Recurrent Neural Networks (RNN)
&
Long Short Term Memory (LSTM)

Sequential data

- The order of inputs matters a lot in sequential data:
- “working love learning we on deep”
  - It does not make sense.
- “We love working on deep learning”
  - Now it has a meaning.

Most data are sequential

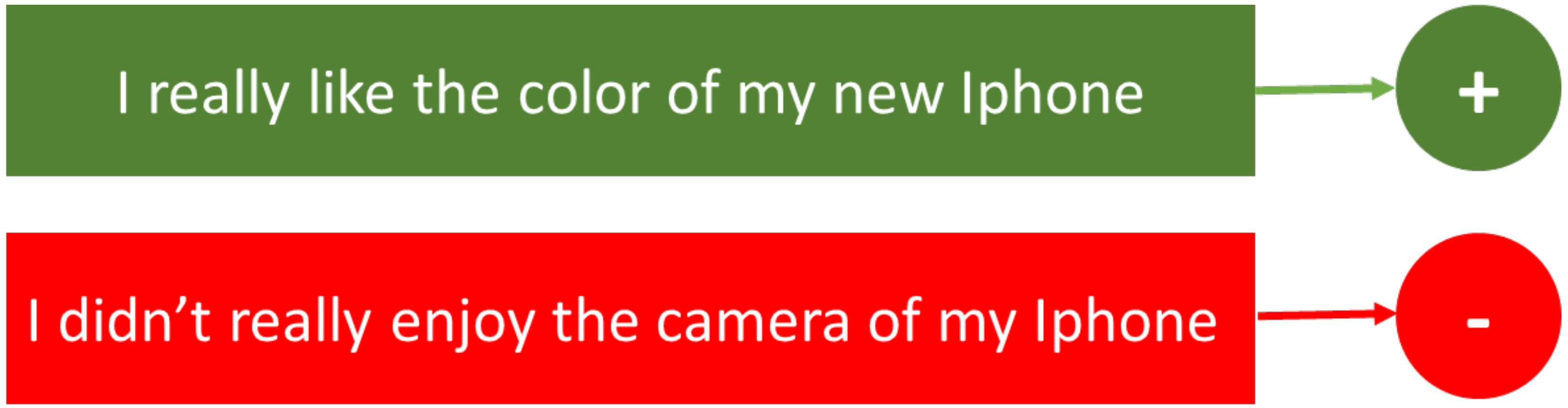
- In sequential data, the order of data bits determines the the event itself.

- Text → a sequence of words
- Video → a sequence of image frames
- Speech → a sequence of vocal data
- Genome → a sequence of genetic bases
- ,...

RNN applications

- The beauty of RNN is its wide range of applications.

- Sentiment Classification:


I really like the color of my new Iphone
+
I didn’t really enjoy the camera of my Iphone
-

RNN applications

- Image Captioning:

A person riding a motorcycle on a dirt road.

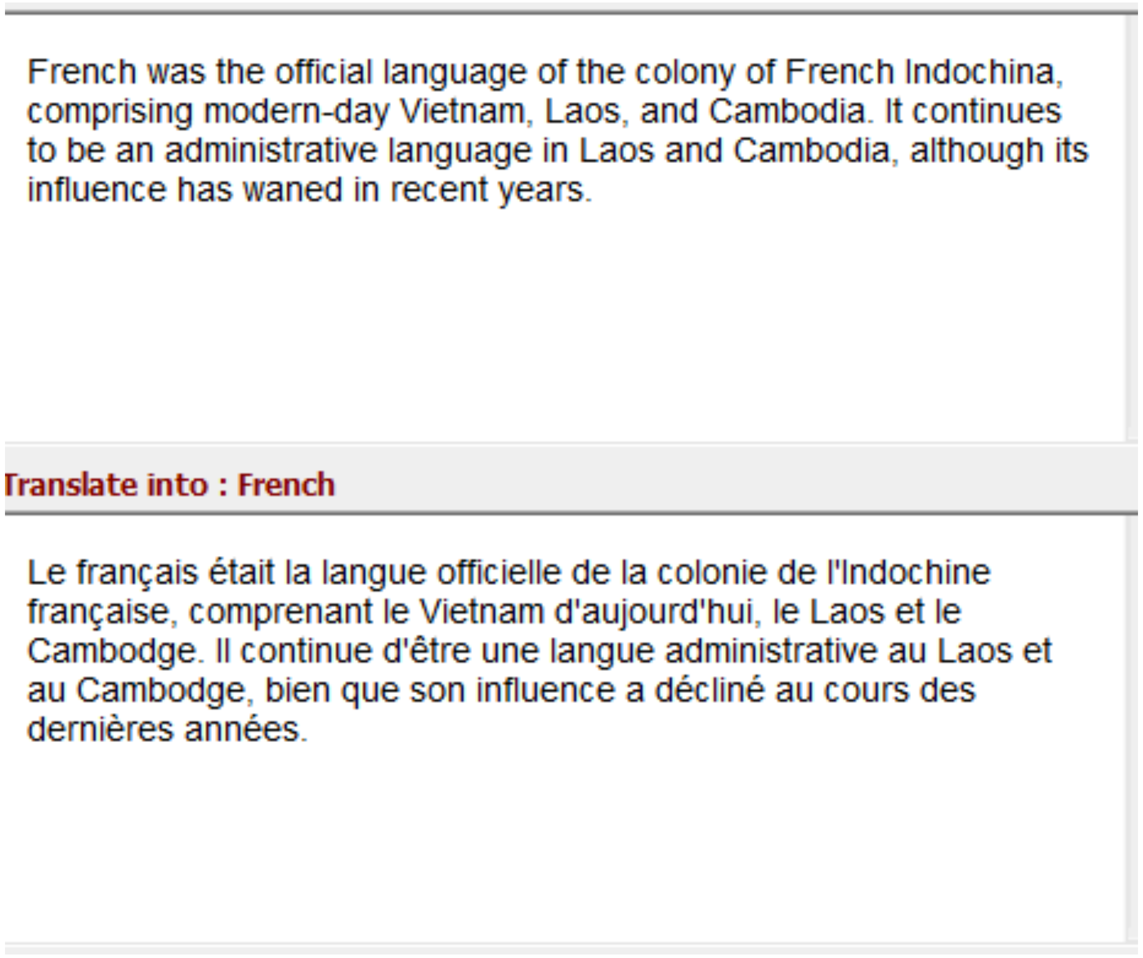
Two dogs play in the grass.

A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

RNN applications

- Language Translation:

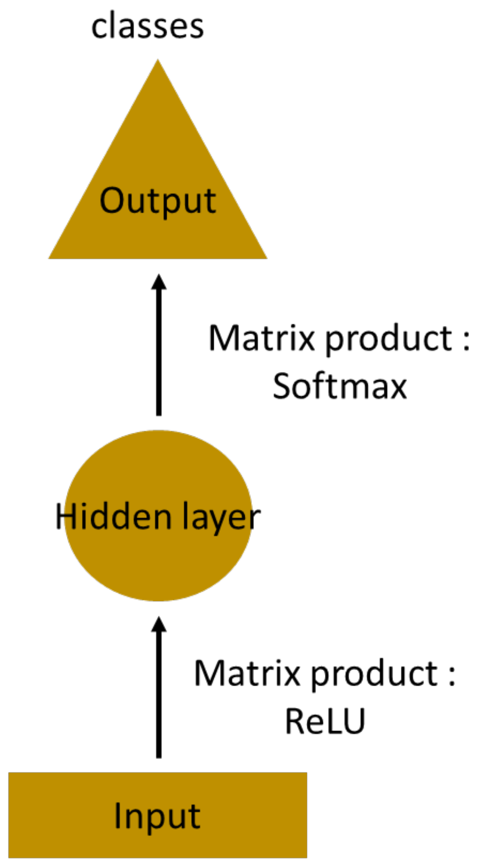

French was the official language of the colony of French Indochina, comprising modern-day Vietnam, Laos, and Cambodia. It continues to be an administrative language in Laos and Cambodia, although its influence has waned in recent years.

Translate into : French

Le français était la langue officielle de la colonie de l'Indochine française, comprenant le Vietnam d'aujourd'hui, le Laos et le Cambodge. Il continue d'être une langue administrative au Laos et au Cambodge, bien que son influence a décliné au cours des dernières années.

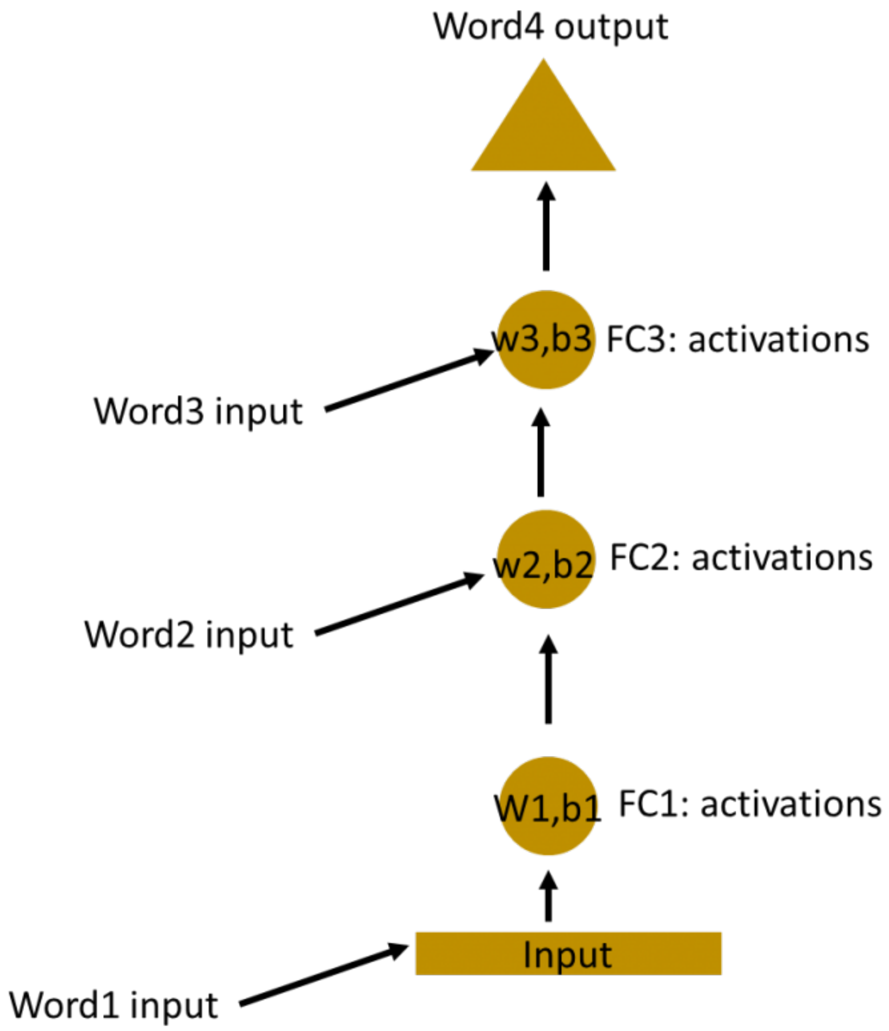
What are Recurrent Neural Networks?

