Recurrent Neural Networks

Adapted from Arun Mallya
Source: Part 1, Part 2
Slides adapted from L. Lazebnik, P. Krahen

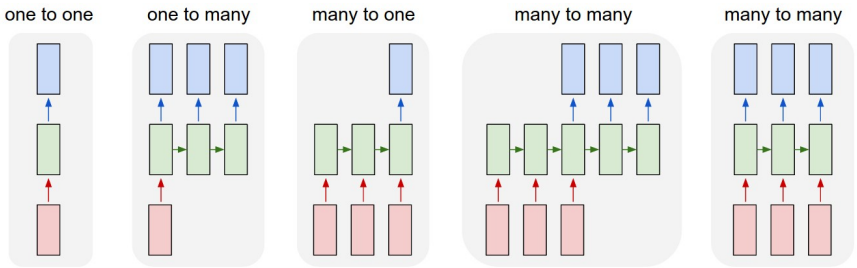
Outline

- Sequential prediction problems
- Vanilla RNN unit
  - Forward and backward pass
  - Back-propagation through time (BPTT)
- Long Short-Term Memory (LSTM) unit
- Gated Recurrent Unit (GRU)
- Applications

Sequential prediction tasks

- So far, we focused mainly on prediction problems with fixed-size inputs and outputs
- But what if the input and/or output is a variable-length sequence?

Types of problems

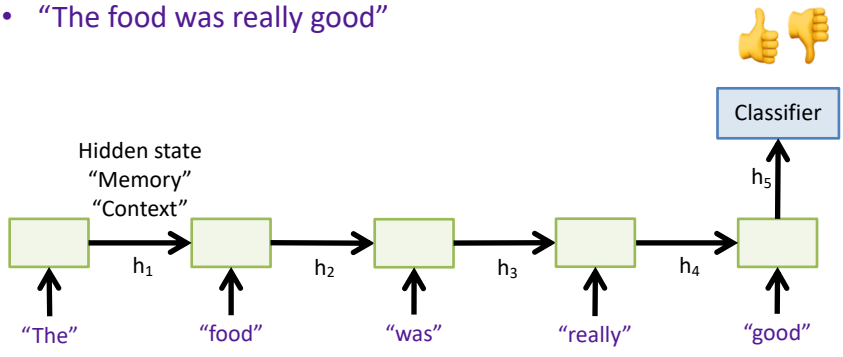


Text classification

- **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)

Language Modeling

Language Modeling

Character RNN

http://tommymullaney.com/projects/char-rnn-gchat

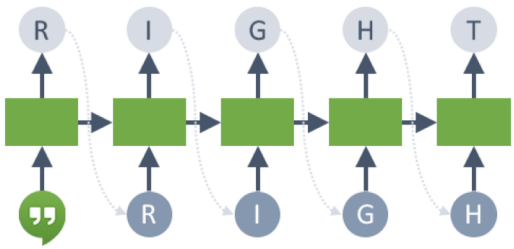
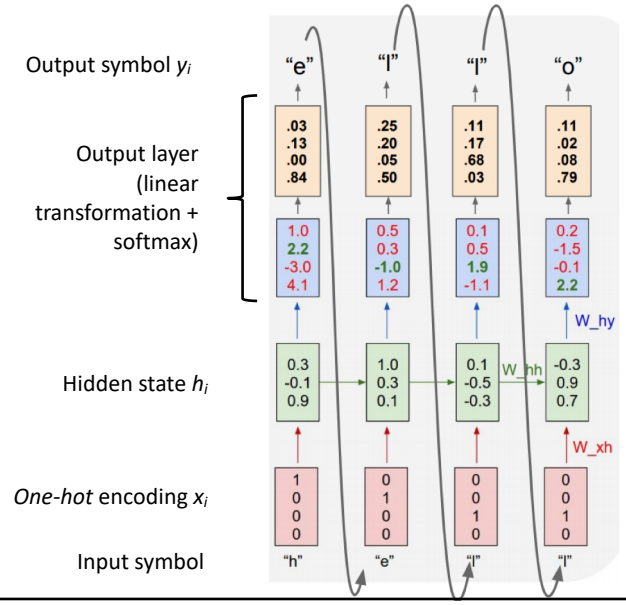


Image source

Character RNN



$$p(y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}) \approx \prod_{i=1}^{n} P_W(y_i | h_i)$$

Image source

Character RNN

- Generating paint colors

Clardic Fug 112 113 84
Snowbonk 201 199 165
Catbabel 97 93 68
Bunflow 190 174 155
Ronching Blue 121 114 125
Bank Butt 221 196 199
Caring Tan 171 166 170
Stargoon 233 191 141
Sink 176 138 110
Stummy Beige 216 200 185
Dorkwood 61 63 66
Flower 178 184 196

Sand Dan 201 172 143
Grade Bat 48 94 83
Light Of Blast 175 150 147
Grass Bat 176 99 108
Sindis Poop 204 205 194
Dope 219 209 179
Testing 156 101 106
Stoner Blue 152 165 159
Burble Simp 226 181 132
Stanky Bean 197 162 171
Turdly 190 164 116

