Recurrent Neural Networks

Many slides from Lana Lazebnik, Arun Mallya

Sequential Prediction Tasks

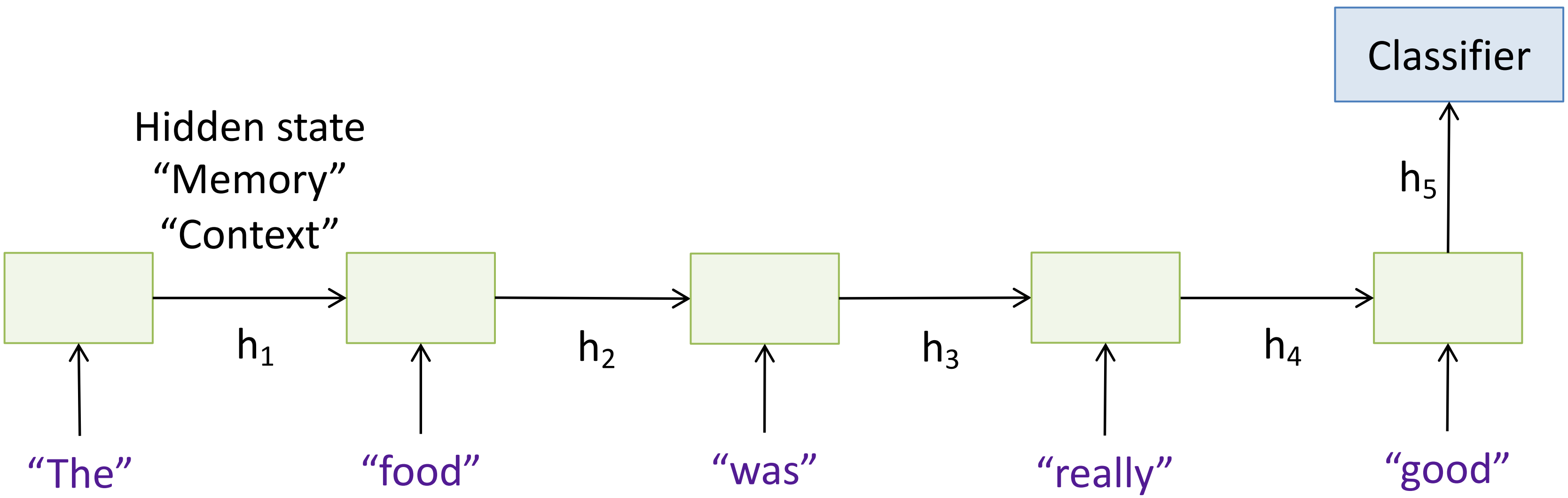
- ConvNets: one-to-one mapping
- What if the input and/or output is a variable-length sequence?

Text Classification

- **Sentiment classification:** classify a restaurant or movie or product review as positive or negative
  - “The food was really good”
  - “The vacuum cleaner broke within two weeks”
  - “The movie had slow parts, but overall was worth watching”
- What feature representation or predictor structure can we use for this problem?

Sentiment Classification

- “The food was really good”



Recurrent Neural Network (RNN)

Image Caption Generation

- Given an image, produce a sentence describing its contents

"The dog is hiding"

Image Caption Generation

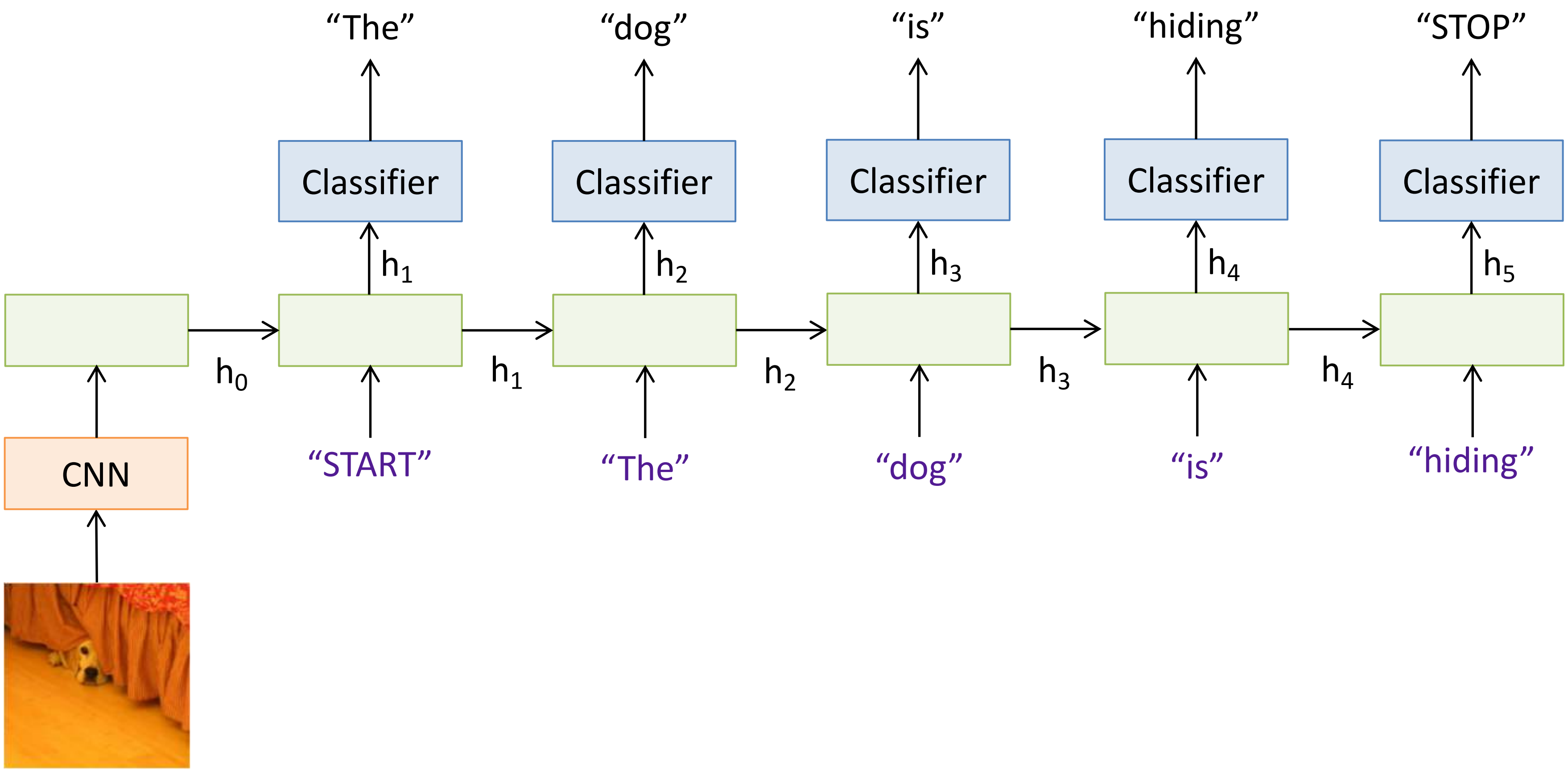
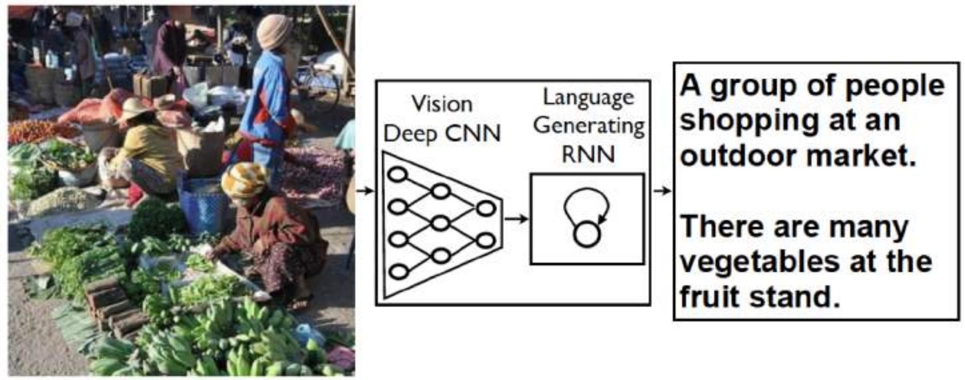
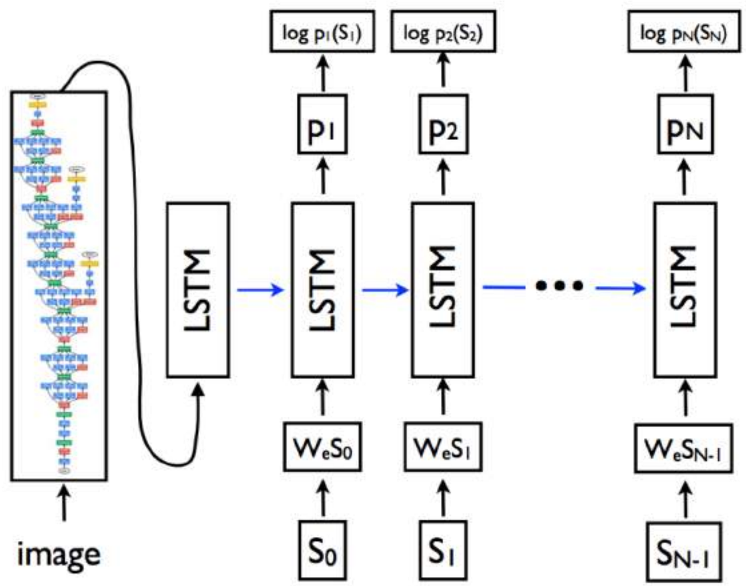


