KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION - ACCREDITED BY NAAC WITH 'A' GRADE
Narayanaguda, Hyderabad.

Embedded Learning
Day 3
RNN

23-01-2025

BY
ASHA M
ASSISTANT PROFESSOR
CSE(AI&ML)
KMIT

CONTENTS

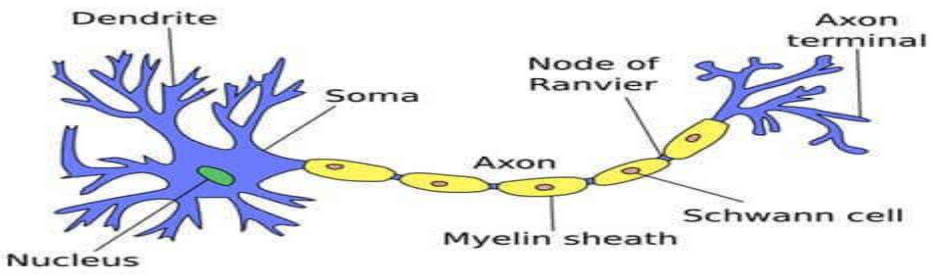
- Introduction
- Neural Networks
- Types of NN’s
  - ANN overview
  - CNN overview
  - RNN overview
- RNN model study
  - What is RNN?
  - How RNN works
  - Process
  - Need for RNNs
  - Advantages, Disadvantages & Applications
- Exercise

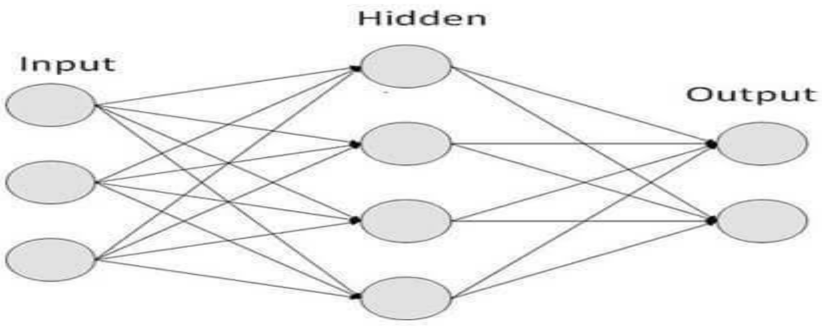
Introduction

- Deep Learning is a subset of machine learning that uses artificial neural networks with many layers (hence "deep") to analyze and learn from large amounts of data. It mimics the way the human brain processes information by recognizing patterns, making predictions, and uncovering complex relationships.

Neural networks

- They consist of interconnected layers of nodes (neurons), where each connection has a weight, and each neuron processes input using an activation function. Neural networks are a foundational component of deep learning and are used in various AI applications to learn and make



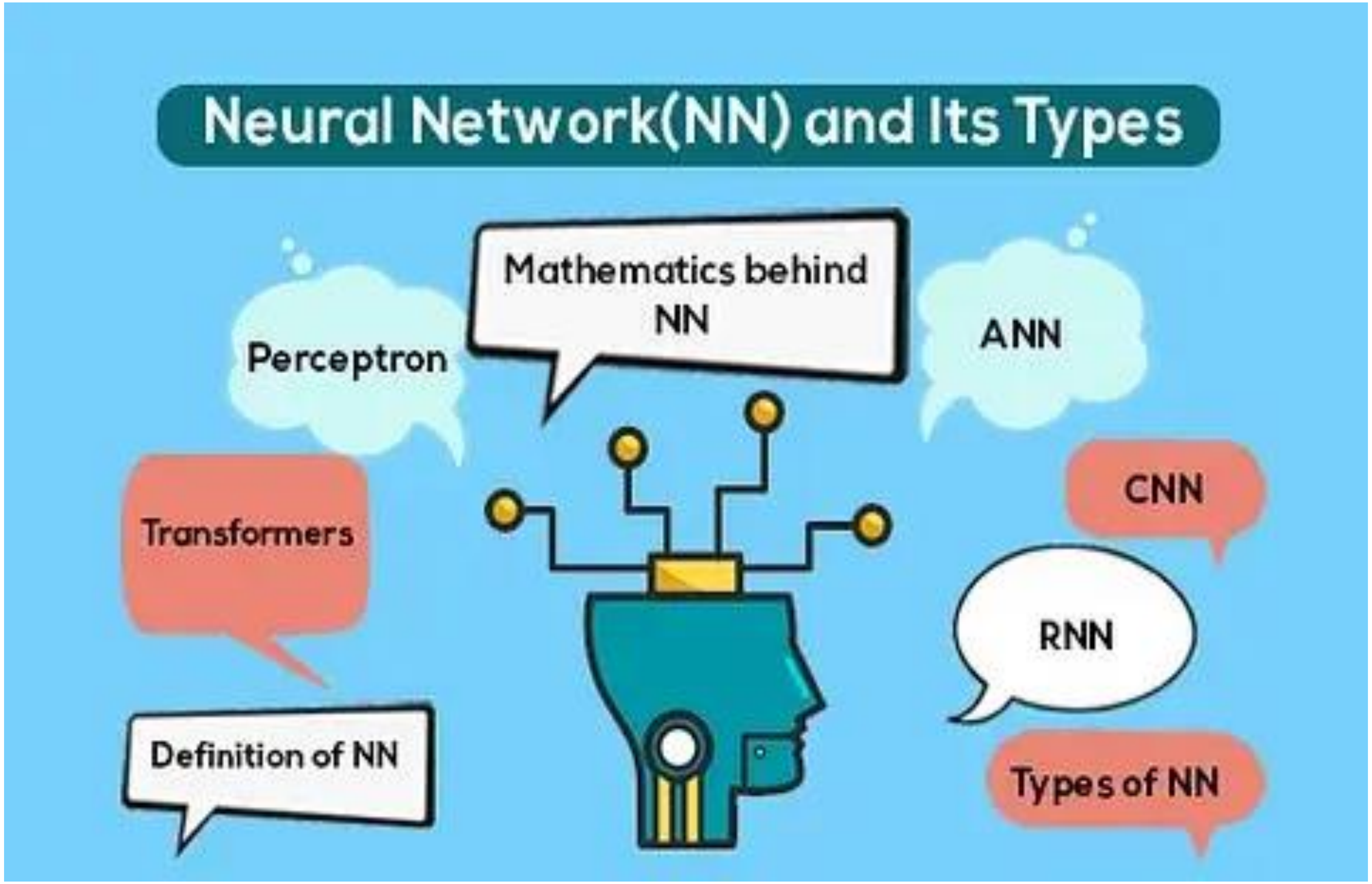


Types of neural networks

The three important types of neural networks are:

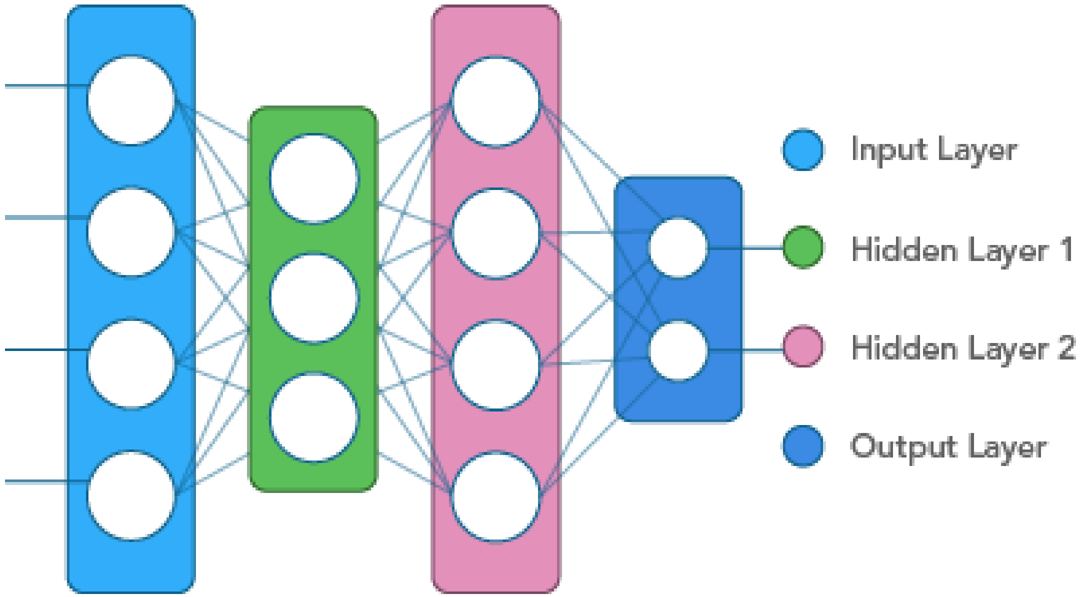
- Artificial Neural Networks (ANN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)


CNN
RNN
ANN


Neural Network(NN) and Its Types
Mathematics behind NN
Perceptron
ANN
Transformers
CNN
RNN
Definition of NN
Types of NN

Artificial Neural Network (ANN)

- An Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks work in the human brain.
- ANNs consist of layers of interconnected neurons (nodes), where each connection has an associated weight and bias that are adjusted during training.



Input Layer

- In first layer, i.e, the input layer, the neurons that send data to the buried layer are located.
- Processes data in a variety of formats specified by the programmer

Hidden Layer

- In-between both the layers, the hidden is situated.
- Performs all the math to uncover hidden patterns and characteristics

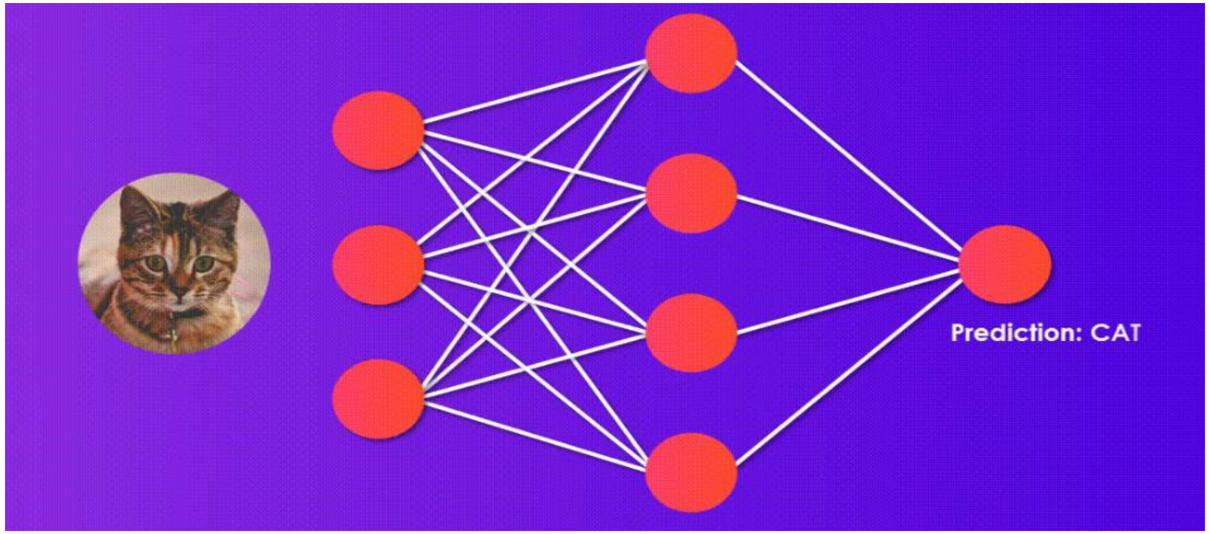
Output Layer

- Through the hidden layer, input is transformed into output, which is then transmitted using this layer.

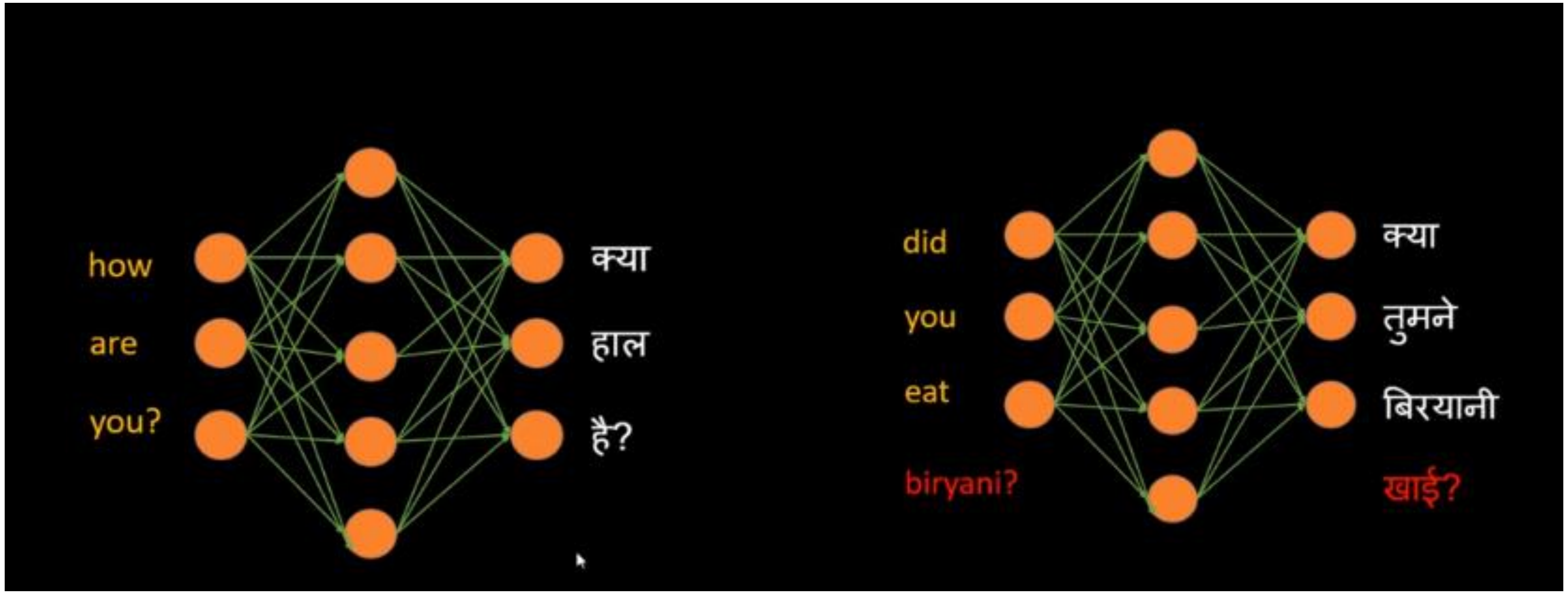
Challenges with Artificial Neural Network (ANN)

- Solving an image classification problem using ANN has the following drawbacks:
  - The number of trainable parameters increases drastically with an increase in the size of the image.
  - No fixed size of neurons
  - Too much of computation
  - No parameter sharing
  - ANN cannot capture sequential information in the input data which is required for dealing with sequence data

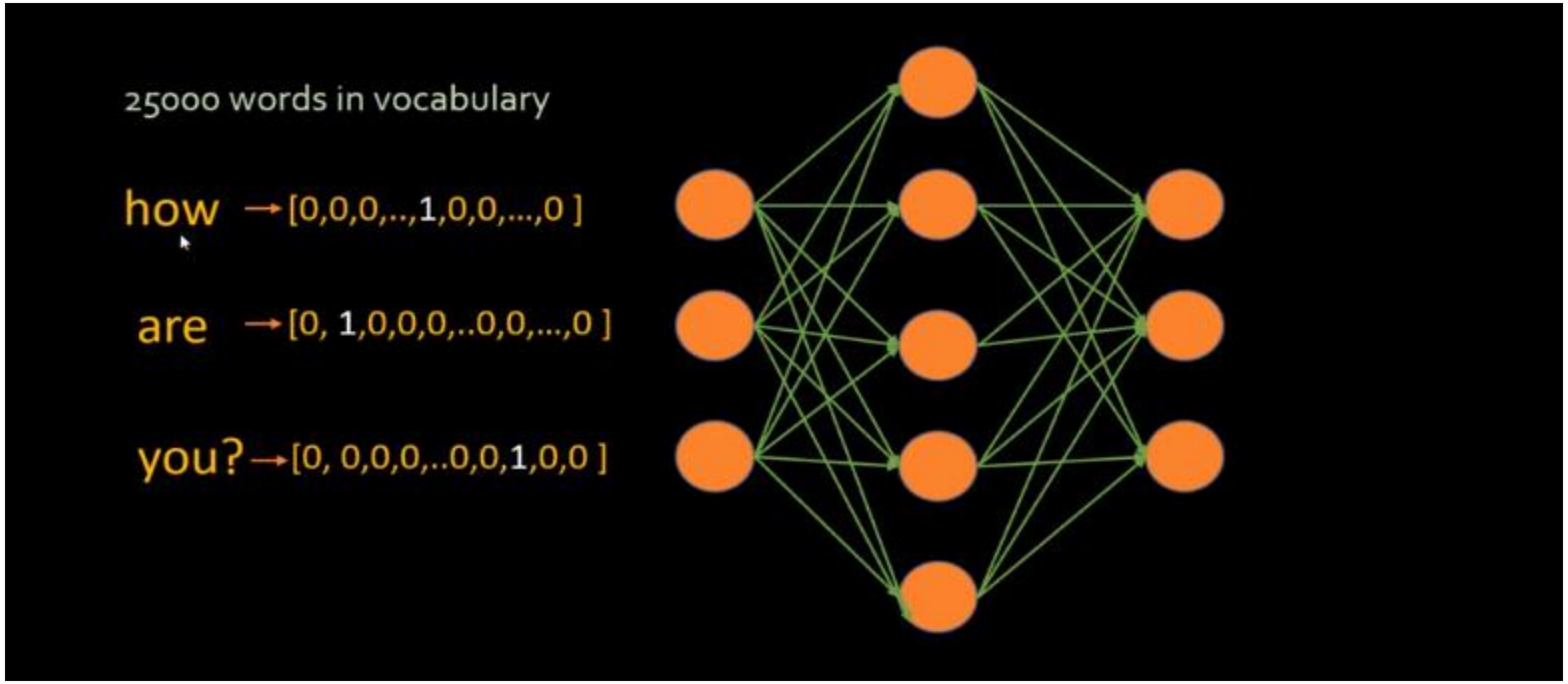
- In the given scenario, if the size of the image is 224*224, then the number of trainable parameters at the first hidden layer with just 4 neurons is 602,112. That’s huge!



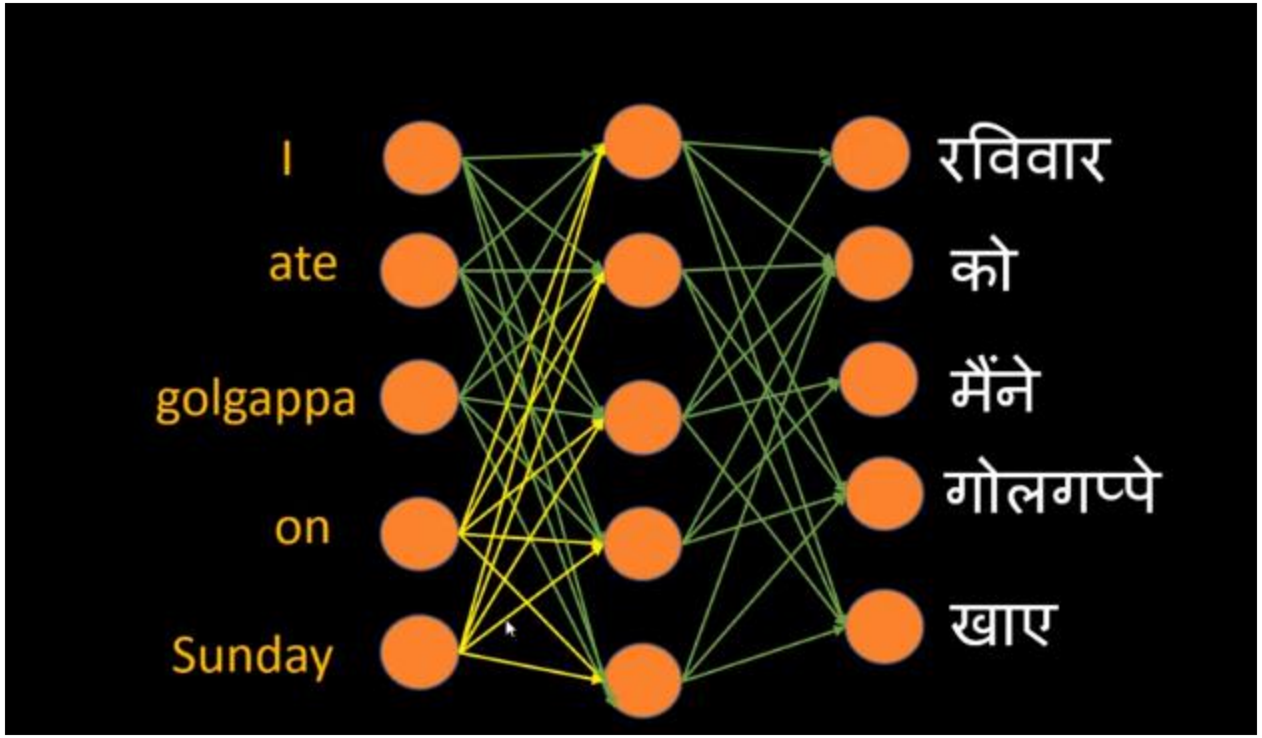
- No fixed size of neurons



- Too much of computation

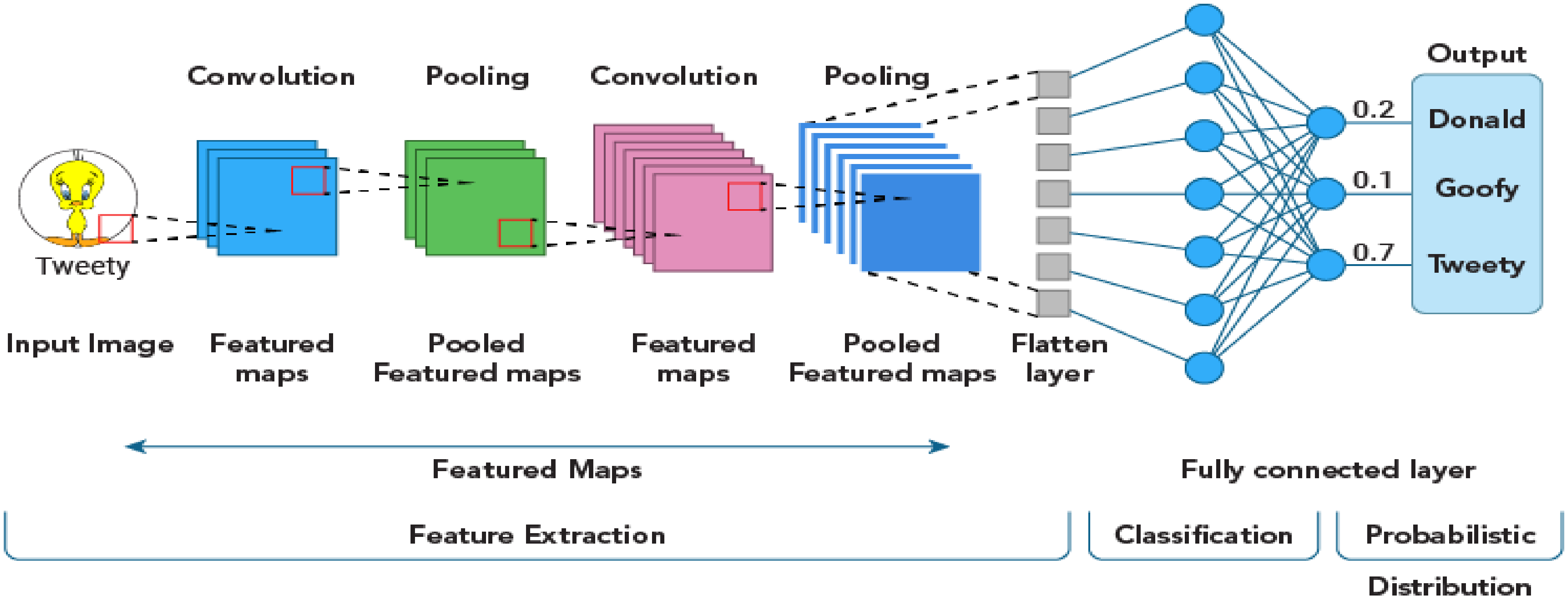


- No parameter sharing



Convolution Neural Networks (CNN)

- Convolution neural networks are mainly credited for their role in image and video recognition, recommendation systems, and image analysis and classification.



How do CNNs work?

CNNs are based on three main layers.

- Convolution layer
- Pooling layer
- Fully-connected layer
- With each layer, the CNN’s complexity in understanding the image increases.
- This means the first layers focus on interpreting simple features in an image such as its edges and colors. As the image processes through layers, the network is able to recognize complex features such as object shapes.
- Finally, the deepest layer is able to identify the target object.

