Deep Sequence Modeling

Ava Amini

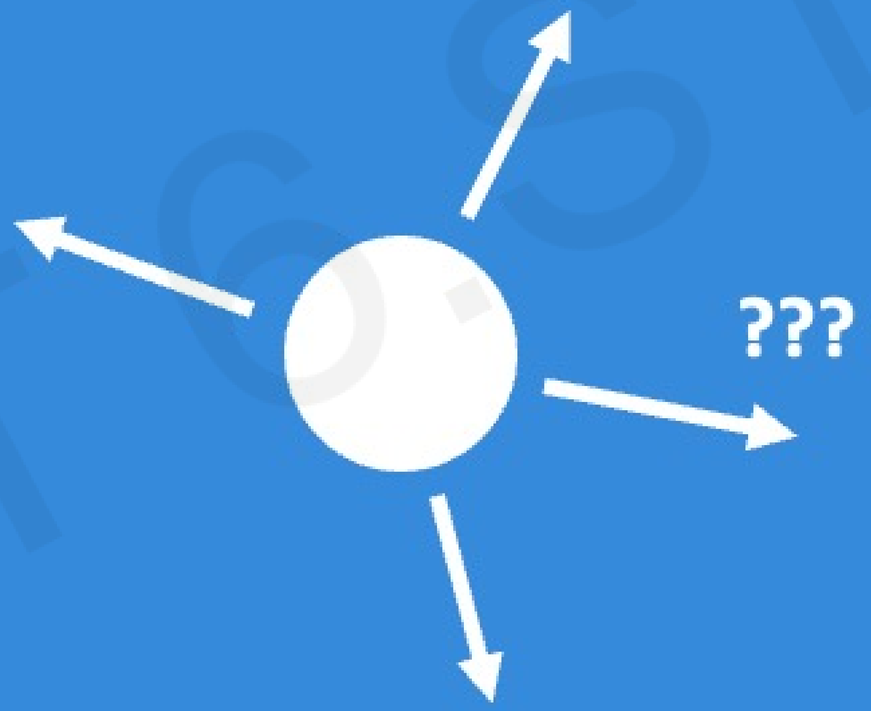
MIT Introduction to Deep Learning

January 6, 2025

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

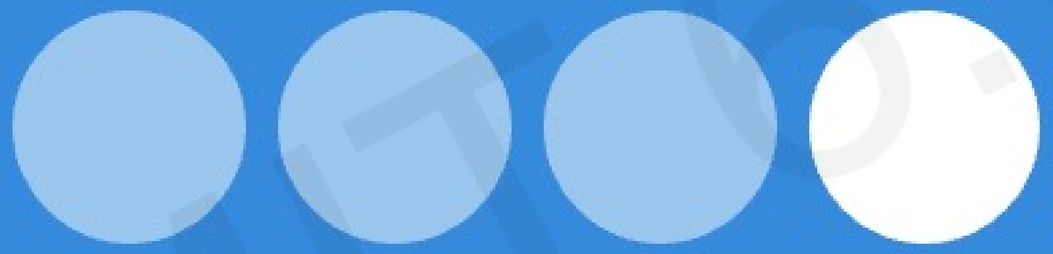
MIT 6.S191

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



MIT 6.S191

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



MIT 6.S191

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

MIT 6.S191

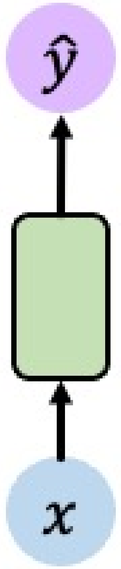
Sequences in the Wild

Audio

MIT 6.S191

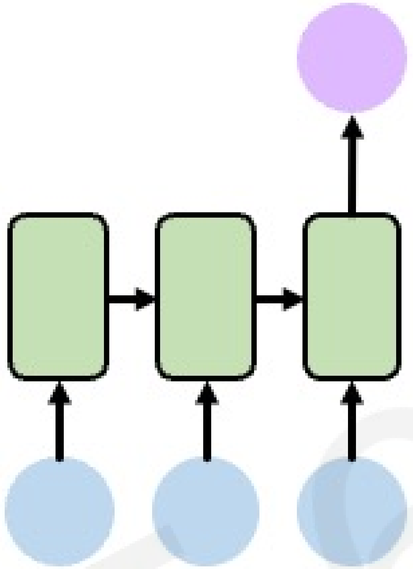
Sequences in the Wild

Sequence Modeling Applications

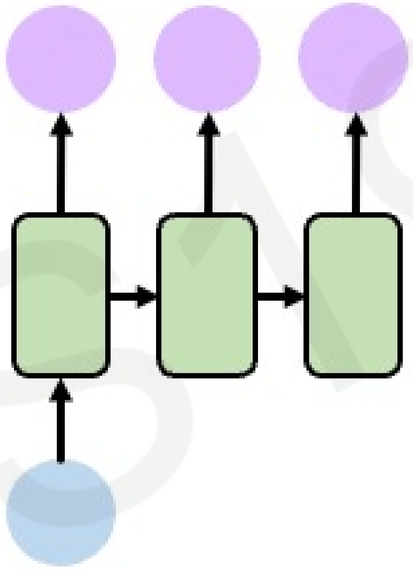


One to One
Binary Classification

"Will I pass this class?"
Student → Pass?

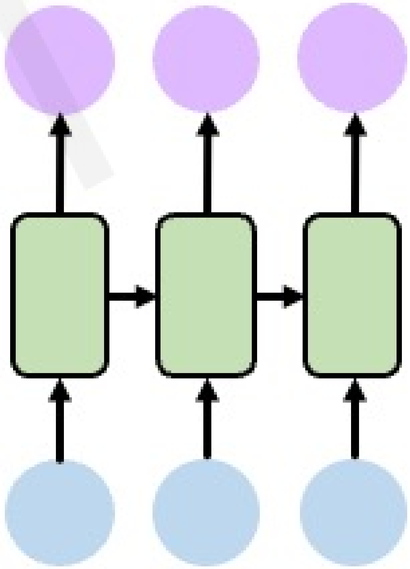


Many to One
Sentiment Classification



One to Many
Image Captioning

"A baseball player throws a ball."



