UNIVERSITY OF WATERLOO

# Recurrent Neural Networks

DSG Meeting

June 7, 2017

Salman Mohammed

David R. Cheriton School of Computer Science

University of Waterloo

# Acknowledgement


Andrej Karpathy
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

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

Richard Socher, Christopher Manning
http://web.stanford.edu/class/cs224n/syllabus.html

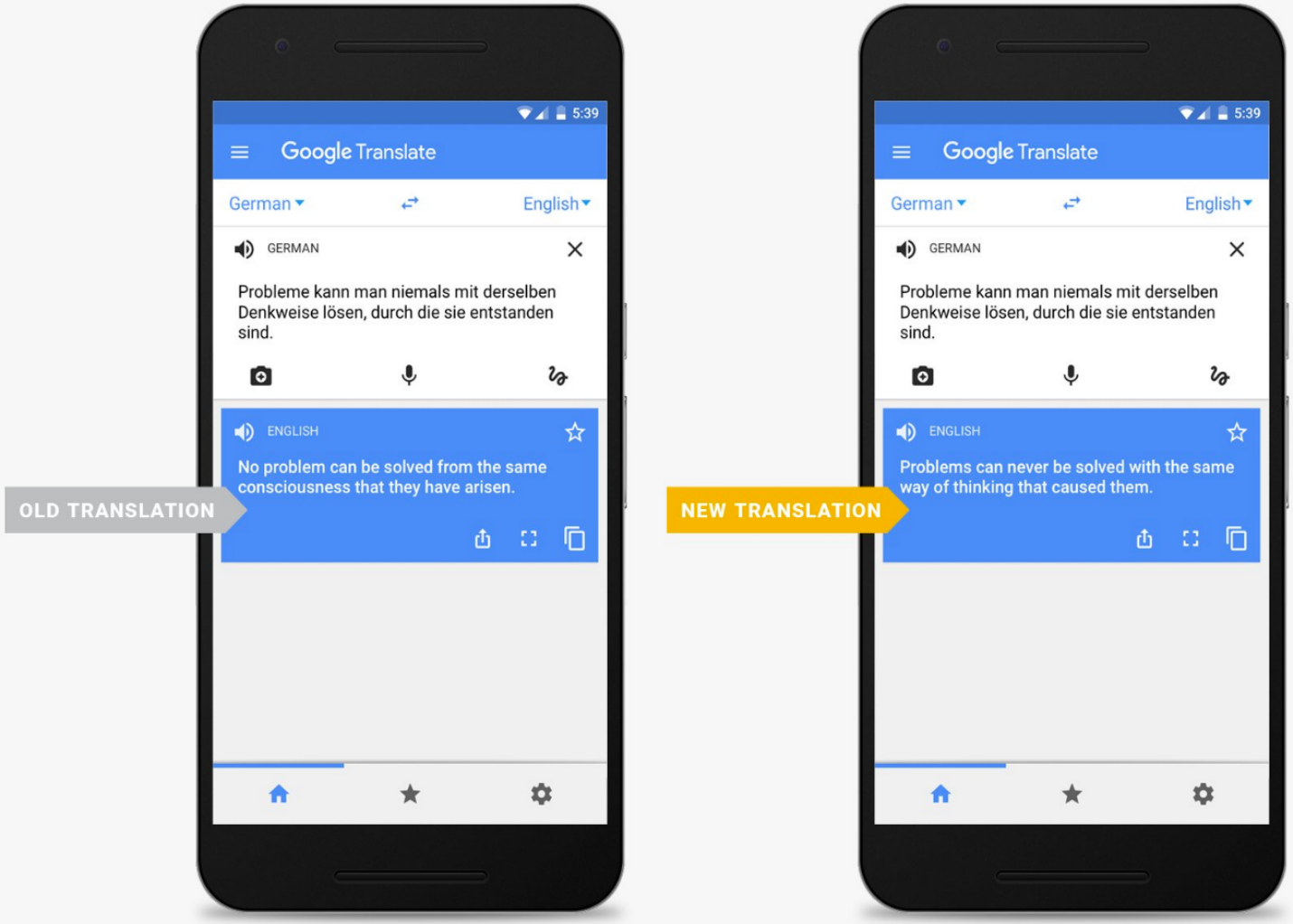
Jimmy Lin
slide template taken from https://lintool.github.io/bigdata-2017w

Denny Britz
http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Harini Suresh
http://introtodeeplearning.com/Sequence%20Modeling.pdf

# Motivation


Source: https://www.apple.com/newsroom/2017/01/hey-siri-whos-going-to-win-the-super-bowl/




Source: https://blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/

# Assuming you know…

Word Vectors

dense vector representation for words

Fully Connected Neural Networks

every node in a layer connected to all nodes in the previous layer

Idea of Backpropagation

training a neural network

# Limitations of NNs

Constrained API

fixed size input(image) and output(classes)

