Week 2

provide a brief summary of what RNNs are, how they function, an overview on their architecture and an understanding of LSTMs, GRUs and Encoder-Decoder models.

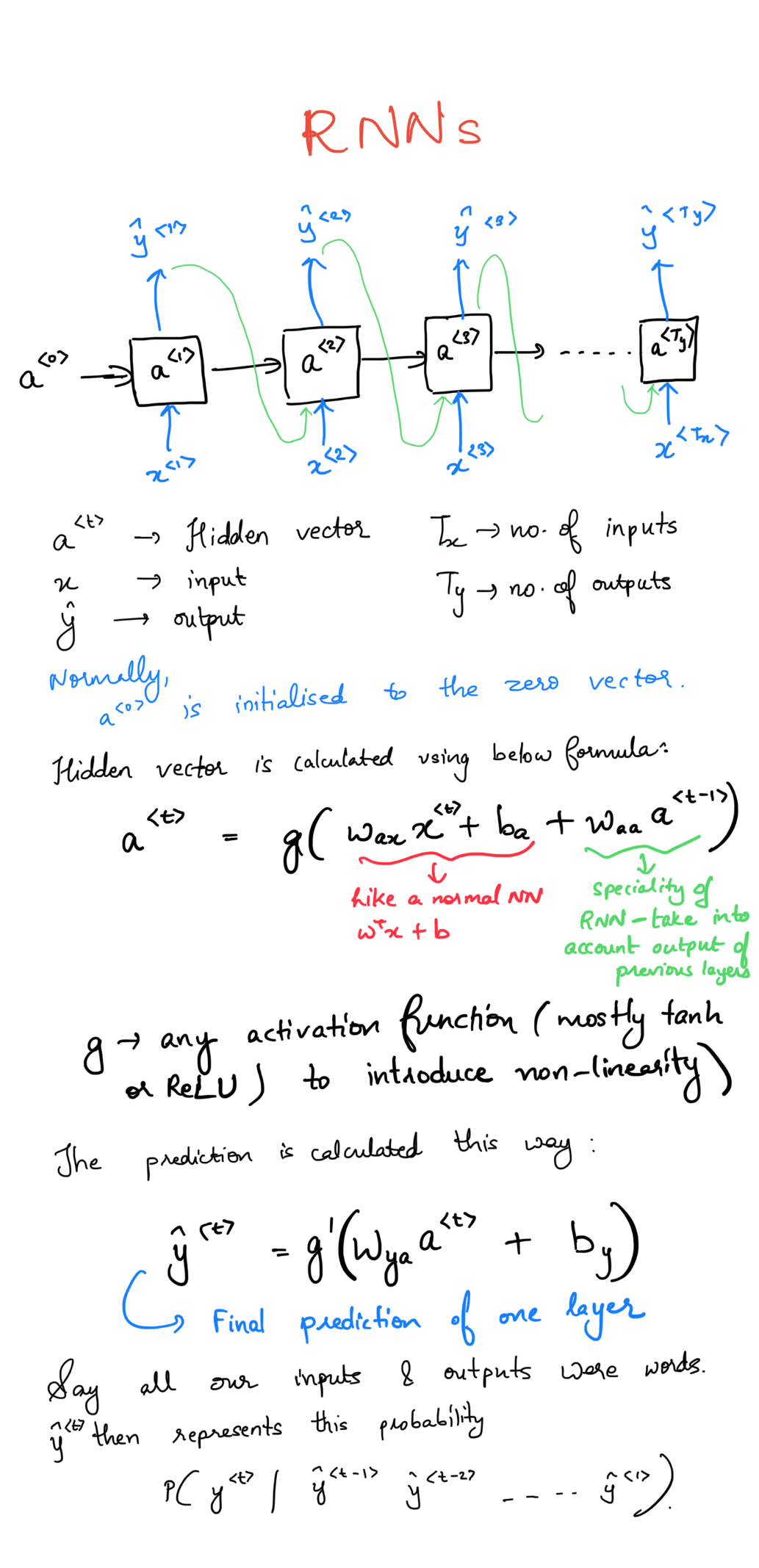
What are they:

RNN stands for Recurrent Neural Networks. They are a form of neural networks that help us deal with sequences in input or output data. Unlike other forms of neural networks where the output only depends on the current input, RNNs can use information from the previous input/output along with the current input to generate the next output. This becomes really helpful in applications like speech recognition, music generation, sentiment analysis, etc.

How they function:

Basically,

* RNNs receive sequential data in batches.
* The data which is the input layer is passed into the hidden layers.
* The hidden layers process the data according to the formula given below and store it in their memory.
* The successive time steps use the current input and stored memory (output of all the previous time steps) to predict the next sequence.
* The output layer generates the network's prediction according to the formula given below.
* Weights for generating output (Wya) and weights used in hidden layers (Wax) are shared parameters across all time steps.
* The loss function used is the binary cross entropy loss.



