# Science of Science Summer School
# Day 4: Deep Learning Recurrent Neural Networks

Daniel E. Acuna

Syracuse University

Virtually hosted by Syracuse University – July 26, 2021 to August 6, 2021

# Agenda

- Pytorch
- Defining  a model in Pytorch
- Multilayer perceptron
- Future steps

# Spark does not have modern layers, activations, and optimization methods

- Modern deep learning uses other type of layers:
  - Convolutional Layers
  - Recurrent layers
  - Pooling layers
- Other types of activations that do not get stuck during learning
  - Rectified Linear Unit (ReLU)
  - Tangential Hyperbolic (Tanh)
- Other optimization algorithms:
  - ADAM, AdaGrad, RMSProp

# General computation graphs

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

# General computation graphs

Suppose that you wanted to compute the *dL* / *dc*

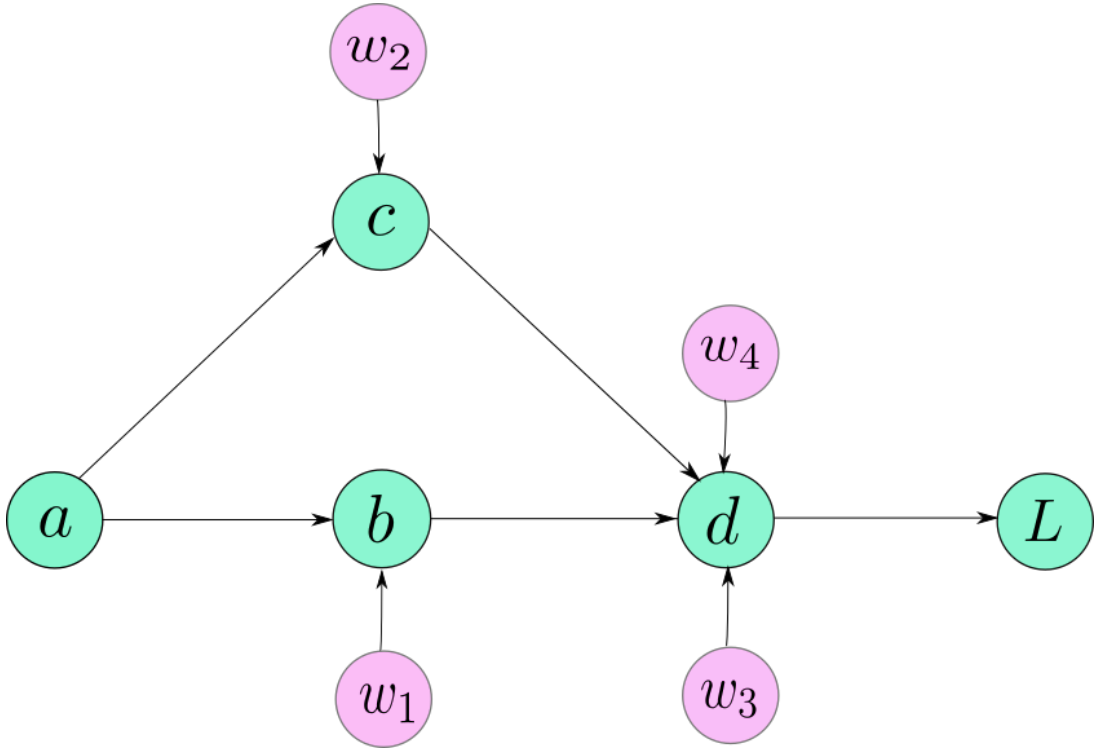
```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

