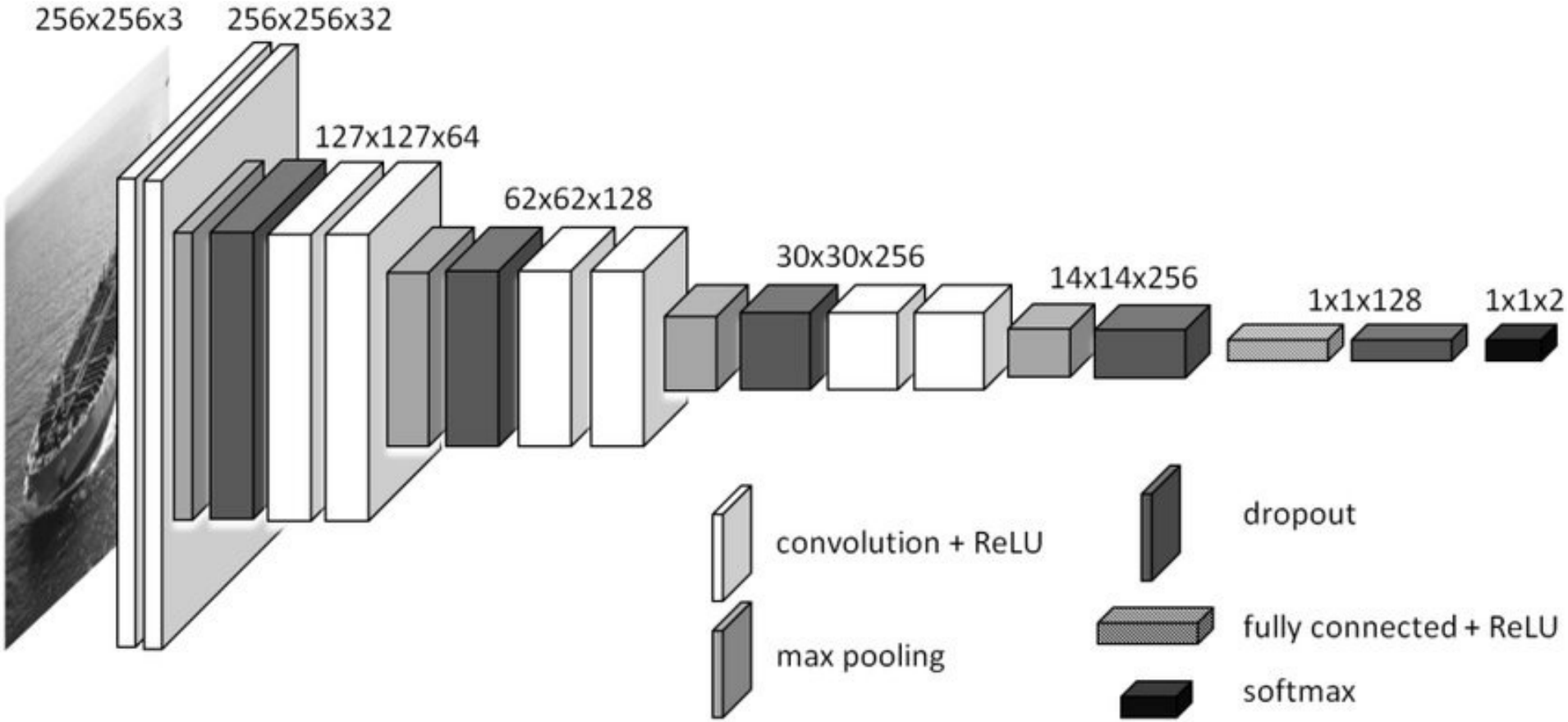
CSCE 633: Machine Learning

Lecture 30: Recurrent Neural Networks

Texas A&M University
Bobak Mortazavi
Zhale Nowroozi

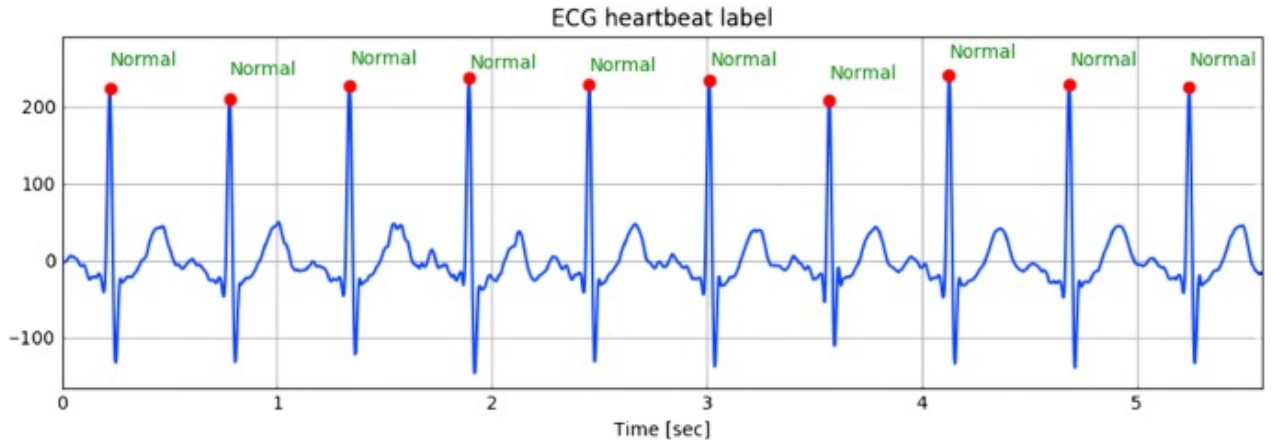
Last Time: Convolutional Neural Networks



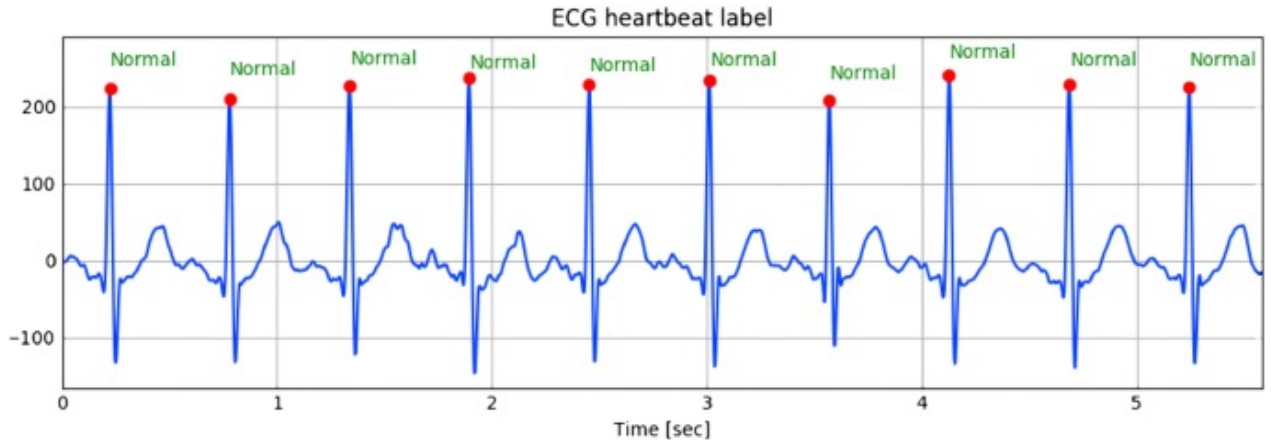
Milicevic, M., Zubrinic, K., Obradovic, I., & Sjekavica, T. (2019). Application of transfer learning for fine-grained vessel classification using a limited dataset. In *Applied Physics, System Science and Computers III: Proceedings of the 3rd International Conference on Applied Physics, System Science and Computers (APSAC2018), September 26-28, 2018, Dubrovnik, Croatia* (pp. 125-131). Springer International Publishing.

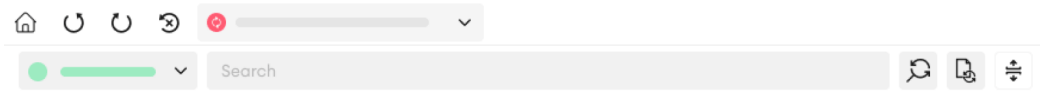
Time-series Data

Time-series Data



Time-series Data

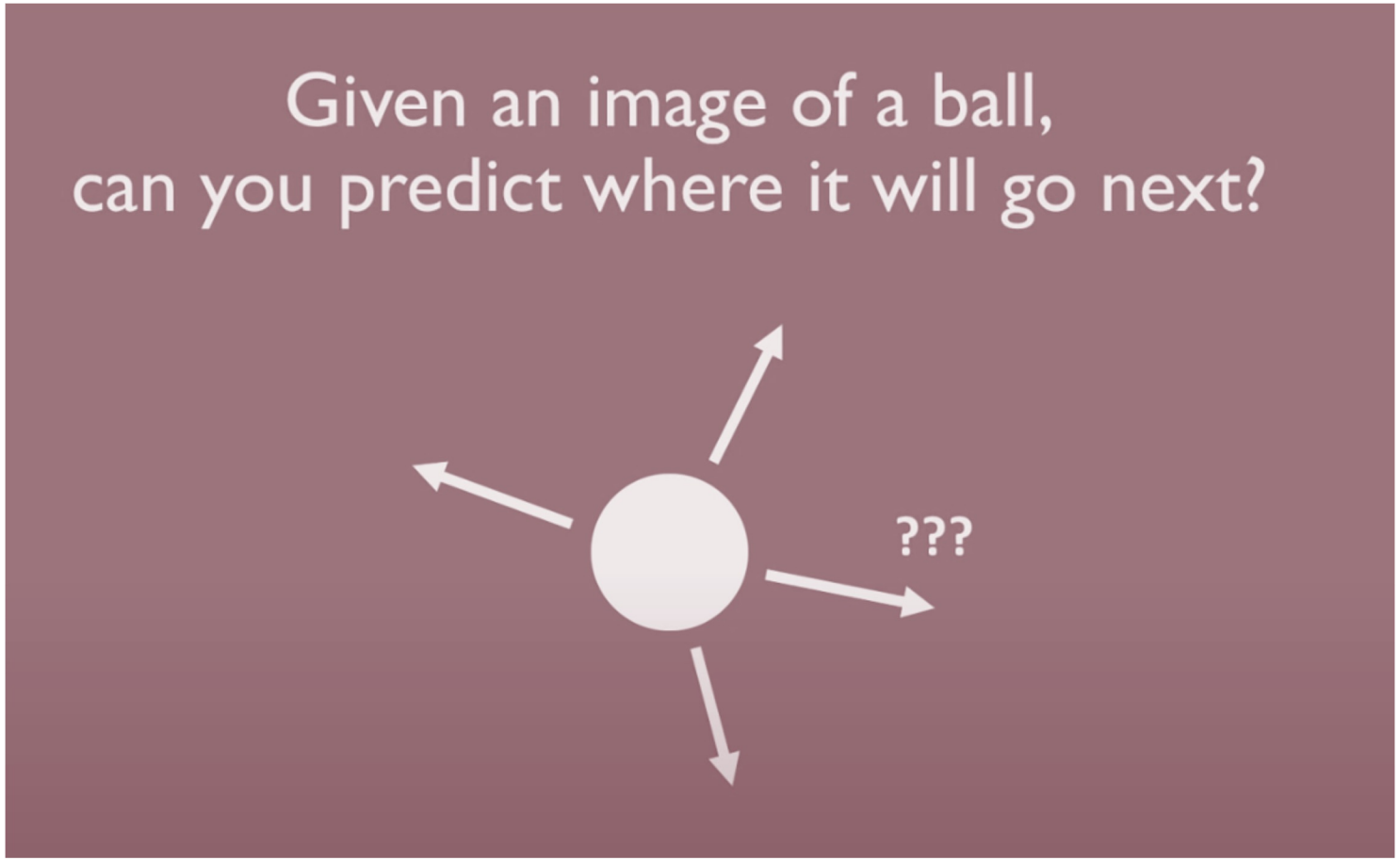




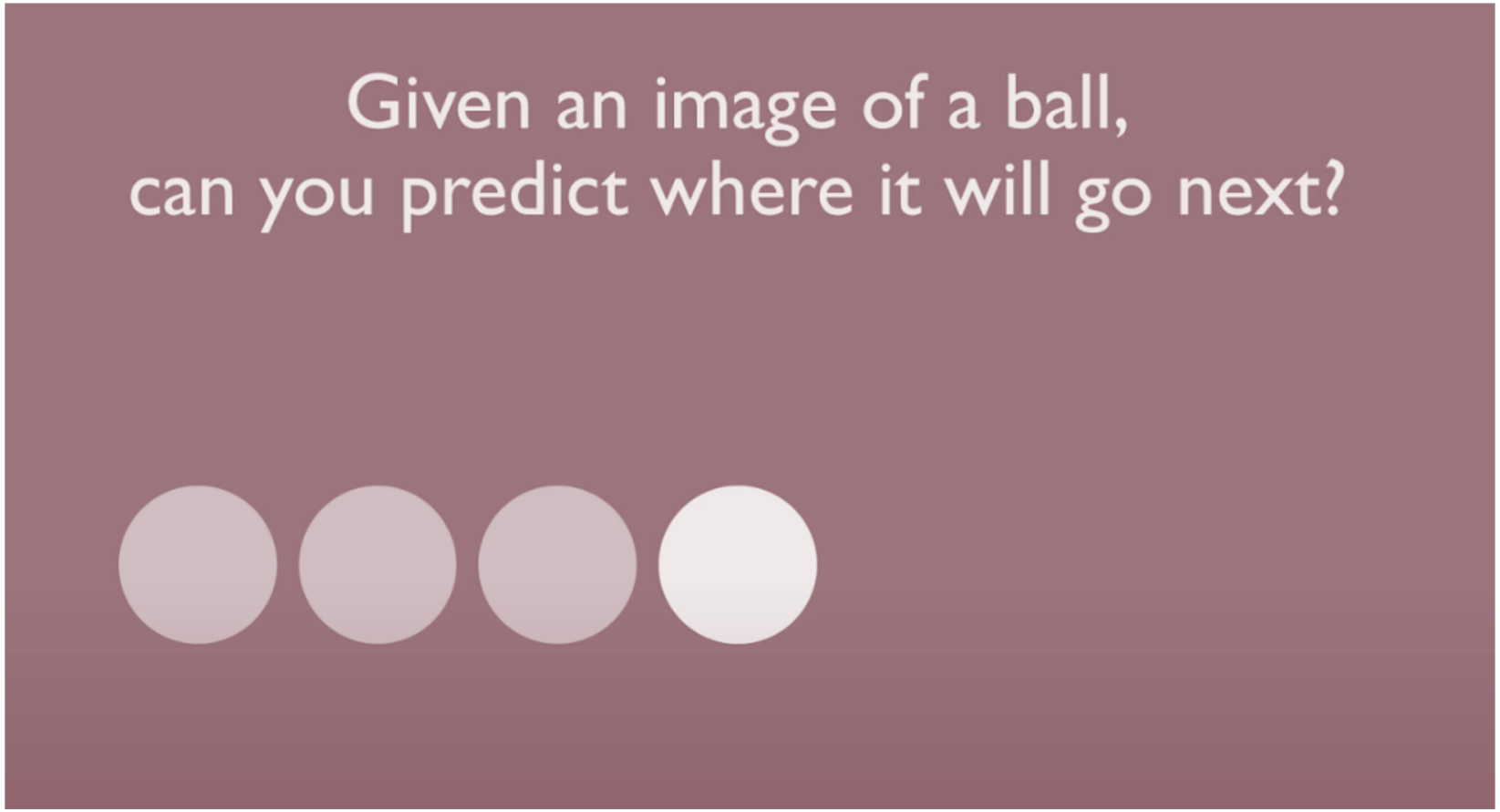
Despite the rapid advances in AI, computer vision (CV) is still challenged in matching the precision of human perception. The training data here is as important as algorithms. The more accurate the input data annotation, the more effective the model prediction.

How do we annotate data, though? There are multiple ways to go with this one, but it all depends on your use case. For the purposes of this article, we'll take a deeper dive into bounding boxes as one of the most extensively used annotation techniques. Moving forward, we'll walk you through the following:

Why RNN?



Why RNN?



