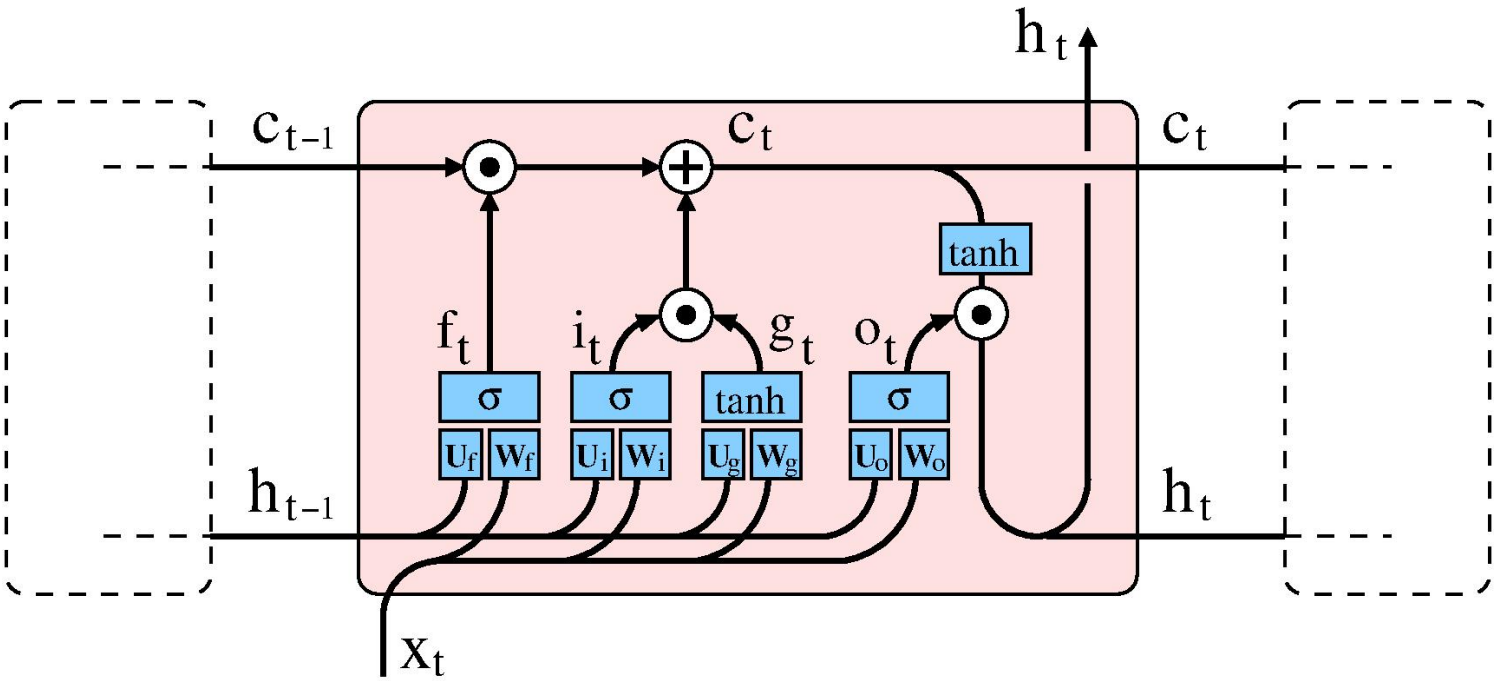
# 4b: Long Short Term Memory

## Long Short Term Memory

Simple Recurrent Networks (SRNs) can learn medium-range dependencies but have difficulty learning long range dependencies. Long Short Term Memory (LSTM) is able to learn long range dependencies using a combination of forget, input and output gates (Hochreiter & Schmidhuber, 1998).




The LSTM maintains a **context** layer which is distinct from the **hidden** layer but contains the same number of units. The full workings of the LSTM at each timestep are described by these equations:

Gates:

$$\mathbf{f}_t = \sigma\left(U_f \mathbf{h}_{t-1} + W_f \mathbf{x}_t + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(U_i\, \mathbf{h}_{t-1} + W_i\, \mathbf{x}_t + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(U_o\, \mathbf{h}_{t-1} + W_o\, \mathbf{x}_t + \mathbf{b}_o\right)$$

Candidate Activation:

$$\mathbf{g}_t = \tanh\left(U_g\, \mathbf{h}_{t-1} + W_g\, \mathbf{x}_t + \mathbf{b}_g\right)$$

State:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

Output:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh \mathbf{c}_t$$

First, the **forget** gate ($f$) is used to determine, for each context unit, a ratio between 0 and 1 by which the value of this context unit will be multiplied. If the ratio is close to zero, the previous value of the corresponding context unit will be largely forgotten; if it is close to 1, the previous value will be largely preserved.