# General computation graphs

Suppose that you wanted to compute the *dL / dc*

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

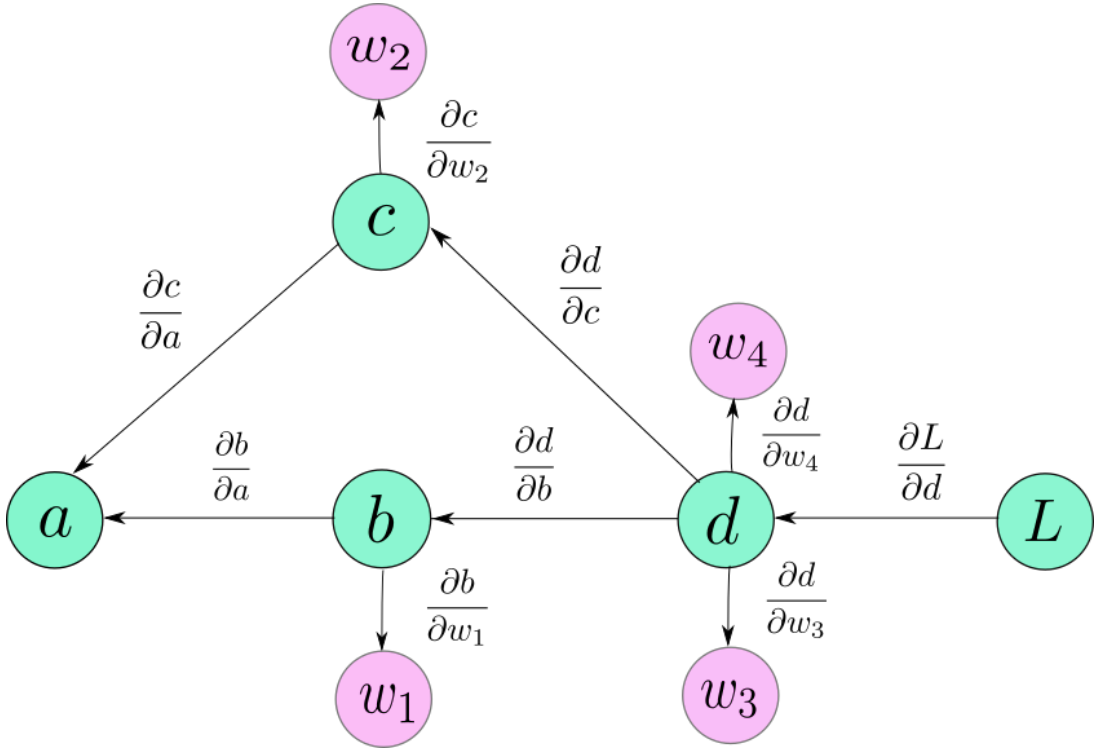



Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

# General computation graphs

Suppose that you wanted to compute the *dL / dc*

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```



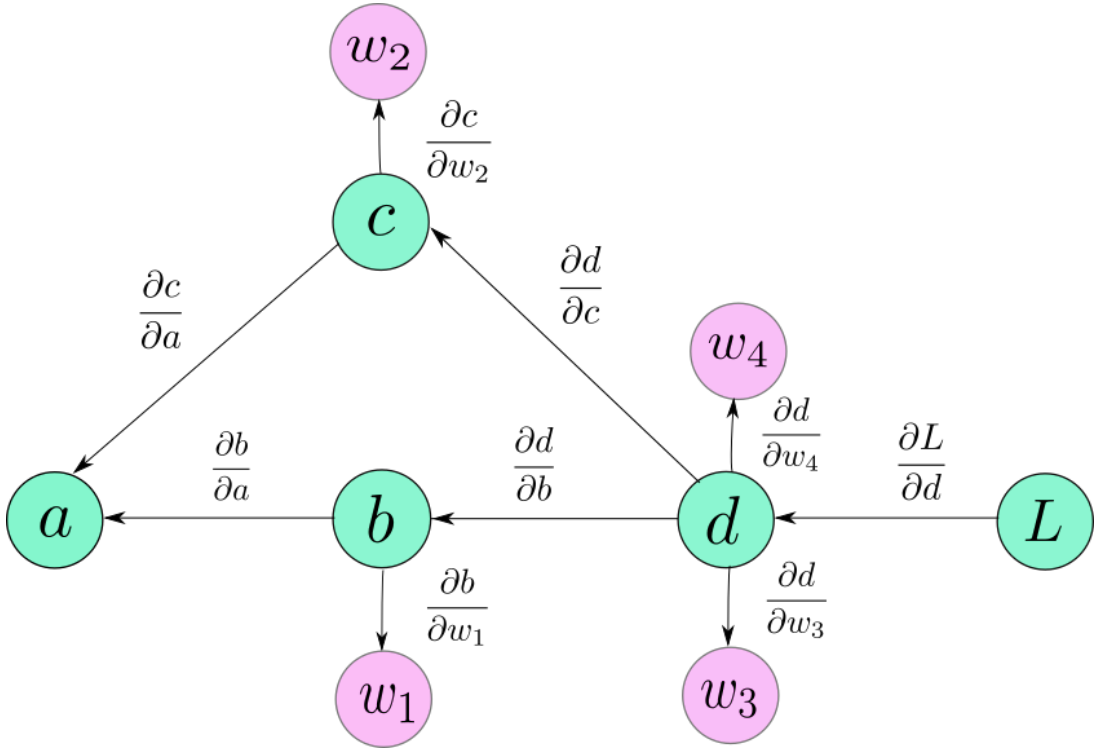

Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

# General computation graphs

Suppose that you wanted to compute the *dL / dc*

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

dL/dc = dL/dd * dd/dc

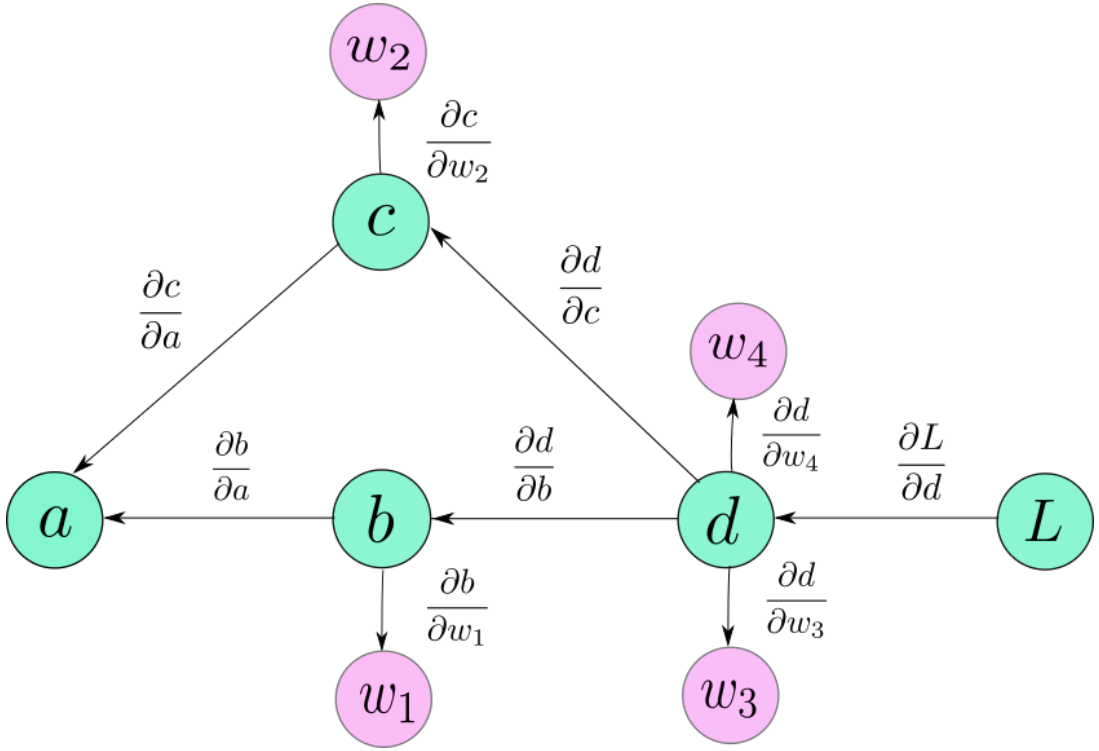



Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

# General computation graphs

What about *dL / da?*

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```




Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

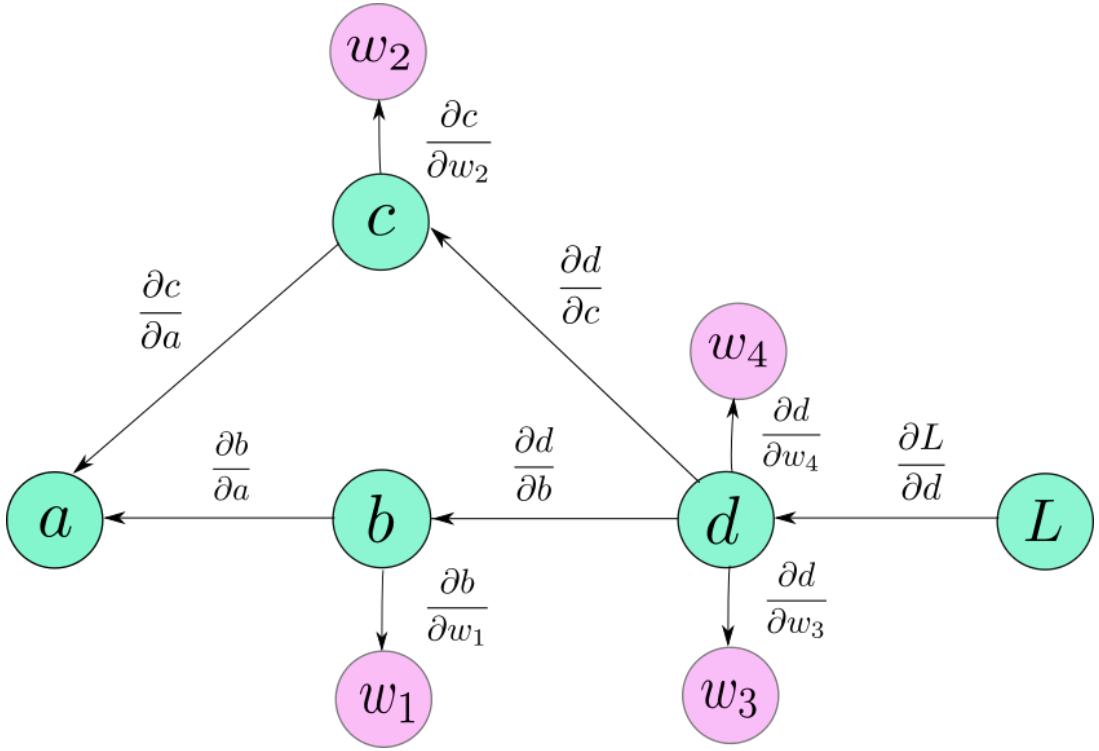
# General computation graphs

What about *dL / da?*

