Lecture 14

Introduction to Recurrent Neural Networks

STAT 453: Deep Learning, Spring 2020

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat453-ss2020/

Lecture Slides:
https://github.com/rasbt/stat453-deep-learning-ss20/tree/master/L14-rnns

A Classic Approach for Text Classification: Bag-of-Words Model

"Raw" training dataset

$\mathbf{x}^{[1]}$ = "The sun is shining"

$\mathbf{x}^{[2]}$ = "The weather is sweet"

$\mathbf{x}^{[3]}$ = "The sun is shining,
the weather is sweet, and
one and one is two"

```
vocabulary = {
    'and': 0,
    'is': 1
    'one': 2,
    'shining': 3,
    'sun': 4,
    'sweet': 5,
    'the': 6,
    'two': 7,
    'weather': 8,
}
```

Training set as design matrix

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 2 & 3 & 2 & 1 & 1 & 1 & 2 & 1 & 1 \end{bmatrix}$$

$\mathbf{y} = [0, 1, 0]$

class labels

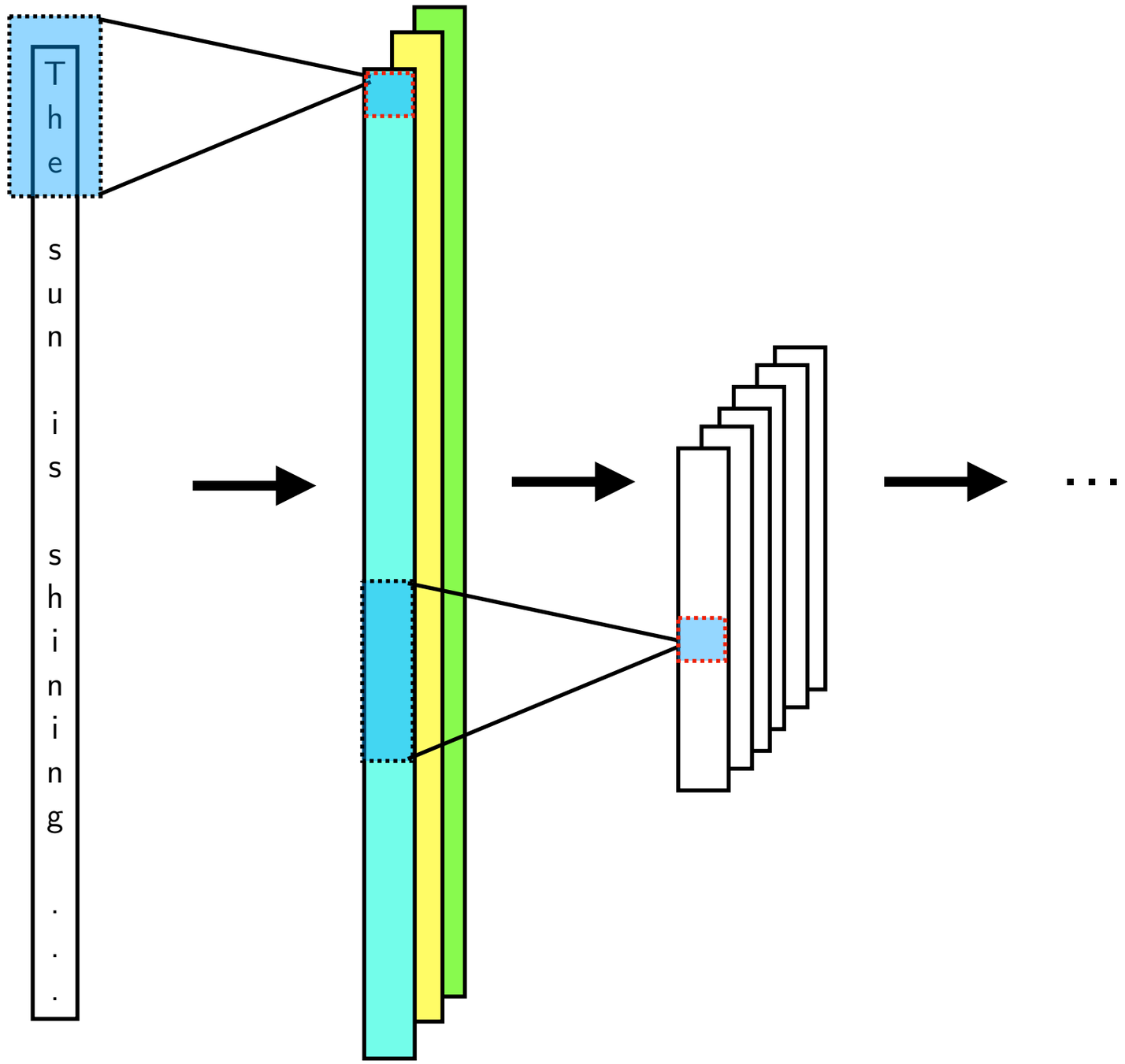
training

Classifier

(e.g., logistic regression, MLP, ...)

Ex.: https://github.com/rasbt/python-machine-learning-book-3rd-edition/tree/master/ch08

1D CNNs for text (and other sequence data)



Lecture Overview

RNNs and Sequence Modeling Tasks

Backpropagation Through Time

Long-short term memory (LSTM)

Many-to-one Word RNNs

Generating Text with Character RNNs

Attention Mechanisms and Transformers

Sequential data is not i.i.d.

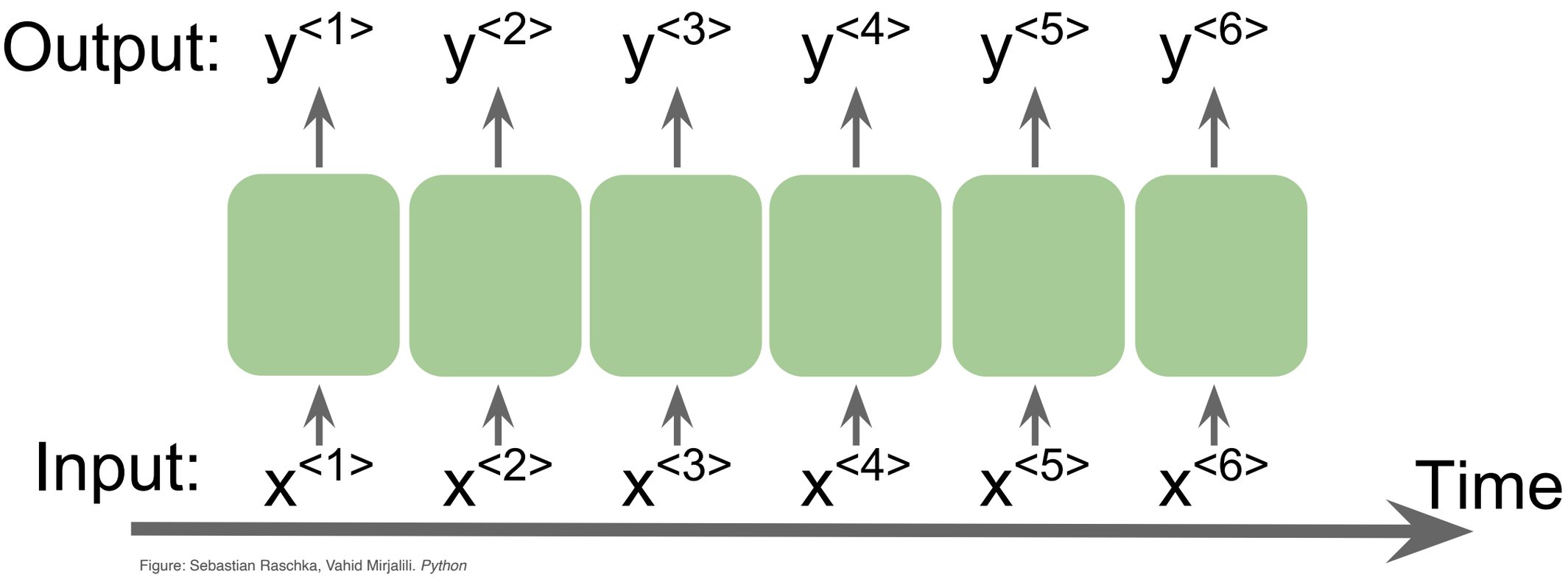
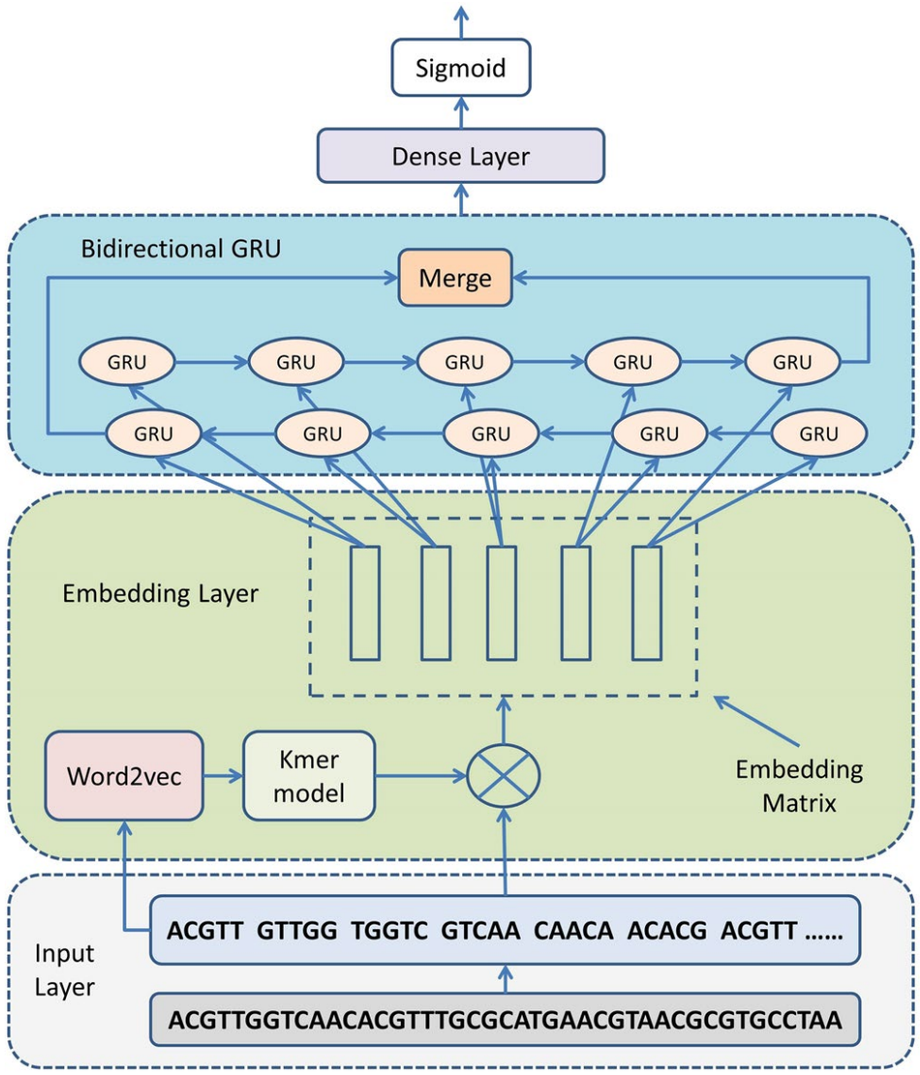


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Applications:
Working with Sequential Data

- Text classification
- Speech recognition (acoustic modeling)
- language translation
- ...



Shen, Zhen, Wenzheng Bao, and De-Shuang Huang. "Recurrent Neural Network for Predicting Transcription Factor Binding Sites." *Scientific reports* 8, no. 1 (2018): 15270.

DNA or (amino acid/protein) sequence modeling

Stock market predictions

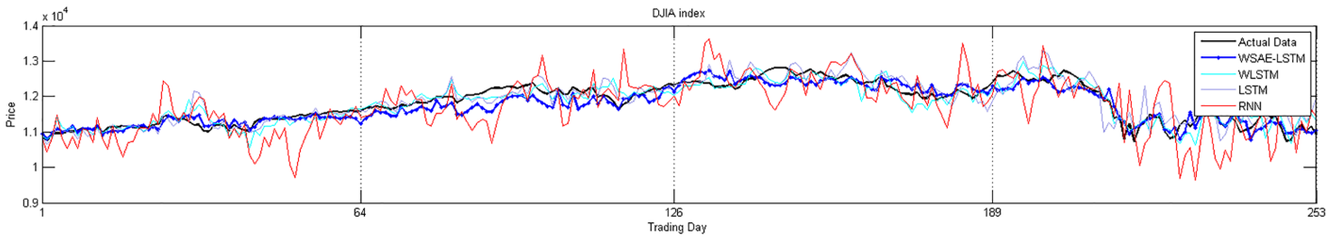


Fig 8. Displays the actual data and the predicted data from the four models for each stock index in Year 1 from 2010.10.01 to 2011.09.30.

https://doi.org/10.1371/journal.pone.0180944.g008

Bao, Wei, Jun Yue, and Yulei Rao. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory." *PloS one* 12, no. 7 (2017): e0180944.

Overview

Networks we used previously: also called feedforward neural networks

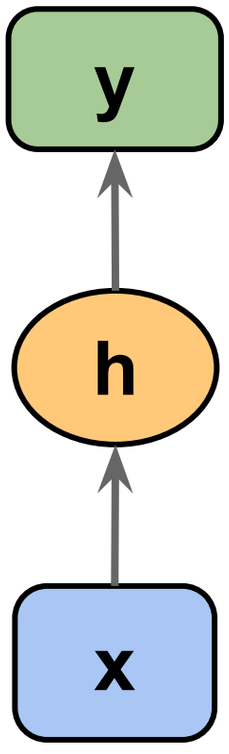
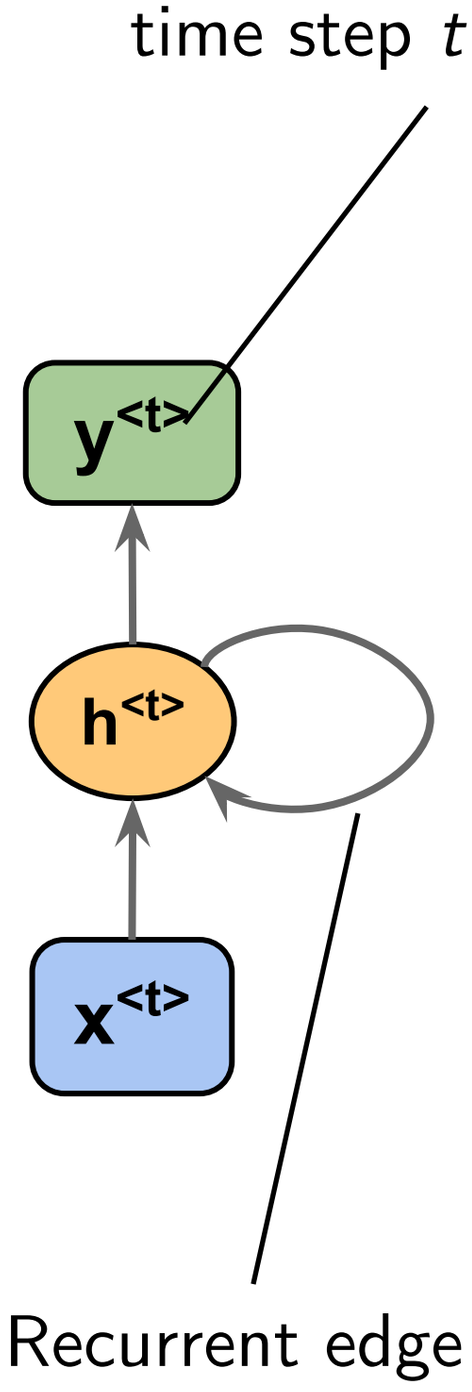
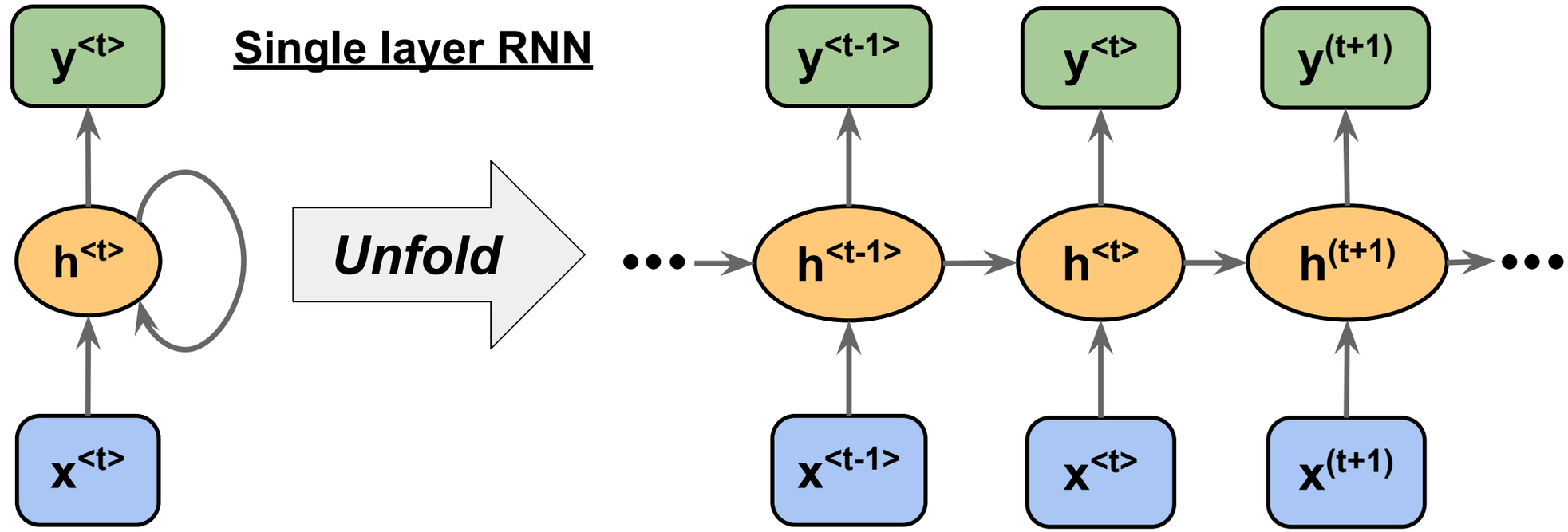


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Recurrent Neural Network (RNN)



