

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

```
x[1] = "The sun is shining"  
x[2] = "The weather is sweet"  
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

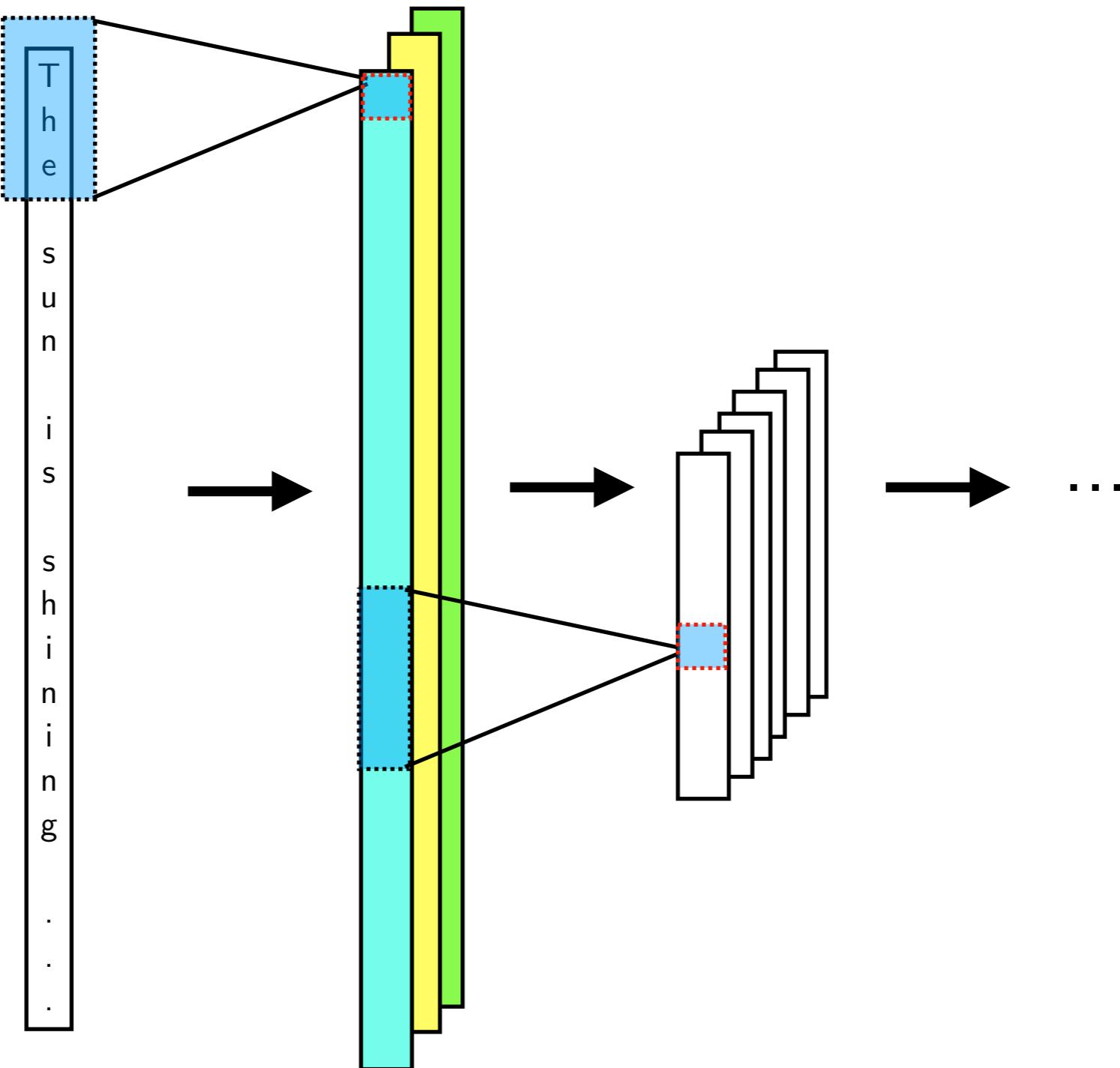
training

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.

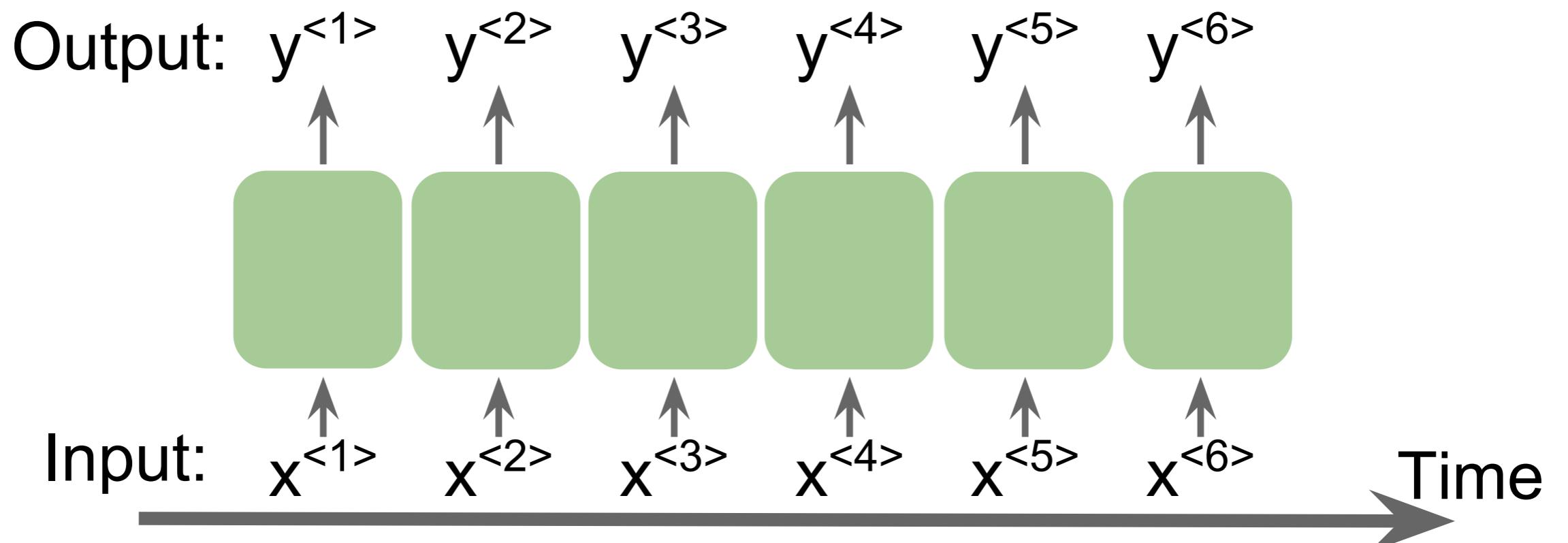


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Applications: Working with Sequential Data

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

Stock market predictions

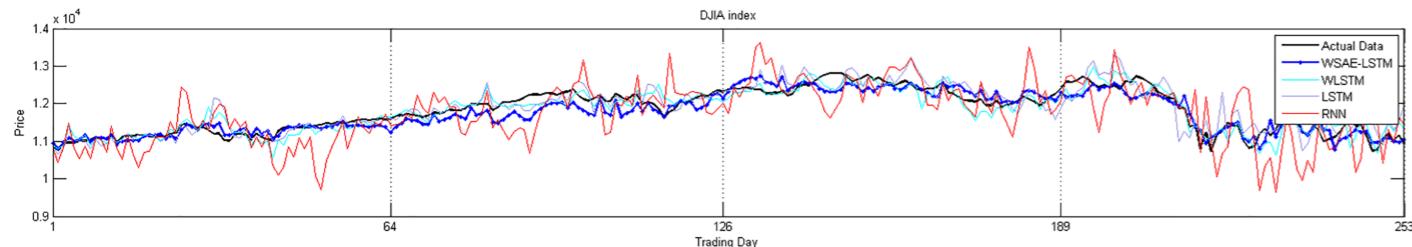
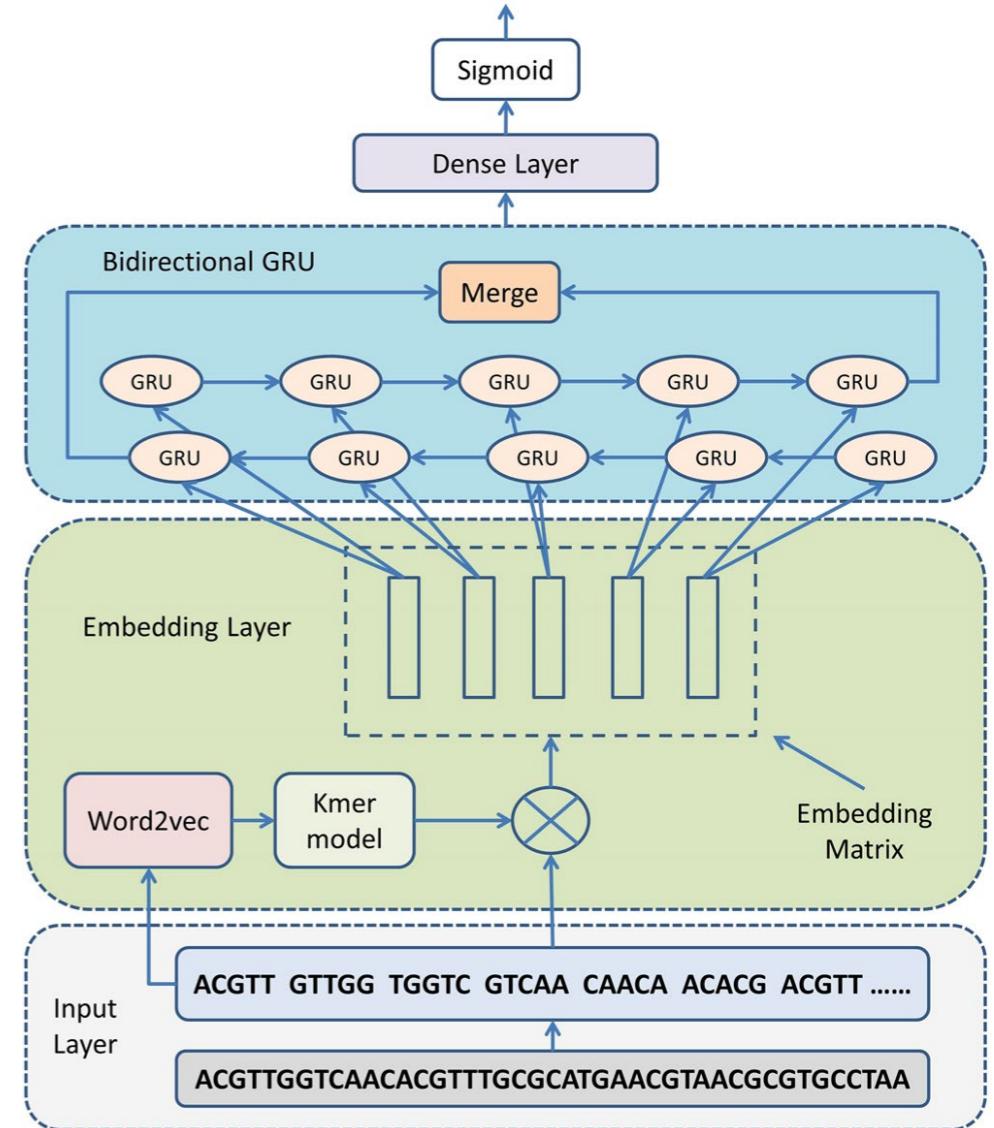


Fig 8. Displays the actual data and the predicted data from the four models for each stock index in Year 1 from 2010.10.01 to 2011.09.30.

<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.



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

Overview

Networks we used previously: also called feedforward neural networks

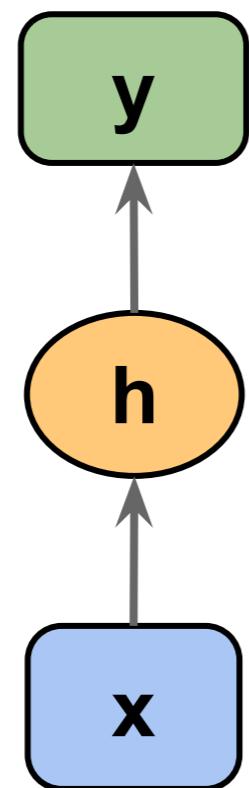
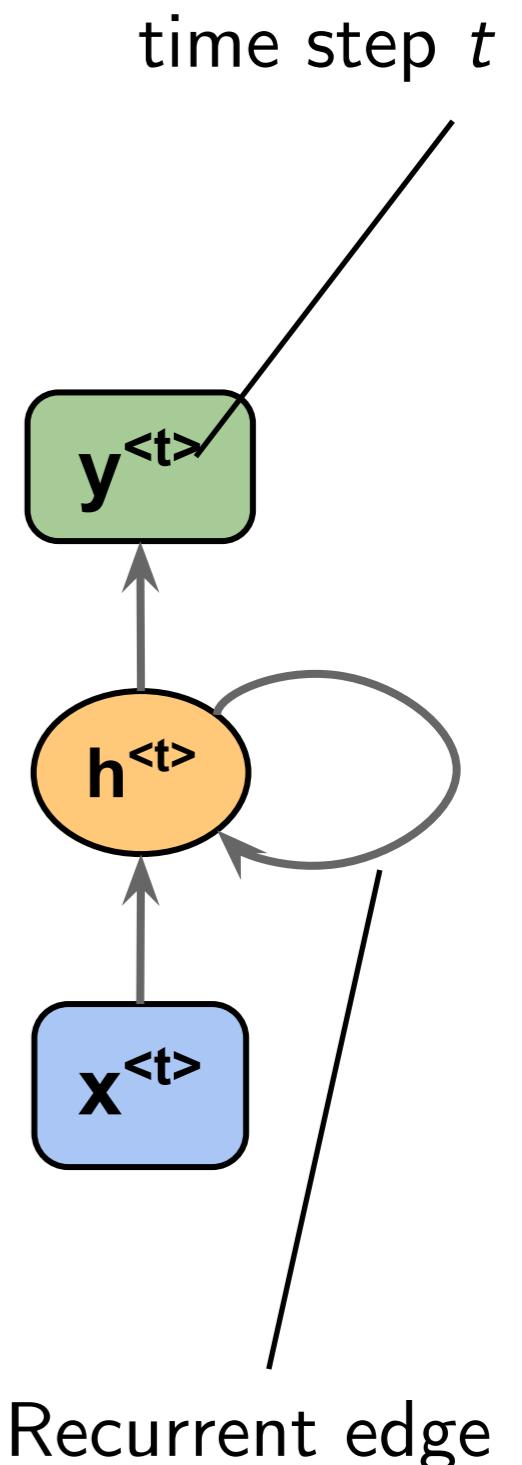


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Recurrent Neural Network (RNN)



Overview

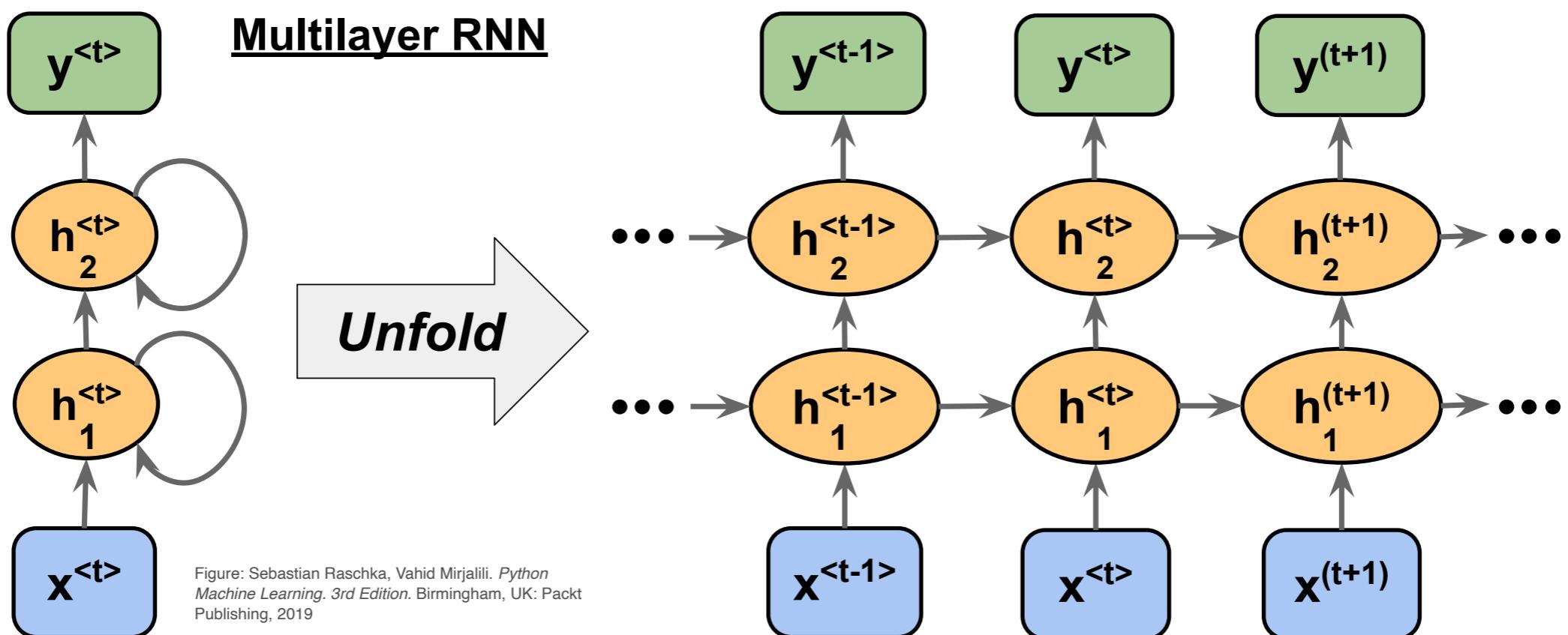
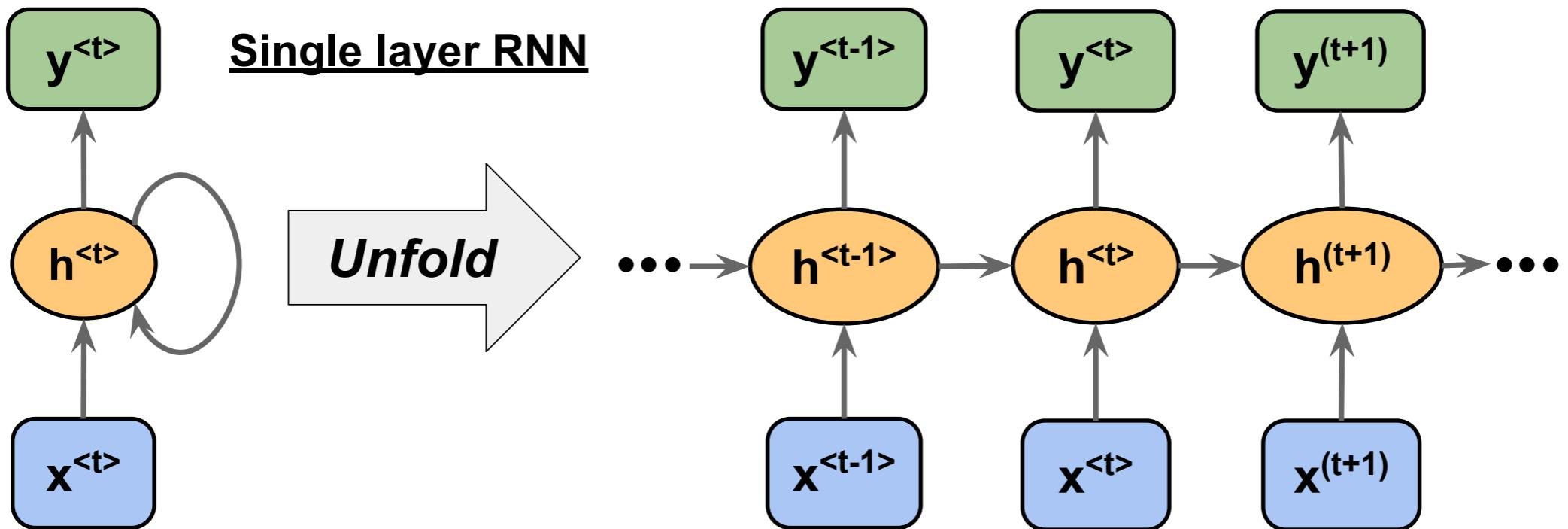
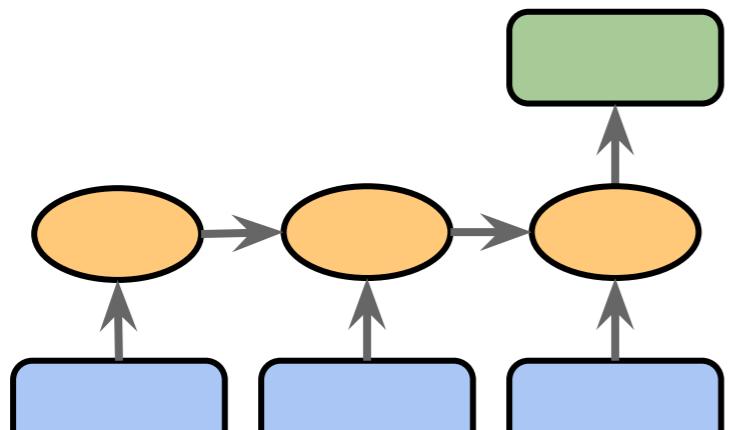
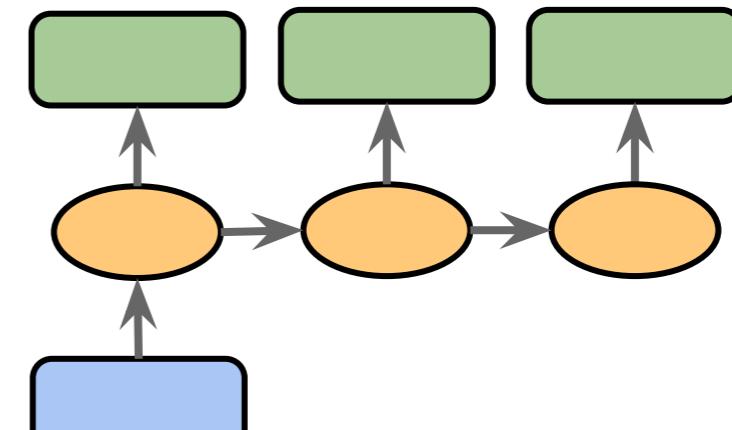


