Deep Learning

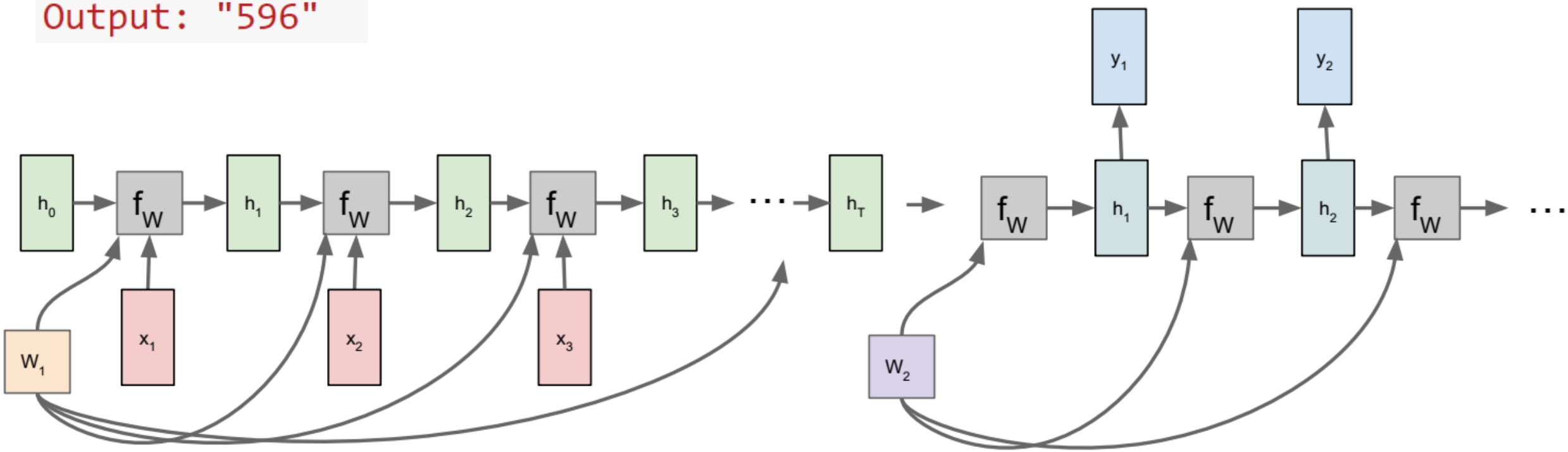
Mohammad Reza Mohammadi

2021

Sequence to Sequence

- Example: train a model to learn to add two numbers, provided as strings

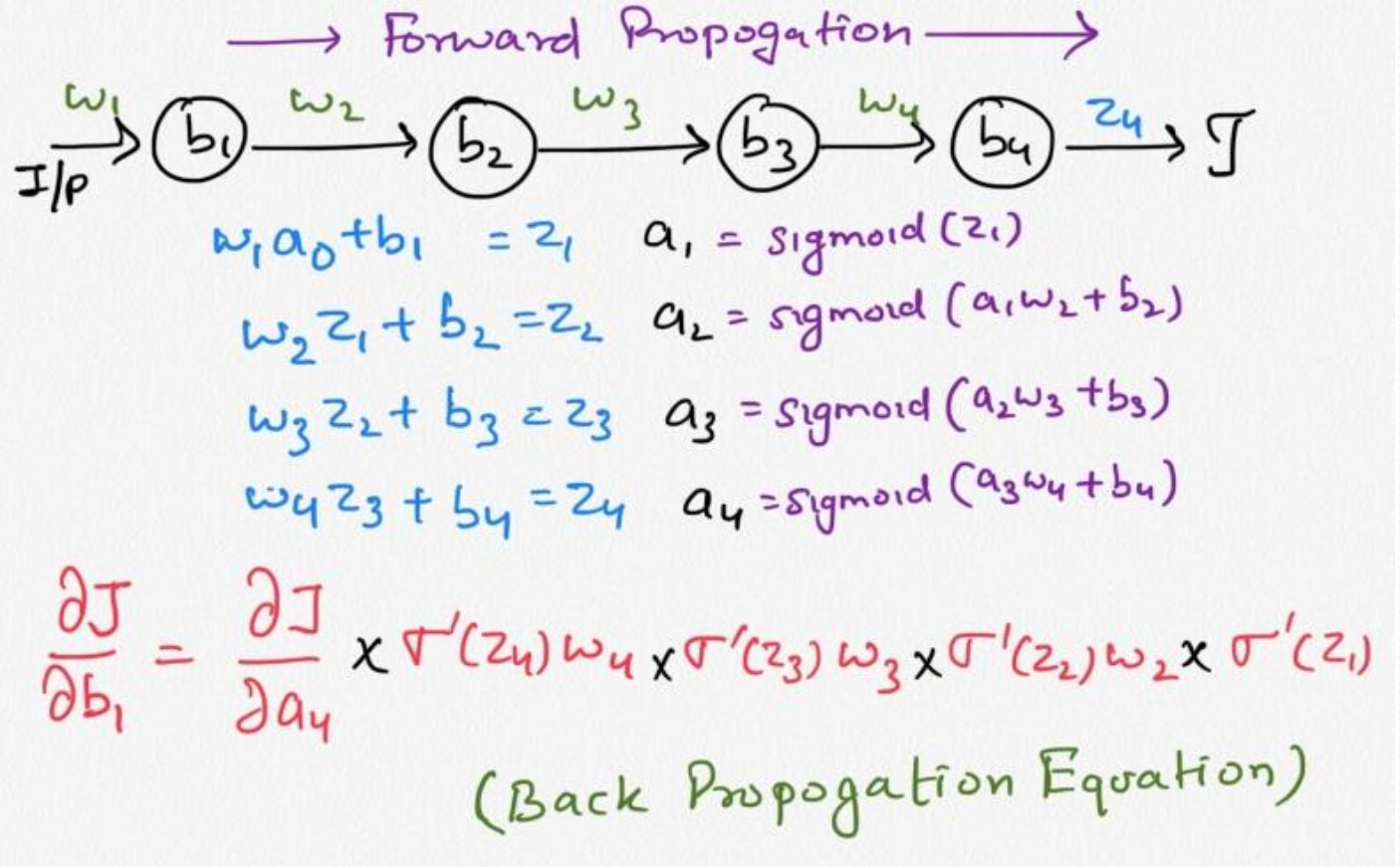
```
Input: "535+61"
Output: "596"
```



https://colab.research.google.com/drive/11qFxWP5ukCOKXXcP2jTBYEKRl3ThBYH6?usp=sharing

