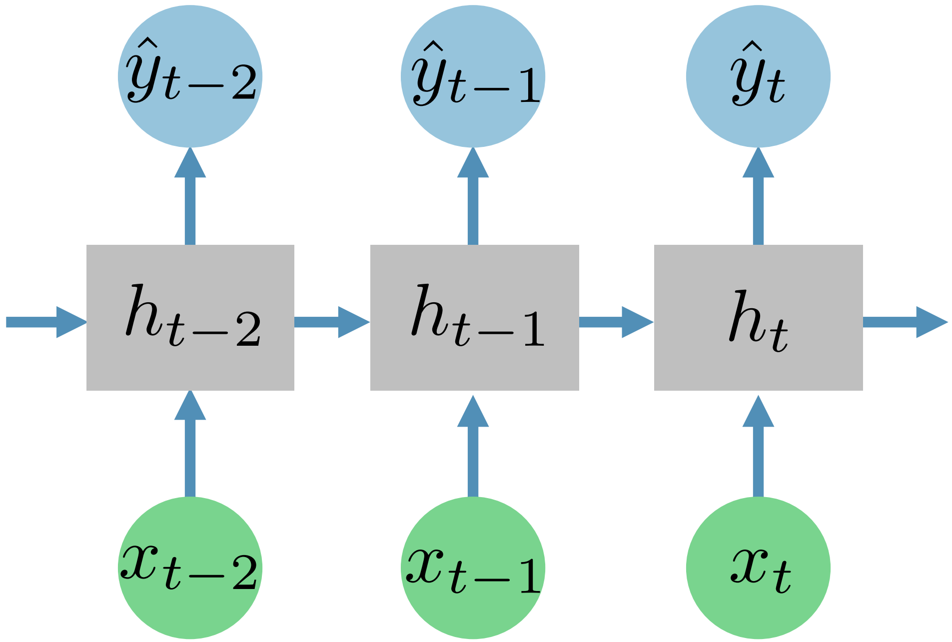
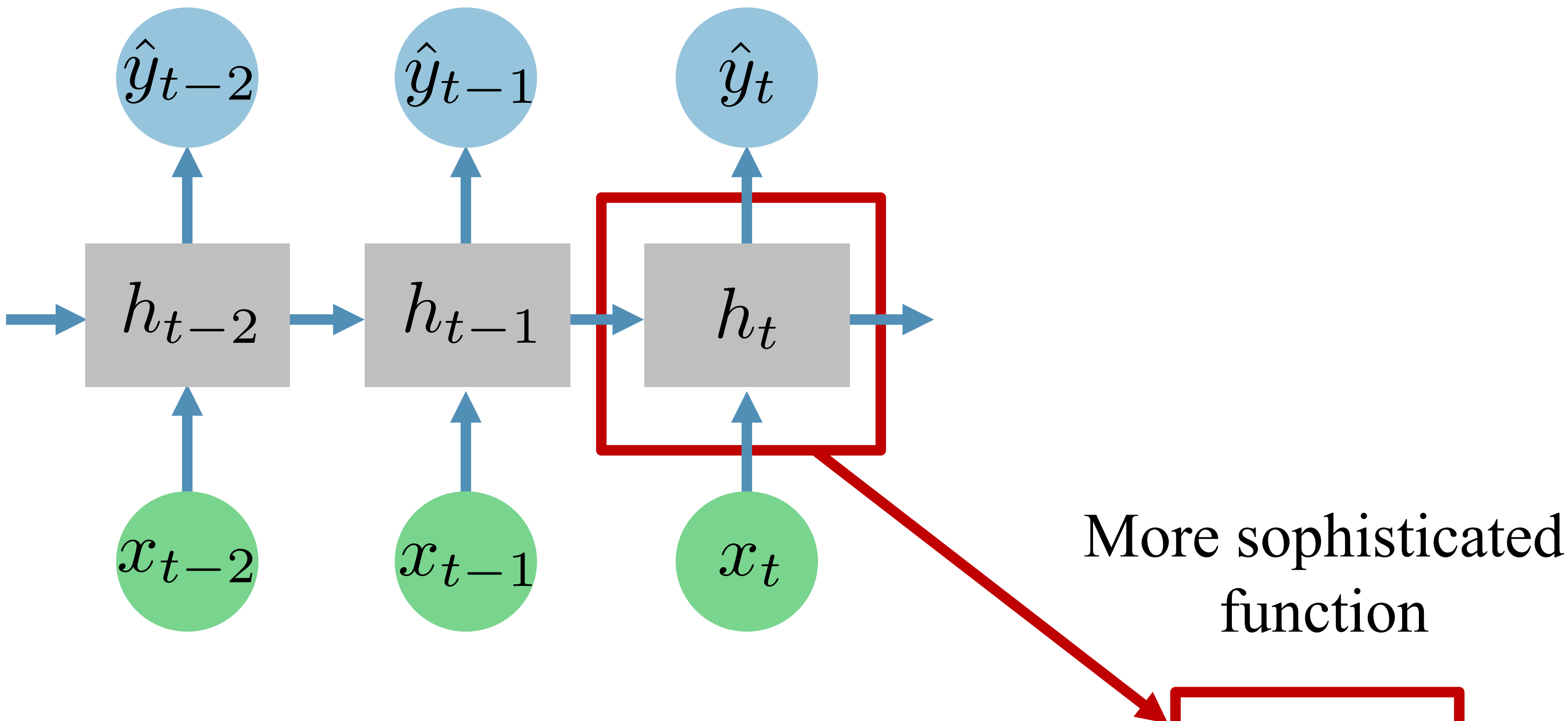
LSTM and GRU

