Natural Language Processing
NLP_CLT_1c_May_4th_2025

Eng. Maytham Ghanoum

Artificial Intelligence & Deep Learning Specialist
MTN Syria – SCS – SVU CLT
+963947222064 - +963982018359
https://www.linkedin.com/in/maytham-ghanoum-697aa5207/
https://www.facebook.com/maytham.ghanoum

RNN

Recurrent Neural Networks:

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

RNN

Recurrent Neural Networks:

Structure of RNNs

Recurrent Neurons: The key feature of RNNs is the recurrent neuron, which maintains a hidden state that is passed from one time step to the next. This hidden state captures information about previous time steps, enabling the network to have memory.

Hidden Layers: Typically, RNNs have one or more hidden layers where the recurrent connections are applied.

Output Layer: This layer produces the final output of the network, which can vary depending on the specific task (e.g., classification, regression).

RNN

Recurrent Neural Networks:

Components of RNNs

Input Layer: Takes in the sequential data.
Recurrent Layer(s): Processes the data step-by-step, maintaining a hidden state.
Hidden State: Captures the information from previous steps to inform future steps.
Output Layer: Produces the final predictions.

Approach of RNNs

Sequence Processing: RNNs process data sequentially. For each time step in the sequence, the network takes the current input and the hidden state from the previous time step to produce the output and the new hidden state.

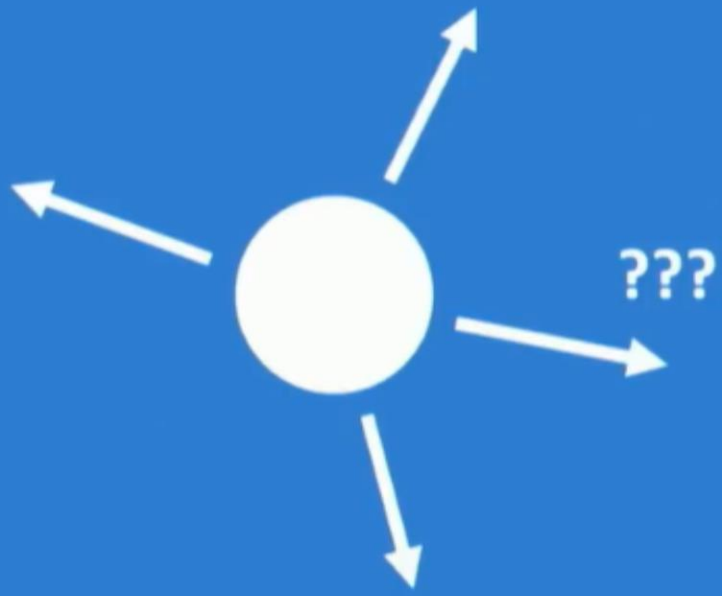
Backpropagation Through Time (BPTT): RNNs are trained using a variation of backpropagation called BPTT, which accounts for the sequential nature of the data by unrolling the network over time and computing gradients accordingly.

RNN

Given an image of a ball,
can you predict where it will go next?

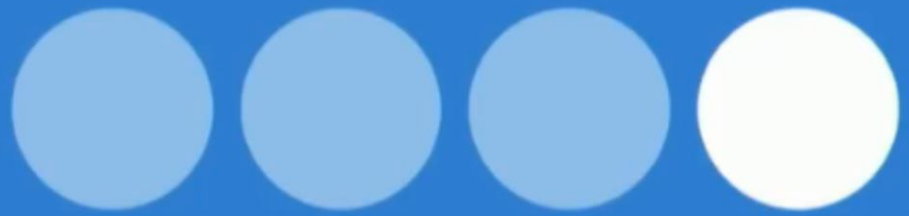
RNN

Given an image of a ball,
can you predict where it will go next?



RNN

Given an image of a ball,
can you predict where it will go next?



