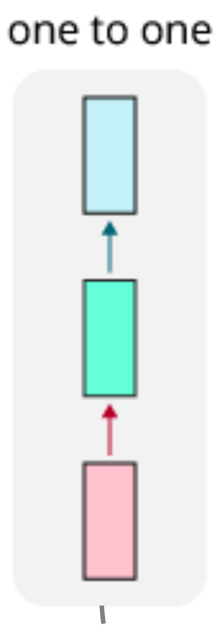
Week 05: Recurrent Neural Networks (RNNs)

Machine Learning 2

Dr. Hongping Cai

Topic 1:
Types of Sequence Problems

So far: Standard
“Feedforward”
Neural Networks

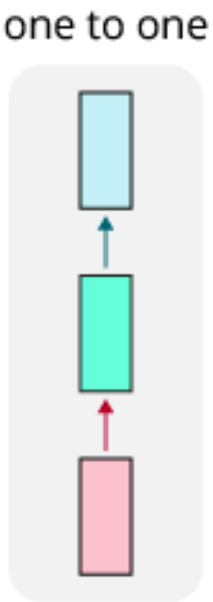


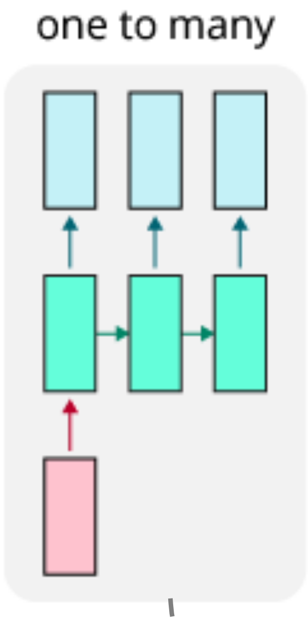
e.g. image classification, house price prediction

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

So far: Standard "Feedforward" Neural Networks

Types of Sequence Problems



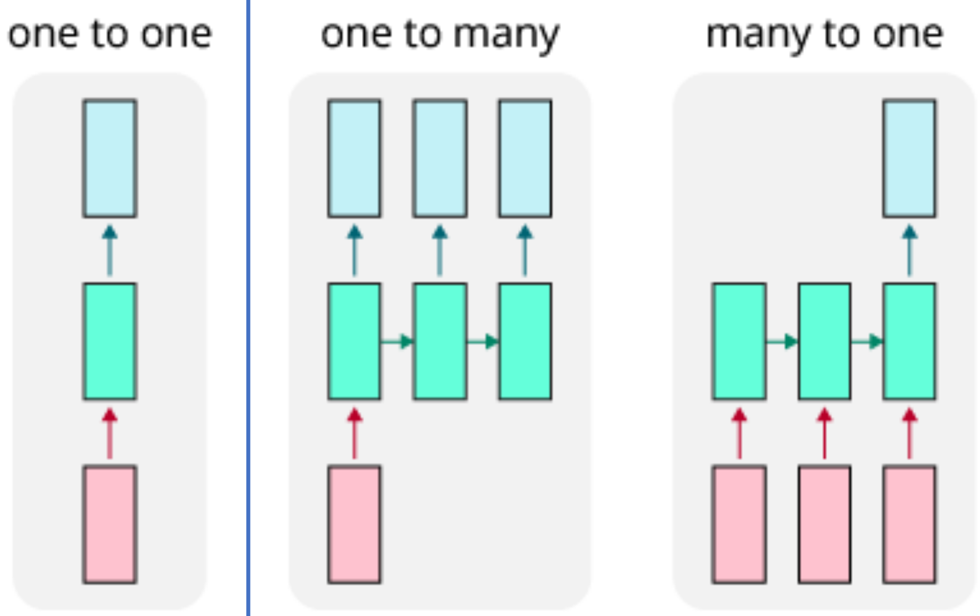


e.g. image captioning

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

So far: Standard “Feedforward” Neural Networks

Types of Sequence Problems



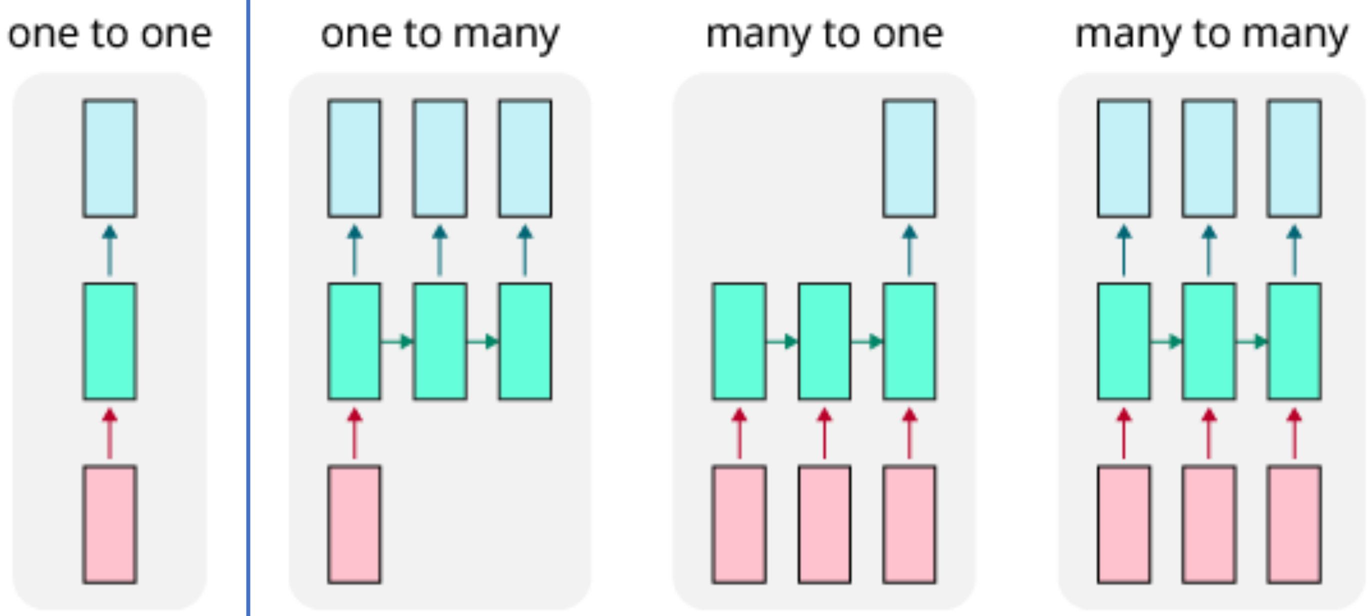
e.g. video classification, sentiment classification

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

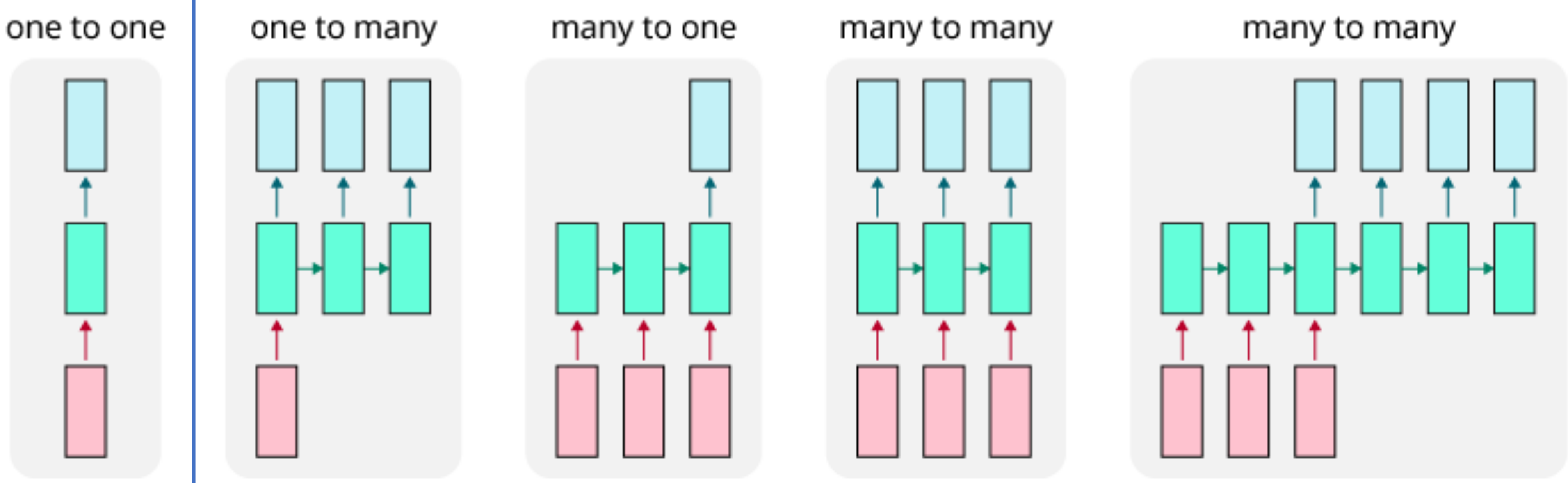
Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard
“Feedforward”
Neural Networks

Types of Sequence Problems

(Input length = output length)

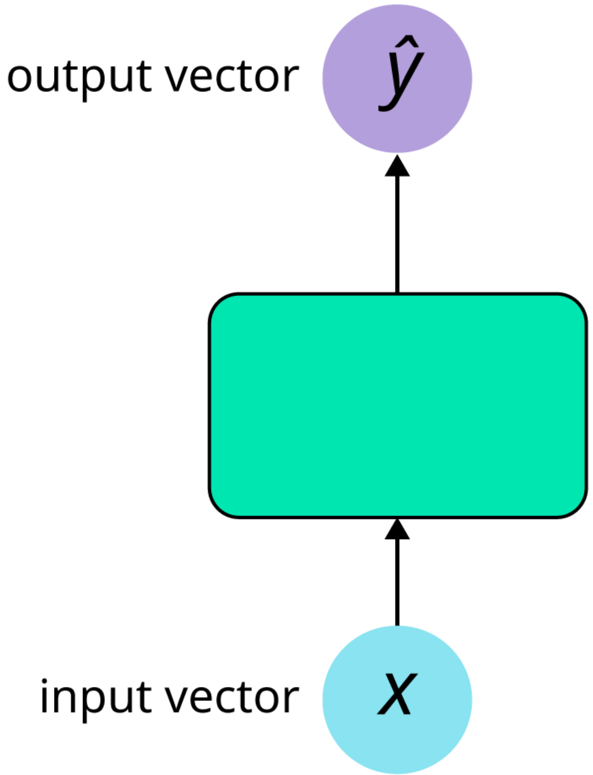
(Input length ≠ output length)



e.g. machine translation, chatbots

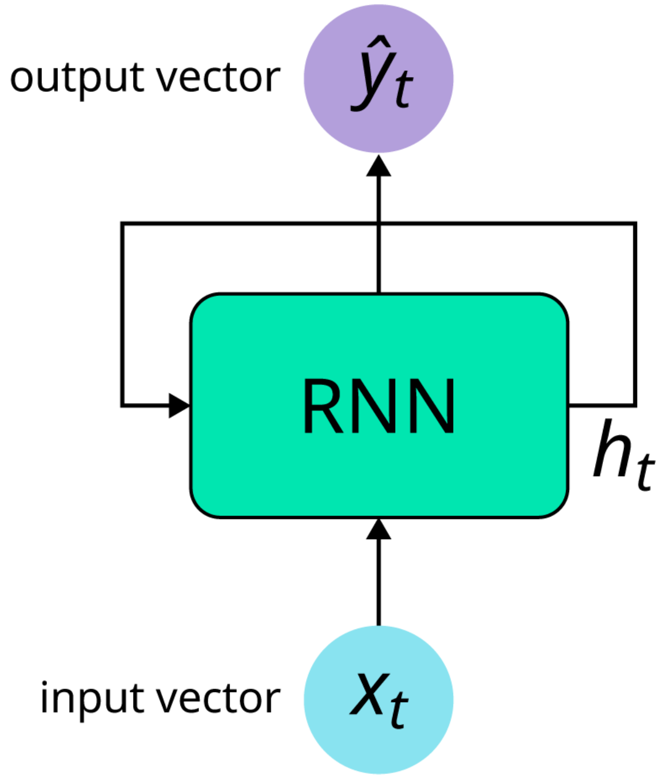
Image from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

So far: Standard "Feedforward" Neural Networks



Recurrent Neural Networks (RNNs) for sequential modelling

Have an **internal loop**

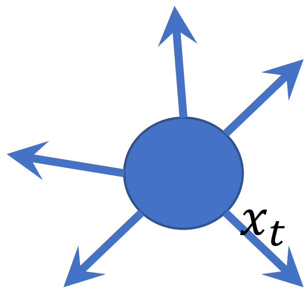


Reference for Topic 1

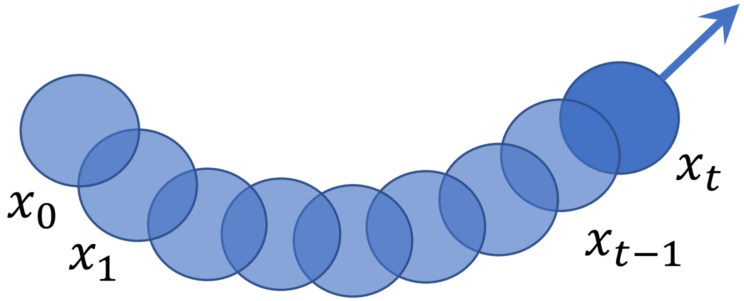
- Blog by Andrej Karpathy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.