Advantages of CNN

- **It automatically detects features without human supervision**. These filters help in extracting the right and relevant features from the input data

Challenges of using CNNs

As for its drawbacks, there are two main ones:

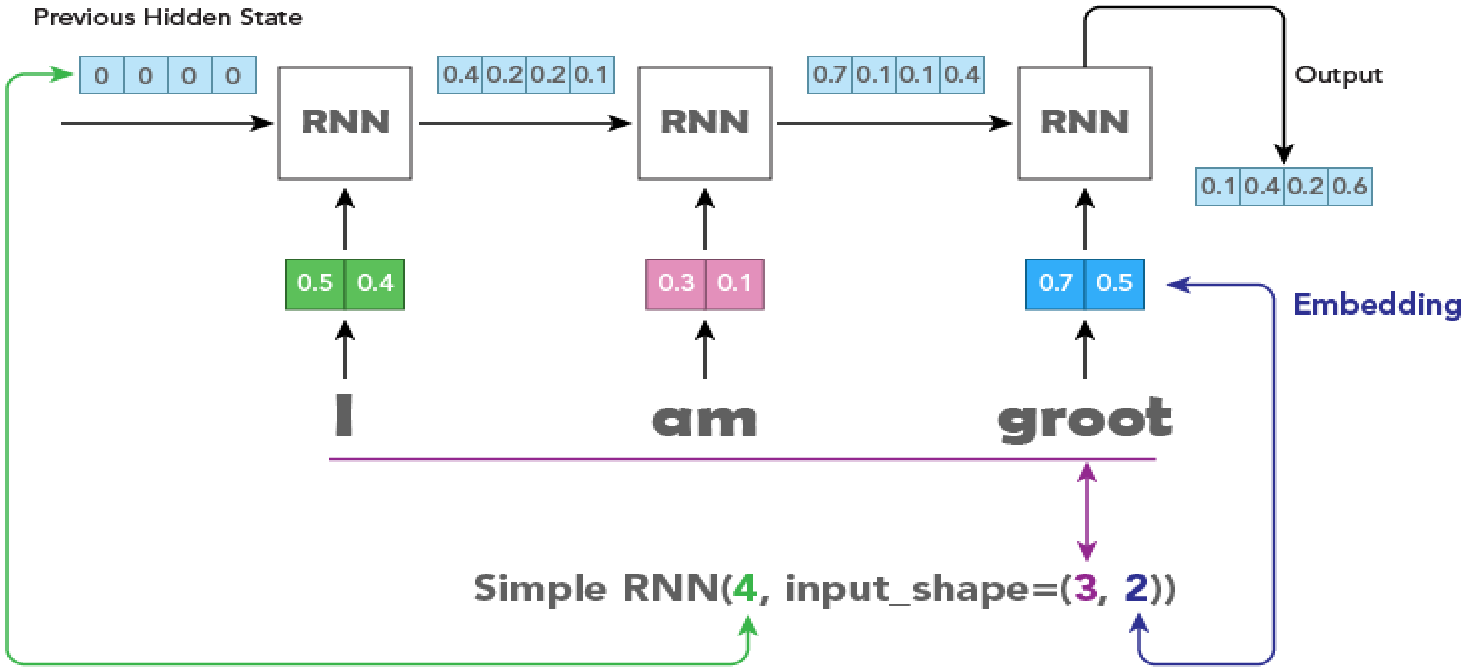
- **Difficulty in dealing with variance in the data presented**. CNN has a hard time processing objects in images that are hidden to an extent. With image classification too, the network has difficulty classifying titled or rotated images. Put simply, CNN can’t encode an object’s orientation and position and can’t process spatially invariant data.
- **Computationally demanding**. Training CNN requires numerous graphical processing units (GPUs). To add to that, if you lack good GPUs, the training becomes slow.

CNN vs ANN vs RNN

- ANNs (Artificial Neural Networks) are helpful for solving complex problems
- CNNs (Convolution Neural Networks) are best for solving computer vision-related problems dealing with images.
- RNNs (Recurrent Neural Networks) are proficient in natural language processing dealing with Sequential data.

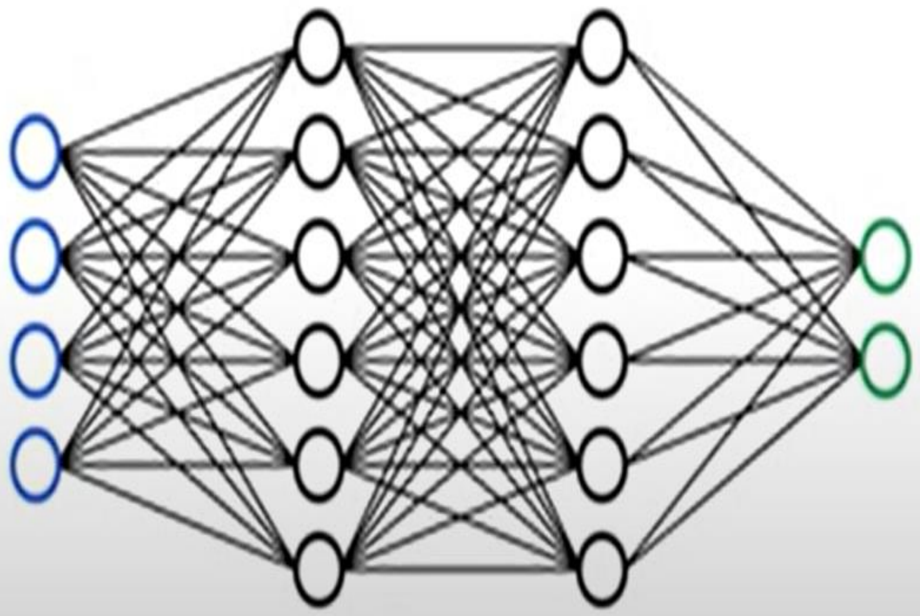
Recurrent neural networks (RNN)

- Recurrent Neural Networks come into picture when there's a need for predictions using sequential data. Sequential data can be a sequence of images, words, etc.

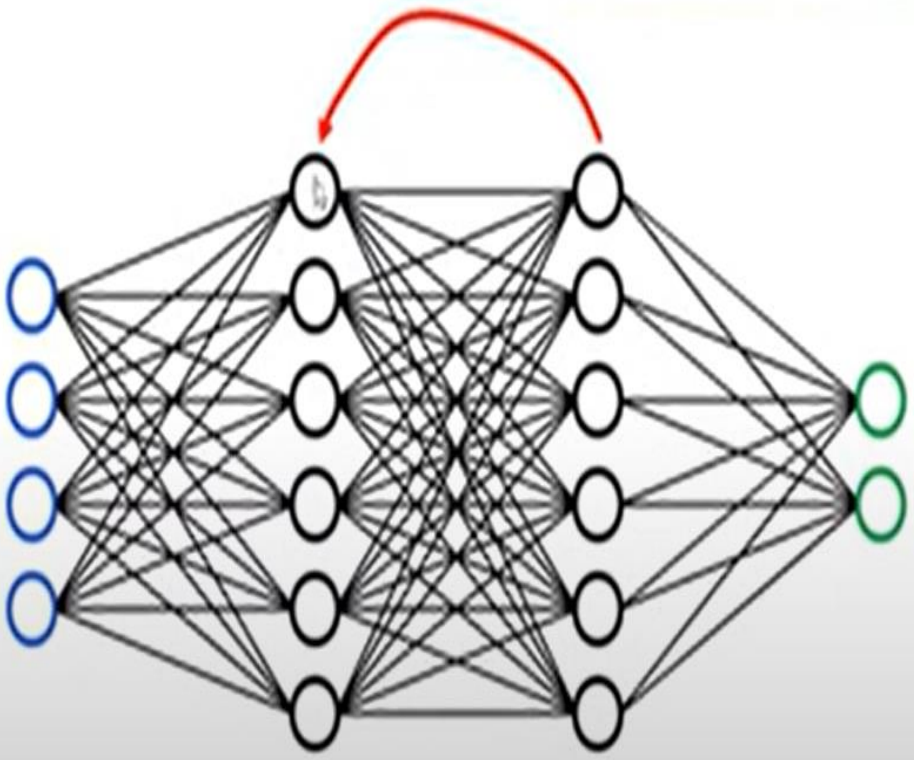


- Recurrent neural networks (RNN) are a class of neural networks that are powerful for modeling sequence data such as time series or natural language.
- RNNs are designed for sequential data, where each neuron connects to the next layer and to neurons within the same layer.

Like ANN and CNN, RNN also learns with training data. From there on, it doesn’t process data on inputted data alone. Instead, it uses data from past inputs to make decisions too. In a nutshell, this architecture is built for having a ‘memory’.



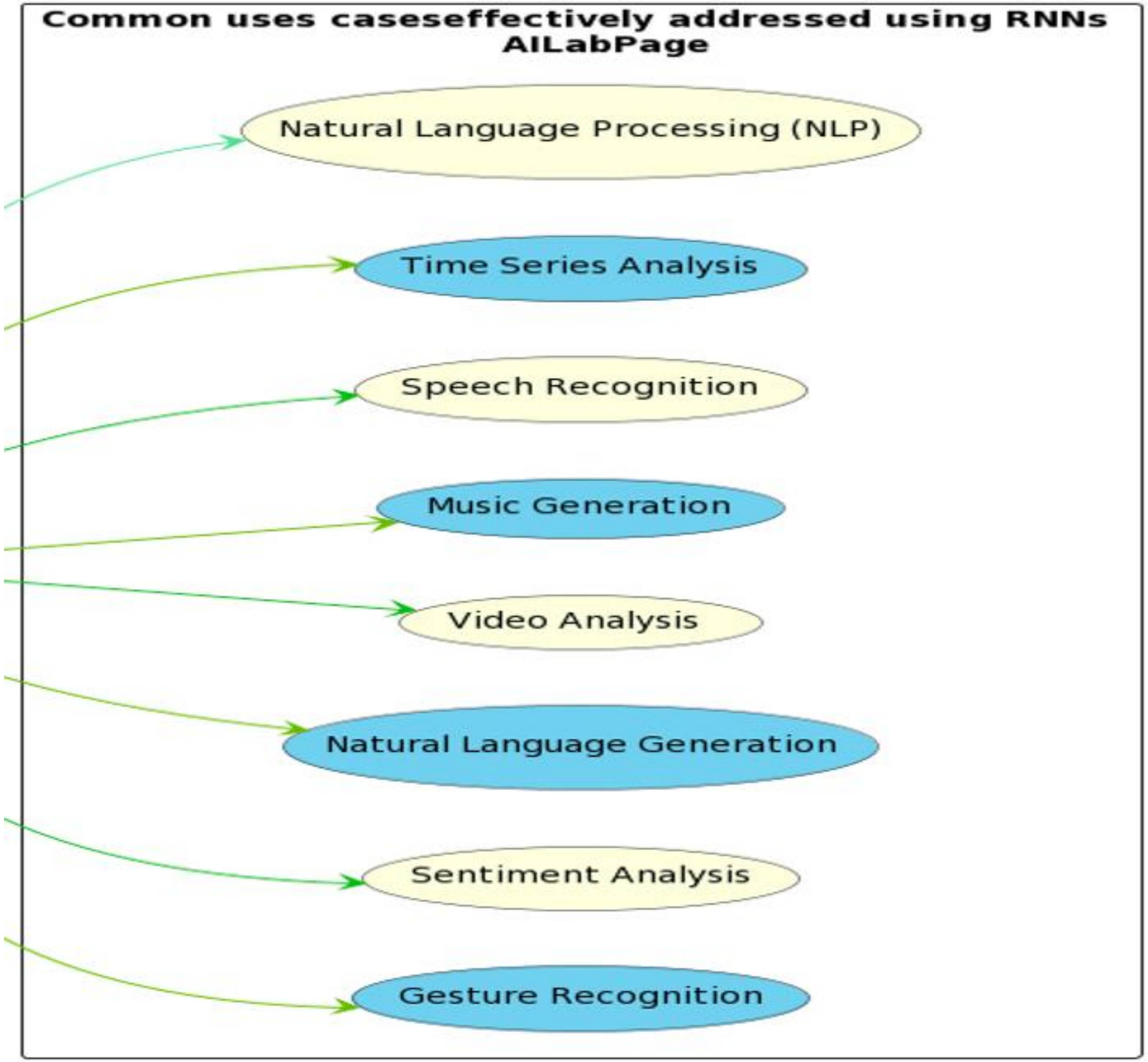
Feed forward network



RNN remembers the past

What should we use RNN for?

- RNNs have found success in a wide range of applications, including:
- **Natural Language Processing (NLP)**: Sentiment analysis, machine translation.
- **Speech Processing**: Speech-to-text, text-to-speech systems.
- **Time-Series Forecasting**: Weather prediction, sales forecasting.
- **Music Generation**: Creating melodies based on input sequences.
- **Video Analysis**: Understanding temporal sequences in video frames.


Common uses caseseffectively addressed using RNNs
AILabPage
Natural Language Processing (NLP)
Time Series Analysis
Speech Recognition
Music Generation
Video Analysis
Natural Language Generation
Sentiment Analysis
Gesture Recognition

	x		y
auto complete	not interested at	→	this time
translation	how are you?	→	क्या हाल है?
NER	Rudolph Smith bought 1000 shares of tesla Inc. in March 2020	→	Rudolph Smith (Person) bought 1000 shares of tesla Inc. (Company) in March 2020 (time)
Sentiment Analysis	Not only the fan was expensive, but it was broken when it arrived.	→	★☆☆☆☆

What should we use RNN for?

What's for dinner?

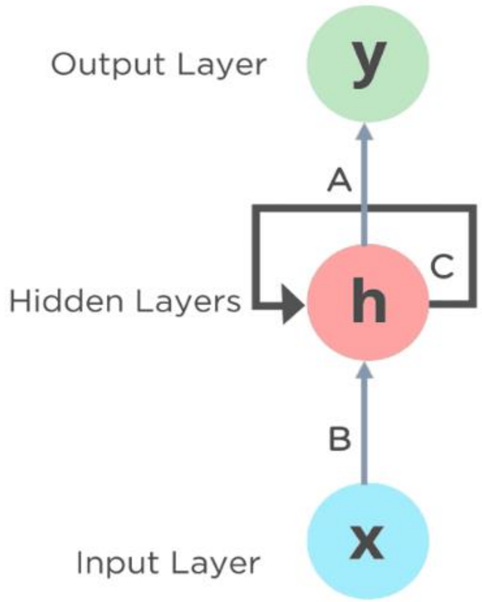
Sunday	Japanese
Monday	Chinese
Tuesday	Thai
Wednesday	Indian
Thursday	Japanese
Friday	Chinese
Saturday	Thai
Sunday	Indian

Kmit

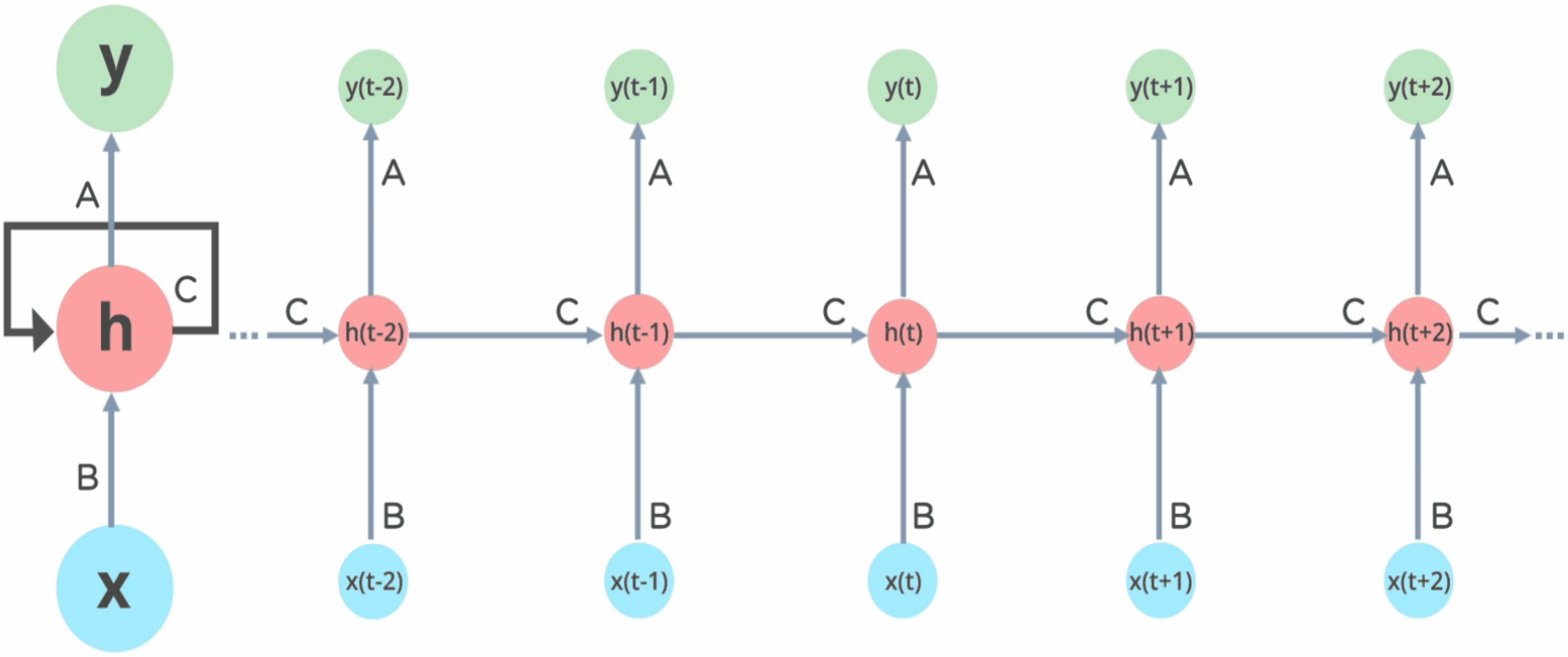
What’s for dinner?

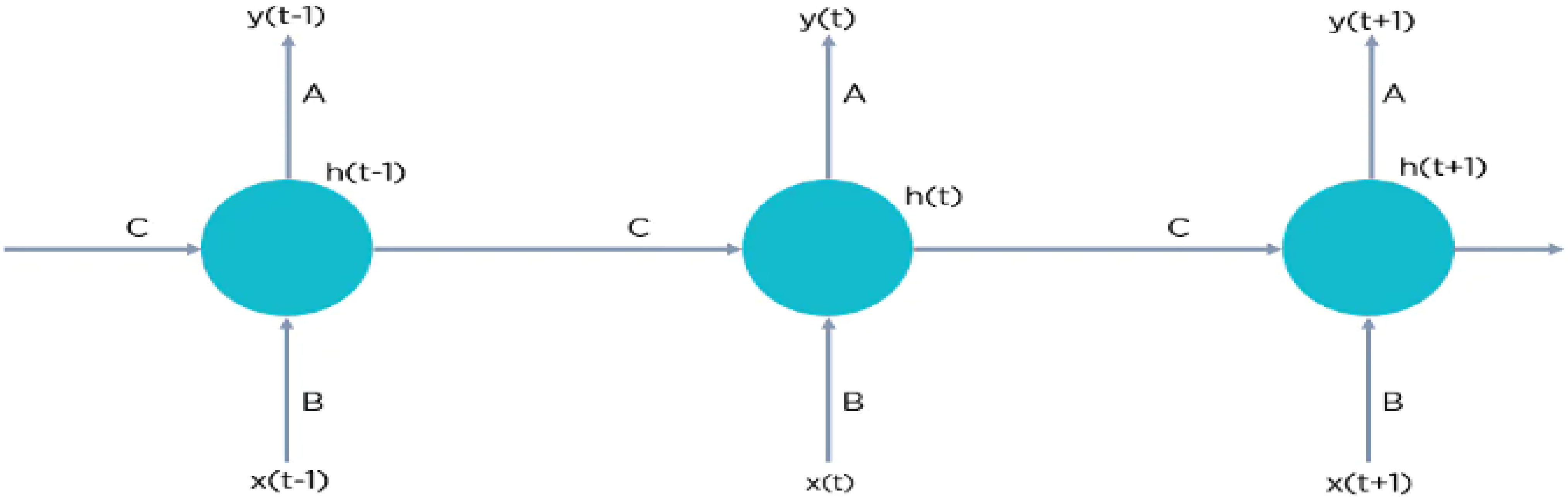
Sunday	Japanese
Monday	Chinese
Tuesday	Thai
Wednesday	Indian
Thursday	Japanese
Friday	Chinese
Saturday	Thai
Sunday	Indian
Monday	Japanese
Tuesday	Chinese
Wednesday	Thai
Thursday	Indian
Friday	Japanese
Saturday	Chinese
Sunday	Thai
Monday	Indian
Tuesday	Japanese
Wednesday	Chinese
Thursday	Thai
Friday	Indian
Saturday	Japanese
Sunday	Chinese
Monday	Thai
Tuesday	Indian
Wednesday	Japanese

- RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.



- RNNs process sequential data, where the output of one step is fed as input to the next.

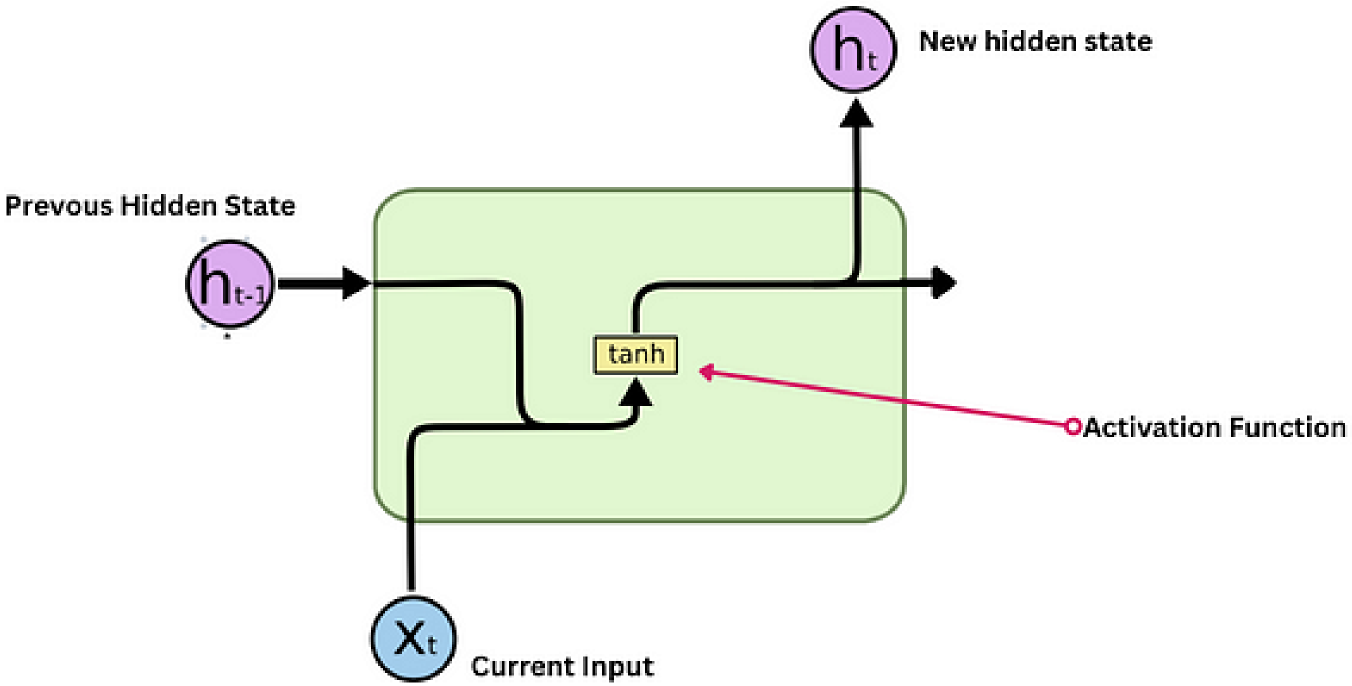




$h(t) = f_c\,(h(t-1), x(t))$

$h(t)$ = new state
f_c = function with parameter c
$h(t-1)$ = old state
$x(t)$ = input vector at time step t

RNN
(Mathematical Intuition)



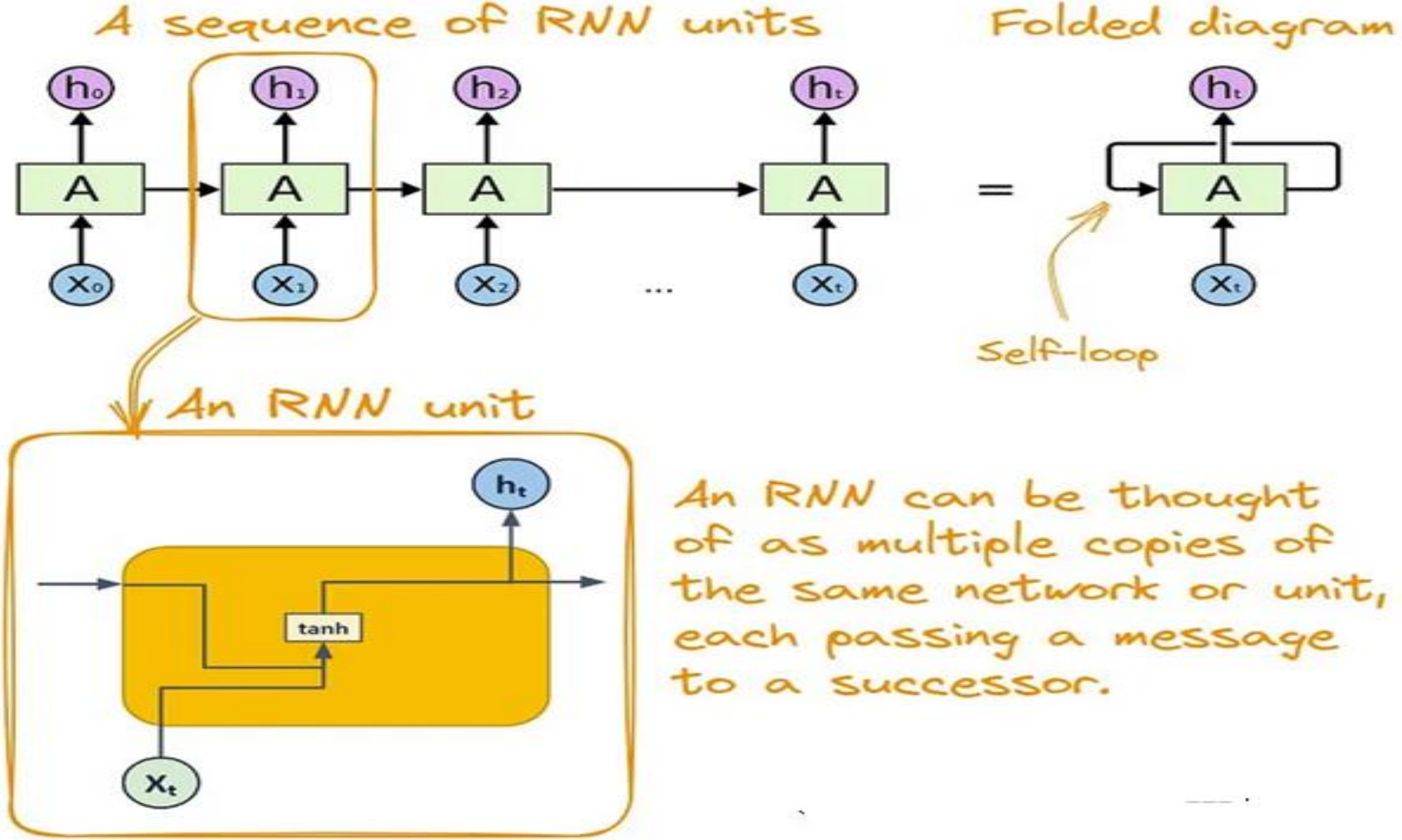
$$h_t = f(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

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

New hidden state

Activation Function

Prevous Hidden State



An RNN can be thought of as multiple copies of the same network or unit, each passing a message to a successor.

