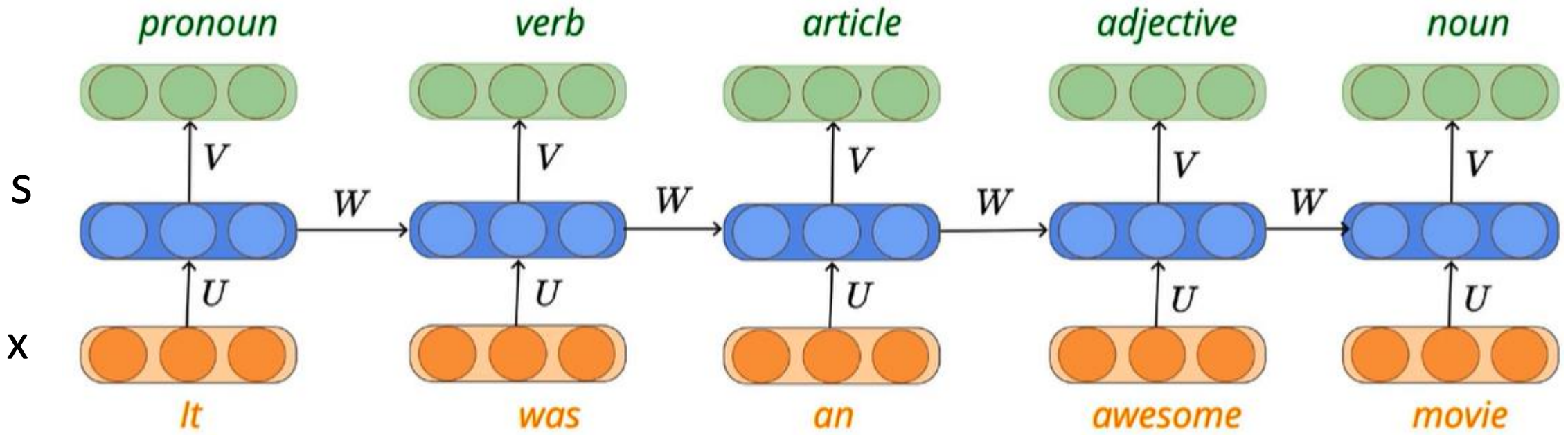
$$h_t = \tanh(\boldsymbol{W}_{hh}^{T} h_{t-1} + \boldsymbol{W}_{xh}^{T} x_t)$$

Input Vector

$$x_t$$

A solution – Recurrent Neural Networks (RNN)

RNN satisfies all the 3 criterias

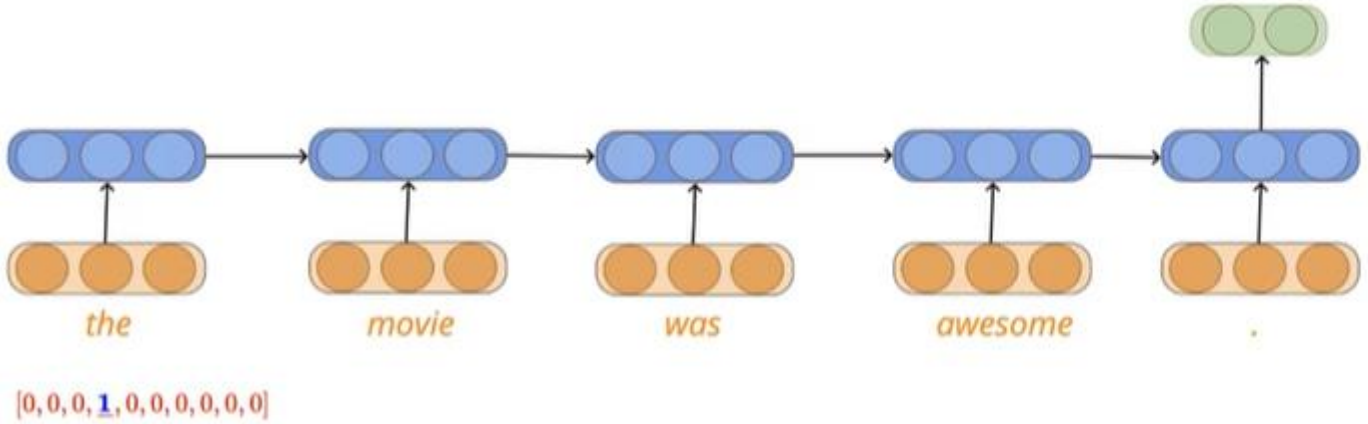


$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$

$$y_i = O(Vs_i + c)$$

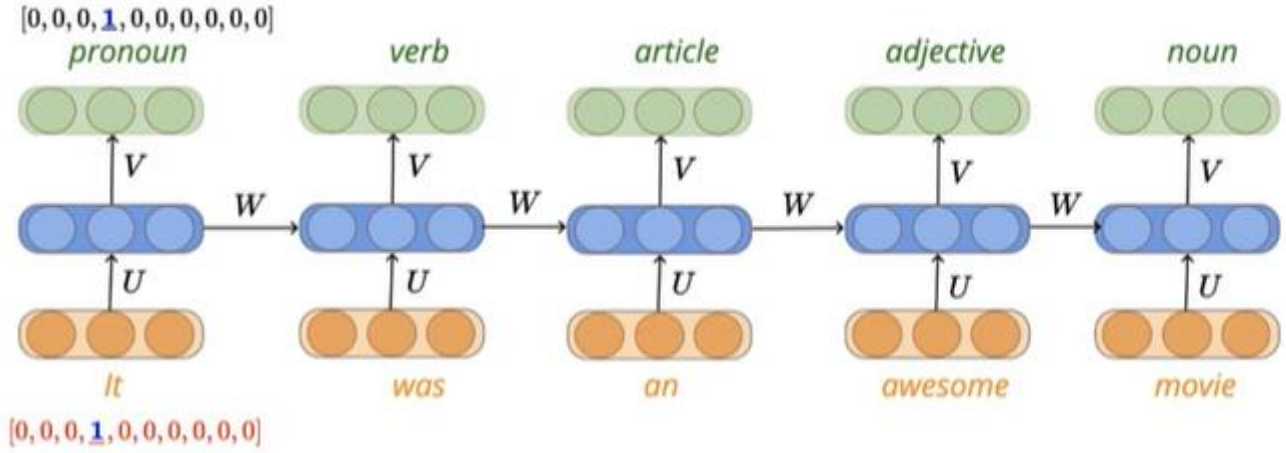
$$y_i = \hat{f}(x_i, s_{i-1}, W, U, V, b, c)$$

Sequence classification problem (eg: sentiment analysis-Polarity)



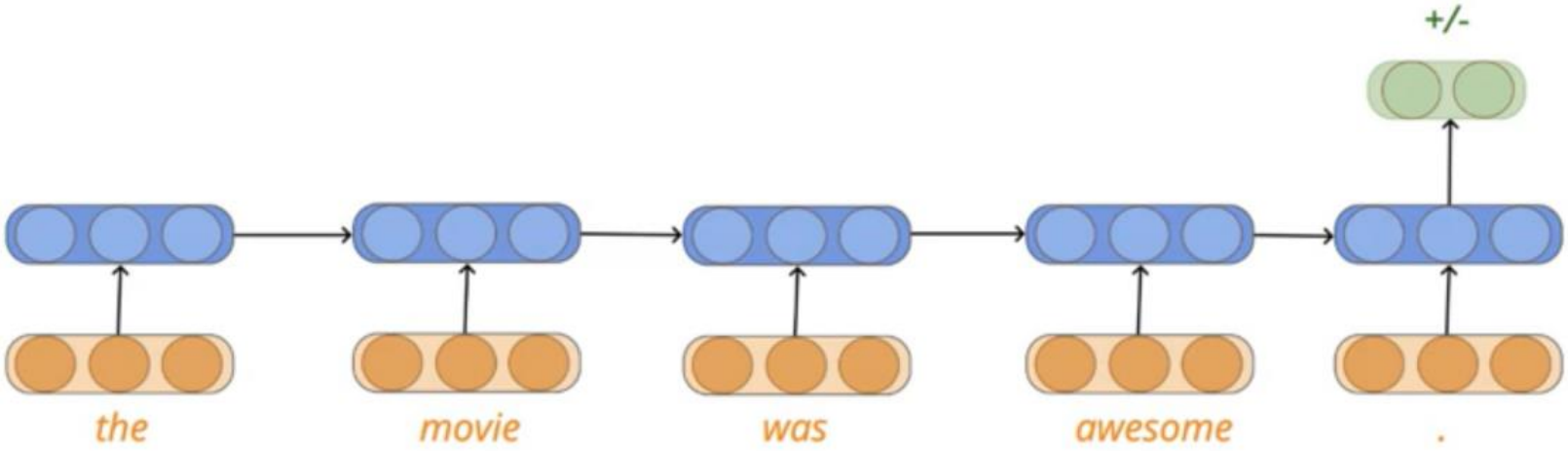
$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$\hat{y} = O(Vs_T + c)$$
$$T = sequence\ length$$

Sequence Labelling problem (eg: parts of speech tagging)



$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$\hat{y}_i = O(Vs_i + c)$$

Loss function for sequence classification problem



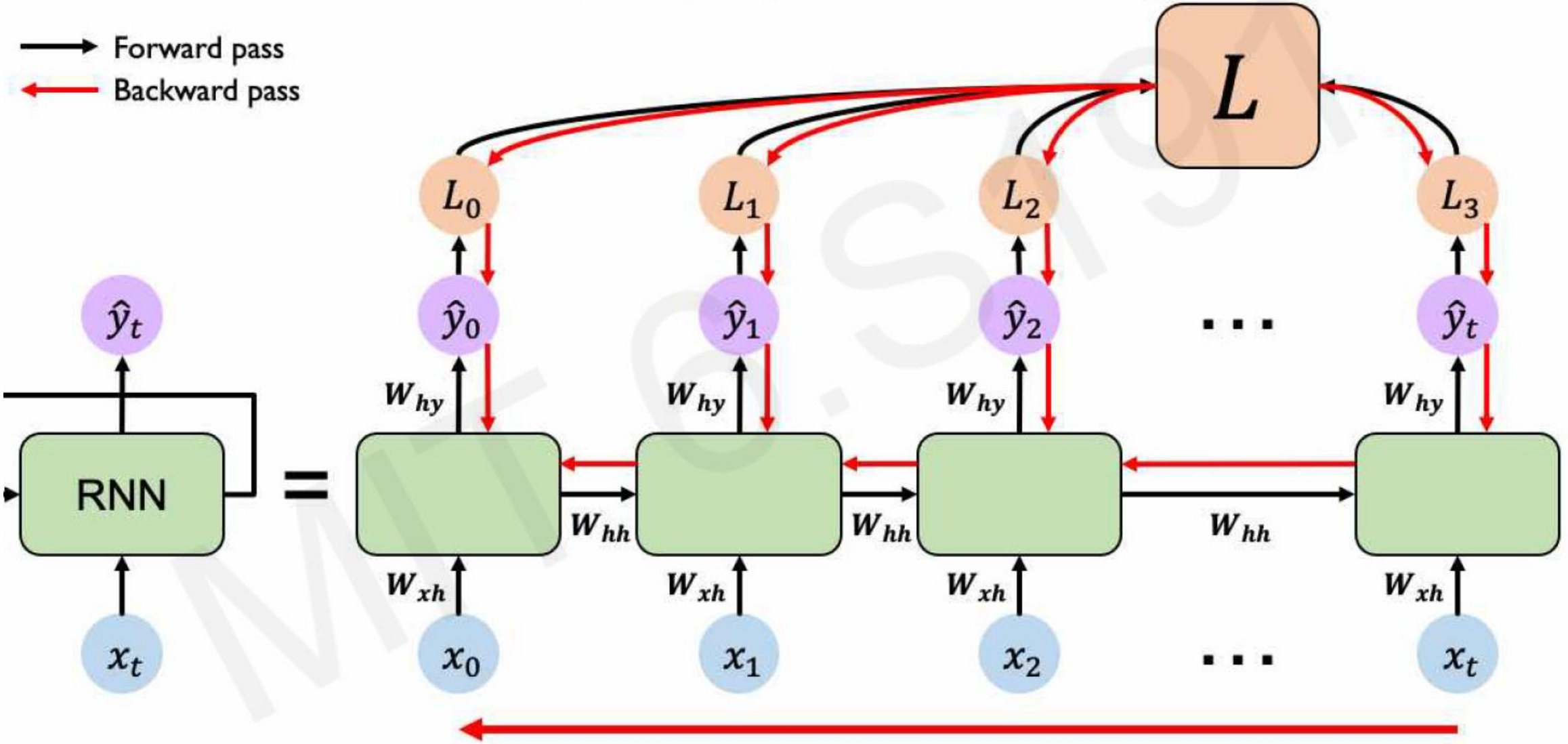
$$\mathscr{L}(\theta) = -\sum_{i=0}^{1} y[i] \log \hat{y}[i]$$
$$= -\log y_c$$
$$= -\log 0.7$$

$$y = [1, 0]$$
$$\hat{y} = [0.7, 0.3]$$

Only one output at the end
Possible output classes [1,0]
Y[i]=0 for one hence -log yc

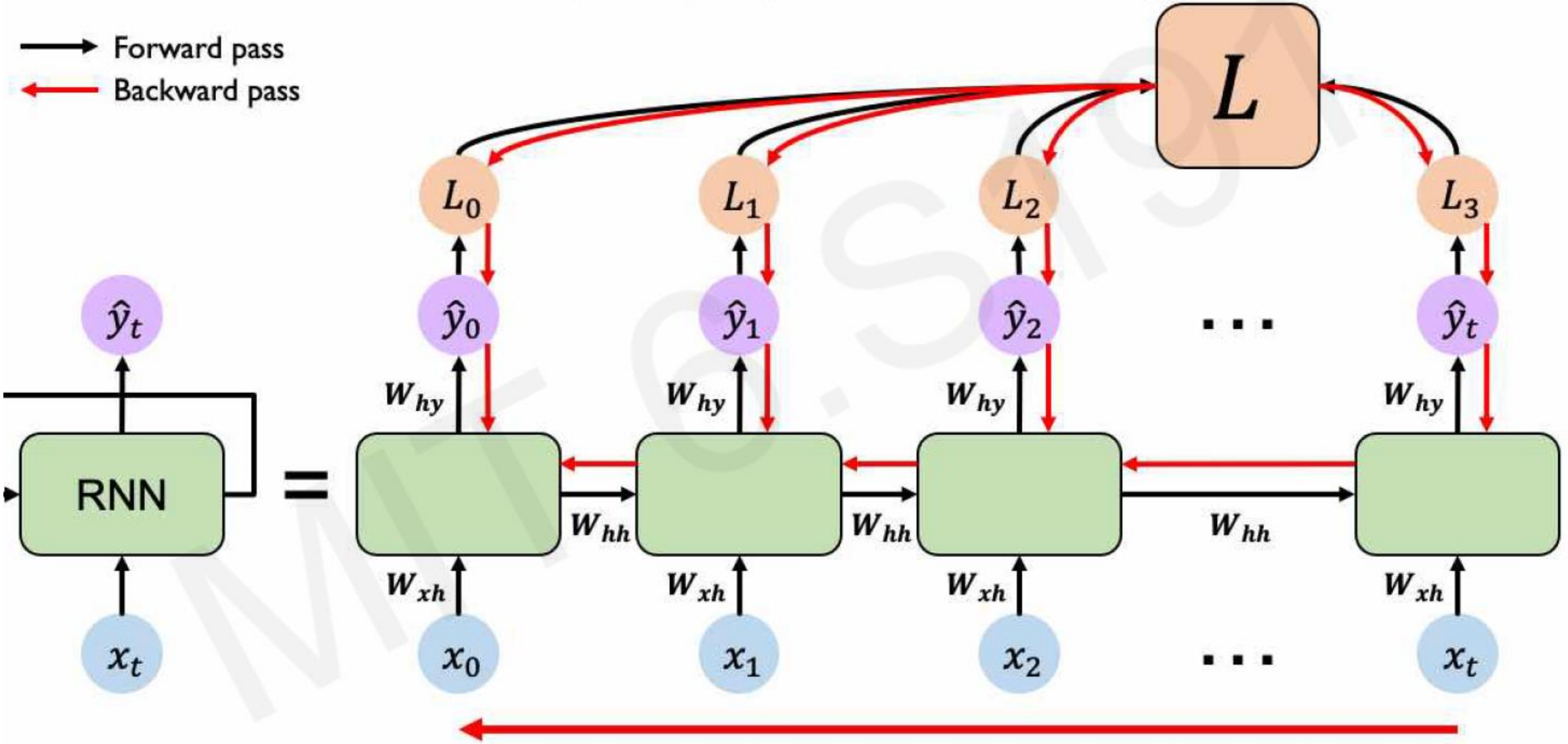
Back propagation through time (BPTT)

Every time step has an output. Sum up Loss for each time step (T)for each data samples (m) and average



Back propagation through time (BPTT)

Every time step has an output. Sum up Loss for each time step (T)for each data samples (m) and average



Back propagation through time (BPTT)

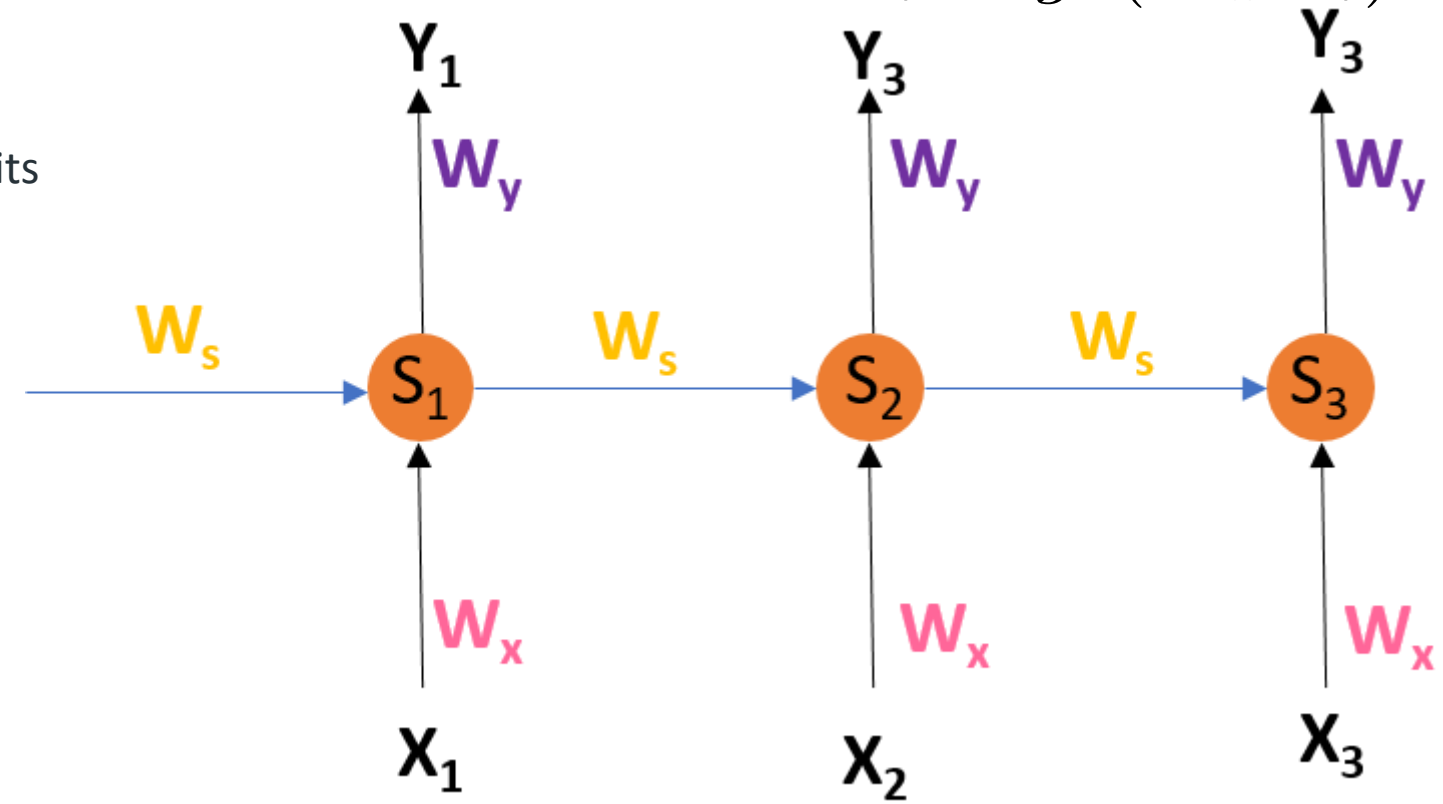
$$S_t = g_1(W_x x_t + W_s S_{t-1})$$

$$Y_t = g_2(W_Y S_t)$$

S1, S2, S3 are the hidden states or memory units at time t1, t2, t3 respectively, and *Ws* is the weight matrix associated with it.

