Artificial Neural Networks and Deep Learning

Week 5

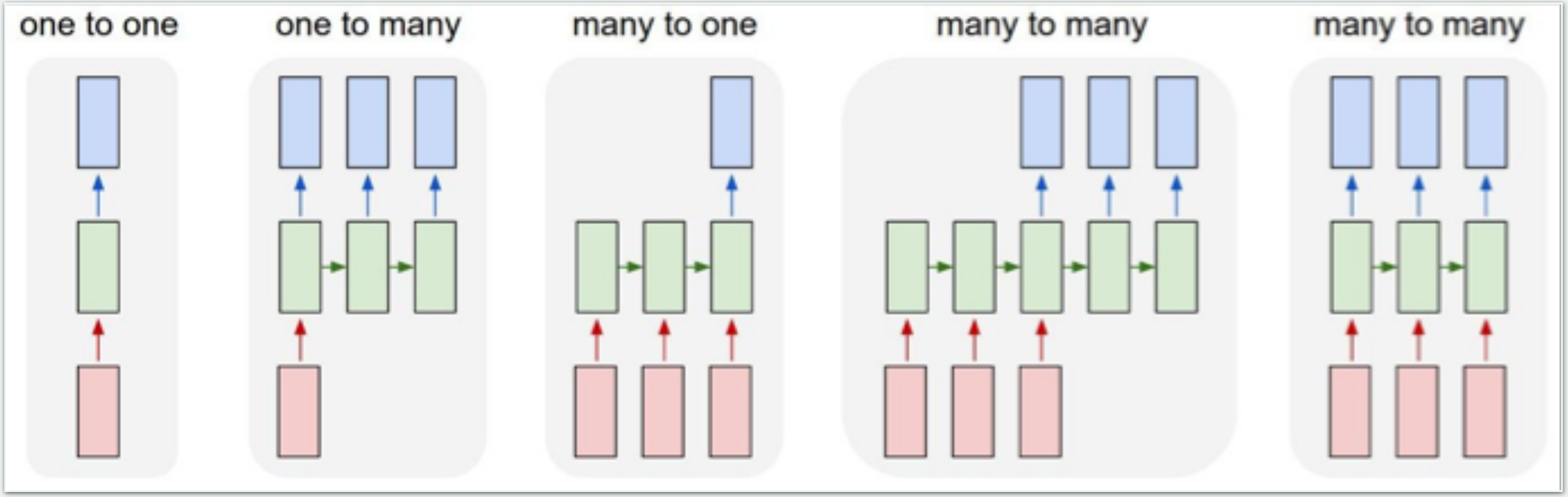
Recurrent neural networks

Recurrent Neural Networks

THE neural network architecture to use for sequential data

Recurrent neural networks

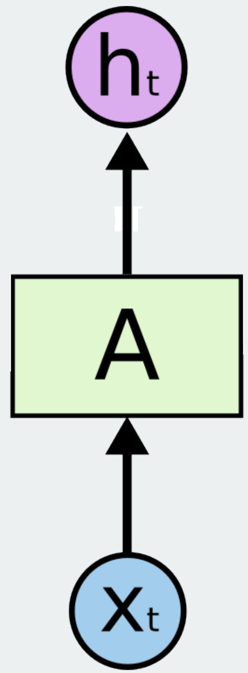
> Ways to process sequential data



http://cs231n.stanford.edu/syllabus.html

The **problem** with **all** feed forward neural networks

- Input and output must be of hard-assigned dimensions
- The network makes a fixed number of computations
- If input is sequence-like (video, sound, etc.) the network is ignorant to the order of samples

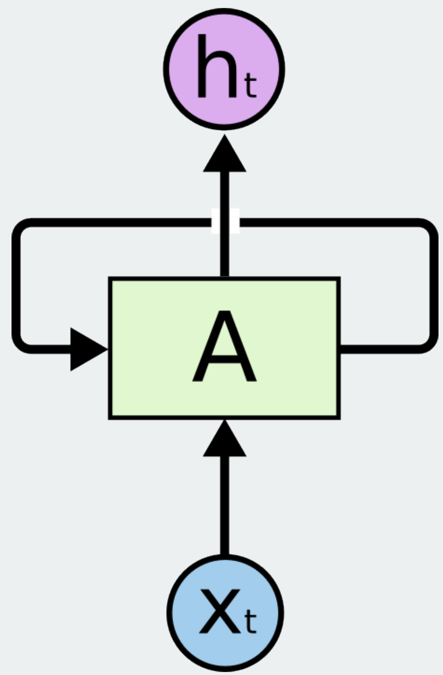


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The **problem** with **all** feed forward neural networks

