CS489/698: Intro to ML

Lecture 14: RNNs

focal Systems

UNIVERSITY OF WATERLOO

Outline

- Motivation
- Intro to RNNs
- Bidirectional RNN
- LSTM
- Applications

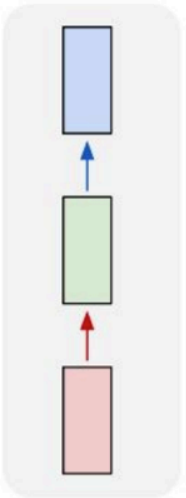
Motivation

- Introducing bias for modeling sequences
- Multiple inputs into a model

Intro to RNNs

“Vanilla” Neural Network

one to one

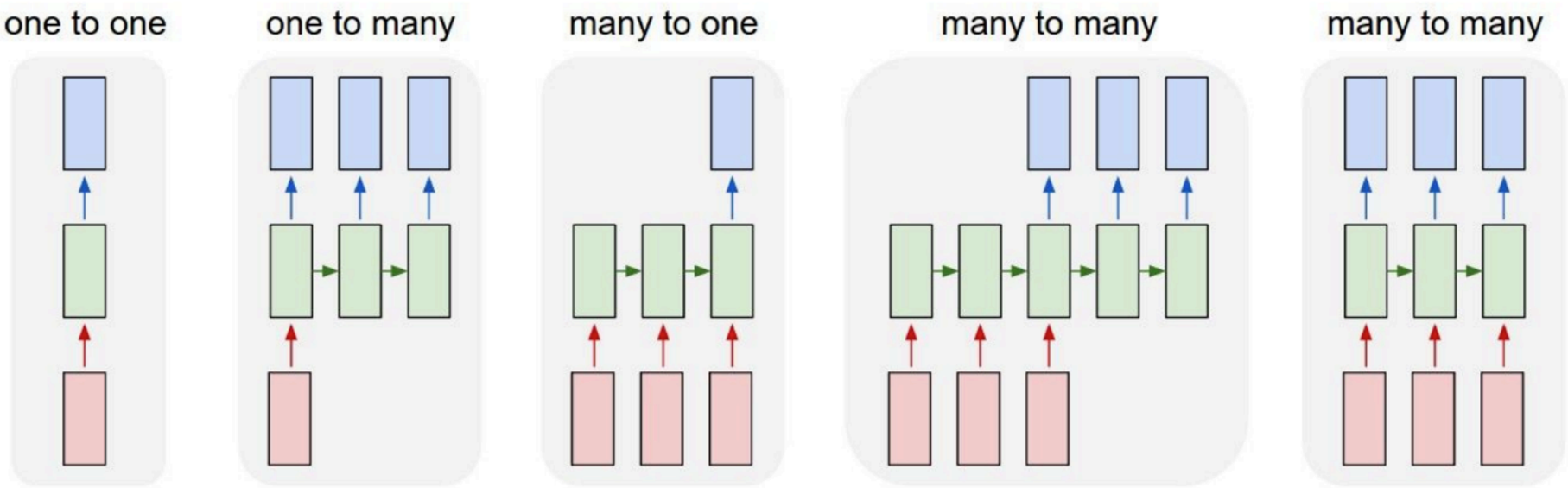


Vanilla Neural Networks

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Intro to RNNs

Recurrent Neural Networks: Process Sequences



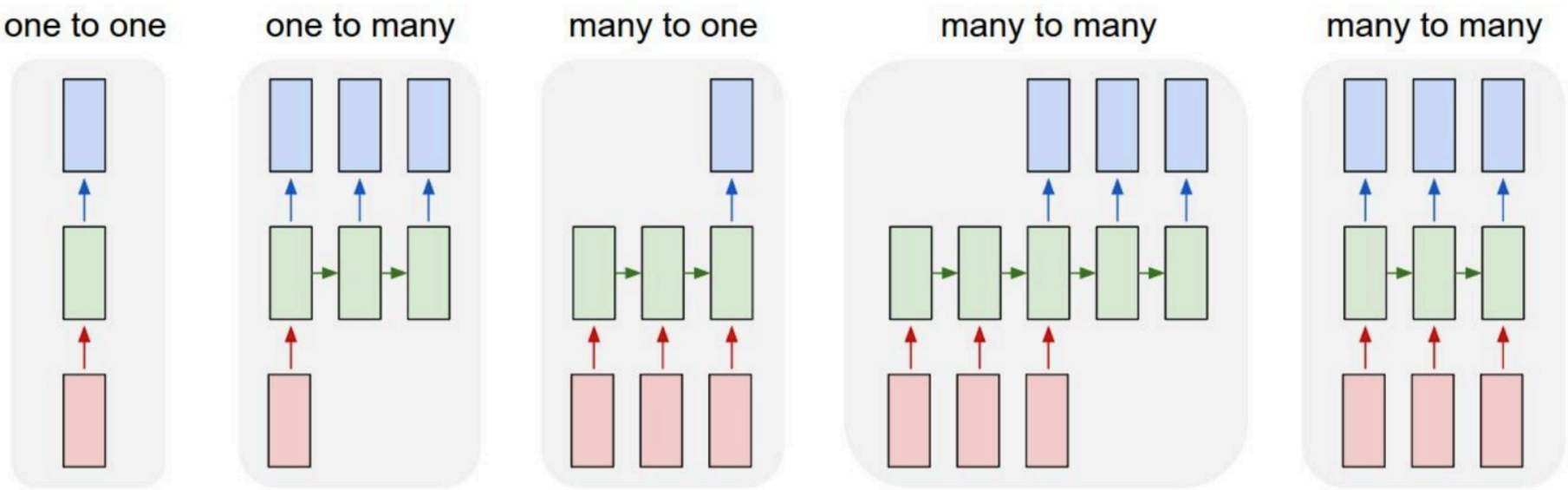
e.g. **Image Captioning**
image -> sequence of words

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Intro to RNNs

Recurrent Neural Networks: Process Sequences



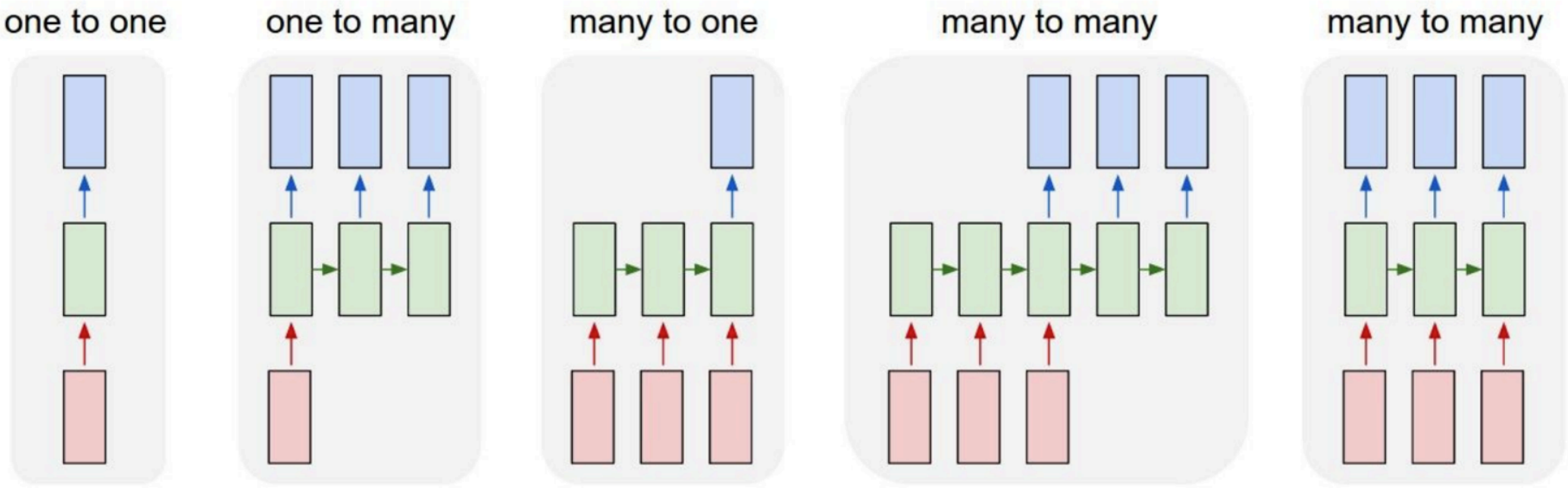
e.g. **Sentiment Classification**
sequence of words -> sentiment

