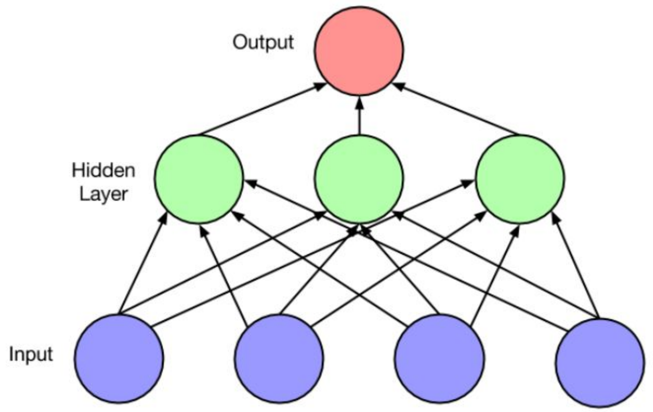
Long Short-Term Memory

Akshay Sood

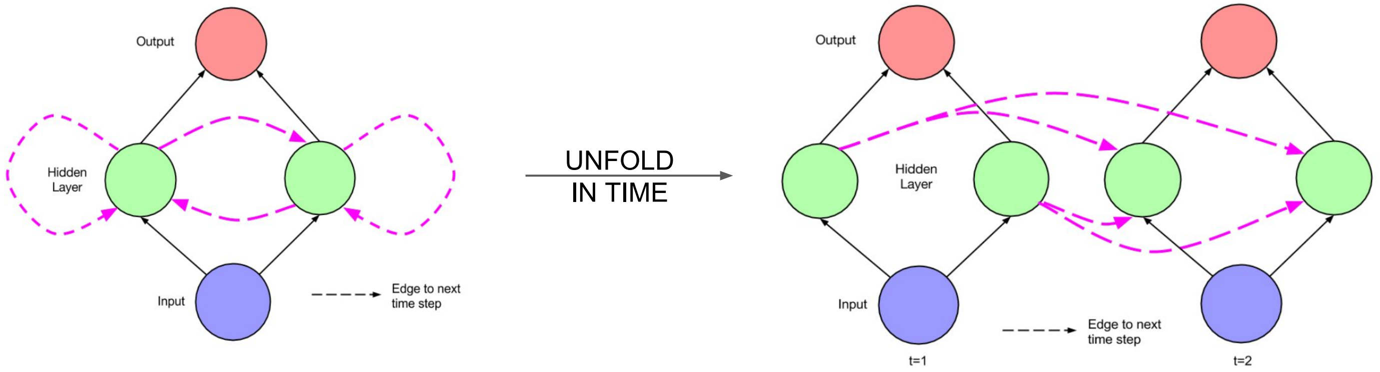
Introduction

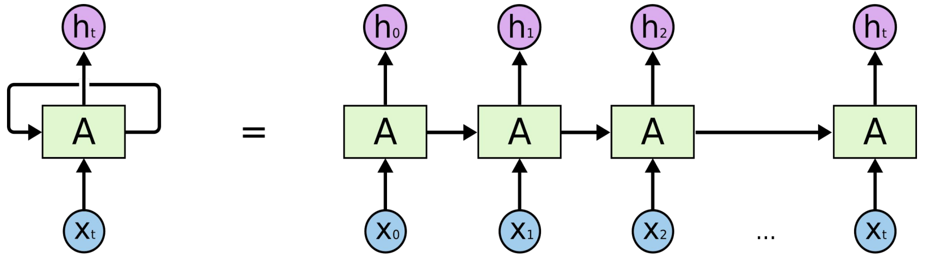
- Feedforward neural networks



Recurrent Neural Networks (RNNs)

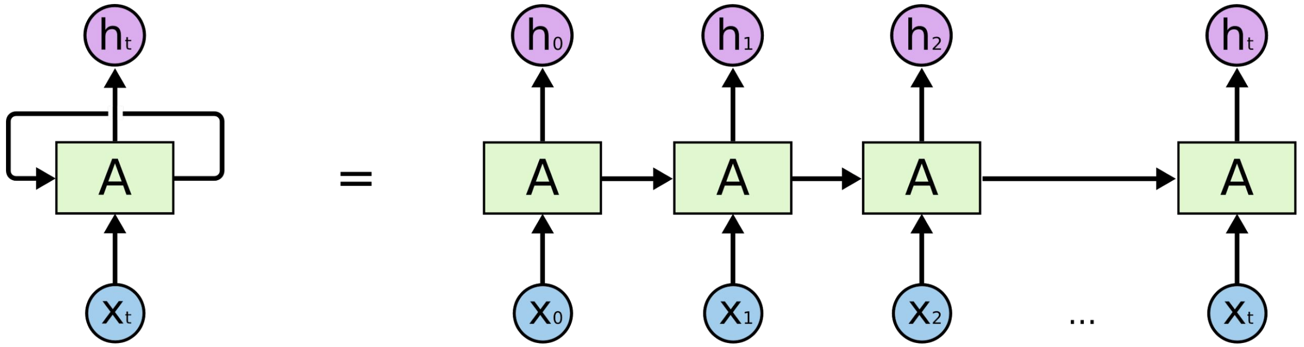
- Networks with feedback loops (recurrent edges)
- Output at current time step depends on current input as well as previous state (via recurrent edges)





