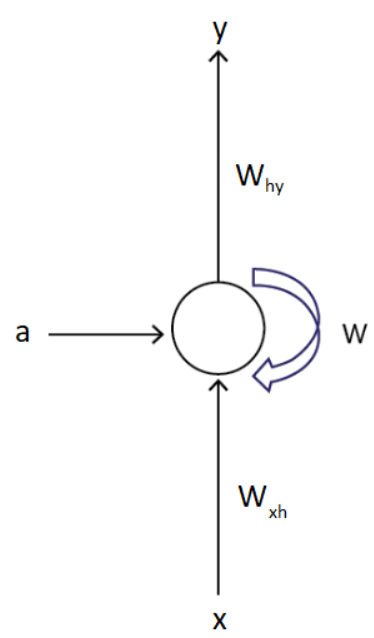
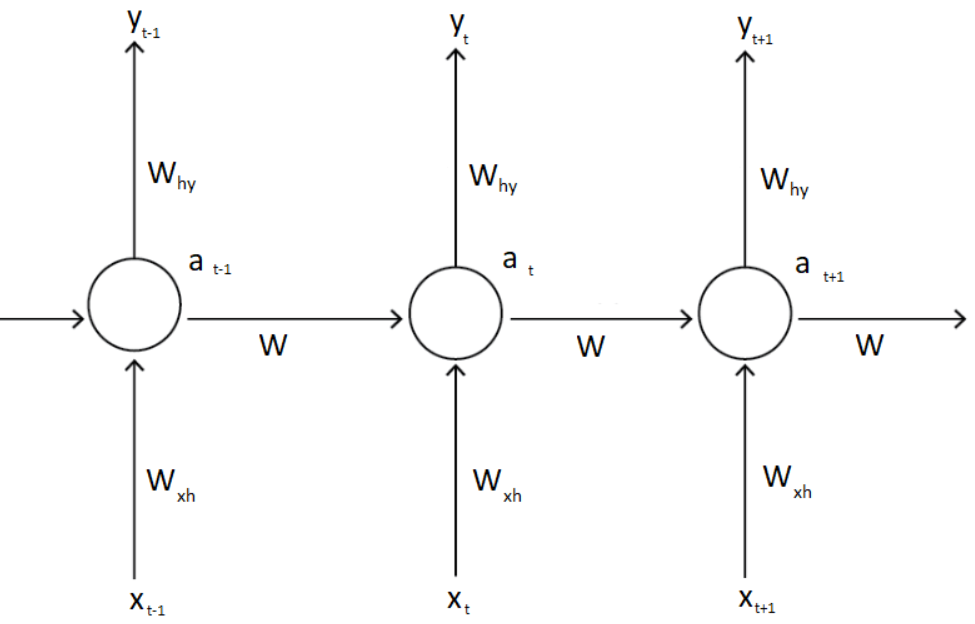
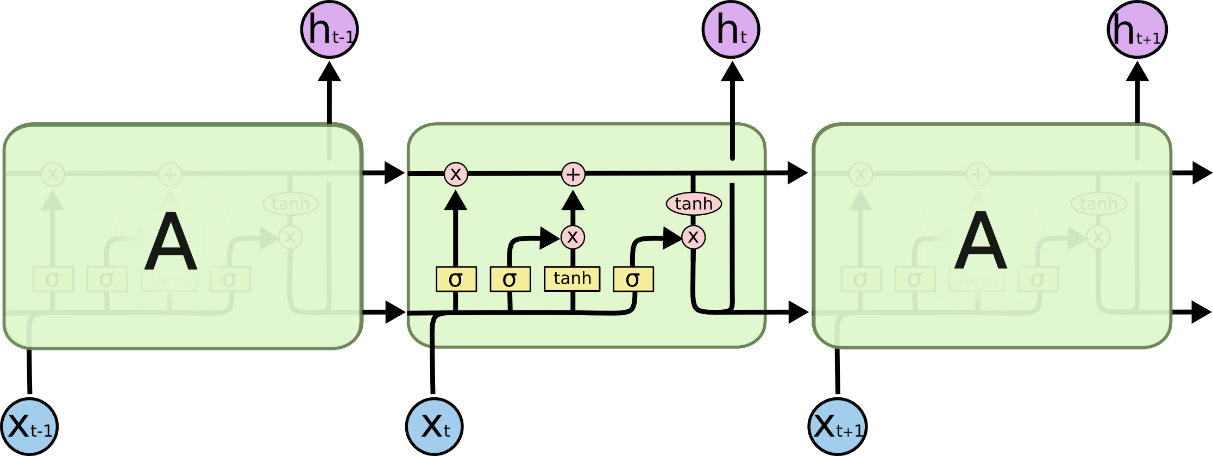
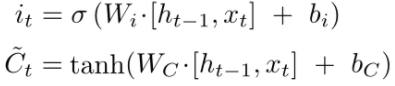
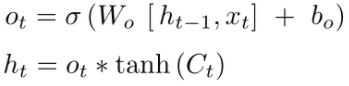
**RNN**

* **Working (Basic Structure and Usage):**Contributed by  [David Rumelhart](https://en.wikipedia.org/wiki/David_Rumelhart) in 1986, Recurrent neural networks are a type of neural networks that have an internal memory which allows output of previous layers to be served as input to the next layers.  
    
