University of Glasgow | School of Computing Science

THE AWARDS 2020 | UNIVERSITY OF THE YEAR

Recurrent Neural Networks

Dr. Fani Deligianni,

fani.deligianni@glasgow.ac.uk

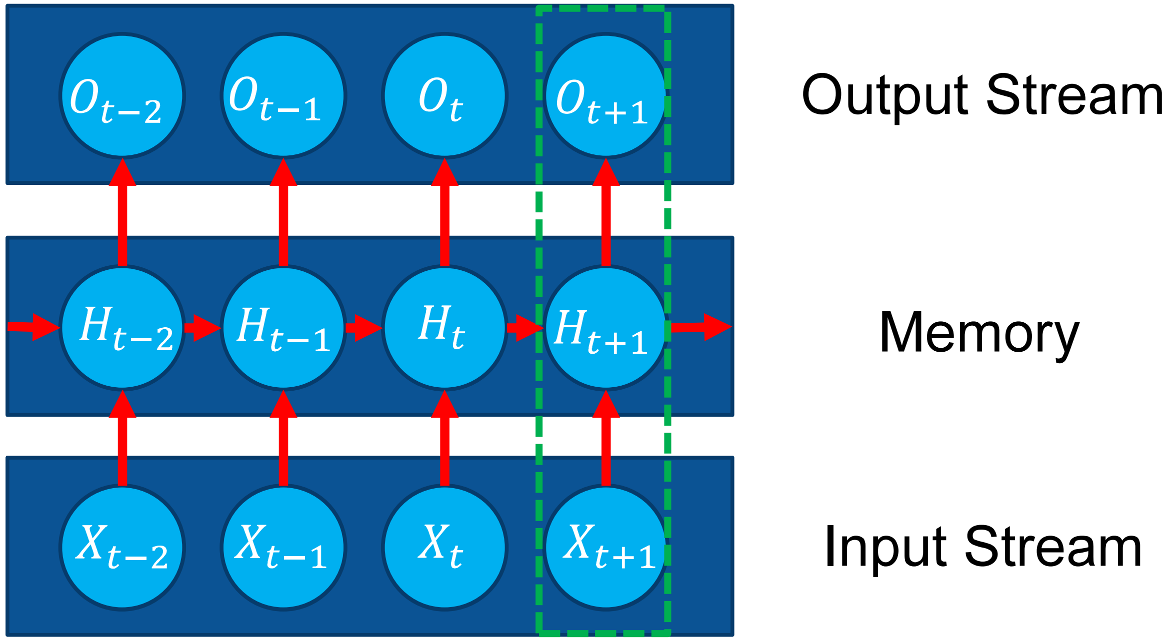
Lecturer (Assistant Professor)

Lead of the Computing Technologies for Healthcare Theme

https://www.gla.ac.uk/schools/computing/staff/fanideligianni

WORLD CHANGING GLASGOW

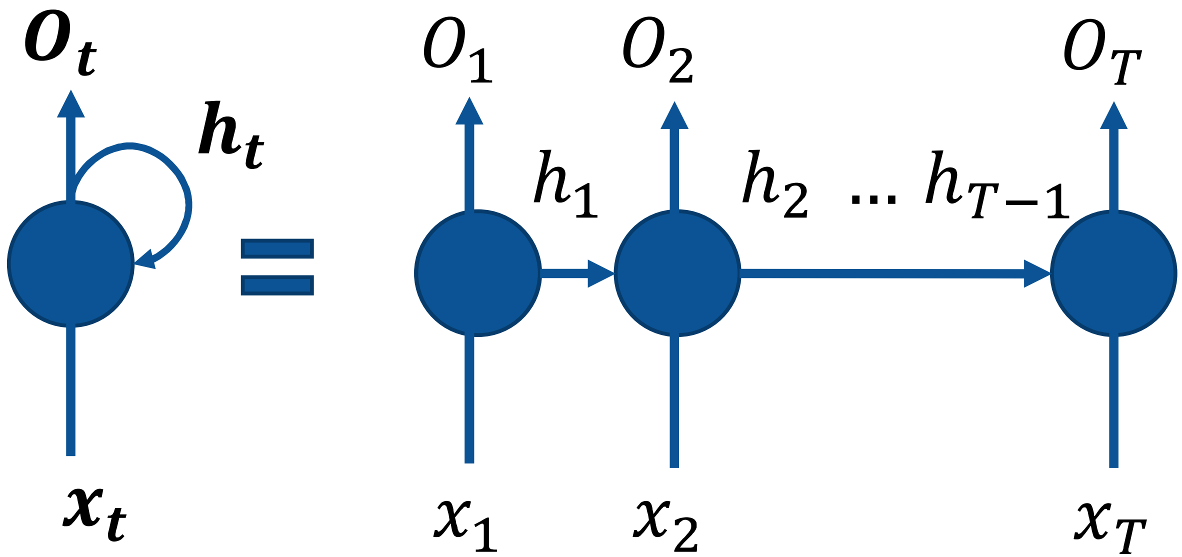
Recurrent Neural Networks



- RNNs accumulate information from each time step
- RNNs can learn representations for variable length sequences

Recurrent Neural Networks

$$\boldsymbol{O}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

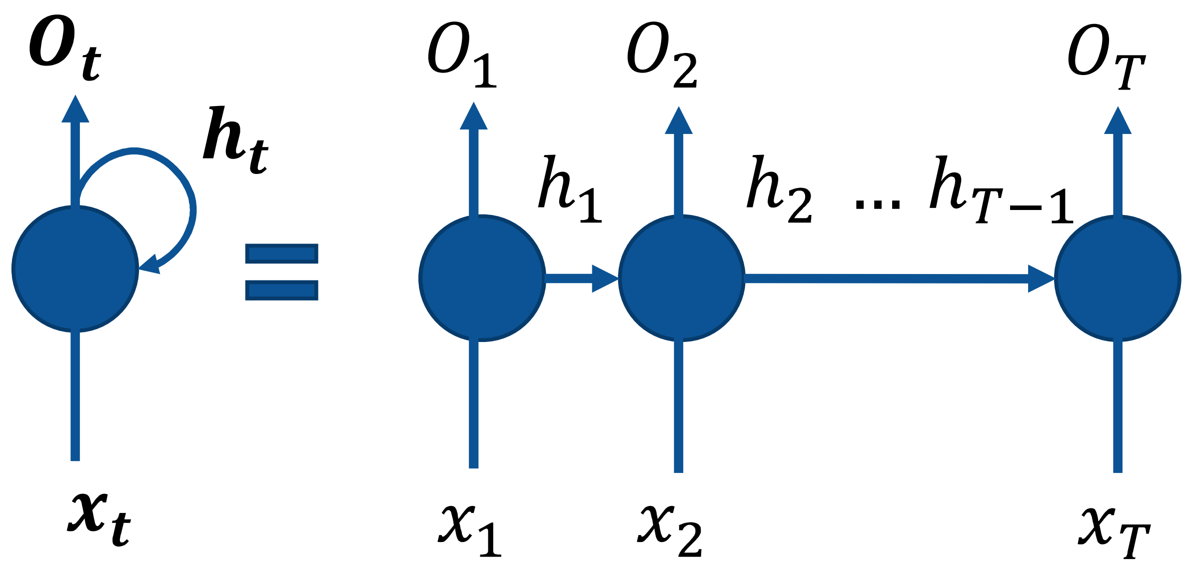


- RNNs accumulate information from each time step
- RNNs can learn representations for variable length sequences

