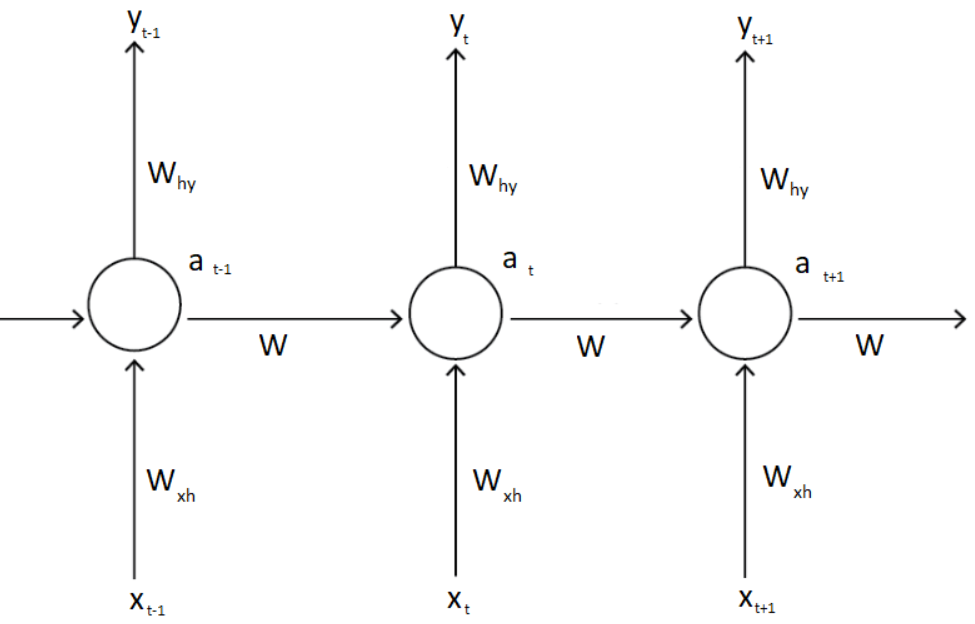
**Title: Applications of Neural Networks with Long Short Term Memory (LSTM)**

* **RNN Structure:**

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. 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 the compressed structure.  
  
Fig 1: Structure of traditional RNN 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 and 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. 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. While performing error back propagation, Gradient of each node is calculated according to the gradient in previous layer. Hence, if the previous gradient is small then the current gradient will be even smaller. This is the reason why gradients shrink as it propagates back and the initial layers cannot learn effectively due to low 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 that 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 is better in analysing emotion of long sentences and can produce better accuracy rate and recall rate. It can get complete sequence and information effectively.