X1, X2, X3 are the inputs at time *t1, t2, t3* respectively, and *Wx* is the weight matrix associated with it.

Y1, Y2, Y3 are the outputs at time *t1, t2, t3* respectively, and *Wy* is the weight matrix associated with it.



We don't independently train the system at a specific time *"t"*. We train it at a specific time *"t"* as well as all that has happened before time *"t"* like t-1, t-2, t-3.

Back propagation through time (BPTT)

$$S_t = g_1(W_x x_t + W_s S_{t-1})$$

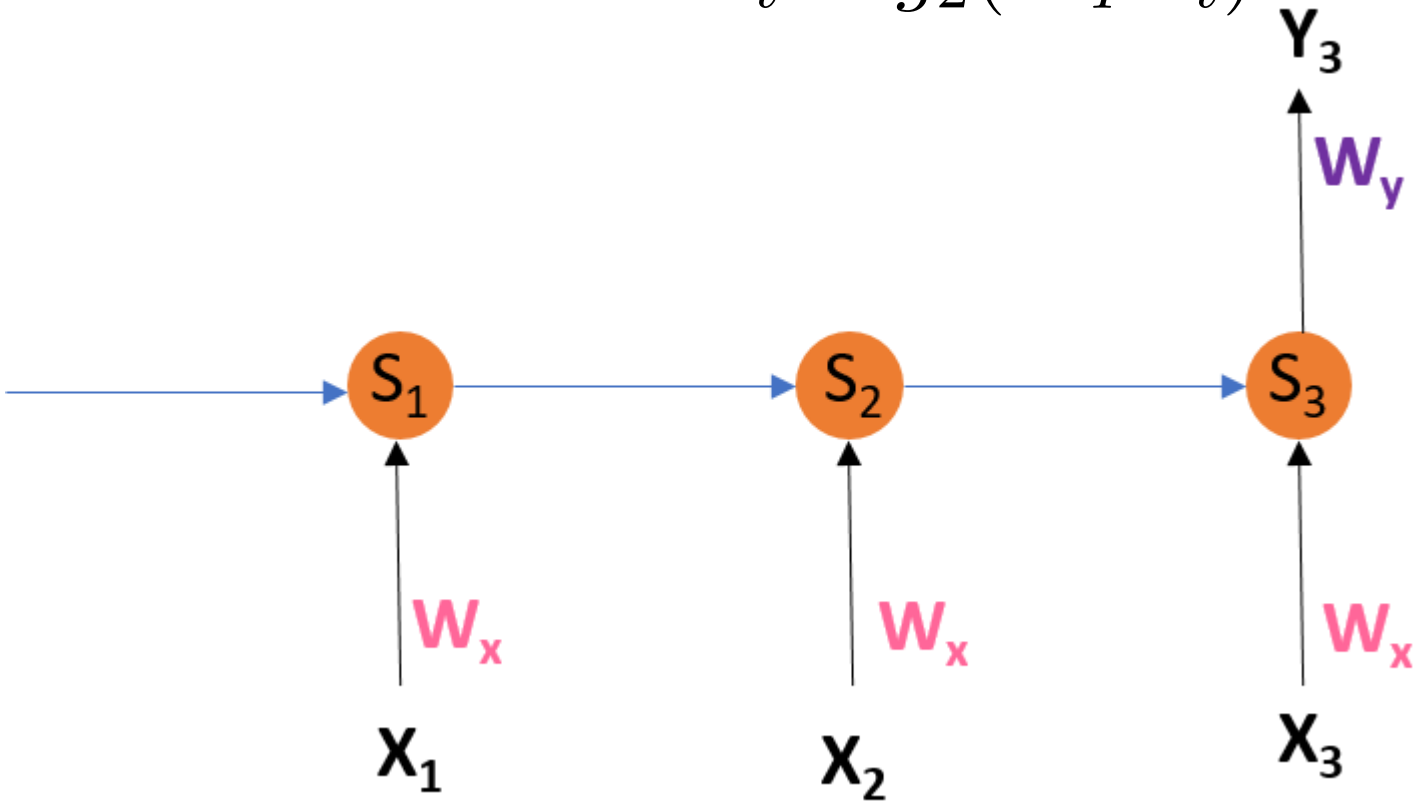
$$Y_t = g_2(W_Y S_t)$$

$$\frac{\partial E_3}{\partial W_X} = (\frac{\partial E_3}{\partial Y_3}.\frac{\partial Y_3}{\partial S_3}.\frac{\partial S_3}{\partial W_X})+$$

$$(\frac{\partial E_3}{\partial Y_3}.\frac{\partial Y_3}{\partial S_3}.\frac{\partial S_3}{\partial S_2}.\frac{\partial S_2}{\partial W_X})+$$

$$(\frac{\partial E_3}{\partial Y_3}.\frac{\partial Y_3}{\partial S_3}.\frac{\partial S_3}{\partial S_2}.\frac{\partial S_2}{\partial S_1}.\frac{\partial S_1}{\partial W_X})$$

$$\frac{\partial E_N}{\partial W_S} = \sum_{i=1}^{N} \frac{\partial E_N}{\partial Y_N}.\frac{\partial Y_N}{\partial S_i}.\frac{\partial S_i}{\partial W_X}$$



Training Algorithm – Back propagation through time (BPTT)

Initialise w, b

Iterate over data:

$compute\ \hat{y}$

$compute\ \mathscr{L}(w, b)$

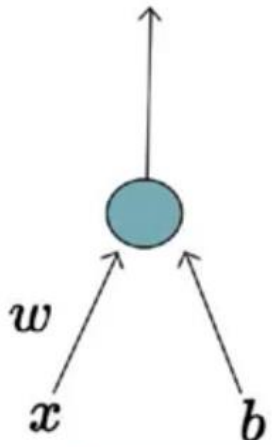
$$w_{11} = w_{11} - \eta \Delta w_{11}$$

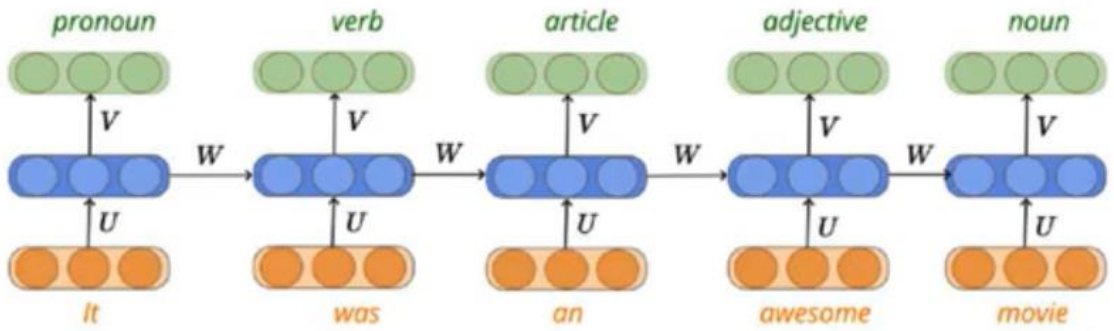
$$u_{12} = u_{12} - \eta \Delta u_{12}$$

....

$$v_{13} = v_{13} - \eta \Delta v_{13}$$

till satisfied





$$Earlier : w, b$$

$$Now : w_{11}, w_{12}, \ldots, u_{11}, u_{12}, \ldots, v_{11}, v_{12}$$

$$Earlier : L(w, b)$$

$$Now : L(W, U, V)$$

Limitations:

This method of Back Propagation through time (BPTT) can be used up to a limited number of time steps like 8 or 10. If we back propagate further, the gradient \delta becomes too small. This problem is called the “Vanishing gradient” problem. The problem is that the contribution of information decays geometrically over time. So, if the number of time steps is >10 (Let’s say), that information will effectively be discarded.

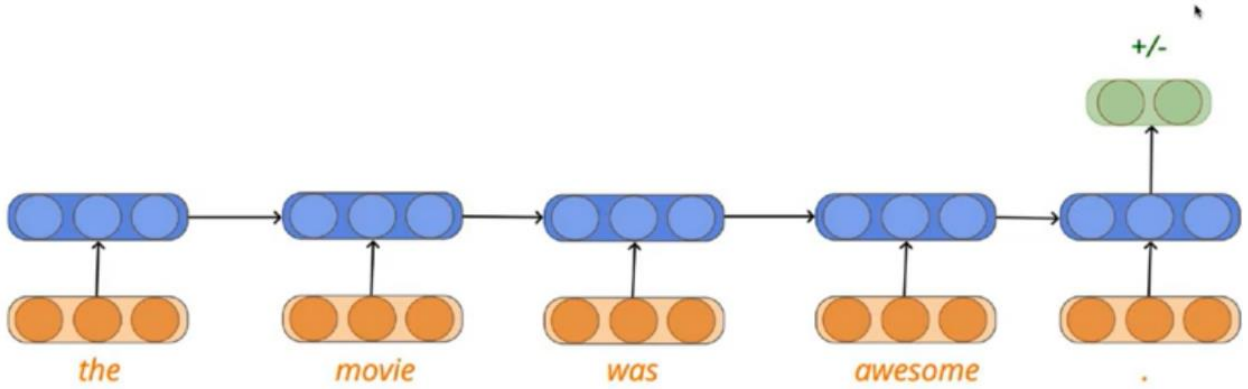
Going Beyond RNNs:

One of the famous solutions to this problem is by using what is called Long Short-Term Memory (LSTM for short) cells instead of the traditional RNN cells. But there might arise yet another problem here, called the exploding gradient problem, where the gradient grows uncontrollably large.

Solution: A popular method called gradient clipping can be used where in each time step, we can check if the gradient \delta > threshold. If yes, then normalize it.

Go for Transformers!

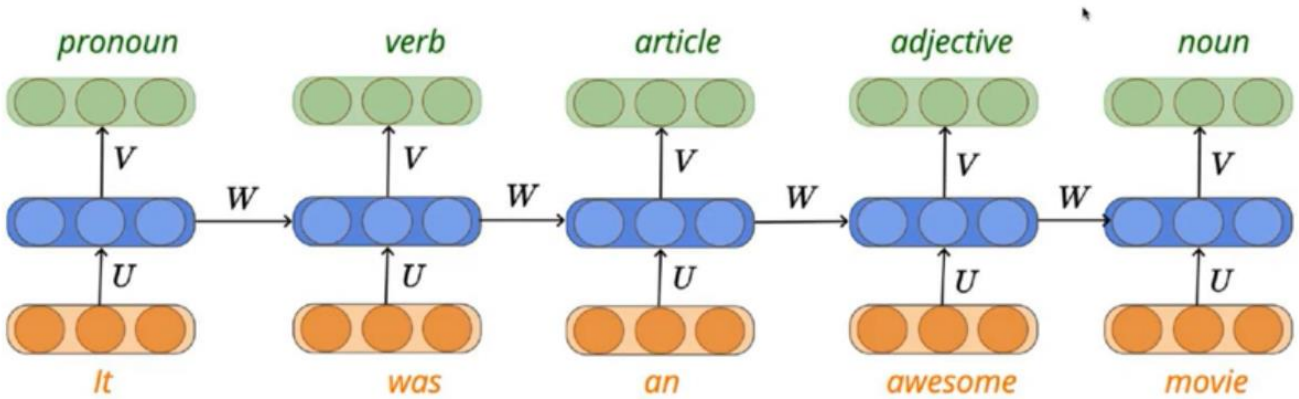
Evaluation- Sequence classification



Predicted y cap	Ground Truth y	Correct/incorrect
1(P)	1(P)	correct
0(N)	0(N)	incorrect
1(P)	0(N)	incorrect
0(N)	0(N)	correct
1(P)	1(P)	correct
0(N)	1(P)	incorrect

$$\textbf{Accuracy} = \frac{No{:}of\ correctly\ classified}{Total\ samples}$$

Evaluation Sequence labelling



Overall accuracy /Accuracy per class

Data sample	Pronoun	verb	article	adjective	noun
D1	P	V	Ar	Ad	N
D2					
D3					
D4					
D5					

- **Overall Accuracy**
- **Accuracy Per Class**
- **Confusion Matrix**

Confusion Matrix

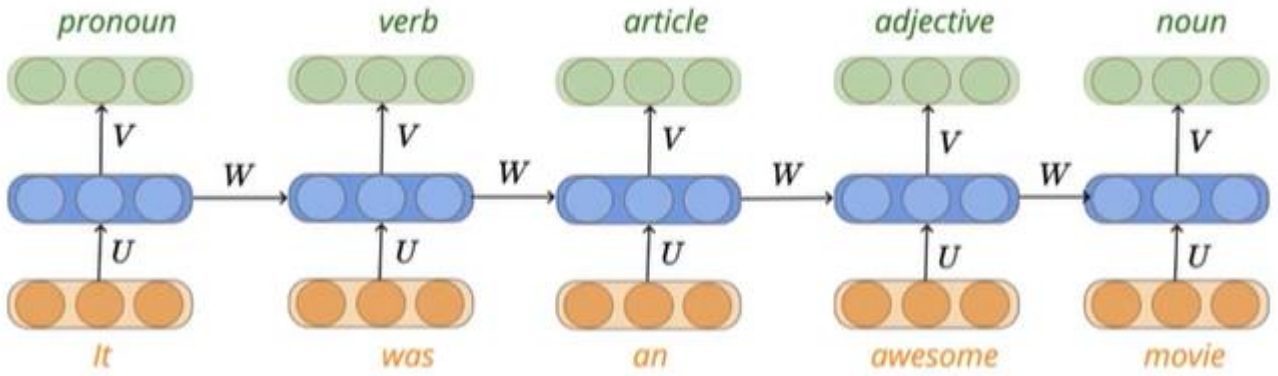
	Pronoun	verb	article	adjective	noun
Pronoun	3	2	3	5	6
verb	2	7			
article			3		
adjective				2	
noun					1

Sequential Models

Long Short-Term Memory Cells(LSTM)

RNN not dealing with longer sequence
(How the state record information when the sequence is very long)

RNN- Recap

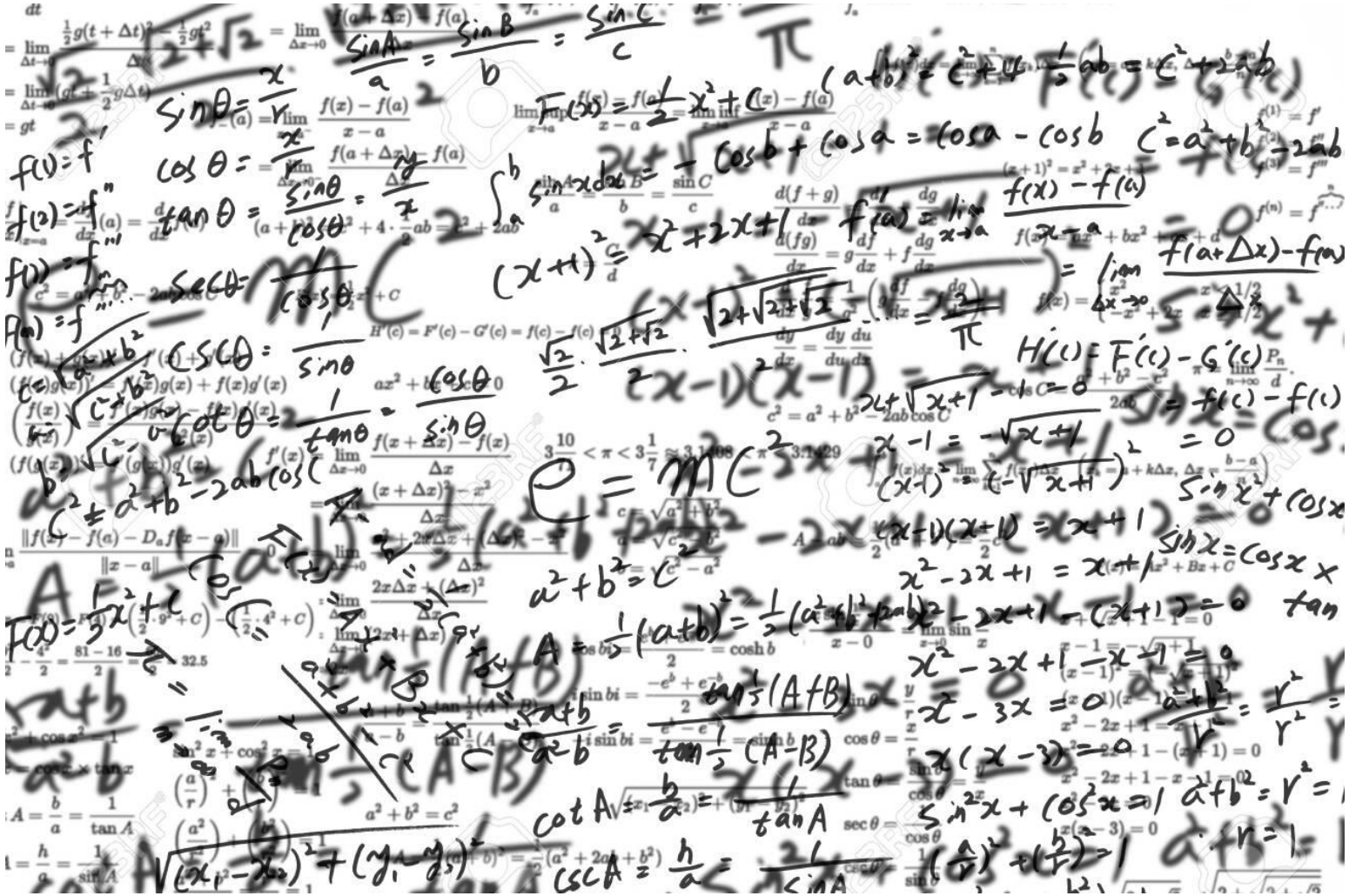


RNN: Exploding and vanishing gradient problem occurs!!
Not suitable for longer sequence

