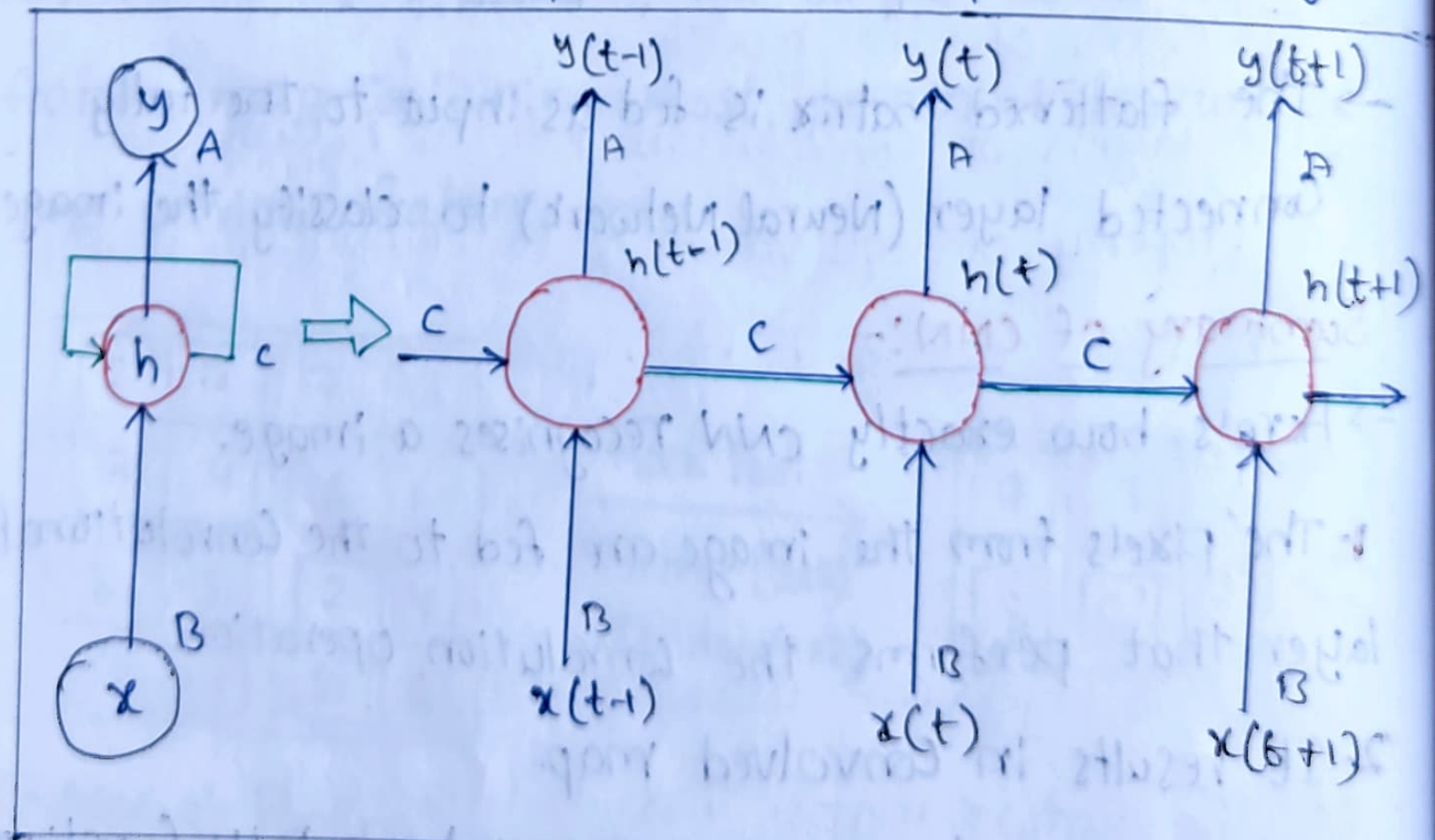
Recurrent Neural Network (RNN)

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.



Fully Connected Recurrent Neural Network

→ A, B, C are the parameters of the Network.

→ Here, "x" is the input layer, "h" is the hidden layer, and "y" is the output layer. A, B, C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.

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

$h(t) \rightarrow$ new state
$f_c \rightarrow$ function with parameter c.
$h(t-1) \rightarrow$ old state
$x(t) \rightarrow$ i/p vector at time step t

* Why RNN's?

