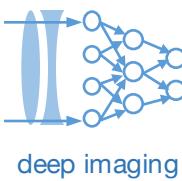


Lecture 21: Recurrent Neural Networks

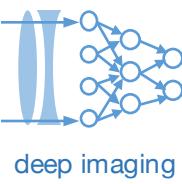
Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

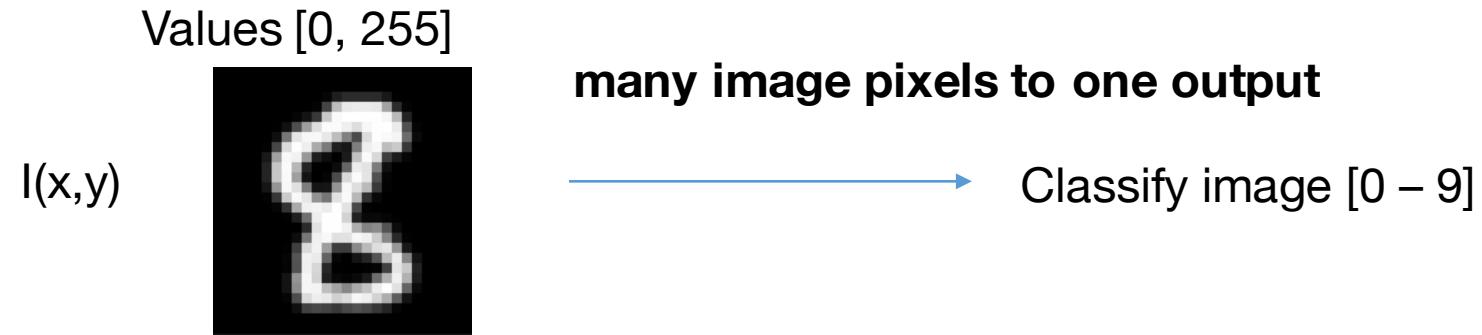


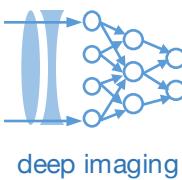
Material used to form this lecture:

- Deep Learning Book (deeplearningbook.org), Chapter 10
- Stanford CS231n, Lecture #10
- F. Visin et al., ReNet: A Recurrent Neural Network Base Alternative to Convolutional Networks
- K. He et al., Mask R-CNN
- S. Hochreiter and J. Schmidhuber, Long short-term memory

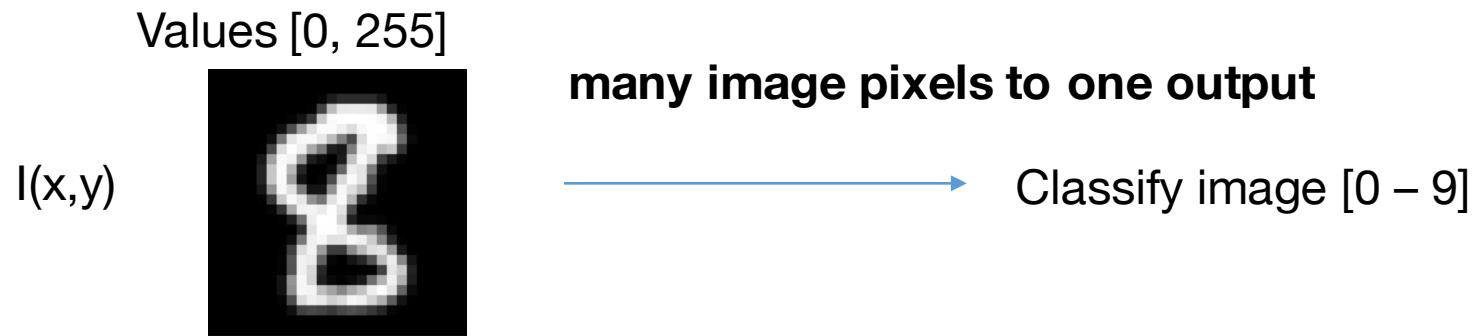


Convolutional neural networks versus recurrent neural networks



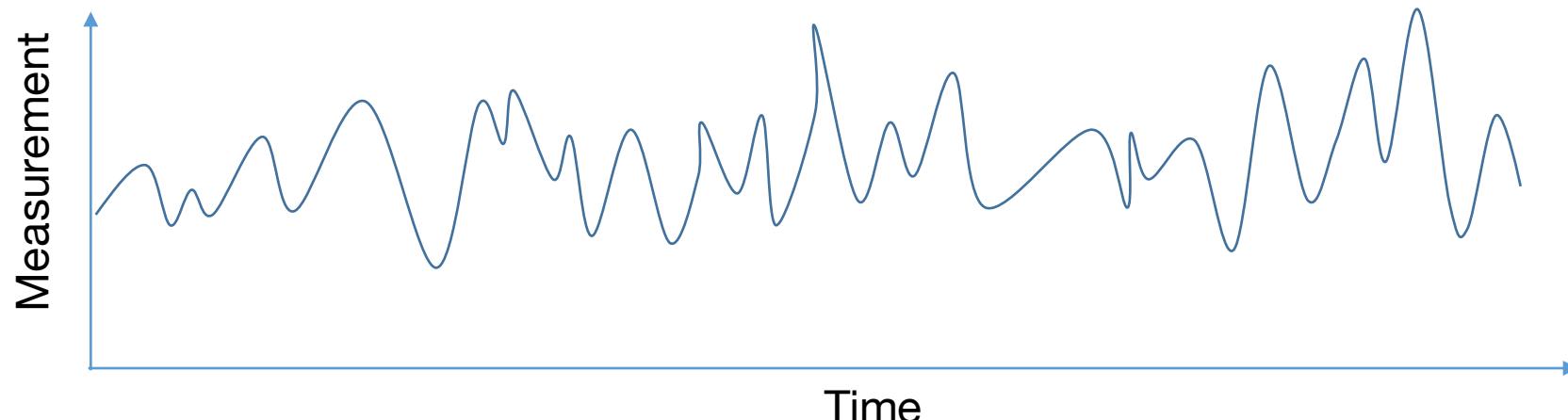


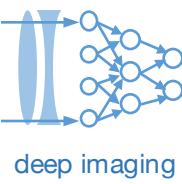
Convolutional neural networks versus recurrent neural networks



RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

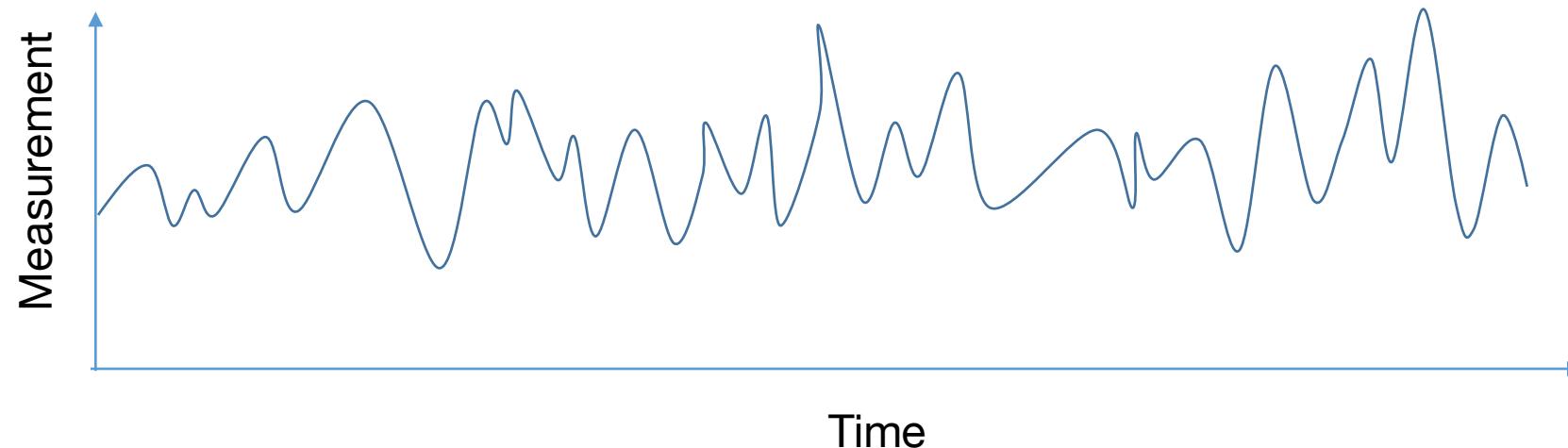


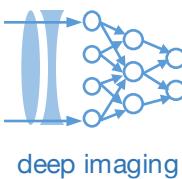


Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

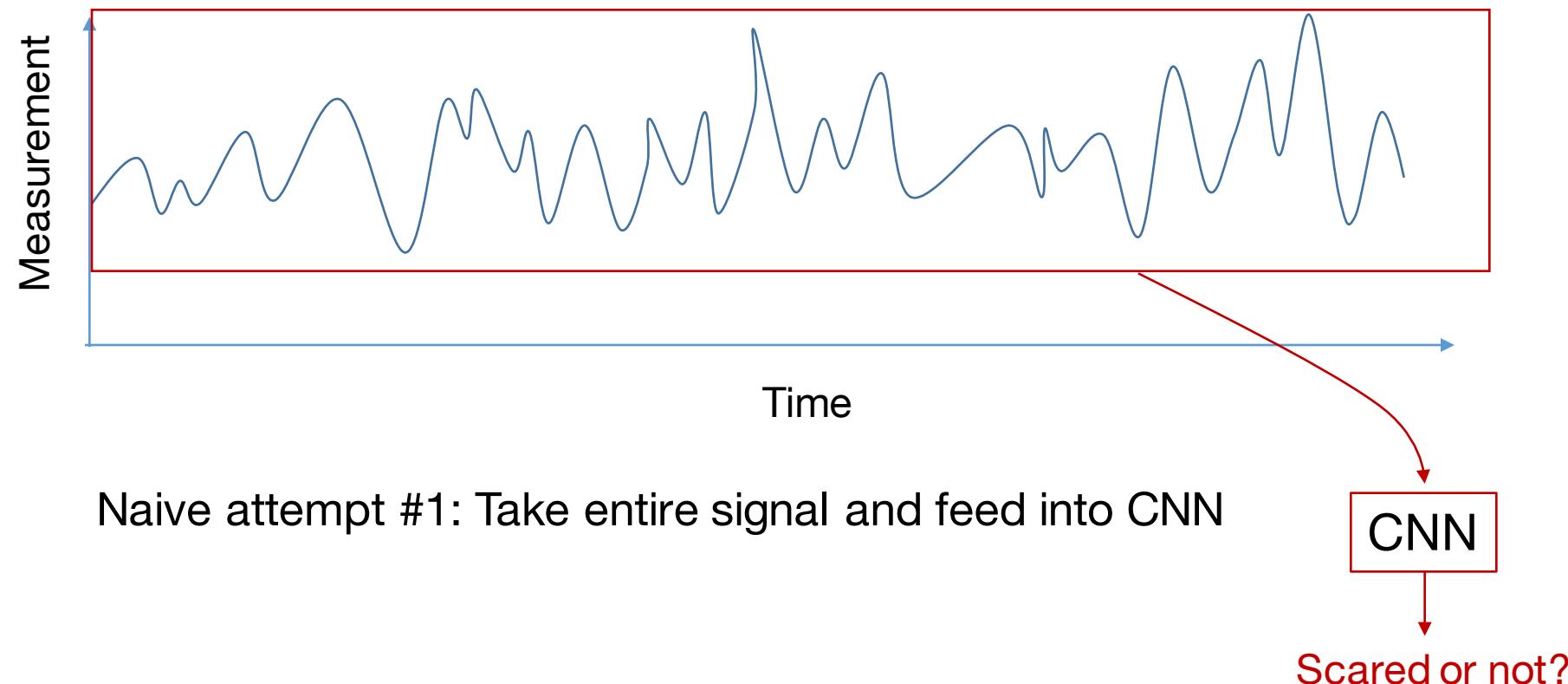




Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

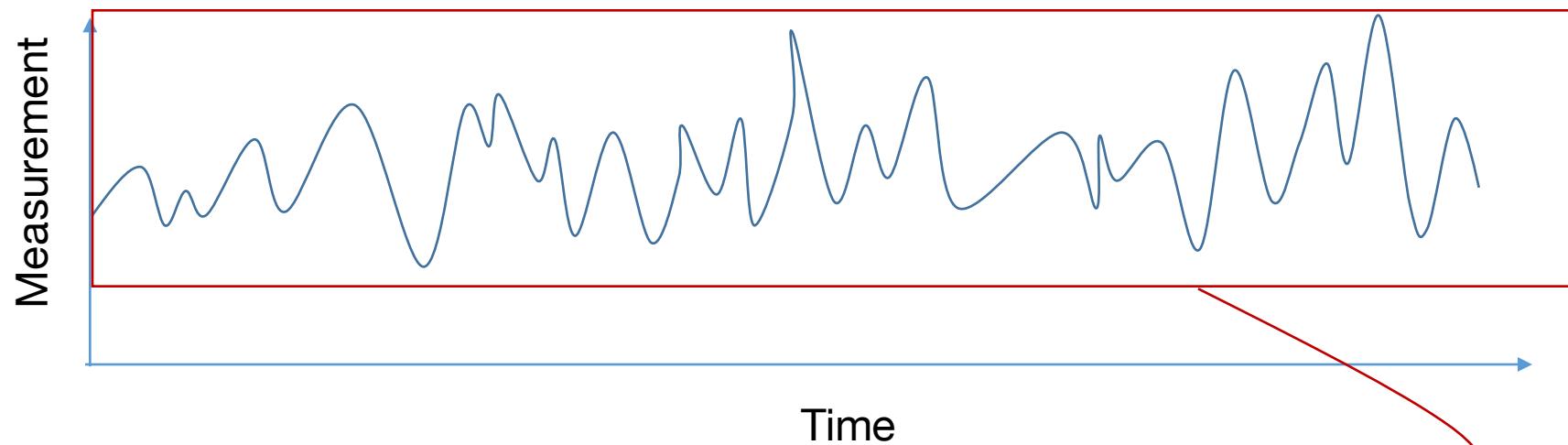
E.g., establish if mouse was scared from this EEG recording



Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

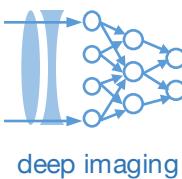
E.g., establish if mouse was scared from this EEG recording



Naive attempt #1: Take entire signal and feed into CNN

- Can have way too many entries to solve efficiently
- Embedded signal of interest may be at different moments

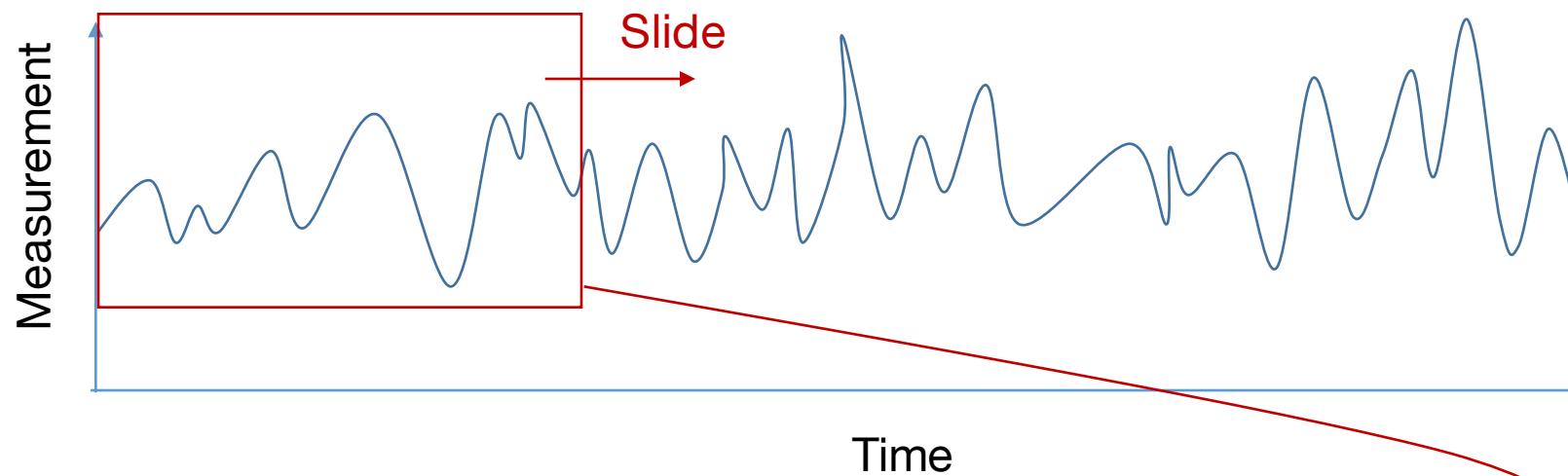




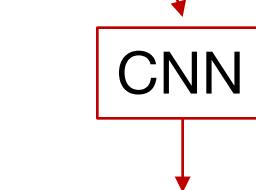
Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

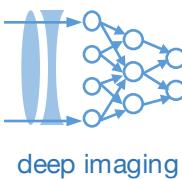


Naive attempt #2: Use a sliding window CNN



Scared or not?