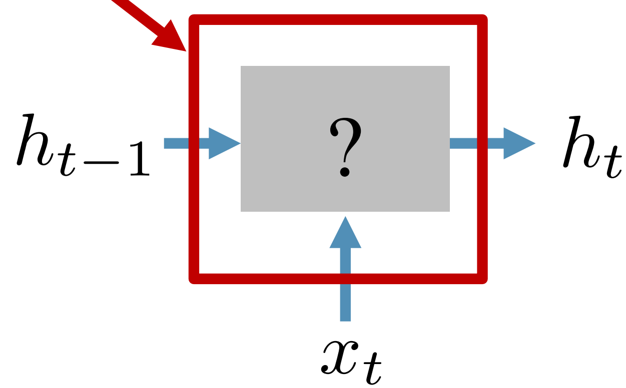
Previously on this week: Simple RNN



$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

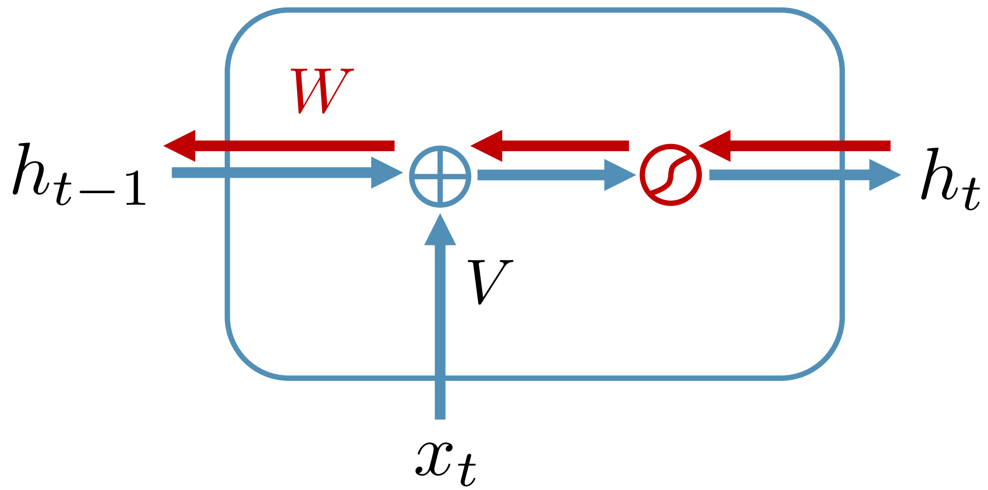
Previously on this week: Simple RNN





$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

Simple RNN



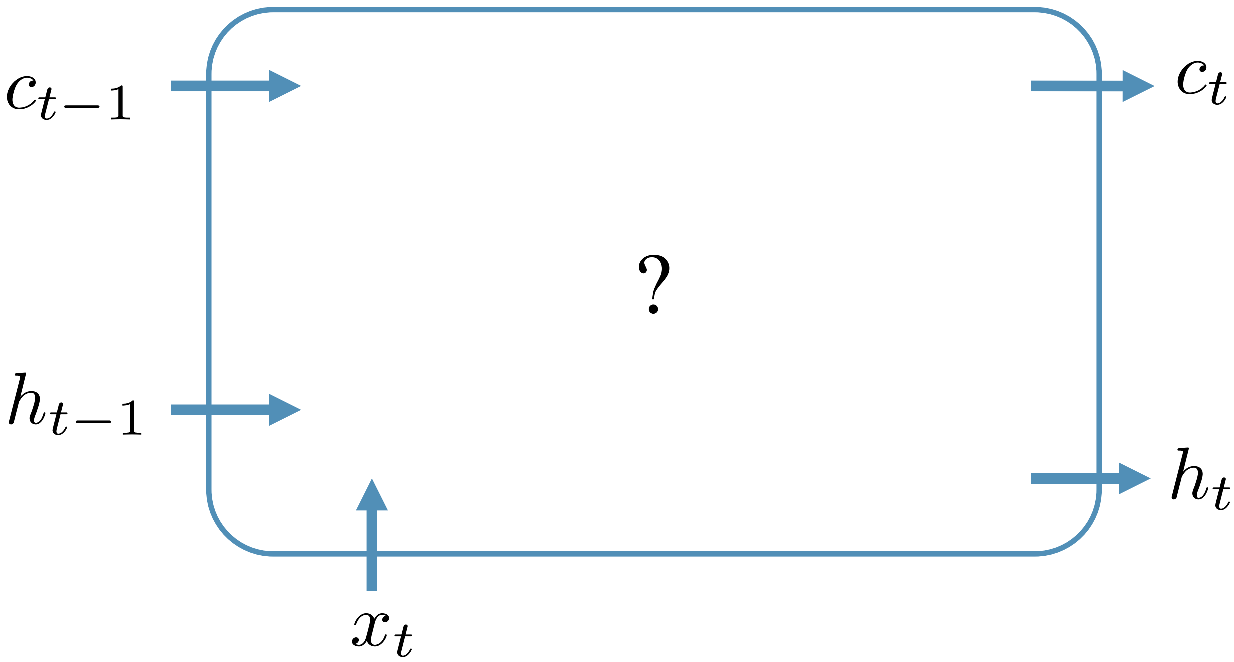
$$h_t = \tilde{f}(Vx_t + Wh_{t-1} + b_h)$$

Backward pass

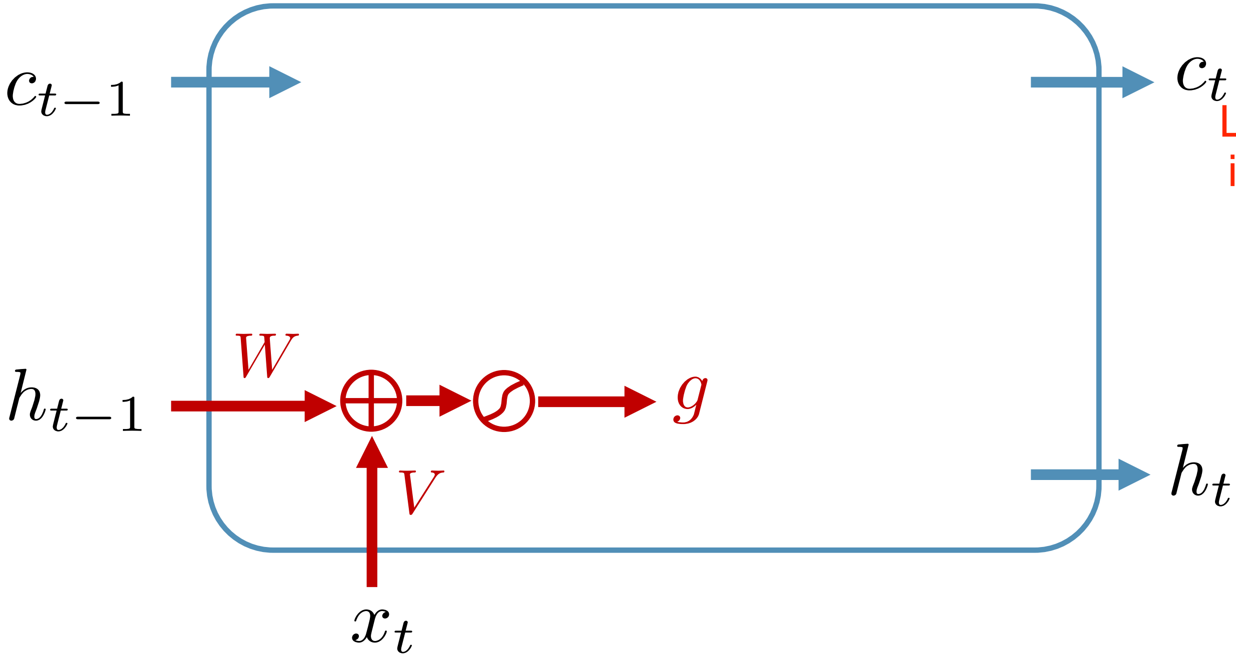
W and nolinearity ⇨ vanishing gradients

We need a short way for the gradients!

LSTM: version 0



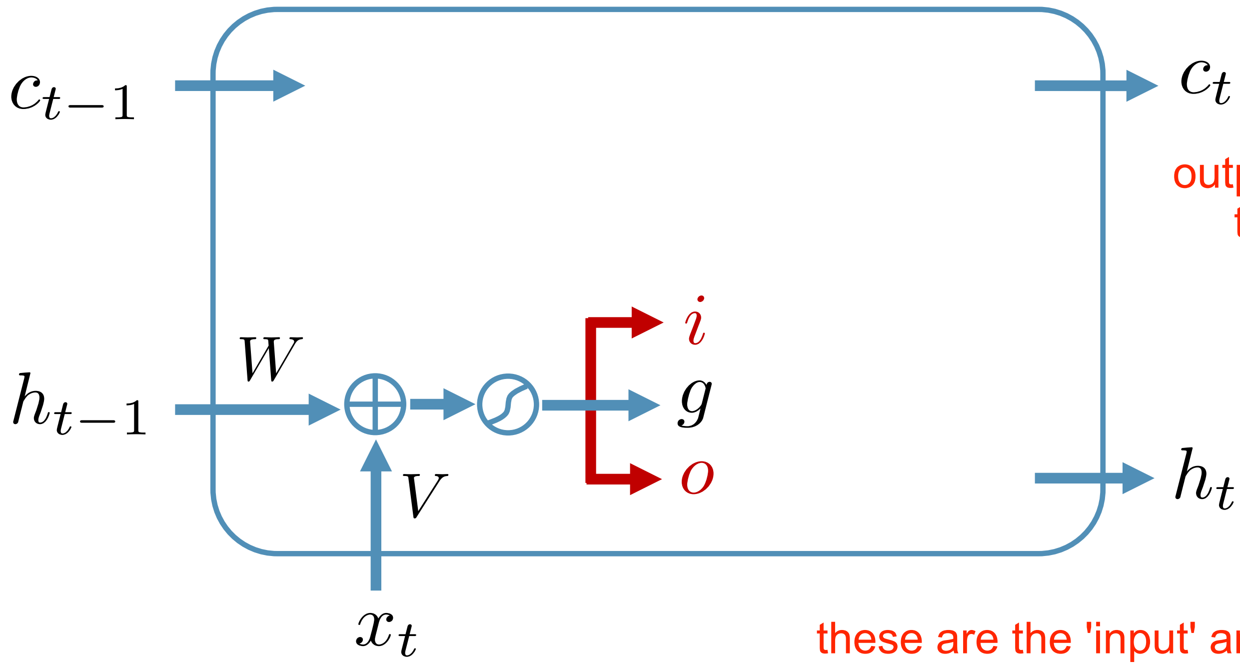
LSTM: version 0



LSTM layer have internal memory 'c' which is fed into the next timesteps.

