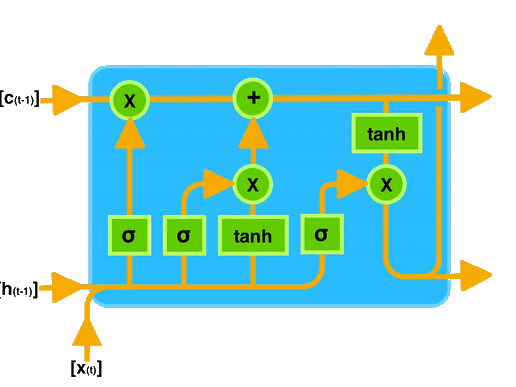
**Unlocking the Power of Long Short-Term Memory (LSTM) Networks**

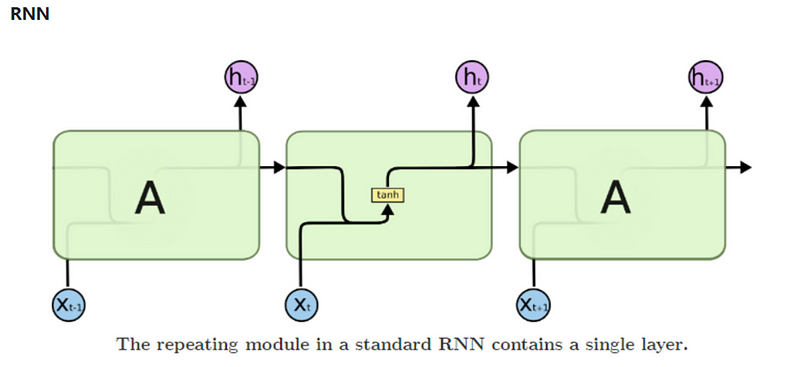


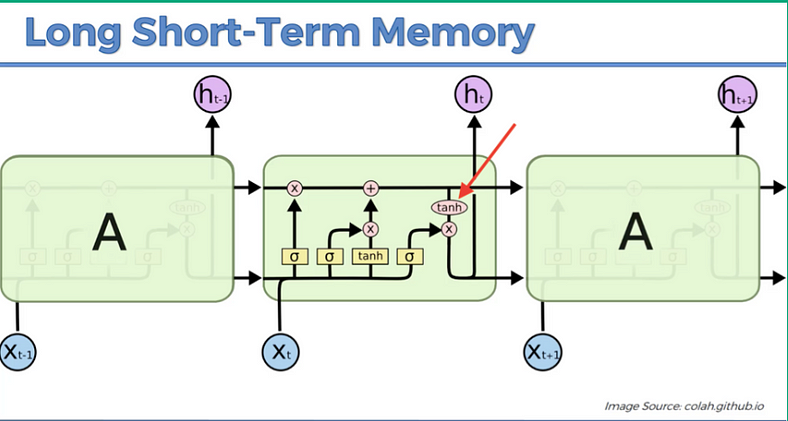
LSTM Architecture

Once upon a time, King Vikram fought bravely against King XYZ and won. But he passed away. Then his son, Vikram Junior, took over. He was even braver than his dad, but sadly, he also died in a battle with King XYZ. Then Vikram Junior’s son, Vikram Super Junior, became king. He wasn’t as strong as his dad and granddad, but he fought King XYZ. Even though it looked like he might lose, he used his smarts to beat King XYZ and get revenge for his family.

After reading the story or any other sequential data, our minds process information word by word, initially focusing on short-term context. For instance, as the story begins with an ancient tale involving King Vikram, our immediate attention is drawn to the events unfolding in the present. However, as the narrative progresses, our minds naturally transition to creating and maintaining long-term context. For example, upon encountering the mention of King Vikram’s demise, we adjust our long-term context accordingly. Subsequently, as new characters like Vikram Junior and Vikram Super Junior are introduced, our minds adapt by integrating them into the evolving long-term context. Each time a character’s role in the story concludes, we update our long-term context accordingly, akin to the way LSTM (Long Short-Term Memory) networks operate, dynamically adjusting their memory of past events as new information is processed.

