Recurrent networks

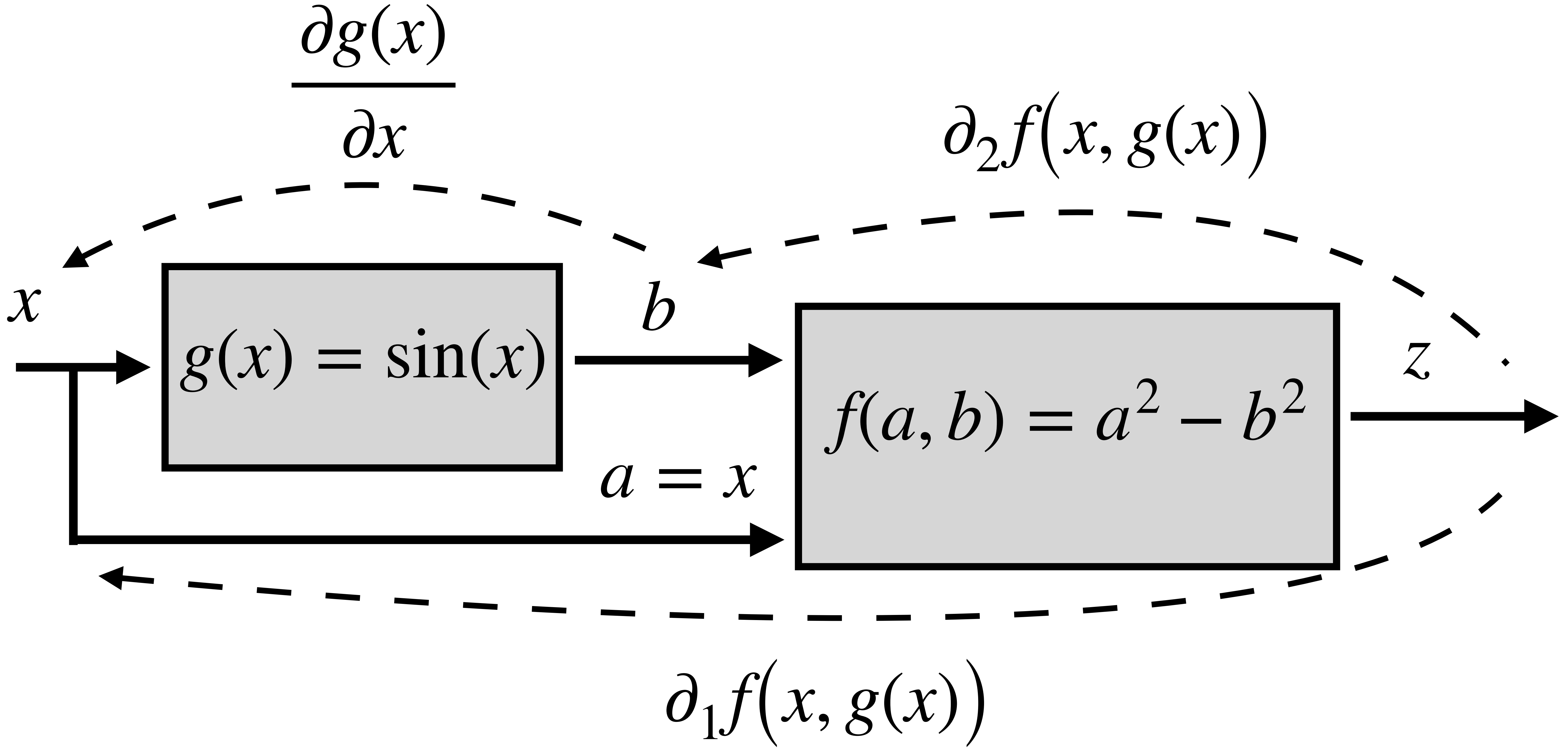
Recurrent nets unrolling in time

Karel Zimmermann
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Prerequisites: derivative of compound function

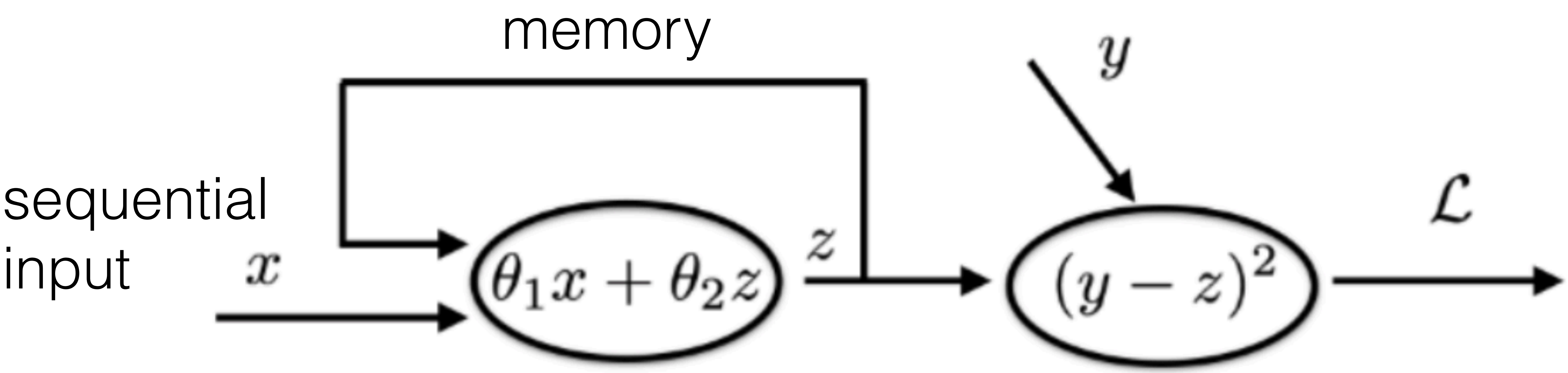
Introduce notation: $\dfrac{\partial f(a,b)}{\partial a} = \partial_1 f(a,b), \ \dfrac{\partial f(a,b)}{\partial b} = \partial_2 f(a,b)$

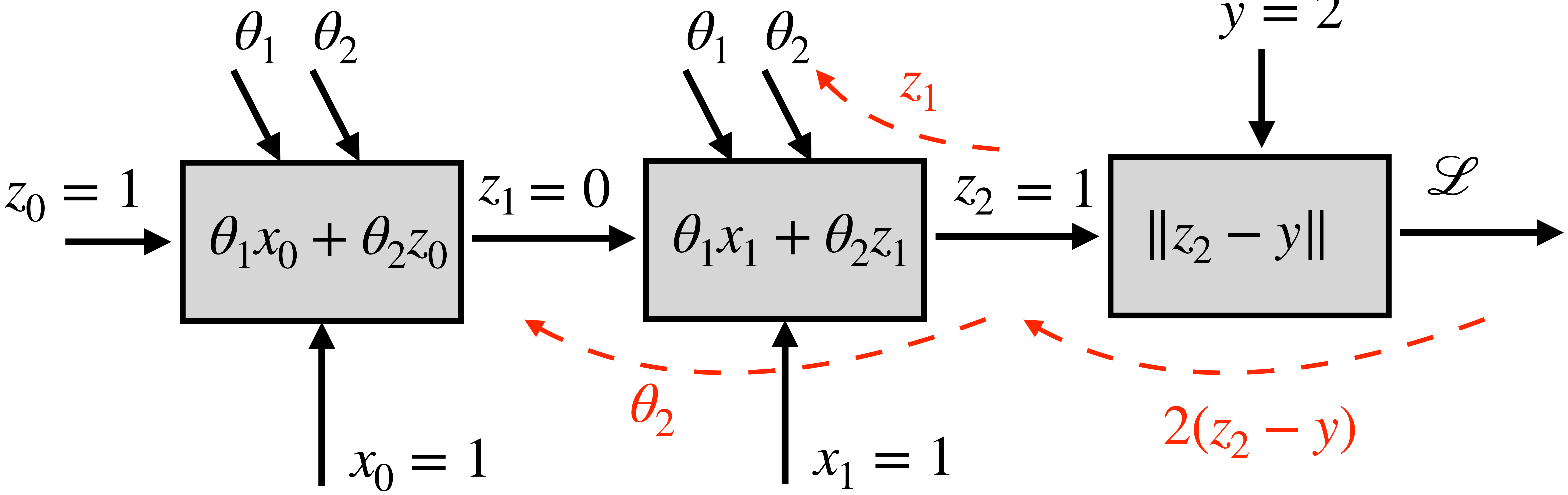
$$f(a,b) = a^2 - b^2 \,, \quad g(x) = \sin(x) \quad \Rightarrow \quad f\big(x, g(x)\big) = x^2 - \sin^2(x)$$



$$\frac{\partial f\big(x, g(x)\big)}{\partial x} = \partial_1 f\big(x, g(x)\big) + \partial_2 f\big(x, g(x)\big)\frac{\partial g(x)}{\partial x} = 2x - 2\sin(x)\cos(x)$$

RNN example with backprop





input sequence:

$x_0 = 1$

$x_1 = 1$

output label:

$y = 2$

initial memory:

$z_0 = 1$

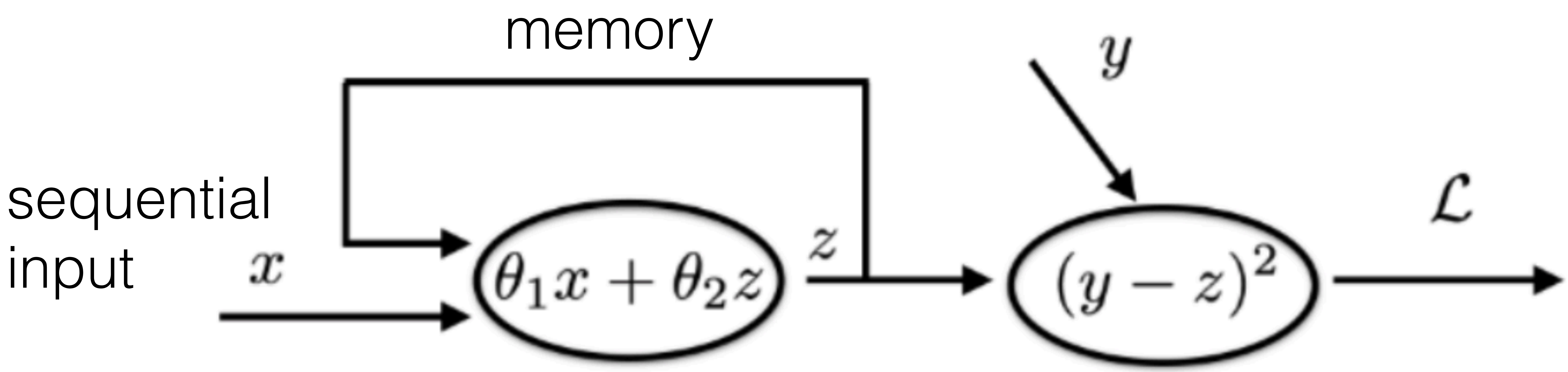
initial parameters:

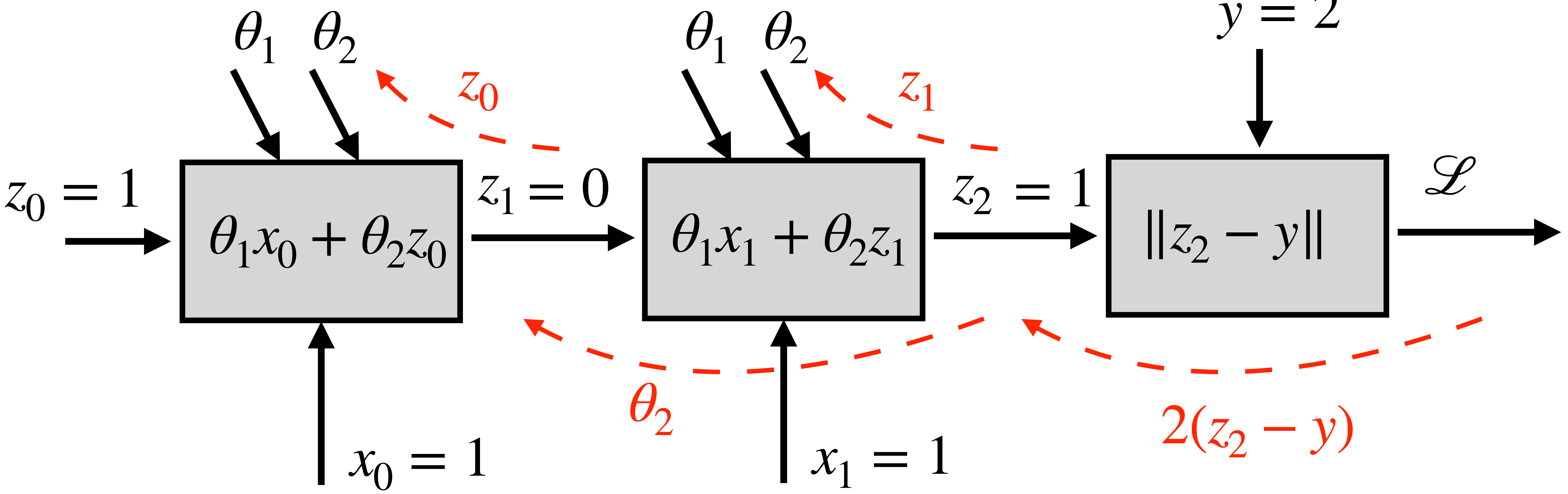
$\theta_1 = 1$

$\theta_2 = -1$

$$\frac{\partial \mathscr{L}}{\partial \theta_2} =$$

RNN example with backprop





input sequence:

$x_0 = 1$

$x_1 = 1$

output label:

$y = 2$

initial memory:

$z_0 = 1$

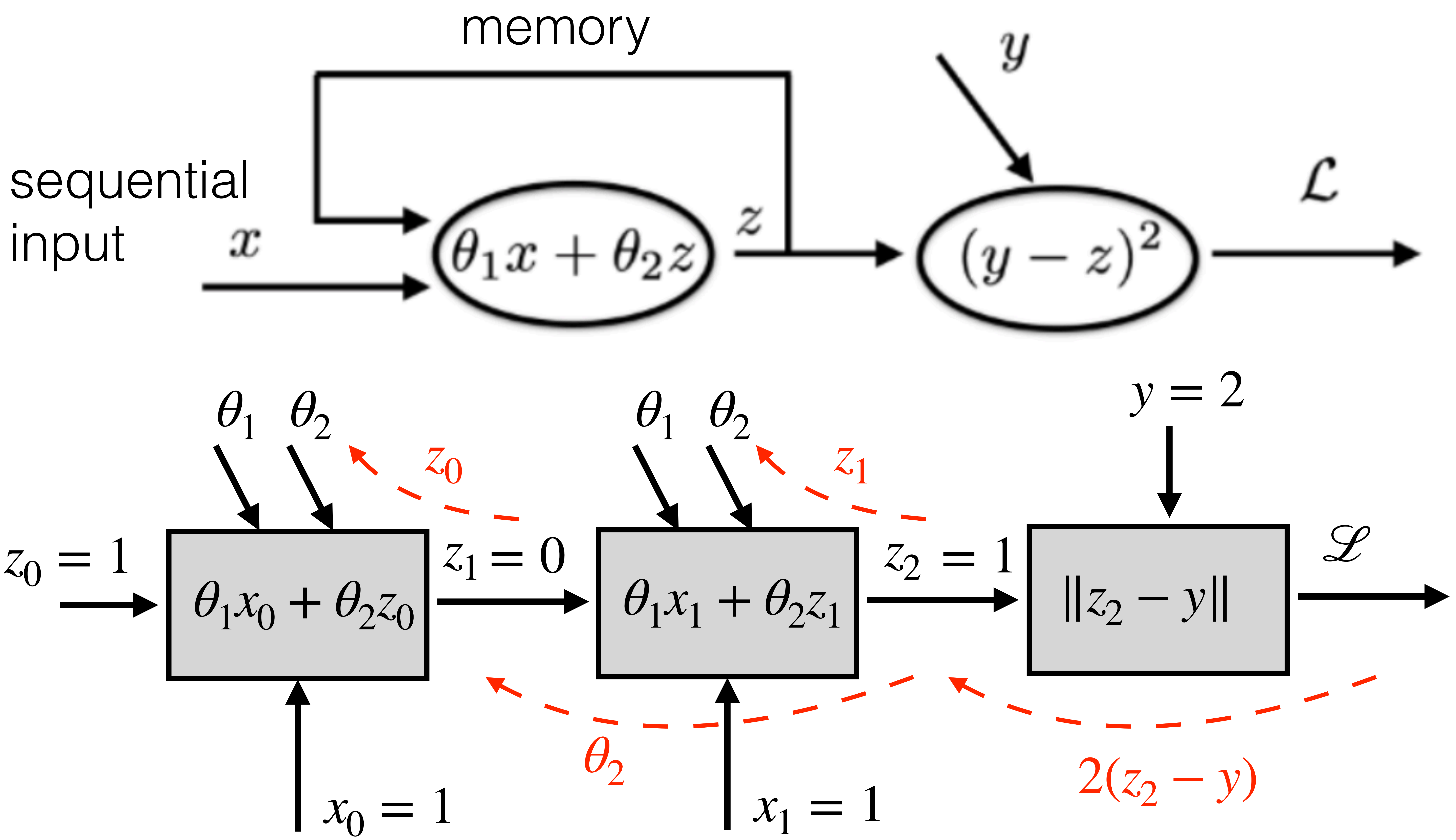
initial parameters:

$\theta_1 = 1$

$\theta_2 = -1$

$$\frac{\partial \mathscr{L}}{\partial \theta_2} =$$

RNN example with backprop



input sequence:

$x_0 = 1$

$x_1 = 1$

output label:

$y = 2$

initial memory:

