Introduction to DL, RNN, LSTM, CNN

24th September 2020

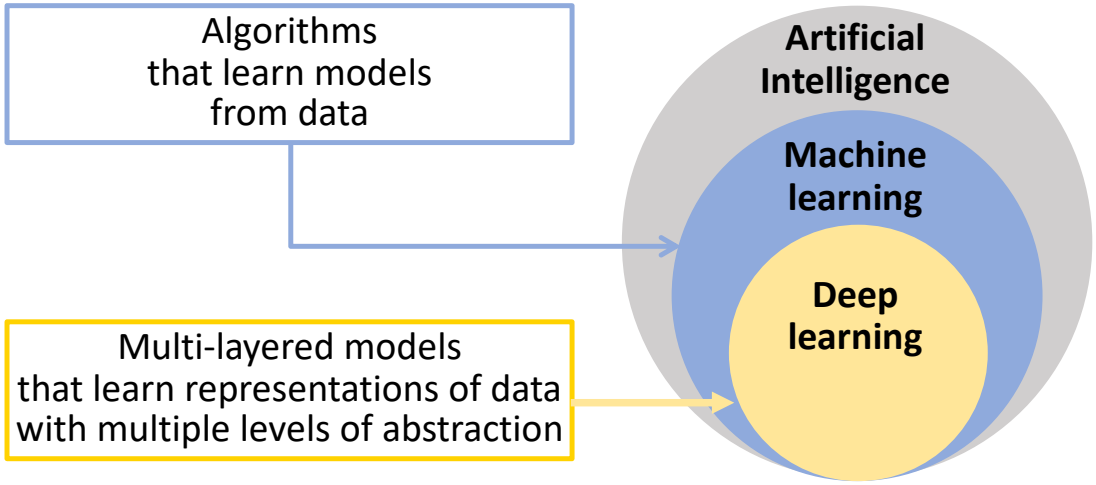
Nishtha Madaan
Research Scientist
India Research Labs, IBM

Sattwati Kundu
Data Scientist
India Software Labs, IBM

AI, Machine Learning and Deep Learning

What is AI and Machine Learning?

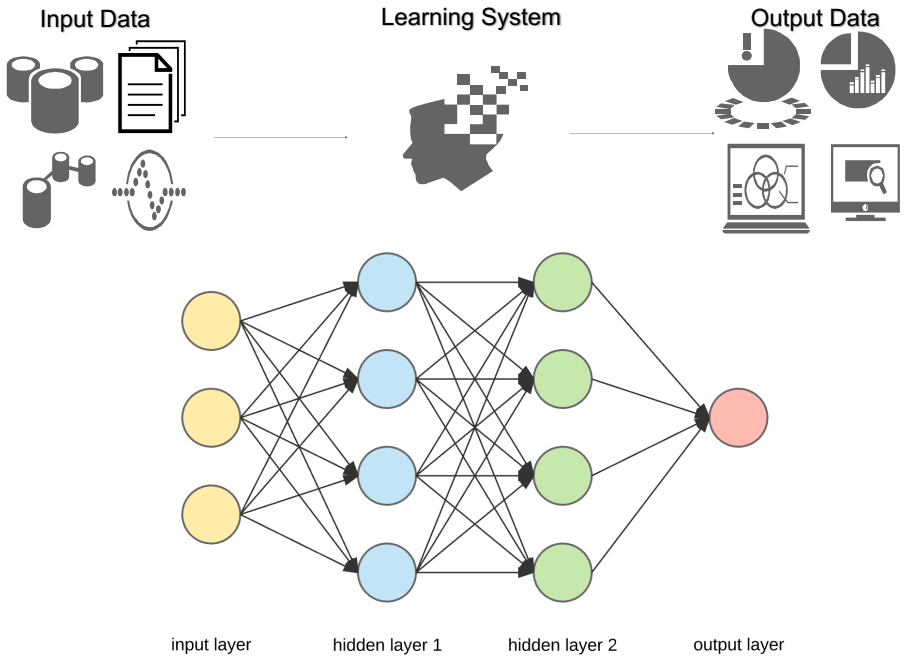
Machine Learning is a subset of **Artificial Intelligence** that uses algorithms to learn from data sets.
Nowadays, Machine learning concentrates most AI use cases.



Principles of machine learning:

1. Learn from data
2. Compare predicted vs actual
3. Re-learn to match better

What is Neural Net and Deep Learning?



Principles of Deep learning:

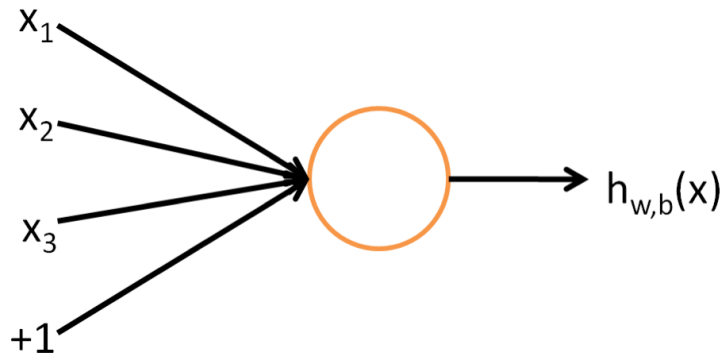
Essentially an autonomous, self-teaching system which use existing data to train algorithms to find patterns and then use that to make predictions about new data.

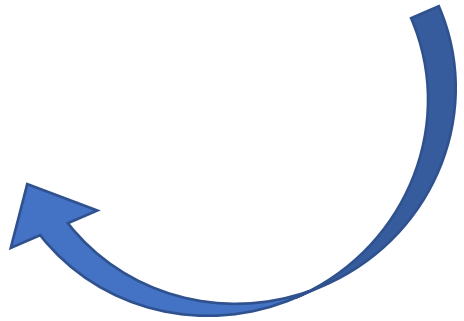
Neural networks mimic the network of neurons in our brain.

The hidden layers process equations to make predictions for output and iterate to fit the output.

Understanding Neural Networks

- As the name suggests, neural networks were inspired by the neural architecture of a human brain and like in a human brain the basic building block is called a Neuron.
- Its functionality is like a human neuron, i.e. it takes in some inputs and fires an output.
- In purely mathematical terms, a neuron in the machine learning world is a placeholder for a mathematical function, and its only job is to provide an output by applying the function on the inputs provided.

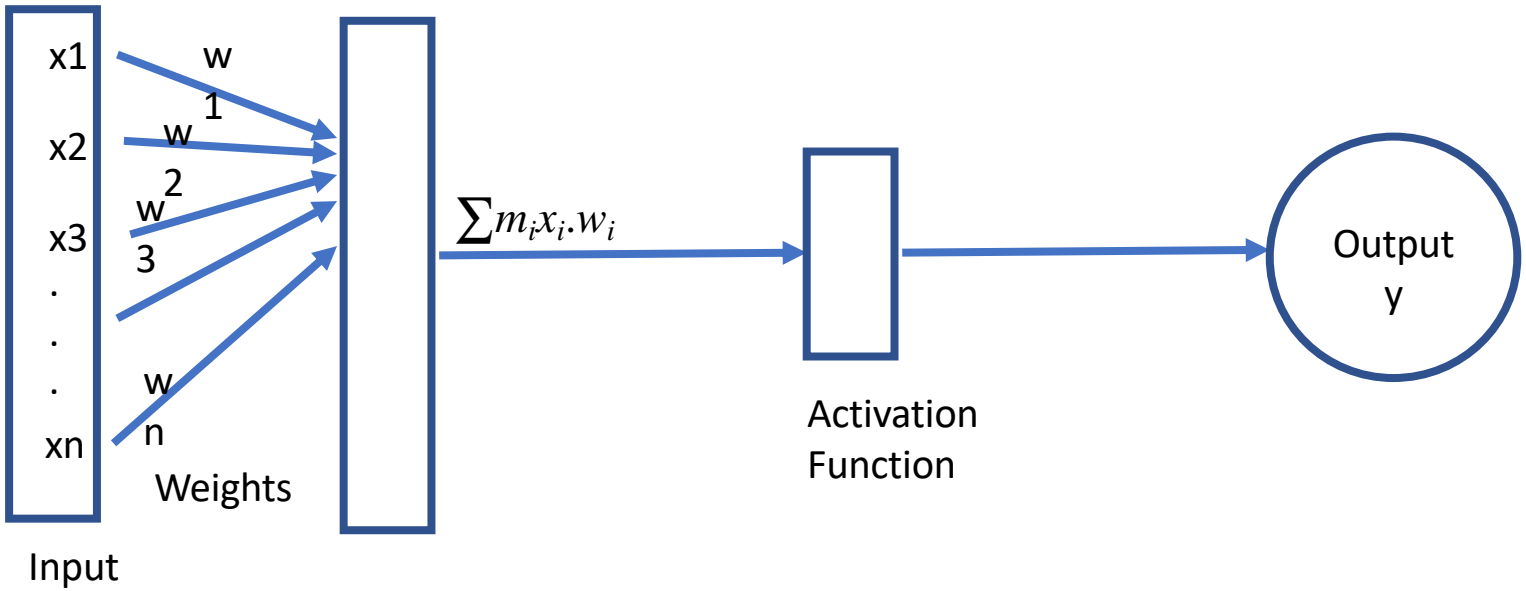




Introducing **Perceptron**

Understanding Neural Networks Contd…

- A perceptron can be understood as anything that takes multiple inputs and produces one output.



$$y_{in} = x1.w1 + x2.w2 + x3.w3 \ldots + xm.wm$$

Activation function:

Activation Function takes the sum of weighted input

(w1*x1 + w2*x2 + w3*x3 + 1*b)

as an argument and returns the output of the neuron.

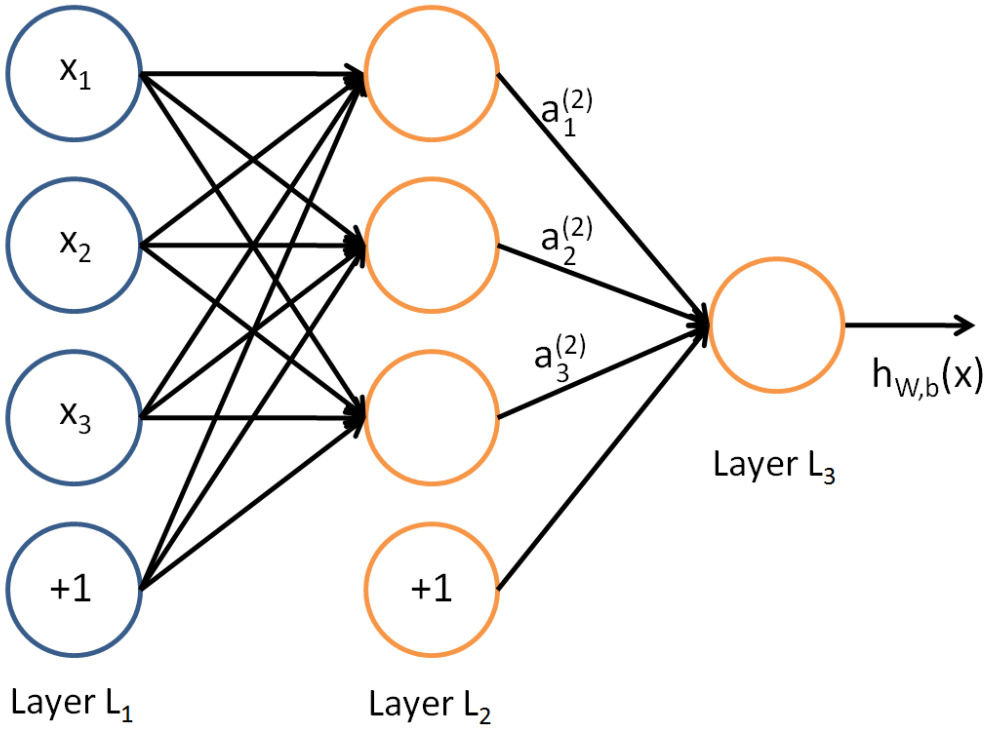
Bias

Note the term b

Bias can be thought of as how much flexible the perceptron is. It is somehow similar to the constant **b** of a linear function **y = ax + b**

Understanding Neural Networks Contd…

- A layer is nothing but a collection of neurons which take in an input and provide an output. Inputs to each of these neurons are processed through the activation functions assigned to the neurons.