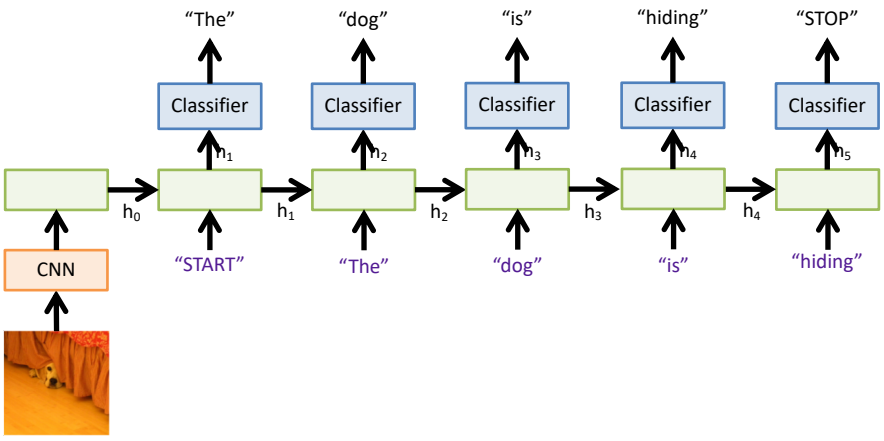
http://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network

Image Caption Generation

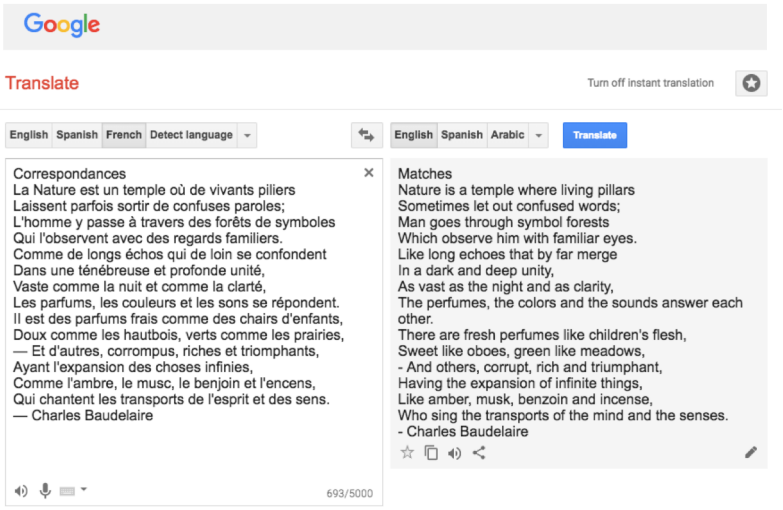
- Given an image, produce a sentence describing its contents

"The dog is hiding"

Image Caption Generation



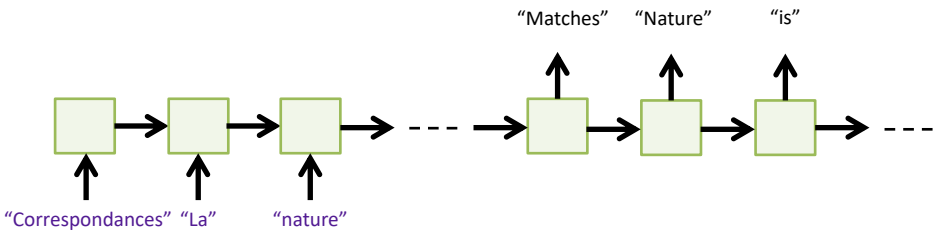
Machine translation



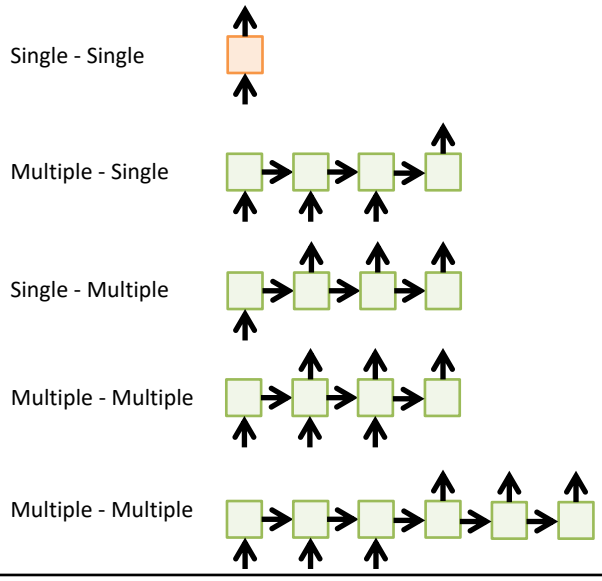
https://translate.google.com/

Machine translation

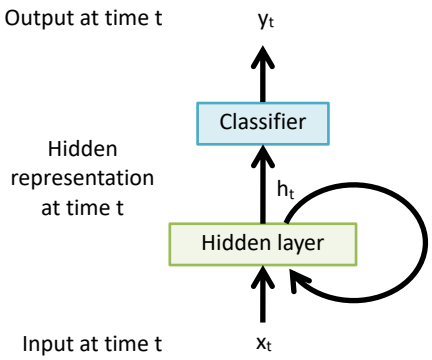
- Multiple input – multiple output (or sequence to sequence)



Summary: Input-output scenarios



Recurrent Neural Network (RNN)

