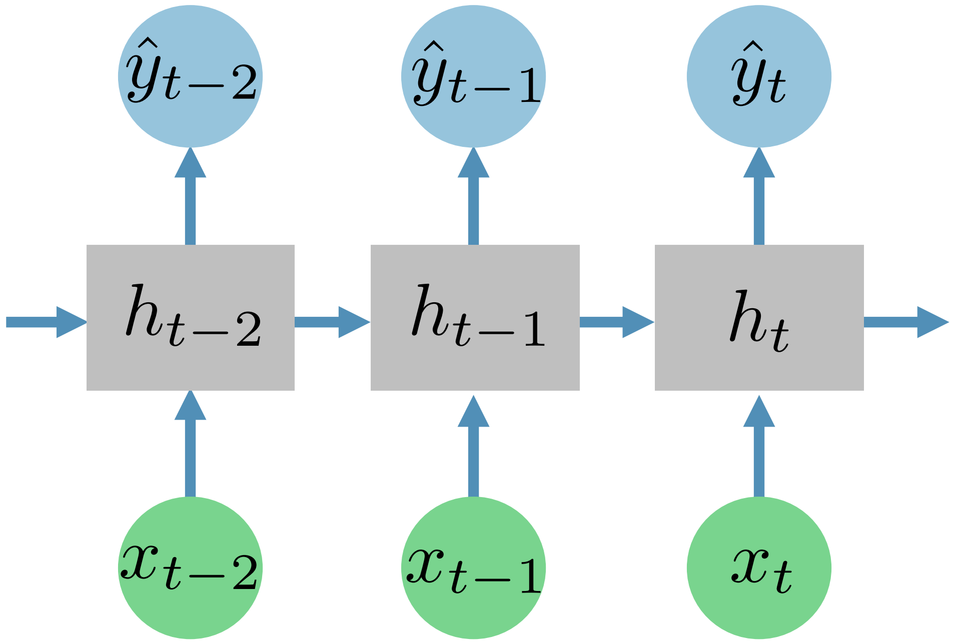
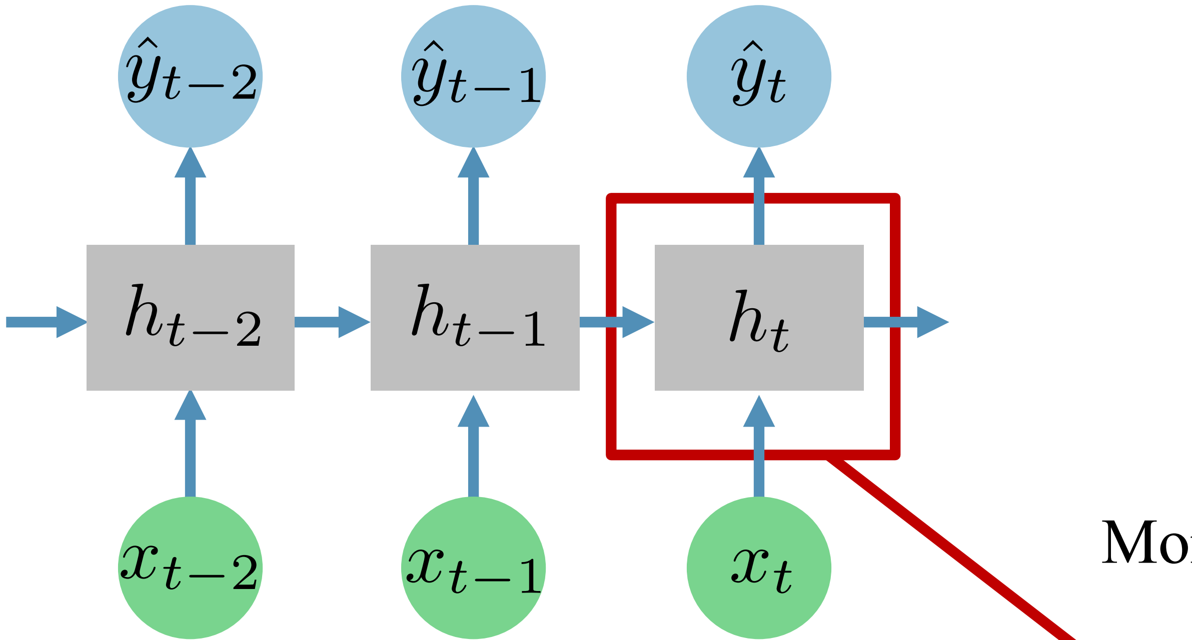
LSTM and GRU

Previously on this week: Simple RNN



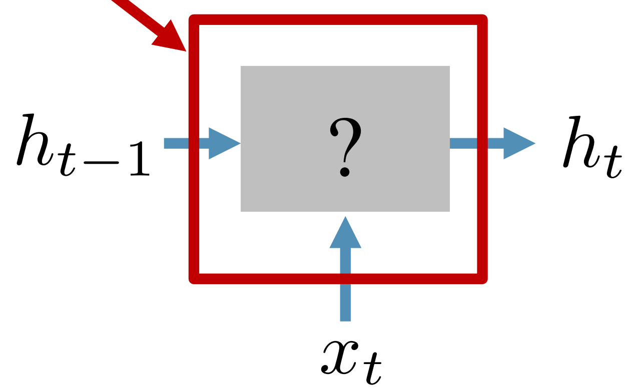
$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