- Assume the task of predicting the next word in a sentence.
- Consider an MLP:



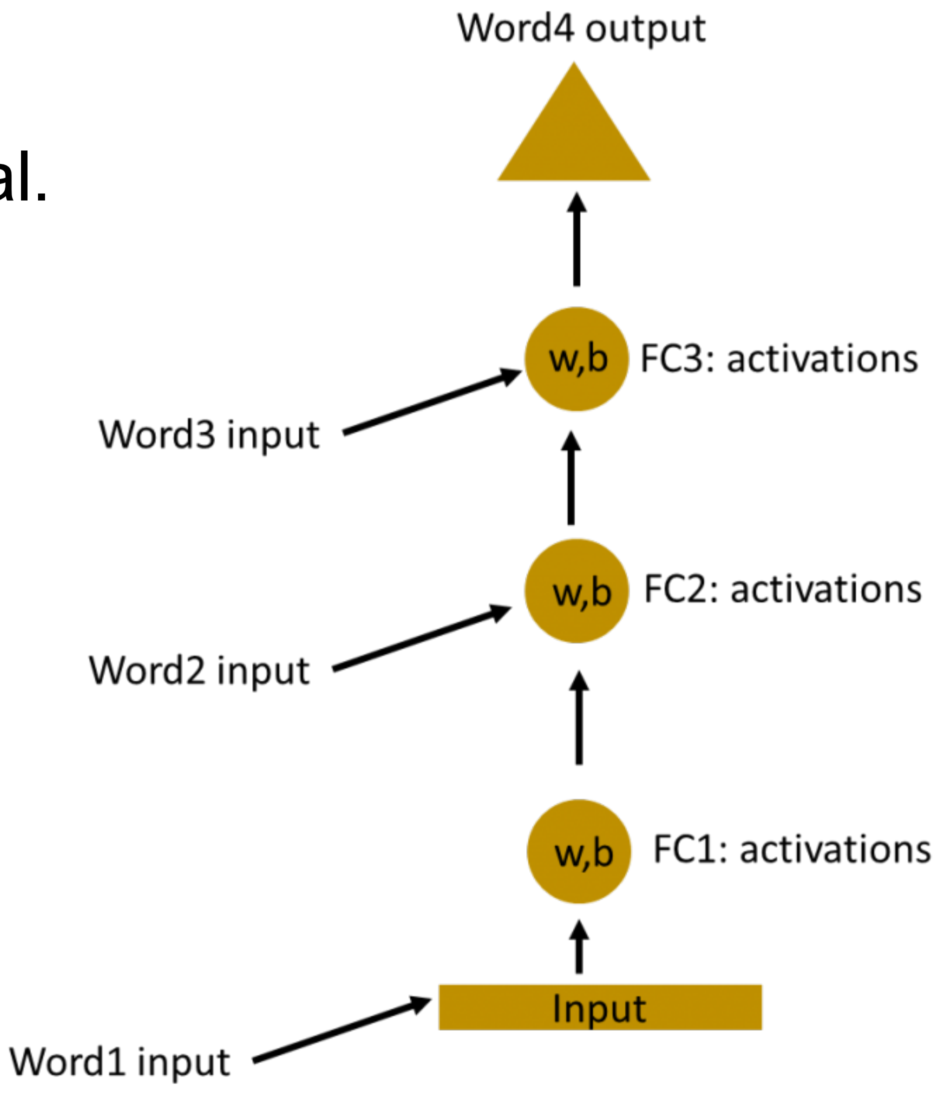
What are Recurrent Neural Networks?

- Assume the task of predicting the next word in a sentence.
- Consider a deep MLP:



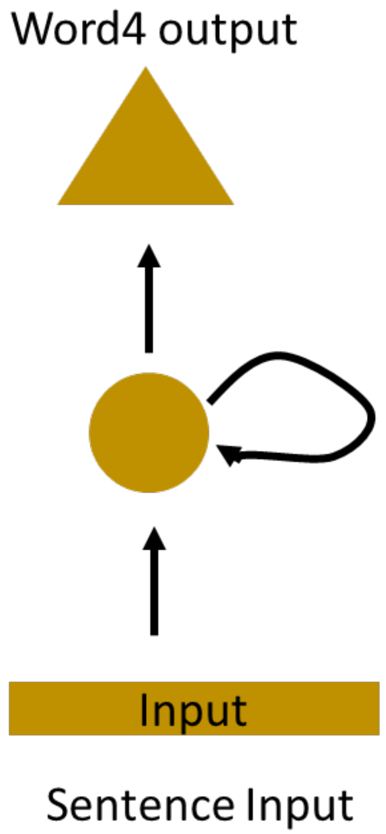
What are Recurrent Neural Networks?

- Assume the task of predicting the next word in a sentence.
- Lets weights and biases
- at different layers to be equal.



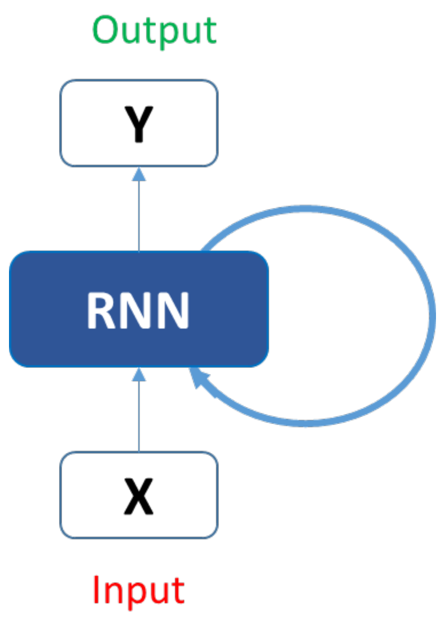
What are Recurrent Neural Networks?

- Assume the task of predicting the next word in a sentence.
- Now we can combine them into on layer:



What are Recurrent Neural Networks?

- A recurrent neuron:
  - stores the state of a previous input
  - And combines with the current input
  - Thereby preserving some relationship of the current input with the previous input.

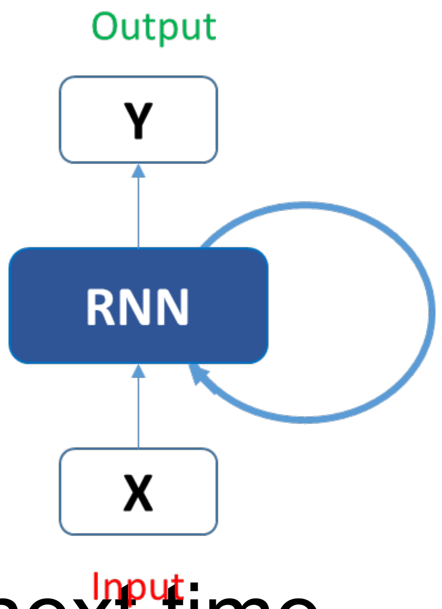


A recurrent neuron model

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

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$

$y_t = W_{hy}h_t$

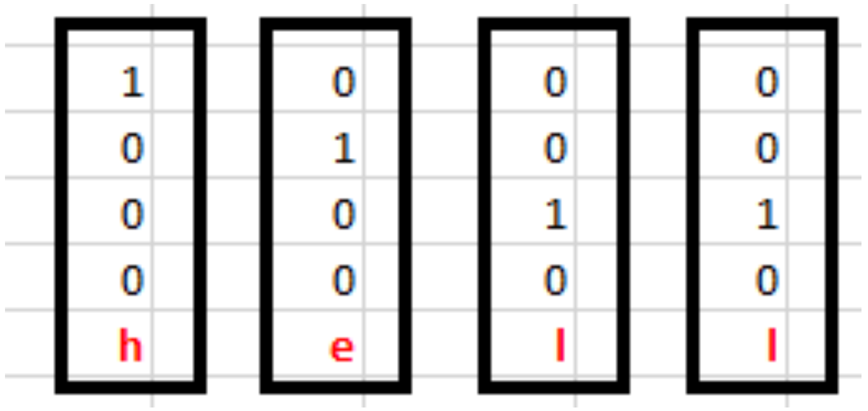


- The current h_t becomes h_{t-1} for the next time step.
- We can go as many time steps as the problem demands.
- The final current state is used to calculate the output y_t.

Example

- Assume the word “hello”.
- We present 4 first letters “hell” and network should predict the last letter “o”.

