Attention and Memory

Karel Zimmermann

http://cmp.felk.cvut.cz/~zimmerk/

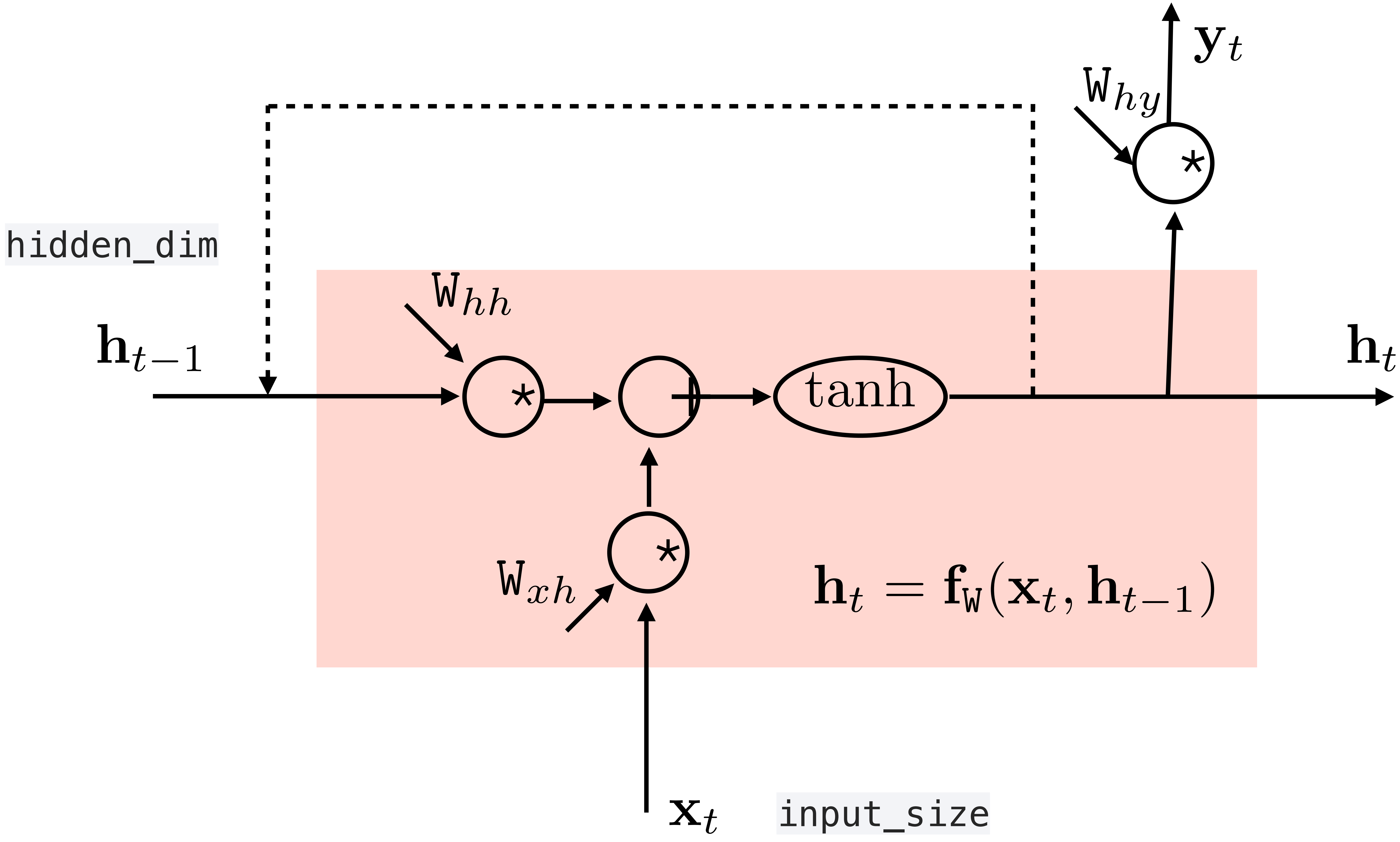
Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

Simple recurrent block

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



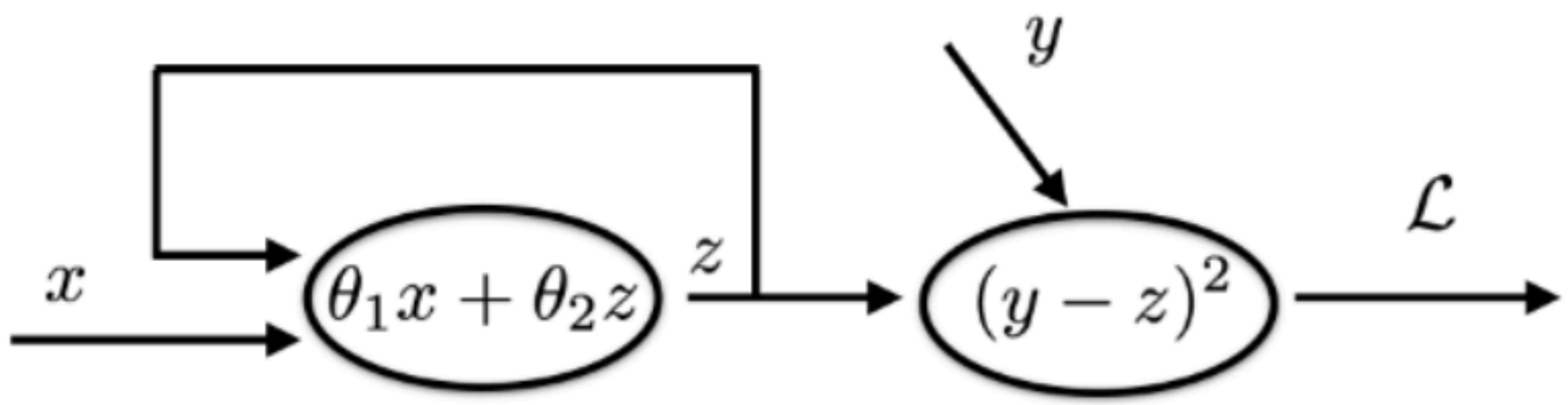
PyTorch: https://pytorch.org/docs/stable/nn.h

RNN example with backprop

Consider linear recurrent neural network with L2 loss depicted on the image below. The network is initialized with parameters $\theta_1 = 1, \theta_2 = 0, z_0 = 0$. You are given the following training sequence:

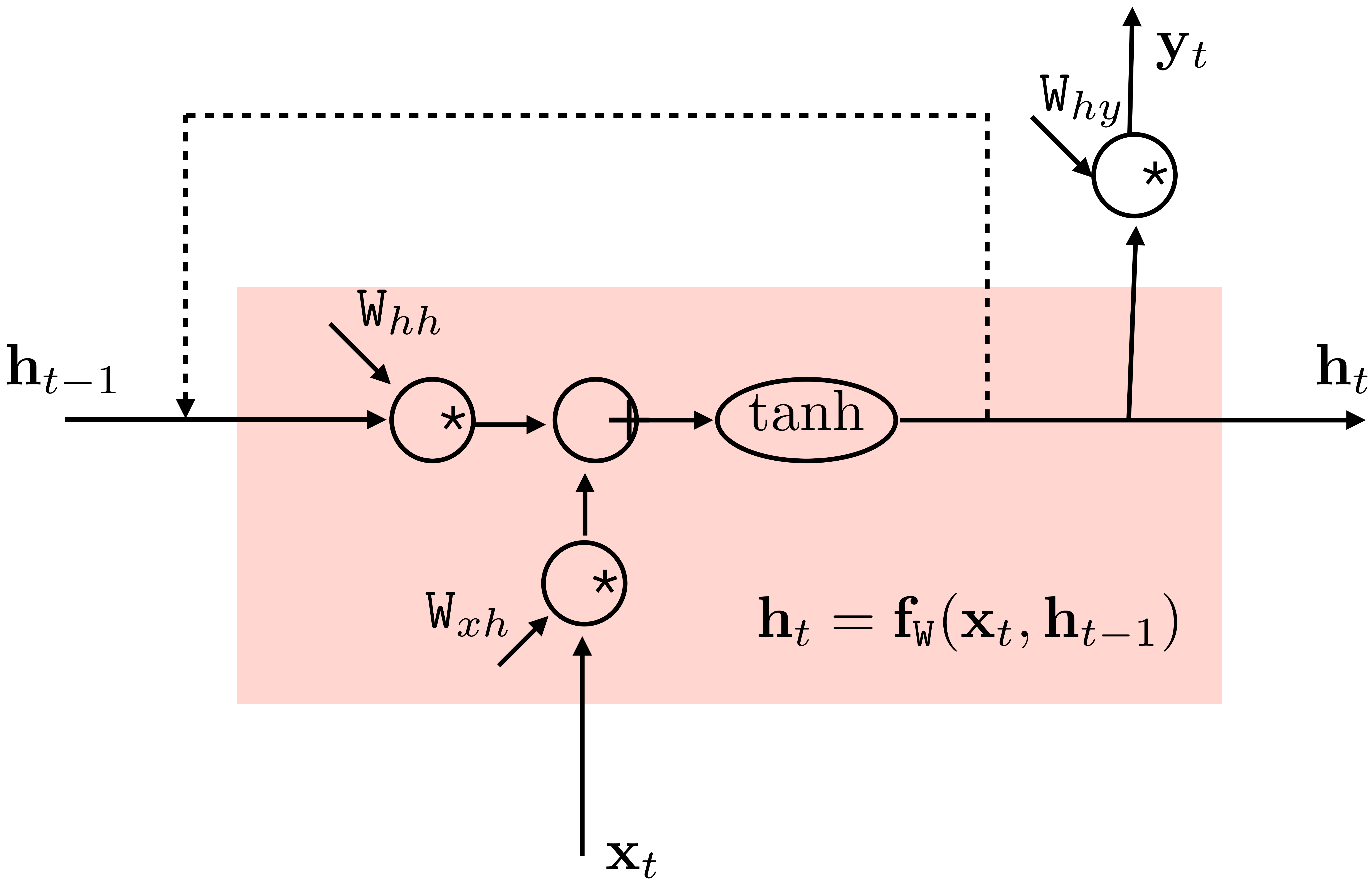
time=1	time=2
$x_1 = 0$	$x_2 = 1$
$y_1 = 1$	$y_2 = 3$

Estimate gradient of the overall loss (computed over all available outputs y_i for both available times $i = 1, 2$) with respect to θ_1.

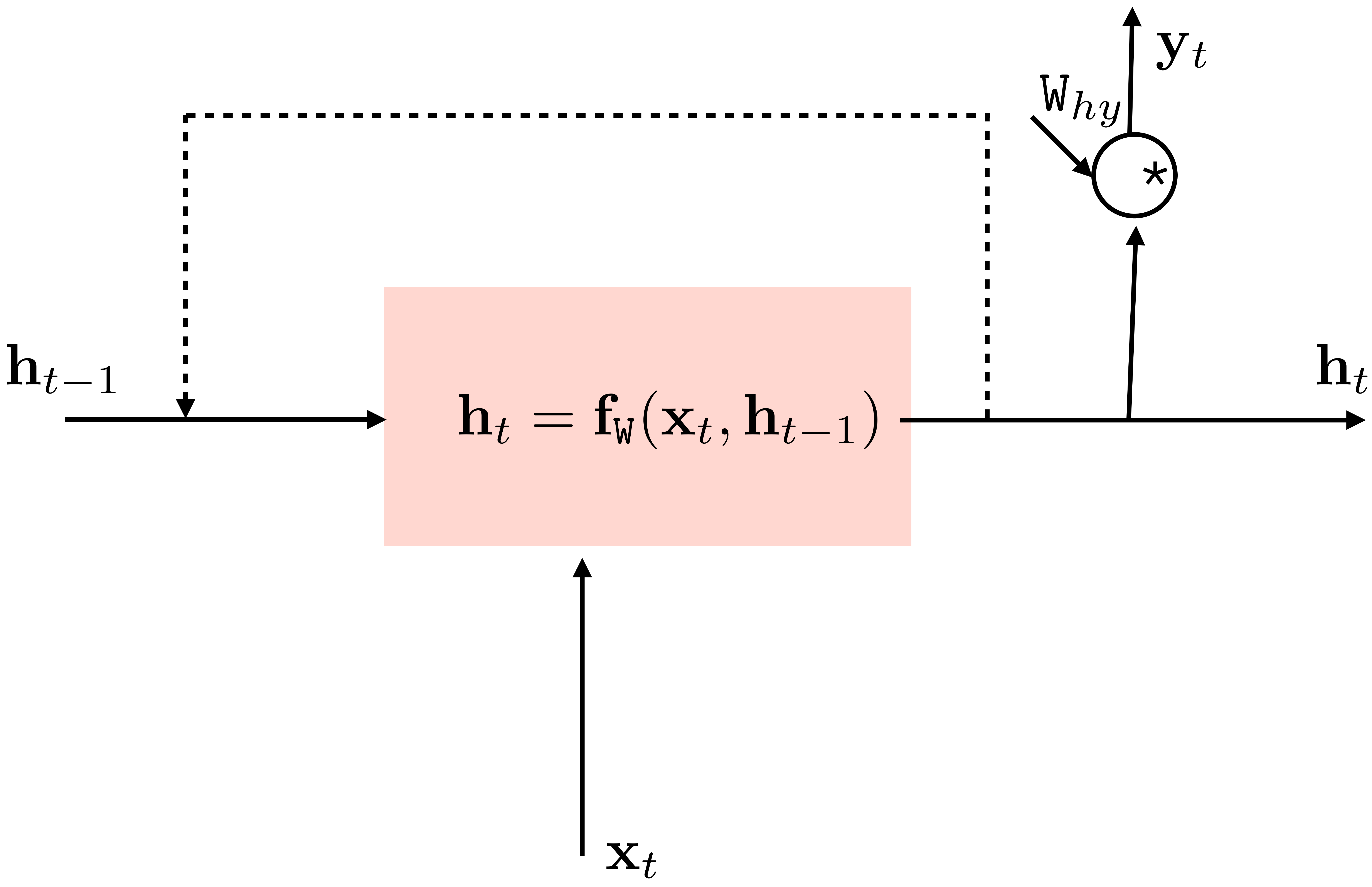


Hint: Unroll the network in time, to obtain a usual feedforward network with two loss nodes. Do the backpropagation as usual.

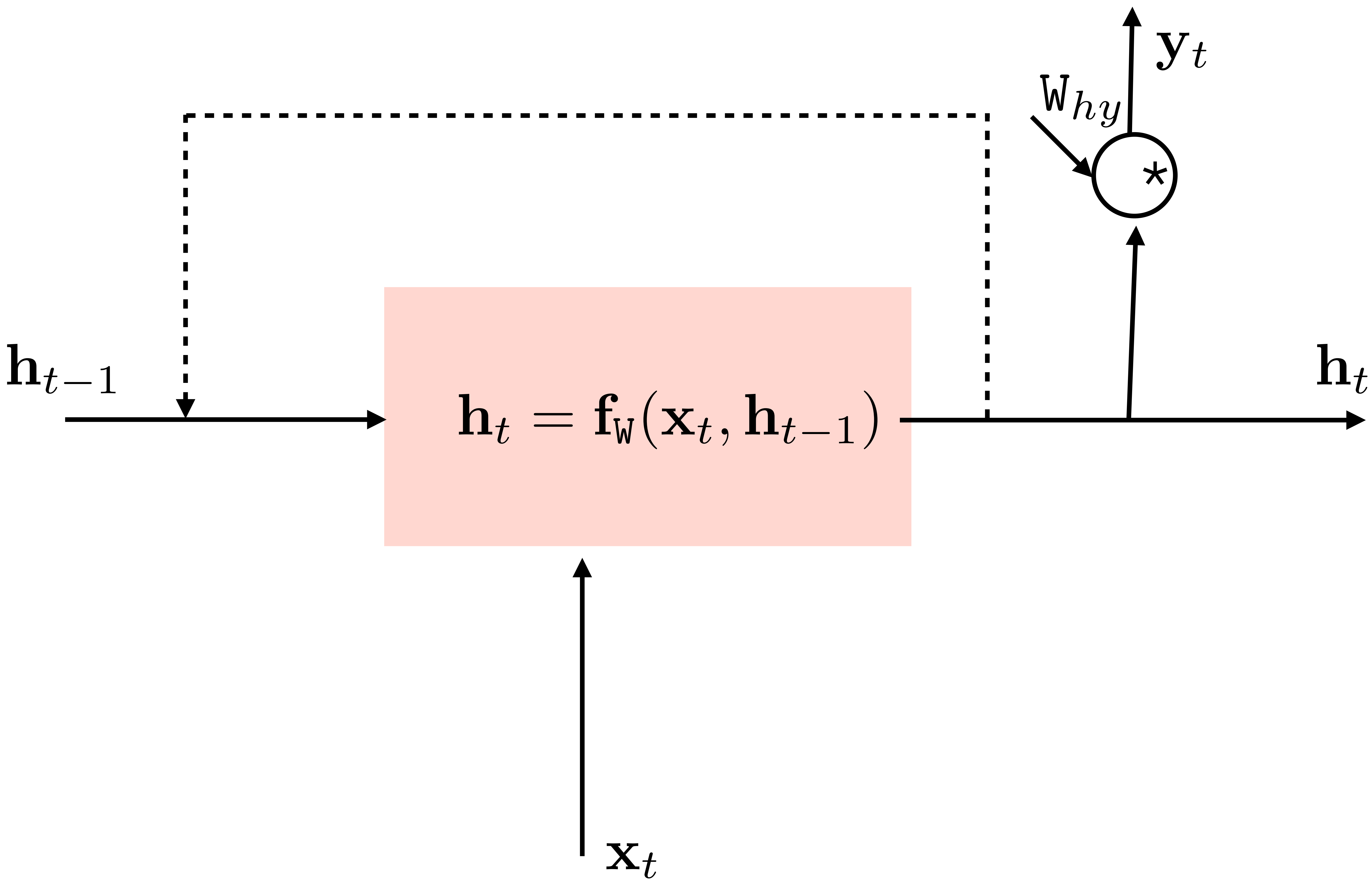
Simple recurrent block - feed-forward pass

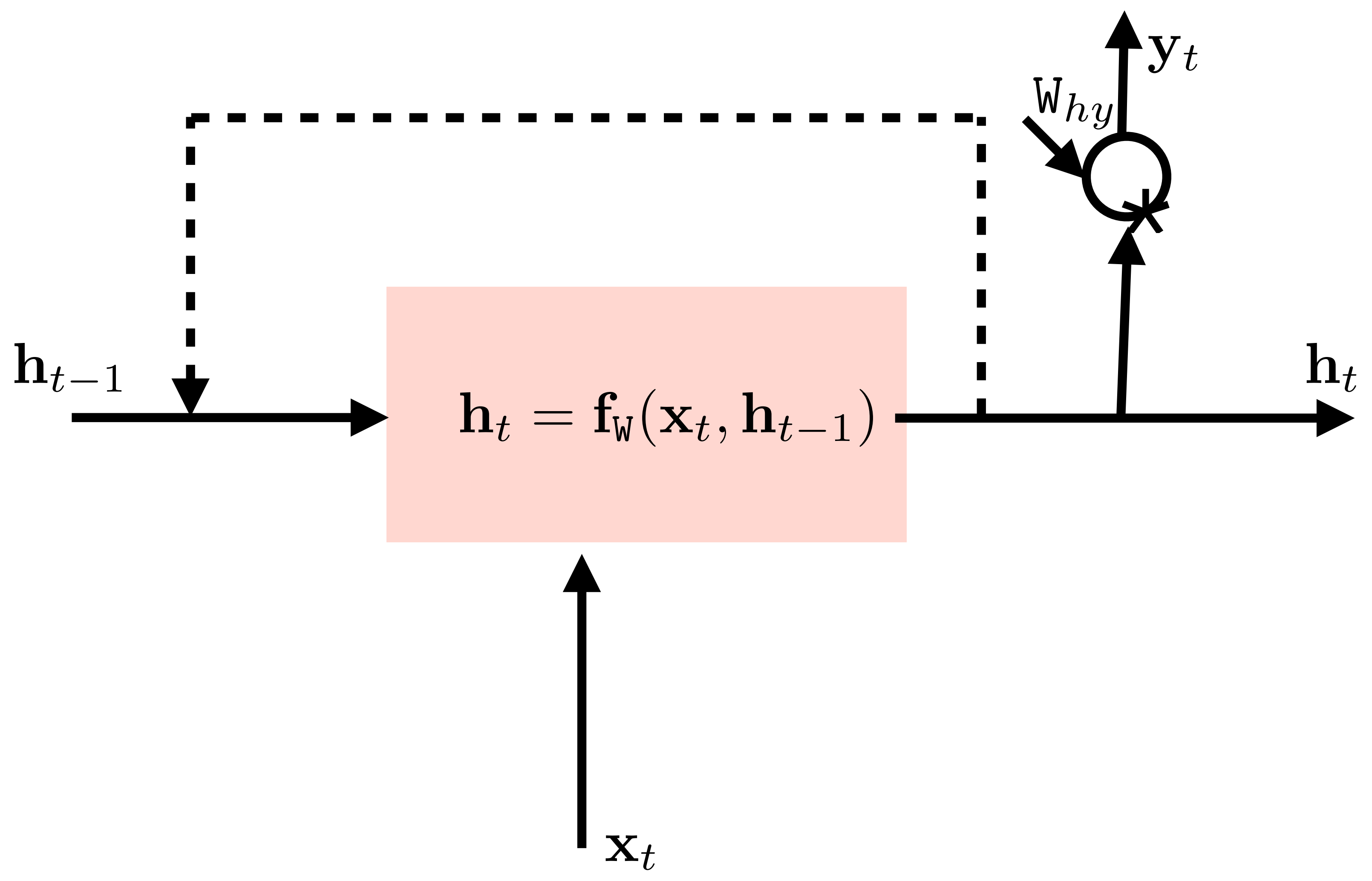


Simple recurrent block - feed-forward pass



Simple recurrent block - feed-forward pass



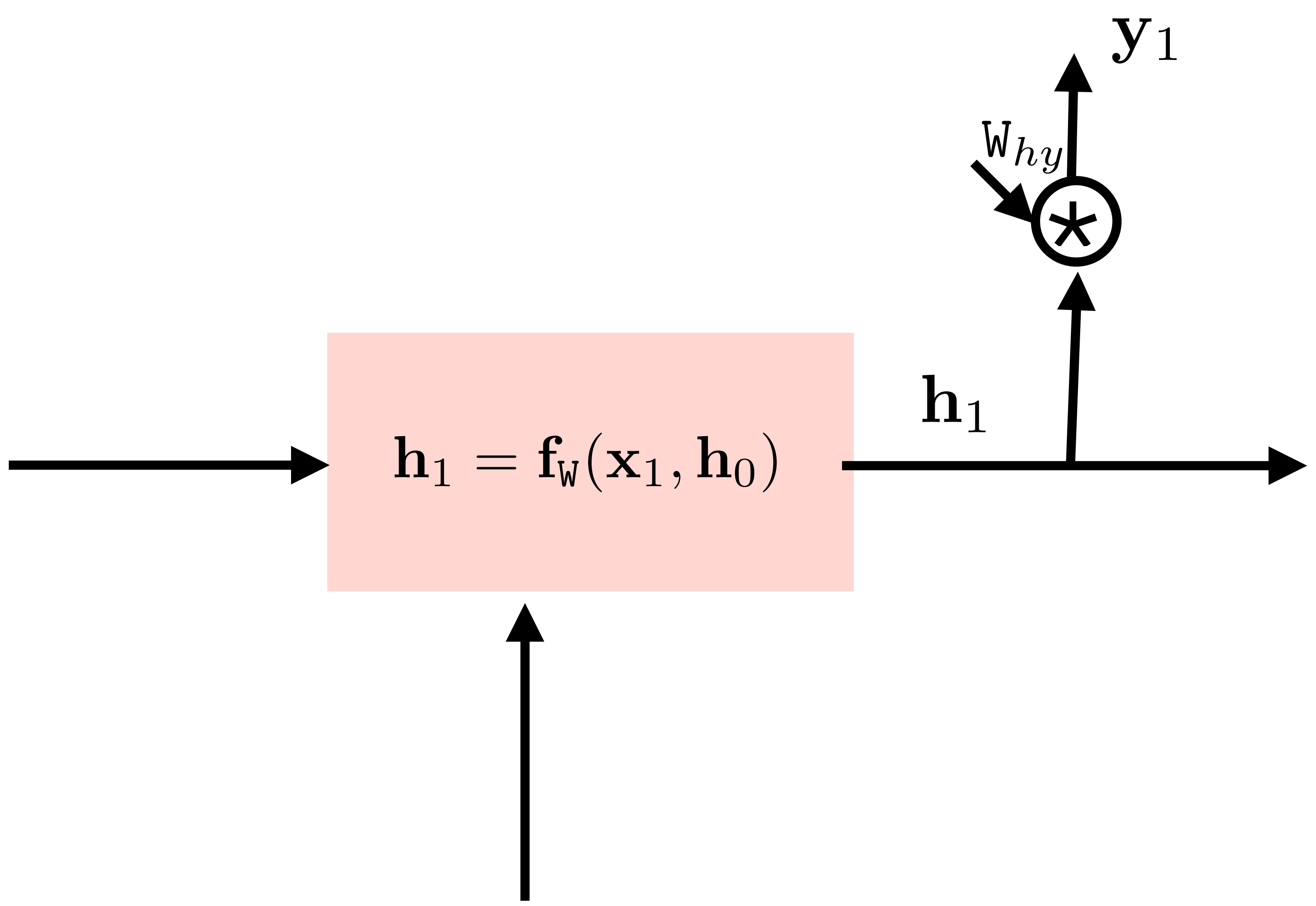


Given a finite input sequence: $\mathbf{h}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$

we remove the recurrent connection by:

- successive substitution of inputs and
- unrolling the net

Simple recurrent block - feed-forward pass

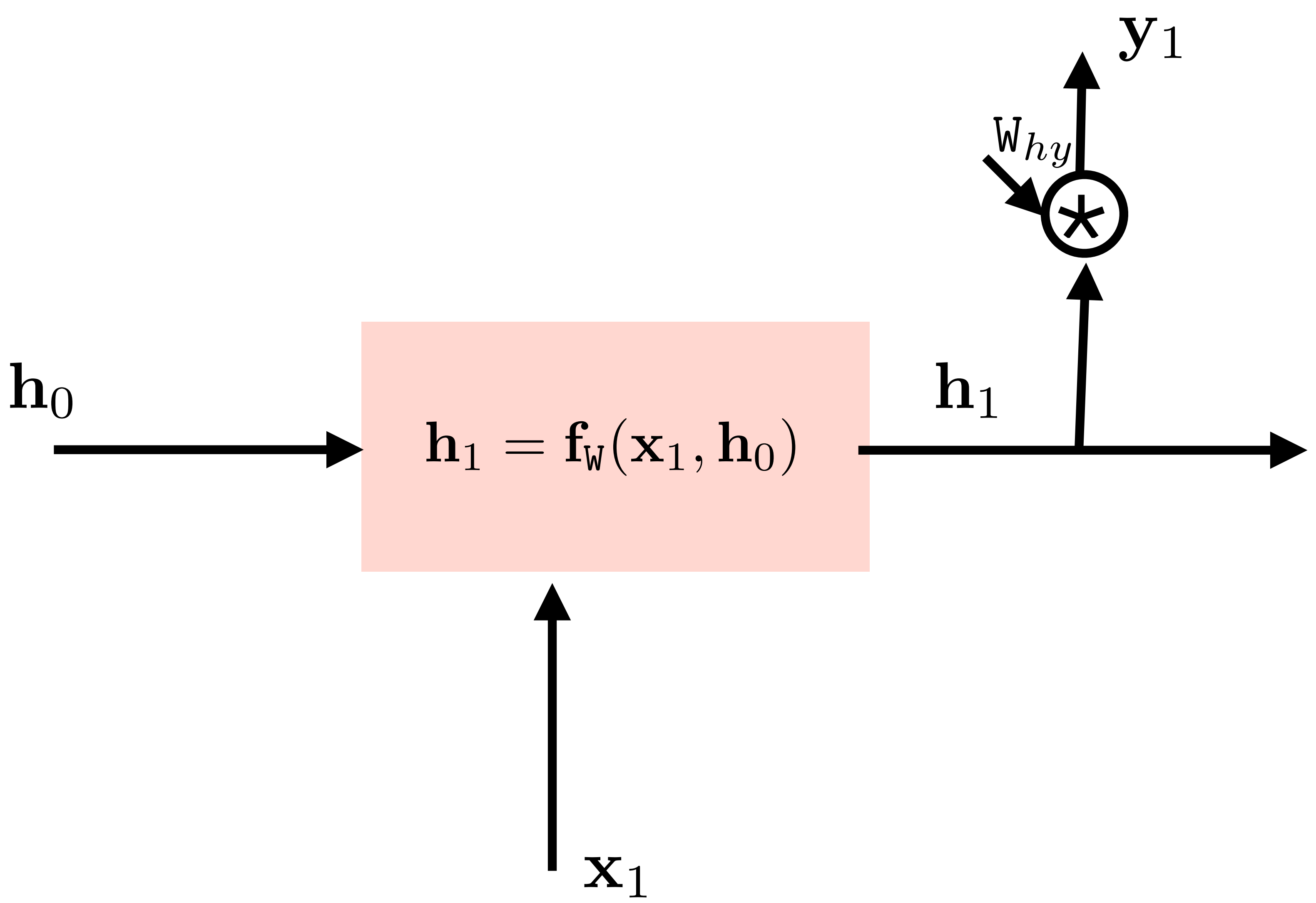


Given a finite input sequence: $\mathbf{h}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$

we remove the recurrent connection by:

- successive substitution of inputs and
- unrolling the net

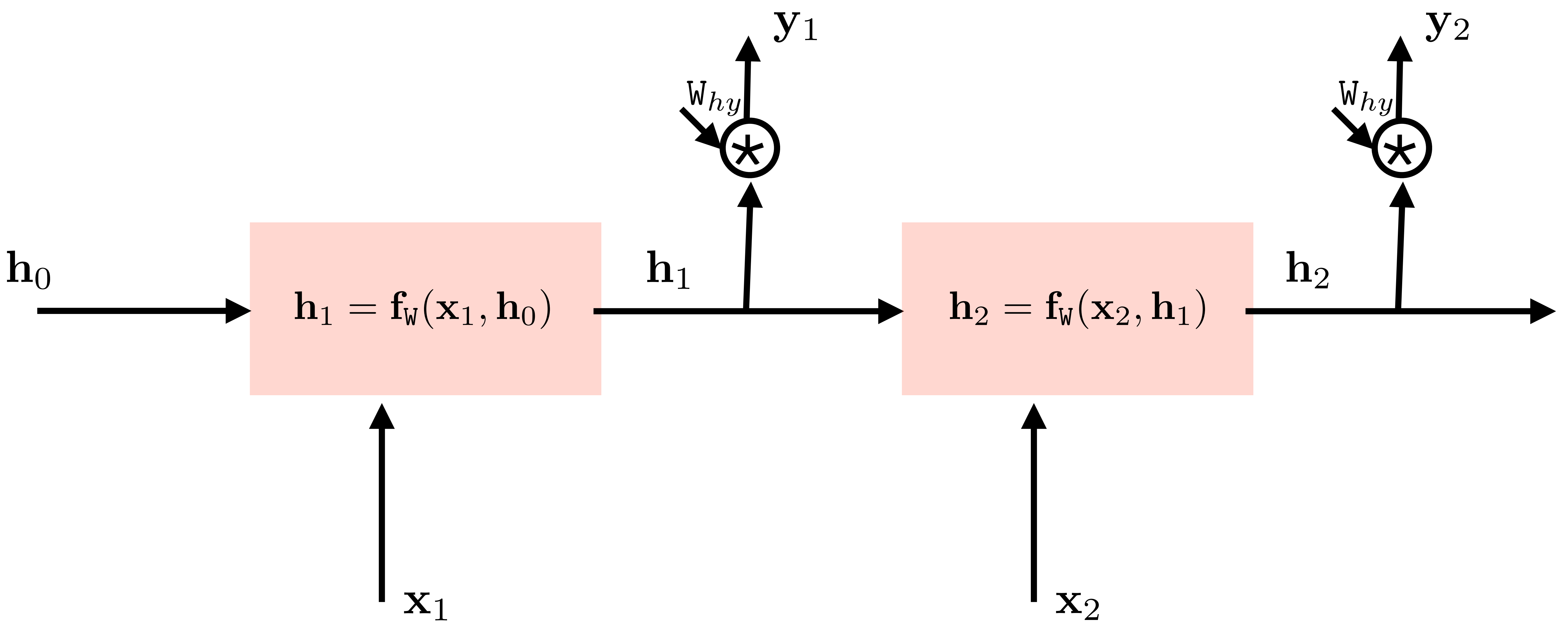
Simple recurrent block - feed-forward pass



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

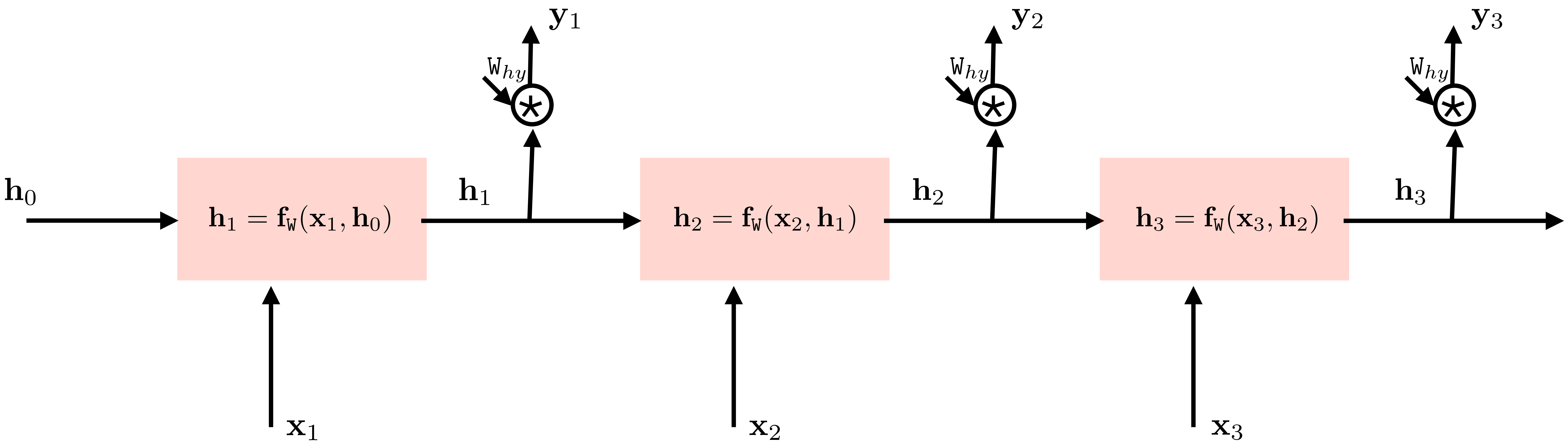


Given a finite input sequence:

we remove the recurrent connection by:

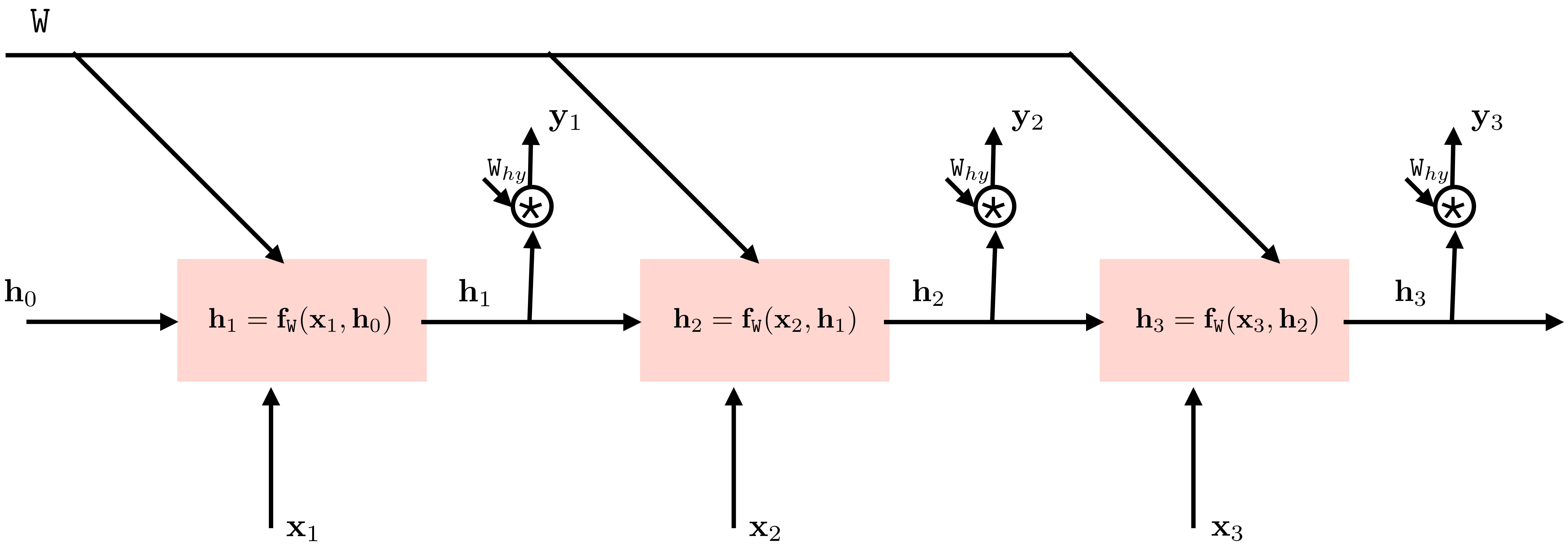
- successive substitution of inputs and
- unrolling the net

Simple recurrent block - feed-forward pass



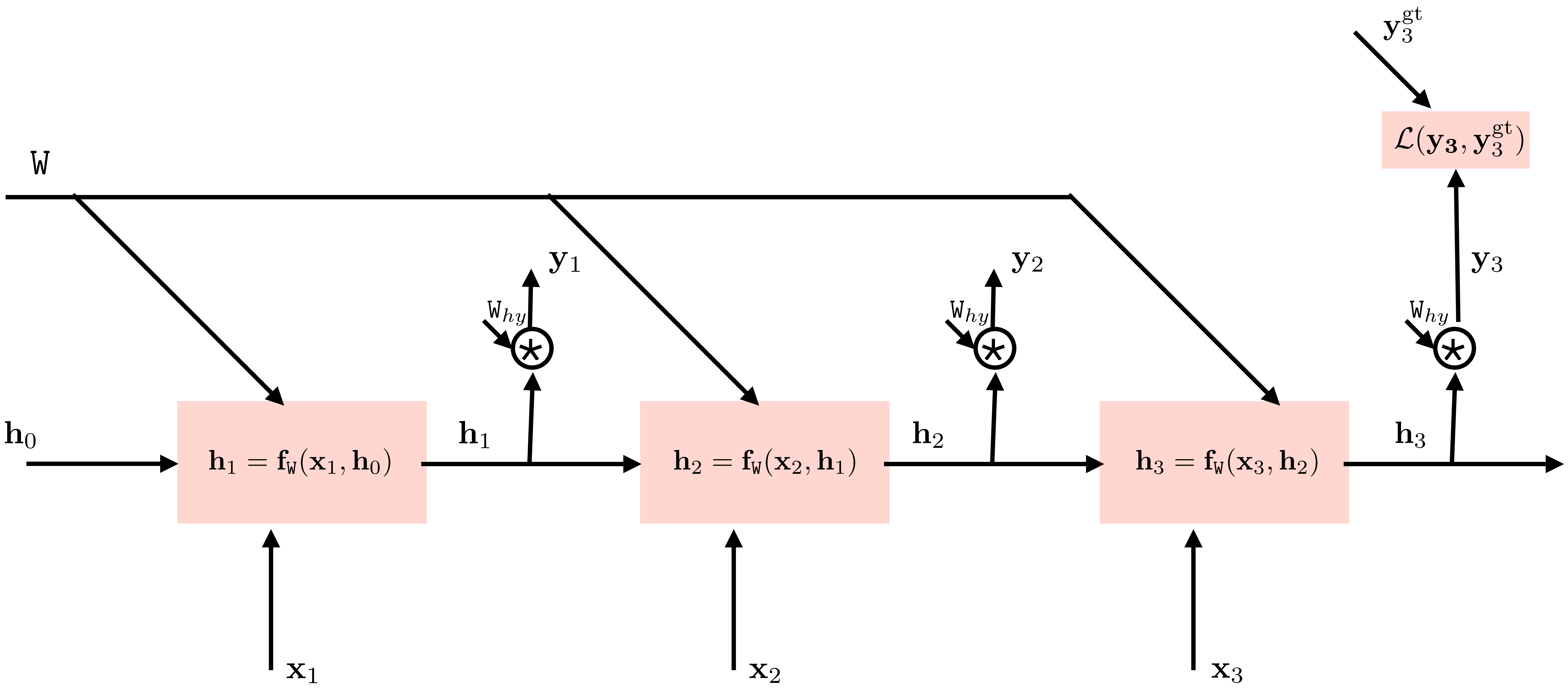
- Unrolled computational graph:
  - it is normal feedforward network

Simple recurrent block - feed-forward pass



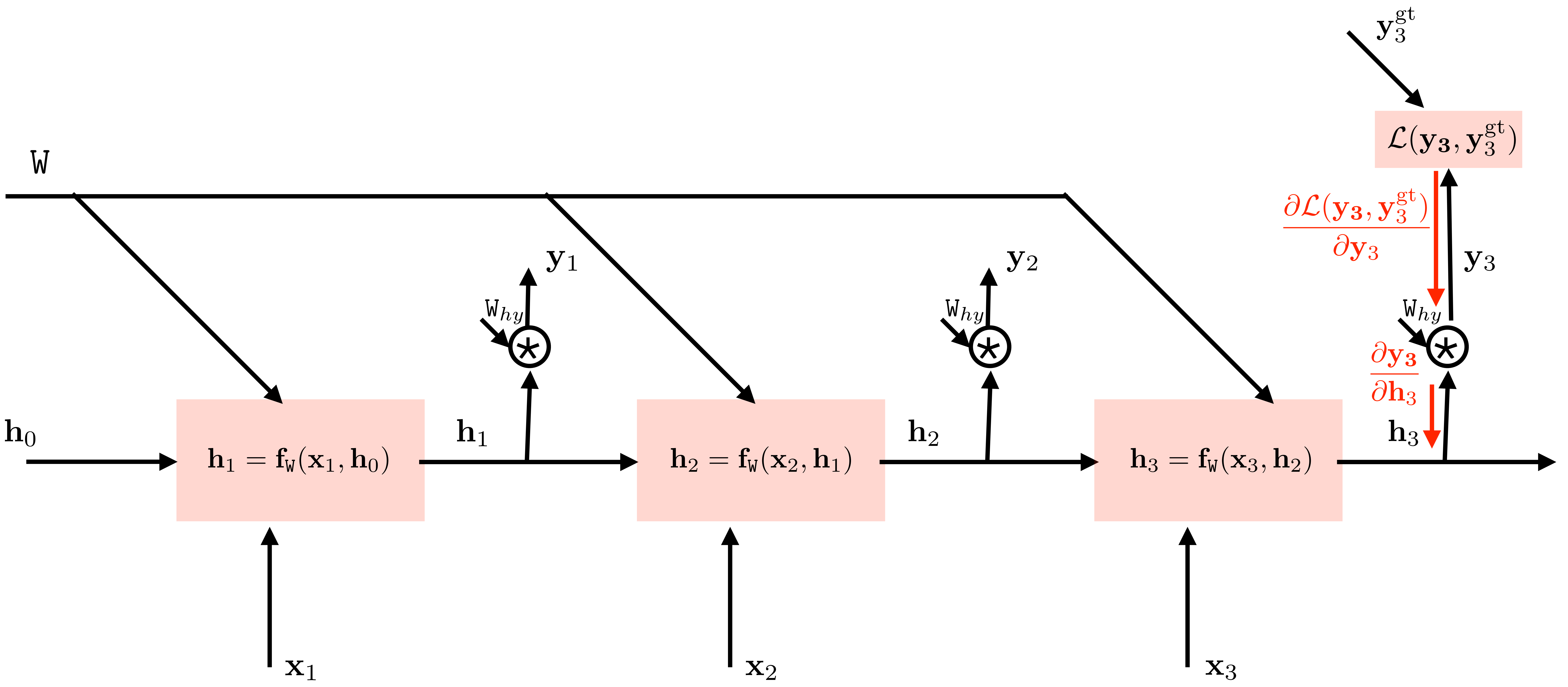
- Unrolled computational graph:
  - it is normal feedforward network
  - it consists of several same blocks with the same weights!

Simple recurrent block - backward pass



- Loss function:
  - cross-entropy loss on the last output only (for simplicity)

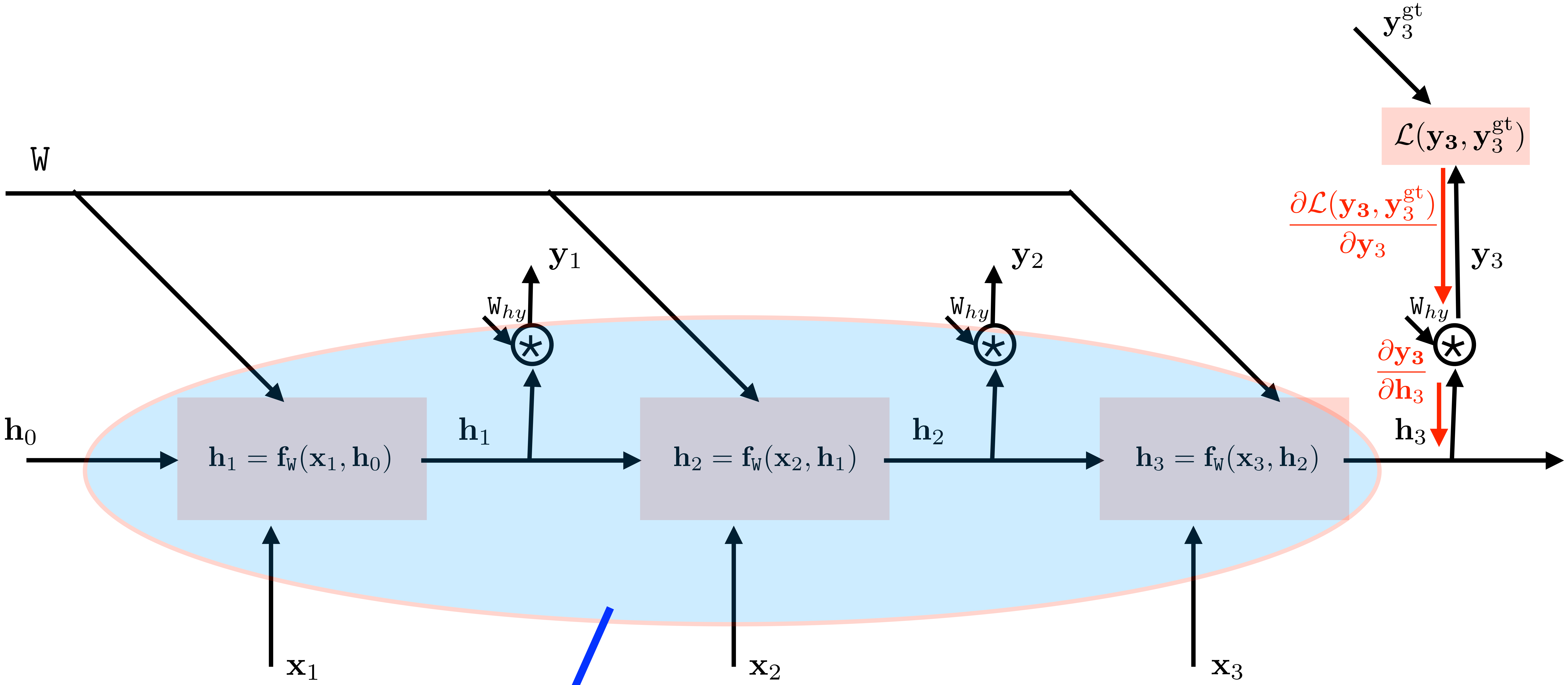
Simple recurrent block - backward pass



- Backprop:

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

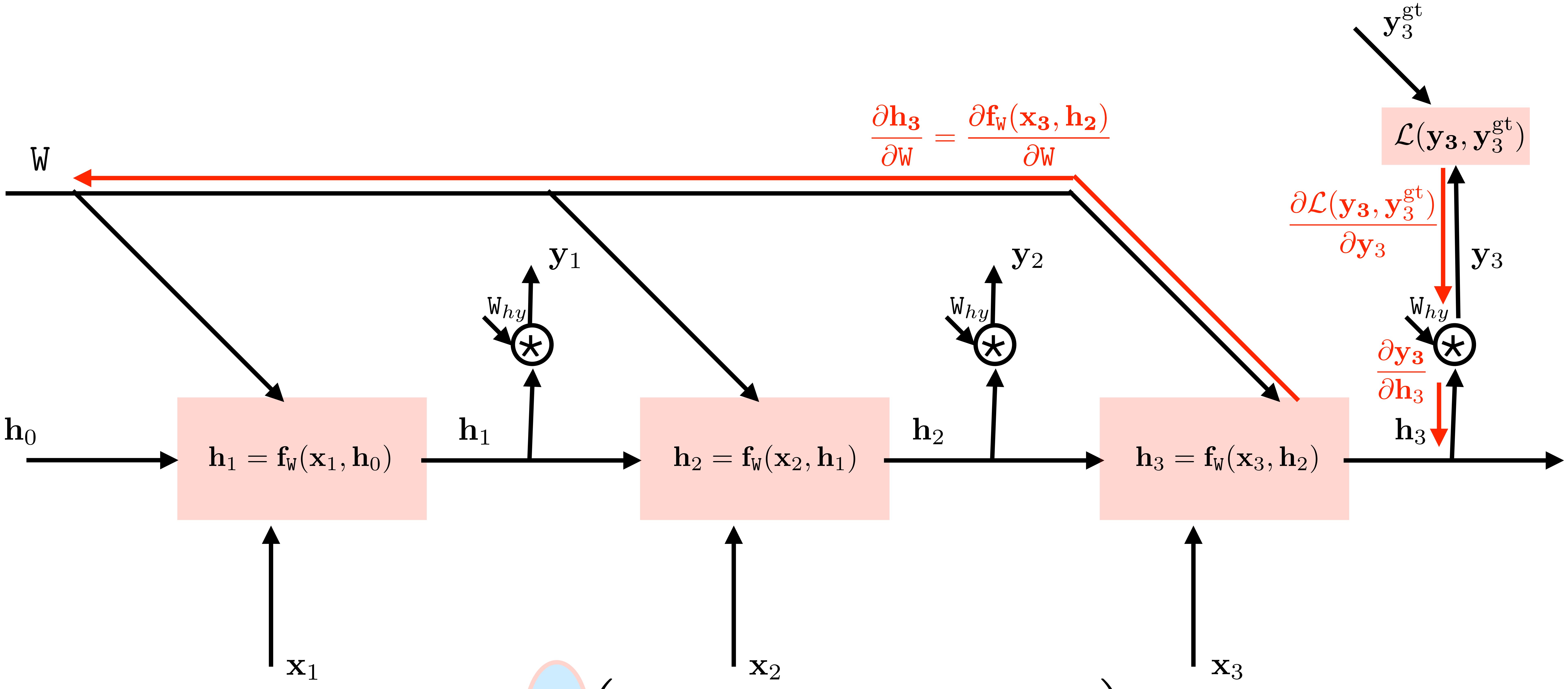
Simple recurrent block - backward pass



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

$$\frac{\partial \mathcal{L}(\mathbf{y_3}, \mathbf{y}_3^{gt})}{\partial \mathtt{W}} = \quad ? \qquad \mathbf{h}_3 = \mathbf{f}_{\mathtt{W}}\Big(\mathbf{x}_3, \mathbf{f}_{\mathtt{W}}\big(\mathbf{x}_2, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