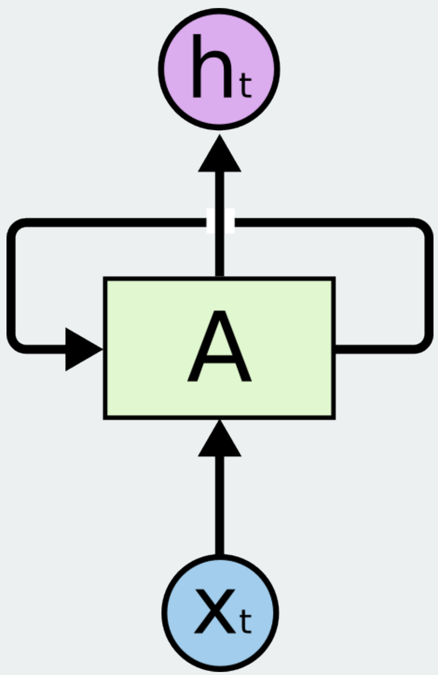
> **Solution:** *Recurrence!*

- Input and output must be of hard-assigned dimensions
- The network makes a fixed number of computations
- If input is sequence-like (video, sound, etc.) the network is ignorant to the order of samples



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

Recurrent neural networks
> Fundamental idea



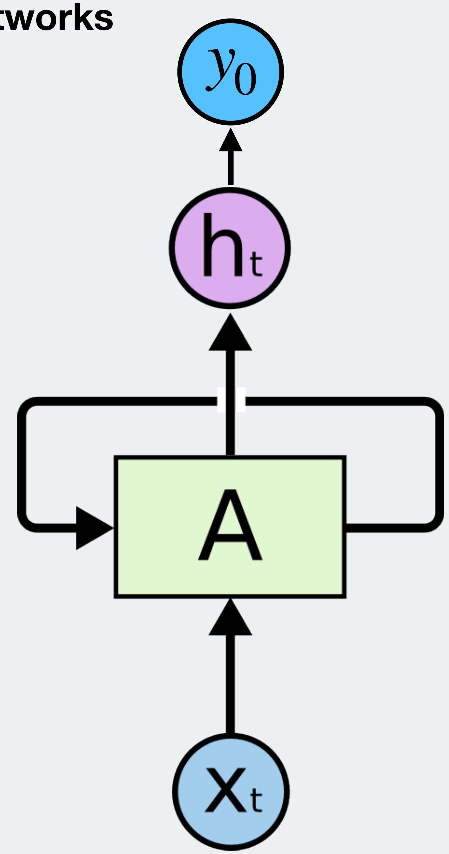
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: W is the same in each iteration