In the case of RNNs, each line of information carries the burden of maintaining both short and long-term context. However, mathematically, it’s challenging to preserve both contexts simultaneously. As a result, the short-term context tends to overshadow the long-term one, akin to how we often remember the latest episode of a Netflix series more vividly than earlier ones. Recognizing this limitation, scientists proposed a solution: incorporating two pathways, one for short-term memory and another for long-term memory. This approach enables the model to prioritize important information, retaining it in long-term memory while discarding less relevant details over time.

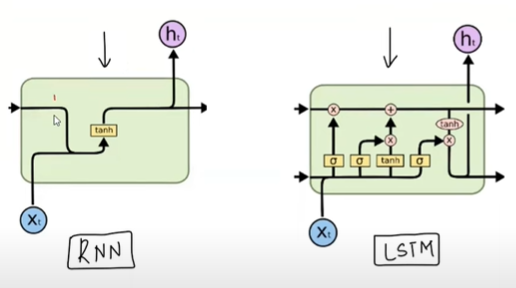




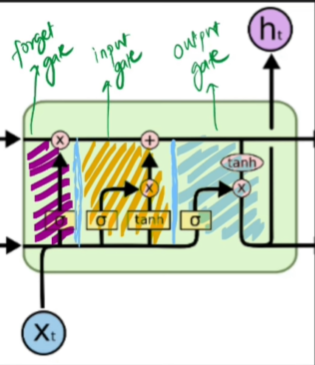
The LSTM architecture is more complicated as compared to RNNs because it has to manage both short-term and long-term context. This means it needs to handle communication between these two types of memory, adding complexity to the model.

**LSTM Architecture :**

The **architecture of LSTM includes three key components: the forget gate**, which decides what information to discard from the long-term memory, the **input gate**, which determines what new information to store in the long-term memory, and the **output gate** in LSTM determines what