```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

We have two paths L->d->b->a
and L->d->c->a so we add them:

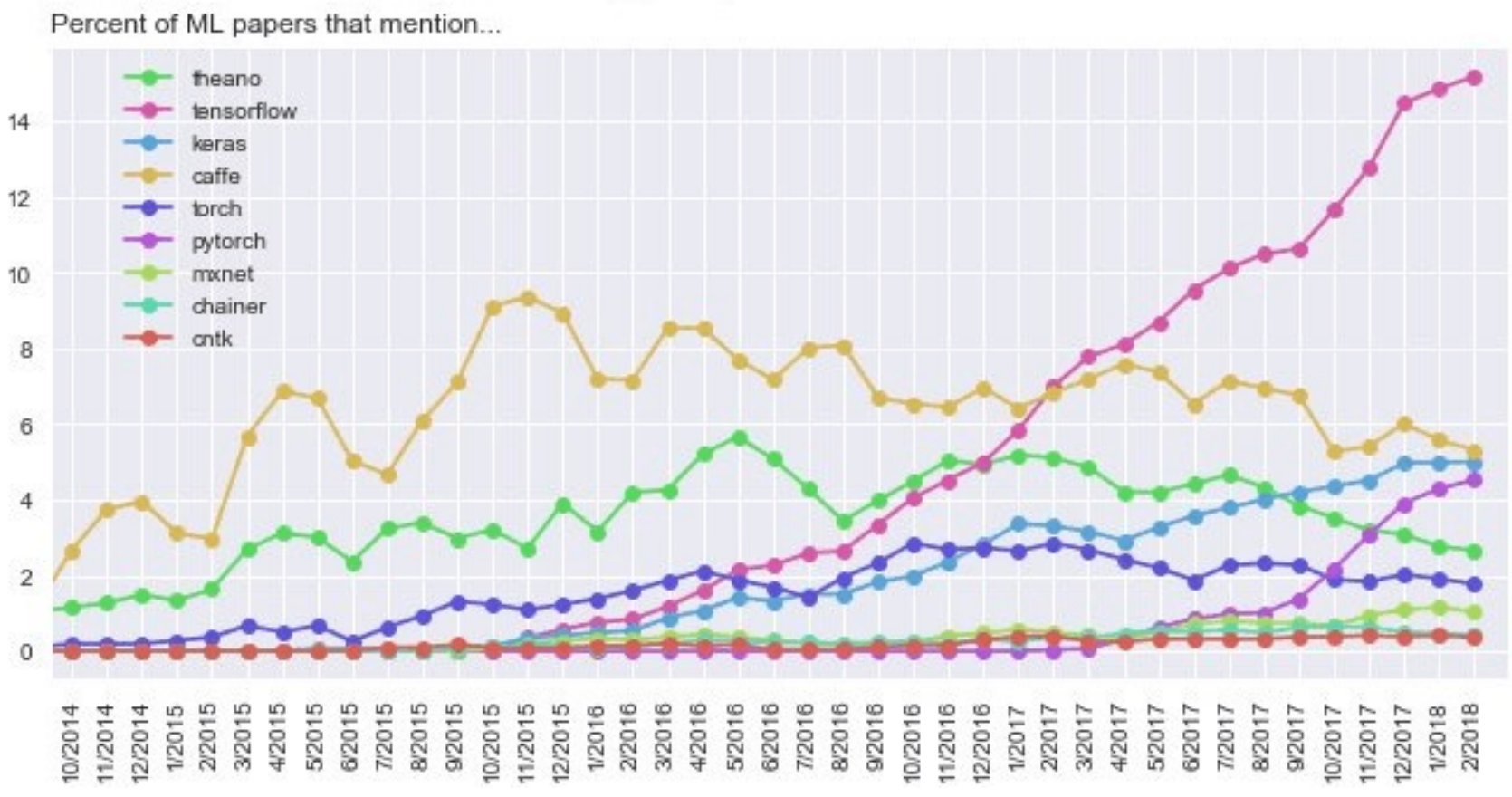
dL/da = dL/dd dd/dc dc/da + dL/dd dd/db db/da




Taken from https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

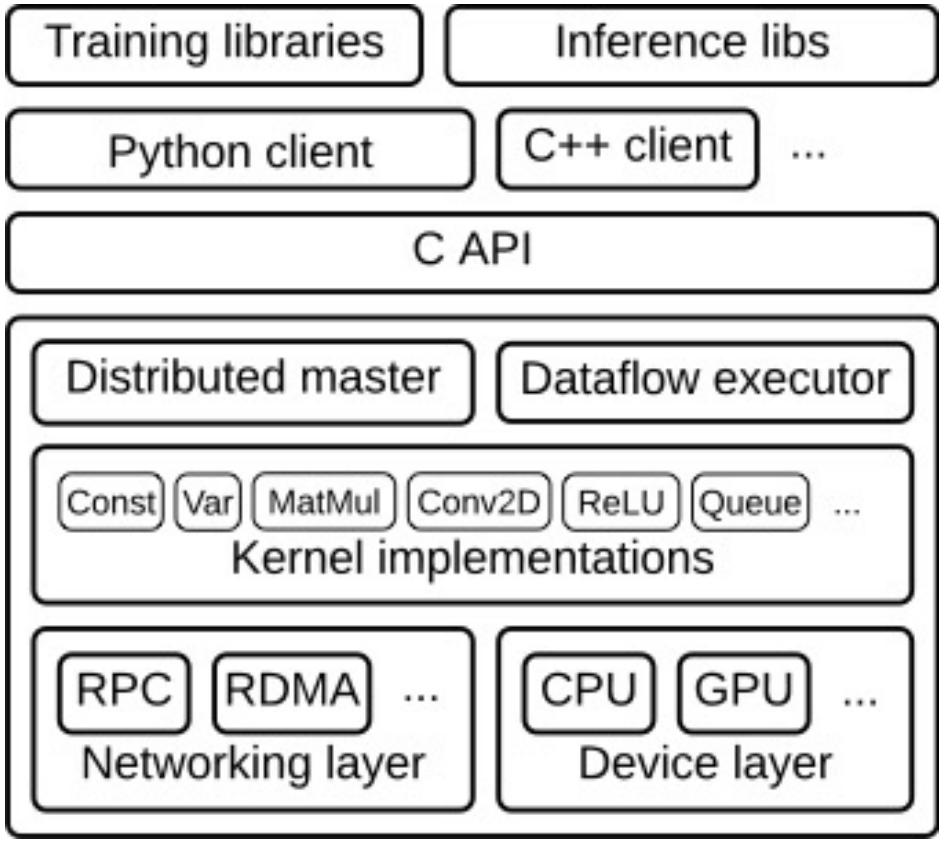
# Software

- Computer packages for deep learning automatically compute and store these derivative graphs



