Recurrent Neural Networks (RNN)

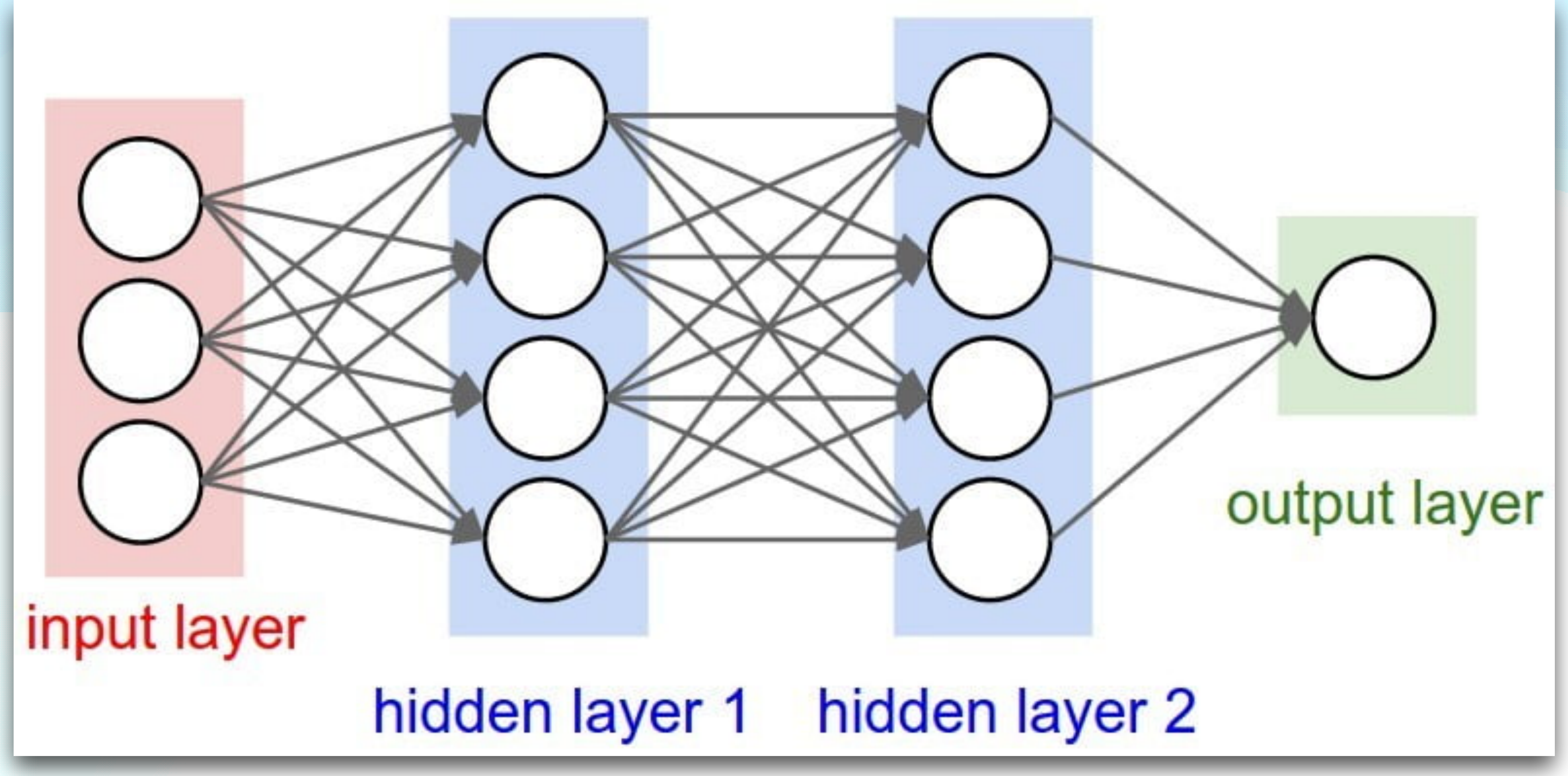
LauzHack Deep Learning Summer Bootcamp

Seyed Parsa Neshaei — July 2024

Several examples and material inspired by the book Deep Learning with Python (François Chollet)

Let's have a recap!

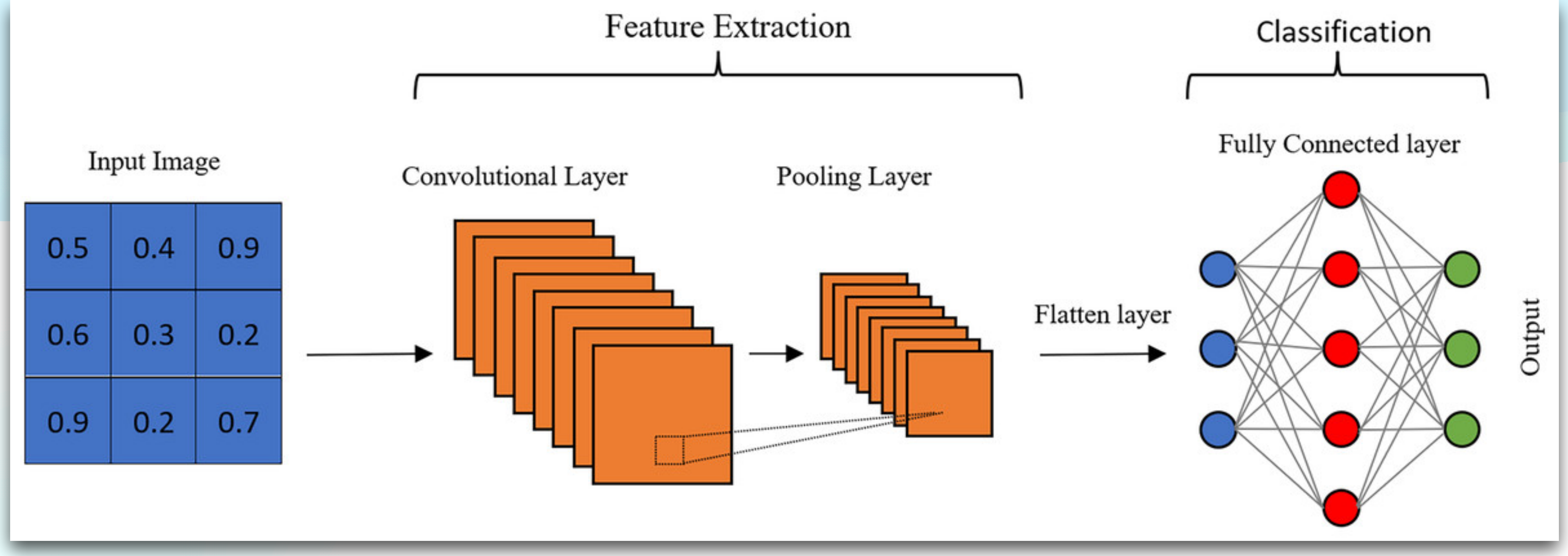
Deep Neural Networks



https://www.bmc.com/blogs/deep-neural-network/

Let’s have a recap!

Convolutional Neural Networks



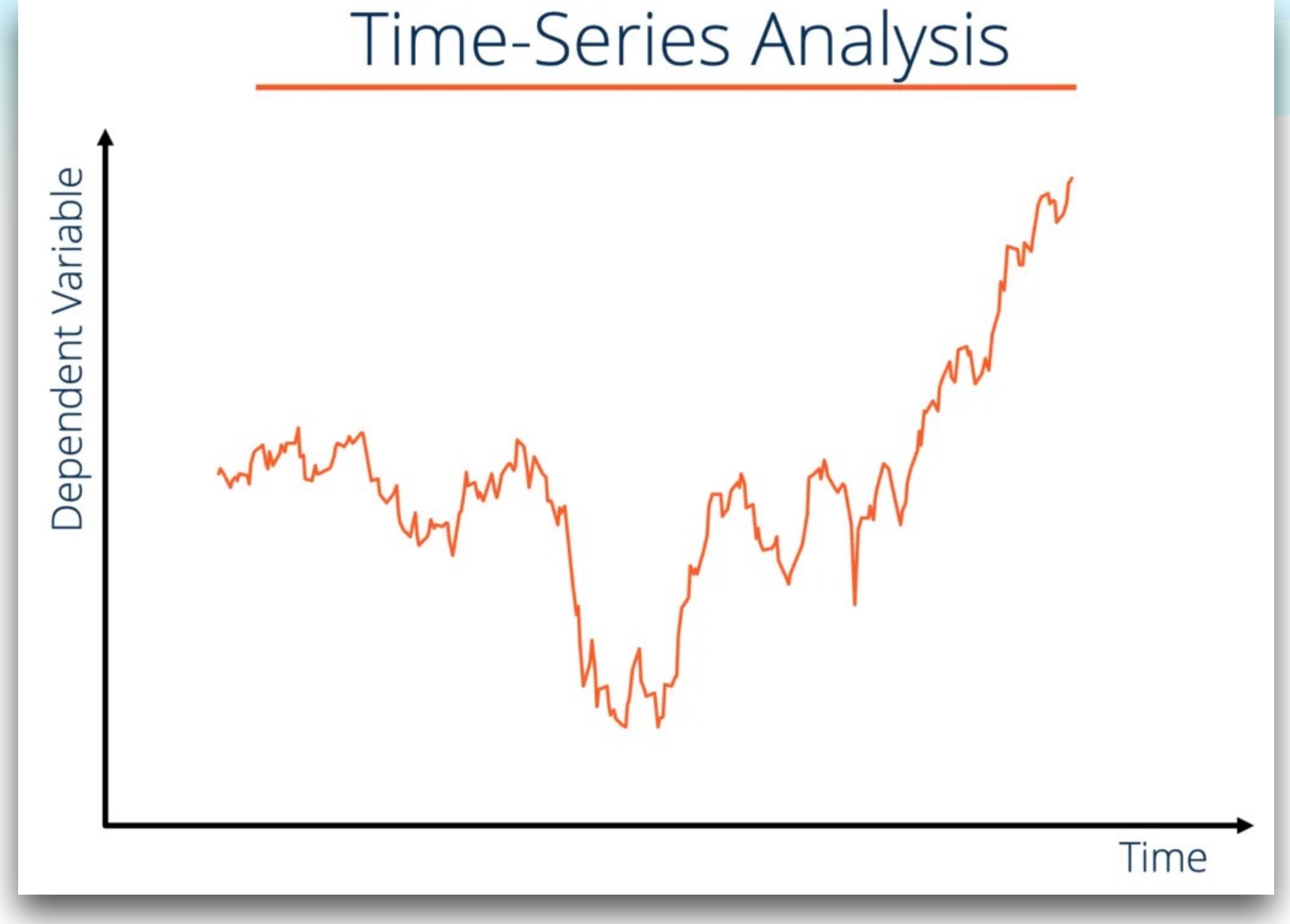
0.5	0.4	0.9
0.6	0.3	0.2
0.9	0.2	0.7

https://peerj.com/articles/cs-1395/

A new task!

