INTRODUCTION TO NEURAL NETWORKS

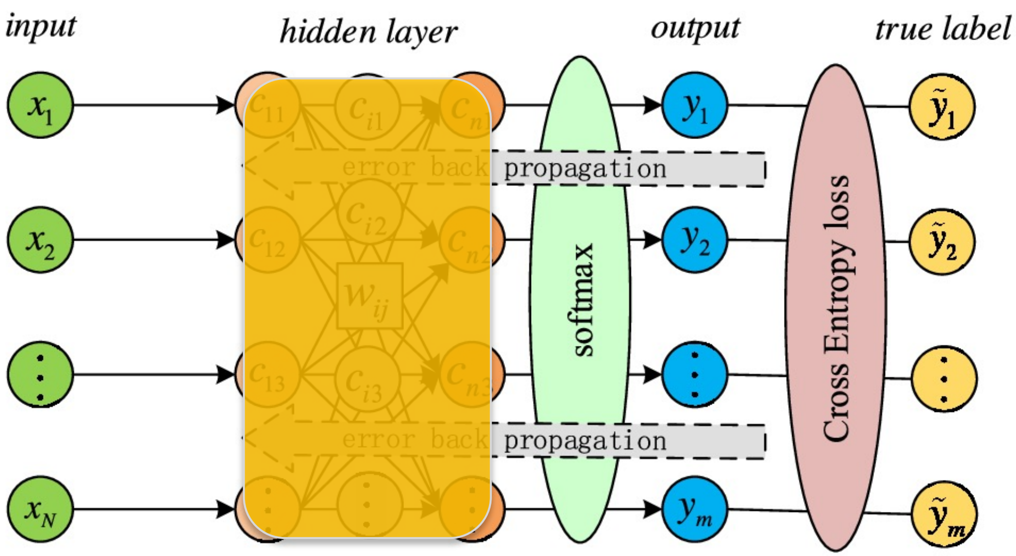
Lecture 7. Introduction to Recurrent Neural Networks

Dr. Jingxin Liu

School of AI and Advanced Computing

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Backpropagation with Softmax and CE

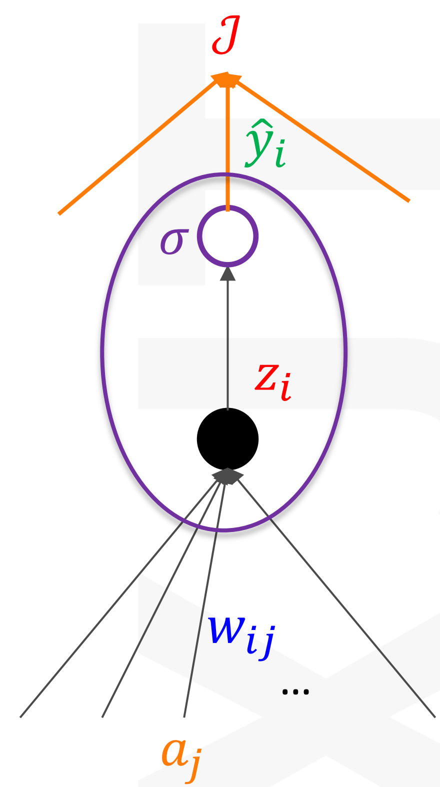


$$\frac{\partial \mathcal{J}}{\partial w_{ij}} = \frac{\partial \mathcal{J}}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}}$$

for last layer with $\mathcal{J} = \sum_i -y_i \log \hat{y}_i$

$$\delta_i = \frac{\partial \mathcal{J}}{\partial z_i} = \frac{\partial \mathcal{J}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i}$$

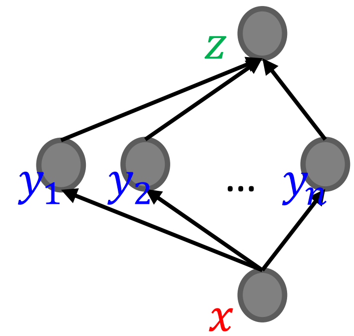
$$= \hat{y}_i - y_i$$



Backpropagation with Softmax and CE

For the $N-1$ layer,

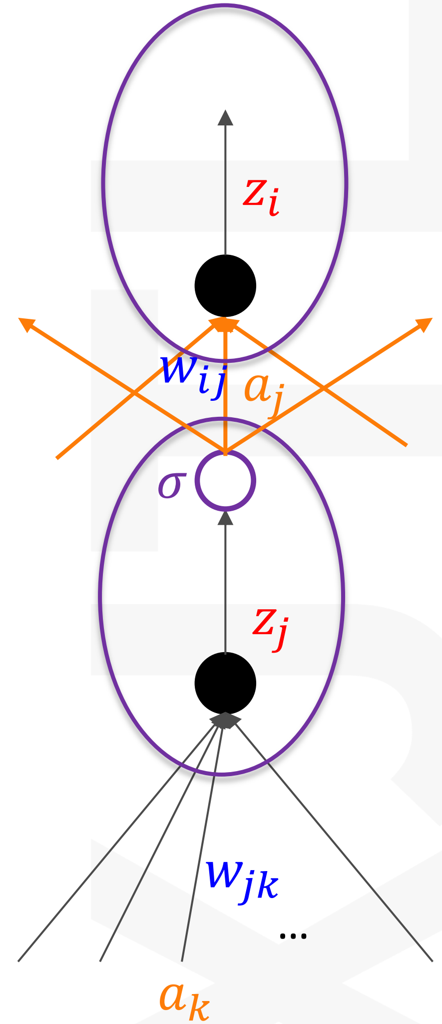
$$\frac{\partial \mathcal{J}}{\partial w_{jk}} = \sum_j \frac{\partial \mathcal{J}}{\partial z_j} \frac{\partial z_j}{\partial w_{jk}}$$



$$\delta_j = \frac{\partial \mathcal{J}}{\partial z_j} = \frac{\partial \mathcal{J}}{\partial a_j} \frac{\partial a_j}{\partial z_j} = \delta_i w_{ij} \cdot \sigma'(z_j)$$

$$\frac{\partial \mathcal{J}}{\partial a_j} = \frac{\partial \mathcal{J}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} \frac{\partial z_i}{\partial a_j} = \delta_i w_{ij}$$

$$\frac{\partial \mathcal{J}}{\partial w_{jk}} = \sum_j \delta_i w_{ij} \sigma'(z_j) a_k$$



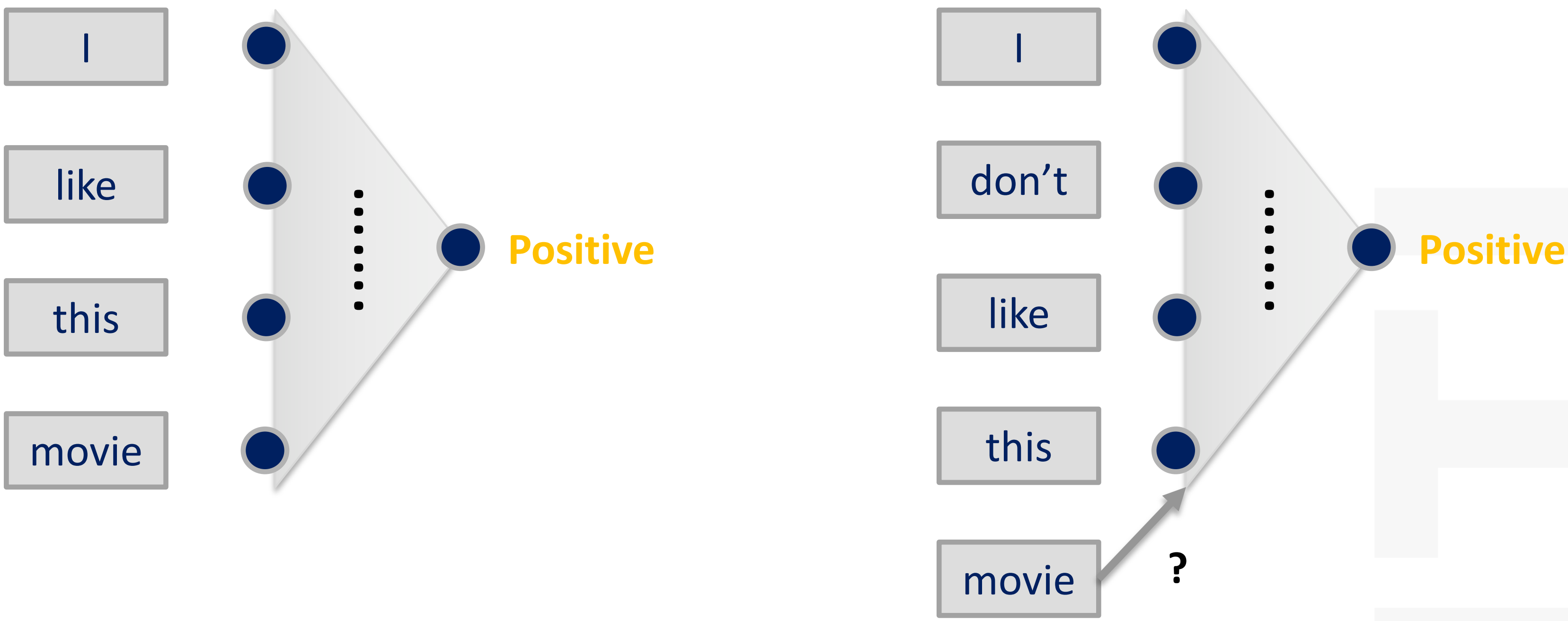
https://doug919.github.io/notes-on-backpropagation-with-cross-entropy/

Table of Contents

Why do we need Recurrent Neural Networks?