The leftmost layer of the network is called the input layer, rightmost layer the output layer (which, in this example, has only one node).

The middle layer of nodes is called the hidden layer because its values are not observed in the training set.

The number of hidden layers, for instance, differ between different networks depending upon the complexity of the problem to be solved.

Another important point to note here is that each of the hidden layers can have a different activation function

Forward Propagation, Back Propagation, and Epochs

- Till now, we have computed the output and this process is known as “**Forward Propagation**“.
- But what if the estimated output is far away from the actual output (high error). In the neural network what we do, we update the biases and weights based on the error.
- This weight and bias updating process is known as **“Back Propagation“.**

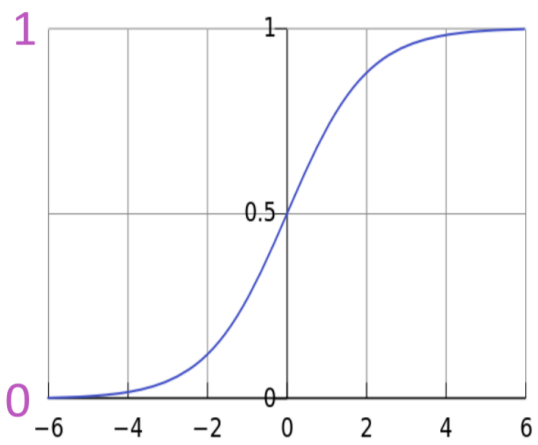
- Back-propagation (BP) algorithms work by determining the loss (or error) at the output and then propagating it back into the network. The weights are updated to minimize the error resulting from each neuron. Subsequently, the first step in minimizing the error is to determine the gradient (Derivatives) of each node w.r.t. the final output.

- This one round of forward and backpropagation iteration is known as one training iteration aka “Epoch“.

Activation Functions

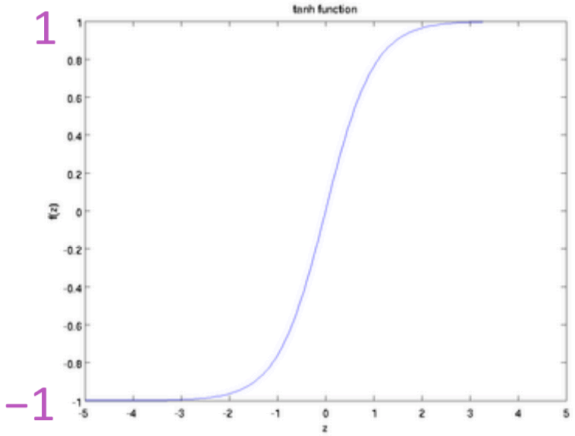
logistic ("sigmoid")

$$f(z) = \frac{1}{1 + \exp(-z)}.$$



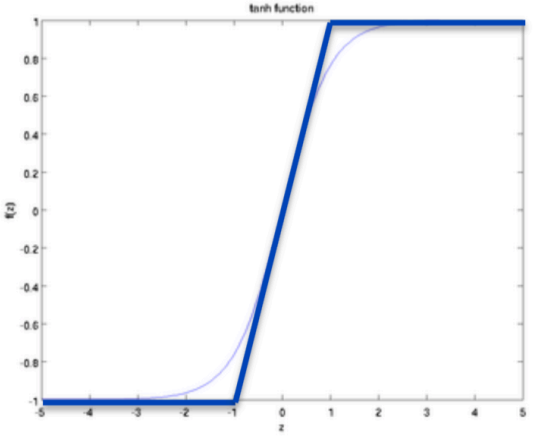
tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$



hard tanh

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$



tanh is just a rescaled and shifted sigmoid (2 × as steep, [−1,1]):

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

Both logistic and tanh are still used in particular uses, but are no longer the defaults for making deep networks

Activation Functions

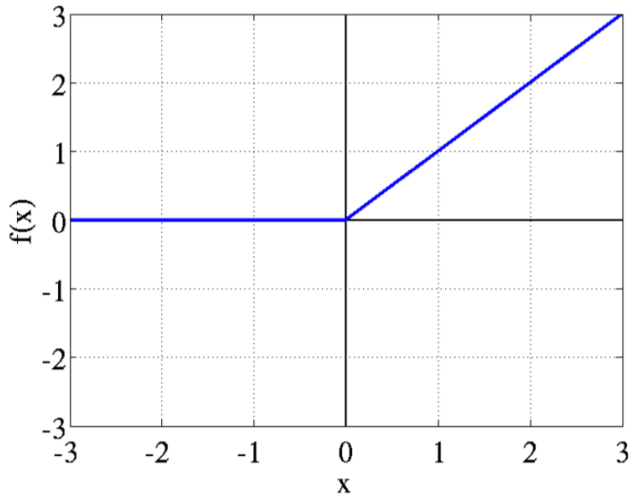
ReLU (rectified linear unit) hard tanh

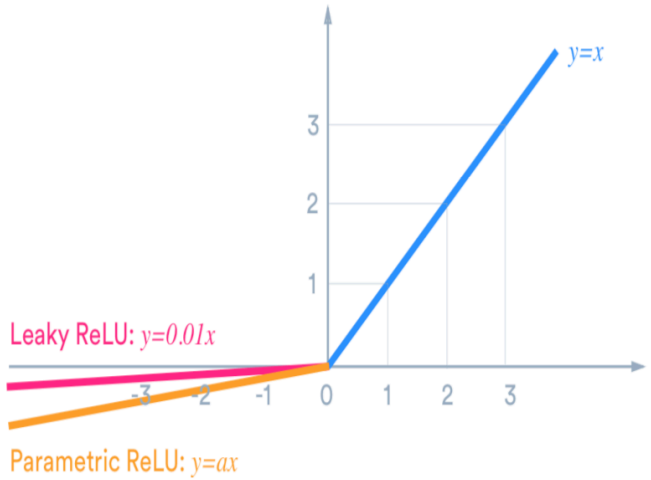
Leaky ReLU / Parametric ReLU

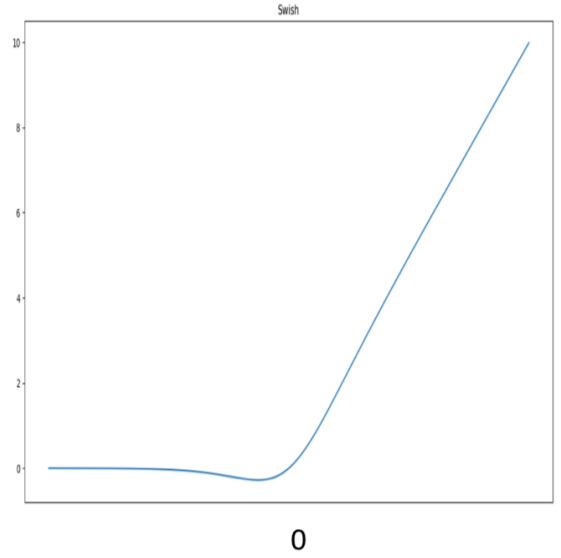
Swish

[Ramachandran, Zoph & Le 2017]

$$\text{rect}(z) = \max(z, 0)$$

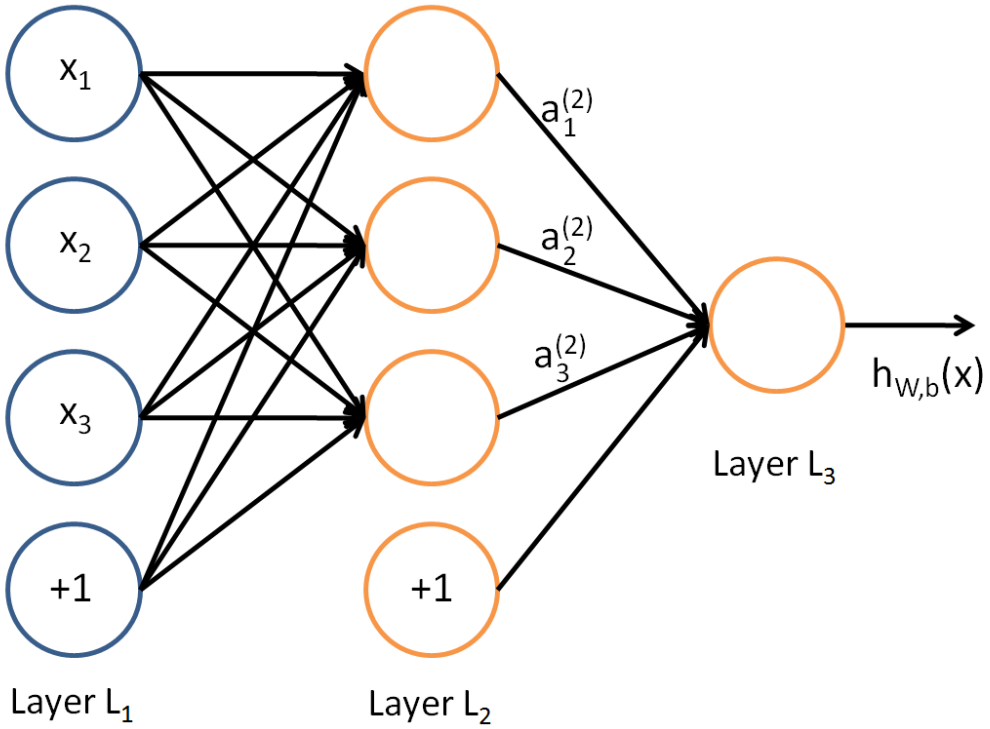






- For building a deep feed-forward network, the first thing you should try is ReLU — it trains quickly and performs well due to good gradient backflow

Why Not Feedforward Networks?



