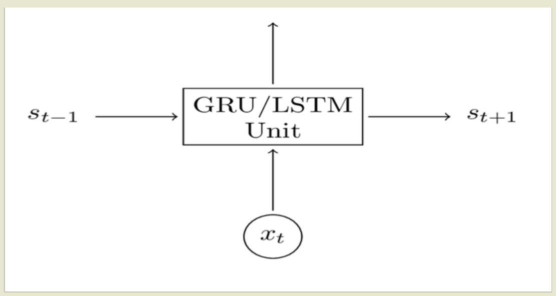
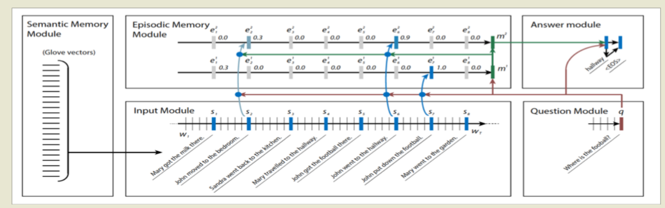
Recurrent Neural Networks





Time Series

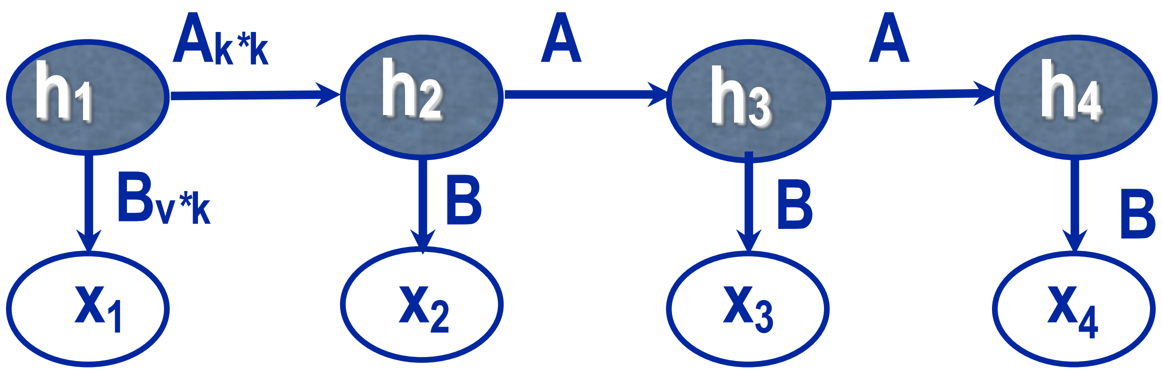
Lyle Ungar, University of Pennsylvania

Recurrent Neural Nets

- **Generalize HMMs or Linear Dynamical Systems**
  - Hidden state dynamical models, but *nonlinear*
- **Needed if you have inputs of varying length**
  - E.g. sequence of observations
    - speech
    - text
    - robots
    - power plants, chemical plants, data centers

Standard HMM

- **HMM learning problem: Estimate A and B**

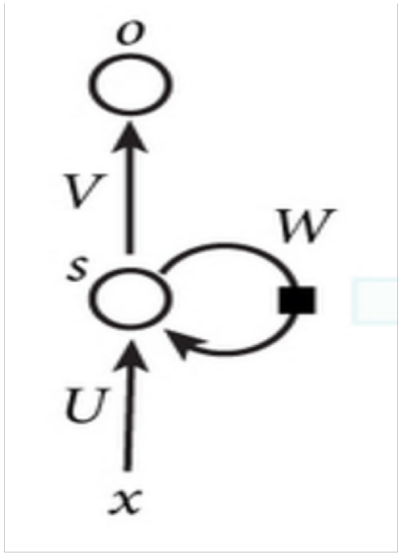


A = Markov transition matrix
B = emission matrix

- **Estimation done via EM**
  - Or spectral methods
- **History is forgotten with an exponential decay**

Simple Recurrent Neural Net

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

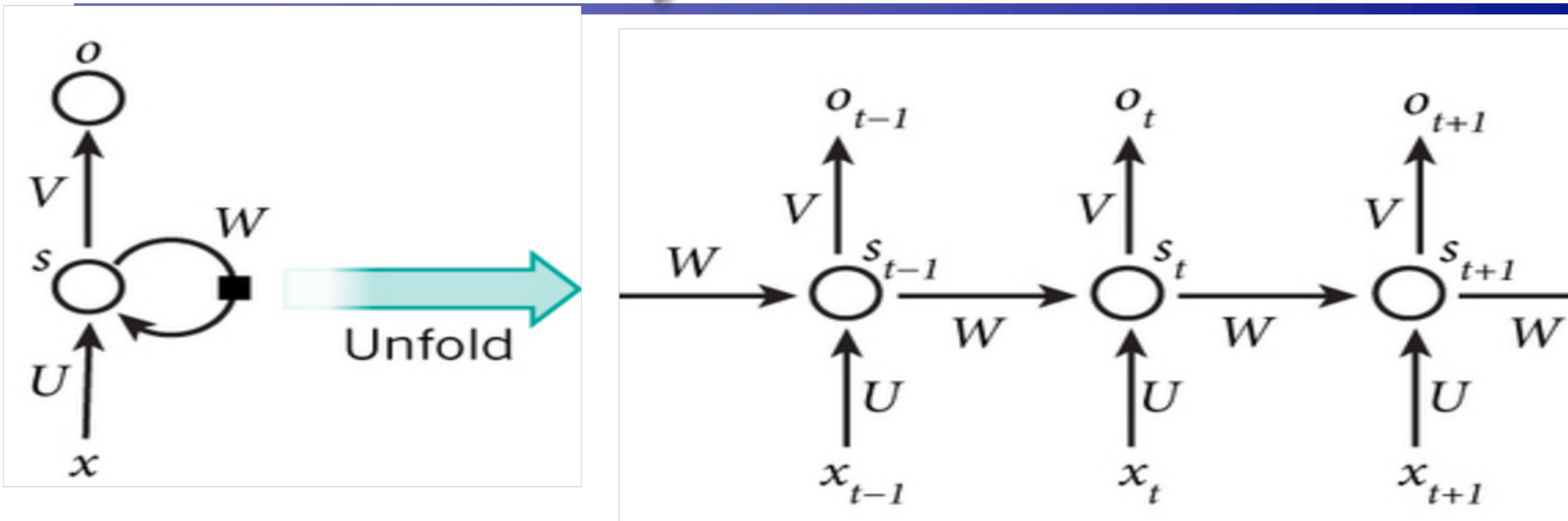


***x_t* = input** (e.g. a word)
***s_t* = hidden state**
***o_t* = output** (e.g. probability of the next word)
y_t = true value (e.g. ***x_{t+1}***)