Recurrent Neural Networks

$$\boldsymbol{O}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{h}_t = \boldsymbol{O}_t$$



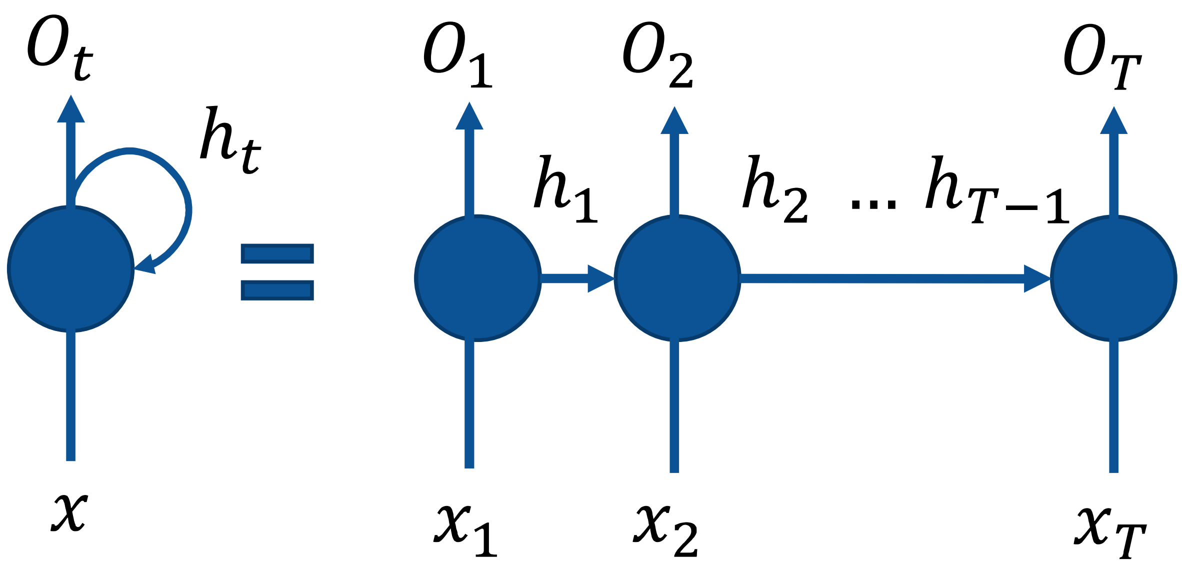
- RNNs accumulate information from each time step
- RNNs can learn representations for variable length sequences

Recurrent Neural Networks

$$\boldsymbol{O}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{h}_t = \boldsymbol{O}_t$$

$$\boldsymbol{h}_t = f(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1})$$

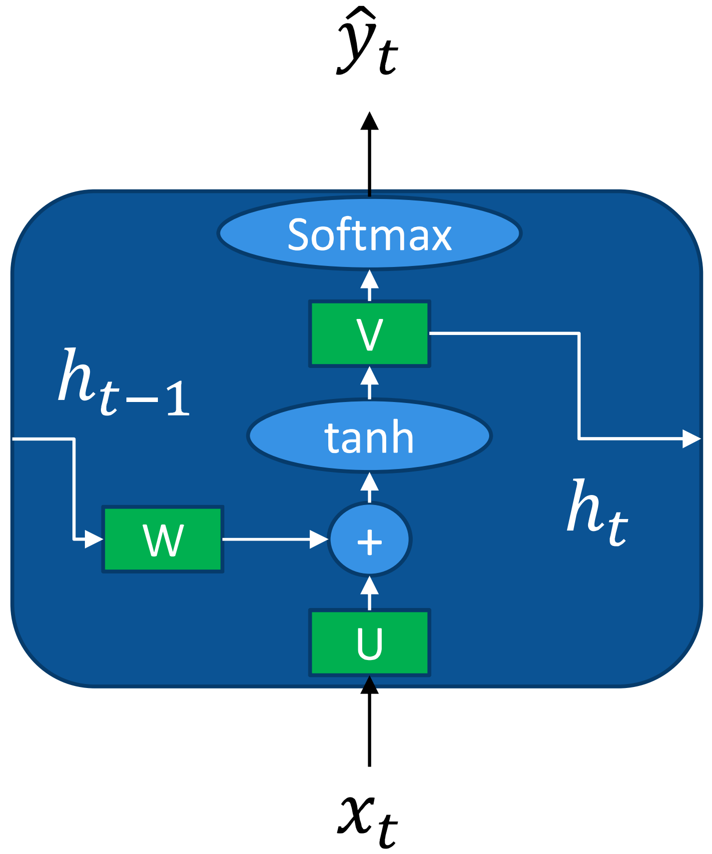


- RNNs accumulate information from each time step
- RNNs can learn representations for variable length sequences

Forward Propagation in RNNs

$$\boldsymbol{h_t} = \tanh(\boldsymbol{U}\boldsymbol{x_t} + \boldsymbol{W}\boldsymbol{h_{t-1}})$$