Previously on this week: Simple RNN

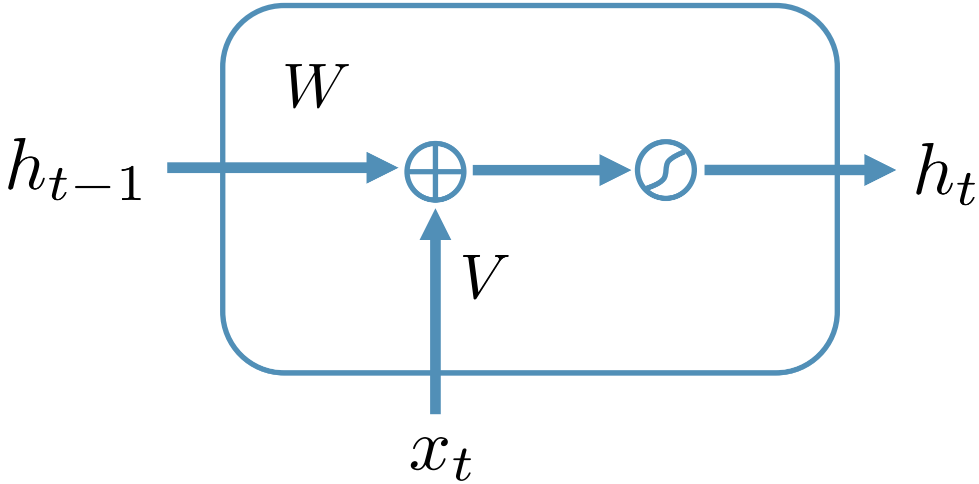


More sophisticated function

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

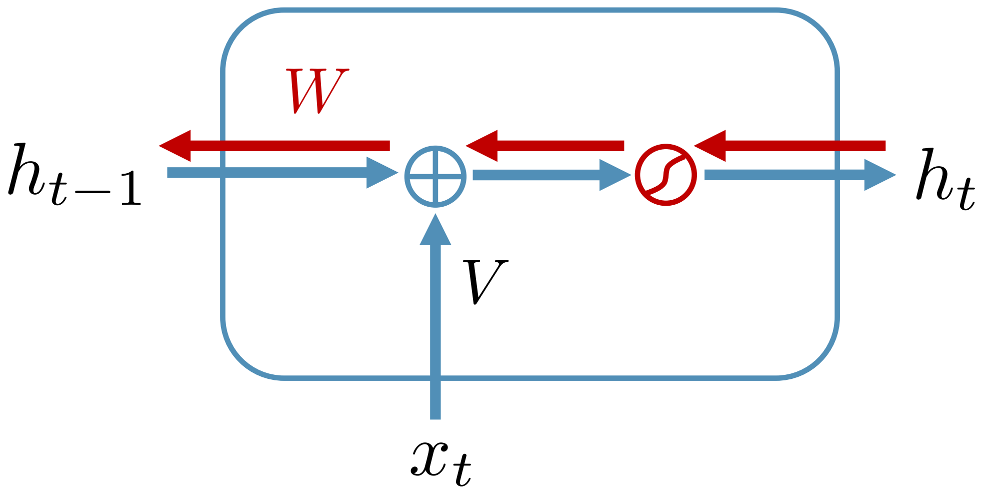


Simple RNN



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

Simple RNN



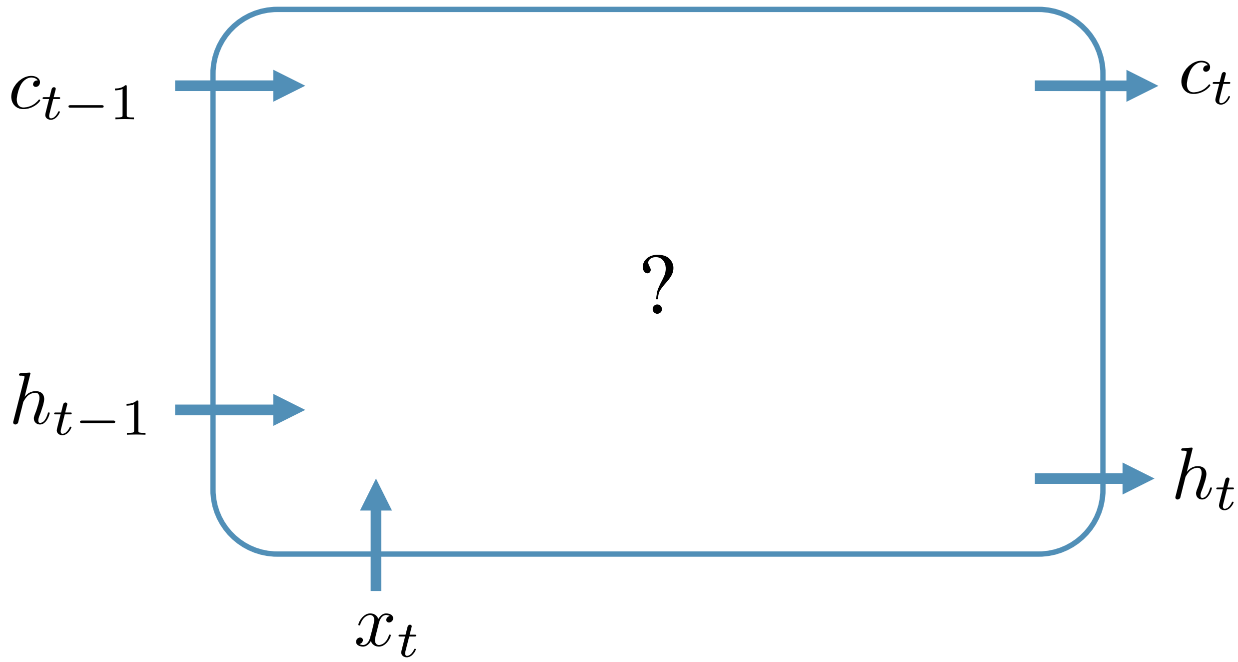
$$h_t = \tilde{f}(Vx_t + Wh_{t-1} + b_h)$$

Backward pass

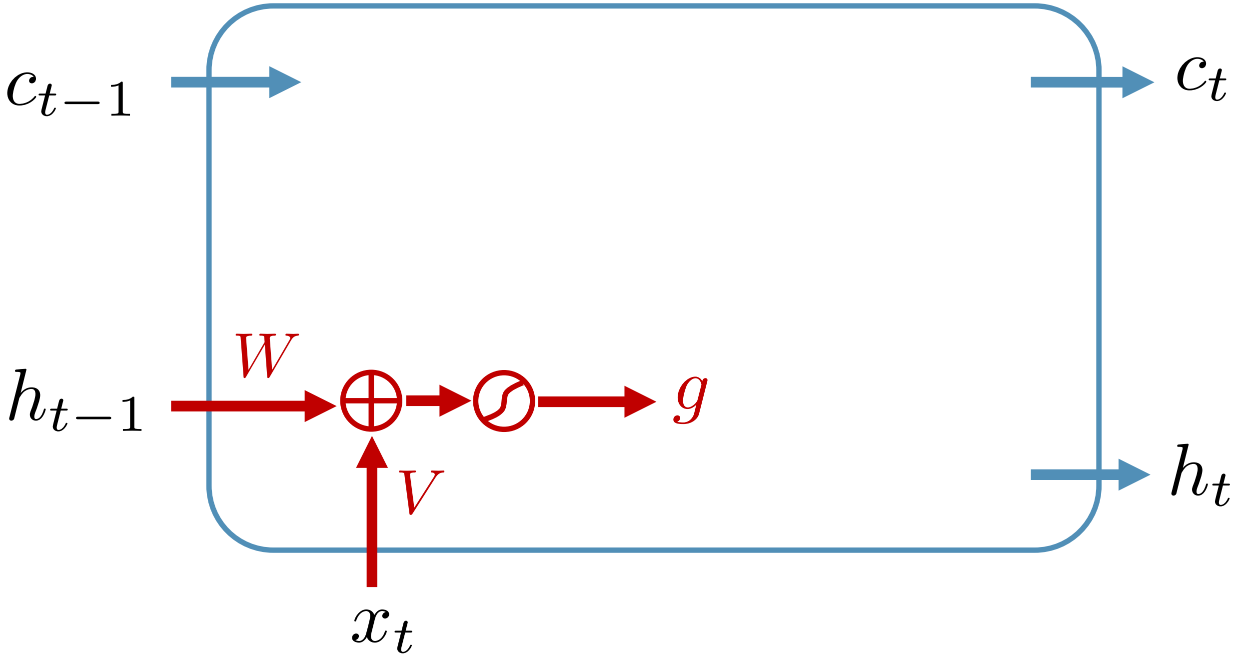
W and nolinearity ➡ vanishing gradients

We need a short way for the gradients!