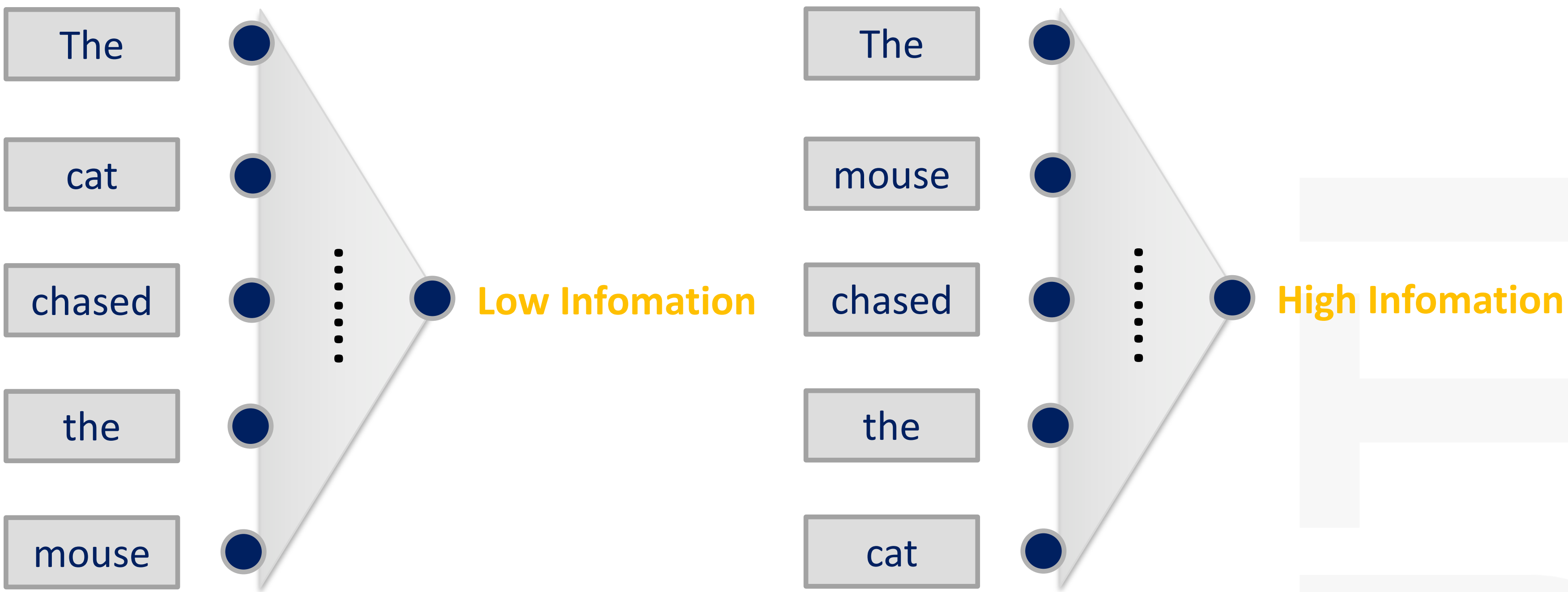
Examples of Sequence Data	Input Data	Output
Speech Recognition		This is RNN
Sentiment Classification	I don’t like this movie.	
Machine Translation	I don’t like this movie.	我不喜欢这部电影
Image Captioning		A cat sitting on a suitcase on the floor

Why do we need Recurrent Neural Networks?



Inputs or outputs can be different lengths in different examples

Why do we need Recurrent Neural Networks?



Share features learned across different positions or time steps

Why do we need Sequential Model?

Feed-Forward Neural Network

- No notion of order in time, and the only input it considers is the current example it has been exposed to

Recurrent Neural Network

- Possibility of processing input of any length
- Model size not increasing with size of input
- Computation takes into account historical information

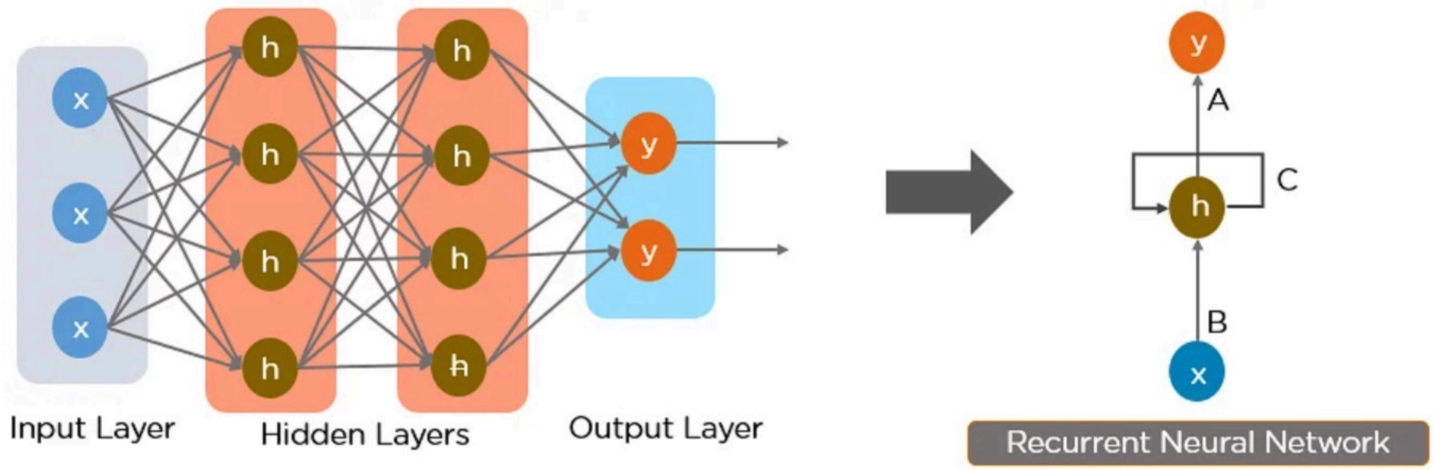
https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Recurrent Neural Networks

A **recurrent neural network (RNN)** is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour.

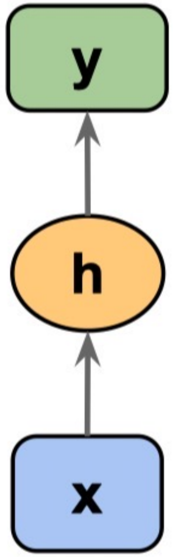
RNNs can use their internal state (memory) to process variable length sequences of inputs.

Main idea: use hidden state to capture information about the past

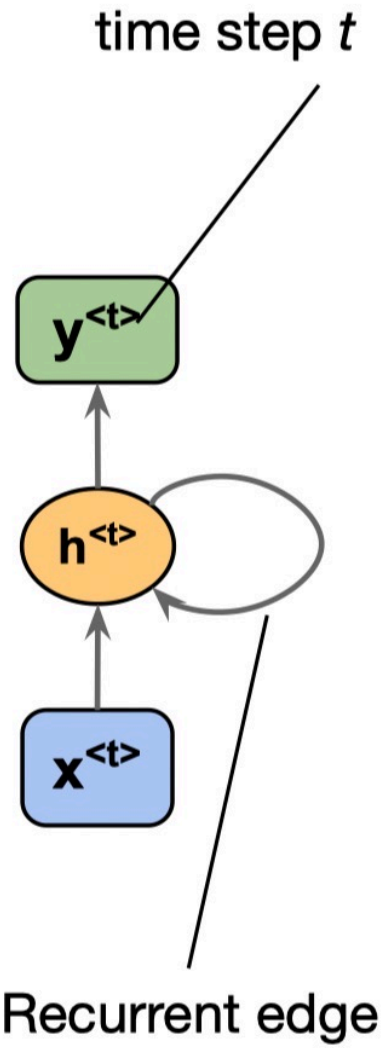


https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Recurrent Neural Networks



