# Neural Networks

Sina K. Maram, M.Sc., P.Eng.

# Neural Networks

Sina K. Maram, M.Sc., P.Eng.

## By the end of this lecture you'd be able to:

- Learn about the architecture of RNN's
- Understand the short-comings of RNN's
- Long-Short-Term-Memory (LSTM) architecture
- Perform Time Series Forecasting using TensorFlow LSTM method
- Apply LSTM on other sequential datasets

# Agenda:

01 **Recurrent Neural Networks**
Architecture and their benefits

02 **Shortcomings of RNN's**
Network Size, Vanishing Gradient

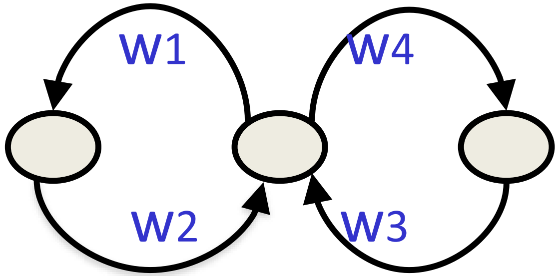
03 **LSTM Architecture**
Architecture, and TensorFlow methods

04 **Class Exercise – Application of LSTM on time Series**
CNN in TensorFlow, MNIST Example

01

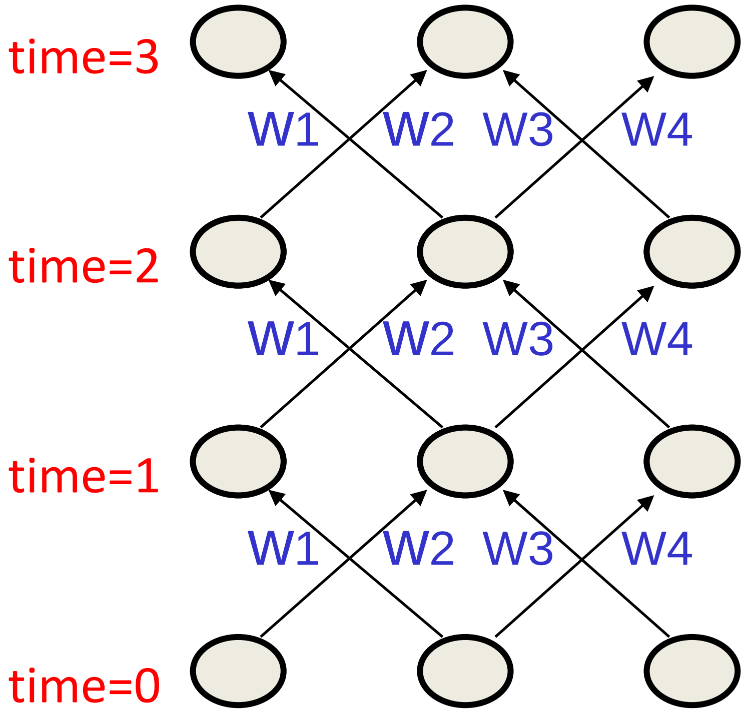
**Recurrent Neural Networks**

Architecture and their benefits




Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.



01

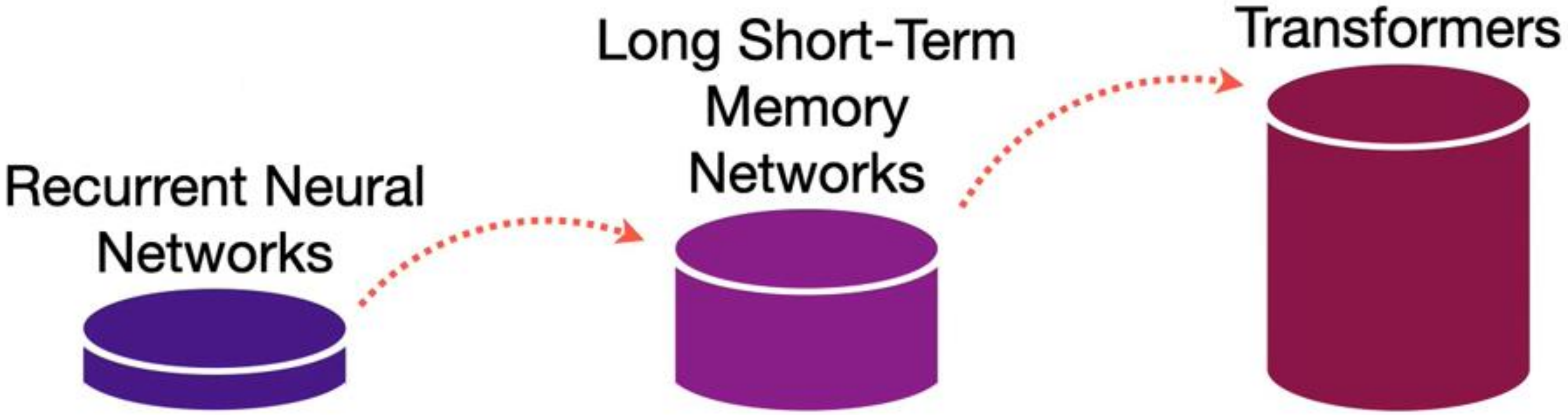
## Recurrent Neural Networks

Architecture and their benefits

https://www.youtube.com/watch?v=AsNTP8Kwu80

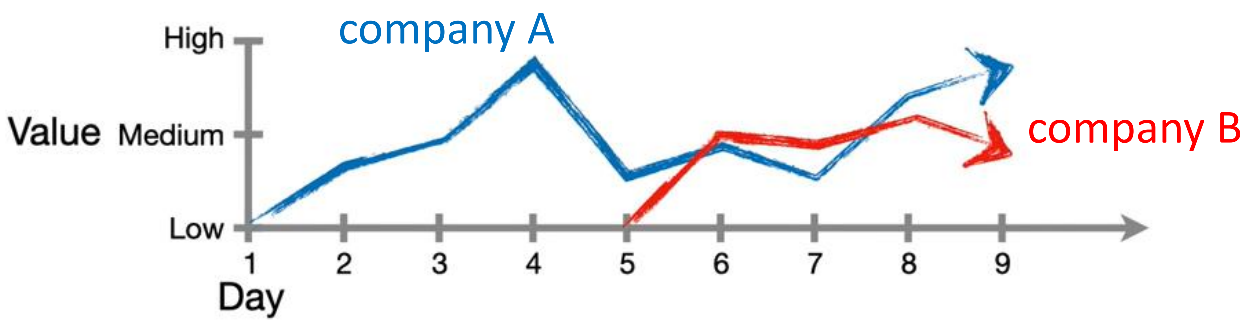
Explains RNN
Concept and unrolling
Explains Vanishing/Exploding Gradient Descents

- **R**ecurrent **N**eural **N**etworks are known to be the stepping stone of more complicated Neural Network architectures such as Long Short-Term Memory Networks and Transformers



- Lets start our topic by stock market prices example.
- Let's say we want to make a model that looks at the sequential data of stock market for a given company and predicts the next day stock price.

- Challenge:
  - We have more sequential data for company A than company B. So how can we ensure that our model is built in such way that takes all these "**<u>SEQUENTIAL</u>**" data regardless of their size.

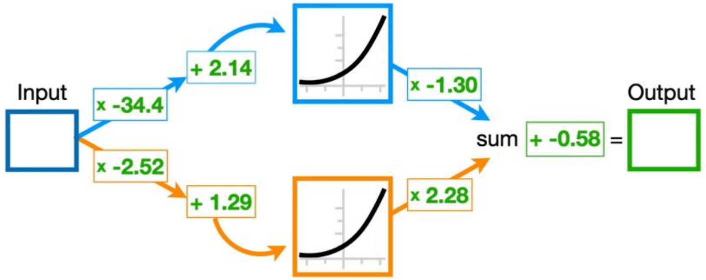


# Recurrent Neural Networks

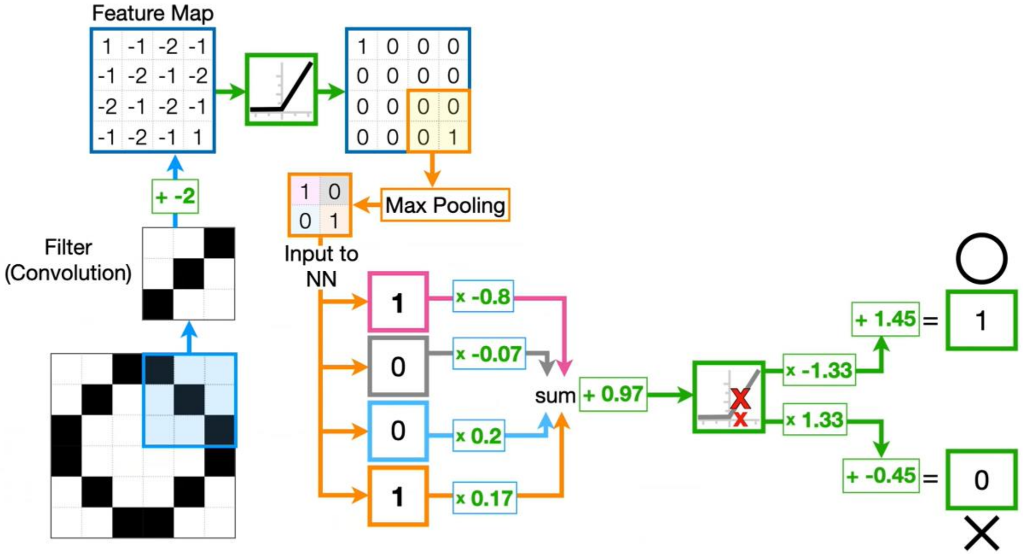
Architecture and their benefits

- In a typical Neural Network architecture (Shown below) We are only dealing with fixed amount of inputs (as shown in the graphs below)

