Neural Networks and Deep Learning

Recurrent Networks I

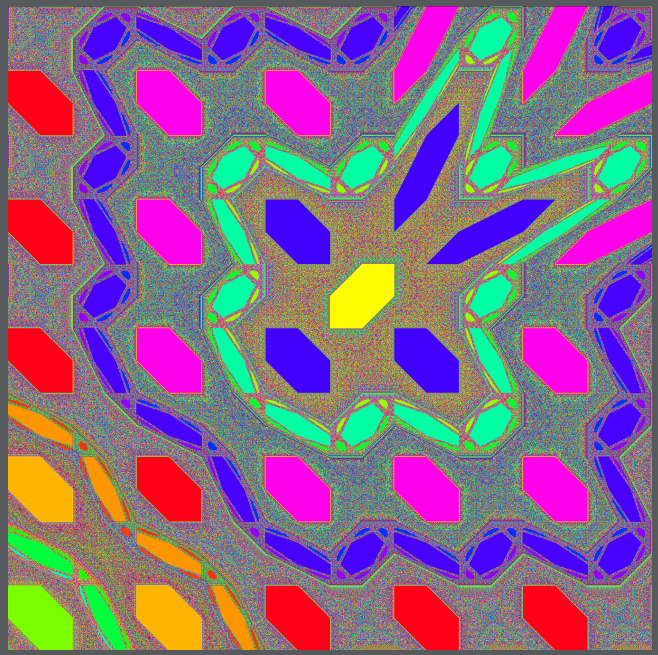
Nicholas Dronen

Department of Computer Science
dronen@colorado.edu

February 18, 2019

University of Colorado **Boulder**

Recurrent Neural Networks



Gingerbread Man Map

Salary Man Car

$$\begin{cases} x_{n+1} &= 1 - y_n + |x_n| \\ y_{n+1} &= x_n \end{cases}$$

Dynamical systems describe the time-dependent evolution of points in geometrical space.

Consider the classical form of a dynamical system:

$$\mathbf{s}^t = f(\mathbf{s}^{(t-1)}; \theta) \tag{1}$$

where $1 < t < \tau$ is the time step index, $\mathbf{s}^t$ is the state at index t, and θ denotes the parameters of the system.

What is the computational graph of the system?

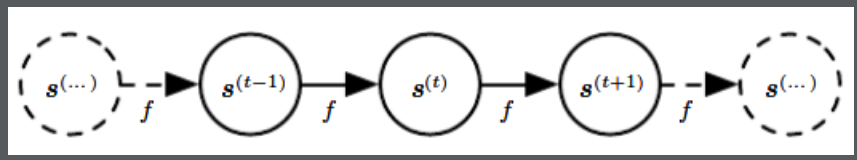
Ignoring the parameter θ, the computational graph of a dynamical system is a single node with a self loop.

With finite τ, the graph resulting from applying the system $\tau - 1$ times can be freed of recursion by removing the self loop and duplicating the node $\tau - 1$ times.

For example, with $\tau = 3$:

$$s^{(3)} = f(s^{(2)}; \theta)$$
$$= f(f(s^{(1)}; \theta); \theta)$$

This is called **unfolding**.



Source: http://www.deeplearningbook.org

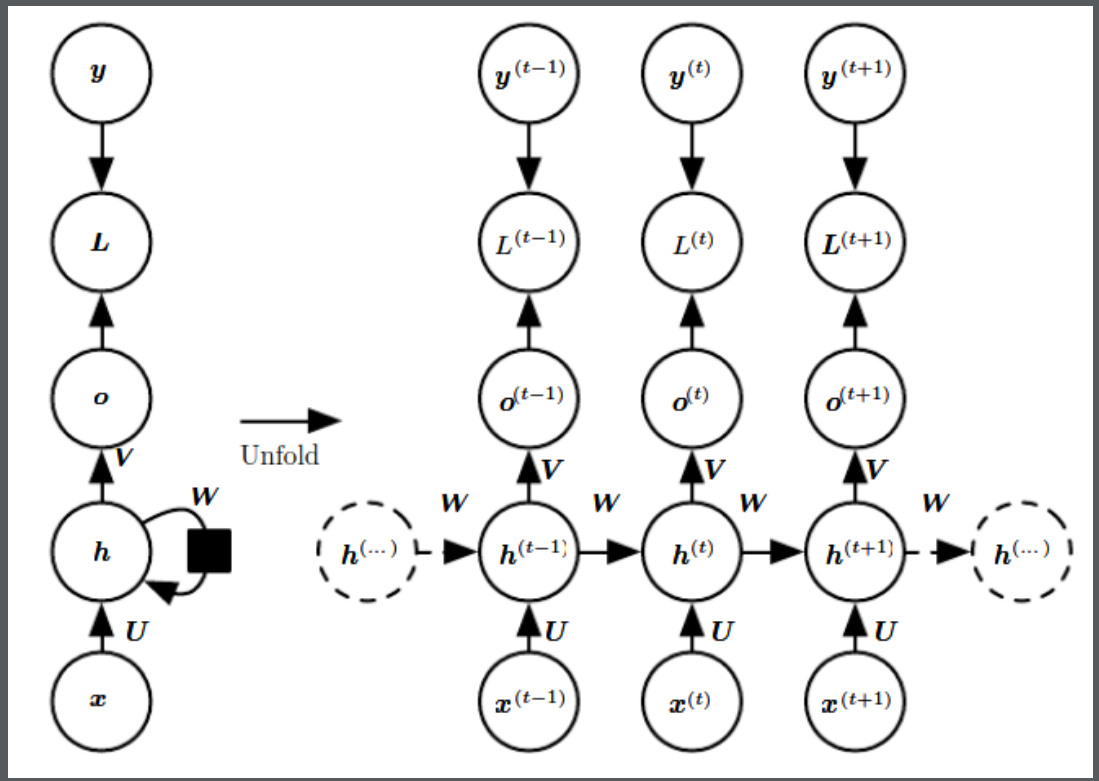
RNN with Multiple (Categorical) Outputs

Assume multiple categorical outputs for repeatedly predicting the next character or word in text.

