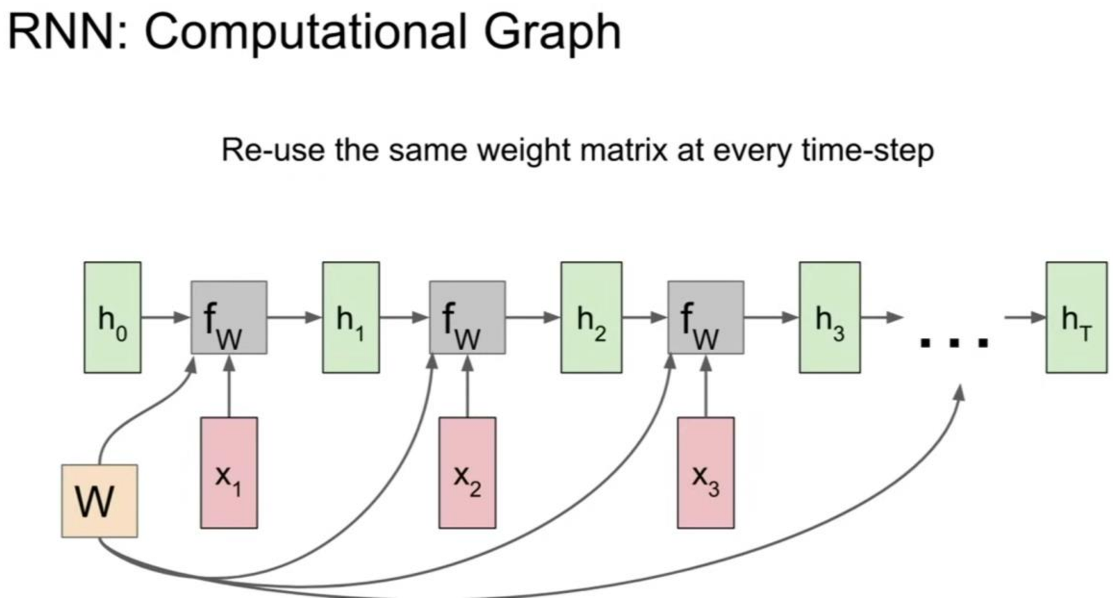
AI CLUB

Task 2

Overview of RNN's

The RNN architecture works on sequential data. It makes use of s feedback loop that allows the network to remember previous predictions and allows it to use those previous predictions and true data to make it's next prediction. The main point of RNN's is the feedback loop wherein a function of the hidden state and the input is used to form the next hidden state.