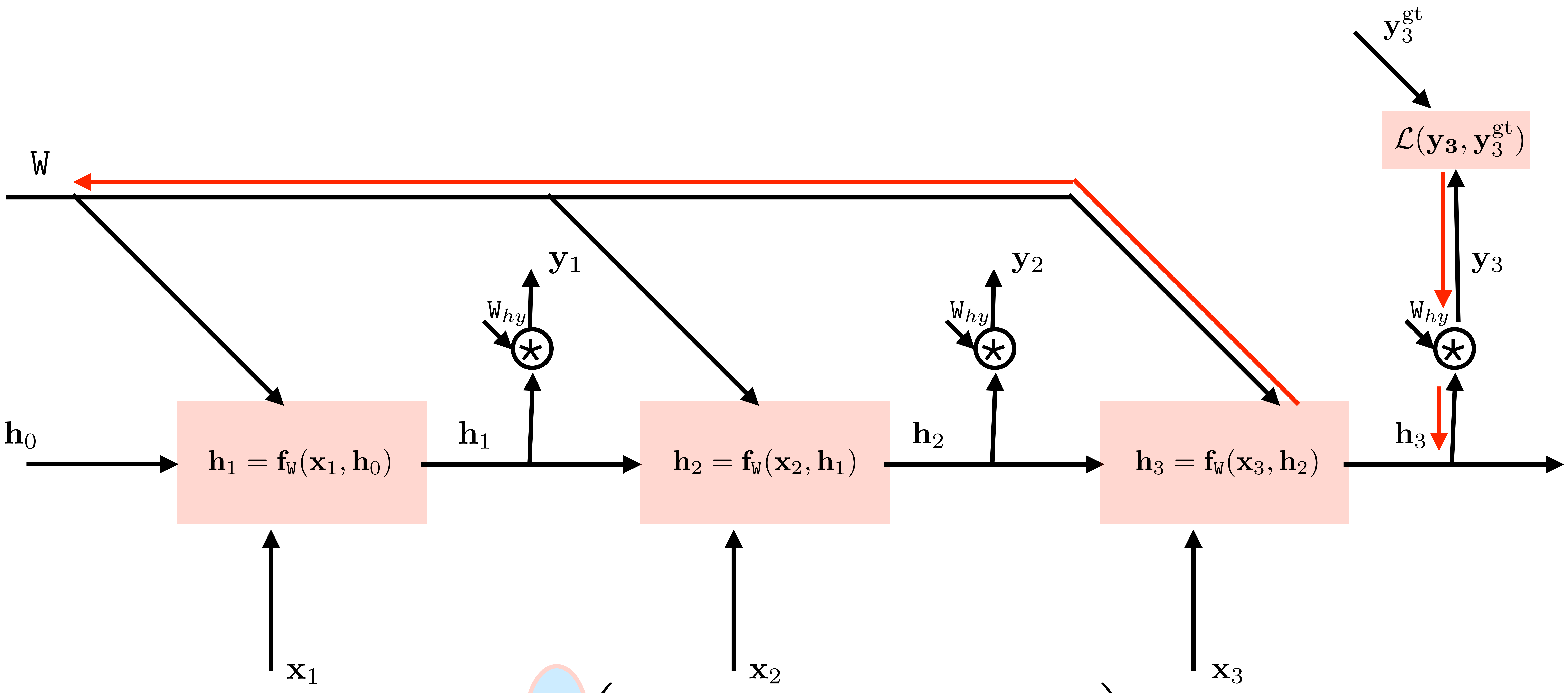
Simple recurrent block - backward pass



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

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

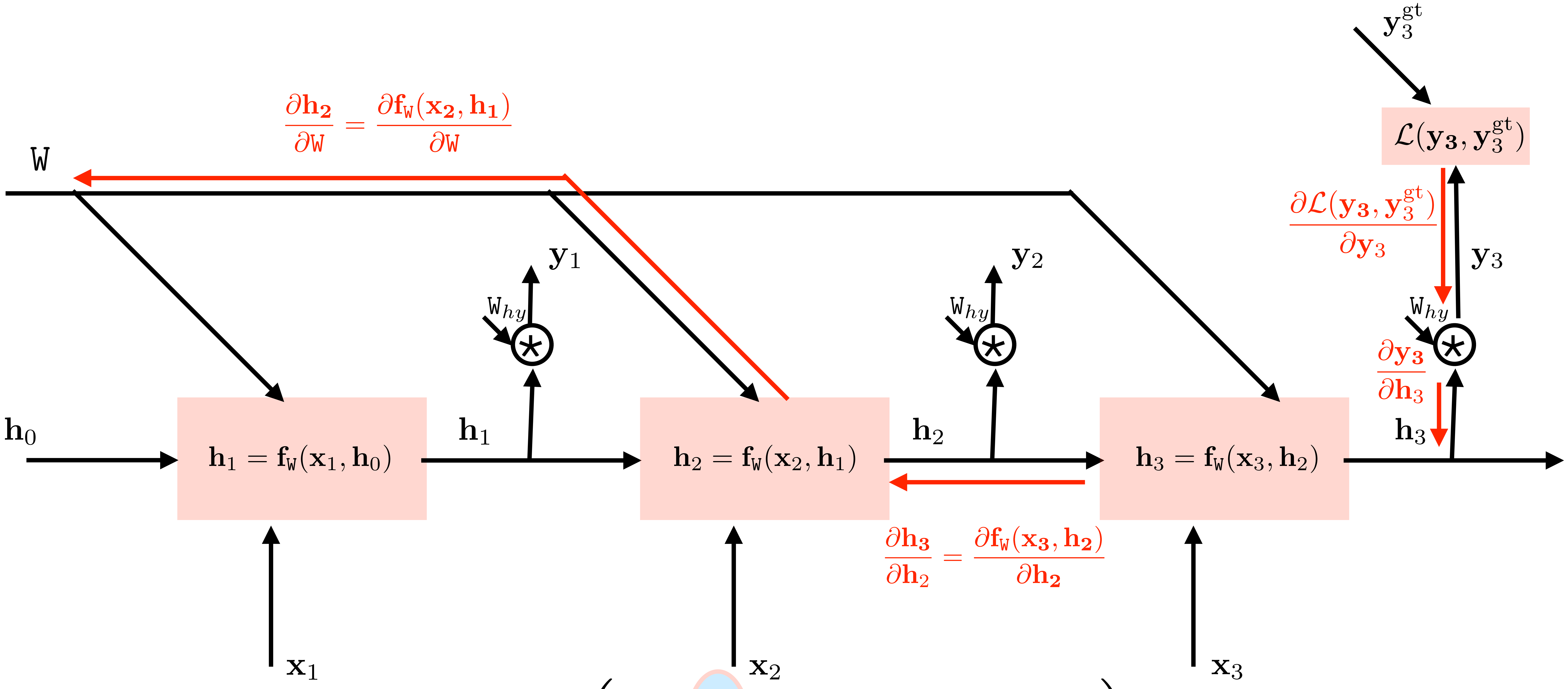
Simple recurrent block - backward pass



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

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

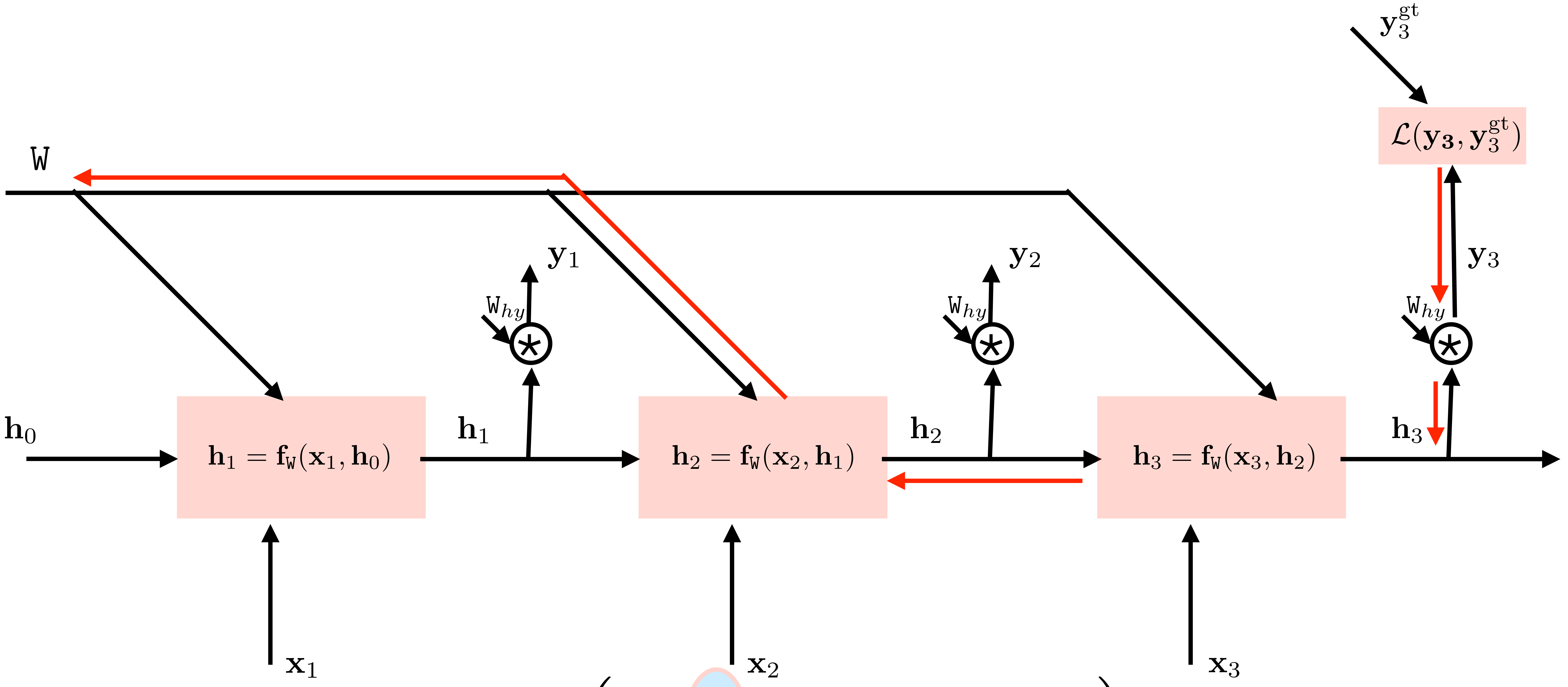
Simple recurrent block - backward pass



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

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

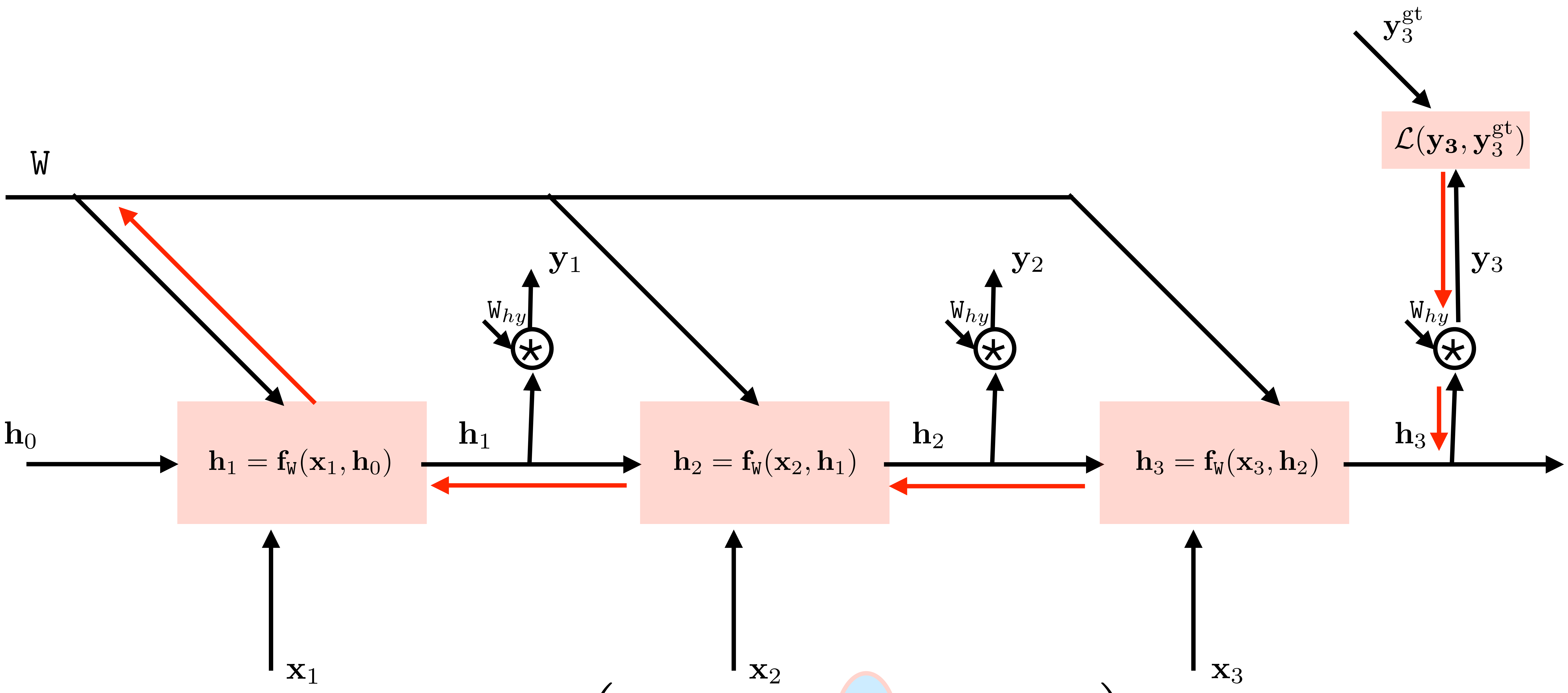
Simple recurrent block - backward pass



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

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

Simple recurrent block - backward pass



$$\mathbf{h}_3 = \mathrm{f}_W\Big(\mathbf{x}_3, \mathrm{f}_W\big(\mathbf{x}_2, \mathrm{f}_W(\mathbf{x}_1, \mathbf{h}_0)\big)\Big)$$

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

RNN vs feedforward network

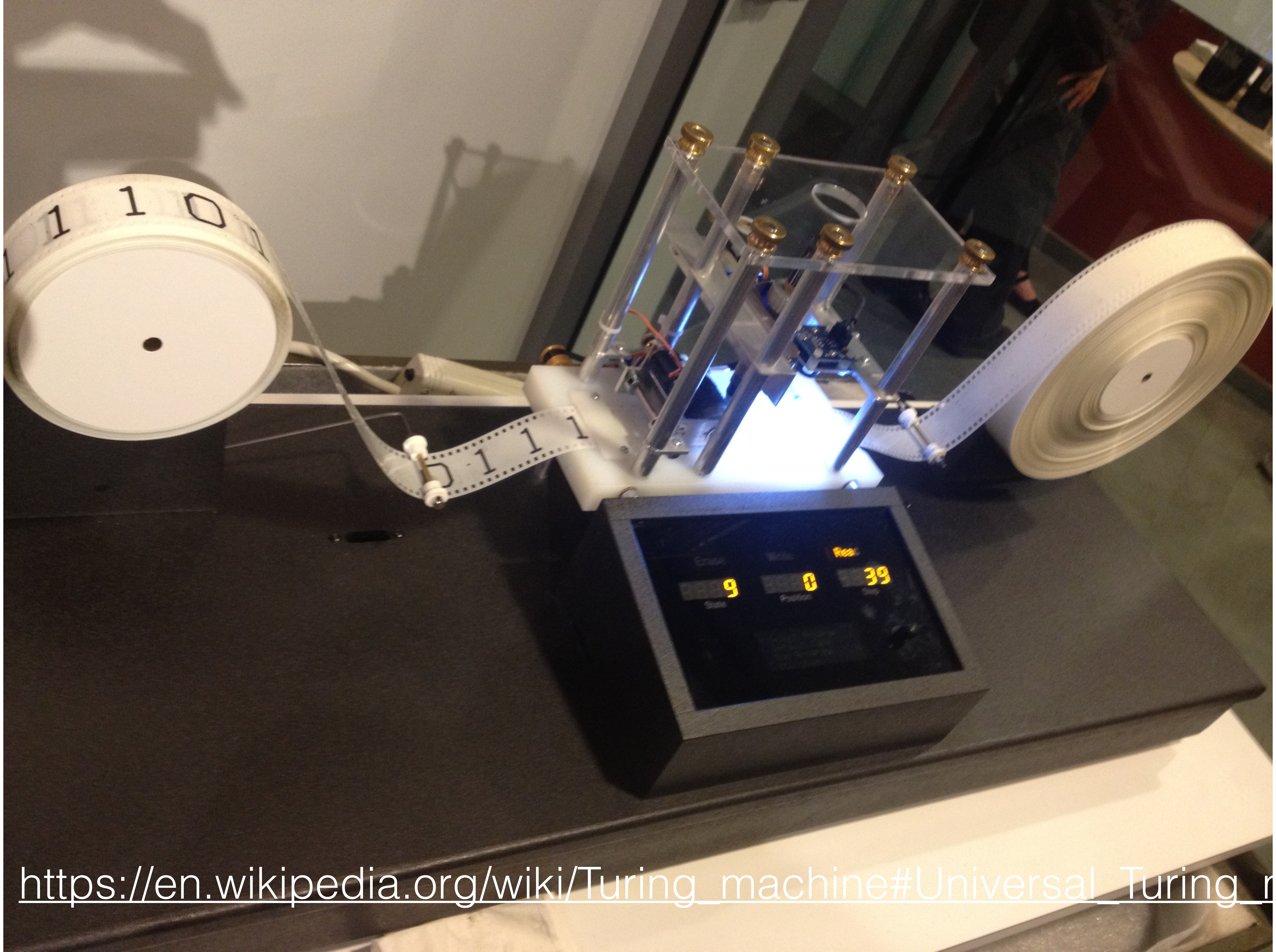
$$\mathbf{h}_t = \mathbf{f}_{\mathtt{W}}(\mathbf{x}_t, \mathbf{h}_{t-1})$$

$$\mathbf{h}_t = \mathbf{f}_{\mathtt{W}}(\mathbf{x}_t, \mathbf{f}_{\mathtt{W}}(\mathbf{x}_{t-1}, \ldots \mathbf{f}_{\mathtt{W}}(\mathbf{x}_1, \mathbf{h}_0))$$

$$\mathbf{h}_t = \mathbf{g}^{(t)}(\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots \mathbf{x}_1)$$

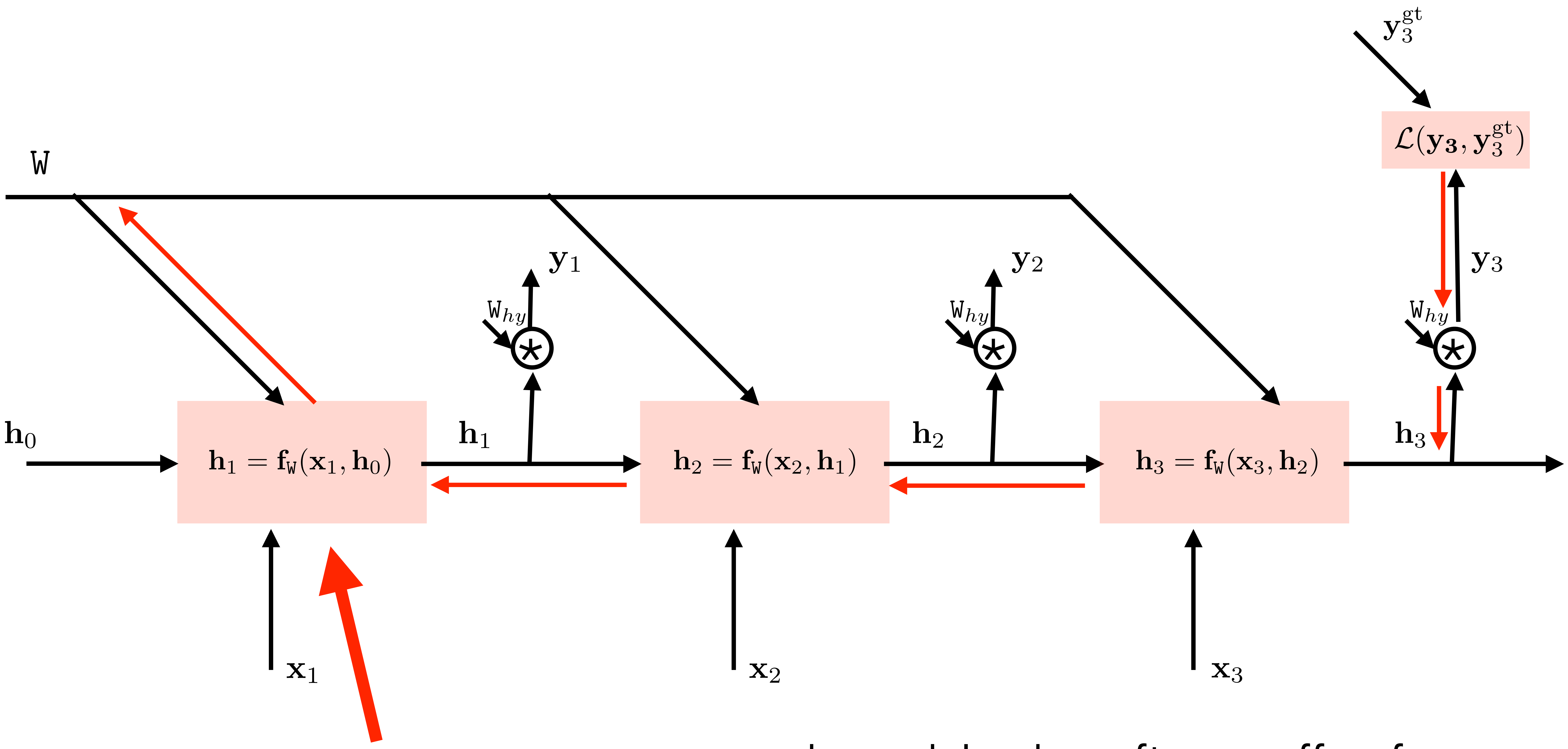
- Advantage of RNN wrt stacking the input sequence into a long vector and using a common feedforward network:
  - RNN works for different sequence lengths
  - RNN share weights between different time instances (similarly as convolution on spatial domain).

- RNN is universal (can compute any function computable by Turing machine)



https://en.wikipedia.org/wiki/Turing_machine#Universal_Turing_m

Simple recurrent block - backward pass

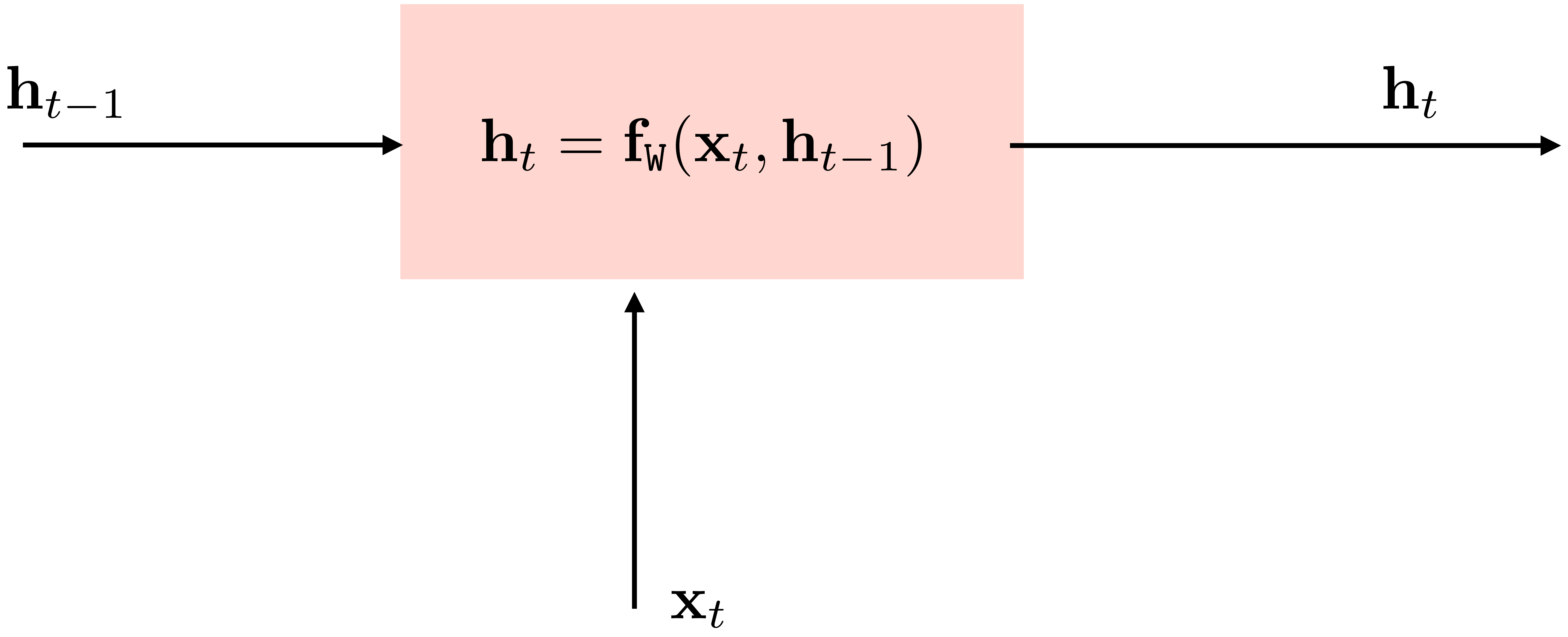


deep blocks often suffer from vanishing gra

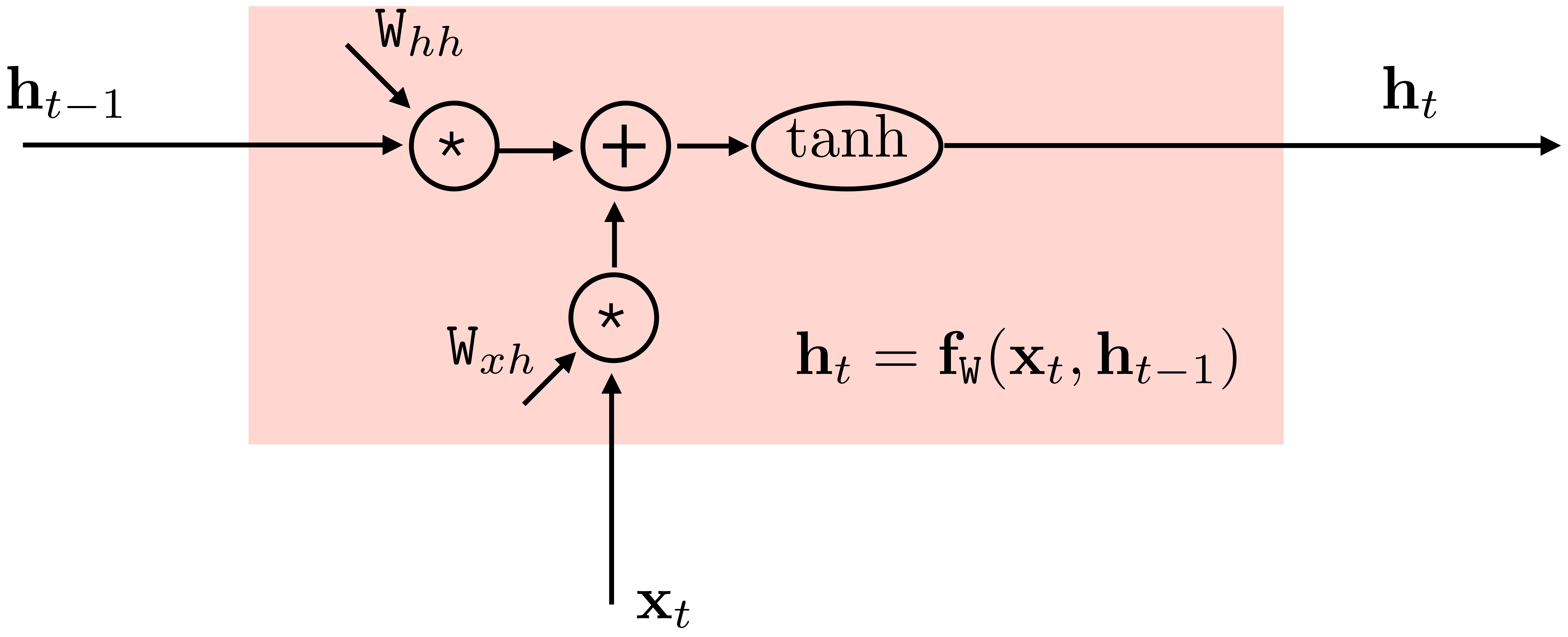
=> better structure needed

LSTM (kind of ResNet for recurrent networks)