RNN

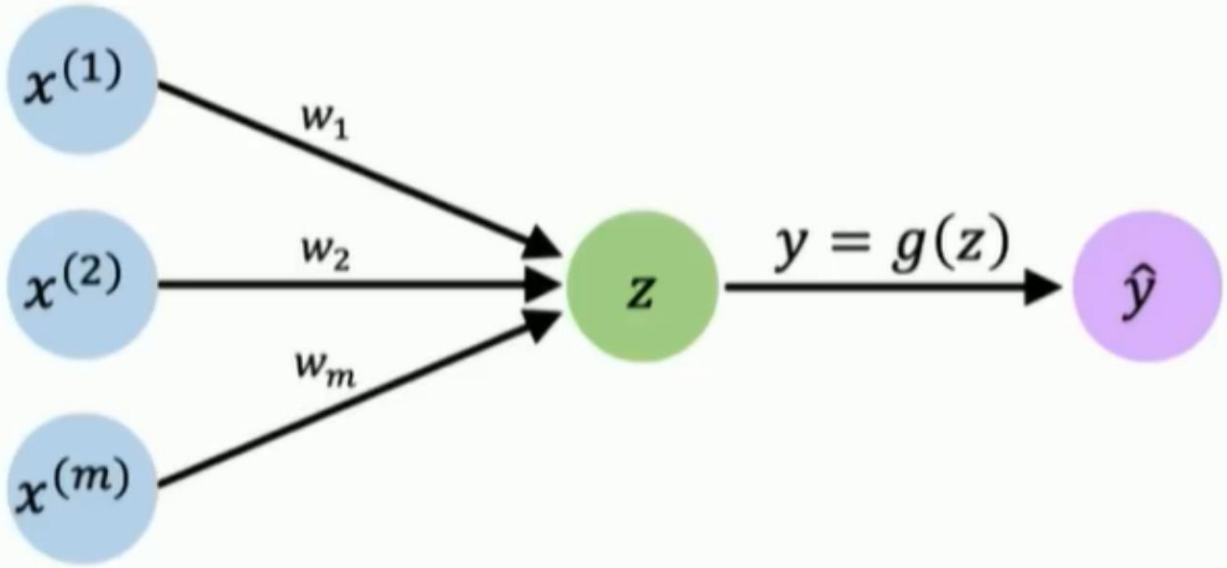
Given an image of a ball,
can you predict where it will go next?