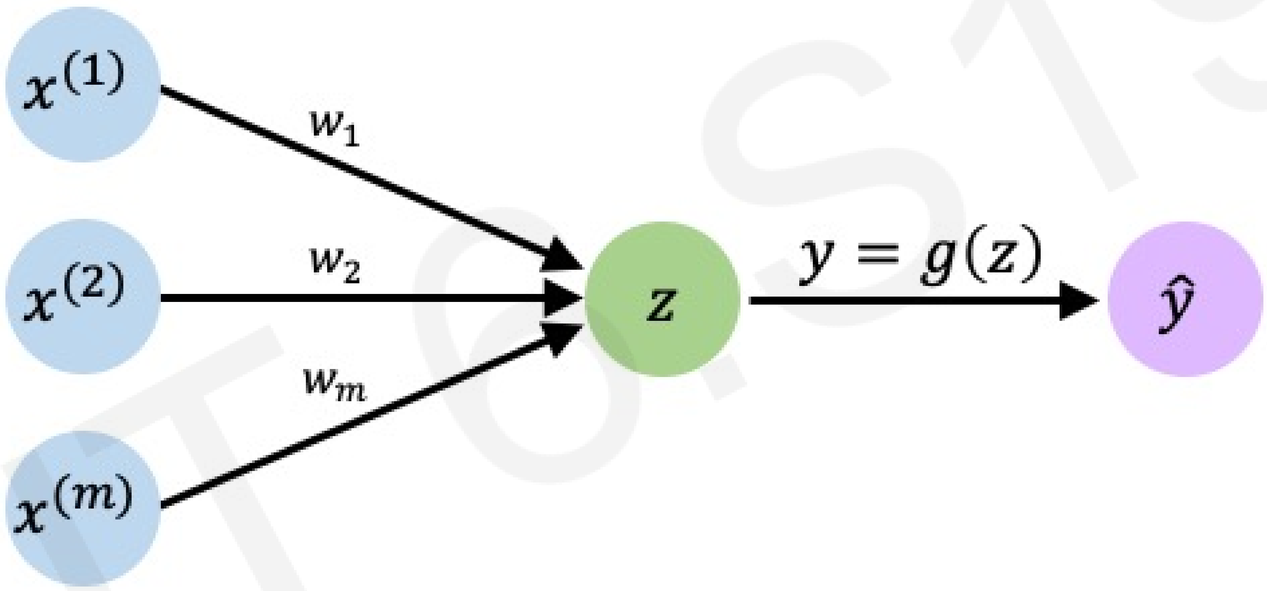
Many to Many
Machine Translation

MIT 6.S191

Neurons with Recurrence

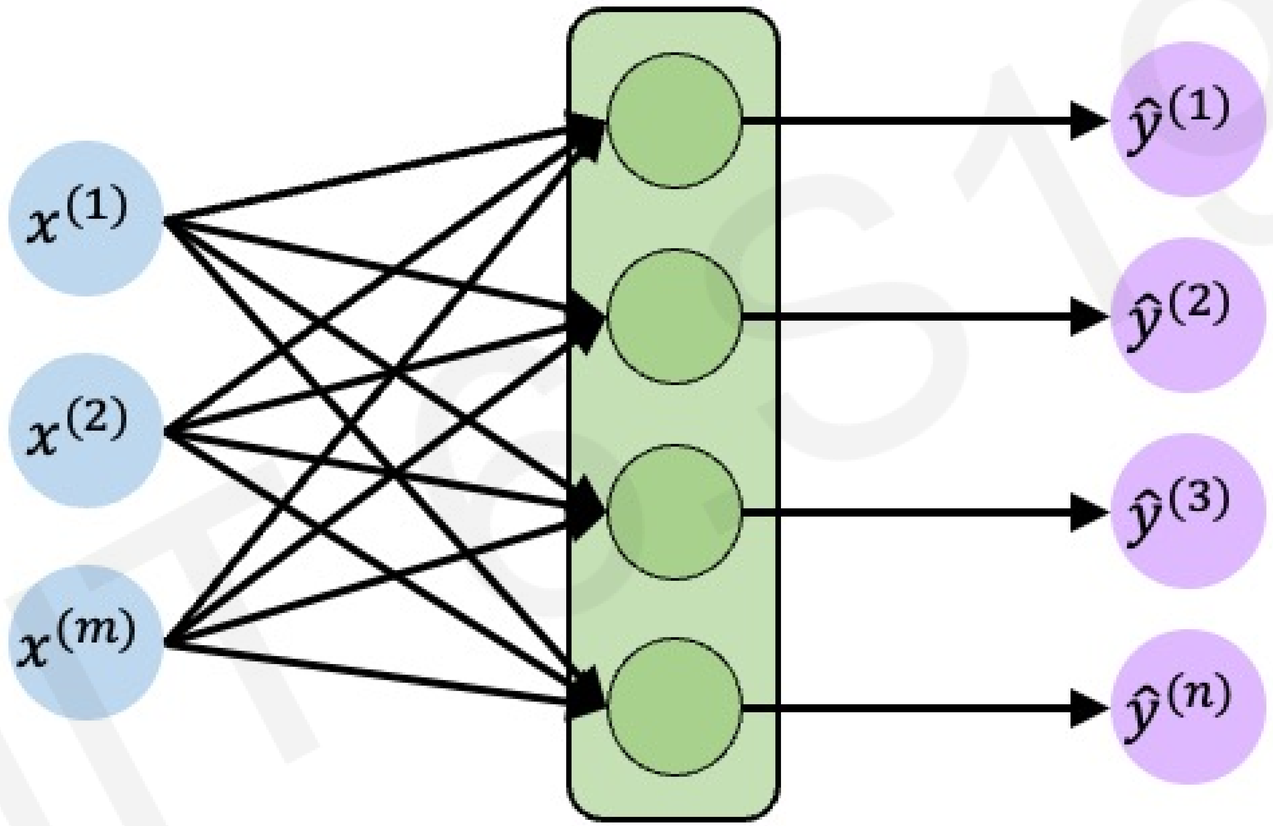
MIT 6.S191

The Perceptron Revisited



MIT 6.S191

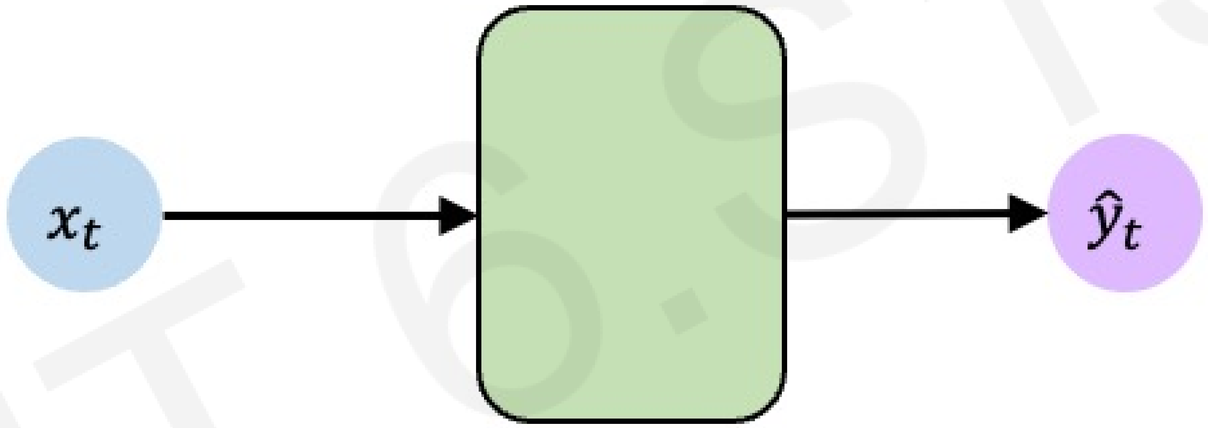
Feed-Forward Networks Revisited



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

MIT 6S191

Feed-Forward Networks Revisited

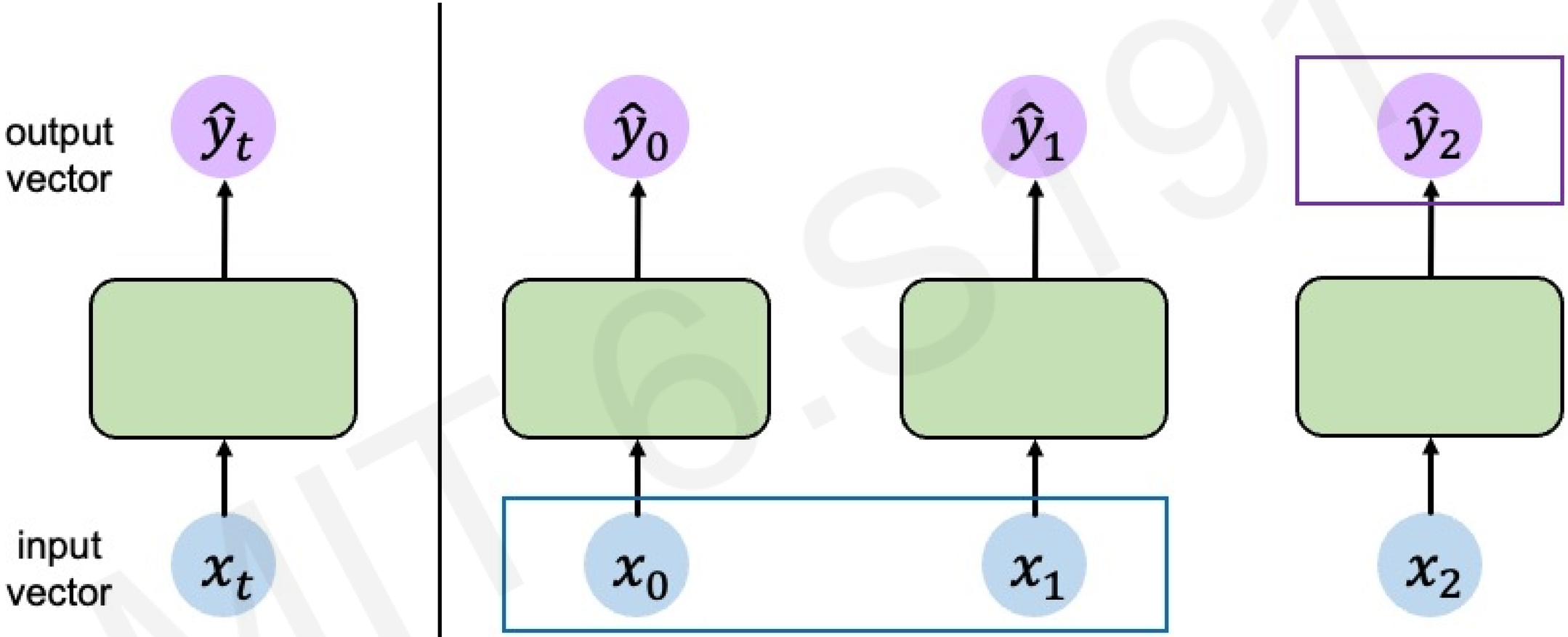


$\boldsymbol{x}_t \in \mathbb{R}^m$

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

MIT 6.S191

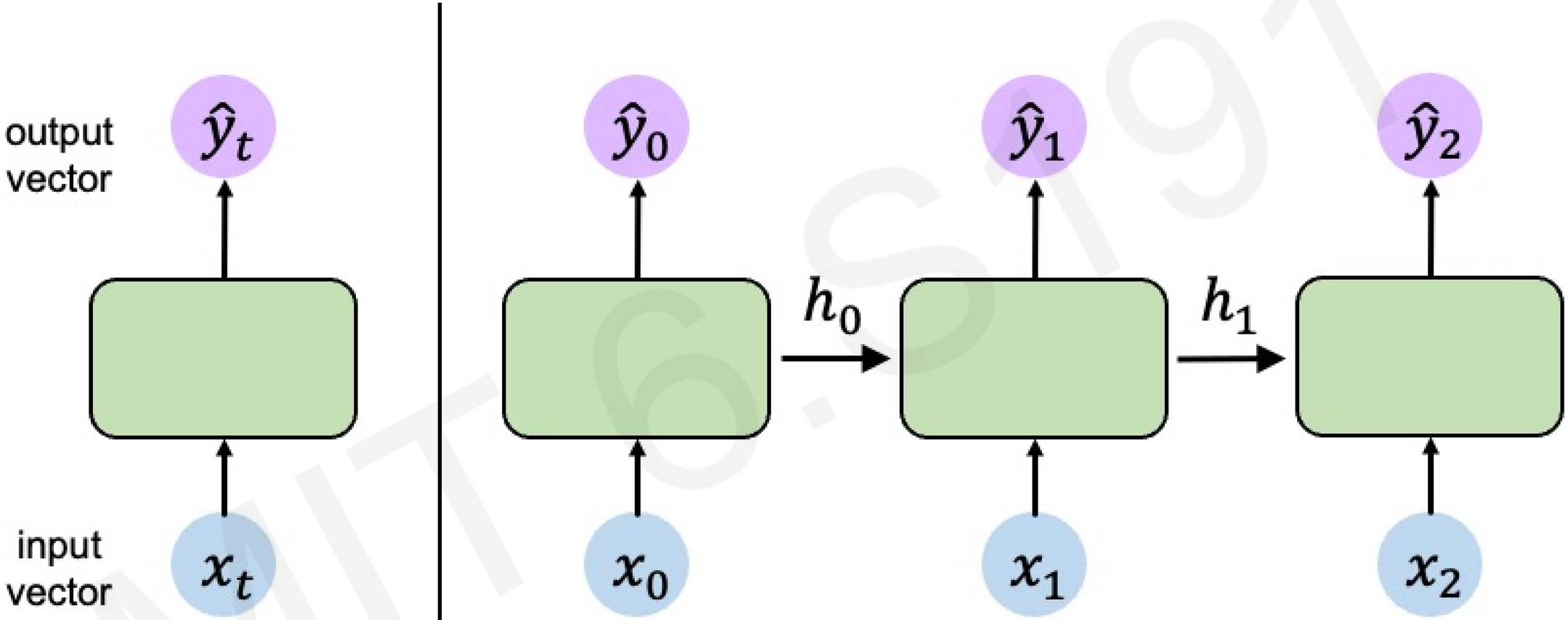
Handling Individual Time Steps



$$\hat{y}_t = f(x_t)$$

MIT 6.S191

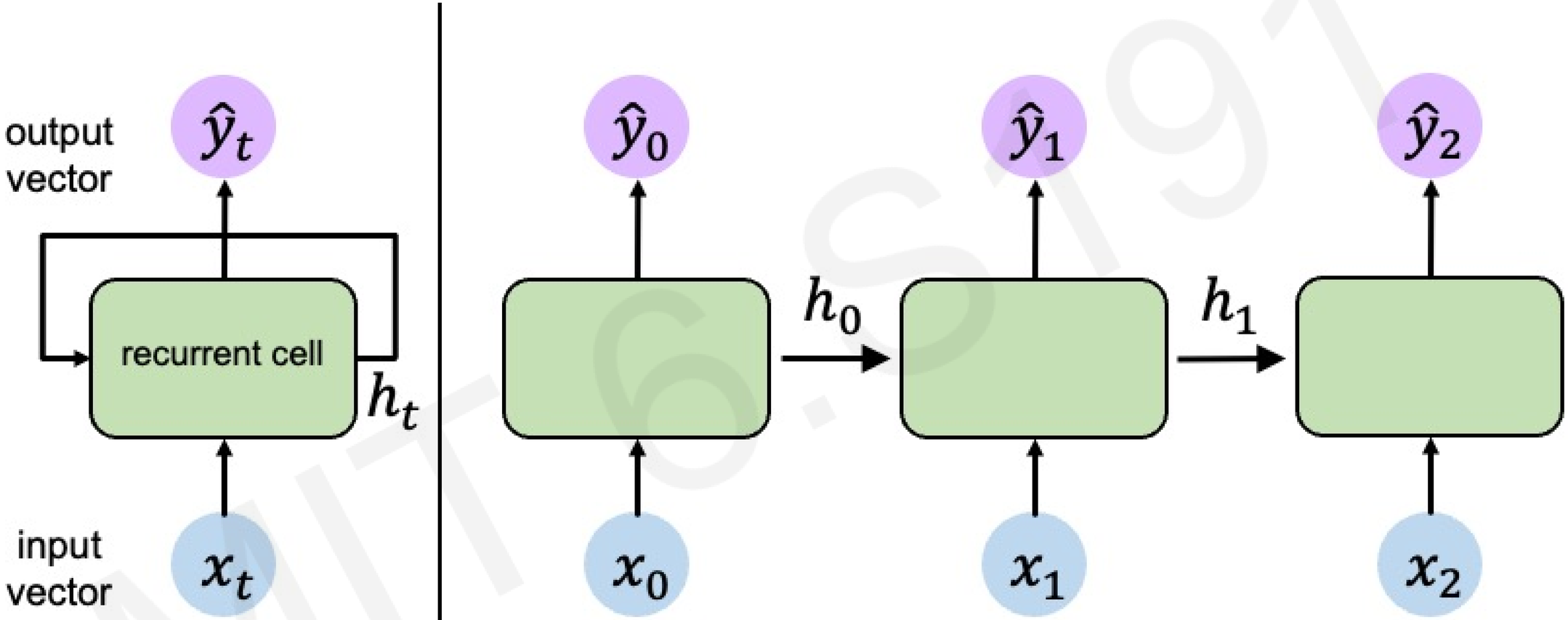
Neurons with Recurrence



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

MIT 6.S191

Neurons with Recurrence



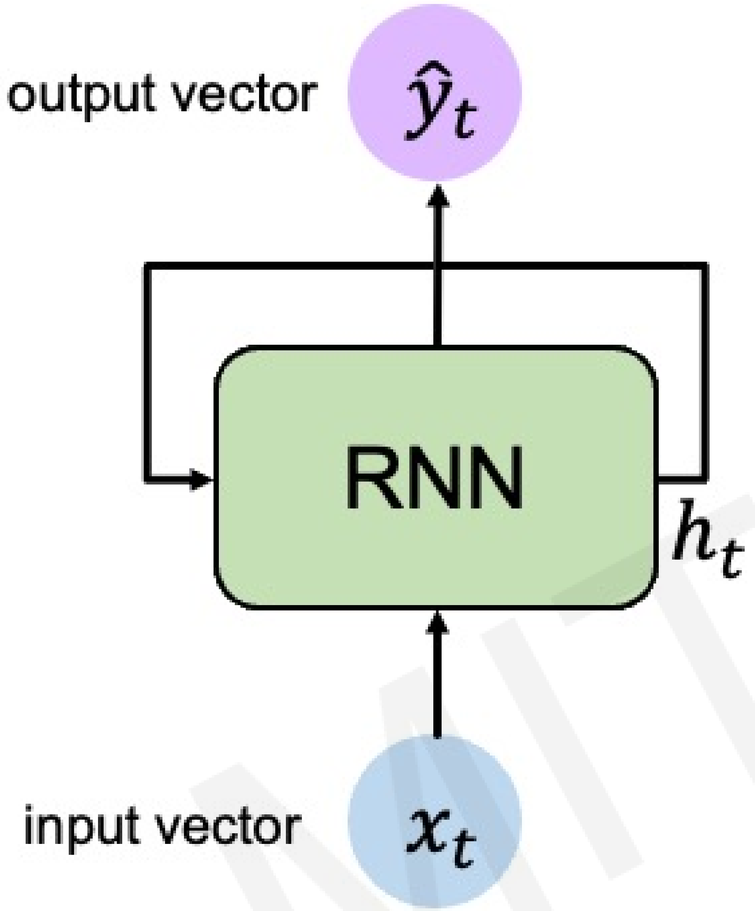
$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

MIT 6.S191

Recurrent Neural Networks (RNNs)

MIT 6.S191

Recurrent Neural Networks (RNNs)



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

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

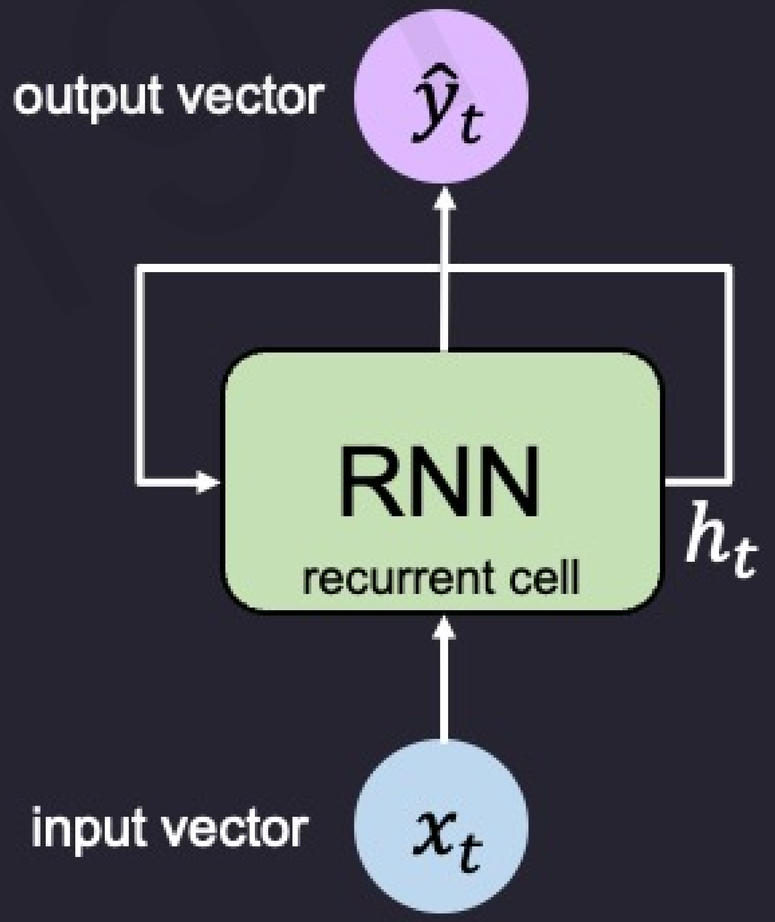
cell state — function with weights W — input — old state

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

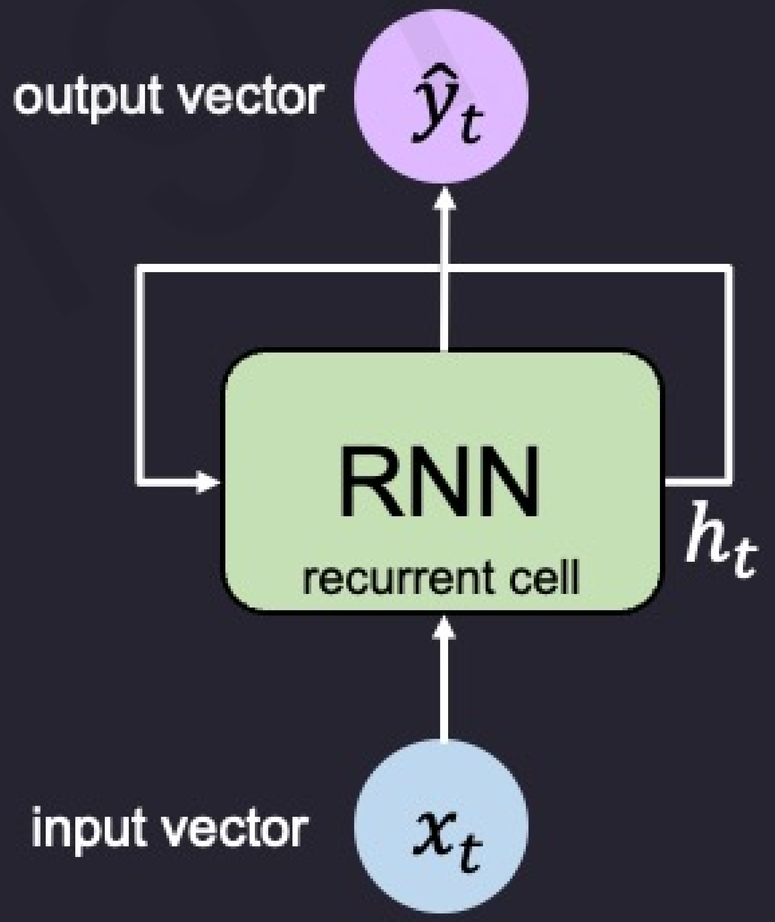
RNN Intuition

```
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!"
```



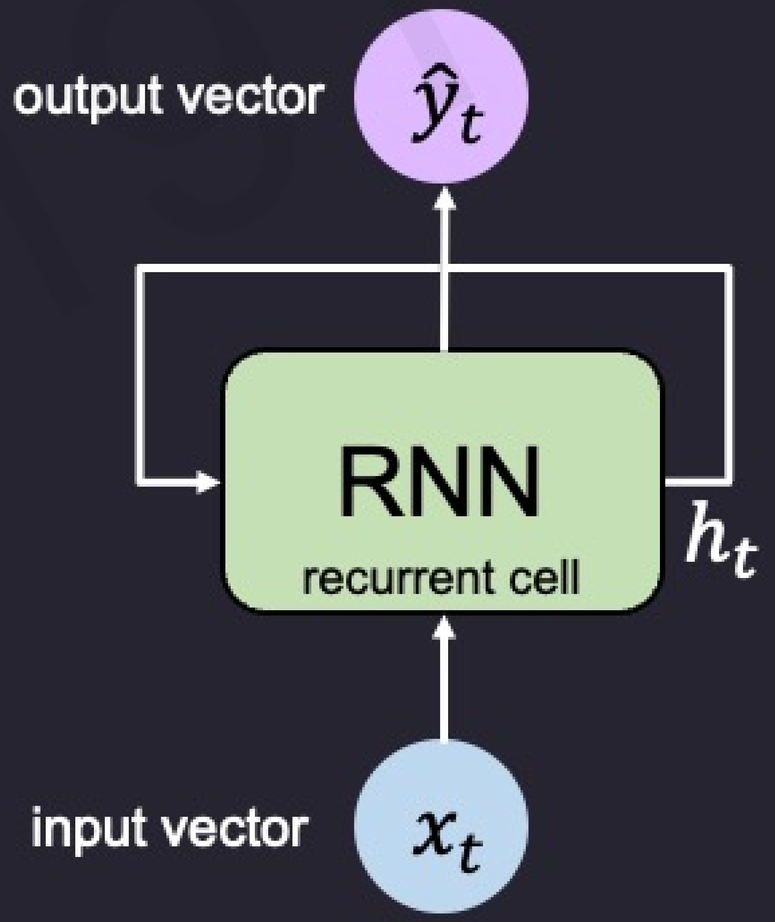
RNN Intuition

```
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!"
```

