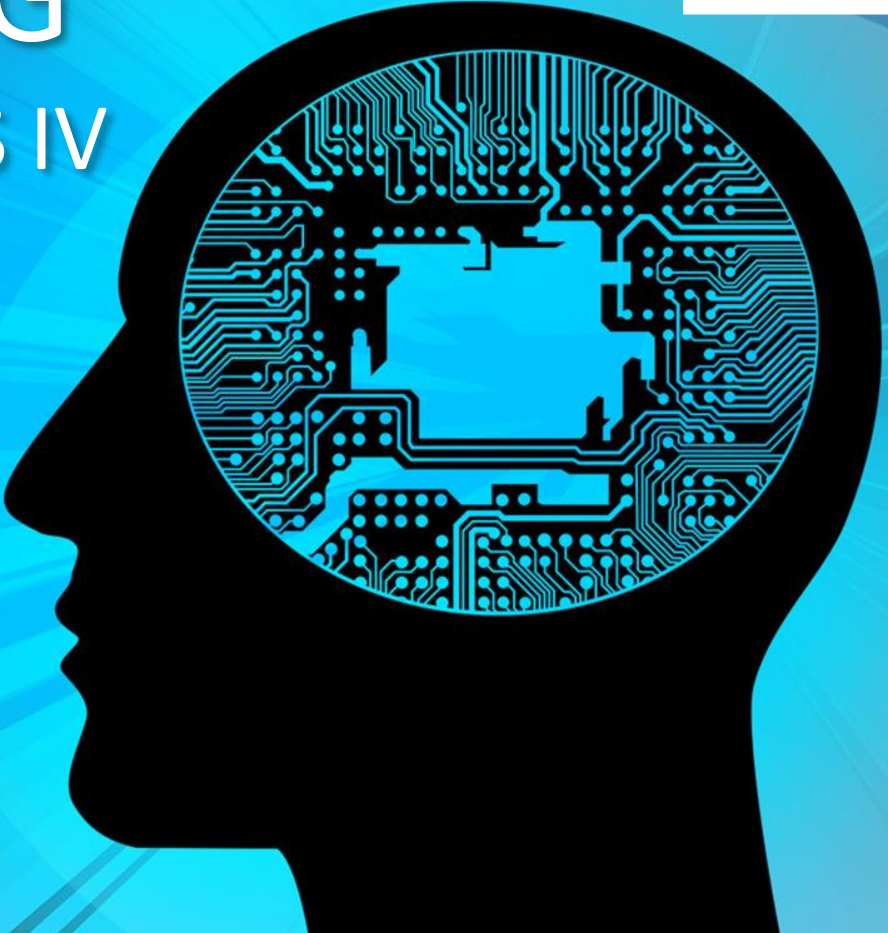
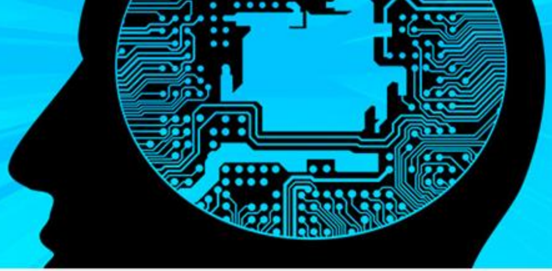
DEEP LEARNING
& NEURAL NETWORKS IV
RECURRENT NEURAL NETWORKS



Thakshila Dasun
BSc. Hons in Mechanical Engineering (Mechatronics)
CIMA, UK
Academy of Innovative Education