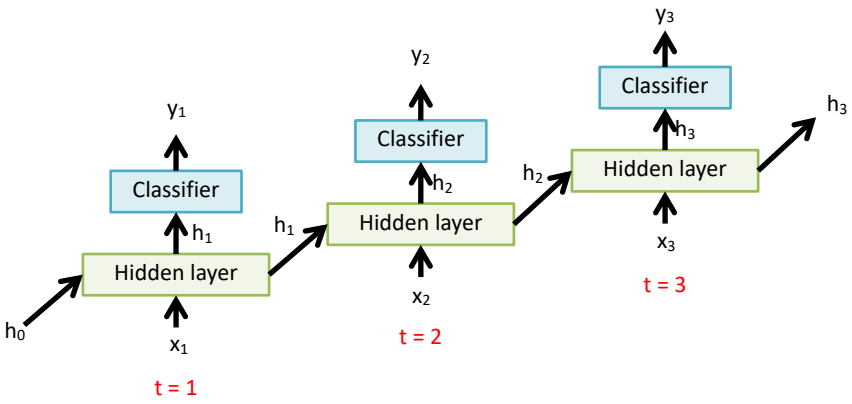


Recurrence:

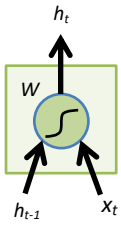
$$h_t = f_W(x_t, h_{t-1})$$

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

Unrolling the RNN



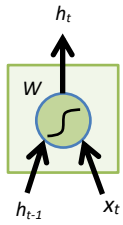
Vanilla RNN Cell



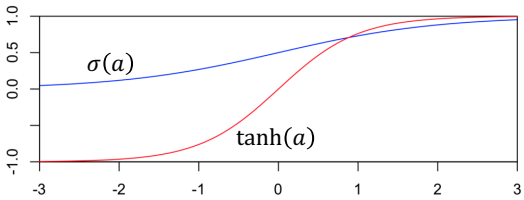
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

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

Vanilla RNN Cell



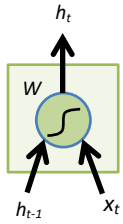
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$



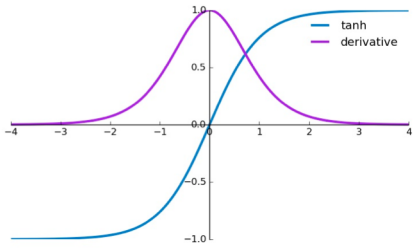
$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$
$$= 2\sigma(2a) - 1$$

Image source

Vanilla RNN Cell



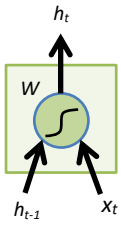
$$h_t = f_W(x_t, h_{t-1})$$
$$= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$



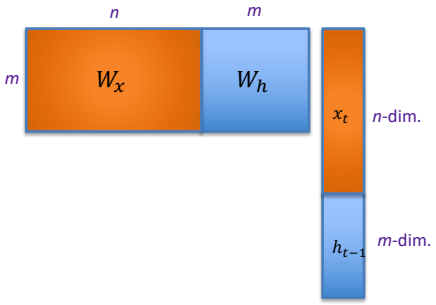
$$\frac{d}{da}\tanh(a) = 1 - \tanh^2(a)$$

Image source

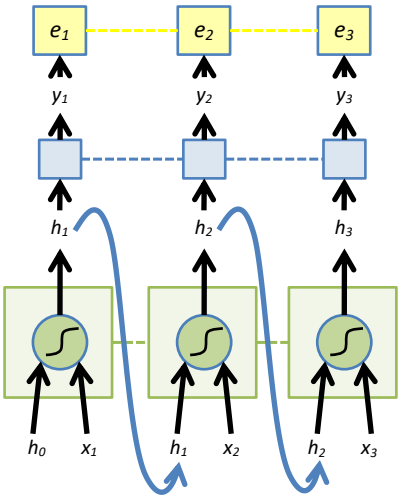
Vanilla RNN Cell



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



RNN Forward Pass



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

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

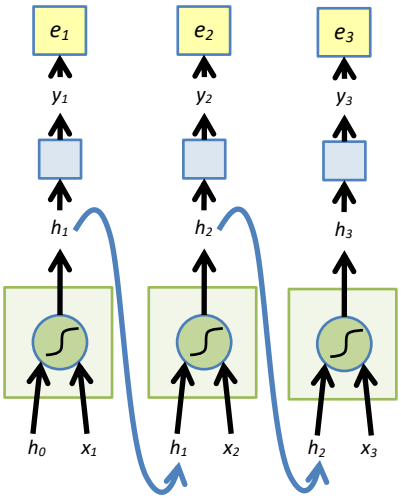
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

------ shared weights

Backpropagation Through Time (BPTT)