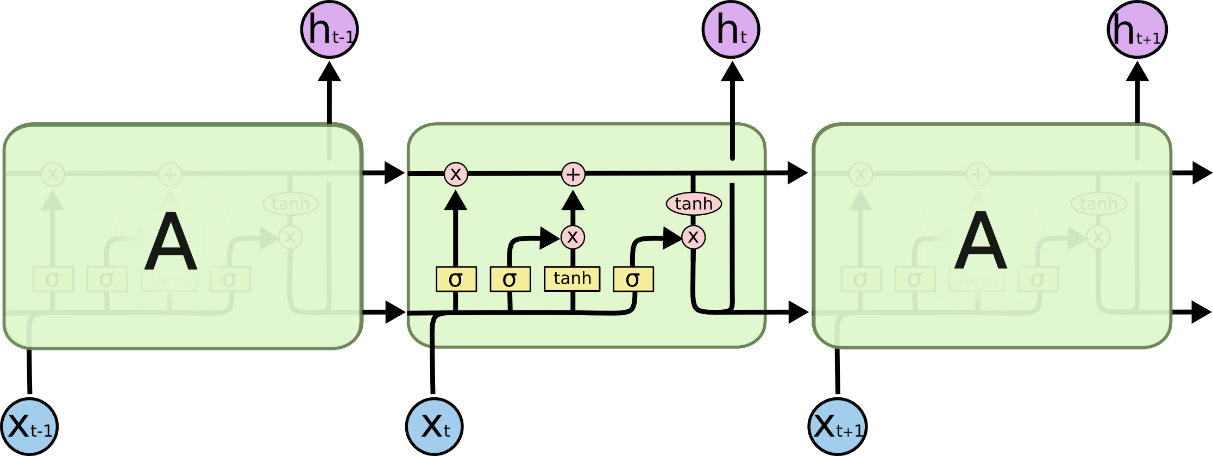
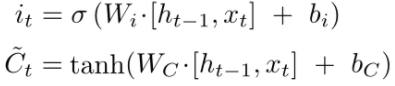
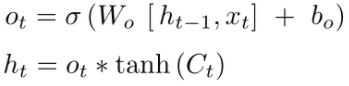
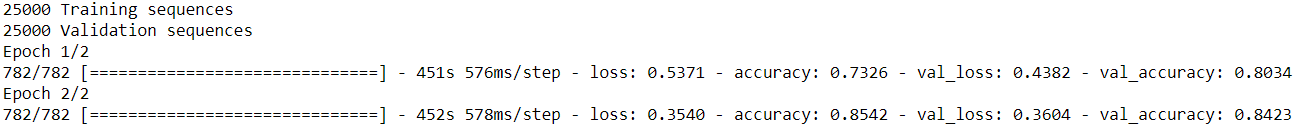
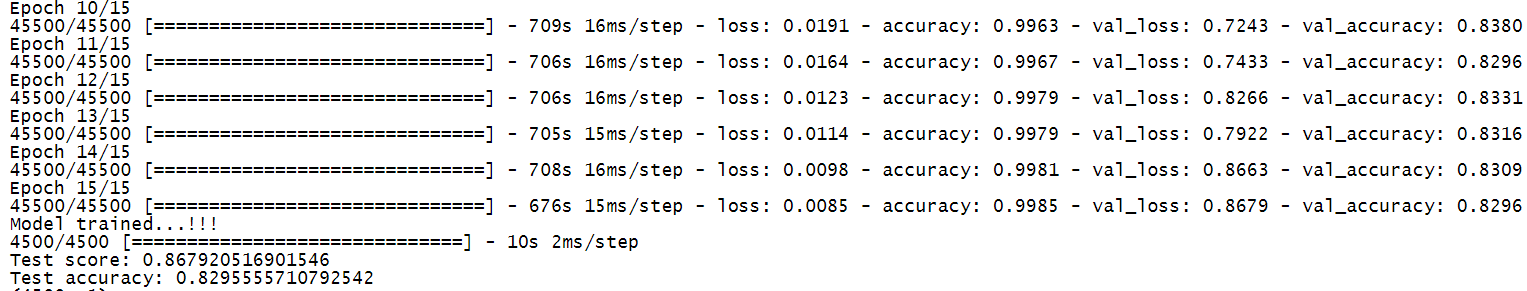
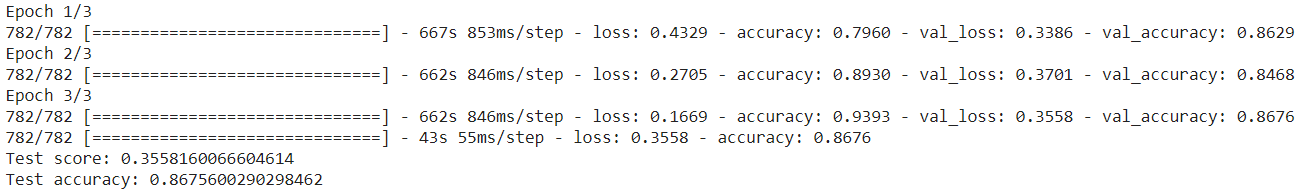
* **LSTM Structure**  
    
  

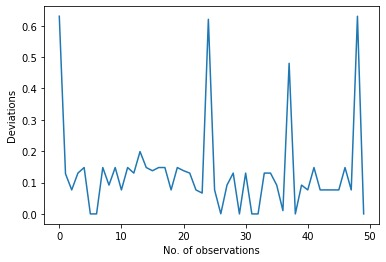
Fig: LSTM Cell with it’s operations [3]  
  
LSTM consists of:  
The information: which is passed through the modules is in the form of vectors.   
Tanh function: is used to regulate the values of a neural network.  
Sigmoid: is used to update/ forget the information. The more the value is closer to 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**
* **Examples**
  + **Sentiment Analysis:**