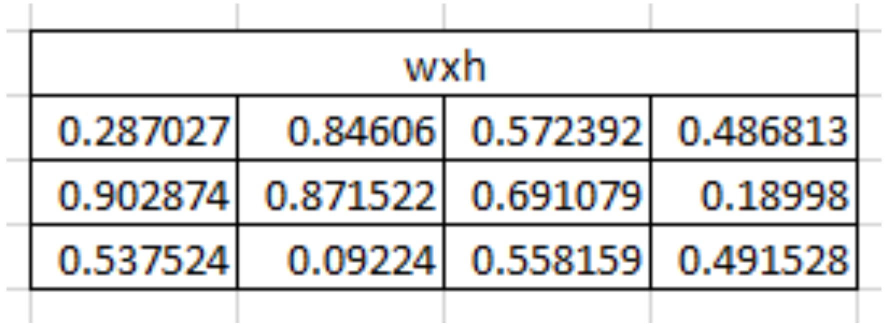
- Lets encode letters into one-hot vectors:



1	0	0	0
0	1	0	0
0	0	1	1
0	0	0	0
h	e	l	l

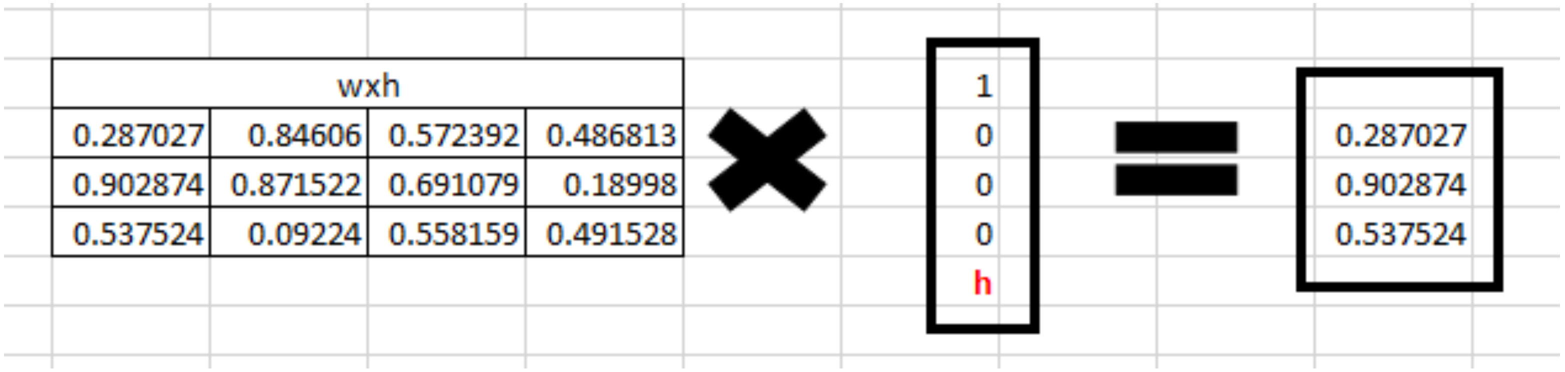
Example

- Lets the input weights of hidden neuron be:



wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

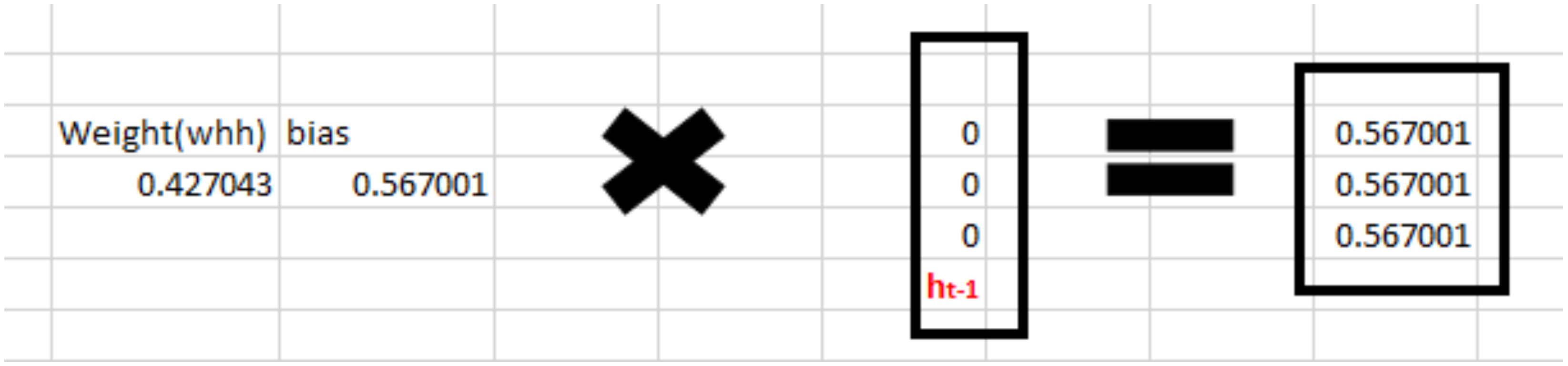
- Step 1: The first input is letter ‘h’:



$$\begin{array}{cccc} \multicolumn{4}{c}{\text{wxh}} \\ 0.287027 & 0.84606 & 0.572392 & 0.486813 \\ 0.902874 & 0.871522 & 0.691079 & 0.18998 \\ 0.537524 & 0.09224 & 0.558159 & 0.491528 \end{array} \times \boxed{\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ h \end{array}} = \boxed{\begin{array}{c} 0.287027 \\ 0.902874 \\ 0.537524 \end{array}}$$

Example

- Step 2:
  - W_{hh} is a 1*1 matrix: $\boxed{0.427043}$
  - Bias is also a 1*1 matrix: $\boxed{0.56700}$
  - For letter “h”, the previous state is [0,0,0], hence:



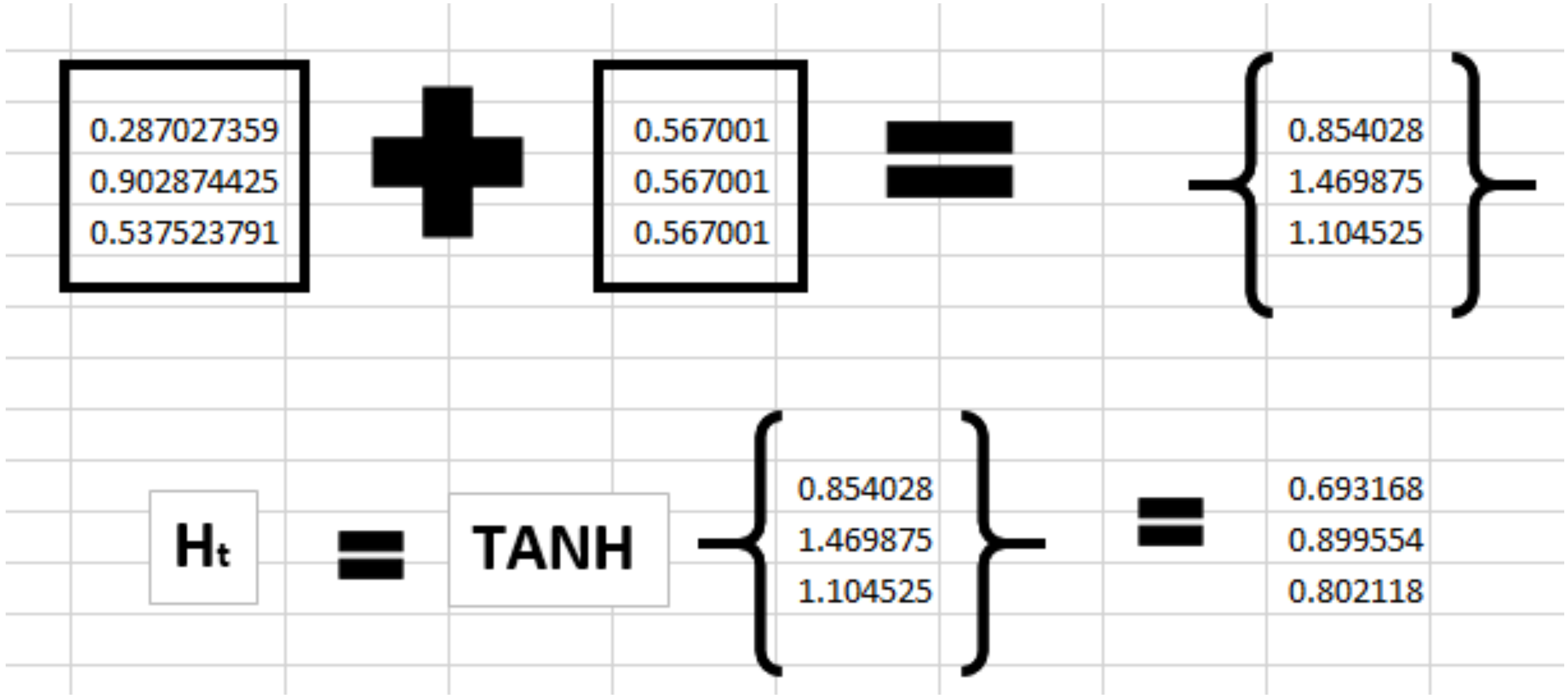
Weight(whh)	bias
0.427043	0.567001

$$\times \boxed{\begin{array}{c}0\\0\\0\\h_{t-1}\end{array}} = \boxed{\begin{array}{c}0.567001\\0.567001\\0.567001\end{array}}$$

Example

- Step 3: Now, we can have the current state:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