RECURRENT NEURAL NETWORKS

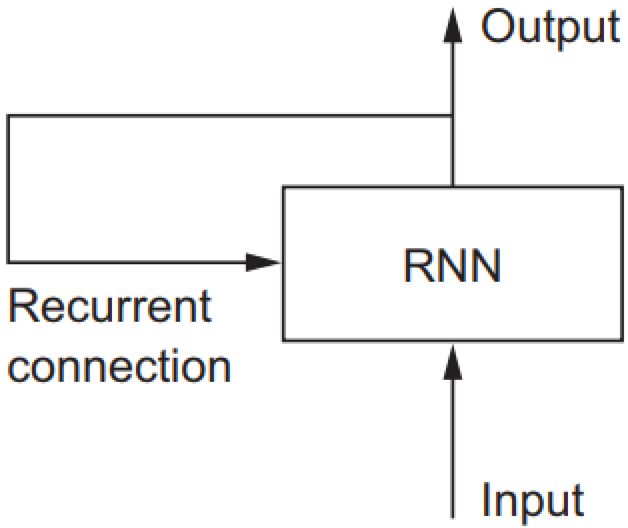


"Recurrent Networks are a **type** of **artificial neural network** designed to **recognize patterns** in sequences of data, such as **text,** genomes, handwriting, the spoken word, numerical times **series** data emanating from sensors, **stock markets** and **government agencies**."

-*Recurrent* means the output at the current time step becomes the input to the next time step. At each element of the sequence, the model considers not just the current input, but what it remembers about the preceding elements-

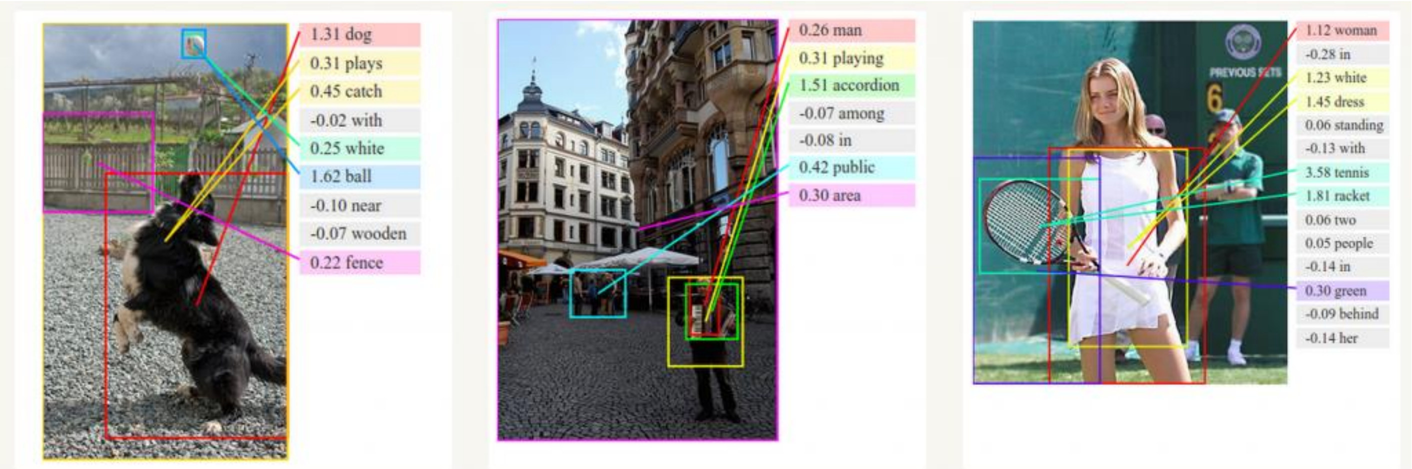
RECURRENT NEURAL NETWORKS

- The logic behind a RNN is to consider the sequence of the input. For us to predict the next word in the sentence we need to remember what word appeared in the previous time step.
- These neural networks are called Recurrent because this step is carried out for every input.
- As these neural network consider the previous word during predicting, it acts like a memory storage unit which stores it for a short period of time.



APPLICATIONS

- **Image recognition and characterization**
  - Recurrent Neural Network along with a ConvNet work together to recognize an image and give a description about it if it is unnamed.
  - This combination of neural network works in a beautiful and it produces fascinating results. Here is a visual description about how it goes on doing this, the combined model even aligns the generated words with features found in the images.