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

Unfolded RNN Forward Pass

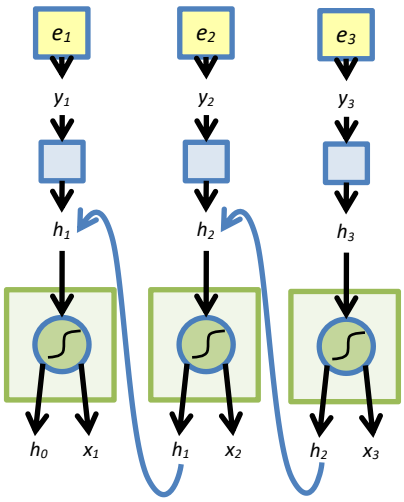


$$e_t = -\log(y_t(GT_t))$$

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

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

Unfolded RNN Backward Pass



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

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

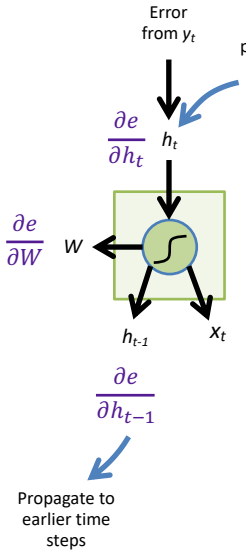
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

Backpropagation Through Time (BPTT)

- Most common method used to train RNNs
- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
- The weight updates 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 for k_2 time steps

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

RNN Backward Pass



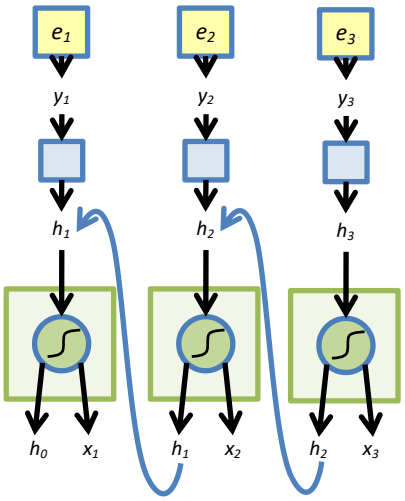
$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

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

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

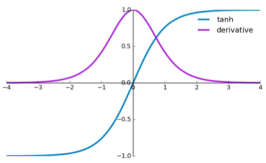
$$\frac{\partial e}{\partial h_{t-1}} = W_h^T \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

RNN Backward Pass



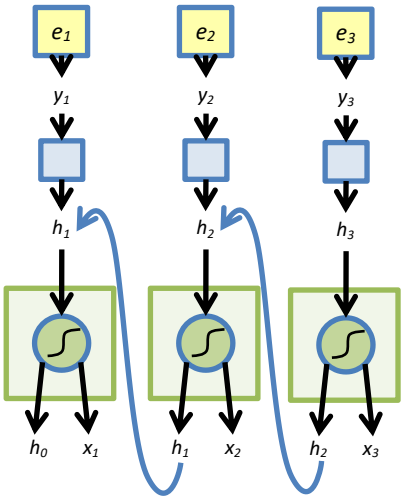
$$\frac{\partial e}{\partial h_{t-1}} = W_h^T \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

Large tanh activations will give small gradients



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

RNN Backward Pass



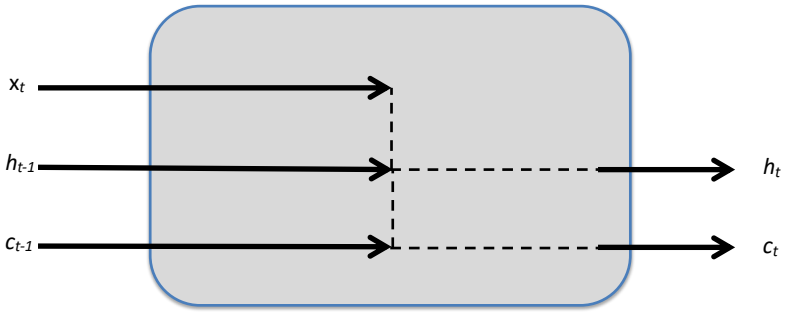
$$\frac{\partial e}{\partial h_{t-1}} = W_h^T\left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

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

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

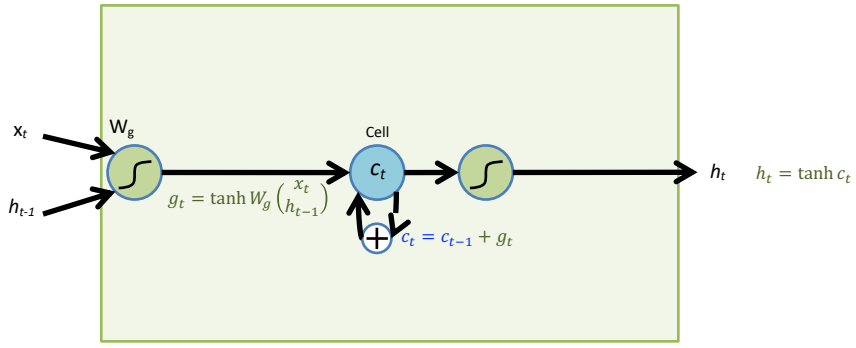
Long Short-Term Memory (LSTM)

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



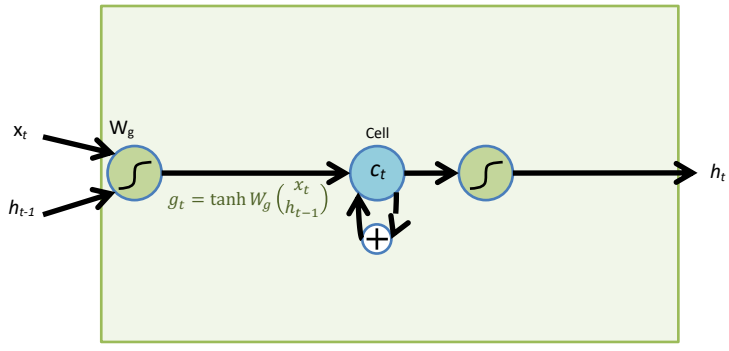
S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9 (8), pp. 1735–1780, 1997