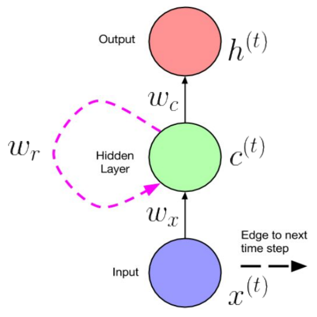
Training RNNs

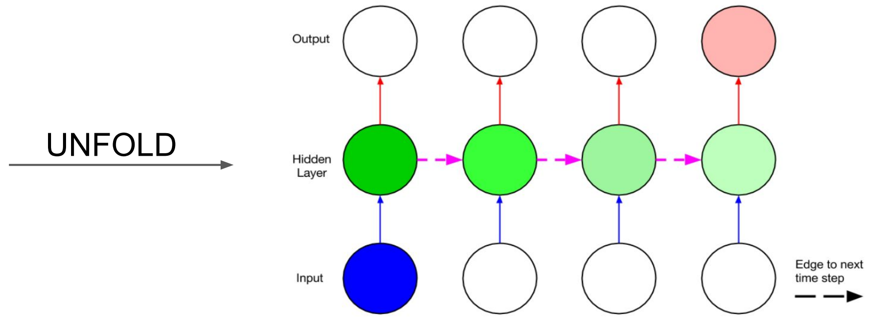
- Backpropagation Through Time (BPTT)
  - Regular (feedforward) backprop applied to RNN unfolded in time
  - Truncated BPTT approximation

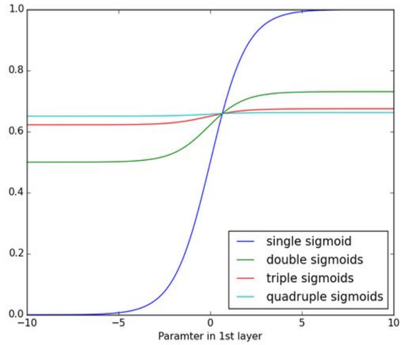


Training RNNs

- Problem: can't capture long-term dependencies due to vanishing/exploding gradients during backpropagation







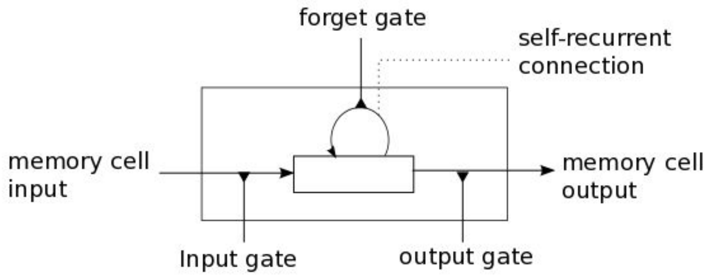
$$h^{(t)} = \sigma(w_c \cdot c^{(t)})$$
$$c^{(t)} = \sigma(w_r \cdot c^{(t-1)} + w_x \cdot x^{(t)})$$

$$\begin{aligned}
h^{(3)} &= \sigma(w_c \cdot c^{(3)}) \\
&= \sigma(w_c \cdot \sigma(w_x \cdot x^{(3)} + w_r \cdot c^{(2)})) \\
&= \sigma(w_c \cdot \sigma(w_x \cdot x^{(3)} + w_r \cdot \sigma(w_x \cdot x^{(2)} + w_r \cdot c^{(1)})))))) \\
&= \sigma(w_c \cdot \sigma(w_x \cdot x^{(3)} + w_r \cdot \sigma(w_x \cdot x^{(2)} + w_r \cdot \sigma(w_x \cdot x^{(1)} + w_r \cdot c^{(0)})))))))
\end{aligned}$$

Long Short-Term Memory networks (LSTMs)

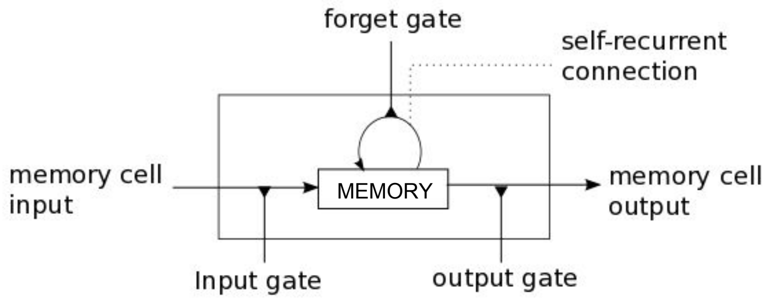
- A type of RNN architecture that addresses the vanishing/exploding gradient problem and allows learning of long-term dependencies

- Recently risen to prominence with state-of-the-art performance in speech recognition, language modeling, translation, image captioning



