

STAT 453: Introduction to Deep Learning and Generative Models

Sebastian Raschka

<http://stat.wisc.edu/~sraschka/teaching>



Lecture 15

Introduction to Recurrent Neural Networks

Lecture Overview

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

There's more than one way to bake a cake

- 1. Different Ways to Model Text**
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

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"

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

class labels

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

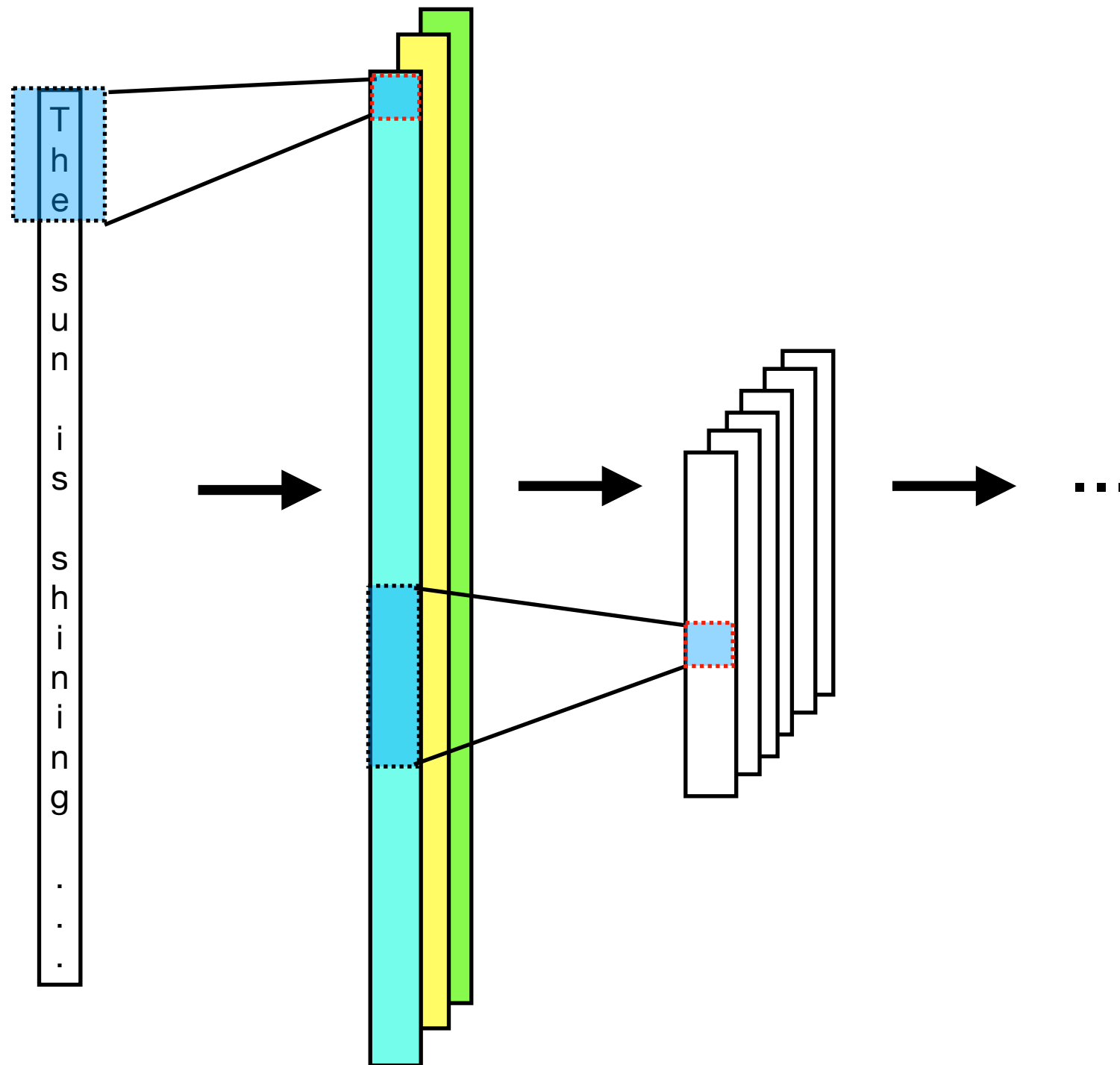
training

Classifier

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

Raschka & Mirjalili. *Python Machine Learning* 3rd Ed.
<https://github.com/rasbt/python-machine-learning-book-3rd-edition/blob/master/ch08/ch08.ipynb>

1D CNNs for text (and other sequence data)



Transformers & Self-Supervised Learning

Input sentence:

A quick brown fox jumps over the lazy dog



15% randomly masked:

A quick brown [MASK] jumps over the lazy dog



Possible classes
(all words)

0.2%	ant
...	...
11%	fox
...	...
0.01%	zoo



How Can We Modify MLPs to Capture Sequence Information?

1. Different Ways to Model Text
- 2. Sequence Modeling with RNNs**
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

Sequence data: order matters

The movie my friend has not seen is good

The movie my friend has seen is not good

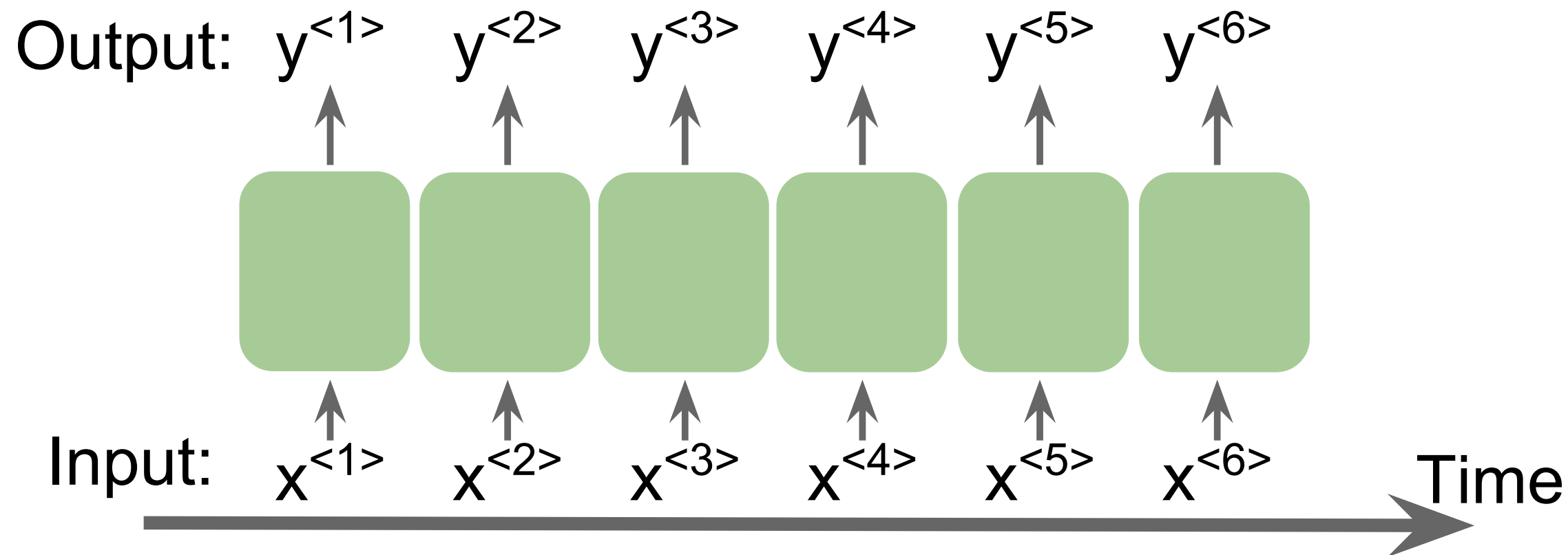


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning, 3rd Edition*. Packt, 2019

Applications: Working with Sequential Data

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

Stock market predictions

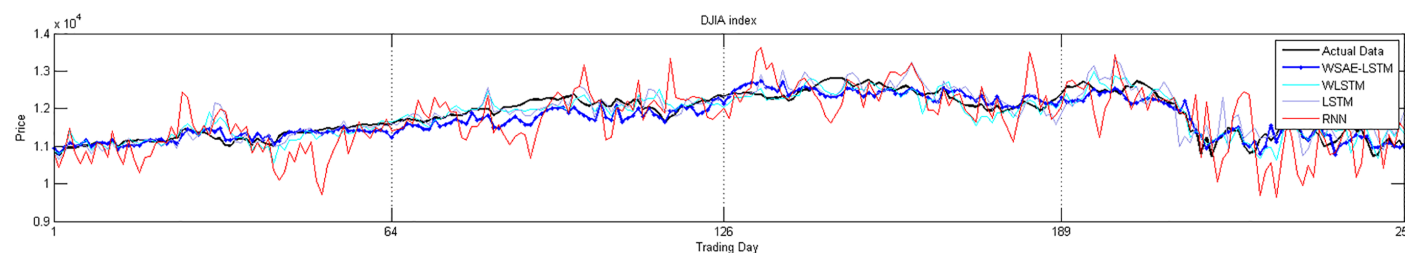
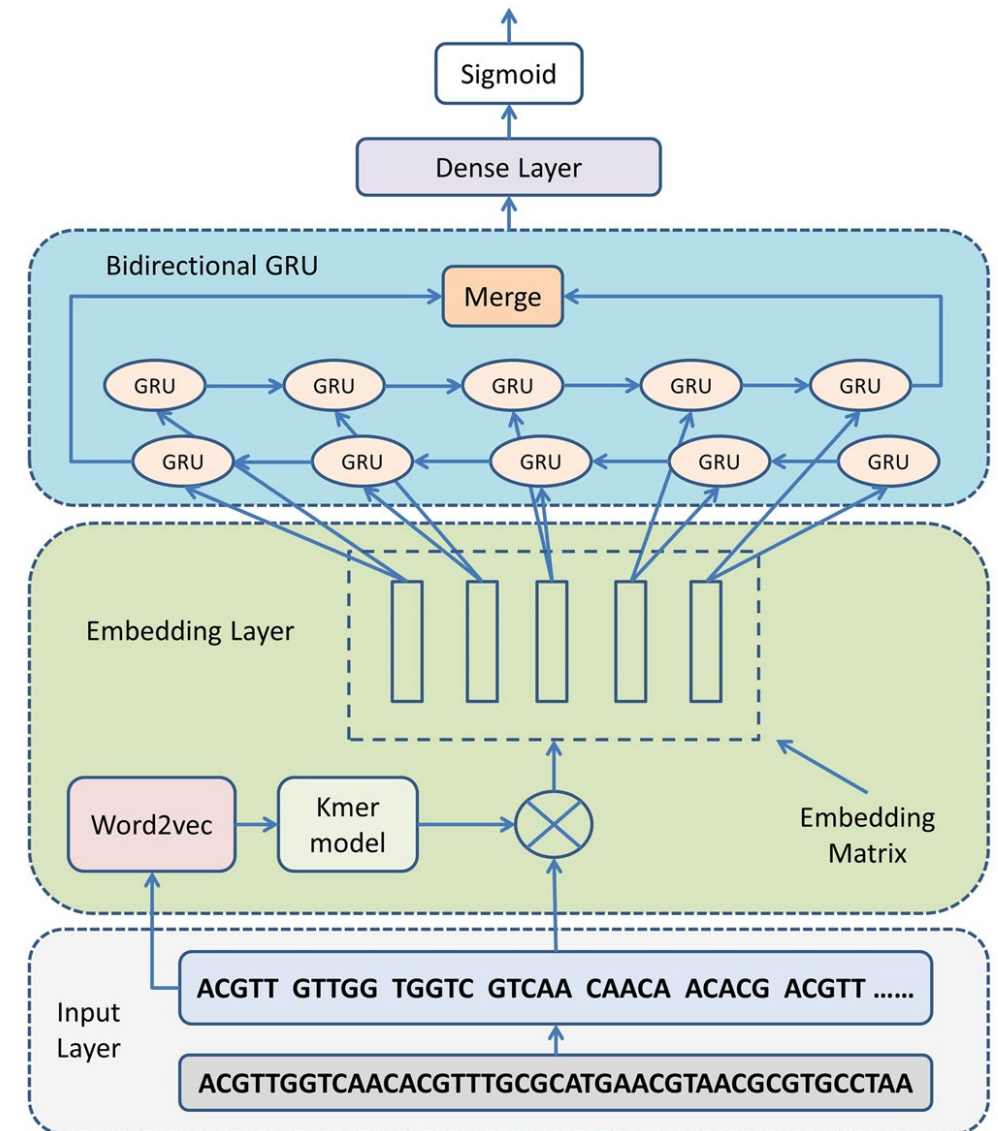


Fig 8. Displays the actual data and the predicted data from the four models for each stock index in Year 1 from 2010.10.01 to 2011.09.30.

<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.

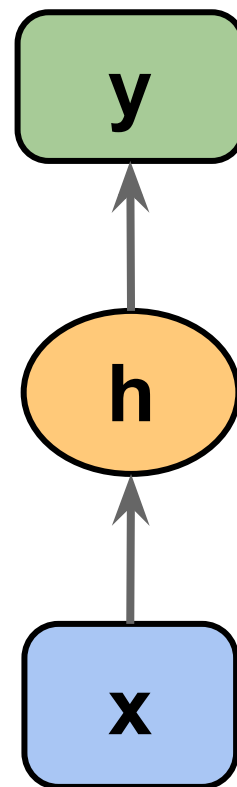


Shen, Zhen, Wenzheng Bao, and De-Shuang Huang. "Recurrent Neural Network for Predicting Transcription Factor Binding Sites." *Scientific reports* 8, no. 1 (2018): 15270.

DNA or (amino acid/protein)
sequence modeling

Overview

Networks we used previously: also called feedforward neural networks



Recurrent Neural Network (RNN)