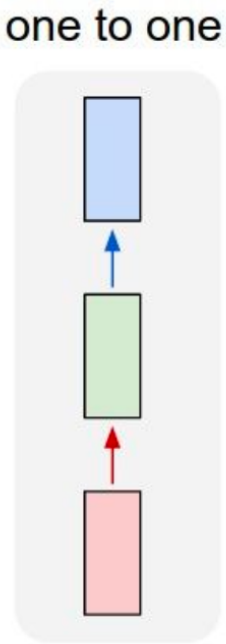
Why RNN?

Sequence Modeling

To model sequences, we need to:

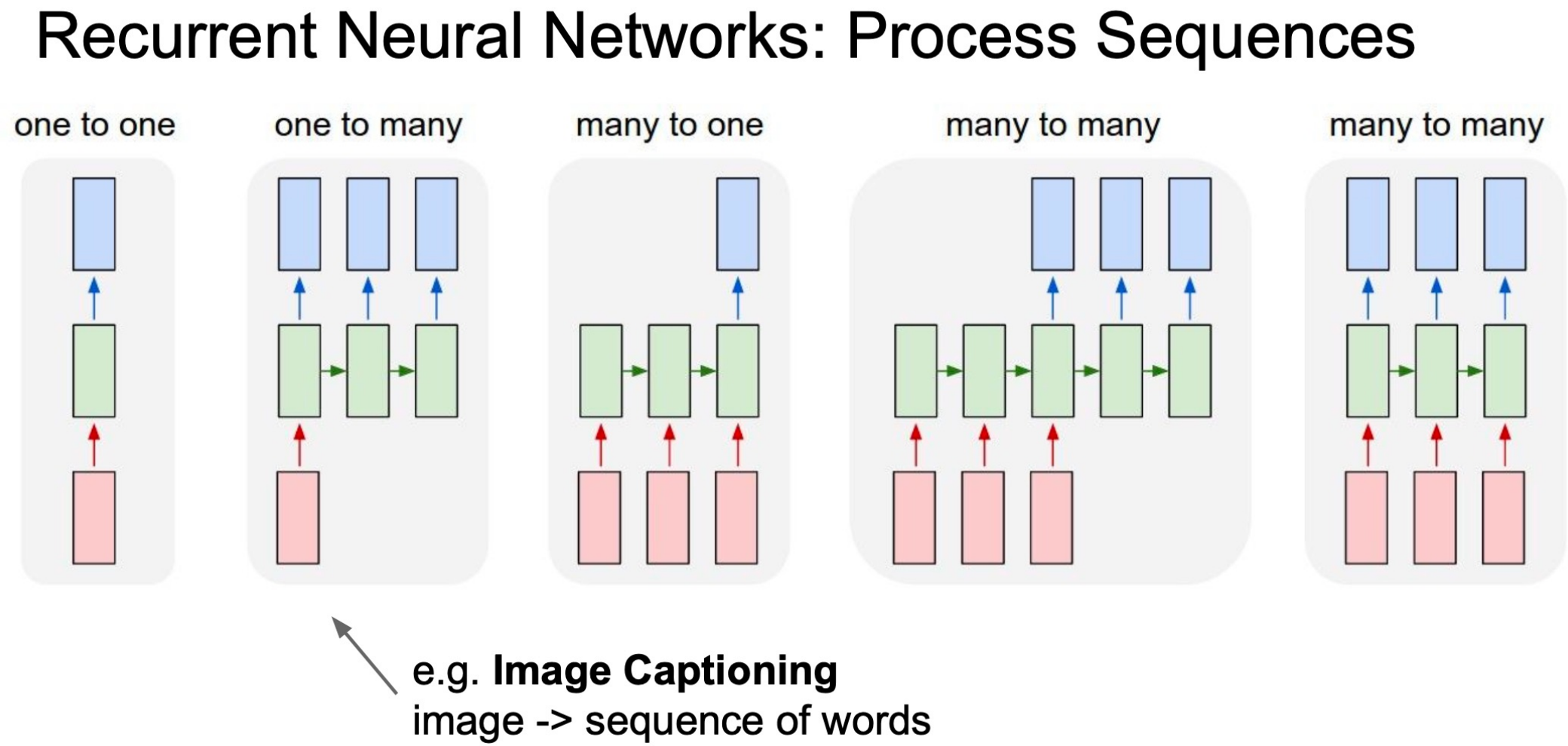
1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence

“Vanilla” Neural Network



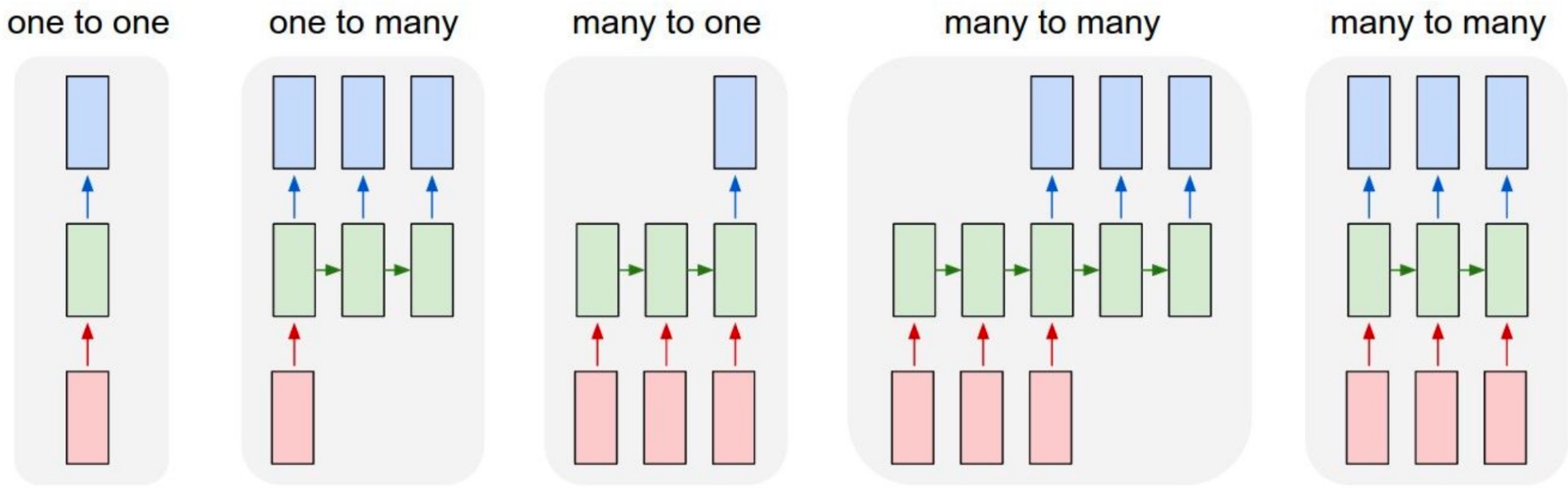
Vanilla Neural Networks

“Vanilla” Neural Network vs RNN



“Vanilla” Neural Network vs RNN

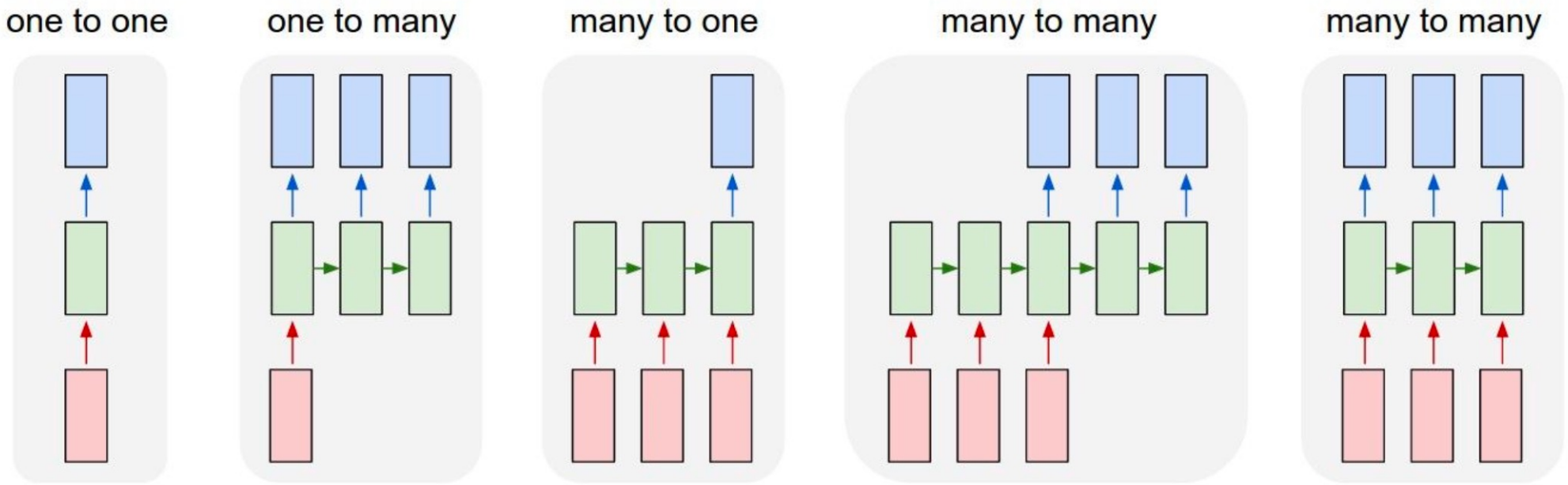
Recurrent Neural Networks: Process Sequences



e.g. **Sentiment Classification**
sequence of words -> sentiment

“Vanilla” Neural Network vs RNN

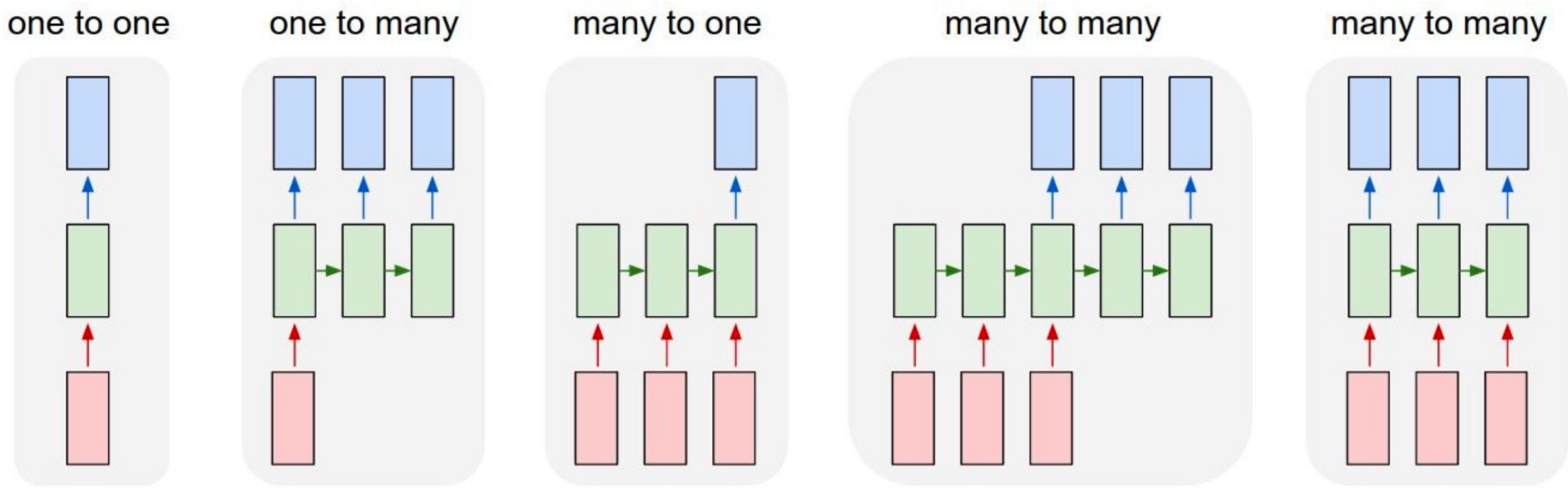
Recurrent Neural Networks: Process Sequences



e.g. **Machine Translation**
seq of words -> seq of words

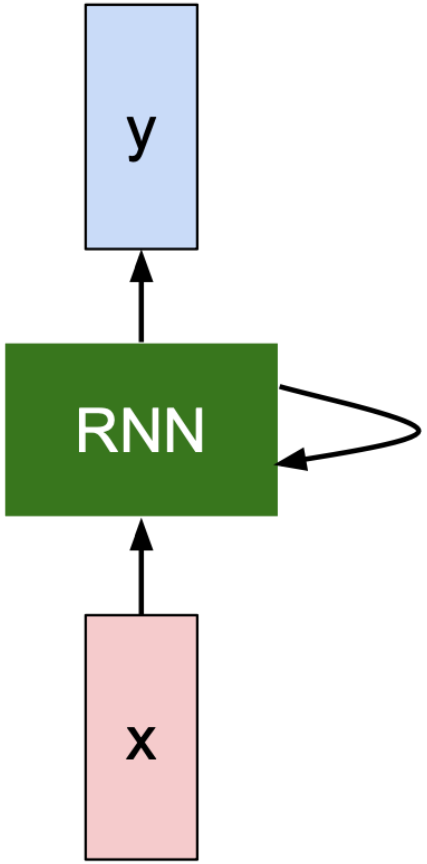
“Vanilla” Neural Network vs RNN

Recurrent Neural Networks: Process Sequences

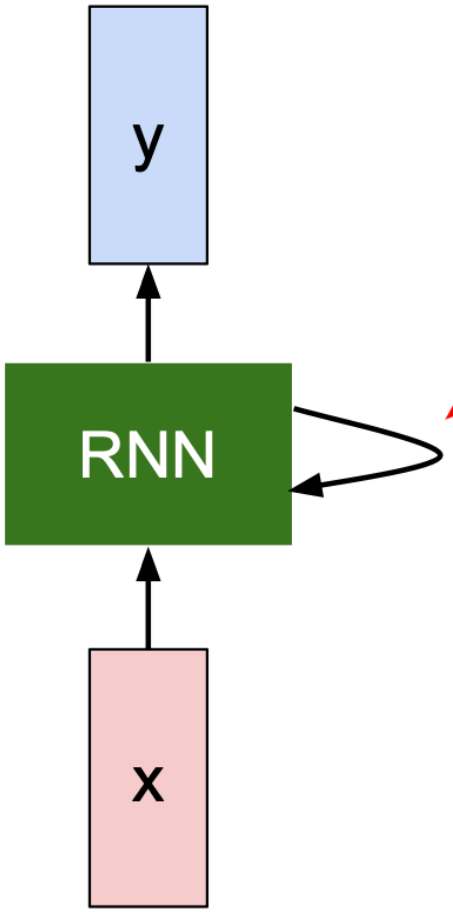


e.g. **Video classification on frame level**

Recurrent Neural Network



Recurrent Neural Network



Key idea: RNNs have an "internal state" that is updated as a sequence is processed

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

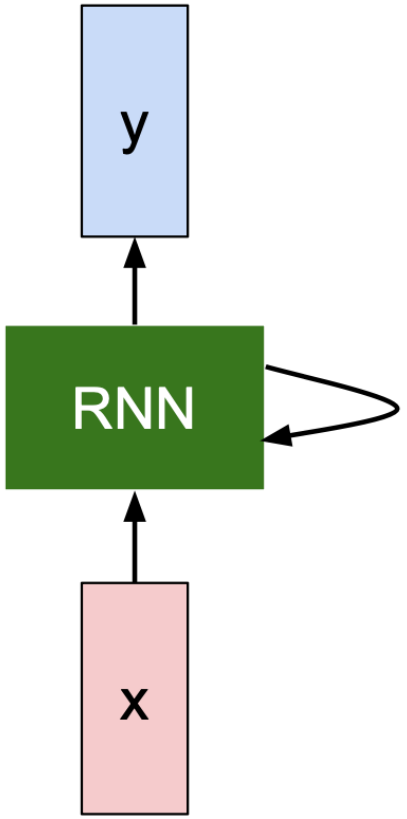
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W

