Recurrent Neural Networks I

CS 4391 Introduction Computer Vision

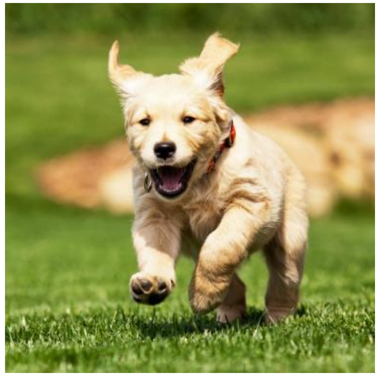
Professor Yu Xiang

The University of Texas at Dallas

Some slides of this lecture are courtesy Stanford CS231n

Single Images

- Convolutional neural networks



Image

CNN

High-level information

- Depth
- Object classes
- Object poses
- Etc.

Sequential Data

- Data depends on time
  - Video

t-1

t

t+1

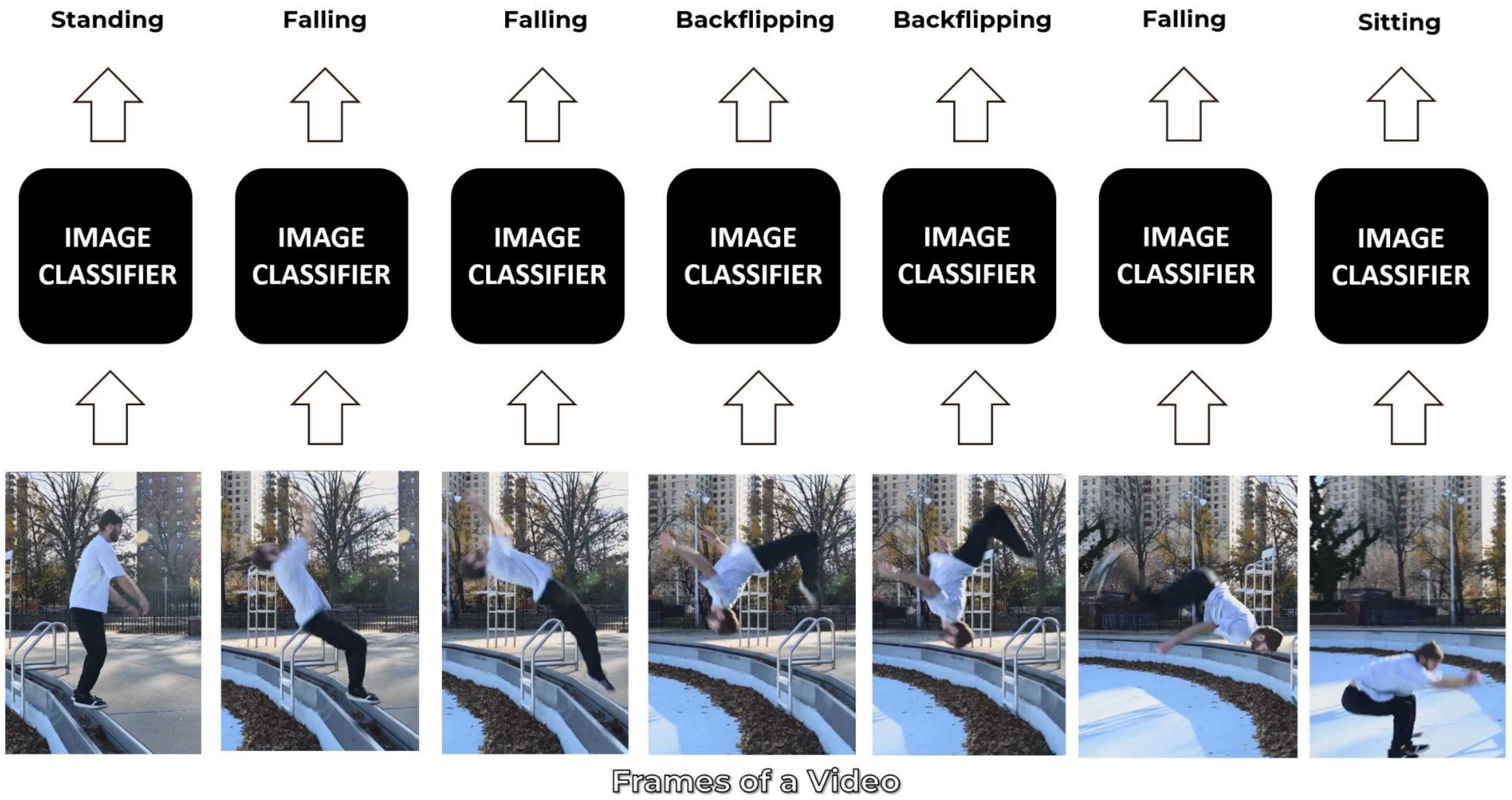
  - Sentence

UT Dallas is a rising public research university in the heart of DFW.