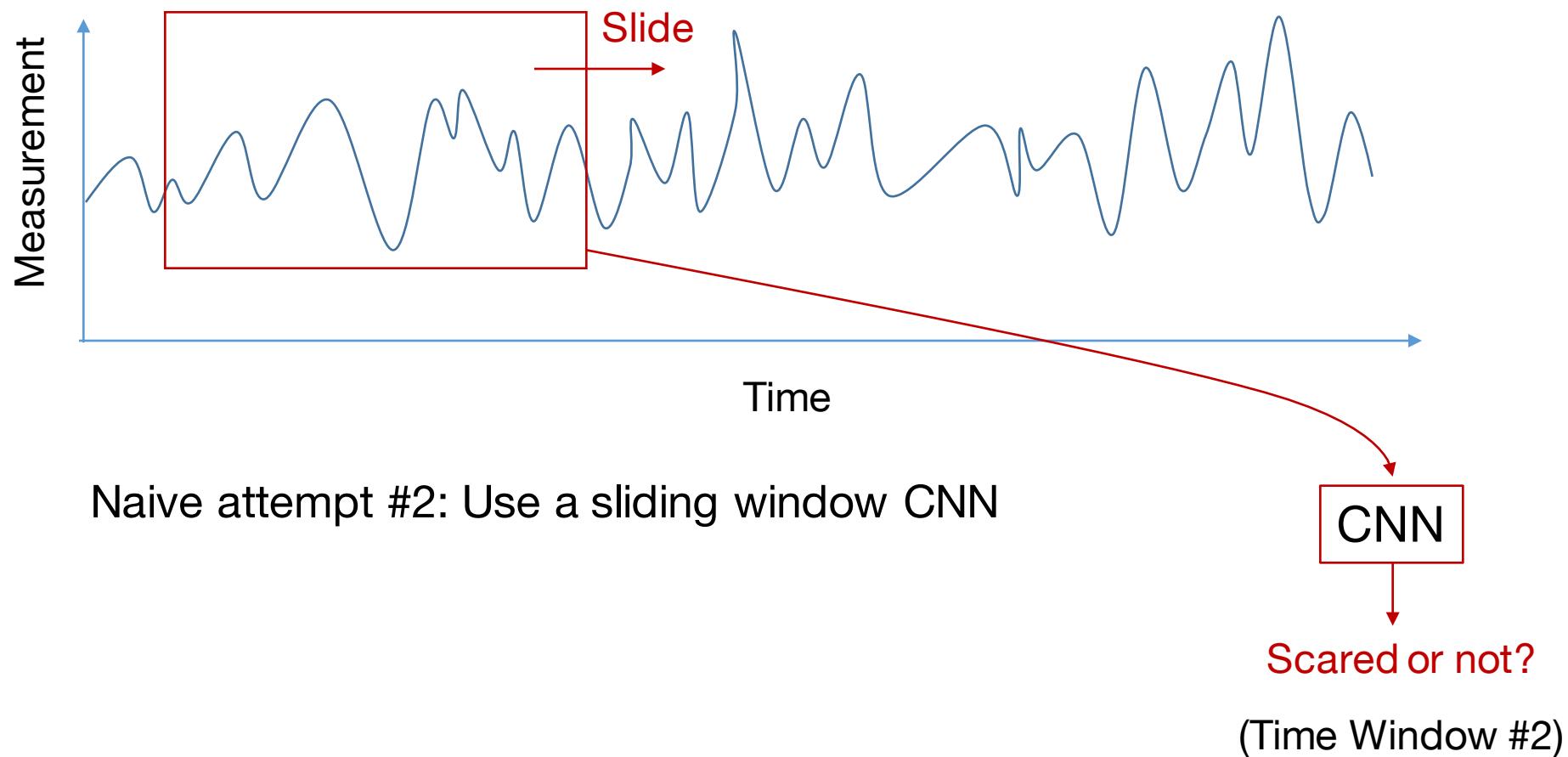
(Time Window #1)



Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

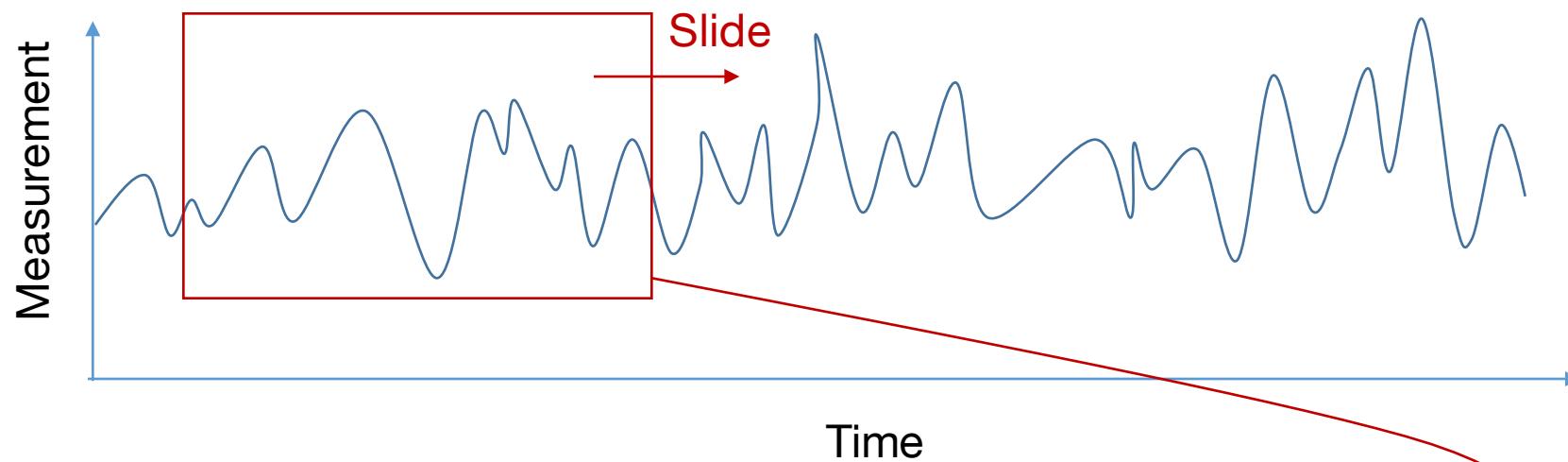
E.g., establish if mouse was scared from this EEG recording



Convolutional neural networks versus recurrent neural networks

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording



Naive attempt #2: Use a sliding window CNN

- Convolutions share features within window, but all features might not lie within window
- RNN's: share parameters across windows!

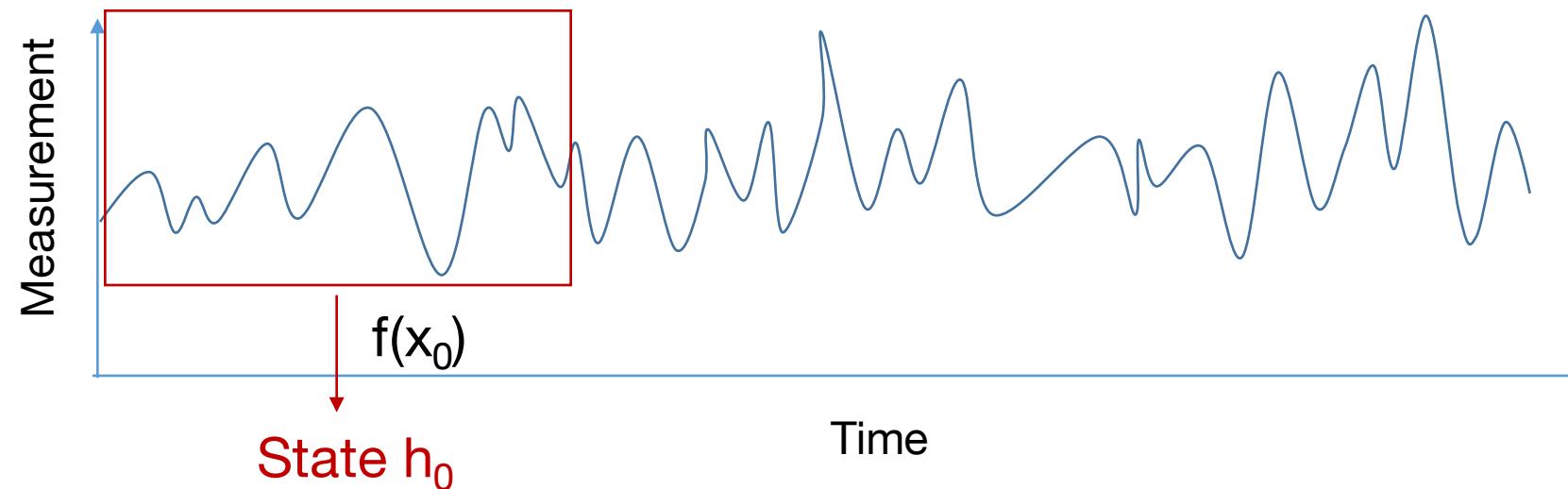
Scared or not?

(Time Window #2)

Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

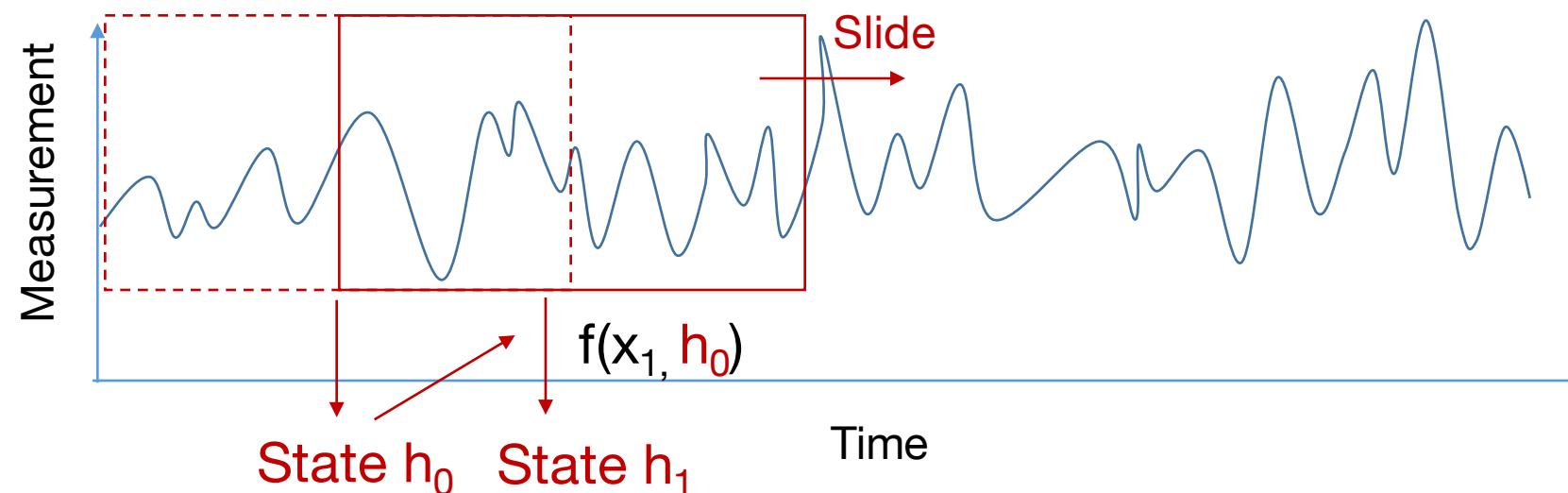


Recurrent neural networks: Generate states (“hidden units”) to use to inform subsequent decisions

Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

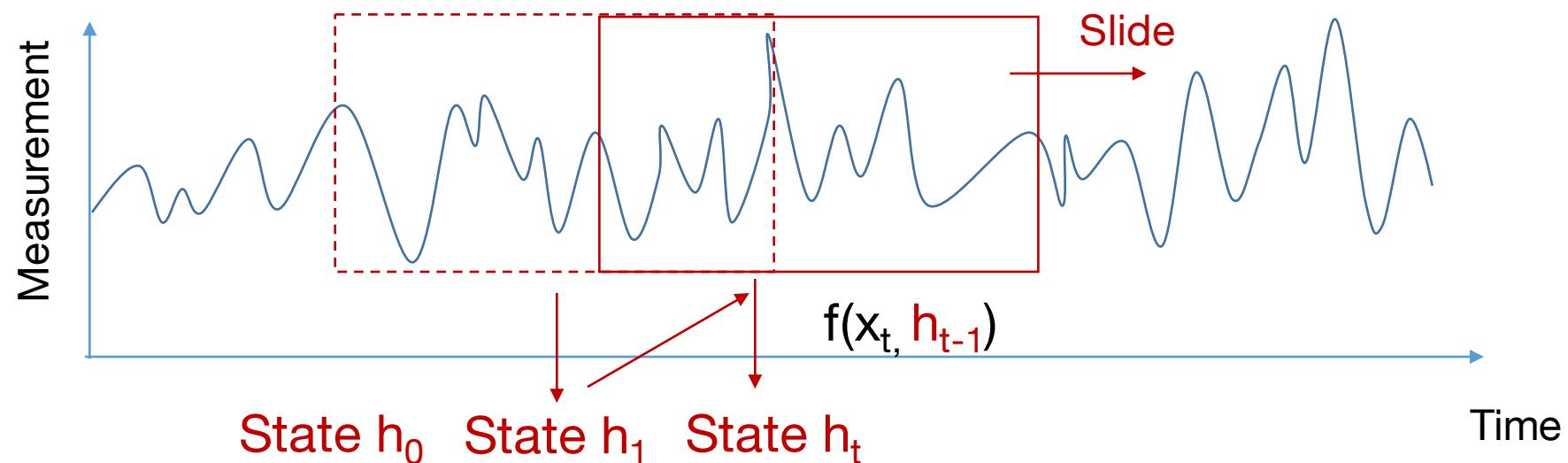


Recurrent neural networks: Generate states (“hidden units”) to use to inform subsequent decisions

Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording

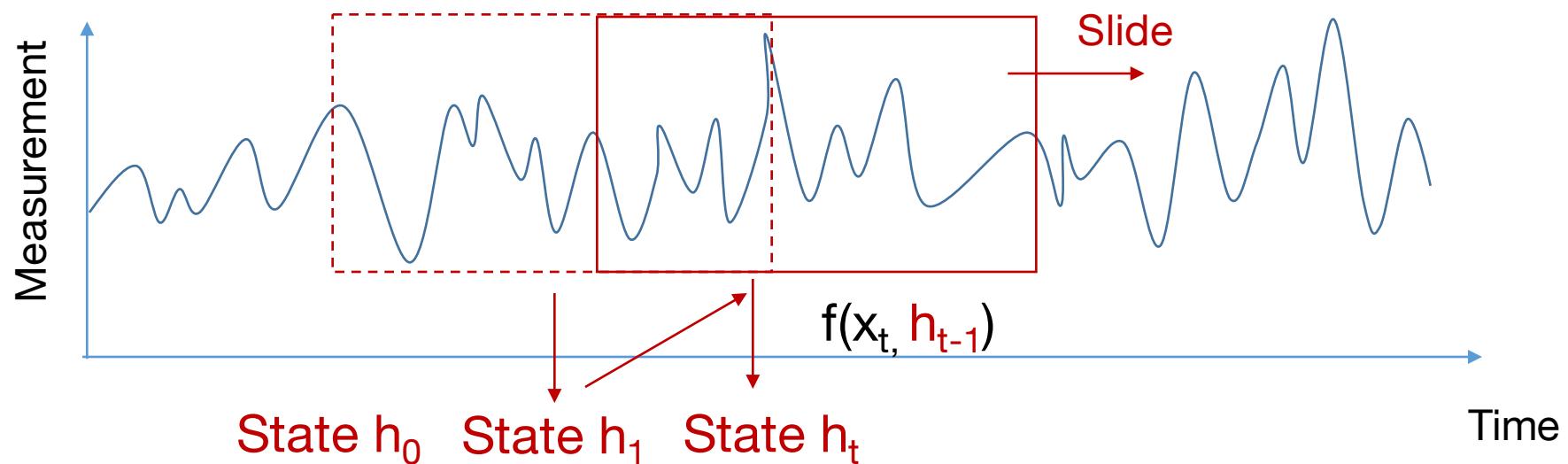


Recurrent neural networks: Generate states (“hidden units”) to use to inform subsequent decisions

Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording



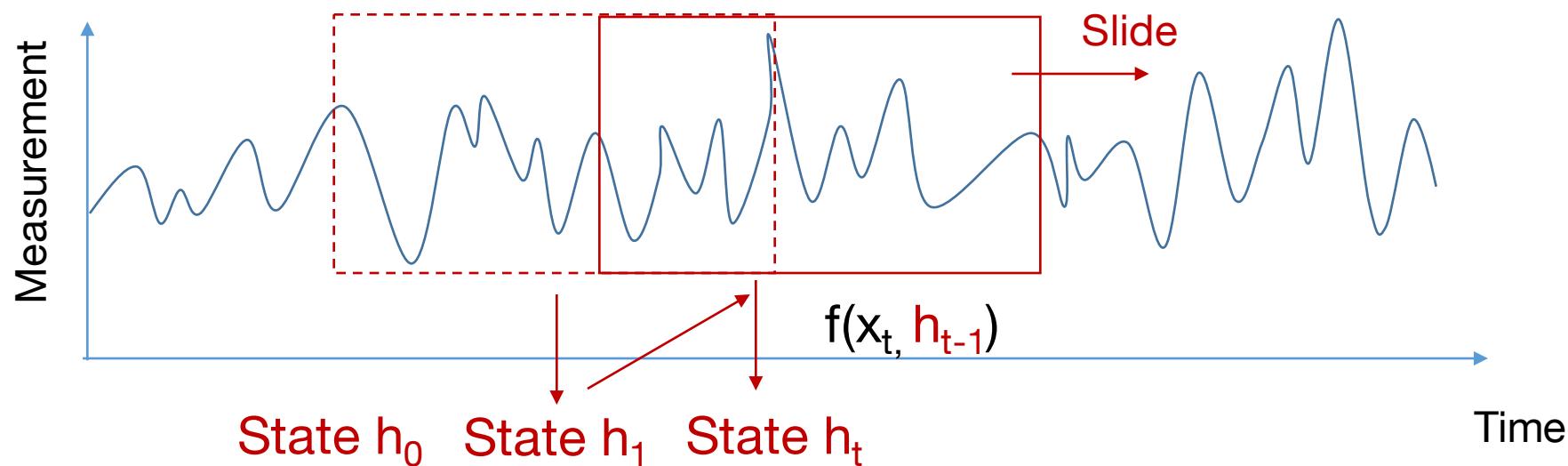
Reasoning unique to temporal data:

- Exploit preferential direction
- Helpful to establish a “memory” of what has been seen in the past
- Effectively learns how to daisy-chain information in signal

Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

E.g., establish if mouse was scared from this EEG recording



$$\begin{aligned}\mathbf{h}^{(t)} &= g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)}, \dots, \mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \\ &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta).\end{aligned}$$

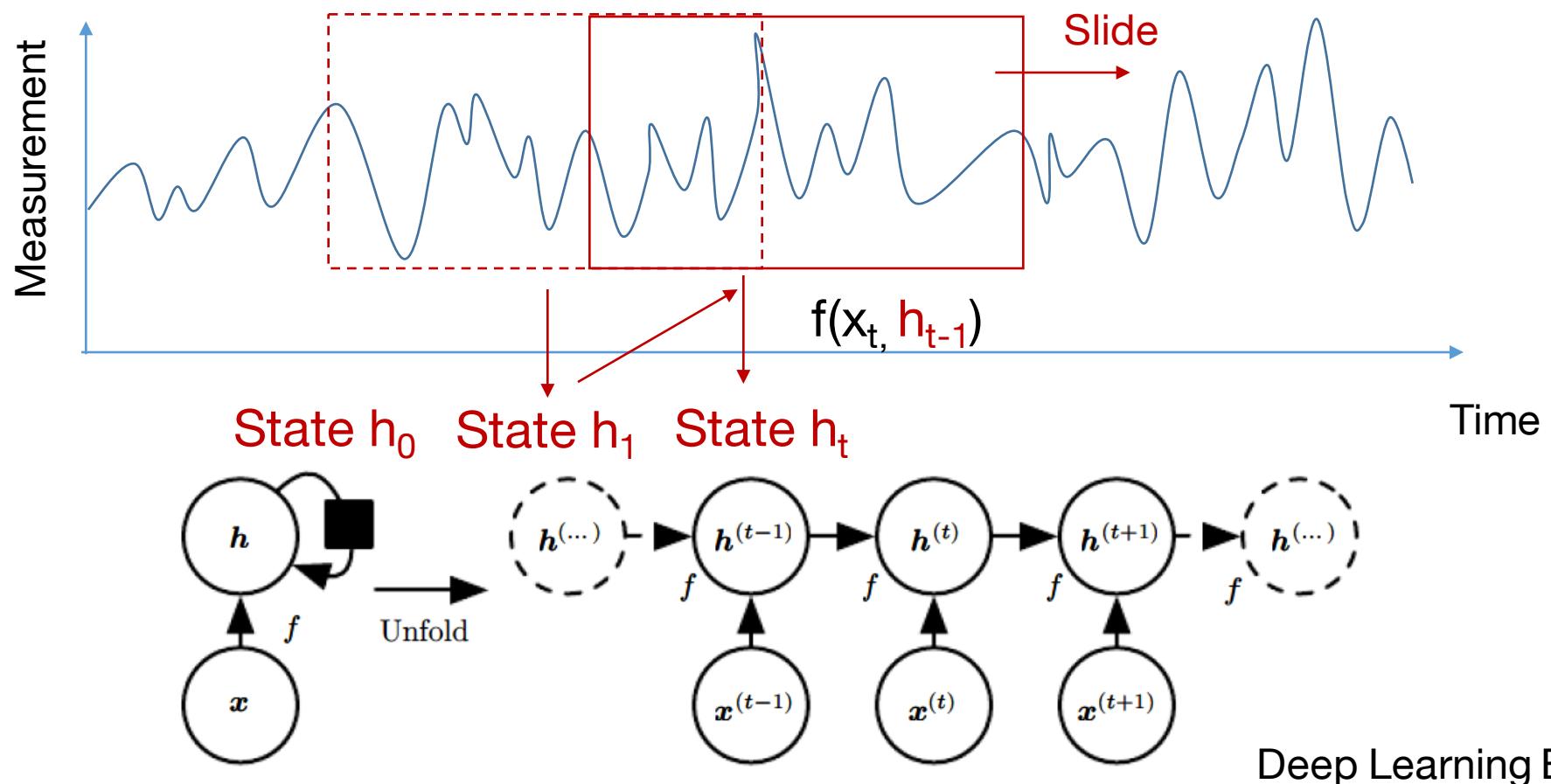
Recursive structure can be unfolded

Deep Learning Book, Ch. 10

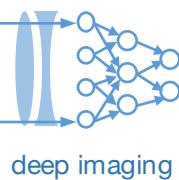
Recurrent neural networks in a nutshell

RNN's: Examine signals as a function of time

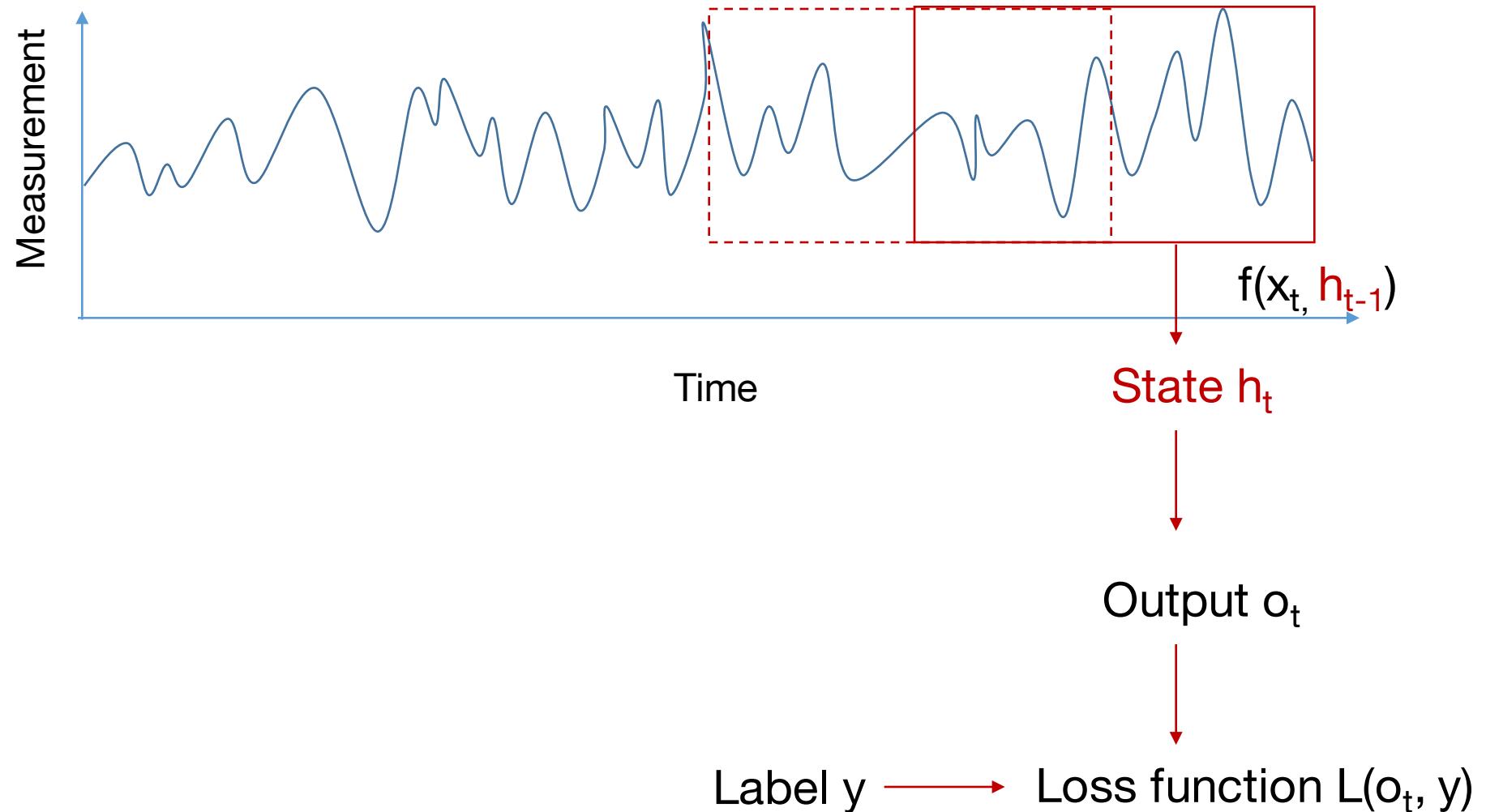
E.g., establish if mouse was scared from this EEG recording