Time series

- Data measured at regular intervals, or step-by-step. Examples:
  - Hourly temperature in a city
  - Heights of ocean tides
  - Heartbeats per minute
  - The stock price of a company
  - Answers of students to exam questions
  - A written text (!): *EPFL I like* vs. *I like EPFL*
- Tasks: *classification* or *forecasting*
- Tip: if data semantics change by permutation



https://corporatefinanceinstitute.com/resources/data-science/time-series-data-analysis/

Time Series

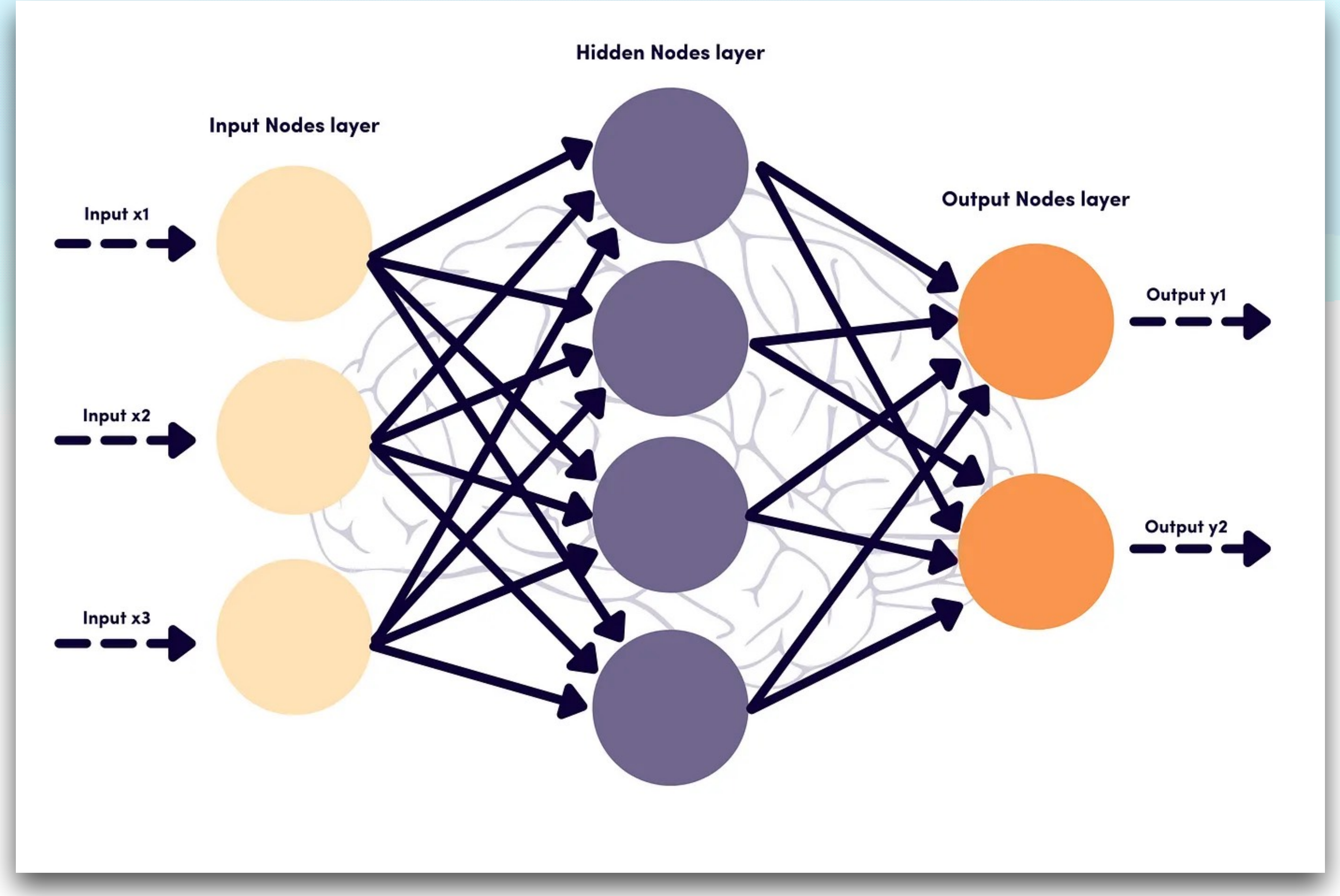
Is classifying/forecasting time series data obvious?

- The model needs to extract trends, rises/falls, cycles, etc.

What is a downside of NNs until now?

No Memory

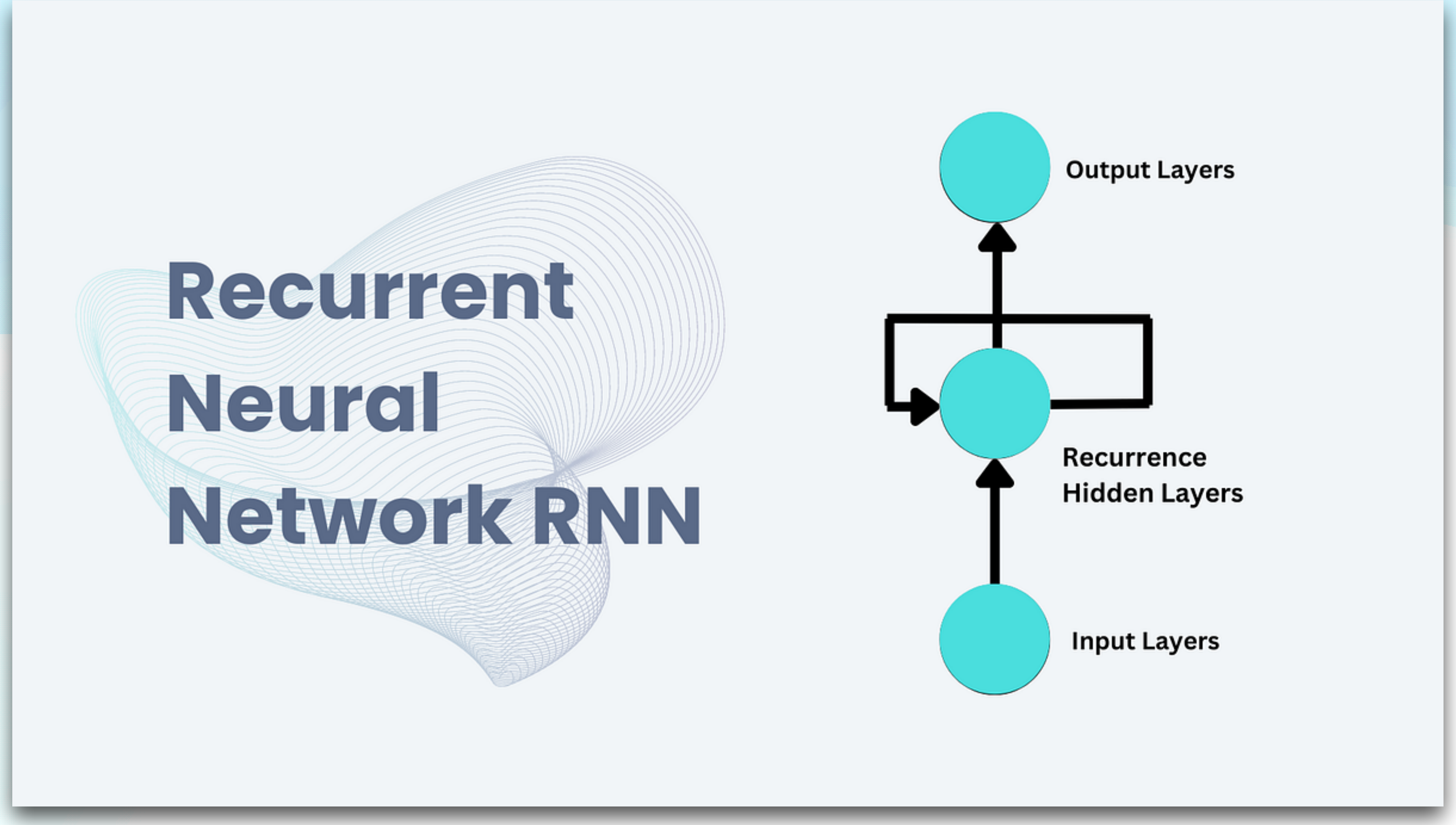
- The processing of each input entry happens independently
- You should *flatten* your data (the whole sequence)
- But time series data come gradually!
- We want to have an *internal state* that changes *over time* (is constantly updated)



https://medium.com/@praveenraj.gowd/understanding-recurrent-neural-networks-rnn-nlp-e2f4cae03a4f

Enter RNNs!