APPLICATIONS

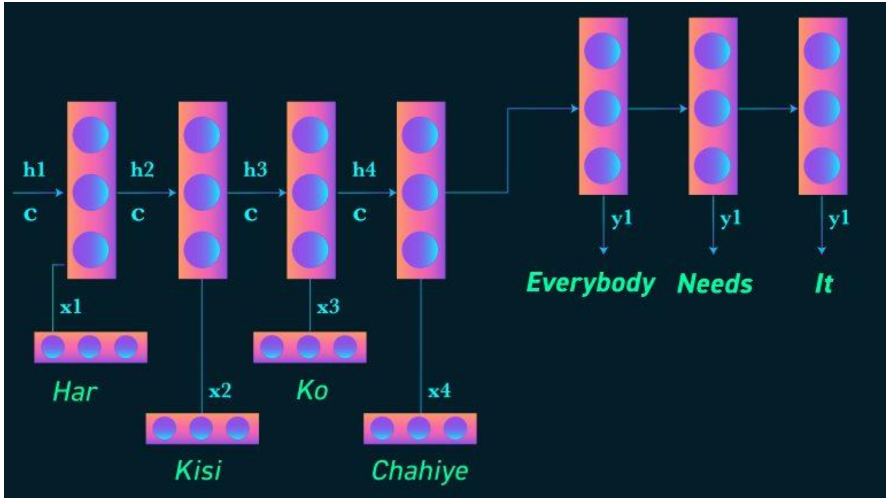
- **Language Modelling and Prediction**
  - In this method, the likelihood of a word in a sentence is considered.
  - The probability of the output of a particular time-step is used to sample the words in the next iteration(memory).
  - In Language Modelling, input is usually a sequence of words from the data and output will be a sequence of predicted word by the model.
  - While training we set ***xt+1 = ot***, the output of the previous time step will be the input of the present time step.

APPLICATIONS

- **Speech Recognition**
  - A set of inputs containing phoneme(acoustic signals) from an audio is used as an input.
  - This network will compute the phonemes and produce a phonetic segments with the likelihood of output.

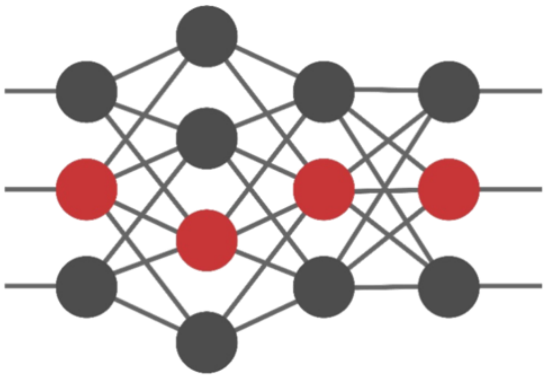
APPLICATIONS

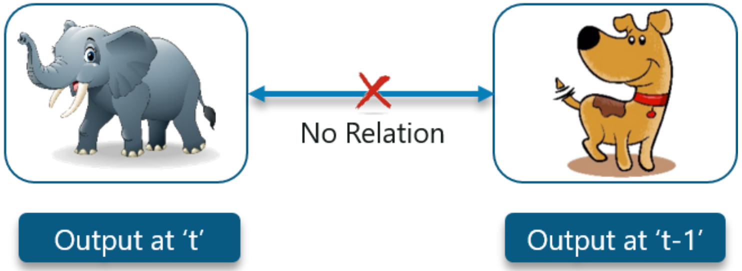
- **Machine Translation**
  - In Machine Translation, the input is will be the source language(e.g. Hindi) and the output will be in the target language(e.g. English).
  - The main difference between Machine Translation and Language modelling is that the output starts only after the complete input has been fed into the network.