RNN Architecture

The basic architecture of an RNN consists of the following components:

1. **Input Sequence:** The input to an RNN is a sequence of data, which can be a sequence of words, characters, or any other elements, depending on the task at hand. In the context of Character-Level RNNs, the input sequence comprises individual characters.

2. **Recurrent Layer:** The recurrent layer processes the input sequence one element at a time. At each time step, it takes the current input and the previous hidden state as input and produces a new hidden state, updating the model’s internal memory. This recurrent nature enables the model to capture sequential dependencies.

3. Output Layer: Depending on the task, the RNN may have an output layer that produces predictions or classifications. For instance, in text classification, the output layer might classify the text into categories, while in text generation, it generates the next element in the sequence.

4. Hidden State: The hidden state, also known as the internal memory of the RNN, plays a crucial role in capturing the context and dependencies in sequential data. It maintains information from previous time steps, allowing the model to consider the entire input sequence when making predictions.

RNN Process

- Data input to the input layer
- Representation of the data in the Input layer is computed and sent to the Hidden Layer
- Hidden Layer conducts Sequence Modeling and Training in Forward or Backward directions
- Multiple Hidden Layers using Forward and Backward direction Sequence Modeling and Training can be used
- Final Hidden Layer sends the processed result to the Output Layer

The steps in RNN training include:

Input at Each Time Step: A single time step of the input sequence is provided to the network.

Calculate Hidden State: Using the current input and the previous hidden state, the network calculates the current hidden state ht.

State Transition: The current hidden state ht then becomes $ht-1$ for the next time step.

Sequential Processing: This process continues across all time steps to accumulate information from previous states.

Output Generation and Error Calculation: The final hidden state is used to compute the network's output, which is then compared to the actual target output to generate an error.

Backpropagation Through Time (BPTT): This error is backpropagated through each time step to update weights and train the RNN.

Advantages Of RNNs

Advantage	Description	Example
Sequence Processing	Handles ordered data efficiently	Language modeling
Temporal Dependency Capture	Retains past information for context	Predicting stock trends
Variable-Length Input/Output	Adapts to varying input/output sequence lengths	Summarizing text
Compact Representation	Encodes data efficiently in hidden states	Time-series data compression
Versatility	Supports multiple input-output mapping types	Machine translation
Context Understanding	Understands sequences holistically	Sentiment analysis
Shared Weights	Reduces parameters and generalizes better	Pattern recognition in time-series

RNN Applications

1) Image Captioning: RNNs are used to caption an image by analyzing the activities present.

"A Dog catching a ball in mid air"

2) Natural Language Processing: Sentiment analysis can be carried out using an RNN for Natural Language Processing (NLP).

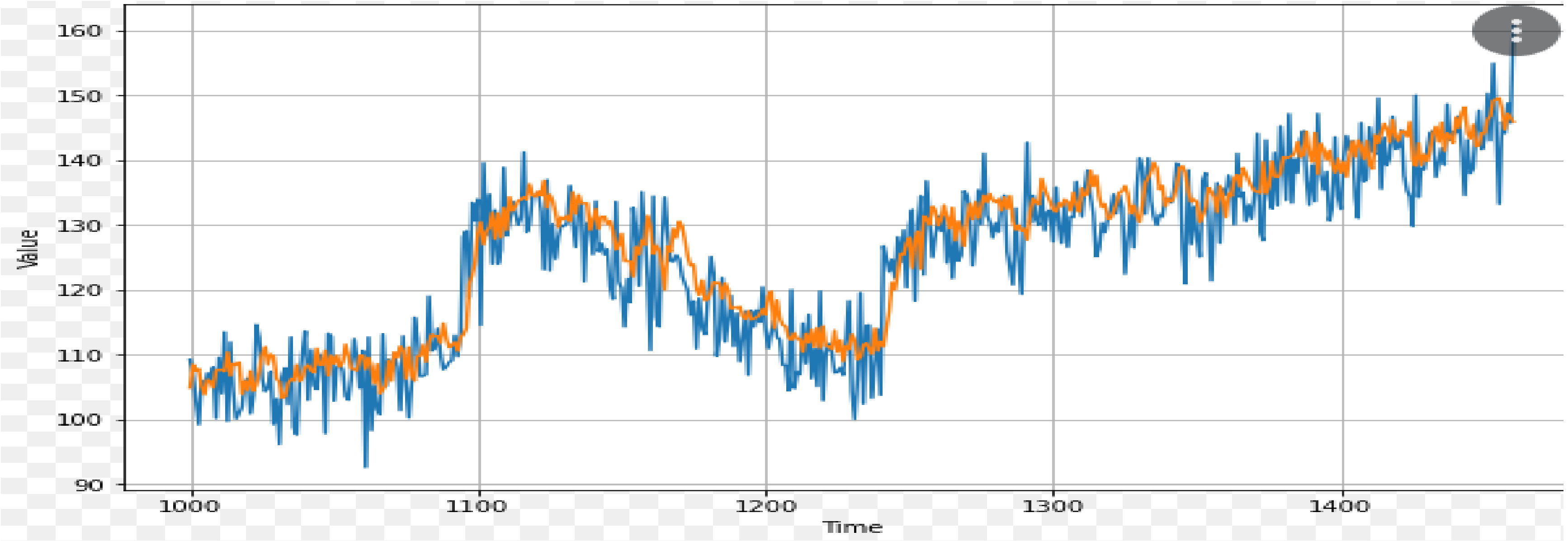
When it rains, look for rainbows.
When it's dark, look for stars.

Positive Sentiment

3) Machine Translation: Given an input in one language, RNNs can be used to translate the input into different languages as output.

Here the person is speaking in English and it is getting translated into Chinese, Italian, French, German and Spanish languages

4) Time Series Prediction: Any time series problem, like predicting the prices of stocks in a particular month, can be solved using an RNN.



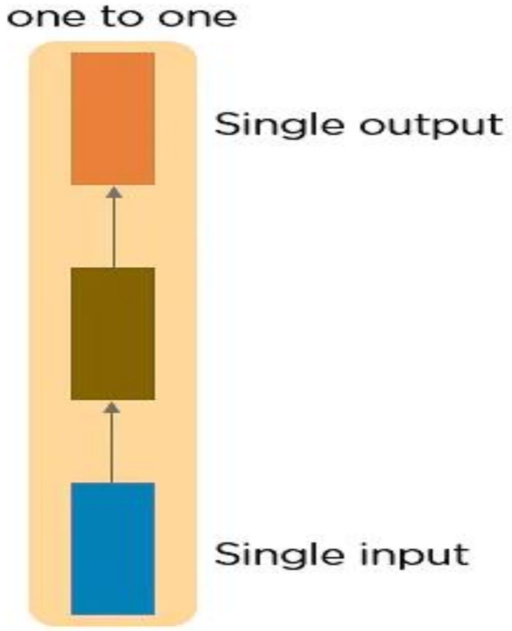
Types of Recurrent Neural Networks

There are four types of Recurrent Neural Networks:

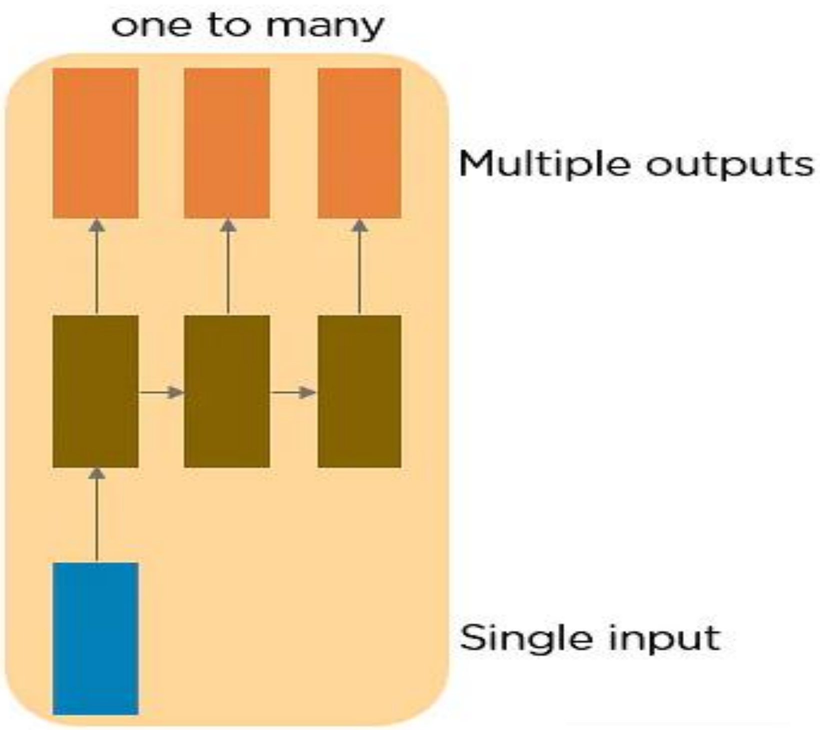
- One to One
- One to Many
- Many to One
- Many to Many

1) One to One RNN

- This type of neural network is known as the Vanilla Neural Network.

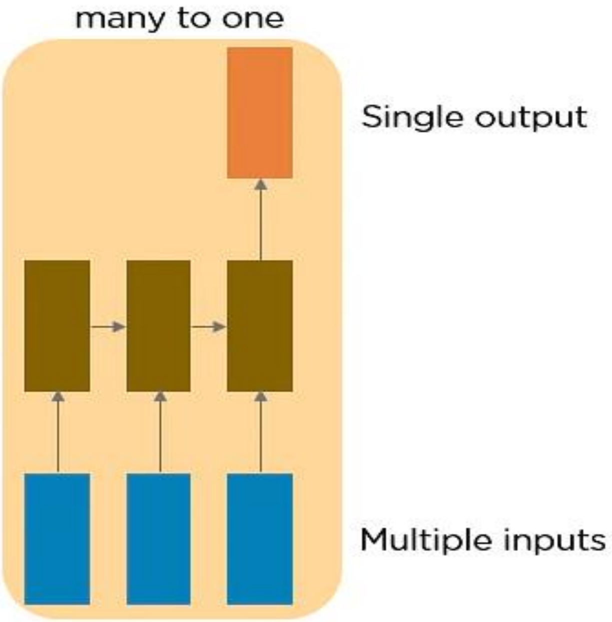


2) One to Many RNN



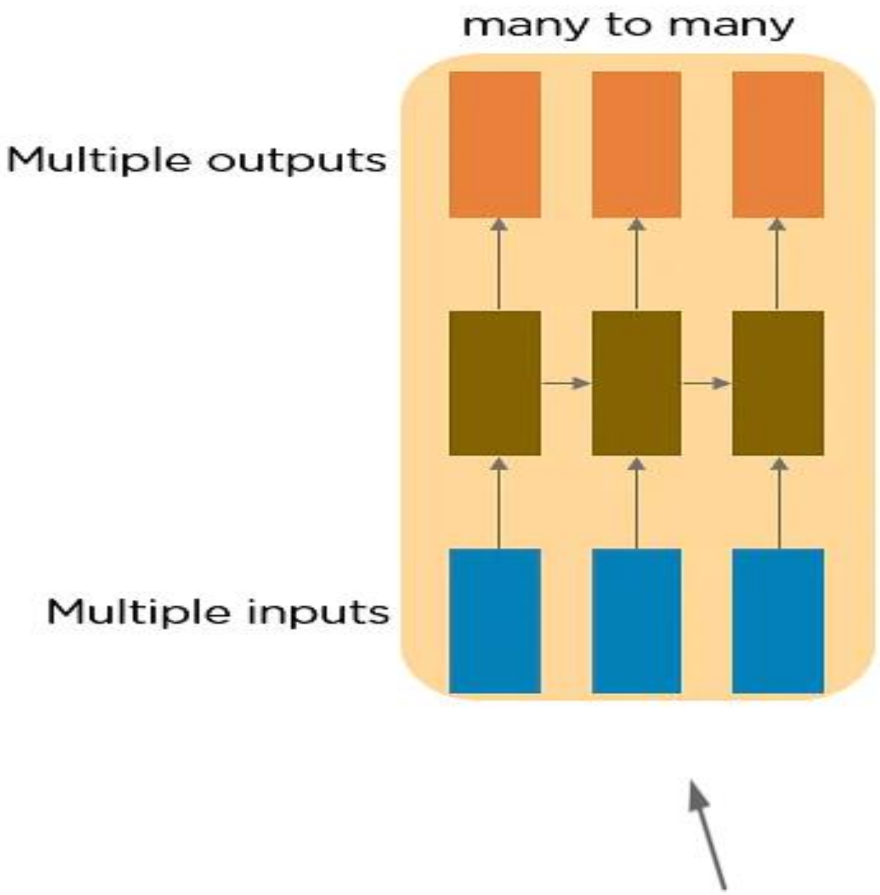
e.g. **Image Captioning**
image -> sequence of words

3) Many to One RNN



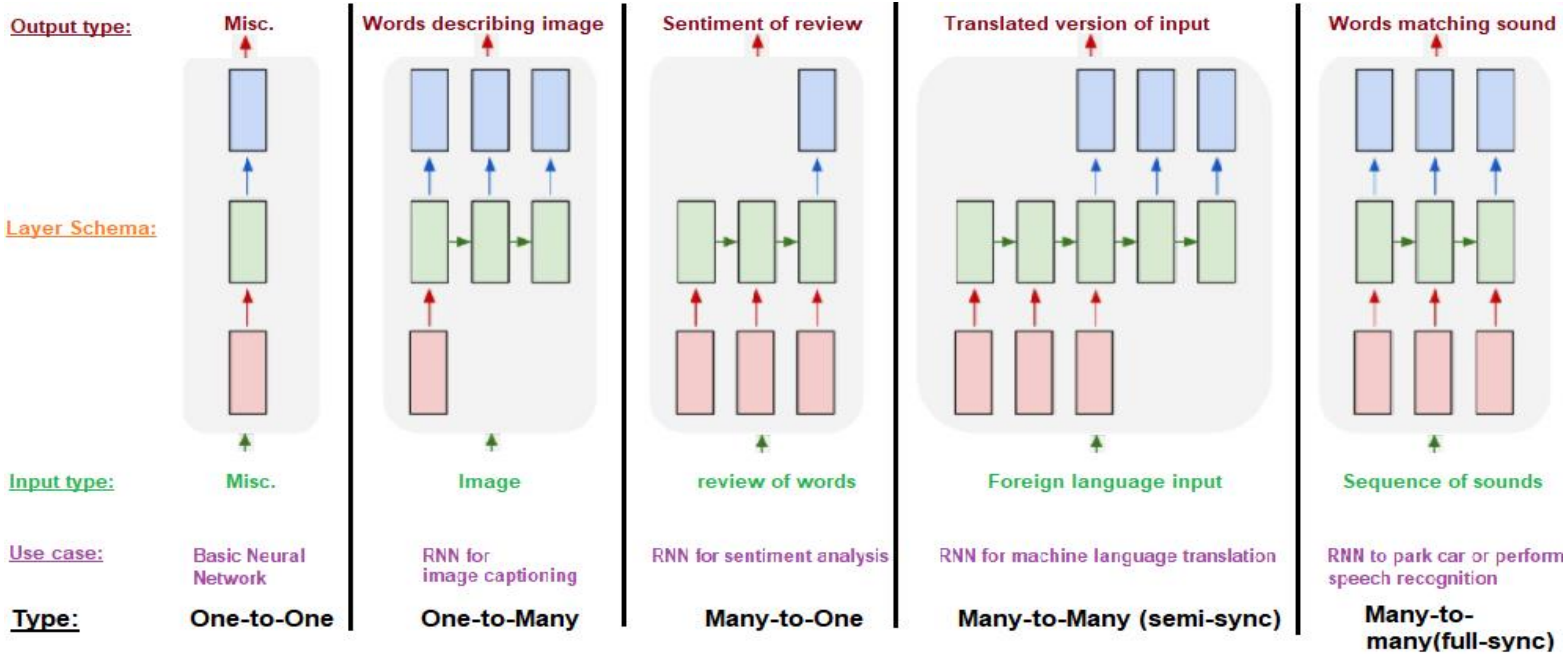
e.g. **action prediction**
sequence of video frames -> action class

4) Many to Many RNN



E.g. **Video Captioning**
Sequence of video frames -> caption

summarizing Different types and Applications of RNN



Two Issues of Standard RNNs

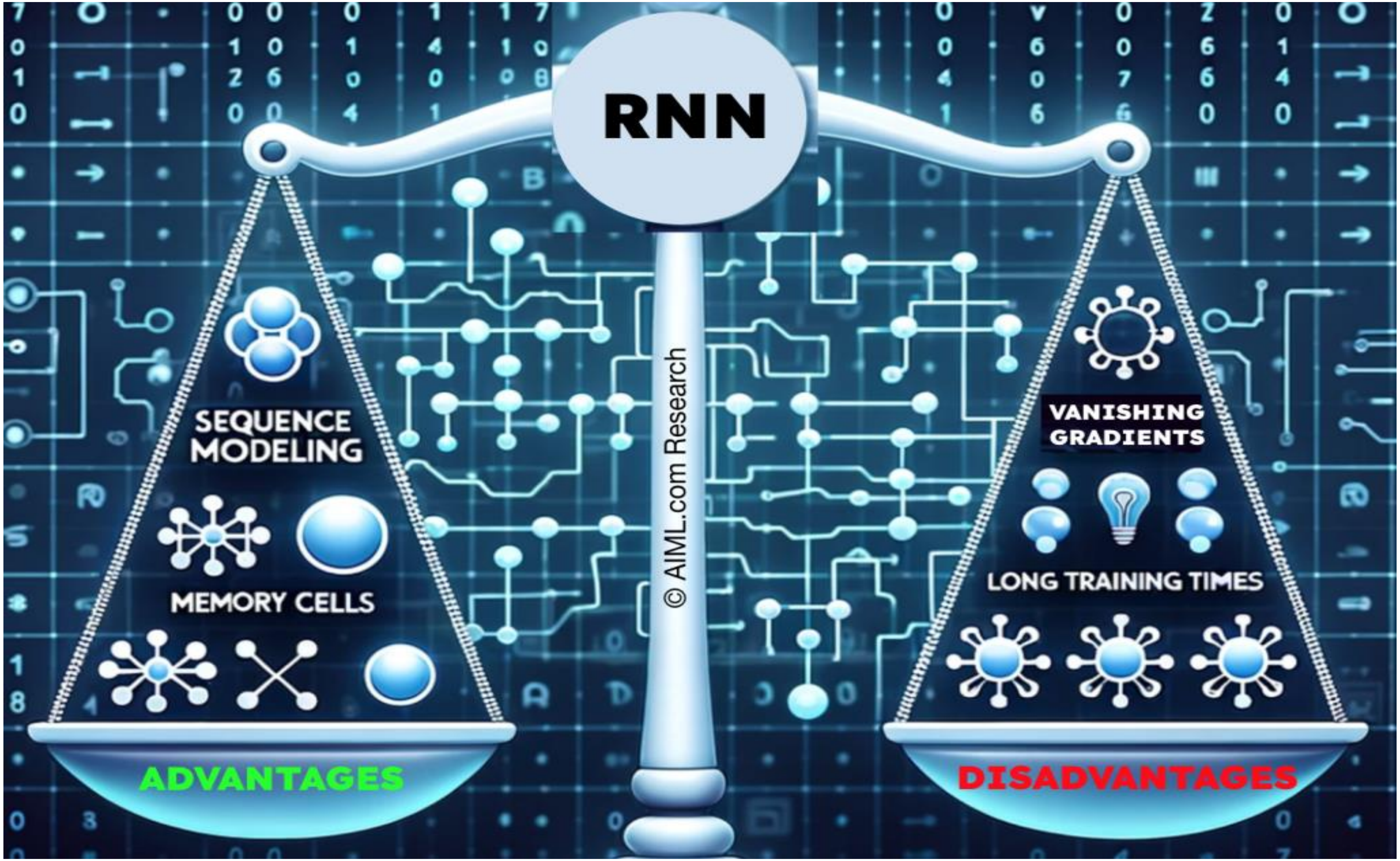
- Now there are problems with the simple implementation of RNN too. They learn through back propagation over .They cannot remember important information that may require in a later time stamp.
- 1. Vanishing Gradient Problem
- 2. Exploding Gradient Problem

As number of hidden layers grow, gradient becomes very small and weights will hardly change . This will hamper the learning process.