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

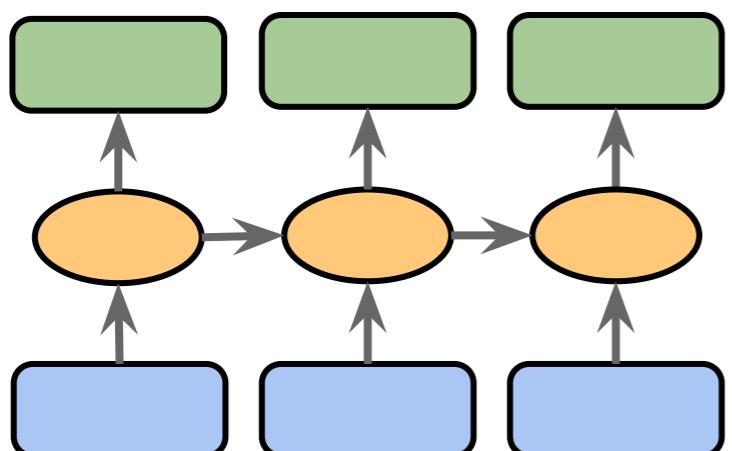
Different Types of Sequence Modeling Tasks



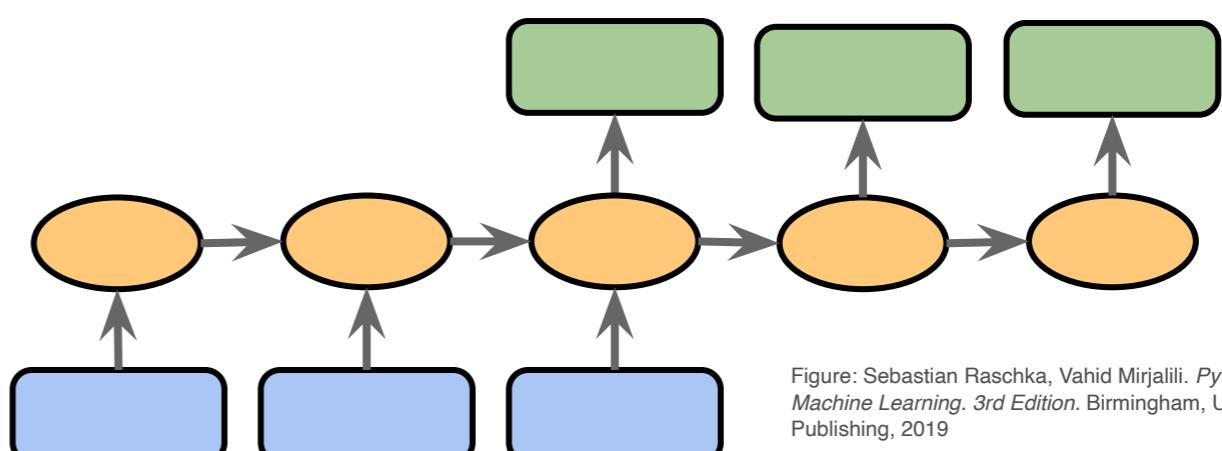
many-to-one



one-to-many



many-to-many



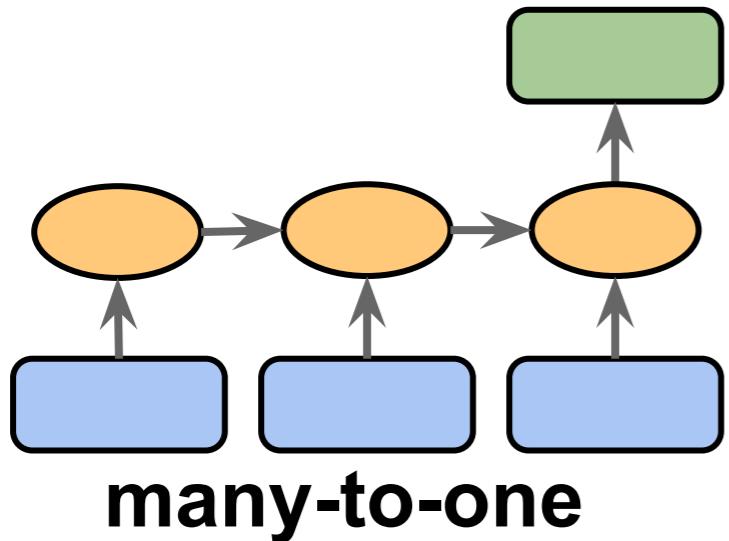
many-to-many

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/>)

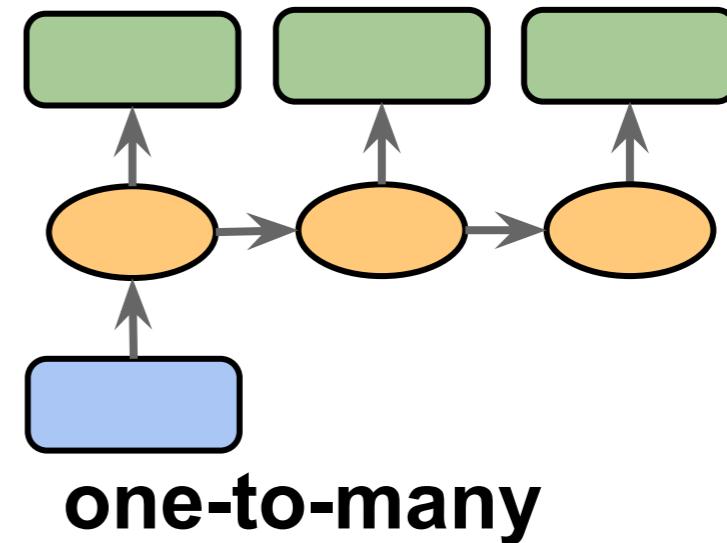
Different Types of Sequence Modeling Tasks



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.

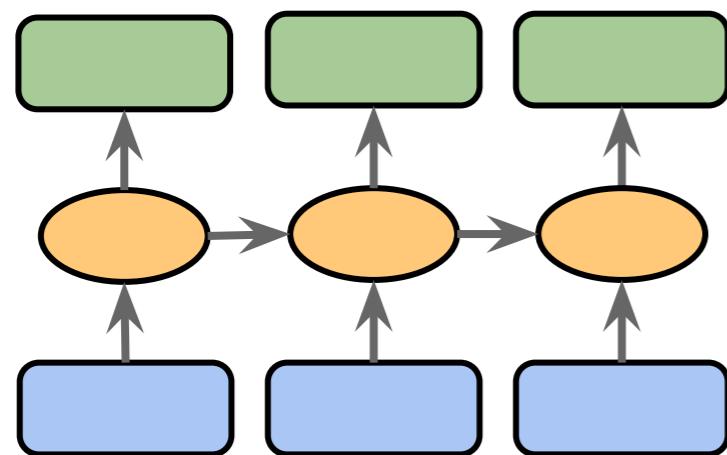
Ex.: Image captioning, where the input is an image, the output is a text description of that image

Different Types of Sequence Modeling Tasks

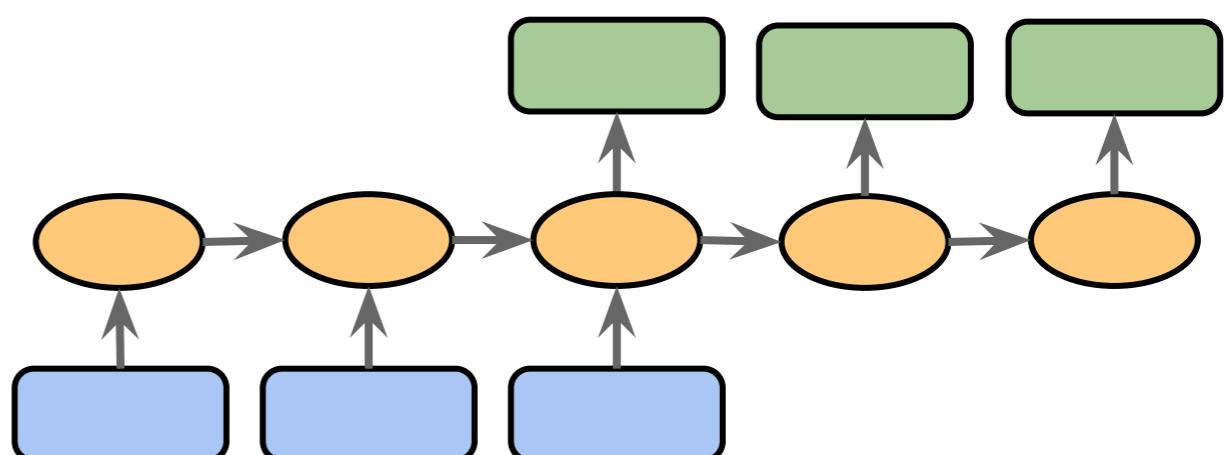
Many-to-many: Both inputs and outputs are sequences. Can be direct or delayed.

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

Translating one language into another (delayed)



many-to-many



many-to-many

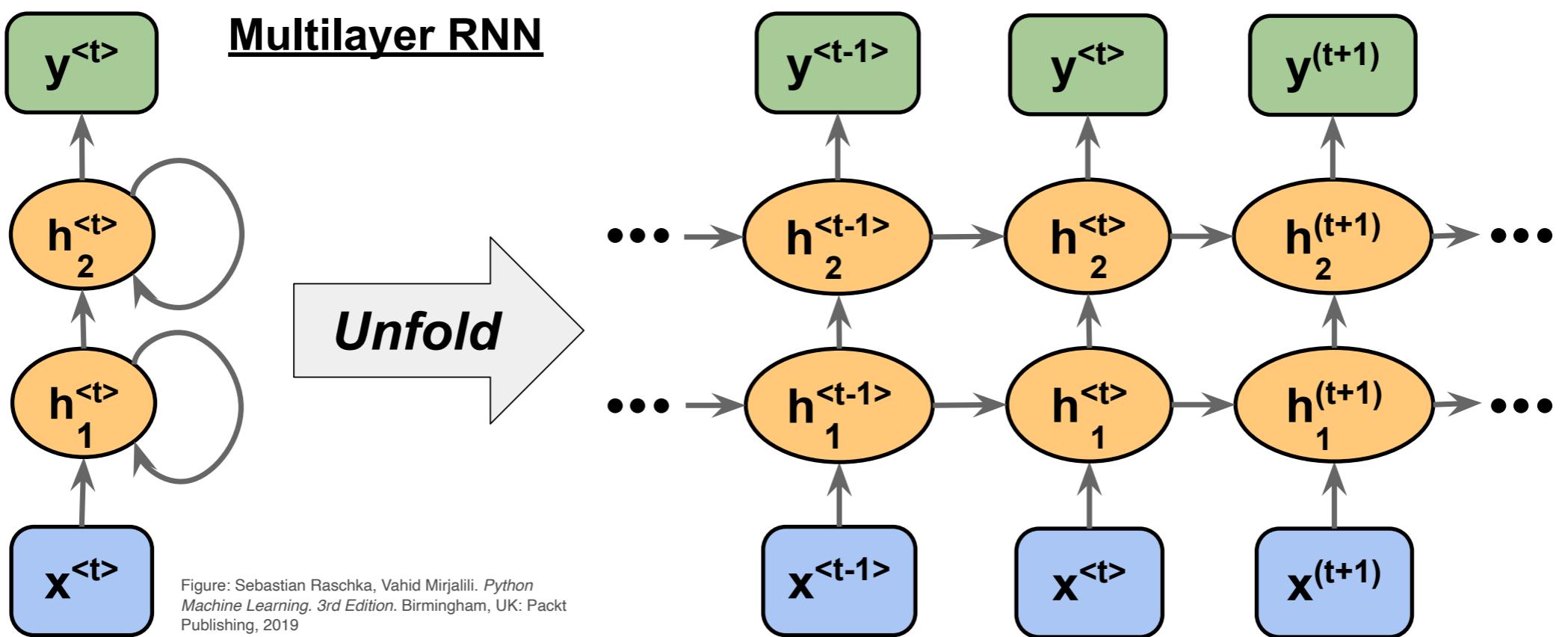
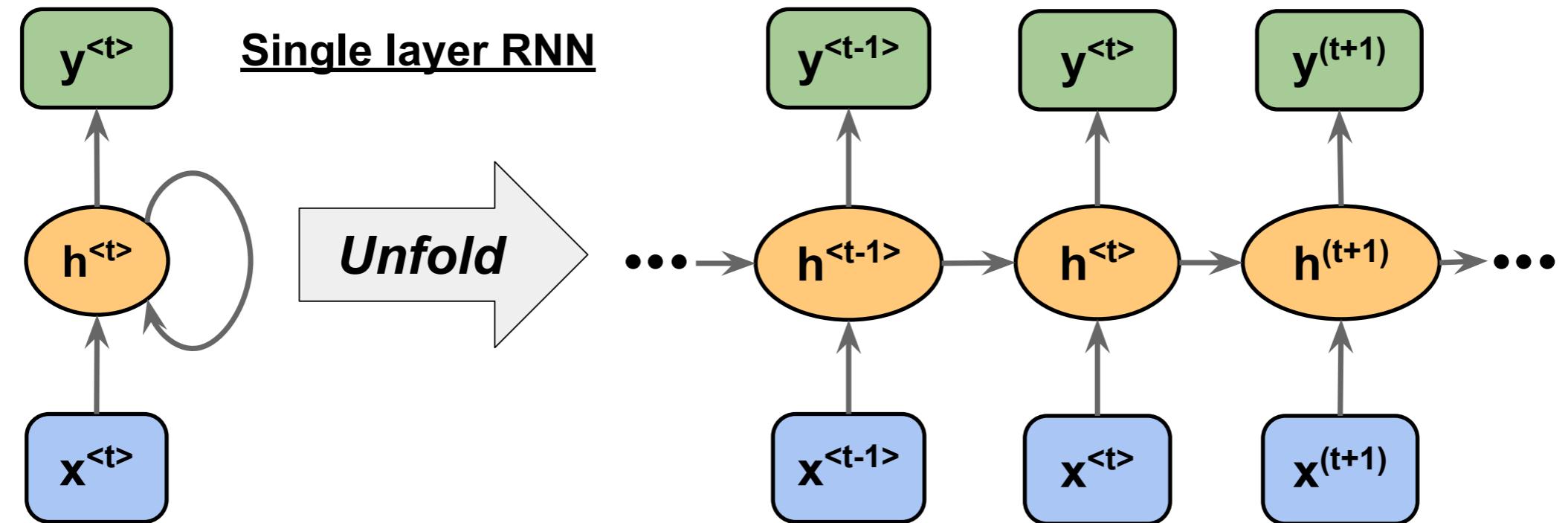


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

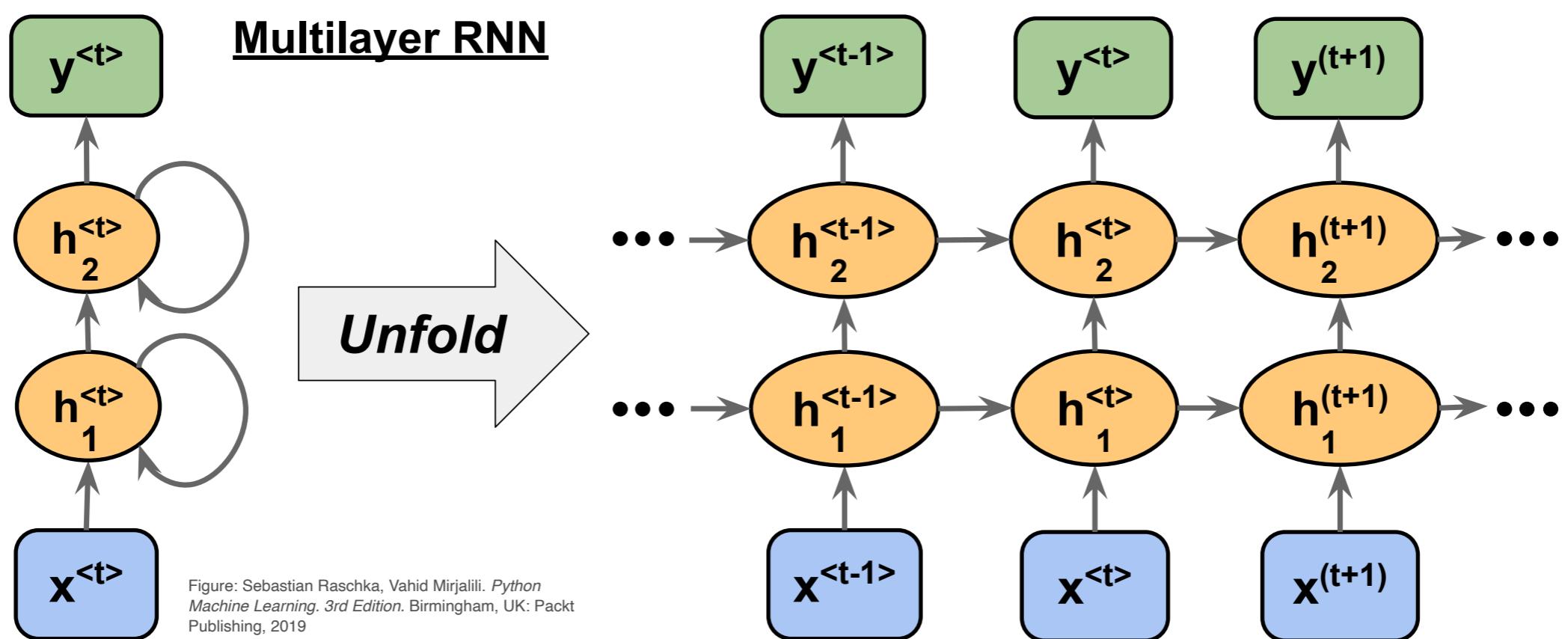
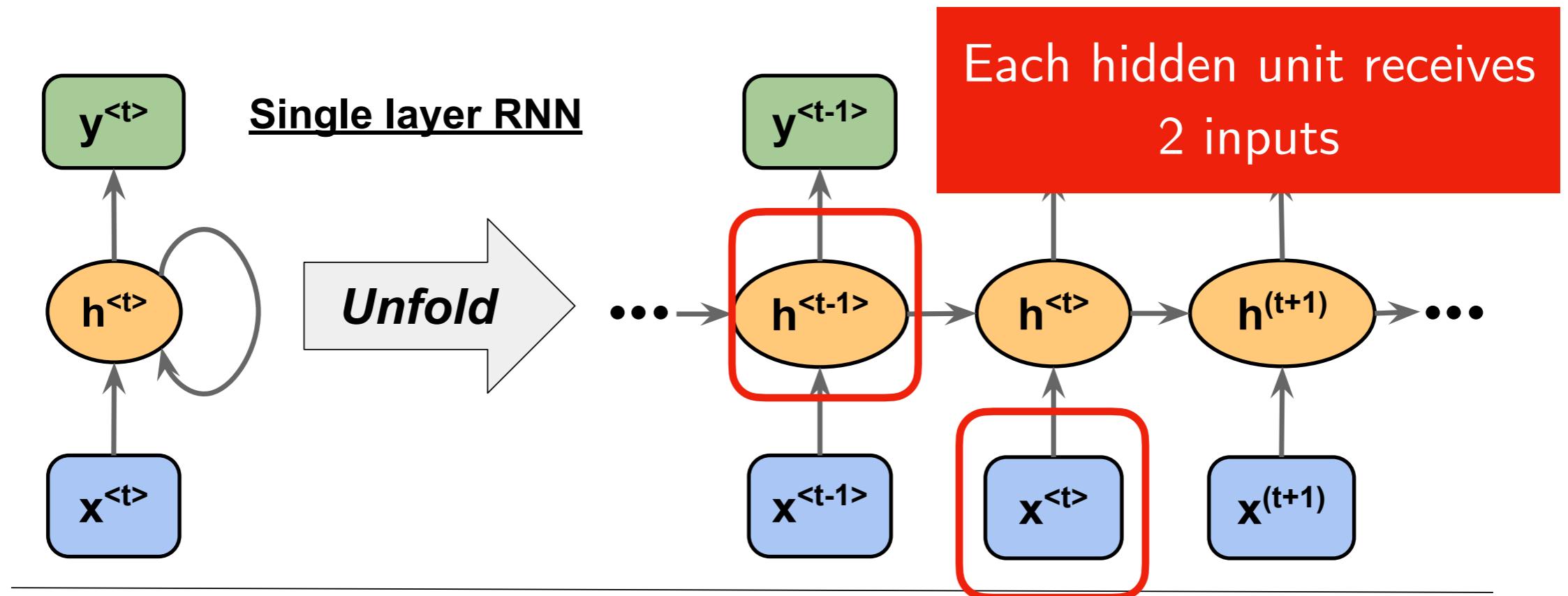


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

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

