Week 6 – Lesson 1: Recurrent Neural Networks (RNNs)

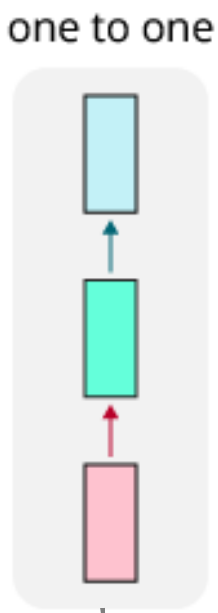
Dr. Hongping Cai

Department of Computer Science

University of Bath

Topic 1:
Types of Sequence Problems

So far: Standard
"Feedforward"
Neural Networks



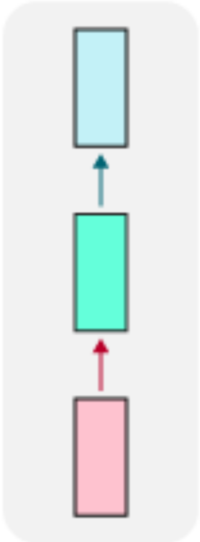
e.g. image classification, house price prediction

Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

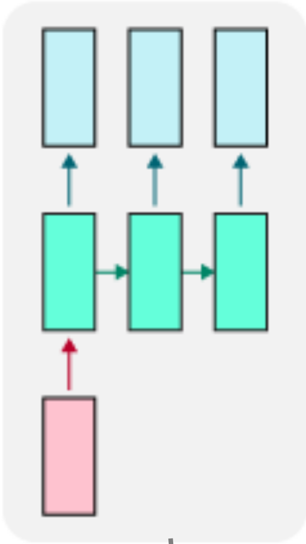
So far: Standard "Feedforward" Neural Networks

Types of Sequence Problems

one to one



one to many

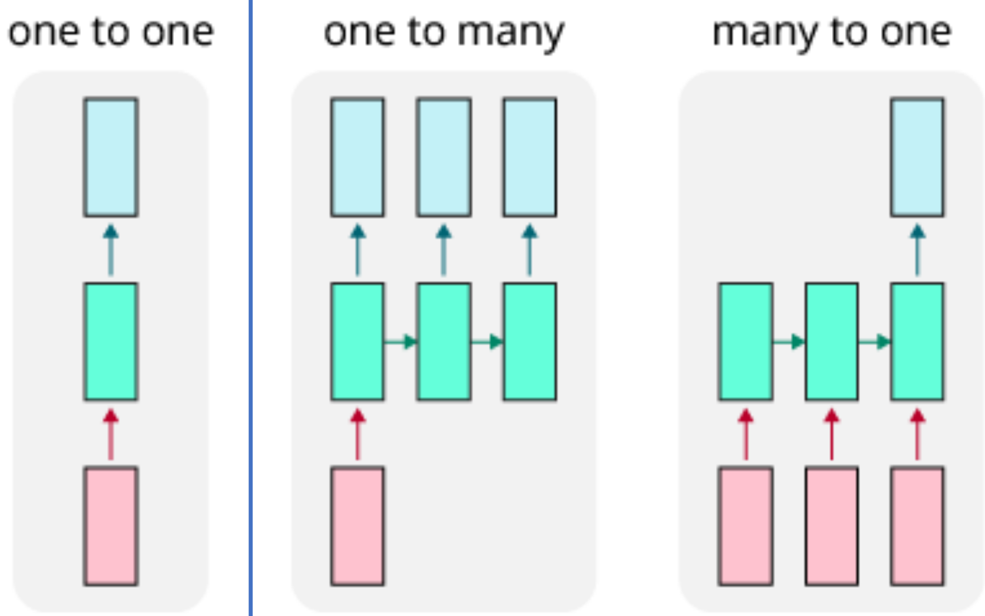


e.g. image captioning

Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard
"Feedforward"
Neural Networks

Types of Sequence Problems



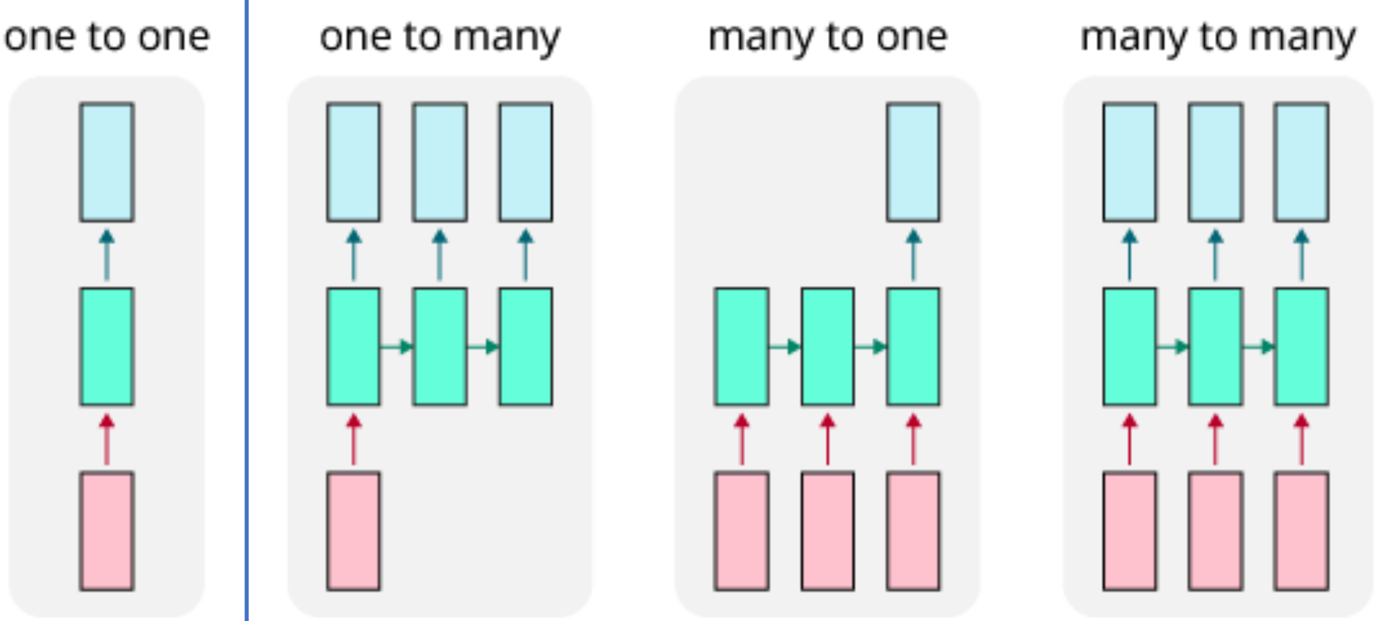
e.g. video classification, sentiment classification

Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard
"Feedforward"
Neural Networks

Types of Sequence Problems

(Input length = output length)



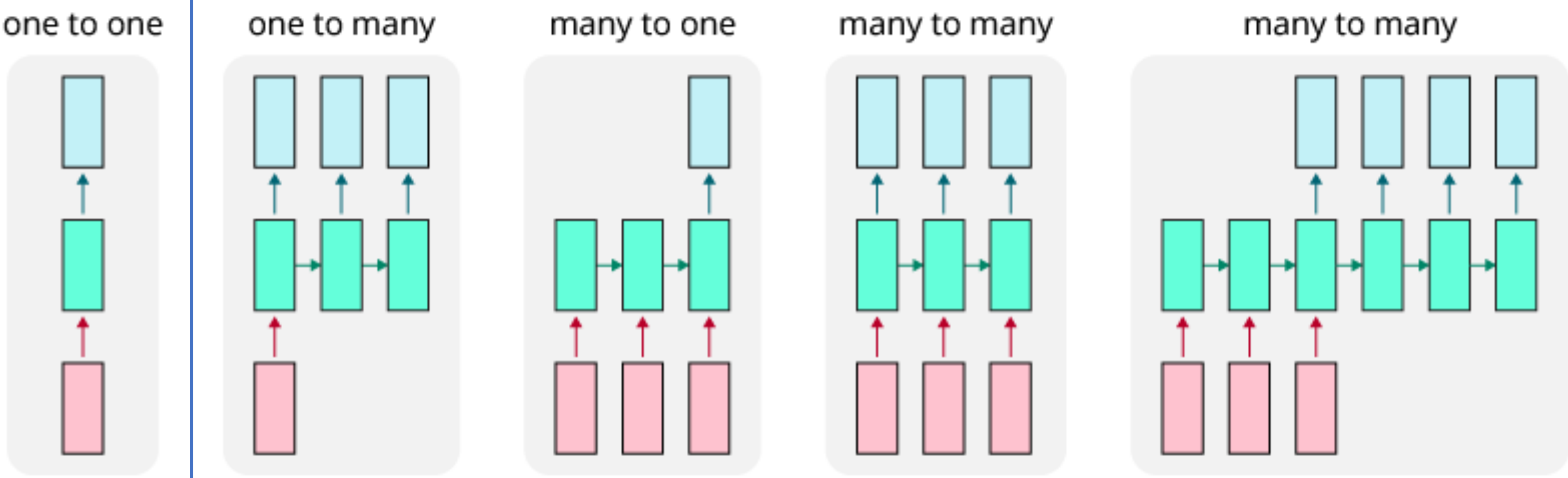
e.g. Per-frame video classification

Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard “Feedforward” Neural Networks