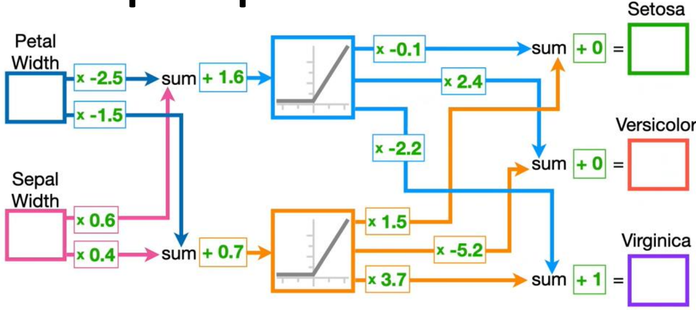
## Single Input Network




## CNNs



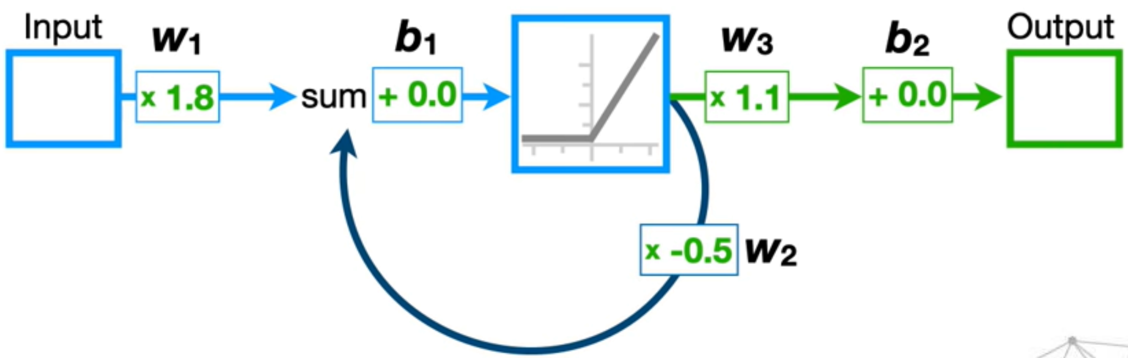

## Multiple Input

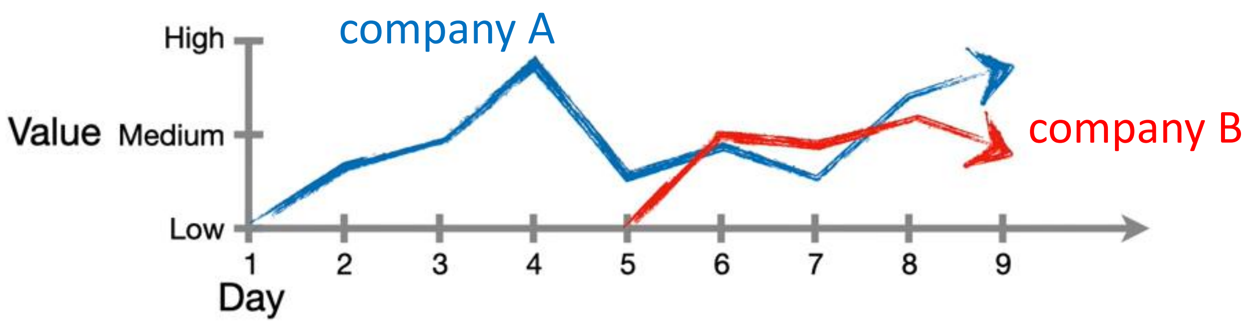


- Recurrent Neural Network addresses this issue:

- Similarity with other architectures:
  - RNN's have weights, nodes and biases similar to DNNs
  - Forward and Back-propagations works the same

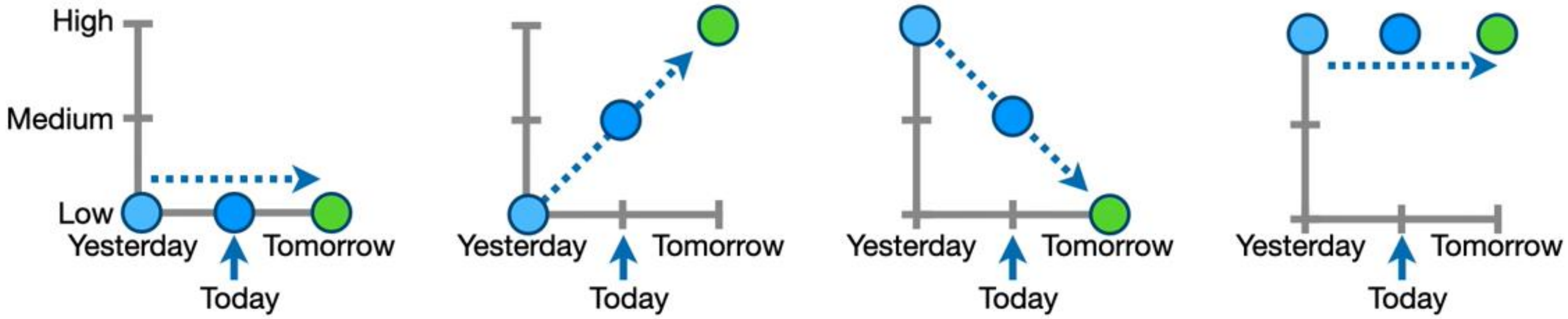
- Difference:
  - The main difference is the feedback loop as shown in the image

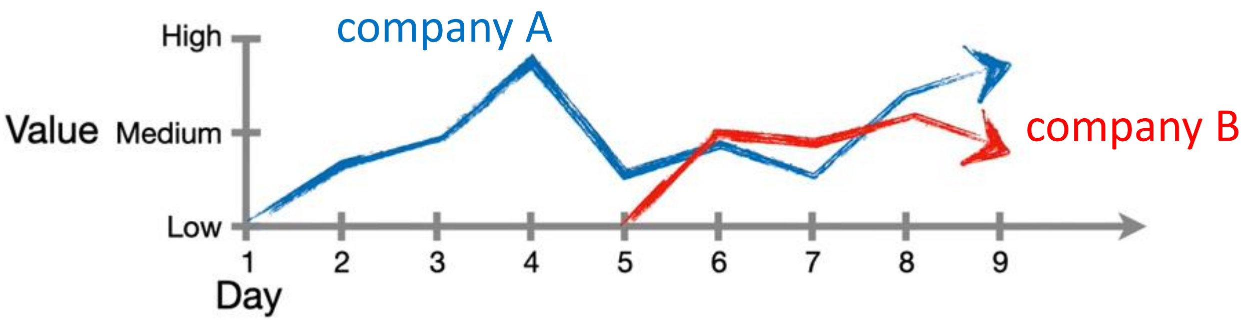


Let’s assume there are the following scenarios when prediction stock price. Let’s assign values for each stock price and run them in our imaginary RNN network to see how they work.