RNN were created because there were a few issues in the feed-forward neural network (ANN's).

→ Cannot handle sequential data
→ Considers only the current input.
→ Cannot memorize previous inputs.

→ The solution to these issues is RNN. which can handle sequential data, accepting the current i/p data and previously received i/ps and also can memorize previous i/p's due to their internal memory.

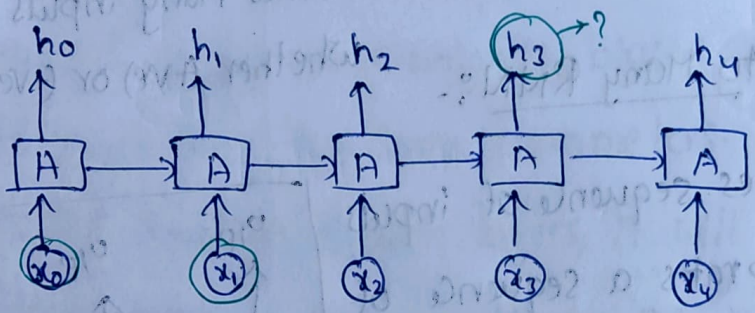
2. Exploding Gradient Problem :-

For sometimes, we only need to look at recent information to perform the present task.

NLP Ex:- When we try to predict the last word

"the clouds are in the _____"

→ RNN can able to predict the word as "sky". Since the gap between the relavant information and the place that it's needed is small, RNN can learn to use the past information.



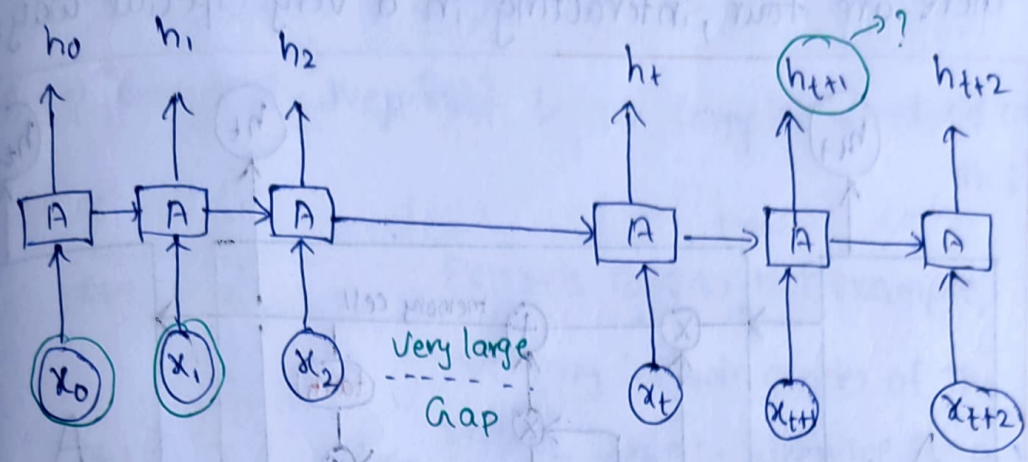
→ But there are also cases where we need more context.

NLP Ex:- Let's trying to predict the last word in the text "I grew up in France. I speak fluent ?"

French.

→ The Recent information the next word is probably the name of a language, but if we want which language, then we need the context of "France" from previous sentence. Here gap between relavent information and the place it is needed is very large.

* Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.



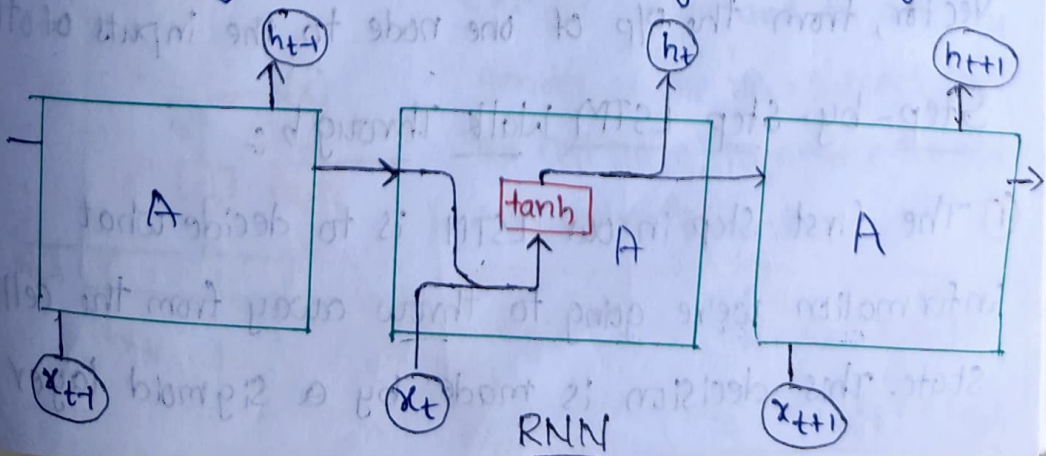
→ Here comes the legend LSTM RNN.