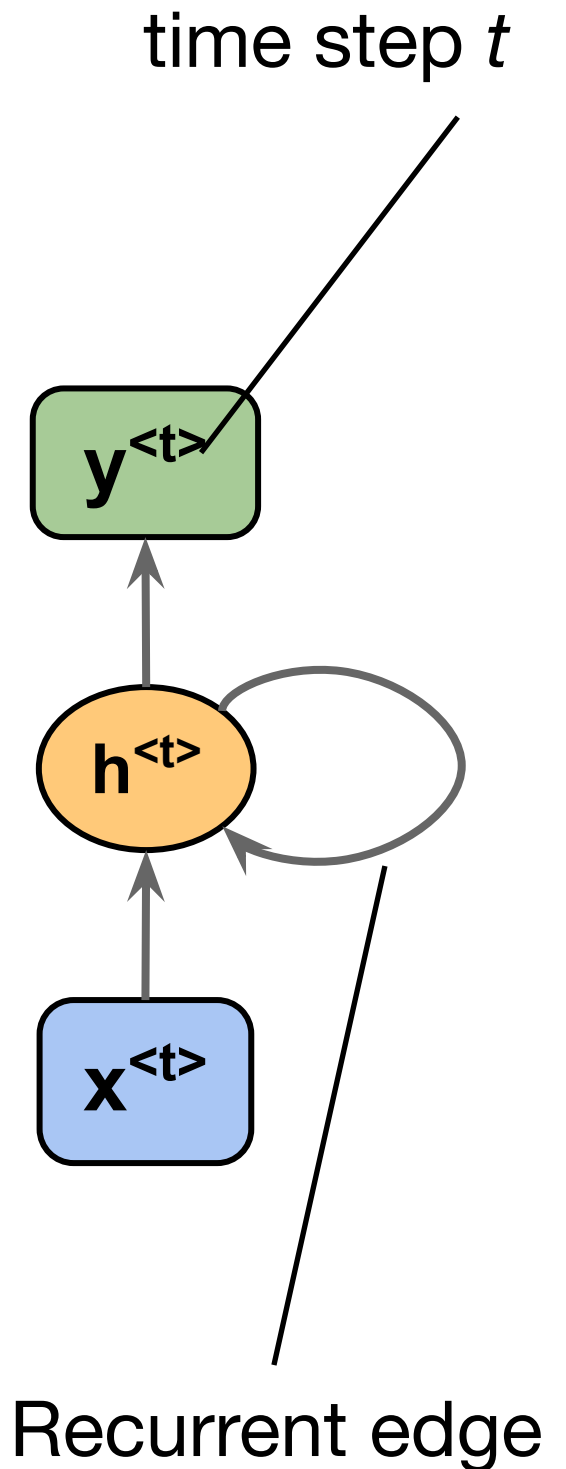


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

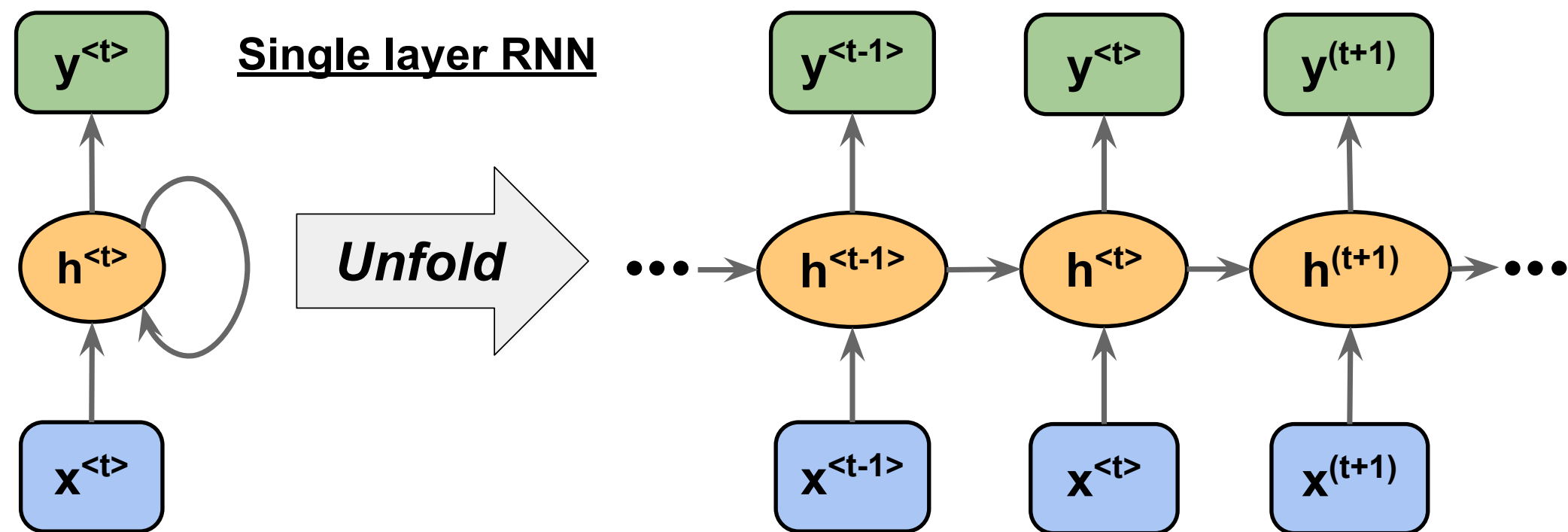


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Packt, 2019

Overview

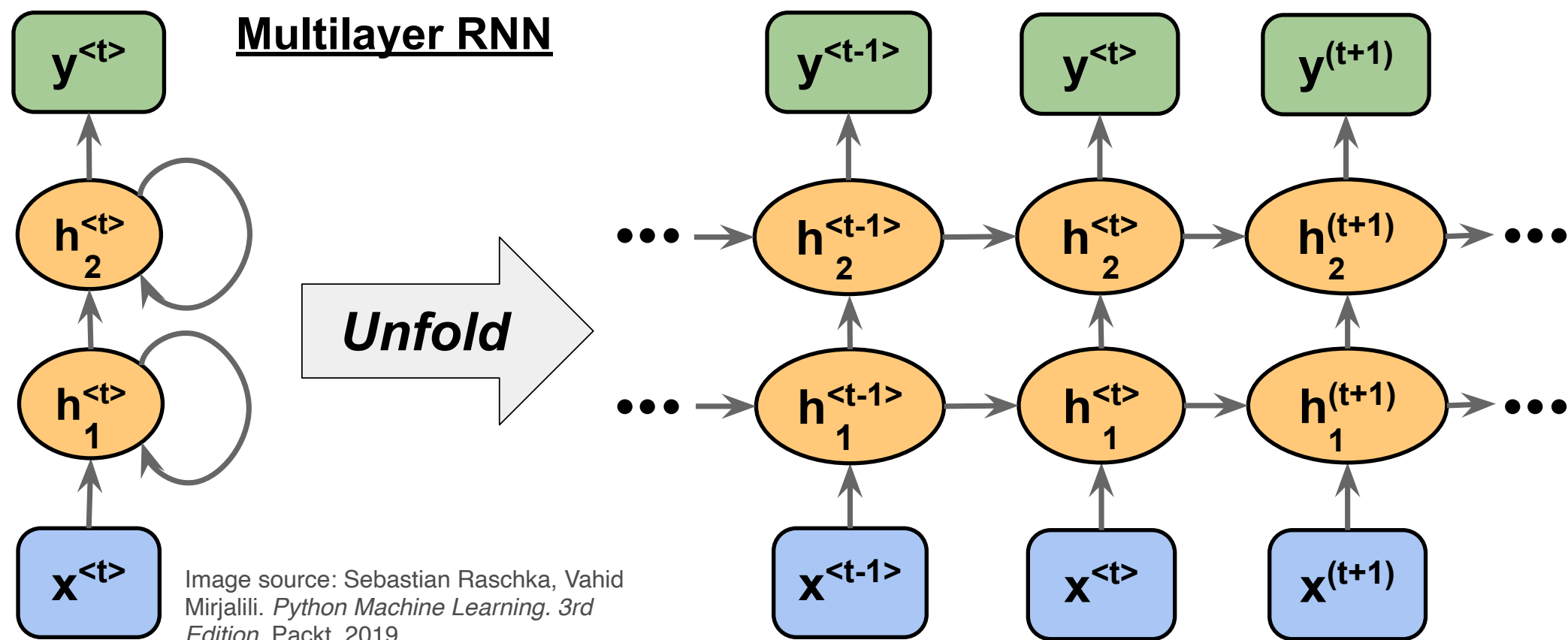
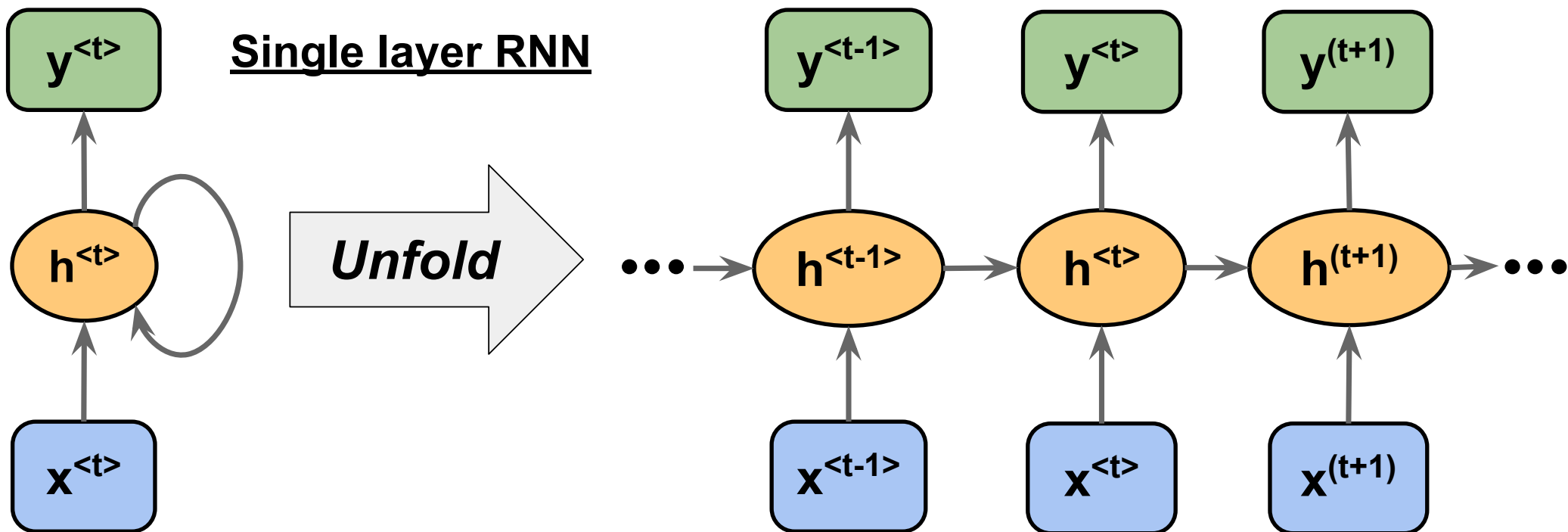


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

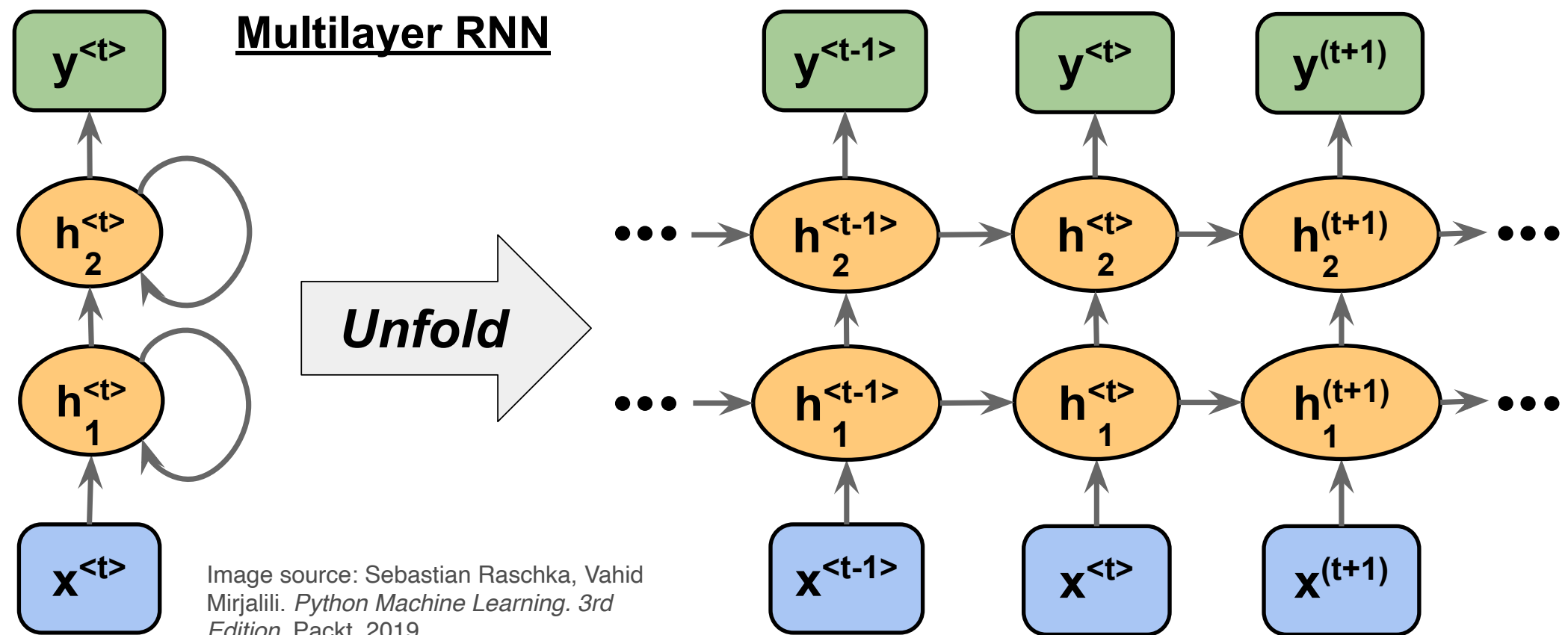
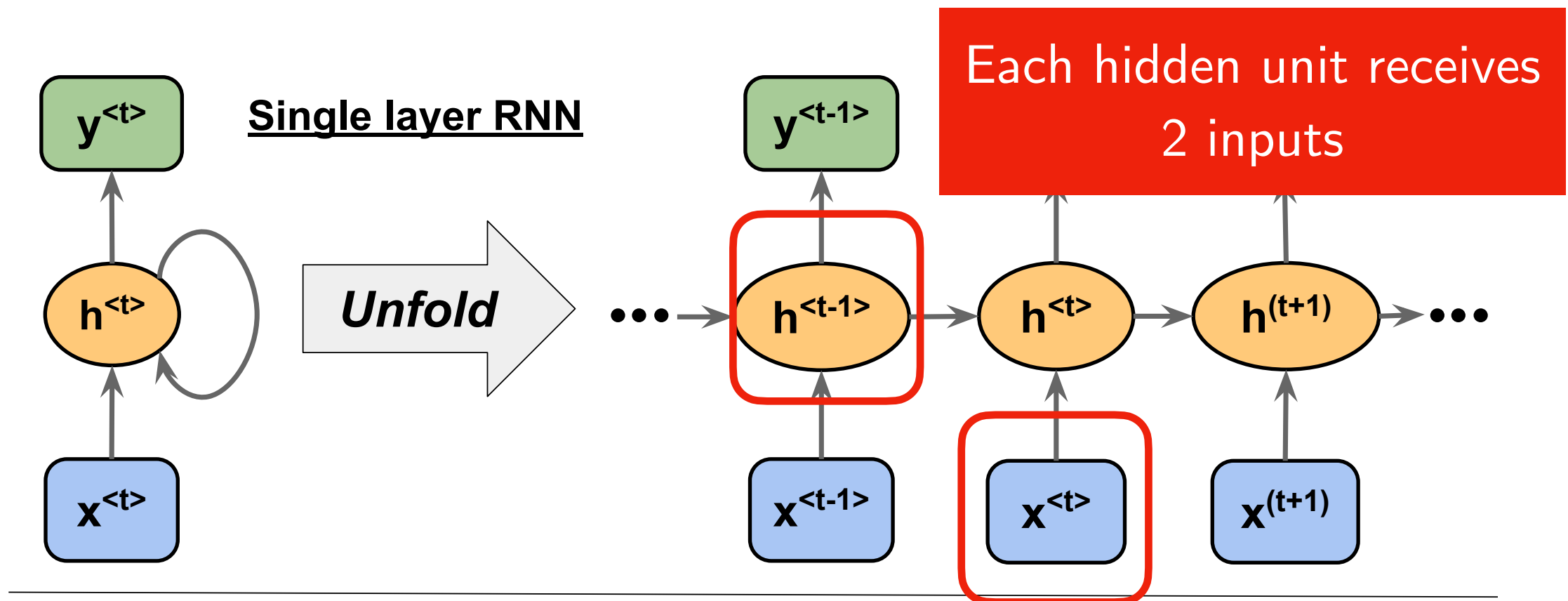


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Packt, 2019

The simple things are never simple are they?