Topic 2:
Recurrent Neural Networks
(RNNs)

Given a location of a ball at present, can you predict where it will go in the next second?



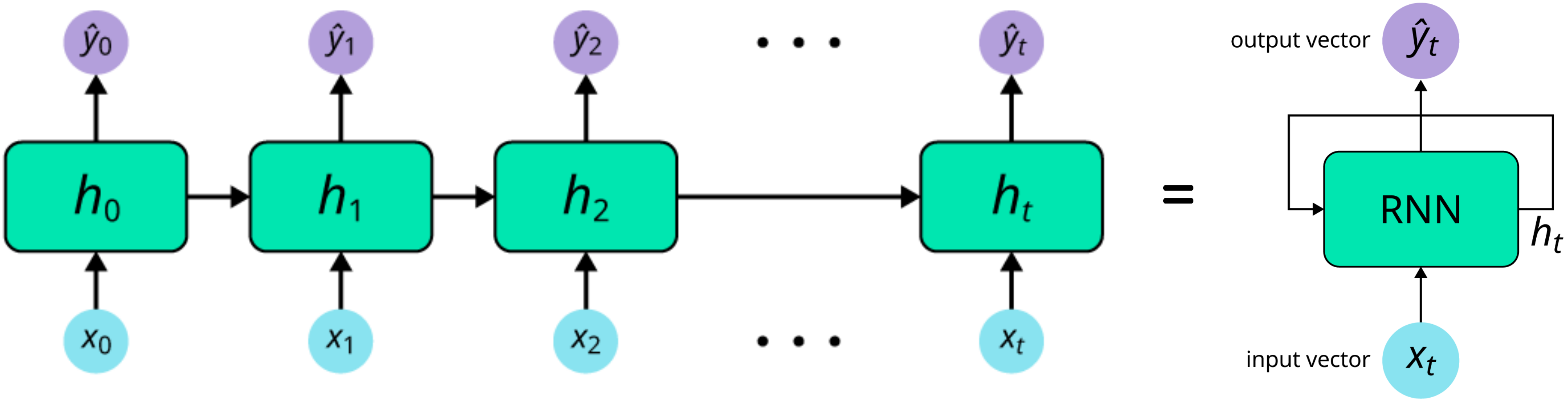
Given a location of a ball at present, can you predict where it will go in the next second?

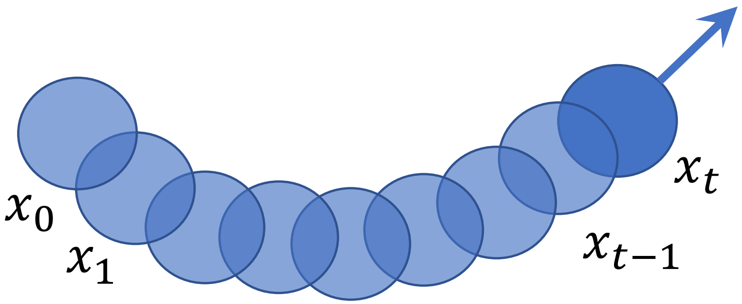


Q: How to model such a dependency of an input sequence?

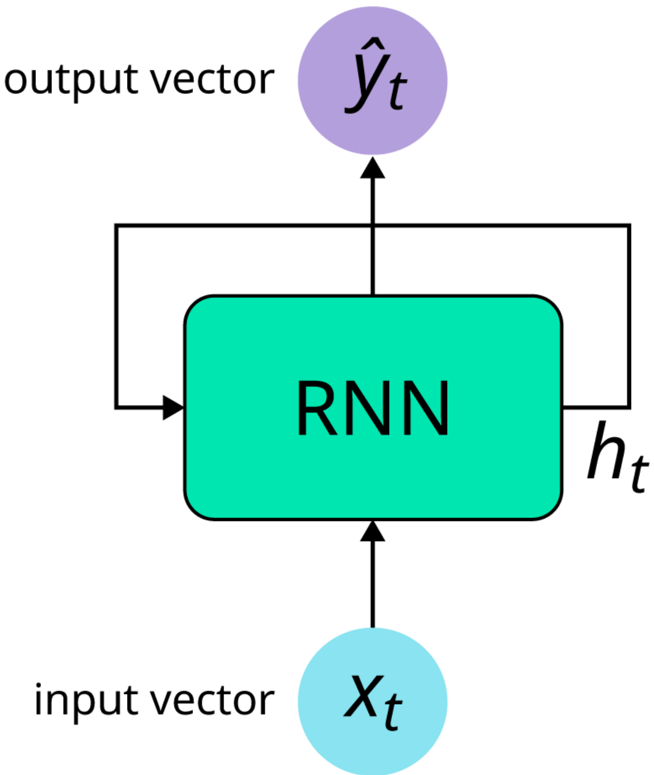
A: Using a hidden state

Recurrent Neural Networks (RNNs)





Recurrent Neural Networks (RNNs)

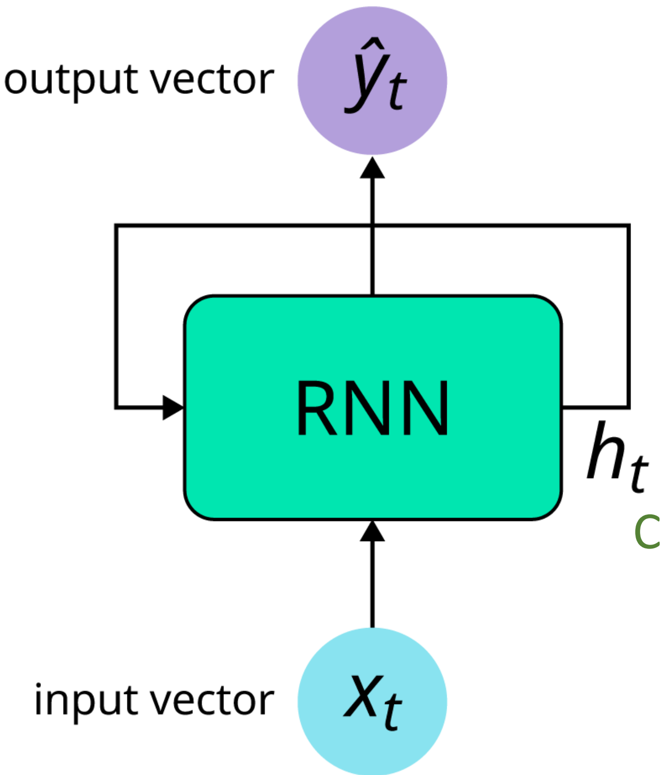


Also called “Vanilla RNNs”

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

The	cat	sat	on	the	mat
x_0	x_1	x_2	x_3	x_4	x_5

“Recurrent” Neural Networks (RNNs)



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

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

Current hidden state

Function parameterized by W

old state

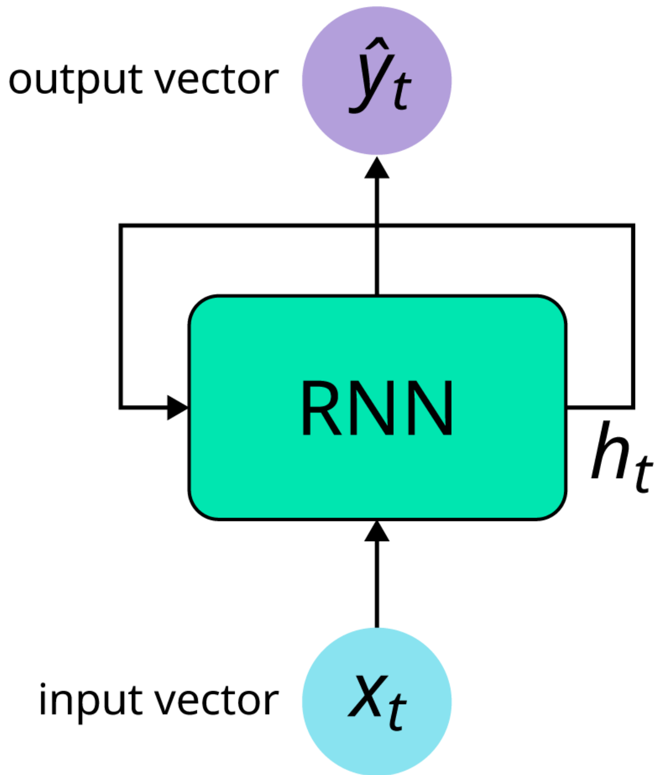
Current input vector

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

...

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

State update and output



Output vector

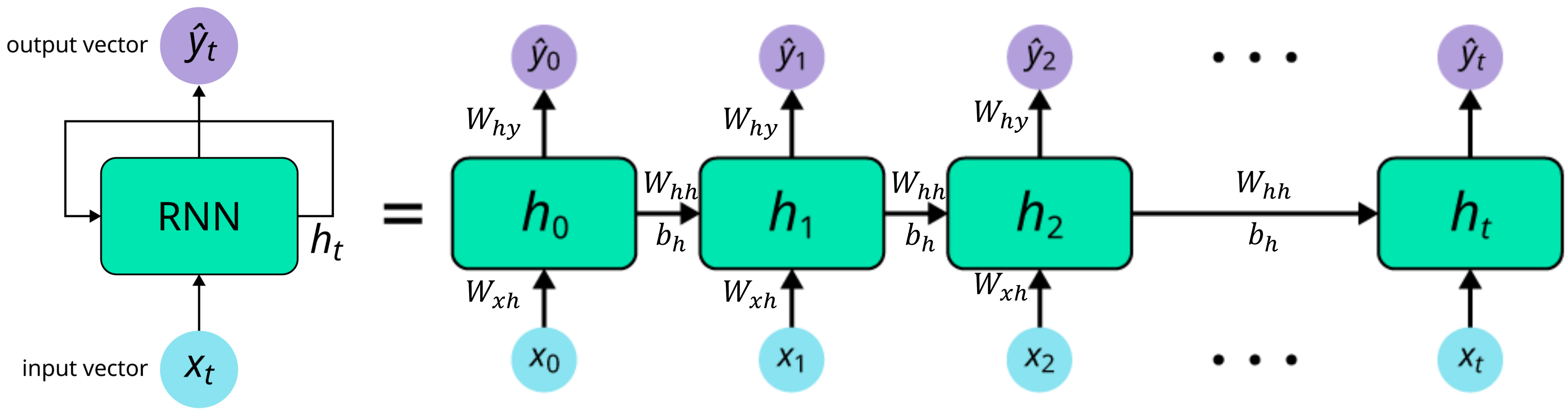
$$\hat{y}_t = g(W_{hy}h_t + b_y)$$

Update the hidden state

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

Unfolding an RNN

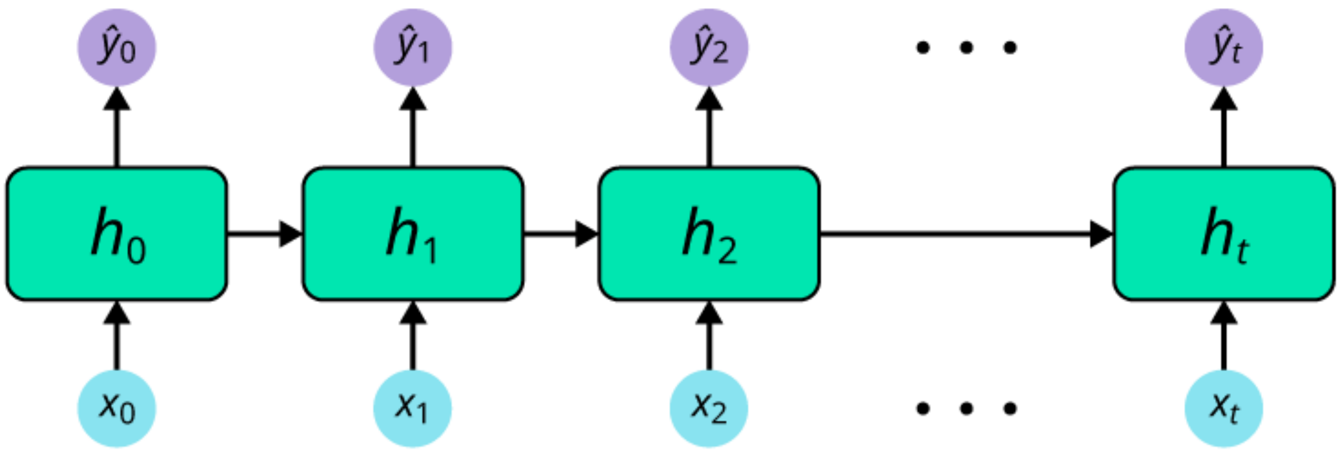
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$
$$\hat{y}_t = g(W_{hy} h_t + b_y)$$



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.