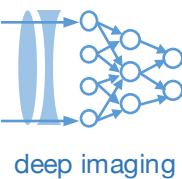


Deep Learning Book, Ch. 10

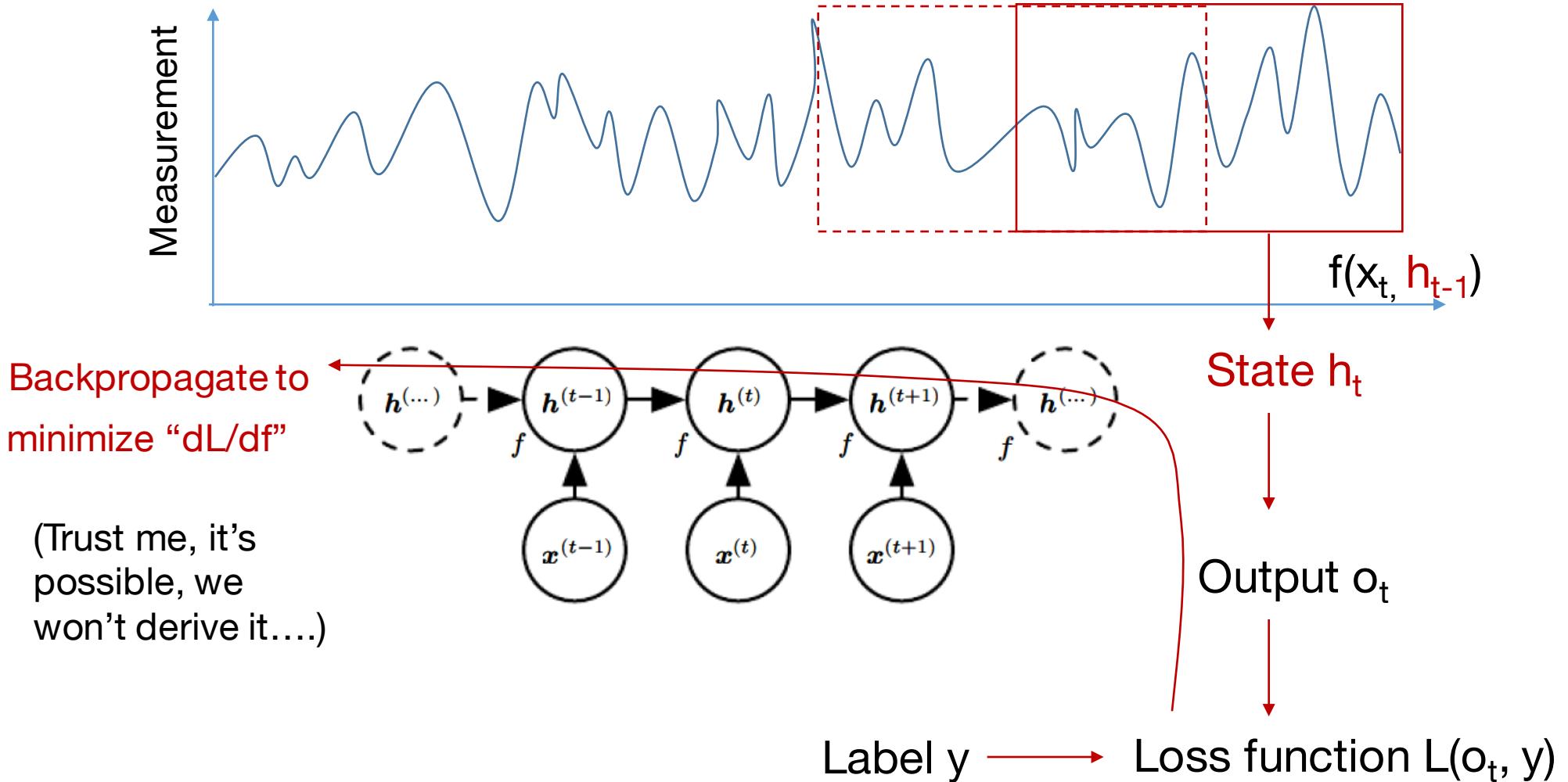


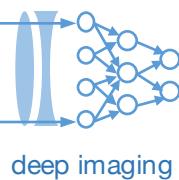
Many-to-many recurrent neural network



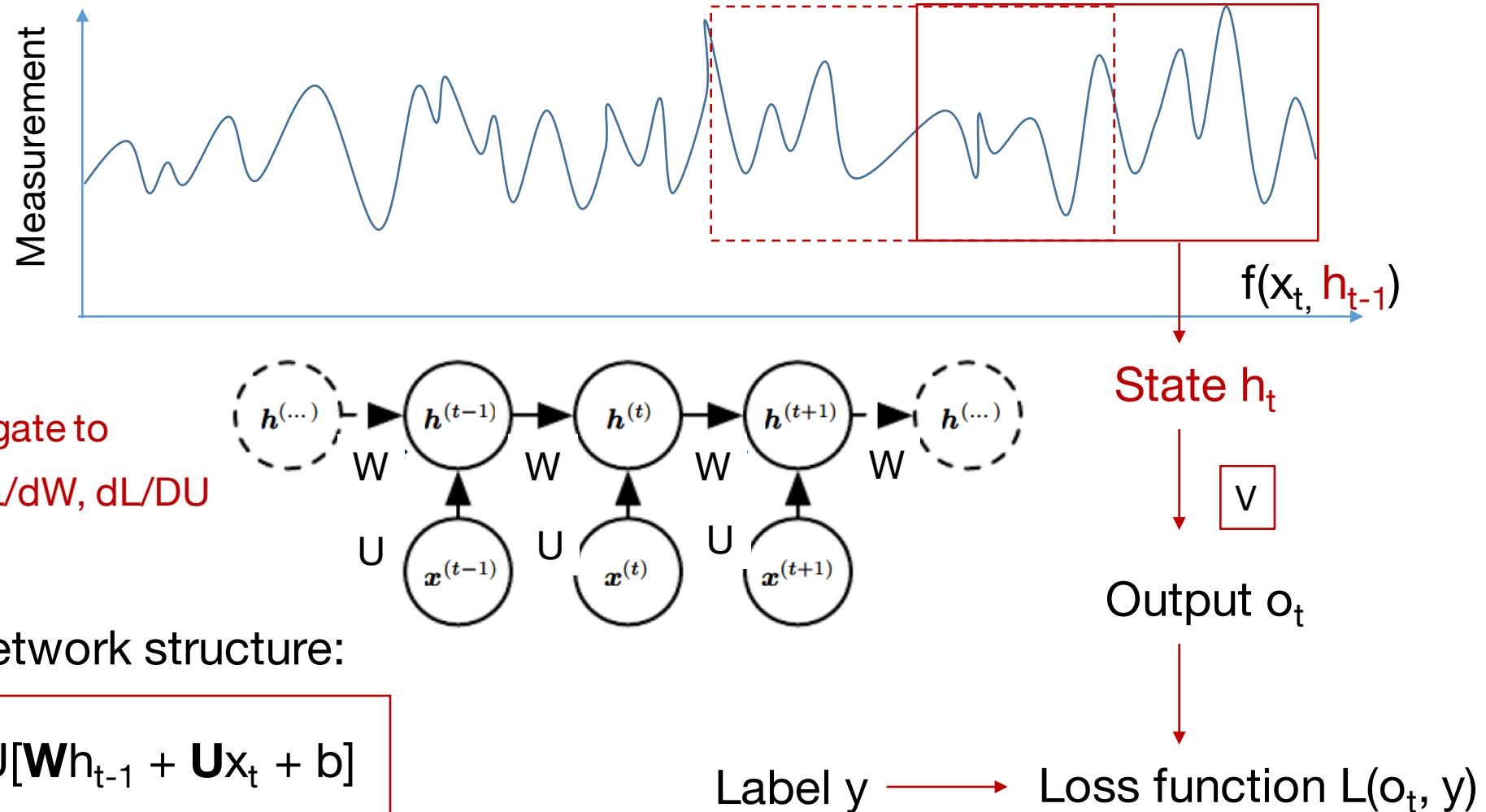


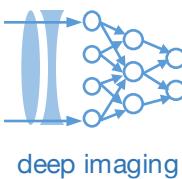
Many-to-many recurrent neural network



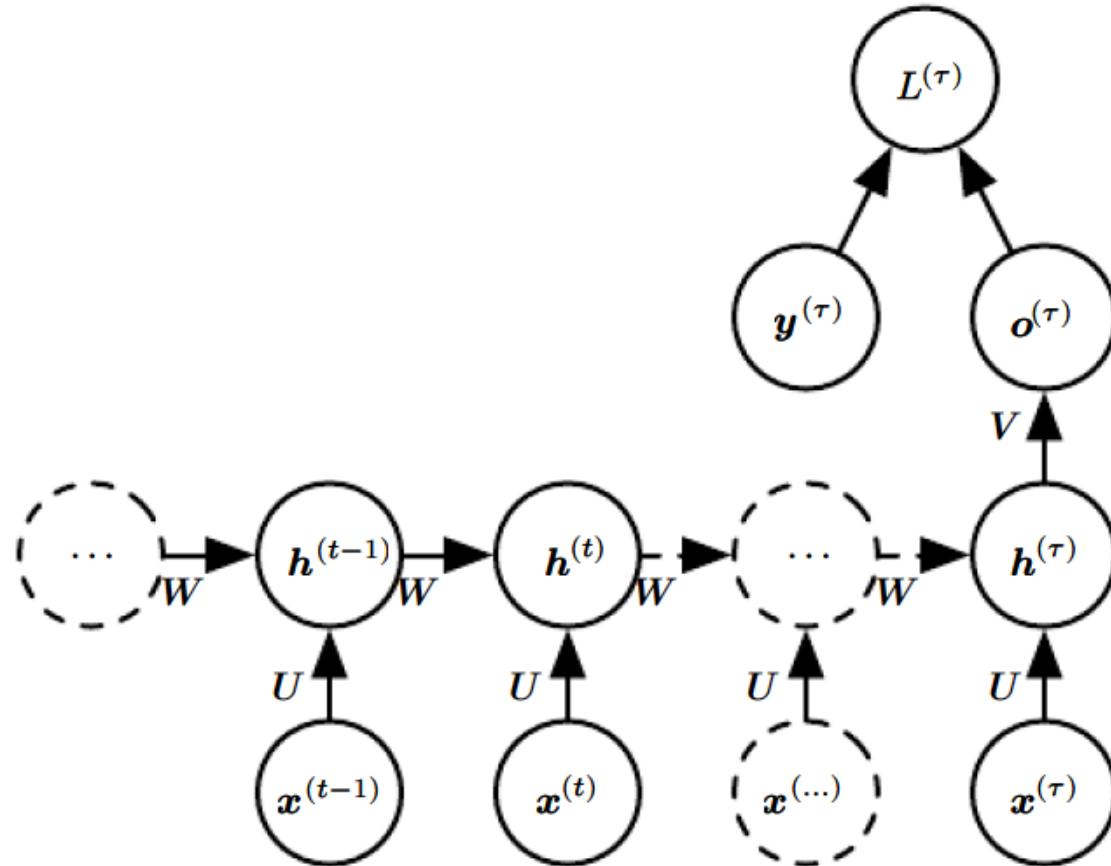


Many-to-many recurrent neural network





Many-to-one recurrent neural network



Learn fixed W and U from n sequences x and labels y

An example use case:

“I went to Nepal in 2009.”

“In 2009, I went to Nepal.”

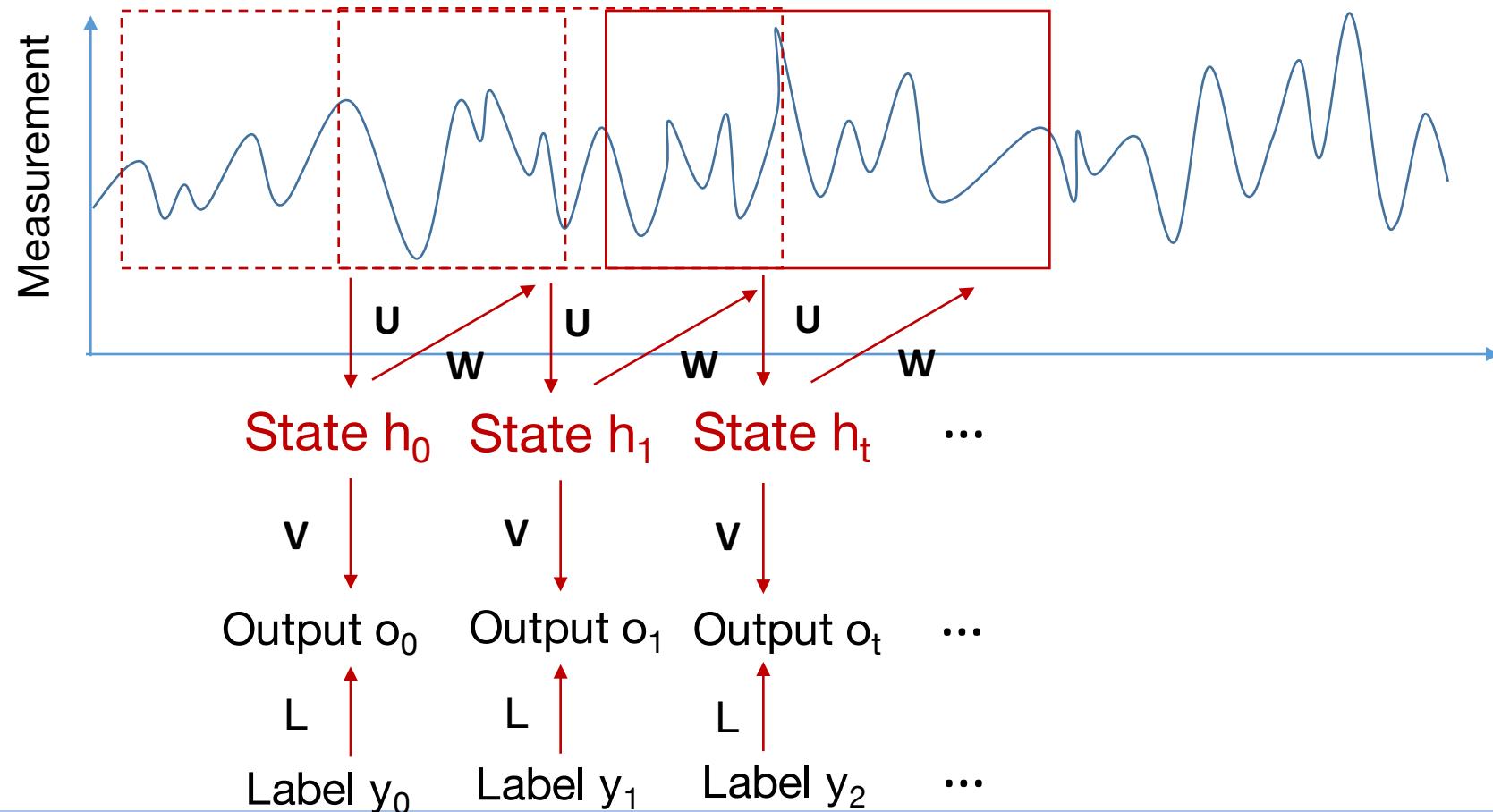
Goal: Extract year each writer went to Nepal from lots of sentences

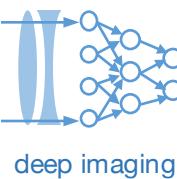
- 2009 is 2nd and 6th word in sentence
- Separated by 1 word and then 3 words

Deep Learning Book, Ch. 10

Many-to-many recurrent neural network

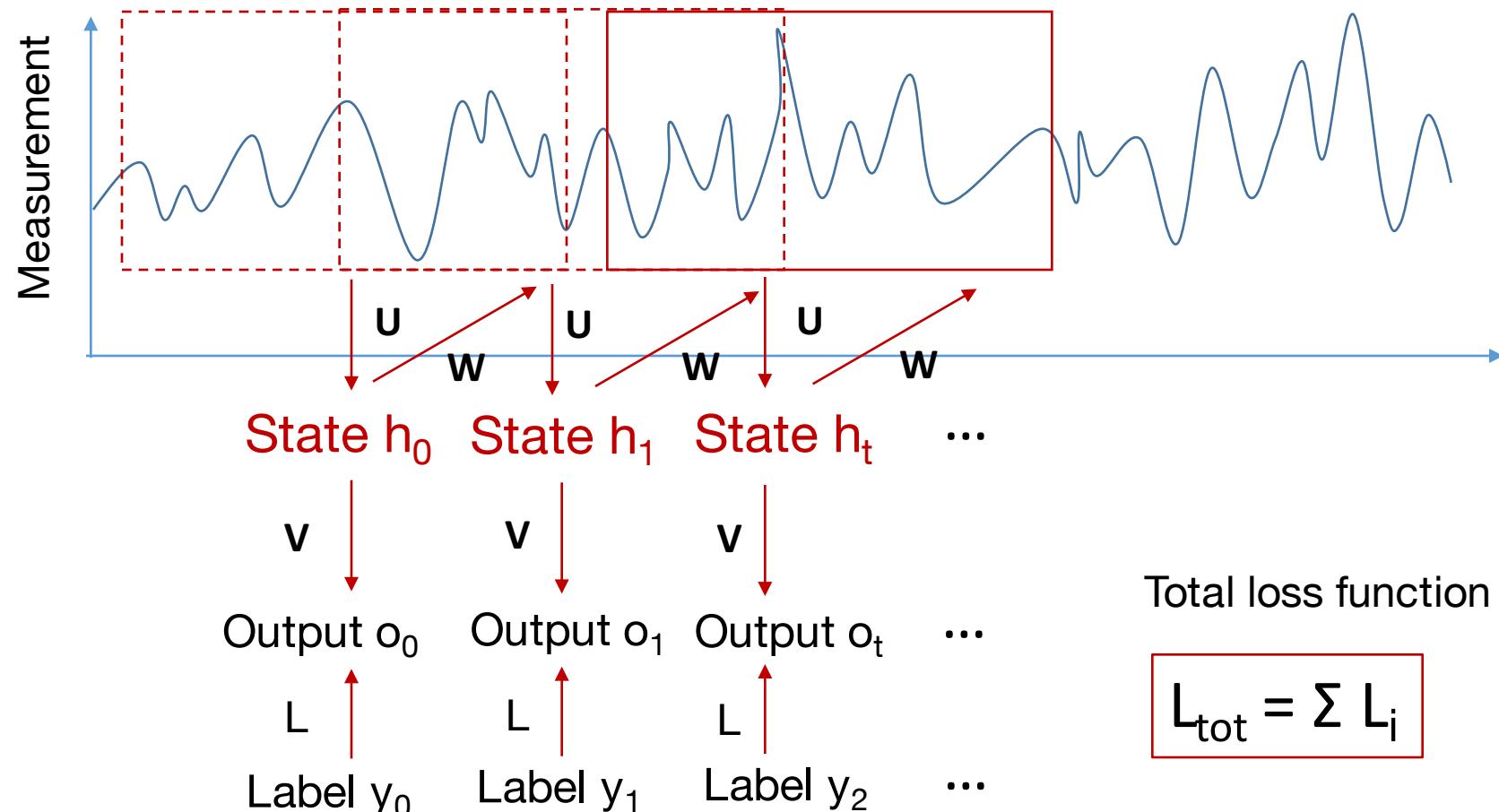
Instead of having one output at the end, can have a trainable output at each step

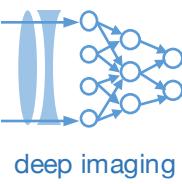




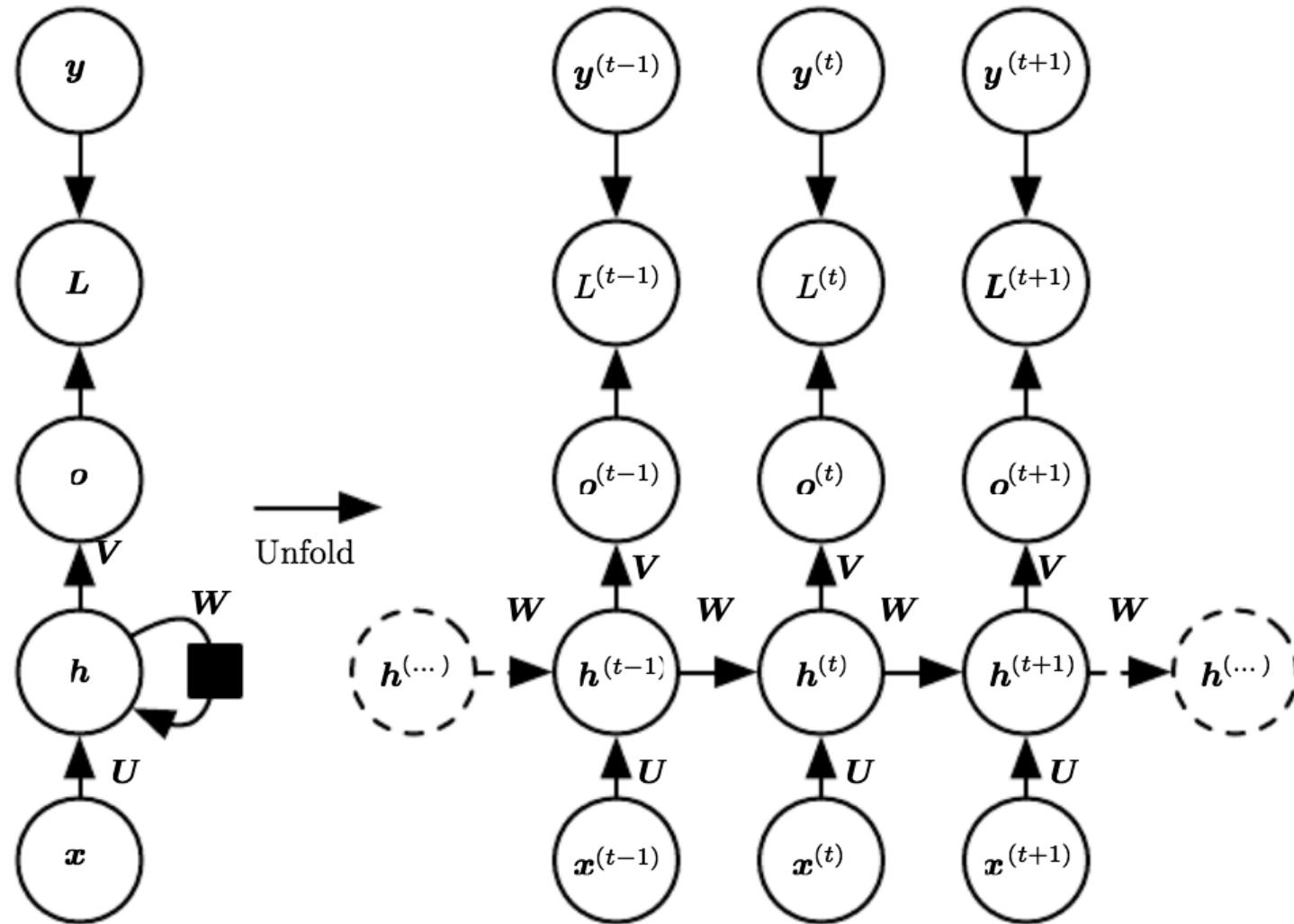
Many-to-many recurrent neural network

Instead of having one output at the end, can have a trainable output at each step





Many-to-many recurrent neural network

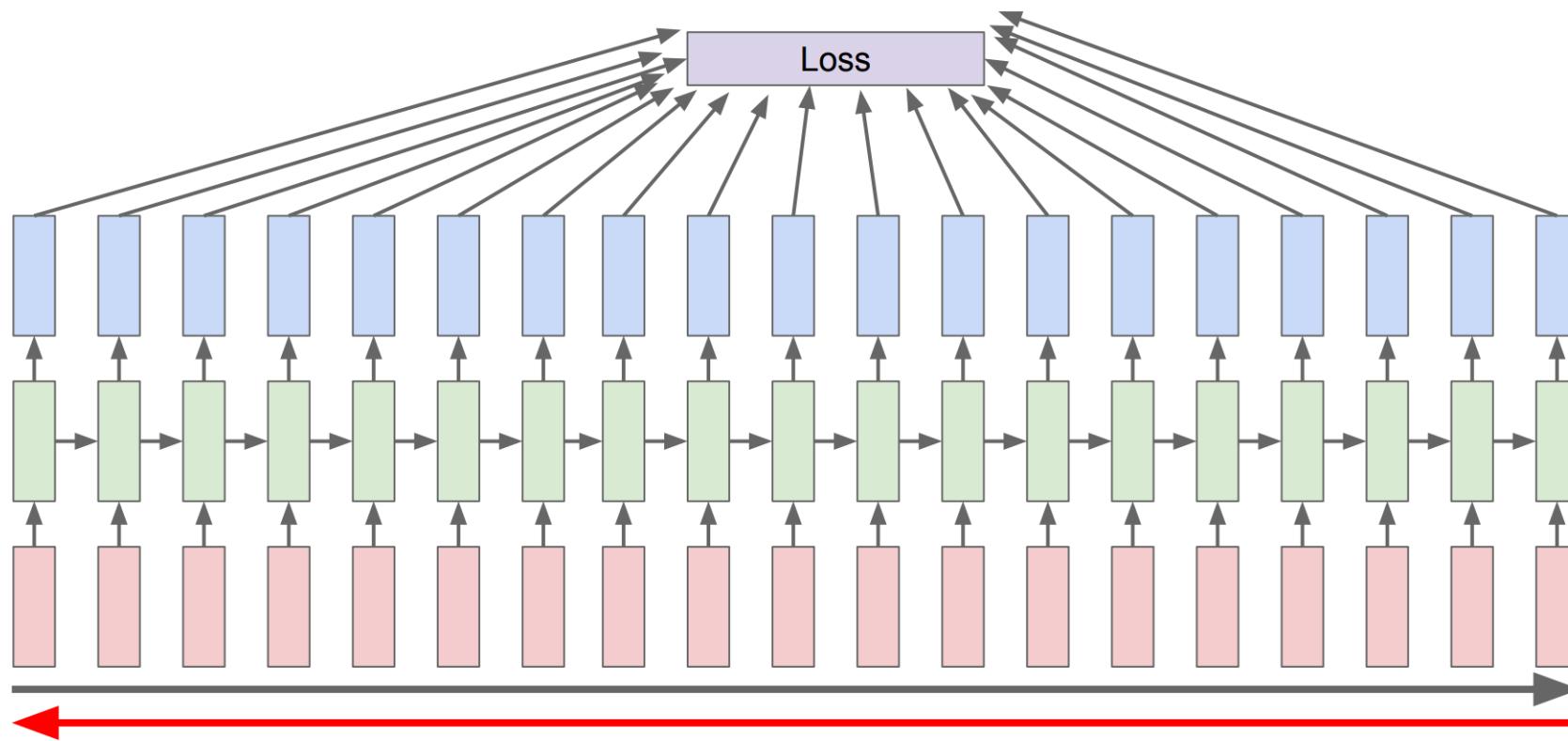


$$\begin{aligned}
 \mathbf{a}^{(t)} &= \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}, \\
 \mathbf{h}^{(t)} &= \tanh(\mathbf{a}^{(t)}), \\
 \mathbf{o}^{(t)} &= \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}, \\
 \hat{\mathbf{y}}^{(t)} &= \text{softmax}(\mathbf{o}^{(t)}),
 \end{aligned}$$