Softmax $\sigma(\mathbf{z})$ transforms the K-dimensional real valued output ***z*** to a distribution – *like logistic regression*

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Like HMMs, unroll RNNs in time

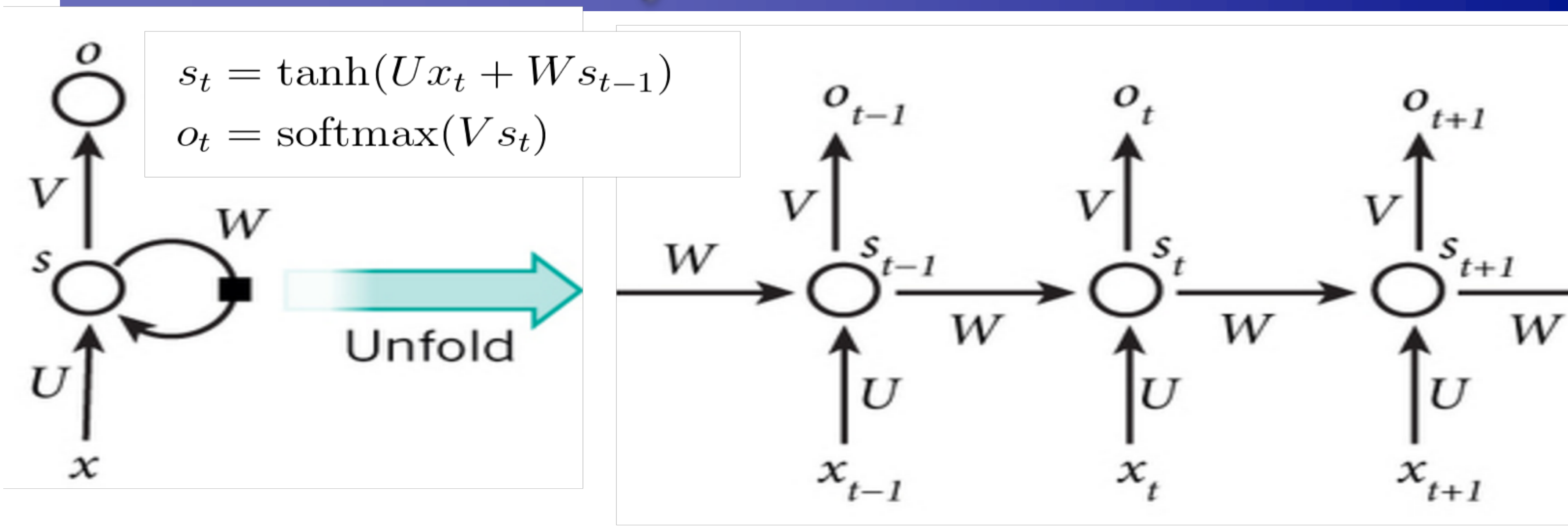


x_t = input (e.g. a word)
s_t = hidden state
o_t = output (e.g. probability of the next word)