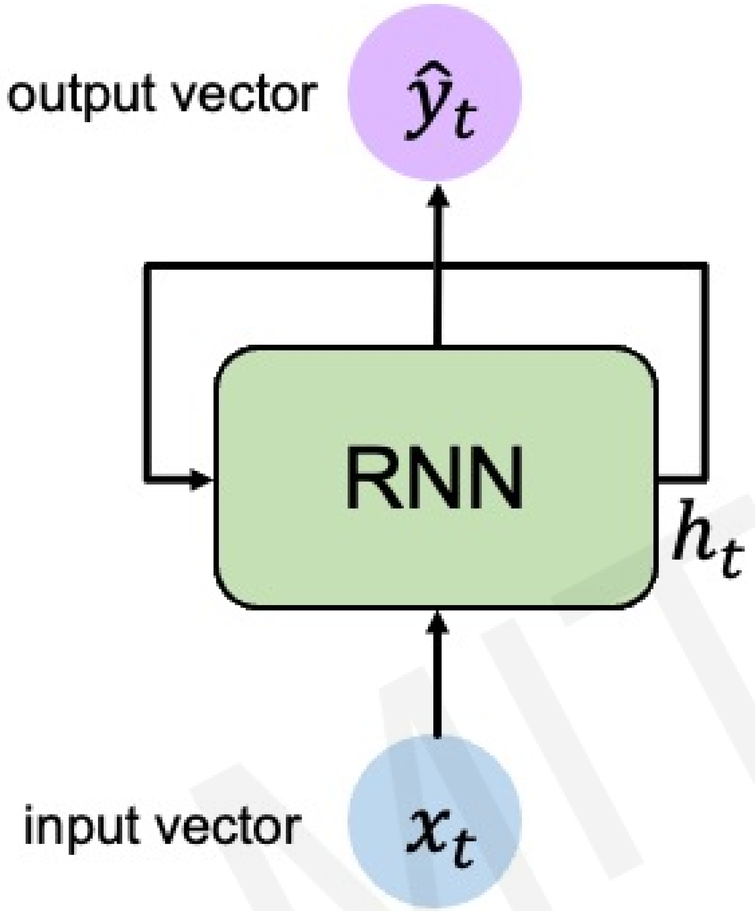


RNN Intuition

```
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!"
```

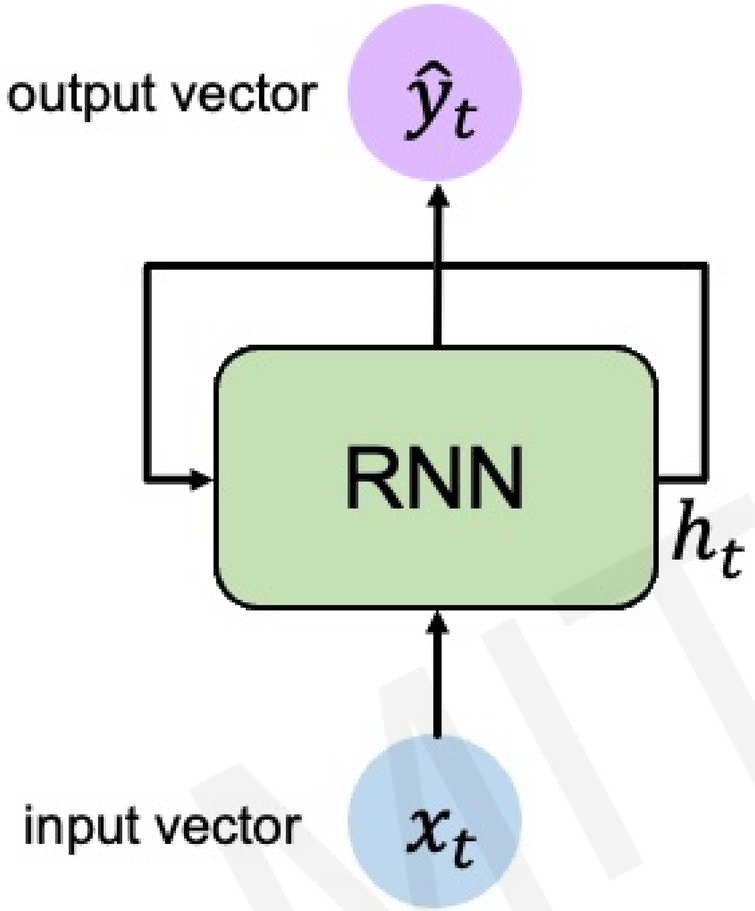


RNN State Update and Output



MIT 6.S191

RNN State Update and Output

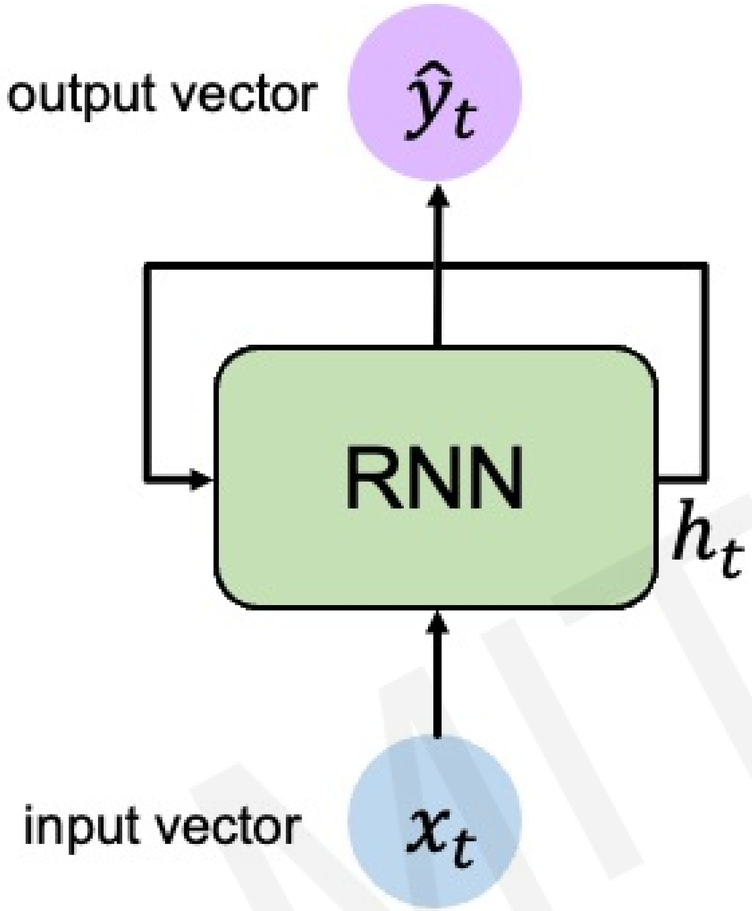


Input Vector

x_t

MIT 6.S191

RNN State Update and Output



Update Hidden State

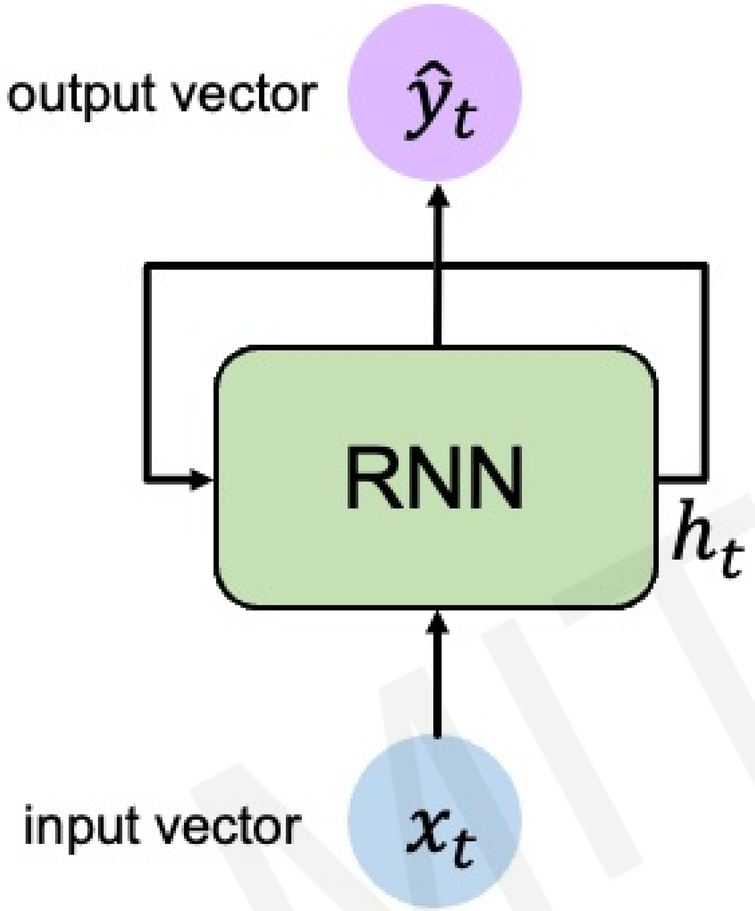
$$h_t = \tanh(\boldsymbol{W}_{hh}^T h_{t-1} + \boldsymbol{W}_{xh}^T x_t)$$

Input Vector

x_t

MIT 6.S191

RNN State Update and Output



Output Vector

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

Update Hidden State

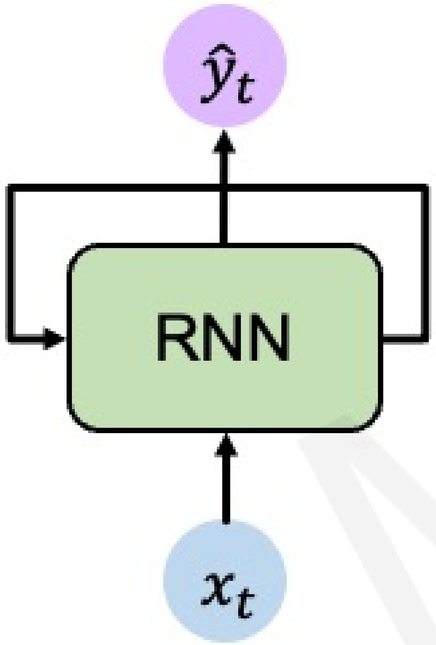
$$h_t = \tanh(\boldsymbol{W}_{hh}^{T} h_{t-1} + \boldsymbol{W}_{xh}^{T} x_t)$$

Input Vector

$$x_t$$

MIT 6.S191

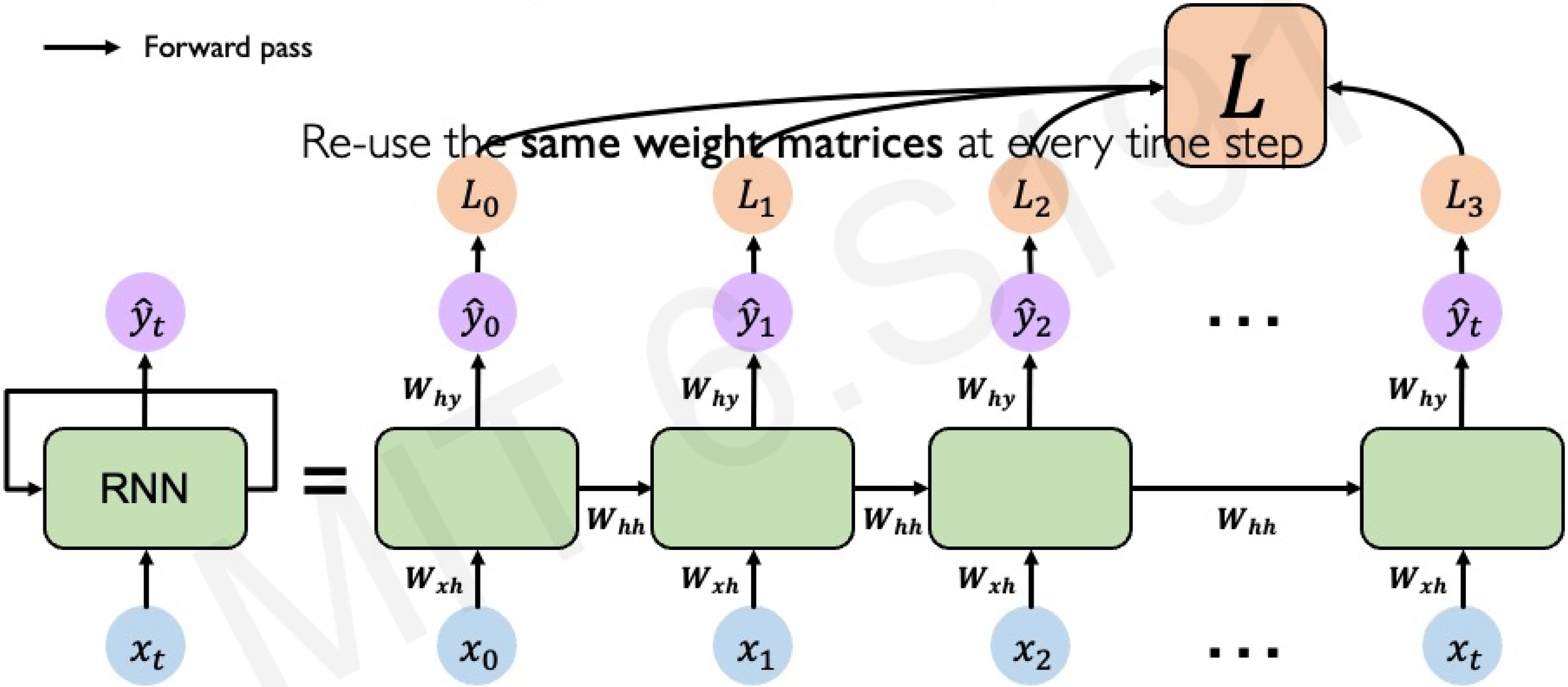
RNNs: Computational Graph Across Time



= Represent as computational graph unrolled across time

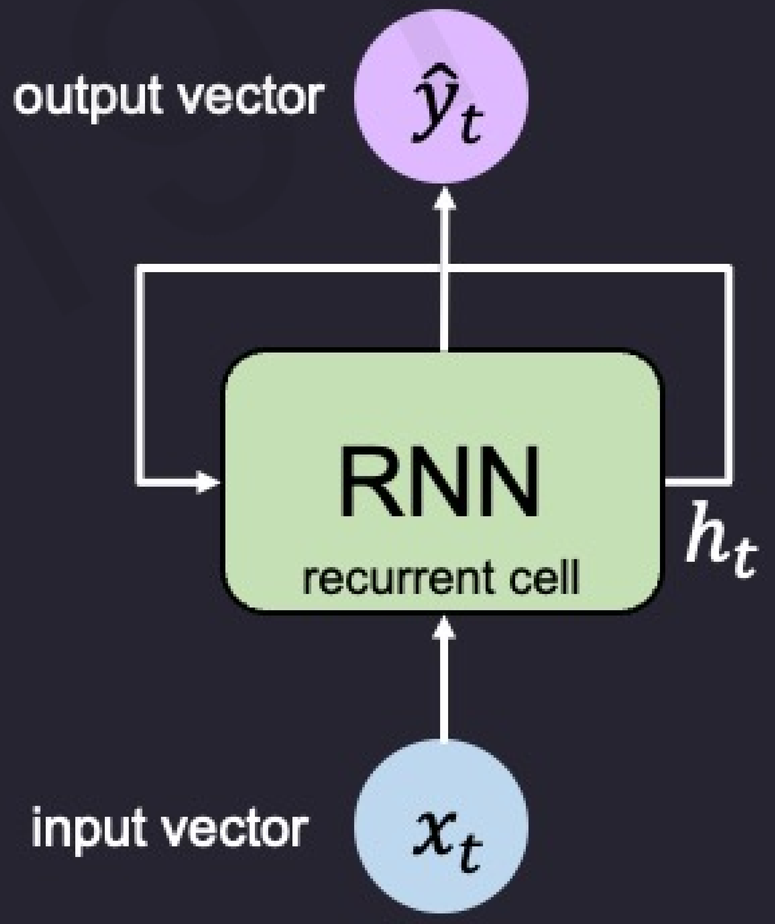
MIT 6.S191

RNNs: Computational Graph Across Time



RNNs from Scratch in TensorFlow

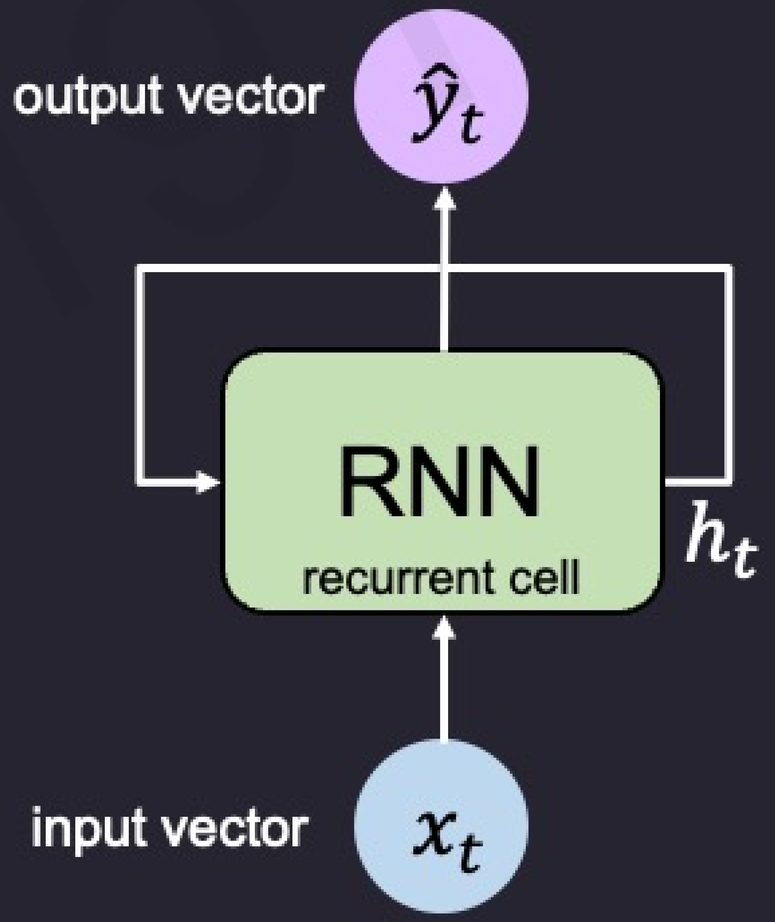
```python
class MyRNNCell(tf.keras.layers.Layer):
  def __init__(self, rnn_units, input_dim, output_dim):
    super(MyRNNCell, self).__init__()

    # Initialize weight matrices
    self.W_xh = self.add_weight([rnn_units, input_dim])
    self.W_hh = self.add_weight([rnn_units, rnn_units])
    self.W_hy = self.add_weight([output_dim, rnn_units])

    # Initialize hidden state to zeros
    self.h = tf.zeros([rnn_units, 1])

  def call(self, x):
    # Update the hidden state
    self.h = tf.math.tanh( self.W_hh * self.h + self.W_xh * x )

    # Compute the output
    output = self.W_hy * self.h

    # Return the current output and hidden state
    return output, self.h
```