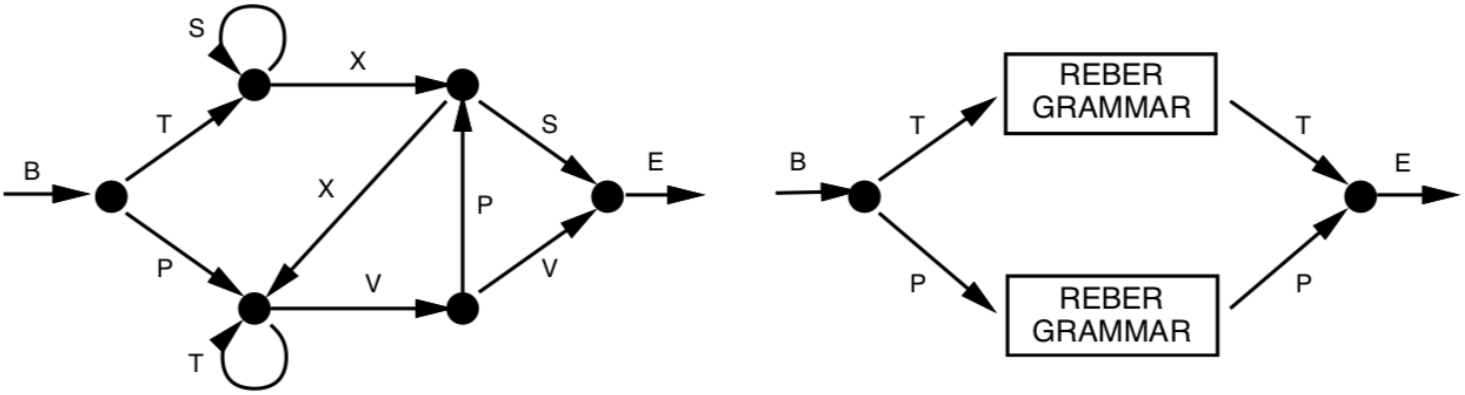
Next, **update** values ($g$) between $-1$ and $+1$ are computed using $\tanh$, and the **input** gate ($i$) is used to determine ratios by which these update values will be multiplied before being added to the current context values.

Finally, the **output** gate ($o$) is computed and used to determine the ratios by which $\tanh$ of the context unit values will be multiplied in order to produce the next **hidden** unit values.

In this way, the context units are able to specialise, with some of them changing their values frequently while others preserve their state for many timesteps, until particular circumstances cause the gates to be 'opened' and allow the value of those units to change.

## Embedded Reber Grammar

The ability of different sequence processing algorithms to learn long range dependencies can be explored using the Reber Grammar and Embedded Reber Grammar.