No relation

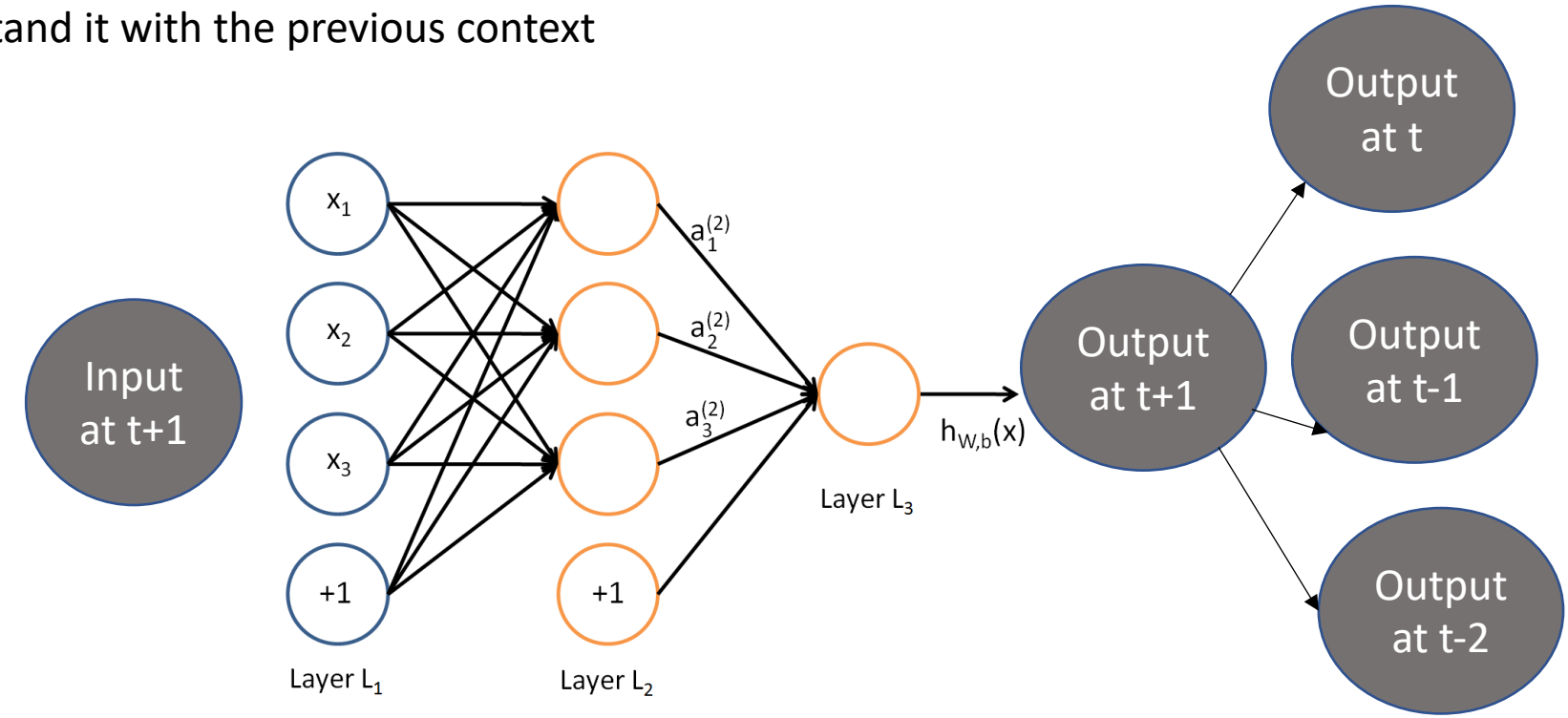
Output at t

Output at t-1

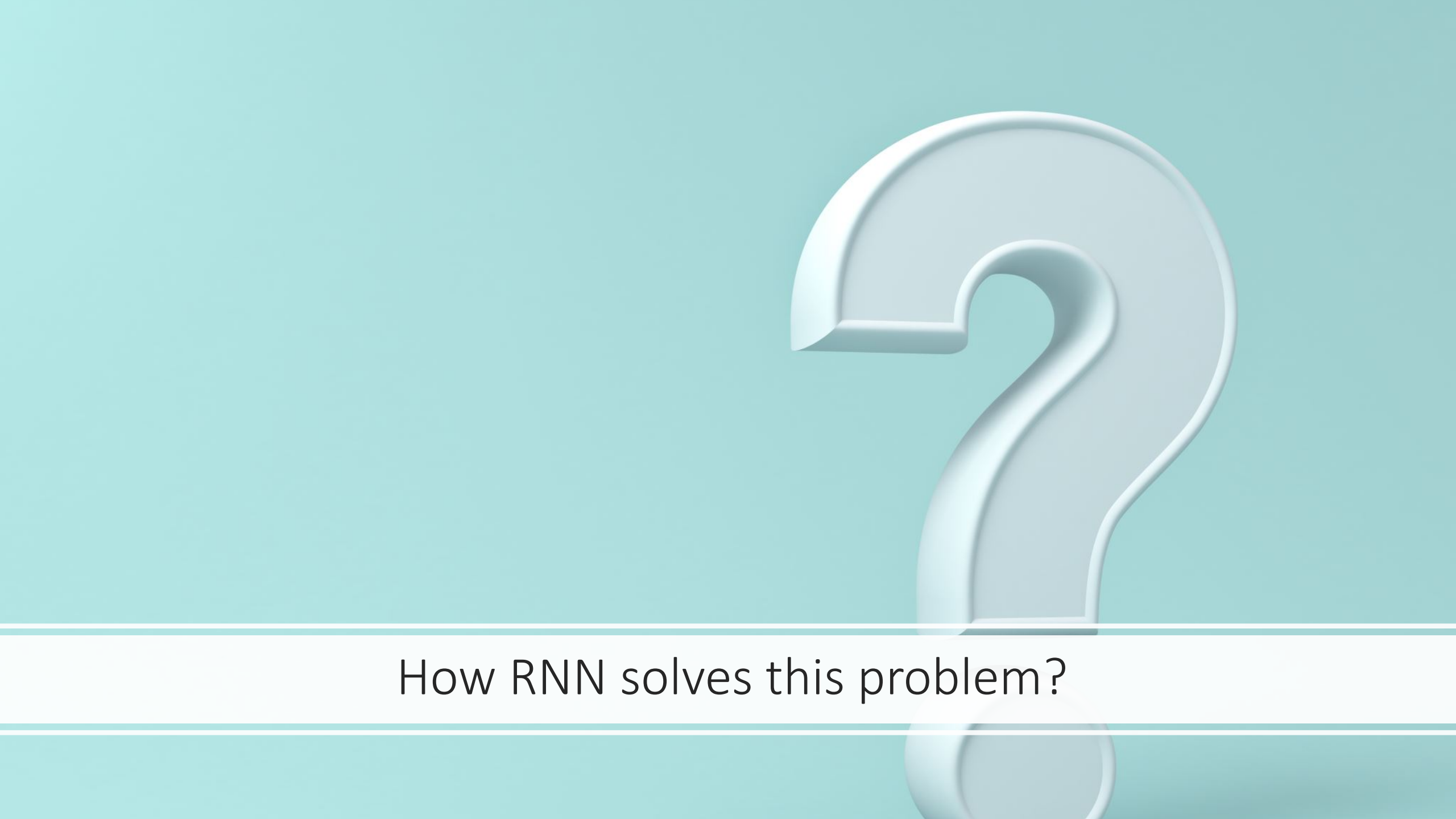
Seeing a photograph of dog at a certain step does not perceive the net to lead to elephant.

Why Not Feedforward Networks?