The LSTM Cell

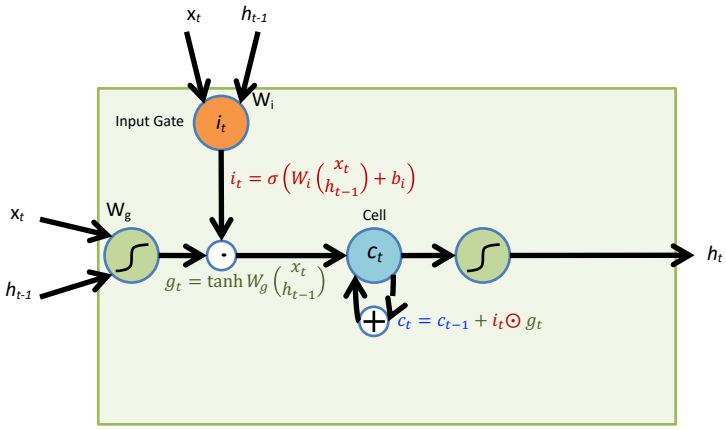


* Dashed line indicates time-lag

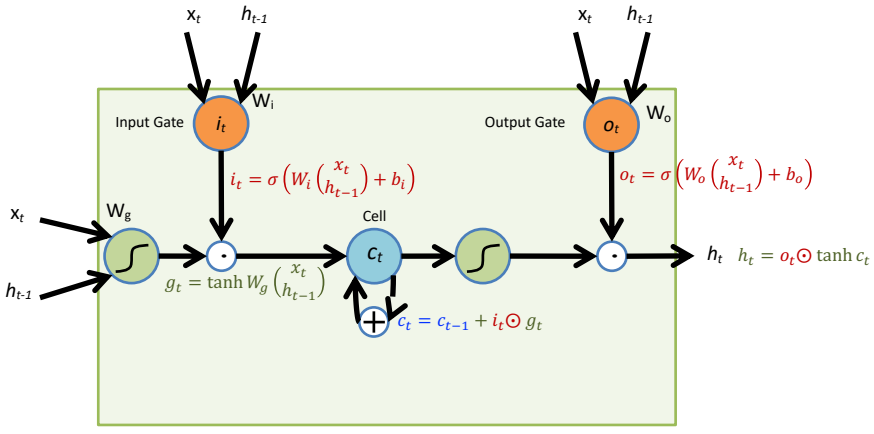
The LSTM Cell



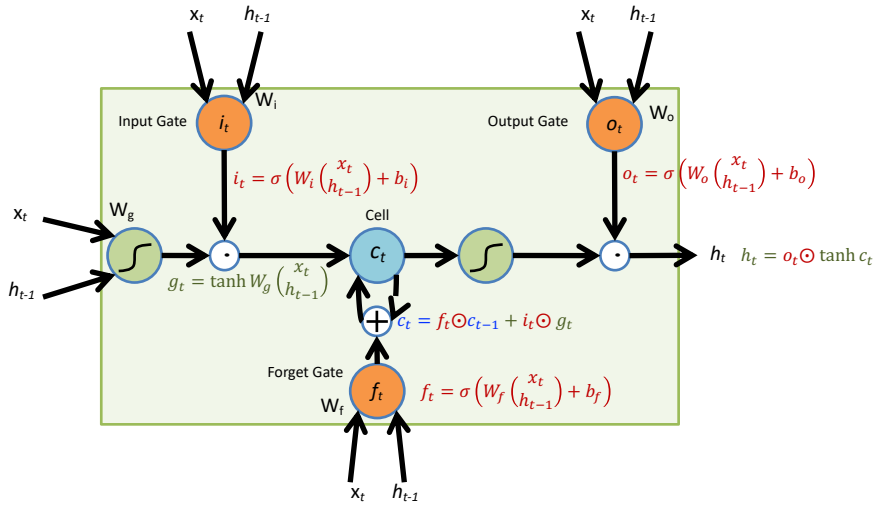
The LSTM Cell



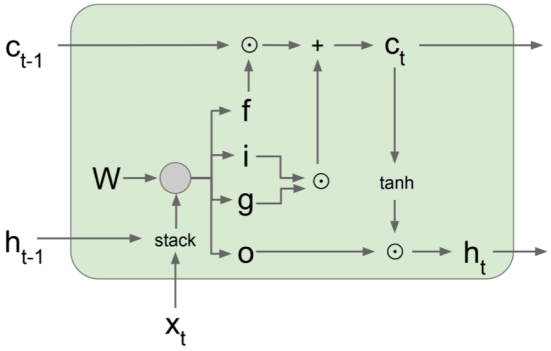
The LSTM Cell



The LSTM Cell



LSTM Forward Pass Summary



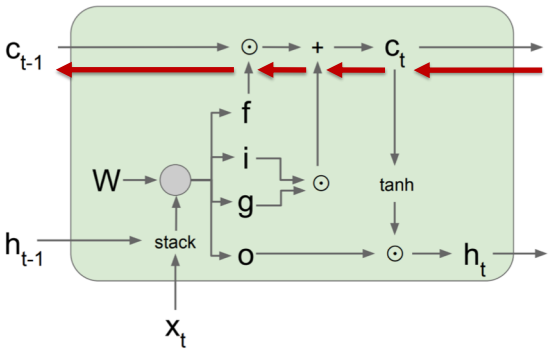
$$\begin{pmatrix} g_t \\ i_t \\ f_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} \begin{pmatrix} W_g \\ W_i \\ W_f \\ W_o \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