$$g_t = \tilde{f}(V_g x_t + W_g h_{t-1} + b_g)$$

LSTM: version 0



output gate affects the output y.

these are the 'input' and 'output' gates.
Same dimension as hidden units.

$$g_t = \tilde{f}(V_g x_t + W_g h_{t-1} + b_g)$$

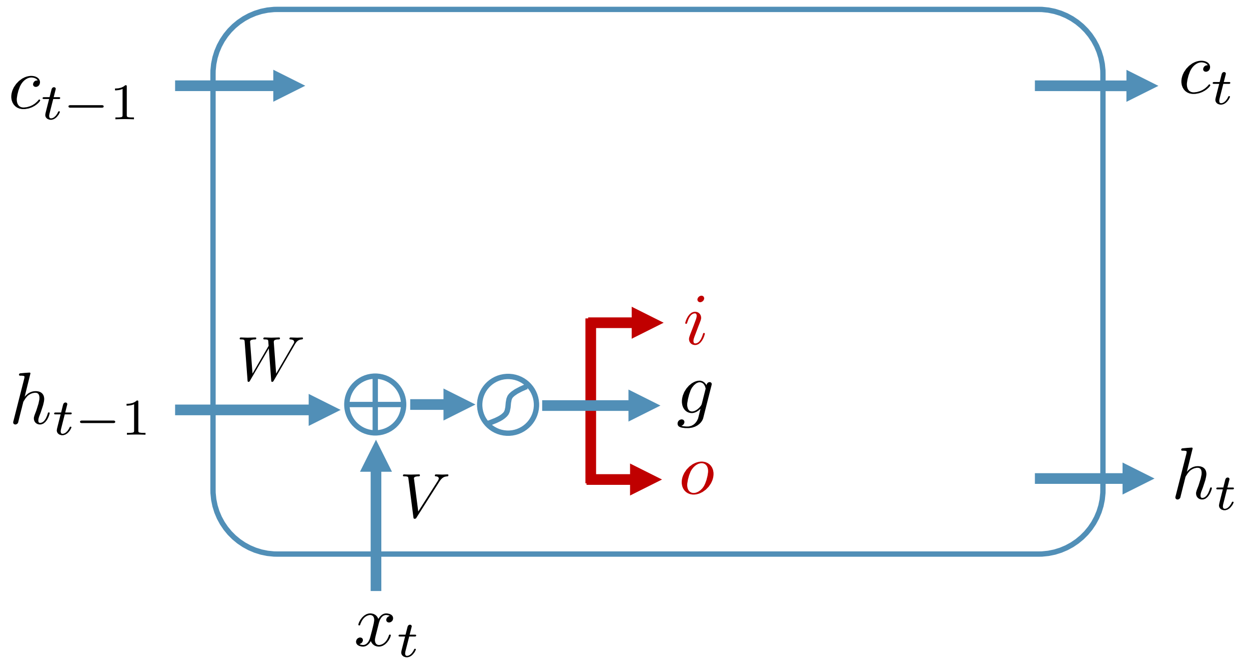
$$i_t = \sigma(V_i x_t + W_i h_{t-1} + b_i)$$

$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

To compute them, we use the same formula. To use nonlinearity over the linear combination.

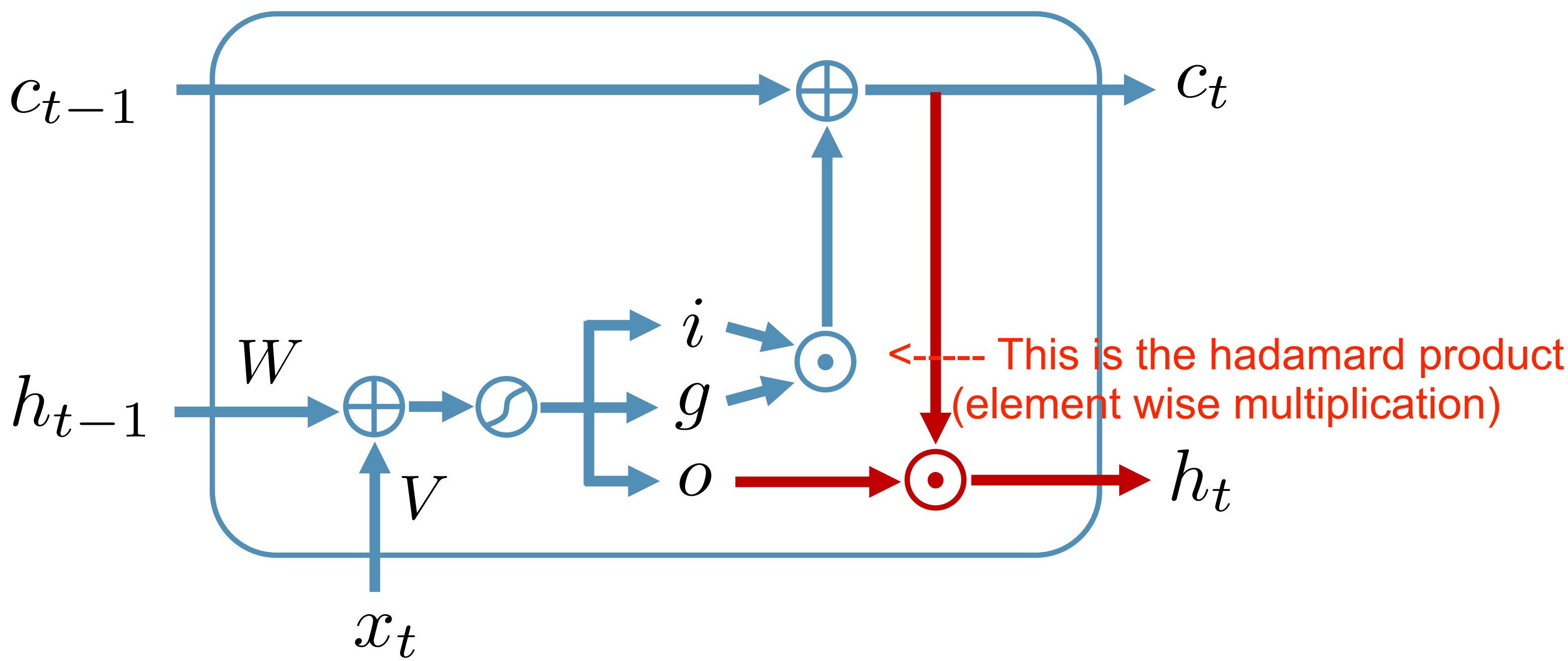
i_t = 1 means an open input gate, 0 closed.
Same goes for o_t.