Feedforward vs Recurrent Neural Networks

- Consider an **image classification** use-case where you have **trained** the neural network to **classify images** of various **animals.**

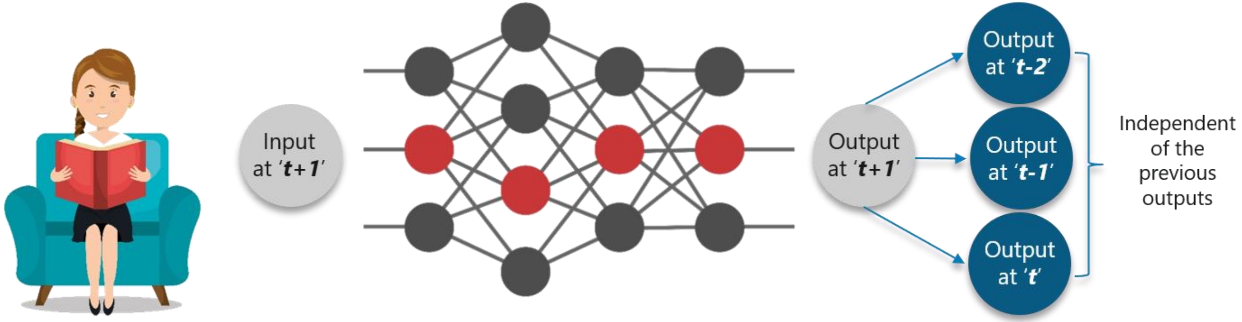




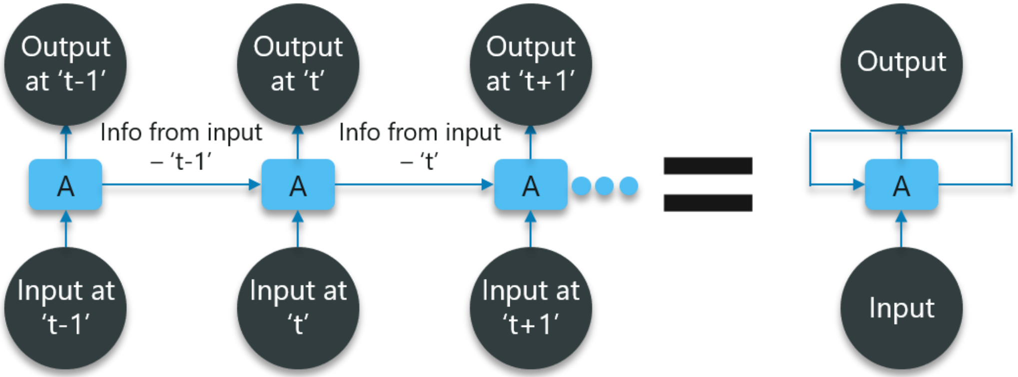
- Here, the **first output** being an **elephant** will have **no influence** of the **previous output** which was a **dog.**
- This means that output at time **'t'** is **independent** of output at time **'t-1'**.

Feedforward vs Recurrent Neural Networks

- The concept is similar to **reading** a **book.** With **every page** you move forward into, you need the **understanding** of the **previous pages** to make **complete** sense of the **information** ahead in most of the **cases.**



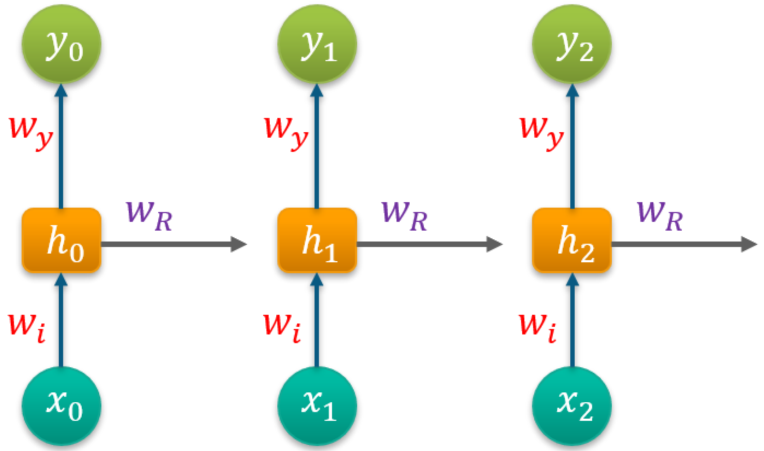
Feedforward vs Recurrent Neural Networks