Overview



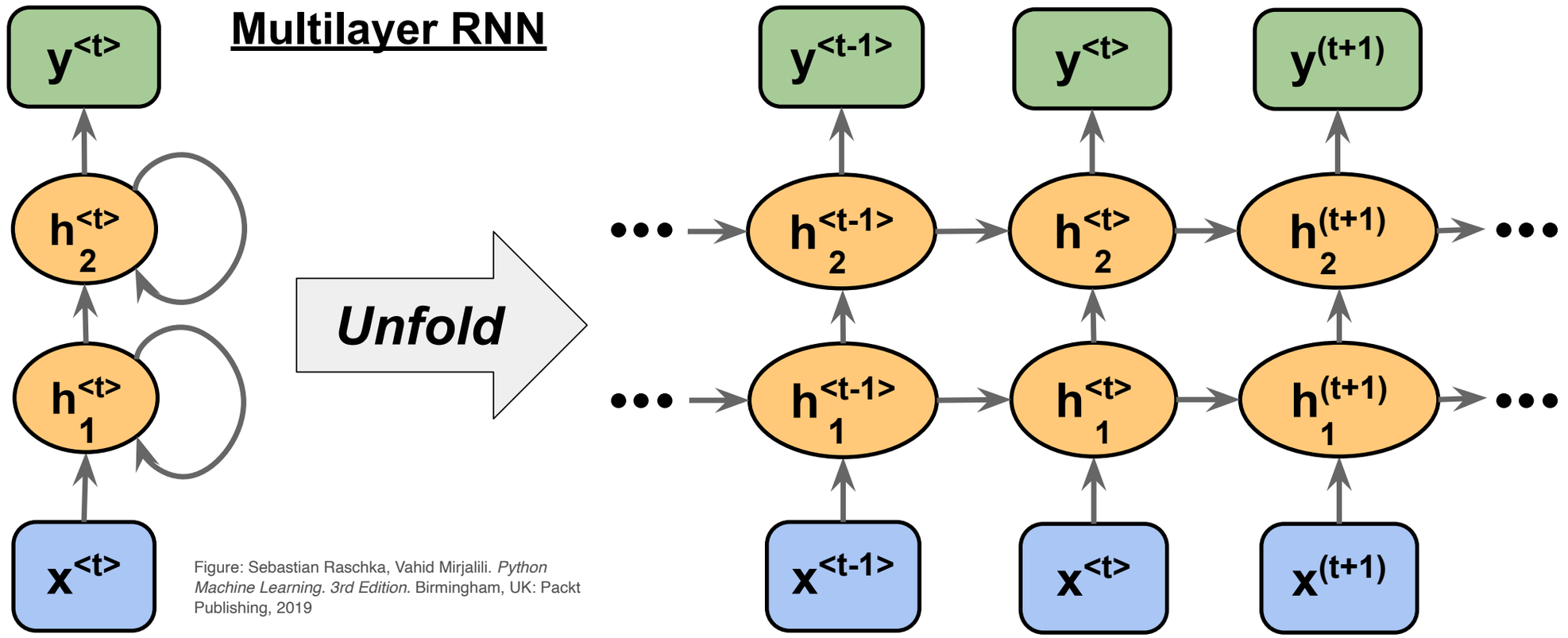
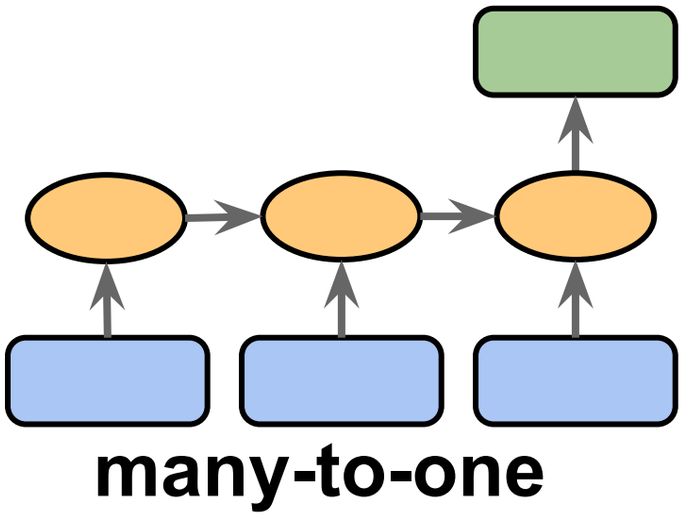
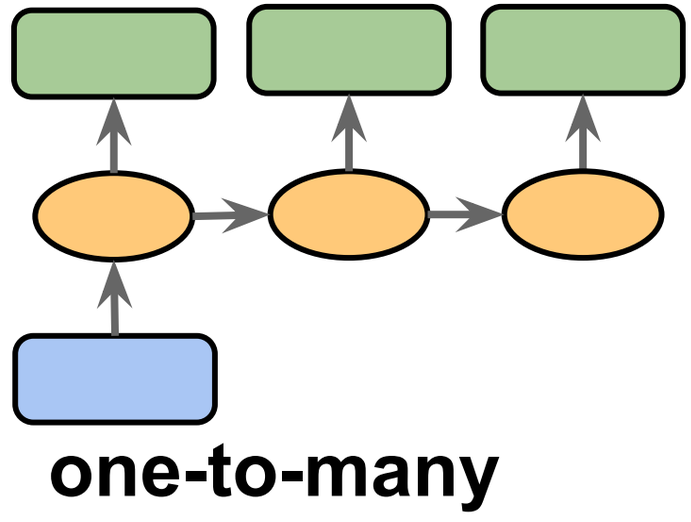
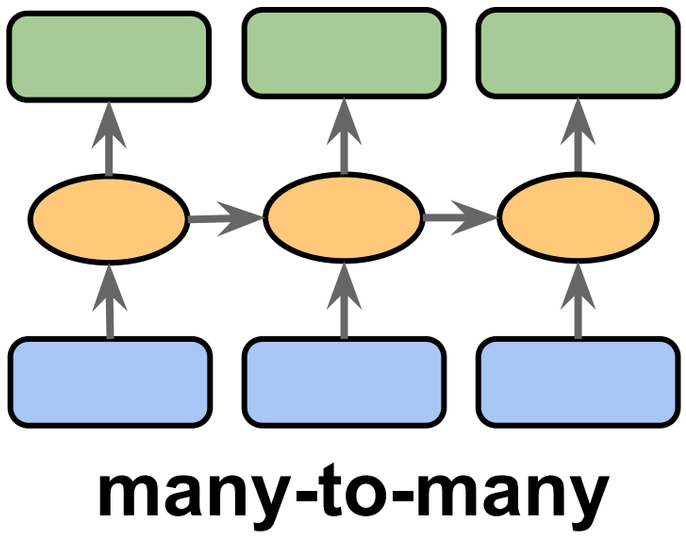


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Different Types of Sequence Modeling Tasks







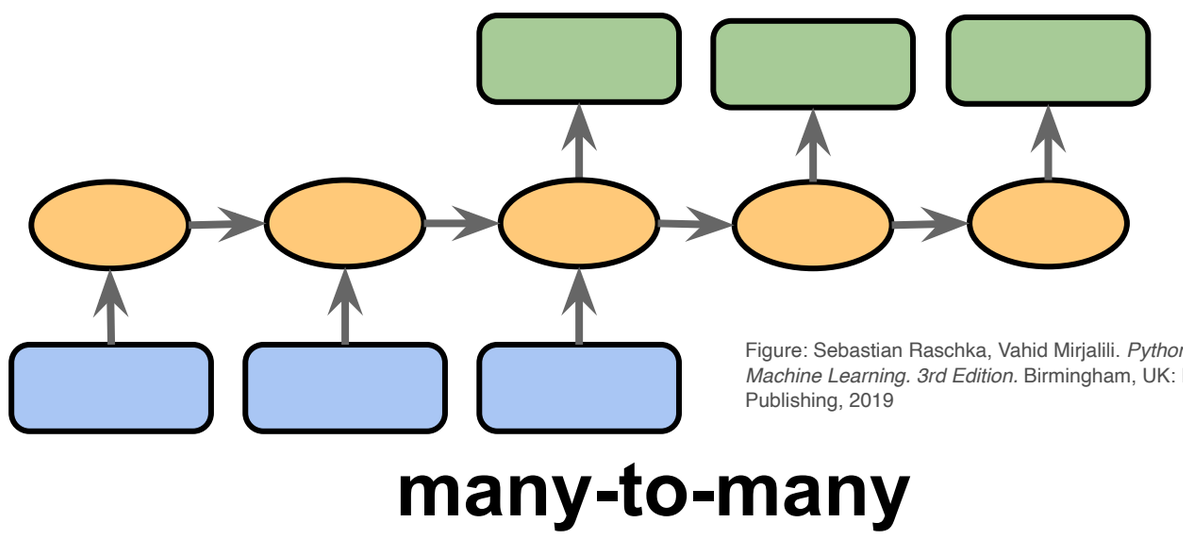
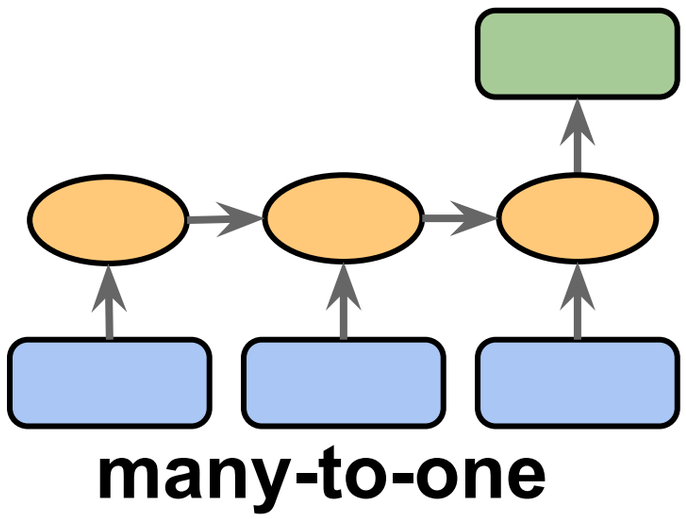


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Figure based on:

The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy (http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

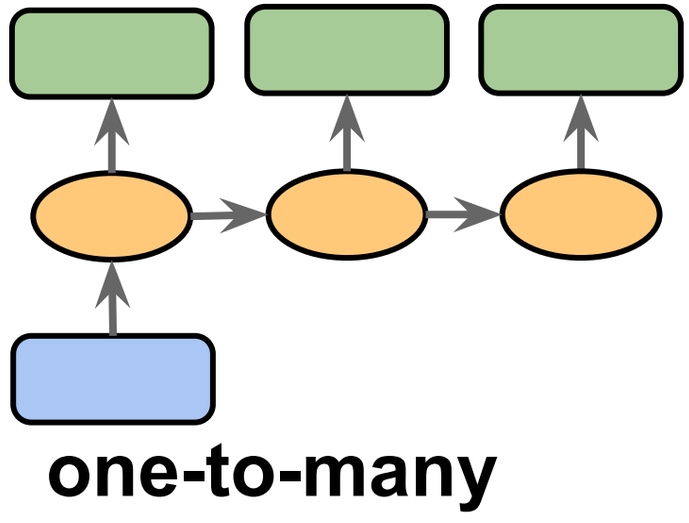
Different Types of Sequence Modeling Tasks



Many-to-one: The input data is a sequence, but the output is a fixed-size vector, not a sequence.

Ex.: sentiment analysis, the input is some text, and the output is a class label.

Different Types of Sequence Modeling Tasks



One-to-many: Input data is in a standard format (not a sequence), the output is a sequence.

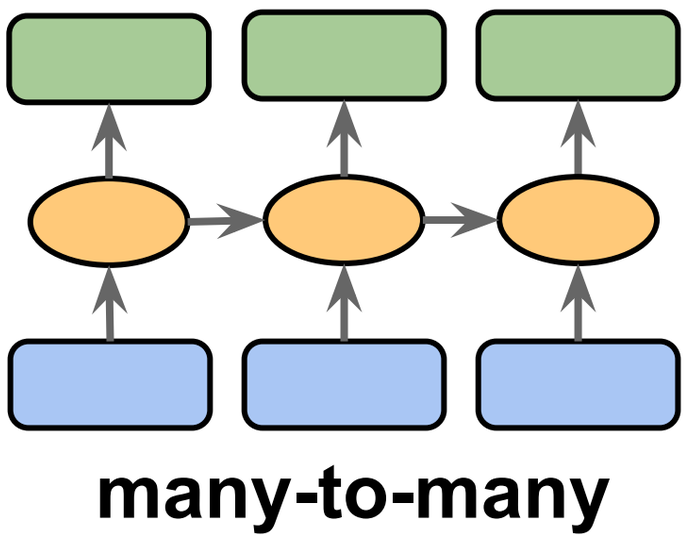
Ex.: Image captioning, where the input is an image, the output is a text description of that image

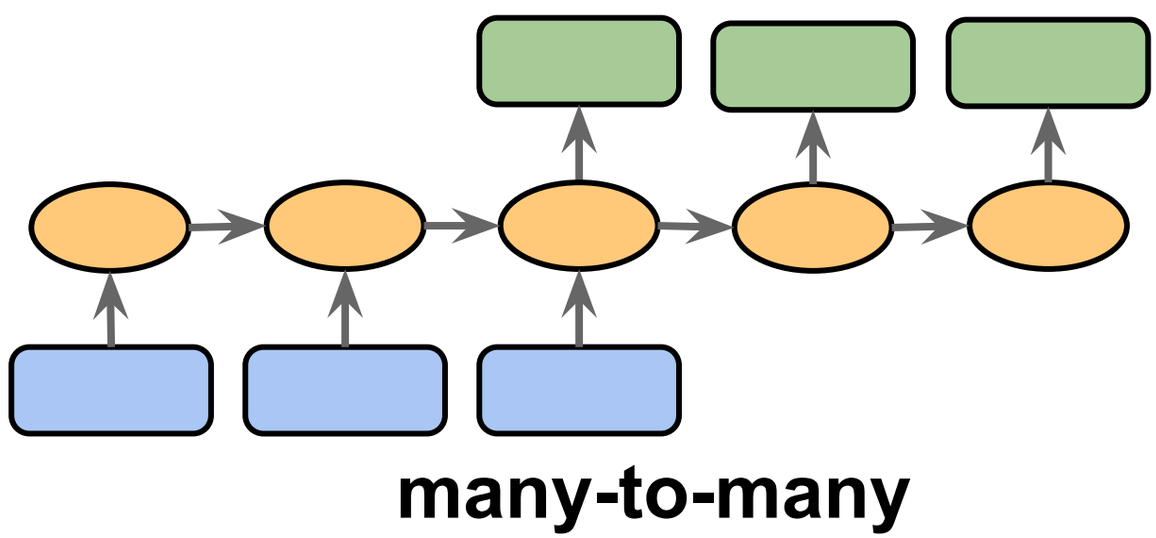
Different Types of Sequence Modeling Tasks

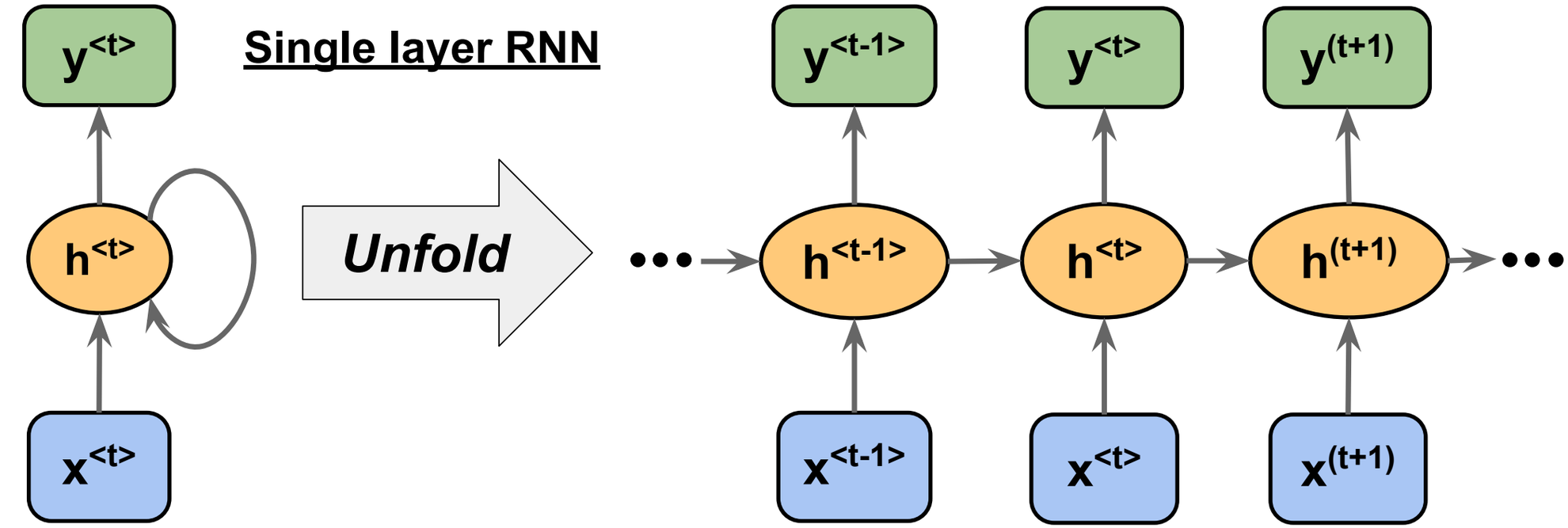
Many-to-many: Both inputs and outputs are sequences. Can be direct or delayed.

Ex.: Video-captioning, i.e., describing a sequence of images via text (direct).

Translating one language into another (delayed)







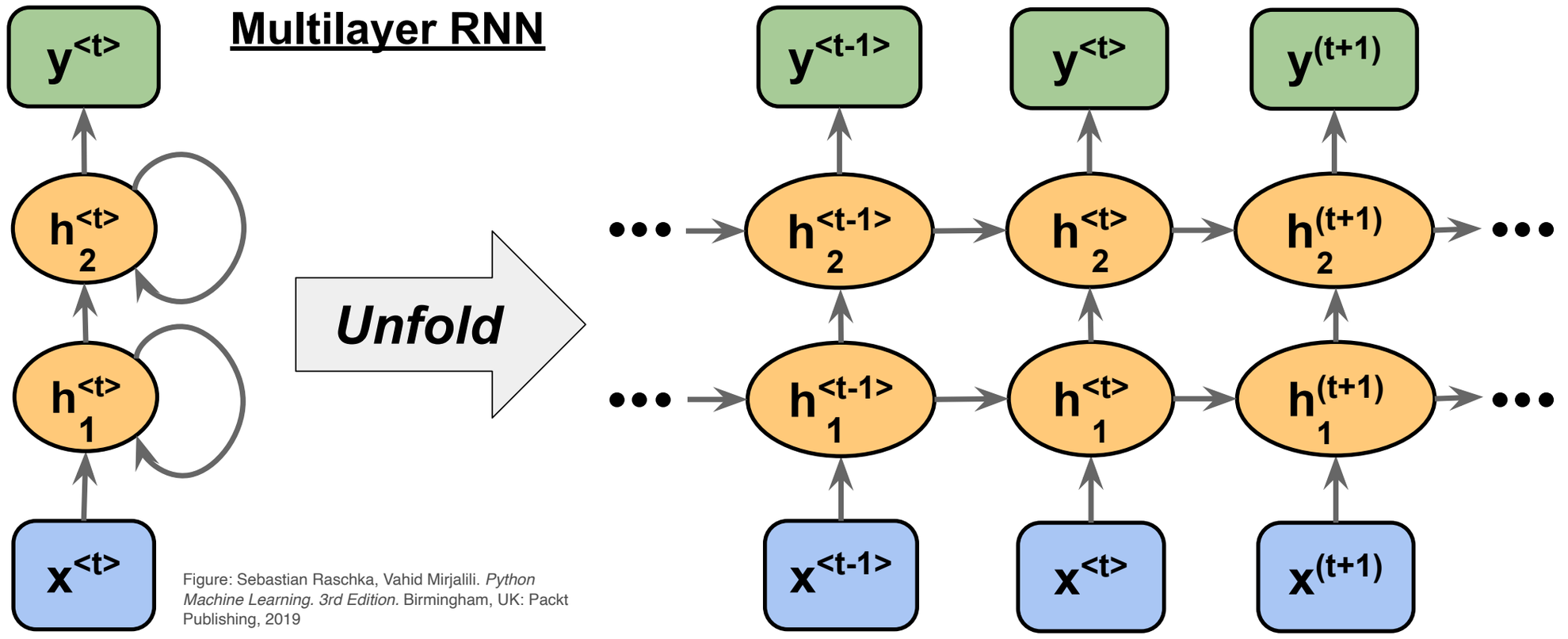
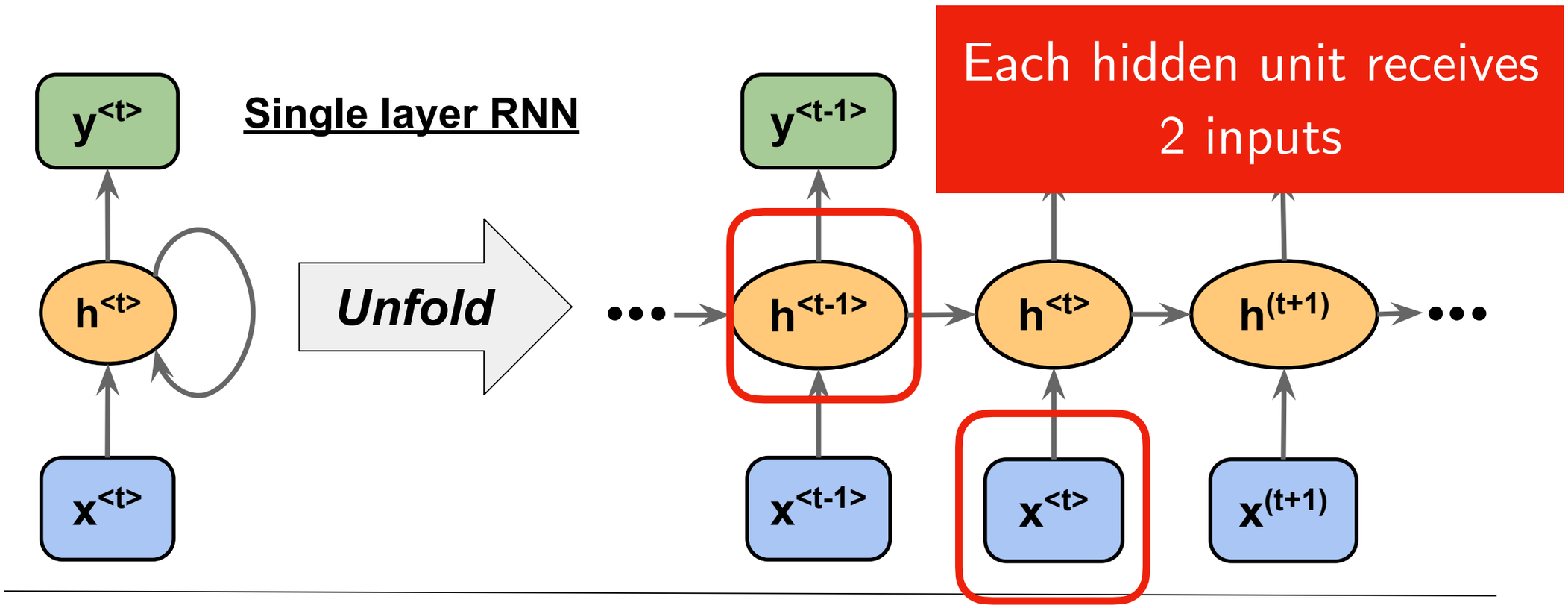