t

Sequential Data Labeling

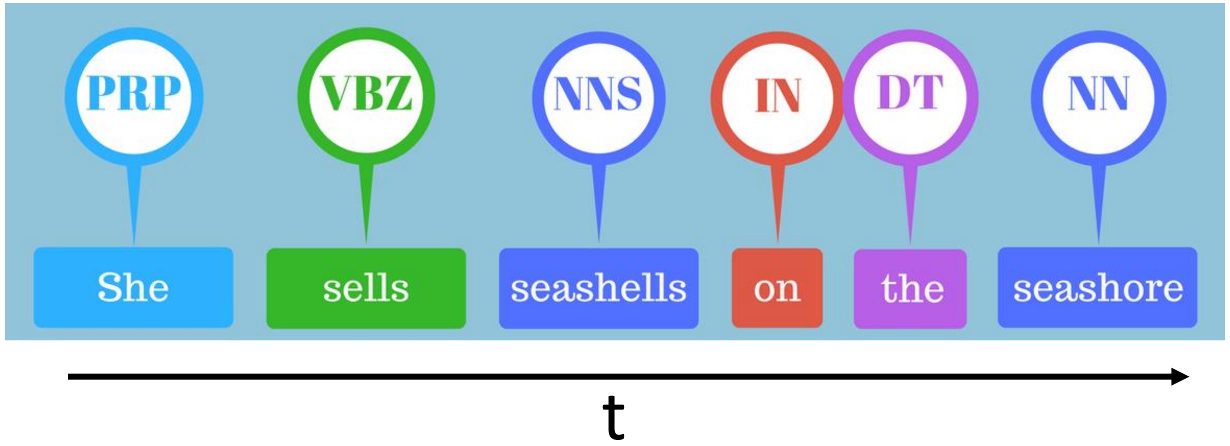
- Video frame labeling



https://bleedai.com/human-activity-recognition-using-tensorflow-cnn-lstm/

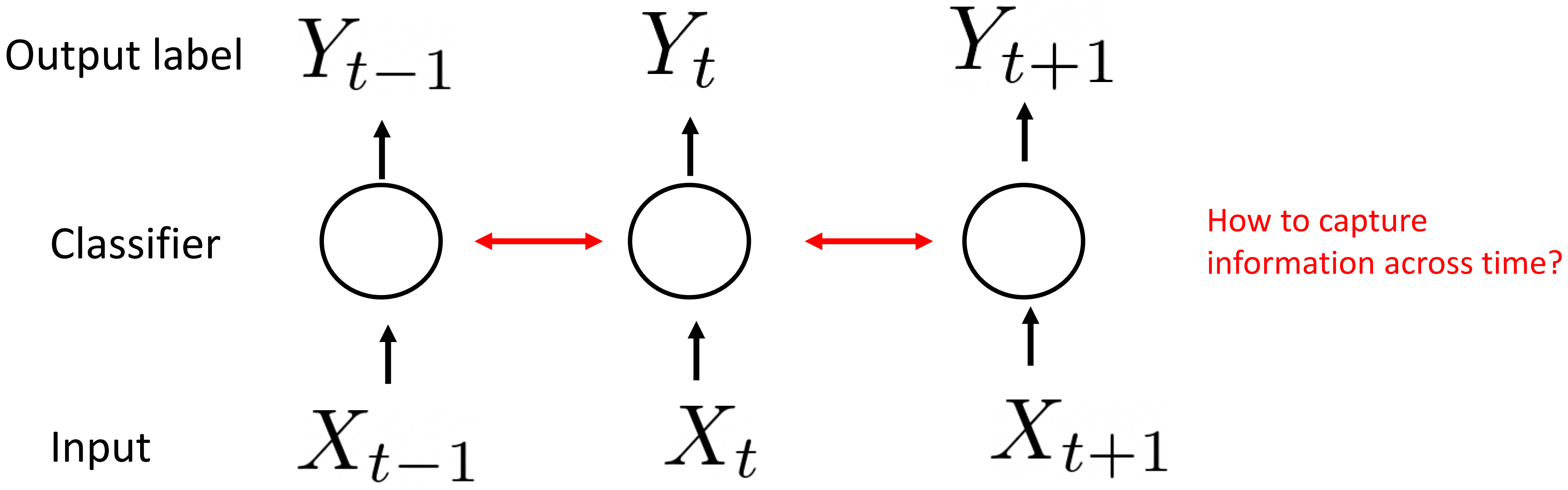
Sequential Data Labeling

- Part-of-speech tagging (grammatical tagging)