Recurrent Neural Networks



https://medium.com/@praveenraj.gowd/understanding-recurrent-neural-networks-rnn-nlp-e2f4cae03a4f

Enter RNNs!

Recurrent Neural Networks

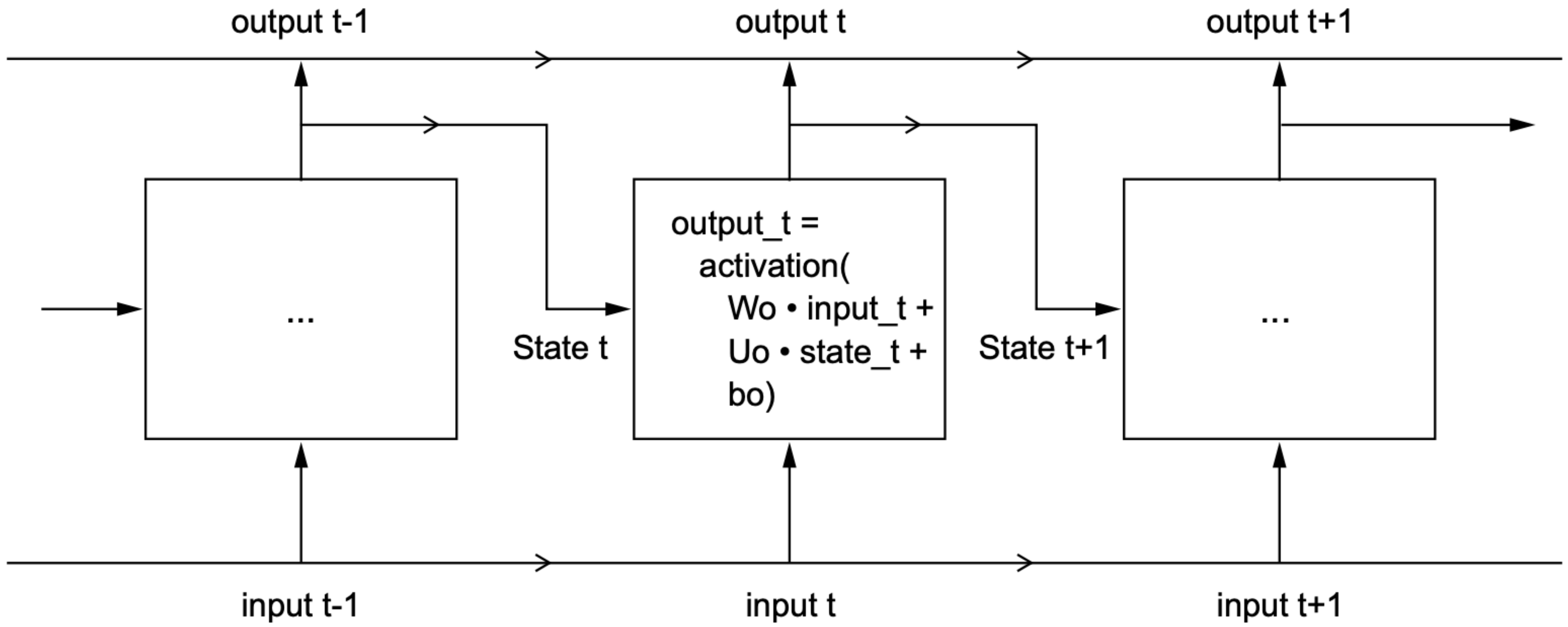


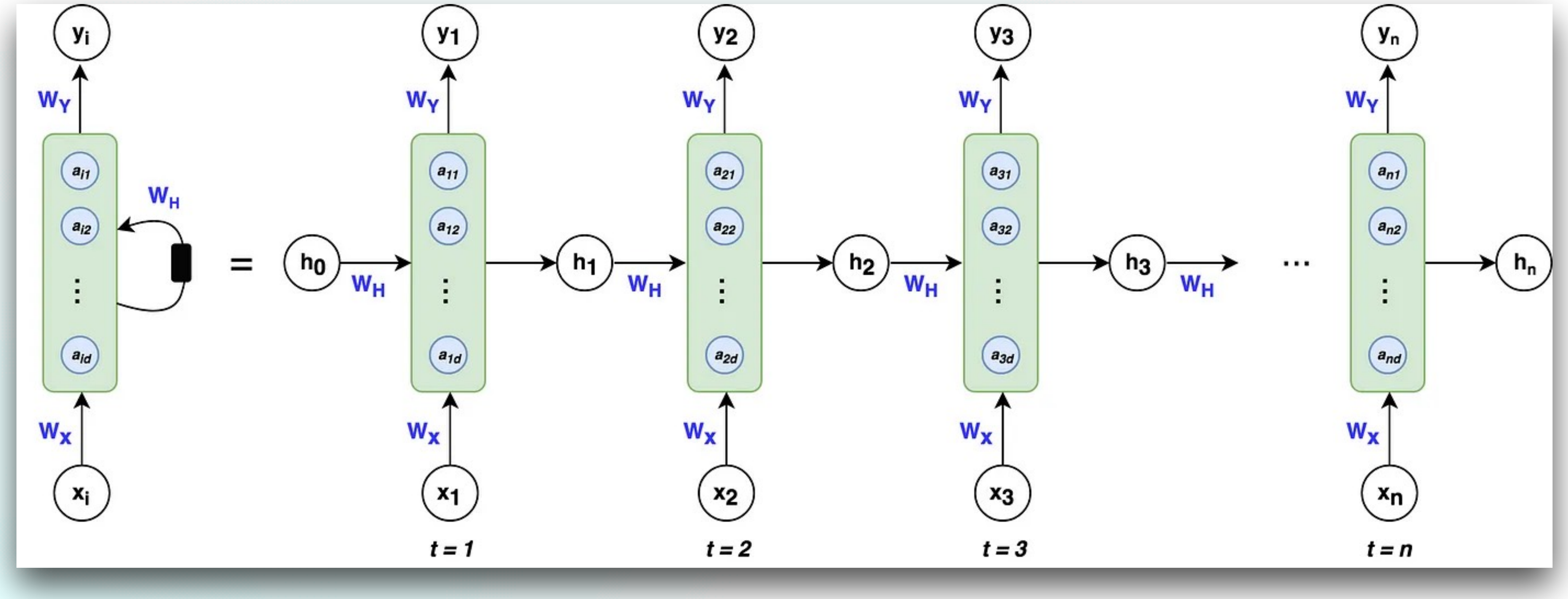
Figure 10.7 A simple RNN, unrolled over time

From the book Deep Learning with Python (François Chollet)

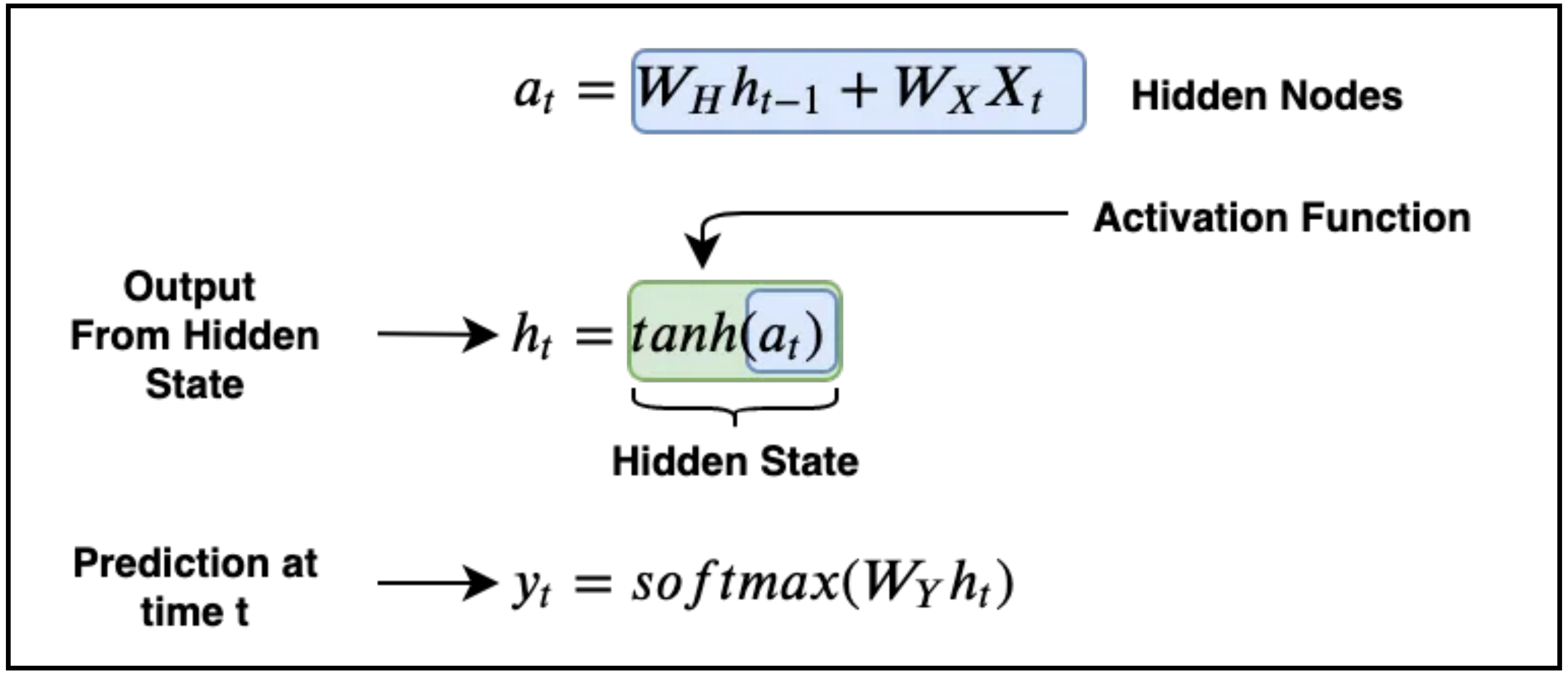
Let’s see it in code!