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

Recurrent neural networks

> Fundamental idea



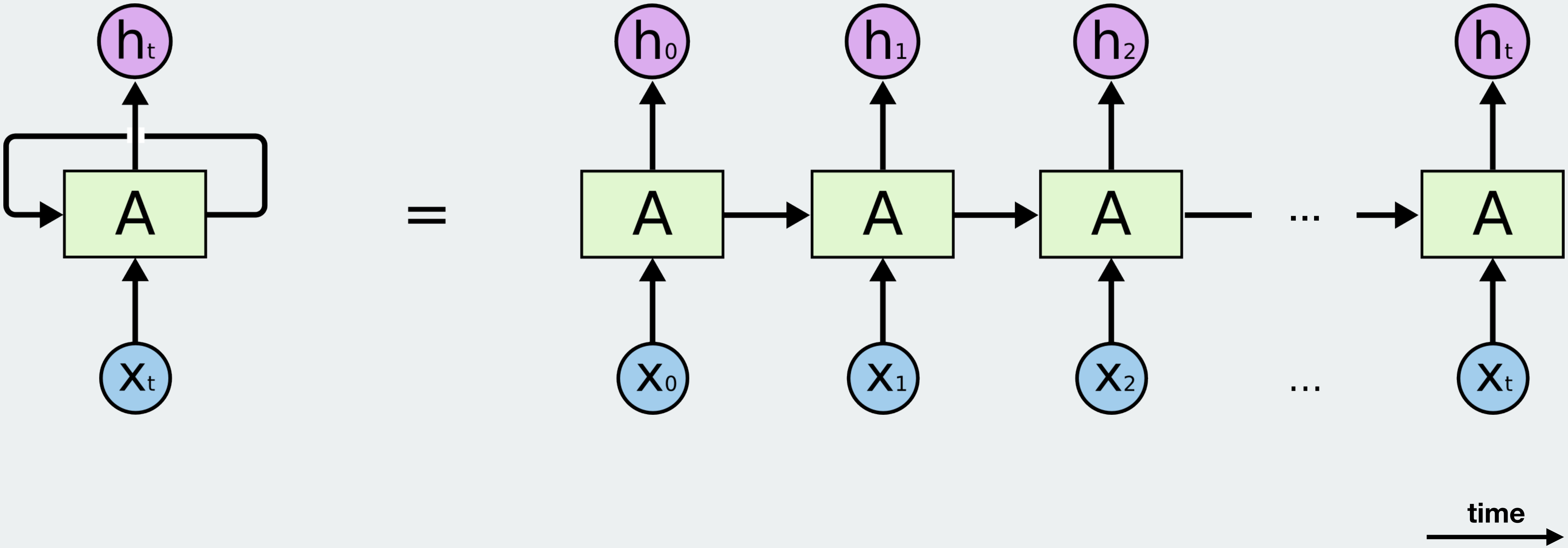
$$h_t = f_W(h_{t-1}, x_t)$$

$$y_t = W_{hy} h_t$$

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

Recurrent neural networks