RNN

Neurons with Recurrence

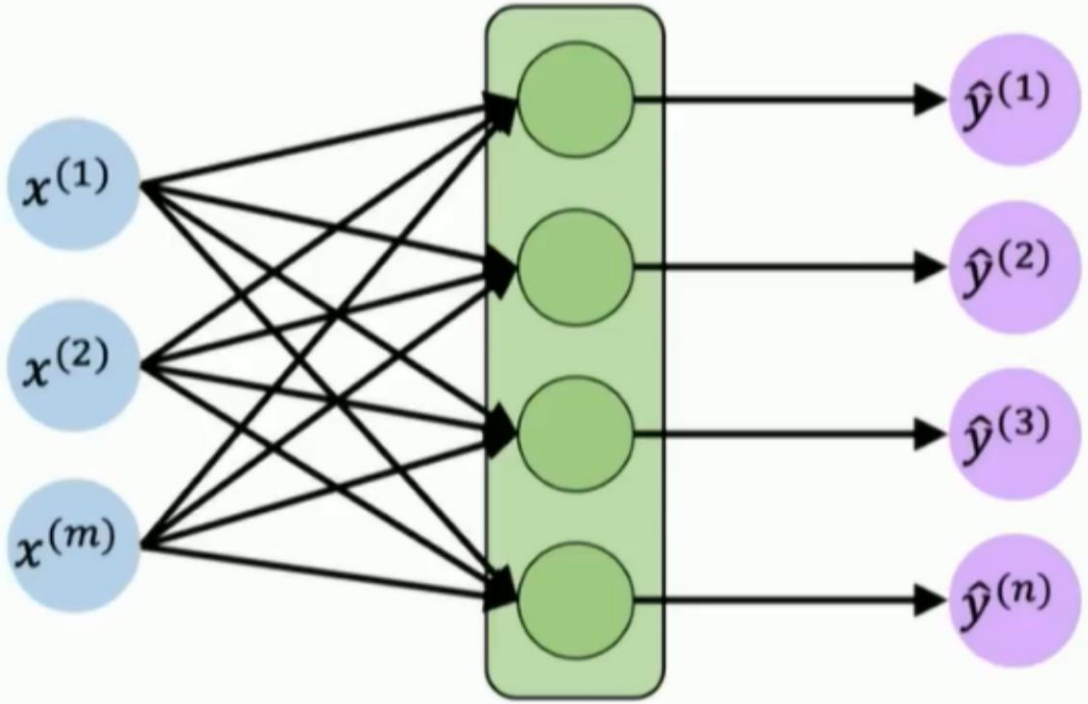
RNN

The Perceptron Revisited



RNN

Feed-Forward Networks Revisited

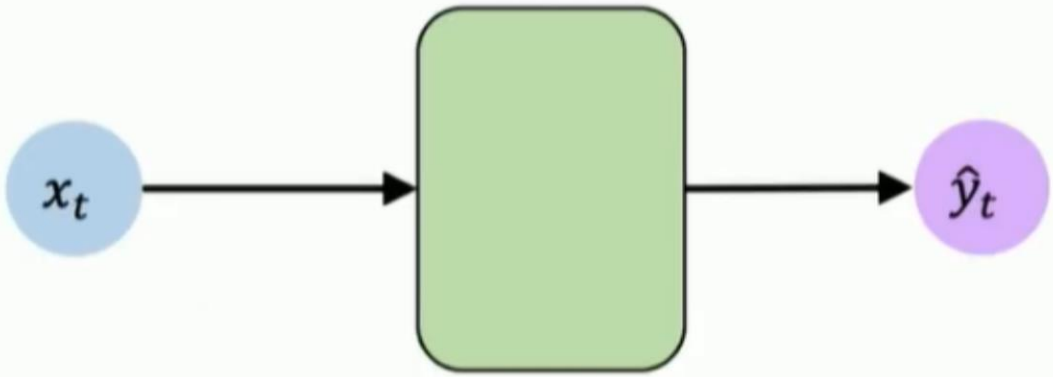


$\boldsymbol{x} \in \mathbb{R}^m$

$\hat{\boldsymbol{y}} \in \mathbb{R}^n$

RNN

Feed-Forward Networks Revisited