RNN vs LSTM

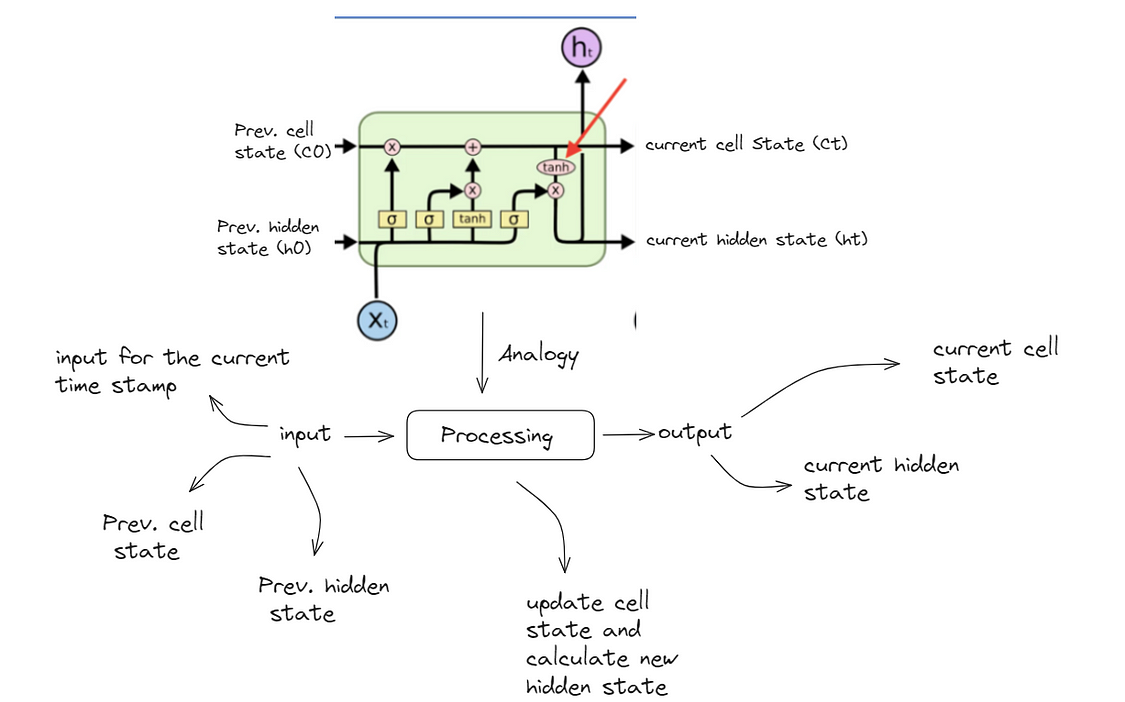


Components of LSTM

information from the long-term memory is used to produce the final output of the LSTM cell at a particular time step. It regulates the flow of information from the long-term memory to the current cell output, ensuring that only relevant information is considered in generating the output.

**LSTM Working :**

In the LSTM model, we can think of three main stages: input, processing, and output. At the input stage, we have three inputs: the input for the current state, previous cell state, and previous hidden state. During processing, the model updates the cell state(c0 -> ct) and calculates the new hidden state (h0 -> ht).



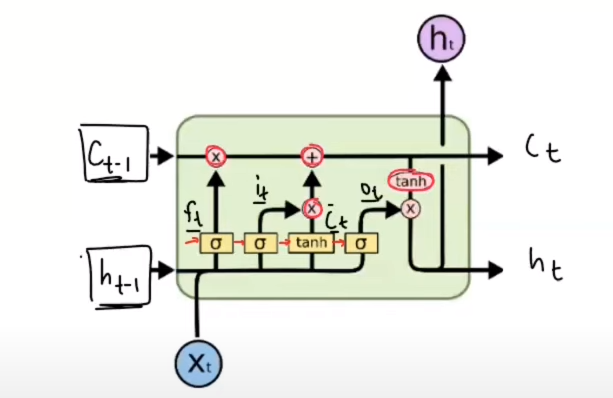
Finally, at the output stage, we have two outputs: the current cell state and the current hidden state. These outputs provide the information needed for further processing or decision-making in the model.

**Understanding the gates in LSTM :**

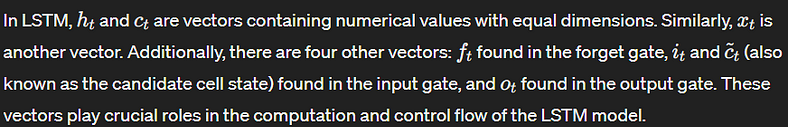
The architecture of LSTM includes three key components: the forget gate, which decides what information to discard from the long-term memory,



the input gate, which determines what new information to store in the long-term memory, and the output gate, which decides what information to use from the long-term memory to produce the output.