LSTM - RNN

Long Short Term Memory Recurrent Neural Networks usually called "LSTMs" are a special kind of RNN capable of learning long-term dependencies.

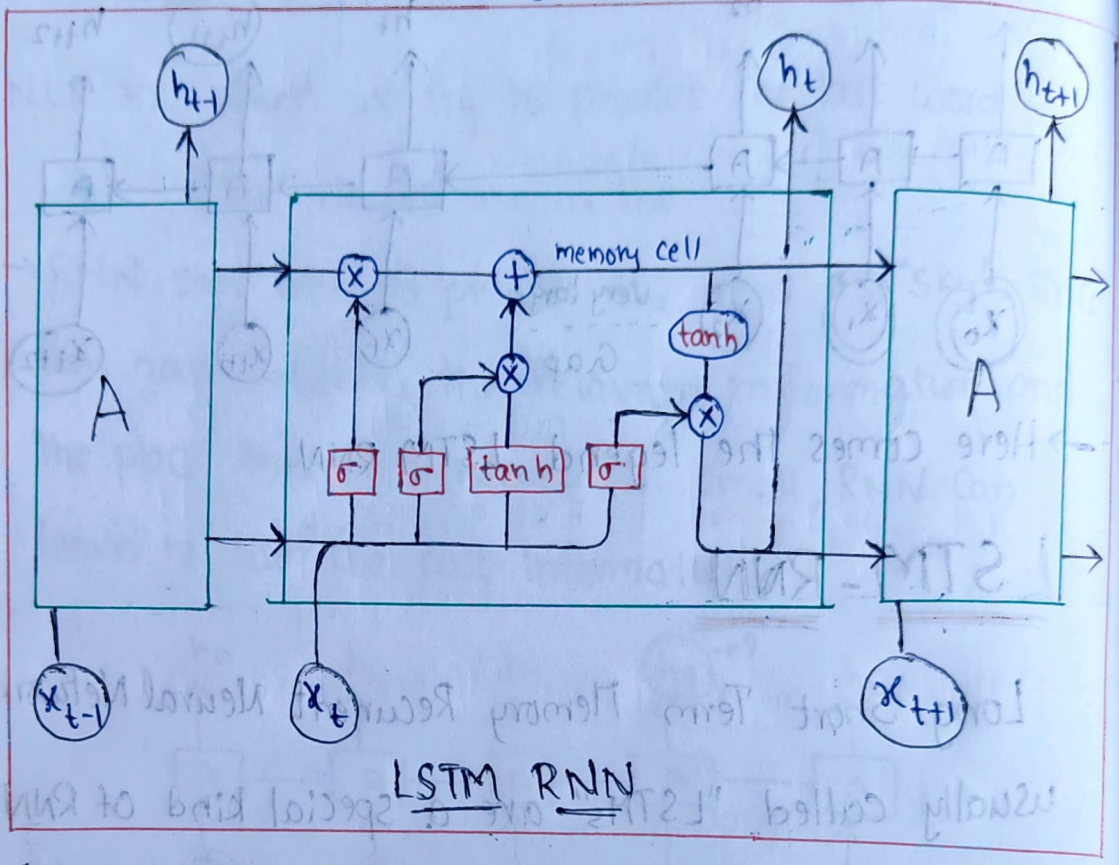
→ LSTMs have ability of remembering information for long periods of time.

→ All RNN's have the form of a chain of repeating modules of neural network. In standard RNNs, the repeating module will be a single tanh layer.