$z_0 = 1$

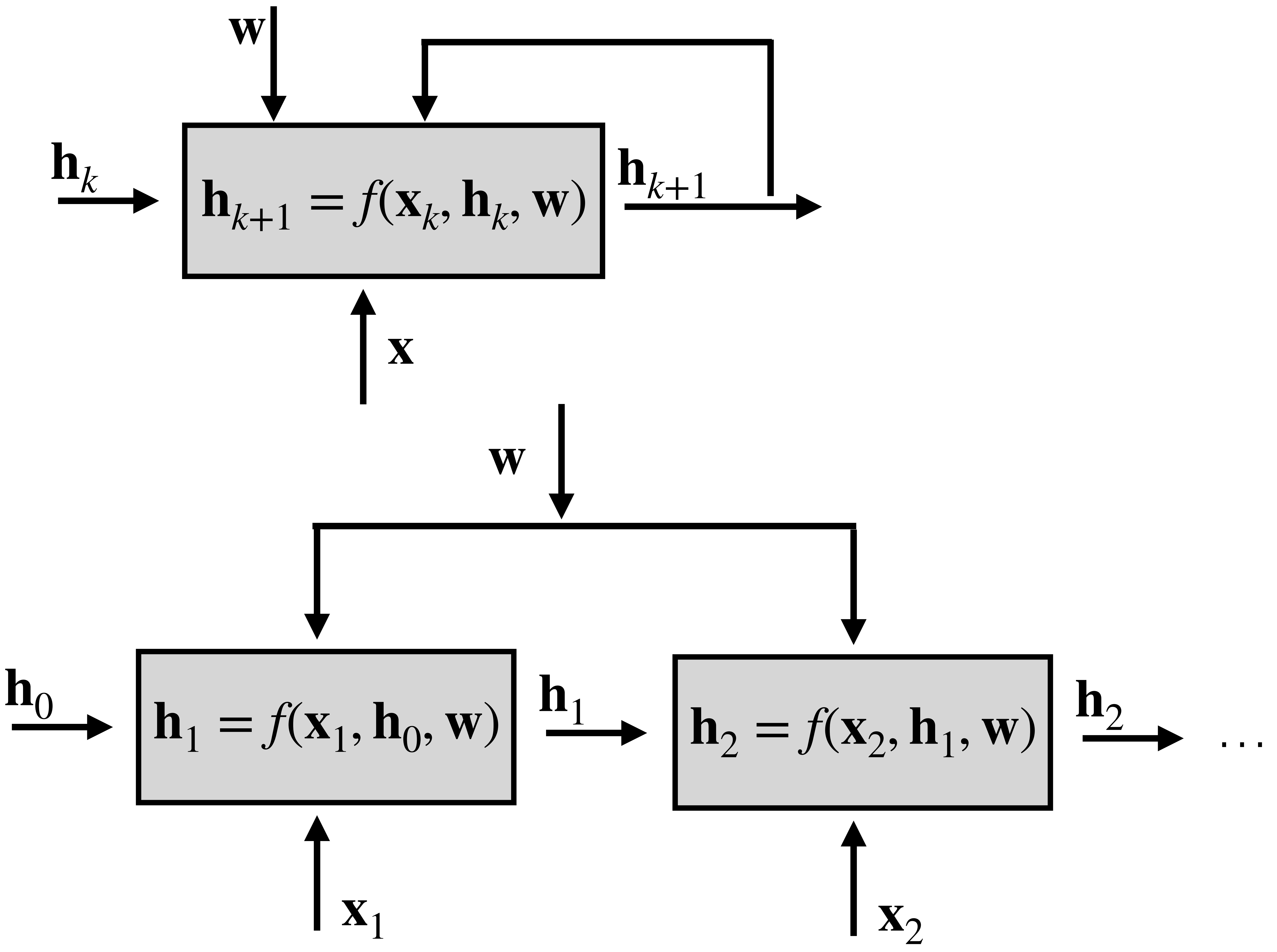
initial parameters:

$\theta_1 = 1$

$\theta_2 = -1$

$$\frac{\partial \mathscr{L}}{\partial \theta_2} = 2(z_2 - y)\, z_1 + 2(z_2 - y)\ \theta_2\, z_0 = 2(1-2)0 + 2(1-2)(-1)1 = 2$$

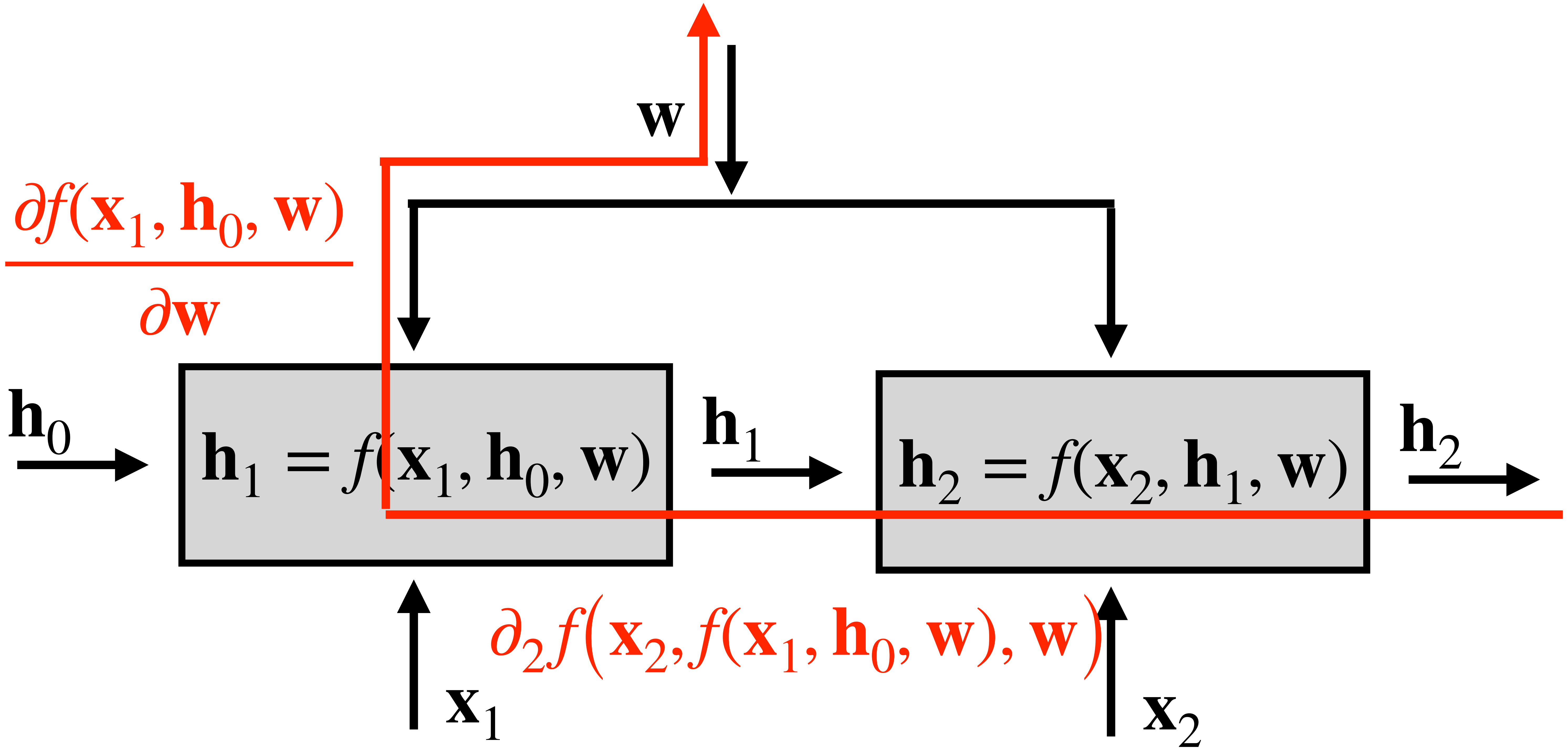
Recurrence resolved by unrolling the computation graph in time



Recurrence resolved by unrolling the computation graph in time

$$\mathbf{h}_1 = f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \quad \mathbf{h}_2 = f(\mathbf{x}_2, \mathbf{h}_1, \mathbf{w}) \qquad \Rightarrow \qquad \mathbf{h}_2 = f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big)$$

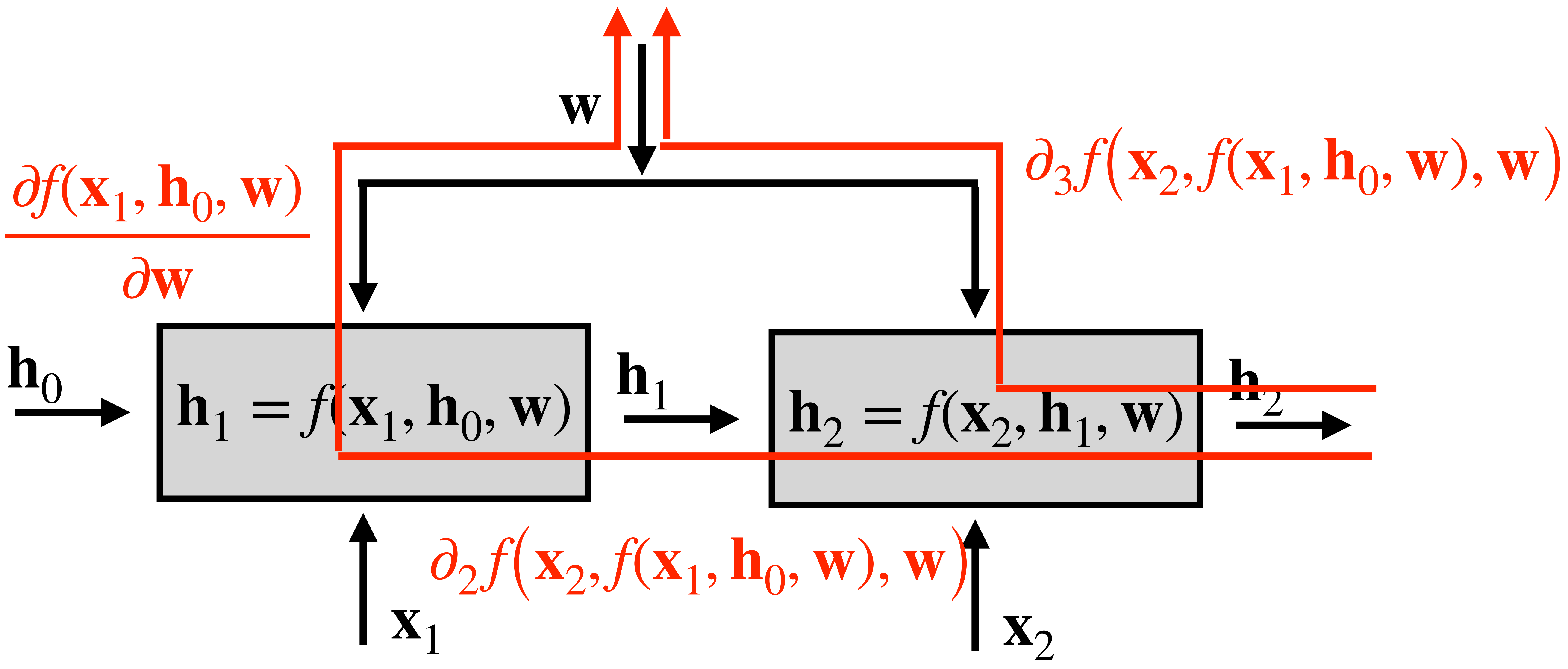
$$\frac{\partial f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big)}{\partial \mathbf{w}} = \partial_2 f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big) \frac{\partial f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w})}{\partial \mathbf{w}}$$



Recurrence resolved by unrolling the computation graph in time

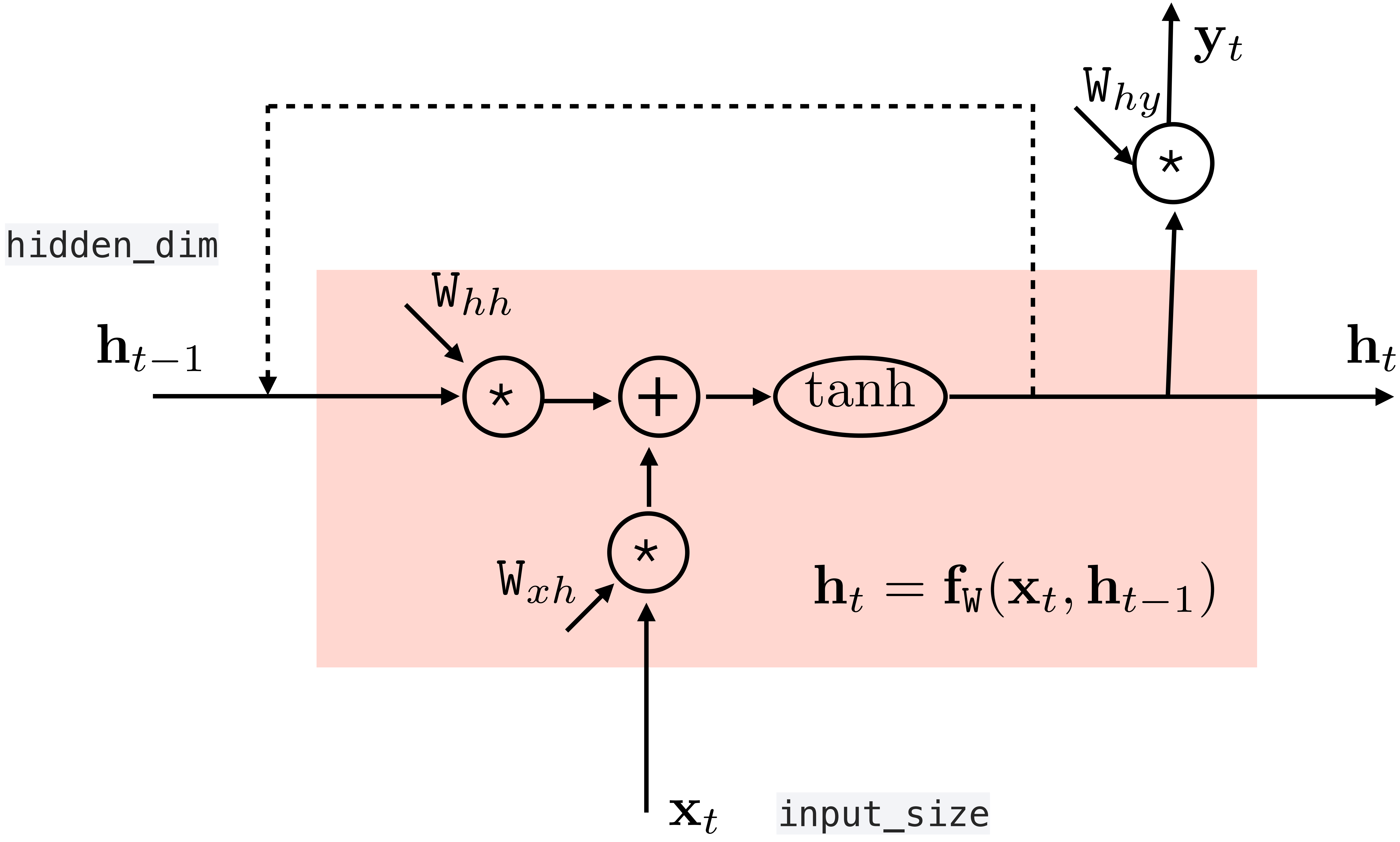
$$\mathbf{h}_1 = f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \quad \mathbf{h}_2 = f(\mathbf{x}_2, \mathbf{h}_1, \mathbf{w}) \qquad \Rightarrow \qquad \mathbf{h}_2 = f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big)$$

$$\frac{\partial f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big)}{\partial \mathbf{w}} = \partial_2 f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big) \frac{\partial f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w})}{\partial \mathbf{w}} + \partial_3 f\big(\mathbf{x}_2, f(\mathbf{x}_1, \mathbf{h}_0, \mathbf{w}), \mathbf{w}\big)$$



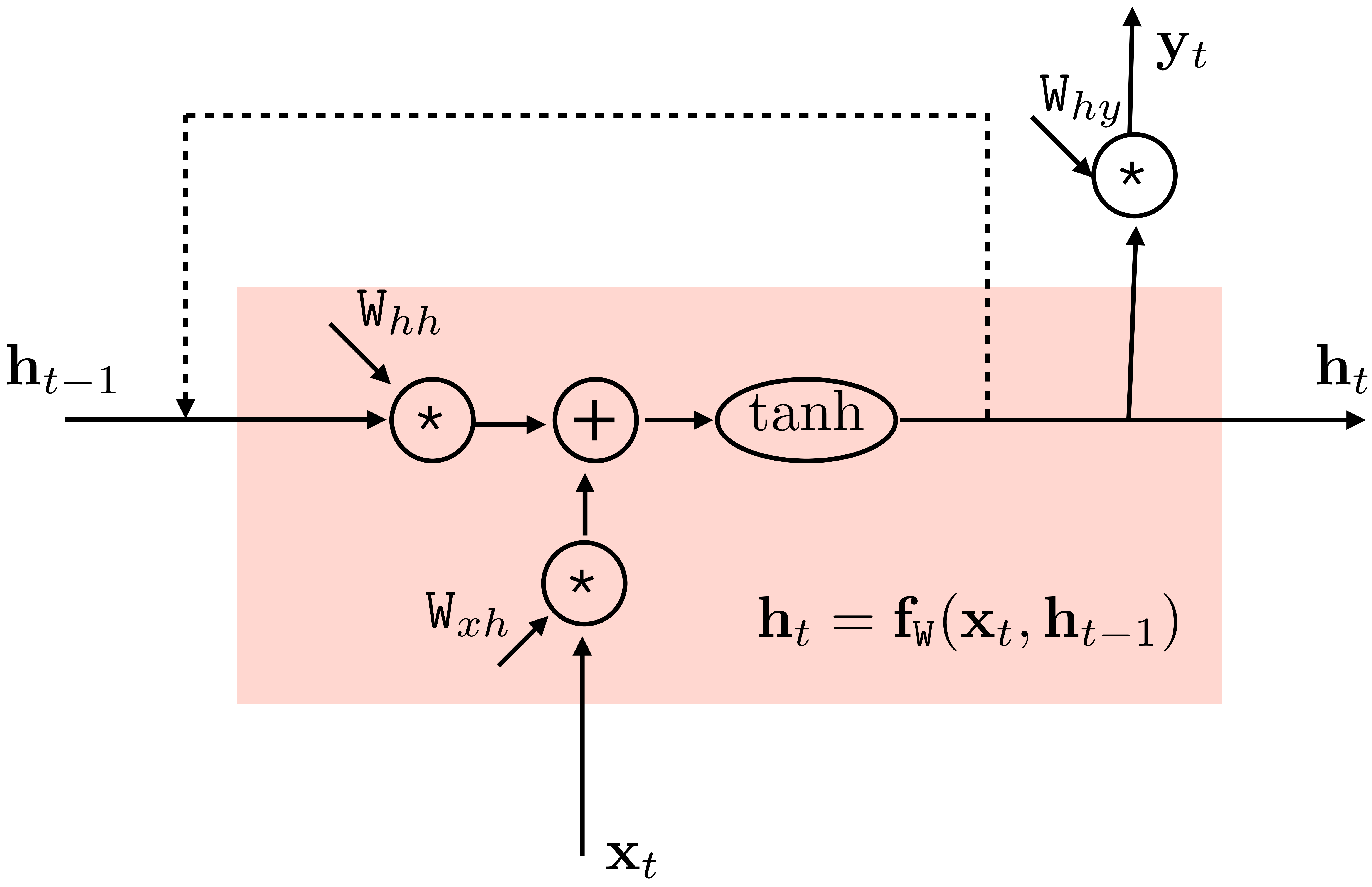
Simple recurrent block

```
torch.nn.RNN(input_size, hidden_dim, n_layers)
```