- In the above diagram, we have certain inputs at 't-1' which is fed into the network. These inputs will lead to corresponding outputs at time 't-1' as well.
- At the **next timestamp,** information from the **previous** input 't-1' is provided **along** with the **input** at 't' to eventually **provide** the **output** at 't' as well.
- This process **repeats,** to ensure that the **latest inputs** are **aware** and can use the **information** from the previous **timestamp** is **obtained.**

Math behind Recurrent Neural Networks

- Consider **‘w’** to be the **weight matrix** and **‘b’** being the **bias**
- At time **t=0,** input is **‘x0’** and the task is to figure out what is **‘h0’.**
- Substituting **t=0** in the **equation** and obtaining the function **h(t) value.**
- Next, the value of **‘y0’** is found out using the **previously calculated values** when applied to the **new formula.**



$$h^{(t)} = g_h\left(w_i x^{(t)} + w_R h^{(t-1)} + b_h\right)$$

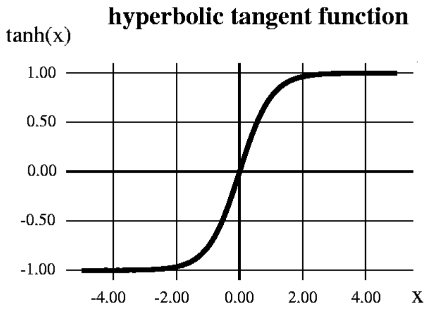
$$y^{(t)} = g_y\left(w_y h^{(t)} + b_y\right)$$

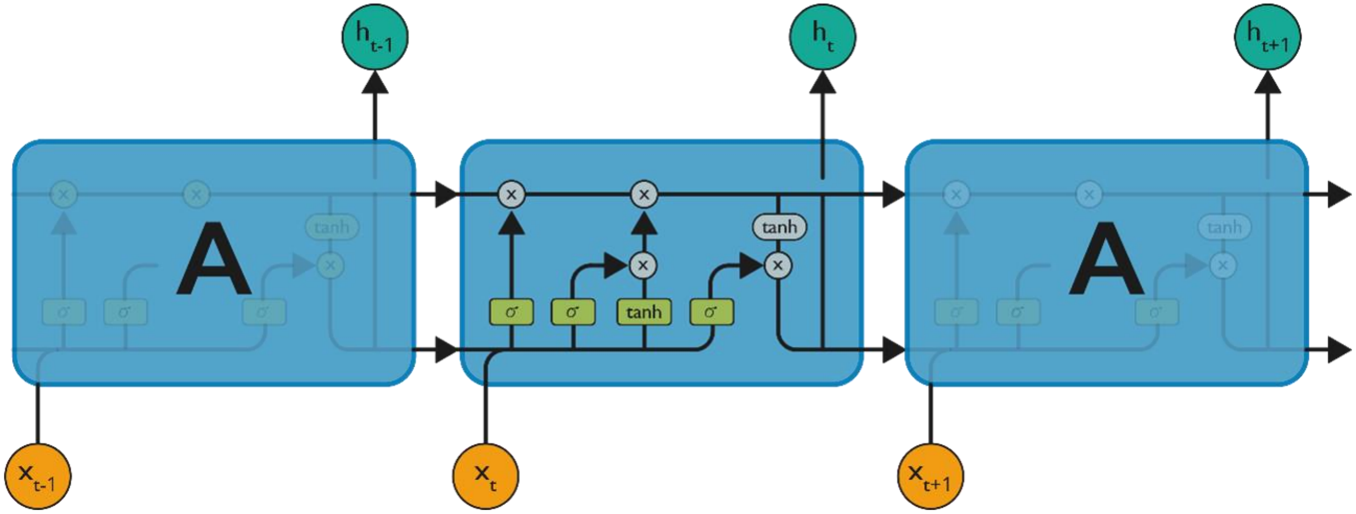
Long Short-Term Memory Networks (LSTMs)