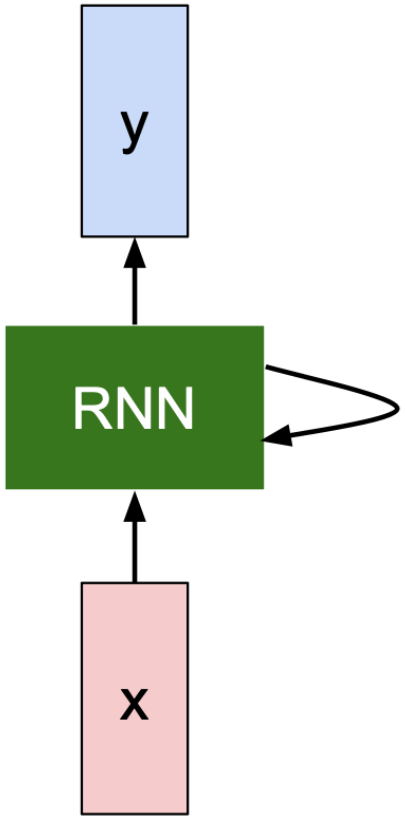


Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



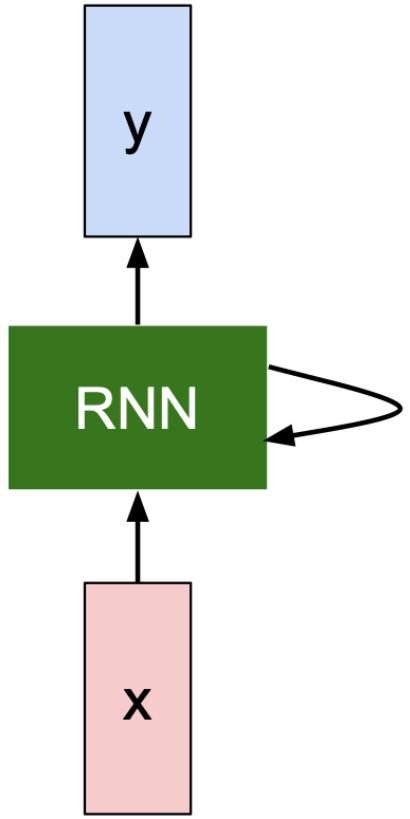
Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

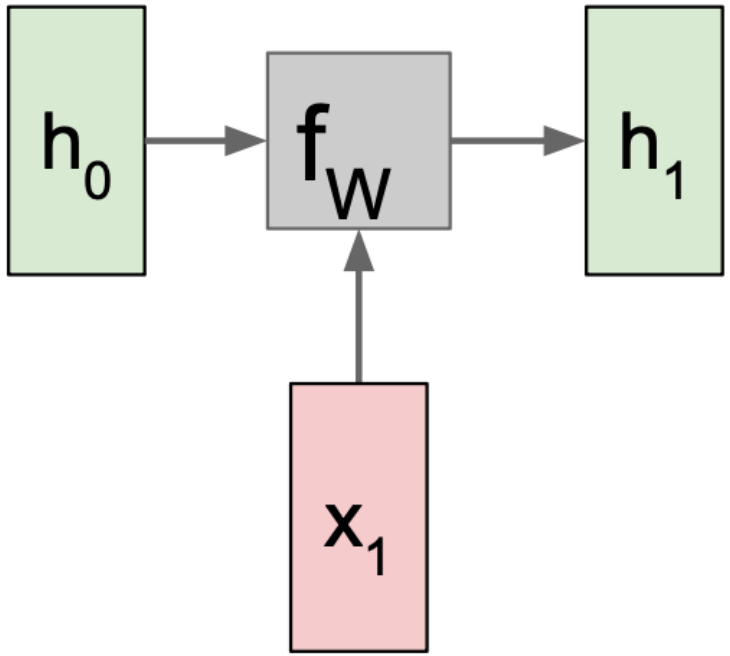
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

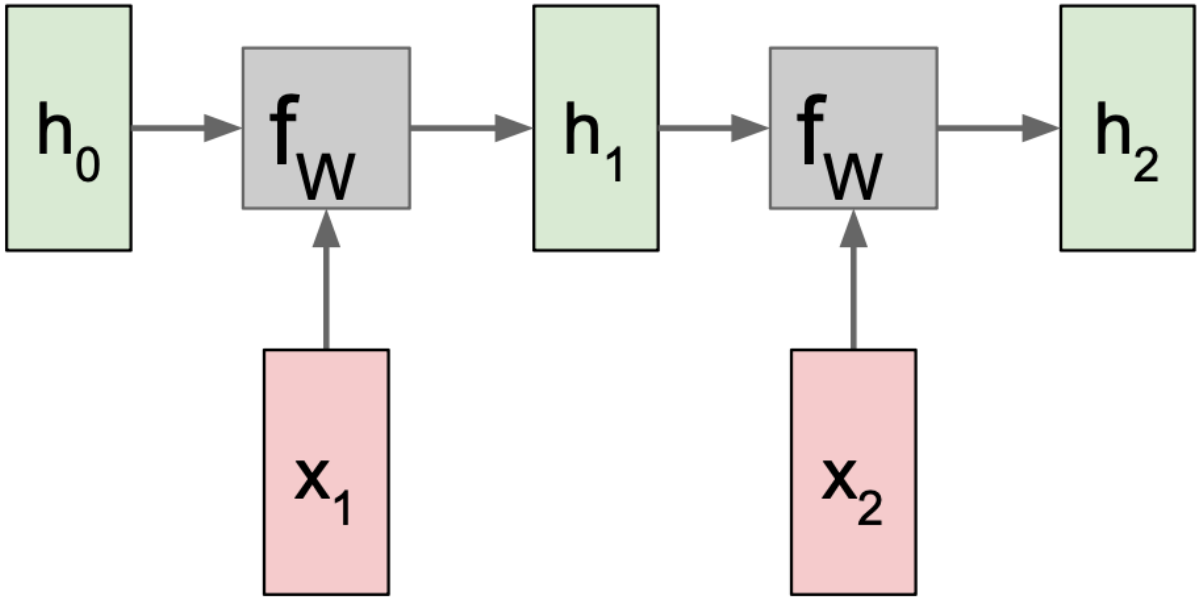


Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

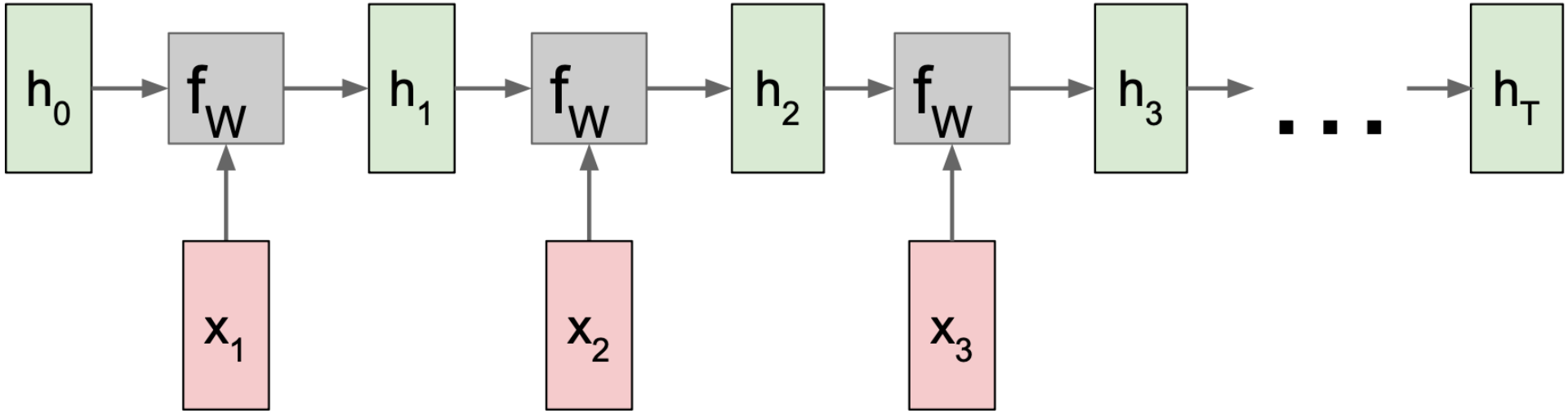
Recurrent Neural Network: Computational Graph



Recurrent Neural Network: Computational Graph

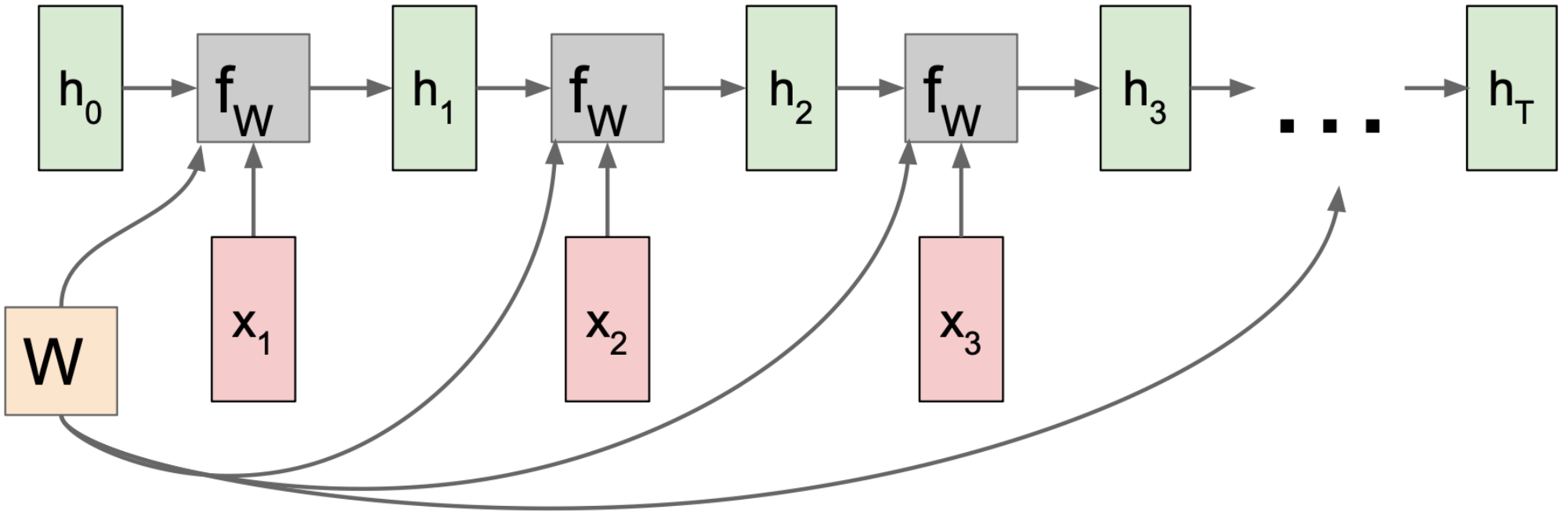


Recurrent Neural Network: Computational Graph

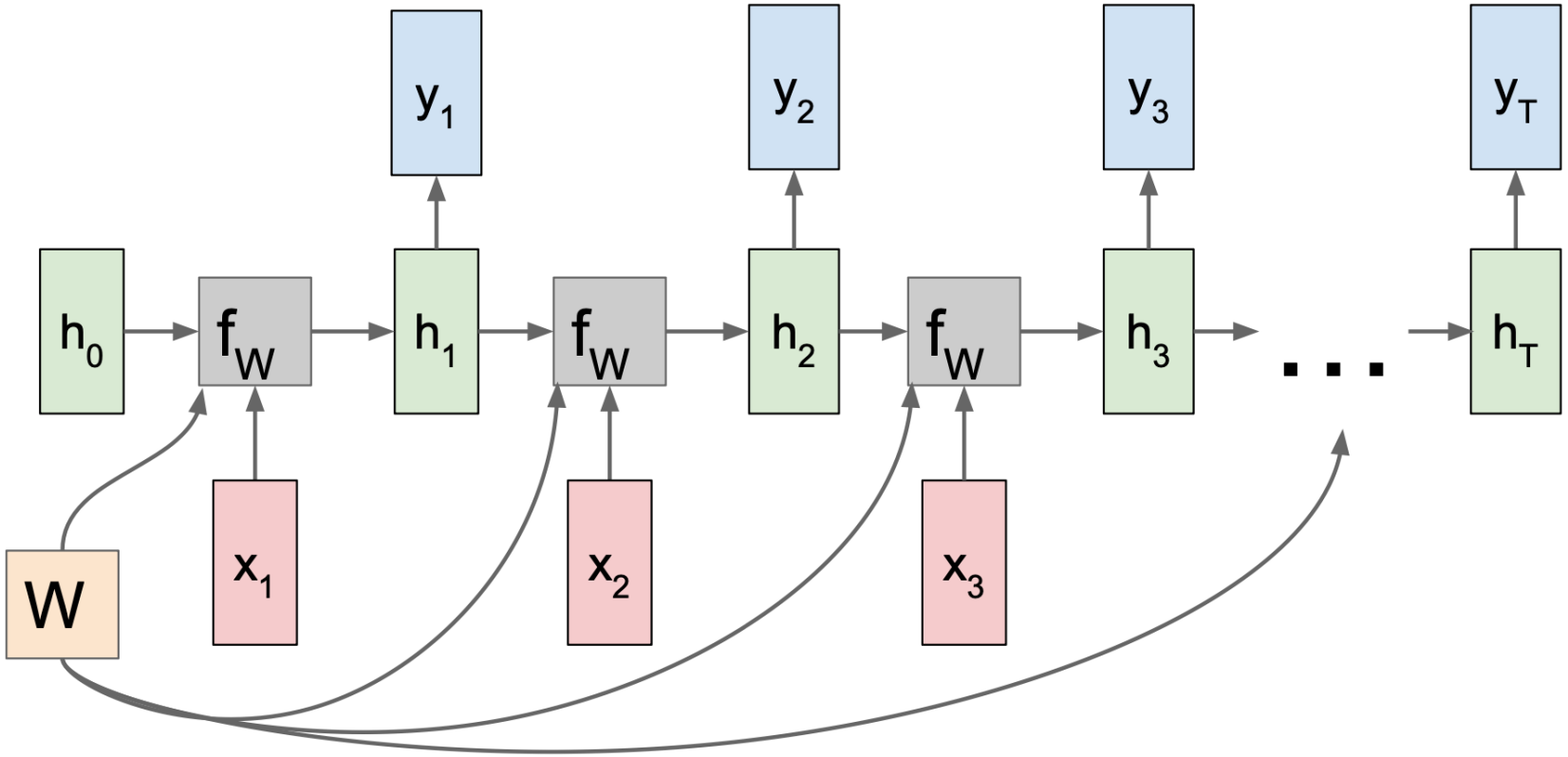


Recurrent Neural Network: Computational Graph

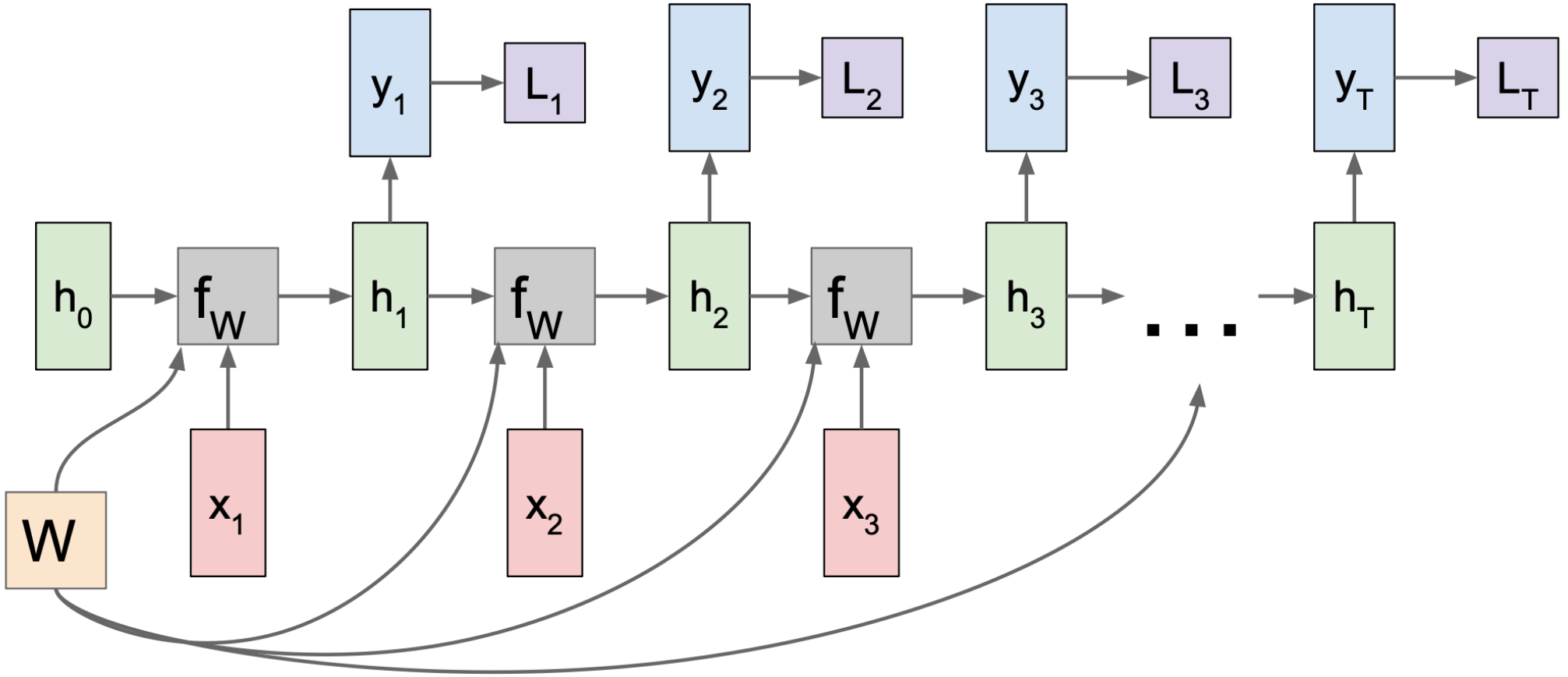
Re-use the same weight matrix at every time-step