- At each new timestep the old information gets morphed by the current input
- One could imagine that after t steps the information stored at time step t − k (for some k < t) gets completely morphed
- Even during backpropagation the information does not flow well

$$s_i = \sigma(Ux_i + Ws_{i-1} + b)$$
$$y_i = O(Vs_i + c)$$

White board Analogy (over time white board become so messy and u cant make out anything)



White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$

$$bd = 33$$

$$bd + a = 34$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$

$$bd + a = 34$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$

$$ac(bd + a) = 170$$

$$bd + a = 34$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$

$$ac(bd + a) = 170$$

$$ad = 11$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac(bd + a) = 170$$

$$ad = 11$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ad + ac(bd + a) = 181$$
$$ac(bd + a) = 170$$
$$ad = 11$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

White board Analogy

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

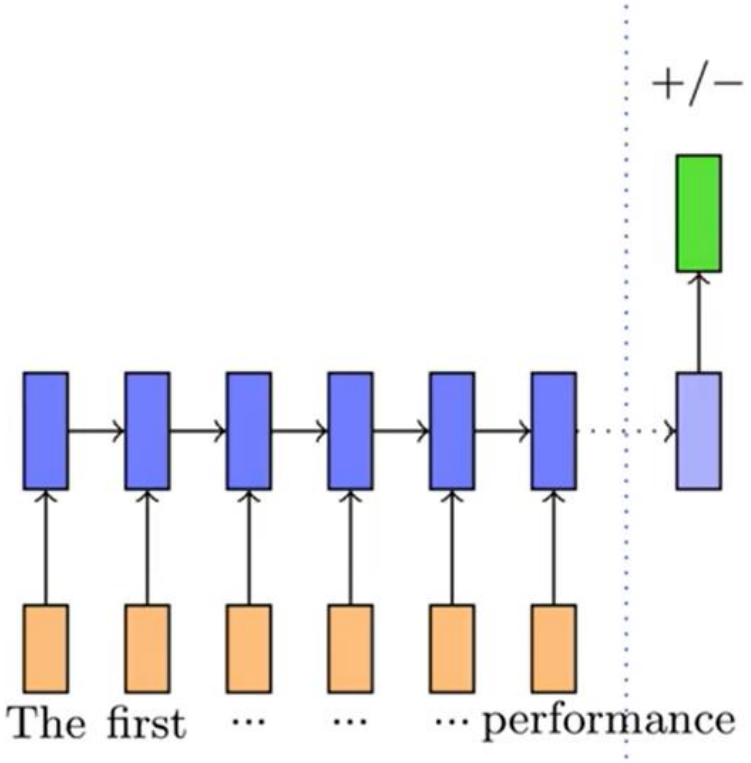
$$ad + ac(bd + a) = 181$$

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

Dealing with longer sequence

(An example where RNN need to selectively read write and forget)



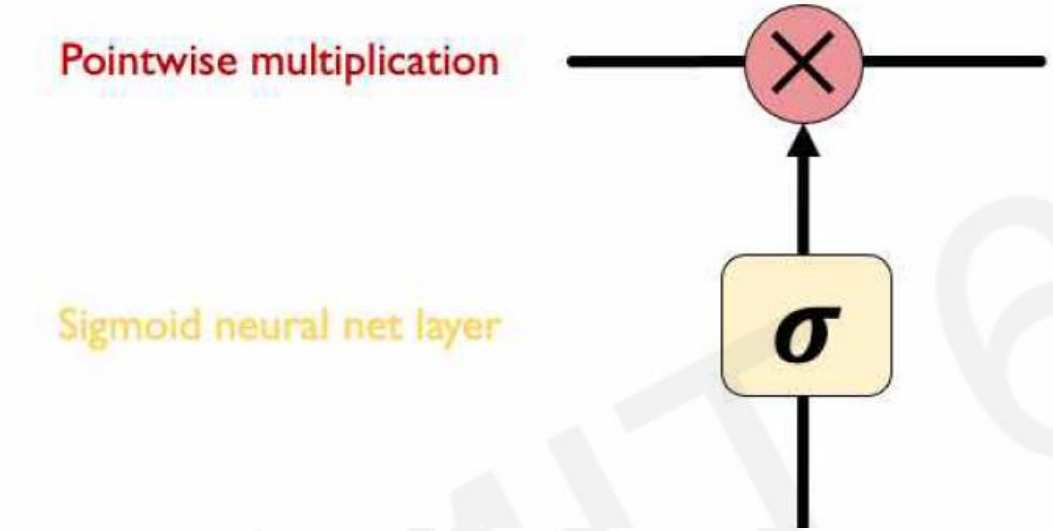
Ideally, we want to

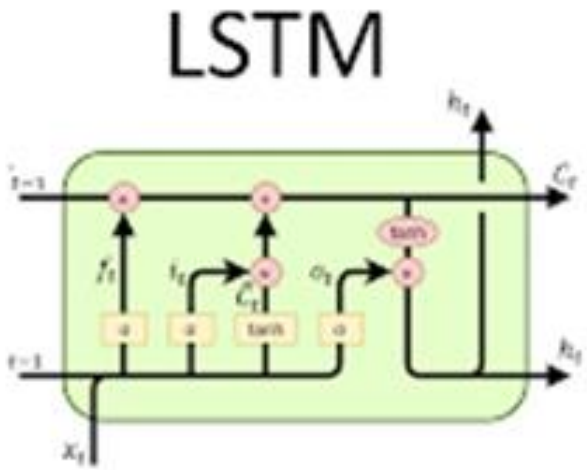
- forget the information added by stop words (a, the, etc.)
- selectively read the information added by previous sentiment bearing words (awesome, amazing, etc.)
- selectively write new information from the current word to the state

Review: The first half of the movie was dry but the second half really picked up pace. The lead actor delivered an amazing performance

Long Short-Term Memory Cells – deals with longer sequences (How to implement selective read write and Forget

Idea: use **gates** to selectively **add** or **remove** information within **each recurrent unit with**

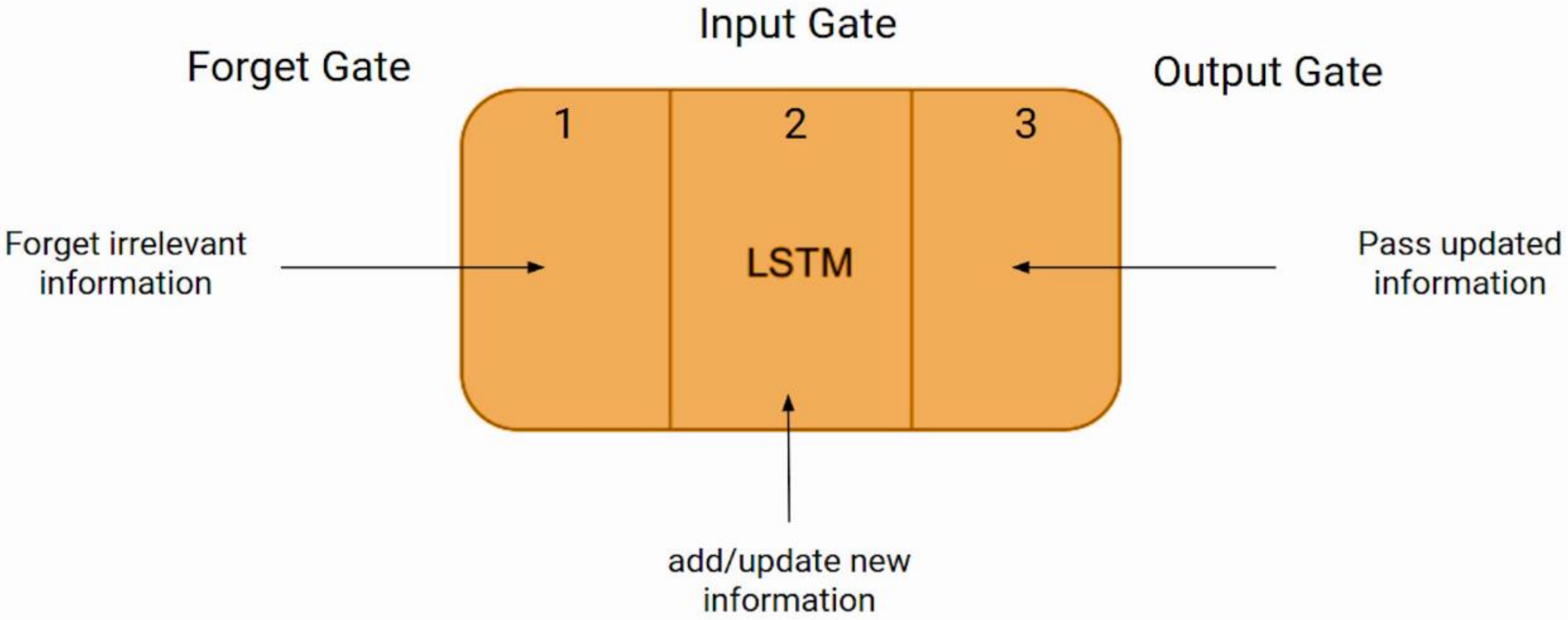




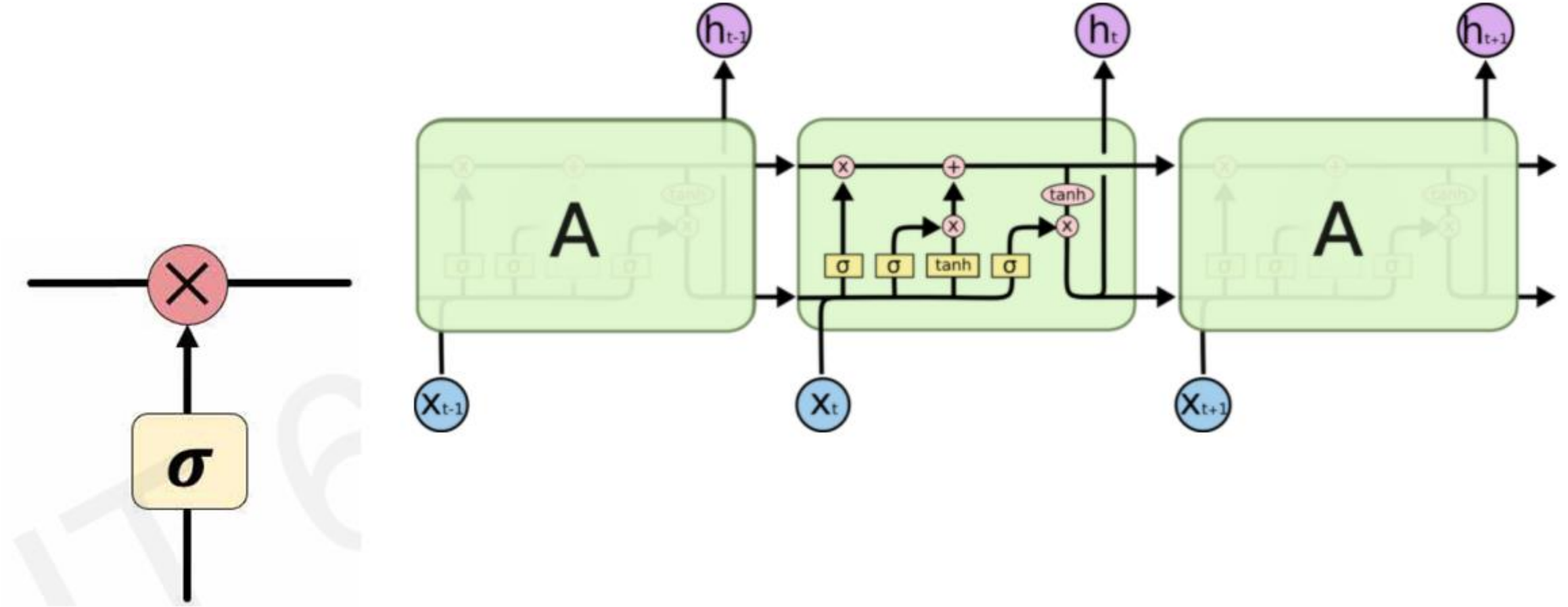
Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

Long Short-Term Memory Cells

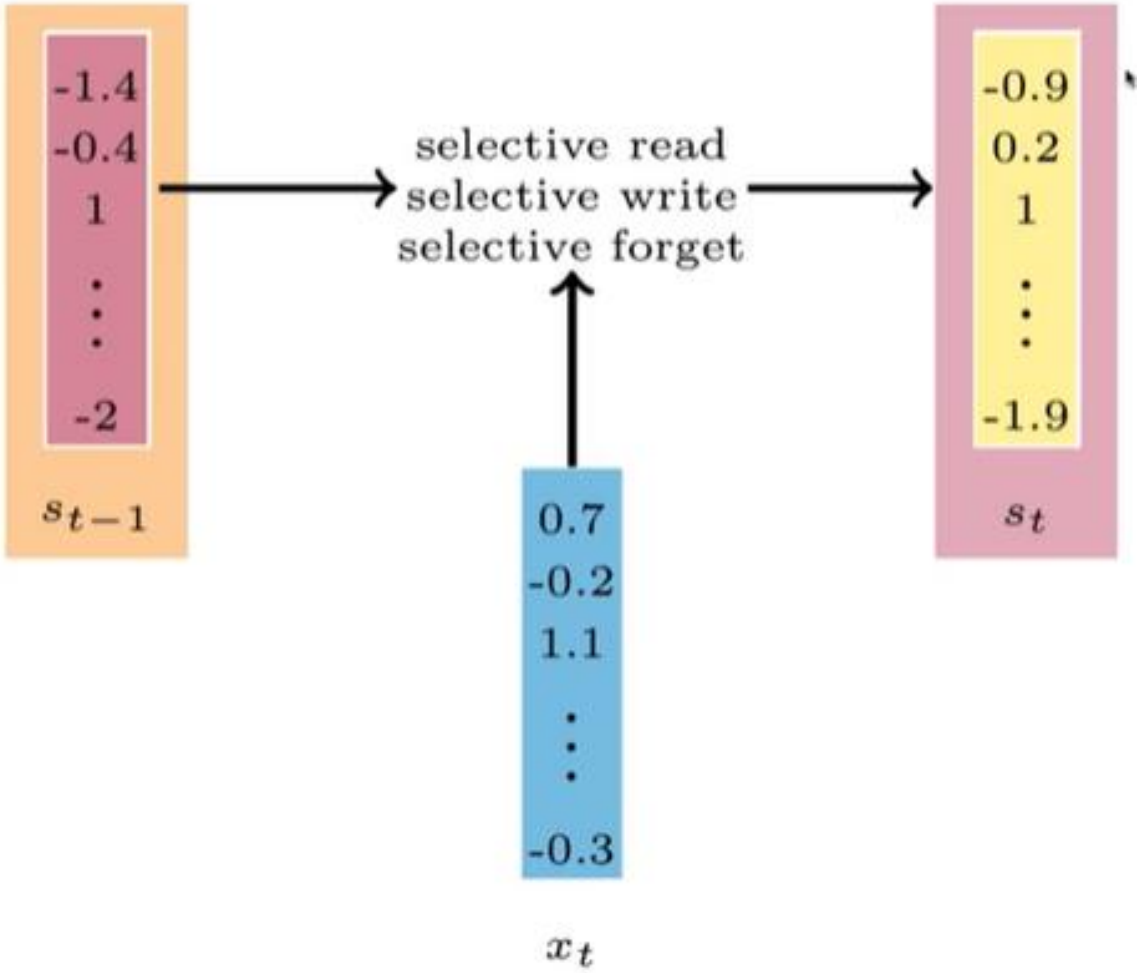
1. Maintain a **cell state**
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input
   - Selectively **update** cell state
   - **Output** gate returns a filtered version of the cell state
3. Backpropagation through time with partially **uninterrupted gradient flow**


Input Gate
Forget Gate
Output Gate
1
2
3
Forget irrelevant information
LSTM
Pass updated information
add/update new information

LSTM



Long Short-Term Memory Cells – deals with longer sequences
(How to implement selective read write and Forget)



LSTM

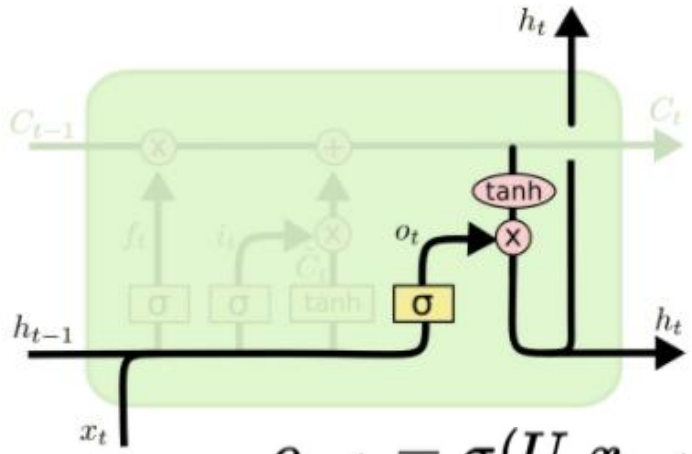
While computing s_t from s_{t-1} we want to make sure that we use selective write, selective read and selective forget so that only important information is retained in s_t

Recap- RNN

$$s_t = \sigma(Ux_t + Ws_{t-1} + b)$$

Selective write using output gate

Bob fought single handedly with the enemy and died for his country. For his contributions brave _____.