- LSTM is a **special kind** of Recurrent Neural Networks which are **capable** of **learning long-term dependencies.**
- Many times only **recent data** is needed in a model to **perform operations.** But there might be a**requirement** from a **data** which was **obtained** in the **past.**
- Consider a **language model** trying to predict the **next word** based on the previous ones. If we are trying to **predict** the **last word** in the sentence say **“The clouds are in the sky”**.
- The context here was **pretty simple** and the last word ends up being **sky** all the time. In such cases, the gap between the **past information** and the **current requirement** can be **bridged** really easily by using **Recurrent Neural Networks.**

Long Short-Term Memory Networks (LSTMs)

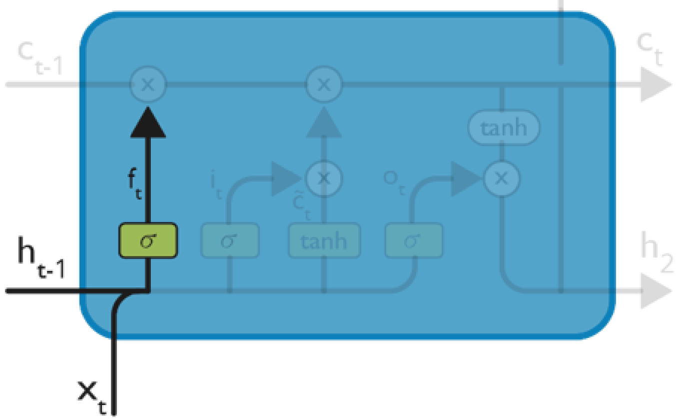
- LSTM have chain-like neural network layer. In a standard recurrent neural network, the repeating module consists of one single function.
- **tanh function** present in the layer. This function is a **squashing function (range of -1 to +1).**





Understanding LSTMs - **Step 1**

- The first step in the **LSTM** is to **identify** that information which is **not required** and will be **thrown away** from the **cell state.**
- This decision is made by a **sigmoid layer** called as **forget gate layer.**
- The **calculation** is **done** by considering the **new input** and the **previous timestamp** which eventually **leads** to the **output** of a number **between 0 and 1** for **each** number in that **cell state.**
- As typical binary, **1** represents to **kee**p the cell state while **0** represents to **trash** it.

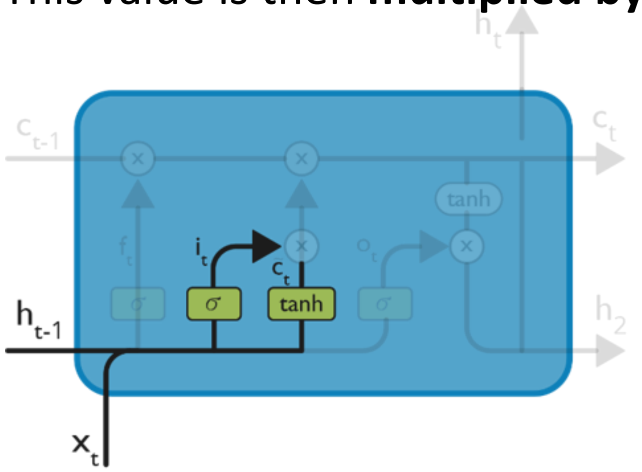


$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

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

Understanding LSTMs - **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: The **calculation** is **done** by considering
  - A **sigmoid layer** called the “input gate layer” decides **which values** will be **updated.**
  - The **tanh layer** creates a **vector** of **new candidate** values, that could be **added** to the state.
- The input from the **previous timestamp** and the new input are **passed** through a **sigmoid function**which gives the value **i(t).**
- This value is then **multiplied by c(t)** and then added to the **cell state.**