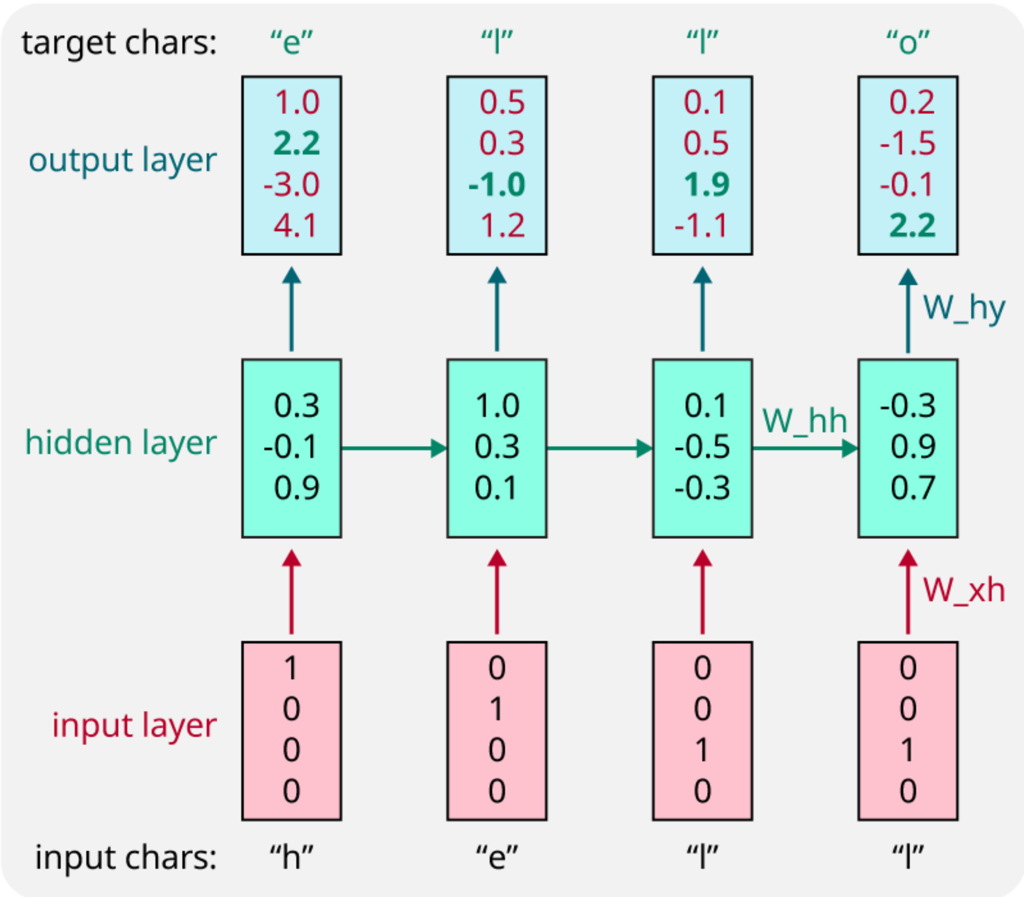


Reference for Topic 2

- 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

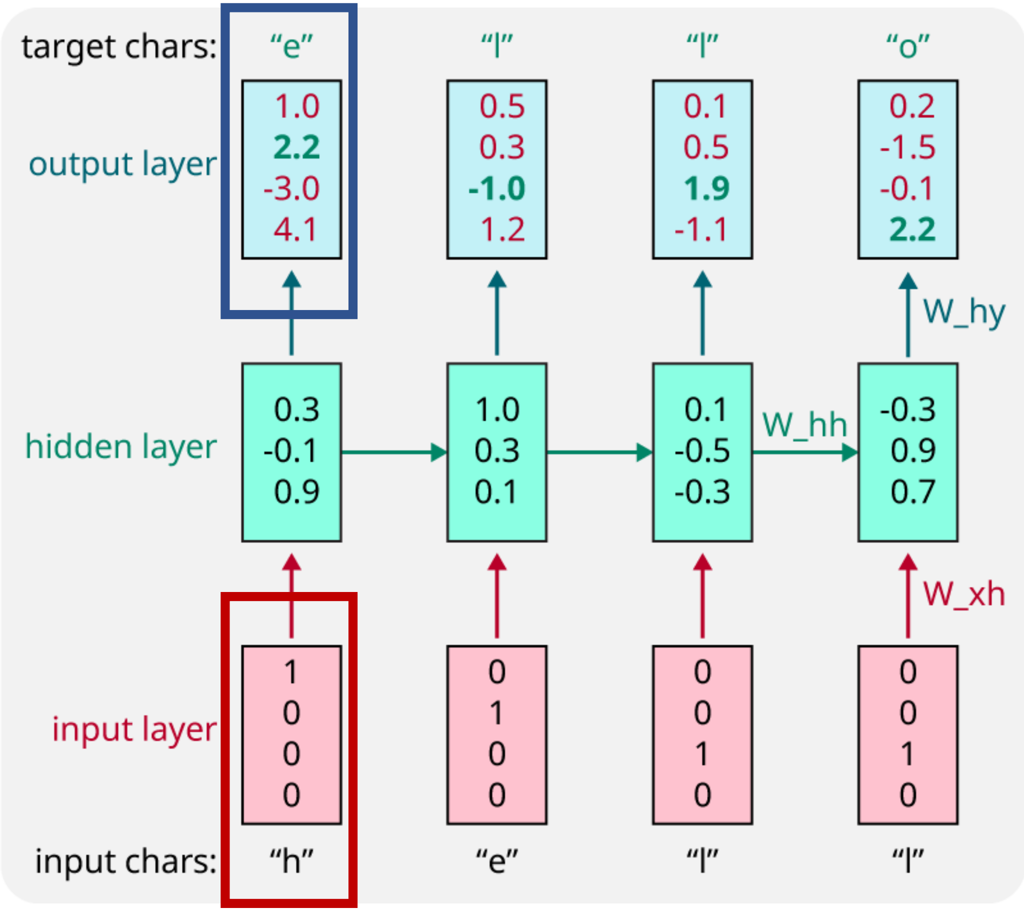
Example: Character-level Language Modelling



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

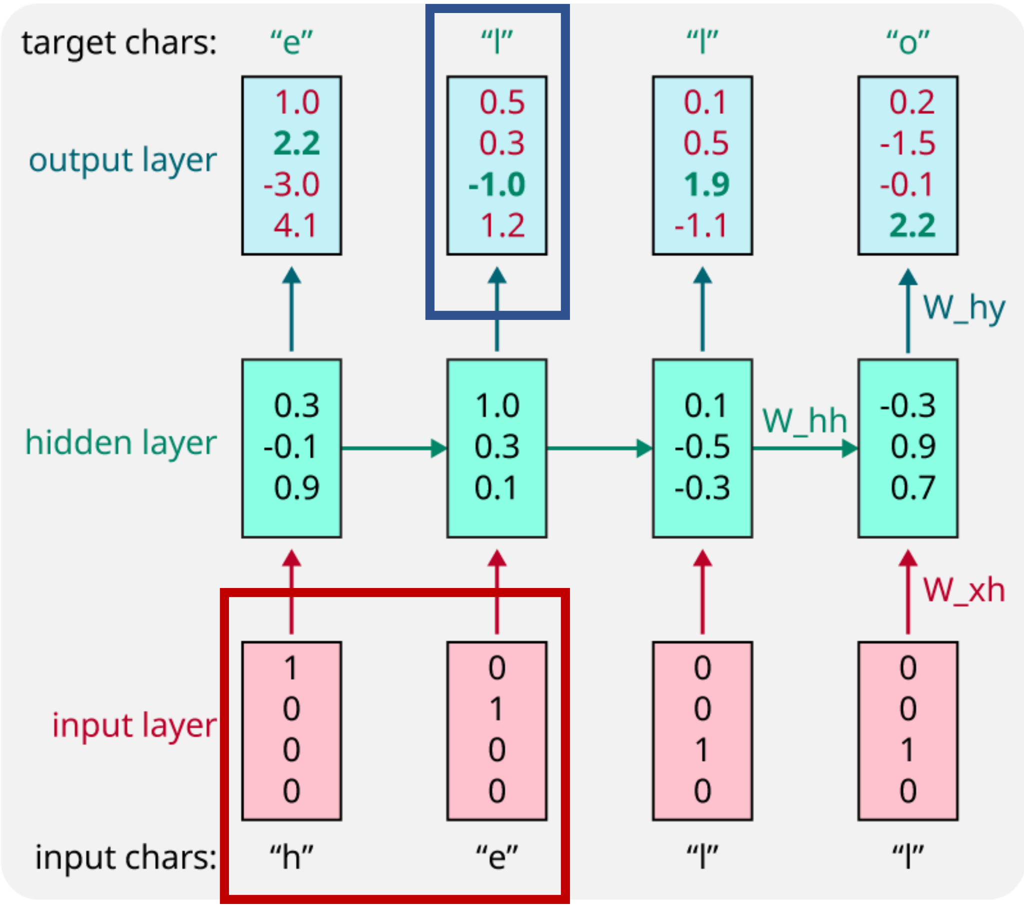
Example from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Given “h”, target output: “e”



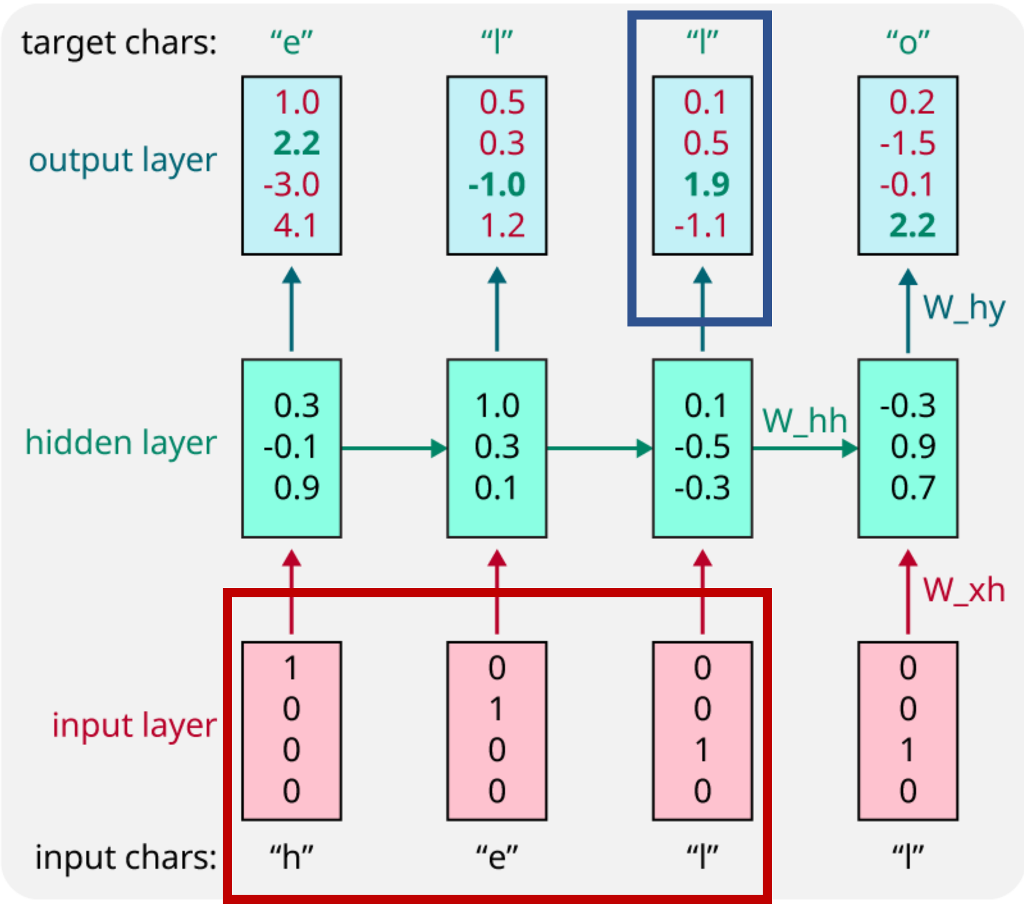
Task: Given characters at time 1, 2, ..., t, predicts the next character (at time t+1)

Given “he”, target output: “l”



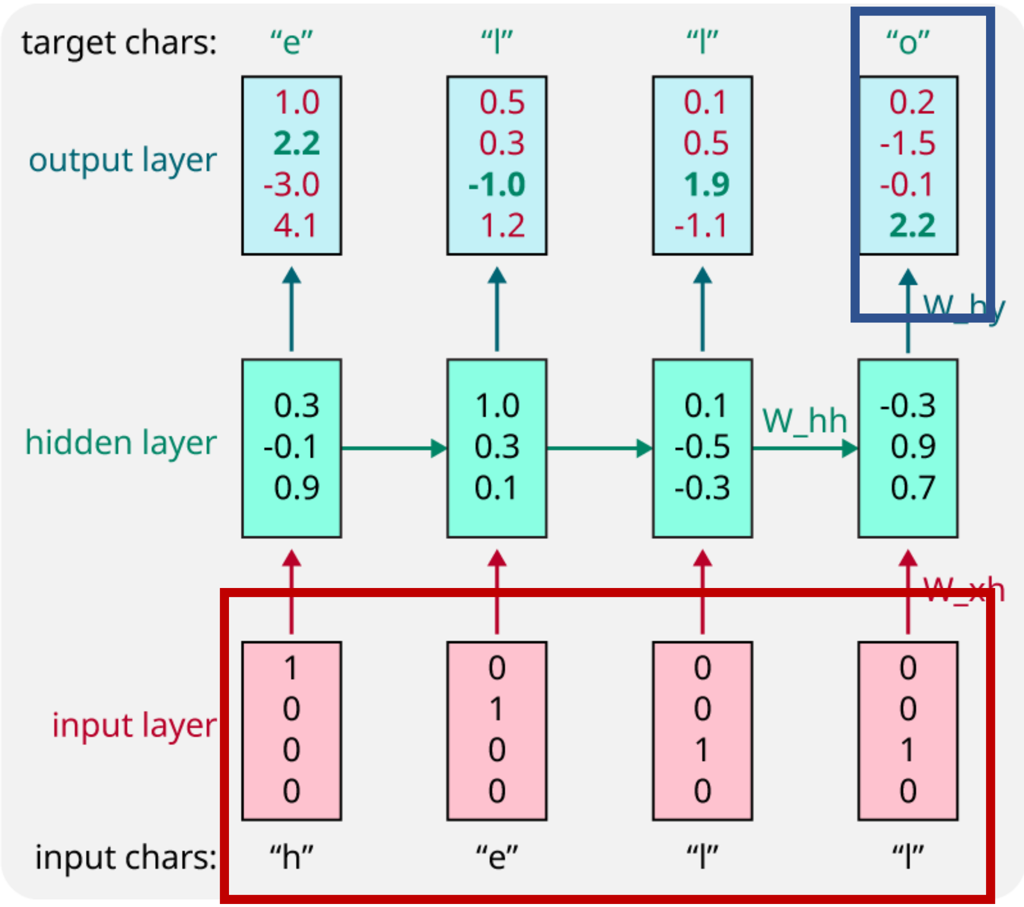
Task: Given characters at time 1, 2, …, t, predicts the next character (at time t+1)