Vanishing and exploding gradients

- We can avoid exploding gradients with gradient clipping

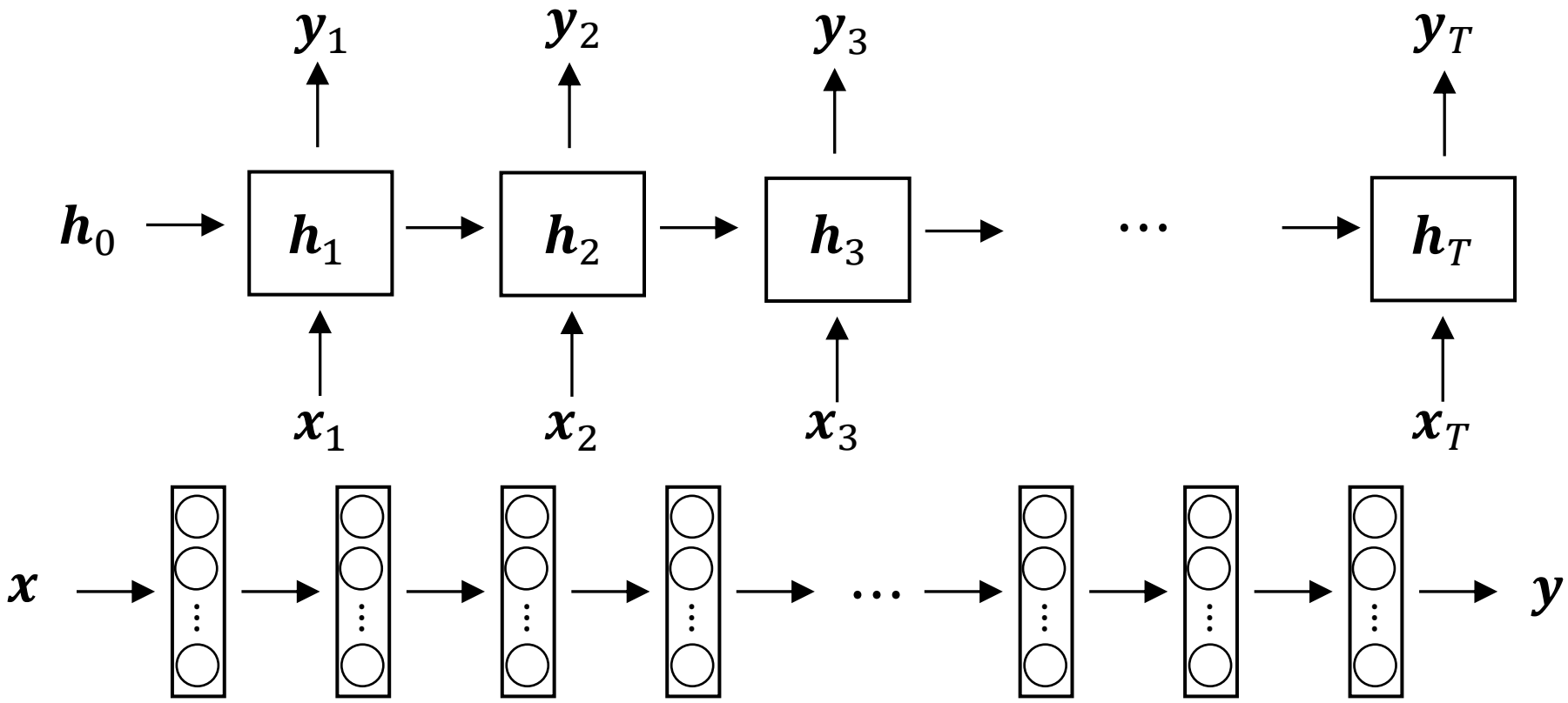


https://realelectricwizard.wordpress.com/2019/03/30/the-mystery-of-vanishing-and-exploding-gradients/

Vanishing gradients with RNNs

The cat, which already ate …, was full.

