Politecnico di Bari

DIPARTIMENTO INTERATENEO DI FISICA

Dottorato in Fisica – XXXIX ciclo - 2024

Machine Learning techniques for particle physics

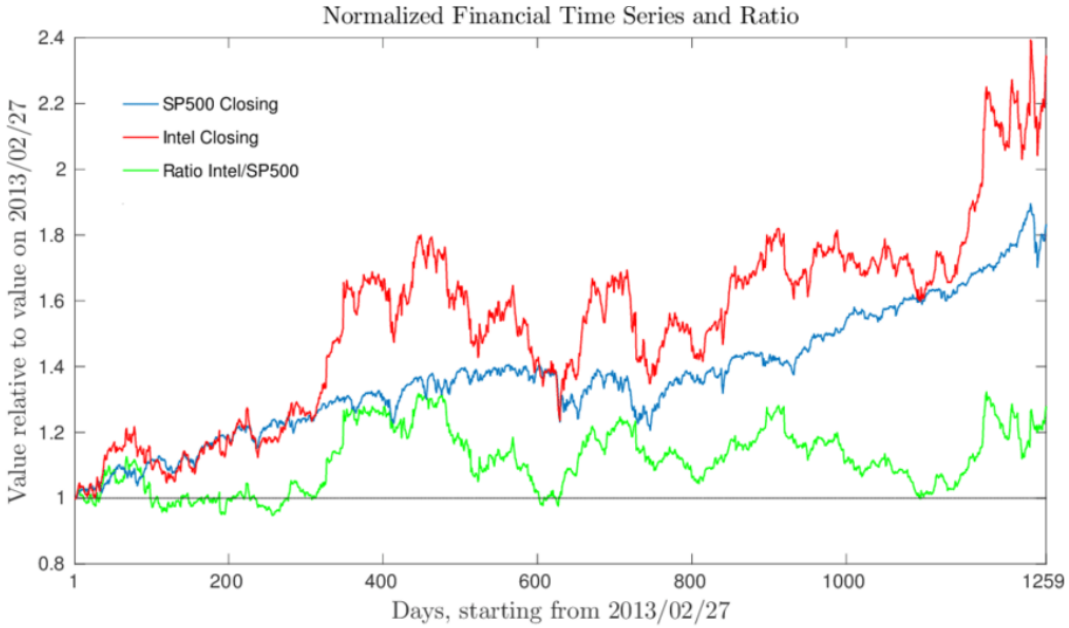
Federica Maria Simone - federica.simone@poliba.it

RNN: Recurrent Neural Networks

New problem: make predictions/classifications based on sequential data, i.e. **vectors or ordered elements**

Represent:

- Time series:
  - Financial data, stock market
  - Healthcare: pulse rate, sugar level
- Text and speech
- Spatiotemporal data:
  - Self-driving and object tracking
  - Plate tectonic activity
- Physics:
  - Accelerator's, detector's data flows, anomaly detection
  - Tracking
- etc.



RNN: Recurrent Neural Networks

New problem: make predictions/classifications based on sequential data, i.e. **vectors or ordered elements**

- **Sequence classification**
- **Sequence labelling**
- **Sequence to sequence transformation**

Sequence classification

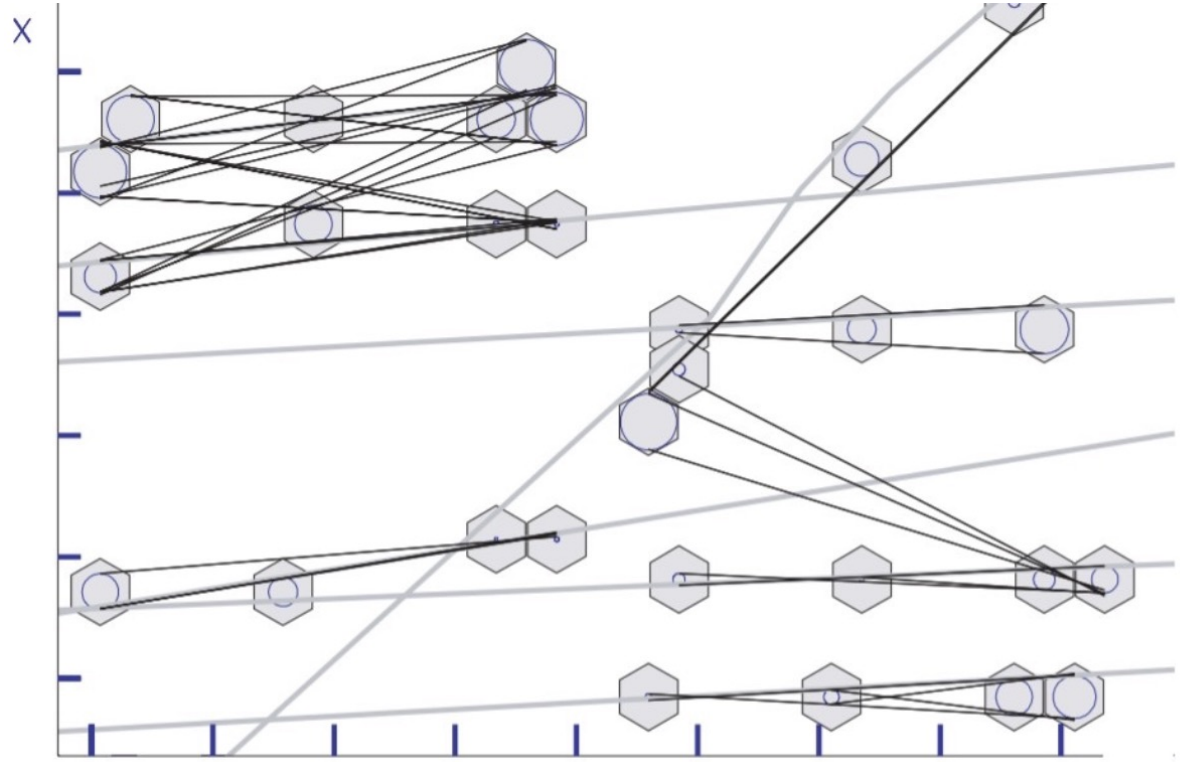
- Problem statement:
  - $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ - sequence of objects $x_i \in V$
  - $y \in \{1 \ldots C\}$ - labels (C is a number of classes)
  - $\{(\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2), \ldots, (\mathbf{x}^m, y_m)\}$ - training data of size m

Aim: predict y given

Examples:

- Medicine: **x** - pulse rate, y - health (healthy, unhealthy)
- Opinion mining: **x** - sentence, y - sentiment (positive, negative)
- **Physics**: ghost tracks filtering. **x** - sequence of hits for some track, y - class (ghost or not)

Example of sequence classification: ghost tracks filtering



Problem: which set of hits form a true track, and which are just a random track-like set of hits (ghost)?

Sequence labelling

Problem statement:

- $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ - sequence of objects $x_i \in V$
- $y = \{y_1, y_2, \ldots, y_n\}$ - **sequence** of labels, $y_i \in \{1 \ldots C\}$ (C is a number of classes)
- $\{(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)\}$ - training data of size m
- **Aim**: predict y given x

Examples:

- Genome annotation: **x** - DNA, y - genes
- Hardware: **x** are system logs, y is system state at each time
- **Physics**: track reconstruction. **x** - set of hits, y - set of tracks (each hit is assigned to some track)

Example of sequence labelling: track reconstruction

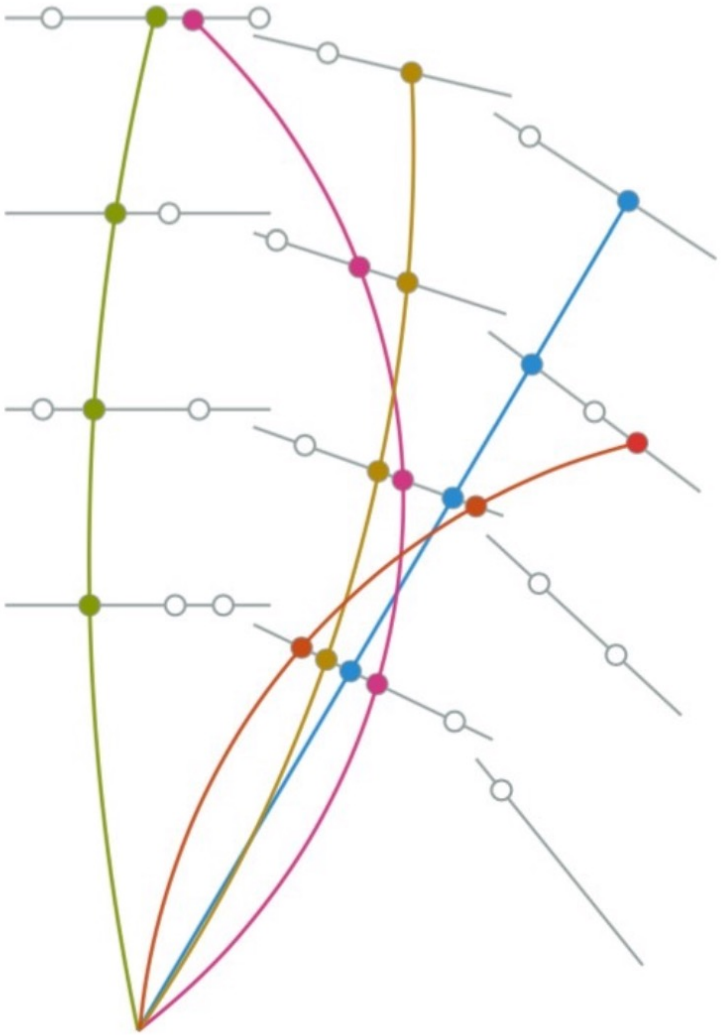


Figure: http://old.inspirehep.net/record/1729722/plots

Sequence to sequence transformation

- Problem statement:
  - $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ - source sequence of objects $x_i \in V_{source}$
  - $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ - target sequence of objects $y_i \in V_{target}$
  - $\{(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)\}$ - training data of size m
  - **Aim**: fit transformation from x to y Examples:
  - Speech to text - **x** is speech, y is text
  - Machine translation - **x** are sentences in English, y are sentences in German
  - Logs compression - **x** are raw logs, y are compressed logs

Example of sequence to sequence transformation

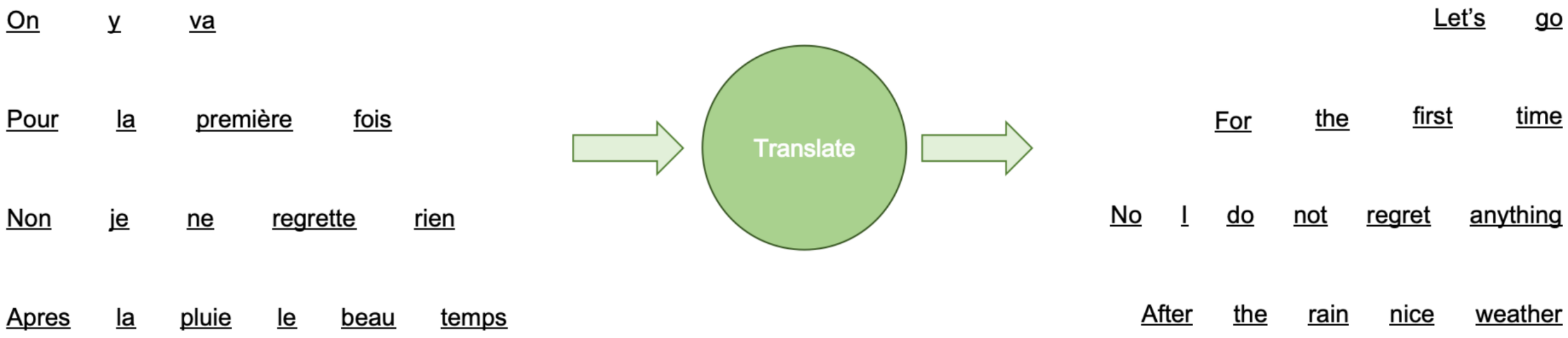
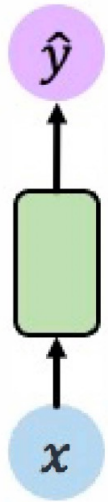


Figure: https://buomsoo-kim.github.io/attention/2020/01/12/Attention-mechanism-3.md/

Sequence Modeling Applications

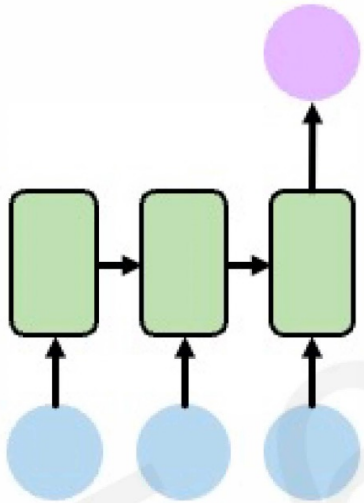


One to One

Binary Classification

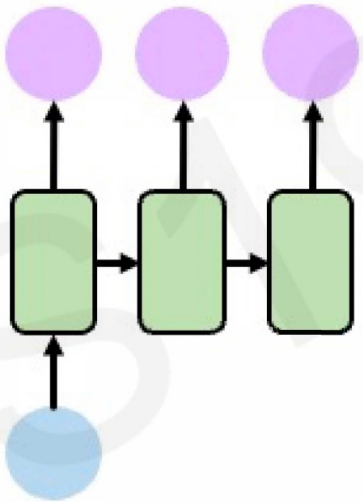
"Will I pass this class?"

Student → Pass?



Many to One

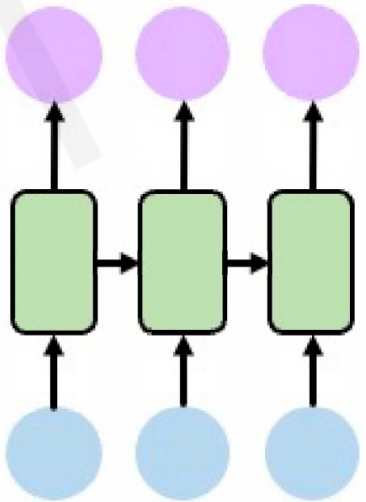
Sentiment Classification



One to Many

Image Captioning

"A baseball player throws a ball."



Many to Many

Machine Translation

The method: recurrent neural networks

RNN's are good at processing sequence data for predictions

Try to say the alphabet in your head.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Now try saying the alphabet backward.

ZYXWVUTSRQPONMLKJIHGFEDCBA

Start at the letter Q:

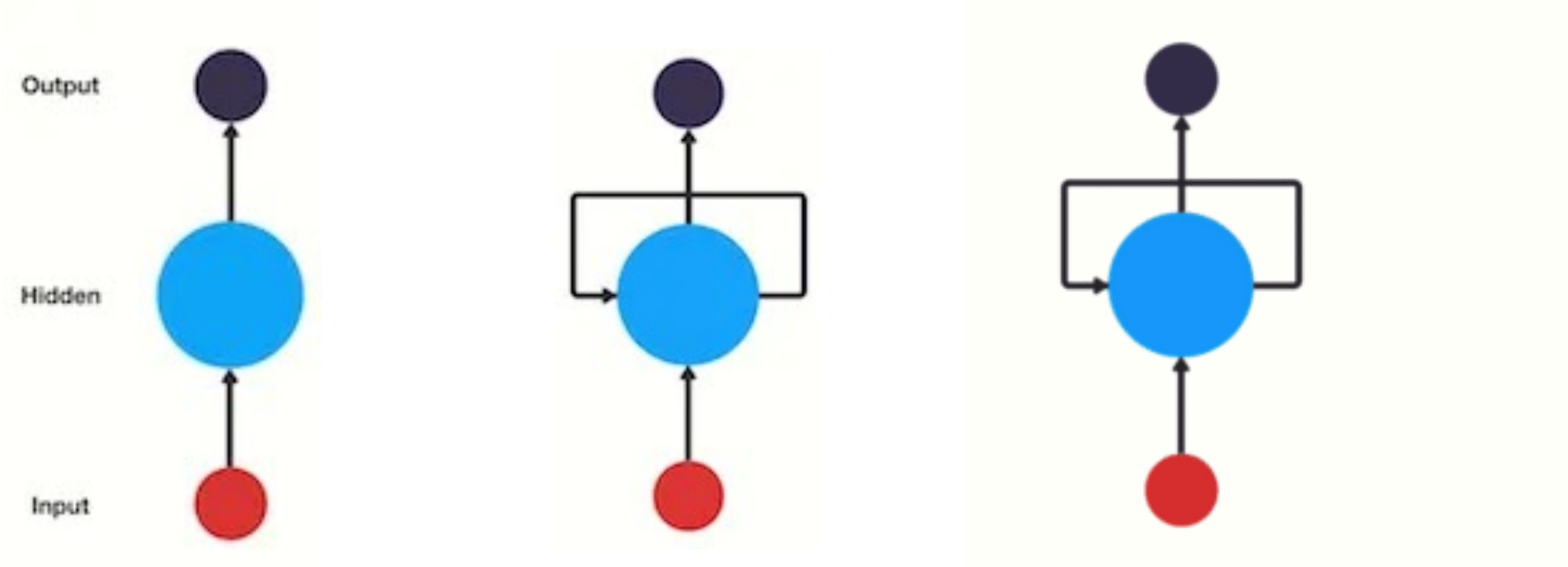
Q

At first, you'll struggle with the first few letters, but then after your brain picks up the pattern, the rest will come naturally. **Sequential memory is a mechanism that makes it easier for your brain to recognize sequence patterns.**

The method: recurrent neural networks

Question: How to perform elementwise processing of sequential data effectively preserving its context?