Feed Forward Neural Network



Recurrent Neural Network

Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Recurrent Neural Networks

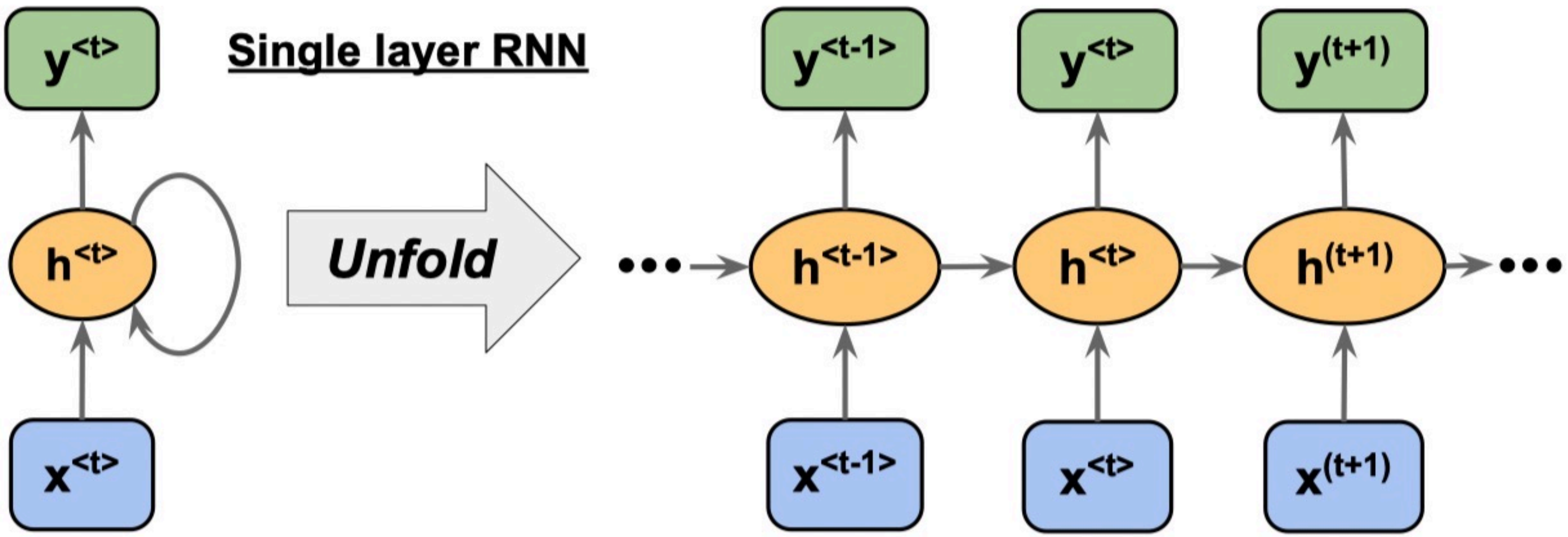


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Recurrent Neural Networks

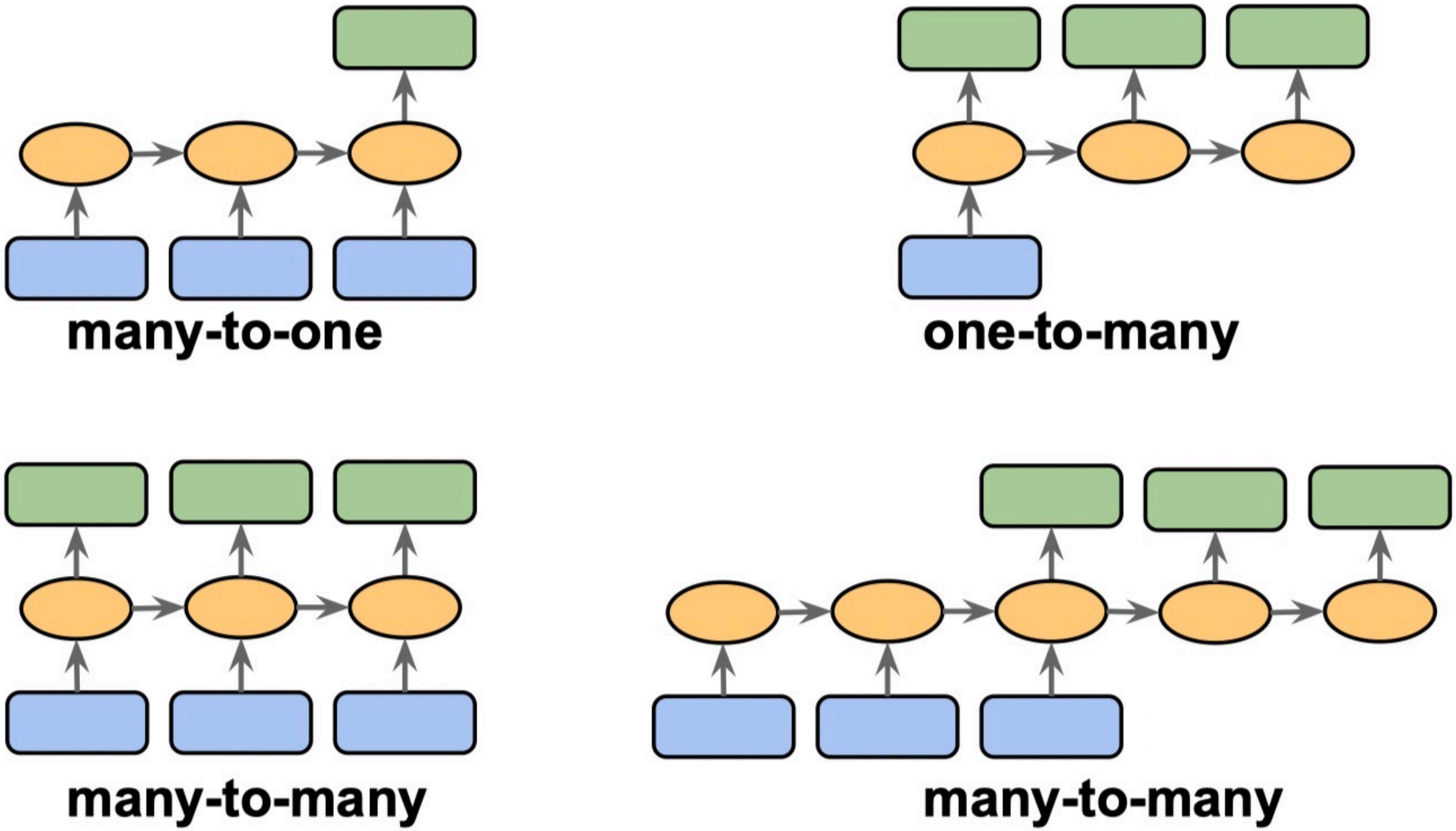
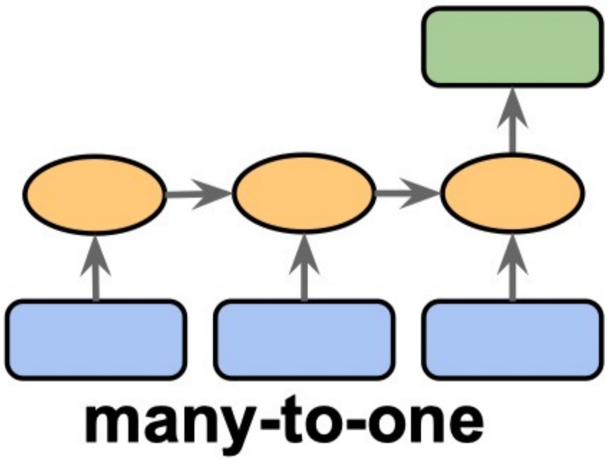
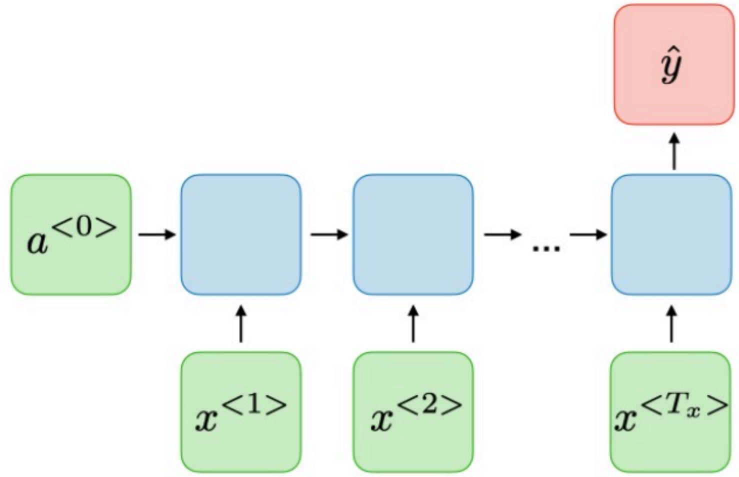


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Recurrent Neural Networks



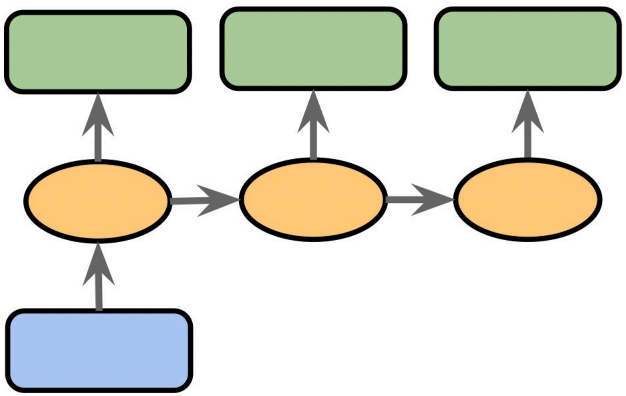


Many-to-One: The input data is a sequence, while the output is a fixed size vector, not sequence.

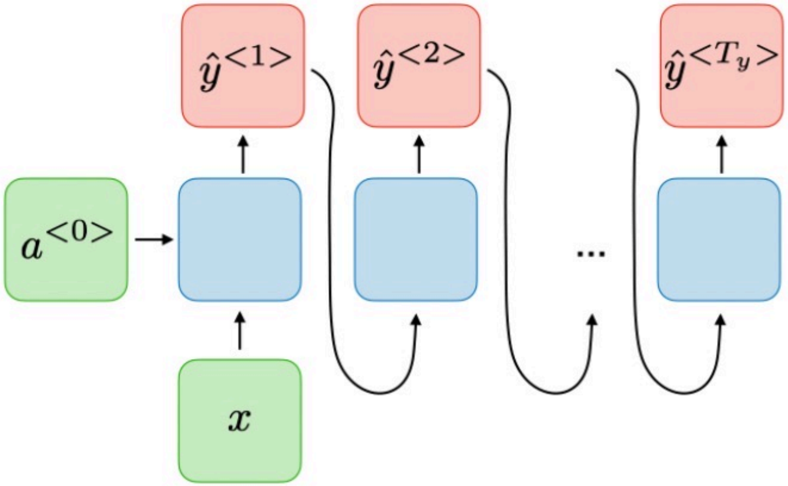
Example: Sentiment analysis, the input is some text, and the output is a class label.

Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks



one-to-many

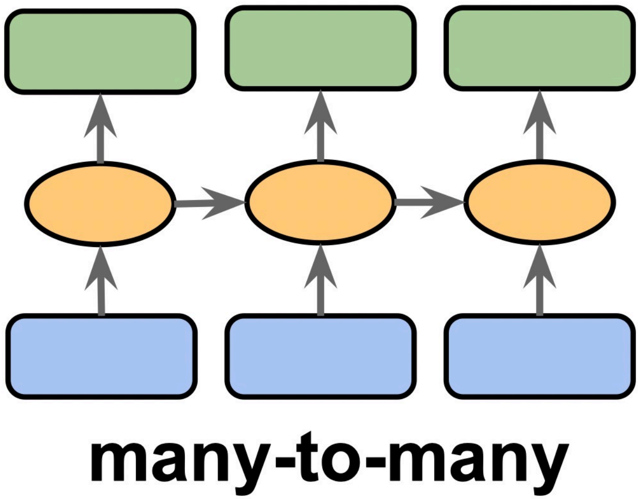


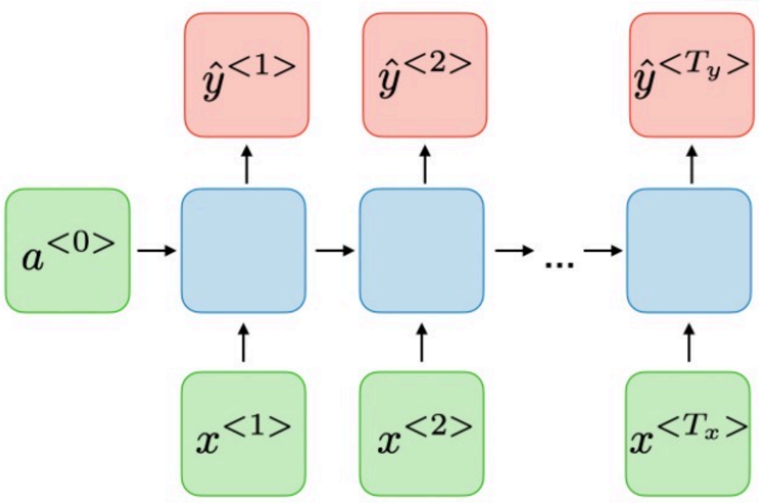
One-to-Many: The input data is a standard format (not a sequence), while the output is a sequence.

Example: Image captioning, where the input is an image, the output is a text description of that image.

Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks



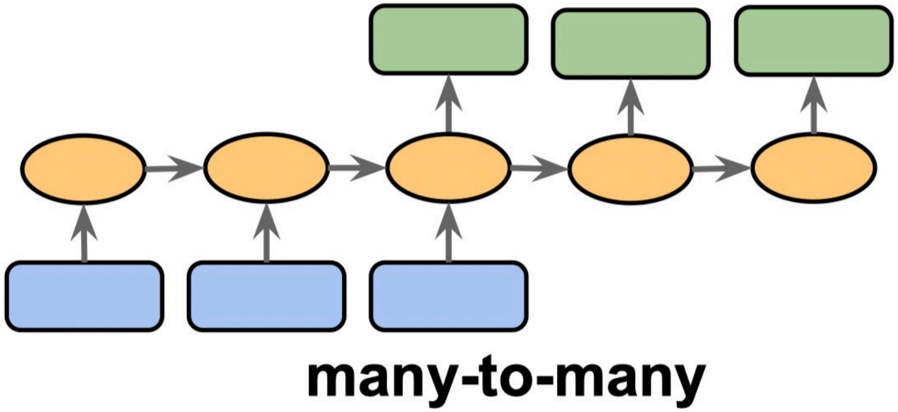


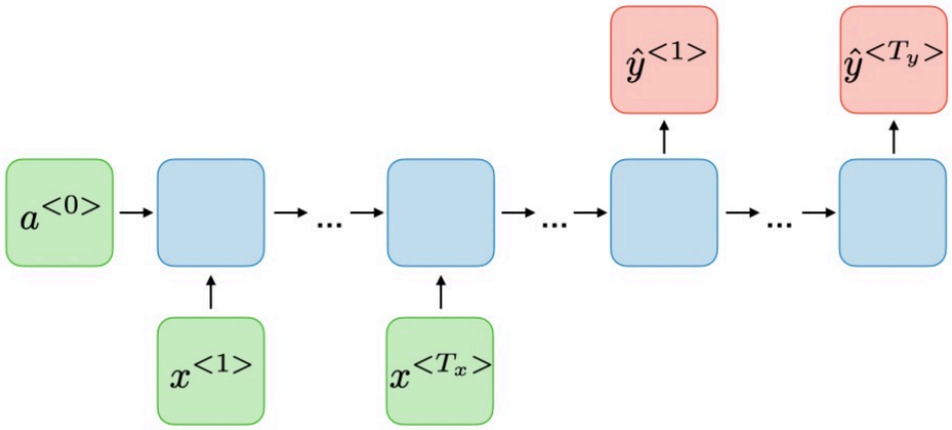
Many-to-Many (direct): Both inputs and outputs are sequences.

Example:
video captioning, describing a sequence of image via text.
Name entity recognition, identify the entities in a text.

Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks



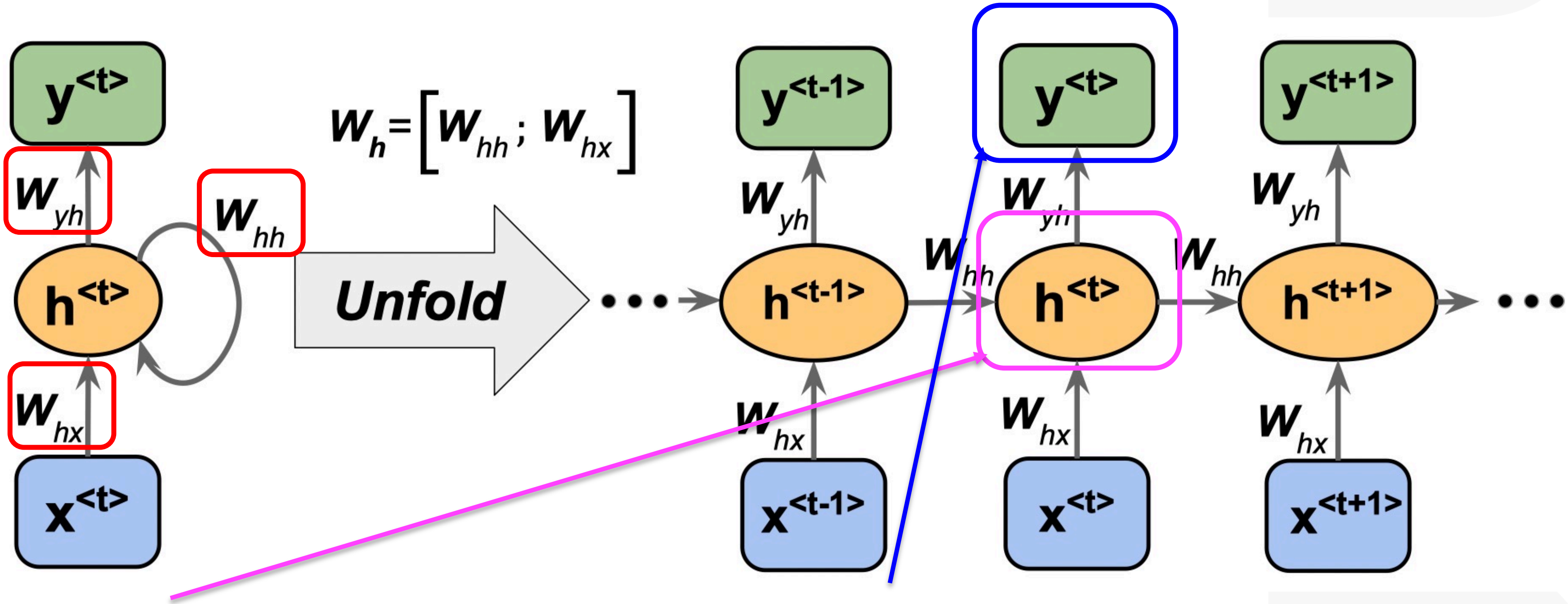


Many-to-Many (delay): Both inputs and outputs are sequences.

Example:
Machine translation.

Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks



Weighted Summation:

$$z_h^{<t>} = W_{hx}\mathrm{x}^{<t>} + W_{hh}\mathrm{h}^{<t-1>} + \mathrm{b}_h$$

Activation:

$$\mathrm{h}^{<t>} = g_h(z_h^{<t>})$$

Weighted Summation:

$$z_y^{<t>} = W_{yh}\mathrm{h}^{<t>} + \mathrm{b}_y$$

Activation:

$$y^{<t>} = g_y(z_y^{<t>})$$

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks

For the input to hidden Layer:

Weighted Summation:

$$z_h^{<t>} = W_{hx}\mathrm{x}^{<t>} + W_{hh}\mathrm{h}^{<t-1>} + \mathrm{b}_h$$

OR

$$z_h^{<t>} = \sum_{i=1}^{I} w_{hi}x_i^{<t>} + \sum_{h\prime=1}^{H} w_{hh\prime}h_i^{<t-1>}$$

OR $$z_h^{<t>} = \sum_{i=0}^{I} w_{hi}x_i^{<t>} + \sum_{h\prime=0}^{H} w_{hh\prime}h_i^{<t-1>}$$

If $t = 1$, set $\mathrm{h}^{<t-1>} = 0$

Vanilla RNN Cell

$$h^{<t>} = tanh\left(\mathrm{W}\left(\begin{bmatrix} h^{<t-1>} \\ x^{<t>} \end{bmatrix}\right)\right)$$

Activation:

$$h^{<t>} = g_h(z_h^{<t>})$$

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks

For the output unit:

Weighted Summation:

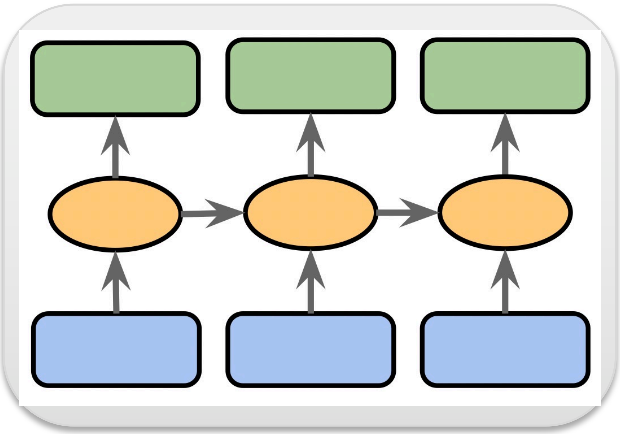
$$z_y^{<t>} = W_{yh} h^{<t>} + b_y$$

OR

$$z_h^{<t>} = \sum_{i=0}^{I} w_{hi} x_i^{<t>}$$

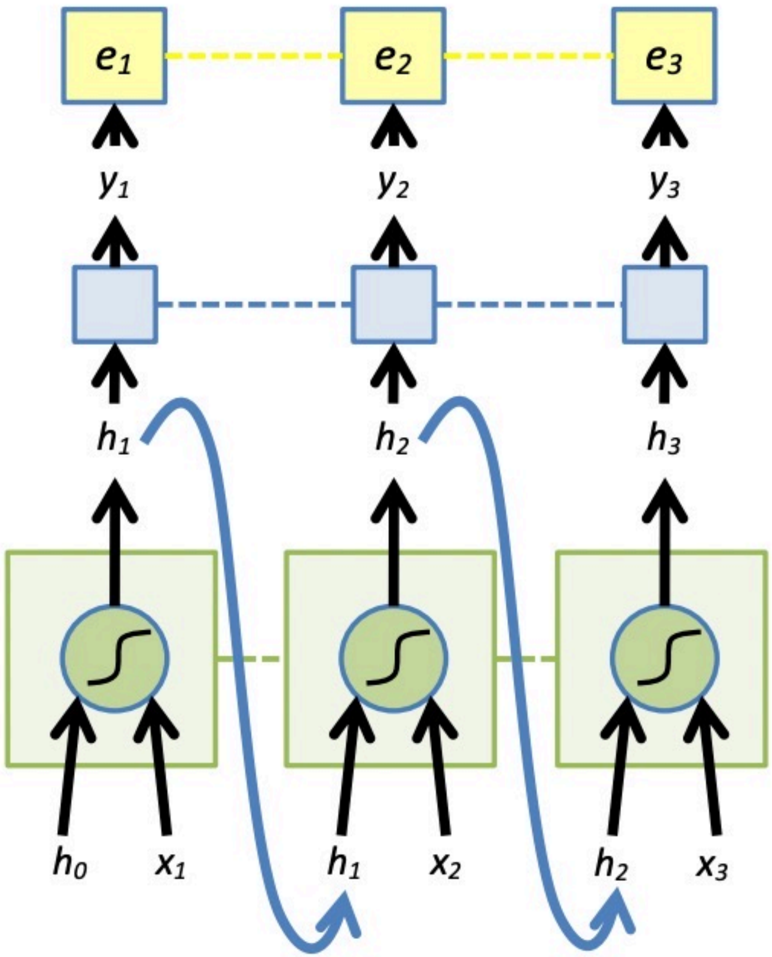
Activation:

$$y^{<t>} = g_y(z_y^{<t>})$$



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Recurrent Neural Networks – Forward Pass



$$e_t$$

$$\mathcal{L} = \sum_{t=1}^{T} e^t$$

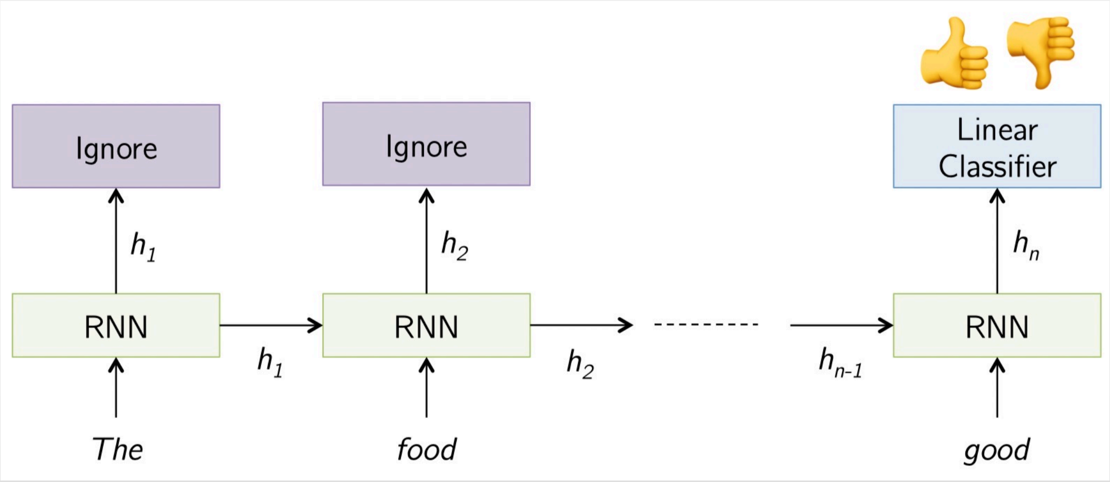
$$y_t = \mathrm{softmax}(W_y h_t)$$

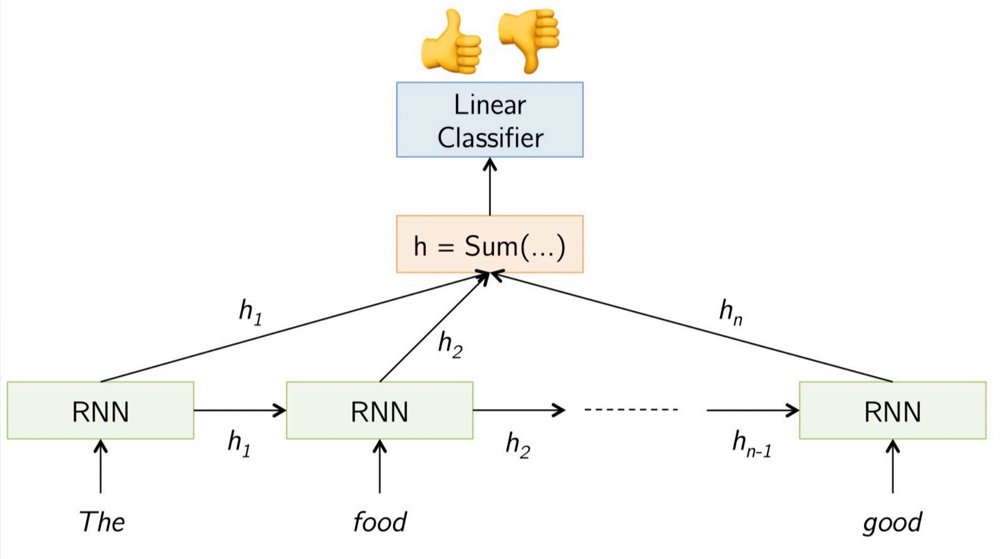
$$h_t = \tanh W \binom{x_t}{h_{t-1}}$$

------ shared weights

https://cs.gmu.edu/~kosecka/cs747/11_rnn.pdf

Recurrent Neural Networks





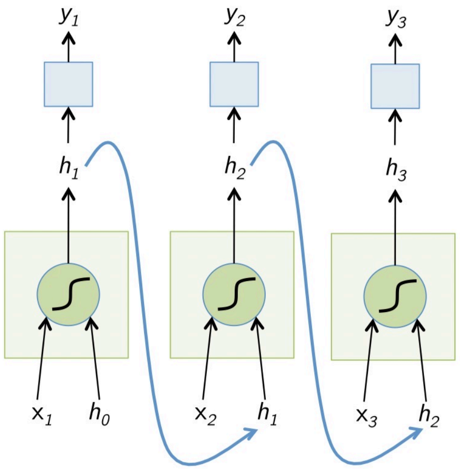
https://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf

Recurrent Neural Networks – BPTT

Backpropagation Through Time (BPTT)

Werbos, Paul J. "Backpropagation through time: what it does and how to do it." Proceedings of the IEEE 78, no. 10 (1990): 1550-1560.