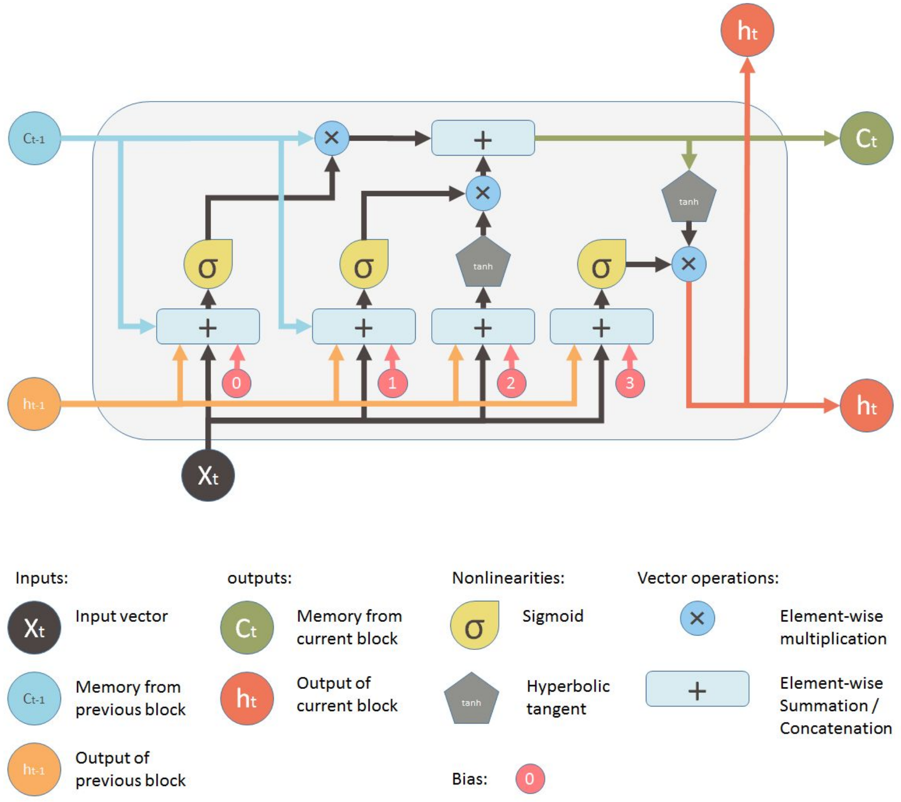
LSTMs

Central Idea: A **memory cell** (interchangeably **block**) which can maintain its state over time, consisting of an explicit memory (aka the **cell state vector**) and **gating units** which regulate the information flow into and out of the memory.

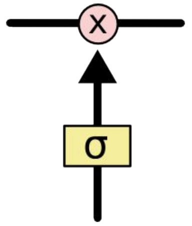


LSTM Memory Cell

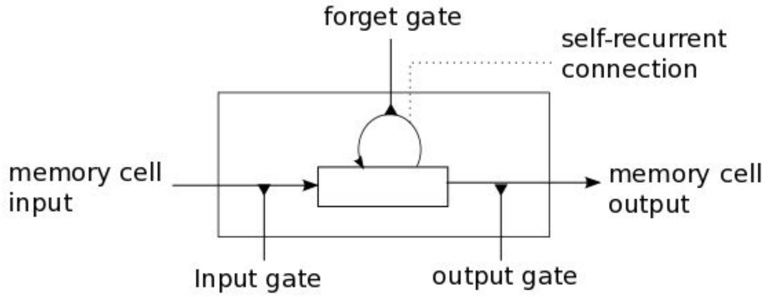
LSTM Memory Cell



ht
Ct-1
Ct
tanh
σ
0
1
2
3
ht-1
ht
Xt
Inputs:
Xt Input vector
Ct-1 Memory from previous block
ht-1 Output of previous block
outputs:
Ct Memory from current block
ht Output of current block
Nonlinearities:
σ Sigmoid
tanh Hyperbolic tangent
Vector operations:
Element-wise multiplication
Element-wise Summation / Concatenation
Bias:
0