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

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

Figure source

LSTM Backward Pass

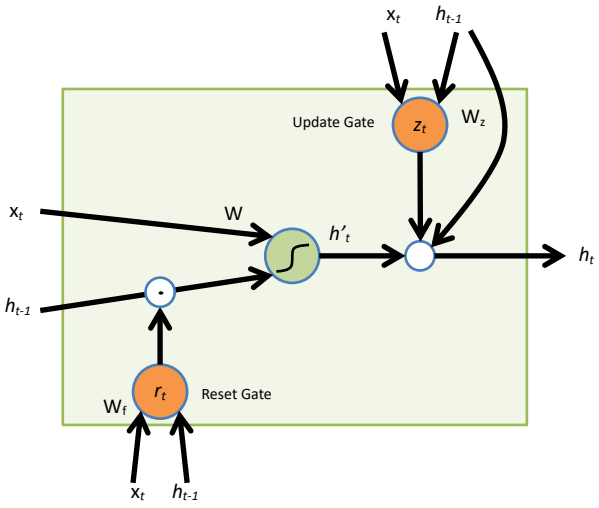


Gradient flow from c_t to c_{t-1} only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh

For complete details: Illustrated LSTM Forward and Backward Pass

Figure source

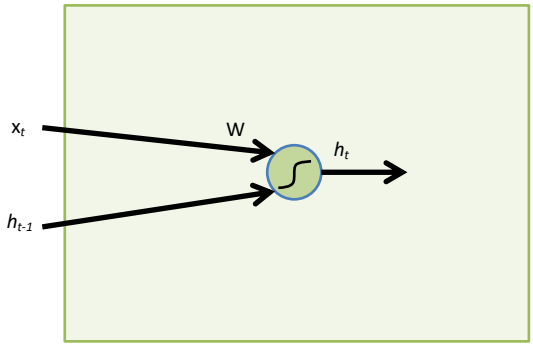
Gated Recurrent Unit (GRU)



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

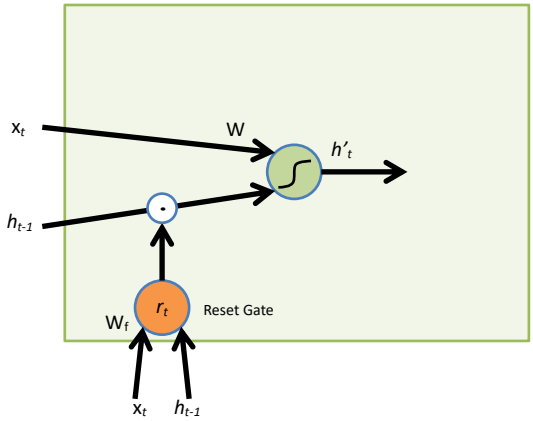
K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014

Gated Recurrent Unit (GRU)



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

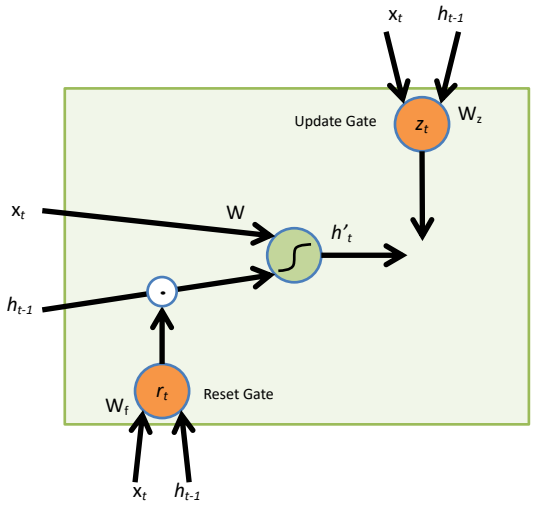
Gated Recurrent Unit (GRU)



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_t\right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}$$

Gated Recurrent Unit (GRU)

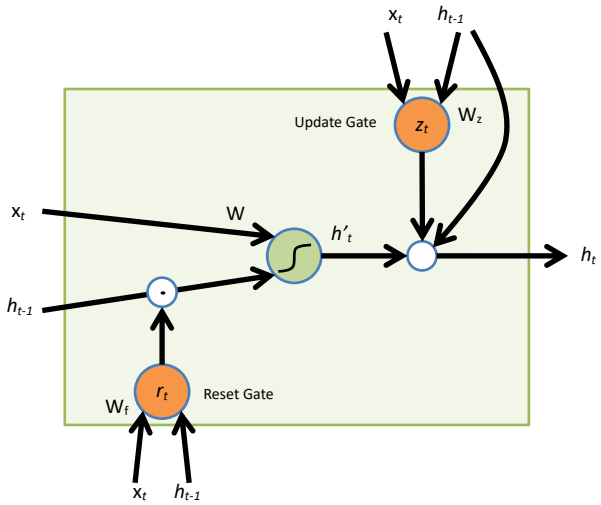


$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_t\right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z\right)$$