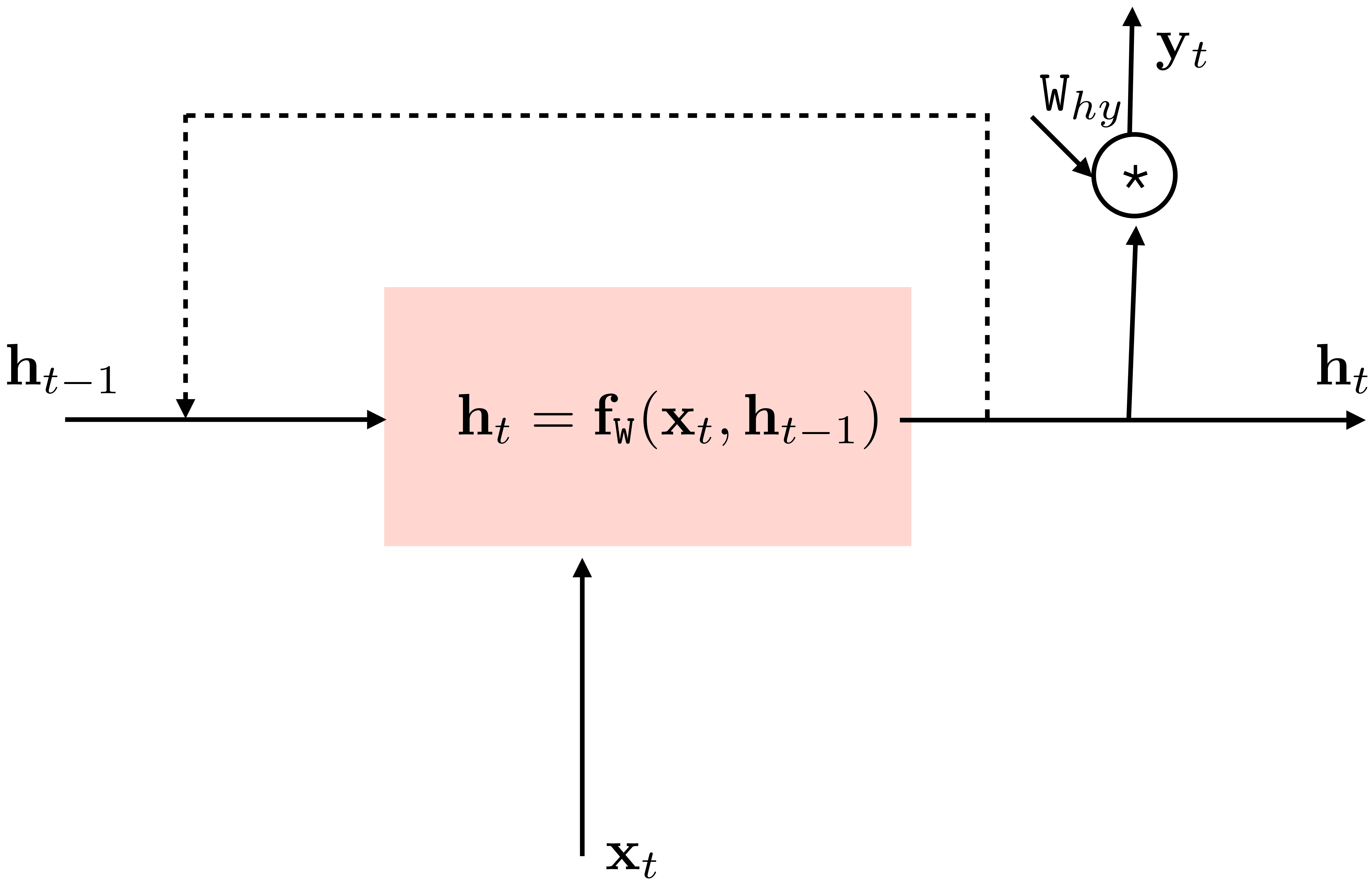


PyTorch: https://pytorch.org/docs/stable/nn.html

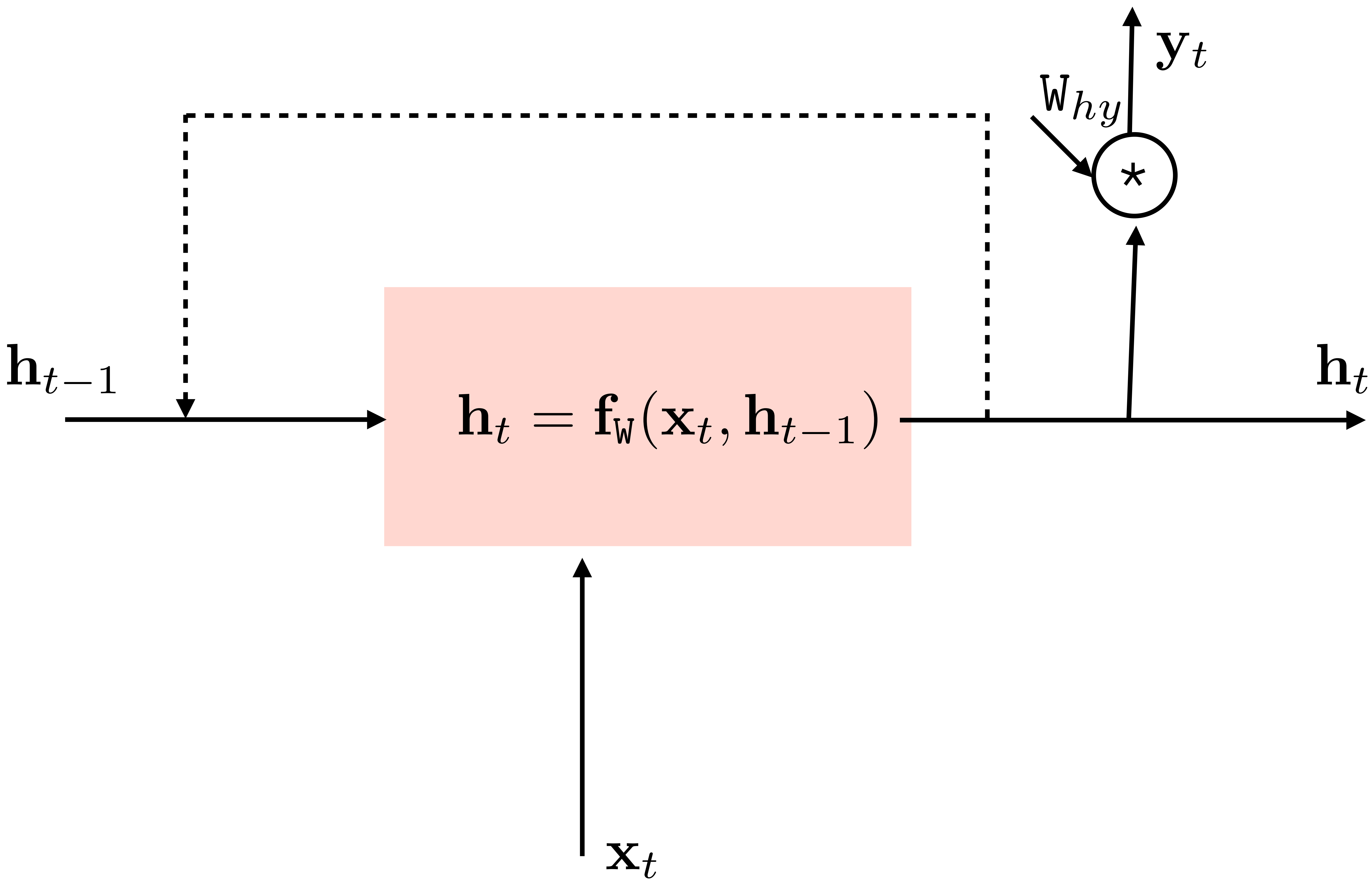
Simple recurrent block - feed-forward pass



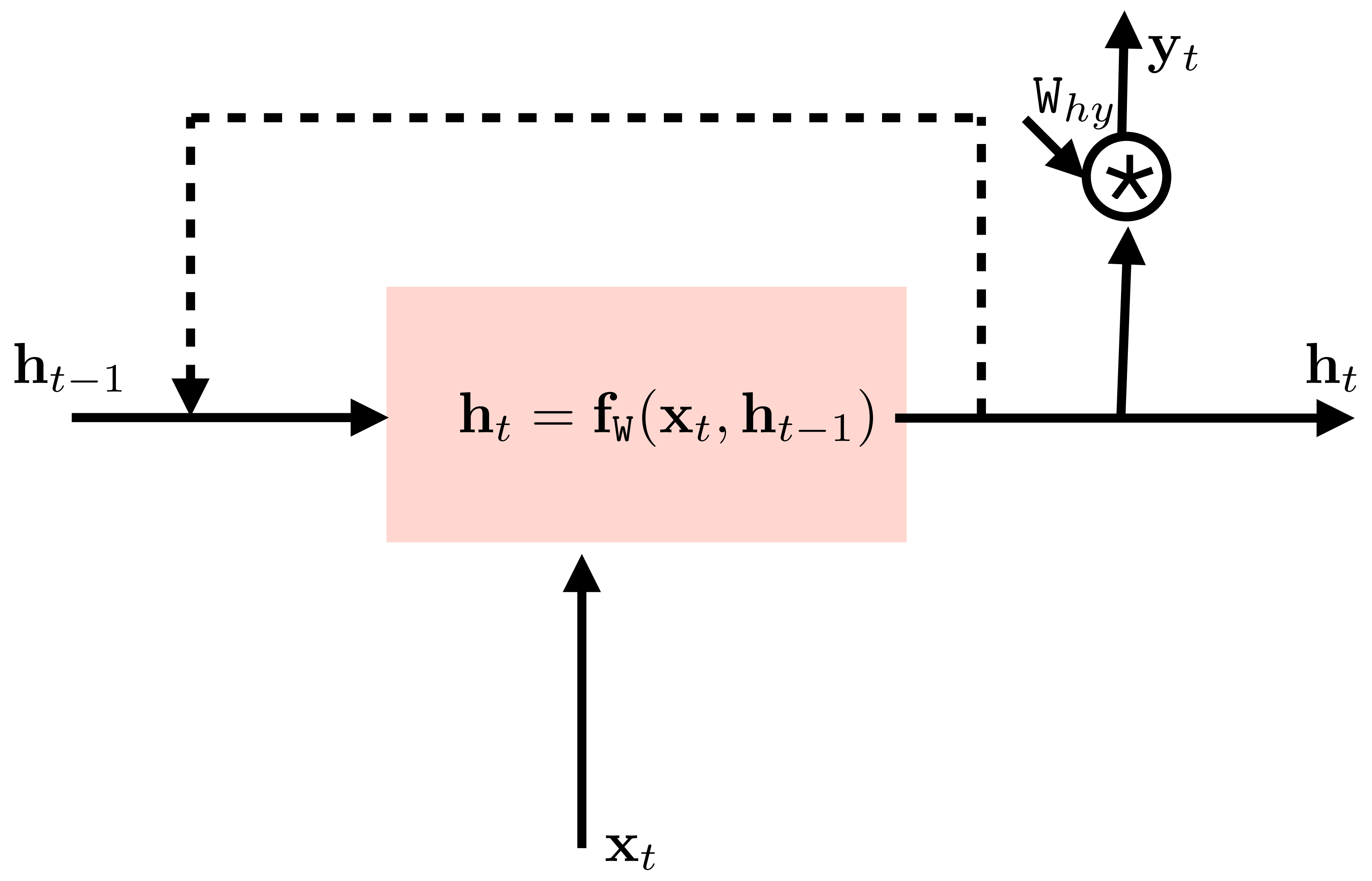
Simple recurrent block - feed-forward pass



Simple recurrent block - feed-forward pass



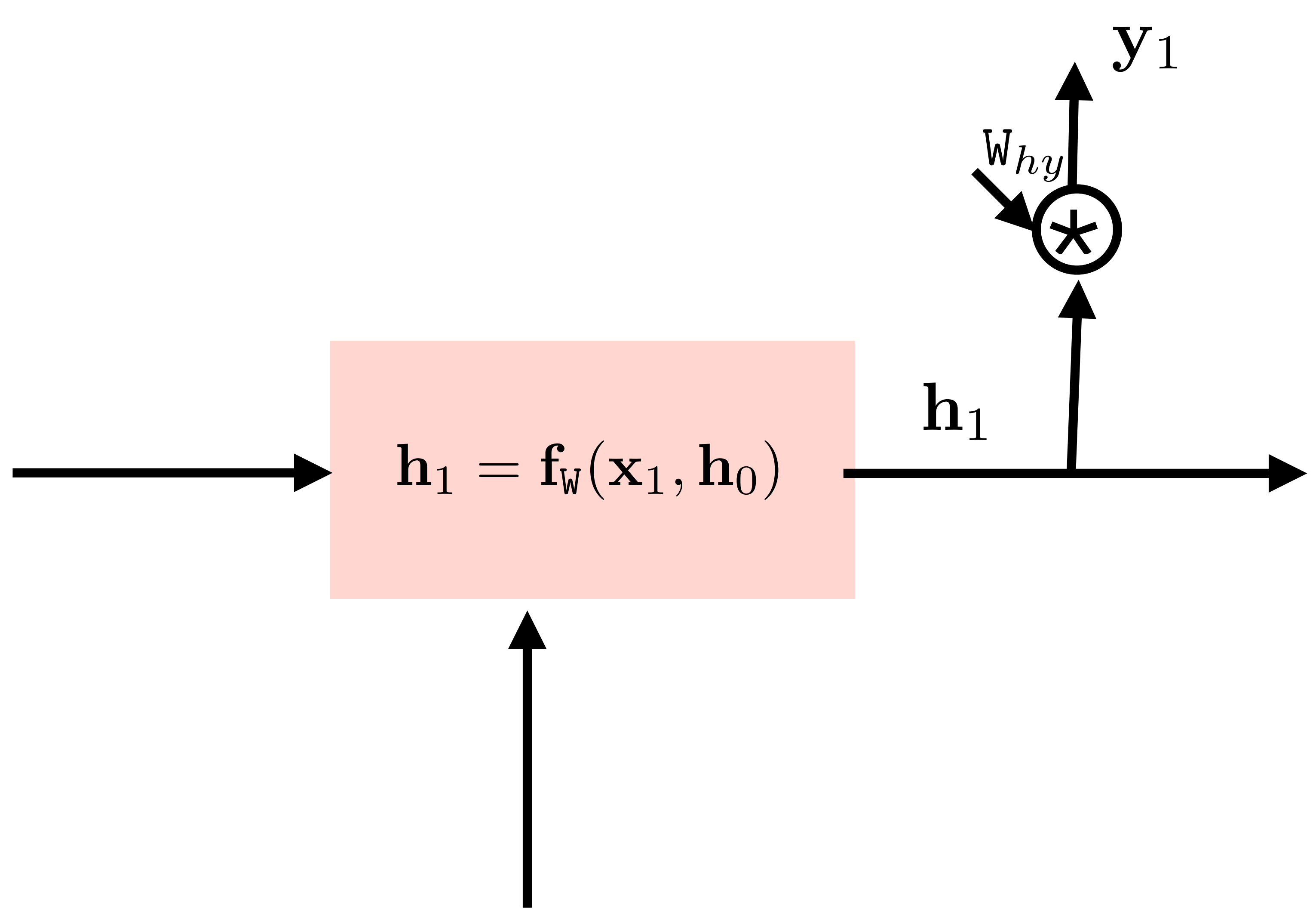
Simple recurrent block - feed-forward pass



Given a finite input sequence: $\mathbf{h}_0 \;\; \mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \mathbf{x}_3$
we remove the recurrent connection by:

- successive substitution of inputs and
- unrolling the net

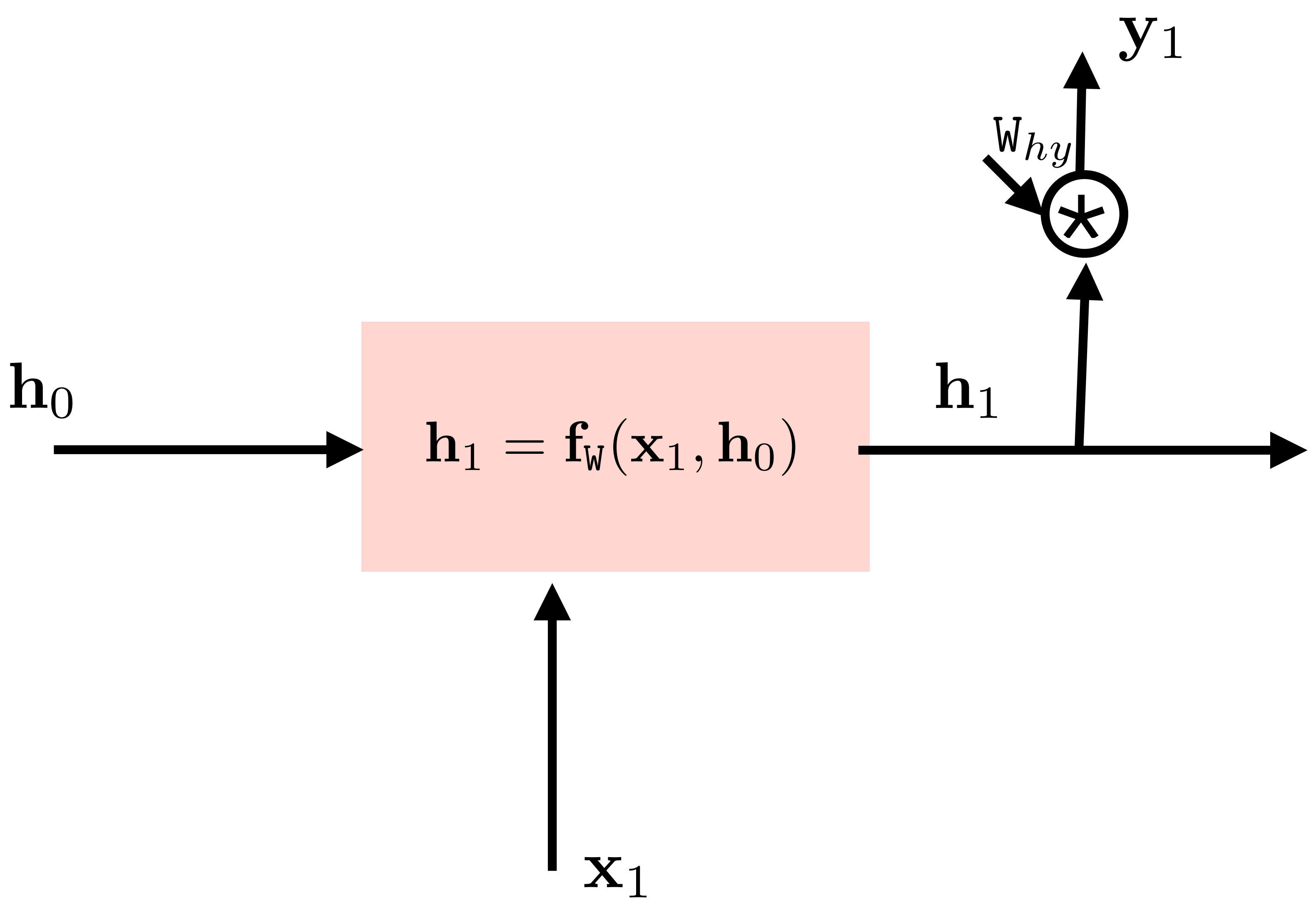
Simple recurrent block - feed-forward pass