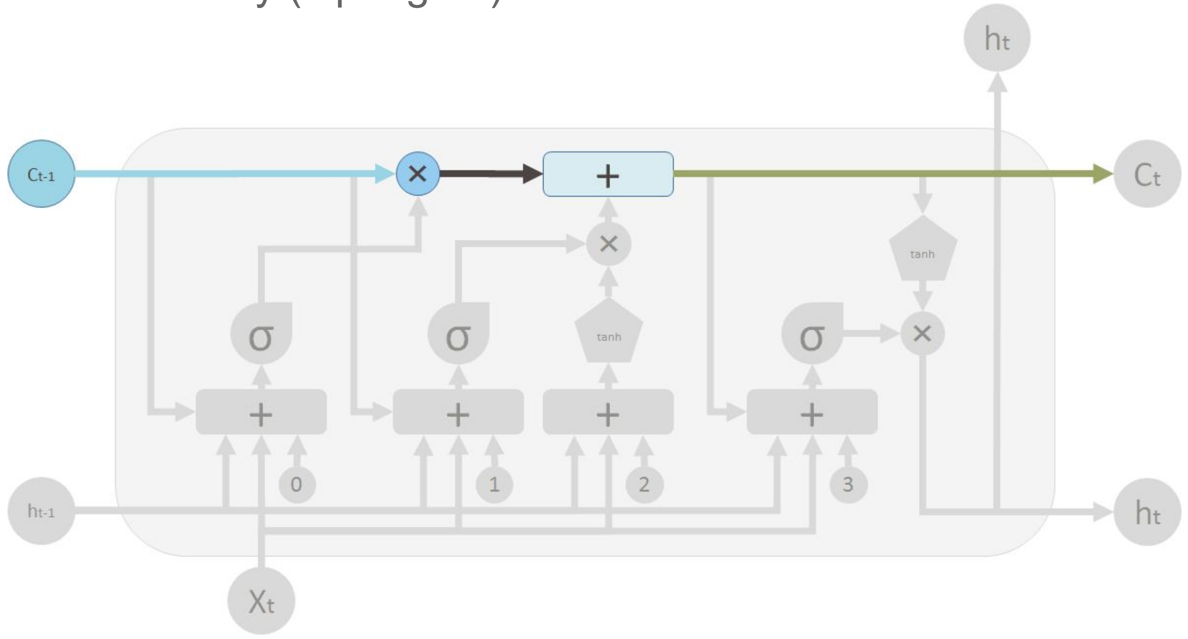
Gate (sigmoid layer followed by pointwise multiplication)

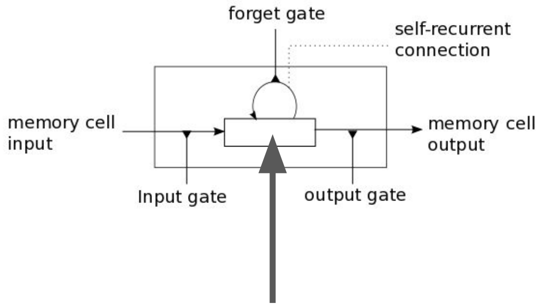


Simplified schematic for reference

Cell state vector

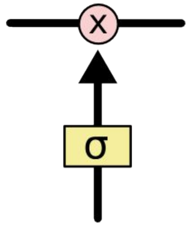
- Represents the memory of the LSTM
- Undergoes changes via forgetting of old memory (forget gate) and addition of new memory (input gate)





Cell state vector

Gates



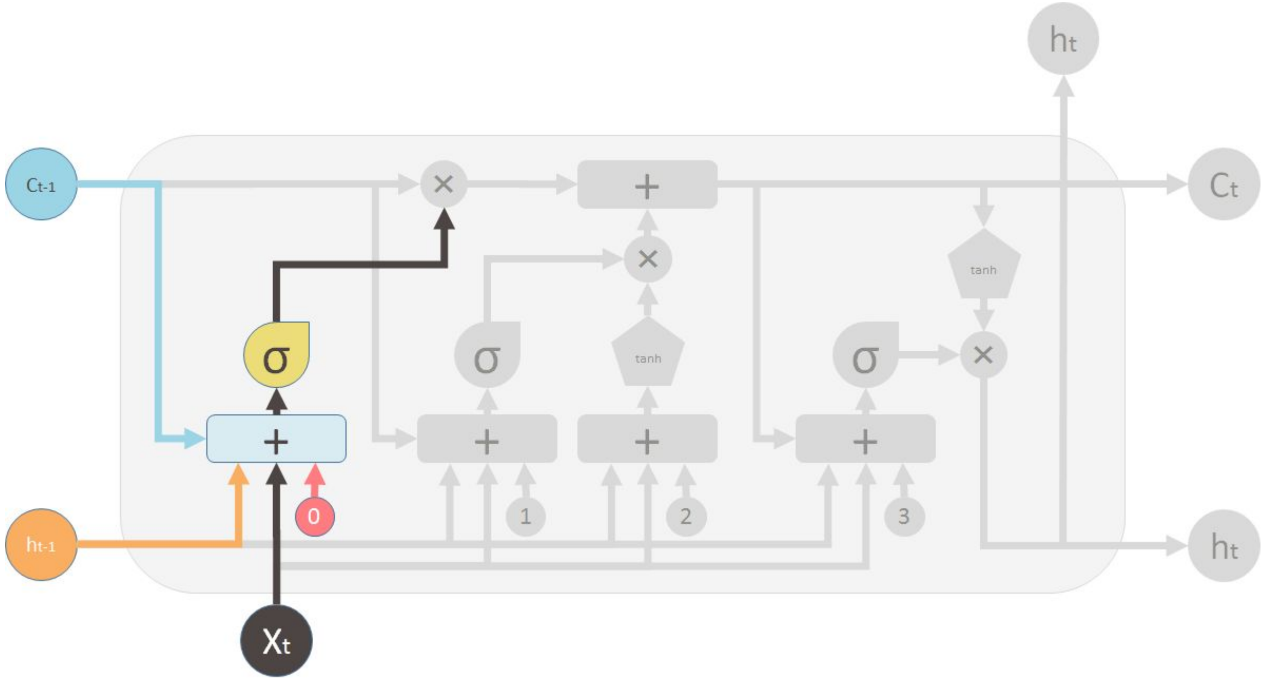
- Gate: sigmoid neural net layer followed by pointwise multiplication operator