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
- 3. Different Types of Sequence Modeling Tasks**
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

Different Types of Sequence Modeling Tasks

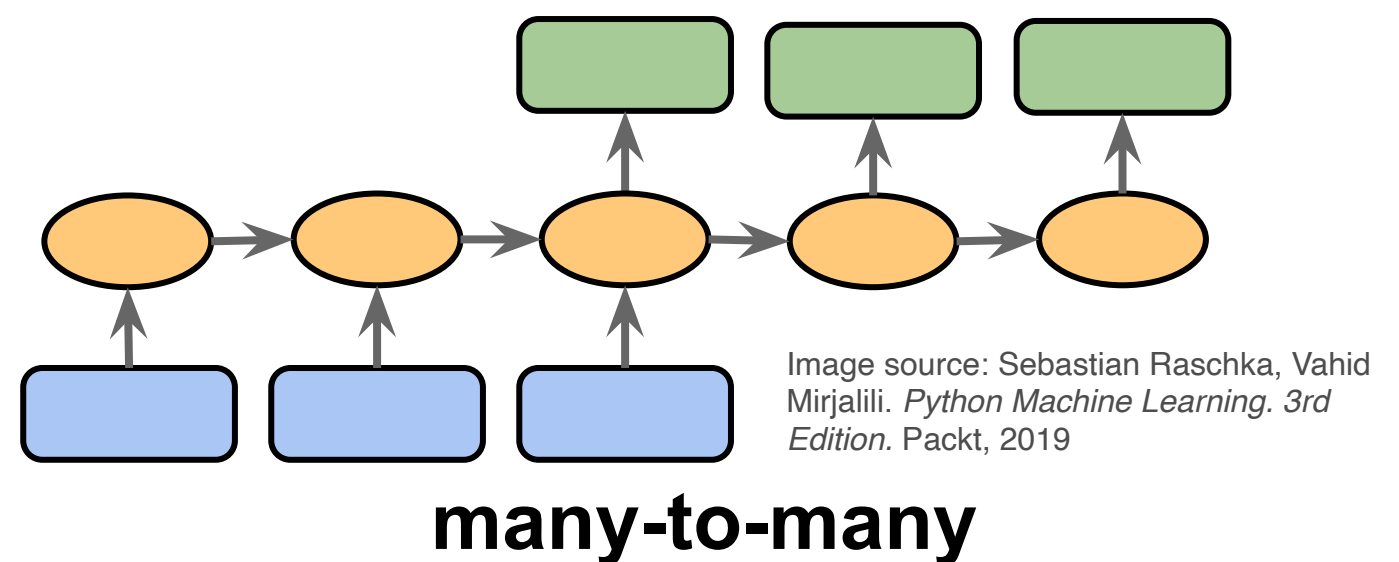
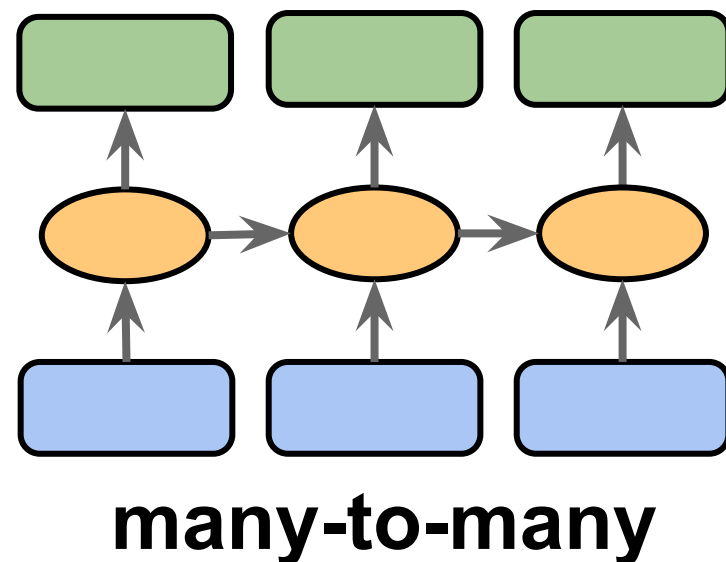
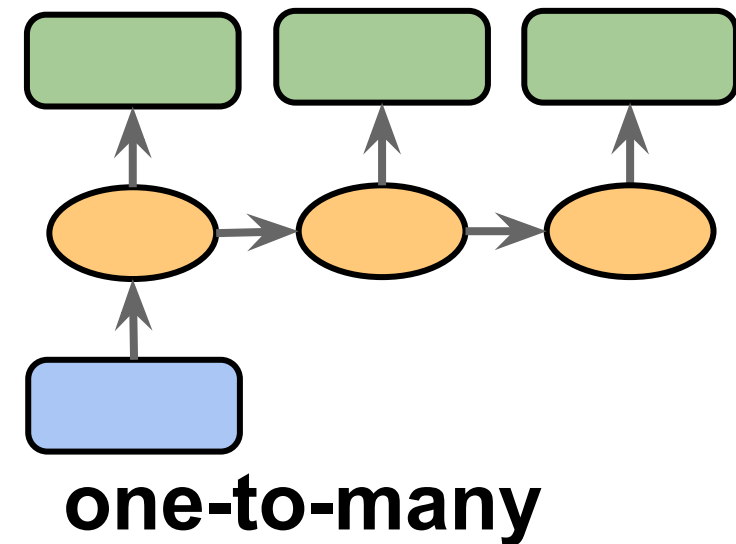
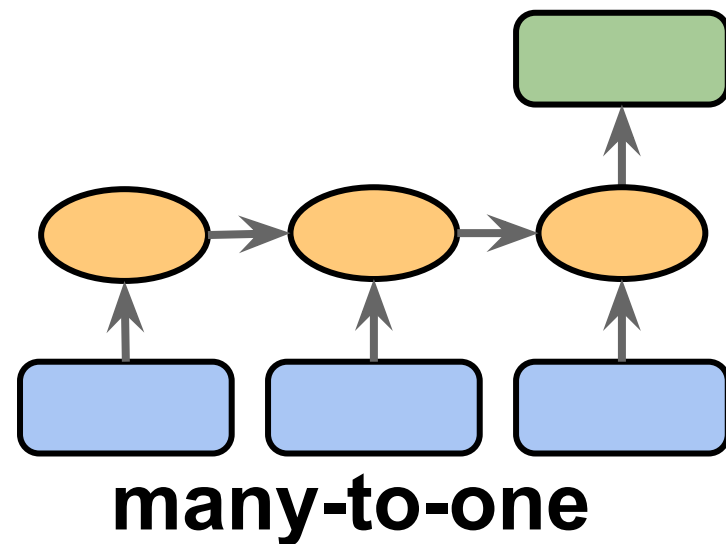
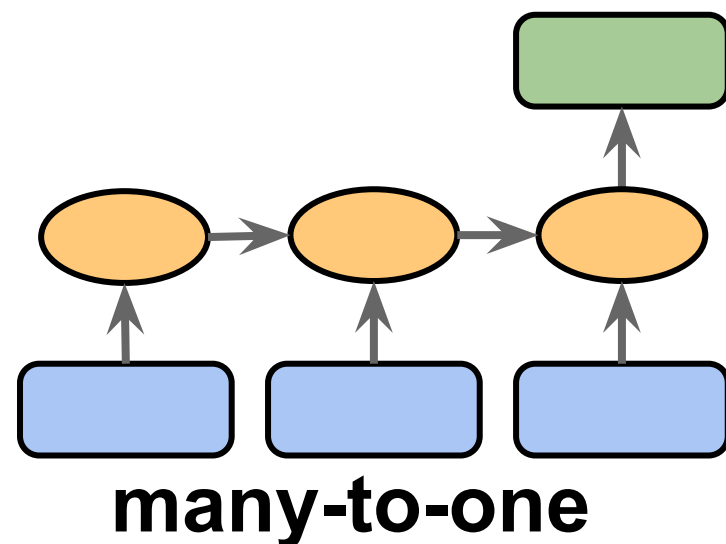


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Packt, 2019

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

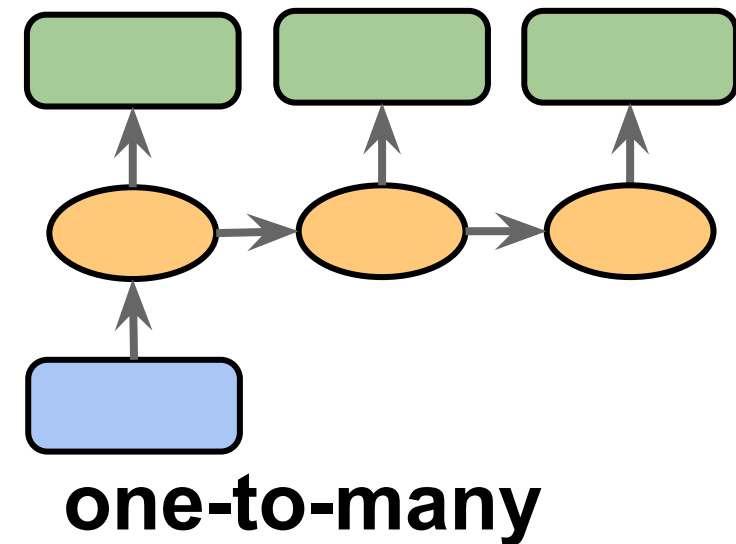
Different Types of Sequence Modeling Tasks



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.

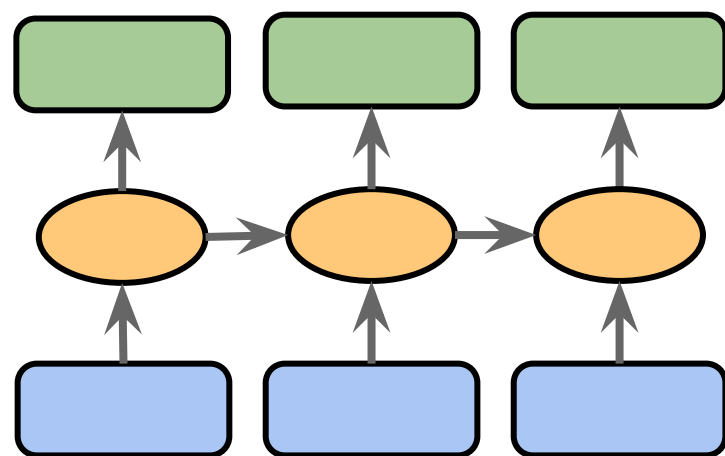
Ex.: Image captioning, where the input is an image, the output is a text description of that image

Different Types of Sequence Modeling Tasks

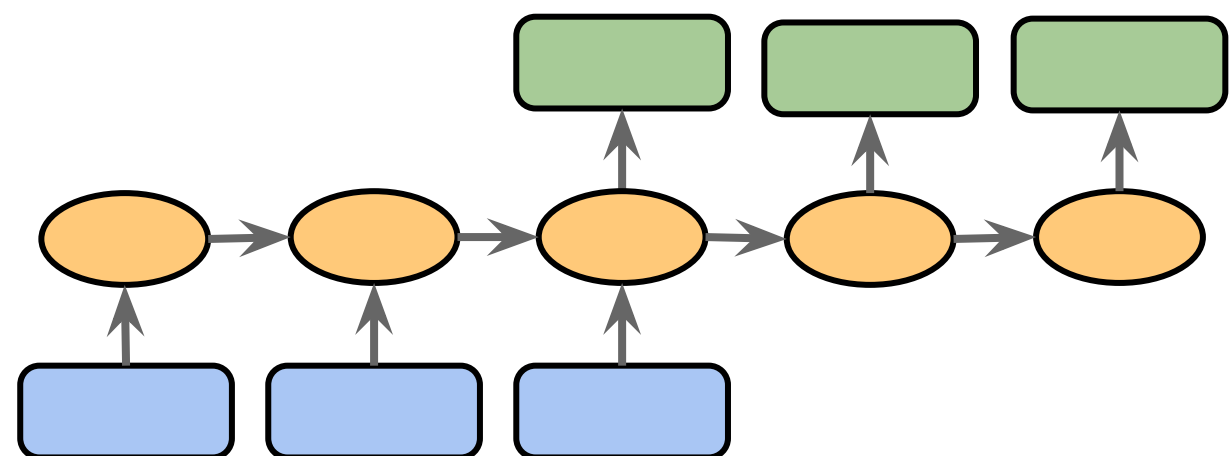
Many-to-many: Both inputs and outputs are sequences. Can be direct or delayed.

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

How RNNs Look Like Under the Hood

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
- 4. Backpropagation Through Time**
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