LSTM: version 0



$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

LSTM: version 0

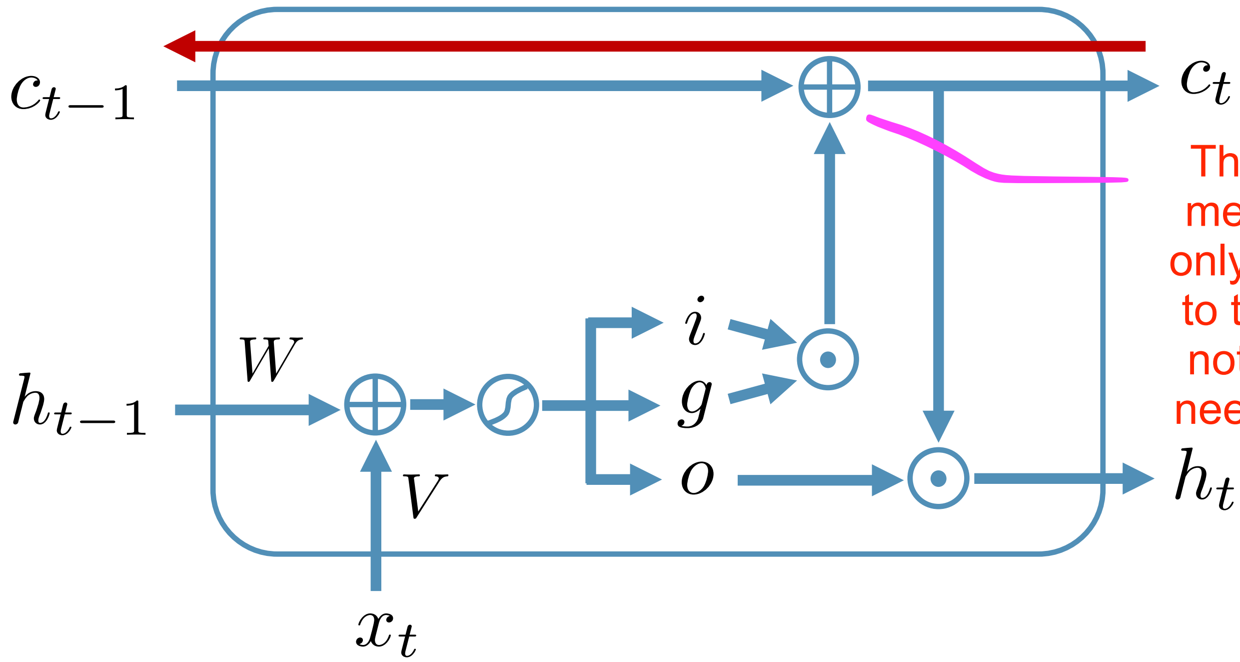


$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

$$c_t = c_{t-1} + i_t \cdot g_t$$

$$h_t = o_t \cdot \tilde{f}(c_t)$$

LSTM: vanishing gradients



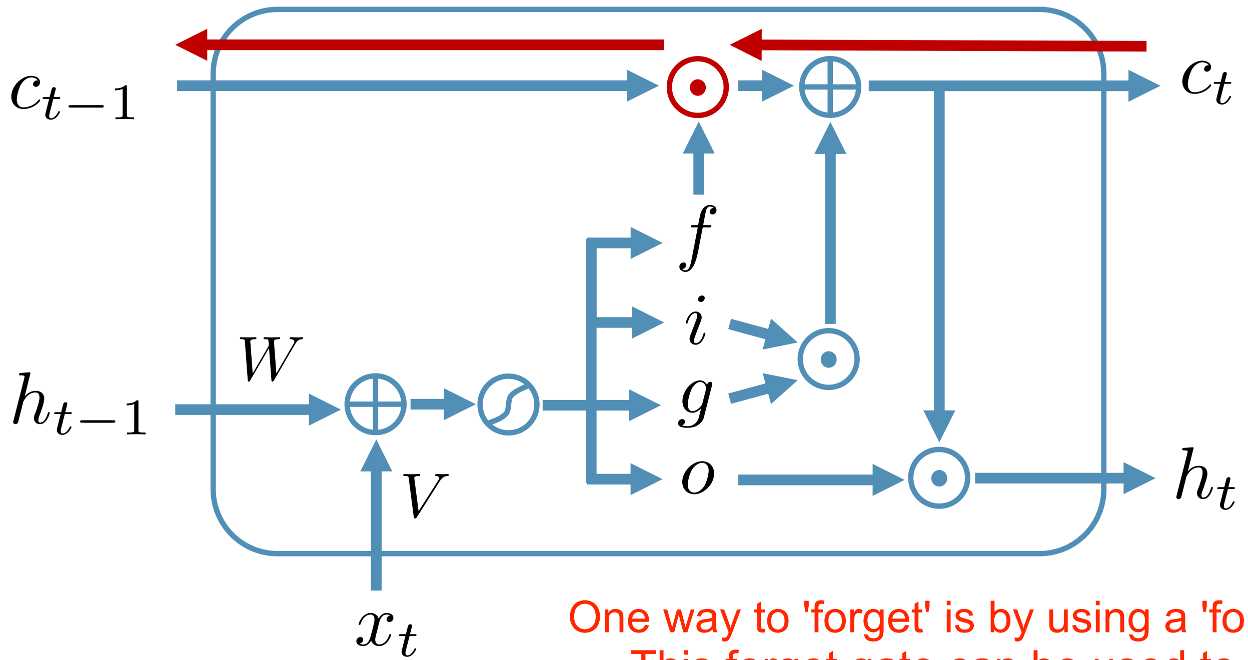
The 'addition' here means that we can only write something to the memory, and not erase it. So we need to improve our model.

$$c_t = c_{t-1} + i_t \cdot g_t \qquad \frac{\partial h_t}{\partial h_{t-1}} \Rightarrow \frac{\partial c_t}{\partial c_{t-1}} = diag(1)$$

Aha. The identity matrix.
Why ?
because c_t = c_{t-1} + i_t g_t.
d c_t / d_c{t-1} = 1
Clearly, we just get the identity when we backprop c_t.

Gradients do not vanish!

LSTM: forget sometimes



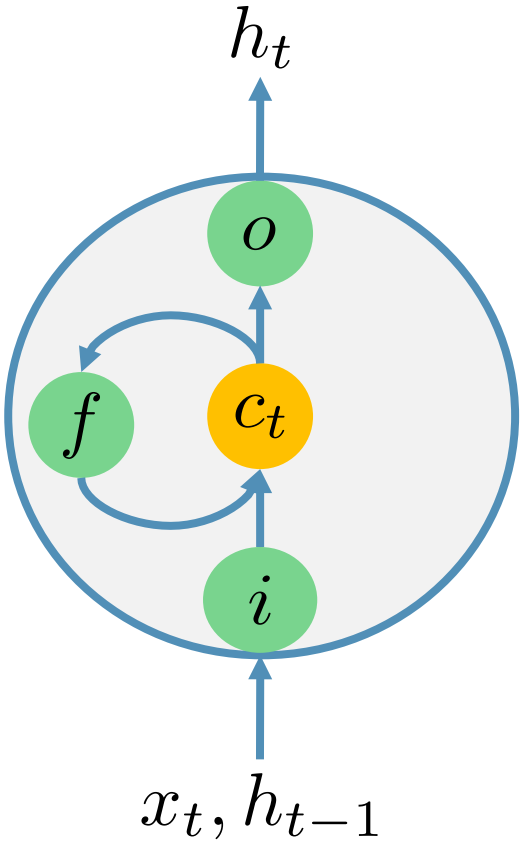
One way to 'forget' is by using a 'forget gate'.
This forget gate can be used to 'forget' memory states.

$$f_t = \sigma(V_f x_t + W_f h_{t-1} + b_f) \qquad c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = diag(f_t) \Rightarrow \text{High initial } b_f$$

Actually, the forget gate may even amplify the vanishing gradient problem, as it uses the sigmoid function. So to deal with this, the bias is initialized with high positive numbers. At the beginning, the gradient is 1 so the LSTM cannot forget anything. Then, we start to learn 'forgetfulness' and take out unnecessary memory.