Types of Sequence Problems

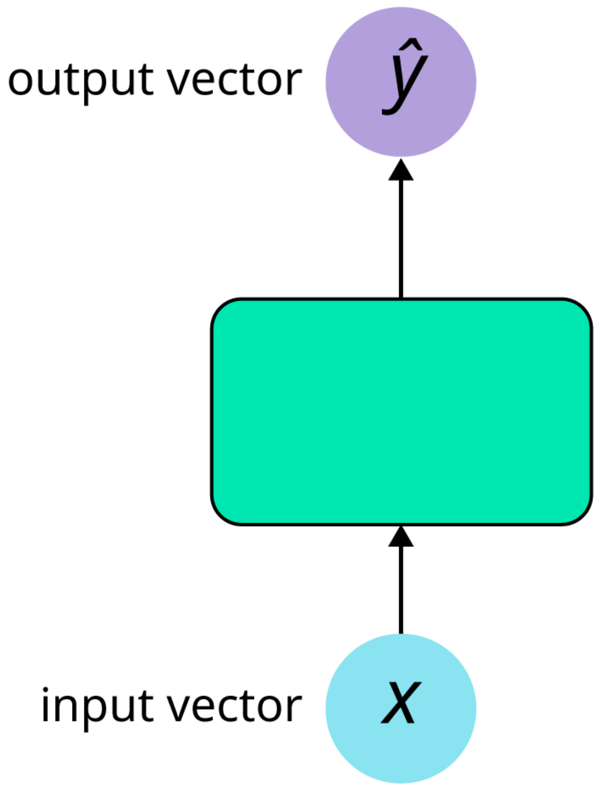
(Input length = output length) (Input length ≠ output length)



e.g. machine translation, chatbots

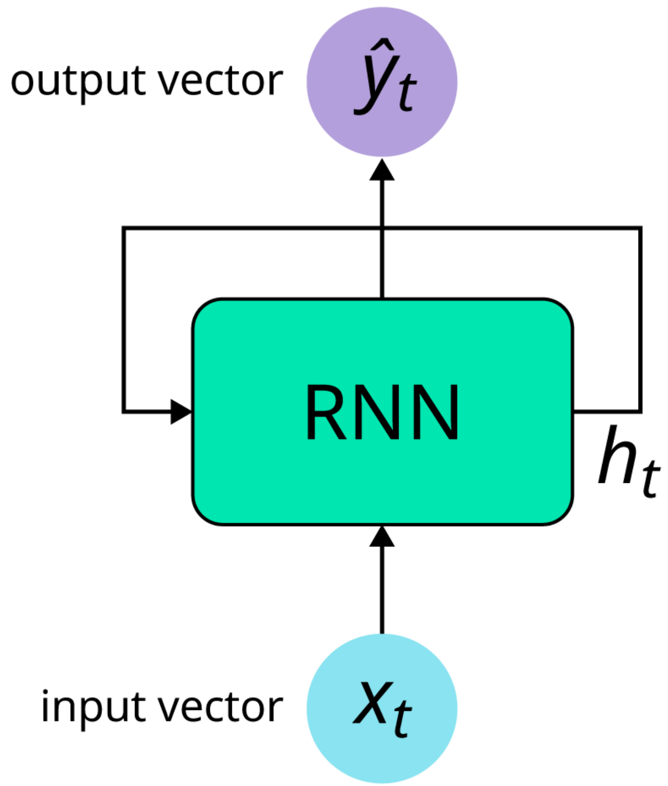
Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard "Feedforward" Neural Networks



Recurrent Neural Networks (RNNs) for sequential modelling

Have an **internal loop**

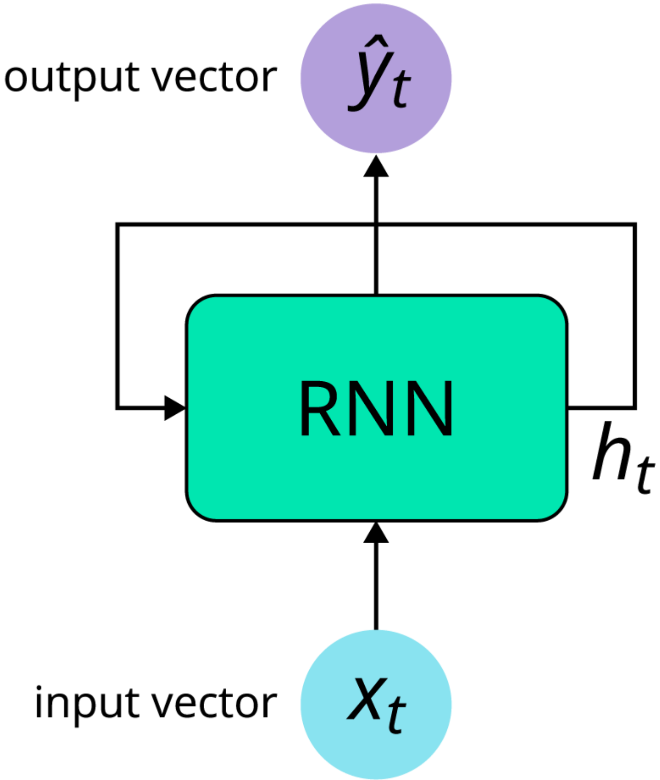


Reference for Topic 1

- Blog by Andrej Karpathy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.

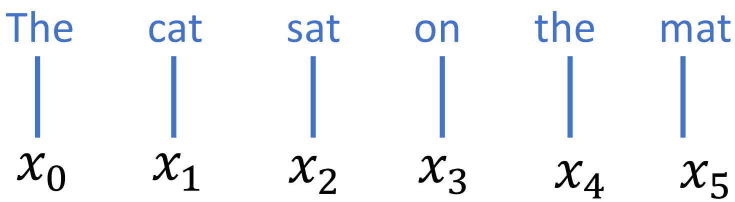
Topic 2:
Recurrent Neural Networks
(RNNs)

Recurrent Neural Networks (RNNs)

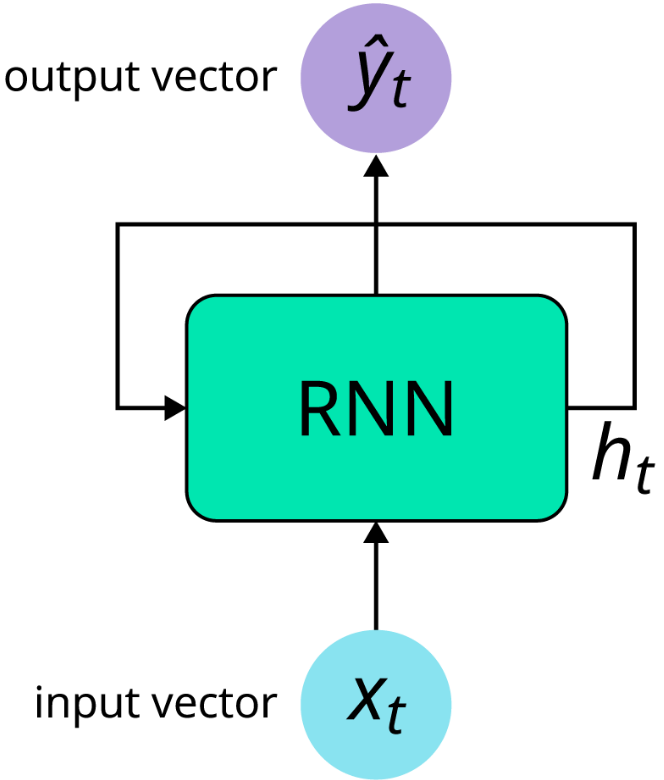


Also called "Vanilla RNNs"

Key idea: RNNs have an **internal/hidden state** h_t that can represent context information.