In this phrase, there could be a number of options for the empty space. But we know that the current input of *'brave'*, is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Bob could be an apt output. This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate. Here is its structure

$$o_{t-1} = \sigma(U_o x_{t-1} + W_o h_{t-2} + b_o)$$

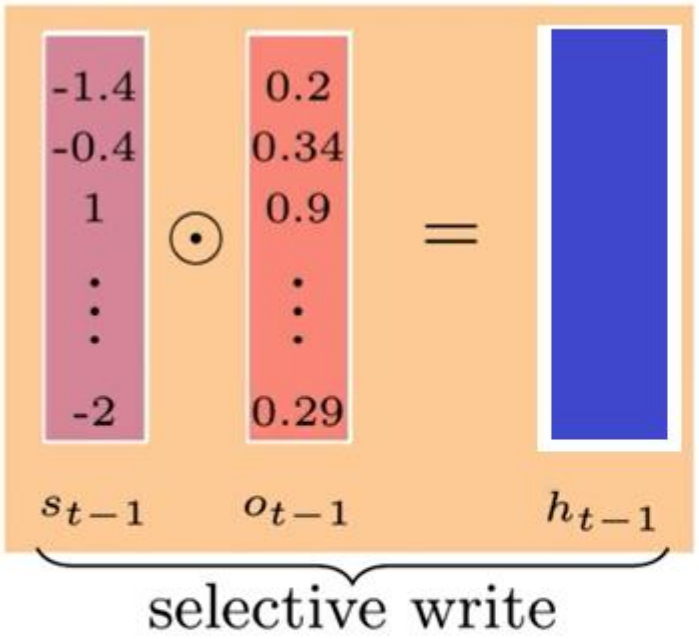
Output gate

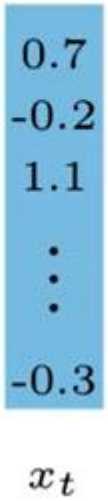
Selective write

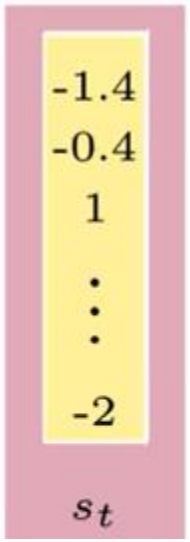
o_t is called the **output** gate

$$o_{t-1} = \sigma(U_o x_{t-1} + W_o h_{t-2} + b_o)$$

$$h_{t-1} = s_{t-1} \odot o_{t-1}$$







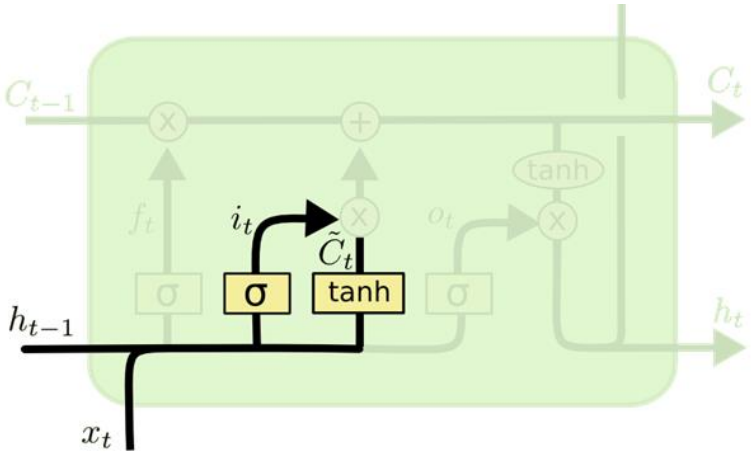
But how do we compute o_{t-1} ? How does the RNN know what fraction of the state to pass on?

- learn o_{t-1} from data
- the only thing that we learn from data is parameters
- **Solution:** express o_{t-1} using parameters

- instead of passing s_{t-1} as it is to s_t we want to pass (write) only some portions of it to the next state
- A reasonable way of doing this would be to assign a value between 0 and 1 which determines what fraction of the current state to pass on to the next state

Selective read using input gate

Bob knows swimming. He told me over the phone that he had served the navy for 4 long years.



Input gate:

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

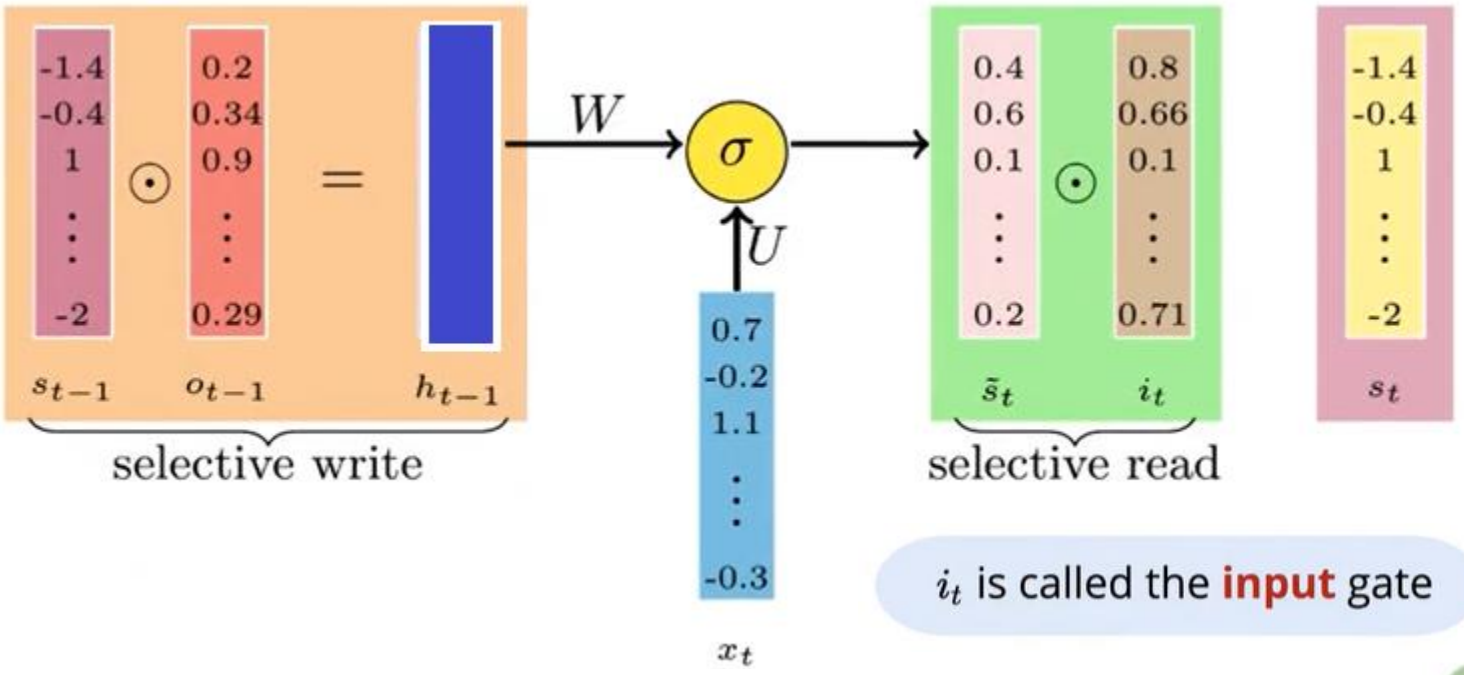
Now the important information here is that “Bob” knows swimming and that he has served the Navy for four years. This can be added to the cell state, however, the fact that he told all this over the phone is a less important fact and can be ignored. This process of selective read can be done via the **input** gate.

Selective Read

$$\tilde{s}_t = \sigma(Ux_t + Wh_{t-1} + b)$$

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

$$= \tilde{s}_t \odot i_t$$



i_t is called the **input** gate

Previous state:

s_{t-1}

Output gate:

$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$

Selectively Write:

$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$

Current (temporary) state:

$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$

Input gate:

$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$

Selectively Read:

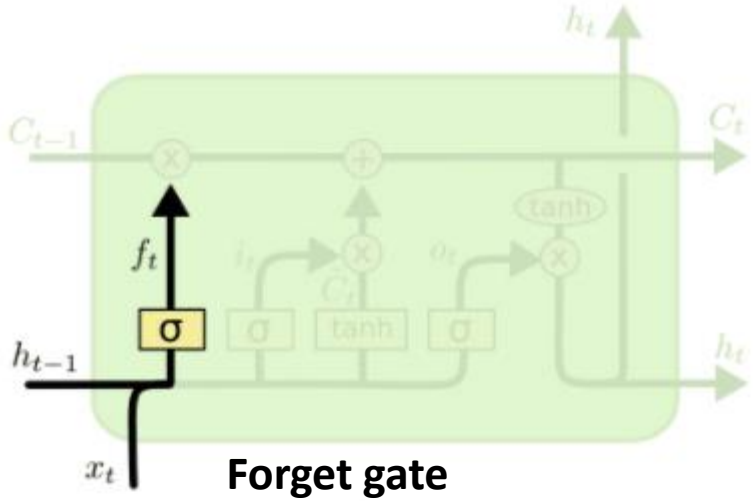
$i_t \odot \tilde{s}_t$

- $\tilde{s}_t$ thus captures all the information from the previous state h_{t-1} and the current input x_t
- However, we may not want to use all this new information and only selectively read from it before constructing the new cell

Selective Forget using Forget Gate

Bob is a nice person. Dan on the other hand is evil.

As soon as the first full stop after "*person*" is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the *subject* of the sentence is *forgotten* and the place for the subject is vacated. And when we start speaking about "*Dan*" this position of the subject is allocated to "*Dan*". This process of forgetting the subject is brought about by the forget gate.



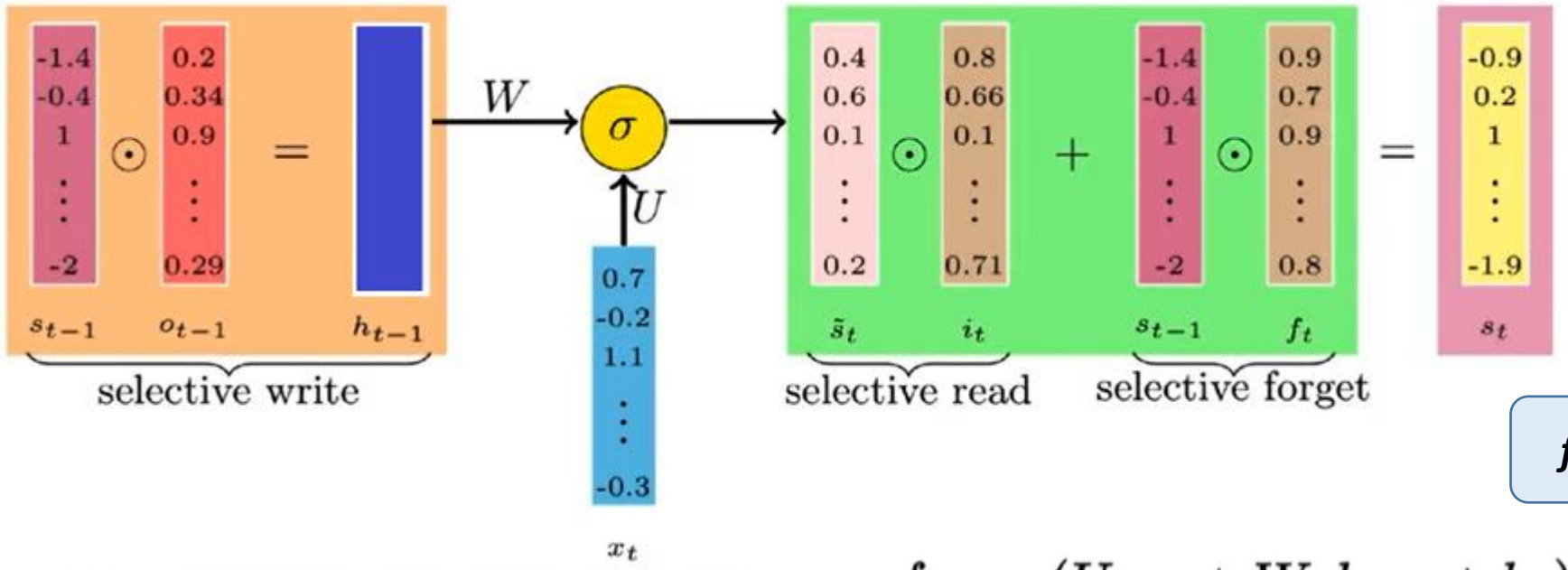
Forget gate

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

Selective Forget

Some LSTMs stop after selective read but the actual version LSTM has forget gate also

Want to make s_t depended on s_{t-1} but only the relevant portion of s_{t-1} (so forgetting some info of s_{t-1} and adding that to $\tilde{s}_t \odot i_t$)



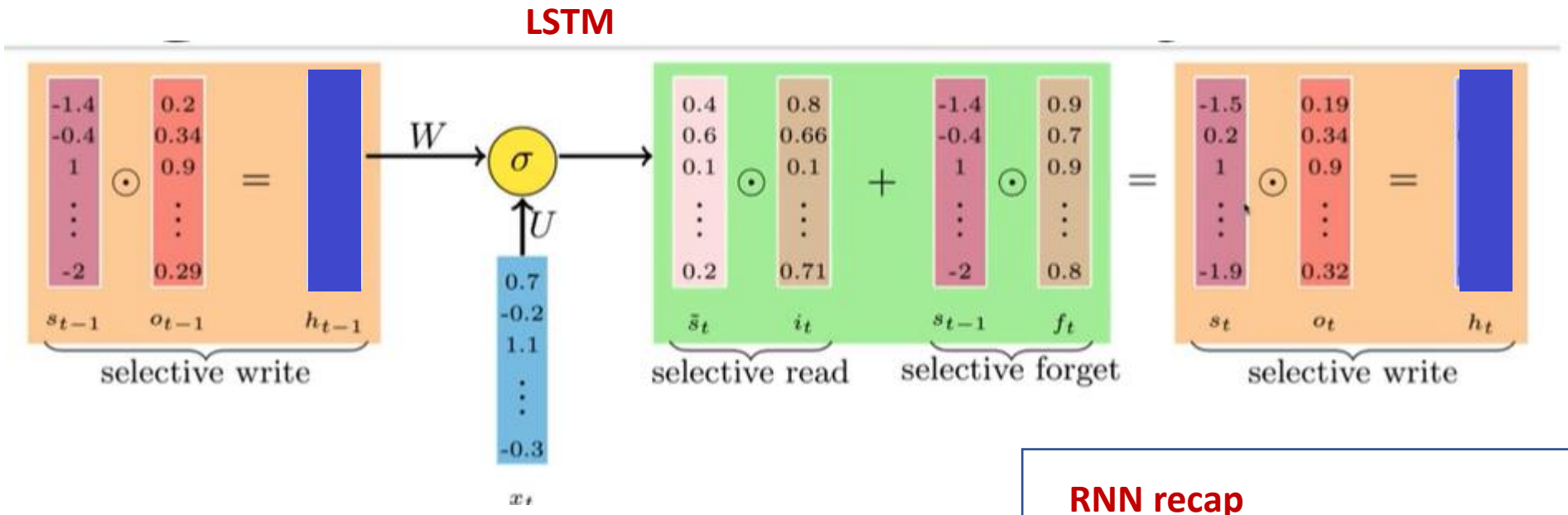
***ft* is called forget gate**

How do we combine $\tilde{s}_t$ and s_{t-1} to get the new state s_t

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$s_t = \tilde{s}_t \odot i_t + s_{t-1} \odot f_t$$

Summary-LSTM



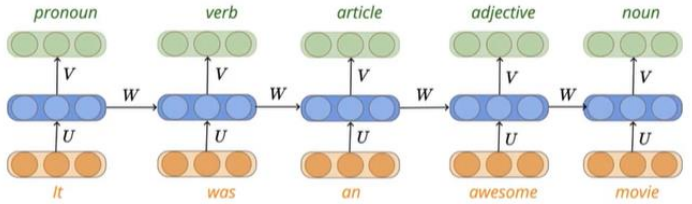
Gates:

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

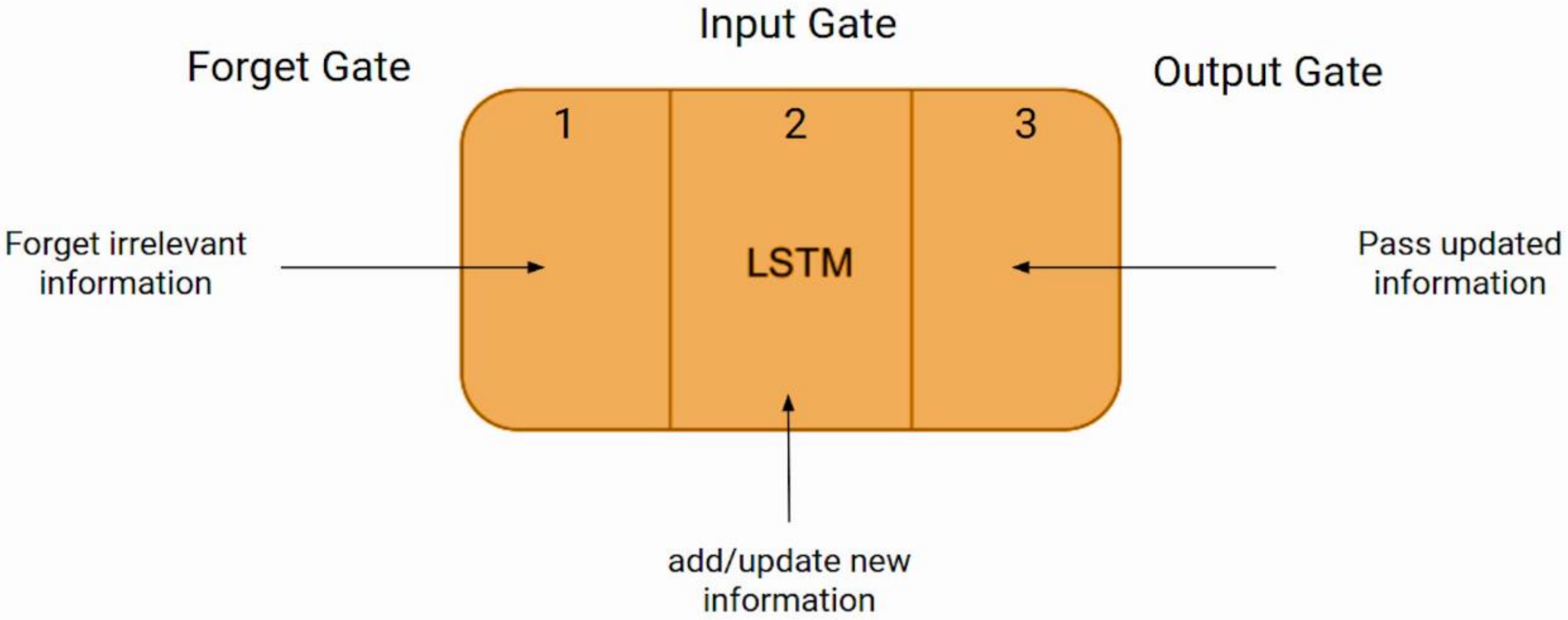
States:

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

RNN recap



$$s_t = \sigma(U x_t + W s_{t-1} + b)$$


Input Gate
Forget Gate
Output Gate
1
2
3
Forget irrelevant information
LSTM
Pass updated information
add/update new information

- **Gated Recurrent Unit (GRU)**

LSTM variants

- LSTM has many variants which include different number of gates and also different arrangement of gates
- The one which we just saw is one of the most popular variants of LSTM
- Another equally popular variant of LSTM is Gated Recurrent Unit which we will see next

Gates:

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States:

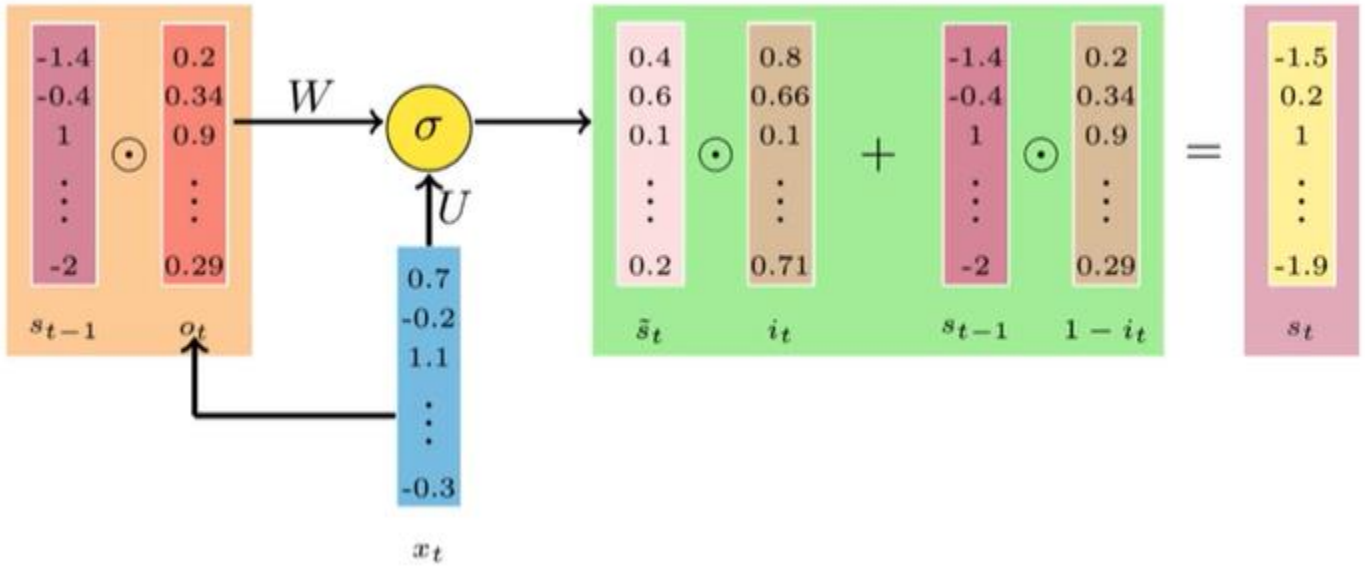
$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

Gated Recurrent Units (GRU)

Recap LSTM

States:

$$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$



Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + Ux_t + b)$$
$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$

Encoder- Decoder Models

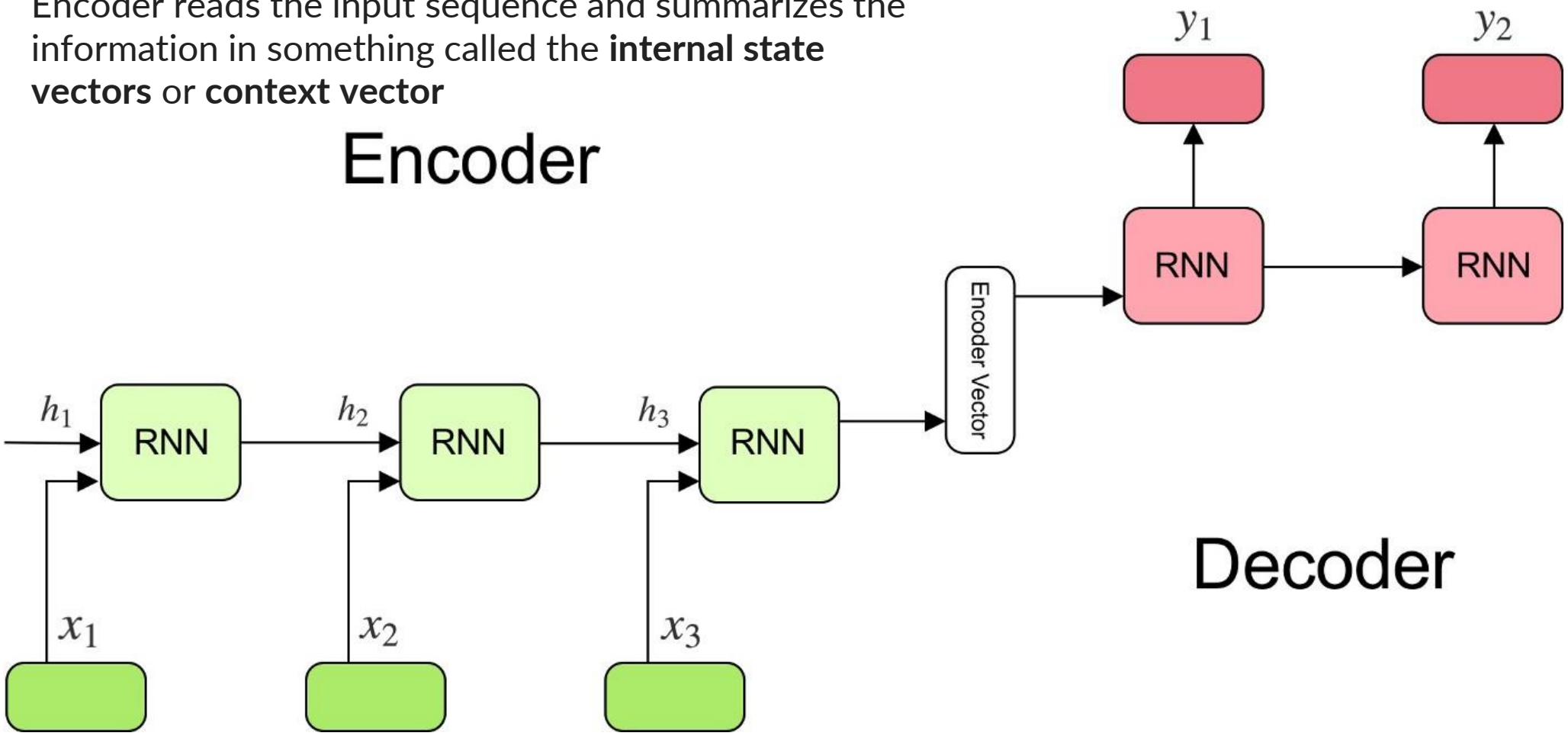
Encoder- Decoder Architecture

The encoder-decoder architecture **can handle inputs and outputs that are both variable-length sequences**, thus is suitable for sequence transduction problems such as machine translation. The encoder takes a variable-length sequence as the input and transforms it into a state with a fixed shape.

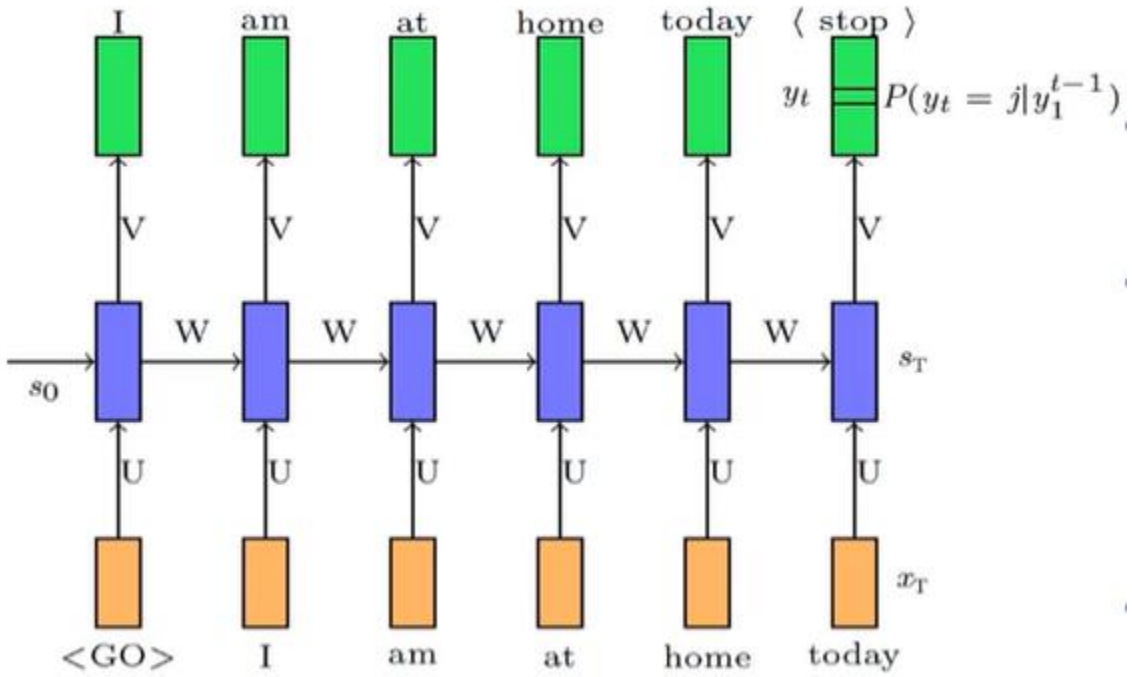
Machine translation is a major problem domain for sequence transduction models, whose input and output are both variable-length sequences. To handle this type of inputs and outputs, we can design an architecture with two major components. The first component is an *encoder*: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape. The second component is a *decoder*: it maps the encoded state of a fixed shape to a variable-length sequence. This is called an *encoder-decoder* architecture,

Encoder- decoder

Encoder reads the input sequence and summarizes the information in something called the **internal state vectors** or **context vector**



Recap – RNN (Autofill words)

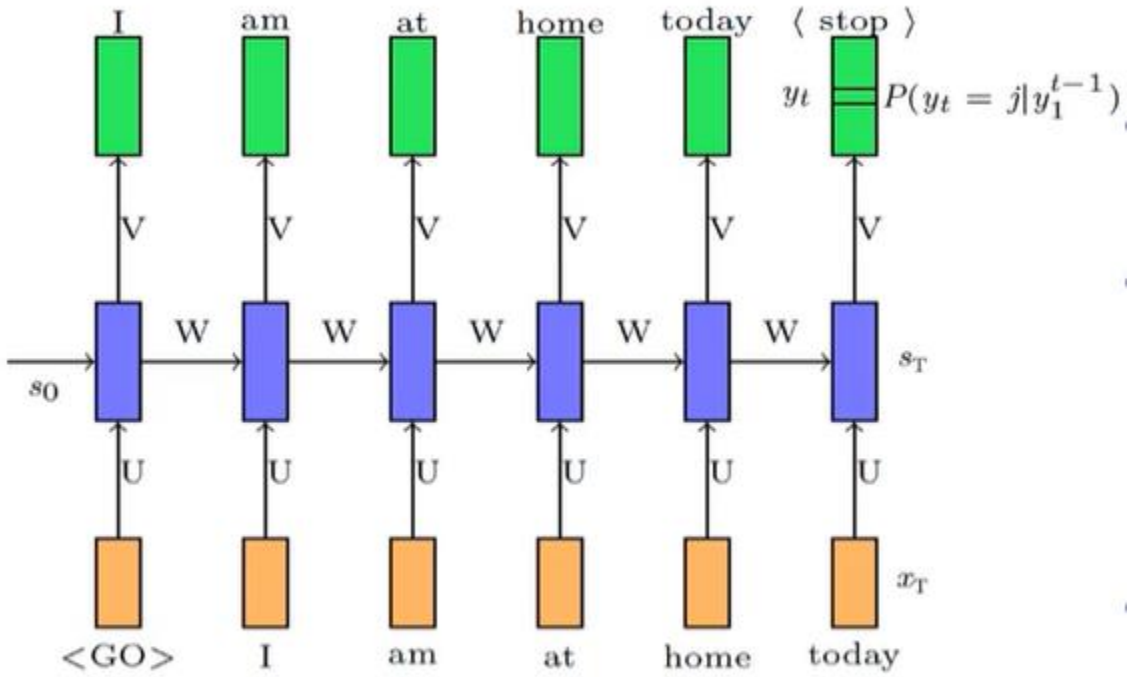


- We will start by revisiting the problem of language modeling
- Informally, given '$t-i$' words we are interested in predicting the t^{th} word
- More formally, given $y_1, y_2, \ldots, y_{t-1}$ we want to find

$$y^* = argmax\ P(y_t|y_1, y_2, \ldots, y_{t-1})$$

- Let us see how we model $P(y_t|y_1, y_2 \ldots y_{t-1})$ using a RNN
- We will refer to $P(y_t|y_1, y_2 \ldots y_{t-1})$ by shorthand notation: $P(y_t|y_1^{t-1})$

Recap - RNN



- We are interested in

$$P(y_t = j|y_1, y_2 \cdots y_{t-1})$$

where $j \in V$ and V is the set of all vocabulary words

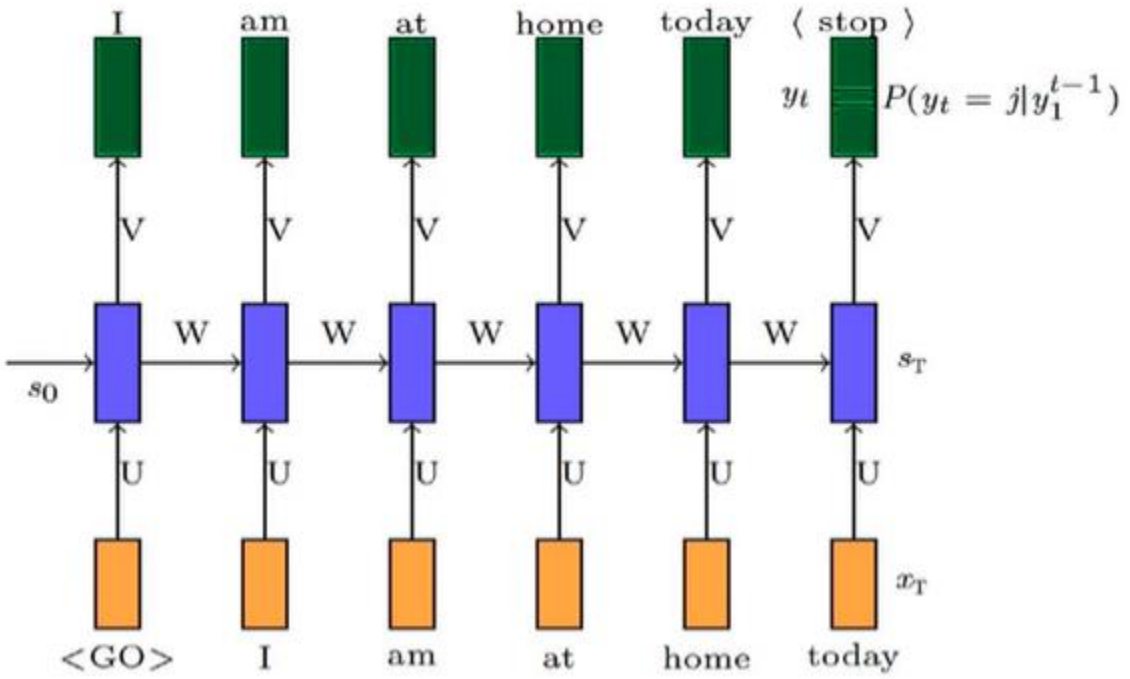
- Using an RNN we compute this as

$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$

- In other words we compute

$$P(y_t = j|y_1^{t-1}) = P(y_t = j|s_t)$$
$$= softmax(Vs_t + c)_j$$

Recap - RNN



Data:

India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, ...

- **Data:** All sentences from any large corpus (say wikipedia)
- **Model:**

$$s_t = \sigma(Ws_{t-1} + Ux_t + b)$$

$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$

- **Parameters:** U, V, W, b, c
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$

$$\mathscr{L}_t(\theta) = -\log P(y_t = \ell_t|y_1^{t-1})$$

$$s_t = \sigma(U\ x_t + W s_{t-1} + b)$$

$$s_t = \mathrm{RNN}(\ s_{t-1}, x_t)$$

$$\tilde{s}_t = \sigma(W(o_t \odot\ s_{t-1}) + U x_t + b)$$

$$s_t = i_t \odot\ s_{t-1} + (1 - i_t) \odot \tilde{s}_t$$

$$s_t = \mathrm{GRU}(\ s_{t-1}, x_t)$$

$$\tilde{s}_t = \sigma(W\ h_{t-1} + U x_t + b)$$

$$s_t = f_t \odot\ s_{t-1} + i_t \odot \tilde{s}_t$$

$$h_t = o_t \odot \sigma(s_t)$$

$$h_t, s_t = \mathrm{LSTM}(\ h_{t-1}, s_{t-1}, x_t)$$

Encoder – Decoder for Image Captioning

Image Captioning Problem

- So far we have seen how to model the conditional probability distribution $P(y_t|y_1^{t-1})$
- More informally, we have seen how to generate a sentence given previous words
- What if we want to generate a sentence given an image?

