VIRGINIA TECH

Recurrent Neural Networks

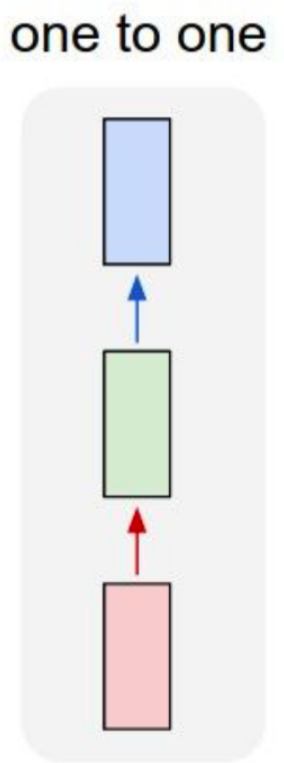
Lifu Huang

lifuh@vt.edu

Torgersen Hall, Suite 3160E

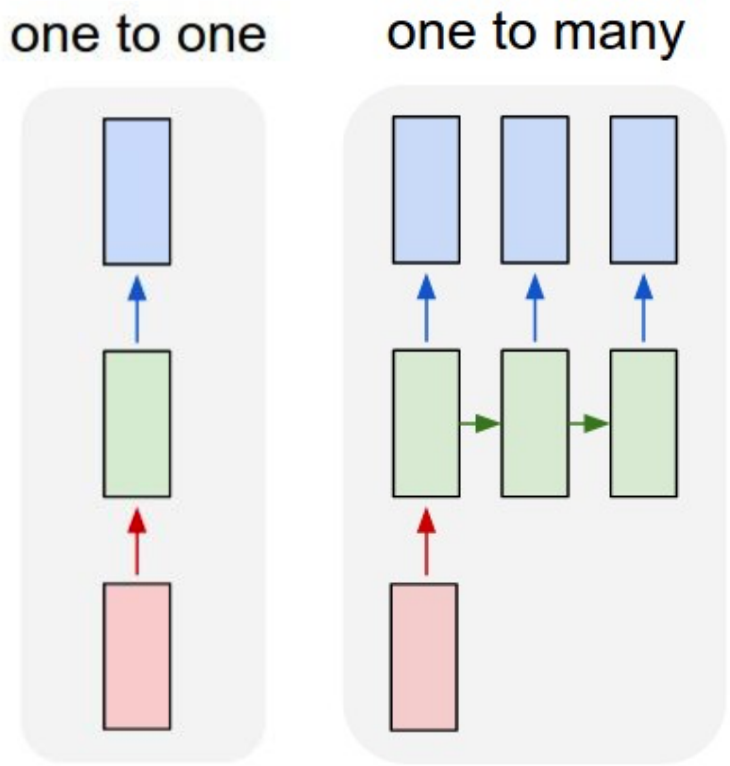
Slides adapted from Feifei Li, Justin Johnson, Serena Yeung

Recurrent Neural Networks: Process Sequences



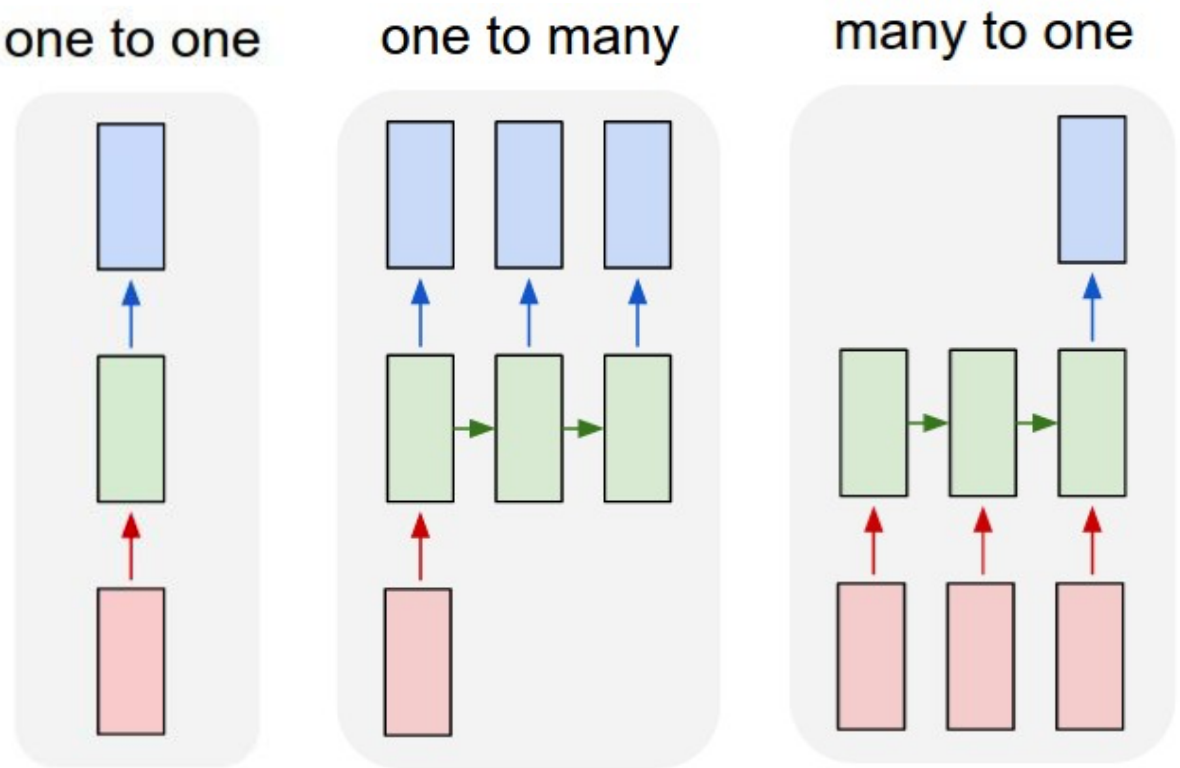
Vanilla Neural Networks
e.g., **Image Classification**
Image -> Label

Recurrent Neural Networks: Process Sequences



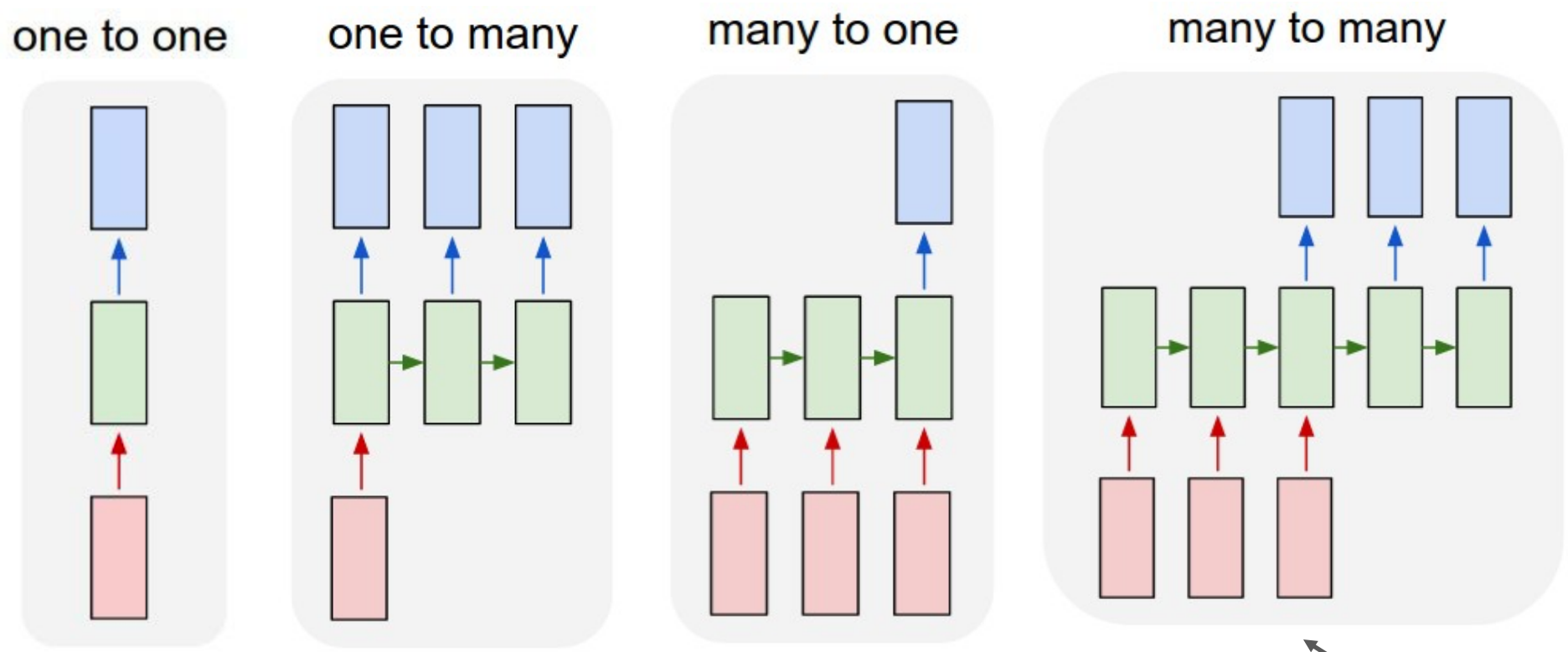
e.g., **Image Captioning**
Image -> Sequence of Words

Recurrent Neural Networks: Process Sequences



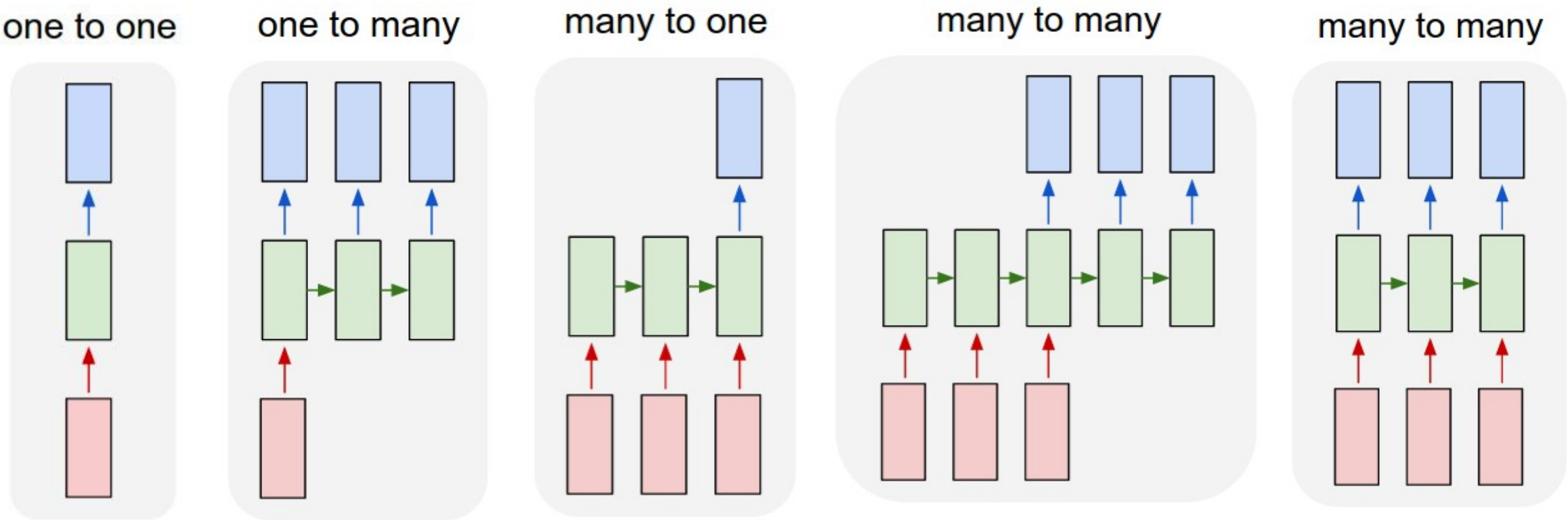
e.g., **Video/Text Classification**
Sequence of images/words -> Label