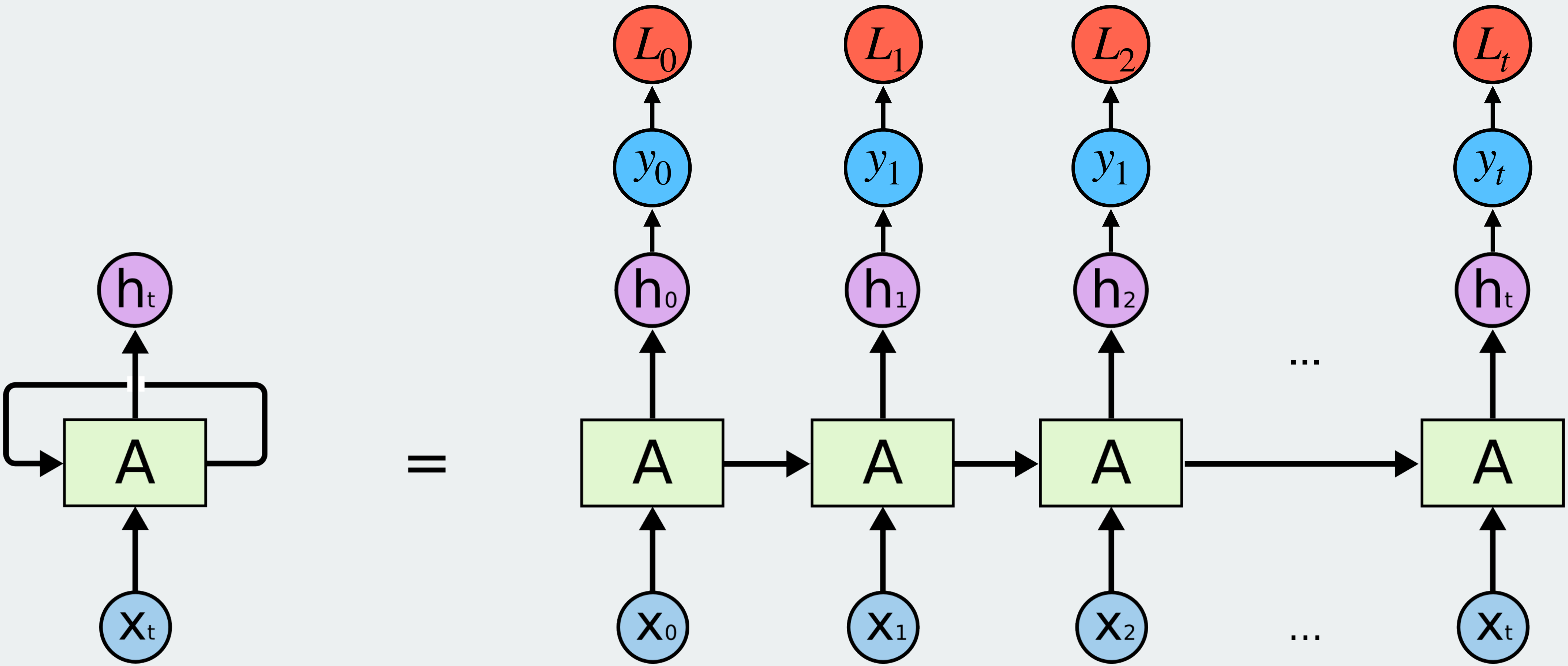
> Unrolled in time



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

Recurrent neural networks

> Backpropagation

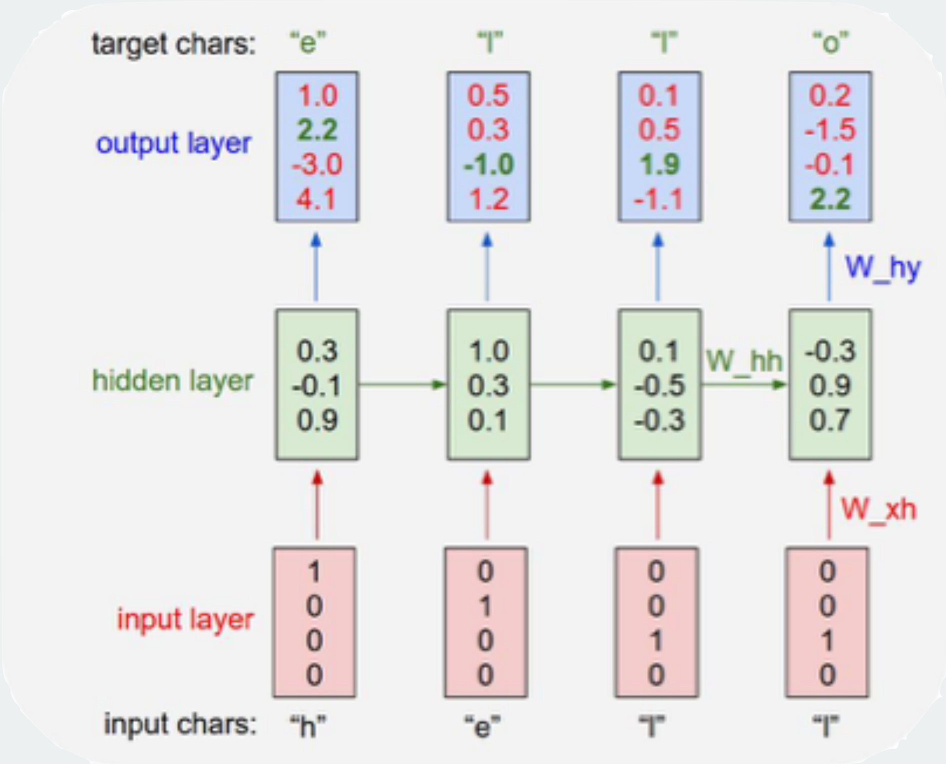


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent neural networks

