



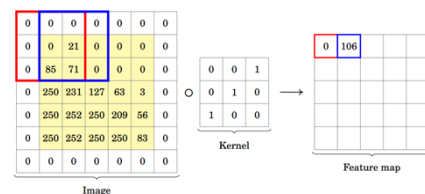
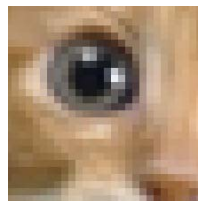
Recurrent Neural Networks



Fixed vs. Sequence

Image

- Fixed input size
- Related pixels are close together



Text

- Sequence, arbitrary input size
- Related words can be far, far away

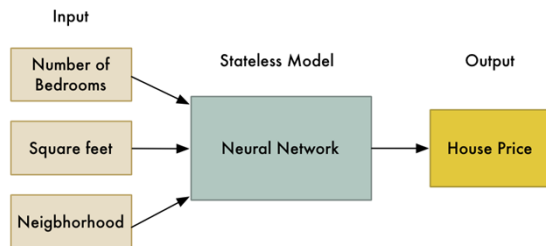
This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

Source: [Tal Perry](#)

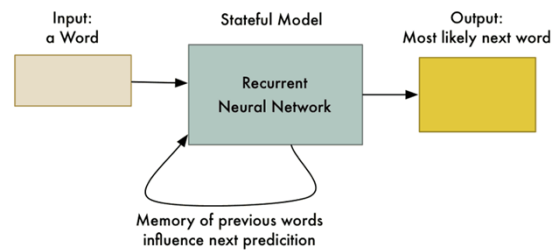


Stateless vs. Stateful

Predict current housing price



Predict the next most likely word

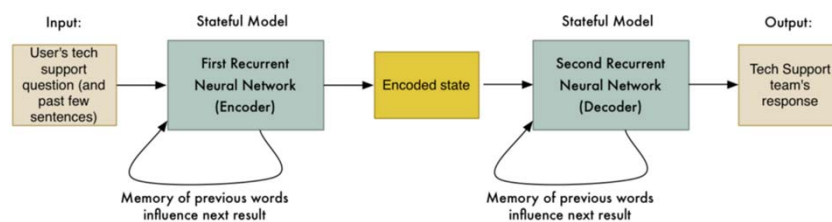
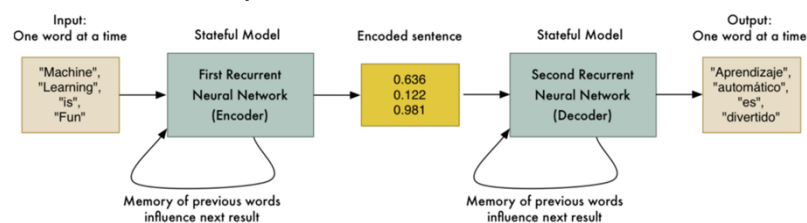


Output so far:
Machine

Source: [Adam Geitgey](#)



Sequence-to-Sequence

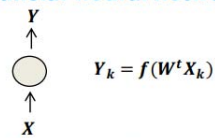


Source: [Adam Geitgey](#)

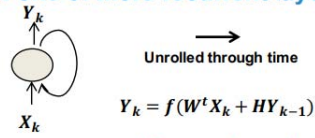


Fixed-to-Sequence, Sequence-to-fixed

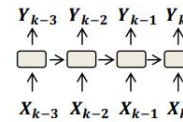
Artificial neural networks with one or more recurrent layers



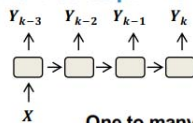
Classical neural network



Recurrent neural network

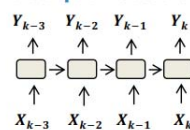


Able to cope with varying size sequences either at the input or at the output



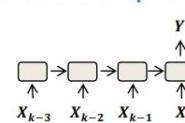
One to many
(fixed size input,
sequence output)

e.g. Image captioning



Many to many
(sequence input to sequence
output)

e.g. Speech to text



Many to one
(sequence input to fixed size
output)

e.g. Text classification



Source: [Yves Mabiala](#)