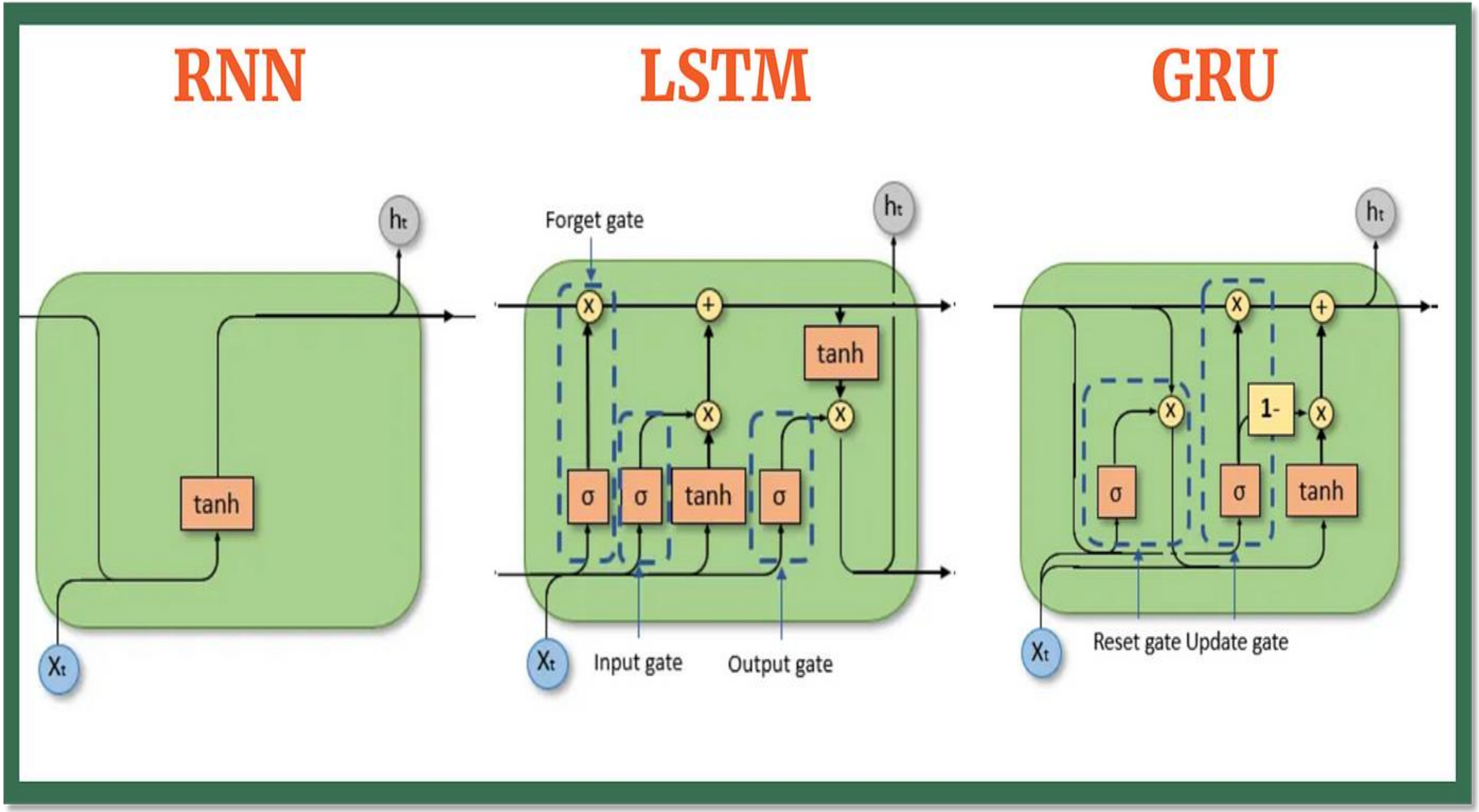
Vanishing Gradients

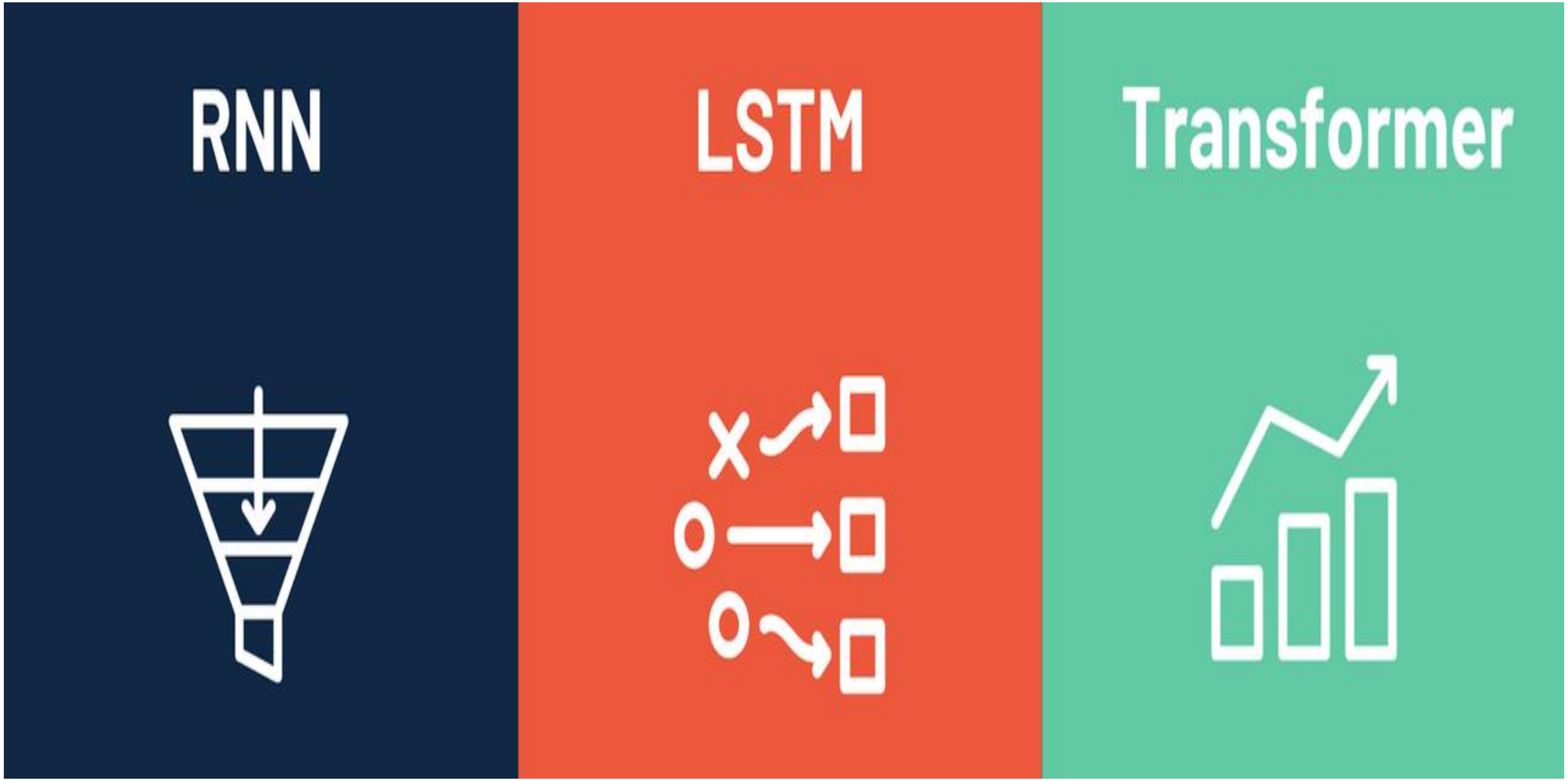
When individual derivatives are large, the final derivate will also become huge and weights would change drastically.

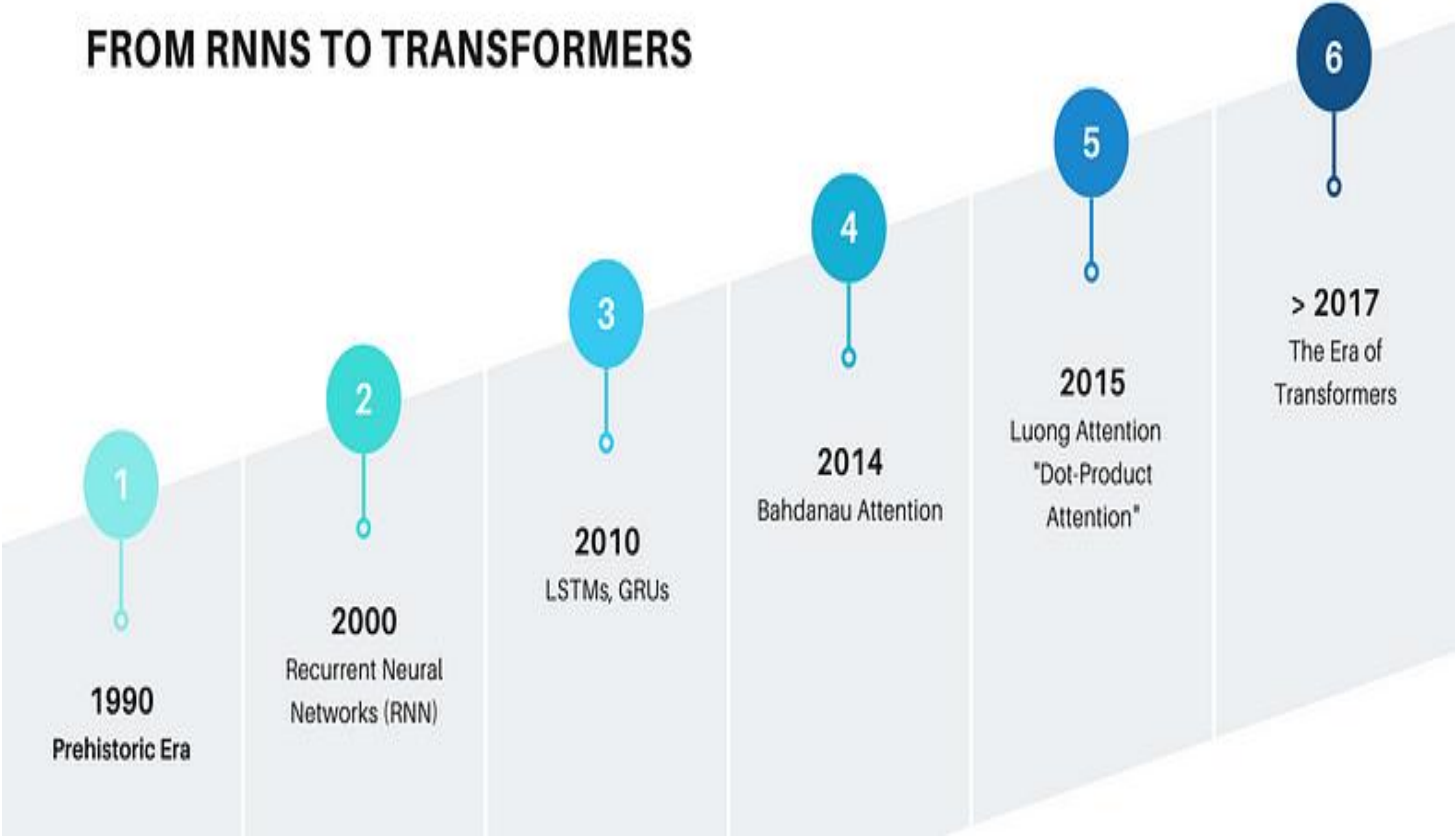
Exploding Gradients


RNN
SEQUENCE MODELING
MEMORY CELLS
ADVANTAGES
VANISHING GRADIENTS
LONG TRAINING TIMES
DISADVANTAGES
© AIML.com Research

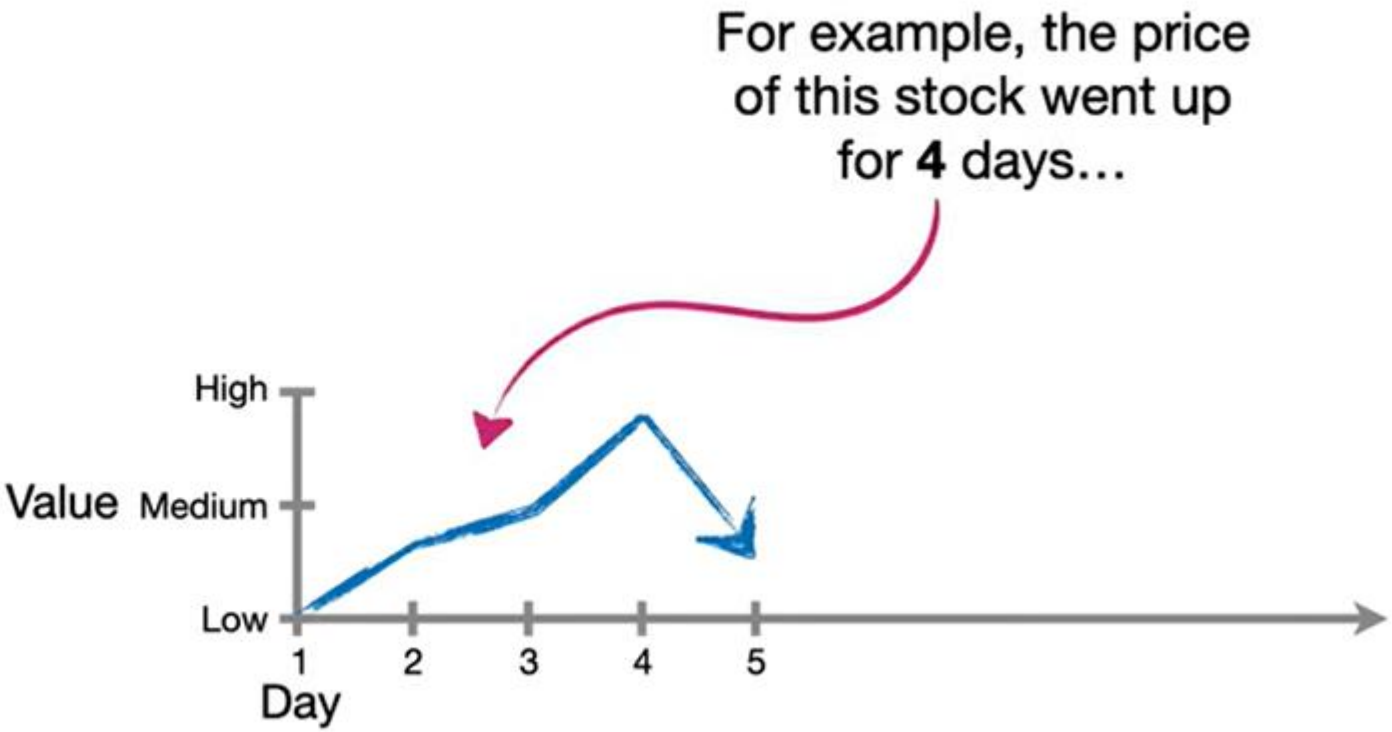
- To overcome these problems we use LSTM (long short term memory),a very special kind of recurrent network and GRU (Gated Recurrent Unit) which is a slightly modified version of LSTM.


RNN
LSTM
GRU
tanh
Forget gate
σ
σ
tanh
σ
tanh
Input gate
Output gate
1-
σ
σ
tanh
Reset gate Update gate

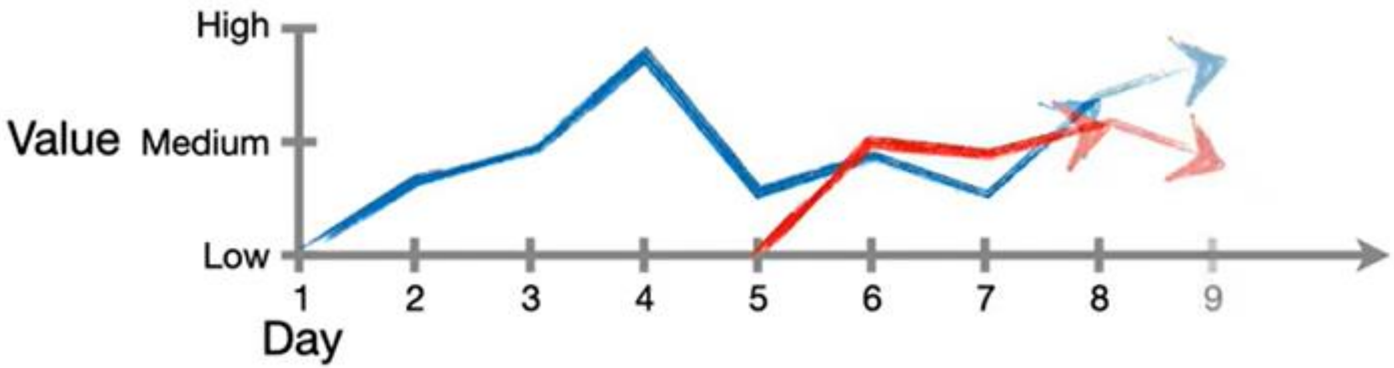

RNN
LSTM
Transformer

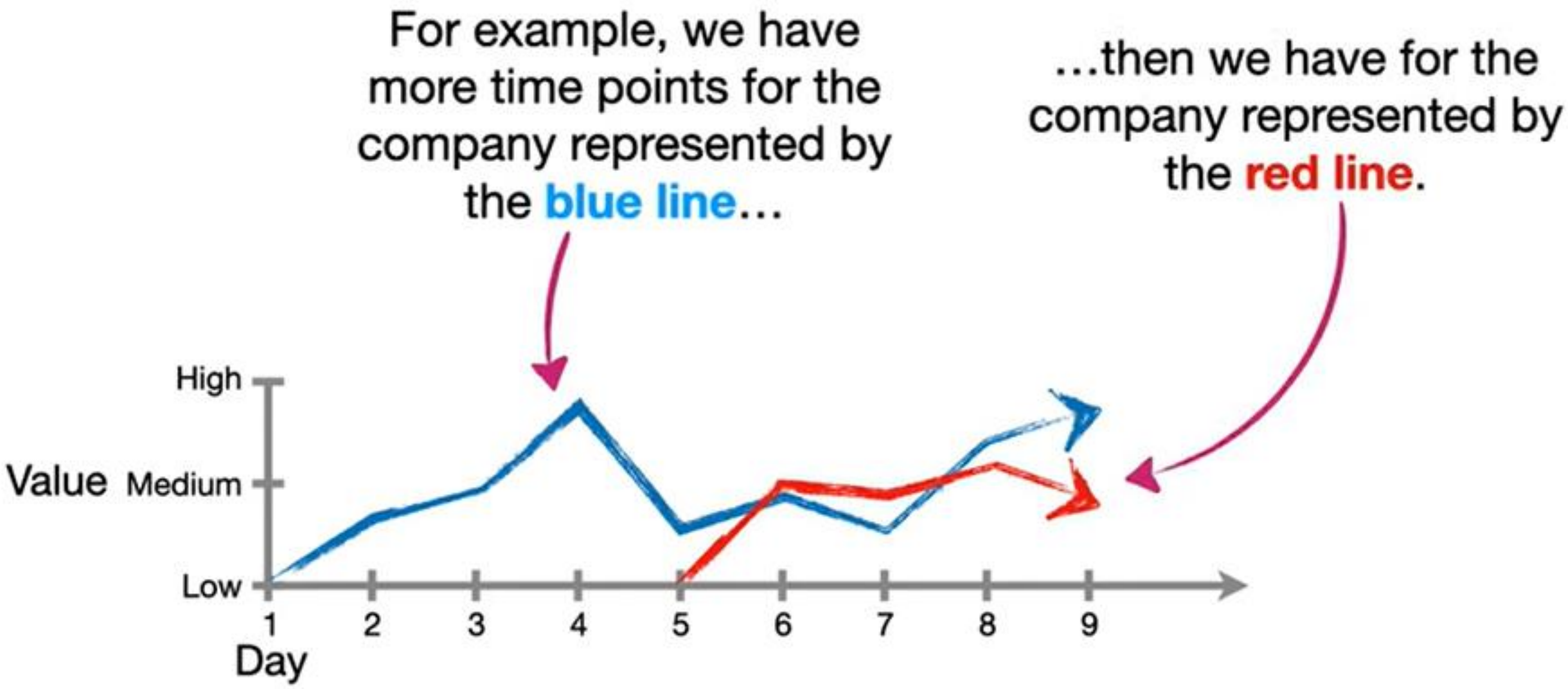

FROM RNNS TO TRANSFORMERS
1
1990
Prehistoric Era
2
2000
Recurrent Neural Networks (RNN)
3
2010
LSTMs, GRUs
4
2014
Bahdanau Attention
5
2015
Luong Attention "Dot-Product Attention"
6
> 2017
The Era of Transformers

Exercise: Google Stock Price Prediction


For example, the price
of this stock went up
for **4** days...
High
Value
Medium
Low
1
2
3
4
5
Day

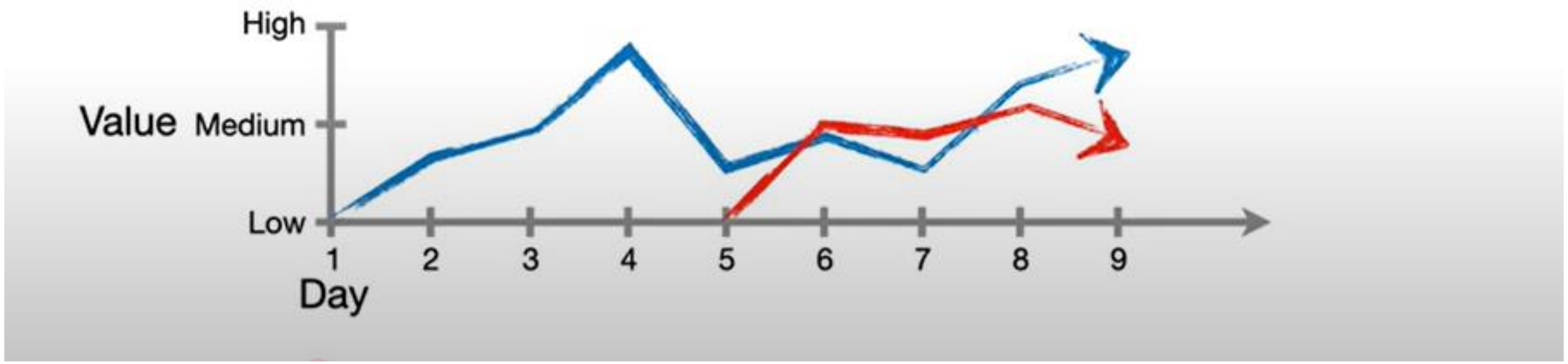
Also, the longer a company has been traded on the stock market, the more data we'll have for it.



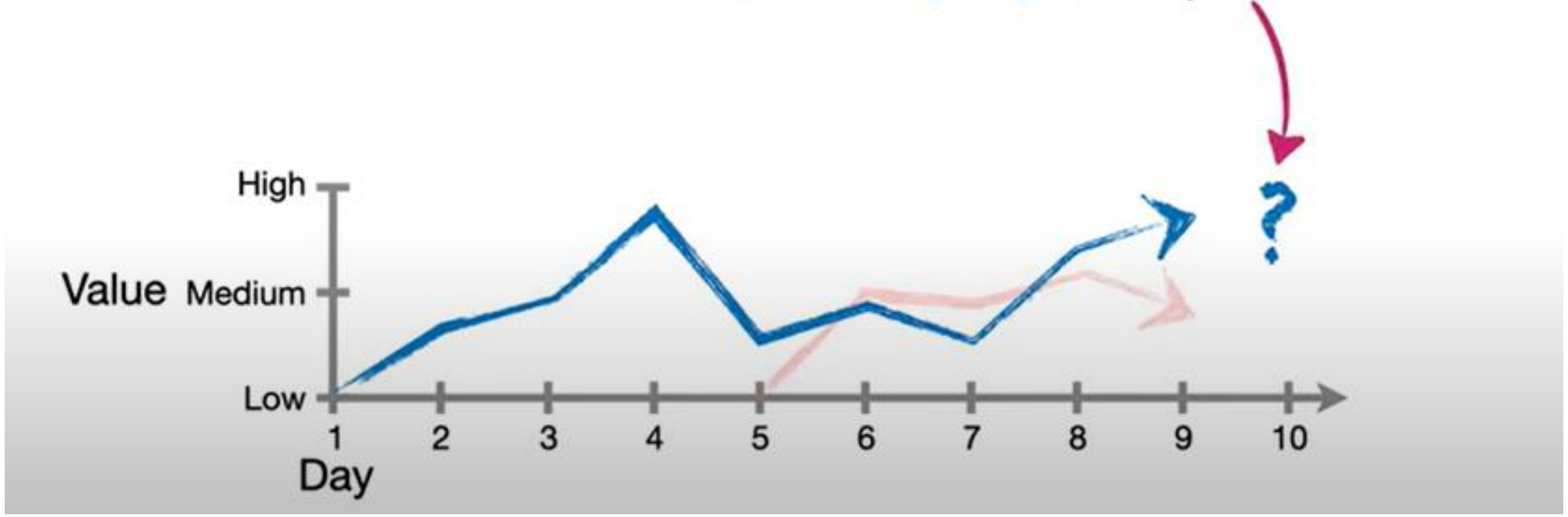


For example, we have more time points for the company represented by the blue line…
…then we have for the company represented by the red line.
High
Value
Medium
Low
1
2
3
4
5
6
7
8
9
Day

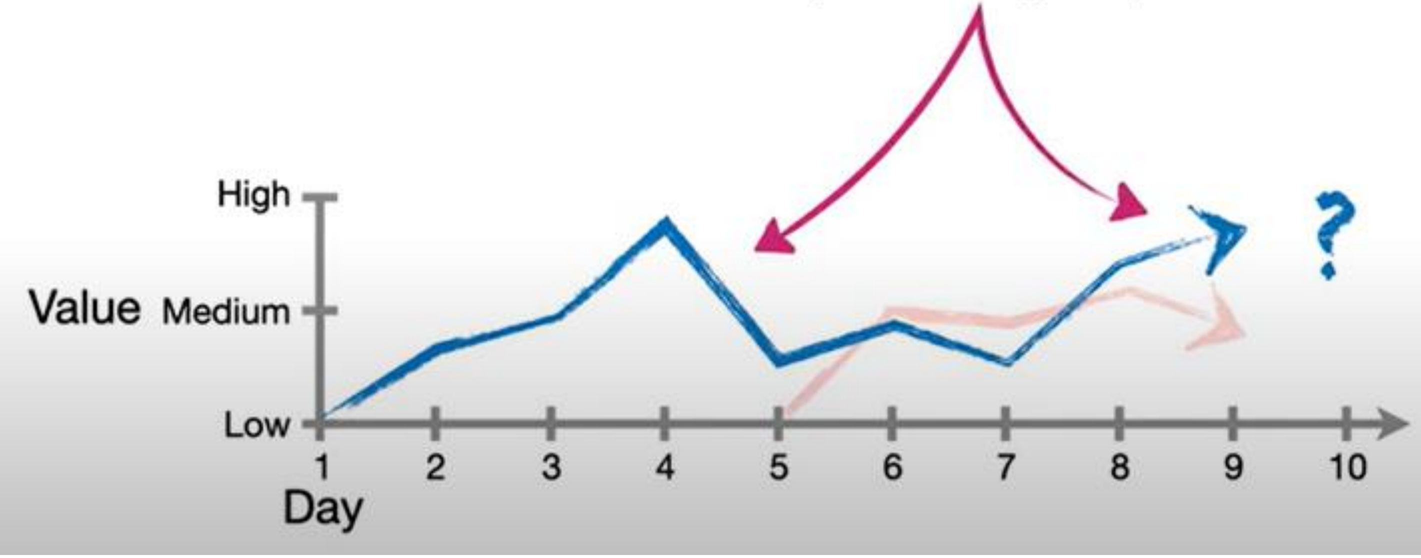
…then we need a neural network that works with different amounts of sequential data.