Recap

- RNNs included a *hidden/memory state* over time, remembering information about sequences
  - Better sequence matching, useful for time series (and textual!) data
  - *Question:* think about how you can make an autocomplete system using it …?



Images from https://medium.com/@praveenraj.gowd/understanding-recurrent-neural-networks-rnn-nlp-e2f4cae03a4f



$$a_t = \boxed{W_H h_{t-1} + W_X X_t}$$ Hidden Nodes

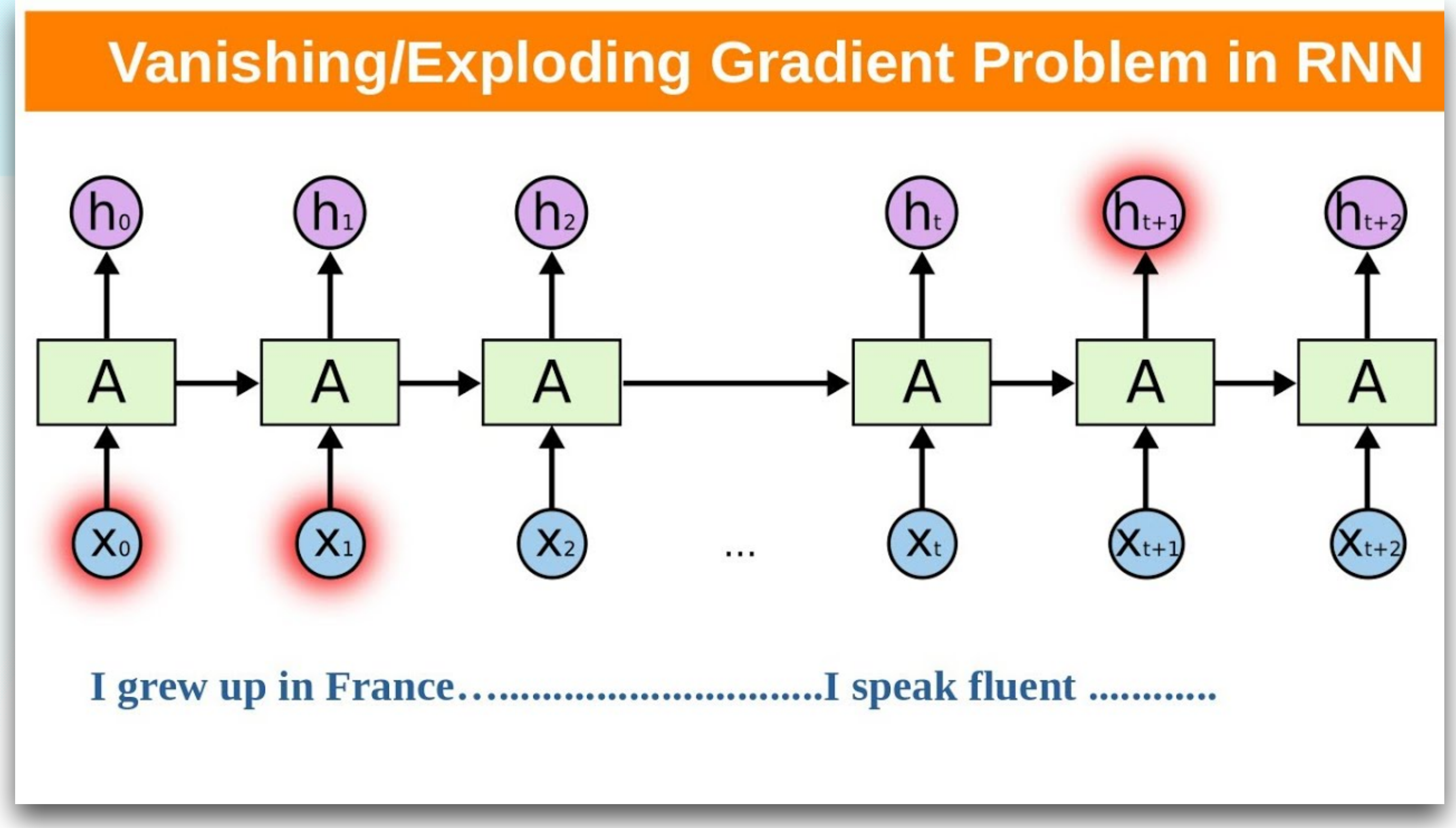
Activation Function

Output From Hidden State $\longrightarrow$ $$h_t = \underbrace{\boxed{tanh(\boxed{a_t})}}_{\text{Hidden State}}$$

Prediction at time t $\longrightarrow$ $$y_t = softmax(W_Y h_t)$$

What if the pattern continues?

- Vanishing gradient problem: multiplying chain rule operations < 1
- Makes it difficult for RNNs to predict based on entries occurring at the start of long sequences (almost no weight update) — long term dependency…



Vanishing/Exploding Gradient Problem in RNN

I grew up in France…..............................I speak fluent ....

https://www.youtube.com/watch?app=desktop&v=nLPRvGogtrI

So, this was an RNN…

h0
h1
h2
ht
ht+1
ht+2
A
A
A
A
A
A
X0
X1
X2
...
Xt
Xt+1
Xt+2

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

So, this was an RNN…

h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

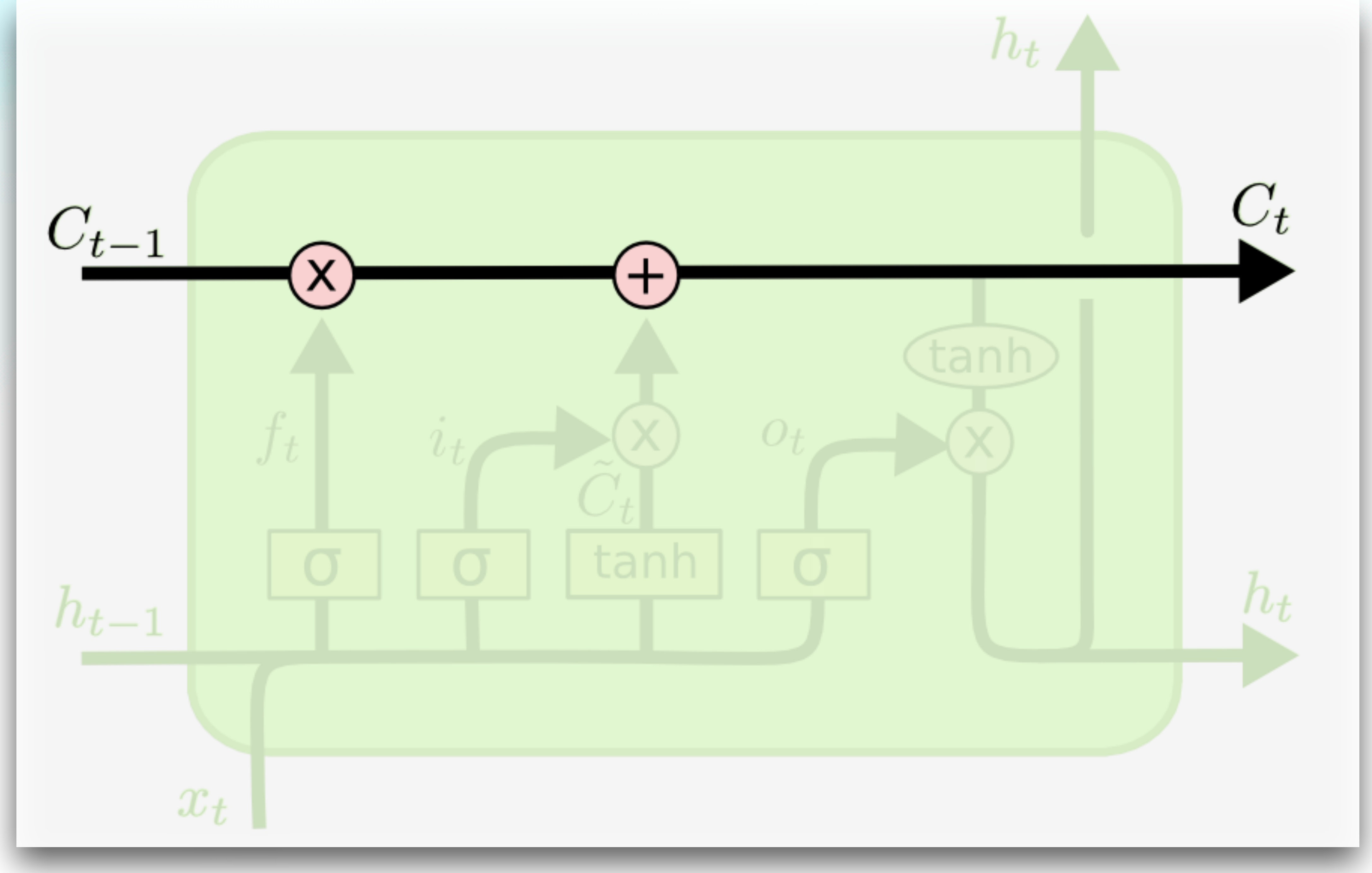
Enter LSTMs!