**What are ht, ct, xt, ft , it, c’t and ot ?**

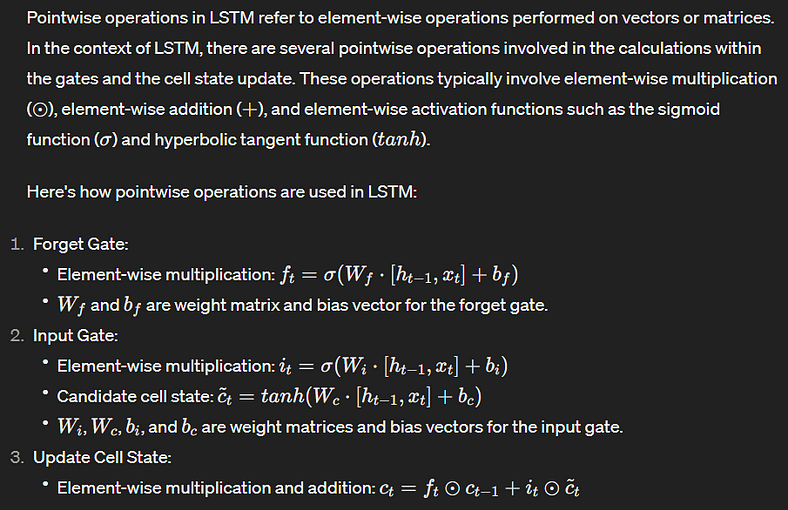


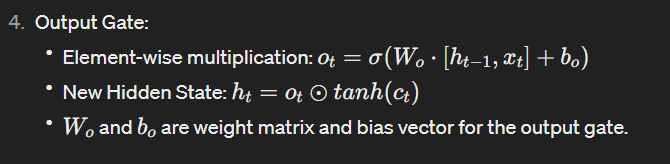
The yellow color boxes represent neural network layers with a specified number of nodes, which is indeed a hyperparameter determined by the user. In an LSTM cell, each gate (forget gate, input gate, and output gate) and the candidate cell state computation involve a neural network layer



with the same number of nodes, and they use either the sigmoid or the tanh activation function.

**Pointwise Operations :**





In the context of sequence models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, \*\*long-term dependencies\*\* and \*\*short-term dependencies\*\* refer to how a model handles and retains information over time.

1. \*\*Short-Term Dependencies\*\*:

- \*\*Short-term dependencies\*\* refer to relationships between elements in a sequence that are \*\*close together\*\* in time.

- These dependencies capture immediate context or information that is relevant for predicting the next element in the sequence.

Example:

Consider a sentence: \*"The cat is on the mat."\*

To predict the next word after "cat," the model only needs to look at the word "The" just before it, which is a \*\*short-term dependency\*\*.

- \*\*RNNs\*\* can handle short-term dependencies relatively well because the information from the recent past is still fresh in the model’s hidden state.

Challenges with Short-Term Dependencies:

In cases where information from only a few steps back is needed, traditional RNNs perform well. However, if relevant information is farther away, it becomes harder for the RNN to retain and use that information due to \*\*vanishing gradients\*\* or \*\*forgetfulness\*\*.

2. \*\*Long-Term Dependencies\*\*:

- \*\*Long-term dependencies\*\* refer to relationships between elements in a sequence that are \*\*far apart\*\* in time.

- These dependencies involve carrying important information over a longer range of time steps to make predictions or decisions.

Example:

Consider the sentence: \*"The cat I saw at the park yesterday was very friendly."\*

If you're predicting the word after "was," you'd need to know that "The cat" from earlier in the sentence is the subject, which is a \*\*long-term dependency\*\*.

- \*\*LSTM\*\* and \*\*GRU\*\* networks were designed to handle these long-term dependencies. They can remember relevant information from many time steps ago and apply it when needed, thanks to mechanisms like \*\*gates\*\* (forget, input, output gates) that control the flow of information.

#### Challenges with Long-Term Dependencies:

In traditional RNNs, when the gap between relevant information and its use becomes large (e.g., 100+ time steps), the model struggles to retain useful information from the distant past due to issues like vanishing gradients. LSTMs address this by maintaining a more stable \*\*cell state\*\* over time.

### Summary of Differences:

| Aspect | \*\*Short-Term Dependencies\*\* | \*\*Long-Term Dependencies\*\* |

|--------|-----------------------------|----------------------------|

| \*\*Definition\*\* | Relationships between elements in a sequence that are close together in time | Relationships between elements in a sequence that are far apart in time |

