deep imaging

Lecture 22: Recurrent Neural Networks

Machine Learning and Imaging

BME 548L
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

Values [0, 255]

I(x,y)

many image pixels to one output

Classify image [0 – 9]

Convolutional neural networks versus recurrent neural networks

Values [0, 255]

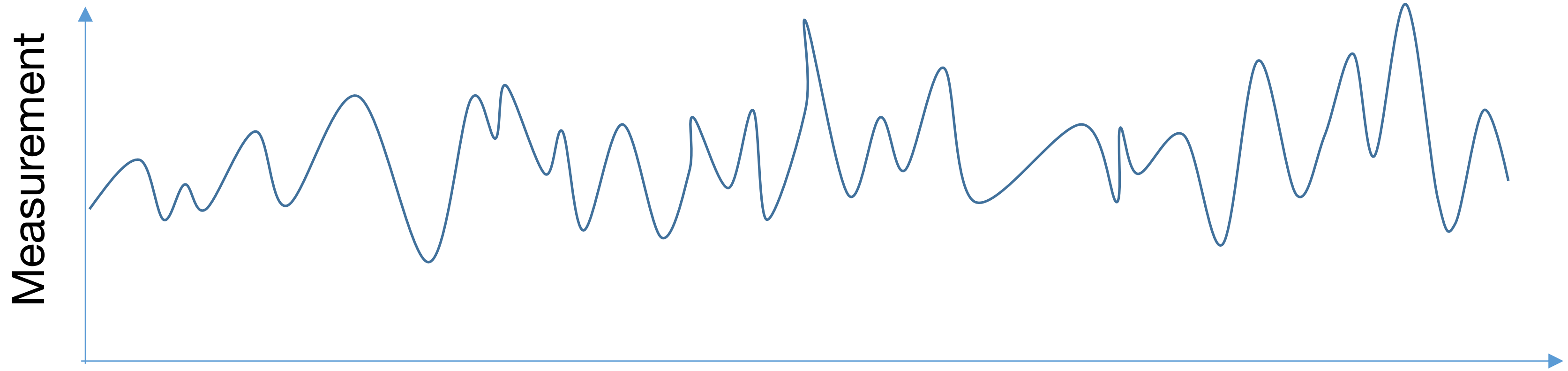
many image pixels to one output

I(x,y)

Classify image [0 – 9]

RNN's: Examine signals as a function of time

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

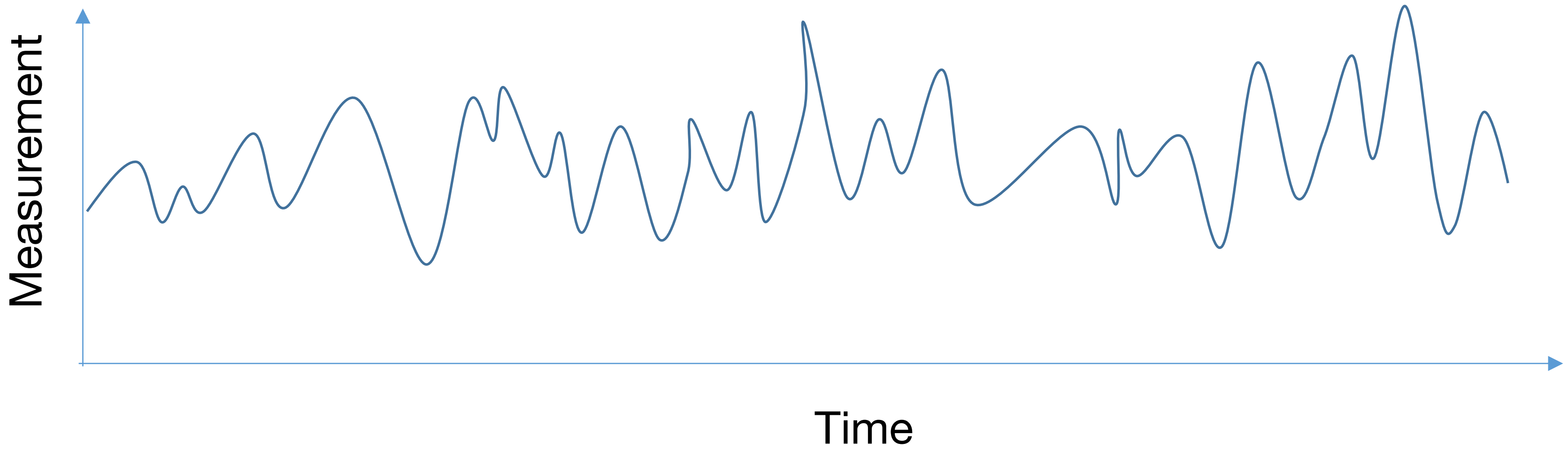


Convolutional neural networks versus recurrent neural networks

deep imaging

RNN's: Examine signals as a function of time

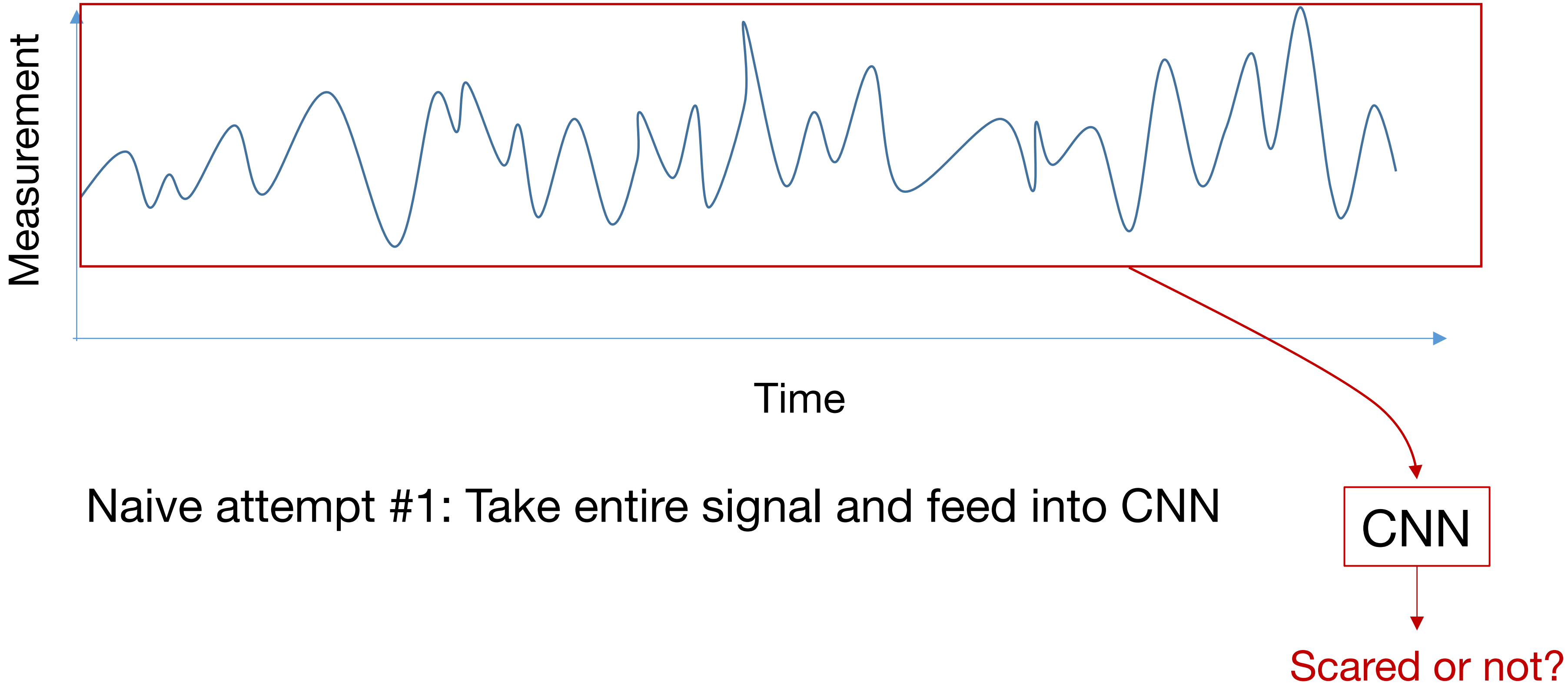
E.g., establish if mouse was scared from this EEG recording



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



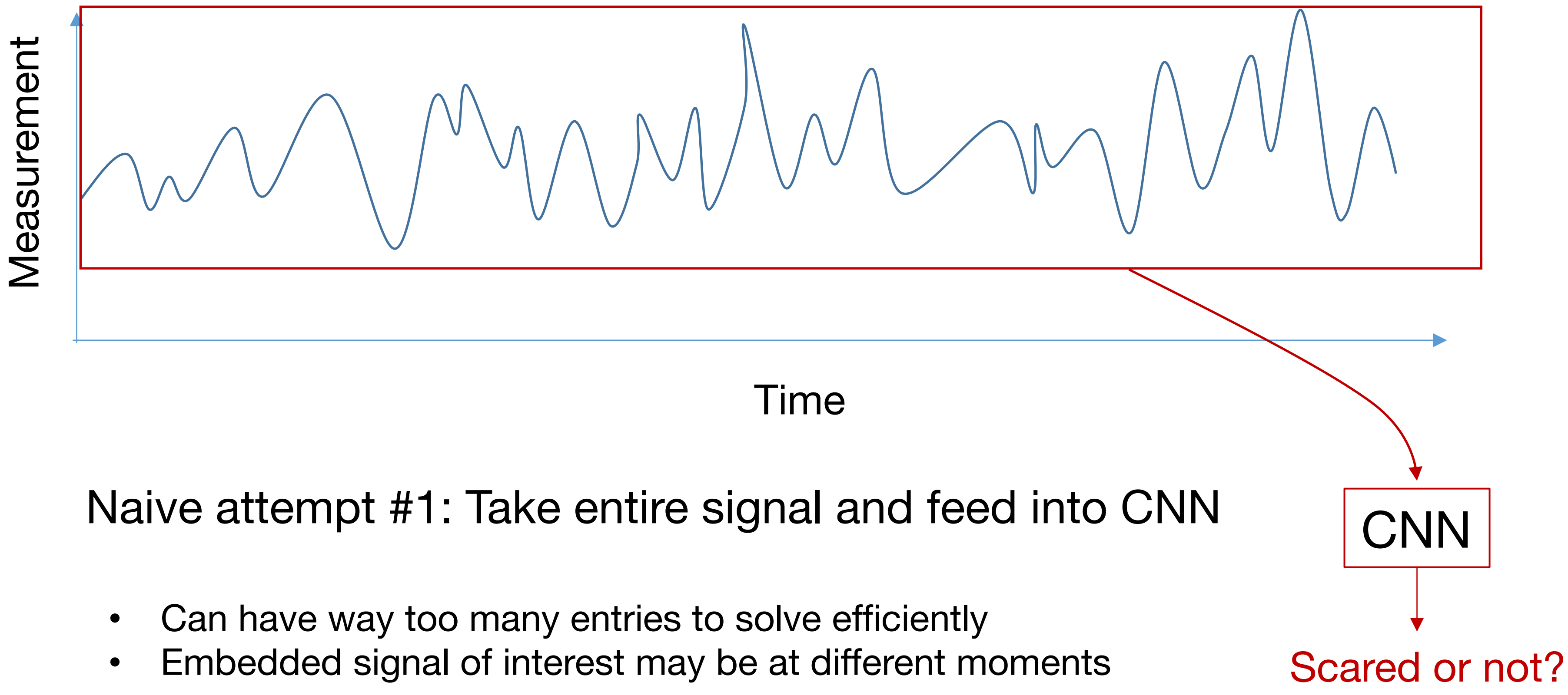
Naive attempt #1: Take entire signal and feed into CNN

Convolutional neural networks versus recurrent neural networks

deep imaging

RNN's: Examine signals as a function of time

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



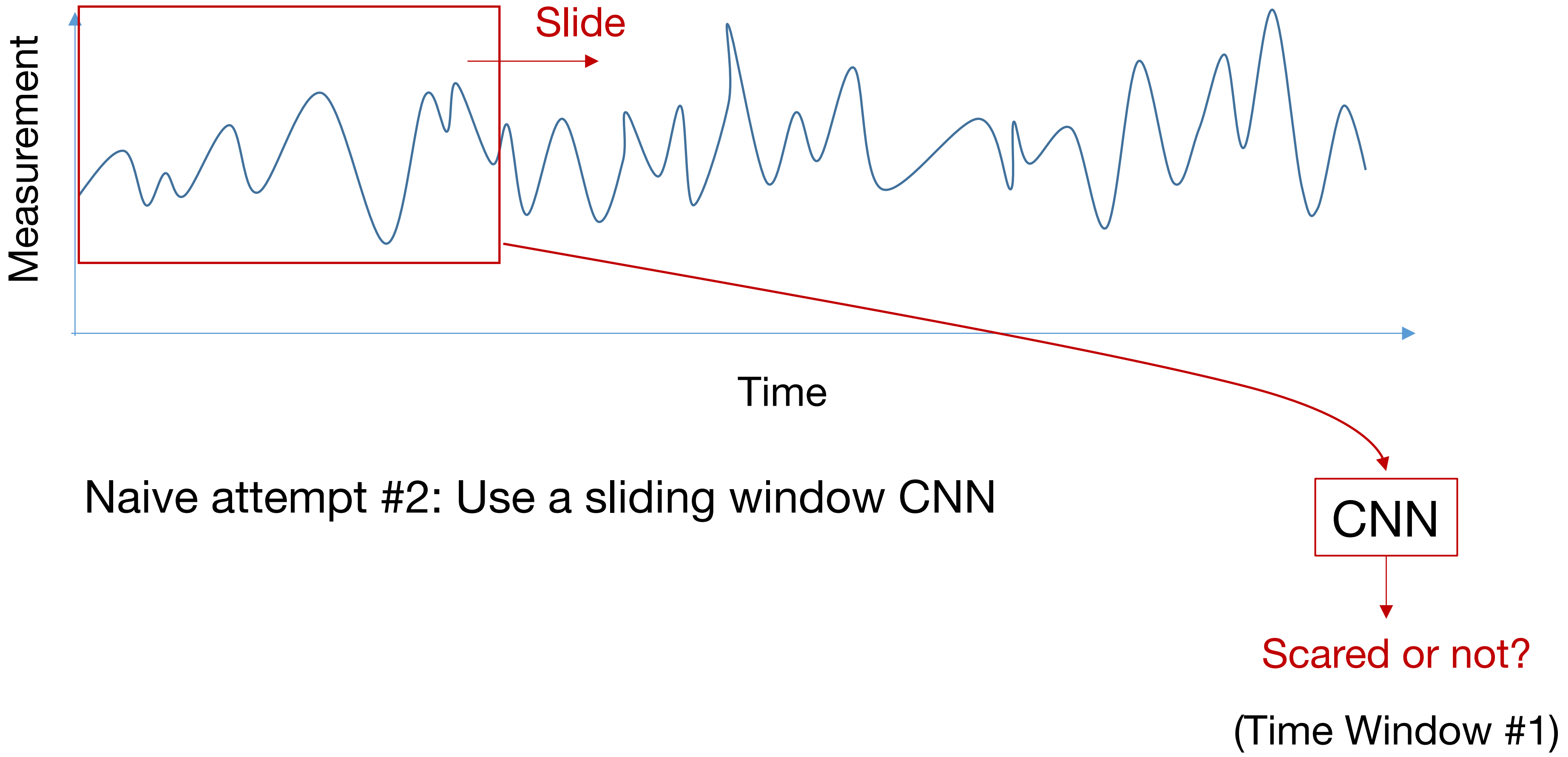
Naive attempt #1: Take entire signal and feed into CNN