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Intro to RNNs

Recurrent Neural Networks: Process Sequences



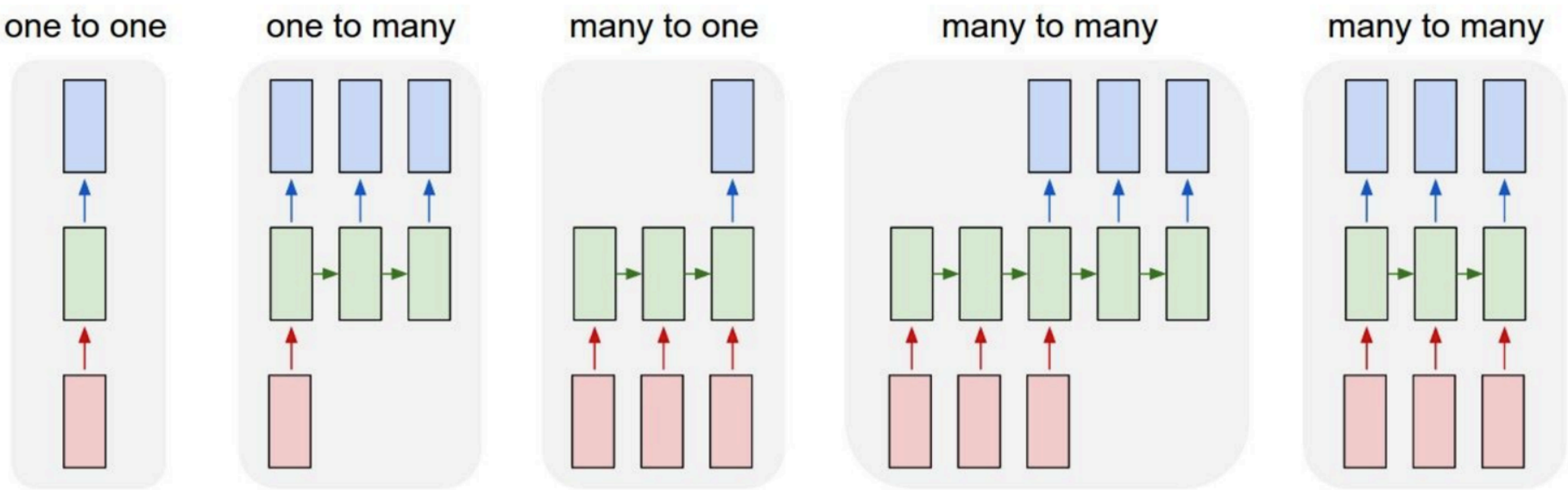
e.g. **Machine Translation**
seq of words -> seq of words

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Intro to RNNs

Recurrent Neural Networks: Process Sequences



UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Unfolding a computation Graph

Unfolding Computation Graphs

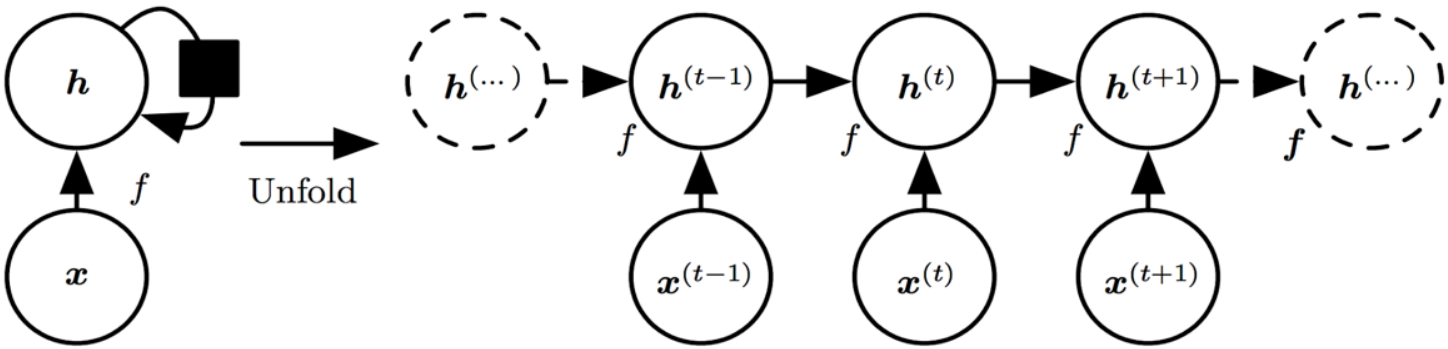
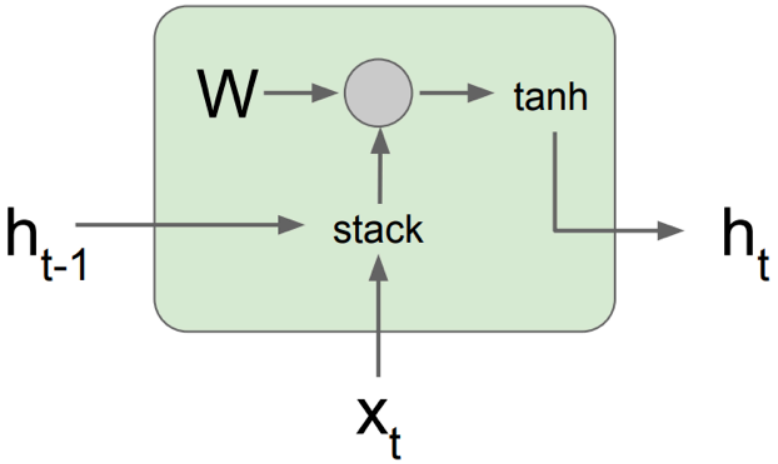


