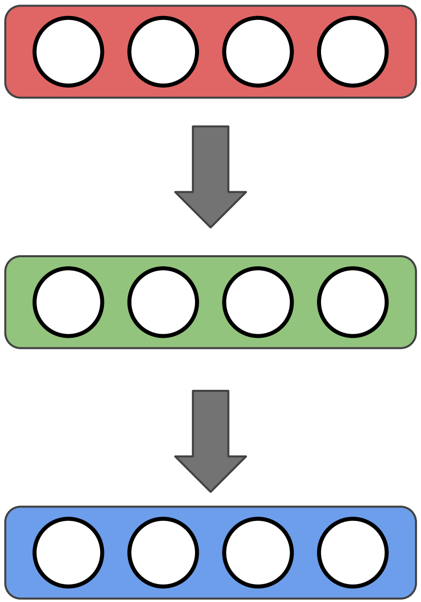
Machine Learning
Recurrent Neural Networks

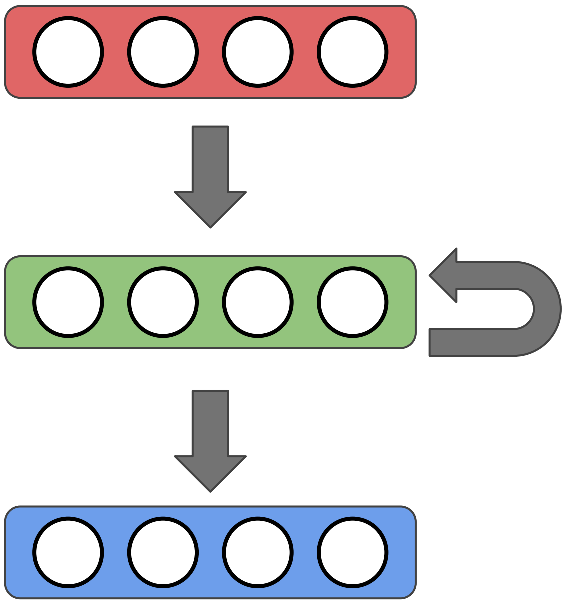
Phil Jackson
Department of Computer Science
Durham University

Cyclic Computation



- Neural nets we've encountered so far have been directed acyclic graphs

Cyclic Computation



- Neural nets we've encountered so far have been directed acyclic graphs
- What happens when we add a cycle?

Cyclic Computation

$$x$$

$$\downarrow$$

$$h = \tanh(W_{ih} x)$$

$$\downarrow$$

$$y = \mathrm{softmax}(W_{ho} h)$$

- Layer activations represented as vectors (x,h,y)

Cyclic Computation

$$x$$

$$\downarrow$$

$$h = \tanh(W_{ih}x)$$

$$\downarrow$$

$$y = \mathrm{softmax}(W_{ho}h)$$

- Layer activations represented as vectors (x,h,y)
- Matrix multiplications compute weighted sums of previous layer activations