Answer: Let's keep all the context in a separate term (hidden state).



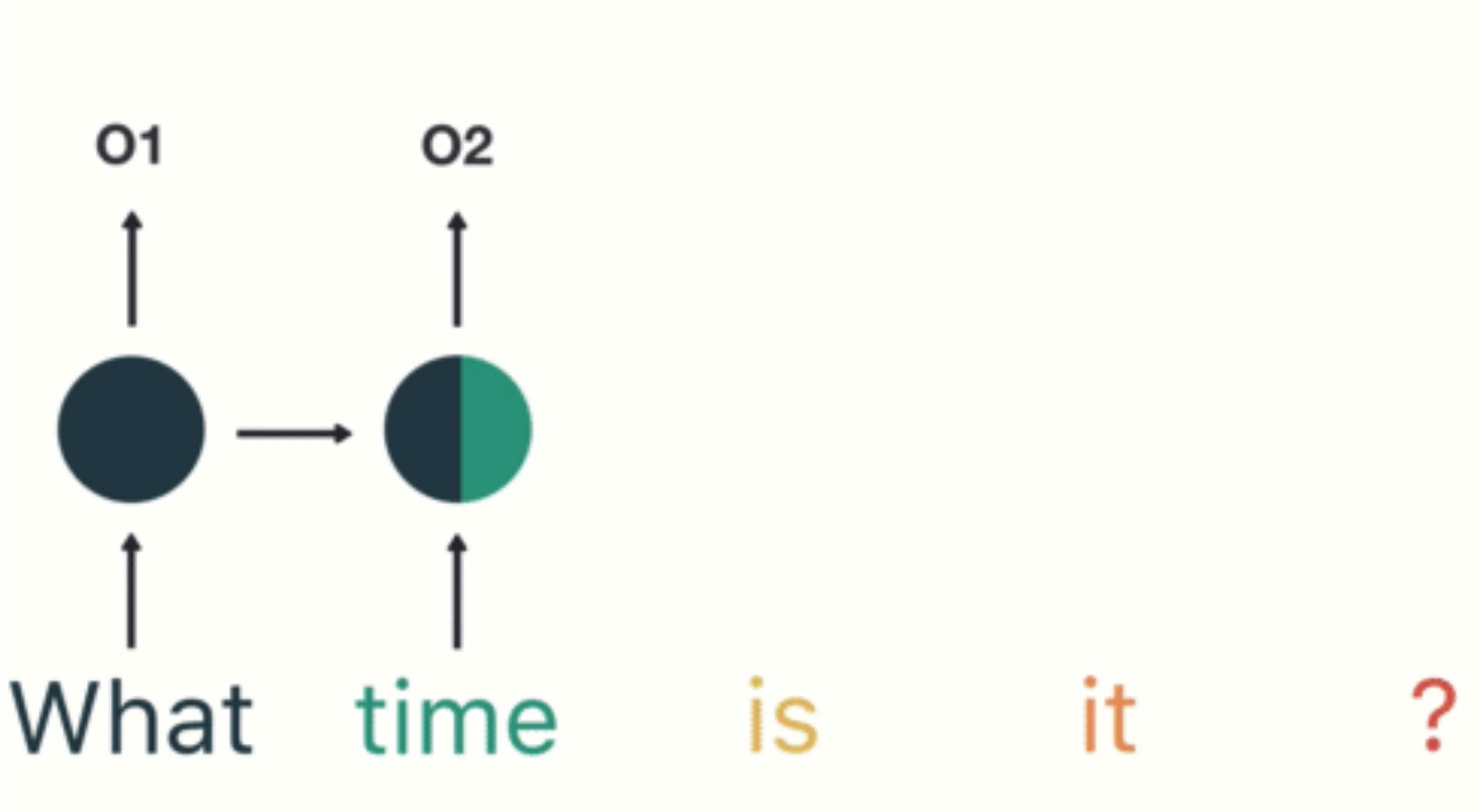
An RNN has a looping mechanism that acts as a highway to allow information to flow from one step to the next.

Intuition: string processing

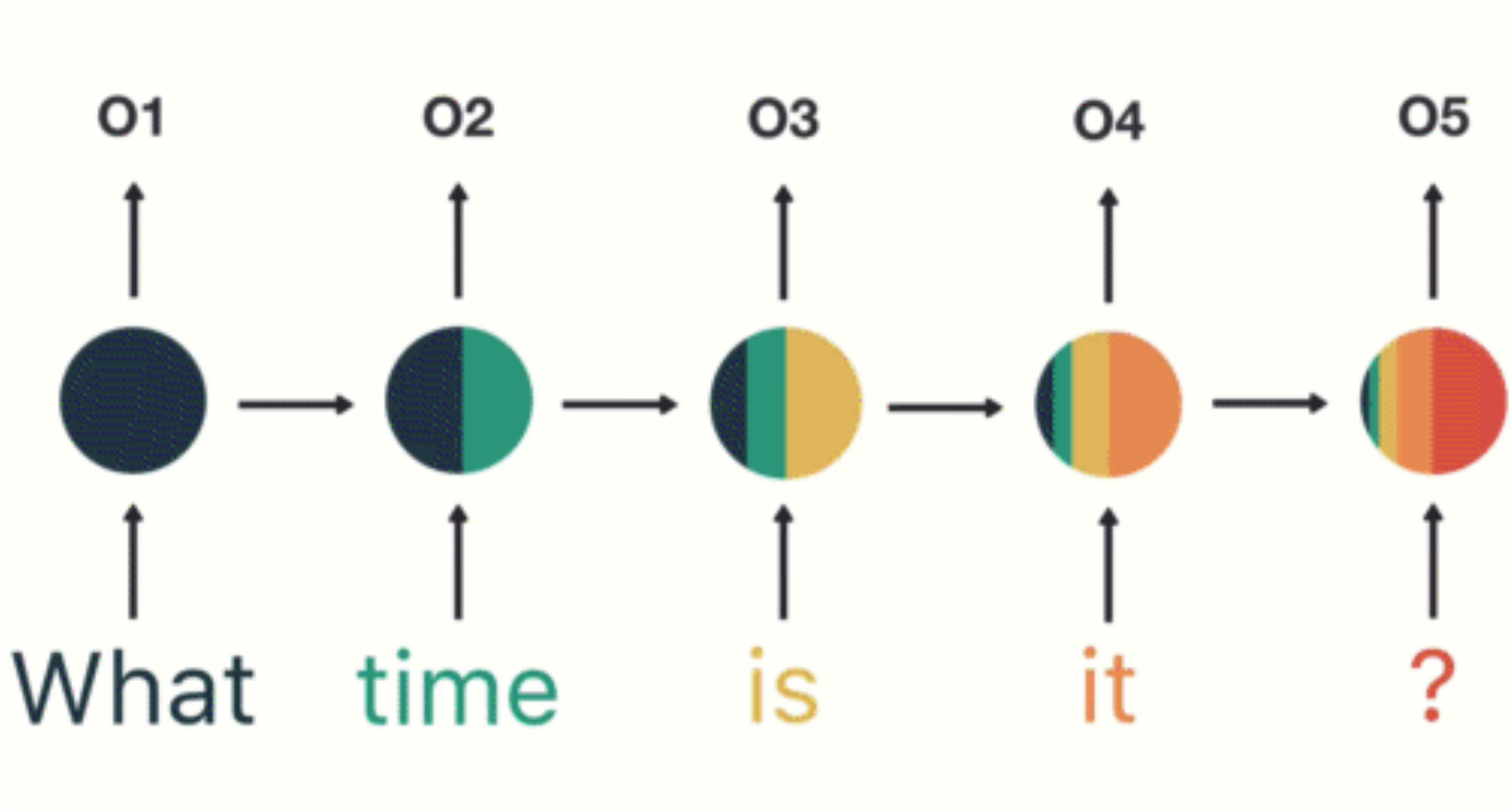
What time is it?

Intuition: string processing

Intuition: string processing



Intuition: string processing



One step back: perceptron

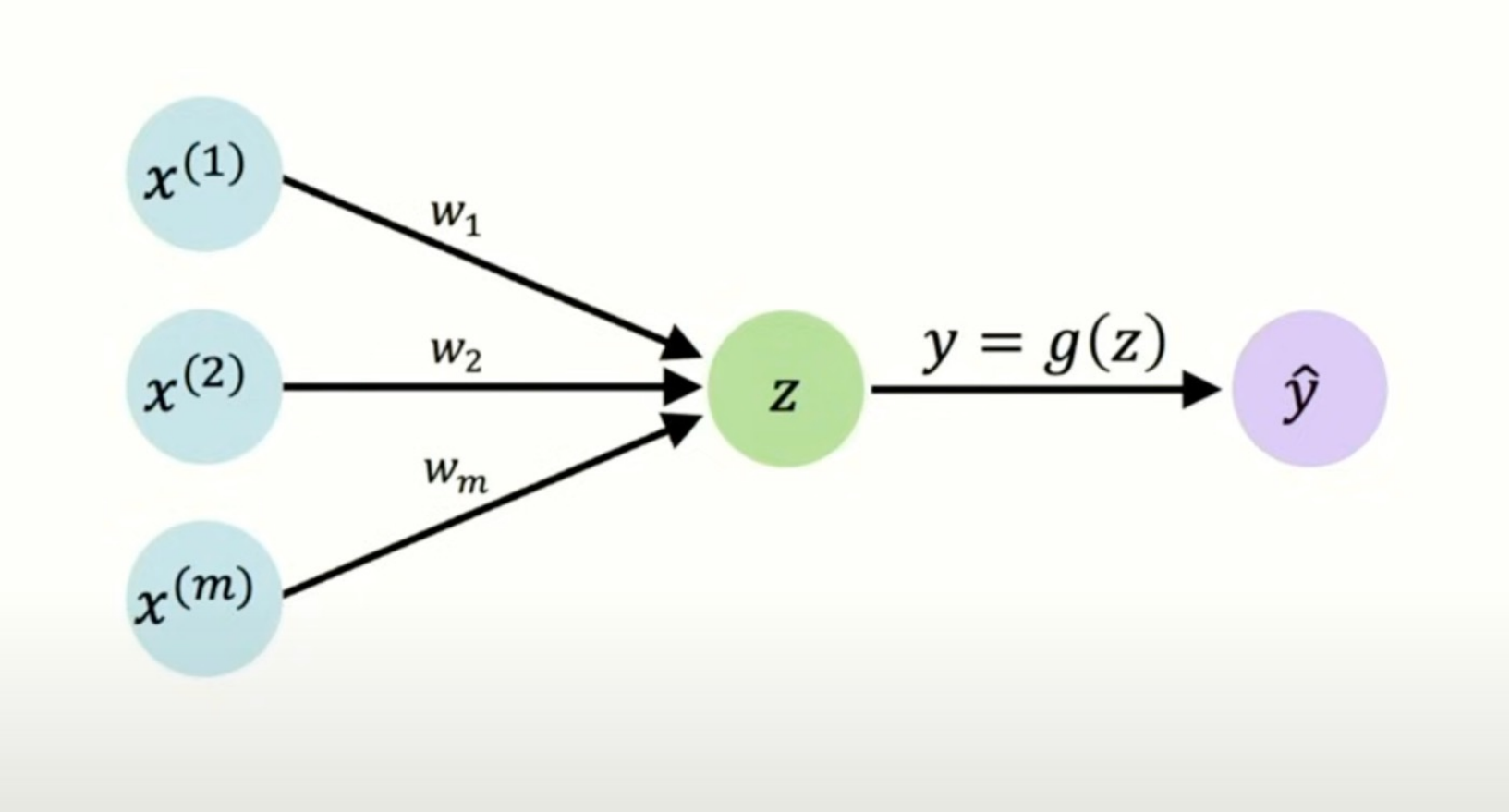


Image: http://introtodeeplearning.com

Feed forward Neural Network

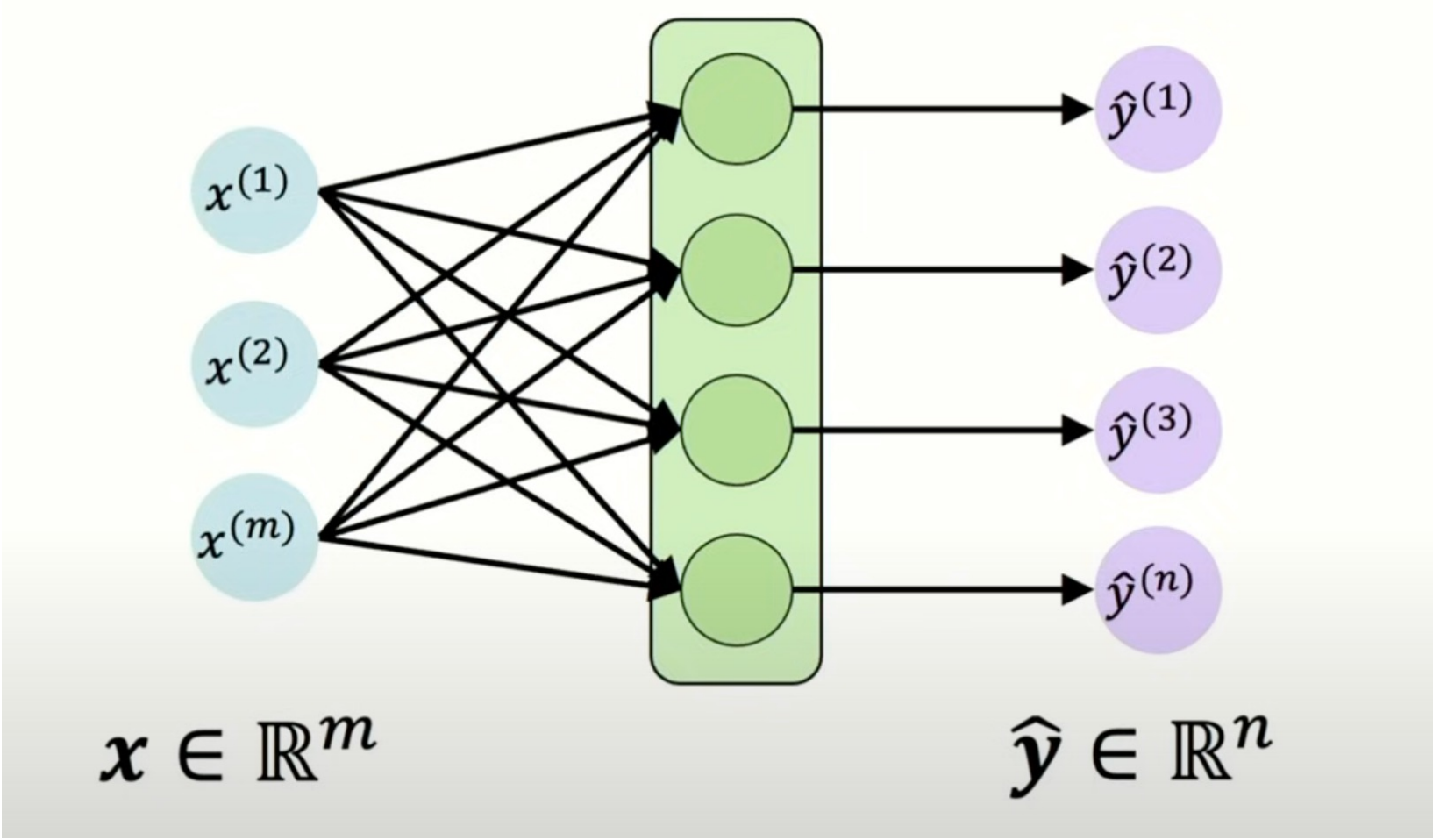
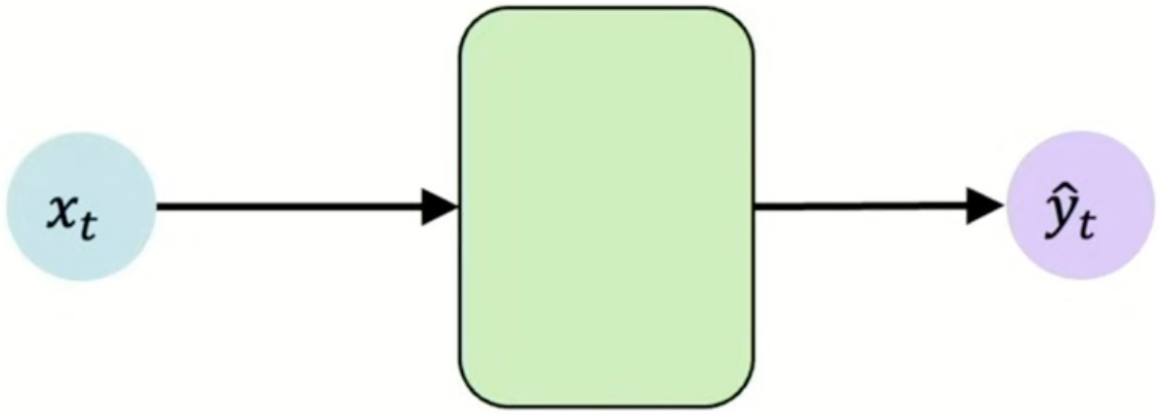