Deep Learning Book, Ch. 10

Several options to treat loss function in many-to-many case

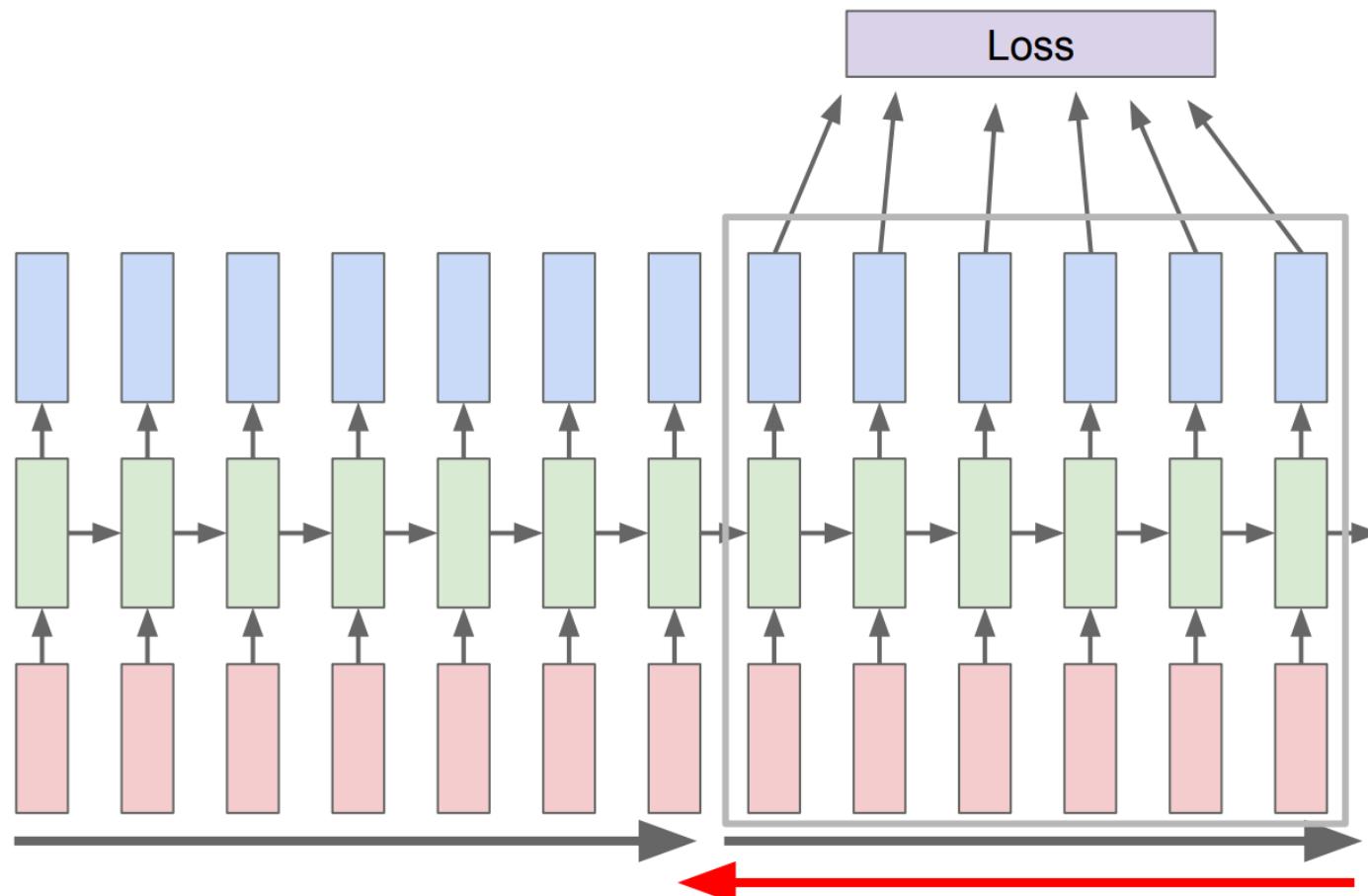
Option #1: Run through full sequence, go back all the way to compute gradient



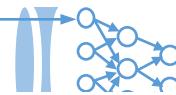
From Stanford CS231n Lecture 10 slides

Several options to treat loss function in many-to-many case

Option #1: Run through chunks at a time



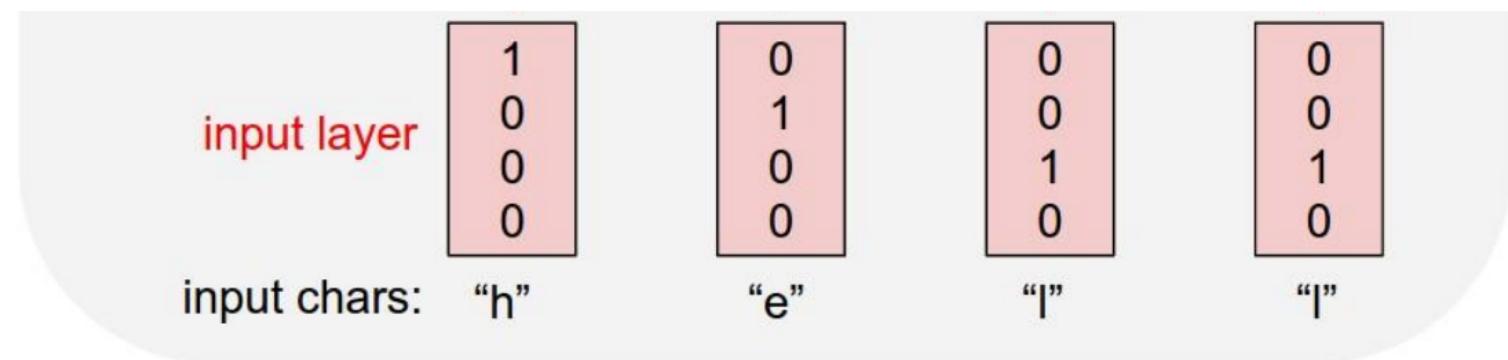
Note: hidden states are always carried forward without any time limit, but you'll just back-propagate loss for a finite number of steps



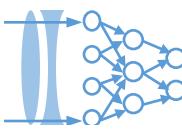
Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