Figure 10.2

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Vanilla RNN

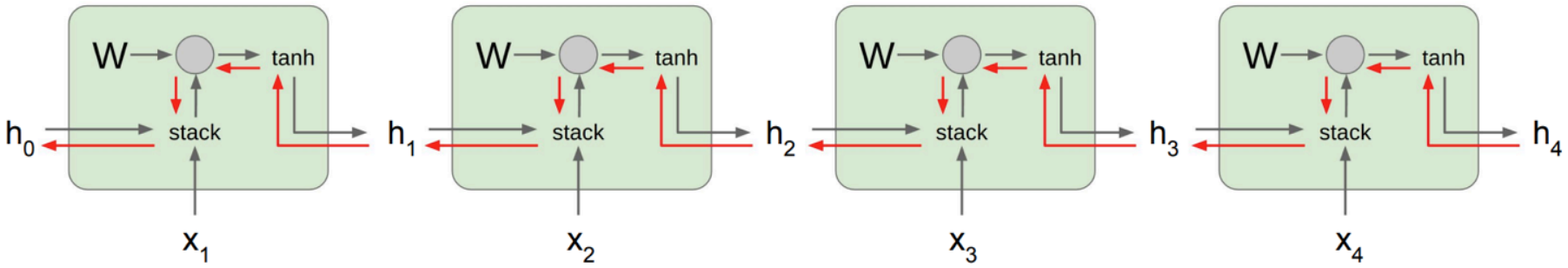


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Vanilla RNN

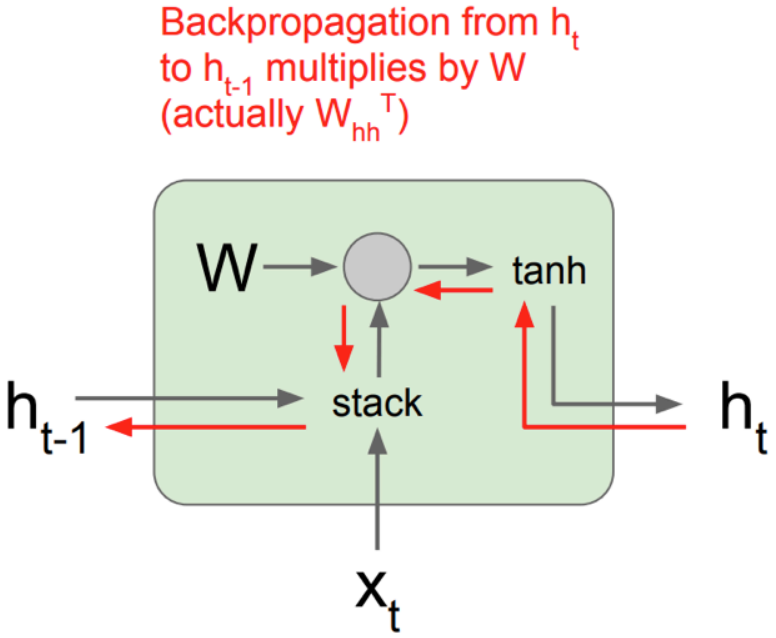


Computing gradient of h_0 involves many factors of W (and repeated tanh)

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Vanilla RNN

The gradient either vanishes or exploded



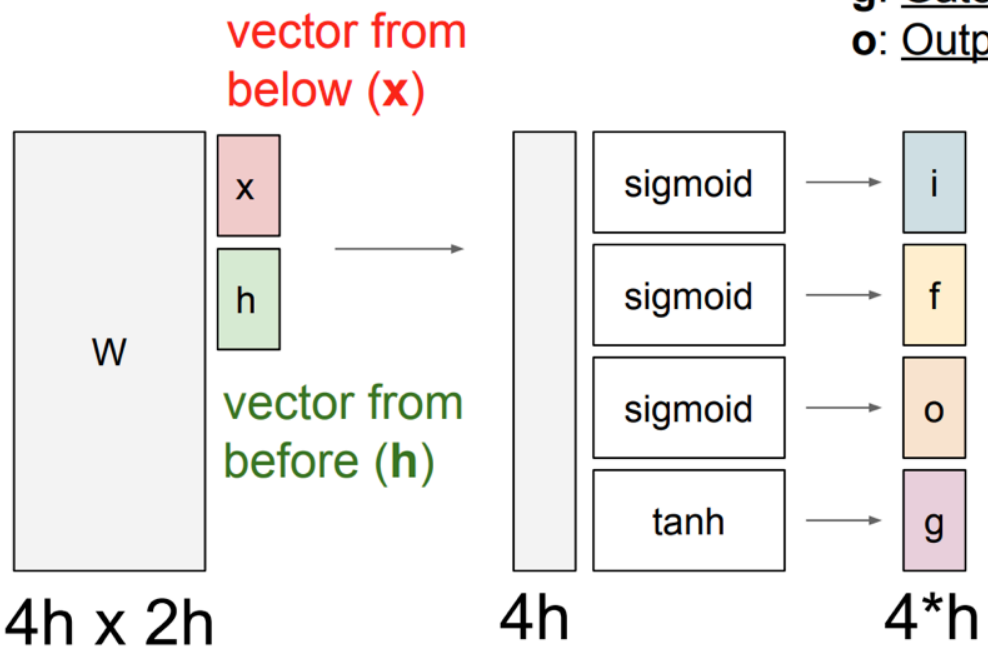
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

LSTM

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



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

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

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

LSTMs are explicitly designed to avoid the long-term dependency problem.

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Vanilla RNN vs Long Short Term Memory

Vanilla RNN

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

LSTM

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

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

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

f(forget gate): A 1 represents “completely keep this” while a 0 represents “completely get rid of this.

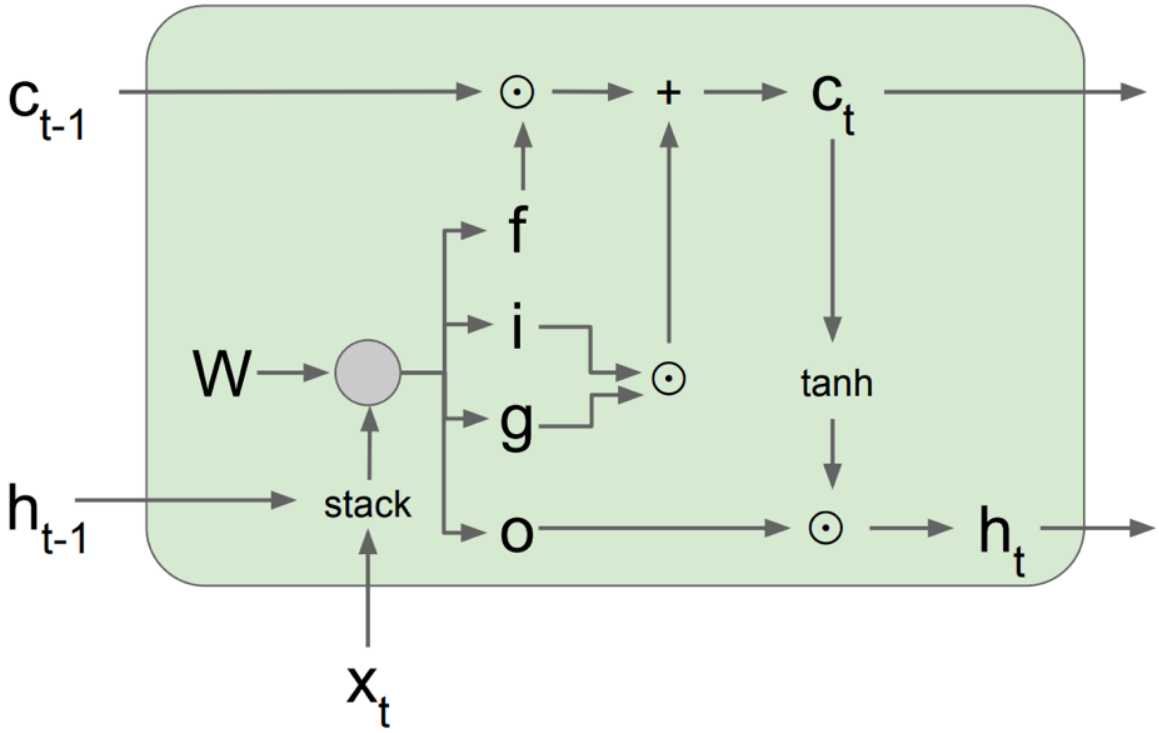
i and g gate: The next step is to decide what new information we’re going to store in the cell state. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, C̃ tC~t, that could be added to the state. In the next step, we’ll combine these two to create an update to the state

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

LSTM Backprop

[Hochreiter et al., 1997]

f: decides how much we want to forget about the previous info
i & g: decides how much we much to add the current info to C
o: how much we want to out put the current info