- Most common method used to train RNNs
- Backpropagation is done at each point in time. Don't be fooled by the fancy name. It's just the standard back-propagation.



$$\mathcal{L} = \sum_{t=1}^{T} e^t$$

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial e^t}{\partial y^t} \frac{\partial y^t}{\partial h^t} \frac{\partial h^t}{\partial W_{hh}}$$

$$\frac{\partial h^t}{\partial W_{hh}} = \sum_{k=1}^{t} \frac{\partial h^t}{\partial h^k} \frac{\partial h^k}{\partial W_{hh}}$$

$$\frac{\partial h^t}{\partial h^k} = \frac{\partial h^t}{\partial h^{t-1}} \frac{\partial h^{t-1}}{\partial h^{t-2}} \cdots \frac{\partial h^{k+1}}{\partial h^k} = \prod_{j=k+1}^{t} \frac{\partial h^j}{\partial h^{j-1}}$$

https://harvard-iacs.github.io/2019-CS109B/lectures/lecture10/presentation/cs109b_lecture10_RNN.pdf
https://mmuratarat.github.io/2019-02-07/bptt-of-rnn

Recurrent Neural Networks – BPTT

Problems: Exploding / Vanishing gradient

- Largest singular value > 1: Exploding gradients
- Largest singular value < 1: Vanishing gradients

Solutions:

- The exploding gradient can be fixed with gradient clipping technique
- For vanishing gradients: LSTM or GNU

Long Short-Term Memory units   Gated Recurrent Unit

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Long Short–Term Memory

The LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.

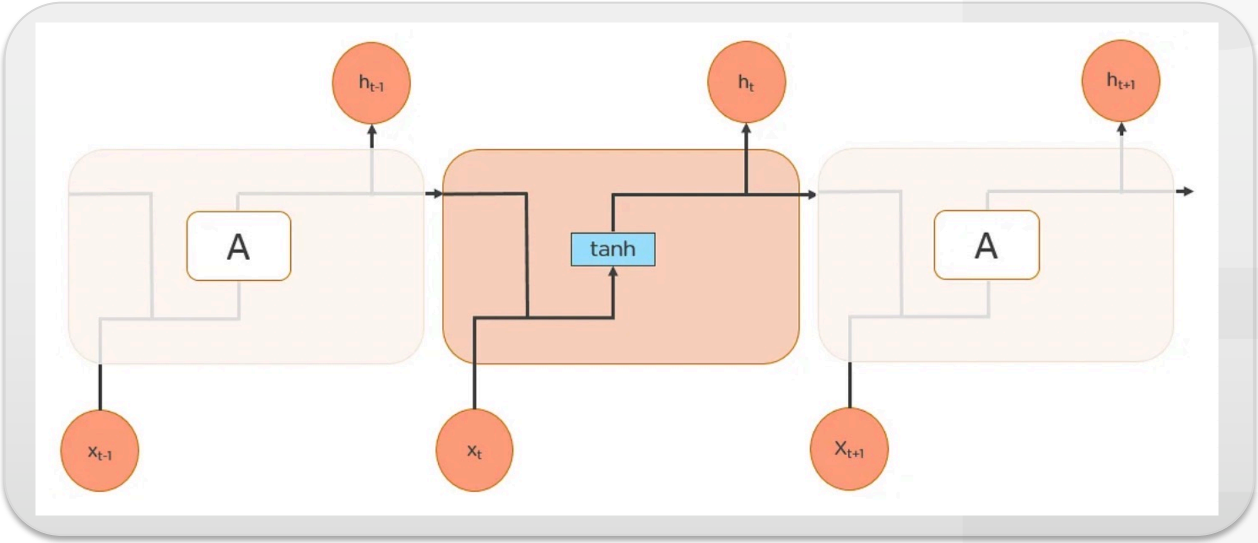
A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

- **Forget Gate:** Whether to erase cell
- **Input Gate:** whether to write to cell
- **Output Gate:** How much to reveal cell

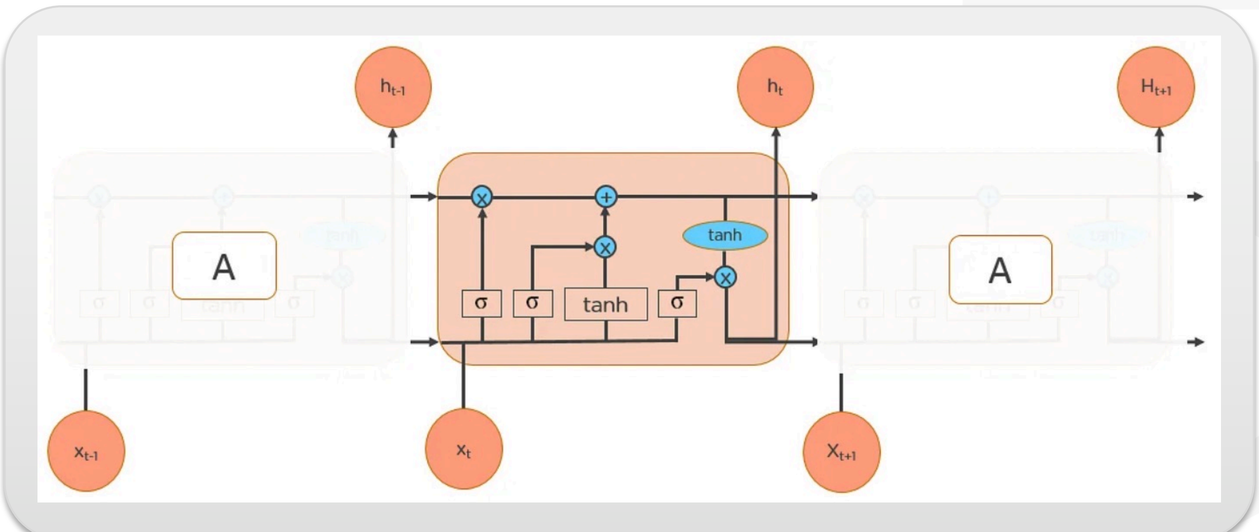
https://en.wikipedia.org/wiki/Long_short-term_memory

Long Short–Term Memory

Vanilla RNN

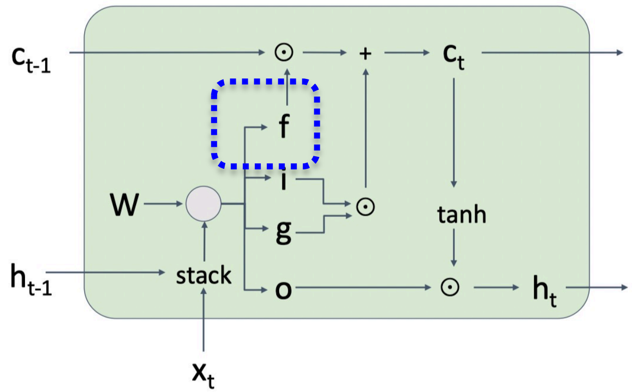


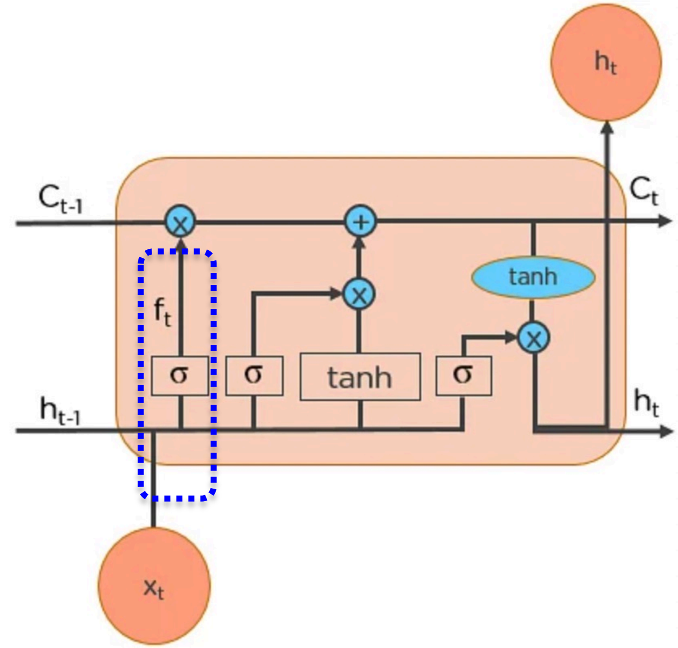
LSTM

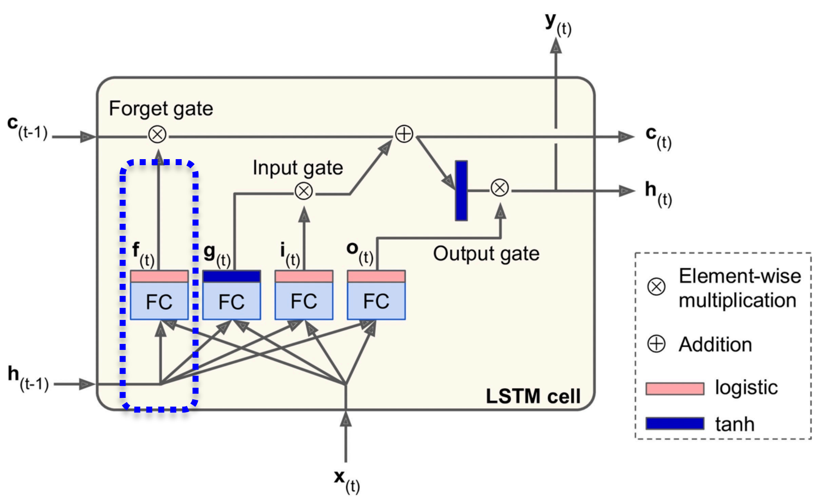


https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Long Short–Term Memory







Step 1: Decide How Much Past Data It Should Remember

The forget gate tries to estimate what features of the cell state should be forgotten.