Modelling Sequences

traditional networks have no sense of ‘state’

use reasoning about previous events to make decision

# Use of RNNs

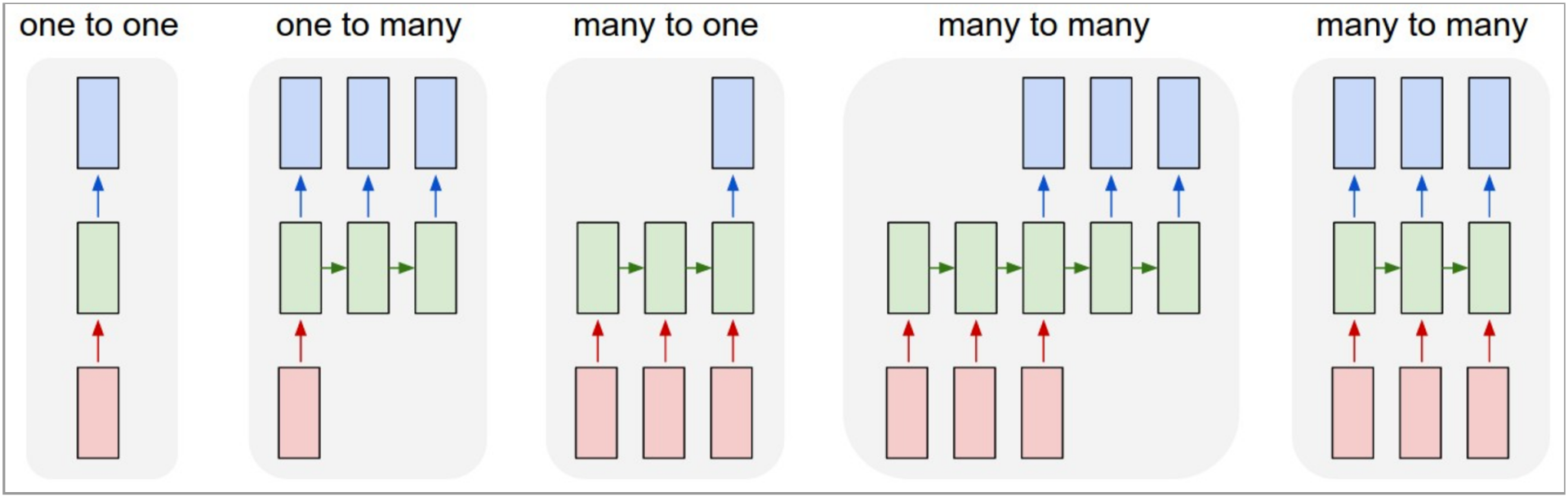



Image Captioning

Machine Translation

Image Classification
(ConvNets)

Sentiment Analysis
Text Classification
Relation Prediction

Entity Detection
Video Frame Classification

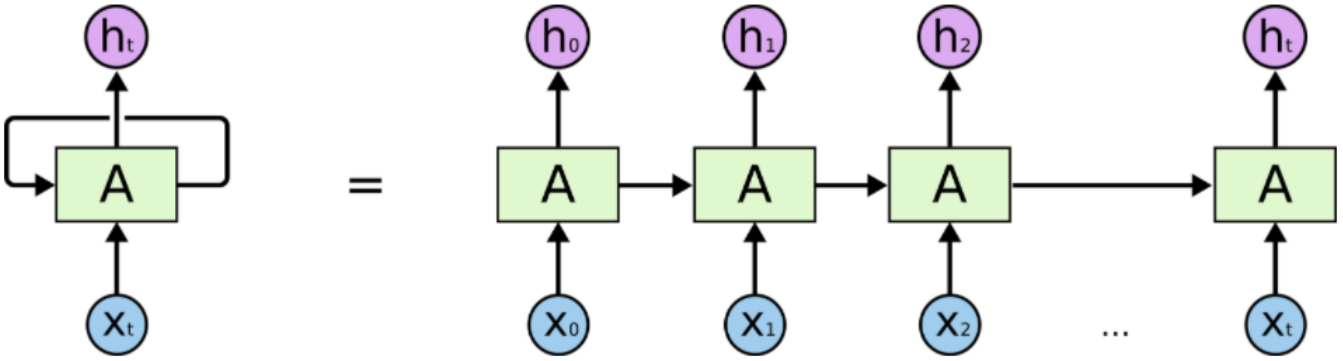
# Recurrent NNs

Input: $x_t$

word embedding

Memory/State: $h_t$

think "sentence embedding"




An unrolled recurrent neural network.