“Recurrent” Neural Networks (RNNs)



Apply a **recurrence formula** at every time step to update the internal/hidden state:

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

internal state

Function parameterized by W

old state

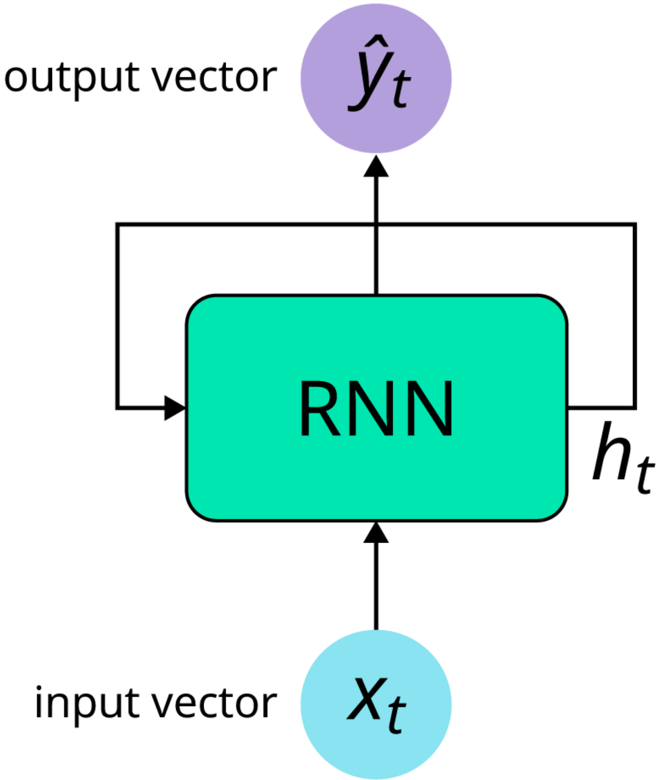
input vector

$$h_{t-1} = f_W(h_{t-2}, x_{t-1})$$

…

$$h_1 = f_W(h_0, x_1)$$

State update and output



Output vector

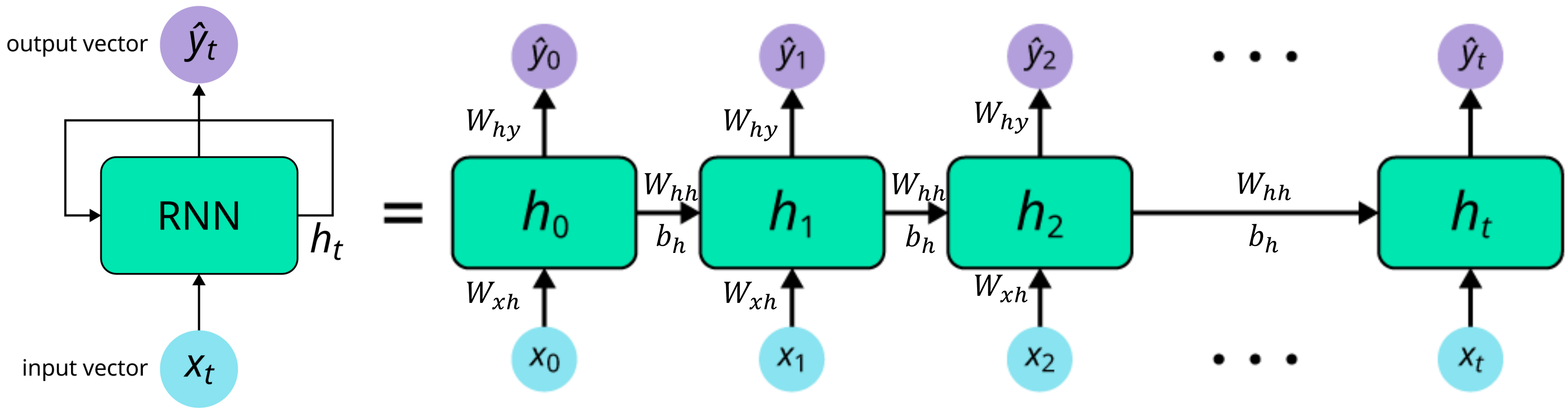
$$\hat{y}_t = W_{hy} h_t$$

Update the hidden state

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

Unfolding an RNN

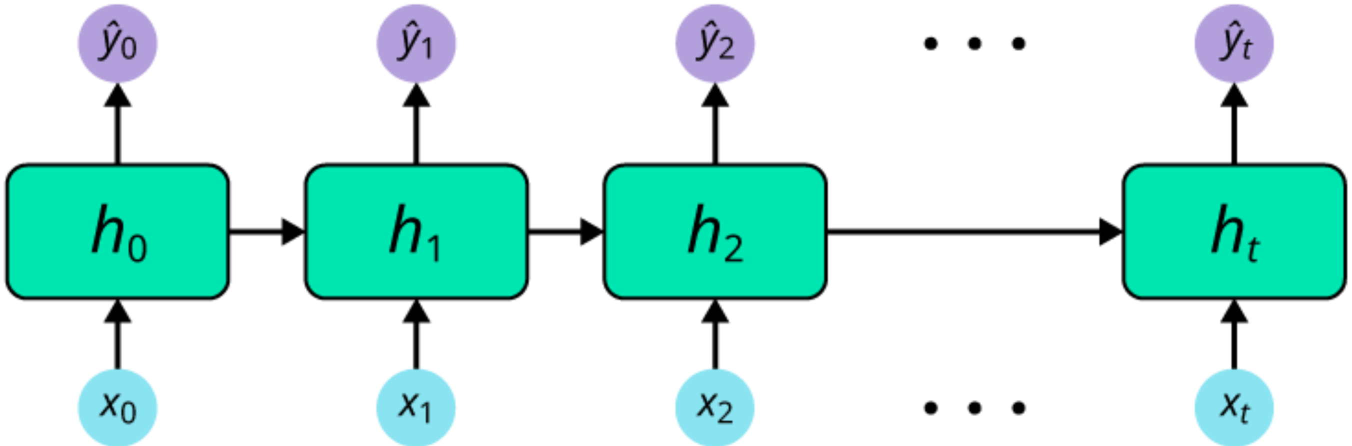
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$
$$\hat{y}_t = W_{hy}h_t$$



The **same weight matrices** are used at every time-step

Summary of RNNs

- Main feature of RNNs is its **hidden state**, considered as the **memory** of the network.
- **Sharing parameters** across all time steps.
- We may not need inputs or output at every time step, depending on the task.

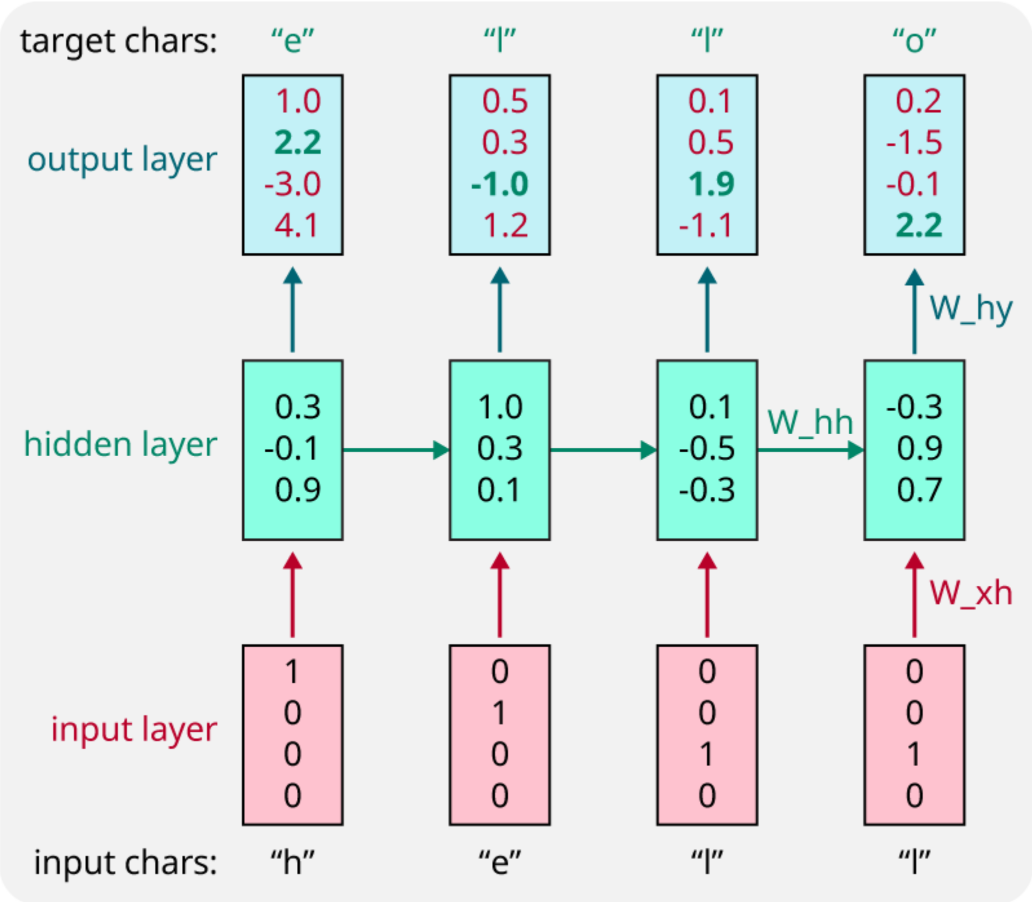


Reference for Topic 2

- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Blog by Andrej Karpathy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Topic 3: A Simple Language-Modelling Example