Architecture:

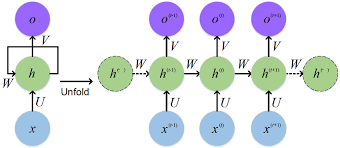
Input Layer: takes an input x<t> at each time step

Hidden Layer(s): Stores the internal state at each time step which stores information about previous inputs, and processes data sequentially. It captures the sequential dependencies of the input data. At each time step t, it updates the hidden state/vector according to the given equation.

Recurrence: output from one time step is passed as one of the inputs to the succeeding time step in the hidden layer. This creates a loop like, recurring structure, giving RNN its name and most distinctive feature

Output Layer: generates the prediction based on the hidden state

The structure can be represented this way:

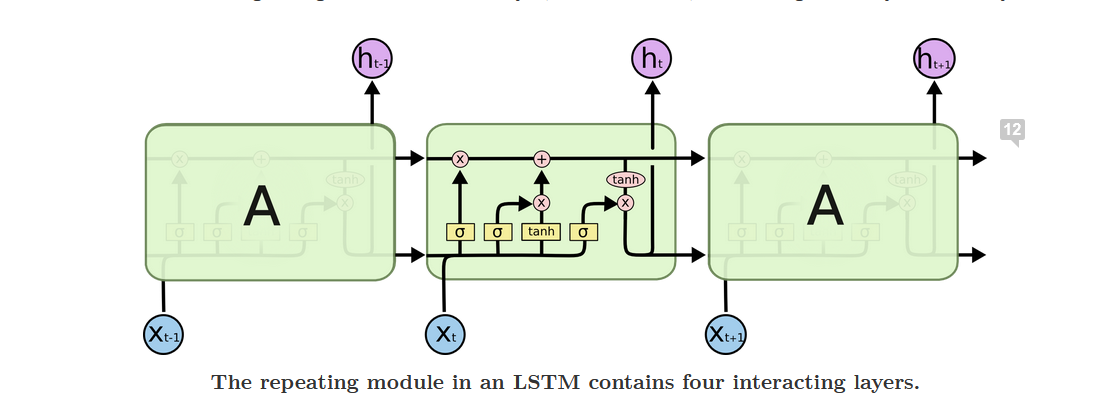


How LSTMs work:

They are a powerful modification to normal recurrent networks that solve the problem of vanishing and exploding gradients thereby helping capture long term dependencies in sequential data.

LSTMs take in 3 inputs - xt, ct-1 and ht-1 and give out 2 outputs – ct and ht.

The structure of LSTMs is shown below:



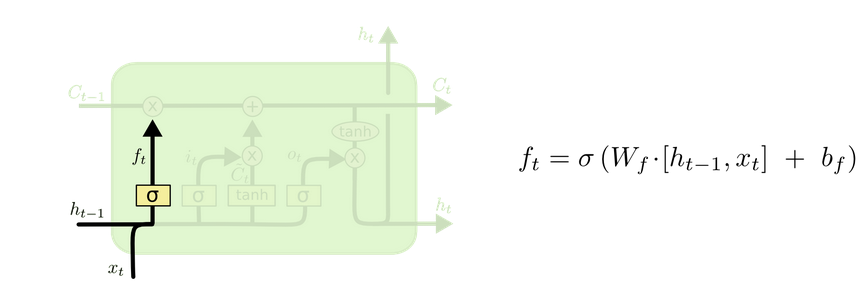


The red arrow pointing to the top horizontal line is the cell state, which is the key to LSTMs.

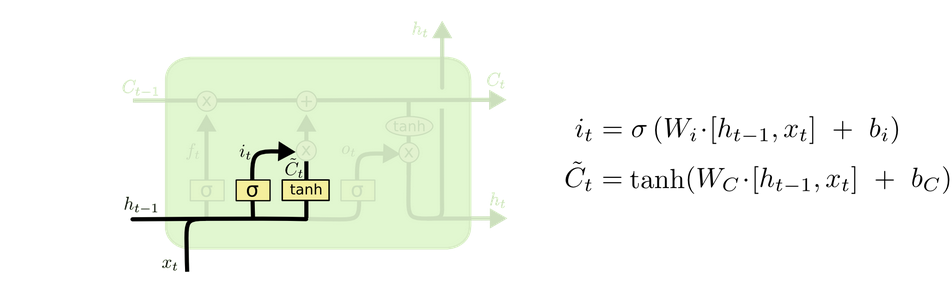
It’s very easy for information to flow along that line without being changed but the LSTM does have the ability to add or remove information from it, through gates.