h_{t-1}
h_t
h_{t+1}
A
x
+
tanh
x
x
σ
σ
tanh
σ
A
x_{t-1}
x_t
x_{t+1}

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Cell State

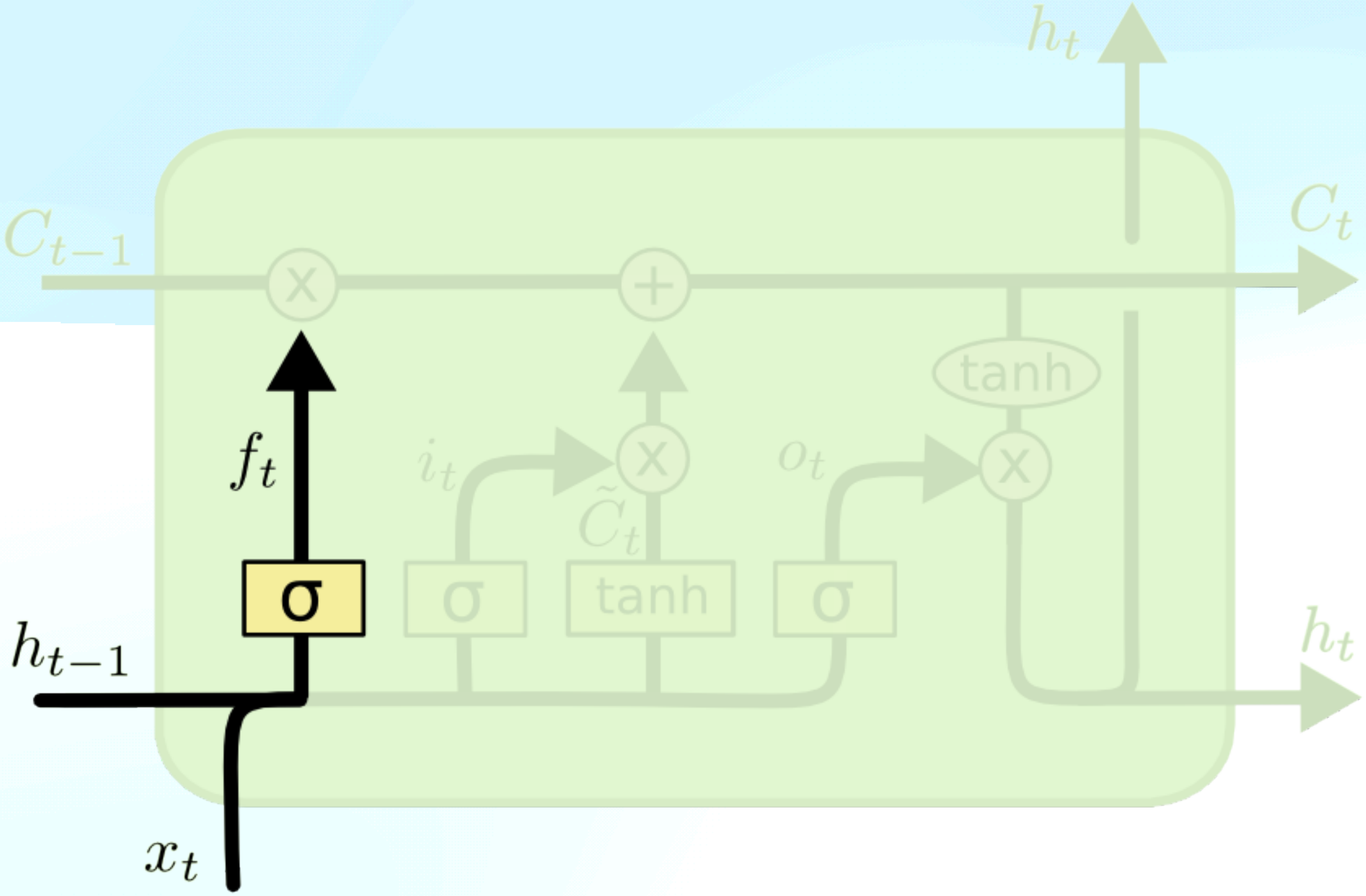
- A line running in the whole chain (info can even flow without change)
- Data can be added/removed to the cell state via several *gates*
  - Output of sigmoid between zero (let nothing go) and one (let everything go)