# Tensorflow (Google)

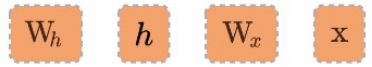
- You have to predefine your computation (“similar” to Spark Dataframes)
- You find bugs only after you execute your code
- Client-server architecture: Python client – C server
- It is the most well-developed, production-ready, and popular
- It is hard to debug and do research on.



# Pytorch (Facebook)

- Everything is in Python
- Eager execution mode: easy to debug
- Dynamic computation graph: it is built as you create your variables
- Easy to try new ideas: heavily used in research

A graph is created on the fly




```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

# Pytorch

- You can install it from https://pytorch.org/
- In notebook.acuna.io, you have to install it using:

```
pip install torch torchvision
```

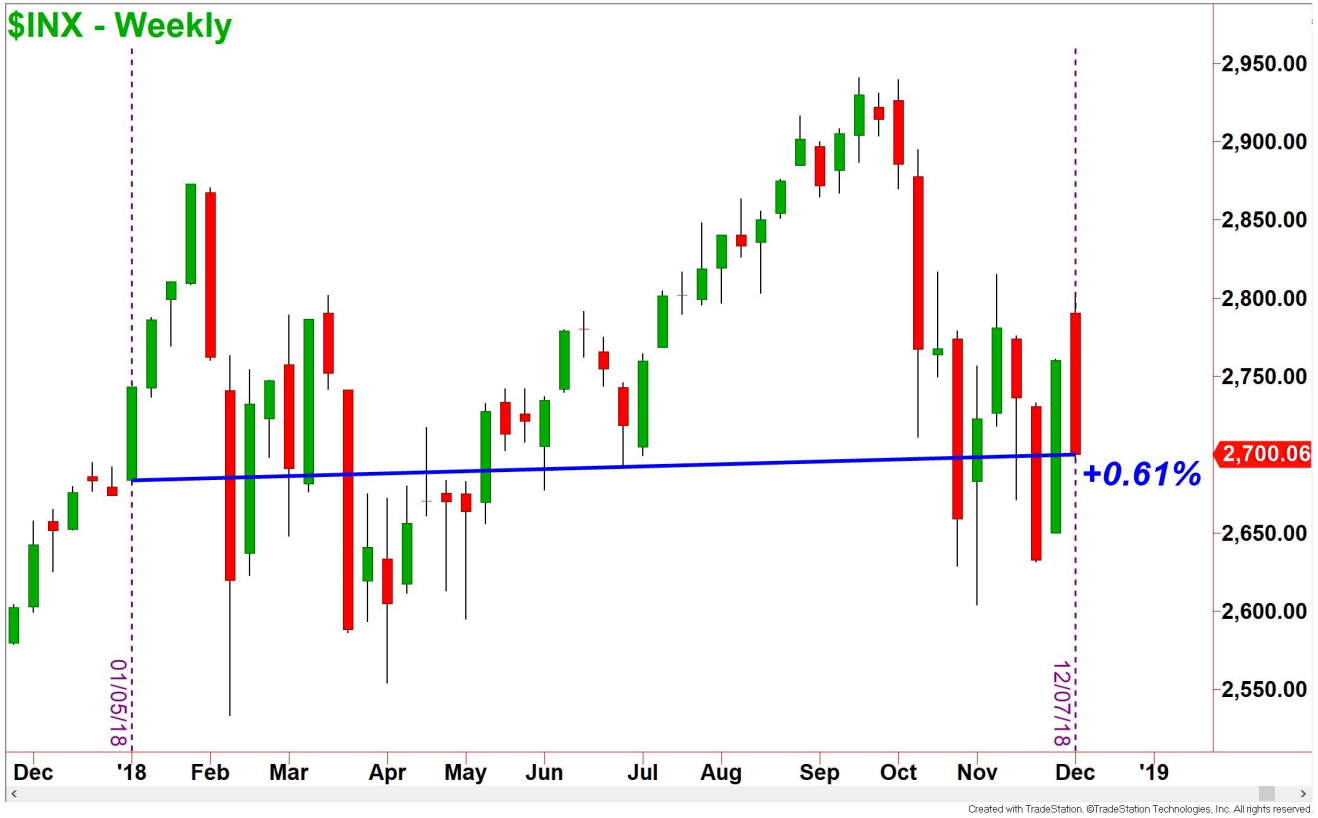
- Unfortunately, you might have to do this if your server is shutdown by Kubernetes

# Demo

# Recurrent neural networks

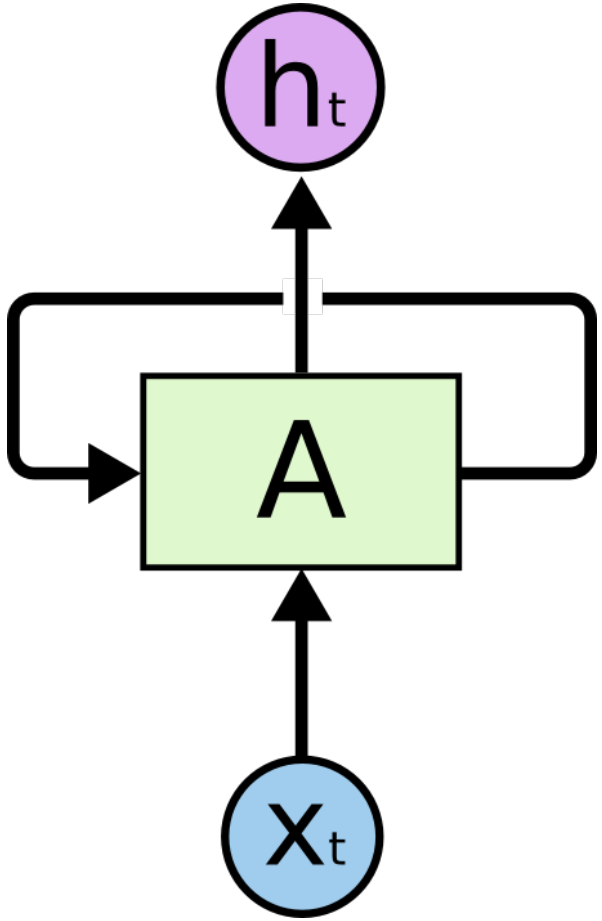
# Recurrent neural networks

- Some data has temporal structure in them

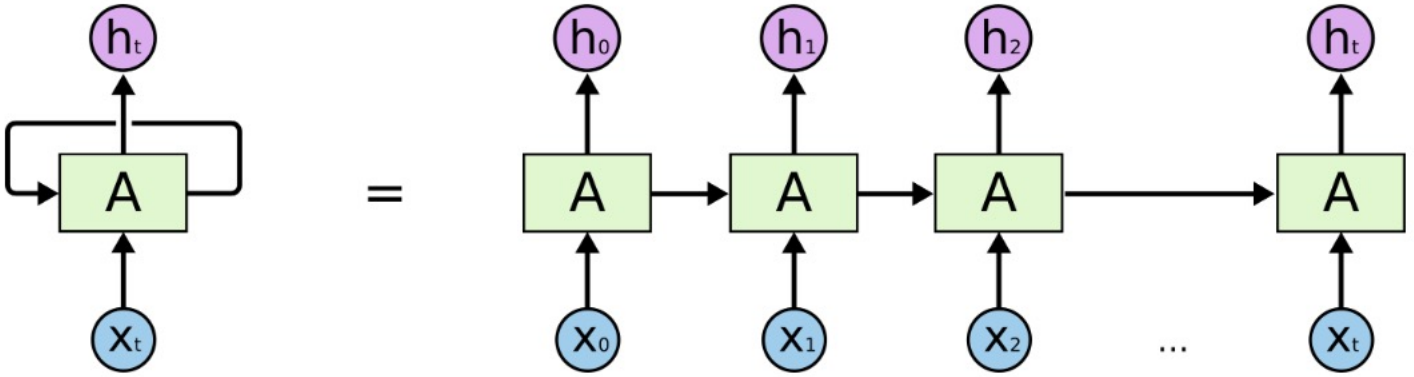


# Recurrent neural networks (2)

- We hope to understand the temporal structure in the data
- X is the input
- A is the *temporal dynamics* of the network
- *h* is the output of the network