Given a finite input sequence: $\mathbf{h}_0 \ \mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3$
we remove the recurrent connection by:

- successive substitution of inputs and
- unrolling the net

Simple recurrent block - feed-forward pass

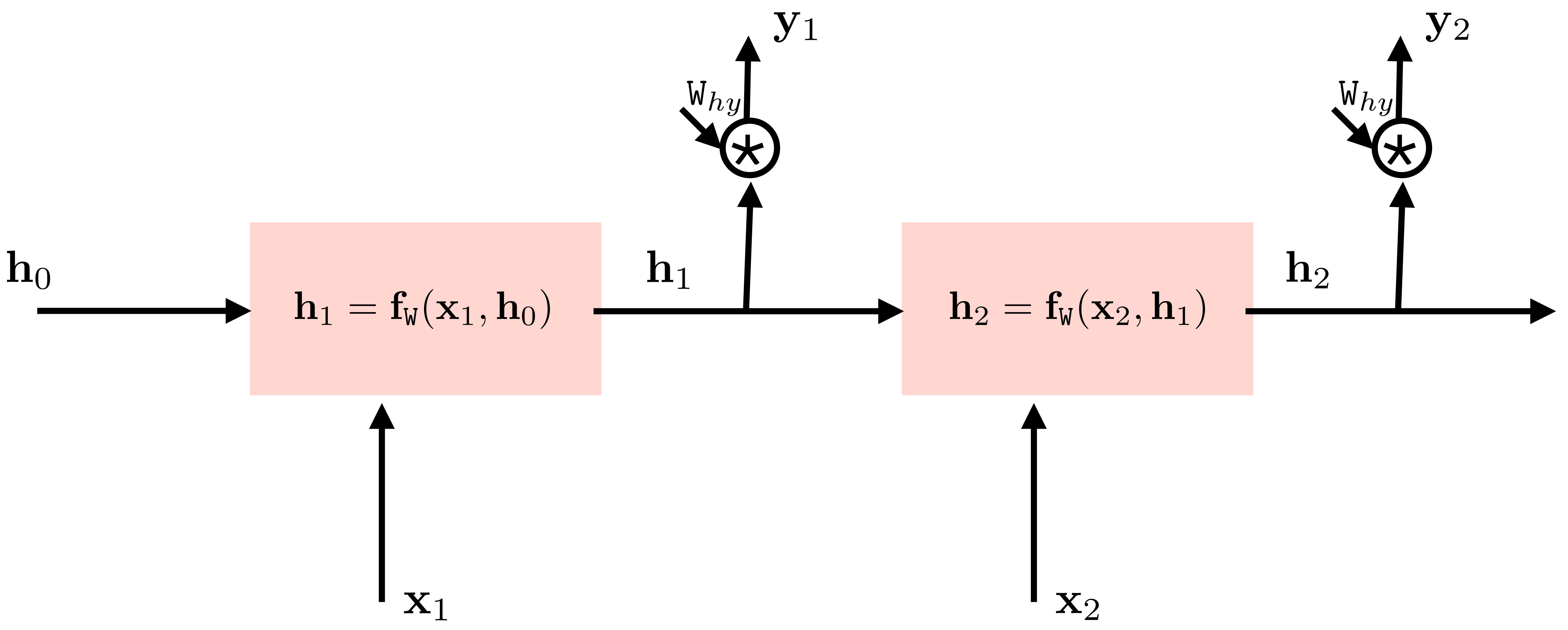


Given a finite input sequence: $\mathbf{x}_2$ $\mathbf{x}_3$

we remove the recurrent connection by:

- successive substitution of inputs and
- unrolling the net

Simple recurrent block - feed-forward pass

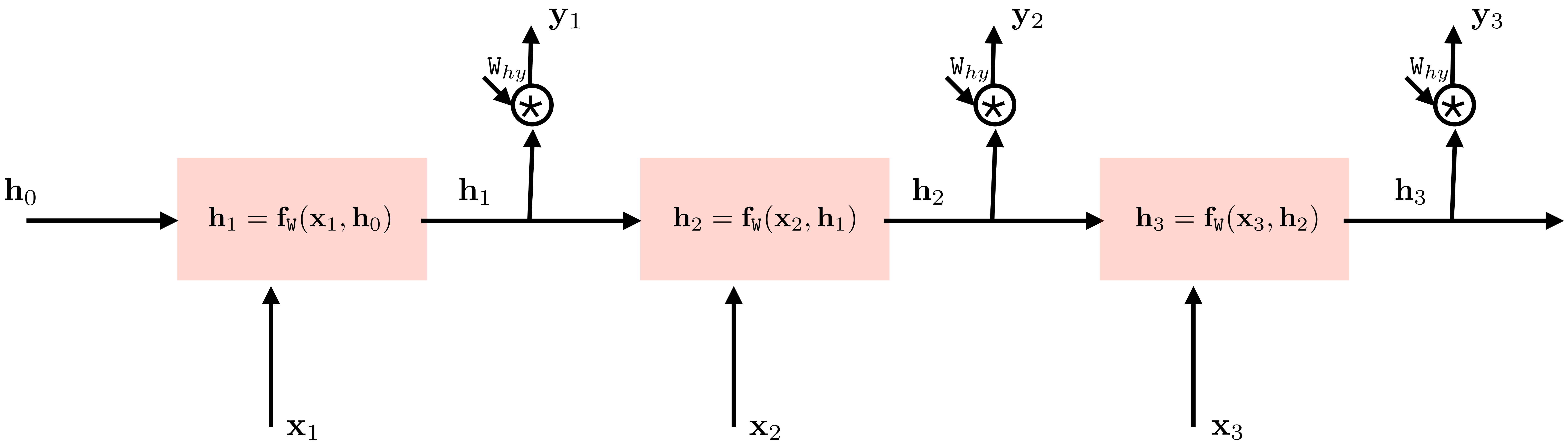


Given a finite input sequence: $\mathbf{x}_3$

we remove the recurrent connection by:

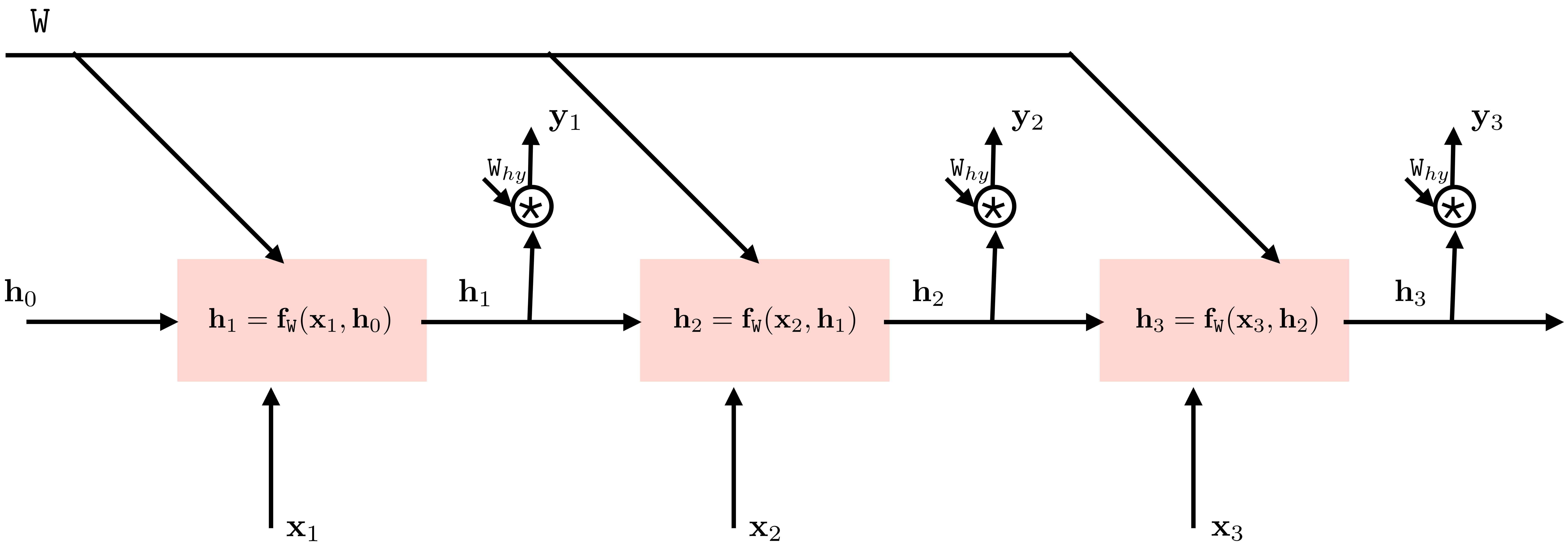
- successive substitution of inputs and
- unrolling the net

Simple recurrent block - feed-forward pass



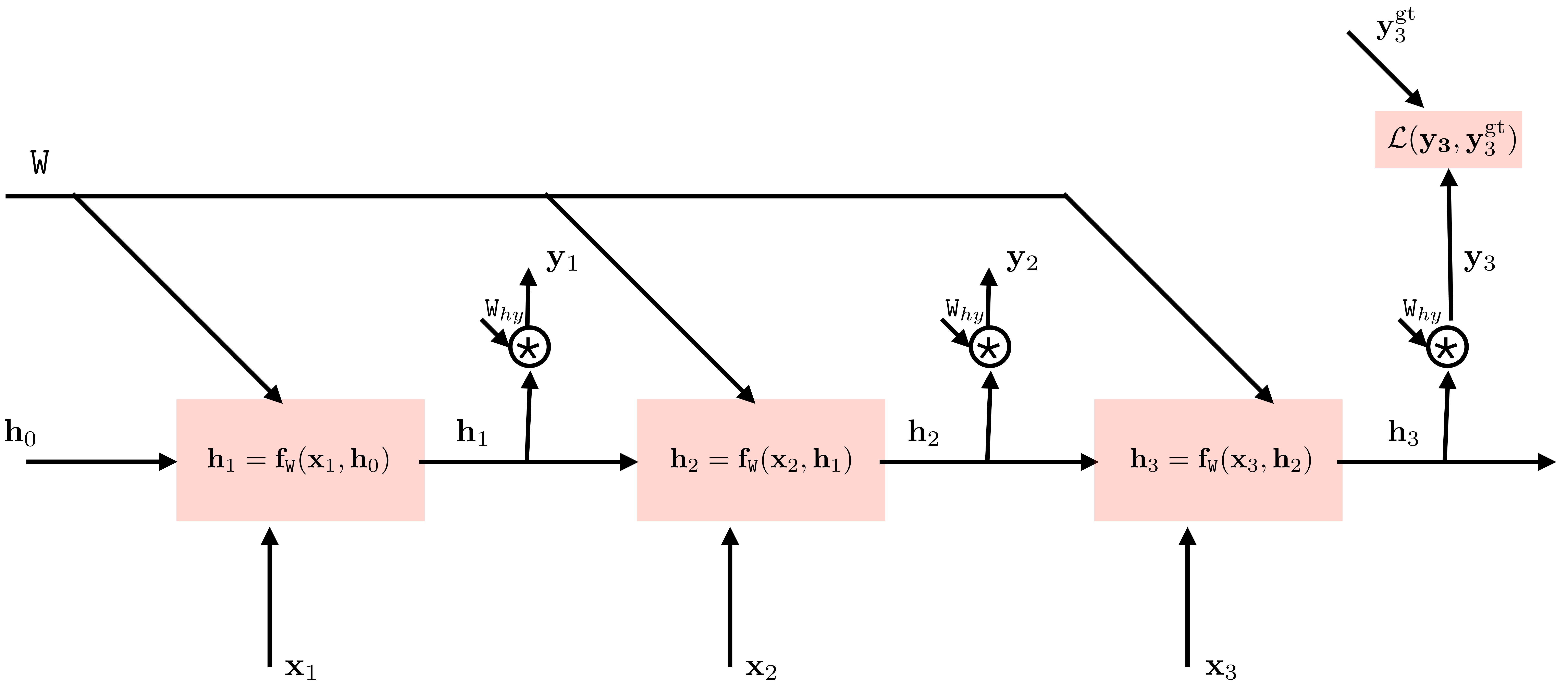
- **Unrolled** computational graph:
  - it is normal feedforward network

Simple recurrent block - feed-forward pass



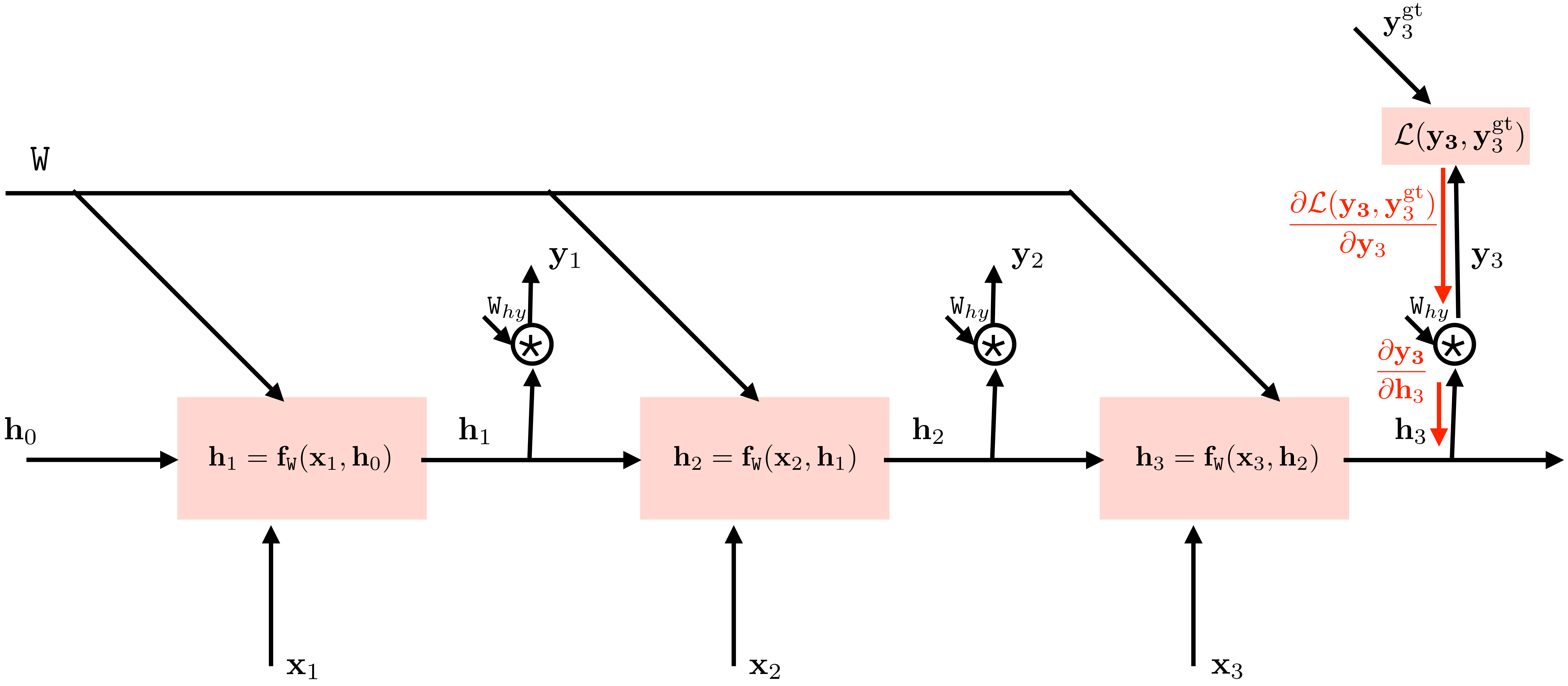
- **Unrolled** computational graph:
  - it is normal feedforward network
  - it consists of several **same blocks** with the **same weights**!