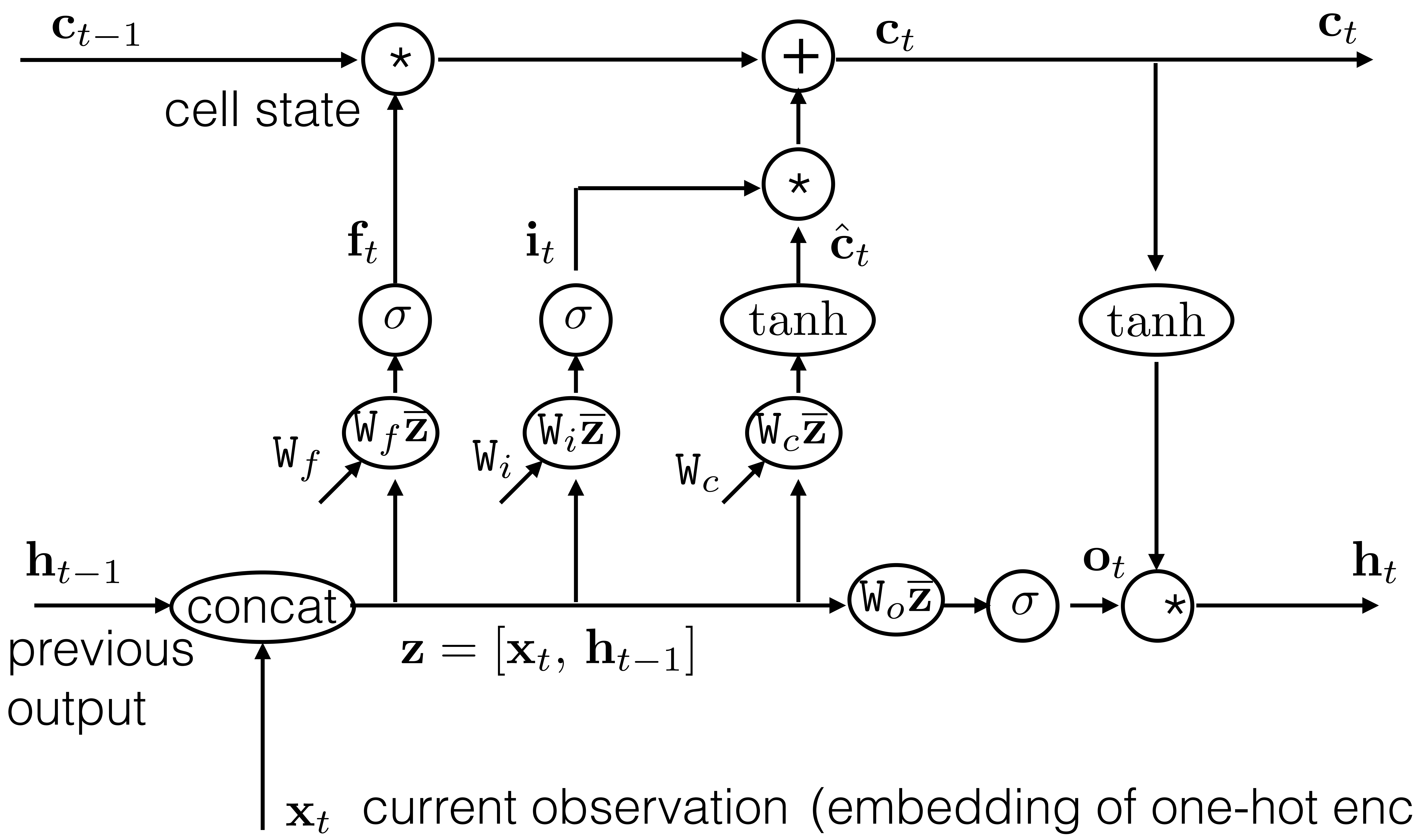
Simple recurrent block



Simple recurrent block

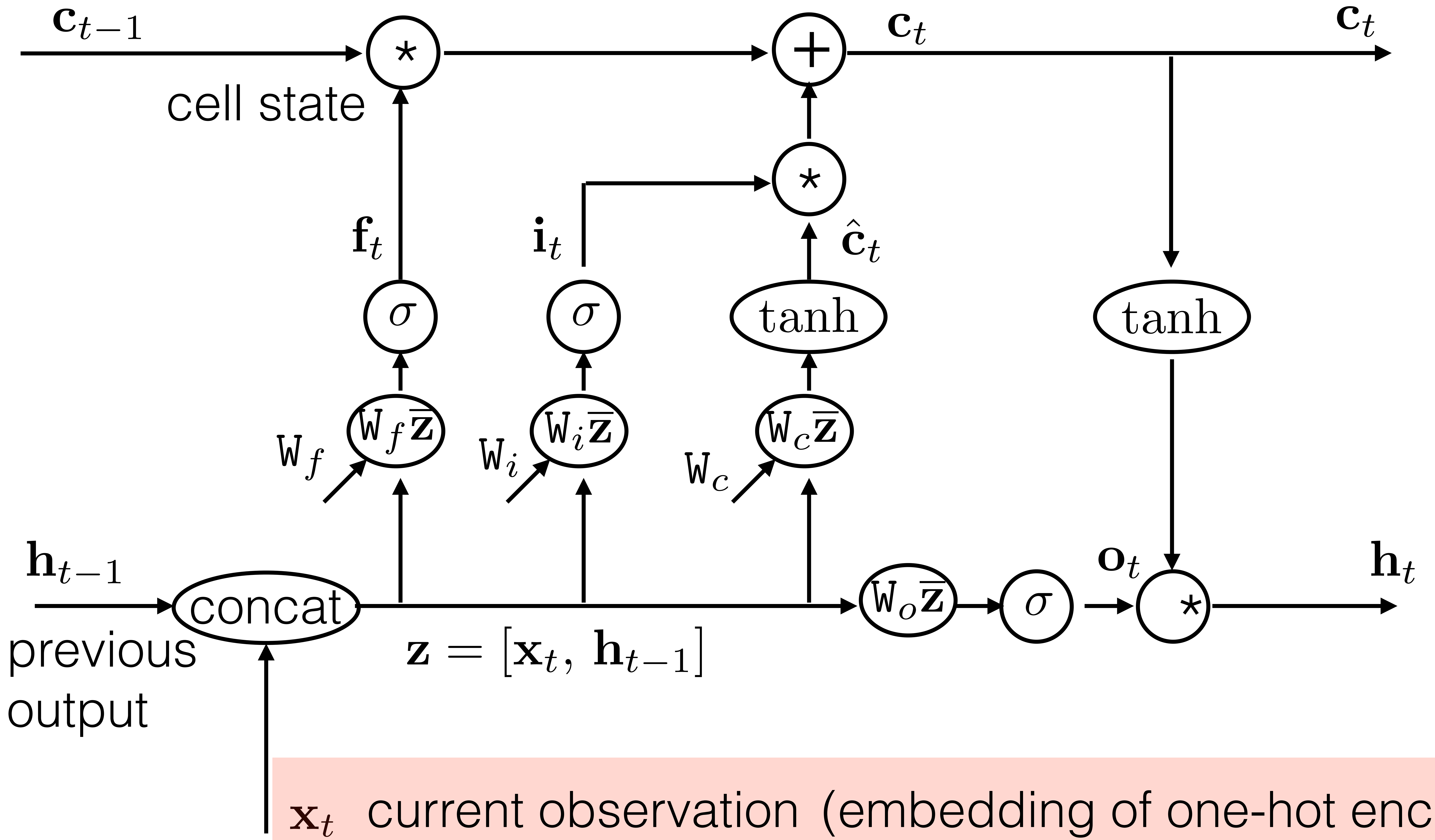


LSTM block



"I live with my parents, …"

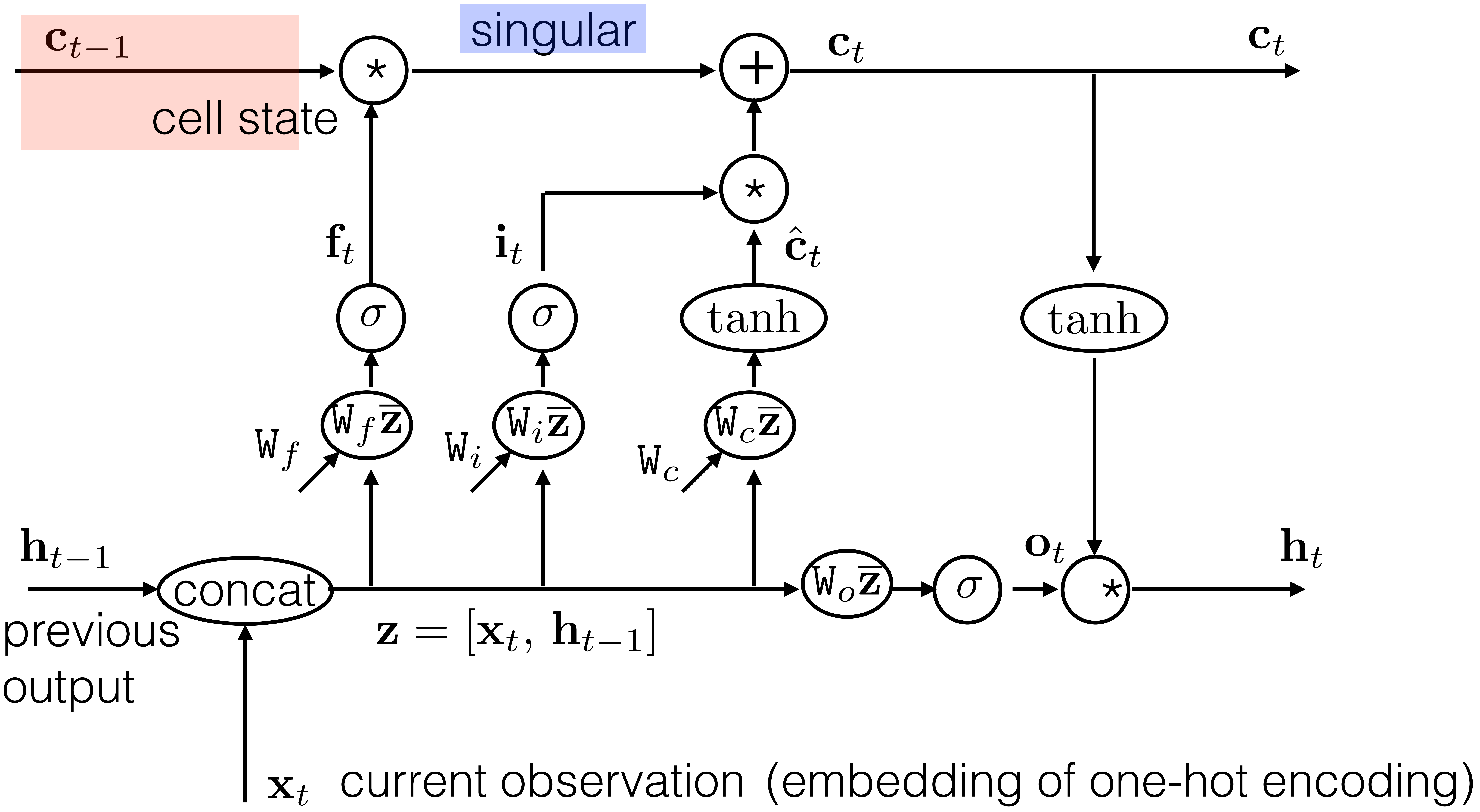
LSTM block



$\mathbf{x}_t$ current observation (embedding of one-hot encoding)

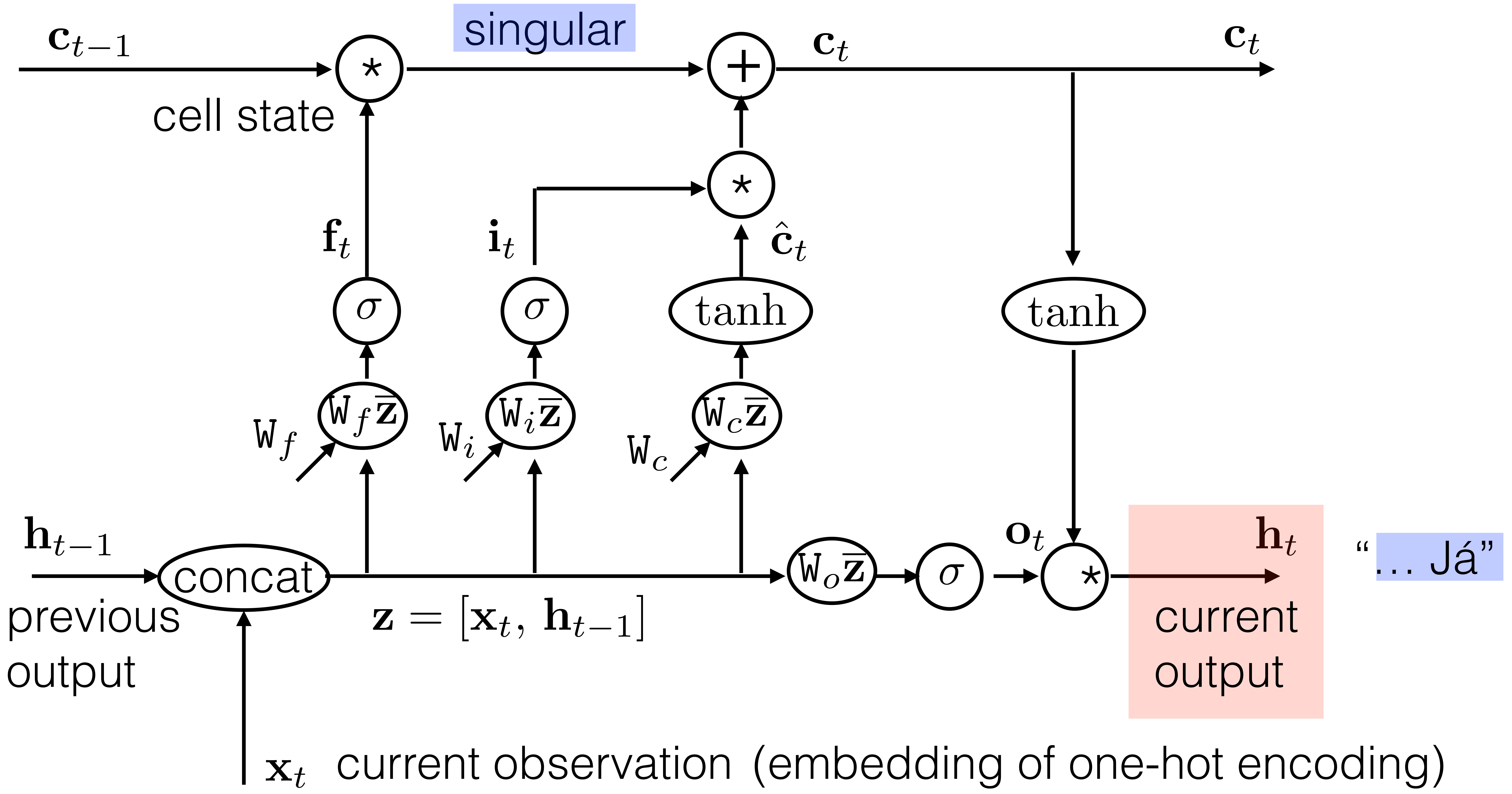
"I live with my parents, …"

LSTM block



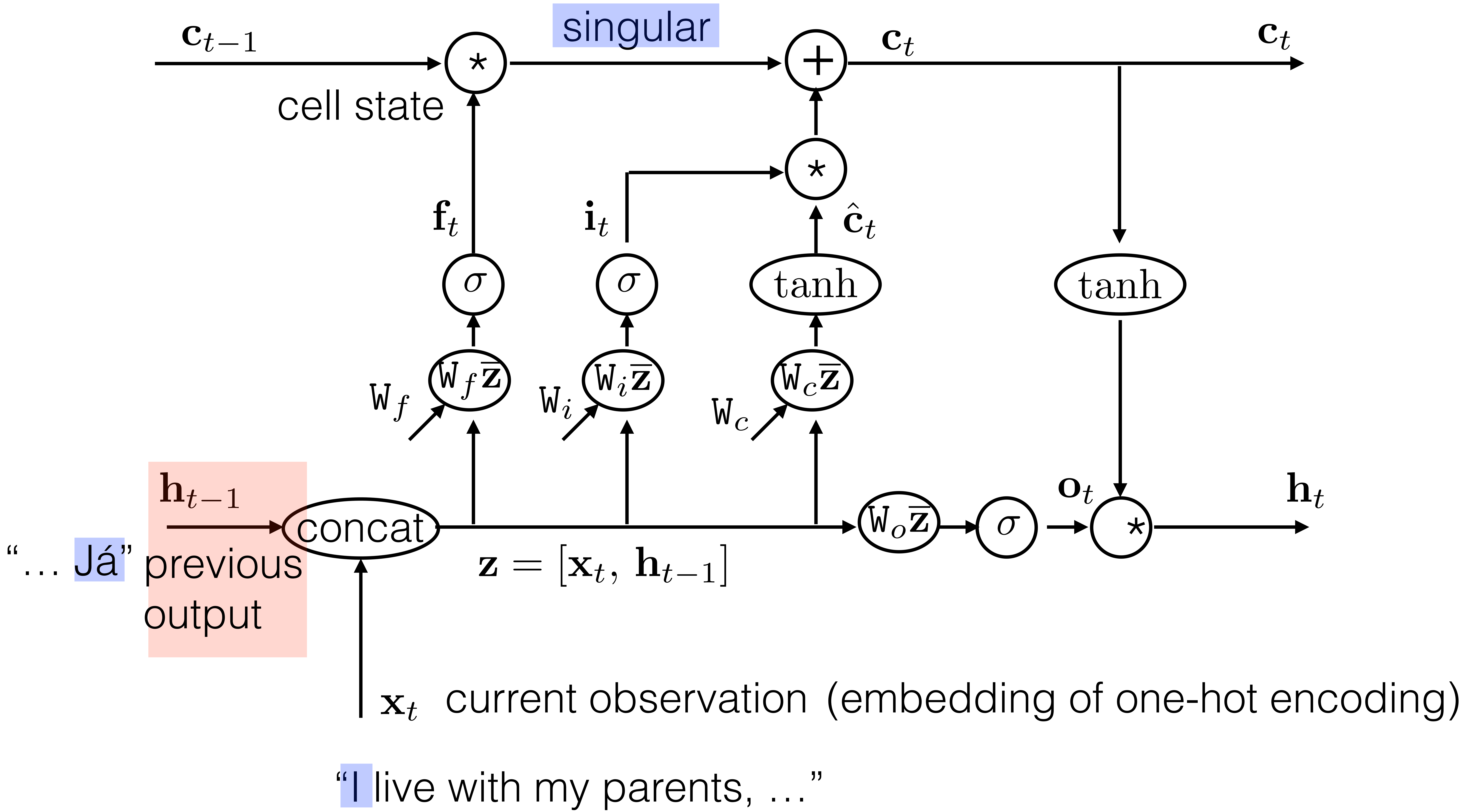
"I live with my parents, …"

LSTM block



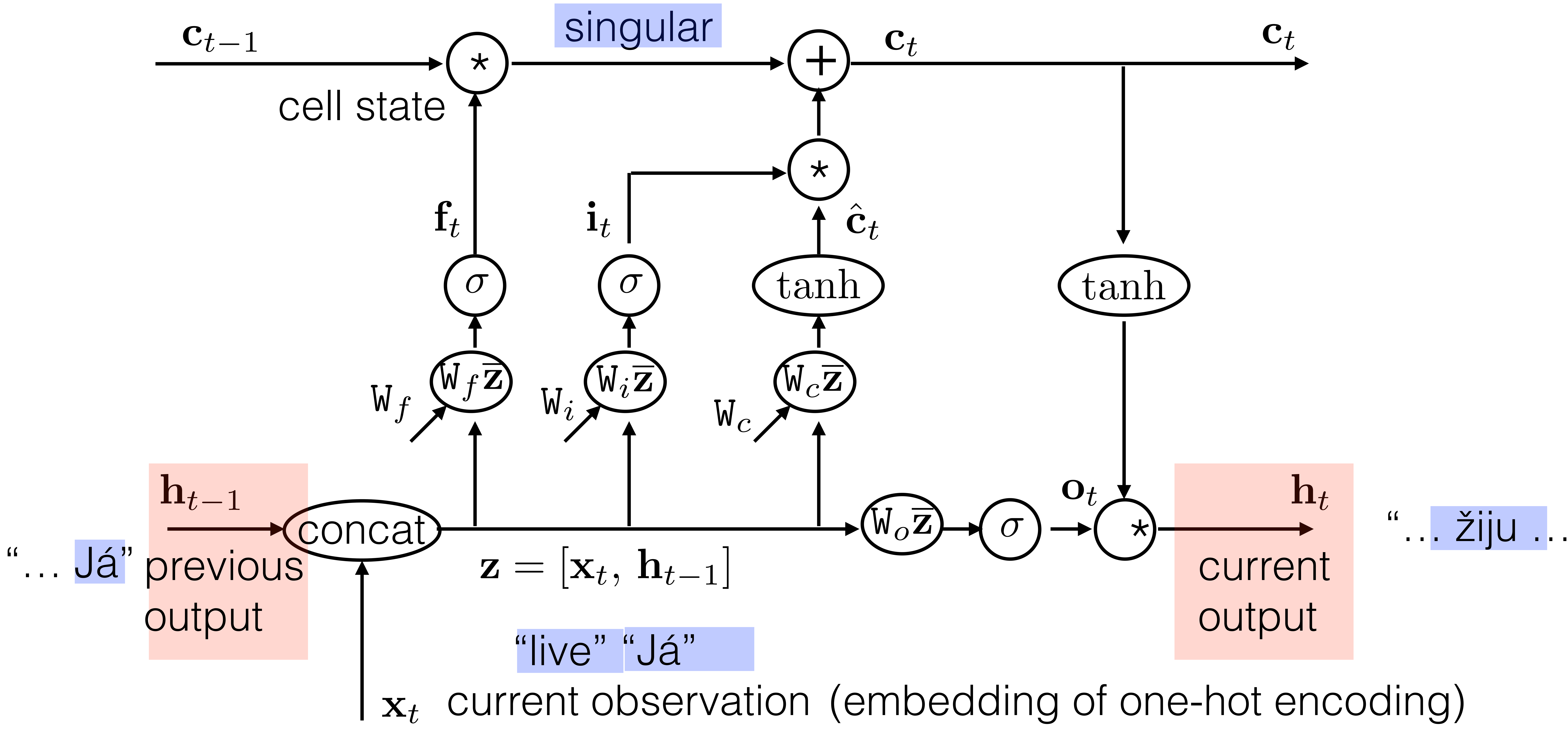
"I live with my parents, …"

LSTM block



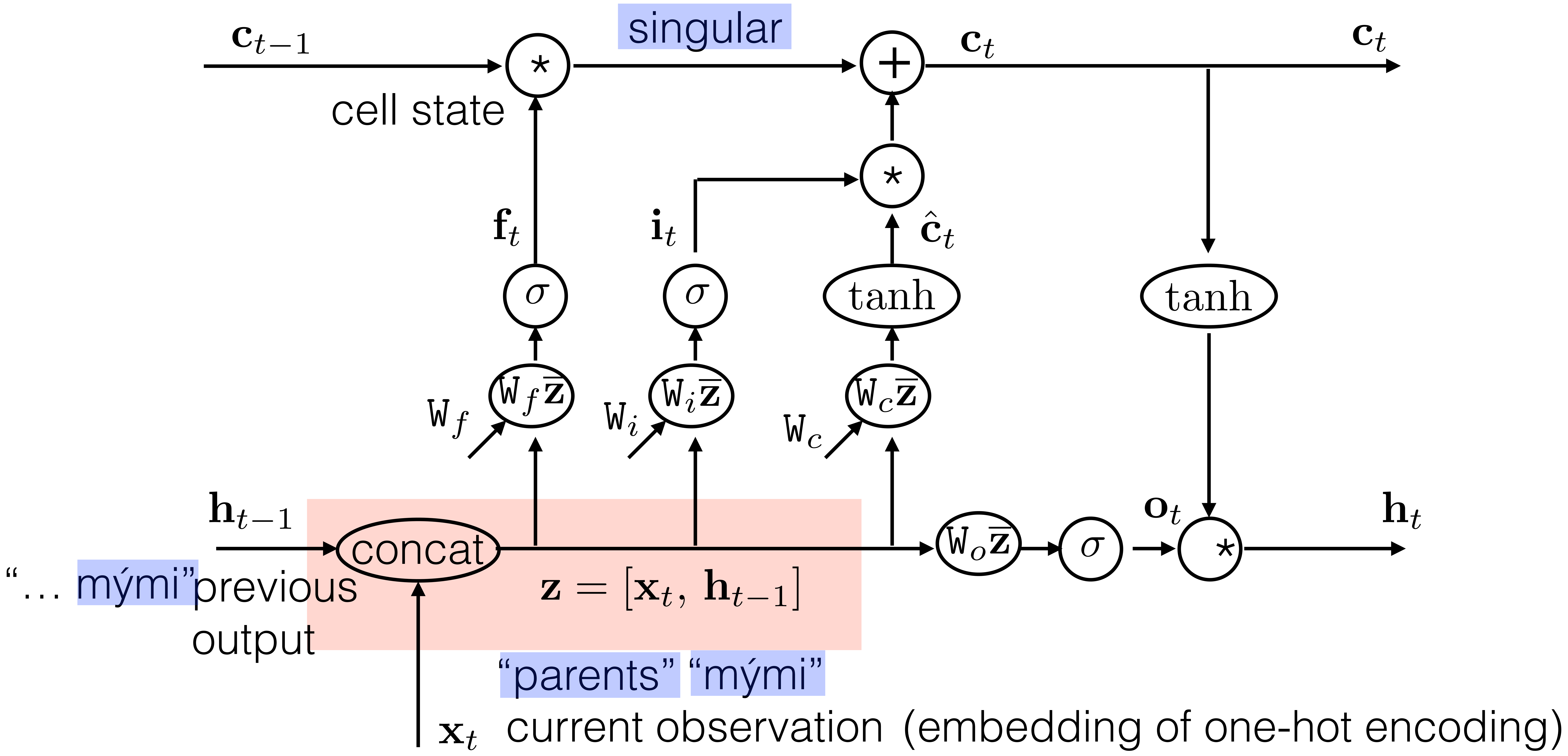
$\mathbf{c}_{t-1}$
singular
cell state
$\mathbf{c}_t$
$\mathbf{c}_t$
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$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

LSTM block

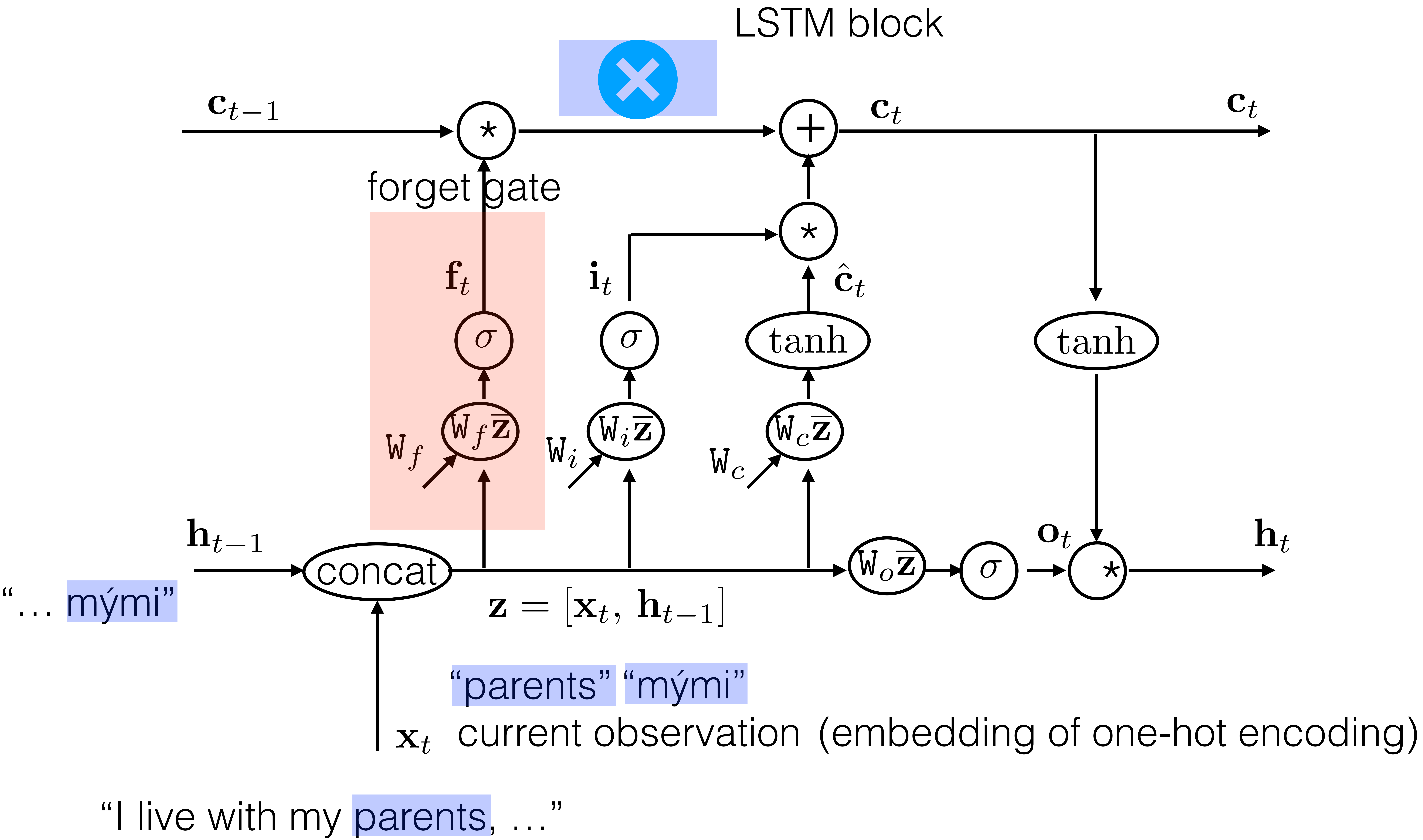


"I live with my parents, …"