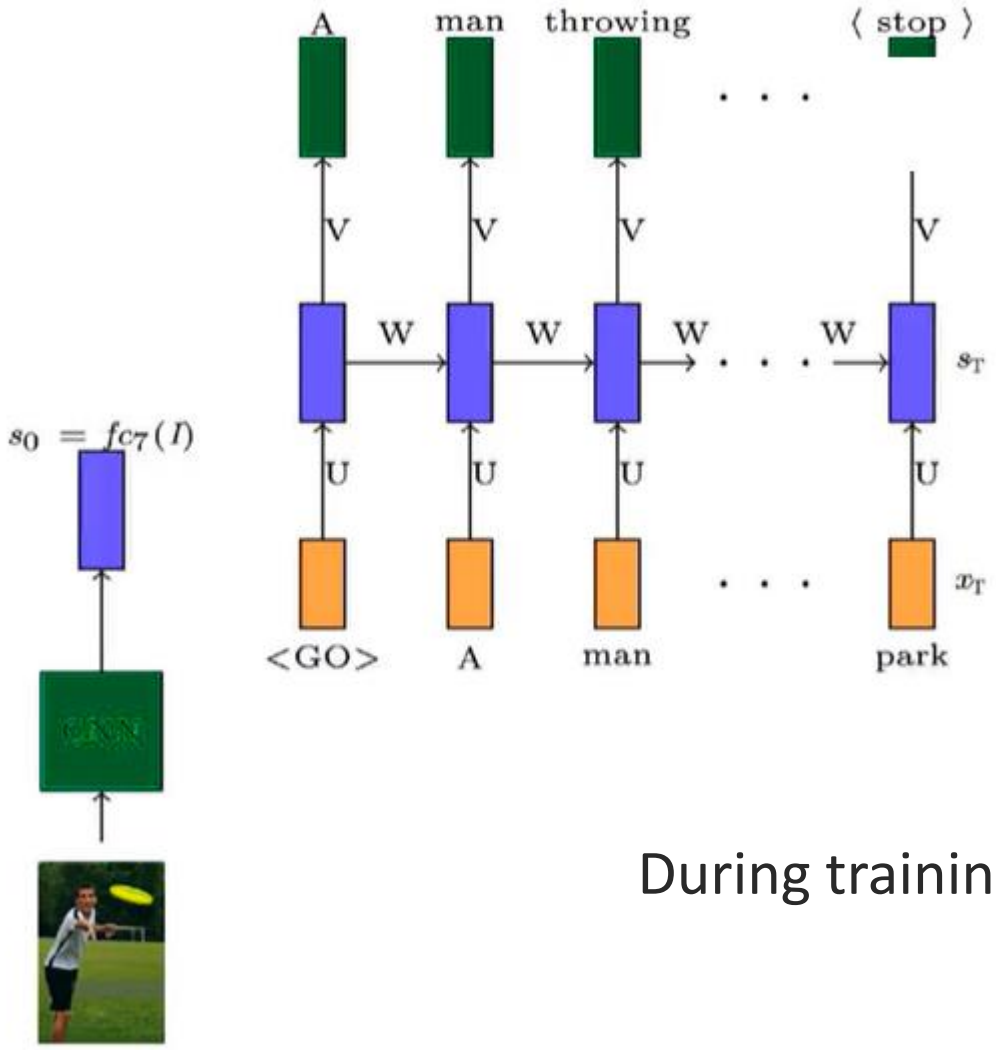
A man throwing
a frisbee in a park

- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c_7}(I))$

Encoder – Decoder for Image Captioning



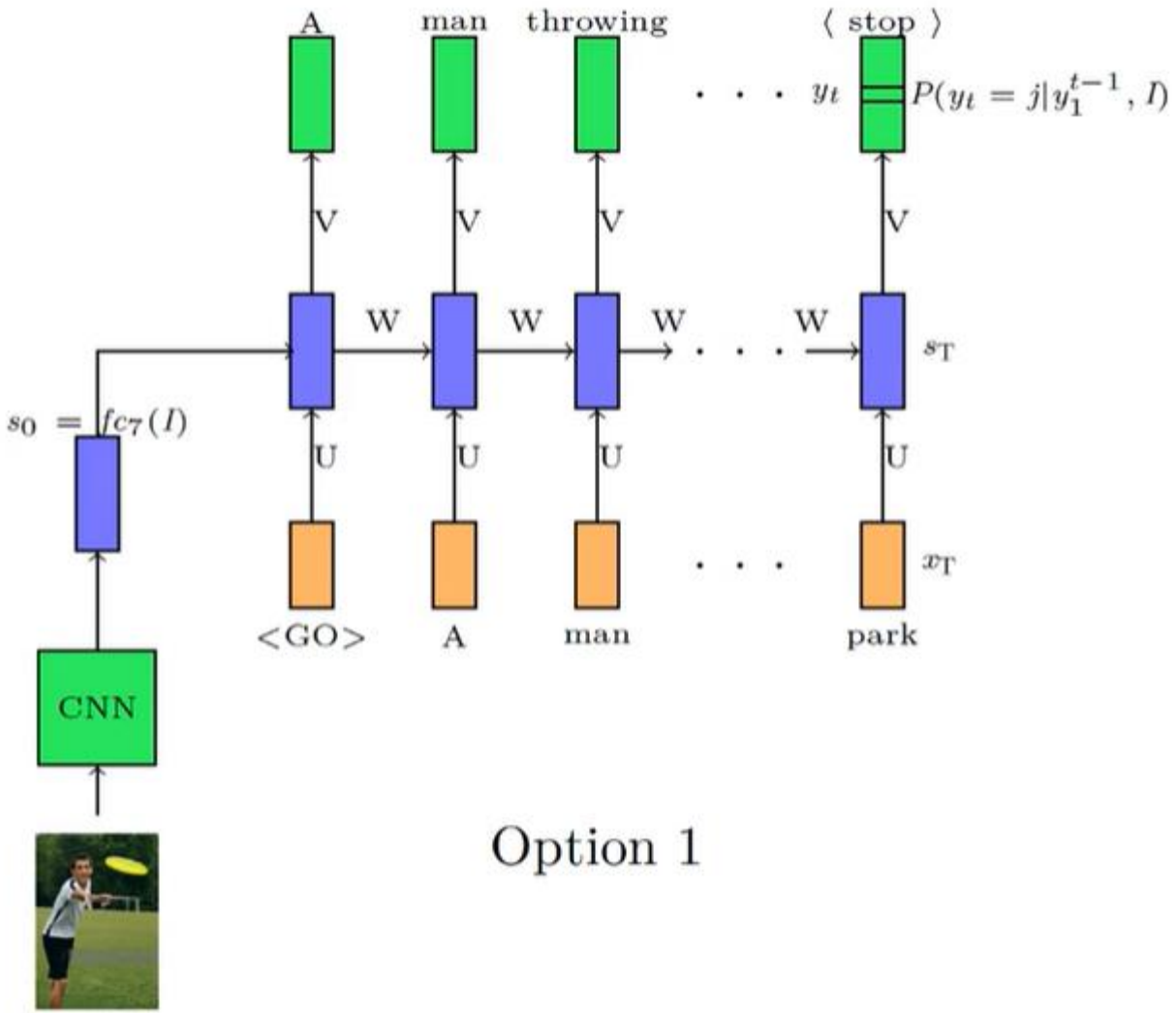
- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c_7}(I))$

During training phase teacher forcing technique is used

Encoder – Decoder for Image Captioning



Option 1

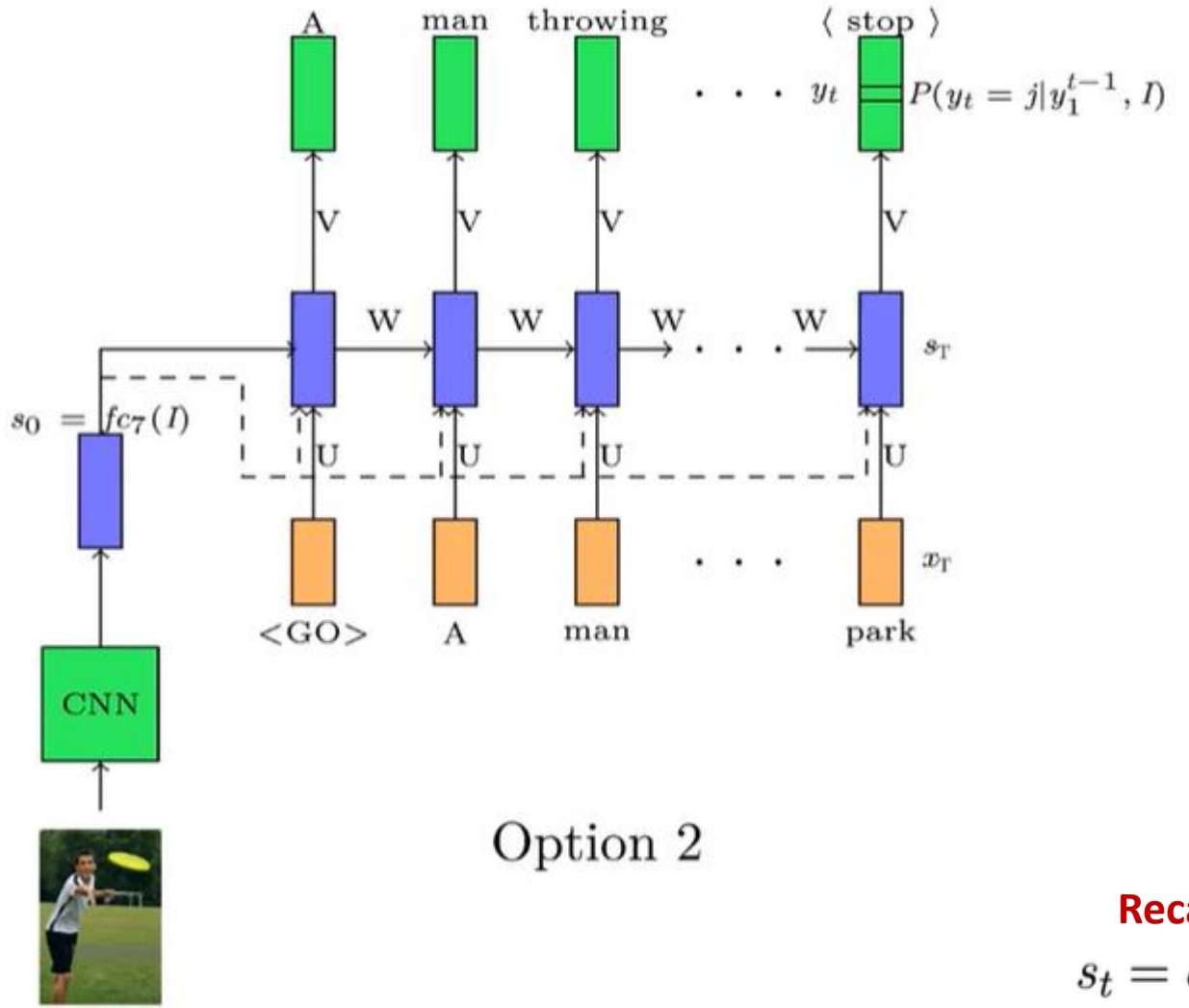
- **Option 1:** Set $s_0 = f_{c_7}(I)$
- Now s_0 and hence all subsequent s_t's depend on $f_{c_7}(I)$

$$S_1 = \sigma(W_{S0} + U_{x1} + b)$$

S0 computed from CNN, x1 <GO>

$$S_2 = \sigma(W_{S1} + U_{x2} + b)$$

Encoder – Decoder for Image Captioning



Option 2

- **Option 2:** Another more explicit way of doing this is to compute

$$s_t = RNN(s_{t-1}, [x_t, f_{c_7}(I)])$$

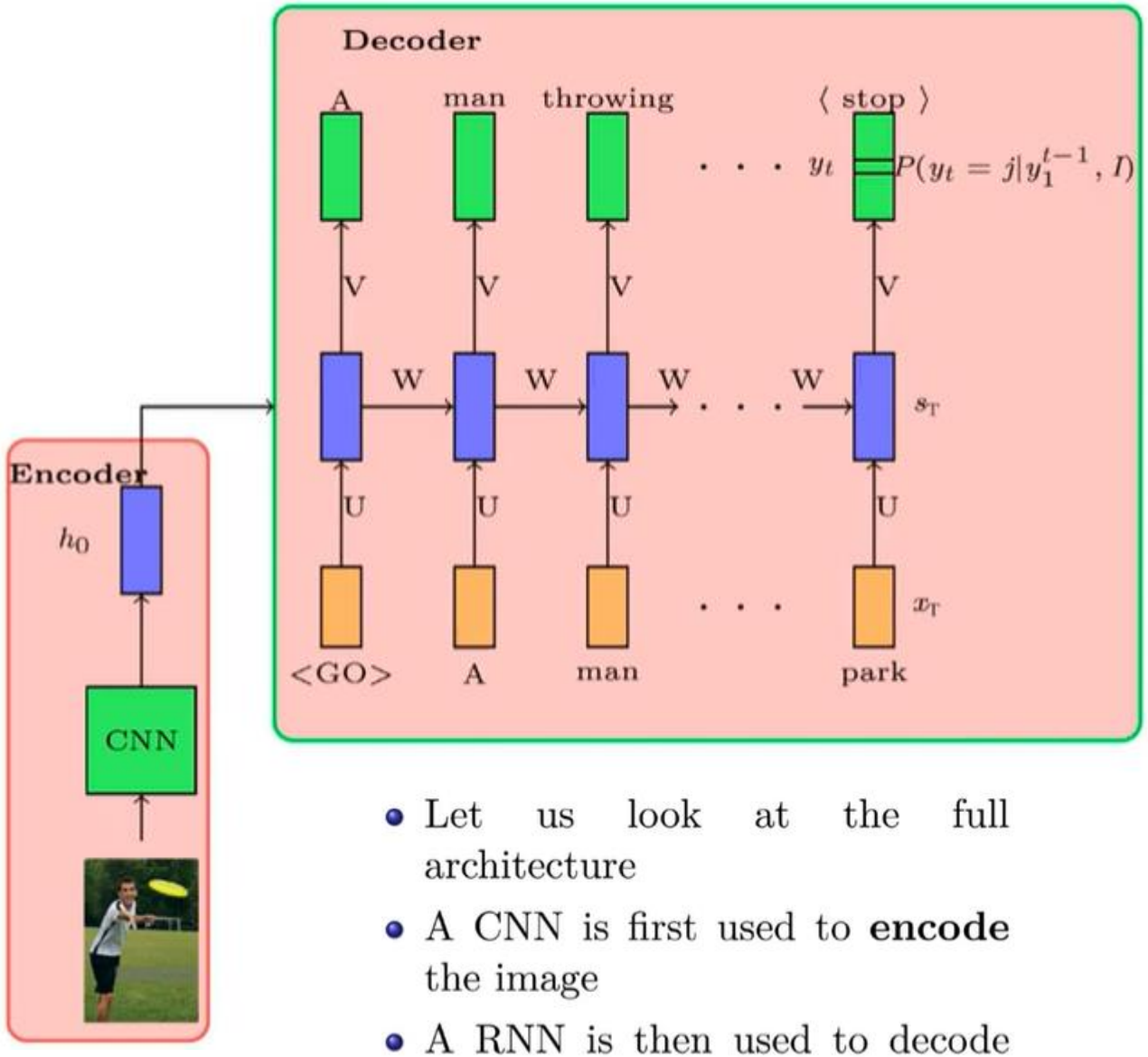
- In other words we are explicitly using $f_{c_7}(I)$ to compute s_t and hence $P(y_t = j)$

$$S_2 = \sigma\left(W_{S1} + U_{[x,1s1]} + b\right)$$

Concatenate x1 and S1

Recap

$$s_t = \sigma(U\, x_t + W s_{t-1} + b) \quad s_t = \mathrm{RNN}(s_{t-1}, x_t)$$



- Let us look at the full architecture
- A CNN is first used to **encode** the image
- A RNN is then used to decode (generate) a sentence from the encoding

- **Task:** Image captioning
- **Data:** $\{x_i = image_i,\ y_i = caption_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**

$$s_0 = CNN(x_i)$$

  - **Decoder:**

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

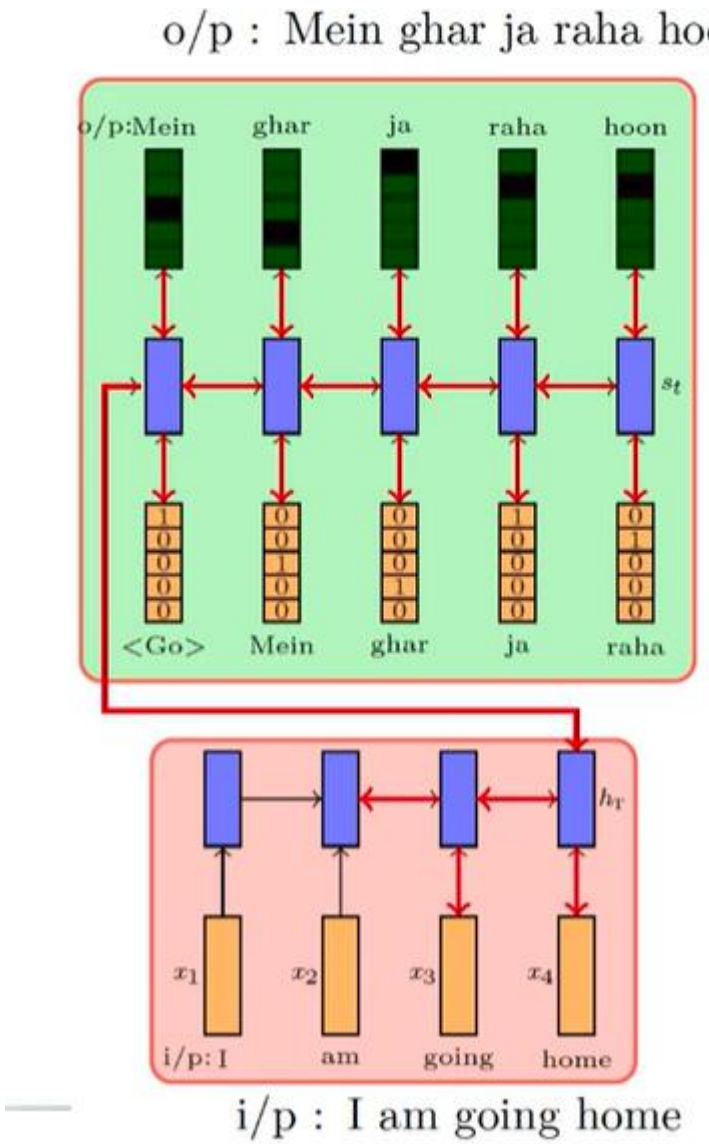
$$P(y_t|y_1^{t-1}, I) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ W_{conv}, b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) \quad = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, I)$$

- **Algorithm:** Gradient descent with backpropagation

Encoder – Decoder for Machine Translation(seq2seq)



- **Task:** Machine translation
- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

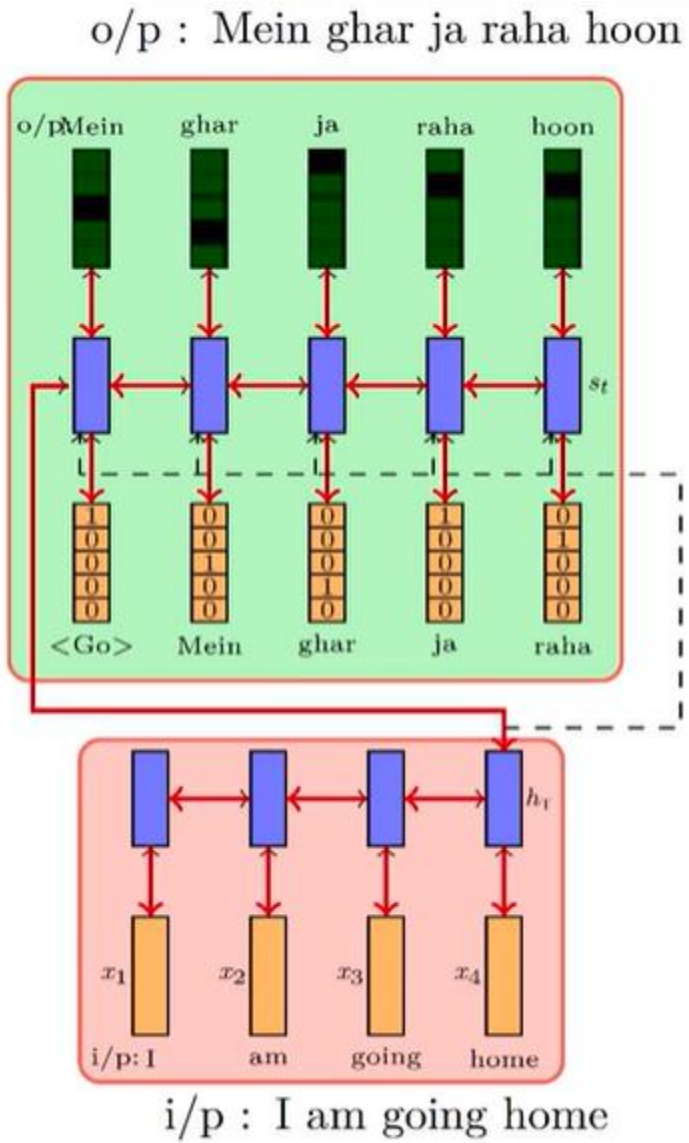
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc}, b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)$$

- **Algorithm:** Gradient descent with backpropagation

Encoder – Decoder for Machine Translation



- **Task:** Machine translation
- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Model (Option 2):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, [h_T, e(\hat{y}_{t-1})])$$