Image: http://introtodeeplearning.com

Feed forward Neural Network revisited



$$\boldsymbol{x_t} \in \mathbb{R}^m \qquad \hat{\boldsymbol{y}}_t \in \mathbb{R}^n$$

Image: http://introtodeeplearning.com

Handling individual time steps

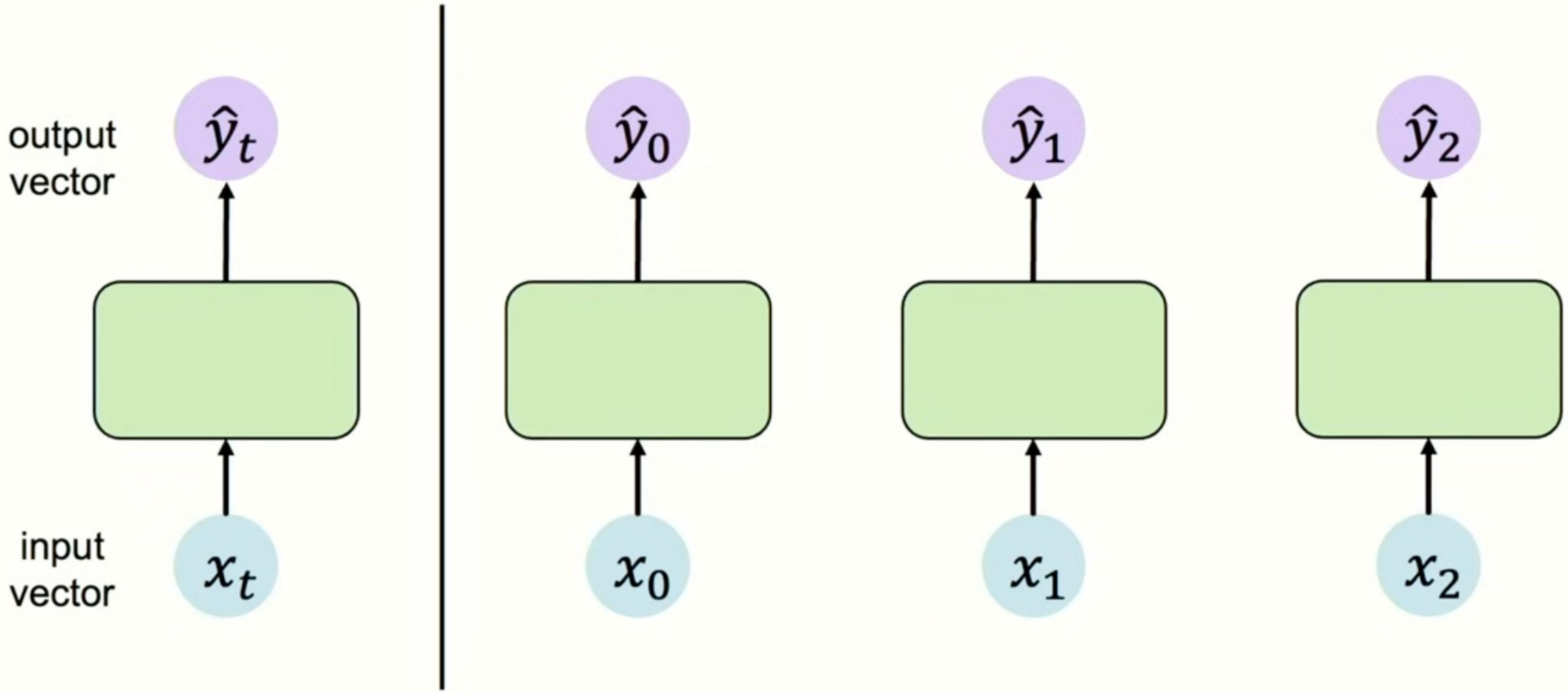


Image: http://introtodeeplearning.com

Handling individual time steps

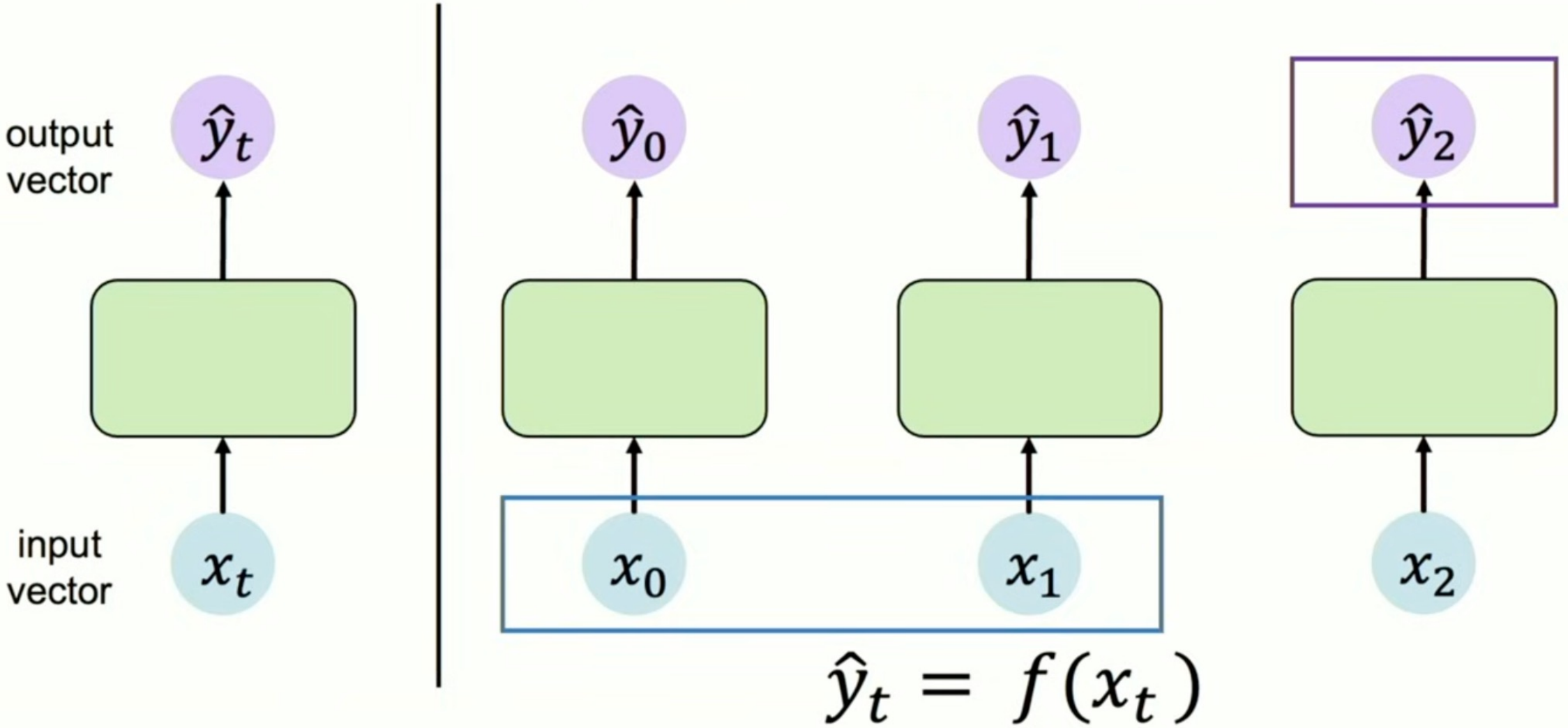


Image: http://introtodeeplearning.com

Handling individual time steps

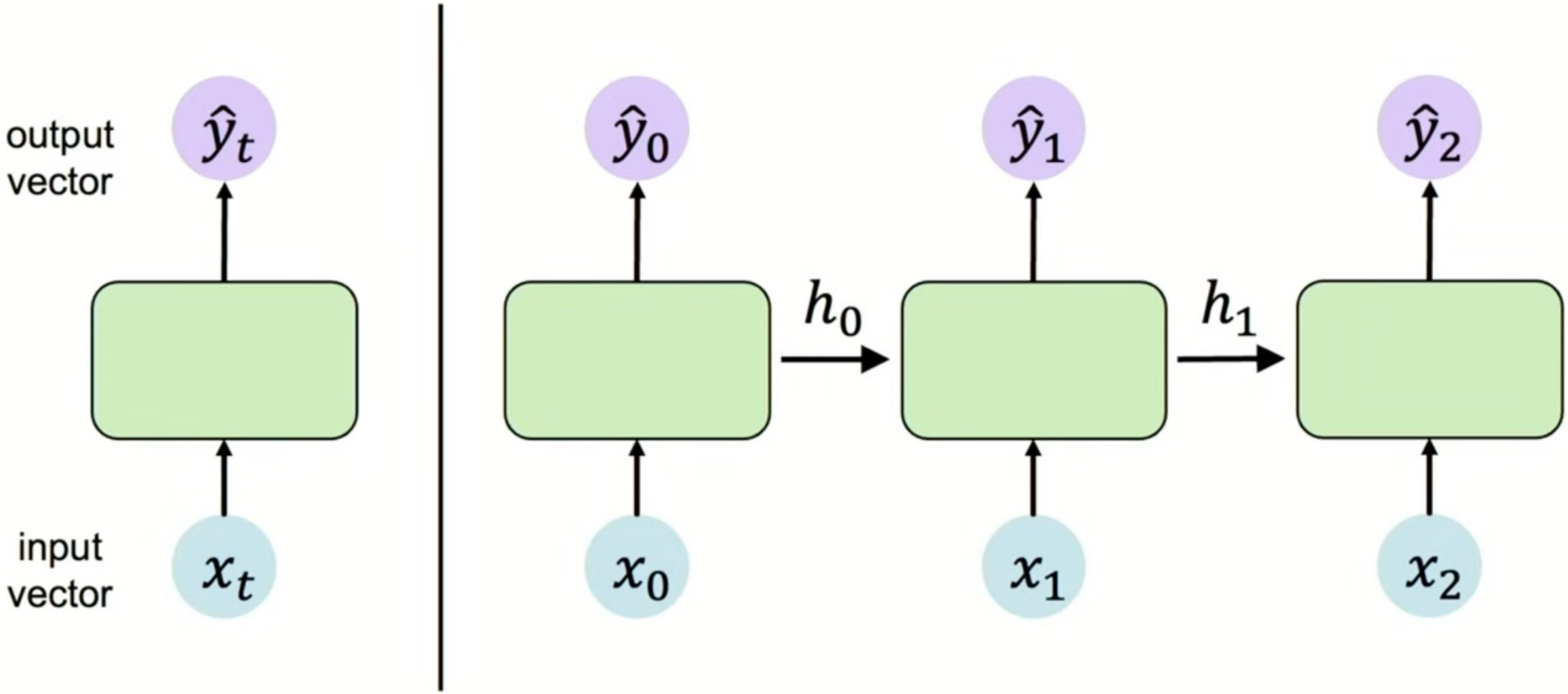
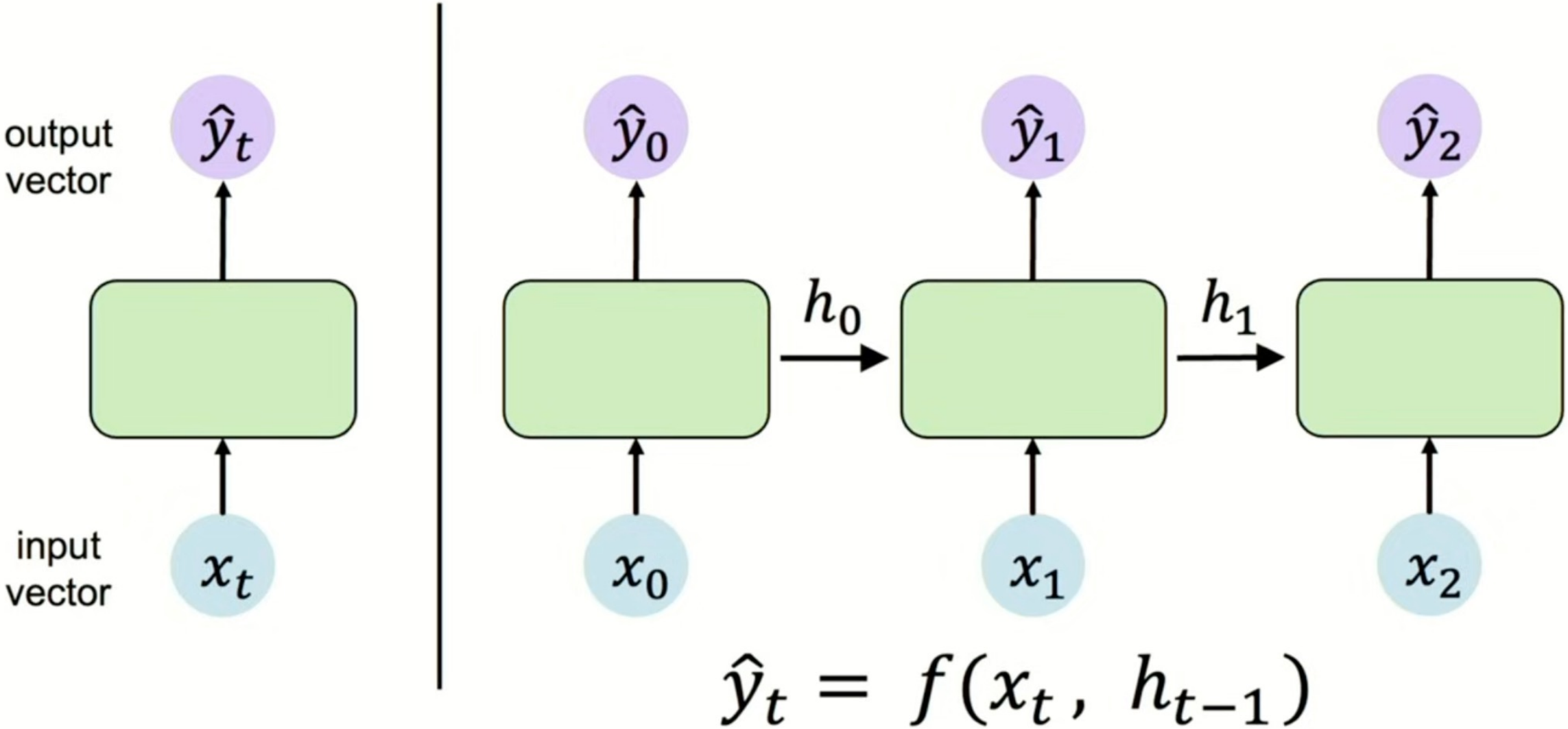