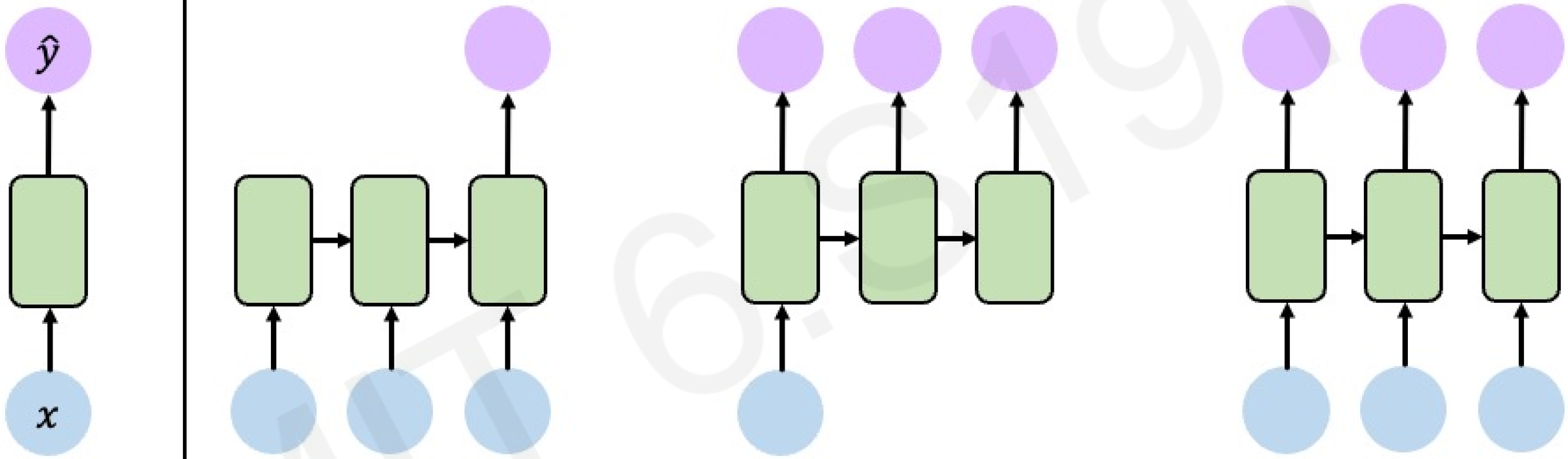
RNN Implementation: TensorFlow & PyTorch

```
from tf.keras.layers import SimpleRNN
    model = SimpleRNN(rnn_units)
```

```
from torch.nn import RNN
model = RNN(input_size, rnn_units)
```



RNNs for Sequence Modeling



One to One
"Vanilla" NN
Binary classification

Many to One
Sentiment Classification

One to Many
Text Generation
Image Captioning

Many to Many
Translation & Forecasting
Music Generation

… and many other architectures and applications

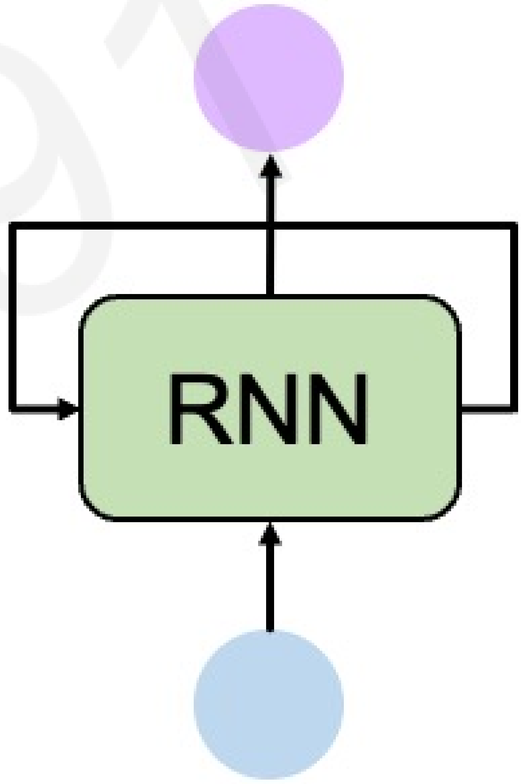
6.S191 Lab!

MIT 6.S191

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

MIT 6.S191

A Sequence Modeling Problem: Predict the Next Word

MIT 6.S191

A Sequence Modeling Problem: Predict the Next Word

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

MIT 6.S191

A Sequence Modeling Problem: Predict the Next Word

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

given these words

MIT 6.S191

A Sequence Modeling Problem: Predict the Next Word

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

given these words

predict the next word

MIT 6.S191

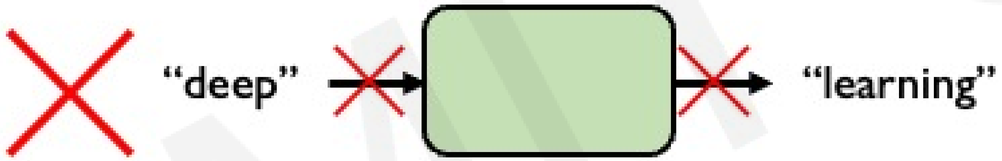
A Sequence Modeling Problem: Predict the Next Word

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

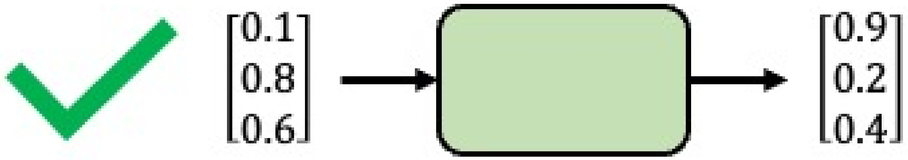
given these words

predict the next word

Representing Language to a Neural Network



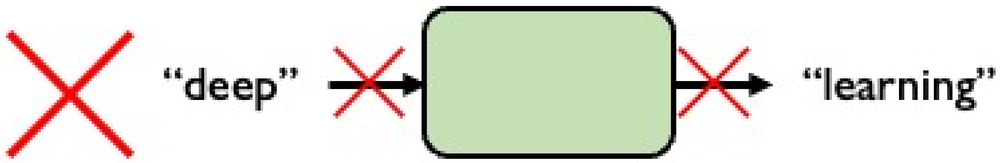
Neural networks cannot interpret words



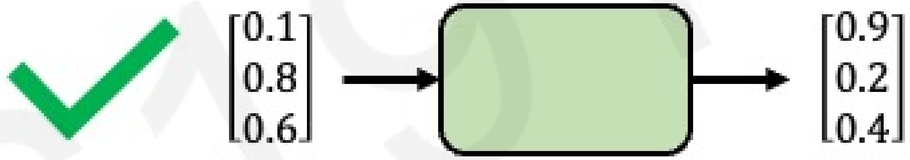
Neural networks require numerical inputs

MIT 6.S191

Encoding Language for a Neural Network

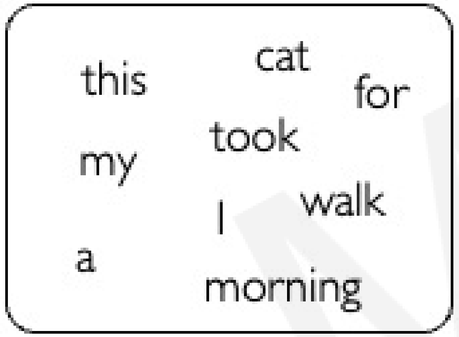


Neural networks cannot interpret words

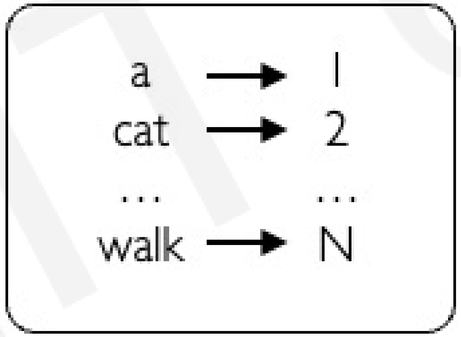


Neural networks require numerical inputs

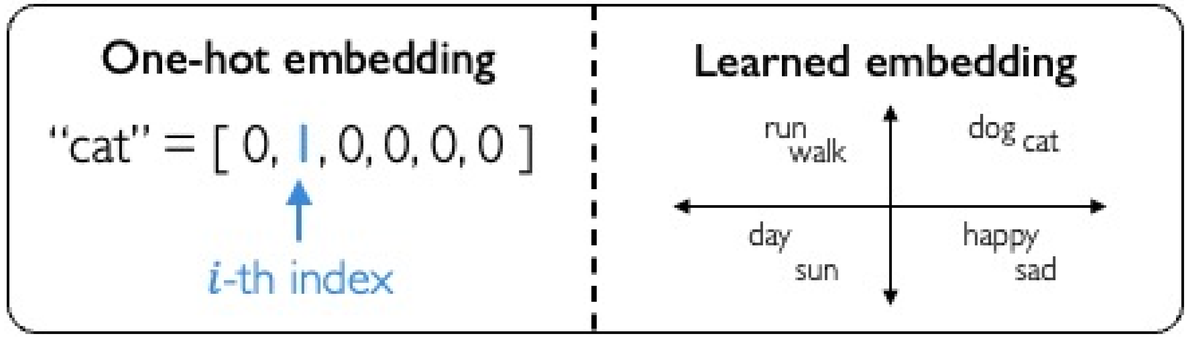
Embedding: transform indexes into a vector of fixed size.



1. Vocabulary:
Corpus of words



2. Indexing:
Word to index



3. Embedding:
Index to fixed-sized vector

MIT 6.S191

Handle Variable Sequence Lengths

The food was <u>great</u>

vs.

We visited a restaurant for <u>lunch</u>

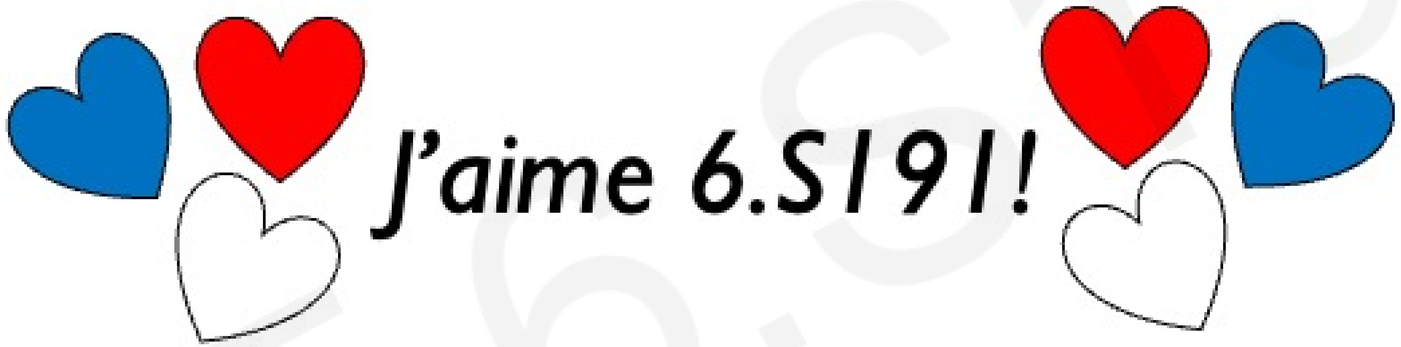
vs.

We were hungry but cleaned the house before <u>eating</u>

MIT 6.S191

Model 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.

MIT 6.S191

Capture Differences in Sequence Order

The food was good, not bad at all.

vs.

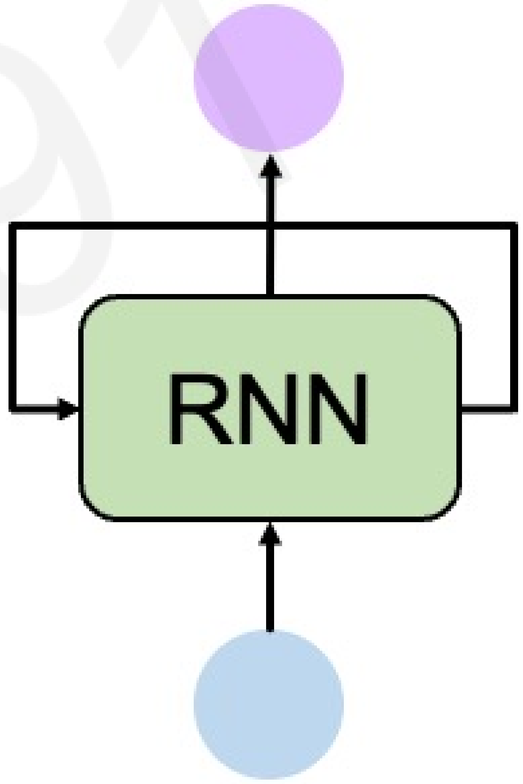
The food was bad, not good at all.

MIT 6.S191

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



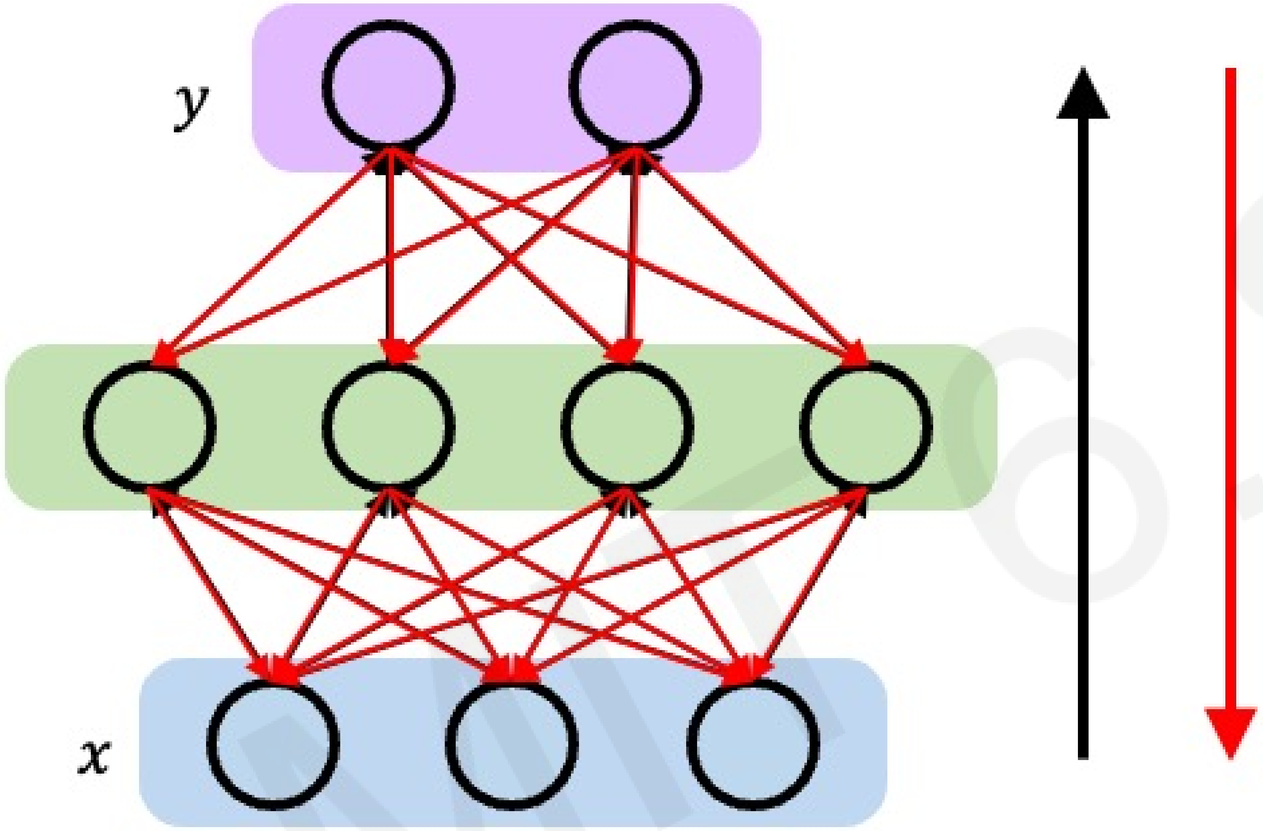
Recurrent Neural Networks (RNNs) meet these sequence modeling design criteria

MIT 6.S191

Backpropagation Through Time (BPTT)

MIT 6.S191

Recall: Backpropagation in Feed Forward Models

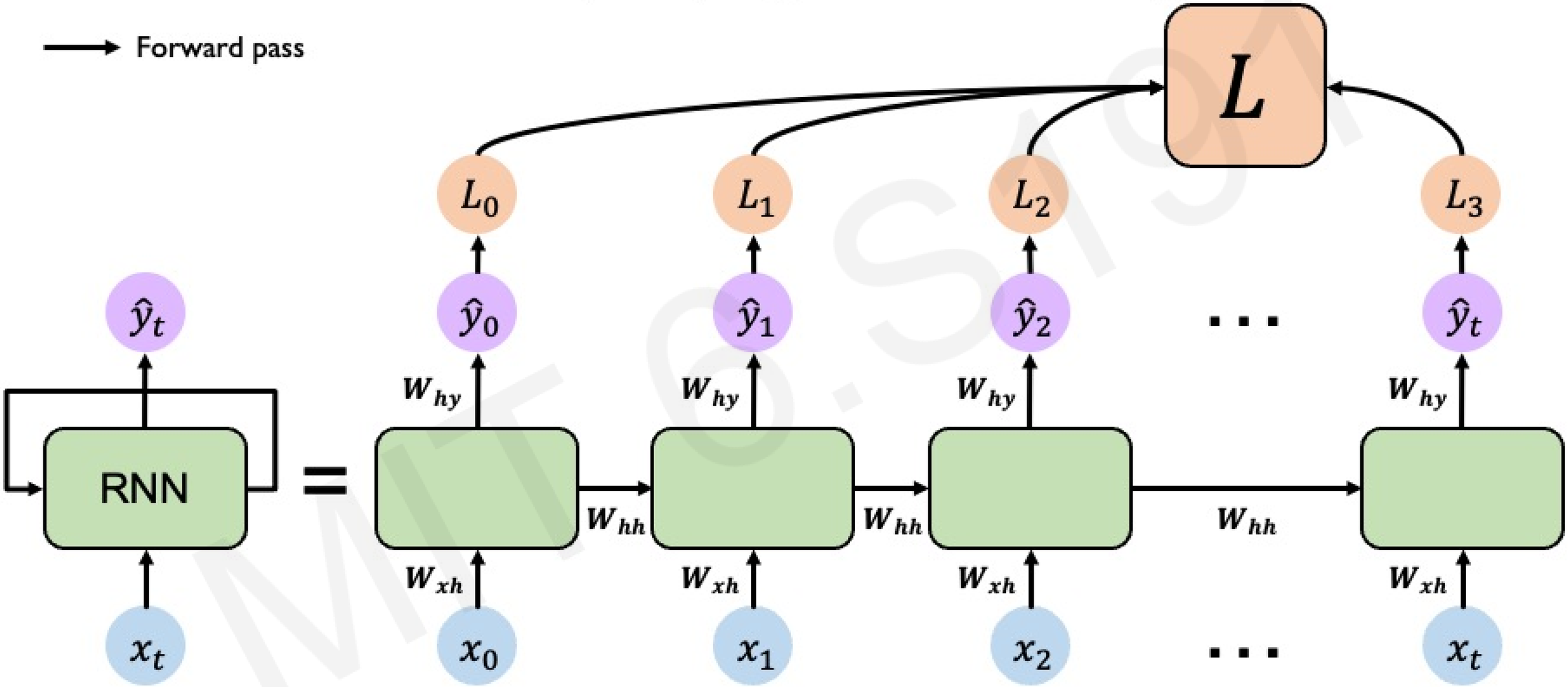


Backpropagation algorithm:

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

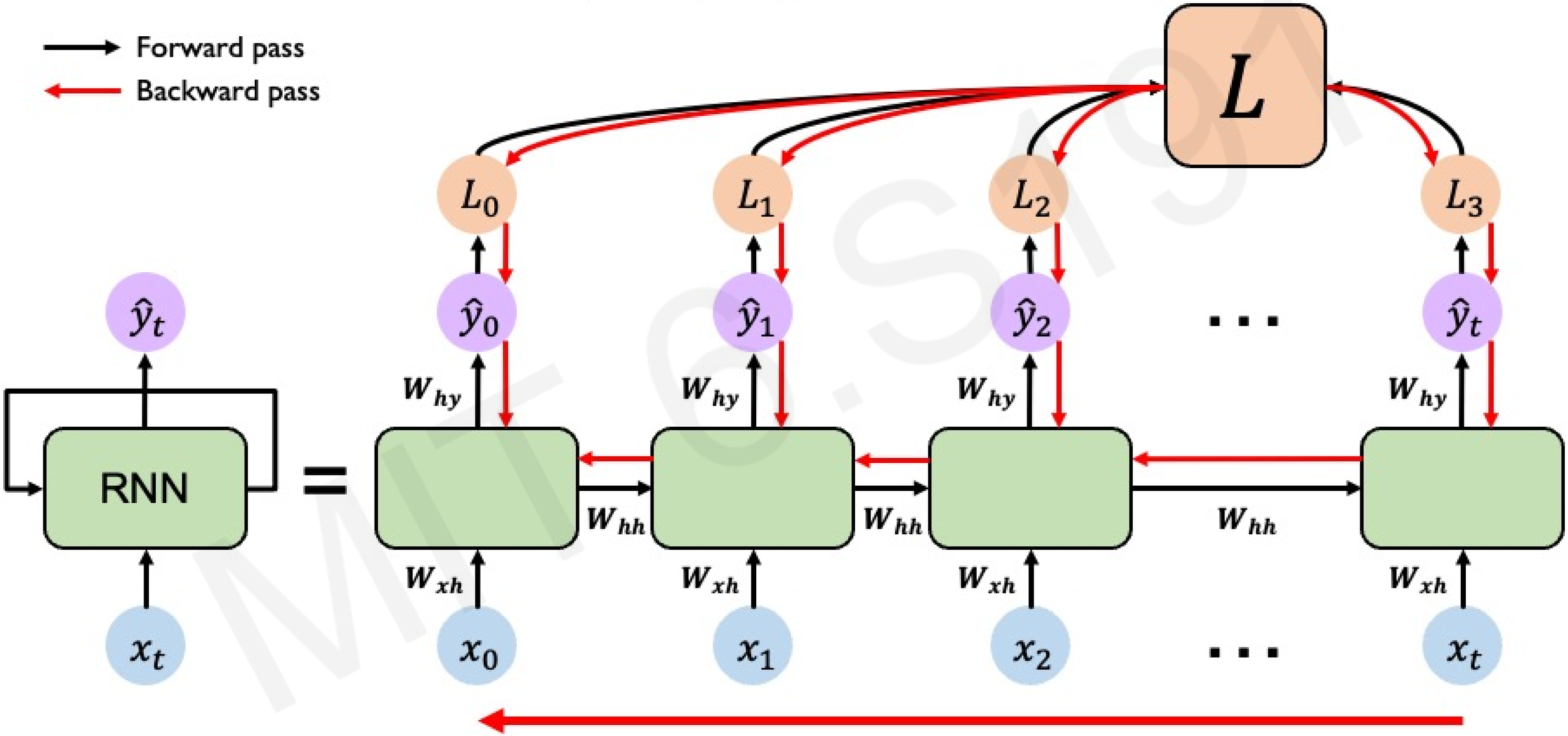
MIT 6.S191

RNNs: Backpropagation Through Time



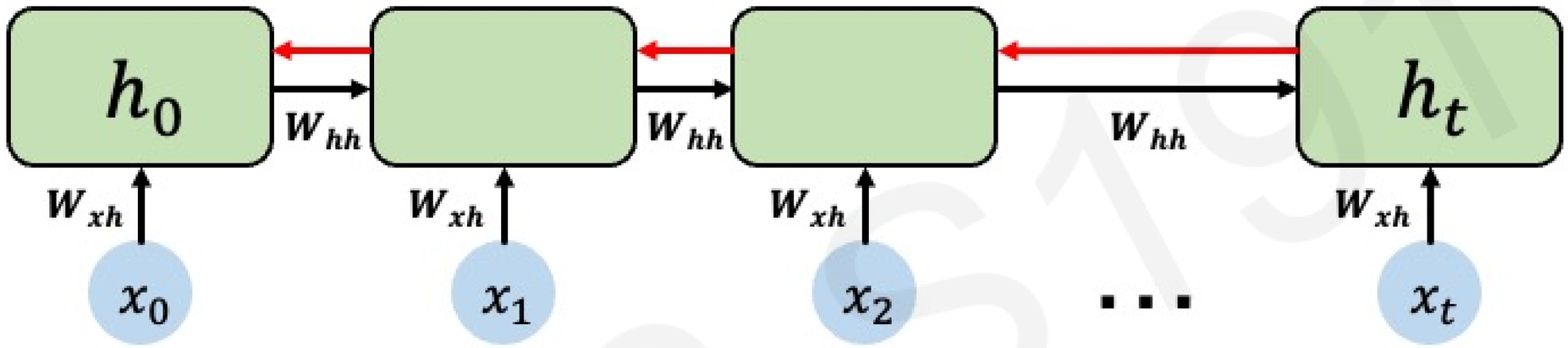
MIT 6.S191

RNNs: Backpropagation Through Time



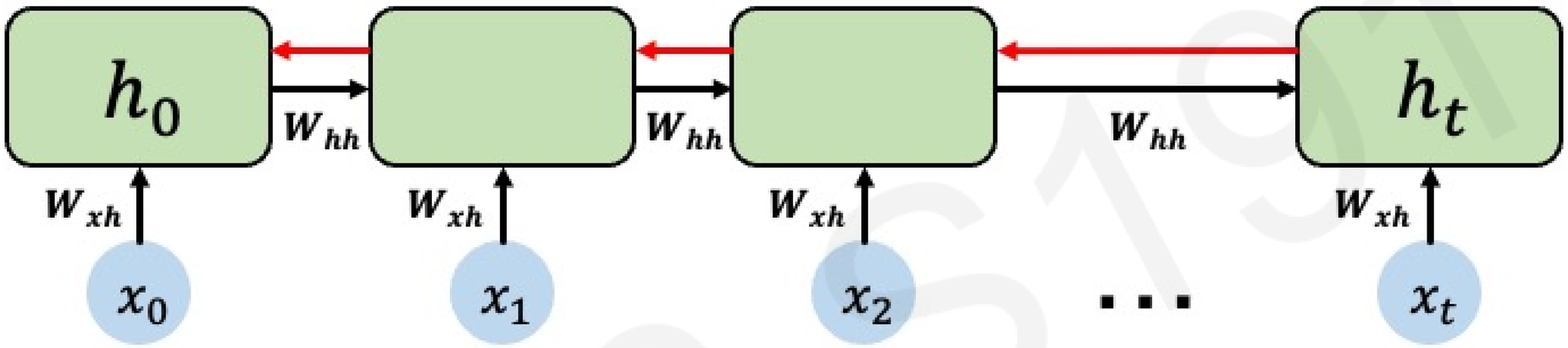
MIT.S191

Standard RNN Gradient Flow



MIT 6.S191

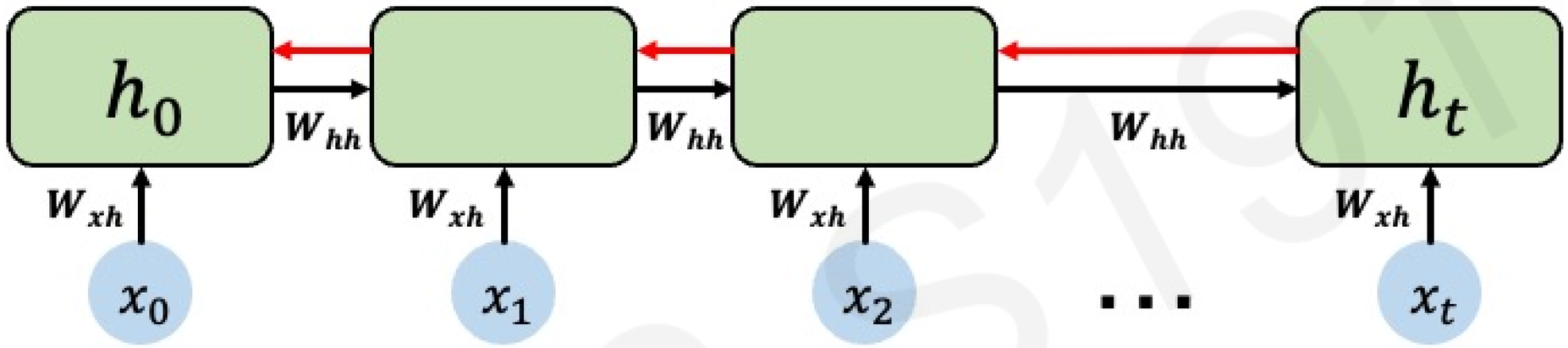
Standard RNN Gradient Flow



Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation**!

MIT 6.S191

Standard RNN Gradient Flow: Exploding Gradients



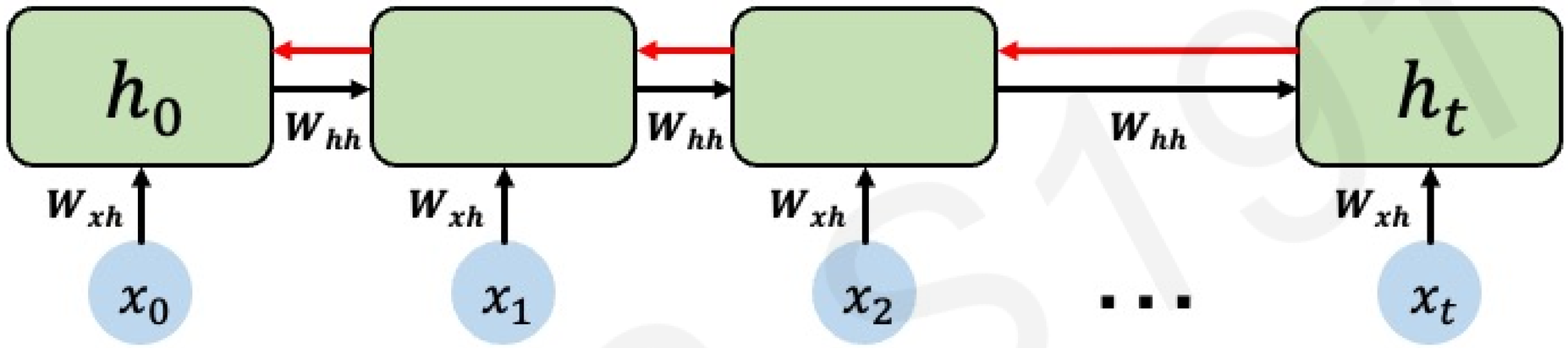
Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation**!

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

MIT 6.S191

Standard RNN Gradient Flow: Vanishing Gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation**!

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Many values < 1:
vanishing gradients

1. Activation function
2. Weight initialization
3. Network architecture

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

MIT 6.S191

The Problem of Long-Term Dependencies

"The clouds are in the ___"

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

MIT 6.S191

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

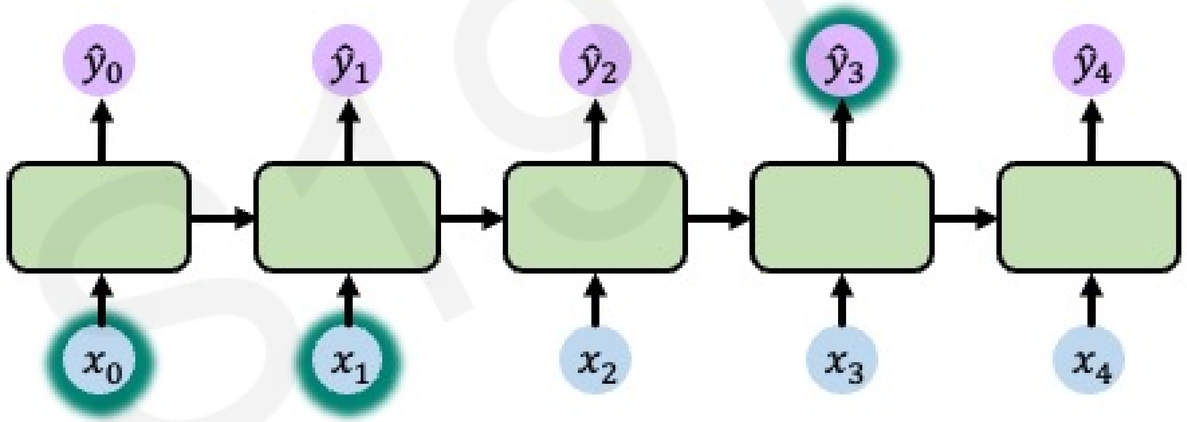
↓

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

↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



MIT 6.S191

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

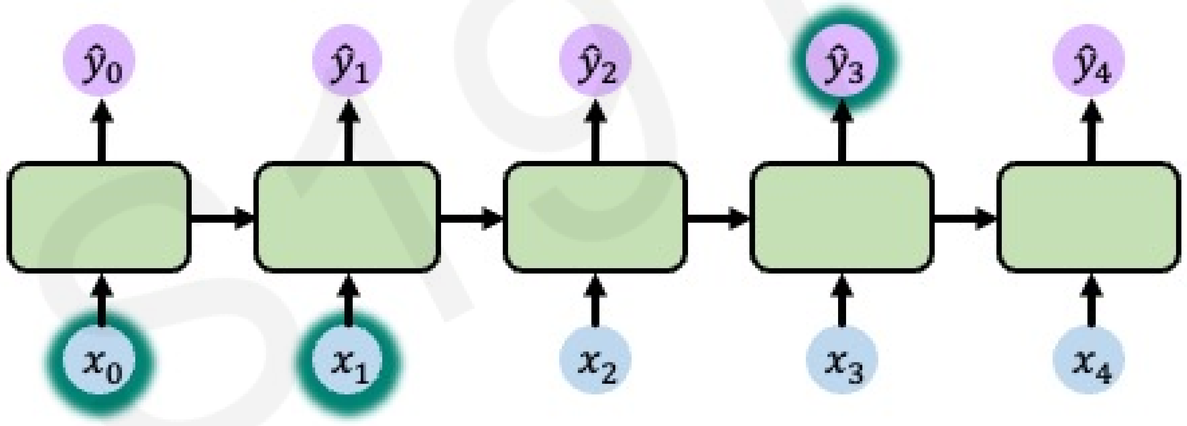
↓

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___ "

MIT 6.S191

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

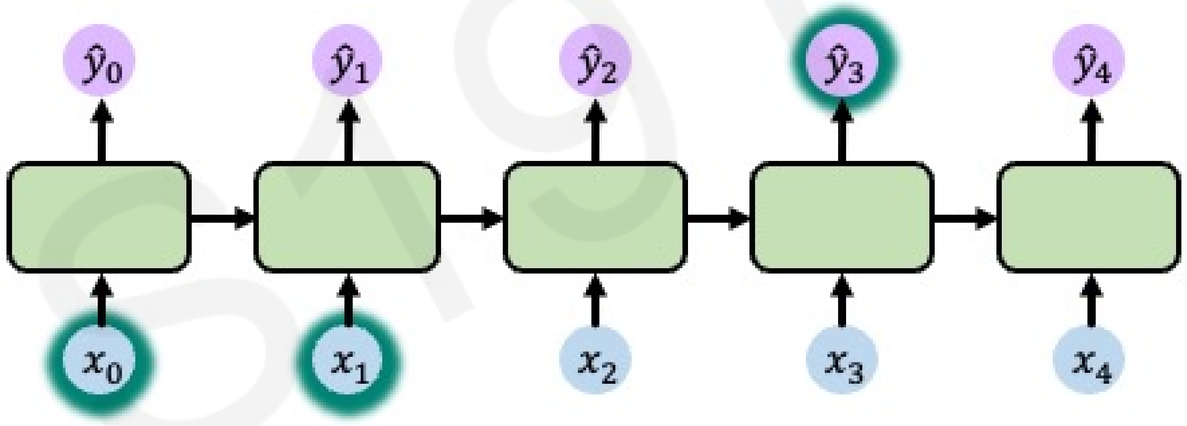
↓

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

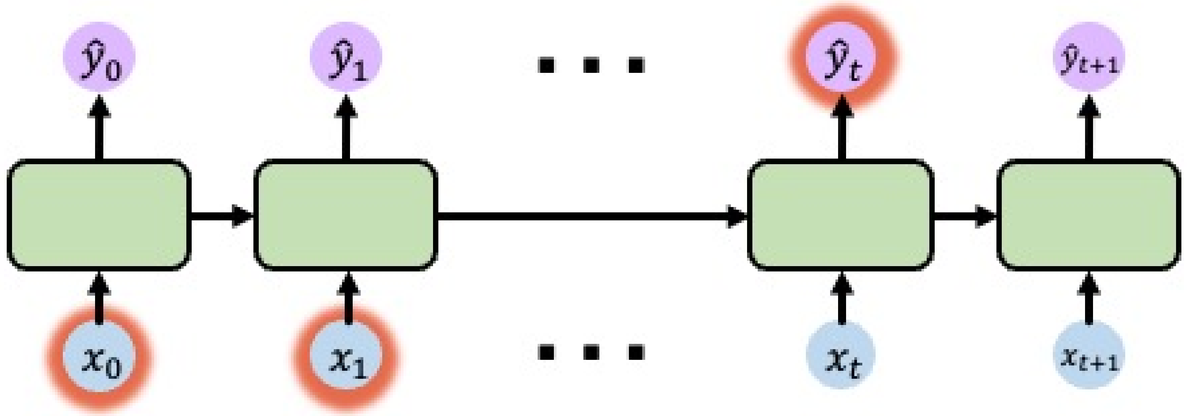
↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



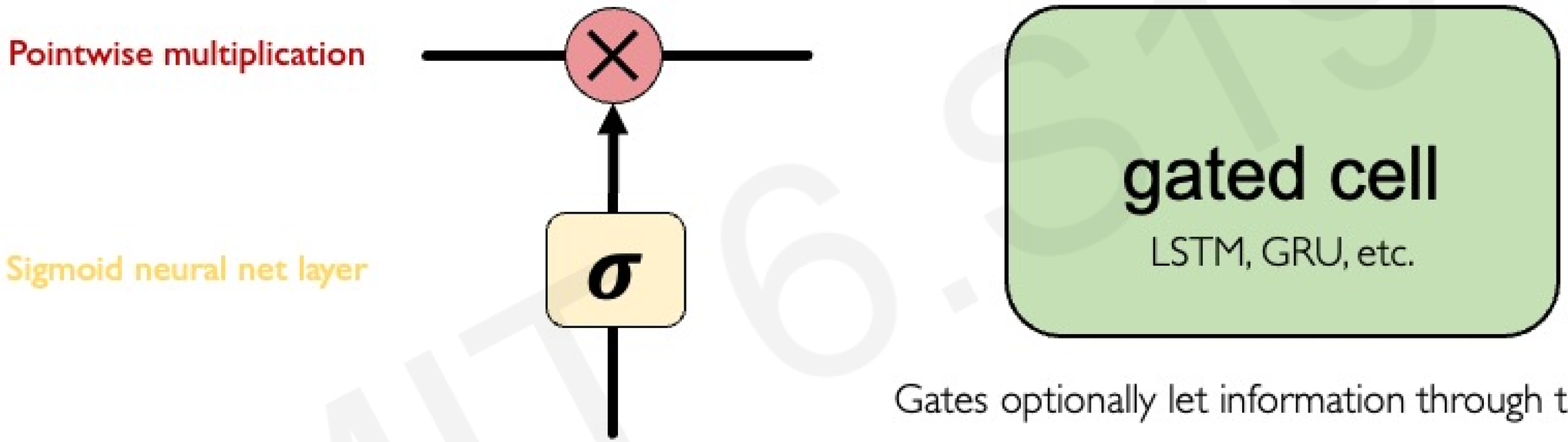
"I grew up in France, … and I speak fluent___ "



MIT 6.S191

Gating Mechanisms in Neurons

Idea: use **gates** to selectively **add** or **remove** information within **each recurrent unit with**



Gates optionally let information through the cell

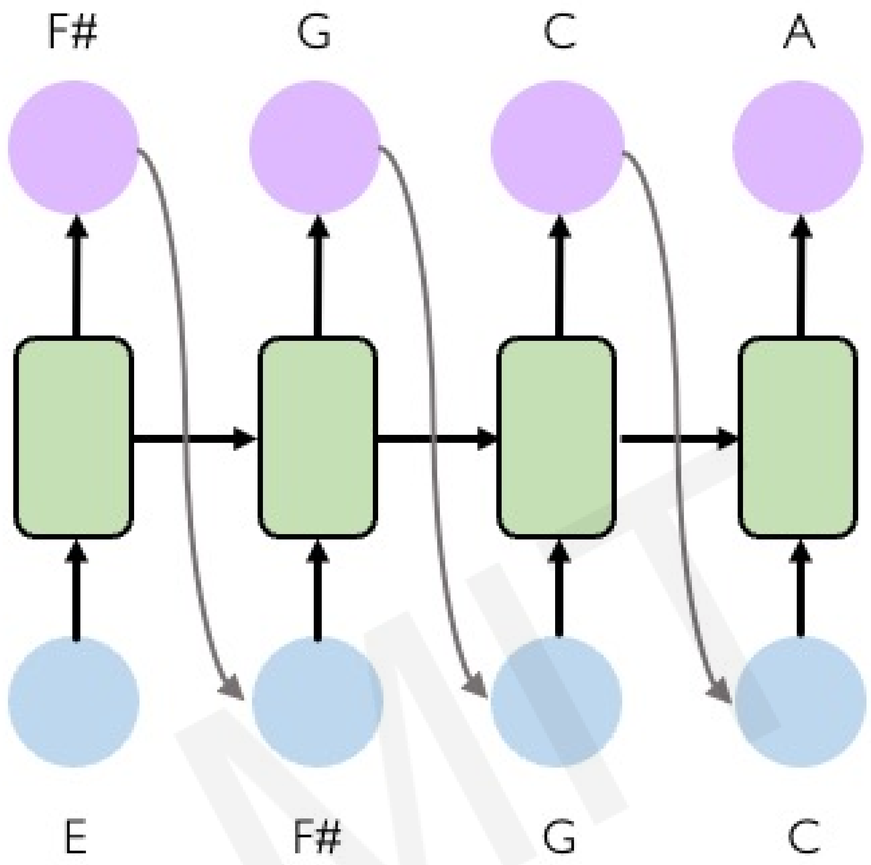
Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

MIT 6.S191

RNN Applications & Limitations

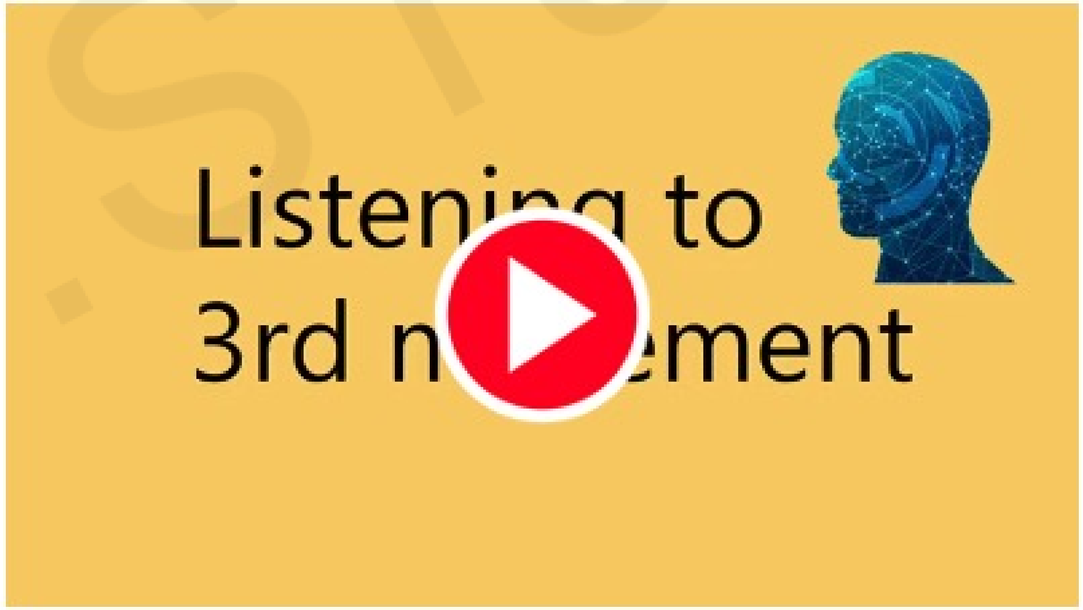
MIT 6.S191

Example Task: Music Generation



Input: sheet music

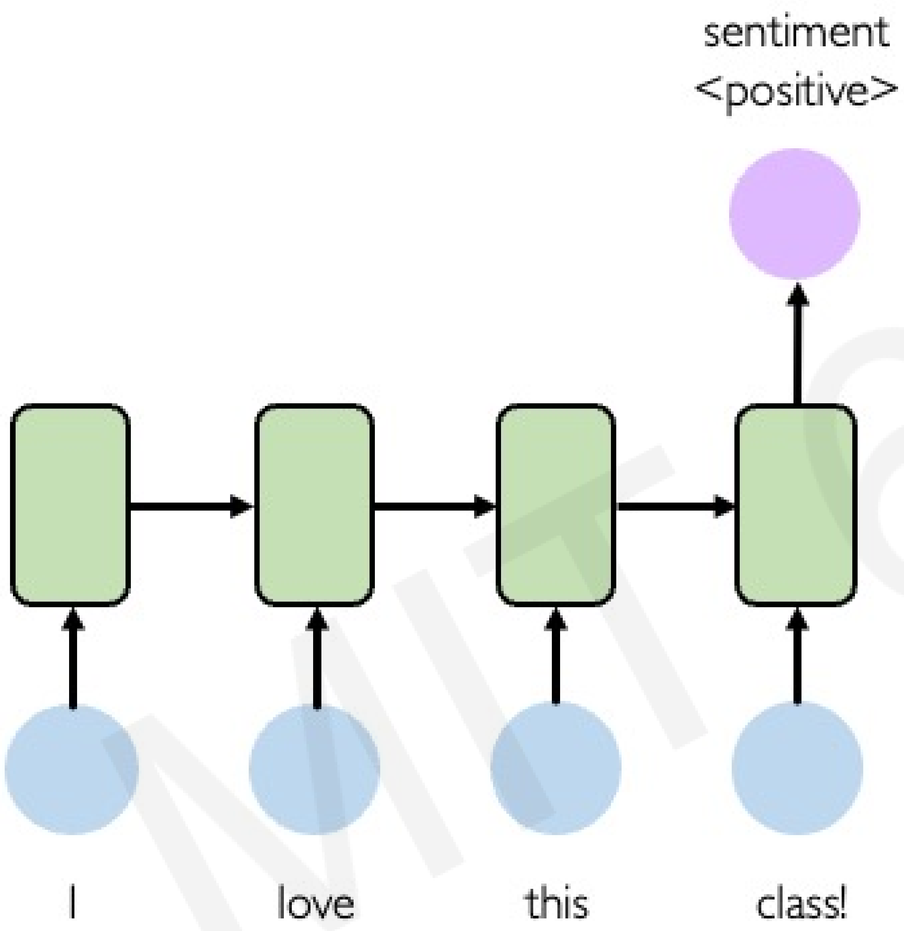
Output: next character in sheet music



6.S191 Lab!

MIT 6.S191

Example Task: Sentiment Classification



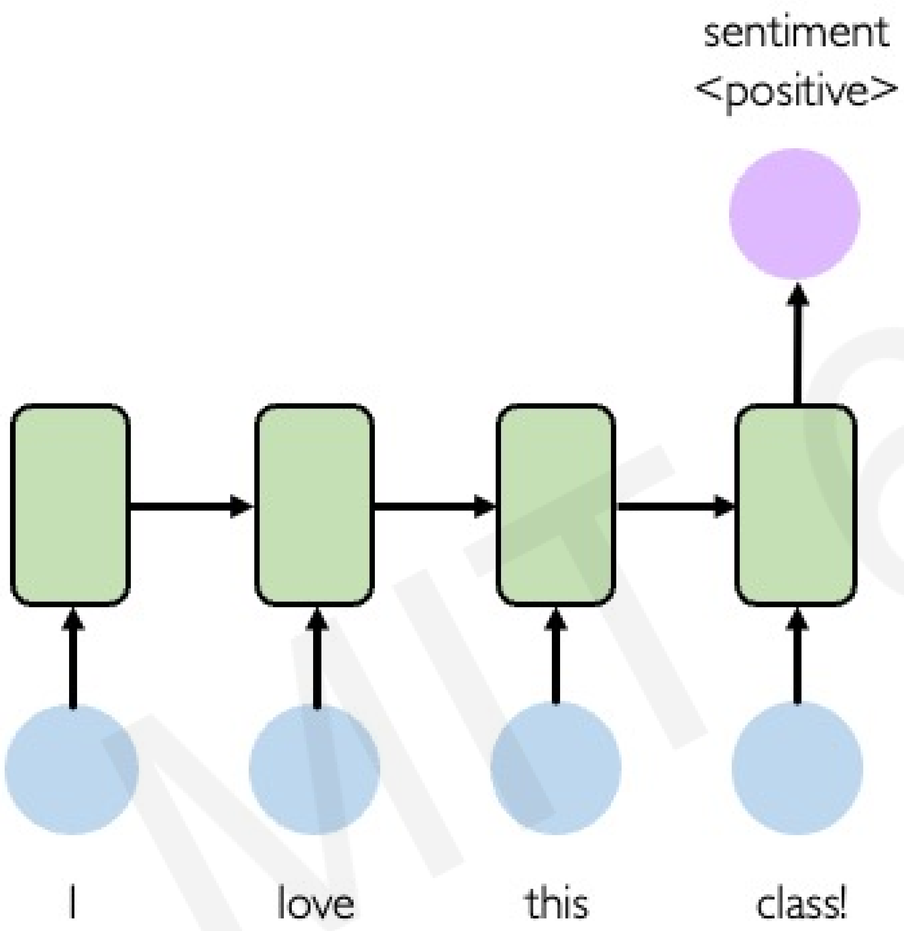
Input: sequence of words

Output: probability of having positive sentiment

```
loss = tf.nn.softmax_cross_entropy_with_logits(y, predicted)
```