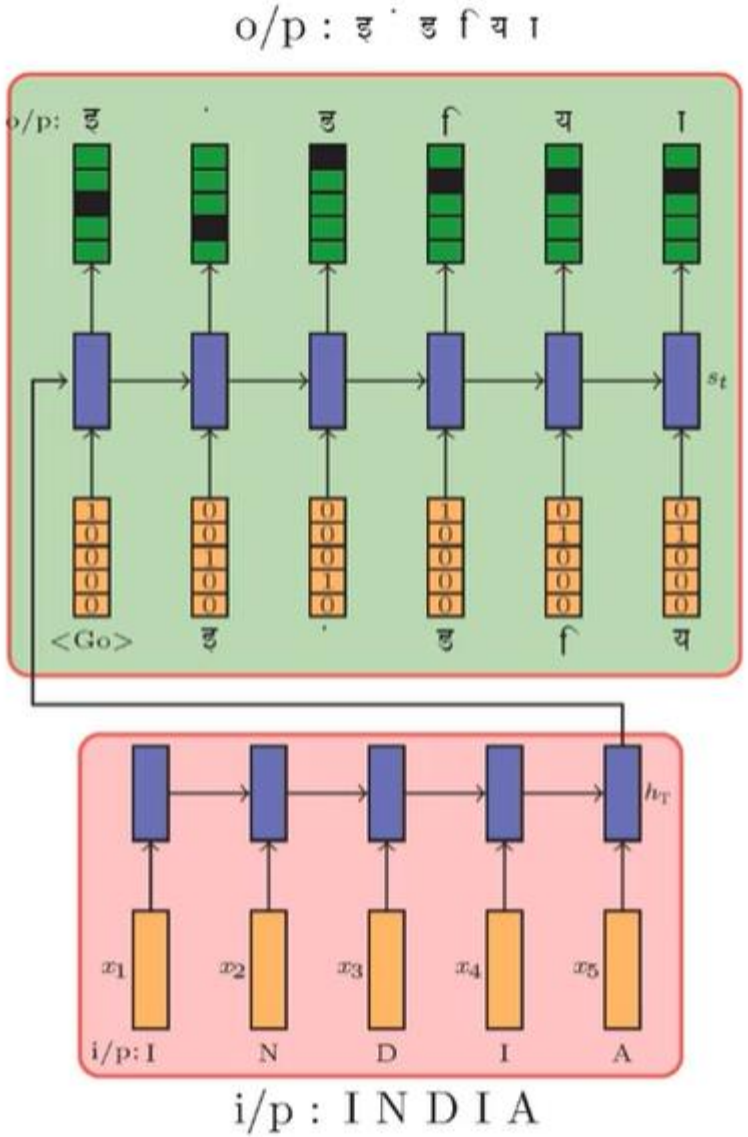
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc}, b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)$$

- **Algorithm:** Gradient descent with backpropagation

Encoder – Decoder for Machine Transliteration



- **Task:** Transliteration
- **Data:** $\{x_i = srcword_i,\ y_i = tgtword_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

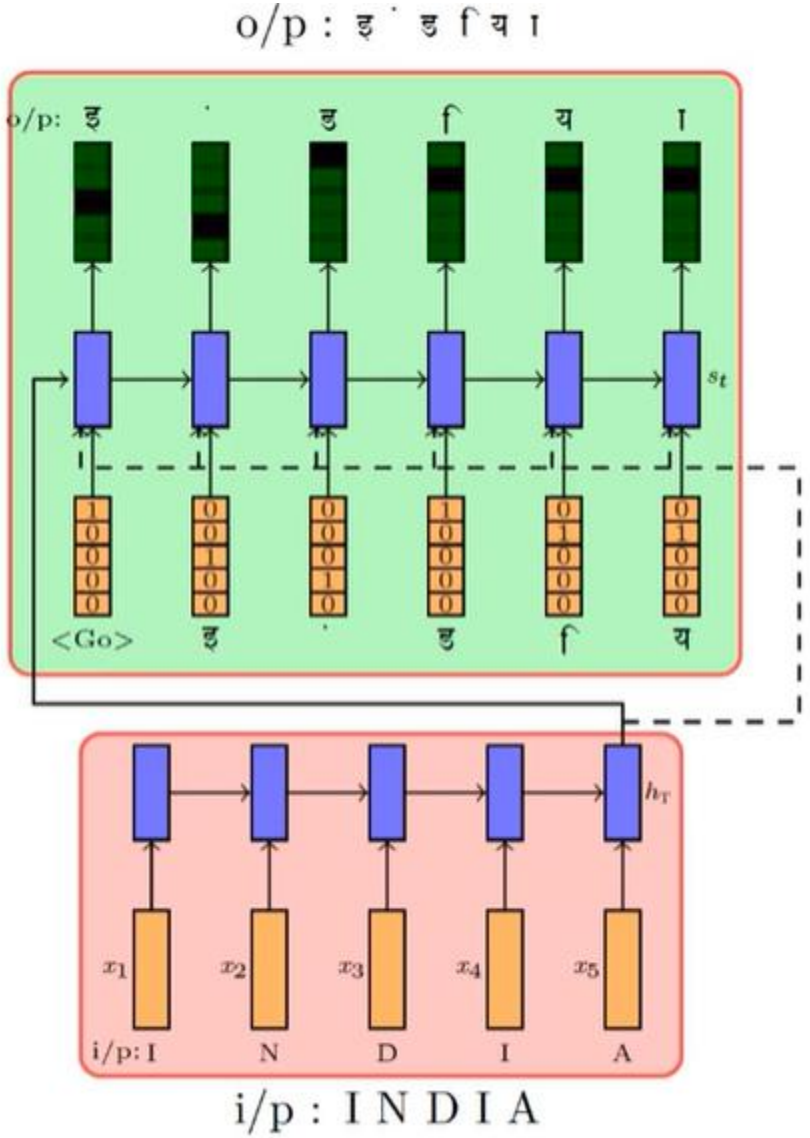
$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)$$

Encoder – Decoder for Machine Transliteration



- **Task:** Transliteration
- **Data:** $\{x_i = srcword_i,\ y_i = tgtword_i\}_{i=1}^{N}$
- **Model (Option 2):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), h_T])$$

$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc}, b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)$$

- **Algorithm:** Gradient descent with backpropagation

Attention Mechanism in Deep Learning

What is Attention?

- In psychology
  - Attention is the cognitive process of selectively concentrating on one or a few things while ignoring others

How many people in the photo?

Who is the teacher ?

pork belly = delicious . || scallops? || I don't even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Examples of attention modeling in sentiment classification

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Attention Mechanism in Deep Learning

- The introduction of the Attention Mechanism in deep learning has improved the success of various models in recent years,
- Continues to be an omnipresent component in state-of-the-art models.
- Therefore, it is vital that we pay Attention to Attention and how it goes about achieving its effectiveness.

Sequential Models : Evolution

RNN

LSTM

GRU

Encoder decoder

Attention Models

Attention based **Encoder Decoder** Models

Transformers- Self Attention Video processing/NLP

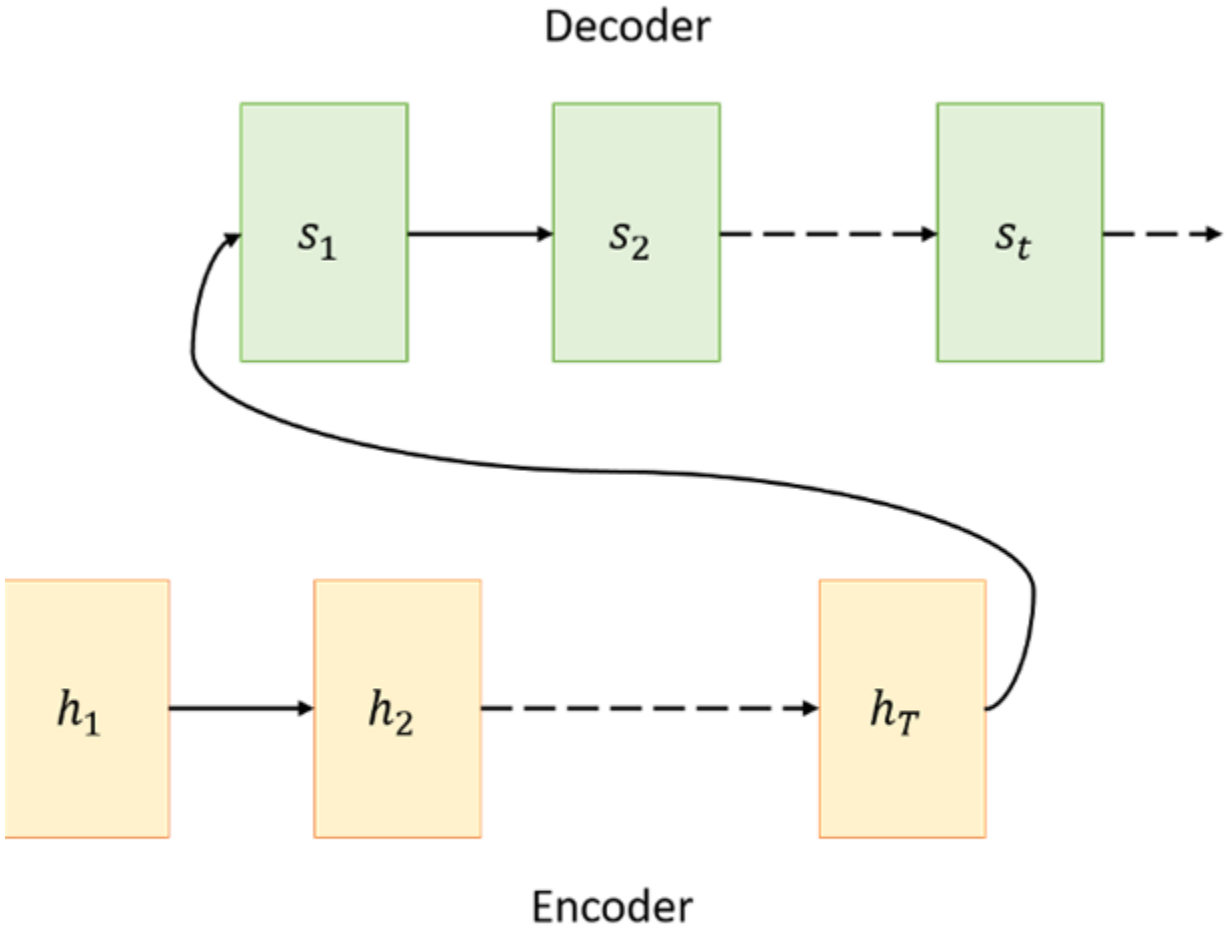
Memory Network Architectures Question Answering (QA) and chat bots

Graph Attention Networks (GAT) social networks, citation networks, protein-protein interactions,

Attention Mechanism in Encoder Decoder Models

Challenges of traditional encoder-decoder framework.

- The encoder has to compress all the input information into a single fixed length vector
- One vector to compress long and detailed input sequences may lead to loss of information
- it is unable to model alignment between input and output sequences,
- Decoder lacks any mechanism to selectively focus on relevant input tokens while generating each output token.



(a) Vanilla Encoder Decoder Architecture

- Dzmitry Bahdanau Jacobs University Bremen, Germany
- KyungHyun Cho Universit´e de Montr´eal Yoshua Bengio

ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

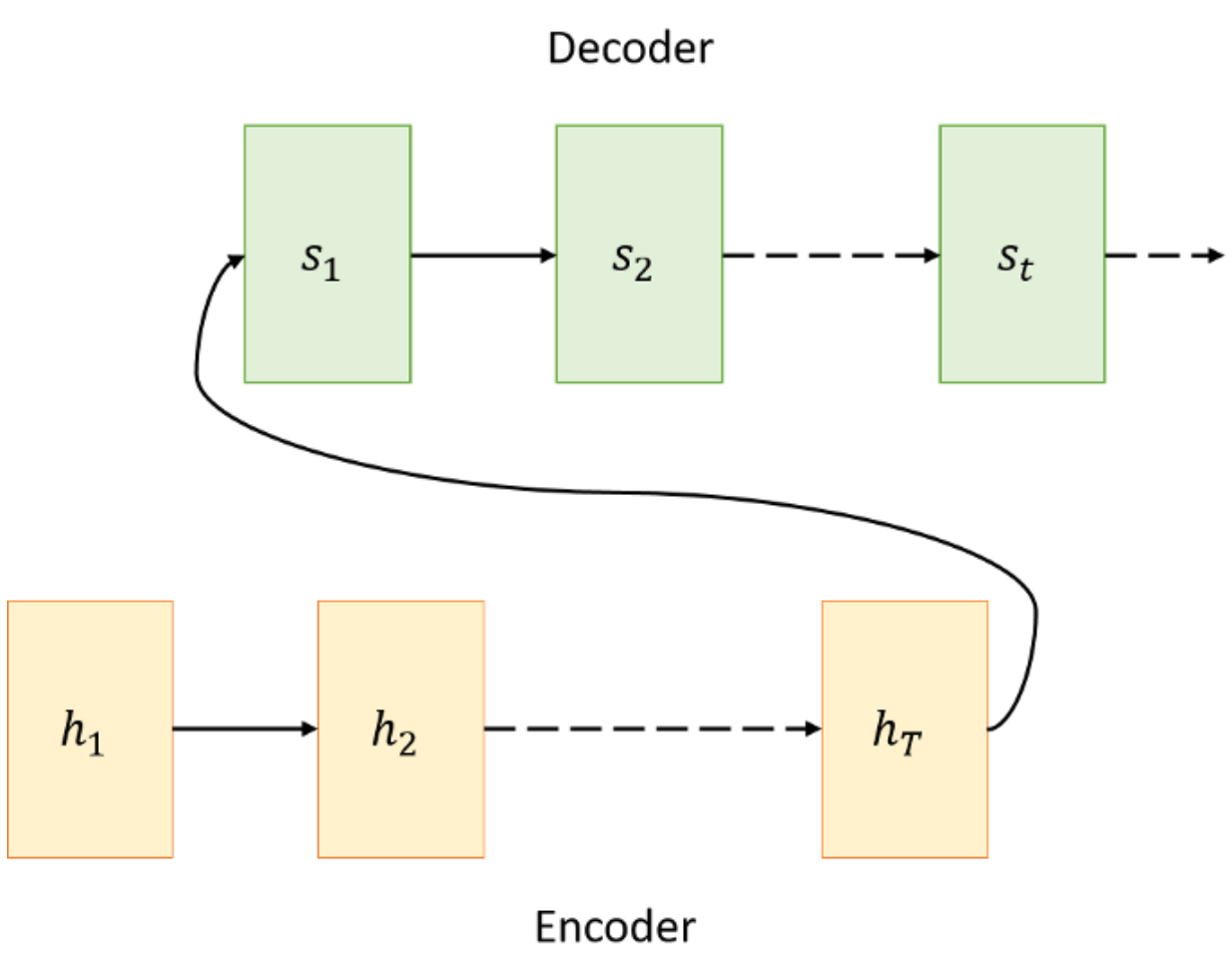
Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

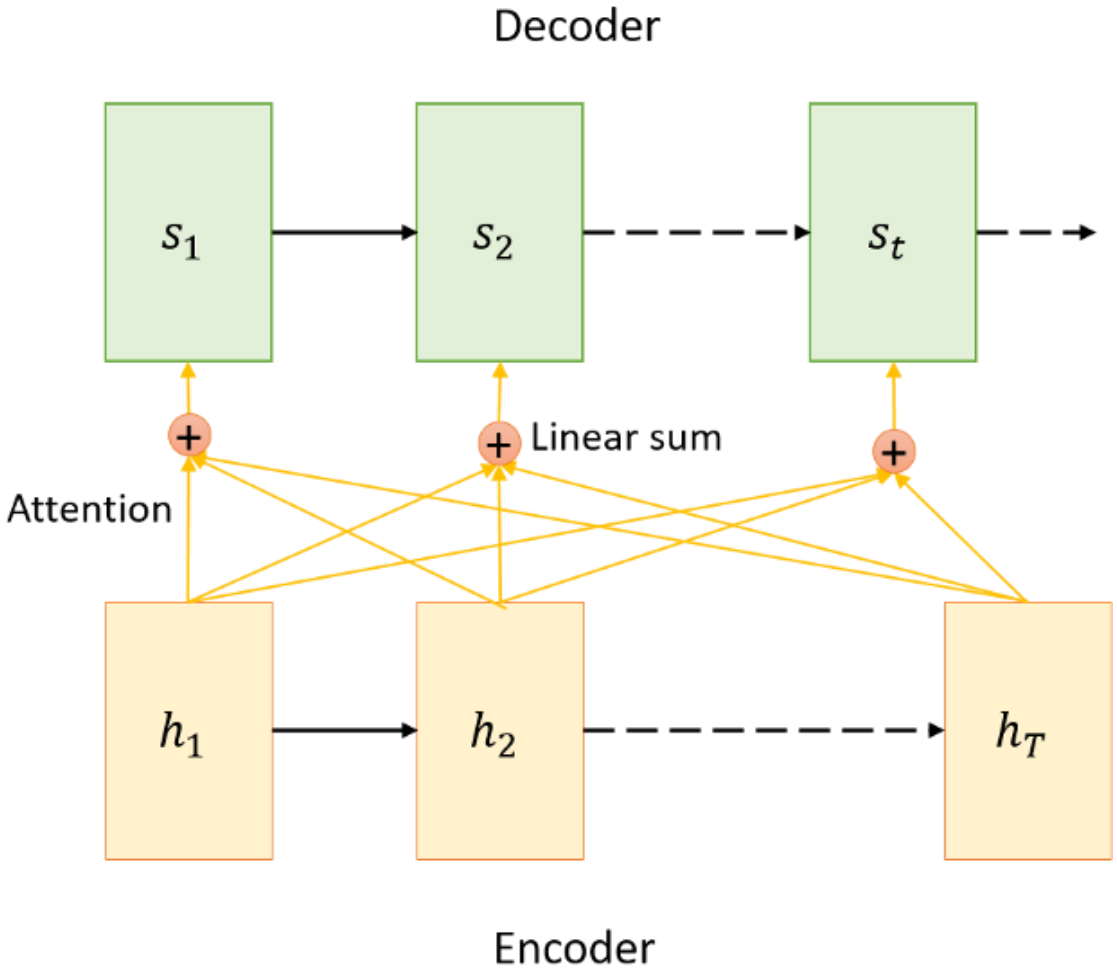
ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achiev -the-art phrase-ba ermore, qualitative analysis reveals that the (soft-)alignments found by the model agree