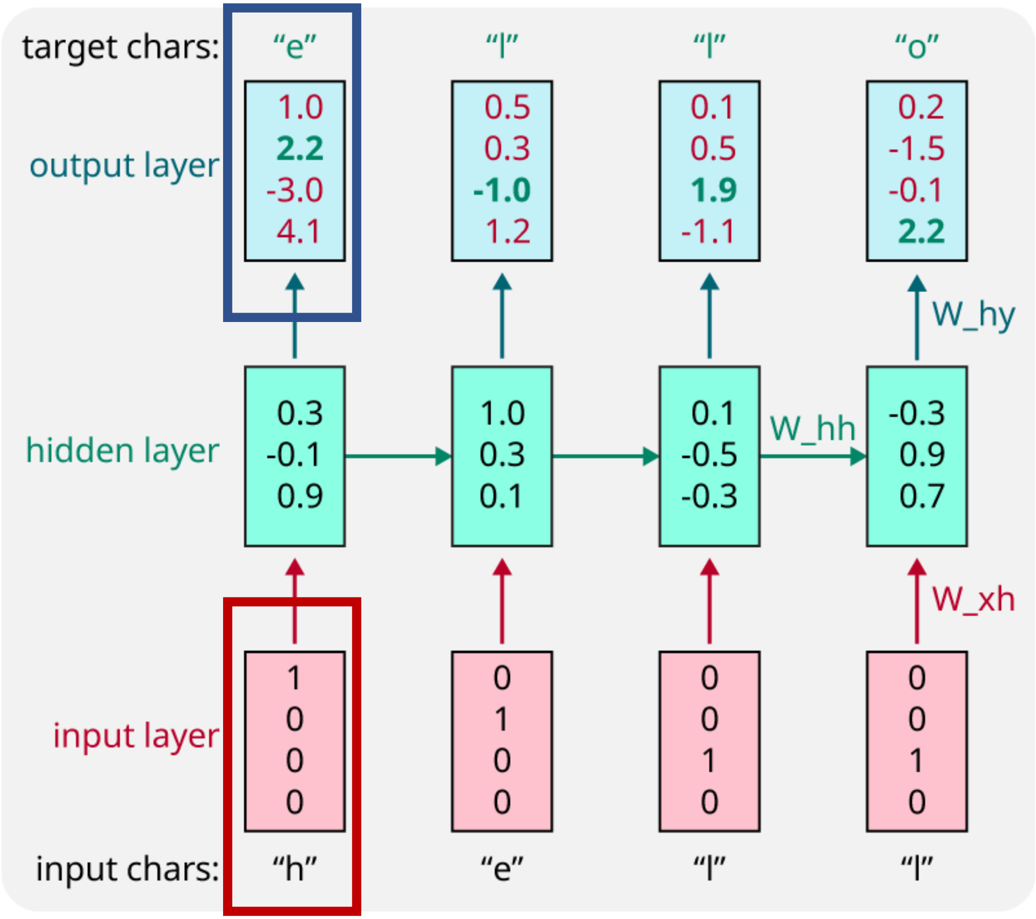
Example: Character-level Language Modelling



Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

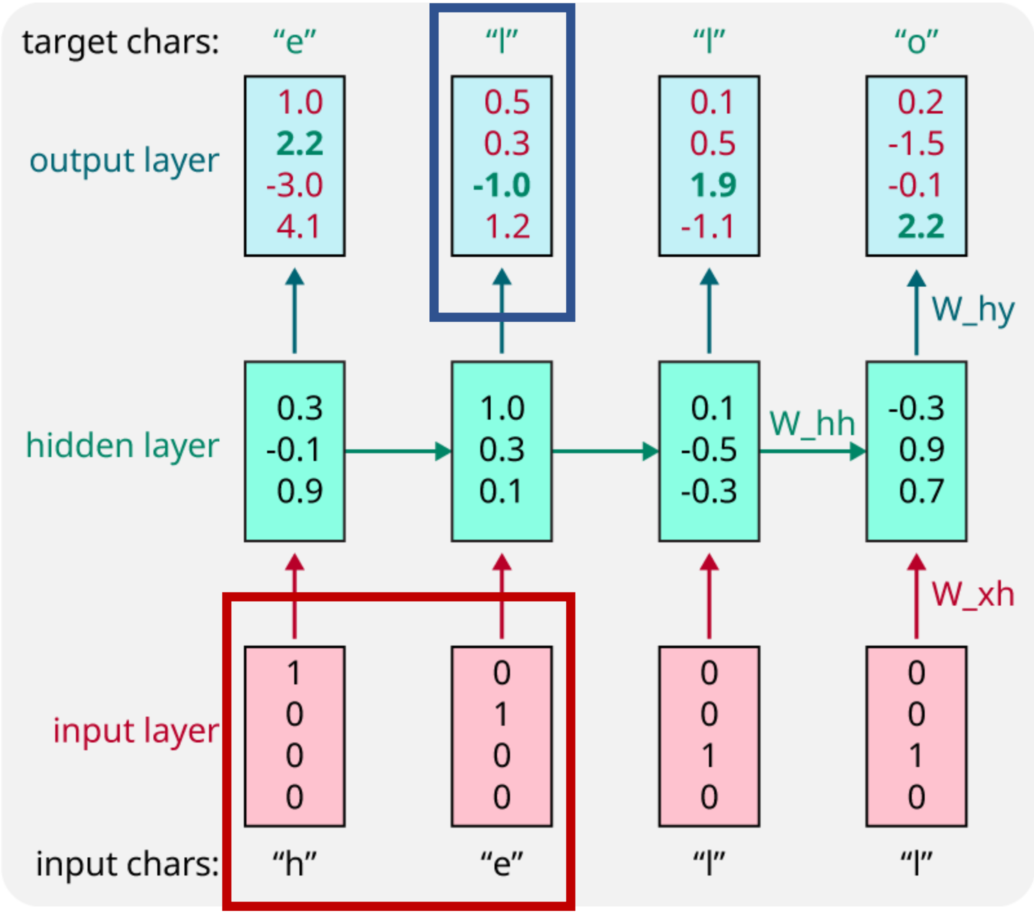
Example from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Given “h”, target output: “e”



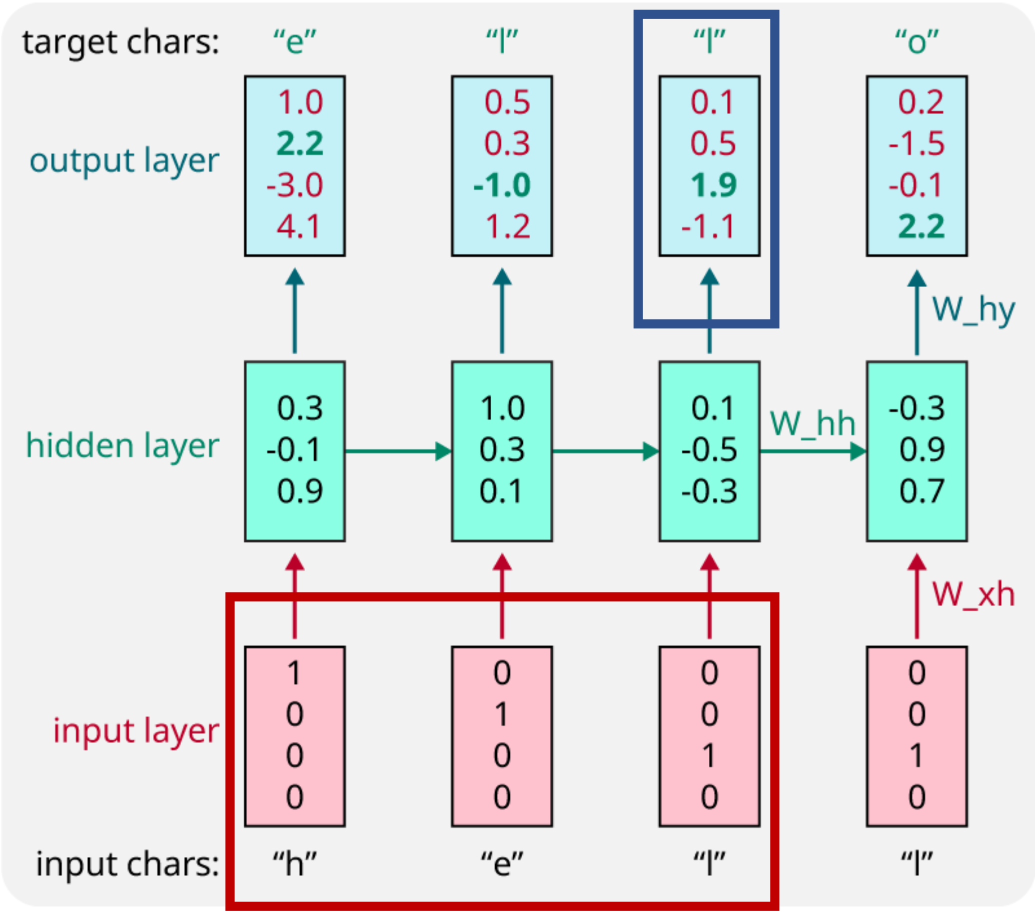
Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