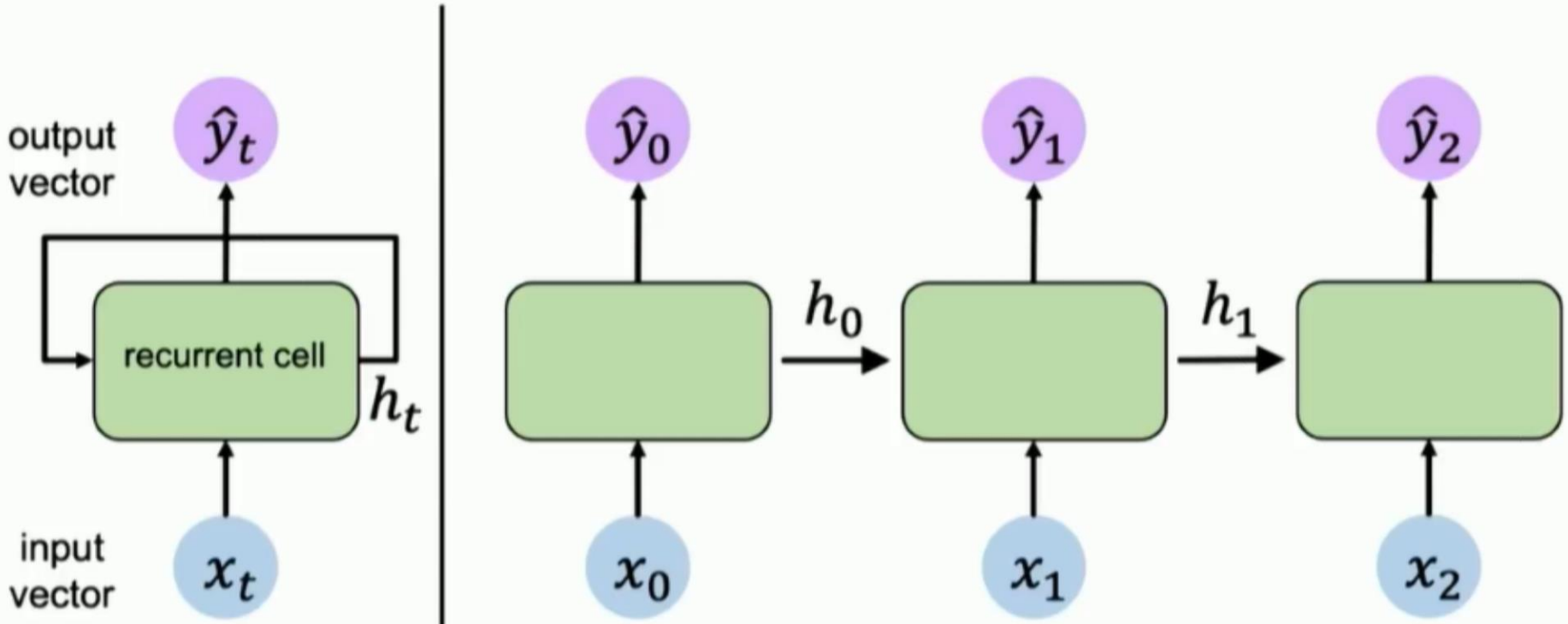


$x_t \in \mathbb{R}^m$

$\hat{y}_t \in \mathbb{R}^n$

RNN

Neurons with Recurrence



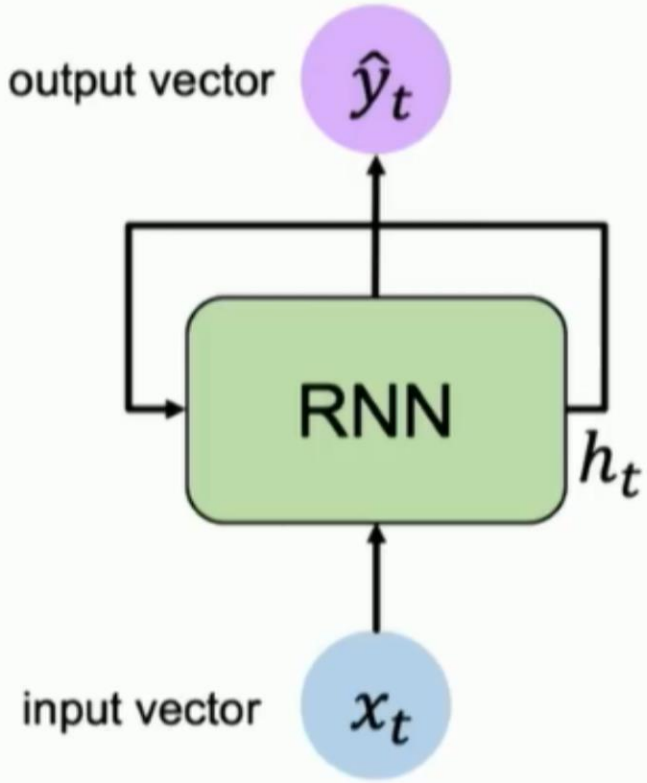
$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

RNN

Recurrent Neural Networks (RNNs)

RNN

Recurrent Neural Networks (RNNs)