Recurrent Neural Networks: Process Sequences



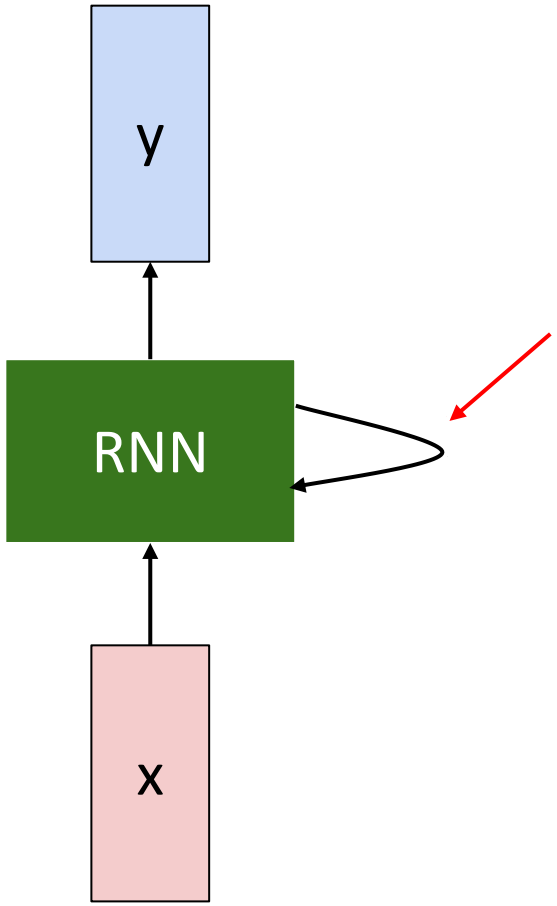
e.g., **Machine Translation**
Sequence of words -> Sequence of words

Recurrent Neural Networks: Process Sequences

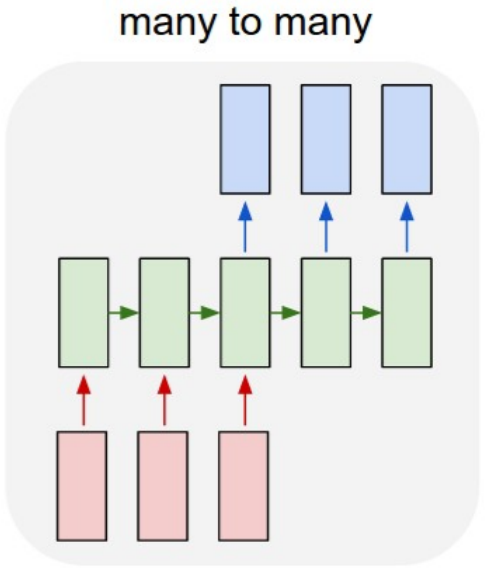


e.g., **Per-frame Video Classification**
Sequence of images -> Sequence of labels

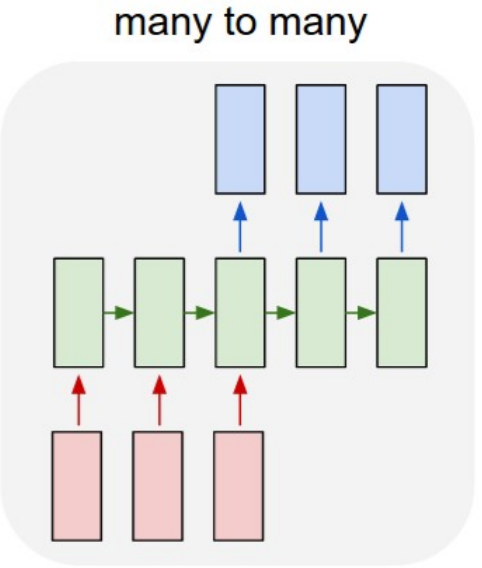
Recurrent Neural Networks

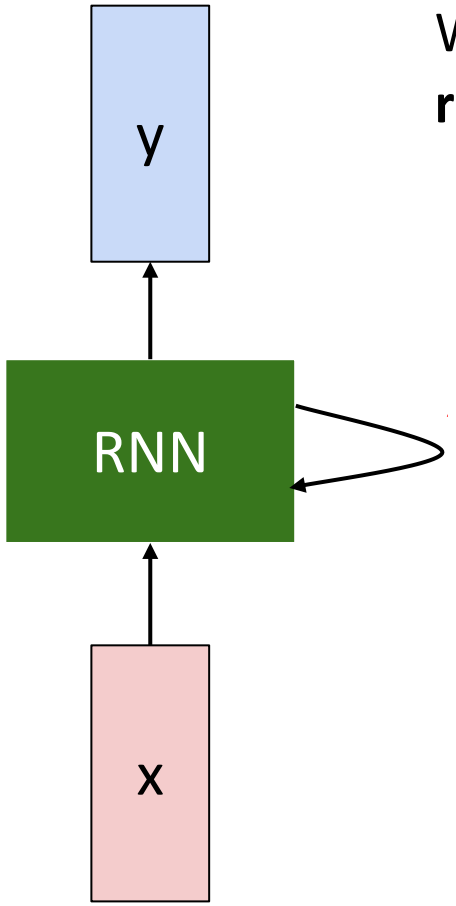


Key idea: RNNs have an "internal state" that is updated as a sequence is processed



Recurrent Neural Networks





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

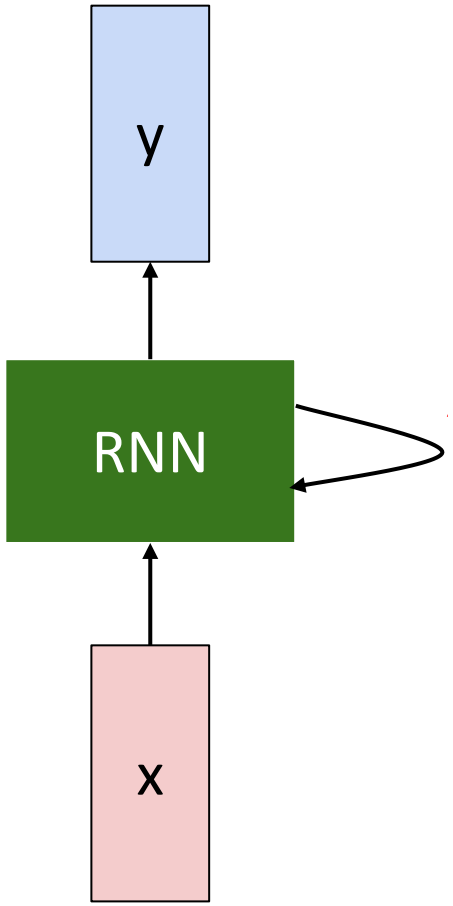
some function with parameters W

old state

input vector at some time step

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

(Vanilla) Recurrent Neural Networks



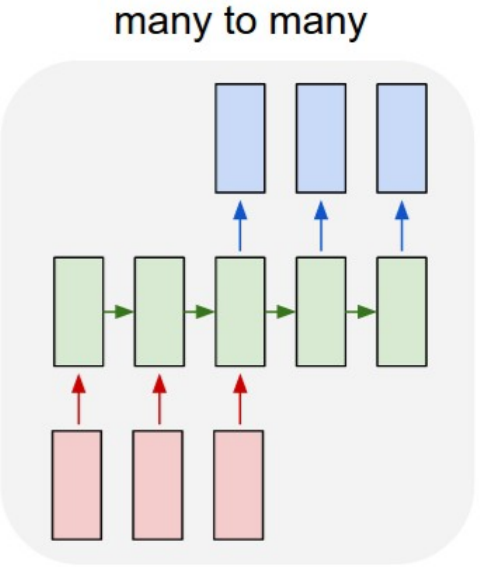
The state consists of a single *"hidden"* vector **h**:

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

(also bias term)

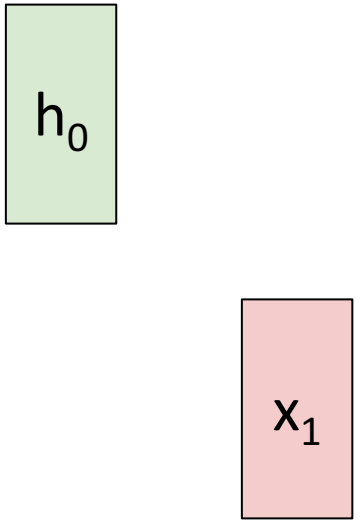
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



