Lecture 24: Long Short Term Memory

COMP 411, Fall 2021

Victoria Manfredi

These slides are based on figures and info from Andrej Karpathy's blog, slides created by Justin Johnson (U of Michigan) and Geoffrey Hinton (U of Toronto), and the book Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

Today’s Topics

Long Short Term Memory

- Overview
- Gradient flow

Long short term memory

OVERVIEW

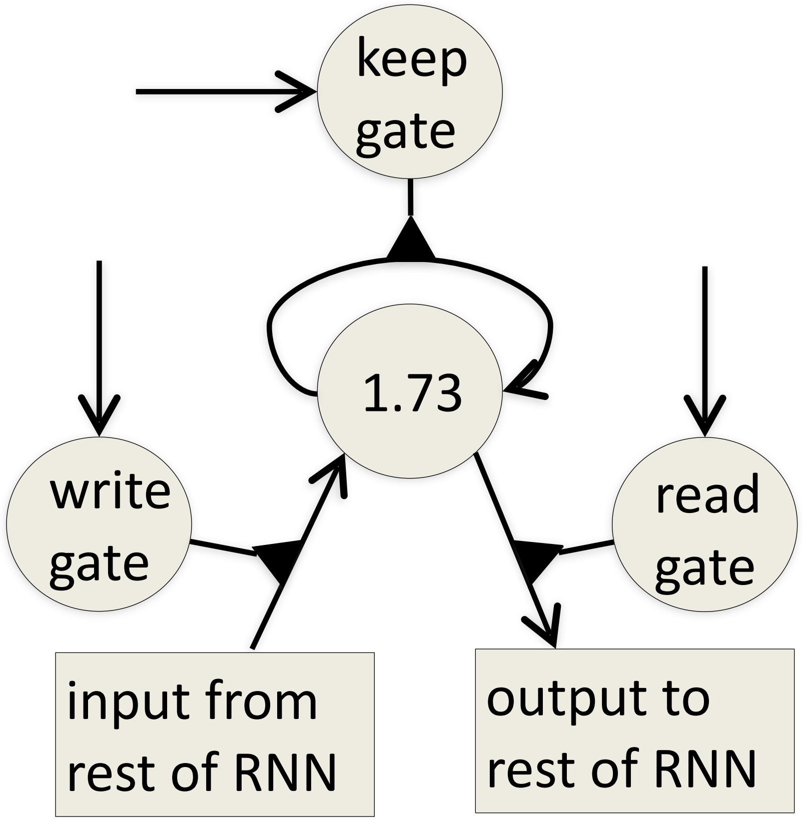
Long Short Term Memory (LSTM)

- Hochreiter & Schmidhuber (1997) solved the problem of getting an RNN to remember things for a long time (like hundreds of time steps).
- They designed a memory cell using logistic and linear units with multiplicative interactions.
- Information gets into the cell whenever its “write” gate is on.
- The information stays in the cell so long as its “keep” gate is on.
- Information can be read from the cell by turning on its “read” gate.

Implementing a memory cell in a neural network

To preserve information for a long time in the activities of an RNN, we use a circuit that implements an analog memory cell.

- A linear unit that has a self-link with a weight of 1 will maintain its state.
- Information is stored in the cell by activating its write gate.
- Information is retrieved by activating the read gate.
- We can backpropagate through this circuit because logistics have nice derivatives.



Reading cursive handwriting

This is a natural task for an RNN

- The input is a sequence of (x,y,p) coordinates of the tip of the pen, where p indicates whether the pen is up or down.

question: what is p? what does it mean whether p is up or down?

- The output is a sequence of characters.

Means space?

Graves & Schmidhuber (2009) showed that RNNs with LSTM are currently the best systems for reading cursive writing.

- They used a sequence of small images as input rather than pen coordinates.

http://www.cs.toronto.edu/~graves/handwriting.html

Long Short Term Memory

GRADIENT FLOW

Vanilla RNN

Vanilla RNN

$$\boxed{\mathbf{h}_t = \tanh\left(\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}\right)}$$

Long Short Term Memory (LSTM)

Vanilla RNN

$$\mathbf{h}_t = \tanh\left(\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix}\mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\\ \mathbf{g}\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix}\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}\odot\mathbf{c}_{t-1} + \mathbf{i}\odot\mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o}\odot\tanh(\mathbf{c}_t)$$

$\odot$ is Hadamard product: element-wise multiplication of matrices.