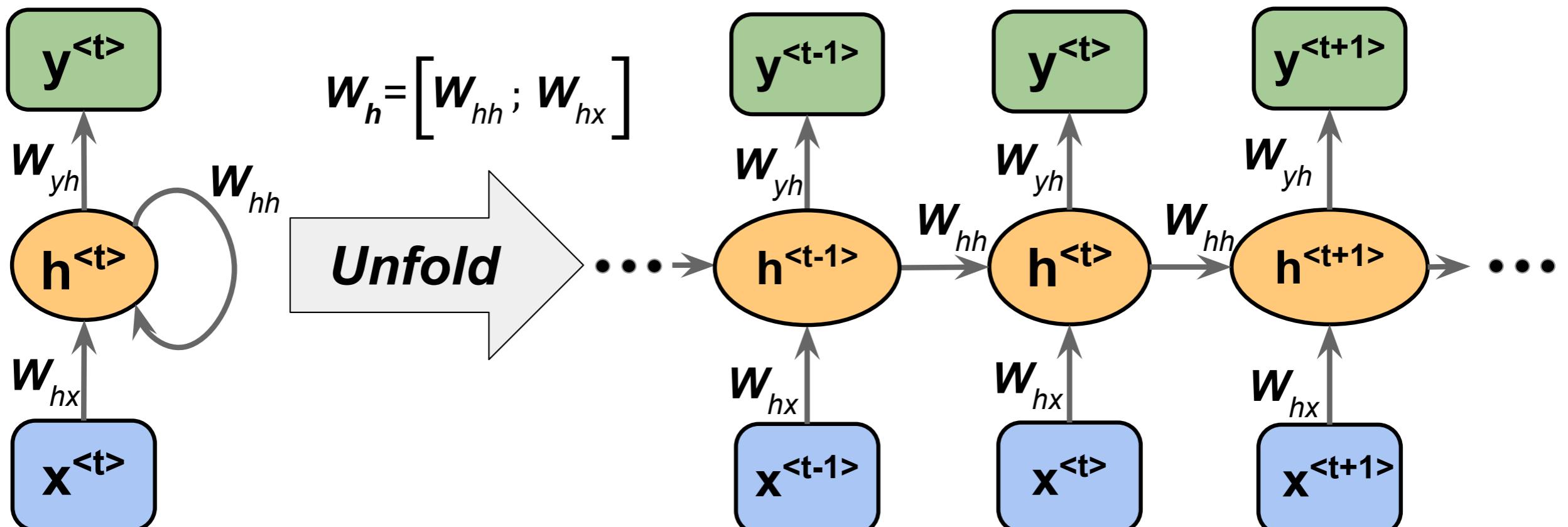


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Weight matrices in a single-hidden layer RNN

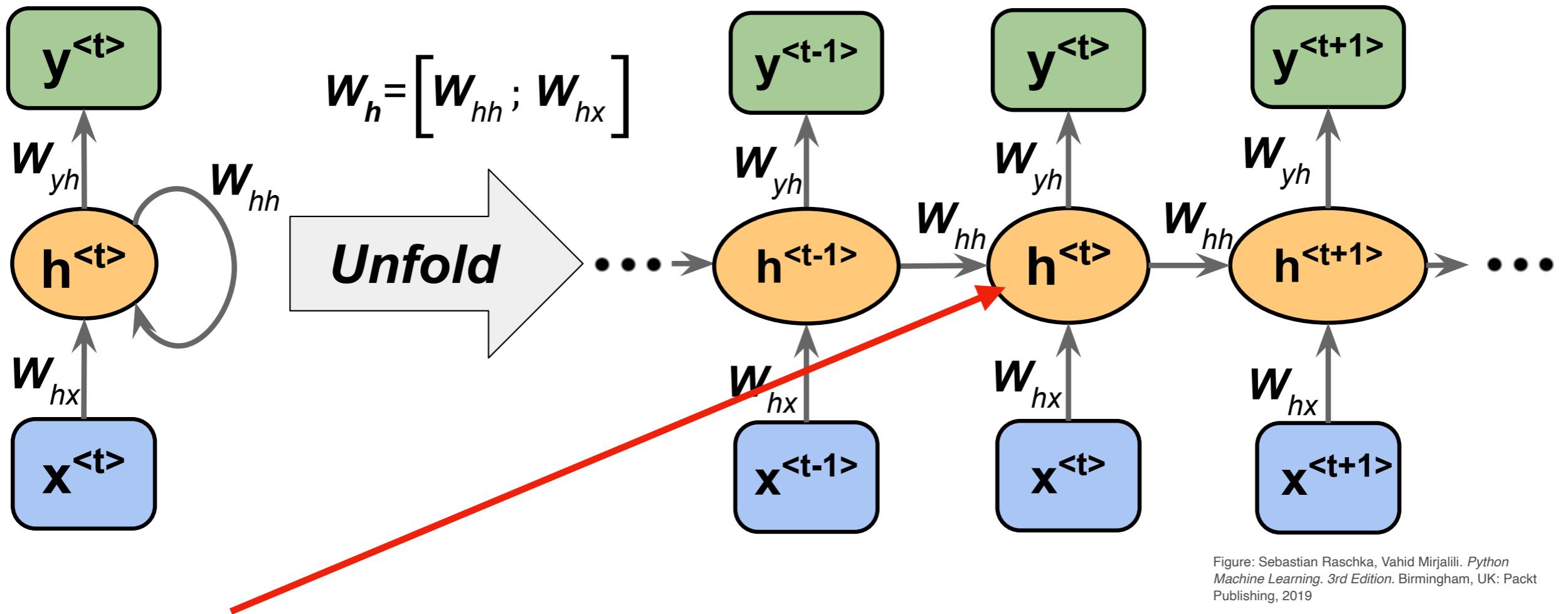


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Net input:

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

Activation:

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{z}_h^{(t)})$$

Weight matrices in a single-hidden layer RNN

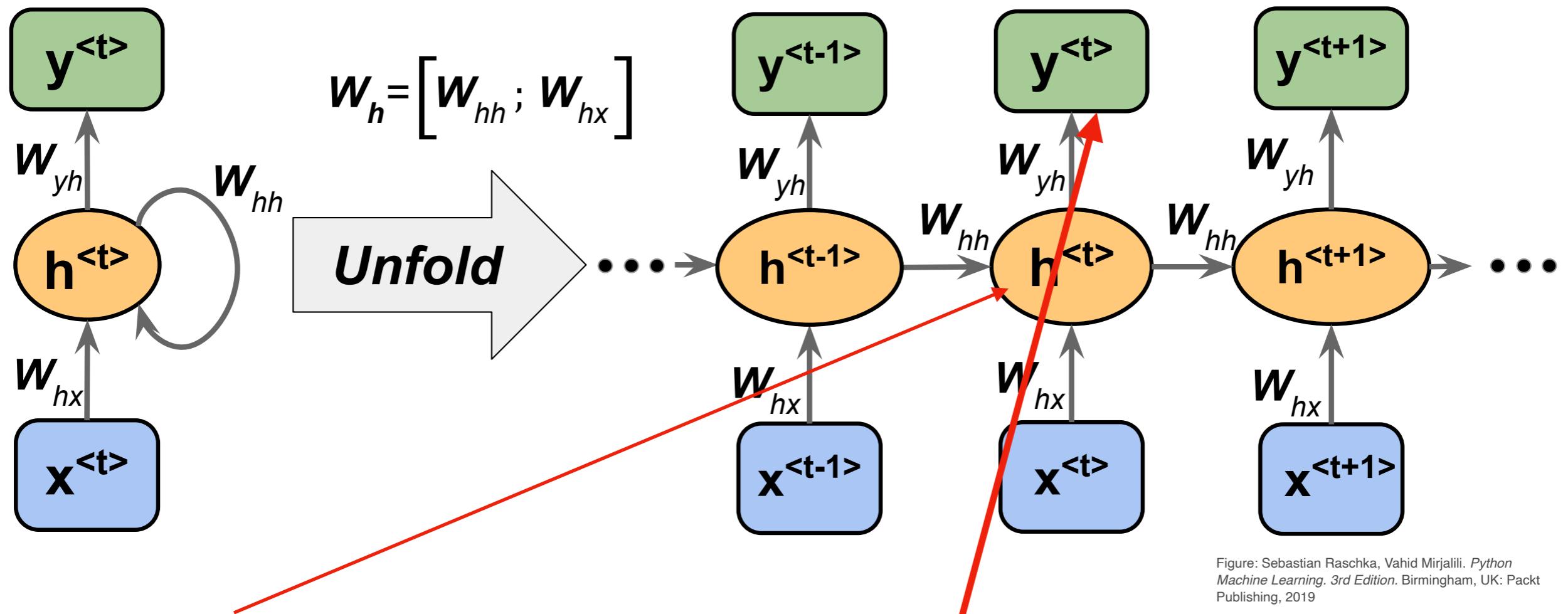


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Net input:

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

Activation:

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{z}_h^{(t)})$$

Net input:

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

Output:

$$\mathbf{y}^{(t)} = \sigma_y(\mathbf{z}_y^{(t)})$$

Backpropagation through time

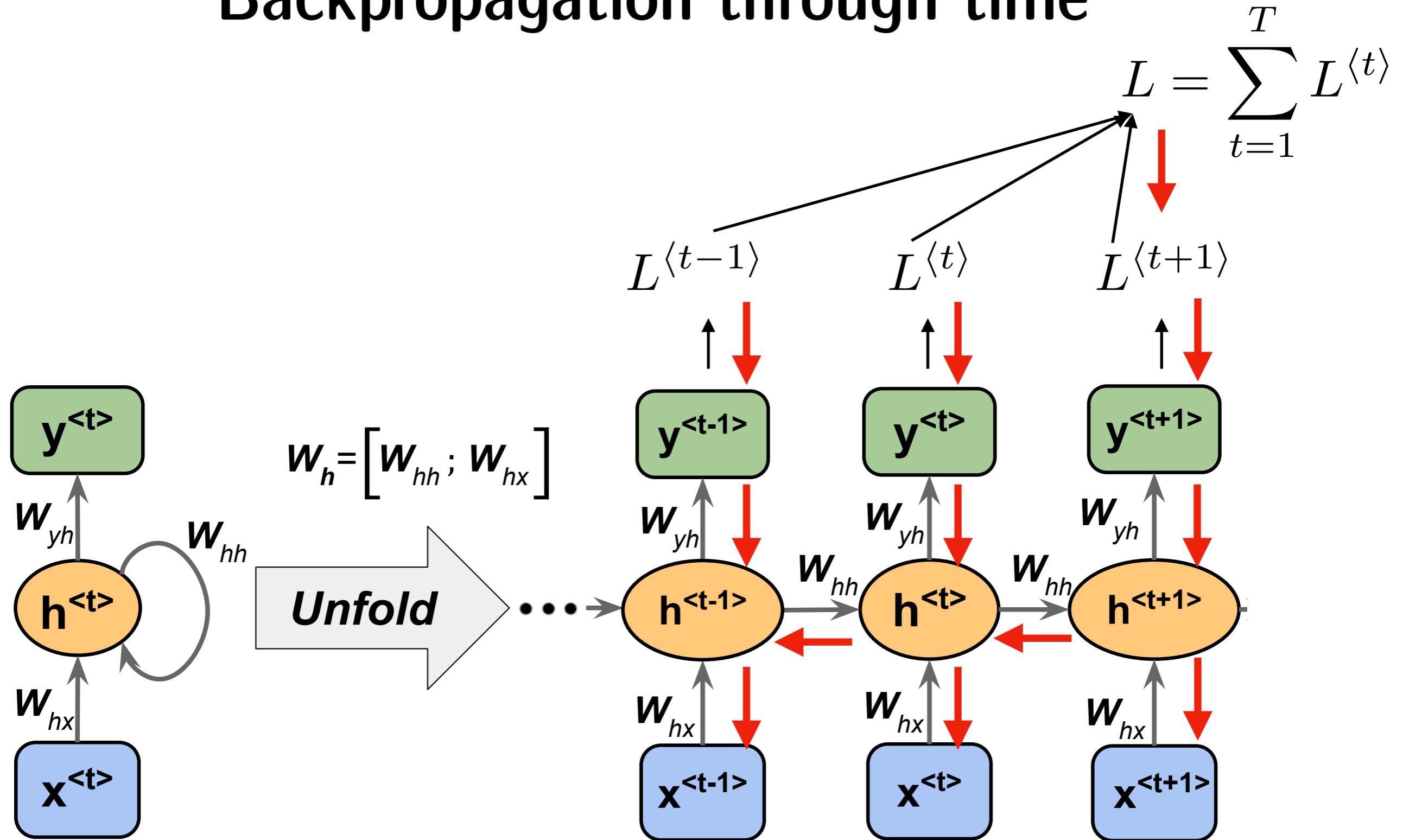
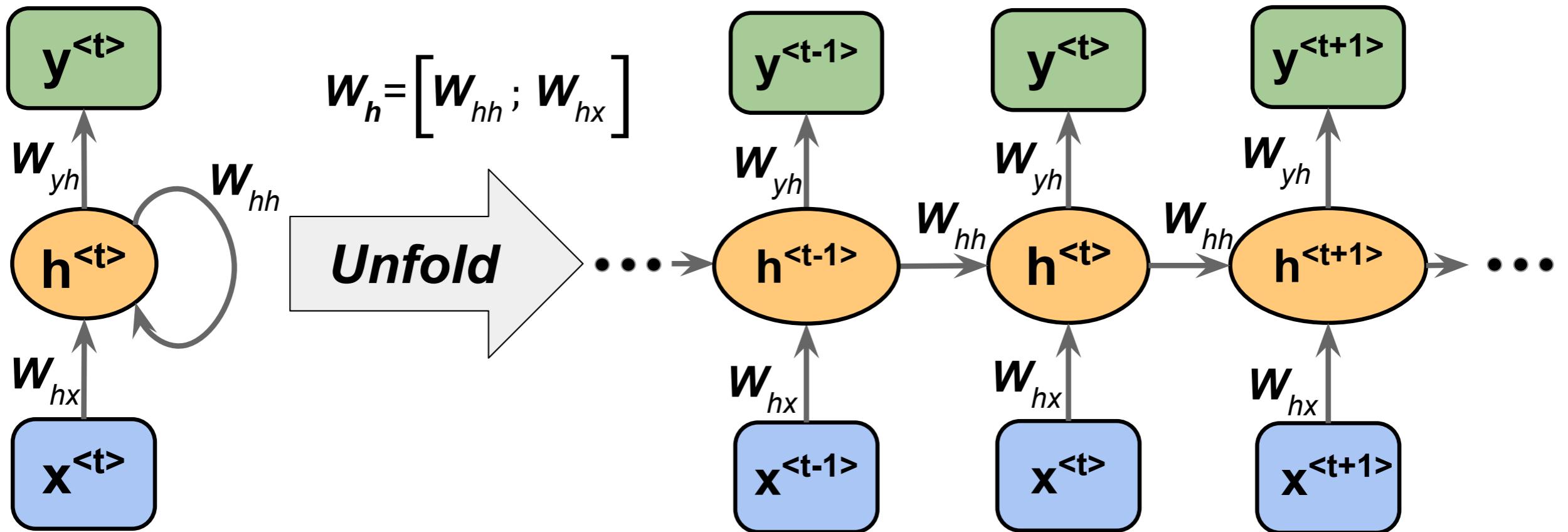


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Backpropagation through time

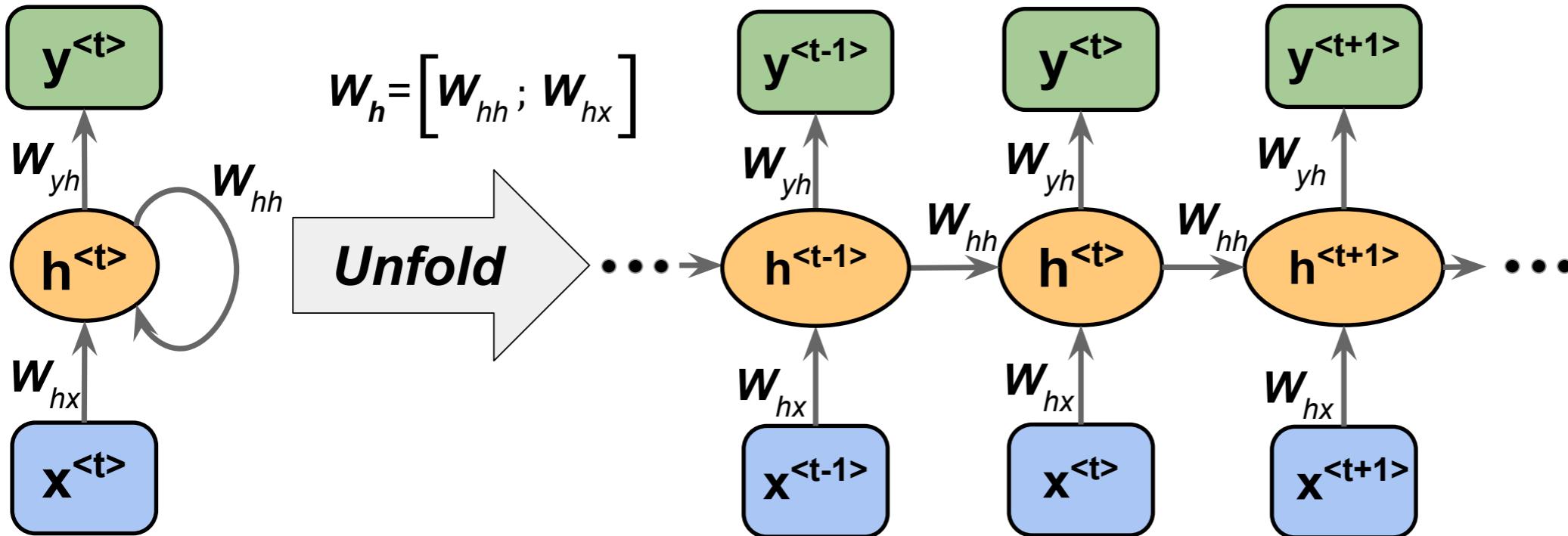


Werbos, Paul J. "[Backpropagation through time: what it does and how to do it.](#)"
Proceedings of the IEEE 78, no. 10 (1990): 1550-1560.

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