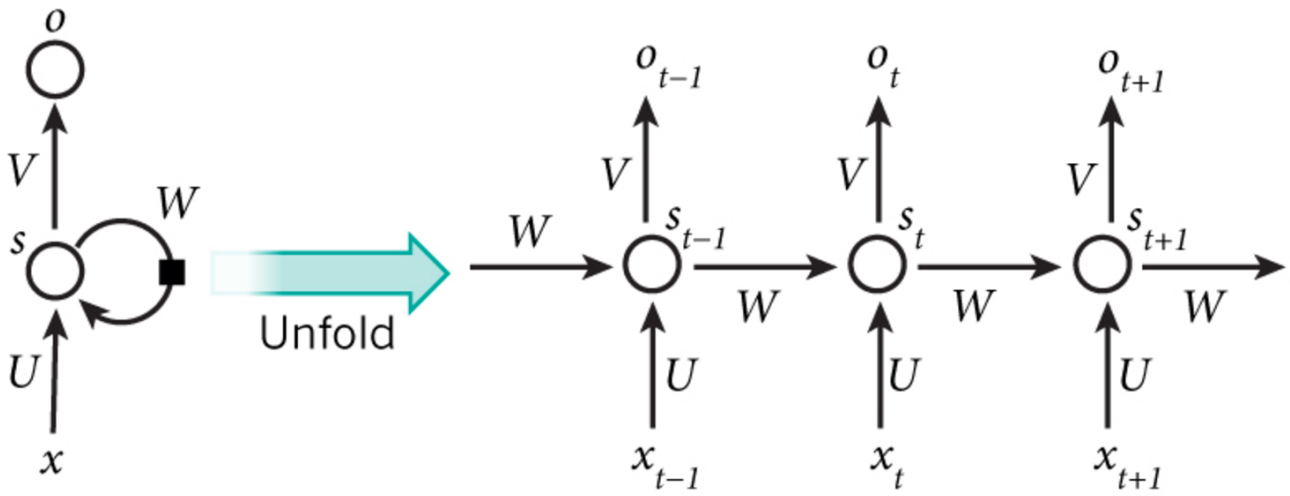
# Recurrent NNs (more detail)

State: $s_t = f(U.x_t + W.s_{t-1})$

*f* is a non-linear function (ReLU, tanh)