Simple recurrent block - backward pass



- **Loss function** could be connected to all outputs in all times
- We connect the **loss** to the **final output** only (for the simplicity)

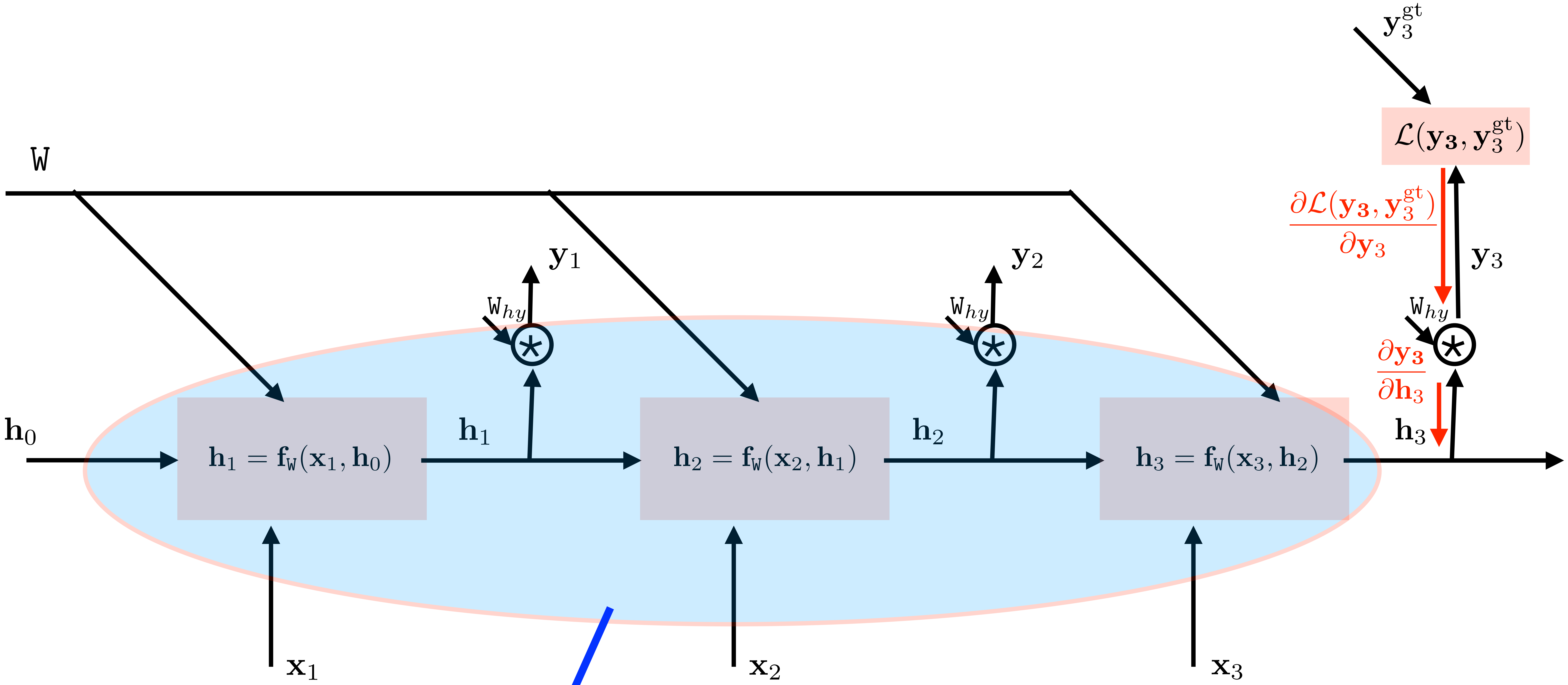
Simple recurrent block - backward pass



- Backprop:

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\text{gt}})}{\partial \mathtt{W}} = \quad ?$$

Simple recurrent block - backward pass

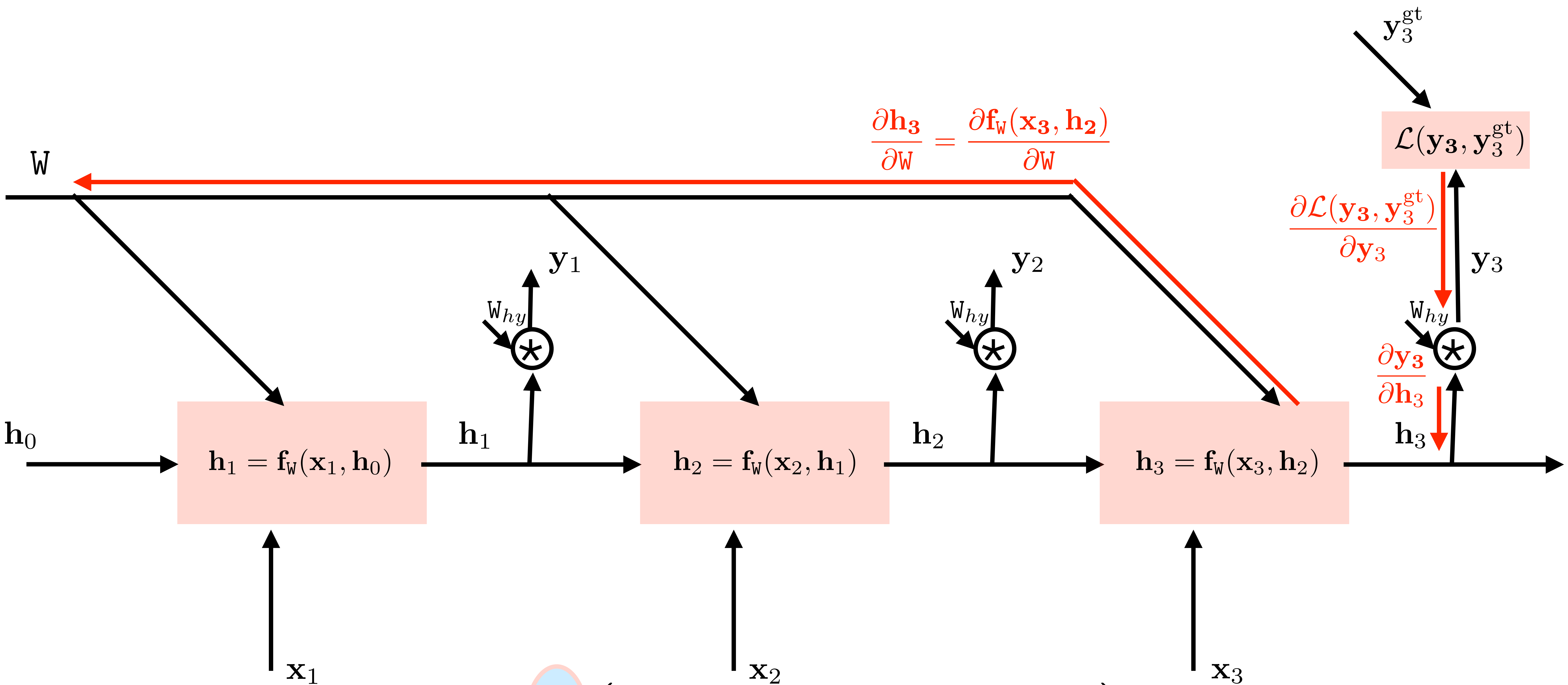


- Backprop:
  - differentiation of multi-dimensional composite function:

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathtt{W}} = \quad ?$$

$$\mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}\big(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

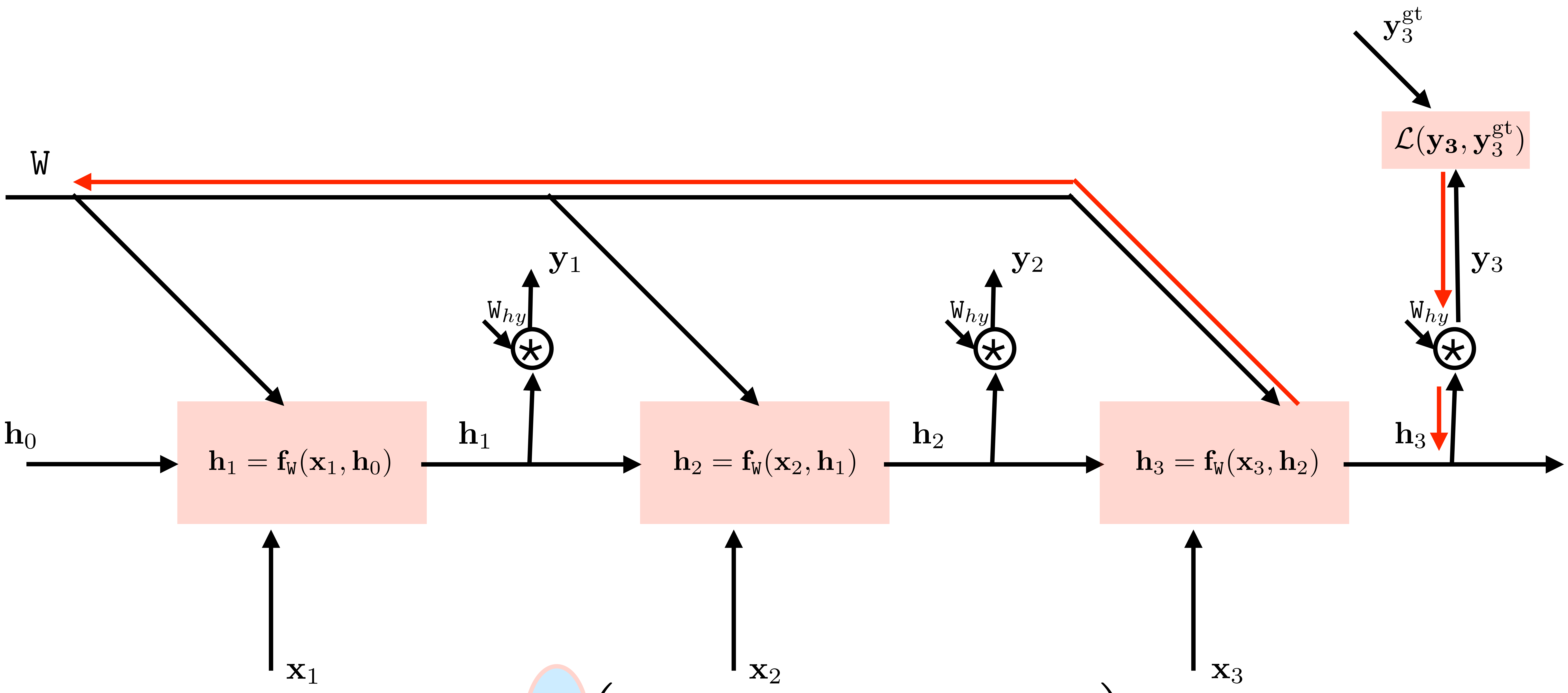
Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}\big(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathtt{W}} = \quad ?$$

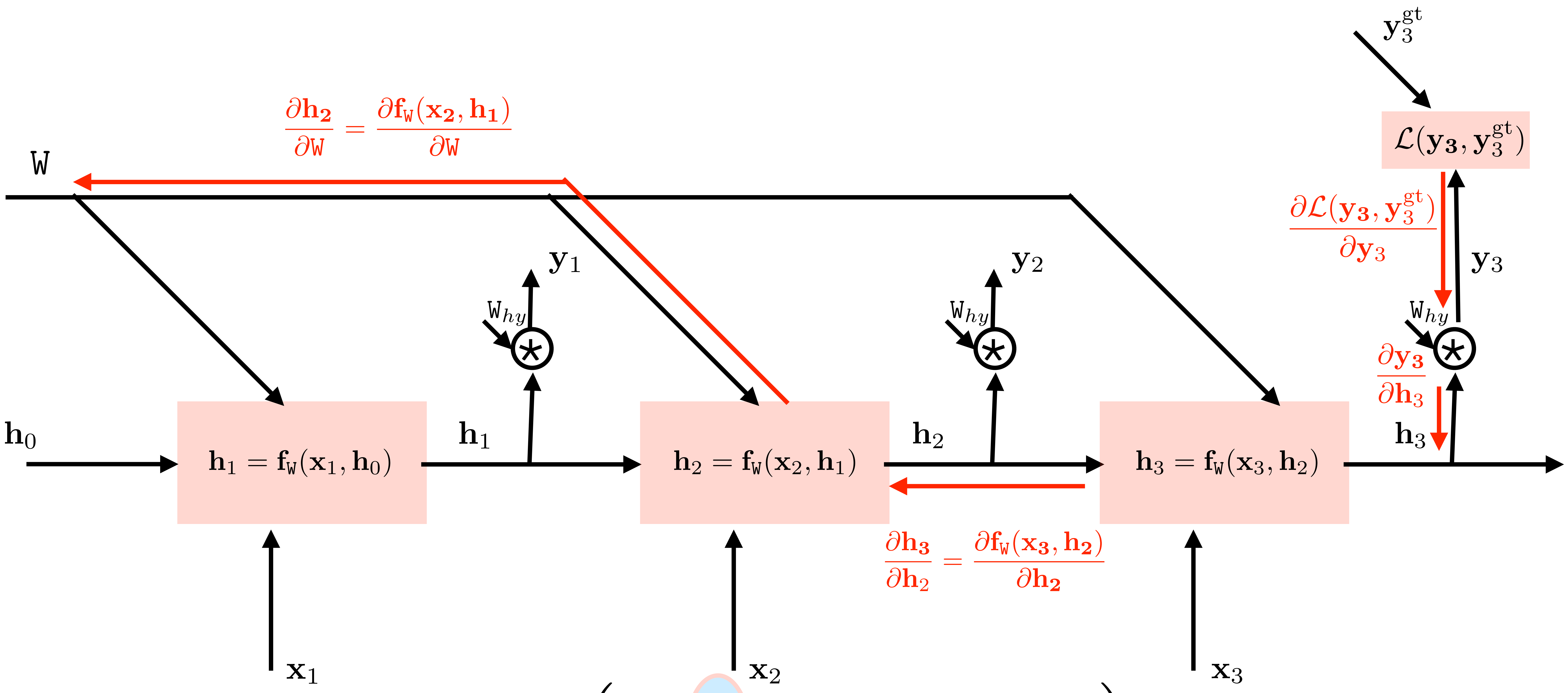
Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}\big(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\text{gt}})}{\partial \mathtt{W}} = \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\text{gt}})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathbf{f}_{\mathtt{W}}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathtt{W}} +$$

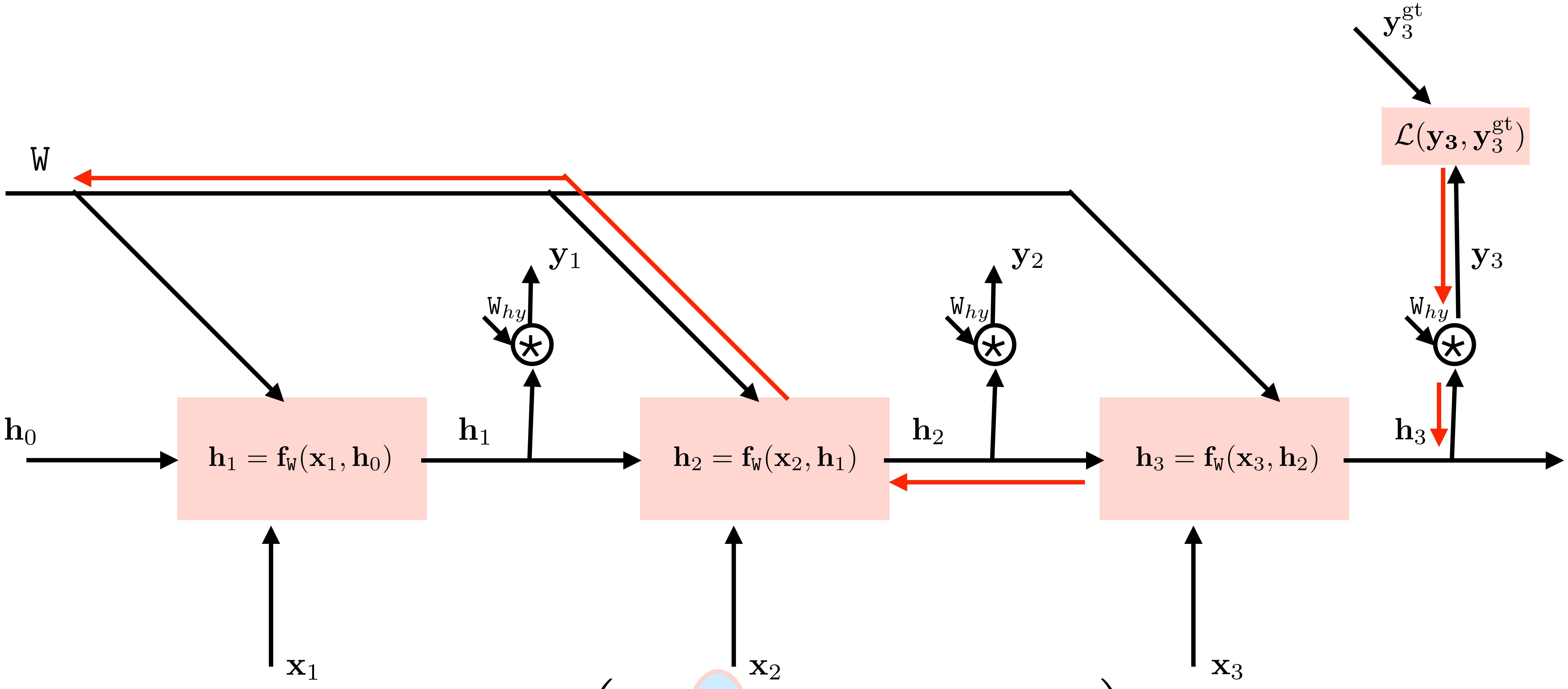
Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0))\Big)$$

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathtt{W}} = \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathbf{f}_{\mathtt{W}}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathtt{W}} +$$

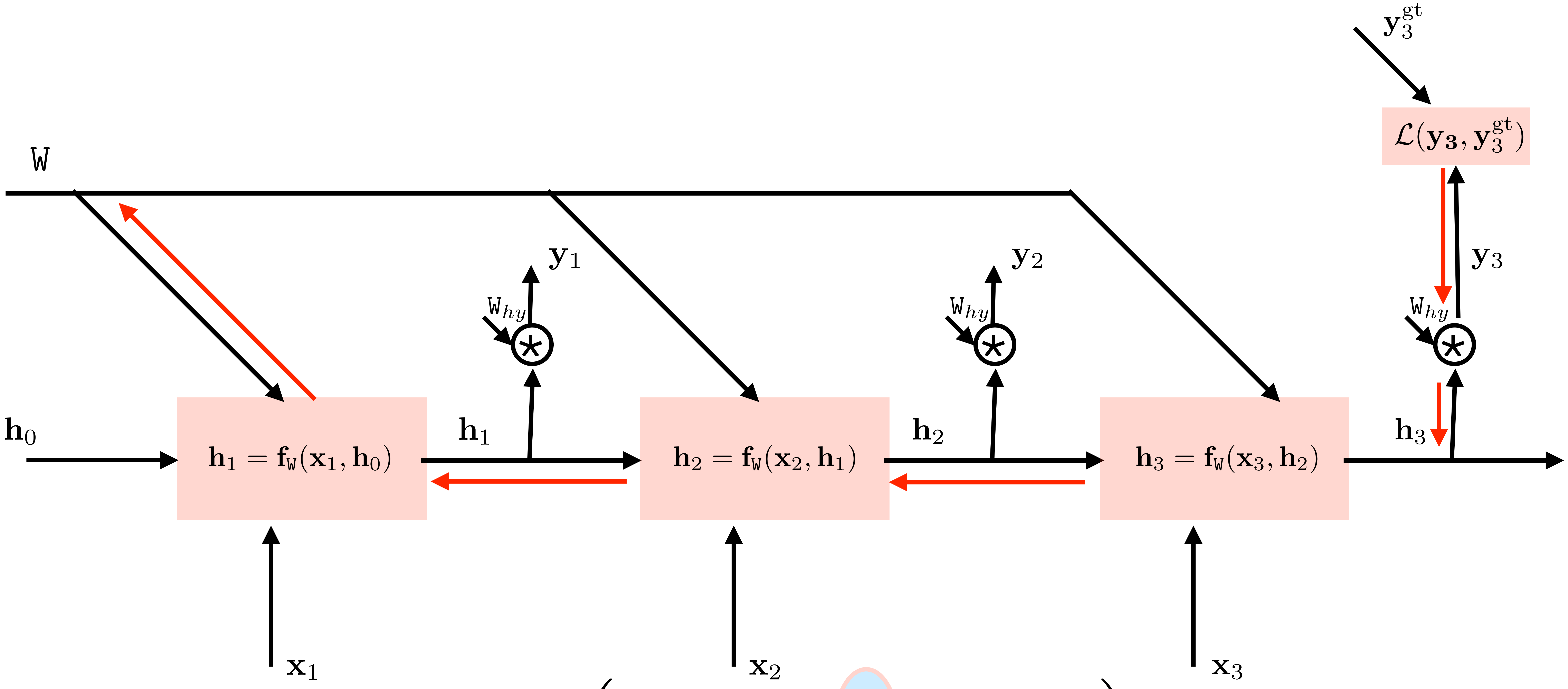
Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0))\Big)$$