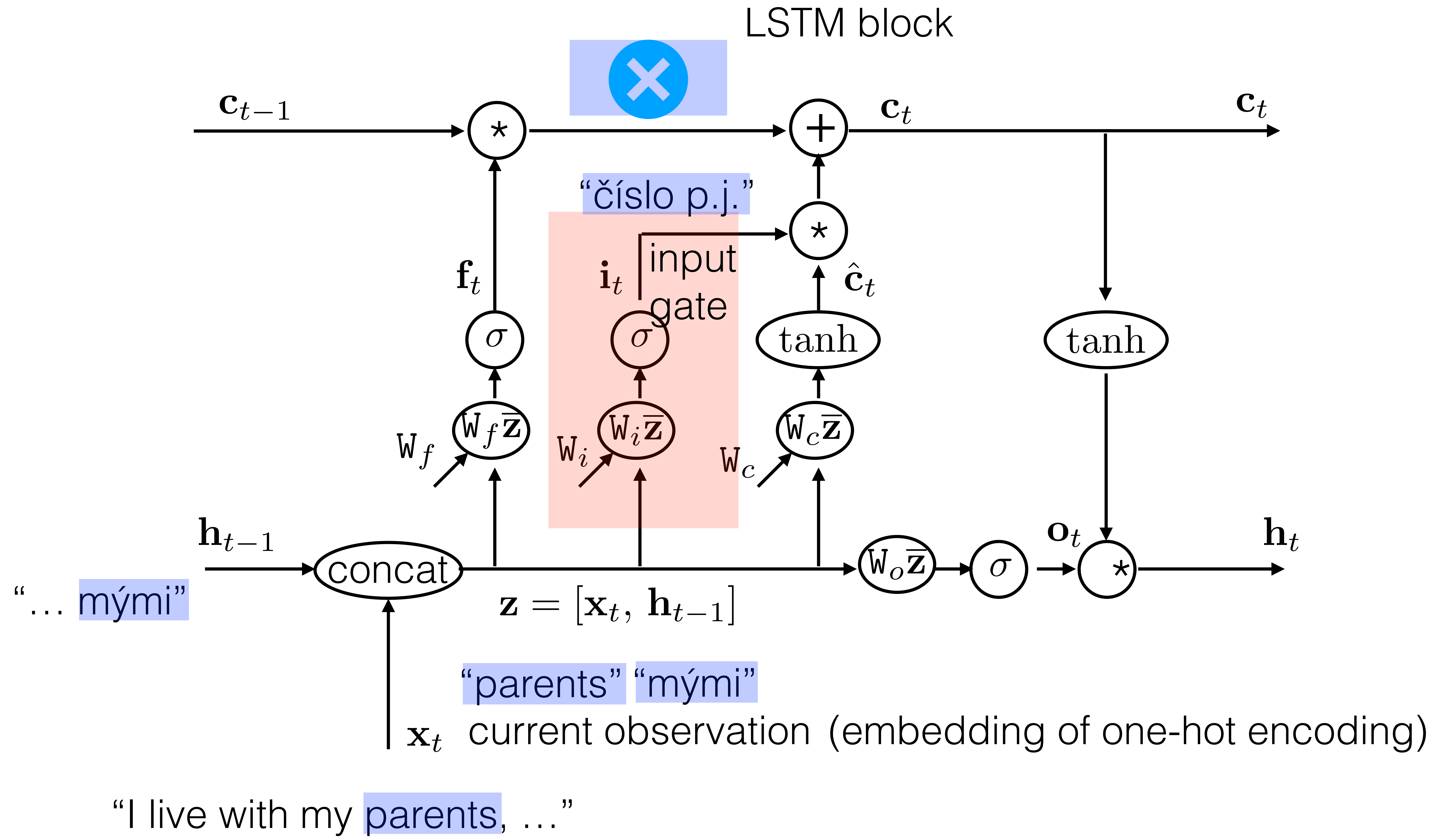
LSTM block



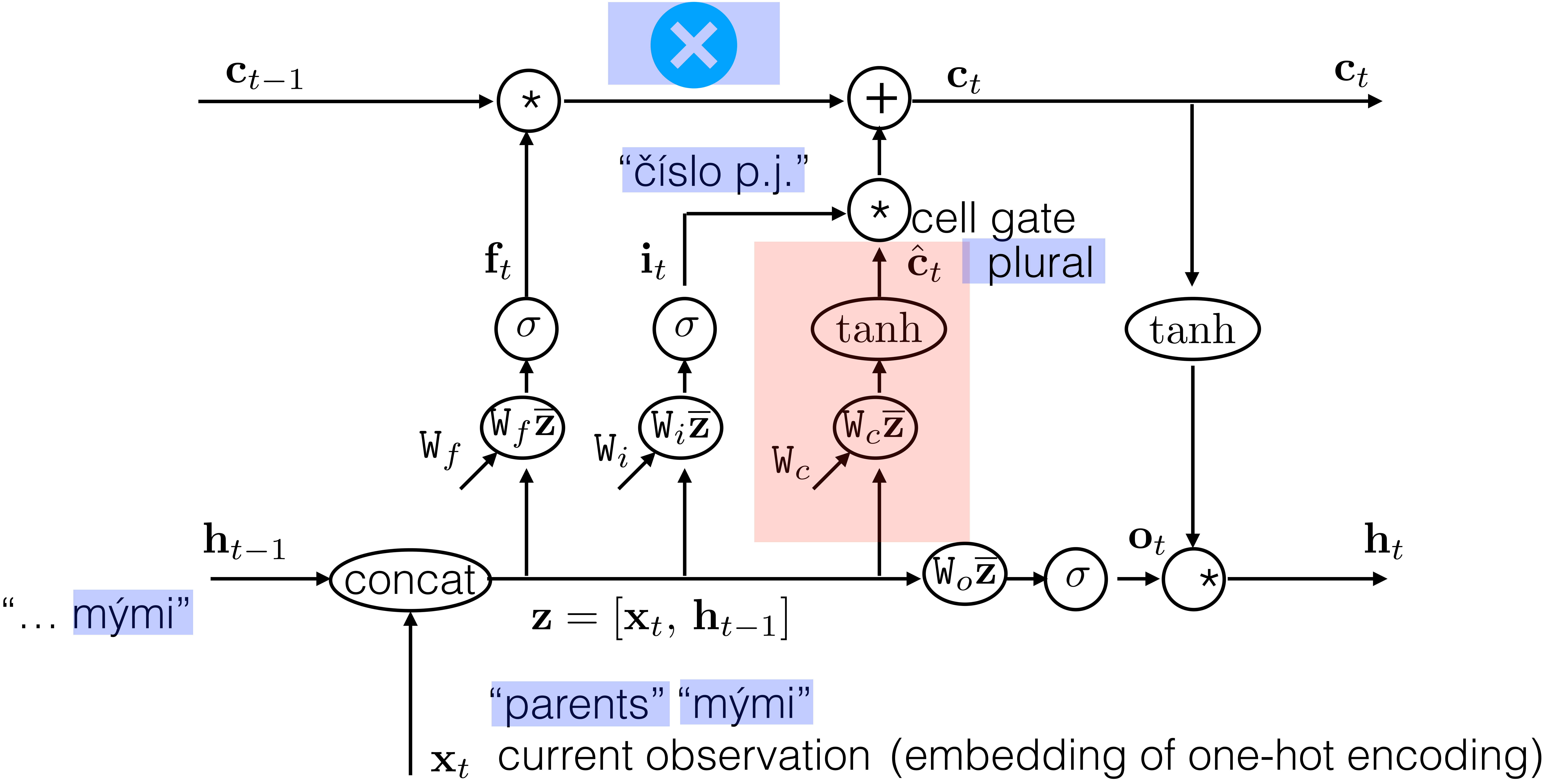
“I live with my parents, …”



LSTM block
$\mathbf{c}_{t-1}$
$*$
$+$
$\mathbf{c}_t$
$\mathbf{c}_t$
forget gate
$*$
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
concat
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
$\mathbf{o}_t$
$*$
$\mathbf{h}_t$
"… 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, …"

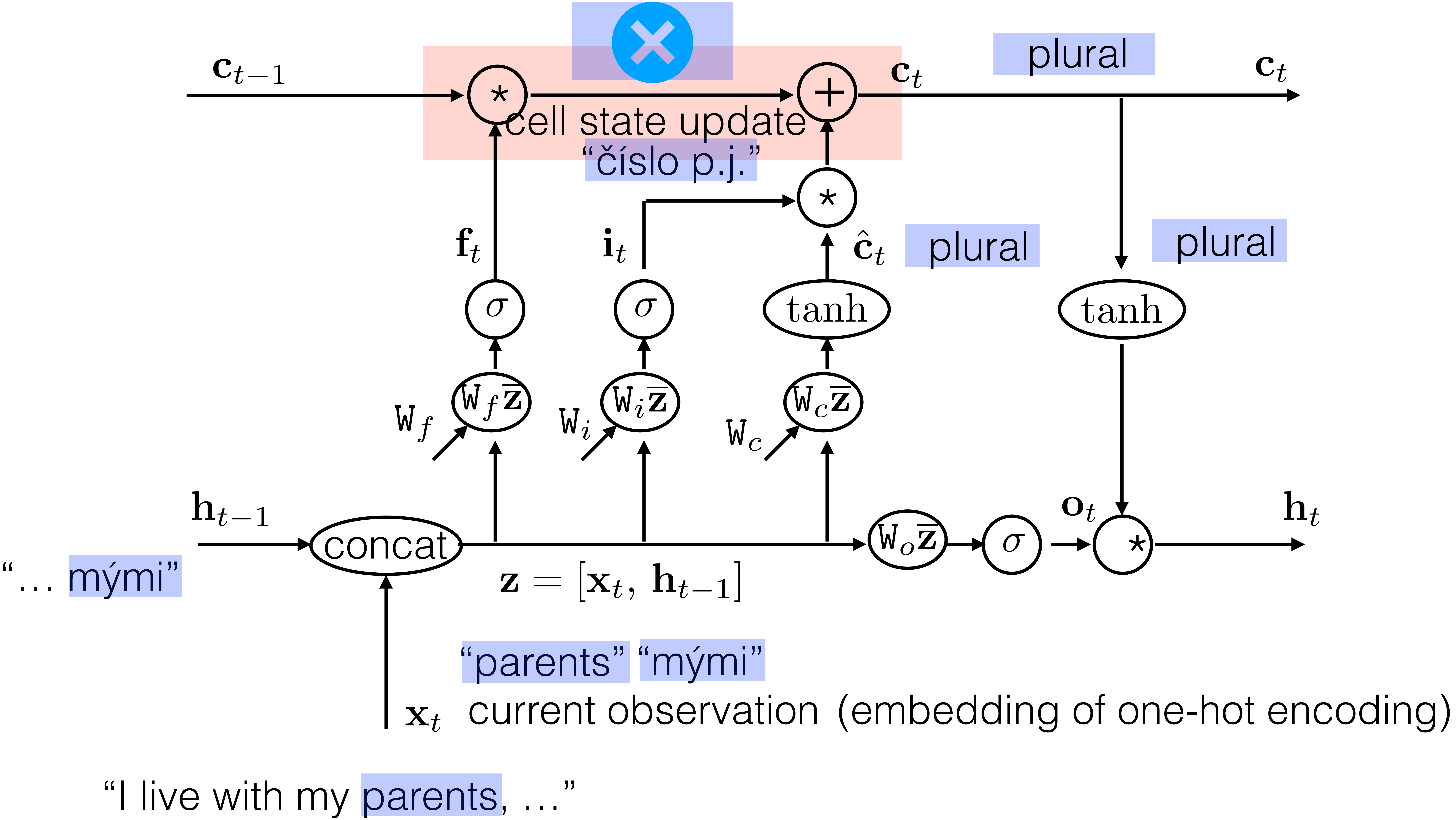

LSTM block
$\mathbf{c}_{t-1}$
$*$
$+$
$\mathbf{c}_t$
$\mathbf{c}_t$
"číslo p.j."
$*$
$\mathbf{f}_t$
$\mathbf{i}_t$
input
gate
$\hat{\mathbf{c}}_t$
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
concat
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
$\mathbf{o}_t$
$*$
$\mathbf{h}_t$
"… 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



“I live with my parents, …”

LSTM block



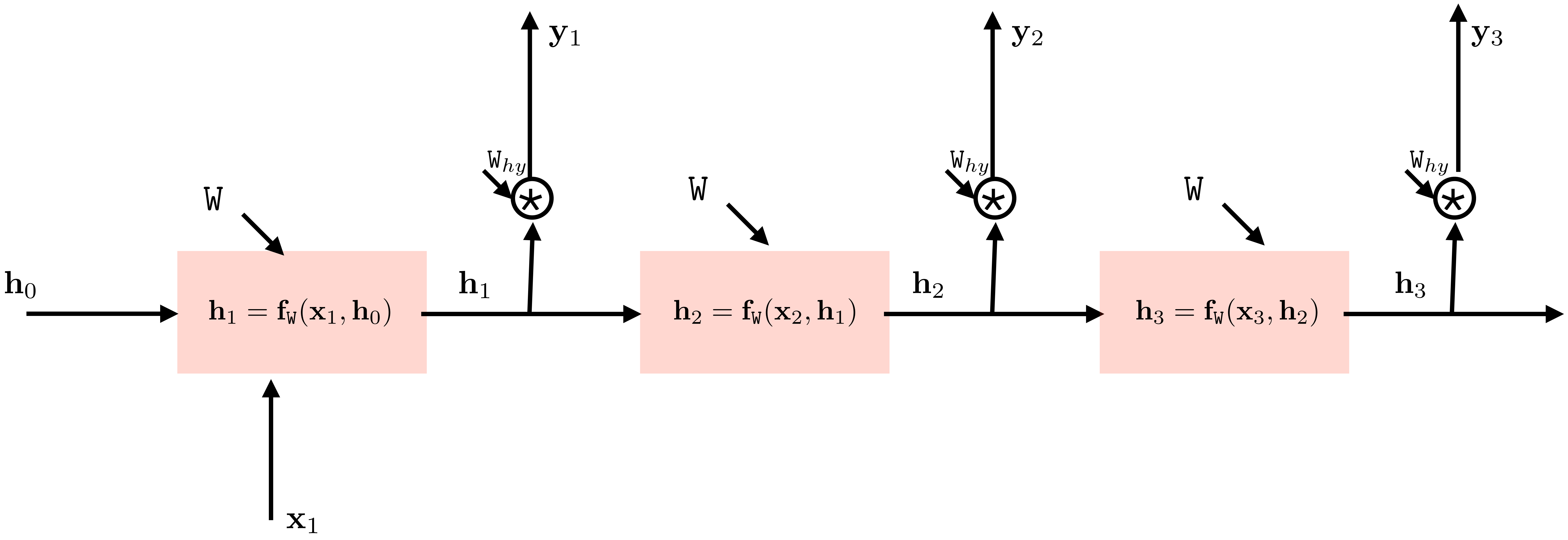
$\mathbf{c}_{t-1}$
cell state update
“číslo p.j.”
$\mathbf{c}_t$
plural
$\mathbf{c}_t$
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
plural
plural
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
$\mathbf{h}_{t-1}$
“… mými”
concat
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
$\mathbf{o}_t$
$\mathbf{h}_t$
$\mathbf{z} = [\mathbf{x}_t,\ \mathbf{h}_{t-1}]$
“parents” “mými”
$\mathbf{x}_t$ current observation (embedding of one-hot encoding)
“I live with my parents, …”

LSTM block
$\mathbf{c}_{t-1}$
$\mathbf{c}_t$
plural
$\mathbf{c}_t$
“číslo p.j.”
$\mathbf{f}_t$
$\mathbf{i}_t$
$\hat{\mathbf{c}}_t$
plural
plural
σ
σ
tanh
tanh
$\mathtt{W}_f\overline{\mathbf{z}}$
$\mathtt{W}_i\overline{\mathbf{z}}$
$\mathtt{W}_c\overline{\mathbf{z}}$
$\mathtt{W}_f$
$\mathtt{W}_i$
$\mathtt{W}_c$
output gate
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$
$\mathbf{h}_t$
concat
$\mathtt{W}_o\overline{\mathbf{z}}$
σ
“… mými”
$\mathbf{z} = [\mathbf{x}_t, \mathbf{h}_{t-1}]$
“… rodiči …”
“parents” “mými”
$\mathbf{x}_t$
current observation
(embedding of one-h
“I live with my parents, …”

LSTM block

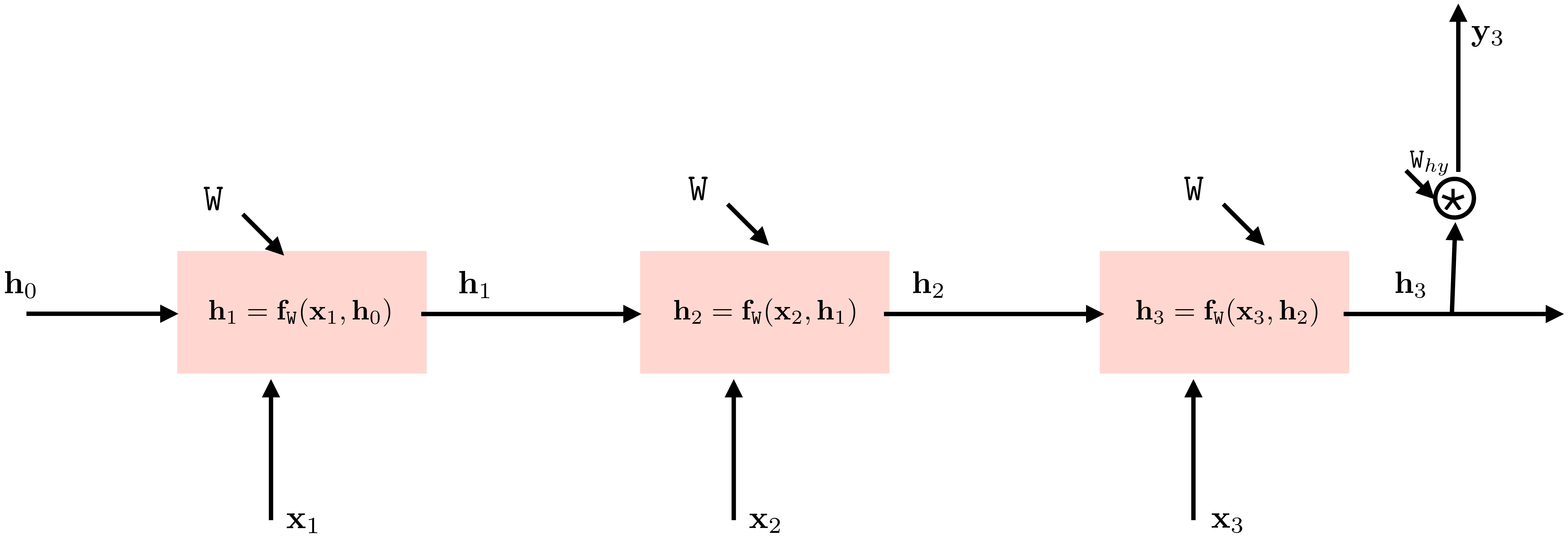
`torch.nn.LSTM(input_size, hidden_dim, n_layers)`