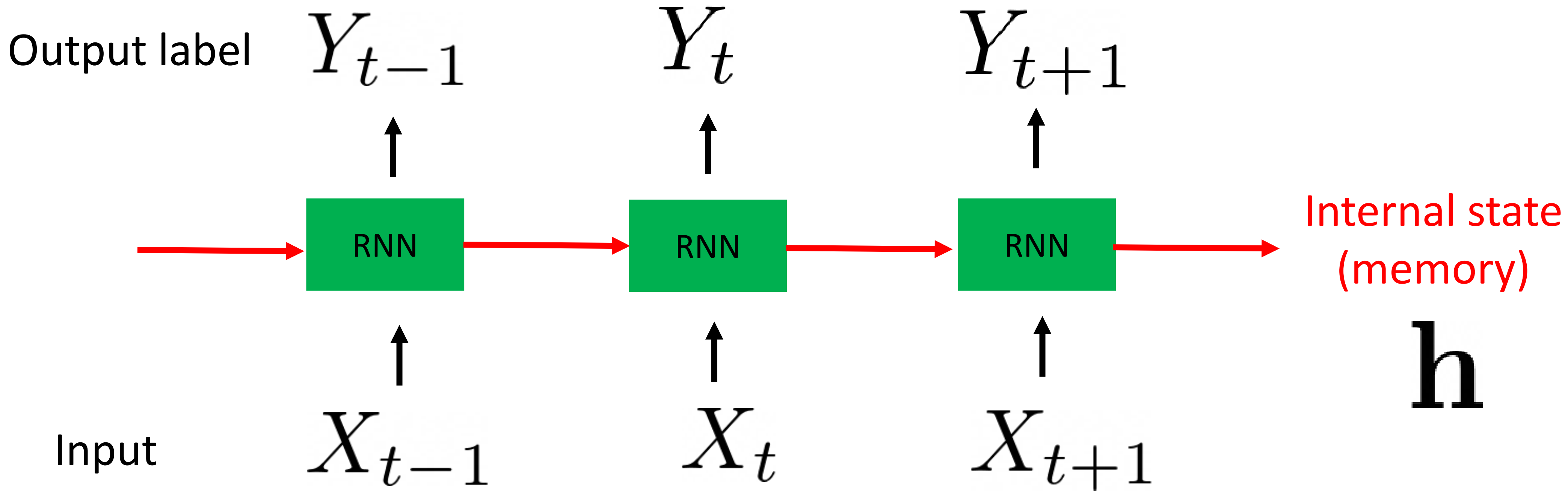


Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

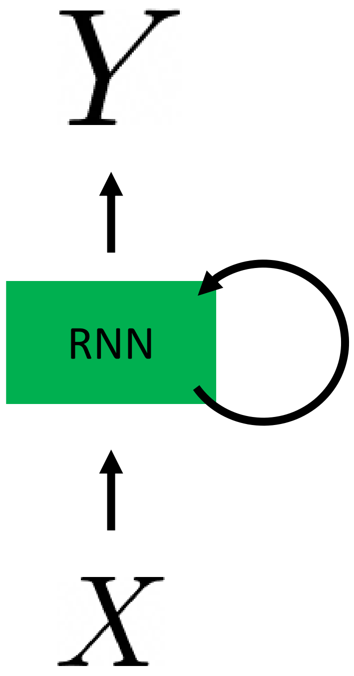
Sequential Data Labeling



Recurrent Neural Networks



Hidden State Update



Updating function with parameters W

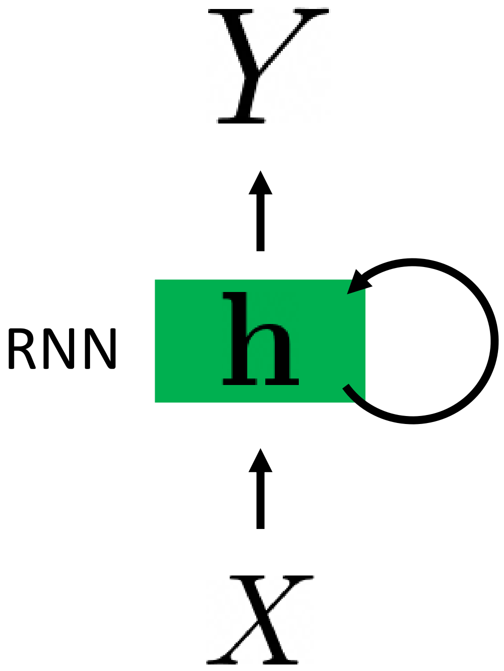
$$\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

Hidden state at time t

Hidden state at time t-1

Input at time t

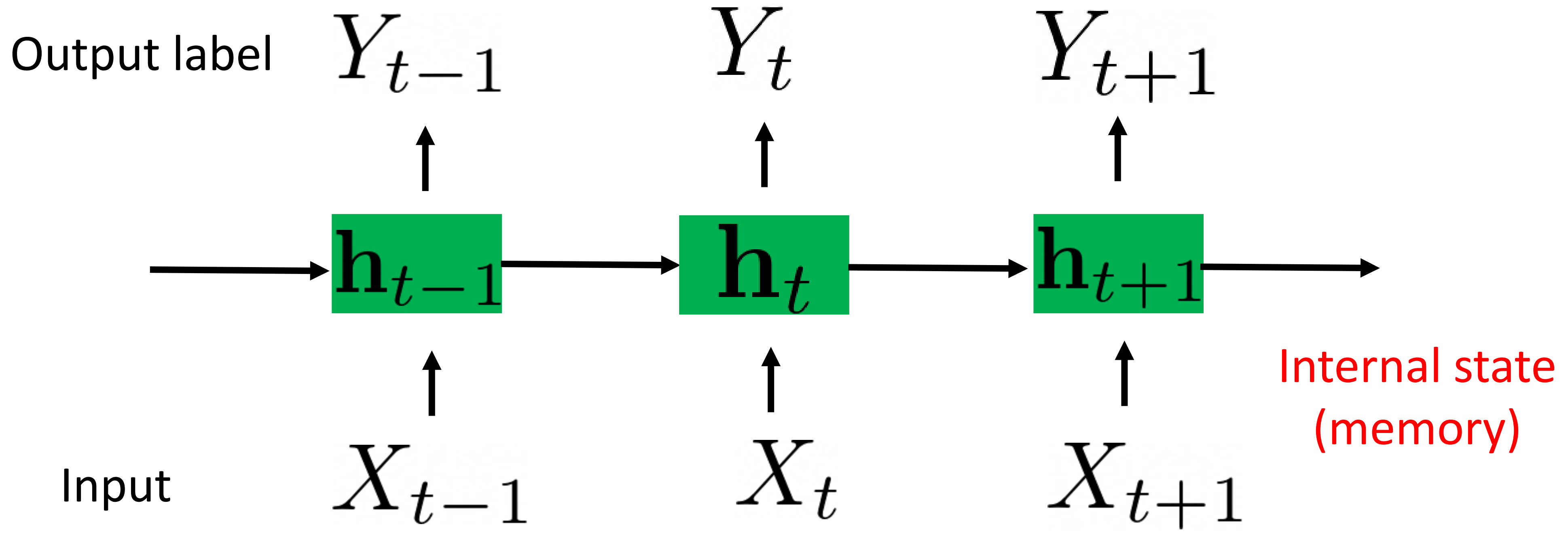
Using the Hidden State



$$\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\mathbf{y}_t = f_{W'}(\mathbf{h}_t)$$

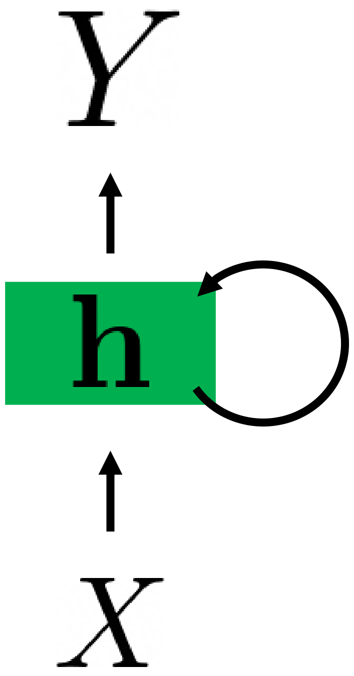
Recurrent Neural Networks



Vanilla RNN

Hidden state updating rule

RNN

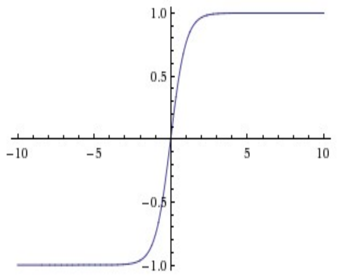


$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$$

$m \times 1$ $\quad m \times m \quad m \times 1 \quad m \times n \quad n \times 1$

tanh tanh(x)