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019



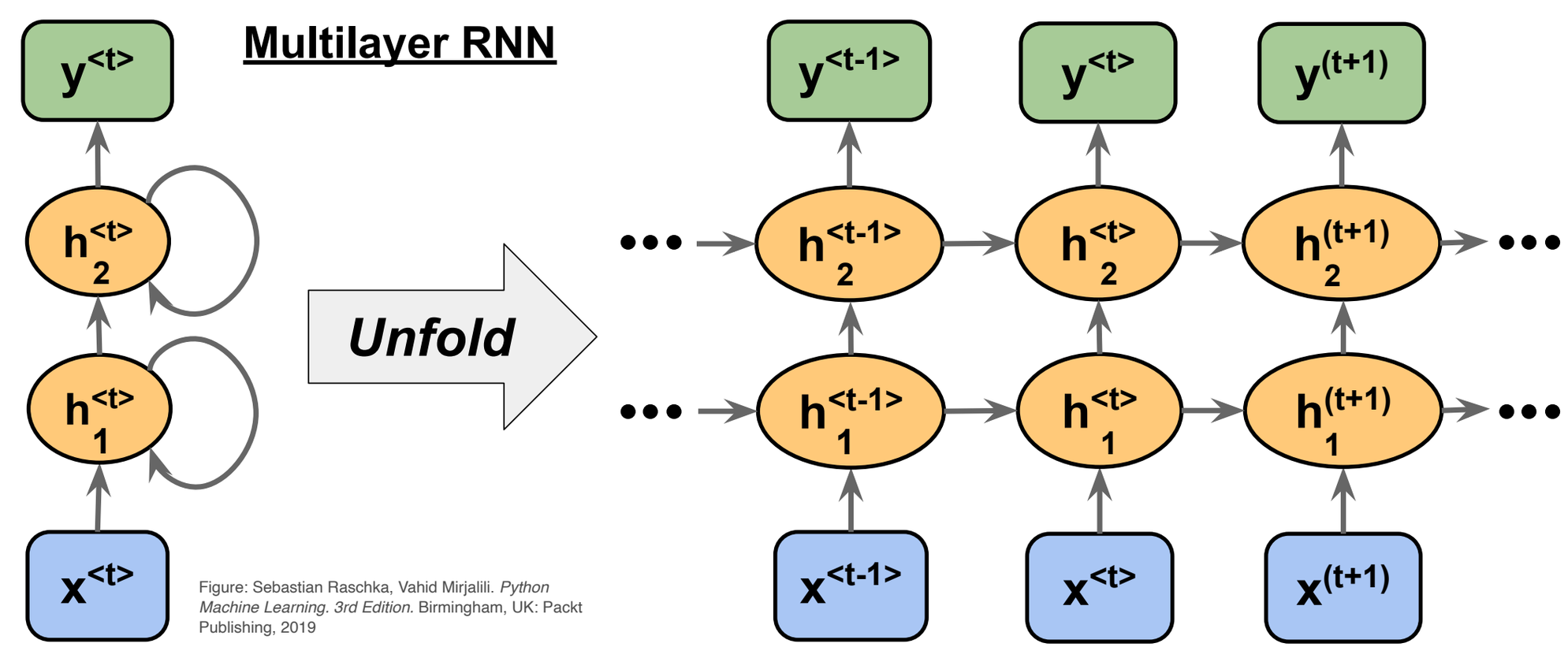


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Lecture Overview

RNNs and Sequence Modeling Tasks

Backpropagation Through Time

Long-short term memory (LSTM)

Many-to-one Word RNNs

Generating Text with Character RNNs

Attention Mechanisms and Transformers

Weight matrices in a single-hidden layer RNN

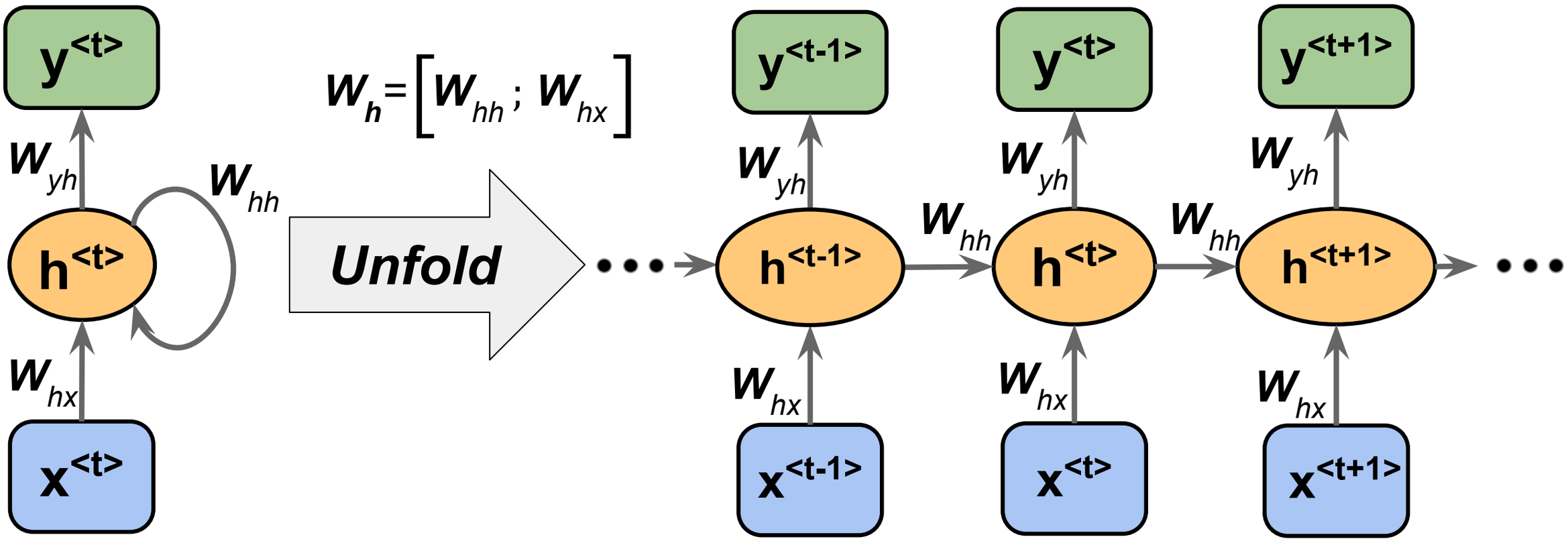


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Weight matrices in a single-hidden layer RNN

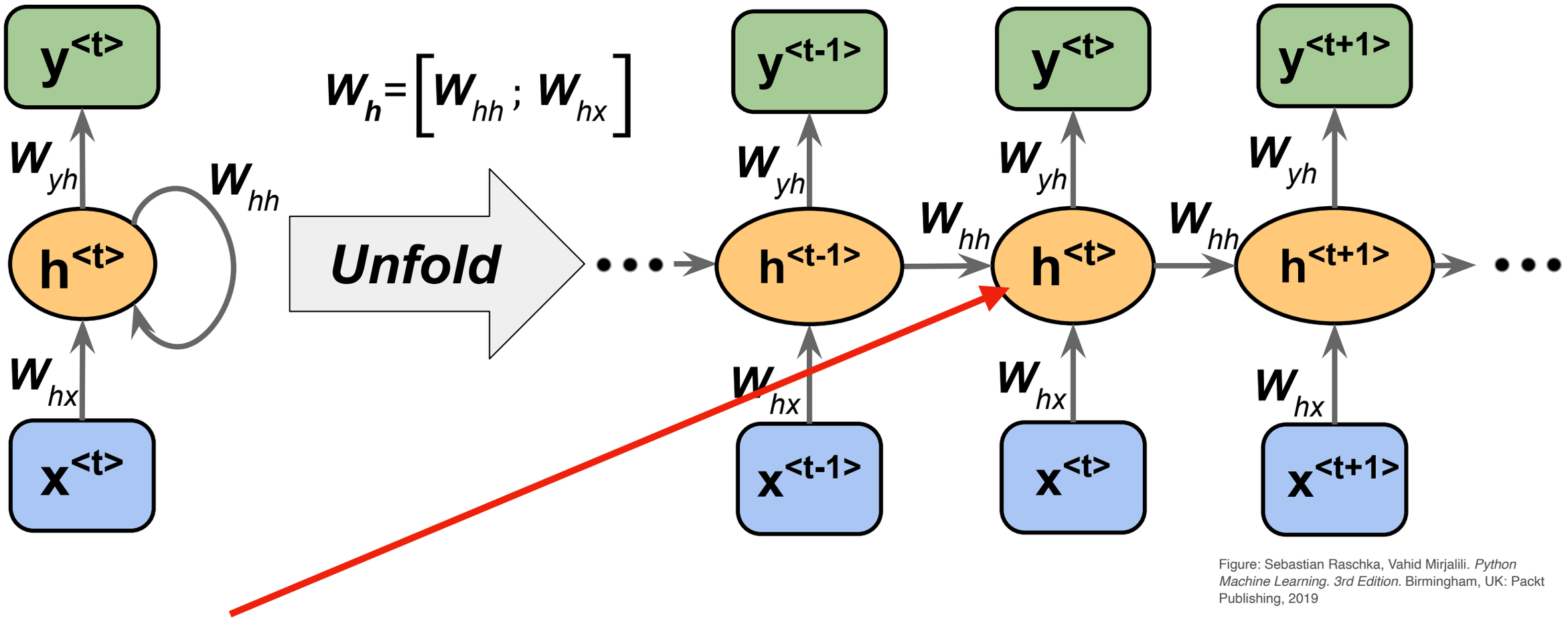


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Net input:

$$\mathbf{z}_h^{\langle t \rangle} = \mathbf{W}_{hx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{hh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{\langle t \rangle} = \sigma_h\left(\mathbf{z}_h^{\langle t \rangle}\right)$$

Weight matrices in a single-hidden layer RNN

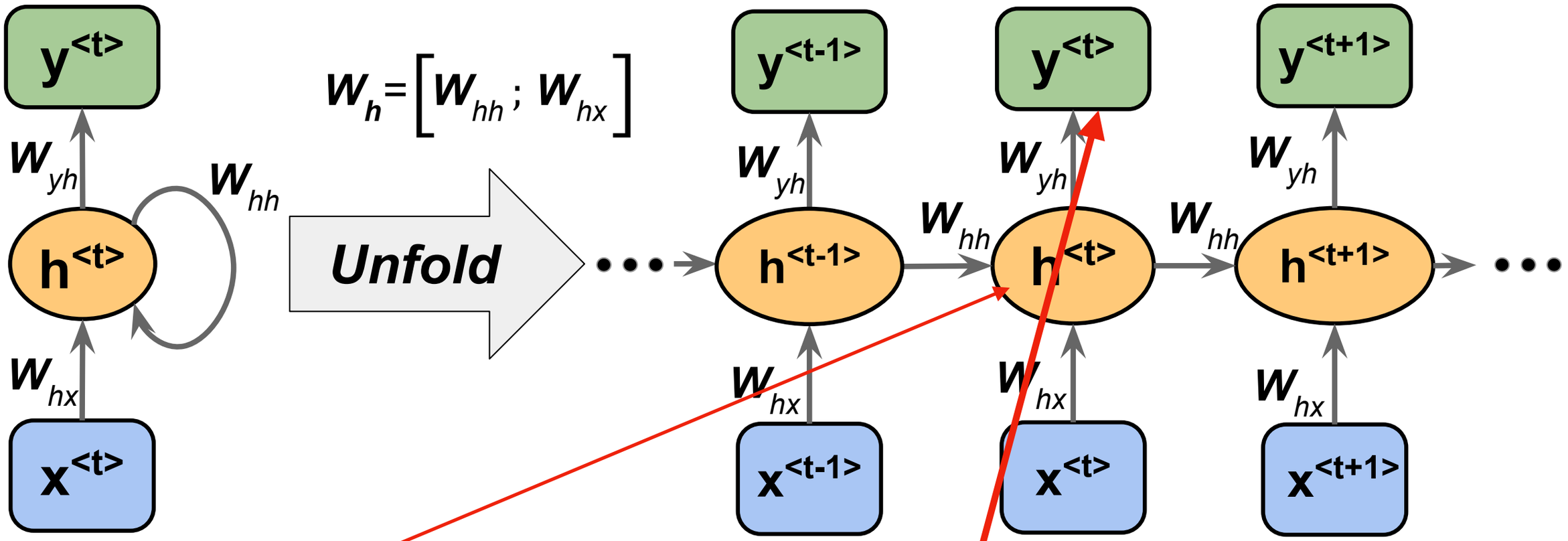


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Net input:

$$\mathbf{z}_h^{\langle t \rangle} = \mathbf{W}_{hx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{hh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{\langle t \rangle} = \sigma_h\left(\mathbf{z}_h^{\langle t \rangle}\right)$$

Net input:

$$\mathbf{z}_y^{\langle t \rangle} = \mathbf{W}_{yh}\mathbf{h}^{\langle t \rangle} + \mathbf{b}_y$$

Output:

$$\mathbf{y}^{\langle t \rangle} = \sigma_y\left(\mathbf{z}_y^{\langle t \rangle}\right)$$

Backpropagation through time

$$L = \sum_{t=1}^{T} L^{\langle t \rangle}$$

$L^{\langle t-1 \rangle}$ $L^{\langle t \rangle}$ $L^{\langle t+1 \rangle}$

$y^{<t>}$ $y^{<t-1>}$ $y^{<t>}$ $y^{<t+1>}$

W_{yh} W_{hh} W_{hx}

$h^{<t>}$ $h^{<t-1>}$ $h^{<t>}$ $h^{<t+1>}$

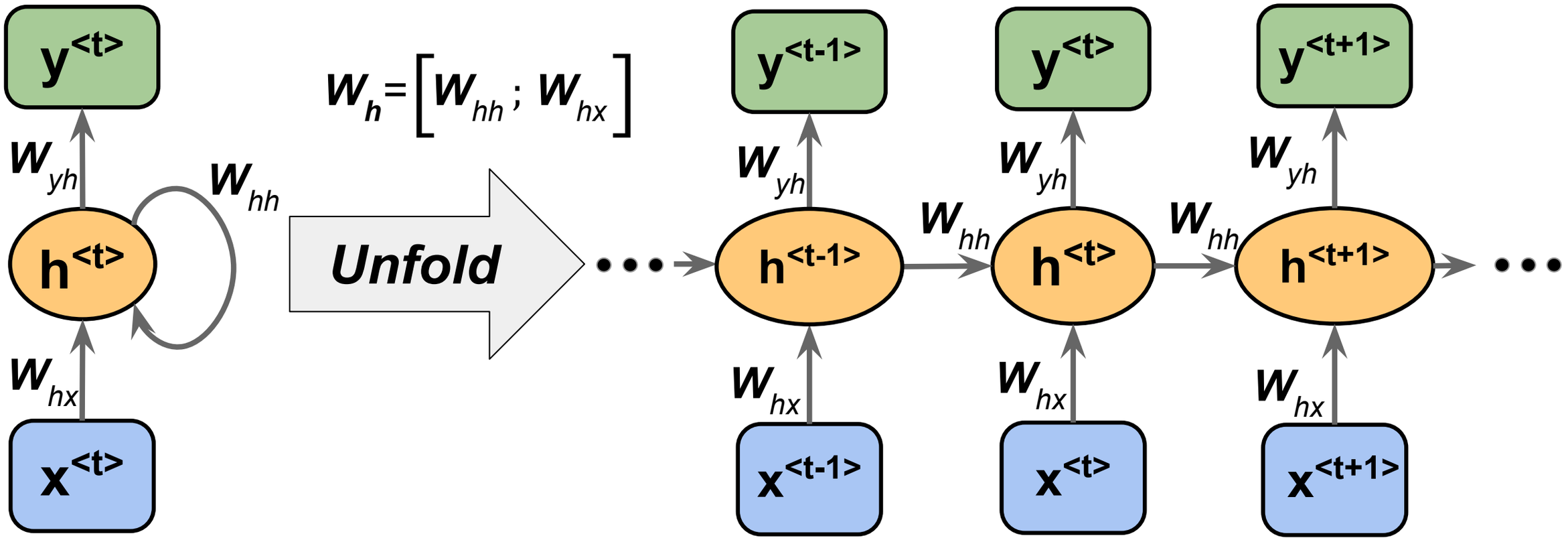
$x^{<t>}$ $x^{<t-1>}$ $x^{<t>}$ $x^{<t+1>}$

$W_h = [W_{hh}; W_{hx}]$

Unfold

Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Backpropagation through time

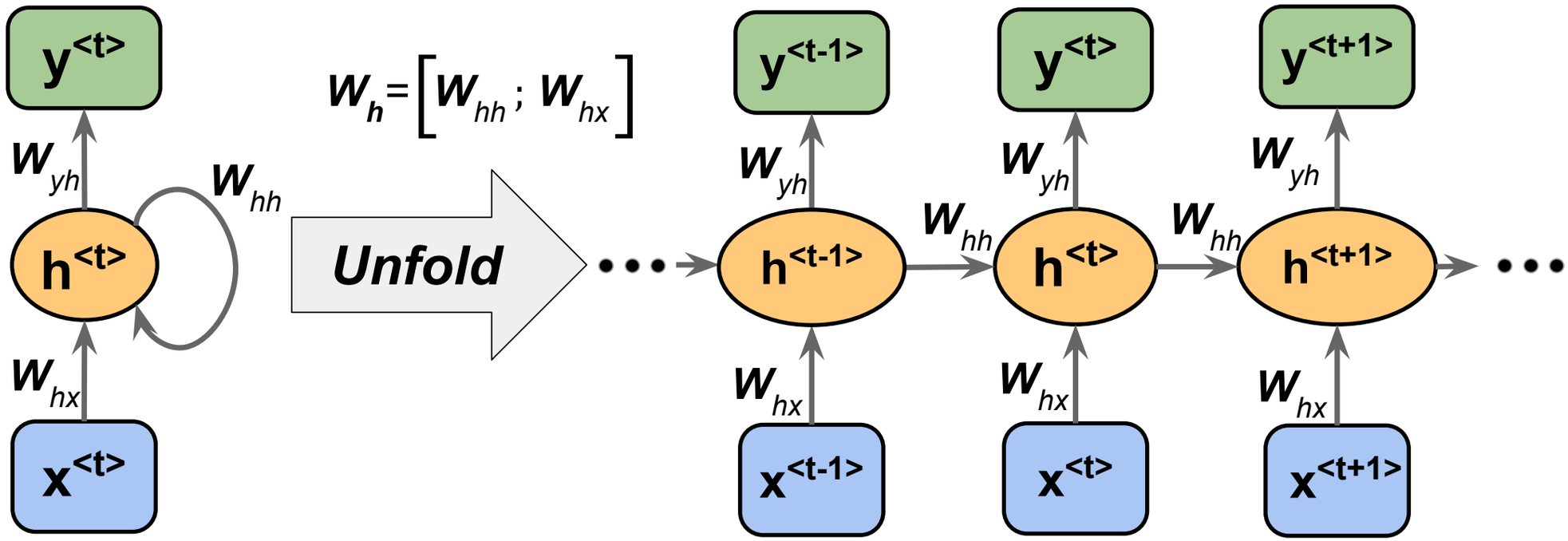


Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

The loss is computed as the sum over all time steps:

$$L = \sum_{t=1}^{T} L^{\langle t \rangle}$$

Backpropagation through time

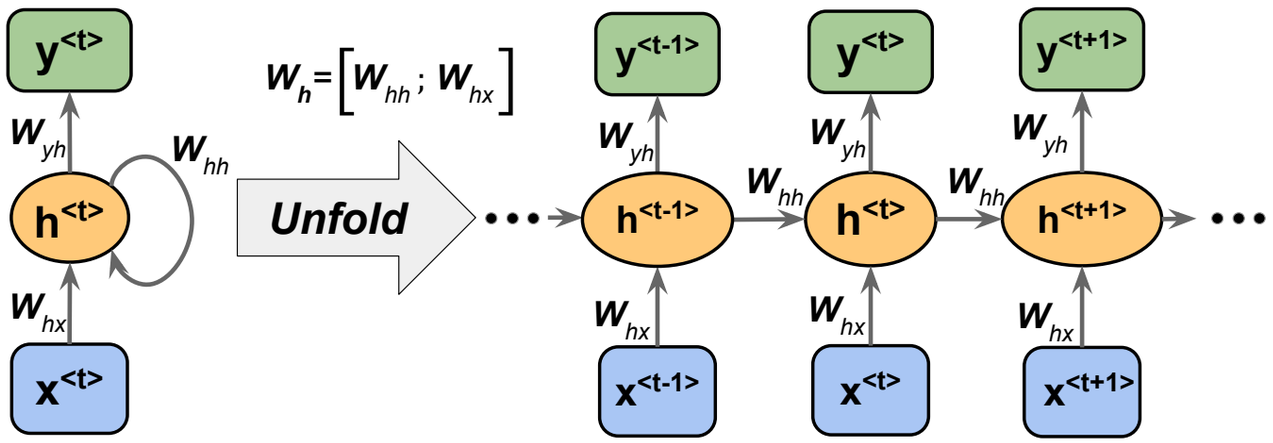


Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)}$$

$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

Backpropagation through time



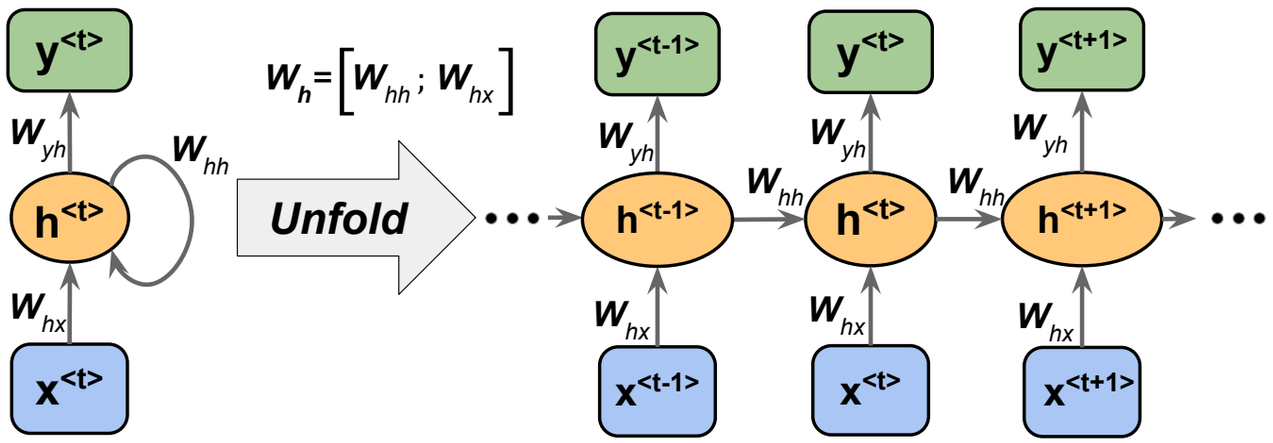
Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)} \qquad \frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

computed as a multiplication of adjacent time steps:

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

Backpropagation through time



Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)} \qquad \frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \boxed{\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

computed as a multiplication of adjacent time steps:

This is very problematic:
Vanishing/Exploding gradient problem!