Image: http://introtodeeplearning.com

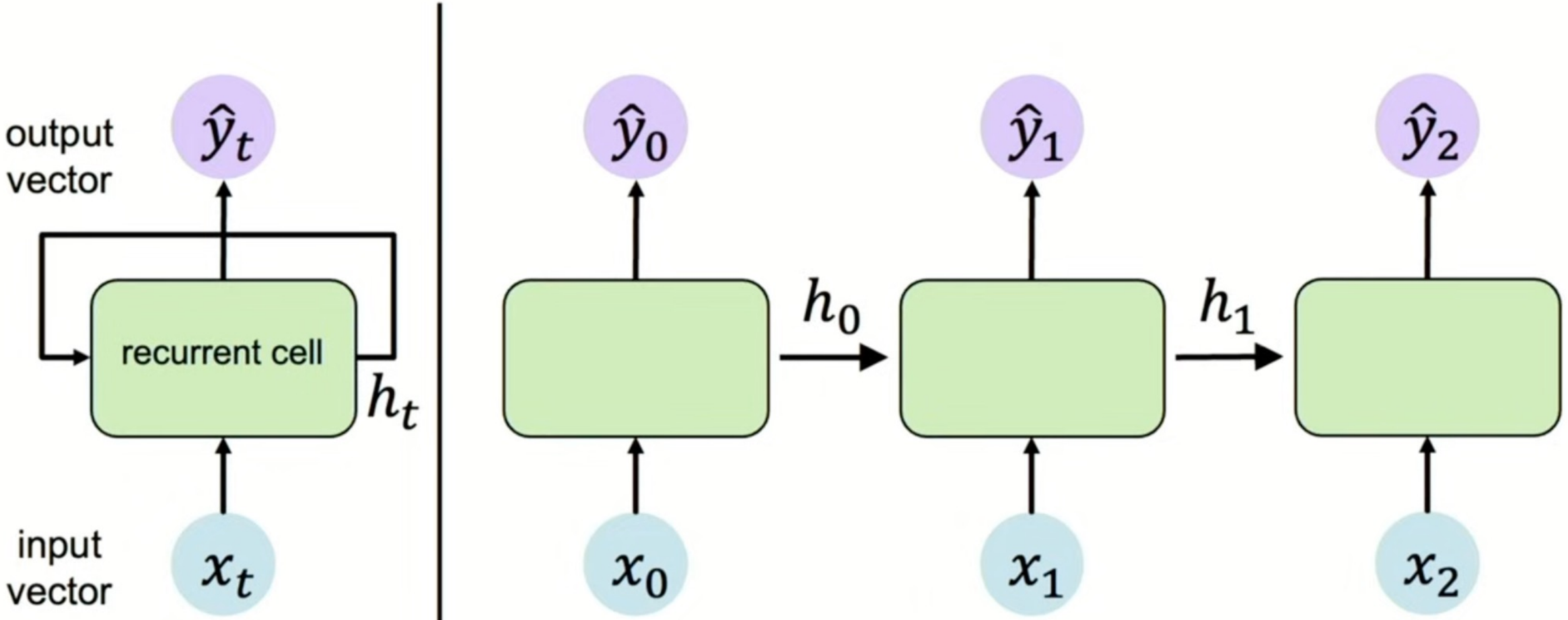
Handling individual time steps



$$\hat{y}_t = f(x_t, h_{t-1})$$

Image: http://introtodeeplearning.com

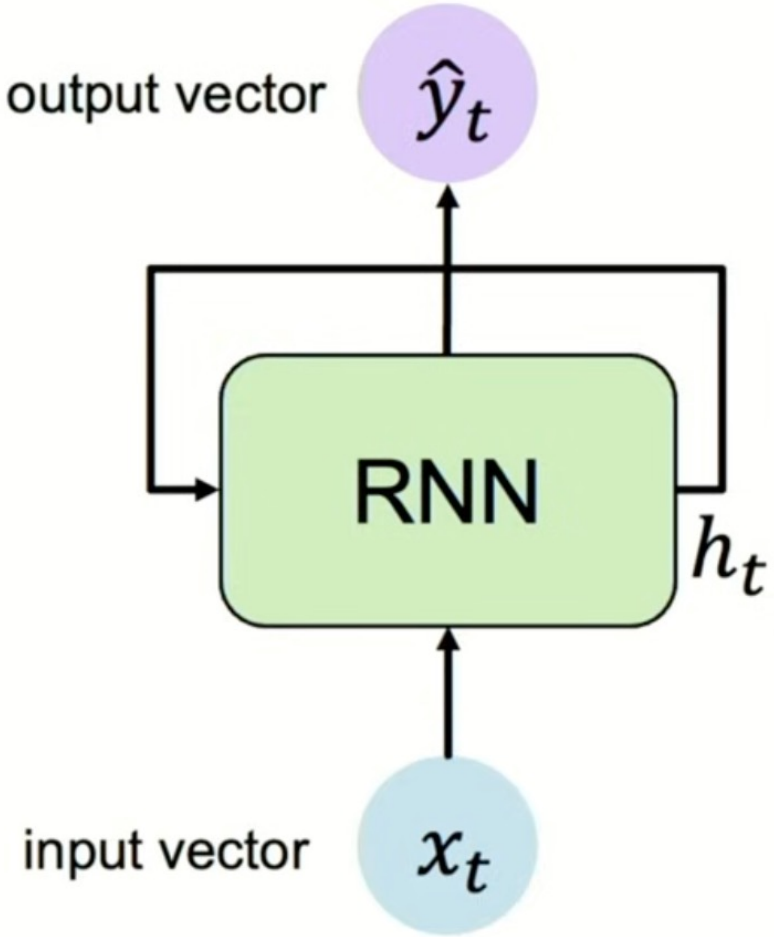
Recurrence



$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

Image: http://introtodeeplearning.com

Recurrence



Apply a **recurrence relation** at every time step to process a sequence:

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

cell state

function with weights W

input

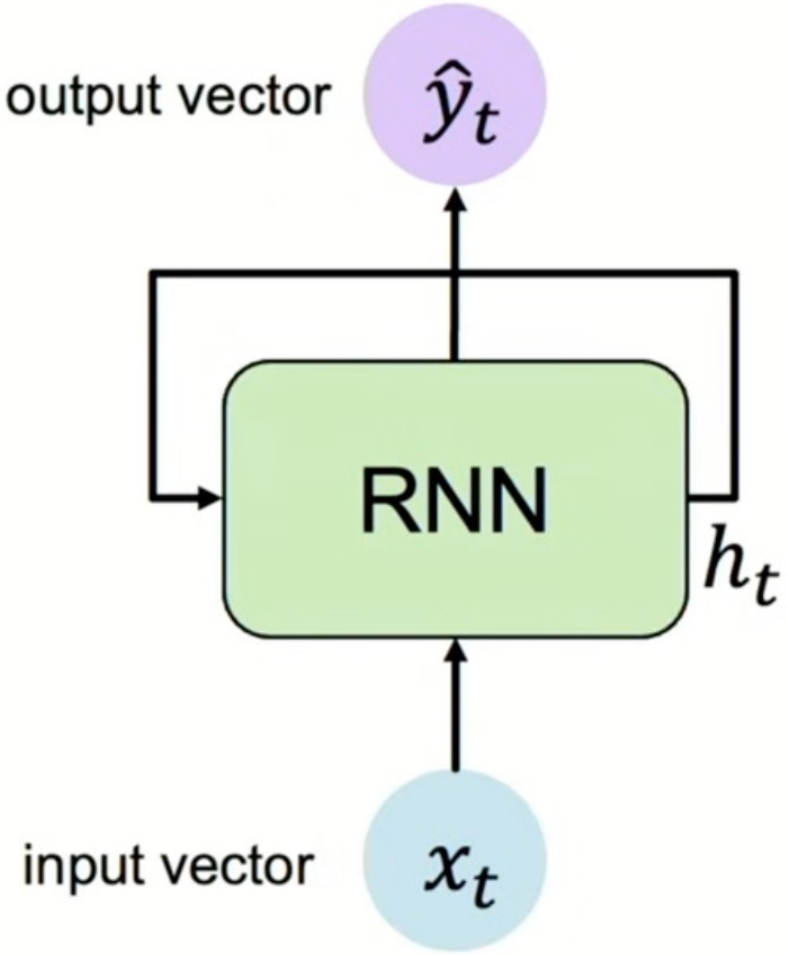
old state

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

RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

Image: http://introtodeeplearning.com

Recurrence



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

$$x_t$$

Image: http://introtodeeplearning.com

“Unfolding” the recurrent NN into a computational graph

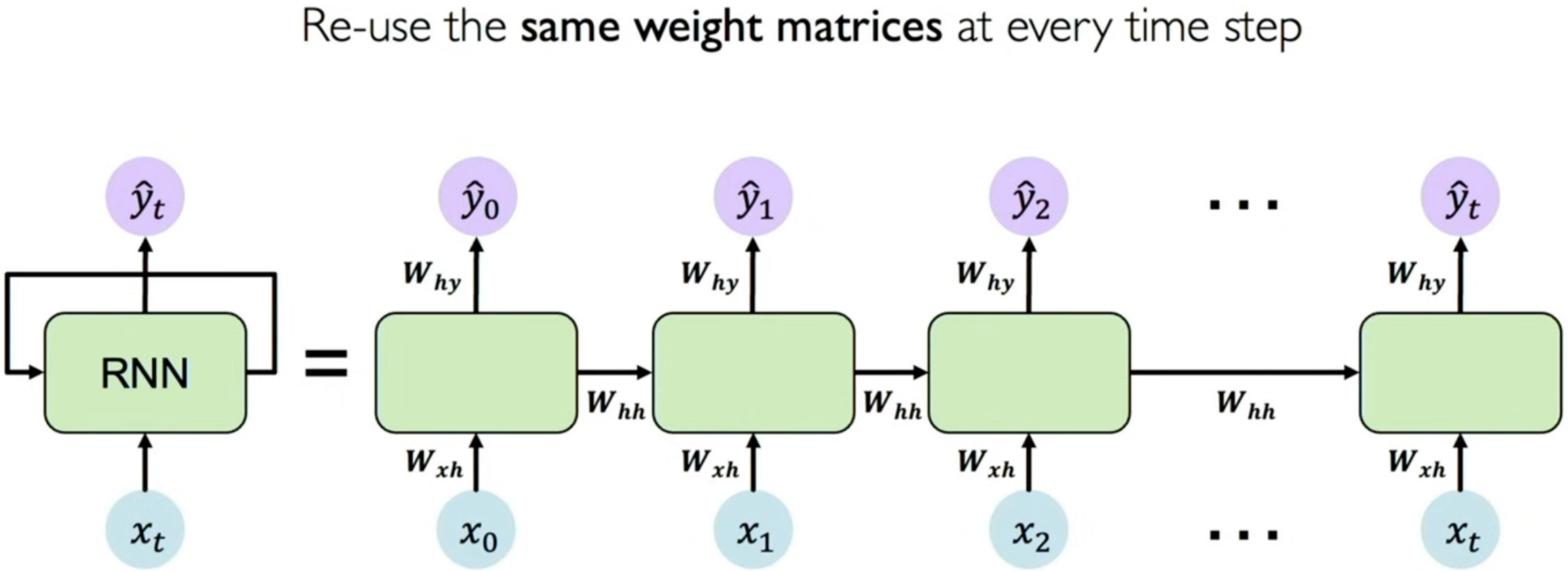


Image: http://introtodeeplearning.com

“Unfolding” the recurrent NN into a computational graph

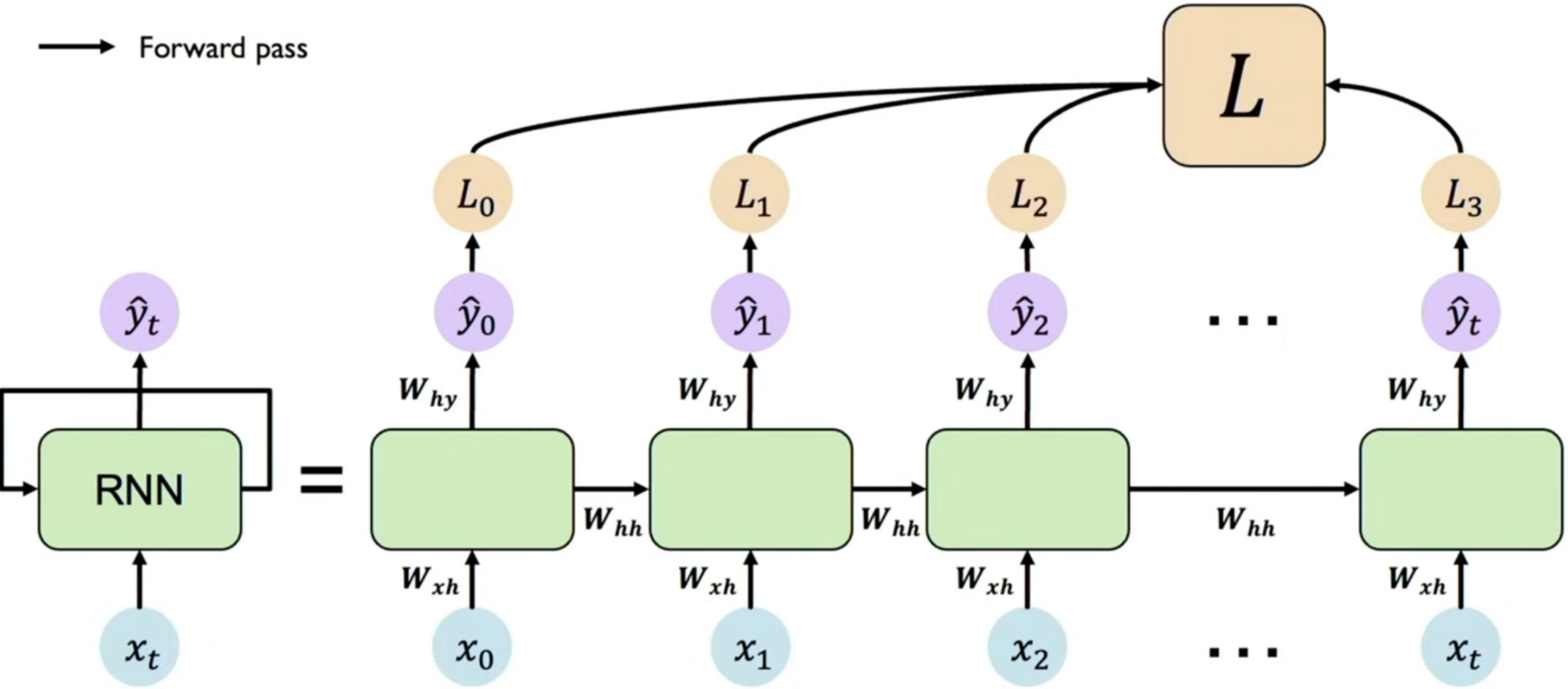
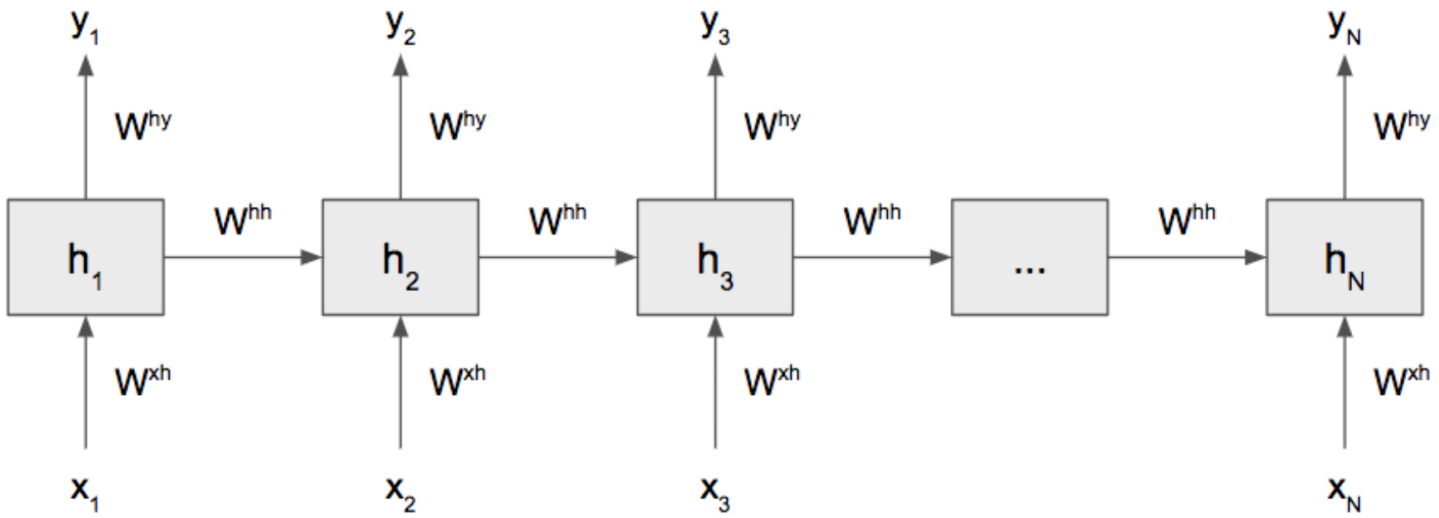