RNN architectures: one-to-many



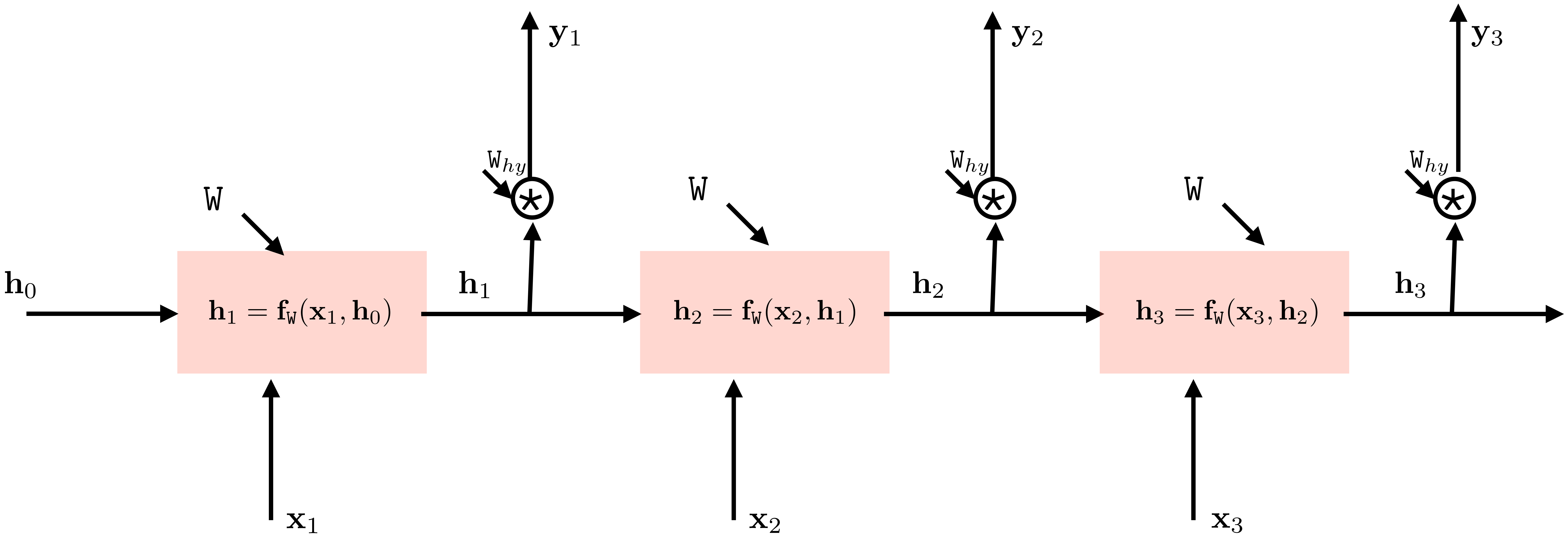
- Music generation
- Image captioning

RNN architectures: many-to-one



- Sentiment classification
- Action recognition

RNN architectures: many-to-many



- Named-entity recognition
- Speech recognition

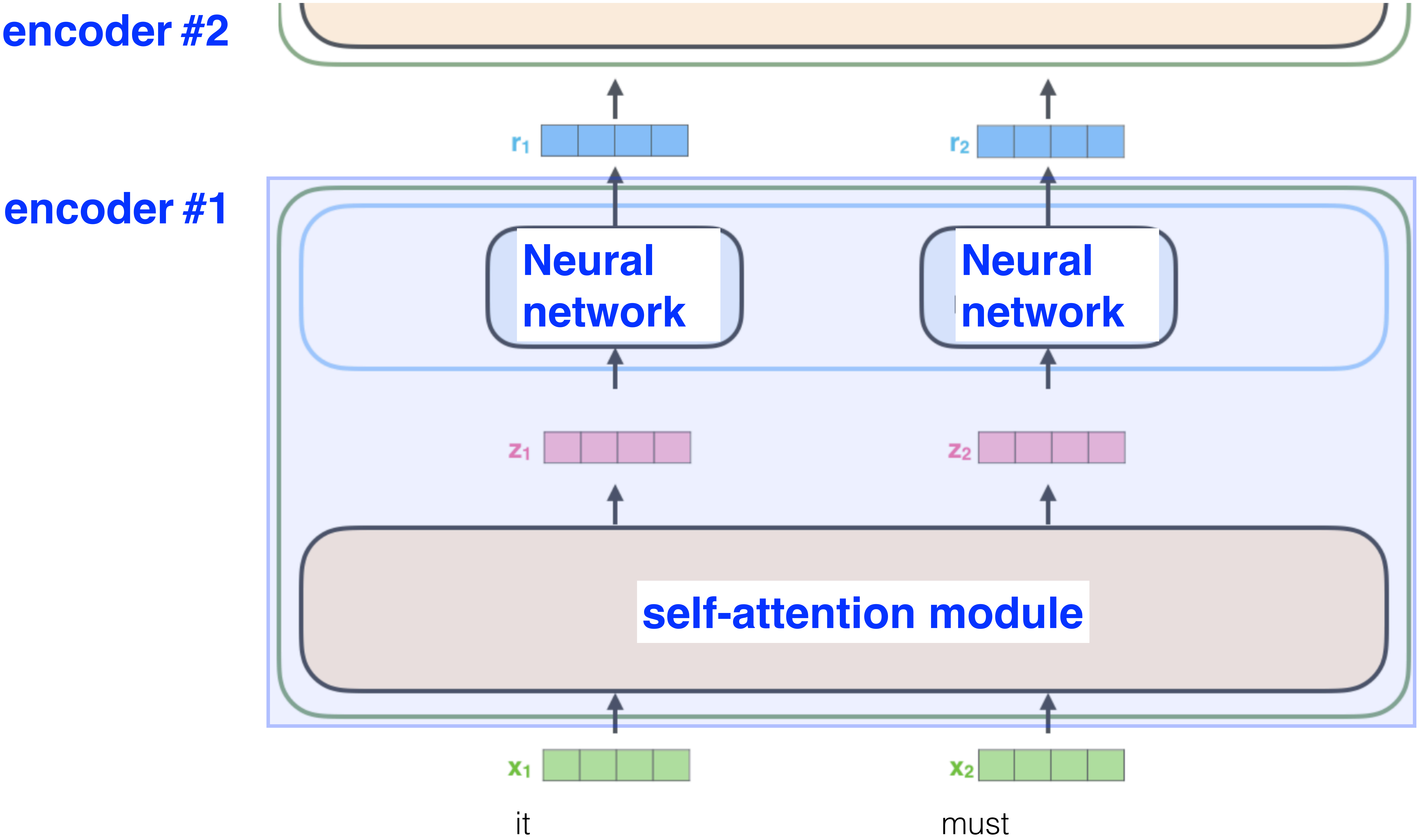
RNN architectures: many-to-many

- Machine translation
- Question answering

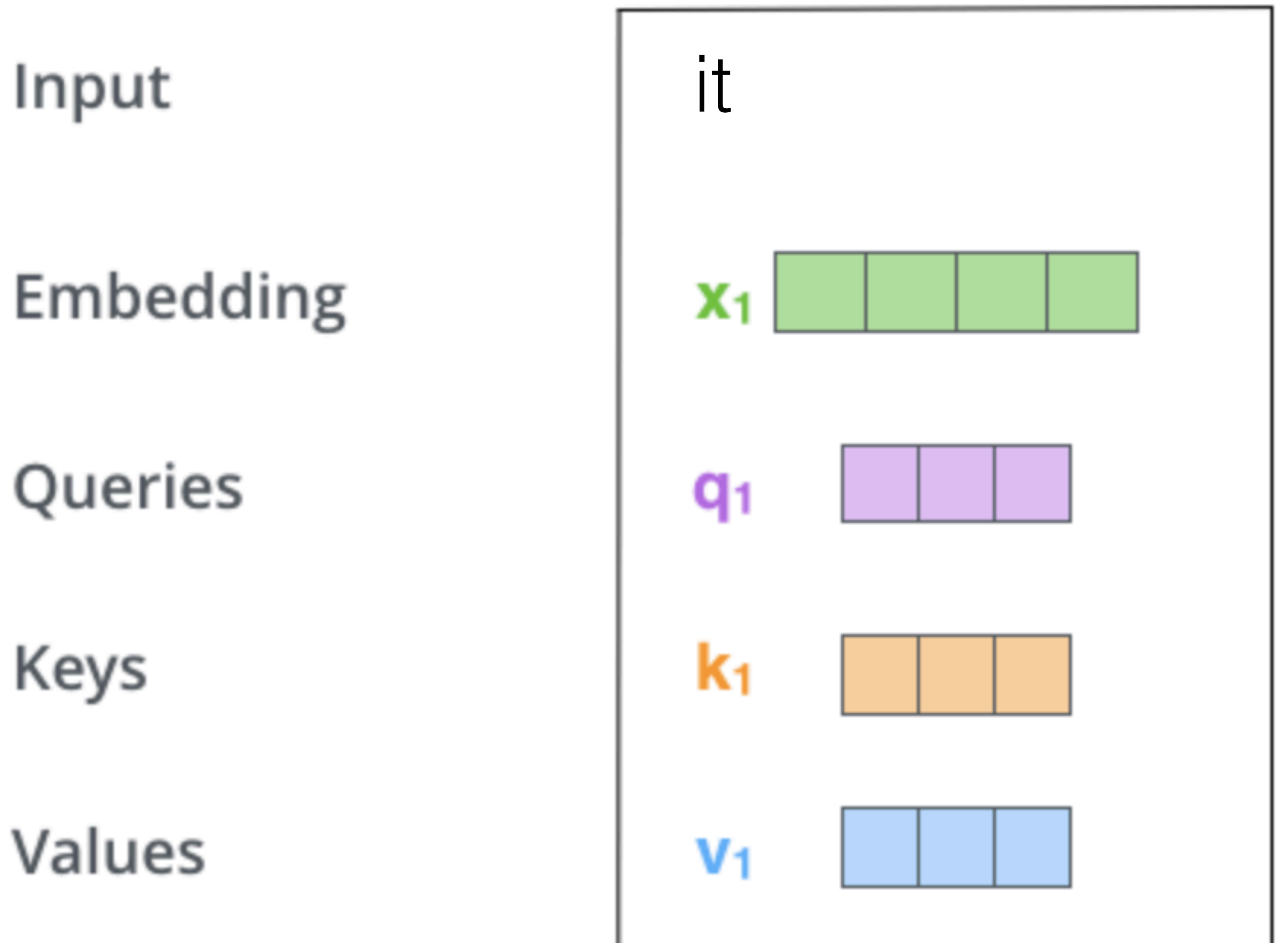
RNN architectures: many-to-many

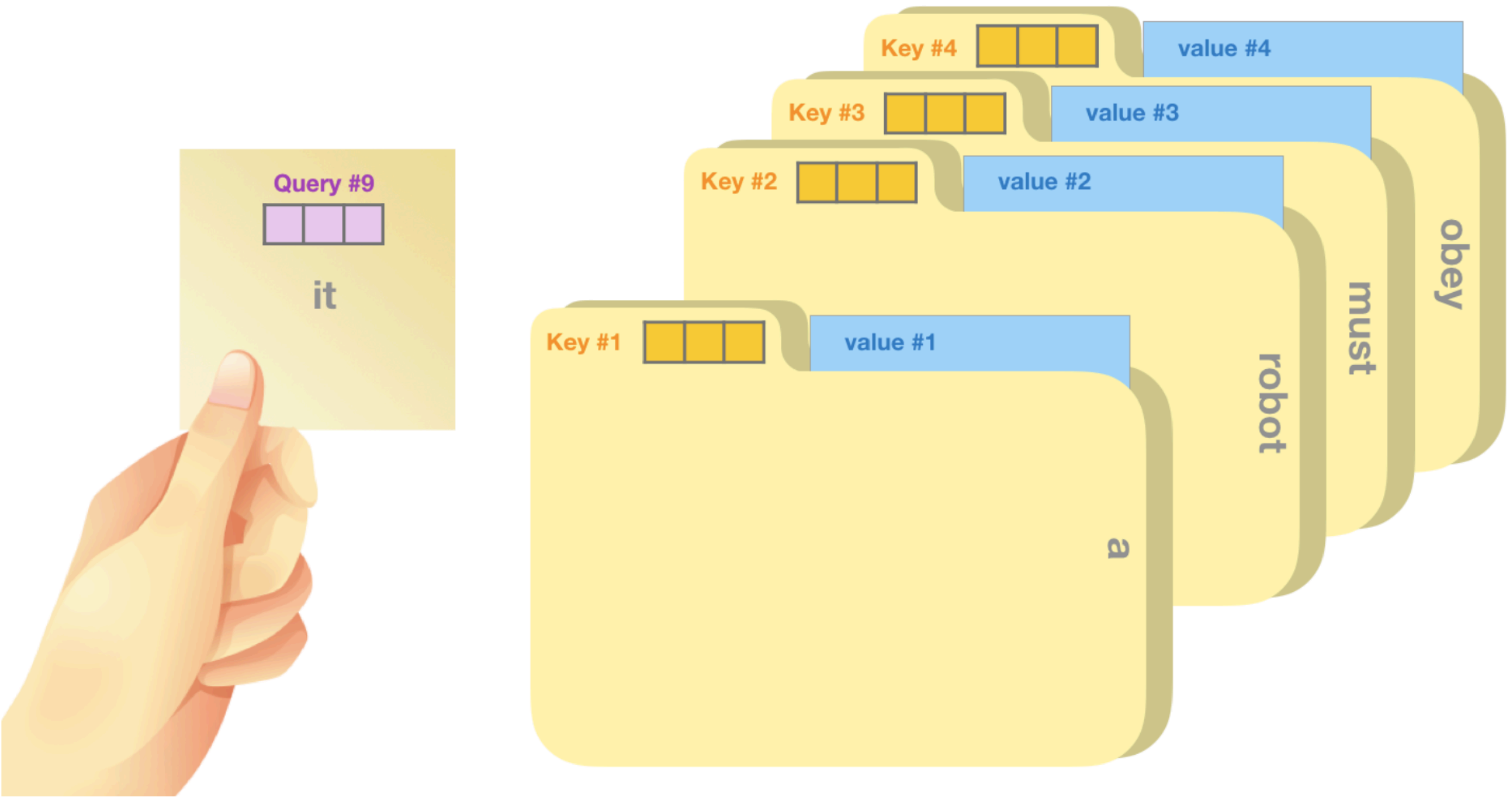
context: fixed-size semantic summary of input sequence

https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
[Waswani, NIPS 2018]

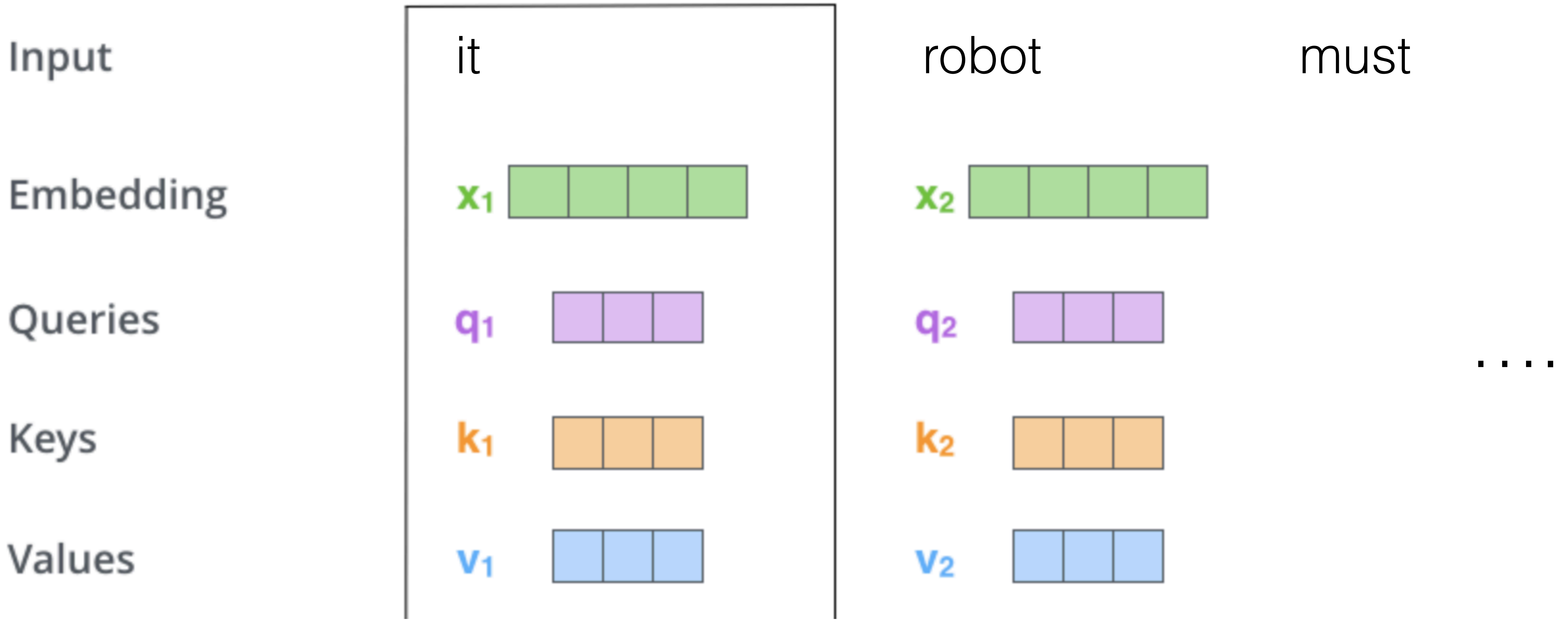


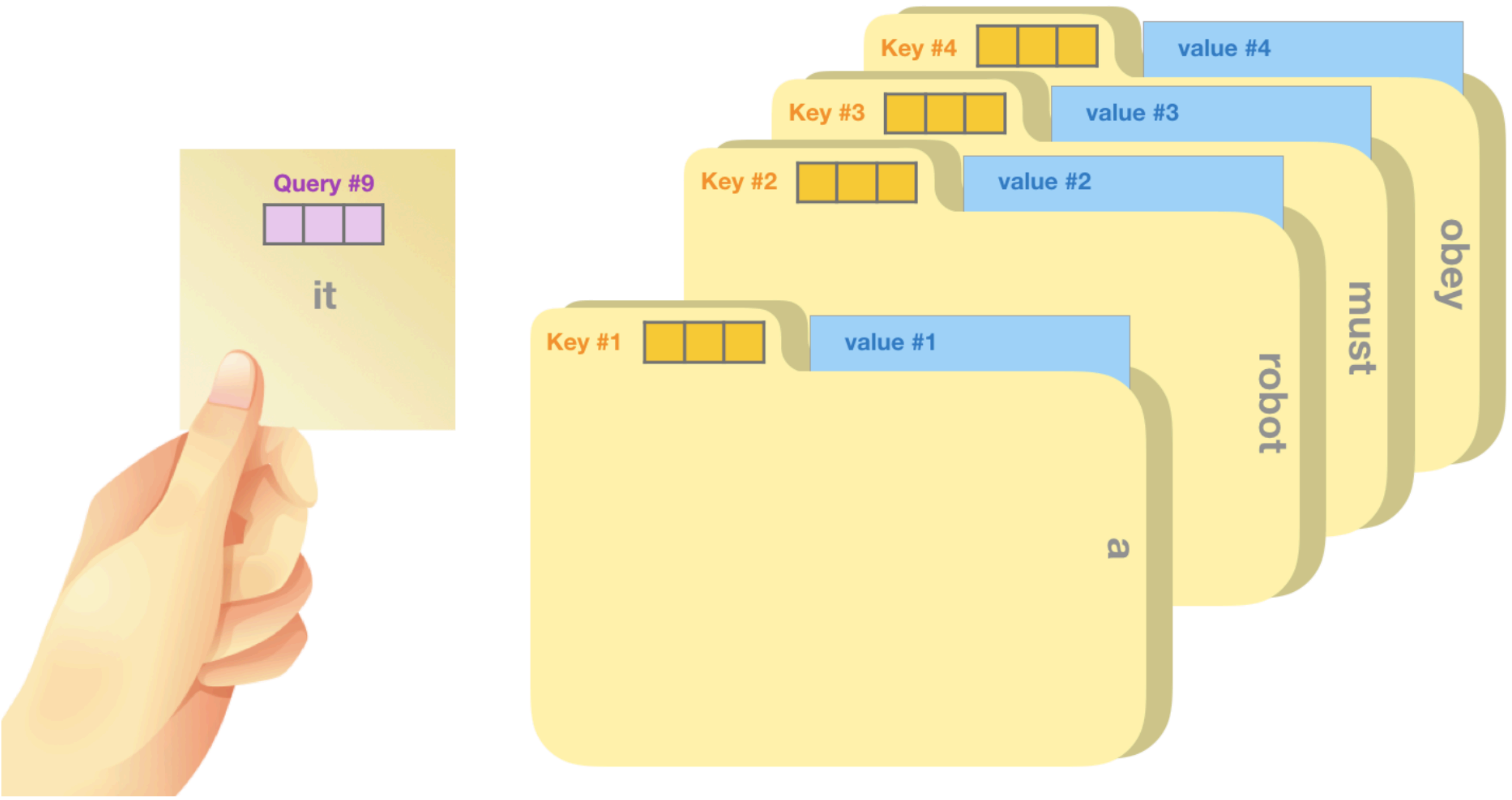
self-attention module





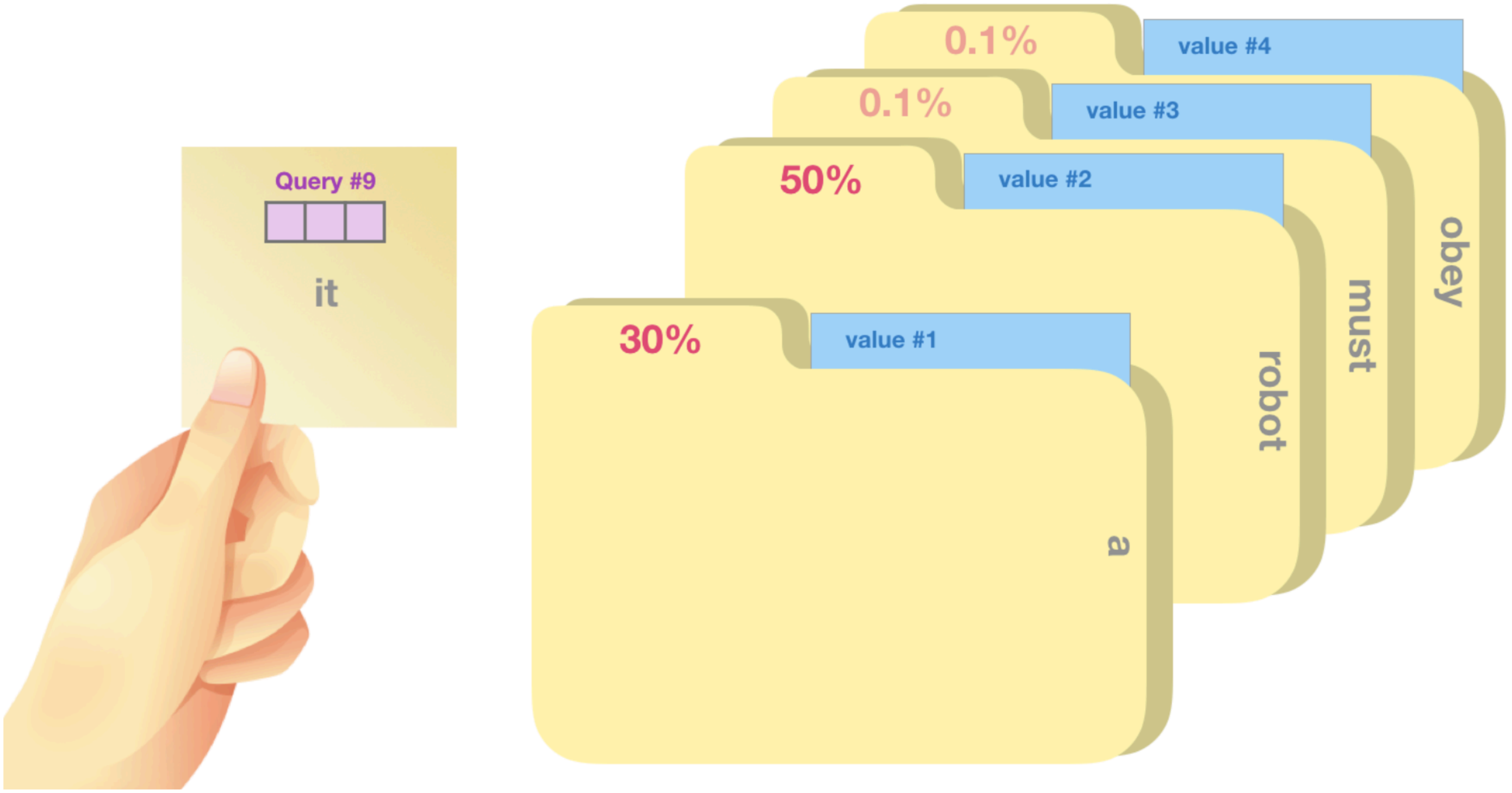
self-attention module



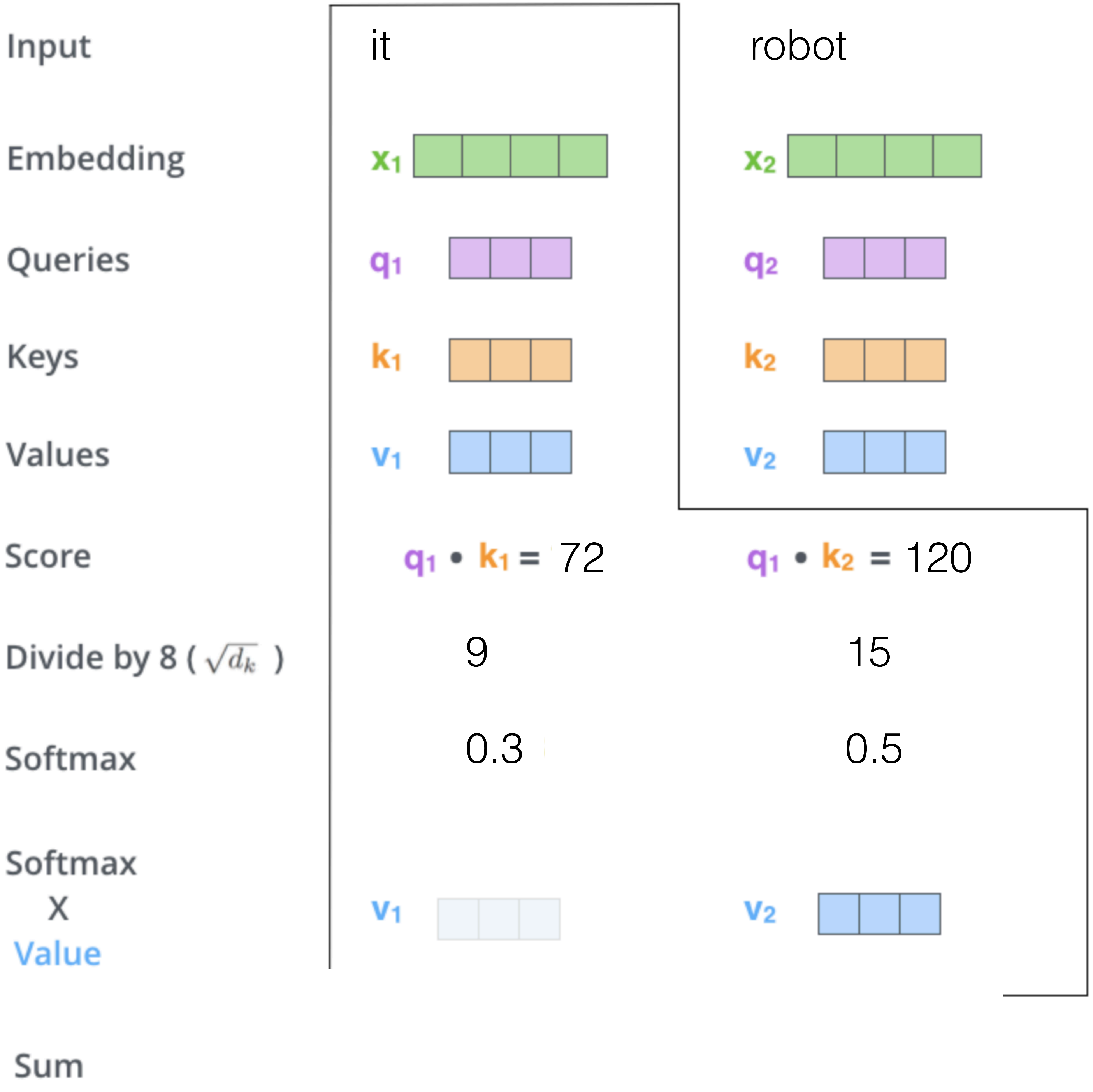


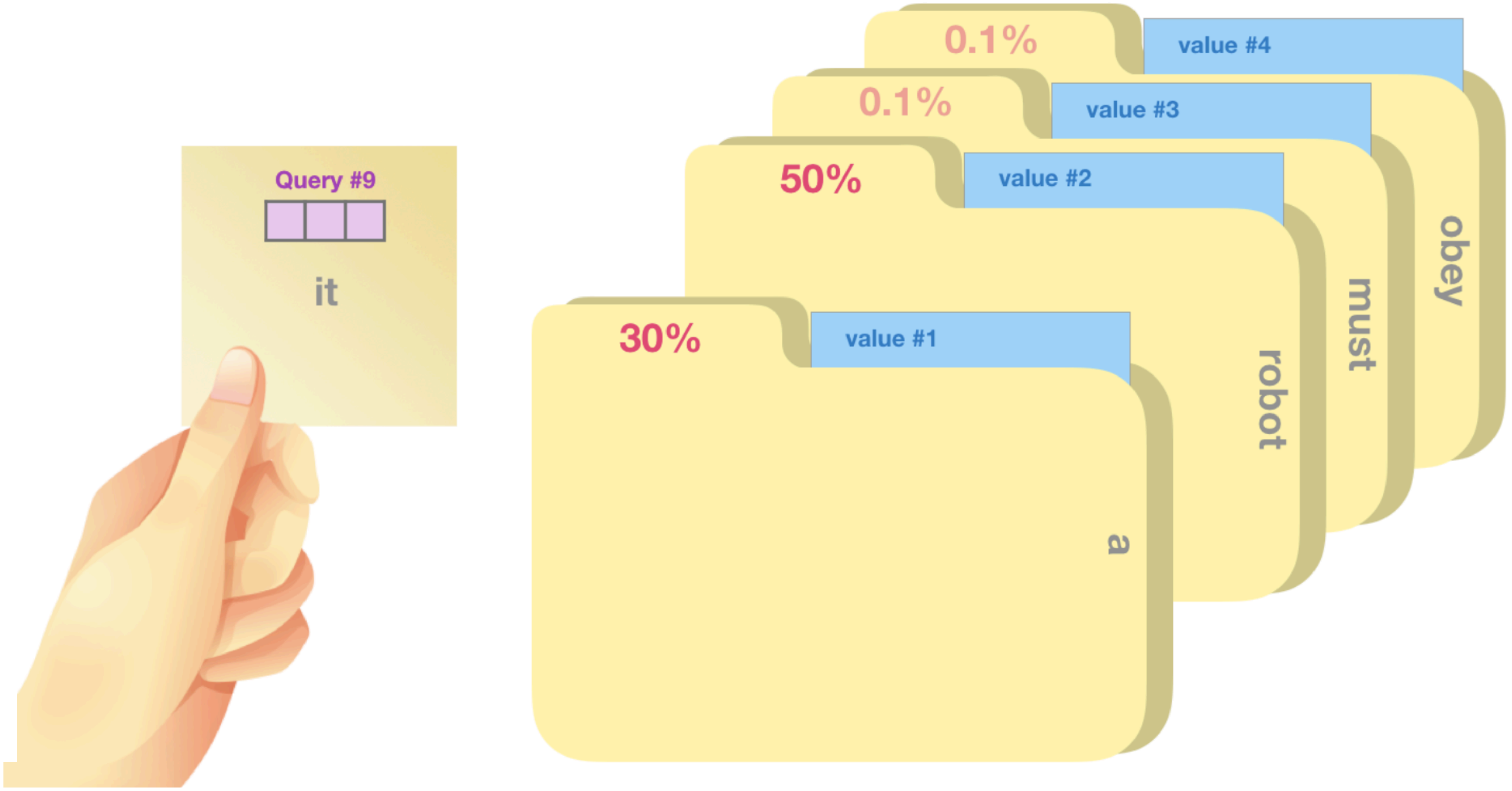
self-attention module

Input	it	robot
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 72$	$q_1 \bullet k_2 = 120$
Divide by 8 ($\sqrt{d_k}$)	9	15
Softmax	0.3	0.5