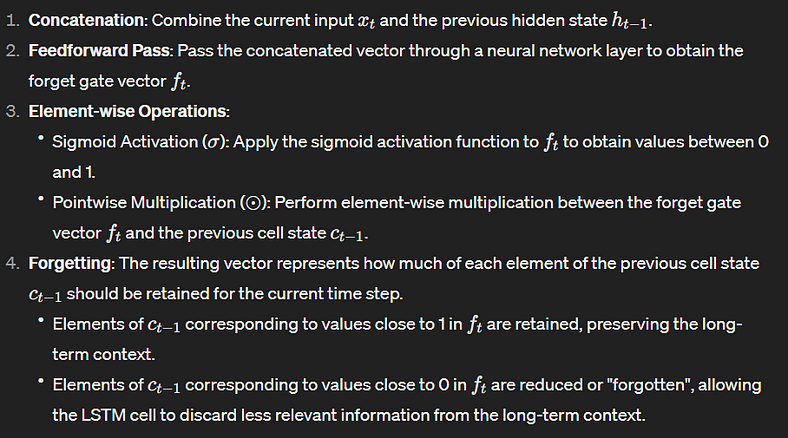
| \*\*Example\*\* | Predicting the next word in a sentence based on the previous few words | Remembering the subject of a sentence that was mentioned much earlier to predict a verb |

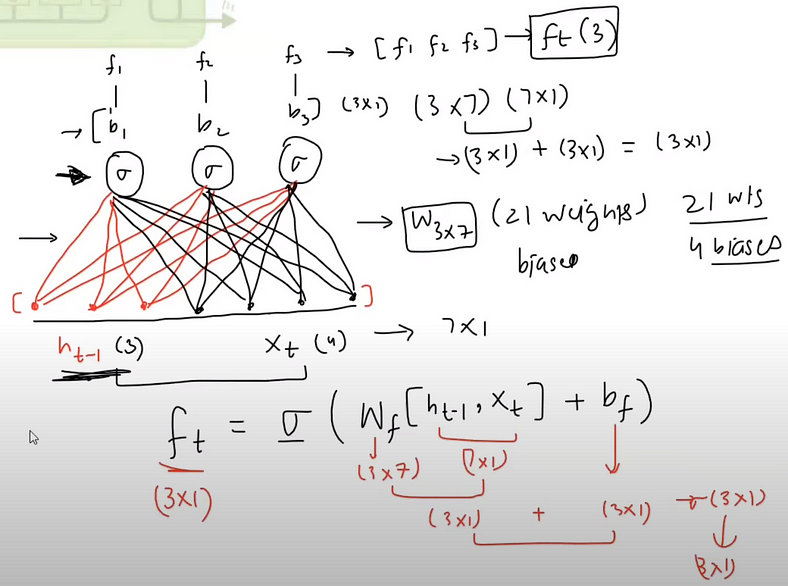
| \*\*Model Handling\*\* | Traditional RNNs perform well for short-term dependencies | LSTMs and GRUs are designed to handle long-term dependencies using gates and memory cells |

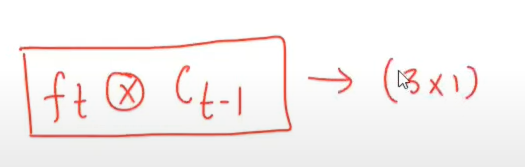
| \*\*Challenges\*\* | Easy to retain recent information, but might over-focus on recent data | Harder to retain distant information in traditional RNNs due to vanishing gradients |

In essence, \*\*short-term dependencies\*\* involve information that is useful within a short range, while \*\*long-term dependencies\*\* require the model to carry useful information over a longer distance in the sequence.