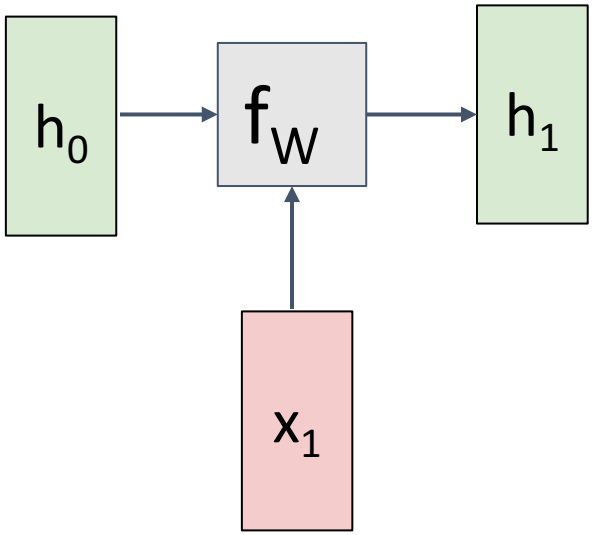
RNN Computational Graph

Initial hidden state
Either set to all 0,
or learn it



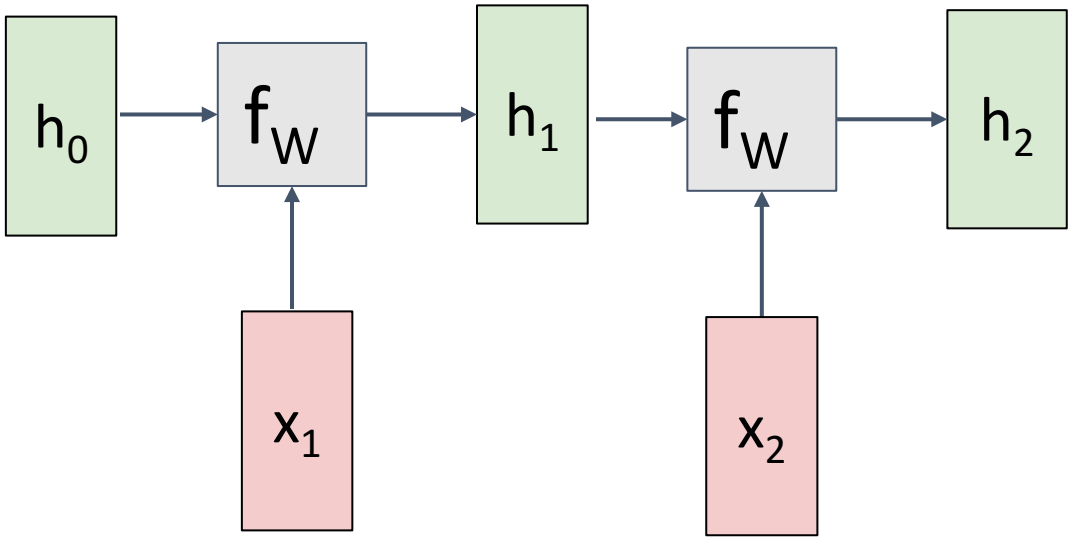
RNN Computational Graph

Initial hidden state
Either set to all 0,
or learn it



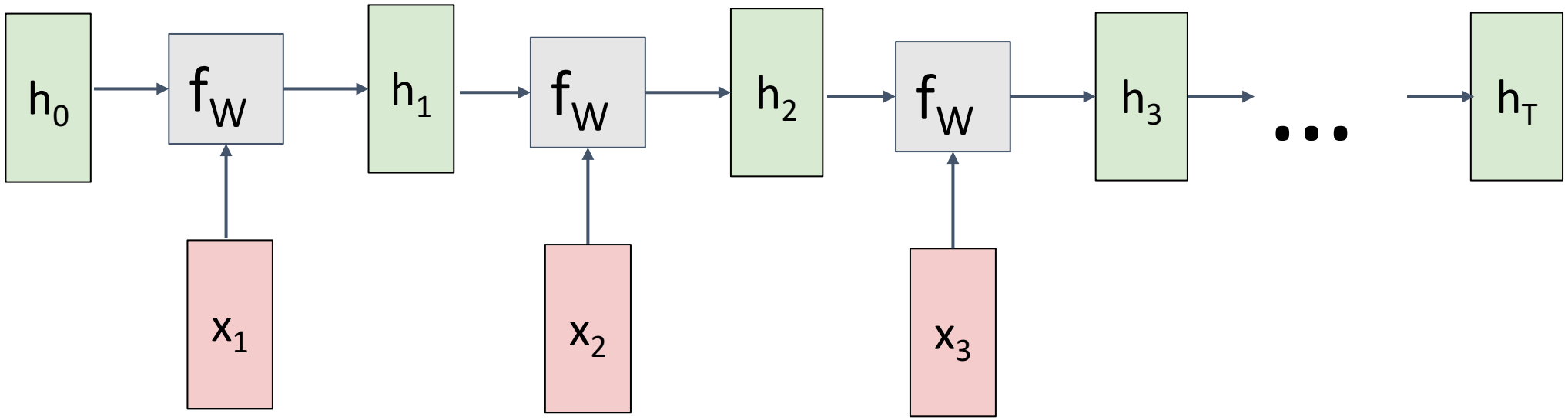
RNN Computational Graph

Initial hidden state
Either set to all 0,
or learn it



RNN Computational Graph

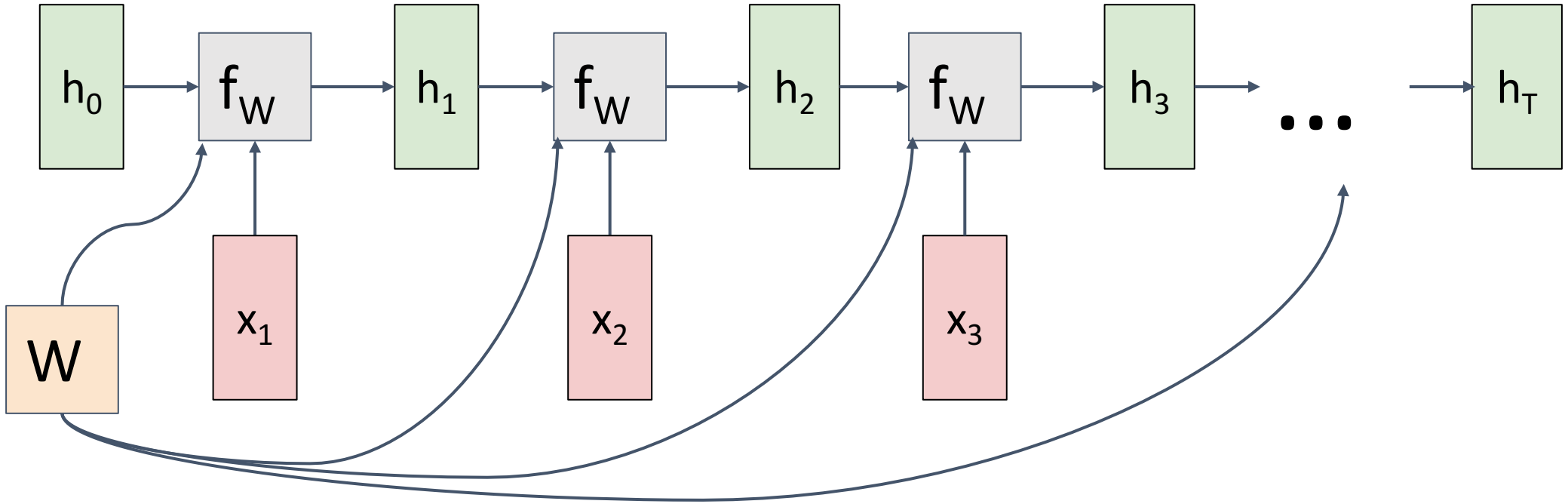
Initial hidden state
Either set to all 0,
or learn it



RNN Computational Graph

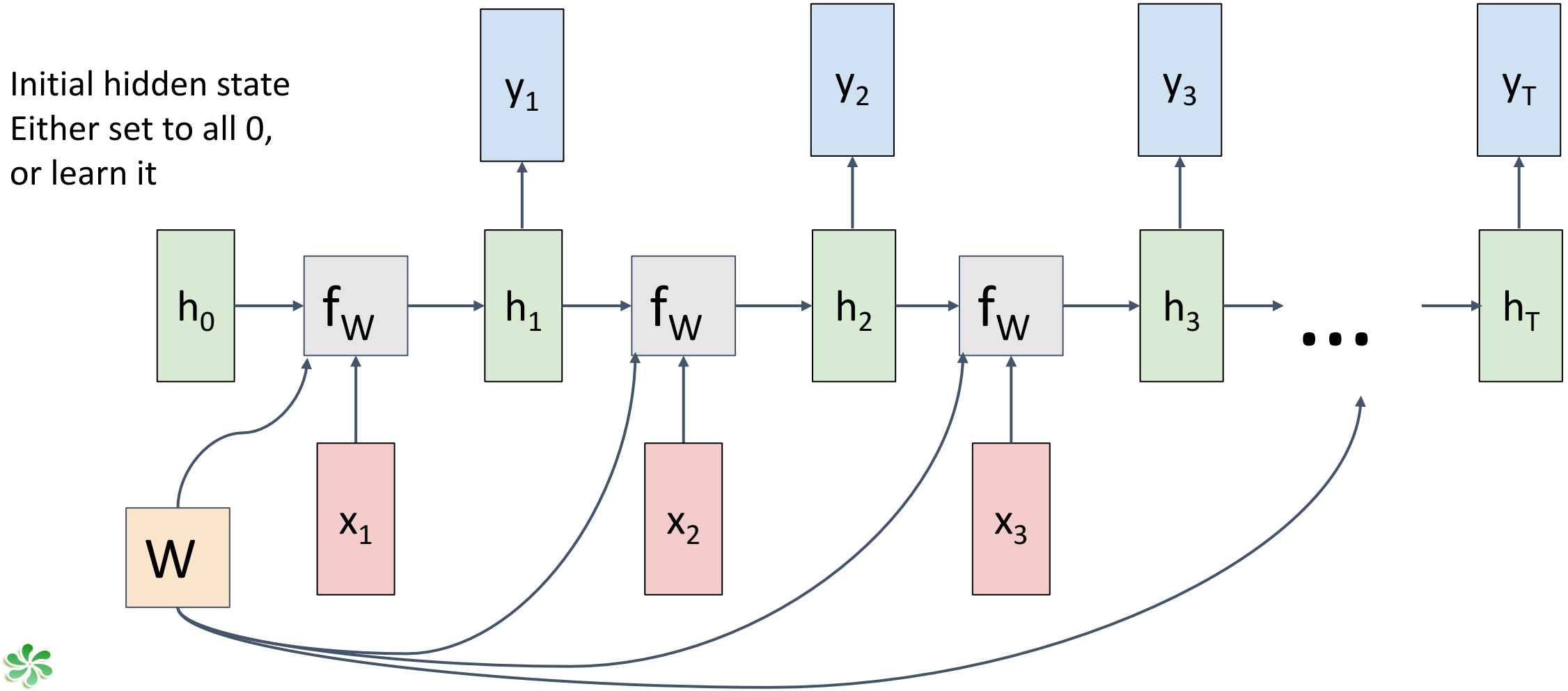
Re-use the same weight matrix at every time-step