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

1) Forget Gate

- Example: forgetting information after a new subject in a sentence

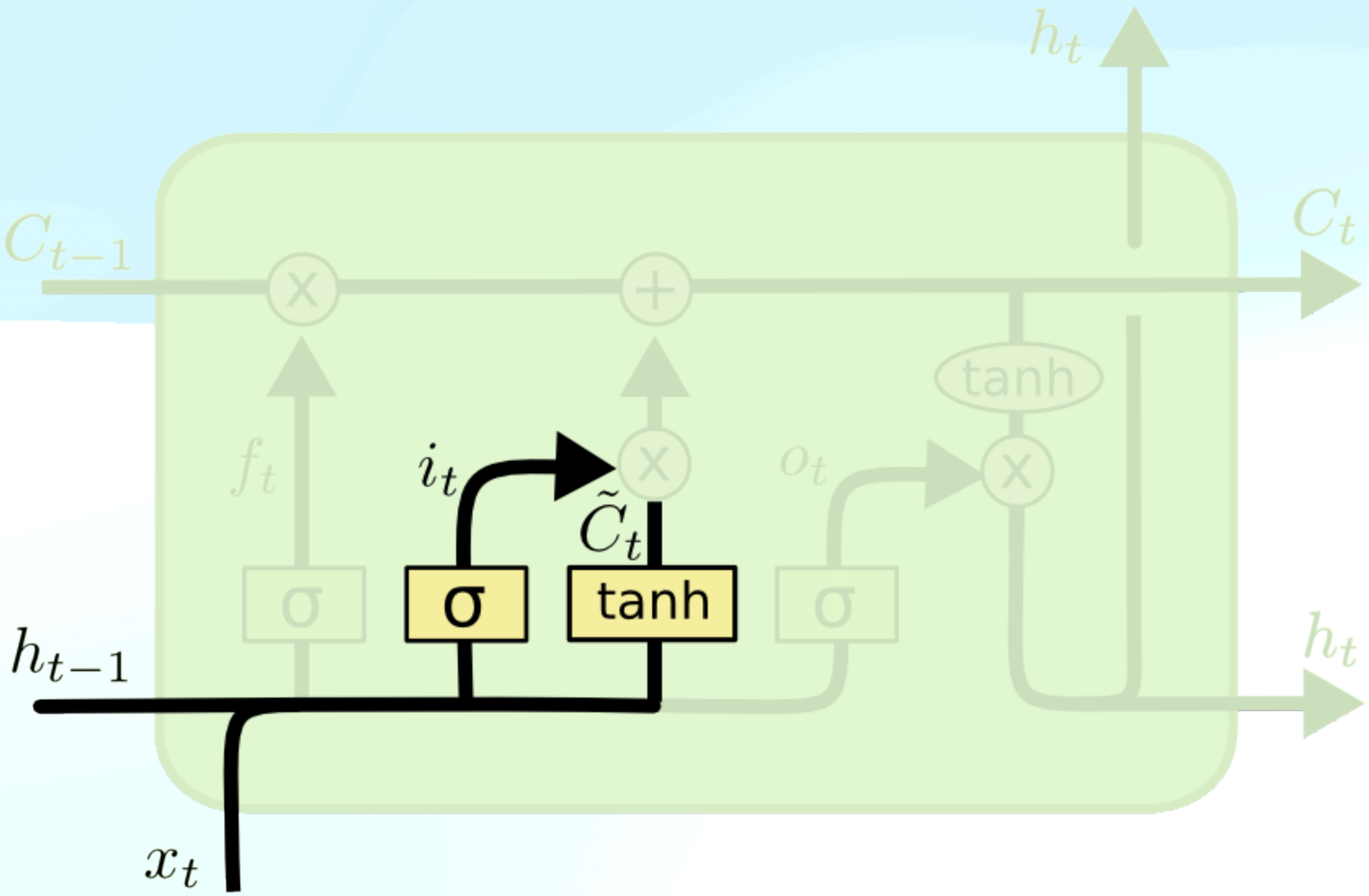


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

2) Input Gate

- What new information are we going to store in the cell state

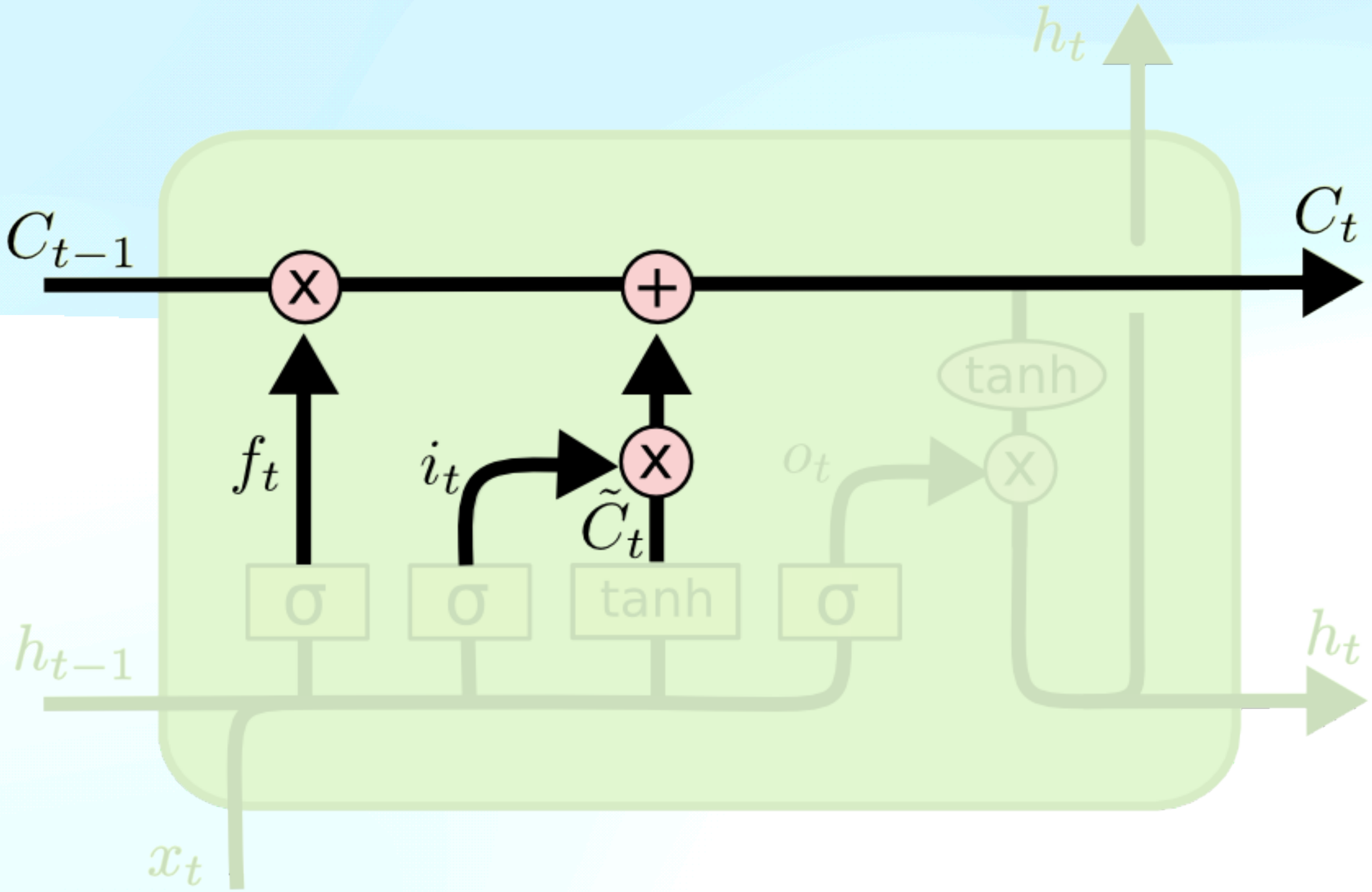


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

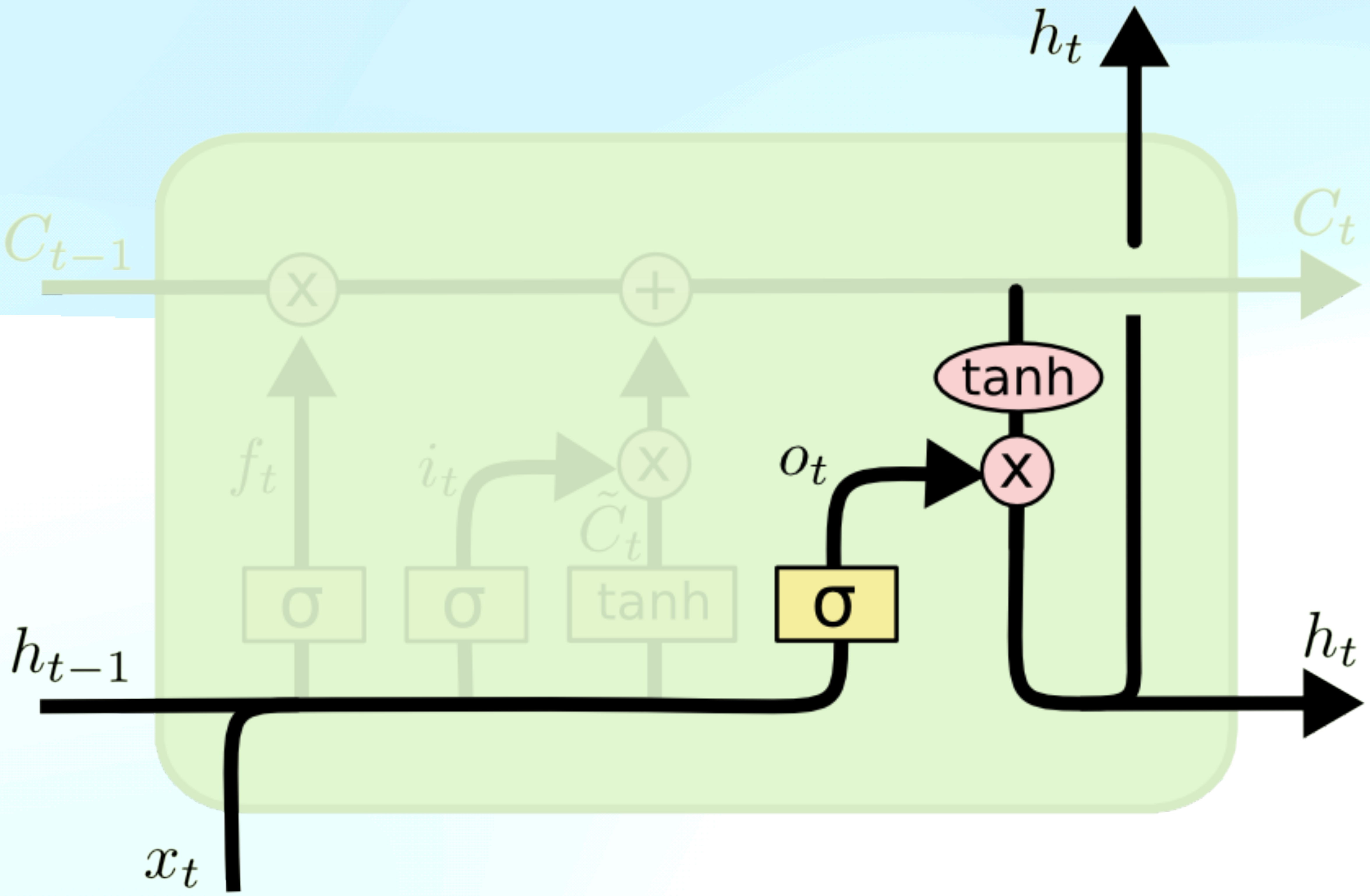
New Cell State



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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3) Output Gate



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

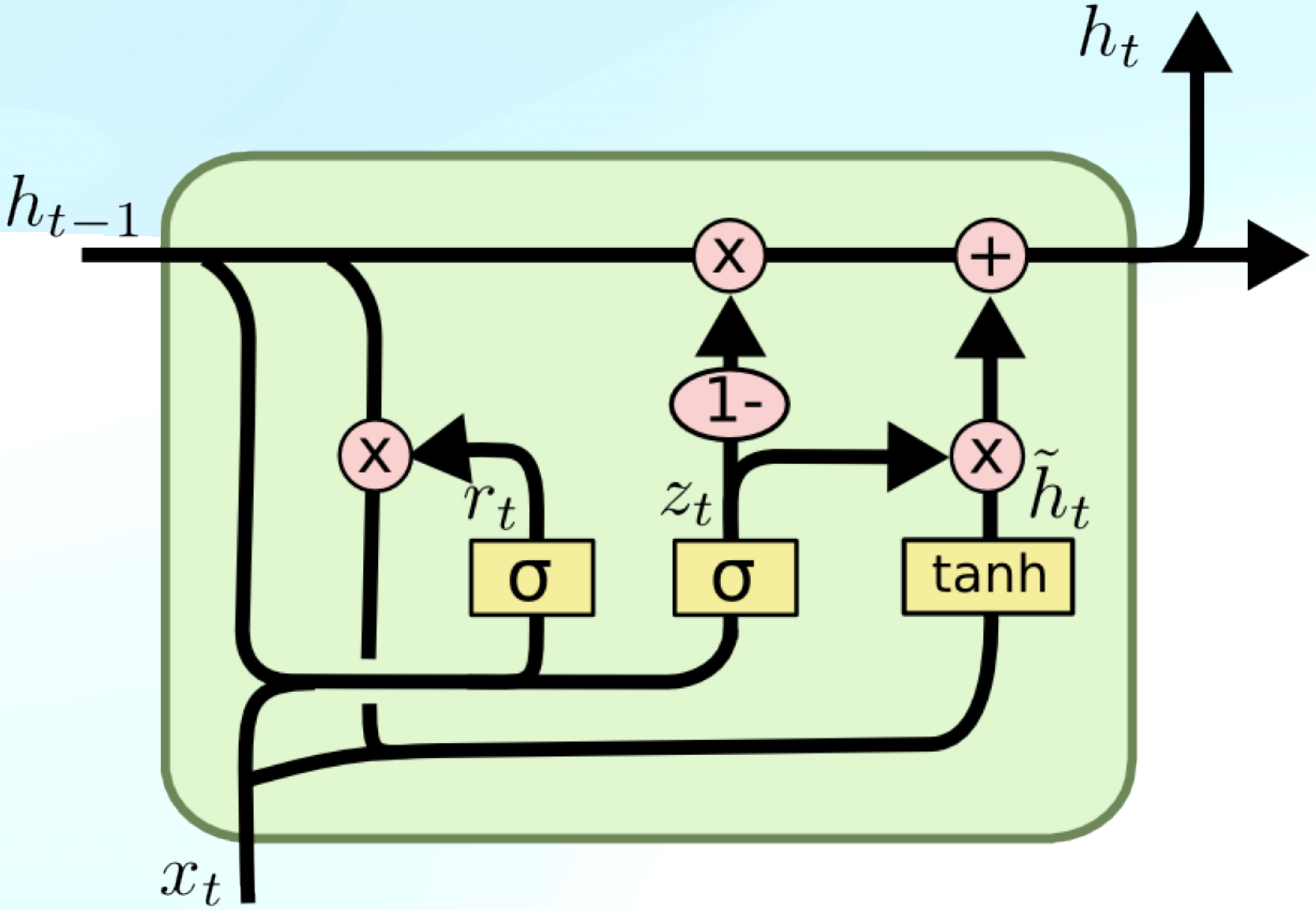
$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Let’s see it in code!