From Stanford CS231n Lecture 10 slides

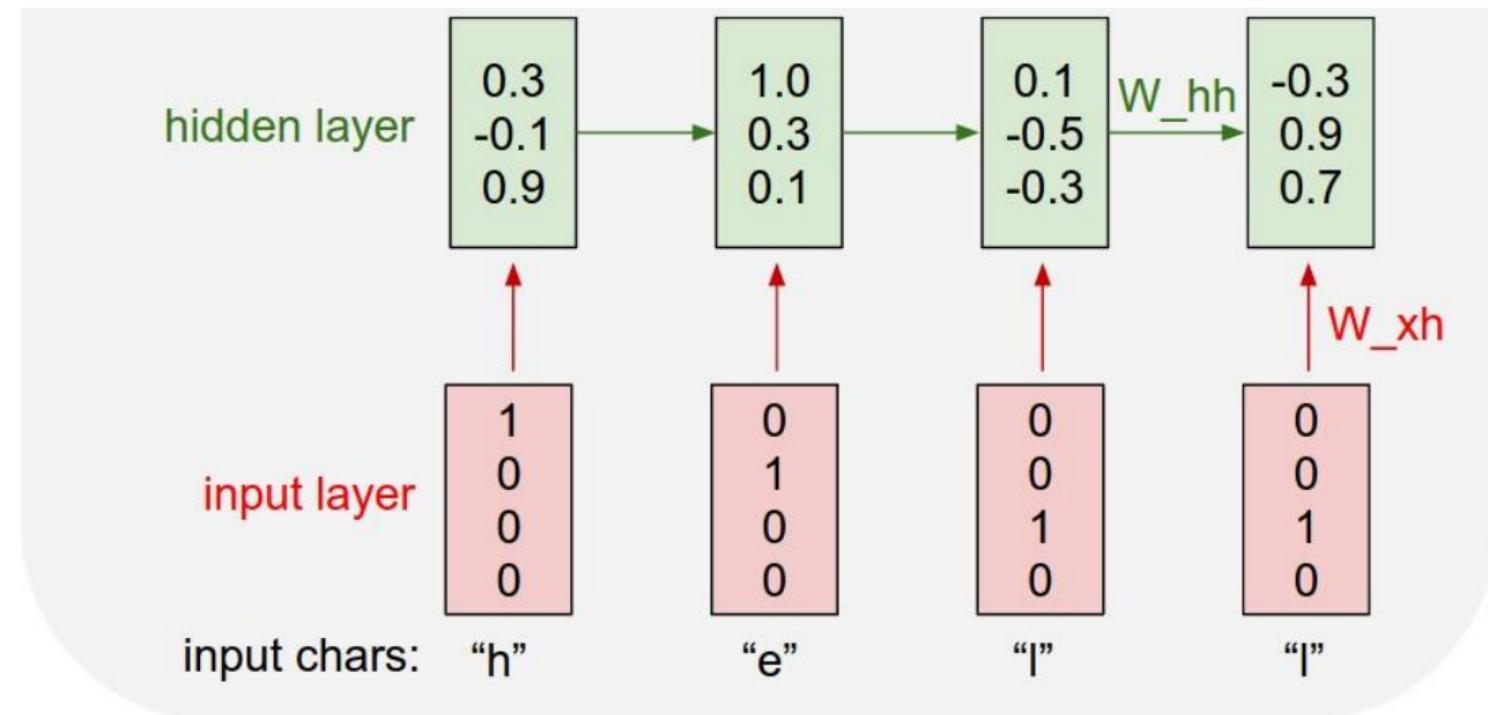


Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



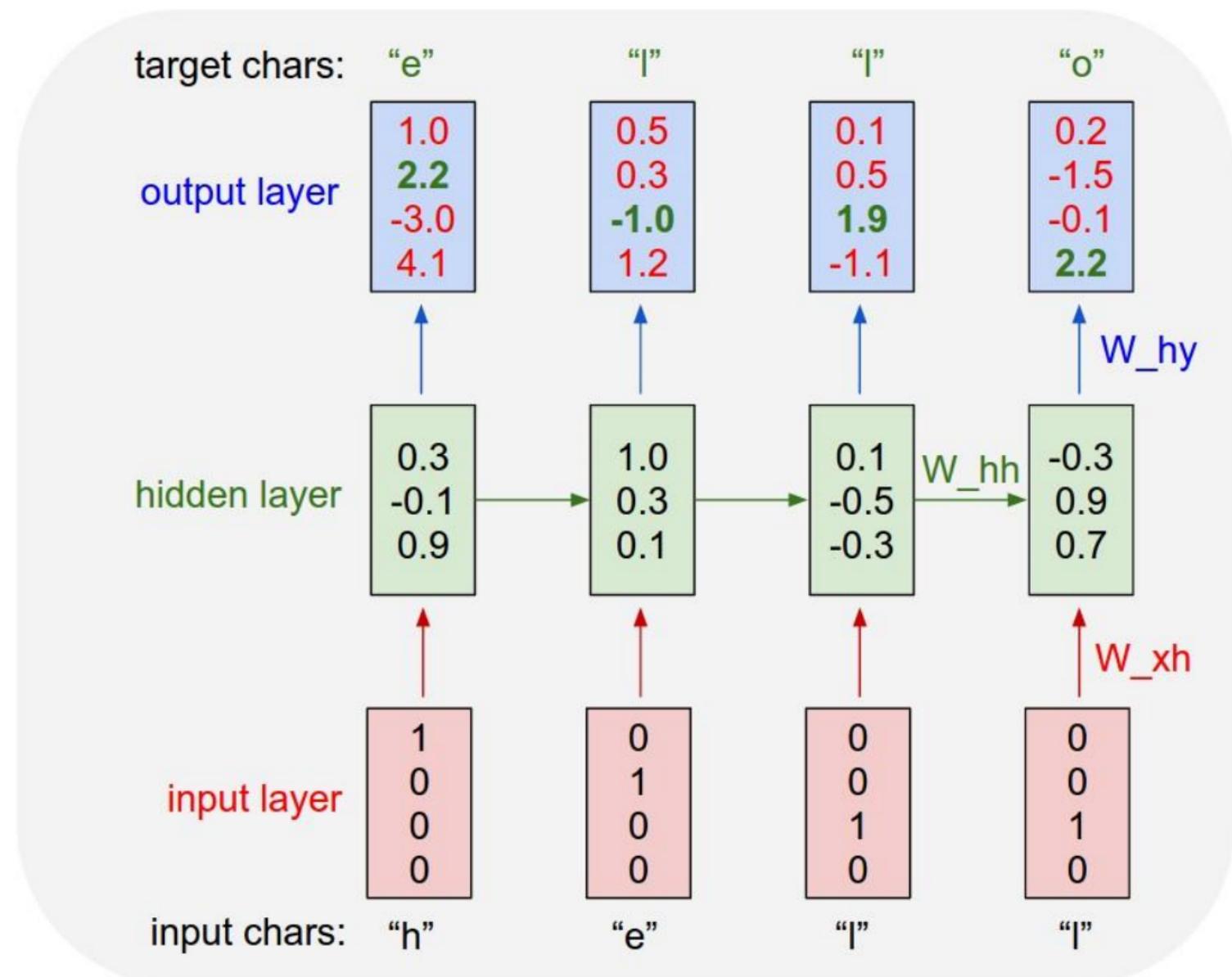
From Stanford CS231n Lecture 10 slides



Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



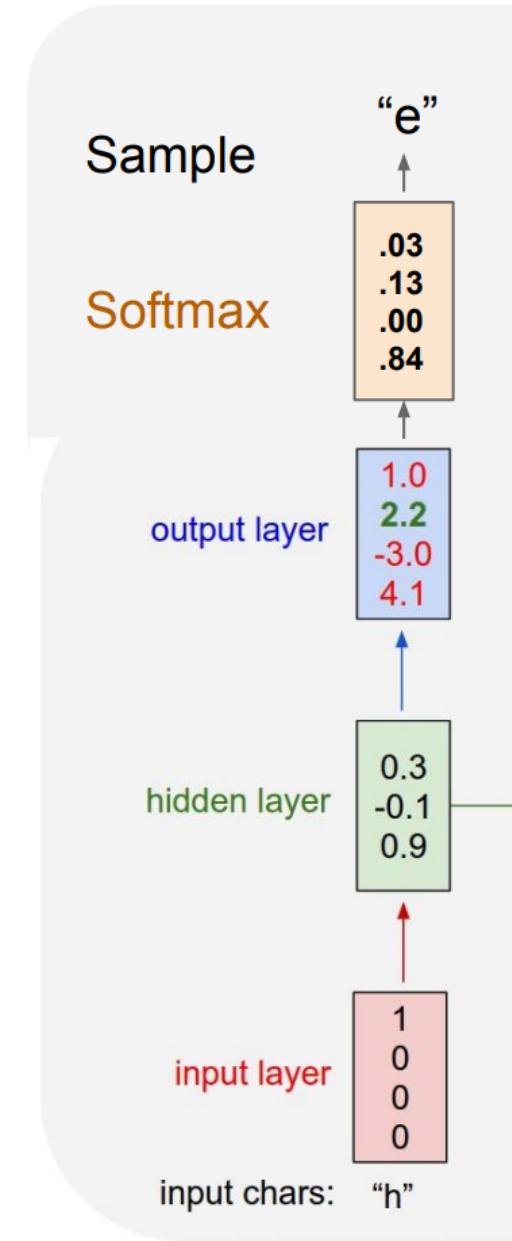
From Stanford CS231n Lecture 10 slides



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model



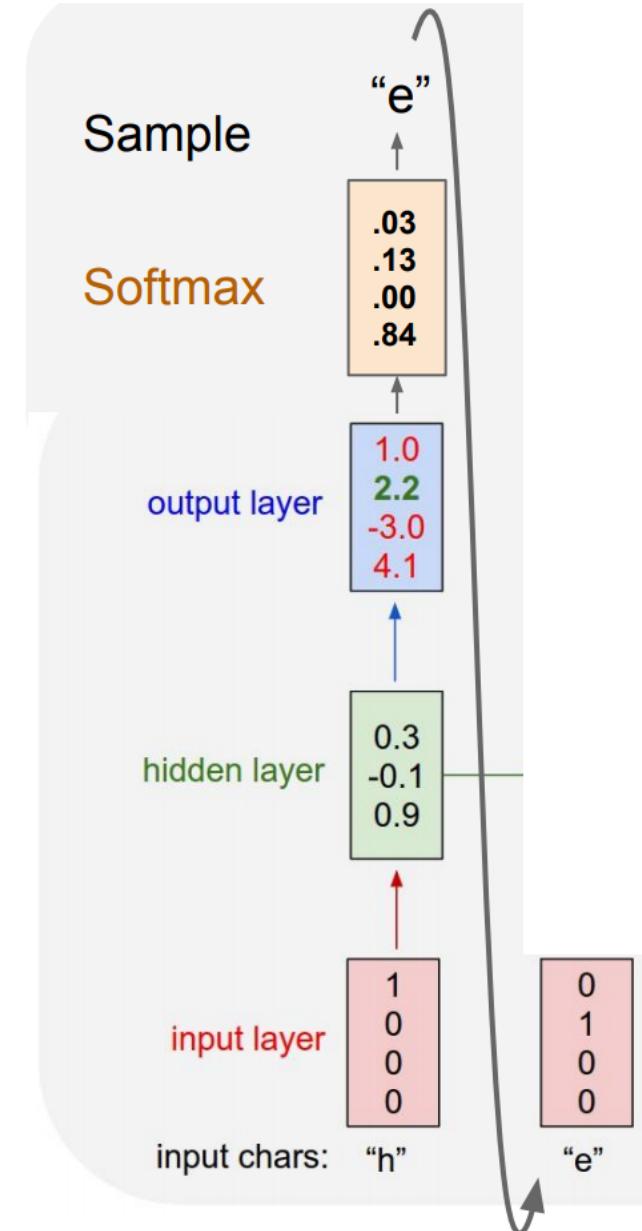
From Stanford CS231n Lecture 10 slides



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model



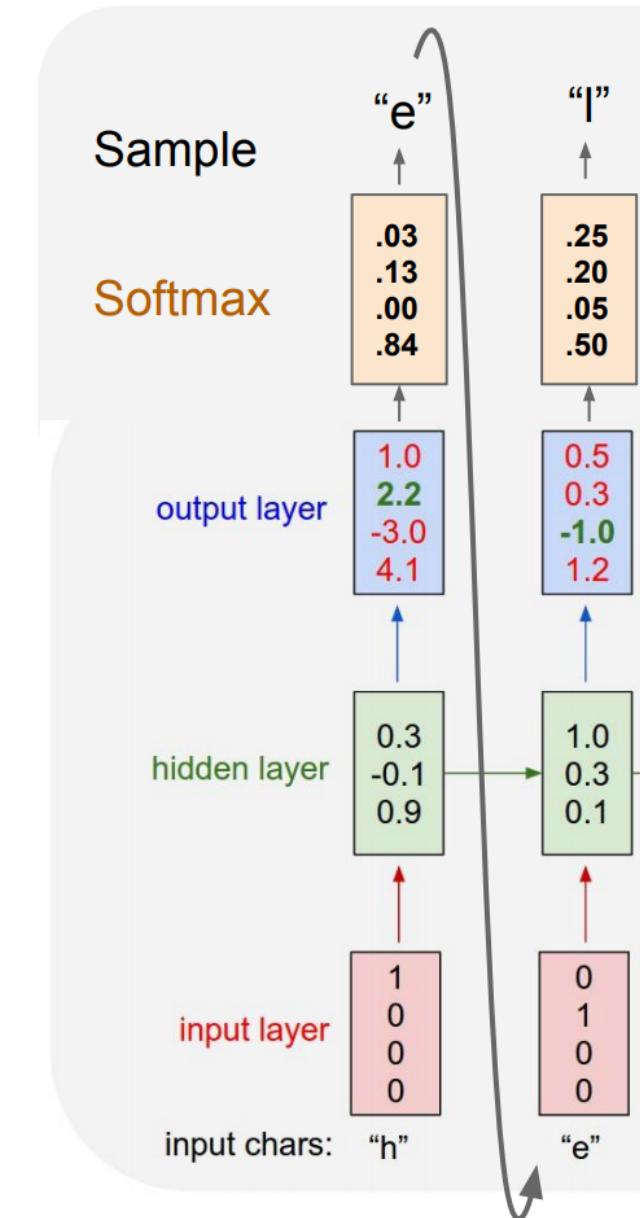
From Stanford CS231n Lecture 10 slides



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model

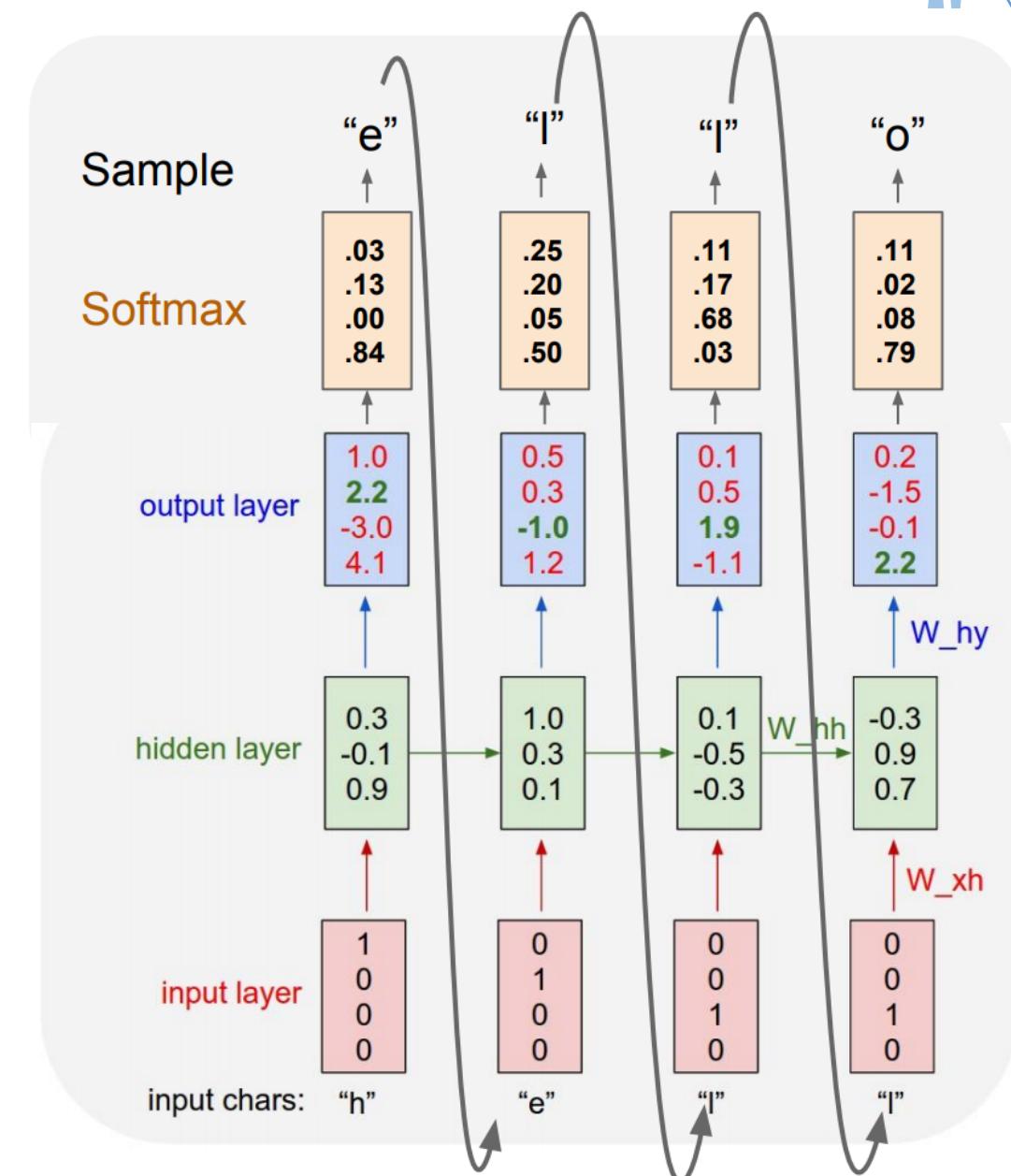


From Stanford CS231n Lecture 10 slides

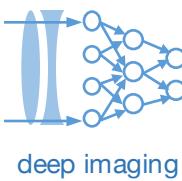
Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

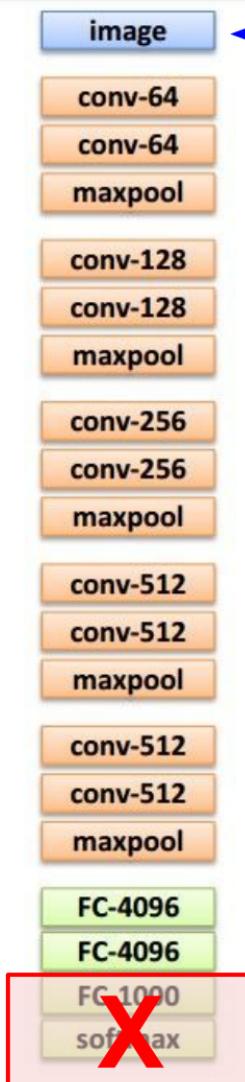
At test-time sample
characters one at a time,
feed back to model



From Stanford CS231n Lecture 10 slides

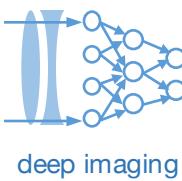


Example: Image captioning

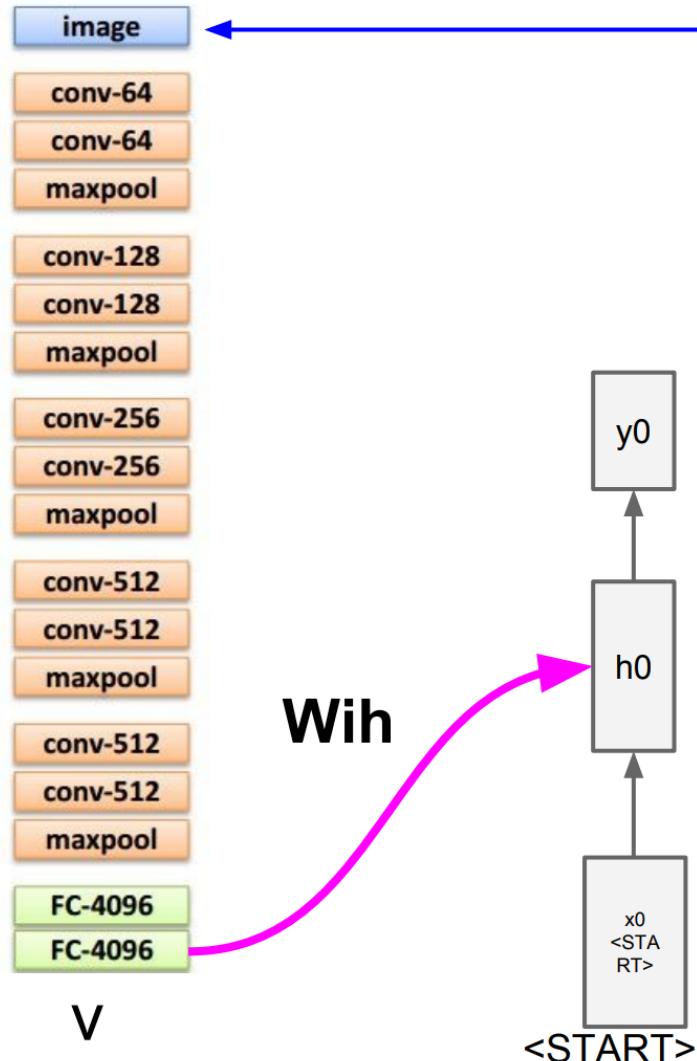


test image

From Stanford CS231n Lecture 10 slides



Example: Image captioning



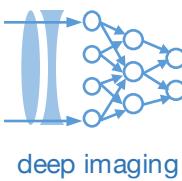
before:

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

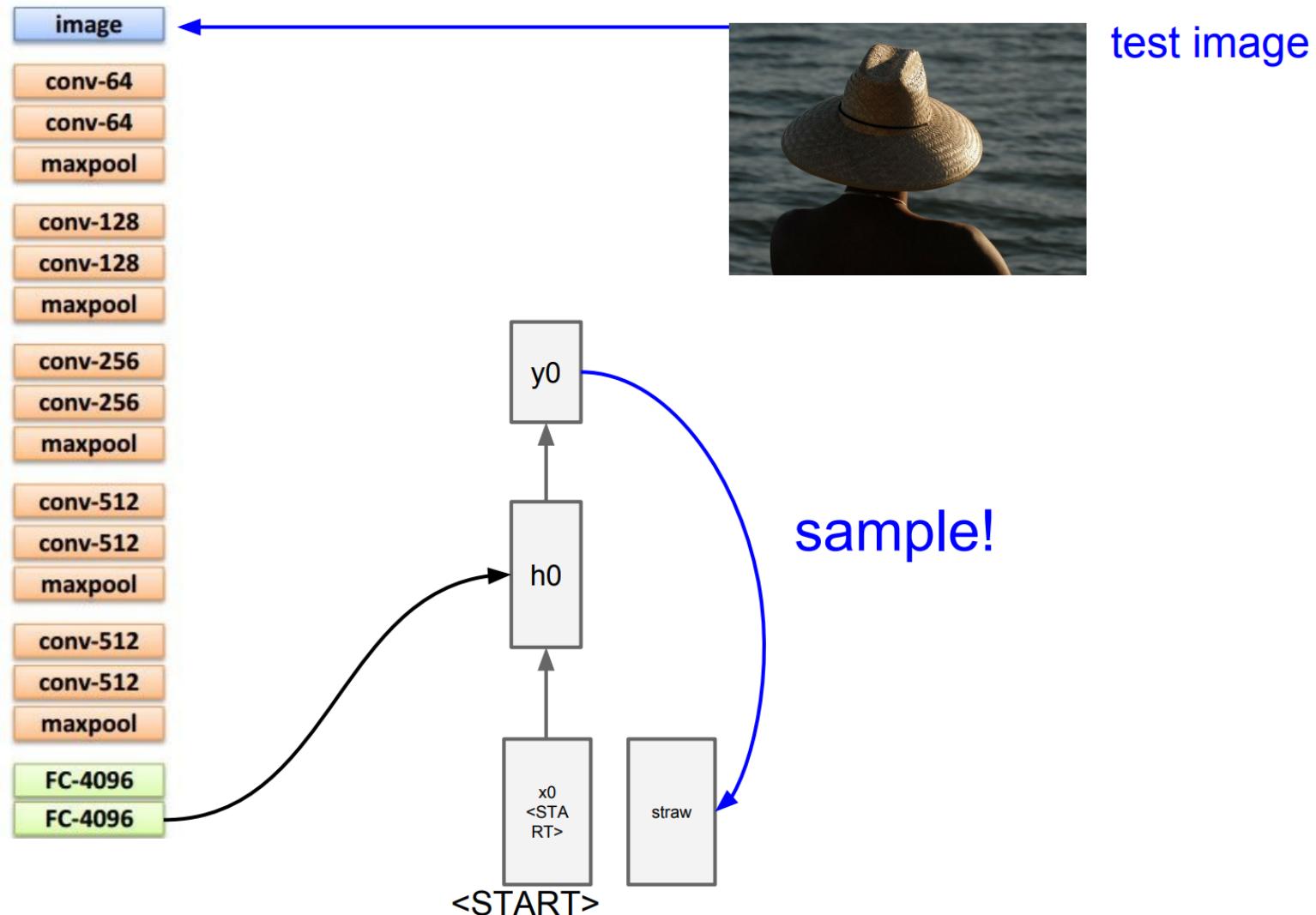
now:

$$h = \tanh(W_{xh} * x + W_{hh} * h + \textcolor{magenta}{W_{ih} * v})$$

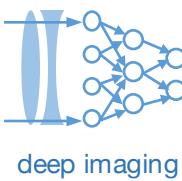
From Stanford CS231n Lecture 10 slides



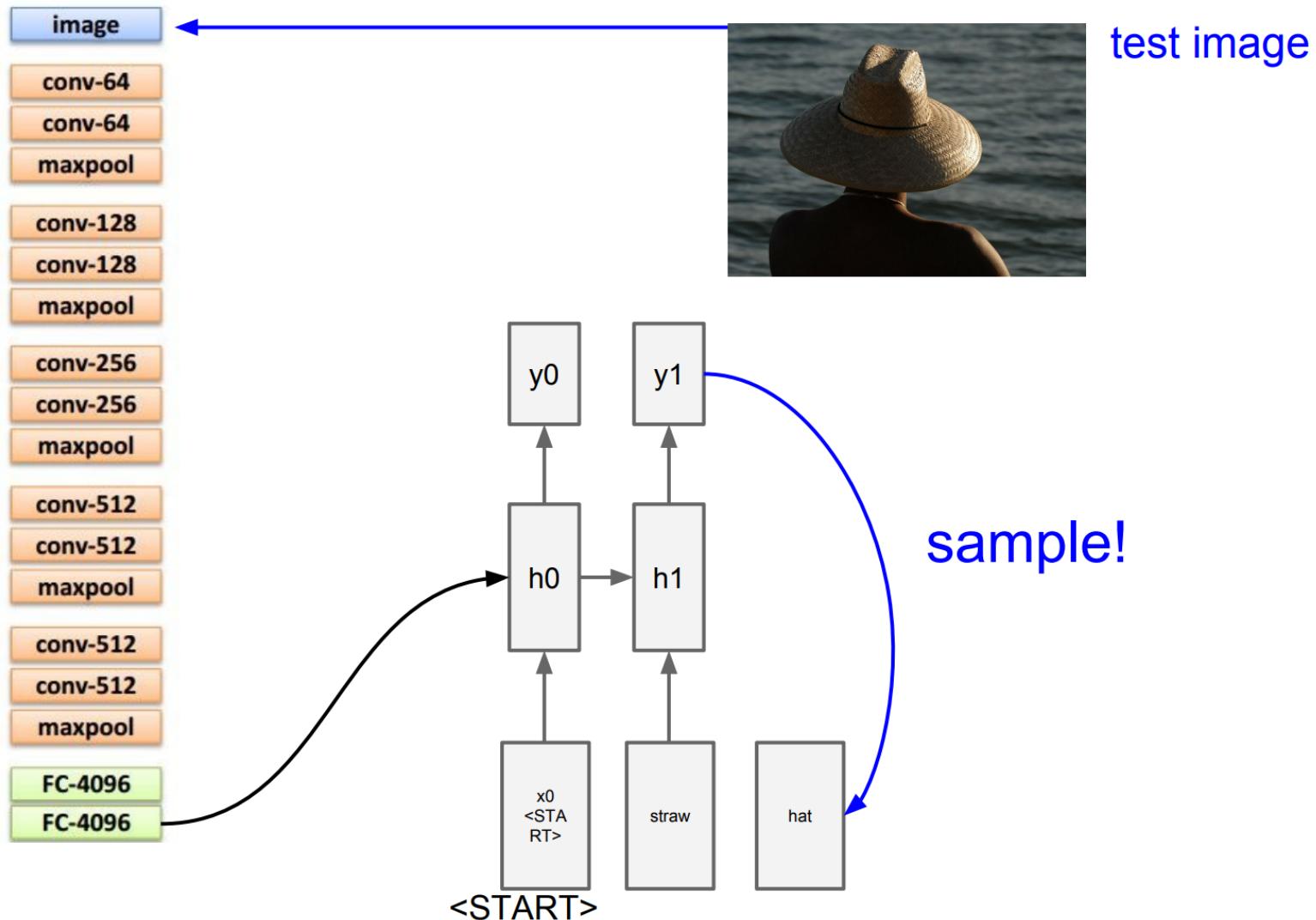
Example: Image captioning



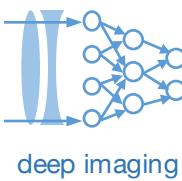
From Stanford CS231n Lecture 10 slides



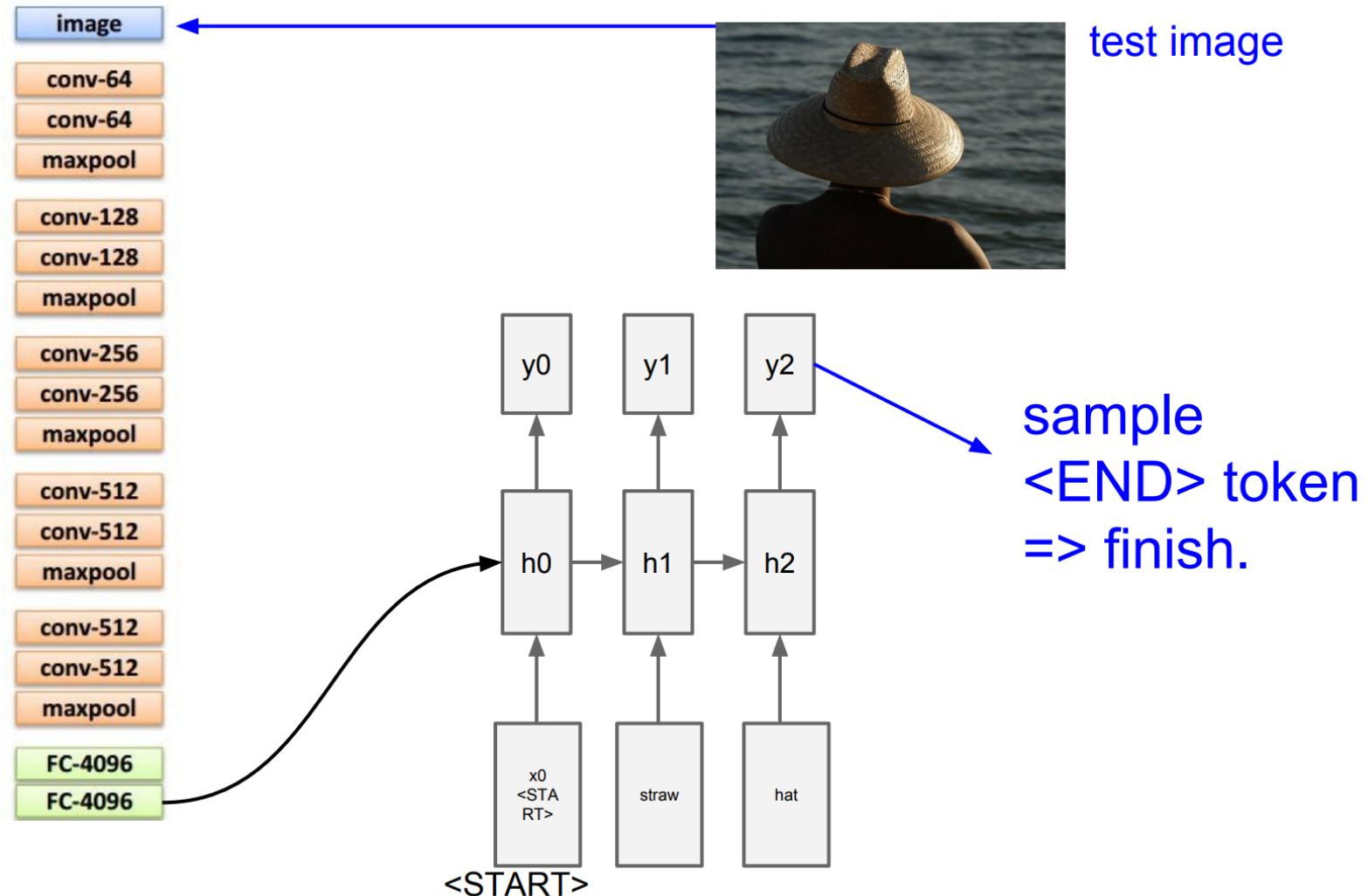
Example: Image captioning



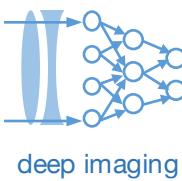
From Stanford CS231n Lecture 10 slides



Example: Image captioning



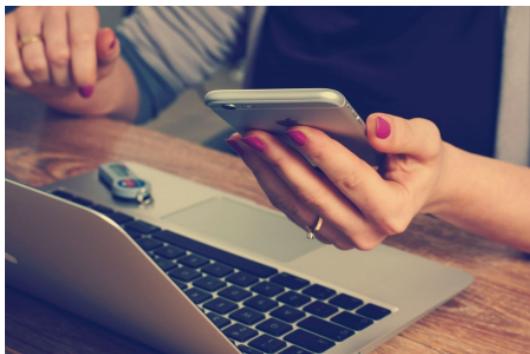
From Stanford CS231n Lecture 10 slides



Example: Image captioning



A woman is holding a cat in her hand



A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



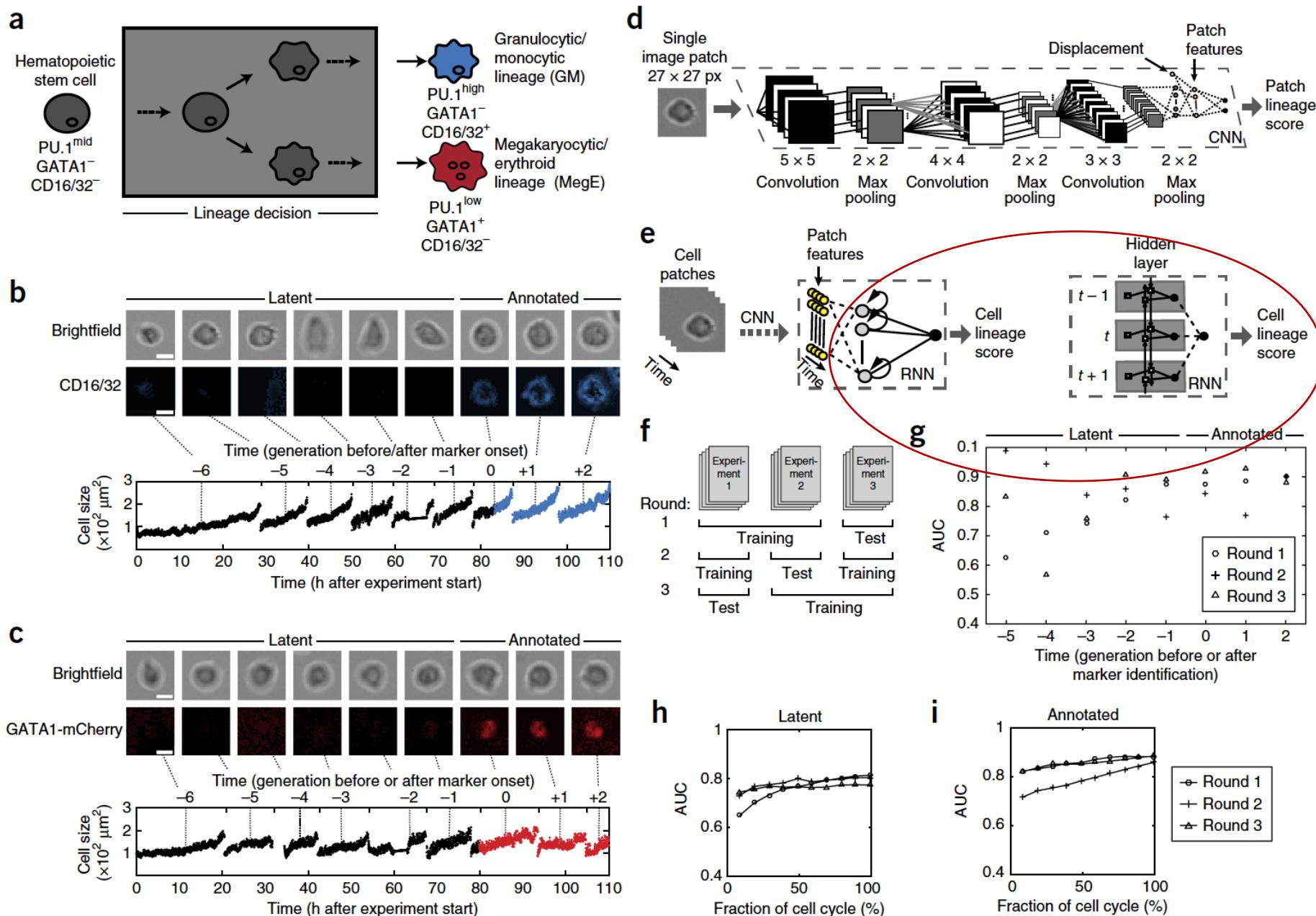
A man in a baseball uniform throwing a ball

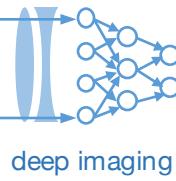
Captions generated using [neuraltalk2](#)
All images are [CC0 Public domain](#): [fur coat](#), [handstand](#), [spider web](#), [baseball](#)



Prospective identification of hematopoietic lineage choice by deep learning

Felix Buggenthin^{1,6}, Florian Buettner^{1,2,6}, Philipp S Hoppe^{3,4}, Max Endele³, Manuel Kroiss^{1,5}, Michael Strasser¹, Michael Schwarzfischer¹, Dirk Loeffler^{3,4}, Konstantinos D Kokkaliaris^{3,4}, Oliver Hilsenbeck^{3,4}, Timm Schroeder^{3,4}, Fabian J Theis^{1,5} & Carsten Marr¹





What are we possibly missing from the many-to-many model?

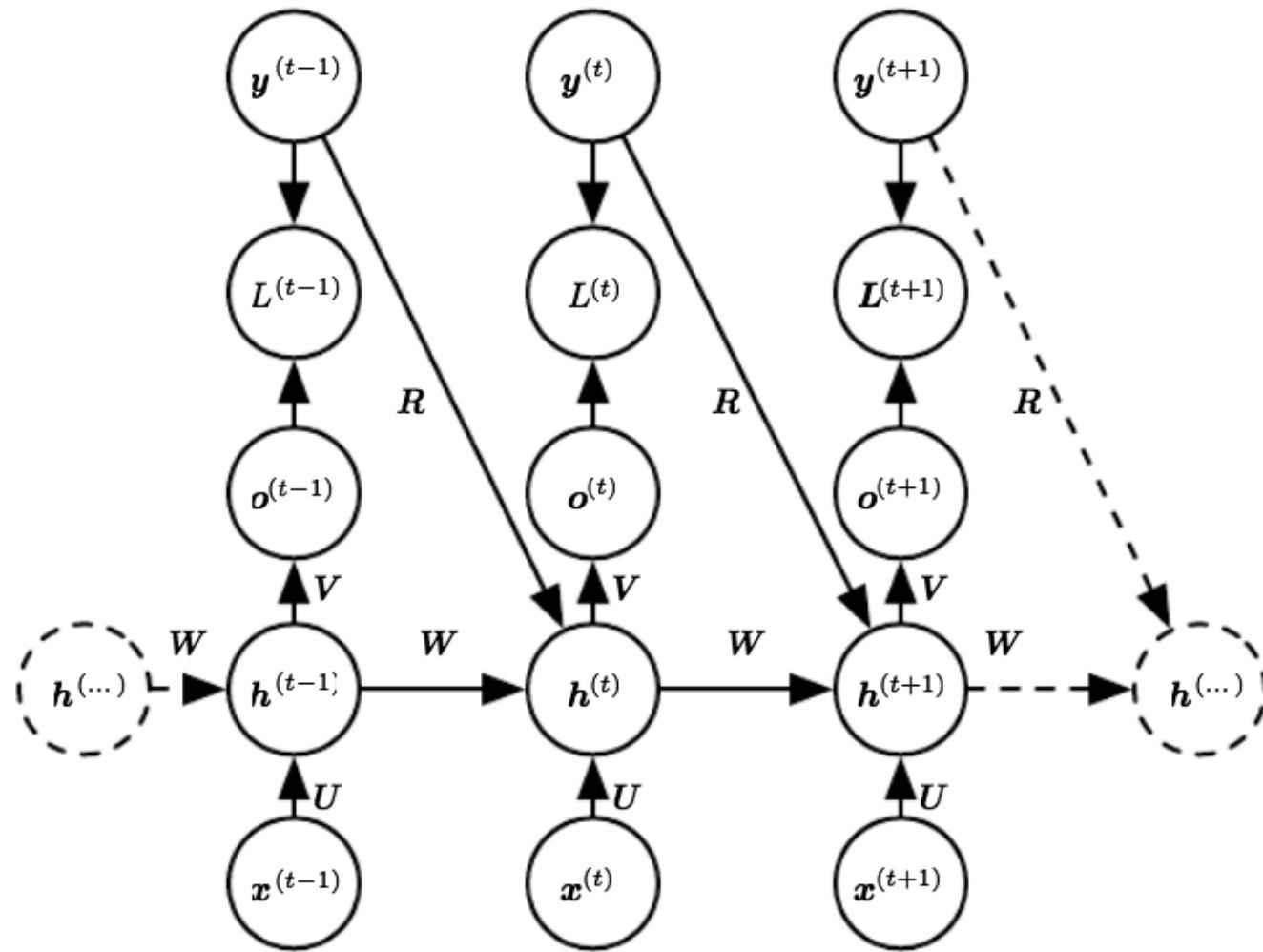
1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

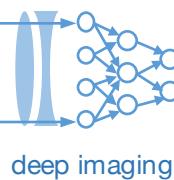
$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)})$$

Let the network become dependent on past labels as well:

$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(t-1)})$$

Conditional recurrent neural network





What are we possibly missing from the many-to-many model?