When we read a book, we understand it with the previous context

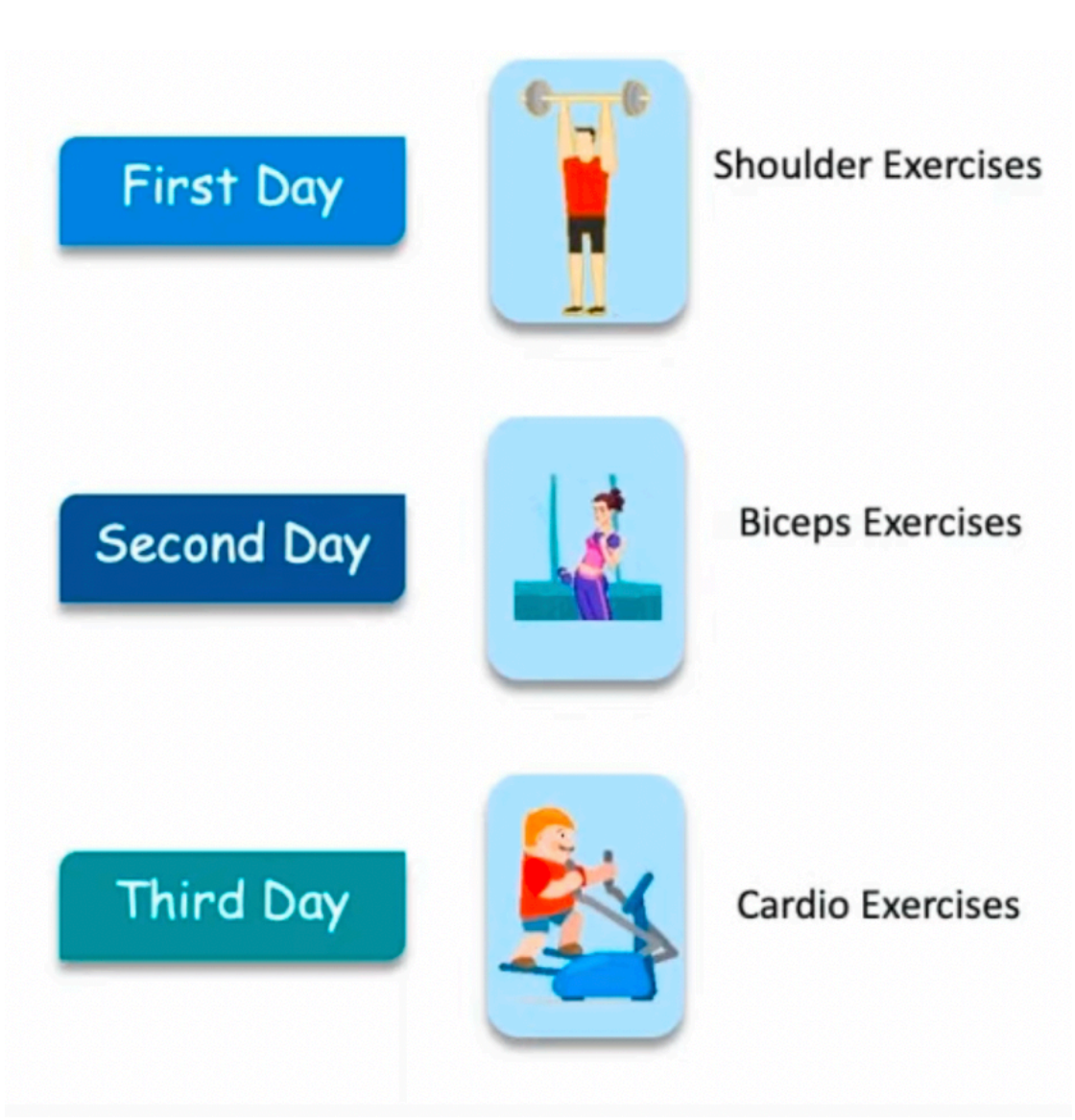


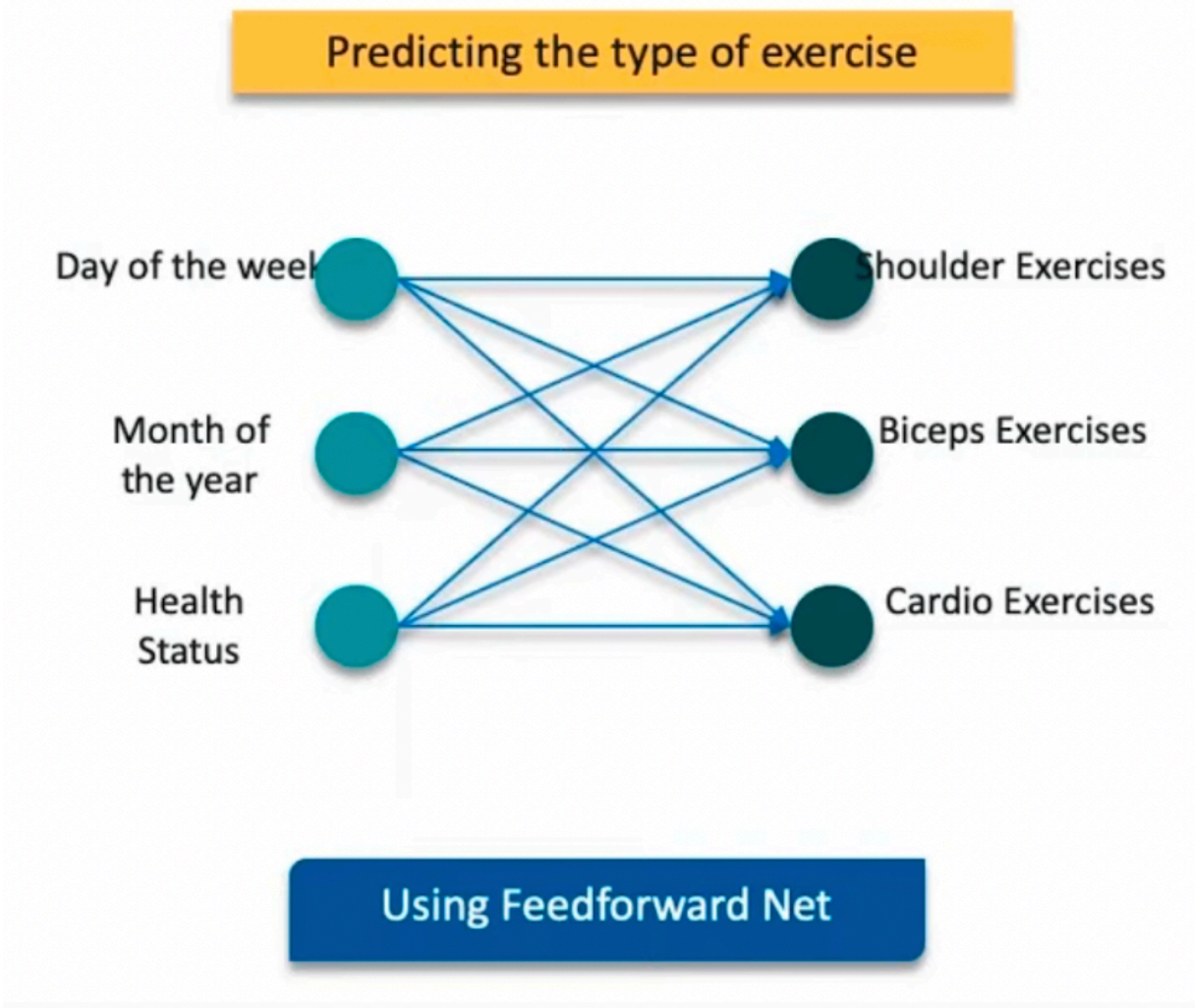
One can not predict the next word in sentence using feedforward nets



How RNN solves this problem?