The cats, which already ate …, were full.

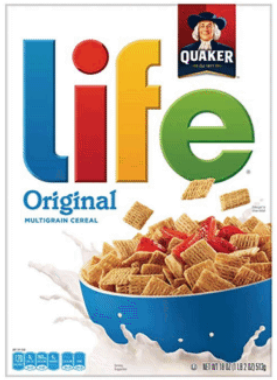


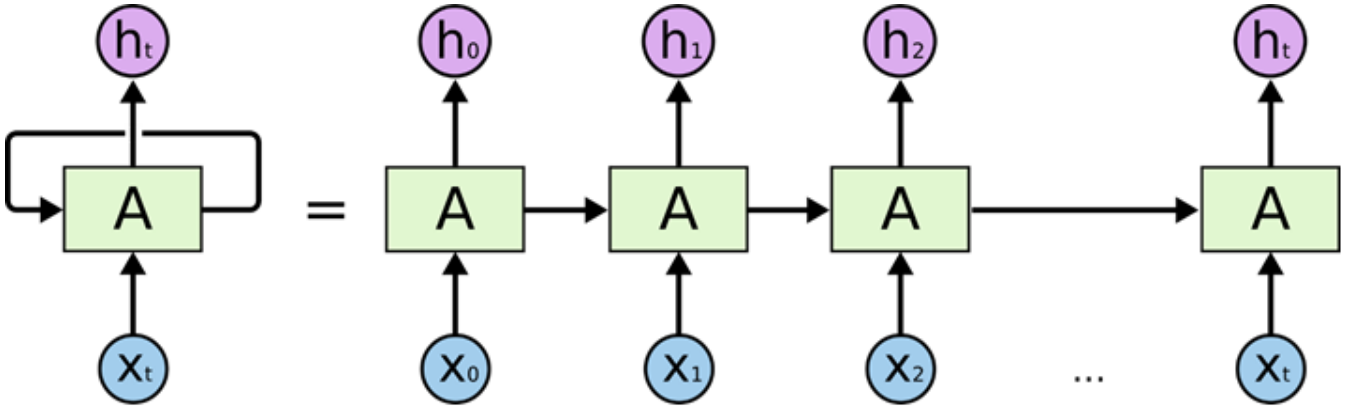
Gated RNNs

- Recurrent Neural Networks suffer from short-term memory
  - RNN's may leave out important information from the beginning
- Gated RNNs are based on the idea of creating paths through time that have derivatives that neither vanish nor explode

The cat, which already ate …, was full.

The cats, which already ate …, were full.

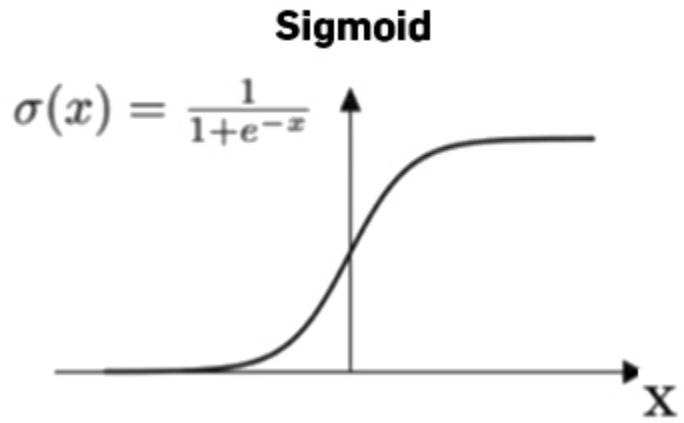




Gated Recurrent Units

Simple RNN

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$



GRU (simplified)

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)} \,.\, \tilde{\boldsymbol{h}}^{(t)} + \left(1 - \boldsymbol{u}^{(t)}\right).\, \boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

The cat, which already ate …, was full.