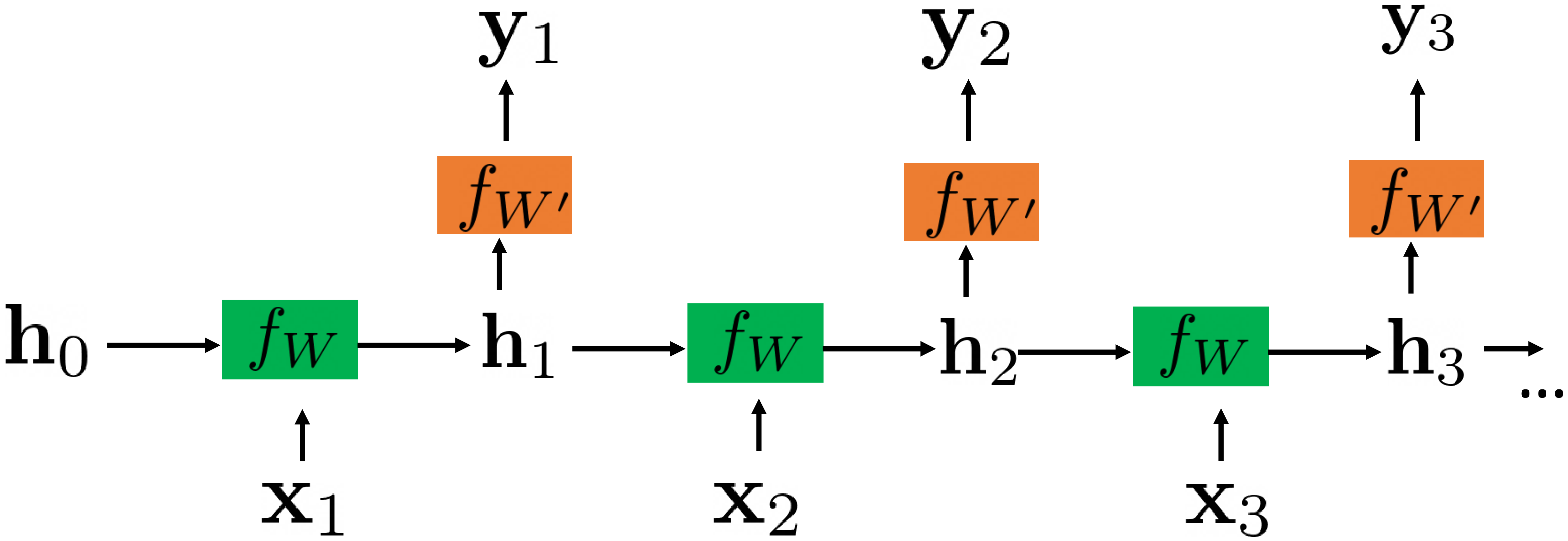
$$\frac{e^{2x}-1}{e^{2x}+1}$$



$$\mathbf{y}_t = W_{hy}\mathbf{h}_t$$

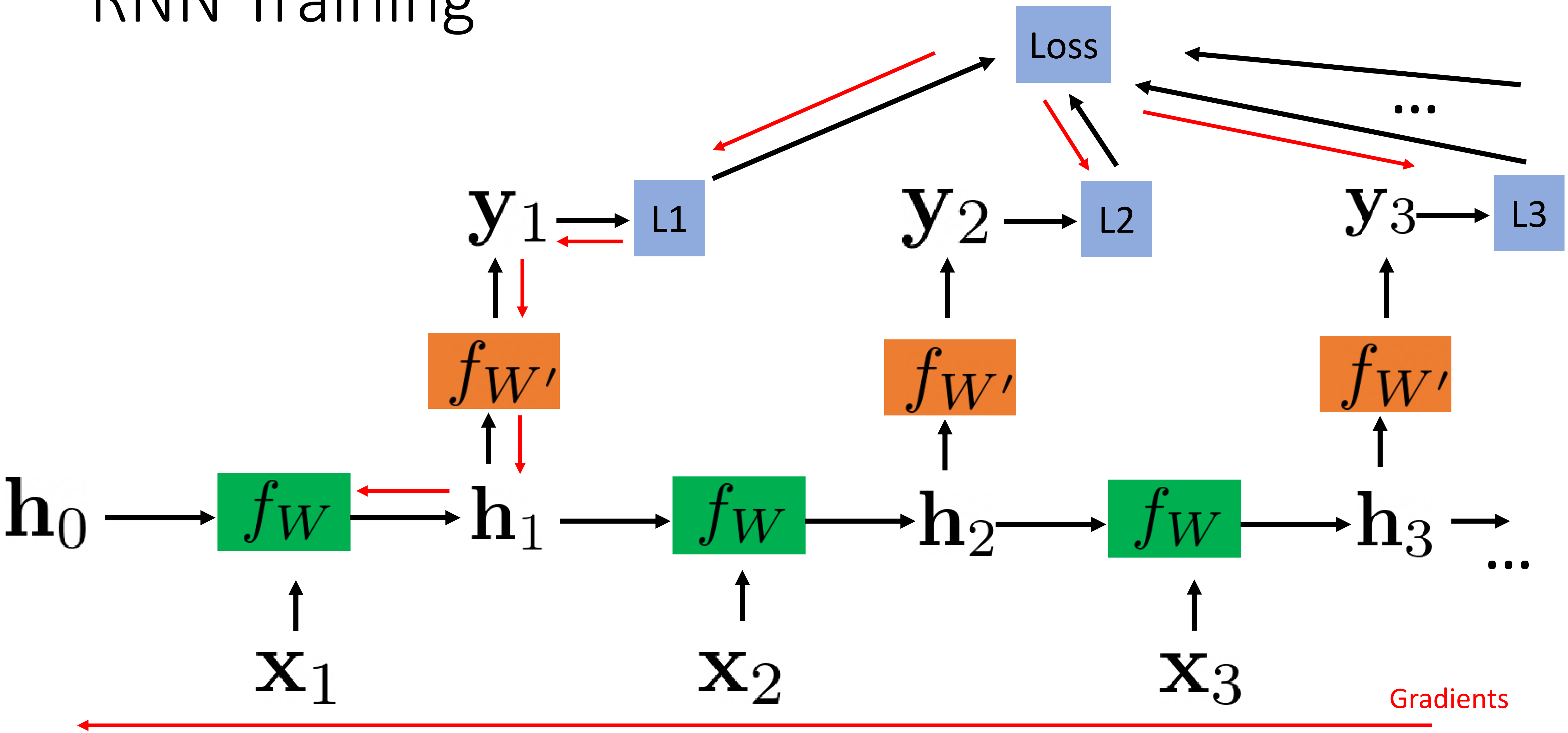