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

A good resource that explains *backpropation through time* nicely:

Boden, Mikael. "A guide to recurrent neural networks and backpropagation." (2002).

Lecture Overview

RNNs and Sequence Modeling Tasks

Backpropagation Through Time

Long-short term memory (LSTM)

Many-to-one Word RNNs

Generating Text with Character RNNs

Attention Mechanisms and Transformers

Solutions to the vanishing/exploding gradient problems

1) Gradient Clipping: set a max value for gradients if they grow to large (solves only exploding gradient problem)

2) Truncated backpropagation through time (TBPTT)
   - simply limits the number of time steps the signal can backpropagate each forward pass. E.g., even if the sequence has 100 elements/steps, we may only backpropagate through 20 or so

3) Long short-term memory (LSTM) -- uses a memory cell for modeling long-range dependencies and avoid vanishing gradient problems

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780.

Long-short term memory (LSTM)

LSTM cell:

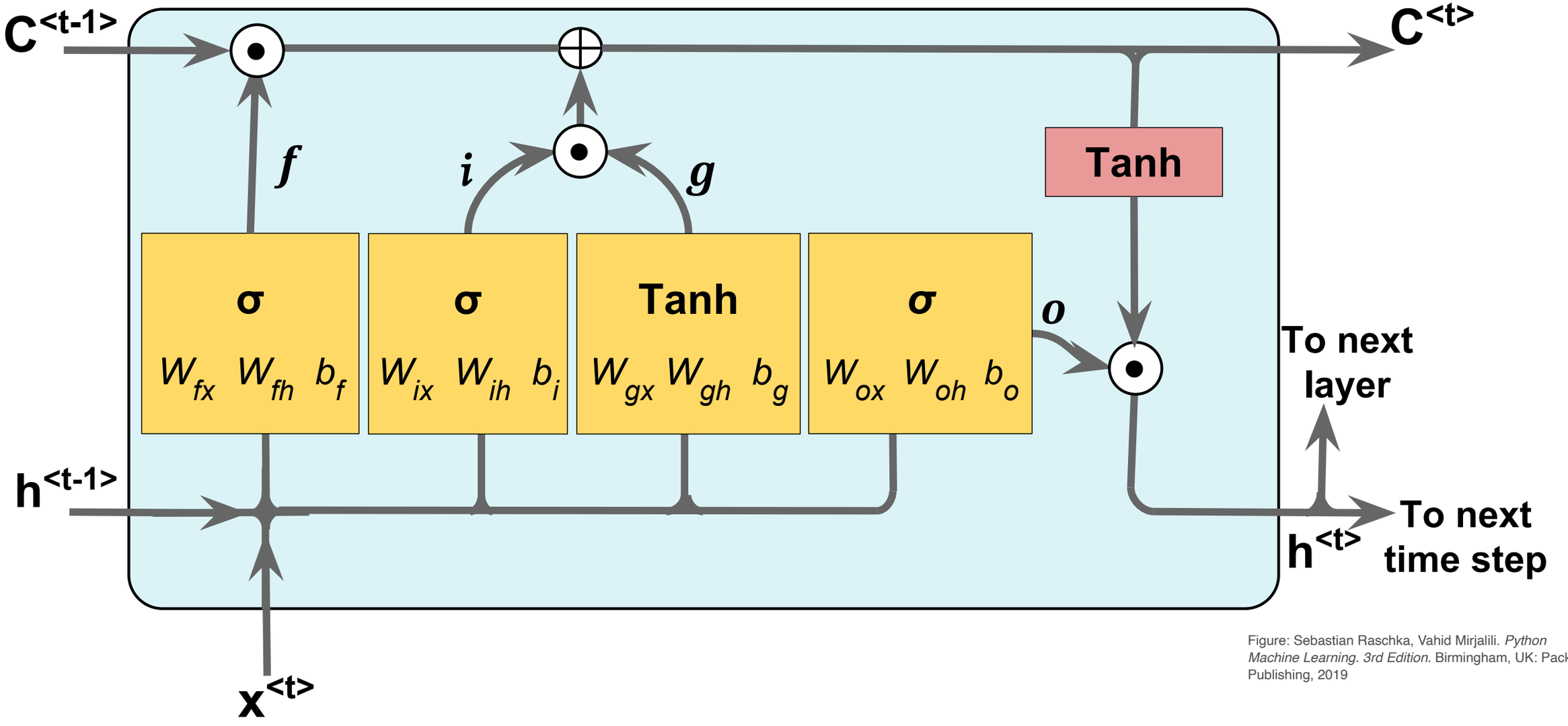
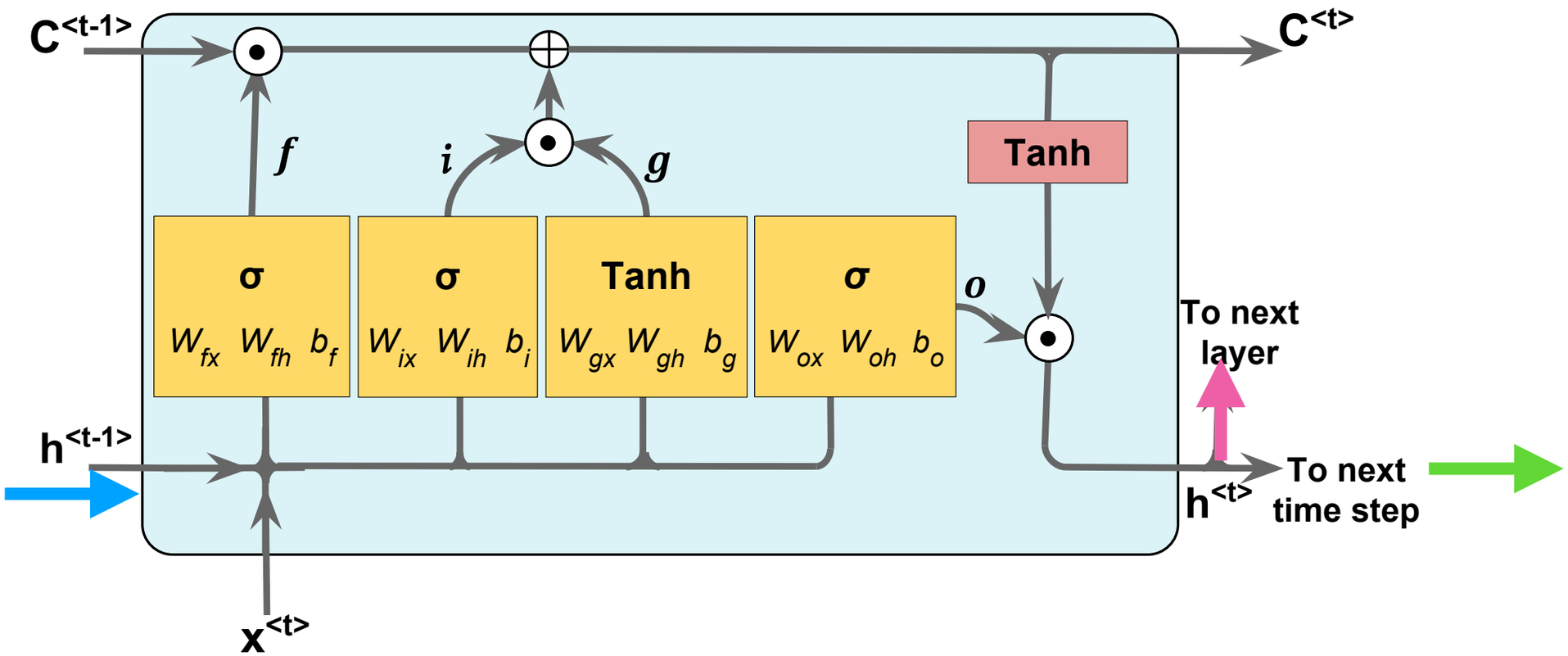
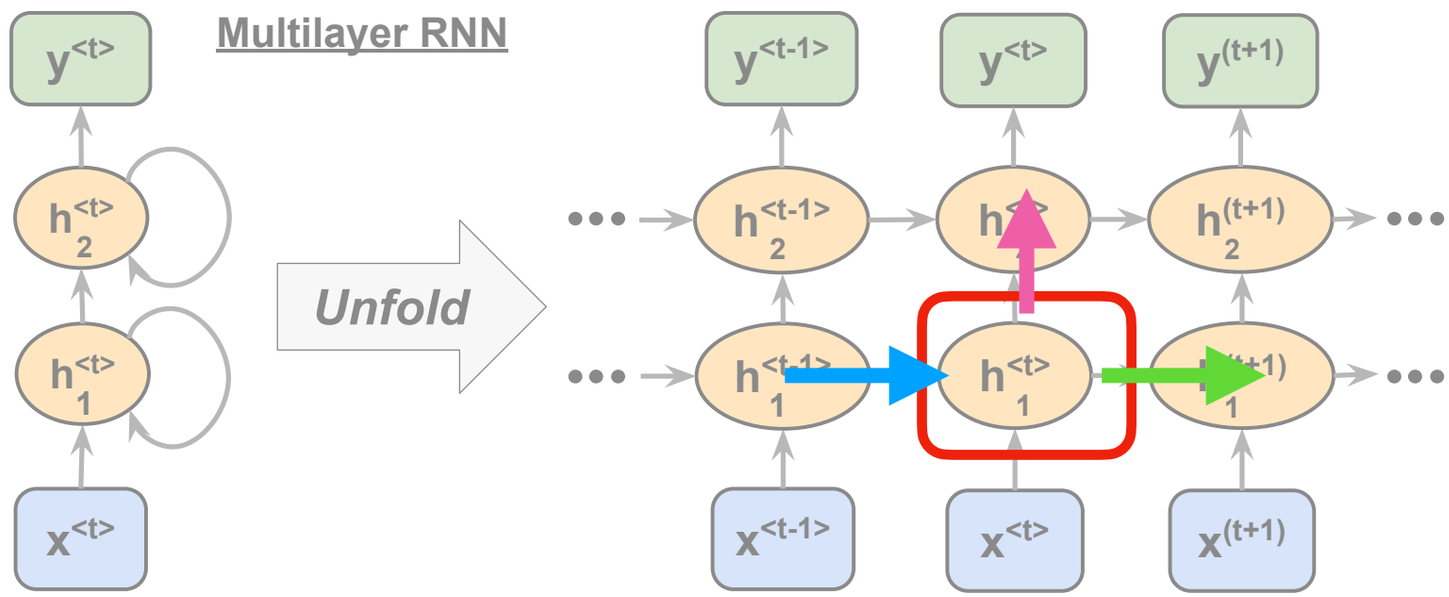


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

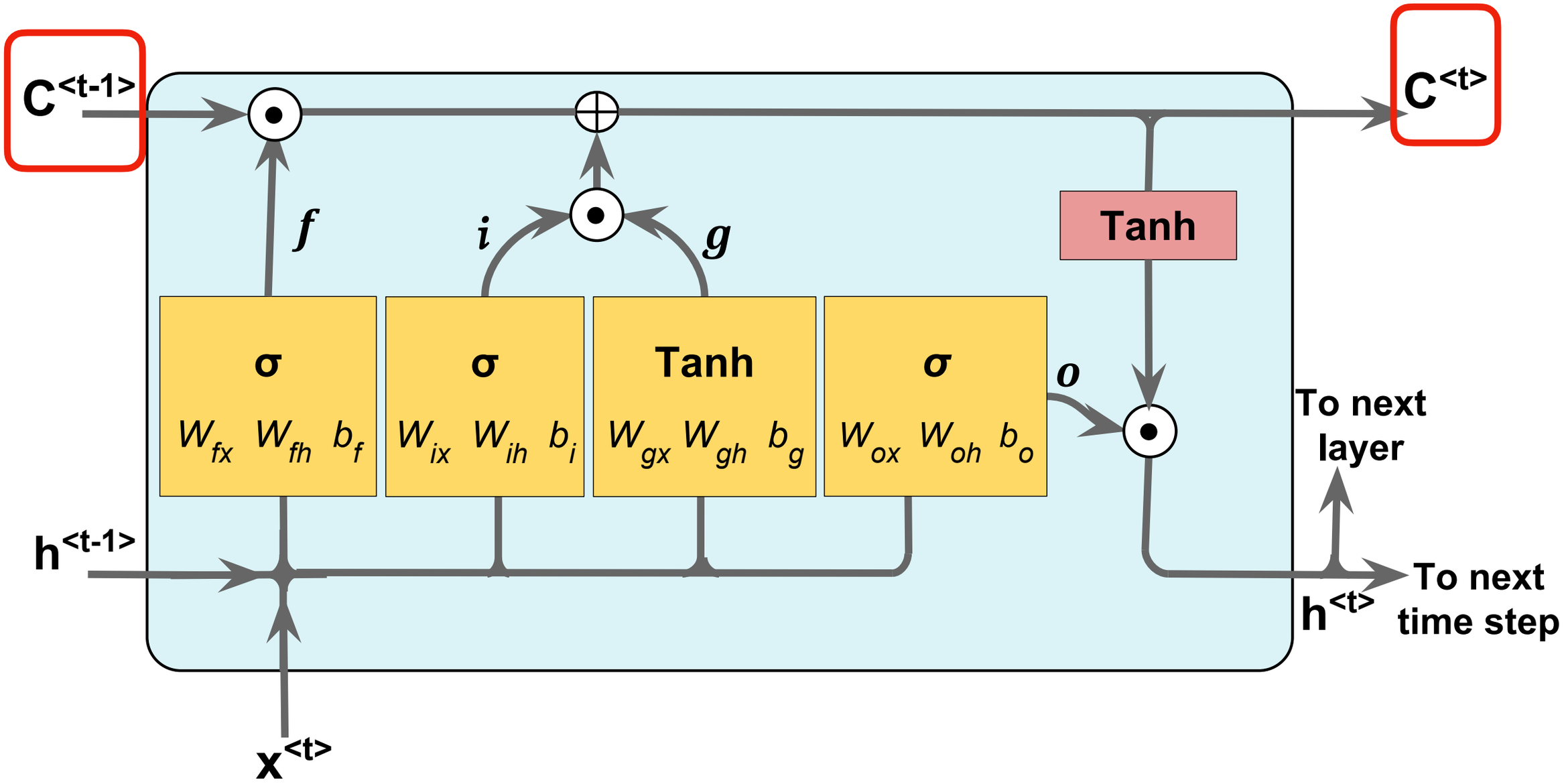

C<t-1>
C<t>
f
i
g
Tanh
σ
σ
Tanh
σ
o
W_fx W_fh b_f
W_ix W_ih b_i
W_gx W_gh b_g
W_ox W_oh b_o
To next layer
h<t-1>
To next time step
h<t>
x<t>


Multilayer RNN
y<t>
h2<t>
h1<t>
x<t>
Unfold
y<t-1>
y<t>
y(t+1)
h2<t-1>
h2(t+1)
h1<t-1>
h1<t>
h1(t+1)
x<t-1>
x<t>
x(t+1)

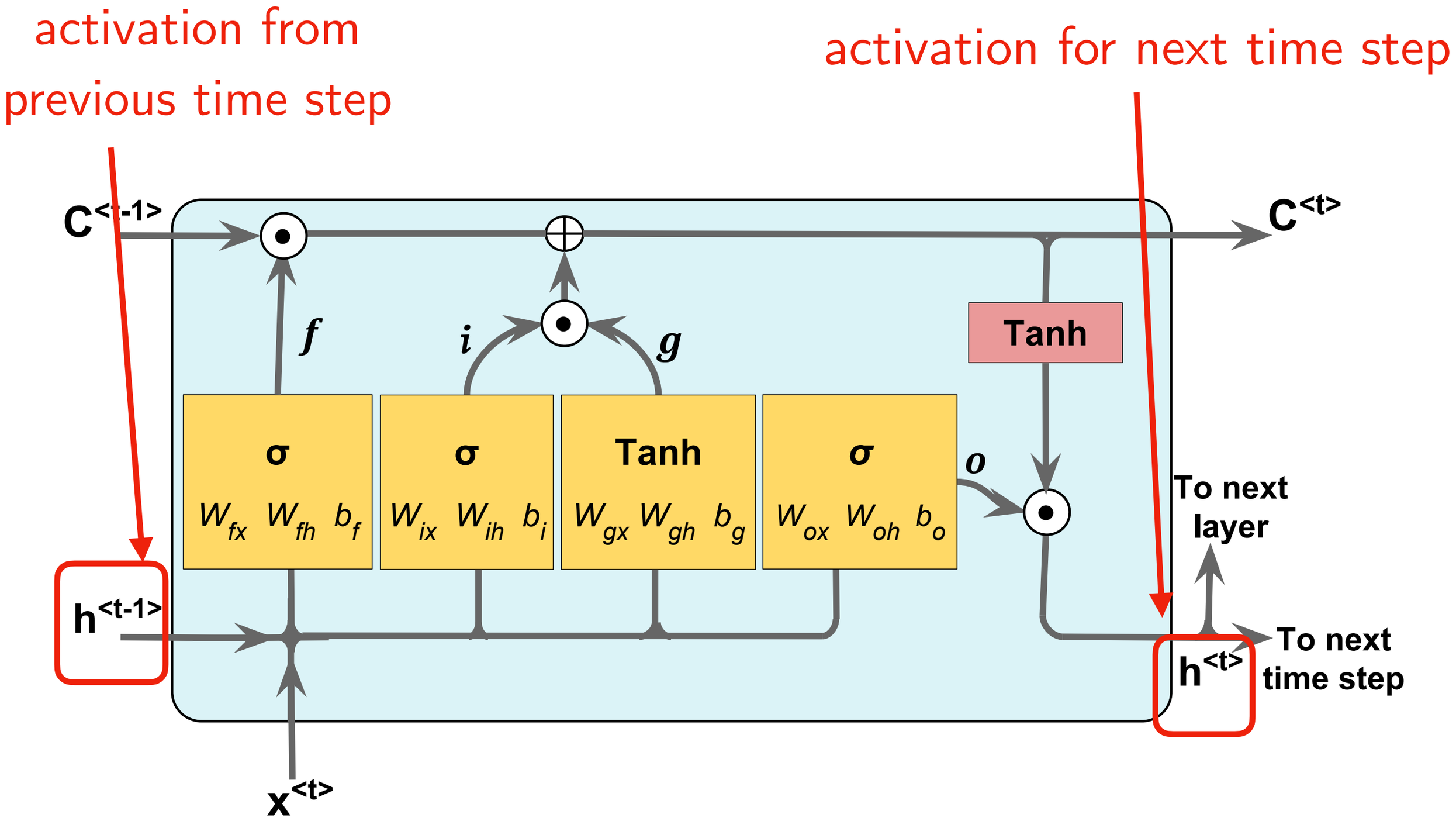
Long-short term memory (LSTM)