Gated Recurrent Units

GRU

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\left(\boldsymbol{r}^{(t)}.\boldsymbol{h}^{(t-1)}\right) + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{h}}^{(t)} + \left(1 - \boldsymbol{u}^{(t)}\right).\boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_{hr}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xr}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

GRU (simplified)

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{h}}^{(t)} + \left(1 - \boldsymbol{u}^{(t)}\right).\boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

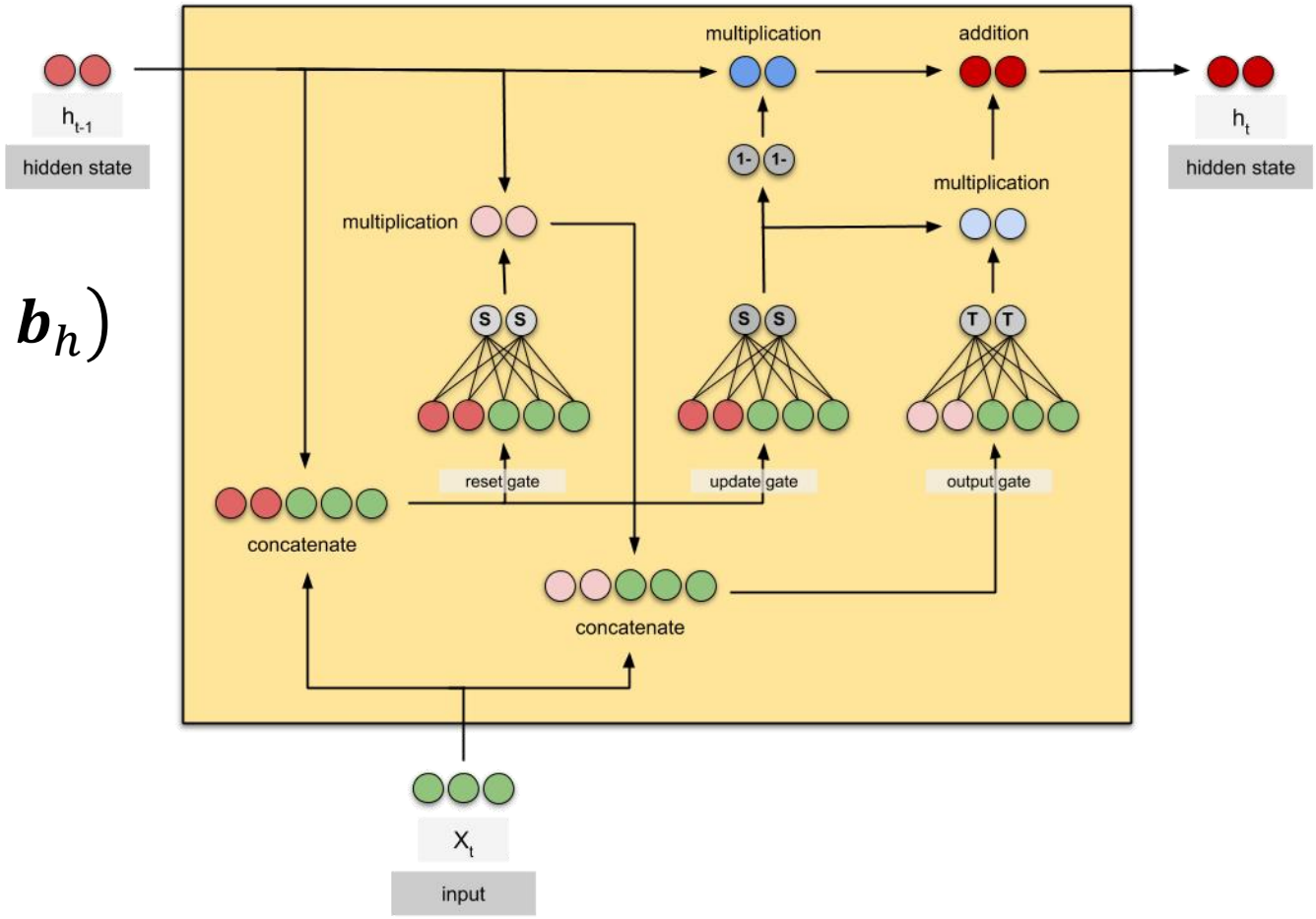
GRU

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\left(\boldsymbol{r}^{(t)}.\boldsymbol{h}^{(t-1)}\right) + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{h}}^{(t)} + \left(1 - \boldsymbol{u}^{(t)}\right).\boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_{hr}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xr}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$



Long Short-Term Memory

GRU

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_{hh}\left(\boldsymbol{r}^{(t)}.\boldsymbol{h}^{(t-1)}\right) + \boldsymbol{W}_{xh}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{h}}^{(t)} + \left(1 - \boldsymbol{u}^{(t)}\right).\boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_{hr}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xr}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

LSTM

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)}.\tanh\left(\boldsymbol{c}^{(t)}\right)$$

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_{hc}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xc}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{c}}^{(t)} + \boldsymbol{f}^{(t)}.\boldsymbol{c}^{(t-1)}$$

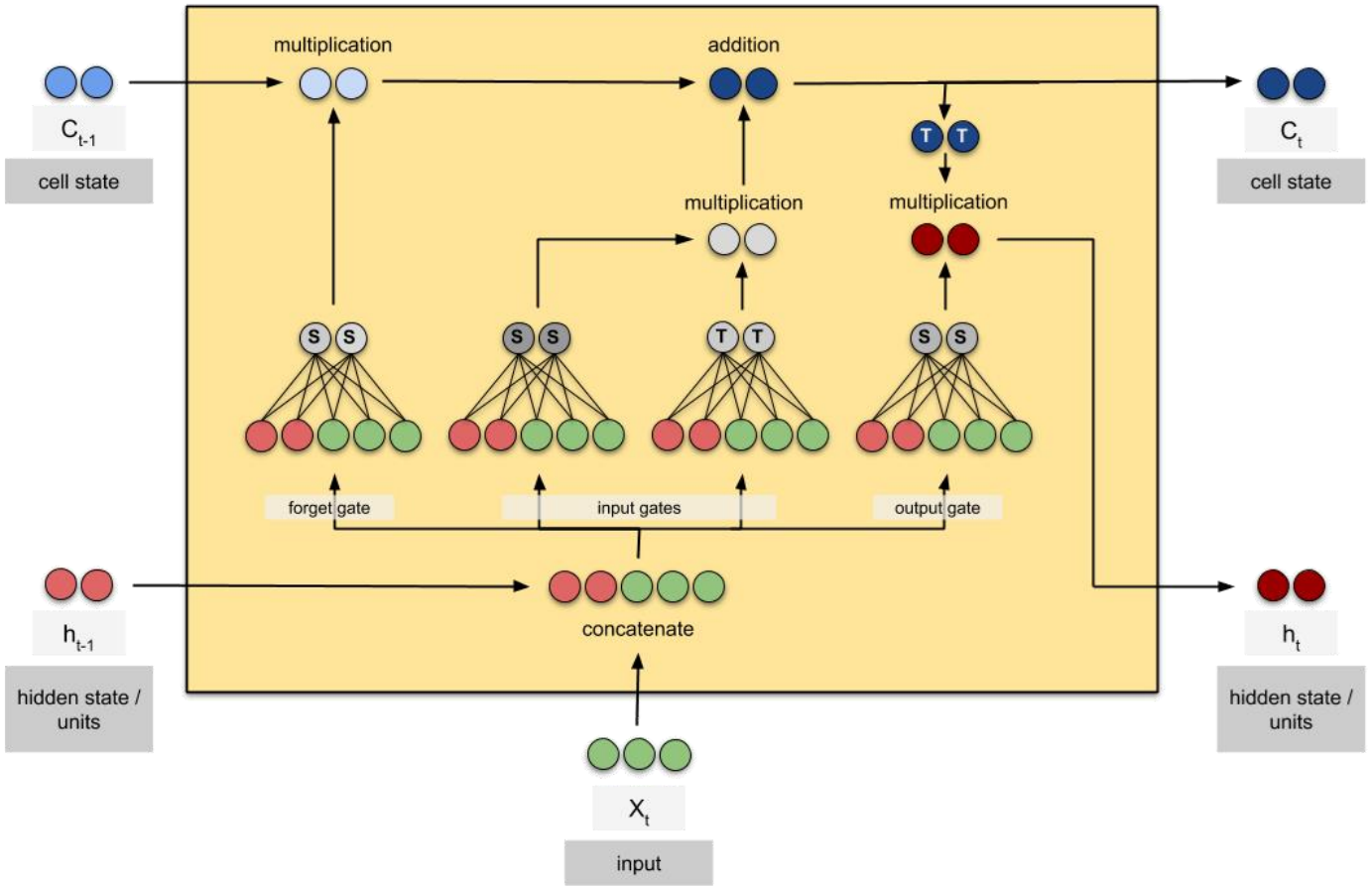
$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_{hf}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xf}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_{ho}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xo}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

LSTM



$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)}.\tanh\left(\boldsymbol{c}^{(t)}\right)$$

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_{hc}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xc}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{u}^{(t)}.\tilde{\boldsymbol{c}}^{(t)} + \boldsymbol{f}^{(t)}.\boldsymbol{c}^{(t-1)}$$

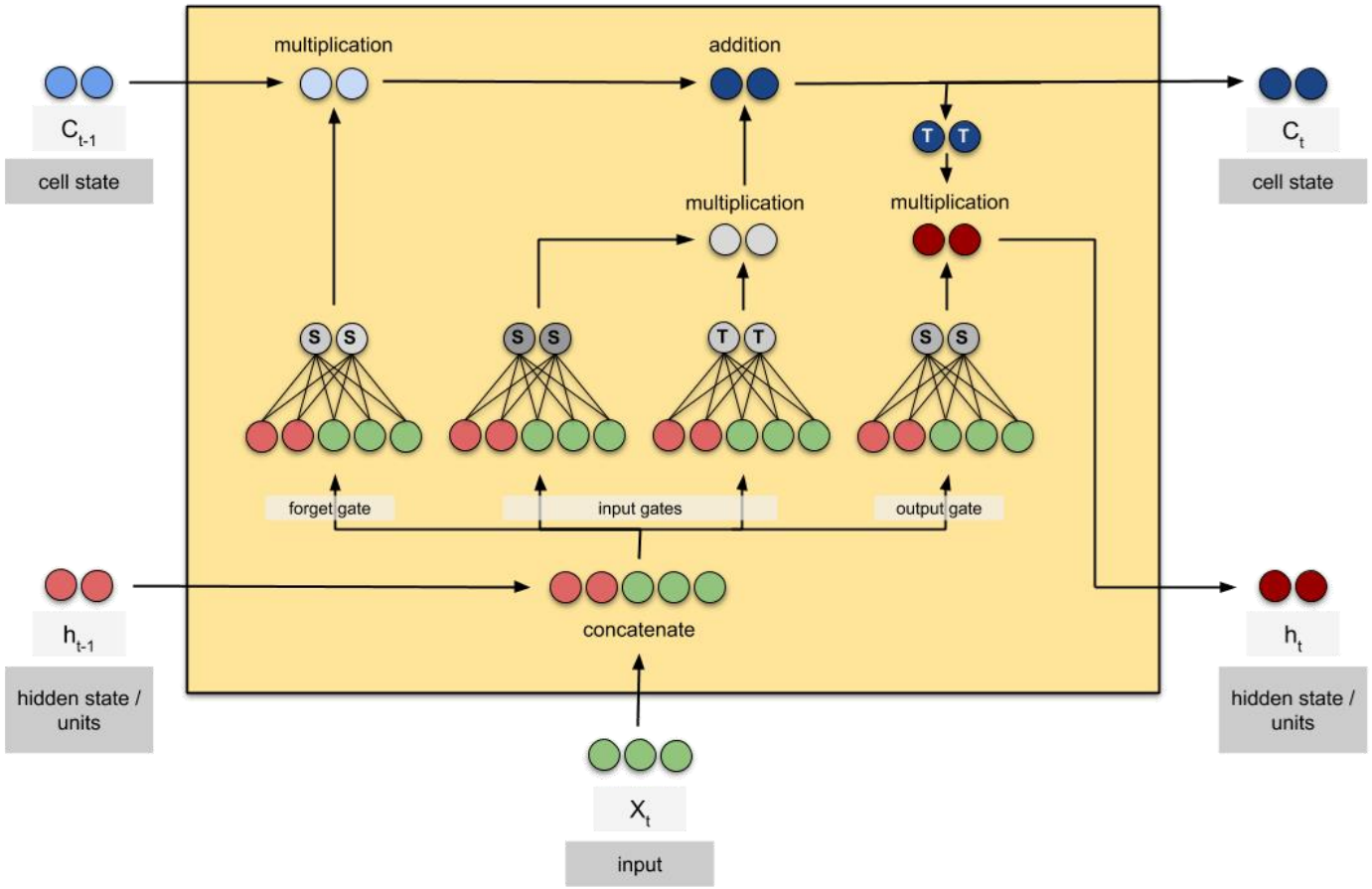
$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_{hu}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xu}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_{hf}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xf}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_{ho}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{xo}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

LSTM



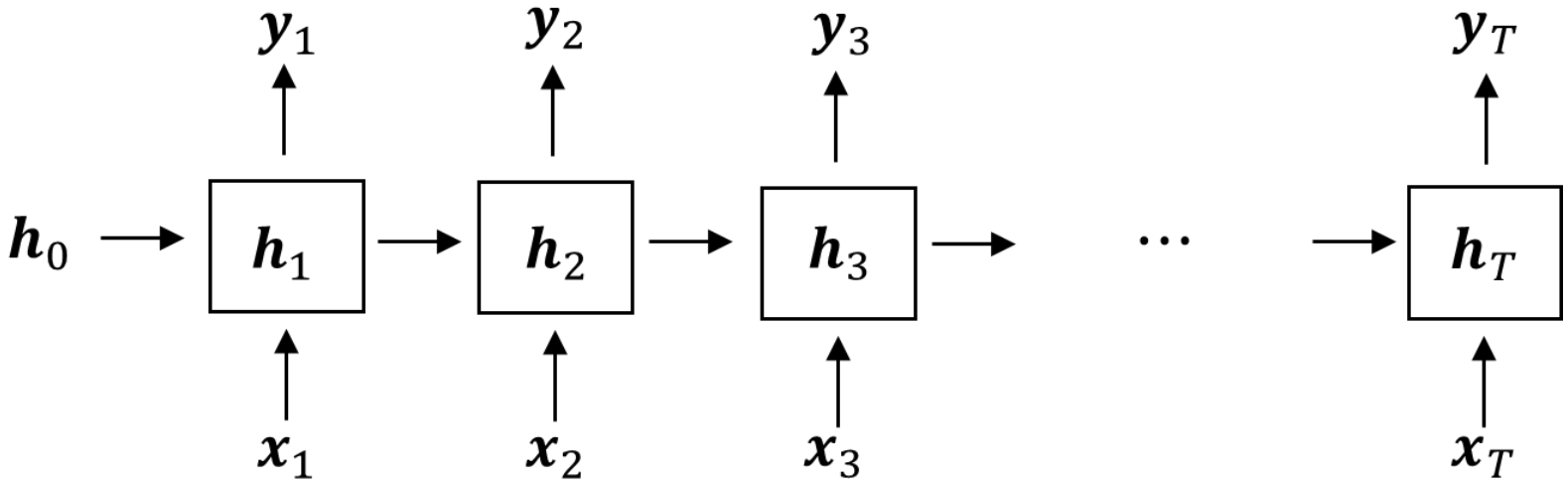

```
def LSTMCELL(prev_ct, prev_ht, input):
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate * it
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct


ct = [0, 0, 0]
ht = [0, 0, 0]
for input in inputs:
    ct, ht = LSTMCELL(ct, ht, input)
```

https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

Bidirectional RNNs

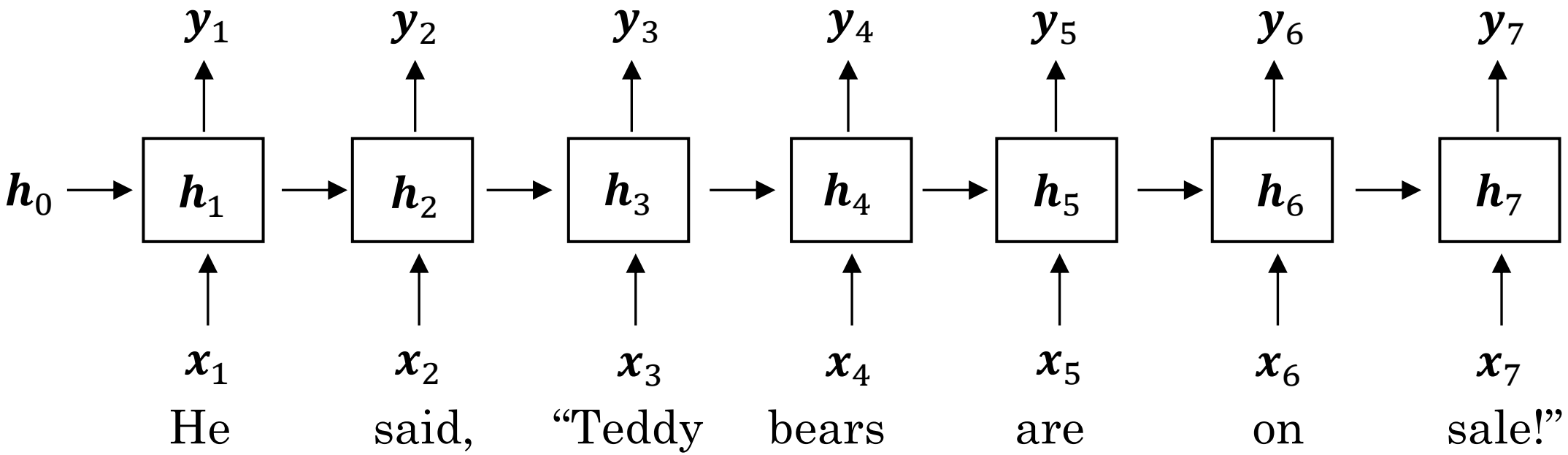
- All of the RNNs we have considered up to now have a "causal" structure
  - The state at time t only captures information from the past, $x(1), \ldots, x(t-1)$, and the present input $x(t)$
- In some applications, we want to output a prediction of $y(t)$ which may depend on the whole input sequence



Bidirectional RNNs

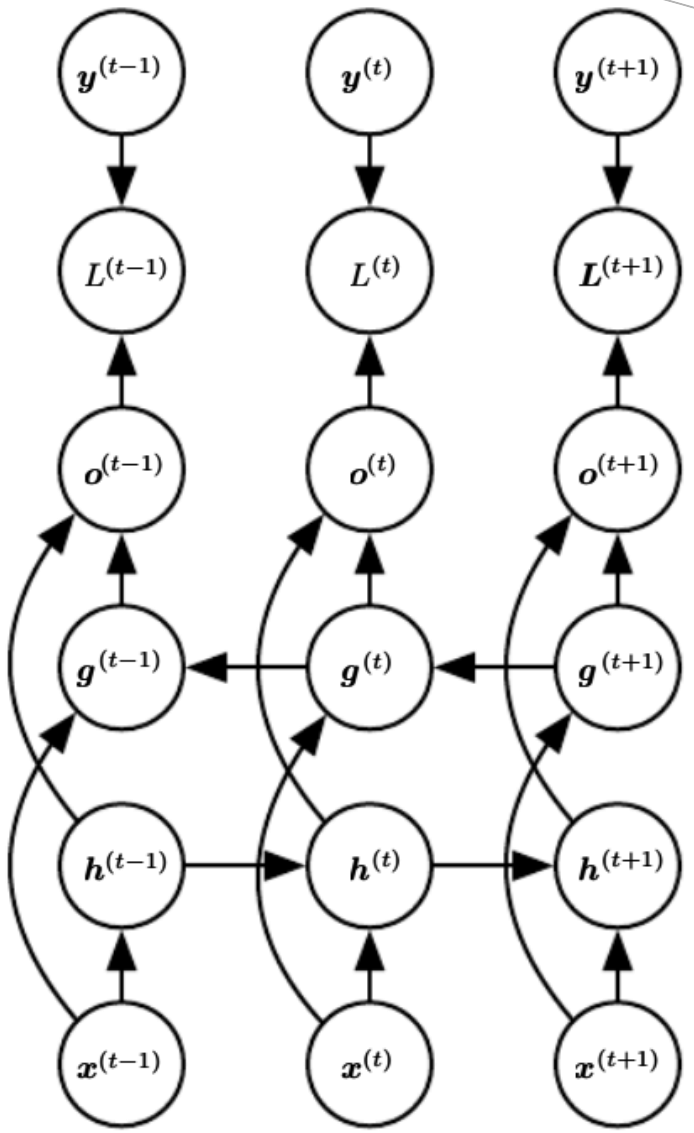
He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