MIT6.S191

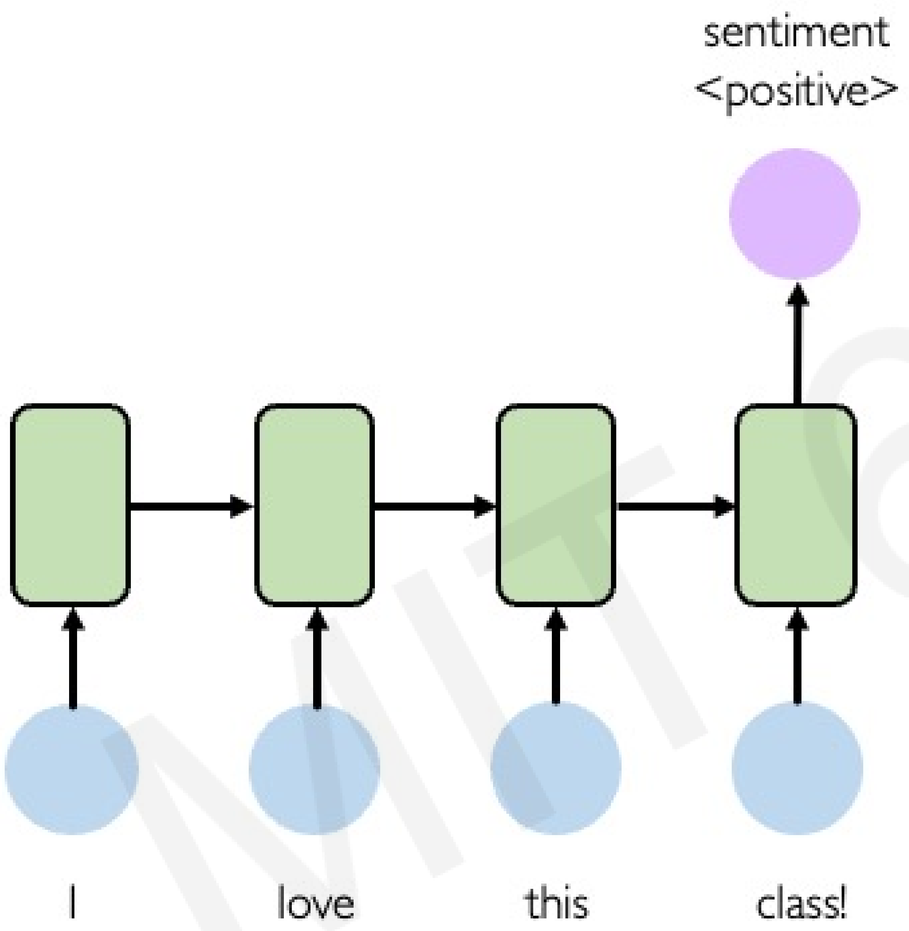
Example Task: Sentiment Classification



Tweet sentiment classification

MIT 6.S191

Limitations of Recurrent Models



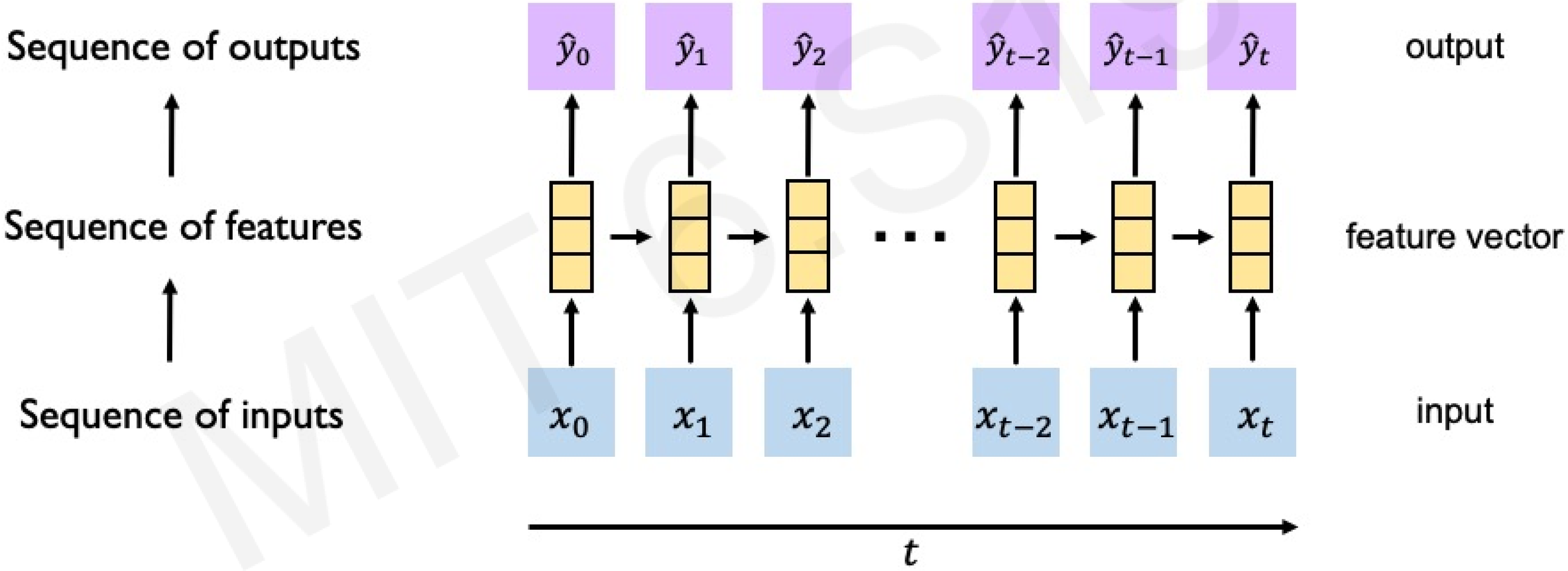
Limitations of RNNs

- Encoding bottleneck
- Slow, no parallelization
- Not long memory

MIT6.S191

Goal of Sequence Modeling

RNNs: recurrence to model sequence dependencies



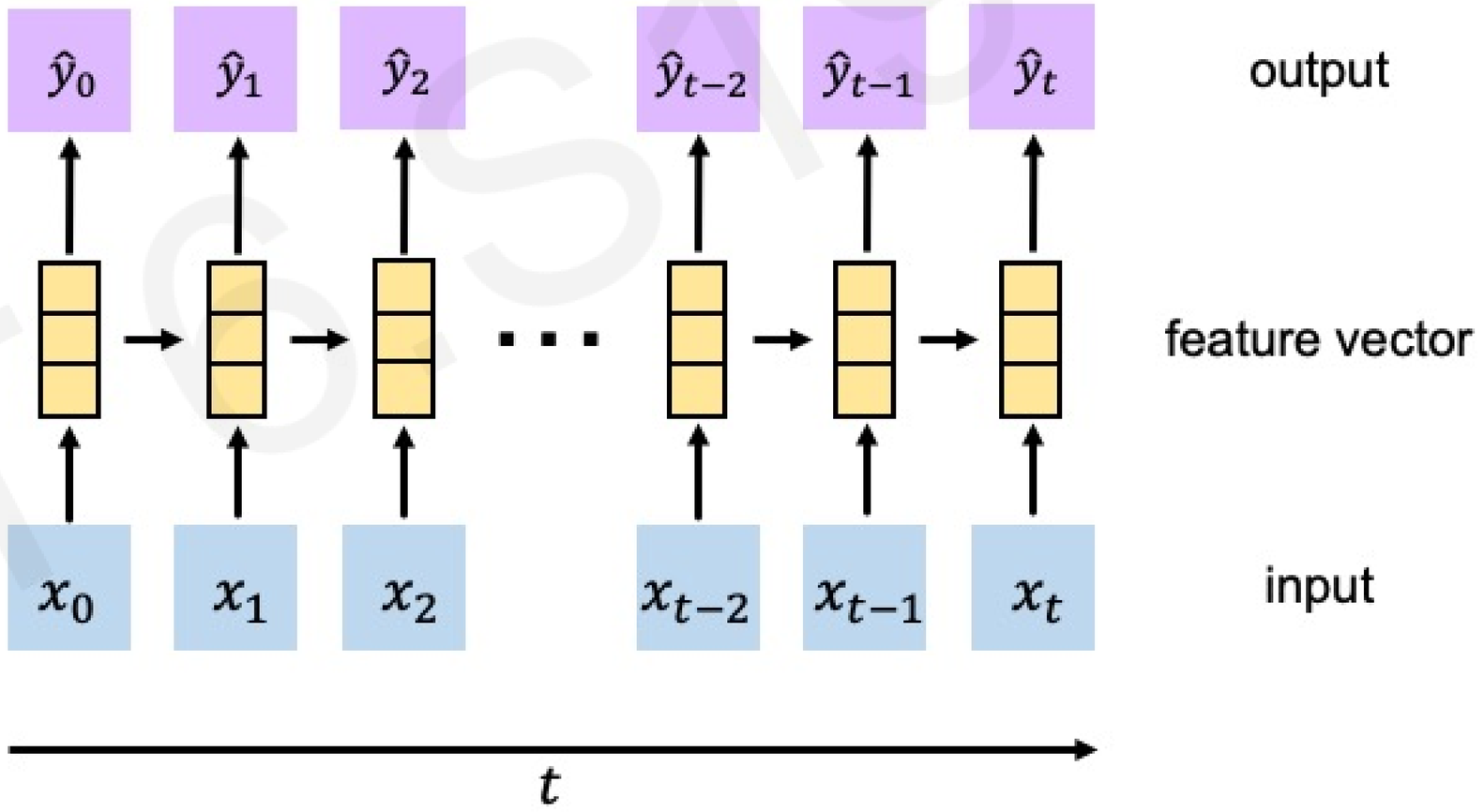
MITS491

Goal of Sequence Modeling

RNNs: recurrence to model sequence dependencies

Limitations of RNNs

- Encoding bottleneck
- Slow, no parallelization
- Not long memory



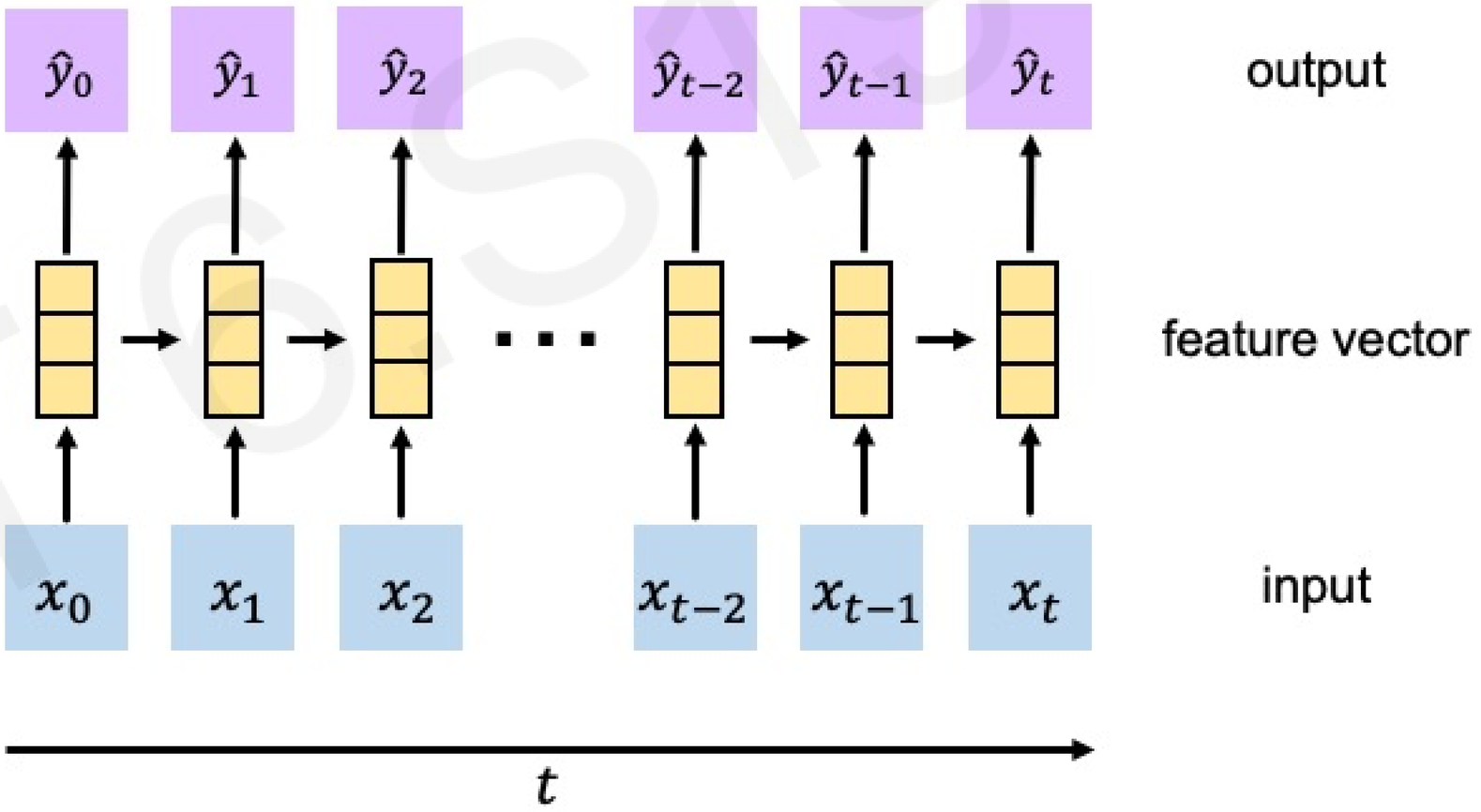
MITS491

Goal of Sequence Modeling

Can we eliminate the need for recurrence entirely?

Desired Capabilities

- Continuous stream
- Parallelization
- Long memory



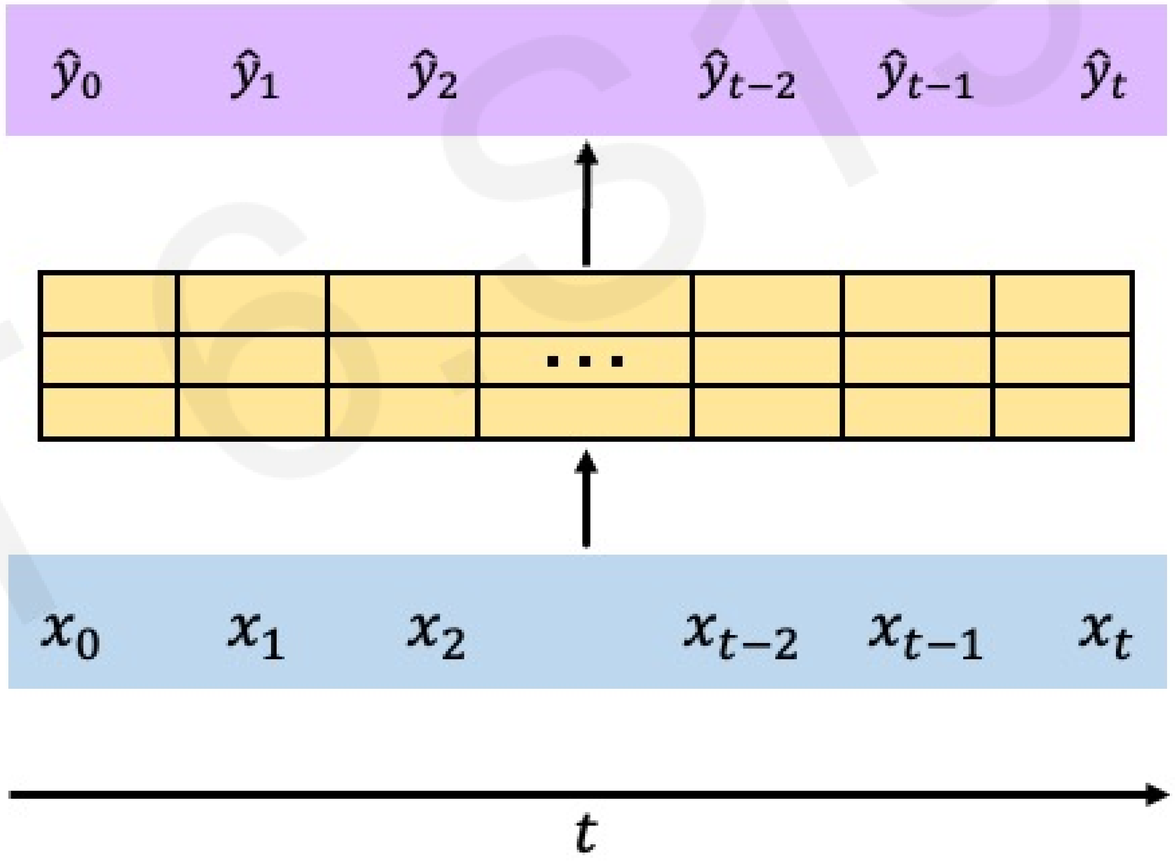
MITS.S191

Goal of Sequence Modeling

Can we eliminate the need for recurrence entirely?