Initial hidden state
Either set to all 0,
or learn it



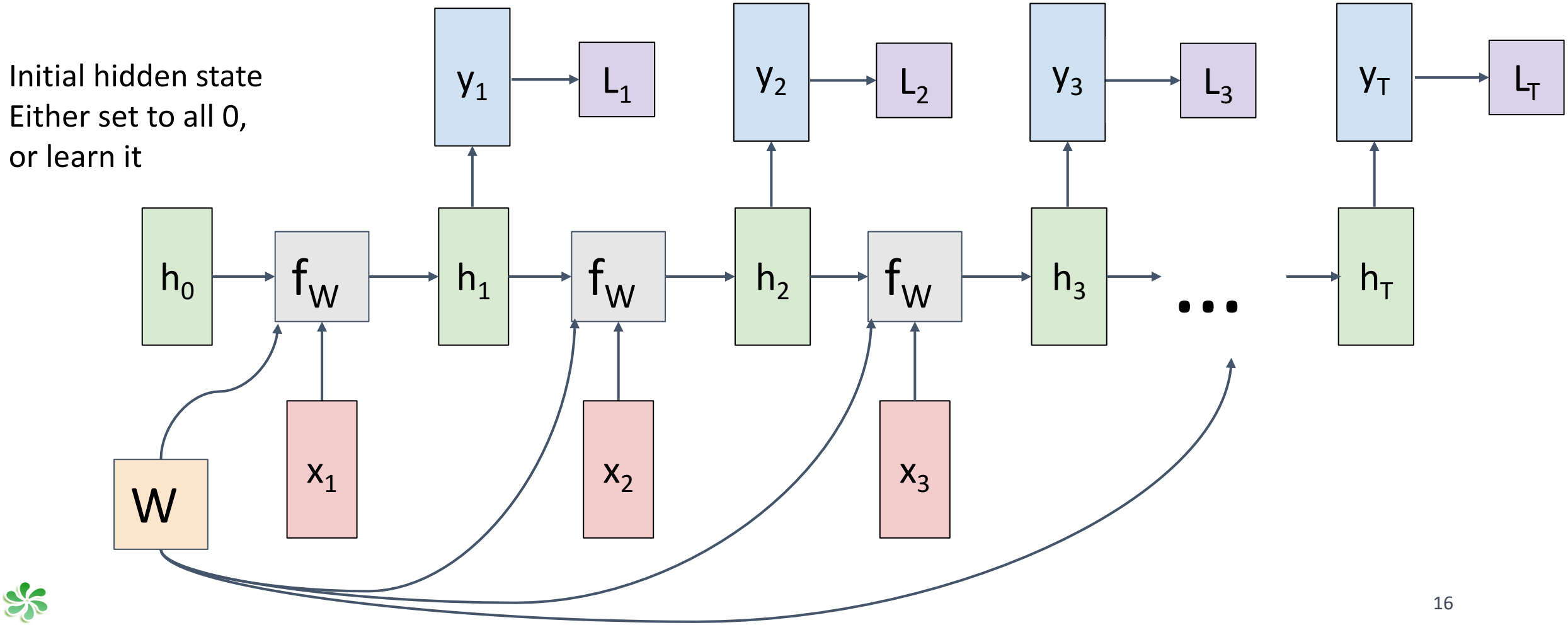
RNN Computational Graph (Many to Many)

- Encode the input sequence and out a new sequence



RNN Computational Graph (Many to Many)

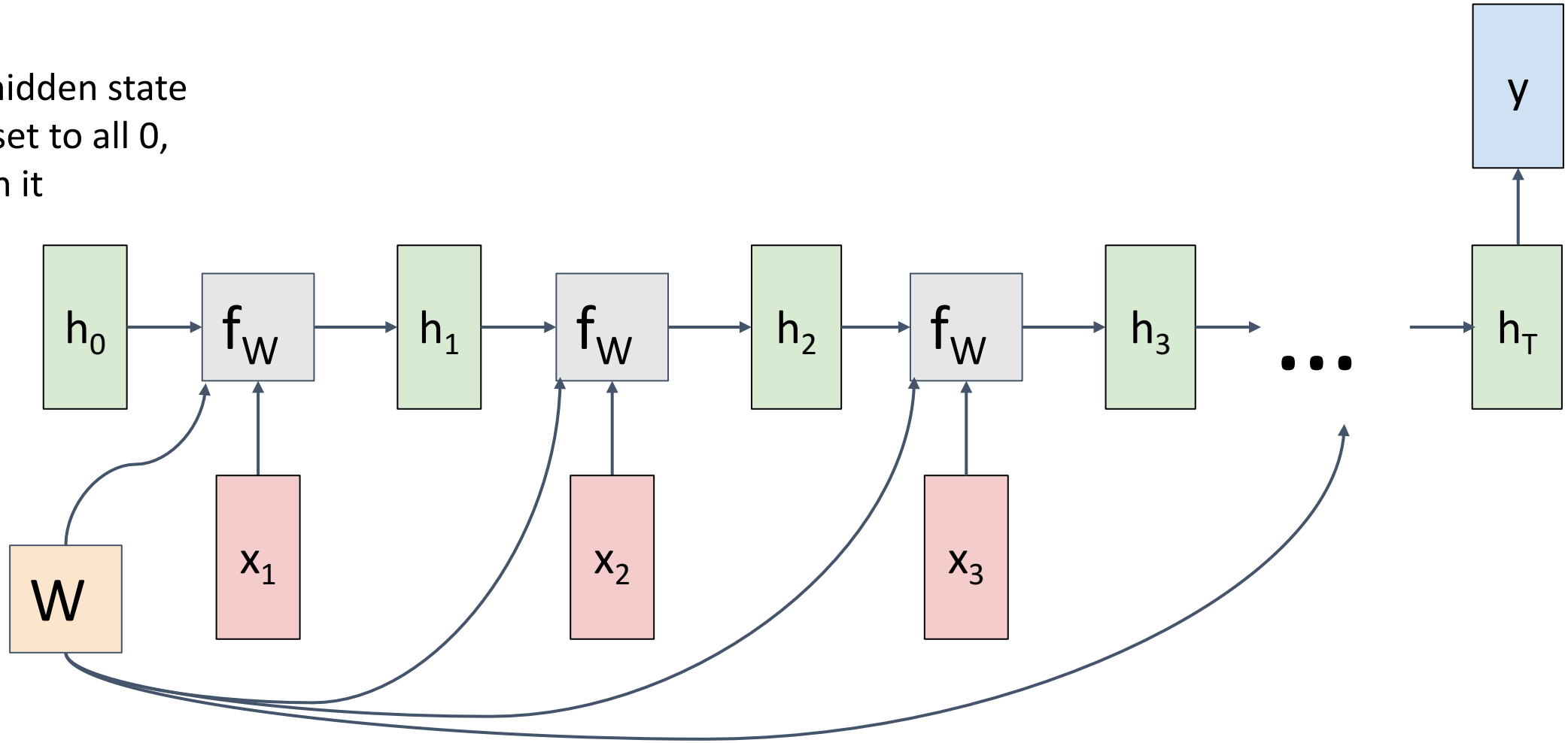
- Encode the input sequence and out a new sequence



RNN Computational Graph (Many to One)

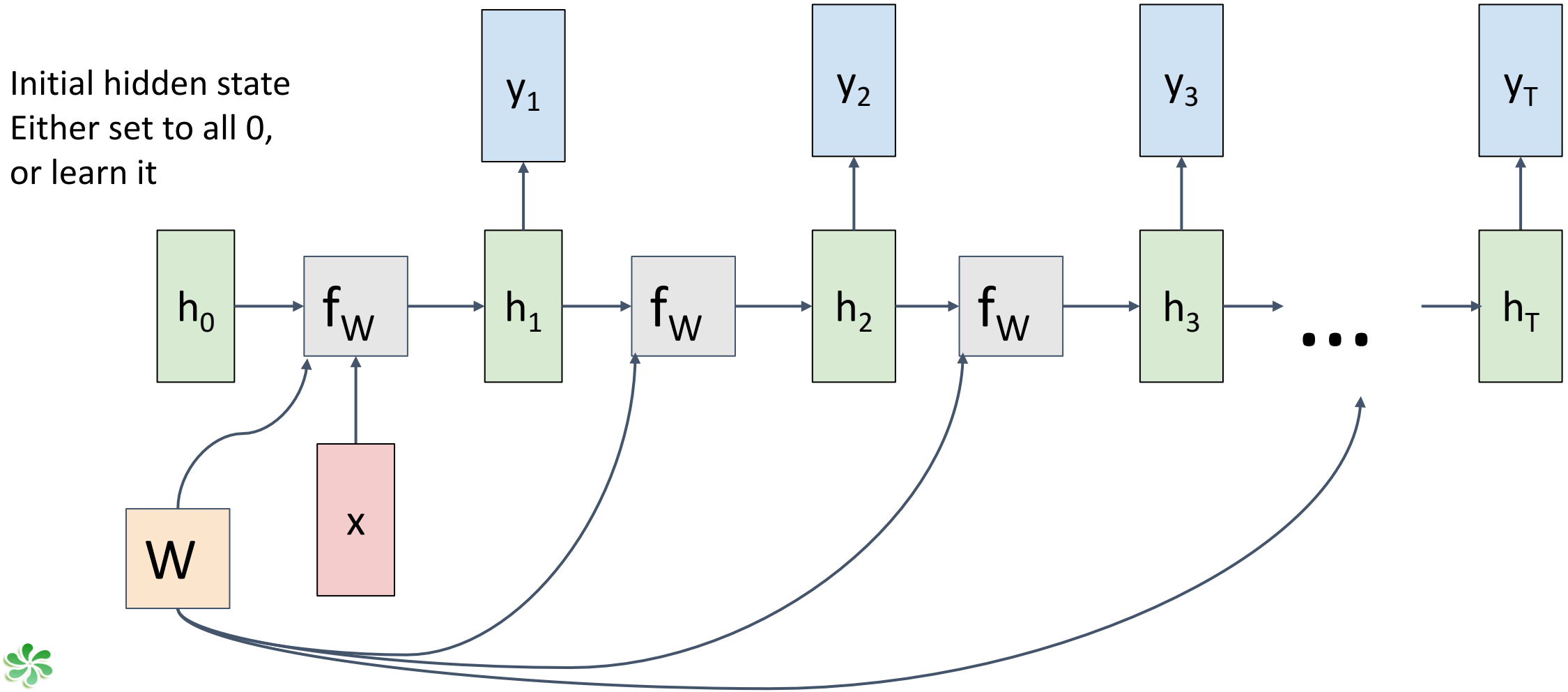
- Encode the input sequence but output a single vector

Initial hidden state
Either set to all 0,
or learn it



RNN Computational Graph (One to Many)