# Recurrent neural networks: simple

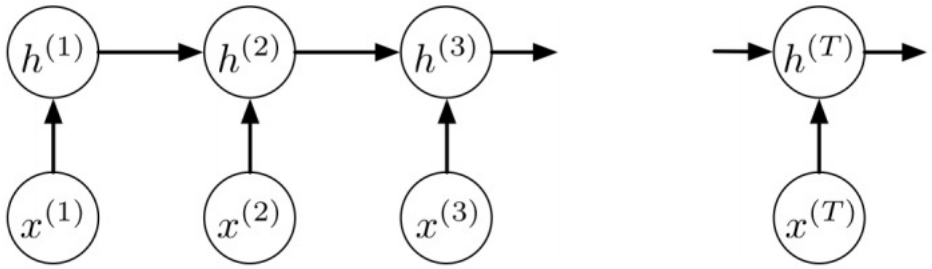



$$\mathbf{h}^{(4)} = f(f(f(\mathbf{h}^{(1)}, \mathbf{x}^{(2)}), \mathbf{x}^{(3)}), \mathbf{x}^{(4)})$$

For example $$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

# RNN: a problem with gradients

- Image this:



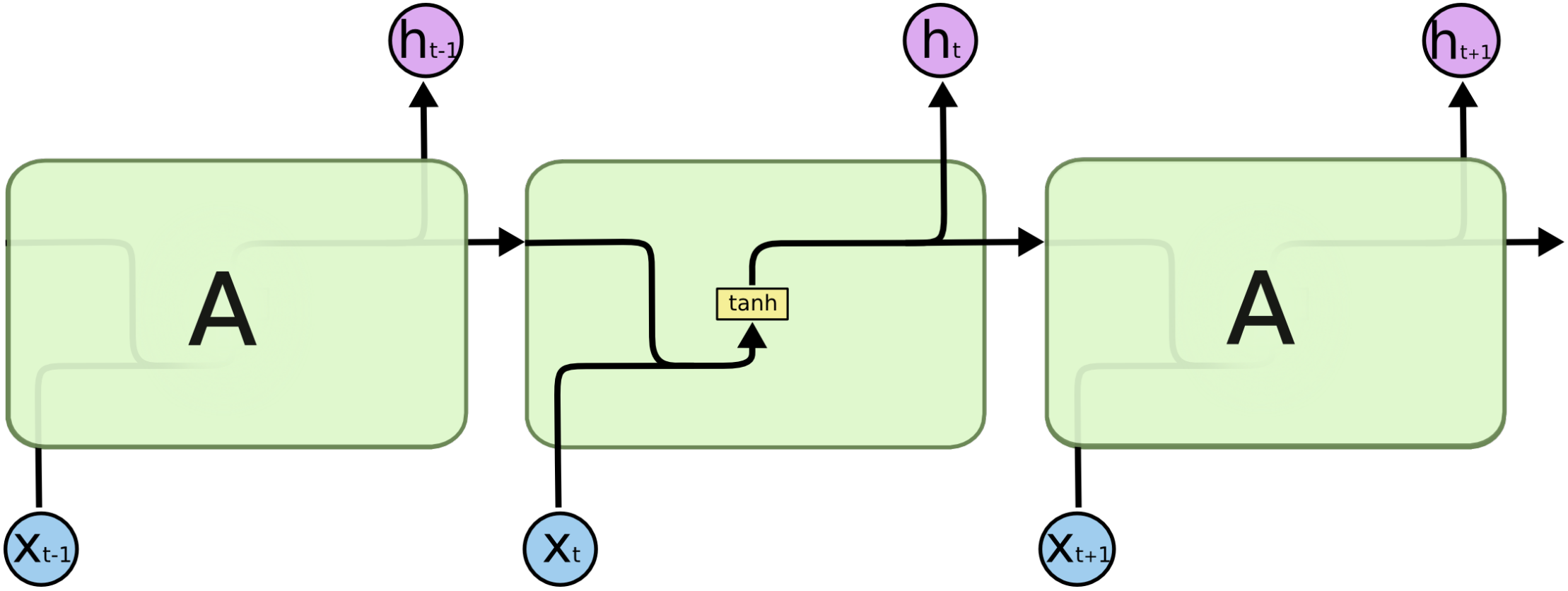

- We alternate between the following two equations:

$$\overline{h^{(t)}} = \overline{z^{(t+1)}}\, w$$
$$\overline{z^{(t)}} = \overline{h^{(t)}}\, \phi'(z^{(t)})$$

$$\overline{h^{(1)}} = w^{T-1}\phi'(z^{(2)}) \cdots \phi'(z^{(T)})\overline{h^{(T)}}$$
$$= \frac{\partial h^{(T)}}{\partial h^{(1)}}\overline{h^{(T)}}$$

- Lets assume that activation is linear, then $\frac{\partial h^{(T)}}{\partial h^{(1)}} = w^{T-1}$

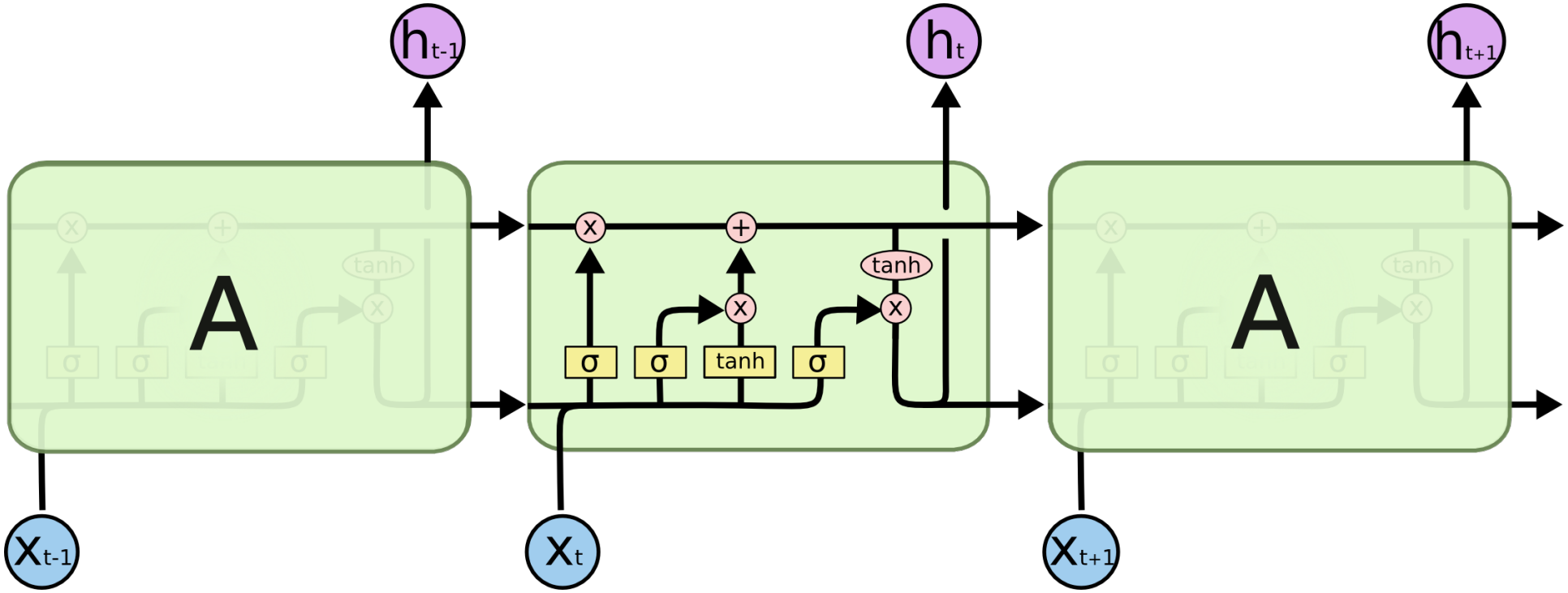
# Long short-term memory: problems with RNN




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

# Long short-term memory

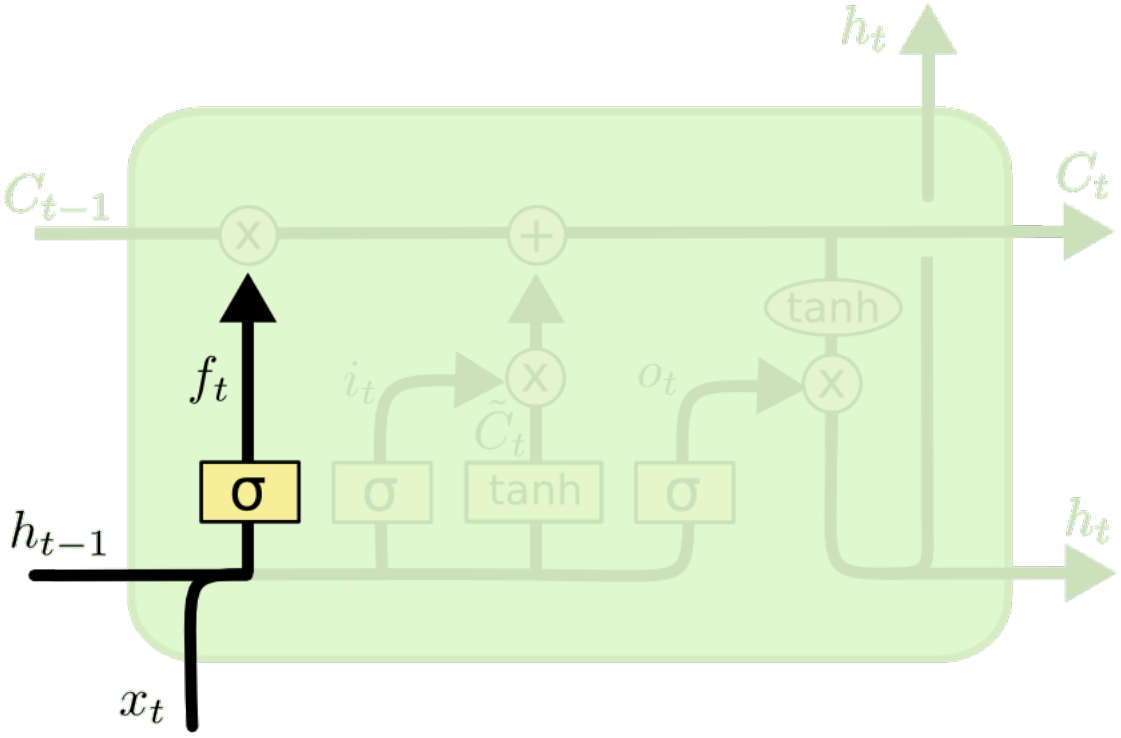
- There is a memory cell *C* that retains information through time




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

# Long short-term memory: the core idea

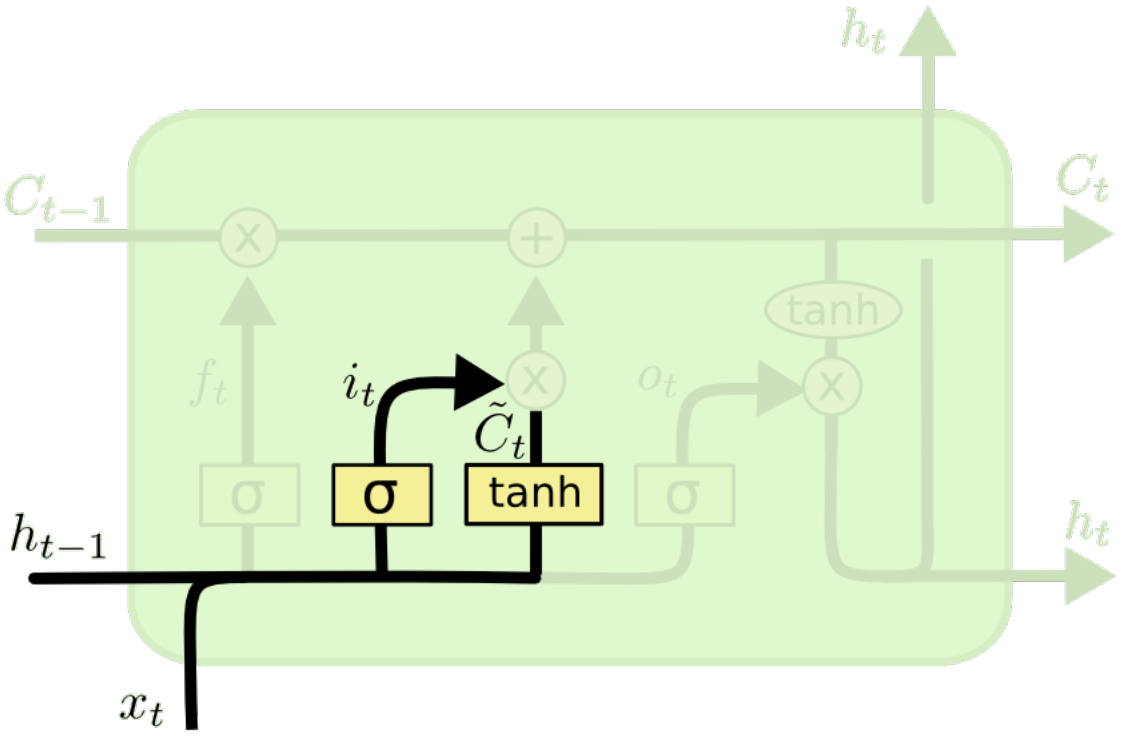
- We selectively choose to *forget* information




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