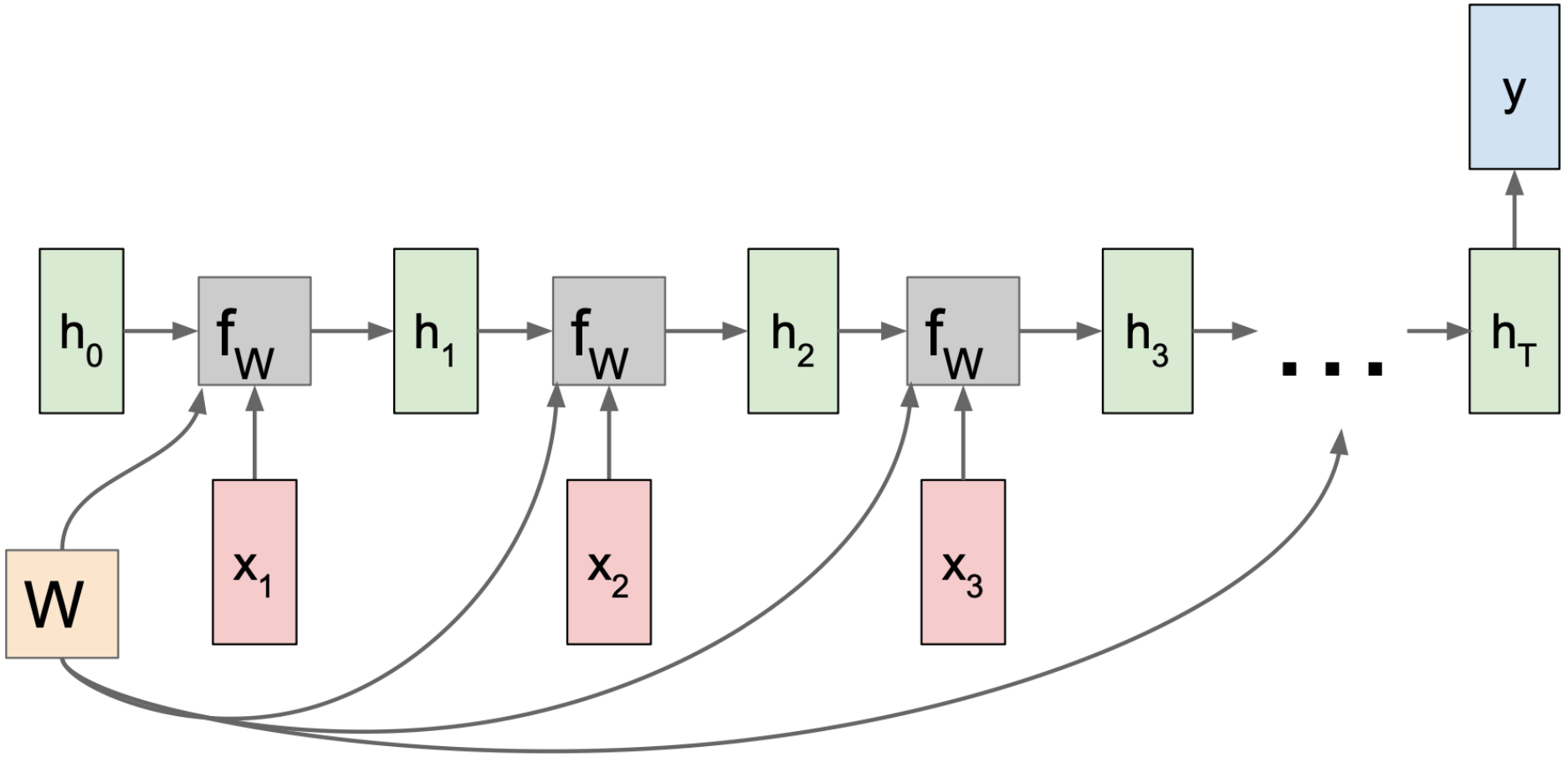
Recurrent Neural Network: Computational Graph: Many to Many



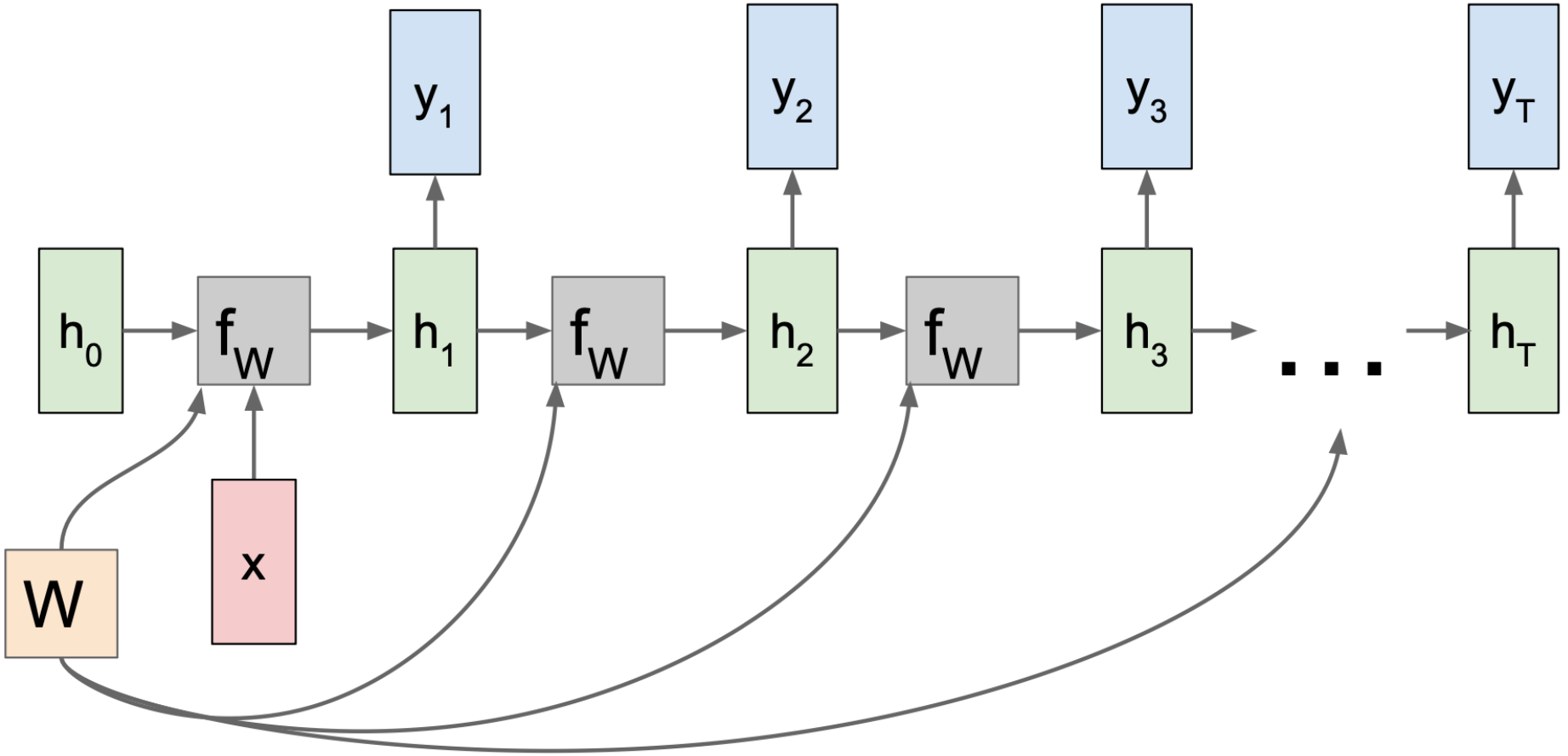
Recurrent Neural Network: Computational Graph: Many to Many



Recurrent Neural Network: Computational Graph: Many to One



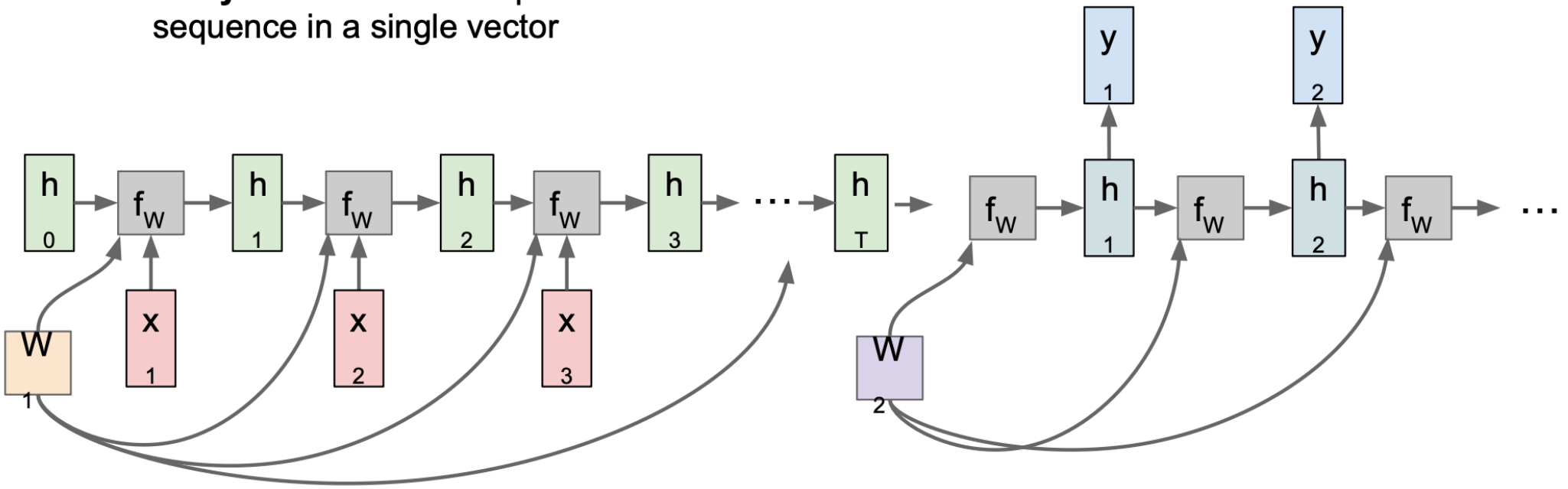
Recurrent Neural Network: Computational Graph: One to Many



Recurrent Neural Network: Computational Graph: Many to one + one to many

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector



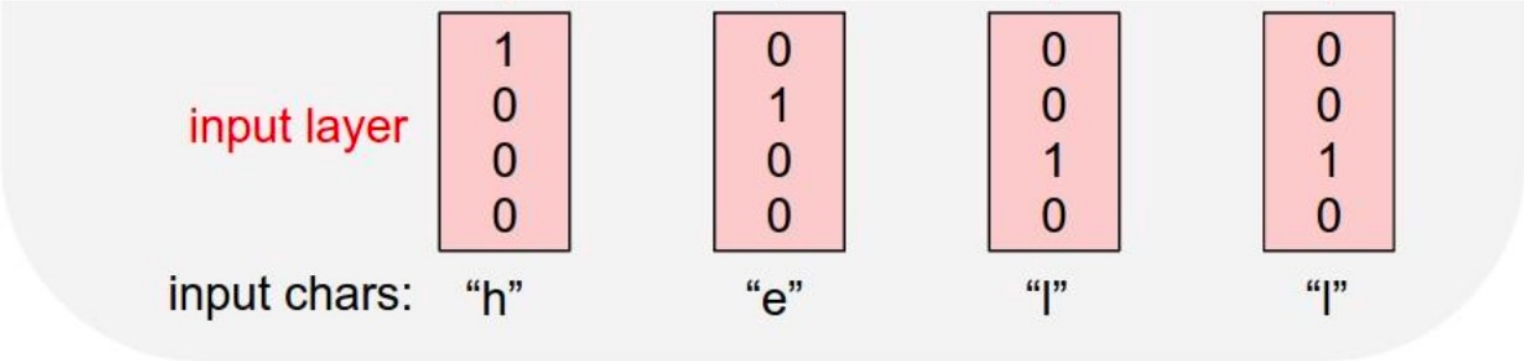
Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