http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html

Like HMMs, unroll RNNs in time

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

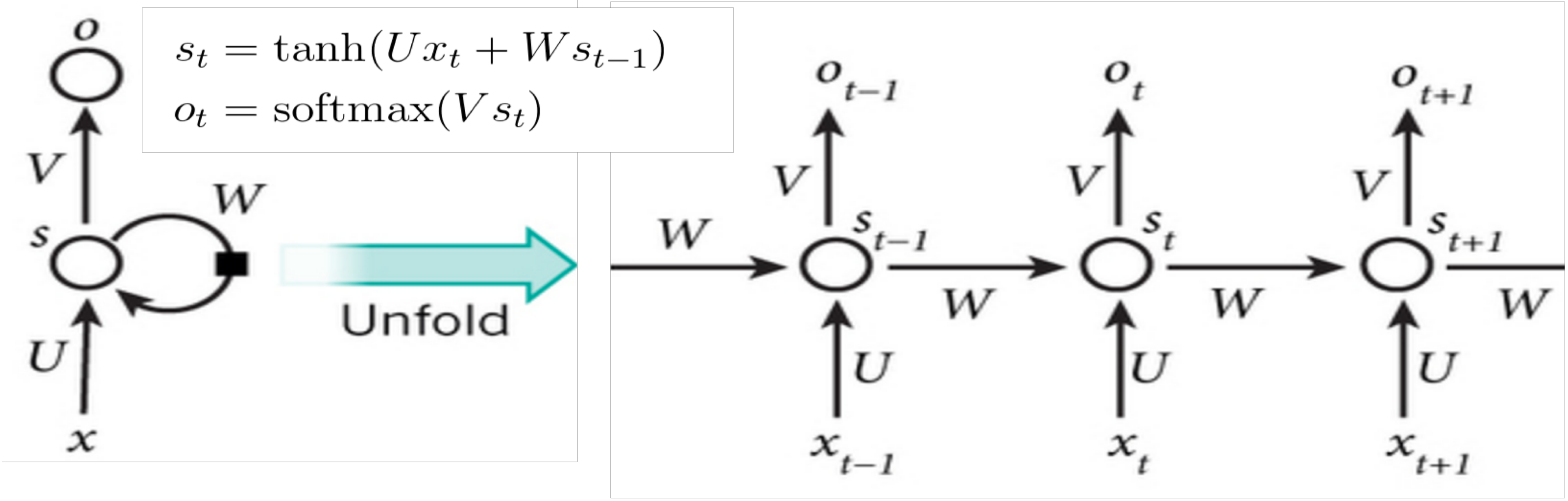


x_t = input (e.g. a word) - v
s_t = hidden state - k
o_t = output - v

What are the dimensions of *U, W, V*?
U *k*v* ***W*** *k*k* ***V*** *v*k*

Like HMMs, unroll RNNs in time

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$



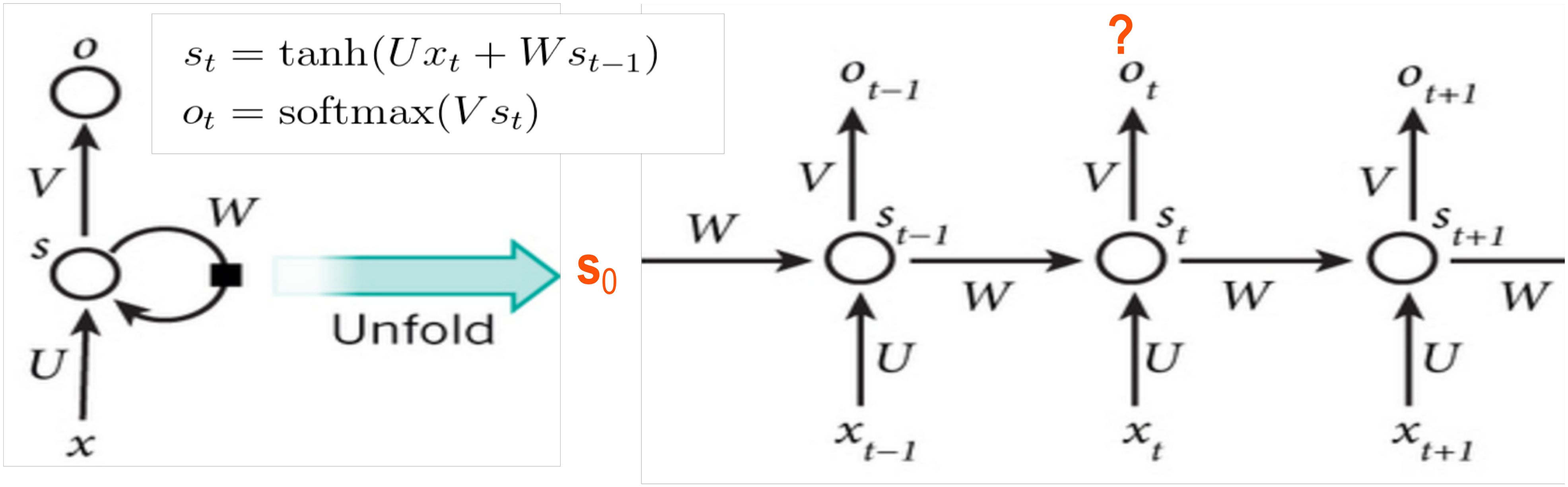
x_t = input (e.g. a word) - v
s_t = hidden state - k
o_t = output - v

What is the usual loss function?

$-\Sigma_t \log(\mathbf{o}_t[y_t])$ *- est. prob.of truth*

where $y_t=i$ gives the true label

Like HMMs, unroll RNNs in time

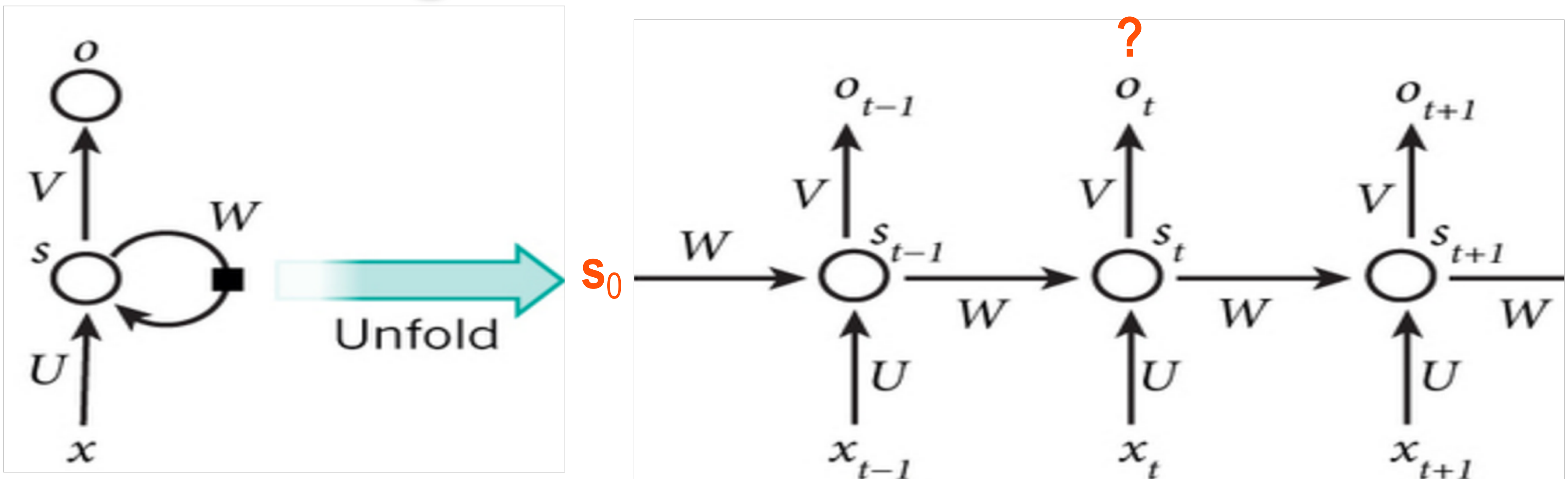


x_t = input - v
s_t = hidden state - k
o_t = output - v

If $s_{t-2} = s_0$, what is o_t in terms of s_0 and x?

$$o_t = \mathrm{softmax}(Vs_t) = \mathrm{softmax}(V \tanh(Ux_t + Ws_{t-1}))$$
$$= \mathrm{softmax}(V \tanh(Ux_t + W \tanh(Ux_{t-1} + Ws_{t-2})))$$

RNN gradients



$\mathbf{o}_t$ = *softmax*($\mathbf{V}$ *tanh*($\mathbf{Ux}_t + \mathbf{W}$ *tanh*($\mathbf{Ux}_{t-1} + \mathbf{Ws}_{t-2}$)

Observe $y_t = i$ What is the stochastic gradient step?