# Long short-term memory: the core idea

- We selectively choose to *input* new information

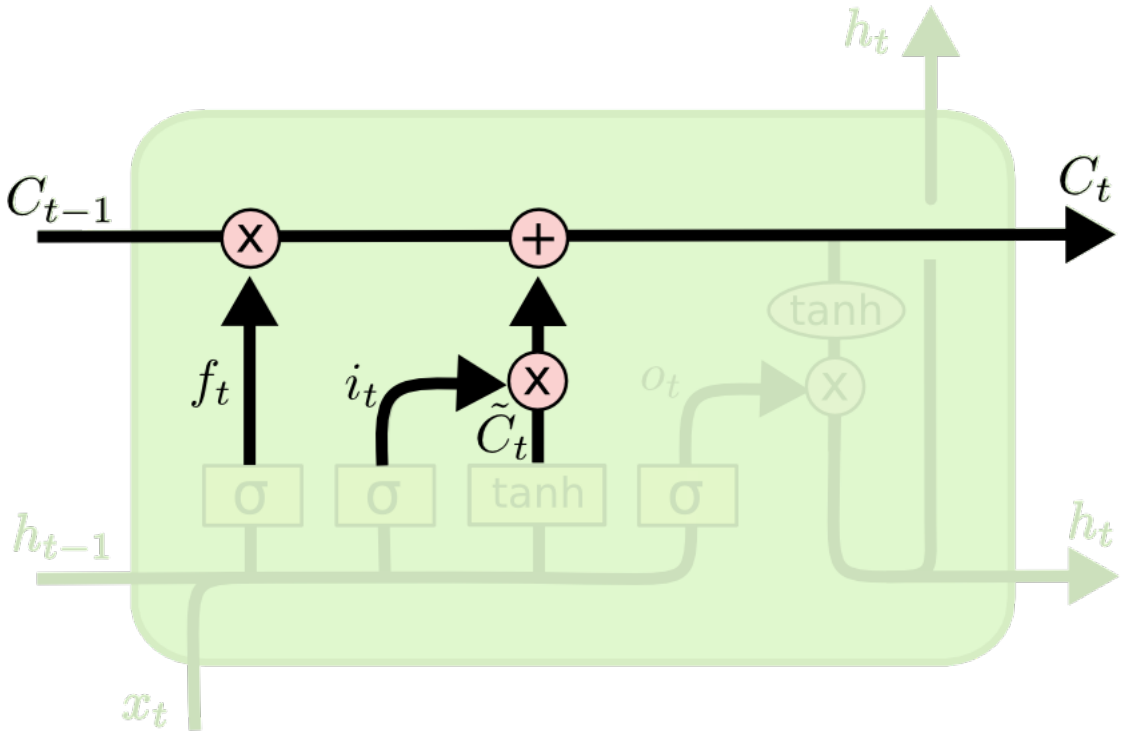



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

# Long short-term memory: the core idea

- The new cell will be a combination of what we retain from the past cell and what we input into the cell




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

# Applications of RNN

- Language modeling
- Time series analysis
- Automatic caption generation from images
- Automatic text generation from text prompt

# Citation worthiness

Scientometrics (2020) 124:399–428
https://doi.org/10.1007/s11192-020-03421-9

Check for updates

## Modeling citation worthiness by using attention-based bidirectional long short-term memory networks and interpretable models

Tong Zeng[1,2] · Daniel E. Acuna[2]

Received: 31 October 2019 / Published online: 28 March 2020

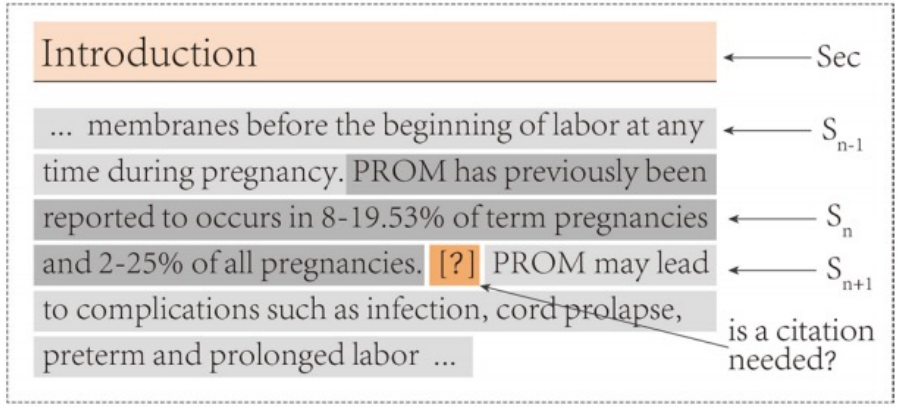




**Fig. 1** Citation worthiness prediction problem. For a given sentence ($S_n$), the goal of the task is to predict whether it needs a citation. The prediction task may use the section, the previous and next sentences (i.e., $S_{n-1}$ and $S_{n+1}$) for such prediction

# Data

**Table 2** Characteristics of the ACL-ARC dataset, whole PMOA-CITE dataset and a sample of PMOA-CITE which contains one million sentences

| Items | ACL-ARC | PMOA-CITE | PMOA-CITE sample |