$$L = \sum_{t=1}^T L^{<t>}$$

Backpropagation through time



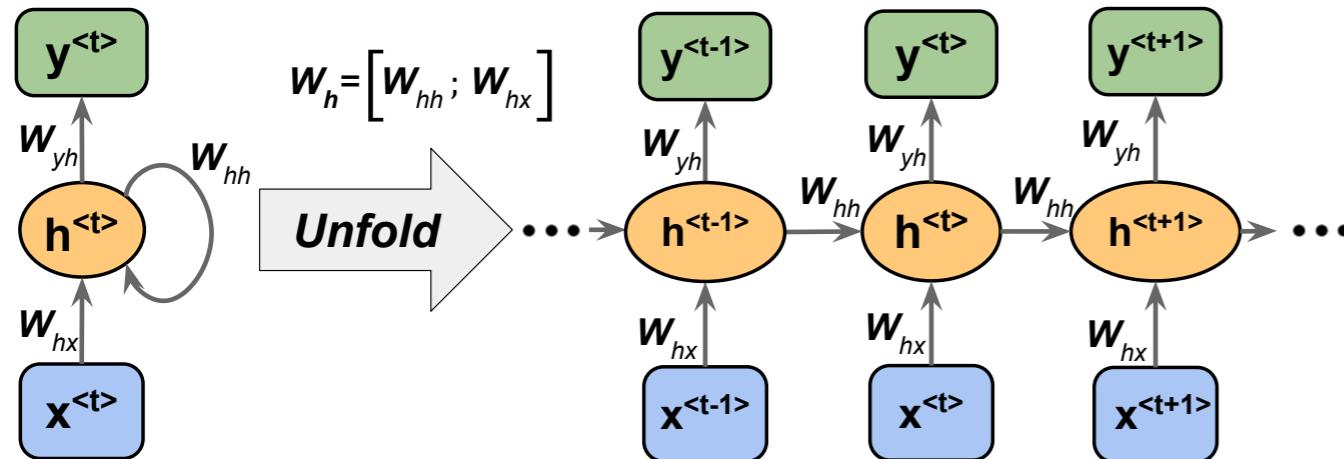
Werbos, Paul J. "[Backpropagation through time: what it does and how to do it.](#)"

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



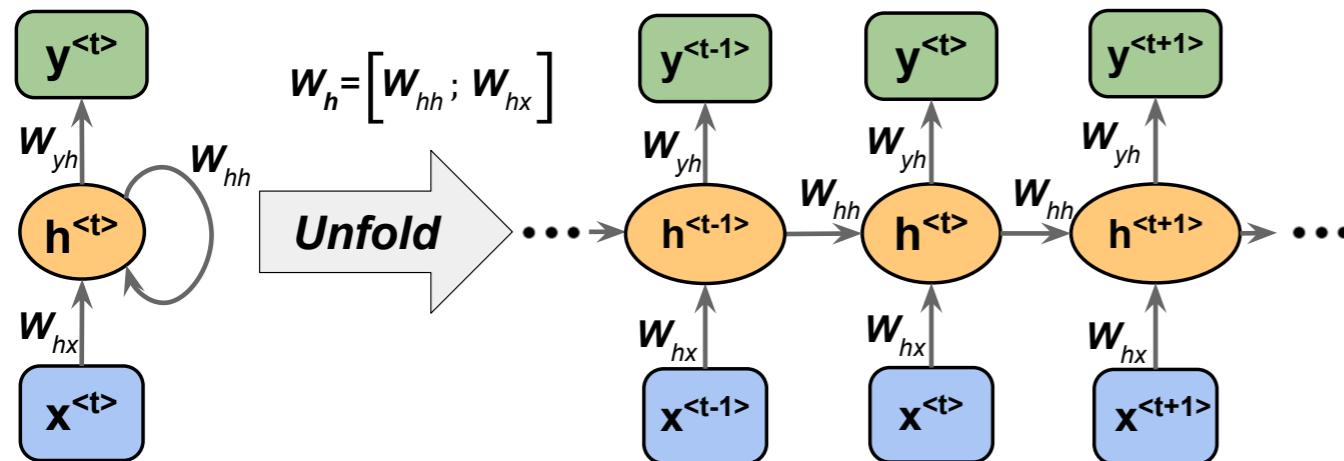
Werbos, Paul J. "[Backpropagation through time: what it does and how to do it.](#)"
Proceedings of the IEEE 78, no. 10 (1990): 1550-1560.

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

computed as a multiplication of adjacent time steps:

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

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.

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

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 *backpropagation 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:

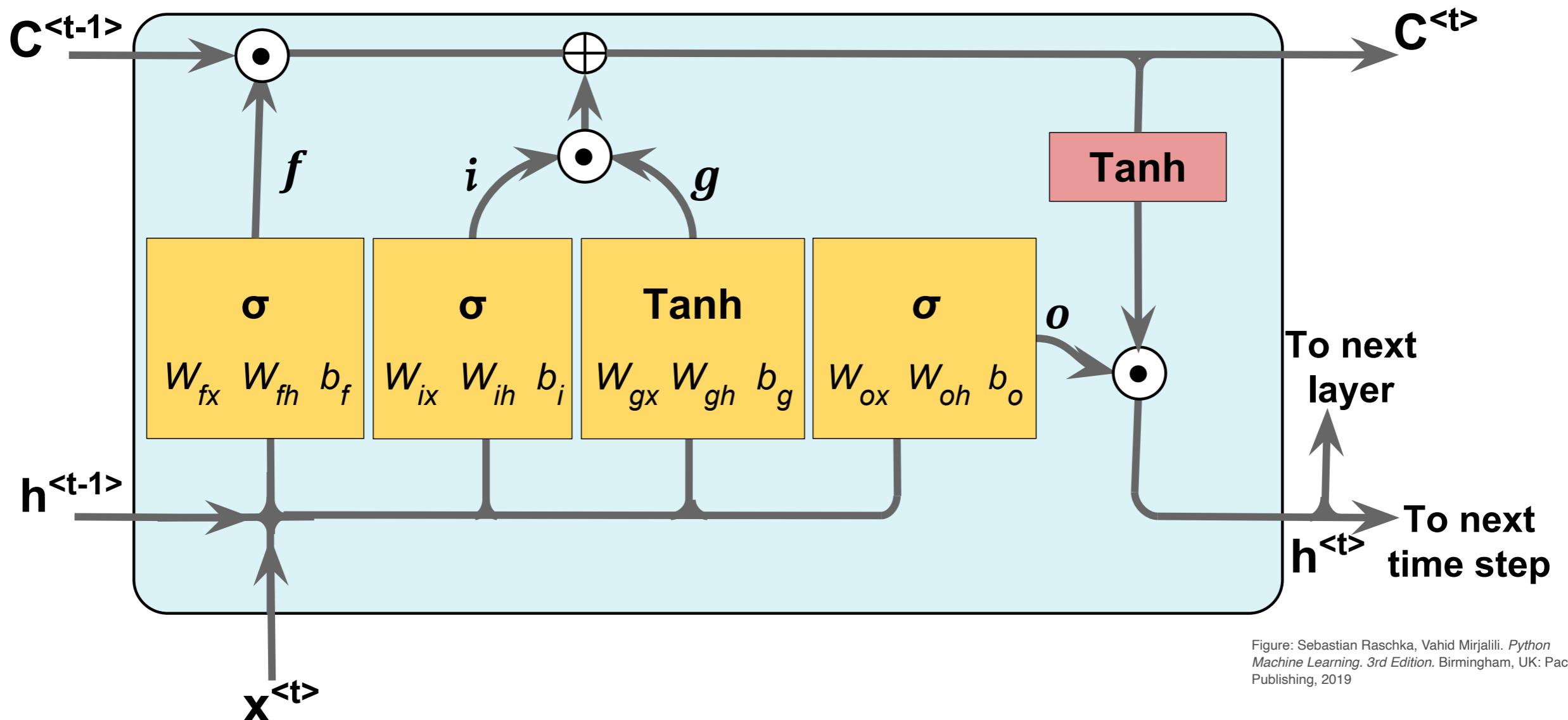
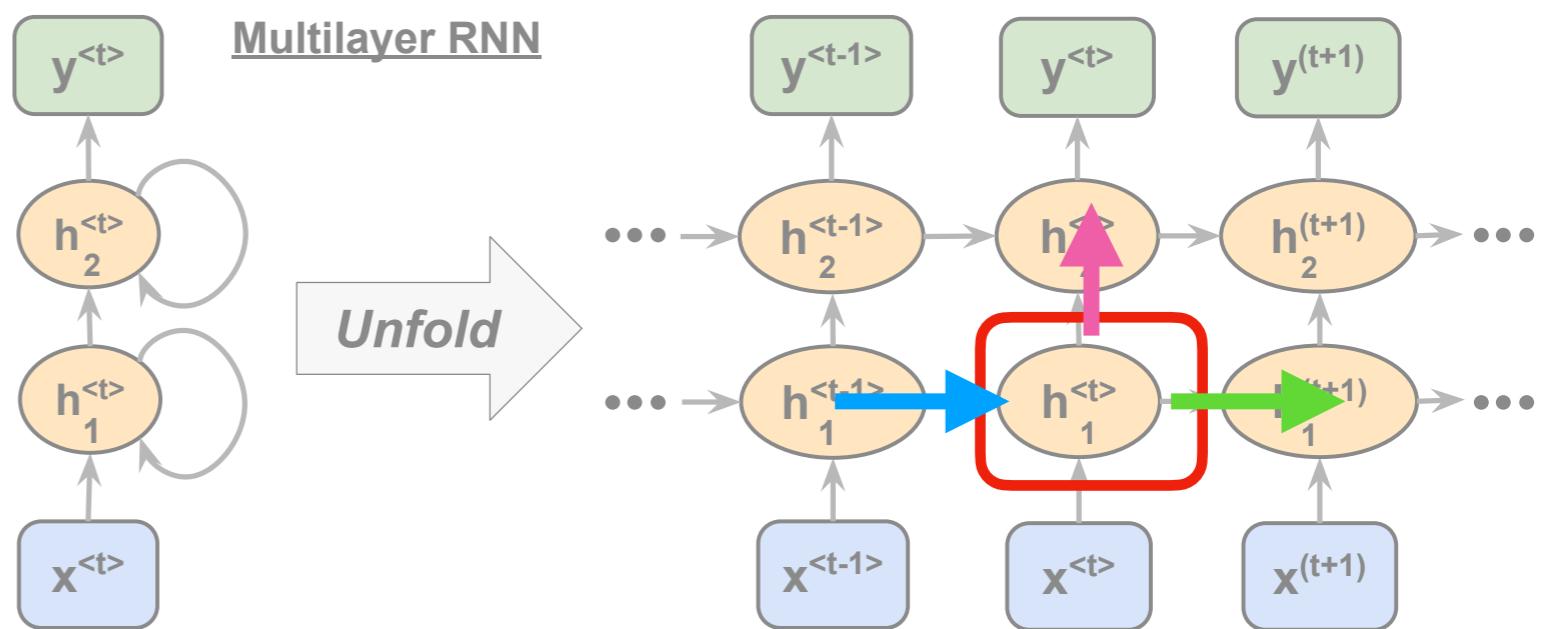
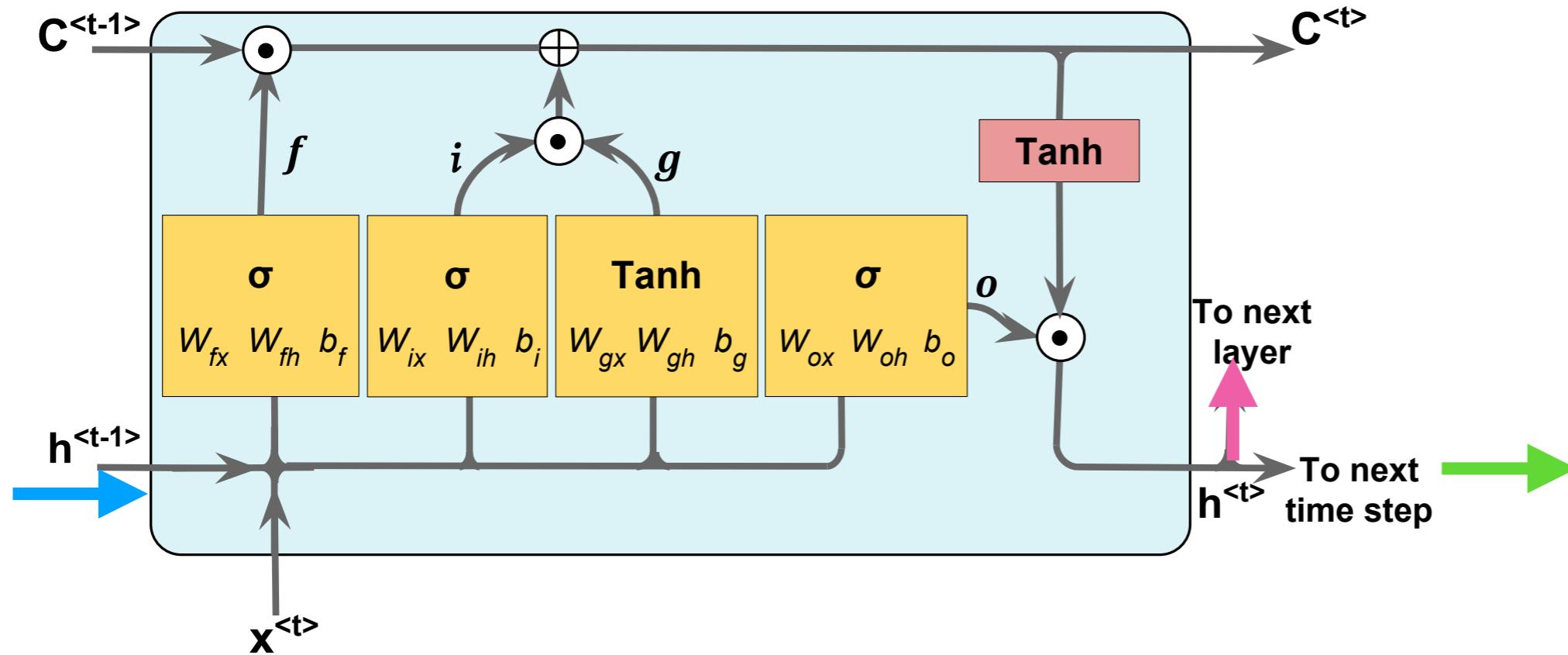


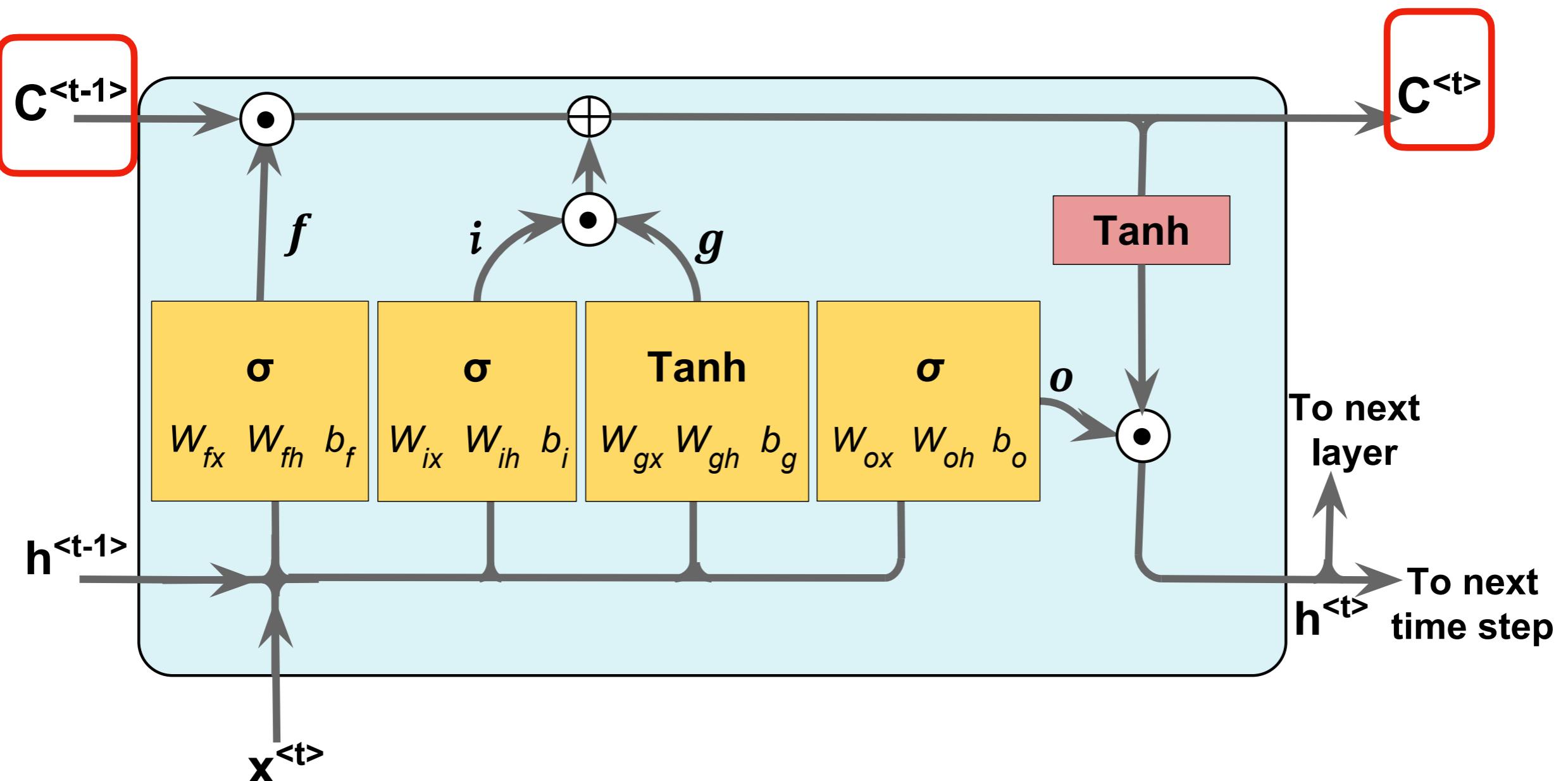
Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019



Long-short term memory (LSTM)

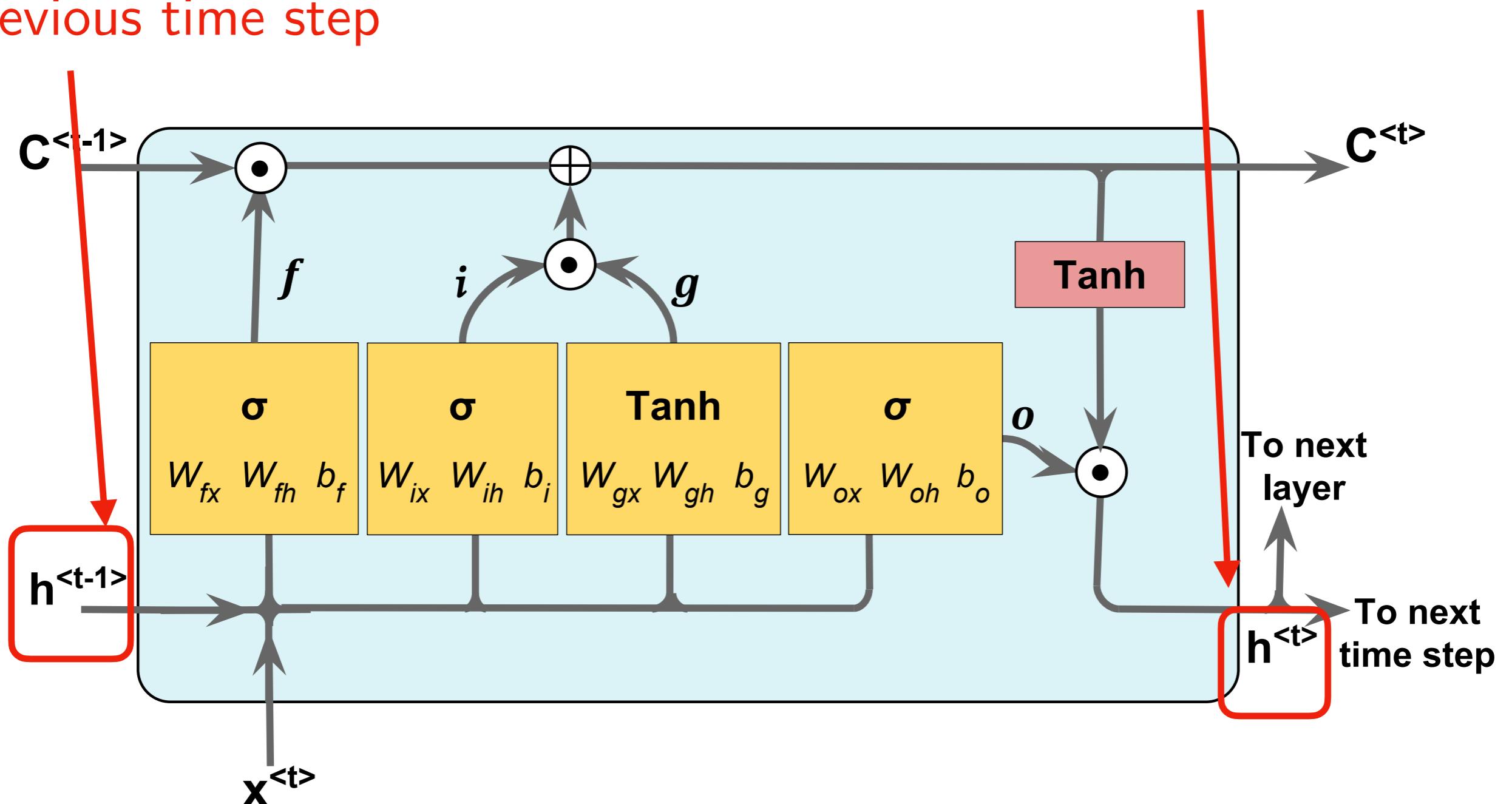
Cell state at previous time step

Cell state at current time step



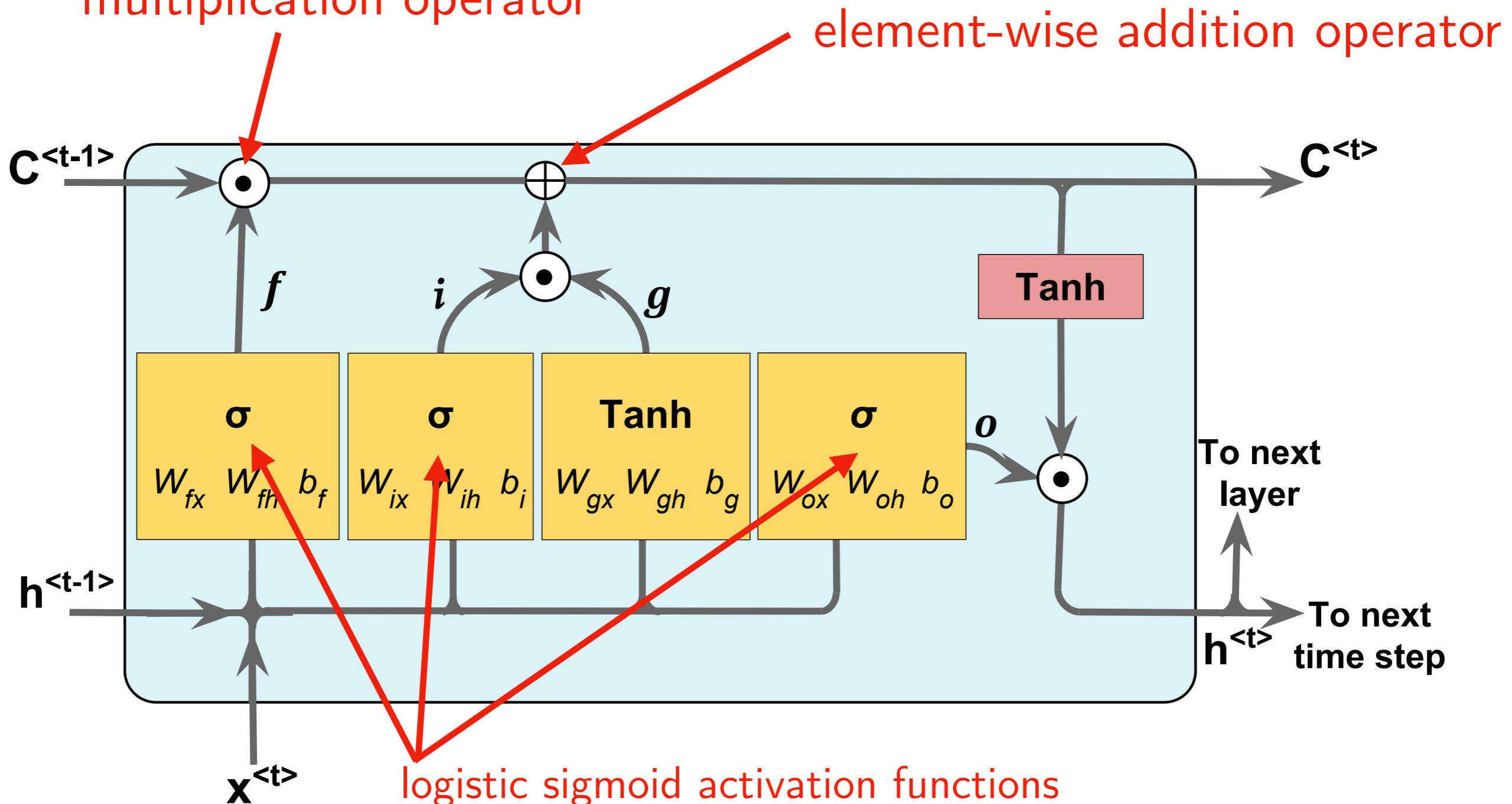