1.0
0.5
0.0

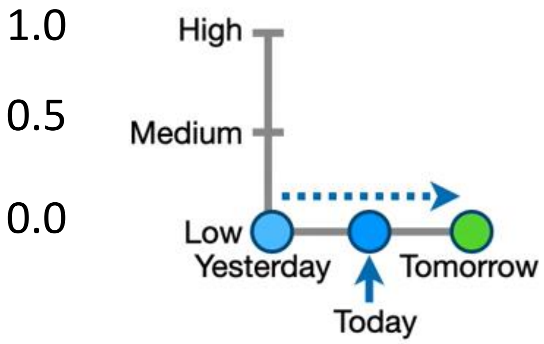


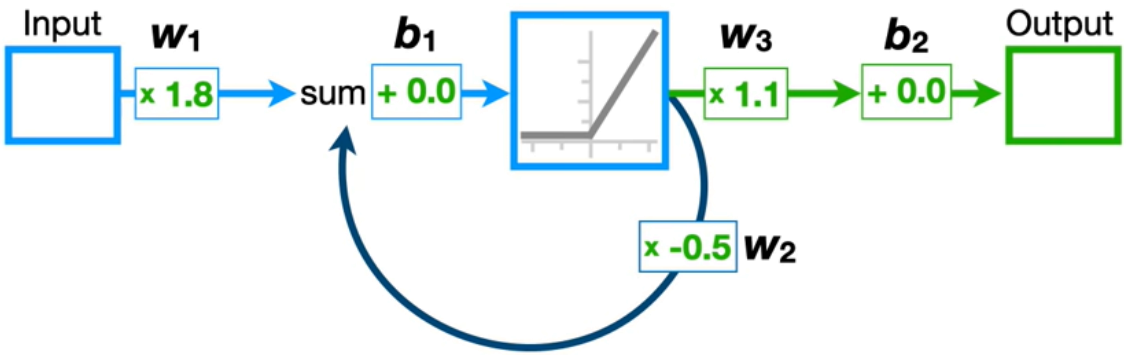



01

## Recurrent Neural Networks

Architecture and their benefits



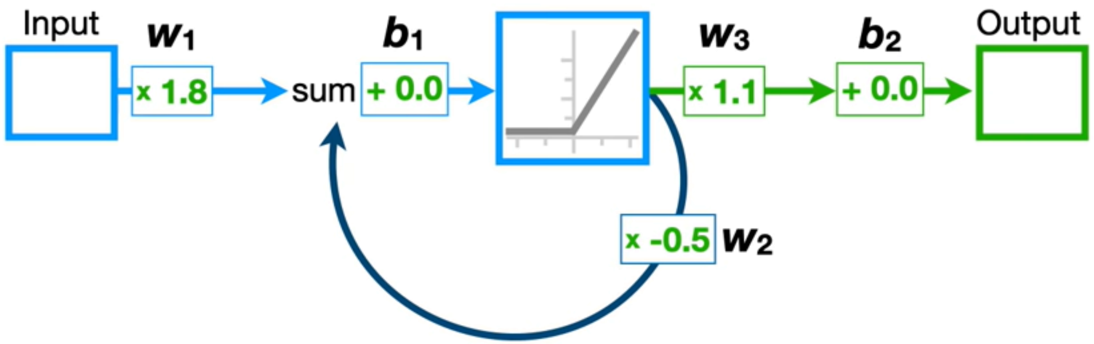

In a typical Neural Network, Yesterday data will be the input to this network and the Output will be Today's prediction. So in order to predict "Tomorrow", you can only pass Today's data to the network. This means you would get any insights from historical days.

01

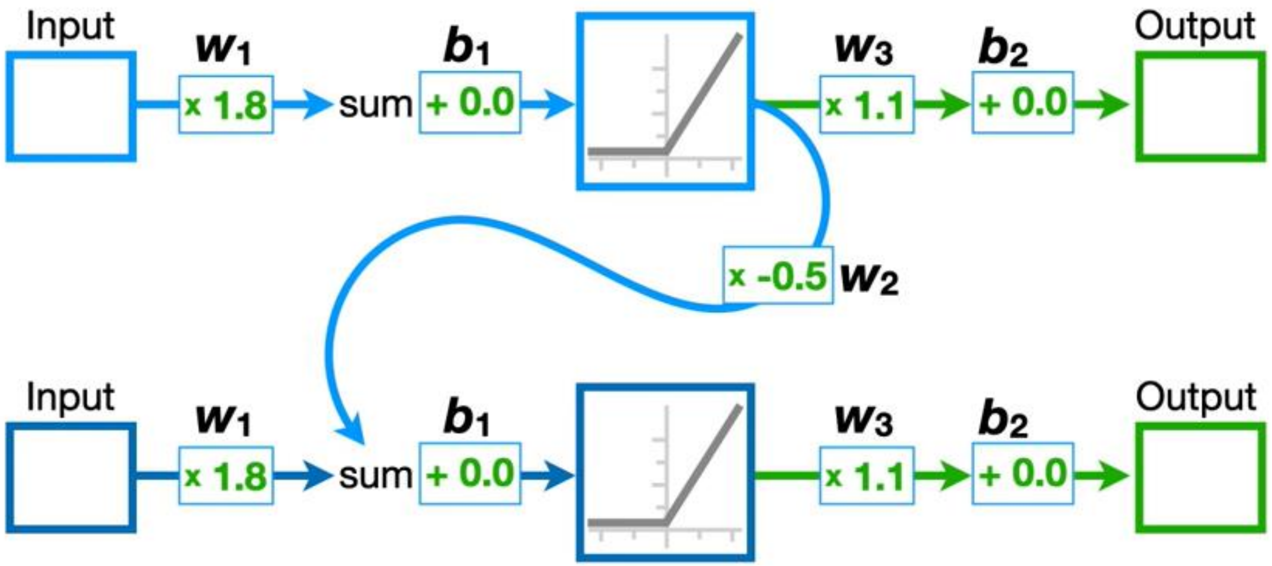
## Recurrent Neural Networks

Architecture and their benefits

### Unrolling RNN:




### Is the same as:



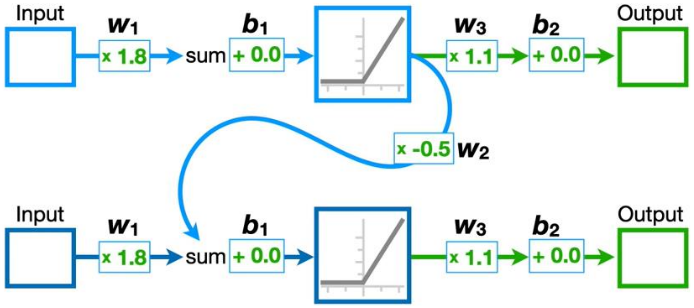
01

## Recurrent Neural Networks

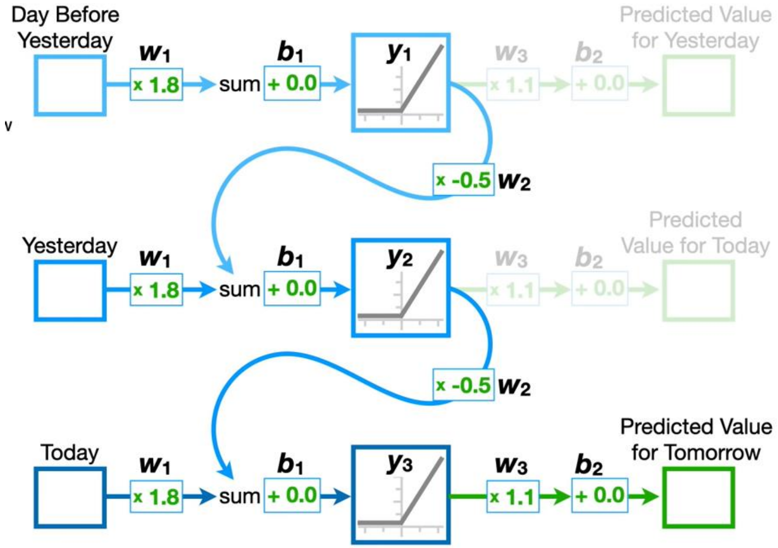
Architecture and their benefits

### Unrolling RNN:

So technically by unrolling the network we have a new architecture that has two inputs and two outputs.




But what if we have 3 days of data?

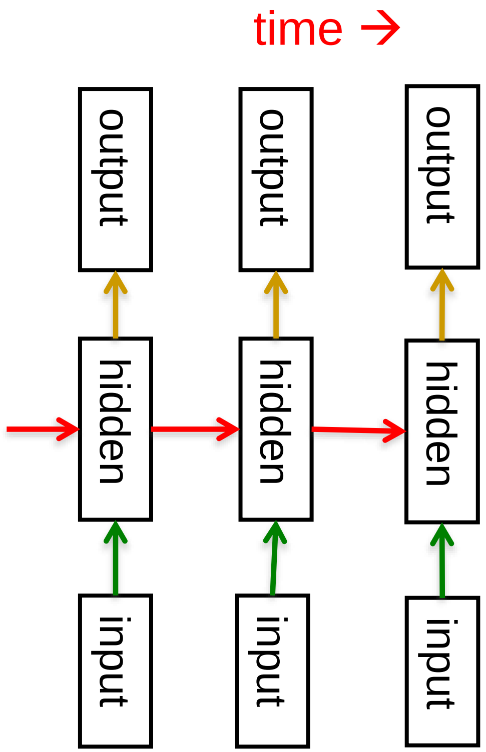


# Recurrent Neural Networks

Architecture and their benefits

**Key Takeaways on RNNs:**

1. Regardless of how many times we unroll a recurrent neural network, the weights and biases are shared across every input. This means regardless of how many days in the past we keep looking, we never increase the number of weights and biases that we need to train. For instance, in the example we reviewed earlier we are only training 5 parameters (W1, W2, W3, b1 and b2)




RNNs are very powerful, because they combine two properties:

1. Distributed hidden state that allows them to store a lot of information about the past efficiently.

2. Non-linear dynamics that allows them to update their hidden state in complicated ways.

02

## Shortcomings of RNN’s

Network Size, Vanishing Gradient

So, if RNNs are so powerful, why they are not used extensively?