$l \times 1 \qquad m \times 1$

RNN Computation Graph

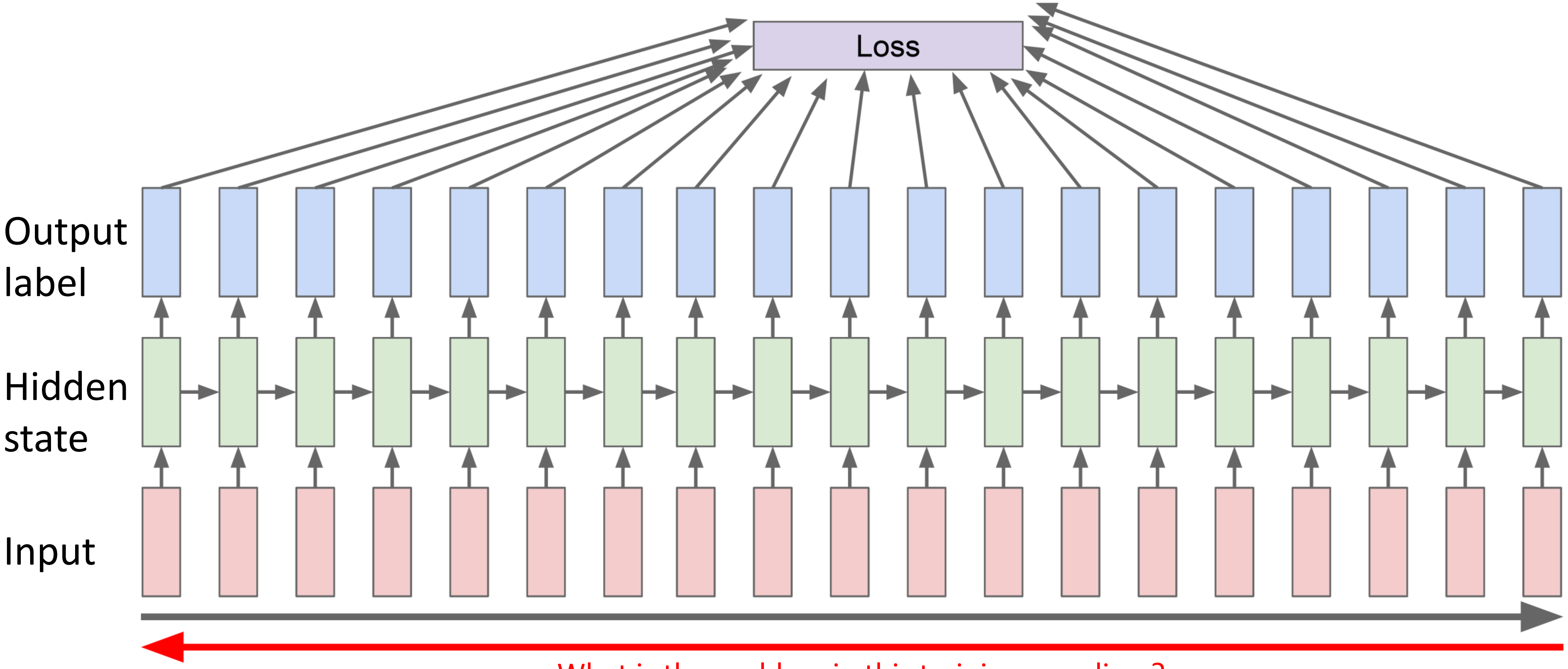


The same set of weights for different time steps f_W $f_{W'}$

RNN Training