Output: $o_t = softmax(V.s_t)$

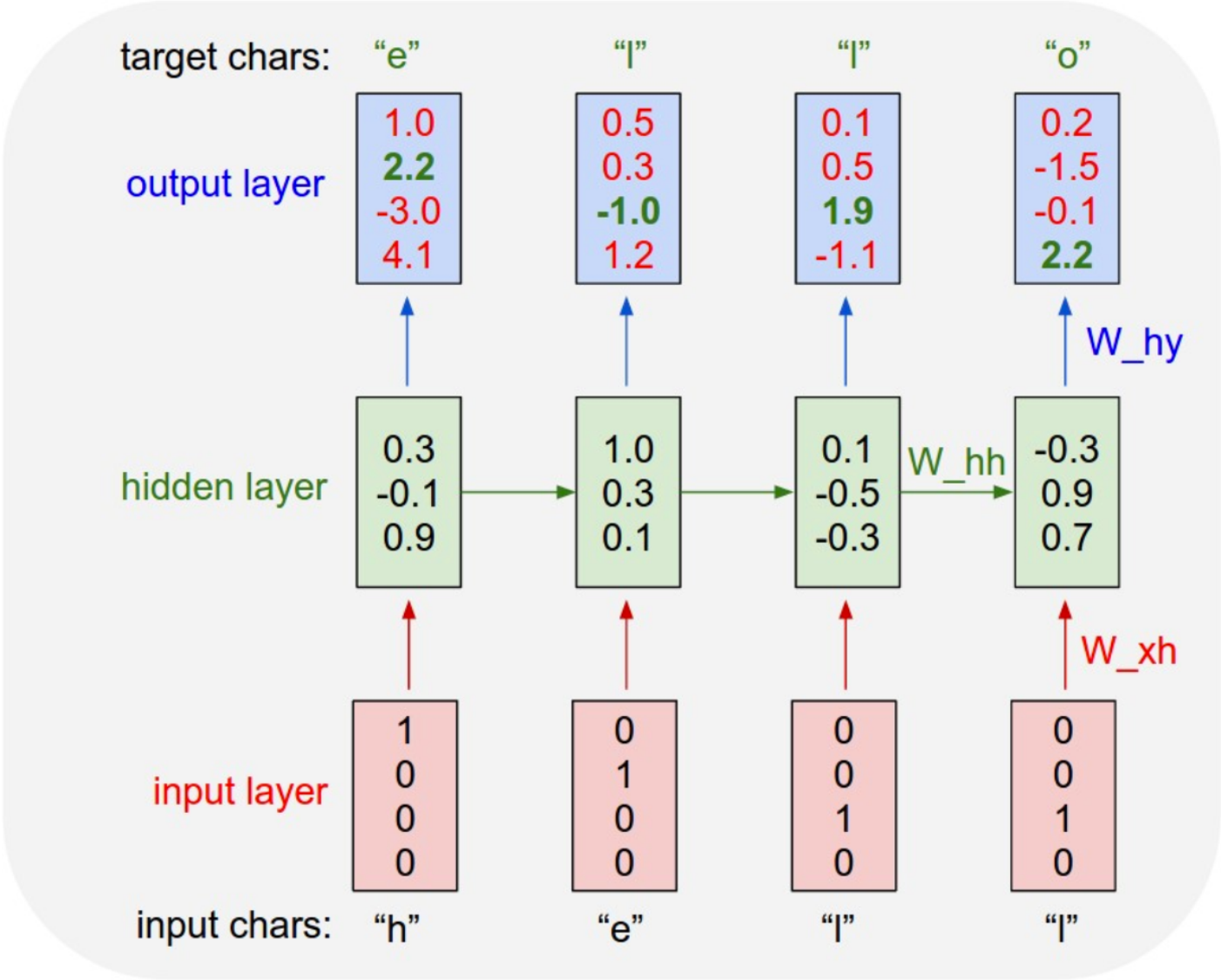
output a vector of probabilities



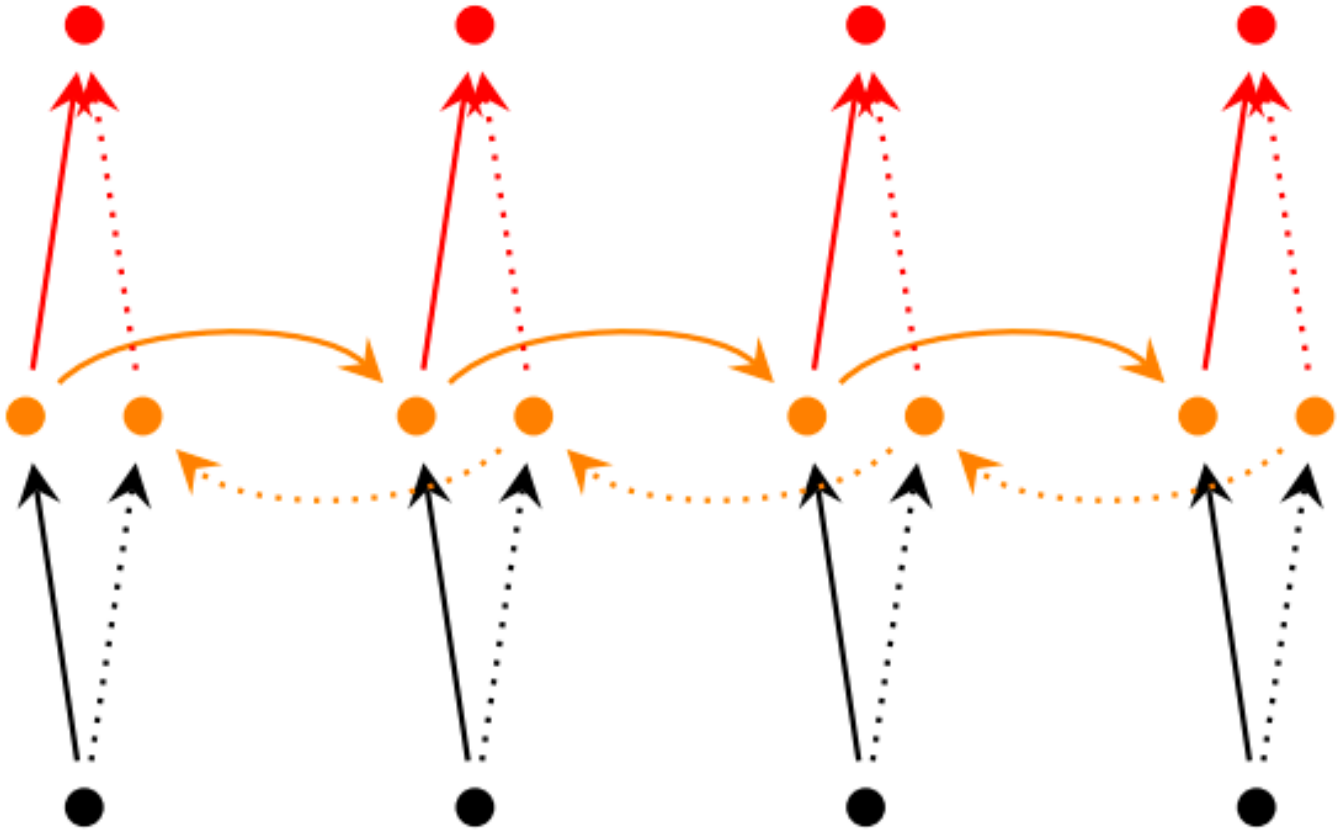

*A recurrent neural network and the unfolding in time of the computation involved in its forward computation.*