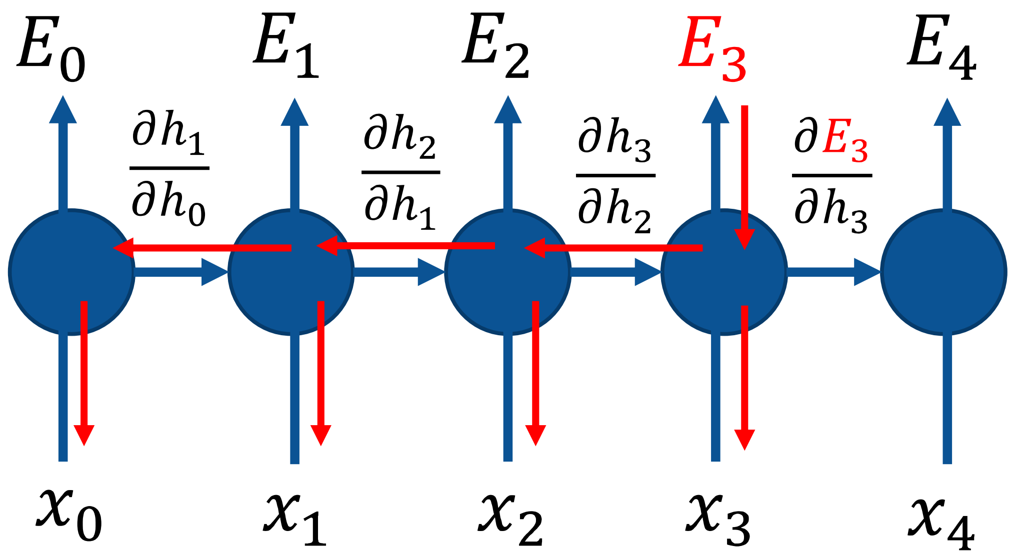
$$\widehat{\boldsymbol{y}}_t = \mathrm{softmax}(\boldsymbol{V}\boldsymbol{h_t})$$



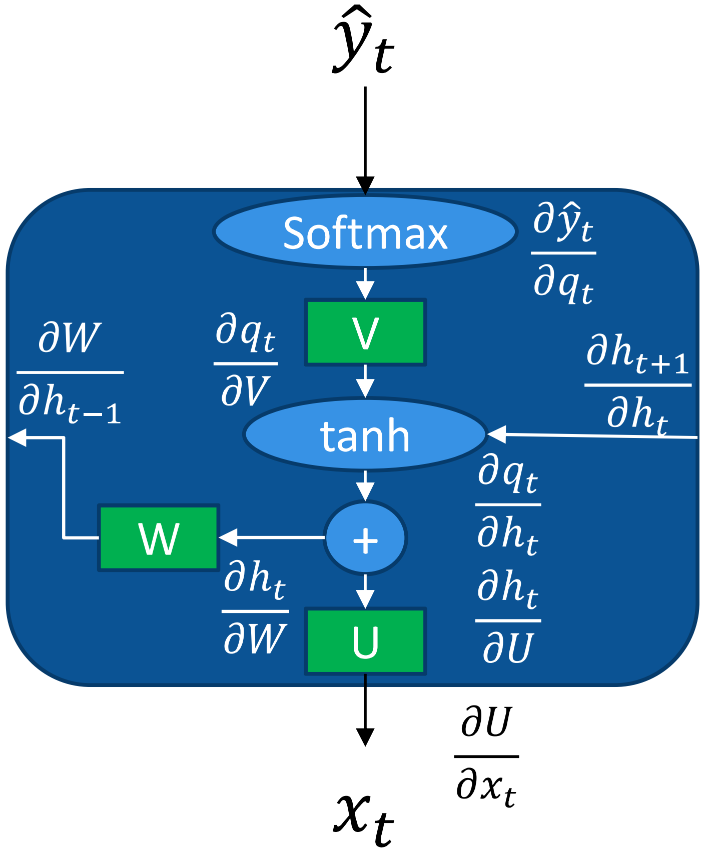
Back Propagation in RNNs

$$E_t = -\boldsymbol{y}_t log\widehat{\boldsymbol{y}}_t$$

$$L(\boldsymbol{y},\widehat{\boldsymbol{y}}_.) = -\frac{1}{N}\sum_t \boldsymbol{y}_t \, log\widehat{\boldsymbol{y}}_t$$



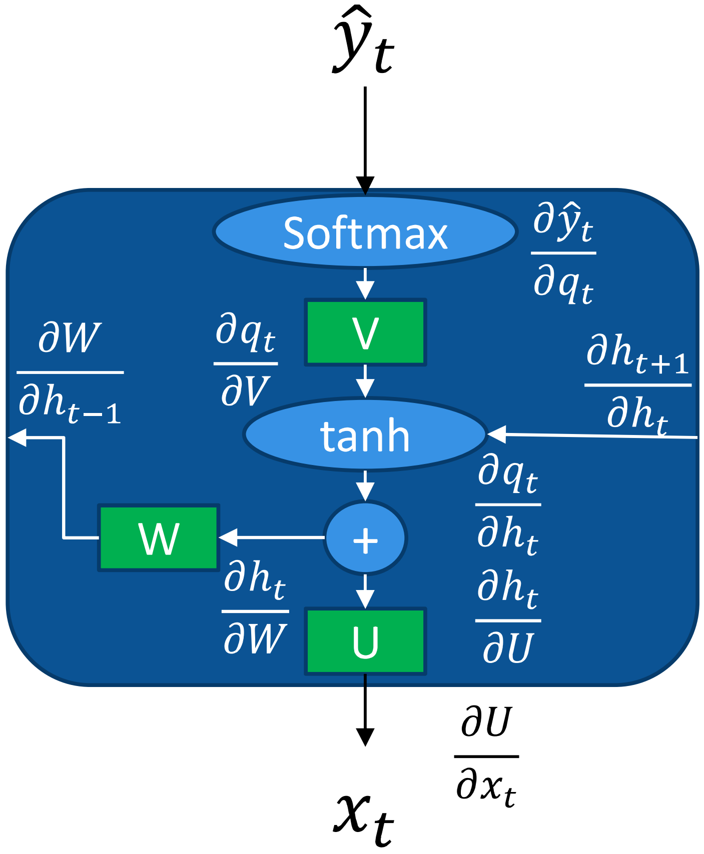
Back Propagation in RNNs



Back Propagation in RNNs

Recursive weights:

$$\frac{\partial E_t}{\partial \boldsymbol{W}} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial q_t}\frac{\partial q_t}{\partial h_t}\frac{\partial h_t}{\partial \boldsymbol{W}}$$