1. **Vanishing/Exploding Gradient problem**

2. **The computational power of RNNs makes them very hard to train**

   For many years we could not exploit the computational power of RNNs despite some heroic efforts (e.g. Tony Robinson’s speech recognizer).

02

## Shortcomings of RNN’s

Network Size, Vanishing Gradient

### Vanishing/Exploding Gradient problem

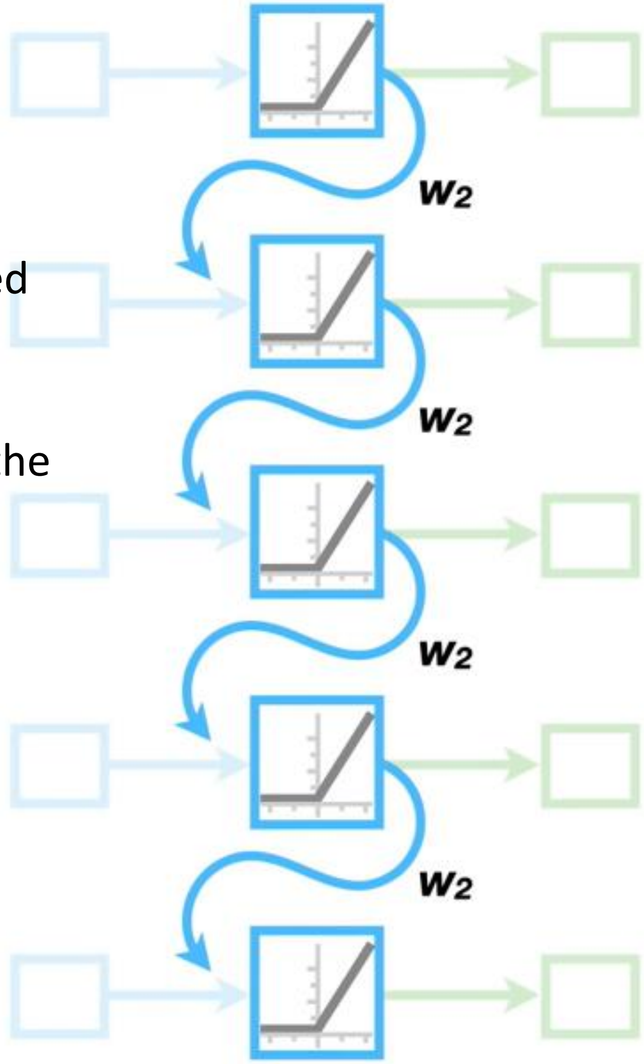
Exploding Gradient:

Let’s assume that our model has set the W2 = 2.0. So the output from the RELU unit from first timestamp, is multiplied by the weight (2) every time it goes through the next roll of network.

In another word, in the n-th roll of the network (n-th day), the output contributing only to day 1 is:

$$INPUT\ DAY\ 1 \times W_2^{Number\ of\ unrolls}$$

$$\frac{d\,SSR}{d\,w_1} = \frac{d\,SSR}{d\,Predicted} \times \frac{d\,Predicted}{d\,y_1} \times \frac{d\,y_1}{d\,x_1} \times \boxed{\frac{d\,x_1}{d\,w_1}} + \ldots$$



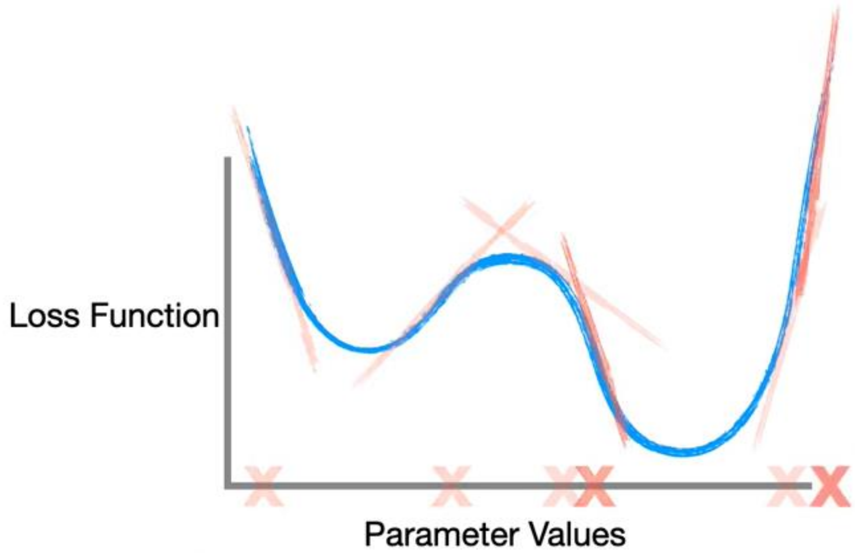

**So what does this mean in the context of backpropagation?**

02

## Shortcomings of RNN’s

Network Size, Vanishing Gradient

Gradient explosion results the model not to be able to converge into a minima because the gradient value is so large that at every iteration we move past the minima. Similar to a scenario where your learning rate is too high




**So do you think limiting weights to values less than 1 will help resolving gradient explosion?**

02

## Shortcomings of RNN's

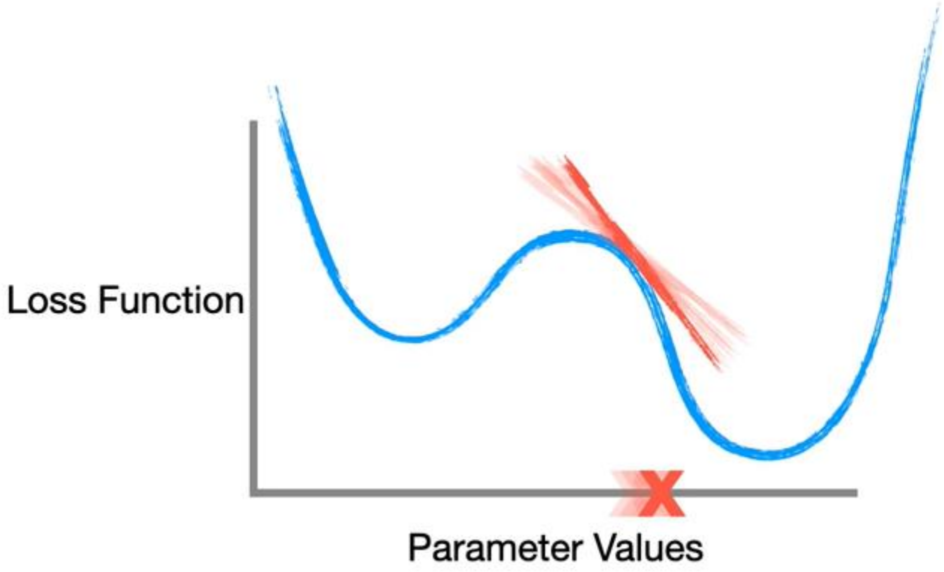
Network Size, Vanishing Gradient

Weights less than 1 will result into a completely an opposite issue! Vanishing Gradient

In another word and by looking back at:

$$INPUT\ DAY\ 1\ \times\ W_2^{Number\ of\ unrolls}$$

The gradient will converge to zero and its impact on backpropagation calculation is that we are moving very slowly across the curve toward the local minima which is inefficient

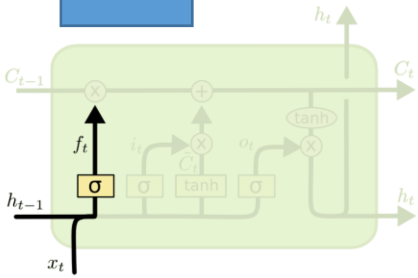



**That's where LSTM (Long Short-Term Memory) Architectures become handy!**

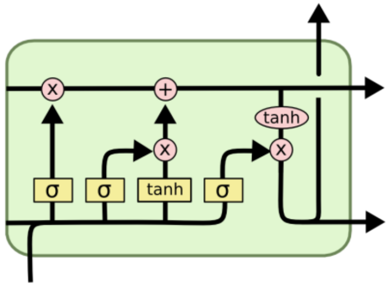
03

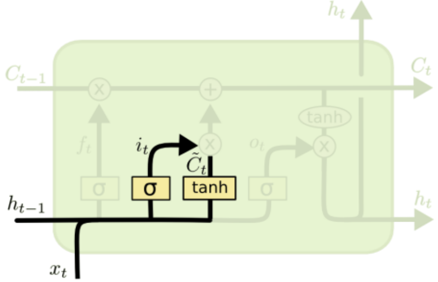
# LSTM Architecture

Architecture, Application and TensorFlow metho



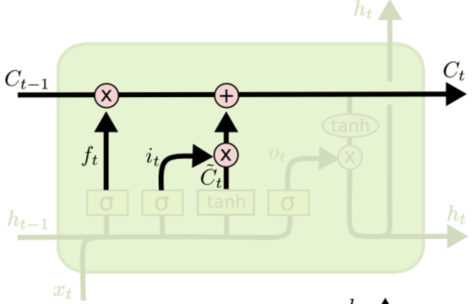
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$