- Take in a single input vector but produce an output sequence

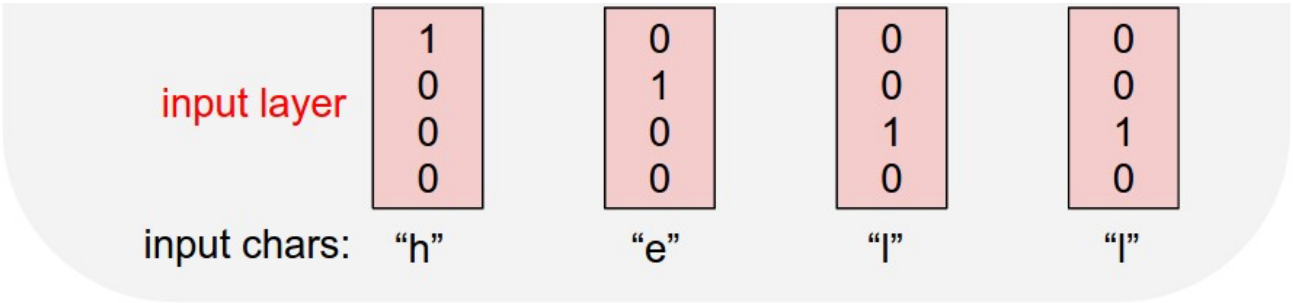


Example: Character-level Language Modeling

Given Characters 1, 2, …, t,
model predicts character t+1

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



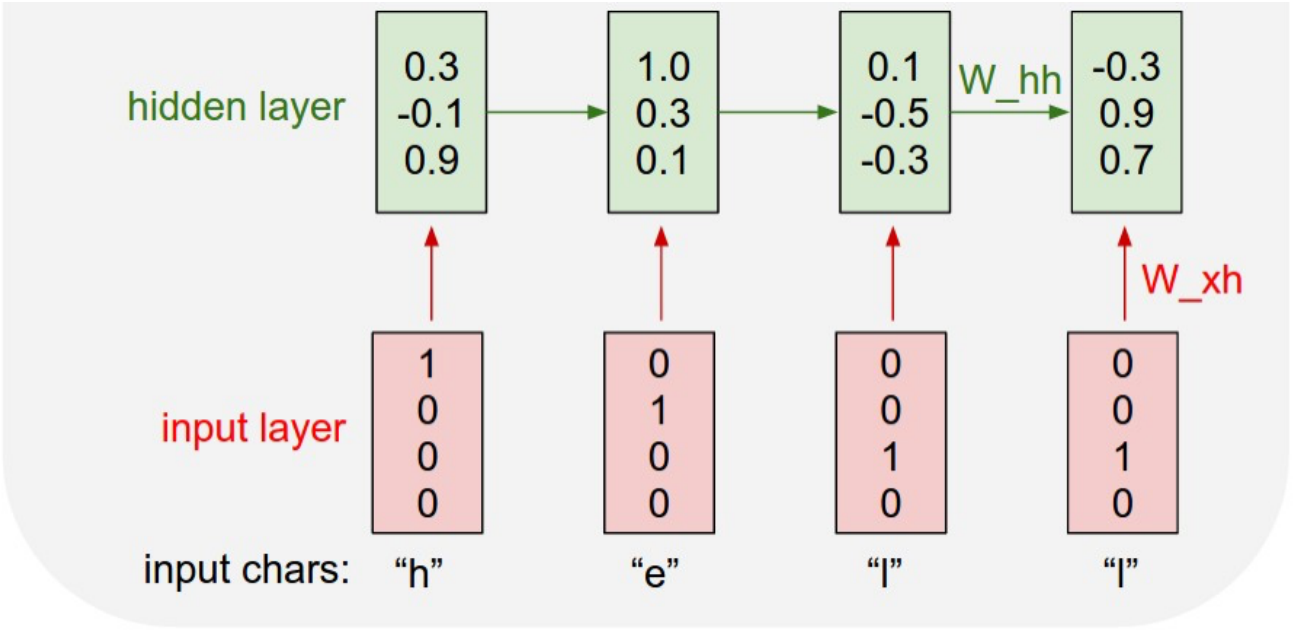
Example: Character-level Language Modeling

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



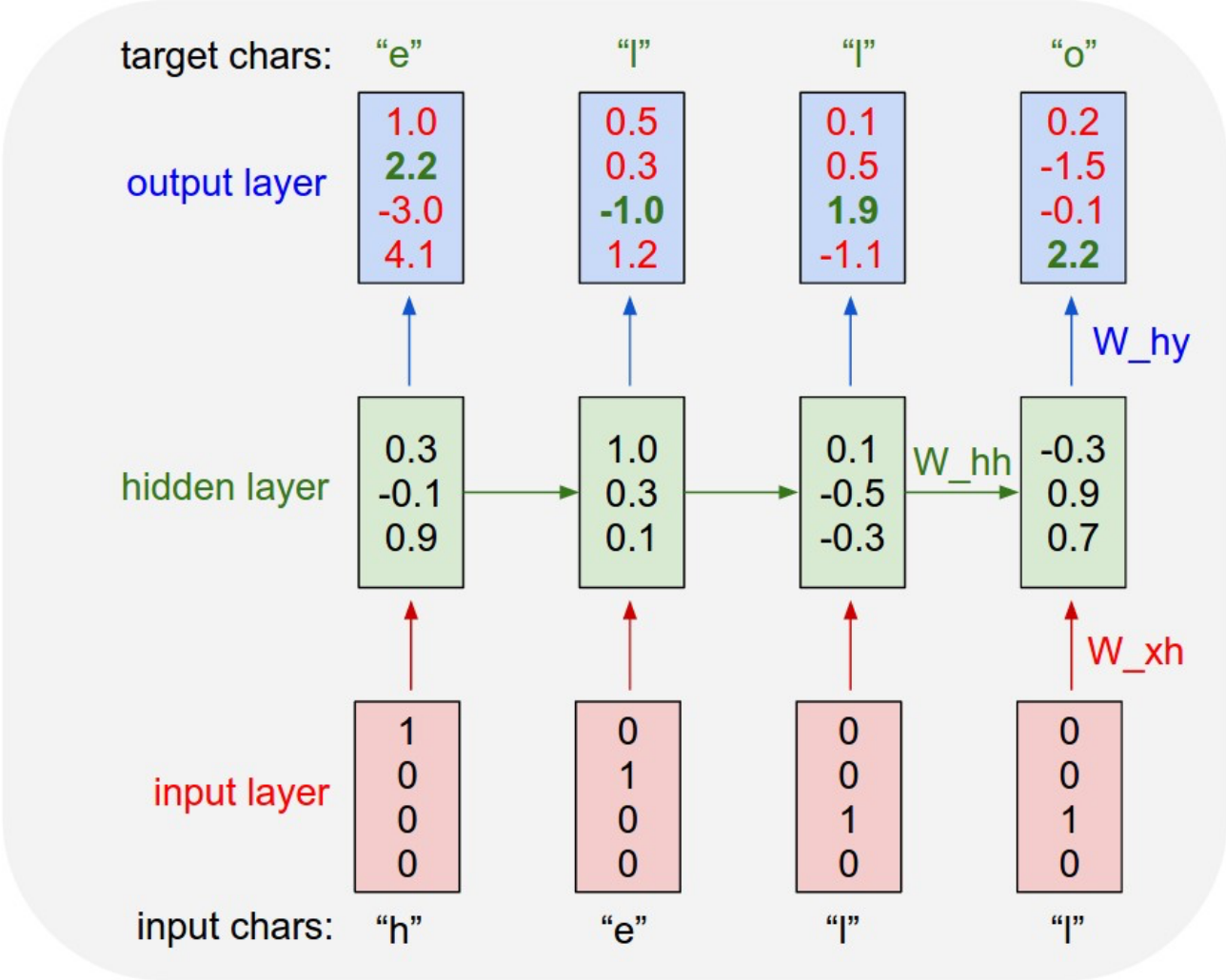
Example: Character-level Language Modeling

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



Example: Character-level Language Modeling

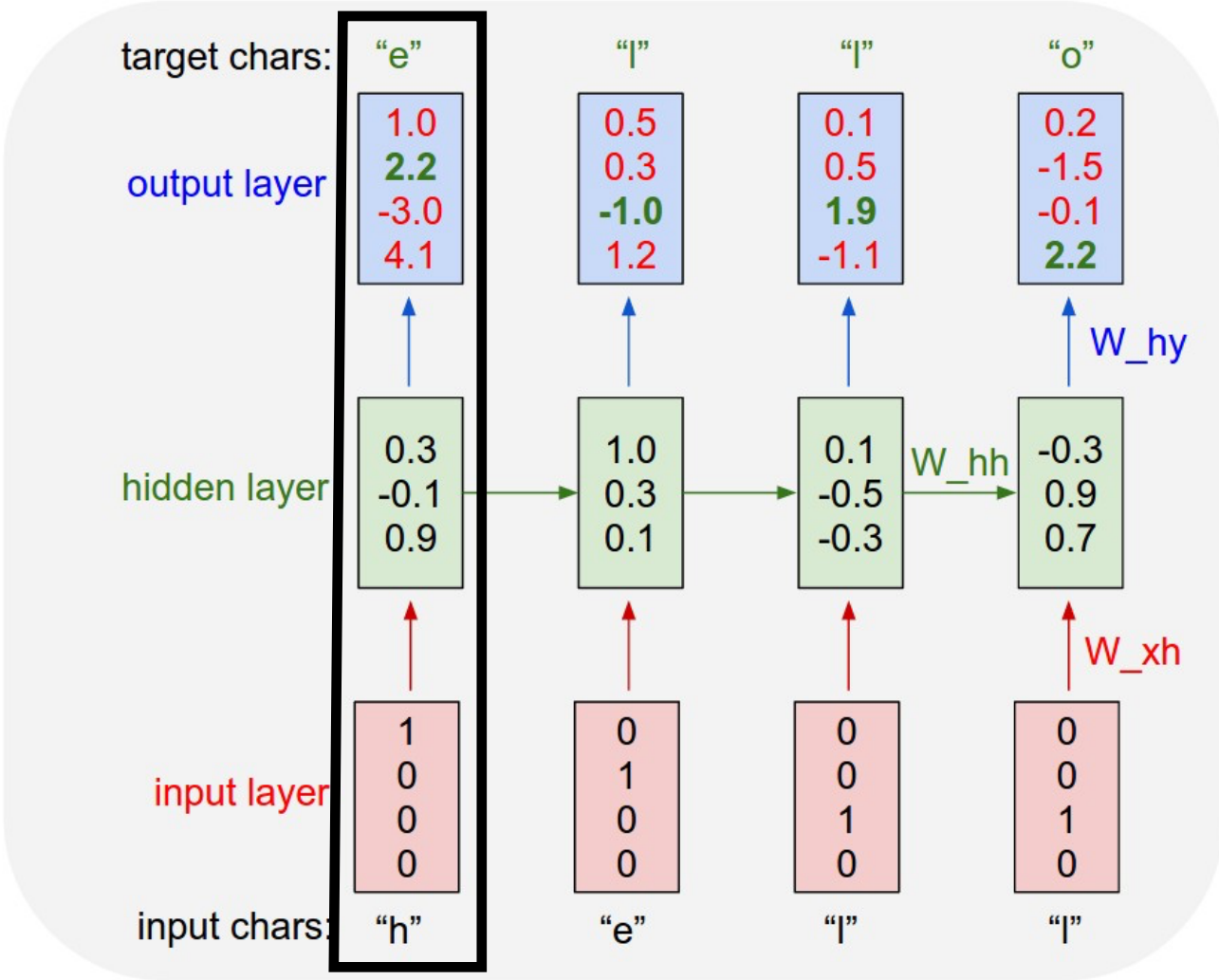
Given “h”, predict “e”

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: ”hello”

Vocabulary: [h, e, l, o]