Cyclic Computation

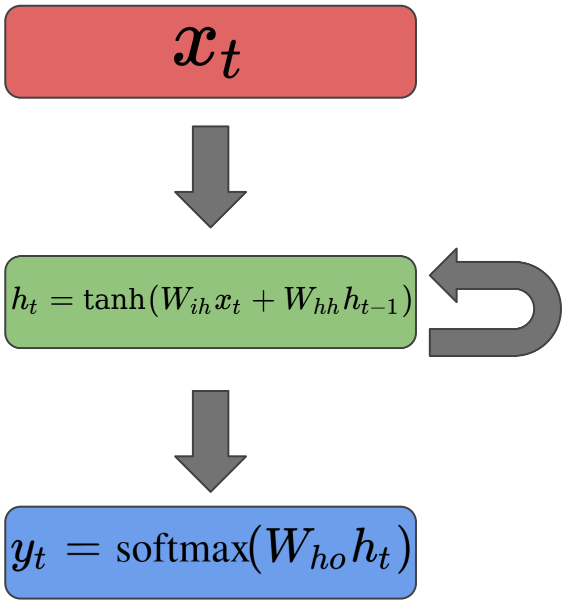
$$x$$

$$h = \tanh(W_{ih} x)$$

$$y = \text{softmax}(W_{ho} h)$$

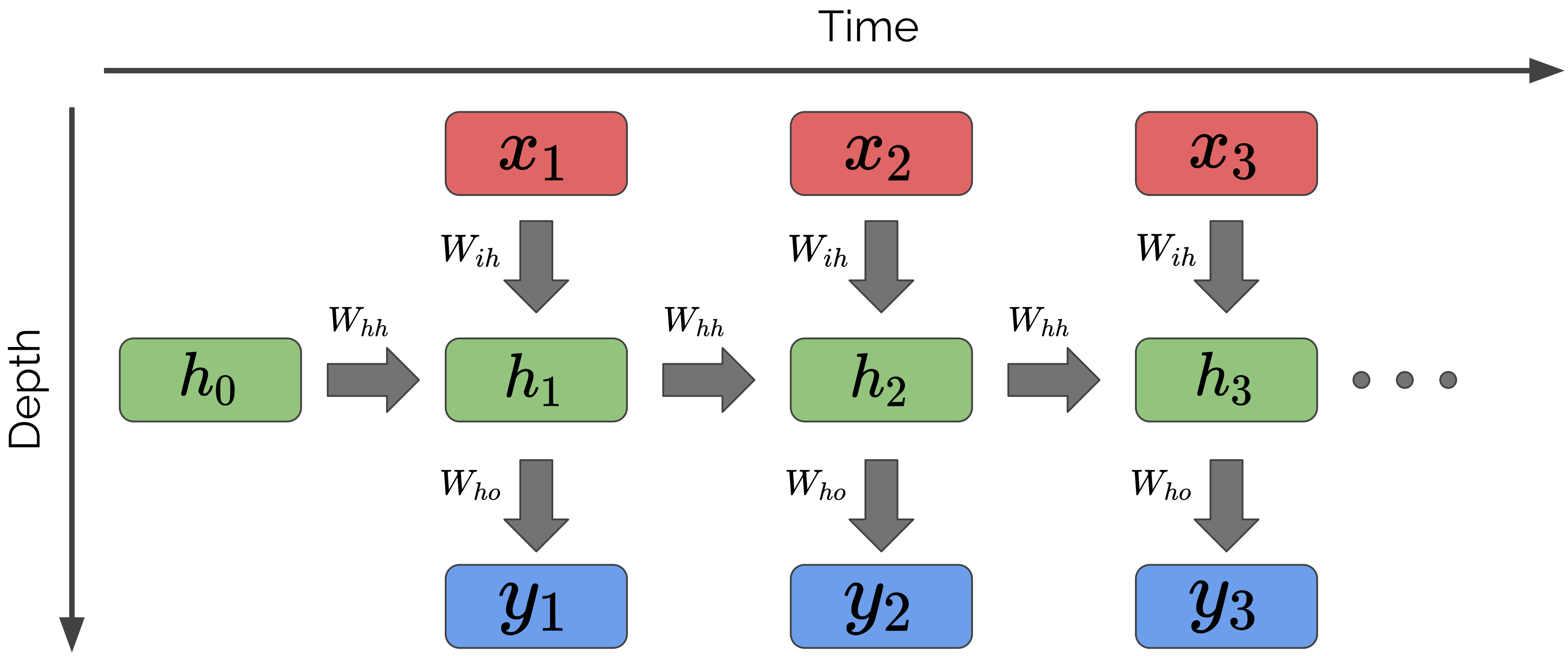
- Layer activations represented as vectors (x,h,y)
- Matrix multiplications compute weighted sums of previous layer activations
- Non-linear function (ReLU, softmax) of the weighted sum input yields activation

Cyclic Computation



- To deal with recurrency, we need a notion of time
- Hidden layer works the same as before, but it now incorporates a weighted sum of its own *previous* activations
- Usually $h_0 = 0$