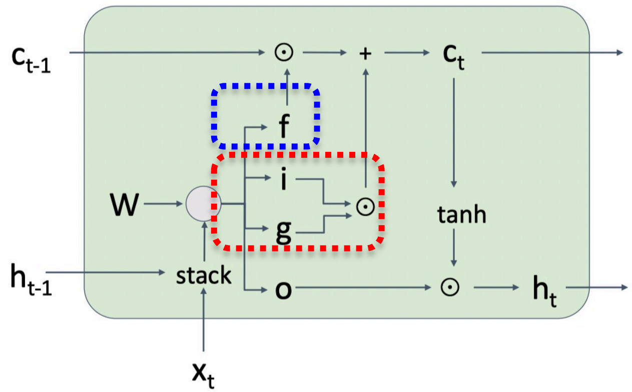
$$f^t = \sigma(W_{hf} \cdot h^{t-1} + W_{xf} \cdot x^t + b_f)$$

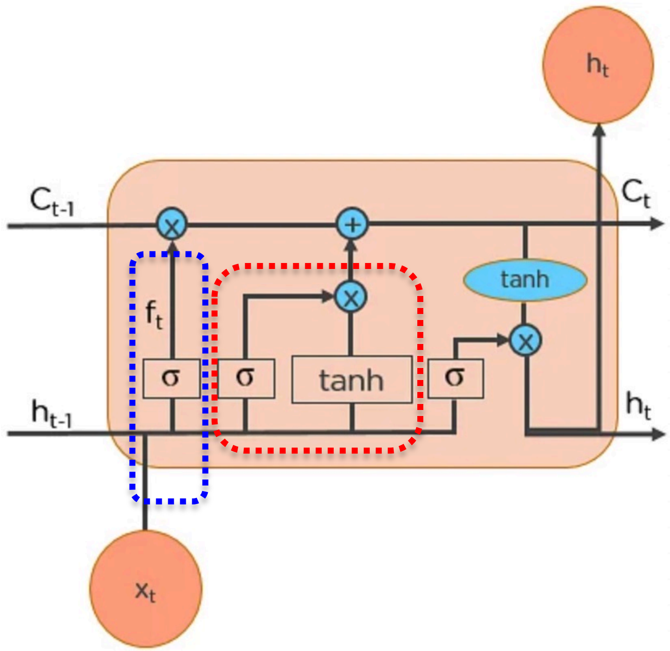
OR

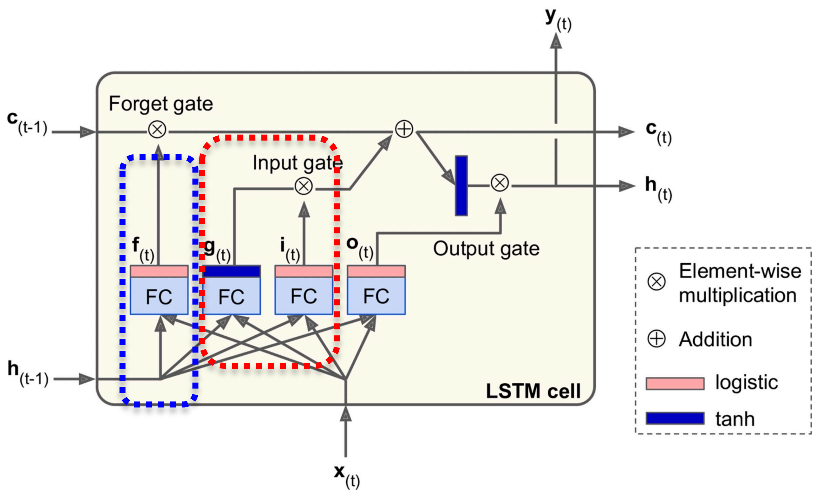
$$f^t = \sigma(W_f \cdot [h^{t-1}, x^t] + b_f)$$

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Long Short–Term Memory







Step 2: Decide How Much This Unit Adds to the Current State

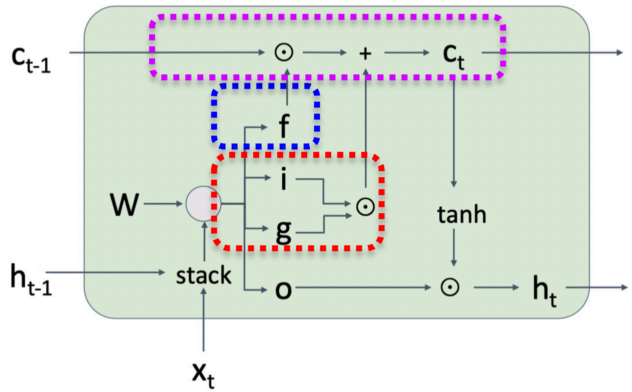
There are two parts. One is the sigmoid function, and the other is the tanh function. In the sigmoid function, it decides which values to let through (0 or 1). tanh function gives weightage to the values which are passed, deciding their level of importance (-1 to 1).

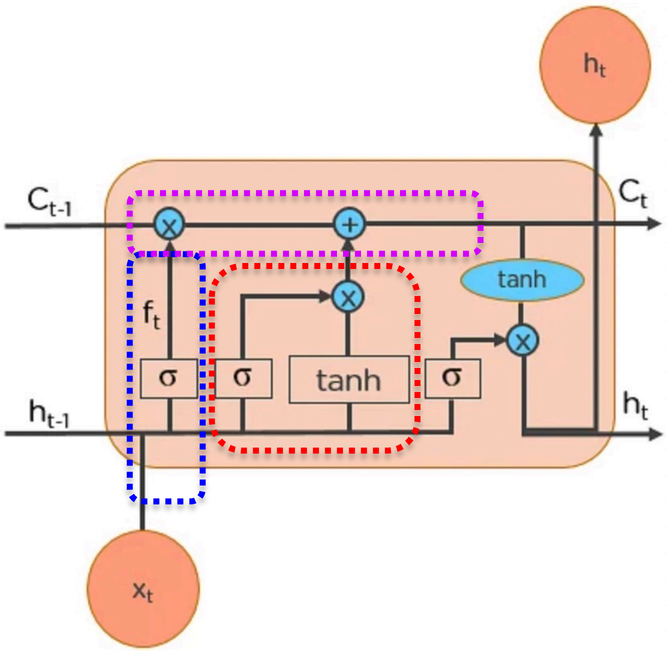
$$\mathrm{i}^t = \sigma(W_i \cdot [\mathrm{h}^{t-1}, \mathrm{x}^t] + \mathrm{b}_i)$$

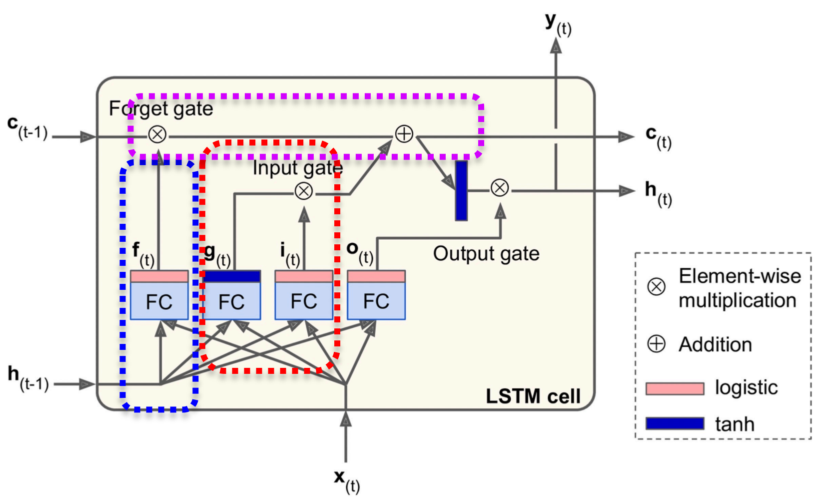
$$\mathrm{g}^t = tanh(W_g \cdot [\mathrm{h}^{t-1}, \mathrm{x}^t] + \mathrm{b}_g$$