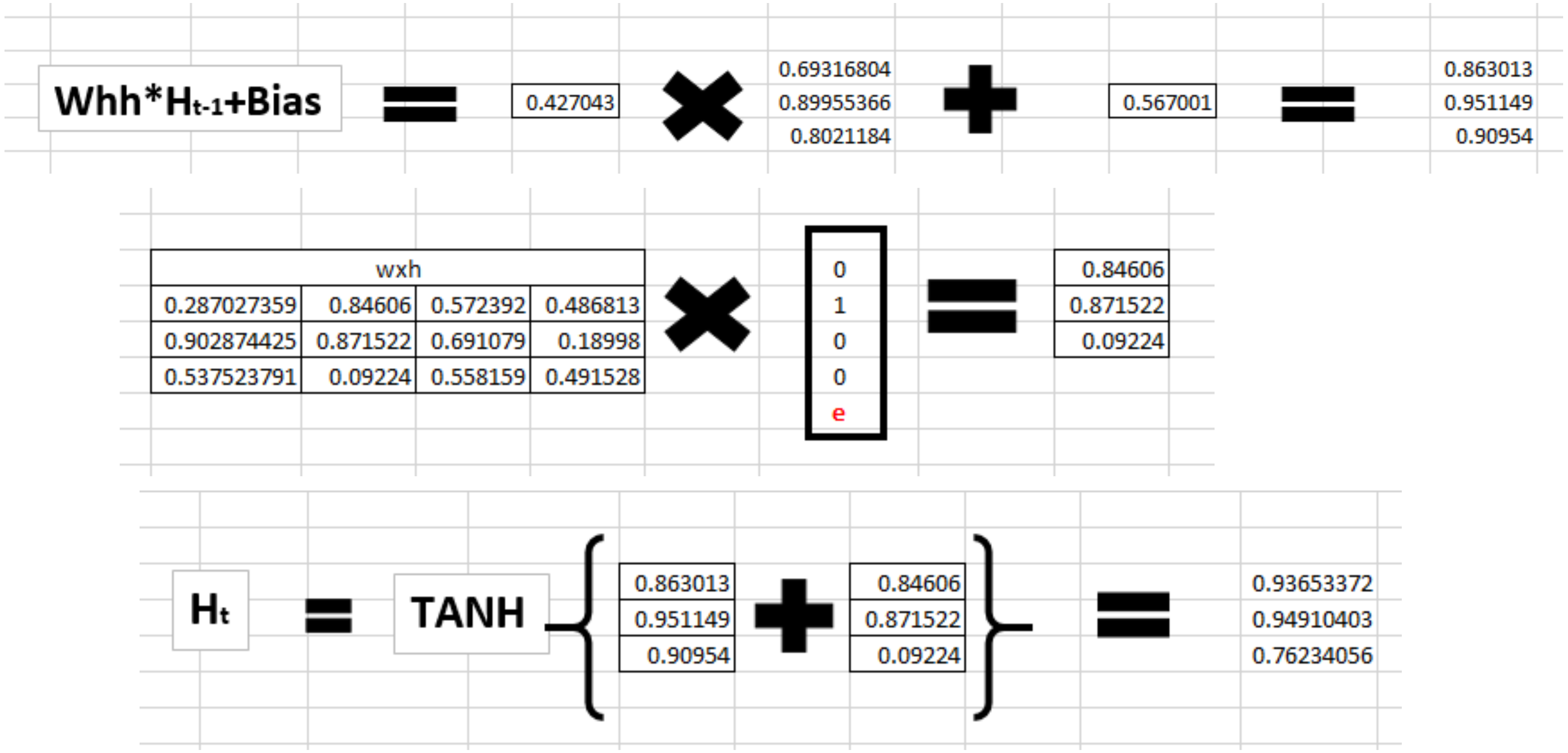
$$\begin{bmatrix} 0.287027359 \\ 0.902874425 \\ 0.537523791 \end{bmatrix} + \begin{bmatrix} 0.567001 \\ 0.567001 \\ 0.567001 \end{bmatrix} = \left\{ \begin{matrix} 0.854028 \\ 1.469875 \\ 1.104525 \end{matrix} \right\}$$

$$H_t = \text{TANH} \left\{ \begin{matrix} 0.854028 \\ 1.469875 \\ 1.104525 \end{matrix} \right\} = \begin{matrix} 0.693168 \\ 0.899554 \\ 0.802118 \end{matrix}$$

Example

- Step 4: Now we go on to the next letter “e”

$$h_t = \tanh\left(W_{hh}h_{t-1} + W_{xh}x_t\right)$$



$$\boxed{\text{Whh*H}_{t\text{-}1}\text{+Bias}} = \boxed{0.427043} \times \begin{matrix} 0.69316804 \\ 0.89955366 \\ 0.8021184 \end{matrix} + \boxed{0.567001} = \begin{matrix} 0.863013 \\ 0.951149 \\ 0.90954 \end{matrix}$$

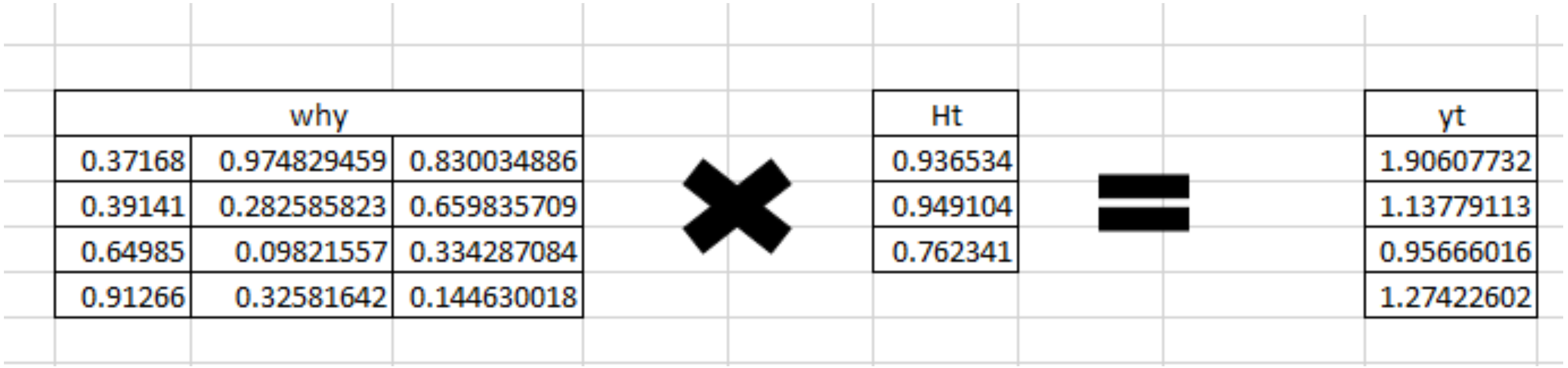
$$\begin{array}{|c|c|c|c|} \hline \multicolumn{4}{|c|}{\text{wxh}} \\ \hline 0.287027359 & 0.84606 & 0.572392 & 0.486813 \\ \hline 0.902874425 & 0.871522 & 0.691079 & 0.18998 \\ \hline 0.537523791 & 0.09224 & 0.558159 & 0.491528 \\ \hline \end{array} \times \boxed{\begin{matrix} 0 \\ 1 \\ 0 \\ 0 \\ e \end{matrix}} = \begin{array}{|c|} \hline 0.84606 \\ \hline 0.871522 \\ \hline 0.09224 \\ \hline \end{array}$$

$$\boxed{H_t} = \boxed{\text{TANH}} \left\{ \begin{array}{|c|} \hline 0.863013 \\ \hline 0.951149 \\ \hline 0.90954 \\ \hline \end{array} + \begin{array}{|c|} \hline 0.84606 \\ \hline 0.871522 \\ \hline 0.09224 \\ \hline \end{array} \right\} = \begin{matrix} 0.93653372 \\ 0.94910403 \\ 0.76234056 \end{matrix}$$

Example

- At each state, RNN would produce the output as well.
- Let's calculate y_t for the letter "e".

$$y_t = W_{hy} h_t$$



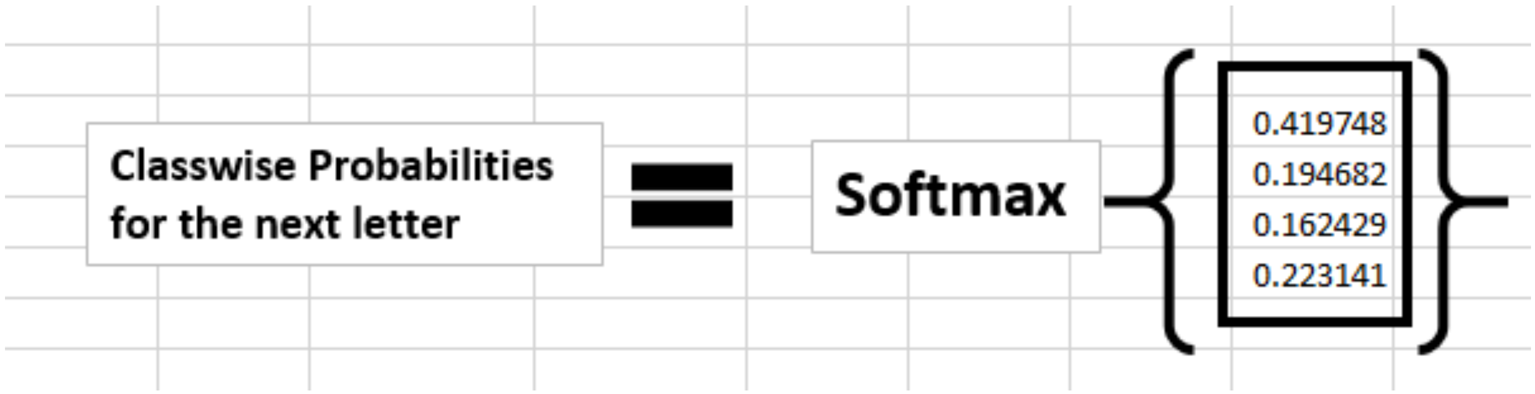
why		
0.37168	0.974829459	0.830034886
0.39141	0.282585823	0.659835709
0.64985	0.09821557	0.334287084
0.91266	0.32581642	0.144630018