LSTM: version 0

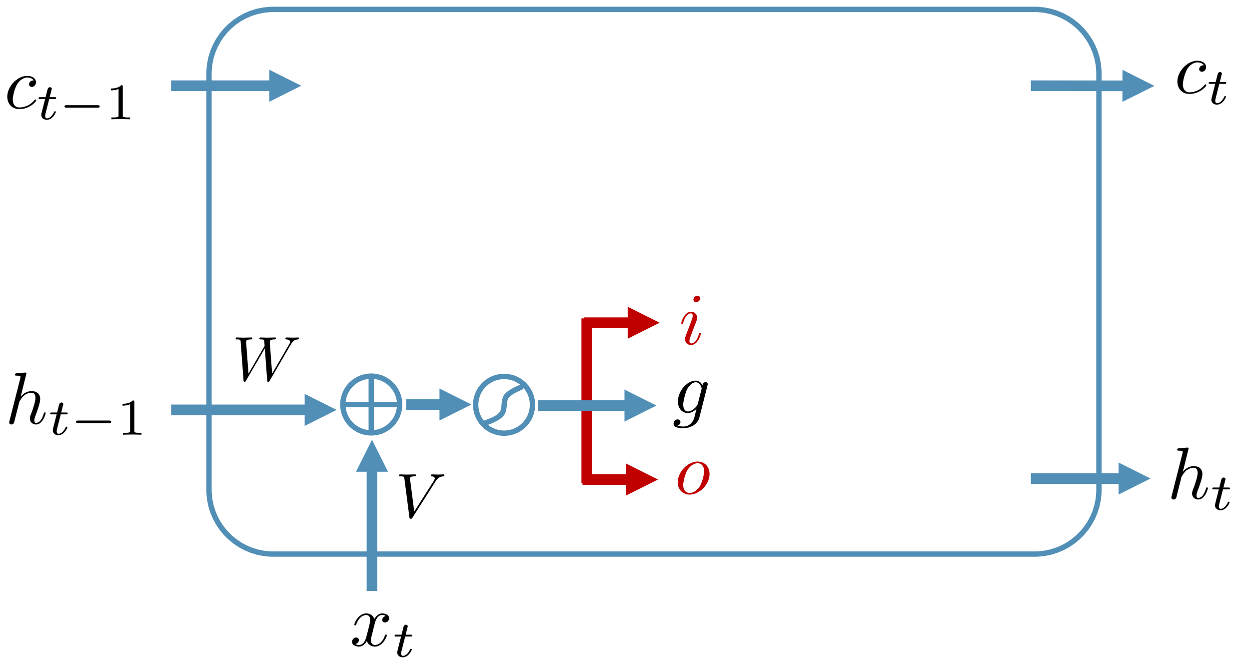


LSTM: version 0



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

LSTM: version 0

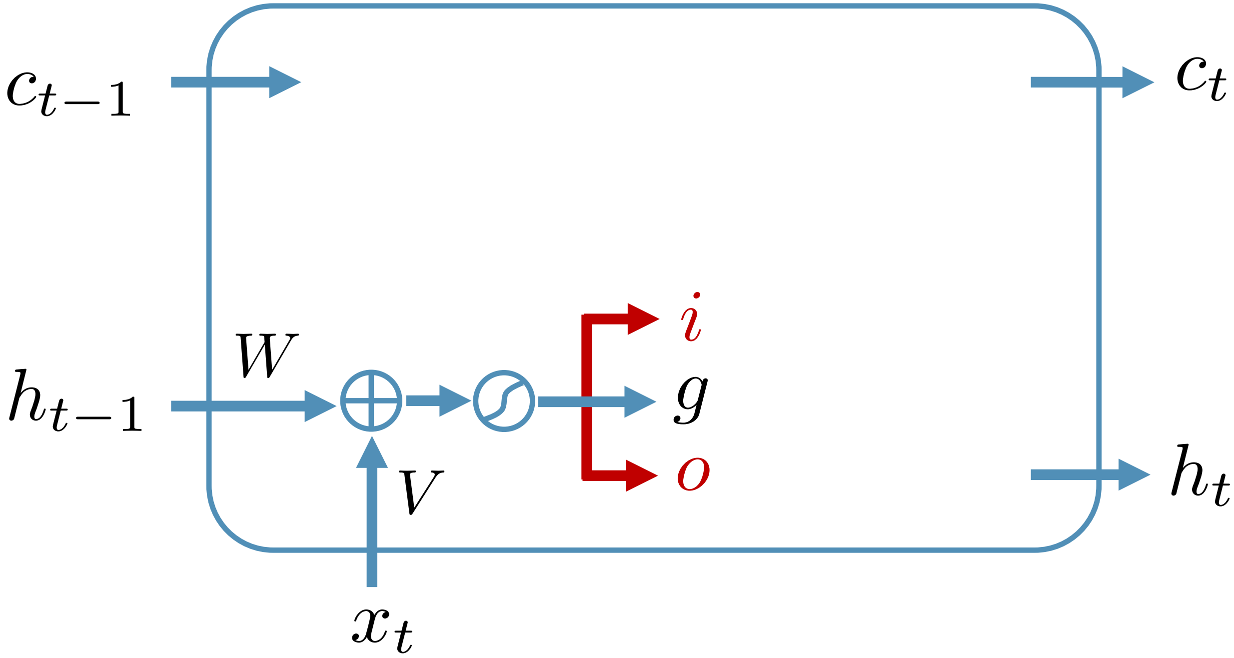


$$g_t = \tilde{f}(V_g x_t + W_g h_{t-1} + b_g)$$

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

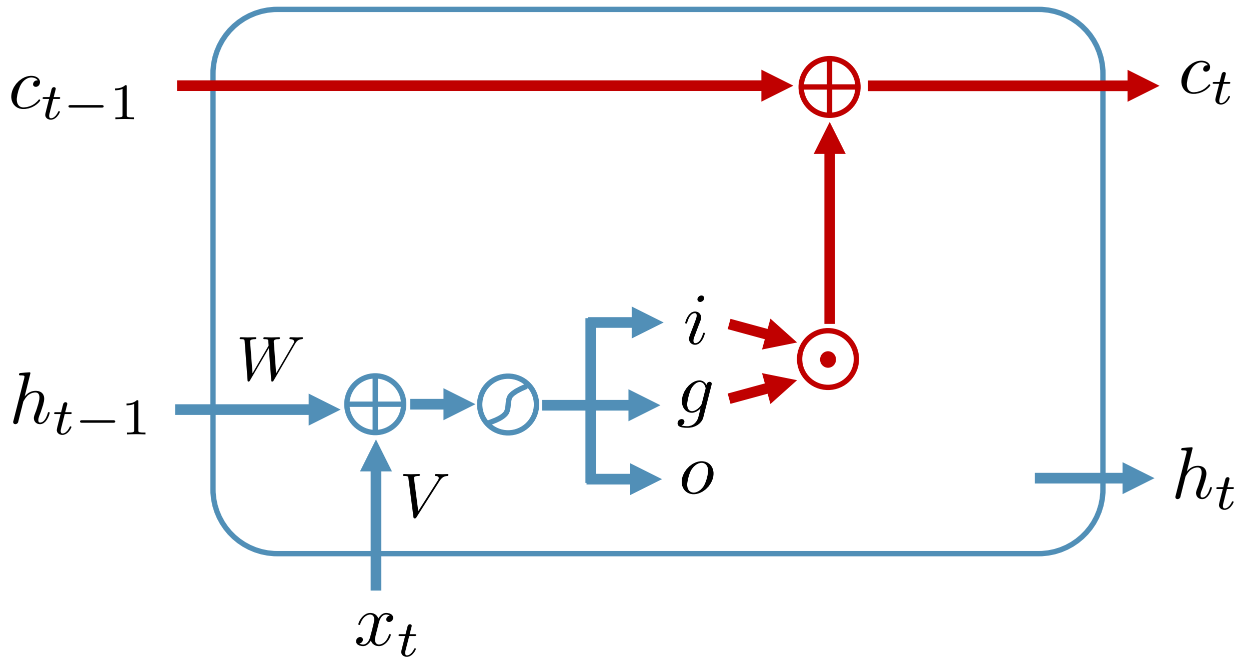
$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

LSTM: version 0



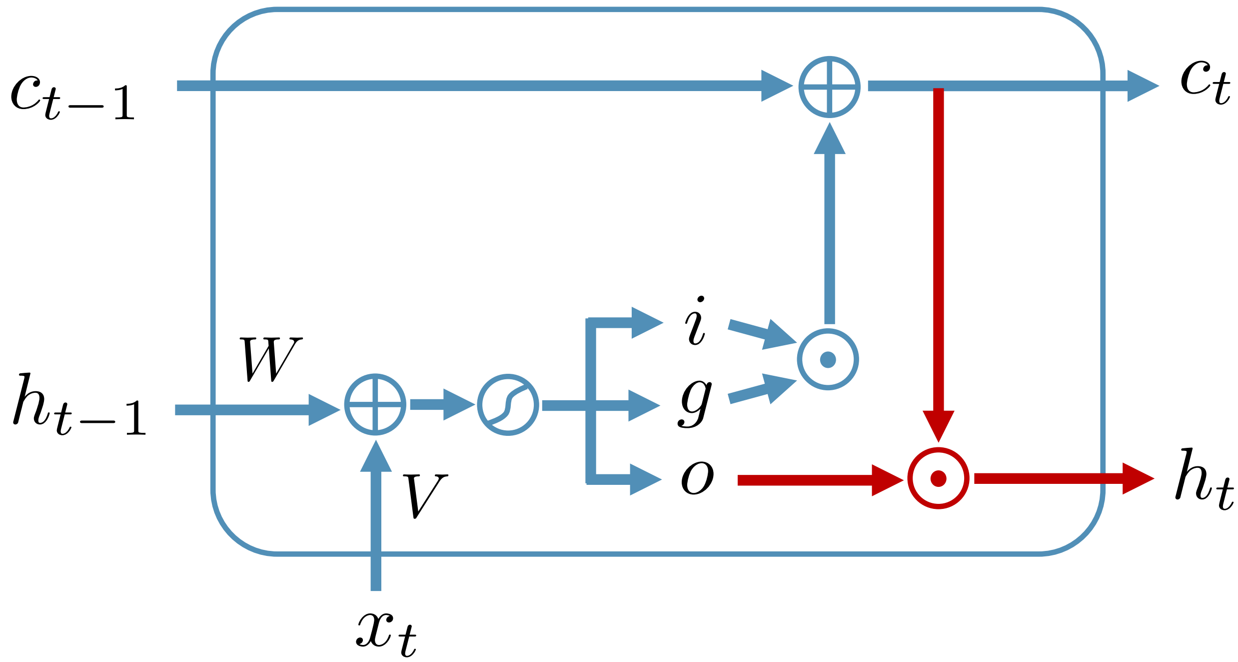
$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