Cell state at previous time step

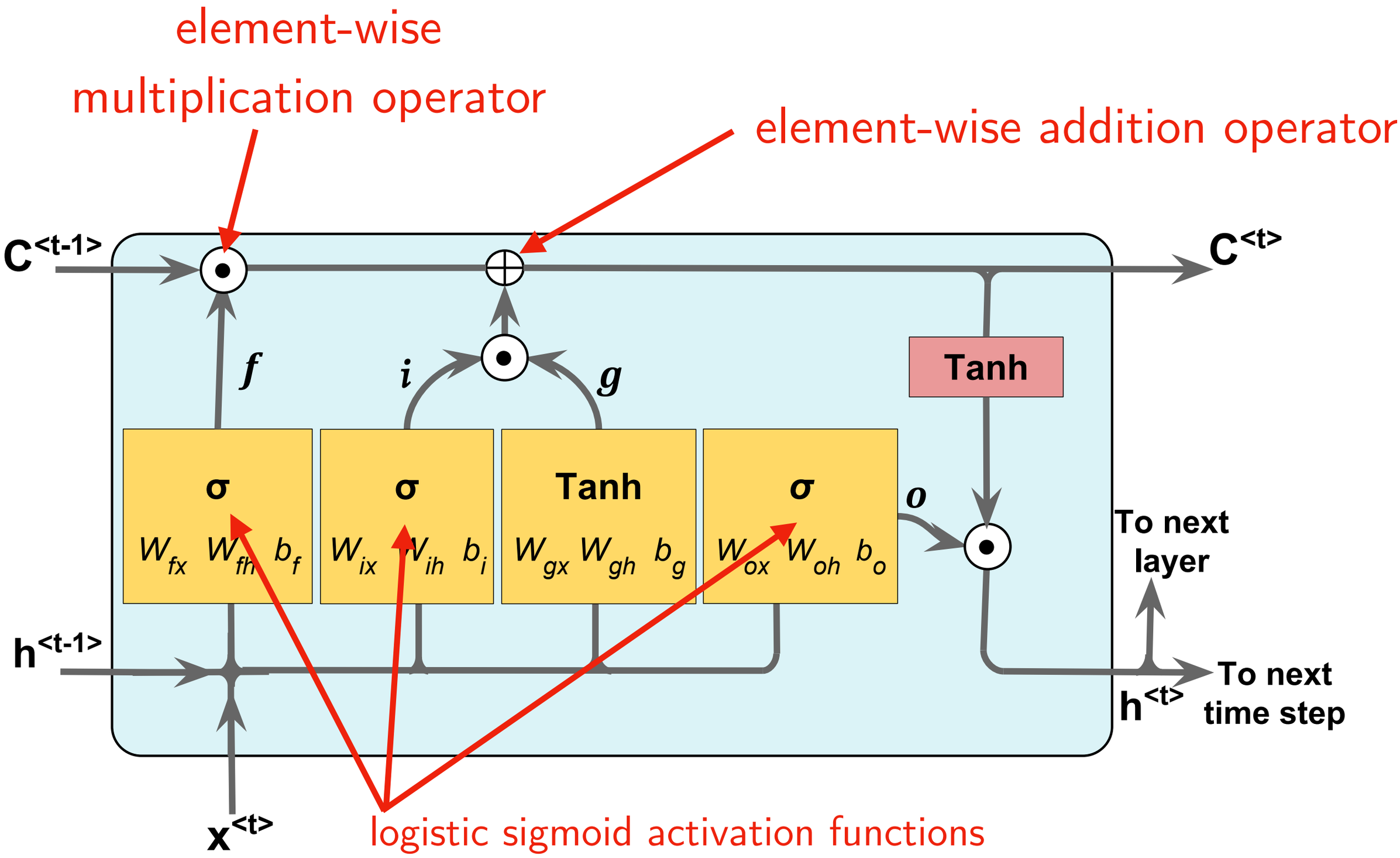
Cell state at current time step



Long-short term memory (LSTM)



Long-short term memory (LSTM)

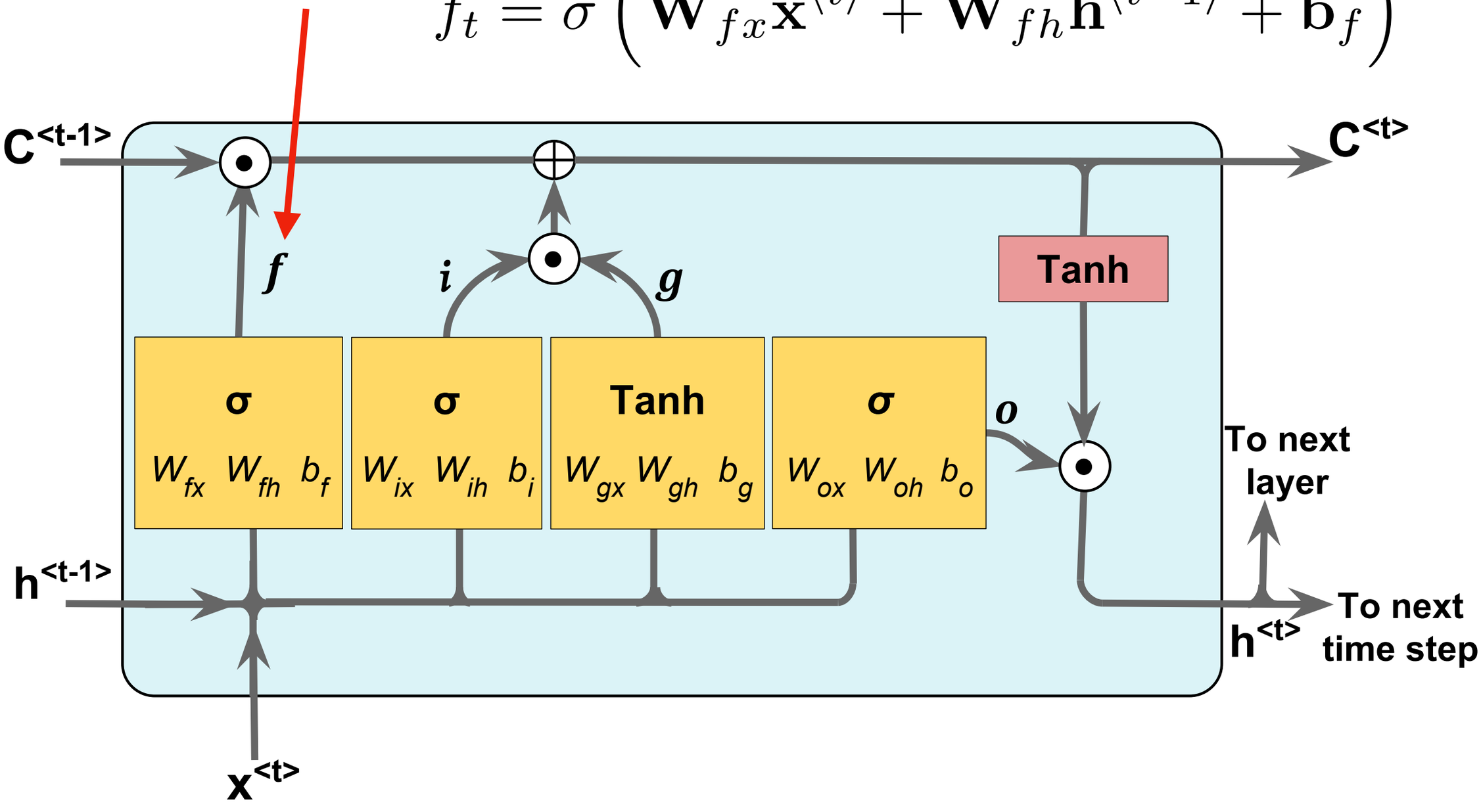


Long-short term memory (LSTM)

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

"Forget Gate": controls which information is remembered, and which is forgotten; can reset the cell state

$$f_t = \sigma\left(\mathbf{W}_{fx}\mathbf{x}^{\langle t\rangle} + \mathbf{W}_{fh}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_f\right)$$

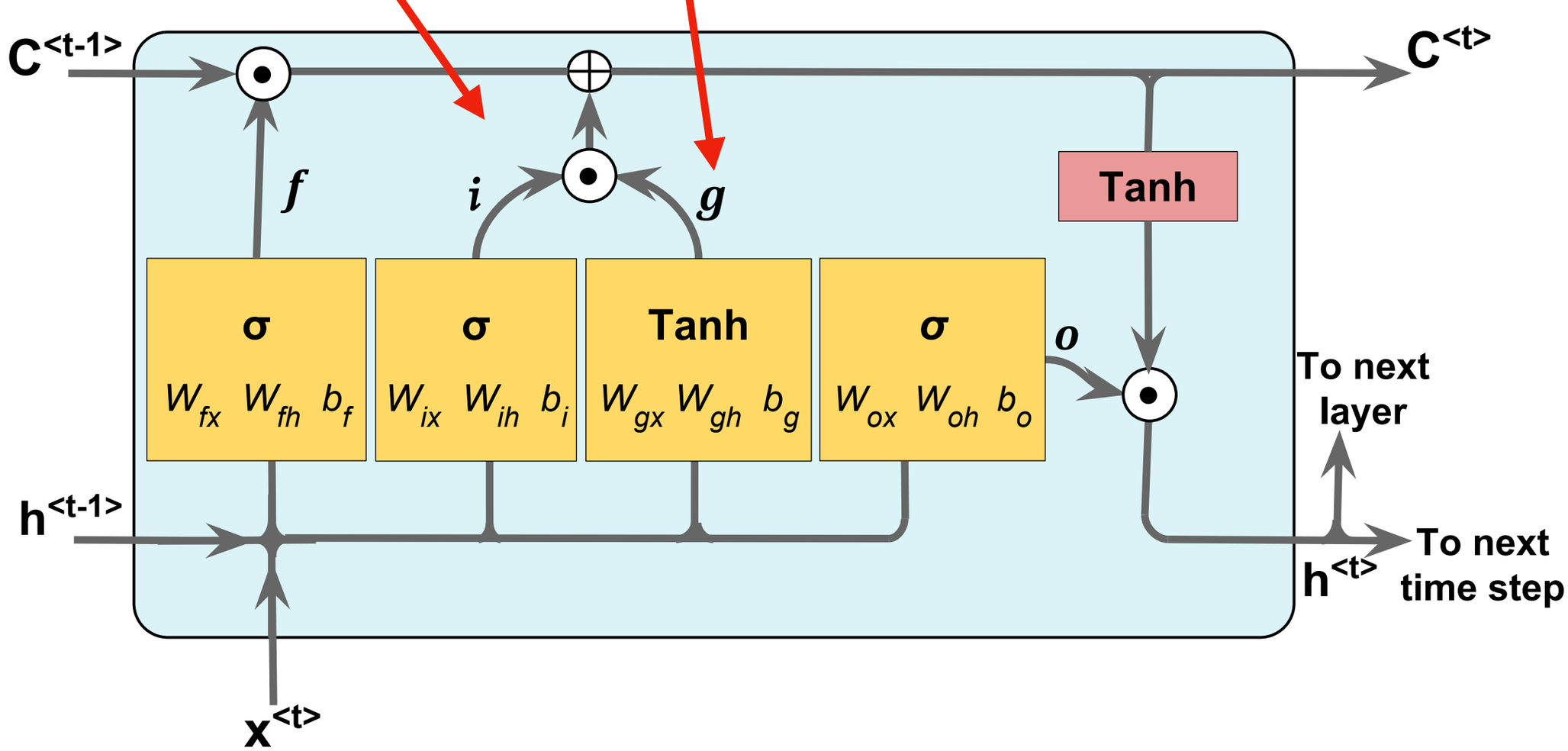


Long-short term memory (LSTM)

"Input Gate": $\mathbf{i}_t = \sigma\left(\mathbf{W}_{ix}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{ih}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_i\right)$

"Input Node":

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{gx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{gh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_g\right)$$



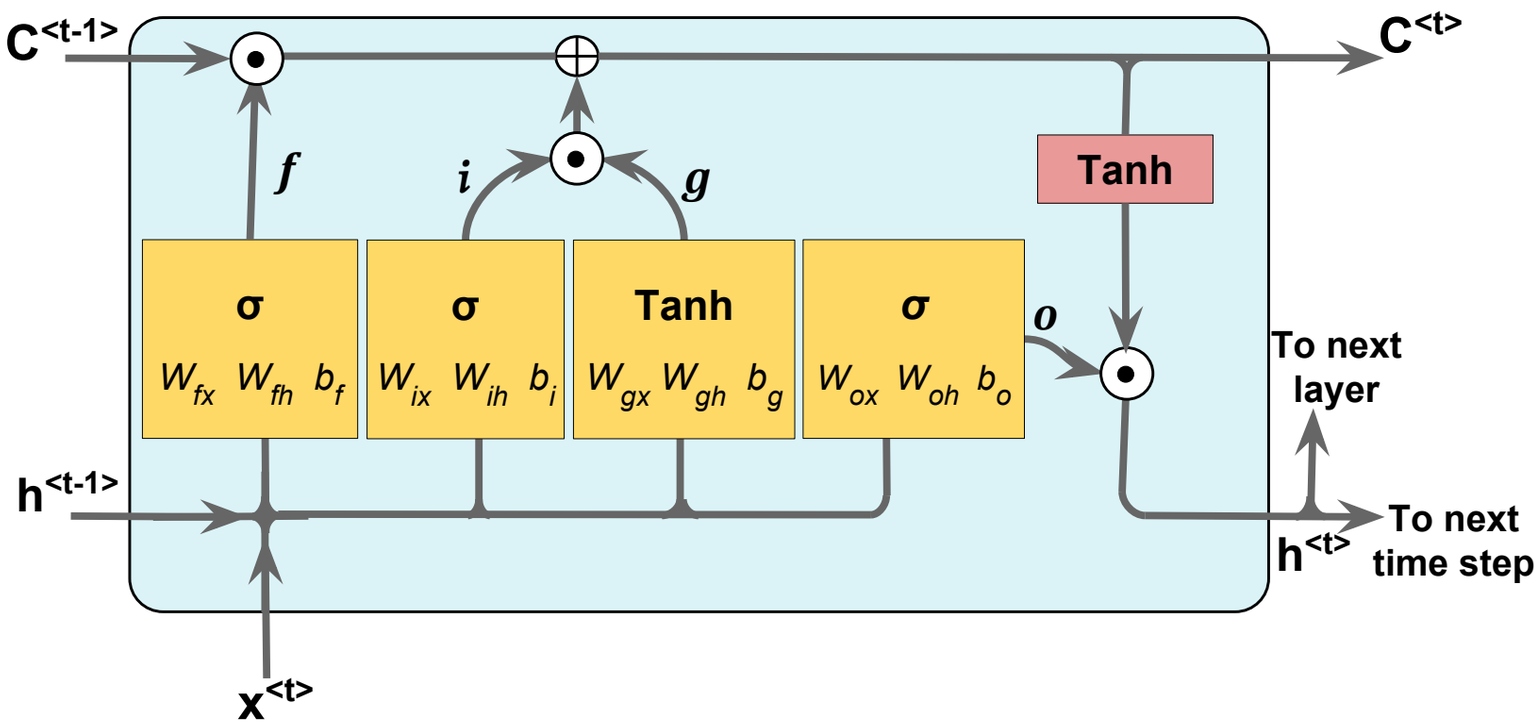
Long-short term memory (LSTM)

Brief summary of the gates so far ...

Forget Gate Input Node Input Gate

$$C^{\langle t \rangle} = \Big(C^{\langle t-1 \rangle} \odot f_t \Big) \oplus (i_t \odot g_t)$$

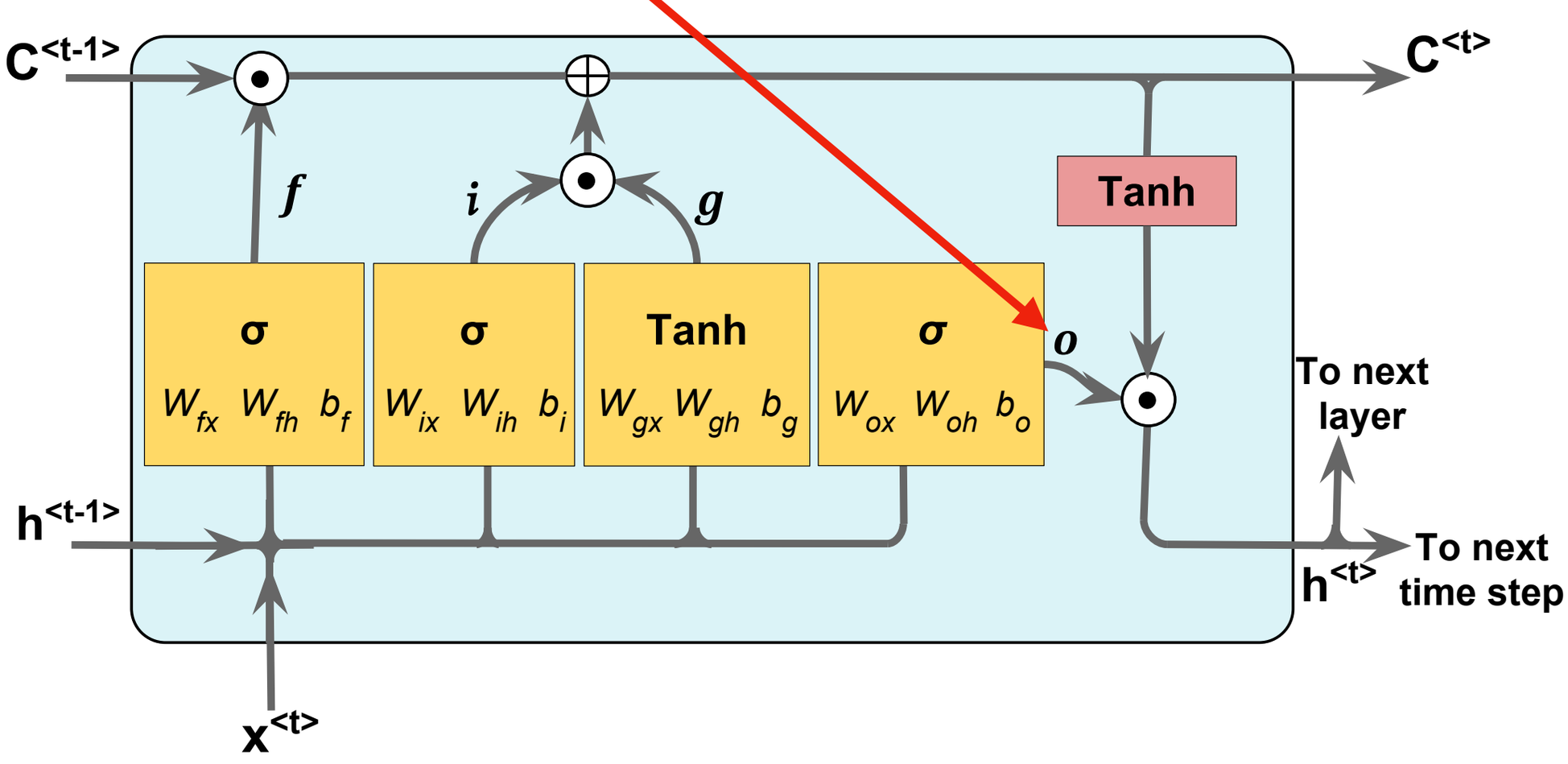
For updating the cell state



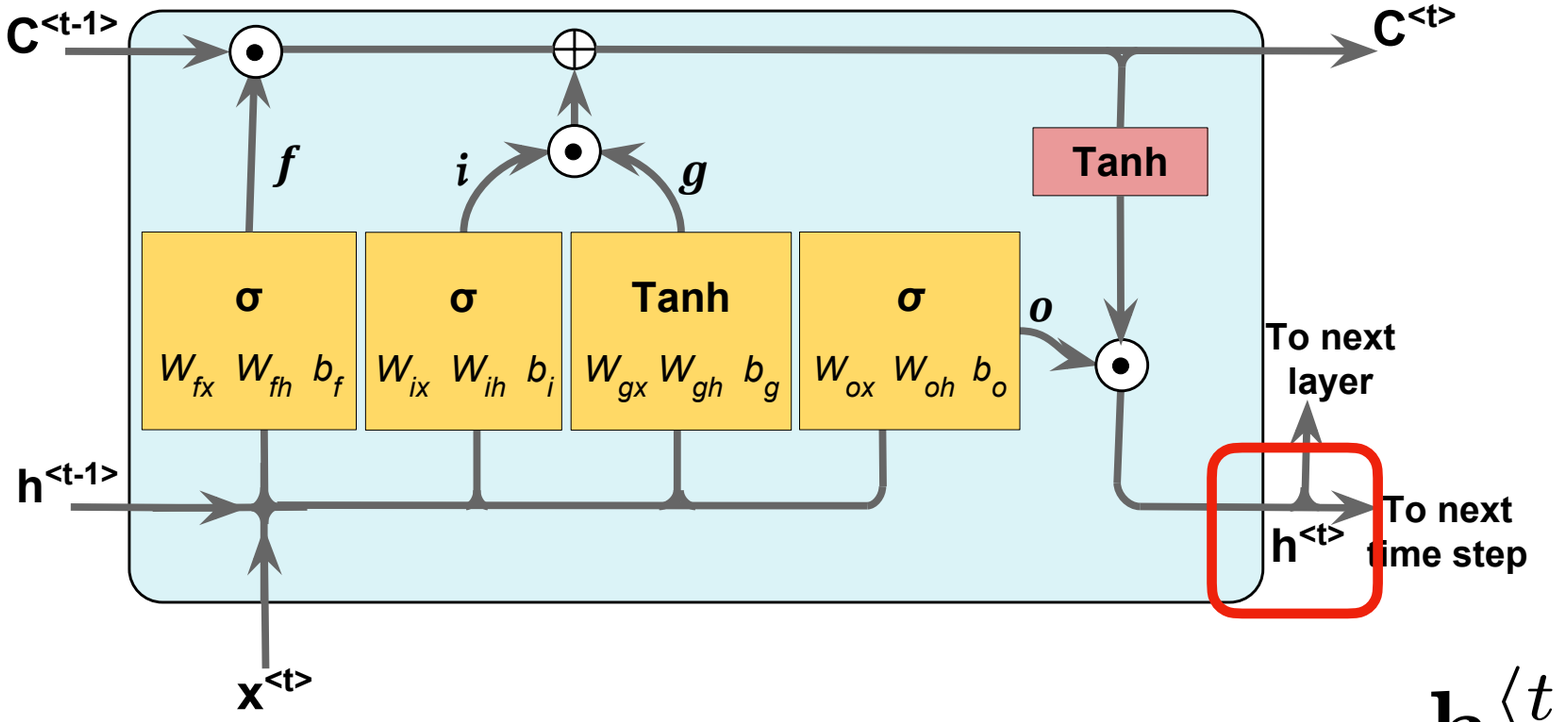
Long-short term memory (LSTM)

Output gate for updating the values of hidden units:

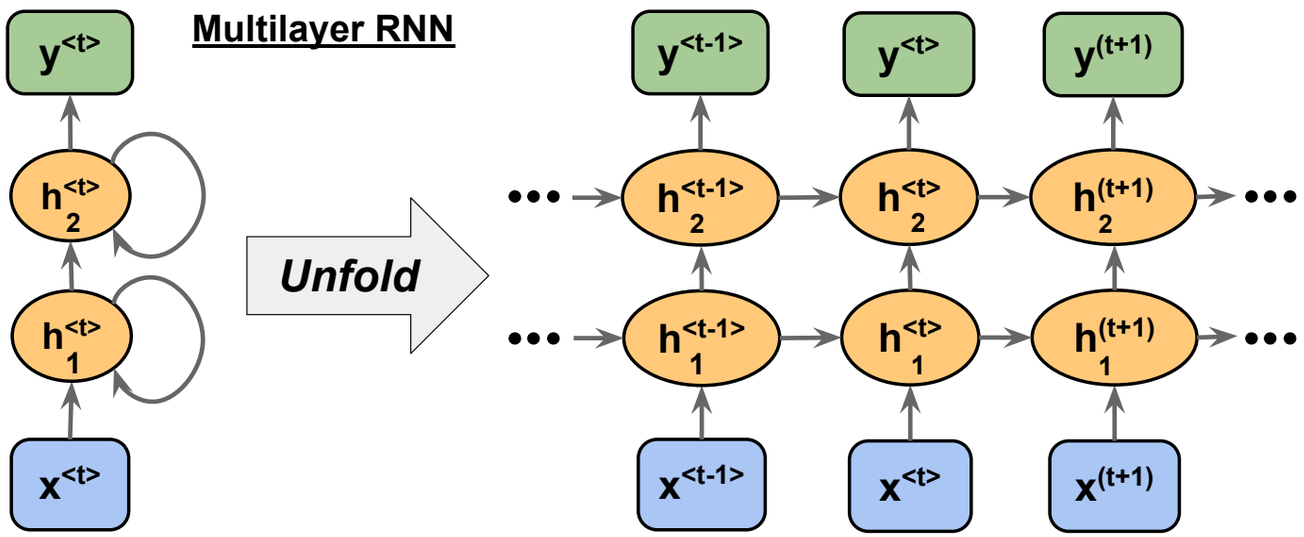
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{ox}\mathbf{x}^{\langle t\rangle} + \mathbf{W}_{oh}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_o\right)$$



Long-short term memory (LSTM)



$$\mathbf{h}^{\langle t \rangle} = \mathbf{o}_t \odot \tanh\left(\mathbf{C}^{\langle t \rangle}\right)$$

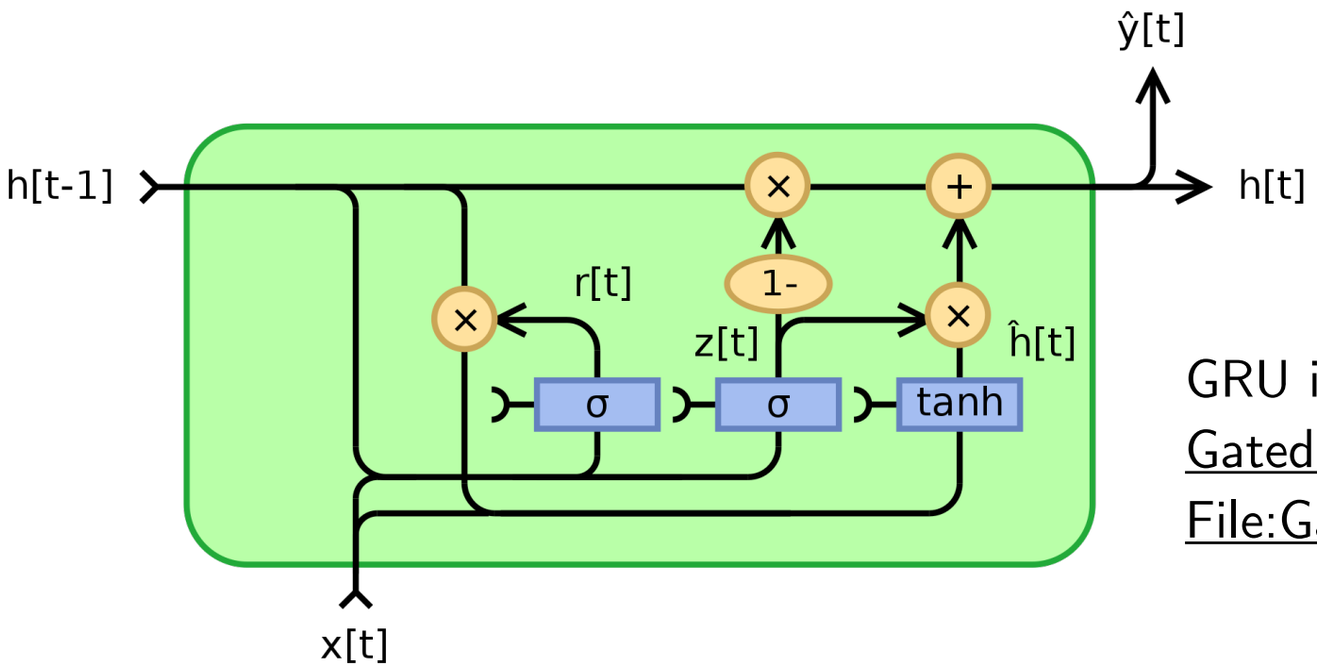


Long-short term memory (LSTM)

- Still popular and widely used today
- A recent, related approach is the Gated Recurrent Unit (GRU)

  Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).
- Nice article exploring LSTMs and comparing them to GRUs

  Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." In *International Conference on Machine Learning*, pp. 2342-2350. 2015.



GRU image source: https://en.wikipedia.org/wiki/Gated_recurrent_unit#/media/File:Gated_Recurrent_Unit,_base_type.svg

RNNs with LSTMs in PyTorch

Conceptually simple, the (very) hard part is the data processing pipeline

```
...
    self.rnn = torch.nn.LSTMCell(input_size, hidden_size)
...


def forward(self, x):
    embedded = self.embedding(text)
    h = self.initial_hidden_state()
    for input in x:
        h = self.rnn(input, h)
```

```
...
    self.rnn = torch.nn.LSTM(input_size, hidden_size)
...


def forward(self, x):
    h_0 = self.initial_hidden_state()
    output, h = self.rnn(x, h_0)
```

These two are equivalent, but the bottom one is substantially faster

Lecture Overview

RNNs and Sequence Modeling Tasks

Backpropagation Through Time

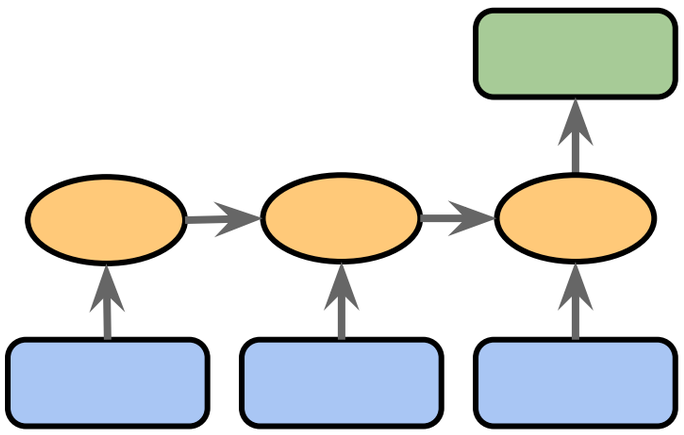
Long-short term memory (LSTM)

Many-to-one Word RNNs

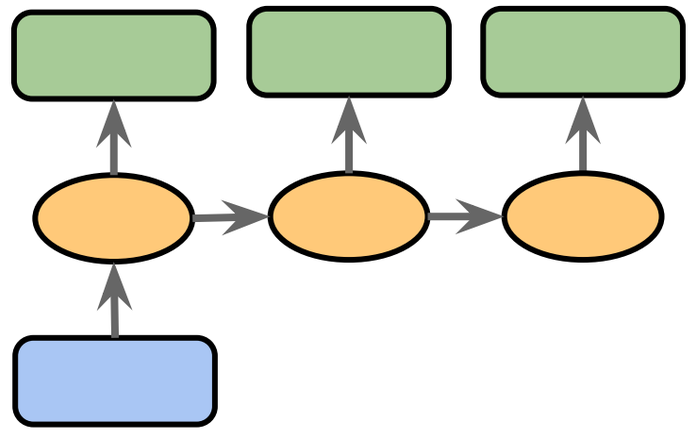
Generating Text with Character RNNs

Attention Mechanisms and Transformers

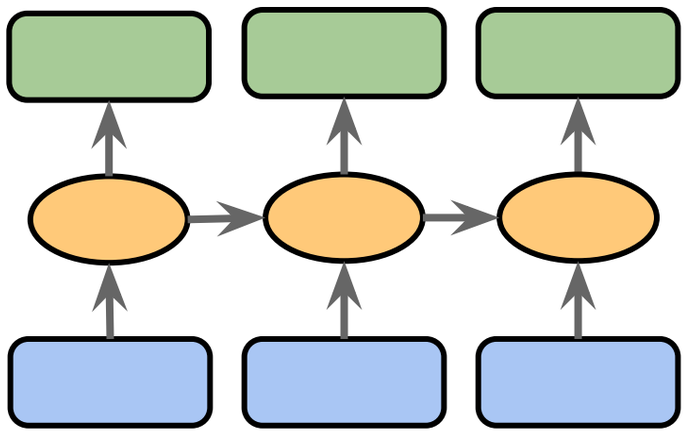
Different Types of Sequence Modeling Tasks



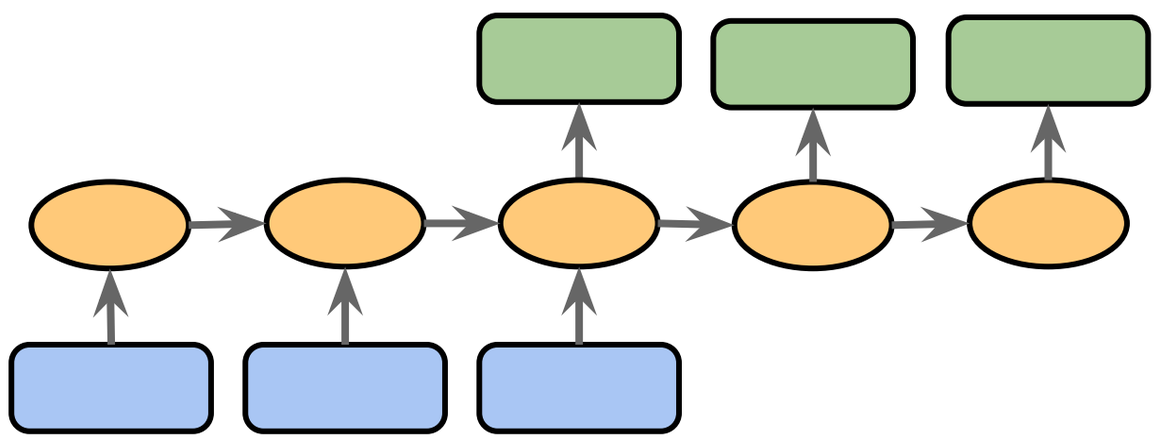
many-to-one



one-to-many



many-to-many



many-to-many

Figure based on:

The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy (http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

Many-to-One

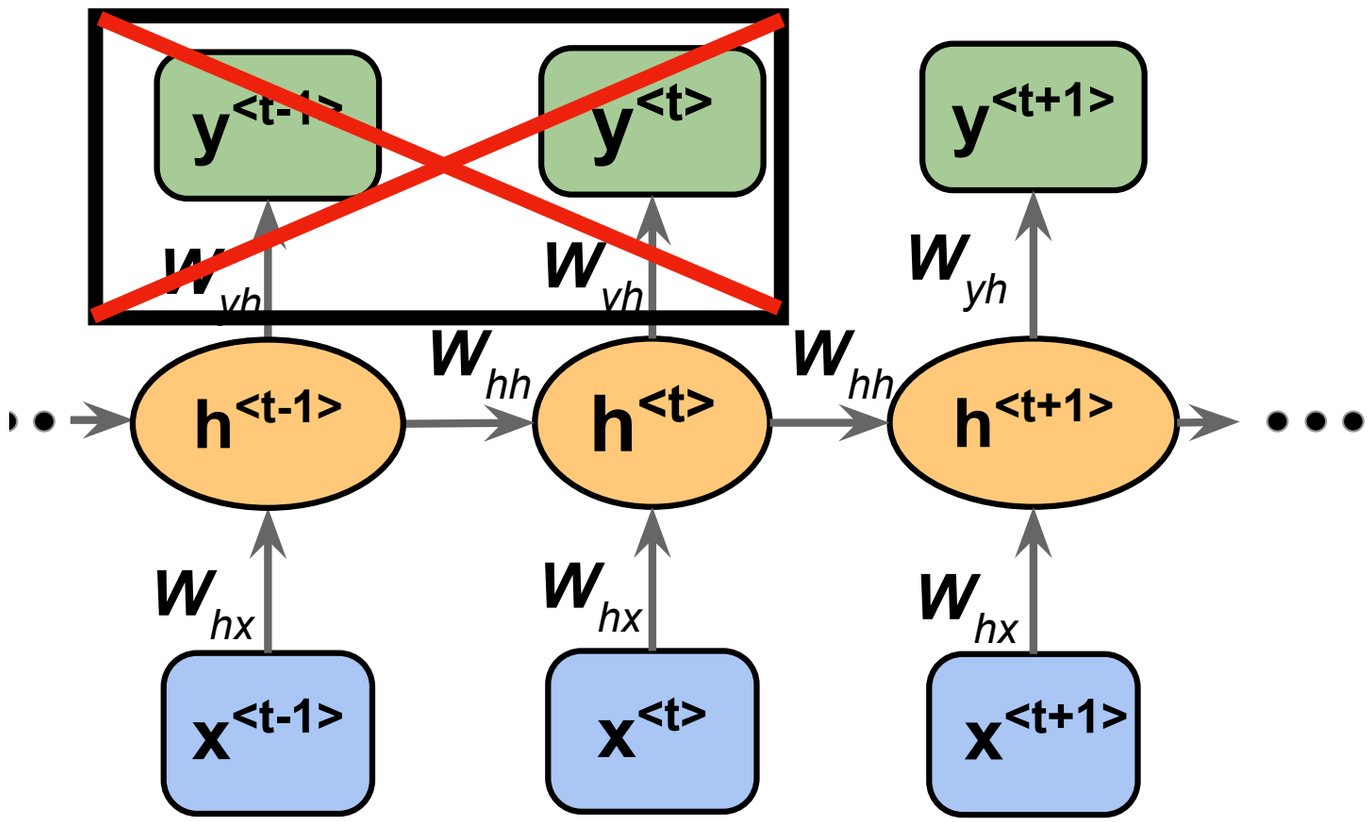


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Lecture Overview

RNNs and Sequence Modeling Tasks

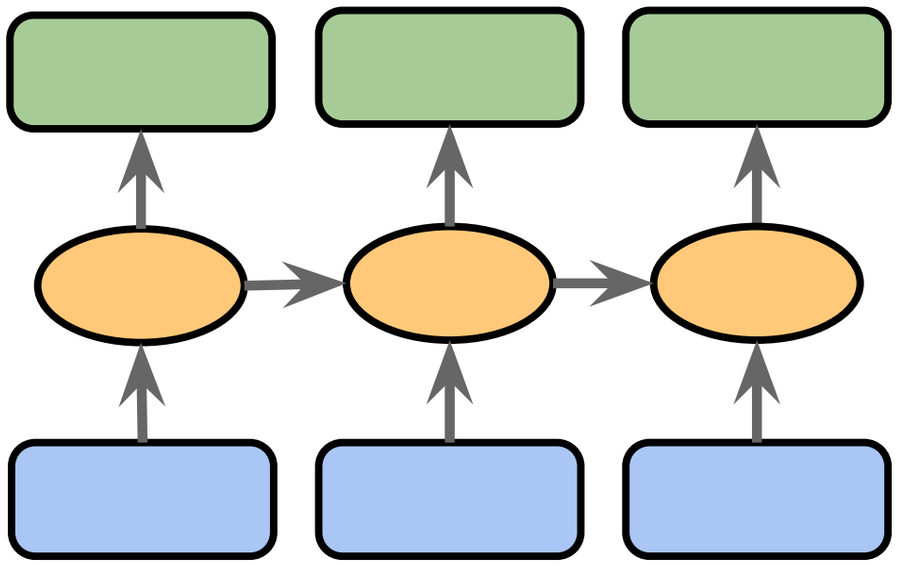
Backpropagation Through Time

Long-short term memory (LSTM)

Many-to-one Word RNNs

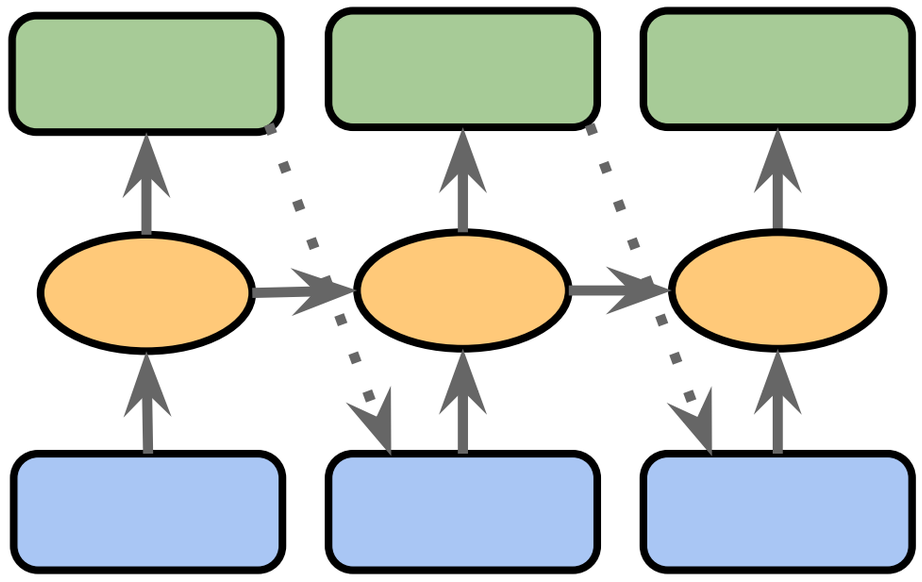
Generating Text with Character RNNs

Attention Mechanisms and Transformers



many-to-many

"training"

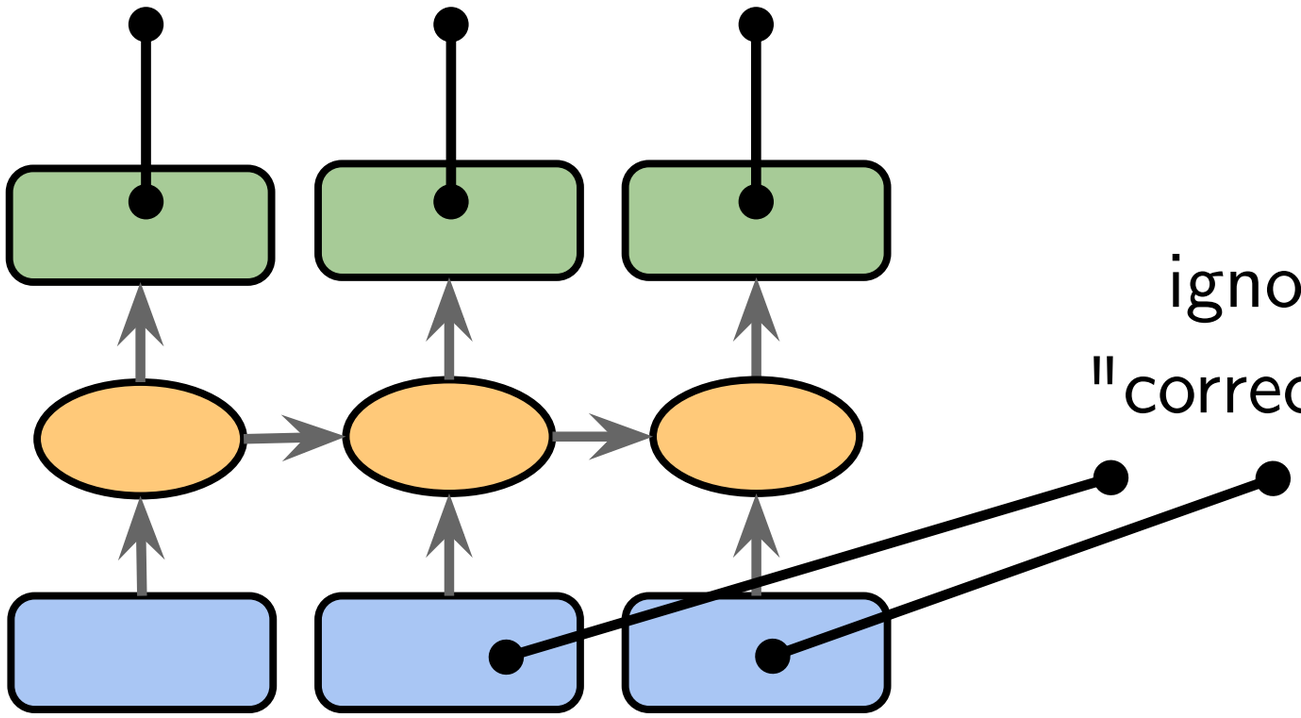


~~many~~-to-many

"one"

"generating new text"

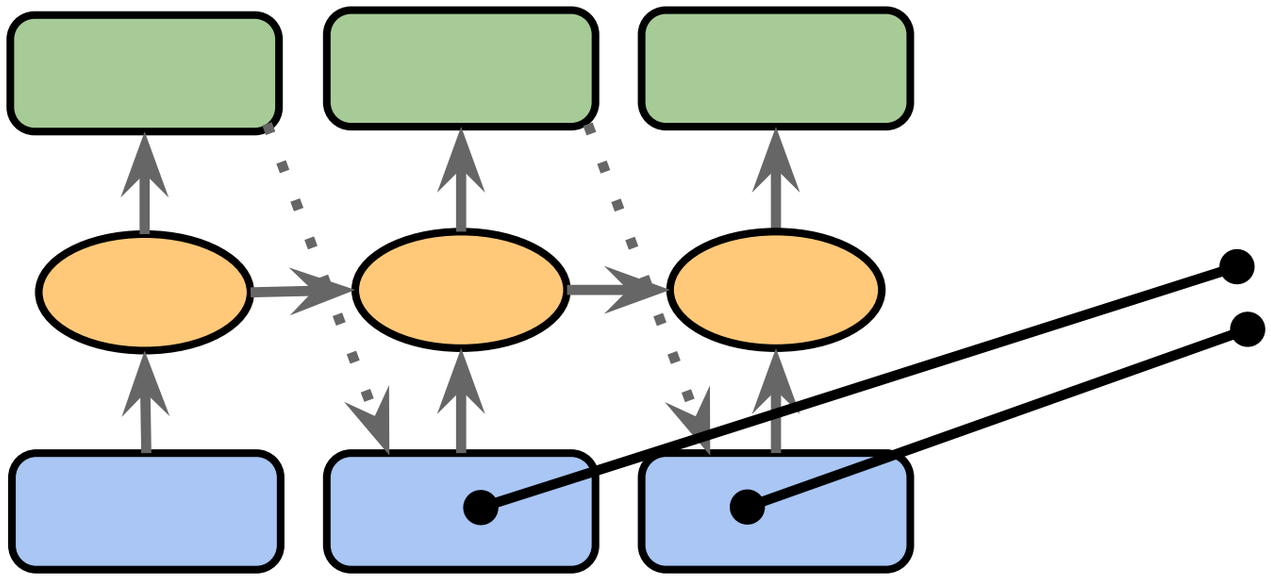
At each time step
Softmax output (probability)
for each possible"next letter"



many-to-many

"training"

For the next input,
ignore the prediction but use the
"correct" next letter from the dataset

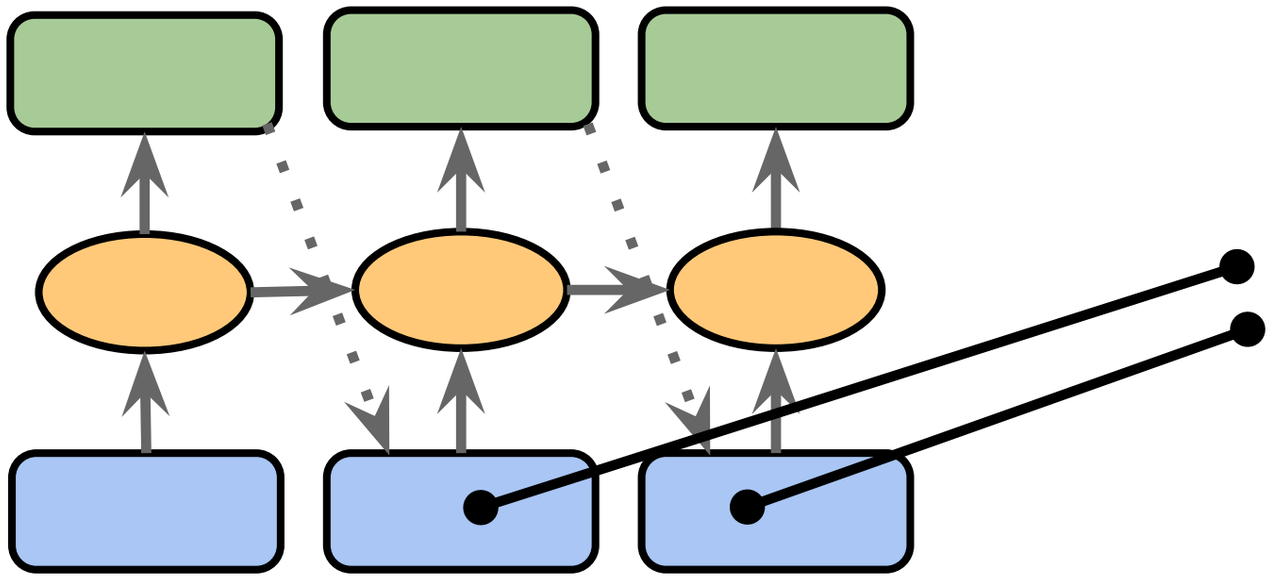


To generate new text, now, sample from the softmax outputs and provide the letter as input to the next time step

~~many~~-to-many

"one"

"generating new text"



To generate new text, now, sample from the softmax outputs and provide the letter as input to the next time step

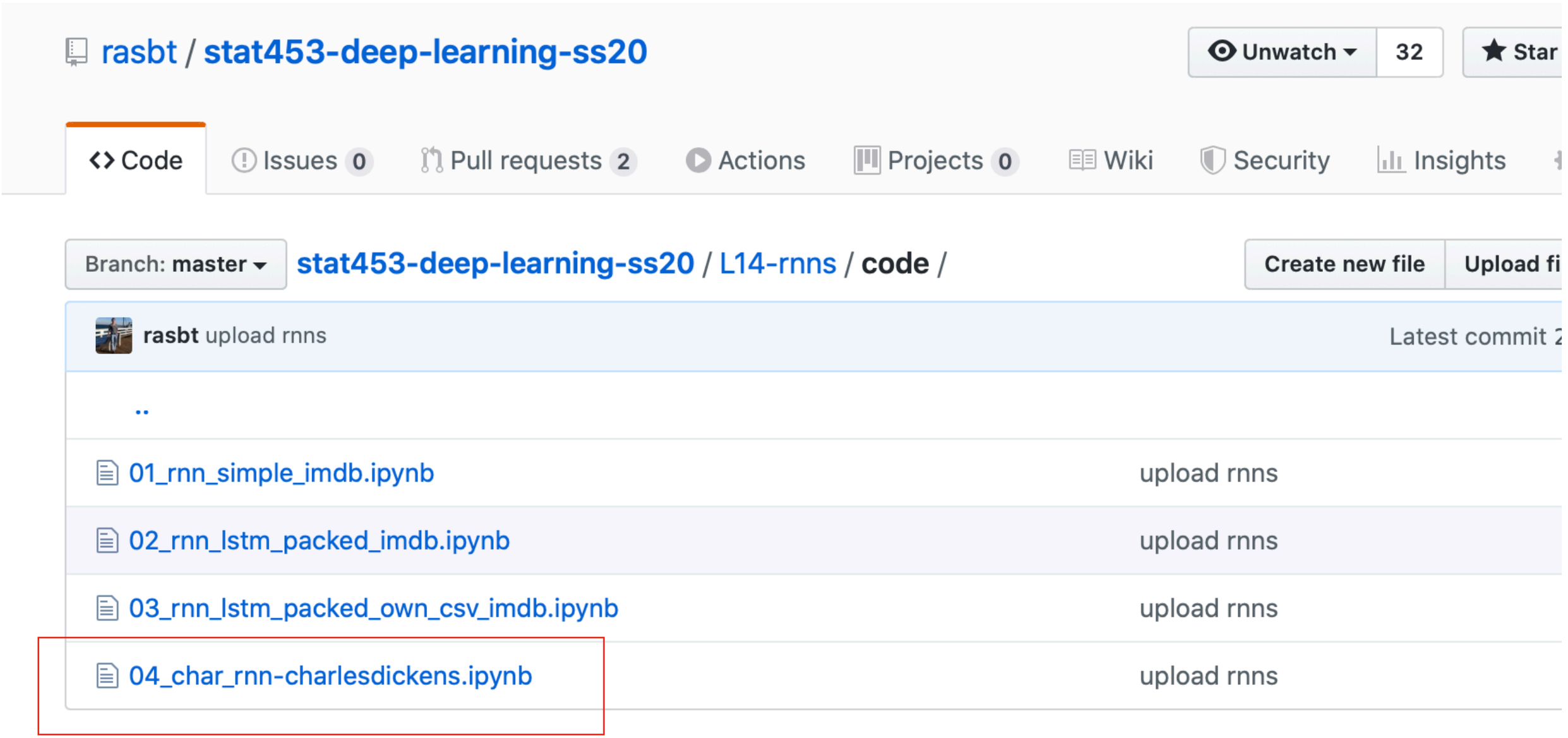
~~many~~-to-many

"one"

"generating new text"

Note that this approach works with both Word- and Character-RNNs

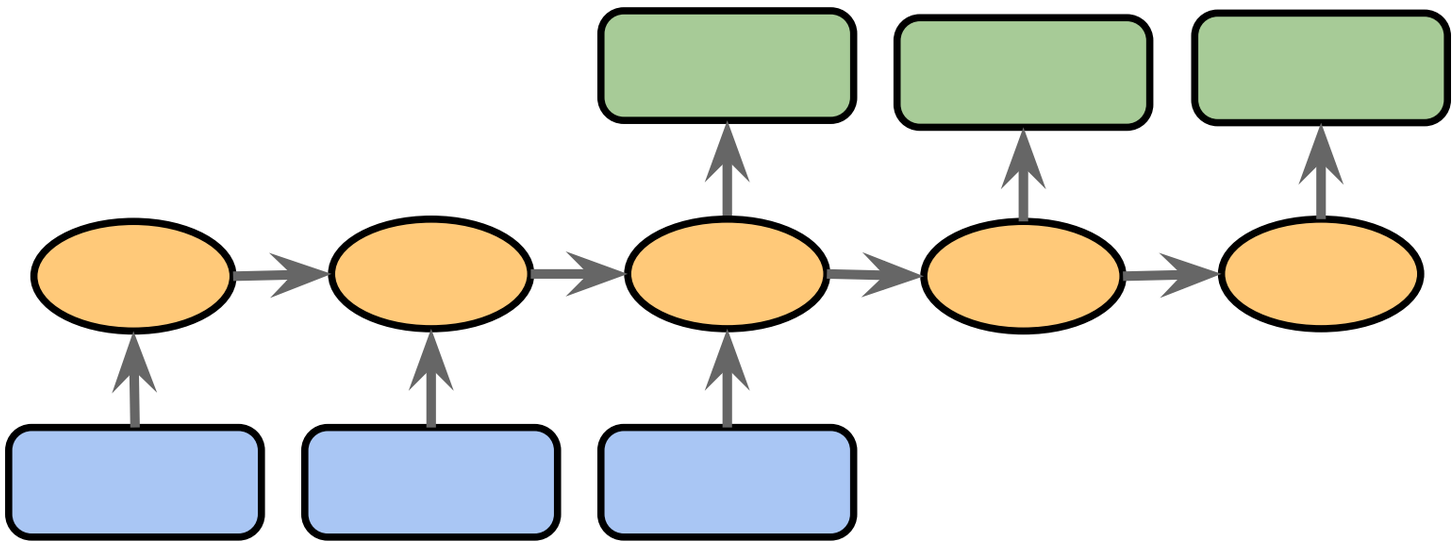
Code Examples



https://github.com/rasbt/stat453-deep-learning-ss20/tree/master/L14-rnns/code

```
Time elapsed: 0.00 min
Iteration 0 | Loss 4.58


Th6#hb:B2iKt0v>E$siC?Q./gyt'y9 o- V ?YA!@hpO%#nsqn&rq1* D!u-*opw;bCc,ut-DAWh)cK&~+}10LOVG/GIpm64-)
(m?,d~3kn?,9Hj8 RmC?
It er%qU?0) bE  nh7 A7i$K\UIc;#RJvcY+_i=r9KS>   v|jf<#y2lz`CCWbDW6W_#_>FzE,{Jbje@ebq

==================================================
Time elapsed: 3.58 min
Iteration 1000 | Loss 1.84


Th
to old. He crour and pert were let of highers brive reall facher, his from ageally bead fore he kn
ewure of or the brown as the shich from of the
go her ty his fross treidef that in at projed to that
==================================================
```

```
Time elapsed: 0.00 min
Iteration 0 | Loss 4.58


Th6#hb:B2iKt0v>E$siC?Q./gyt'y9 o- V ?YA!@hpO%#nsqn&rq1* D!u-*opw;bCc,ut-DAWh)cK&~+}10LOVG/GIpm64-)
(m?,d~3kn?,9Hj8 RmC?
It er%qU?0) bE  nh7 A7i$K\UIc;#RJvcY+_i=r9KS>   v|jf<#y2lz`CCWbDW6W_#_>FzE,{Jbje@ebq

==================================================
Time elapsed: 3.58 min
Iteration 1000 | Loss 1.84


Th
to old. He crour and pert were let of highers brive reall facher, his from ageally bead fore he kn
ewure of or the brown as the shich from of the
go her ty his fross treidef that in at projed to that

==================================================


==================================================
Time elapsed: 64.67 min
Iteration 18000 | Loss 1.56


The say close,
and the danger folded there down from his a the danking at her..

"It have have as atton to being and happle and forfartuen way, them gonged
the baker's shore they good and Brongth with a

==================================================
Time elapsed: 68.27 min
Iteration 19000 | Loss 1.77
```



Translation with a Sequence to Sequence Network and Attention (English to French)

https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Figure based on:

The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy (http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

Lecture Overview

RNNs and Sequence Modeling Tasks

Backpropagation Through Time

Long-short term memory (LSTM)

Many-to-one Word RNNs

Generating Text with Character RNNs

Attention Mechanisms and Transformers

Attention Mechanism

- originally developed for language translation:
  Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

  "... allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word ..."

"traditional" encoder+decoder RNN

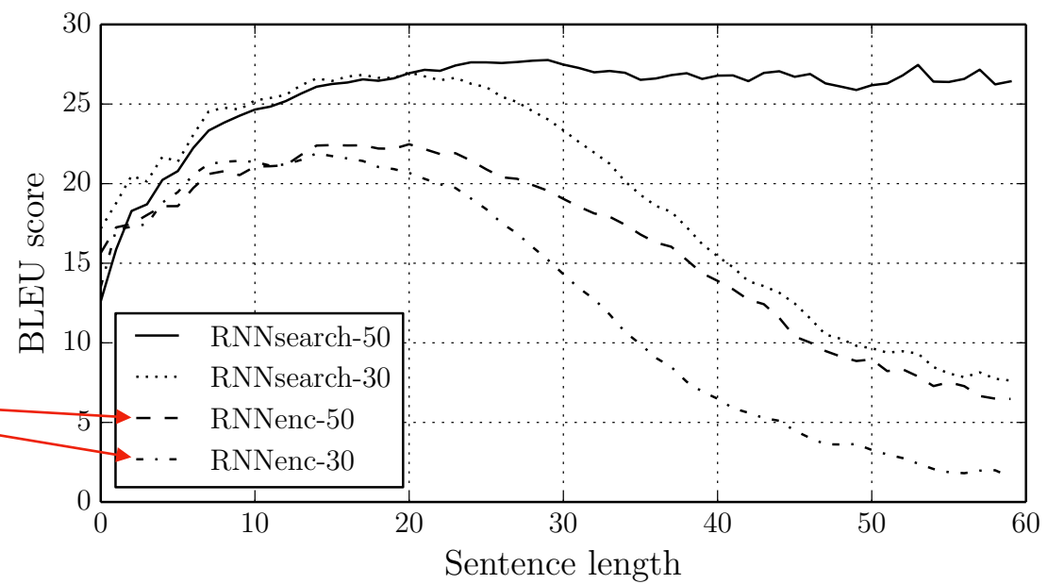


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Attention Mechanism

Assign attention weight to each word to know how much "attention" the model should pay to each word (i.e., for each word, the network learns a "context")

Attention Mechanism

- originally developed for language translation:
  Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Assign attention weight to each word to know how much "attention" the model should pay to each word (i.e., for each word, the network learns a "context")

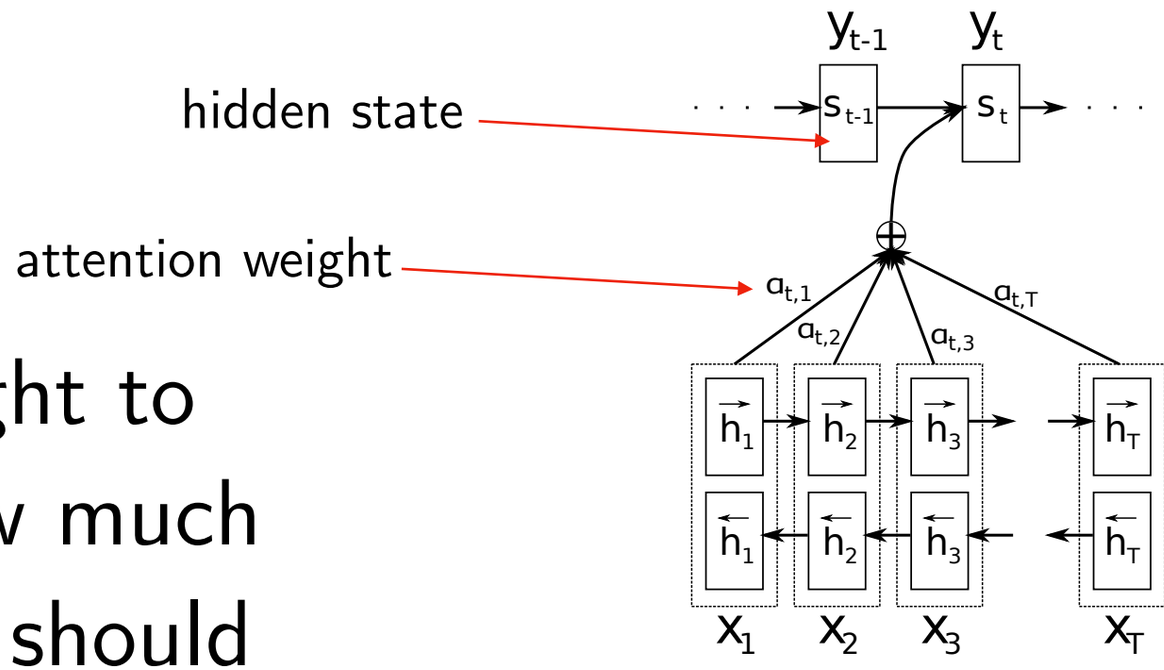
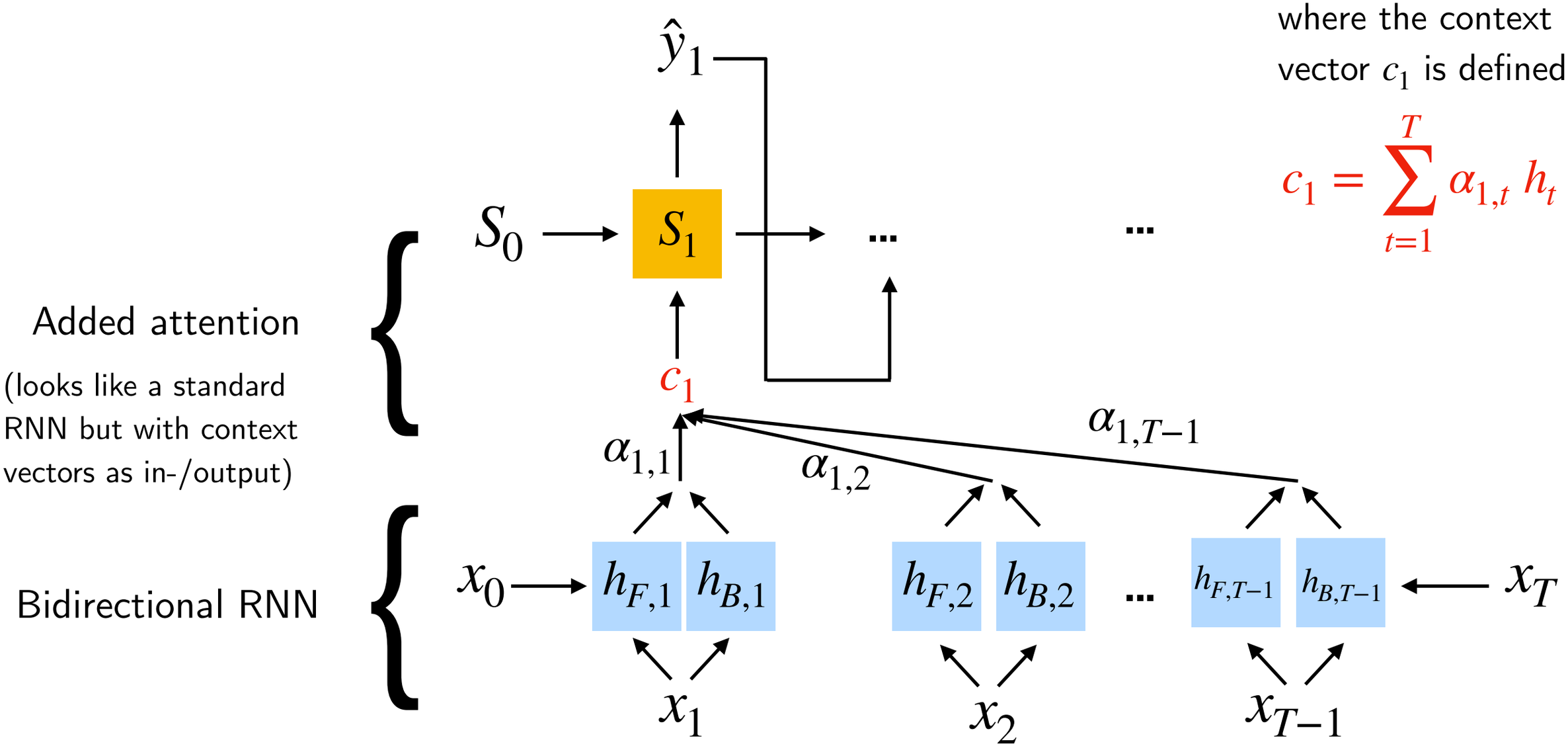


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

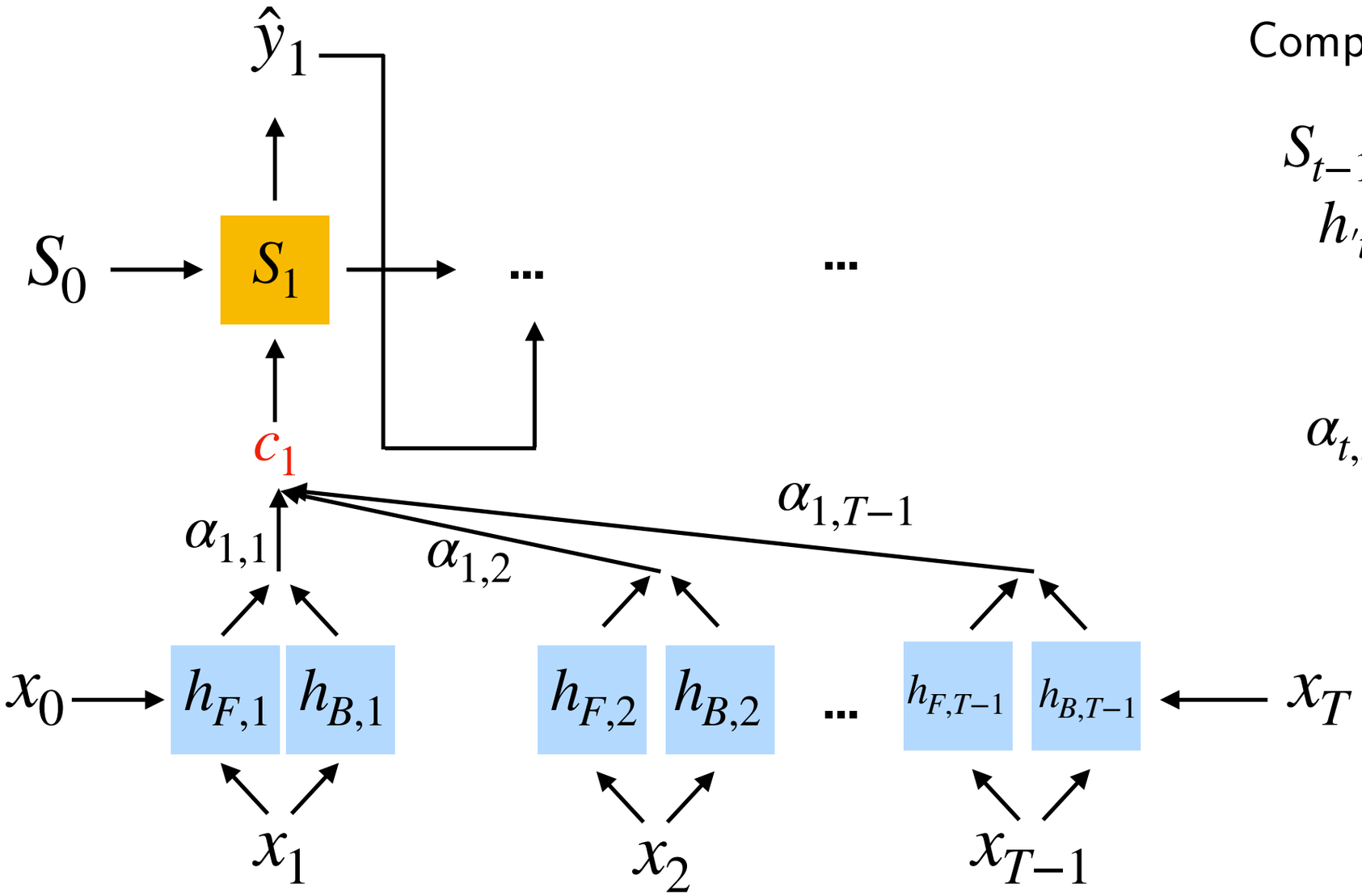
RNN Attention Mechanism

where the context vector c_1 is defined as

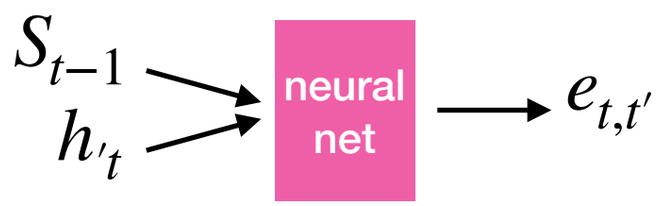
$$c_1 = \sum_{t=1}^{T} \alpha_{1,t} \, h_t$$



Attention Mechanism

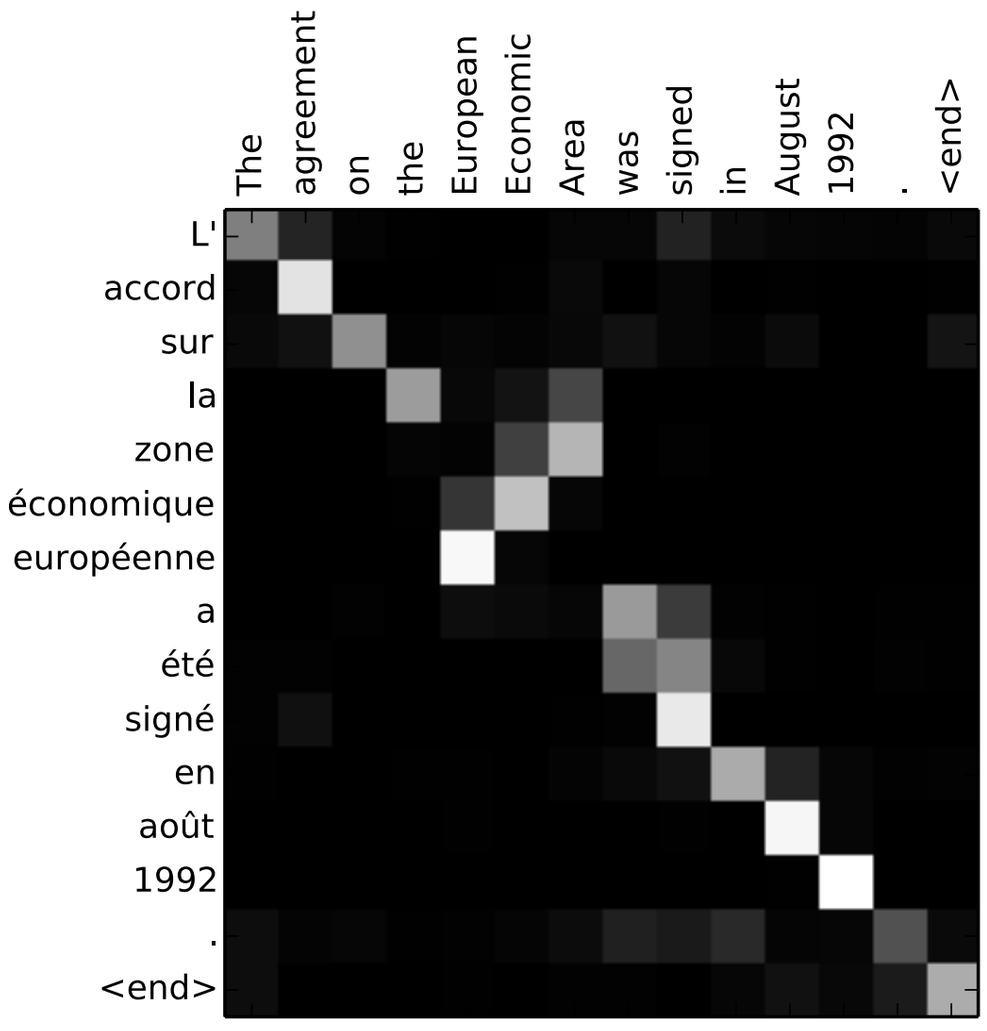


Computing attention weights

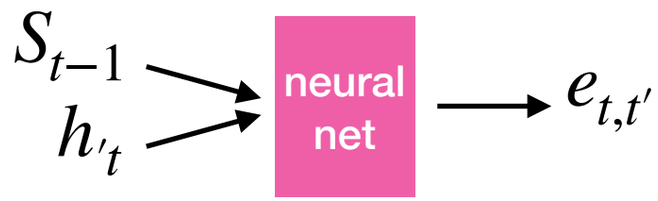


$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

Attention Mechanism



Computing attention weights



$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Self-Attention Mechanism

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

https://arxiv.org/abs/1706.03762

A Basic Version of Self-Attention

Step 1: Compute dot products

Step 2: Normalize using softmax

$$W_{ij} = softmax\left(\left[\boldsymbol{x}^{(i)\top}\boldsymbol{x}^{(j)}\right]_{j\in[0,T]}\right)$$

Step 3: Compute output

$$\boldsymbol{o}^{(i)} = \sum_{j=0}^{T} W_{ij}\,\boldsymbol{x}^{(j)}$$

Input sequence:

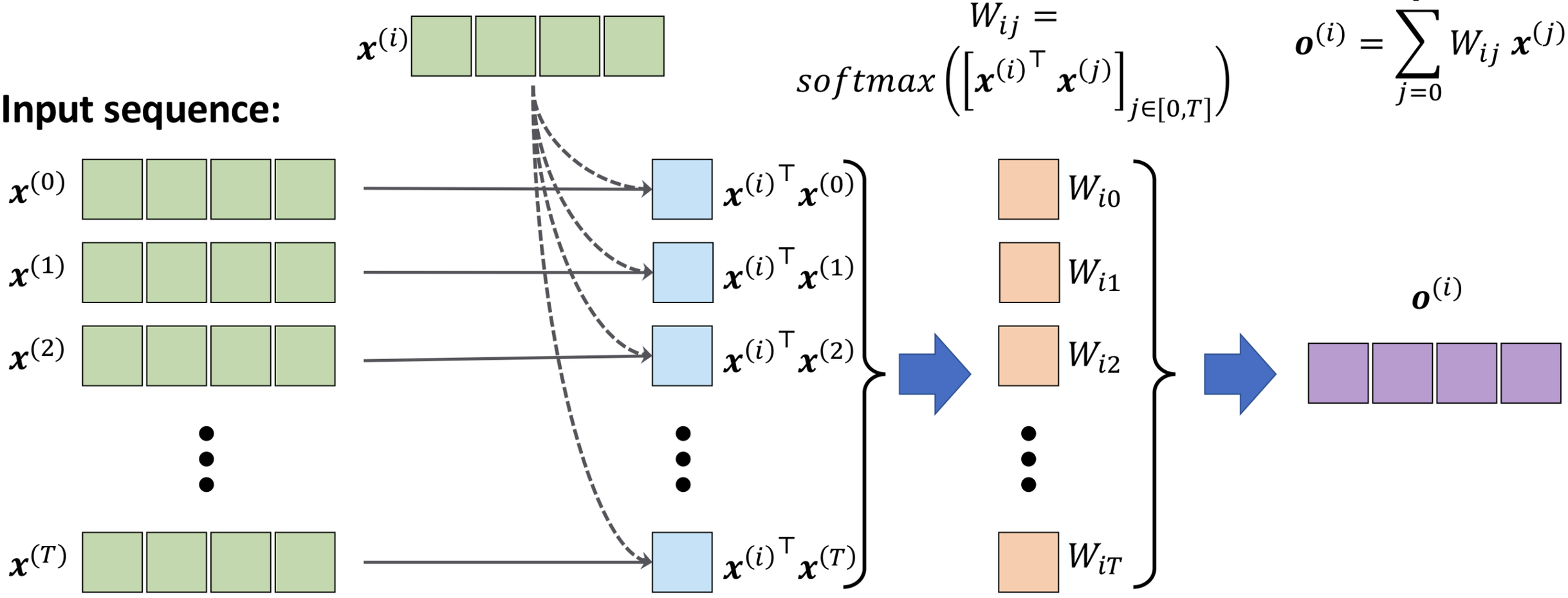


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Multi-head Attention and Transformer Blocks

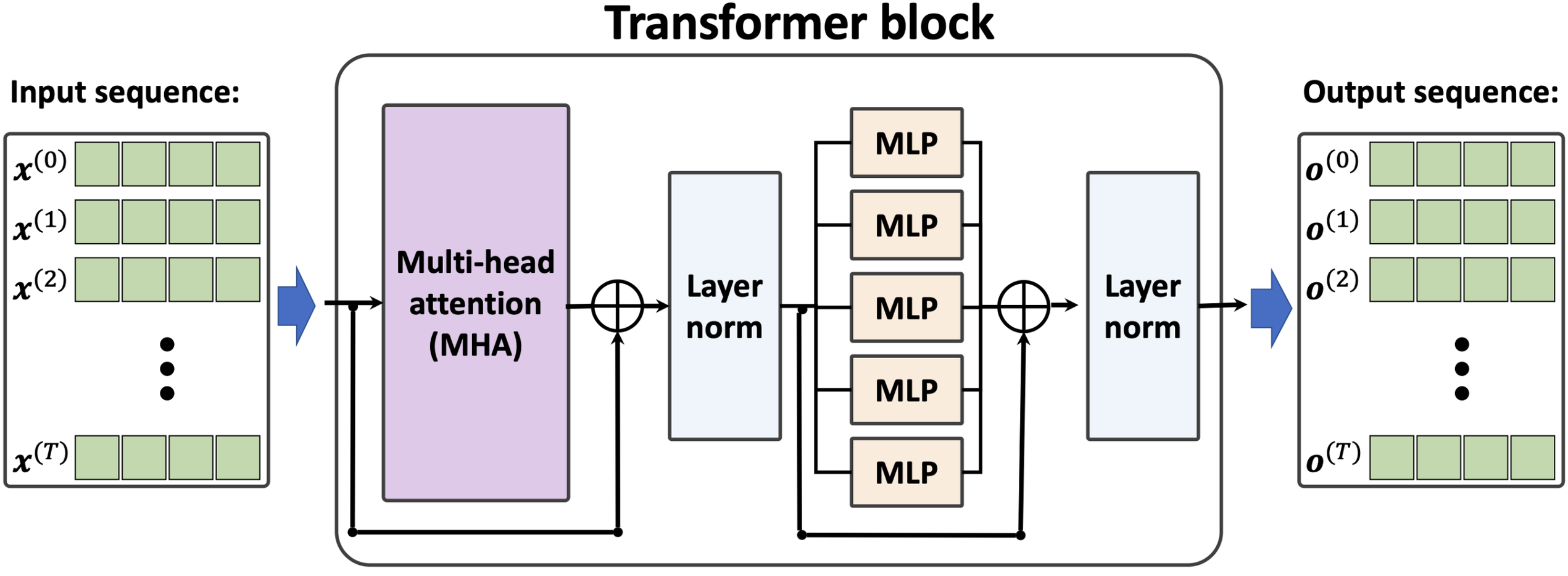


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Implementations:

https://pytorch.org/hub/huggingface_pytorch-transformers/

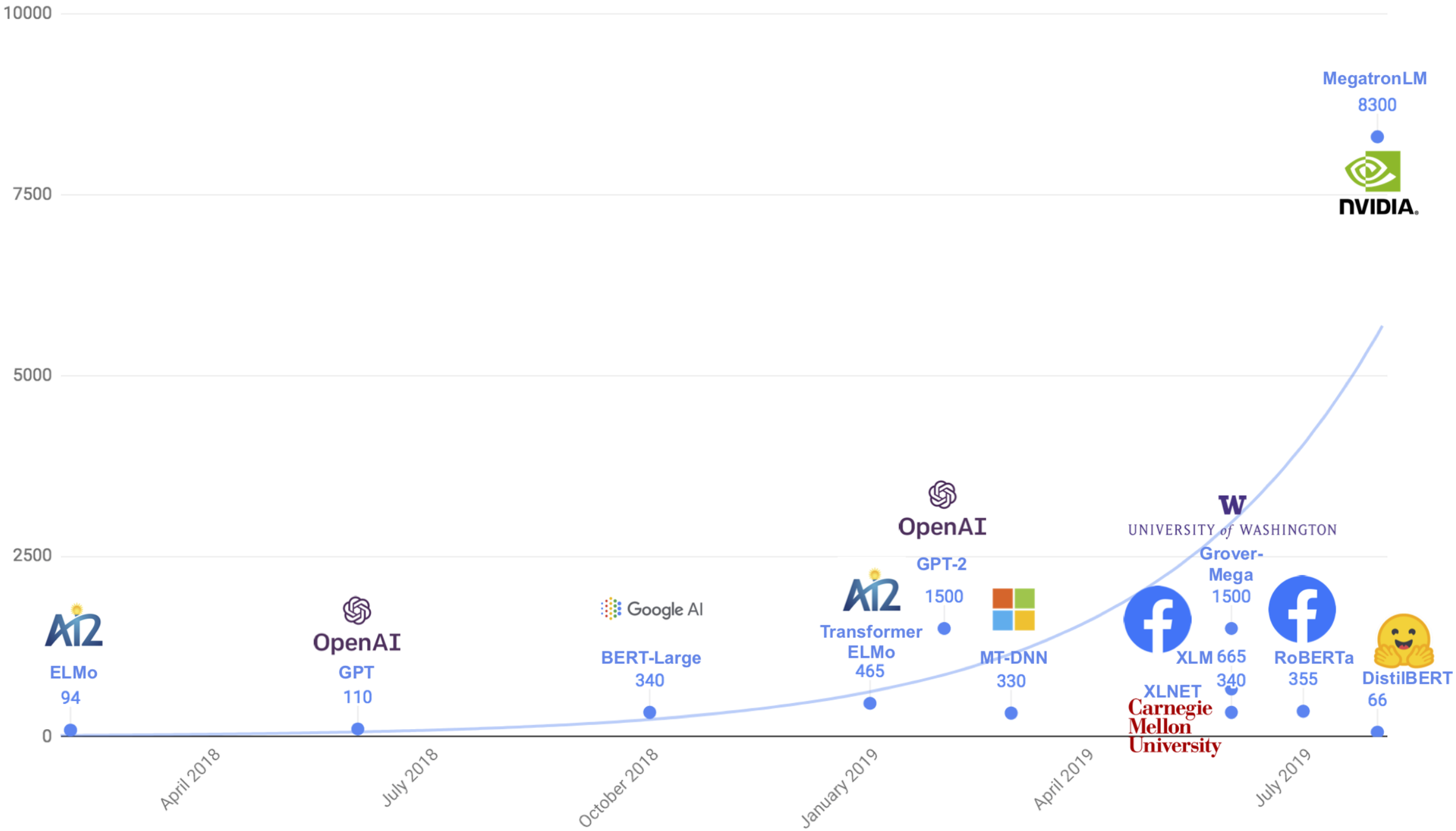


Image Source: https://medium.com/huggingface/distilbert-8cf3380435b5

Optional Reading Material

Chapter 10, Sequence Modeling: Recurrent and Recursive Nets:

https://www.deeplearningbook.org/contents/rnn.html