*Source: Nature*

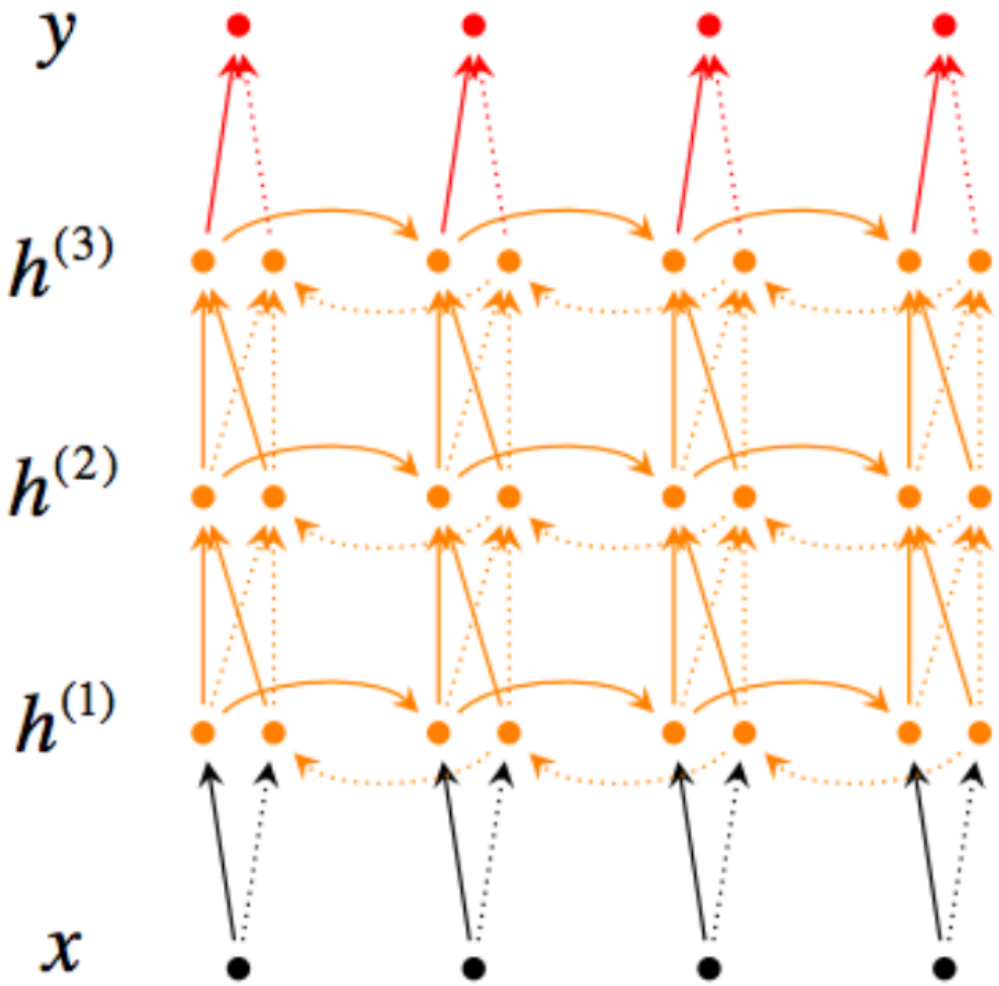
# Example: Character Language Model



# Bi-directional RNNs



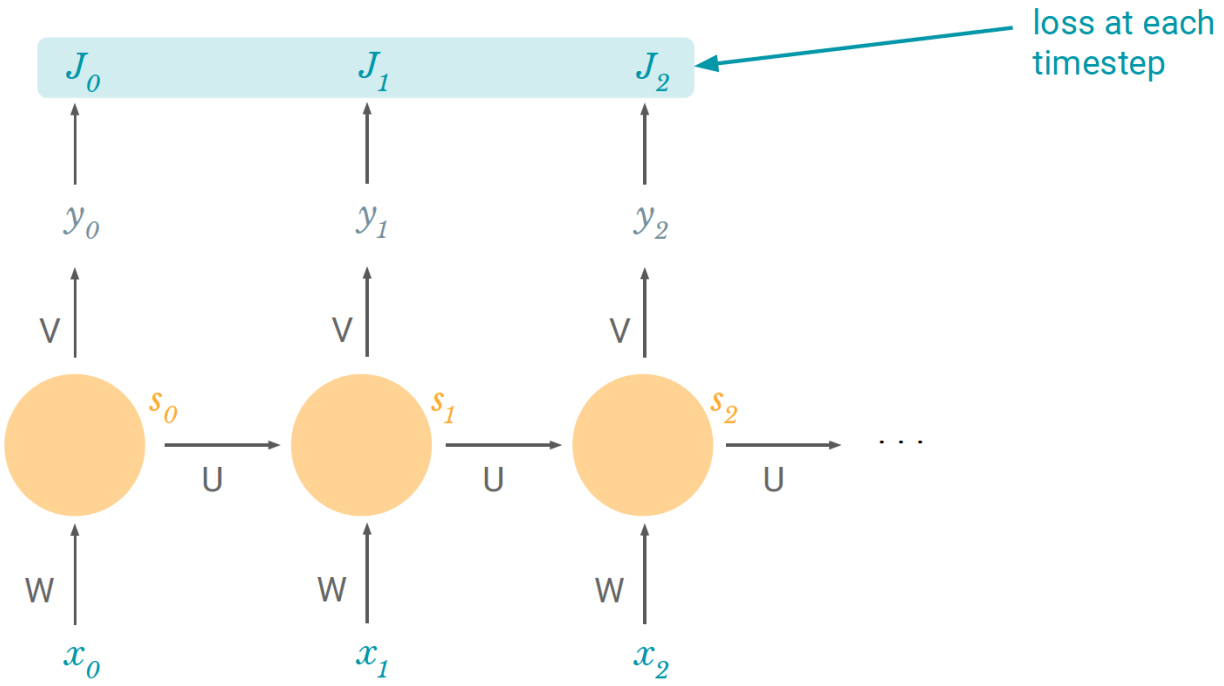
# Deep Bi-directional RNNs



# Backpropagation Through Time (BPTT)

Weights shared by all the time steps in the network

To calculate gradient at $t$=3, backpropagate 2 steps and sum up gradients



# Problem with RNNs

Learning long-term dependencies

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

Vanishing/Exploding gradient problem

notice that the same weight matrix is multiplied at each time step during forward and backward propagation