Err = -*log*($\mathbf{o}_t[i]$)

Find *d Err/d*$\mathbf{V}$, *d Err/d*$\mathbf{U}$, *d Err/d*$\mathbf{W}$

RNN Gradients

- $\mathbf{o}_t = softmax(\mathbf{V}\ tanh(\mathbf{Ux}_t + \mathbf{W}\ tanh(\mathbf{Ux}_{t-1} + \mathbf{Ws}_{t-2})$
- **Observe** $y_t = i$ What is the stochastic gradient step?
- $Err = -log(\mathbf{o}_t[i])$

$d\ Err/d\mathbf{V} = -(d\ log(\mathbf{o}_t[i]/d\mathbf{o}_t[i]) \quad d\mathbf{o}_t[i]/d\mathbf{V}$

$= -(1/\mathbf{o}_t[i]) \quad d\ softmax(\mathbf{z})/d\mathbf{z} \quad dz/d\mathbf{V}$

$\mathbf{z} = \mathbf{V}\ tanh(\mathbf{Ux}_t + \mathbf{W}\ tanh(\mathbf{Ux}_{t-1} + \mathbf{Ws}_{t-2})$

$d\ softmax(\mathbf{z})/dz_j = -1/(\Sigma_k e^{z_k})^2\ e^{z_j}\ e^{z_k}$ for k not equal to j

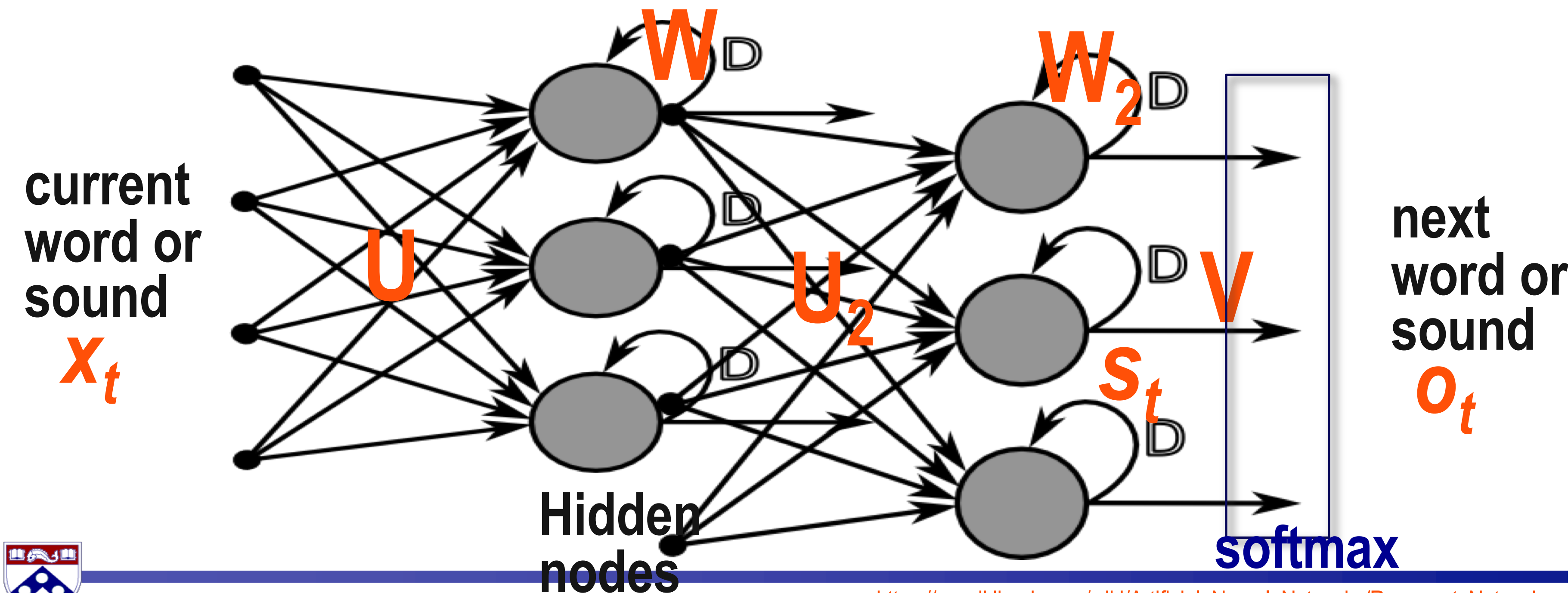
$= -1/(\Sigma_k e^{z_k})^2\ e^{2z_j} + e^{z_j}/(\Sigma_k e^{z_k})$ for k=j

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Recurrent Neural Nets (RNNs)

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

Can use multiple layers



Gated RNNs

- **Standard RNNs, like HMMs, tend to forget things exponentially fast**
- **Solution: Gated RNN**
  - Stores hidden state

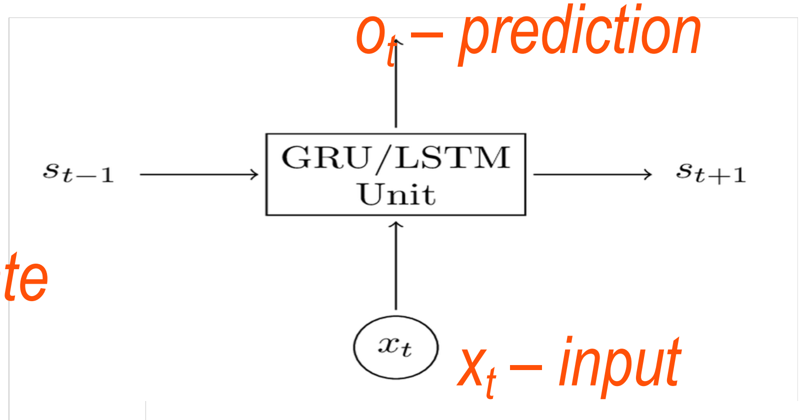
$z = \sigma(U^z x_t + W^z s_{t-1})$ *z: update gate*