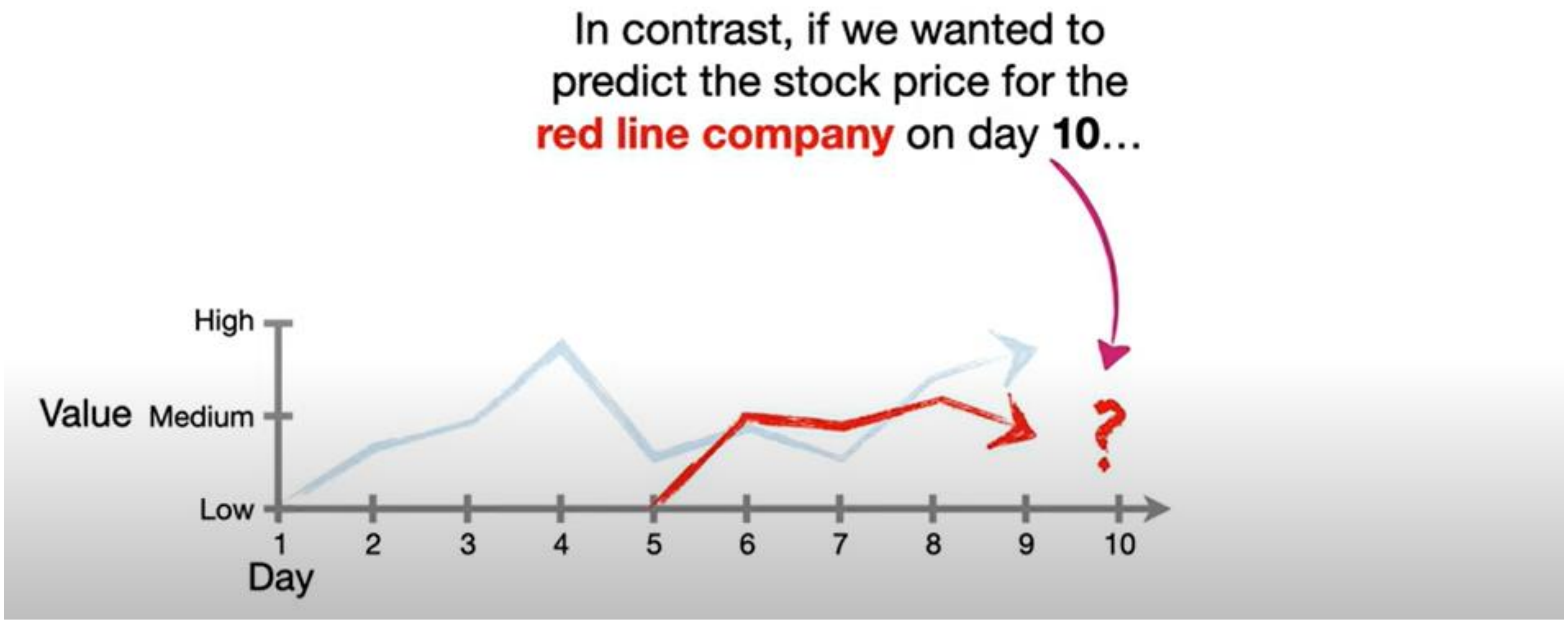


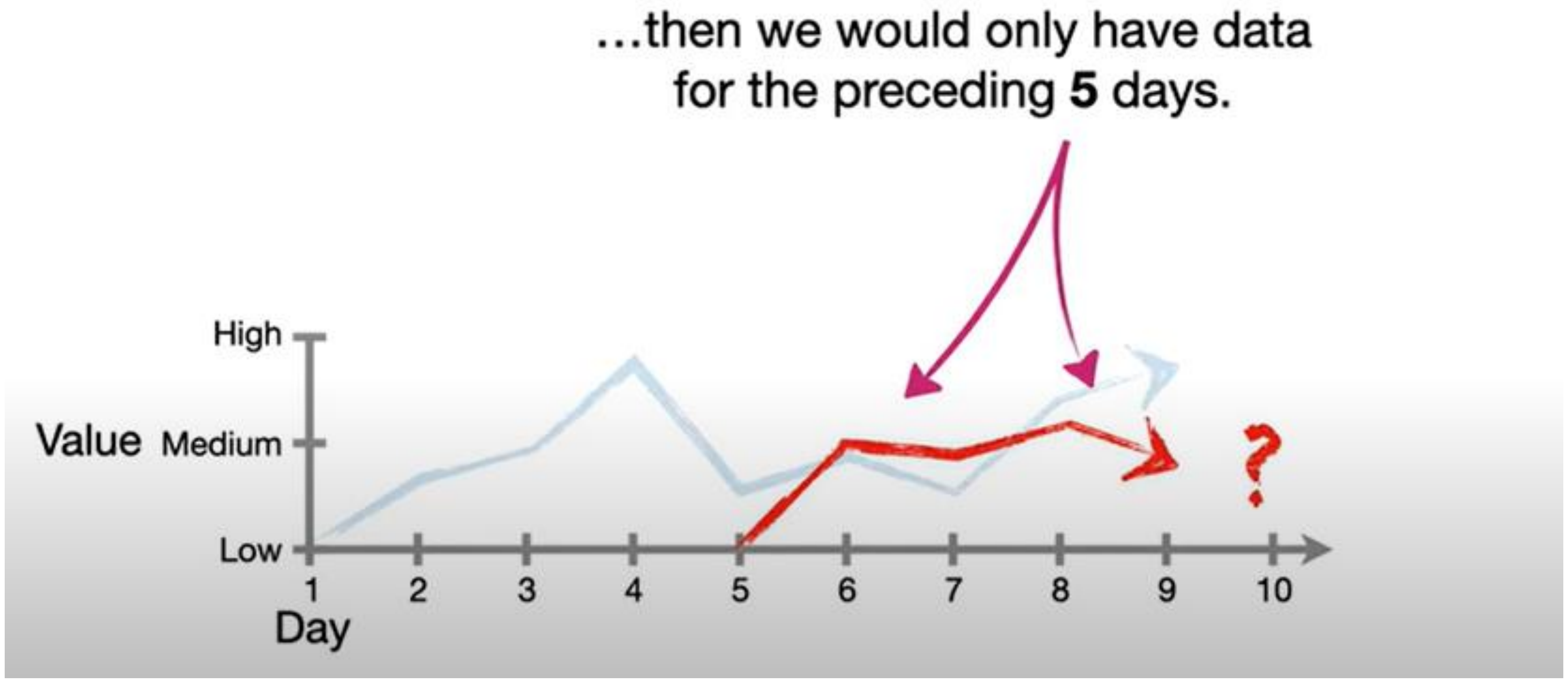
In other words, if we want to predict the stock price for the **blue line company** on day **10**…



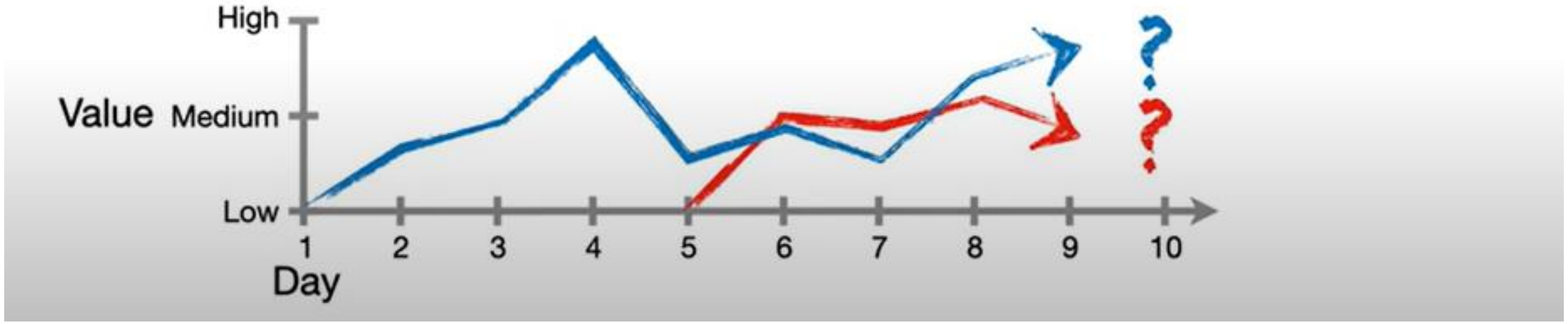
…then we might want to use the data from all **9** of the preceding days.



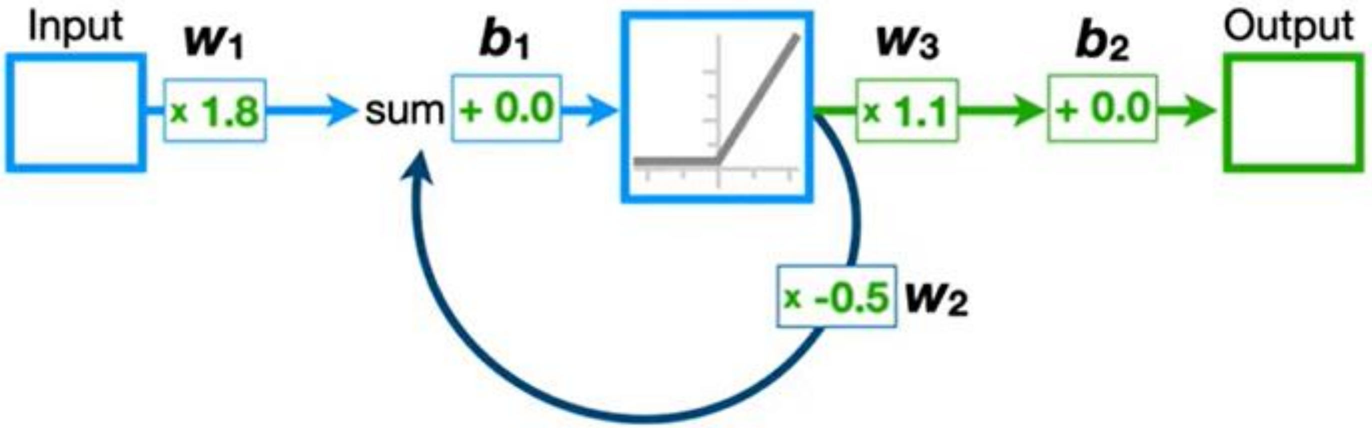

In contrast, if we wanted to predict the stock price for the red line company on day 10…
High
Value
Medium
Low
1
2
3
4
5
6
7
8
9
10
Day
?


…then we would only have data
for the preceding **5** days.
High
Value
Medium
Low
1
2
3
4
5
6
7
8
9
10
Day
?

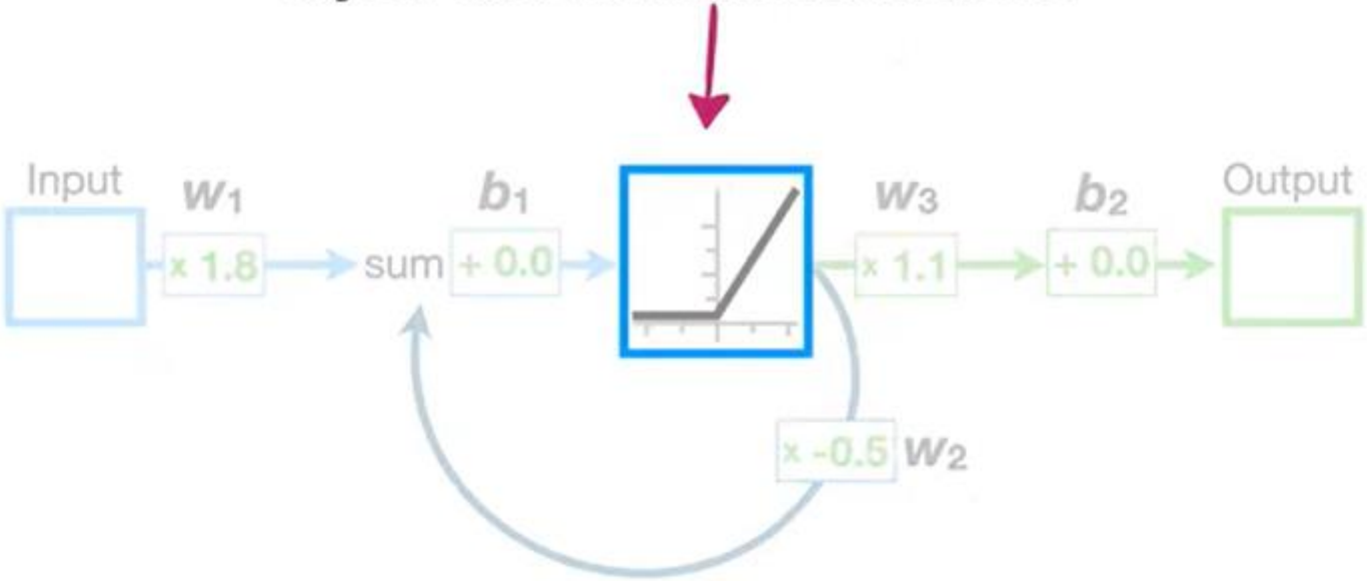
The good news is that one way to deal with the problem of having different amounts of input values is to use a **Recurrent Neural Network (RNN)**.



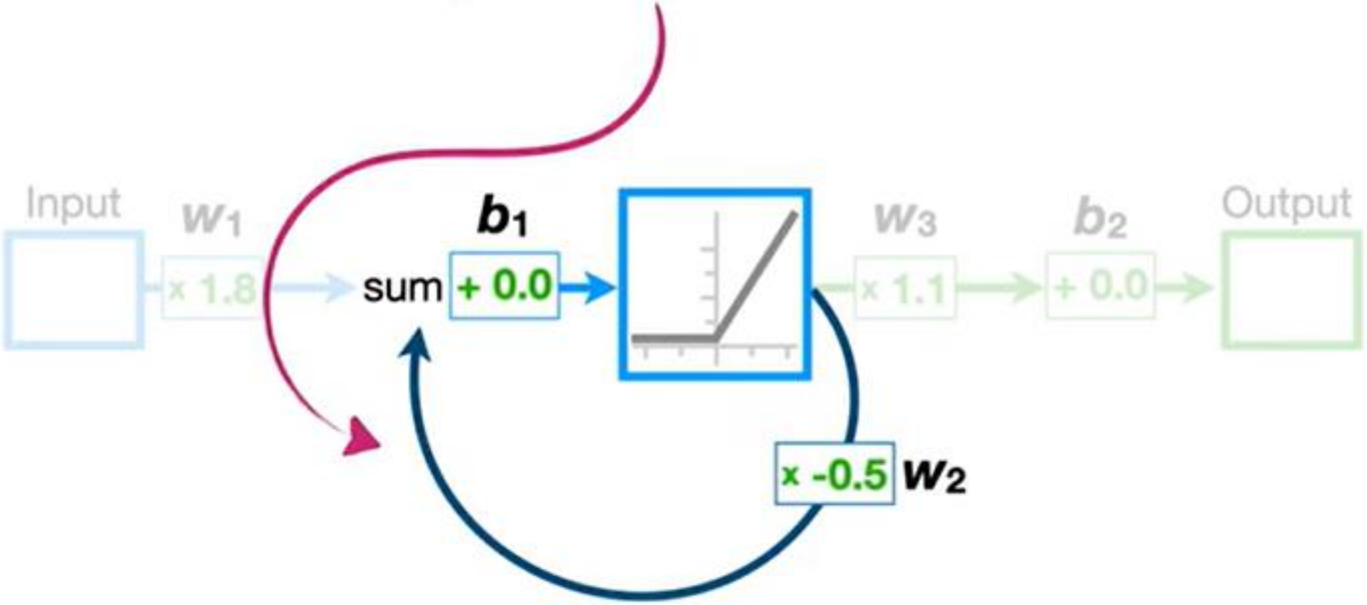
Just like the other neural networks that we've seen before, **Recurrent Neural Networks** have **weights**, **biases**,



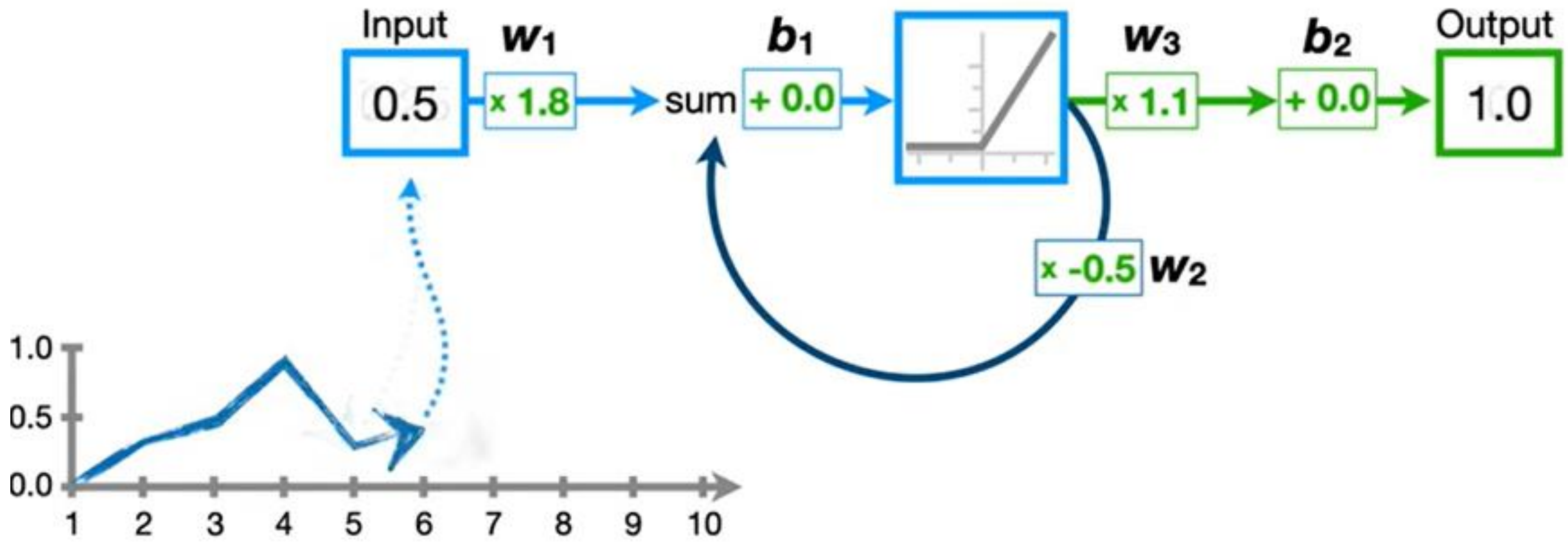
Just like the other neural networks that we've seen before, **Recurrent Neural Networks** have **weights, biases, layers** and **activation functions.**



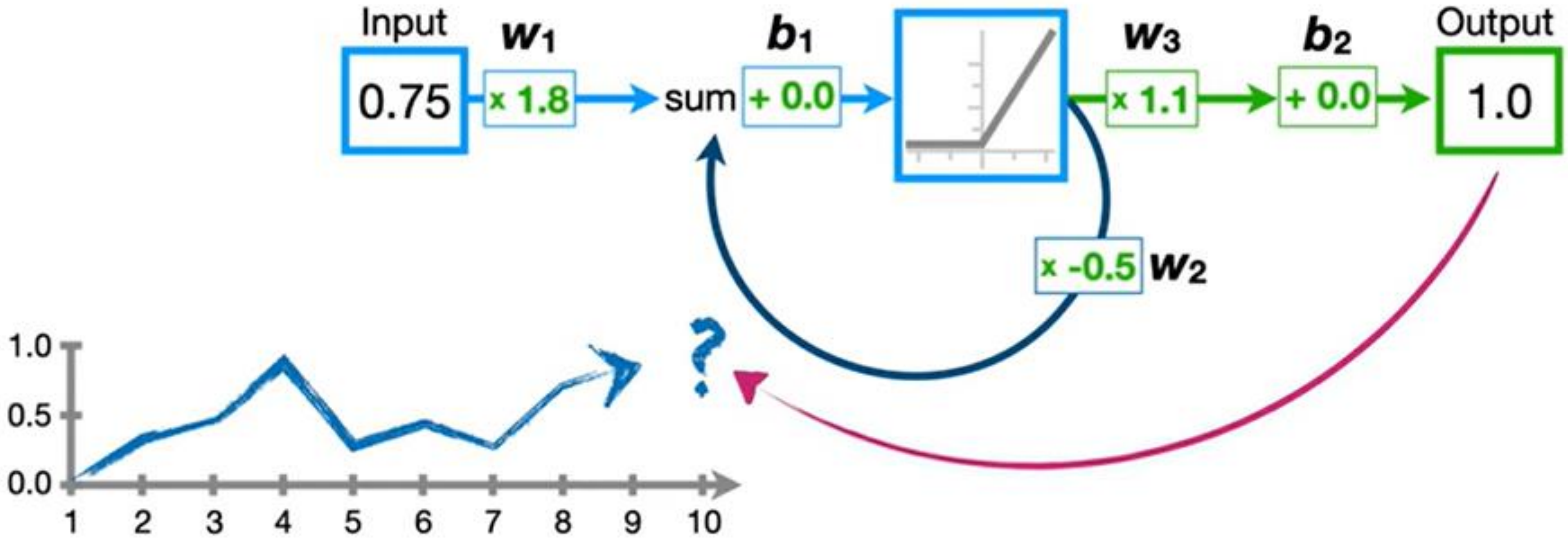
The big difference is that **Recurrent Neural Networks** also have **feedback loops.**

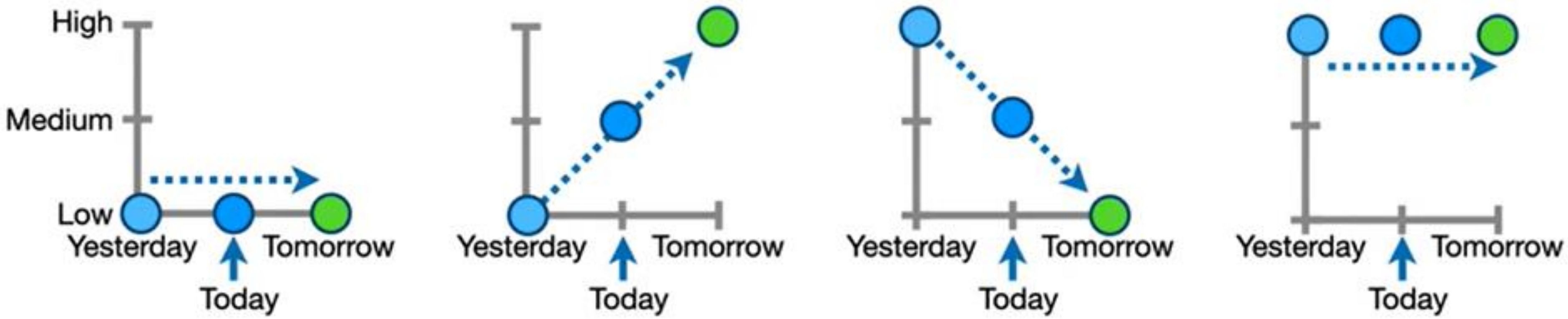


…the **feedback loop** makes it possible to use *sequential* input values, like stock market prices collected over time, to make predictions.