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathtt{W}} = \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathbf{f}_{\mathtt{W}}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathtt{W}} + \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{\mathrm{gt}})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathbf{f}_{\mathtt{W}}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathbf{h_2}} \frac{\partial \mathbf{f}_{\mathtt{W}}(\mathbf{x_2}, \mathbf{h_1})}{\partial \mathtt{W}} +$$

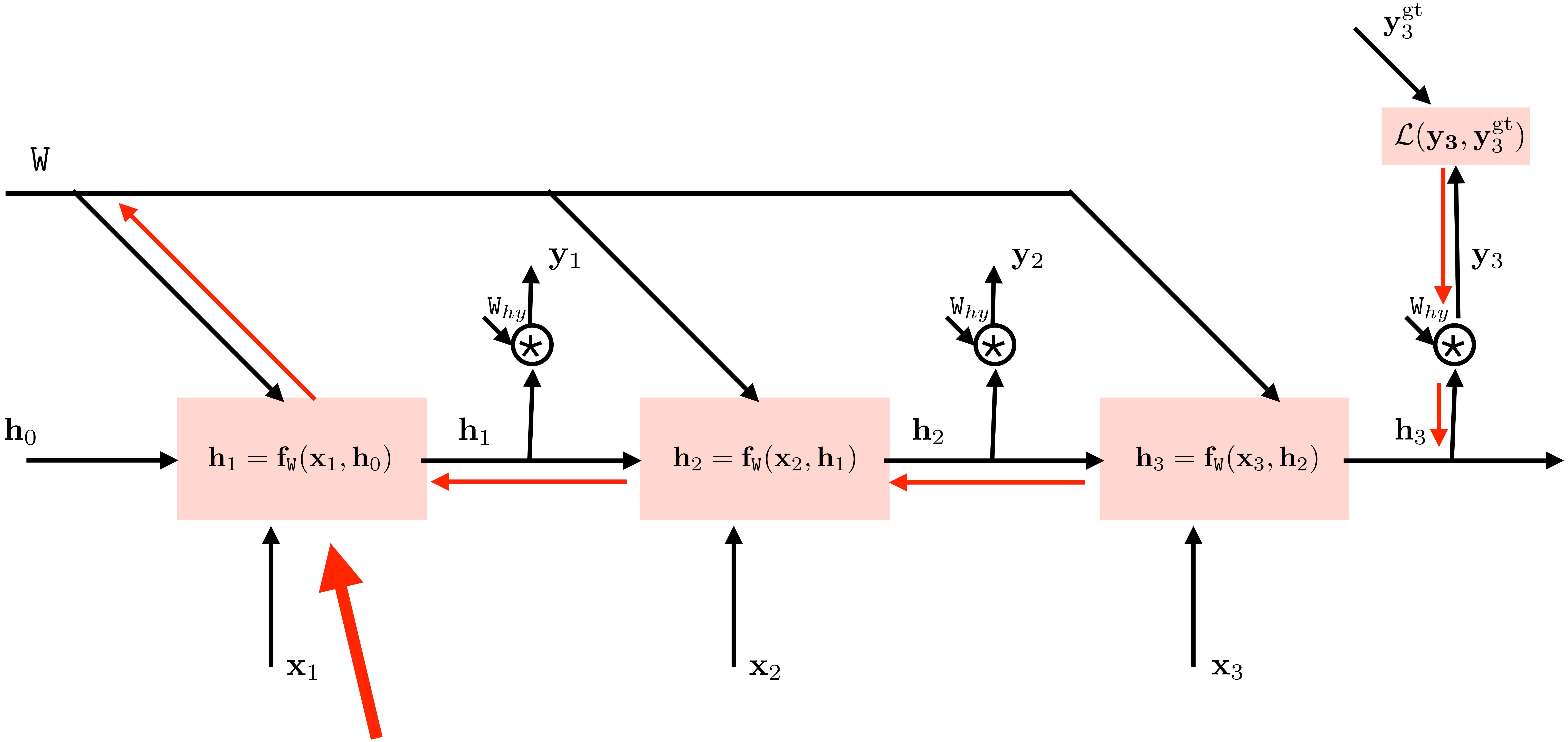
Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathtt{f}_\mathtt{W}\Big(\mathbf{x}_3, \mathtt{f}_\mathtt{W}\big(\mathbf{x}_2, \mathtt{f}_\mathtt{W}(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{gt})}{\partial \mathtt{W}} = \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{gt})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathtt{f}_\mathtt{W}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathtt{W}} + \frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{gt})}{\partial \mathbf{y}_3} \frac{\partial \mathbf{y_3}}{\partial \mathbf{h}_3} \frac{\partial \mathtt{f}_\mathtt{W}(\mathbf{x_3}, \mathbf{h_2})}{\partial \mathbf{h_2}} \frac{\partial \mathtt{f}_\mathtt{W}(\mathbf{x_2}, \mathbf{h_1})}{\partial \mathtt{W}} + \ \ldots$$

Simple recurrent block - backward pass

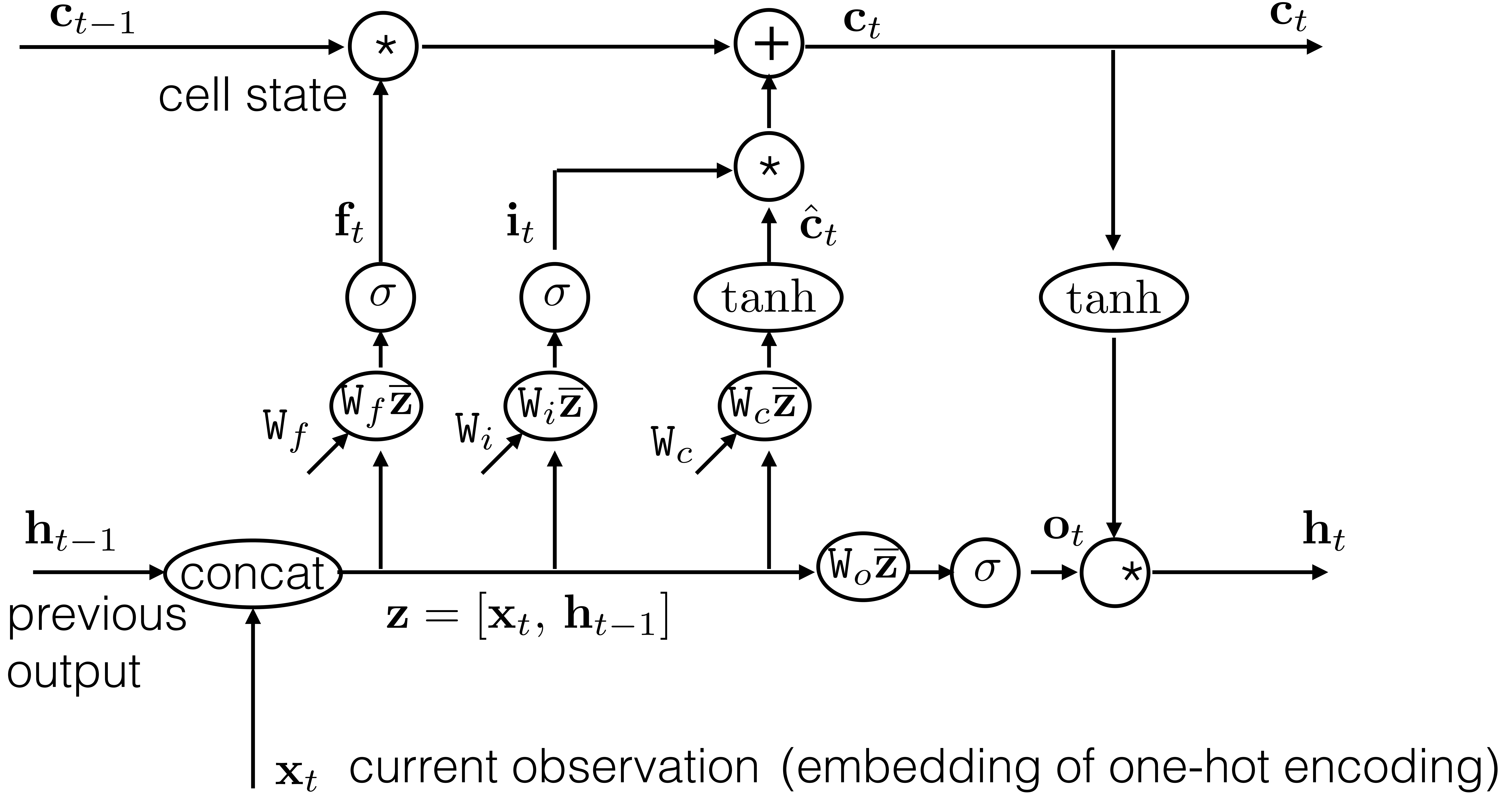


deep blocks often suffer from vanishing gradient

=> better structure needed

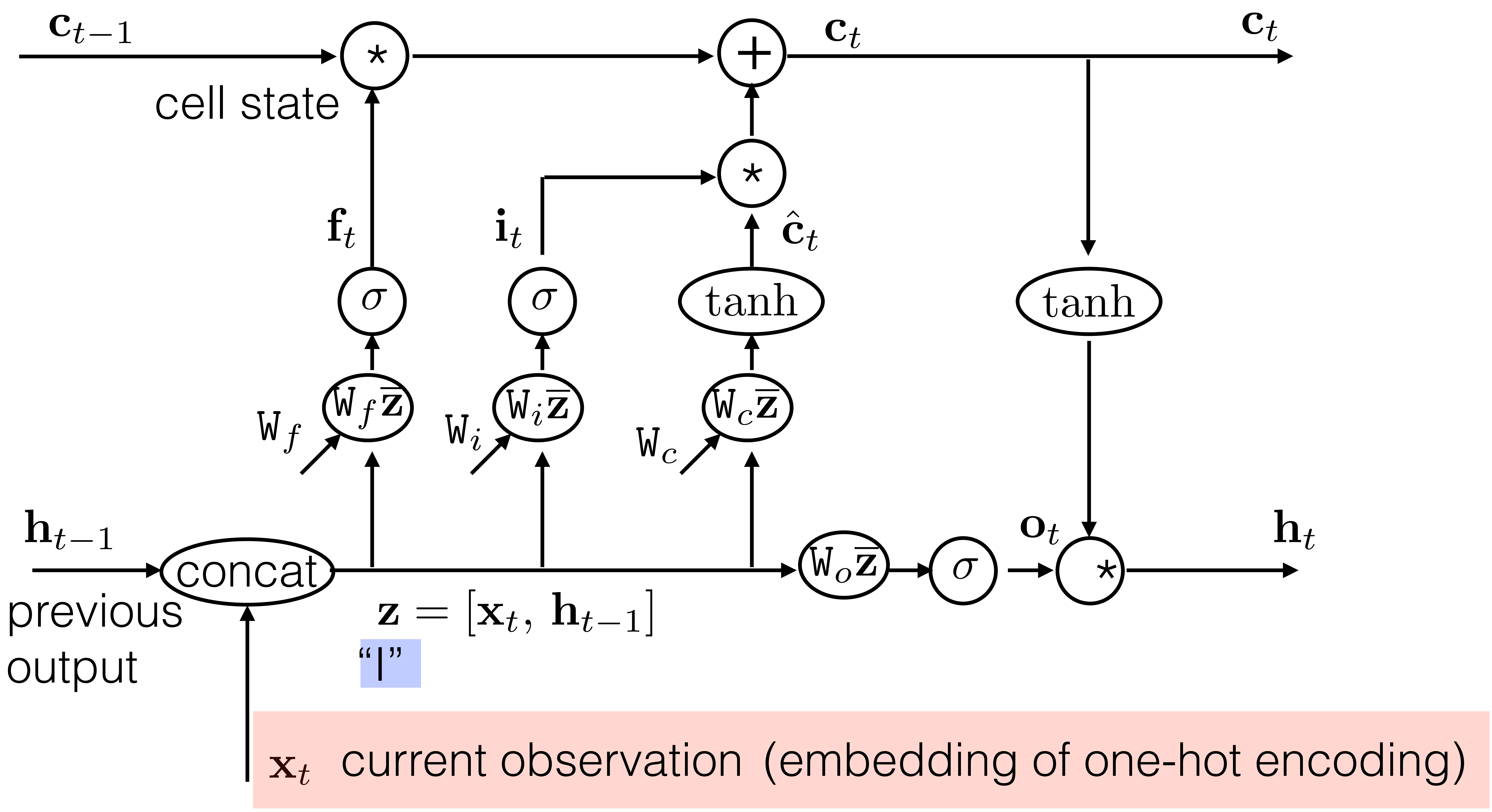
LSTM (kind of ResNet for recurrent networks)

LSTM block



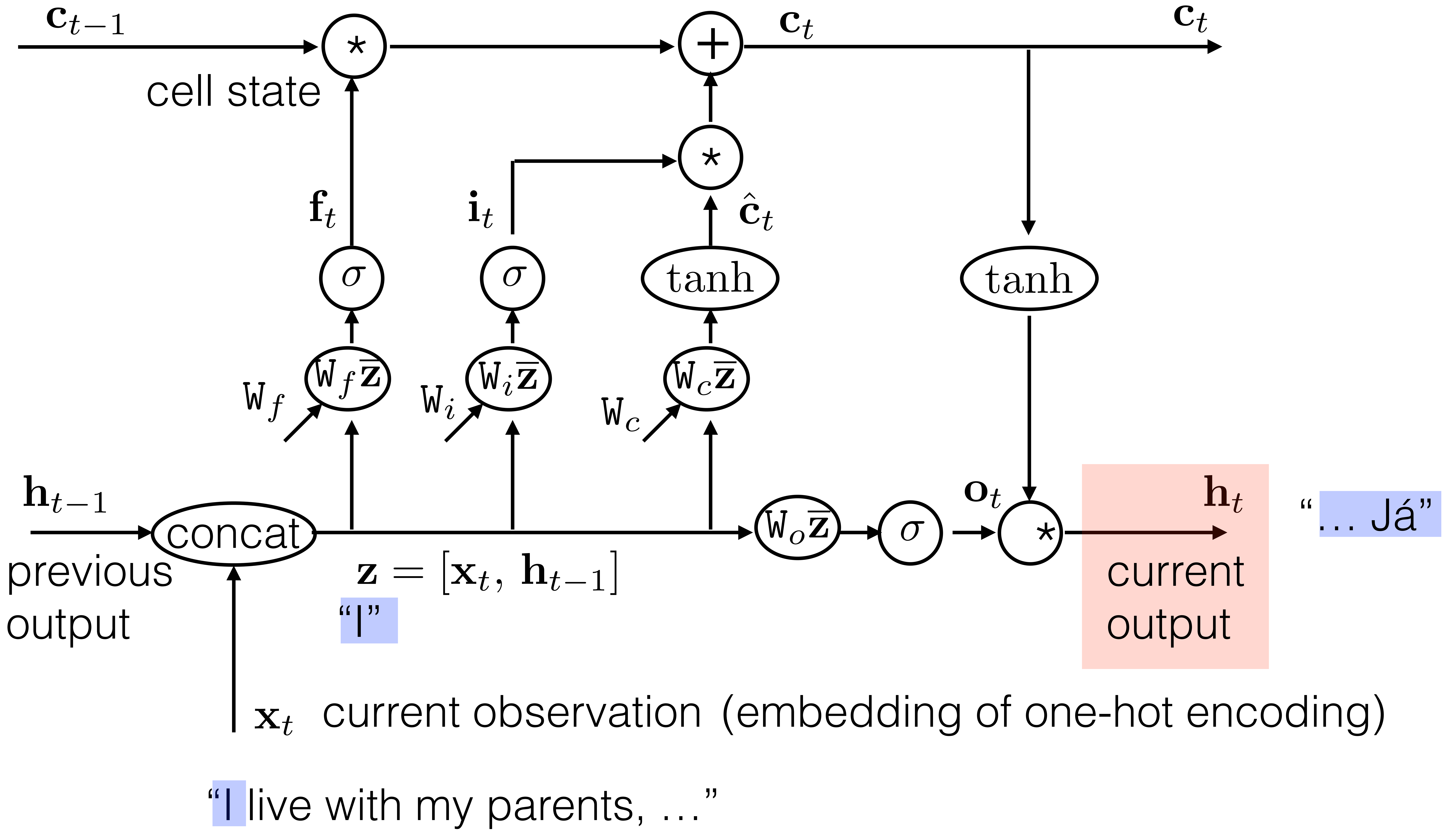
"I live with my parents, …"

LSTM block



"I live with my parents, …"

LSTM block

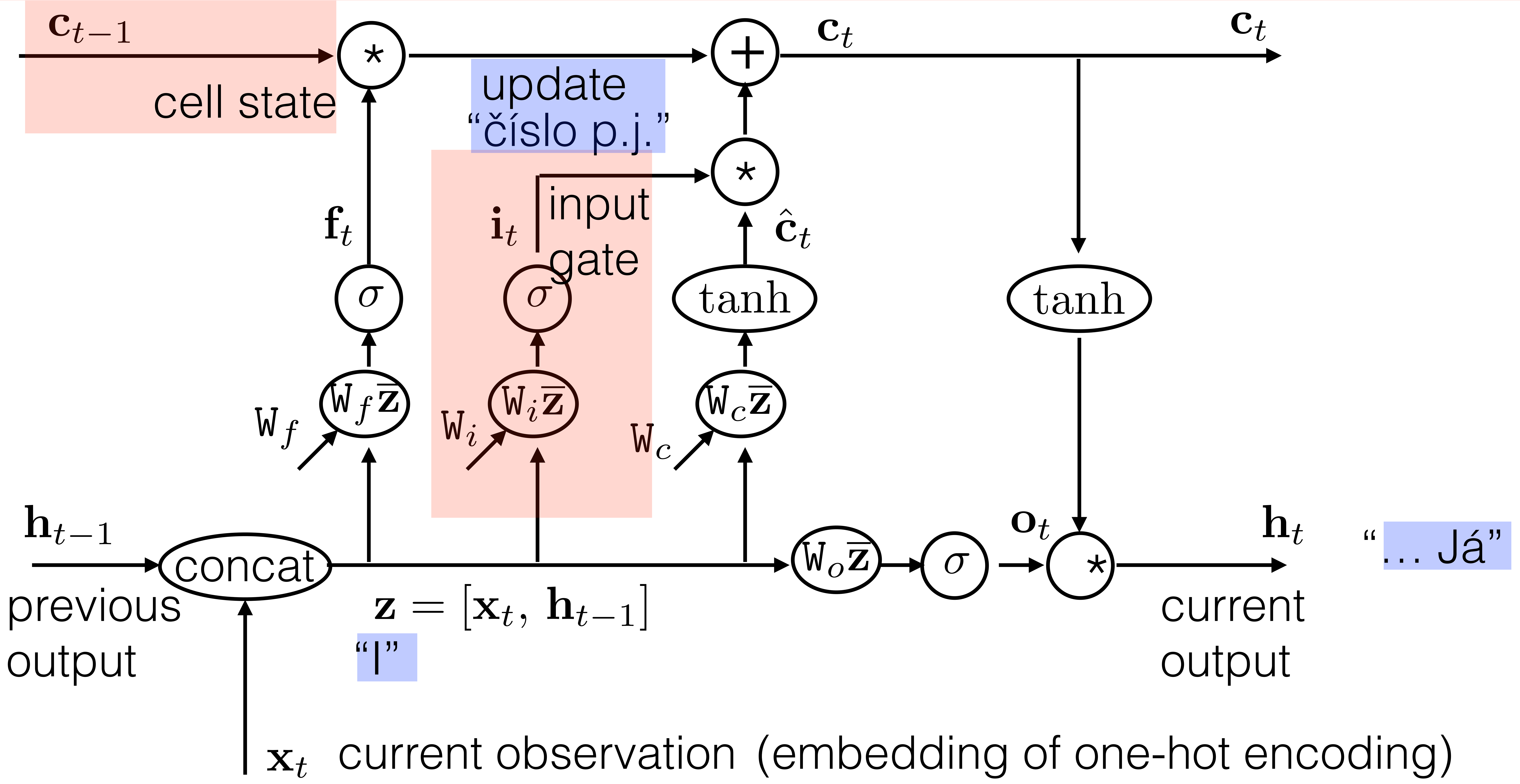


$\mathbf{c}_{t-1}$
cell state
$\mathbf{c}_t$
$\mathbf{c}_t$
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
concat
previous output
$\mathbf{z} = [\mathbf{x}_t, \mathbf{h}_{t-1}]$
"I"
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
$\mathbf{o}_t$
$\mathbf{h}_t$
current output
"… Já"
$\mathbf{x}_t$ current observation (embedding of one-hot encoding)
"I live with my parents, …"