Recurrent Neural Network

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



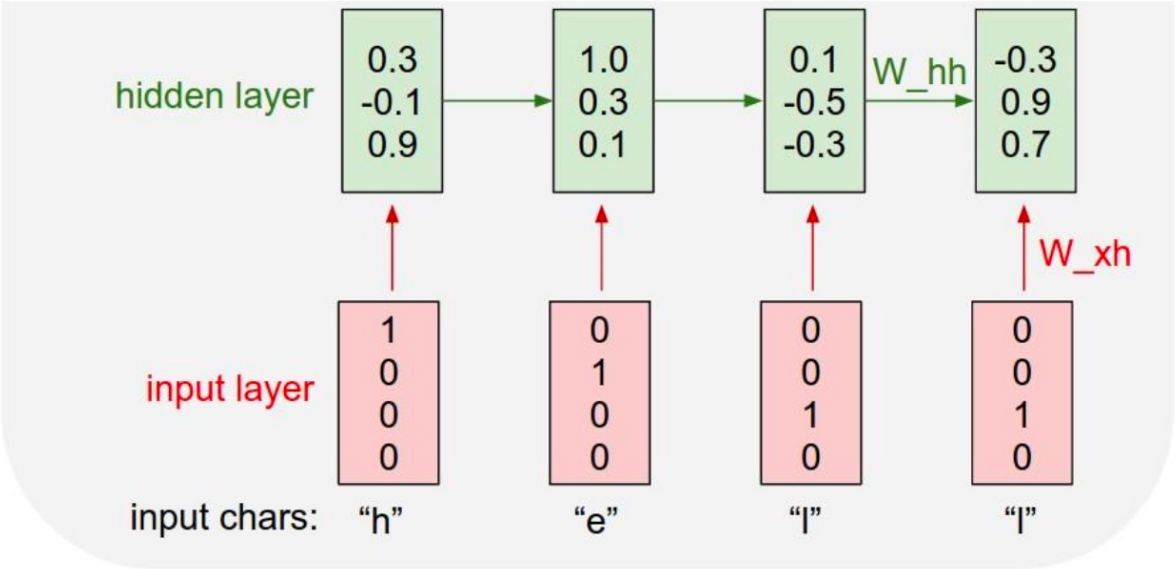
Recurrent Neural Network

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

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

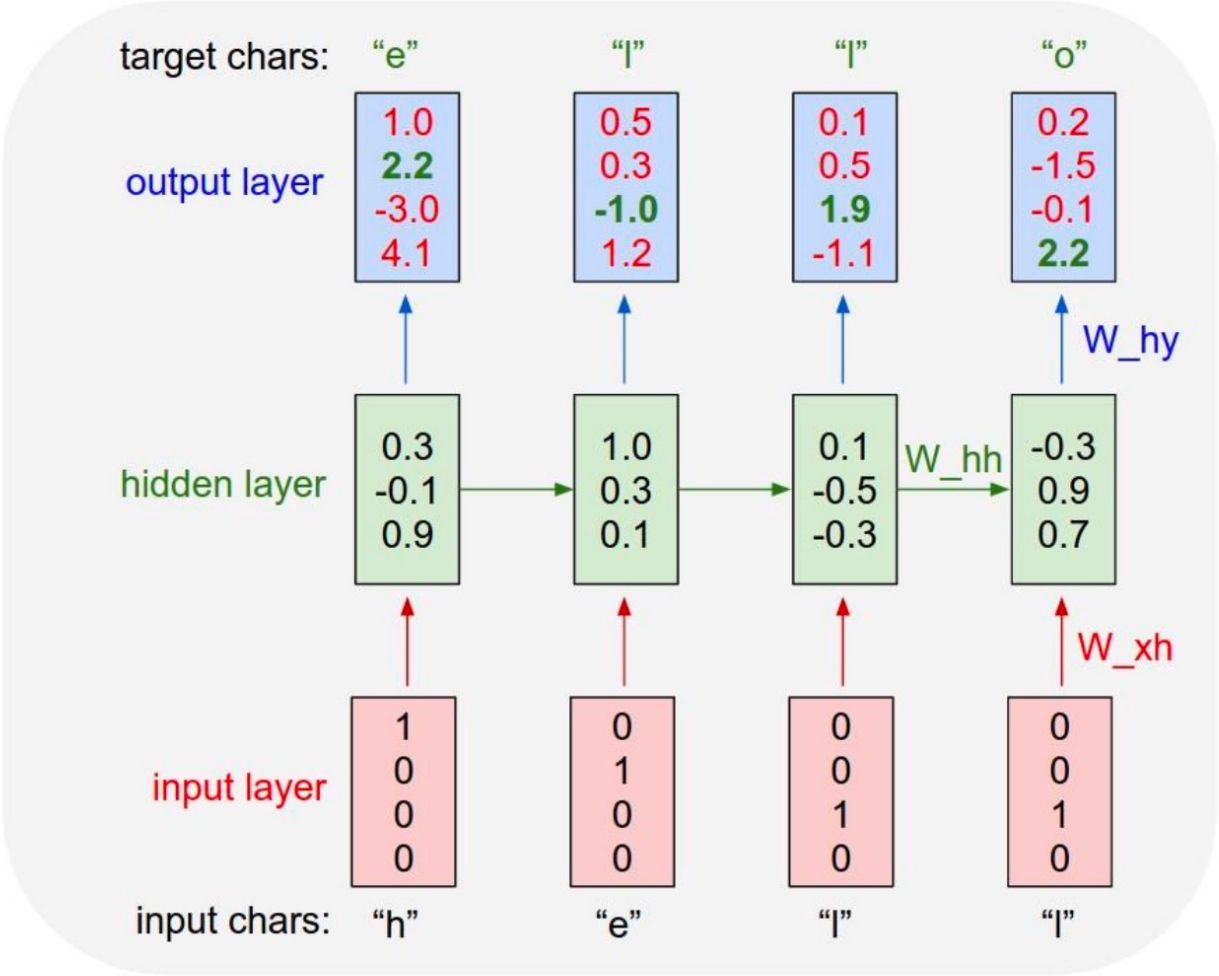


Recurrent Neural Network

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

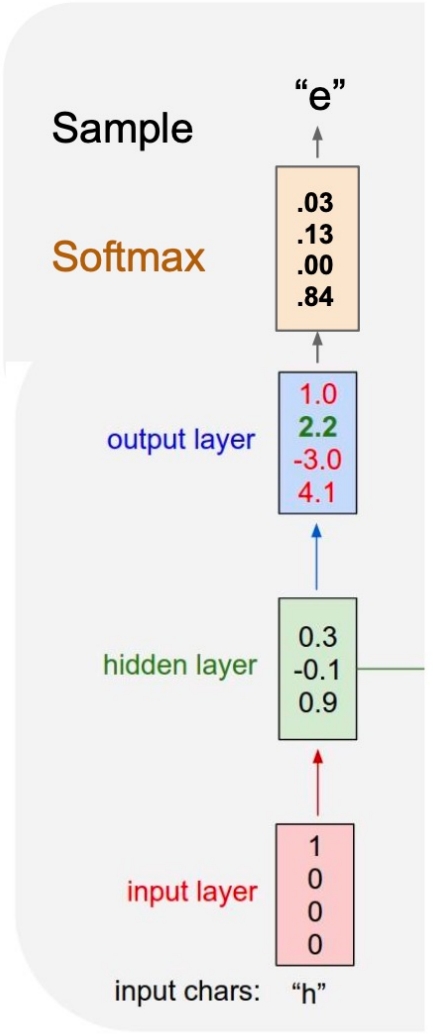


Recurrent Neural Network

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

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

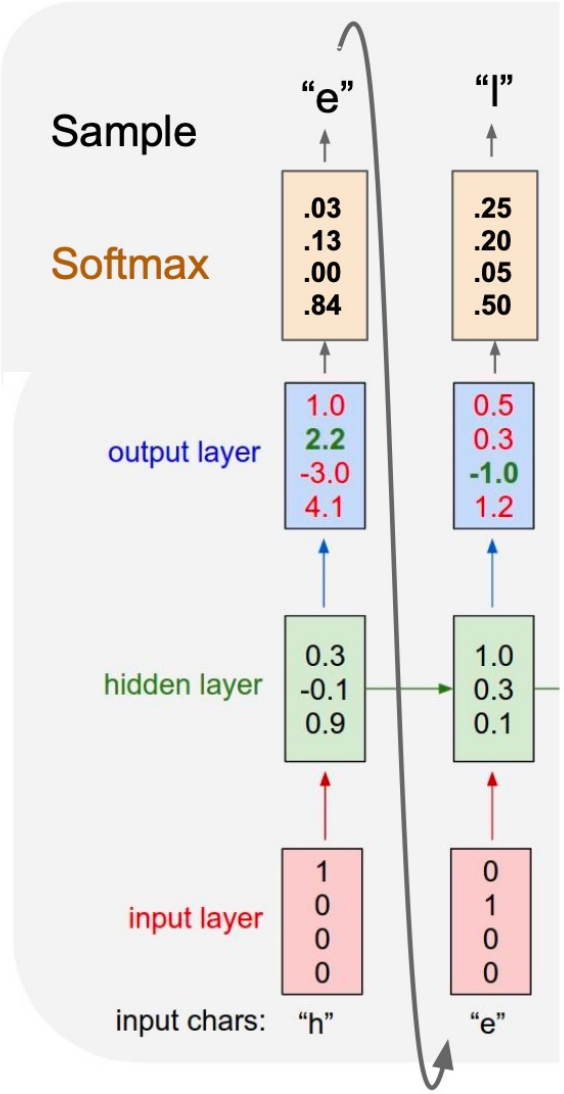


Recurrent Neural Network

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

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

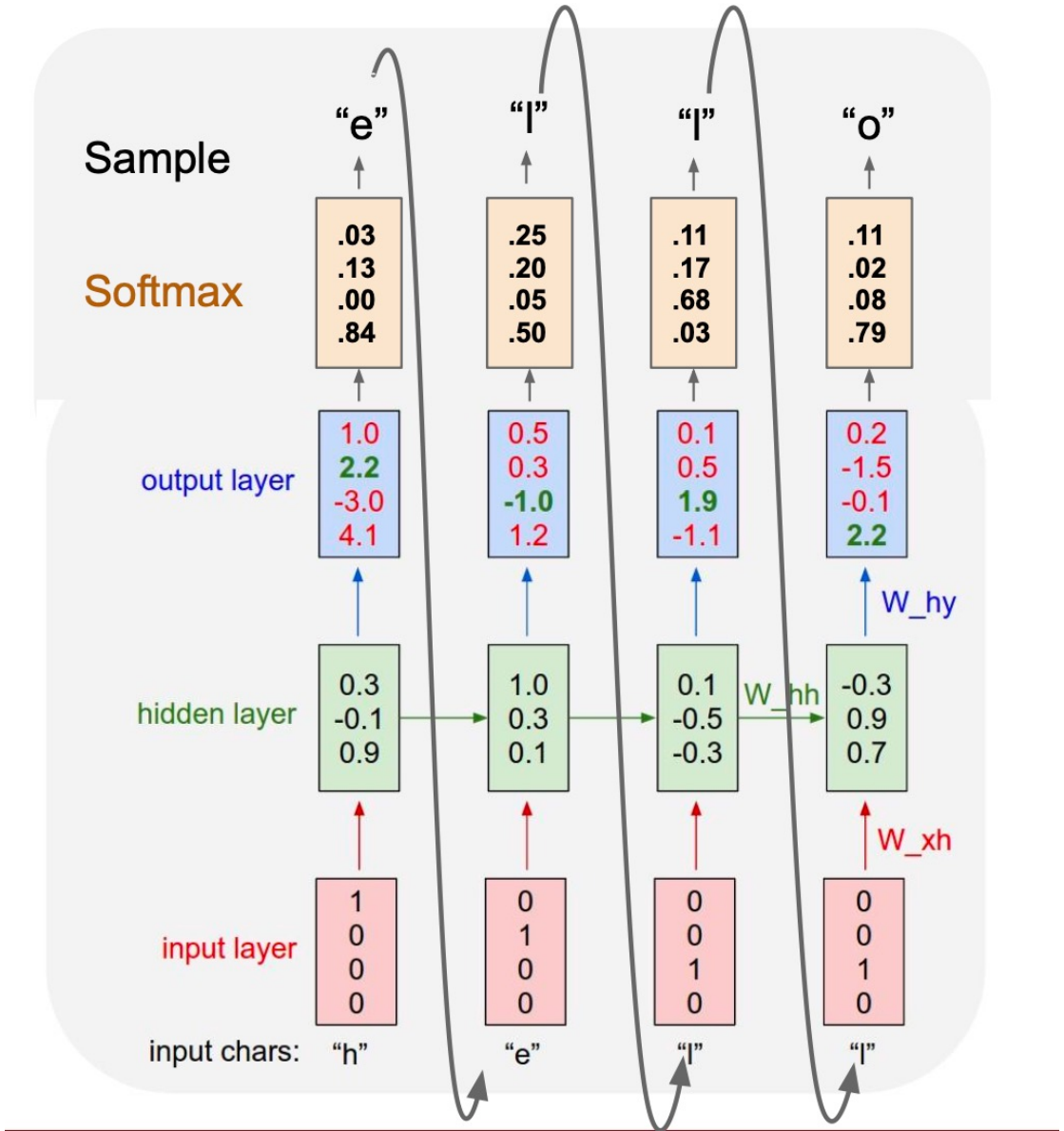


Recurrent Neural Network

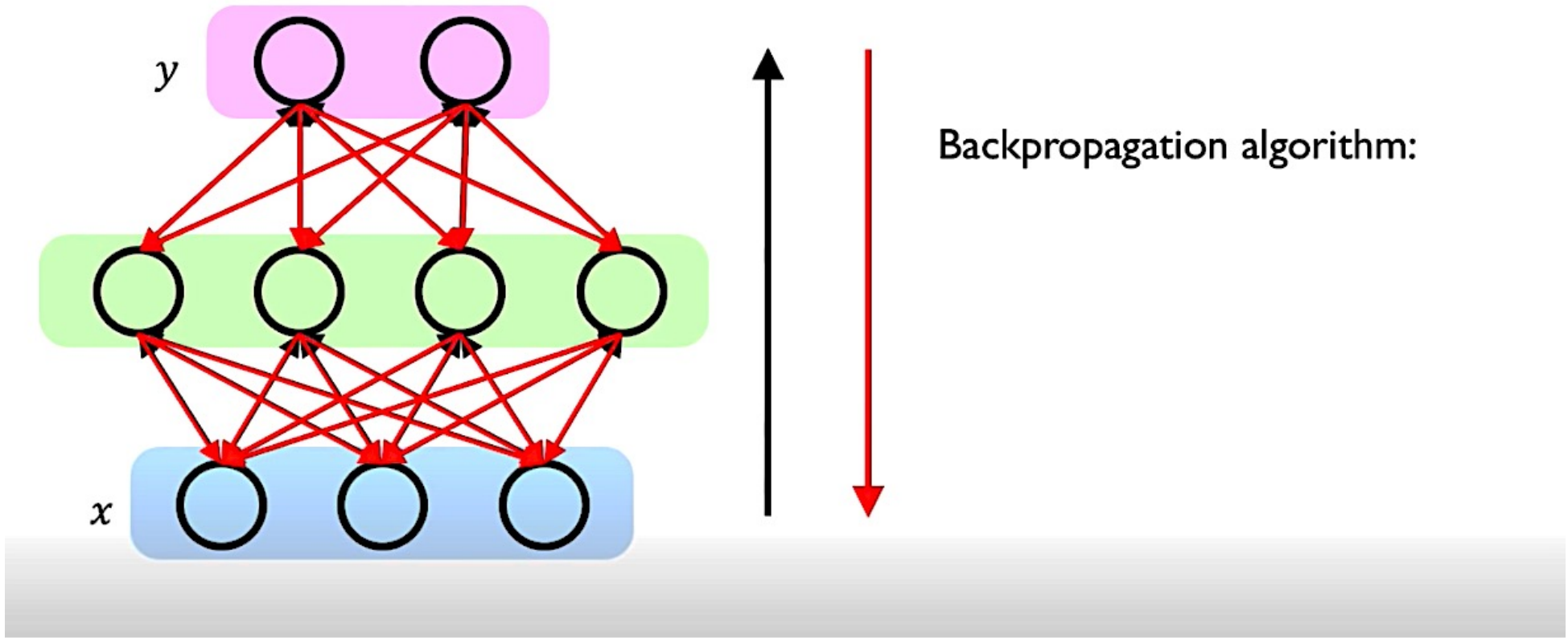
Example: Character-level Language Model

Vocabulary: [h,e,l,o]

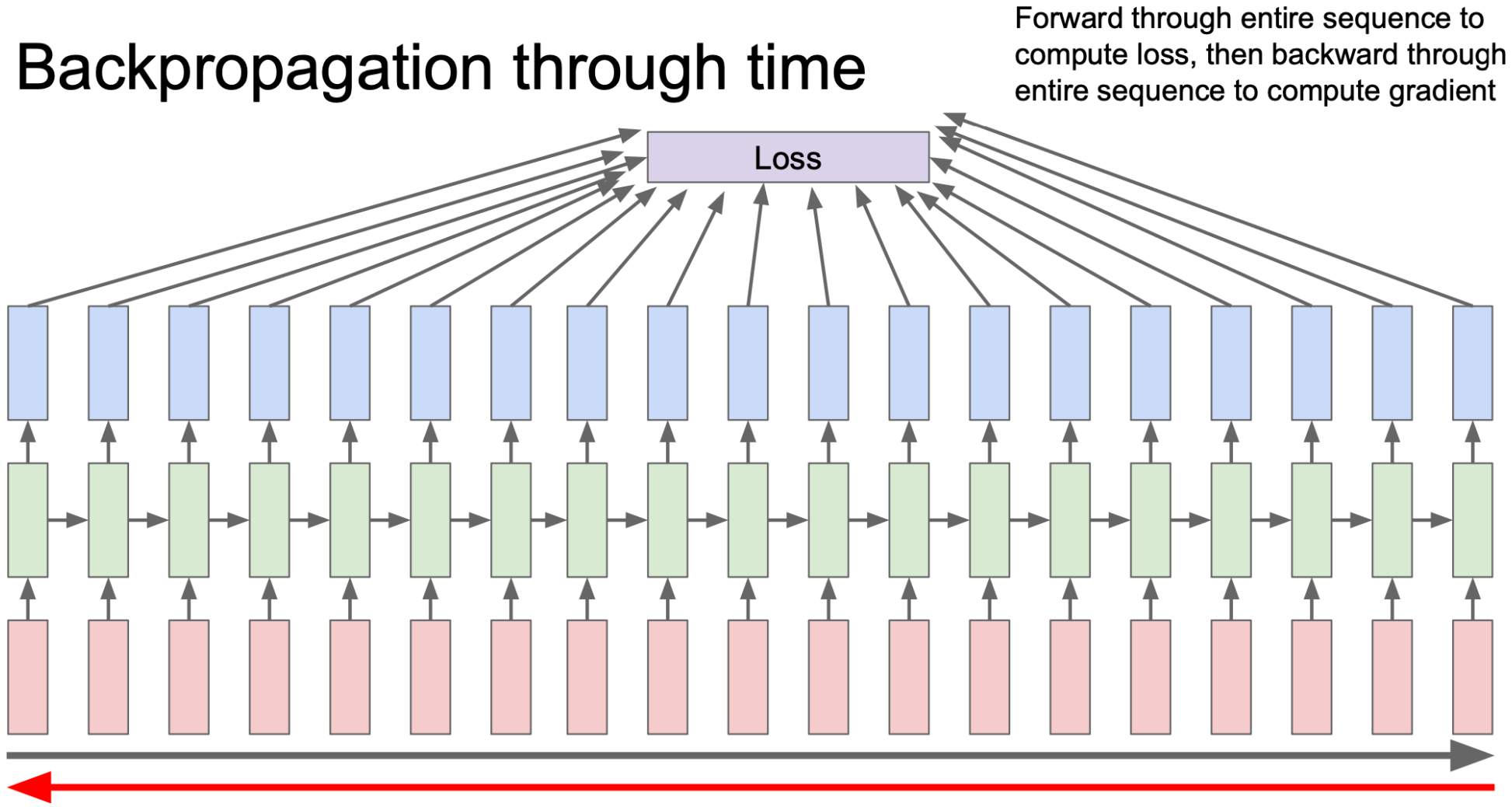
At **test-time** sample characters one at a time, feed back to model