LSTM: version 0



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

LSTM: version 0

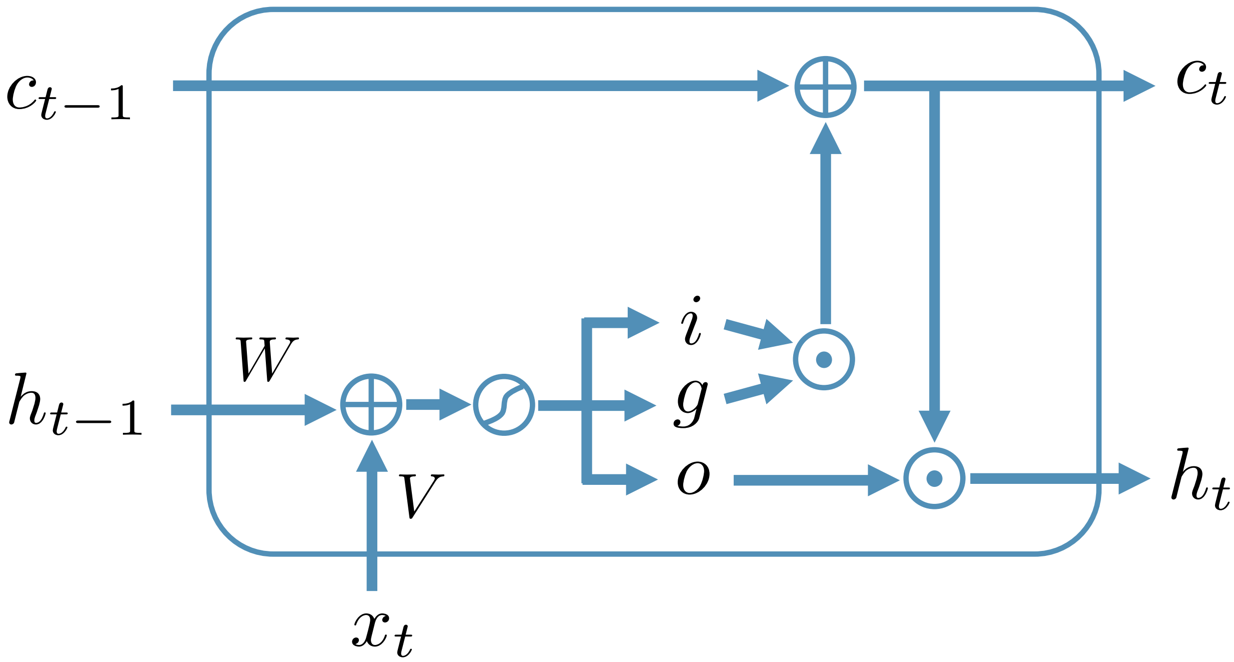


$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

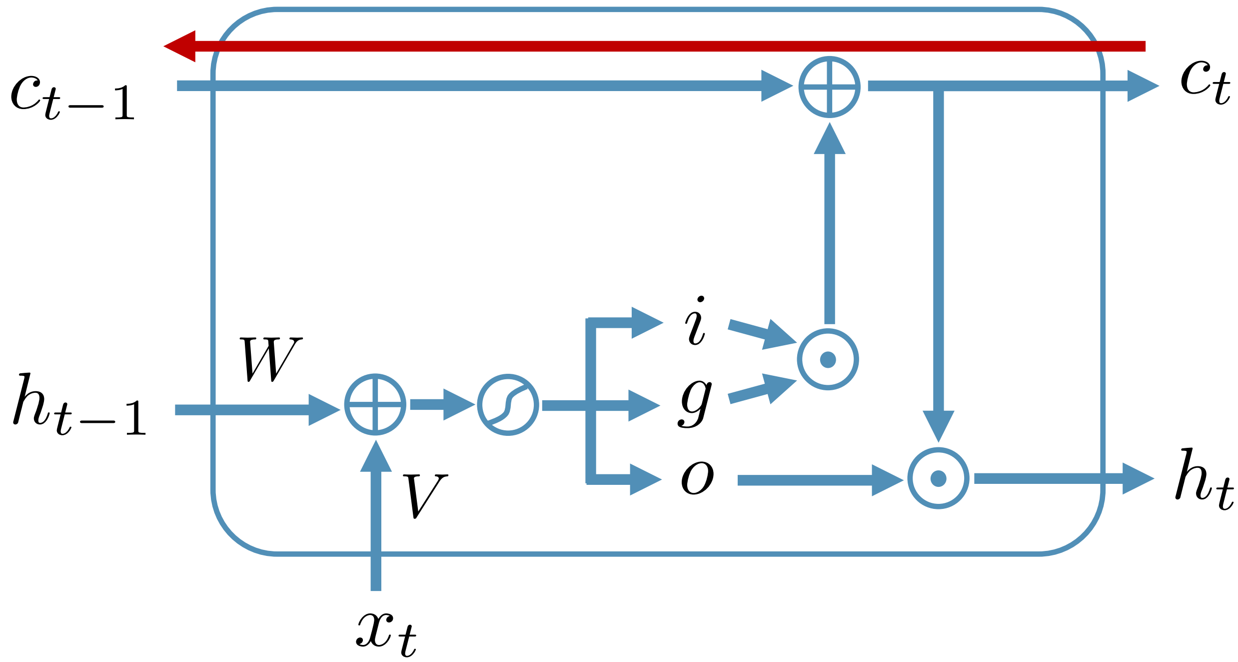
$$c_t = c_{t-1} + i_t \cdot g_t$$

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

LSTM: vanishing gradients



LSTM: vanishing gradients

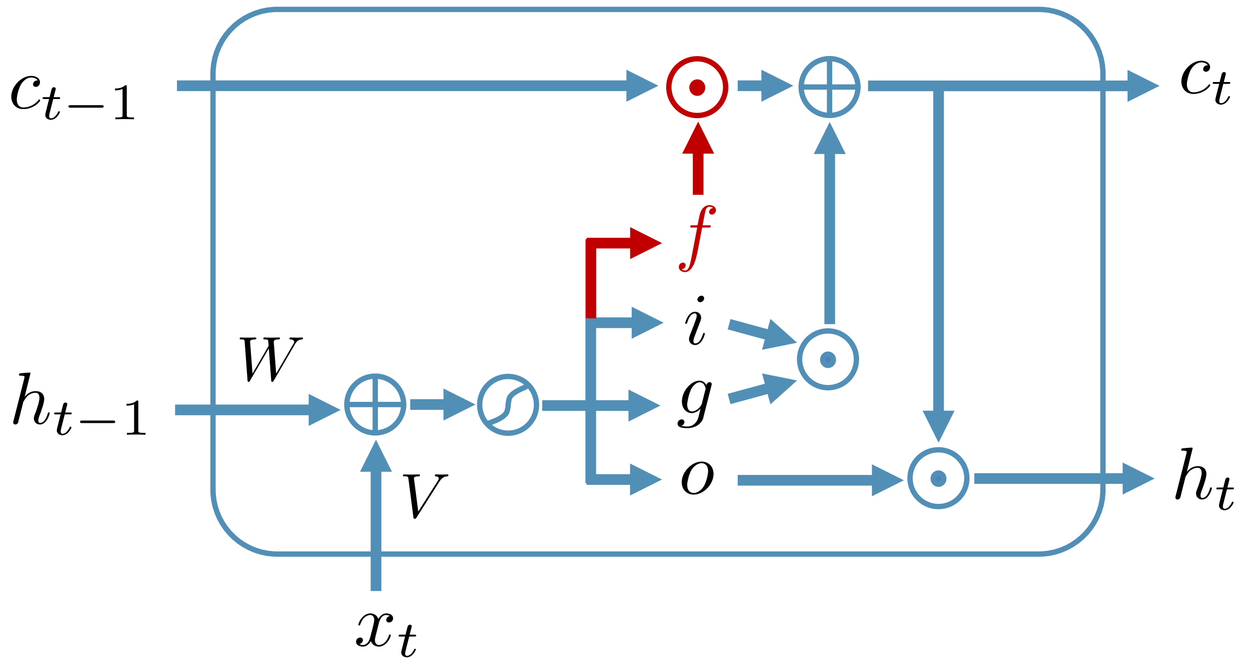


$$c_t = c_{t-1} + i_t \cdot g_t$$

$$\frac{\partial h_t}{\partial h_{t-1}} \Rightarrow \frac{\partial c_t}{\partial c_{t-1}} = diag(1)$$

Gradients do not vanish!

LSTM: forget sometimes

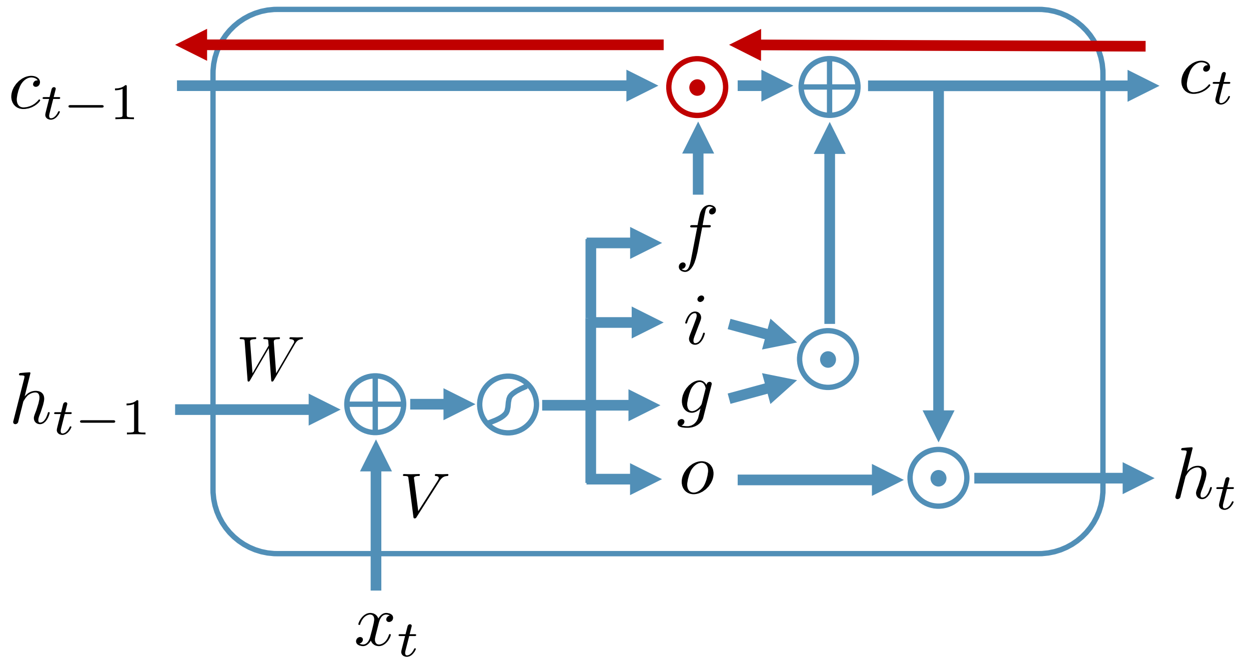


$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ \color{red}{f_t} \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