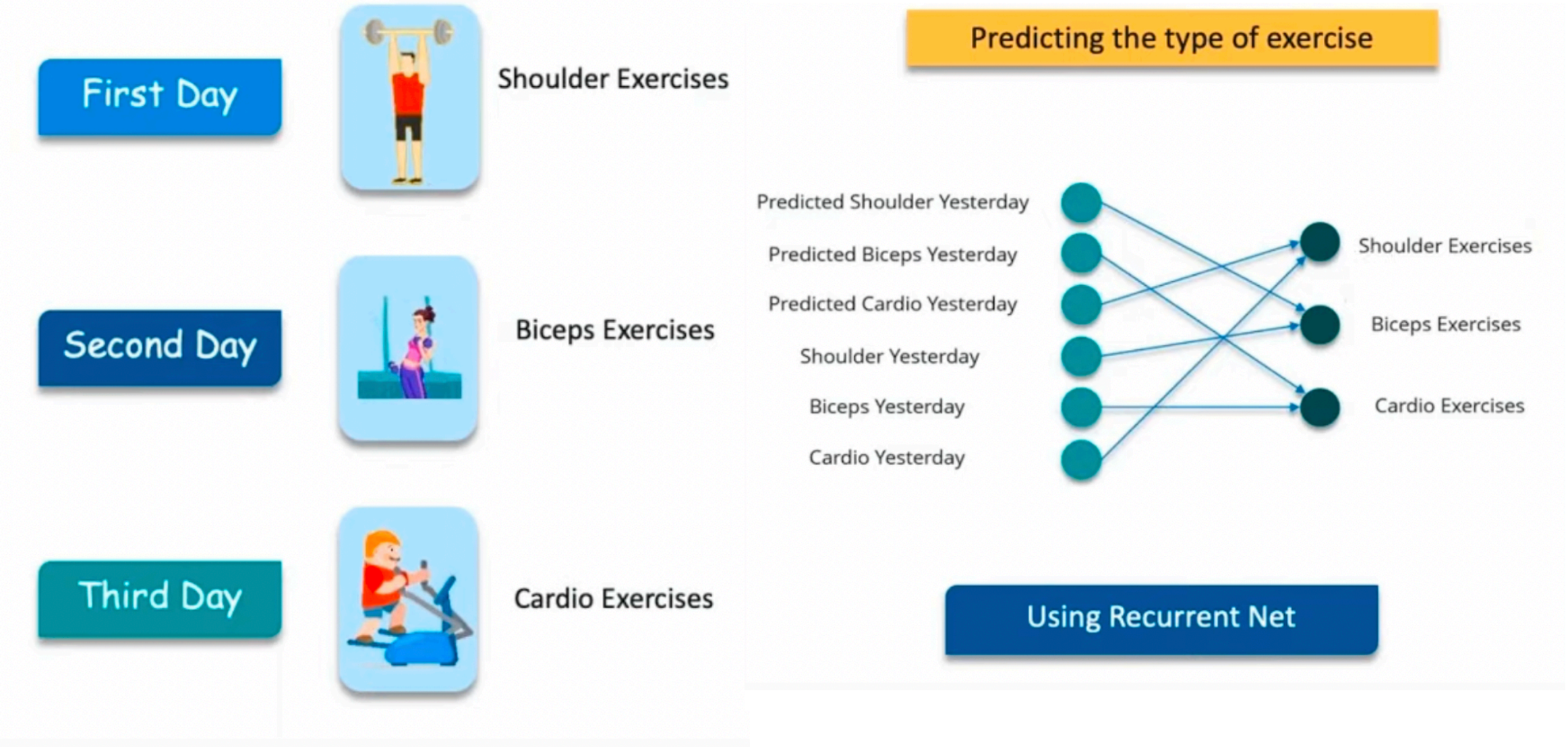
What is Recurrent Neural Networks





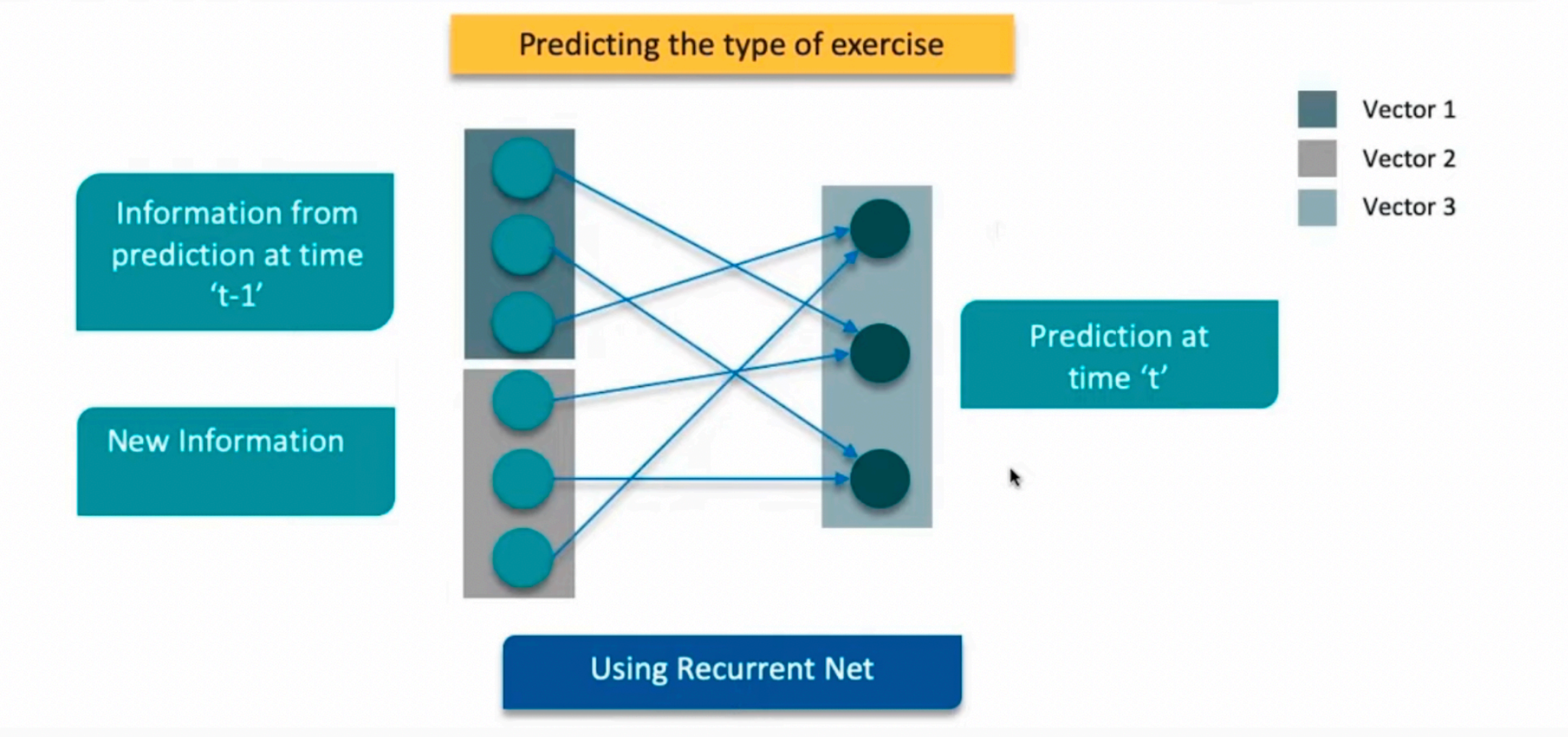
Reference: Edureka

What is Recurrent Neural Networks



Reference: Edureka

What is Recurrent Neural Networks



Reference: Edureka

What is Recurrent Neural Networks

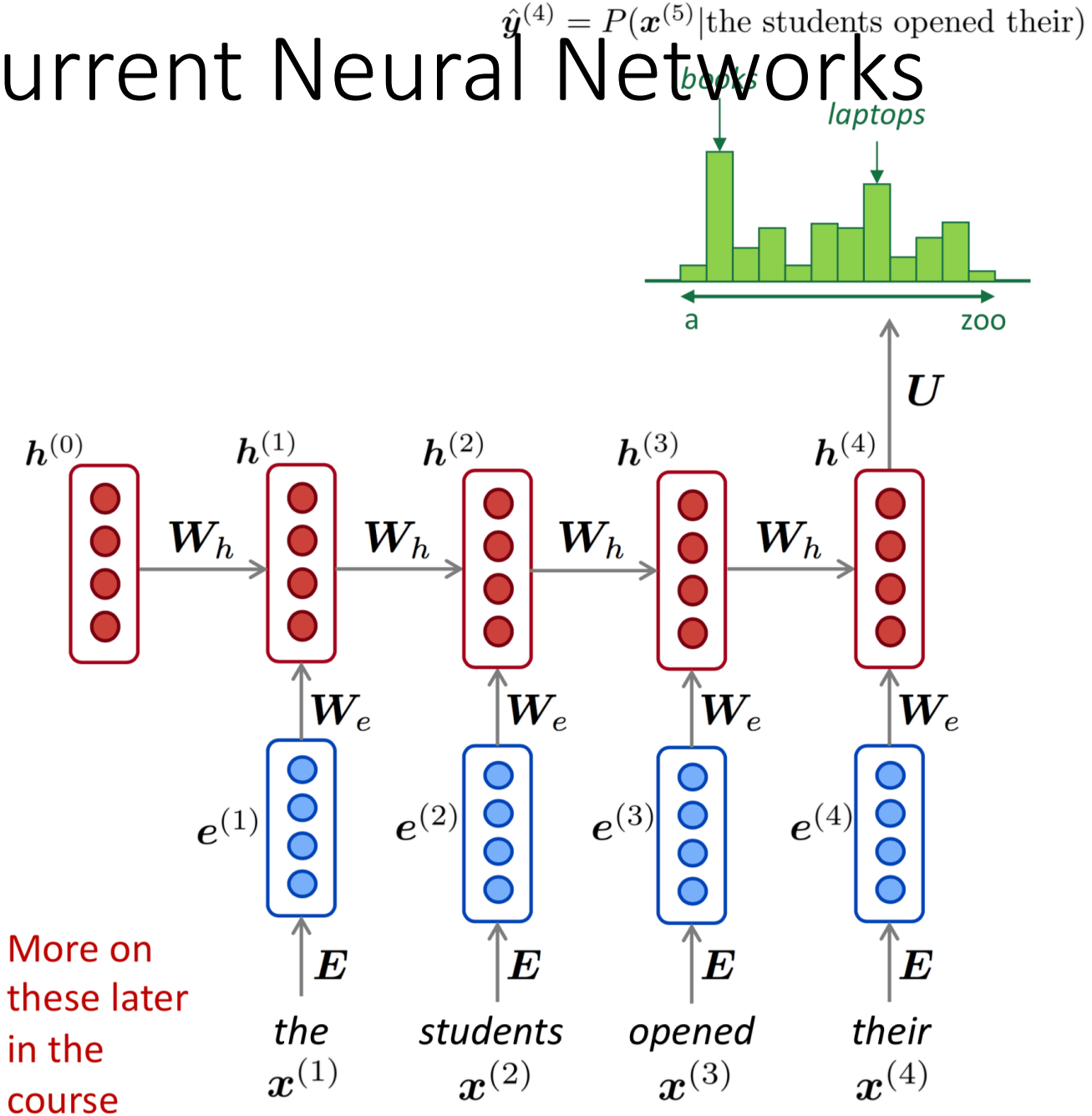
RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

More on these later in the course



Reference: Chris Manning

How to train a Recurrent Neural Network

- Recurrent Neural Networks uses a back propagation algorithm but it is applied for every time step.

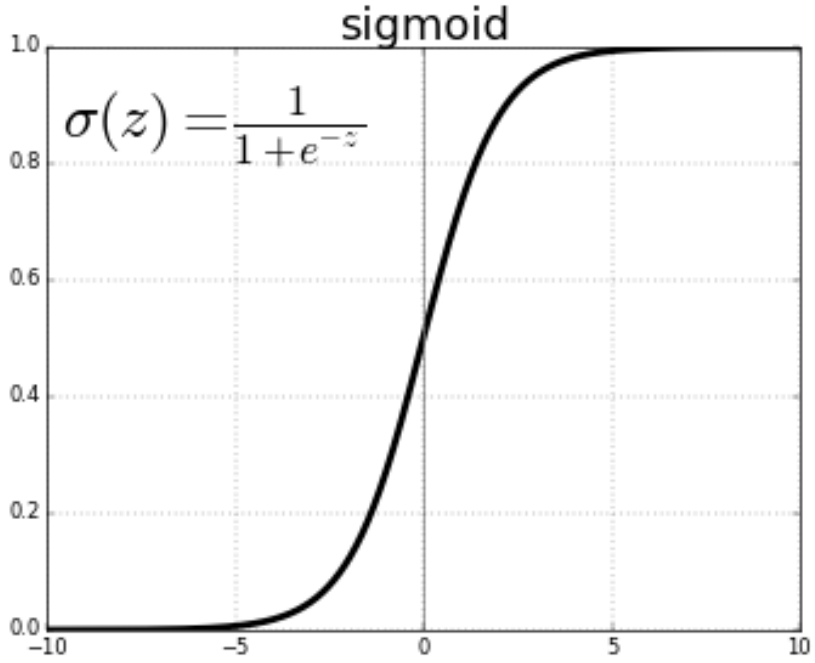
Issues with RNN!

Vanishing Gradient

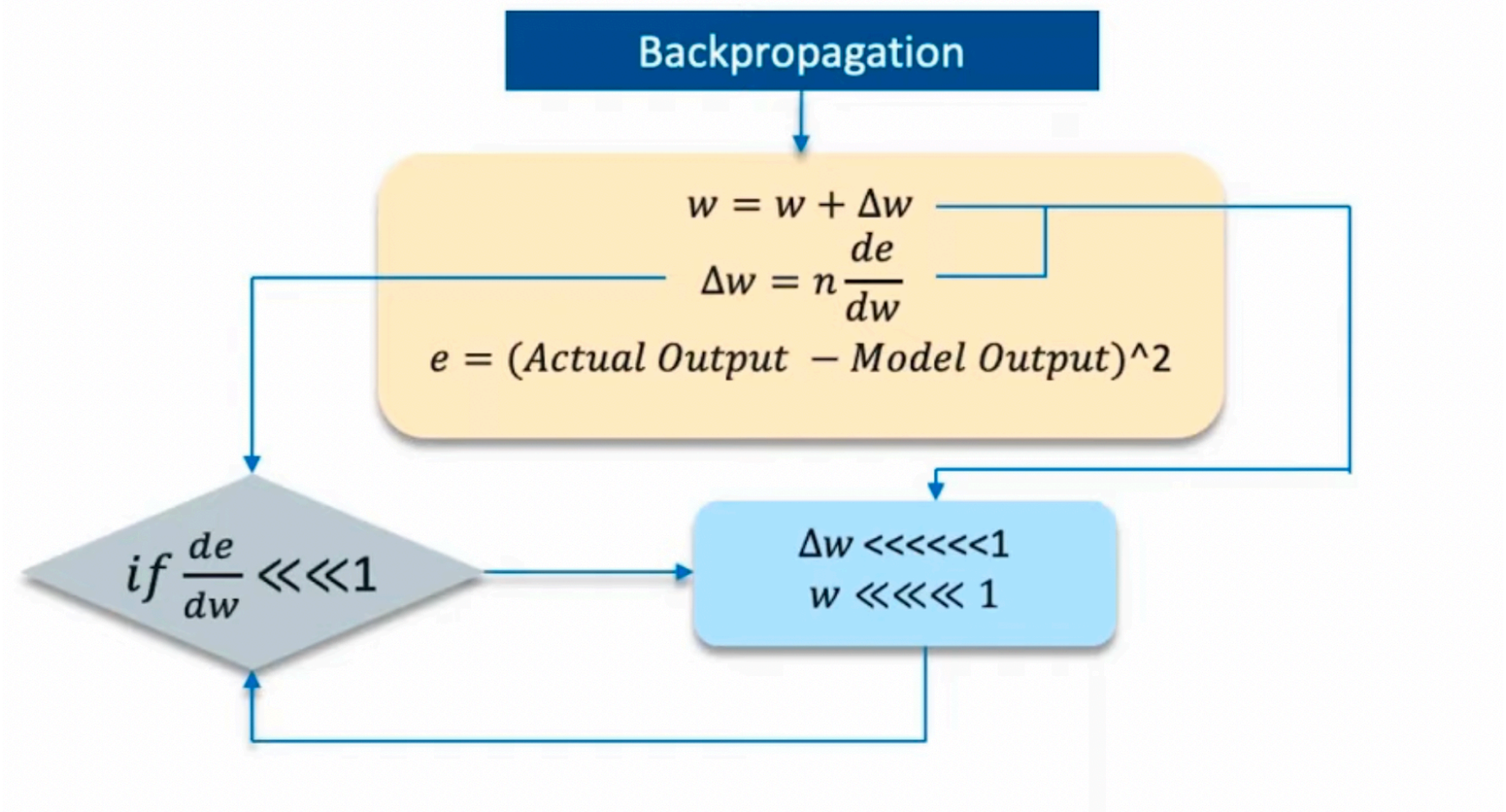
Exploding Gradient

Vanishing Gradient

- As can be seen from the picture a sigmoid function squashes it’s input into a very small output range [0,1] and has very steep gradients.
- Thus, there remain large regions of input space, where even a large change produces a very small change in the output.
- This is referred to as the problem of **vanishing gradient**.
- This problem increases with an increase in the number of layers and thus stagnates the learning of a neural network at a certain level