> Example: predicting next character

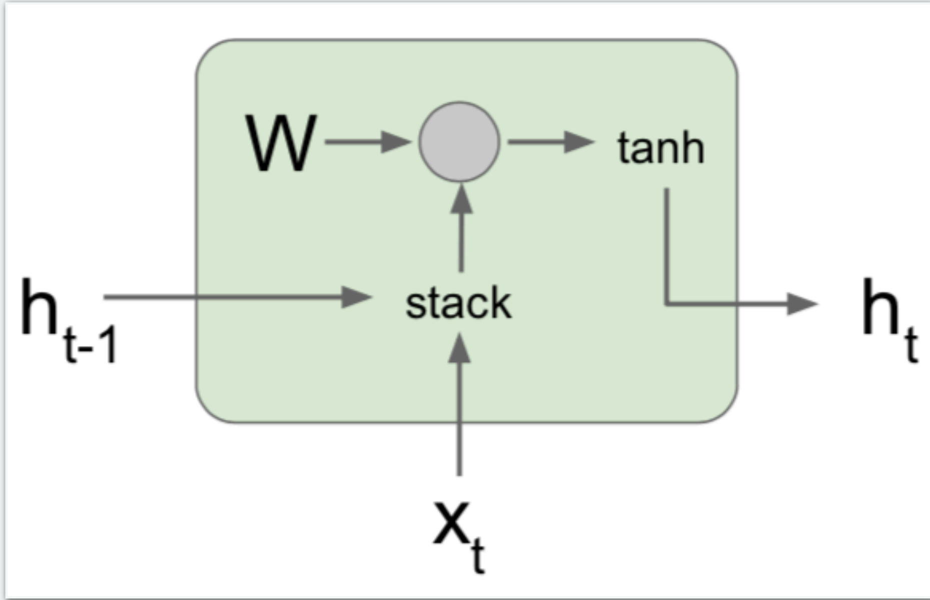
training sequence**: “hello”**



http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN, architecture



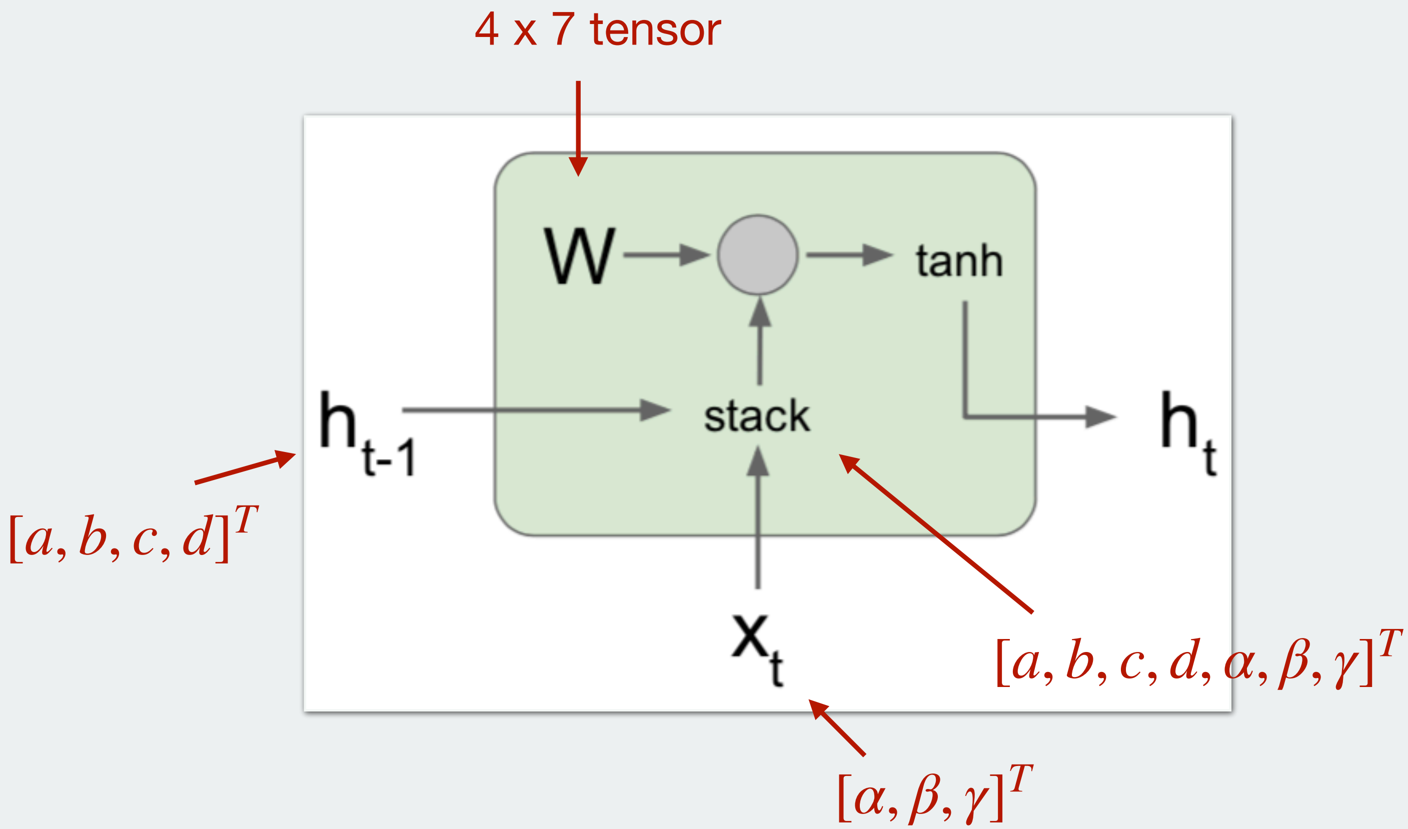
http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN, architecture