|---|---|---|---|
| Articles | N/A | 2,075,208 | 6,754 |
| Sections | N/A | 9,903,173 | 32,198 |
| Paragraphs | N/A | 62,351,079 | 202,047 |
| Sentences | 1,228,052 | 309,407,532 | 1,008,042 |
| Sentences without citations | 1142275 | 249,138,591 | 811,659 |
| Sentences with citations | 85777 | 60,268,941 | 196,383 |
| Average characters per sentence | 131 | 132 | 132 |
| Average words per sentence | 22 | 20 | 20 |

# Architecture

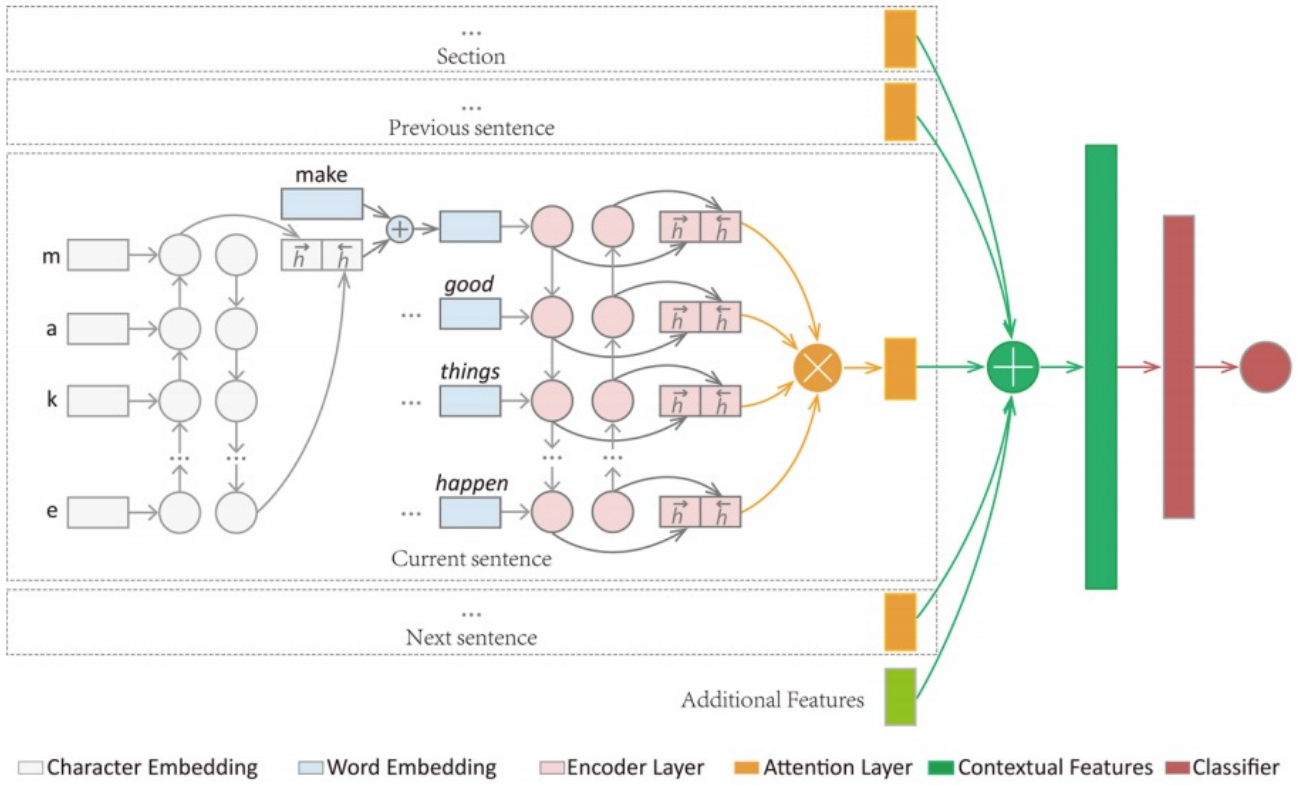



**Fig. 4** The architecture of the proposed attention-based BiLSTM neural network

# Performance

**Table 4** The performance of models on the PMOAS dataset. The hyper-parameters of our models are chosen on the validation set and the performances reported are based on a hold-out testing set

| Model | Precision | Recall | $F_1$ |
|---|---|---|---|
| Att-BiLSTM$_{sdp}$ | 0.900 | 0.764 | 0.827 |
| Att-BiLSTM$_{dp}$ | 0.886 | 0.788 | 0.834 |
| Att-BiLSTM$_{cos}$ | 0.883 | 0.795 | 0.837 |
| Contextual-Att-BiLSTM$_{sdp}$ | 0.907 | 0.797 | 0.848 |
| Contextual-Att-BiLSTM$_{dp}$ | 0.908 | 0.807 | 0.854 |
| Contextual-Att-BiLSTM$_{cos}$ | 0.907 | 0.811 | **0.856** |