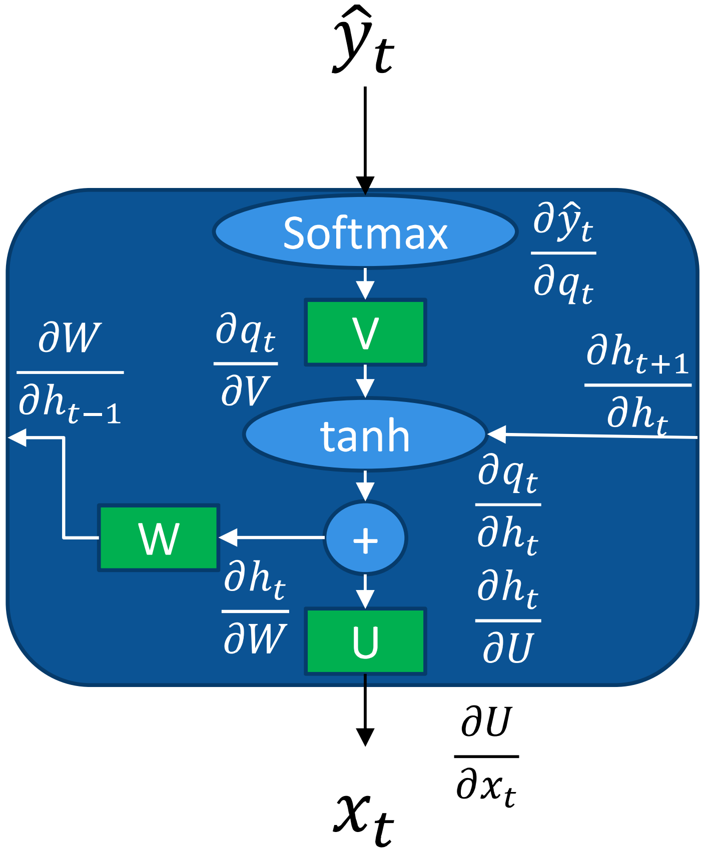


Back Propagation in RNNs

Recursive weights:

$$\frac{\partial E_t}{\partial \boldsymbol{W}} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial q_t}\frac{\partial q_t}{\partial h_t}\frac{\partial h_t}{\partial \boldsymbol{W}}$$

$$\frac{\partial h_t}{\partial \boldsymbol{W}} \rightarrow \frac{\partial h_t}{\partial \boldsymbol{W}} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial \boldsymbol{W}}$$



Back Propagation in RNNs

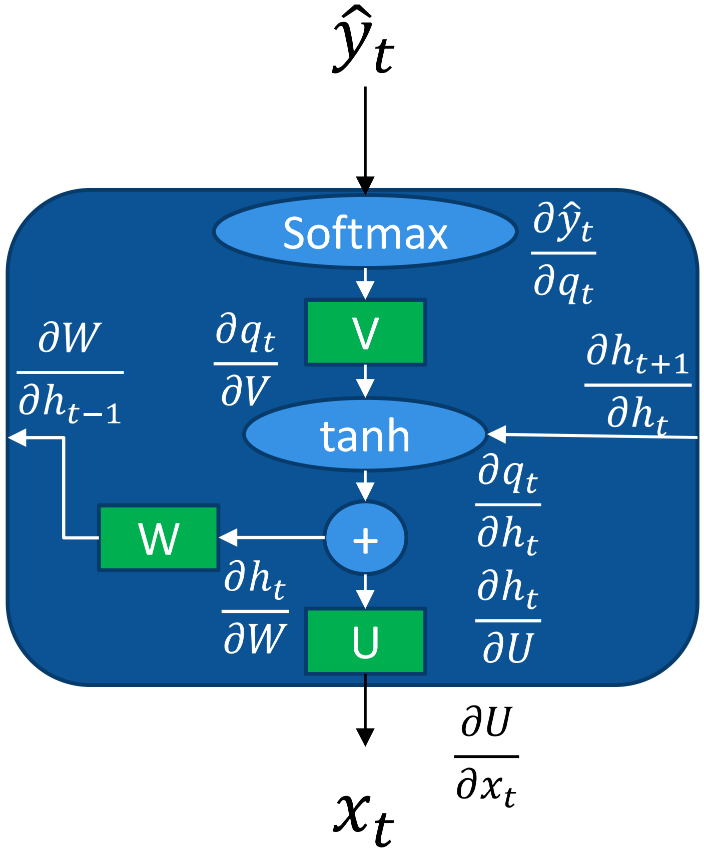
Recursive weights:

$$\frac{\partial E_t}{\partial \boldsymbol{W}} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial q_t}\frac{\partial q_t}{\partial h_t}\frac{\partial h_t}{\partial \boldsymbol{W}}$$

$$\frac{\partial h_t}{\partial \boldsymbol{W}} \rightarrow \frac{\partial h_t}{\partial \boldsymbol{W}} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial \boldsymbol{W}}$$

$$\frac{\partial h_{t-2}}{\partial \boldsymbol{W}}$$

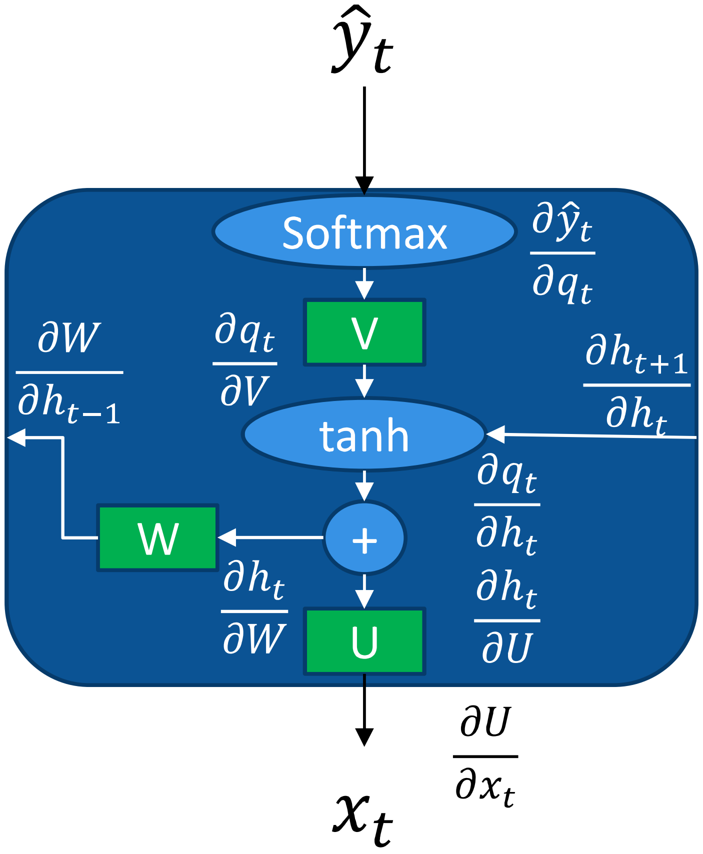
...