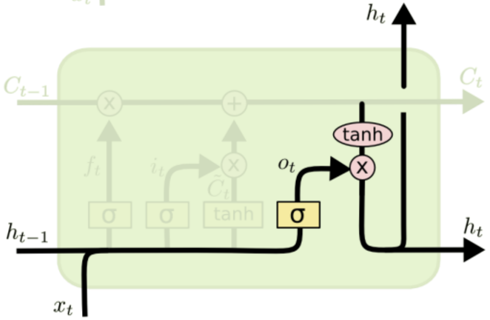
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$i_t$ decides what component is to be updated.
$C'_t$ provides change contents



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

Updating the cell state



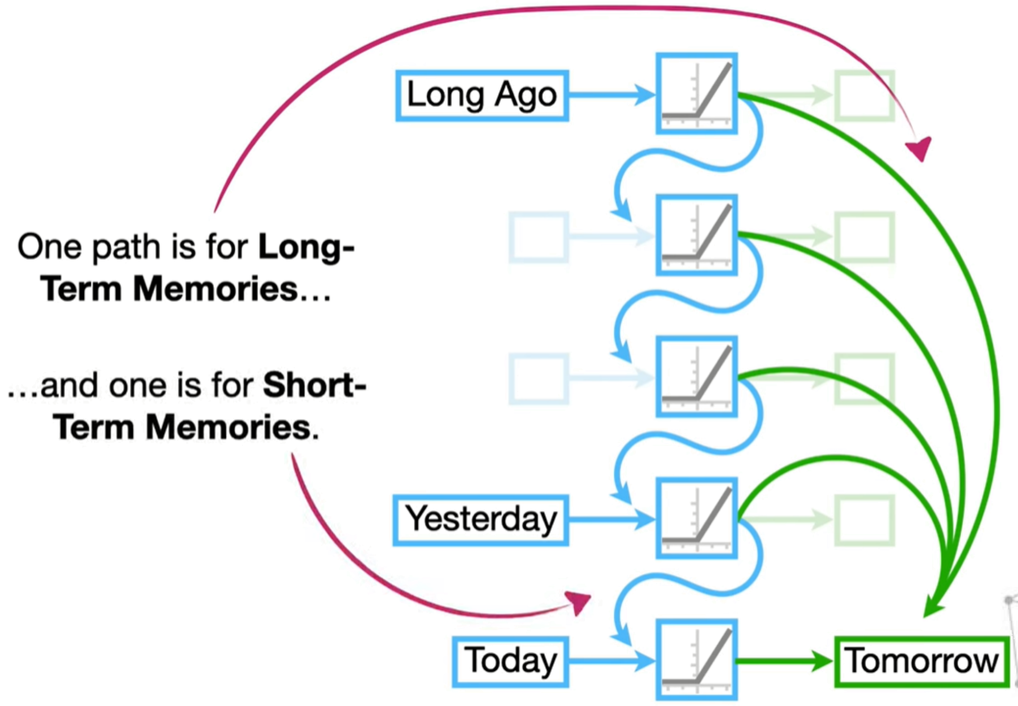
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what part of the cell state to output

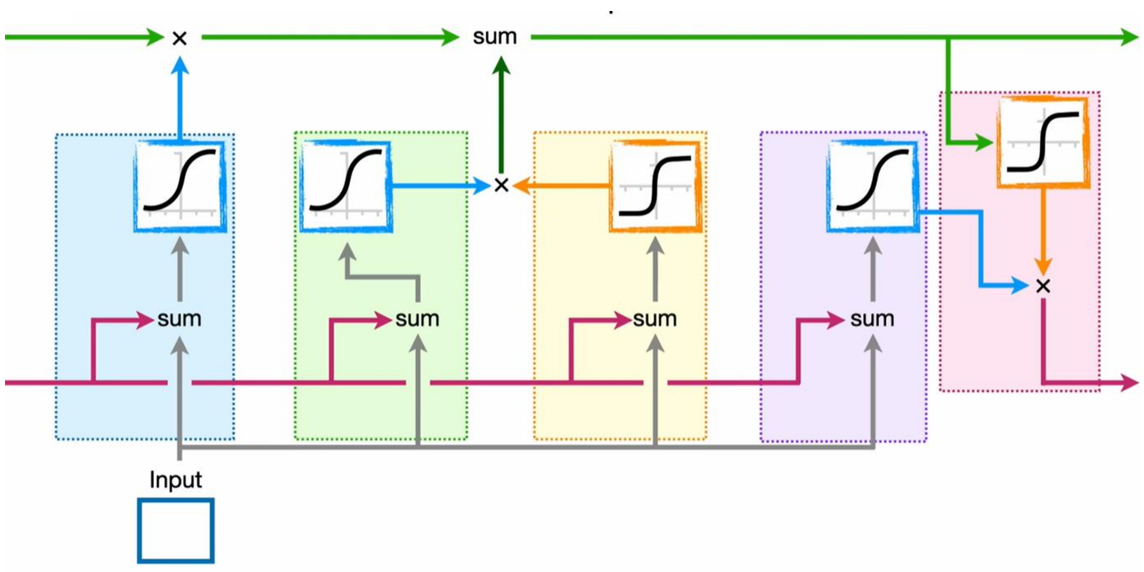
Long Short-Term Memory architecture is there to address the gradient vanishing/explosion caused by passing over older signals through the same node repeatedly.

The node structure for LSTM is designed is comprised of:

- A path for Long-term memory
- A path for Short-term memory



This is how a node for an LSTM network looks like:




Couple of takeaways:

1. The activations used in LSTM network are:

- Sigmoid
- TanH

Whereas in RNN's this was a RELU activation function

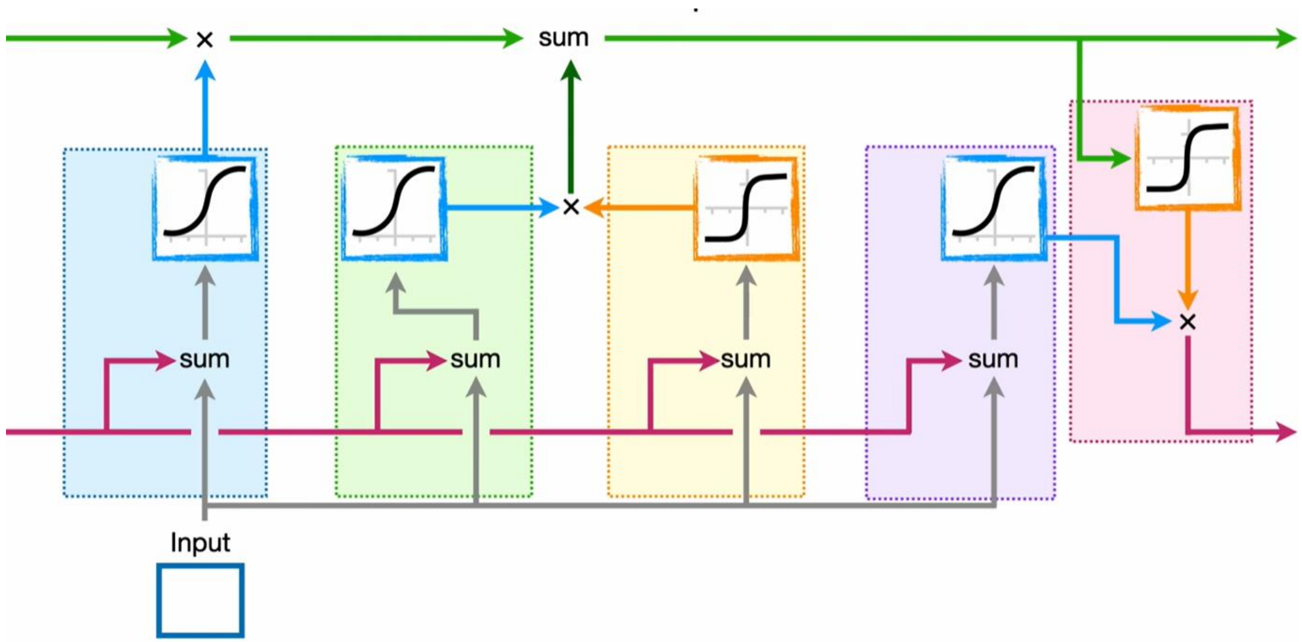
03

**LSTM Architecture**

Architecture, Application and TensorFlow methods

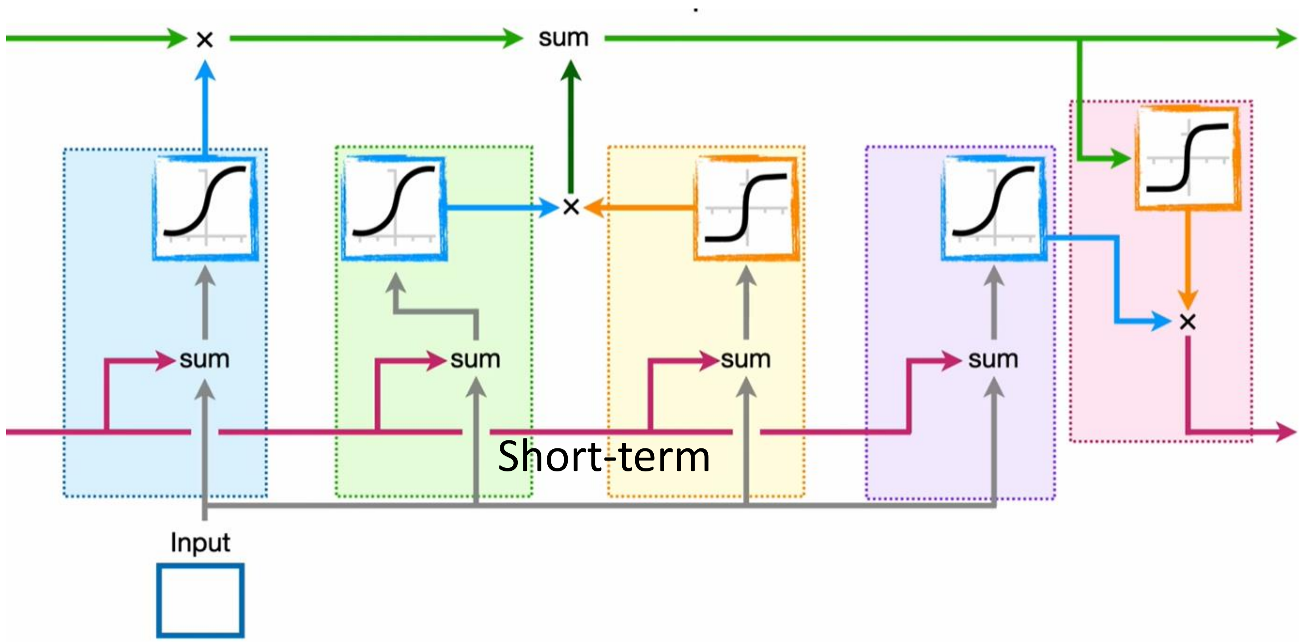
This is how a node for an LSTM network looks like:

Cell State




- Cell States represent Long Term Memory
- There are no weights or biases associated to it however its value gets affected by multiplication and summation of other signals within the cell
- Because of the reason mentioned above, the cell state can get rolled into different nodes without the risk of vanishing or exploding gradient

This is how a node for an LSTM network looks like:



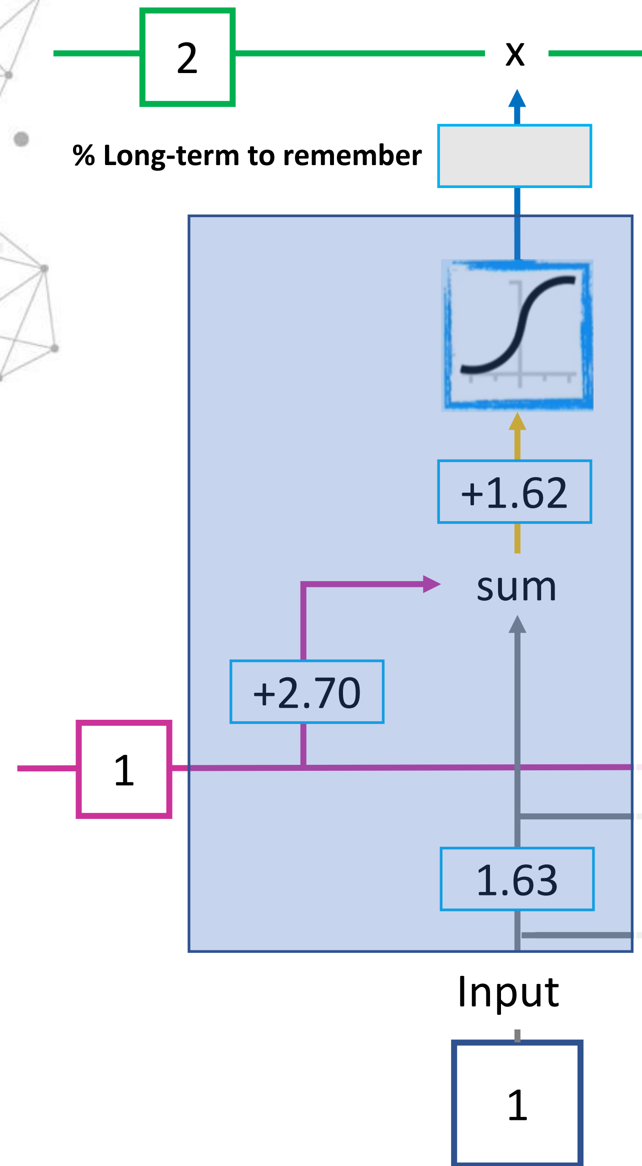

- Purple line represents short-term state
- They are directly connected to weights that can modify them

03

## LSTM Architecture

Architecture, Application and TensorFlow methods

This is how a node for an LSTM network looks like:




The highlighted block of LSTM architecture is called **THE FORGET GATE**
This gate, ironically is responsible for determining how much of the long-term signal is being remembered!

**Let's see how it works:**

Let's assume:

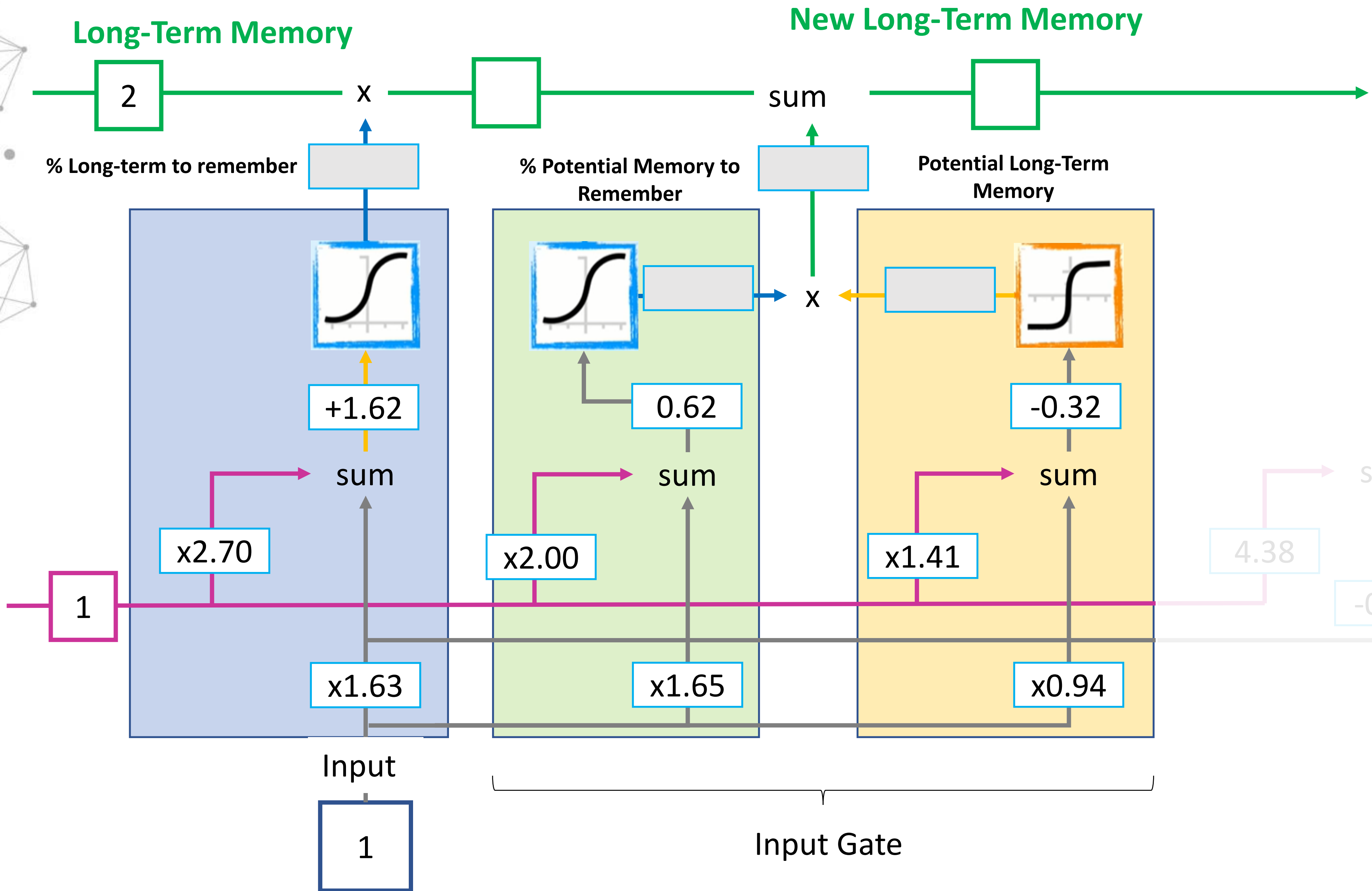
- the output coming from Long Term memory gate is: 2
- The output coming from Short term memory gate is: 1
- And input is: 1