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

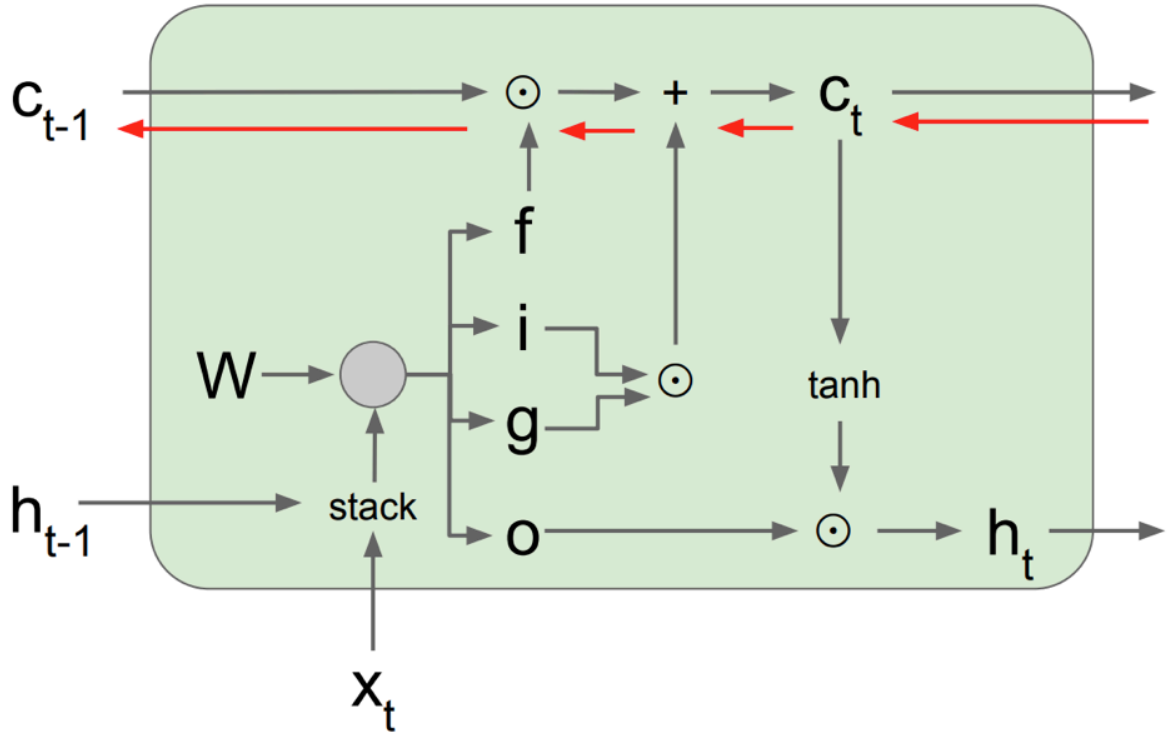
$$c_t = f \odot c_{t-1} + i \odot g$$

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

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

LSTM Backprop

[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

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

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

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

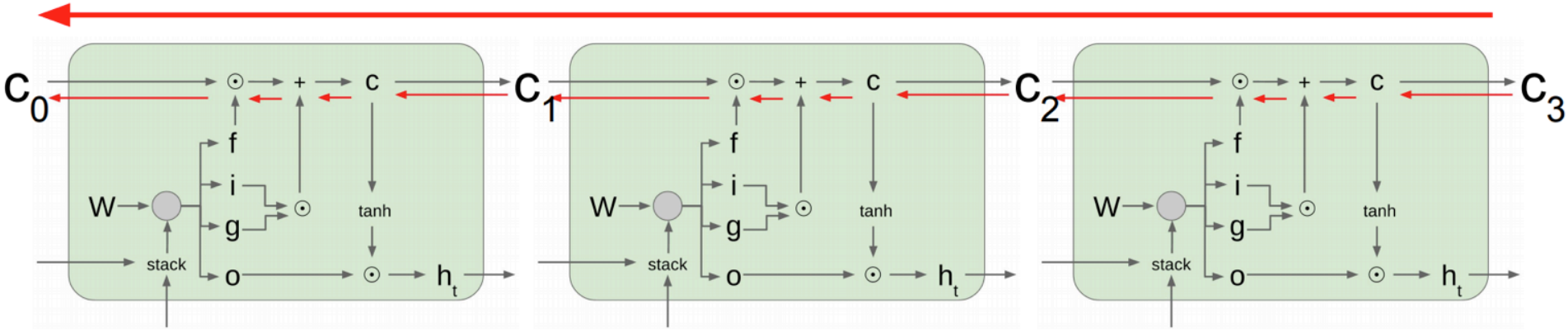
UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

LSTM backprop

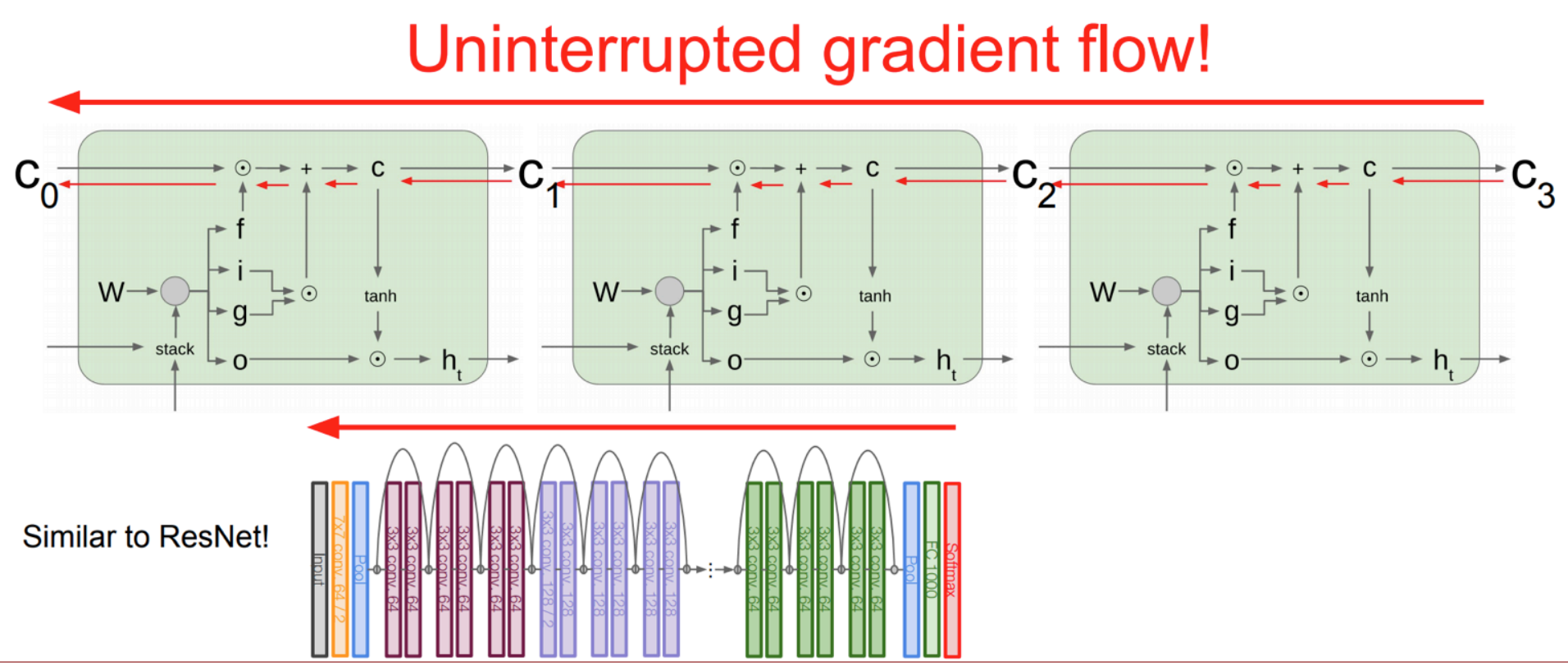
[Hochreiter et al., 1997]

Uninterrupted gradient flow!