Example: Character-level Language Modeling

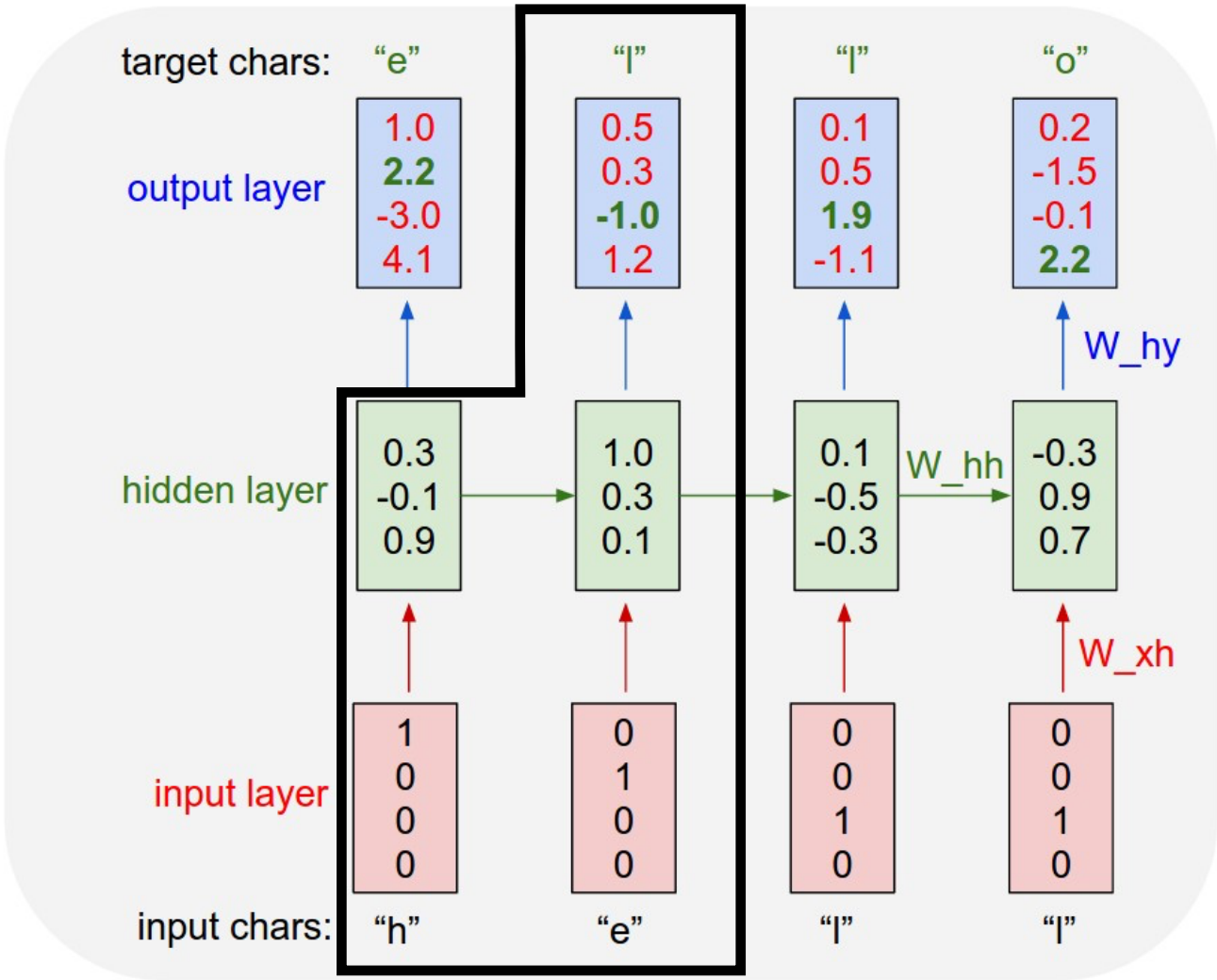
Given "he", predict "l"

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Example: Character-level Language Modeling

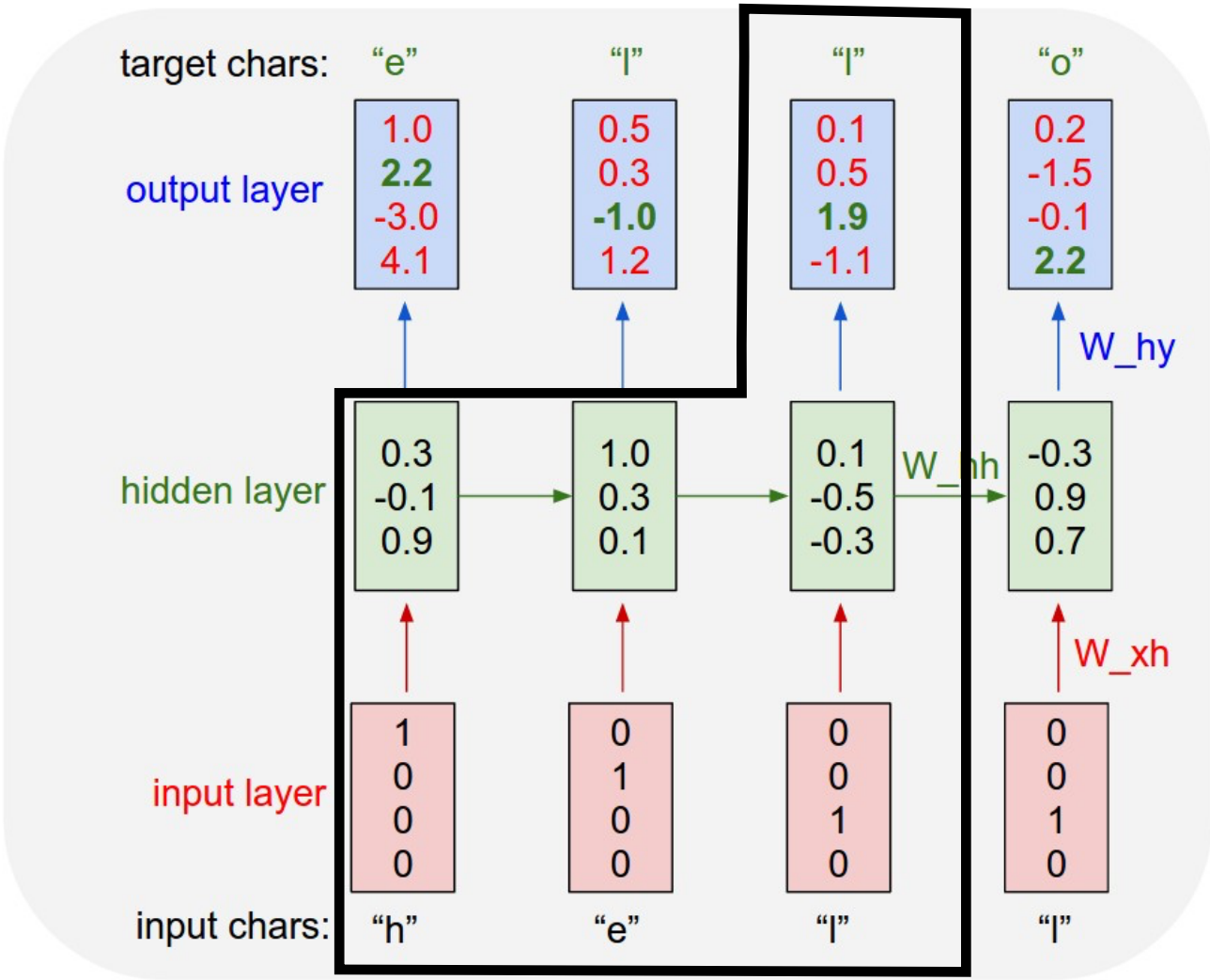
Given "hel", predict "l"

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Example: Character-level Language Modeling

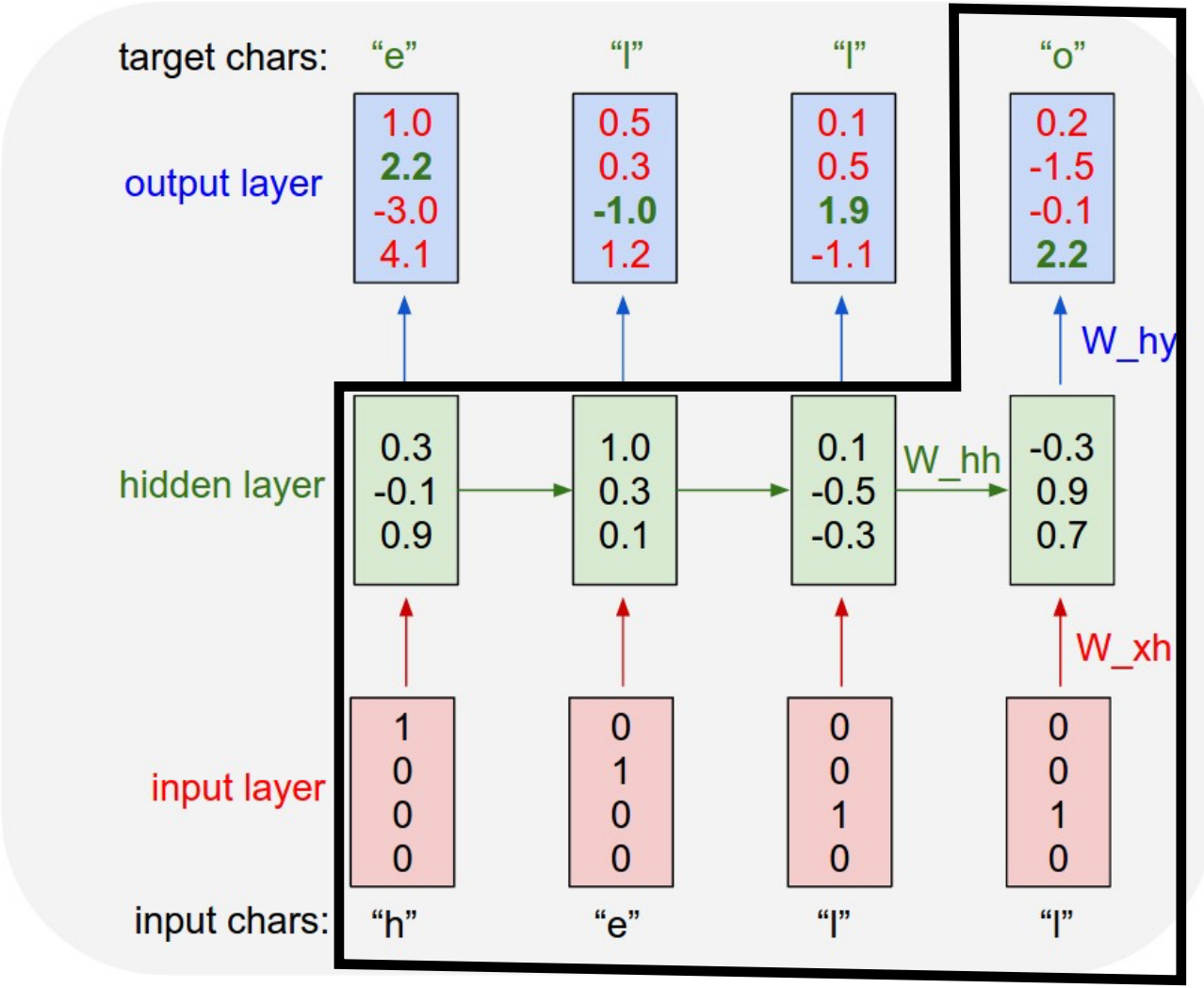
Given "hell", predict "o"

Given Characters 1, 2, …, t,
model predicts character t+1

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]