Given “he”, target output: “l”



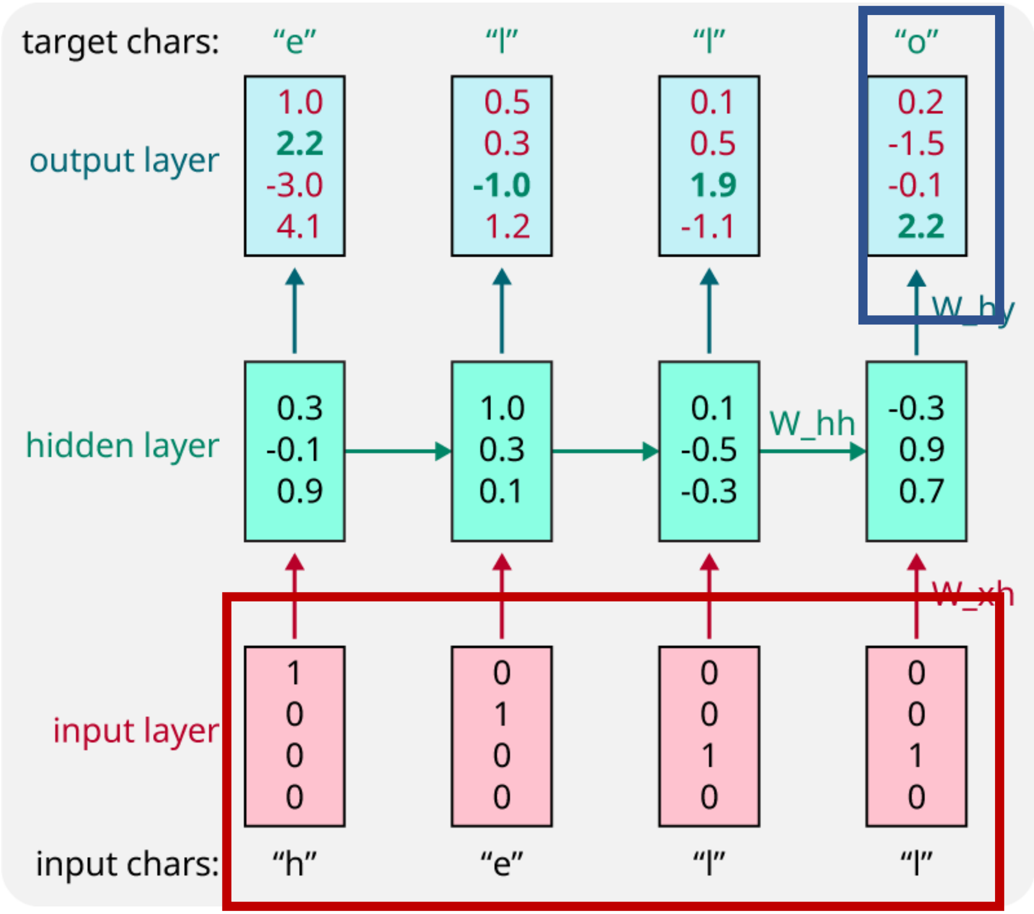
Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

Given “hel”, target output: “l”



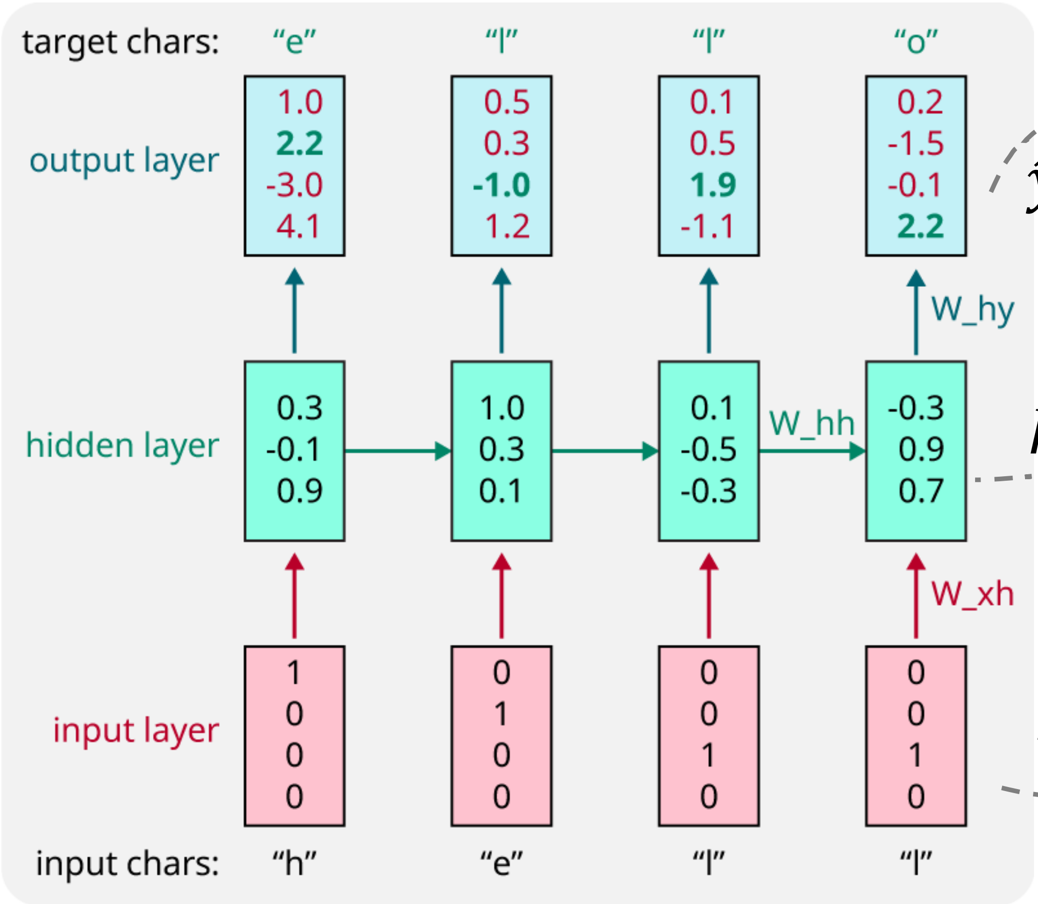
Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

Given “hell”, target output: “o”



Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

Example: Character-level Language Modelling

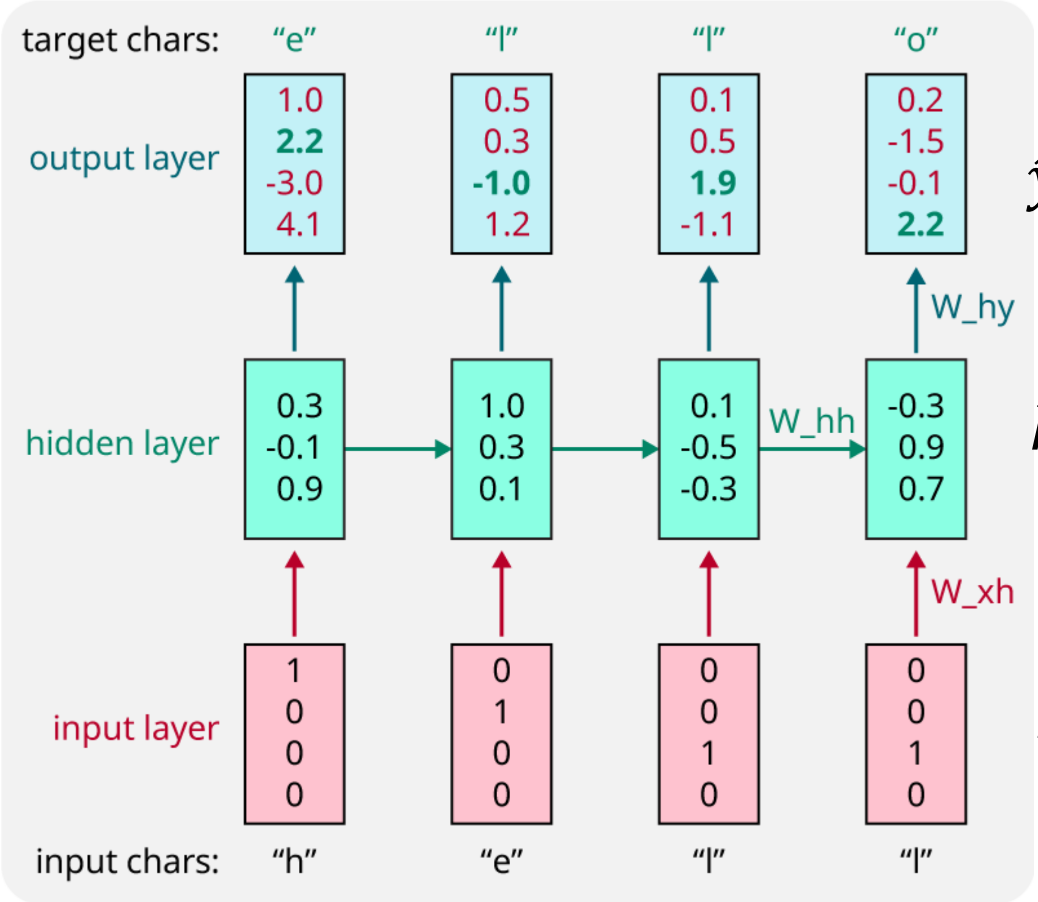


$\hat{y}_t$ The output layer provides **confidences** for the next character. We want the green numbers to be high and red numbers to be low.

h_t We use 3 units for the hidden states

x_t **One-hot** encoding of characters (all zero except for a single one at the index of the character in the vocabulary.)