# Long Short Memory Networks (LSTMs)

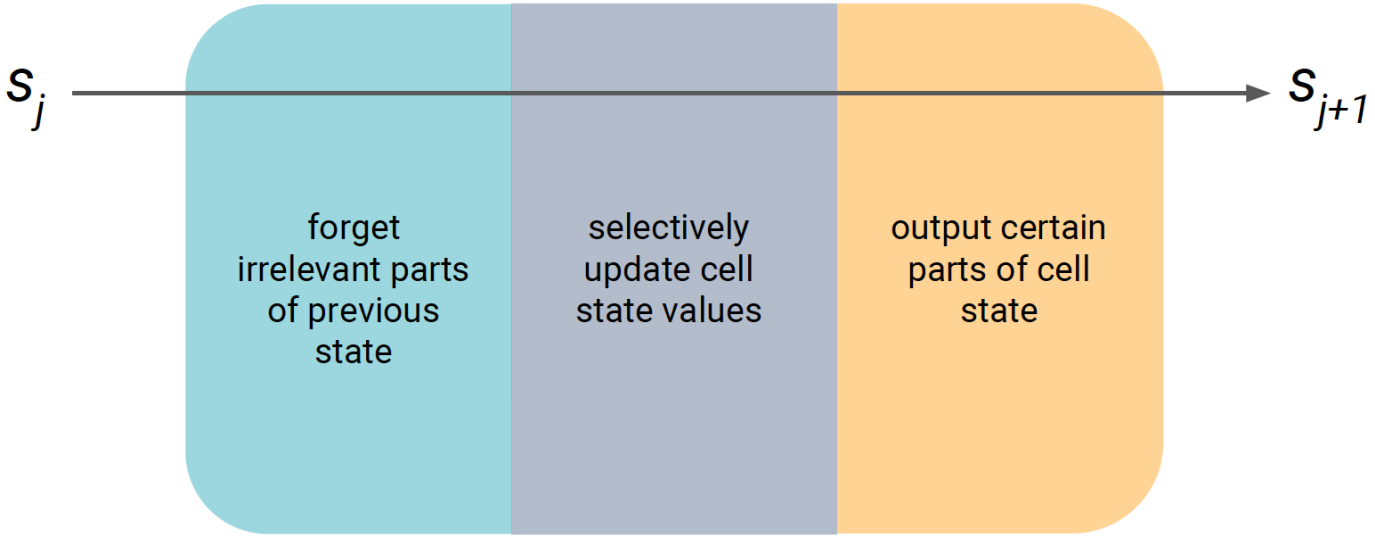
Avoid long term dependency problem

remember information for a long time

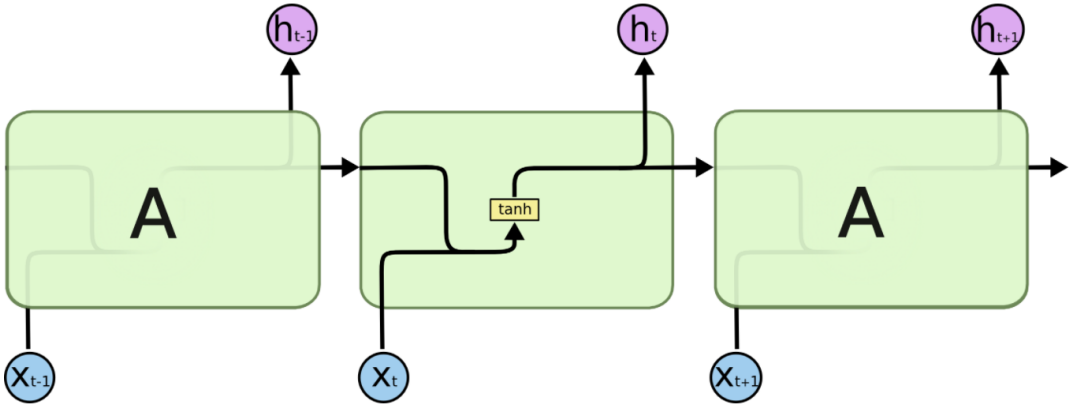
Idea: gated cells

complex node with gates controlling what information is passed through