With the expanding popularity of deep learning, sentiment classification is an interesting and major topic in the field of Natural Language Processing (NLP). A basic task in sentiment analysis is classifying the polarity of a given text whether the expressed opinion is positive, negative or neutral. More Advanced, “beyond polarity” sentiment classification looks, for instance, at emotional states such as “angry”, “sad” and “happy”. [4]  
  
Initilly, we developed 2 LSTM models which consisted of 3 layers,1. Embedding, 2. LSTM and 3. Dense.  
First with standard and well-processed Imdb dataset provided by keras library and other with custom Imdb dataset [5] having 50000 reviews. Both the dataset were well-splitted into training and validation data. We got 84.23% and 82.95% accuracy respectively. We used One-Hot Encoding for word embeddings in the second model. During model training we also faced issues like overfitting and large difference between training and validation accuracy. Overfitting was easily solved by Early Stopping in which we reduced the number of epochs and stopped training as soon as it began diverging. And for the second issue was resolved by adding a Regularization term, in our case it was just hyperparameter tuning, thanks to the parameters ‘Dropout’ and ‘Recurrent Dropout’ provided by Keras Library. Meanwhile, Keras also updated it's internal features and we had to face some challanges. But we decided to go for a better model so that we might get better performance/results.  
  
  
Fig: LSTM Model trained on Keras Imdb Dataset  
Github Link: <https://github.com/Safir-Mohammad-Mustak-Shaikh/Sentiment-Analysis-on-Keras-IMDB-Dataset>  
  
  
Fig: LSTM Model trained on Custom Imdb Dataset  
Github Link: <https://github.com/Safir-Mohammad-Mustak-Shaikh/Sentiment-Analysis-on-Custom-Dataset-using-One-Hot-Encoding>  
  
Then with the same datasets, we built 2 Bidirectional LSTM models from which one fulfilled/served our purpose but the other one did not. So, finally, we came up with a Bi-LSTM model trained on Keras' inbuilt Imdb dataset and it gave 86.76% accuracy. Before training, we dealt with data pre-processing techniques like Removing stopwords and special characters that does not carry any meaningful information. Then, we converted the documents into lowercase and transformed the words into vectors of numbers in Imdb encoded format (Word Embedding). **This was one of our biggest research on converting the words into Keras encoded Imdb vector format.** Next, we truncated/padded the documents with a fixed length of 200 words. We have also set the vocabulary size as 20000. The model contains 4 layers,  
1. Embedding, 2. BidirectionalLSTM X 2 and 3. Dense.  
While creating the model, we have chosen the best optimization algorithm called "Adam". Since, Cross-Entropy is recommended for classification problems, we have used the same, i.e. "Binary Cross Entropy" in our case as we are aiming for 2 output classes. Also, we have saved the trained model and the corresponding weights so that we can use it later in any module by just loading the model (without the need of training again).  
  
  
Fig: Bi-Directional LSTM model trained on Keras Imdb Dataset  
Github Link: <https://github.com/Safir-Mohammad-Mustak-Shaikh/Bi-Directional_LSTM_Sentiment_Classification_Keras>  
  
After this we also calculated the correlation of the LSTM model values with the actual sentiment values from the labelled data. The Pearson correlation coefficient for LSTM = 0.882 and the Spearman correlation coefficient for LSTM = 0.886 which clearly depicts that LSTMs are better in long range dependencies.

  
Fig: Plot for LSTM predicted values deviated from user ratings

* **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**

As theRNNs practically fail to consider long-range sequences, a special unit of RNN called LSTM achieves to do so with a lot ease and efficiency. LSTM is widely used in many neural network applications and it has produced extra-ordinary results in many fields with Time-Series data, Long Sequence data etc.

* **Acknowledgments**

This research was possible under the guidance and direction of Prof. Dr. Jorg Schafer and Tutor Fatima Butt. This paper is the final report for the course Learning from Data (LFD) taught in the Masters programme High Integrity Systems (HIS).

* **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].  
[4] Medium. 2020. *Sentiment Analysis : Solutions And Applications Survey*. [online] Available at: <https://towardsdatascience.com/sentiment-analysis-solutions-and-applications-survey-9e52d3ea2ac7#:~:text=Problem%20definition,positive%2C%20negative%2C%20or%20neutral.> [Accessed 5 September 2020].

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).