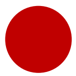
LSTM: extreme regimes



LSTM: extreme regimes

Types of information storage.

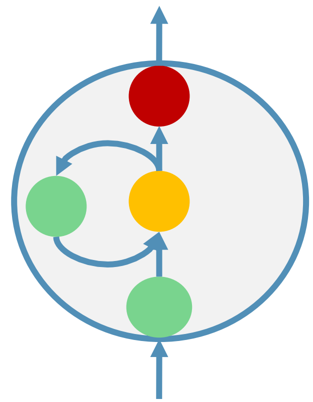
NOTE: close means 0, open means 1.

 - gate is close

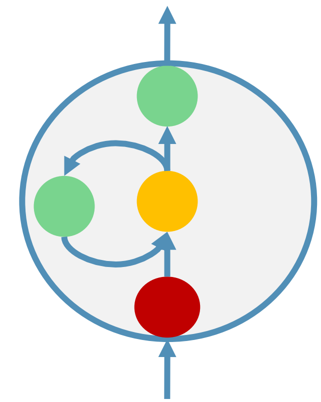
 - gate is open

remember, output can be gated.

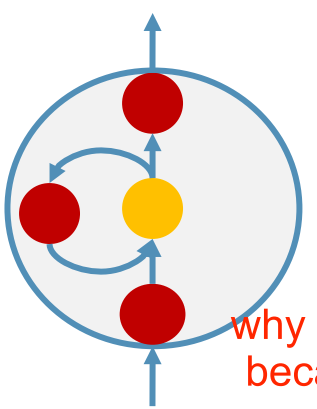
Captures info



Releases info



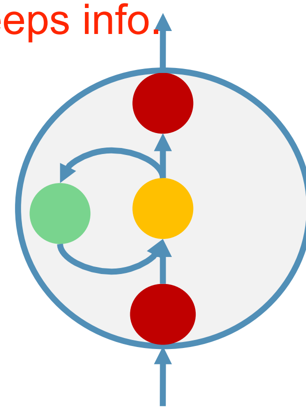
Erases info



why erase info? because f = 0, meaning that we forget our memory.

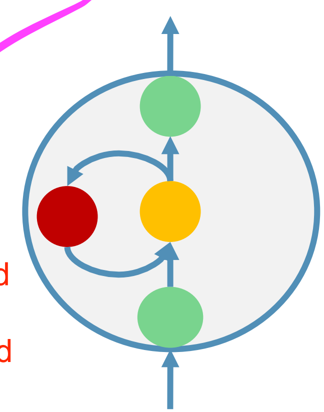
Keeps info

f = 1 : keeps info.



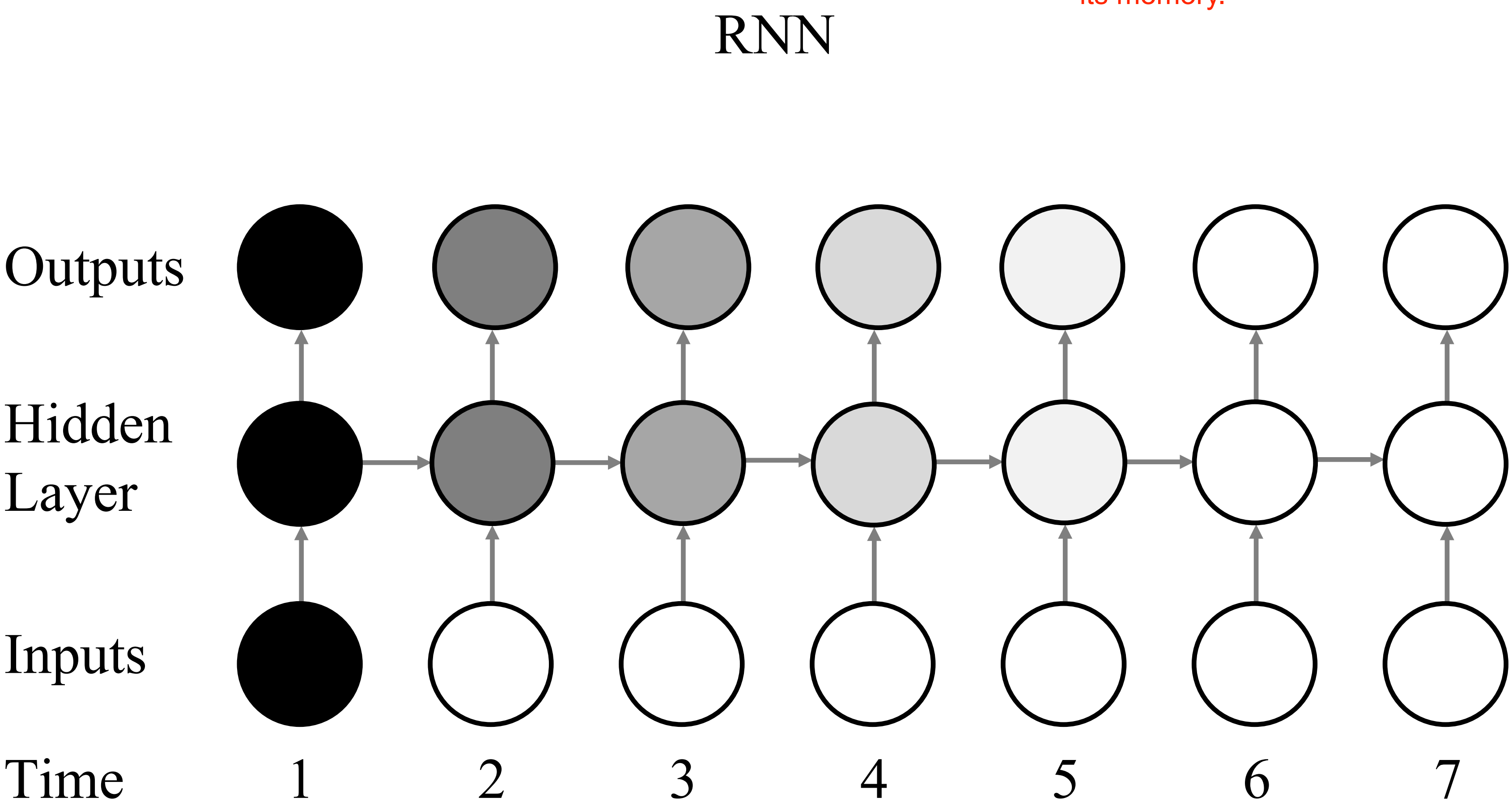
if forget cell is closed, then we basically have an RNN.
The input gates and output gates just function as standard dense layers, presumably.