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

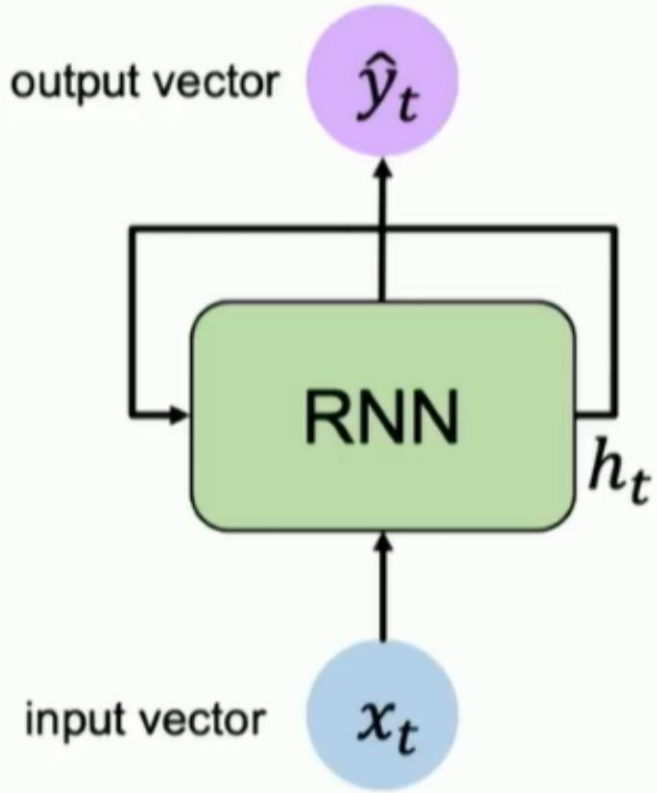
old state

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

RNN

RNN State Update and Output



Output Vector

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

Update Hidden State

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

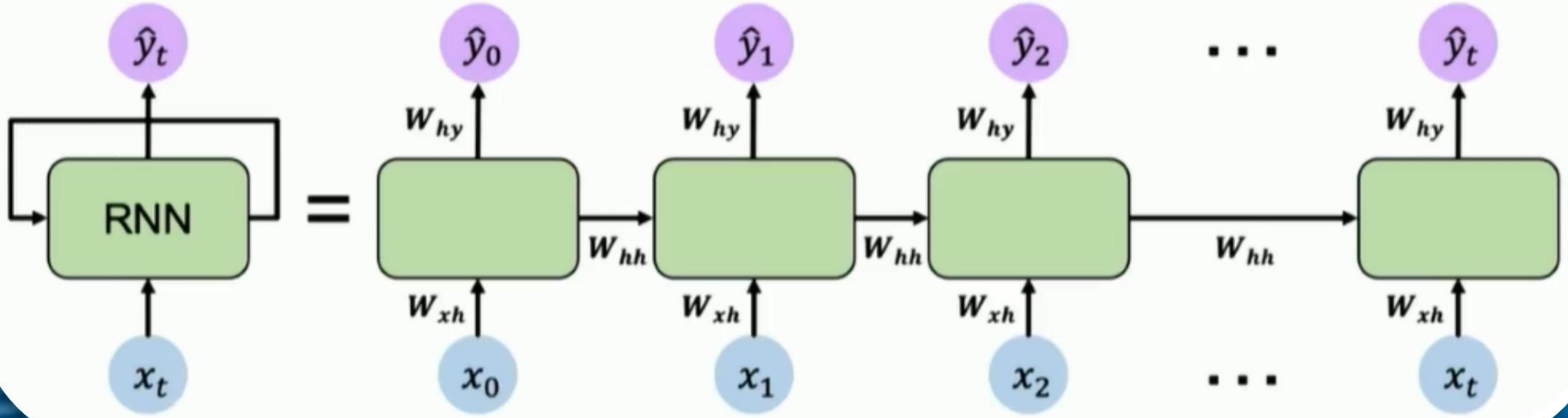
Input Vector