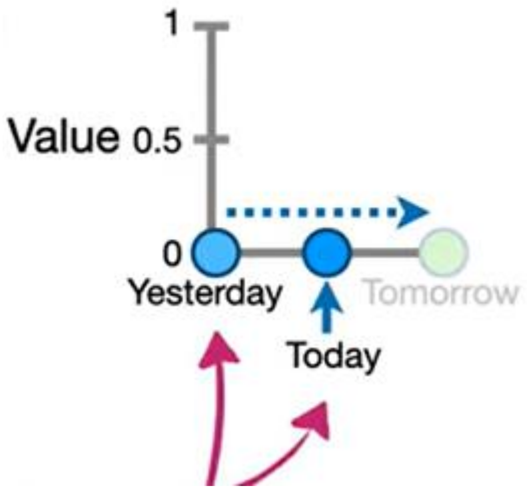


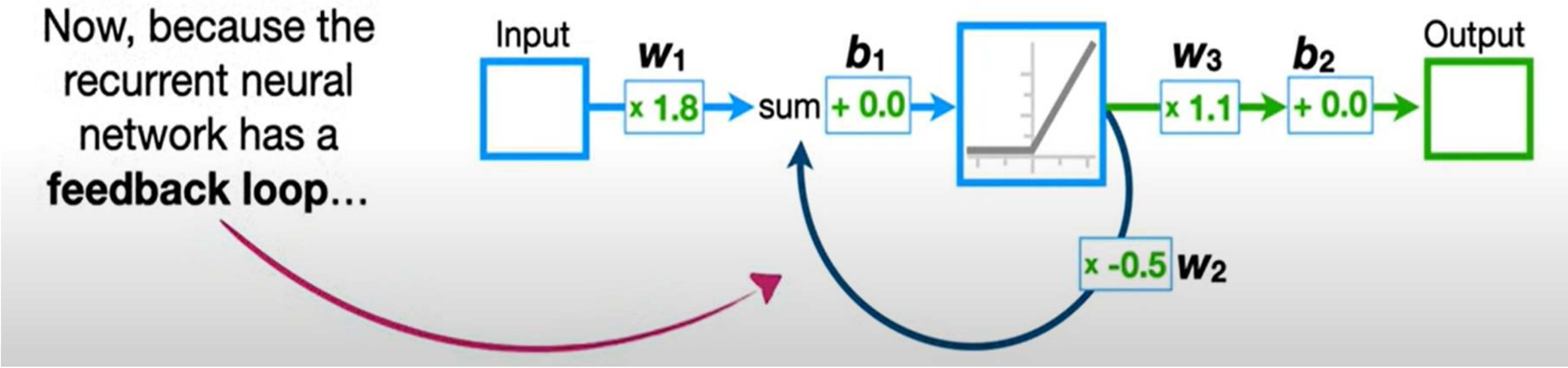
To understand how, exactly, this **Recurrent Neural Network** can make predictions with sequential input values…

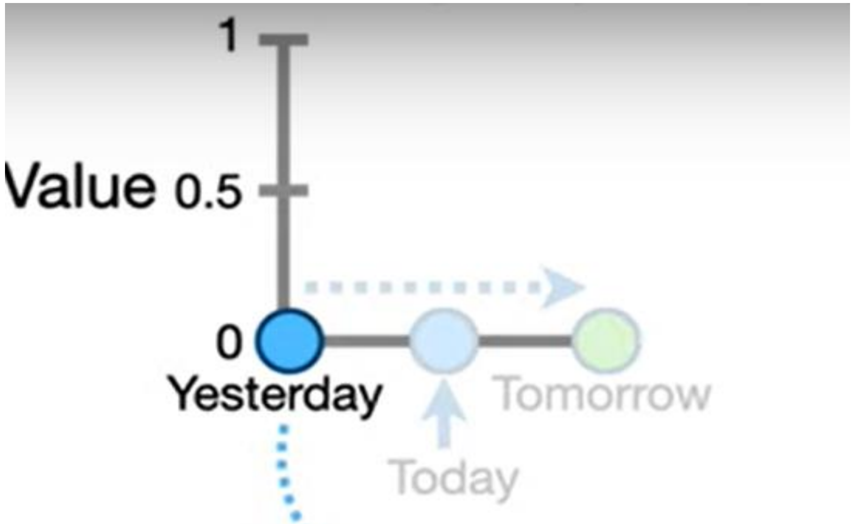




...we can talk about how to run **yesterday** and **today's** data through a **Recurrent Neural Network** to predict **tomorrow's** price.


1
Value 0.5
0
Yesterday
Today
Tomorrow


Now, because the recurrent neural network has a **feedback loop**…
Input
w_1
x 1.8
sum
b_1
+ 0.0
w_3
x 1.1
b_2
+ 0.0
Output
x -0.5
w_2

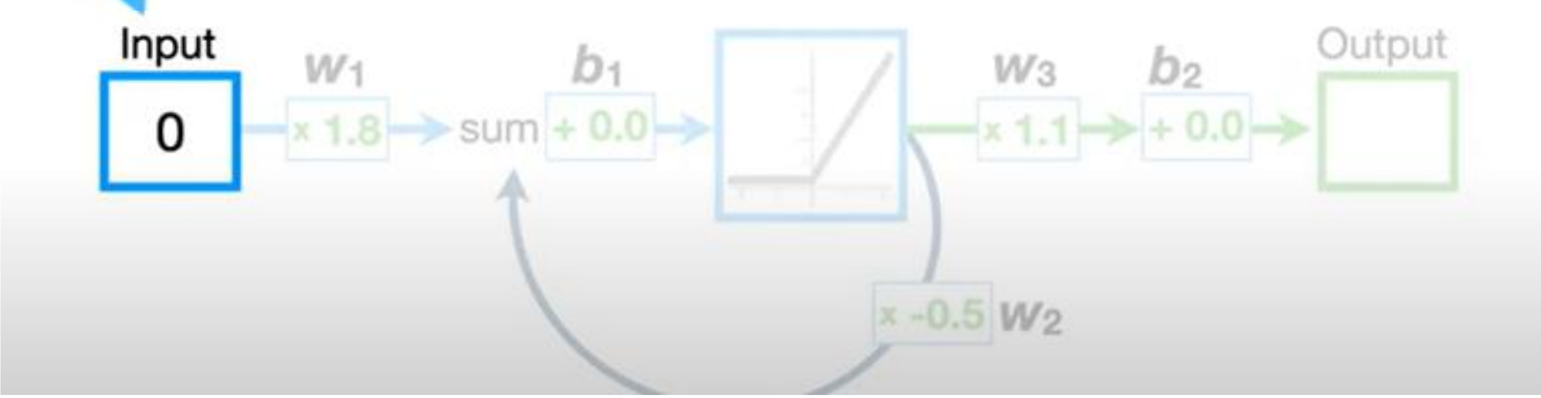


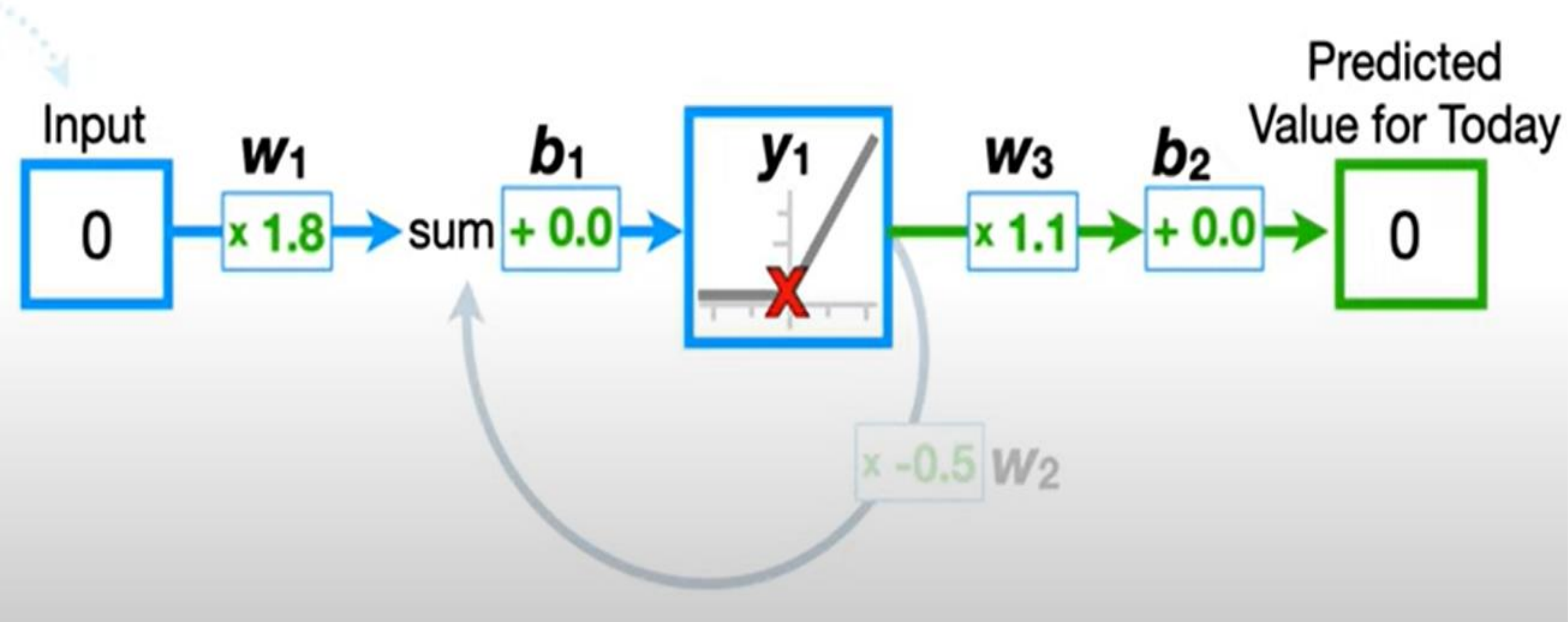
Yesterday $\times\, w_1 + b_1$ = x-axis coordinate

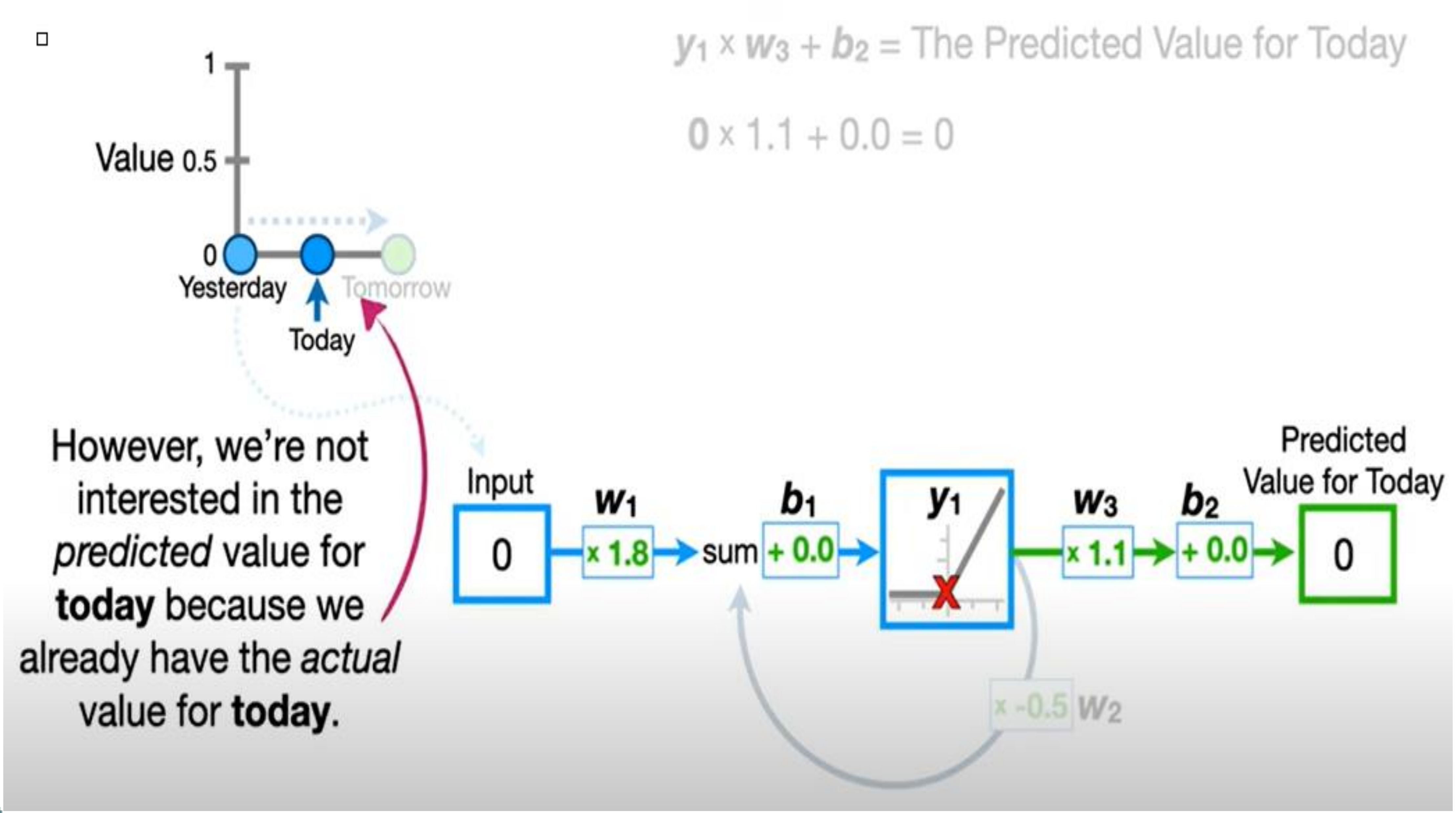
$\mathbf{0} \times 1.8 + 0.0 = 0$

$f(0) = \max(0, 0)$ = y-axis coordinate

Now we can do the math just like we would for any other neural network.



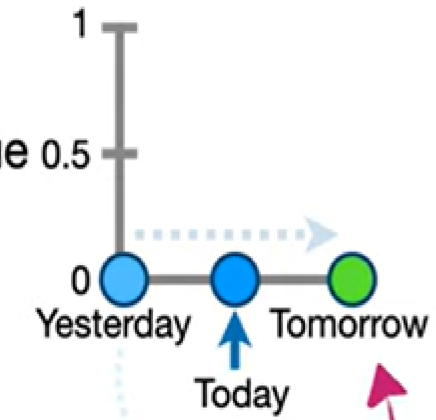

Input
0
w_1
x 1.8
sum
b_1
+ 0.0
y_1
w_3
x 1.1
b_2
+ 0.0
Predicted
Value for Today
0
x -0.5
w_2



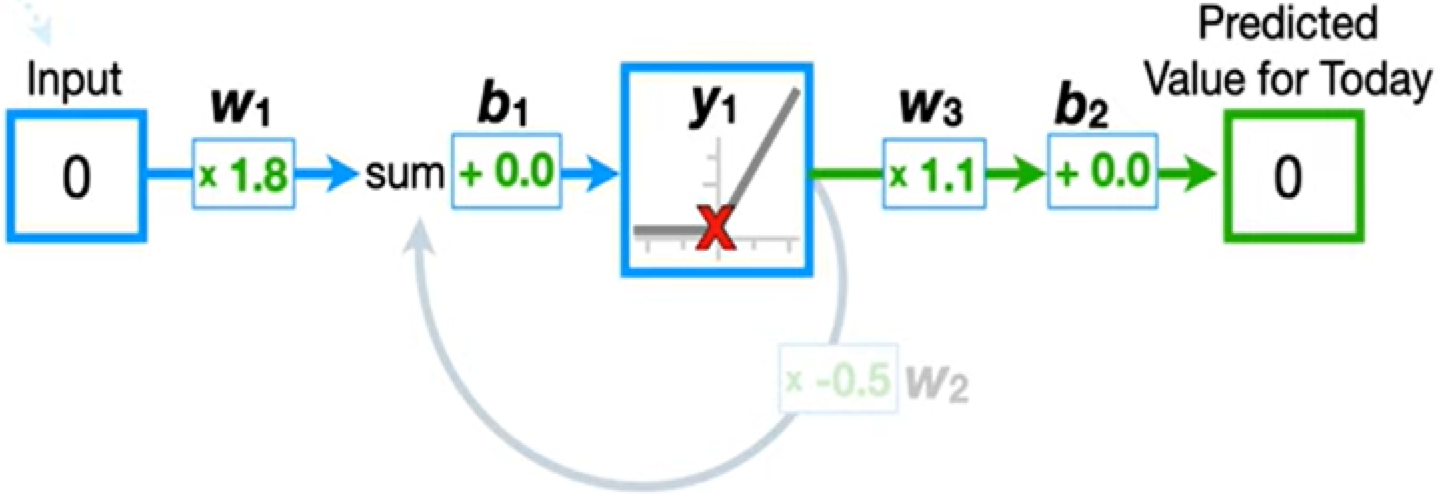
$y_1 \times w_3 + b_2$ = The Predicted Value for Today
$0 \times 1.1 + 0.0 = 0$
1
Value 0.5
0
Yesterday
Today
Tomorrow
However, we're not interested in the *predicted* value for **today** because we already have the *actual* value for **today**.
Input
0
w_1
x 1.8
sum
b_1
+ 0.0
y_1
w_3
x 1.1
b_2
+ 0.0
Predicted Value for Today
0
x -0.5
w_2

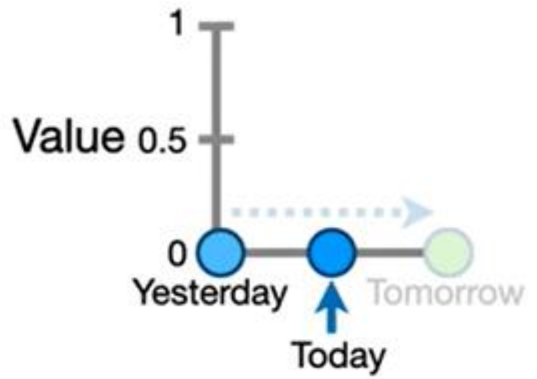
$y_1 \times w_3 + b_2$ = The Predicted Value for Today

$0 \times 1.1 + 0.0 = 0$

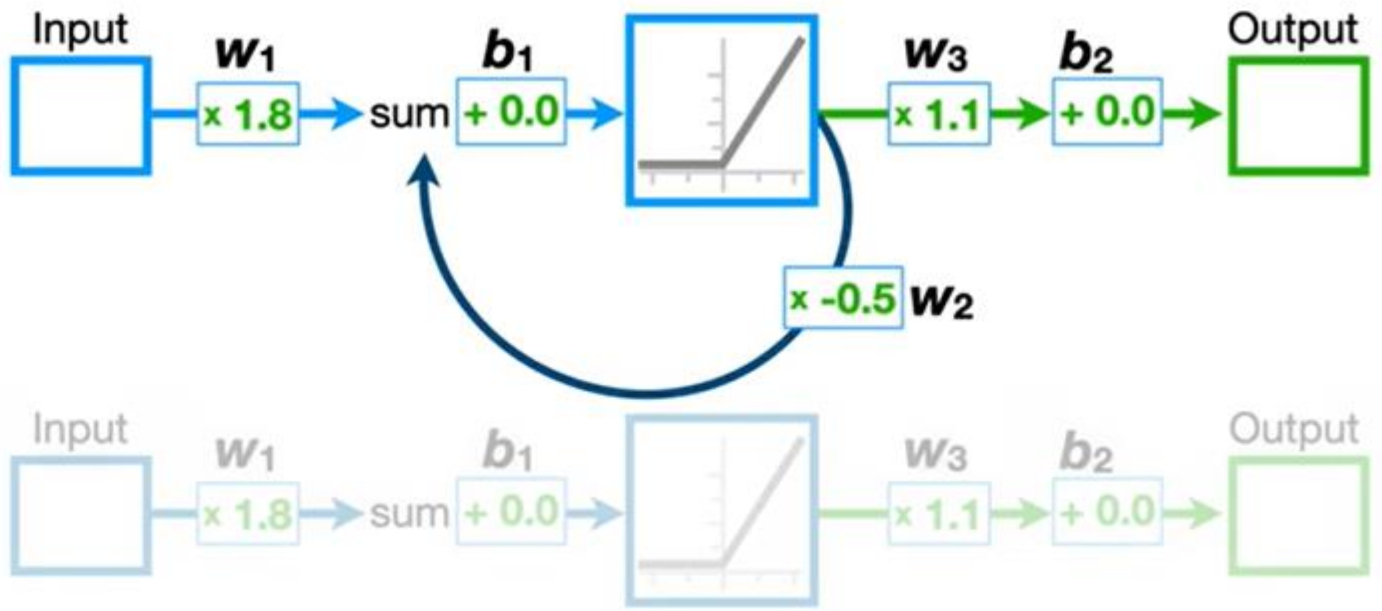


Instead, we want to use both **yesterday** and **today's** value to predict **tomorrow's** value.

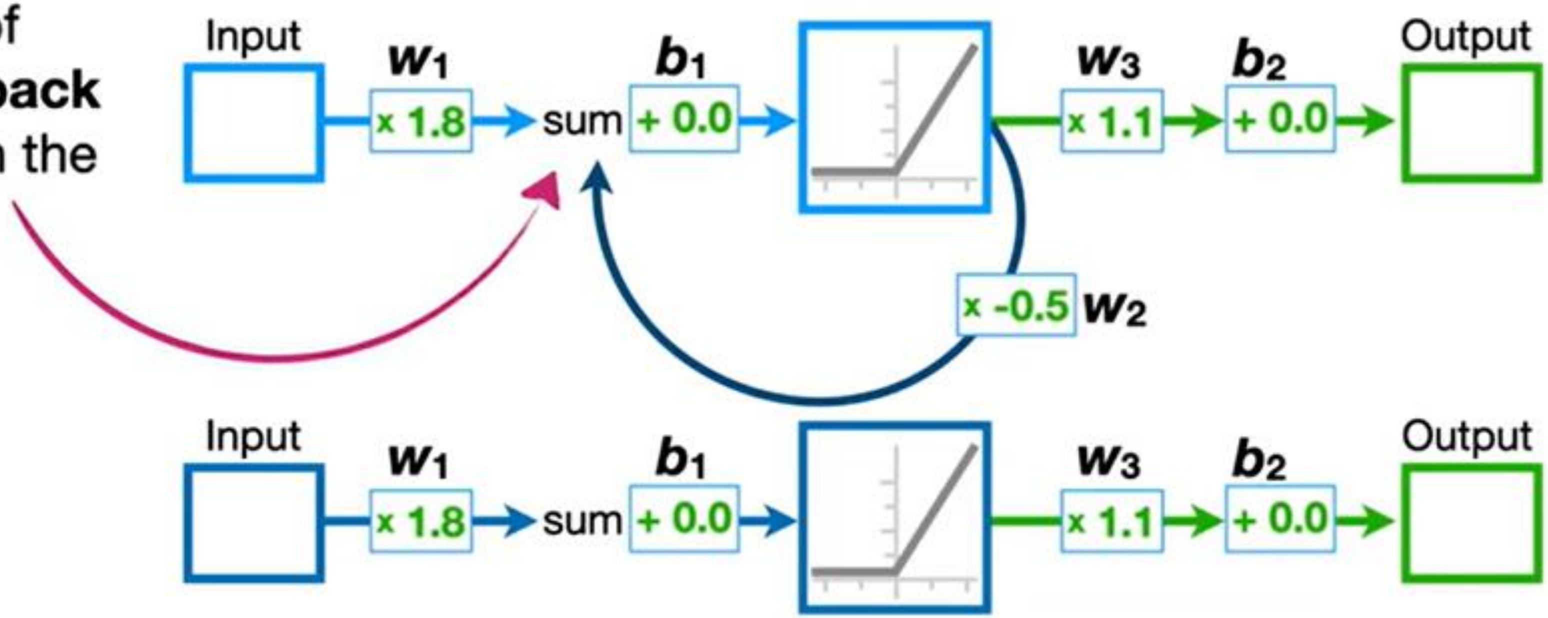




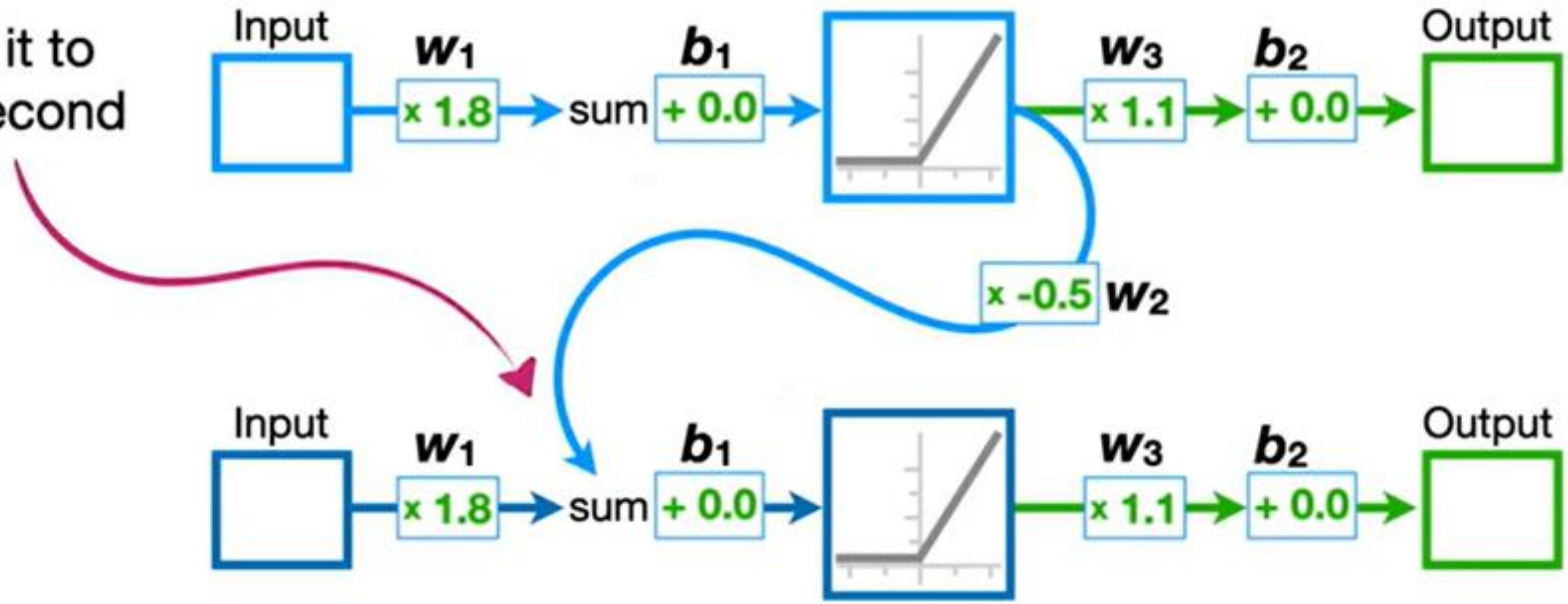
…we can **unroll** the feedback loop by making a copy of the neural network for each input value.