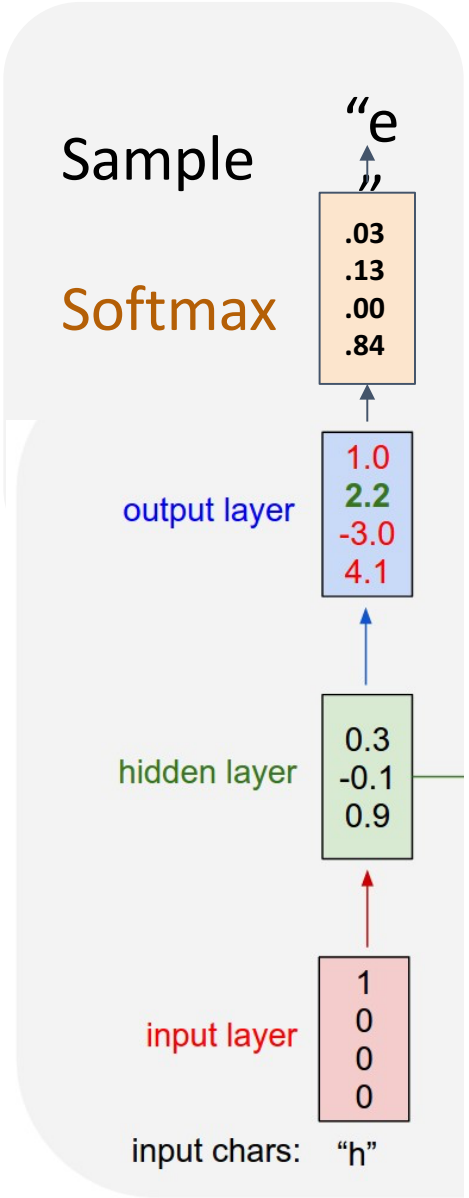


Example: Character-level Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

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

Vocabulary: [h, e, l, o]

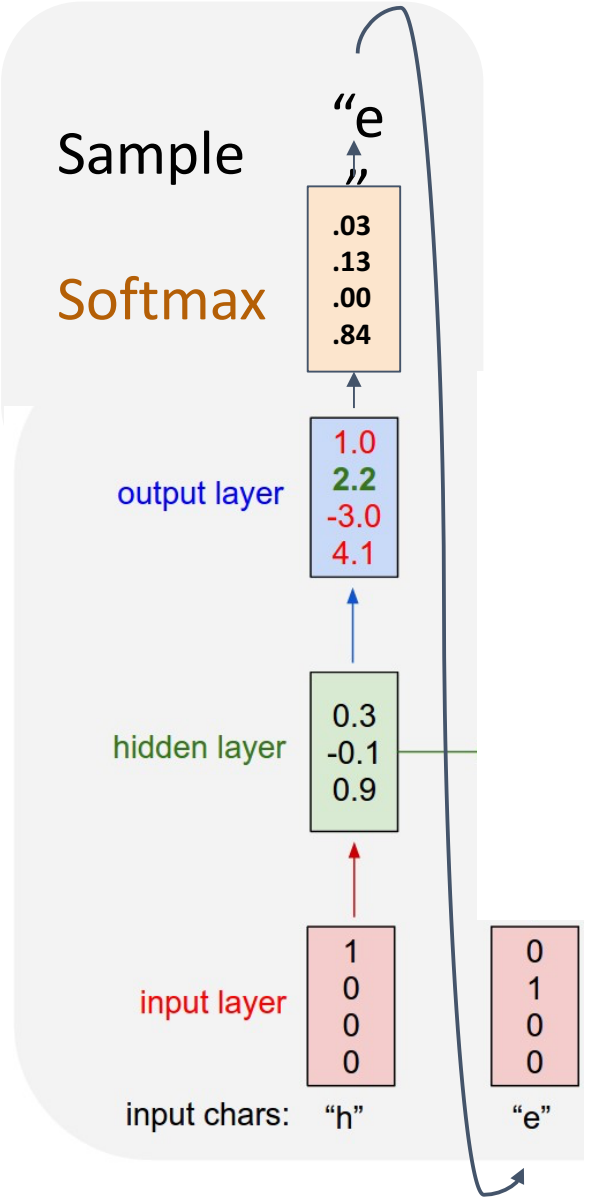


Example: Character-level Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

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

Vocabulary: [h, e, l, o]

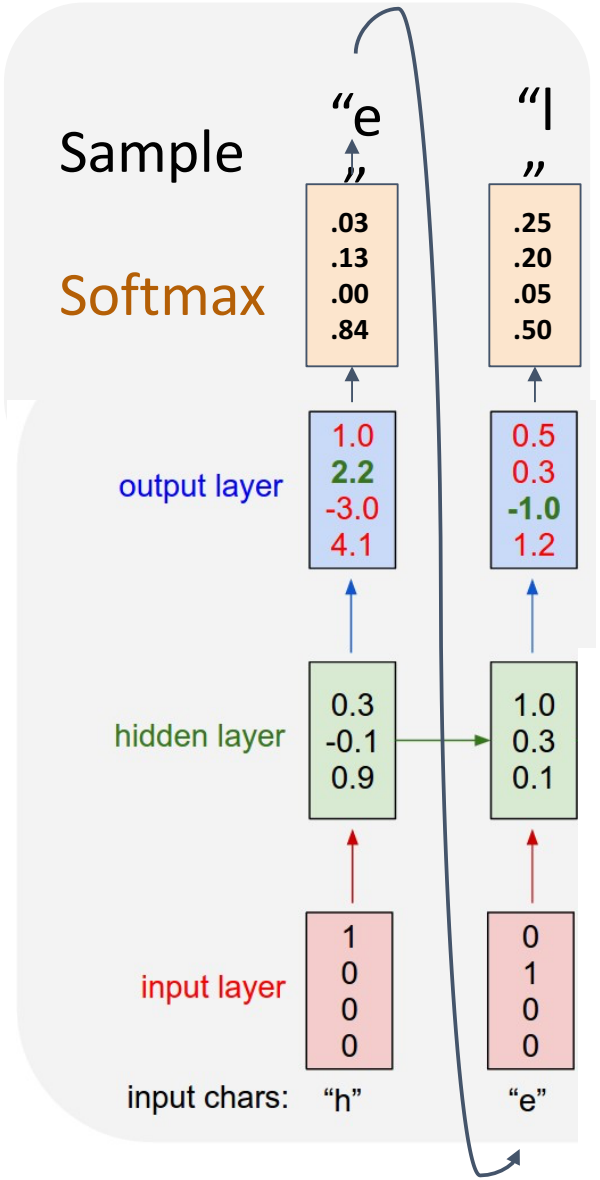


Example: Character-level Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

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

Vocabulary: [h, e, l, o]

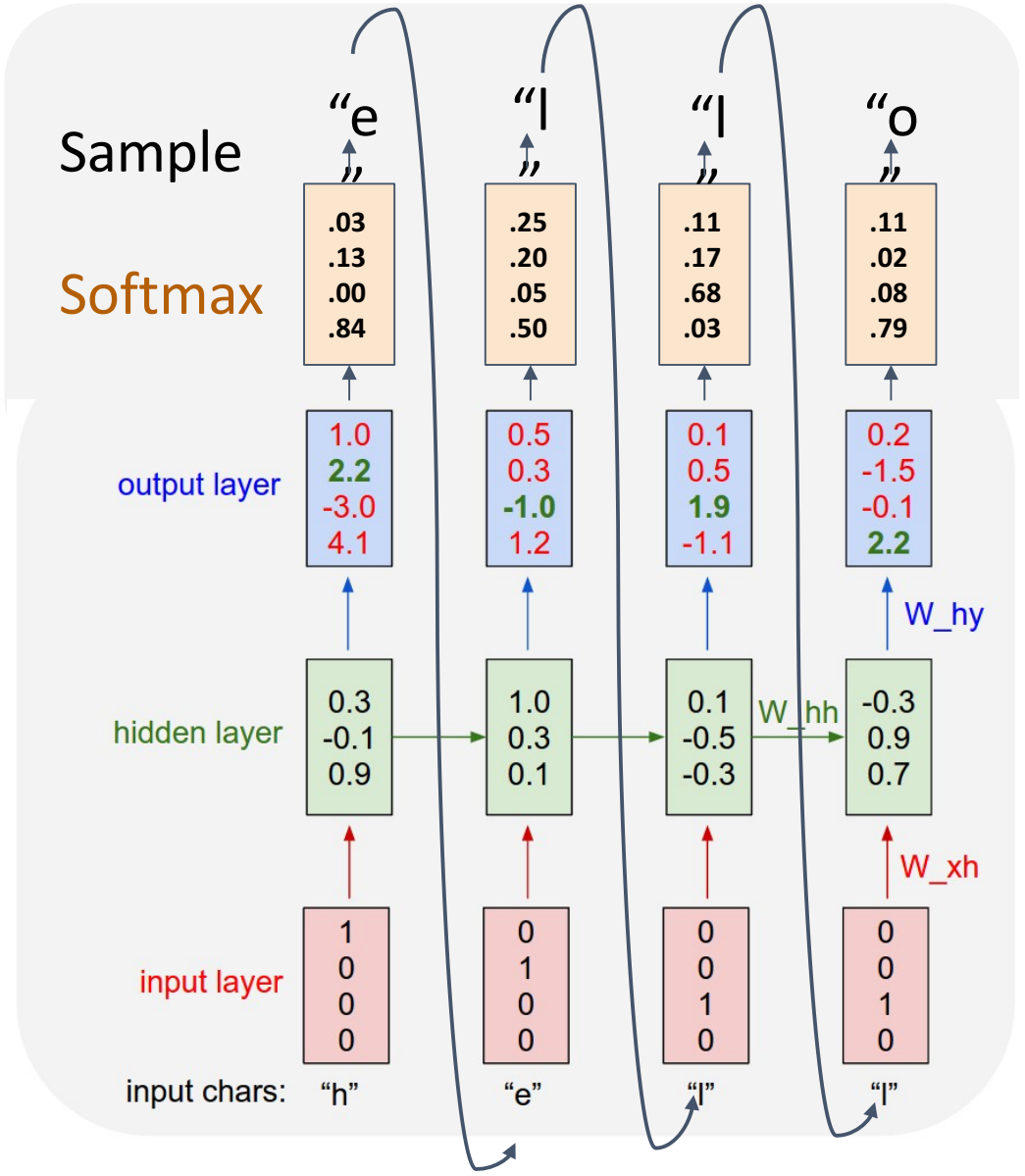


Example: Character-level Language Modeling

At test-time, **generate** new text: sample characters one at a time, feed back to model

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

Vocabulary: [h, e, l, o]