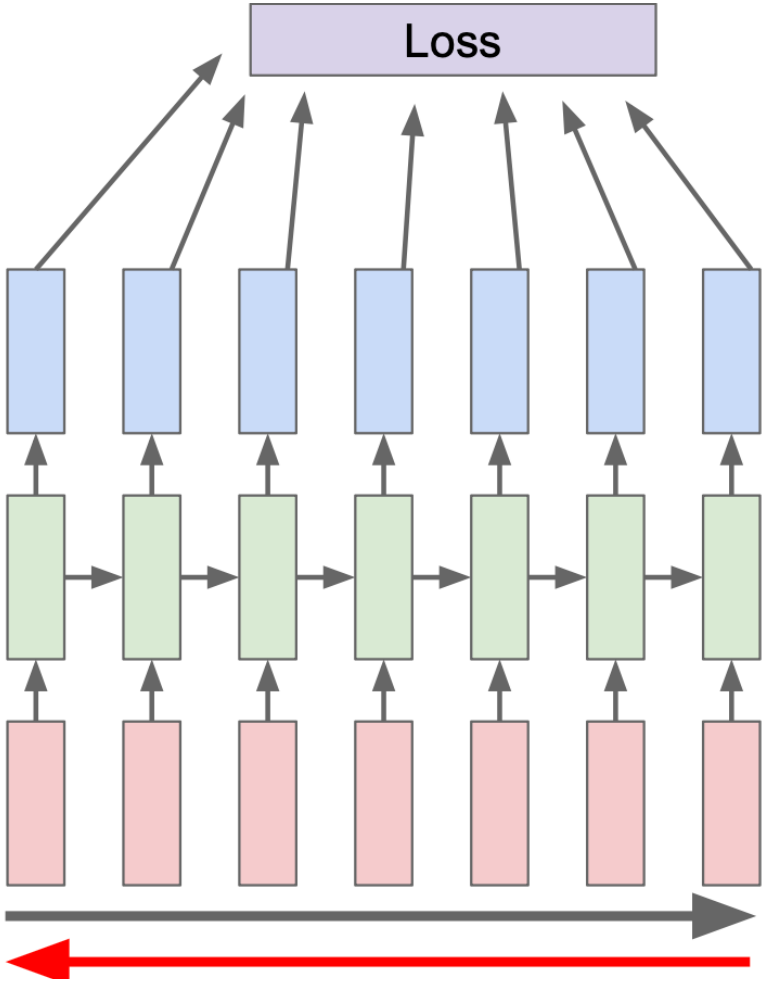
FFNN: Forward and Backward Propagation



Recurrent Neural Network: Backpropagation

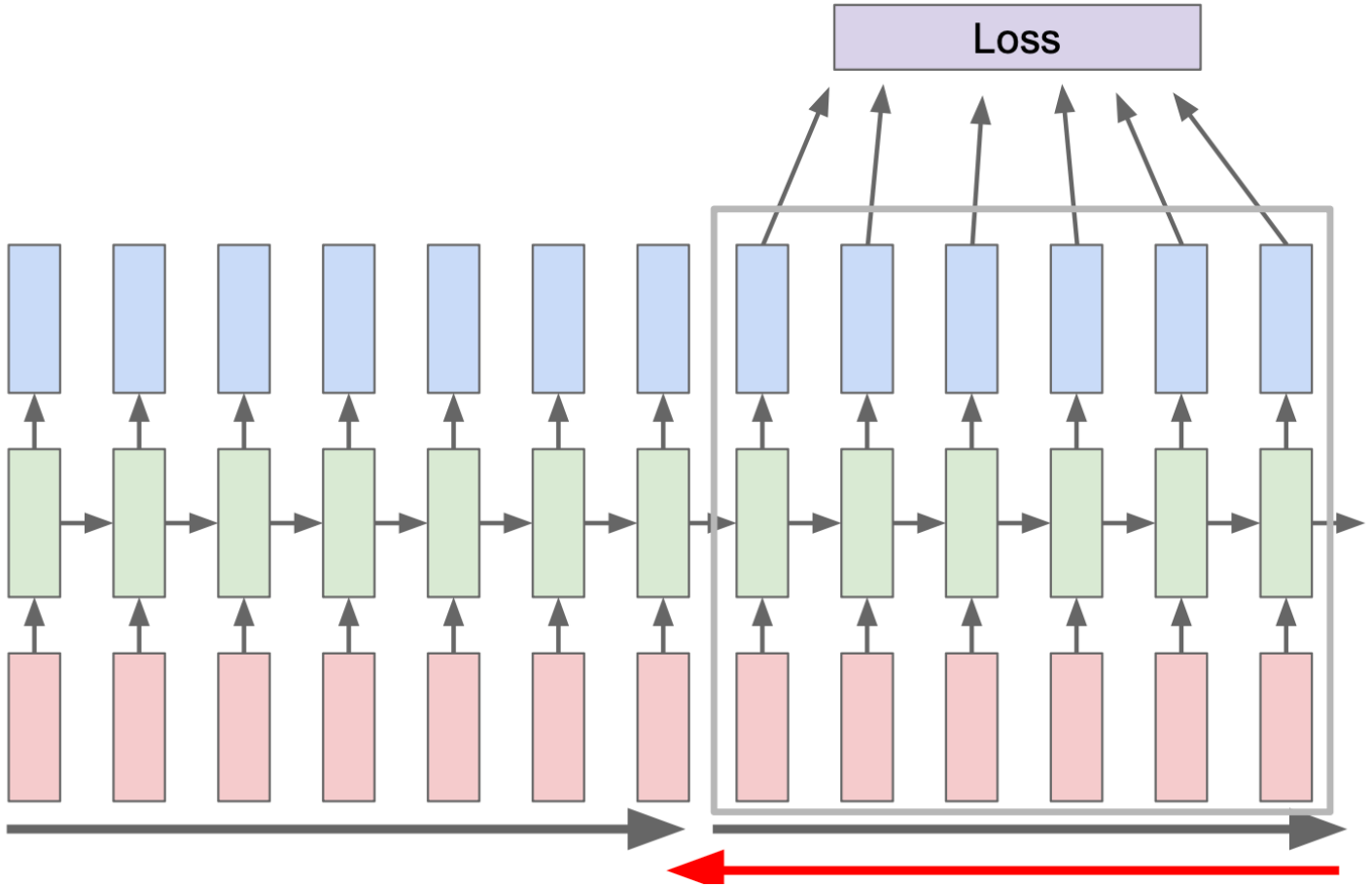


Recurrent Neural Network: Truncated Backpropagation



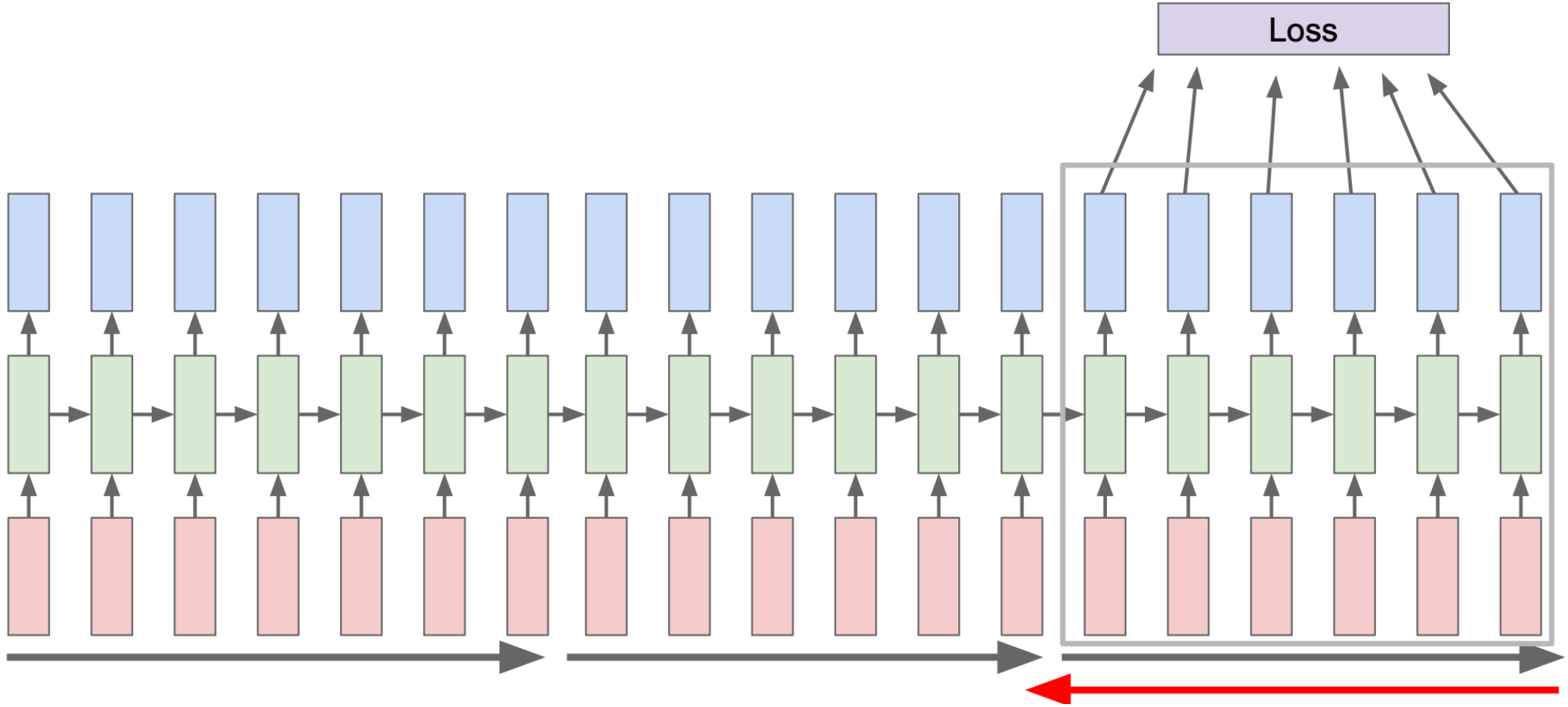
Run forward and backward through chunks of the sequence instead of whole sequence

Recurrent Neural Network: Truncated Backpropagation

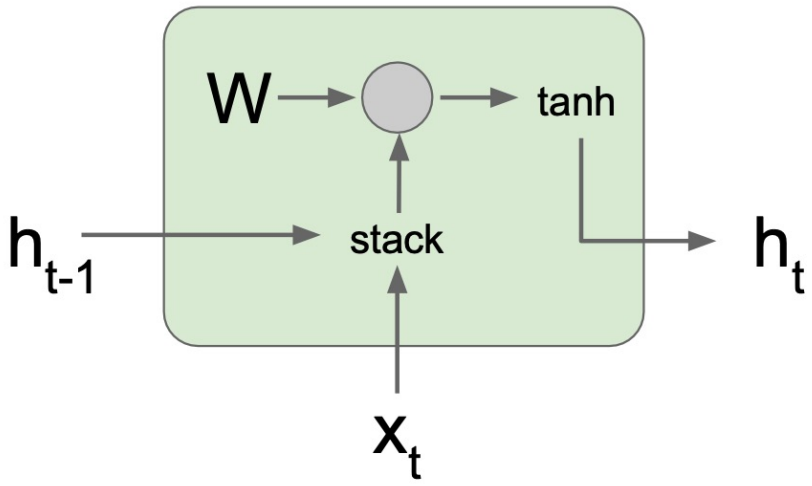


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Recurrent Neural Network: Truncated Backpropagation



Vanilla RNN Gradient Flow

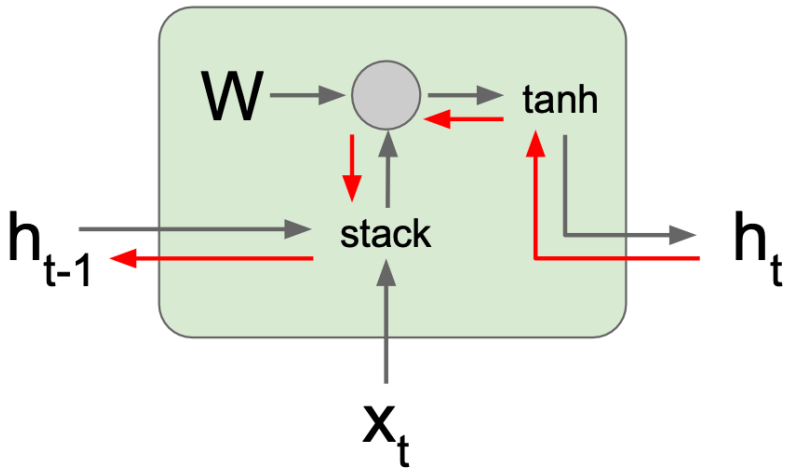


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Vanilla RNN Gradient Flow

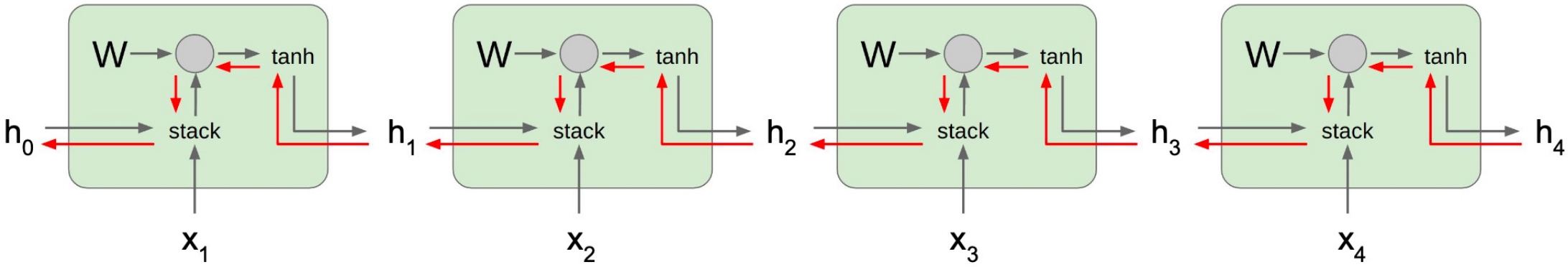
Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