**QUESTION: What happened if the input = -10**

03

## LSTM Architecture

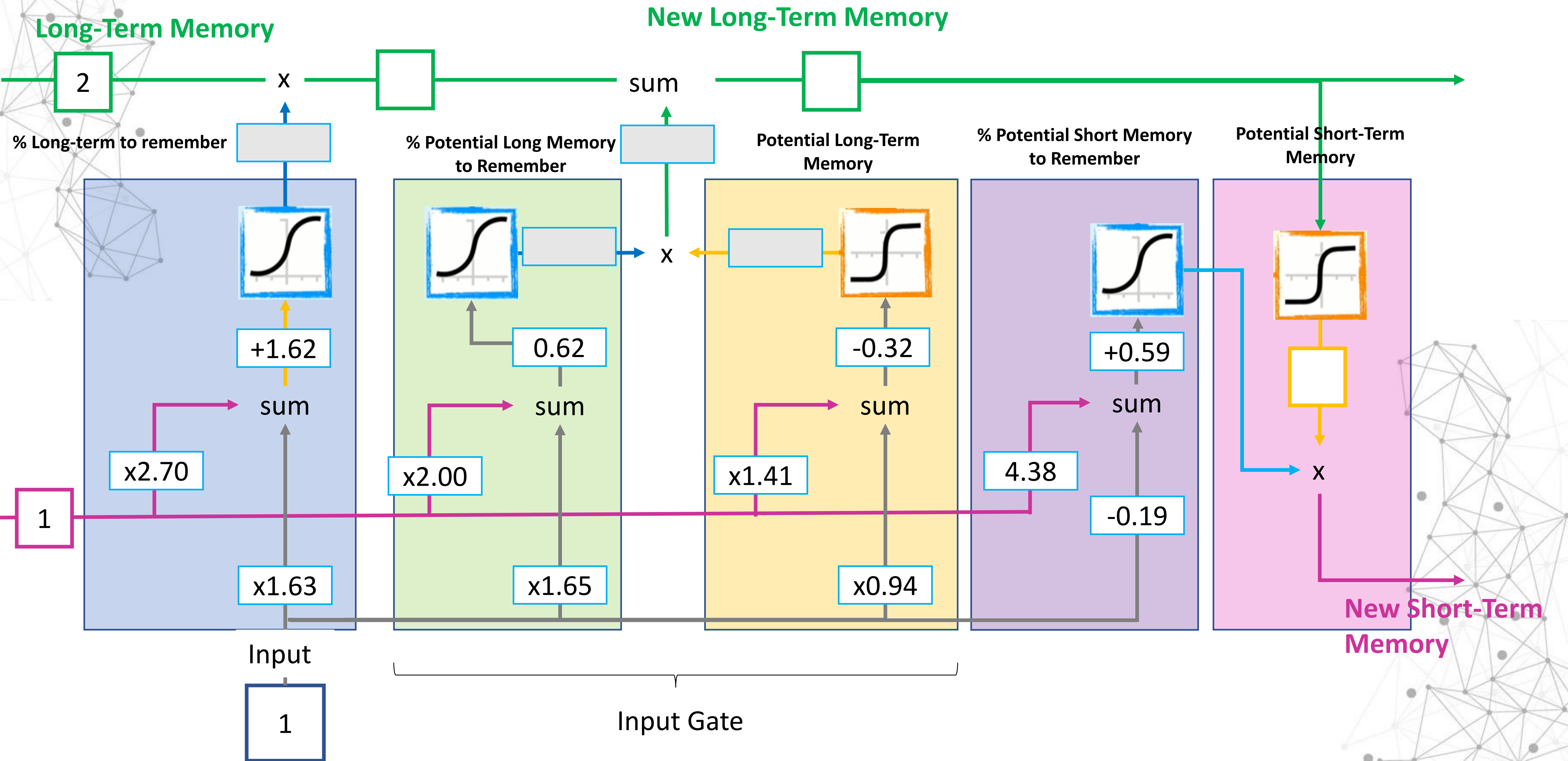
Architecture, Application and TensorFlow methods