- Gates control the flow of information to/from the memory

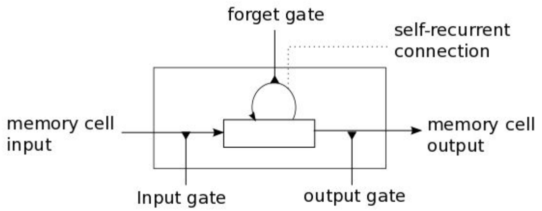
- Gates are controlled by a concatenation of the output from the previous time step and the current input and optionally the cell state vector.

Forget Gate

- Controls what information to throw away from memory

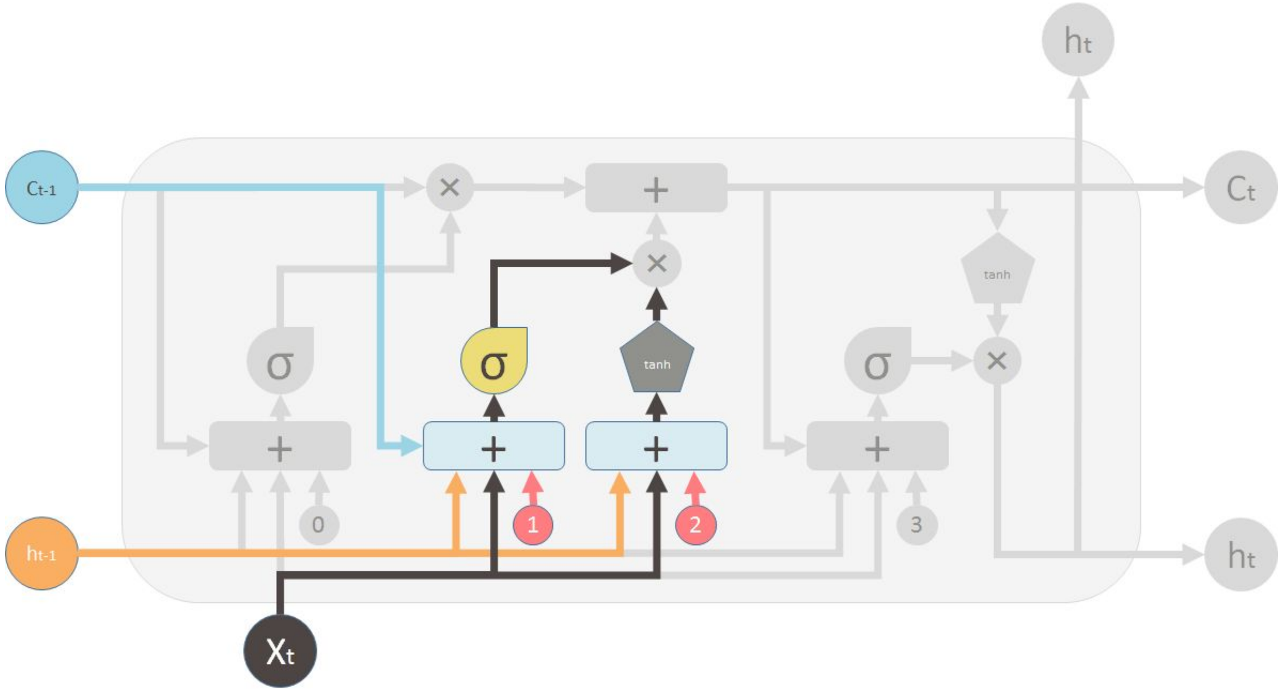


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$

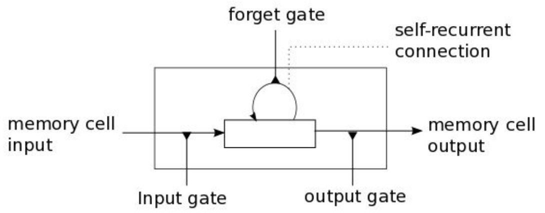


Input Gate

- Controls what new information is added to cell state from current input

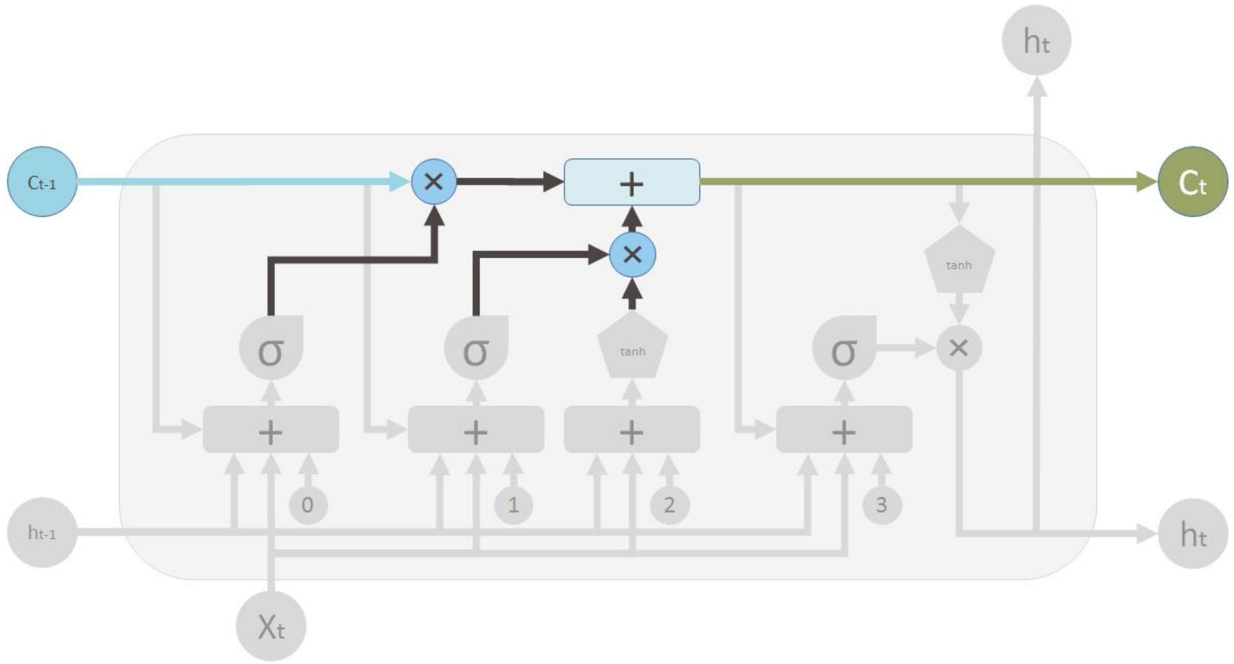


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

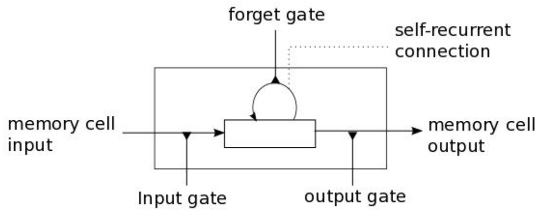


Memory Update

- The cell state vector aggregates the two components (old memory via the forget gate and new memory via the input gate)