- 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

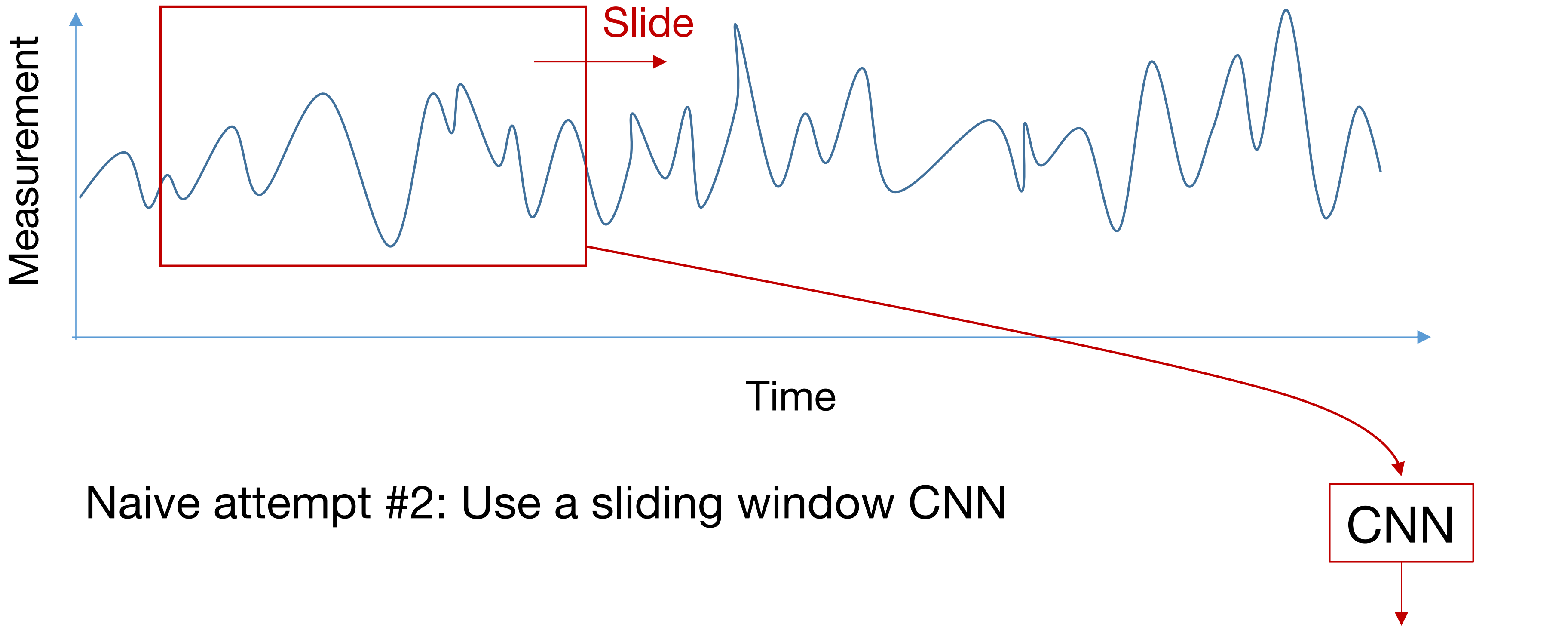


Naive attempt #2: Use a sliding window CNN

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

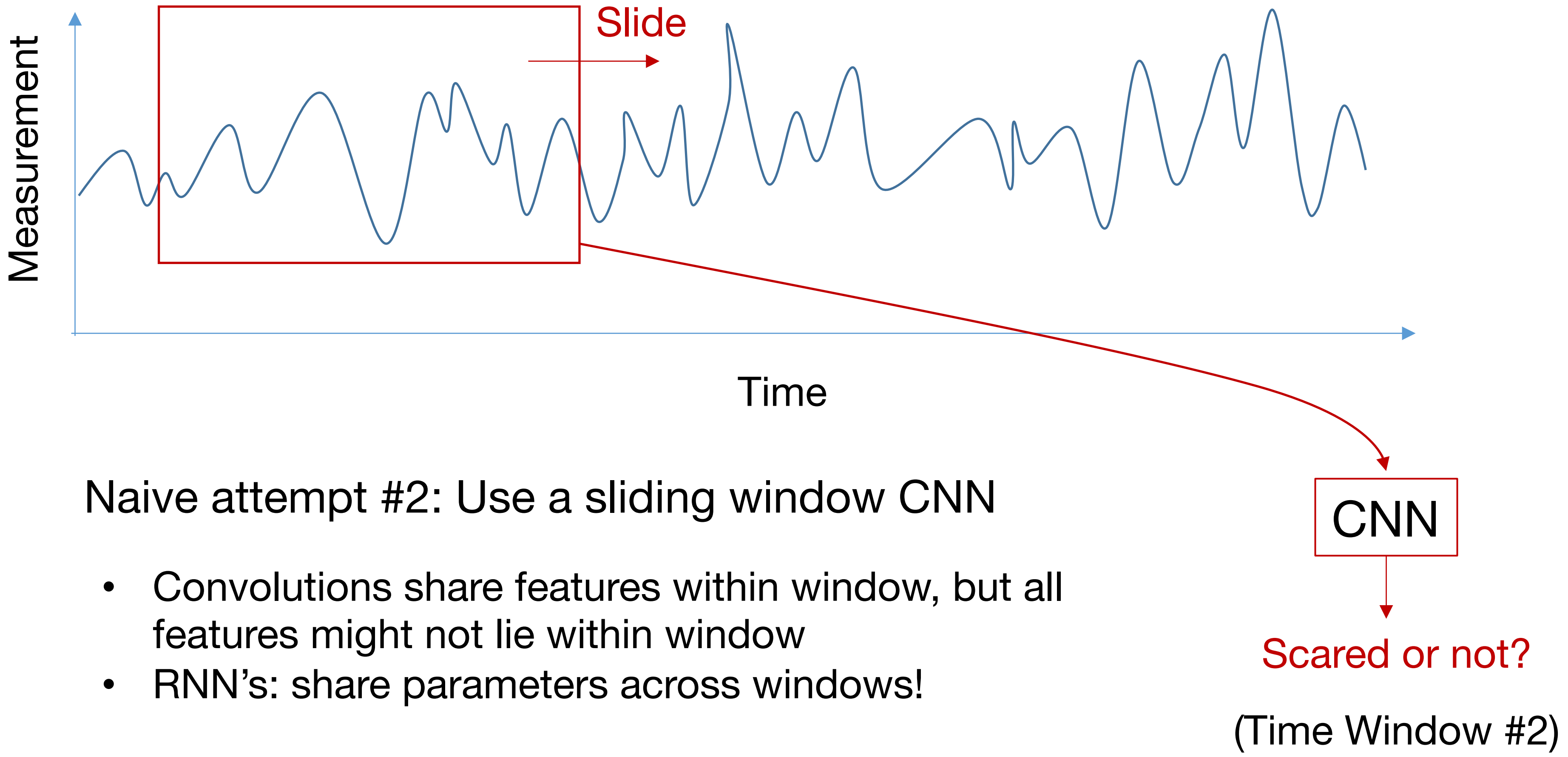
(Time Window #2)

Convolutional neural networks versus recurrent neural networks

deep imaging

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

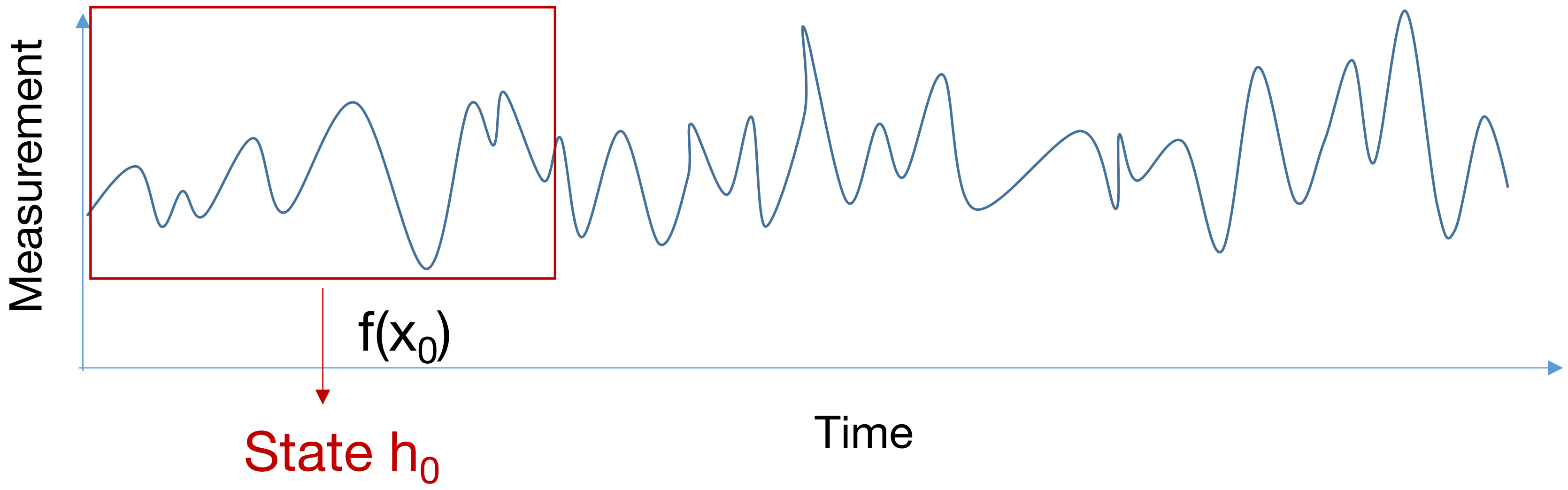
- Convolutions share features within window, but all features might not lie within window
- RNN's: share parameters across windows!

Recurrent neural networks in a nutshell

deep imaging

RNN’s: Examine signals as a function of time

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



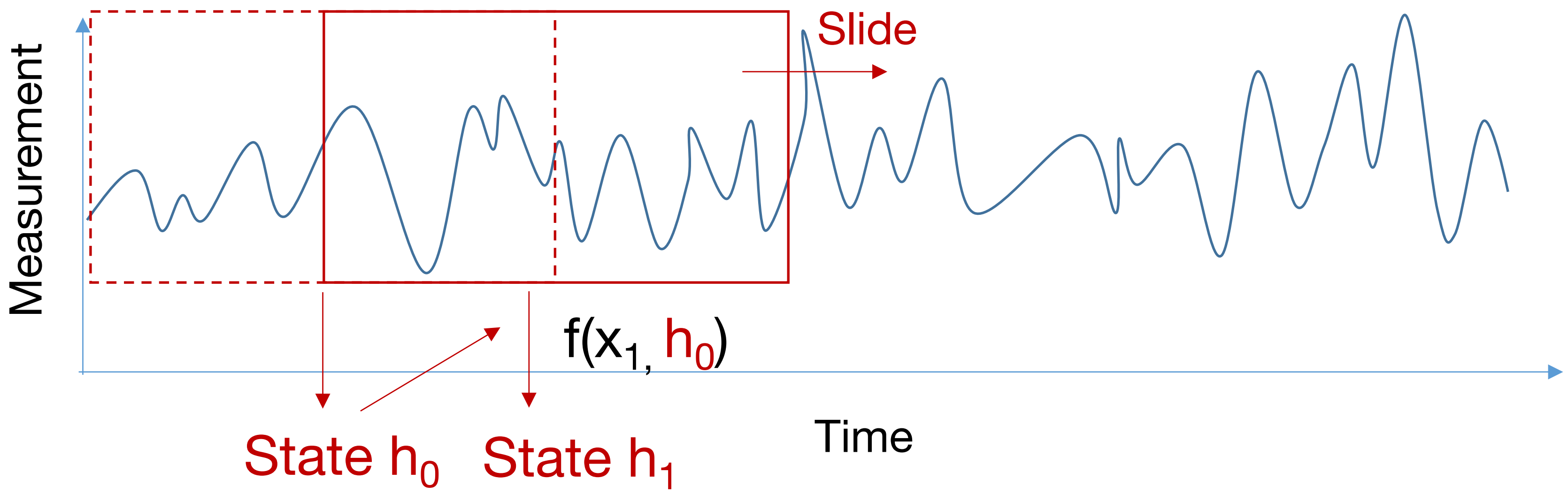
Recurrent neural networks: Generate states (“hidden units”) to use to inform subsequent decisions

Recurrent neural networks in a nutshell

deep imaging

RNN's: Examine signals as a function of time

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

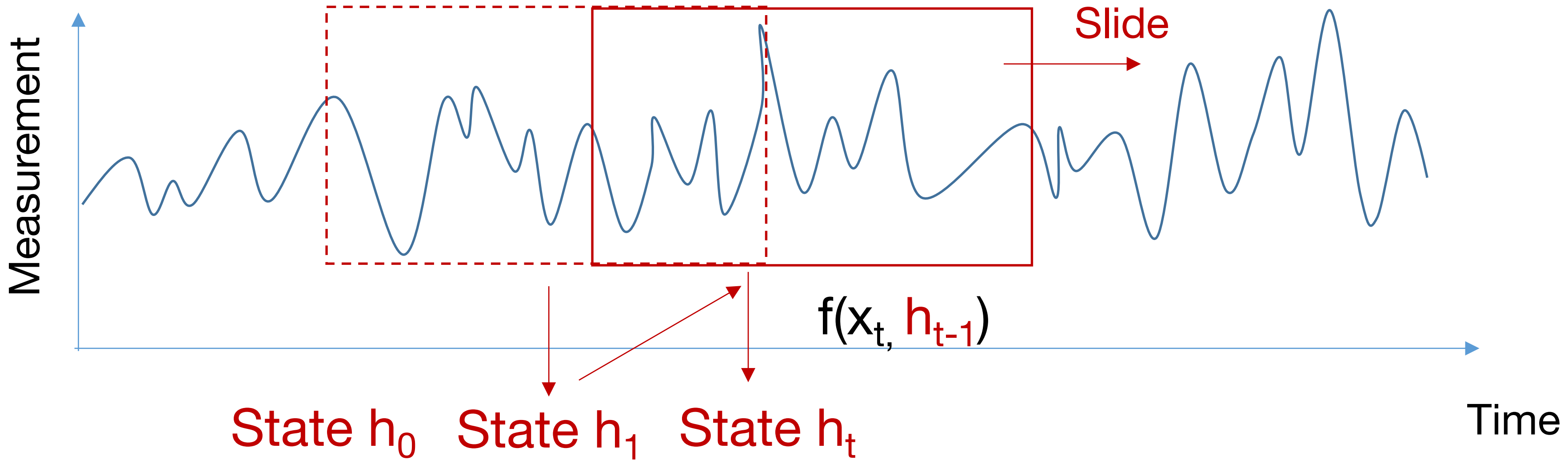


Recurrent neural networks: Generate states ("hidden units") to use to inform subsequent decisions

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

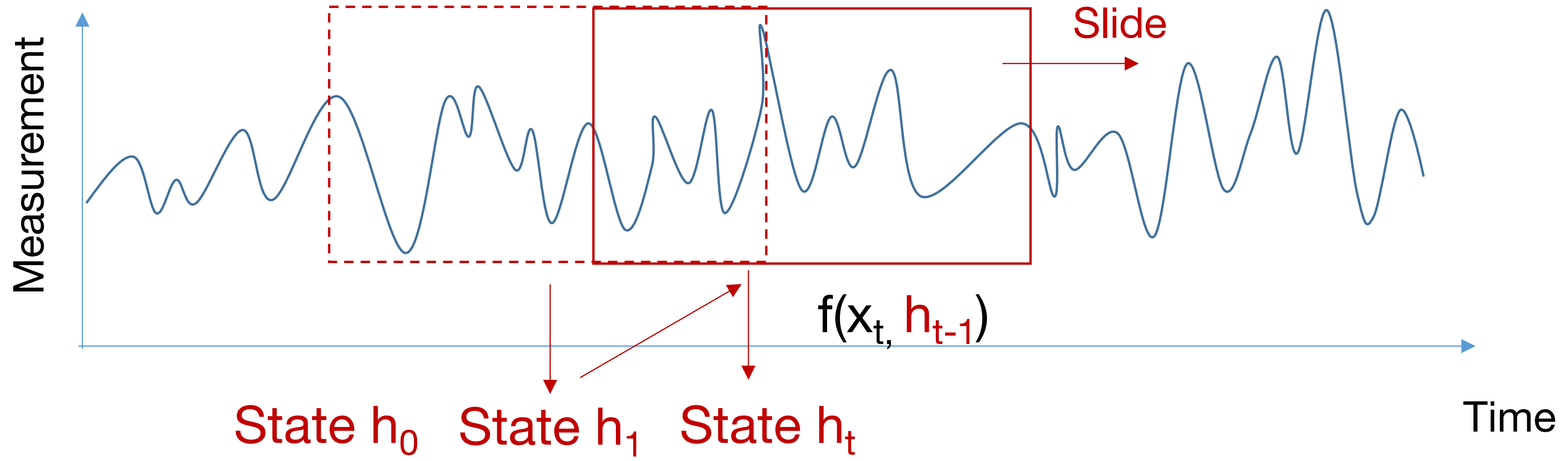


Recurrent neural networks: Generate states ("hidden units") to use to inform subsequent decisions