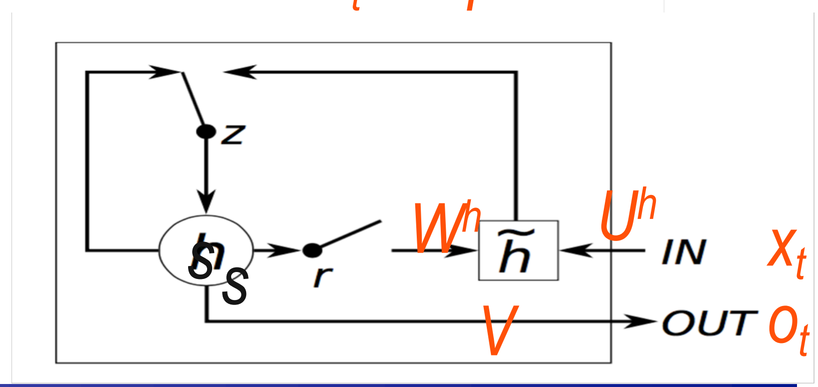
$r = \sigma(U^r x_t + W^r s_{t-1})$ *r: reset gate*

$h = \tanh(U^h x_t + W^h (s_{t-1} \circ r))$

$s_t = (1-z) \circ h + z \circ s_{t-1}$ *s_t : hidden state*

r=0 resets h
z=1 keeps state
z=0 updates it to h
r=1's, z=0's gives simple RNN
$\circ$ *is pointwise multiplication*



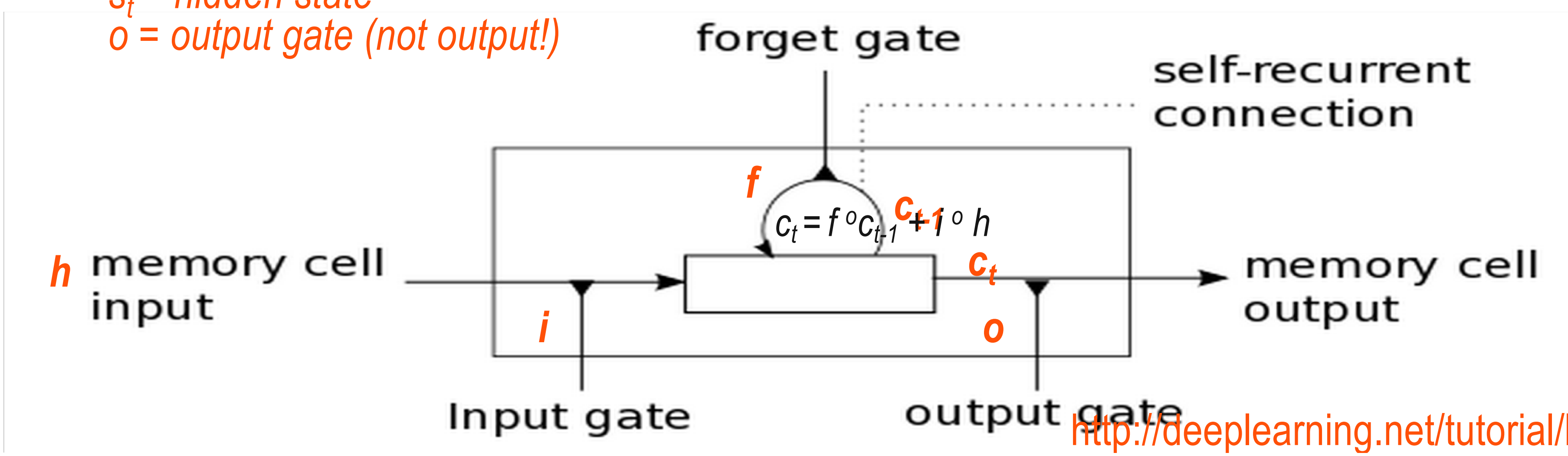


http://deeplearning.net/tutorial/lstm.html

Long Short Term Memory (LSTM)