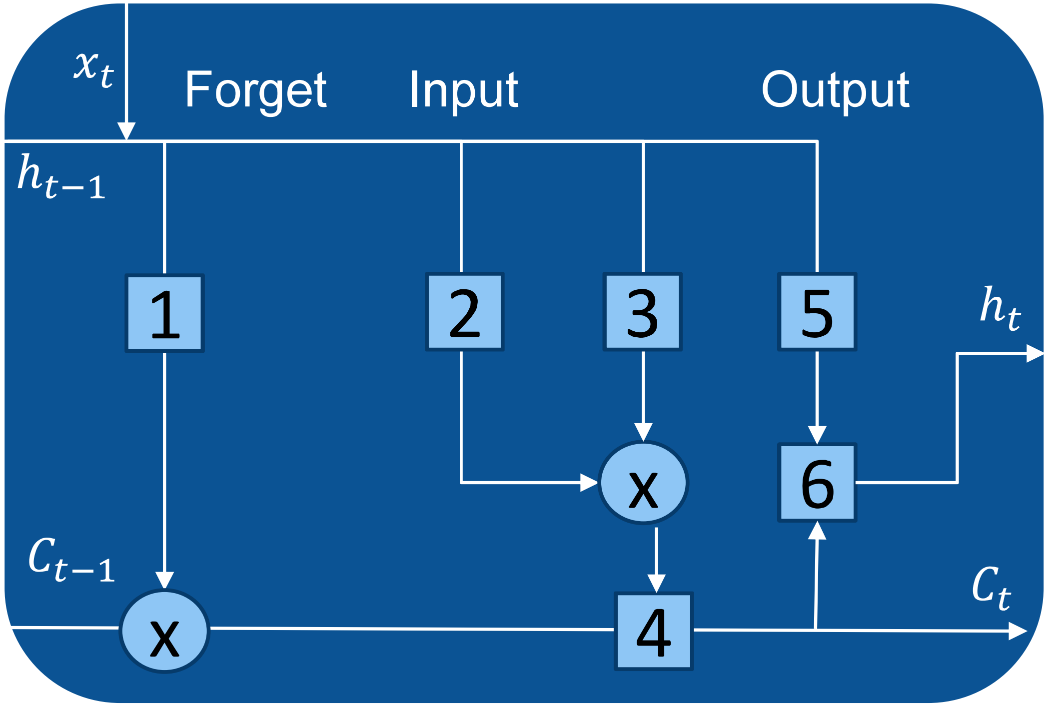
Back Propagation in RNNs

Vanishing/Exploding gradient:

- Careful initialization
- Adaptive rates (Adam optimizer)
- Second order derivatives to predict the occurrence of vanishing gradient



Long Short-Term Memory (LSTM)

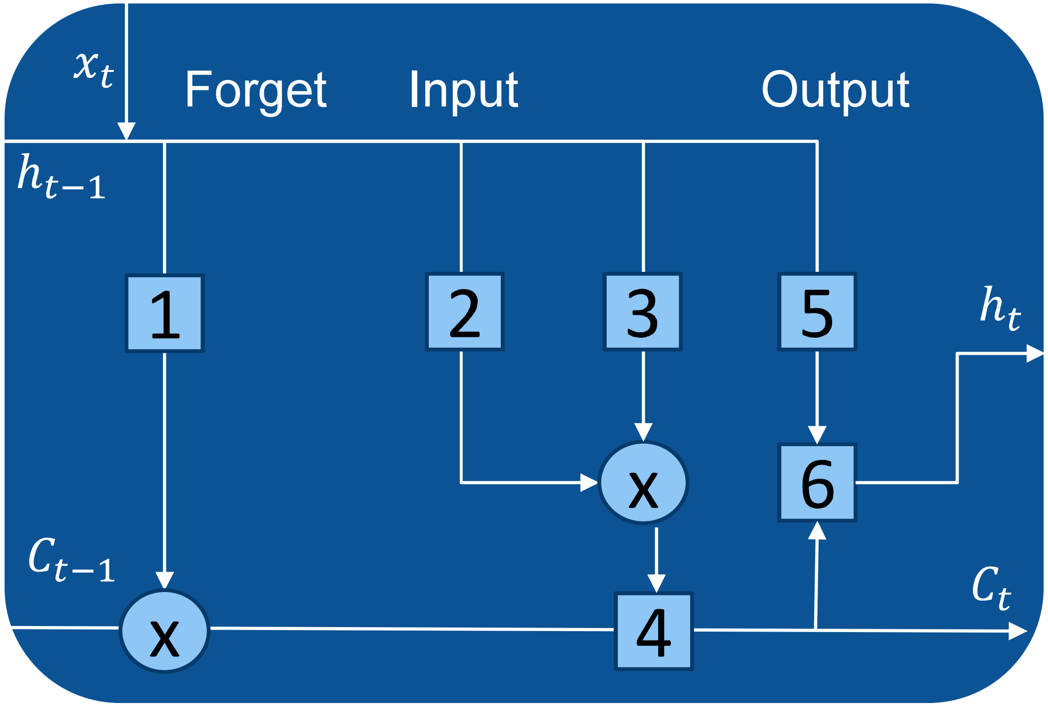


$\boxed{1}$ $$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Long Short-Term Memory (LSTM)



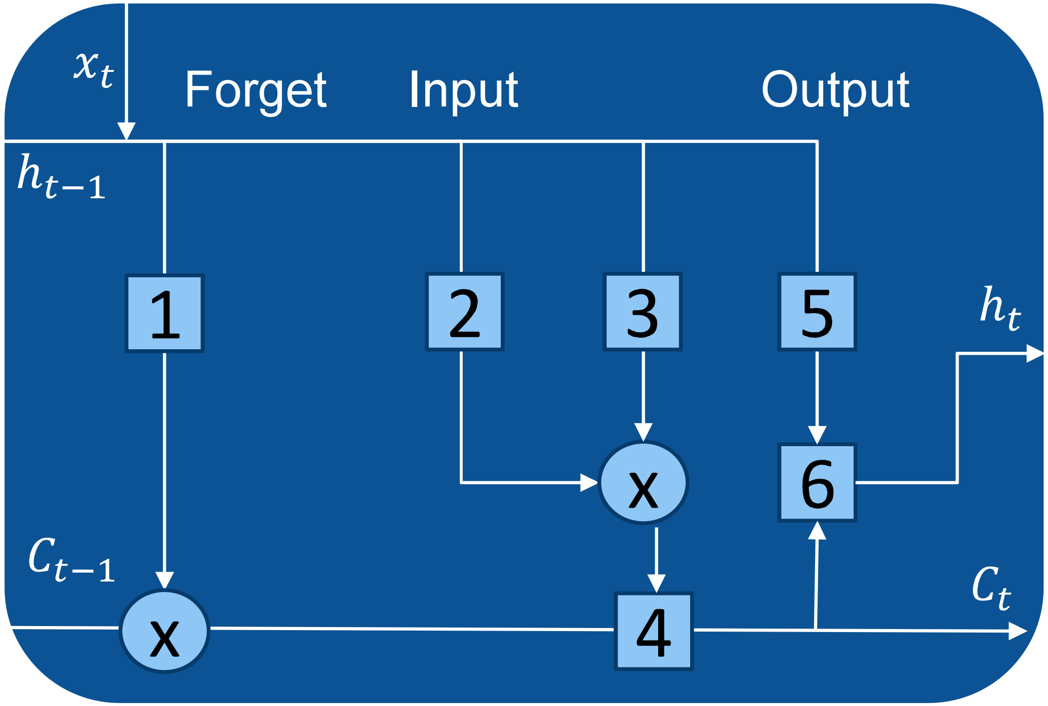
$\boxed{1}$ $$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