LSTM block
cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]
$\mathbf{c}_{t-1}$
cell state
$\mathbf{c}_t$
$\mathbf{c}_t$
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
concat
previous output
$\mathbf{z} = [\mathbf{x}_t, \mathbf{h}_{t-1}]$
"I"
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
$\mathbf{o}_t$
$\mathbf{h}_t$
"… Já"
current output
$\mathbf{x}_t$ current observation (embedding of one-hot encoding)
"I live with my parents, …"

LSTM block

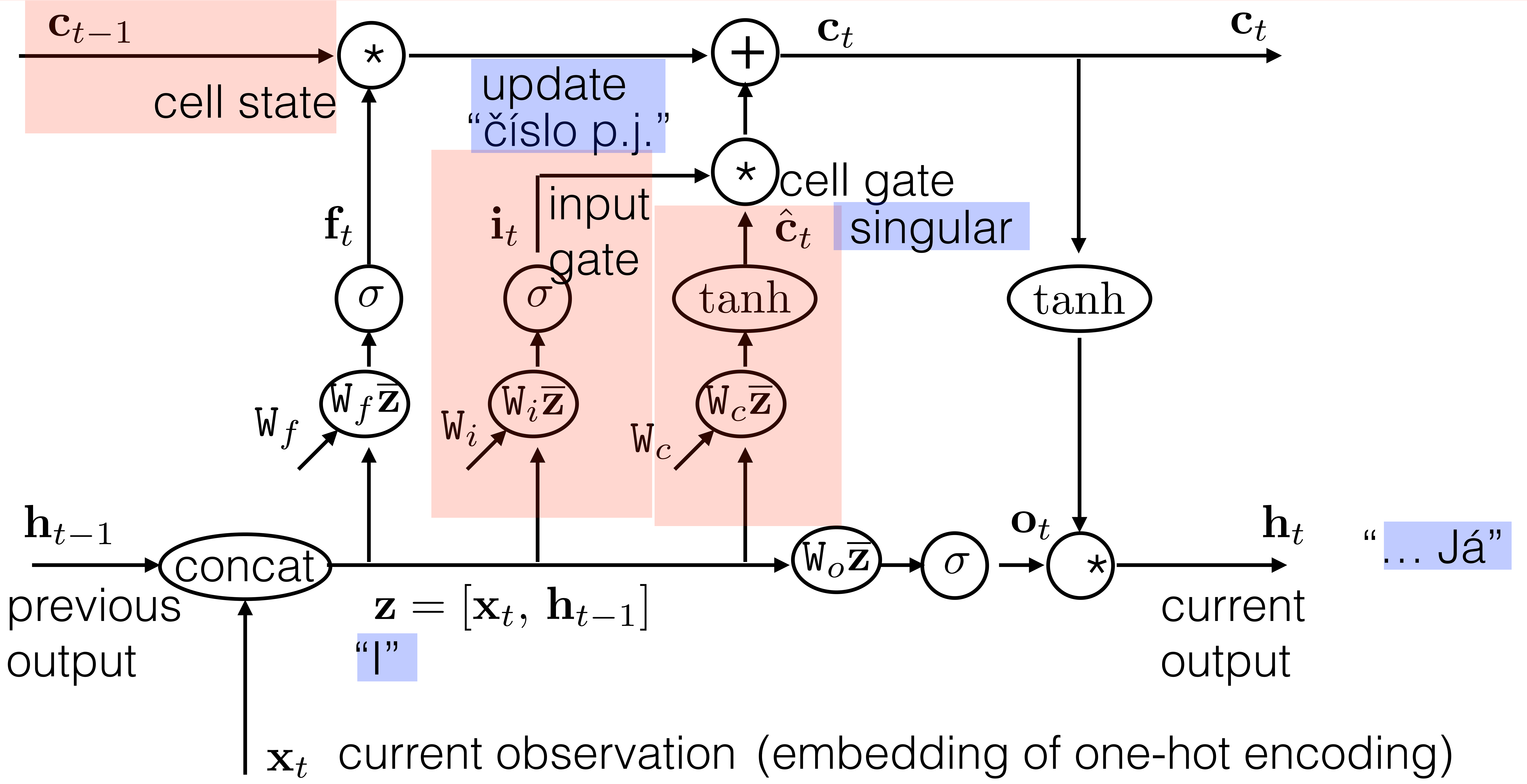
cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]



“I live with my parents, …”

LSTM block

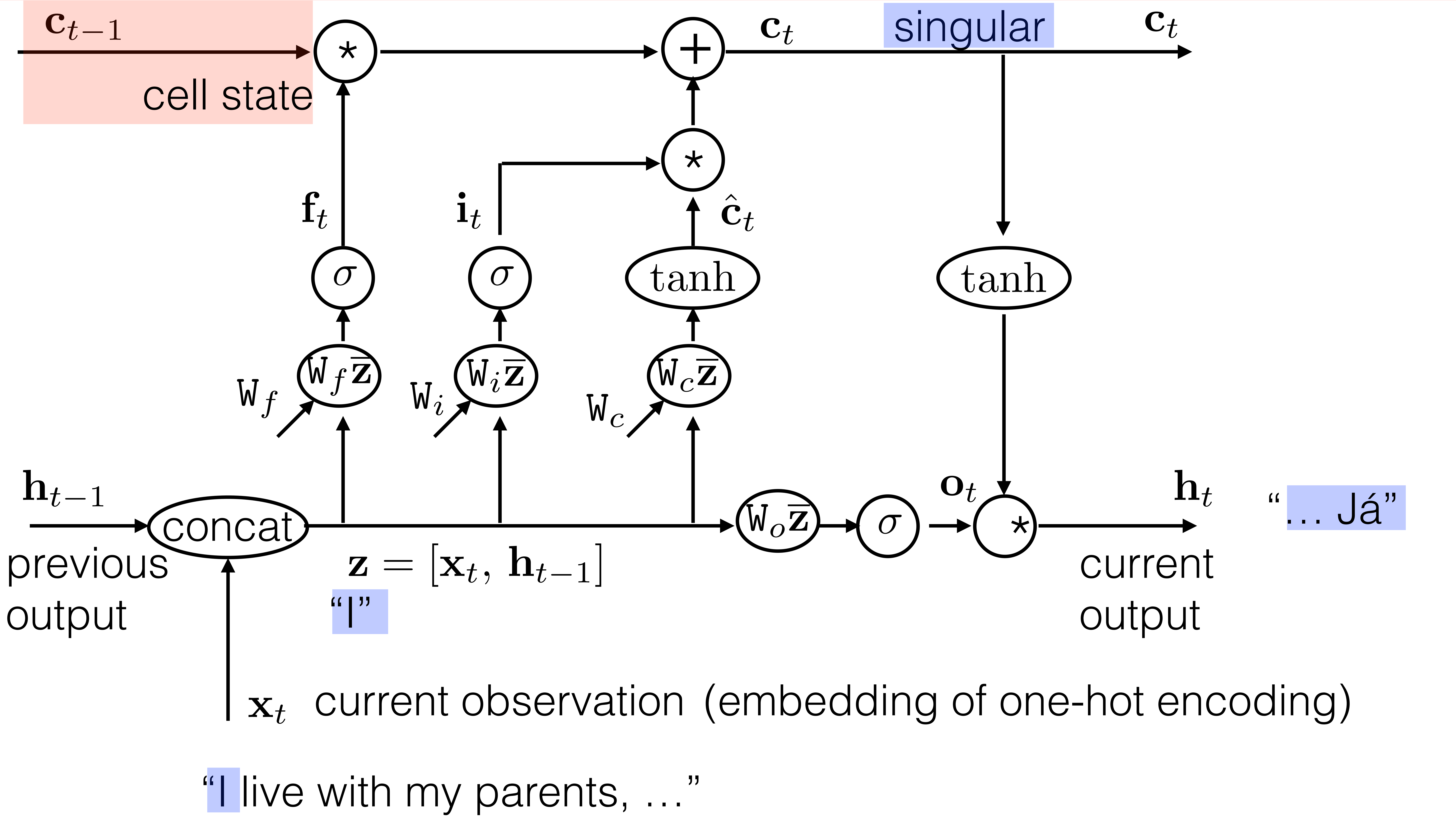
cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]



“I live with my parents, …”

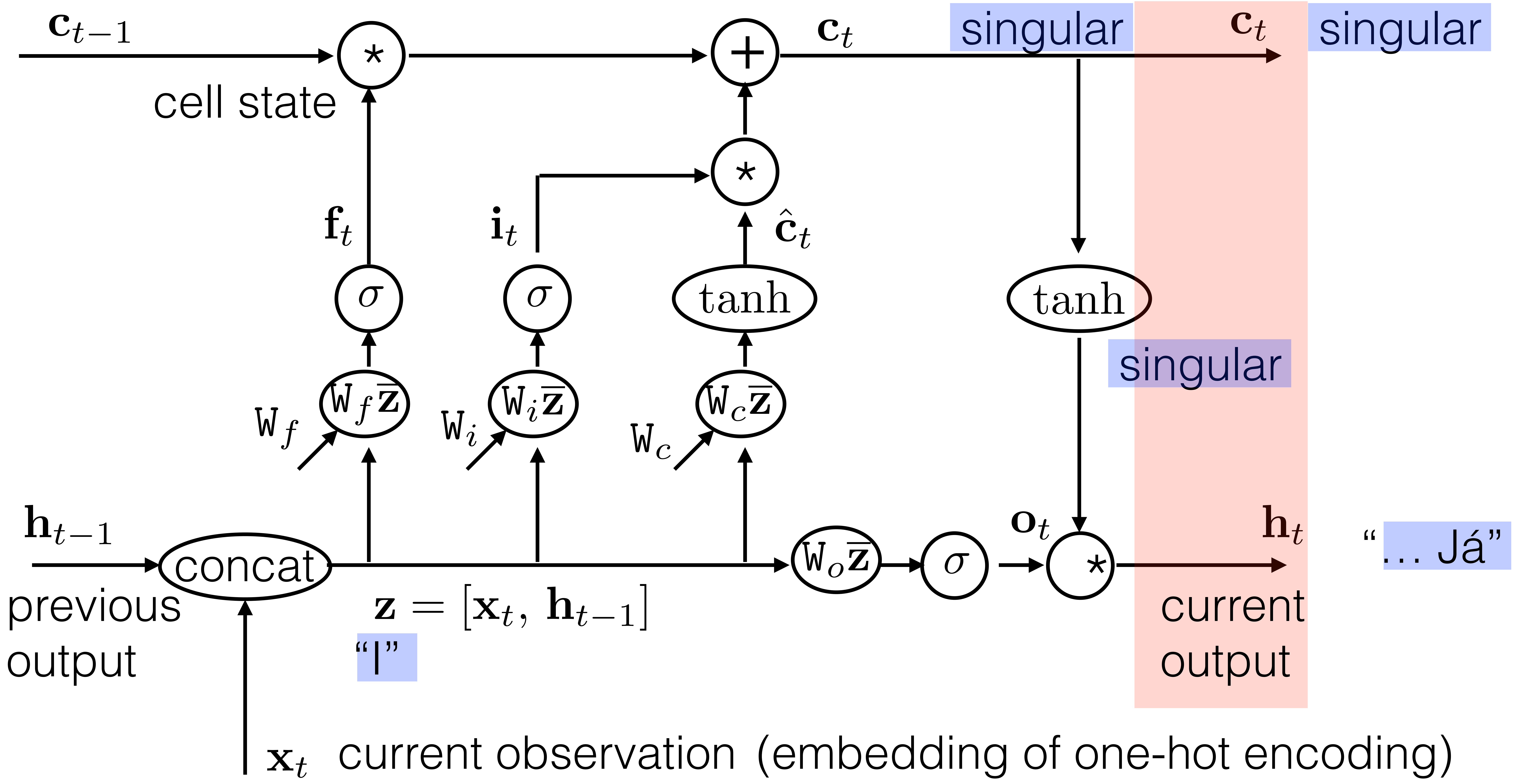
LSTM block

cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]



LSTM block

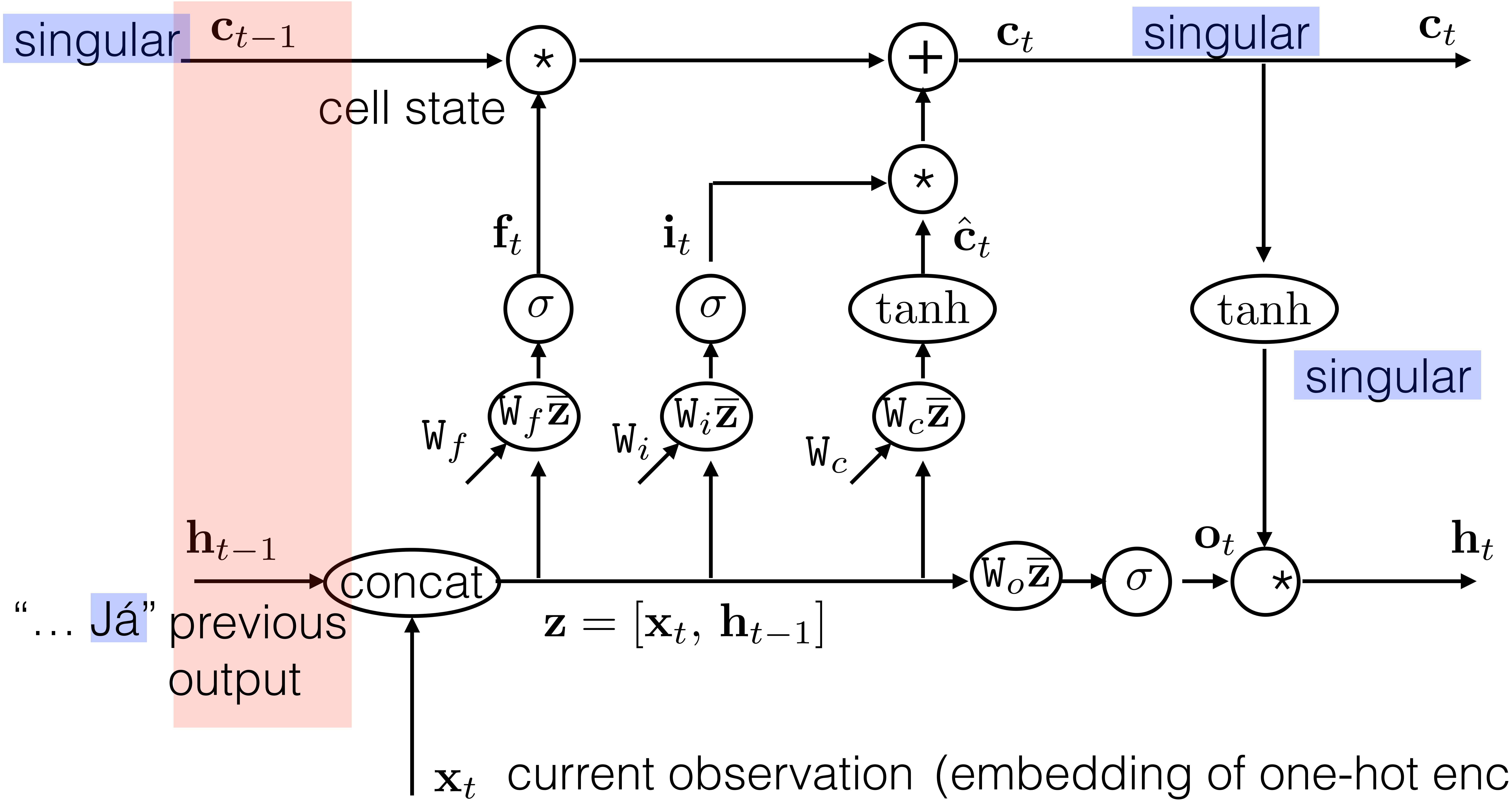
cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]



"I live with my parents, …"