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

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

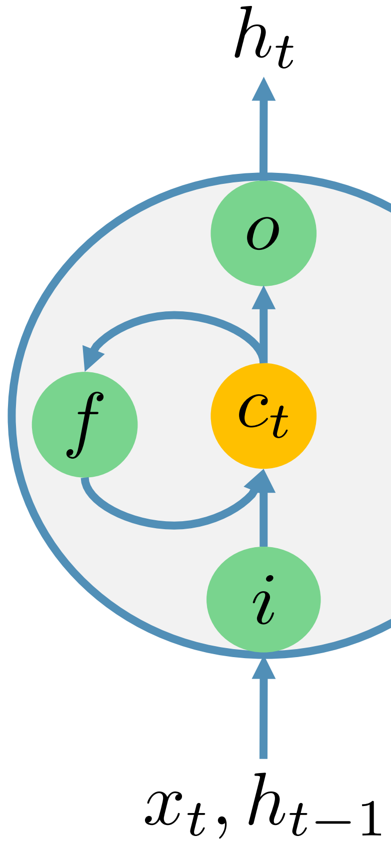
LSTM: forget sometimes



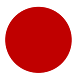
$$f_t = \sigma(V_f x_t + W_f h_{t-1} + b_f) \qquad c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

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

LSTM: extreme regimes

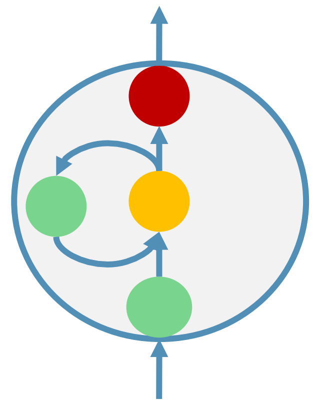


LSTM: extreme regimes

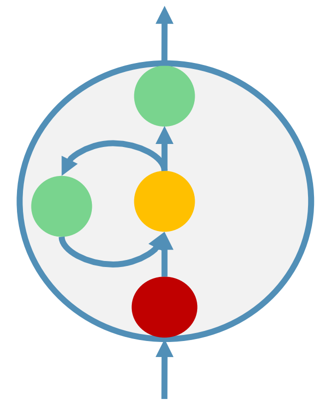
 - gate is close

 - gate is open

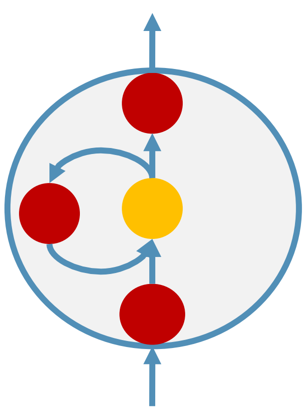
Captures info



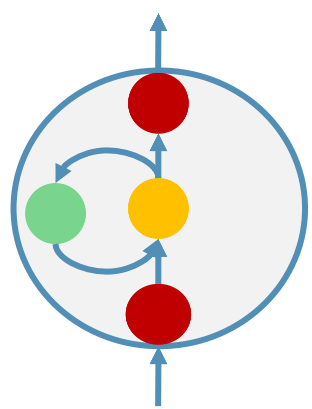
Releases info



Erases info



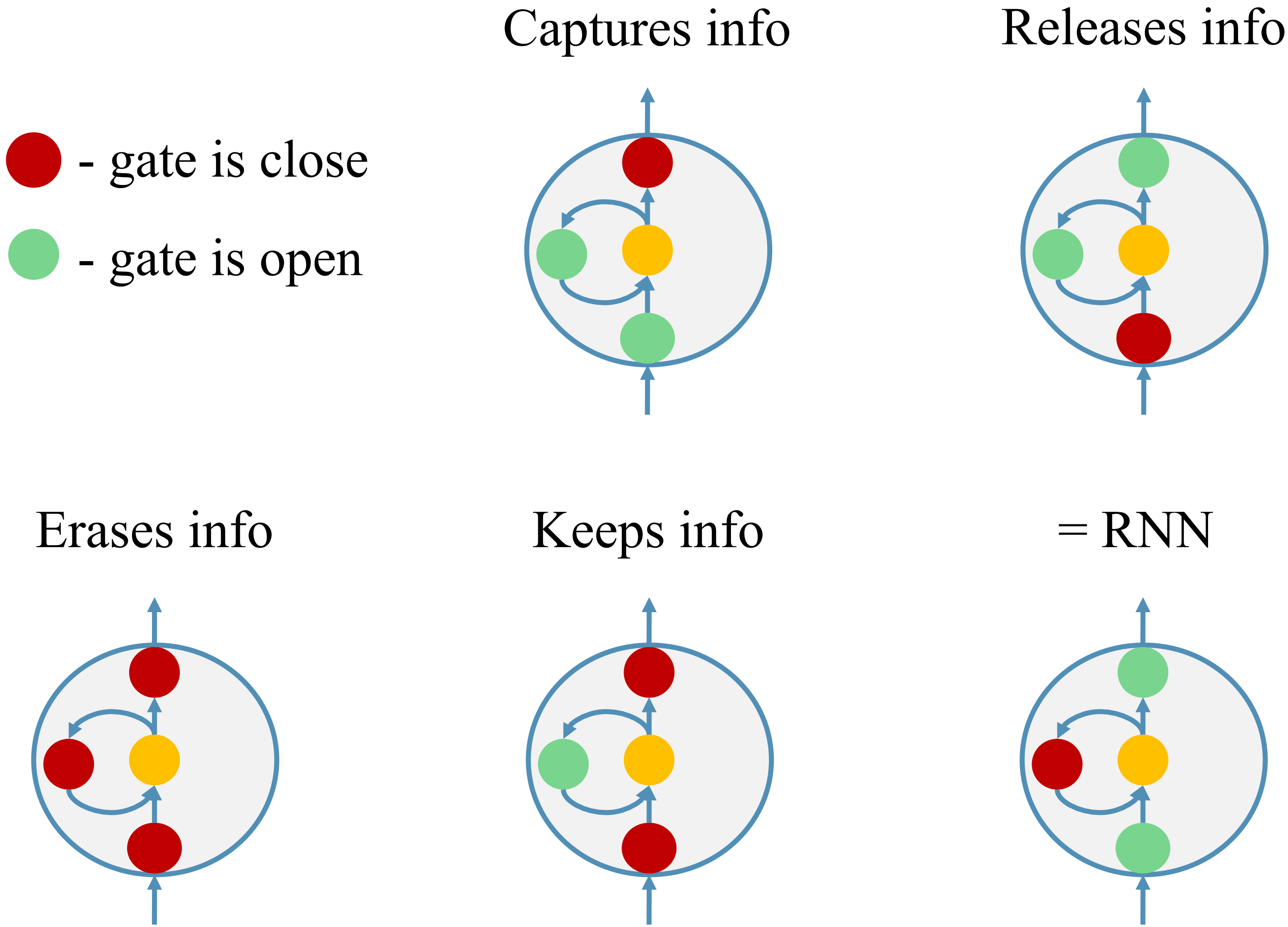
Keeps info



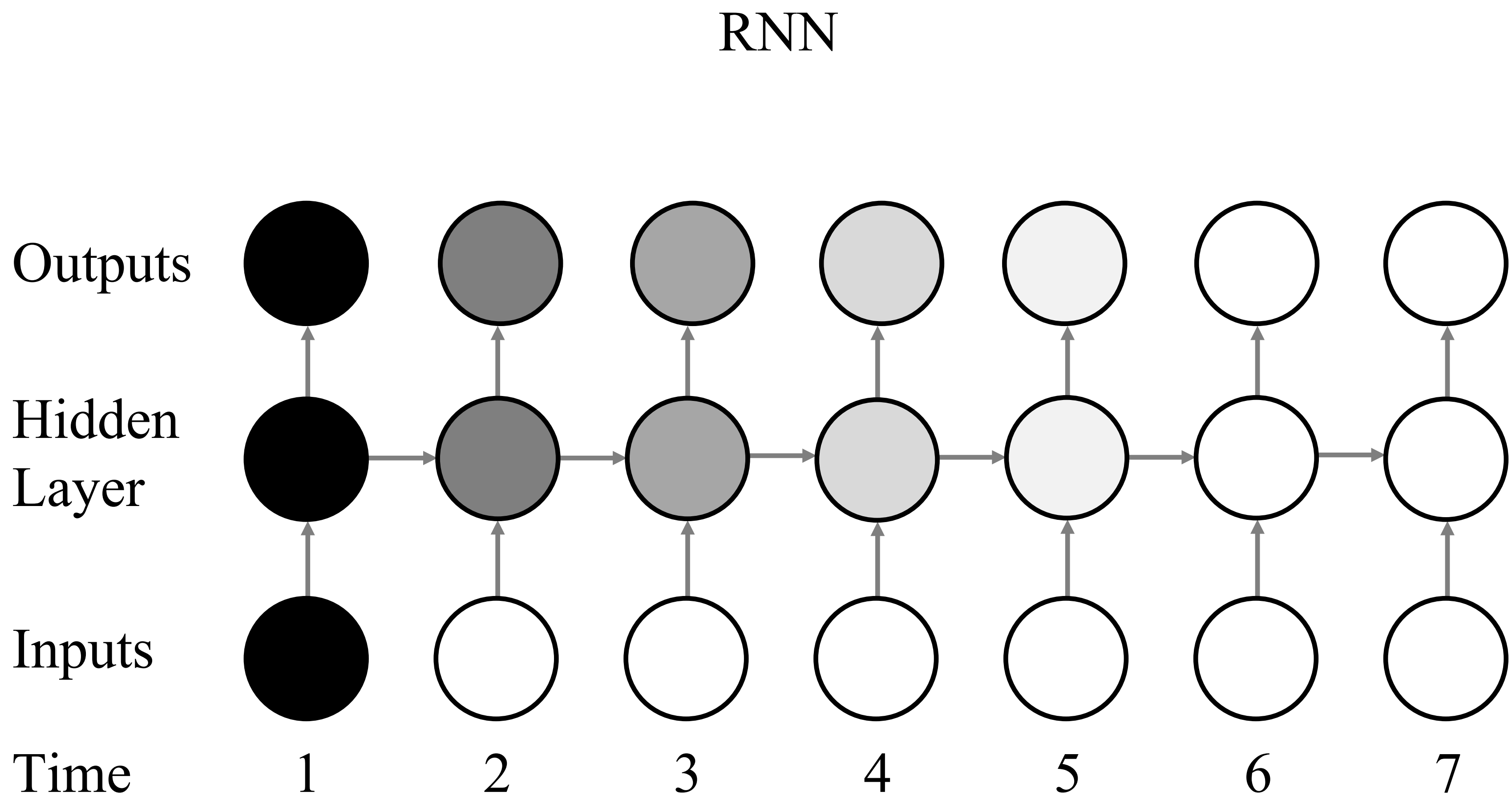
= RNN

?