4: because dotting (h, t) onto it should result in a new vector with 4 elements

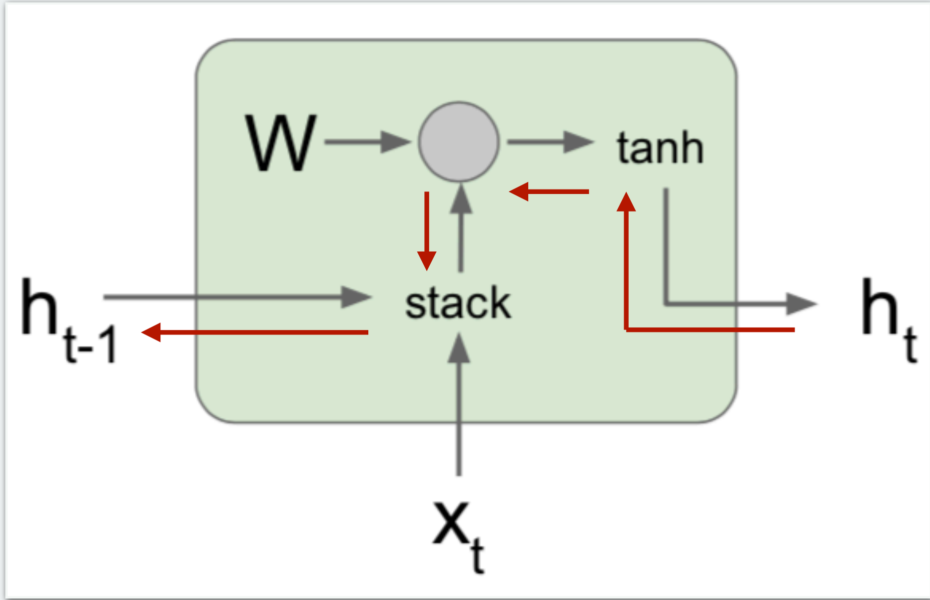
7: because the (h, t) vector we are dotting onto it has 7 elements in it



http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

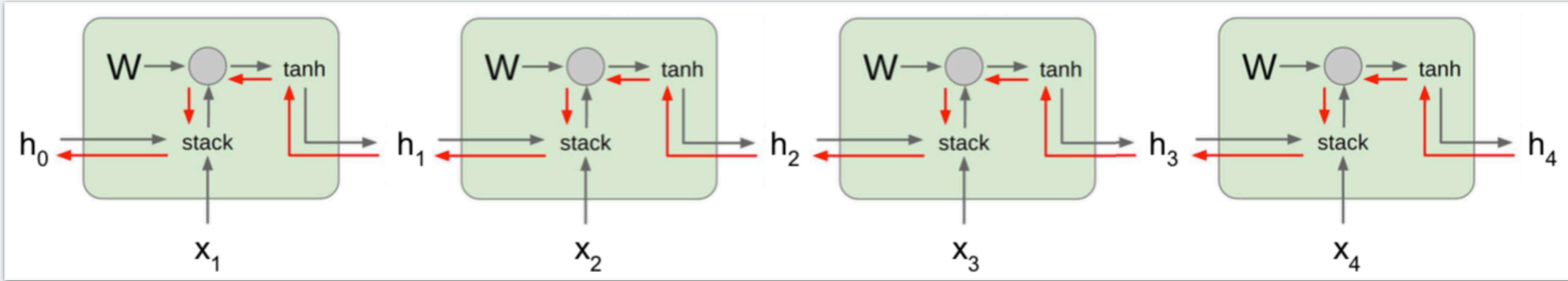
> Vanilla RNN, backpropagation



http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN, backpropagation



Problem: Repeated multiplications by **W** during backpropagation

Leads to: Exploding/vanishing gradients

Solution: Gradient clipping (solves exploding gradients), or **change architecture**

http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN vs. LSTM

Vanilla RNN

$$h_t = \tanh\left(W\begin{bmatrix}h_{t-1}\\ x_t\end{bmatrix}\right)$$

Long Short Term Memory (LSTM)

$$\begin{bmatrix}i\\ f\\ o\\ g\end{bmatrix} = \begin{bmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{bmatrix} W\begin{bmatrix}h_{t-1}\\ x_t\end{bmatrix}$$