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997

Long Short Term Memory (LSTM)

Vanilla RNN

$$\boxed{\mathbf{h}_t = \tanh\left(\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}\right)}$$

LSTM

$$\begin{pmatrix}\mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\\ \mathbf{g}\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix}\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}\odot\mathbf{c}_{t-1} + \mathbf{i}\odot\mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o}\odot\tanh(\mathbf{c}_t)$$

Two vectors at each timestep:

Cell state $\rightarrow \mathbf{c}_t$

Hidden state $\rightarrow \mathbf{h}_t$

Long Short Term Memory (LSTM)

Vanilla RNN

$$\mathbf{h}_t = \tanh\left(\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix}\mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\\ \mathbf{g}\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix}\mathbf{W}\begin{pmatrix}\mathbf{h}_{t-1}\\ \mathbf{x}_t\end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}\odot\mathbf{c}_{t-1} + \mathbf{i}\odot\mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o}\odot\tanh(\mathbf{c}_t)$$

Compute four **gates** at each timestep

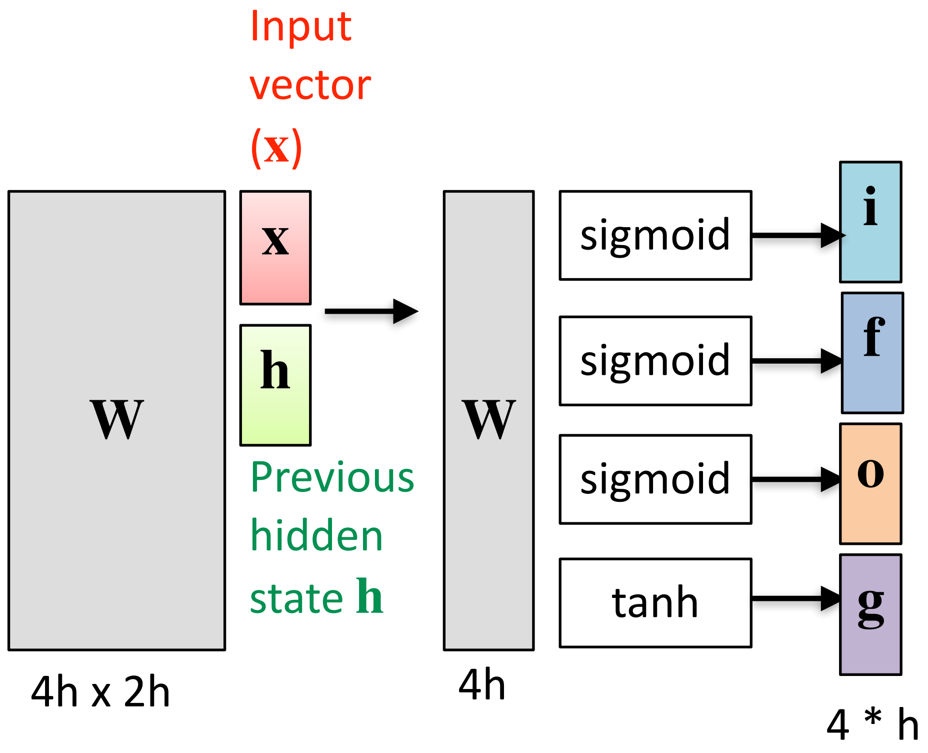
Long Short Term Memory (LSTM)

$\mathbf{i}$: **Input gate**, whether to write to cell
$\mathbf{f}$: **Forget gate**, whether to erase cell
$\mathbf{o}$: **Output gate**, how much to reveal cell
$\mathbf{g}$: **Gate gate (?)**, how much to write to cell