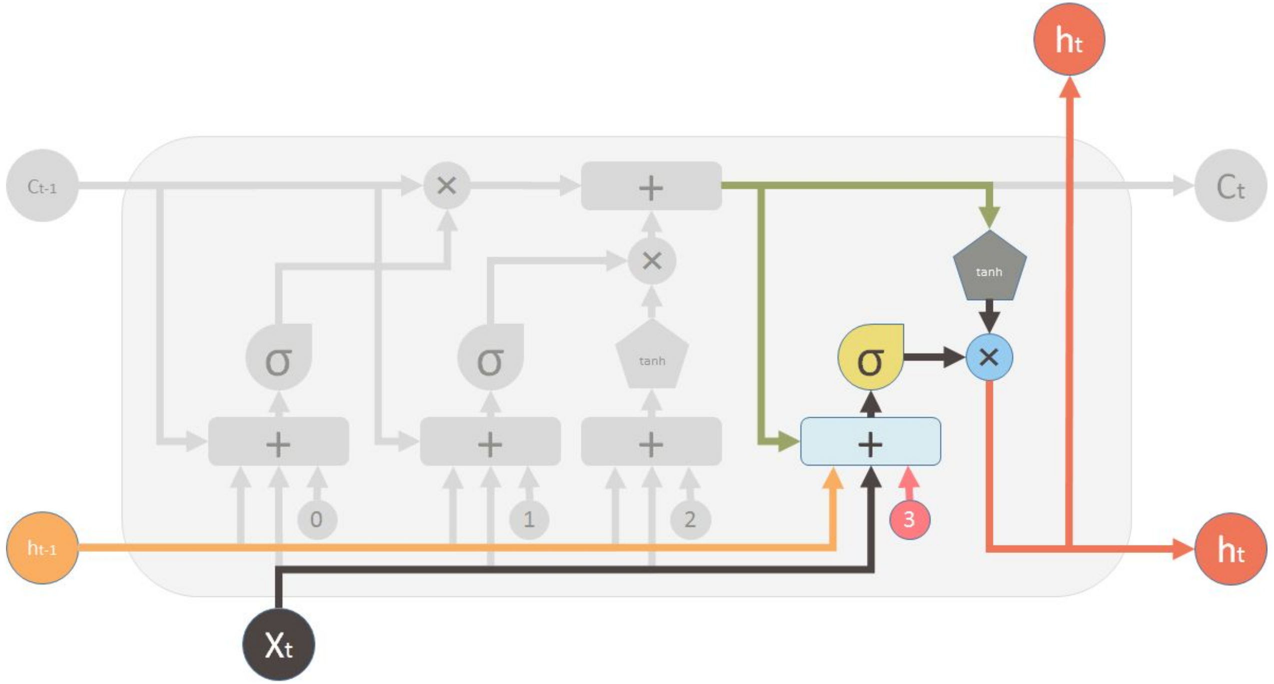


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



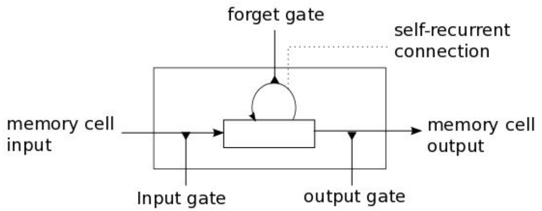
Output Gate

- Conditionally decides what to output from the memory

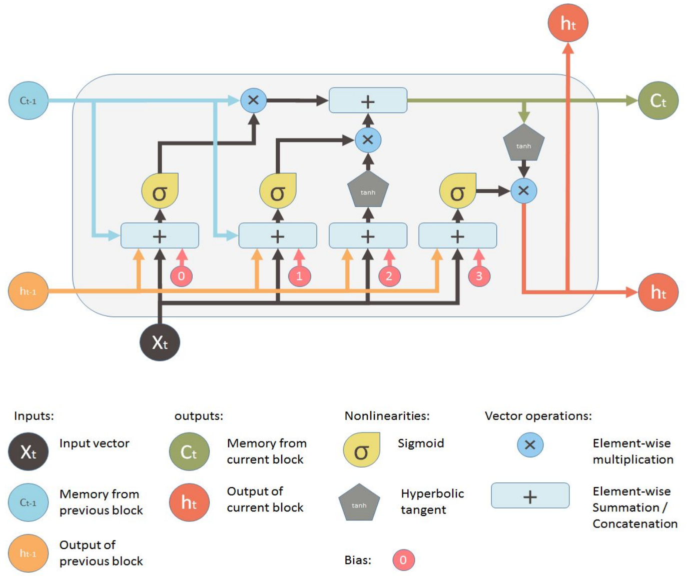


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



LSTM Memory Cell Summary



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

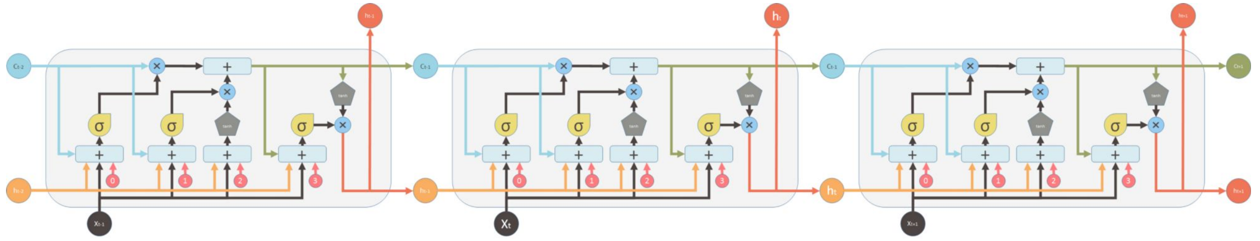
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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

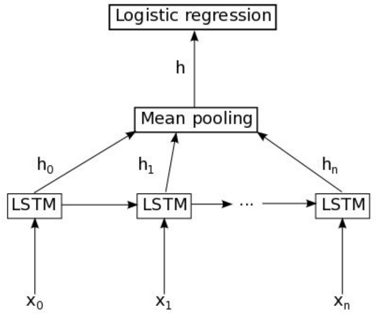