Unrolling RNNs



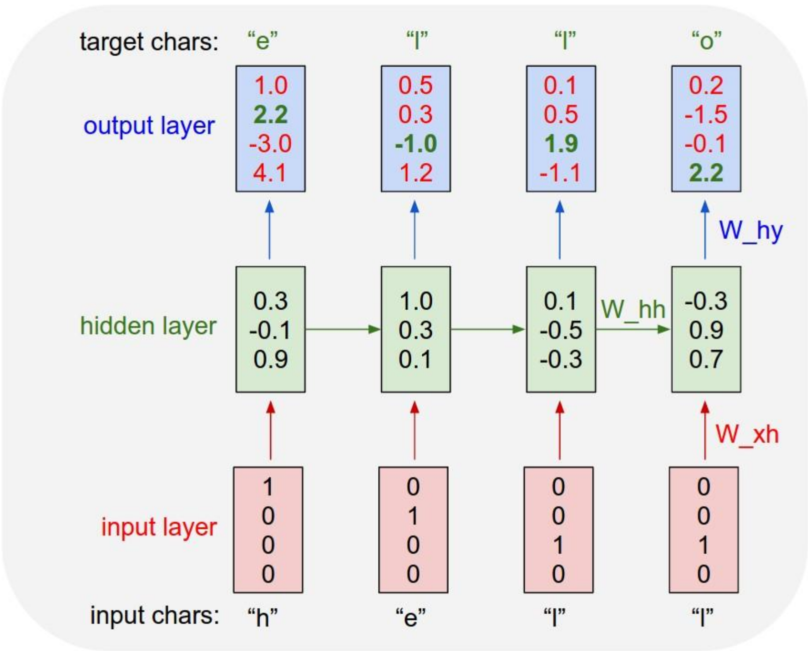
Just Remember

Durham University

Every output depends on every previous input.

Sequence Modelling

Durham University



Source: The Unreasonable Effectiveness of Recurrent Neural Networks, Andrej Karpathy

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

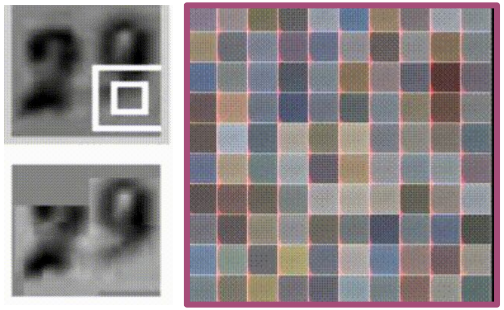
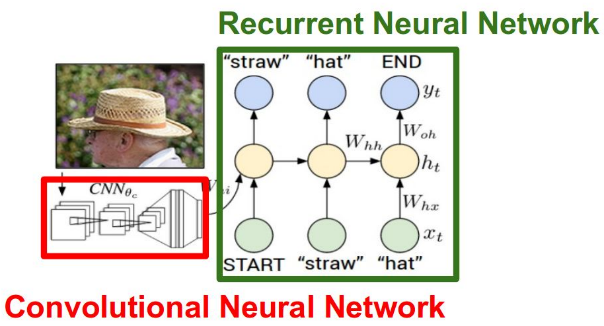