Image Caption Generation





O. Vinyals, A. Toshev, S. Bengio, D. Erhan, [Show and Tell: A Neural Image Caption Generator](), CVPR 2015

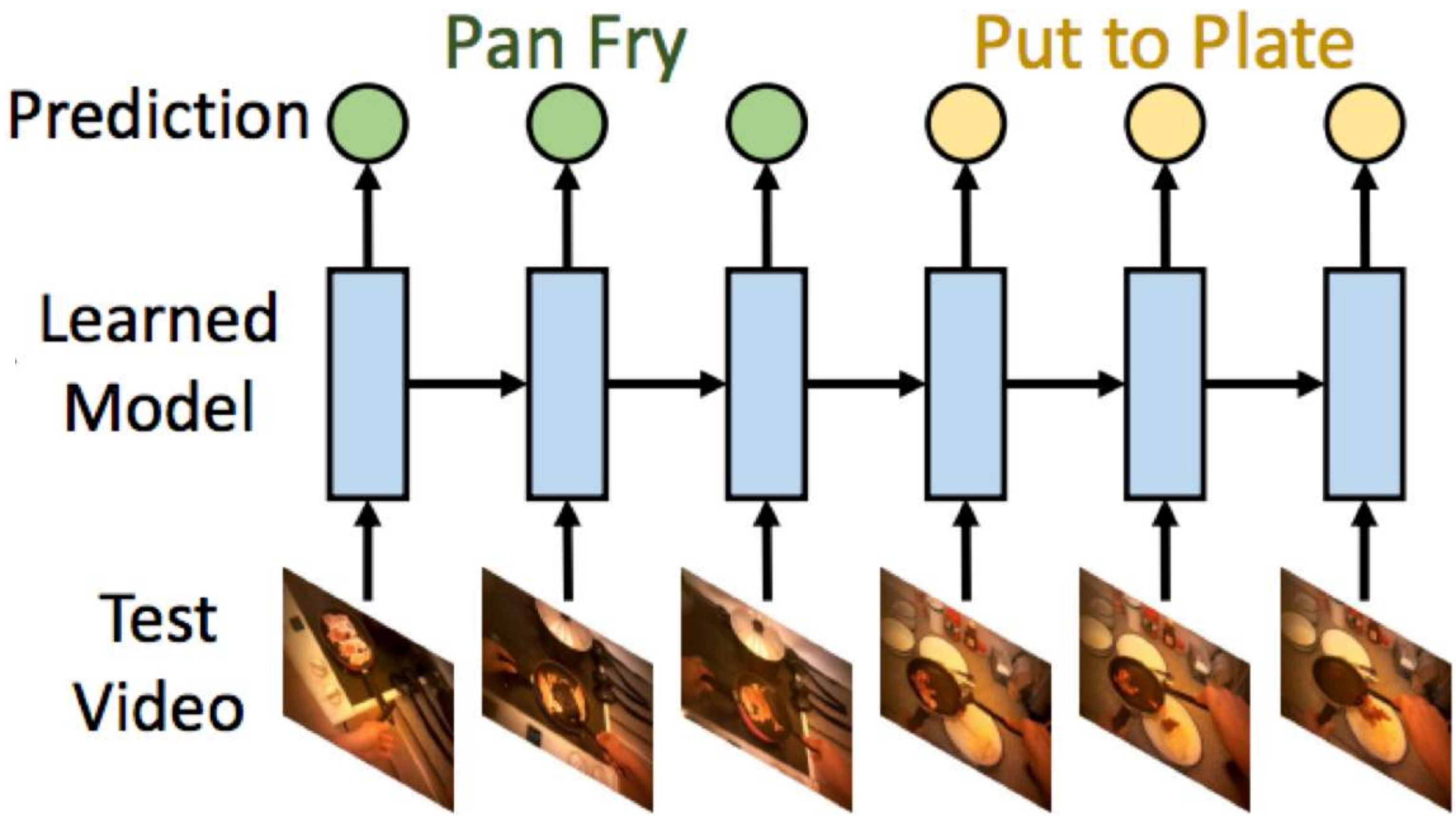
Image Caption Generation

Temporal Action Segmentation

- Given a video, annotate each frame with an action label

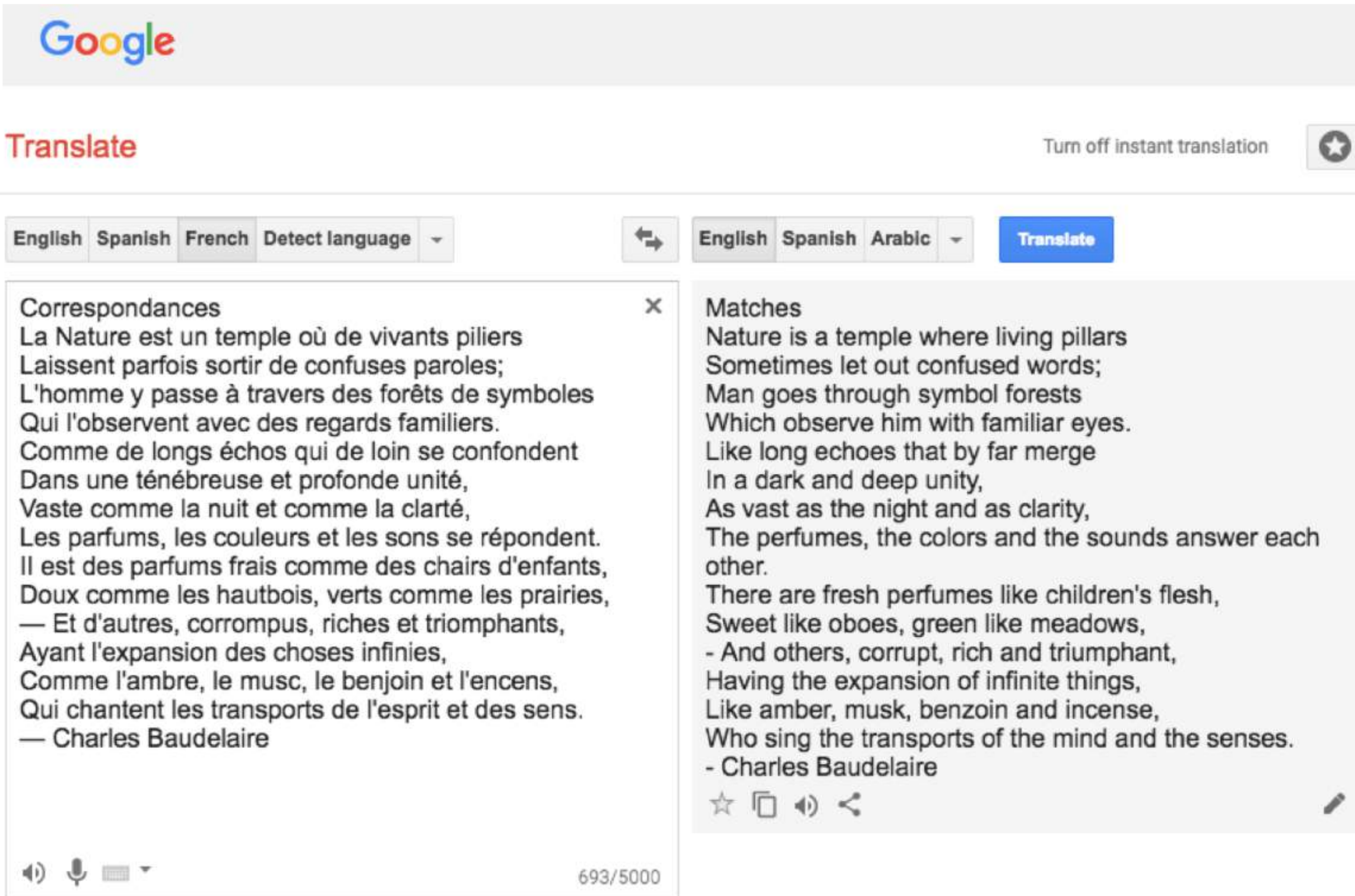
[Huang, Fei-Fei, Niebles]

Temporal Action Segmentation



[Huang, Fei-Fei, Niebles]

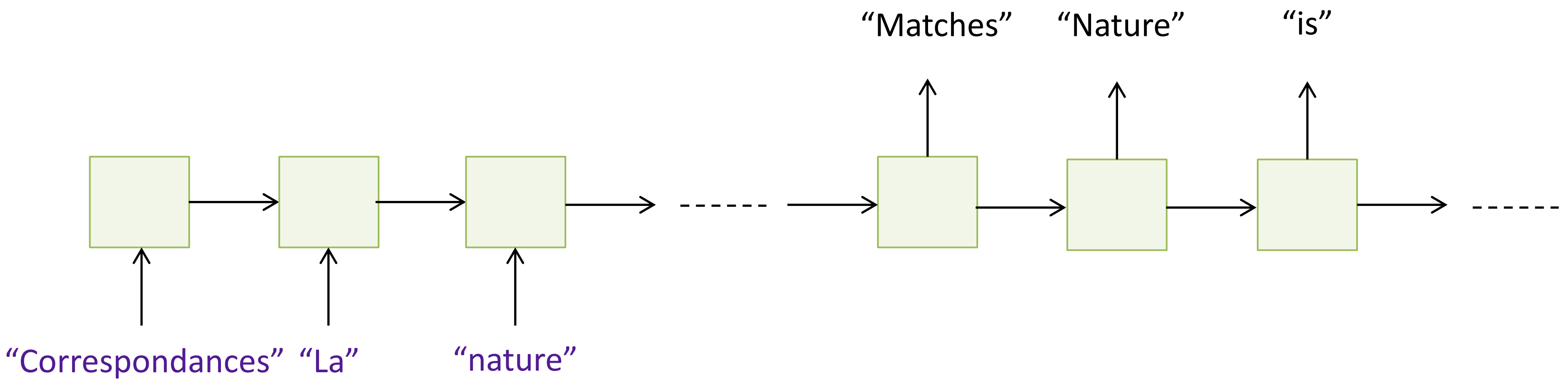
Machine Translation



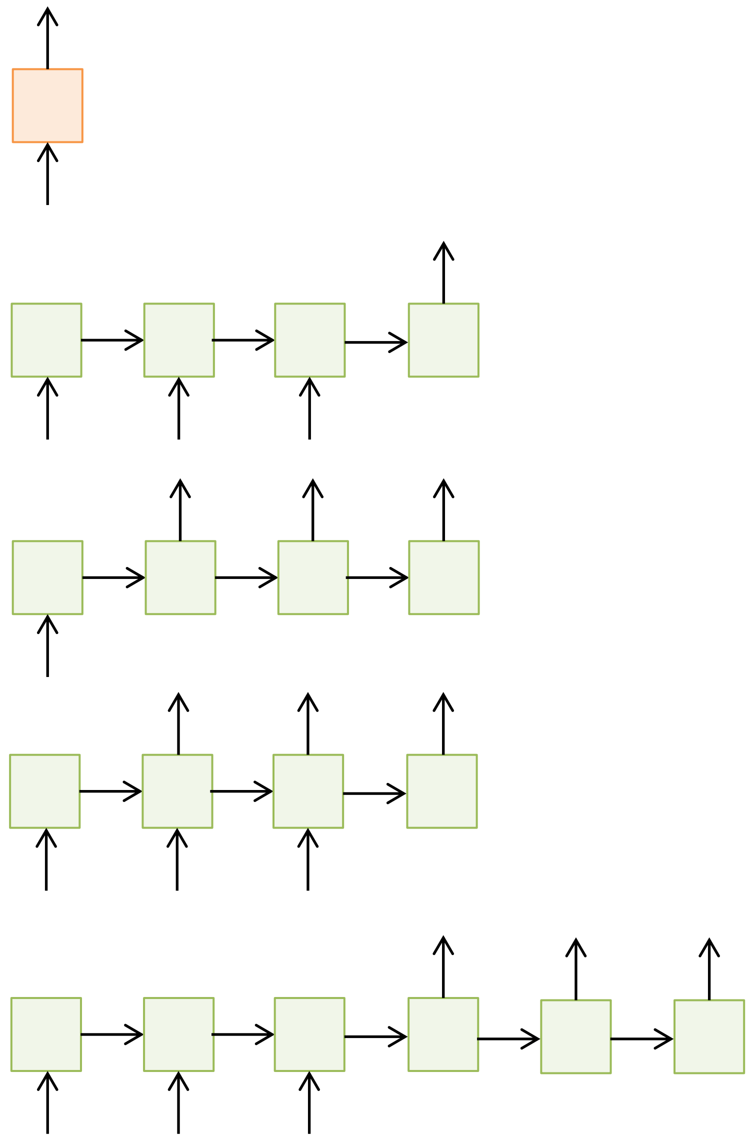
https://translate.google.com/

Machine Translation

- Multiple input – multiple output (or sequence to sequence)



Summary: Input-output Scenarios



Single - Single — Feed-forward Network (ConvNets)