A variation: Gated Recurrent Unit (GRU)

- Combines forget and input gates into an *update* gate
- Merges the cell state and hidden state



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

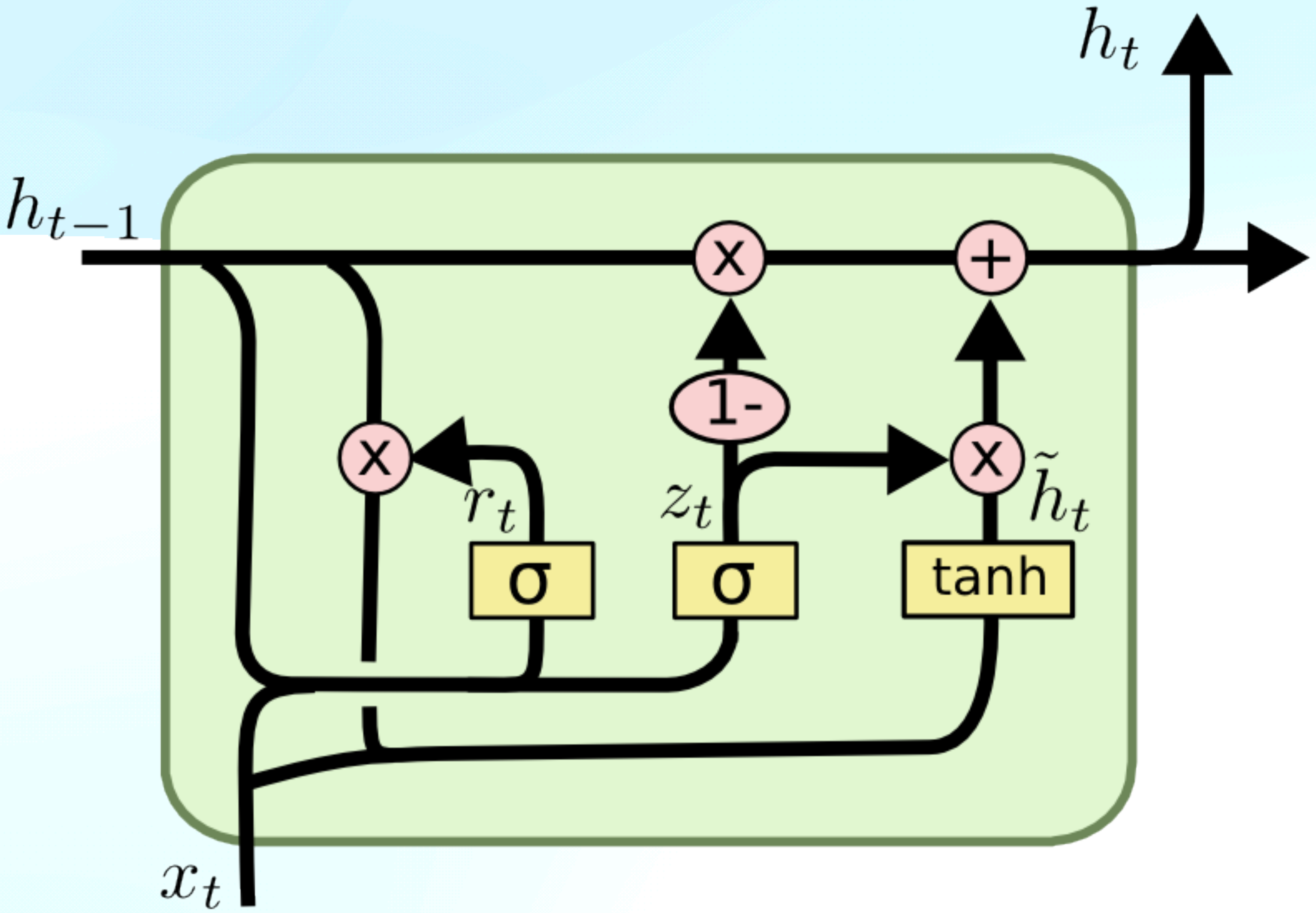
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Reset Gate in GRU

- Changes the hidden state (short-term memory)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

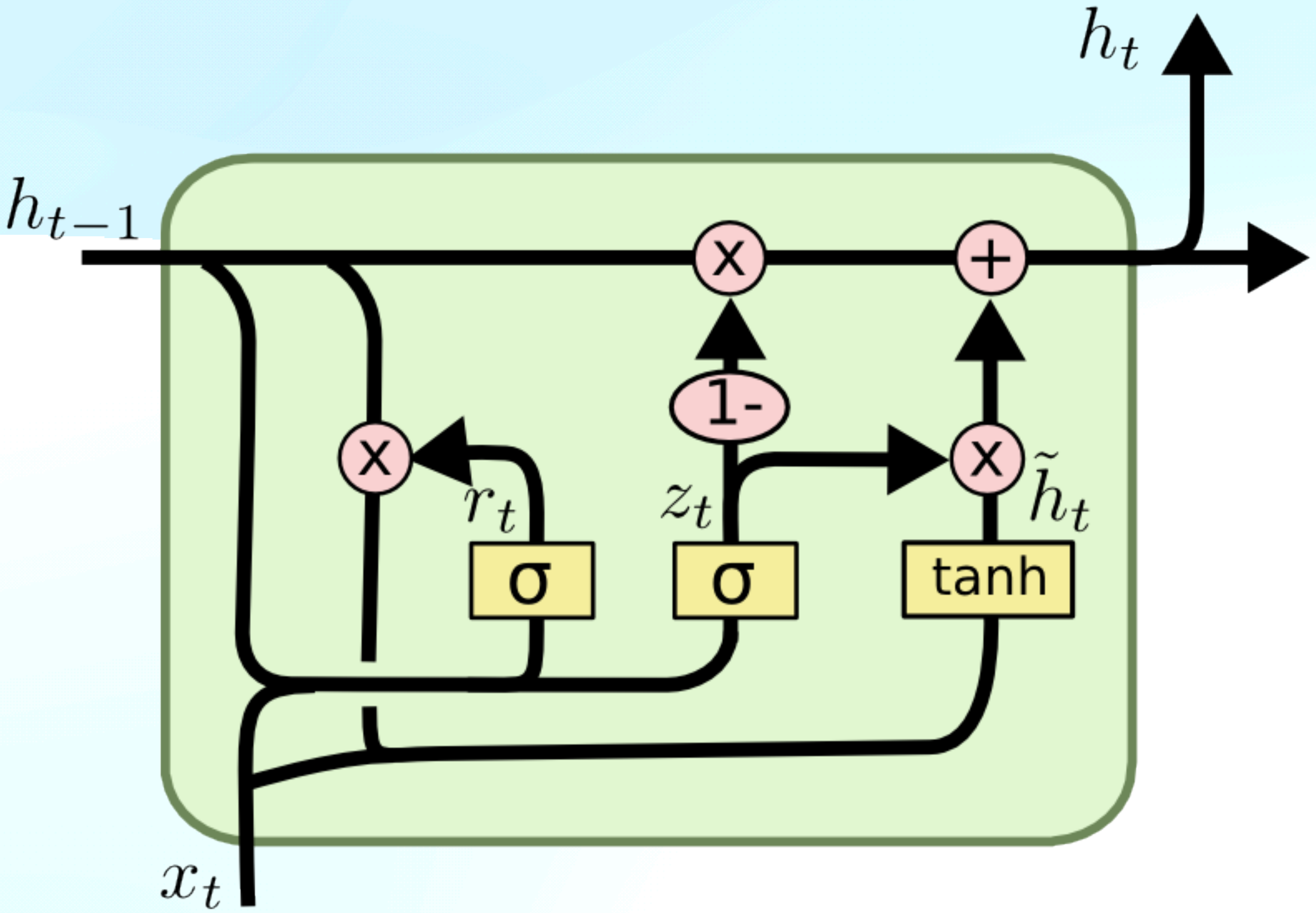
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Update Gate in GRU

- For long-term memory



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Let’s see it in code!

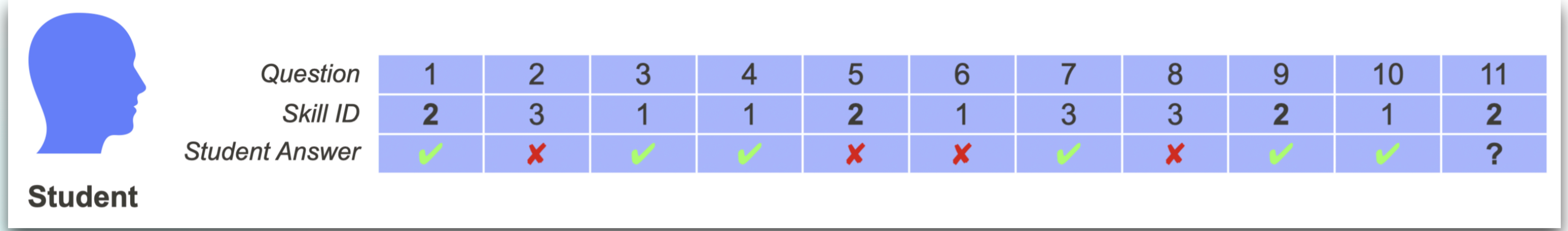
Recap

- LSTMs and GRUs can help mitigate the vanishing gradient problem
- Also useful for text (come to NLP lectures in the *more advanced topics* week!)
    - Embeddings
- One main concern: limited interpretability
- The next step is attention and transformers -> tomorrow!