×

Ht
0.936534
0.949104
0.762341

=

yt
1.90607732
1.13779113
0.95666016
1.27422602



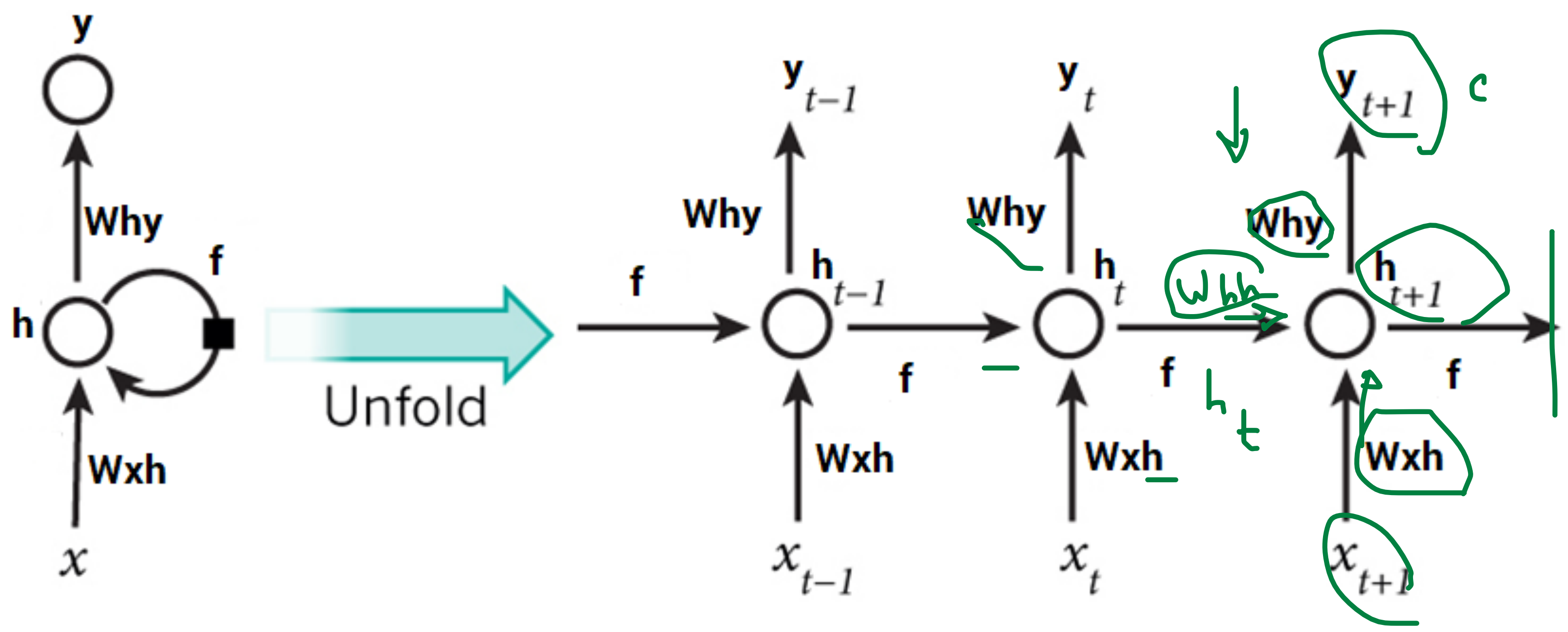
Classwise Probabilities for the next letter = Softmax

0.419748
0.194682
0.162429
0.223141

The model says the letter after "e" should be h,

Back propagation through time (BPTT)

to understand and visualize the back propagation, let's unroll the network at all the time steps



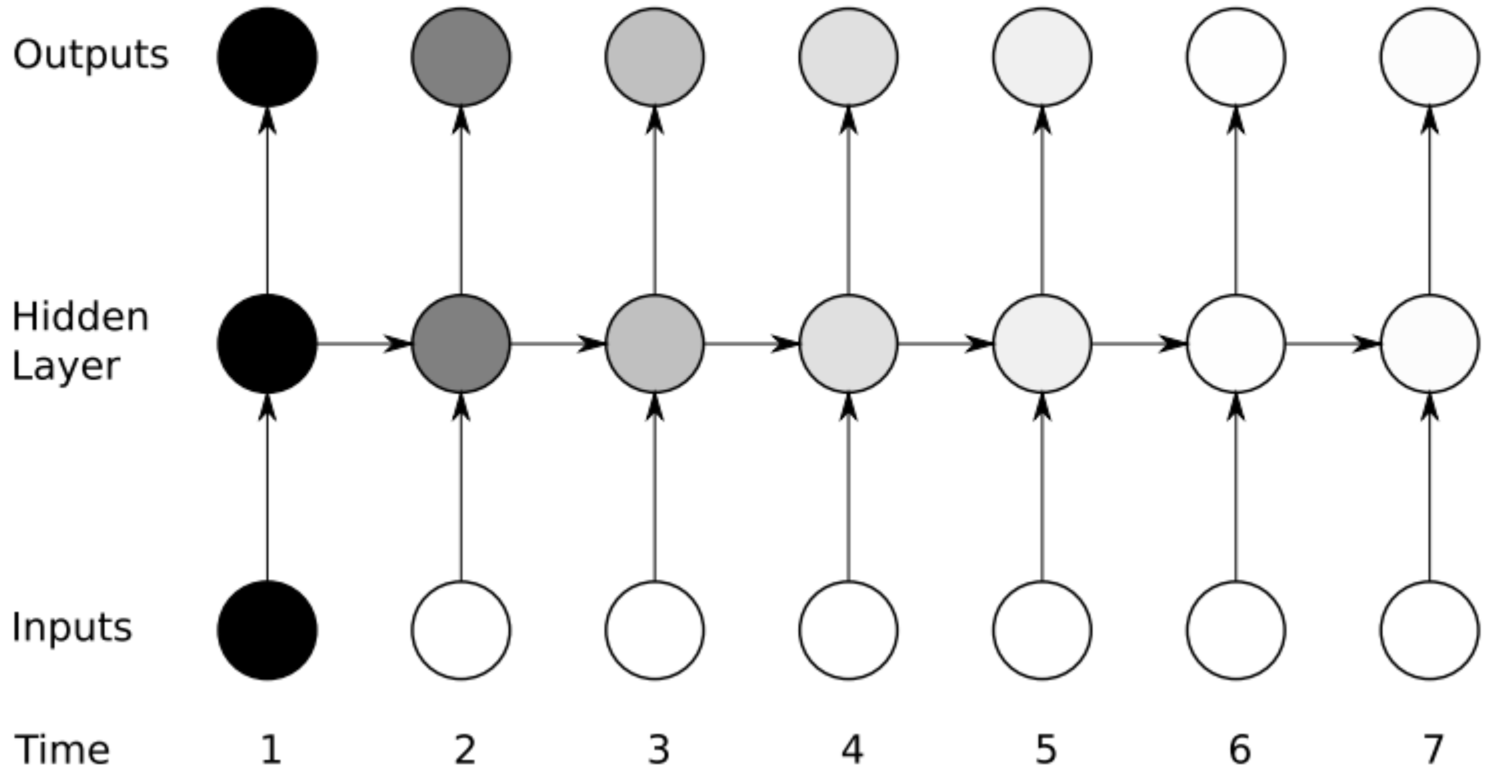
In forward path, the inputs enter and move forward at each time step.
In backward propagation , we are going back in time to change the weights,
Hence we call it the **Back propagation through time (BPTT).**

Back propagation through time (BPTT)

- Remember that the network is unrolled for all the time steps.
- For the unrolled network, the gradient is calculated for each time step with respect to the weight parameter.
- Now that the weight is the same for all the time steps the gradients can be combined together for all time steps.

Vanishing gradient

- RNNs might have a difficulty in learning long range dependencies (due to gradient vanishing).



Vanishing gradient

- To calculate the error after the third time step with respect to the first one we have:

- ∂E/∂W = ∂L/∂y3 *∂y3/∂h3 *∂h3/∂y2 *∂y2/∂h1 ..

- IF one of the gradients approached 0, all the gradients would rush to zero.

Limitations of RNNs

- Recurrent Neural Networks work just fine when we are dealing with short-term dependencies.

The colour of the sky is ______.

- This problem has nothing to do with the context of the statement.
- RNN need not remember what was said before this, or what was its meaning, all they need to know is that **in most cases** the sky is blue.

The colour of the sky is blue.

Long recalls

- Something that was said long before, cannot be recalled in RNNs when making predictions in the present.

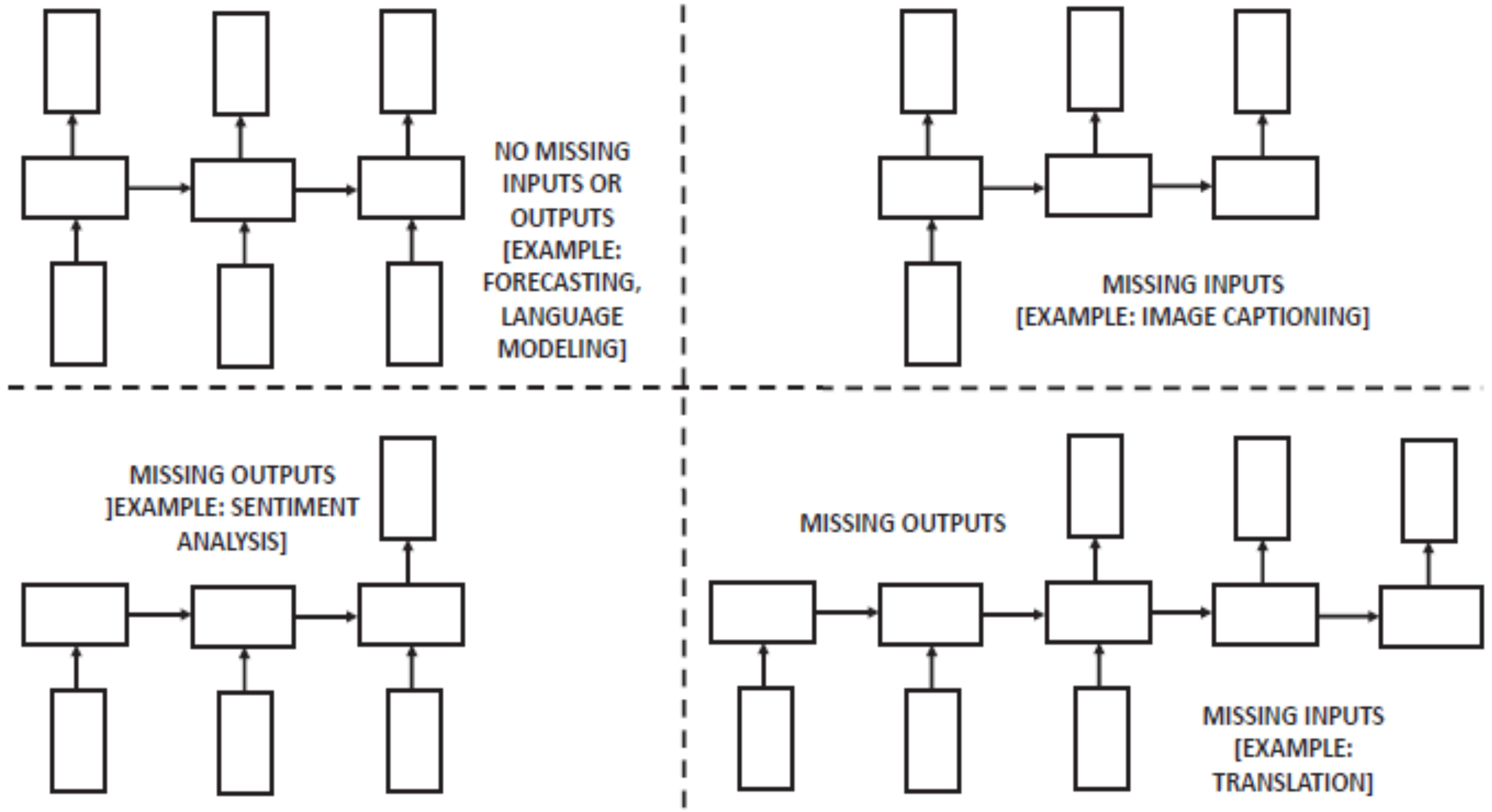
I spent 20 long years working for the under-privileged kids in Spain. I then moved to Africa.