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

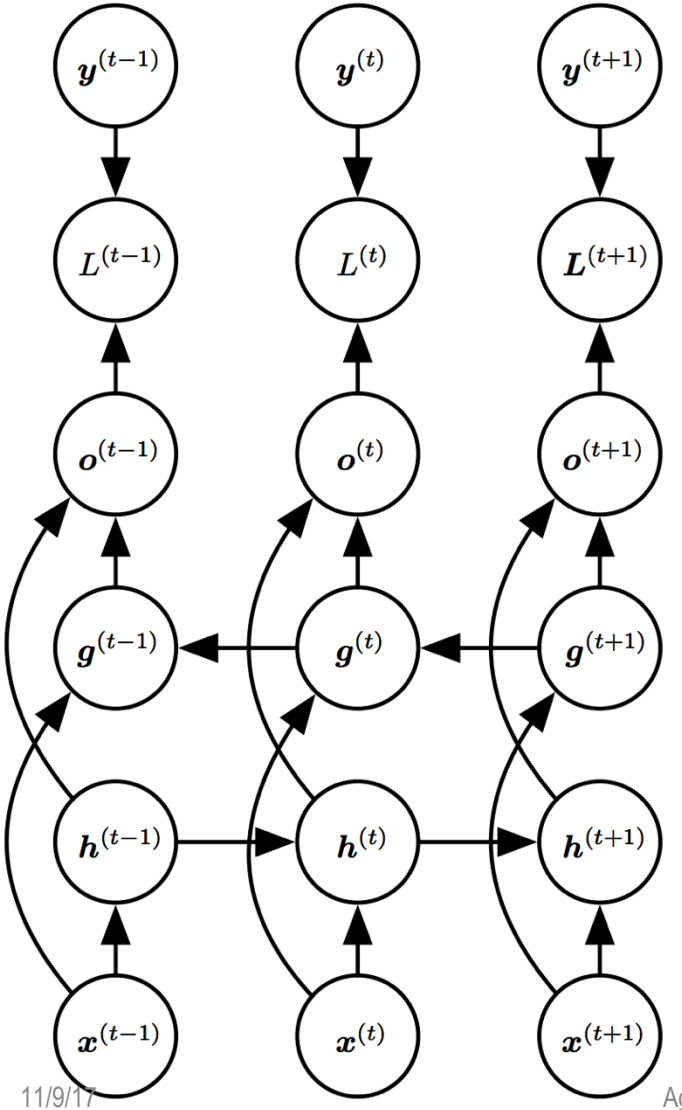
LSTM vs ResNet



UNIVERSITY OF WATERLOO

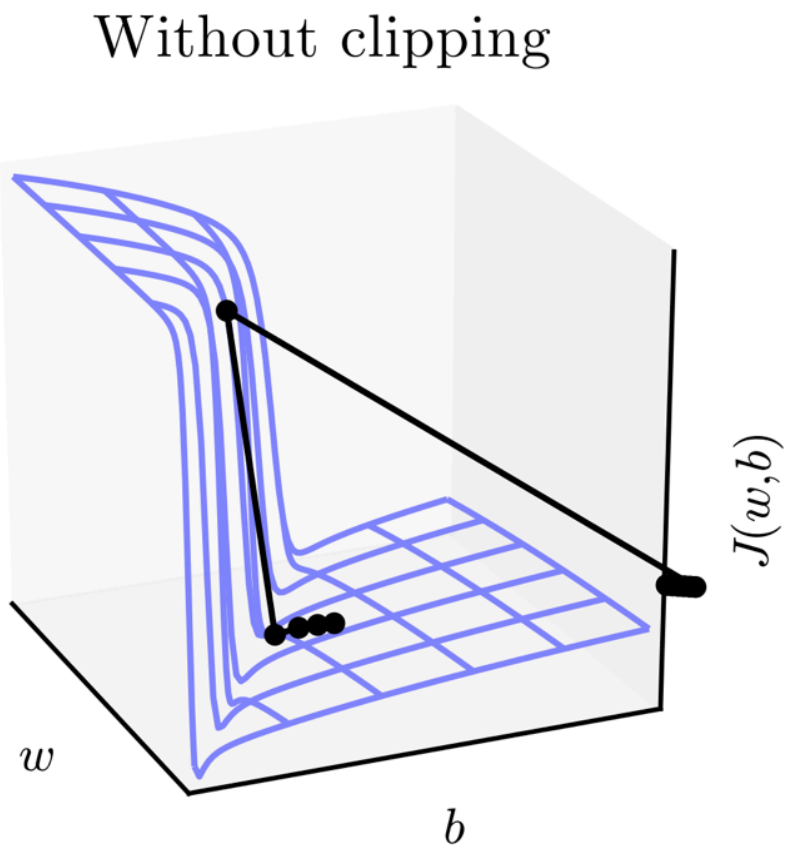
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