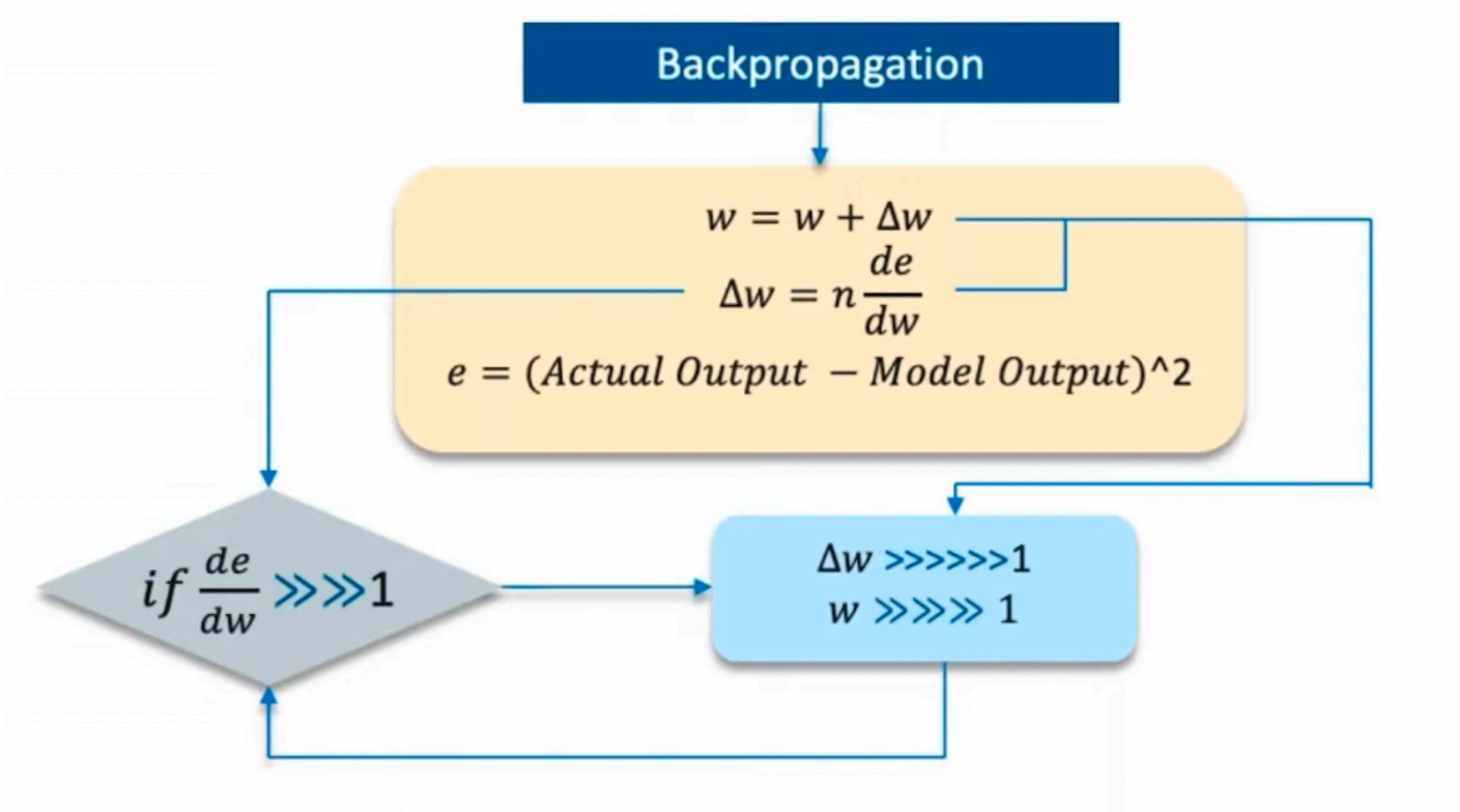


Vanishing Gradient



Reference: Edureka

Exploding Gradient



Reference: Edureka

How to overcome these challenges

Vanishing gradients

- *ReLU activation function*

  We can use activation functions like ReLU, which gives output one while calculating gradient

- *RMSprop*

  Clip the gradient when it goes higher than a threshold

- *LSTM, GRUs*

  Different network architectures that has been specially designed can be used to combat this problem

Exploding gradients

- *Truncated BTT*

  Instead of starting backpropagation at the last time stamp, we can choose a smaller time stamp like 10 (we will lose the temporal context after 10 time stamps)

- *Clip gradients at threshold*

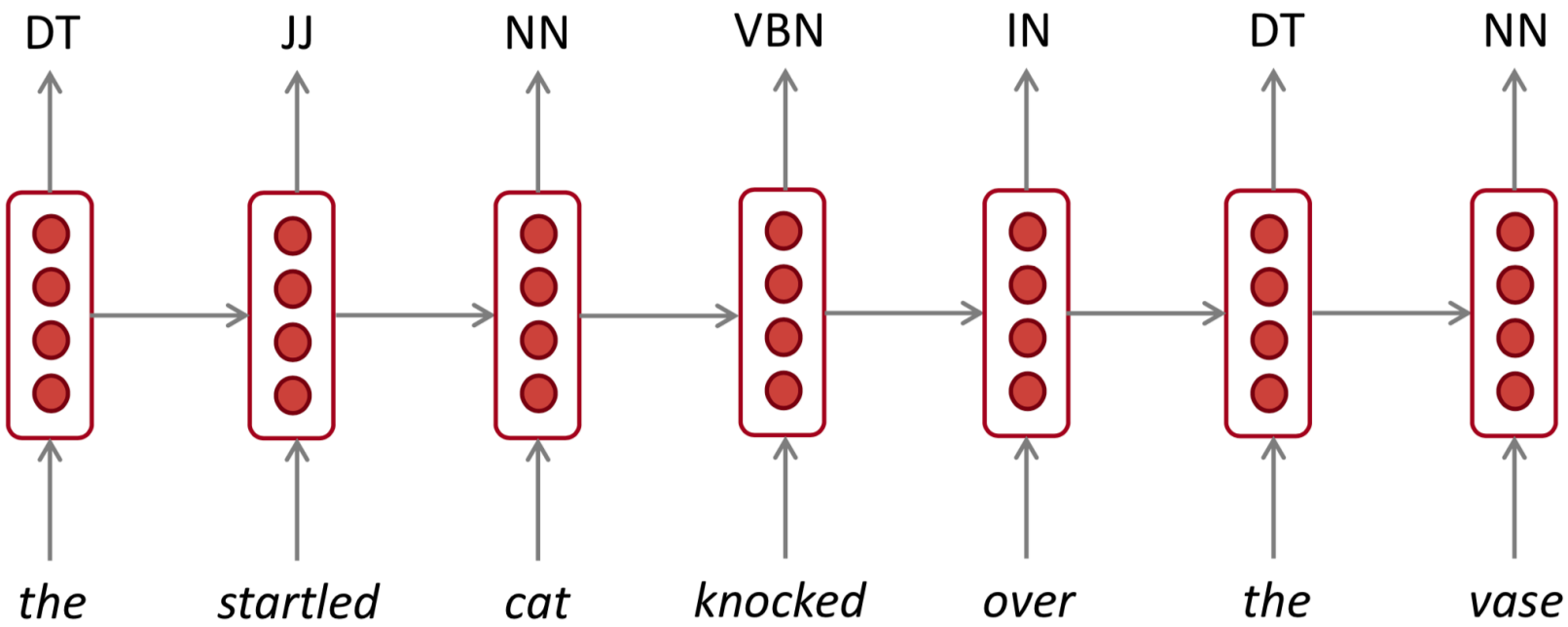
  Clip the gradient when it goes higher than a threshold

- *RMSprop to adjust learning rate*

Reference: Edureka

RNNs can be used for tagging

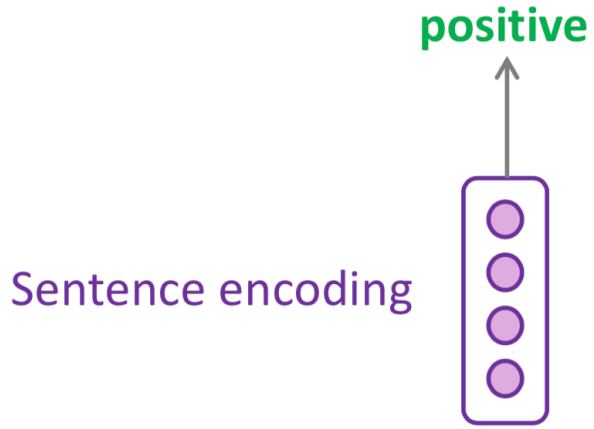
e.g. part-of-speech tagging, named entity recognition