$\boxed{2}$ $$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Long Short-Term Memory (LSTM)



$\boxed{1}$ $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$

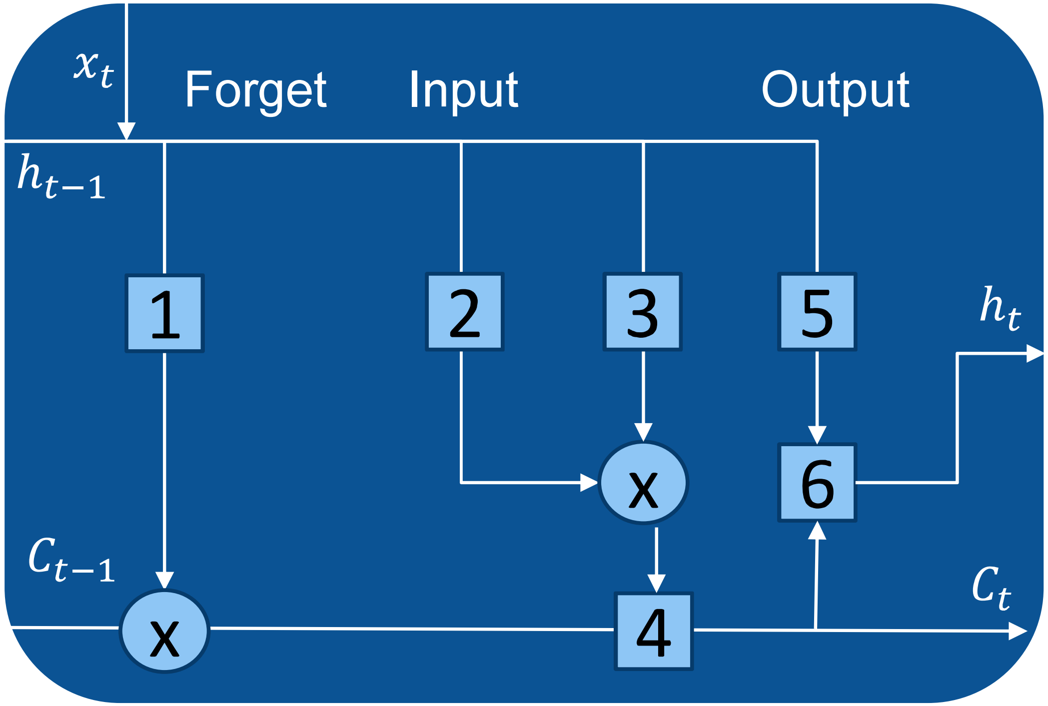
$\boxed{2}$ $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$

$\boxed{3}$ $\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Long Short-Term Memory (LSTM)

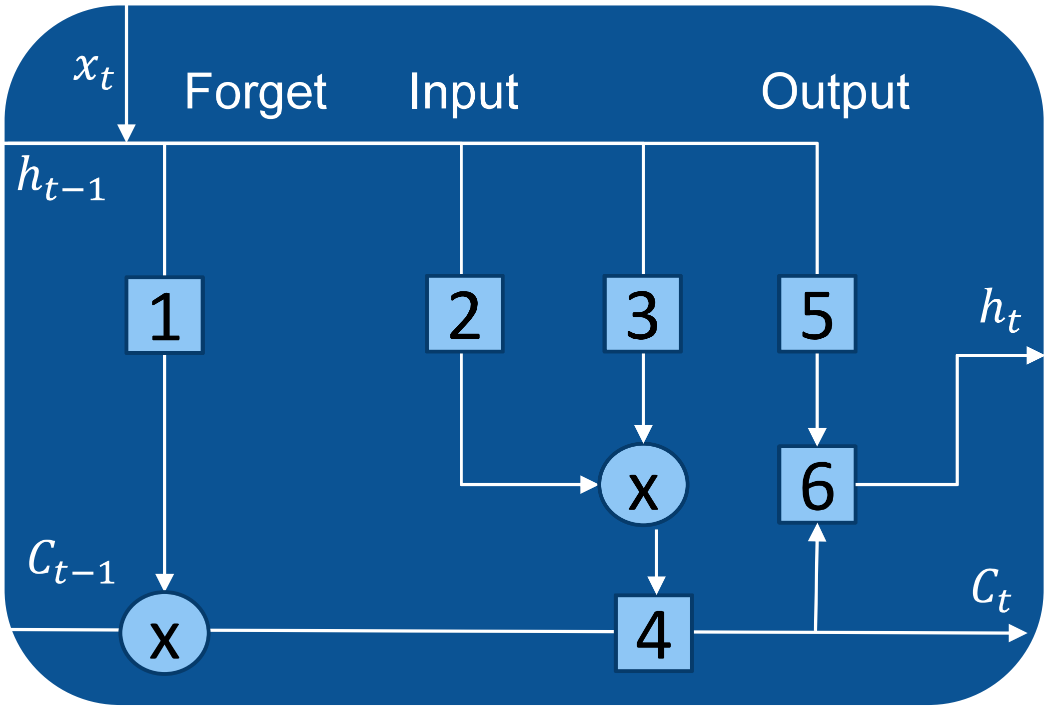


1. $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$
2. $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$
3. $\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$
4. $C_t = (f_t * C_{t-1} + i_t * \tilde{C}_t)$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Long Short-Term Memory (LSTM)