Gated Recurrent Unit (GRU)



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_t\right)$$

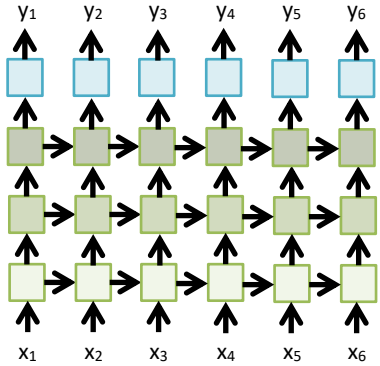
$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$$

Multi-layer RNNs

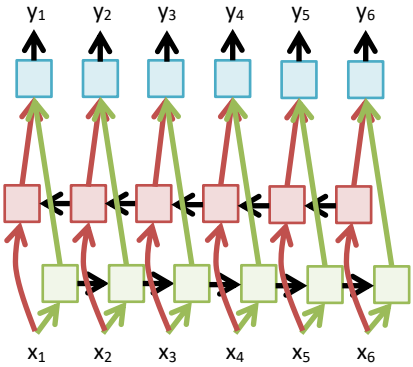
- We can of course design RNNs with multiple hidden layers



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

Bi-directional RNNs

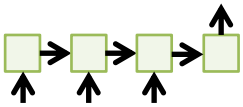
- RNNs can process the input sequence in forward and in the reverse direction



- Popular in speech recognition

Use Cases

Multiple input – Single output



Sequence Classification

Single - Multiple

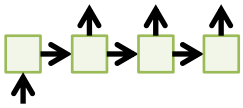


Image Captioning

Multiple - Multiple

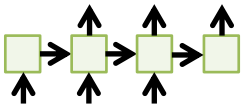
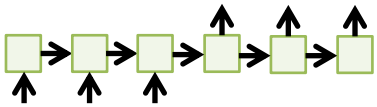