Attention mechanism in Encoder Decoder

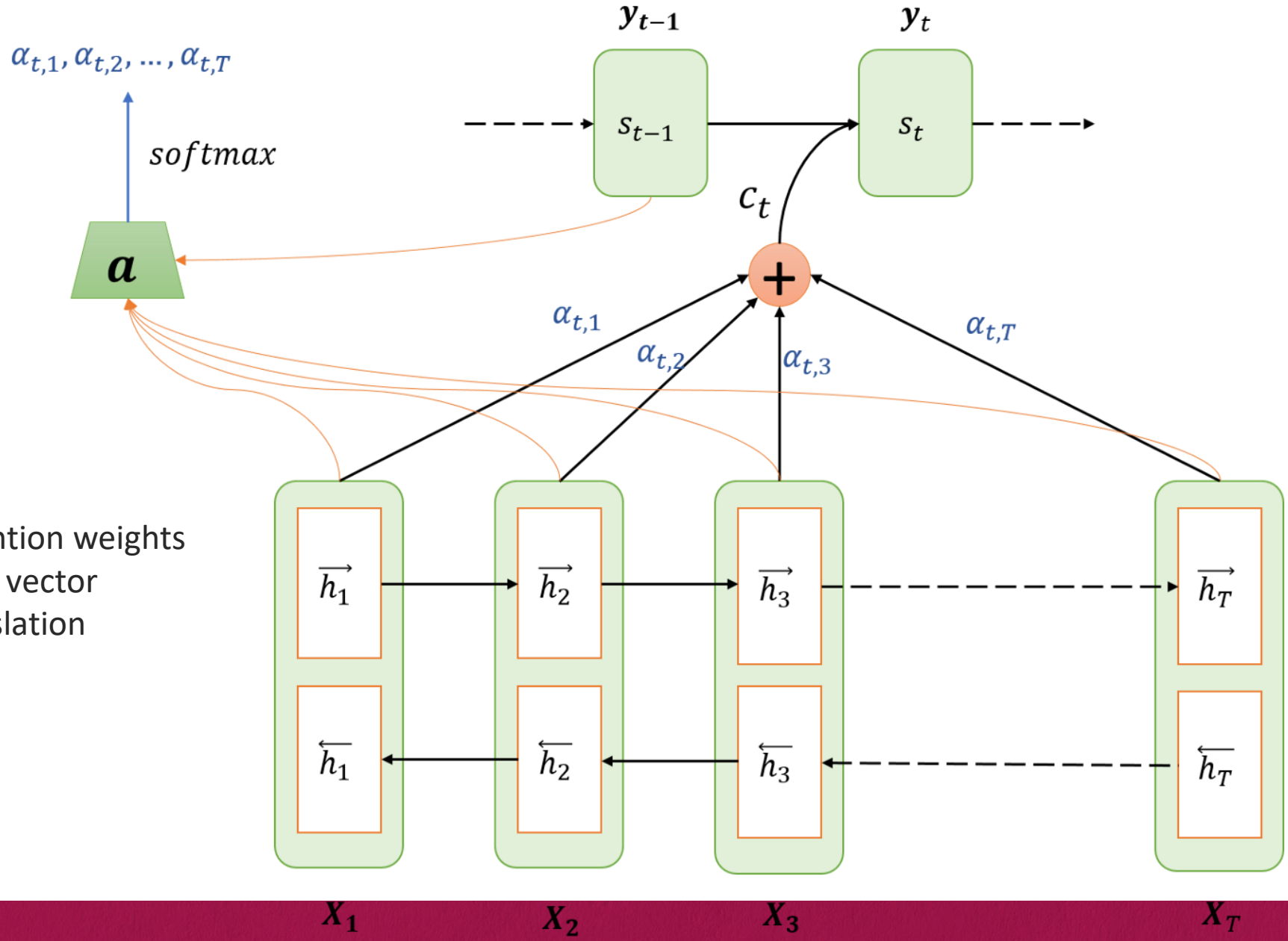


(a) Vanilla Encoder Decoder Architecture



(b) Attention Mechanism

Attention Mechanism



1.Encoding
2.Computing Attention weights
3.Creating context vector
4.Decoding / Translation

Attention Mechanism

o/p : I am going home
t_1 : [1 0 0 0 0]
t_2 : [0 0 0 0 1]
t_3 : [0 0 0.5 0.5 0]

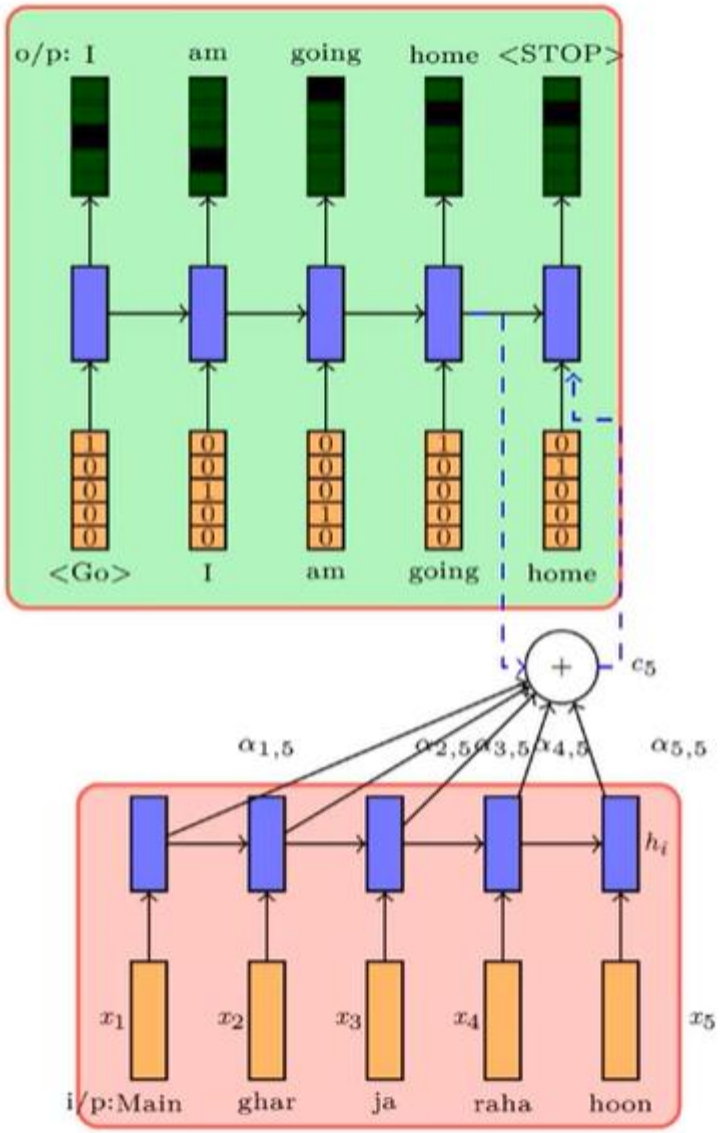
i/p : Main ghar ja raha hoon

o/p : I am going home
t_1 : [1 0 0 0 0]
t_2 : [0 0 0 0 1]
t_3 : [0 0 0.5 0.5 0]
t_4 : [0 1 0 0 0]

i/p : Main ghar ja raha hoon

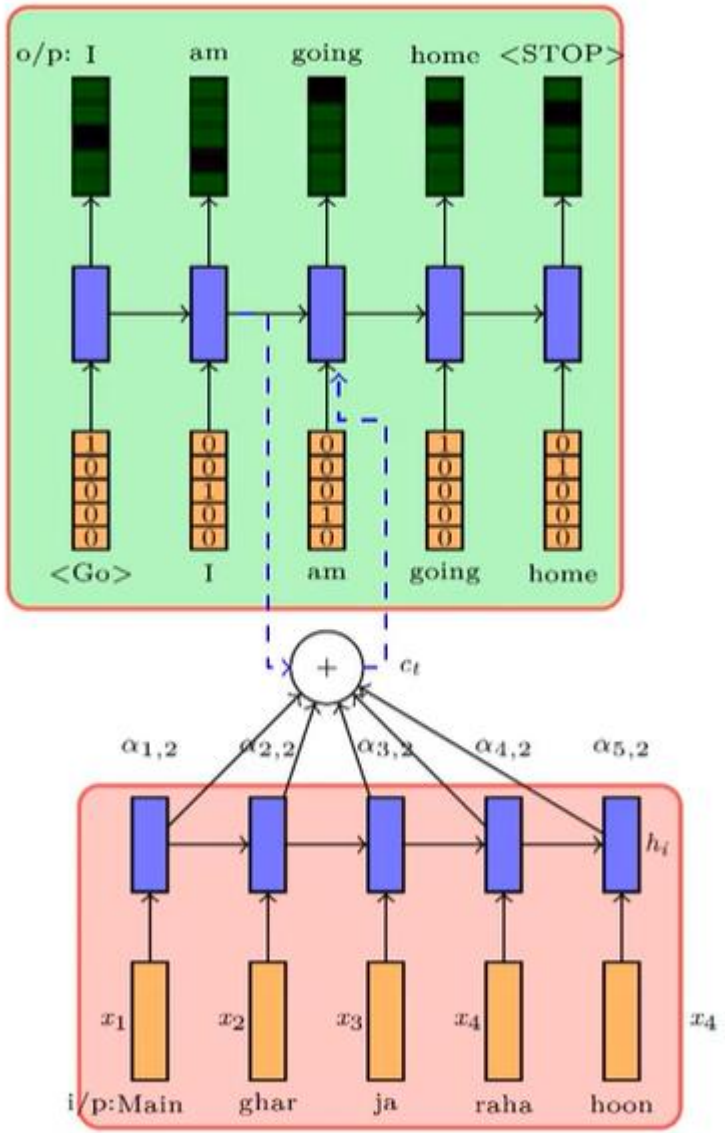
- Humans try to produce each word in the output by focusing only on certain words in the input
- Essentially at each time step we come up with a distribution on the input words
- This distribution tells us how much attention to pay to each input words at each time step
- Ideally, at each time-step we should feed only this relevant information (i.e. encodings of relevant words) to the decoder

Machine Translation with Attention Mechanism



- We could just take a weighted average of the corresponding word representations and feed it to the decoder
- For example at timestep 3, we can just take a weighted average of the representations of 'ja' and 'raha'
- Intuitively this should work better because we are not overloading the decoder with irrelevant information (about words that do not matter at this time step)
- How do we convert this intuition into a model ?

Machine Translation with Attention Mechanism



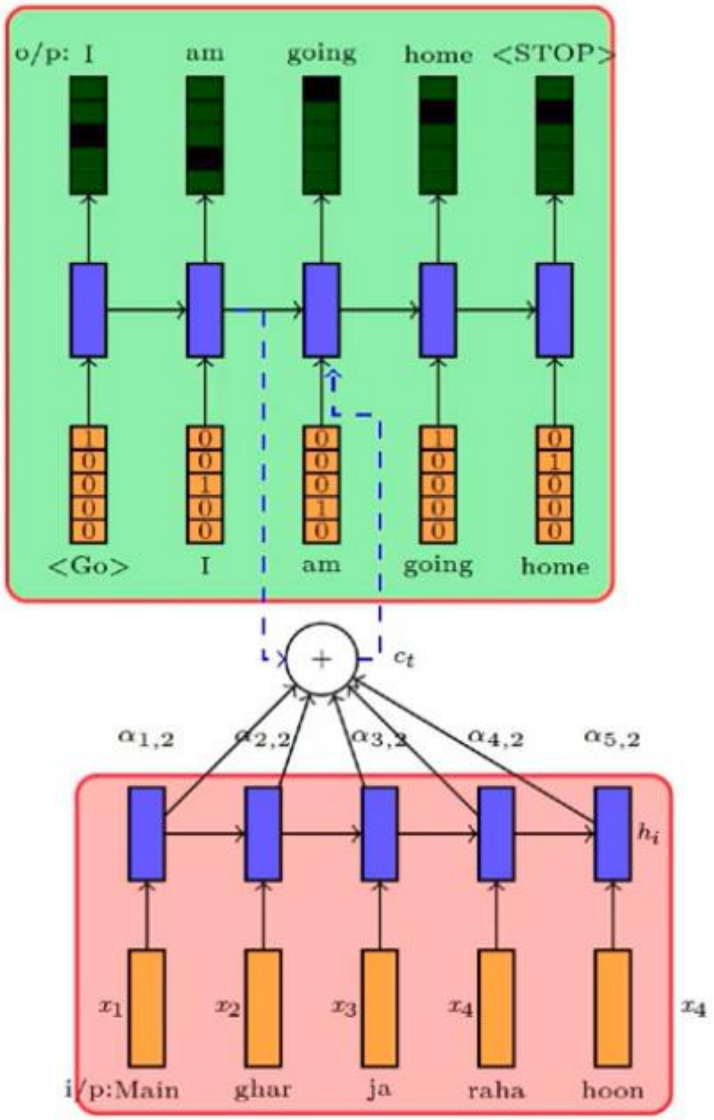
- Of course in practice we will not have this oracle
- The machine will have to learn this from the data
- To enable this we define a function

$$e_{jt} = f_{ATT}(s_{t-1}, h_j)$$

- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word (we will see the exact form of f_{ATT} later)
- We can normalize these weights by using the softmax function

$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})}$$

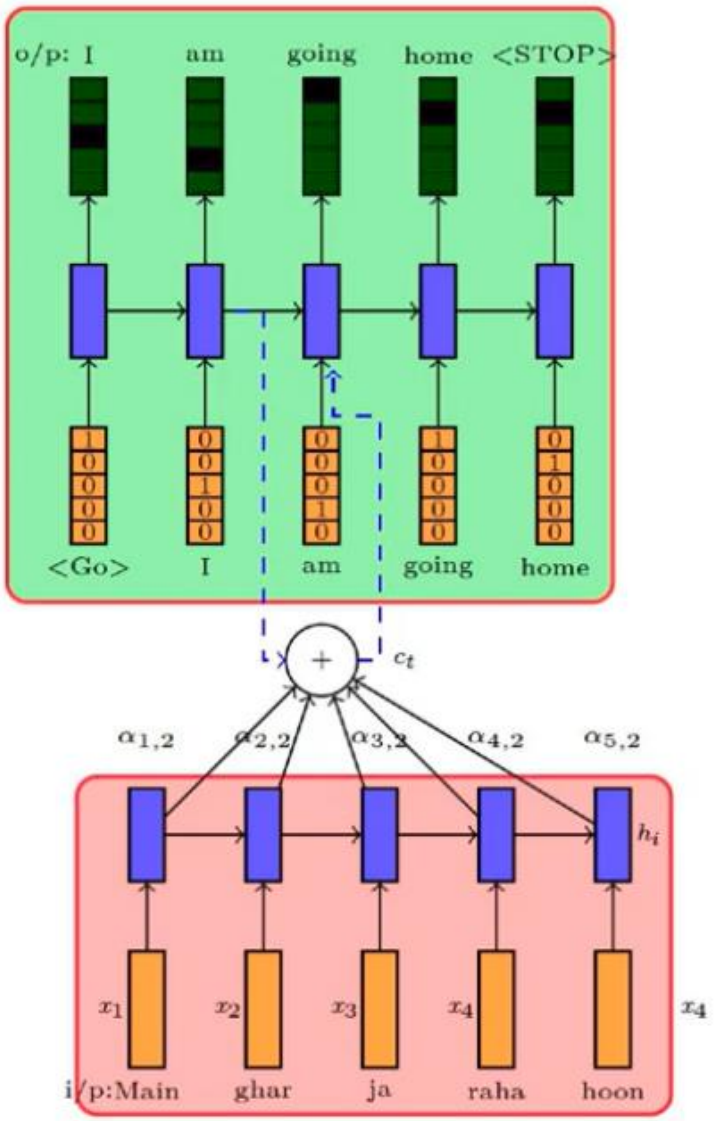
Machine Translation with Attention Mechanism



$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})} \qquad c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$

- α_{jt} denotes the probability of focusing on the j^{th} word to produce the t^{th} output word
- We are now trying to learn the α's instead of an oracle informing us about the α's

Machine Translation with Attention Mechanism

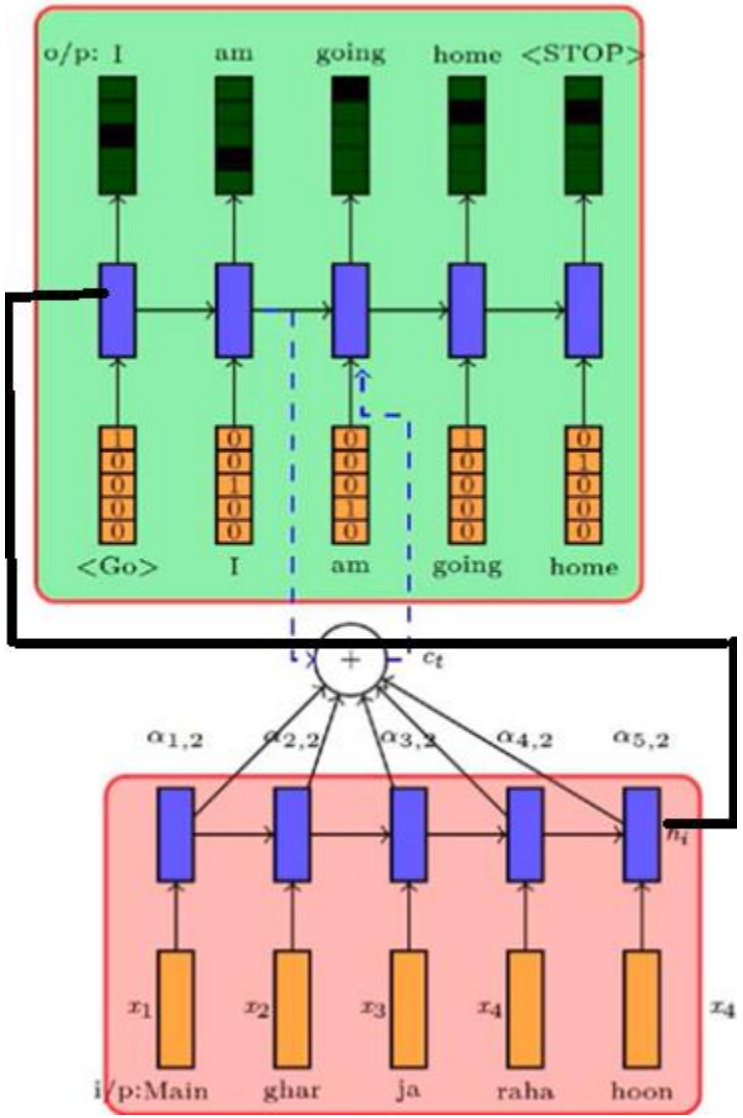


- From now on we will refer to the decoder RNN's state at the t-th timestep as s_t and the encoder RNN's state at the j-th time step as h_j
- Given these new notations, one (among many) possible choice for f_{ATT} is

$$e_{jt} = V_{att}^T \tanh(U_{att} s_{t-1} + W_{att} h_j)$$

- $V_{att} \in \mathbb{R}^d$, $U_{att} \in \mathbb{R}^{d \times d}$, $W_{att} \in \mathbb{R}^{d \times d}$ are additional parameters of the model
- These parameters will be learned along with the other parameters of the encoder and decoder

Machine Translation with Attention Mechanism



- **Task:** Machine Translation
- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Encoder:**

$$h_t = RNN(h_{t-1}, x_t)$$
$$s_0 = h_T$$

- **Decoder:**

$$e_{jt} = V_{attn}^T tanh(U_{attn} h_j + W_{attn} s_{t-1})$$
$$\alpha_{jt} = softmax(e_{jt})$$
$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$
$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$
$$\ell_t = softmax(Vs_t + b)$$

- **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b, U_{attn}, V_{attn}
- **Loss** and **Algorithm** remains same

Namah Shivaya