Given “hel”, target output: “l”



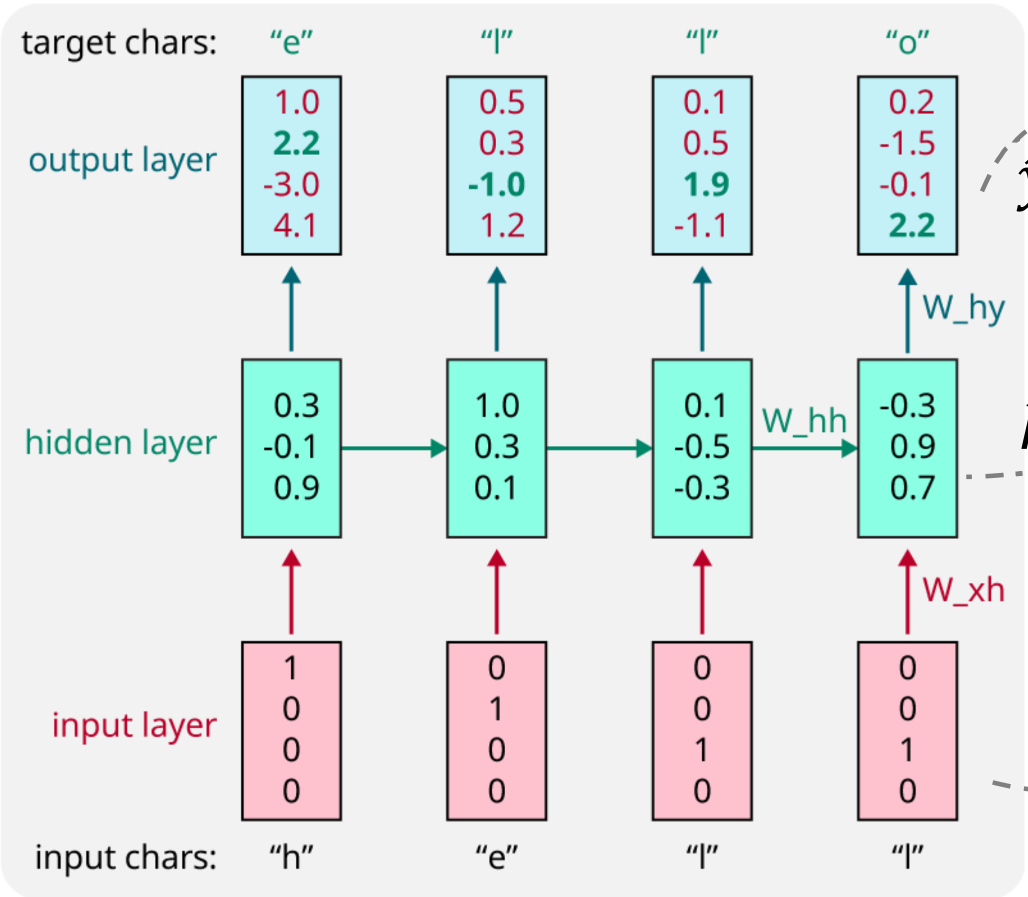
Task: Given characters at time 1, 2, …, t, predicts the next character (at time t+1)

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

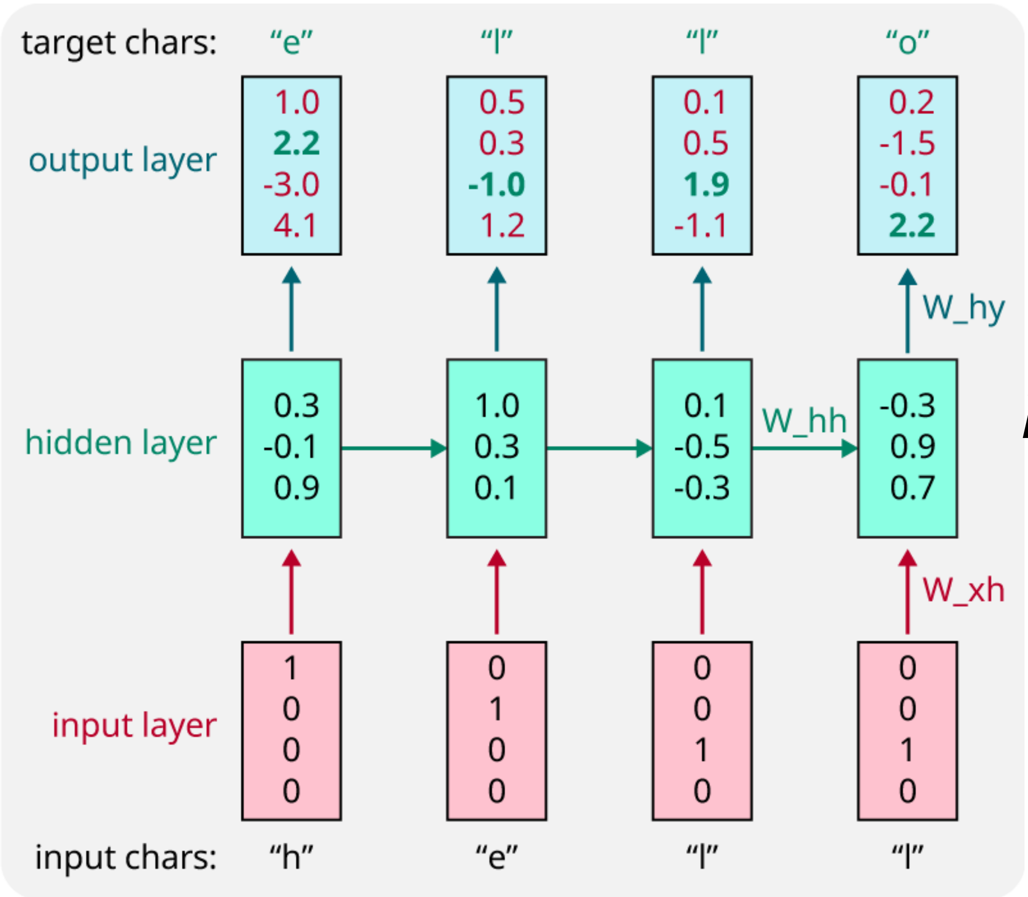


$\hat{y}_t$ The output layer provides **confidences** for the next character. We want the green numbers to be high and red numbers to be low.

h_t We use 3 dimentions 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



$$\hat{y}_t = g(W_{hy}h_t + b_y)$$

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

x_t

Simplified RNN notation:

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

$h_t, b_h \in \mathbb{R}^H, x_t \in \mathbb{R}^M$

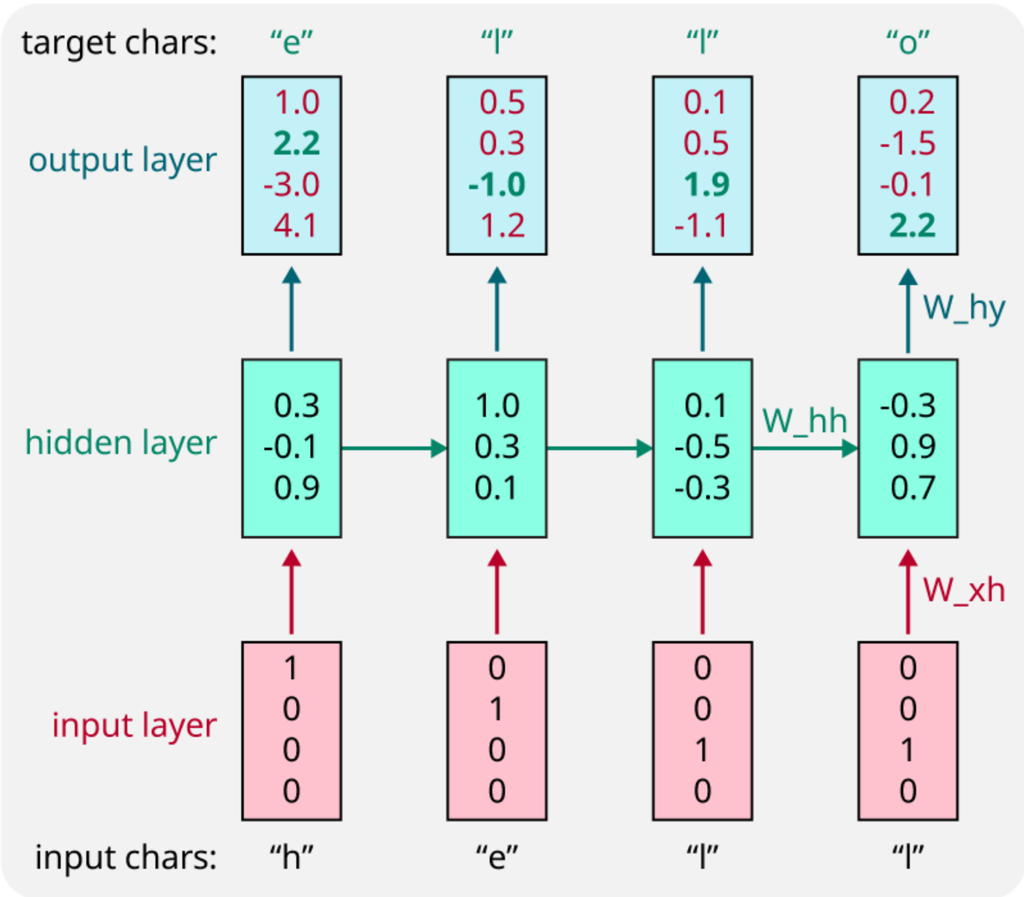
Q: $W \in ?$ $W \in \mathbb{R}^{H\times(H+M)}$

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



Initiate the weights

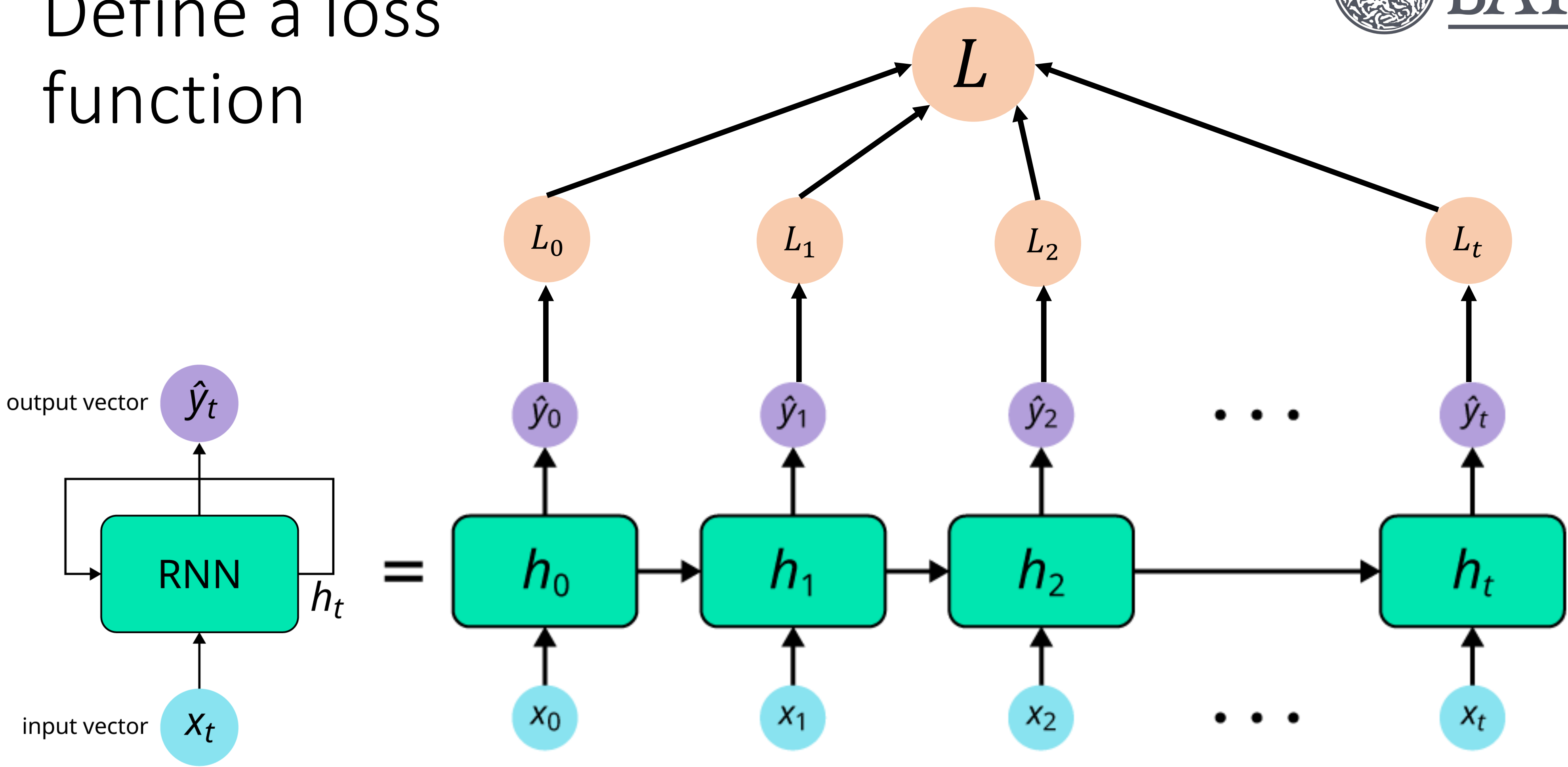
Calculate the loss L.