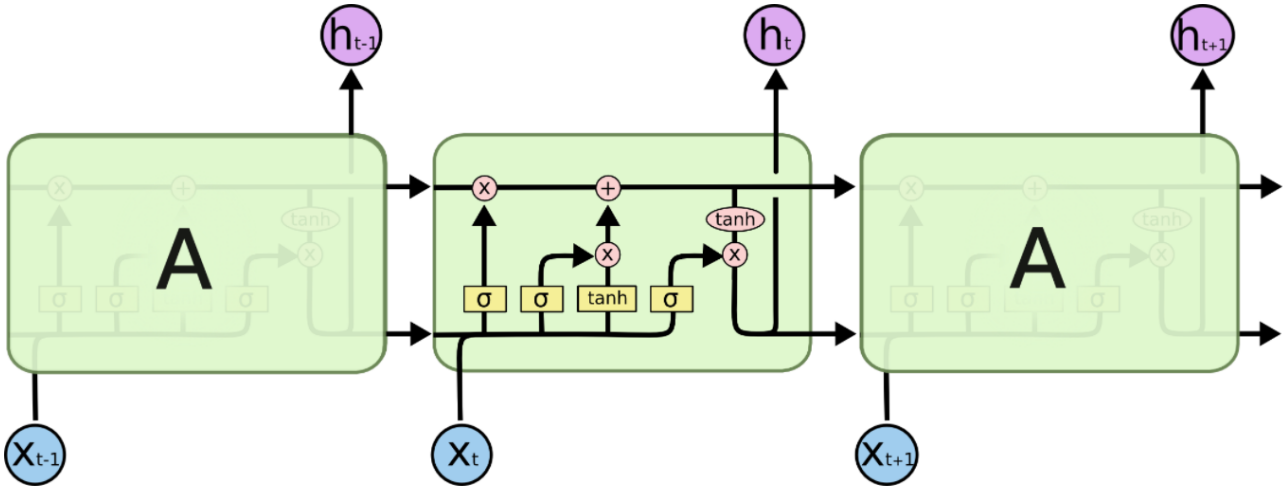
maintains an additional "cell state" - $c_t$



# RNNs vs. LSTMs

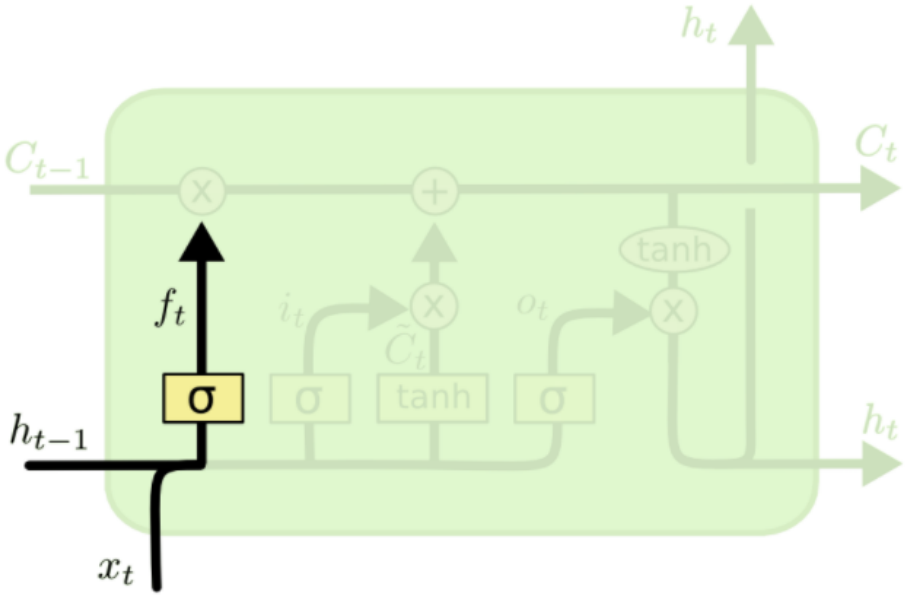



The repeating module in a standard RNN contains a single layer.