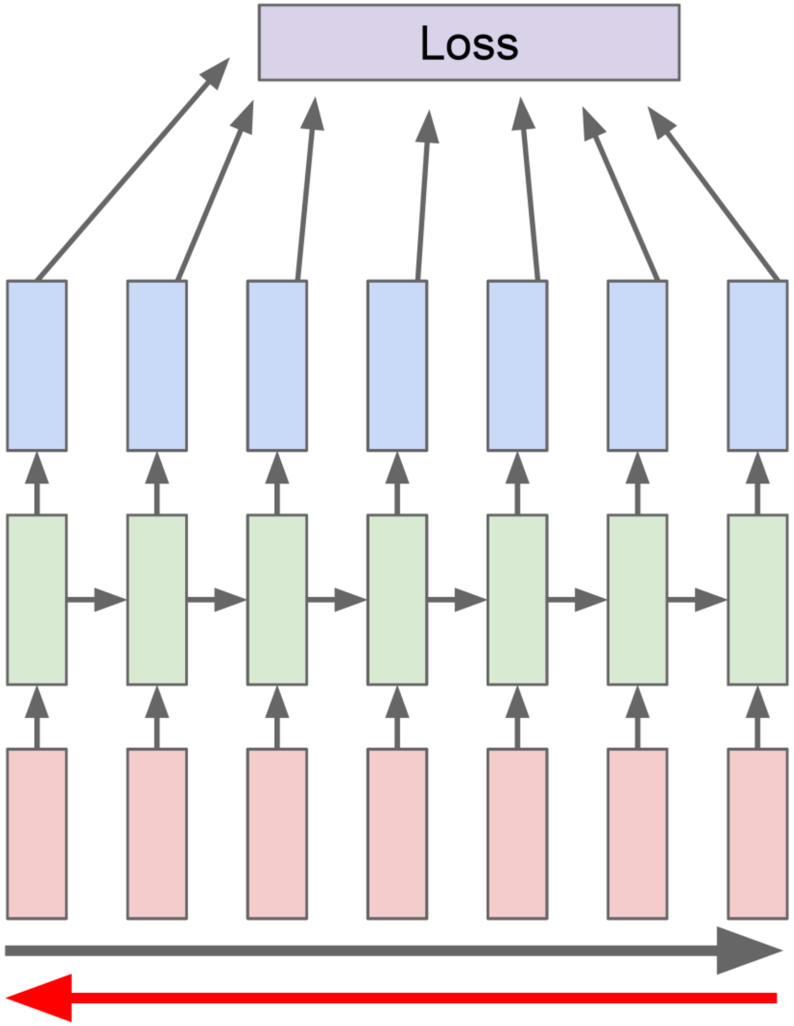


Backpropagation through Time



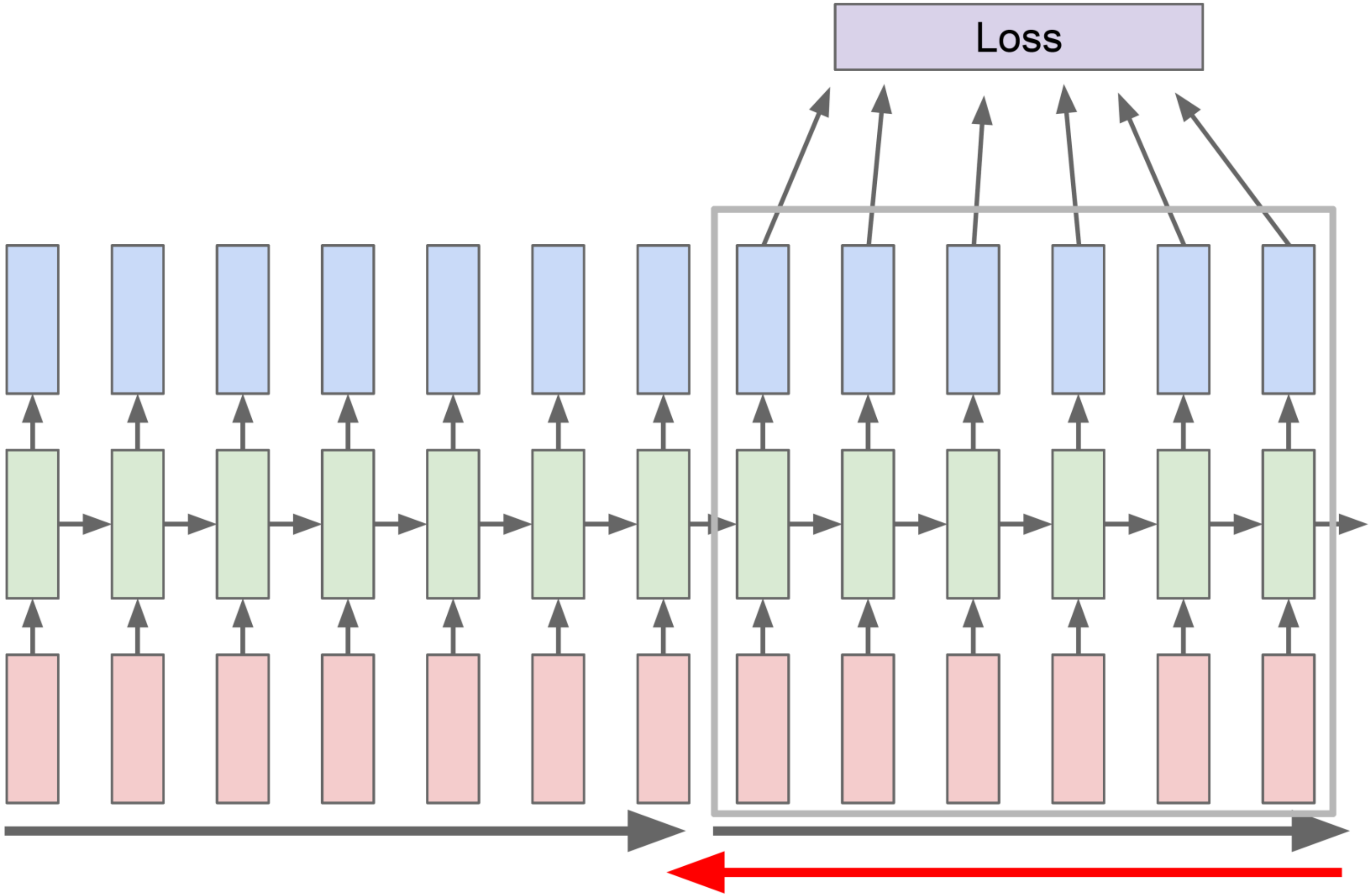
What is the problem in this training paradigm?