Image: http://introtodeeplearning.com

Recurrent Neural Networks

- Input: (x_i, h_i) from sequence of input vectors $x = \{x_1, x_2, \ldots, x_n\}$ and hidden states (context) $h = \{0, h_2, \ldots, h_{n-1}\}$, $x_i \in R^{d_{in}}$, $h \in R^{d_{hid}}$
- Output: (y_i, h_{i+1}) from sequence of output vectors $y = \{y_1, y_2, \ldots, y_n\}$ and next hidden states (context) $h = \{h_2, h_3 \ldots, h_n\}$, $y_i \in R^{d_{out}}$, $h \in R^{d_{hid}}$



Stacked RNN

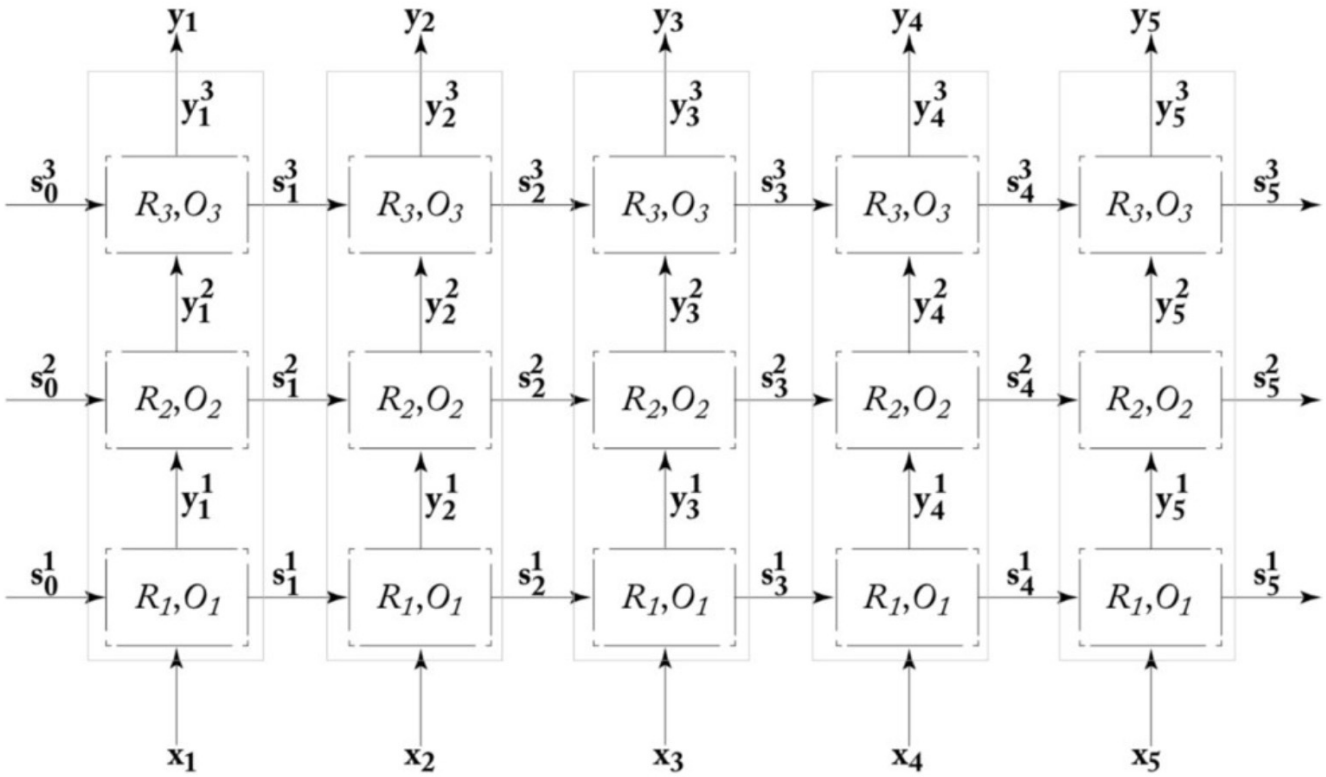


Figure: Goldberg, Yoav. Neural network methods for natural language processing

Apply backpropagation to RNN: recap on feed forward NN

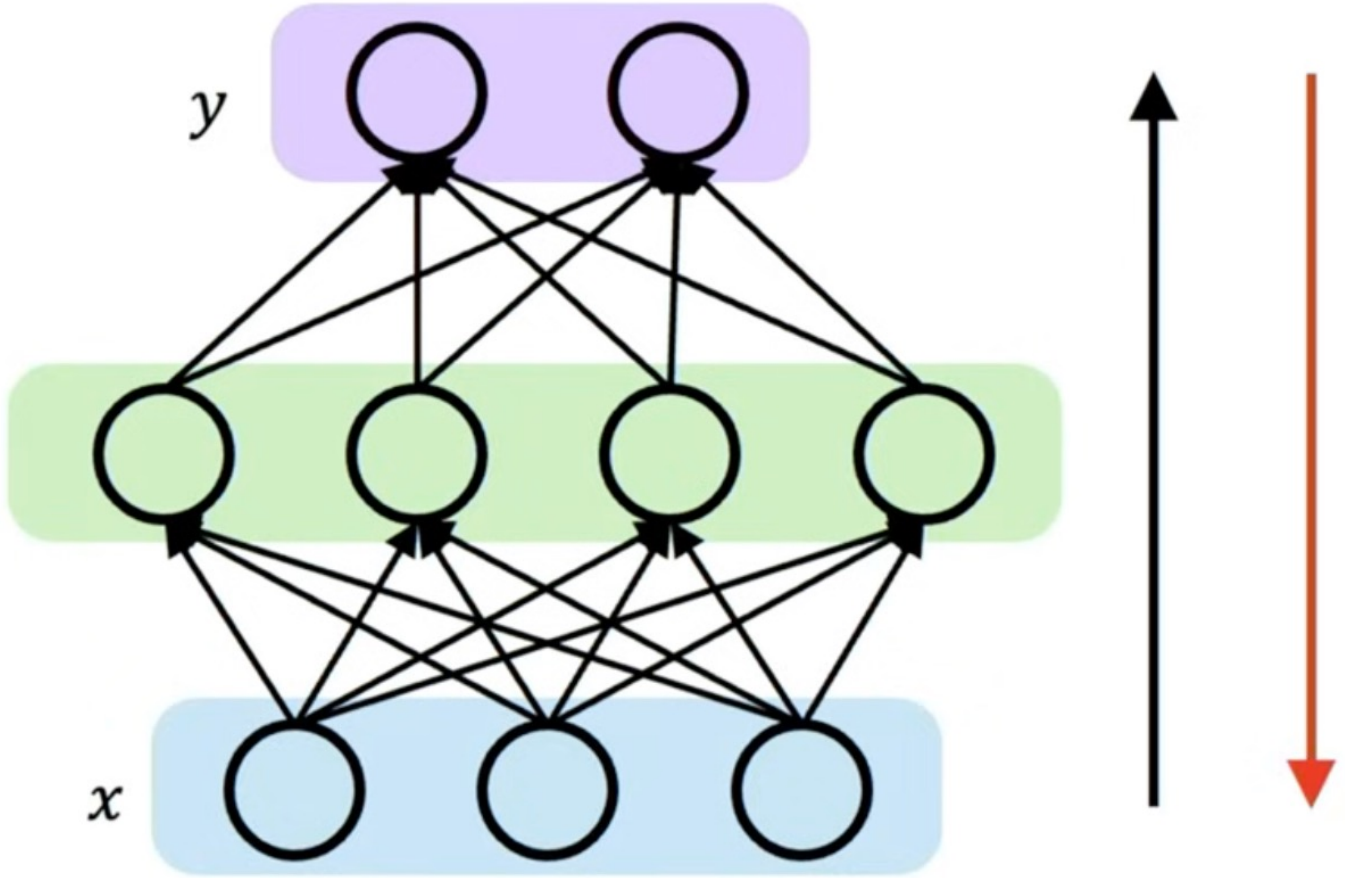
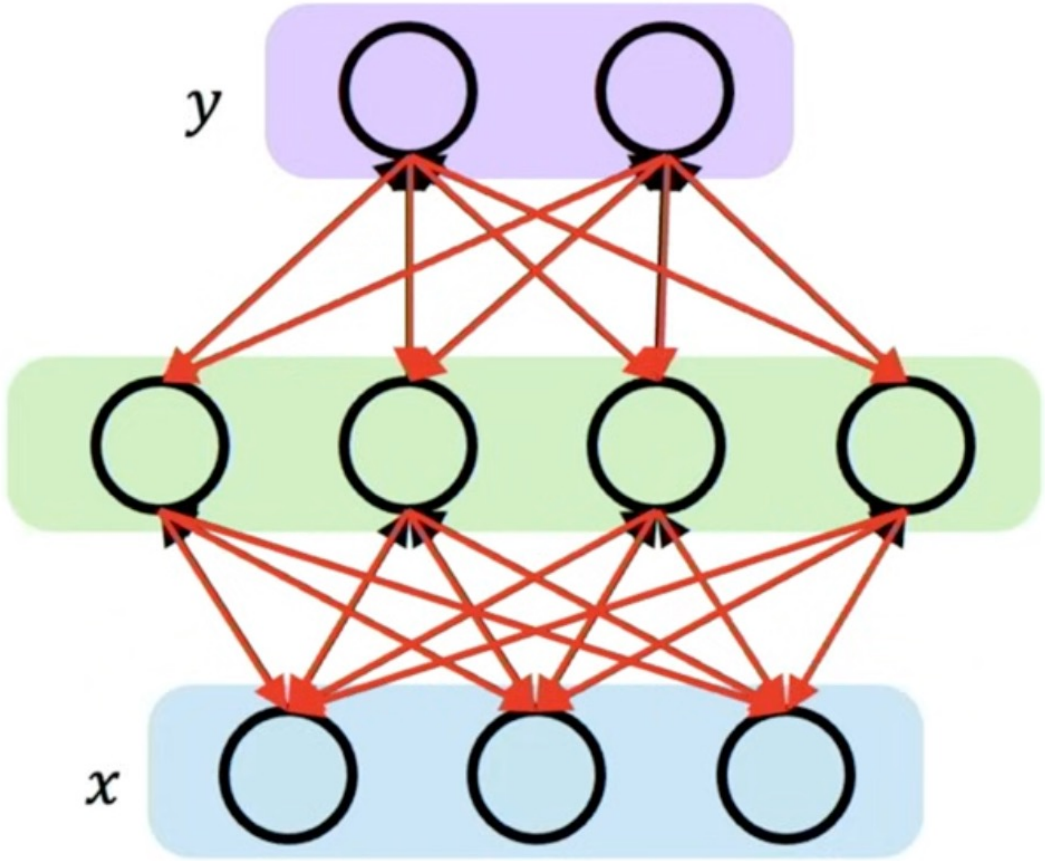


Image: http://introtodeeplearning.com

Apply backpropagation to RNN: recap on feed forward NN



Backpropagation algorithm:

1. Take the derivative (gradient) of the loss with respect to each parameter
2. Shift parameters in order to minimize loss