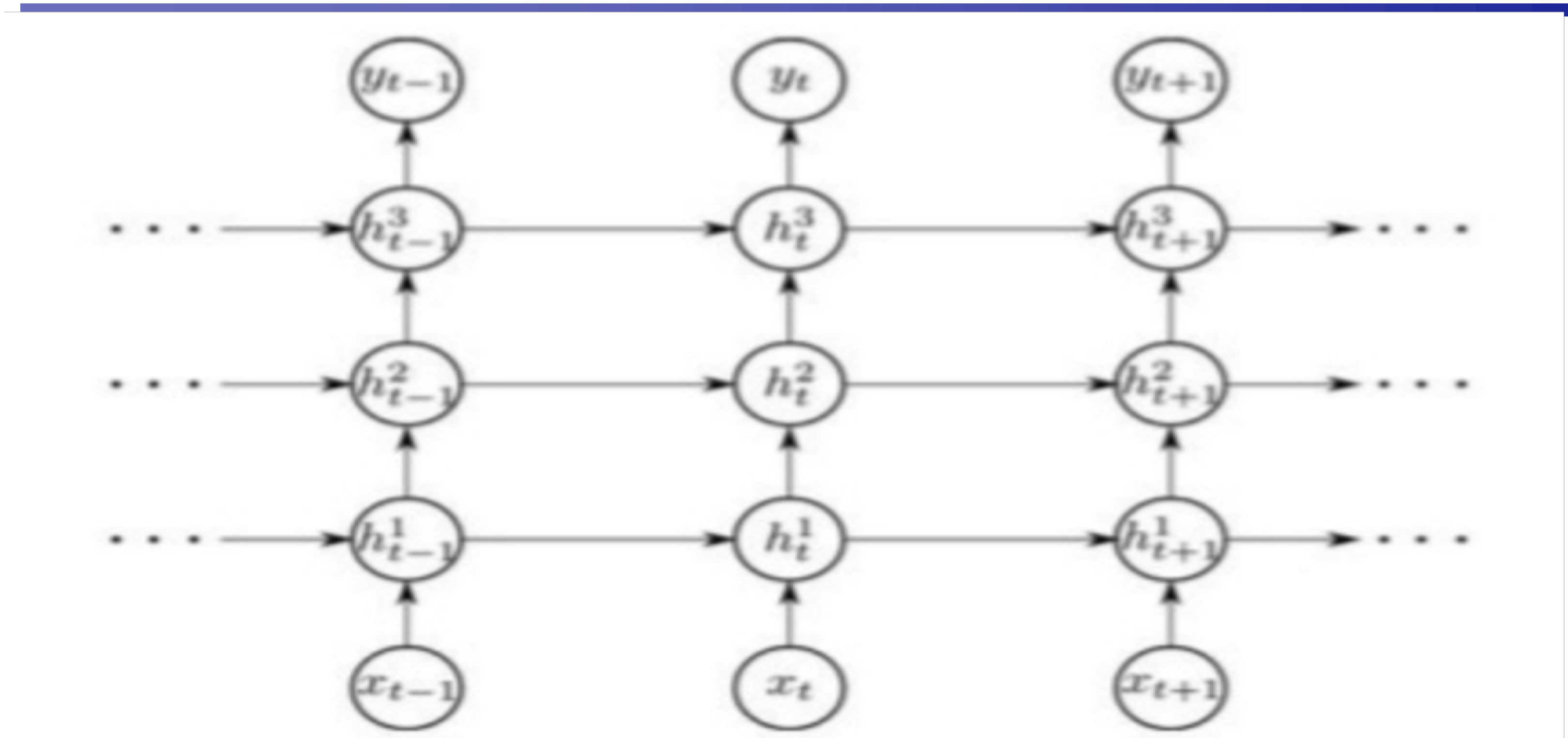
- **LSTM is a kind of gated RNN**
  - Just with more, different gates
  - Don't worry about what they are!!!

x_t – observation
s_t – hidden state
o = output gate (not output!)



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

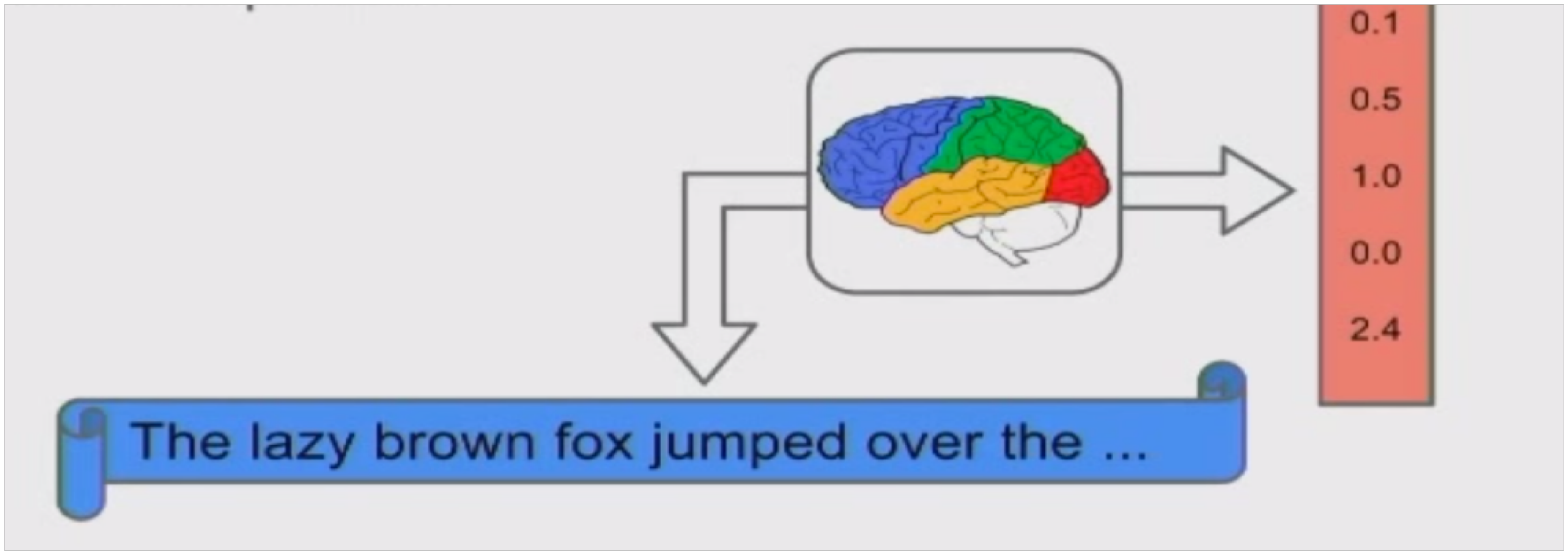
Recurrent Nets can be stacked



Recurrent Neural Nets

- **Predict a label for each observation**
  - $\mathbf{y}_t = f(\mathbf{x}_t, \mathbf{s}_t)$
- **Predict the next observation given past observations**
  - $\mathbf{y}_t = \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{s}_t)$
- **Or map one sequence to another sequence**
  - **An encoder**
    - sentence (sequence of words) to vector
  - **A decoder**
    - vector to sentence (sequence of words)