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$



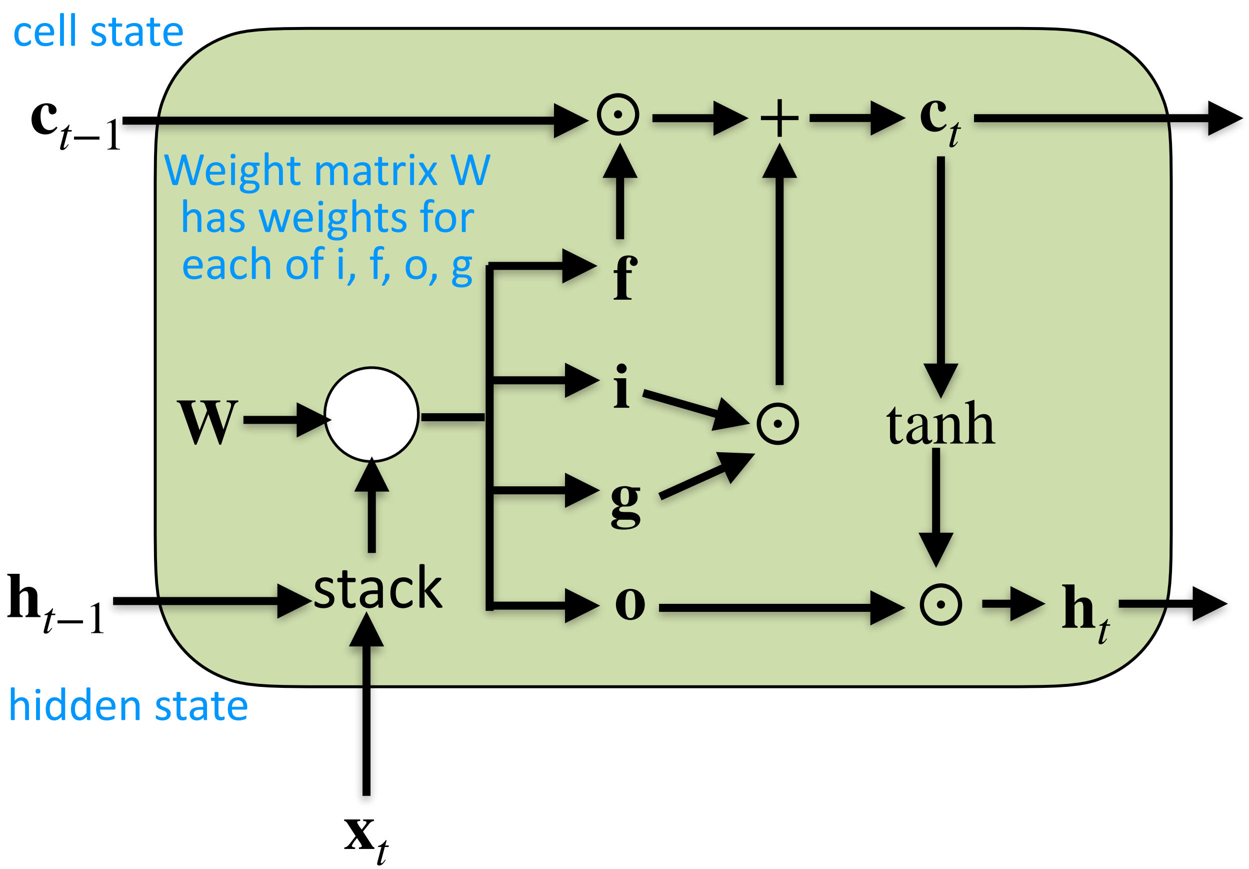
sigmoid σ: if 0 take none of input if 1 take all

Long Short Term Memory (LSTM)