Image: http://introtodeeplearning.com

Backpropagation through time

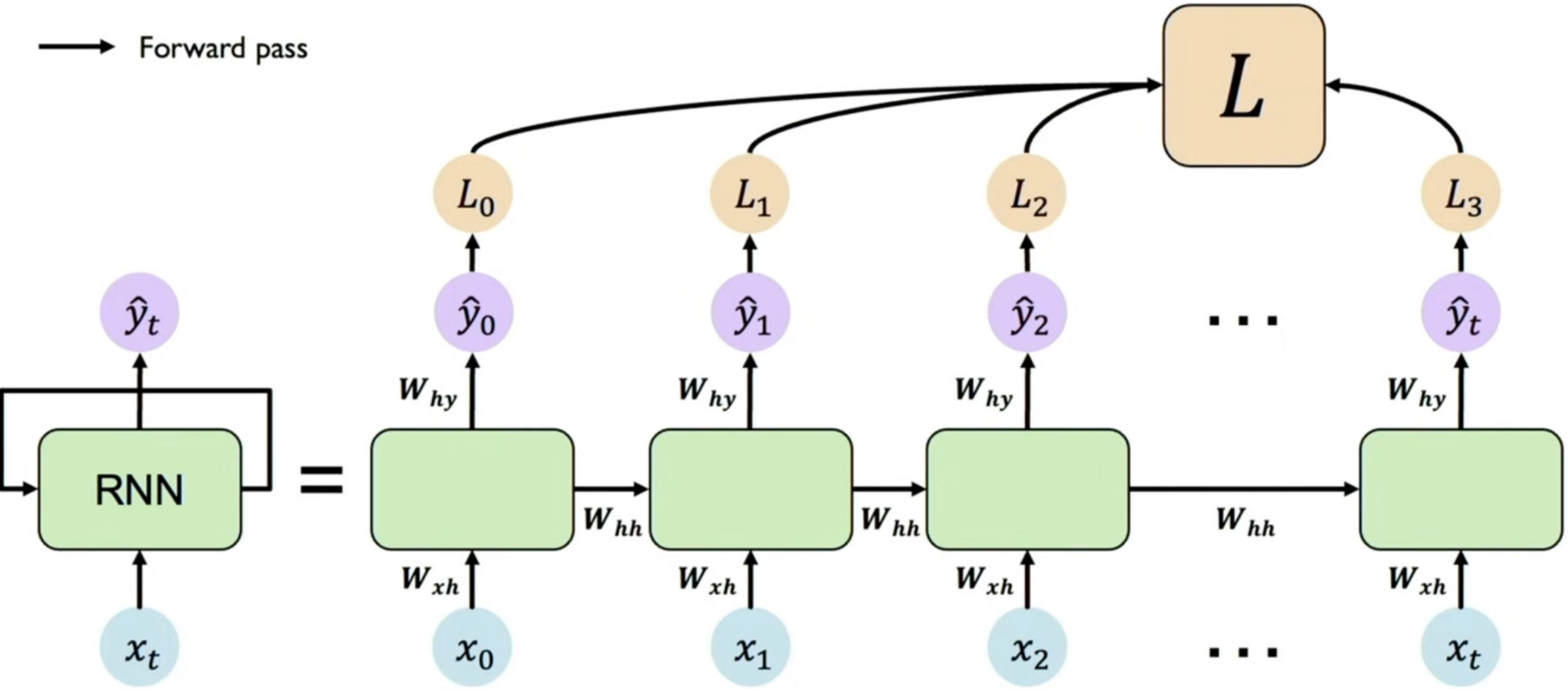


Image: http://introtodeeplearning.com

Backpropagation through time

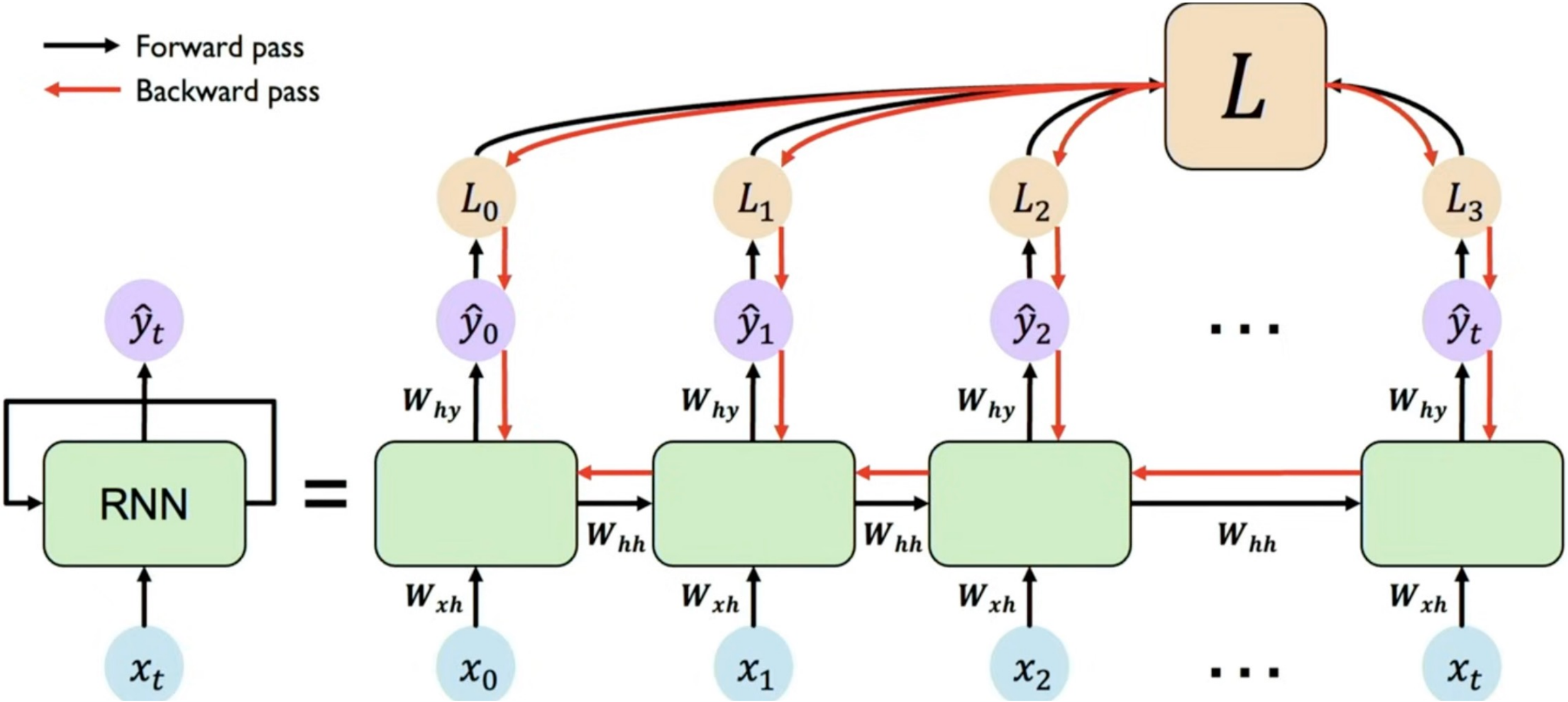
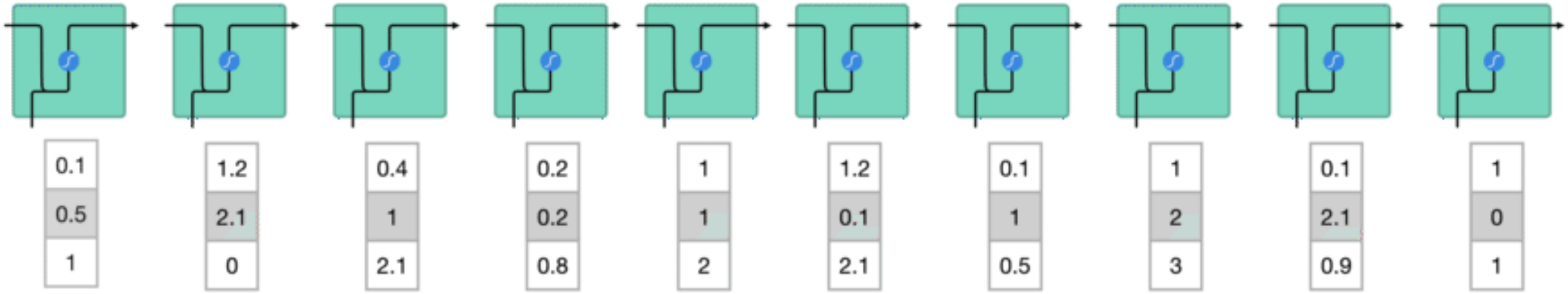


Image: http://introtodeeplearning.com

RNNs in Action

The RNN processes the sequence of vectors one by one.



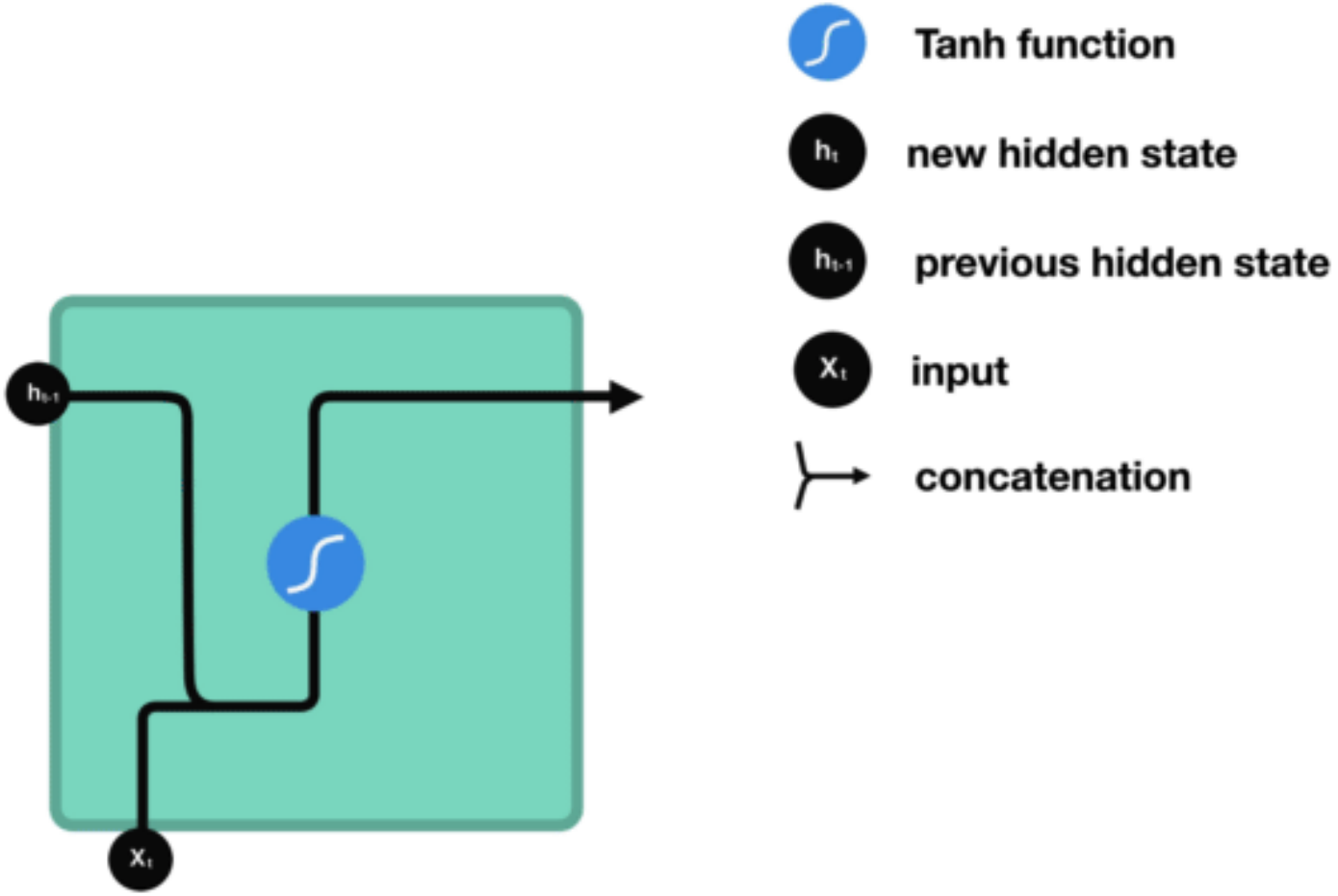
RNNs in Action

Let's look at a cell of the RNN to see how you would calculate the hidden state.

First, the input and previous hidden state are combined to form a vector.

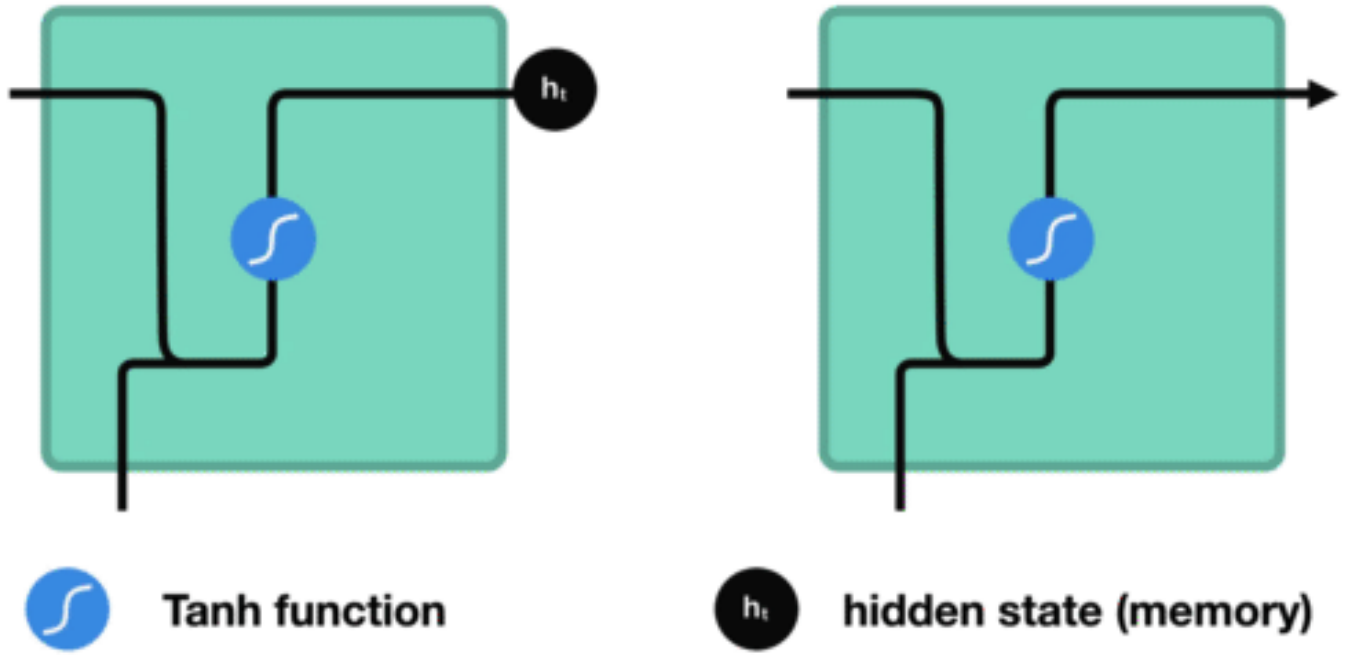
That vector now has information on the **current input and previous inputs**.

The vector goes through the tanh activation, and the **output is the new hidden state**, or the memory of the network.



RNNs in Action

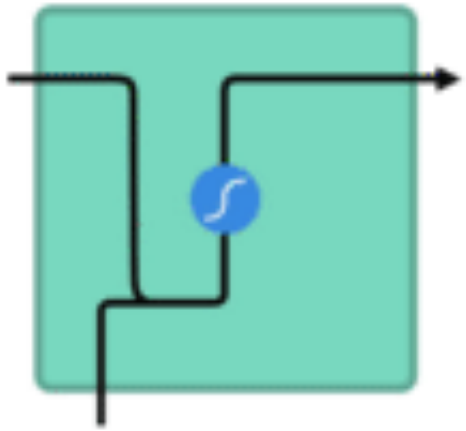
While processing, it passes the previous hidden state to the next step of the sequence.

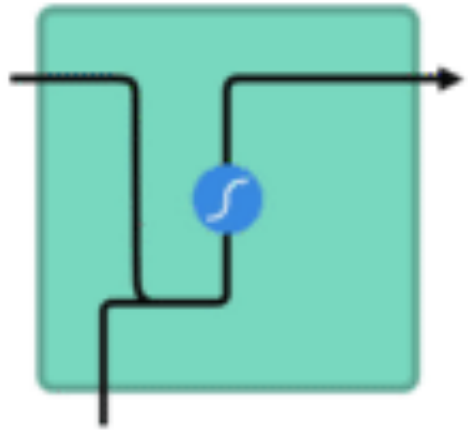


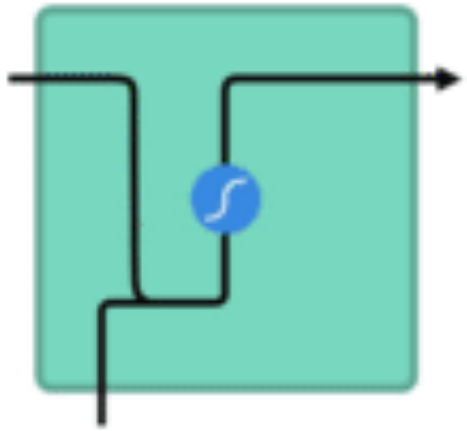
The hidden state acts as the neural networks memory. It holds information on previous data the network has seen before.

RNNs in Action

5
01
0.5







RNN vs. Fully-connected
Advantages and disadvantages

Advantages:

- Preserves sequence context
- Good performance and flexibility
- Effectively handles longer sequences (longer memory)
- Handles sequences with variable size
- Model complexity does not depend on sequence length

Disadvantages:

- **Moves in sequence in the only direction** (Some sequences must be processed in both directions)
- Still has short memory due to vanishing gradients problems
- Exploding gradients due to cycles in recurrent connections

Vanishing gradient and short memory

The Final hidden state of the RNN has a very small contribution from the early layers.

Then the earlier layers fail to do any learning as the internal weights are barely being adjusted due to extremely small gradients. The gradient values will exponentially shrink as it propagates through each time step.

Practically: there is a possibility that the word “what” and “time” are not used.

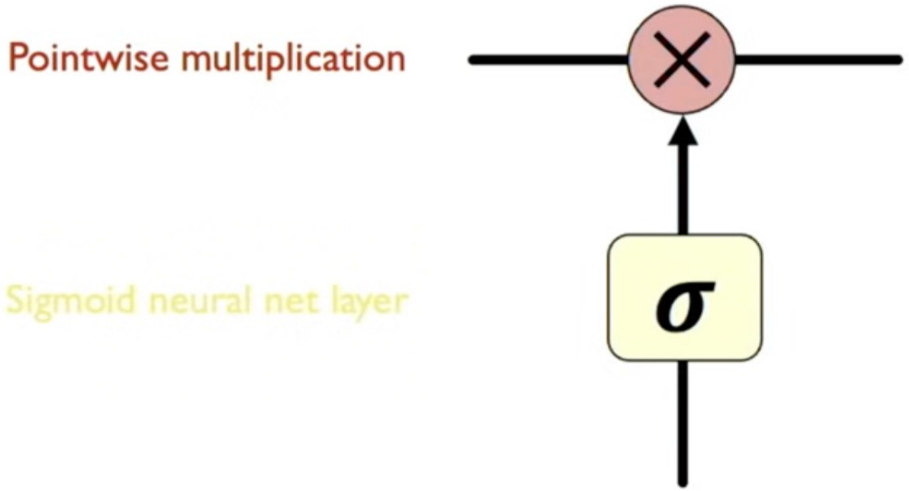
The network then has to make the best guess with “is it?”.

The network has a short-term memory.