....

I can speak fluent ________.

- The reason behind this is the problem of Vanishing Gradient.

Recurrent Neural Network Applications are Architecture Sensitive!


NO MISSING INPUTS OR OUTPUTS [EXAMPLE: FORECASTING, LANGUAGE MODELING]
MISSING INPUTS [EXAMPLE: IMAGE CAPTIONING]
MISSING OUTPUTS]EXAMPLE: SENTIMENT ANALYSIS]
MISSING OUTPUTS
MISSING INPUTS [EXAMPLE: TRANSLATION]

Backpropagation Through Time

- Loss function : The negative logarithms of the softmax probability of the correct words at the various time-stamps
- If the output vector $\bar{y}_t$ can be written as $[\hat{y}_t^1 \dots \hat{y}_t^d]$, it is first converted into a vector of *d* probabilities using the softmax function:

$$[\hat{p}_t^1 \dots \hat{p}_t^d] = \text{Softmax}([\hat{y}_t^1 \dots \hat{y}_t^d])$$

$$\text{L} = -\sum_{t=1}^{T} \log(\hat{p}_t^{j_t})$$

$$\frac{dL}{d\hat{y}_t^k} = \hat{p}_t^k - \text{I}(\text{k}-j_t)$$

$I(k, j_t)$ is an indicator function that is 1 when k and j_t are the same, and 0,otherwise

Backpropagation Through Time

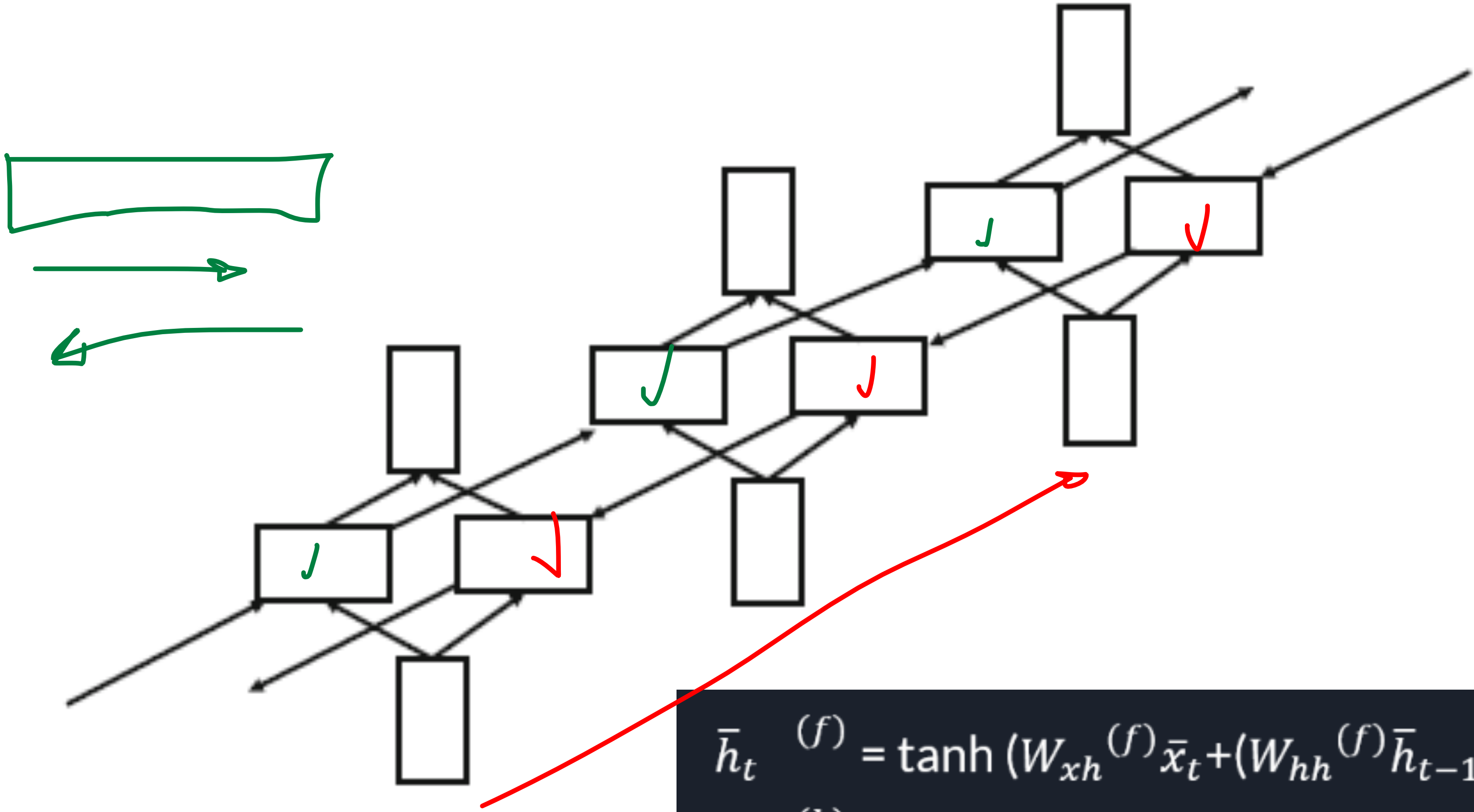
add all the (shared) weights corresponding to different instantiations of an edge in time. In other words:

$$\frac{dL}{dW_{xh}} = \sum_{t=1}^{T} \frac{dL}{dW_{xh}{}^{(t)}}$$

$$\frac{dL}{dW_{hh}} = \sum_{t=1}^{T} \frac{dL}{dW_{hh}{}^{(t)}}$$

$$\frac{dL}{dW_{hy}} = \sum_{t=1}^{T} \frac{dL}{dW_{hy}{}^{(t)}}$$

Bidirectional Recurrent Networks

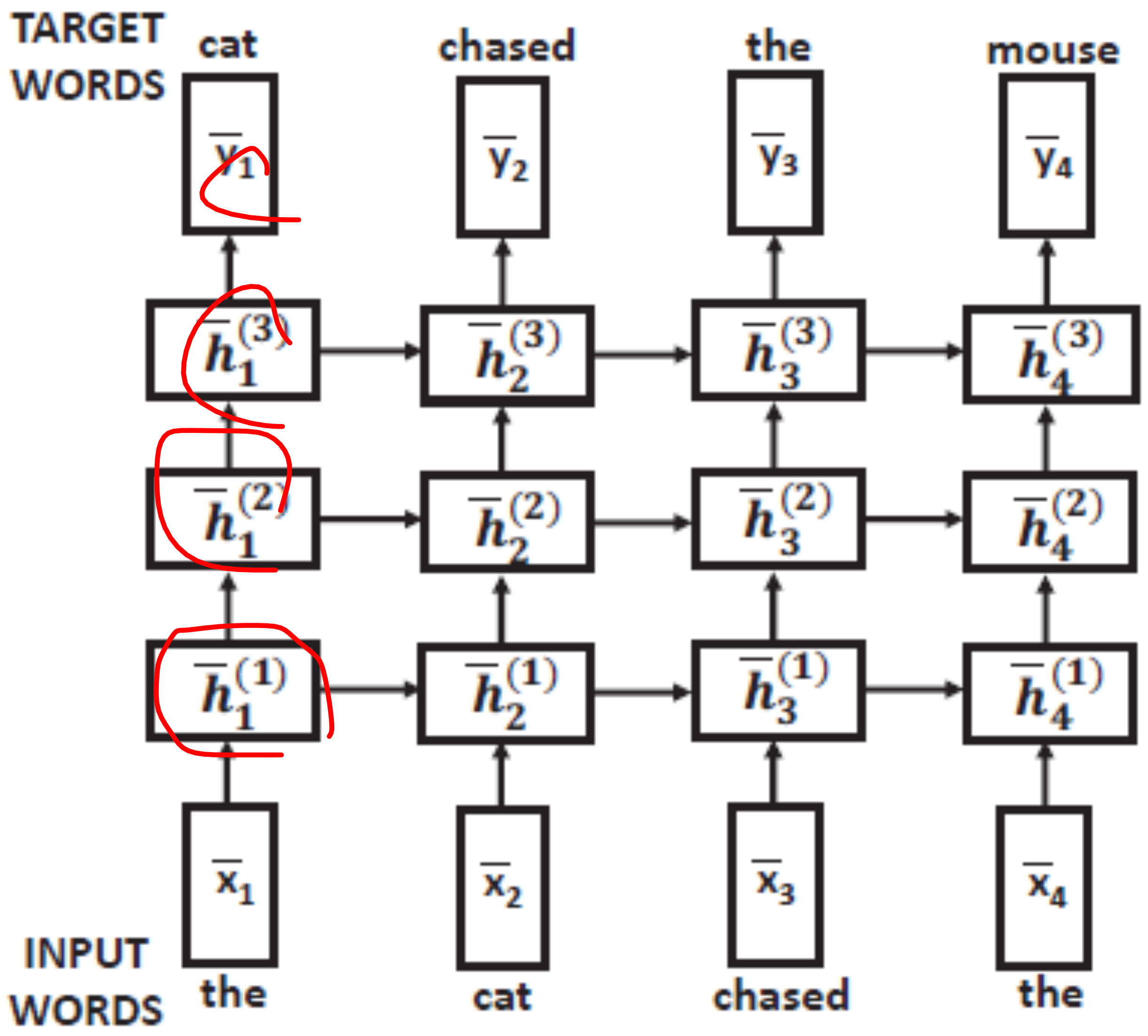


$$\bar{h}_t^{(f)} = \tanh\,(W_{xh}{}^{(f)}\bar{x}_t + (W_{hh}{}^{(f)}\bar{h}_{t-1}{}^{(f)})$$