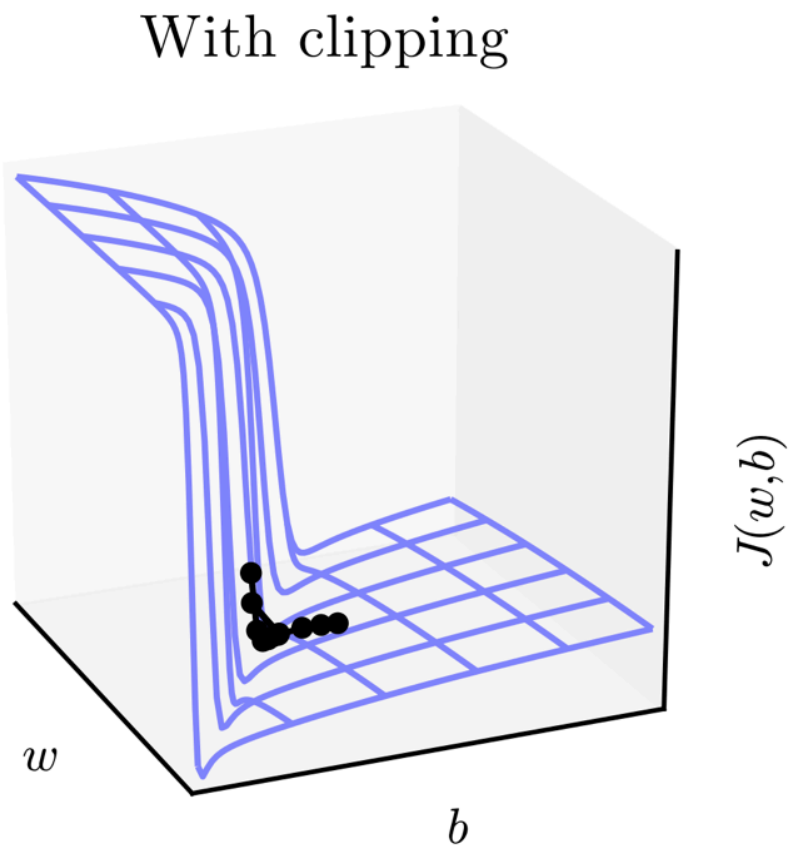
Bi-Directional RNN



UNIVERSITY OF WATERLOO

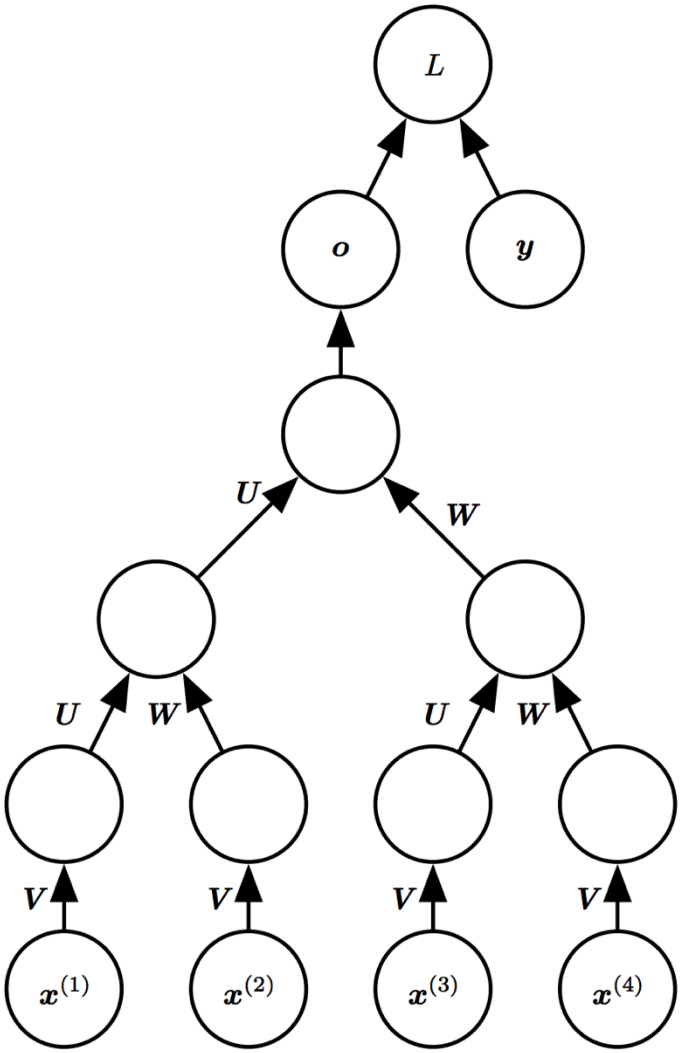
Gradient Clipping





Recursive Networks

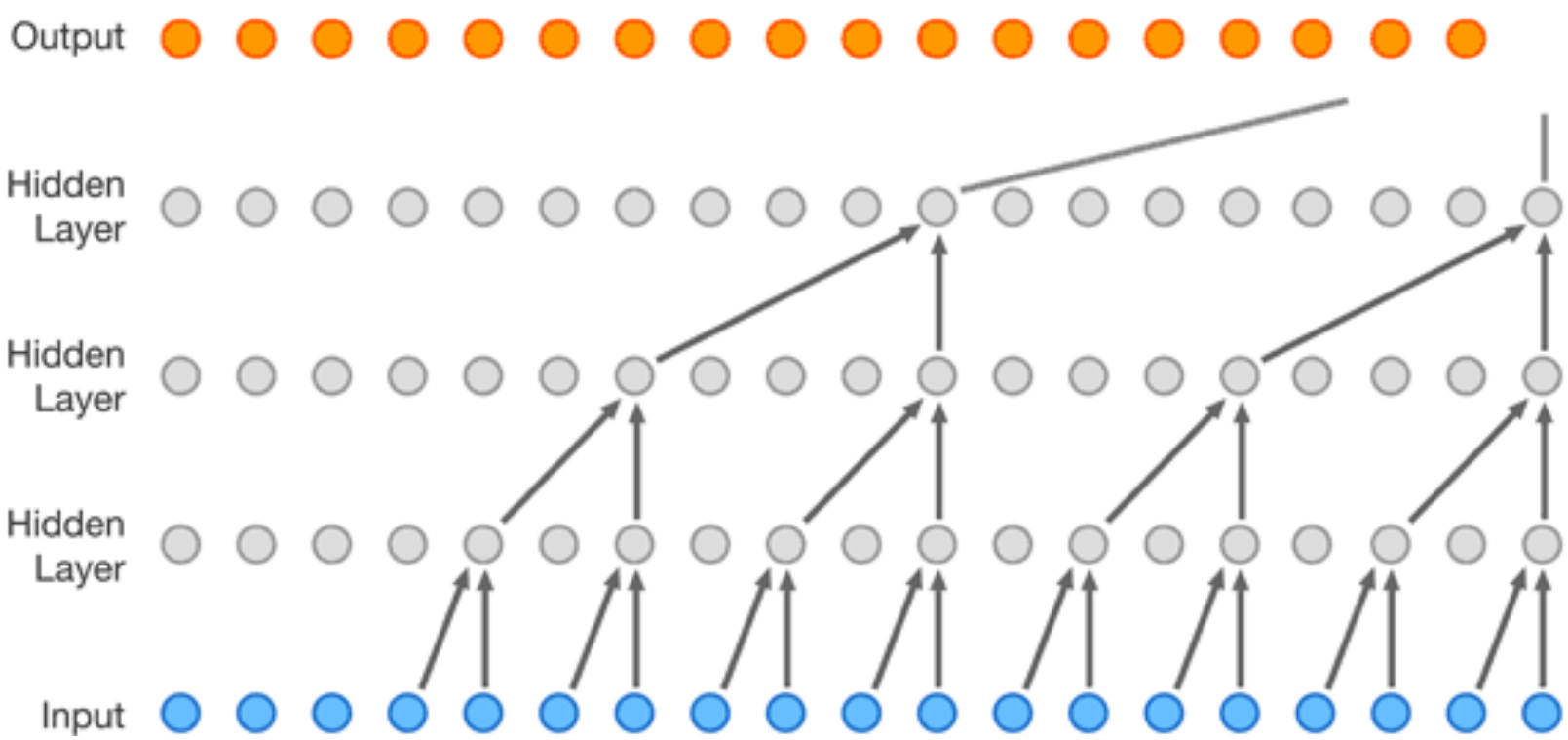
- Superset of Recurrent
- Recurrent means processed in sequential order
- Recursive means processed in any order



