IMPERIAL

Neural Language Models

03/12/2025

Shamsuddeen Muhammad
Google DeepMind Academic Fellow,
Imperial College London
https://shmuhammadd.github.io/

Idris Abdulmumin
Postdoctoral Research Fellow,
DSFSI, University of Pretoria
https://abumafrim.github.io/

Today

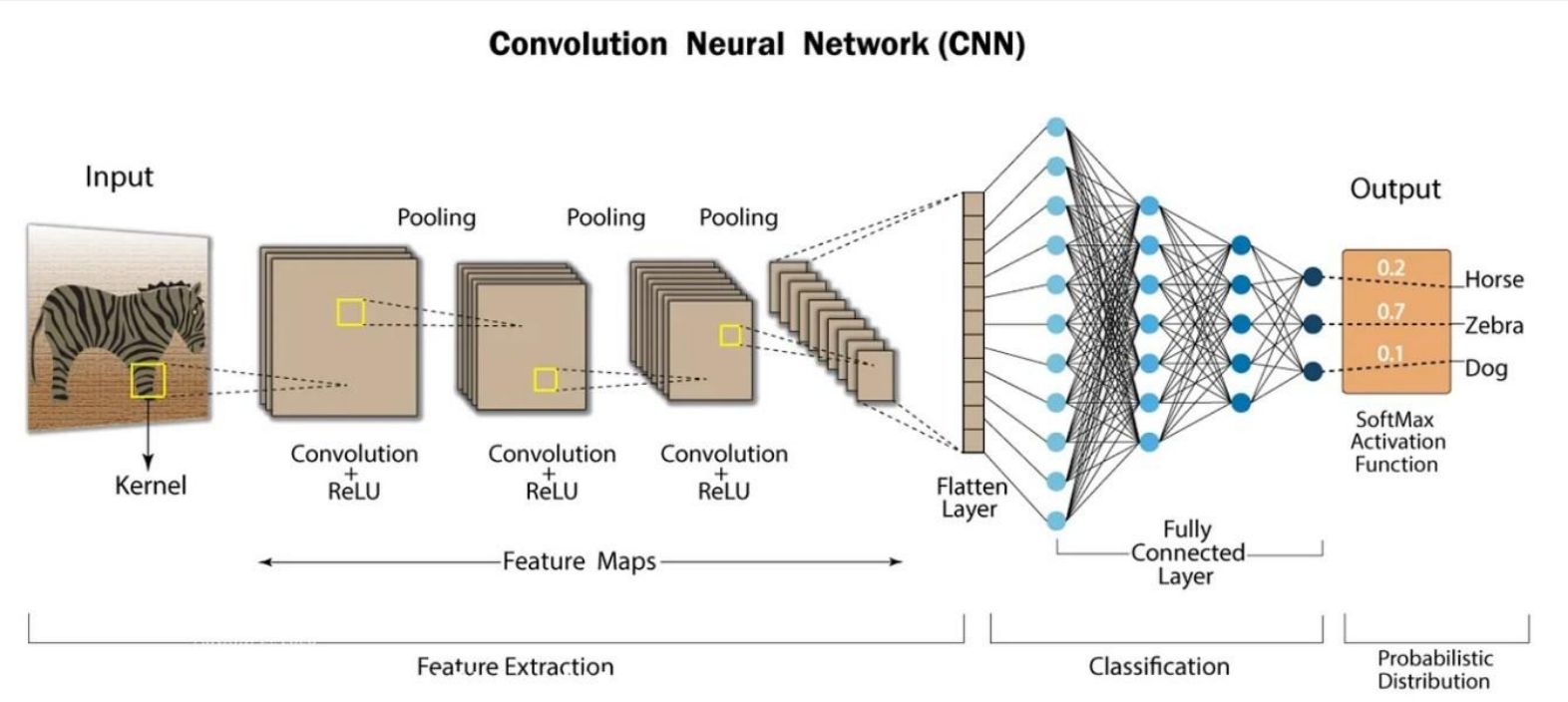
Chapter 6 Neural Networks:

https://web.stanford.edu/~jurafsky/slp3/6.pdf

Prerequisite

- Prerequisite
  - Neural Network
  - Loss function, backpropagation
  - CNN

CNN Revisited



- A **CNN** is a type of neural network that's designed to work with ***image data***.
- Robust Performance: Achieves state-of-the-art results in computer vision tasks.

CNN Revisited

- They are made up of a series of layers:
  - The **convolutional** layers: responsible for extracting features from the images.
  - The **pooling** layers: responsible for reducing the size of the image after the convolutional layers have extracted the features.
  - **Activation** layers: Introduce non-linearity to the model (ReLU (Rectified Linear Unit), Sigmoid)

CNN Revisited

In NLP, CNNs apply **1-dimensional convolutions** over sequences of word embeddings to automatically learn **local linguistic patterns** (e.g., n-grams such as *“not good”*, *“very happy”*)

Kim, Y. (2014). *Convolutional Neural Networks for Sentence Classification.*

CNN Revisited

- Lack of Sequential Order Awareness
  - CNNs treat input as spatially ordered (like pixels in an image) but not temporally ordered.
- In sequential tasks (e.g., text or time-series data), the order of input is critical.

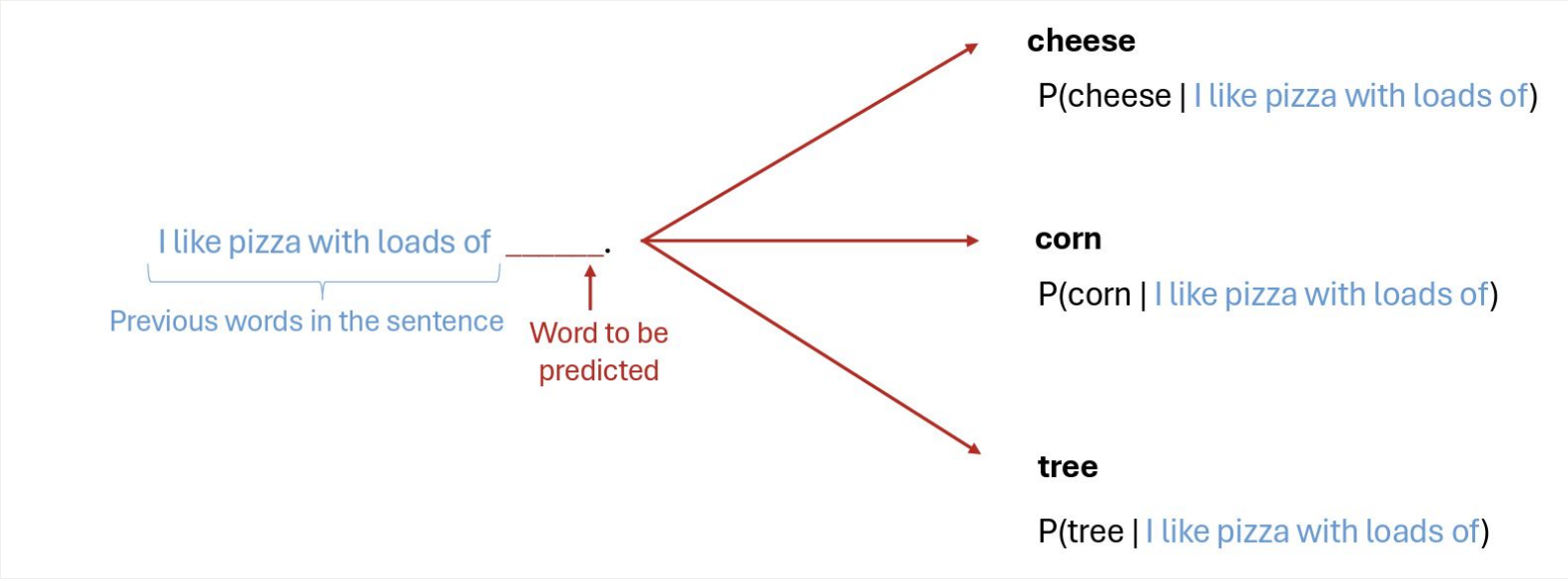
CNN Revisited

CNNs require the input data to have a **fixed size** (e.g., images of a specific resolution).

This makes them unsuitable for sequences of varying lengths.

Recall

A **language model (LM)** is a system that **predicts the next word** (or sequence of words) in a sentence.



Recall: Language Modeling (LM)

Goal: compute the probability of a sentence or sequence of words W:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

Related task: probability of an upcoming word:

$$P(w_5|w_1,w_2,w_3,w_4) \text{ or } P(w_n|w_1,w_2 \ldots w_{n-1})$$

A language model computes either of these:

$$P(W) \quad \text{or} \quad P(w_n|w_1,w_2 \ldots w_{n-1})$$

Categories for Language Models

- Statistical Language Models
- Neural Language Models (NNLMs)
- Transformer-Based Language Models
- Large Language Models (LLMs)
- Multilingual & Low-Resource LMs

Limitation of N-Gram Language Model

- N-gram models struggle to capture **long-range dependencies** in natural language.
- They only look at the previous *n* words, so important relationships across the sentence are lost.
- They lack global coherence and fail on complex syntax.

Traditional Models

The following classical approach treats text as unordered.

- Rule-Based & Pattern Matching Systems
- Classical Machine Learning : Logistic Regression, Naive Bayes, SVM (Support Vector Machines), Decision Trees / Random Forests
- TF-IDF + Linear Classifiers (Logistic Regression, SVM)
- Topic Models such as LDA

Limitations of Classical Models

- Ignore word order (bag-of-words assumption)
- Cannot capture long-range dependencies
- Fail on tasks needing context
- Limited semantic understanding

Simple Example for Students

Sentence A: *"Football makes people happy."*
Sentence B: *"People make football happy."*

Both contain the same words - BoW models treat them as identical, even though the meanings differ.

Sequence Modelling

Sequence modelling refers to building models that can ***represent, process, and predict ordered data***, *where* ***the order of elements matters***.

- Natural language is inherently sequential: meaning depends on previous and next words.

- Many NLP tasks cannot be solved by treating text as unordered bags of words.
  - ○

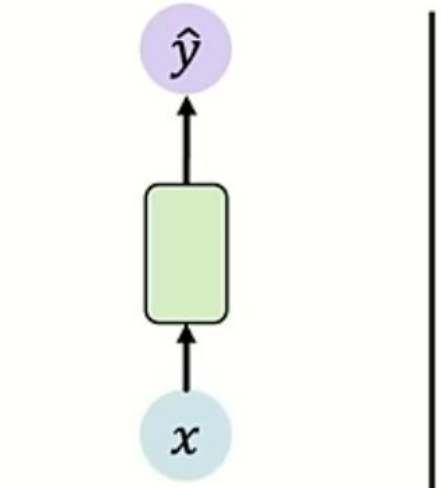
Classical sequence architectures

Classical sequence architectures:

- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- Gated Recurrent Units (GRUs)
- Encoder–Decoder architectures (precursor to Transformers)

Sequence Modelling Application

Sequence Modelling Application

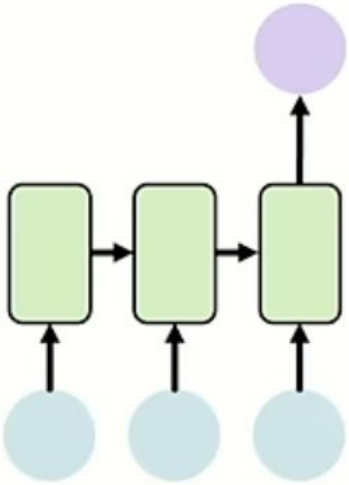


One to One

Binary Classification

"Will I pass this class?"

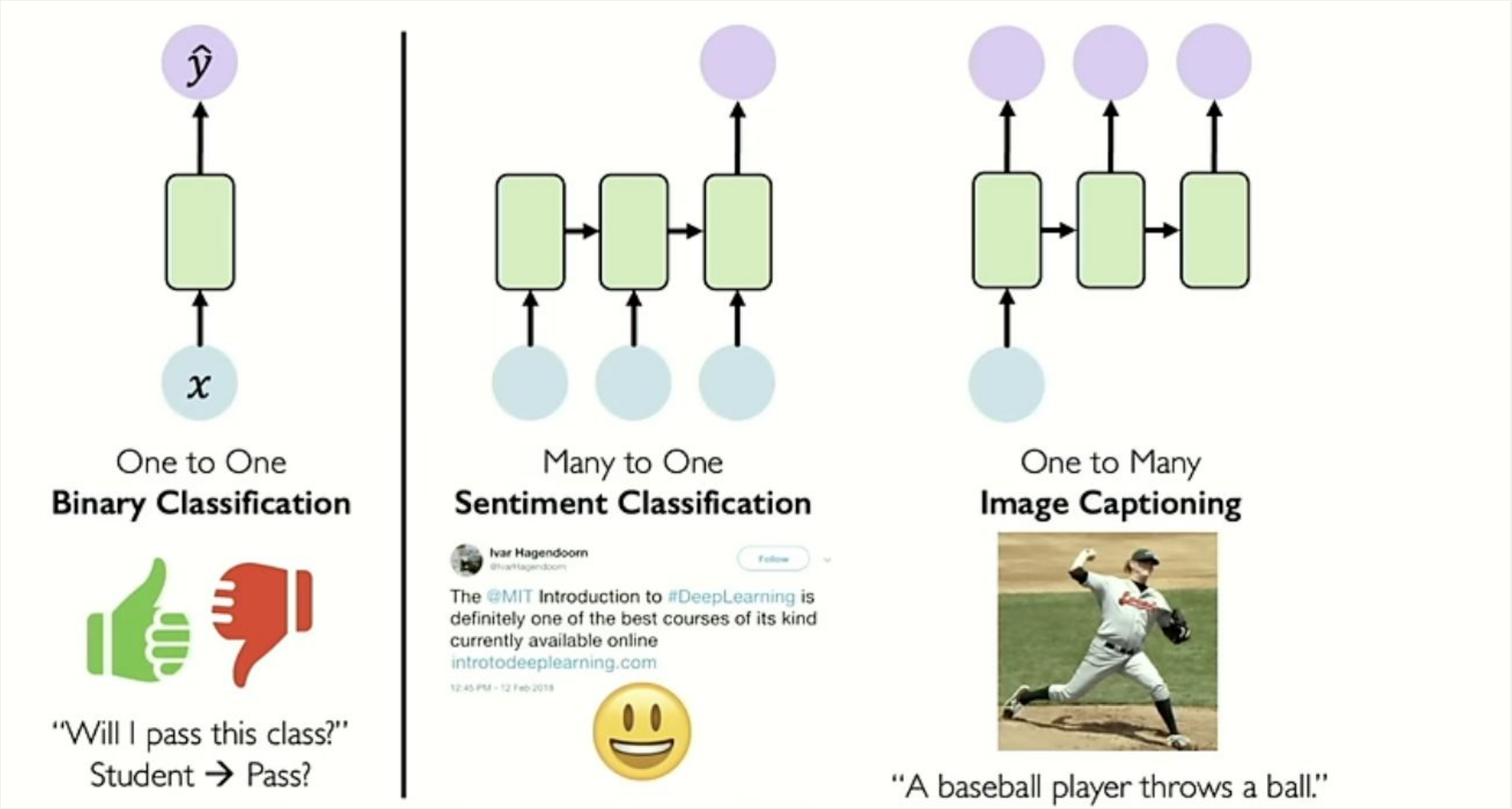
Student → Pass?



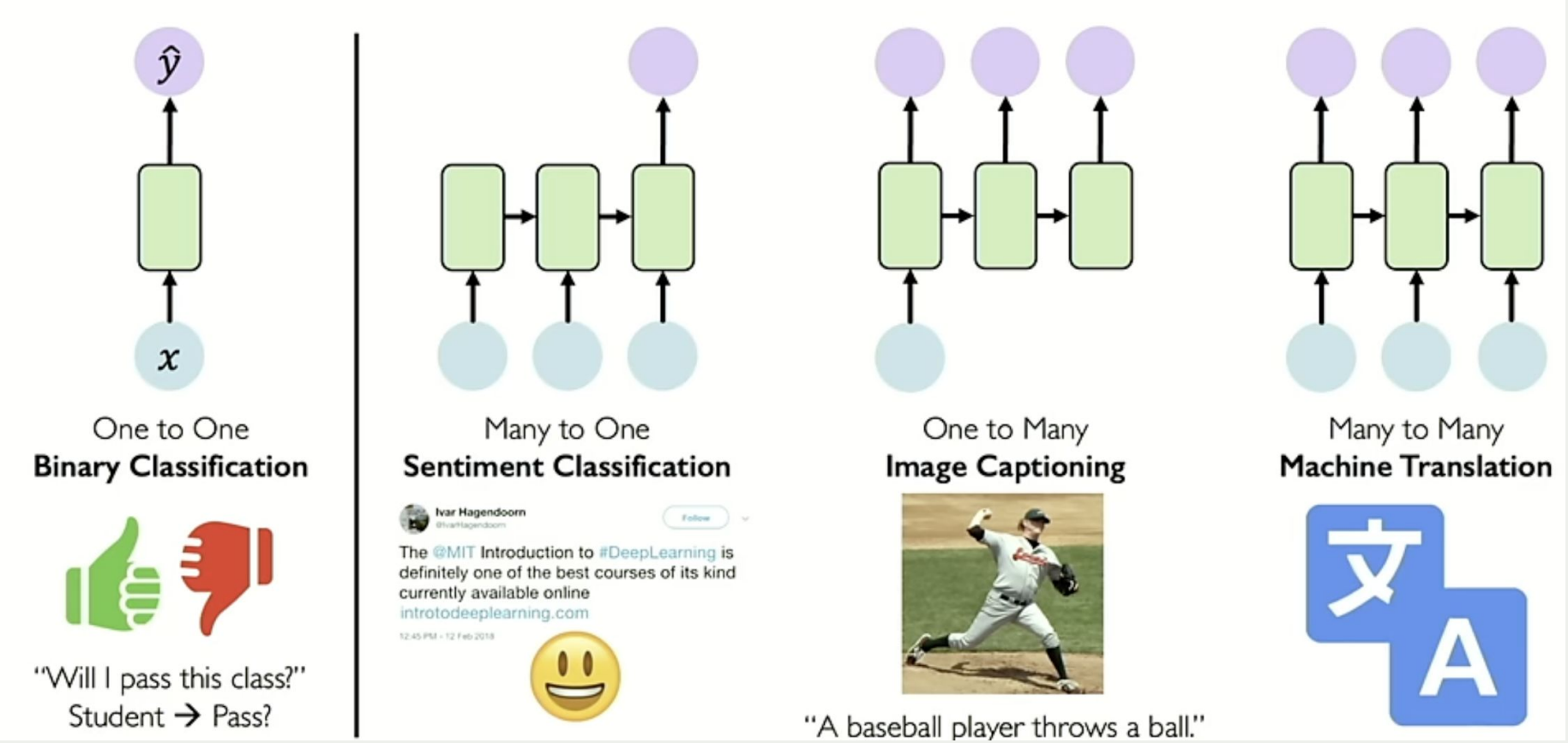
Many to One

Sentiment Classification

Sequence Modelling Application



Sequence Modelling Application

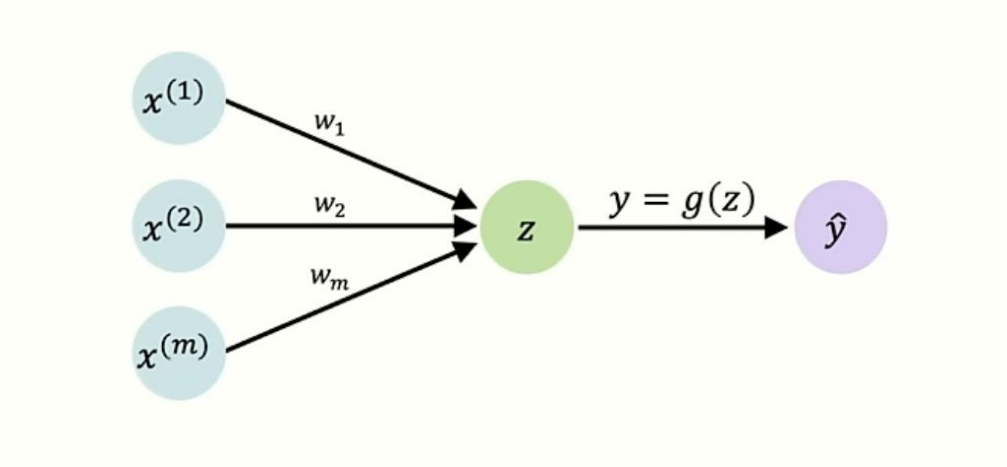


Neurons with Recurrence

Perceptron Revisited

- The **Perceptron** is one of the earliest and simplest learning algorithms for binary classification.

- It learns a linear decision boundary by repeatedly adjusting its weight vector based on mistakes.

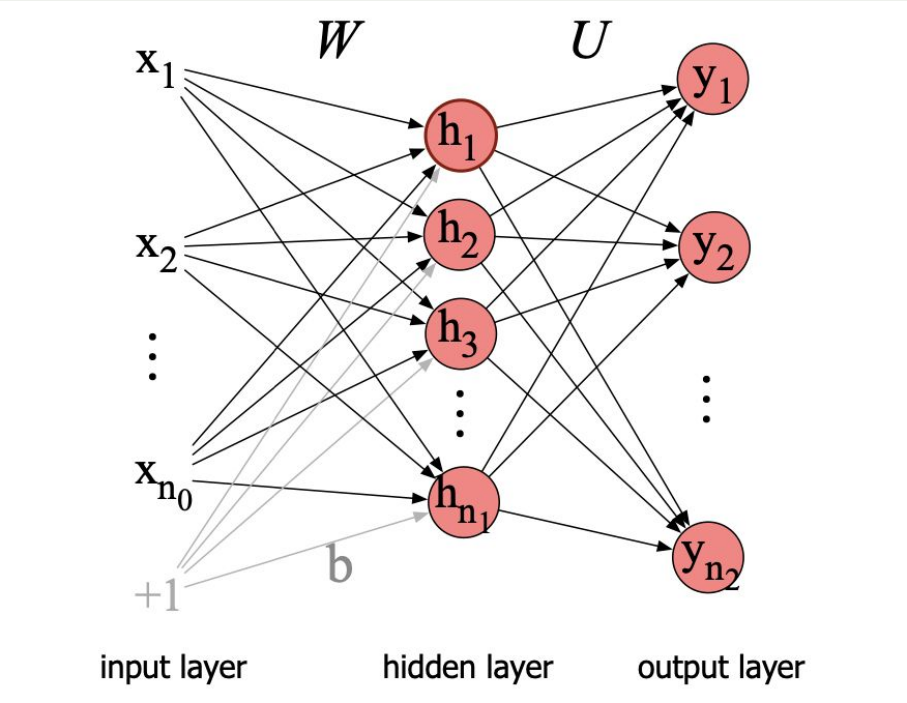


Feed-Forward Network

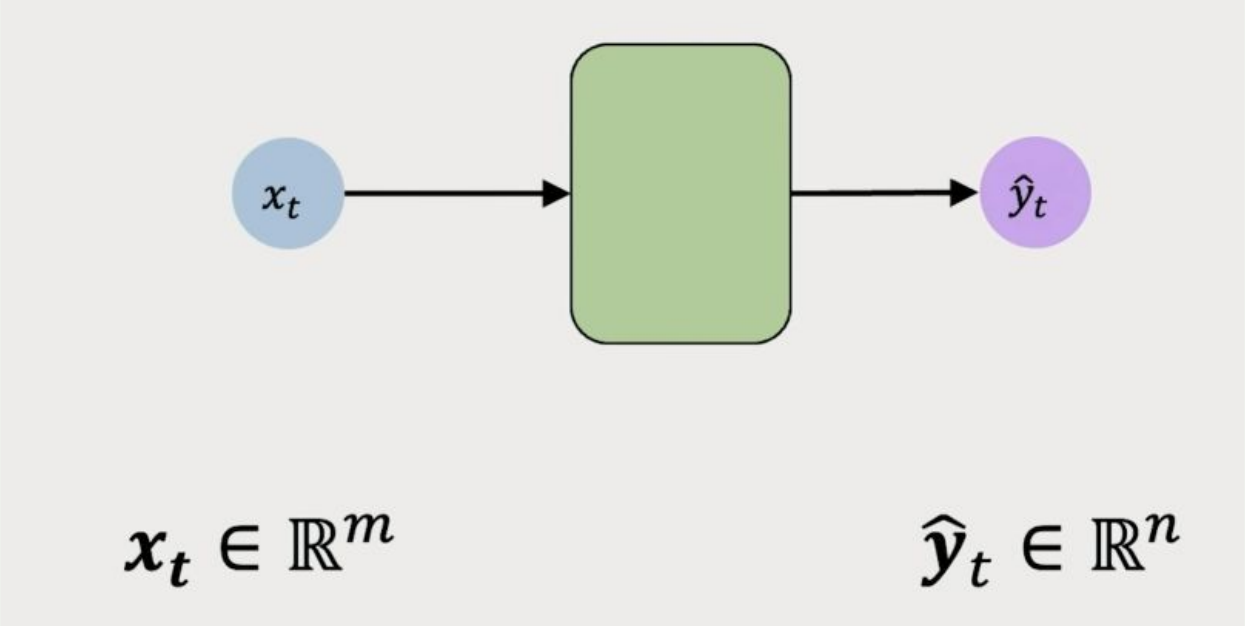
A **Feed-Forward Neural Network (FNN)**, also called a *multi-layer perceptron (MLP)*, is a basic neural architecture where information flows in one direction

layers of perceptrons → can model nonlinear decision boundaries and richer semantic relationships.

No notion of time stap of sequence

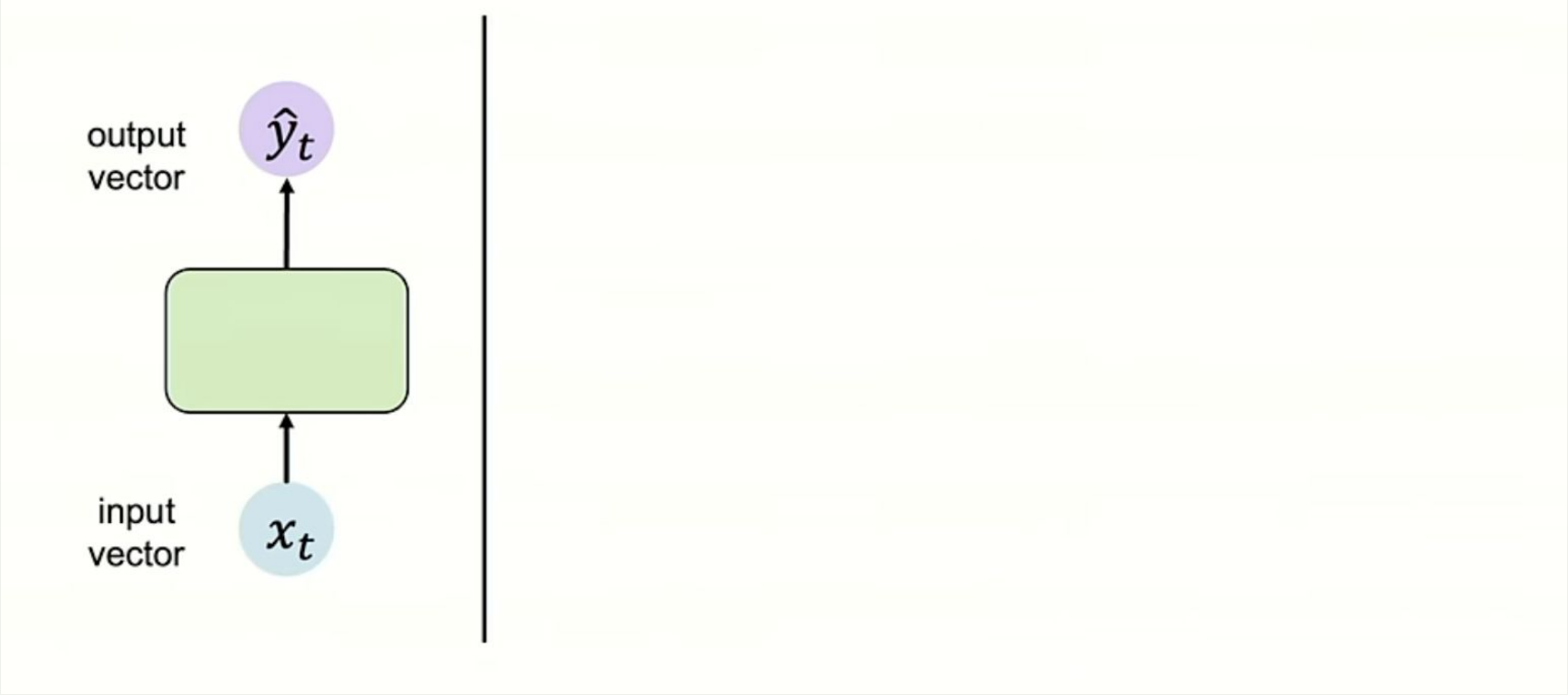


Feed-Forward Network Revisited

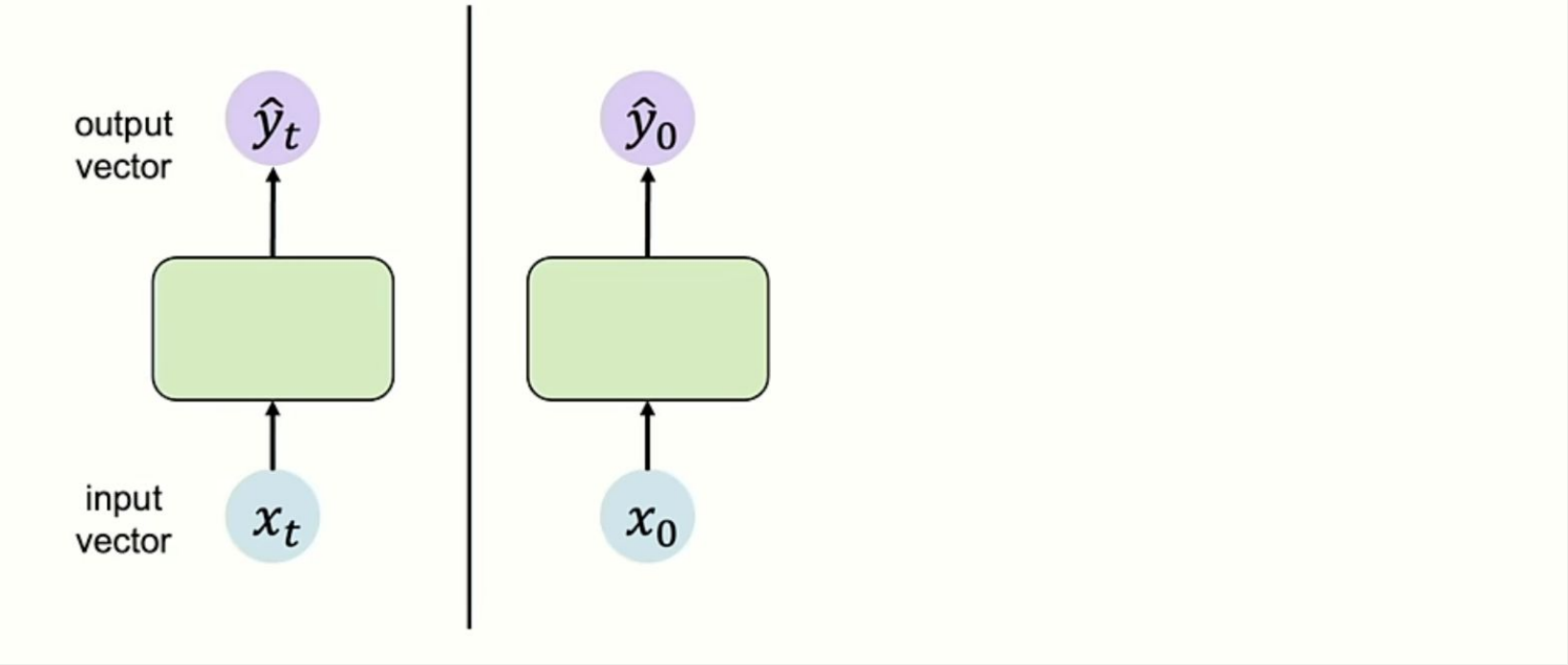


- Still no notion don't handle sequence data
- How can we handle sequence data?

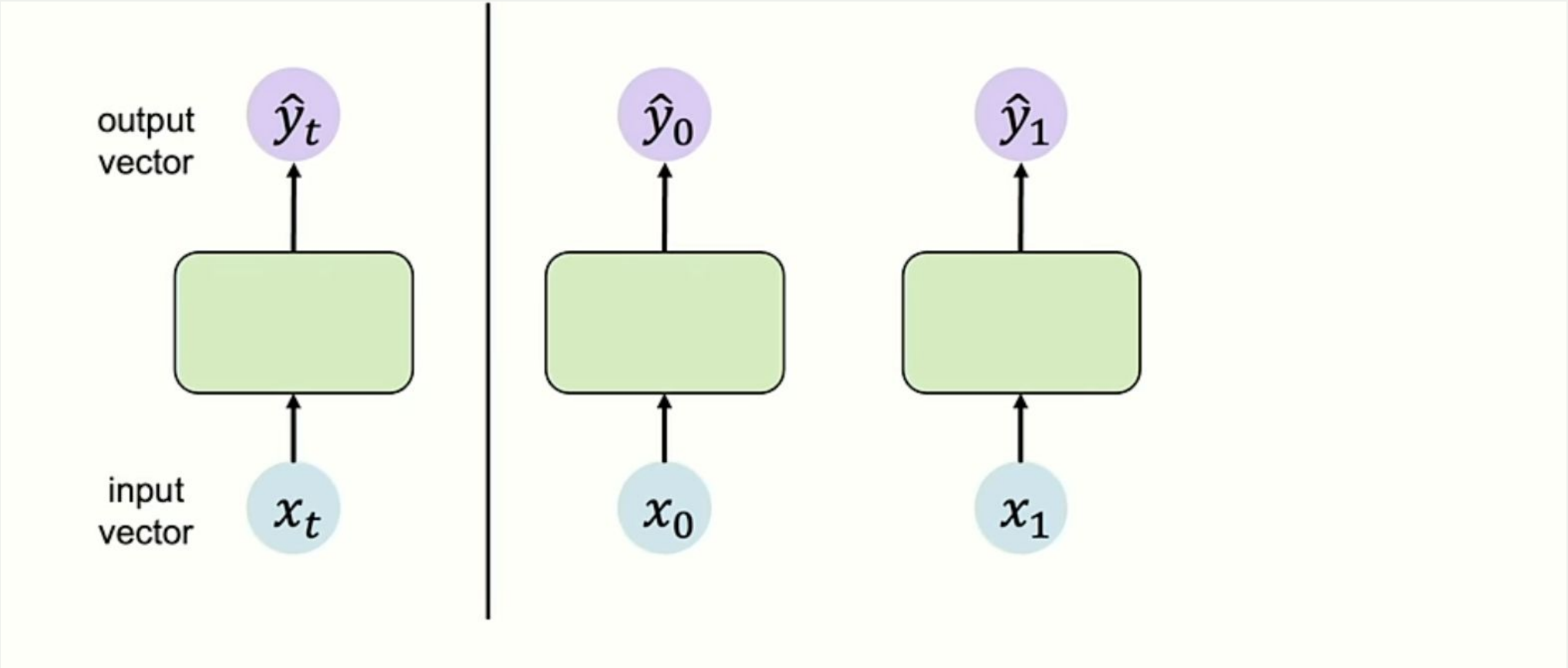
Handling Individual Time-Step



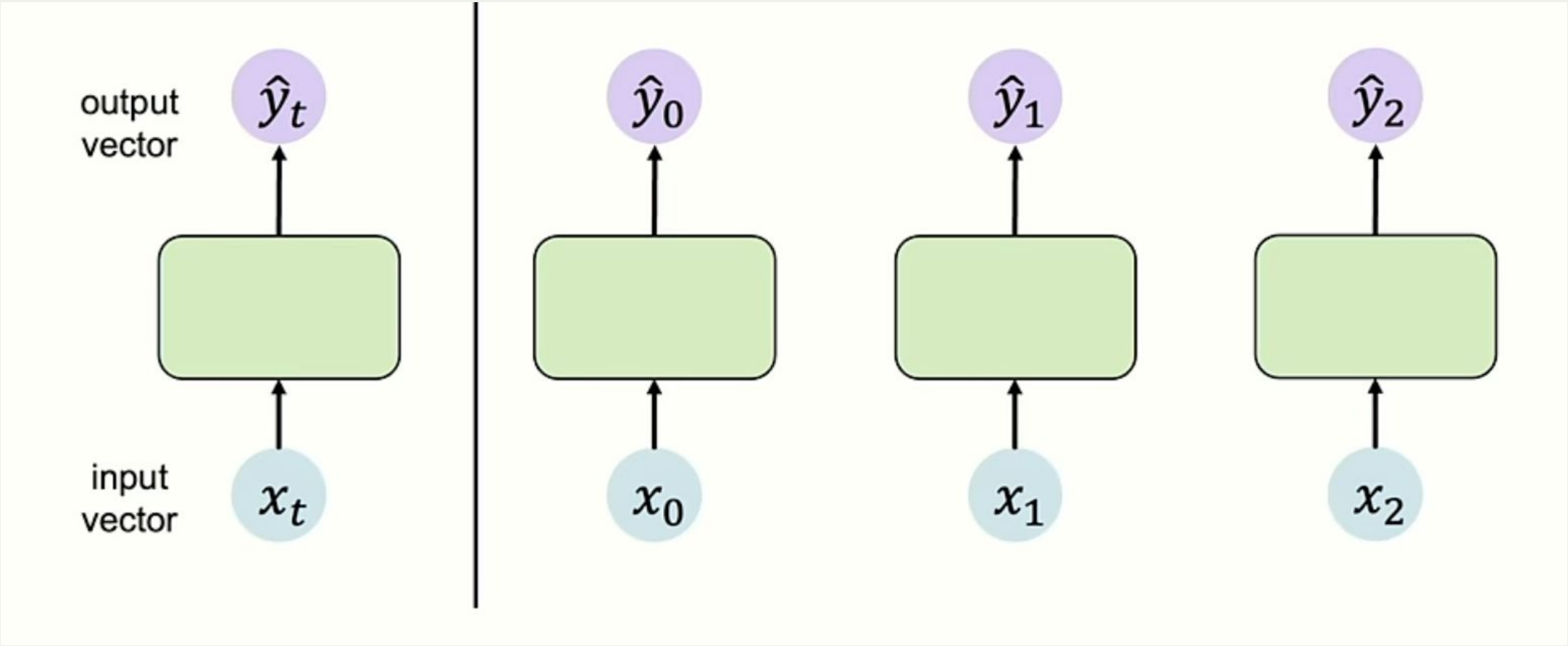
Handling Individual Time-Step



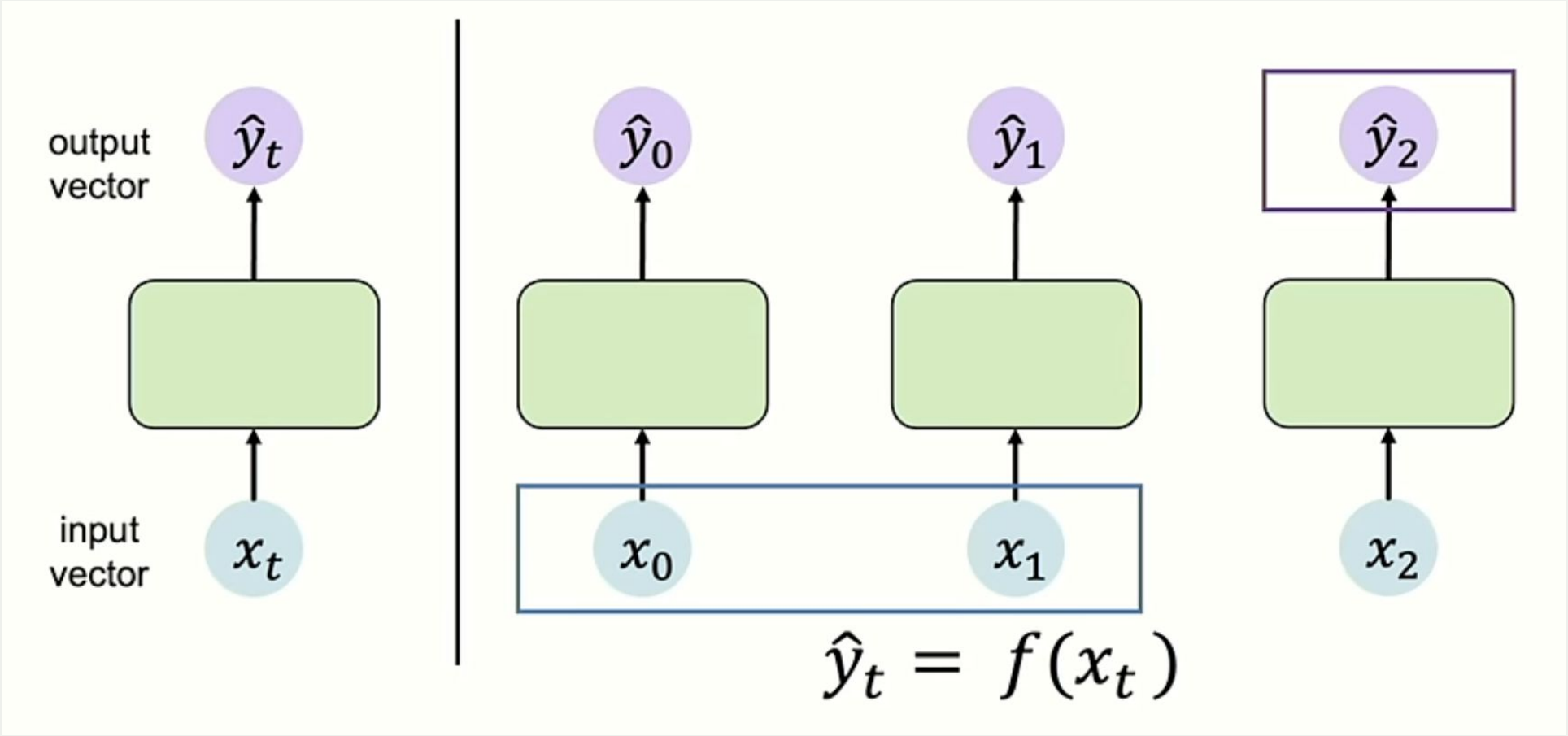
Handling Individual Time-Step



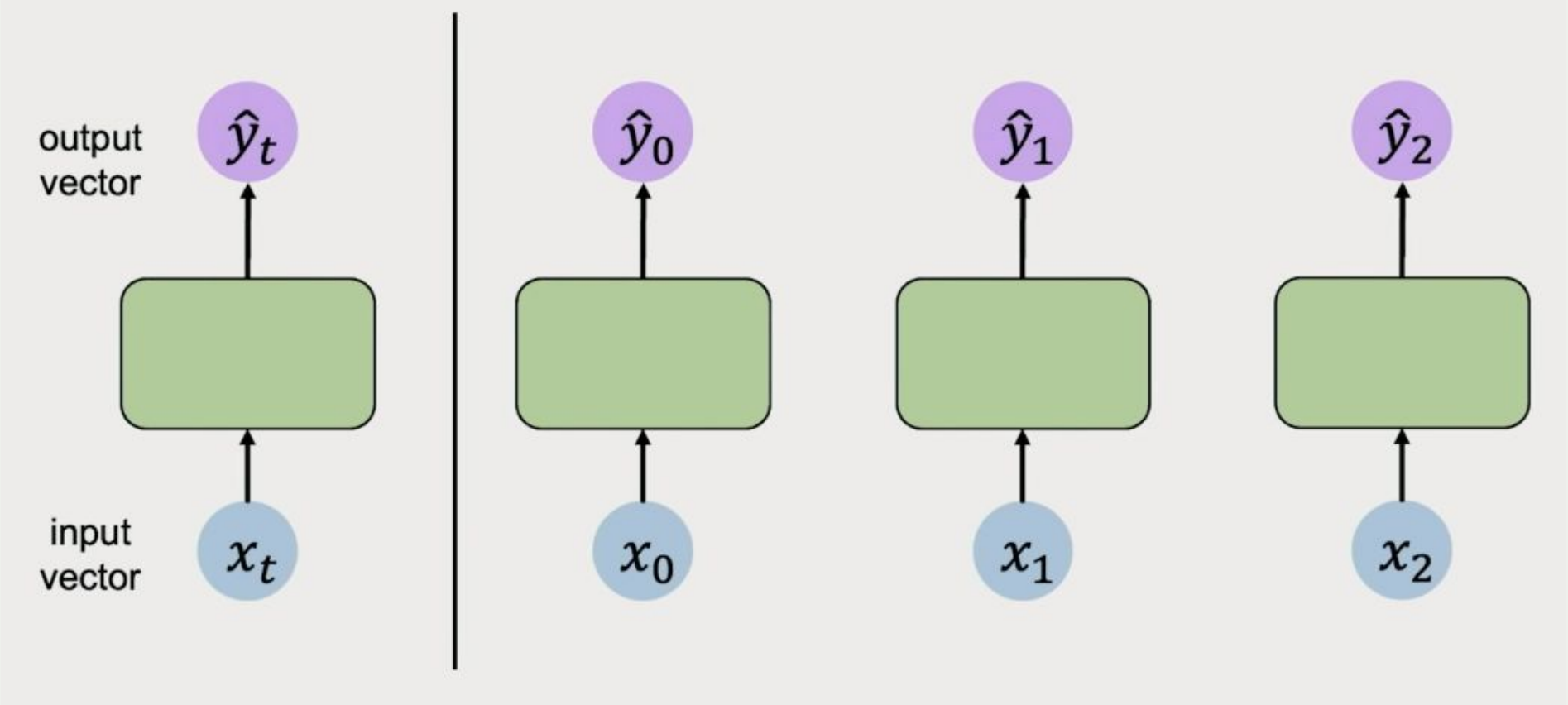
Handling Individual Time-Step



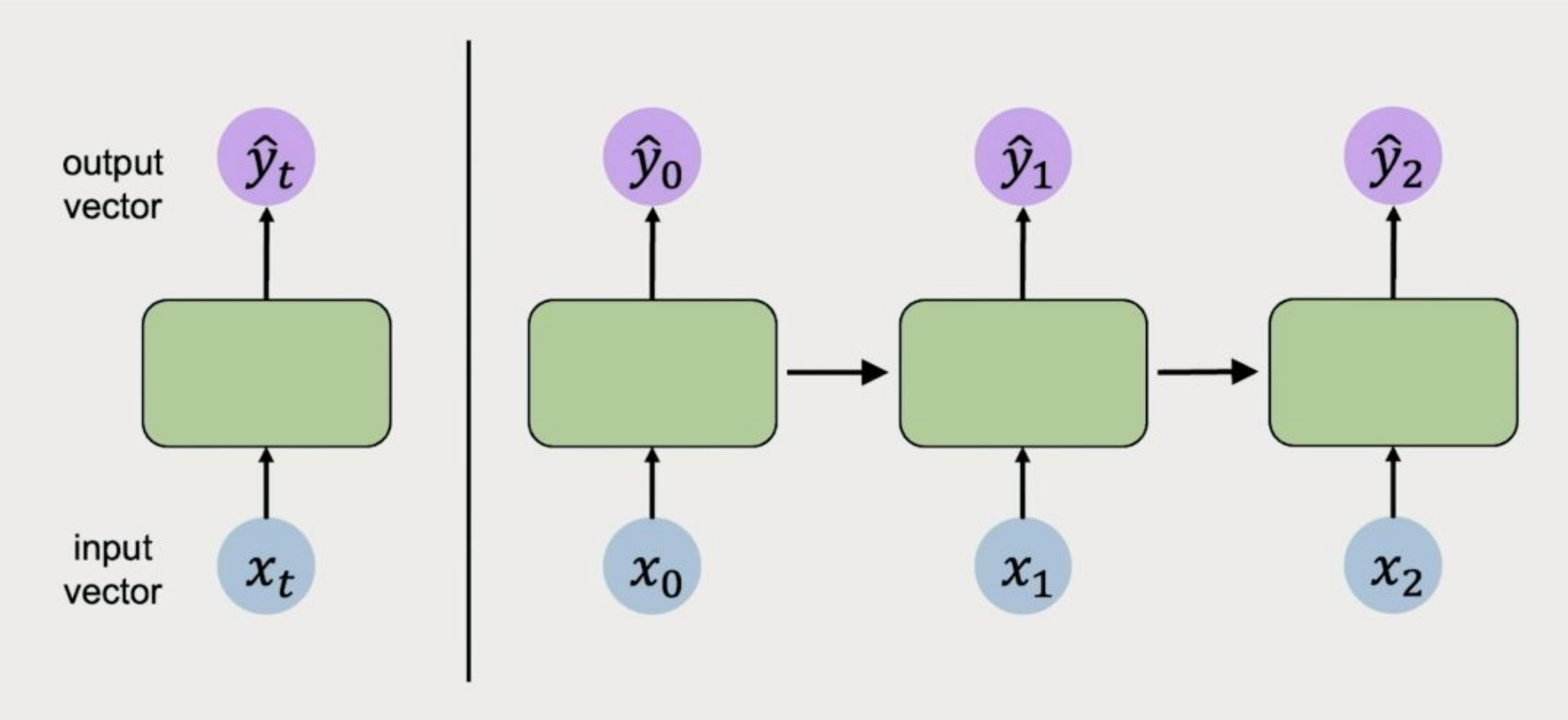
Handling Individual Time-Step



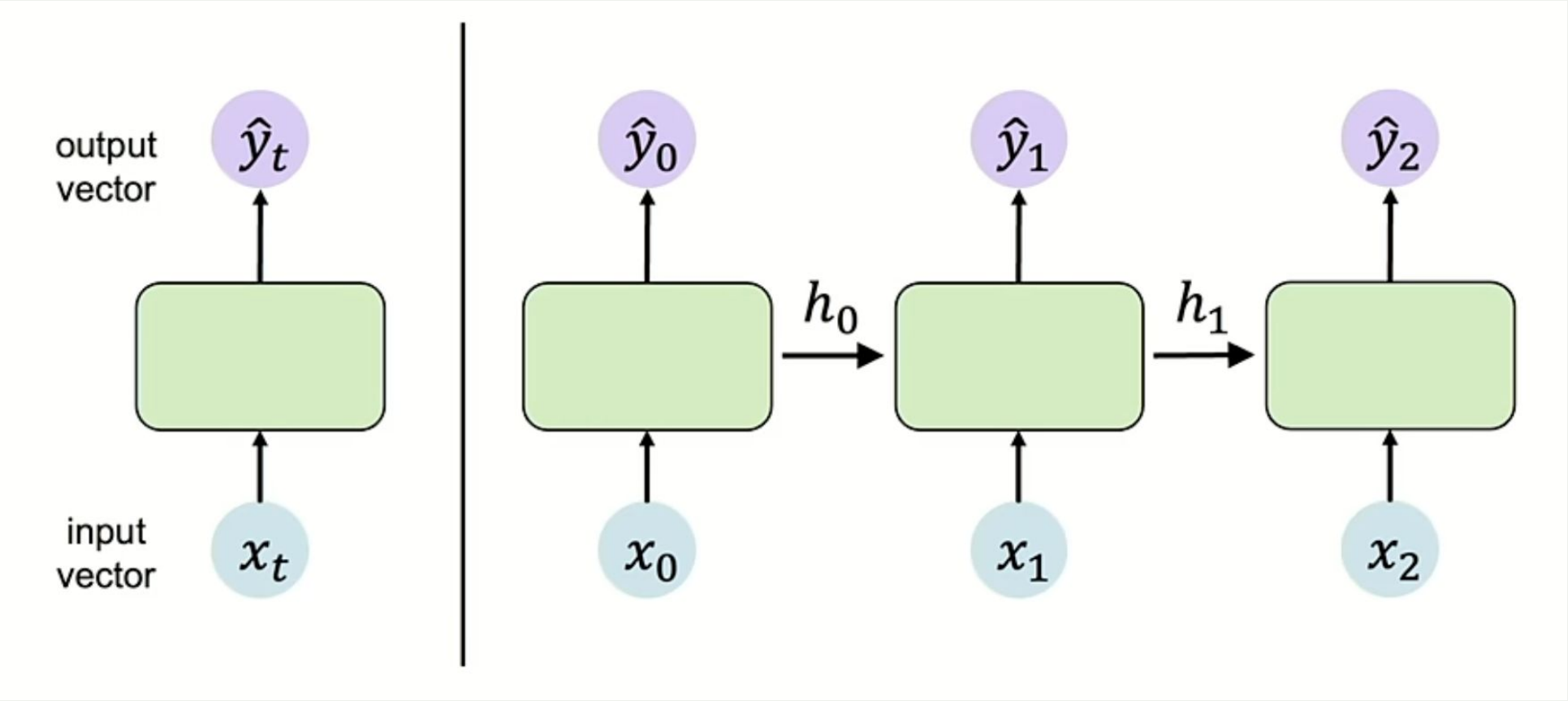
Neurons with Recurrent



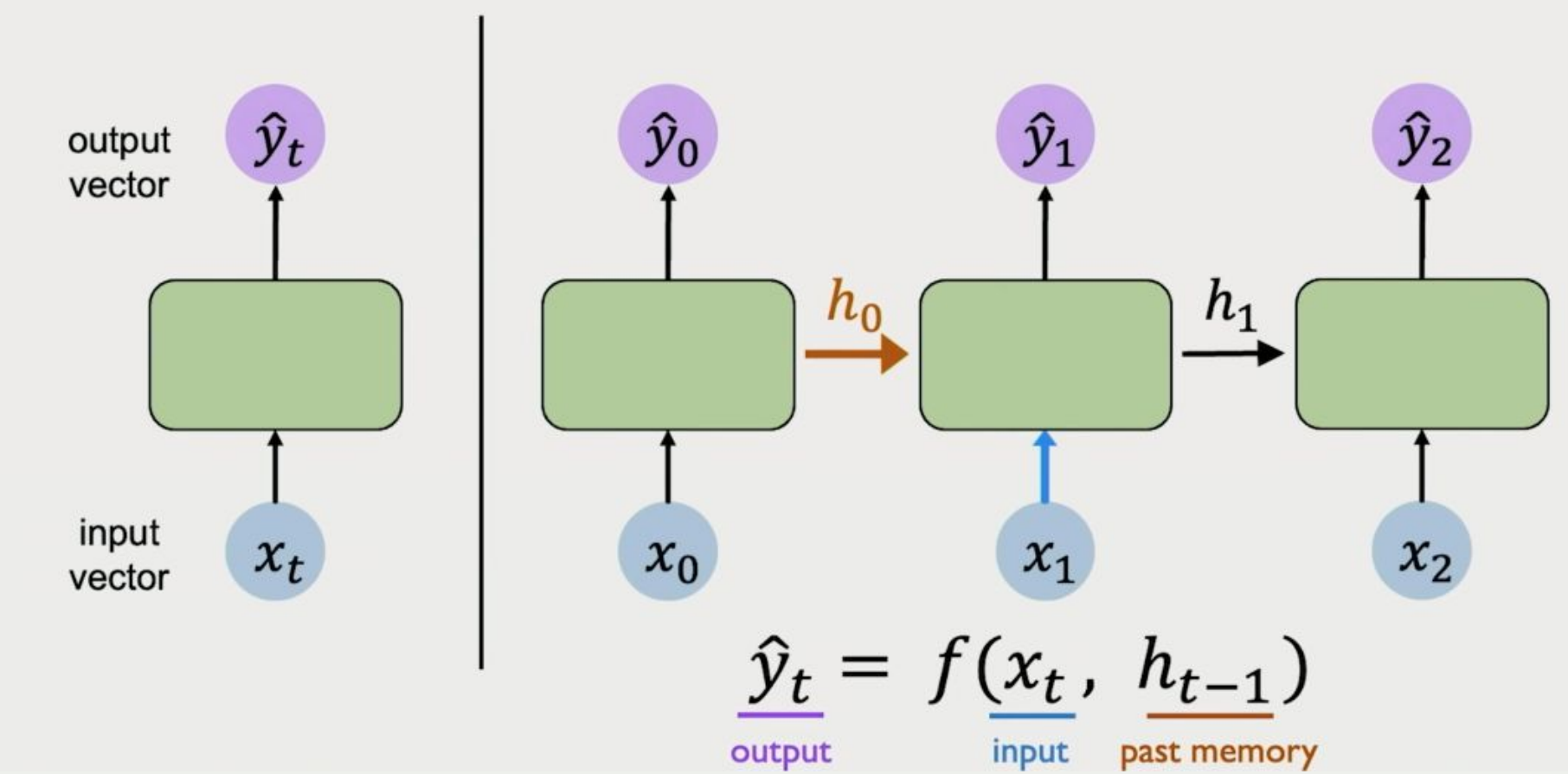
Neurons with Recurrent



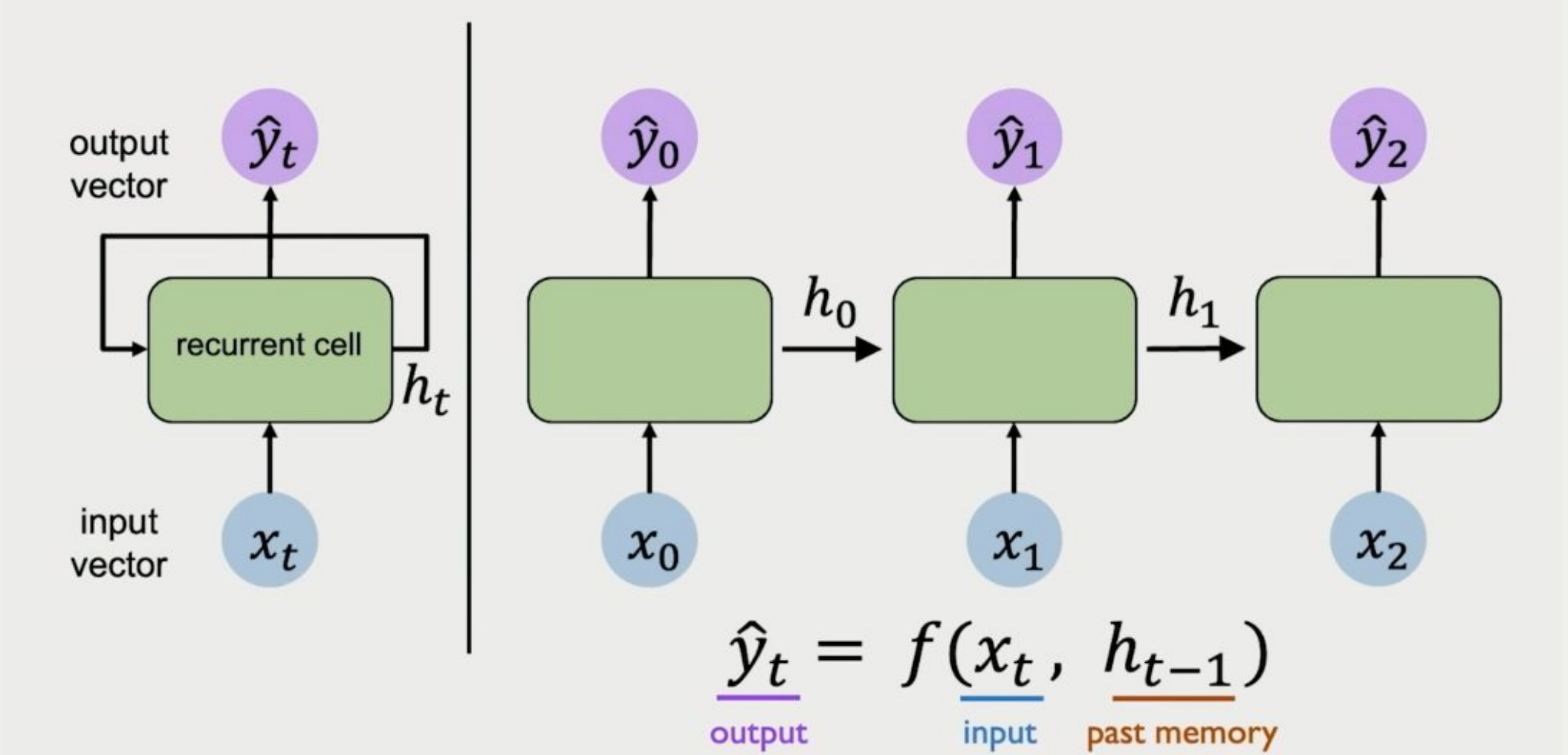
Neurons with Recurrent



Neurons with Recurrent



Neurons with Recurrent



Recurrent

The term "recurrent" generally means occurring repeatedly or in cycles, and in the context of Recurrent Neural Networks (RNNs), it specifically refers to the mechanism of repeatedly applying the same computational process to sequential data.

The recurrence enables the network to "remember" previous states over time, giving RNNs the ability to capture patterns and dependencies across sequential data.

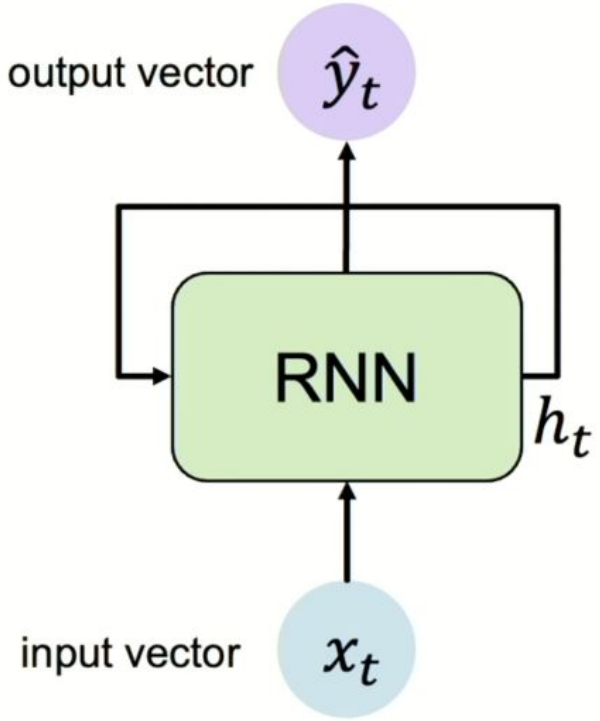
Recurrent

The recurrence is foundational concept sequence modelling called Recurrent Neural Networks (RNN)

- **Text sequences** (e.g., sentences in natural language processing),
- **Time-series data** (e.g., stock prices, sensor readings),
- **Speech data** (e.g., audio signals),
- **Video frames** (e.g., in action recognition).

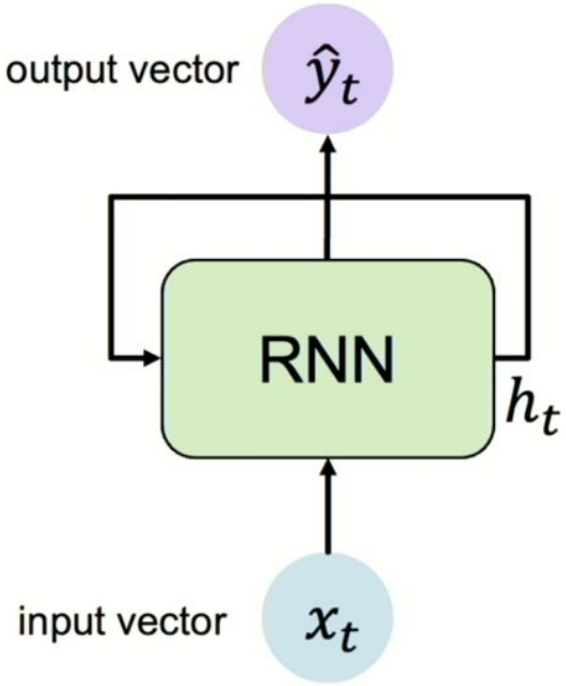
Recurrent Neural Networks (RNNs)

Recurrent Neural Network



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

Recurrent Neural Network



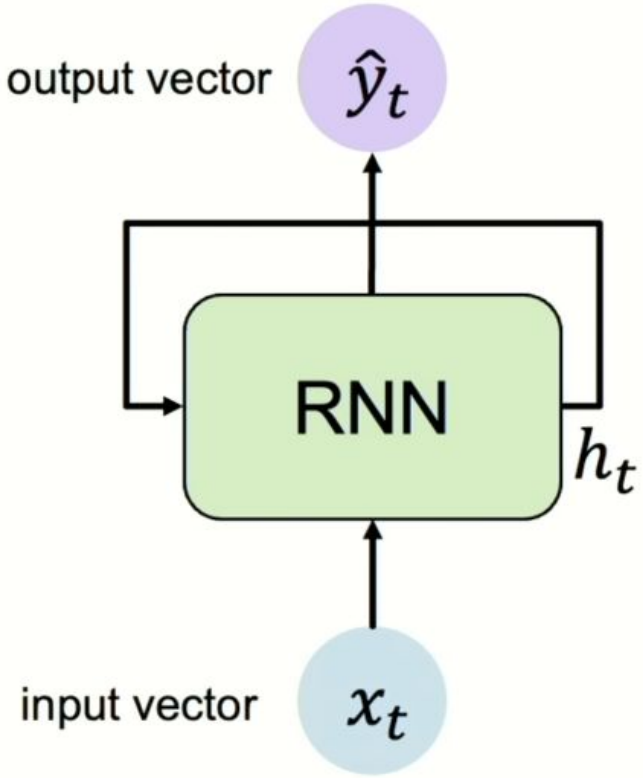
Apply a **recurrence relation** at every time step to process a sequence:

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

cell state

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

Recurrent Neural Network



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

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

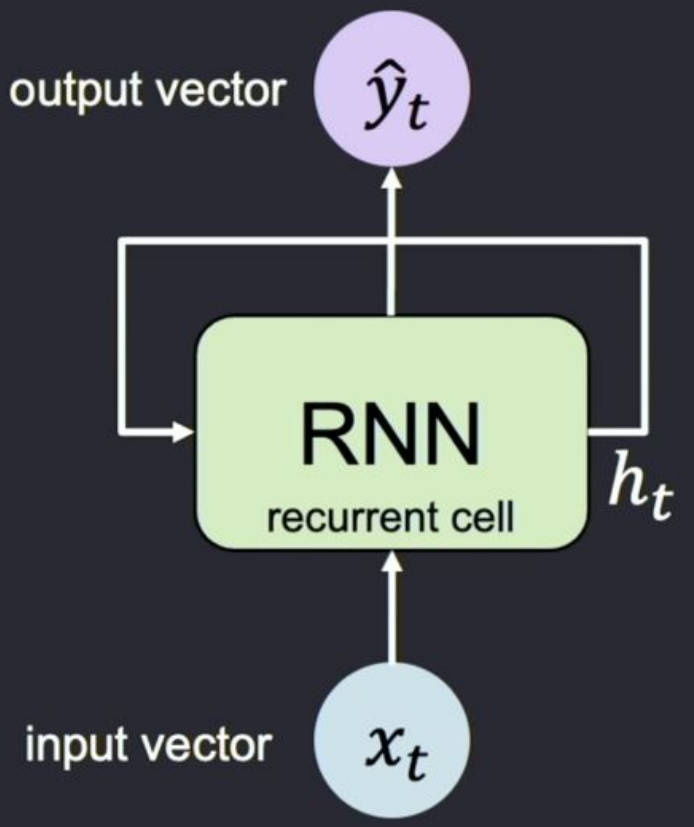
cell state | function with weights W | input | old state

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

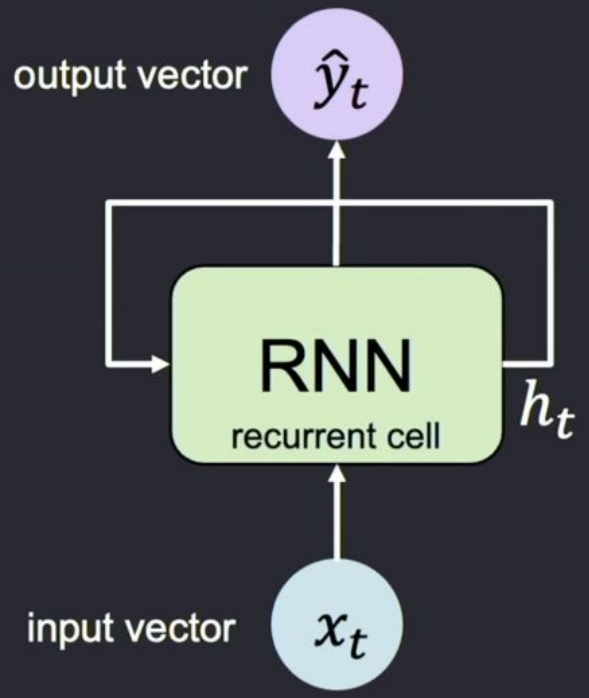
RNN Intuition: Pseudocode

```
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = my_rnn(word, hidden_state)

next_word_prediction = prediction
# >>> "networks!"
```



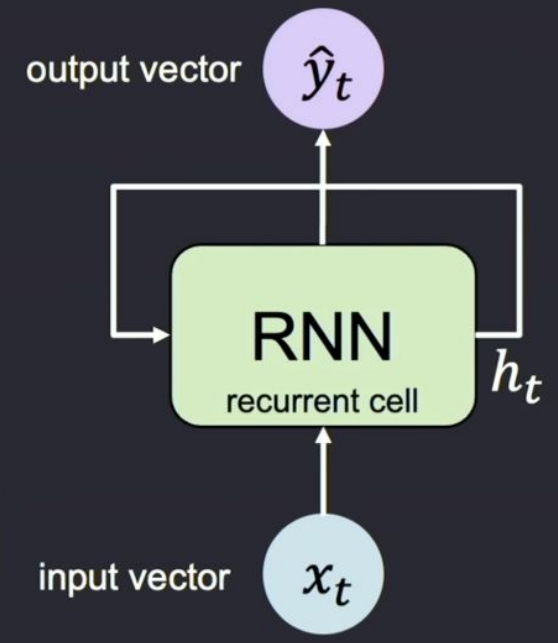
RNN Intuition: Pseudocode

```
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = my_rnn(word, hidden_state)

next_word_prediction = prediction
# >>> "networks!"
```

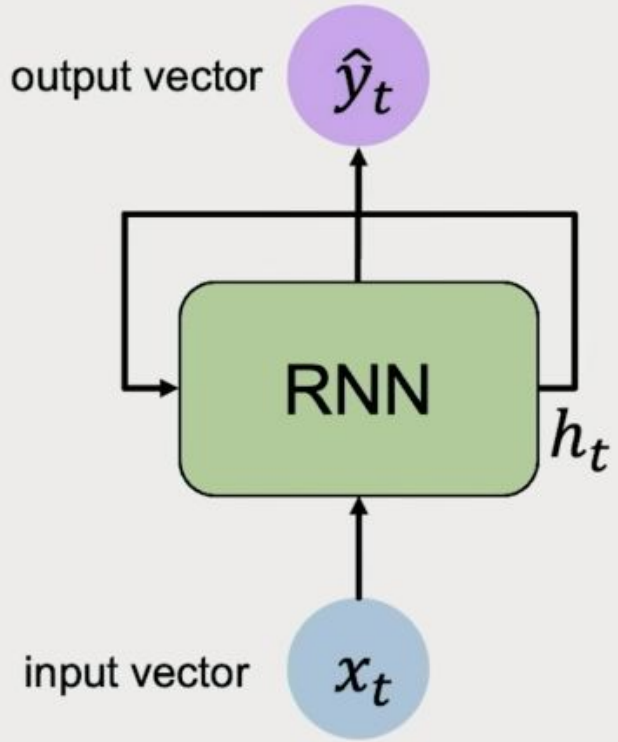


RNN Intuition: Pseudocode

```
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = my_rnn(word, hidden_state)

next_word_prediction = prediction
# >>> "networks!"
```



RNN State Update and Output



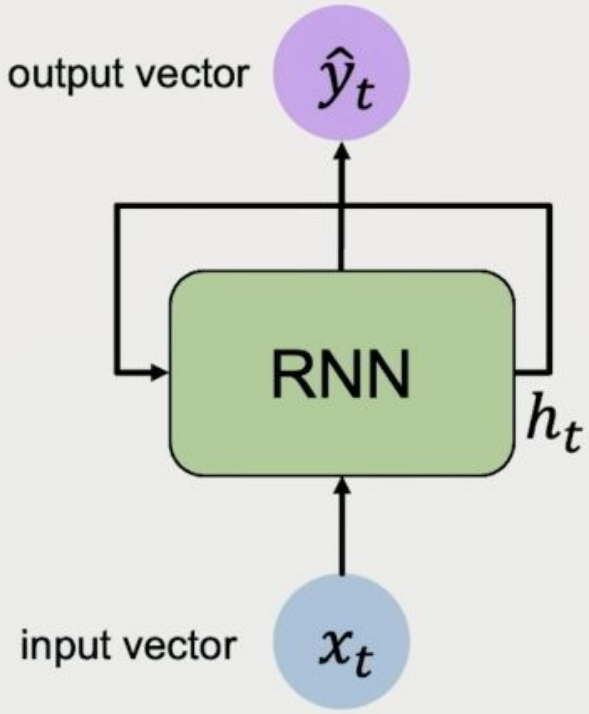
Update Hidden State

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

Input Vector

x_t

RNN: Computational Graph Across Time



Output Vector

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

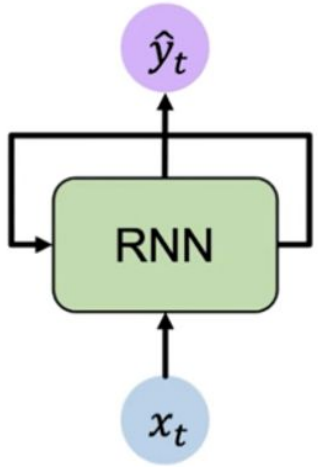
Update Hidden State

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

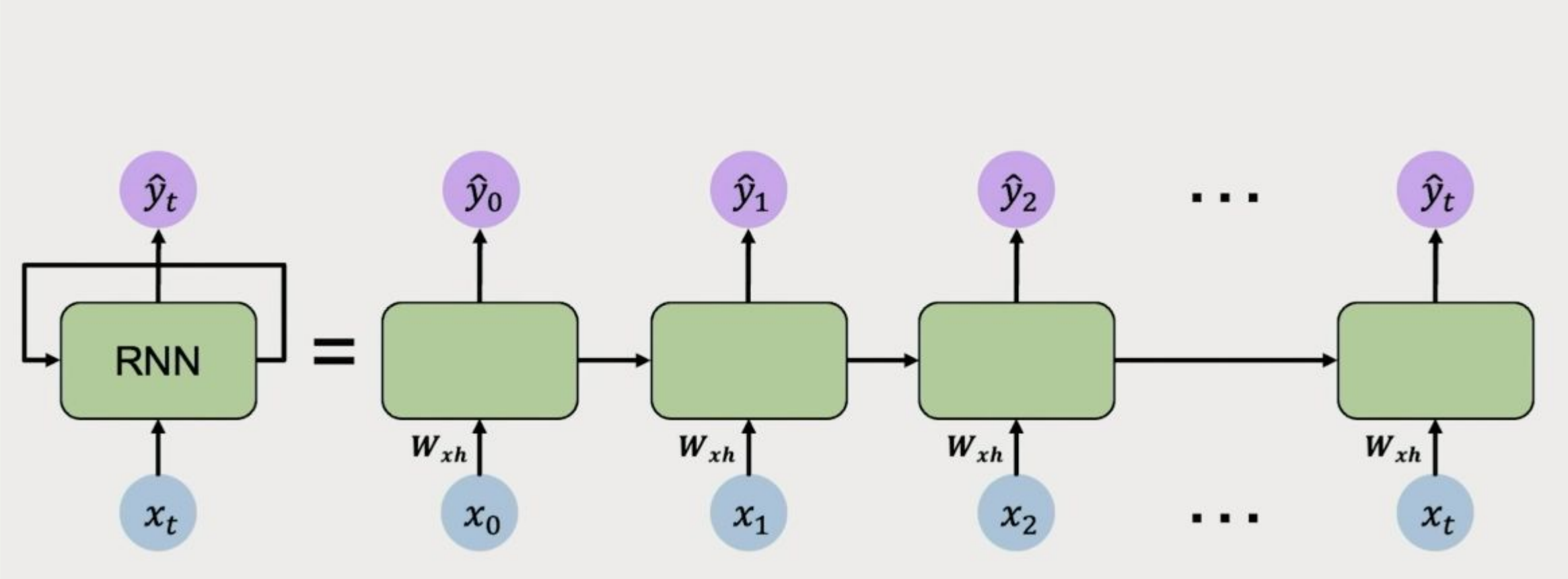
Input Vector

$$x_t$$

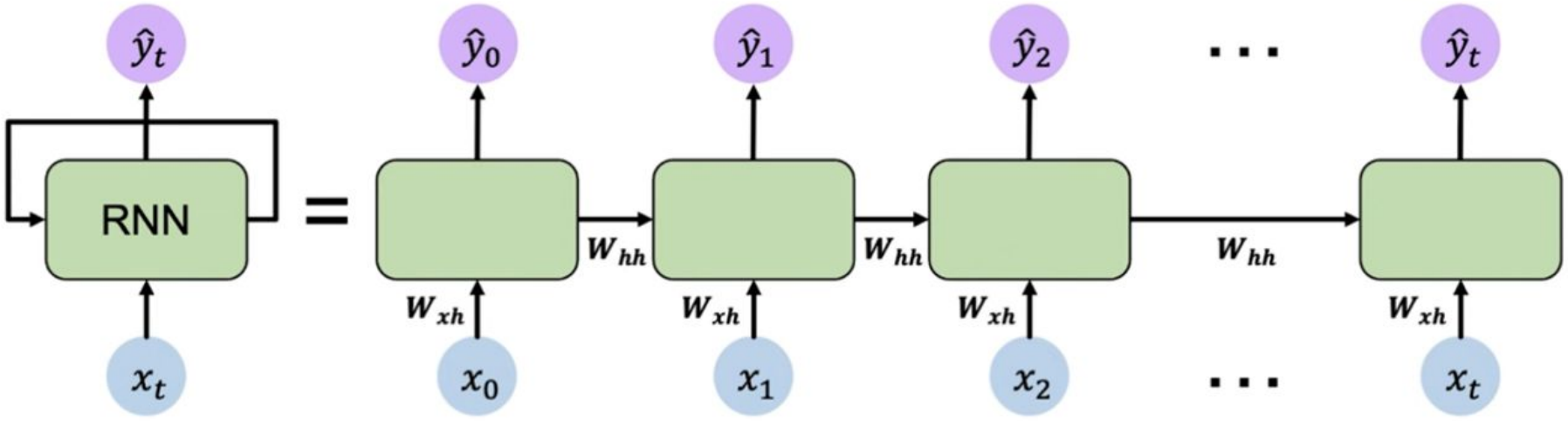
RNN: Computational Graph Across Time



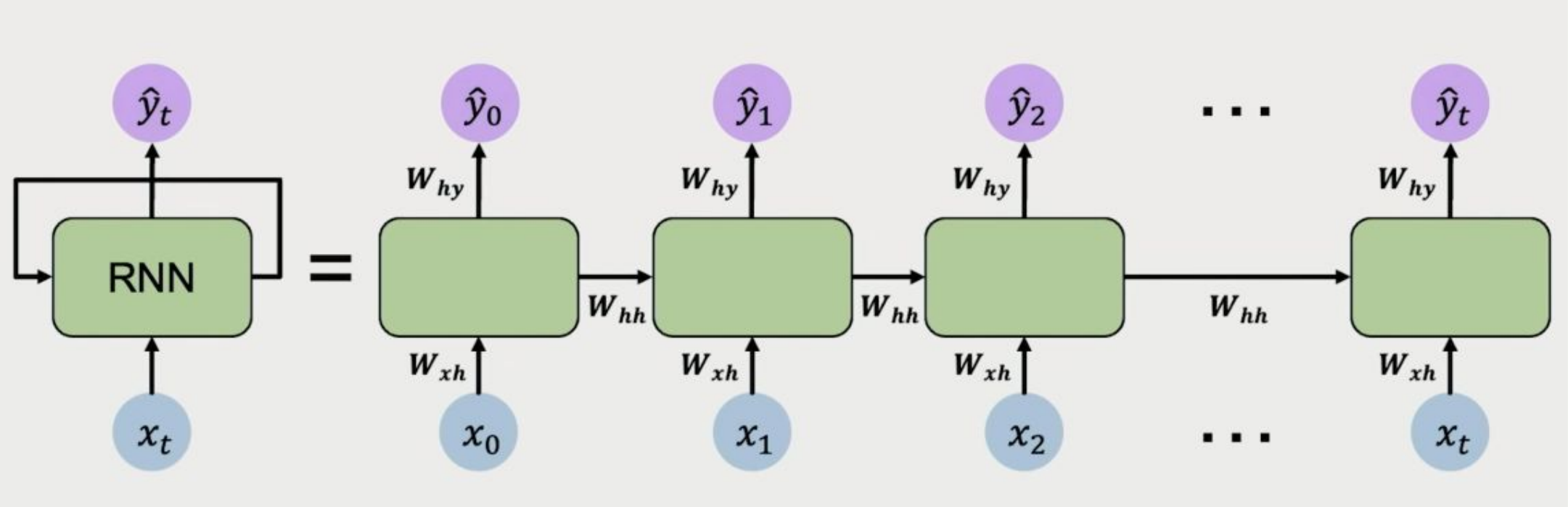
RNN: Computational Graph Across Time



RNN: Computational Graph Across Time

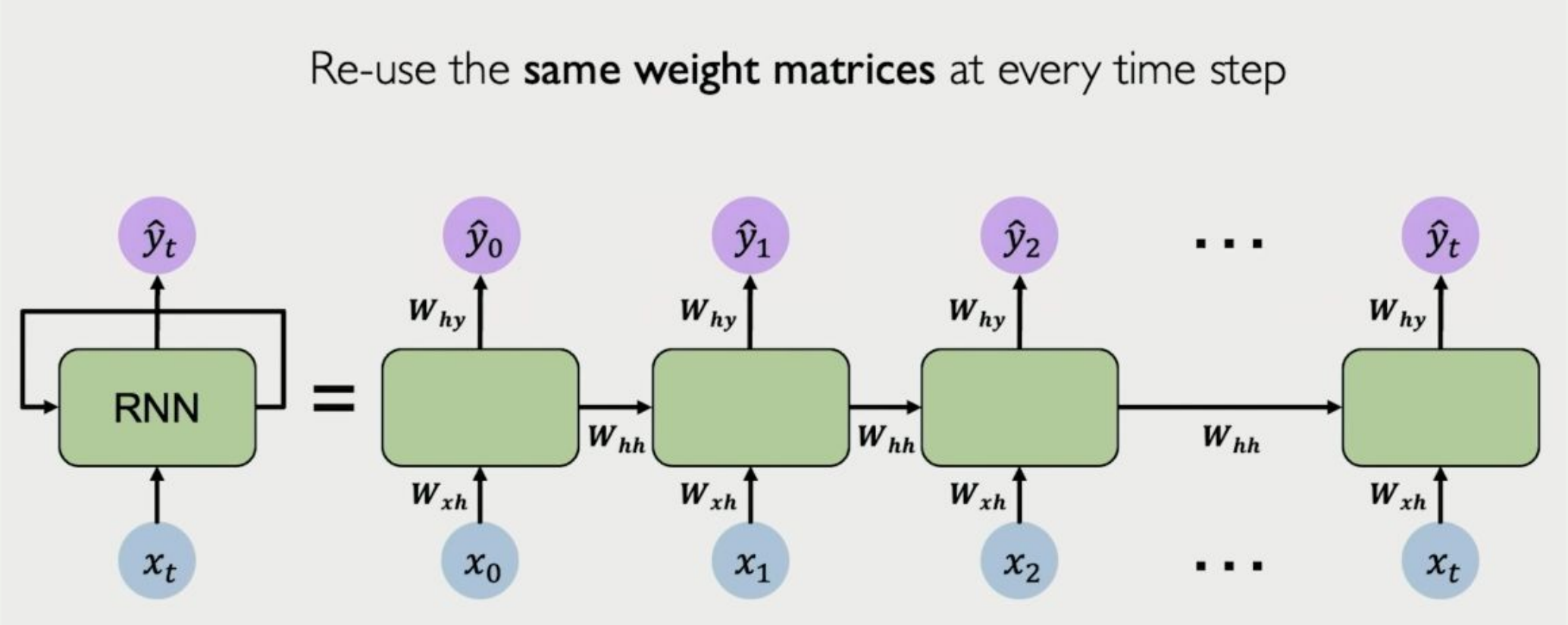


RNN: Computational Graph Across Time

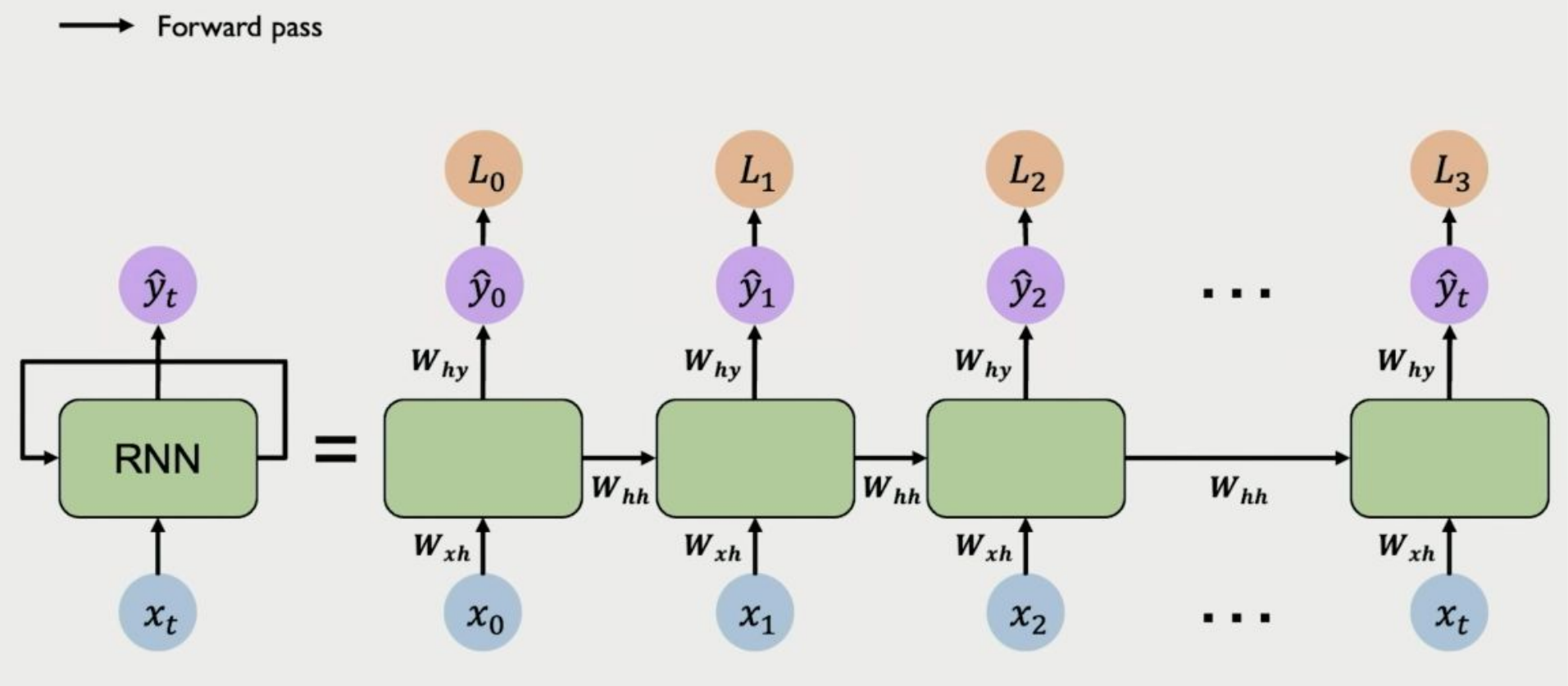


Reuse the same weight at every time step

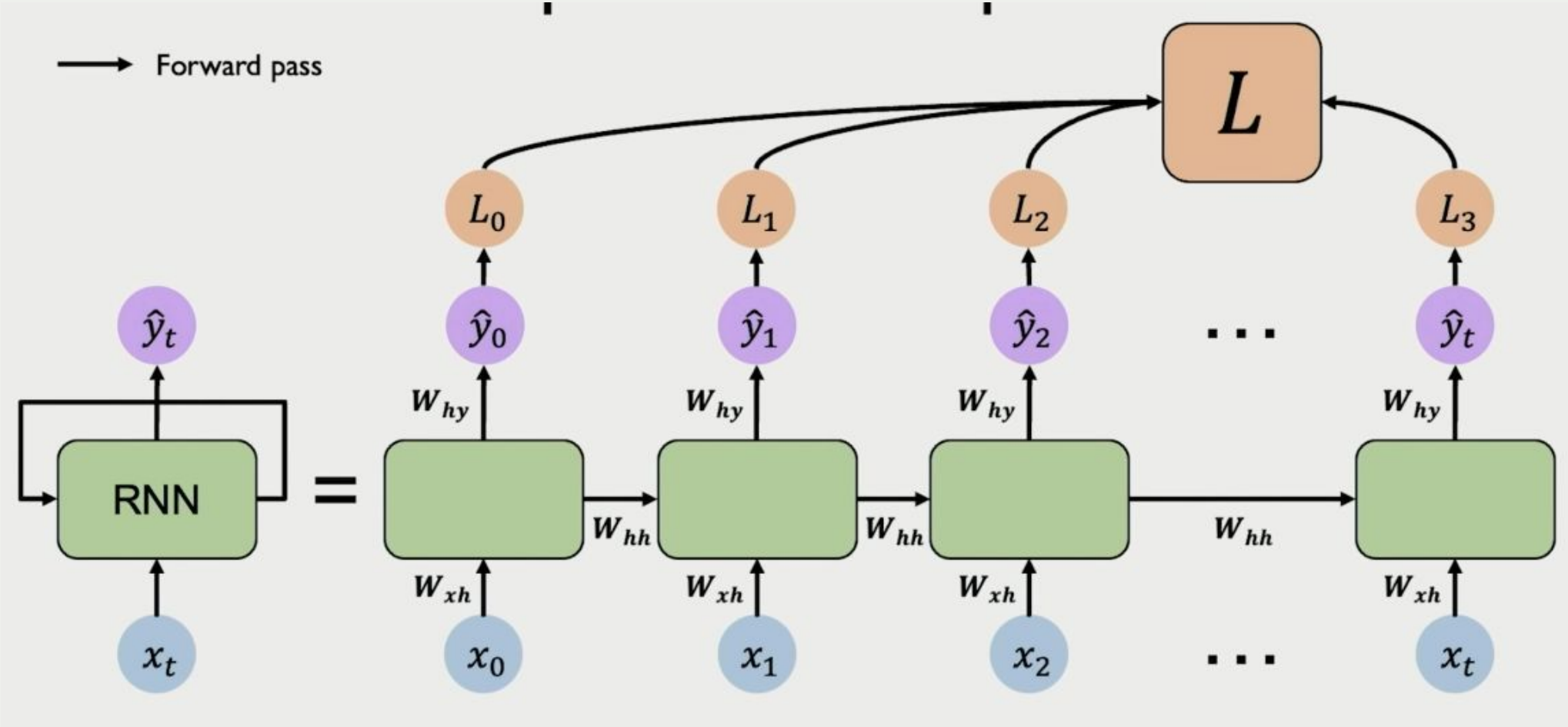
RNN: Computational Graph Across Time



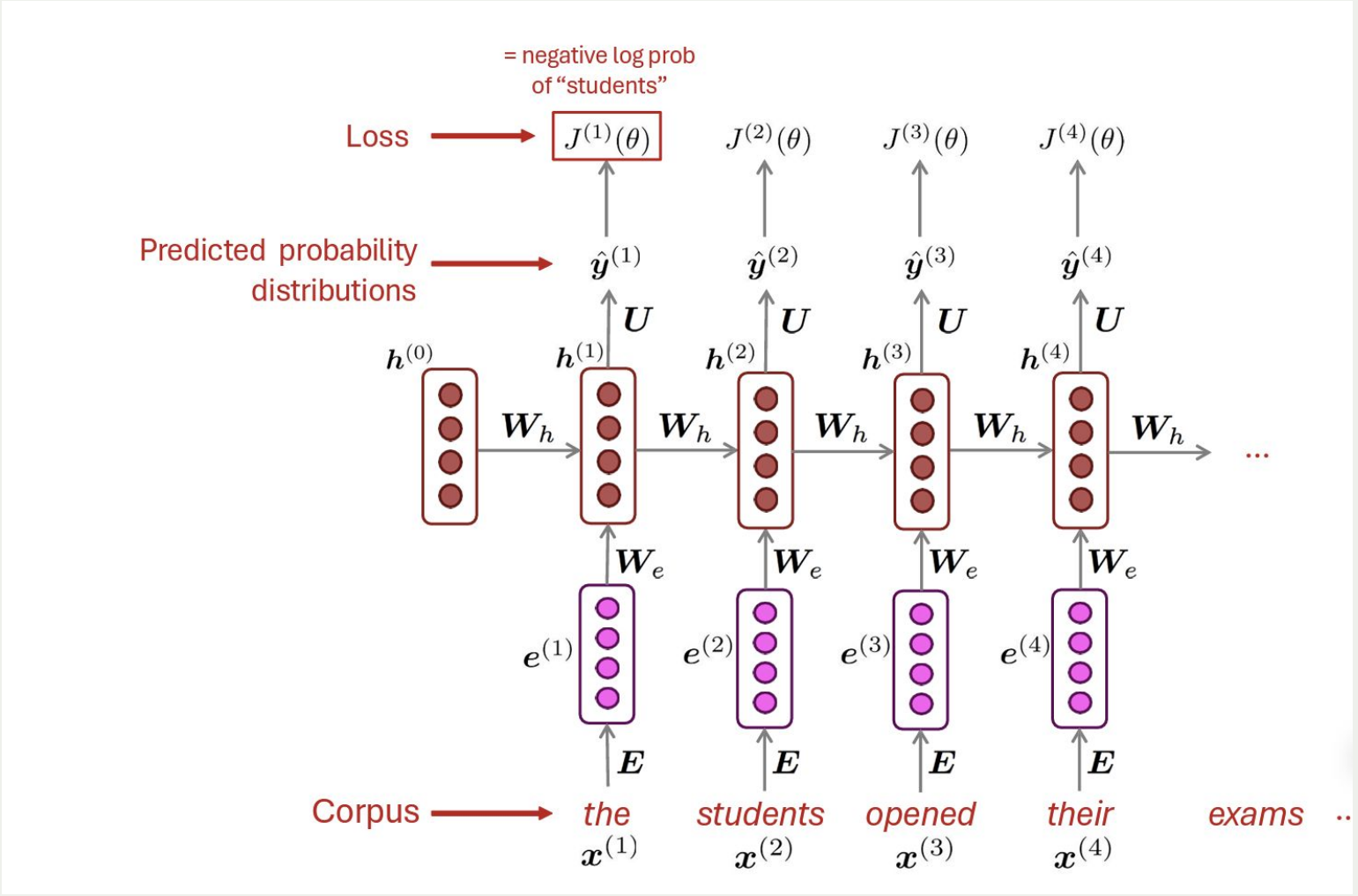
RNN: Computational Graph Across Time



RNN: Computational Graph Across Time



Training an RNN Language Model

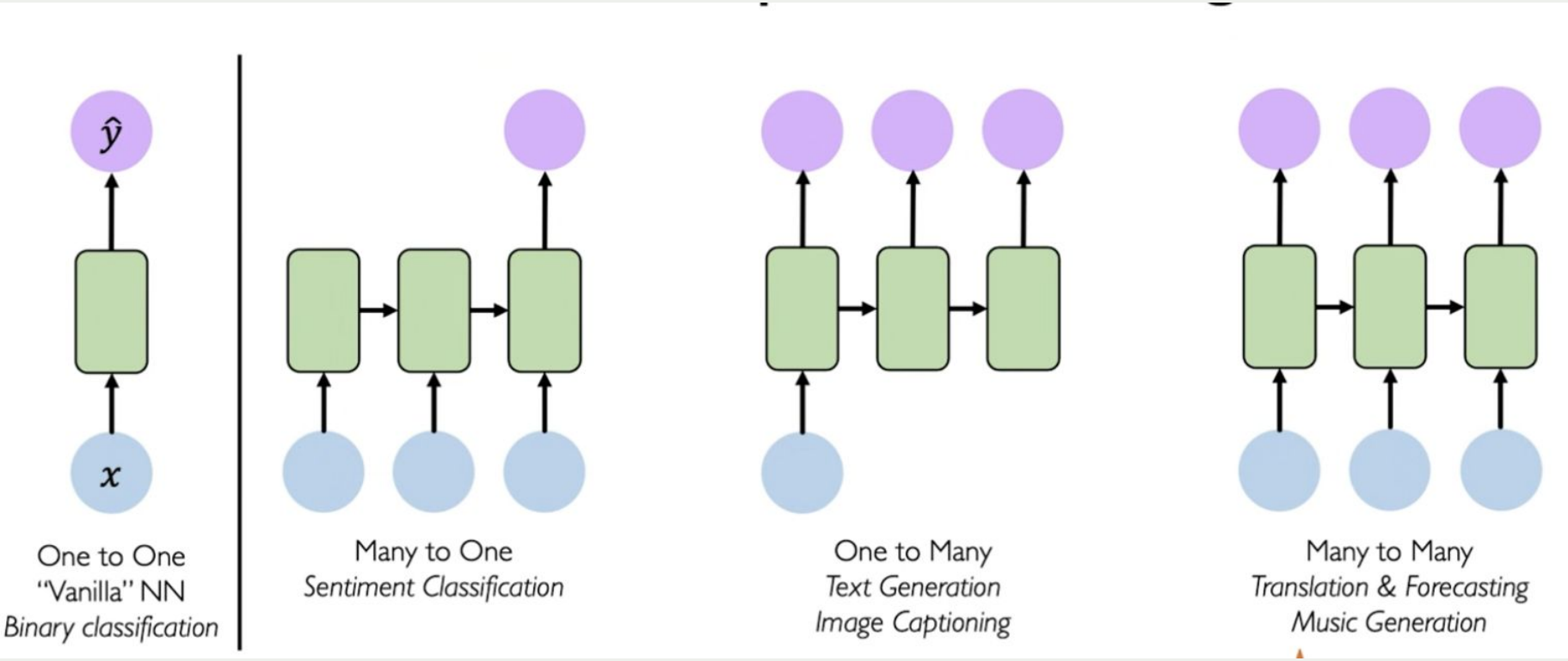


Chapter 6 Neural Networks: Speech and Language Processing. Daniel Jurafsky & James H. Martin.

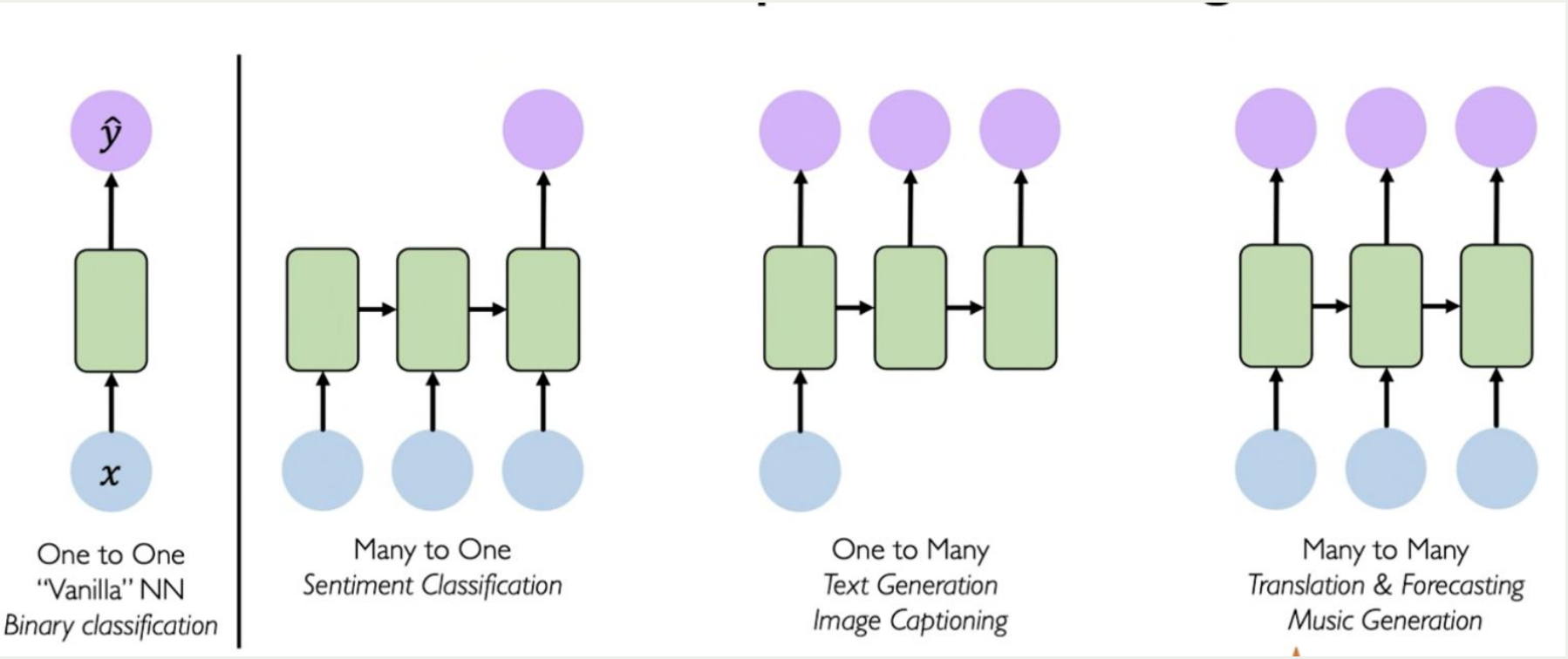
Practical Session: RNN from Scratch

- CNN for text classification
- RNN for text classification

RNNs for Sequence Modelling



RNNs for Sequence Modelling

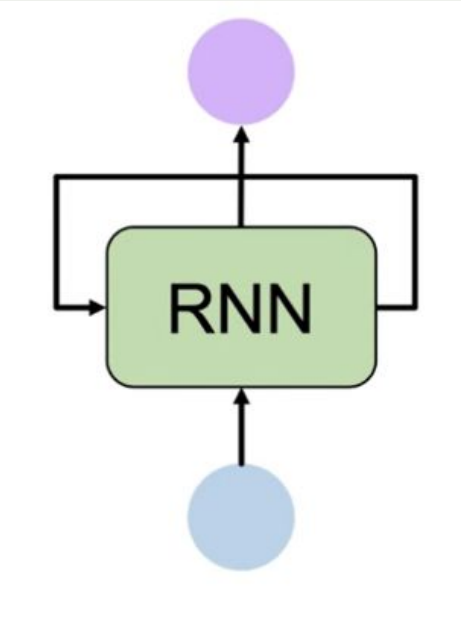


… and many other architectures and applications

Sequence Modelling: Design Criteria

To model sequences, we need to:

- Handle variable-length sequences
- Track long-term dependencies
- Maintain information about order
- Share parameters across the sequence

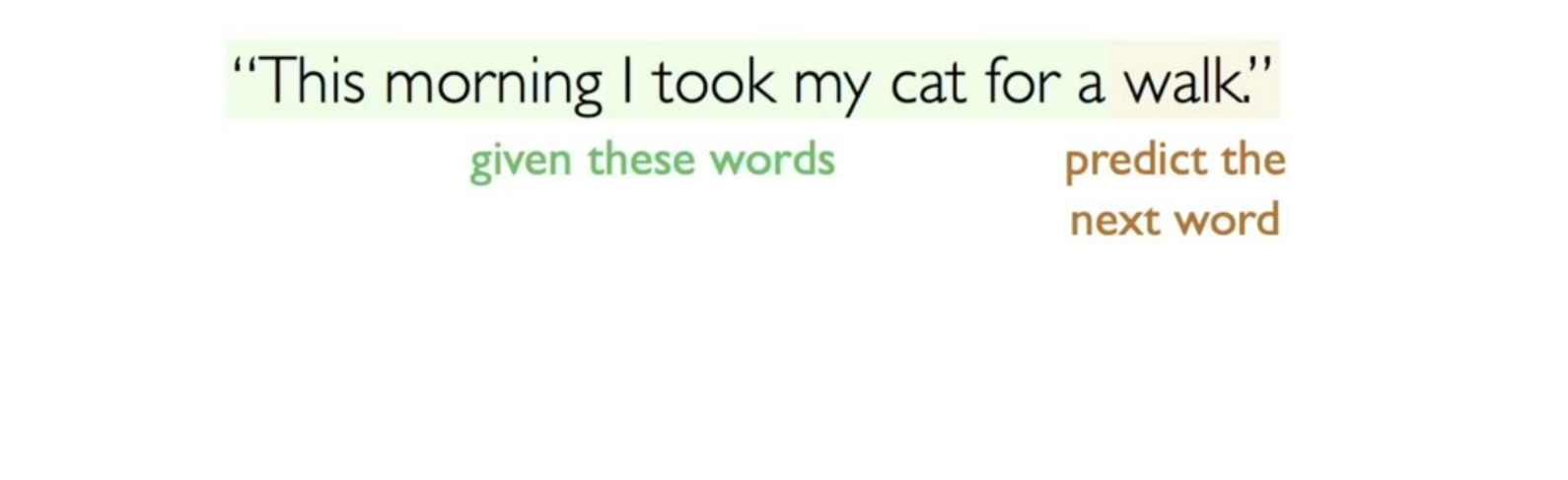


Recurrent Neural Networks (RNNs) meet these sequence modeling design criteria.

Predict the next word

A Sequence Modelling Problem

A Sequence Modelling Problem



Representing Language to a Neural Network

A Sequence Modelling Problem

"This morning I took my cat for a walk."

given these words

predict the next word

Representing Language to a Neural Network

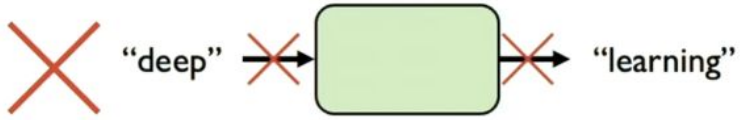
A Sequence Modelling Problem

"This morning I took my cat for a walk."

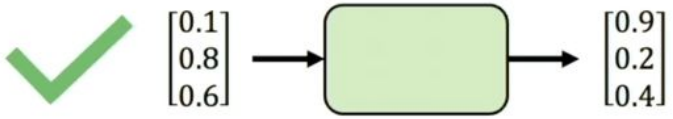
given these words

predict the next word

Representing Language to a Neural Network

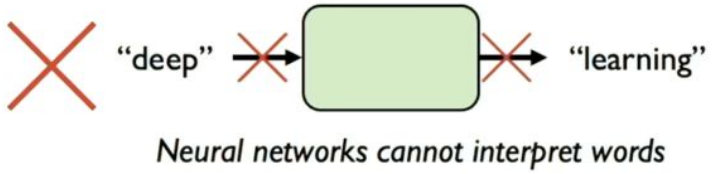


Neural networks cannot interpret words

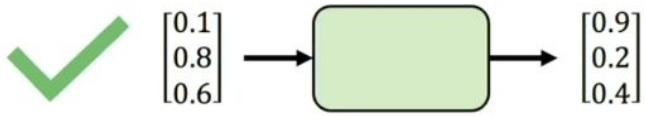


Neural networks require numerical inputs

Encoding Language for a Neural Network



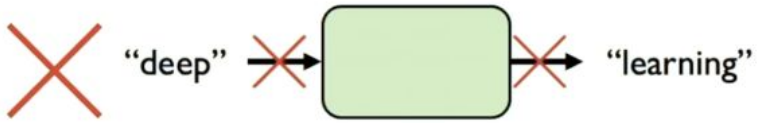
Neural networks cannot interpret words



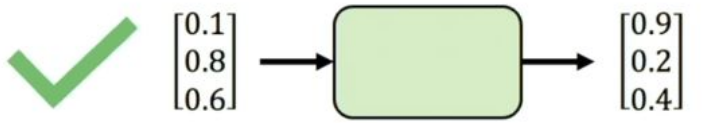
Neural networks require numerical inputs

Embedding: transform indexes into a vector of fixed size.