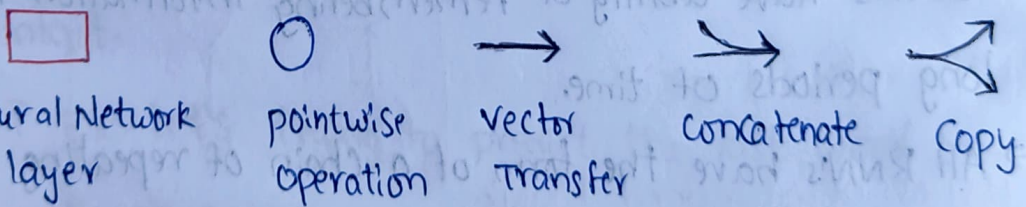
RNN

→ LSTMs also have this chain like structure, but the repeating module has a different structure there are four, interacting in a very special way.



LSTM RNN

Notations:



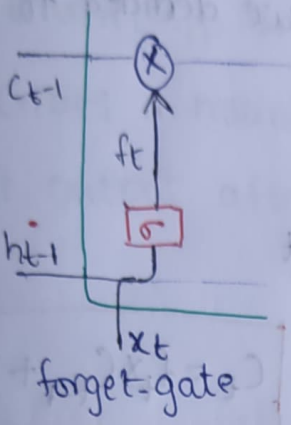
→ In the above diagram, each line carries an entire vector, from the o/p of one node to the inputs of others.

Step-by-Step LSTM Walk Through:

① The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer

called the "forget gate layer". It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 (sigmoid(σ)) for each number in the cell state C_{t-1}.

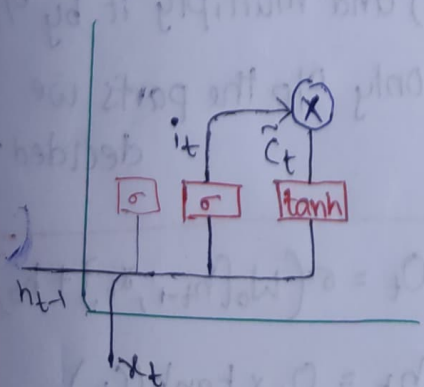
1 → "Completely keep this" 0 → "Completely get rid of this"



Ex:- Lets consider NLP example, It may include gender of the present subject. When we see a new subject we want to forget gender (old subject).

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

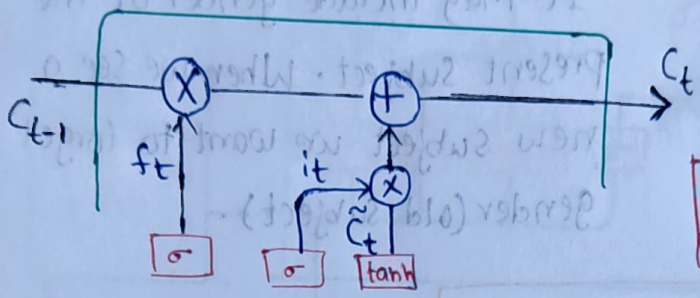
② The next step is to decide what new information we're going to store in the cell state. This has two parts. A sigmoid layer called "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$ that could be added to the state. In next step, we'll combine these two to create an update to the state.



Ex:- If we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

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

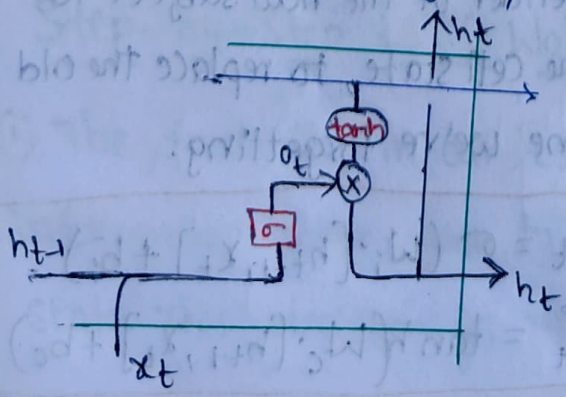
③ Now, we will update the old cell state, C_{t-1} into the new cell state C_t. We multiply the old state f_t, forgetting the things we decide to forget earlier. Then we add $i_t \times \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.



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

Ex:- where we'd actually drop the info old subject (gender) and add the new information, as we decided prev.

④ Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we 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 (to push values between -1 & 1) and multiply it by o/p of the sigmoid gate, so we only o/p the parts we decided to.



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

$$h_t = o_t \times \tanh(C_t)$$

Ex:- NLP → "John played tremendously well and won for his team. For his contributions, brave _____ was awarded player of the match.

→ There could be many choices for empty space. The current i/p brave is adjective, and adjective describes a noun(John). So, "John" could be the best output after brave.