Truncated Backpropagation through Time



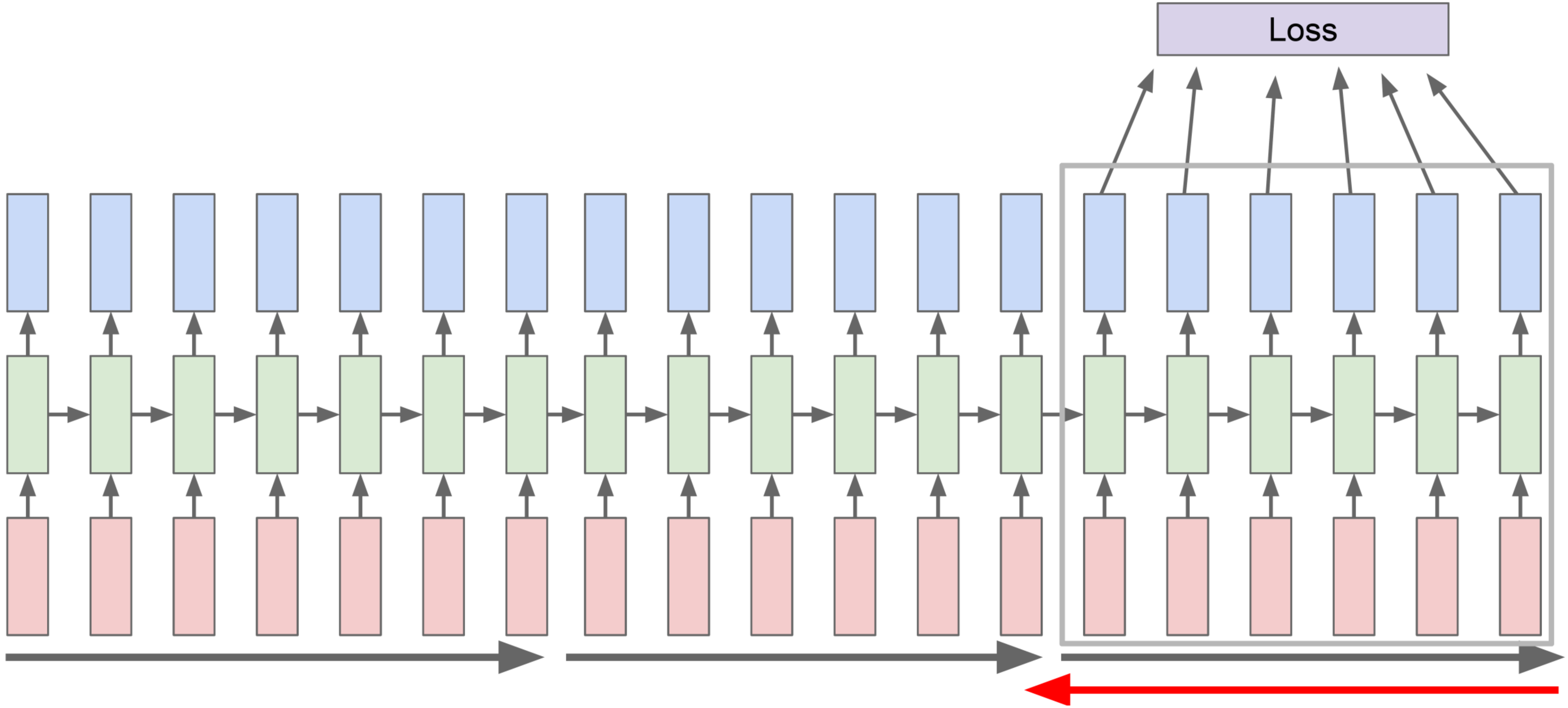
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through Time