Multiple - Single — Sentiment Classification

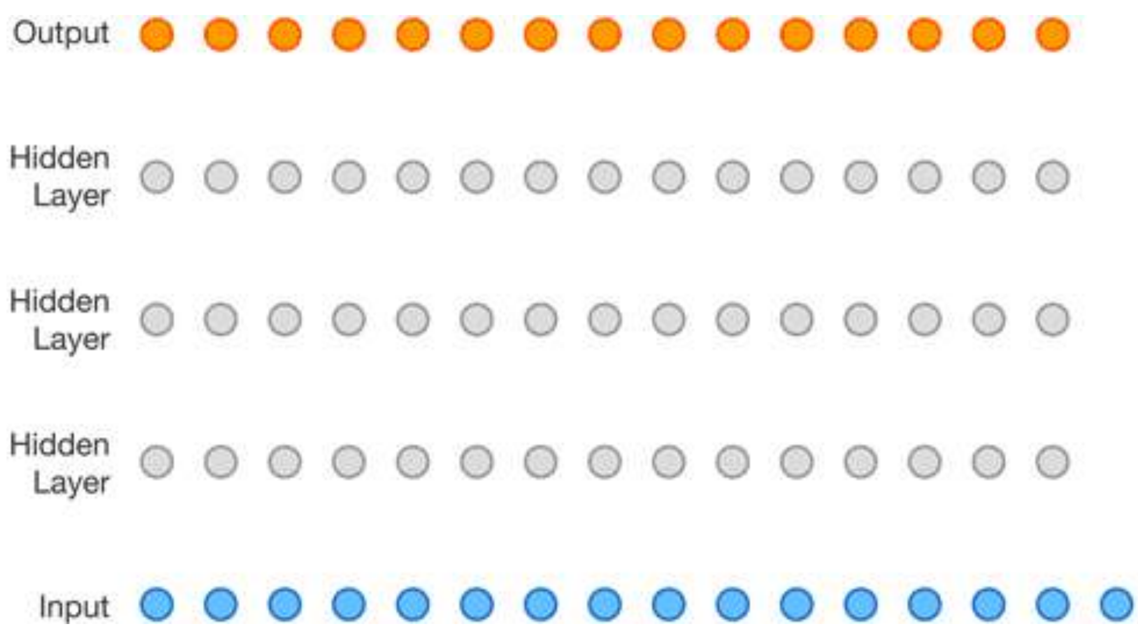
Single - Multiple — Image Captioning

Multiple - Multiple — Temporal Action Segmentation

Multiple - Multiple — Machine Translation

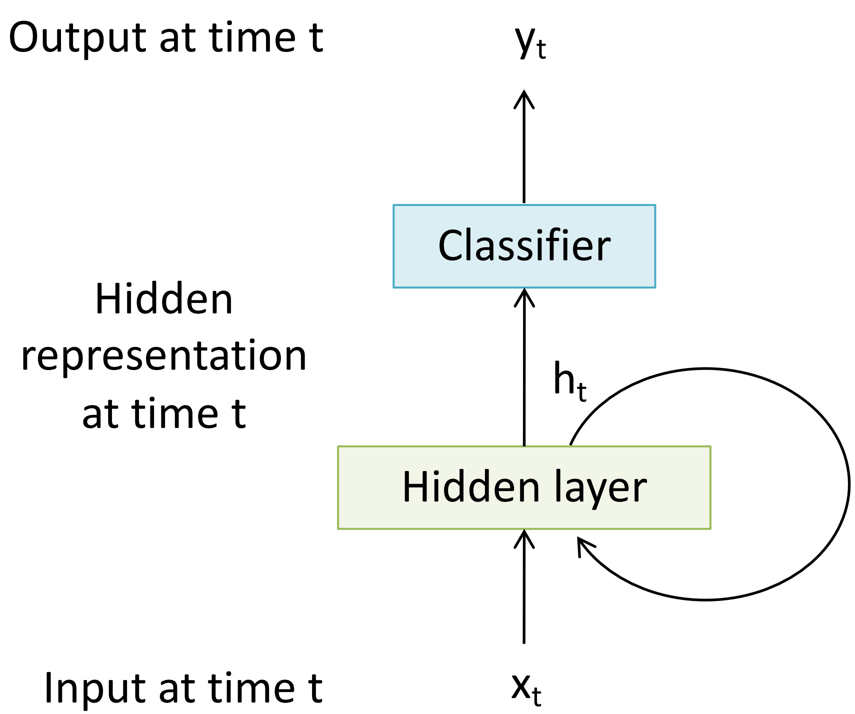
A Simple Solution: 1D ConvNets

- 1D feed-forward convolutional networks
  - Fixed size input / output + Sliding windows



WaveNet: A Generative Model for Raw Audio, DeepMind

Recurrent Neural Network (RNN)

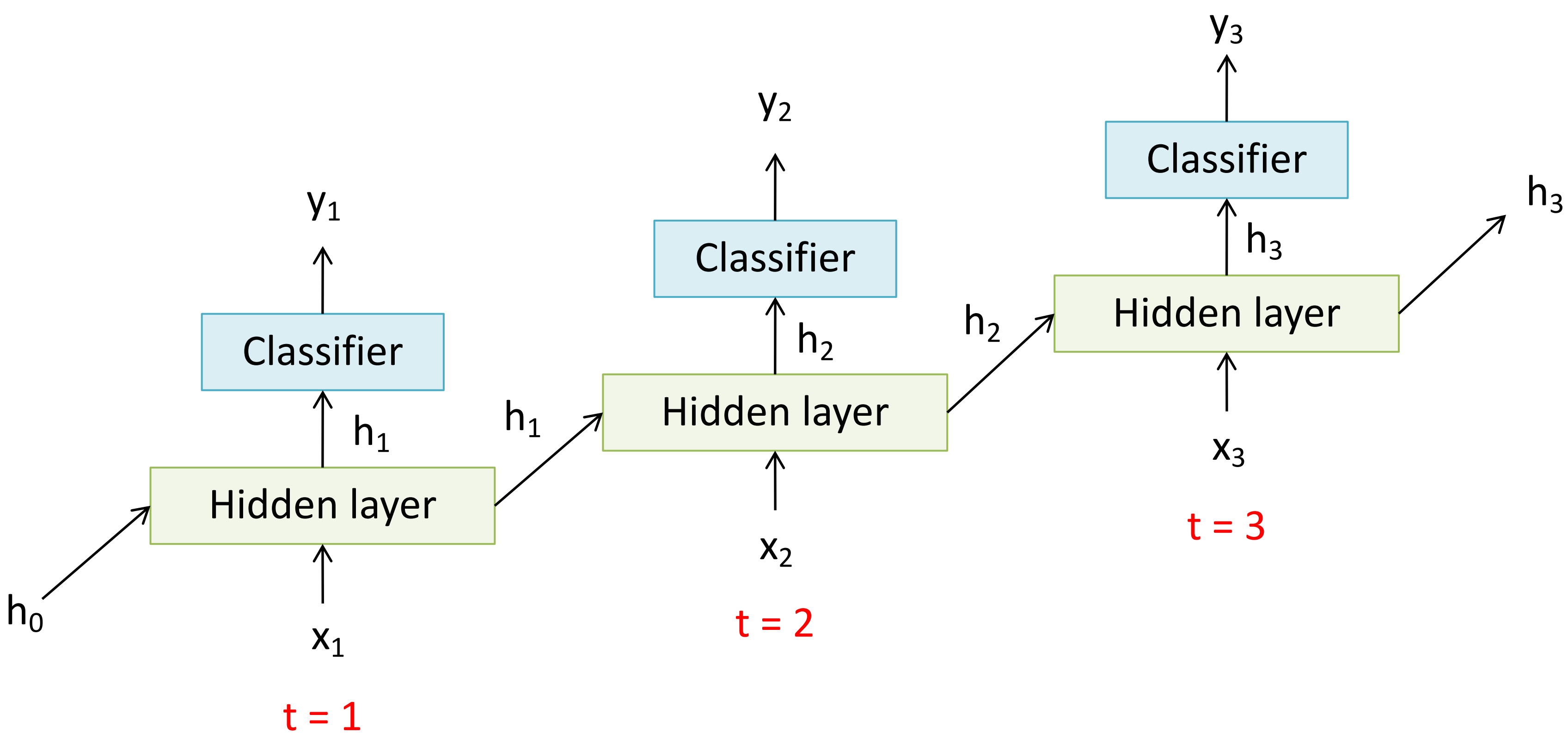


Recurrence:

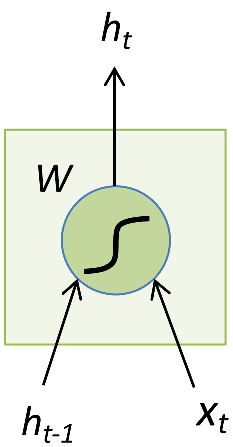
$$h_t = f_W(x_t, h_{t-1})$$

new state | function of W | input at time t | old state

Unrolling the RNN



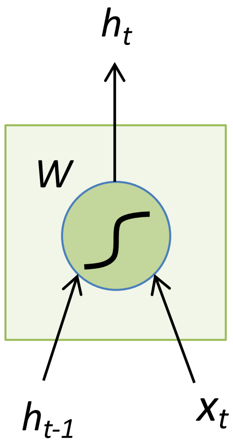
Vanilla RNN Cell



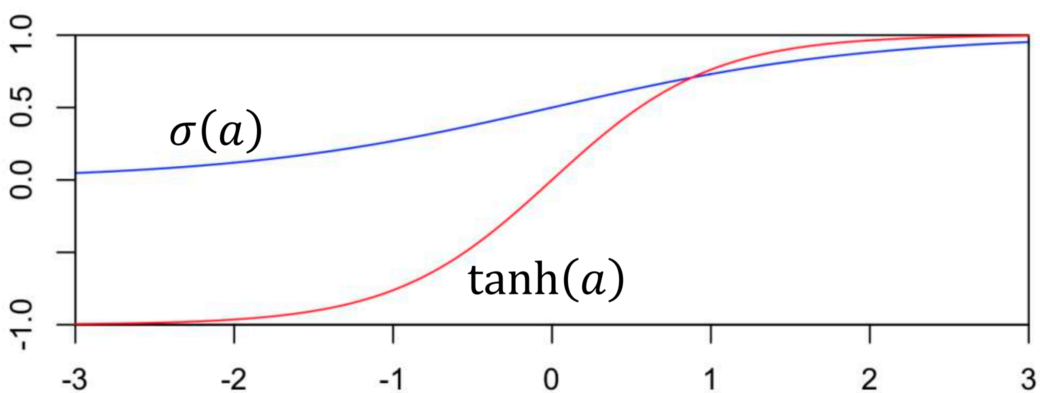
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

J. Elman, Finding structure in time, Cognitive science 14(2), pp. 179–211, 1990

Vanilla RNN Cell



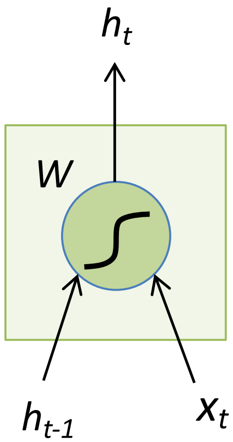
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$