Encoding Language for a Neural Network

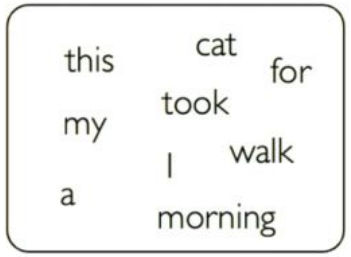


Neural networks cannot interpret words



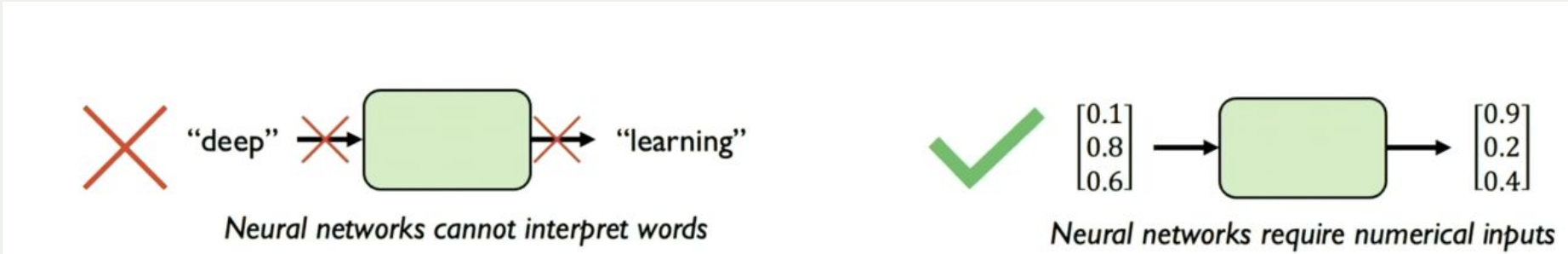
Neural networks require numerical inputs

Embedding: transform indexes into a vector of fixed size.

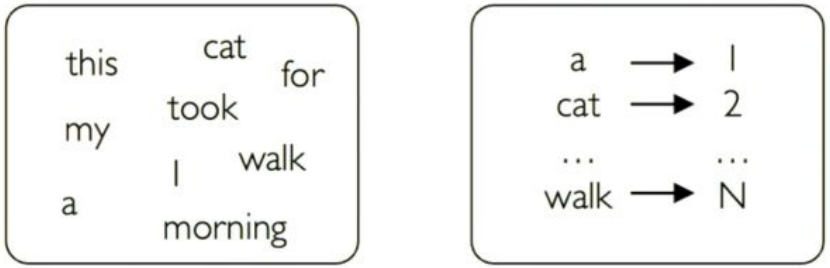


1. Vocabulary:
Corpus of words

Encoding Language for a Neural Network



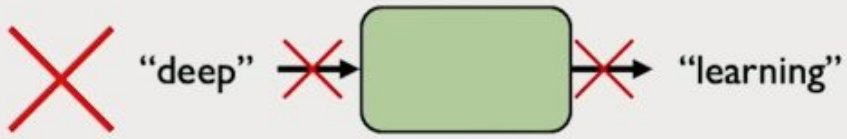
Embedding: transform indexes into a vector of fixed size.



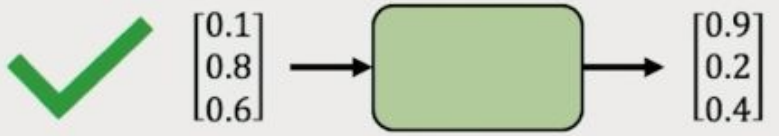
1. Vocabulary:
Corpus of words

2. Indexing:
Word to index

Encoding Language for a Neural Network

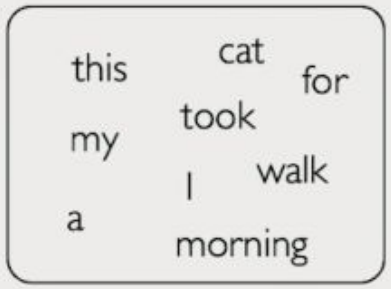


Neural networks cannot interpret words

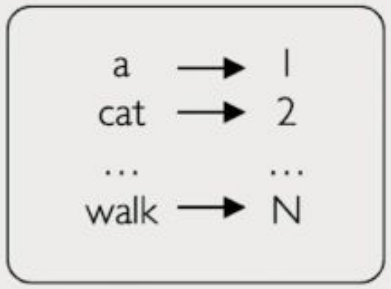


Neural networks require numerical inputs

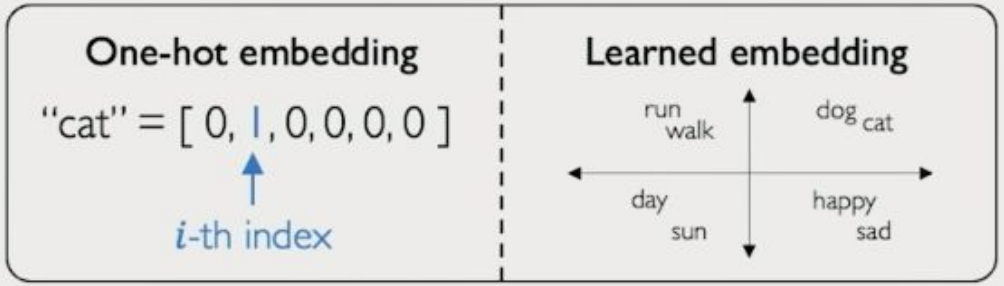
Embedding: transform indexes into a vector of fixed size.



1. Vocabulary:
Corpus of words



2. Indexing:
Word to index

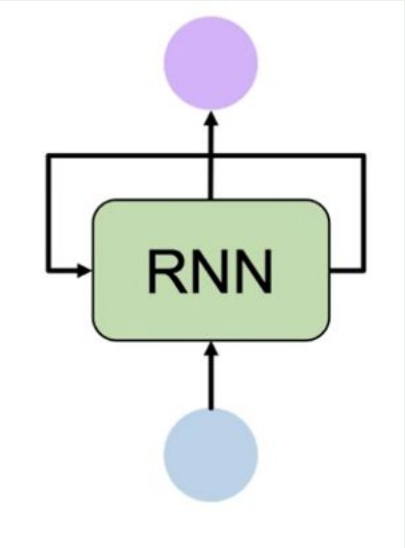


3. Embedding:
Index to fixed-sized vector

Sequence Modelling: Design Criteria

To model sequences, we need to:

- **Handle variable-length sequences**
- Track **long-term** dependencies
- Maintain information about **order**
- **Share parameters** across the sequence



Recurrent Neural Networks (RNNs) meet these sequence modelling design criteria.

Handle variable-length sequences

The food was great

vs.

We visited a restaurant for lunch

vs.

We were hungry but cleaned the house before eating

Track long-term dependencies

"**France** is where I grew up, but I now live in Boston. I speak fluent ___."

We need information from **the distant past** to accurately predict the correct word.

Capture Differences in Sequence Order

The food was good, not bad at all.

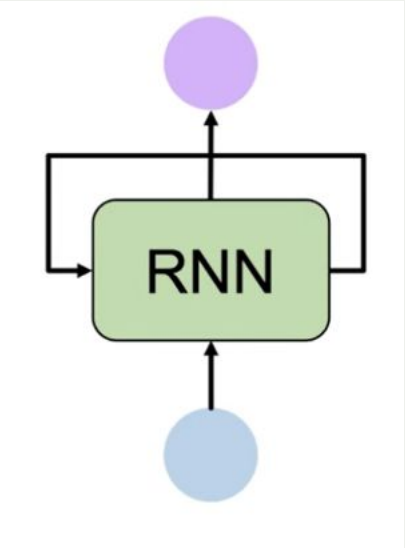
vs.

The food was bad, not good at all.

Sequence Modelling: Design Criteria

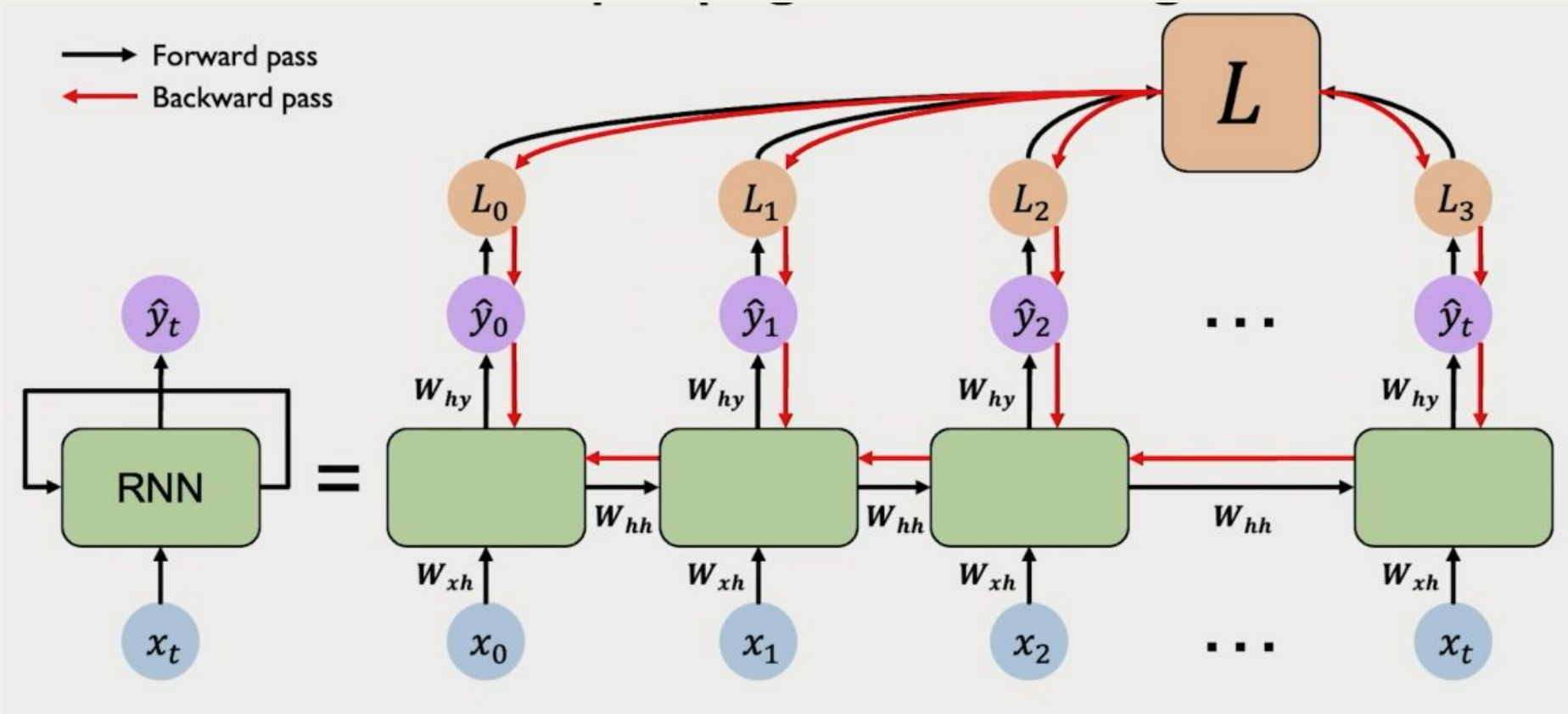
To model sequences, we need to:

- **Handle variable-length sequences**
- **Track long-term dependencies**
- **Maintain information about order**
- **Share parameters** across the sequence

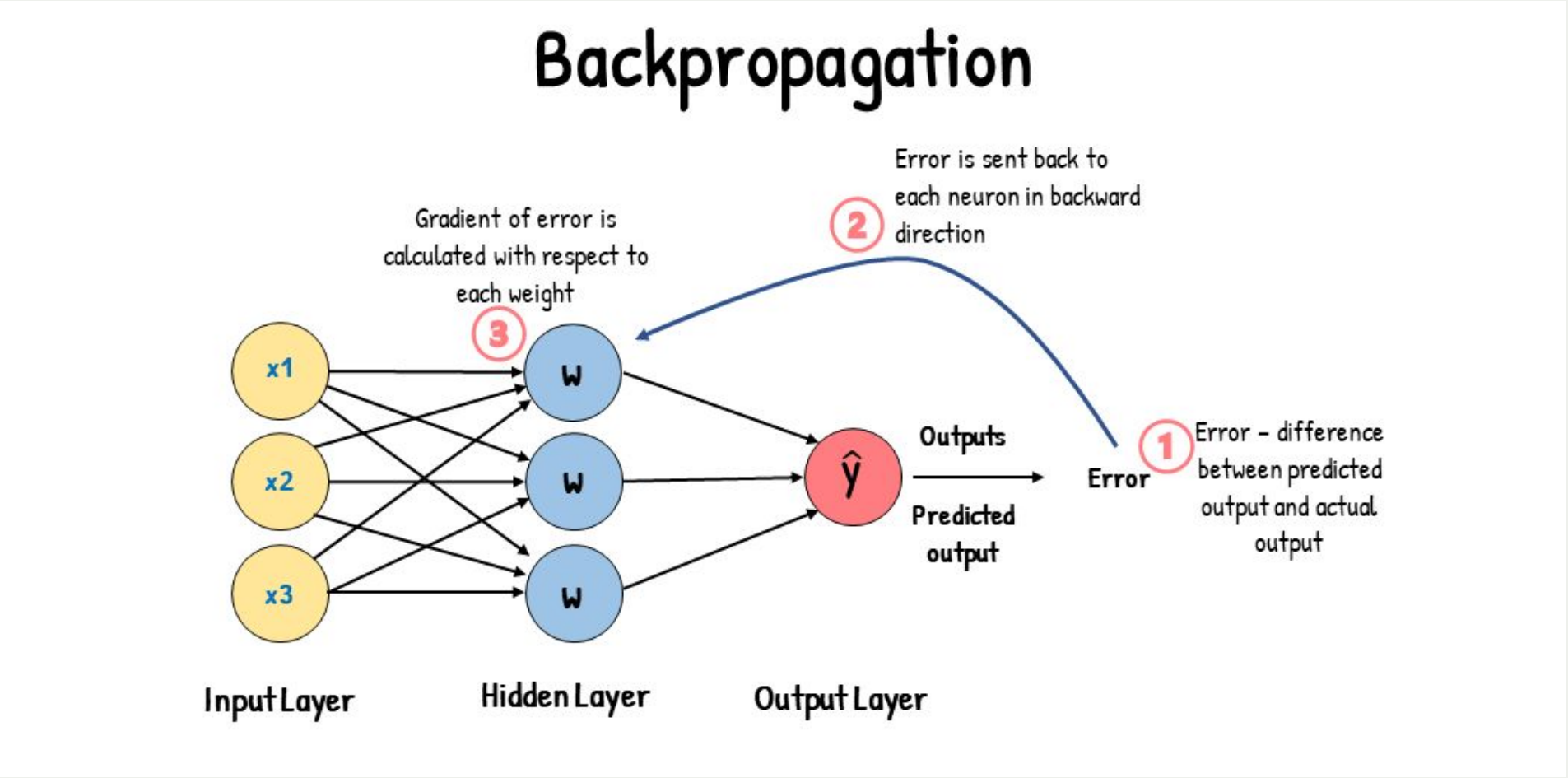


Recurrent Neural Networks (RNNs) meet these sequence modelling design criteria.

Recall: Backpropagation in Feed Forward



Recall: Backpropagation in Feed Forward



Problems with RNNs

Problem

- Vanishing Gradient
- Exploding Gradient
- Difficulty with Long-Term Memory
- Slow Training
- Limited Parallelization

Q and A