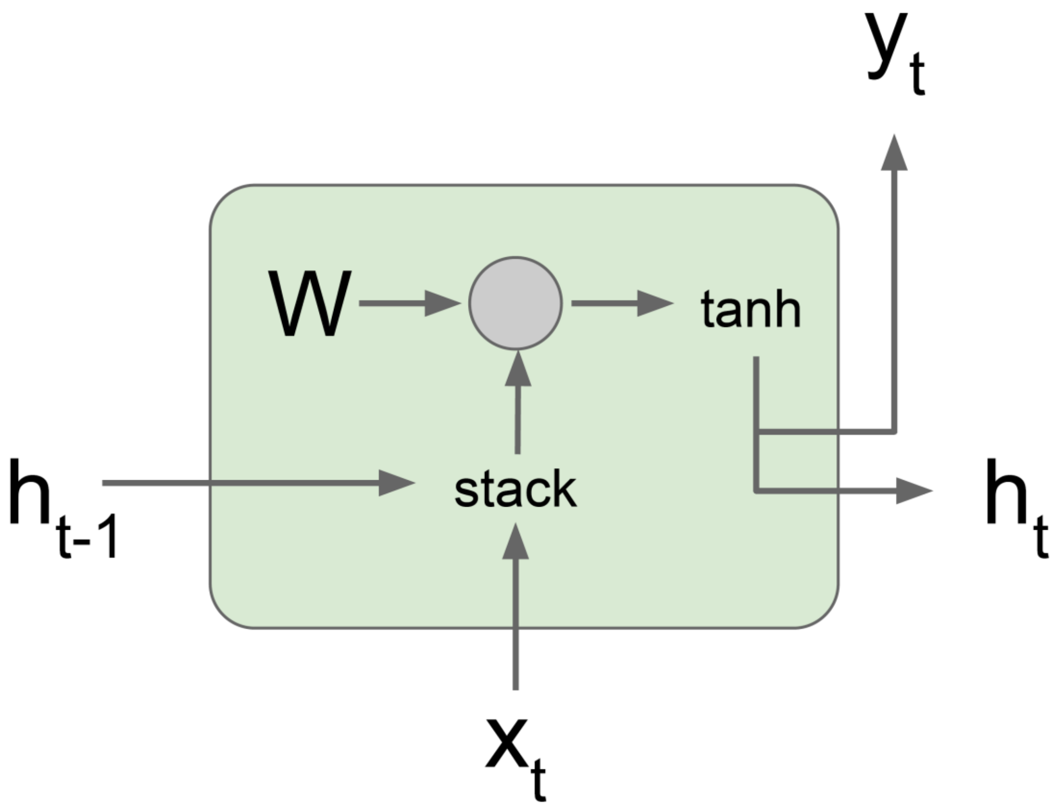


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through Time



Vanilla RNN Gradient Flow

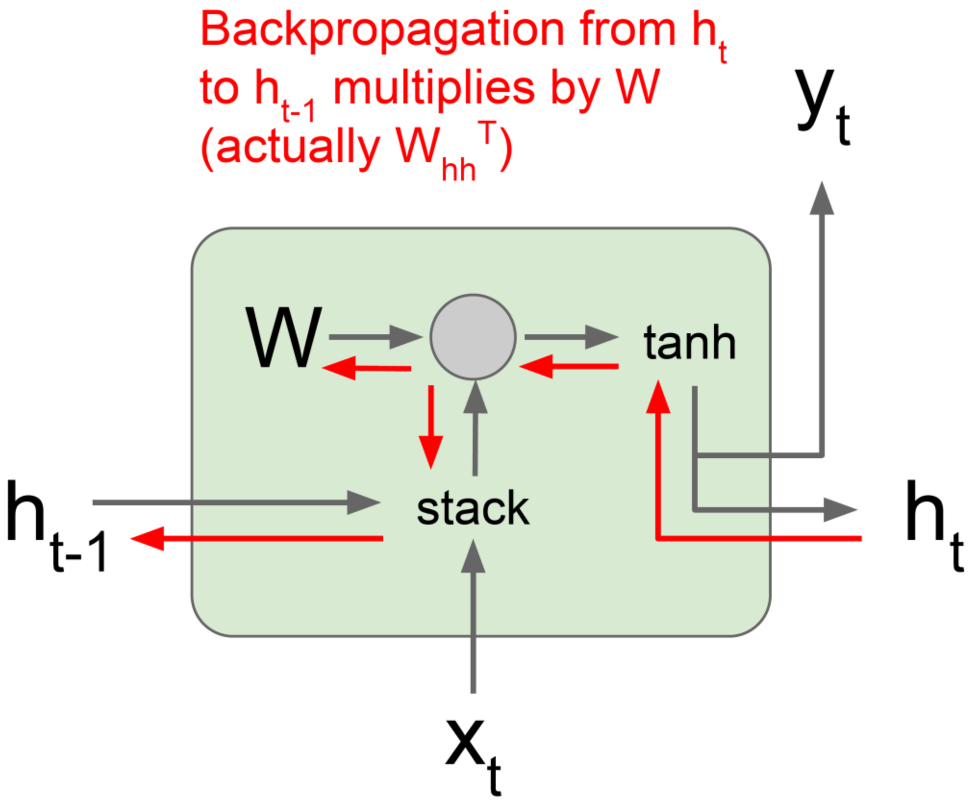


$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$$

$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

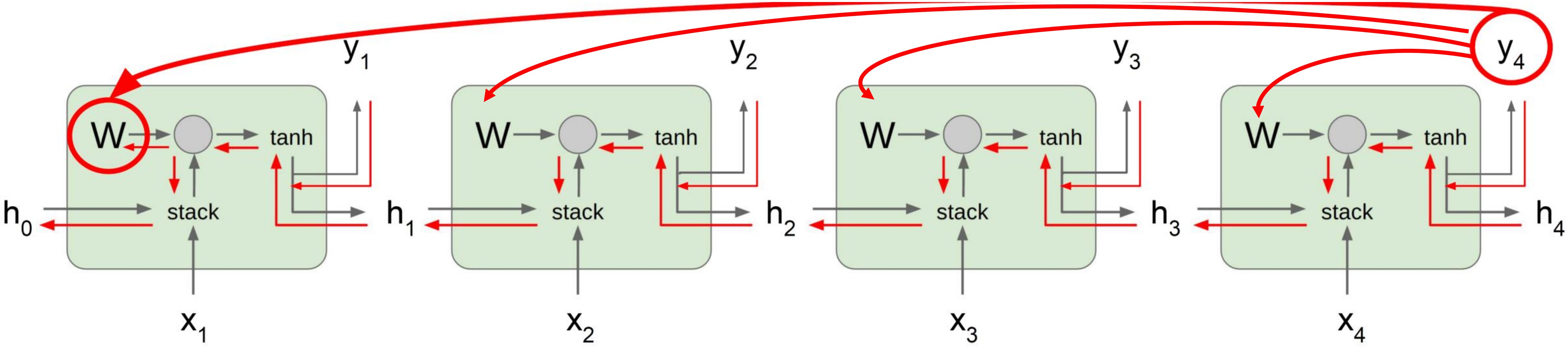
Vanilla RNN Gradient Flow



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix}\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$\frac{\partial h_t}{\partial h_{t-1}} = tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$

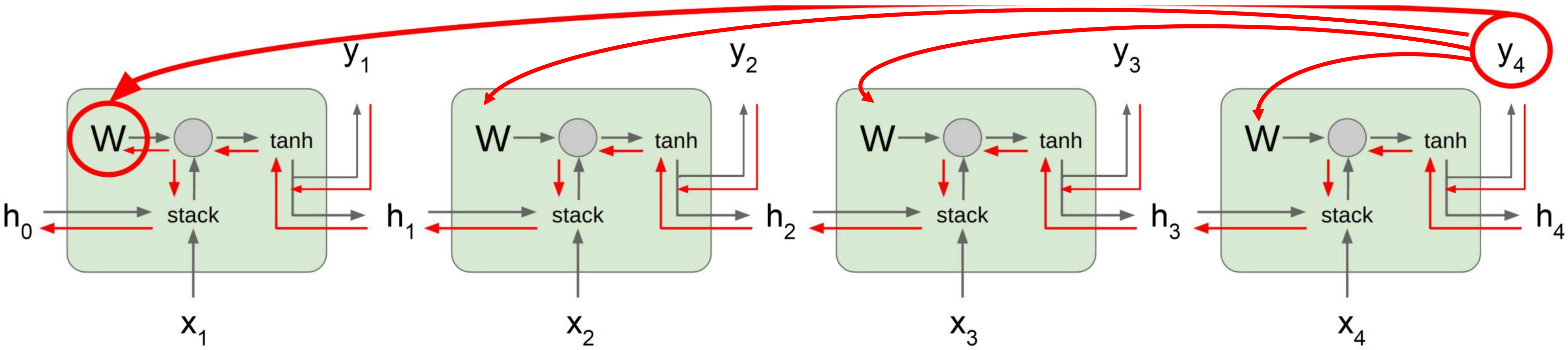
Vanilla RNN Gradient Flow



$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right) \frac{\partial h_1}{\partial W}$$

Vanilla RNN Gradient Flow



$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T}\left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right)\frac{\partial h_1}{\partial W}$$

- Vanishing gradients $\quad \left\|\frac{\partial h_t}{\partial h_{t-1}}\right\|_2 < 1$
- Exploding gradients $\quad \left\|\frac{\partial h_t}{\partial h_{t-1}}\right\|_2 > 1$

https://en.wikipedia.org/wiki/Matrix_norm

Vanilla RNN Gradient Flow

- Exploding gradients $\|\frac{\partial h_t}{\partial h_{t-1}}\|_2 > 1$
  - Gradient clipping

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

- Vanishing gradients $\|\frac{\partial h_t}{\partial h_{t-1}}\|_2 < 1$
  - Change RNN architecture

Summary

- RNNs can be used for sequential data to capture dependencies in time
- LSTMs and GRUs are better then vanilla RNNs
- It is difficult to capture long-term dependencies in RNNs
- Use transformers (in future lectures)

Further Reading

- Stanford CS231n, lecture 10, Recurrent Neural Networks http://cs231n.stanford.edu/
- Long Short Term Memory https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
- Gated Recurrent Units https://arxiv.org/pdf/1412.3555.pdf