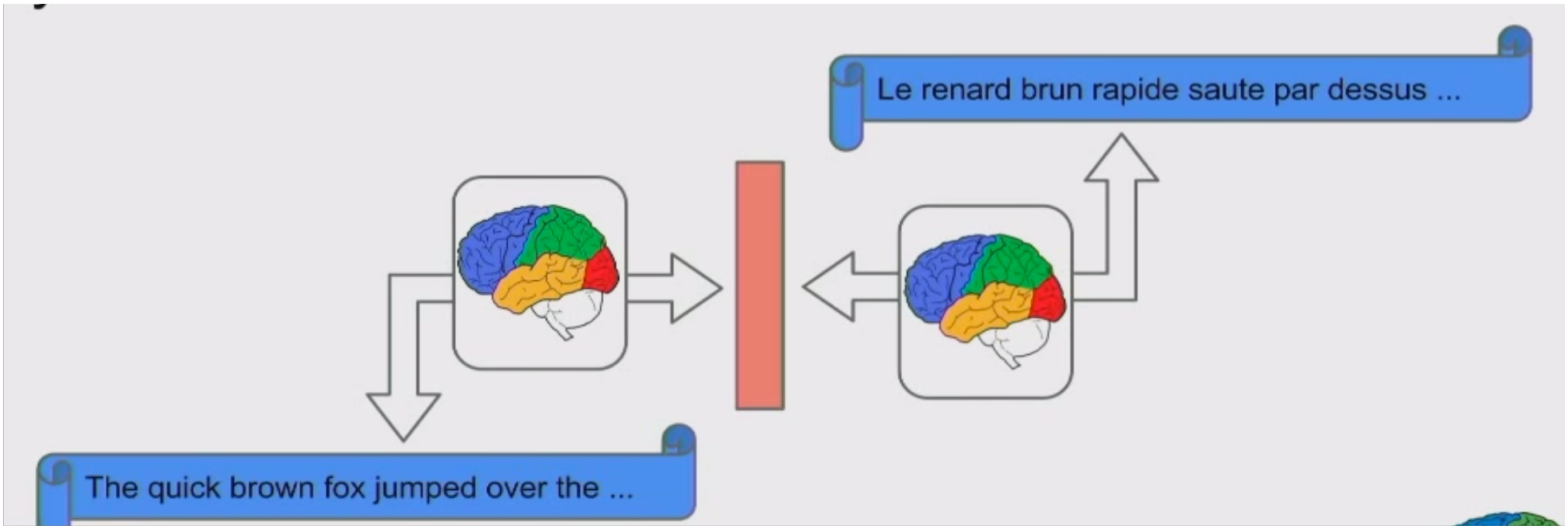
LSTM encodes a sentence



Jeff Dean, google

https://www.youtube.com/watch?v=90-S1M7Ny_o&spfreload=1

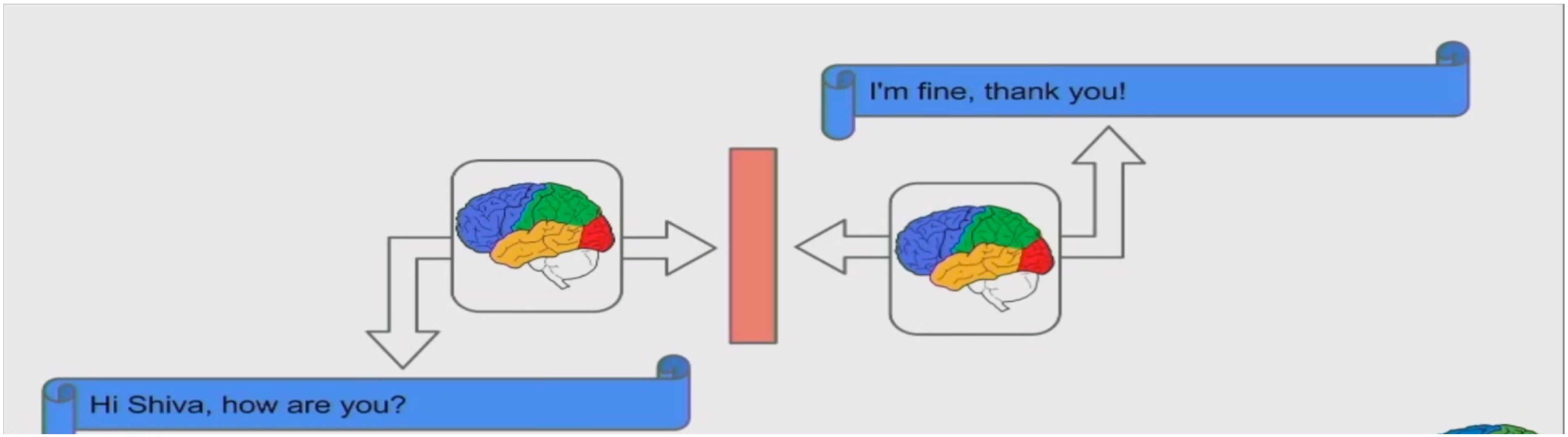
Sequence to sequence (Seq2seq)



Encode and Decode = translate

Jeff Dean, google

Seq2seq chatbot



Encode and Decode = chatbot

Jeff Dean, google

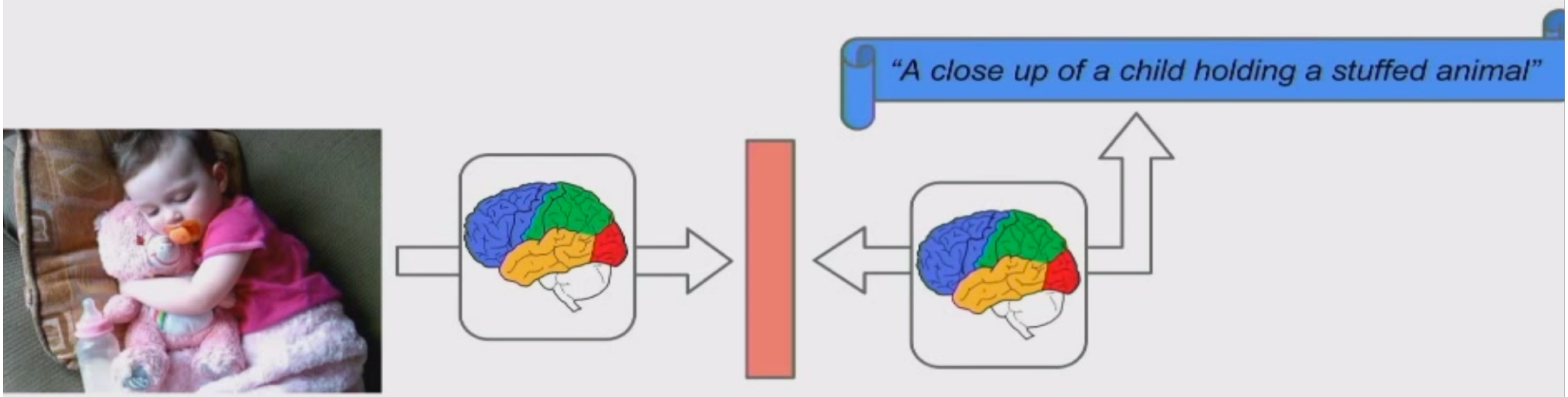
It works (sort of)