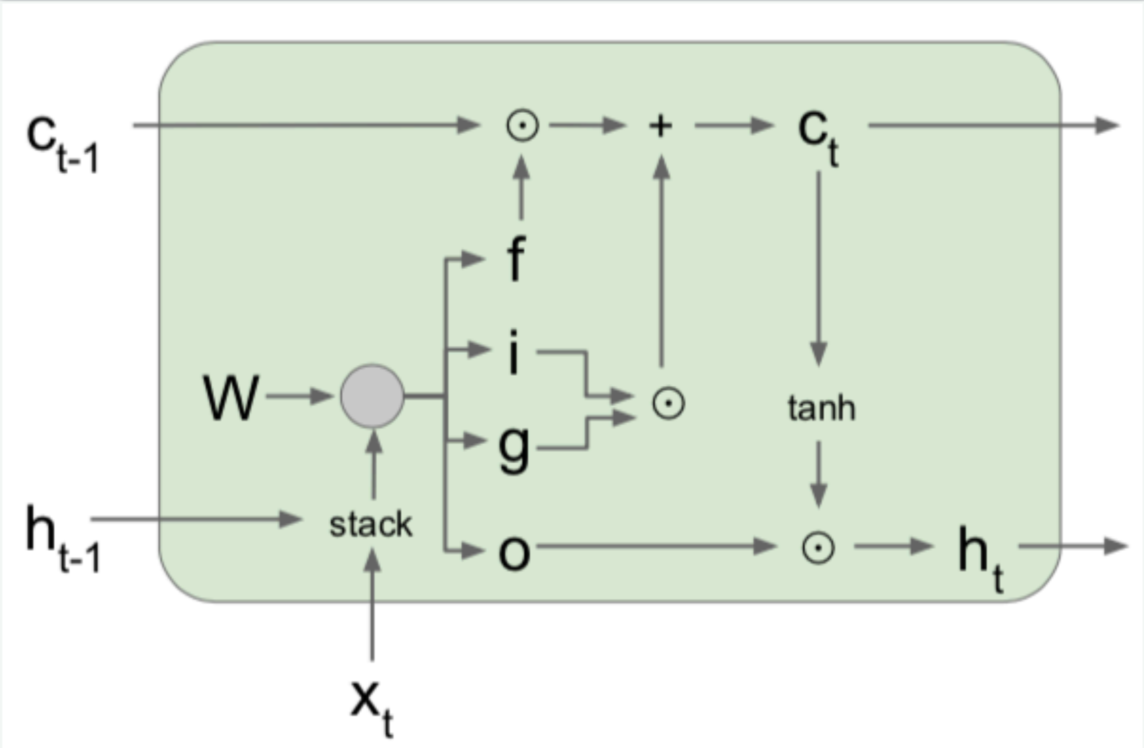
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN vs. LSTM



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

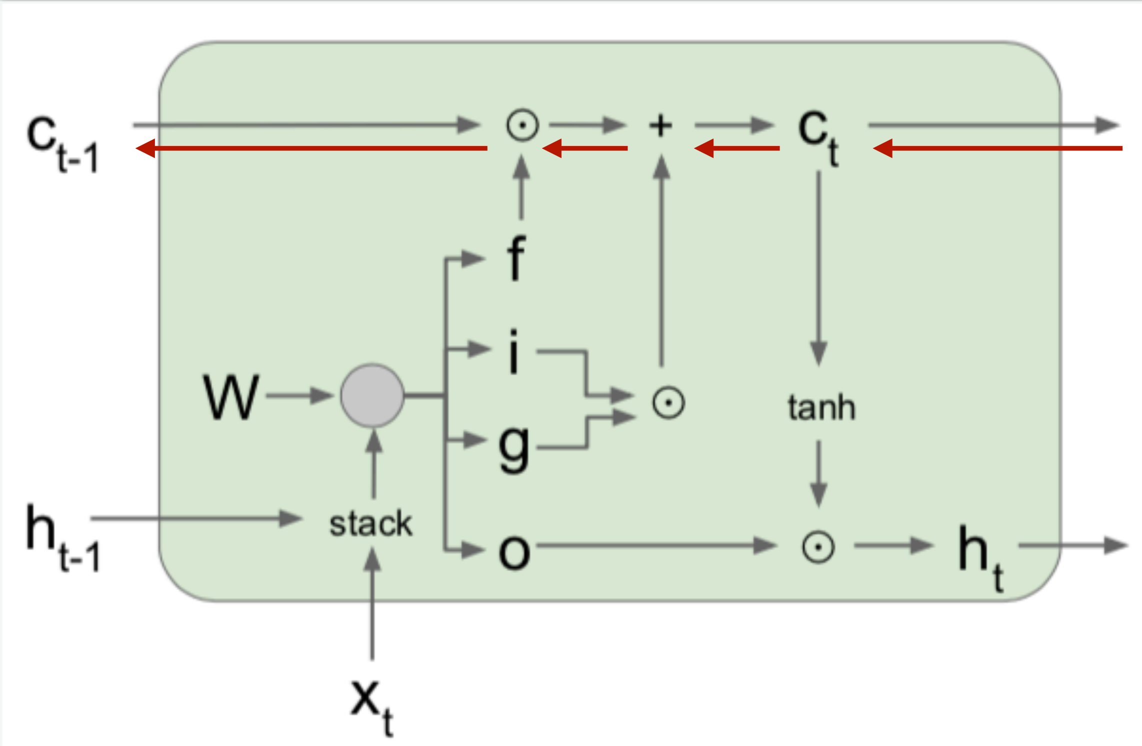
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks
> Vanilla RNN vs. LSTM