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

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Vanilla RNN Gradient Flow

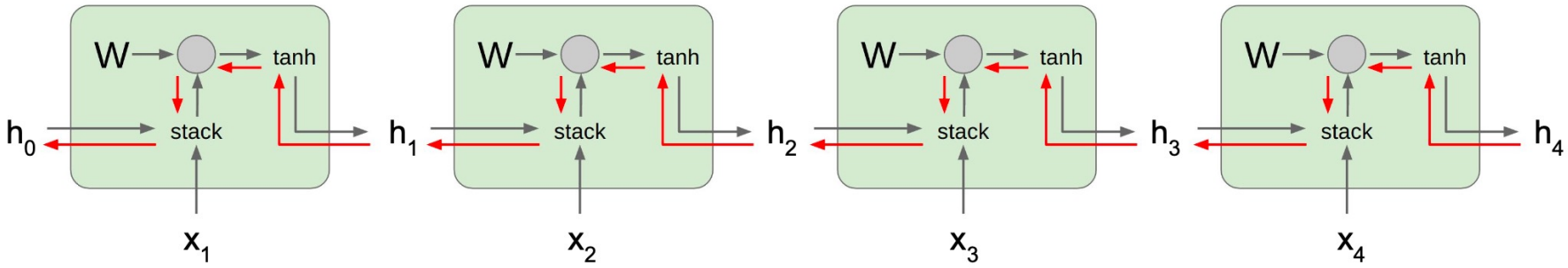


Computing gradient of h0 involves many factors of W (and repeated tanh)

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



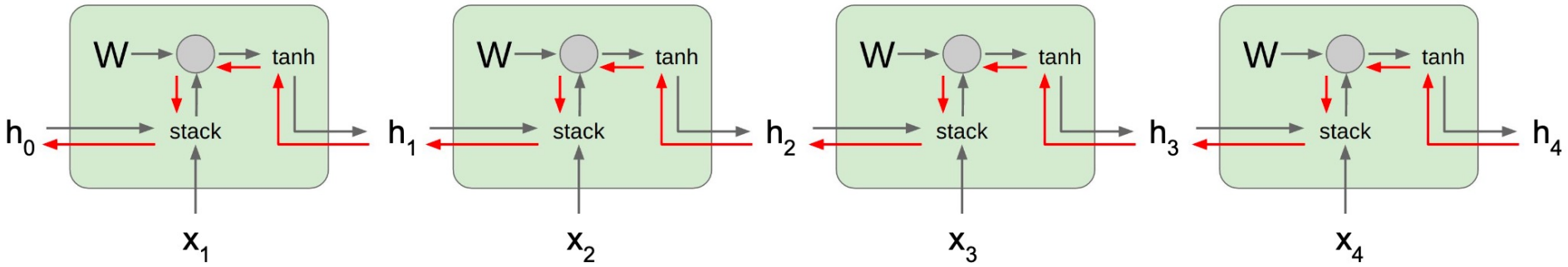
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

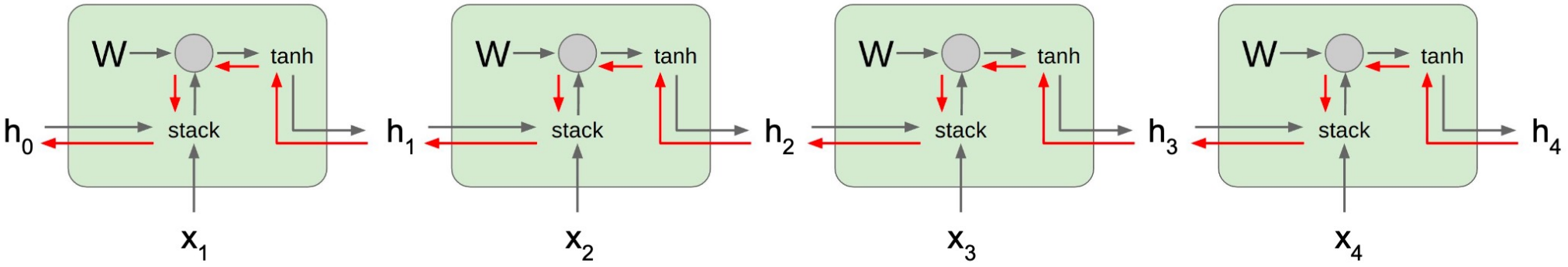
→ **Gradient clipping**: Scale gradient if its norm is too big

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

Largest singular value < 1: **Vanishing gradients**

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients → Change RNN architecture

LSTM: Long Short Term Memory

Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$ $\quad$ $W^l \ [n \times 2n]$

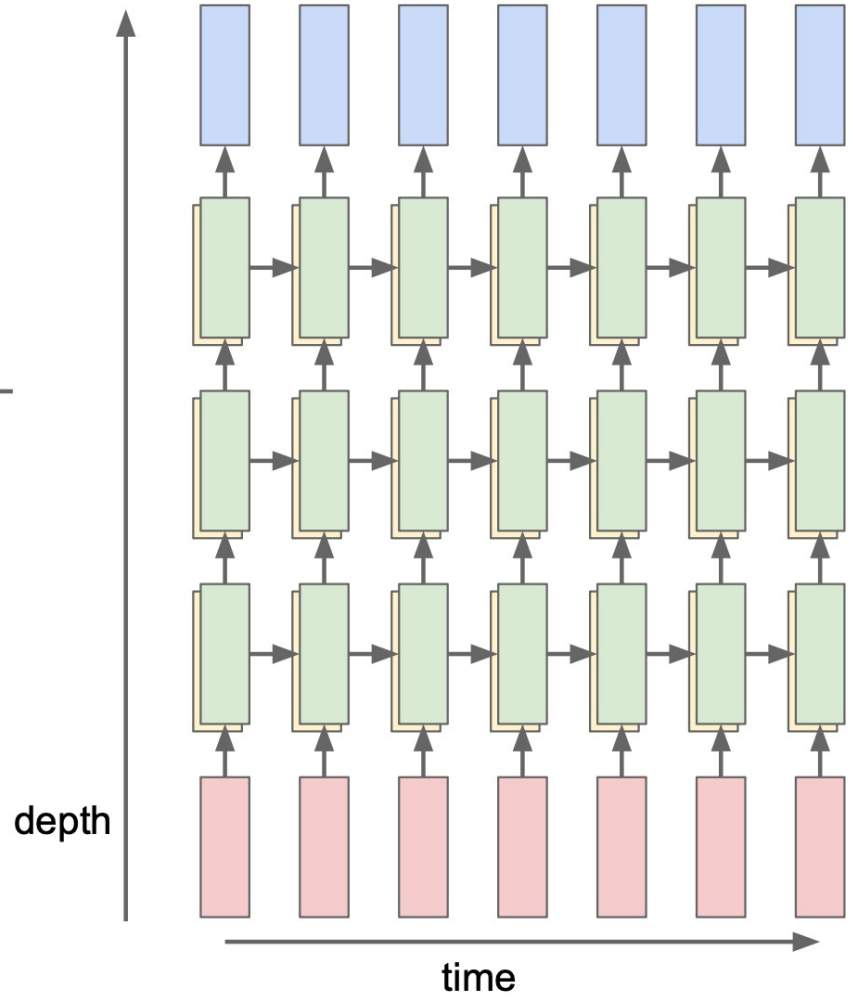
LSTM:

$W^l \ [4n \times 2n]$

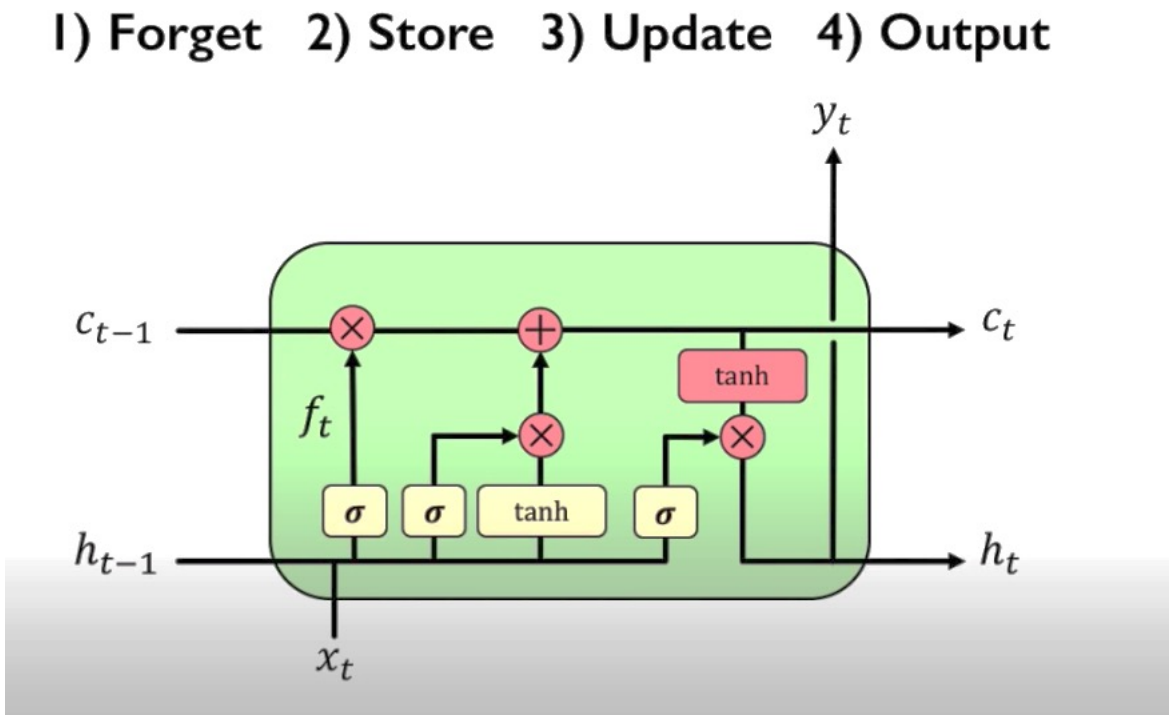
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



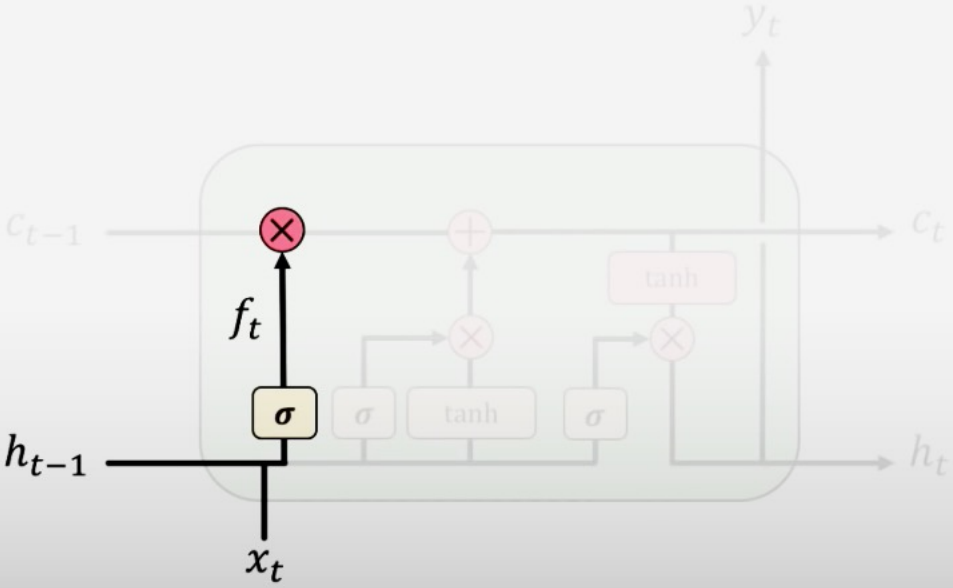
LSTM


1) Forget 2) Store 3) Update 4) Output
y_t
c_{t-1}
c_t
f_t
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
x_t

LSTM: Forget Gate

1) Forget 2) Store 3) Update 4) Output

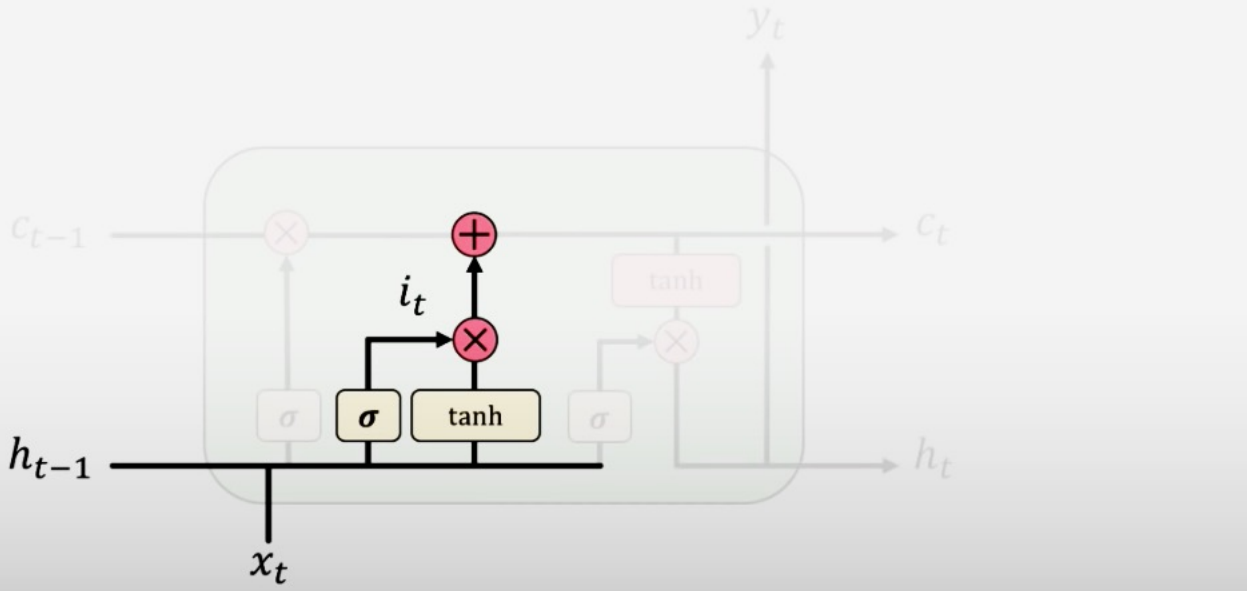
LSTMs **forget irrelevant** parts of the previous state