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



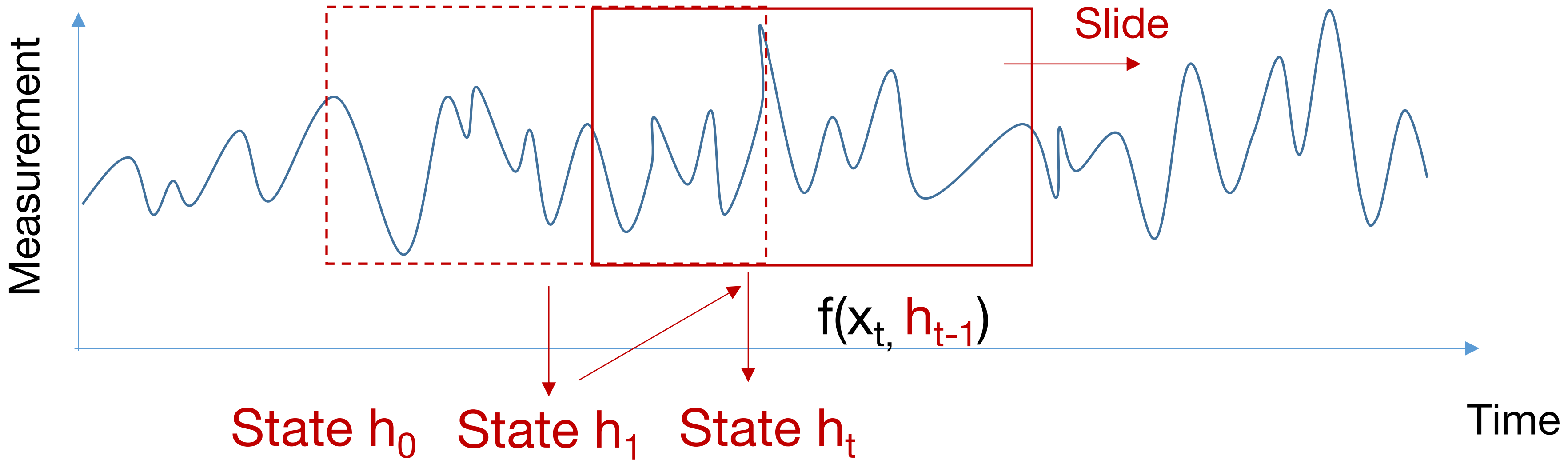
Reasoning unique to temporal data:
- Exploit preferential direction
- Helpful to establish a "memory" of what has been seen in the past
- Effectively learns how to daisy-chain information in signal

Recurrent neural networks in a nutshell

deep imaging

RNN's: Examine signals as a function of time

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



$$\boldsymbol{h}^{(t)} = g^{(t)}\left(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right)$$
$$= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).$$

Recursive structure can be unfolded

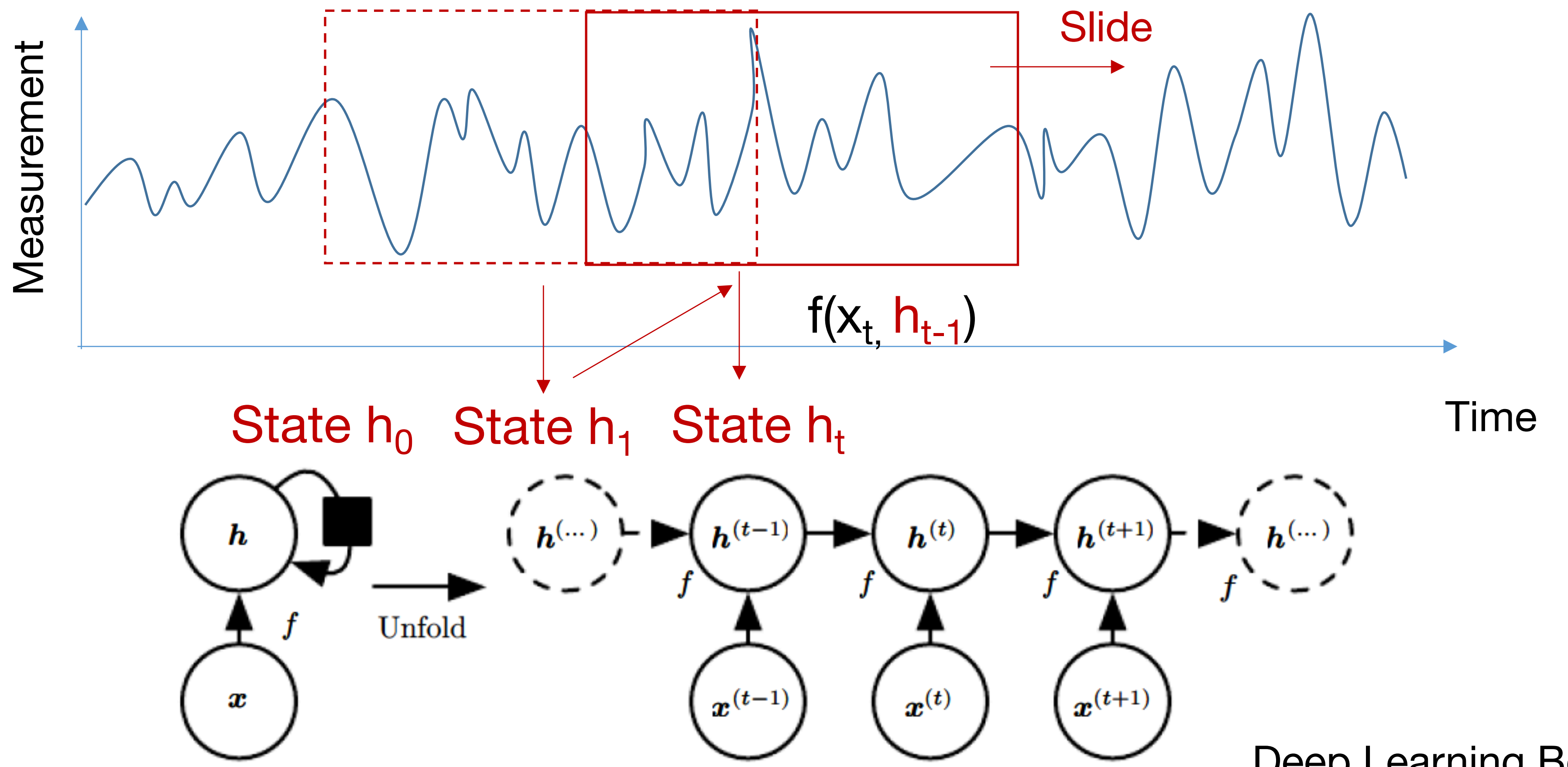
Deep Learning Book, Ch. 10

Recurrent neural networks in a nutshell

deep imaging

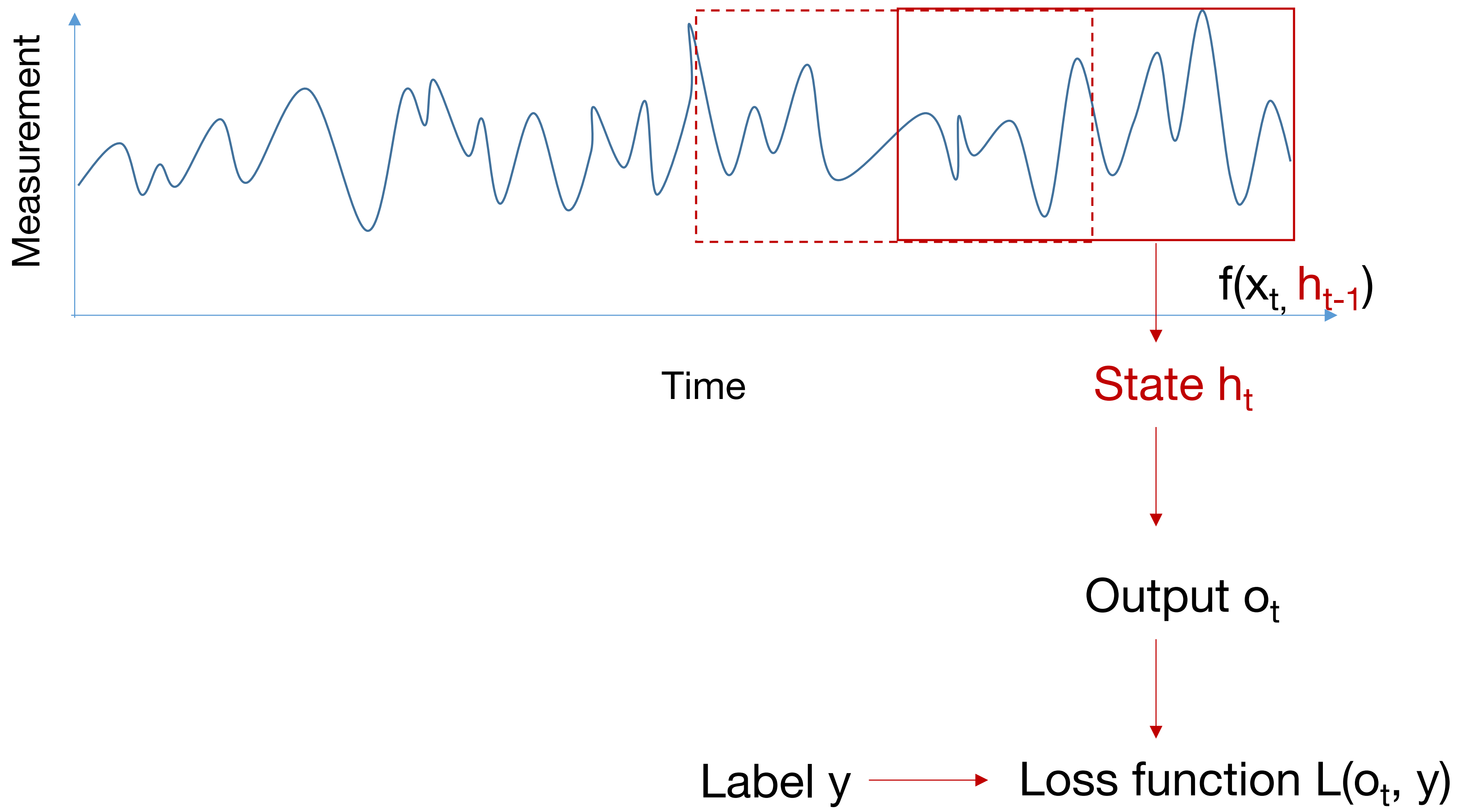
RNN's: Examine signals as a function of time

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

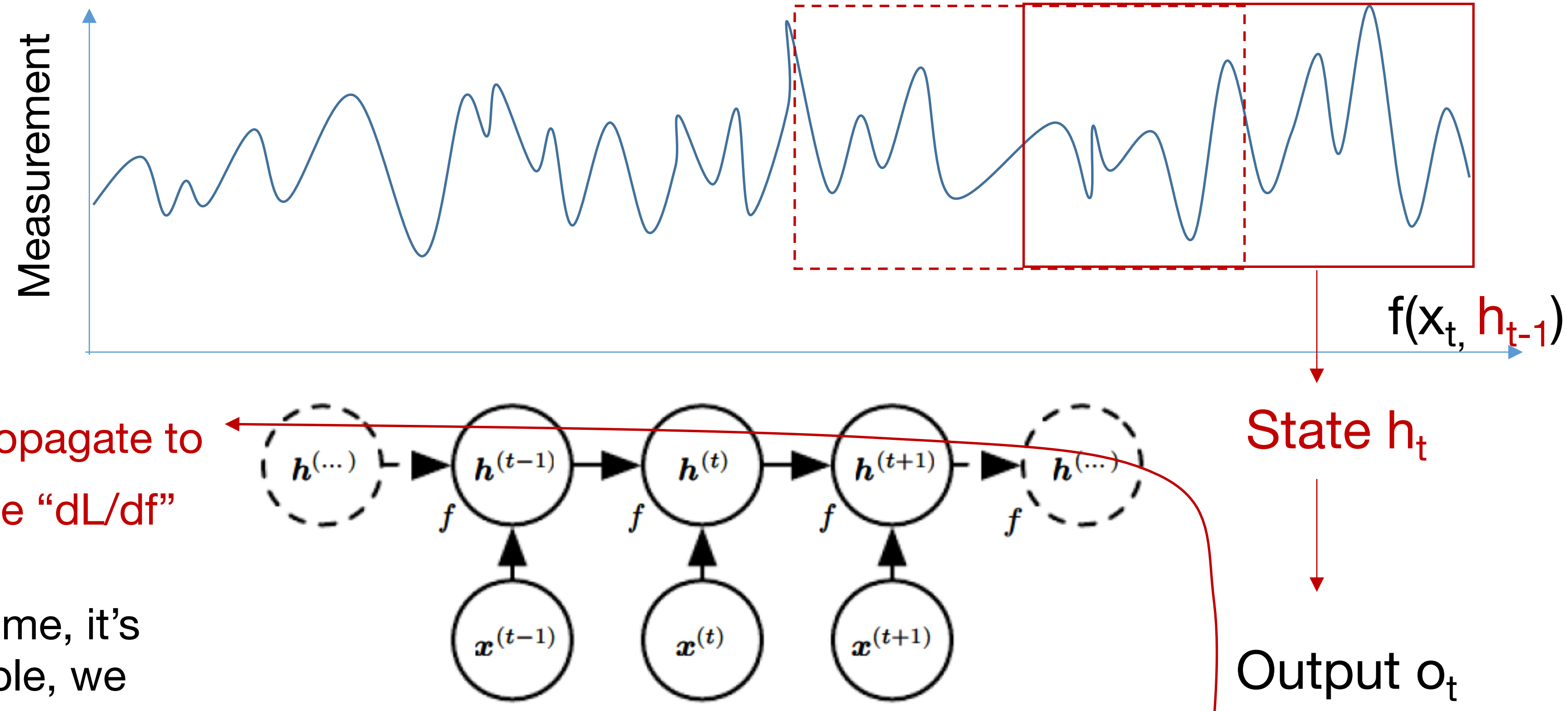


Deep Learning Book, Ch. 10

Many-to-many recurrent neural network



Many-to-one recurrent neural network



Backpropagate to minimize "dL/df"

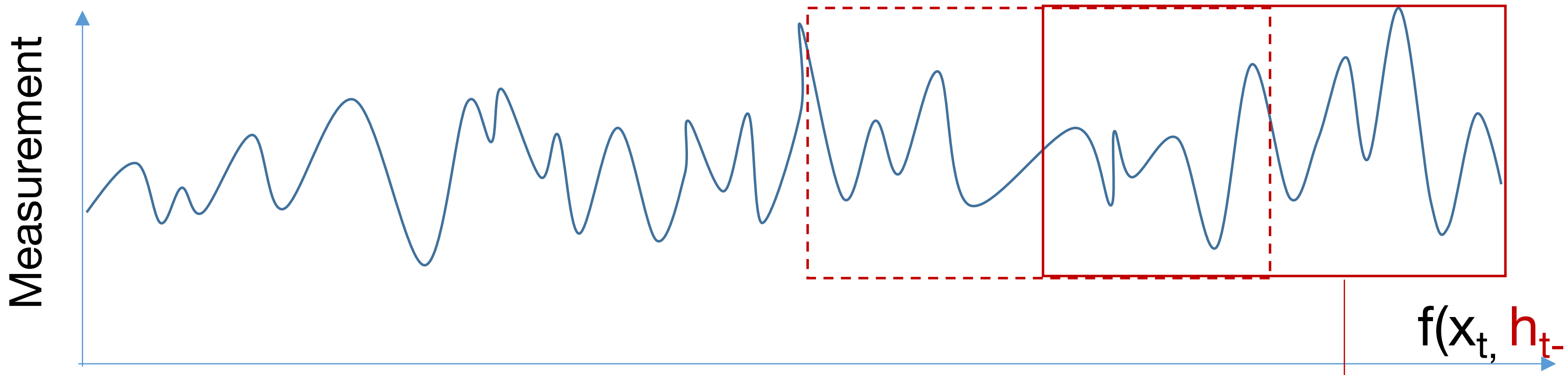
(Trust me, it's possible, we won't derive it….)