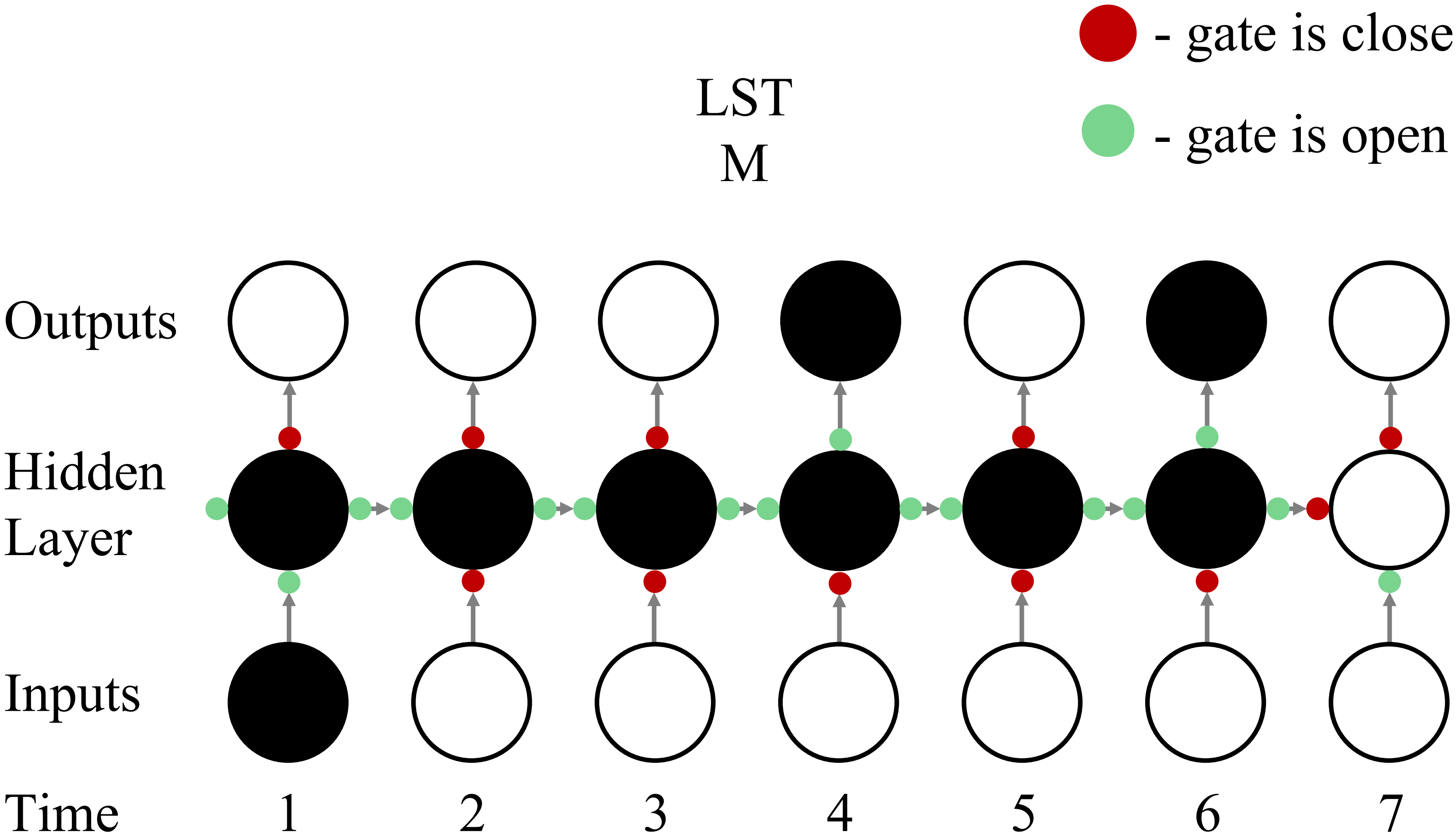
LSTM: extreme regimes



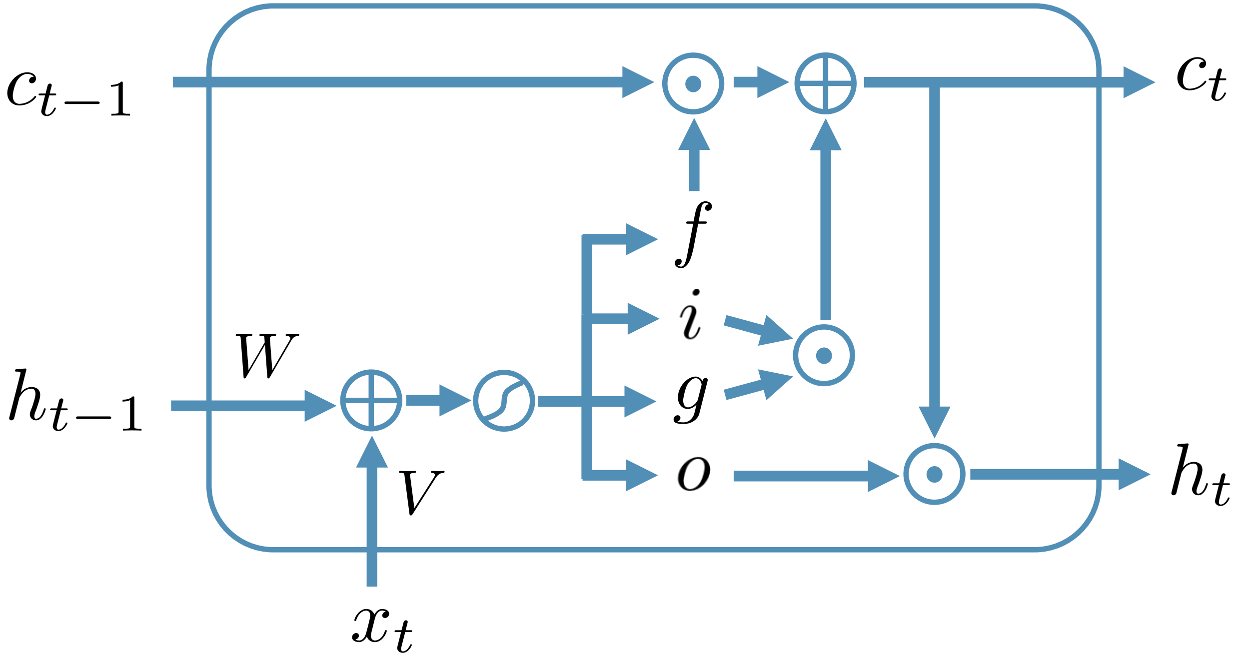
LSTM: information flow



LSTM: information flow



LSTM

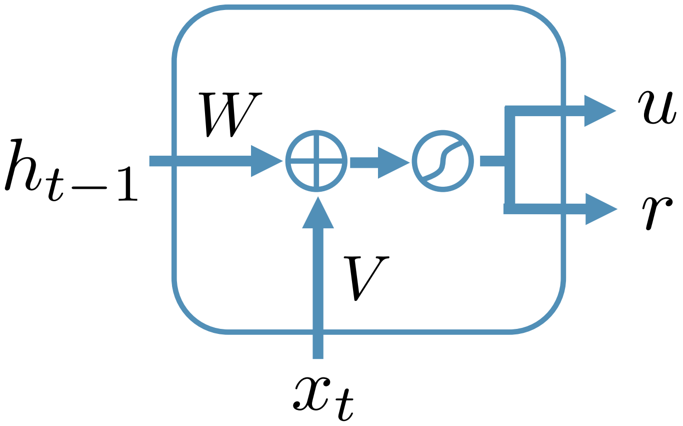


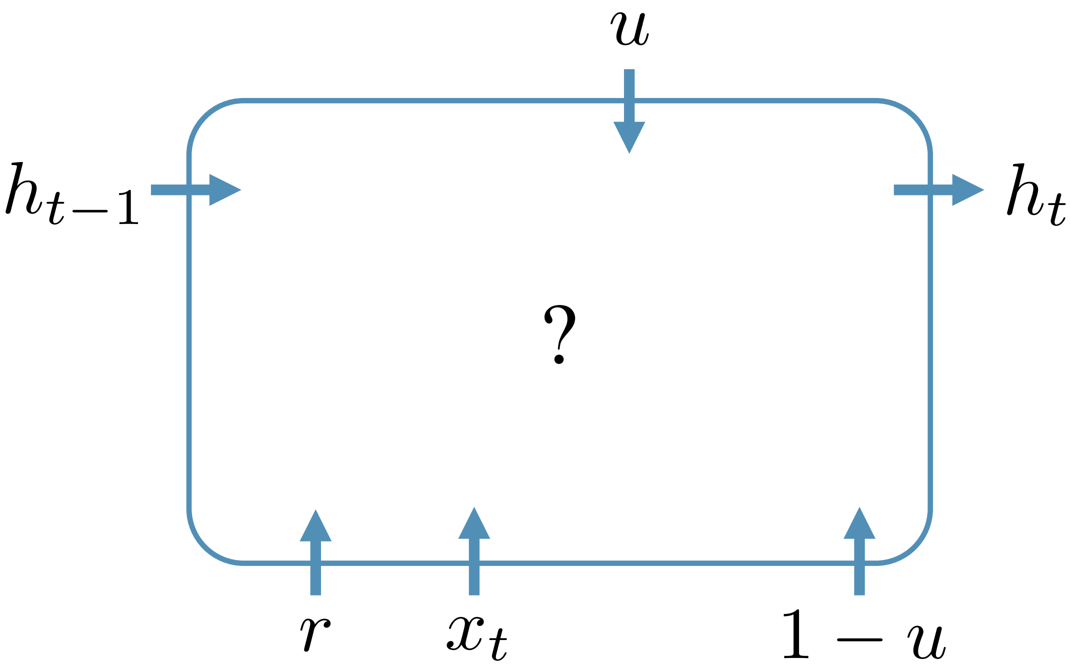
$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

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

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