$$\bar{h}_t^{(b)} = \tanh\,(W_{xh}{}^{(b)}\bar{x}_t + (W_{hh}{}^{(b)}\bar{h}_{t+1}{}^{(b)})$$

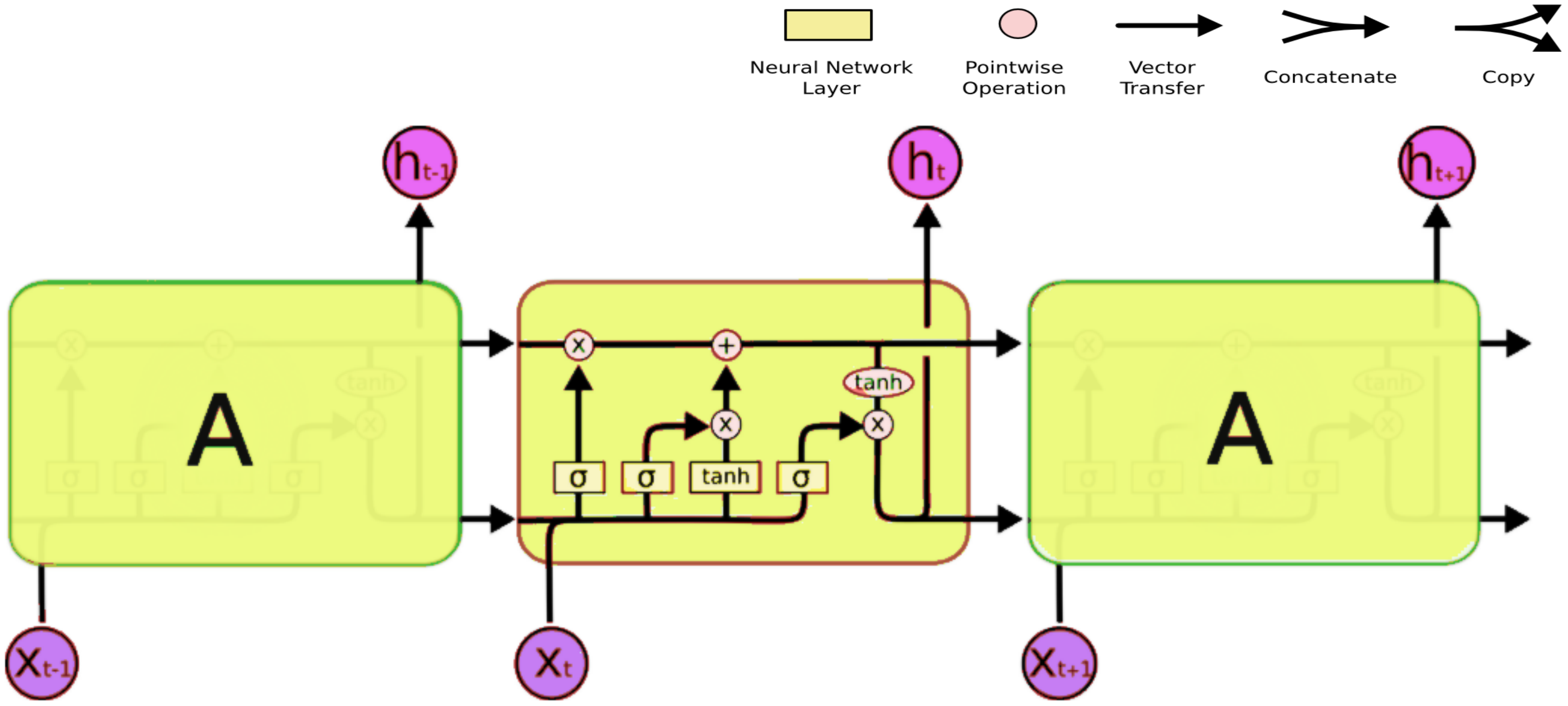
$$\bar{y}_t = W_{hy}{}^{(f)}\bar{h}_t{}^{(f)} + W_{hy}{}^{(b)}\bar{h}_{t+1}{}^{(\boldsymbol{b})}$$

Multilayer Recurrent Networks



LSTM (Long Short-Term Memory) Networks

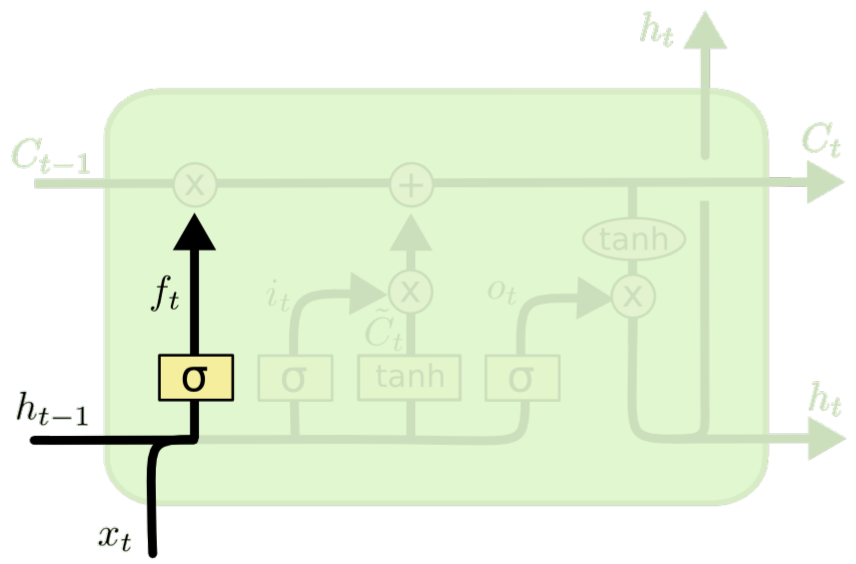
- LSTM is a RNN with advanced memory management.
- Each LSTM unit has:
  - A cell state (specifying long-term memory)
  - A forget gate (What should be removed from long-term memory)
  - An input gate (what shold be added to long-term memory)
  - An output gate (specifying the output and short-term memory)



• Forget Gate:

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

- As the point after "person" is encountered, the forget gate realizes that there may be a change of context in the next sentence.
- The value of the forget gate is multiplied by the value of cell state.

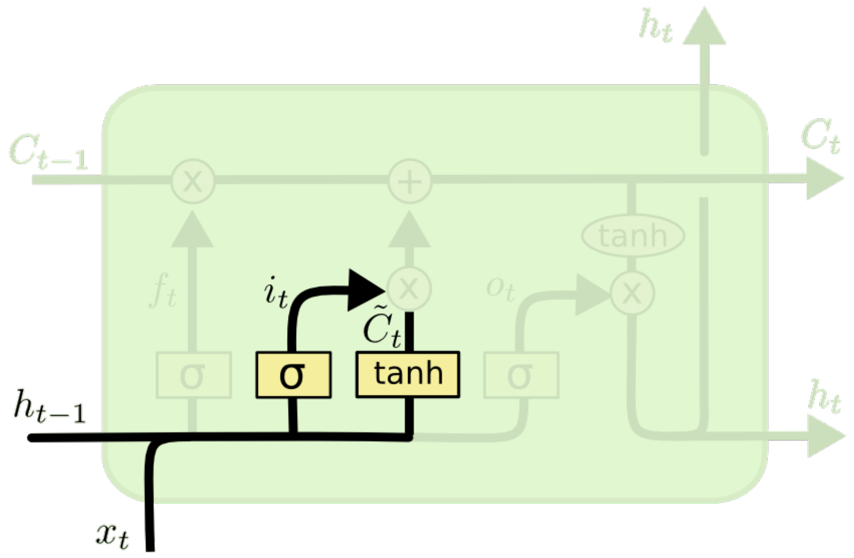


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Input gate

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

- The important infor 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.