In bold, we highlight the best performance

# Interpreting results with logistic regression

| Term | Feature importance | Sign of influence |
|---|---|---|
| Results | 0.0174270 | – |
| Methods | 0.0128930 | – |
| Materials | 0.0059190 | – |
| Case | 0.0024290 | – |
| Report | 0.0014750 | – |
| Experimental | 0.0010170 | – |
| Authors | 0.0008160 | – |
| Conclusion | 0.0007460 | – |
| Presentation | 0.0006070 | – |
| Contributions | 0.0006050 | – |

The plus sign (+) means the feature has a positive influence on the citation worthiness. The minus sign (−) means the feature has a negative influence on the citation worthiness

**Table 9** Term (uni-gram or bi-gram) importance of the section type

| Term | Feature importance | Sign of influence |
|---|---|---|
| Introduction | 0.027079 | + |
| Intro | 0.015247 | + |
| Background | 0.008828 | + |
| Discussion | 0.003698 | + |
| Cancer | 0.00155 | + |
| Mechanisms | 0.001328 | + |
| Cells | 0.000695 | + |
| Role | 0.000562 | + |
| Cell | 0.000521 | + |
| Receptors | 0.000379 | + |

https://cite-worthiness.scienceofscience.org/#/

# New developments

- Transformers and attention
  - No need to sequentially predict – all can be done in one step
  - ”attention” mechanism idea taken from the brain
- https://deepai.org/machine-learning-model/text-generator