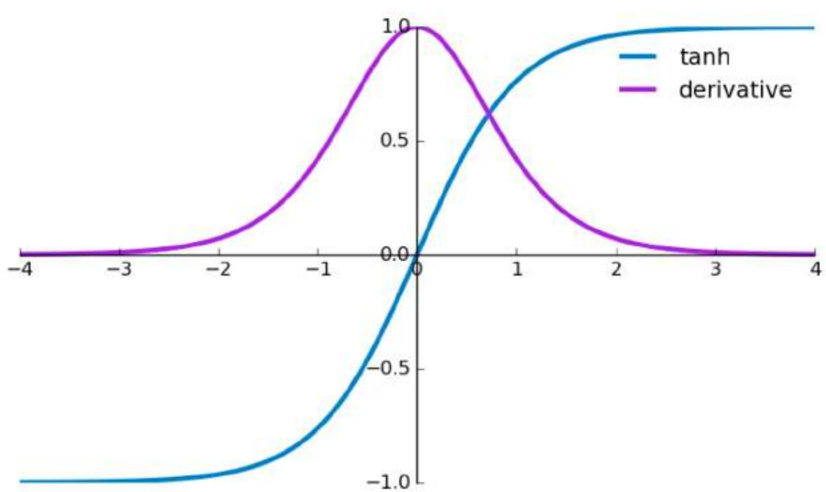
$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$
$$= 2\sigma(2a) - 1$$

Image source

Vanilla RNN Cell



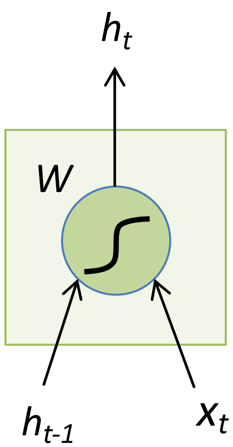
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$



$$\frac{d}{da}\tanh(a) = 1 - \tanh^2(a)$$

Image source

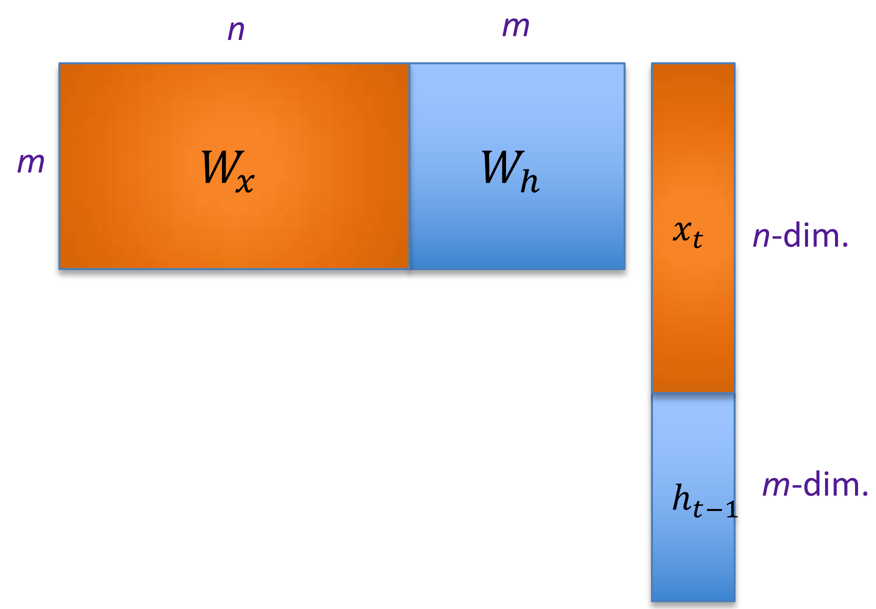
Vanilla RNN Cell



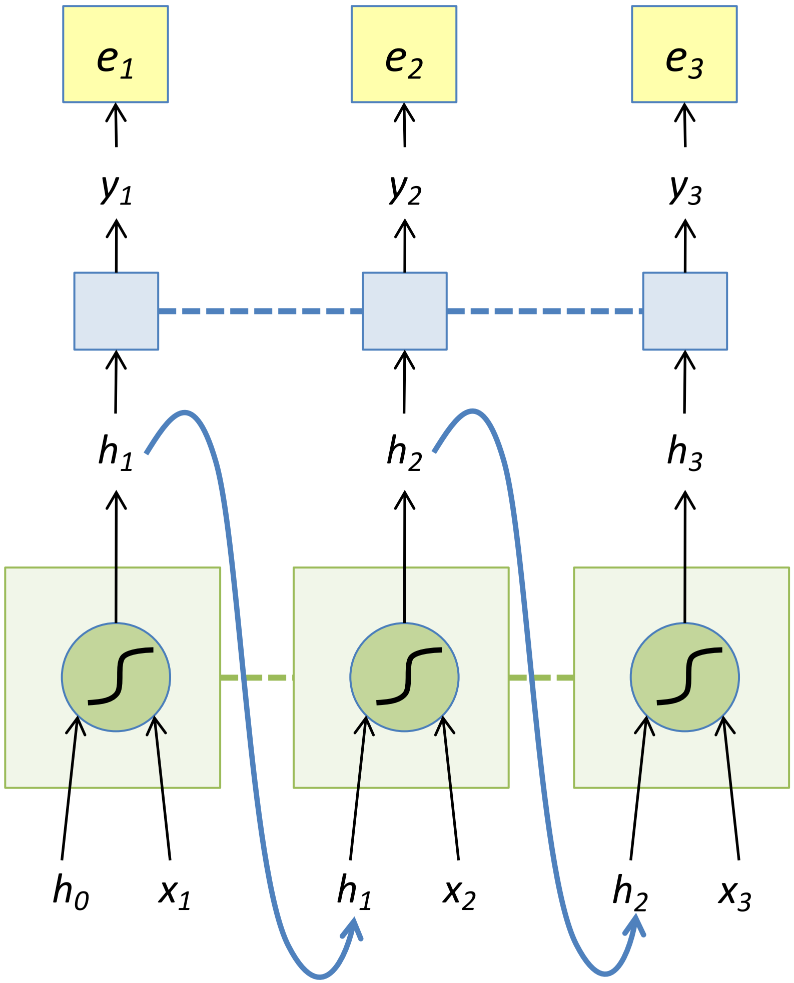
$$
\begin{aligned}
h_t &= f_W(x_t, h_{t-1}) \\
&= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \\
&= \tanh(W_x x_t + W_h h_{t-1})
\end{aligned}
$$

Q: Why not using ReLU?

- Training is unstable
- Need good initialization and careful training [Le, Jaitly, Hinton]



RNN Forward Pass



$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

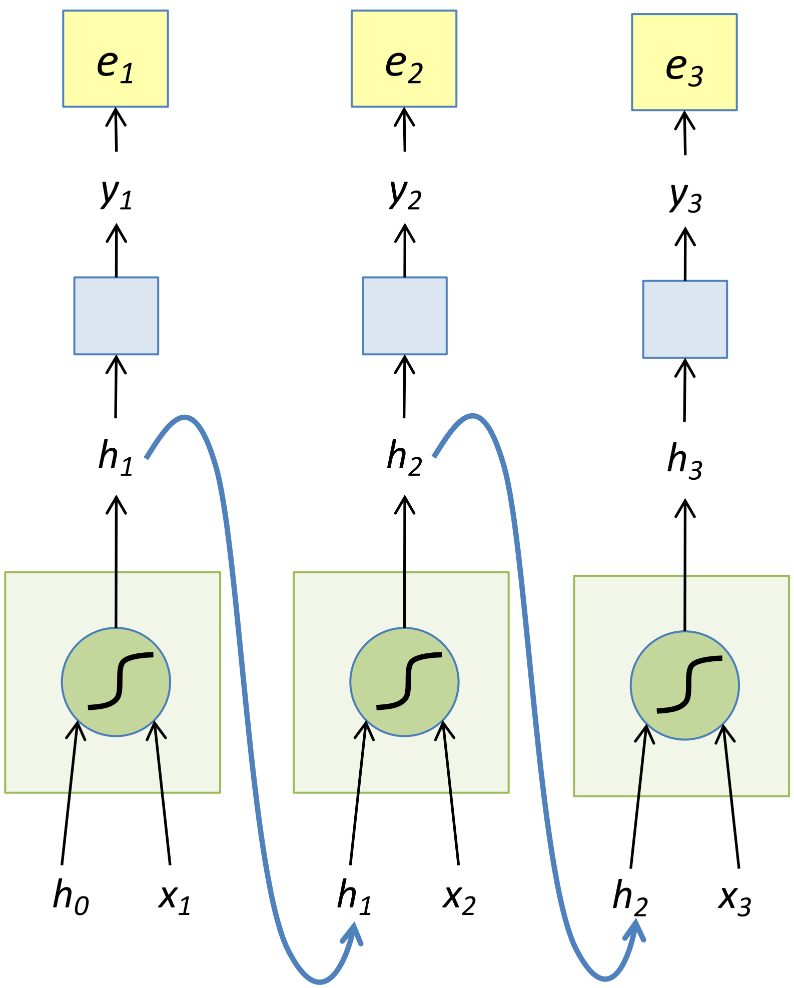
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

------ shared weights

Backpropagation Through Time (BPTT)

- Most common method used to train RNNs
- The unfolded RNN = one big feed-forward network that accepts the whole time series as input
- Gradients are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights

Unfolded RNN Forward Pass

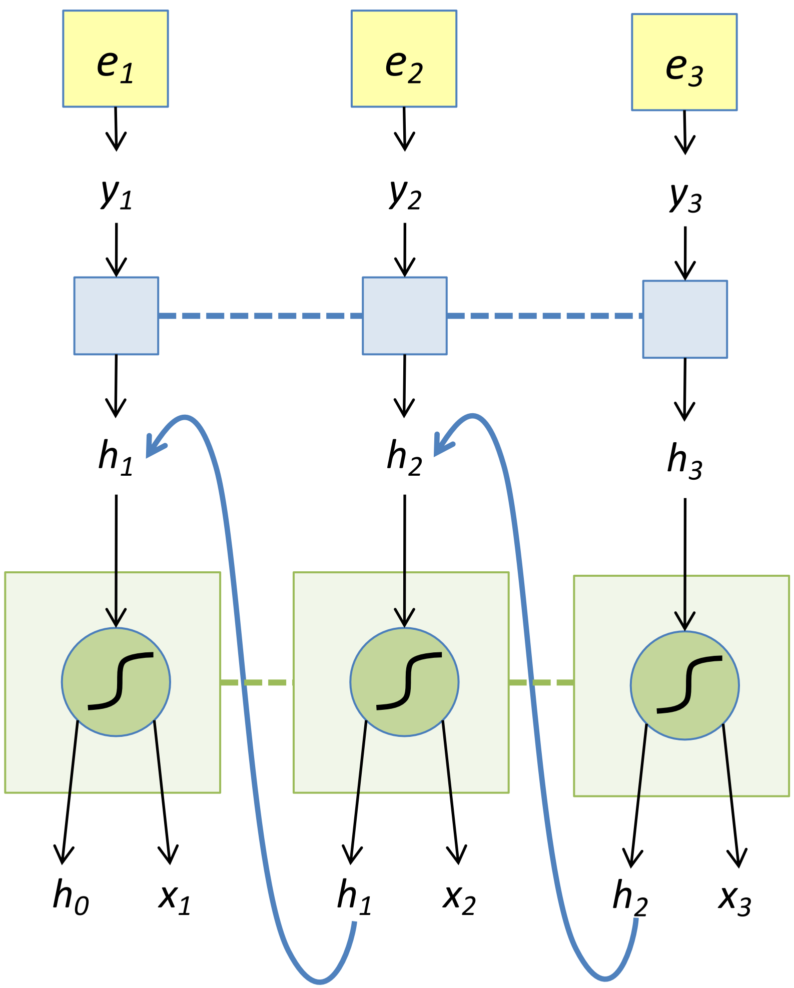


$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

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

Unfolded RNN Backward Pass



$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

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

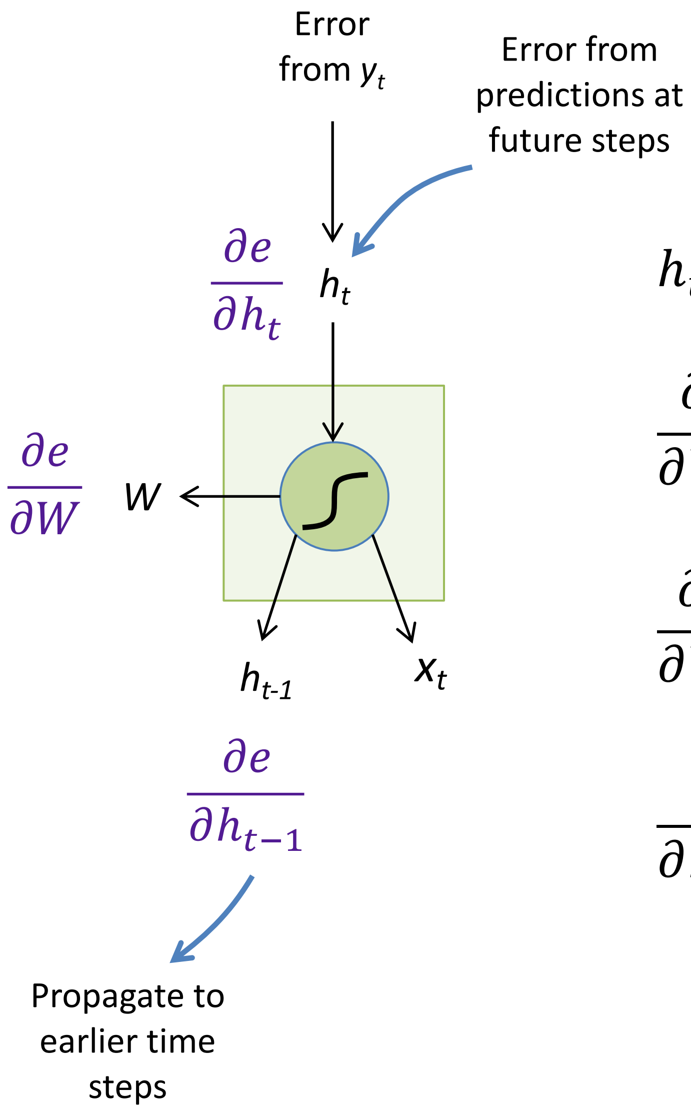
------ Averaging gradients

Backpropagation Through Time (BPTT)

- Most common method used to train RNNs
- The unfolded RNN = one big feed-forward network that accepts the whole time series as input
- Gradients are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights
- In practice, *truncated* BPTT is used: run the RNN forward k_1 time steps, propagate backward k_2 time steps
  - Bucketing based on the length of the training sequences

https://machinelearningmastery.com/gentle-introduction-backpropagation-time/
http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf

RNN Backward Pass



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

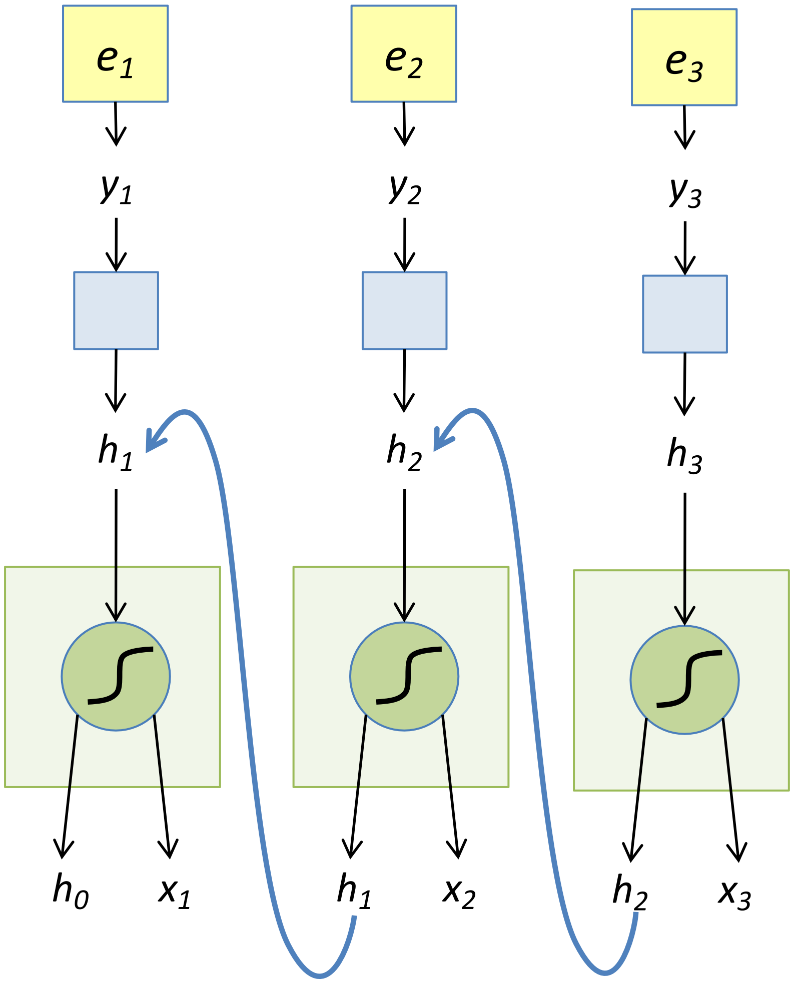
Element-wise multiplication

$$\frac{\partial e}{\partial W_h} = \frac{\partial e}{\partial h_t} \boxed{\odot} \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) h_{t-1}^T$$

$$\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) x_t^T$$

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

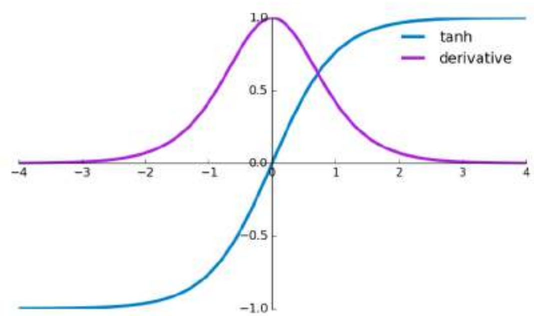
RNN Backward Pass



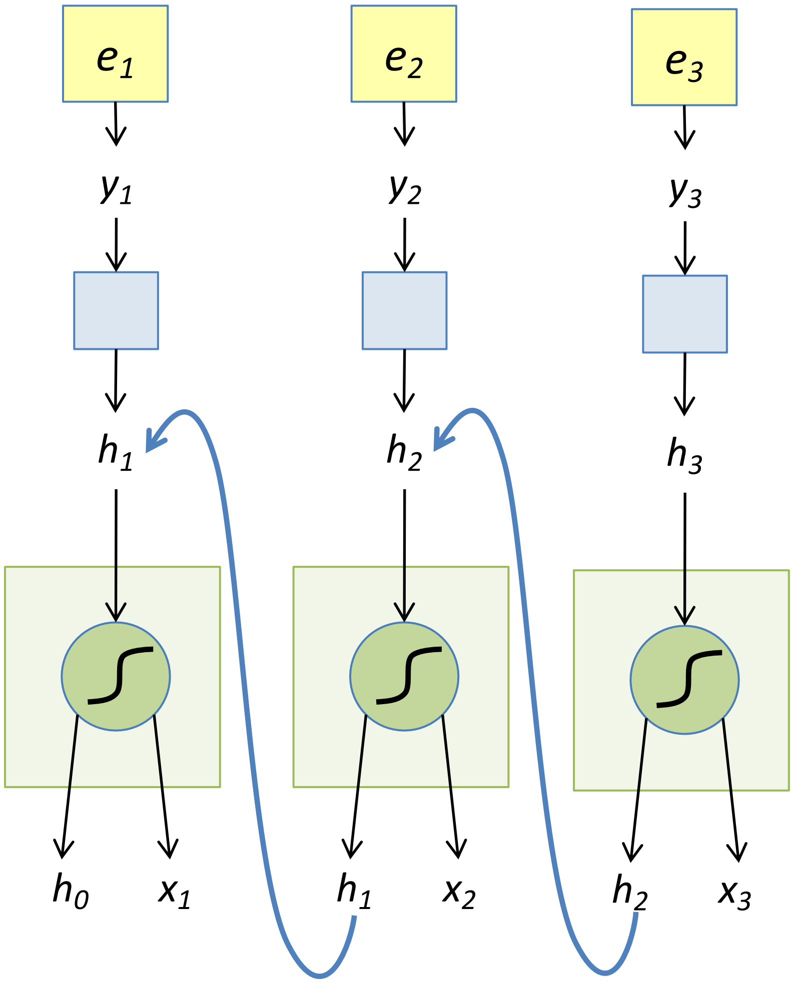
Consider $\frac{\partial e_n}{\partial h_k}$ for $k \ll n$

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T\left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

Large tanh activations will give small gradients



RNN Backward Pass



$$\frac{\partial e}{\partial h_{t-1}} = W_h^T\big(1 - \tanh^2(W_x x_t + W_h h_{t-1})\big) \odot \frac{\partial e}{\partial h_t}$$

Gradients will vanish if largest singular value of W_h is less than 1

Consider $\frac{\partial e_n}{\partial h_k}$ for $k \ll n$

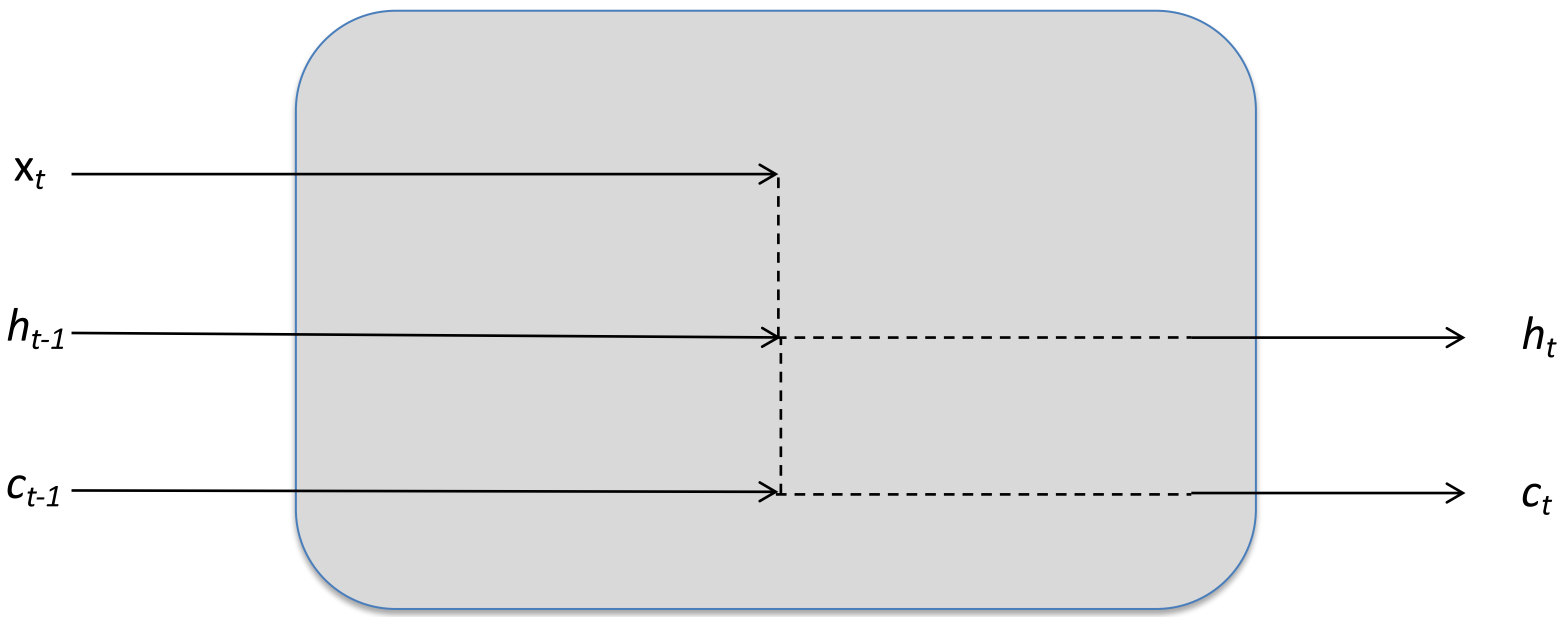
RNNs?