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

Update the weights

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

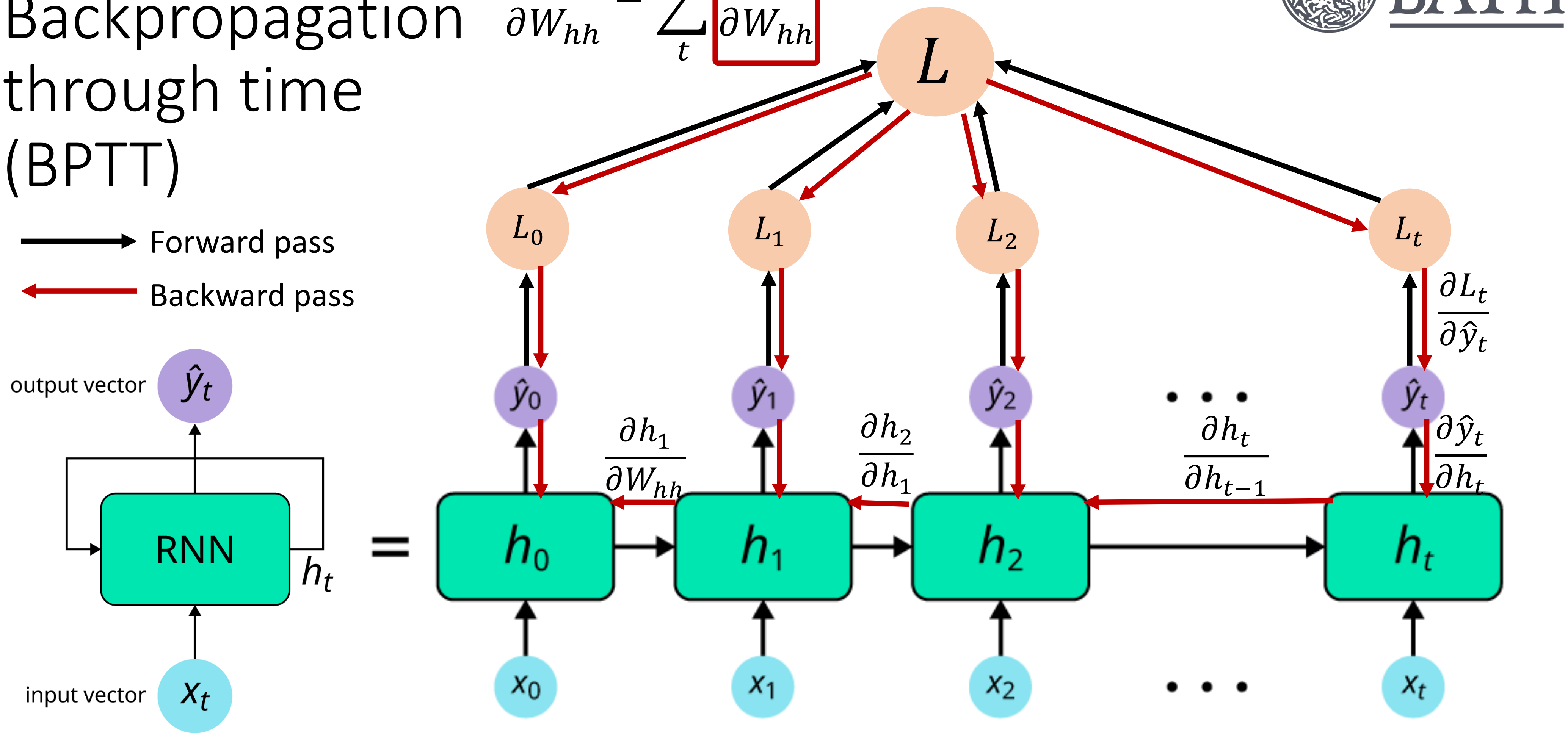
Define a loss function



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

Backpropagation through time (BPTT)

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



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

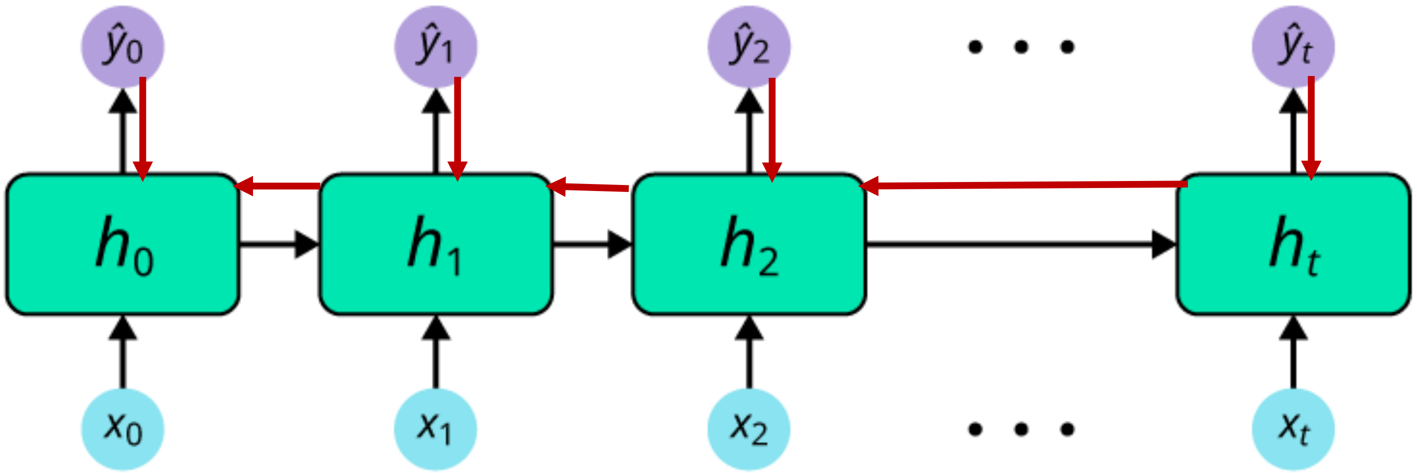
See Weberna's blog for detailed equations: https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html

RNN gradient flow

Many values > 1:
exploding gradients

Gradient clipping: scale big gradients

Computing the gradient wrt W involves multiplying many factors

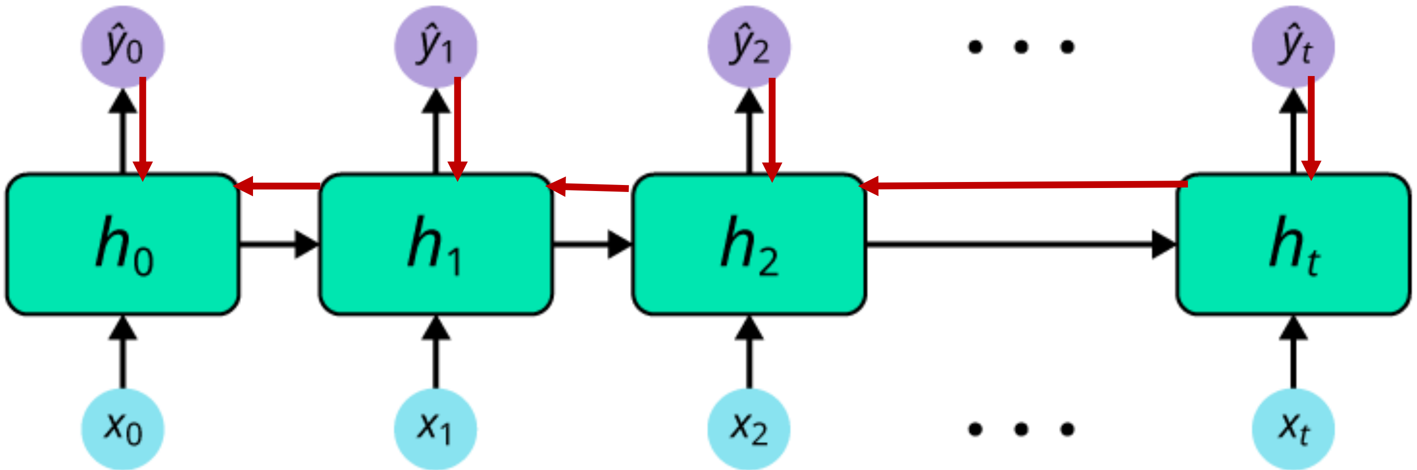


Many values < 1:
vanishing gradients

RNN gradient flow

Many values > 1:
exploding gradients

Computing the gradient wrt W involves multiplying many factors



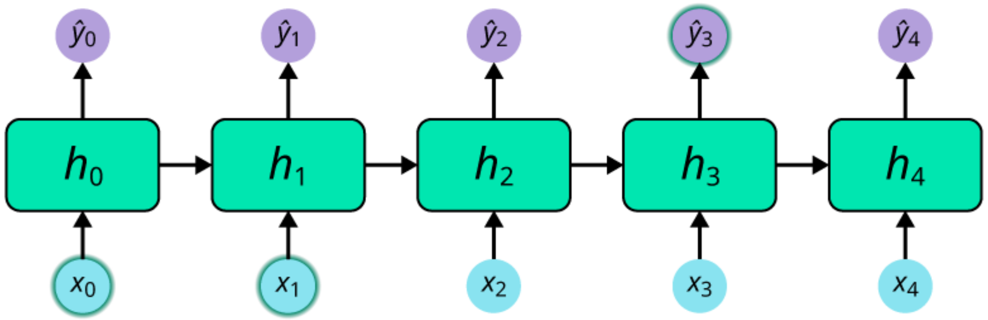
Many values < 1:
vanishing gradients

→ **Change RNN architecture**

The vanishing gradient problem

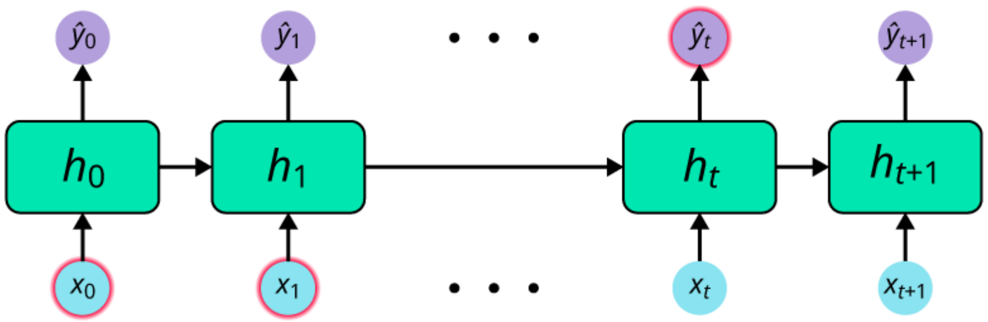
- Short term dependencies:

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

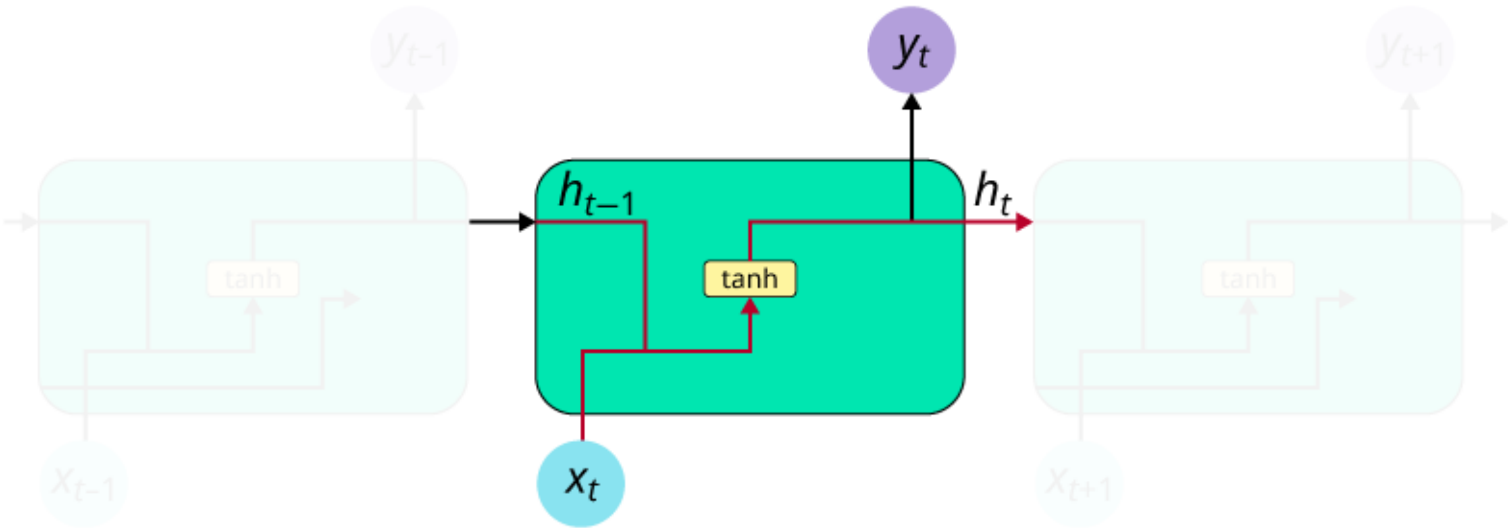
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Blog by Denny Brits: http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html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

Topic 5: LSTM-1

Standard/Vanilla RNNs

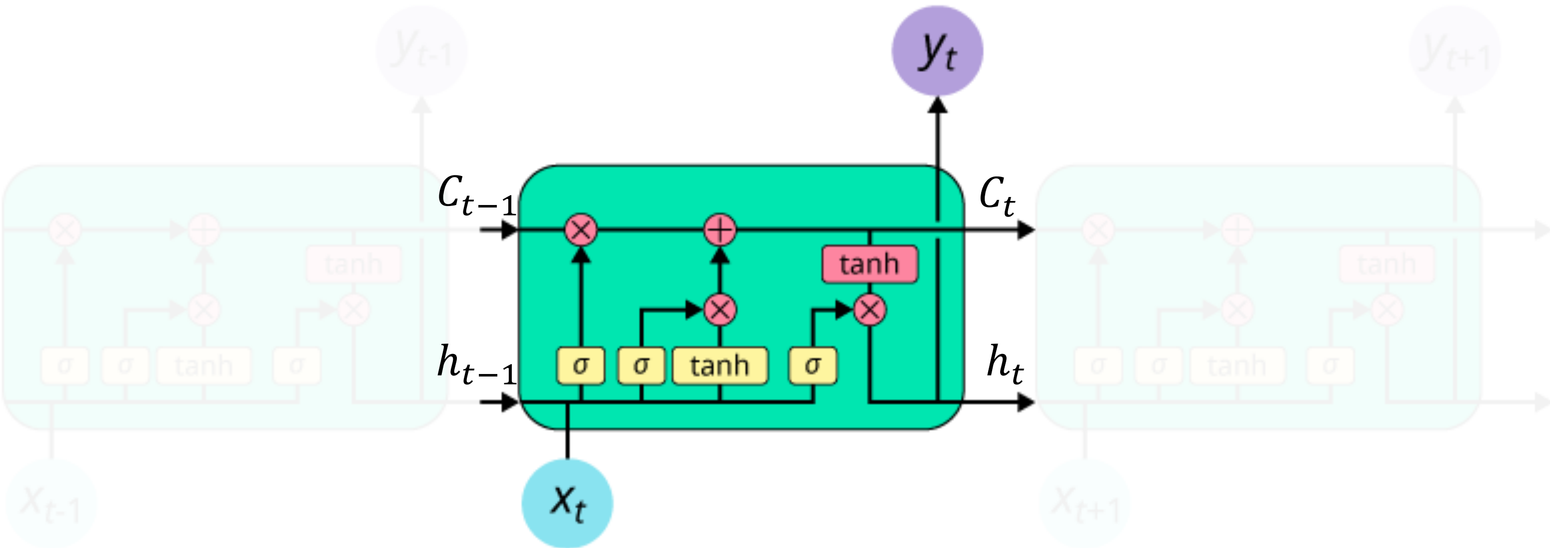
- In a standard RNN, repeating modules contain a **simple computation** node.

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



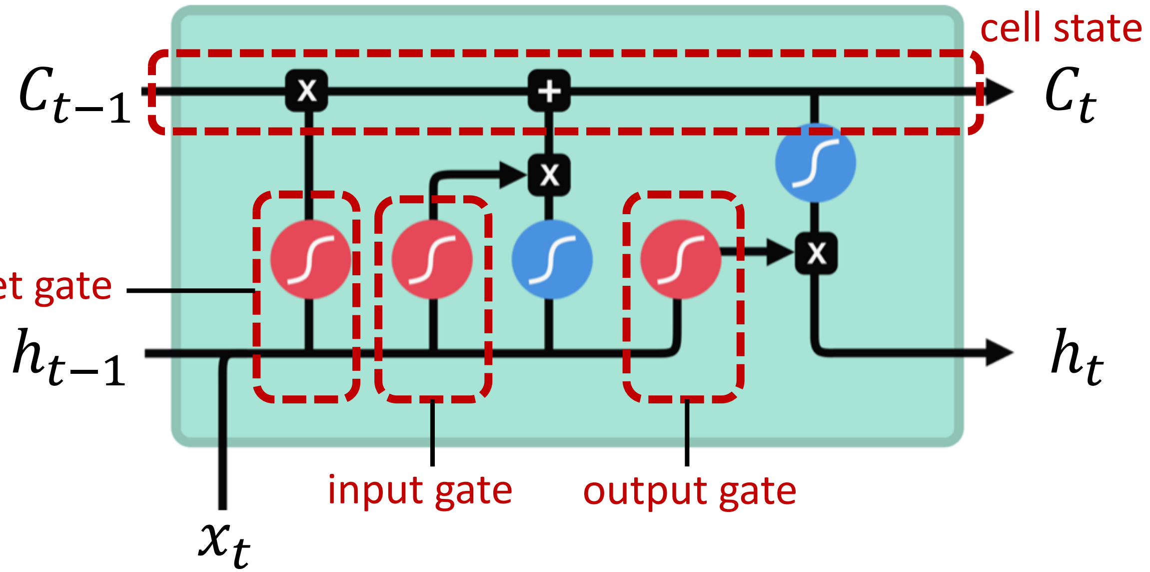
Long-Short Term Memory (LSTM)

- LSTMs are explicitly designed to deal with the long-term dependency problem.
- LSTM modules contain computational blocks that control information flow.



Two core concepts of LSTMs

- **Gates**: to control what information is to keep and forget.
- **Cell state**: act as a transport highway that transfers relative information all way down the sequence chain, thus store long-term information.



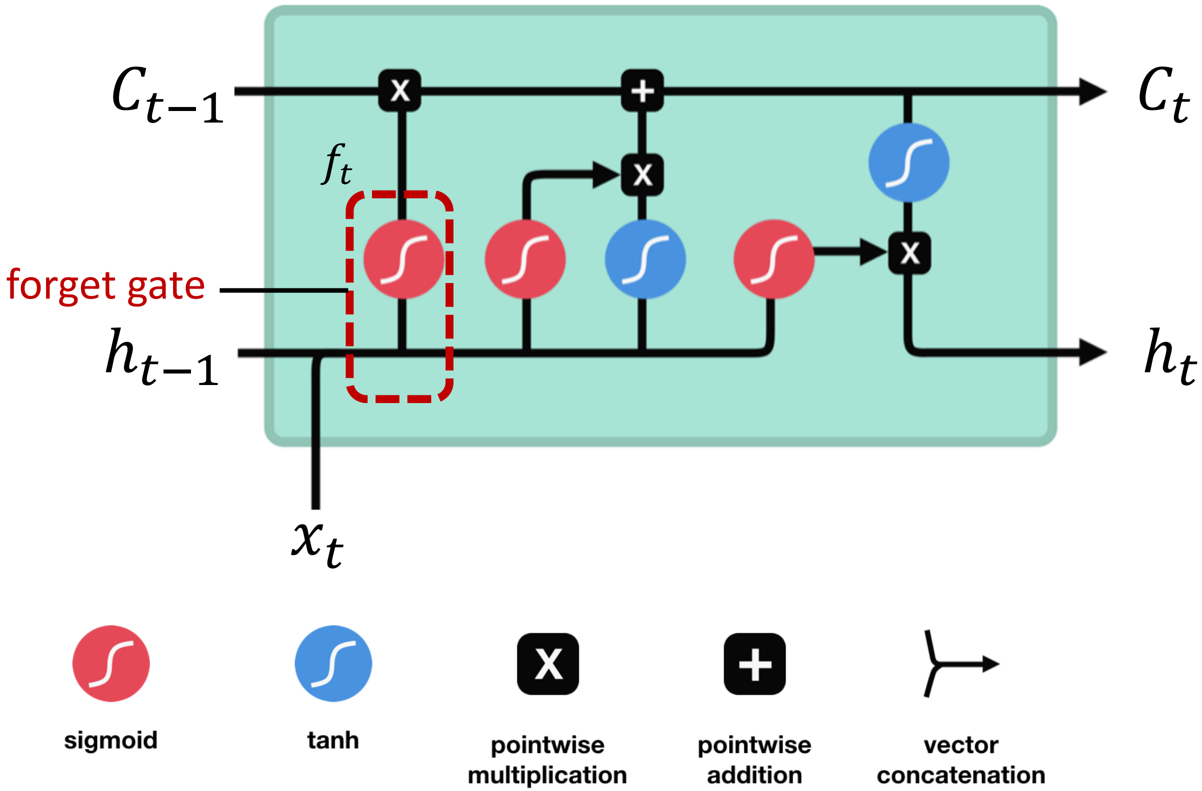
All gate values are between 0 (discard) and 1 (keep).

Image from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Forget gate

- **Forget gate**: forget irrelevant parts of the previous state.

$$f_t = \sigma(W_f \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_f)$$



Animation from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

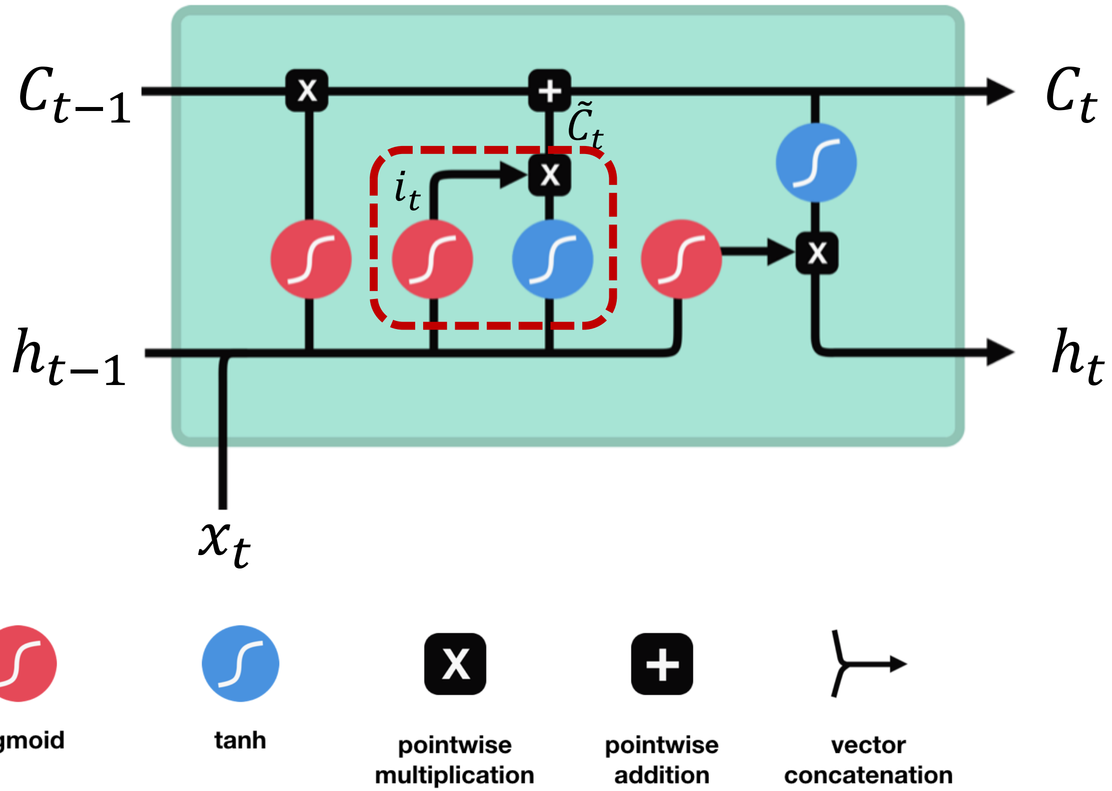
Input gate

- **Input gate**: decides what information is relevant to add from the current step.

$$i_t = \sigma(W_i \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_i)$$

- **New cell content**: the new content to be written to the cell

$$\tilde{C}_t = \tanh(W_C \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_C)$$



Animation from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

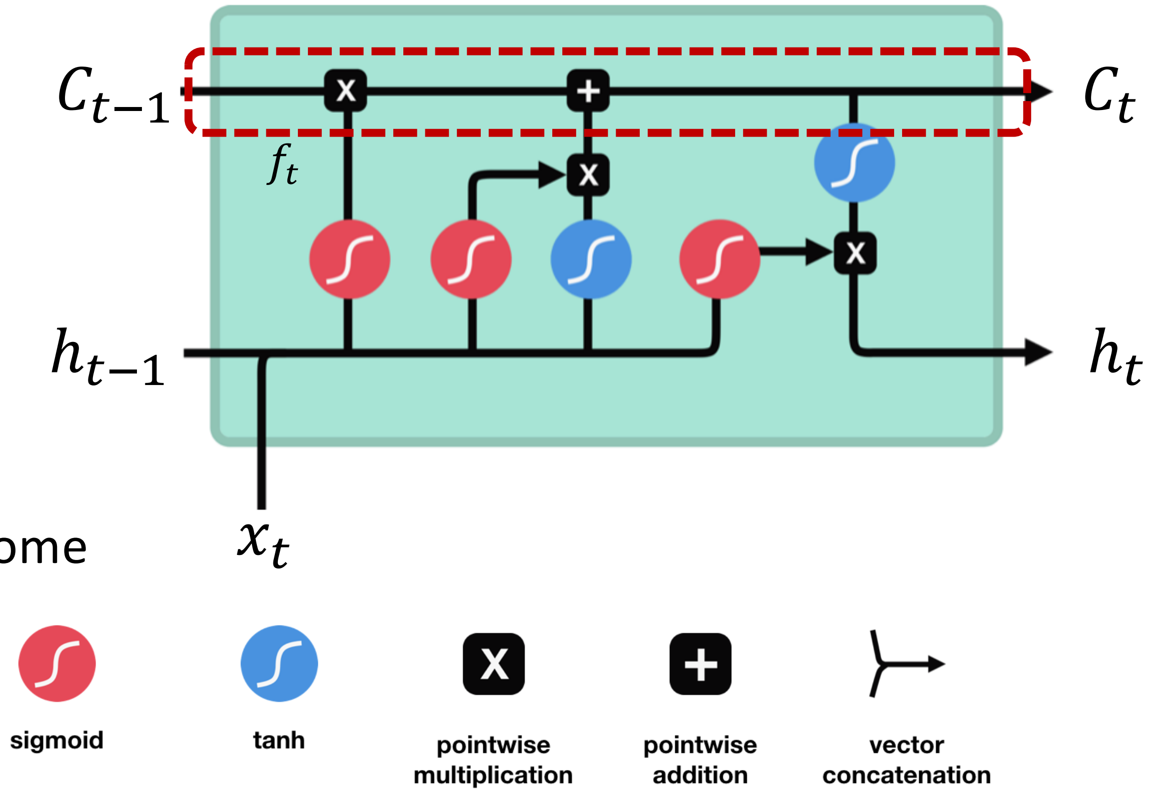
Cell state

- **Cell state**: update the cell state to new values that the network finds relevant.

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

Keep ("input") some new cell content

Erase ("forget") some content from the previous state



Animation from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

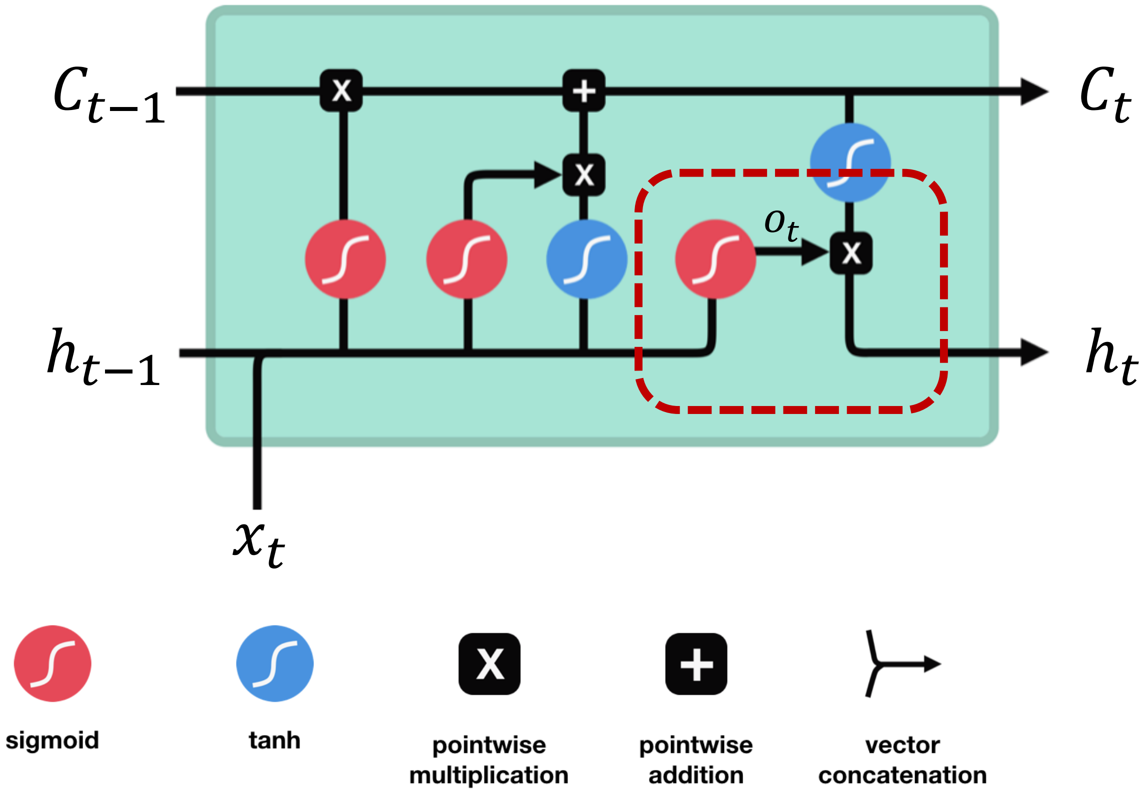
Output gate

- **Output gate**: determines what parts of the cell are output to the hidden state.

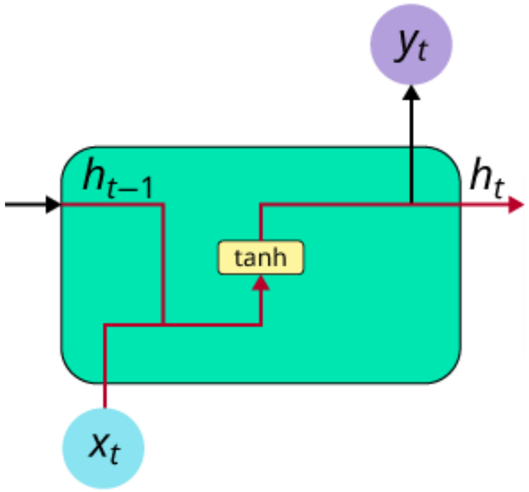
$$o_t = \sigma\left(W_o \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_o\right)$$

- **Hidden state**: read ("output") some content from the cell.

$$h_t = o_t \circ \tanh(C_t)$$



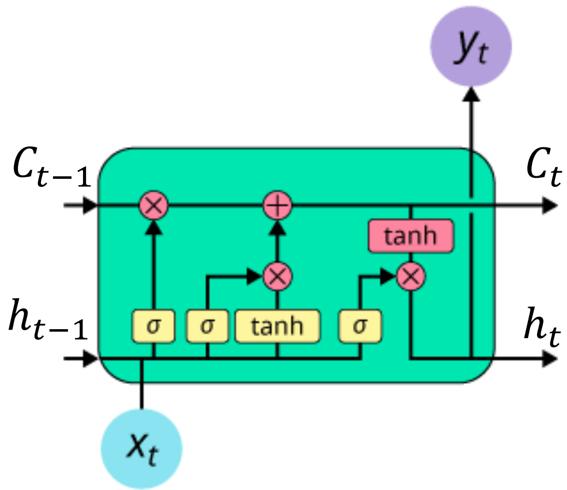
Animation from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21



Vanilla RNN:

$h_t, b_h \in \mathbb{R}^H$
$x_t \in \mathbb{R}^M$
$W \in \mathbb{R}^{H\times(H+M)}$

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



LSTM:

$f_t \in \mathbb{R}^H$
$i_t \in \mathbb{R}^H$
$o_t \in \mathbb{R}^H$
$\tilde{C}_t \in \mathbb{R}^H$
$C_t \in \mathbb{R}^H$
$h_t \in \mathbb{R}^H$
$x_t \in \mathbb{R}^M$
$W_f, W_i, W_o, W_C \in \mathbb{R}^{H\times(H+M)}$

$$f_t = \sigma\left(W_f \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_f\right)$$

$$i_t = \sigma\left(W_i \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_i\right)$$

$$o_t = \sigma\left(W_o \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_o\right)$$

$$\tilde{C}_t = \tanh\left(W_C \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_C\right)$$

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

$$h_t = o_t \circ \tanh(C_t)$$

UNIVERSITY OF BATH

Reference for Topic 5

- Blog by Colah: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Blog by Michael Phi: Illustrated Guide to LSTM's and GRU's: A step to step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Topic 6: LSTM-2

LSTM Architecture

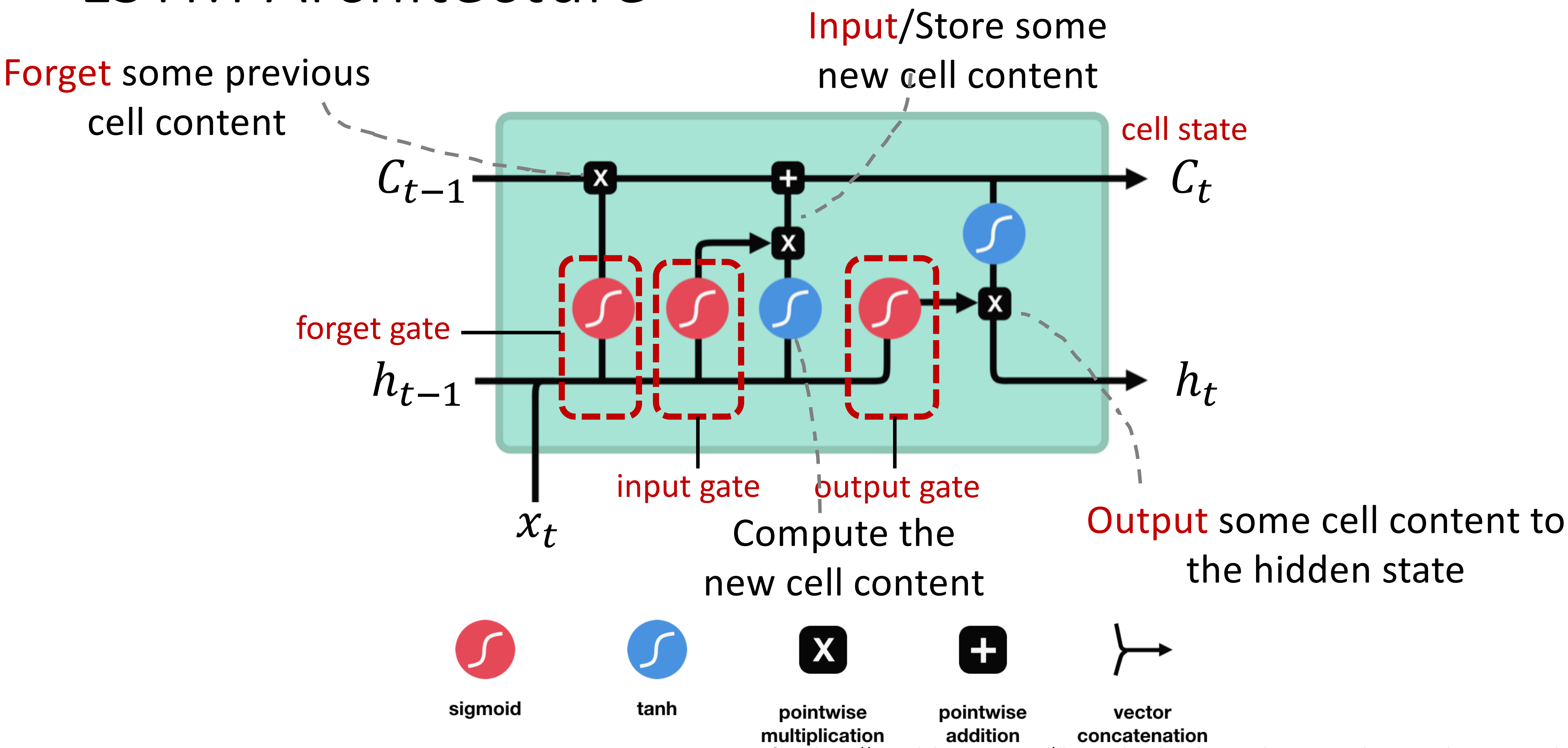
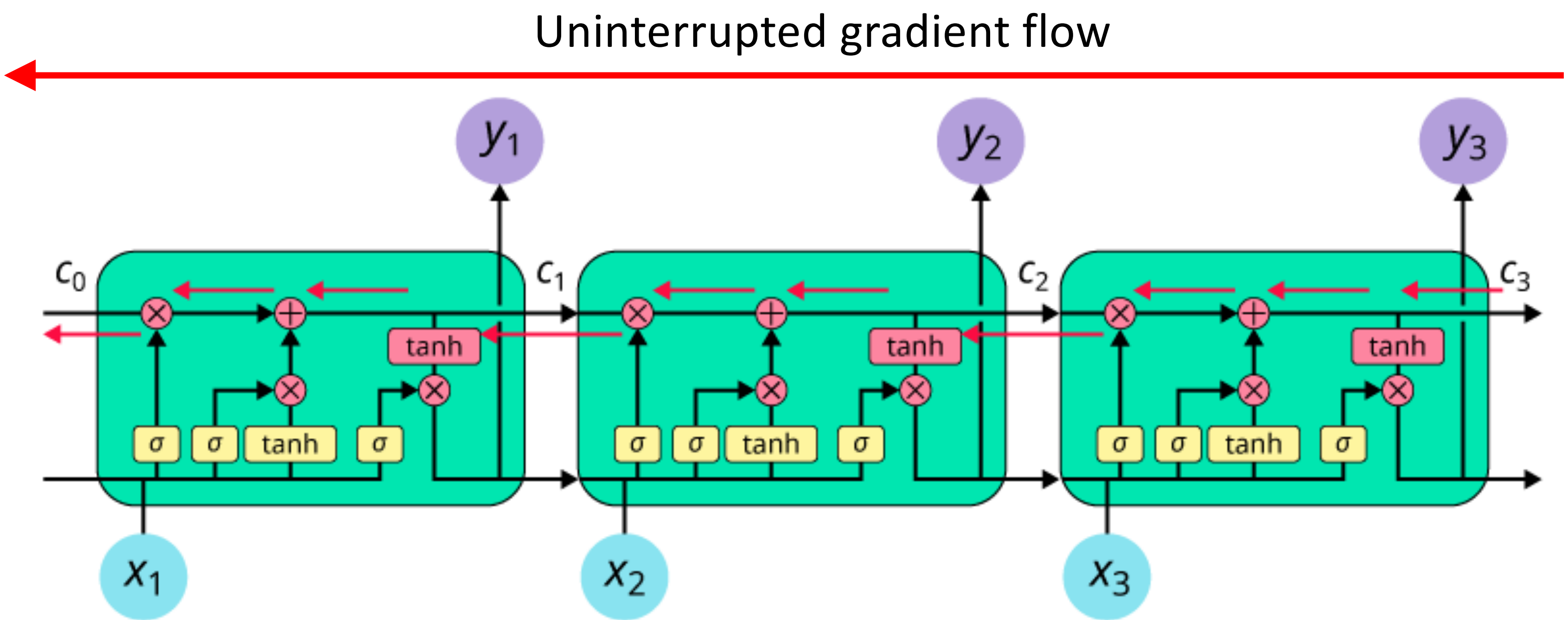


Image from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LSTM Gradient Flow



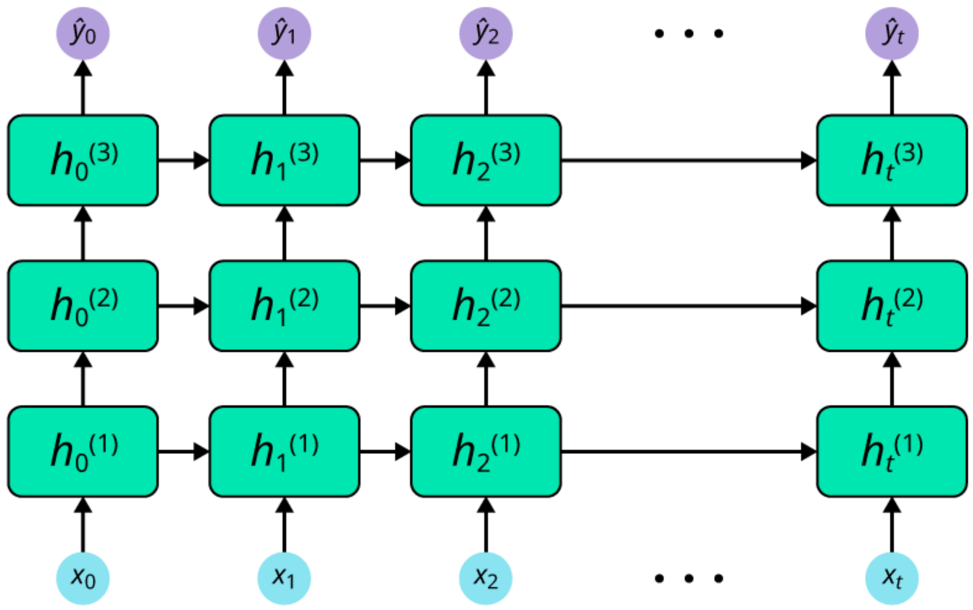
LSTMs **solve the vanishing/exploding gradient problem** using an additive gradient structure.

A detailed explanation: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Advanced use of RNNs/LSTMs

- **Multi-layer RNNs (Deep RNNs)**: stack more than one RNN. It increases the representation power of the network, at the cost of higher computational loads.

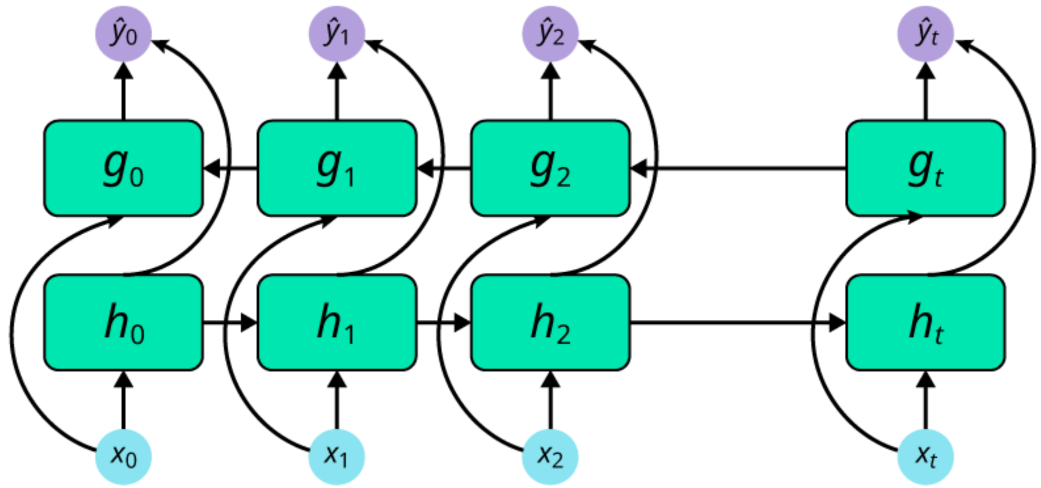
```
from keras.layers import LSTM
...
model.add(LSTM(32), return_sequences=True)
model.add(LSTM(32), return_sequences=True)
model.add(LSTM(32))
...
```



`True` means: output all the hidden states

Advanced use of RNNs/LSTMs

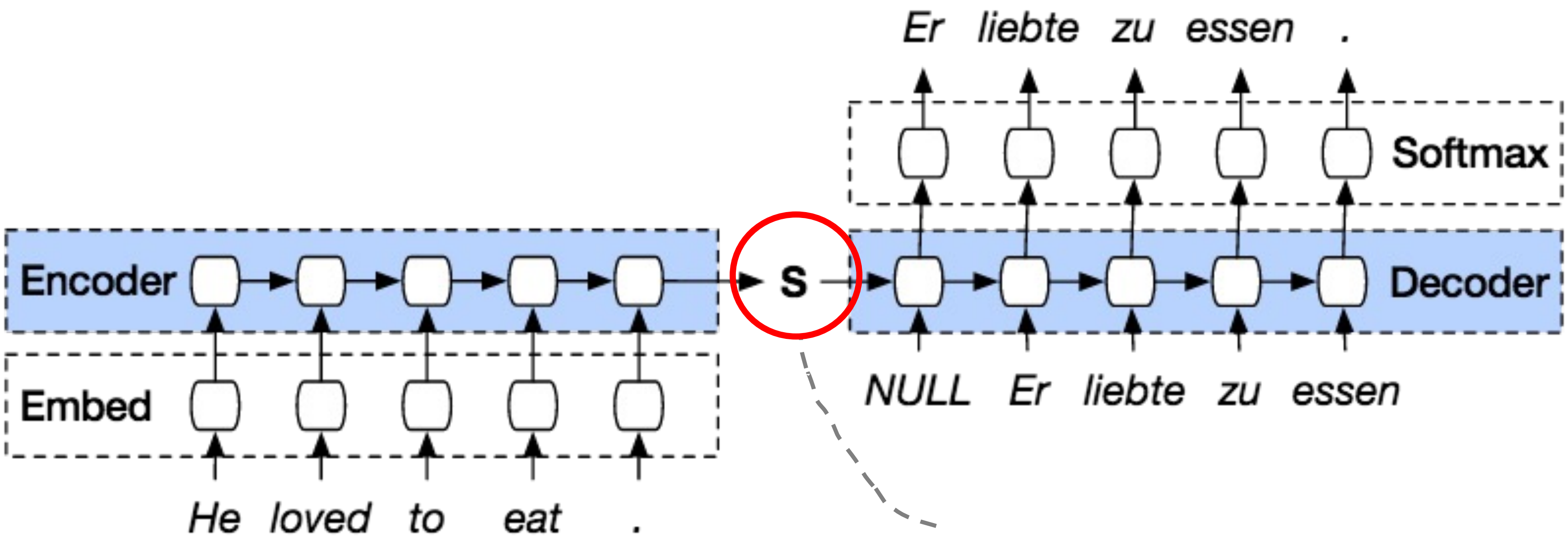
- **Bidirectional RNNs**: process a sequence in both directions, capturing pattens that may be missed by the chronological-order version alone.




```
from keras.layers import Bidirectional, LSTM
...
model.add(Bidirectional(LSTM(32)))
...
```

Example Tasks: Neural Machine Translation (NMT)

- Seq2seq [Sutskever et al, 2014][Cho et al, 2014]



Problem: **Encoding bottleneck**
One solution**: Attention Based seq2seq**

Image from: https://smerity.com/articles/2016/google_nmt_arch.html

Example Tasks: Image Caption Generation

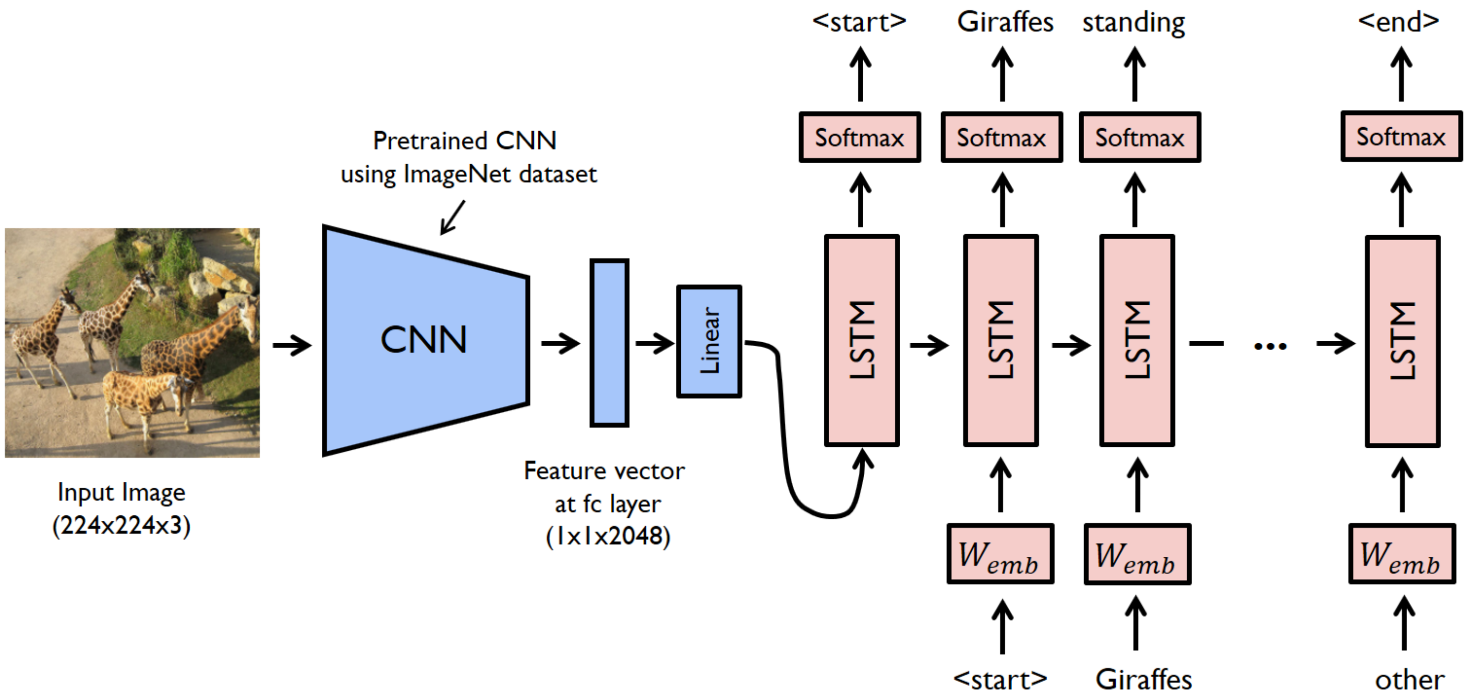


Image from: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

Summary of LSTMs

- Beside the hidden state, also maintain a **cell state** to store **long-term information.**
- Use **gates** to control the flow of information
  - **Forget** gate: gets rid of irrelevant **old** information
  - **Input** gate**:** stores relevant information from **current** input
  - **Output** gate: **output** a filtered version of the cell state
- Backpropagation through time with **uninterrupted gradient flow**, to avoid the vanishing/exploding gradient problem.

Reference for Topic 6

- Blog by Colah: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Blog by Michael Phi: Illustrated Guide to LSTM's and GRU's: A step to step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html
- Blogs by Nir Arbel: How do LSTM networks solve the problem of vanishing gradients: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577