Example: Character-level Language Modelling

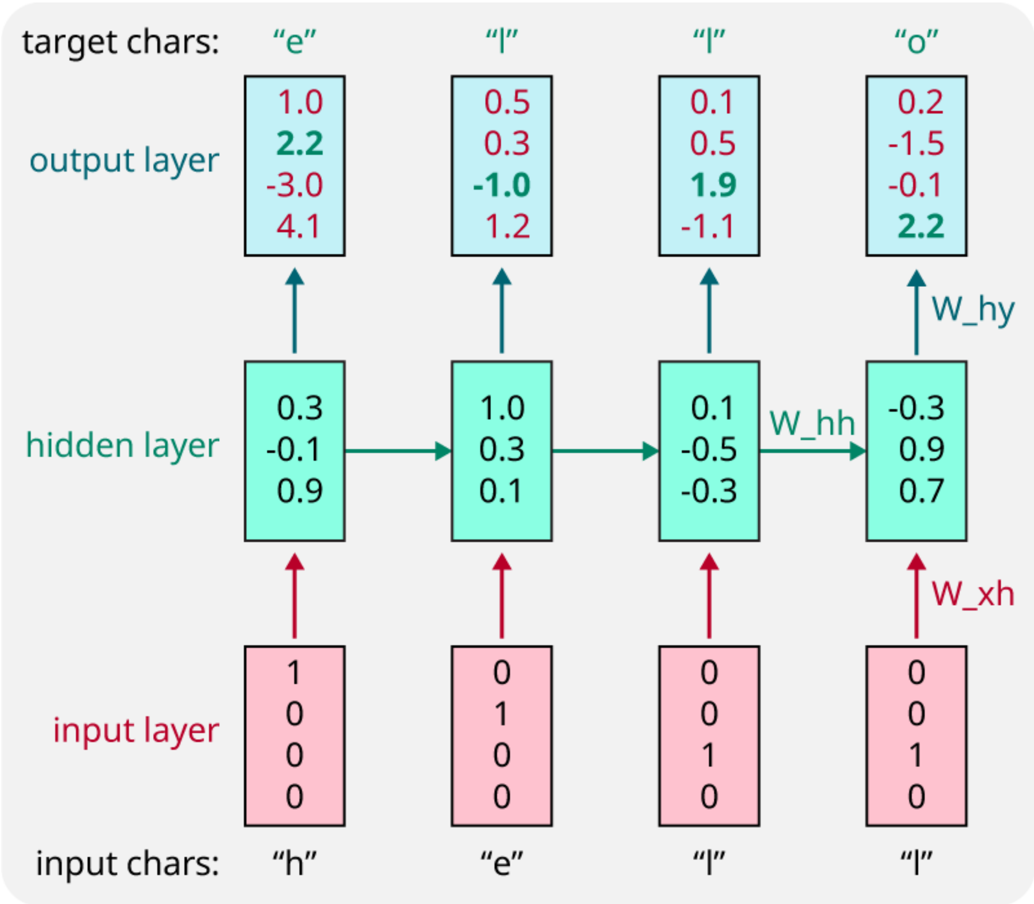


$$\hat{y}_t = W_{hy} h_t$$

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

$$x_t$$

Training process



Initiate the weights

Calculate the loss L.

Calculate the derivatives $\frac{\partial L}{\partial W}$ with **backpropagation through time (BPTT)**

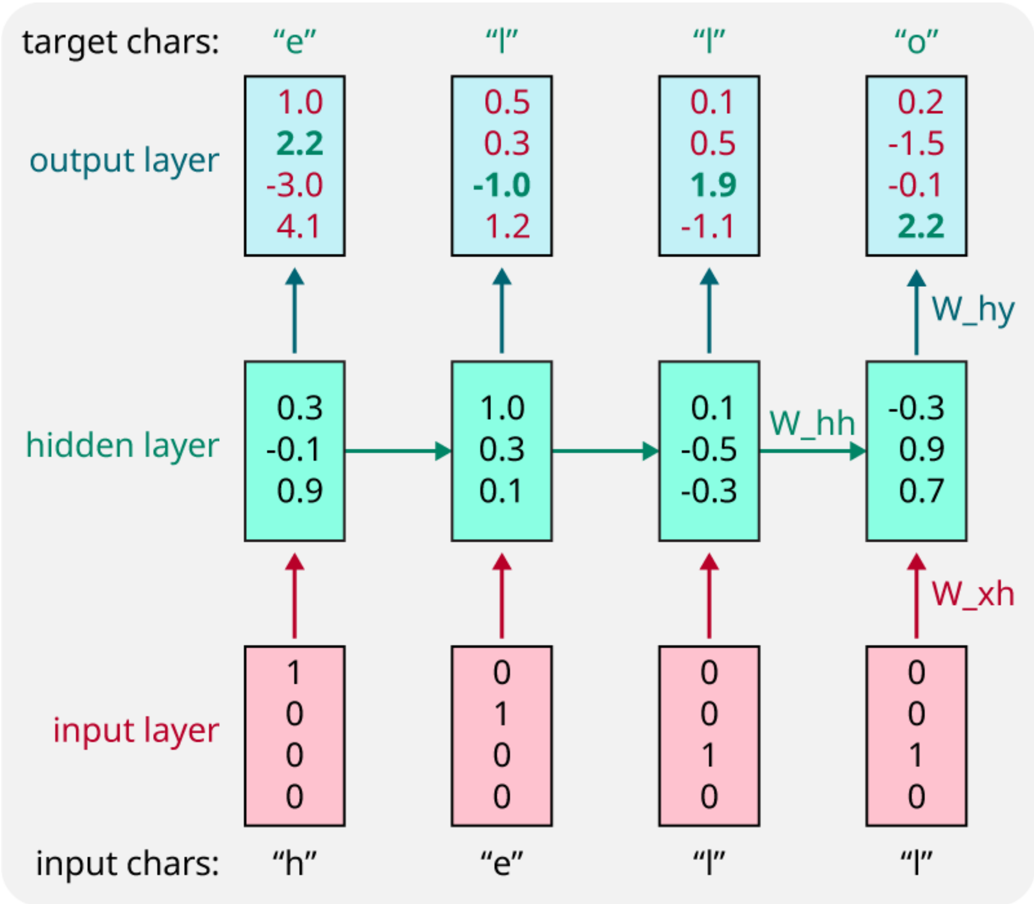
Update the weights
$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