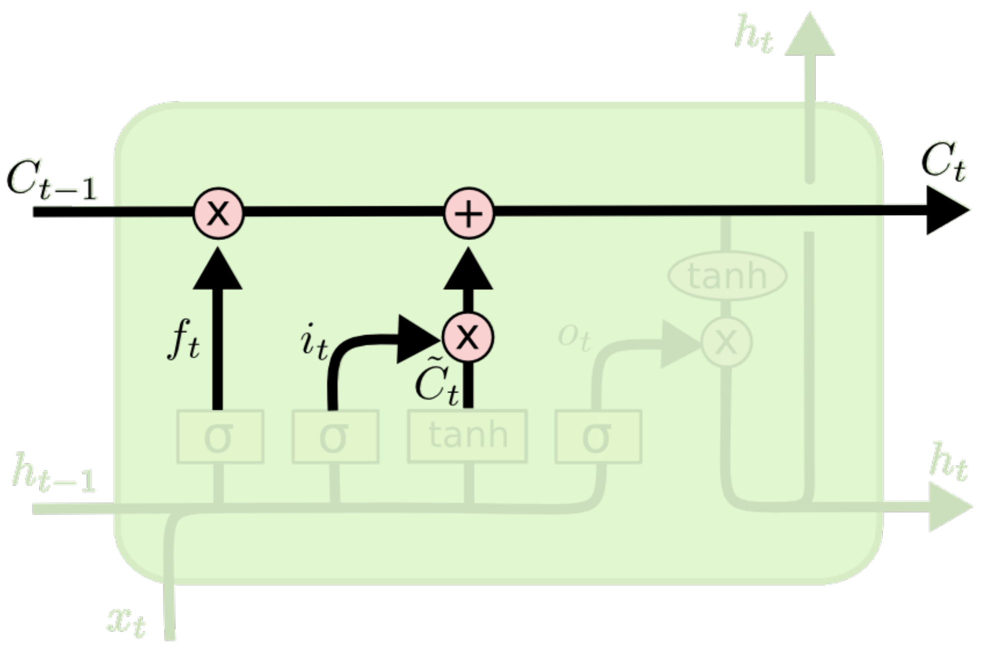


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Input gate

- The input gate is responsible for the addition of information to the cell state.
- Input gate has 3 parts:
  - Creating a vector containing **all possible values** that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the **tanh function**, which outputs values from -1 to +1.
  - Regulating what values need to be added to the cell state by involving a **sigmoid function**.
  - Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Updating cell state

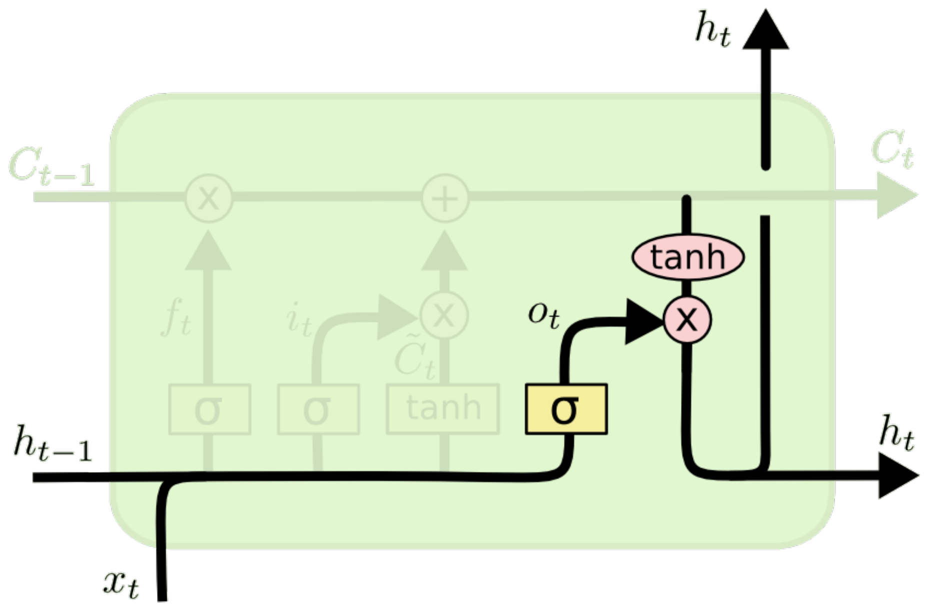


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output gate

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

- There might be several options for empty space.
- But in cell state we know that Bob has died for his country.
- So, based on the cell state we remove other options but Bob.



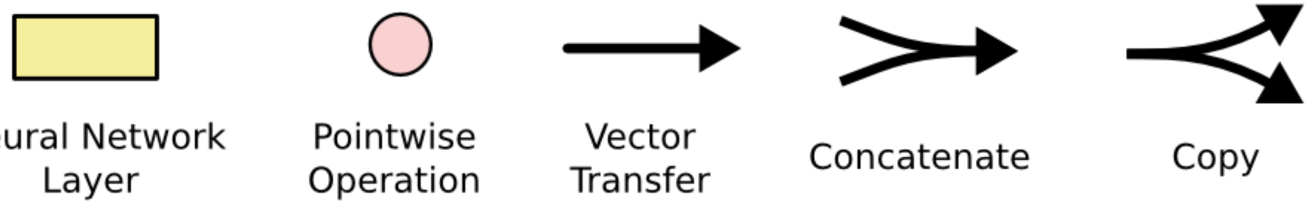
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

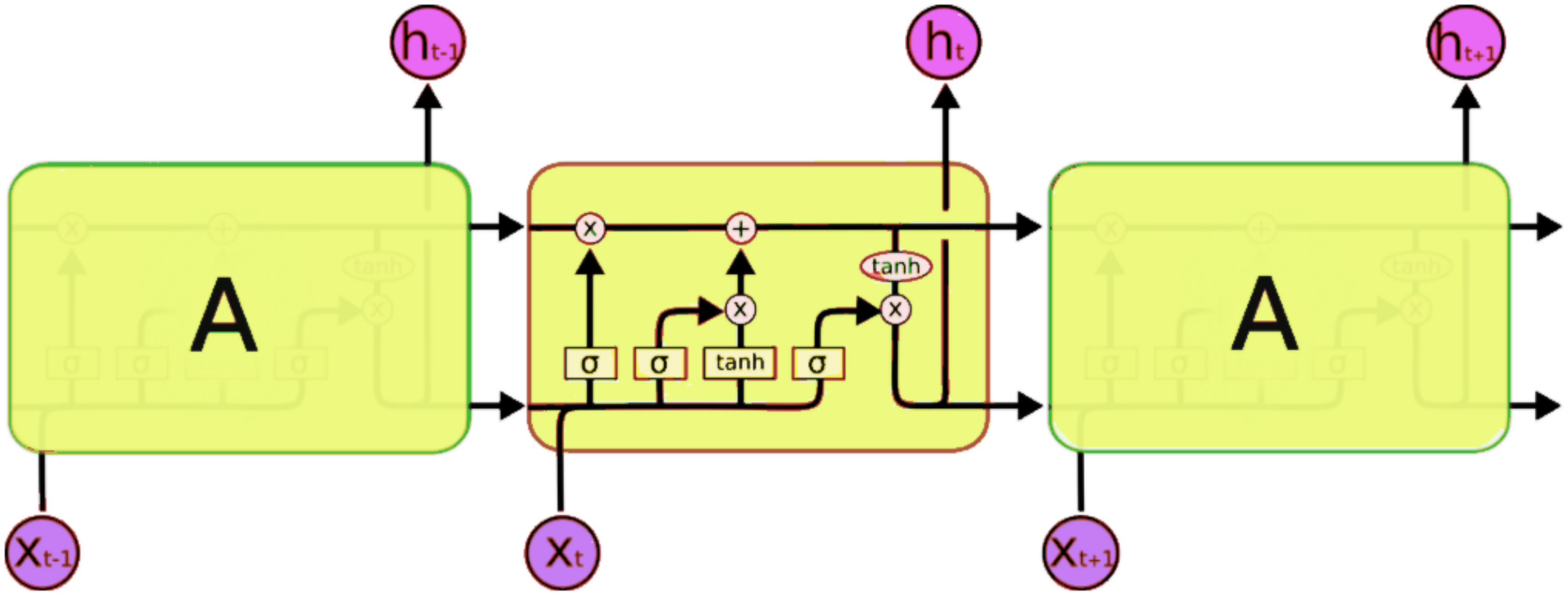
$$h_t = o_t * \tanh\left(C_t\right)$$

Output gate

- 1- All possible output information is provided by the **sigmoid** function based on the input, x_t, and previous state, h_t.
- 2- Applying **tanh** to the cell state, scales its values to the range -1 to +1.
- 3-Multiply the regulatory filter (step2) to the vector created in step 1, and sending it out as a **output** and also to the **hidden state** of the next cell.

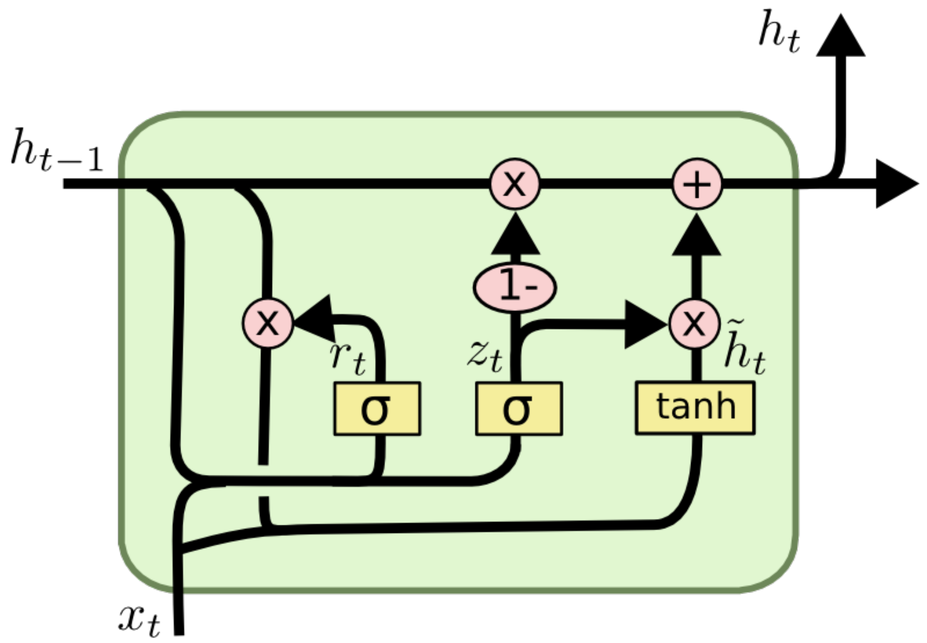
LSTM


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy


h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
×
+
×
×
X_{t-1}
X_t
X_{t+1}

Gated Recurrent Unit (GRU)

It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$