User> hello , i have a problem with my machine
BrainStop> hi
BrainStop> this is shiva .
BrainStop> how are you doing today ?
User> hi shiva , how are you ?
BrainStop> i ' m fine , thank you !
BrainStop> how may i assist you today ?
User> i want to access using vpn
BrainStop> currently the computer is connected to corp network ?
User> no
BrainStop> check this solution :
BrainStop> https : / / INTERNAL_HOSTNAME_REDACTED / articles / connecting - to - disconnecting - from - ssl - vpn # heading - 4
User> thanks , bye !
BrainStop> thank you .

Jeff Dean, google

Can map images to words

Connect sequence and image models, you get a captioning system



It works (sort of)

A man holding a tennis racquet on a tennis court.

Two pizzas sitting on top of a stove top oven

A group of young people playing a game of Frisbee

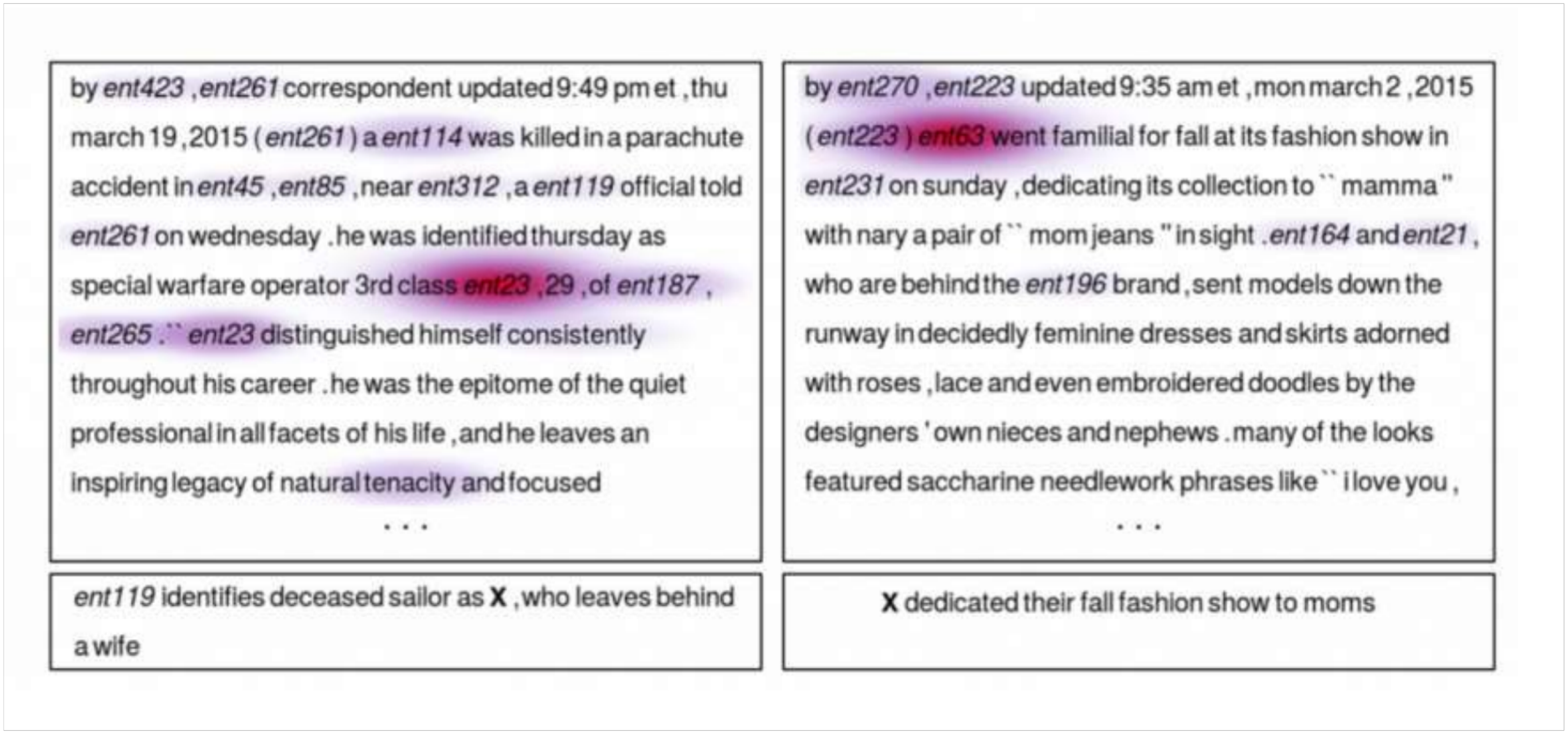
A man flying through the air while riding a snowboard

Jeff Dean, google

Language inputs to RNN

- Words ("one-hot")
- Characters ("one-hot")
- Word embeddings
  - Typically 300 dimensional

Attention-based Q&A



Dynamic Network Summary

- **Gated Neural Nets generalize HMMs, Kalman filters**
  - But are far more powerful!
- **They have replaced HMMs for speech to text and machine translation**
- **Lots of black magic “engineering”**
  - Unclear what matters about the network structure
    - Number and size of layers, regularization
    - Forms of gating (LSTM …), attention …
  - Gradient descent is tricky
- **Good software: tensorflow, pytorch …**