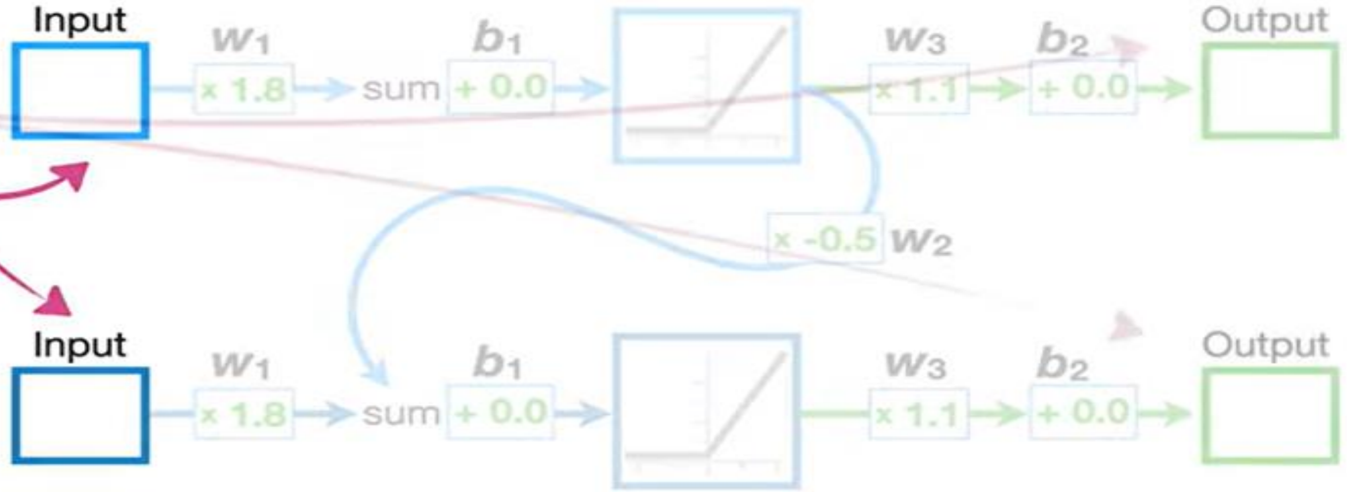
Now, instead of pointing the **feedback loop** to the sum in the first copy…

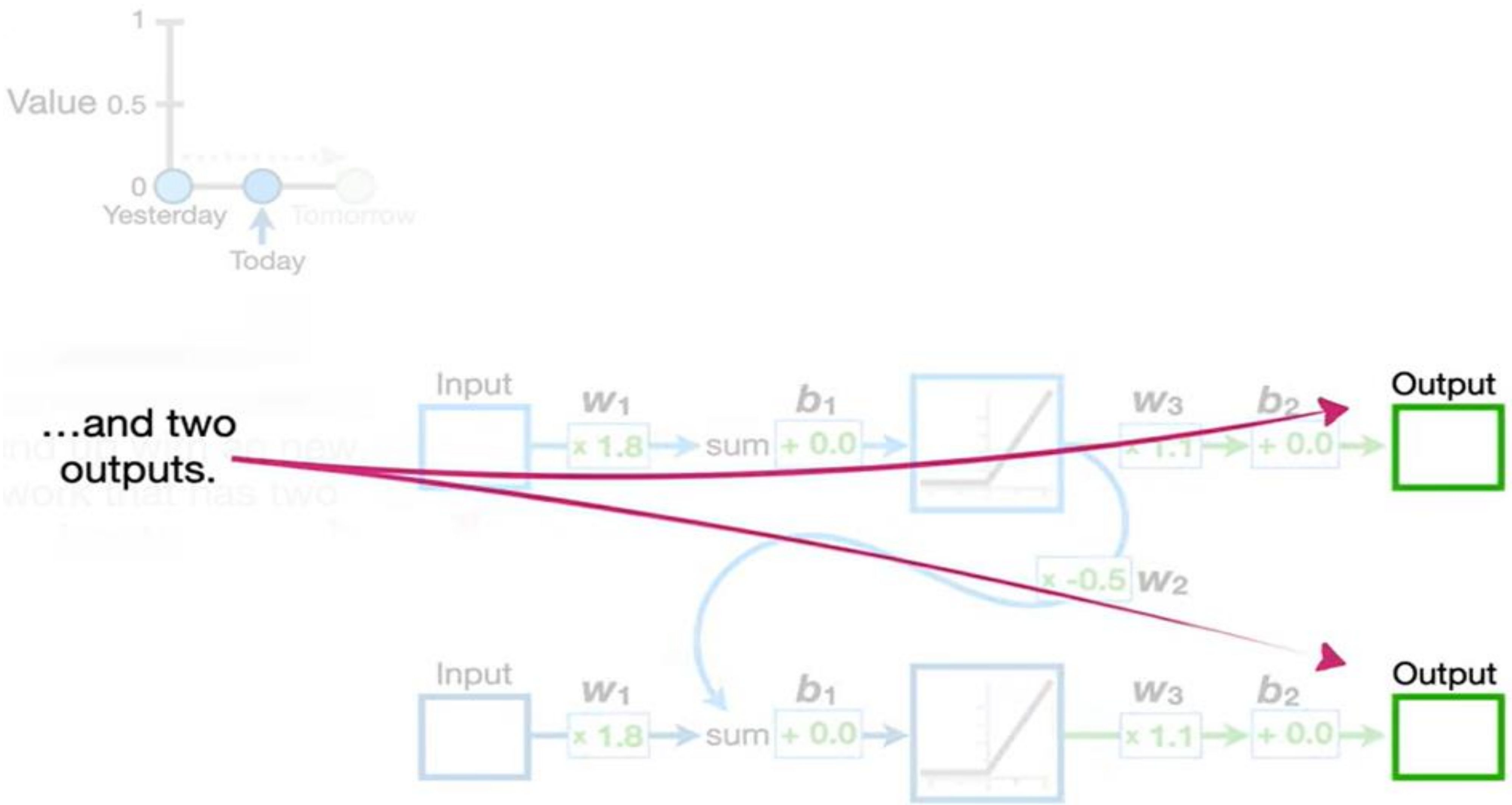


...we can point it to the sum in the second copy.

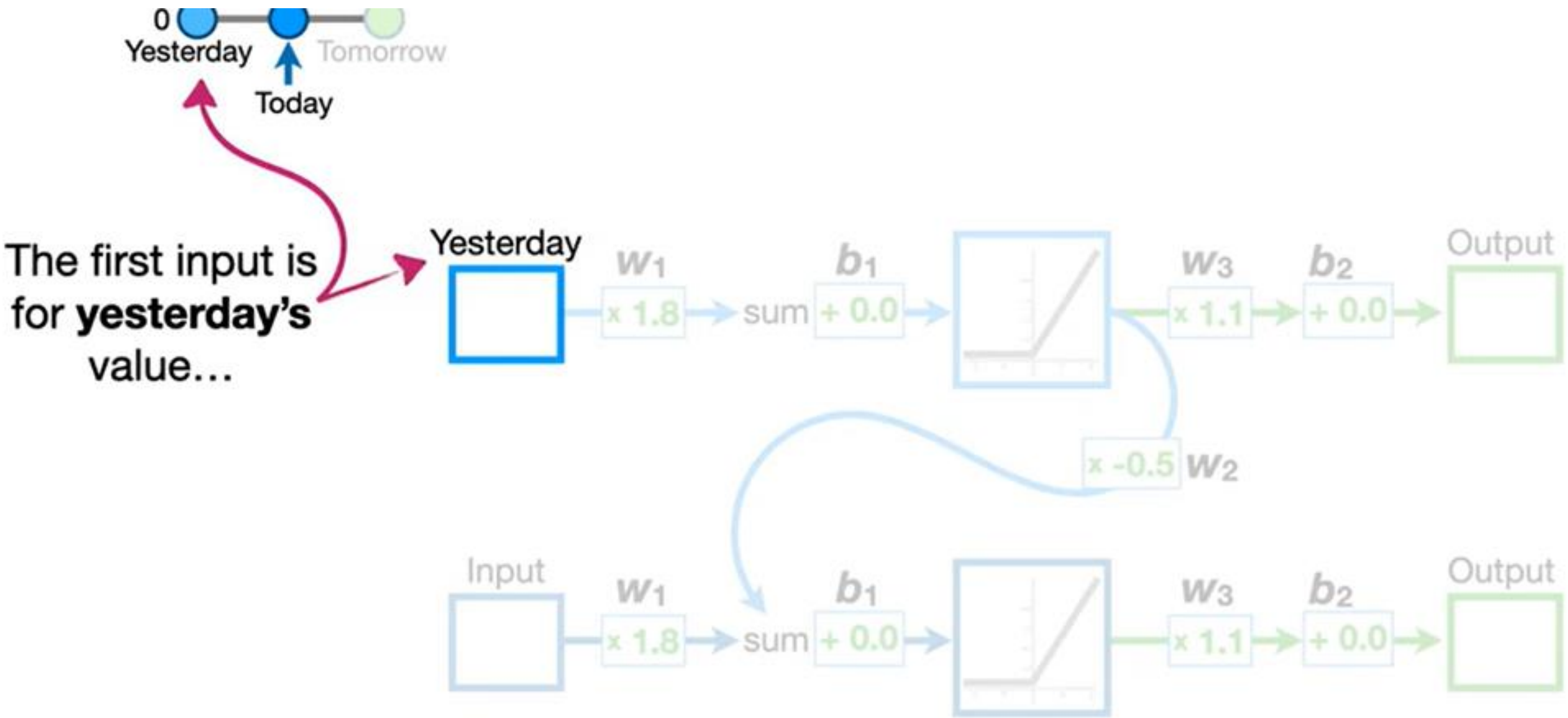


By **unrolling** the recurrent neural network, we end up with an new network that has two inputs...

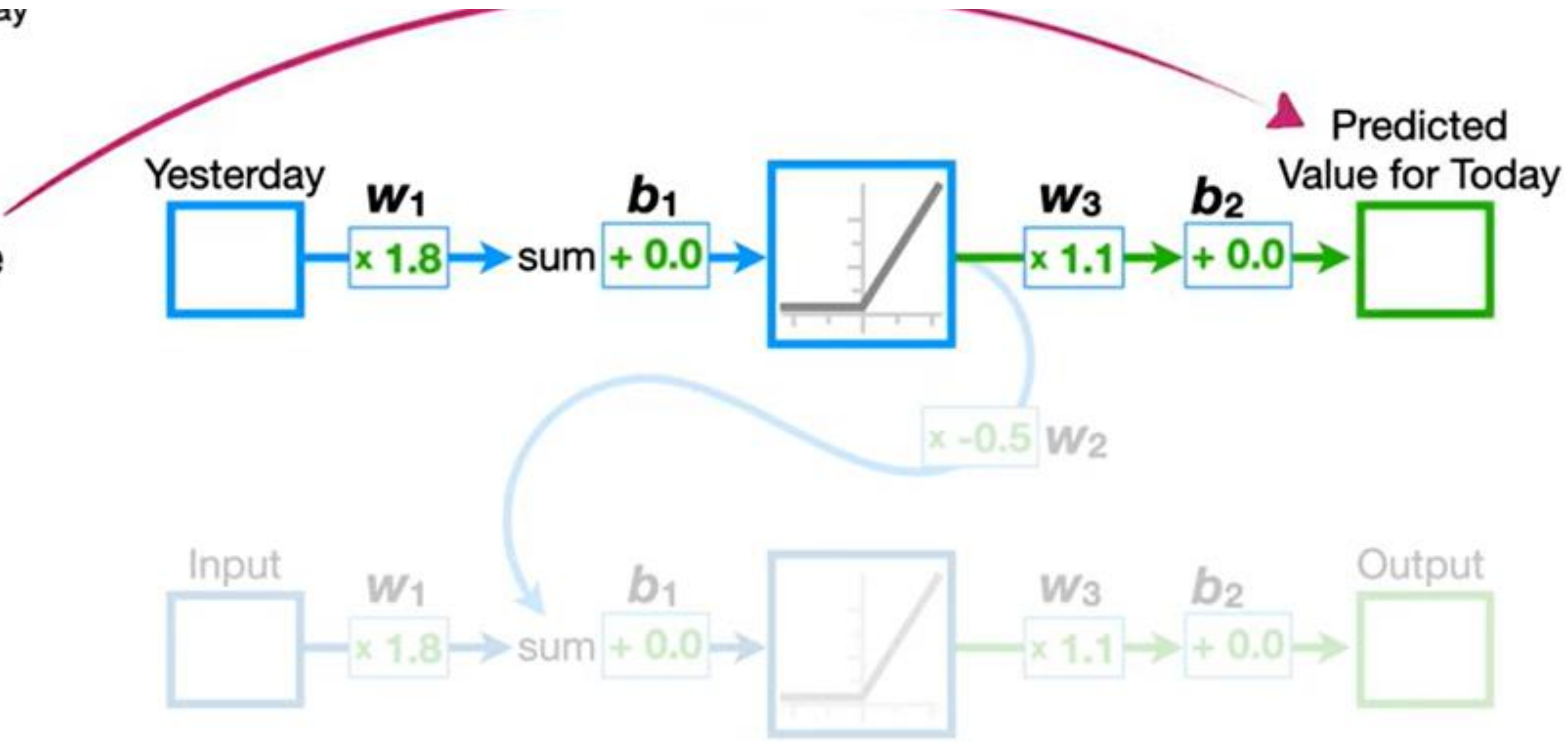




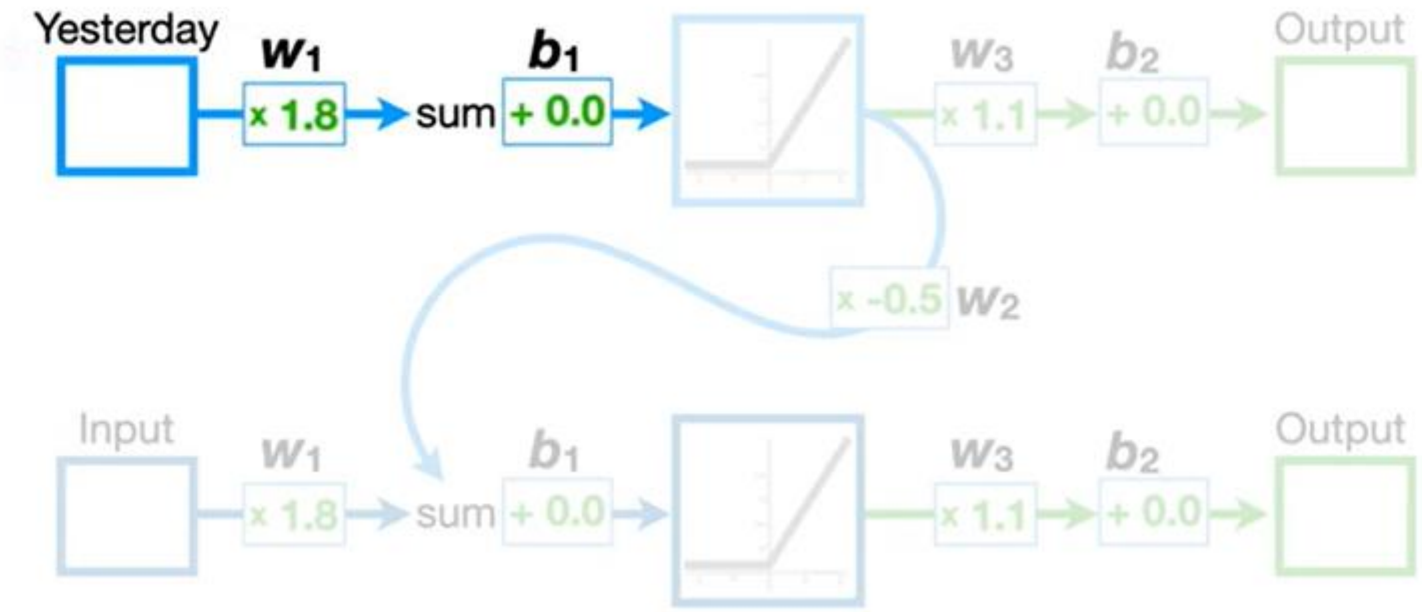
1
Value 0.5
0
Yesterday
Today
Tomorrow
...and two outputs.
Input
w_1
x 1.8
sum
b_1
+ 0.0
w_3
x 1.1
b_2
+ 0.0
Output
x -0.5 w_2
Input
w_1
x 1.8
sum
b_1
+ 0.0
w_3
x 1.1
b_2
+ 0.0
Output


0
Yesterday
Today
Tomorrow
The first input is for **yesterday's** value...
Yesterday
w_1
x 1.8
sum
b_1
+ 0.0
w_3
x 1.1
b_2
+ 0.0
Output
x -0.5 w_2
Input
w_1
x 1.8
sum
b_1
+ 0.0
w_3
x 1.1
b_2
+ 0.0
Output

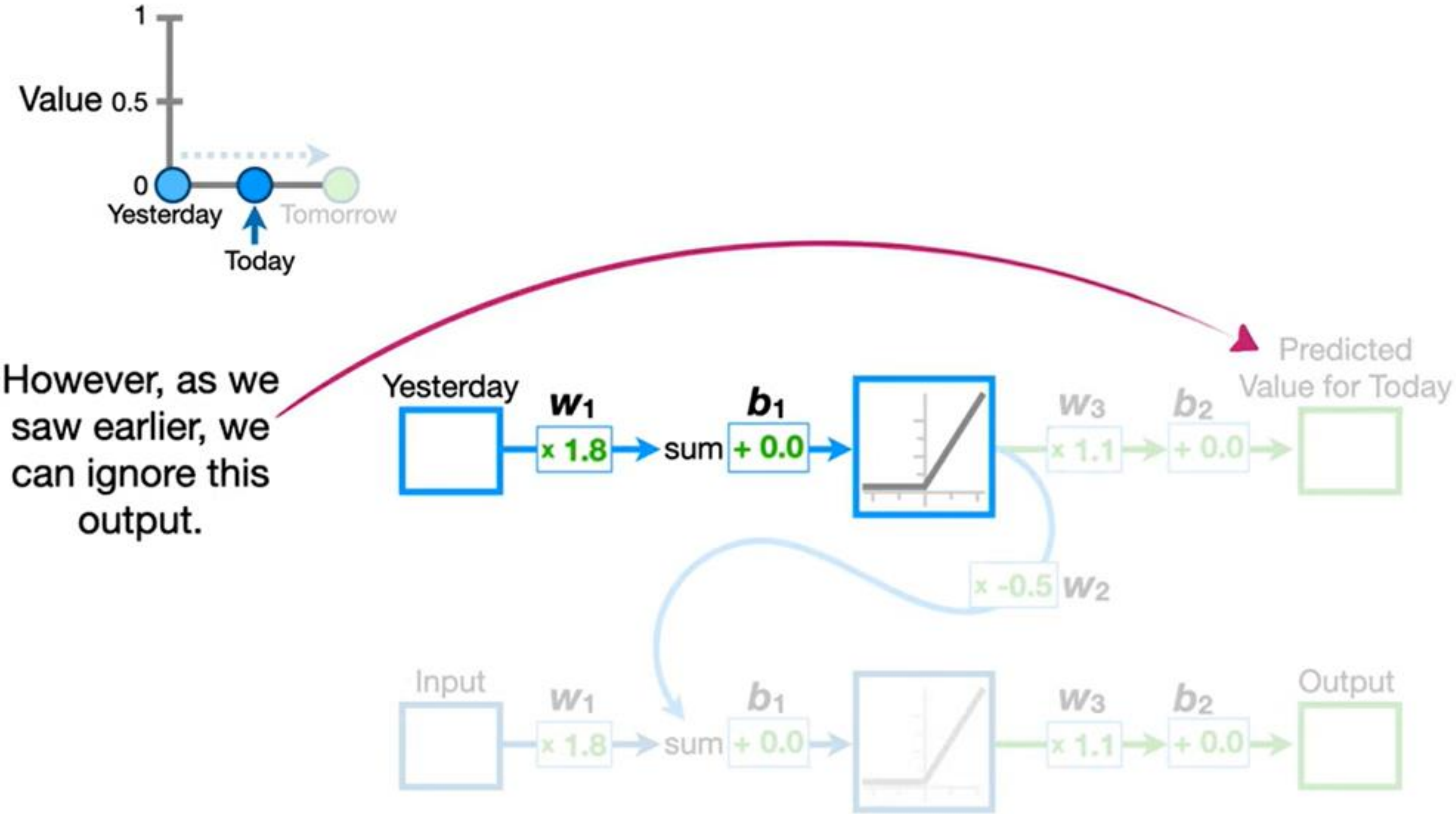
…we get the predicted value for **today**.

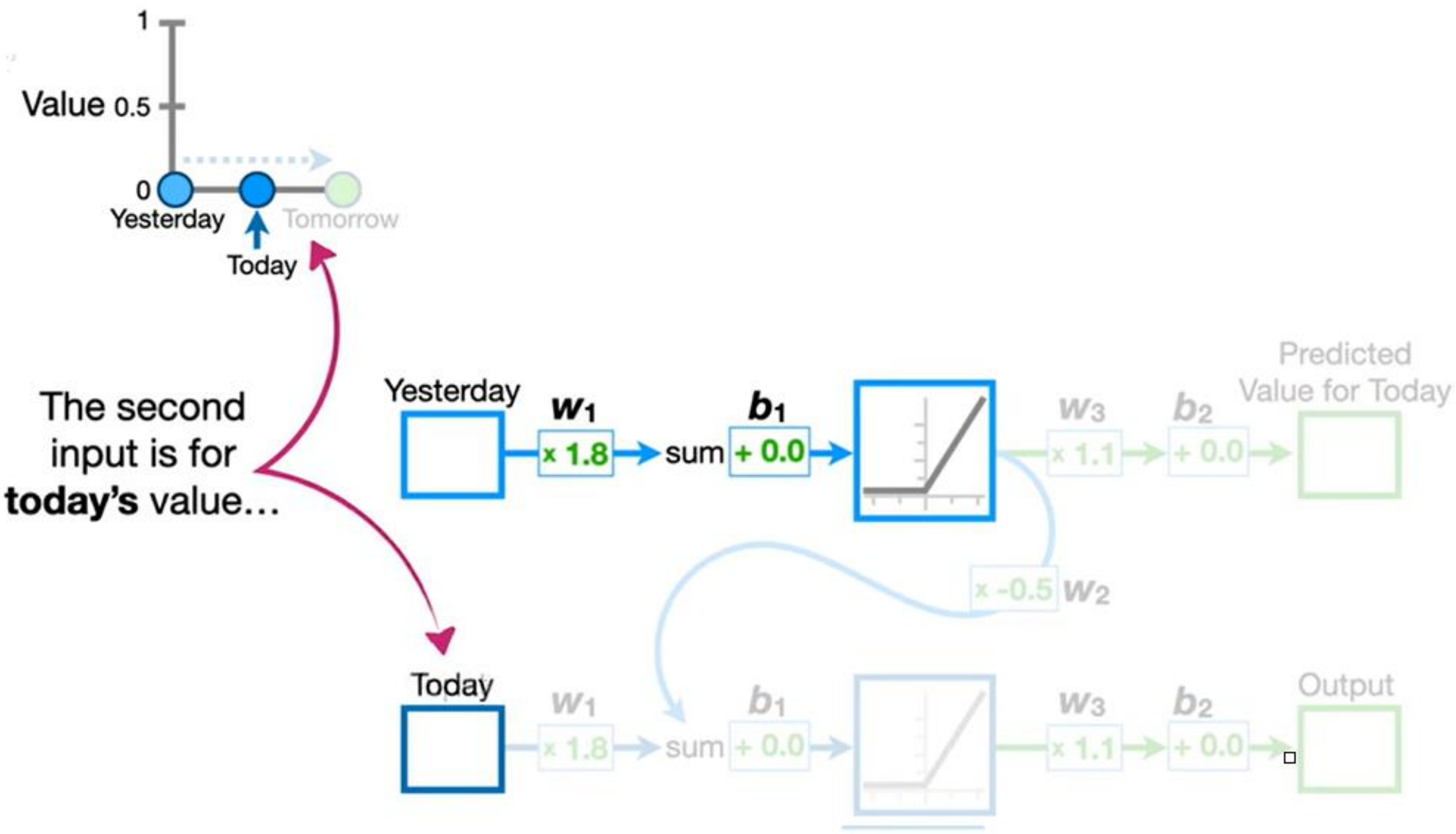


...and if we do the math straight through to the first output like we did earlier...

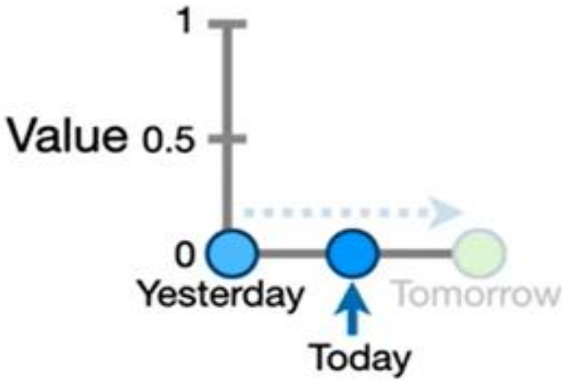


However, as we saw earlier, we can ignore this output.