= RNN

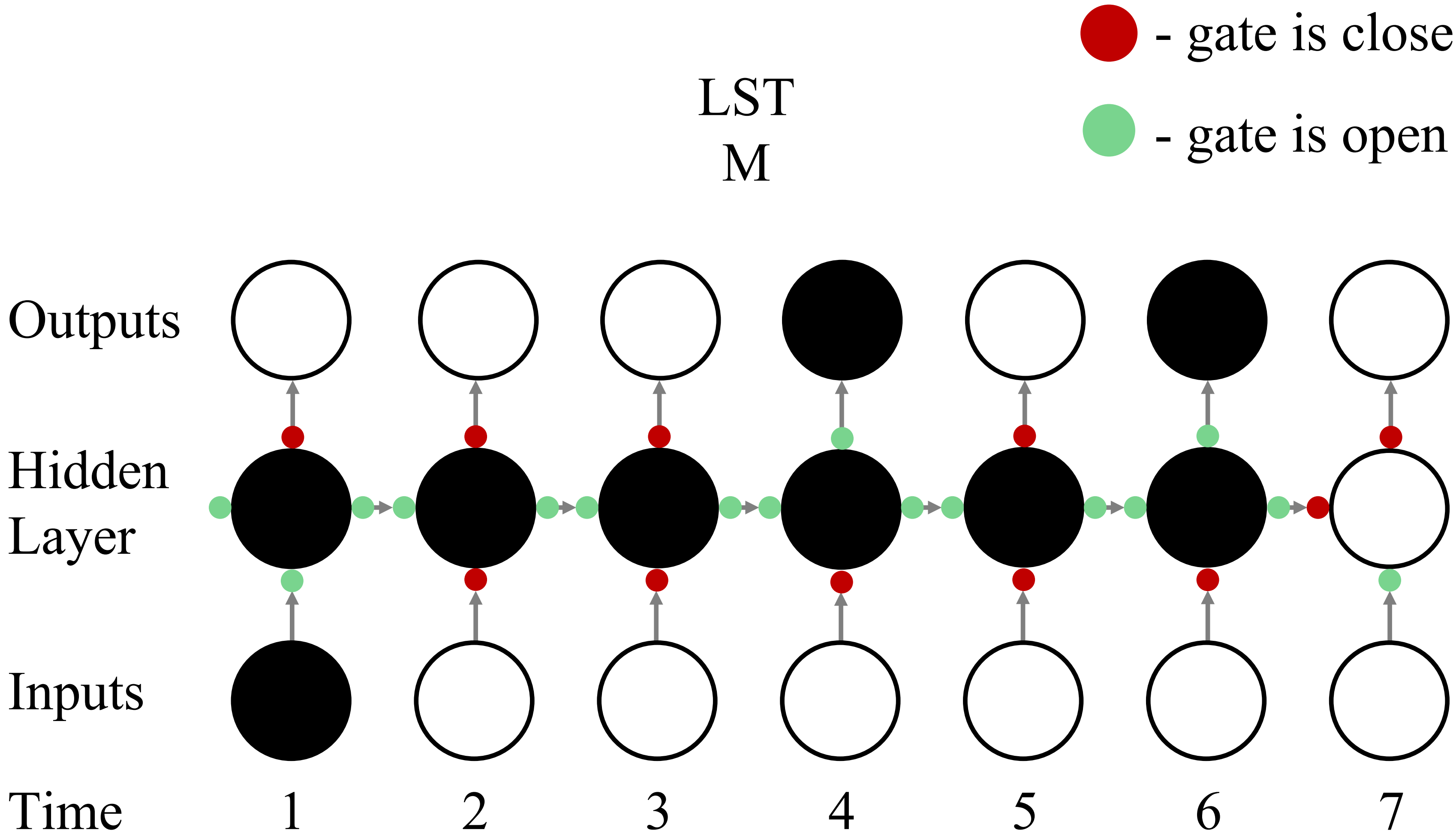


LSTM: information flow

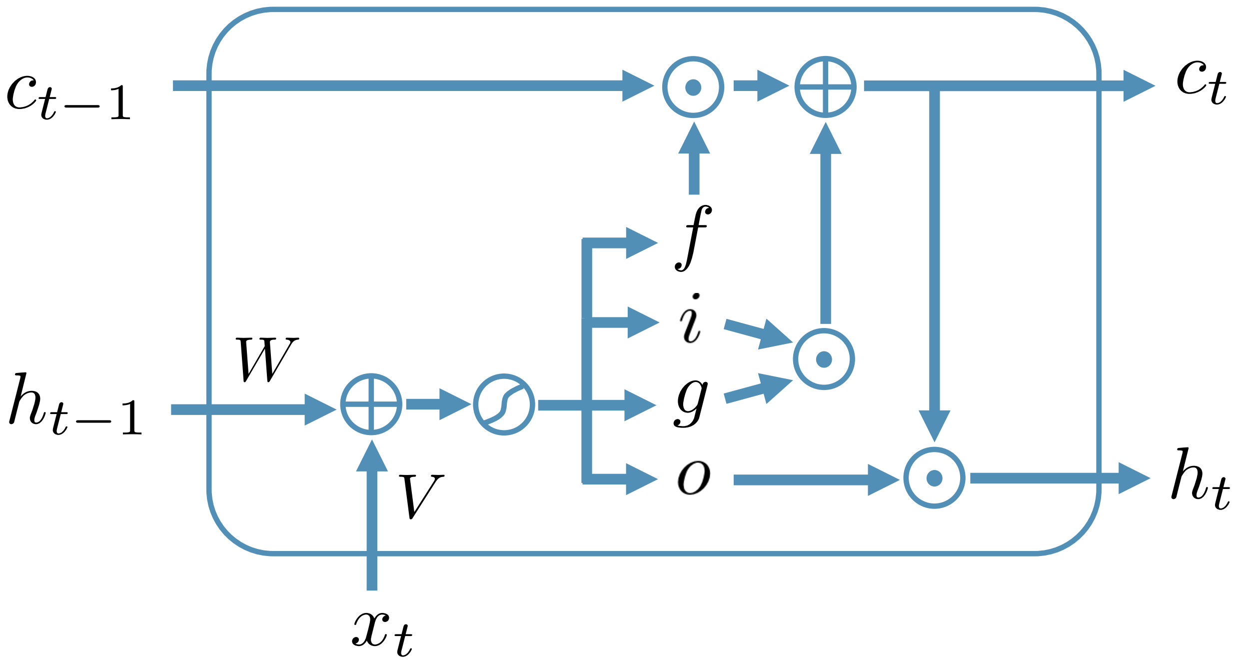
Then RNN
gradually forgets
its memory.



LSTM: information flow



LSTM



$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

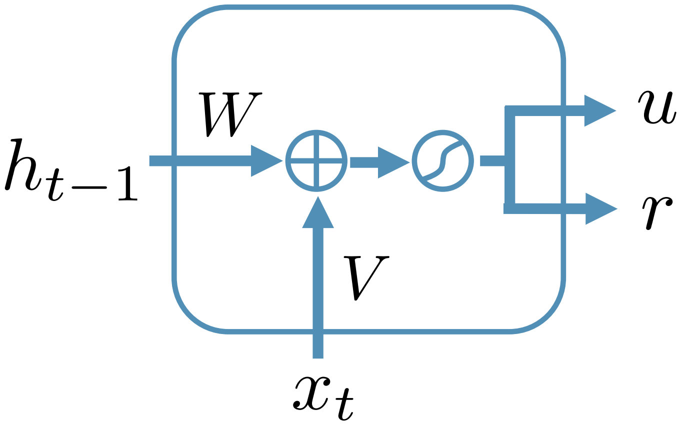
$$h_t = o_t \cdot \tilde{f}(c_t)$$

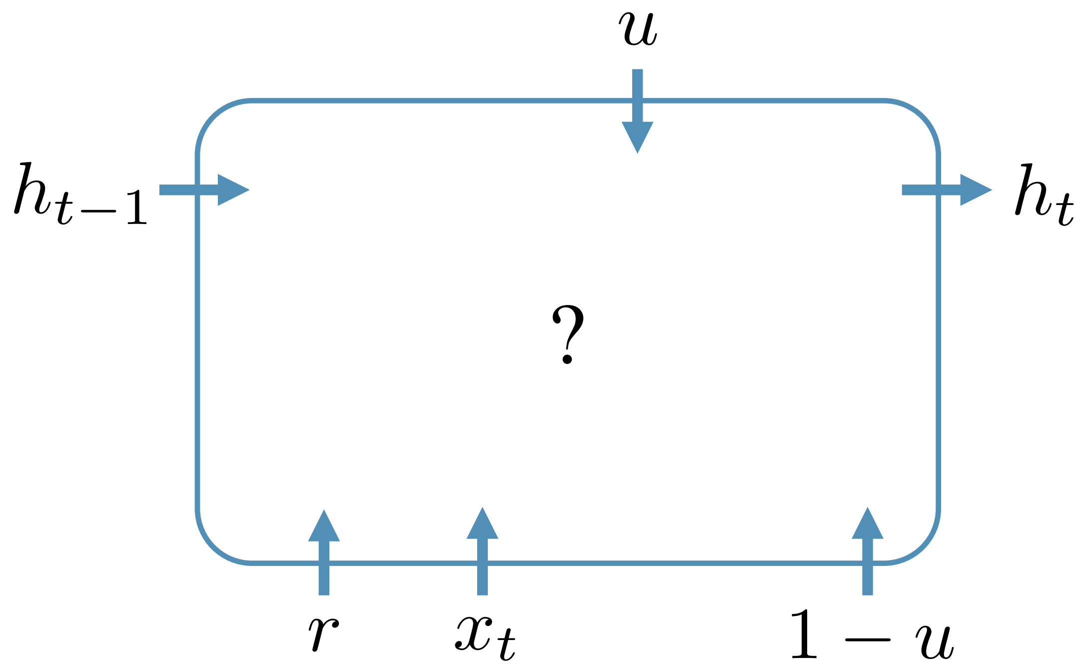
LSTM drawbacks:

- need to keep track of many gradients, backprop very long.
Also, lots of variables meaning that it may overfit.