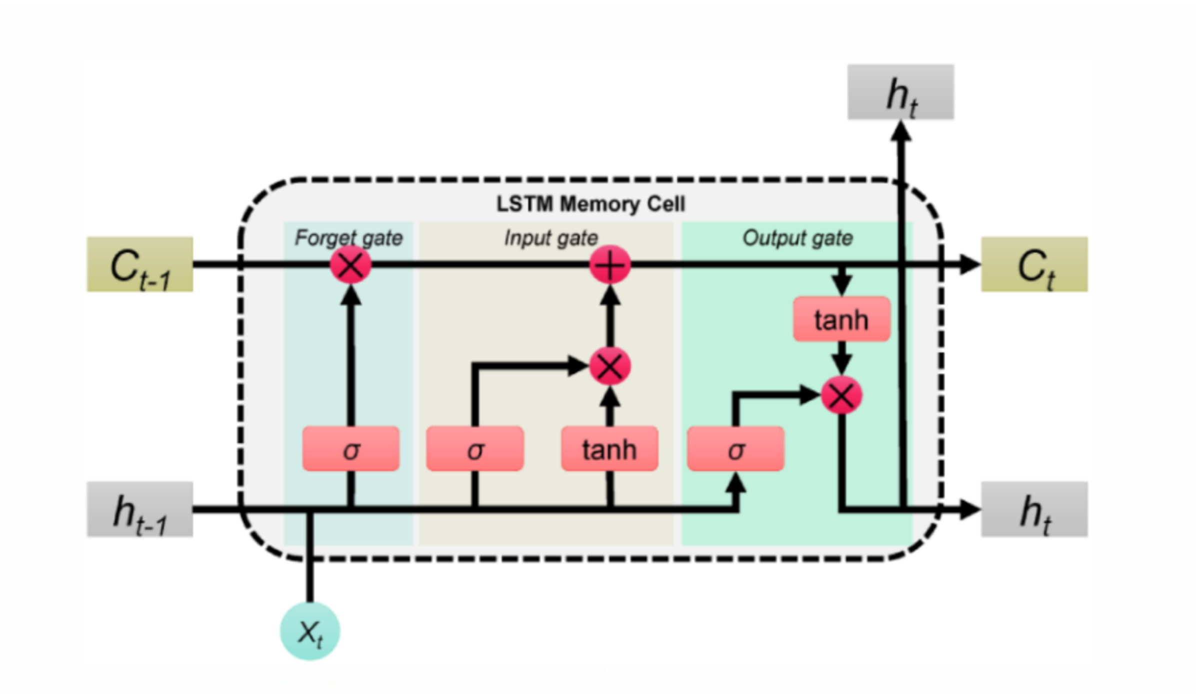


The above graph represents how data would flow in an RNN. First we initialize a hidden state h_0 then we supply the input x_1 then f_w is applied which works the same way as a normal neural network where $f_w = act(w_1x_1 + w_2x_2 + b_1)$ the activation function used can be sigmoid or relu or tanh etc. The output of the f_w is used as the next hidden state and the process continues. The previous inputs influence the future hidden states and this way the inputs are remembered. An important point to note is that recurrent neural networks use only the same weight matrix W for all the hidden state calculations thus reducing the parameters and allowing for variable amounts of sequential data to be processed.

The problem with RNNs is the exploding/vanishing gradient problem wherein if the weight of the hidden state is initialised to something more than 1 during backpropagation of the network with reasonable large amount of data the gradients become so massive that it becomes impossible to find the minima of the loss functions. If the hidden state gradient is initialised to something less than 1 then the gradient becomes so small it also becomes impossible to find the minima of the loss function.

LSTMs (Long-Short Term Memory):

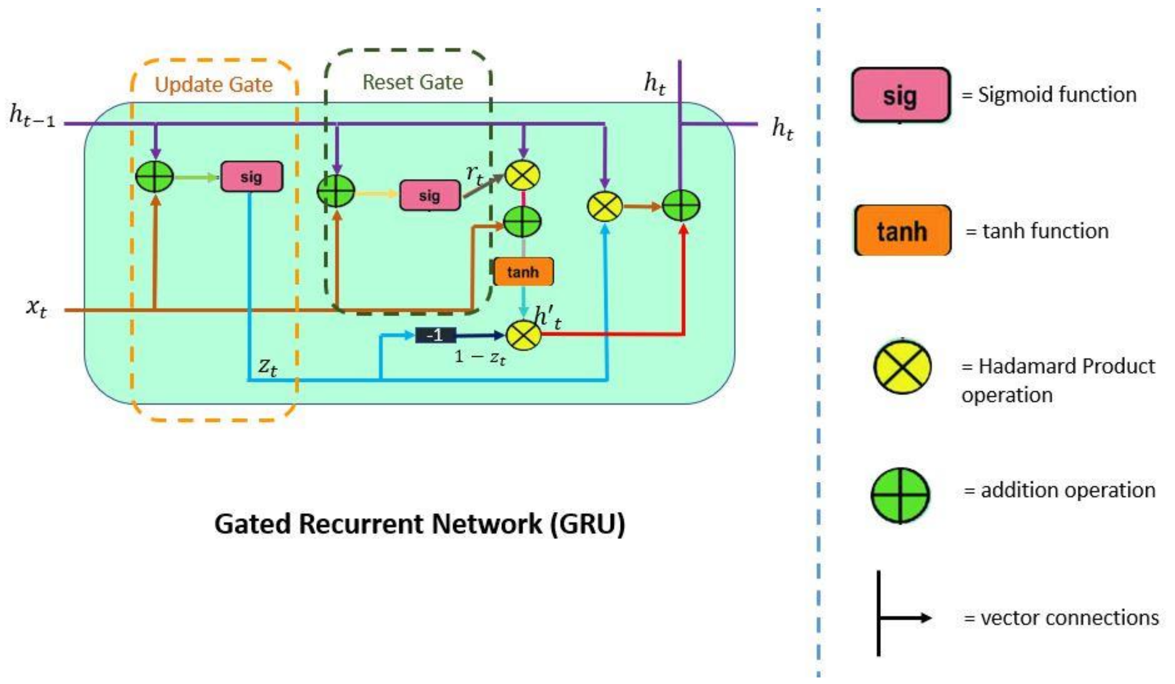
LSTMs are a variation on the vanilla RNNs. They have a long term memory and a short term memory where in the long term memory doesn't have explicit weights and biases whereas the short term memory does.