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

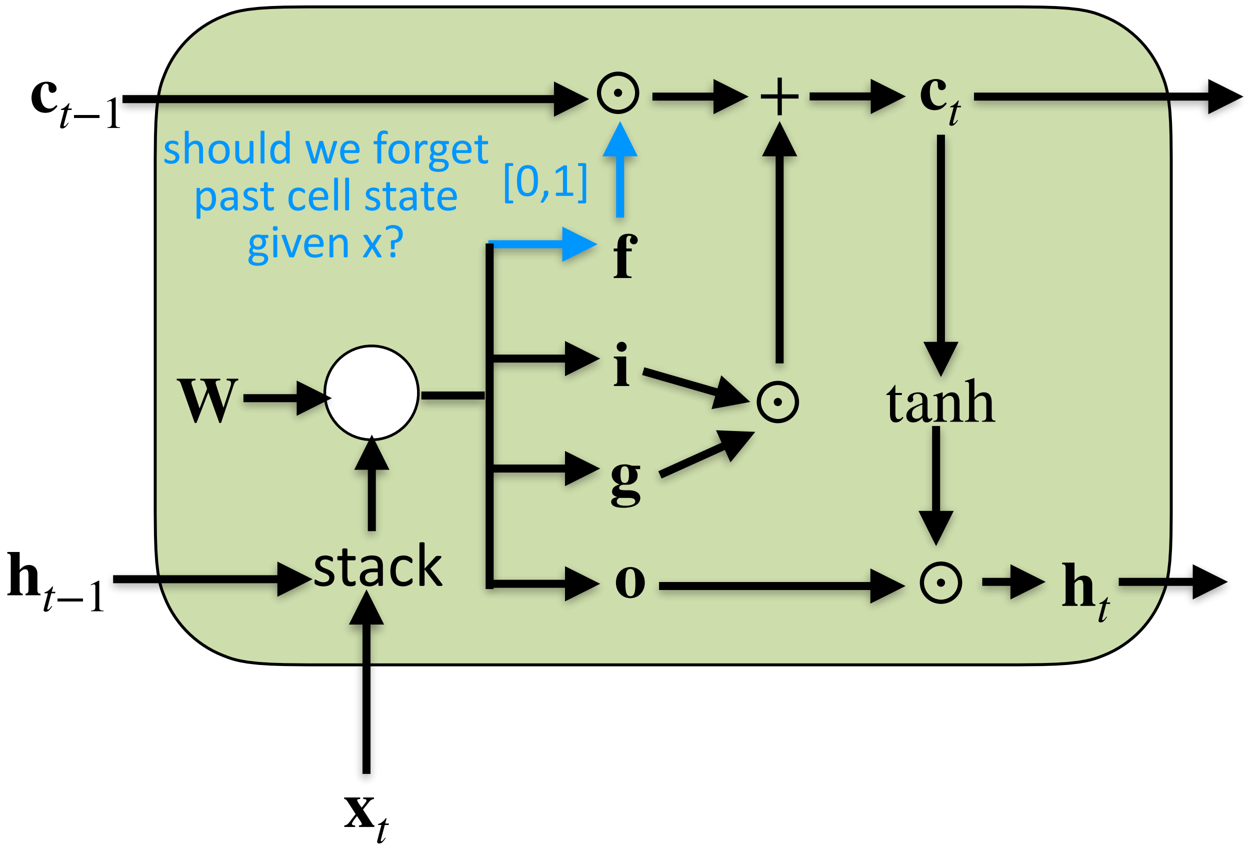


Long Short Term Memory (LSTM)