Long-short term memory (LSTM)

activation from
previous time step activation for next time step



Long-short term memory (LSTM)

element-wise
multiplication operator

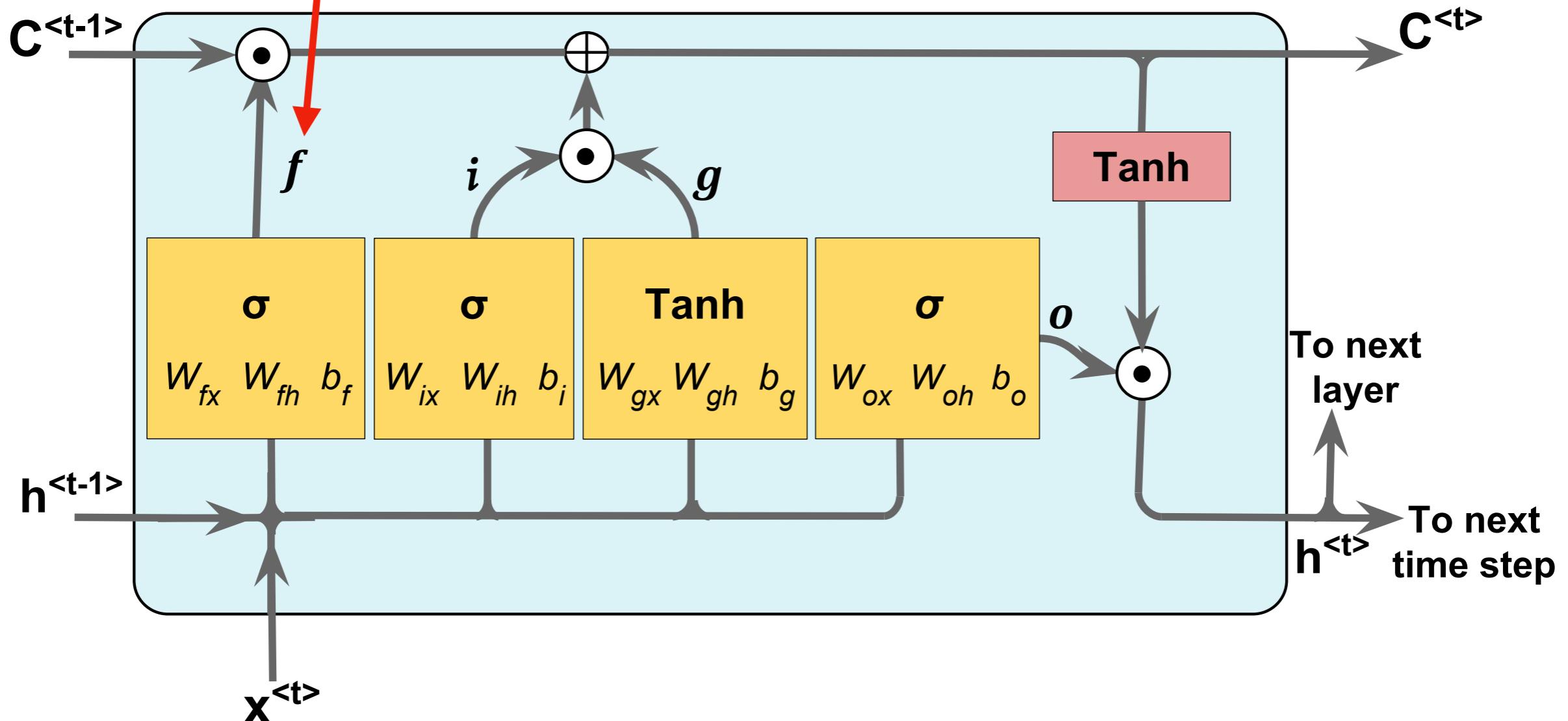


Long-short term memory (LSTM)

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

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

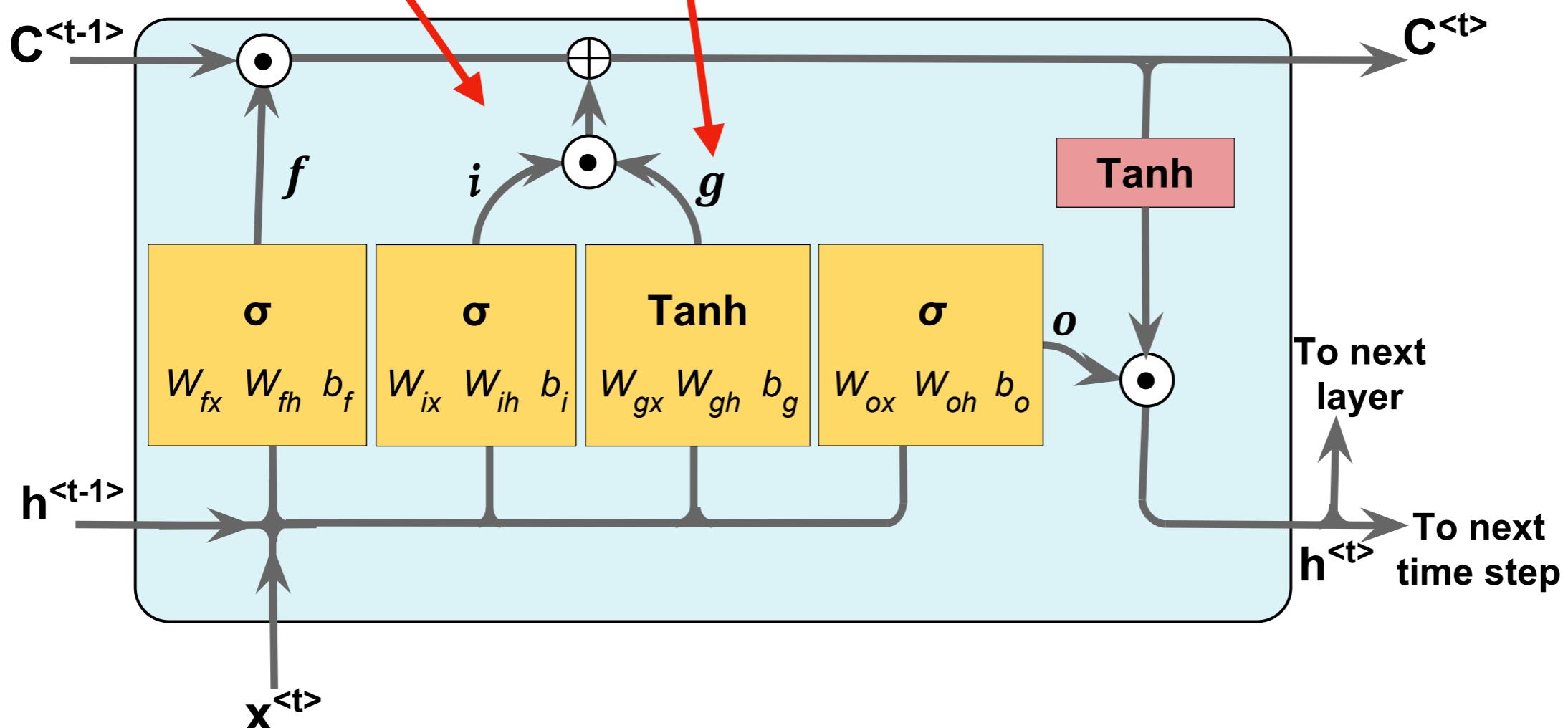


Long-short term memory (LSTM)

"Input Gate": $i_t = \sigma(\mathbf{W}_{ix}\mathbf{x}^{(t)} + \mathbf{W}_{ih}\mathbf{h}^{(t-1)} + \mathbf{b}_i)$

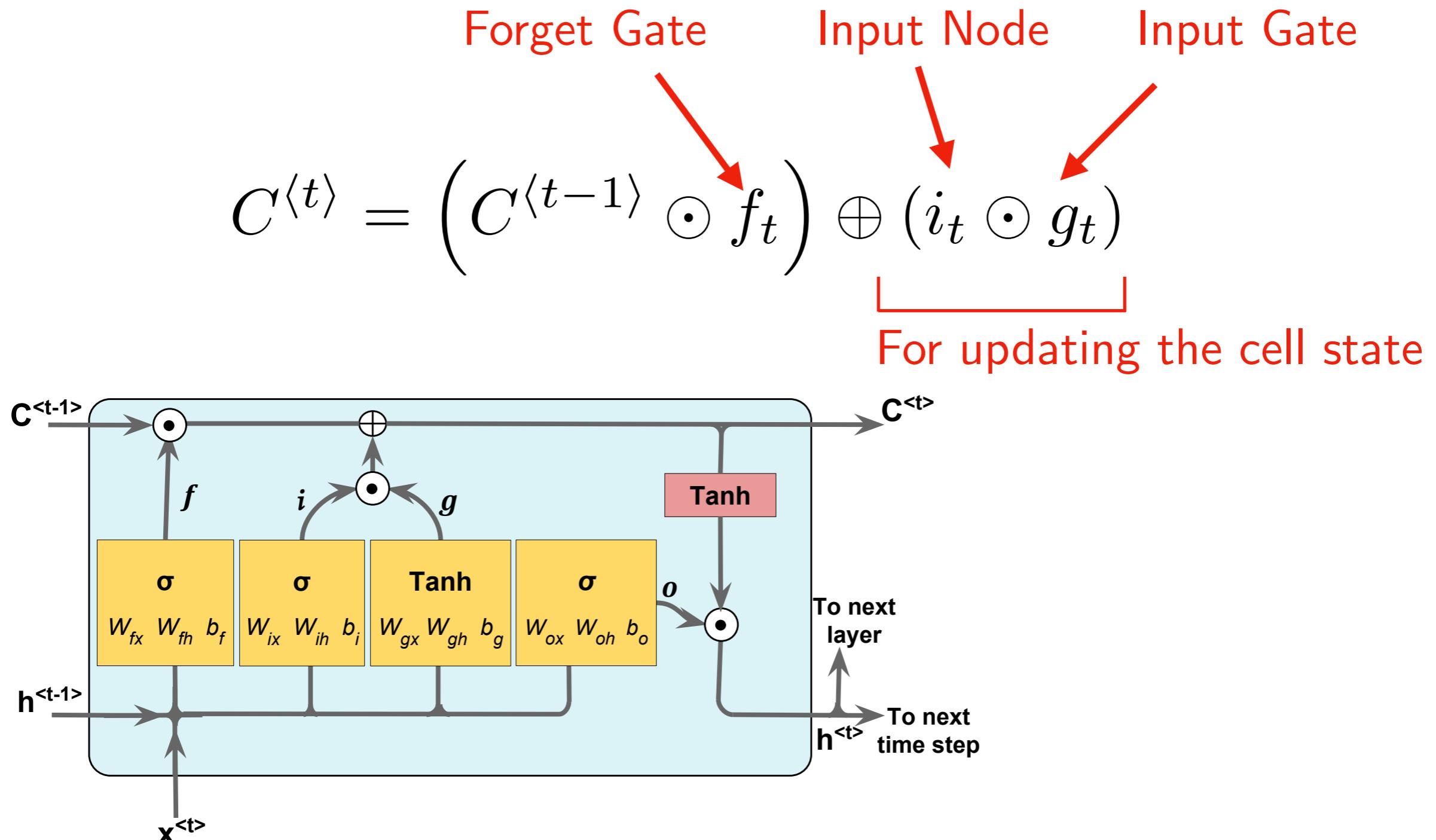
"Input Node":

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



Long-short term memory (LSTM)

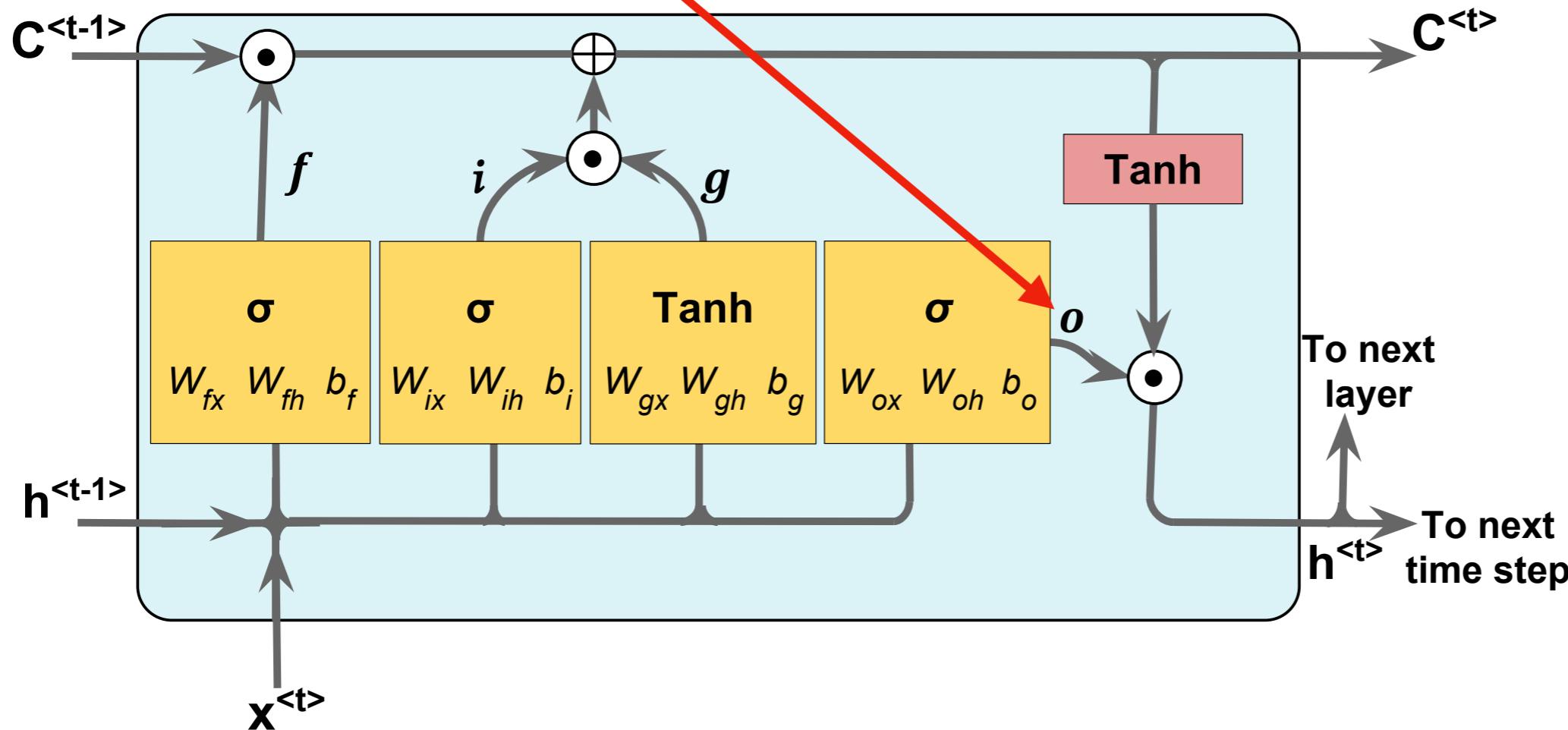
Brief summary of the gates so far ...



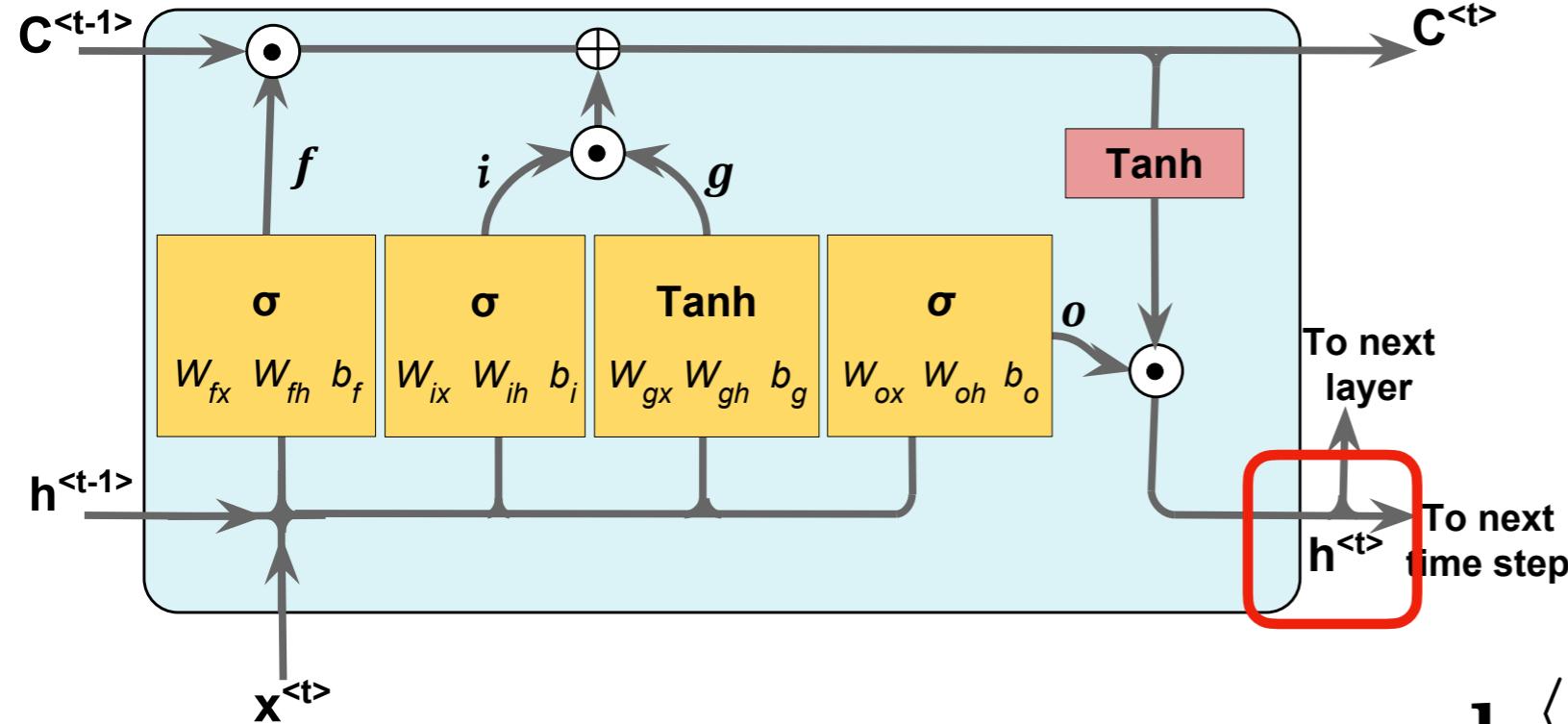
Long-short term memory (LSTM)

Output gate for updating the values of hidden units:

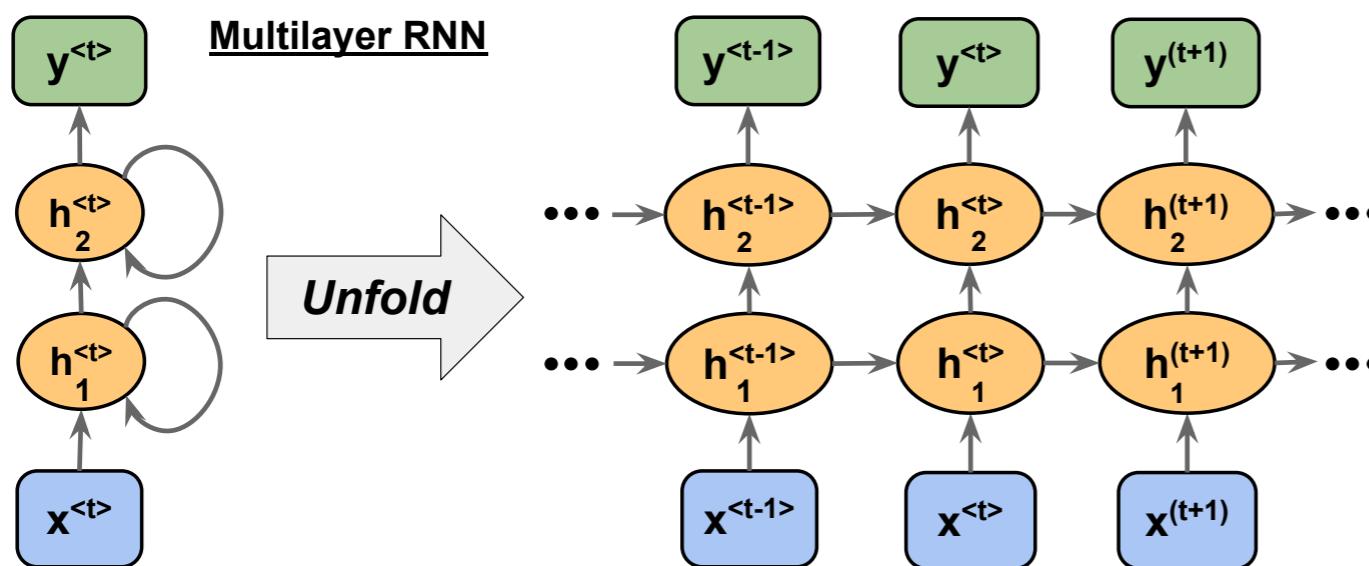
$$\mathbf{o}_t = \sigma \left(\mathbf{W}_{ox} \mathbf{x}^{(t)} + \mathbf{W}_{oh} \mathbf{h}^{(t-1)} + \mathbf{b}_o \right)$$



Long-short term memory (LSTM)

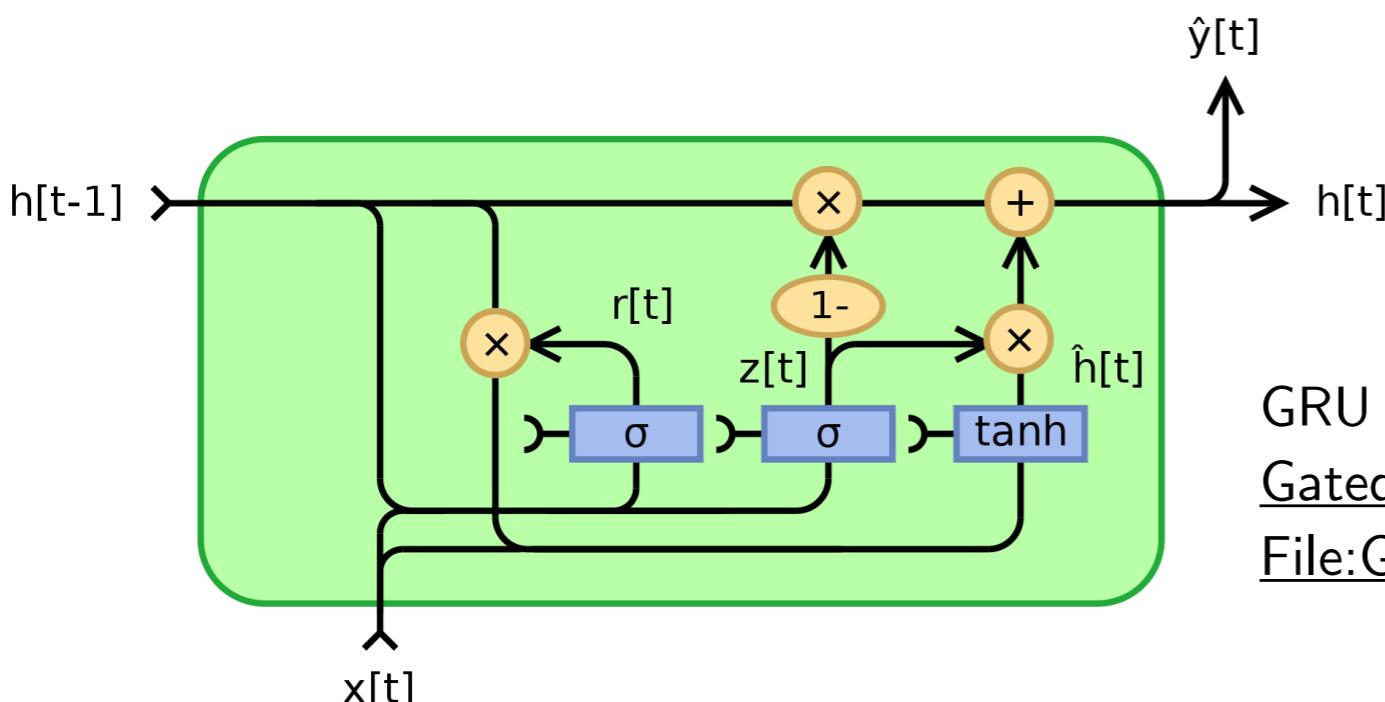


$$\mathbf{h}^{(t)} = \mathbf{o}_t \odot \tanh(\mathbf{C}^{(t)})$$



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)
```