State h_t

Output o_t

Label y → Loss function $L(o_t, y)$

Many-to-one recurrent neural network



$f(x_t, h_{t-1})$

State h_t

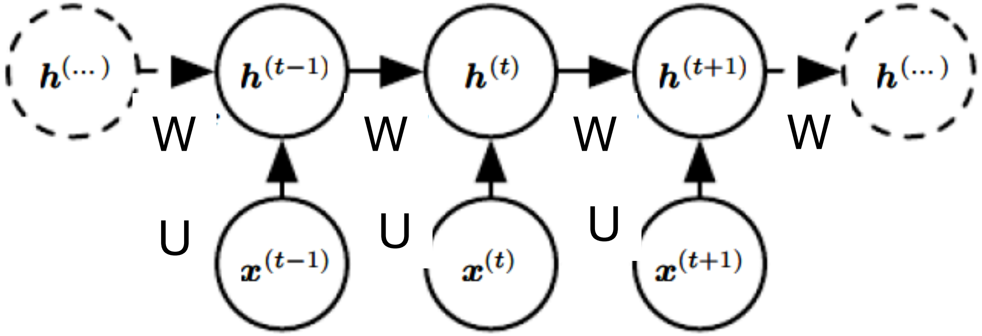
$\boxed{V}$

Output o_t

Label y → Loss function $L(o_t, y)$

Backpropagate to

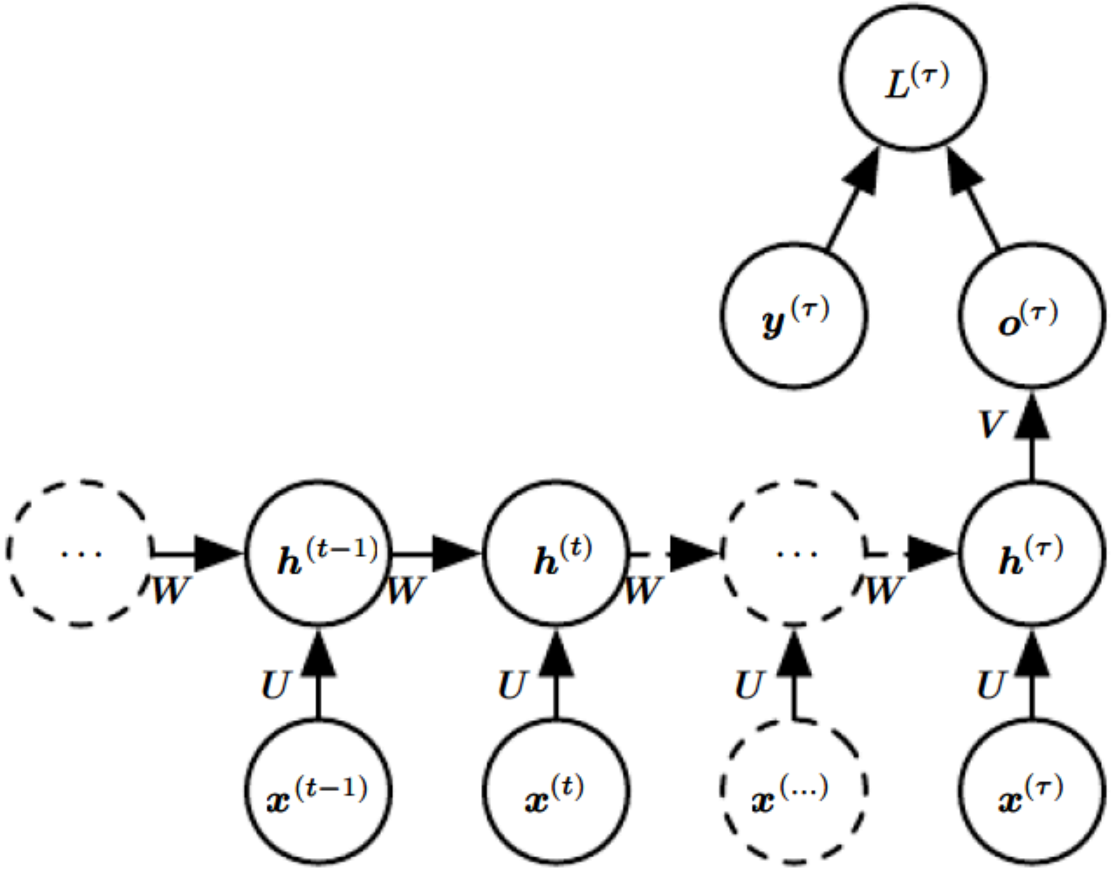
minimize dL/dW, dL/DU



Simple network structure:

$$\boxed{h_t = \mathrm{ReLU}[\mathbf{W}h_{t-1} + \mathbf{U}x_t + b]}$$

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."

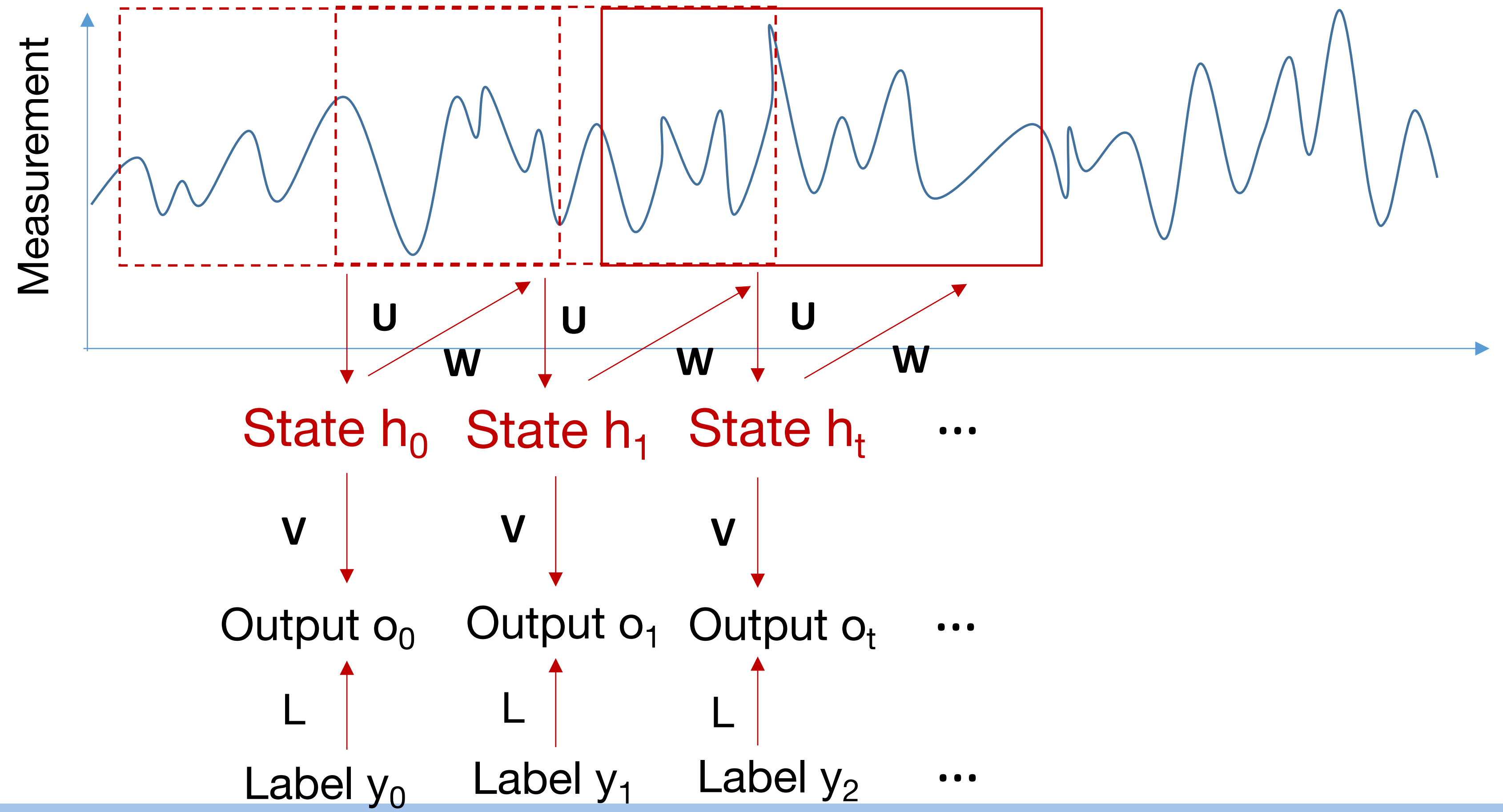
Goal: Extract year each writer went to Nepal from lots of sentences

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

Deep Learning Book, Ch. 10

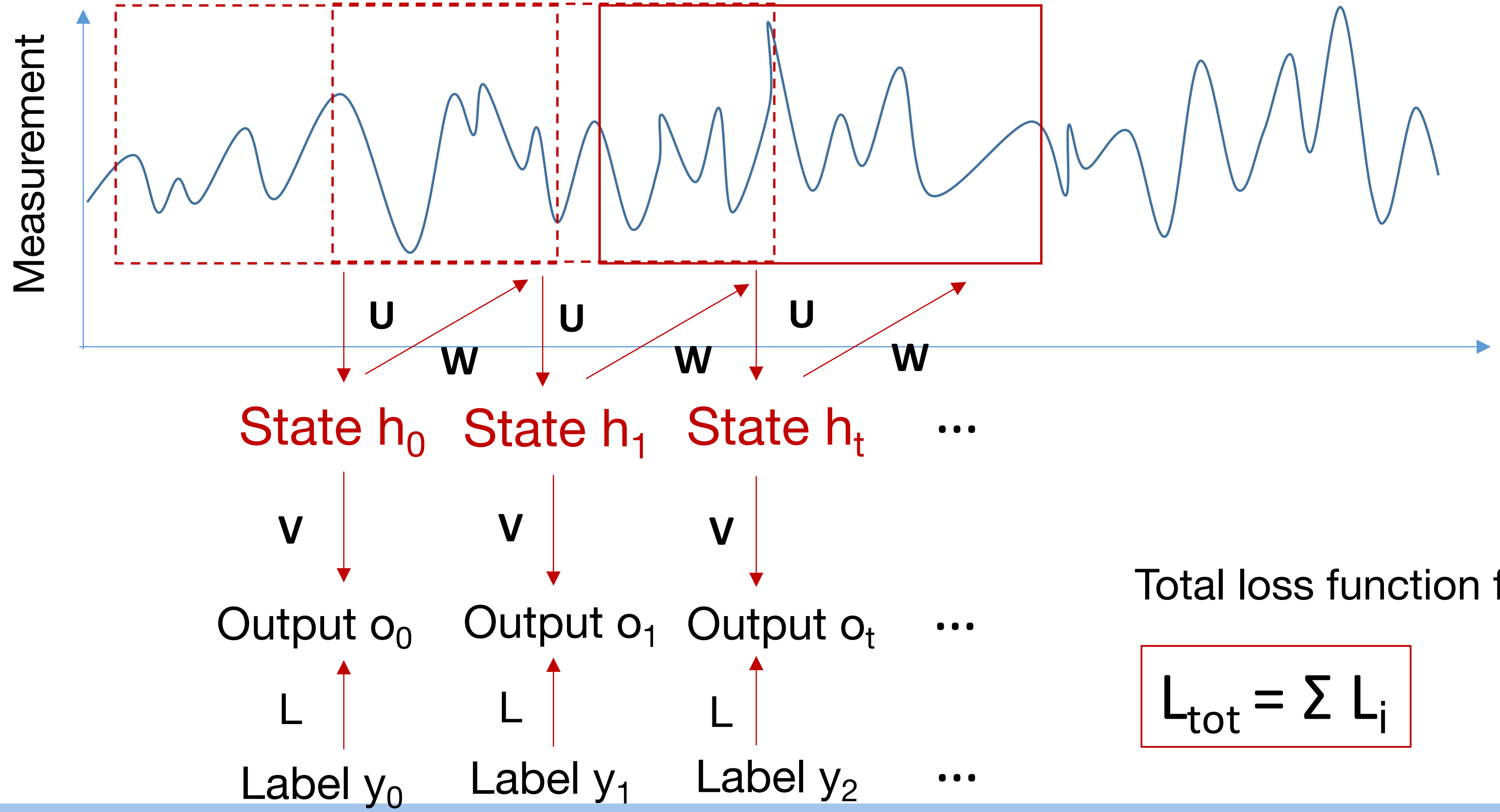
Many-to-many recurrent neural network

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



Many-to-many recurrent neural network

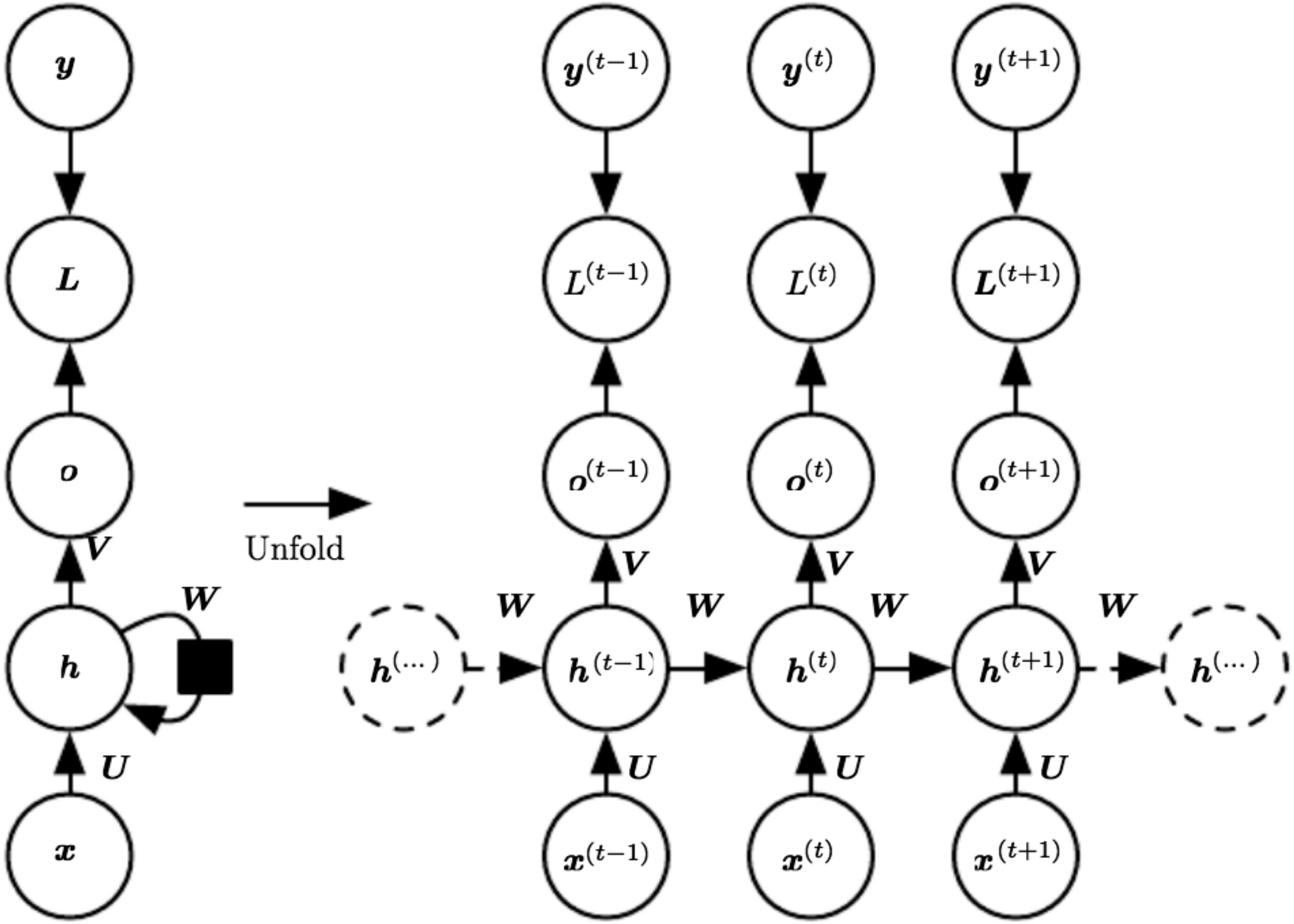
Instead of having one output at the end, can have a trainable output at each step