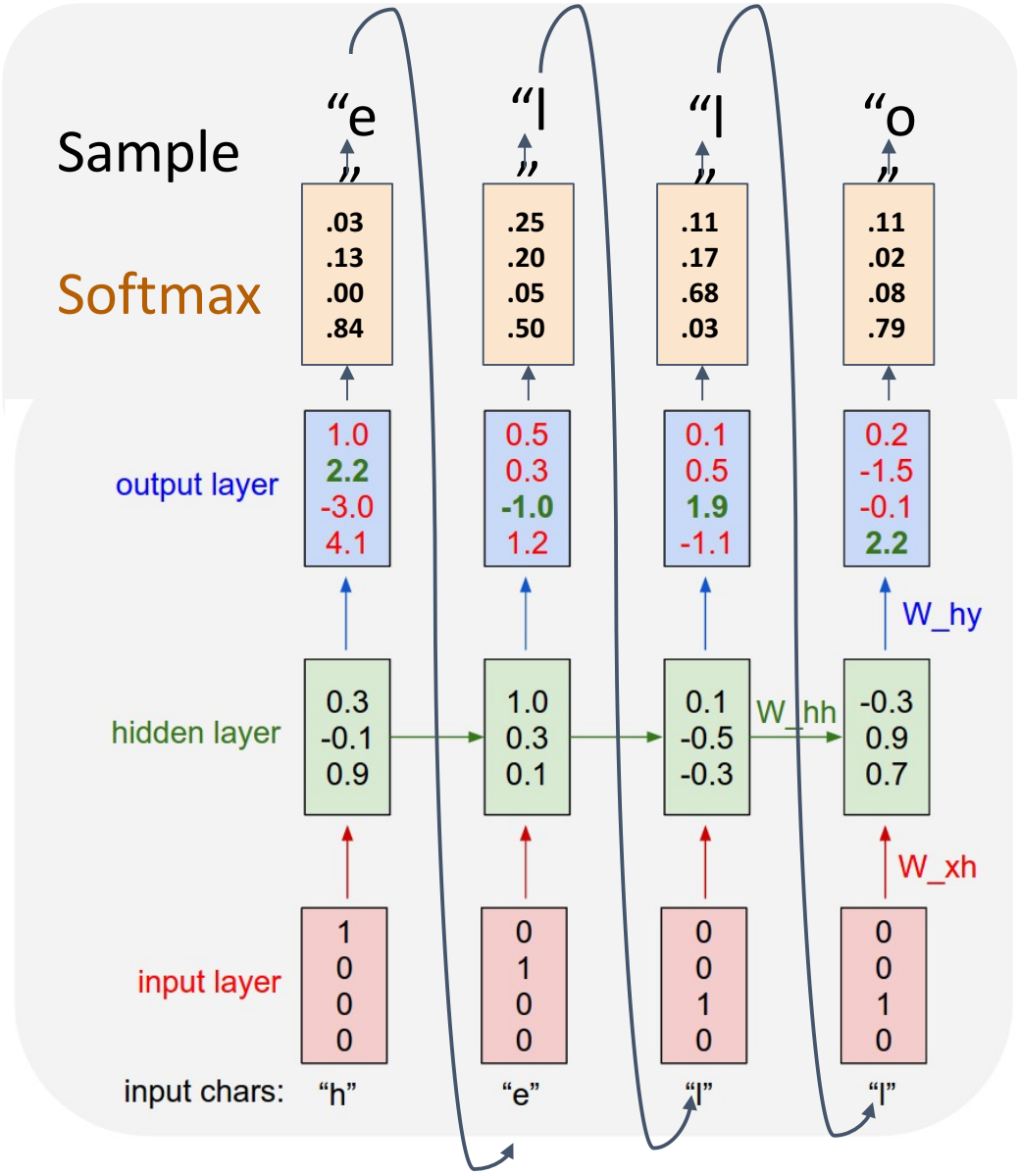


Example: Character-level Language Modeling

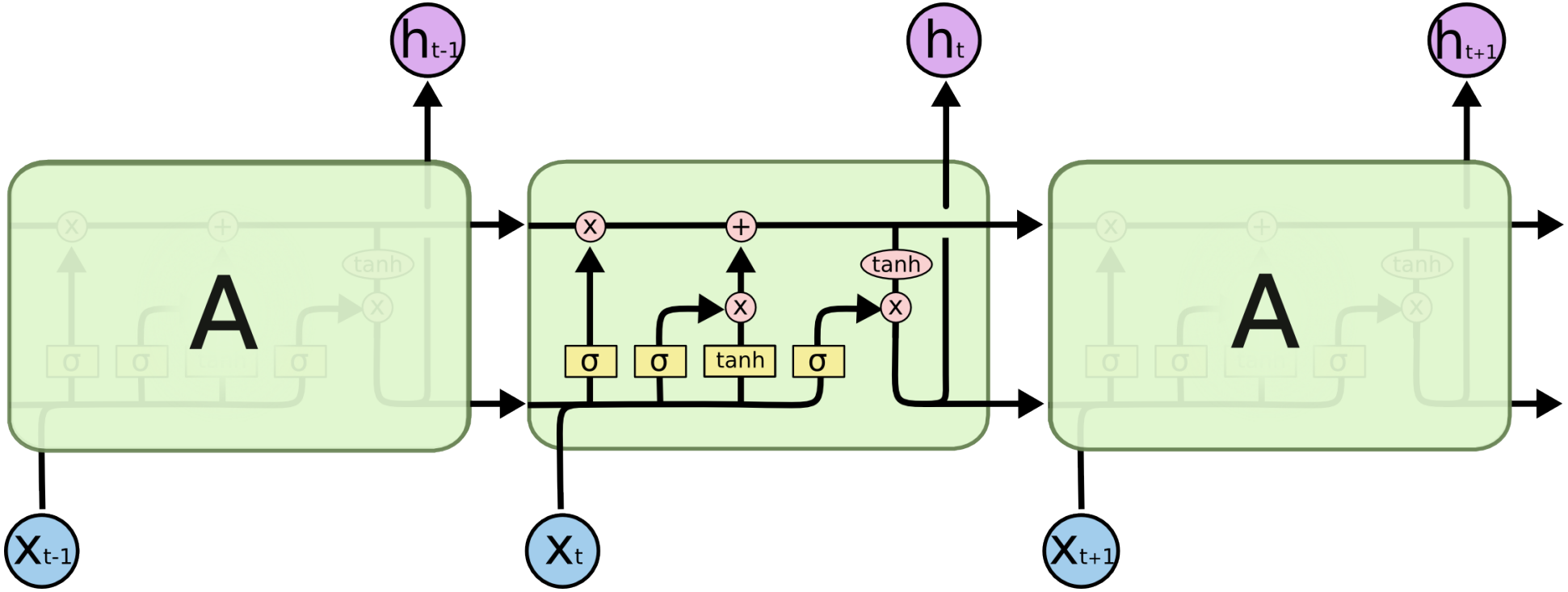
So far: encode inputs
as **one-hot-vector**

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{14} \\ w_{31} & w_{32} & w_{33} & w_{14} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix}$$

Matrix multiply with a one-hot vector just extracts a column from the weight matrix. Often extract this into a separate **embedding** layer



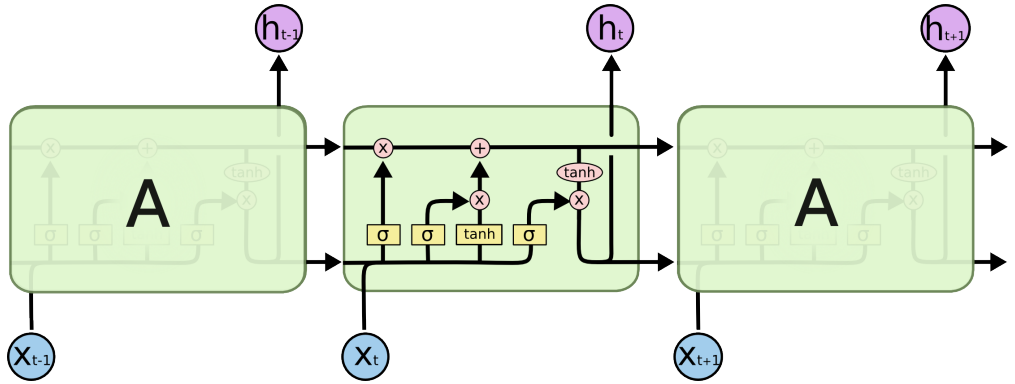
Long Short Term Memory (LSTM)



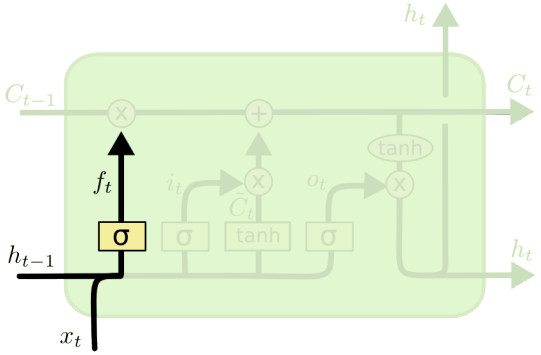
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

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

Gate Functions in Long Short Term Memory (LSTM)



1. Forget Gate: How much information to be forgot from the cell

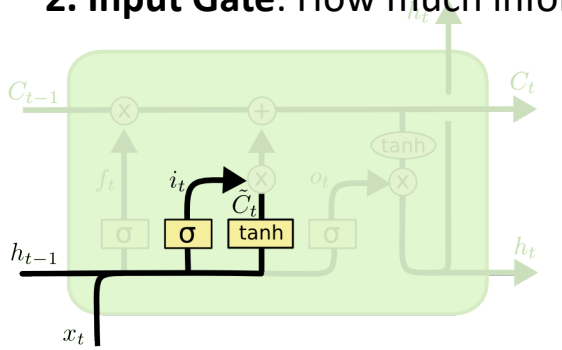


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

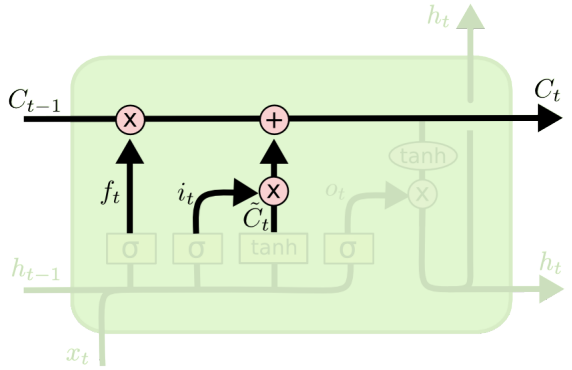
Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation

2. Input Gate: How much information to be put into the cell



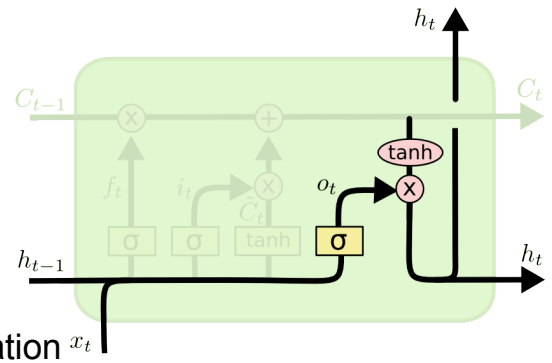
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



3. Update the Cell

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



4. Output Gate: What to be output from the Cell

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Applications of Recurrent Neural Networks

- Named Entity Recognition
  - Recognize all entity names from a given sentence
  - Given a sentence, use RNN, e.g., LSTM, to encode each token and obtain a hidden representation, then classify it into a particular category, e.g., organization (ORG), person (PER), location (LOC), other (O)