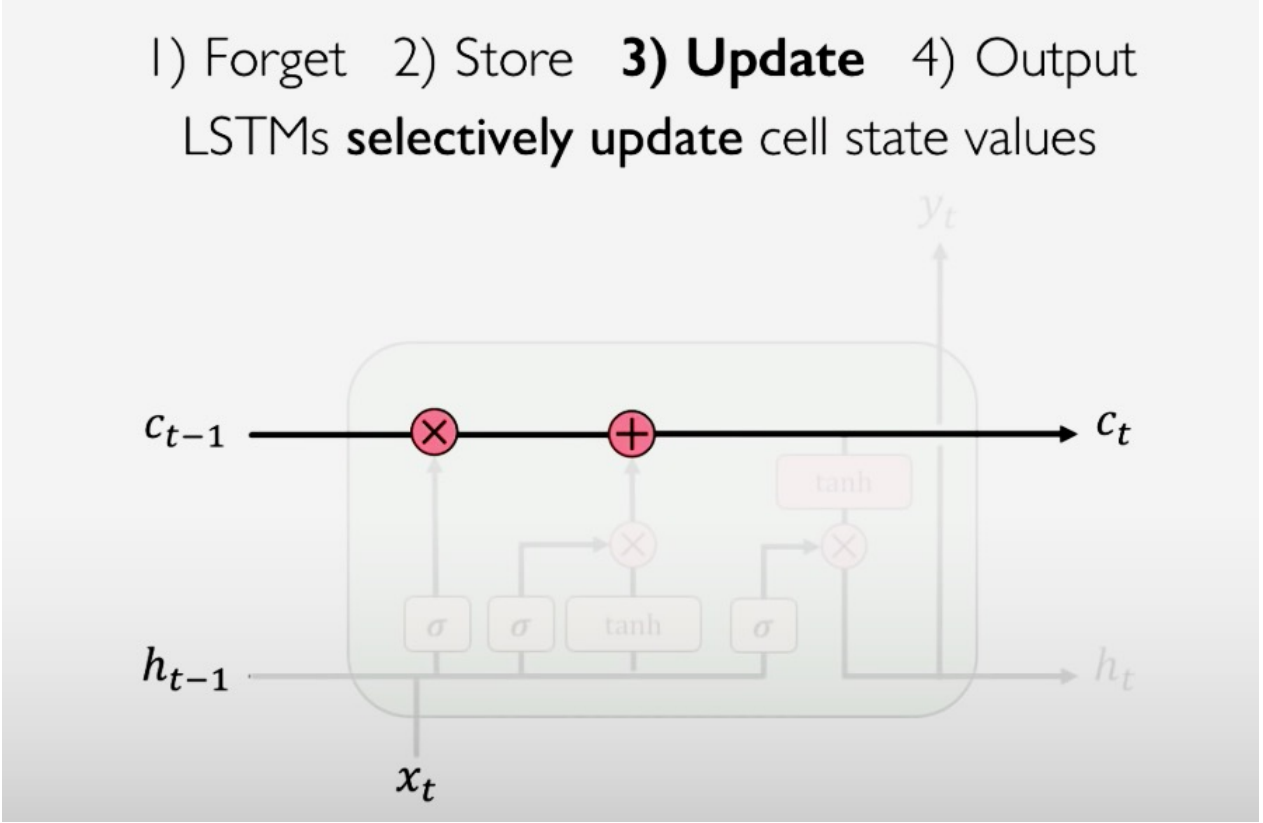
LSTM: Store Gate

1) Forget **2) Store** 3) Update 4) Output

LSTMs **store relevant** new information into the cell state

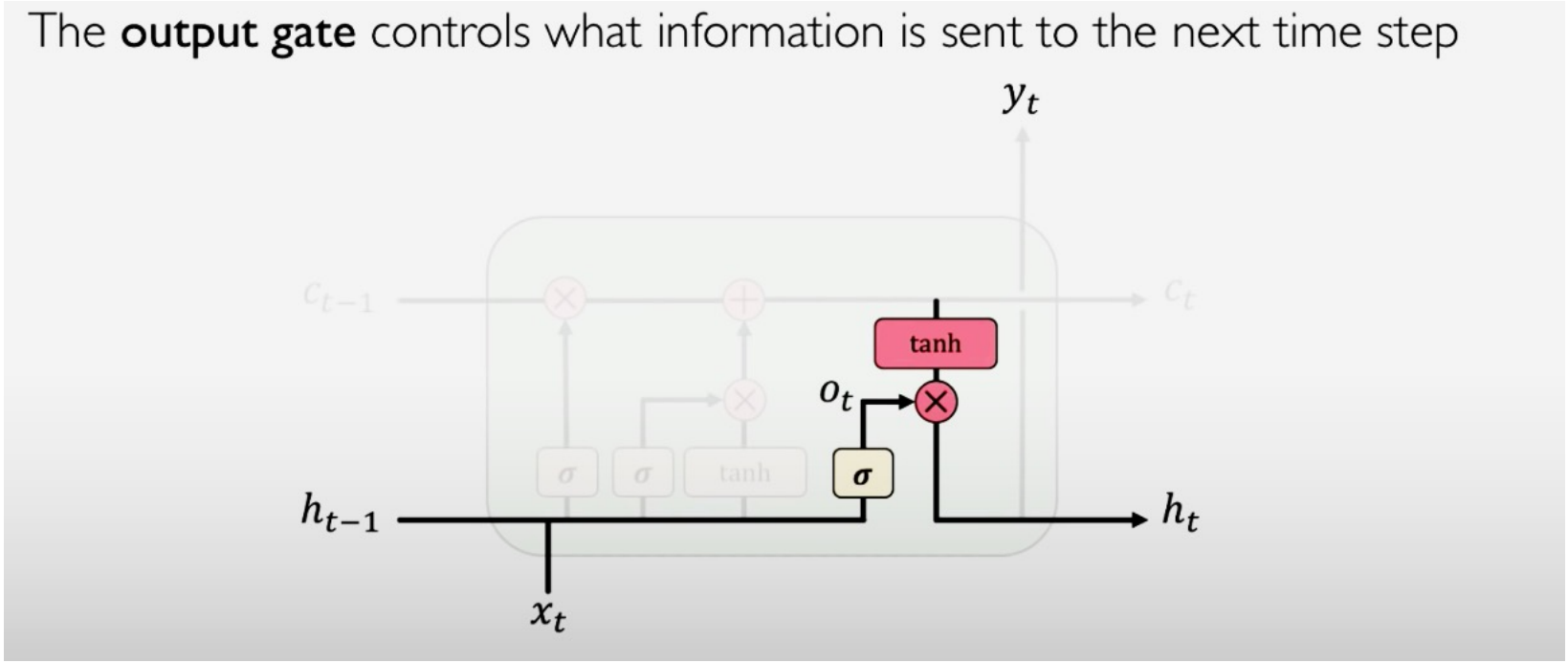


LSTM: Update Gate



LSTM: Output Gate

The **output gate** controls what information is sent to the next time step



LSTM: Long Short Term Memory

Vanilla RNN

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

LSTM

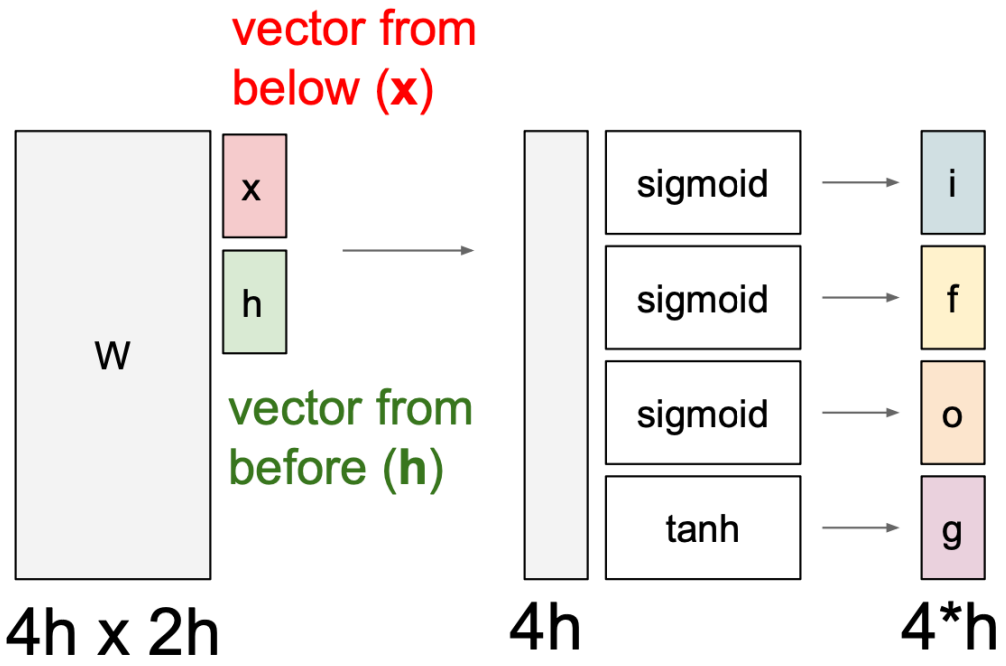
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997

LSTM: Long Short Term Memory



i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell

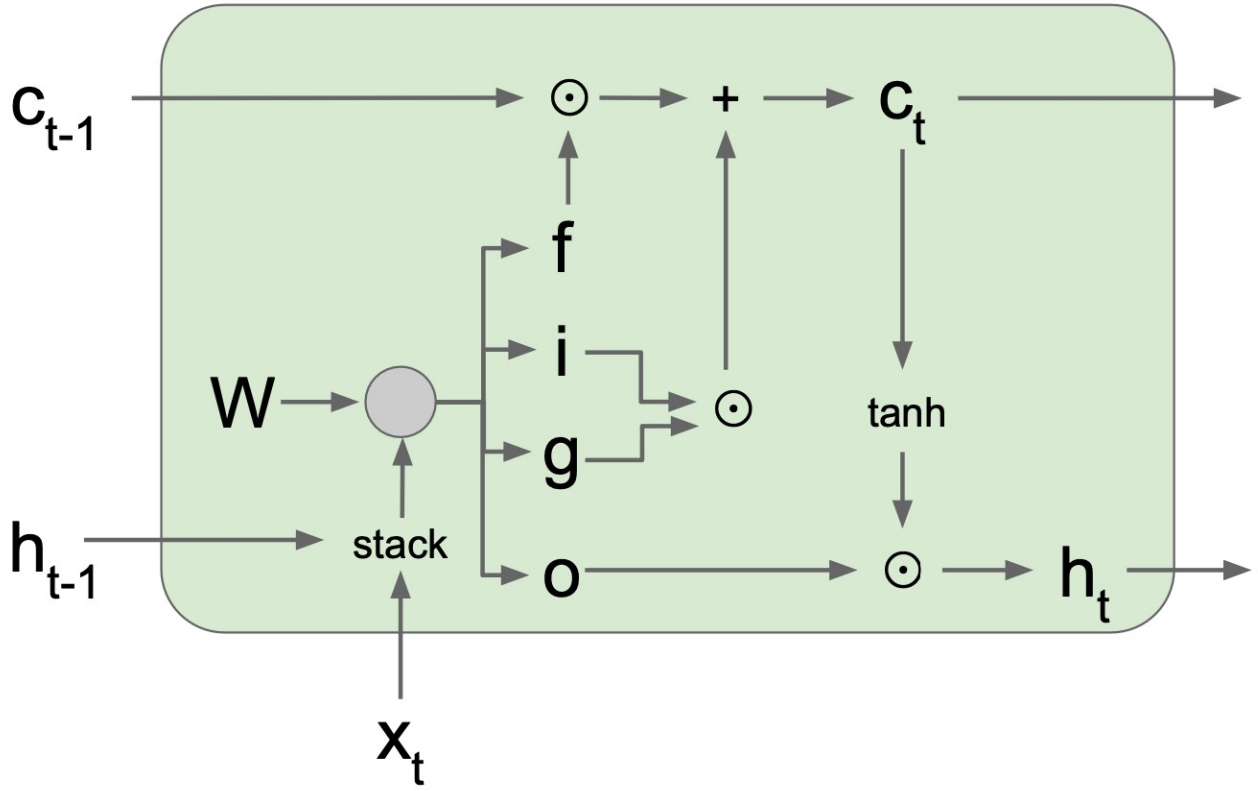
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

LSTM: Long Short Term Memory



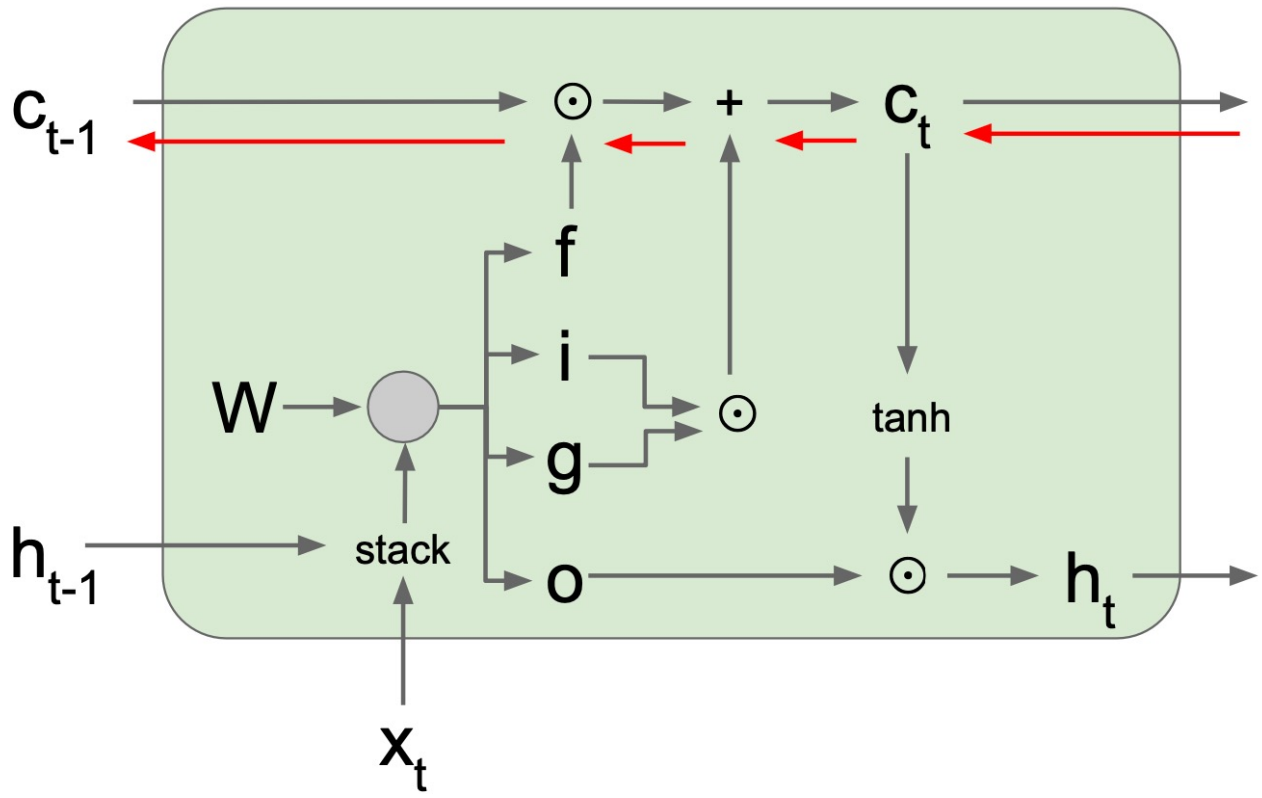
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997

LSTM: Long Short Term Memory: Gradient Flow



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Other RNN Variants: GRU

GRU [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

[*LSTM: A Search Space Odyssey*, Greff et al., 2015]

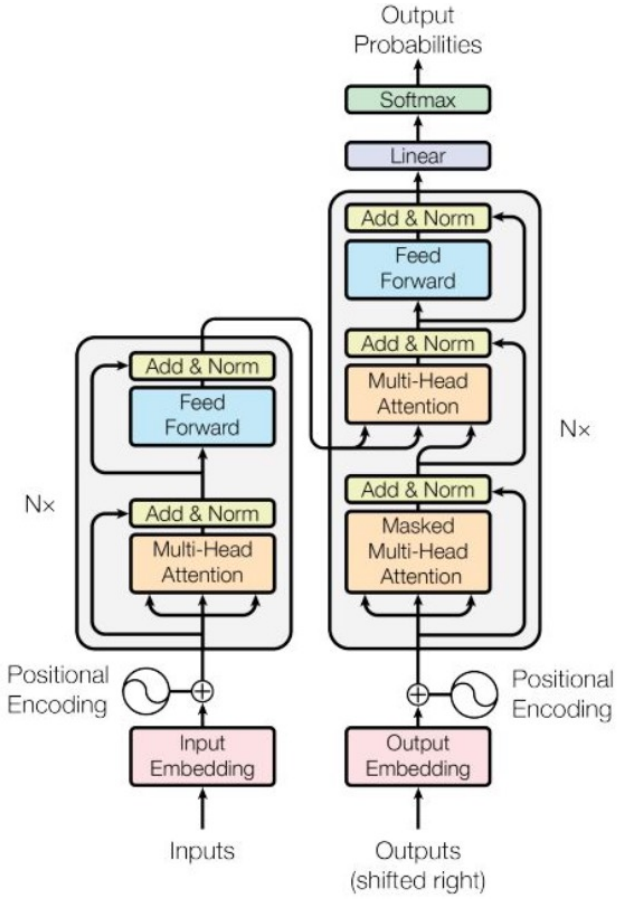
Recently in Natural Language Processing... New paradigms for reasoning over sequences

• New “Transformer” architecture no longer processes inputs sequentially; instead, it can operate over inputs in a sequence in parallel through an attention mechanism

• Has led to many state-of-the-art results and pre-training in NLP, for more interest see e.g.

- - “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, Devlin et al., 2018
- - OpenAI GPT-2, Radford et al., 2018

[“Attention is all you need”, Vaswani et al., 2018]



Summary

- What is a feedforward neural network?
- Why do we need a convolutional neural network?
- What is the main component of CNNs?
- What are some examples of timeseries data?
- How does RNN handle the data?
- What is the main difference of RNN and LSTM?
- Next time: transformers

References

1. Deep Learning Slides by Fei-Fei LI & Justin Johnson & Serena Yeung & Charu C. Aggarwal
2. MIT Deep Learning Slides: http://introtodeeplearning.com/