**Forget Gate workflow :**

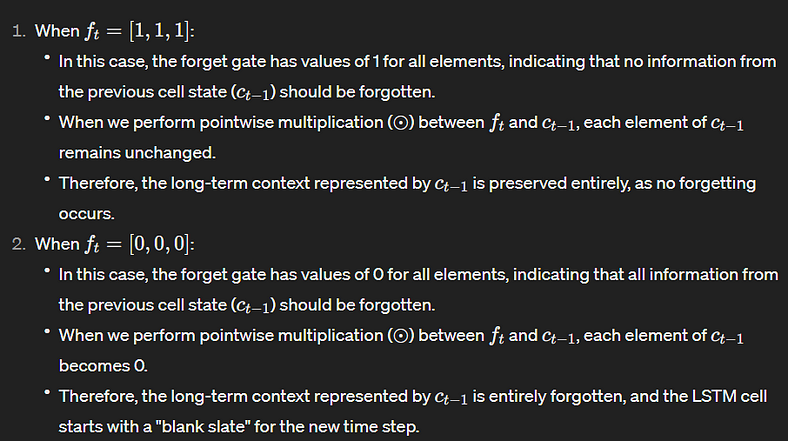


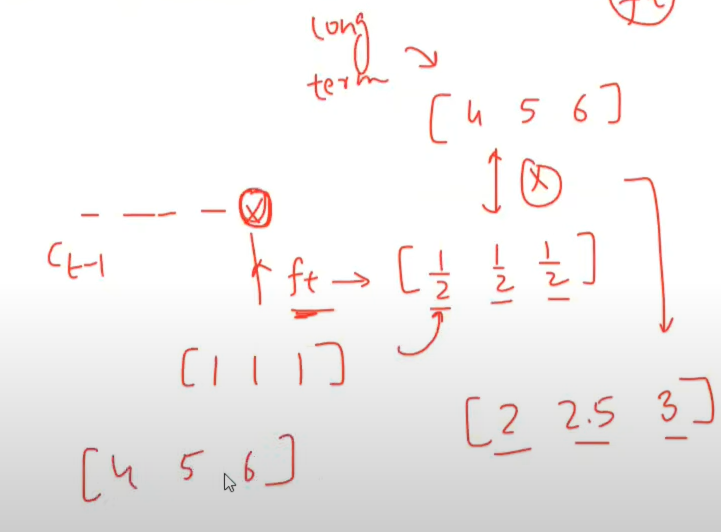


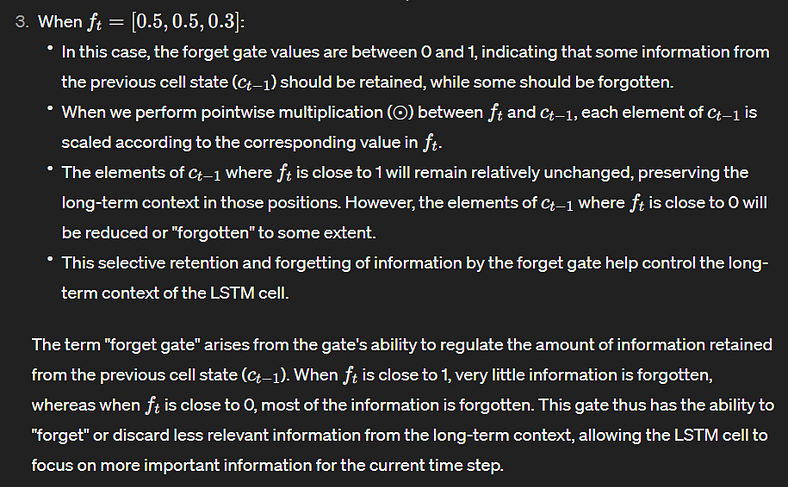


**How forget controls the long term context ?**

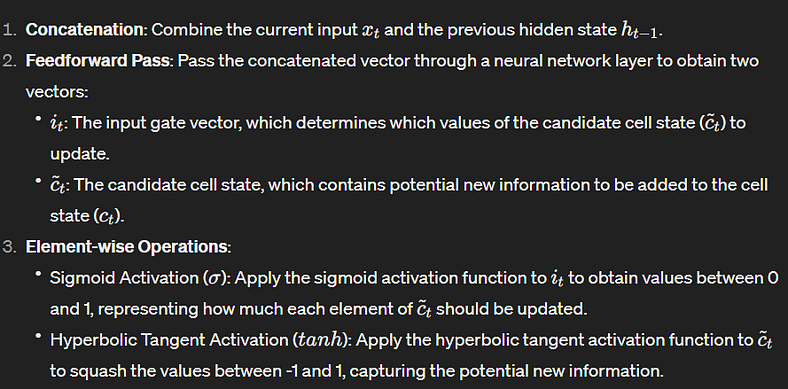
Let’s take an example to illustrate how the forget gate controls the long-term context and why it’s called the “forget” gate.