$$x_t$$

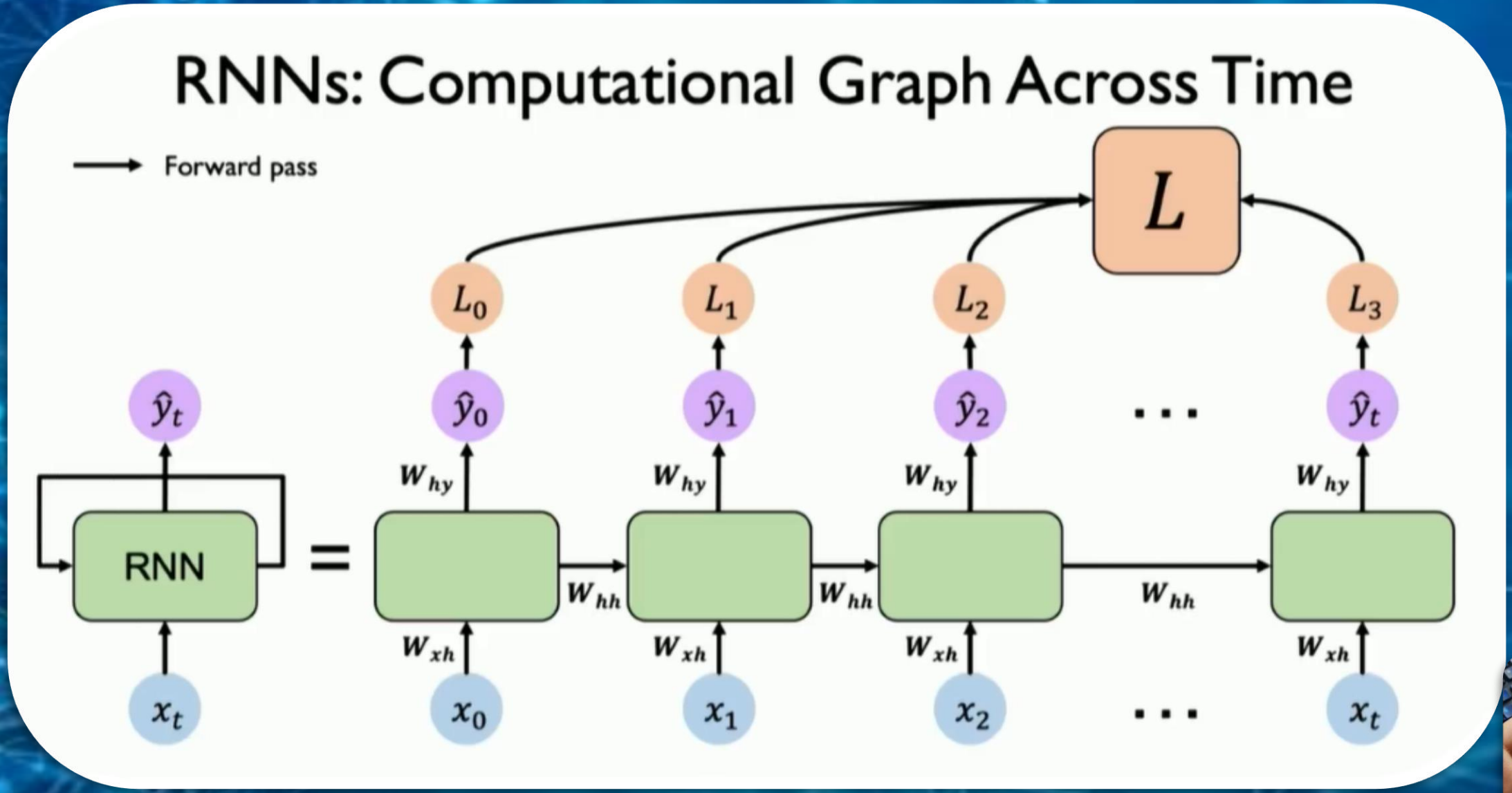
RNN

RNNs: Computational Graph Across Time

Re-use the **same weight matrices** at every time step



RNN

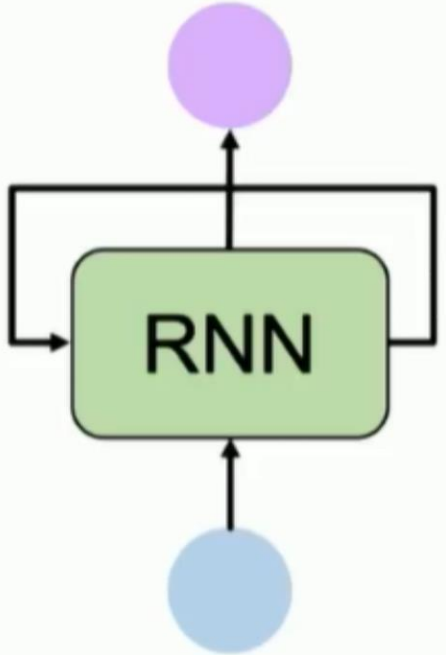

RNNs: Computational Graph Across Time
Forward pass
L
L_0
L_1
L_2
L_3
$\hat{y}_t$
$\hat{y}_0$
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_t$
RNN
=
W_{hy}
W_{hh}
W_{xh}
x_t
x_0
x_1
x_2
x_t