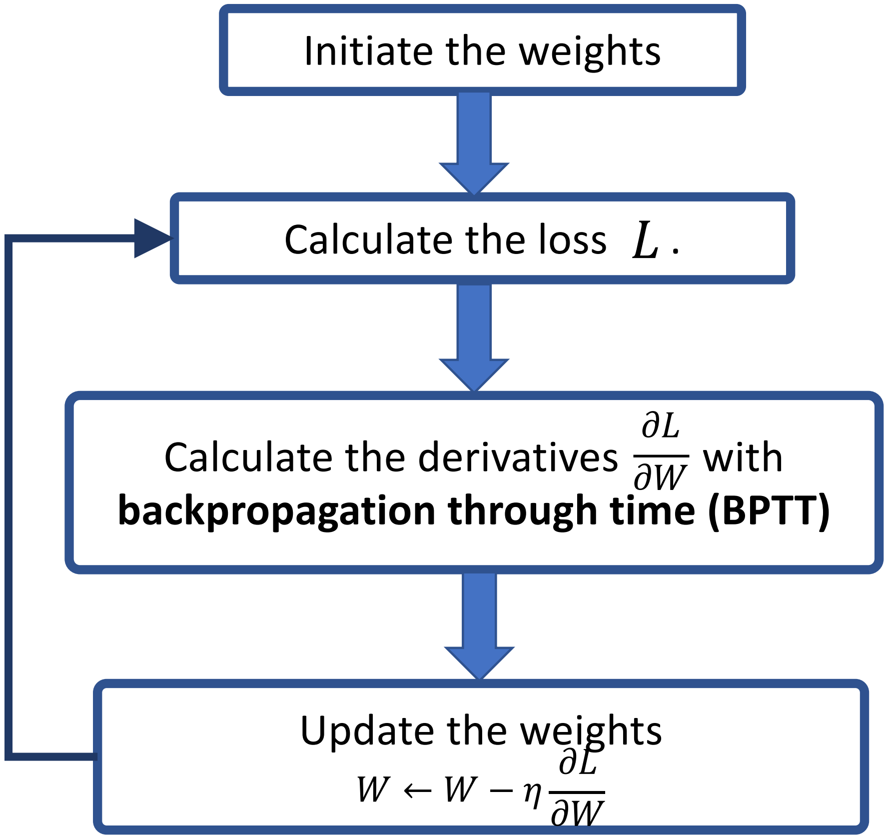
Reference for Topic 3

- Blog by Andrej Karpathy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

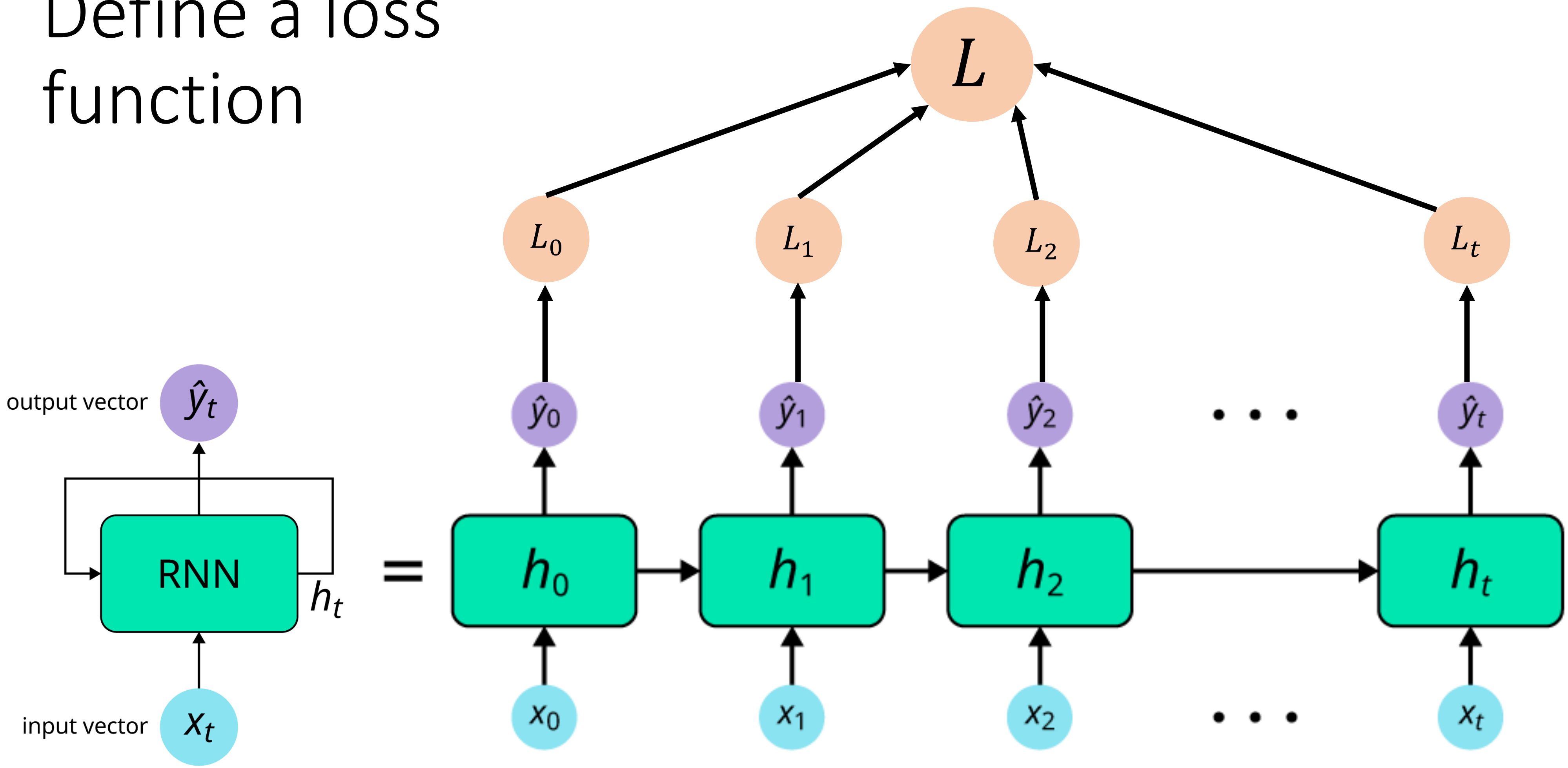
Topic 4: Backpropagation Through Time (BPTT)

Training process





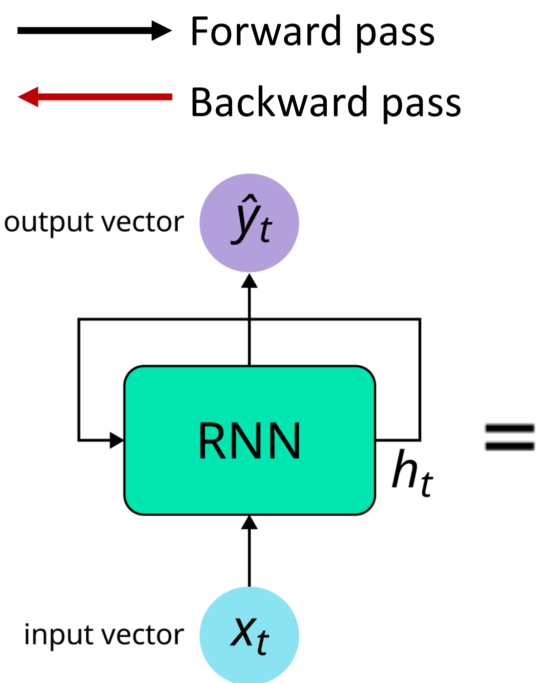
Define a loss function

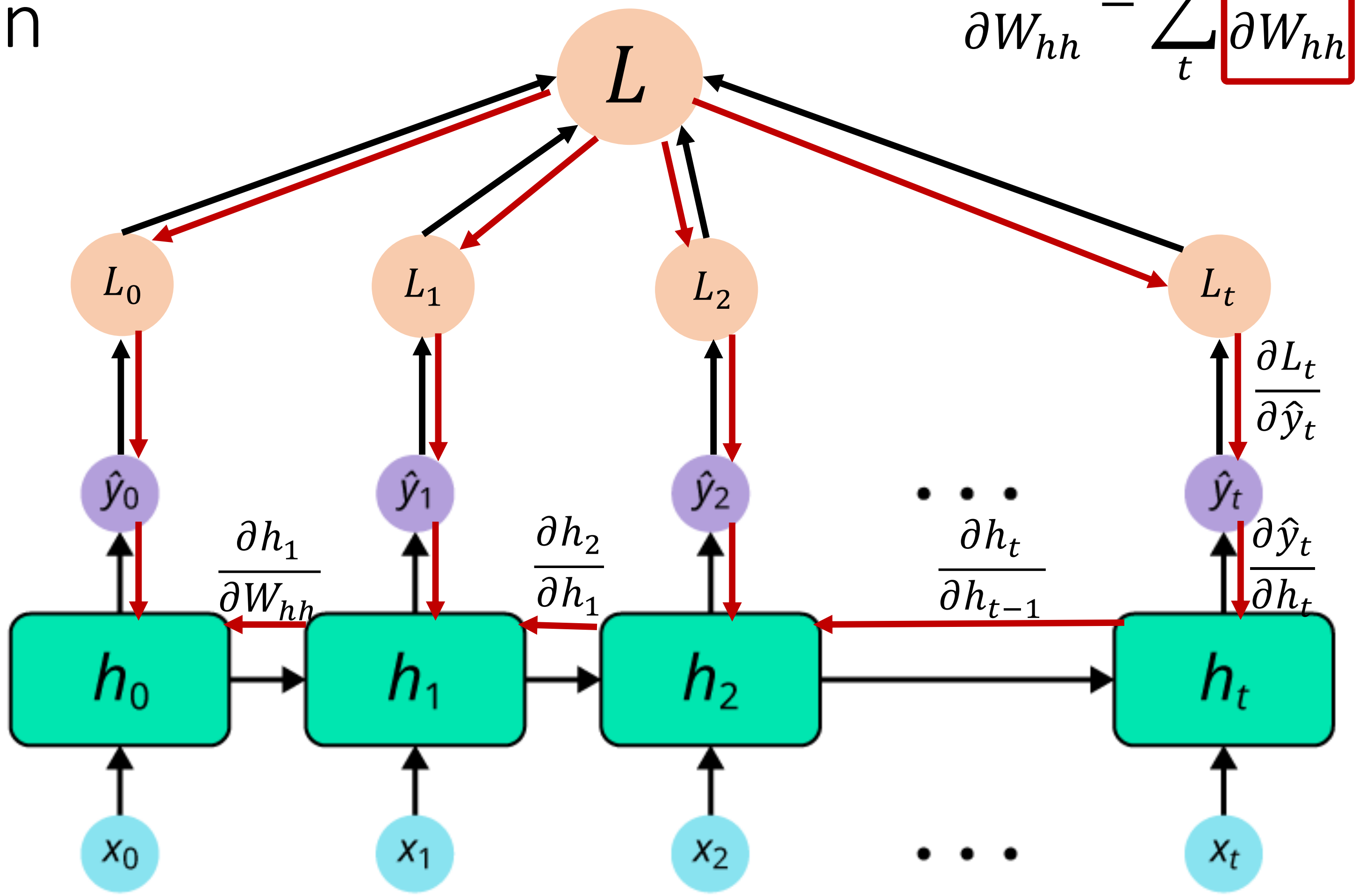


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Backpropagation through time (BPTT)

$$\frac{\partial L}{\partial W_{hh}} = \sum_t \boxed{\frac{\partial L_t}{\partial W_{hh}}}$$