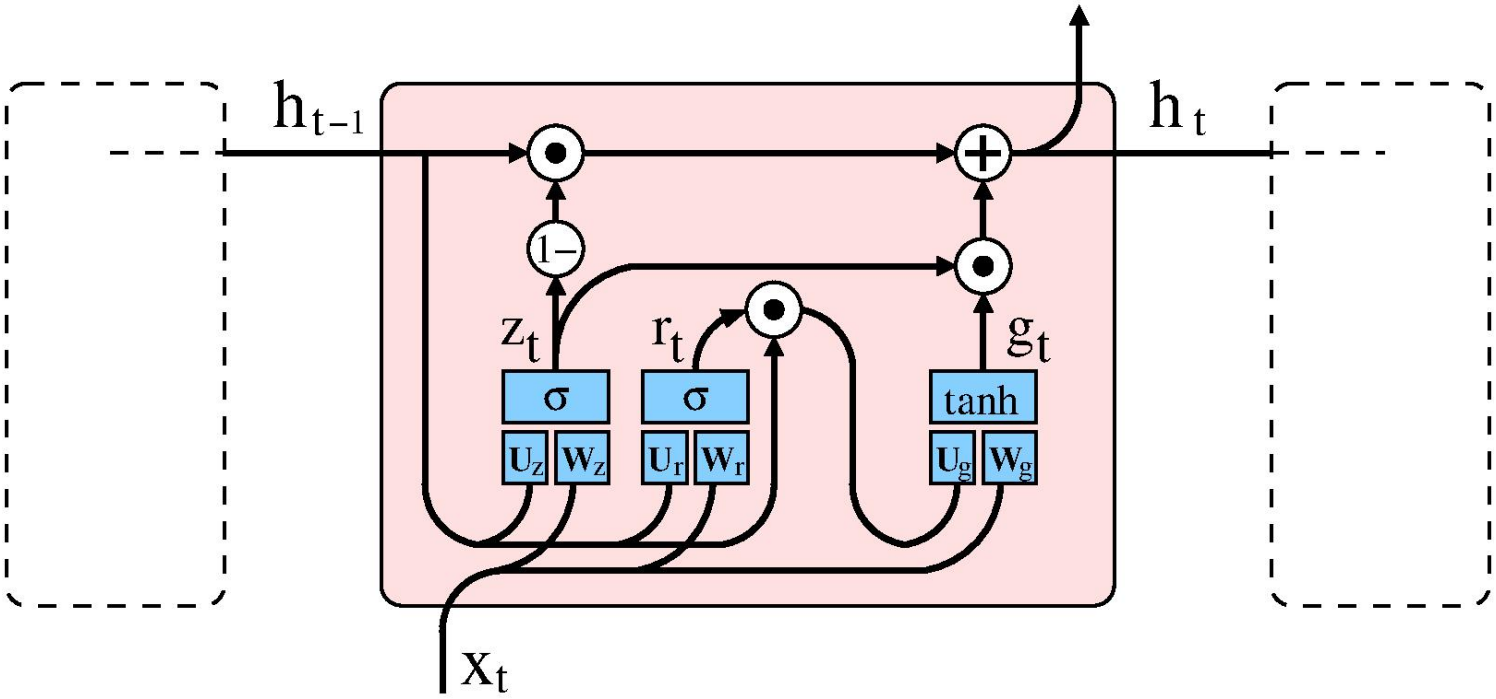

[image source: Fahlman, 1991]

The **Reber Grammar** (RG) is defined by the finite state machine shown on the left. When there is a choice between two transitions, they are understood to be chosen with equal probability. The **Embedded Reber Grammar** (ERG) is shown on the right, where each box marked 'REBER GRAMMAR' contains an identical copy of the finite state machine on the left. The difficulty in learning the ERG is that the network must remember which transition (T or P) occurred after the initial B, and retain this information while it is processing the transitions associated with the RG in one of the two identical boxes, in order to correctly predict the T or P occurring before the final E.

In the exercises for this week, you will be demonstrating that the SRN is able to learn the RG but struggles to learn the ERG, whereas the LSTM can also learn the ERG. We can imagine that one of the context units is somehow assigned the task of retaining the knowledge of the initial T or P, and that

this knowledge is preserved by appropriately high and low values for the forget gate and the input and output gate, respectively.

## Gated Recurrent Unit




The Gated Recurrent Unit (GRU) is similar to LSTM but has only two gates instead of three. Its update equations are as follows:

Gates:

$$\mathbf{z}_t = \sigma\left(U_z \mathbf{h}_{t-1} + W_z \mathbf{x}_t + \mathbf{b}_z\right)$$
$$\mathbf{r}_t = \sigma\left(U_r \mathbf{h}_{t-1} + W_r \mathbf{x}_t + \mathbf{b}_r\right)$$

Candidate activation:

$$\mathbf{g}_t = \tanh\left(U_g \left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right) + W_g \, \mathbf{x}_t + \mathbf{b}_g\right)$$

Output:

$$\mathbf{h}_t = \left(1 - \mathbf{z}_t\right) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \mathbf{g}_t$$

## Optional video

## References

Fahlman, S. E. (1991). *The recurrent cascade-correlation architecture* (Technical Report CMU-CS-91-100). Carnegie-Mellon University, Department of Computer Science.

Hochreiter, S., & Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, *9*(8), 1735-1780.

## Further Reading:

Two excellent web resources for LSTM:

- Understanding LSTM Networks (Colah, 2015. Github)
- LSTM (Long Short Term Memory) (Huerta, n.d. christianherta.de)
- Use of Reber Grammar

# Revision 6: Recurrent Networks and LSTM

This is a revision quiz to test your understanding of the material from Week 4 on Recurrent Networks and LSTM.

You must attempt to answer each question yourself, before looking at the sample answer.

**Question 1** *Submitted Oct 8th 2022 at 8:26:54 pm*

Explain the format and method by which input was fed to the NetTalk system, and the target output.


q1.png

**Question 2** *Submitted Oct 8th 2022 at 8:24:20 pm*

Explain the role of the context layer in an Elman network.

asdf

**Question 3** *Submitted Oct 8th 2022 at 8:24:38 pm*

Draw a diagram of an LSTM and write the equations for its operation.

asdf

**Question 4** *Submitted Oct 8th 2022 at 8:25:19 pm*

Draw a diagram of a Gated Recurrent Unit and write the equitions for its operation.

asdf

**Question 5** *Submitted Oct 8th 2022 at 8:25:41 pm*

Briefly describe the problem of *long range dependencies*, and discuss how well each of the following architectures is able to deal with long range dependencies:

- sliding window approach
- Simple Recurrent (Elman) Network
- Long Short Term Memory (LSTM)

- Gated Recurrent Unit (GRU)

asdf

# Coding: SRN and LSTM trained on Reber Grammar

In this exercise, we will test the ability of **SRN** and **LSTM** to learn the Reber Grammar (RG) and Embedded Reber Grammar (ERG). Specifically, we will test the following:

- Train an SRN on the Reber Grammar
- Train an SRN on the Embedded Reber Grammar
- Train an LSTM on the ERG
- We'll have one more setting where we train the model only on sequences that exceed a specified minimum length.