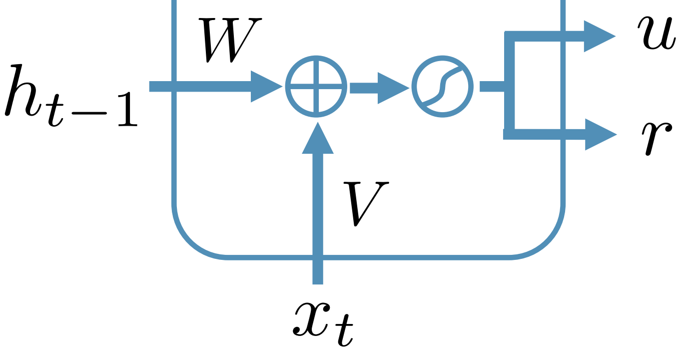
GRU

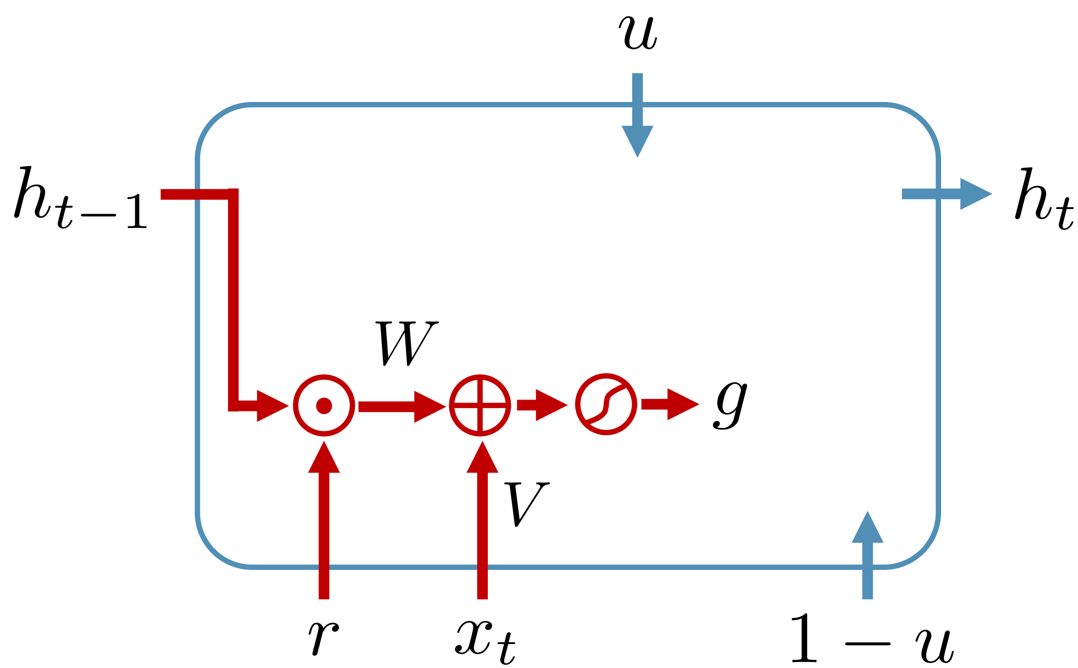




$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

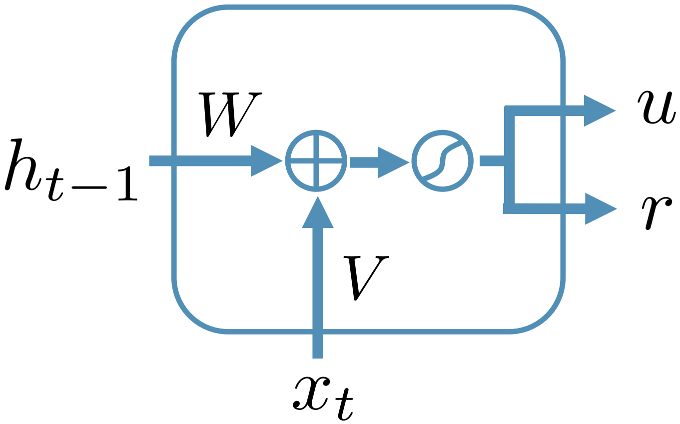
GRU

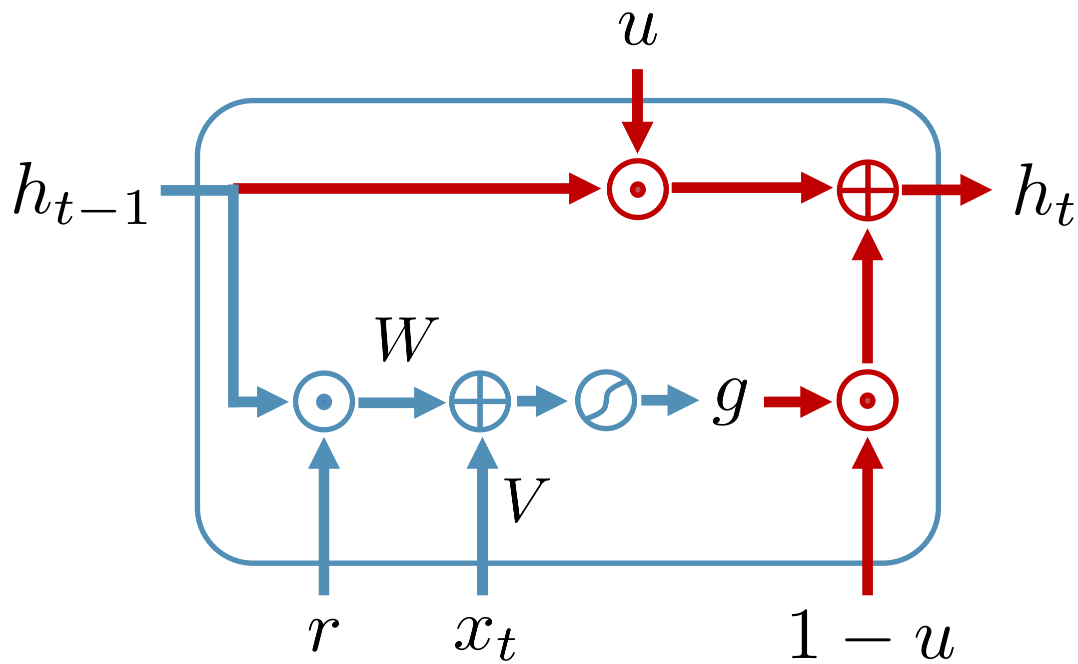




$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b) \qquad g_t = \tilde{f}\big(V_g x_t + W_g(h_{t-1} \cdot r_t) + b_g\big)$$