UNIVERSITY OF WATERLOO

WaveNet

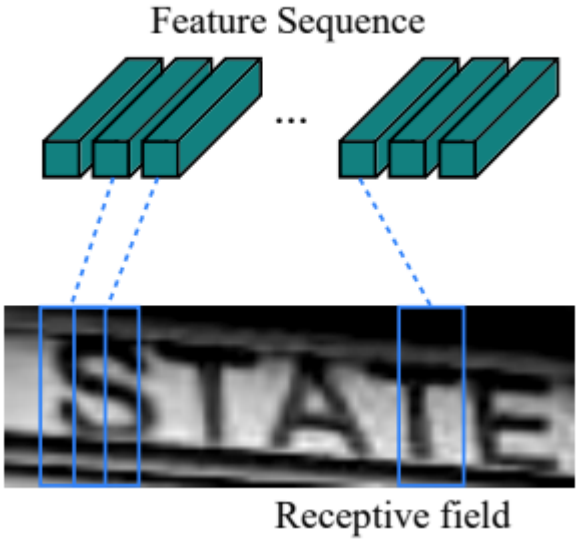
- Audio Generation

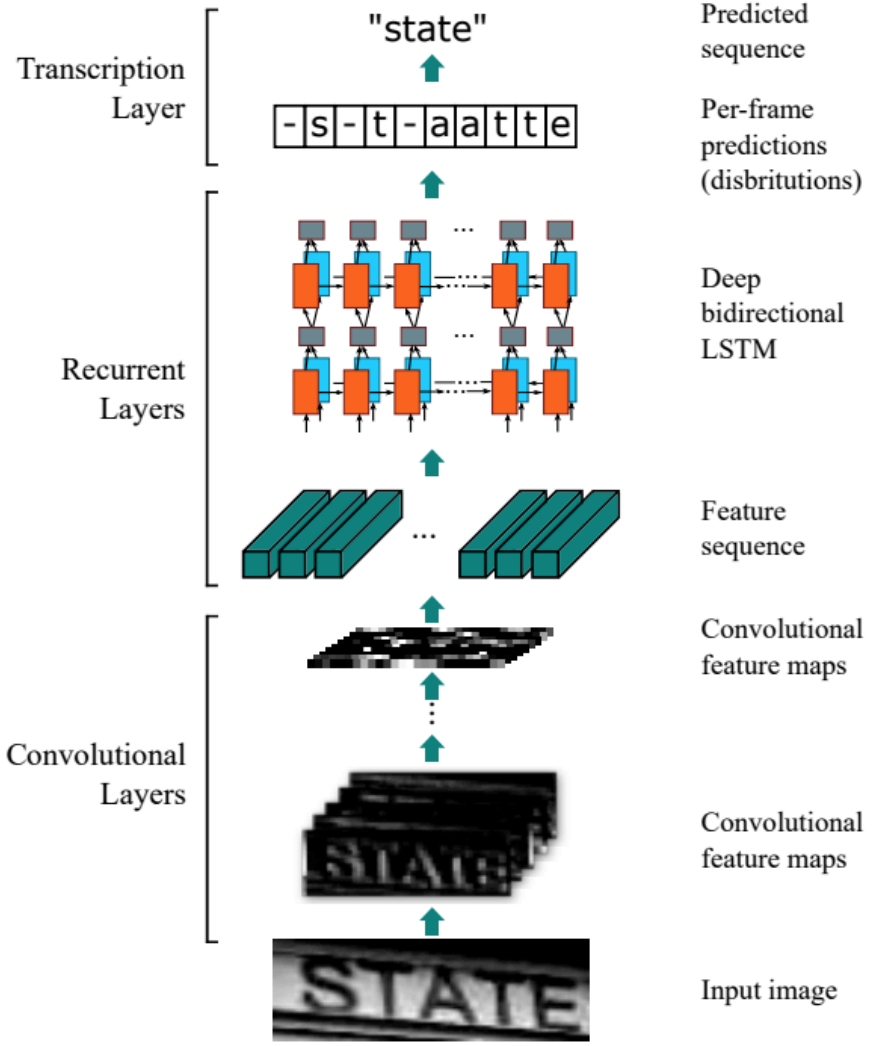


UNIVERSITY OF WATERLOO

Convolutional Recurrent Neural Network

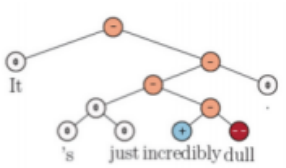
- Object Character Recognition in the wild