$$\alpha^t = Wh^{t-1} + Ux^t + b_\alpha$$
$$h^t = \tanh(\alpha^t)$$
$$o^t = Vh^t + b_o$$
$$\hat{y}^t = \text{softmax}(o^t)$$

U	Input-to-hidden weights
V	Hidden-to-output weights
W	Hidden-to-hidden (recurrent) weights
b_α	Input bias
b_o	Output bias
h^0	Initial state

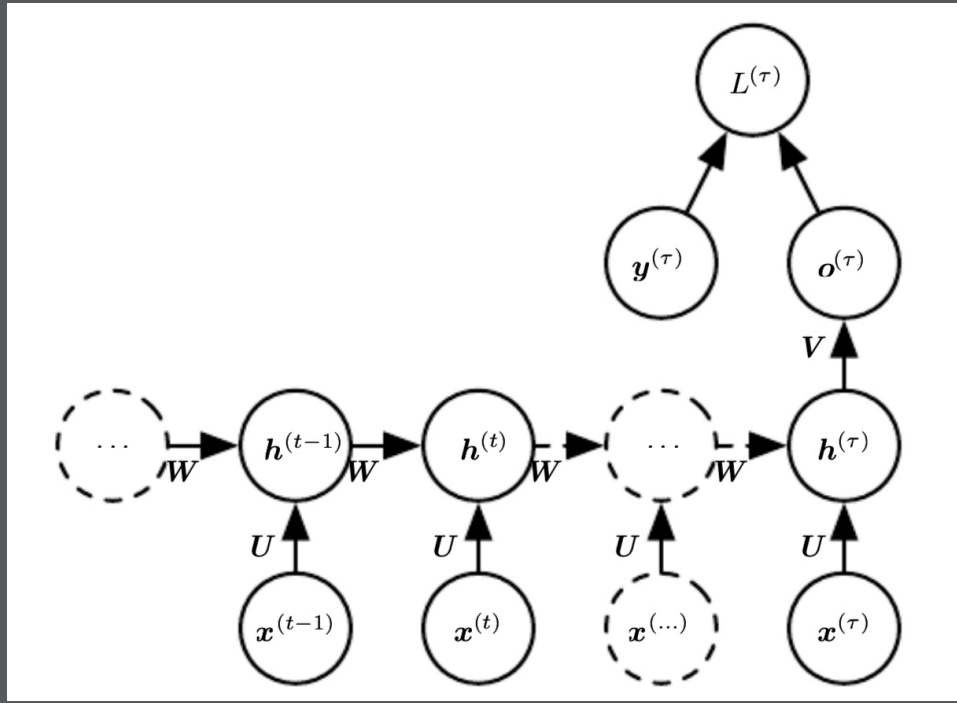
Computational Graph of RNN with Multiple Outputs



(*Left*) Circuit diagram of recurrent network. (*Right*) The same graph unfolded over time, so each node is associated with a particular time step t. Source: http://www.deeplearningbook.org

Computational Graph of RNN with a Single Output

As with RNN on previous slide, $h^{(t)}$ depends on $h^{(t-1)}$, so this network is not readily paralellized.

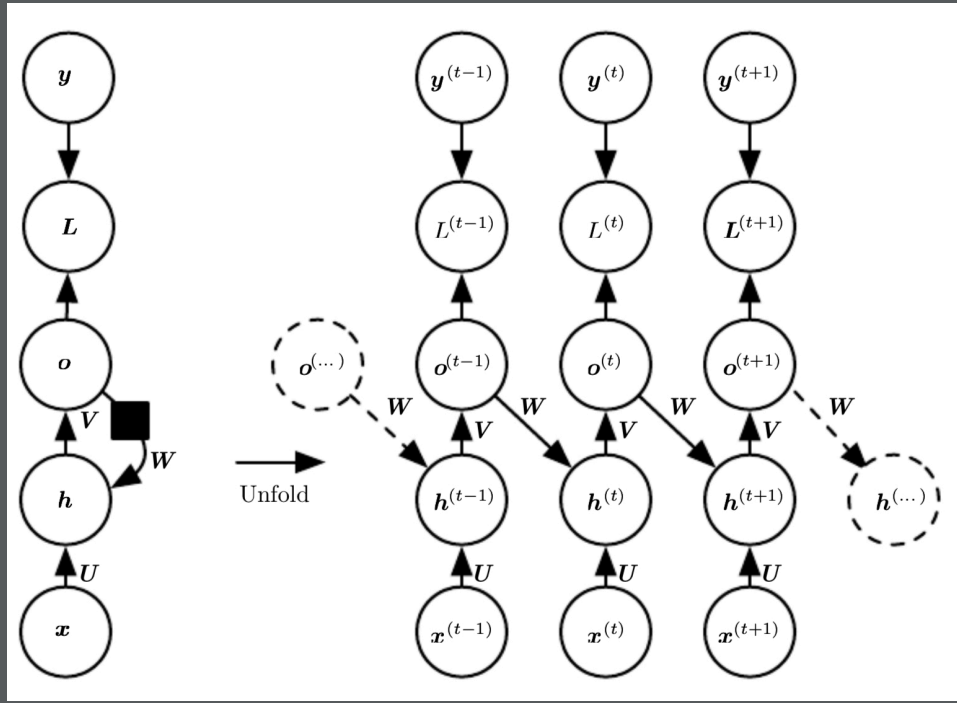


Source: http://www.deeplearningbook.org

Computational Graph of RNN with Output-to-Hidden Recurrence

Because the input to the hidden state $h^{(t)}$ of this network is the output $o^{(t)}$, and the output is most directly conditioned by the loss, this network is less powerful than the previous two.