Vanishing gradient and short memory

Mitigation strategies:

- Choice of activation function
- Parameter initialization (i.e. identity matrix)
- Gated cells

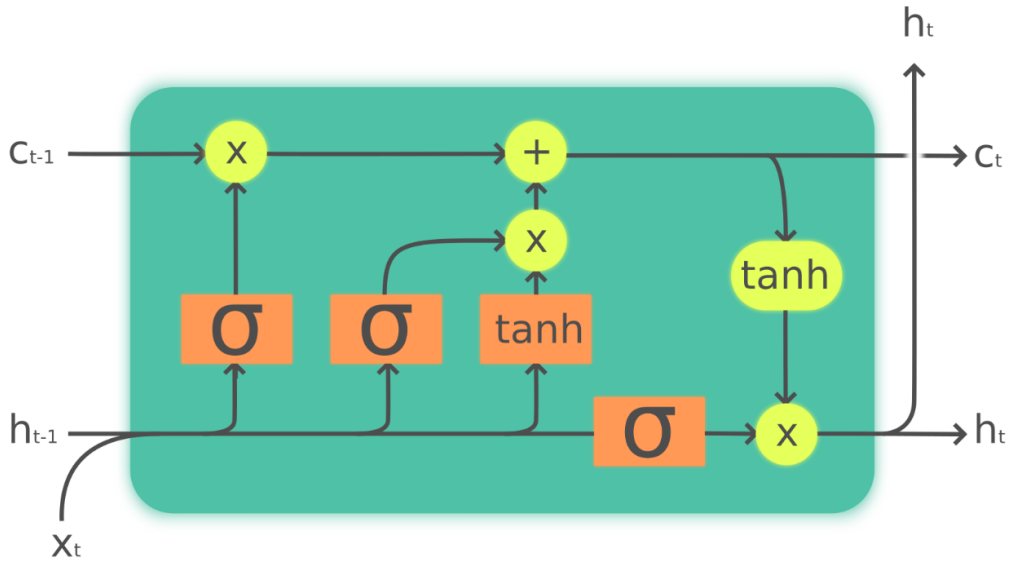


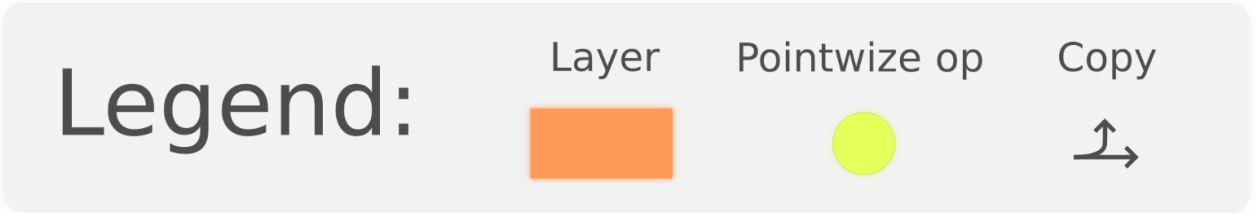
Gates optionally let information through the cell

LSTM Long Short-Term Memory

(Hochreiter & Schmidhuber, 1997)

Idea: regulate the flow of context by special gates (auxilary neural networks) and split memory (hidden state) in two separate terms: short-term h and long-term c

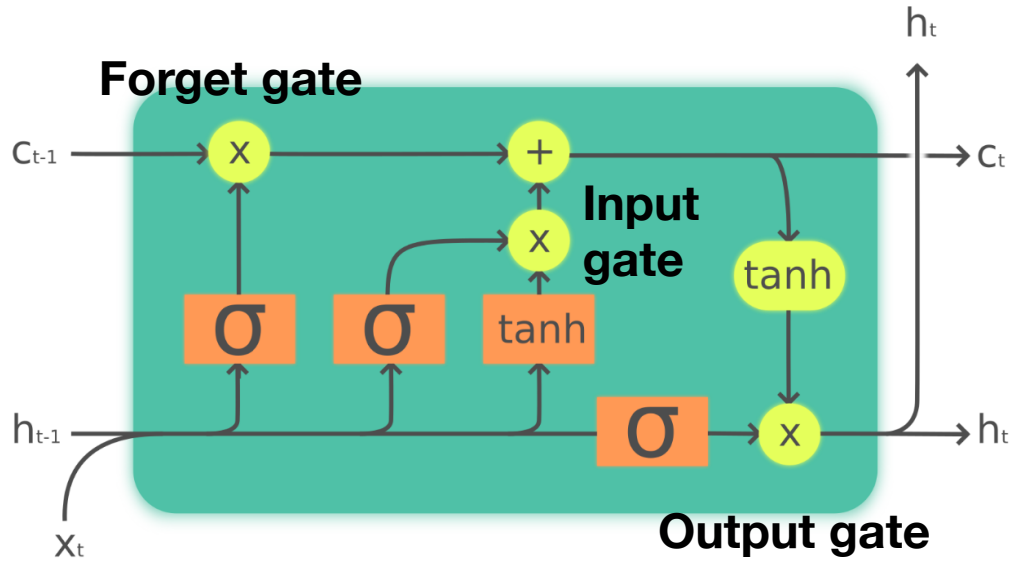


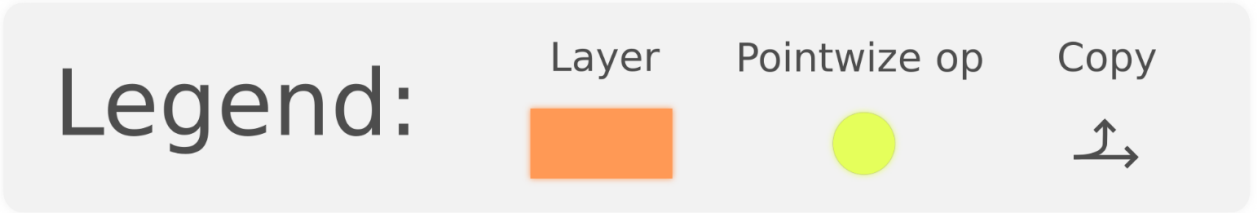


LSTM Long Short-Term Memory

(Hochreiter & Schmidhuber, 1997)

Idea: regulate the flow of context by special gates (auxilary neural networks) and split memory (hidden state) in two separate terms: short-term h and long-term c





LSTM vs RNN

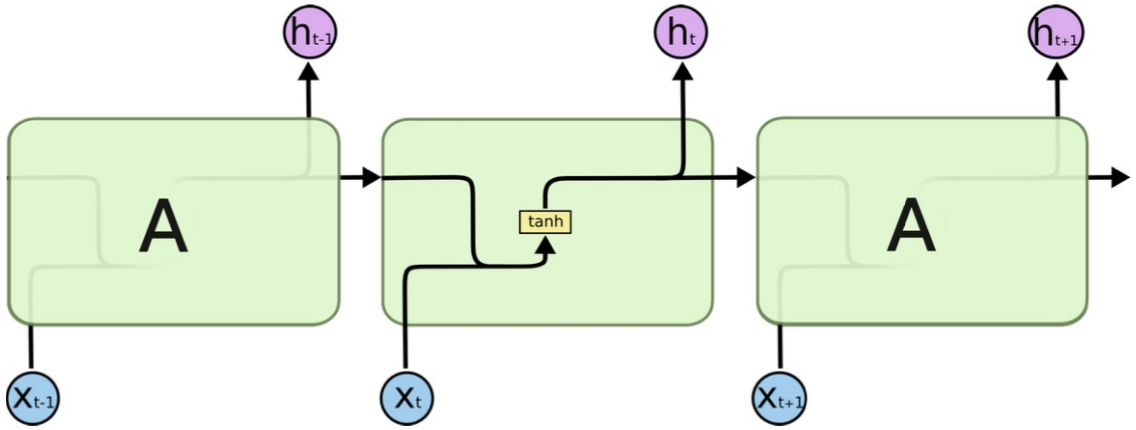


Figure 1: Basic RNN

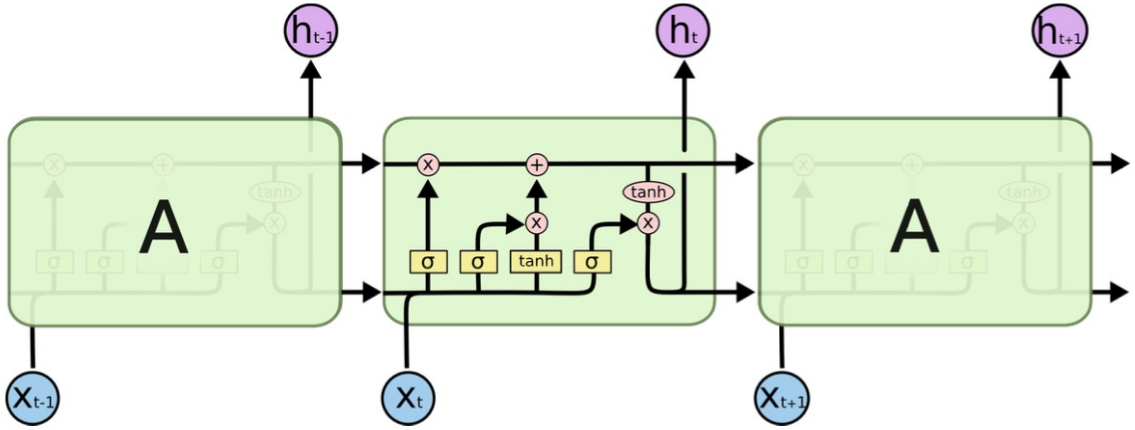
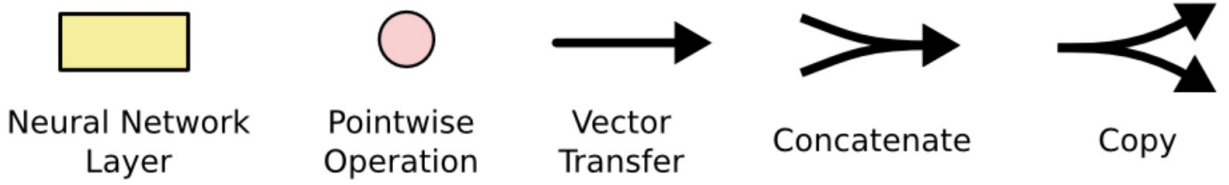
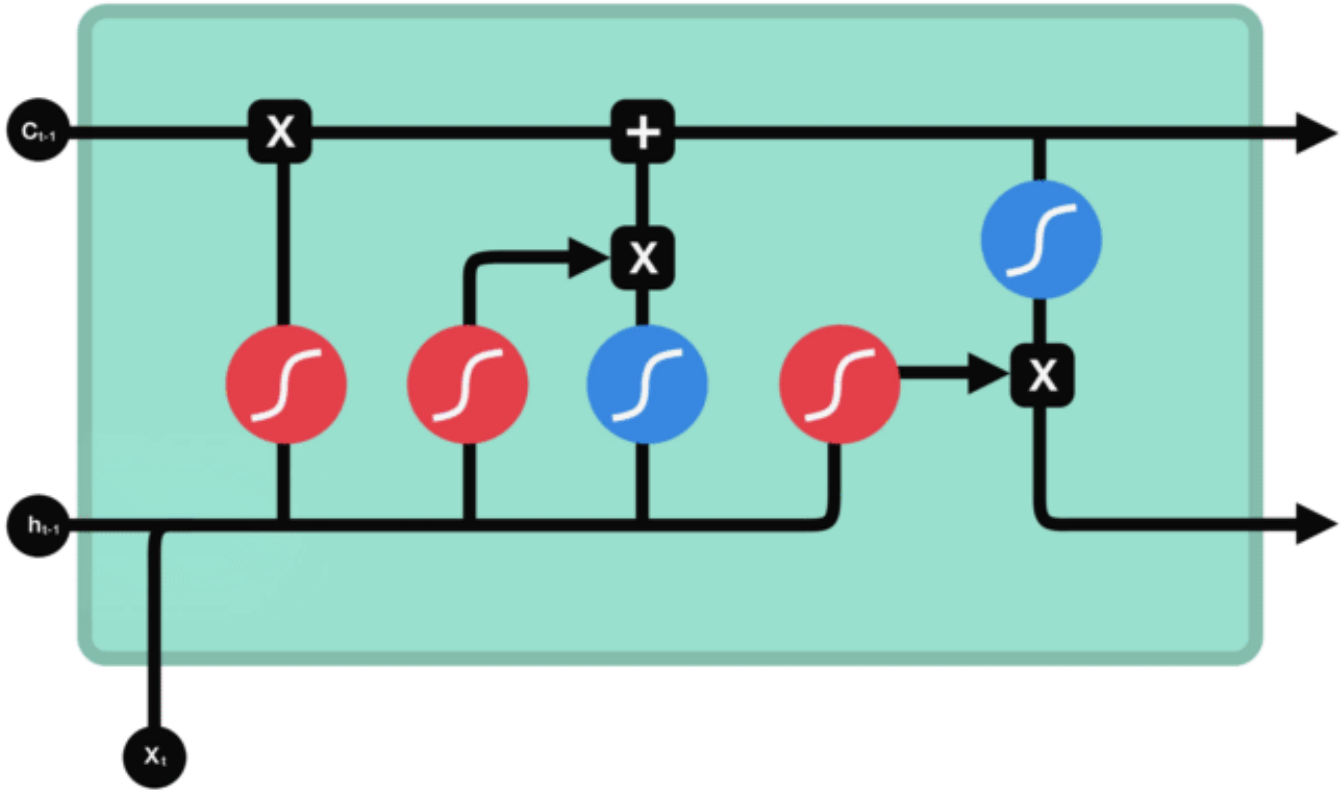