Image + Text: Dense Captioning

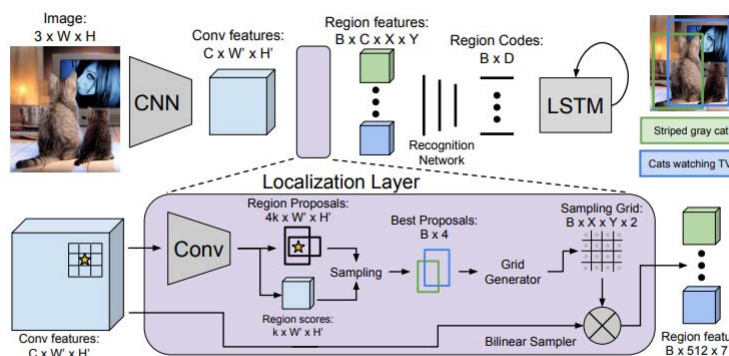


Figure 2. Model overview. An input image is first processed a CNN. The Localization Layer proposes regions and smoothly extracts a batch of corresponding activations using bilinear interpolation. These regions are processed with a fully-connected recognition network and described with an RNN language model. The model is trained end-to-end with gradient descent.

Source: <https://arxiv.org/pdf/1511.07571.pdf>



Recurrent Unit (Cell)

Stores state

Information from previous step(s)

In practice

Long Short Term Memory,
Gated Recurrent Unit

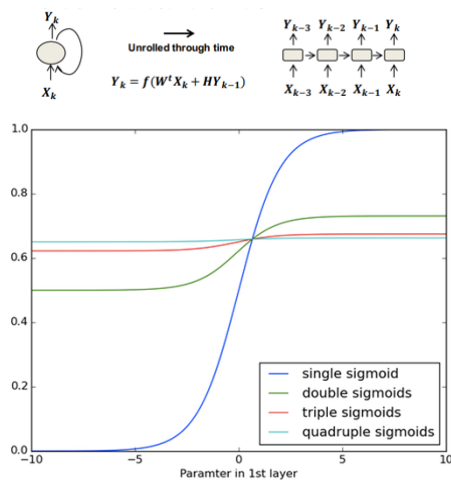
```
rnn = RNN()
y = rnn.step(x) # x: input, y: output

class RNN:
    def step(self, x):
        # update h, the hidden state
        self.h = np.tanh(np.dot(self.W_hh,
                                self.h) + np.dot(self.W_xh, x))
        # compute the output vector
        y = np.dot(self.W_hy, self.h)
        return y
```

Source: [Andrej Karpathy](#)



Vanishing / Exploding Gradient Problem



Source: [deeplearning4j](#)





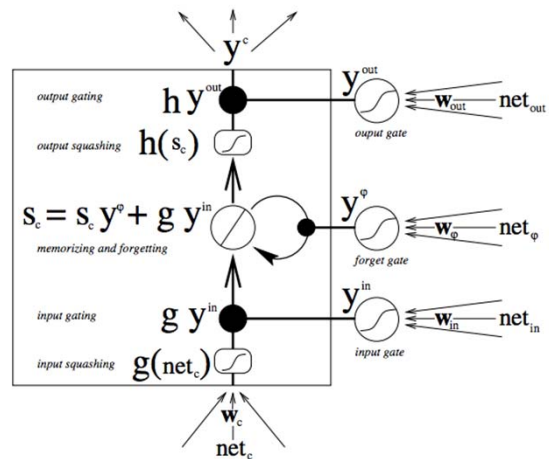
Long Short Term Memory

3 sigmoid “logic gates”

- input gate, forget gate, output gate
- weights (w) filter what to forget or remember