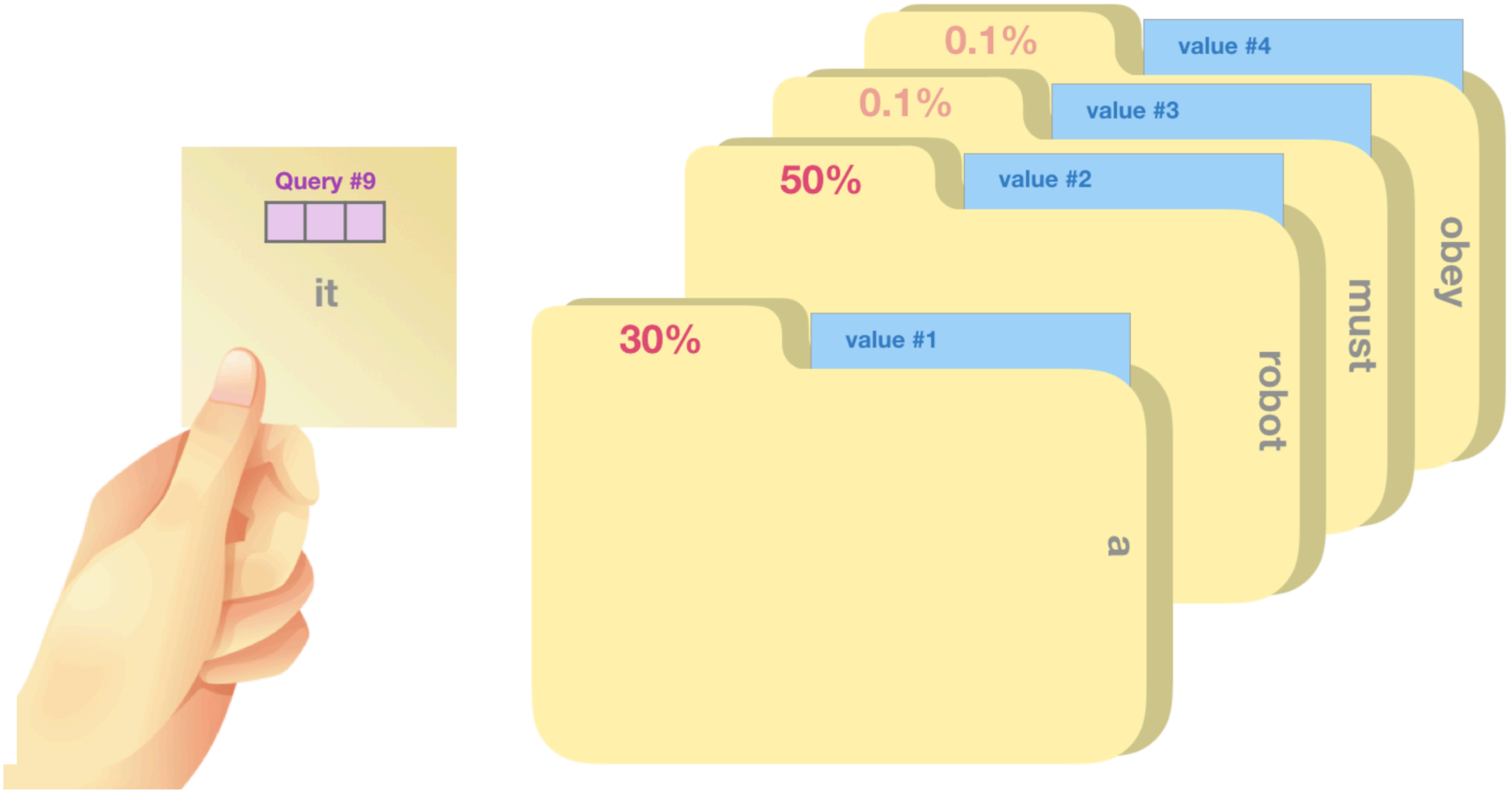
self-attention module



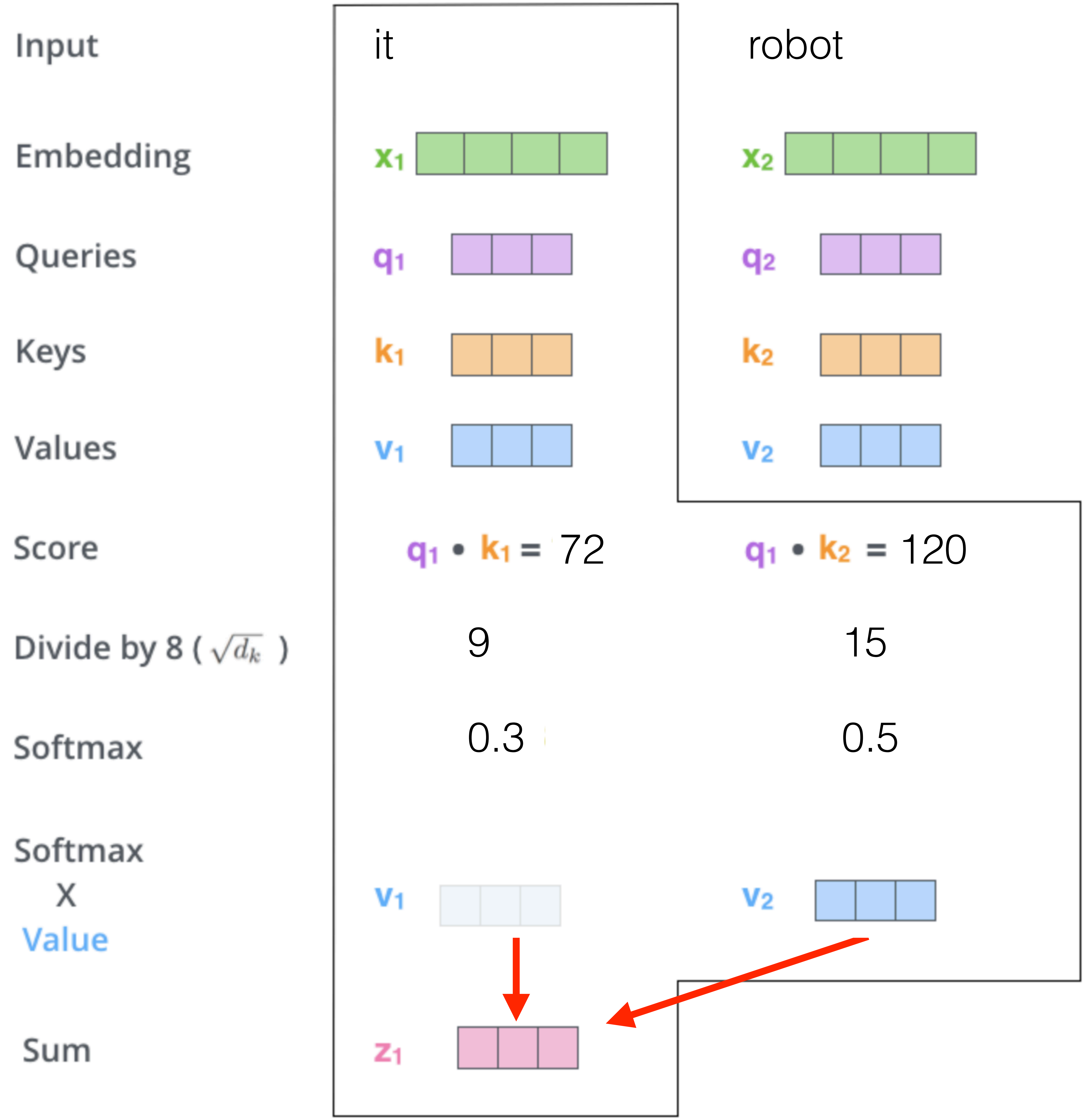


self-attention module

Input	it	robot
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 72$	$q_1 \bullet k_2 = 120$
Divide by 8 ($\sqrt{d_k}$)	9	15
Softmax	0.3	0.5
Softmax X Value	v_1	v_2
Sum	z_1	



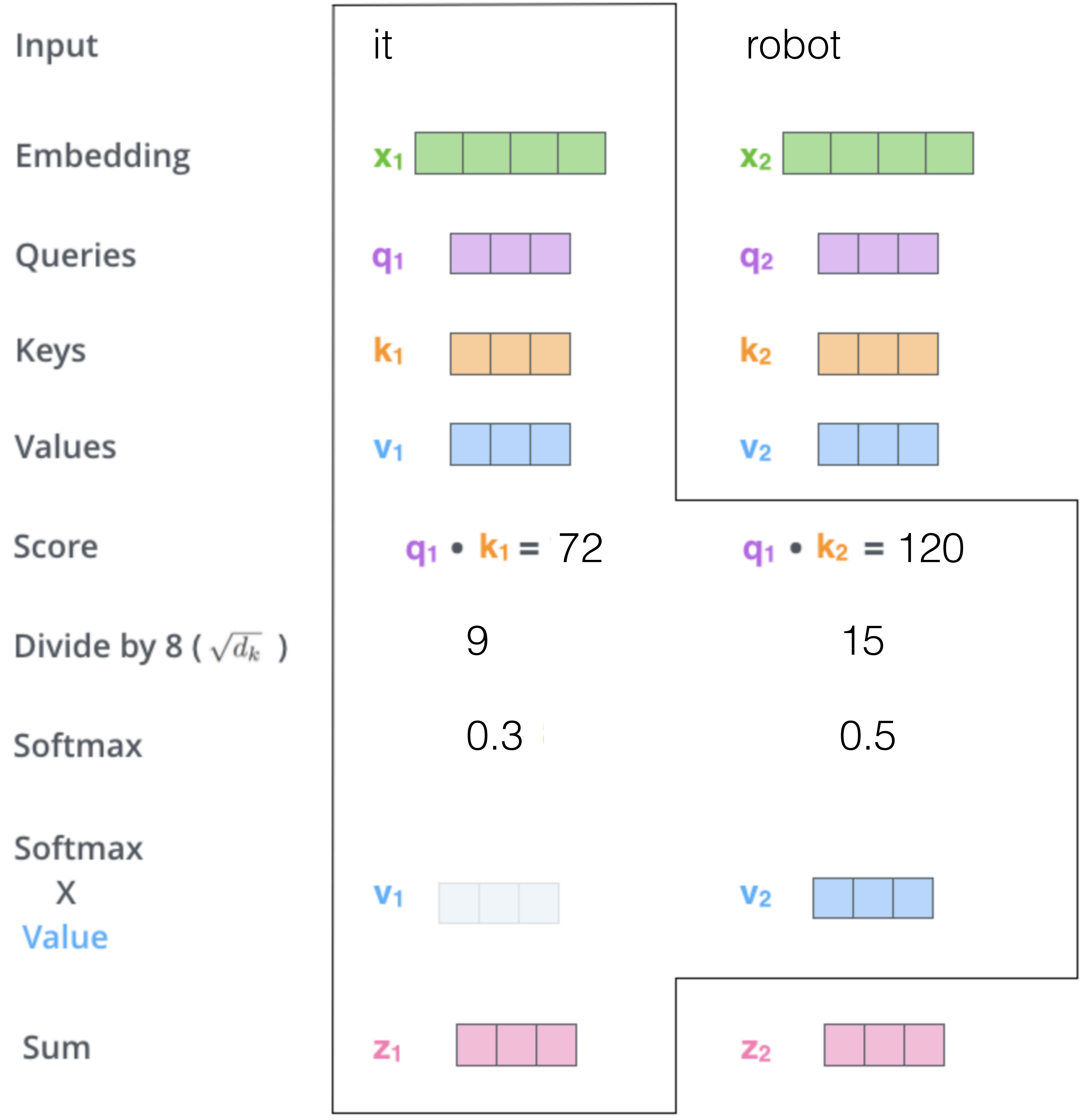
self-attention module

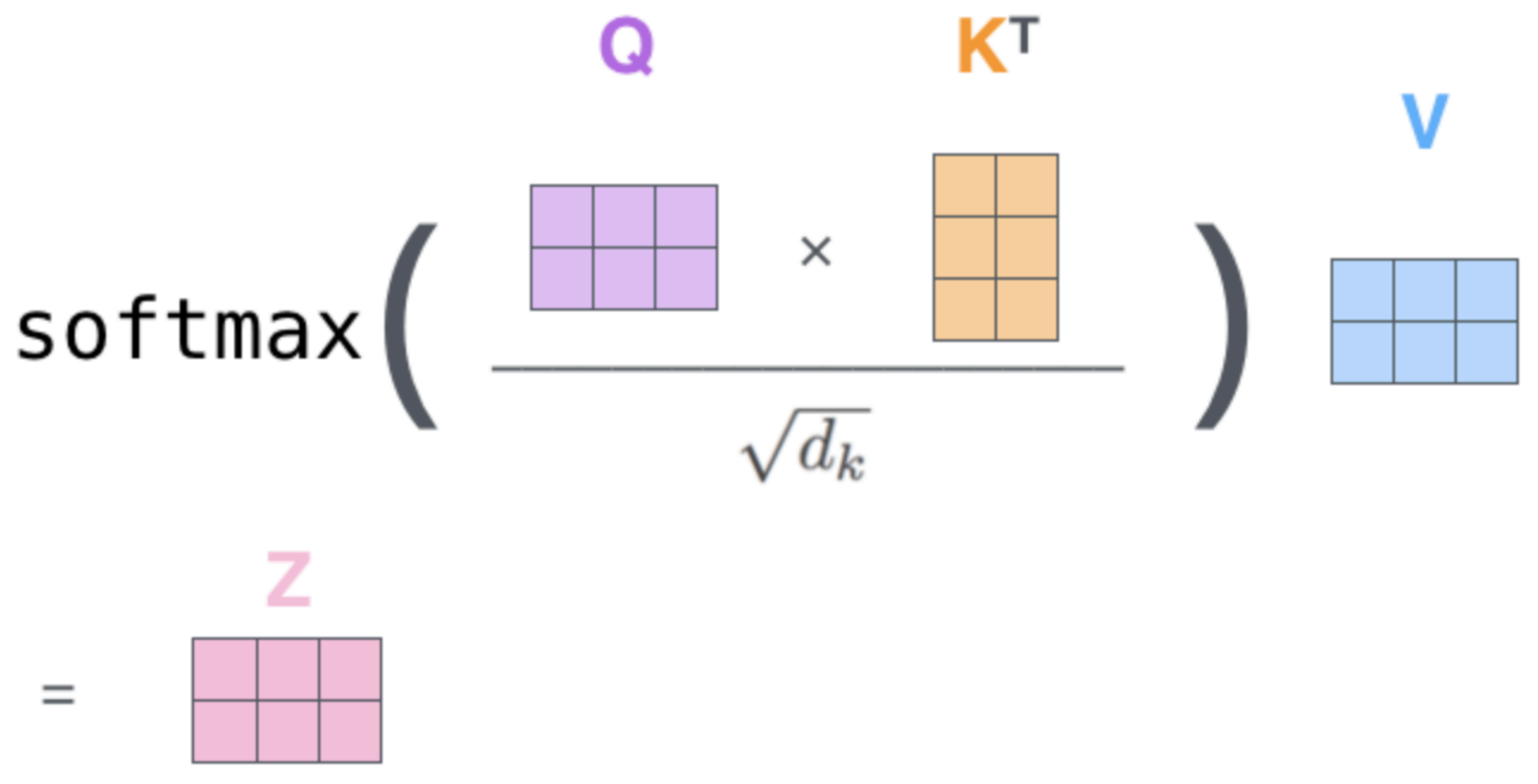


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

output “z” is weighted

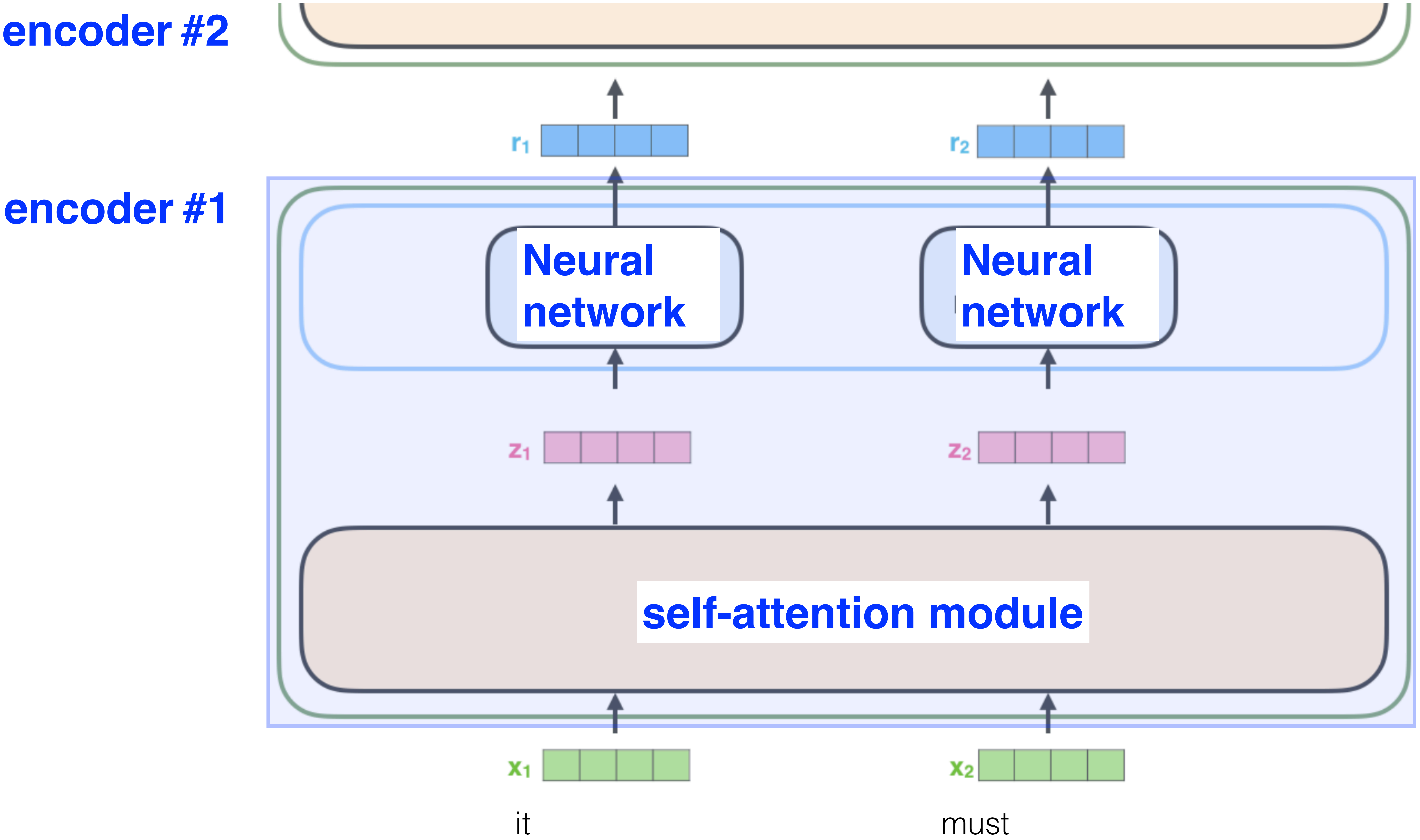
self-attention module



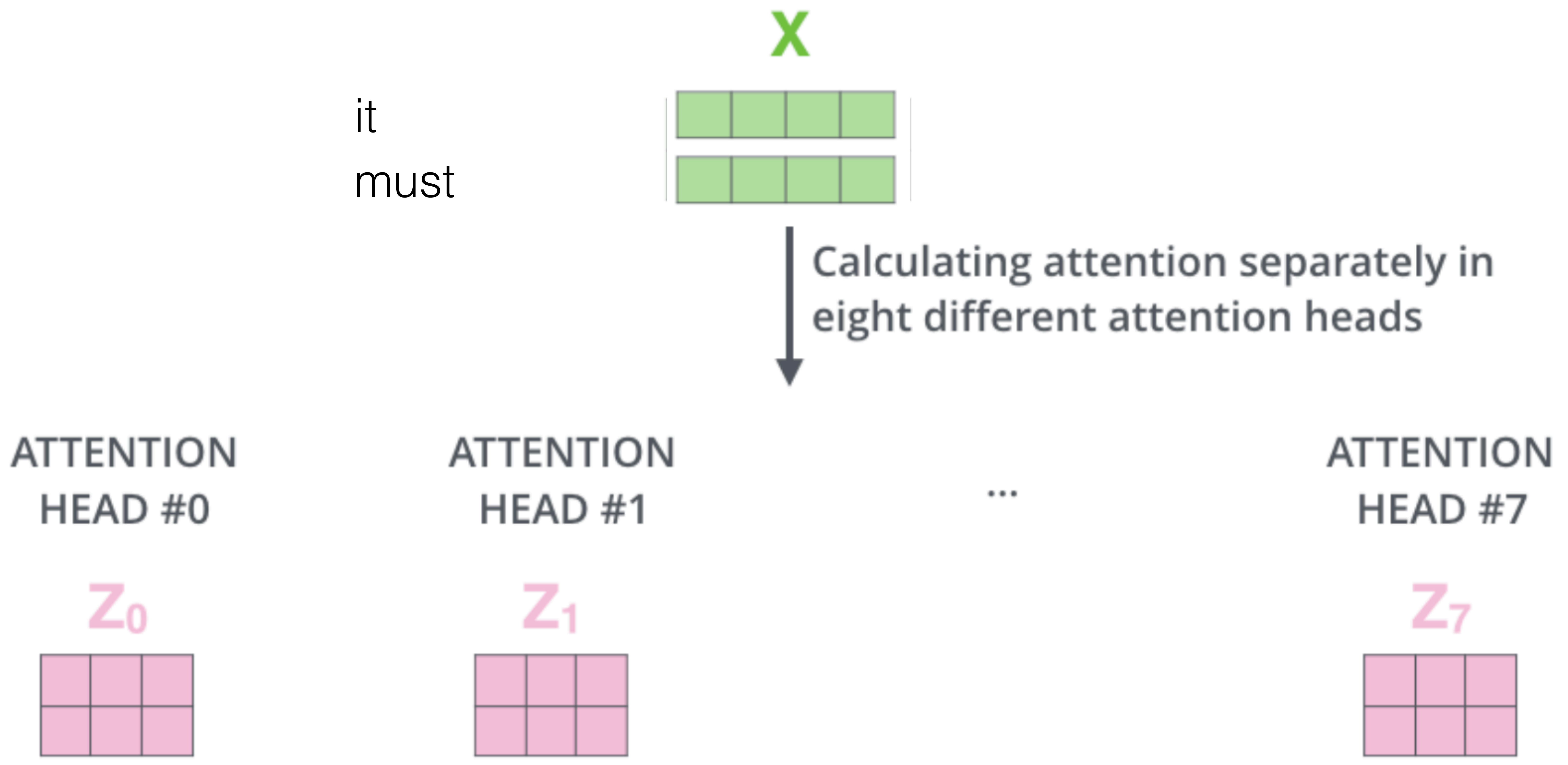


Proceed similarly for every single word.

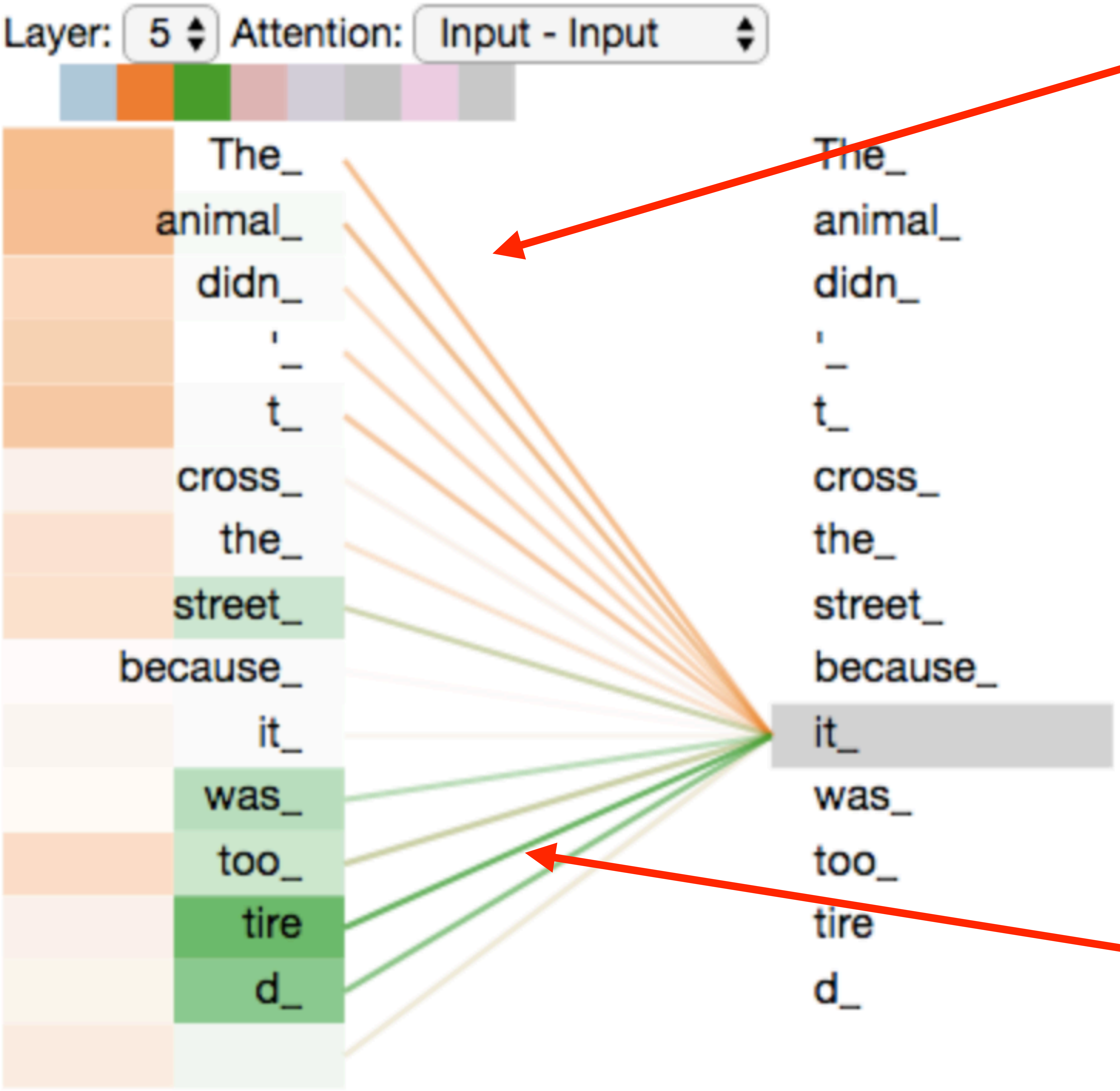
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
[Waswani, NIPS 2018]



encoder


X
it
must
Calculating attention separately in eight different attention heads
ATTENTION HEAD #0
ATTENTION HEAD #1
...
ATTENTION HEAD #7
Z_0
Z_1
Z_7

Tensor2tensor:

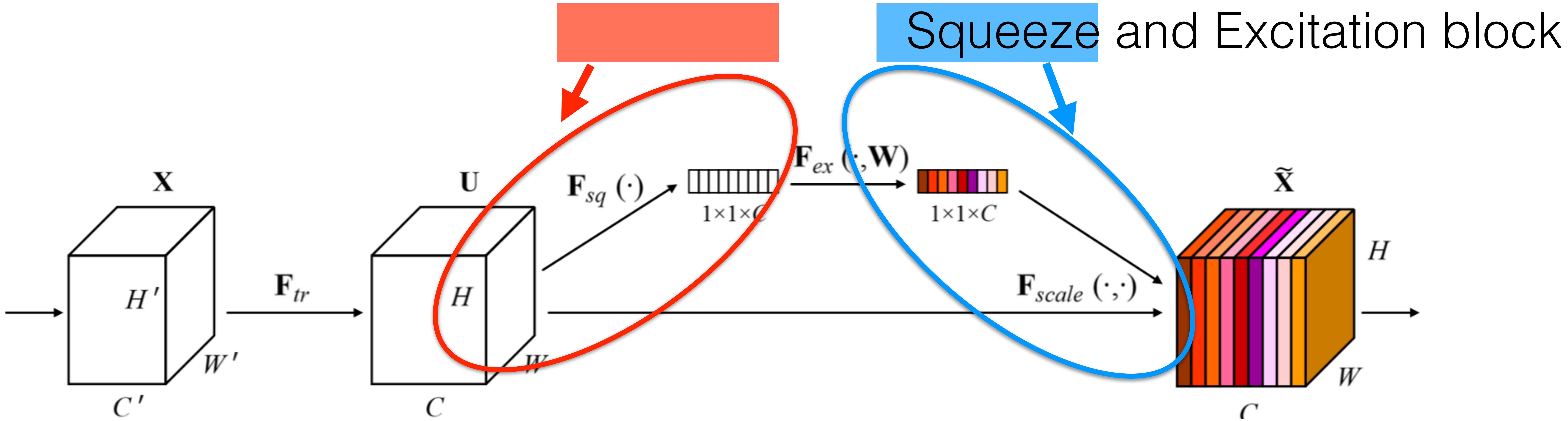


attention weights of **orange** attention head

attention weights of **green** attention head

https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

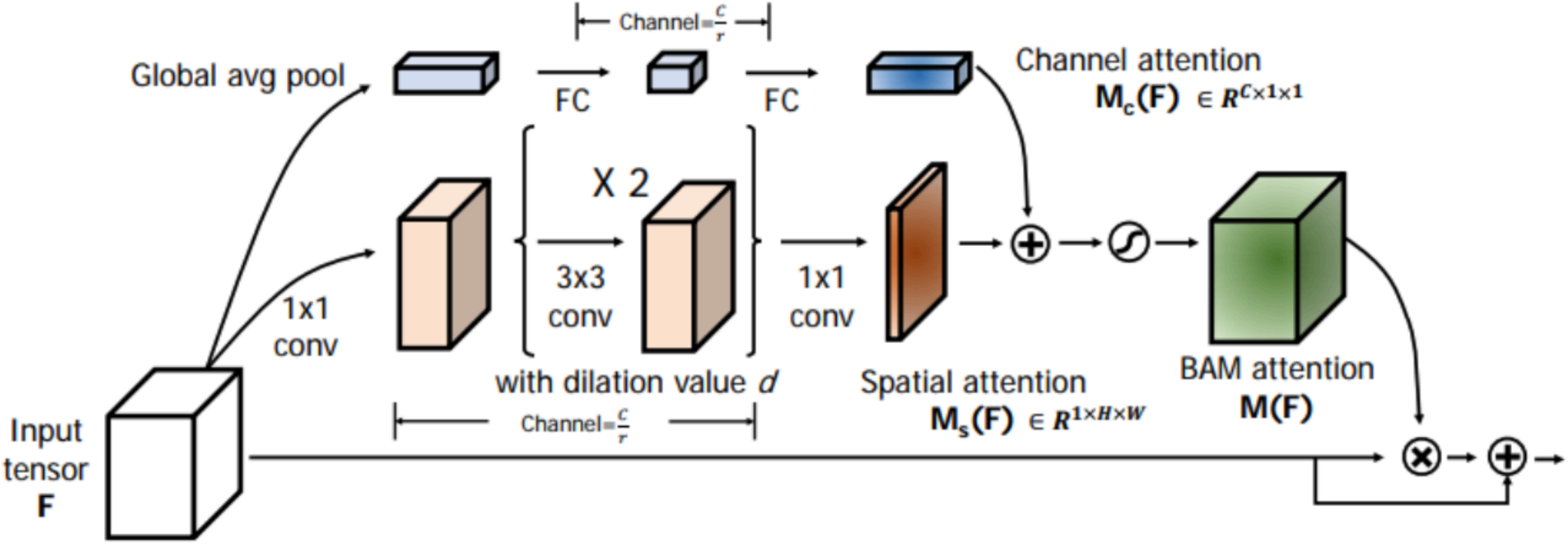
Attention in images
Channel attention



- Enhancement of ResNet, InceptionNet and DenseNet architectures by SE blocks consistently decrease error on ImageNet, COCO, …