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



LSTM Training

- Backpropagation Through Time (BPTT) most common
- What weights are learned?
  - Gates (input/output/forget)
  - Input tanh layer
- Outputs depend on the task:
  - Single output prediction for the whole sequence (e.g. below)
  - One output at each time step (sequence labeling)

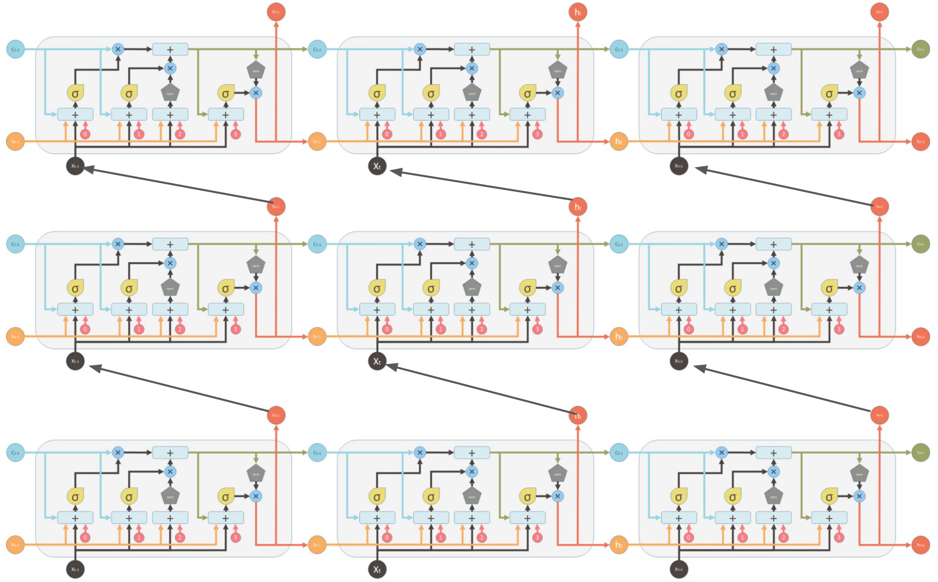


Deep LSTMs

- Deep LSTMs can be created by stacking multiple LSTM layers vertically, with the output sequence of one layer forming the input sequence of the next (in addition to recurrent connections within the same layer)