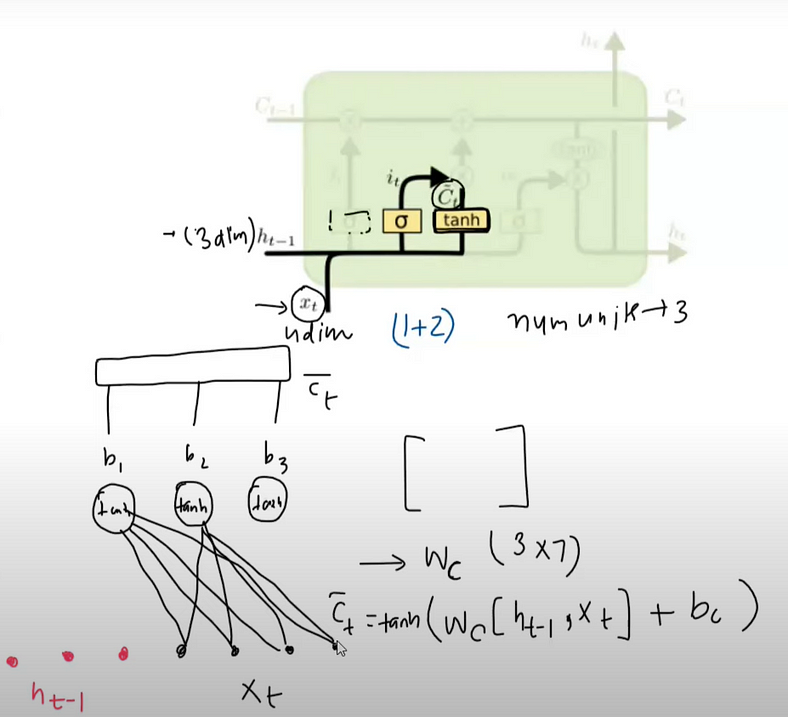


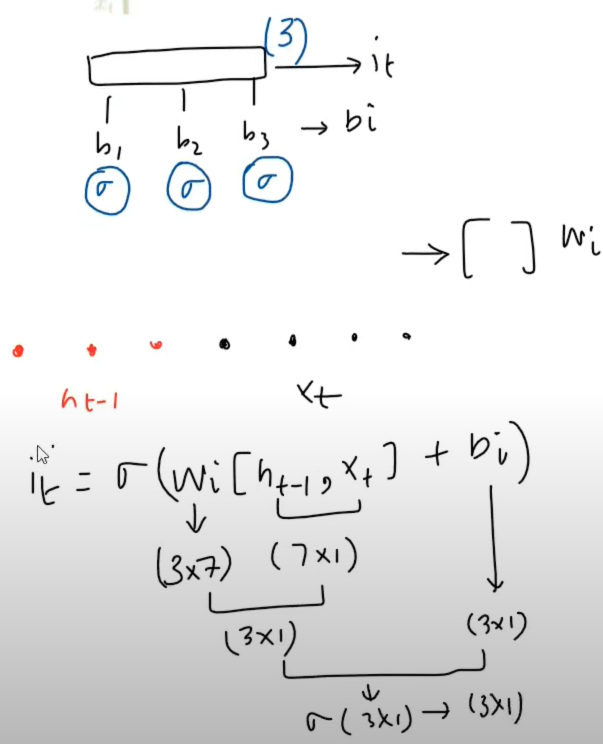


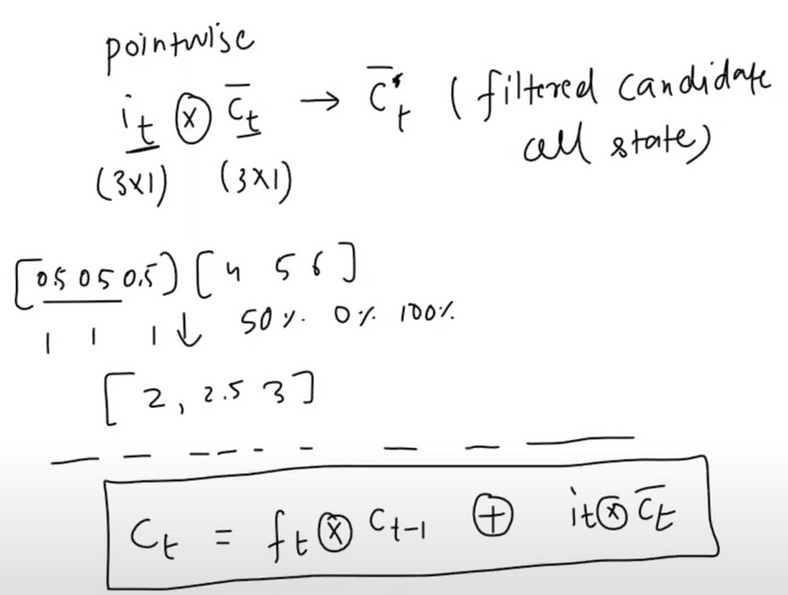


**Input Gate workflow :**

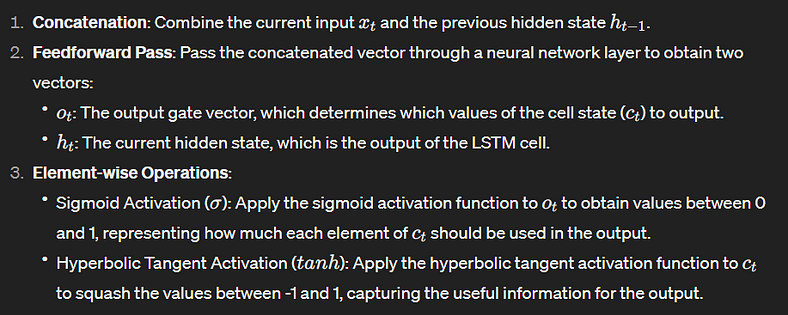