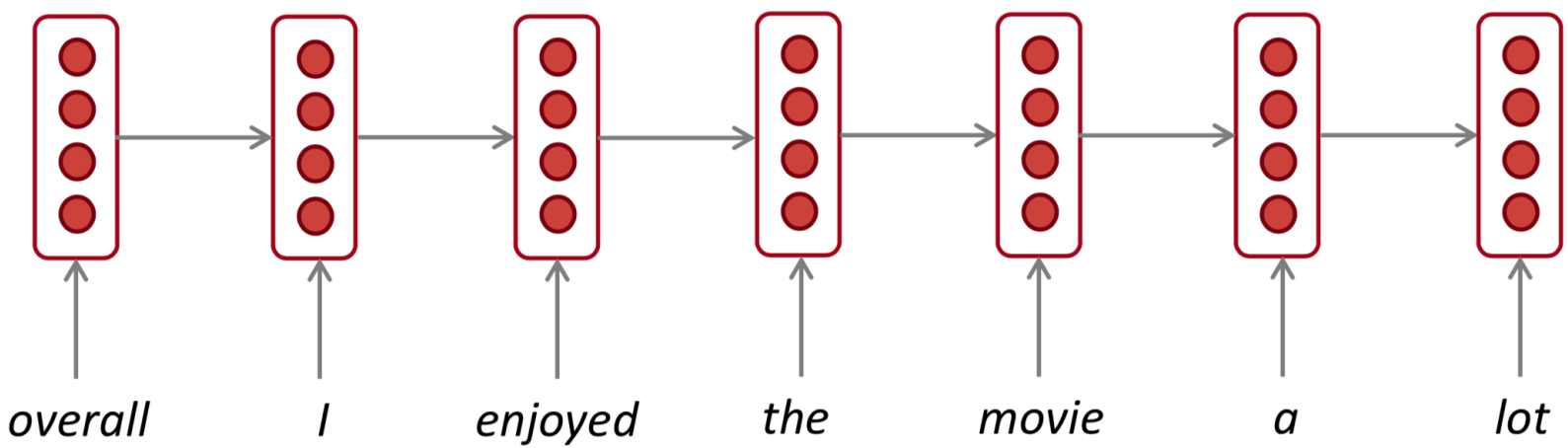
Reference: Chris Manning

RNNs can be used for sentence classification

e.g. sentiment classification



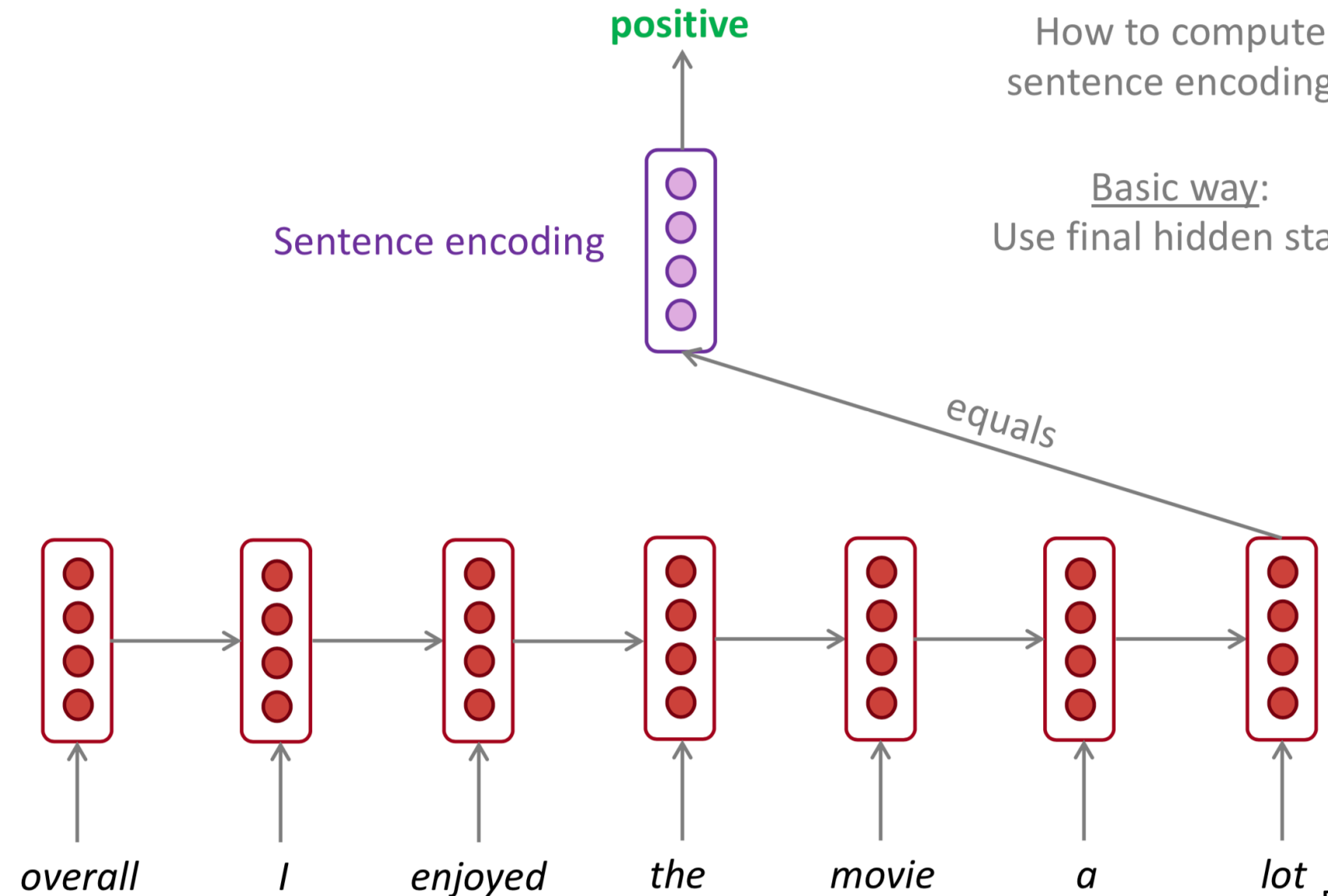
How to compute sentence encoding?



Reference: Chris Manning

RNNs can be used for sentence classification

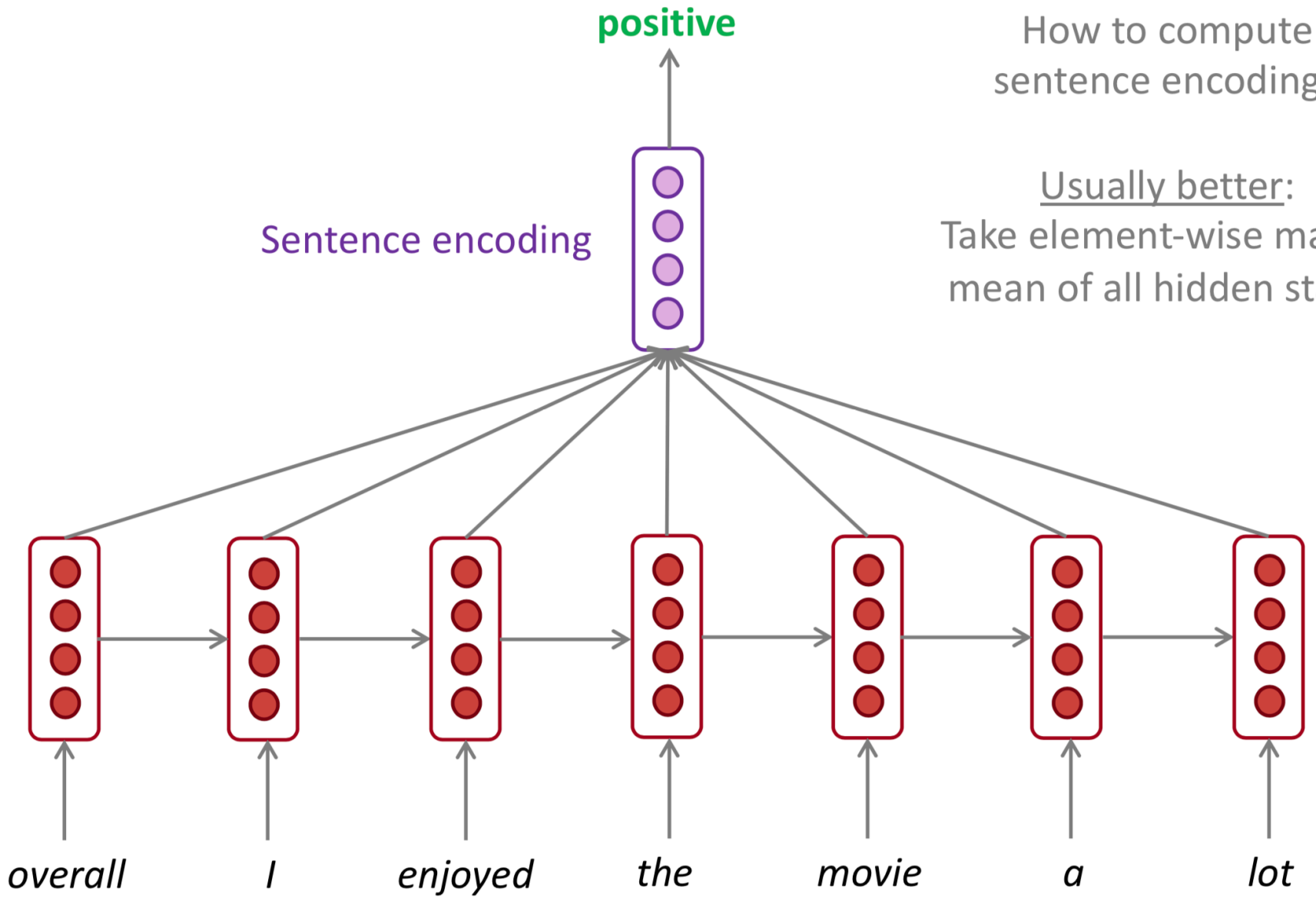
e.g. sentiment classification



Reference: Chris Manning

RNNs can be used for sentence classification

e.g. sentiment classification



How to compute sentence encoding?

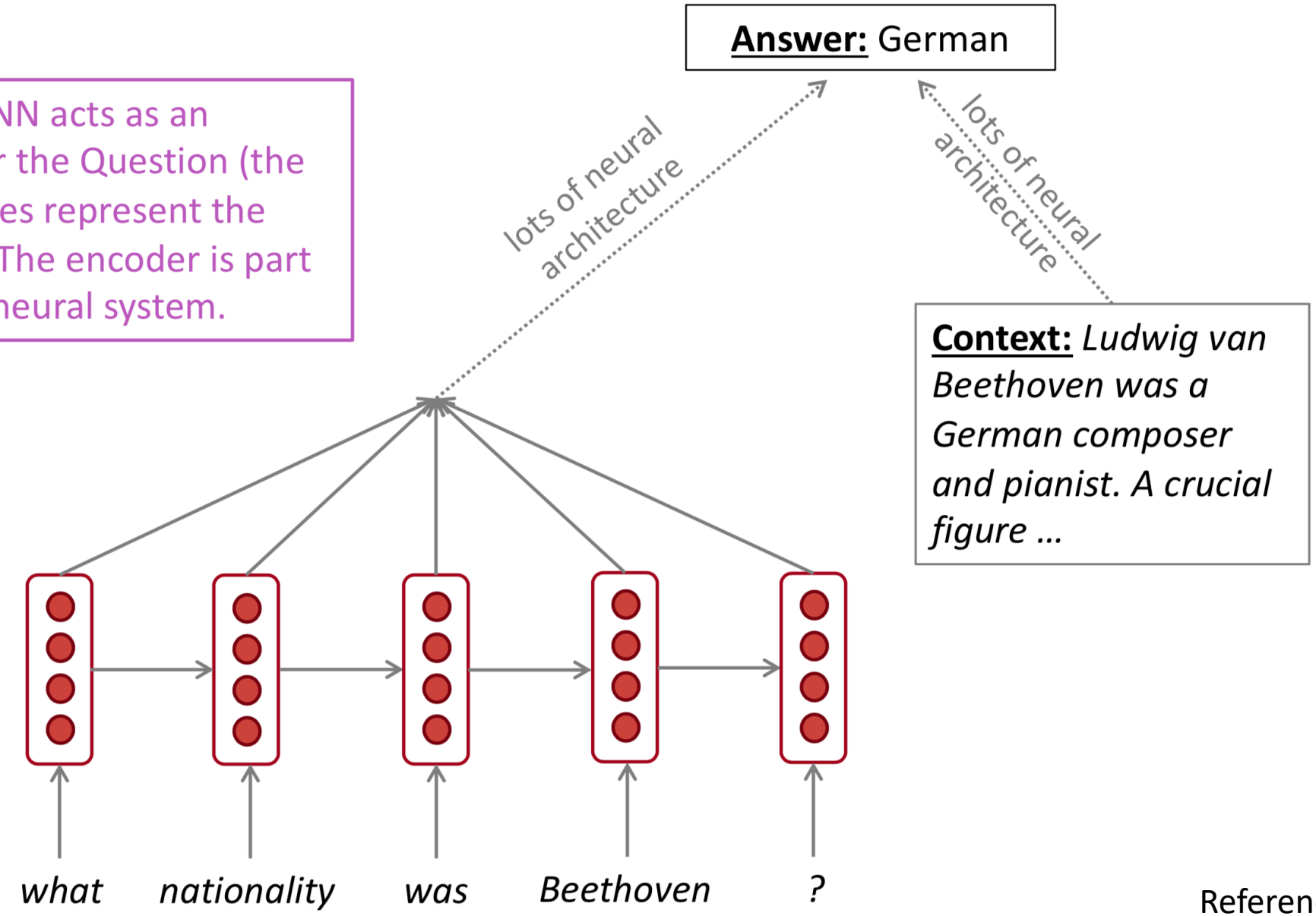
Usually better:
Take element-wise max or mean of all hidden states

Reference: Chris Manning

RNNs can be used as an encoder module

e.g. question answering, machine translation, *many other tasks!*

Here the RNN acts as an **encoder** for the Question (the hidden states represent the Question). The encoder is part of a larger neural system.