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

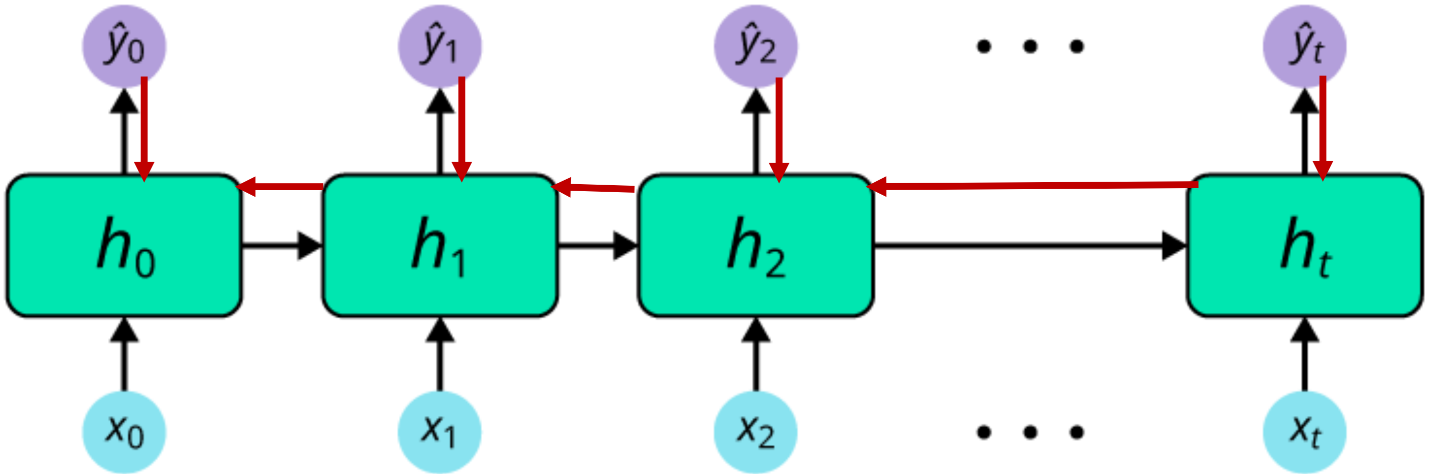
See Weberna's blog for detailed equations: https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html

RNN gradient flow

Many values > 1:
exploding gradients

Gradient clipping: scale big gradients

Computing the gradient wrt W involves multiplying many factors

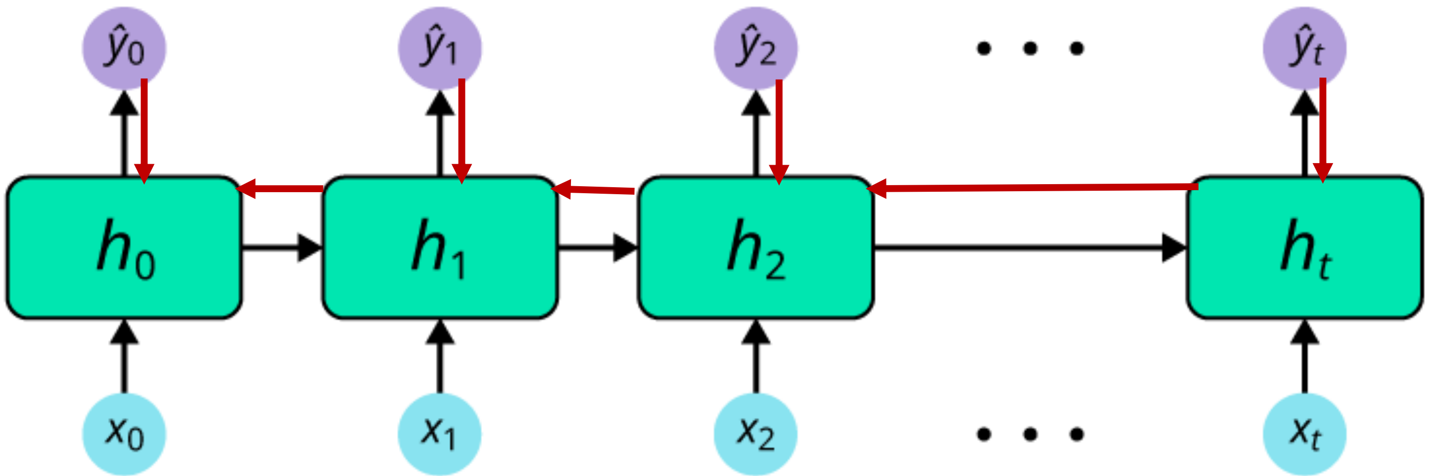


Many values < 1:
vanishing gradients

RNN gradient flow

Many values > 1:
exploding gradients

Computing the gradient wrt W involves multiplying many factors

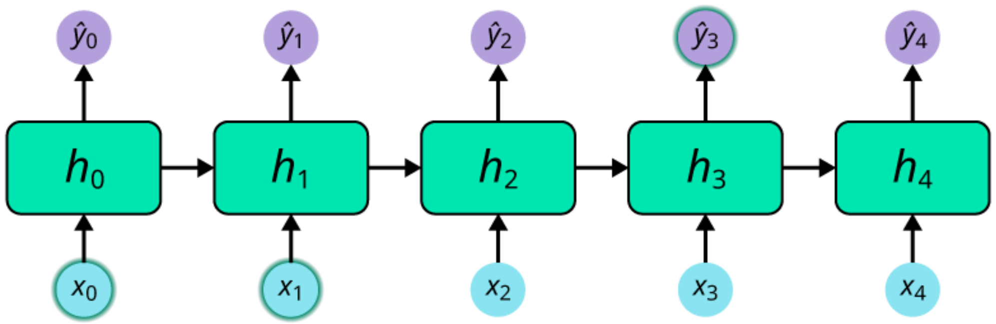


Many values < 1:
vanishing gradients → **Change RNN architecture**

The vanishing gradient problem

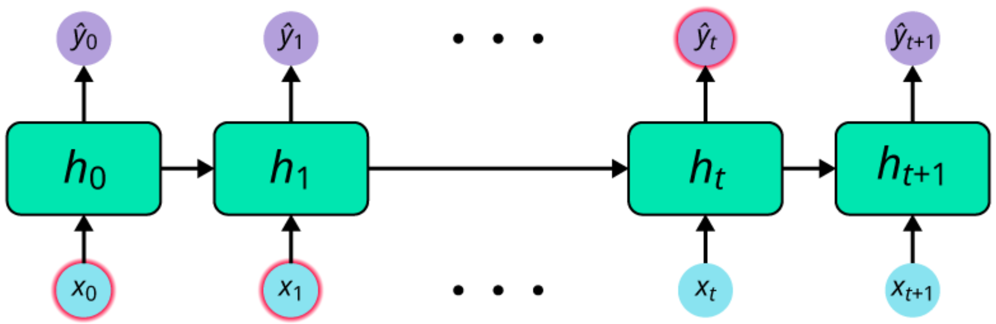
- Short term dependencies:

May I have some water to drink



- Long term dependencies:

It started raining. Mia still played in the garden, with her cloth all wet



Standard RNNs have **difficulties in modelling long-term dependencies** because of vanishing gradient problem.

Solutions

- Key idea: use a **more complex recurrent unit** with **gates** to control the flow of information.
  - Long Short Term Memory (LSTM) ⟵ Next topic
    - ❑Sepp Hochreiter et al., "Long short-term memory", 1997.
  - Gated Recurrent Units (GRU)
    - ❑Cho et al "Learning phrase representations using RNN encoder-decoder for statistical machine translation", 2014

Reference for Topic 4

- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Blog by Denny Brits: http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html
- Blog by Weberna: https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html