In this architecture the C_t represent the long term memory, h_t represent the short term memory and x_t represent the input. In the forget gate we first decide the percentage of the long term memory that we want to remember. In the input gate we then find the amount of potential long term memory to remember. In the output gate we make use of the long term memory and the short term memory to find the new short term memory which is the output of the LSTM. This architecture takes care of the vanishing gradient problem and doesn't allow the gradients to become 0

GRUs:

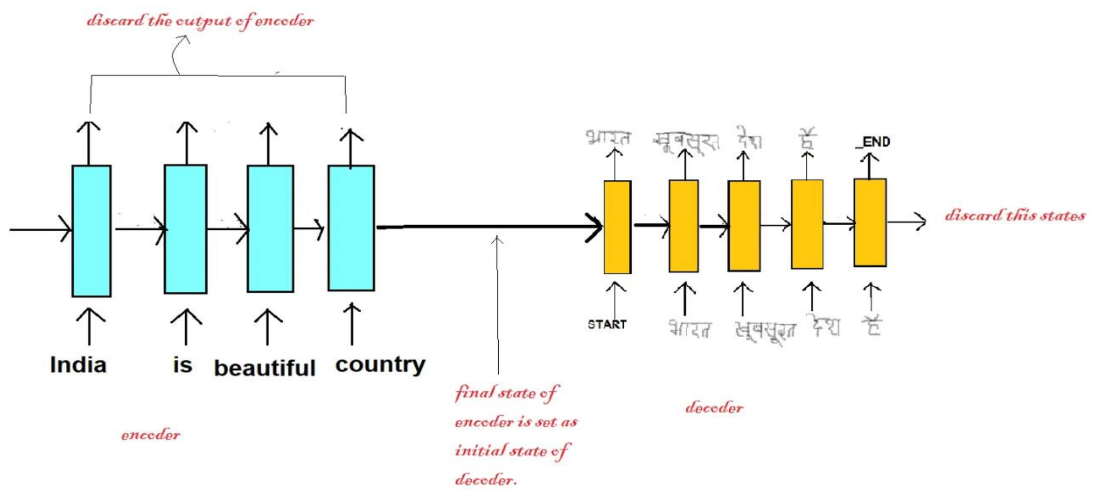
GRUs have two gates namely the reset gate and the update gate. The reset gate determines how much of the previous hidden state to forget and the update gate determines how much of the new input vector to incorporate into the new hidden state. They are similar to LSTMs but with smaller number of parameters making it computationally less expensive. May not perform as well in situations that require the need to remember long term dependencies like LSTMs can.

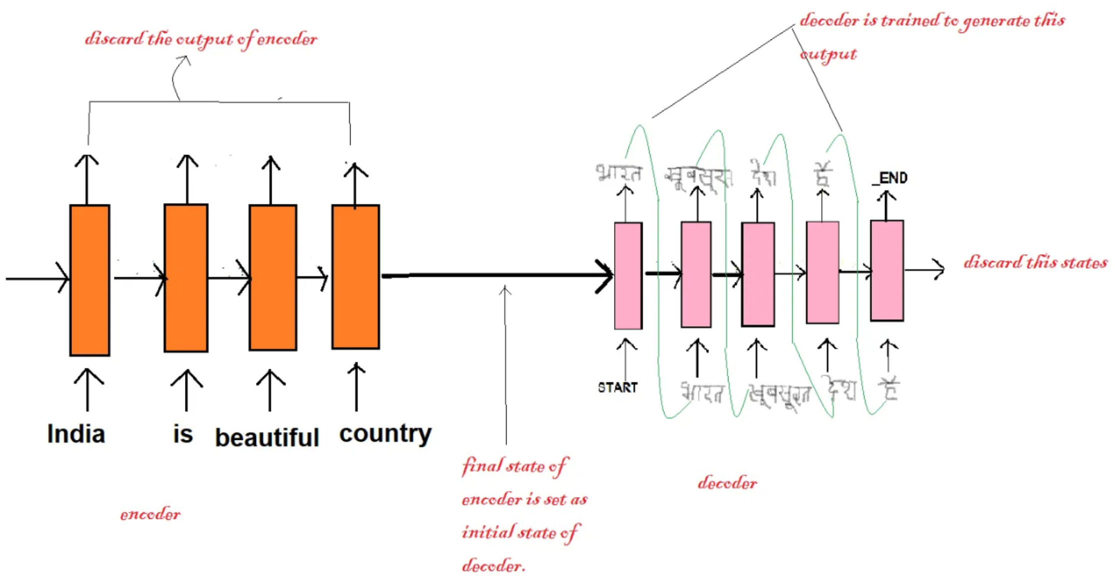


Gated Recurrent Network (GRU)

Encoder Decoder model:

The main unit of this model is an LSTM or GRU unit. The encode part of the model takes in an input like a word at each time step, the final output of the last unit is taken as the output of the Encoder part of the model. This gives us a tensor that stores the data of the various inputs given to it. The decoder model works differently during training and testing periods. During training period the decoder model is forcefully given the correct output at each time step and not the predicted output. In the testing phases the predicted output is given as the input in each time step. The initial hidden states in each unit of decoder model is the same as the final output of the encoder model.




discard the output of encoder
decoder is trained to generate this output
भारत
खूबसूरत
देश
है
_END
discard this states
START
भारत
खूबसूरत
देश
है
India
is
beautiful
country
final state of encoder is set as initial state of decoder.
decoder
encoder