Total loss function for sequence:

$$\boxed{L_{tot} = \Sigma\, L_i}$$

Many-to-many recurrent neural network

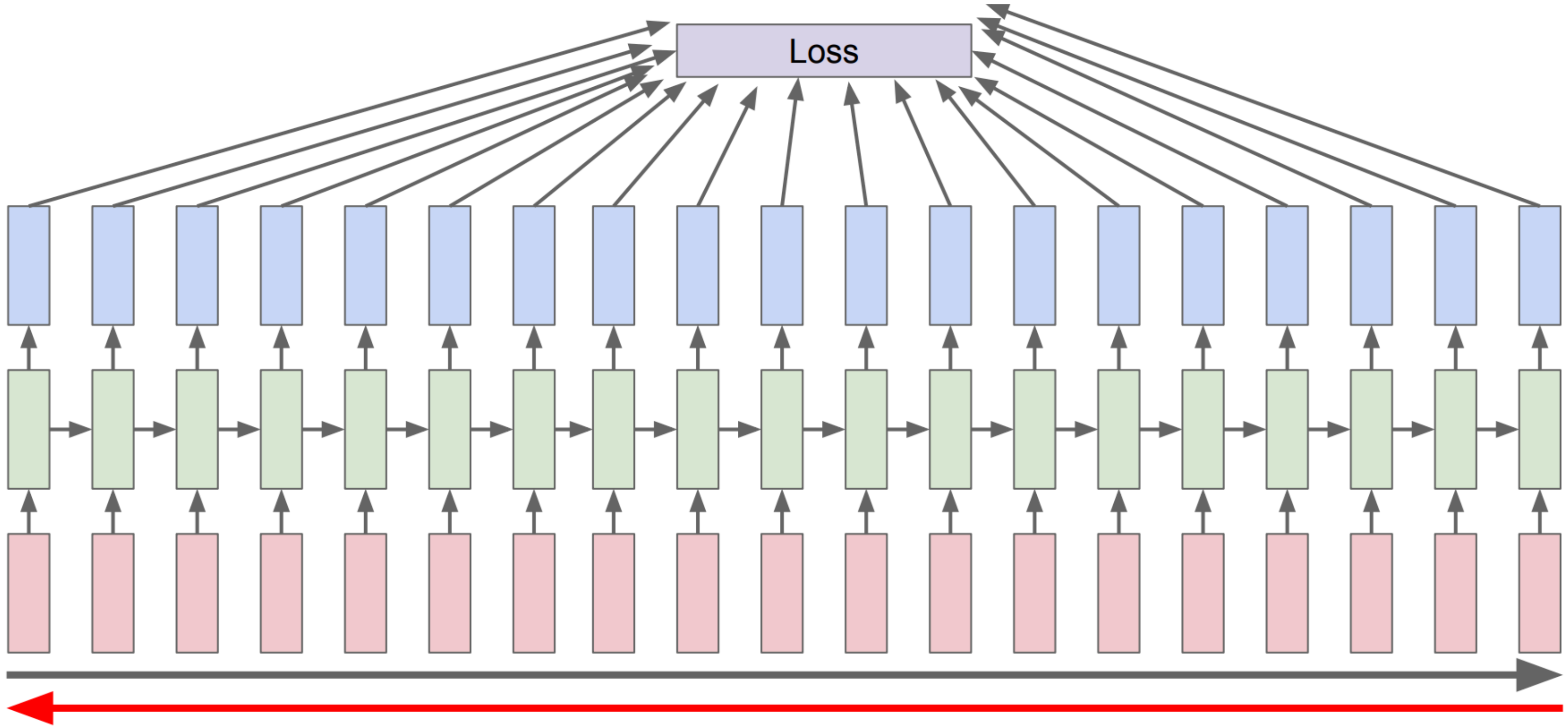


$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)}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

Deep Learning Book, Ch. 10

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

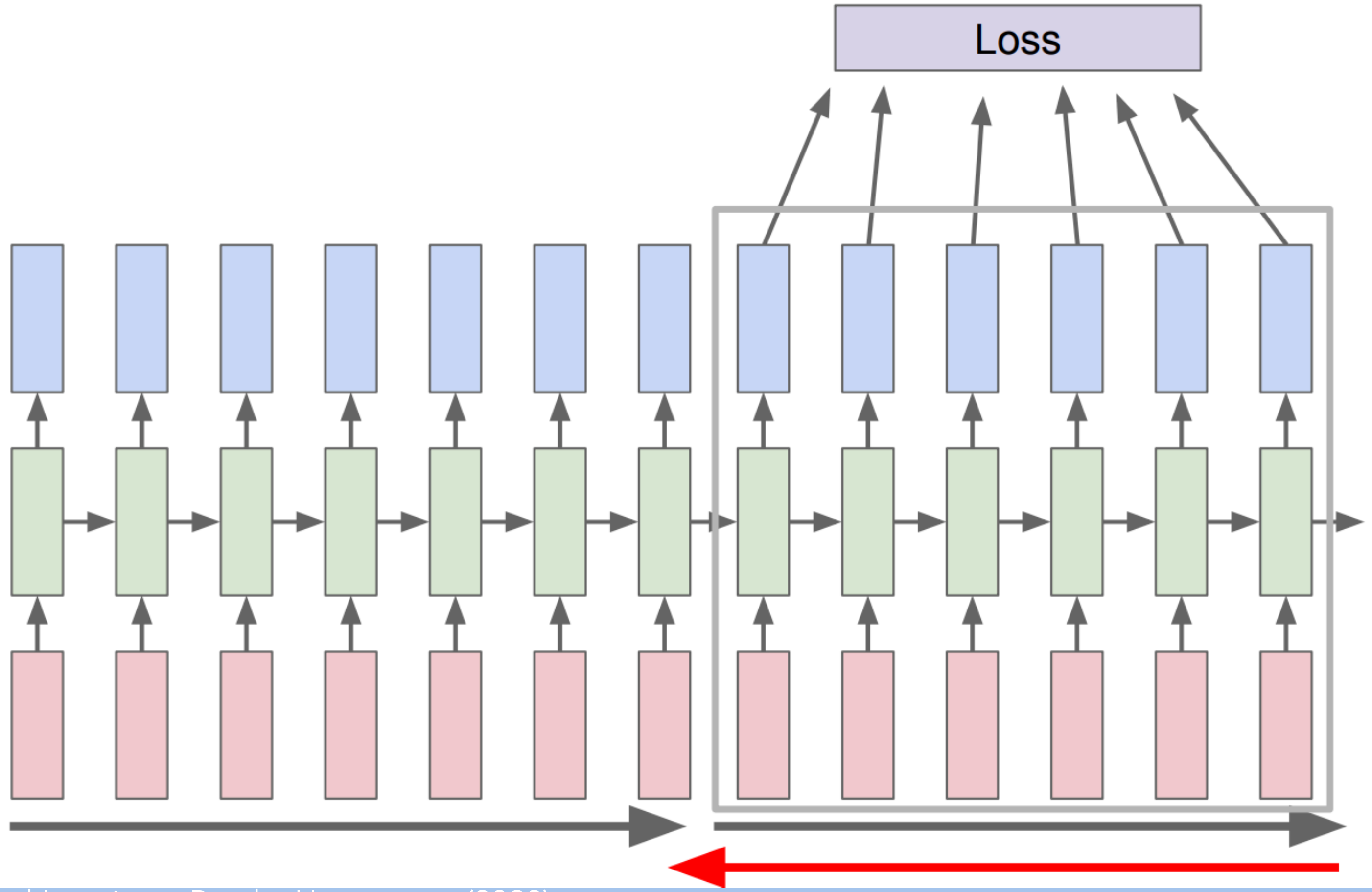
Option #1: Run through full sequence, go back all the way to compute gradient



From Stanford CS231n Lecture 10 slides

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

Option #2: Run through chunks at a time



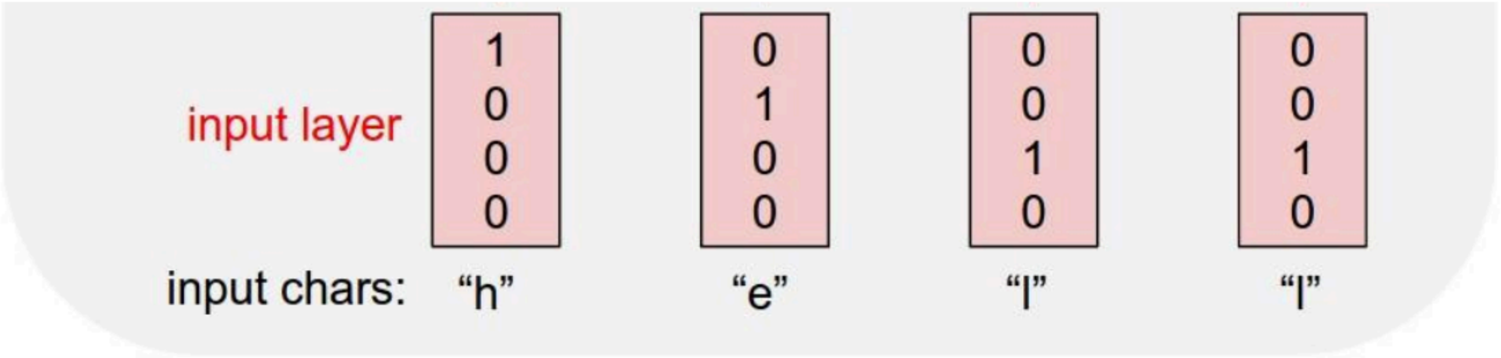
Note: hidden states are always carried forward without any time limit, but you'll just back-propagate loss for a finite number of steps

From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



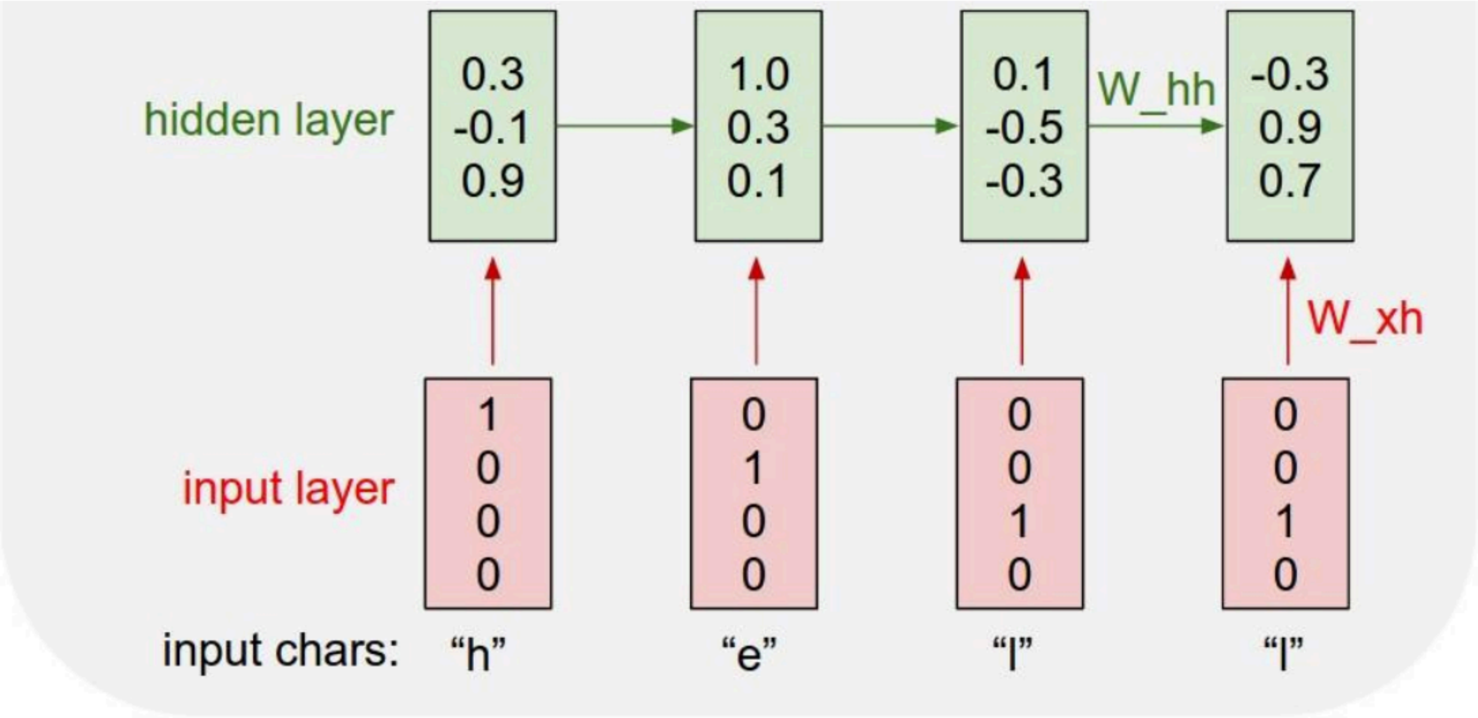
From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model