1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

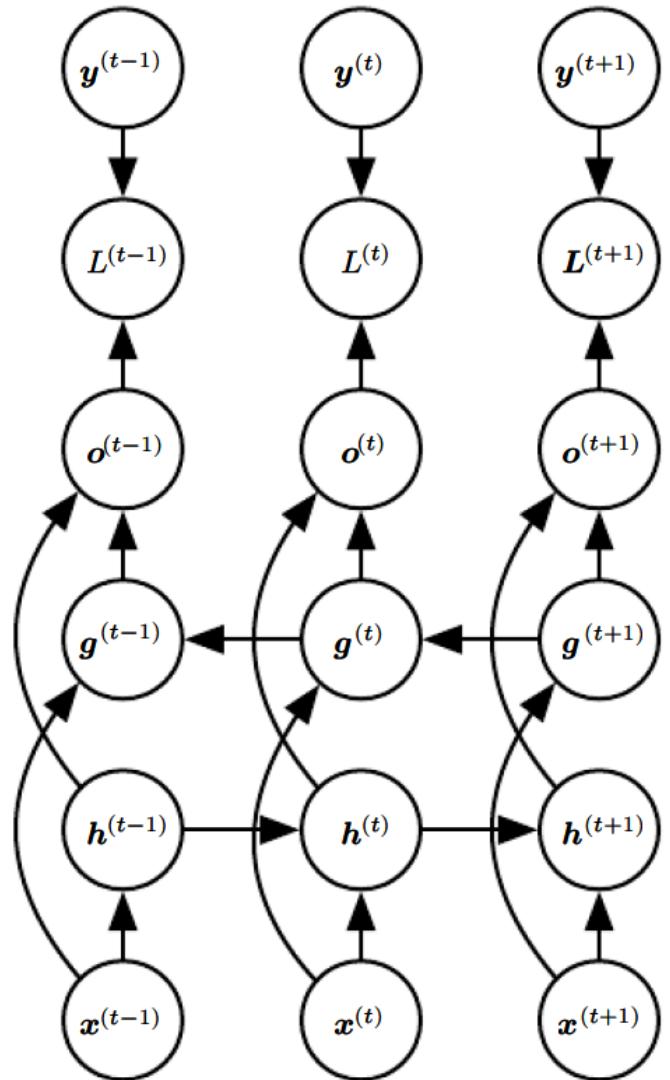
$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)})$$

Let the network become dependent on past labels as well:

$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(t-1)})$$

2. Only considering one direction in sequence/time...

Other extensions: bi-directional analysis



- Consider future and past events jointly
- Add a third matrix that takes future hidden states in as well
- E.g., sentence structure is not purely causal
- Handwriting recognition, speech analysis, etc.

What are we possibly missing from the many-to-many model?

1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

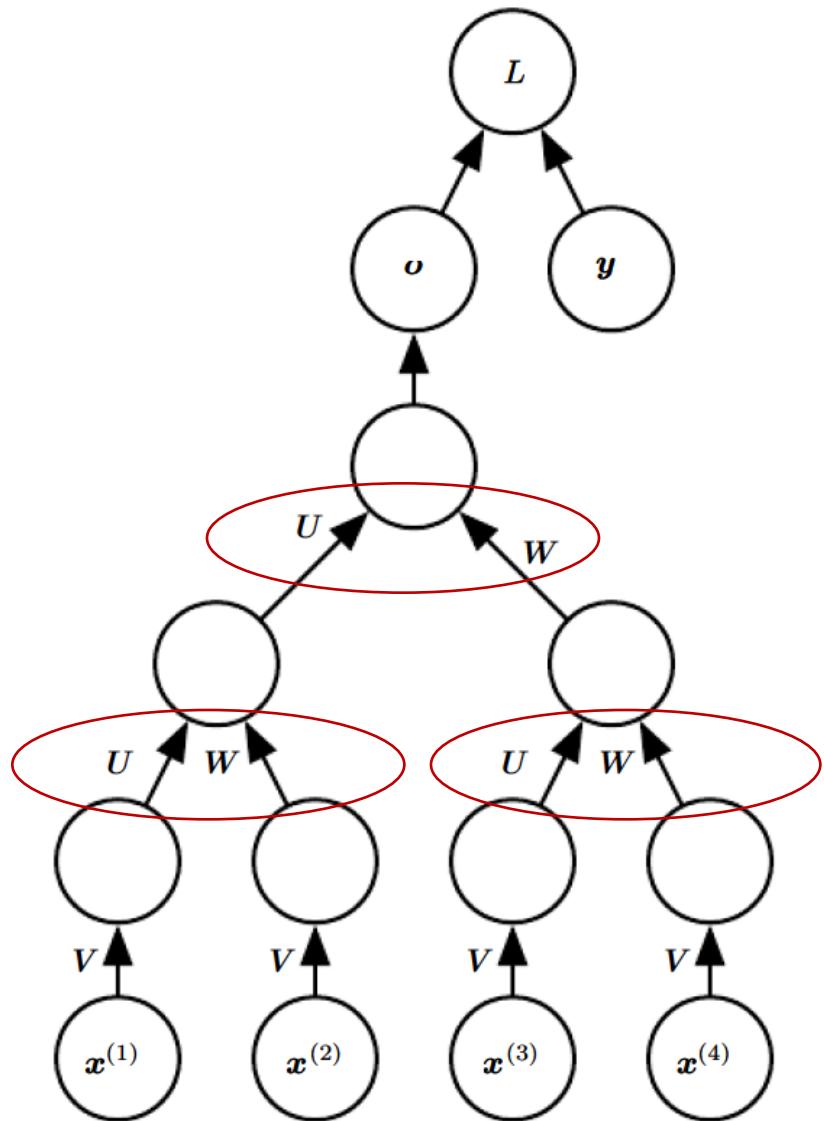
$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)})$$

Let the network become dependent on past labels as well:

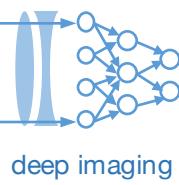
$$\log p(\mathbf{y}^{(t)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(t-1)})$$

2. Only considering one direction in sequence/time...
3. Chaining things together is not necessarily the ideal way to maintain long connections

Other extensions: recursive neural networks



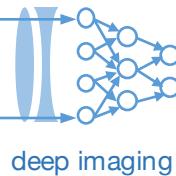
- Use tree-like structure instead of chain-like structure to embed temporal relationships
- Reduce n nonlinear relationships connecting time a to time b to $n \log n$
- Obviously lots of extensions/variants here



RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\mathbf{h}^{(t)} = \mathbf{W}^\top \mathbf{h}^{(t-1)} \quad \longrightarrow \quad \mathbf{h}^{(t)} = (\mathbf{W}^t)^\top \mathbf{h}^{(0)},$$



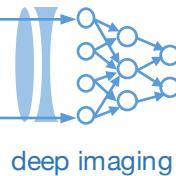
RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\mathbf{h}^{(t)} = \mathbf{W}^\top \mathbf{h}^{(t-1)} \quad \longrightarrow \quad \mathbf{h}^{(t)} = (\mathbf{W}^t)^\top \mathbf{h}^{(0)},$$

If \mathbf{W} admits it, can perform eigenvector decomposition to obtain,

$$\mathbf{W} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top,$$



RNN's have limited memory and can suffer from exploding gradients

Hidden weights effectively follow a recursive relationship:

$$\mathbf{h}^{(t)} = \mathbf{W}^\top \mathbf{h}^{(t-1)} \quad \longrightarrow \quad \mathbf{h}^{(t)} = (\mathbf{W}^t)^\top \mathbf{h}^{(0)},$$

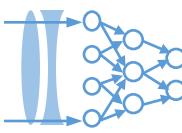
If \mathbf{W} admits it, can perform eigenvector decomposition to obtain,

$$\mathbf{W} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top,$$

In this space, power relationship \mathbf{W}^t alters just eigenvalues, does not rotate eigenvectors:

$$\mathbf{h}^{(t)} = \mathbf{Q}^\top \boldsymbol{\Lambda}^t \mathbf{Q} \mathbf{h}^{(0)}$$

Thus, if the eigenvector is large (the largest), it will explode. Remaining eigenvectors eventually vanish



The long short-term memory network

Additions:

- Self-loop to maintain “memory”
 - Allow gradients to flow for a long time
- Weight of self-loop gated by “Forget gate”
 - Forgetting depends on data
 - Memory time scale is thus dynamic
- Output gate
 - Can turn on/shut off everything

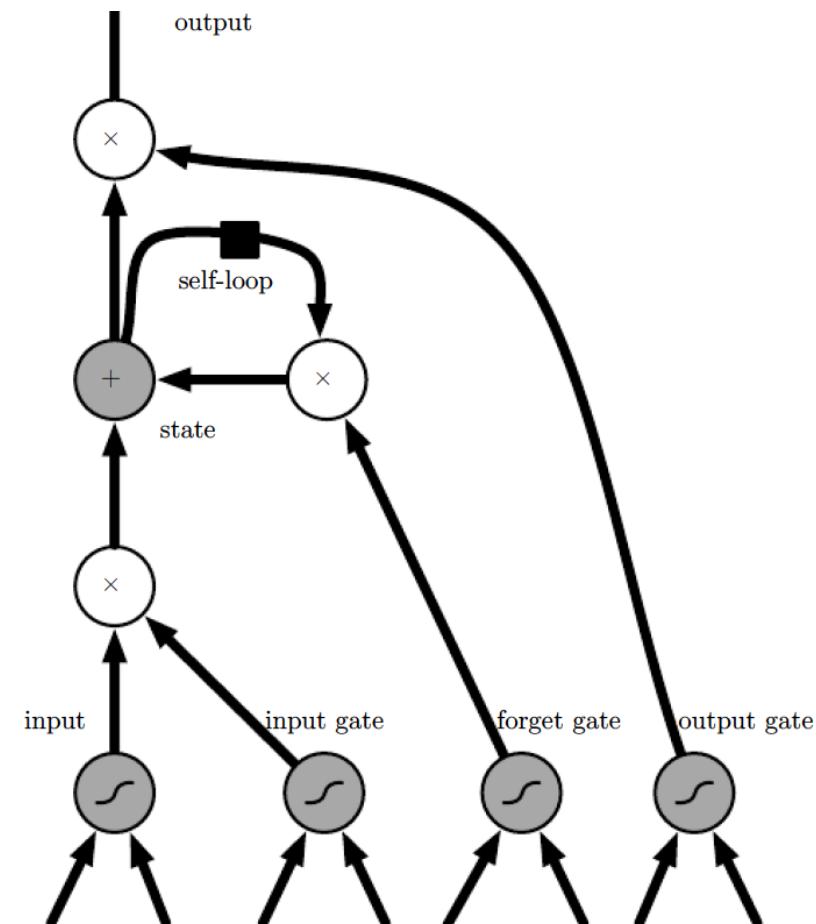
S. Hochreiter and J. Schmidhuber (1997)

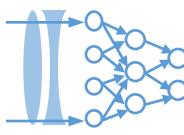
Long Short-Term Memory

<https://dl.acm.org> › citation

by S Hochreiter - 1997 - Cited by 24398 - Related articles

Nov 1, 1997 - Hochreiter, S., & Schmidhuber, J. (1996). Bridging long time lags by weight guessing and "long short-term memory." In F. L. Silva, J. C. Principe, ...





The long short-term memory network

Forget gate:

$$f_i^{(t)} = \sigma \left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right)$$

Internal state:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right)$$

External input gate:

$$g_i^{(t)} = \sigma \left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right)$$

LSTM output:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)}$$

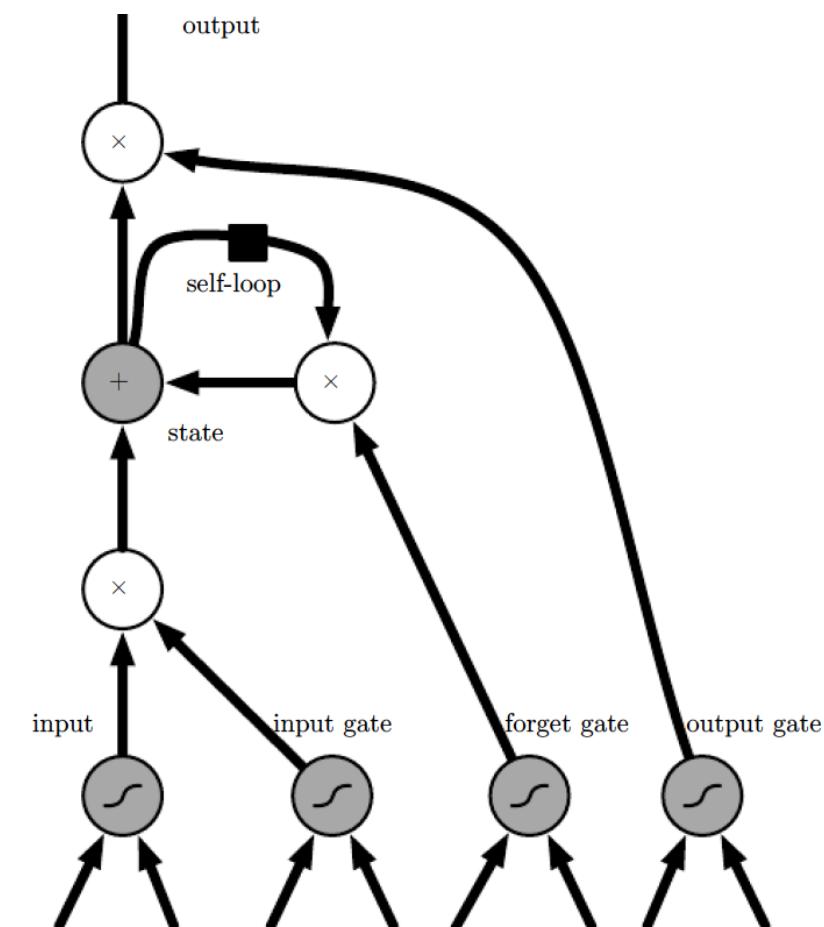
S. Hochreiter and J. Schmidhuber (1997)

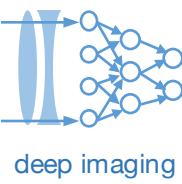
Long Short-Term Memory

<https://dl.acm.org> › citation

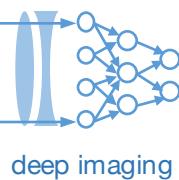
by S Hochreiter - 1997 - Cited by 24398 - Related articles

Nov 1, 1997 - Hochreiter, S., & Schmidhuber, J. (1996). Bridging long time lags by weight guessing and "long short-term memory." In F. L. Silva, J. C. Principe, ...





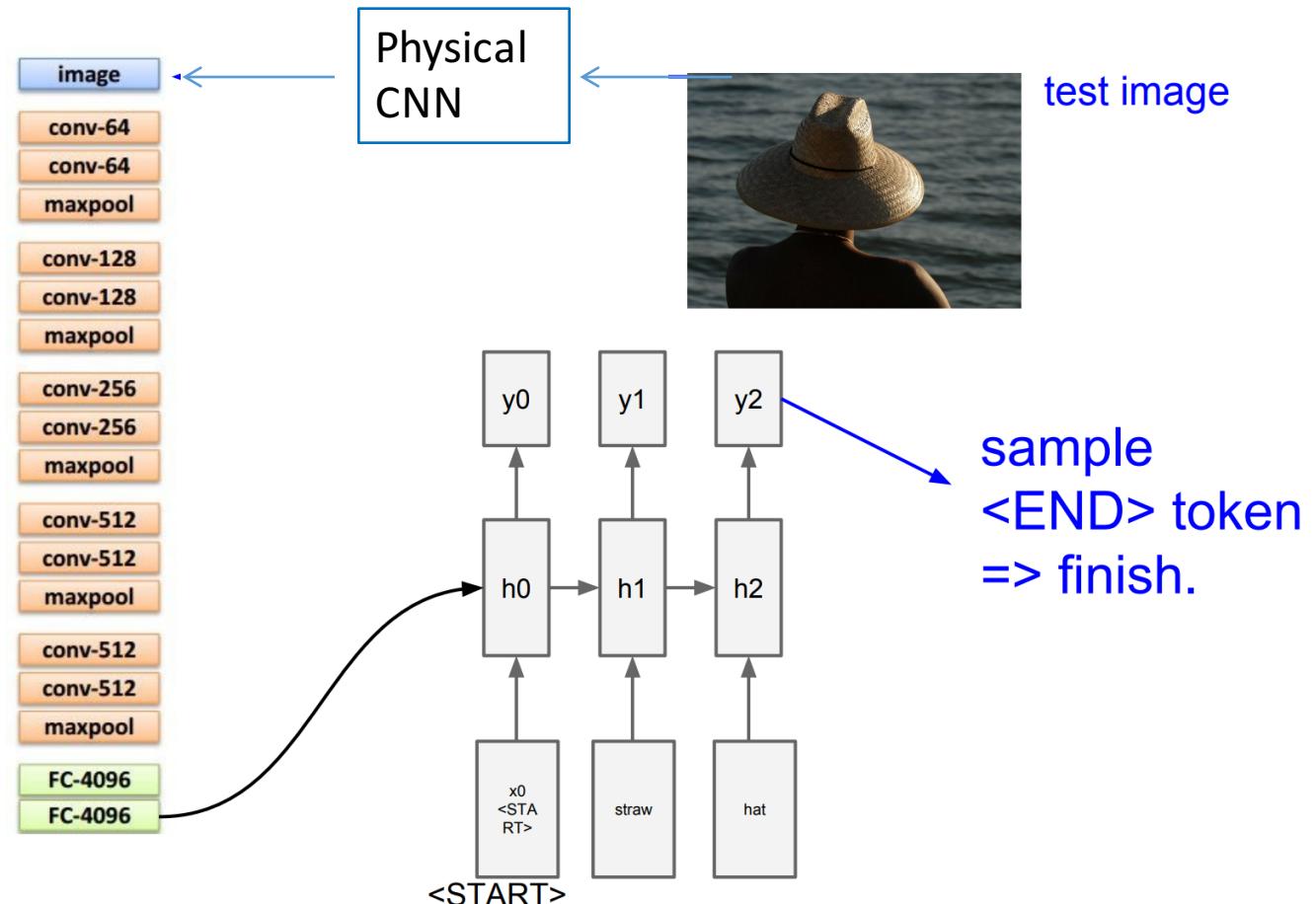
Brainstorming time – physical layers in an RNN???



Brainstorming time – physical layers in an RNN???

Here's a simple example -

Design an optimal X to produce the best image captions



Brainstorming time – physical layers in an RNN???

Take a bit of time and try to write down the following:

- With your image data (or some data that you are interested), what might you input into an RNN?
- What might be a useful output?
- What physical parameter might be useful to tweak to improve this output?
- Can you think of a way to model that parameter?