"Gradient highway": solves vanishing/exploding gradient problems



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

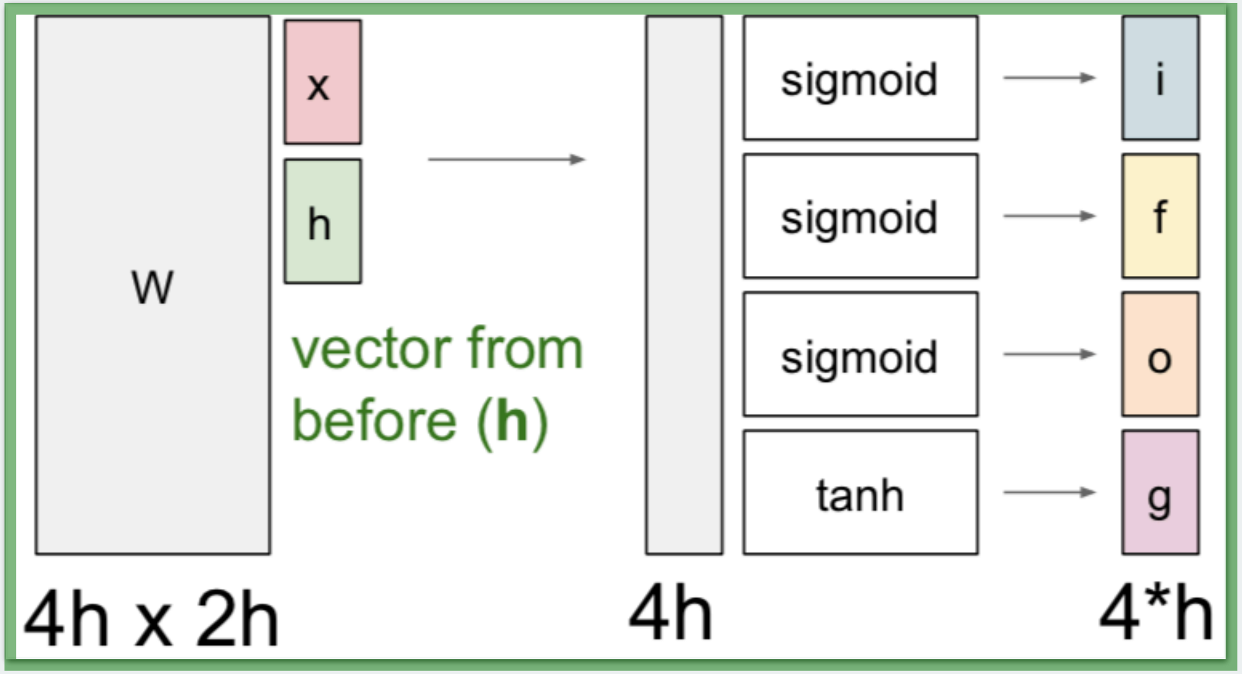
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent neural networks

> Vanilla RNN vs. LSTM



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html