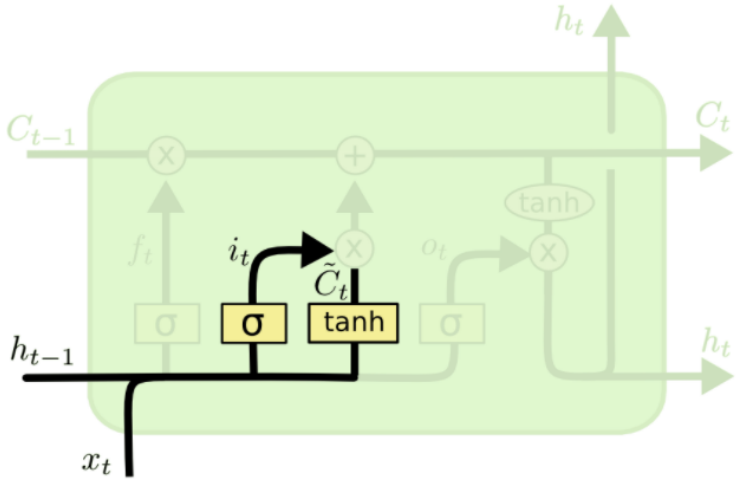

The repeating module in an LSTM contains four interacting layers.

# Forget Gate



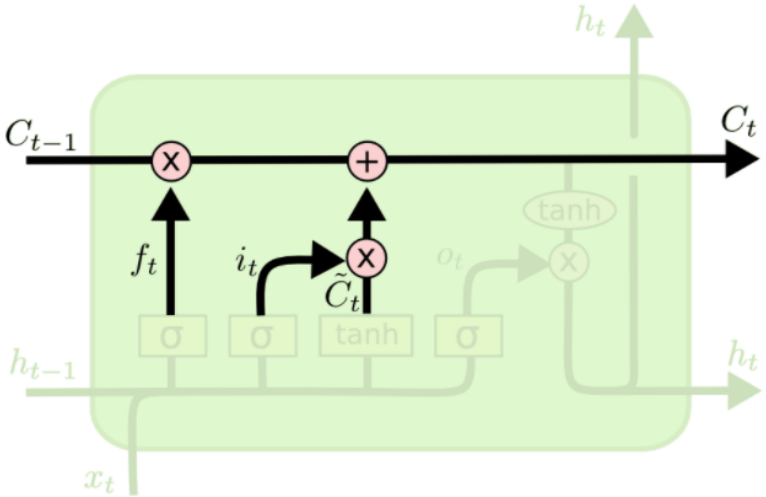

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

# Update Cell State



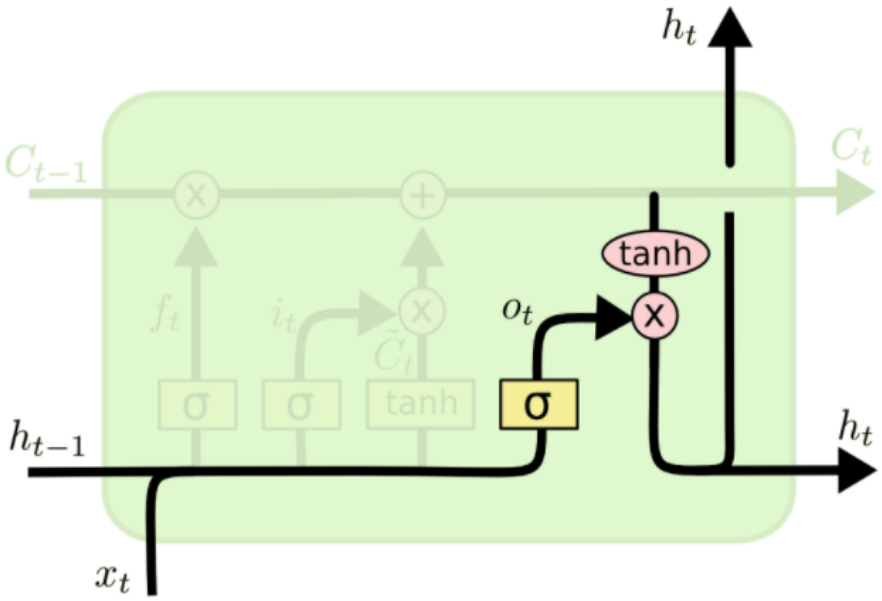

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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




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

# Output Gate




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

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


Source: Google

# Tricks of the Trade

Activation function: try ReLU

prevents from shrinking gradients

Optimization algorithm: try Adam

computes adaptive learning rate; usually faster convergence

read: http://sebastianruder.com/optimizing-gradient-descent/index.html

Weight initialization: use Xavier initialization

make sure weights start out 'just right'

Prevent overfitting: dropout, L2 regularization

dropout prevents feature co-adaptation

remember to scale model weights at test time for droput

# Tricks of the Trade (cont'd)

## Random Hyperparameter Search

grid search is a bad idea; read: https://arxiv.org/abs/1206.5533

Some features more important than others

## Batch Normalization

make activations unit gaussian distribution at the beginning of the training

insert BatchNorm layer immediately after fully connected/convolutional layers

## Initialize recurrent weight matrix, $W^{hh}$, to identity matrix

helps vanishing gradient problem. read: https://arxiv.org/pdf/1504.00941.pdf

## Gradient clipping

helps exploding gradient problem


Source: Google