Bidirectional RNNs

- Bidirectional RNNs combine an RNN that moves forward through time beginning from the start of the sequence with another RNN that moves backward through time beginning from the end of the sequence
  - $h(t)$ stands for the state of the sub-RNN that moves forward through time
  - $g(t)$ stands for the state of the sub-RNN that moves backward through time
  - $o(t)$ depends on both the past and the future but is most sensitive to the input values around time t

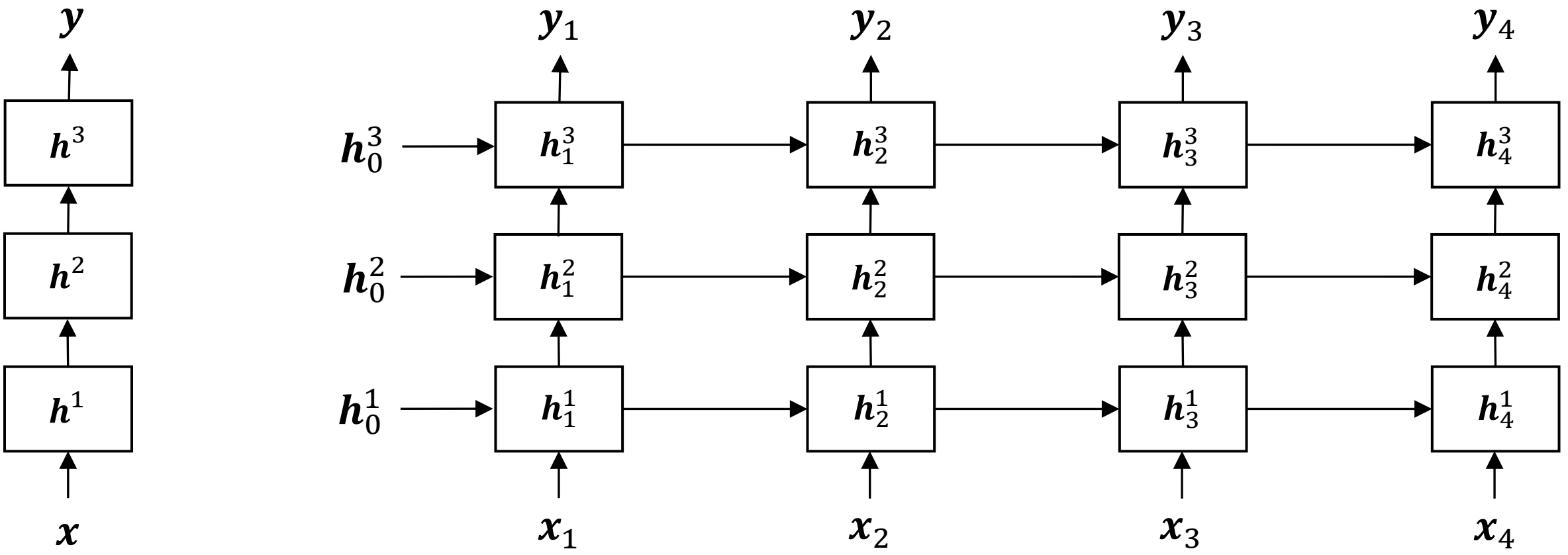


Deep Recurrent Networks

- For learning very complex functions sometimes is useful to stack multiple layers of RNNs together to build even deeper versions of these models

Deep Recurrent Networks

$$h_3^2 = \tanh\left(W^2 \left[h_2^2 \quad h_3^1\right] + b^2\right)$$



Deep Recurrent Networks

- The same parameters $\boldsymbol{W^2}$ and $\boldsymbol{b^2}$ are used for every one of the computations at this layer
- The first layer would have its own parameters and so on
- For RNNs, having three layers is already quite a lot.
- The blocks can be SimpleRNN, GRU, or LSTM
- They can be bidirectional