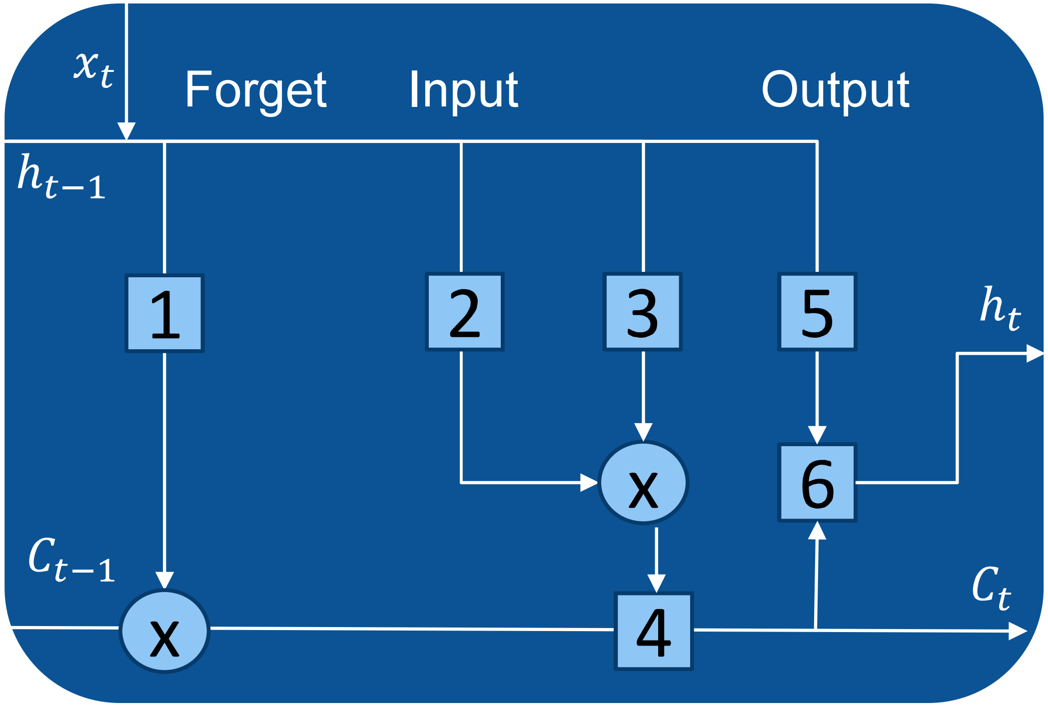


1. $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$
2. $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$
3. $\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$
4. $C_t = (f_t * C_{t-1} + i_t * \tilde{C}_t)$
5. $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Long Short-Term Memory (LSTM)



1 $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$

2 $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$

3 $\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$

4 $C_t = (f_t * C_{t-1} + i_t * \tilde{C}_t)$

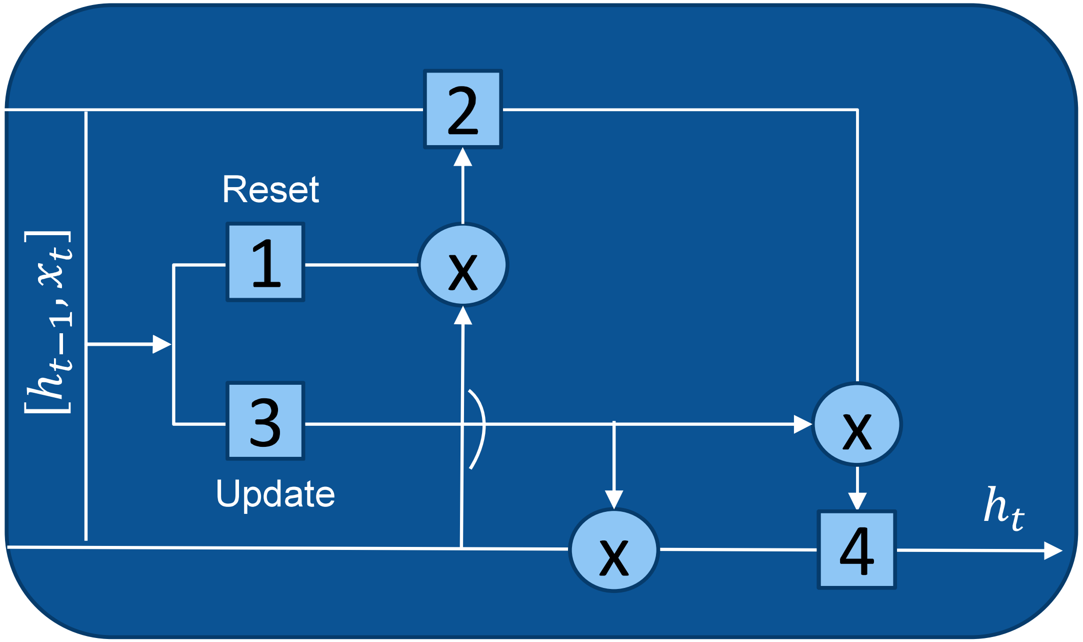
5 $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$

6 $h_t = o_t * \tanh(C_t)$

To solve vanishing/exploding gradient:

- Introduce gates (Input, Output, Forget)

Gated Recurrent Unit

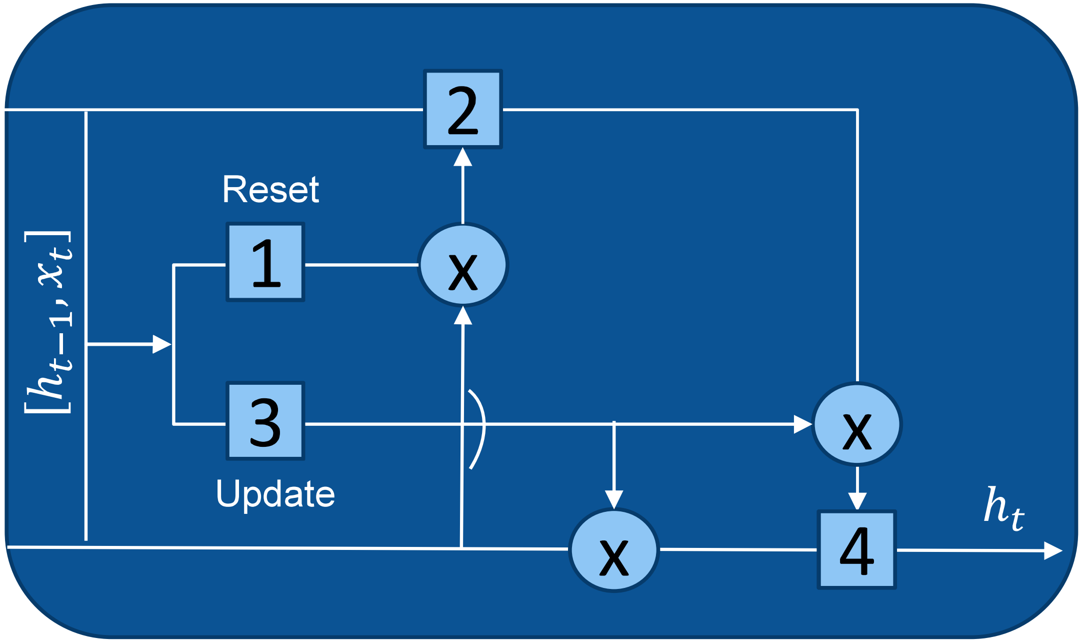


$\boxed{1}$ $r_t = \sigma(W_r * [h_{t-1}, x_t])$

Key difference with LSTM:

- The forget and input gates are replaced by reset and update gates
- There is no output gait

Gated Recurrent Unit



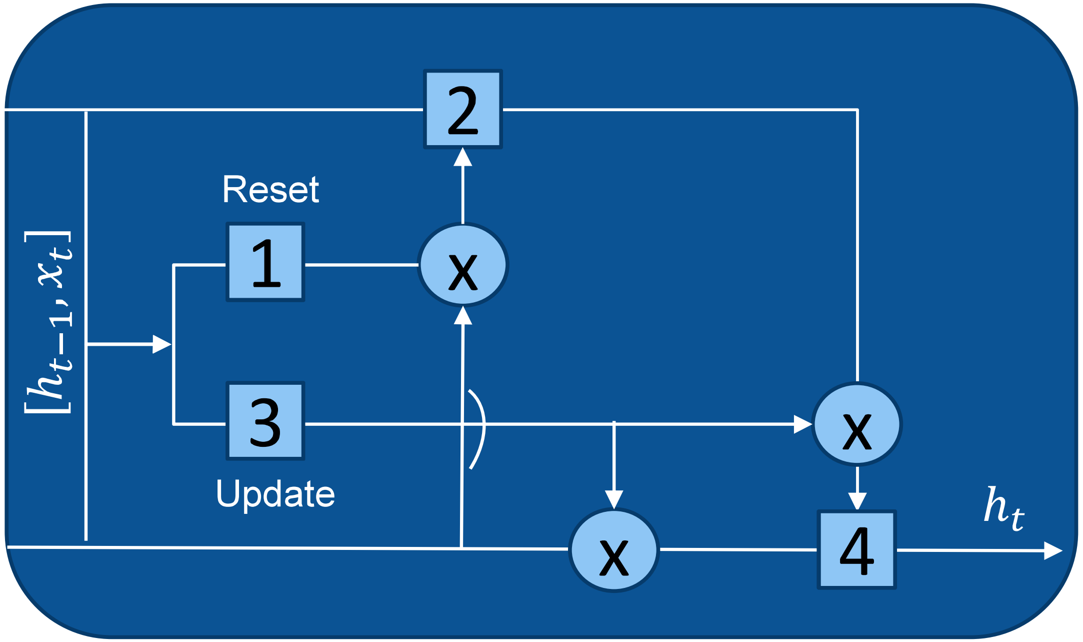
1 $r_t = \sigma(W_r * [h_{t-1}, x_t])$

2 $\tilde{h}_t = tanh(W_. * [r_t * h_{t-1}, x_t])$

Key difference with LSTM:

- The forget and input gates are replaced by reset and update gates
- There is no output gait

Gated Recurrent Unit



$\boxed{1}$ $r_t = \sigma(W_r * [h_{t-1}, x_t])$

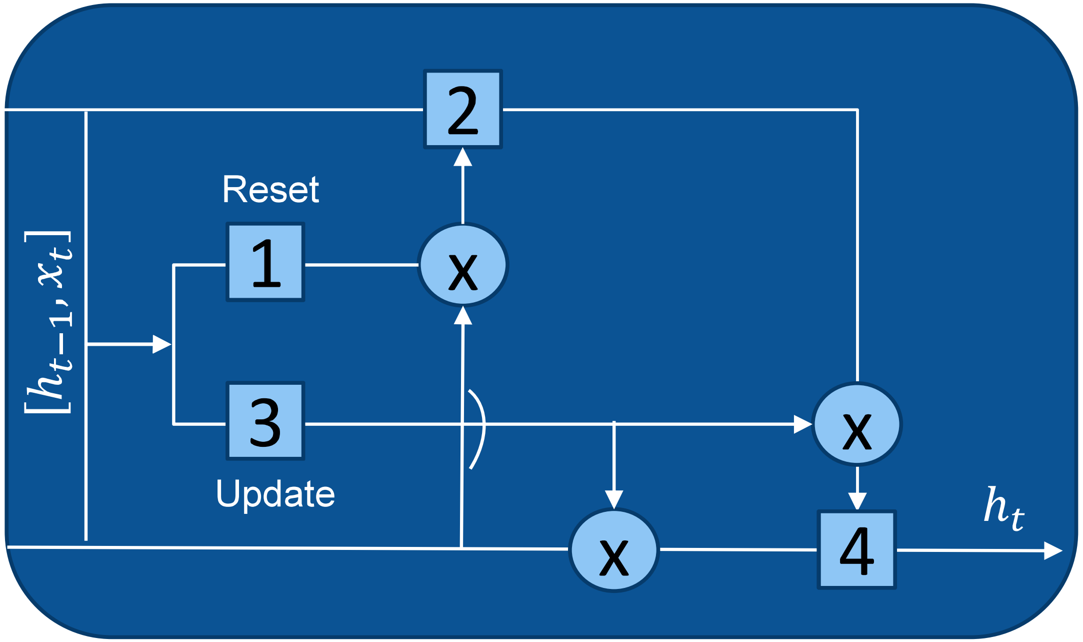
$\boxed{2}$ $\tilde{h}_t = tanh(W_. * [r_t * h_{t-1}, x_t])$

$\boxed{3}$ $z_t = \sigma(W_z * [h_{t-1}, x_t])$

Key difference with LSTM:

- The forget and input gates are replaced by reset and update gates
- There is no output gait

Gated Recurrent Unit



1 $r_t = \sigma(W_r * [h_{t-1}, x_t])$

2 $\tilde{h}_t = tanh(W_{.} * [r_t * h_{t-1}, x_t])$

3 $z_t = \sigma(W_z * [h_{t-1}, x_t])$

4 $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

Key difference with LSTM:

- The forget and input gates are replaced by reset and update gates
- There is no output gait

Summary

- RNNs handle time-series data by implementing a memory mechanism
- Compared to CNNs are more difficult to train due to the vanishing gradients
- LSTM and GRU are types of RNNs that have been introduced to alleviate the problem of the vanishing/exploding gradients

References

- Ravi et al. Deep Learning for Health Informatics, IEEE Journal of Biomedical and Health Informatics, 21(1), 2017
- Kamath, Deep Learning for NLP Applications, Springer, 2019
- Foster, Generative Deep Learning – Teaching Machines to Paint, Write, Compose and Play, O’Reilly, 2019