LSTM block

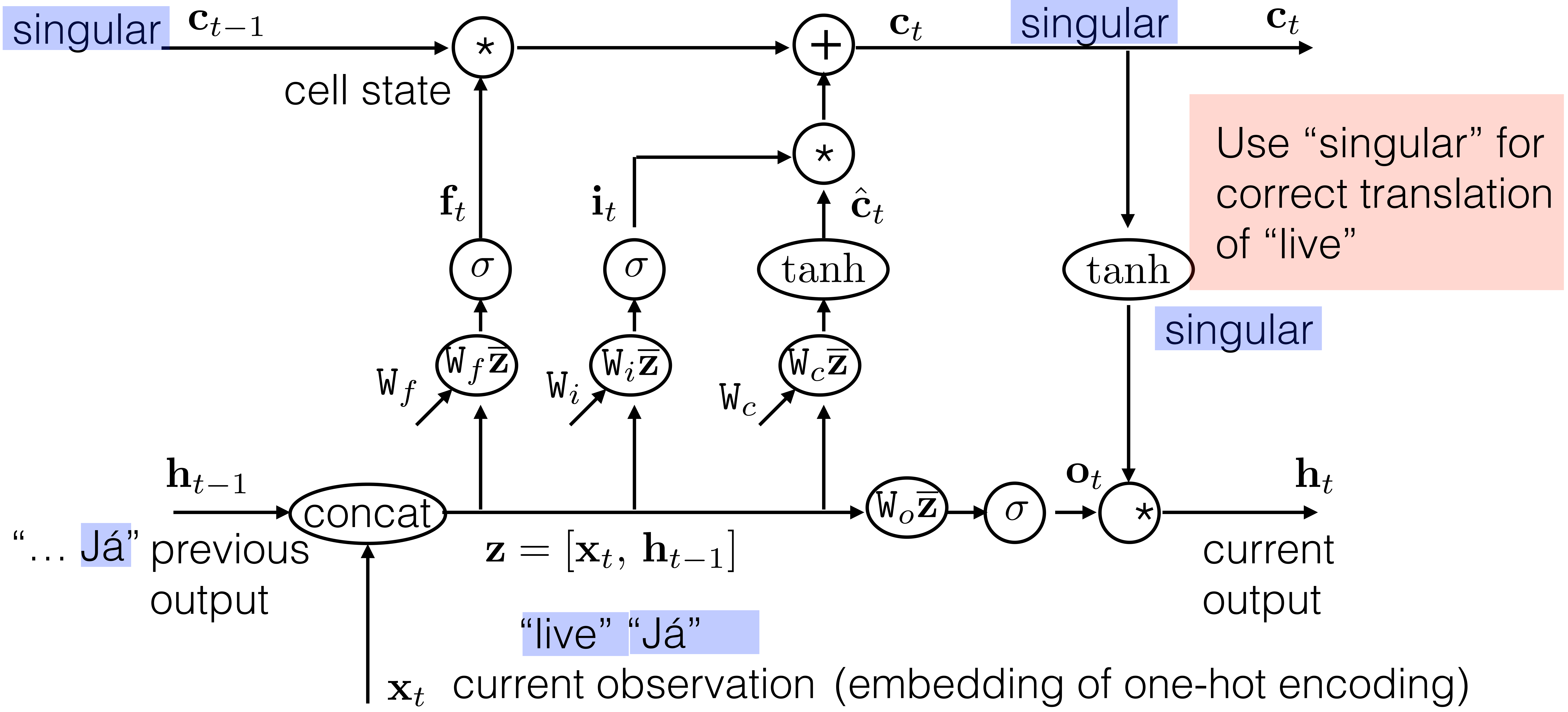
cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]



"I live with my parents, …"

LSTM block

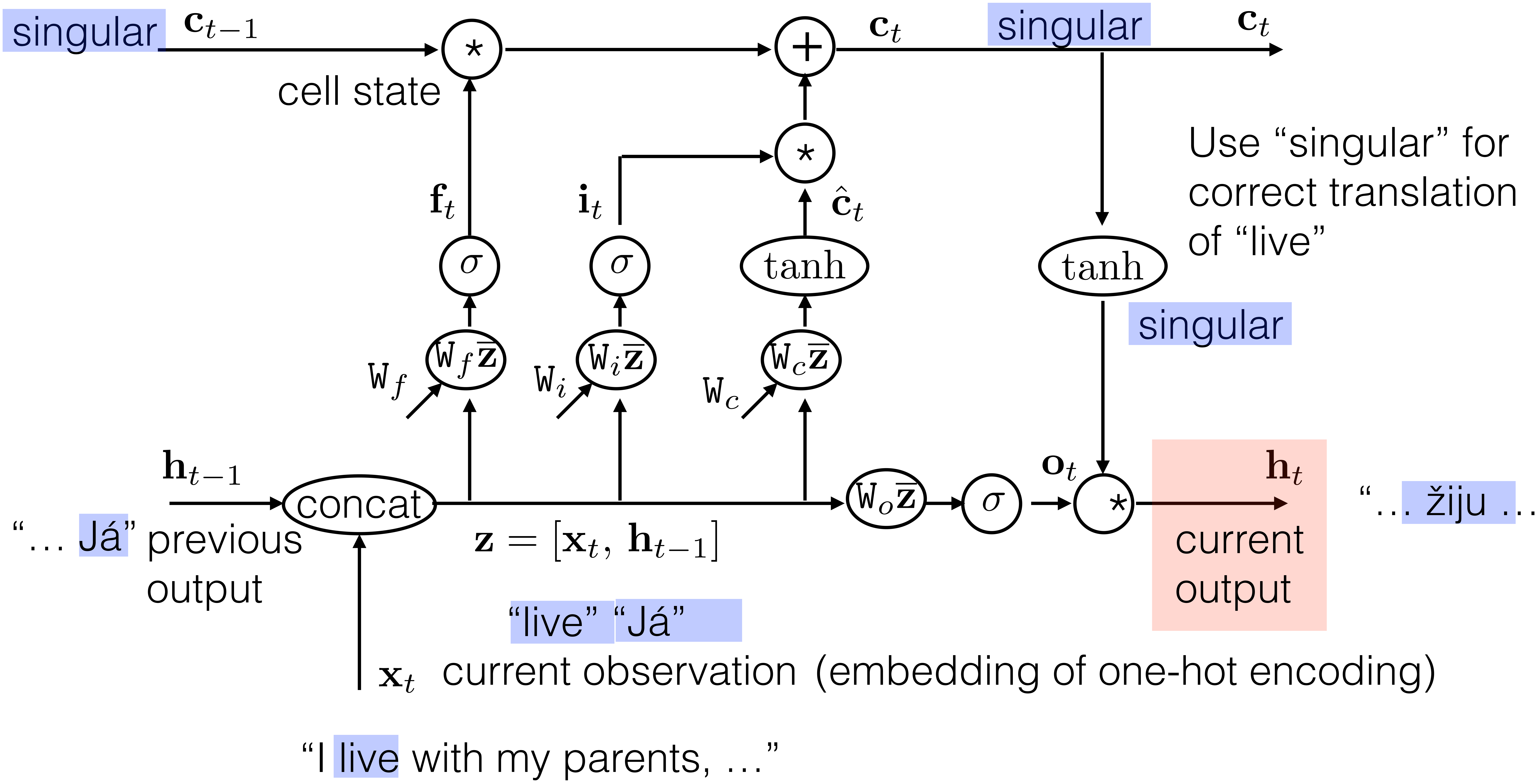
cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]



"I live with my parents, …"

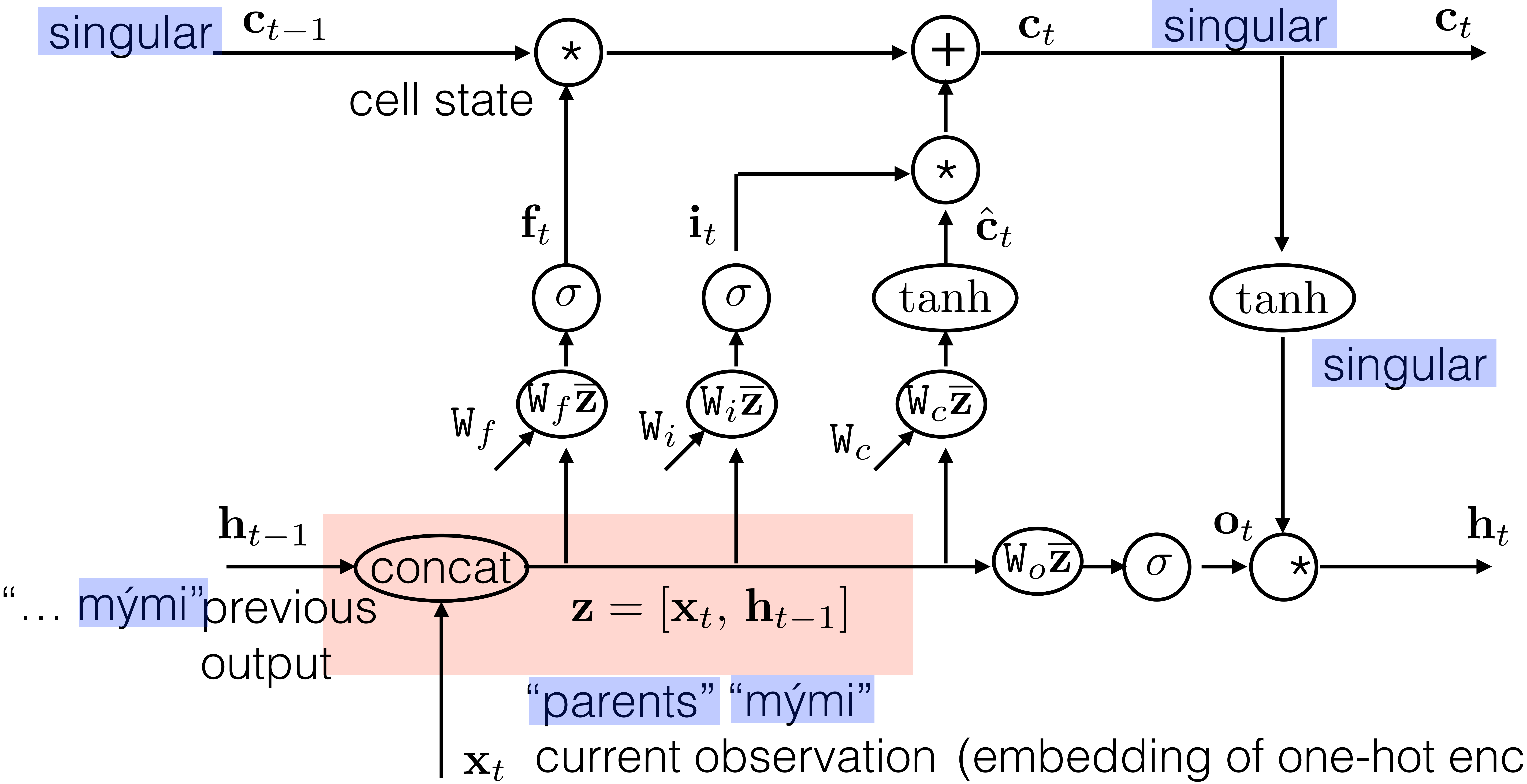
LSTM block

cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]



LSTM block

cell state is long term memory= vector [**singular**/plural, feminine/masculine, case, …]

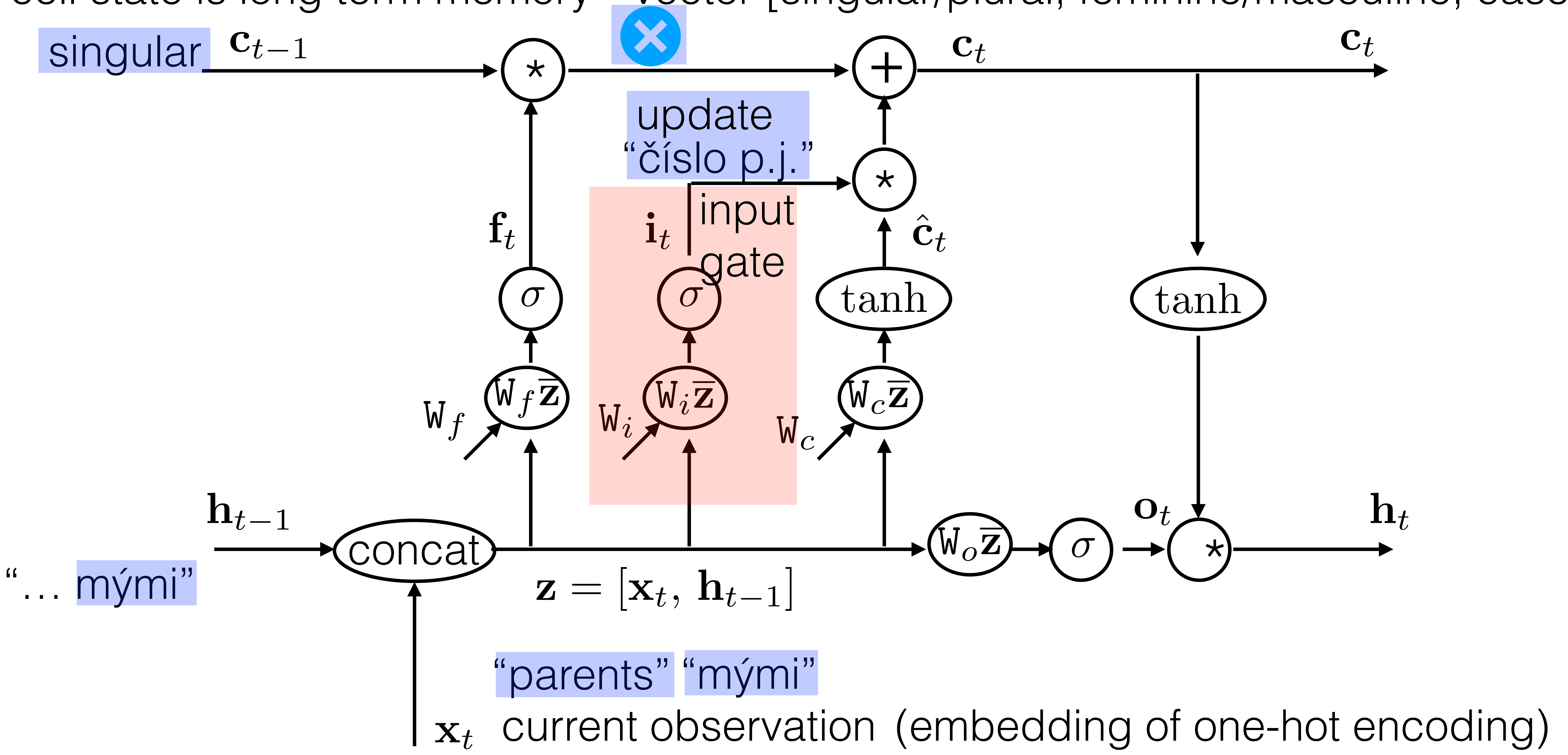


"I live with my parents, …"


LSTM block
cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]
singular
$\mathbf{c}_{t-1}$
$\mathbf{c}_t$
$\mathbf{c}_t$
forget gate
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_i$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
concat
$\mathbf{o}_t$
$\mathbf{h}_t$
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
“… mými”
$\mathbf{z} = [\mathbf{x}_t, \mathbf{h}_{t-1}]$
“parents” “mými”
$\mathbf{x}_t$ current observation (embedding of one-hot encoding)
“I live with my parents, …”

LSTM block

cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]



"I live with my parents, …"

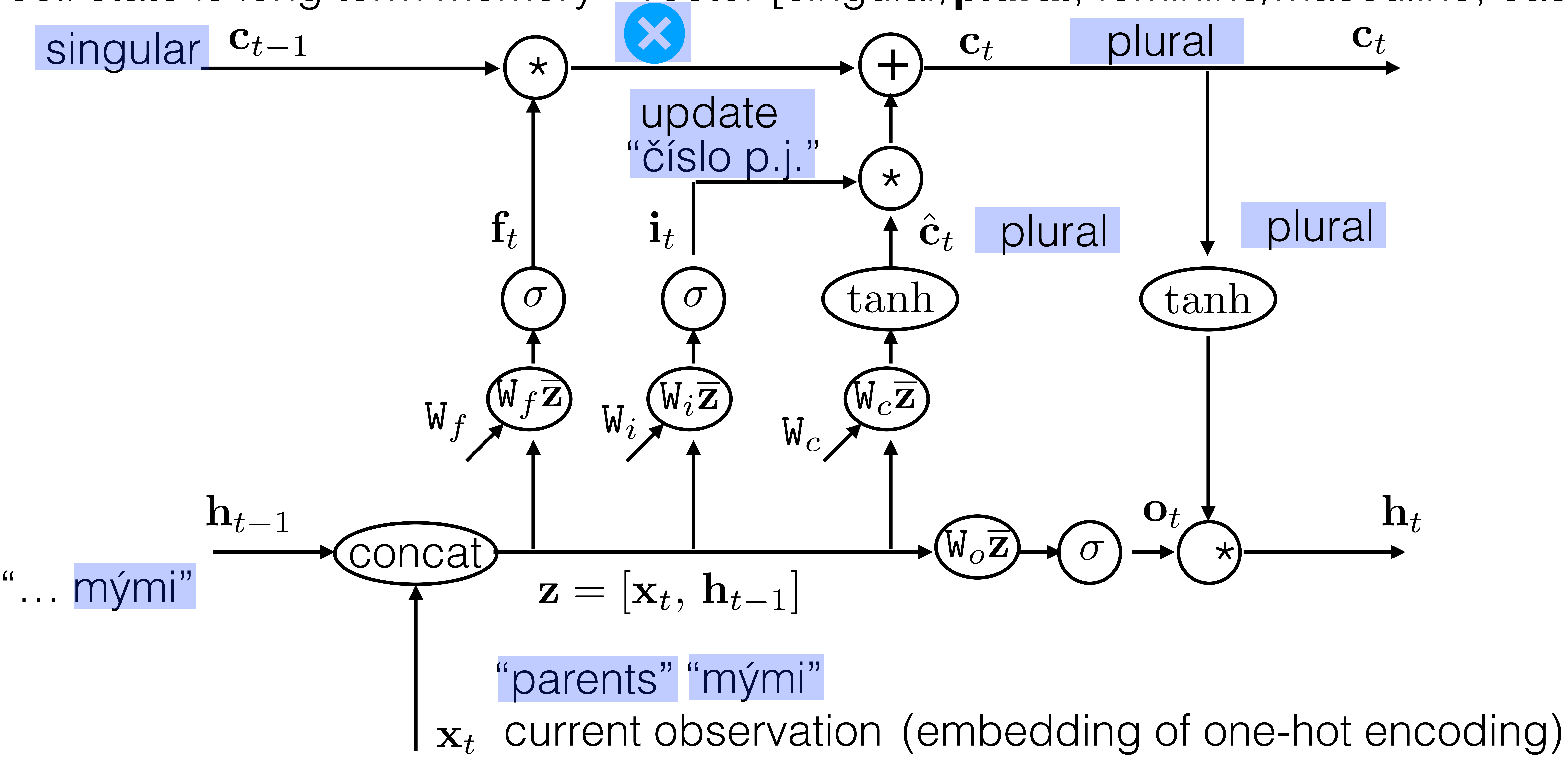
LSTM block

cell state is long term memory= vector [singular/plural, feminine/masculine, case, …]

“I live with my parents, …”

LSTM block

cell state is long term memory= vector [singular/**plural**, feminine/masculine, case, …]



“I live with my parents, …”

LSTM block

cell state is long term memory= vector [singular/**plural**, feminine/masculine, case, …]

LSTM block

```
torch.nn.LSTM(input_size, hidden_dim, n_layers)
```

Summary

- recurrence is **resolved by unrolling** the computational graph in time.
- memory is attention through time [Alex Graves 2020]