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

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

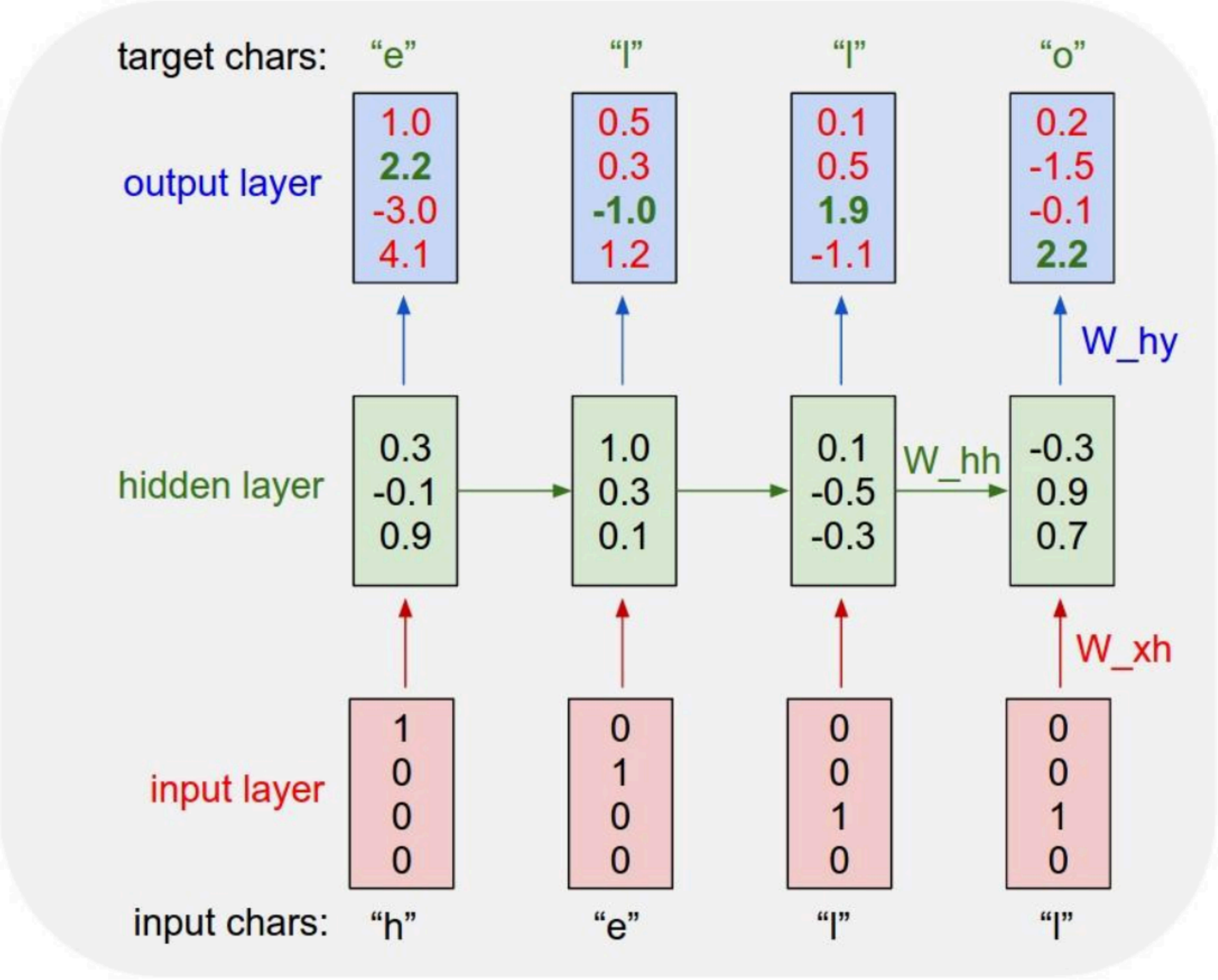


From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

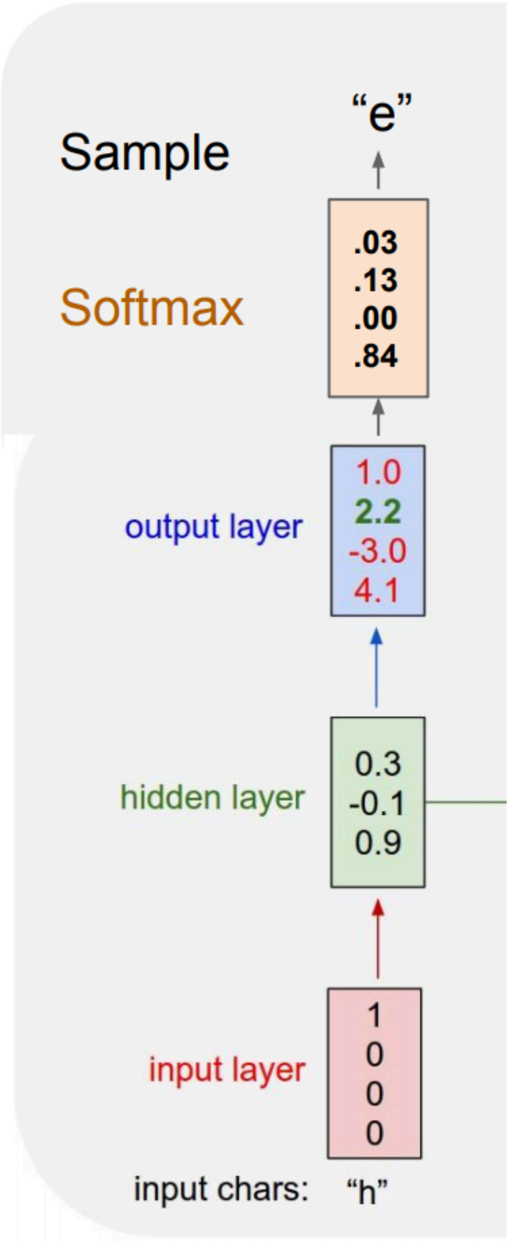


From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

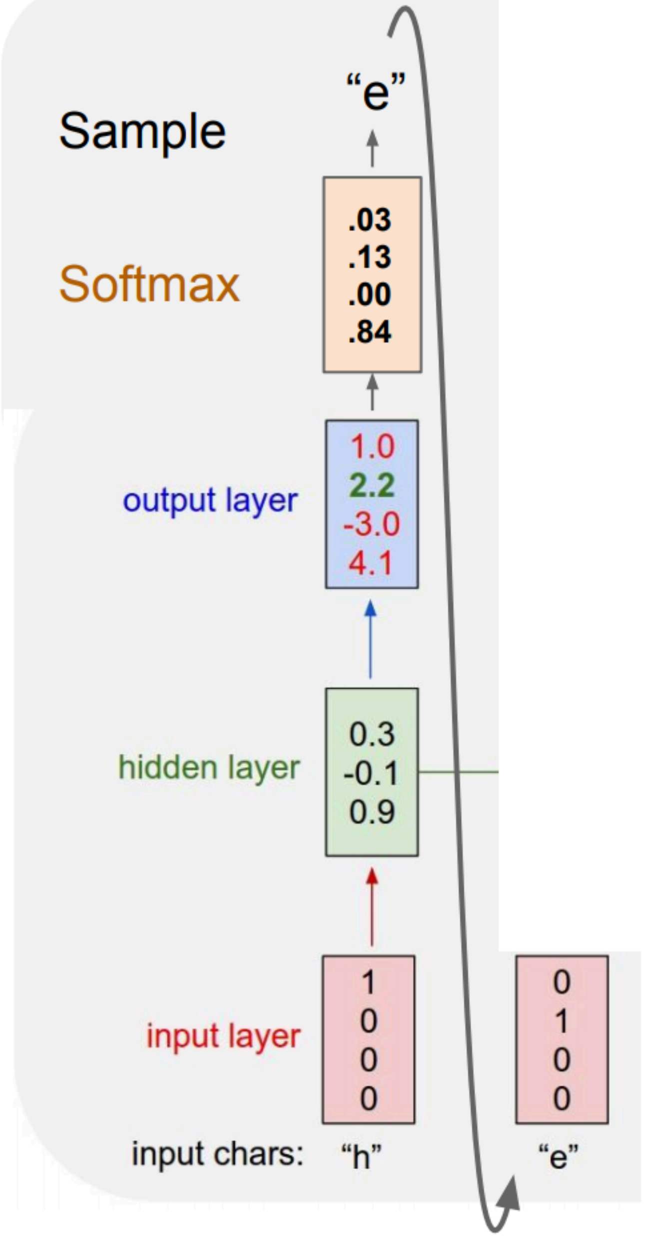


From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

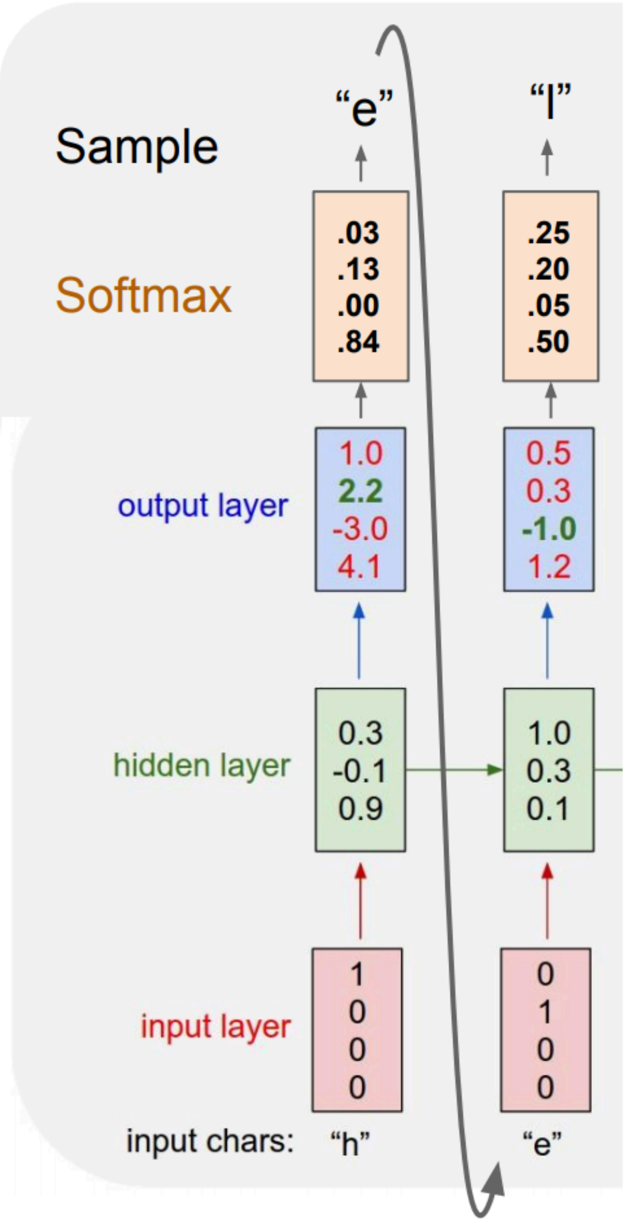


From Stanford CS231n Lecture 10 slides

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

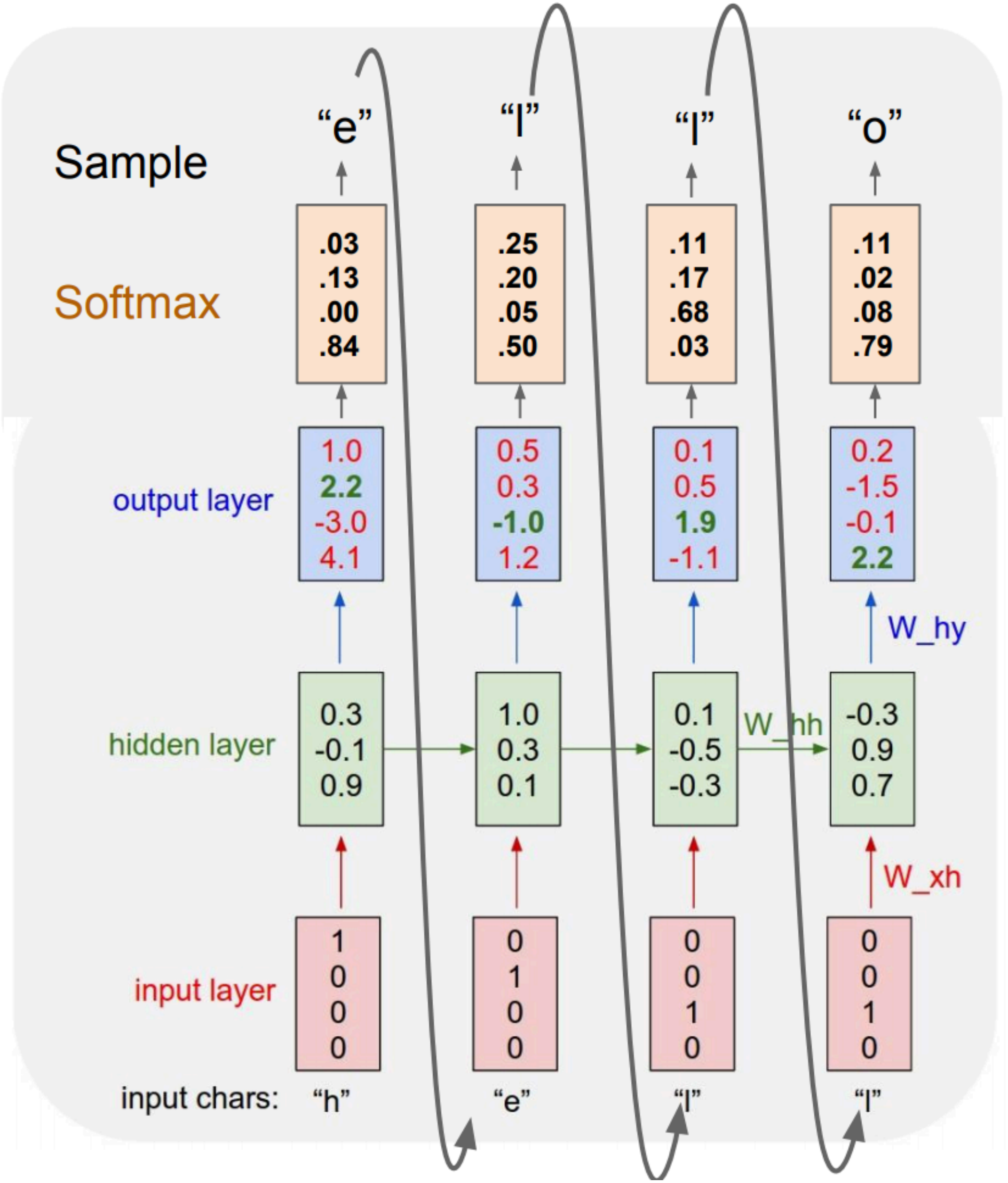


From Stanford CS231n Lecture 10 slides

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

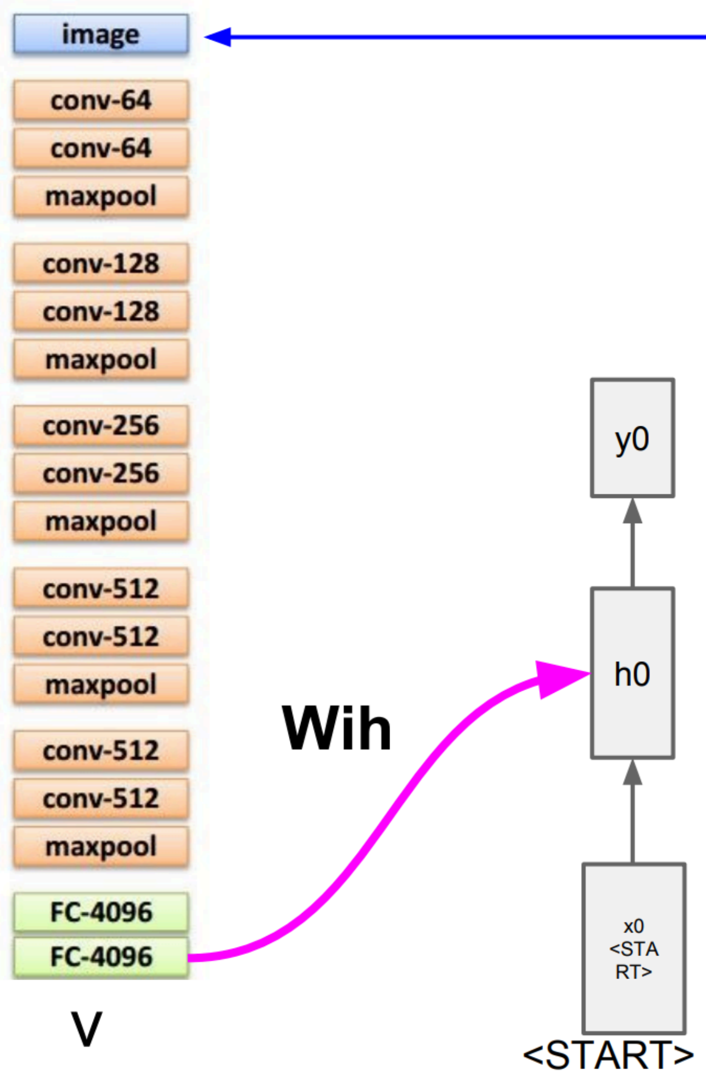
deep imaging

Example: Image captioning

test image

From Stanford CS231n Lecture 10 slides

Example: Image captioning



test image

before:

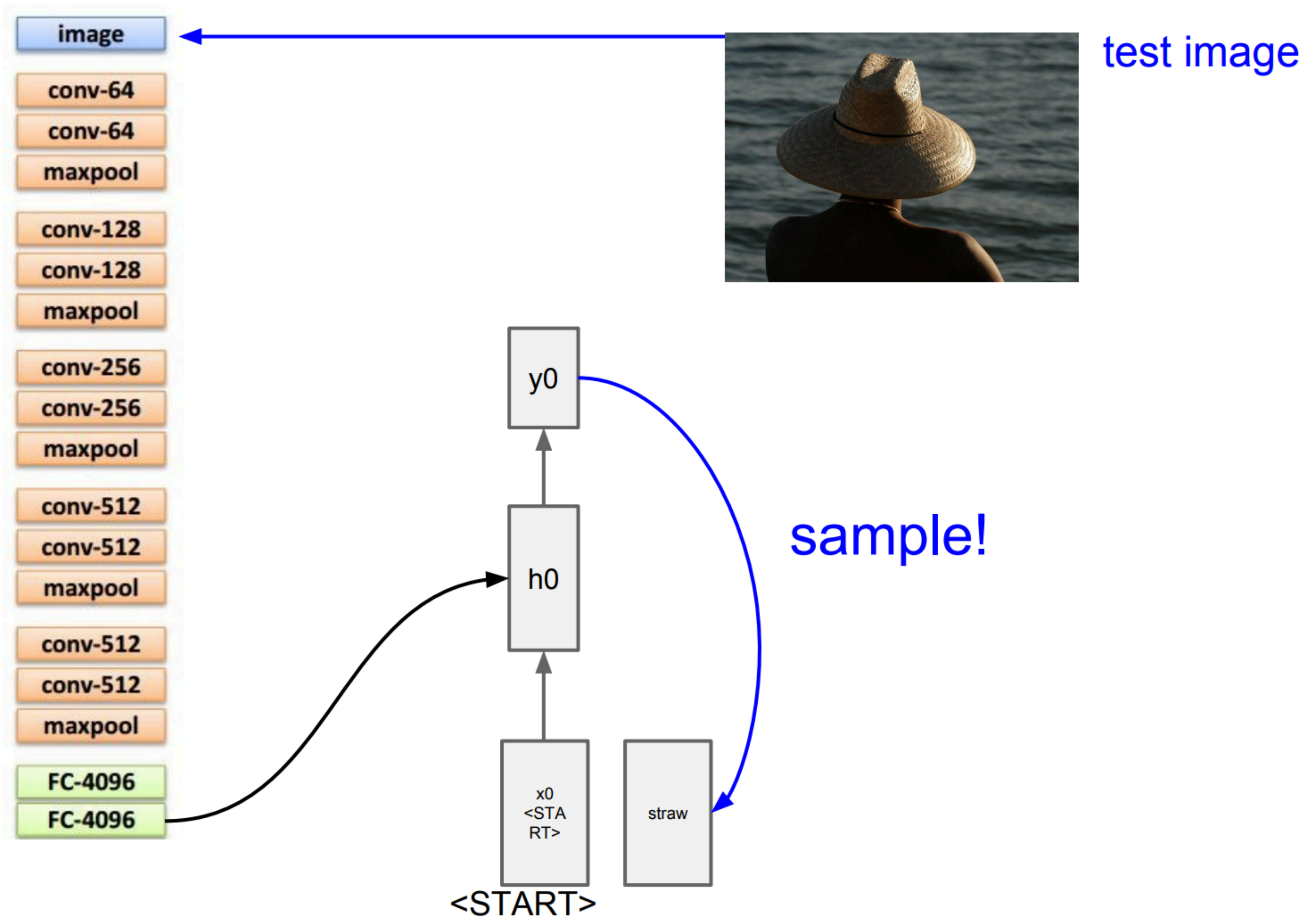
$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

now:

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

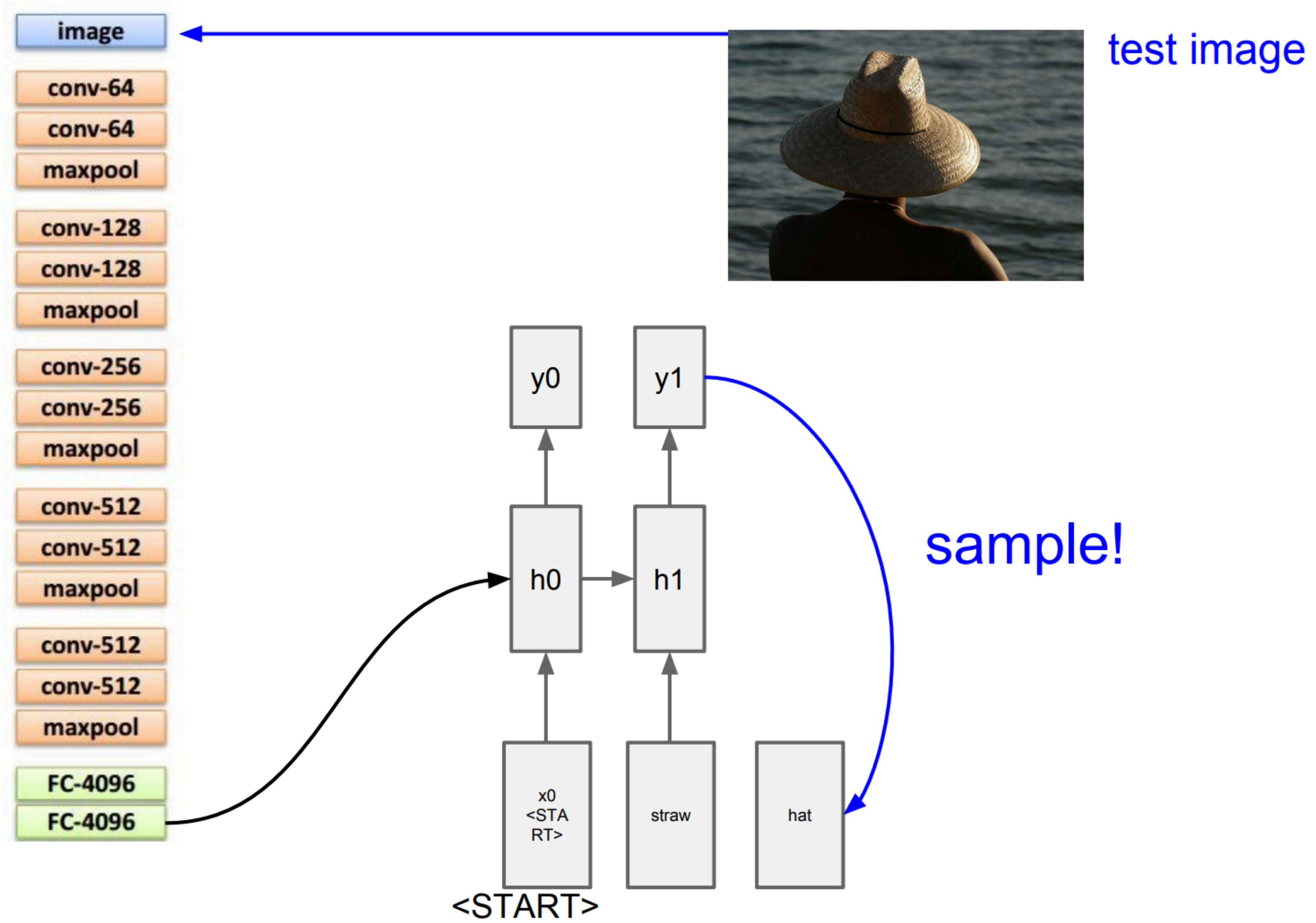
From Stanford CS231n Lecture 10 slides

Example: Image captioning



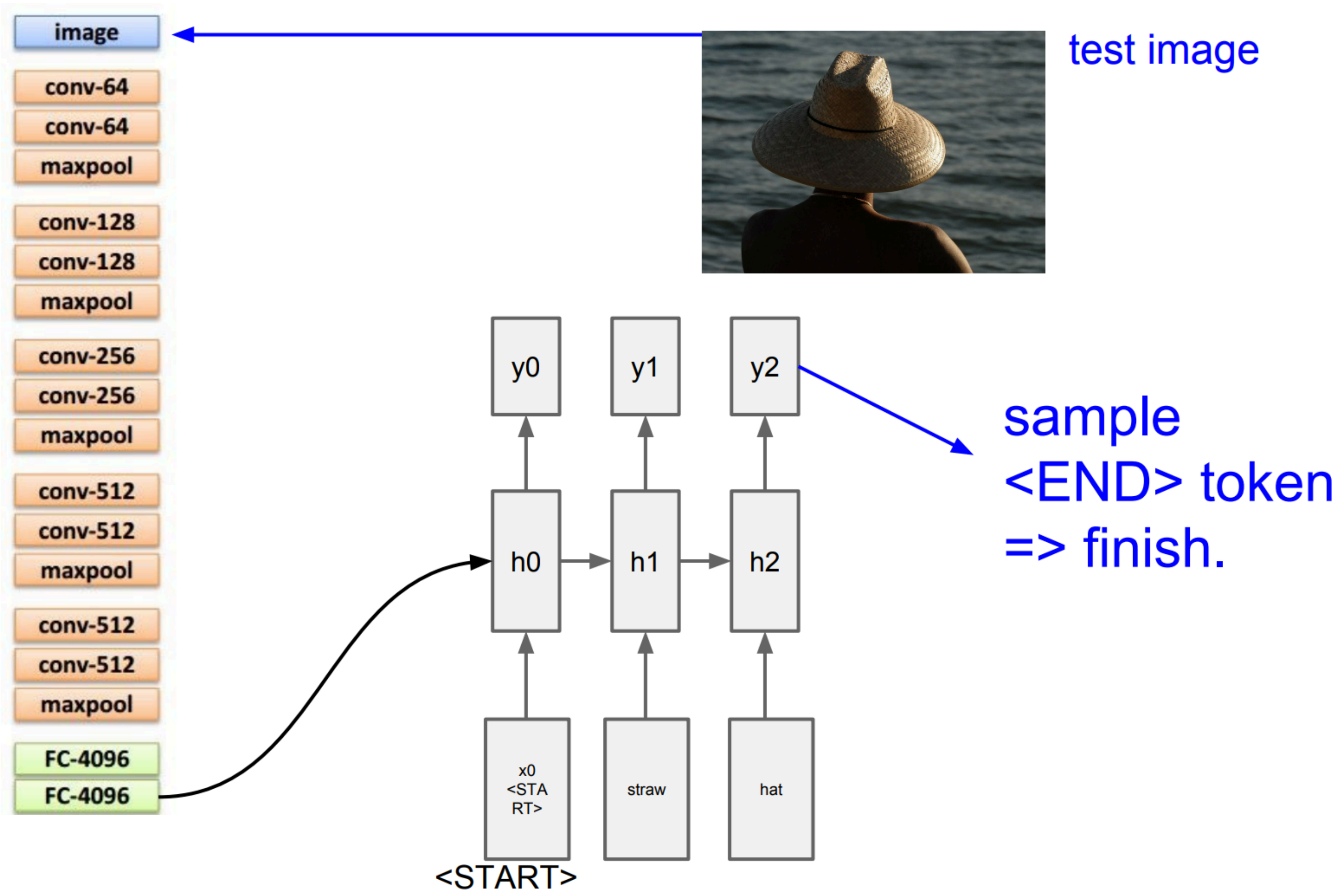
From Stanford CS231n Lecture 10 slides

Example: Image captioning



From Stanford CS231n Lecture 10 slides

Example: Image captioning



From Stanford CS231n Lecture 10 slides

Example: Image captioning

Image Captioning: Failure Cases

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

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

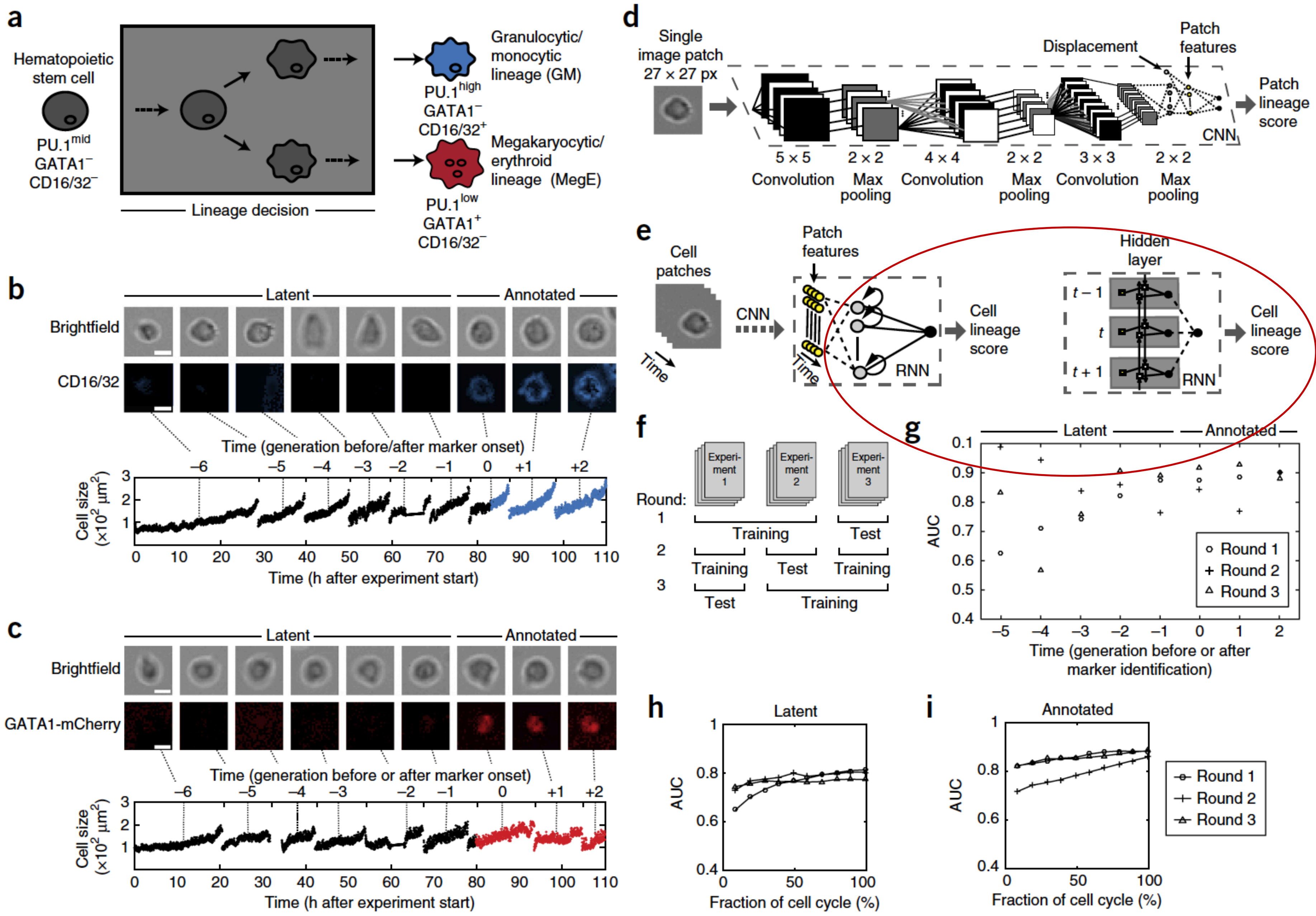
A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

From Stanford CS231n Lecture 10 slides

Prospective identification of hematopoietic lineage choice by deep learning

Felix Buggenthin[1,6], Florian Buettner[1,2,6], Philipp S Hoppe[3,4], Max Endele[3], Manuel Kroiss[1,5], Michael Strasser[1], Michael Schwarzfischer[1], Dirk Loeffler[3,4], Konstantinos D Kokkaliaris[3,4], Oliver Hilsenbeck[3,4], Timm Schroeder[3,4], Fabian J Theis[1,5] & Carsten Marr[1]



deep imaging

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

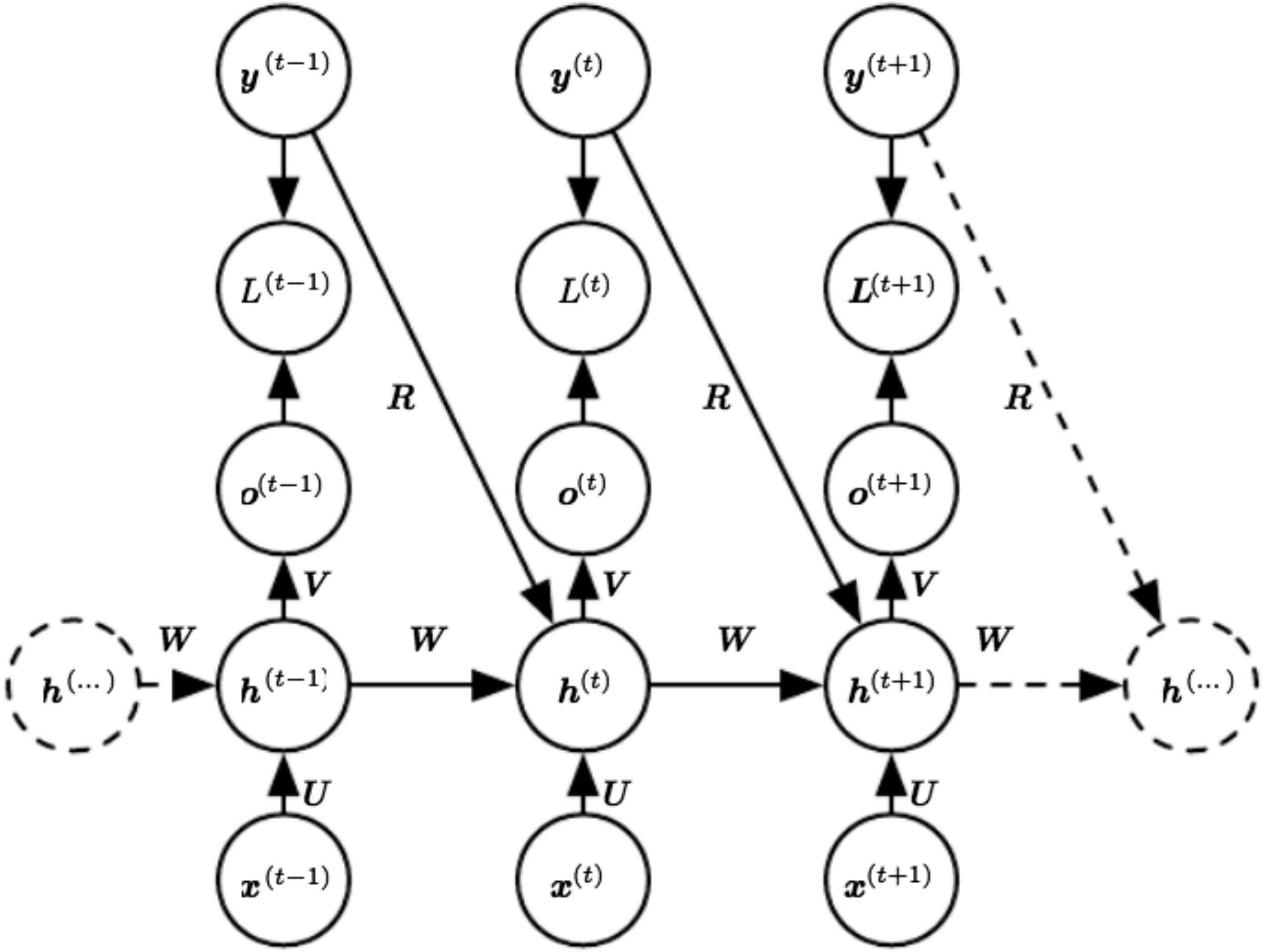
1. Not taking advantage of structure of output labels (assuming they are conditionally independent)