- Training can be time / memory consuming (unrolling produces huge feedforward models)

- Gradient vanishing (largest singular value of W_h<1)

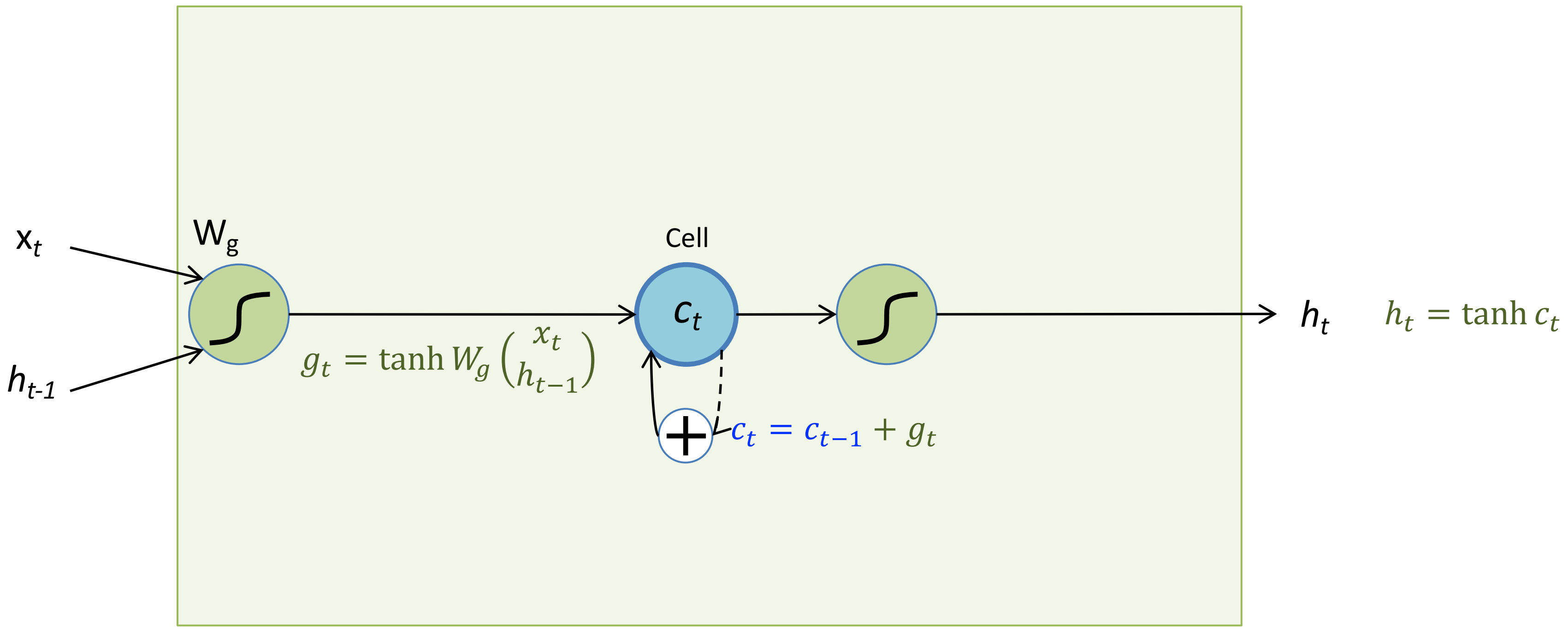
Long Short-Term Memory (LSTM)

- Add a *memory cell* that is not subject to matrix multiplication or squishing, thereby avoiding gradient decay



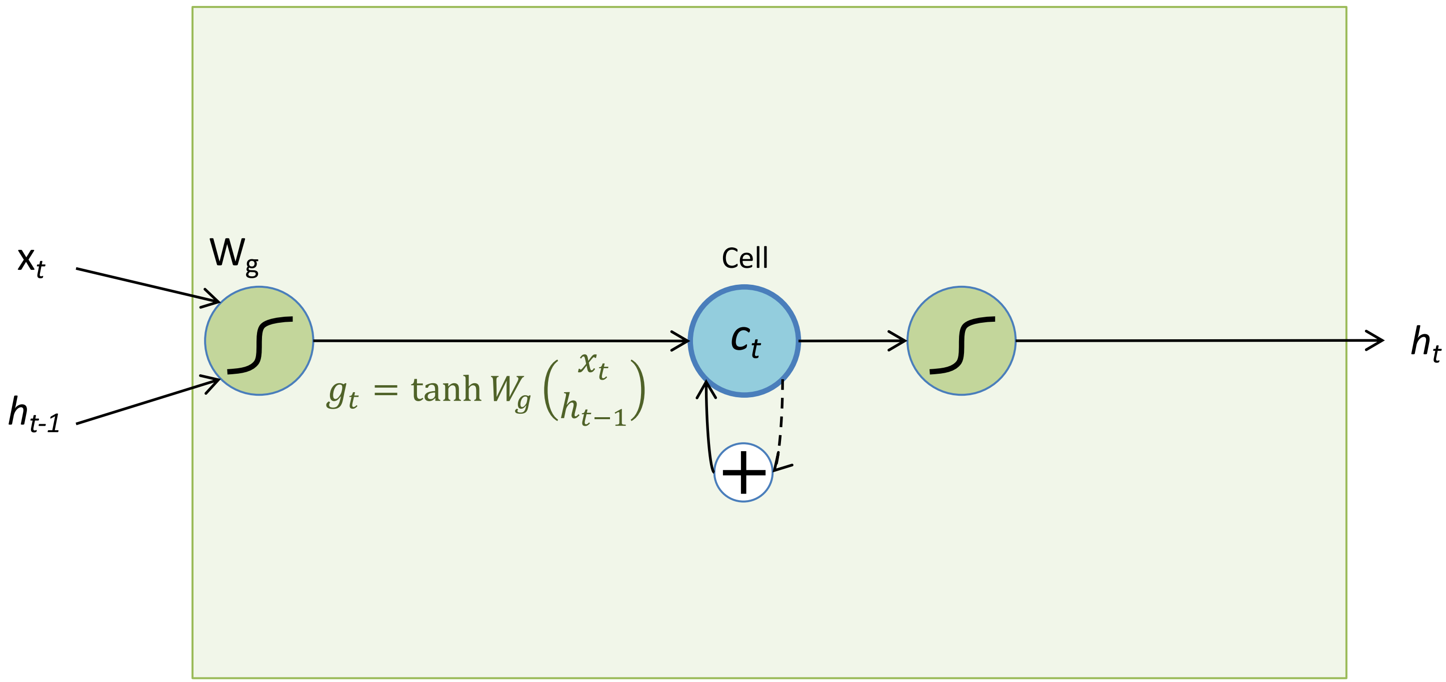
S. Hochreiter and J. Schmidhuber, [Long short-term memory](), Neural Computation 9 (8), pp. 1735–1780, 1997