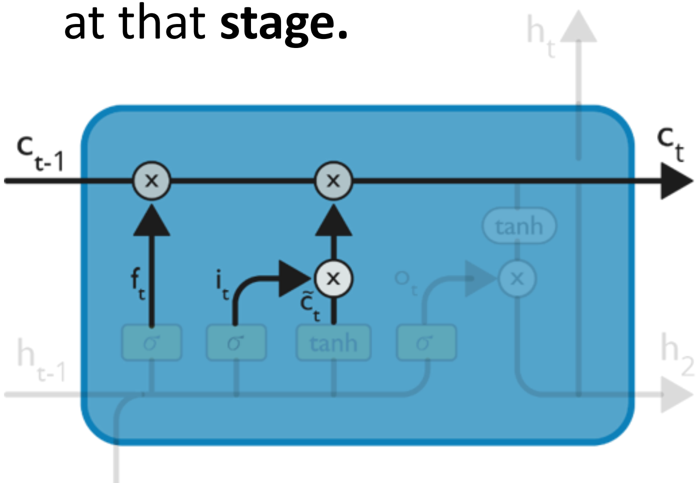


$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

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

Understanding LSTMs - **Step 3**

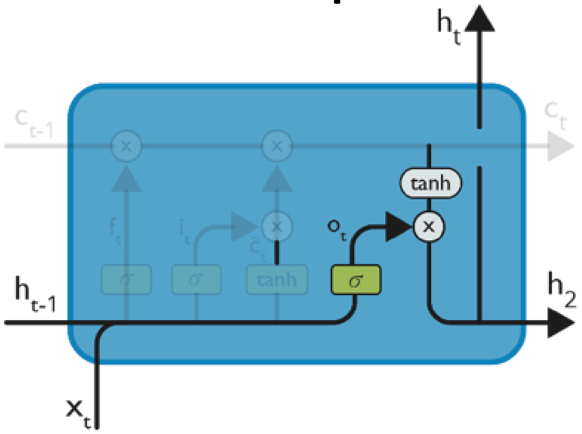
- Now, we will **update** the **old cell state Ct−1,** into the **new** cell state **Ct.**
- First, we **multiply** the **old** state **(Ct−1)** by f(t), **forgetting** the things we **decided** to **leave behind** earlier.
- Then, we **add i_t* c˜_t.** This is the **new candidate** values, **scaled** by how much we decided to **update each state** value.
- In the second step, we decided to do **make use** of the **data** which is only required at that **stage.**



$$c_t = f_t * c_{t-1} + i_t{}^* \, c\tilde{}_t$$

Understanding LSTMs - **Step 4**

- We will run a **sigmoid layer** which decides what **parts** of the **cell state** we're going to **output.**
- Then, we put the **cell state** through **tanh** (push the values to be between −1 and 1)
- Later, we **multiply** it by the **output** of the **sigmoid gate,** so that we only output the **parts** we decided to.
- The **calculation** in this step is pretty much **straightforward** which **eventually** leads to the **output.**
- However, the **output** consists of only the **outputs** there were decided to be **carry forwarded** in the **previous** steps and not all the **outputs** at once.



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

$$h_t = o_t * \tanh(c_t)$$

Summary

- In the first step, we found out what was needed to be dropped.
- The second step consisted of what new inputs are added to the network.
- The third step was to combine the previously obtained inputs to generate the new cell states.
- Lastly, we arrived at the output as per requirement.