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

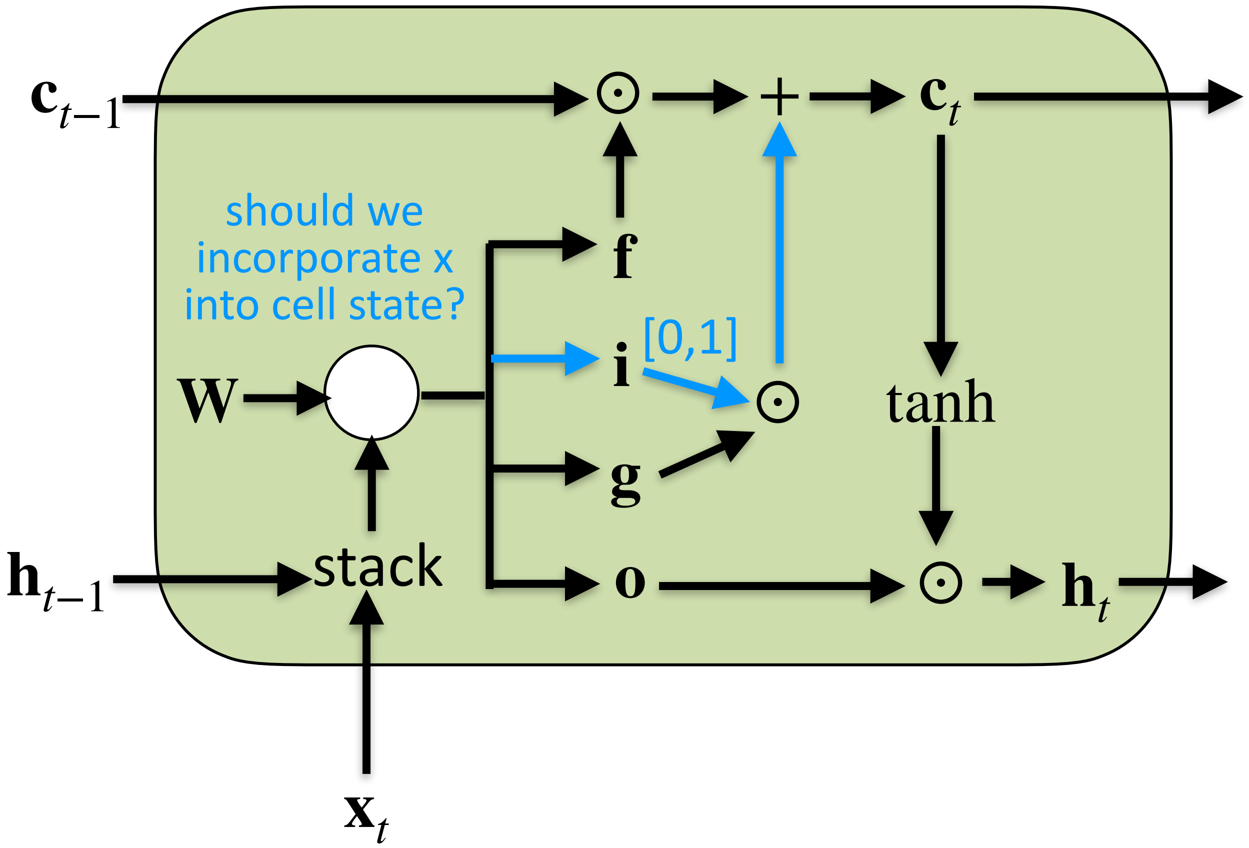


Long Short Term Memory (LSTM)

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

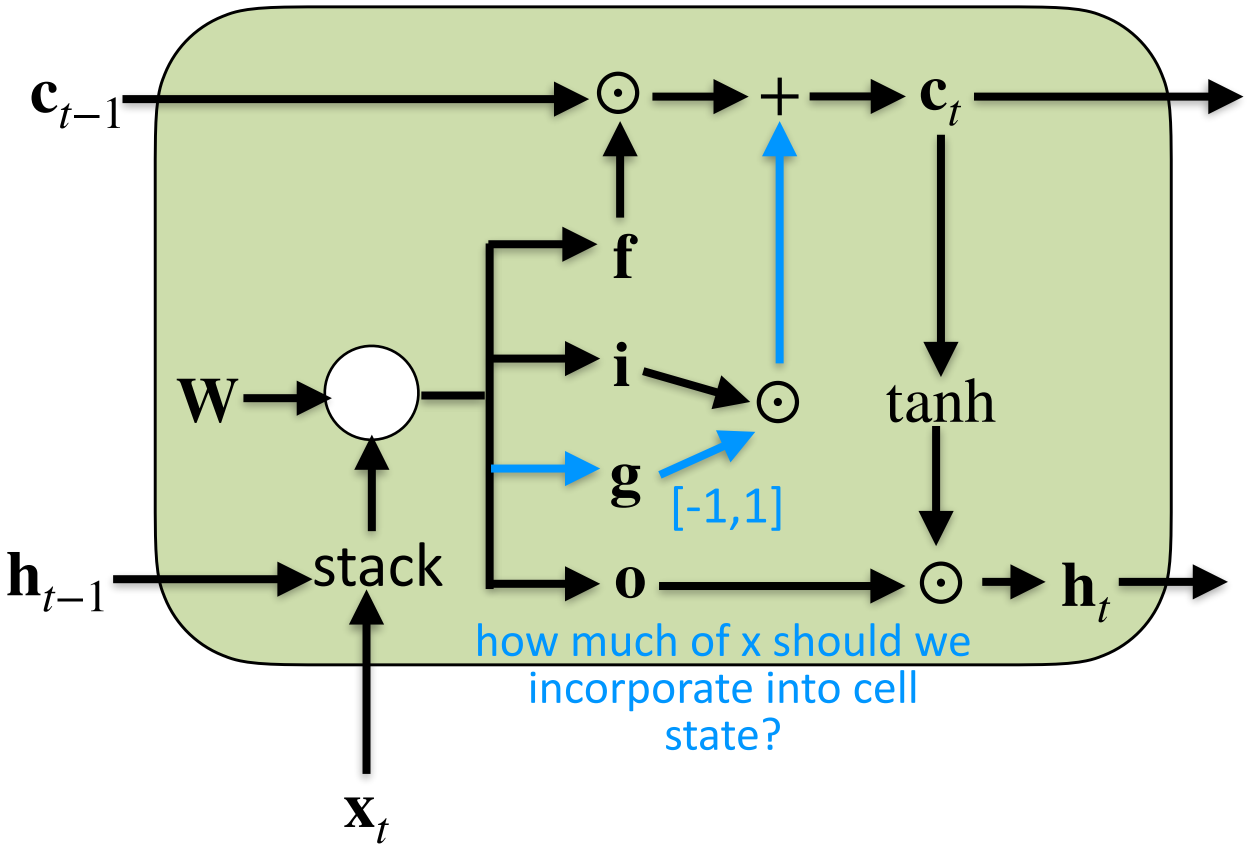


Long Short Term Memory (LSTM)