GRU

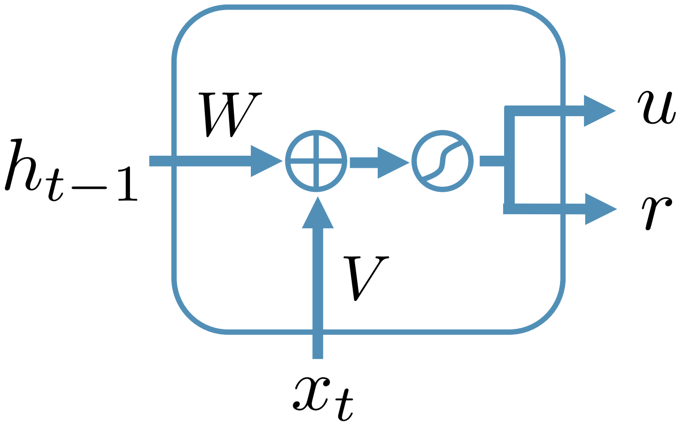
The GRU is a better alternative than the LSTM.

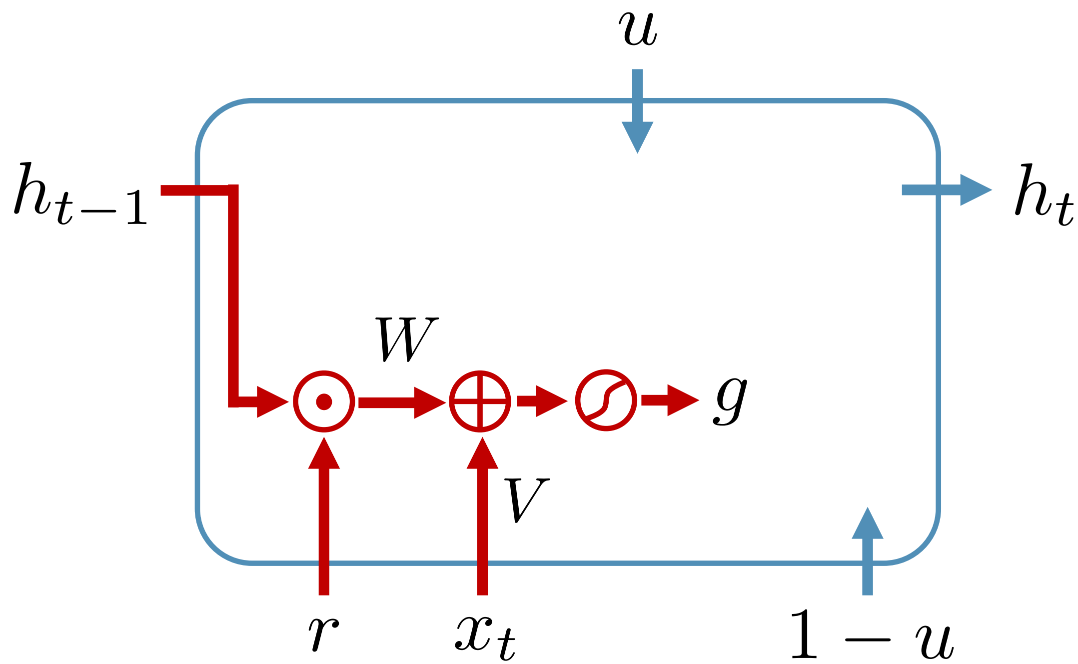




$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

GRU

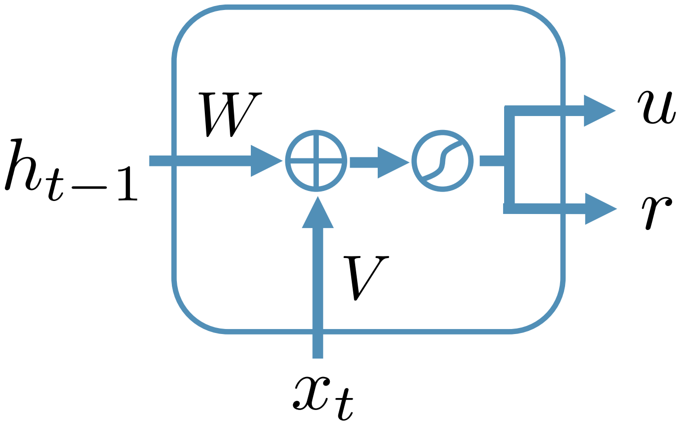


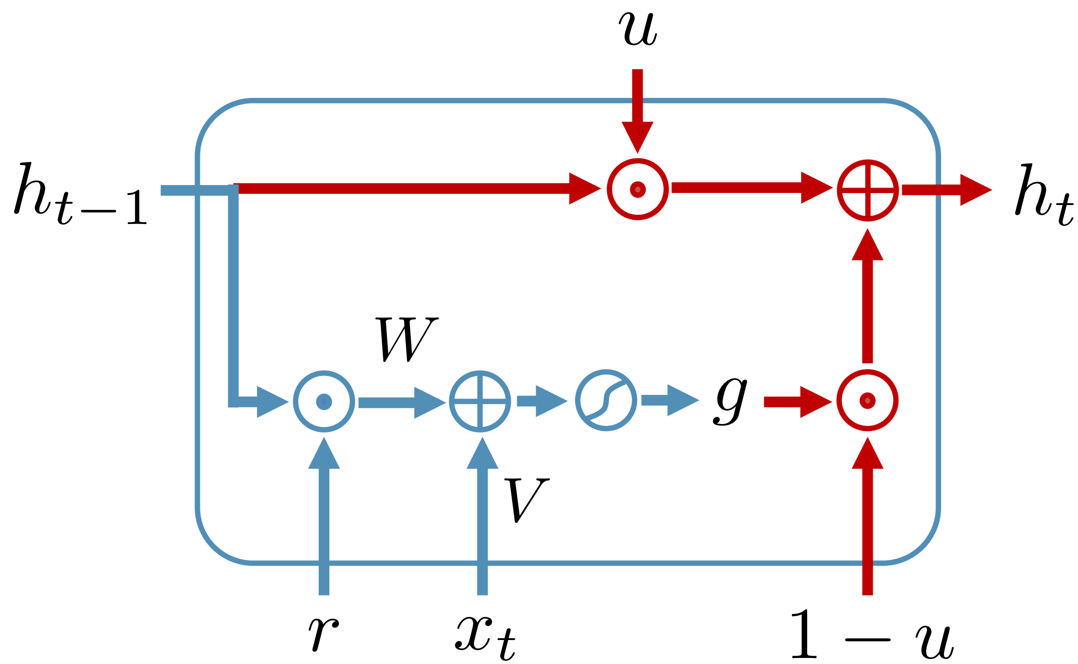


$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b) \qquad g_t = \tilde{f}\left(V_g x_t + W_g(h_{t-1} \cdot r_t) + b_g\right)$$

r_t is the reset gate.
It controls which parts of the hidden units in the previous timestep are used in the information vector g.