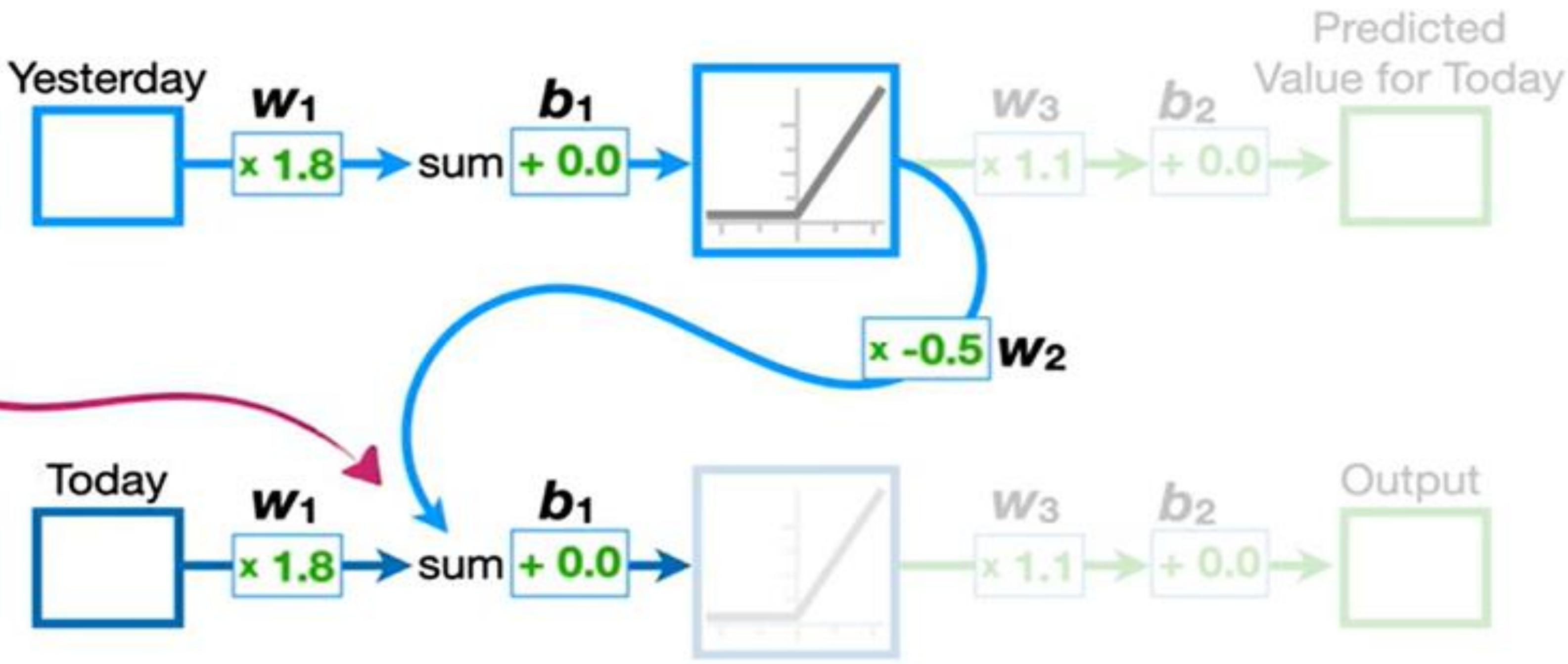


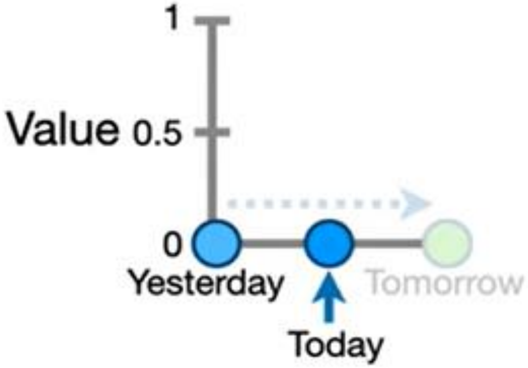


The second input is for **today's** value…

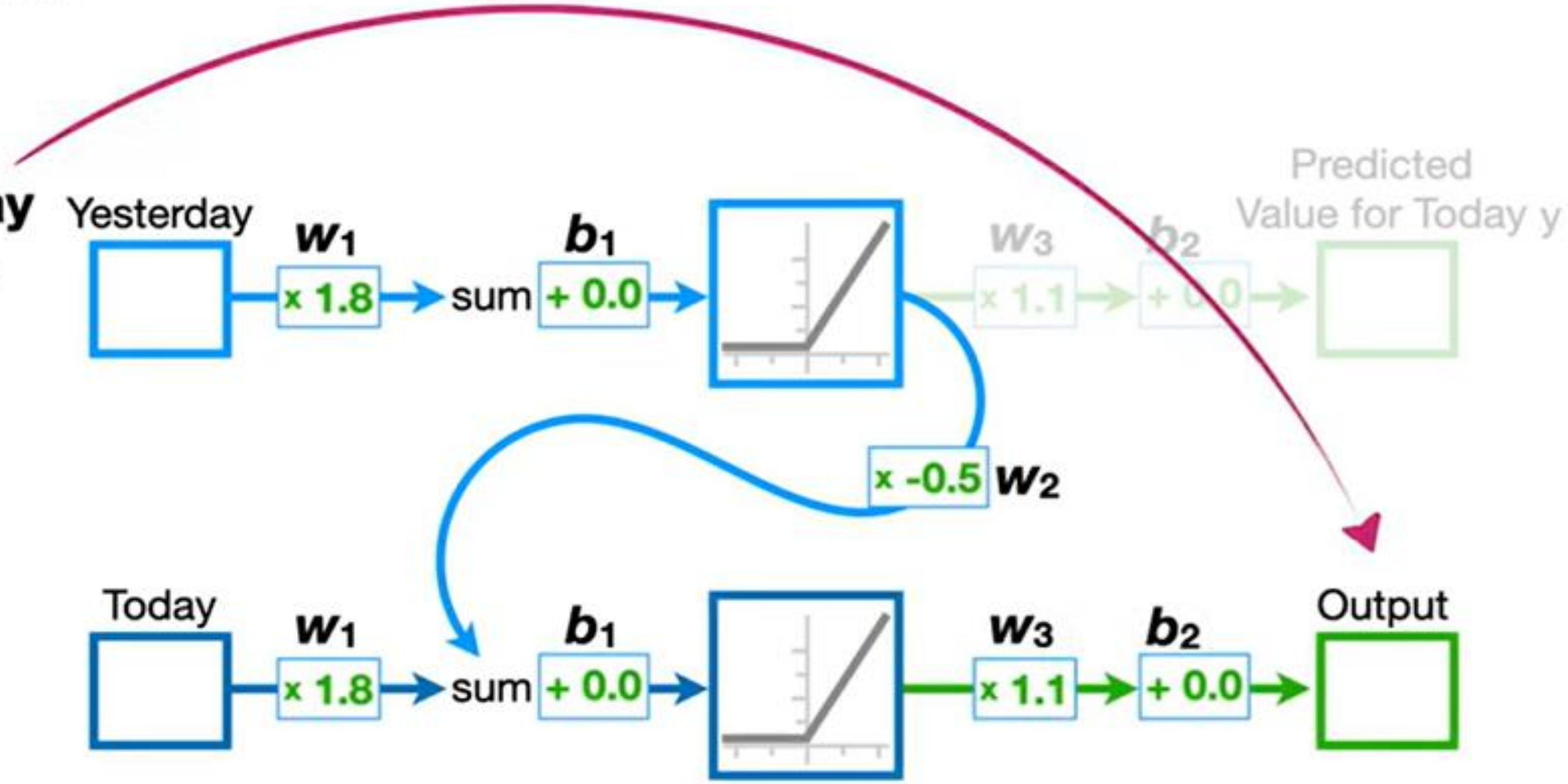


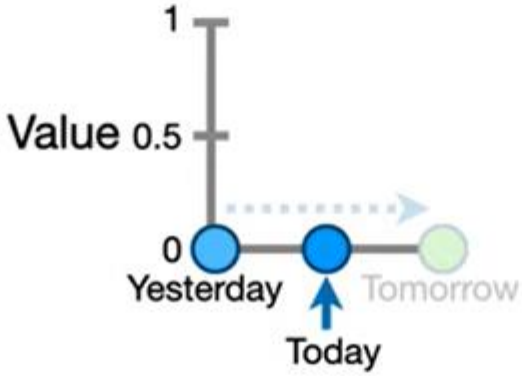
...and the connection between the first activation function and the second summation...



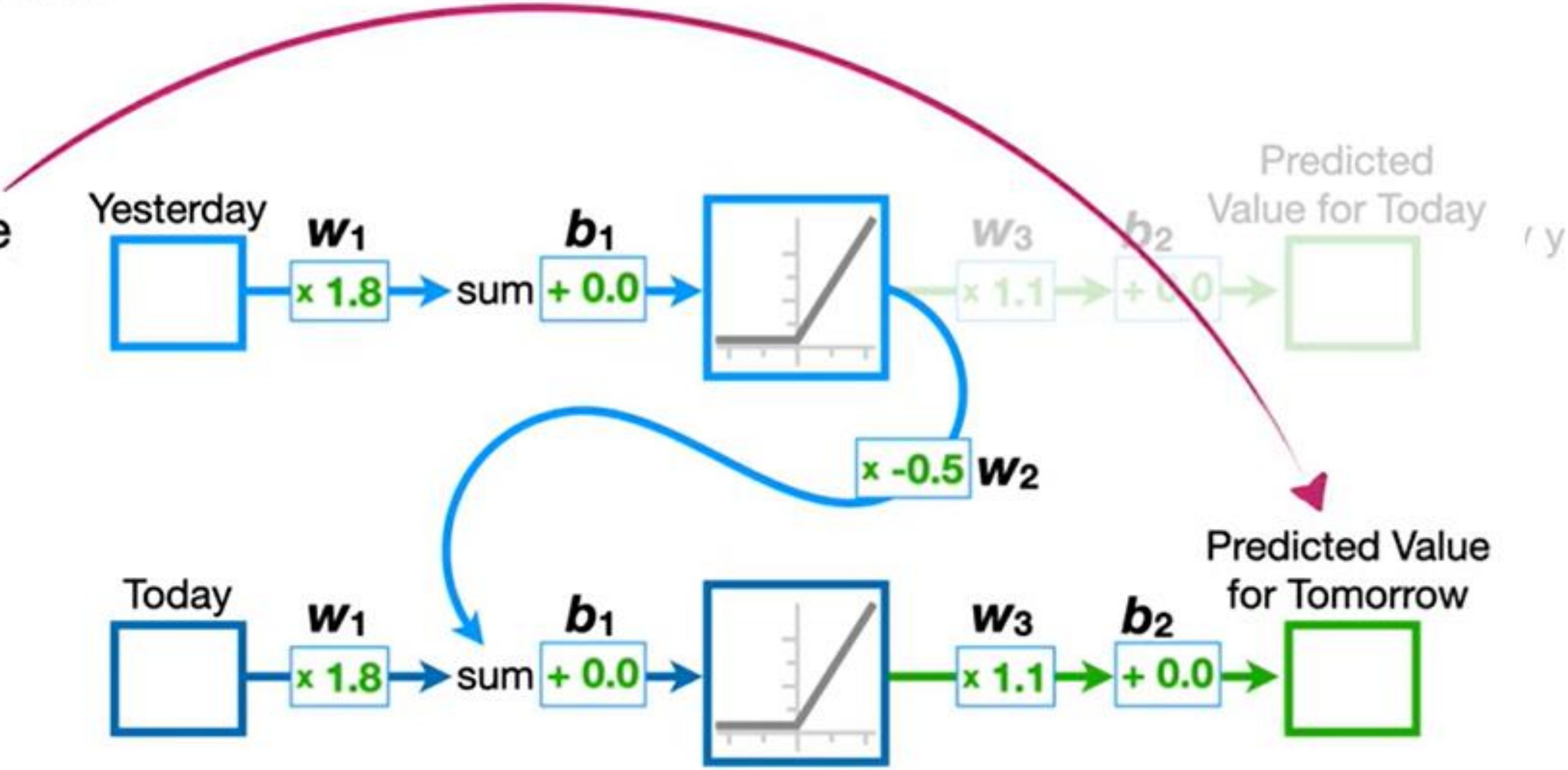


...allows both **yesterday** and **today's** values to influence the final output...



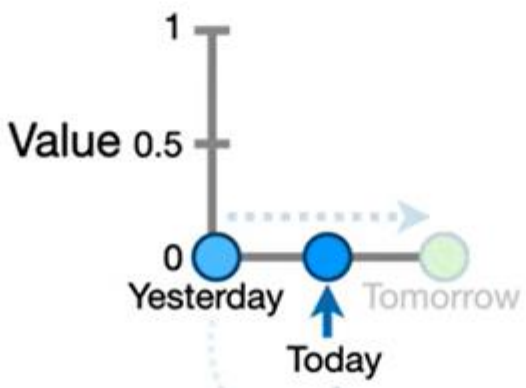


...which gives us the predicted value for **tomorrow.**

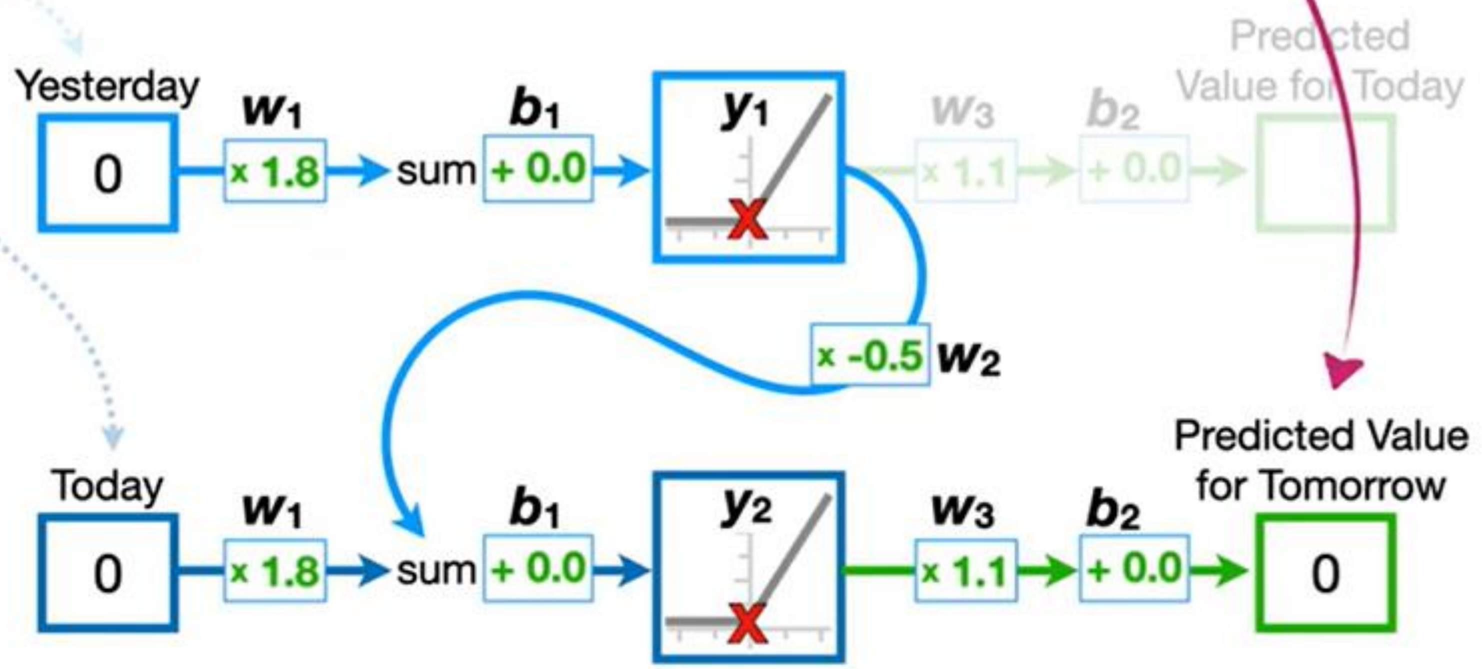


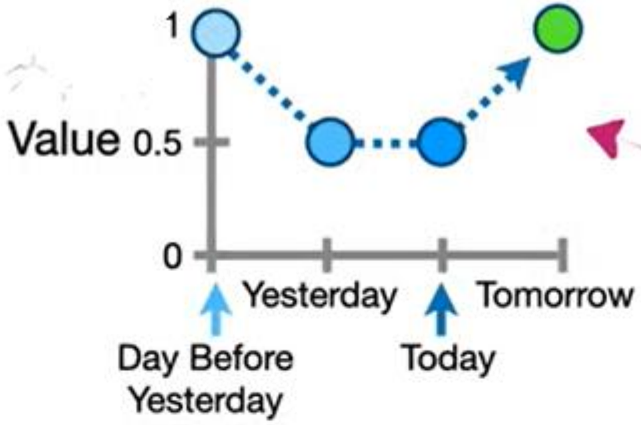
$$y_2 \times w_3 + b_2 = \text{Prediction for Tomorrow}$$

$$\mathbf{0} \times 1.1 + 0.0 = 0$$

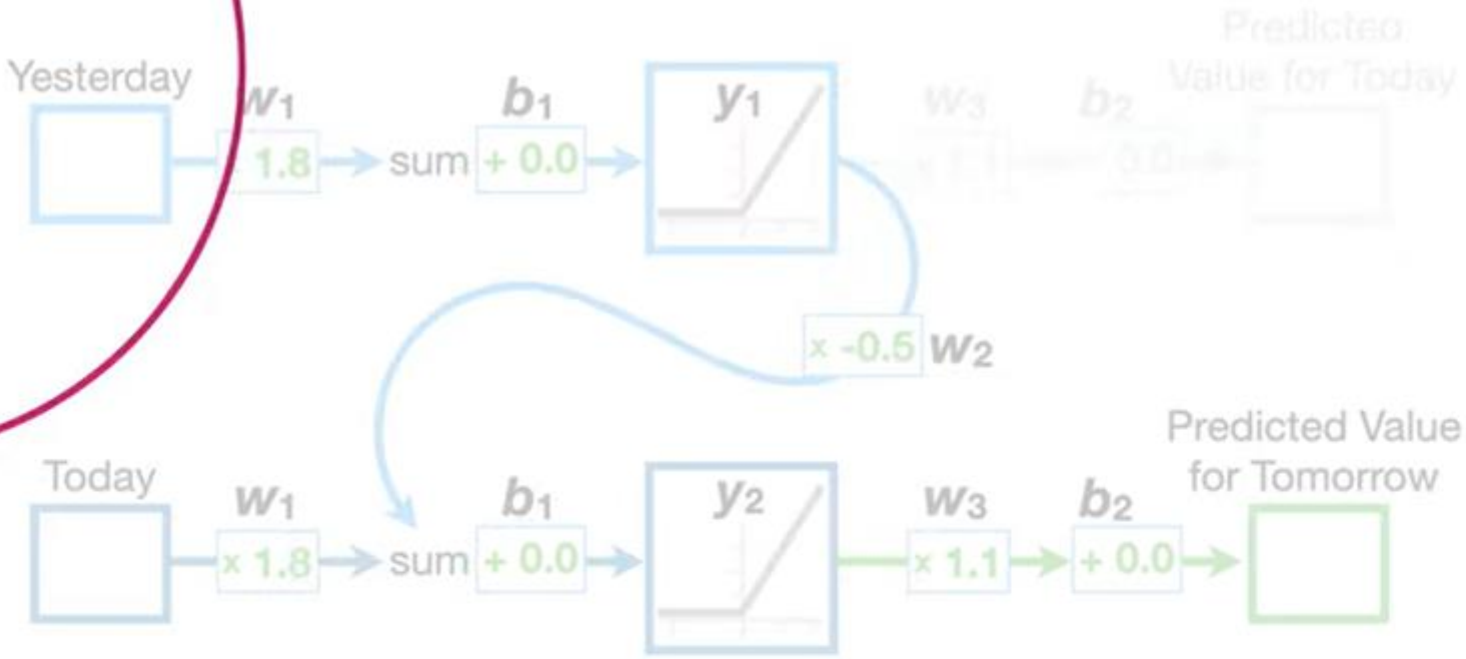


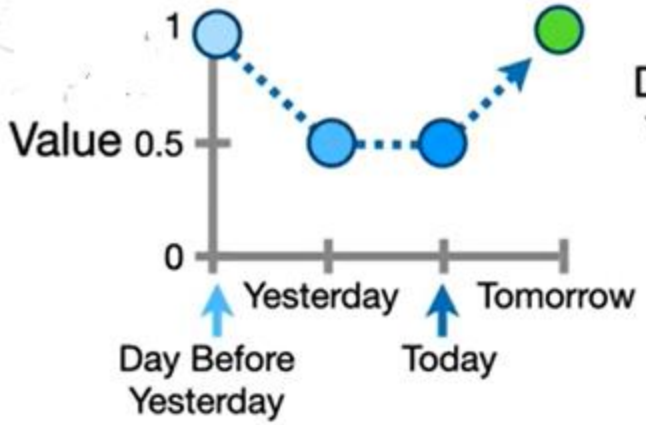
...and that gives us a predicted value for **tomorrow, 0**...



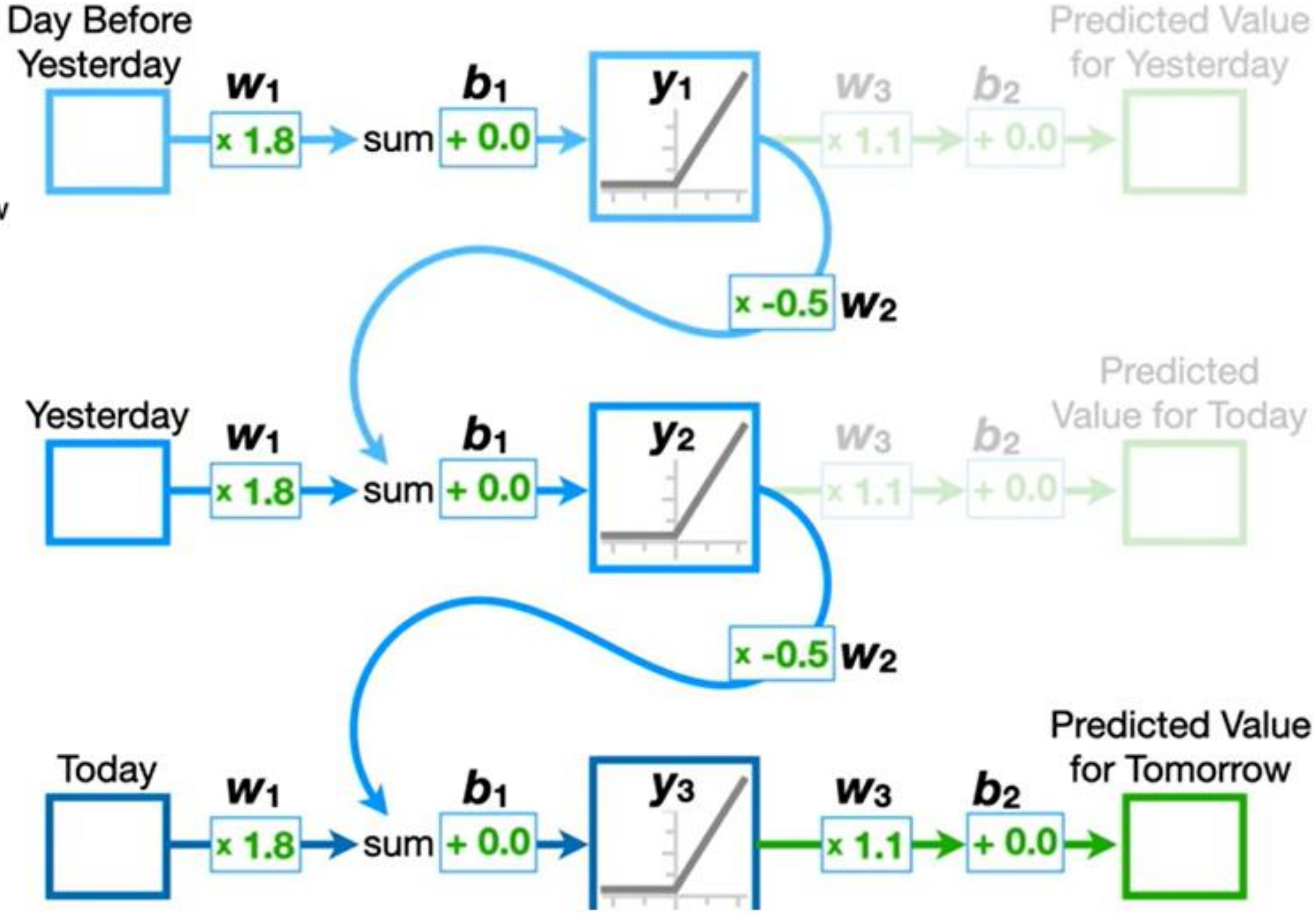


When we want to use **3** days of data to make a prediction about **tomorrow's** price, like this…





...then we just keep **unrolling** the recurrent neural network until we have an input for each day of data.



NOTE: Regardless of how many times we **unroll** a recurrent neural network, the **weights** and **biases** are shared across every input.