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Long Short–Term Memory





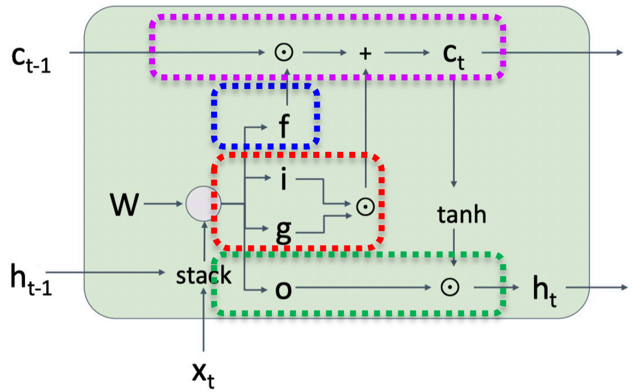


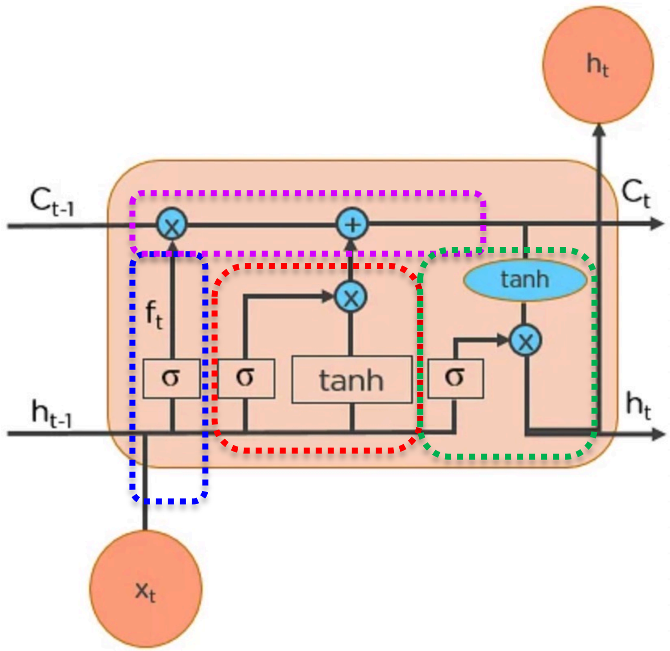
Step 3: Update Cell State

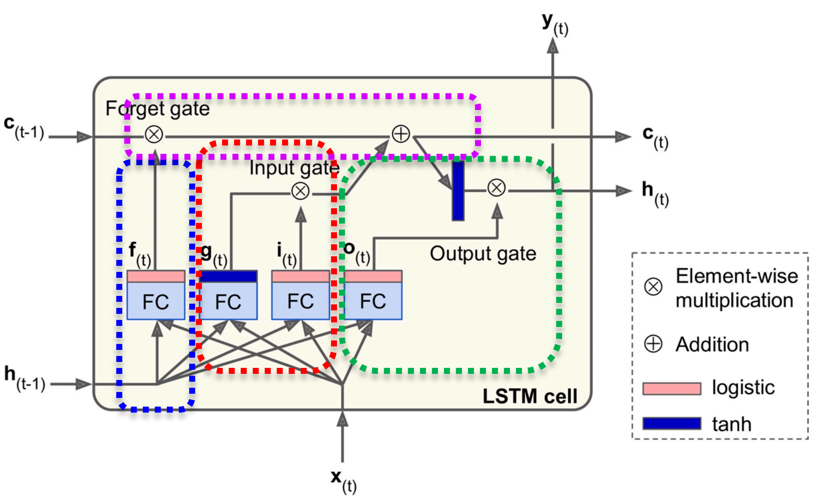
$$c^t = f^t \otimes c^{t-1} + i^t \otimes g^t$$

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Long Short–Term Memory







Step 4: Decide What Part of the Current Cell State Makes It to the Output

First, we run a sigmoid layer, which decides what parts of the cell state make it to the 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.

$$o^t = \sigma(W_o \cdot [h^{t-1}, x^t] + b_o)$$

$$h^t = o^t \otimes \tanh(c^t)$$

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

Long Short–Term Memory

Consequently,

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} + b \right)$$

$$c^t = f \otimes c^{t-1} + i \otimes g$$

$$h^t = o \otimes \tanh(c^t)$$

f_t = forget gate
Decides which information to delete that is not important from previous time step

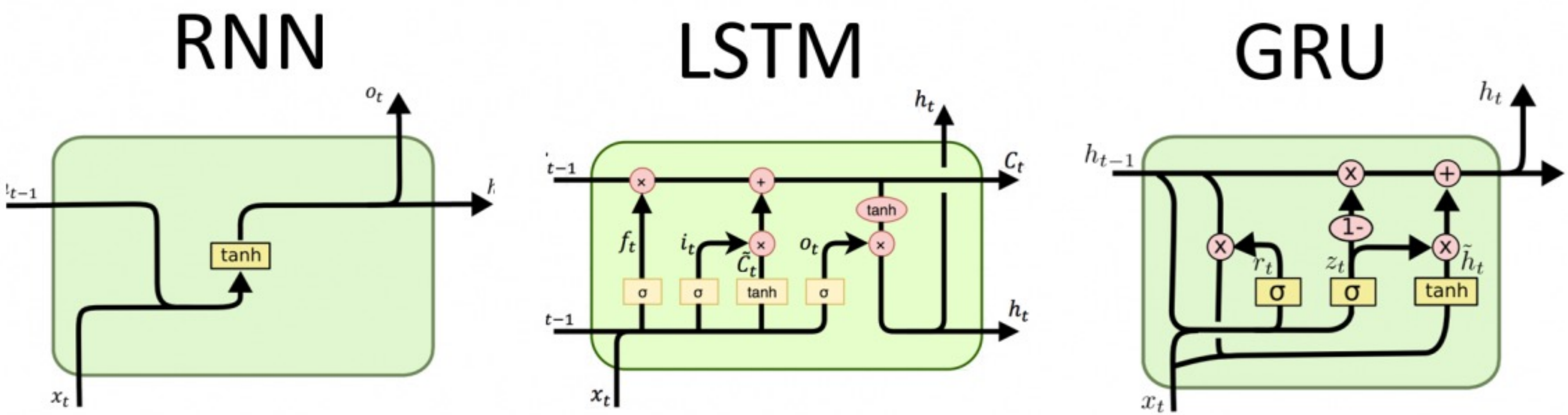
i_t = input gate
Determines which information to let through based on its significance in the current time step

o_t = output gate
Allows the passed in information to impact the output in the current time step

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn?source=sl_frs_nav_playlist_video_clicked

RNN Variants – Gated Recurrent Unit (GRU)

GRU like LSTMs, attempts to solve the Vanishing gradient problem in RNN.



Update Gate:

➢ to determine how much of the past information (from previous time steps) needs to be passed along to the future.

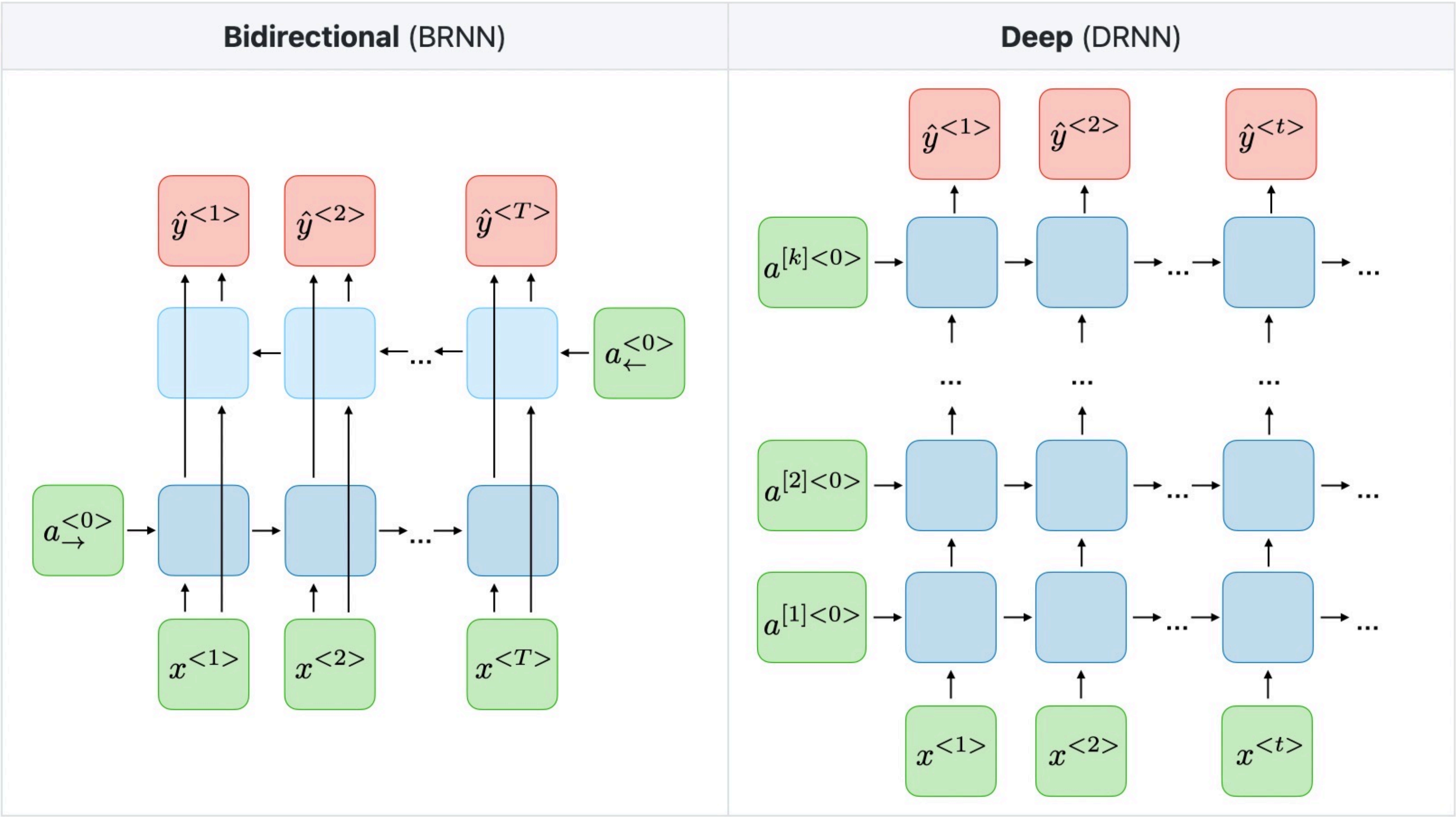
➢ to learn to copy information from the past such that gradient is not vanished.

Reset Gate:

➢ model how much of information to forget by the unit

https://harvard-iacs.github.io/2020-CS109B/lectures/lecture14/presentation/cs109b_lecture_rnns.pdf

RNN Variants



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

RNN Variants

RNNs or Feedback Network come in many variants.

- Hopfield Network
- Boltzmann Machine
- Competitive Network
- Kohonen's SOM
- …

Find details in supplementary reading matierials…

https://en.wikipedia.org/wiki/Recurrent_neural_network