GRU



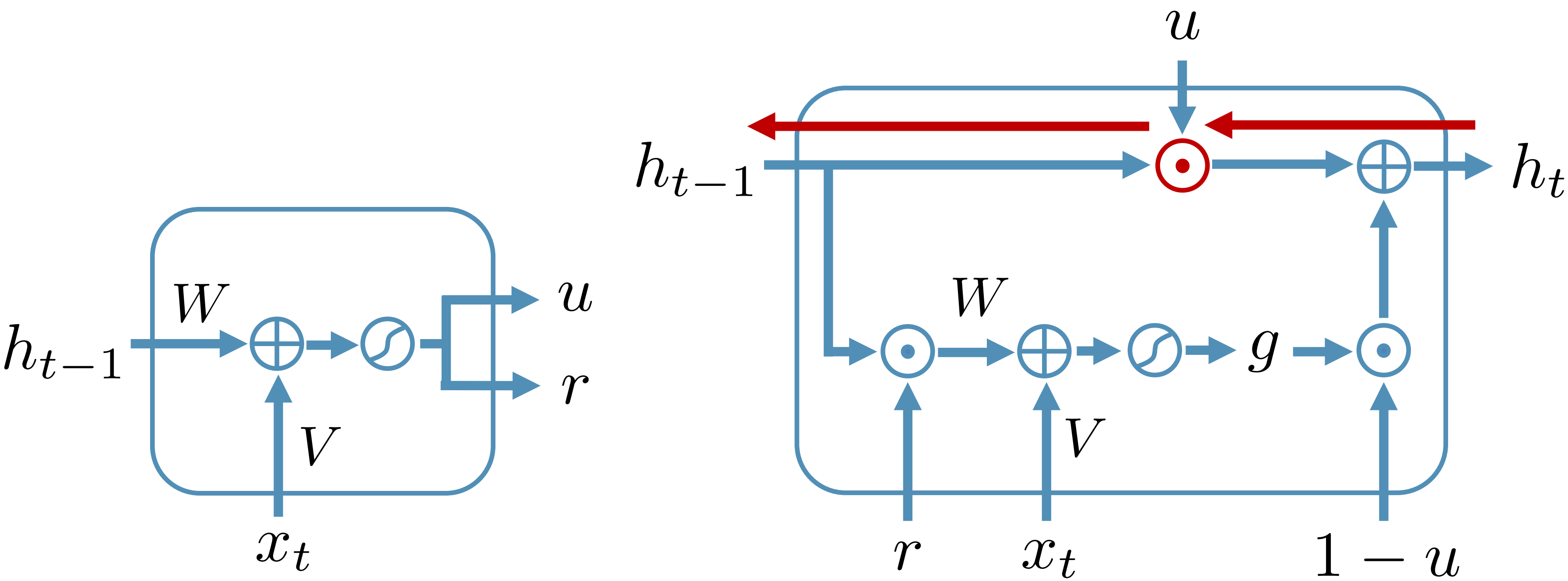


$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b) \qquad g_t = \tilde{f}\left(V_g x_t + W_g(h_{t-1} \cdot r_t) + b_g\right)$$

$$h_t = (1 - u_t) \cdot g_t + u_t \cdot h_{t-1}$$

u_t is the update gate.
It controls the balance between storing the previous values of the hidden units, and writing new information into hidden units.
Wors as combination of input and forget gates in LSTM.

GRU: vanishing gradients



To remedy vanishing gradient problem, we initialize bias with high positive value.

$$u_t = \sigma(V_u x_t + W_u h_{t-1} + b_u) \qquad h_t = (1-u_t)\cdot g_t + u_t \cdot h_{t-1}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = diag(1-u_h)\cdot\frac{\partial g_h}{\partial h_{h-1}} + diag(u_h)$$

High initial b_u

LSTM or GRU?

LSTM → more flexible

GRU → less parameters

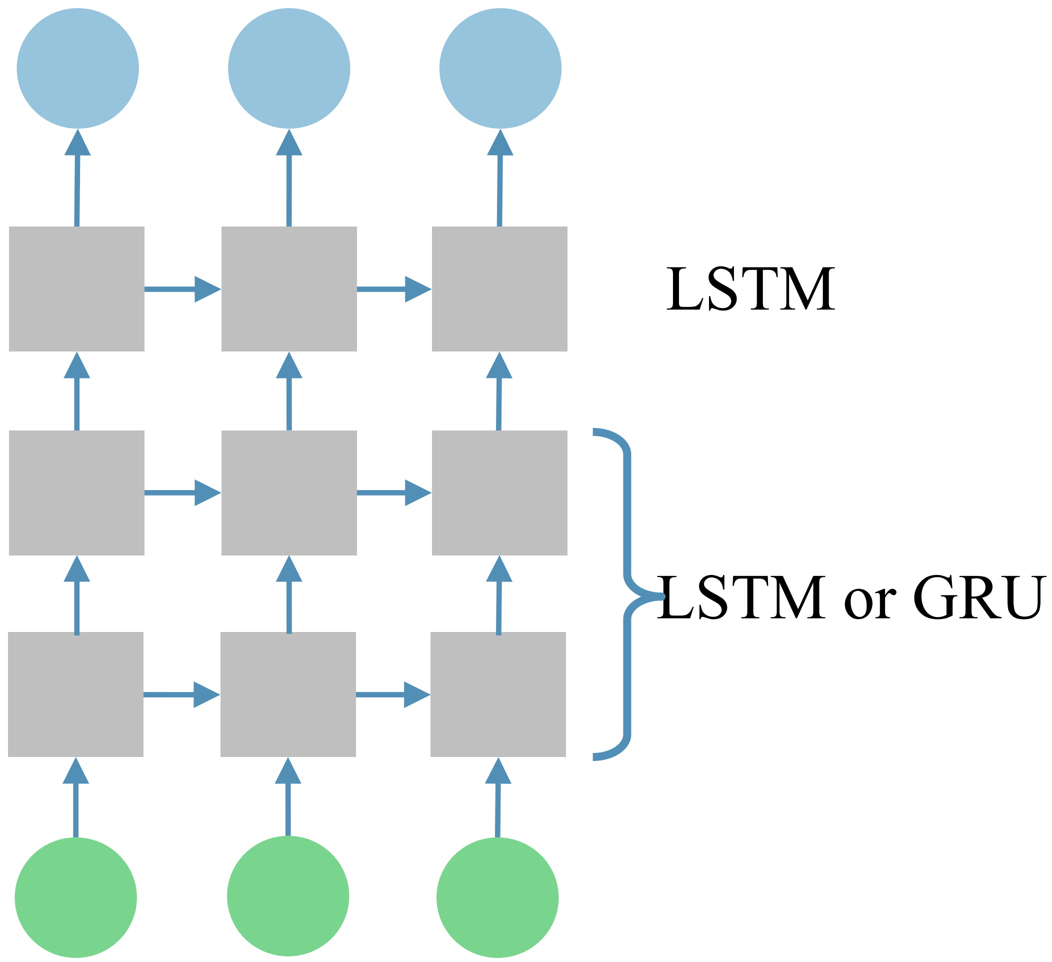
Main benefits of each.

If difference in accuracy negligible, use GRU.

First train LSTM → Second train GRU

Compare and choose

LSTM or GRU: stack more layers



Summary

- Gated recurrent architectures: LSTM and GRU.
- They do not suffer from vanishing gradients that much because there is an additional short way for the gradients through them

In the next video:

How to use RNNs to solve different practical tasks