Weight matrices in a single-hidden layer RNN

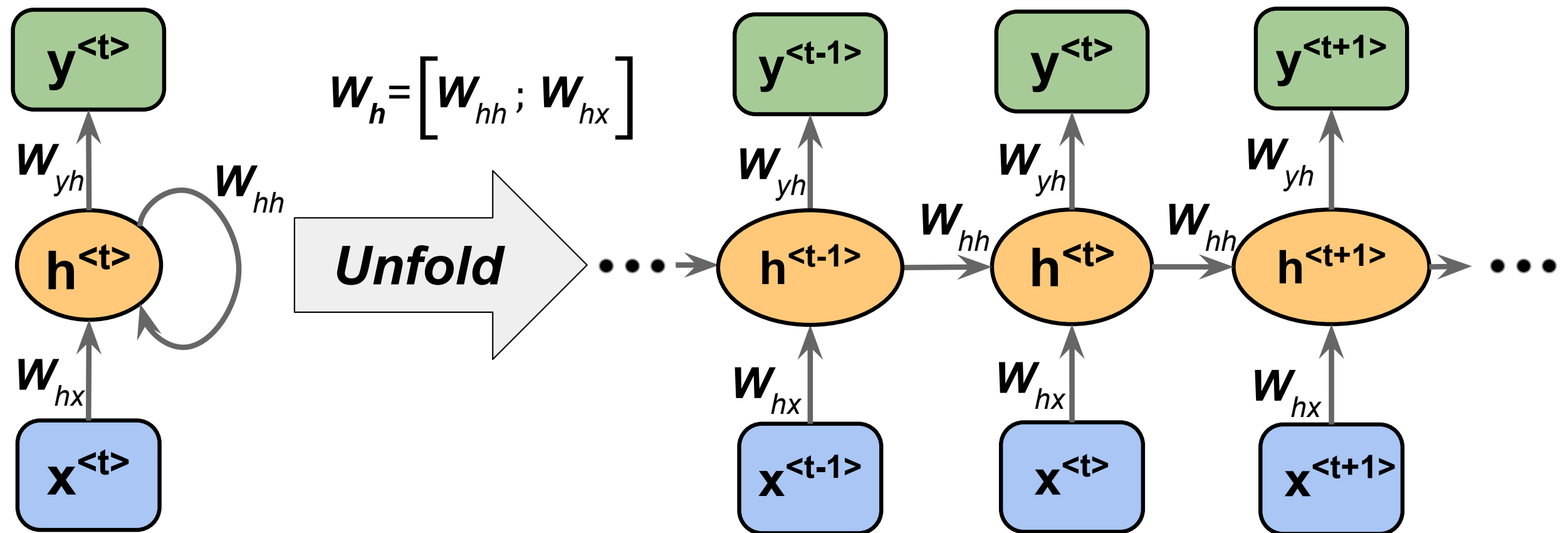


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

Weight matrices in a single-hidden layer RNN

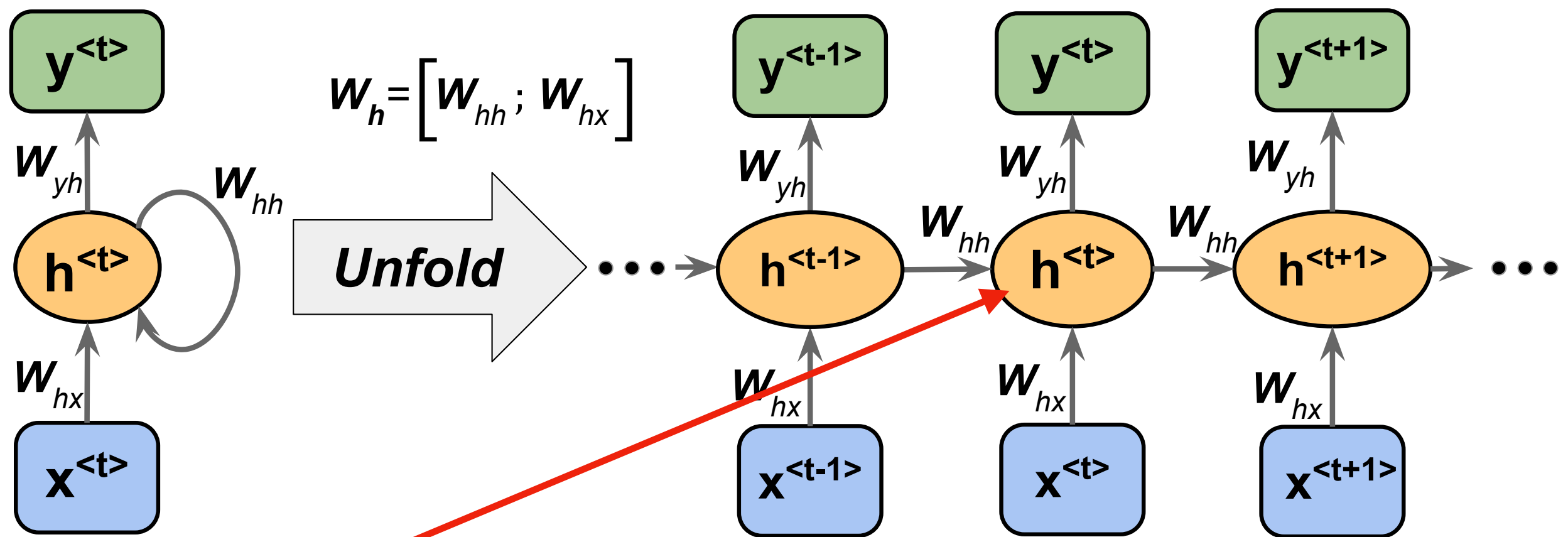


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

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

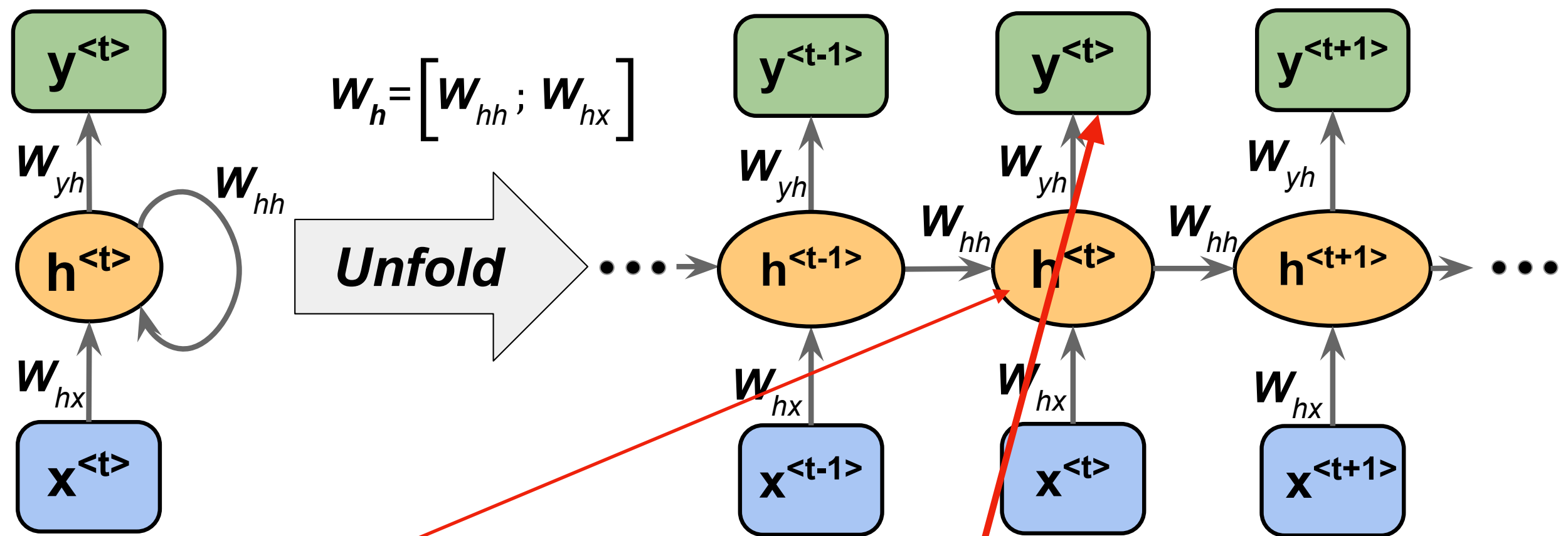


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

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:

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

Net input:

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

Output:

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

Backpropagation through time

The overall loss can be computed as the sum over all time steps

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

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

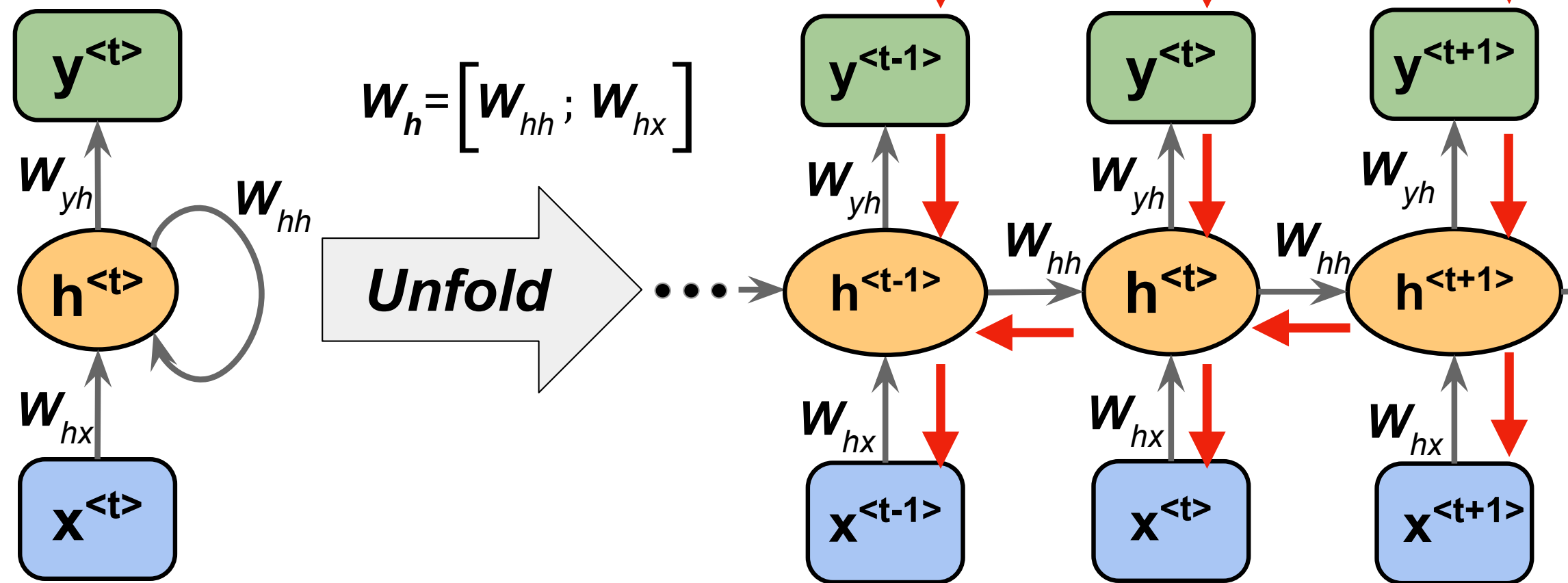
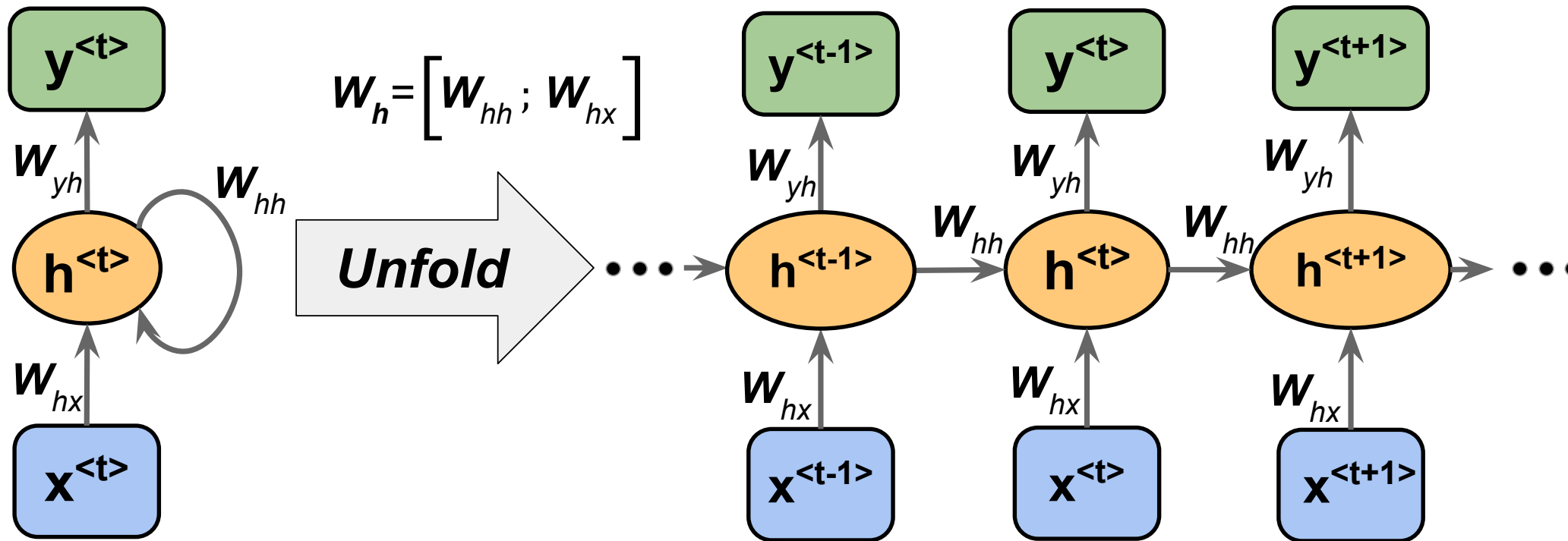


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

Backpropagation through time

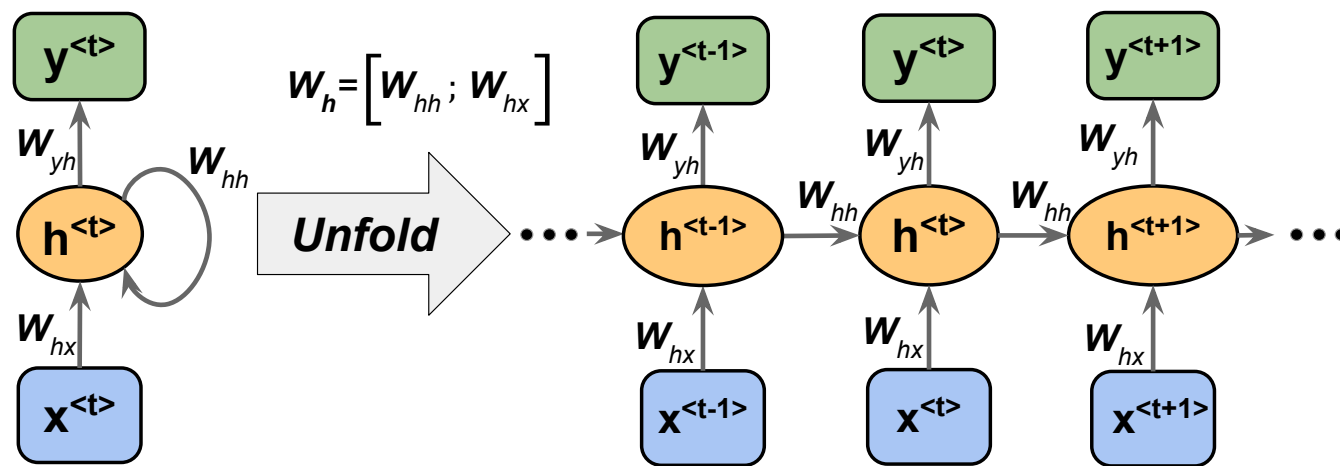


Werbos, Paul J. "[Backpropagation through time: what it does and how to do it.](#)" *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

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

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

Backpropagation through time



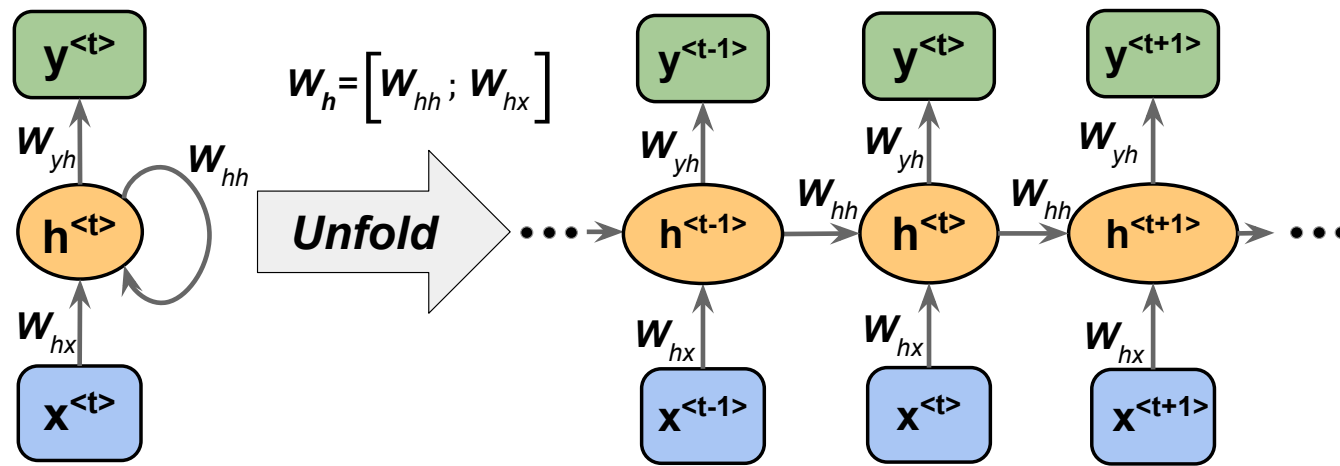
Werbos, Paul J. "[Backpropagation through time: what it does and how to do it.](#)" *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

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

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!

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

Modeling Long Range Dependencies

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
- 5. Long-Short Term Memory (LSTM)**
6. Many-to-one Word RNNs
7. RNN Classifiers in PyTorch

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 after 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:

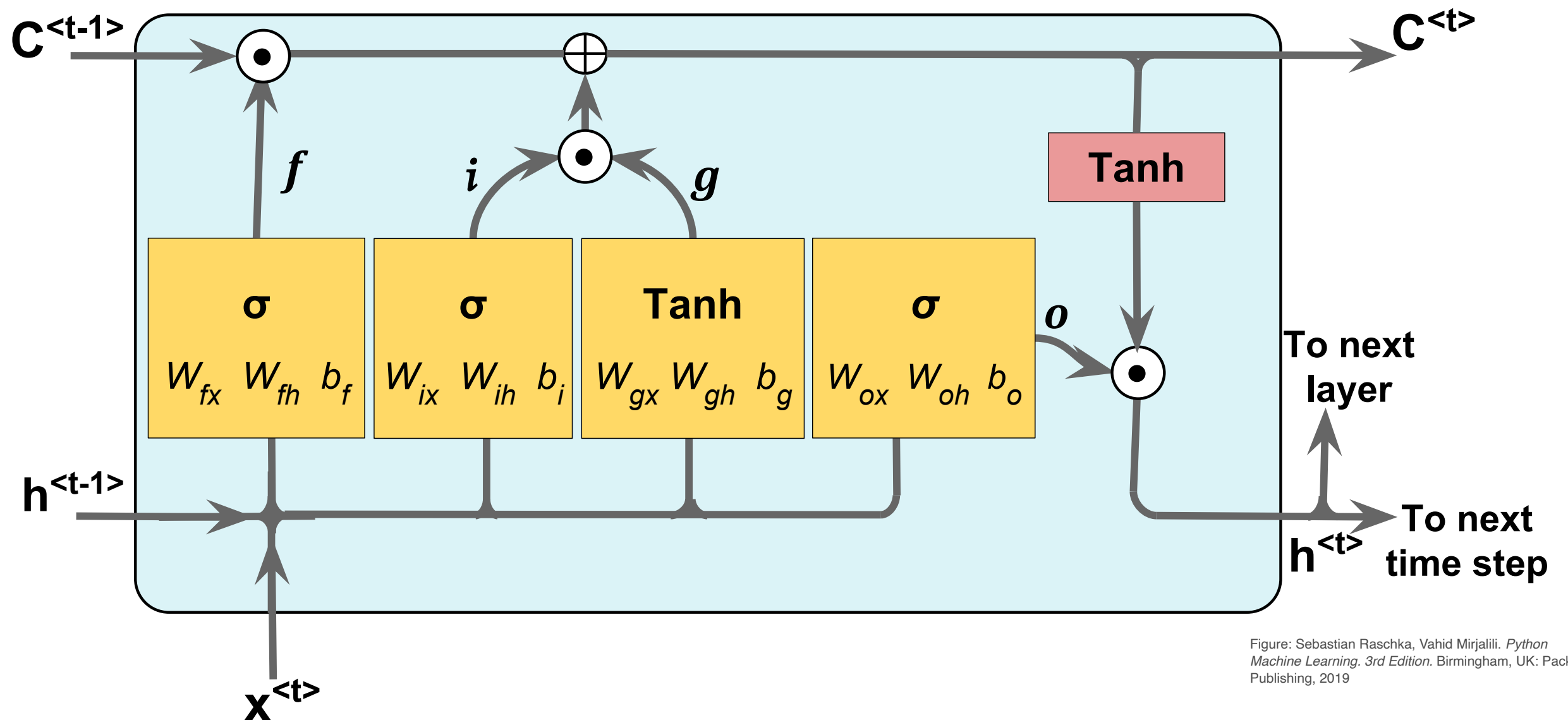
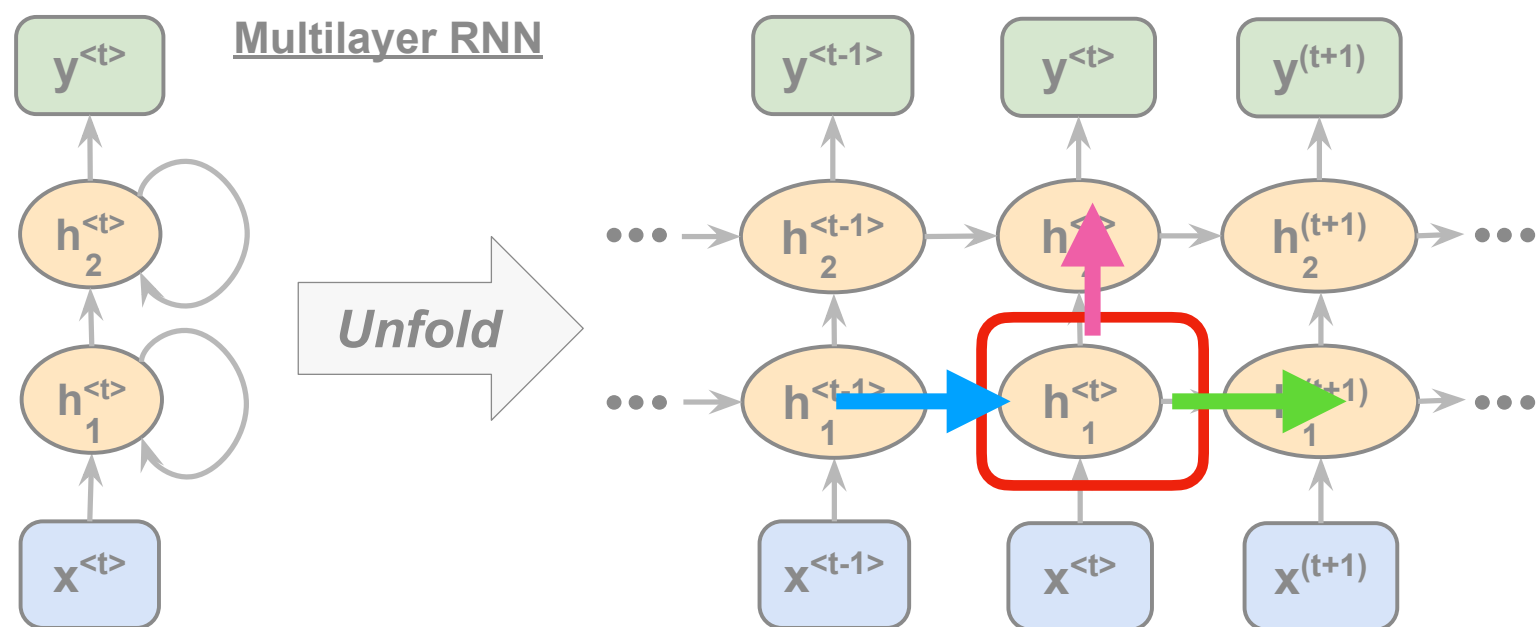
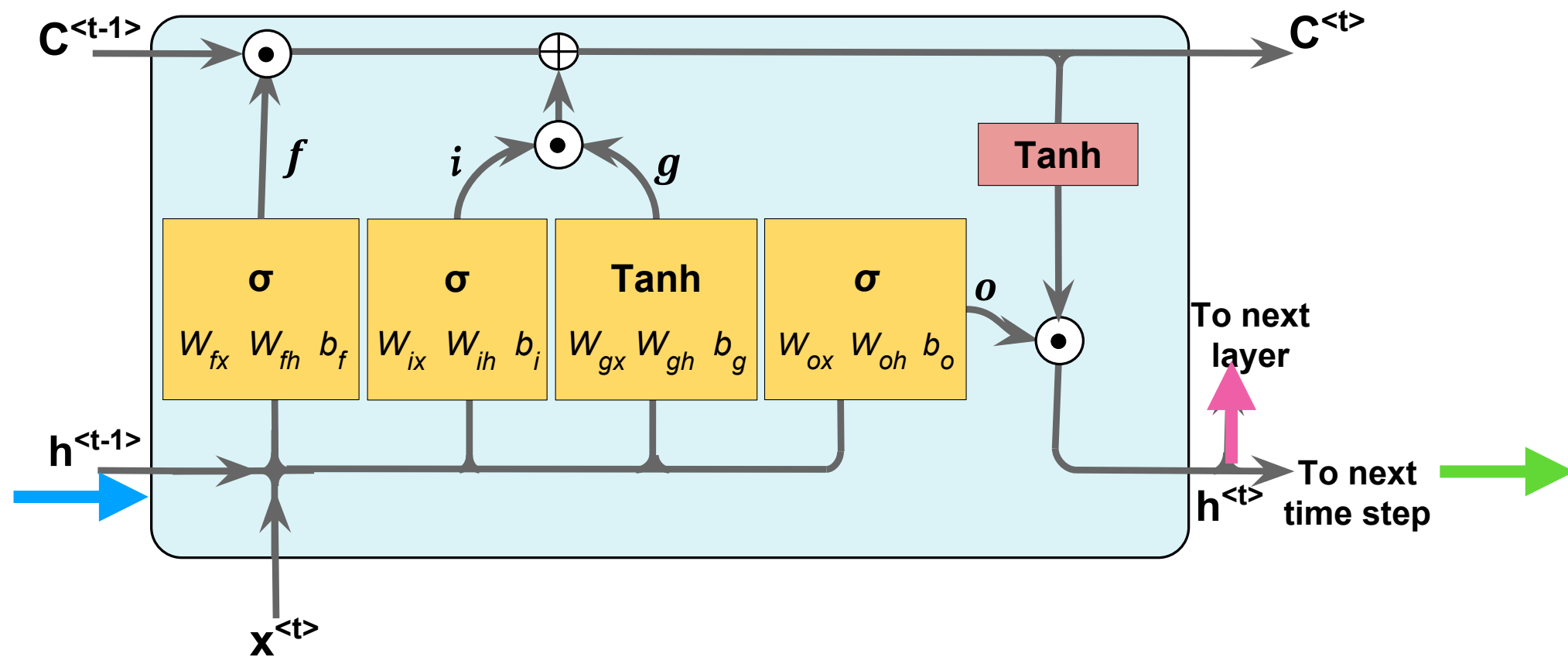


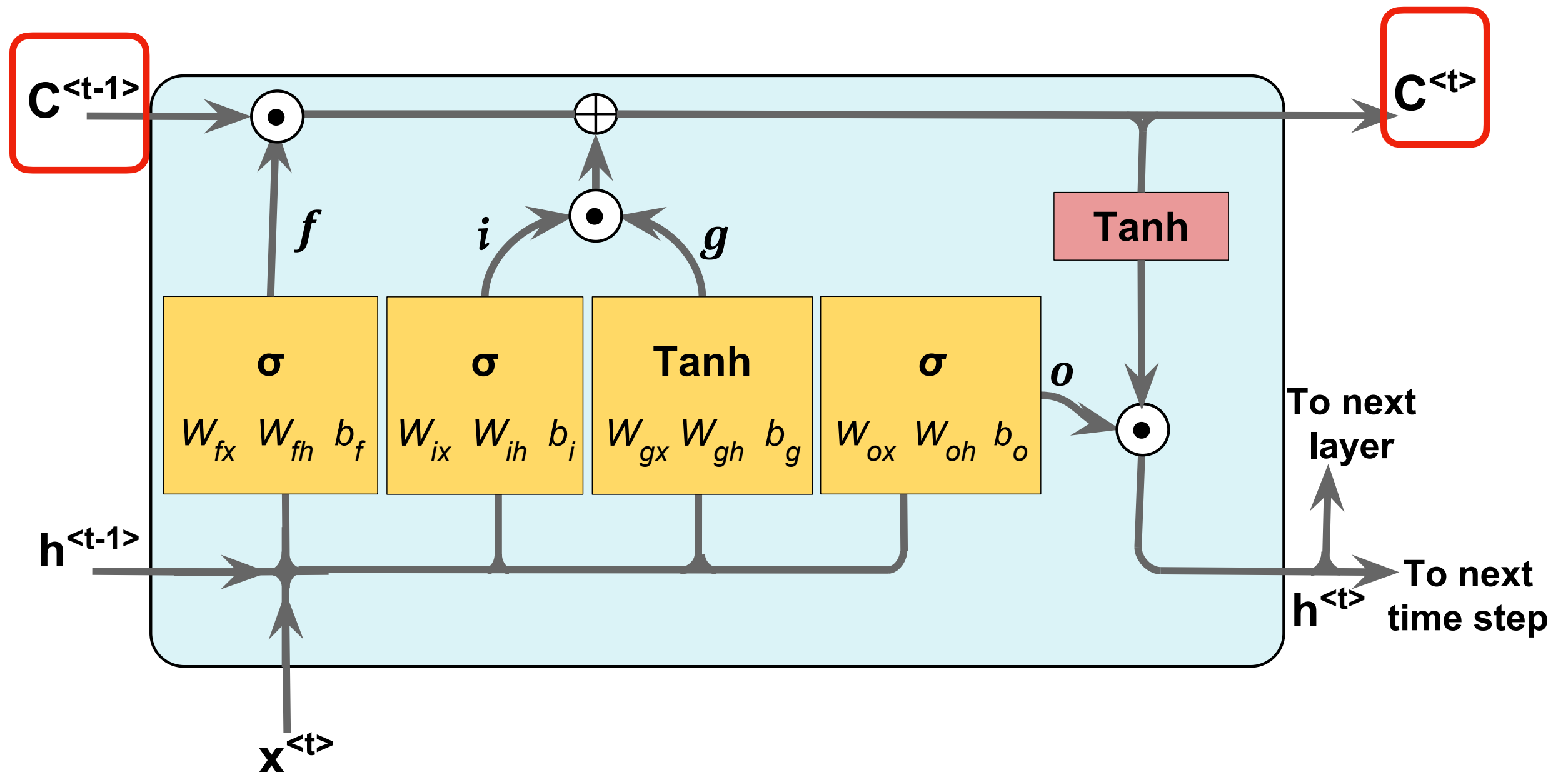
Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning, 3rd Edition*. Birmingham, UK: Packt Publishing, 2019



Long-short term memory (LSTM)

Cell state at previous time step

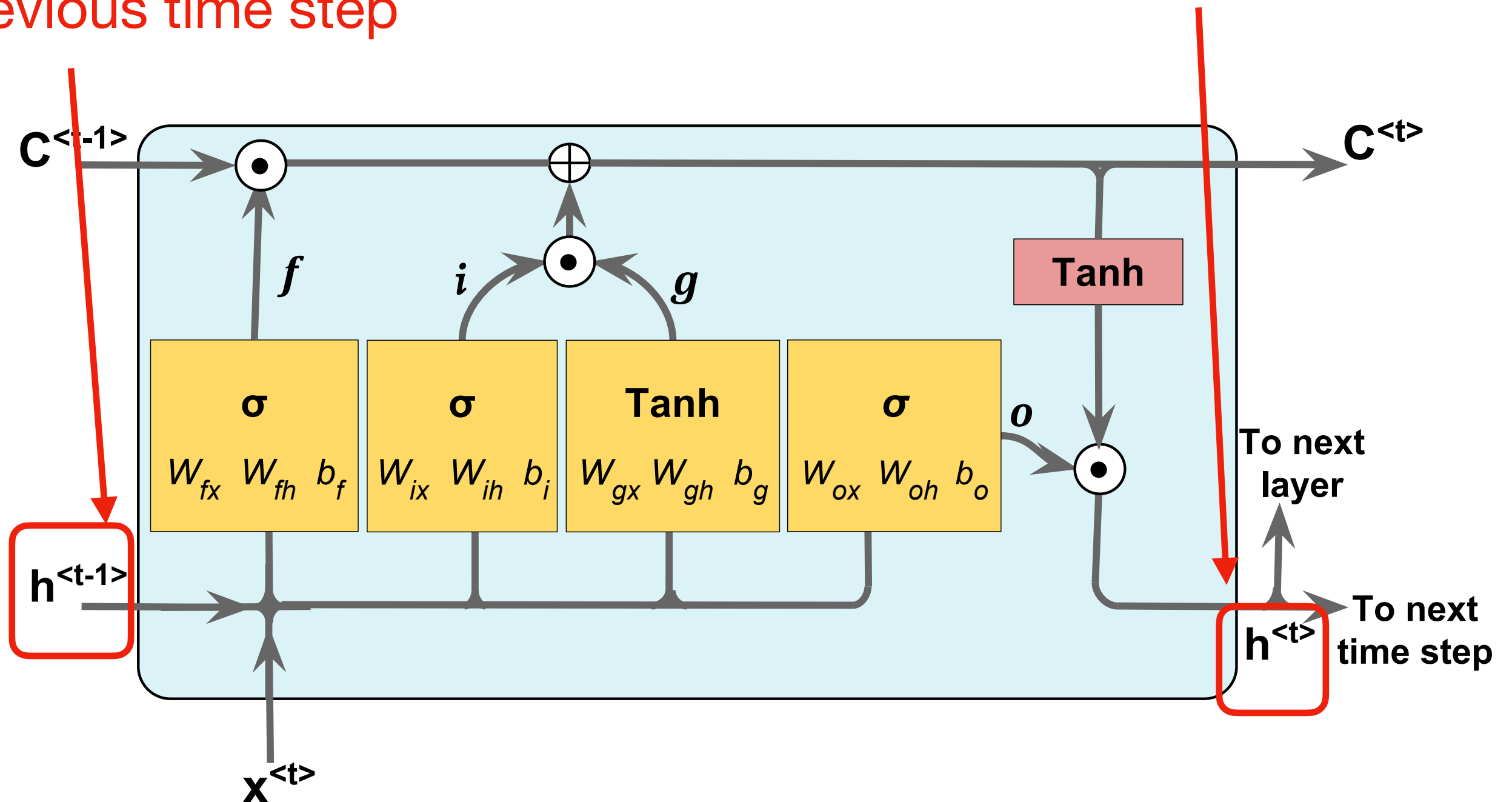
Cell state at current time step



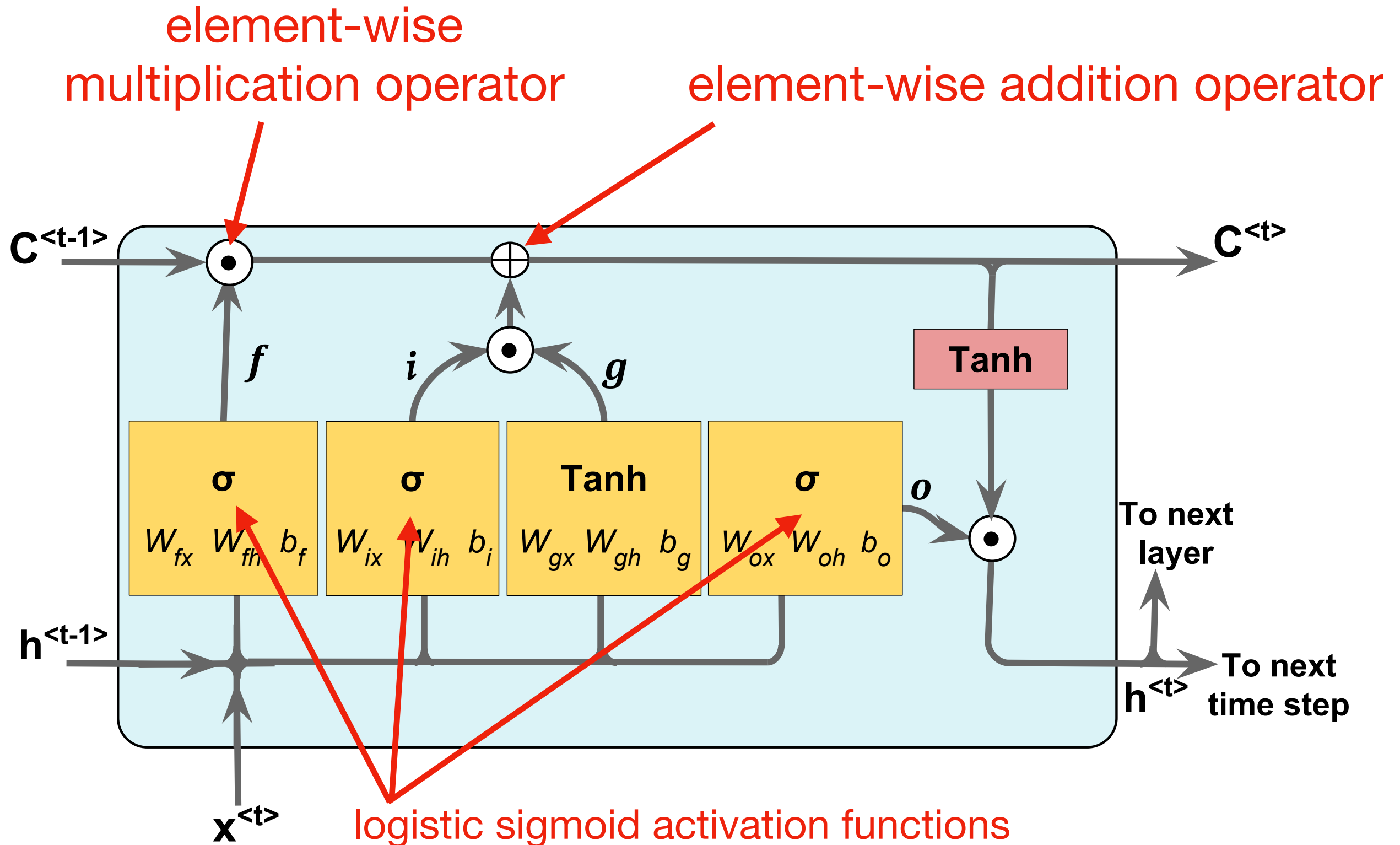
Long-short term memory (LSTM)

activation from
previous time step

activation for next time step



Long-short term memory (LSTM)

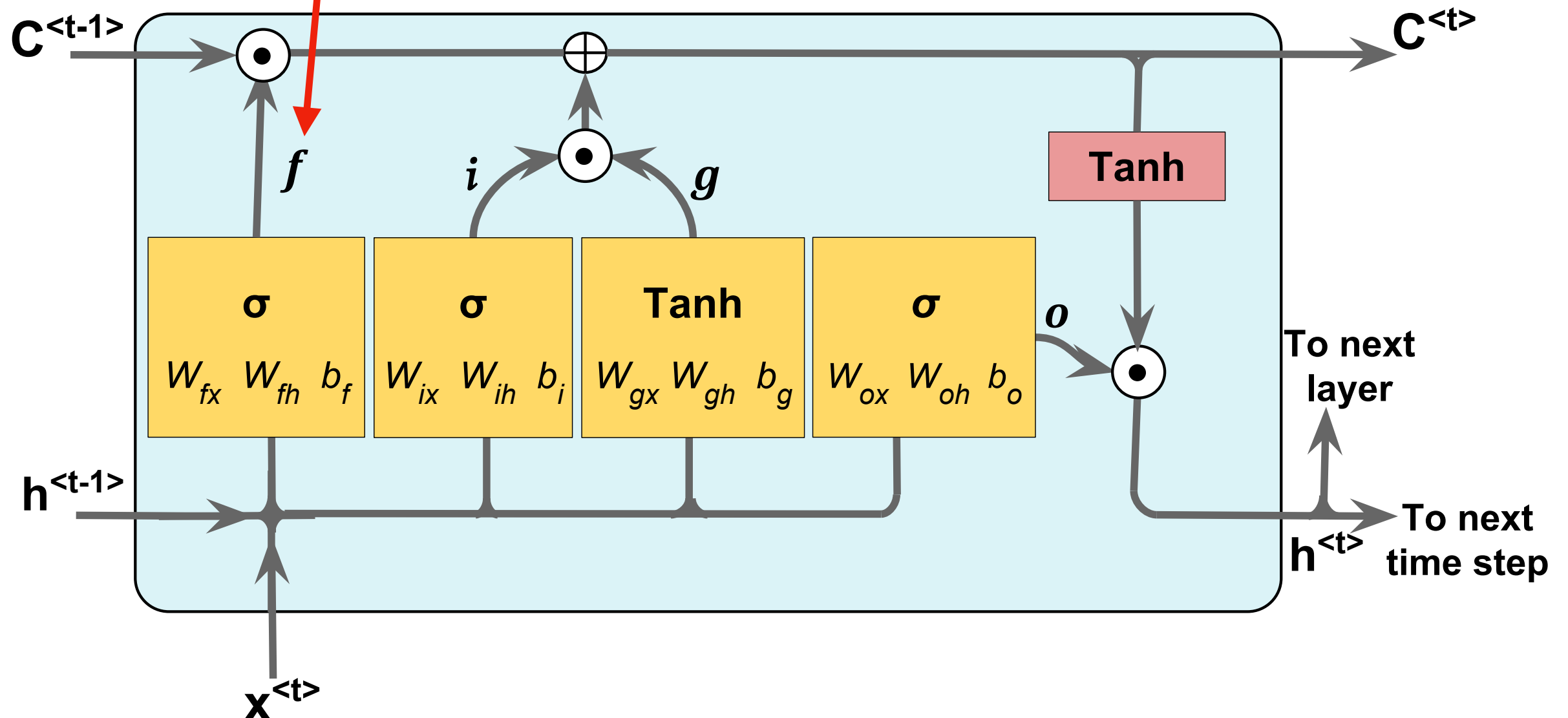


Long-short term memory (LSTM)

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

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

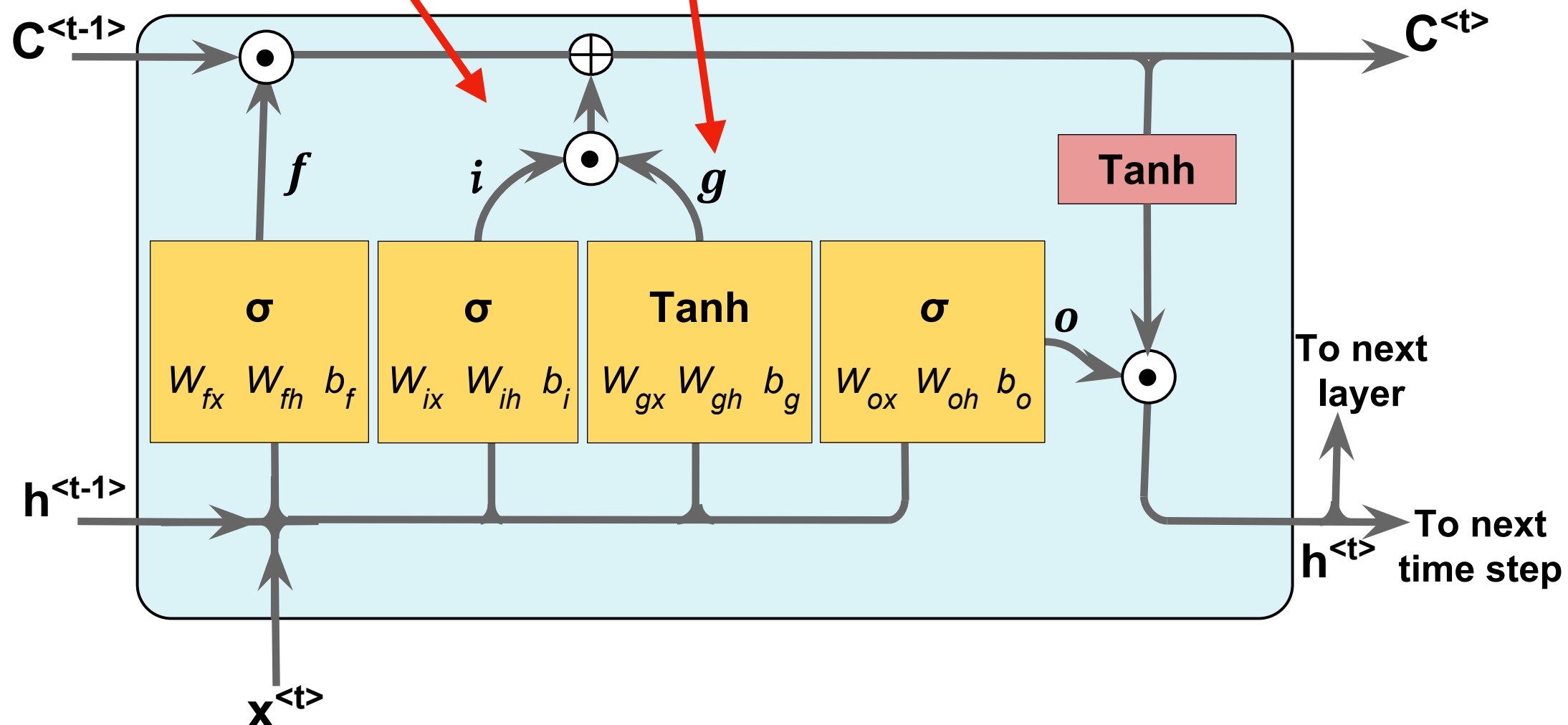


Long-short term memory (LSTM)

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

"Input Node":

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

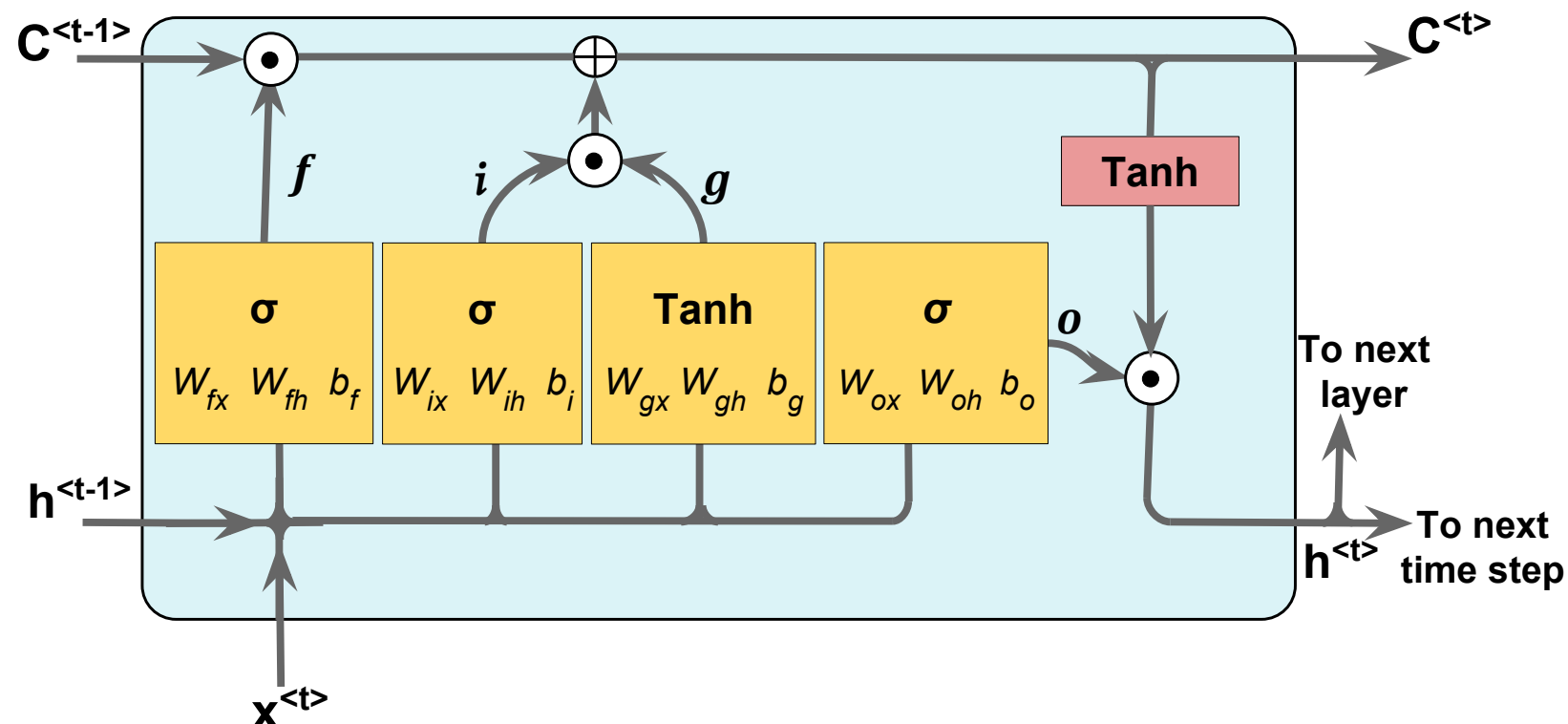


Long-short term memory (LSTM)

Brief summary of the gates so far ...

Forget Gate Input Node Input Gate

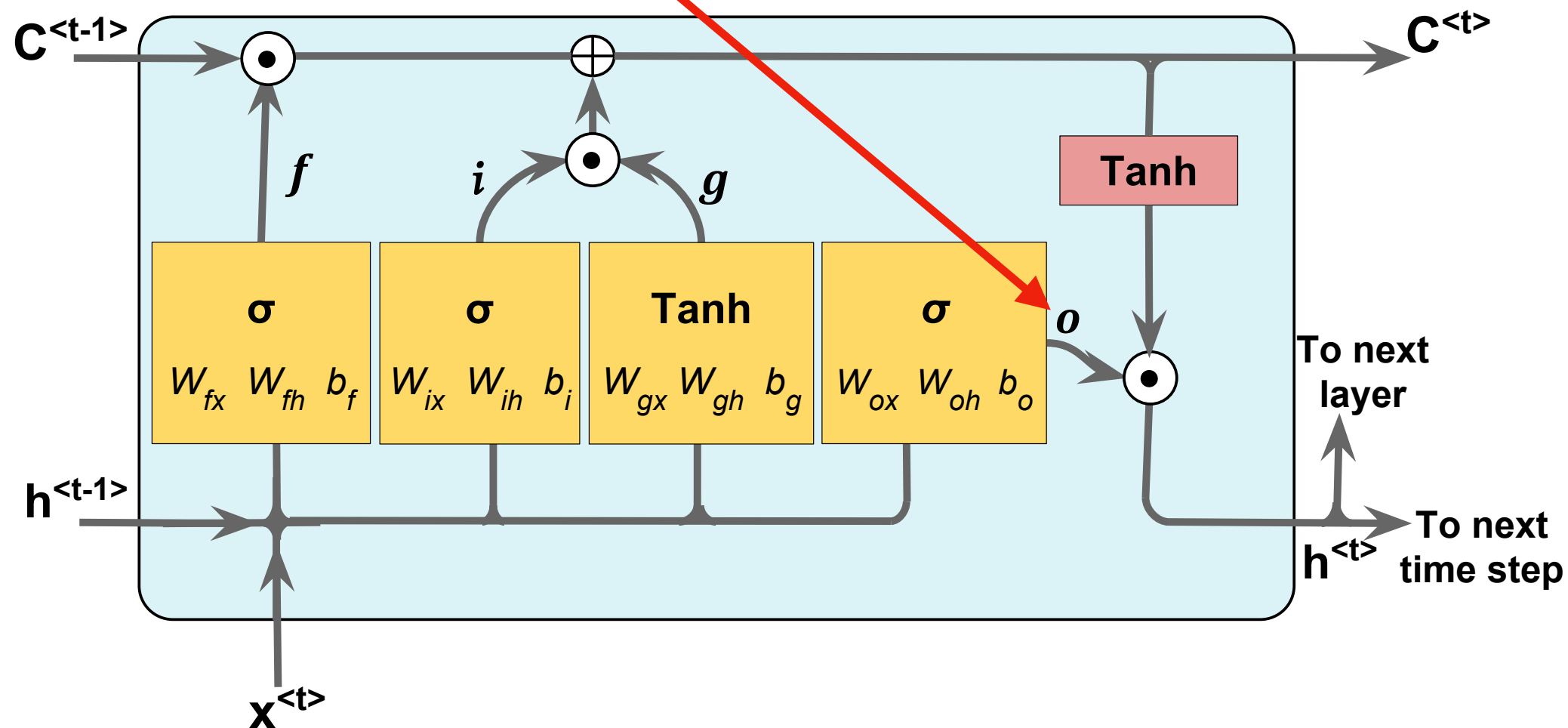
$$C^{<t>} = \left(C^{<t-1>} \odot f_t \right) \oplus \underbrace{\left(i_t \odot g_t \right)}_{\text{For updating the cell state}}$$



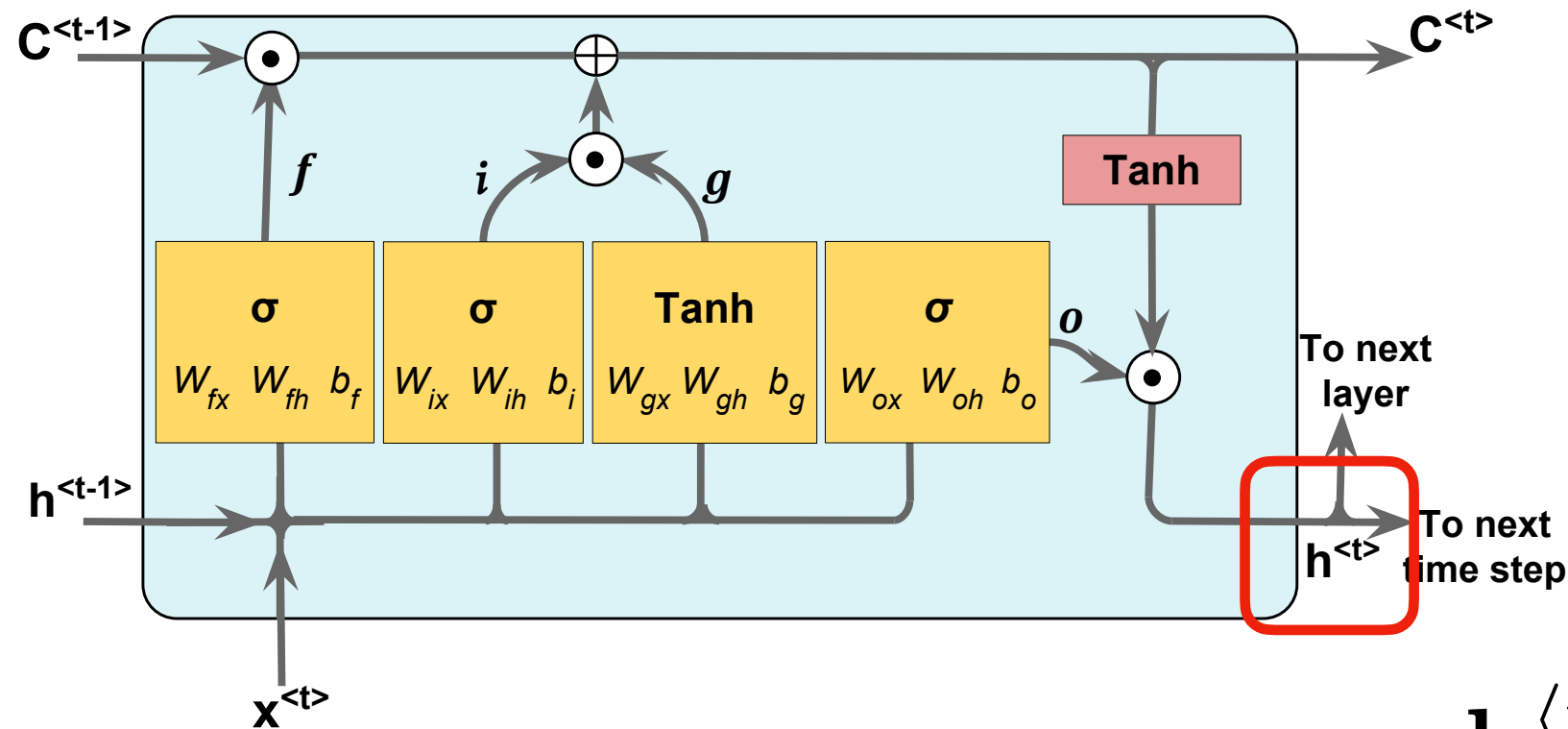
Long-short term memory (LSTM)

Output gate for updating the values of hidden units:

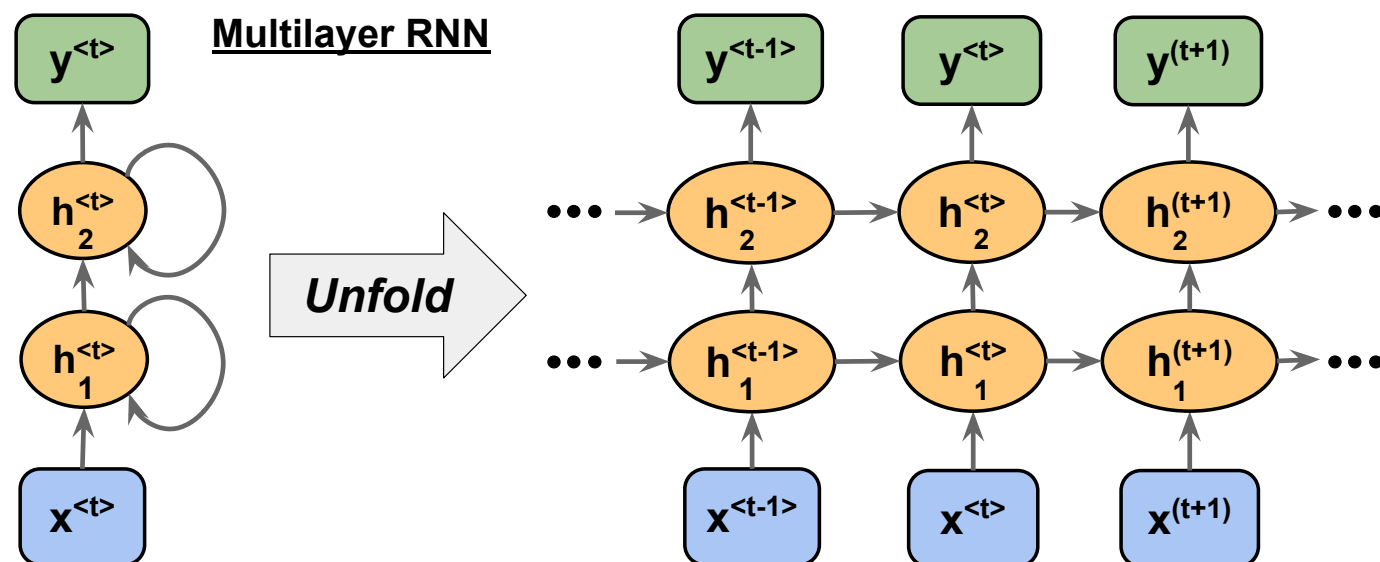
$$\mathbf{o}_t = \sigma \left(\mathbf{W}_{ox} \mathbf{x}^{(t)} + \mathbf{W}_{oh} \mathbf{h}^{(t-1)} + \mathbf{b}_o \right)$$



Long-short term memory (LSTM)

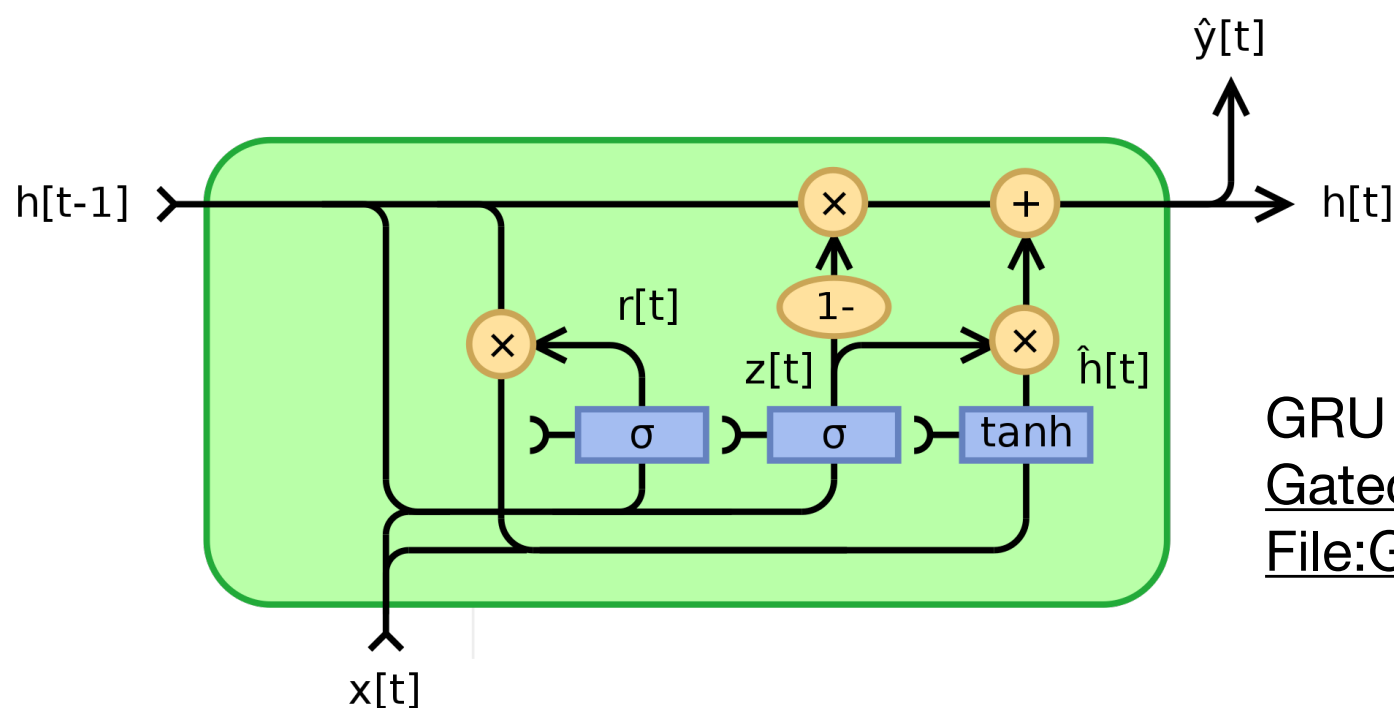


$$\mathbf{h}^{<t>} = \mathbf{o}_t \odot \tanh(\mathbf{C}^{<t>})$$



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](https://en.wikipedia.org/wiki/Gated_recurrent_unit#/media/File:Gated_Recurrent_Unit_base_type.svg)

Implementing an RNN / LSTM Classifier

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
- 6. Many-to-one Word RNNs**
7. RNN Classifiers in PyTorch

Different Types of Sequence Modeling Tasks

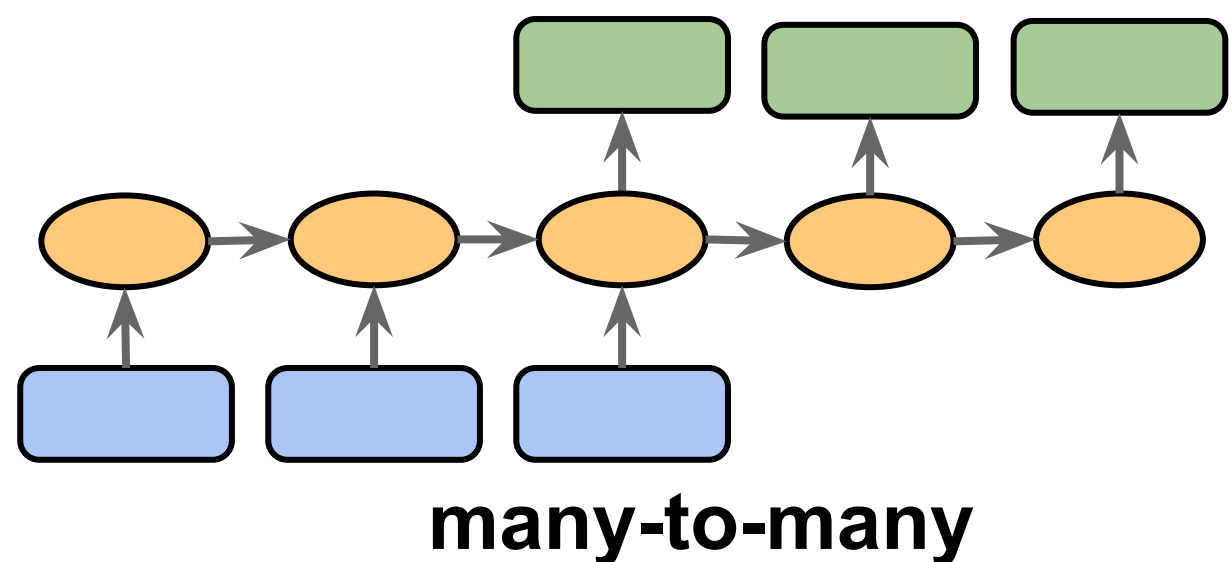
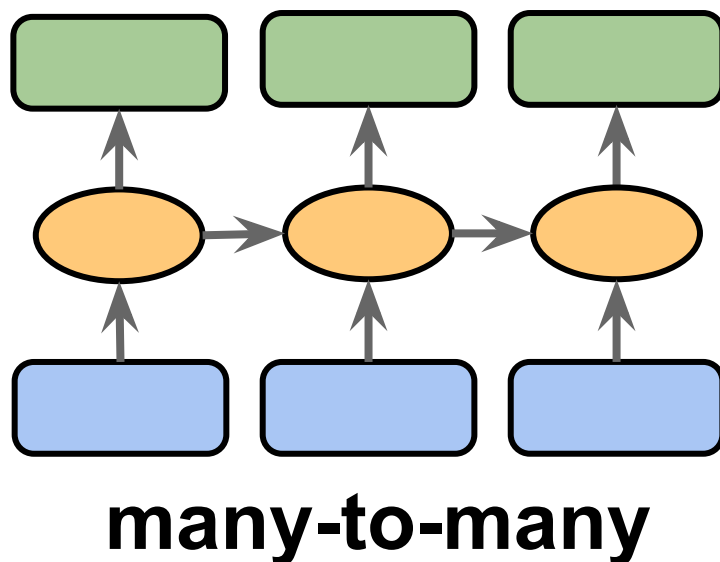
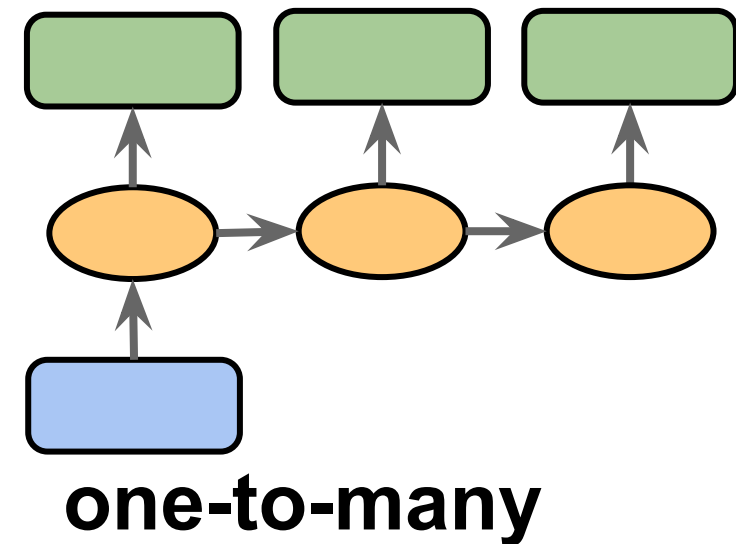
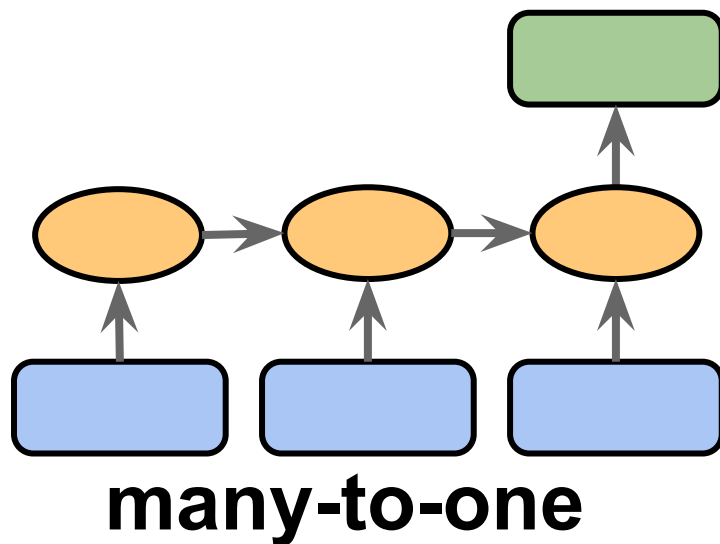


Figure based on:
The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy (<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>)

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"

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

class labels

```
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, ...)

Raschka & Mirjalili. *Python Machine Learning* 3rd Ed.
<https://github.com/rasbt/python-machine-learning-book-3rd-edition/blob/master/ch08/ch08.ipynb>

RNN Step 1): Building the Vocabulary

"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"

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

class labels



```
vocabulary = {  
    '<unk>': 0,  
    'and': 1,  
    'is': 2  
    'one': 3,  
    'shining': 4,  
    'sun': 5,  
    'sweet': 6,  
    'the': 7,  
    'two': 8,  
    'weather': 9,  
    '<pad>': 10  
}
```

RNN Step 2): Training Example Texts to Indices

"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 = {
 '<unk>': 0,
 'and': 1,
 'is': 2
 'one': 3,
 'shining': 4,
 'sun': 5,
 'sweet': 6,
 'the': 7,
 'two': 8,
 'weather': 9,
 '<pad>': 10
}

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

[7 5 2 4 ... 10 10 10]

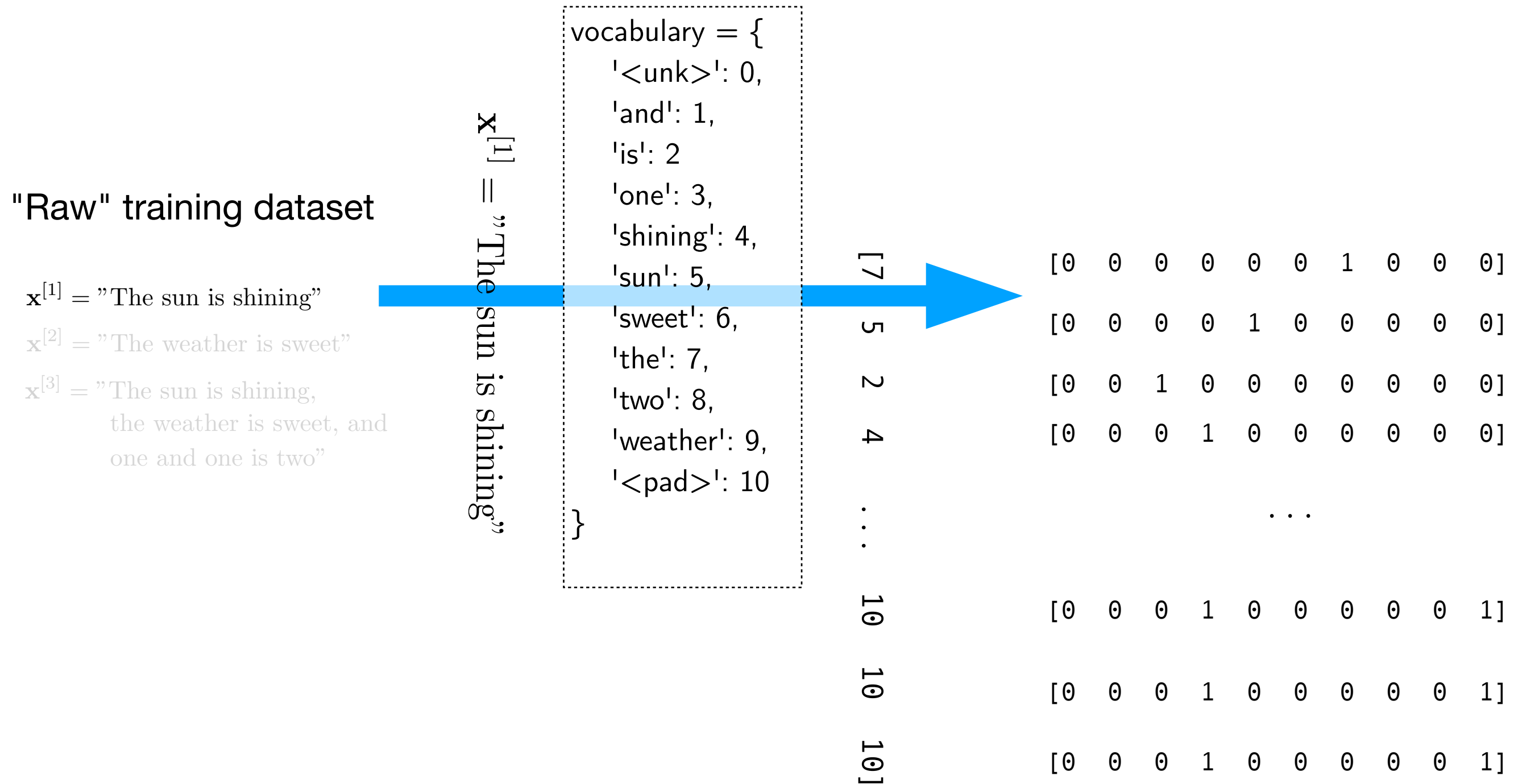
$\mathbf{x}^{[2]}$ = "The weather is sweet"

[7 9 2 6 ... 10 10 10]

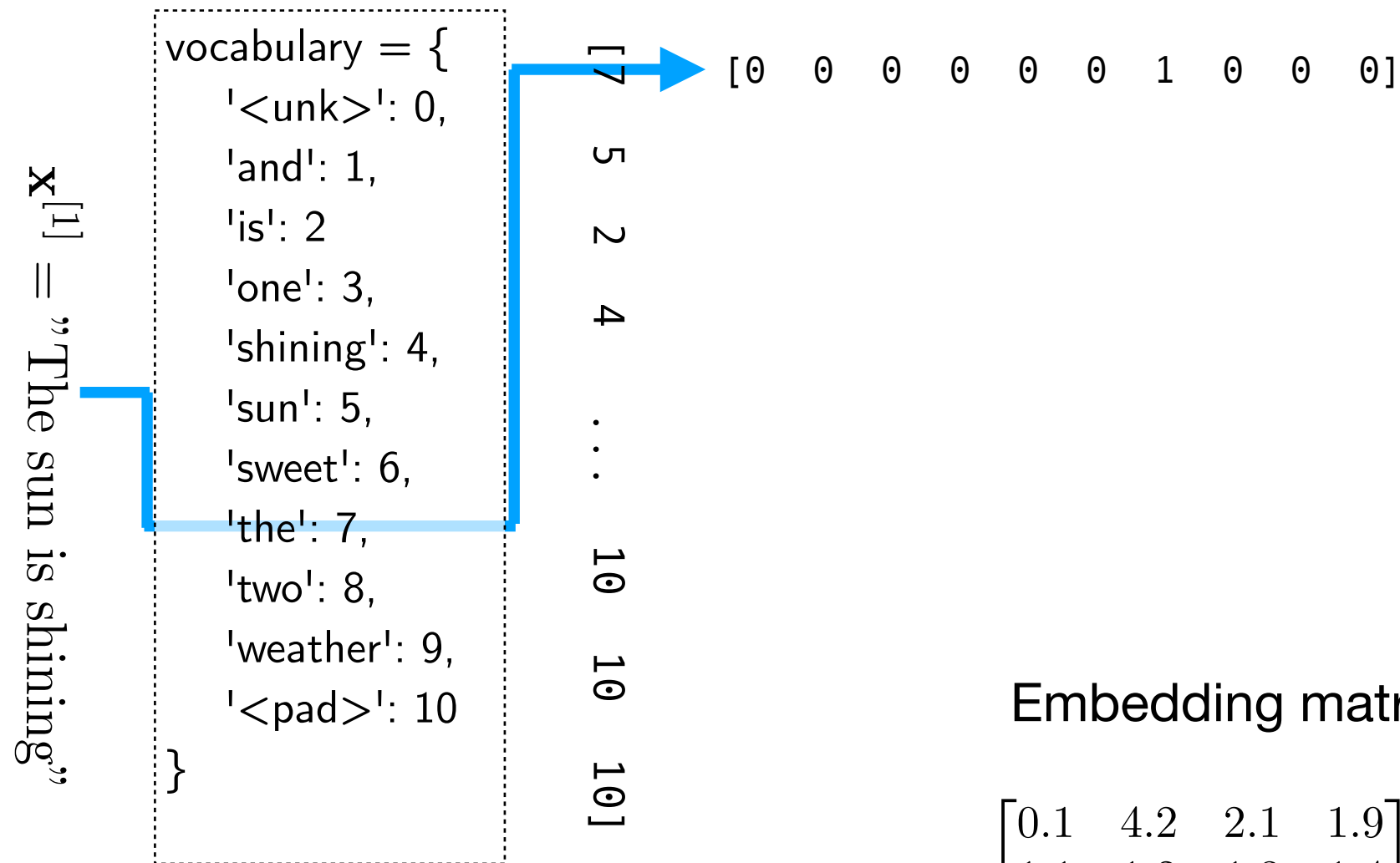
$\mathbf{x}^{[3]}$ = "The sun is shining,
the weather is sweet, and
one and one is two"

[7 5 2 4 ... 3 2 8]

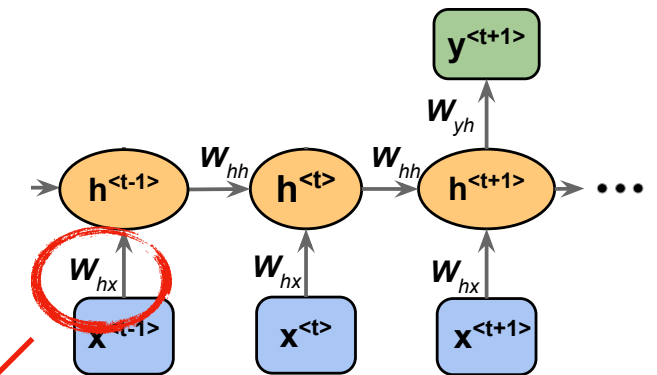
RNN Step 3): Indices to One-Hot Representation



RNN Step 4): One-Hot to Real via Embedding Matrix



Embedding is a linear layer



Embedding matrix

One-hot vector
 [0 0 0 0 0 0 1 0 0 0]

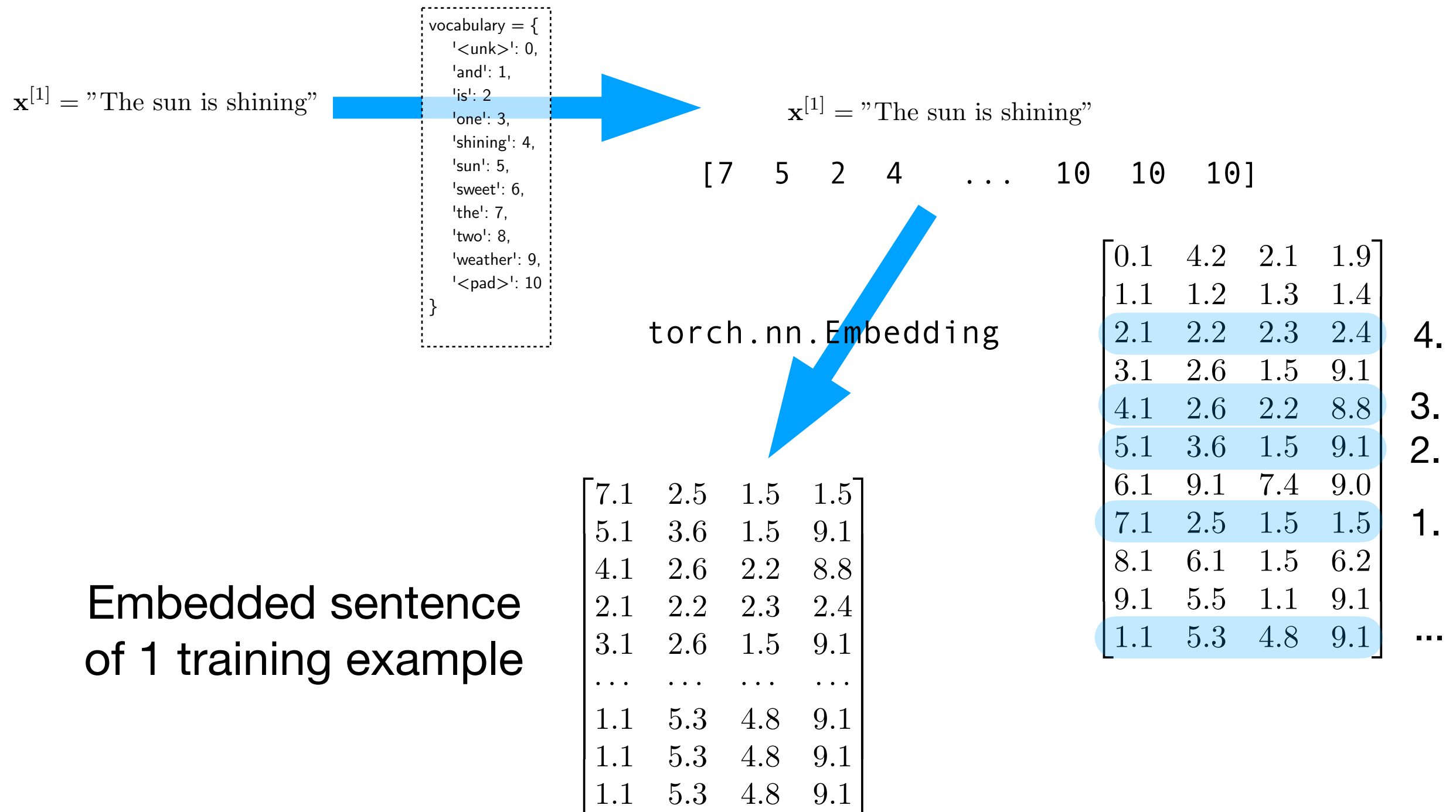
$$\times \begin{bmatrix} 0.1 & 4.2 & 2.1 & 1.9 \\ 1.1 & 1.2 & 1.3 & 1.4 \\ 2.1 & 2.2 & 2.3 & 2.4 \\ 3.1 & 2.6 & 1.5 & 9.1 \\ 4.1 & 2.6 & 2.2 & 8.8 \\ 5.1 & 3.6 & 1.5 & 9.1 \\ 6.1 & 9.1 & 7.4 & 9.0 \\ 7.1 & 2.5 & 1.5 & 1.5 \\ 8.1 & 6.1 & 1.5 & 6.2 \\ 9.1 & 5.5 & 1.1 & 9.1 \\ 1.1 & 5.3 & 4.8 & 9.1 \end{bmatrix}$$

$$= [7.1 \quad 2.5 \quad 1.5 \quad 1.5]$$

Hidden layer output

In Practice, Skip Steps 3 & 4 And ...

use a lookup function (`torch.nn.Embedding`)




IMDB Dataset

- Dataset for binary sentiment classification
- 25,000 highly polar movie reviews for training, and 25,000 for testing

<https://ai.stanford.edu/~amaas/data/sentiment/>


	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0
3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0








Good Resource

 **bentrevett** / **pytorch-sentiment-analysis**

[Code](#) [Issues 14](#) [Pull requests 1](#) [Actions](#) [Projects 1](#) [Wiki](#) [Security](#) [Insights](#)

[master](#) [3 branches](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

 **bentrevett** update to torchtext 0.9 2b666b3 on Mar 12 [108 commits](#)

	assets	fixed typos in max pool figure and size of tensors after convolutiona...	2 years ago
	custom_embeddings	added appendix c - handling embeddings	2 years ago
	data	added optional appendix for how to use your own dataset with torch...	3 years ago
	experimental	refactor of data loading	6 months ago
	.gitignore	fix bug with max_length in tokenizer. improved loading vectors. add...	7 months ago
	1 - Simple Sentiment Analysis.ipynb	update to torchtext 0.9	last month
	2 - Upgraded Sentiment Analysis.i...	update to torchtext 0.9	last month

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

https://colab.research.google.com/github/pytorch/text/blob/master/examples/legacy_tutorial/migration_tutorial.ipynb#scrollTo=EC054Wlr0-xB

Lecture Overview

1. Different Ways to Model Text
2. Sequence Modeling with RNNs
3. Different Types of Sequence Modeling Tasks
4. Backpropagation Through Time
5. Long-Short Term Memory (LSTM)
6. Many-to-one Word RNNs
- 7. RNN Classifiers in PyTorch**