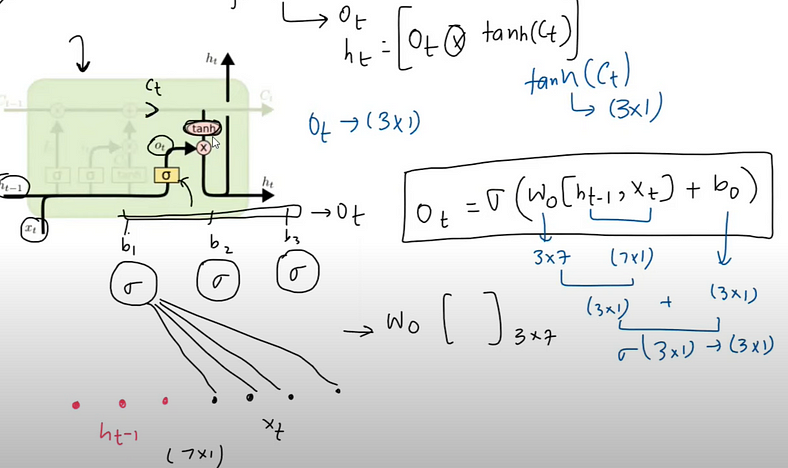


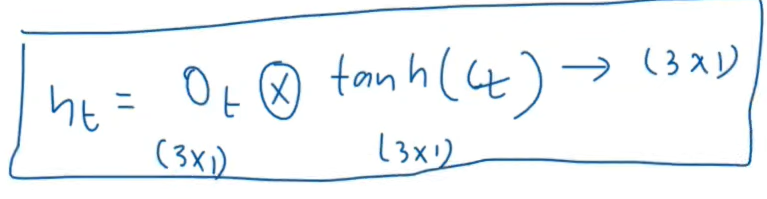




**Output Gate workflow :**







***Now see the following LSTM animation to understand the working of LSTM :***