There are 3 gates, discussed in detail below.

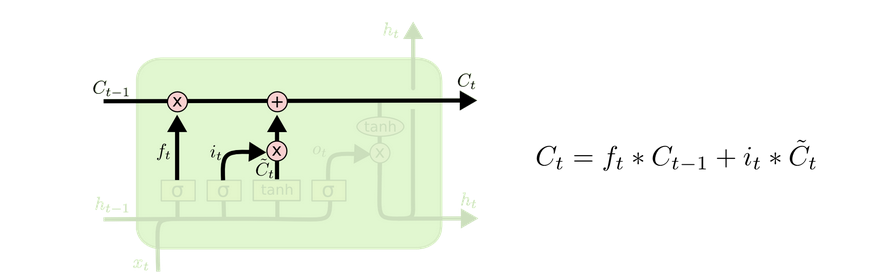
1. **Forget gate:** this is the first gate that is encountered and it decides what information should be discarded from the cell state. The decision is made based on the result from a sigmoid activation function which takes in ht-1 and xt as input. Output value of 1 represents “keep everything” and output value of 0 represents “get rid of everything”

****

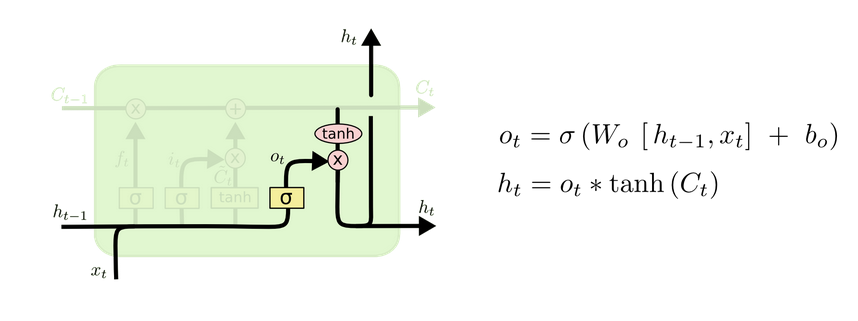
1. **Input gate:** decides what new information should be stored in the cell state. This has 2 parts. To decide which values will be updated, the previous hidden state(ht-1) and current input(xt) into a sigmoid function, which is called the “input gate layer”. Next, a tanh layer creates a vector *C*~*t* of new candidate values, which could possibly be added to the cell state.

****

1. **Cell state:** this gate involves replacing the previous cell state with the new one. It combines the outputs of the previous 2 steps. The old cell state is multiplied by ft, forgetting the things we decided need to be forgotten. Then, elementwise addition is performed on our outputs from the input gate (it and *C*~*t).* This gives us our new cell state.

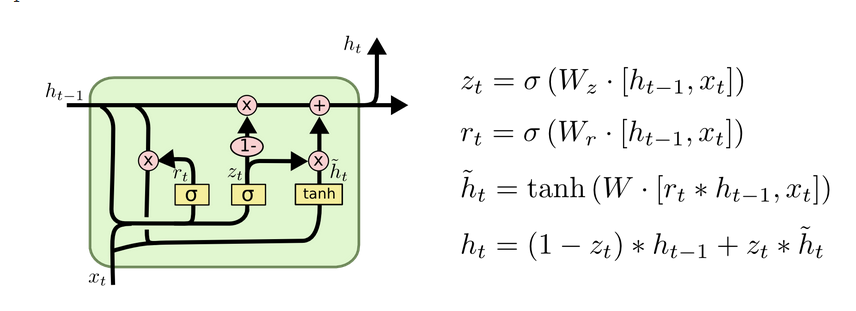
****

Finally, to decide what we’re going to output,



GRUs:

It is like a variation to LSTMs, one that makes it simpler. It combines the forget and input gates into a single “update gate”. In GRU, the cell state and hidden state are combined. In total, GRU has 2 gates – Update gate (to decide how much of the previous memory to retain and how much to update with the current input using just 1 hyperparameter) and the Reset gate (to tell us about the relevance of the previous memory).



Encoder – decoder

This type of RNN architecture is generally used when the length of input and output data differs, such as in machine translation, speech recognition, etc. it’s also called the sequence-to-sequence model.

The model can be divided into 2 parts:

**Encoder**

* Takes in the input sequence and converts it into an intermediate vector of fixed length.
* It captures the essential features of the input data into a compressed form, summarizing its key information.
* RNNs, LSTMs and GRUs can be used to build it.



Decoder:

* Takes in the intermediate vector as its input and uses the stack of several recurrent units to predict an output *y\_t* at a time step *t*.
* Each recurrent unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state.
* Final output is predicted using softmax function.