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

```
...
    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)
```

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

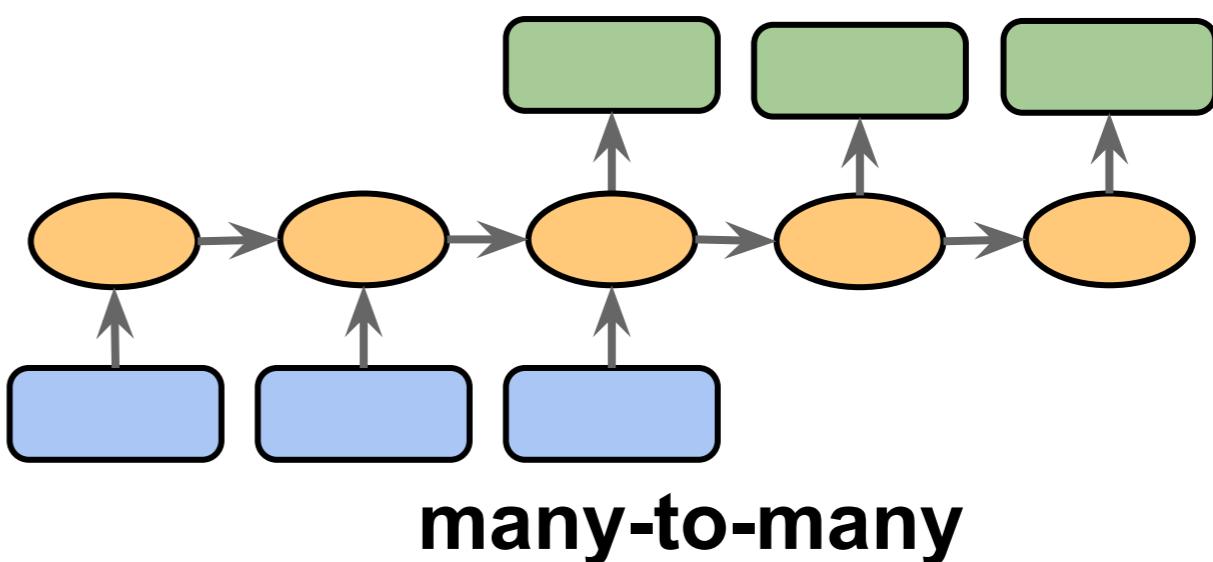
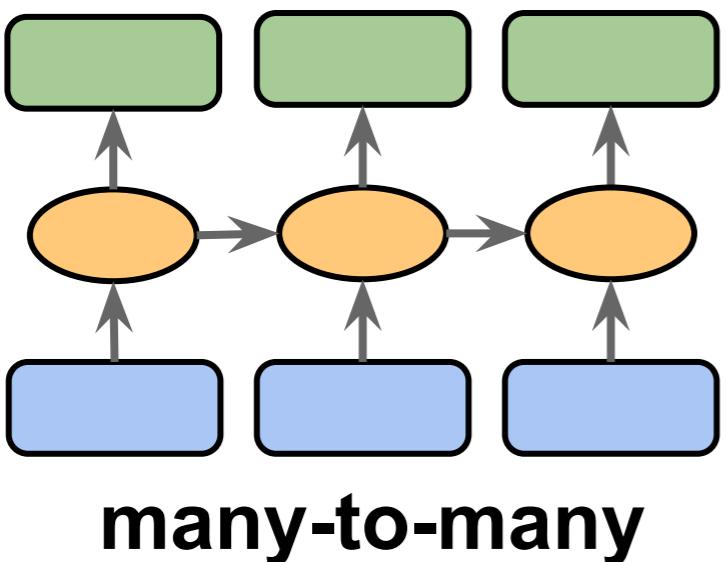
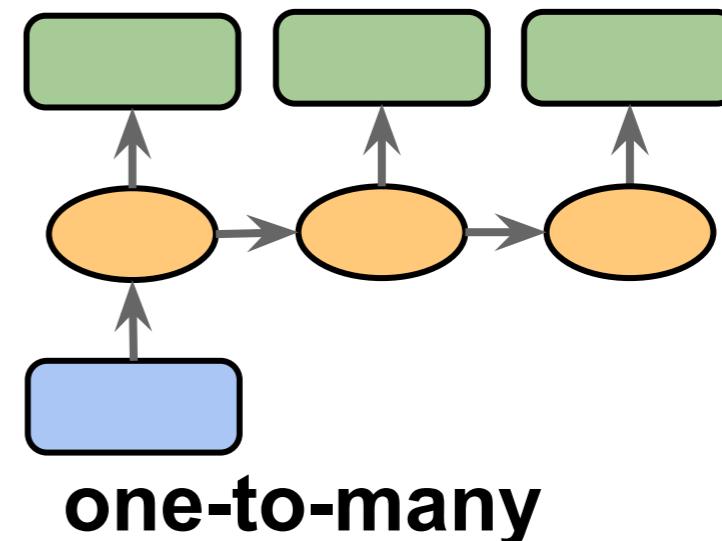
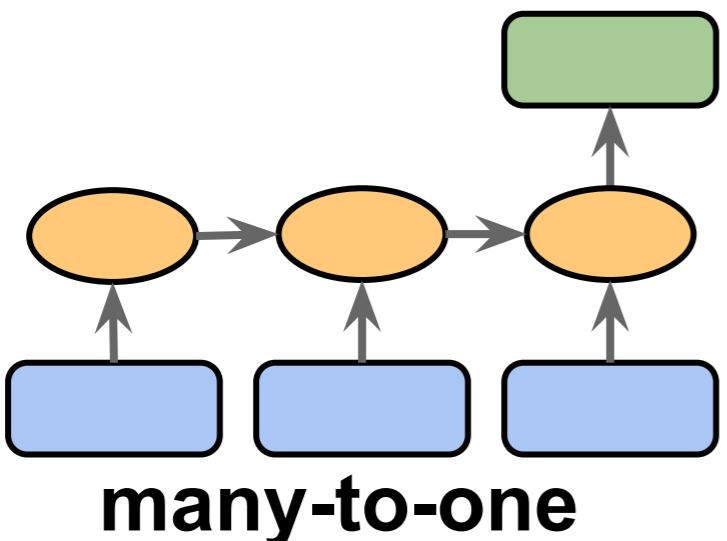


Figure based on:

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

Many-to-One

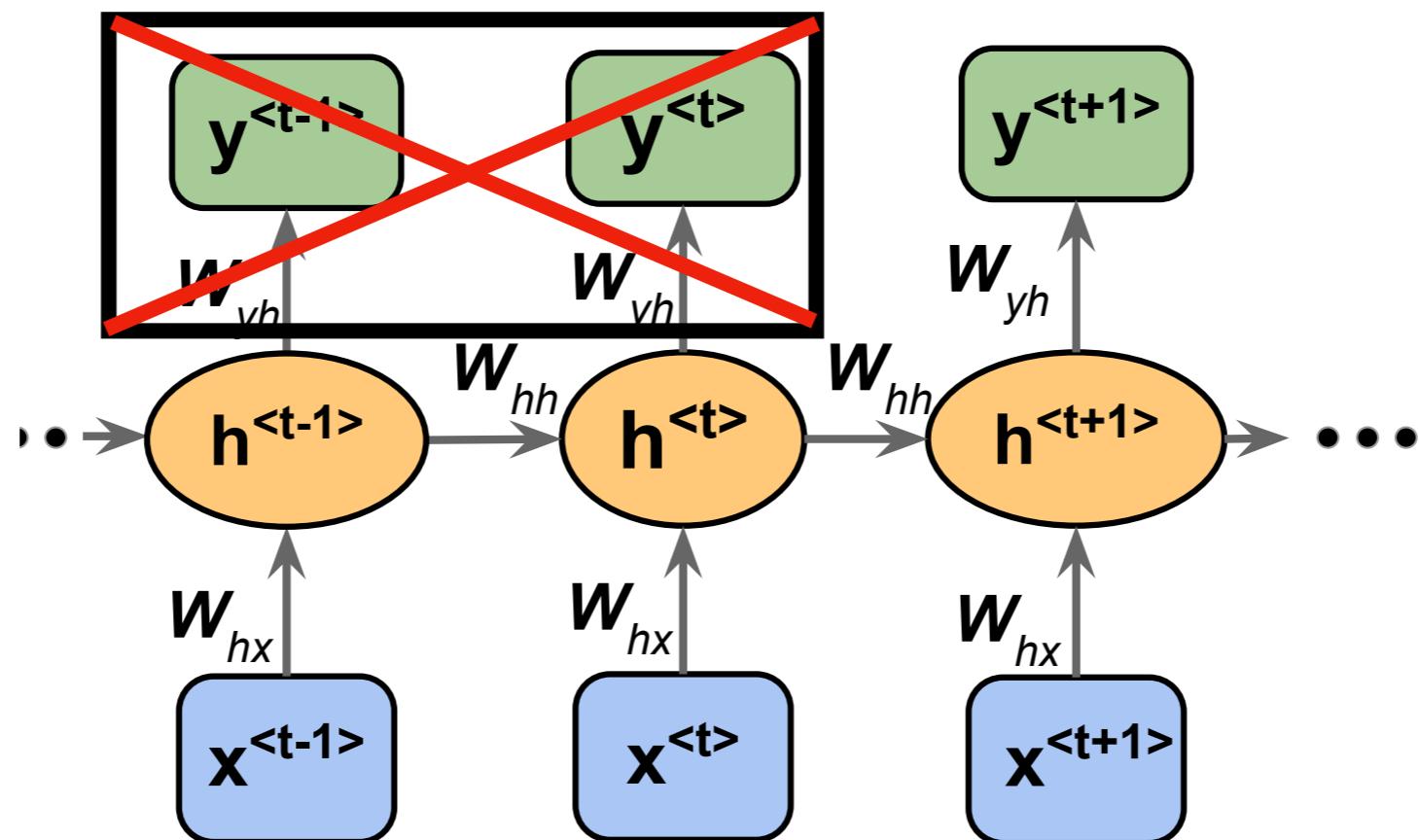


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Code Examples

 [rasbt / stat453-deep-learning-ss20](#) Unwatch ▾ 32 Star

Code Issues 0 Pull requests 2 Actions Projects 0 Wiki Security Insights

Branch: master stat453-deep-learning-ss20 / L14-rnns / code / Create new file Upload file

 rasbt upload rnns Latest commit 2

..

 01_rnn_simple_imdb.ipynb	upload rnns
 02_rnn_lstm_packed_imdb.ipynb	upload rnns
 03_rnn_lstm_packed_own_csv_imdb.ipynb	upload rnns
 04_char_rnn-charlesdickens.ipynb	upload rnns

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

Search or jump to... Pull requests Issues Marketplace Explore

bentrevett / [pytorch-sentiment-analysis](#) Watch 62 Unstar 1.7k Fork 440

Code Issues 6 Pull requests 0 Actions Projects 1 Wiki Security Insights

Tutorials on getting started with PyTorch and TorchText for sentiment analysis.

pytorch sentiment-analysis tutorial rnn lstm fasttext torchtext sentiment-classification cnn cnn-text-classification
lstm-sentiment-analysis pytorch-tutorial pytorch-tutorials pytorch-nlp nlp natural-language-processing recurrent-neural-networks
word-embeddings transformers bert

85 commits 1 branch 0 packages 0 releases 2 contributors MIT

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

	bentrevett clarified loading own datasets	Latest commit 79bb86a on Jan 28	
	assets fixed typos in max pool figure and size of tensors after convolutional...	12 months ago	
	custom_embeddings added appendix c - handling embeddings	13 months ago	
	data added optional appendix for how to use your own dataset with torchtext	2 years ago	
	.gitignore updated gitignore for vs code	5 months ago	
	1 - Simple Sentiment Analysis.ipynb reran all notebooks with latest pytorch and torchtext to ensure still...	7 months ago	
	2 - Upgraded Sentiment Analysis.... added explicit notes to copy embeddings using weight.data and not weight	7 months ago	
	3 - Faster Sentiment Analysis.ipynb reran all notebooks with latest pytorch and torchtext to ensure still...	7 months ago	
	4 - Convolutional Sentiment Anal... added batch_first to remove a permute	5 months ago	
	5 - Multi-class Sentiment Analysi... added model.eval() in predict sentiment functions (#31)	12 months ago	

<https://github.com/bentrevett/pytorch-sentiment-analysis>

Many-to-One ("Word"-RNN)

1 - Simple Sentiment Analysis (Simple RNN)

<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/1%20-%20Simple%20Sentiment%20Analysis.ipynb>

2 - Updated Sentiment Analysis (LSTM)

<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/2%20-%20Upgraded%20Sentiment%20Analysis.ipynb>

Optional:

Using TorchText with Your Own Datasets

<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/A%20-%20Using%20TorchText%20with%20Your%20Own%20Datasets.ipynb>

4 - Convolutional Sentiment Analysis

<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/4%20-%20Convolutional%20Sentiment%20Analysis.ipynb>

Many-to-One ("Character"-RNN)

Classifying Names with a Character-level RNN

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

Lecture Overview

RNNs and Sequence Modeling Tasks

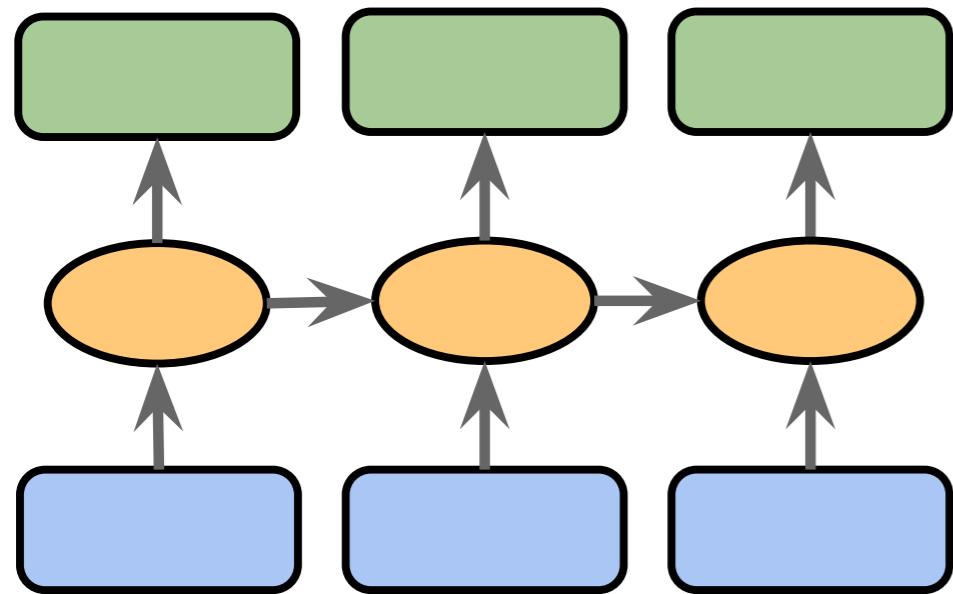
Backpropagation Through Time

Long-short term memory (LSTM)

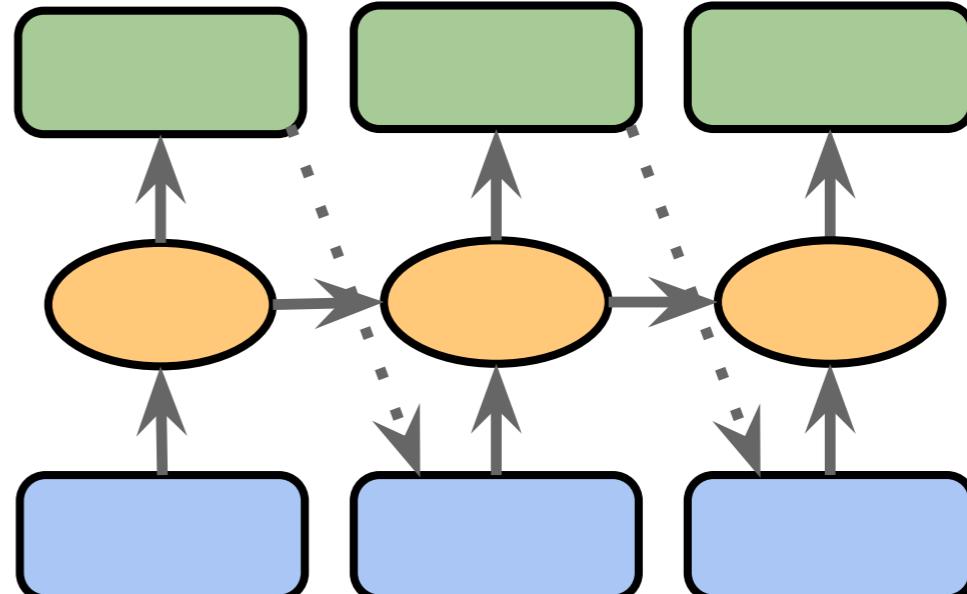
Many-to-one Word RNNs

Generating Text with Character RNNs

Attention Mechanisms and Transformers

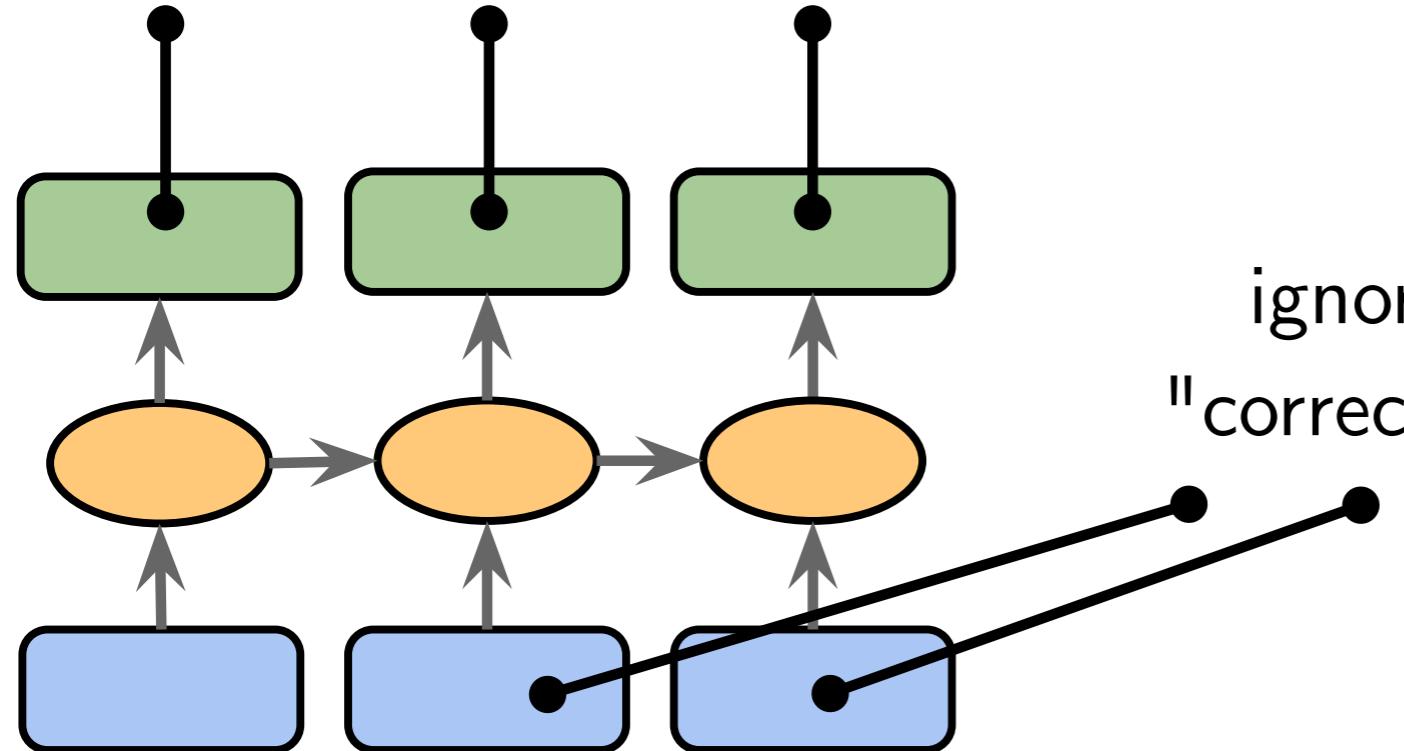


many-to-many
"training"



~~**many-to-many**~~
"one"
"generating new text"

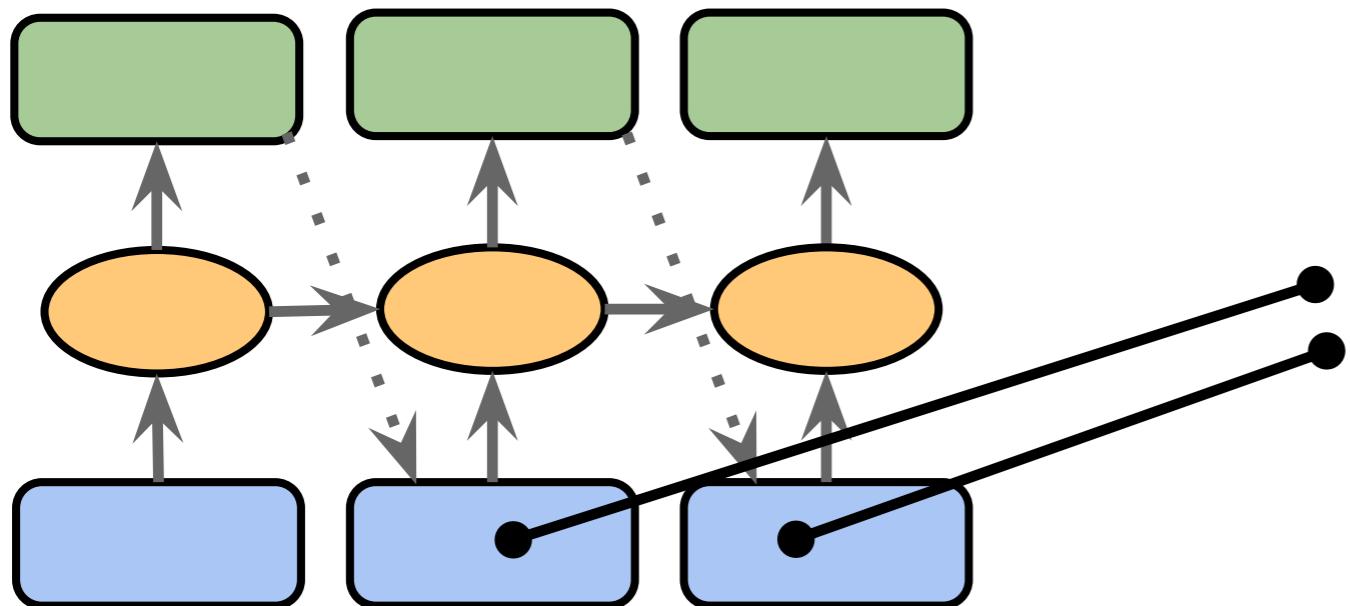
At each time step
Softmax output (probability)
for each possible "next letter"



many-to-many

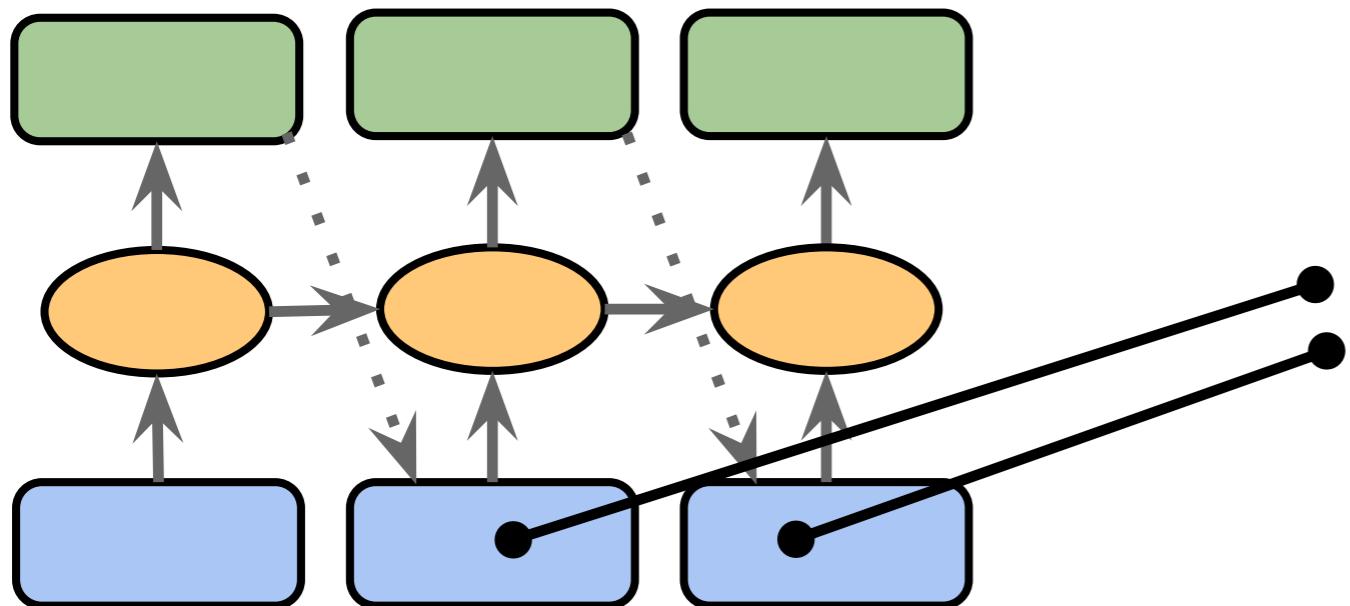
"training"

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



~~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"

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

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

Code Examples

 [rasbt / stat453-deep-learning-ss20](#) Unwatch ▾ 32 Star

Code Issues 0 Pull requests 2 Actions Projects 0 Wiki Security Insights

Branch: master stat453-deep-learning-ss20 / L14-rnns / code / Create new file Upload file

 rasbt upload rnns Latest commit 2

..

01_rnn_simple_imdb.ipynb	upload rnns
02_rnn_lstm_packed_imdb.ipynb	upload rnns
03_rnn_lstm_packed_own_csv_imdb.ipynb	upload rnns
04_char_rnn-charlesdickens.ipynb	upload rnns

<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!@hp0%#nsqn&rql*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!@hp0%#nsqn&rql*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

Additional Tutorials:

<https://github.com/bentrevett/pytorch-seq2seq>

bentrevett / pytorch-seq2seq

Watch 42 Star 1.4k Fork 383

Code Issues 12 Pull requests 1 Actions Projects 1 Wiki Security Insights

Tutorials on implementing a few sequence-to-sequence (seq2seq) models with PyTorch and TorchText.

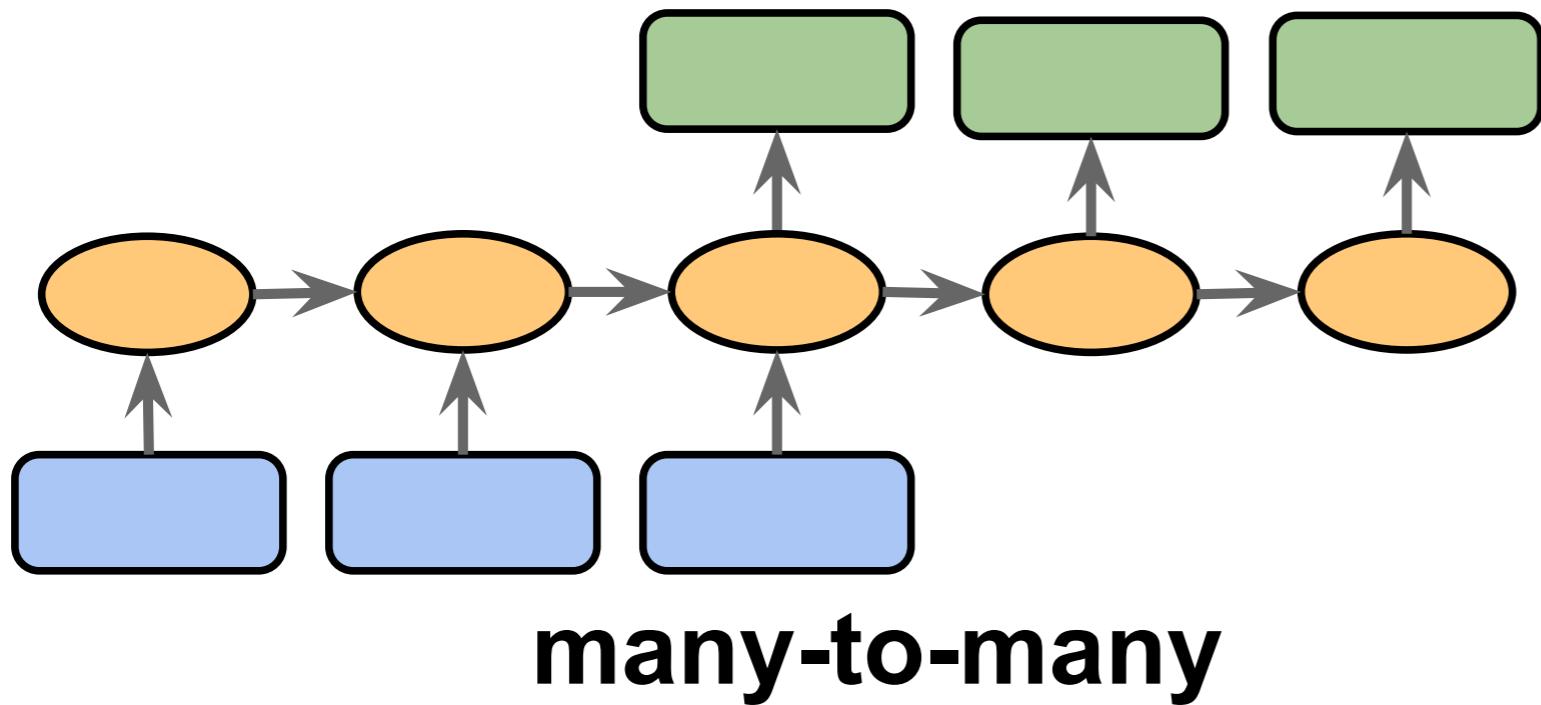
pytorch seq2seq sequence-to-sequence tutorial rnn gru lstm torchtext pytorch-tutorial pytorch-implmentation encoder-decoder encoder-decoder-model neural-machine-translation pytorch-seq2seq attention transformer cnn-seq2seq pytorch-implementation pytorch-tutorials pytorch-nlp

133 commits 1 branch 0 packages 0 releases 3 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

bentrevett fixed unsqueeze (#96) Latest commit 041994c yesterday

File	Description	Time Ago
assets	updated assets/images to include embedding layer	3 months ago
.gitignore	updated gitignore as models now saved in main directory	12 months ago
1 - Sequence to Sequence Learni...	clarified difference between tokens and embeddings of tokens	3 months ago
2 - Learning Phrase Representati...	clarified difference between tokens and embeddings of tokens	3 months ago
3 - Neural Machine Translation by...	updated notebooks to reflect change of v from tensor to linear layer	2 months ago
4 - Packed Padded Sequences, ...	(#79) fixed src_len tensor size in encoder	2 months ago
5 - Convolutional Sequence to Se...	(#87) fixed typos	22 days ago
6 - Attention is All You Need.ipynb	fixed unsqueeze (#96)	yesterday
LICENSE	Initial commit	2 years ago
README.md	updated required pytorch and torchtext	3 months ago



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/>)
Sebastian Raschka

STAT 453: Intro to Deep Learning and Generative Models

SS 2020

55

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 ..."

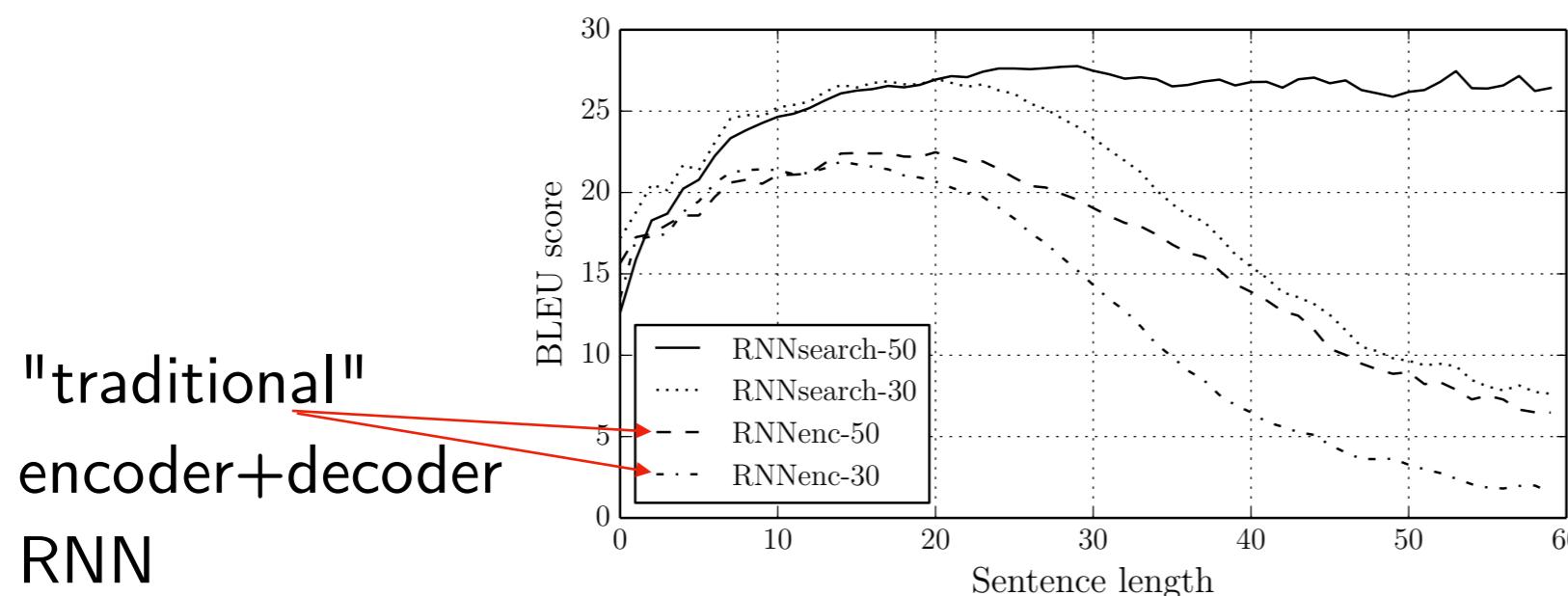


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

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")

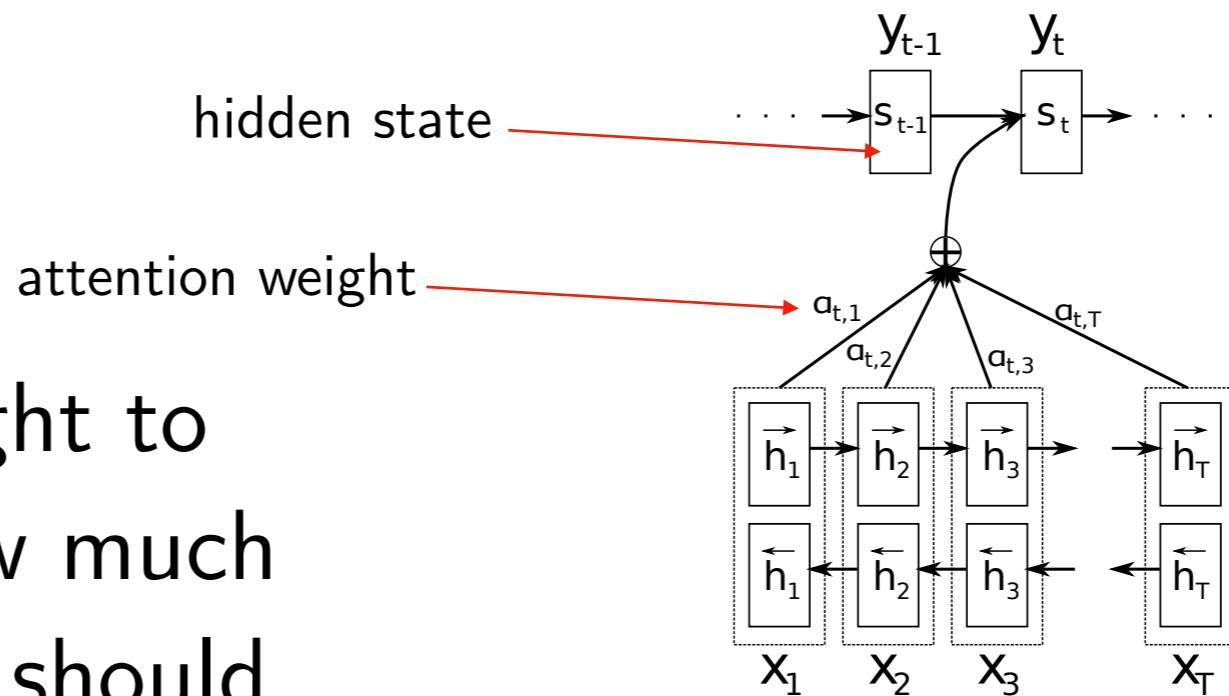
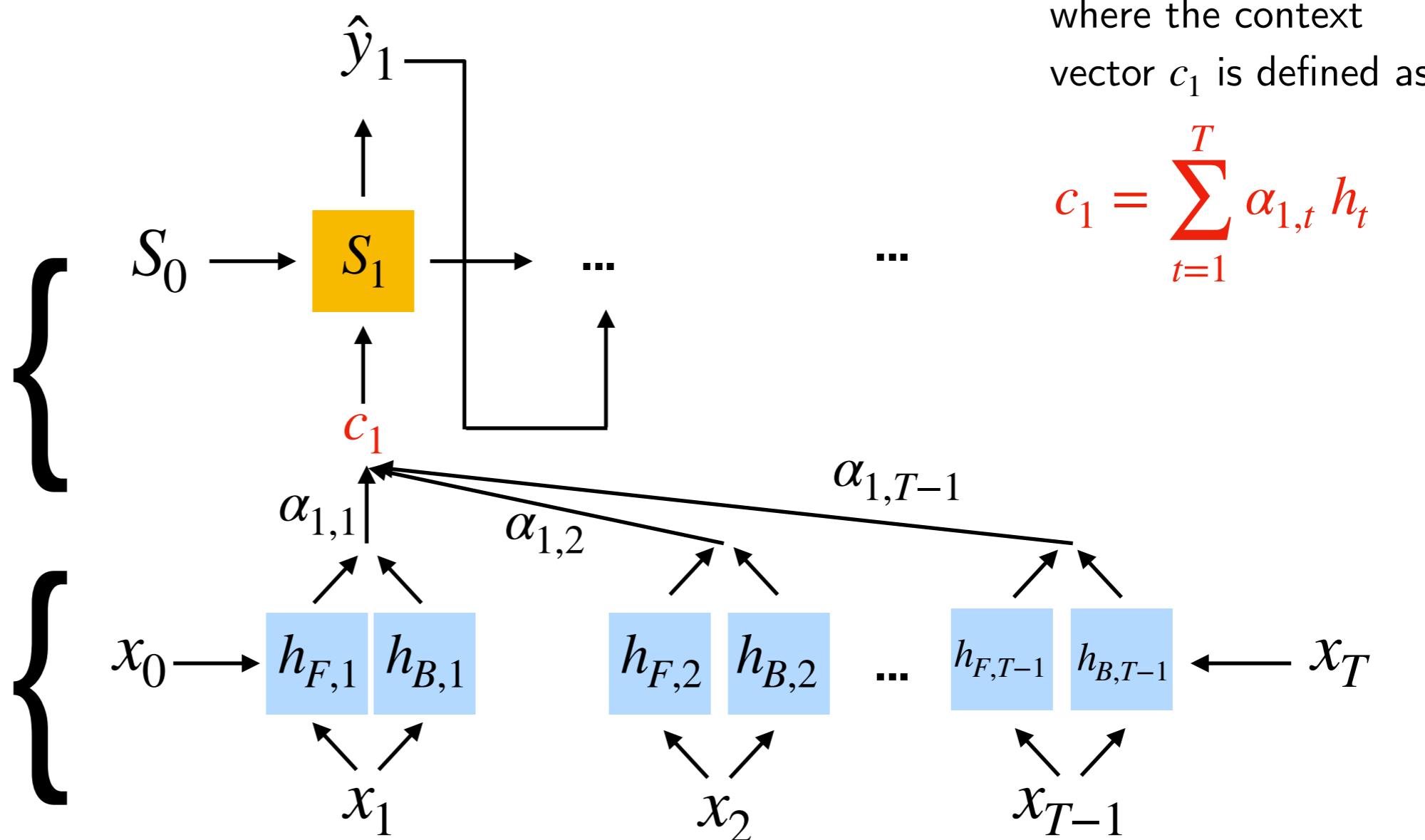


Figure 1: The graphical illustration of the proposed model trying to generate the t -th target word y_t given a source sentence (x_1, x_2, \dots, x_T) .

RNN Attention Mechanism

Added attention
(looks like a standard RNN but with context vectors as in-/output)

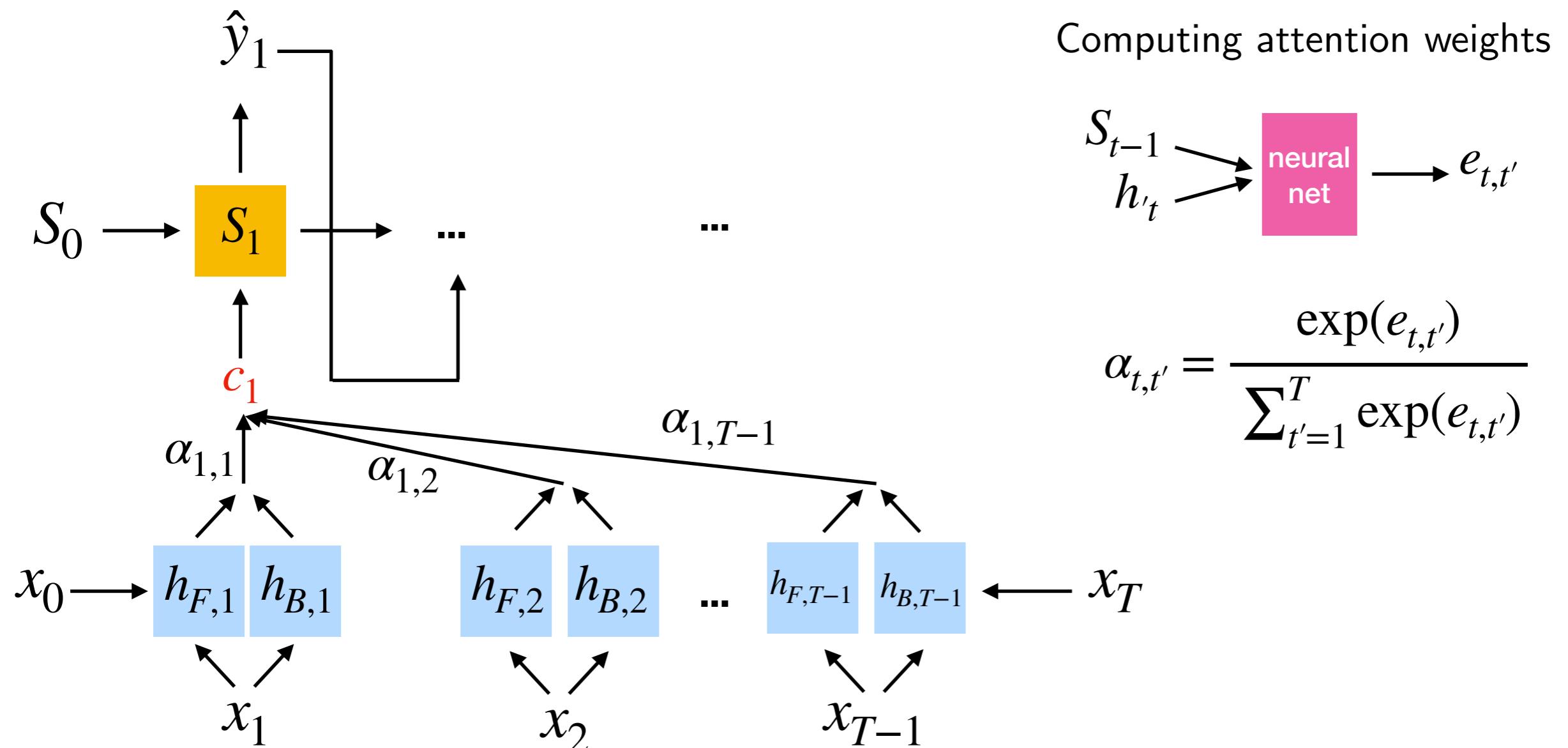
Bidirectional RNN



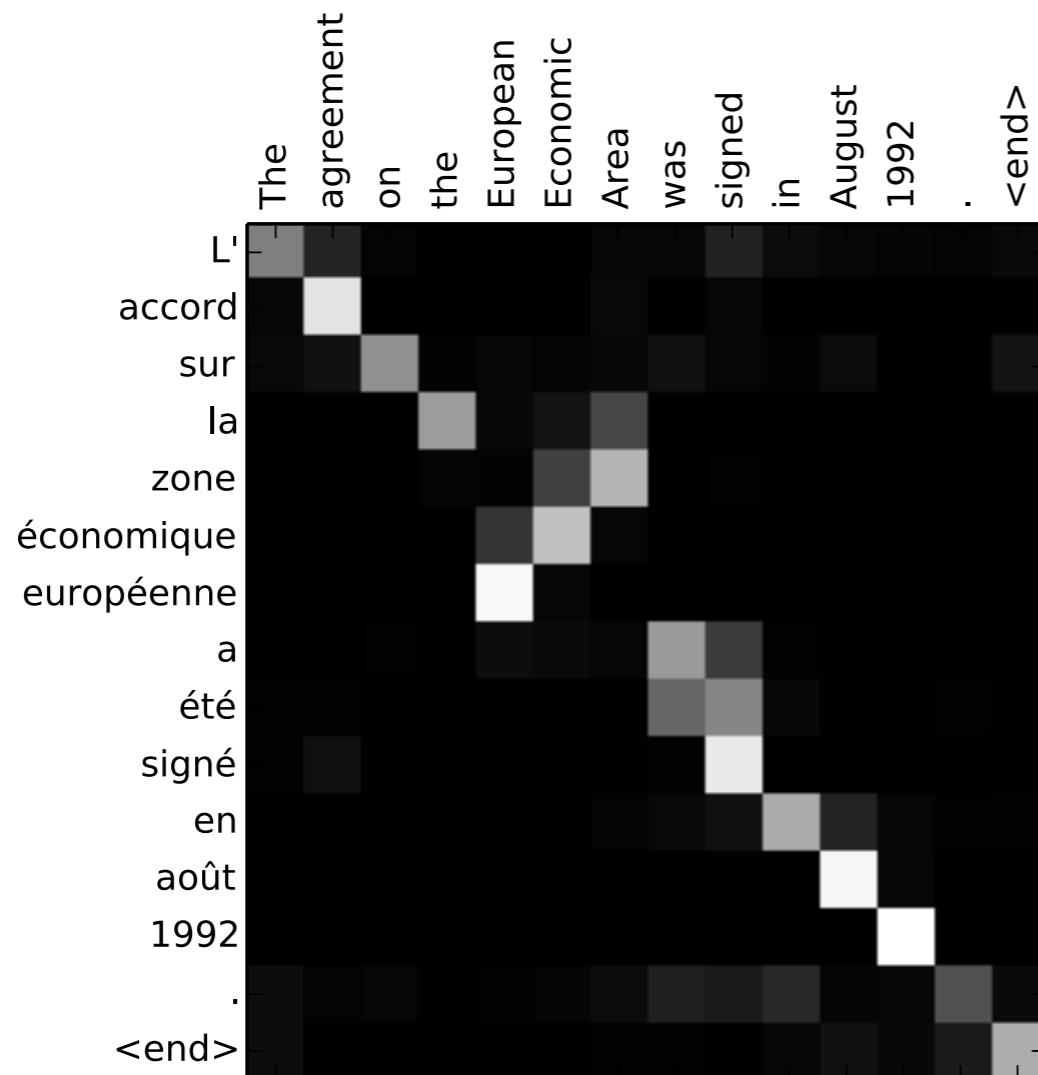
where the context vector c_1 is defined as

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

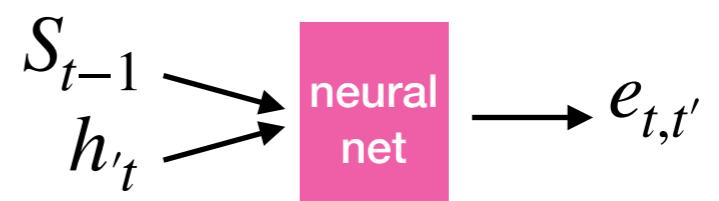
Attention Mechanism



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

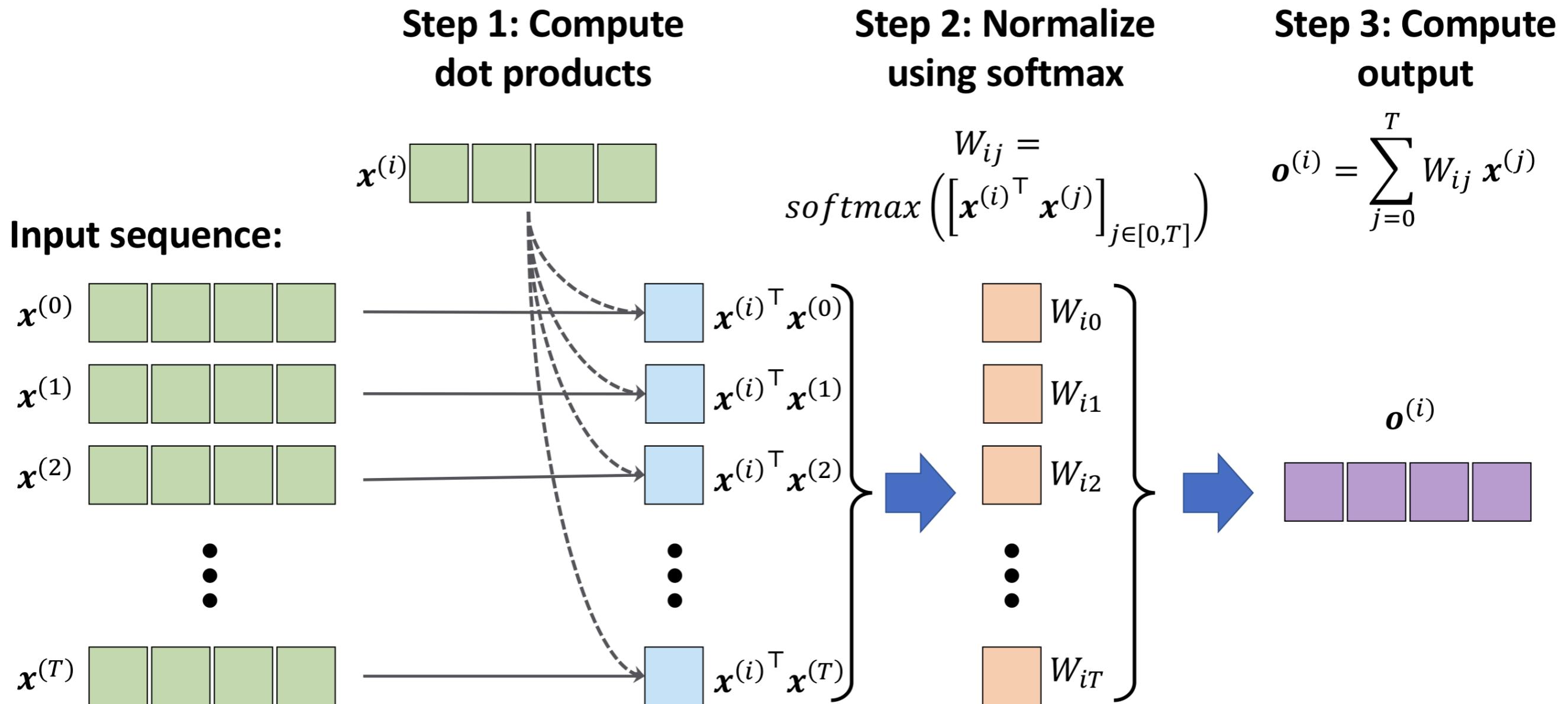


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Multi-head Attention and Transformer Blocks

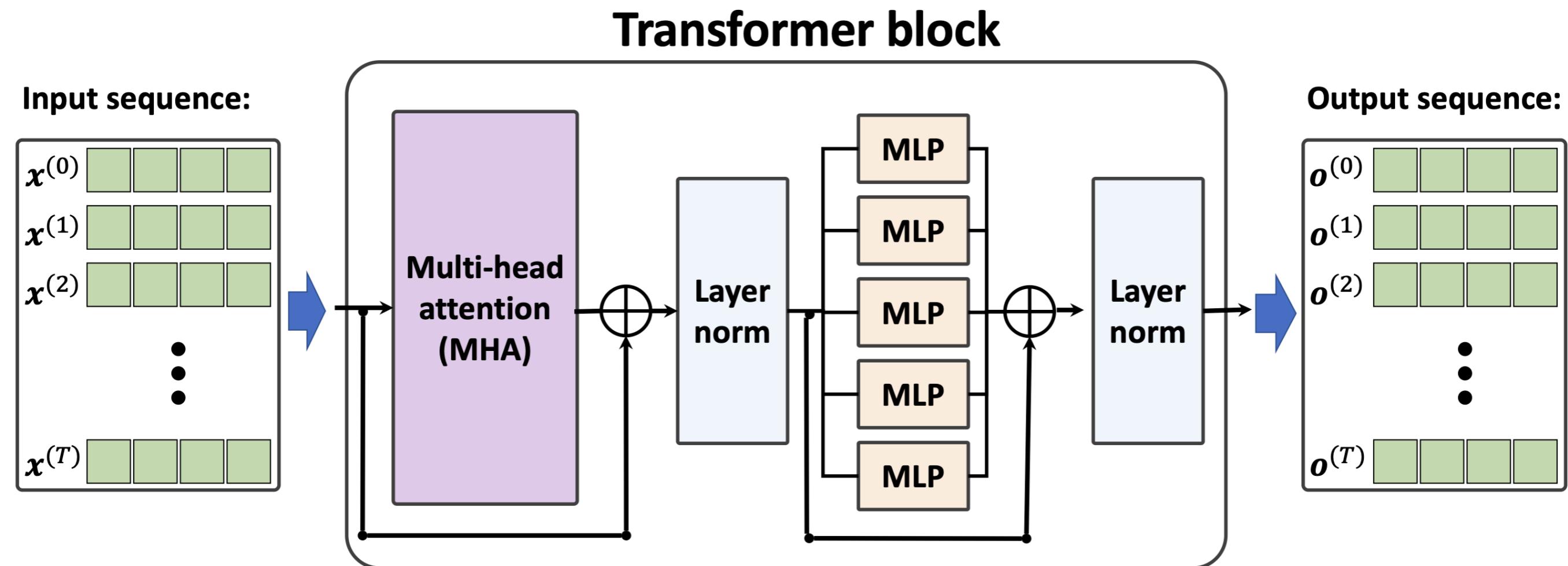


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Birmingham, UK: Packt Publishing, 2019

Implementations:

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

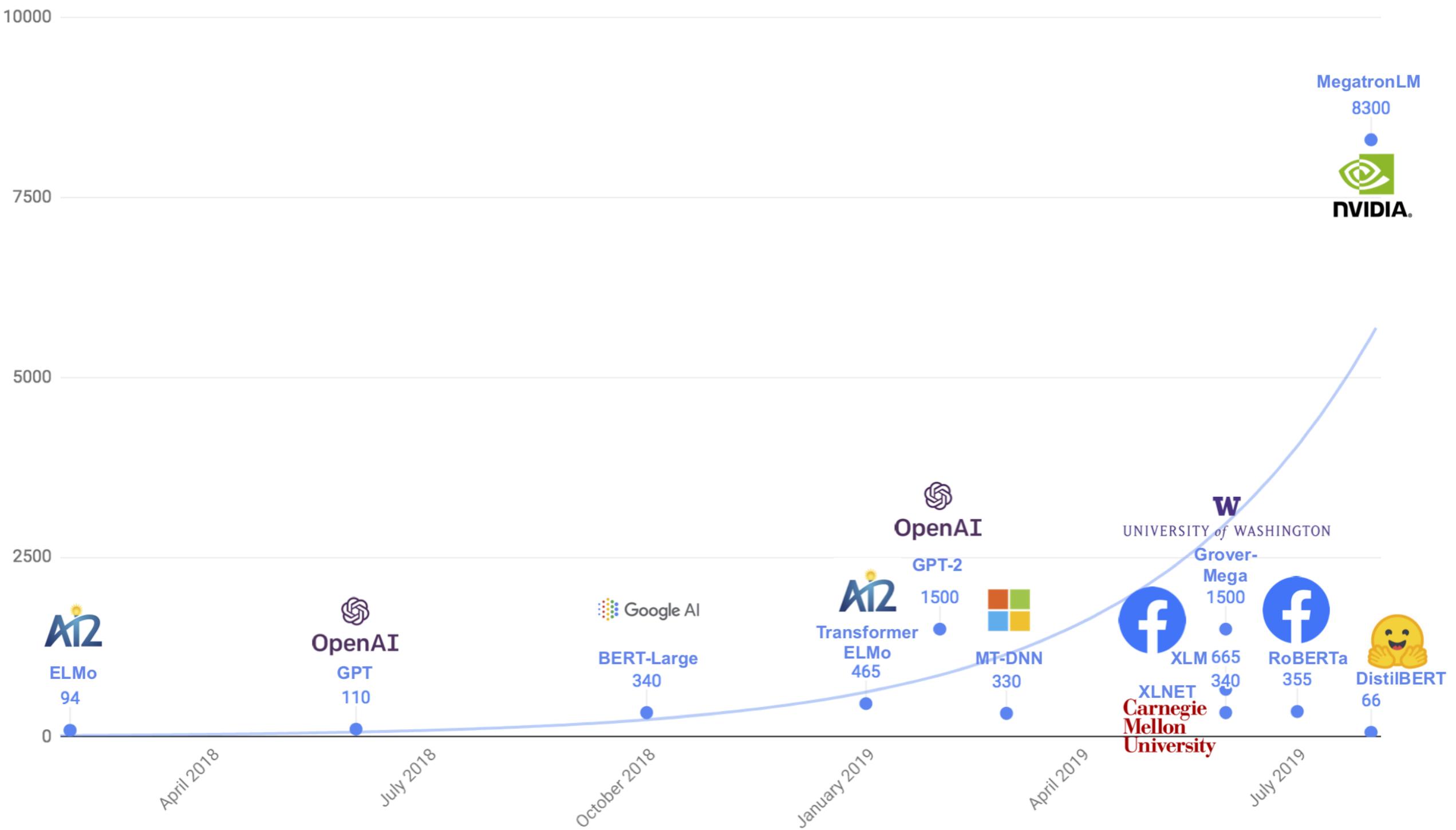


Image Source: <https://medium.com/huggingface/distilbert-8cf3380435b5>

Optional Reading Material

Chapter 10, Sequence Modeling: Recurrent and Recursive Nets:

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