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

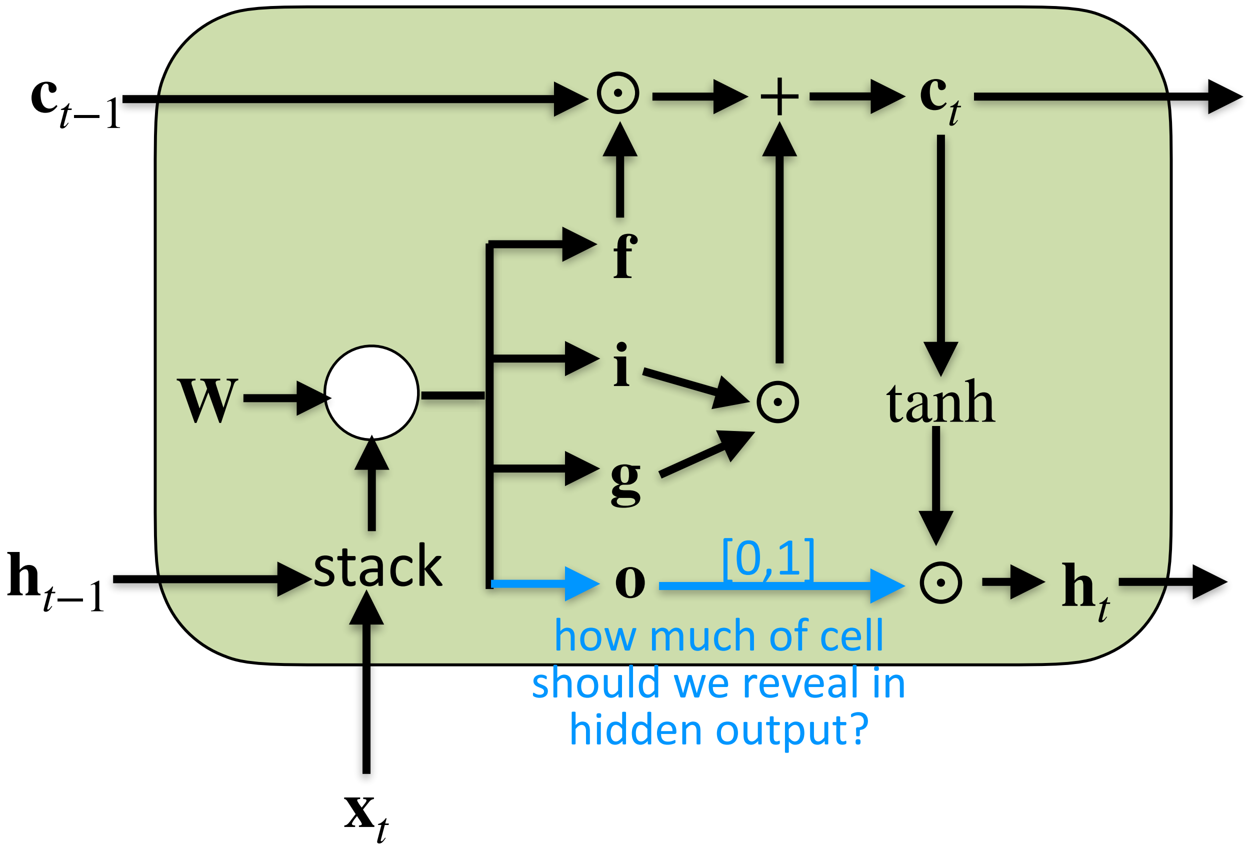


Long Short Term Memory (LSTM)

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$



Long Short Term Memory (LSTM)

Backpropagation and LSTMs:

https://christinakouridi.blog/2019/06/19/backpropagation-lstm/

https://www.kdnuggets.com/2019/05/understanding-backpropagation-applied-lstm.html

Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don’t work very well
- Common to use LSTM or GRU (Gated Recurrent Unit)
  - Additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish
  - Exploding is controlled with gradient clipping
  - Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed