Universidade de São Paulo
Escola Politécnica - Engenharia de Computação e Sistemas Digitais



# Recurrent Neural Networks

Prof. Artur Jordão

# Definition

## Recurrent Neural Networks



- Recurrent neural networks (RNNs) are models that capture the dynamics of **sequences** via recurrent connections

- Recurrent neural networks differ from feedforward networks (e.g., MLPs) by allowing cycles in their computation graphs

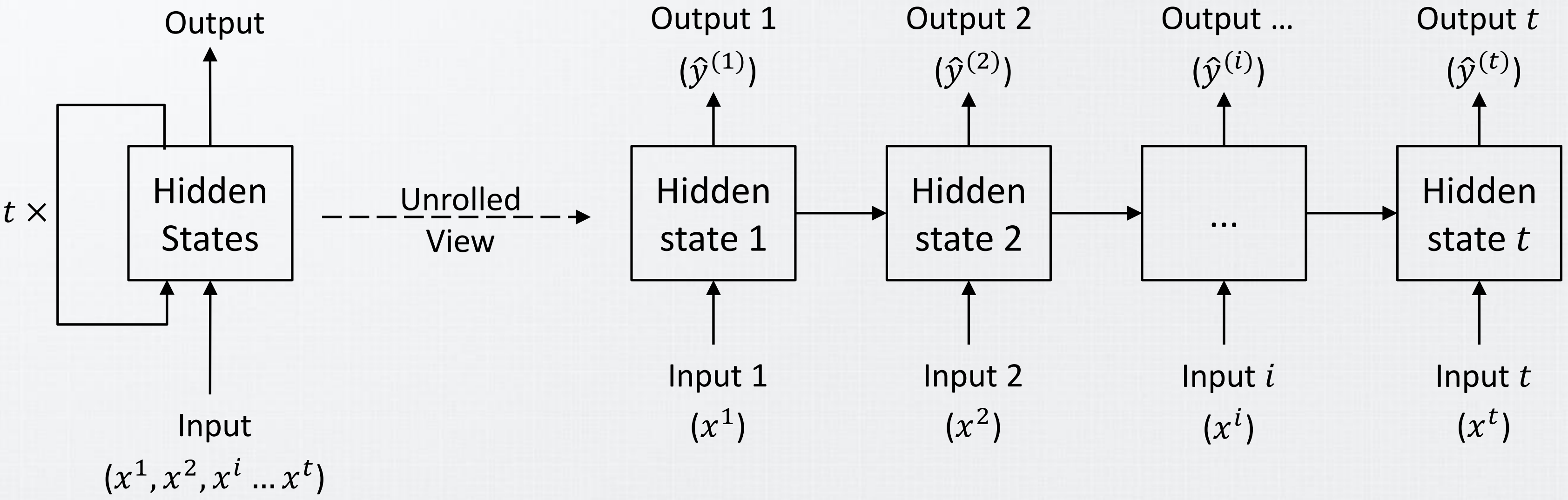
- RNNs are able to handle sequential and temporal data
  - They remember earlier parts of the sequence to interpret or contextualize later elements when making predictions

# Architectural Design

## Recurrent Neural Networks



- Recurrent neural networks unroll across time steps (or sequence steps), applying the **same underlying parameters** at each step
  - **The weights are shared across all time steps**



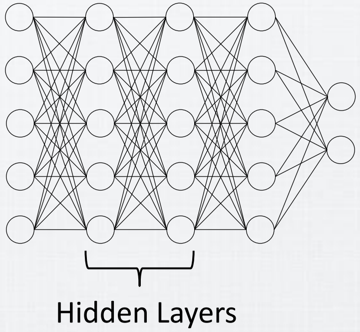
# Hidden Layers vs. Hidden States

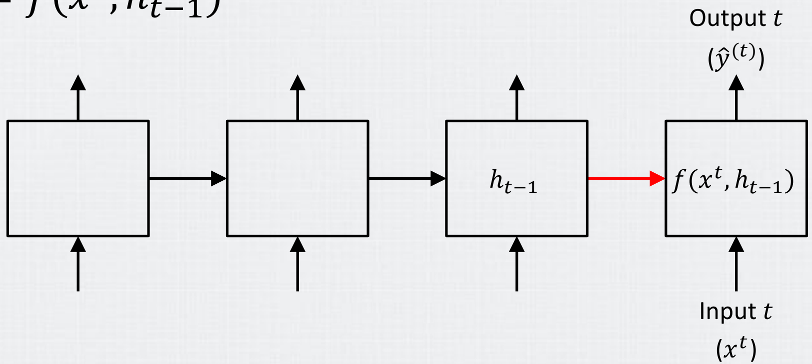
## Recurrent Neural Networks



- Hidden layers
  - Layers that are hidden from view on the path from input to output
  - Layers where the training data doesn't reveal the desired output

- Hidden states
  - Inputs for a given step that can only be computed by looking at data from previous time steps
  - The hidden state at any time step: $h_t = f(x^t, h_{t-1})$



# Hidden Layers vs. Hidden States

## Recurrent Neural Networks



- Recurrent neural networks are neural networks with **hidden states**
    - Inputs from earlier time steps influence the RNN's response to the current input
    - Store all the data it has observed (memory)

- Causal structure
    - The state at time $t$ captures information from the past: $x^{(1)}, \ldots x^{(t-1)}$ as well as the current input $x^{(t)}$

# Connections

## Recurrent Neural Networks



- The RNN has the following connections
  - Input to hidden
  - Hidden to hidden
  - Hidden to output

- Input to hidden
  - Connections parameterized by a weight matrix $U$

- Hidden to hidden
  - Recurrent connections parameterized by a weight matrix $W$

- Hidden to output
  - Connections parameterized by a weight matrix $V$

# Forward Propagation

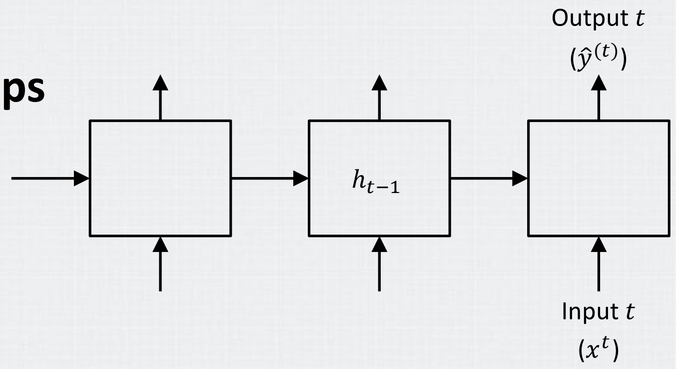
## Recurrent Neural Networks



- Forward propagation starts with specifying the initial state $h^{(0)}$

- For each time step from $t = 1$ to $t = \tau$
  - $a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$
  - $h^{(t)} = tanh\left(a^{(t)}\right)$
  - $o^{(t)} = c + Vh^{(t)}$
  - $\hat{y}^{(t)} = softmax(o^{(t)})$

- The matrices $\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}$ **are shared across all time steps**
  - Vanishing and exploding gradient problem



# Loss Function

## Recurrent Neural Networks



- The total loss for a given sequence of $t$ $(x^{(1)}, \dots x^{(t)})$ values paired with a sequence of $y$ values would then be just the sum of the losses over all the time steps
  - Assuming a recurrent network that maps an input sequence to an output sequence of the same length

- $L\left(\left\{x^{(1)}, \dots x^{(t)}\right\}, \left\{y^{(1)}, \dots y^{(t)}\right\}\right) = \sum_{i=0}^{t} L^{(i)}$

- Computing the gradient of this loss function w.r.t the parameters is computationally expensive
  - The runtime (and memory cost) is $O(t)$
  - Parallelization cannot reduce this cost because the forward propagation graph is **inherently sequential**: each time step must be computed after the previous one

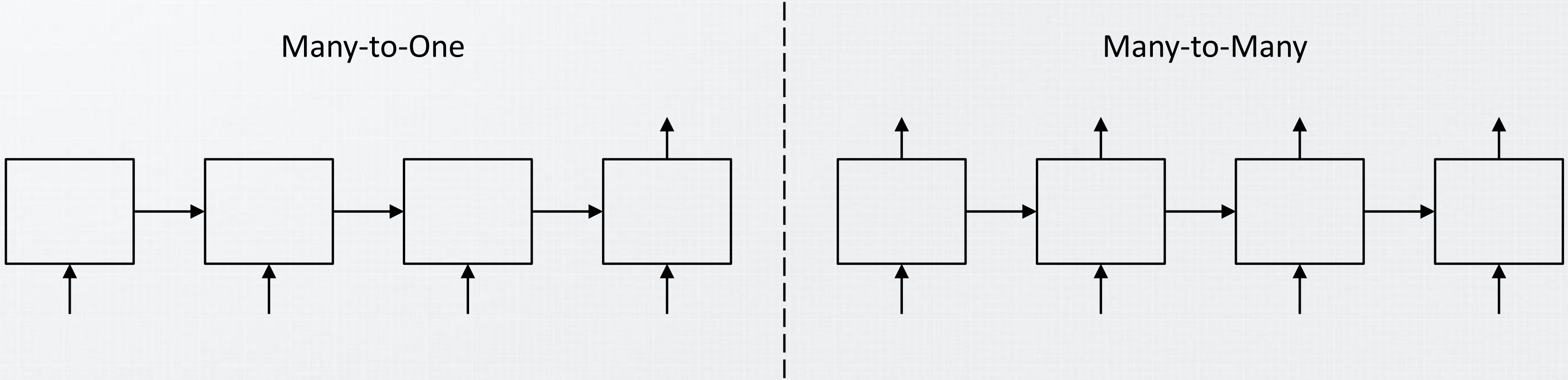
# Architectural Designs

## Recurrent Neural Networks



- Recurrent networks that produce an output at each time step
  - Many-to-many

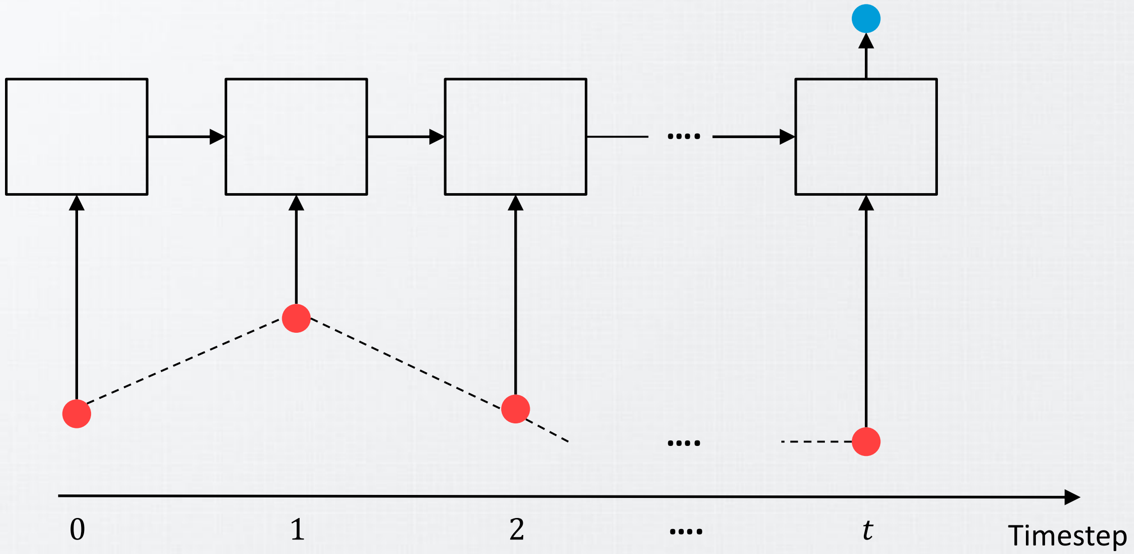
- Recurrent networks that read an entire sequence and then produce a single output
  - Many-to-one



# Architectural Design

## Recurrent Neural Networks





# Variants of Recurrent Networks

# Introduction

## Variants of RNNs



- RNNs are known to forget information, preventing the modeling of **long-range relationships**

- The problems of learning long-term dependencies
  - Vanishing and exploding gradients

- Memory cell
  - It retains information over time

- Gated RNNs
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Units (GRU)

# Long Short-Term Memory (LSTM)

## Variants of RNNs



- Gating units
    - Vectors that control the flow of information in the LSTM via element-wise multiplication of the corresponding information vector

- The values for the gating units are always in the range $[0,1]$ and are obtained as the outputs of a sigmoid function applied to the current input and the previous hidden state

# Long Short-Term Memory (LSTM)

## Variants of RNNs



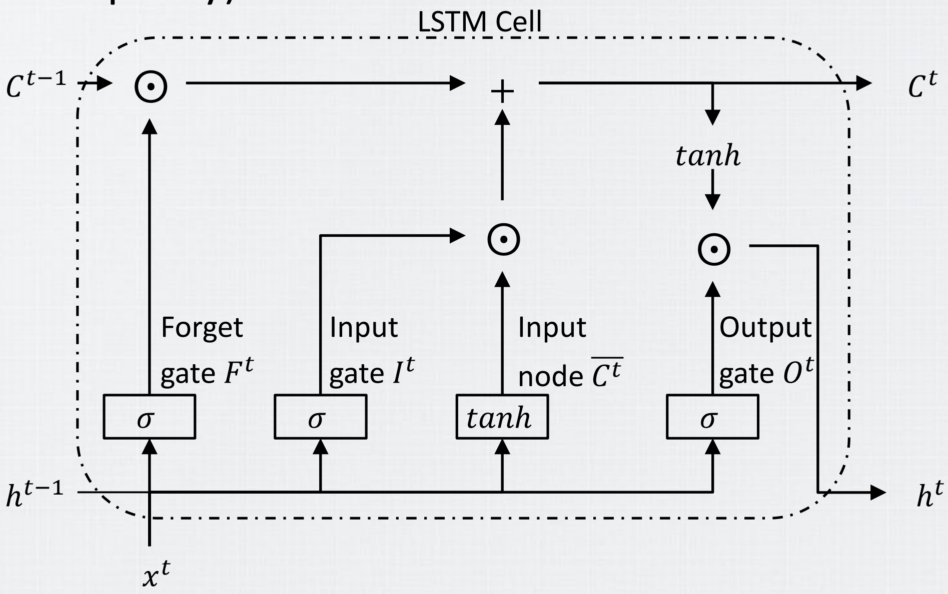
- Input Gate ($I^t$)
  - Determines how much of the input node's value should be added to the current memory cell's internal state
- Forget Gate ($F^t$)
  - Determines whether each element of the memory cell' is remembered (copied to the next time step) or forgotten (reset to zero)
- Output Gate ($O^t$)
  - Determines whether the memory cell should influence the output at the current time step

# Long Short-Term Memory (LSTM)

## Variants of RNNs



- The update is (the bias is omitted for simplicity):
  - $I^t = \sigma(W_I x^t + W_{hI} h^{(t-1)})$
  - $F^t = \sigma(W_F x^t + W_{hF} h^{(t-1)})$
  - $O^t = \sigma\left(W_O x^t + W_{hO} h^{(t-1)}\right)$



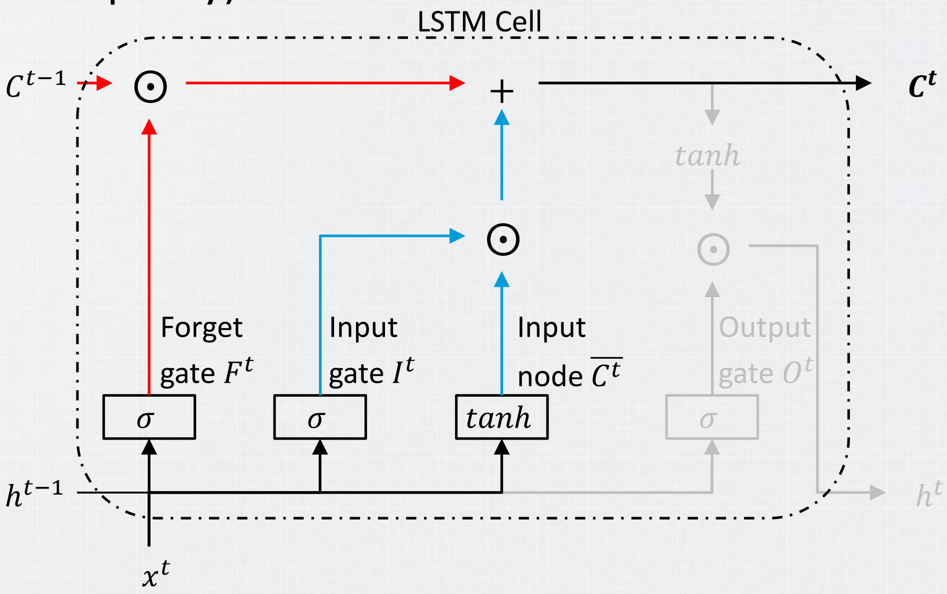

- Input node
  - $\overline{C^t} = \tanh(W_c x^t + W_{hc} h^{(t-1)})$

- Memory Cell Internal State
  - $C^t = F^t \odot C^{(t-1)} + I^t \odot \overline{C^t}$

# Long Short-Term Memory (LSTM)

## Variants of RNNs

- The update is (the bias is omitted for simplicity):
  - $I^t = \sigma(W_I x^t + W_{hI} h^{(t-1)})$
  - $F^t = \sigma(W_F x^t + W_{hF} h^{(t-1)})$
  - $O^t = \sigma(W_O x^t + W_{hO} h^{(t-1)})$




- Input node
  - $\overline{C^t} = \tanh(W_c x^t + W_{hc} h^{(t-1)})$

- Memory Cell Internal State
  - $C^t = \underbrace{F^t \odot C^{(t-1)}}_{} + \underbrace{I^t \odot \overline{C^t}}_{}$

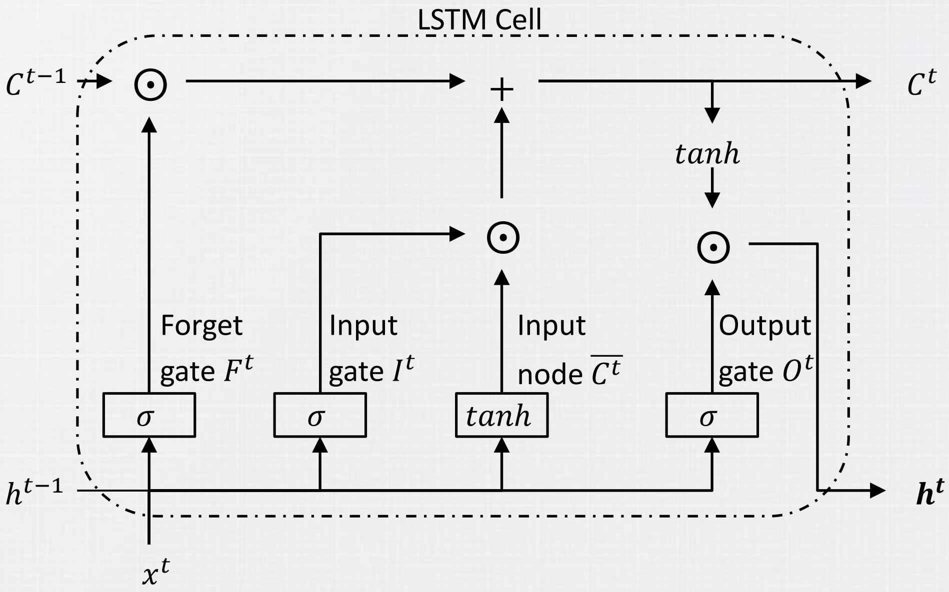
addresses how much of the old cell internal state $C^{(t-1)}$ we retain

governs how much we take new data into account via $\overline{C^t}$

# Long Short-Term Memory (LSTM)

## Variants of RNNs

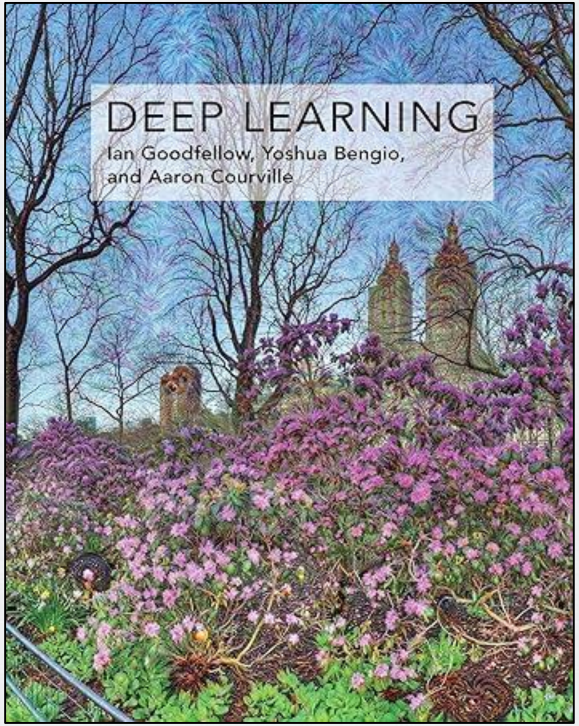
- Hidden State
  - $h^t = O^t \odot \tanh(C^t)$



# Bibliography

# Bibliography

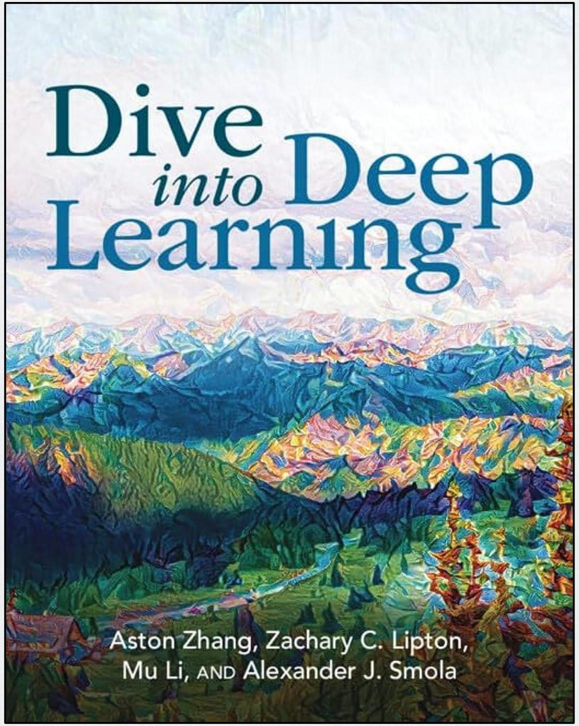
- Deep Learning
  - Chapter 10
    - 10.2 Recurrent Neural Network



# Bibliography




- Dive into Deep Learning
  - Chapter 9
    - 9.4.2 Recurrent Neural Networks with Hidden States

# Bibliography



- The Hundred-page Machine Learning Book
  - Chapter 6
    - 6.2.2 Recurrent Neural Network