The LSTM Cell

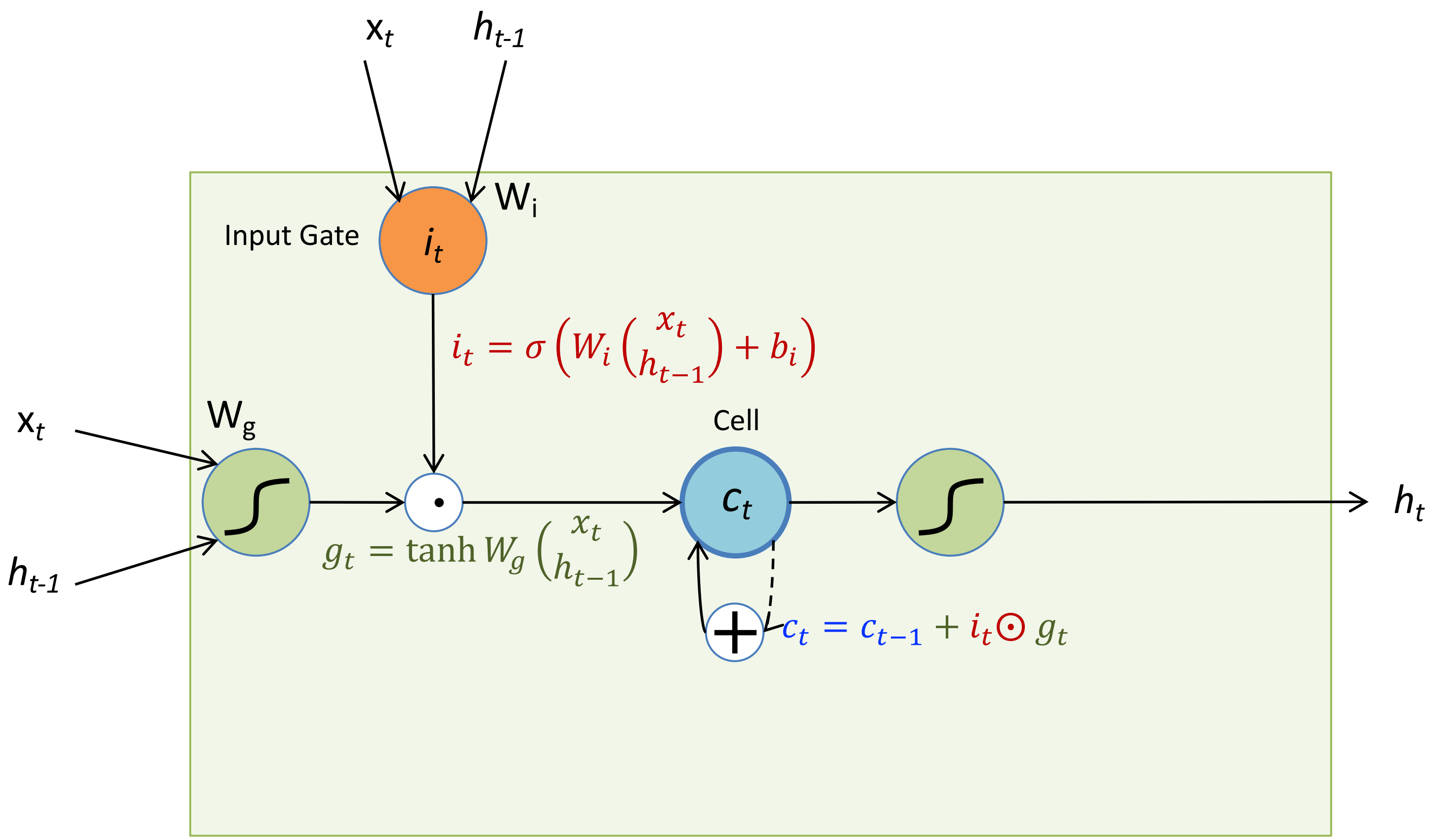


* Dashed line indicates time-lag

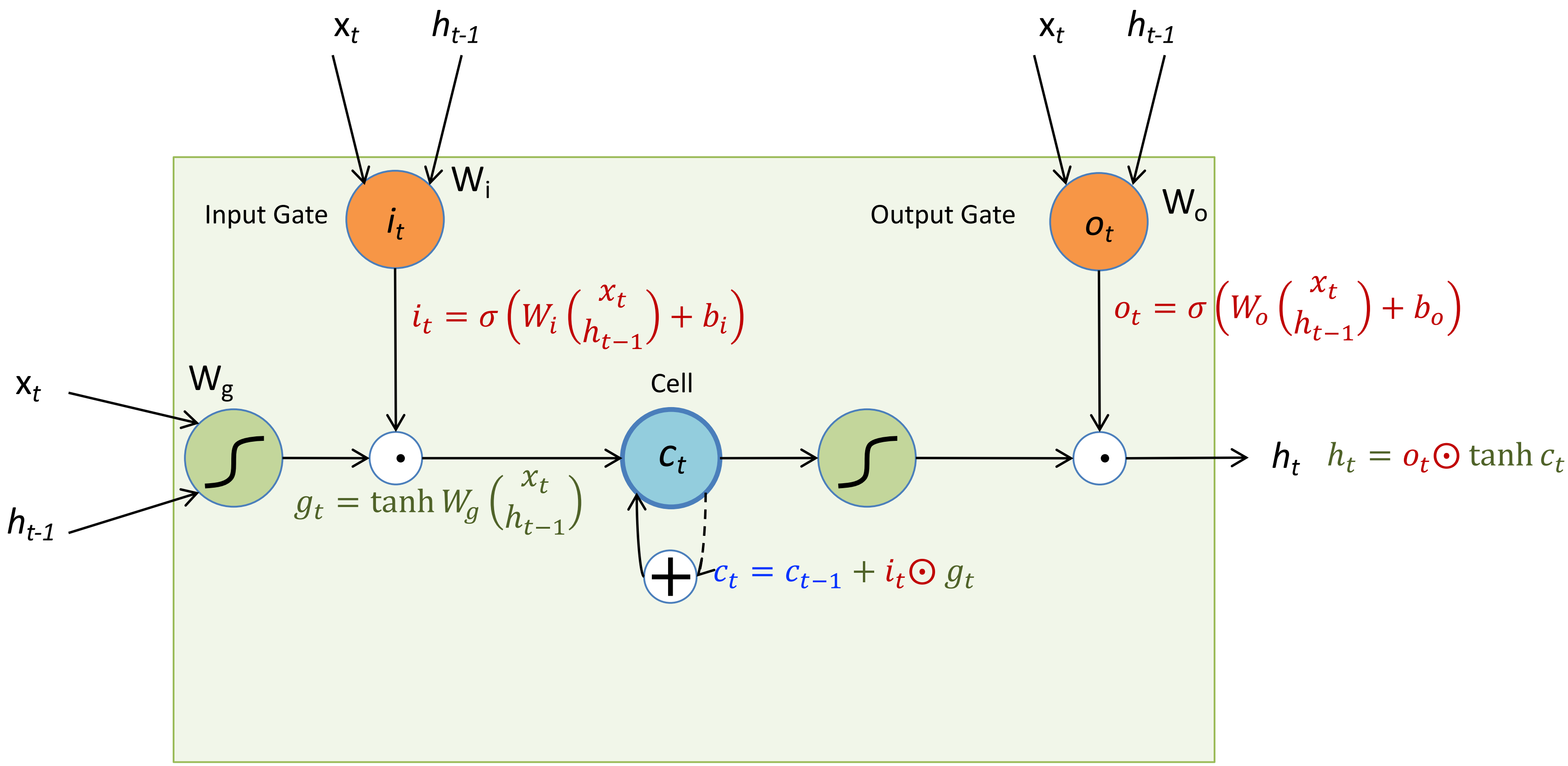
The LSTM Cell



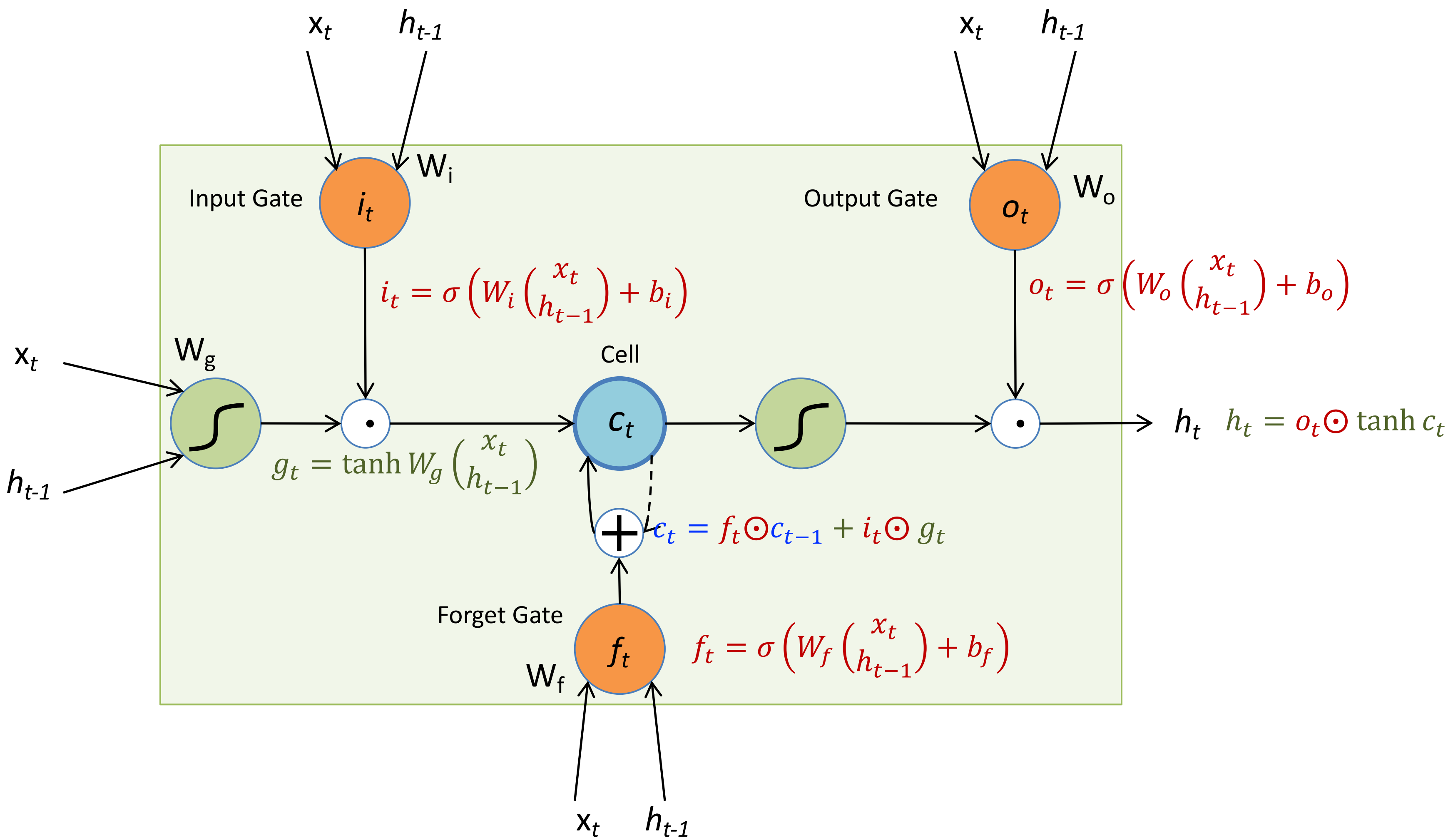
The LSTM Cell



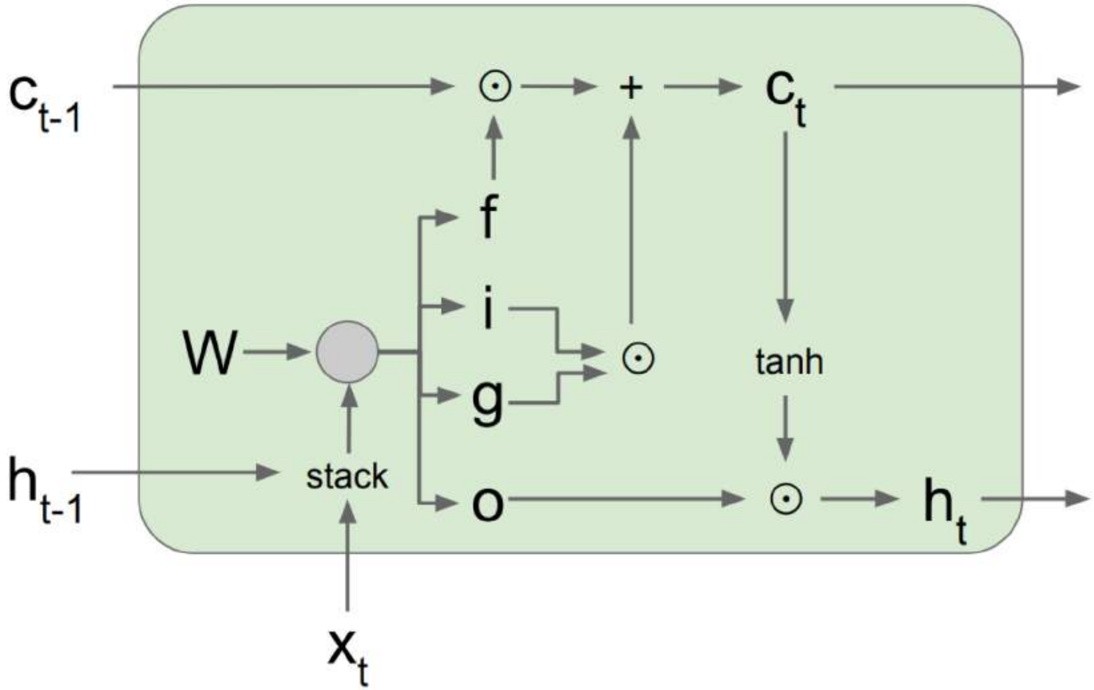
The LSTM Cell



The LSTM Cell



LSTM Forward Pass Summary



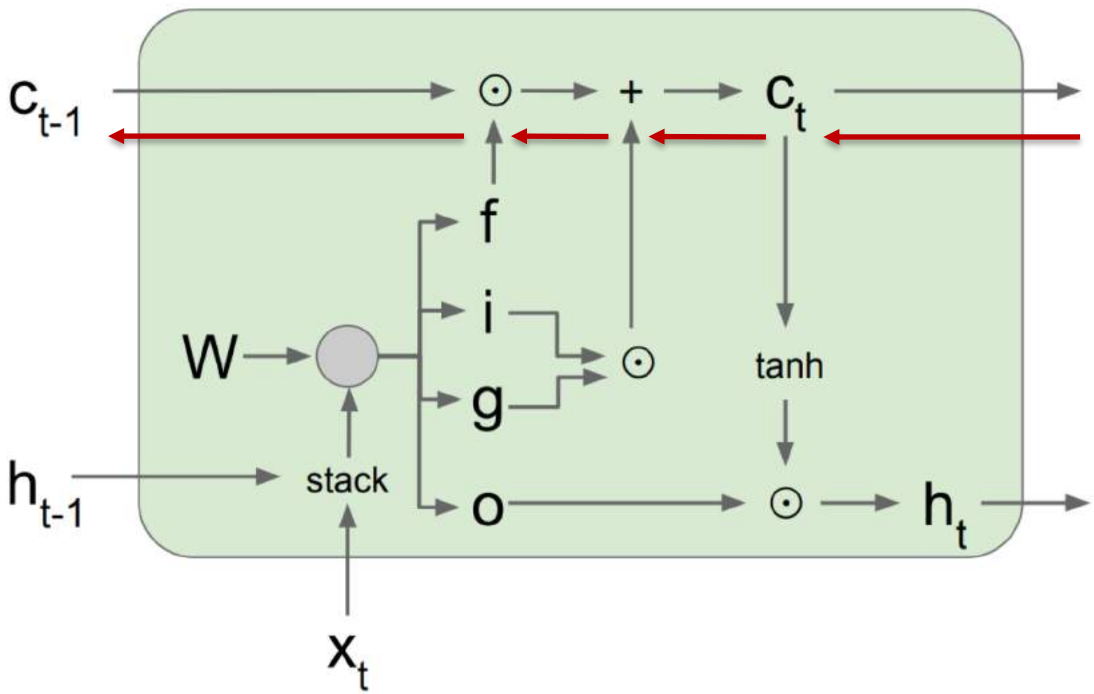
$$\begin{pmatrix} g_t \\ i_t \\ f_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} \begin{pmatrix} W_g \\ W_i \\ W_f \\ W_o \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh c_t$$

Figure source

LSTM Backward Pass

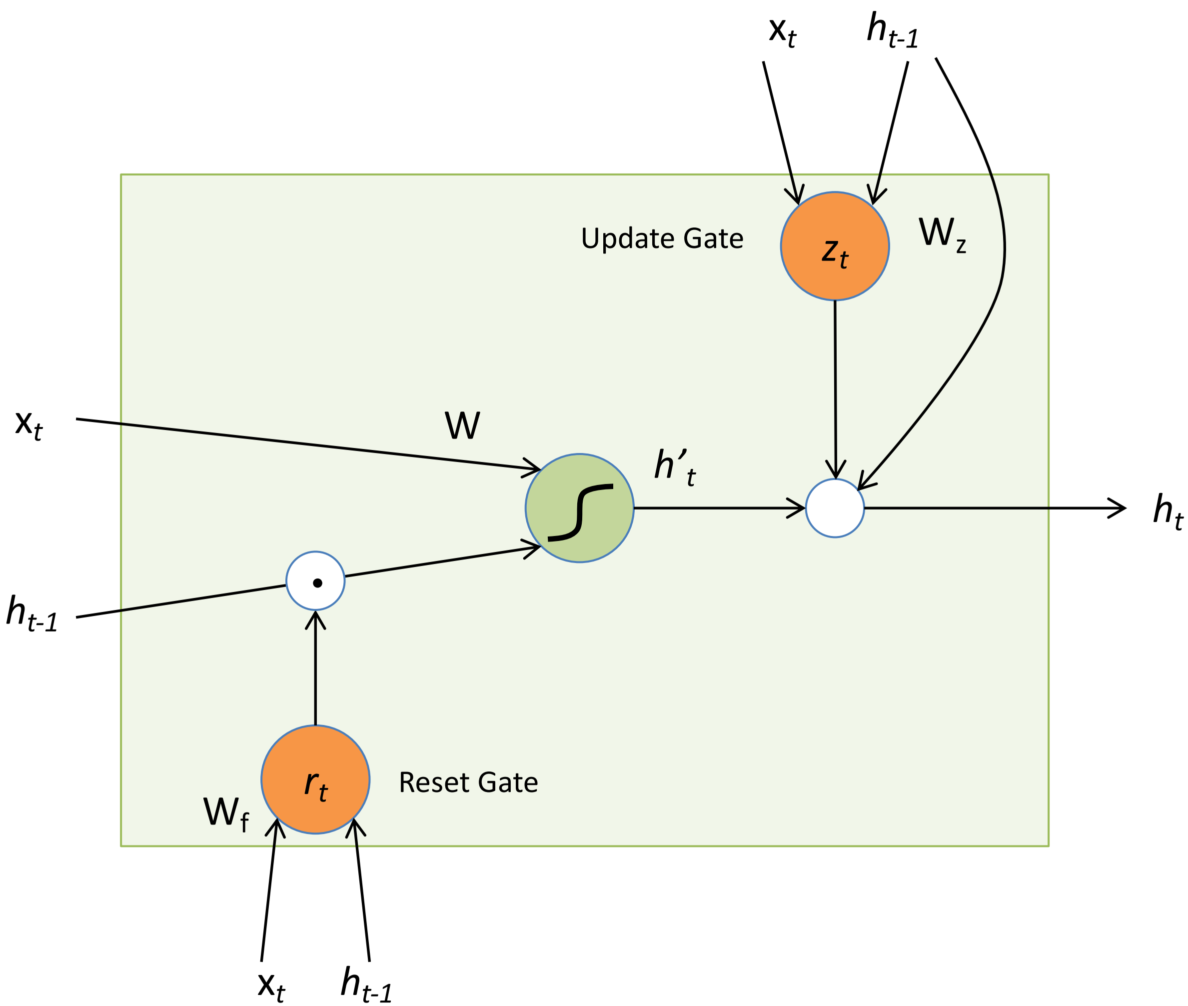


Gradient flow from c_t to c_{t-1} only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh

For complete details: [Illustrated LSTM Forward and Backward Pass]

[Figure source]

Gated Recurrent Unit (GRU)



- Get rid of separate cell state
- Merge "forget" and "output" gates into "update" gate

K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, [Learning phrase representations using RNN encoder-decoder for statistical machine translation](), ACL 2014

LSTMs?

- Training can be time / memory consuming (unrolling produces huge feedforward models)