  Fig 1: Structure of traditional RNN [1]  
    
  The simple structure of RNN contains an input state (x), hidden state and an output state (y), whereas wxh, w and wxy are the weights of the network. As it’s name indicates, we get a recurrent structure as we unfold above compressed structure:  
    
  Fig: RNNs when Unfolded [1]  
    
  @Kshitij Add Mathematical Part (y=?, Loss Function etc.)  
  <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>  
  <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>
* **Usage**RNNs allow data to persist as they have loops in them.RNNs are mostly used in the field of Natural Language Processing (NLP) and Speech Recognition to recognize sequential characteristics of data and predict the future’s likely scenario  
  RNN is extensively used among other types in applications having sequence of data and even in convolutional layers in order to make the pixels effective.   
  In 1993, a Neural History Compressor solved a difficult deep learning task that required more than 1000 layers of RNN. [2]
* **Problem**  
  Despite being so powerful, RNN has several drawbacks like Gradient vanishing and exploiding as it is difficult to capture long term dependencies because of multiplicative gradient that can change exponentially with respect to the number of layers. Therefore, they cannot be applicable for long sequence of data. Theoretically, RNNs perform well for long-term dependencies but practically they fail to prove it.  
  // Extra Down  
  When doing back propagation, each node in a layer calculates it’s gradient with respect to the effects of the gradients, in the layer before it. So if the adjustments to the layers before it is small, then adjustments to the current layer will be even smaller. That causes gradients to exponentially shrink as it back propagates down. The earlier layers fail to do any learning as the internal weights are barely being adjusted due to extremely small gradients.
* **Why LSTM is needed**  
  To overcome these drawbacks of RNNs, they are modified to develop Long Short Term Memory (LSTM) networks. Created as a solution to Short-Term memory, LSTMs are need of an hour perform way better than other techniques for long-range dependencies. Coined by 2 computer scientists, [Sepp Hochreiter](https://en.wikipedia.org/wiki/Sepp_Hochreiter) and  [Jürgen Schmidhuber](https://en.wikipedia.org/wiki/J%C3%BCrgen_Schmidhuber) in 1997, the initial version of LSTM architecture did not include Forget gate which was added later in 1999. Many applications which could not be solved by RNNs would have remained unimplemented if LSTMs did not exist.

**LSTM**

* **Structure**  
    
  Fig: LSTM Cell with it’s operations [3]  
  LSTM consists of:  
  The information: which is passed through the modules is in the form of vectors.   
  Tanh function: is used to regulate the values of a neural network.  
  Sigmoid: is used to update/ forget the information. The more the value is closer to one, it’s completely passed on and the closer it is to zero, it is eliminated.  
  The cell state: is the memory of the network. Information is added or removed to the cell state by gates.  
  The gates decide what information stays or forgot by the network. Each gate in itself is a neural network.  
  Forget Gate: Decides what information should be kept/thrown away from previous steps.  
  Input Gate: Decides what new information to be added in current step.  
  Output Gate: Decides next hidden state.  
  Having this architecture, LSTMs do not need to strife to learn the behavior of practically remembering data for long time as it’s their default property.
* **Functioning**  
  As the neural networks mimic the functioning of the brain, LSTMs mimic the functioning of our memory. Internal mechanisms of LSTMs are operated through ‘Gates’ that regulate the flow of data. The gates take key decisions such as which information in the sequence is crucial and which can be ignored. LSTM maintains long chain of information in which ‘the information which is stored from the beginning’ is available. This information is then used for predictions.  
  First, data from previous hidden state and data from current input serve as input to the sigmoid activation function. This is Forget gate.  
    
  Next, Input gate is encountered which is responsible for updating the cell state. The same input as Forget gate is passed on to Input gate which consists of 2 layers. First, Sigmoid layer and second, Tanh layer. The sigmoid layer processes the input in order to determine which values will be updated and the tanh layer processes the input so that the flow of the network is properly regulated. Then, multiply the outputs of both layers to decide which information to keep.  
    
  Meanwhile, cell state is updated. The previous cell state (Ct-1) is pointwise multiplied with result of forget gate. The result is then added with the result of Input Gate to generate a new Cell state to be passed along.  
    
  Finally, the output gate decides the next hidden state which predicts the results based on it’s information from previous inputs. The same inputs as of Forget gate and Input gate is passed on to the Sigmoid function and the updated cell state is passed to Tanh function and the output of both these functions is multiplied to create a new hidden state to be passed on to the next time step.  
    
  @Kshitij Maths Part Just explain the terms  
  https://colah.github.io/posts/2015-08-Understanding-LSTMs/
* **Error Backpropagation  
  @Kshitij**
* **Operations**
  + **Usage**
  + **Examples**
* **Current Work Being Done**  
  During recent few years, LSTMs have gained humongous popularity by it’s incredible and irreplaceable success in fields like Speech Recognition, Language Modeling and Translation, Image Captioning, Text Classification etc.
* **Conclusion**
* **References**  
  [1] Medium. 2020. *Introduction To The Architecture Of Recurrent Neural Networks (Rnns)*. [online] Available at: <https://medium.com/towards-artificial-intelligence/introduction-to-the-architecture-of-recurrent-neural-networks-rnns-a277007984b7> [Accessed 3 September 2020].  
  [2] Schmidhuber, Jürgen (1993). [*Habilitation thesis: System modeling and optimization*](ftp://ftp.idsia.ch/pub/juergen/habilitation.pdf) *(PDF)*. Page 150 ff demonstrates credit assignment across the equivalent of 1,200 layers in an unfolded RNN.   
  [3] Colah.github.io. 2020. Understanding LSTM Networks -- Colah's Blog. [online] Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> [Accessed 4 September 2020].