Squeeze and Excitation Networks [Hu et al, CVPR oral, 2017]
https://arxiv.org/pdf/1709.01507.pdf

Attention in images
Channel+spatial attention

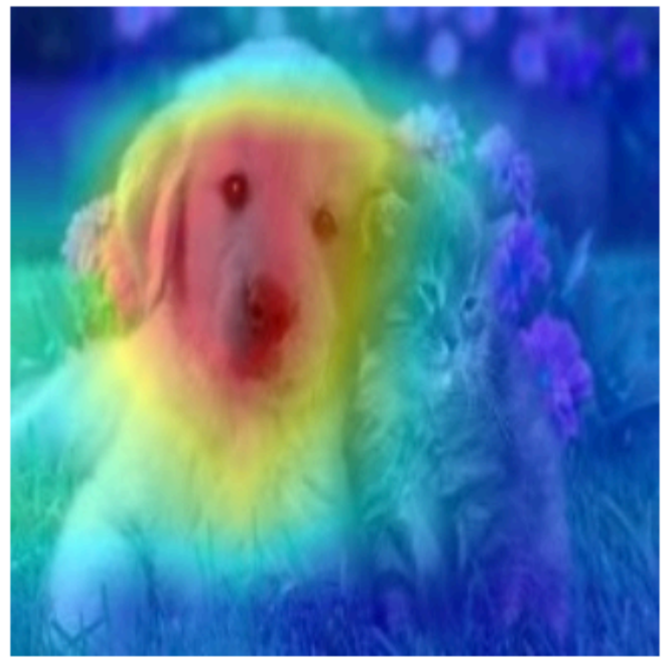


$$\mathbf{M}(\mathbf{F}) = \sigma(\mathbf{M_c}(\mathbf{F}) + \mathbf{M_s}(\mathbf{F})),$$

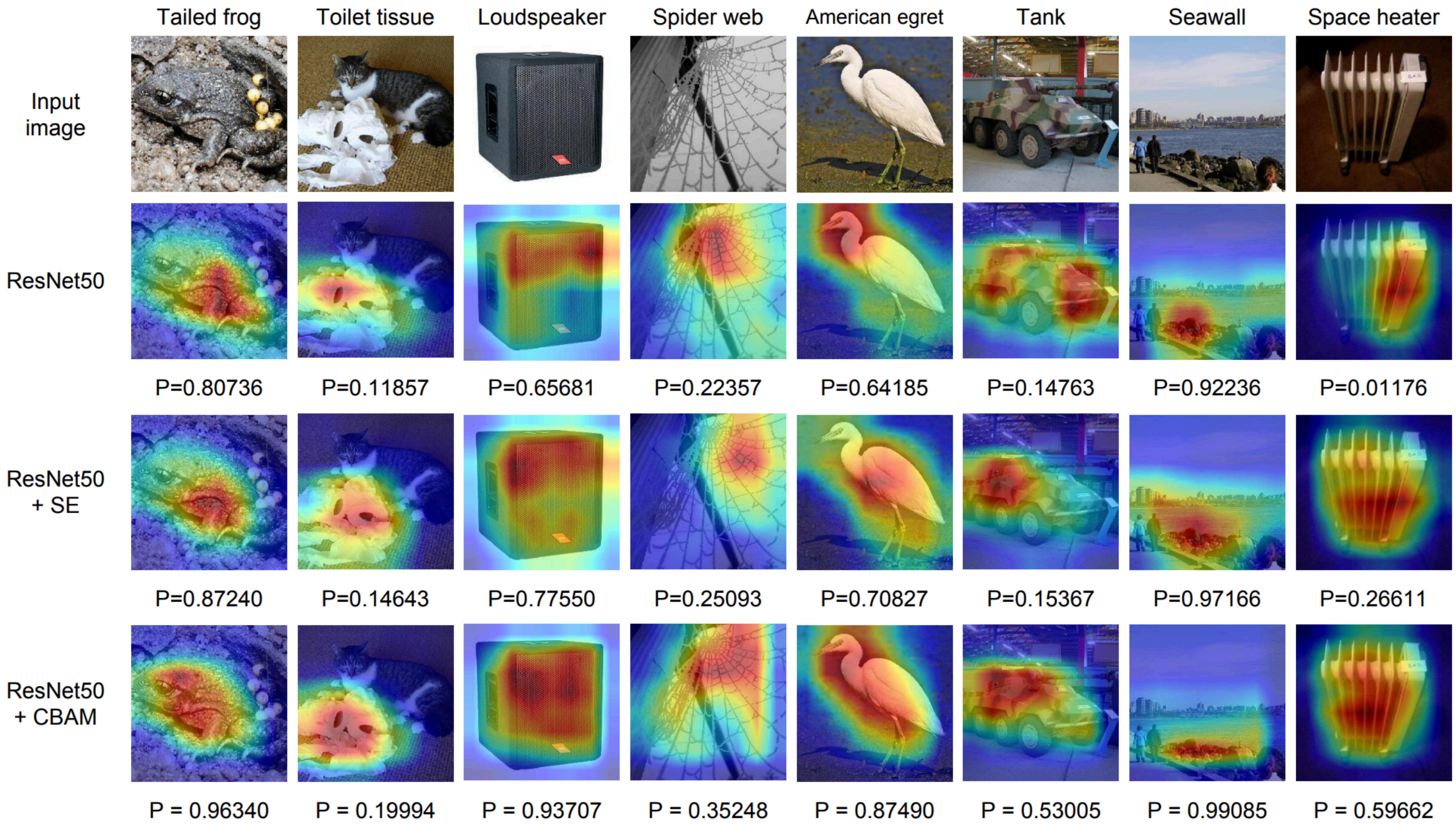
Attention modules [Woo et al,ECCV, 2018
https://arxiv.org/pdf/1807.06521v2.pdf

GradCAM [Selvaraju et al,ICCV, 2018]

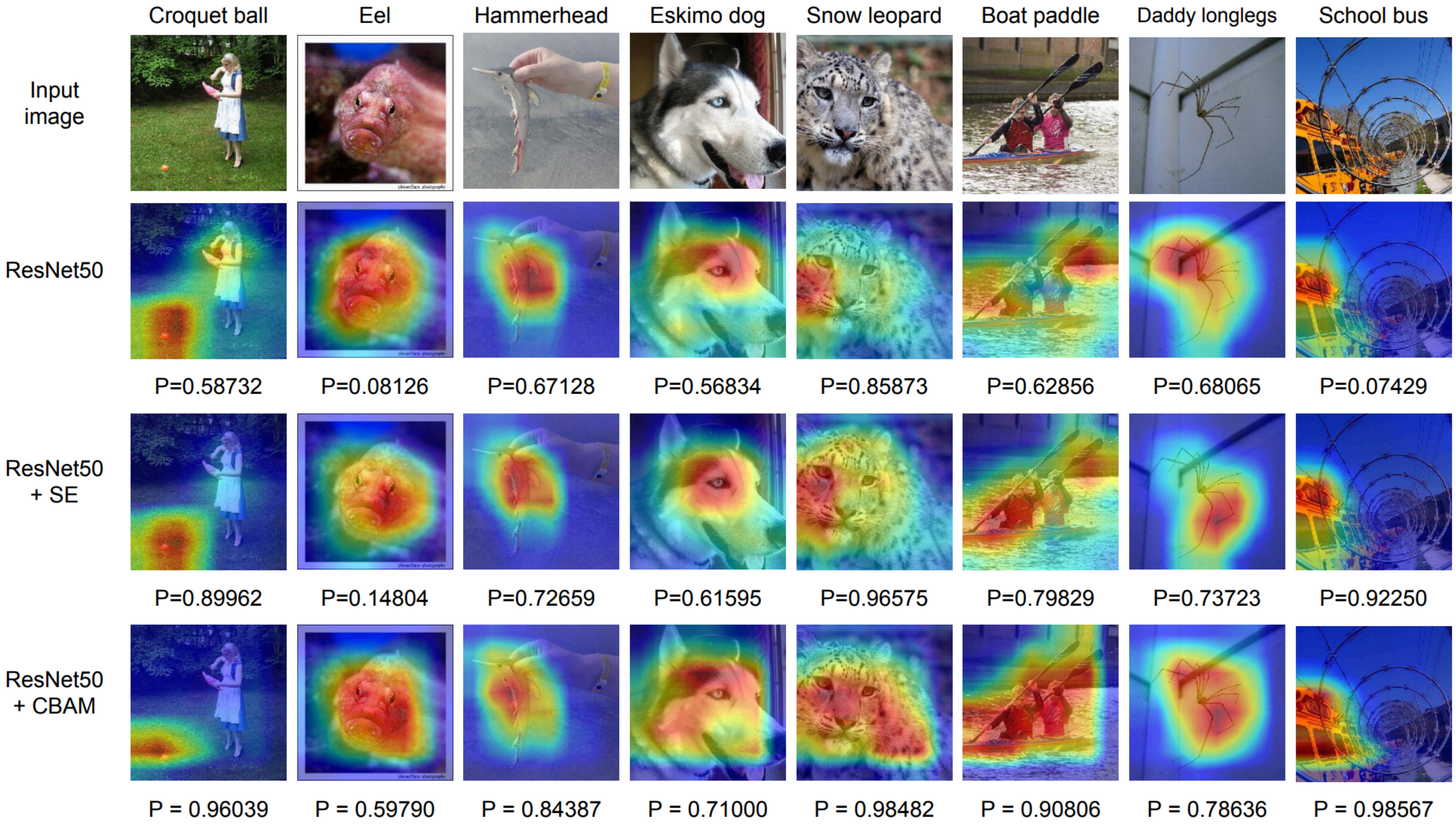
https://github.com/jacobgil/pytorch-grad-cam

Category	Image	GradCAM	AblationCAM	ScoreCAM
Dog				
Cat				

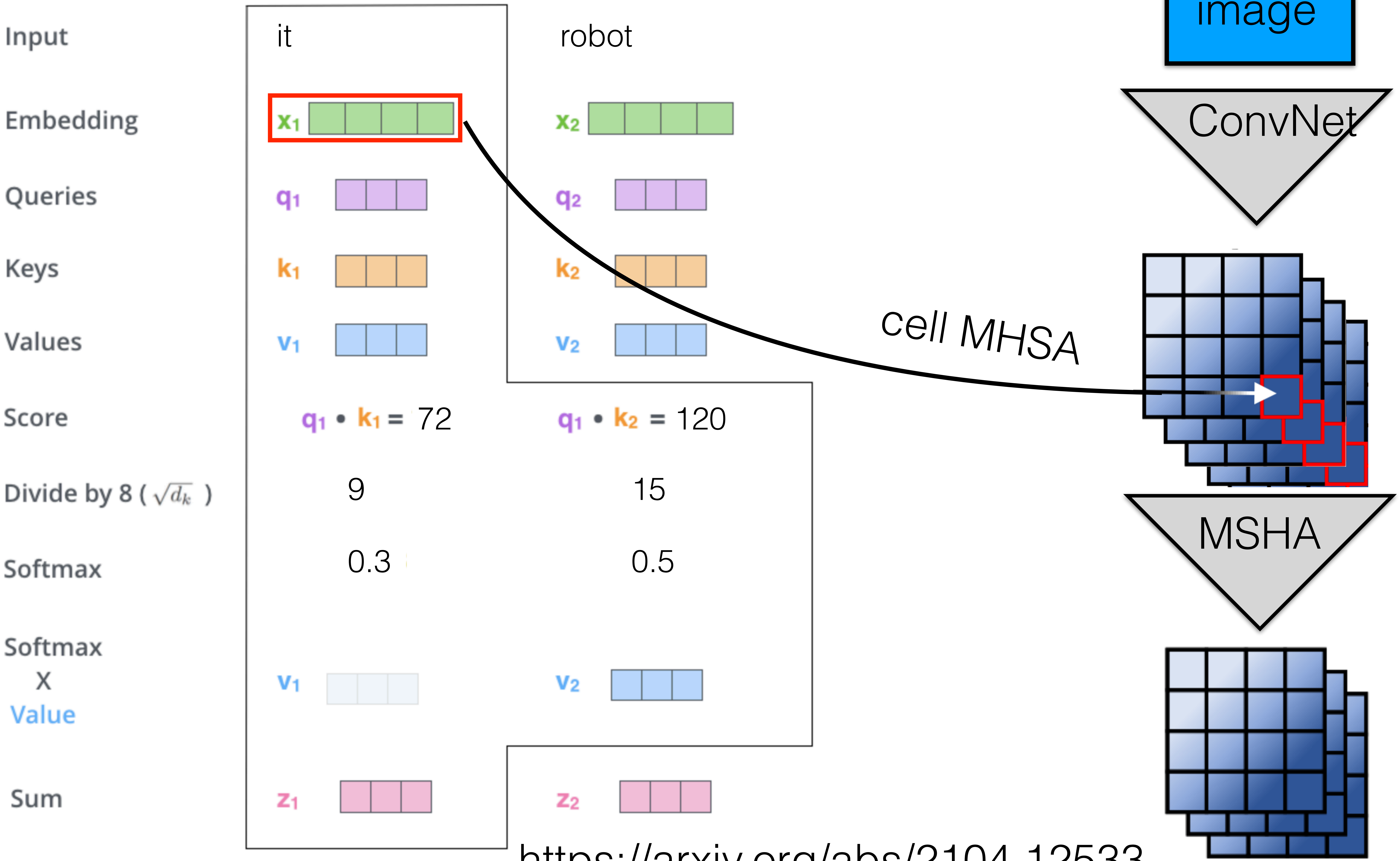
Attention modules [Woo et al,ECCV, 2018]
https://arxiv.org/pdf/1807.06521v2.pdf



Attention modules [Woo et al,ECCV, 2018]
https://arxiv.org/pdf/1807.06521v2.pdf

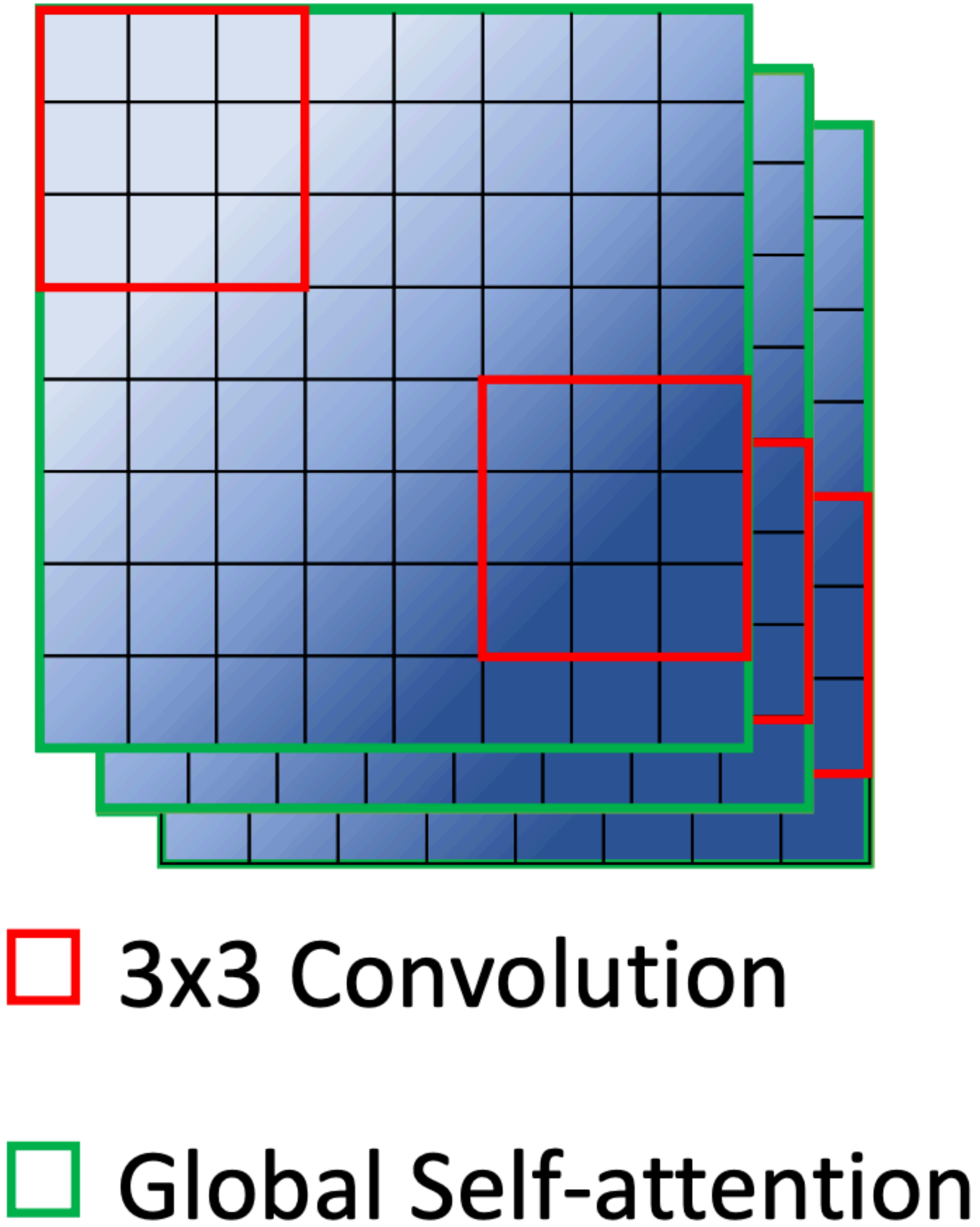


Visformer



https://arxiv.org/abs/2104.12533

Visformer



https://arxiv.org/abs/2104.12533

Visformer

Network		ResNet-50
FLOPs (G)		4.1
Parameters (M)		25.6
Full data	base setting	77.43
	elite setting	78.73

https://arxiv.org/abs/2104.12533

Visformer

Network		ResNet-50
FLOPs (G)		4.1
Parameters (M)		25.6
Full data	base setting	77.43
	elite setting	78.73

ResNet has no significant benefit from introducing 20M augmentations due to its limited expressing power

https://arxiv.org/abs/2104.12533

Visformer

Network		ResNet-50	Visformer-S
FLOPs (G)		4.1	4.9
Parameters (M)		25.6	40.2
Full data	base setting	77.43	77.20
	elite setting	78.73	82.19

Visformer (ResNet with MHSA) exhibit improvement in elite setting

https://arxiv.org/abs/2104.12533

Visformer

Network		ResNet-50	Visformer-S	DeiT-S
FLOPs (G)		4.1	4.9	4.6
Parameters (M)		25.6	40.2	21.8
Full data	base setting	77.43	77.20	63.12
	elite setting	78.73	82.19	80.07

Direct usage of “language transformers” without convolutions is more prone to overfitting

https://arxiv.org/abs/2104.12533

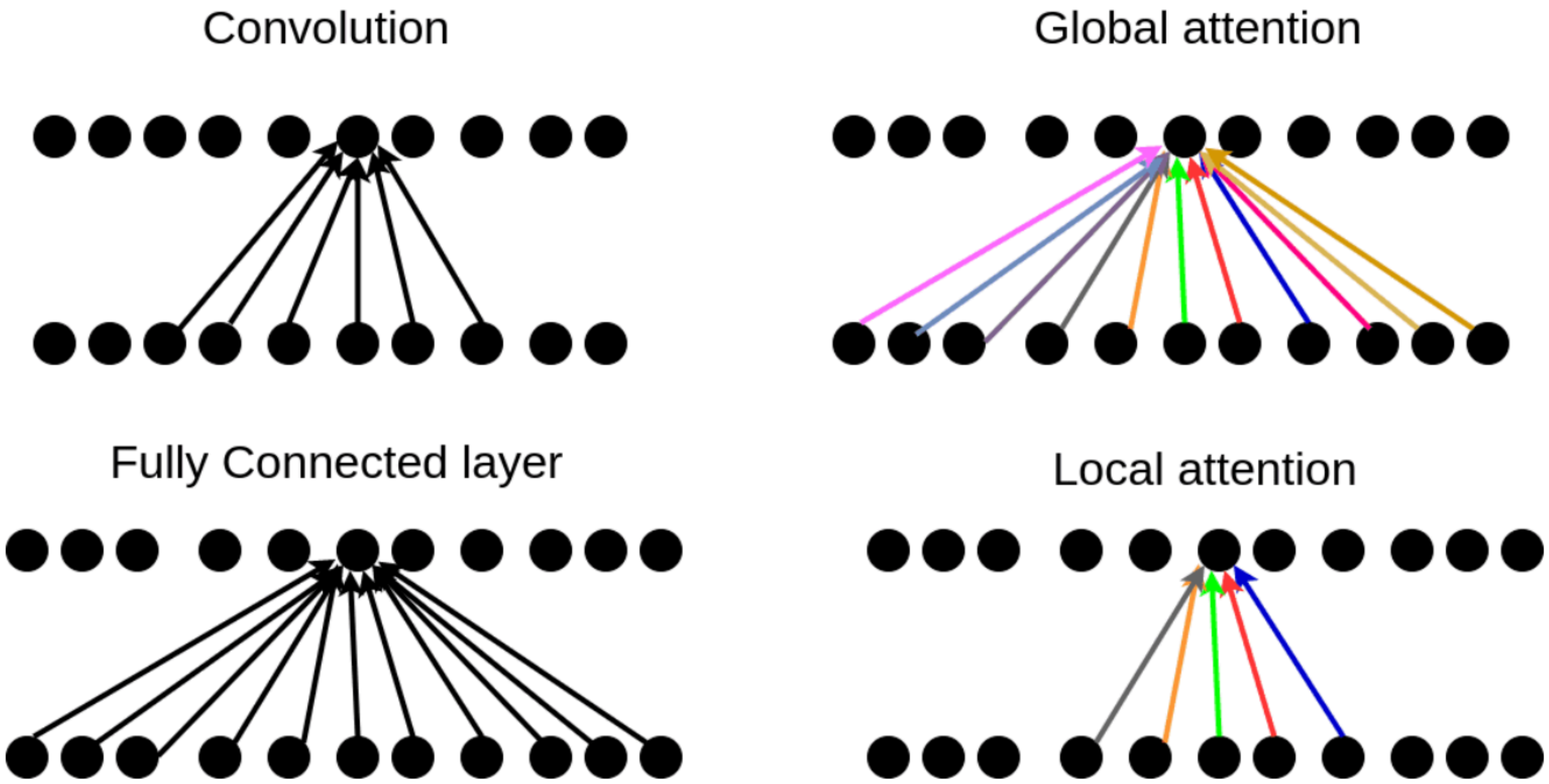
Attention in RL

Summary

- self-attention overfits (requires large dataset)

- memory is attention through time [Alex Graves 2020]
- pyTorch library: https://github.com/The-AI-Summer/self-attention-cv

```
model = MultiHeadSelfAttention(dim=64)
```



Future of robotics?

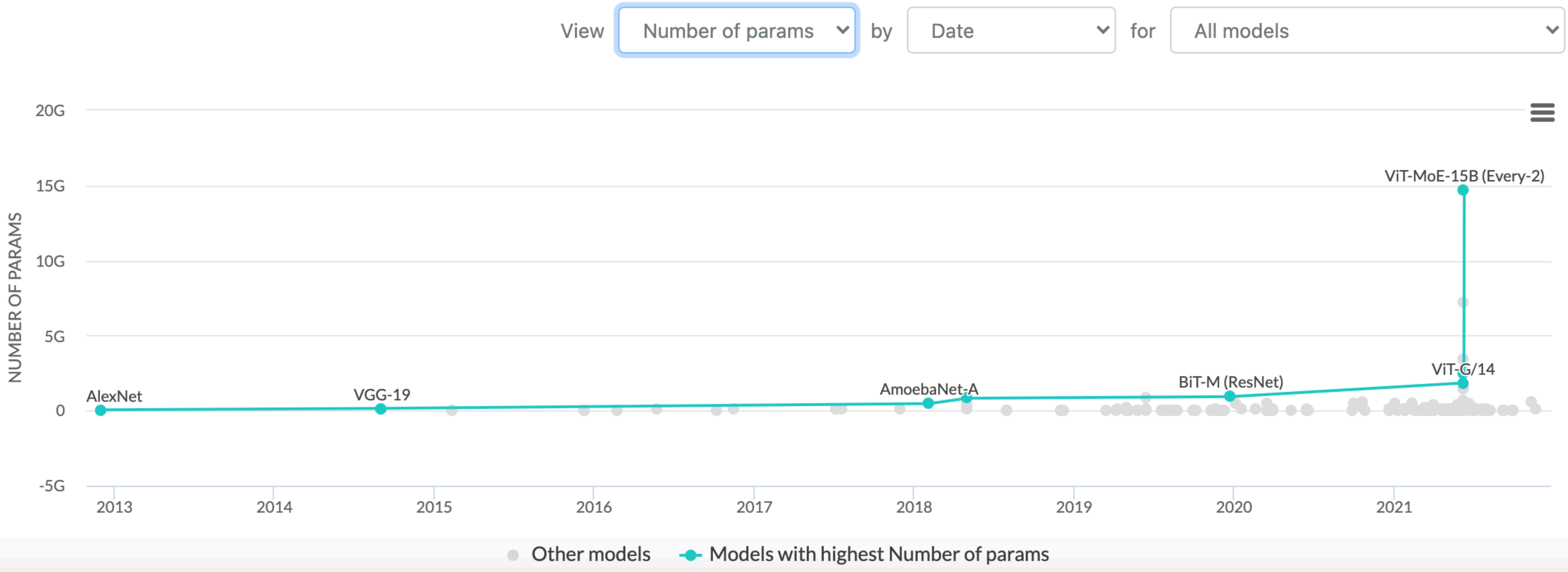
- Most of predictions were wrong
  - 1954 IBM predicted that natural language processing will be solved in 3 years
  - 1965 Herbert Simon: machines will replace humans in all manual works
  - 1970 Marvin Minsky: machines will have general AI comparable with humans
  - 2014 Rei Kurzweil: the same for for 2029, now talks about 2045

- Rodney Brooks prediction score card:
https://rodneybrooks.com/predictions-scorecard-2021-january-01/

- False generalization
  - AI is better in solving particular instances (image processing, stabilization)
  - Rather unique successes than exponentially growing start general AI

Future of robotics?

- Moorse law:
  - number of tranzistors in integrated ciruit will double every year
- Observation:
  - size of models (learning time => number of GPUs+memory) grows explonentially



Future of robotics?

- Moorse law:
  - number of tranzistors in integrated ciruit will double every year
- Observation:
  - size of models (learning time => number of GPUs+memory) grows explonentially
- What we get from it?
  - Improvement in accuracy of AI models slows down