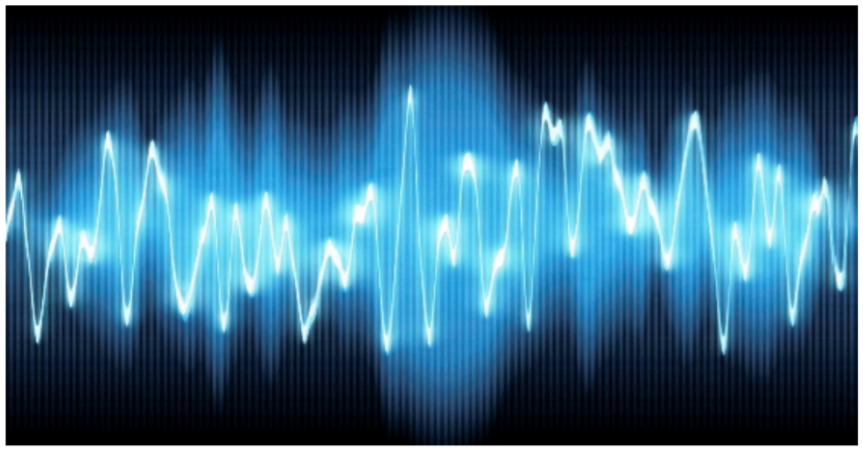


Reference: Chris Manning

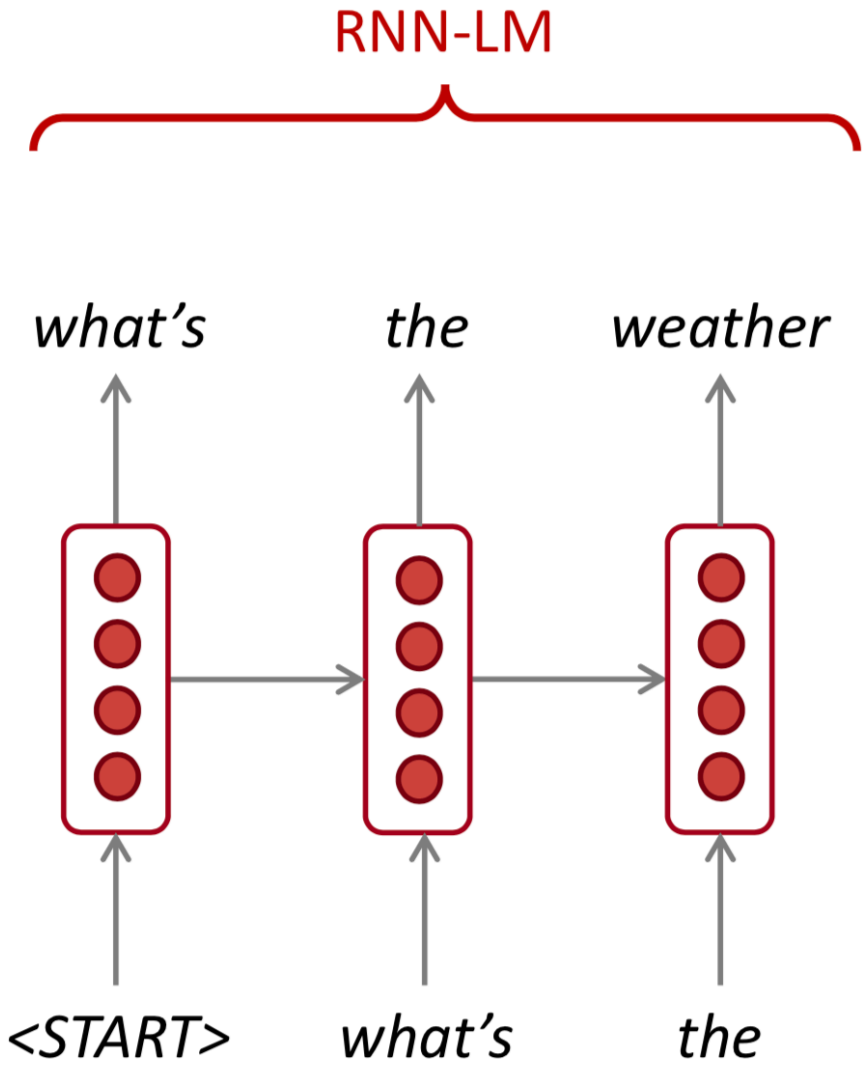
RNN-LMs can be used to generate text

e.g. speech recognition, machine translation, summarization

Input (audio)



conditioning



Reference: Chris Manning

A note on terminology

The RNN described in this lecture = simple/vanilla/Elman RNN 

In further slides, You will learn about other RNN flavors

like GRU  and LSTM  and multi-layer RNNs 

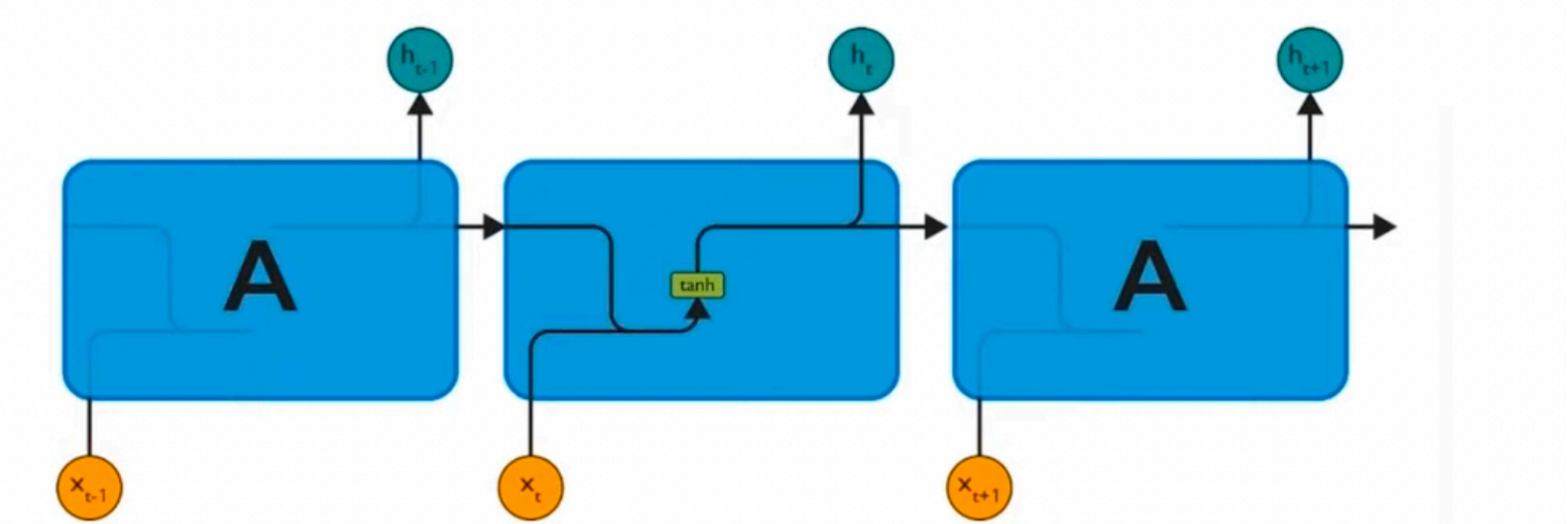
By the end of workshop: You will understand phrases like

"stacked bidirectional LSTM with residual connections and self-attention"

Reference: Chris Manning

Long Short Term Memory Networks

LSTM's are special kind of RNN's
These are capable of learning long-term dependencies



The repeating module in a standard RNN contains a single layer

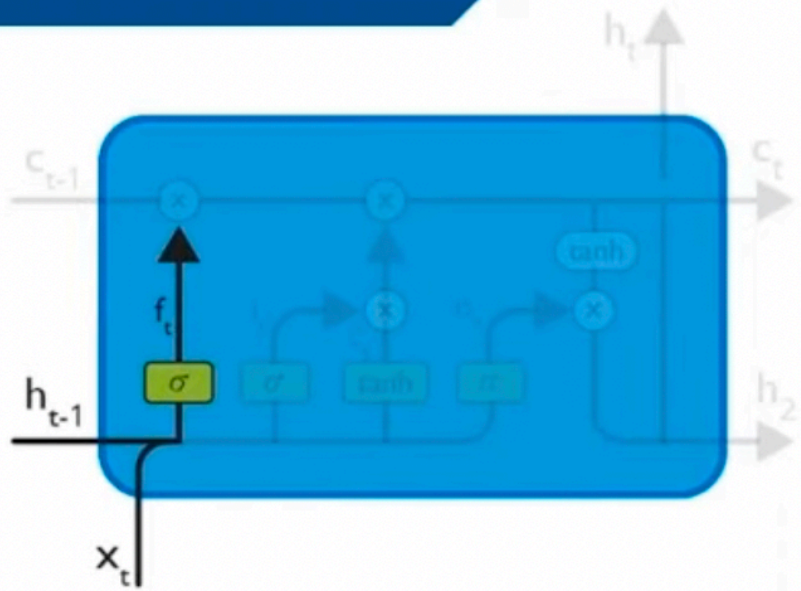
Reference: Edureka

Long Short Term Memory Networks

Step-1

The first step in the **LSTM** is to identify those information that are not required and will be thrown away from the cell state. This decision is made by a sigmoid layer called as forget gate layer.

$$w_f = Weight$$
$$h_{t-1} = Output\ from\ the\ previous\ time\ stamp$$
$$x_t = New\ input$$
$$b_f = Bias$$



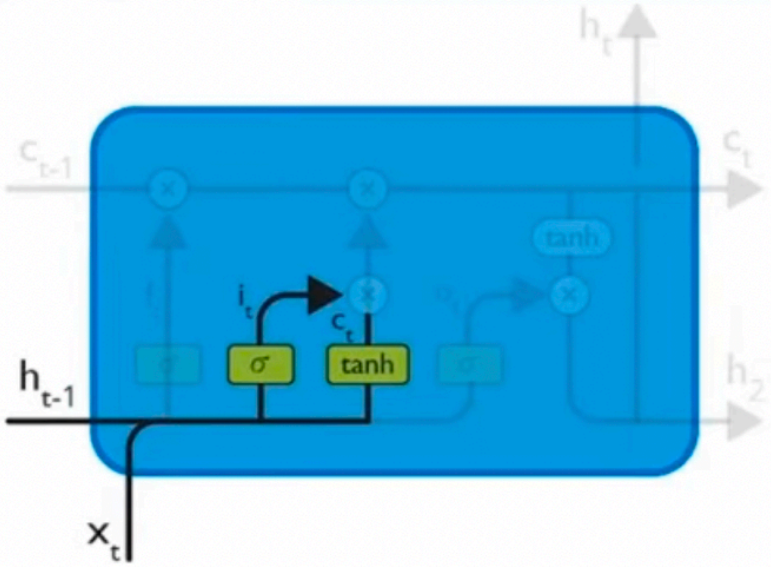
$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

Reference: Edureka

Long Short Term Memory Networks

Step-2

The next step is to decide, what new information we're going to store in the cell state. This whole process comprises of following steps. A **sigmoid layer** called the "input gate layer" decides which values will be updated. Next, a **tanh layer** creates a vector of new candidate values, that could be added to the state.



$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c)$$

In the next step, we'll combine these two to update the state.

Reference: Edureka

Hands-on RNN

Convolutional Neural Networks

- Convolutional Neural Networks (CNN) is a class of deep neural networks which is most commonly applied to analyzing visual imagery.
- It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers.
- Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions.
- To understand it in detail one needs to understand what convolution and pooling are. Both of these concepts are borrowed from the field of Computer Vision and are defined below.
- Convolution: Convolution operates on two signals (in 1D) or two images (in 2D): one can think of one as the “input” signal (or image), and the other (called the kernel) as a “filter” on the input image, producing an output image (so convolution takes two images as input and produces a third as output).
- In simple terms it takes in an input signal and applies a filter over it, essentially multiplies the input signal with the kernel to get the modified signal. Mathematically, a convolution of two functions f and g is defined as

$$(f * g)(i) = \sum_{j=1}^{m} g(j) \cdot f(i - j + m/2)$$

which, is nothing but dot product of the input function and a kernel function.

Convolutional Neural Networks

- In case of Image processing, it is easier to visualize a kernel as sliding over an entire image and thus changing the value of each pixel in the process.
- Pooling: Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.