Memory unit

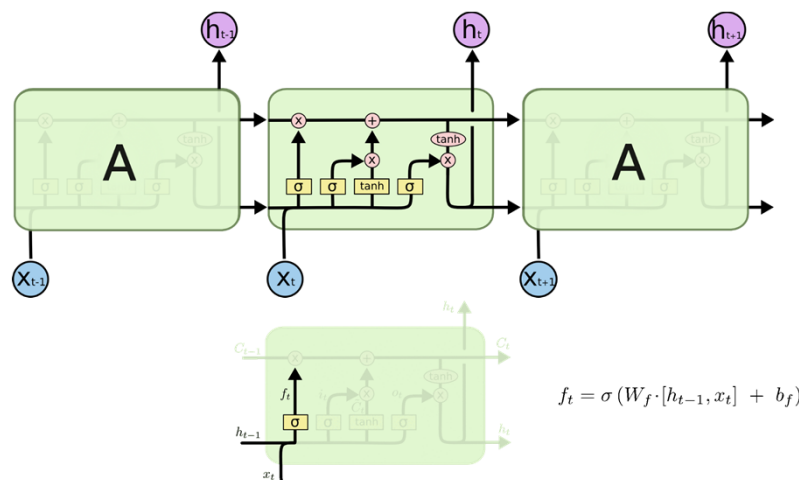
- Partially forgets existing memory, adds new memory
- No vanishing gradient



Source: deeplearning4j.org



Repeating over time



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

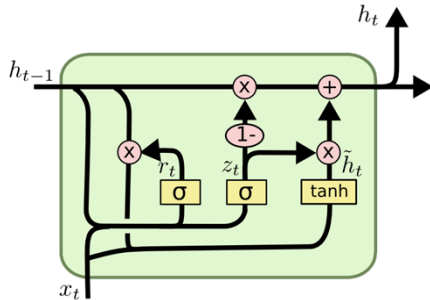


Gated Recurrent Unit

2 sigmoid “logic gates”

- reset gate (r), update gate (z)

No memory unit



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



GRU vs. LSTM

When to use GRU vs. LSTM

- GRU: simpler, faster to train, needs less data
- GRU: is newer research
- LSTM should work better with longer sequences

- First try GRU

- If lots of data, use LSTM too

			performance	lr	hidden	batch	sentLen	filter-size	margin
S	TextC	CNN	82.38	0.2	20	5	60	3	-
		GRU	86.32	0.1	30	50	60	-	-
		LSTM	84.51	0.2	20	40	60	-	-
	RC (F1)	CNN	68.02	0.12	70	10	20	3	-
		GRU	68.56	0.12	80	100	20	-	-
		LSTM	66.45	0.1	80	20	20	-	-
TE (acc)	TE (acc)	CNN	77.13	0.1	70	50	50	3	-
		GRU	78.78	0.1	50	80	65	-	-
		LSTM	77.85	0.1	80	50	50	-	-
	SemMatch AS (MAP & MRR)	CNN	(63.69,65.01)	0.01	30	60	40	3	0.3
		GRU	(62.58,63.59)	0.1	80	150	40	-	0.3
		LSTM	(62.00,63.26)	0.1	60	150	45	-	0.1
QRM (acc)	CNN	71.50	0.125	400	50	17	5	0.01	
	GRU	69.80	1.0	400	50	17	-	0.01	
	LSTM	71.44	1.0	200	50	17	-	0.01	
SeqOrder	PQA (hit@10)	CNN	54.42	0.01	250	50	5	3	0.4
		GRU	55.67	0.1	250	50	5	-	0.3
		LSTM	55.39	0.1	300	50	5	-	0.3
	ContextDep POS tagging (acc)	CNN	94.18	0.1	100	10	60	5	-
		GRU	93.15	0.1	50	50	60	-	-
		LSTM	93.18	0.1	200	70	60	-	-
Bi-GRU		94.26	0.1	50	50	60	-	-	
Bi-LSTM		94.35	0.1	150	5	60	-	-	

Table 1: Best results on CNN, GRU and LSTM in NLP tasks

Source: [Wenpeng Yin et al.](#)



Hyperparameters: tuning

General: <https://cs231n.github.io/neural-networks-3>

- RNNs have higher risk of overfitting

RNNs for text: <https://arxiv.org/abs/1707.06799>

- Choice of word embeddings
- Optimizers: Adam, Nadam, RMSProp
- Dropout: on both output and recurrent units
- Gradient normalization: better performance, avoid exploding gradient