RNN

Sequence Modeling: Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence



Recurrent Neural Networks (RNNs) meet these sequence modeling design criteria

RNN

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

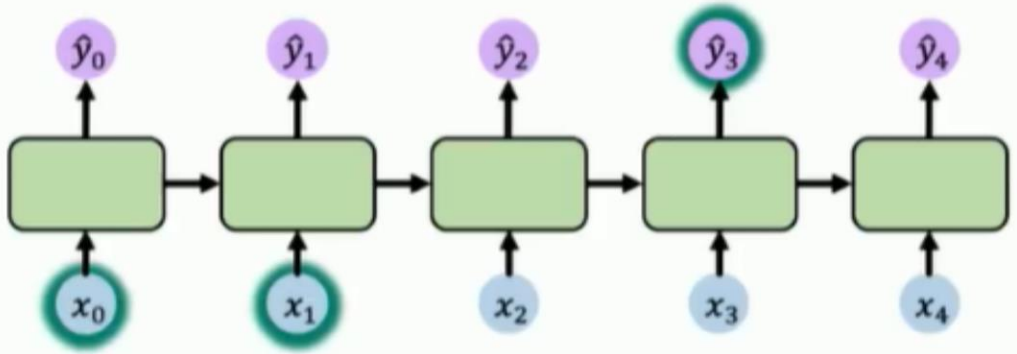
↓

Errors due to further back time steps have smaller and smaller gradients

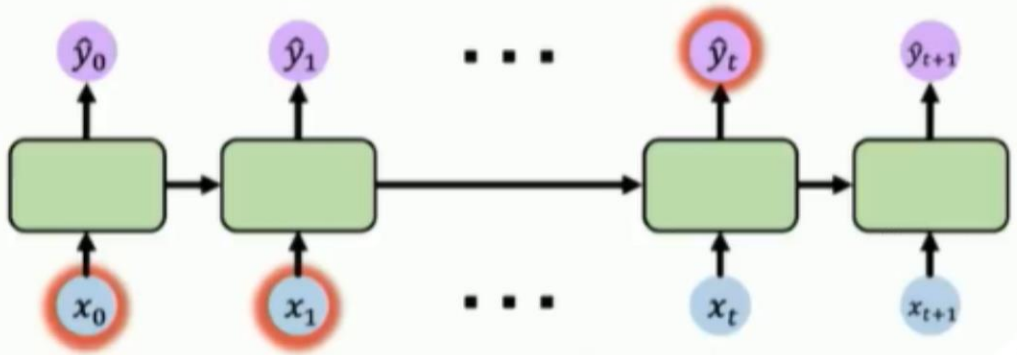
↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



"I grew up in France, … and I speak fluent___ "



RNN

Long Short-Term Memory:

A type of recurrent neural network (RNN) architecture. It has become one of the most popular and effective models for various sequence learning tasks, such as natural language processing, time series prediction, and speech recognition.

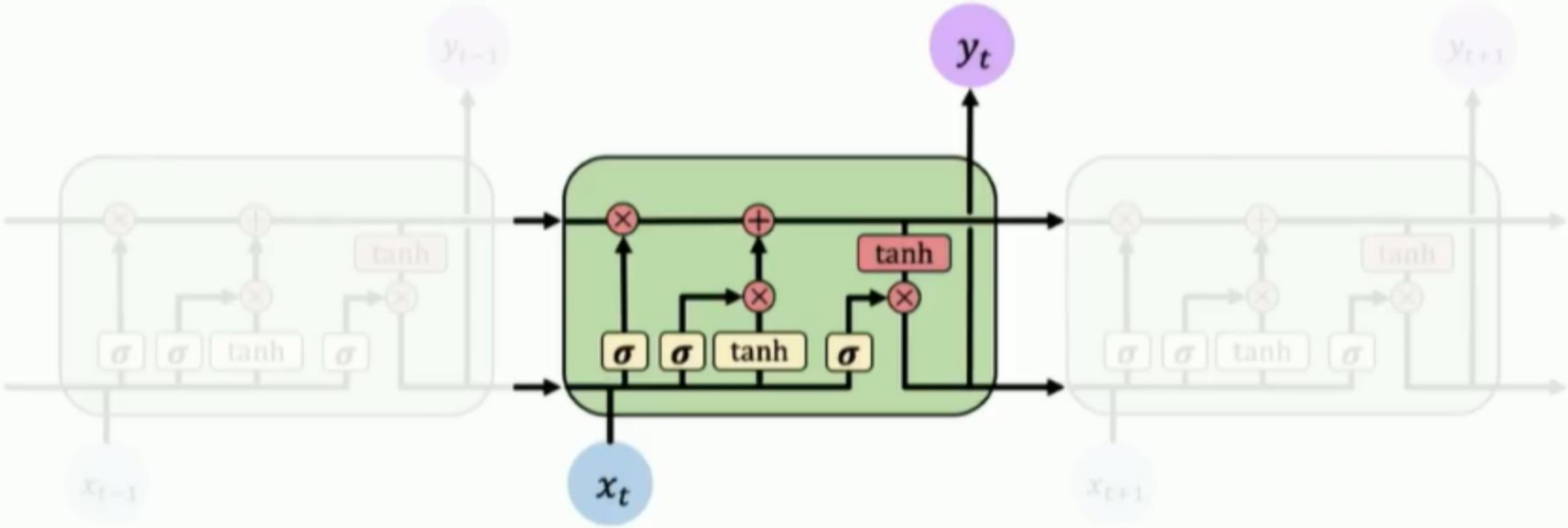
LSTMs are designed to remember information for long periods, making them well-suited for tasks where understanding the context over long sequences is crucial. The key innovation of LSTMs is their ability to mitigate the vanishing gradient problem, which plagued traditional RNNs.