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

Let the network become dependent on past labels as well:

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

Conditional recurrent neural network



deep imaging

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)

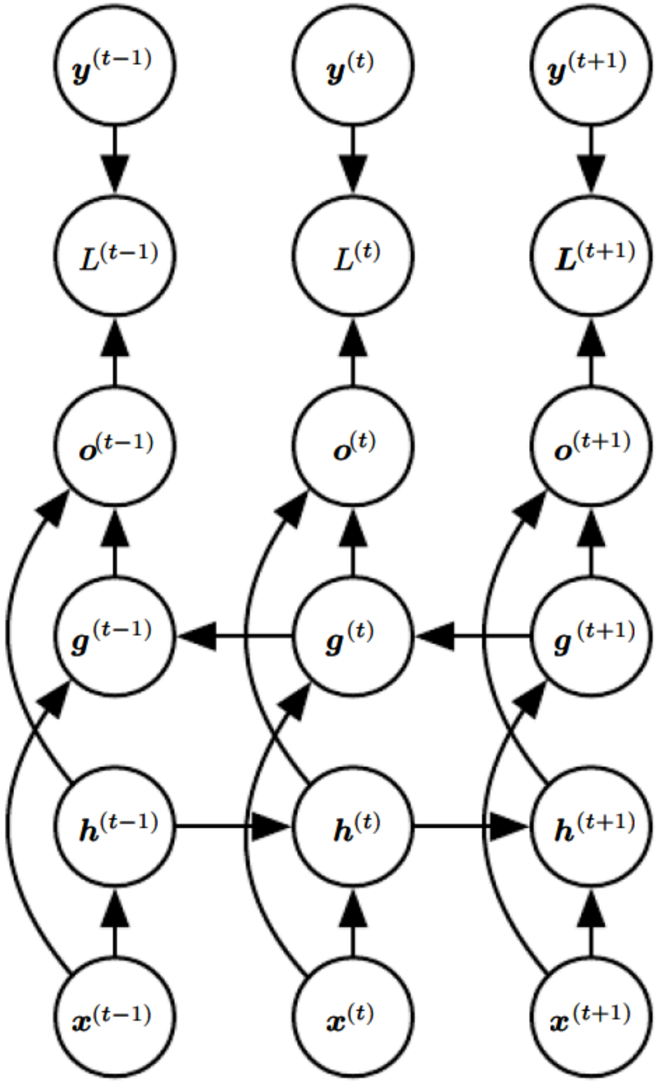
$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

Let the network become dependent on past labels as well:

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

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.

deep imaging

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)

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

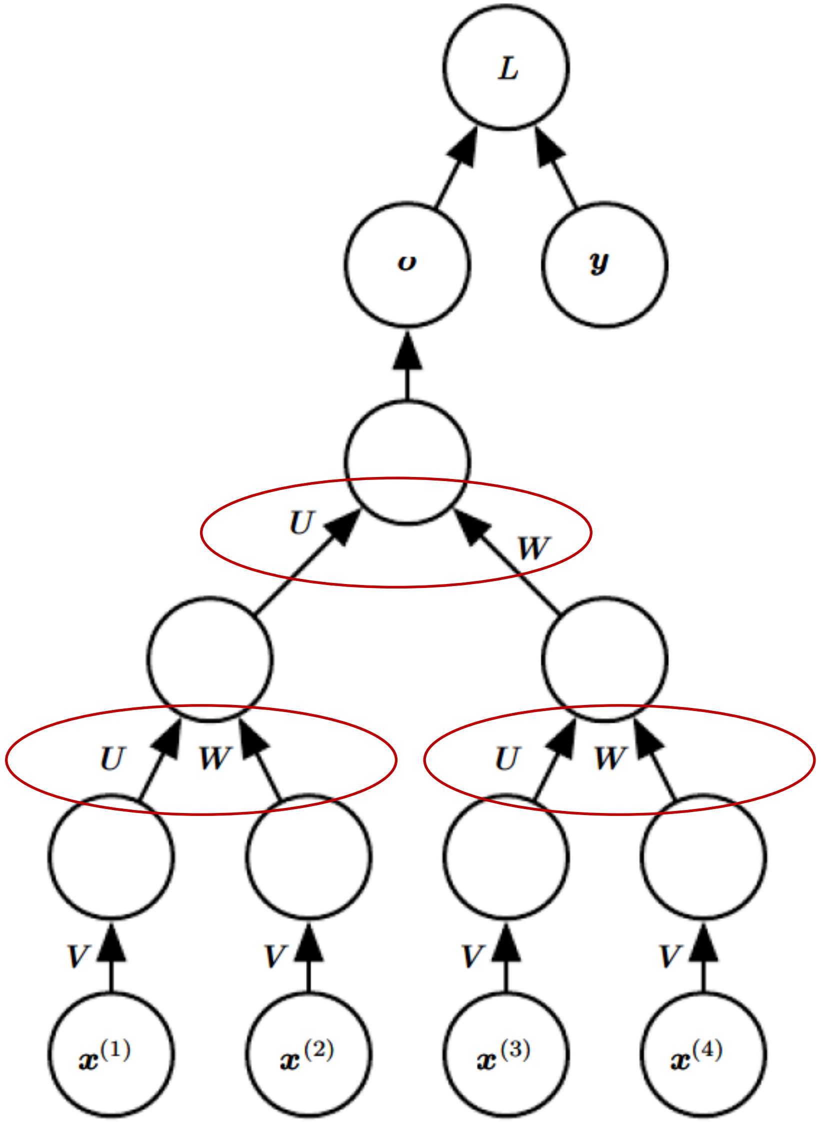
Let the network become dependent on past labels as well:

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

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 to 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:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top} \boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top} \boldsymbol{h}^{(0)},$$

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

Hidden weights effectively follow a recursive relationship:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top}\boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top}\boldsymbol{h}^{(0)},$$

If W admits it, can perform eigenvvector decomposition to obtain,

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

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

Hidden weights effectively follow a recursive relationship:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^\top \boldsymbol{h}^{(t-1)} \longrightarrow \boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^t\right)^\top \boldsymbol{h}^{(0)},$$

If W admits it, can perform eigenvvector decomposition to obtain,

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

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

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

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

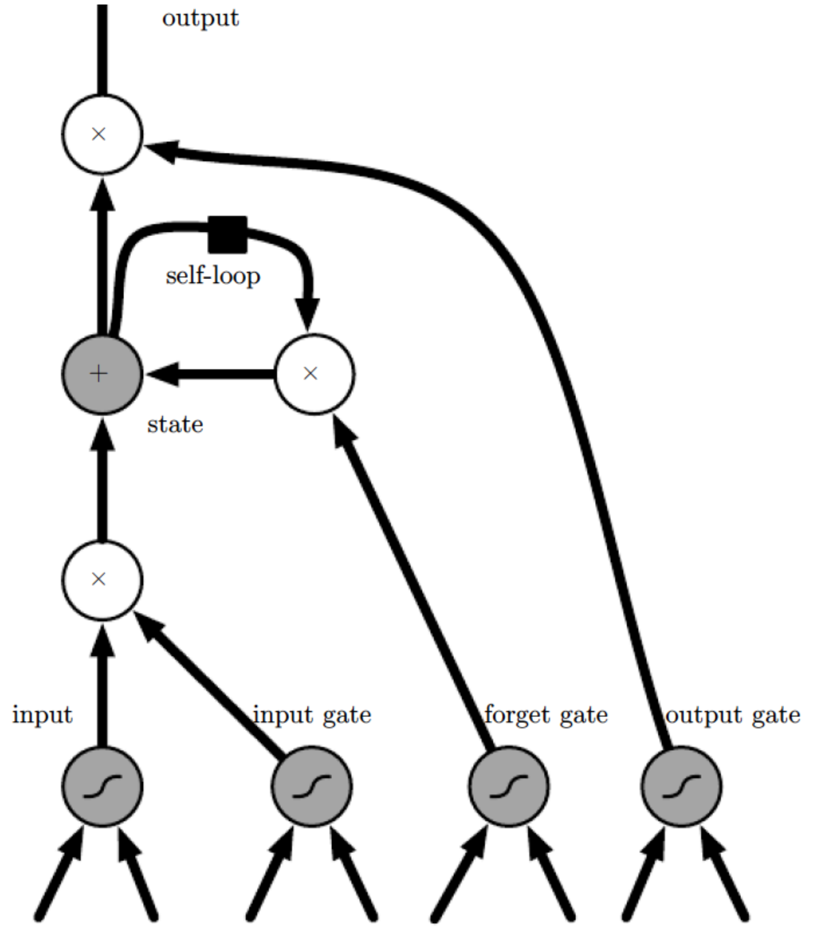
The long short-term memory network

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, ...

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)

The long short-term memory network

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, ...

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:

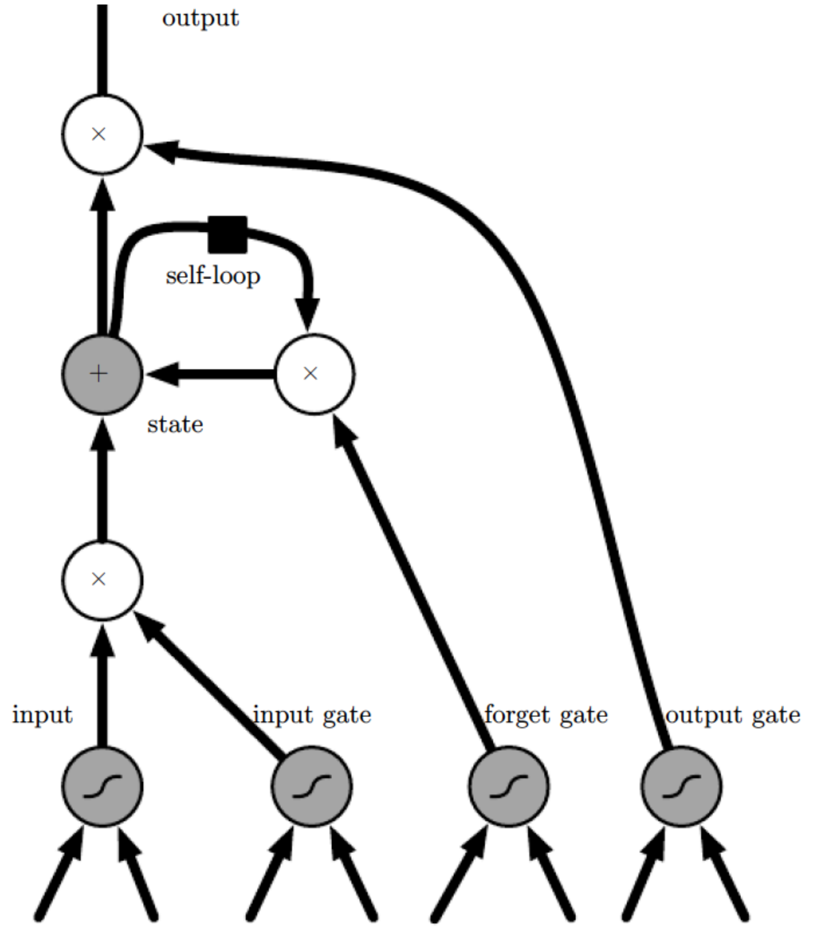
$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

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:

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

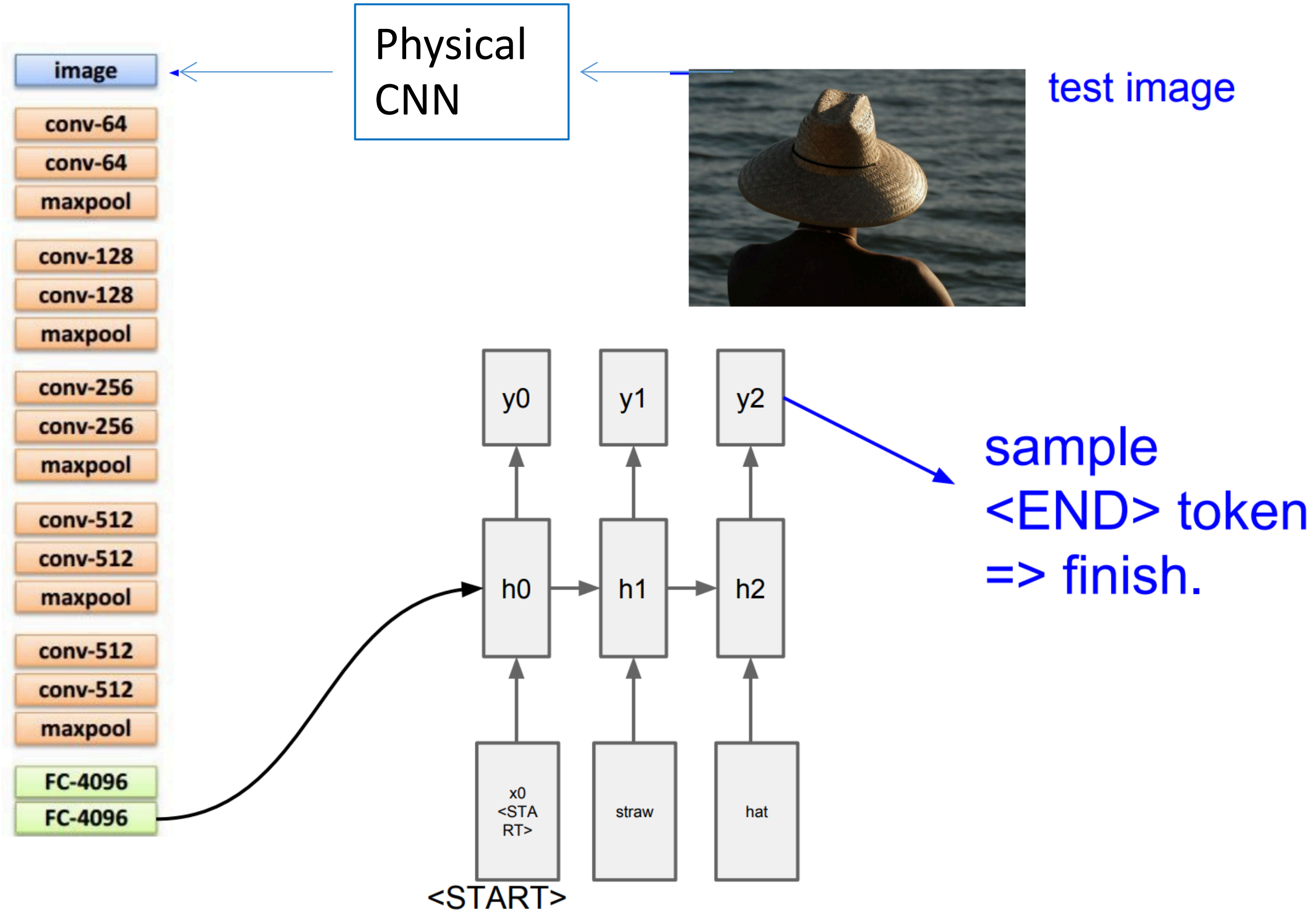


Brainstorming time – physical layers in an RNN???

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?