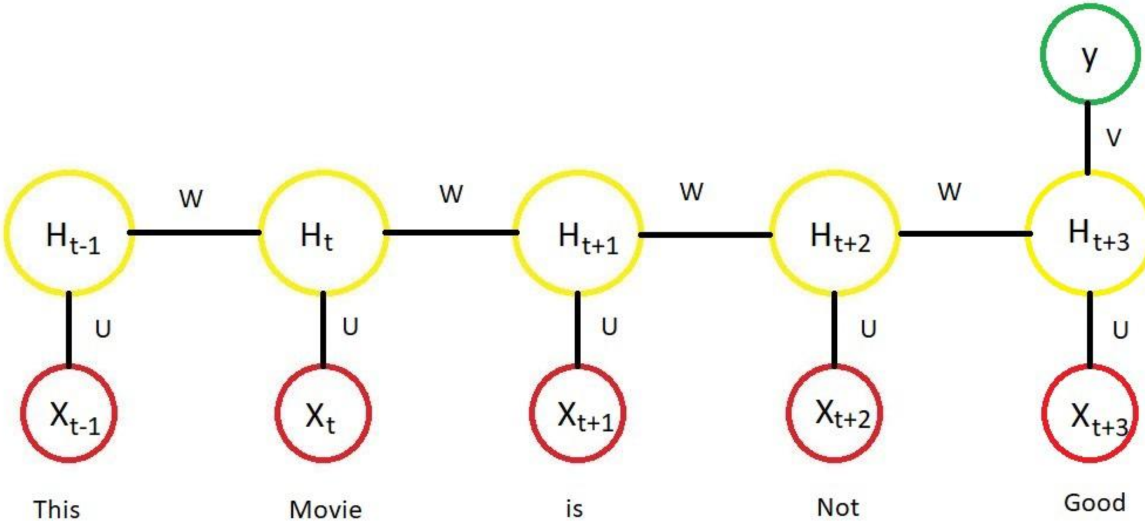
Image Captioning



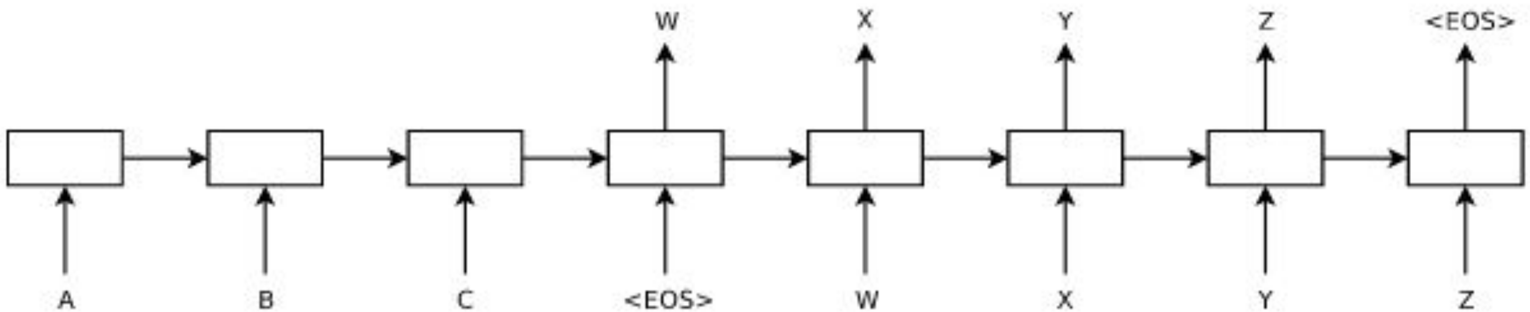
Source: Deep Visual-Semantic Alignments for Generating Image Descriptions https://cs.stanford.edu/people/karpathy/cvpr2015.pdf

- h_o initialized with output from a convolutional neural network
- Previous output word passed as next input

Sentiment Analysis

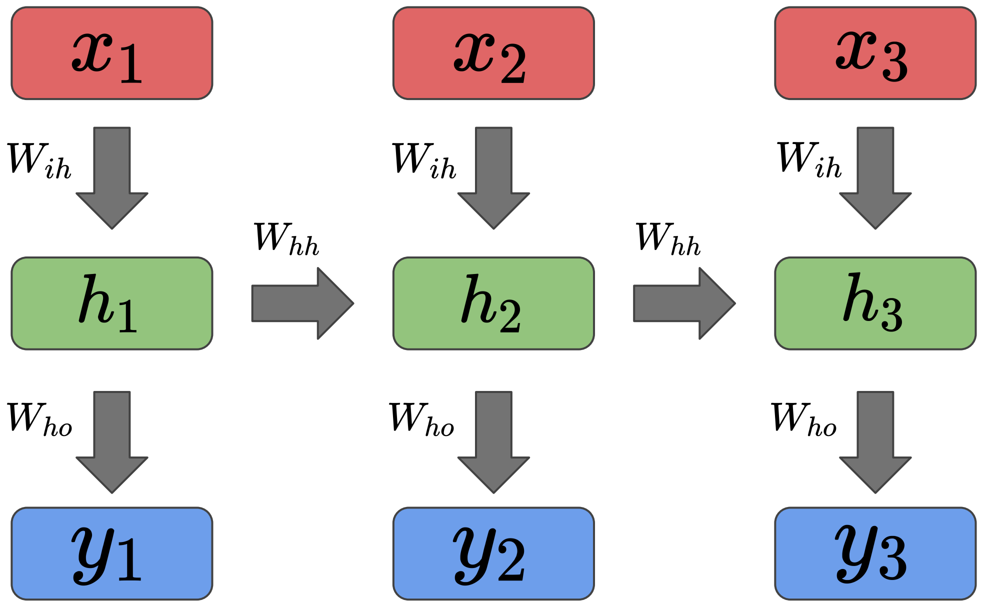


Sequence to Sequence



Source: Sequence to Sequence Learning with Neural Networks
https://arxiv.org/pdf/1409.3215.pdf

Training RNNs



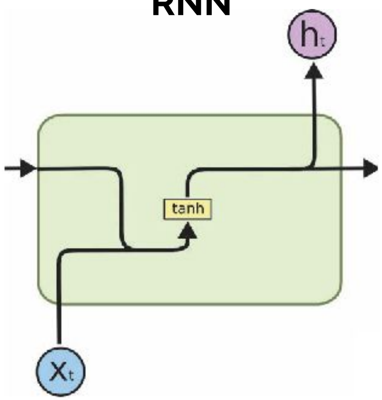
- Use BackPropagation Through Time (BPTT)
- Quite memory intensive; may have to use truncated BPTT
- Unrolled diagrams come in very handy when thinking about this

Drawbacks of RNNs

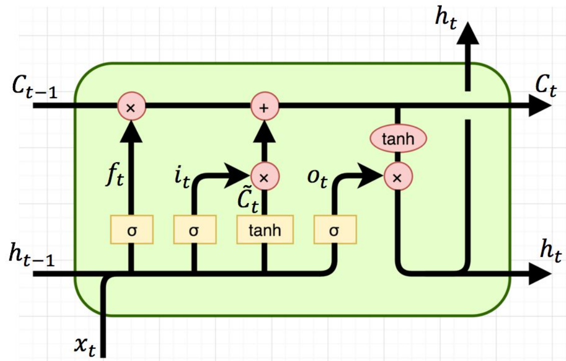
- Harder to parallelize because of their sequential nature, therefore slower than feed-forward networks
- Difficult to train, vulnerable to both vanishing gradients and exploding gradients
  - Solution: clamp the gradients
- RNNs are forgetful; often have trouble learning long-term dependencies

LSTMs and GRUs

RNN

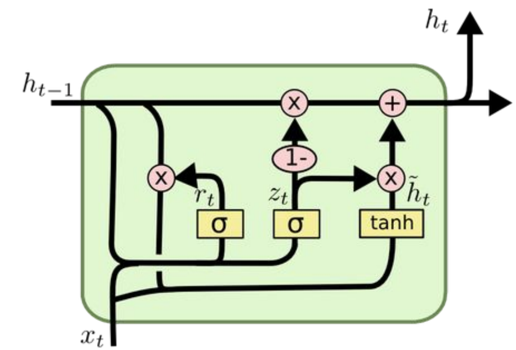


LSTM

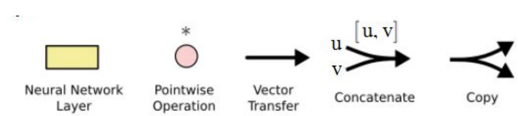


$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

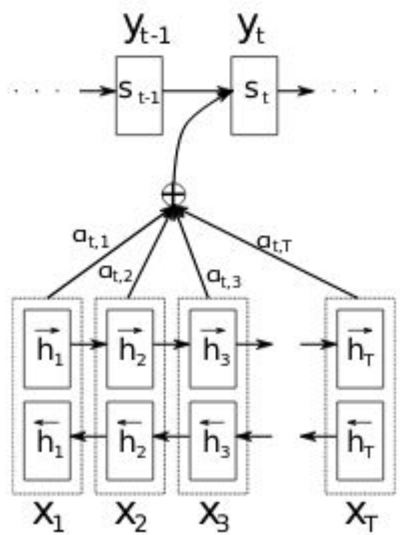
GRU



$$z_t = \sigma\left(x_t U^z + h_{t-1} W^z\right)$$
$$r_t = \sigma\left(x_t U^r + h_{t-1} W^r\right)$$
$$\tilde{h}_t = \tanh\left(x_t U^h + (r_t * h_{t-1}) W^h\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Attentional RNNs



- Solves the problem of a fixed size hidden state having to encode arbitrary length input sequences
- Remember all previous hidden states, predict attention weights over them when producing output sequence

Source: Neural Machine Translation by Jointly Learning to Align and Translate

https://arxiv.org/pdf/1409.0473v7.pdf

Take away points

- RNNs are neural networks with internal state
- Every output is affected by every previous input
- RNNs are very versatile and well suited for sequence modeling
- To compute gradients, must unroll and backpropagate through time
- In practice, LSTM and GRU layers are normally used since they outperform vanilla RNNs