- Increases the number of parameters - but given sufficient data, performs significantly better than single-layer LSTMs (Graves et al. 2013)

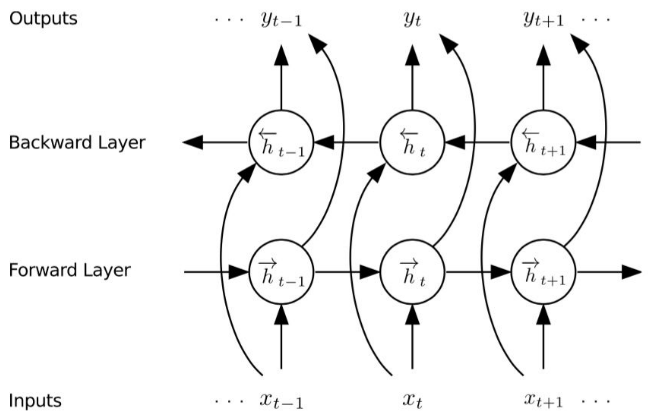
- Dropout usually applied only to non-recurrent edges, including between layers



Bidirectional RNNs

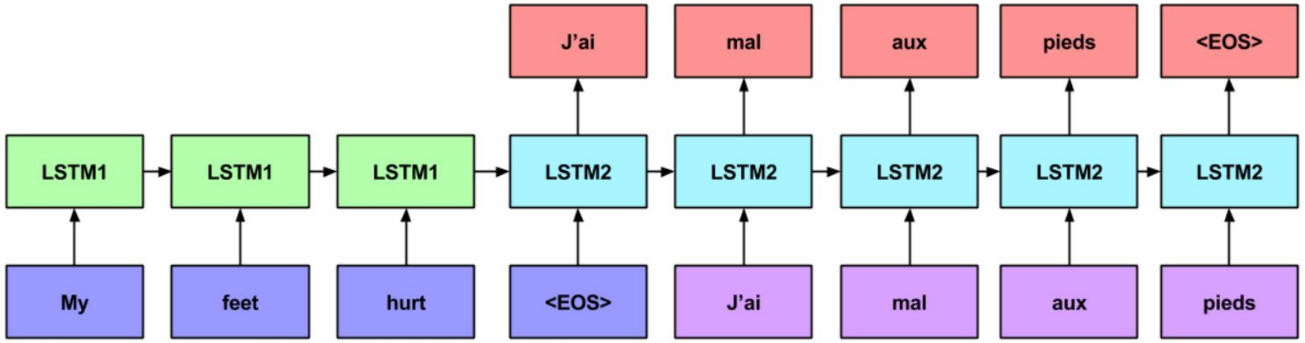
- Data processed in both directions processed with two separate hidden layers, which are then fed forward into the same output layer

- Bidirectional RNNs can better exploit context in both directions, for e.g. bidirectional LSTMs perform better than unidirectional ones in speech recognition (Graves et al. 2013)



LSTMs for Machine Translation (Sutskever et al. 2014)

- Encoder and decoder LSTMs



Demos

- Handwriting generation demo: http://www.cs.toronto.edu/~graves/handwriting.html
- Music composition: http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/
- Image captioning and other stuff: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

References

- Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780. (The original paper on LSTMs; the forget gate was added later)
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (A great blog post introducing LSTMs)
- Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning." (A nice review of RNNs, including LSTMs, bidirectional RNNs and state-of-the-art applications)
- https://deeplearning4j.org/lstm (Another nice introduction to recurrent networks and LSTMs, with code examples - Deeplearning4j is a deep learning platform for Java and Scala)
- Sutskever, I., Vinyals, O., & Le, Q. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information. (A paper that proposes two LSTMs (one for encoding, one for decoding) for machine translation)
- Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. Acoustics, Speech and Signal. (A paper that proposes deep bidirectional LSTMs for speech recognition)
- Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. (Paper introducing image captioning using ConvNet + LSTM)
- https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714 (Neat LSTM explanation diagrams)
- http://deeplearning.net/tutorial/lstm.html (Tutorial applying LSTM to sentiment analysis)
- https://xkcd.com/1093/

Other useful links

- http://deeplearning.net/tutorial/lstm.html
- https://github.com/zhongkaifu/RNNSharp
- http://blog.leanote.com/post/wjgaas@126.com/RNN-and-LSTM-List
- https://deeplearning4j.org/lstm
- https://apaszke.github.io/lstm-explained.html
- https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714