Figure 2: LSTM



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

Short Memory Gate



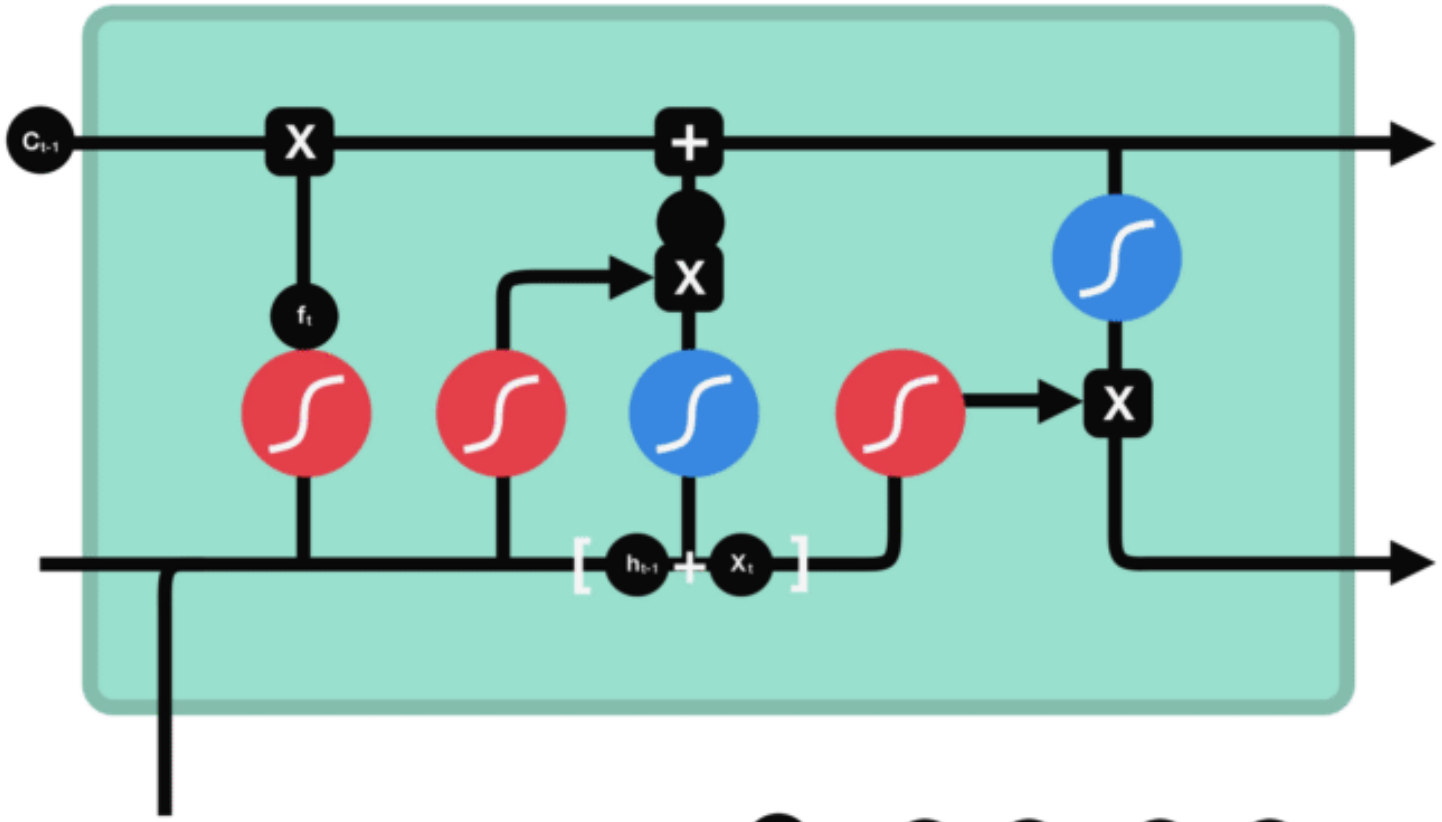
c_{t-1} previous cell state

f_t forget gate output

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

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

Long Memory Update: Cell State



C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

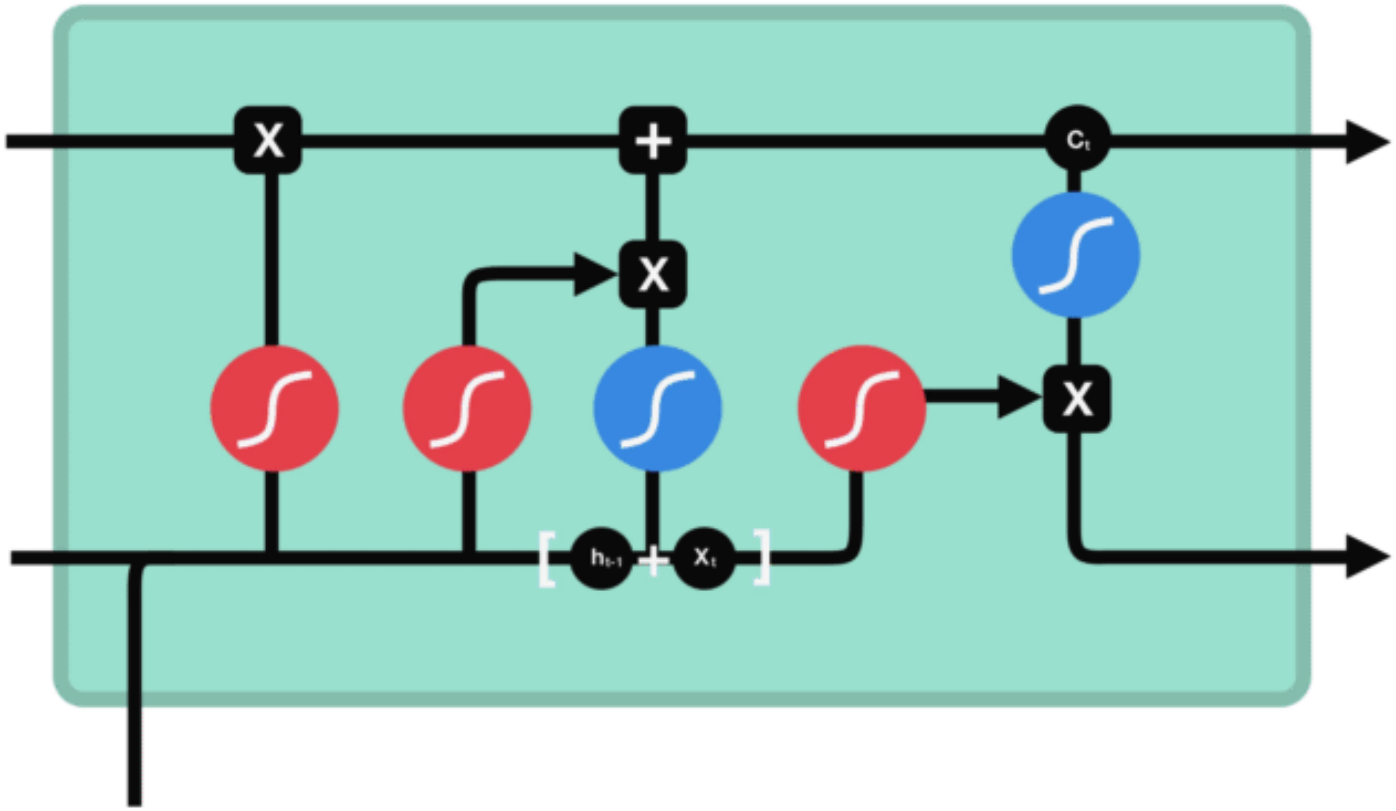
$\tilde{C}_t$ candidate

C_t new cell state

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

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

Output Gate



c_{t-1} previous cell state

f_t forget gate output

i_t input gate output

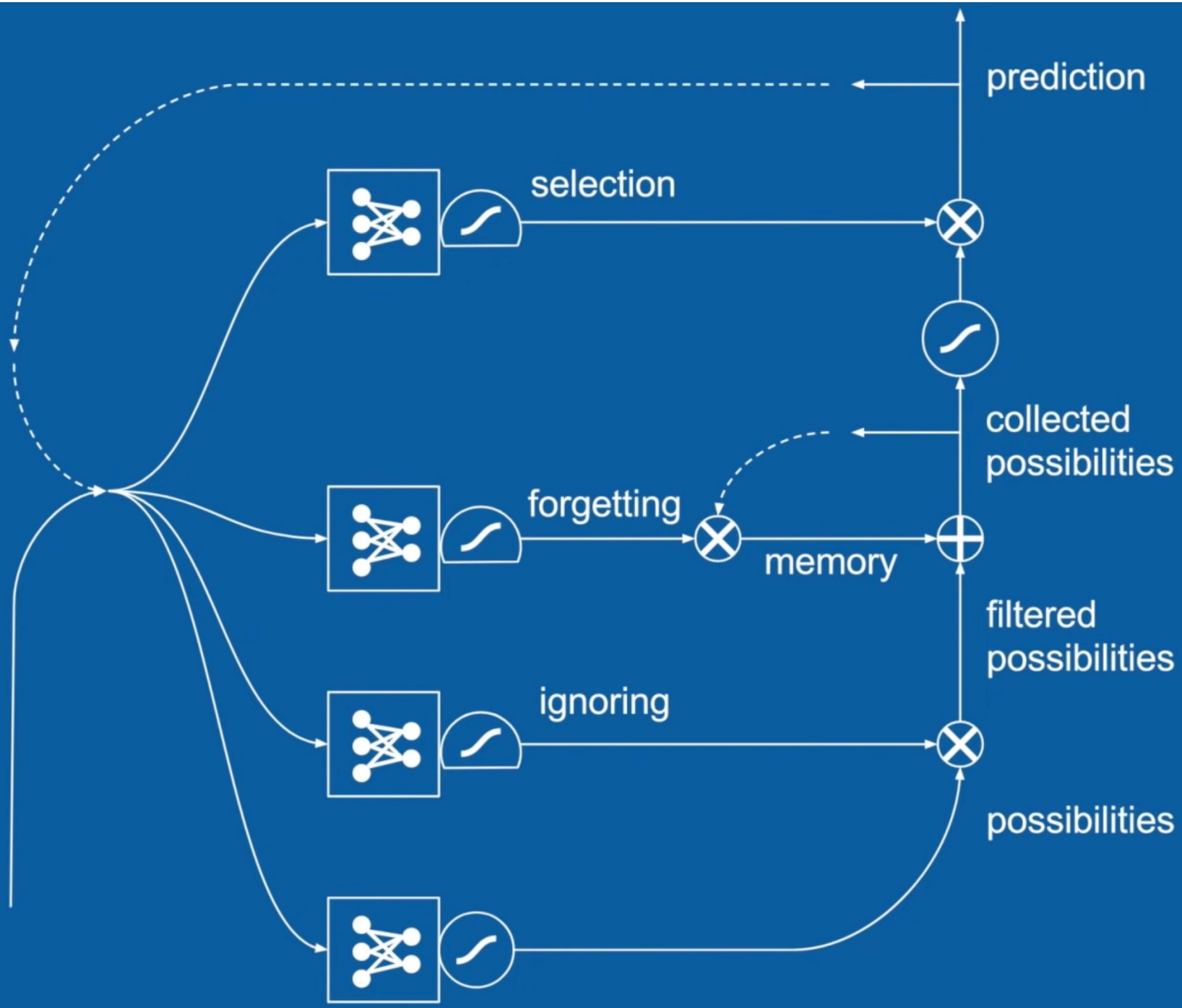
$\tilde{c}_t$ candidate

c_t new cell state

o_t output gate output

h_t hidden state

long short-term memory



Jane saw Spot.
Doug …

LSTM vs RNNs

Advantages:

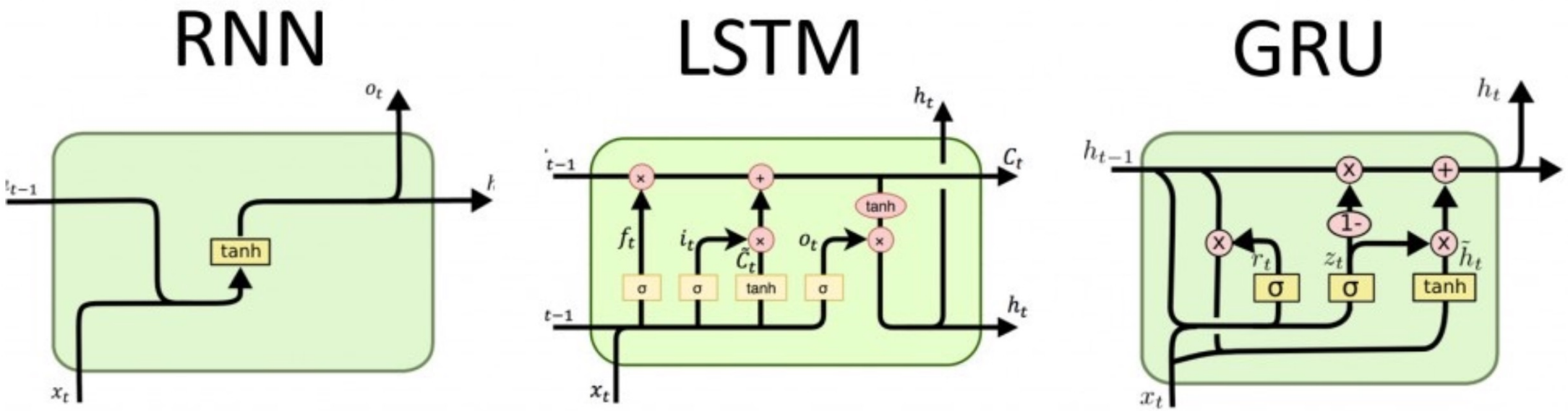
- Preserves sequence context
- Good performance and flexibility

Disadvantages:

- ~~Moves in sequence in the only direction~~ (Some sequences must be processed in both directions)
- ~~Short memory due to vanishing gradients problems~~
- ~~Exploding gradients due to cycles in recurrent connections~~
- **Slower** and heavier than RNN

Gated Recurrent Units

Idea: combine long-term and short-term updates in single channel

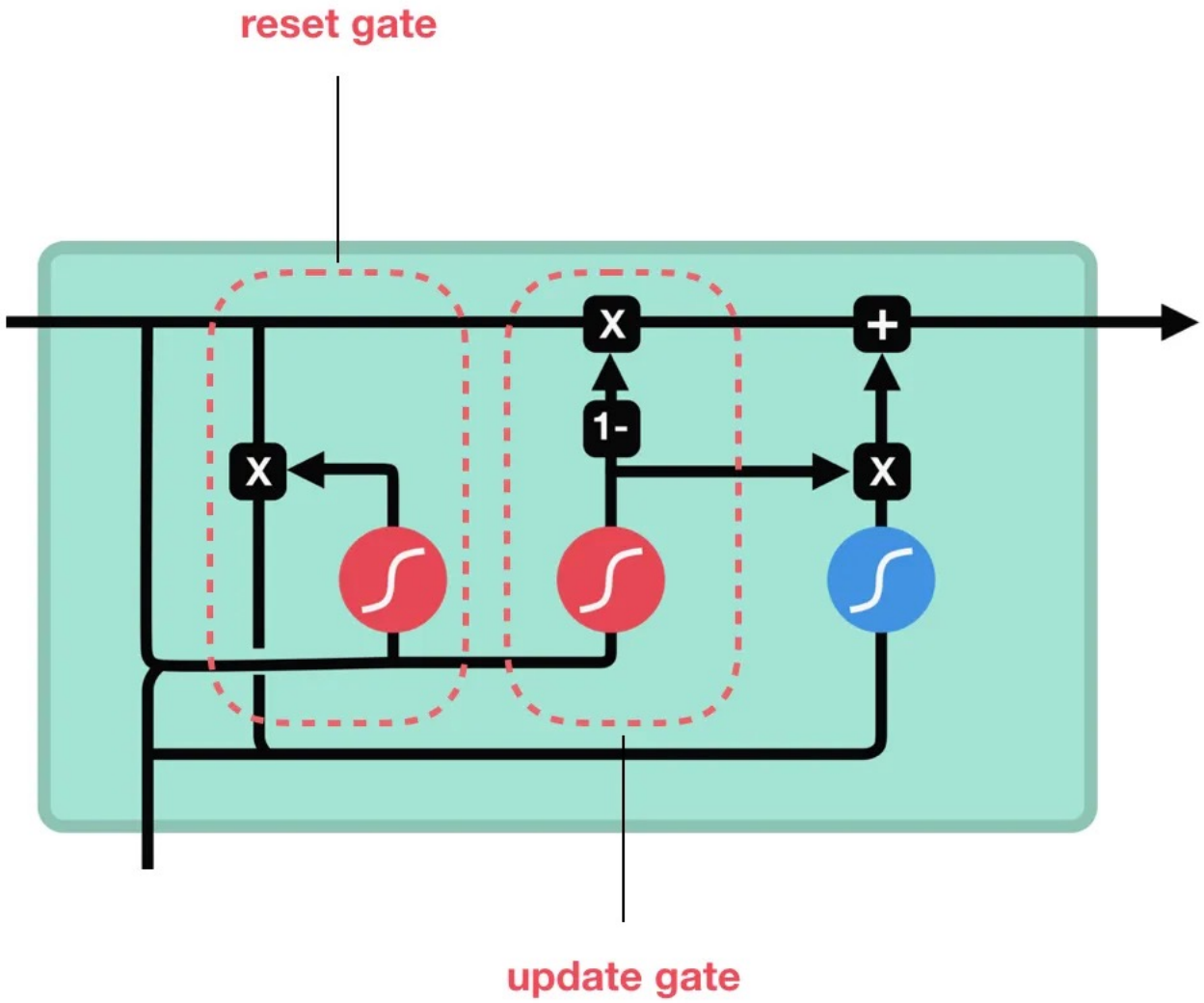


The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM.

GRU's got rid of the cell state and used the hidden state to transfer information.

GRU

The **reset gate** is another gate is used to decide how much past information to forget.



The **update gate** acts similar to the forget and input gate of an LSTM.

It decides what information to throw away and what new information to add.

GRU vs. LSTM. Advantages and disadvantages

Advantages:

- All context stored in single hidden state
- Lower model complexity
- Faster

Disadvantages:

- Shorter memory

Summary

Advantages:

- RNNs are popular and successful for variable-length sequences
- The gating models such as LSTM are suited for long-range error propagation

Problems:

- No transfer learning options
- The sequentiality prohibits parallelization within instances
- Long-range dependencies still tricky, despite gating (long-term dependencies are actually not so long due to gradient vanishing)

RNN and HEP applications

ATLAS Collaboration, "Identification of Jets Containing *b*-Hadrons with Recurrent Neural Networks at the ATLAS Experiment",.

ATLAS Collaboration, "Identification of hadronic tau lepton decays using neural networks in the ATLAS experiment",.

R. Teixeira de Lima, "Sequence-based Machine Learning Models in Jet Physics", arXiv:2102.06128.

K. Goto et al., "Development of a Vertex Finding Algorithm using Recurrent Neural Network", arXiv:2101.11906.

M. Wielgosz, A. Skoczeń, and M. Mertik, "Using lstm recurrent neural networks for monitoring the lhc superconducting magnets", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **867** (2017) 40–50, doi:https://doi.org/10.1016/j.nima.2017.06.020.

A. Schmitt, K. Fu, S. Fan, and Y. Luo, "Investigating deep neural networks for gravitational wave detection in advanced ligo data", in *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*, p. 73–78. Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3339363.3339377.

Hands-on

Text generation with RNN

https://www.tensorflow.org/text/tutorials/text_generation

Hands-on

Machine translation with RNN

https://github.com/fsimone91/course_ml4hep/blob/2024/notebooks/RNNs/RNNs.ipynb