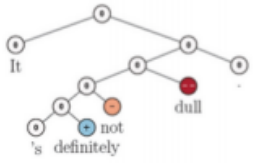


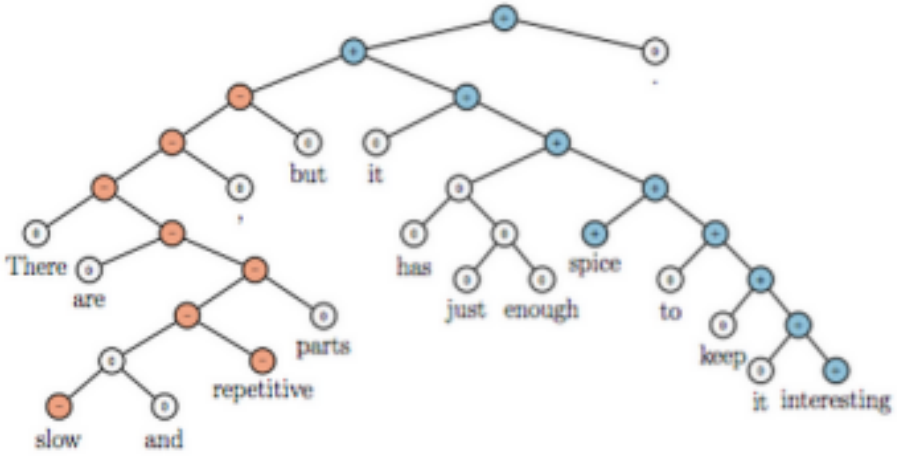


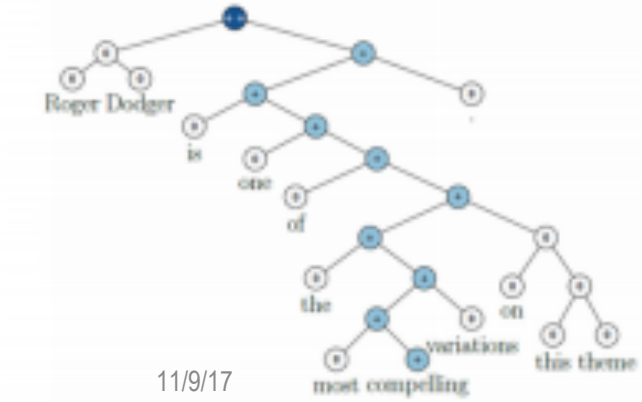
Sentiment analysis - RNTN

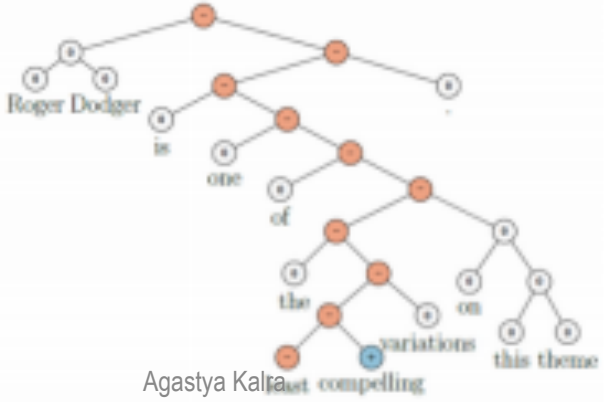
Recursive Networks





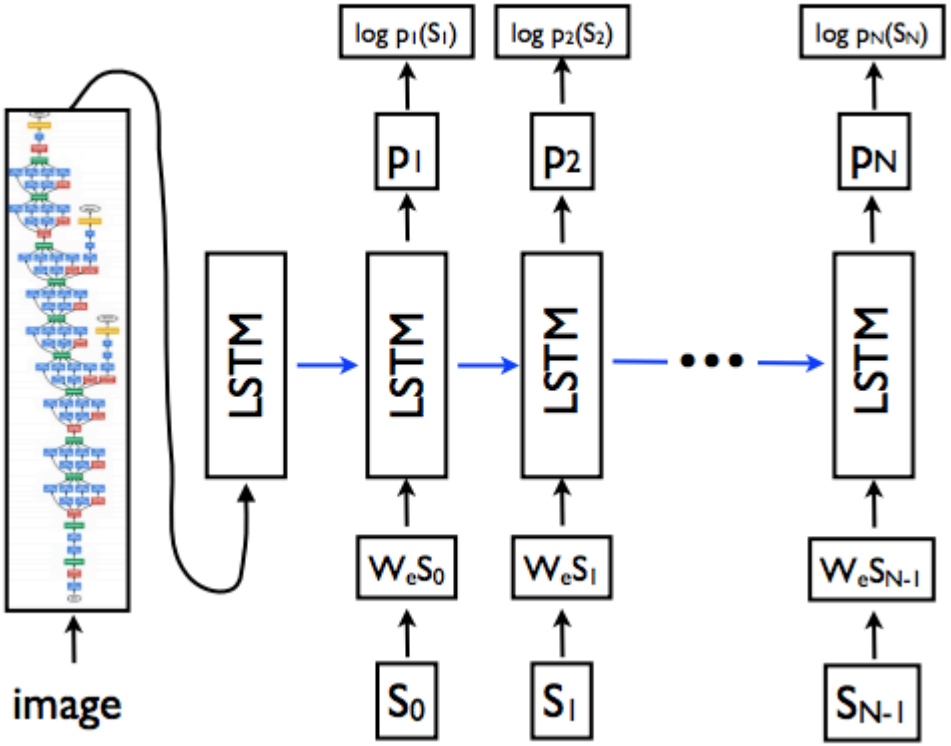


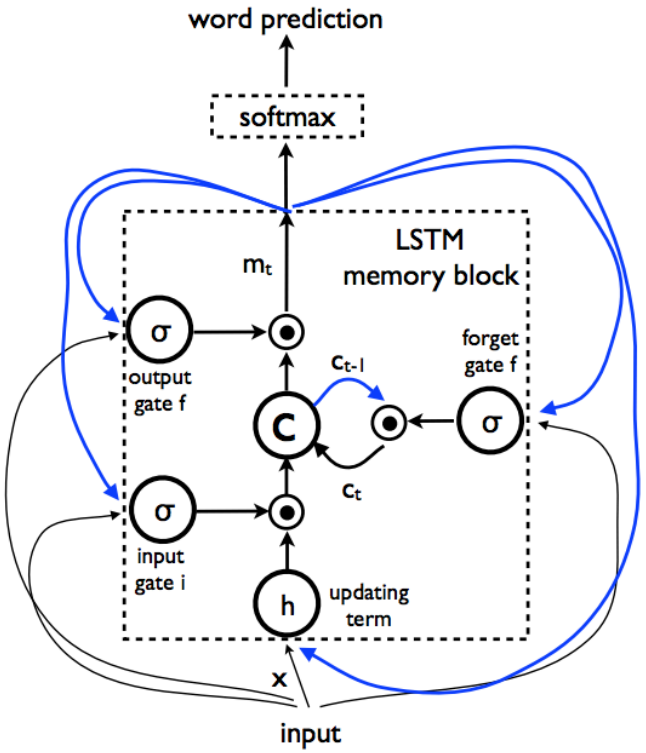




UNIVERSITY OF WATERLOO

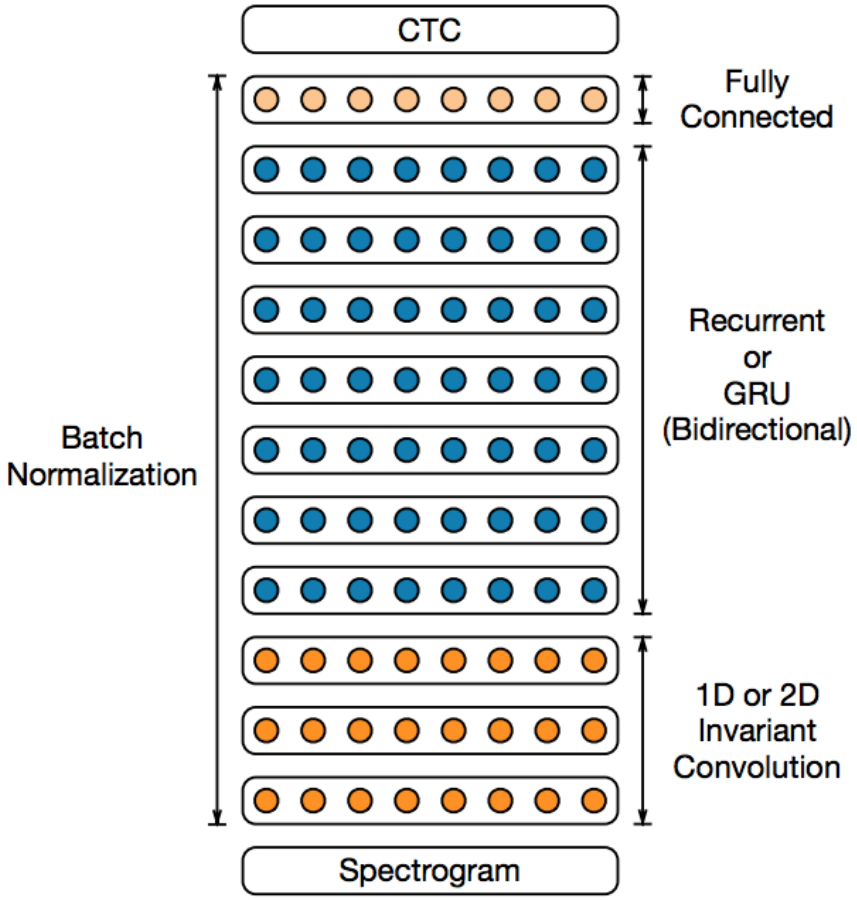
Image Captioning

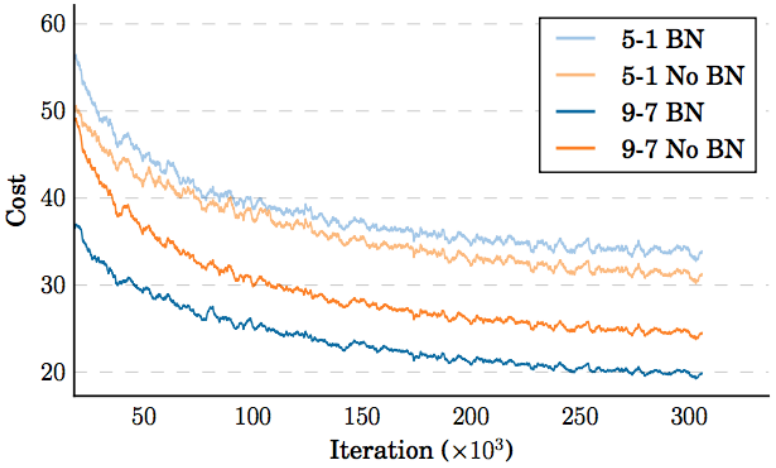




UNIVERSITY OF WATERLOO

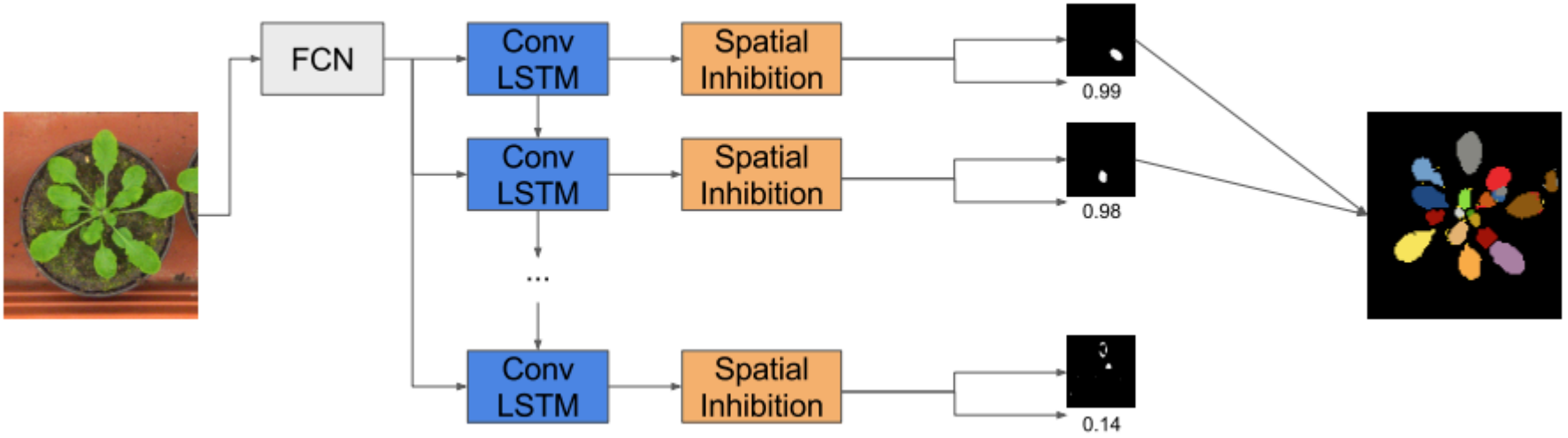
Speech Recognition





UNIVERSITY OF WATERLOO

Recurrent Instance Segmentation



Deep Learning for Retail