A real case study!

Student Knowledge Tracing

- Students interact with a learning system over time and answer questions
  - Imagine assignments in a massive online open course
- Students answer questions of certain topics right or wrong
- Can we predict their answers to the next questions?
  - Why is this useful?



Student

Question	1	2	3	4	5	6	7	8	9	10	11
Skill ID	**2**	3	1	1	**2**	1	3	3	**2**	1	**2**
Student Answer	✓	✗	✓	✓	✗	✗	✓	✗	✓	✓	**?**

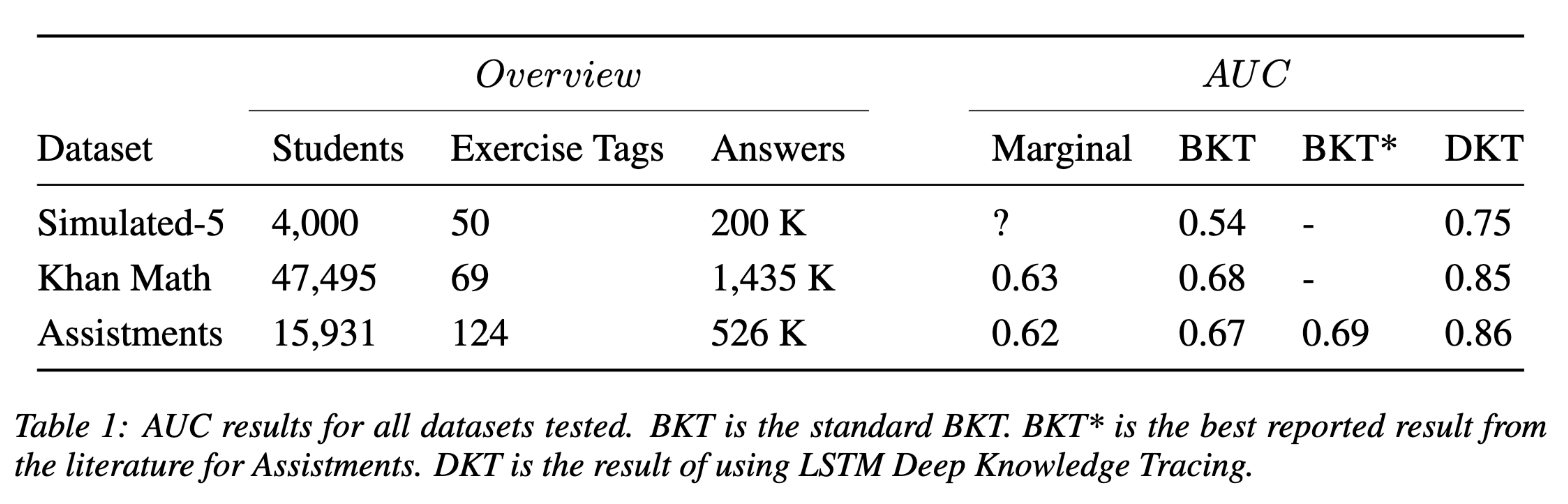
Neshaei, Seyed Parsa, et al. "Towards modeling learner performance with large language models." arXiv preprint arXiv:2403.14661 (2024).

A real case study!

Student Knowledge Tracing

- Researchers have explored Bayesian and Markov models to predict the *mastery* of each skill among students

- Piech et al. (2015) achieved state-of-the-art-at-the-time results with LSTMs



	Overview			*AUC*			
Dataset	Students	Exercise Tags	Answers	Marginal	BKT	BKT*	DKT
Simulated-5	4,000	50	200 K	?	0.54	-	0.75
Khan Math	47,495	69	1,435 K	0.63	0.68	-	0.85
Assistments	15,931	124	526 K	0.62	0.67	0.69	0.86

Table 1: AUC results for all datasets tested. BKT is the standard BKT. BKT is the best reported result from the literature for Assistments. DKT is the result of using LSTM Deep Knowledge Tracing.*

Piech, Chris, et al. "Deep knowledge tracing." Advances in neural information processing systems 28 (2015).

Batch Normalization

Batch Normalization

What is it?

- A technique to improve model generalizability and speeds up training (enables higher learning rates)
- Find the mean and SD of mini-batches of inputs in training, then normalize
- Also makes the model more robust to the initial initialization
- Just add after activation functions
- People do NOT suggest its use for RNNs!

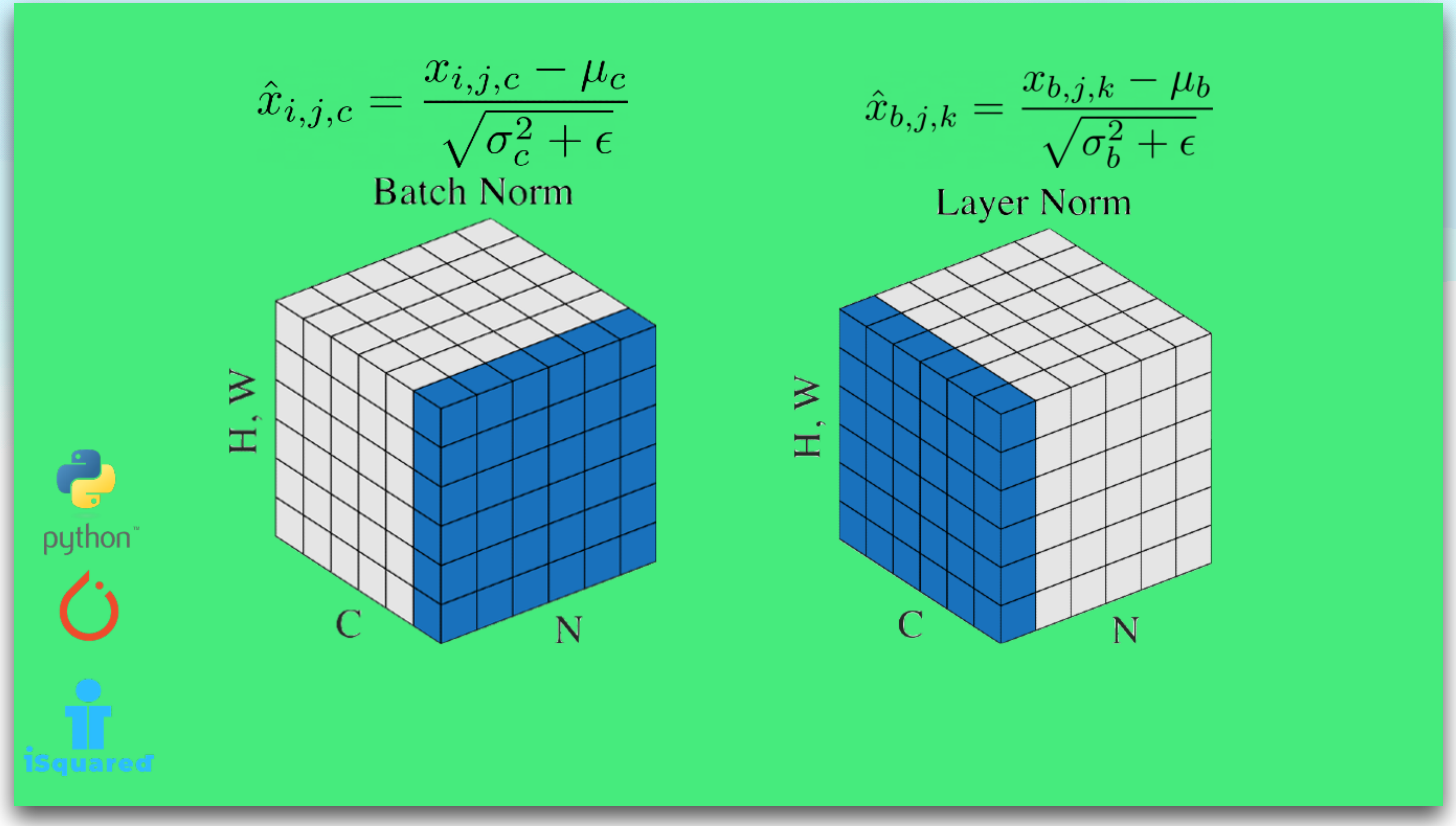
```
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()
        self.fc1 = nn.Linear(128, 64)
        self.bn1 = nn.BatchNorm1d(64)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(64, 10)

def forward(self, x):
        x = self.fc1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Code from https://medium.com/@pouyahallaj/batch-normalization-a-deep-dive-for-machine-learning-enthusiasts-60e4a76d10e8

Layer Normalization

What is it?

- Normalize the inputs of layers, not batches
- Also used with RNNs
- Common in Transformers (come tomorrow!)
- `torch.nn.LayerNorm`


$\hat{x}_{i,j,c} = \frac{x_{i,j,c} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}$
Batch Norm
H, W
C
N
$\hat{x}_{b,j,k} = \frac{x_{b,j,k} - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}}$
Layer Norm
H, W
C
N
python
iSquared

https://isquared.digital/blog/2023-03-15-illustrated-batch-vs-layer-norm/

Thank you!