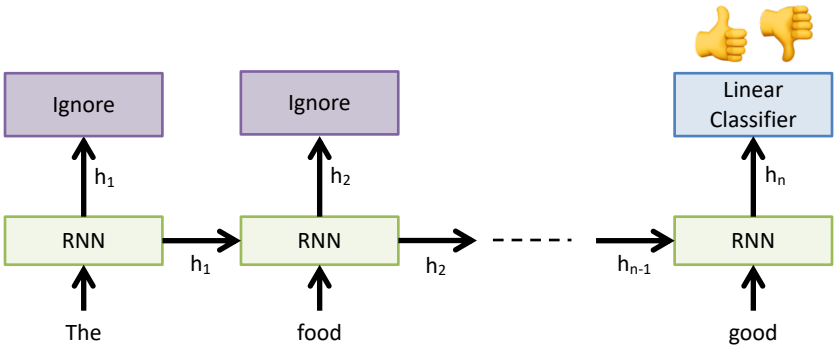
Image Captioning

Multiple - Multiple

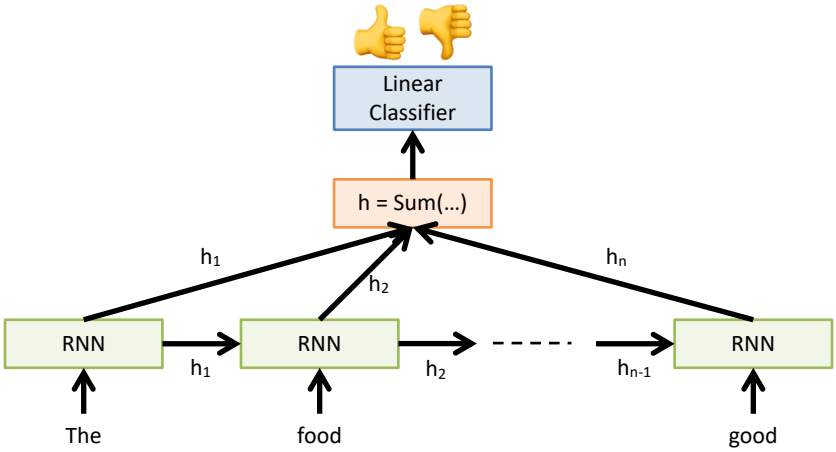


Translation

Sequence Classification

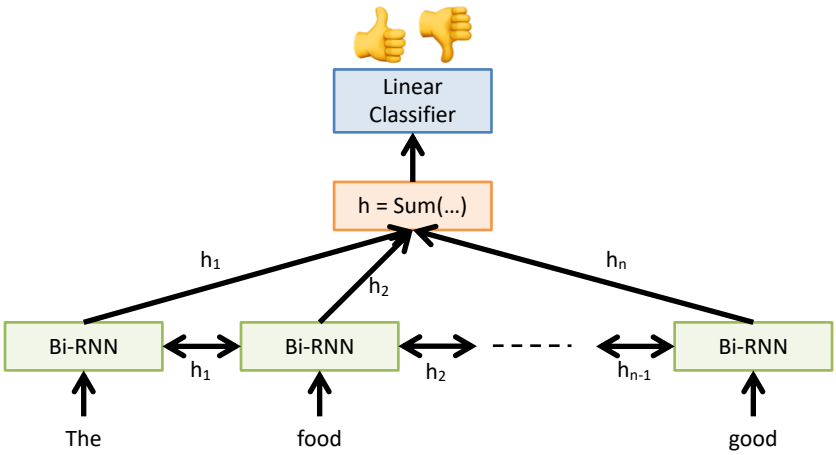


Sequence Classification



http://deeplearning.net/tutorial/lstm.html

Sequence Classification



Character RNN

100th iteration

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

300th iteration

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

700th iteration

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

2000th iteration

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

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

Image source

Image Caption Generation

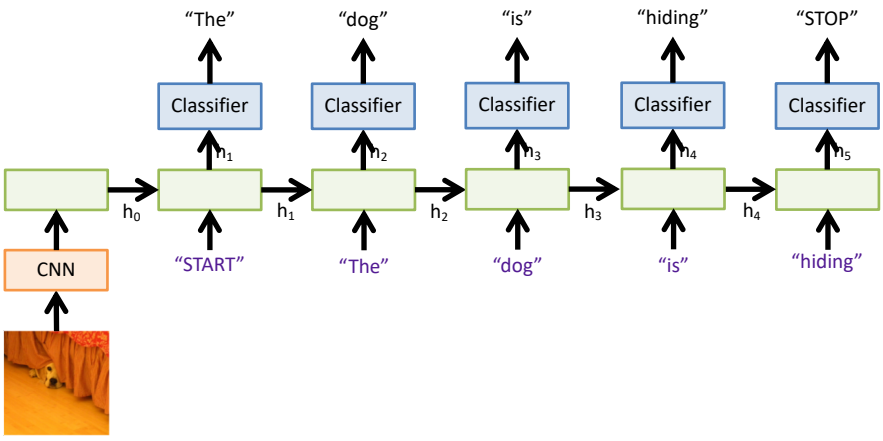
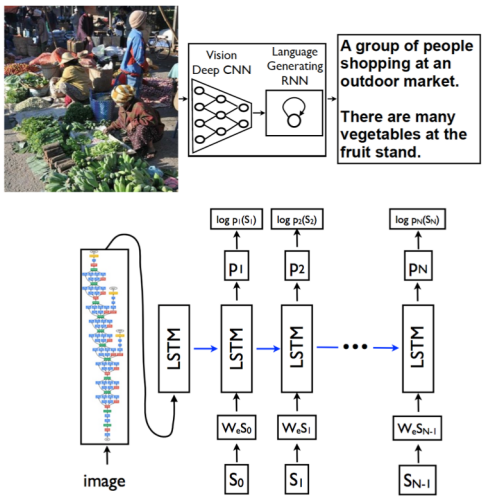
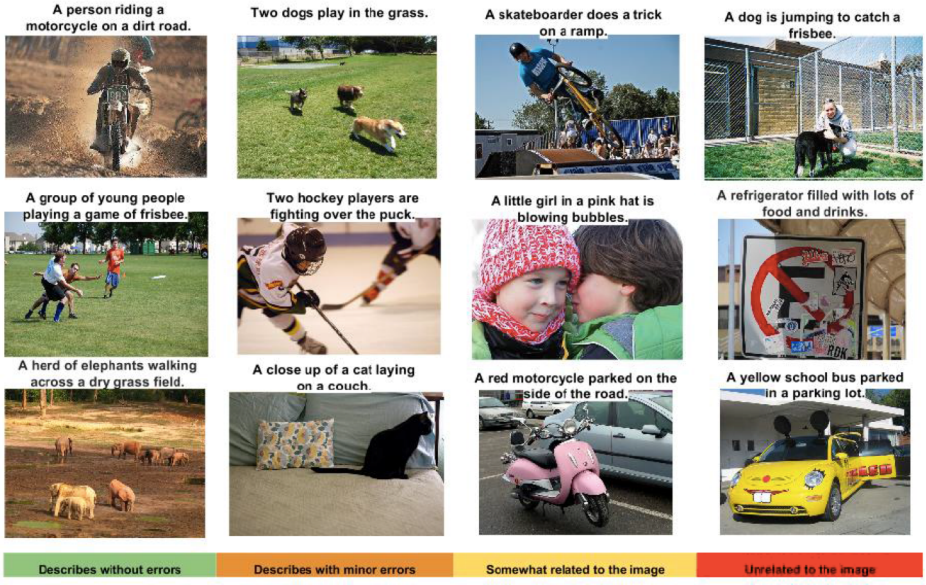


Image Caption Generation



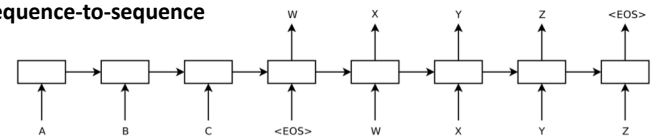
O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and Tell: A Neural Image Caption Generator, CVPR 2015

Image Caption Generation



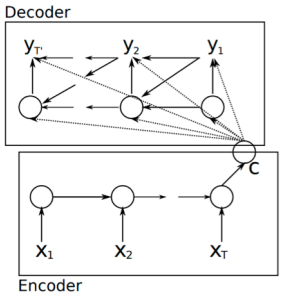
Machine Translation

Sequence-to-sequence



I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks, NIPS 2014

Encoder-decoder



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

Useful Resources / References

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