Why?



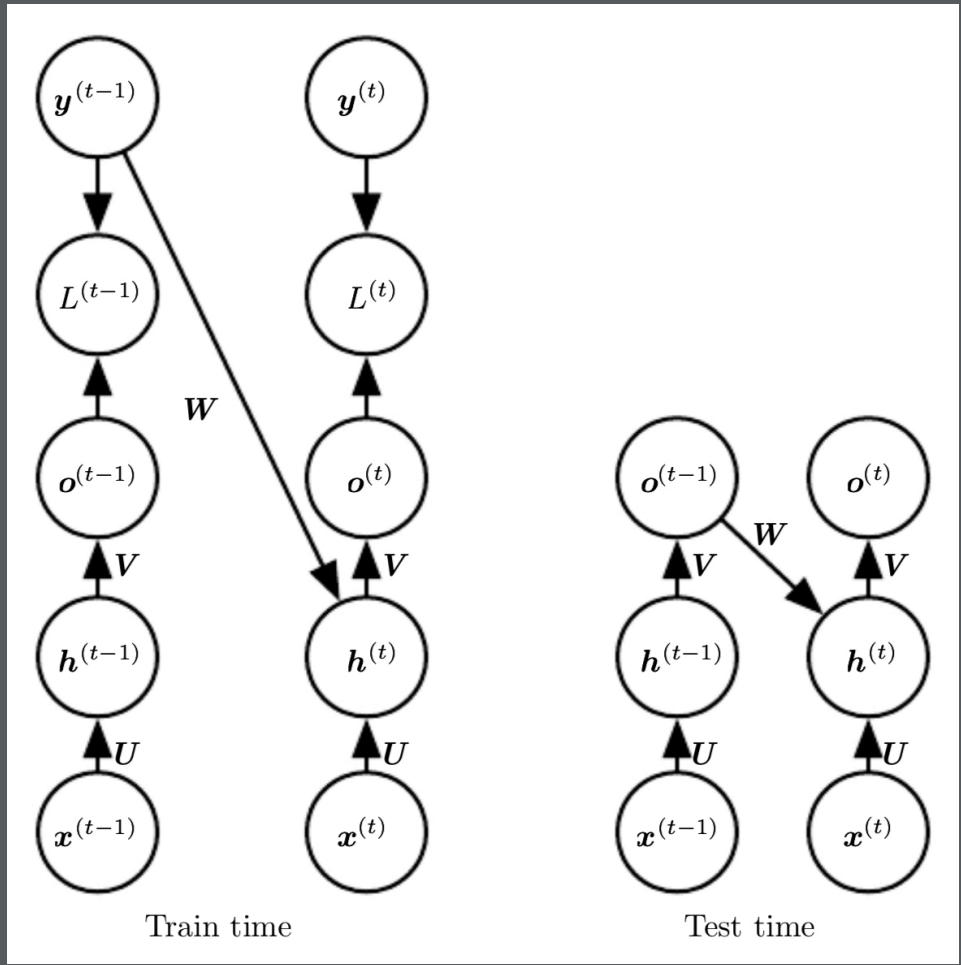
Source: http://www.deeplearningbook.org

Teacher Forcing

Teacher forcing allows one to directly input the target $y^{(t-1)}$ to the hidden state h^t at training time. This makes parallelism possible.

At inference time, the *output* of the network can be input to the hidden state.

What are the risks of doing this?

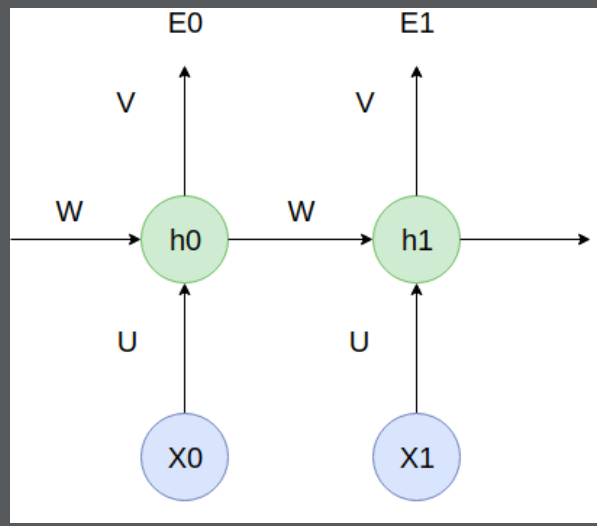


Source: http://www.deeplearningbook.org

Backpropgation Through Time (BPTT)

Backpropagating through an unrolled RNN requires

$$\frac{\partial E^1}{\partial V} = \frac{\partial E^1}{\partial \hat{y}^1}\frac{\partial \hat{y}^1}{\partial V}$$
$$\frac{\partial E^1}{\partial W} = \frac{\partial E^1}{\partial \hat{y}^1}\frac{\partial \hat{y}^1}{\partial h^1}\frac{\partial h^1}{\partial W}$$



Note the dependency of h^1 on h^0.

$$\frac{\partial E^1}{\partial W} = \frac{\partial E^1}{\partial \hat{y}^1}\frac{\partial \hat{y}^1}{\partial h^1}\frac{\partial h^1}{\partial h^0}\frac{\partial h^0}{\partial W}$$

Backpropagation Through Time (BPTT)

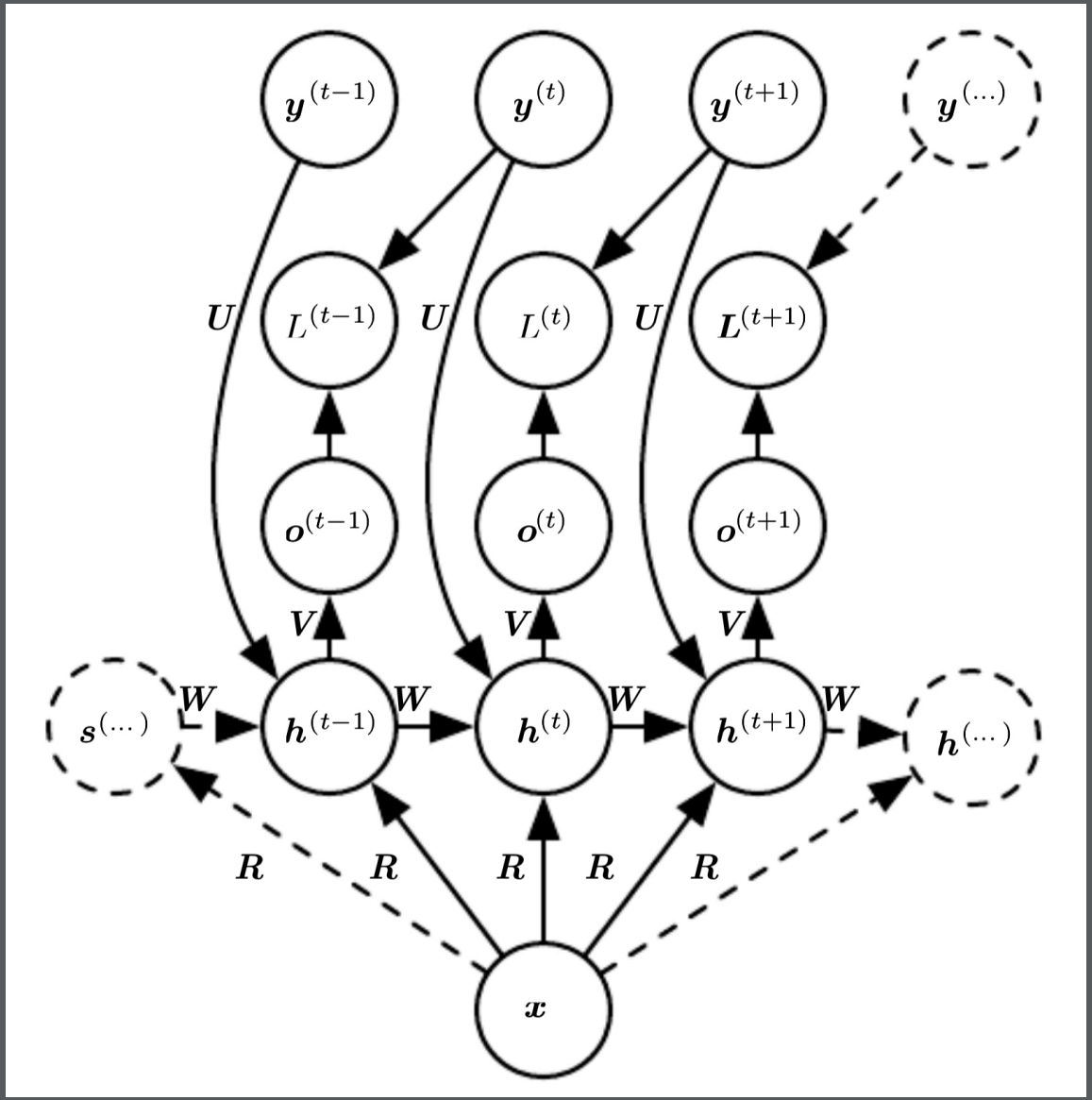
Generally,

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E^t}{\partial W}$$

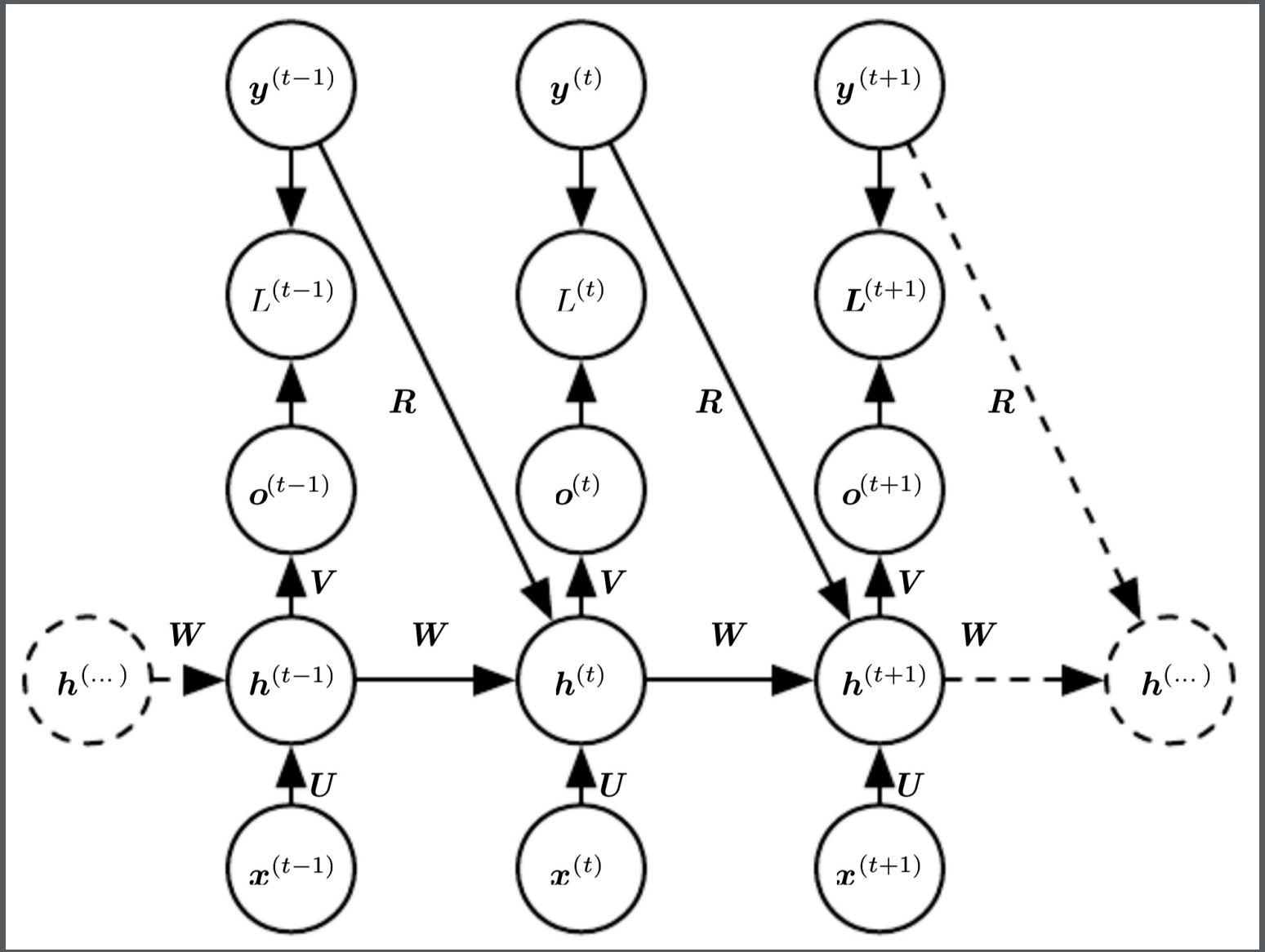
$$\frac{\partial E^t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E^t}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial h^k}{\partial W}$$

We can limit the computational cost of this procedure by starting with $k = t - a$, where a is a constant and $a < t$. This is *truncated backpropagation through time* (TBPTT).

Conditioning RNNs on Context



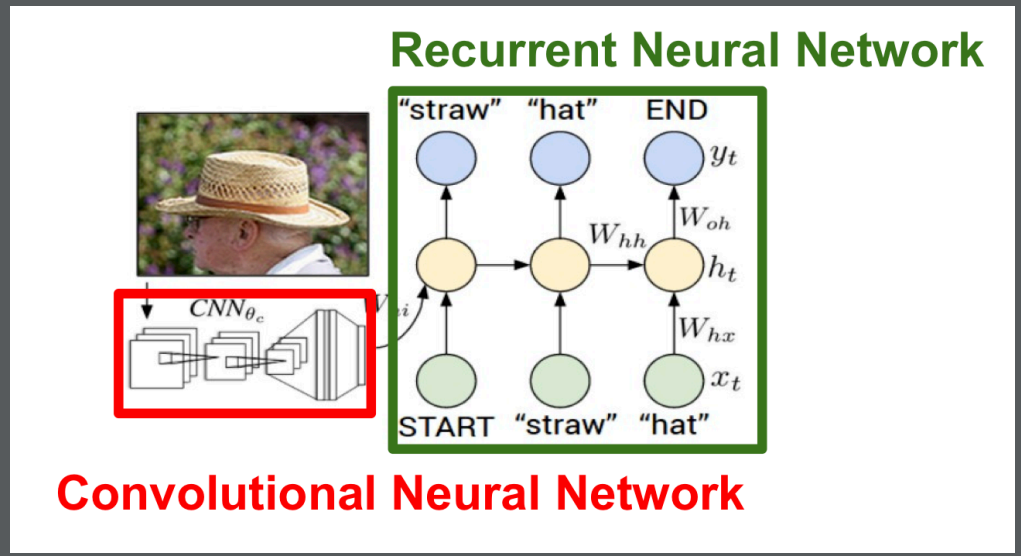
Conditioning RNNs on Context



Conditioning Example: Image Captioning

A ConvNet+RNN architecture, leveraging context information, used for automatic captioning.

Source: Andrey Karpathy, Fei-Fei Li



Knowledge of the future can help us understand the present better.

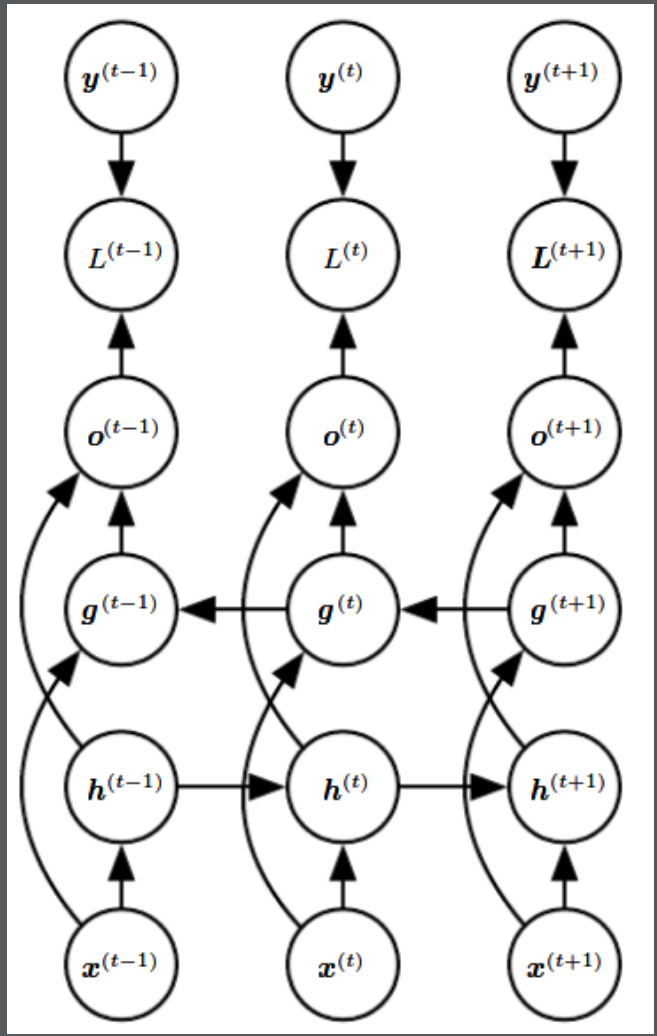
In some languages, the subject of a sentence appears first, then the object, then the verb. These are called SOV language. Japanese is SOV. English is SVO.

Translating between SOV and SVO languages would be easier if we could know the source verb (SOV $\rightarrow$ SVO) or object (SVO $\rightarrow$ SOV) close to the time we want to output the target verb.

Doesn't peeking into the future contradict the *online* nature of dynamical systems?

Bidirectional RNNs

A bidirectional RNN (Schuster and Paliwal, 1997) has hidden state for *each direction* of the input. Here, h carries information forward in time, and g carries it backward. Higher layers of the network have access to representations of each time step conditioned on both its left and right context.

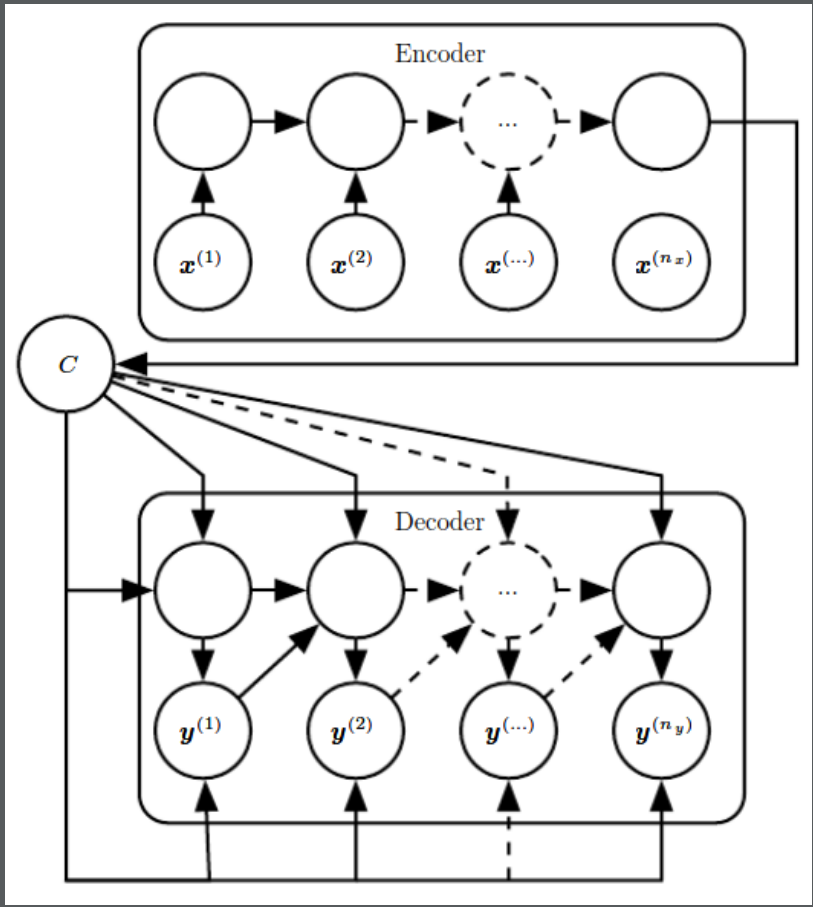


Encoder-Decoder RNNs

In some tasks, like machine translation, input and output lengths differ.

An encoder-decoder RNN (Badhanau et al, 2016) decouples input and output lengths by having an encoder map the input to a context C and a decoder map C to the target output.

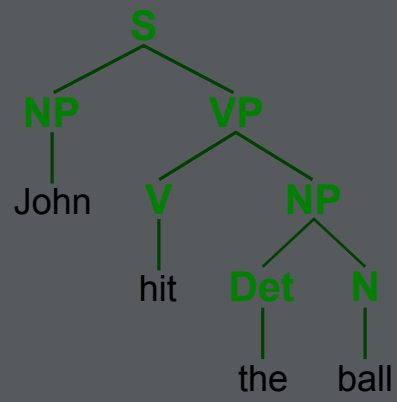
How does the decoder know when to stop?



Example of an encoder-decoder or sequence-to-sequence RNN architecture.
Source: http://www.deeplearningbook.org

Recursive Neural Networks

Some interesting sequential data can be represented hierarchically, and the hierarchical representations can be represented sequentially.



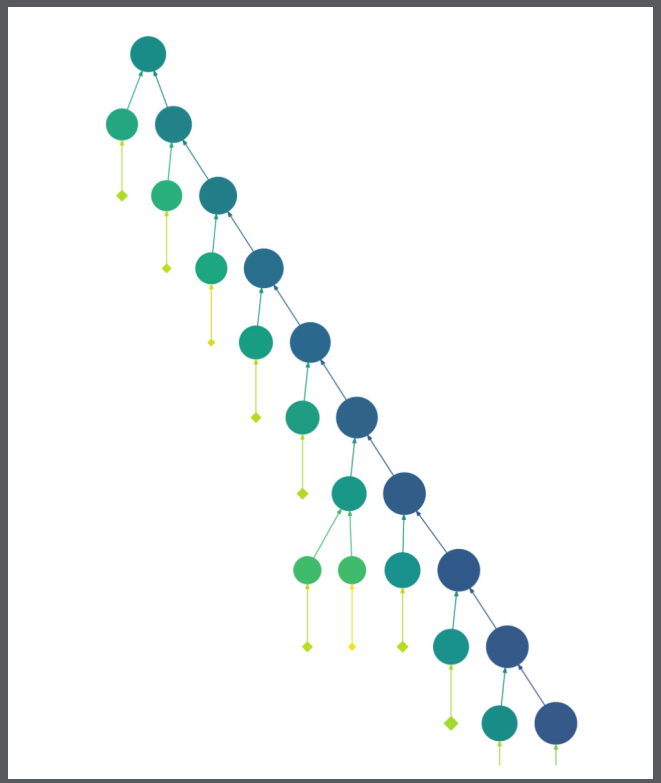
A constituency-based parse tree

John has a dog . → (S (NP NNP)$_{NP}$ (VP VBZ (NP DT NN)$_{NP}$)$_{VP}$.)$_{S}$

Linearization of a parse tree

Recursive Neural Networks

And some hierarchical data are interesting.



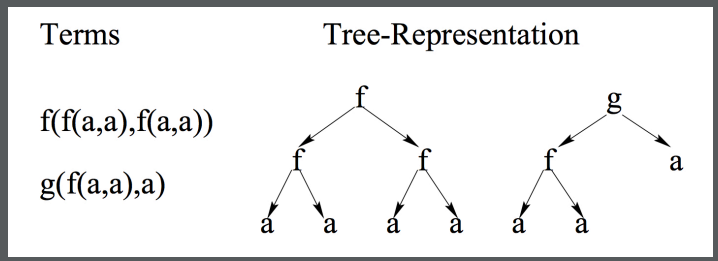
Typical tree structures for 1 TeV quark jet. Source: https://arxiv.org/abs/1711.02633

Recursive Neural Networks

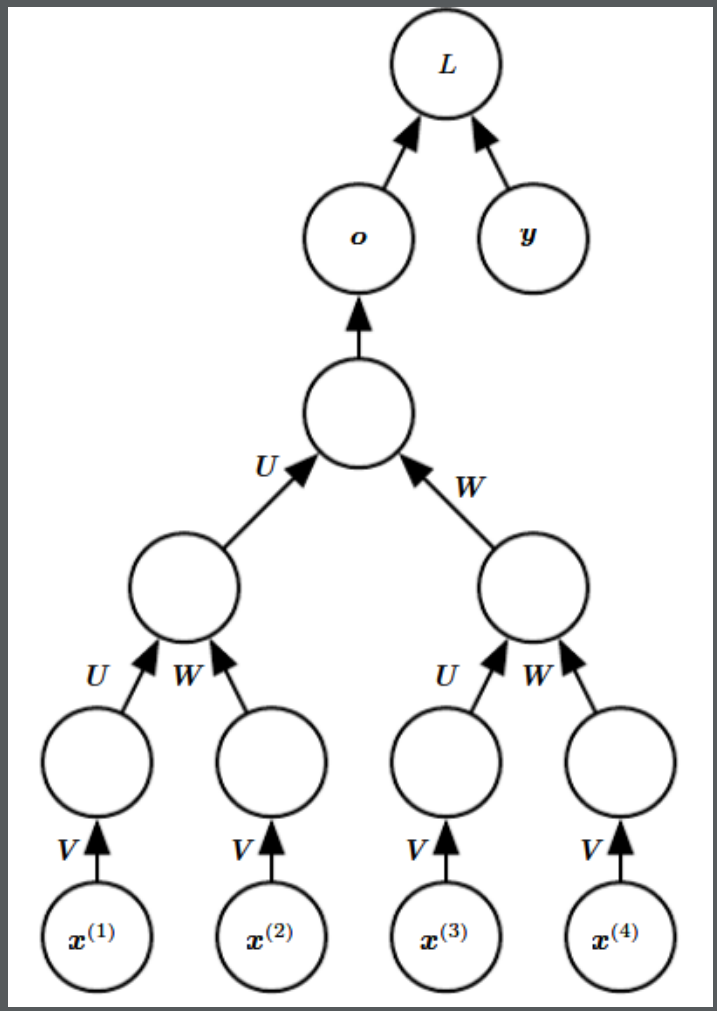
The computational graph of a recursive network is a tree.

Term of art for the backward pass is backpropagation through structure (BPTS).

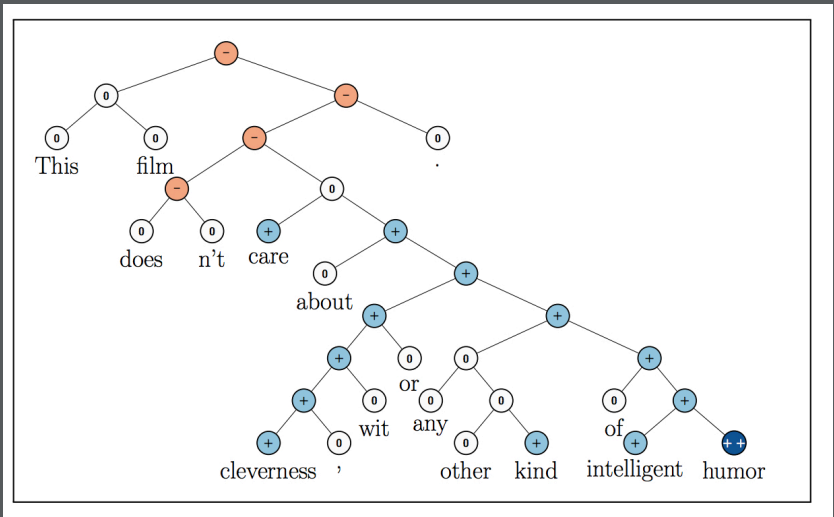
What are V, U, and W?

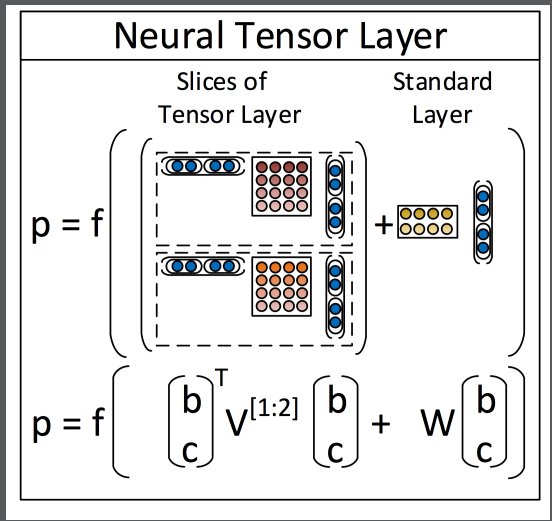


Learning Task-Dependent Distributed Representations by Backpropagation Through Structure, Goller and Andreas Küchler, 1996



Recursive Neural Networks



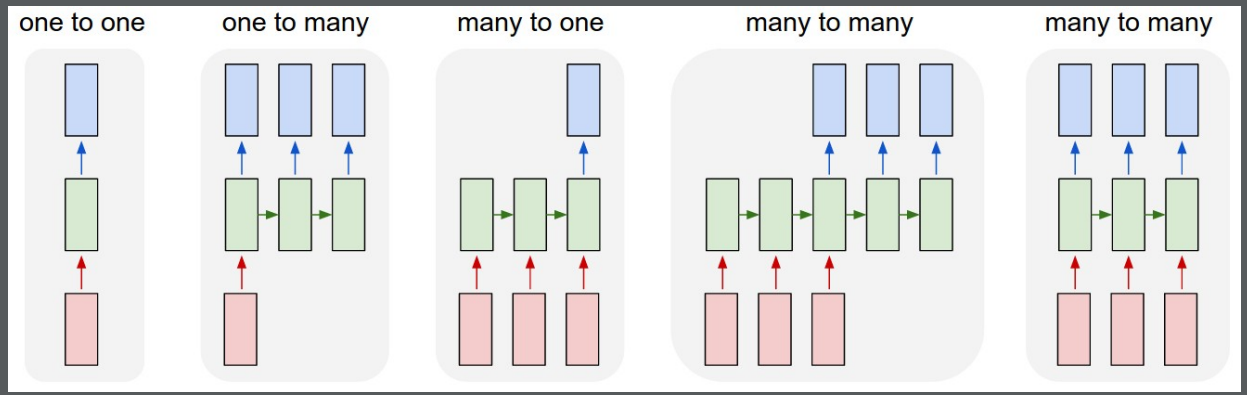


RecNNs are attractive because trees may allow discovery of long-term dependencies. Why?

Matrix weights may be too low-variance for the problem. Socher et al (2013) proposed tensor weights in a paper that also introduced a sentiment classification dataset.

RecNNs enjoyed some success in the literature several years ago but were quickly supplanted by more powerful RNNs.

RNN Design Patterns



Each rectangle is a vector, arrows represent functions (e.g. matrix multiply). *red*: input, *blue*: output, *green*: hidden state.

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

- One to one → no recurrence, a feed-forward network.
- One to many → image captioning (map an image to a sentence)
- Many to one → sentence classification
- Many to many → machine translation
- Many to many → video classification task labeling each frame