03

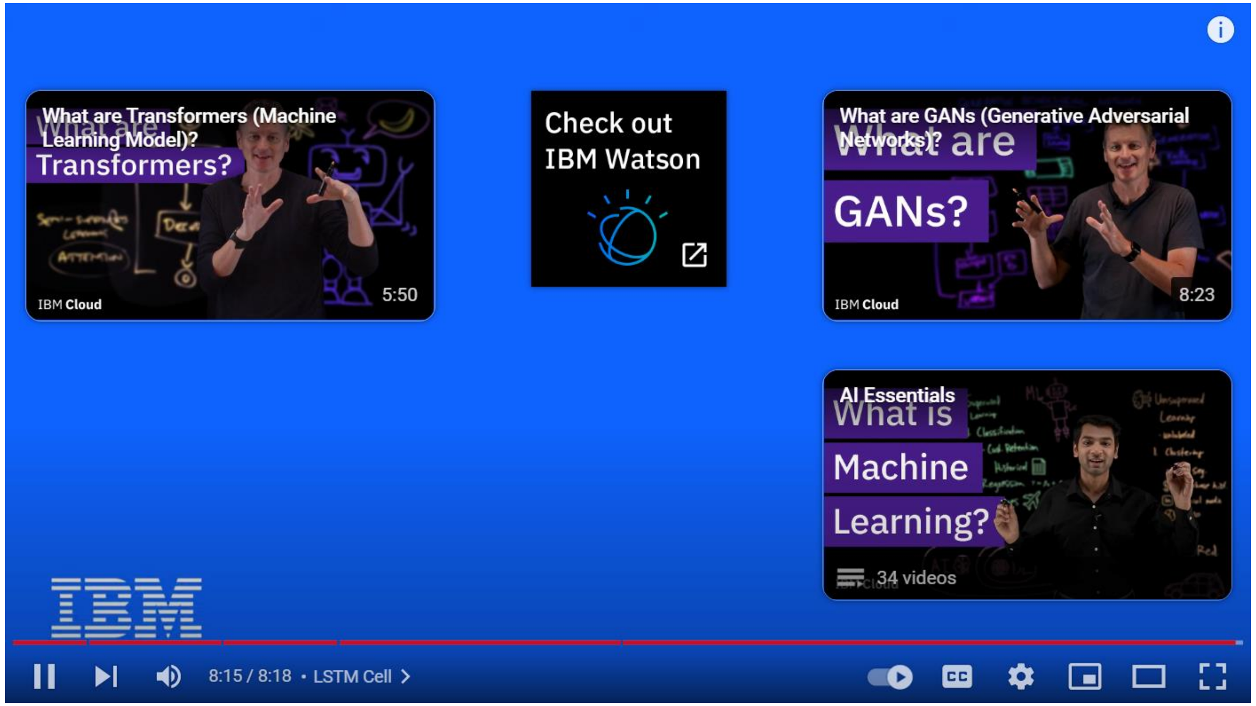
# LSTM Architecture

Architecture, Application and TensorFlow methods



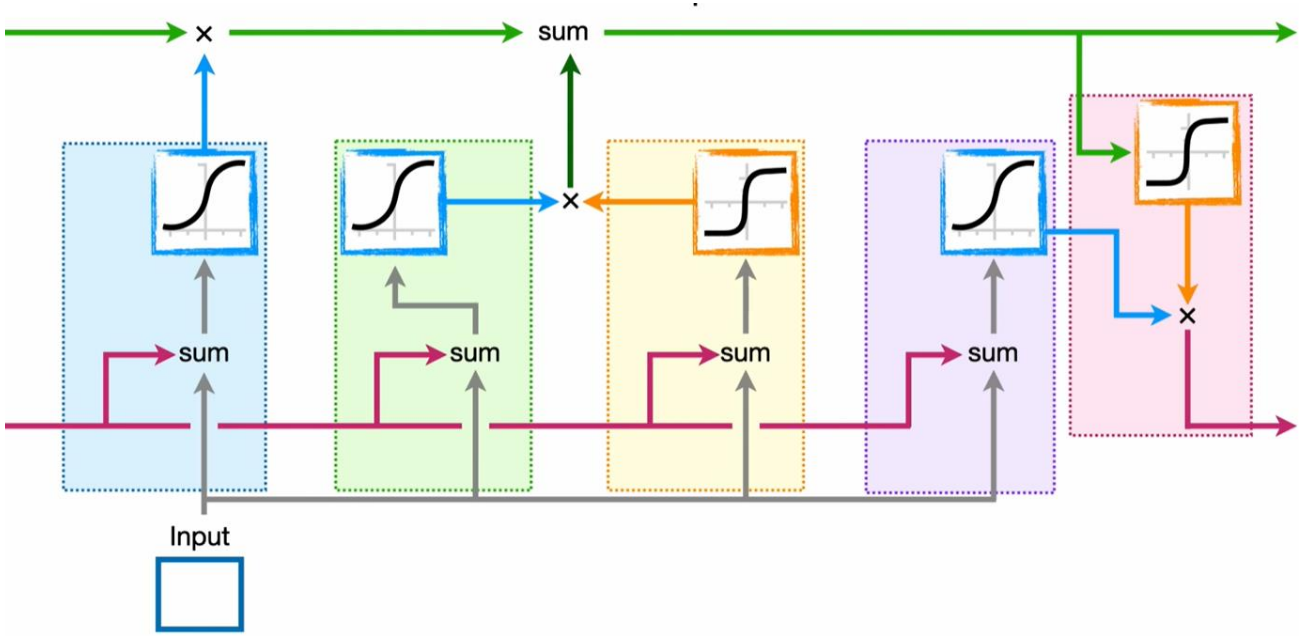
# **IBM** Description: (Video to append)

https://www.youtube.com/watch?v=b61DPVFX03I



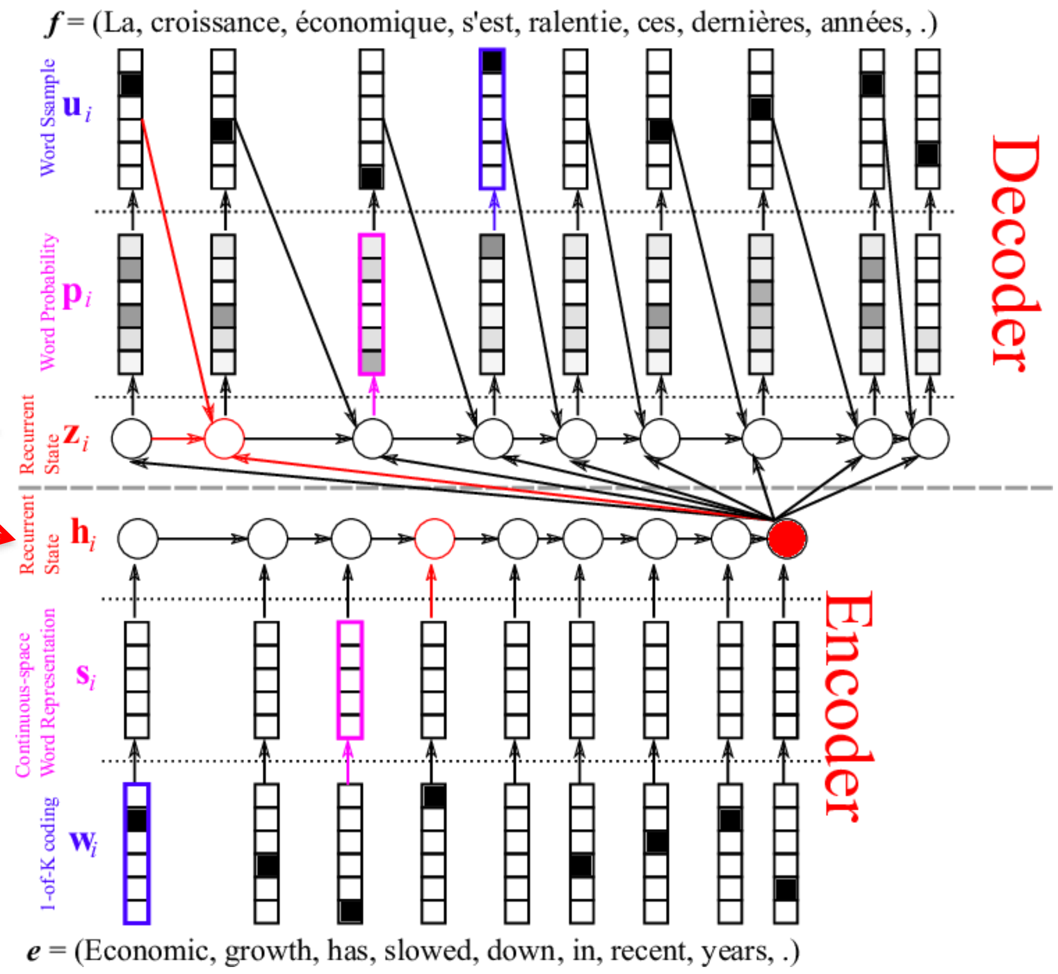
# Class Exercise 1

# Home-Made LSTM Cell!



## Neural Machine Translation

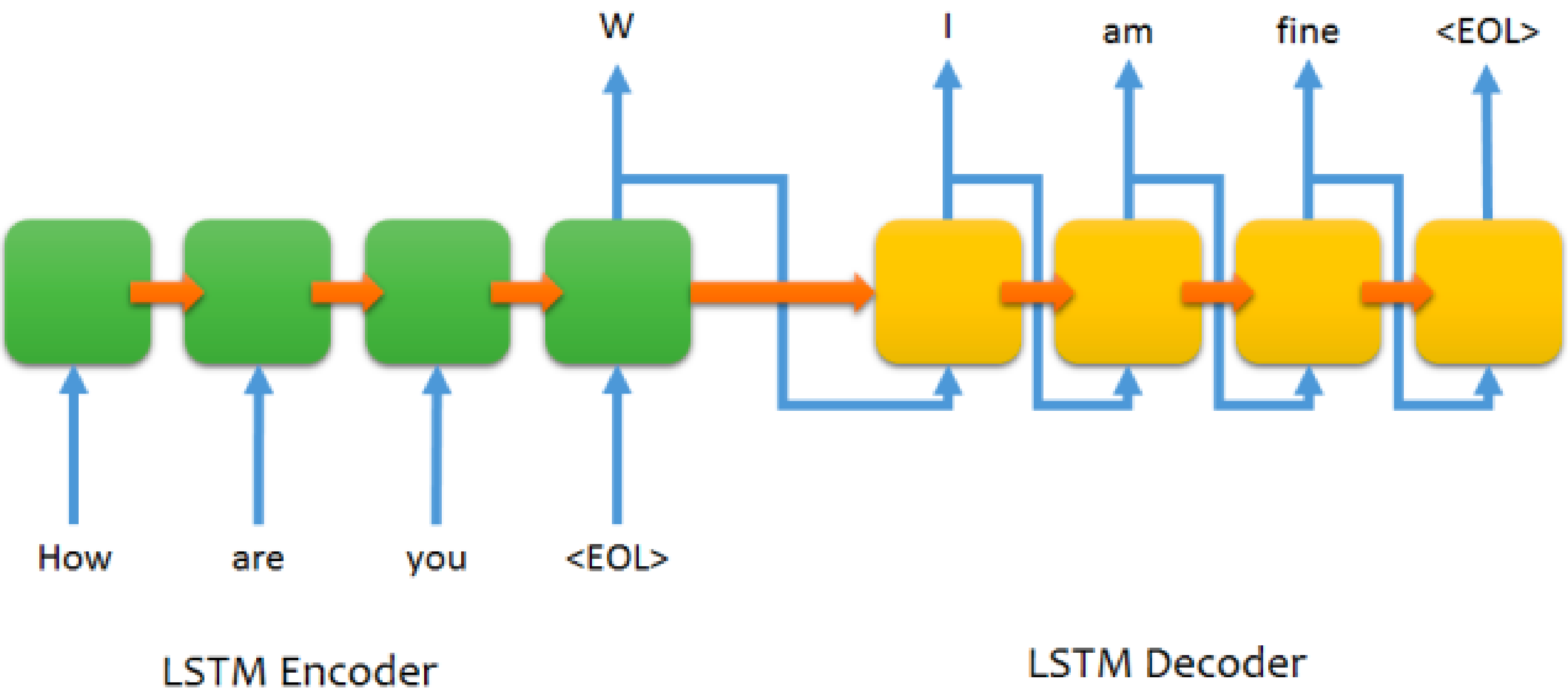
LSTM



03

**LSTM Architecture**
Architecture, Application and TensorFlow methods

## Sequence to Sequence Chat Model



# Class Exercise 2

# Application of LSTM on time Series

# Home Exercise

# Music Composition