RNN

Long Short Term Memory (LSTMs)

Gated LSTM cells **control information flow:**

1) Forget 2) Store 3) Update 4) Output



LSTM cells are able to track information throughout many timesteps

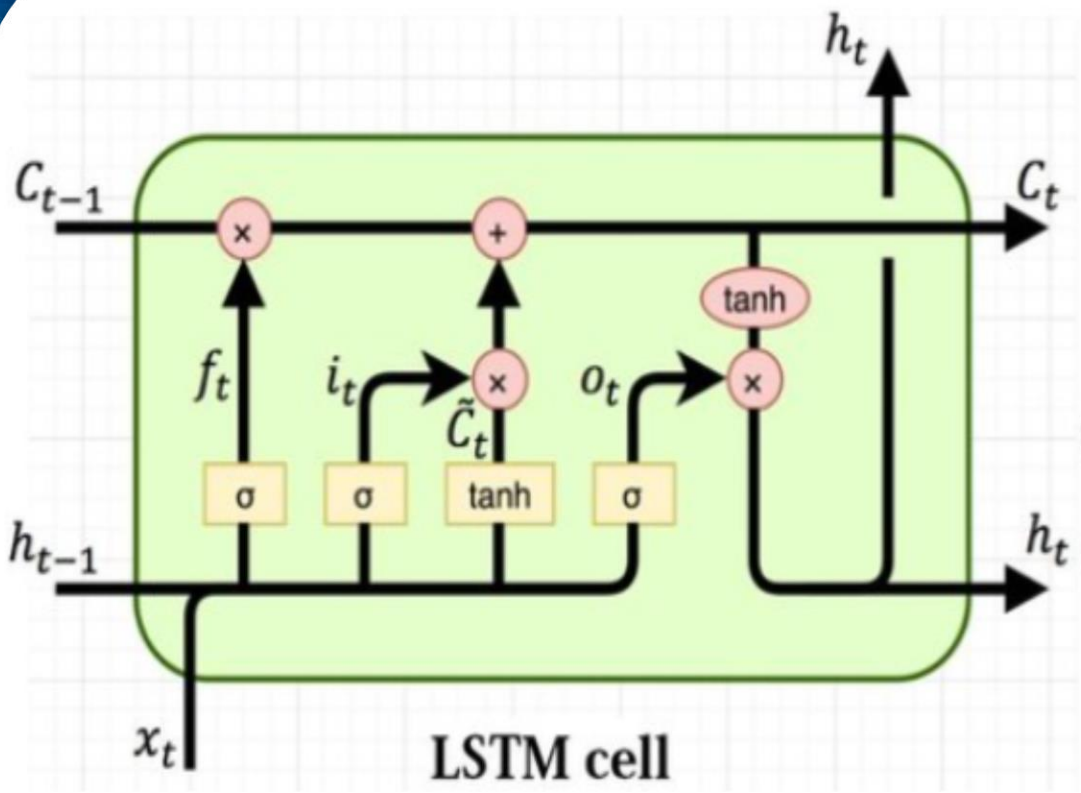
```
tf.keras.layers.LSTM(num_units)
```

RNN

LSTMs: Key Concepts

1. Maintain a **cell state**
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input
   - Selectively **update** cell state
   - **Output** gate returns a filtered version of the cell state
3. Backpropagation through time with partially **uninterrupted gradient flow**

RNN



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$

$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$

$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$

$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$

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

Structure of the LSTM cell and equations that describe the gates of an LSTM cell.

RNN

How LSTM Addresses These Limitations

Memory Cell:

Description: LSTM introduces a memory cell, a component that maintains its state over time and controls when to read, write, and forget information.
Impact: This allows LSTMs to retain information over long sequences, effectively mitigating the vanishing gradient problem.

Gates Mechanism:

Input Gate: Controls how much of the new information to let into the memory cell.
Forget Gate: Determines what information should be discarded from the memory cell.
Output Gate: Controls how much of the memory cell's information to use in the output.
Impact: These gates enable selective updates to the cell state, ensuring relevant information is retained over long sequences while irrelevant information is discard

RNN

How LSTM Addresses These Limitations

Training Stability:

Description: By mitigating the vanishing and exploding gradient problems, LSTMs make the training process more stable.

Impact: This stability allows LSTMs to be effectively trained on long sequences, capturing long-term dependencies.

Enhanced Learning Capability:

Description:

The architectural enhancements of LSTMs (memory cell and gating mechanisms) allow them to learn complex patterns and relationships in sequential data.

Impact: This makes LSTMs highly effective for a wide range of sequence learning tasks, including language modeling, translation, and time series forecasting.