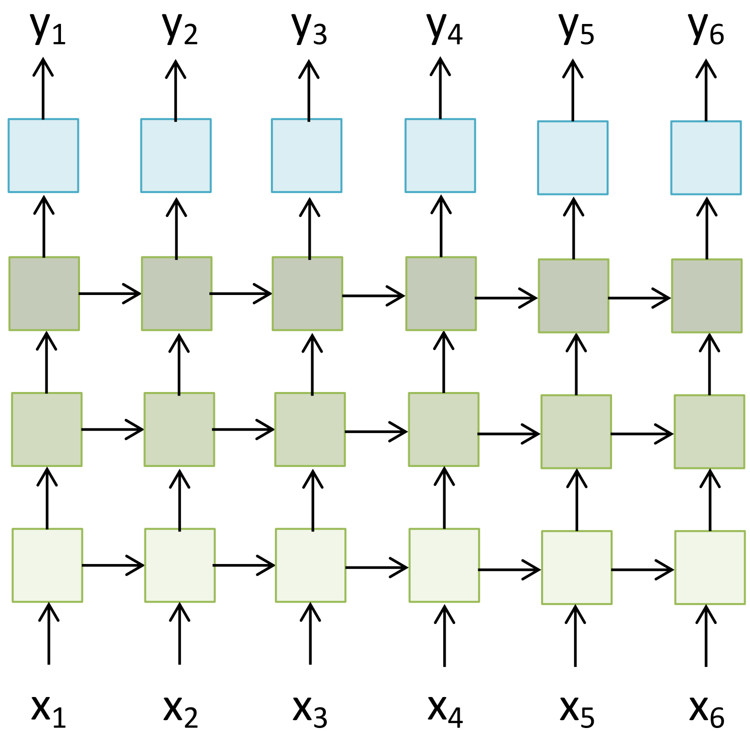
- ~~Gradient vanishing (largest singular value of W_h<1)~~
- More complicated achitecture

- Very successful in NLP (+transformer) and Vision

- BPTT with K-step unrolling → can be replaced by a feedforward model? [Miller, Hardt]

Multi-layer RNNs

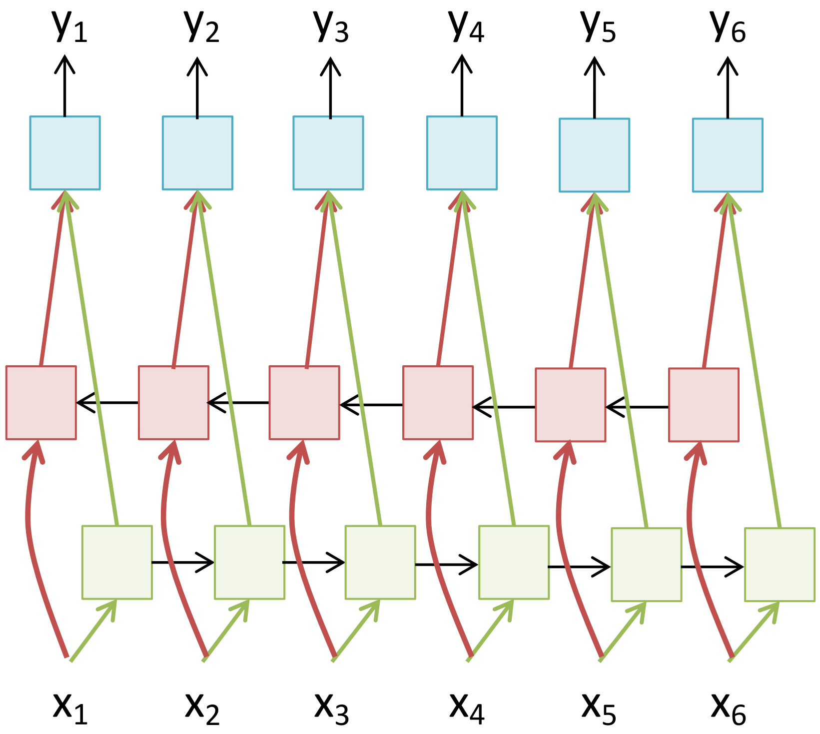
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction



- Popular in speech recognition / NLP

Useful Resources / References

- http://cs231n.stanford.edu/slides/winter1516_lecture10.pdf
- http://www.cs.toronto.edu/~rgrosse/csc321/lec10.pdf
- http://slazebni.cs.illinois.edu/fall18/lec15_rnn.pdf

- R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013
- S. Hochreiter, and J. Schmidhuber, Long short-term memory, Neural computation, 1997 9(8), pp.1735-1780
- F.A. Gers, and J. Schmidhuber, Recurrent nets that time and count, IJCNN 2000
- K. Greff , R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, LSTM: A search space odyssey, IEEE transactions on neural networks and learning systems, 2016
- K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014
- R. Jozefowicz, W. Zaremba, and I. Sutskever, An empirical exploration of recurrent network architectures, JMLR 2015