GRU



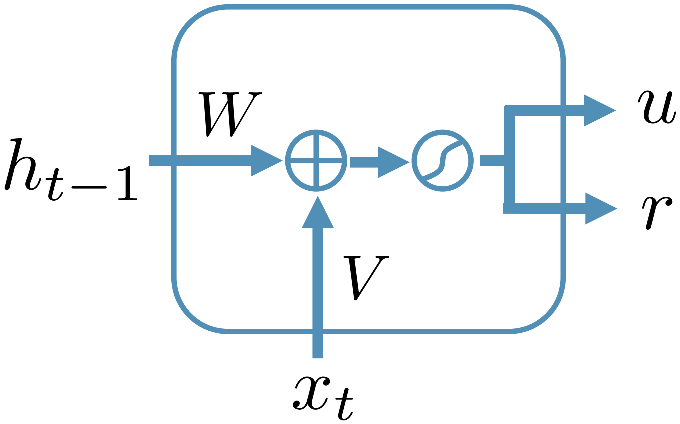


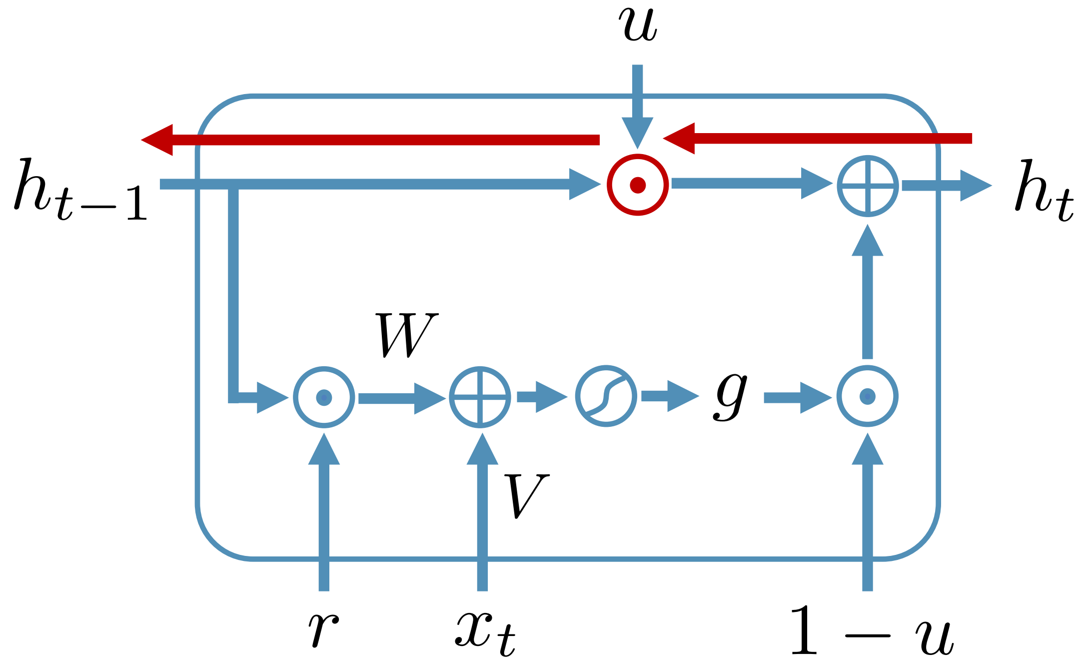
$$\begin{pmatrix} r_t \\ \color{red}{u_t} \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

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

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

GRU: vanishing gradients

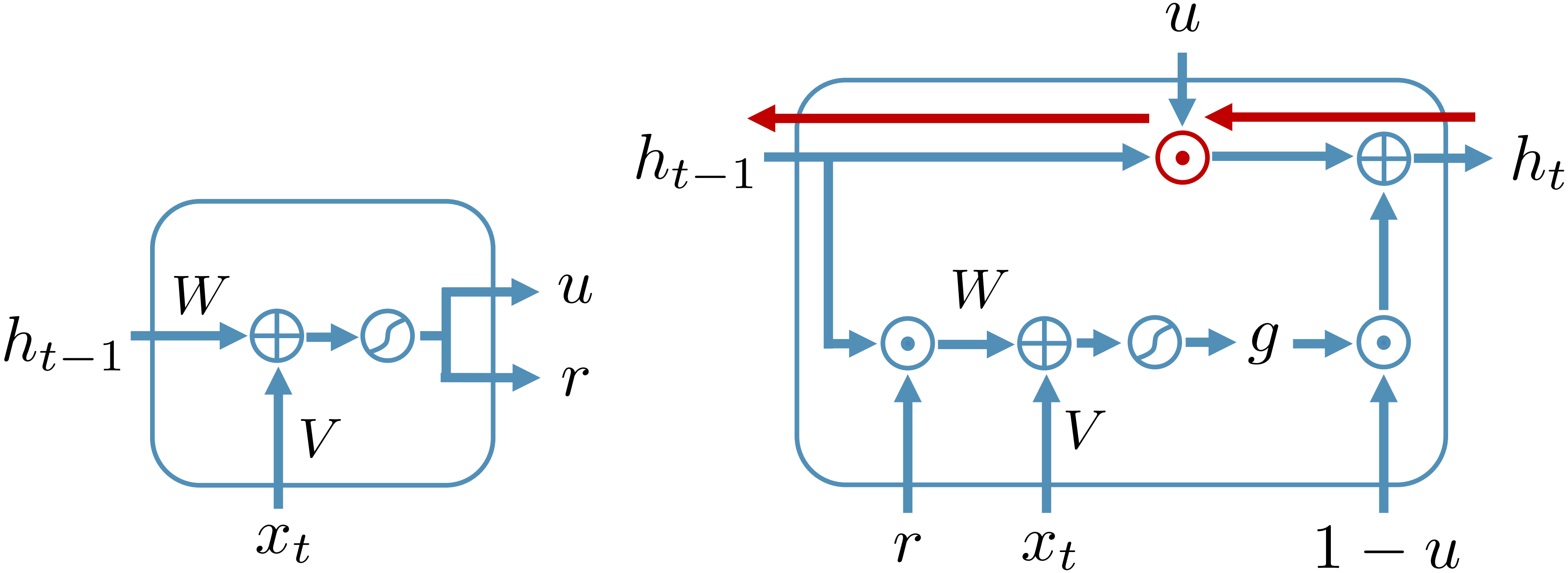




$$u_t = \sigma(V_u x_t + W_u h_{t-1} + b_u)$$

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

GRU: vanishing gradients



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

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

⇒ High initial b_u

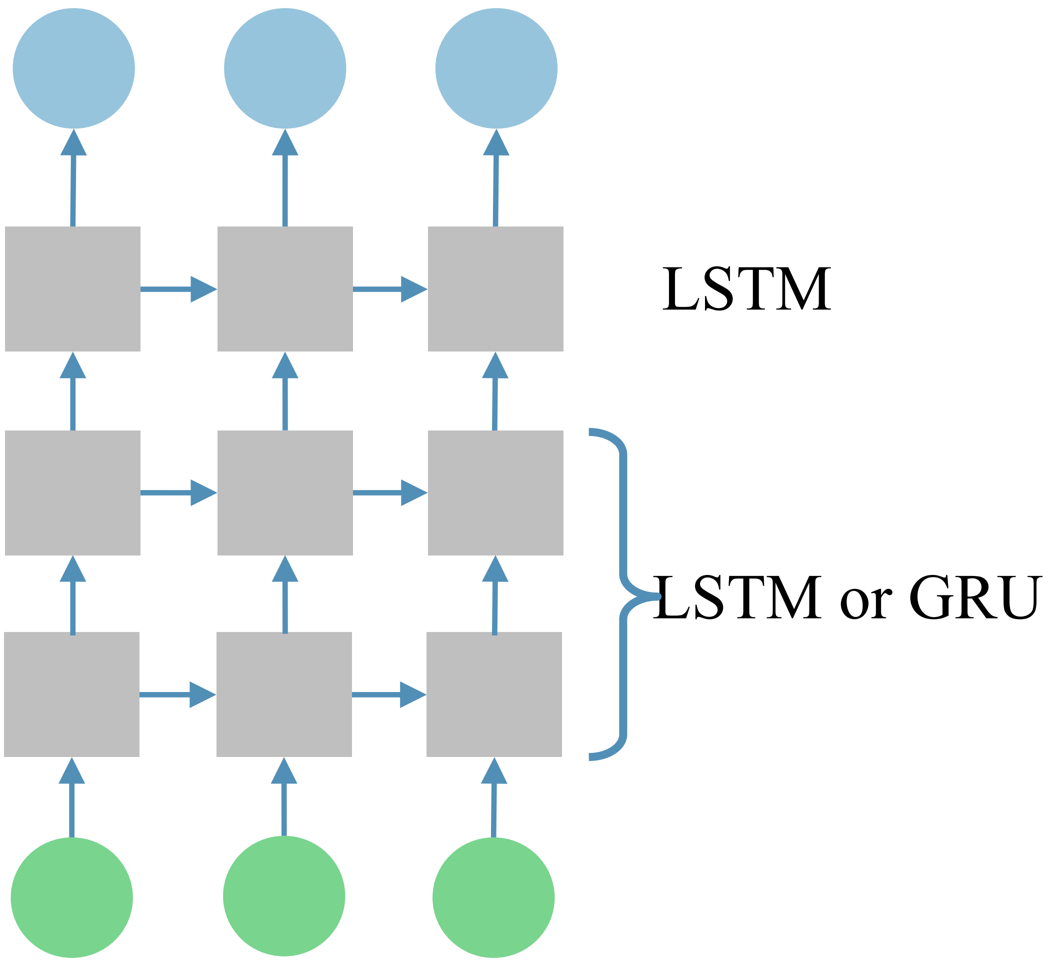
LSTM or GRU?

LSTM → more flexible

GRU → less parameters

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