Desired Capabilities

- Continuous stream
- Parallelization
- x_0 Long memory



MIT 6.S191

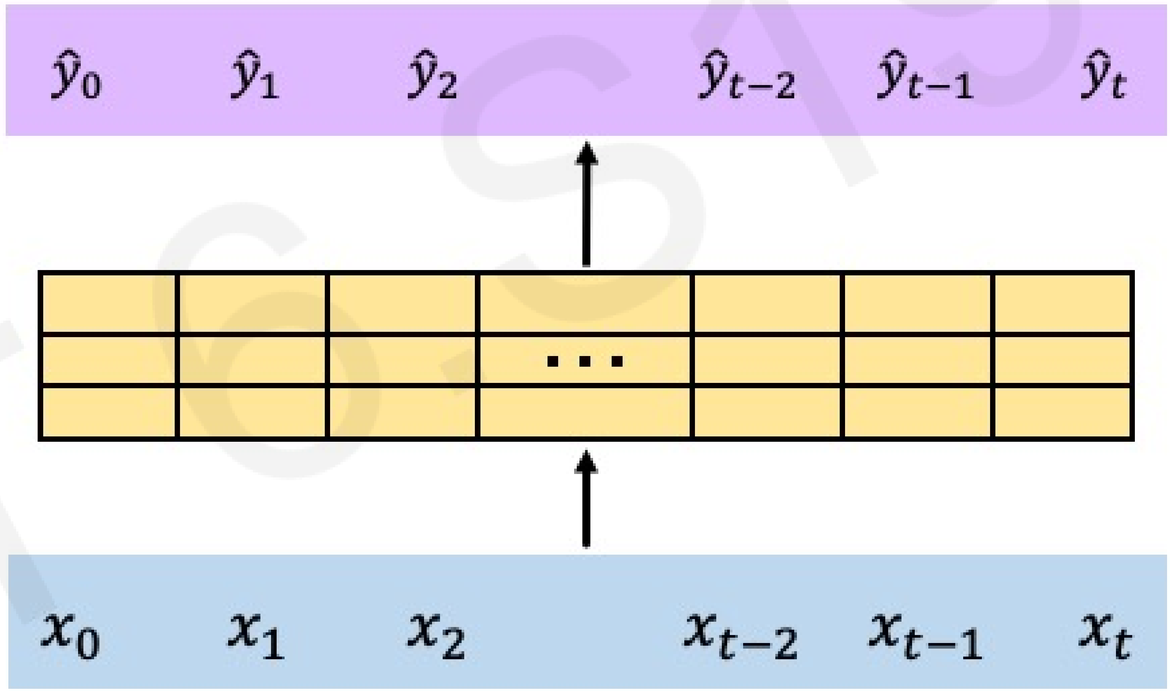
Goal of Sequence Modeling

Idea 1: Feed everything into dense network

- ✓ No recurrence
- ✗ Not scalable
- ✗ No order
- ✗ No long memory

Idea: Identify and attend to what's important

Can we eliminate the need for recurrence entirely?



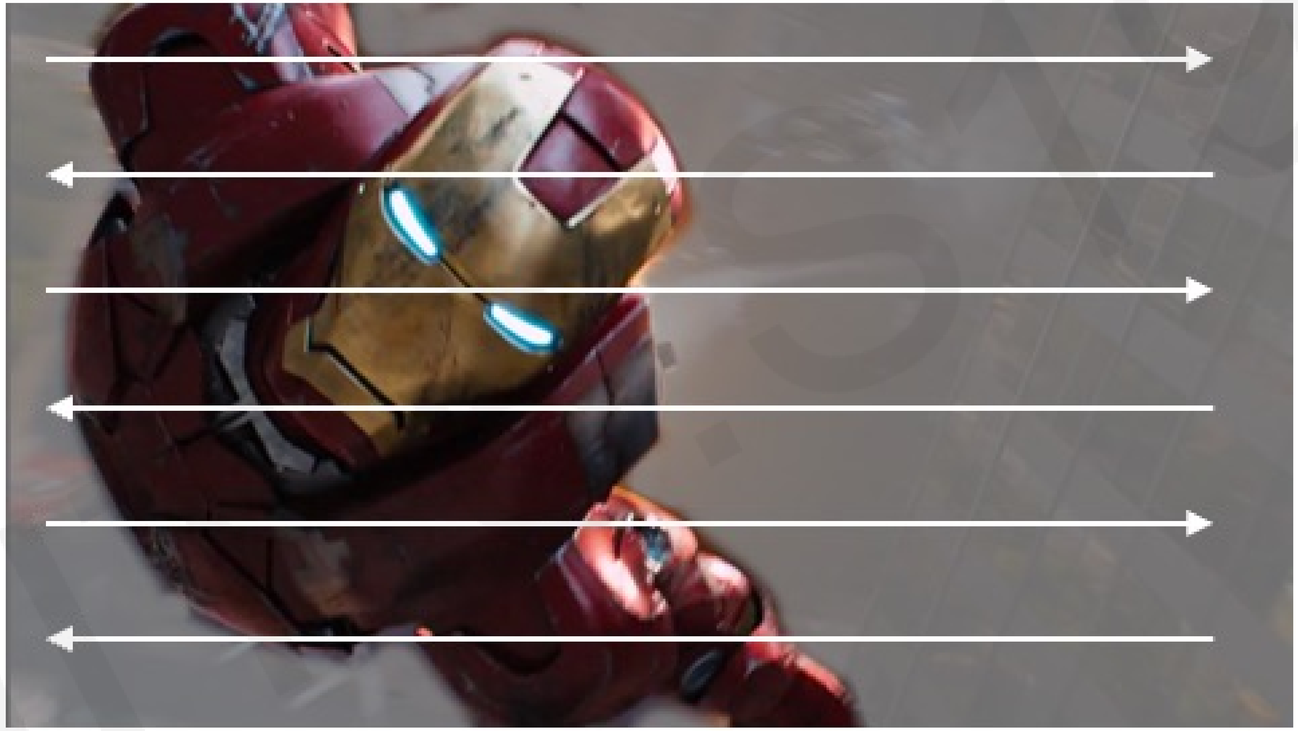
MIT 6.S191

Attention Is All You Need

MIT 6.S191

Intuition Behind Self-Attention

Attending to the most important parts of an input.

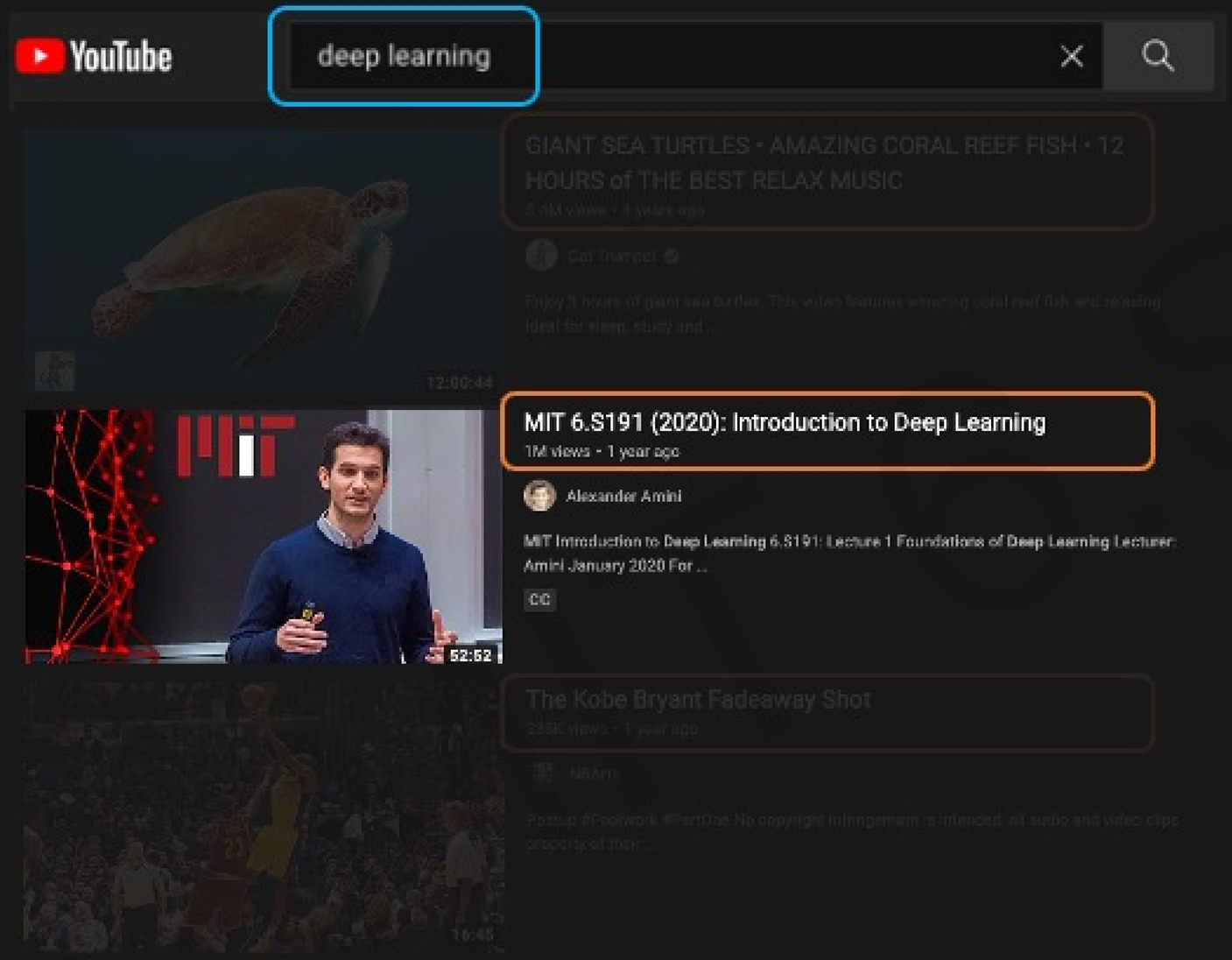


1. Identify which parts to attend to
2. Extract the features with high attention

Similar to a search problem!

A Simple Example: Search

Understanding Attention with Search



Query (Q)

Key (K_1)

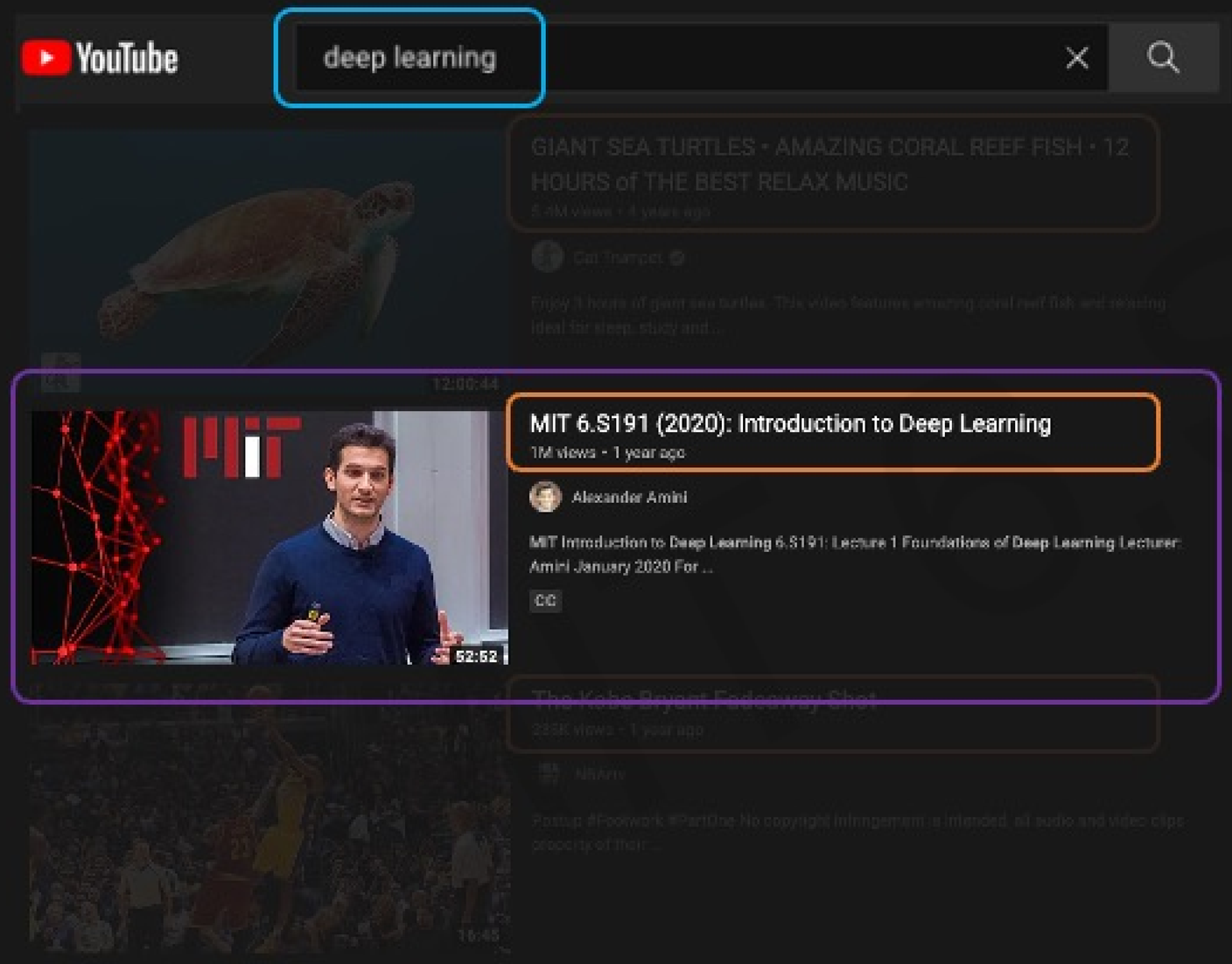
Key (K_2)

Key (K_3)

How similar is the key to the query?

1. **Compute attention mask:** how similar is each key to the desired query?

Understanding Attention with Search



Query (Q)

Key (K_1)

Key (K_2)

Value (V)

Key (K_3)

2. **Extract values based on attention:**
Return the values highest attention

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

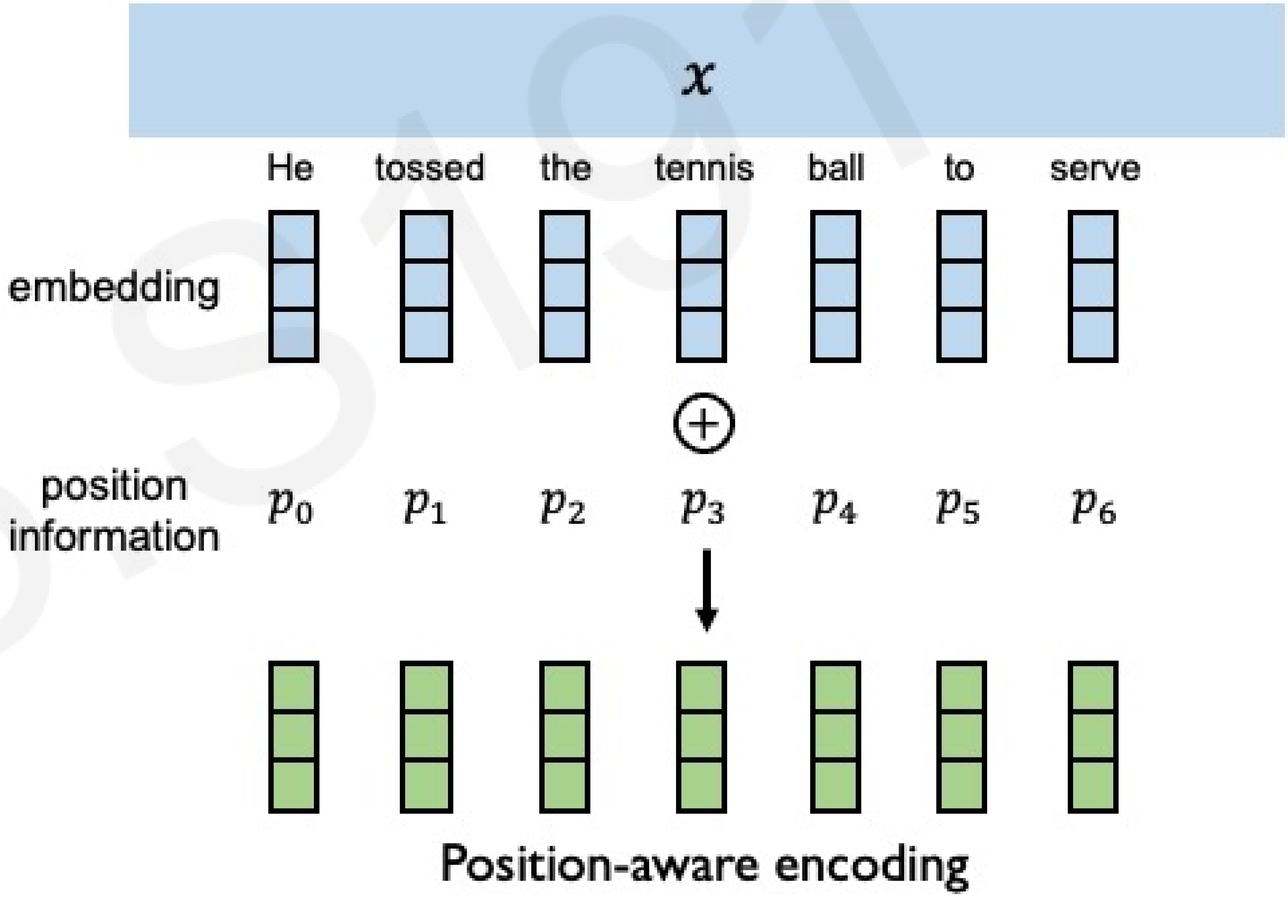
Data is fed in all at once! Need to encode position information to understand order.

MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



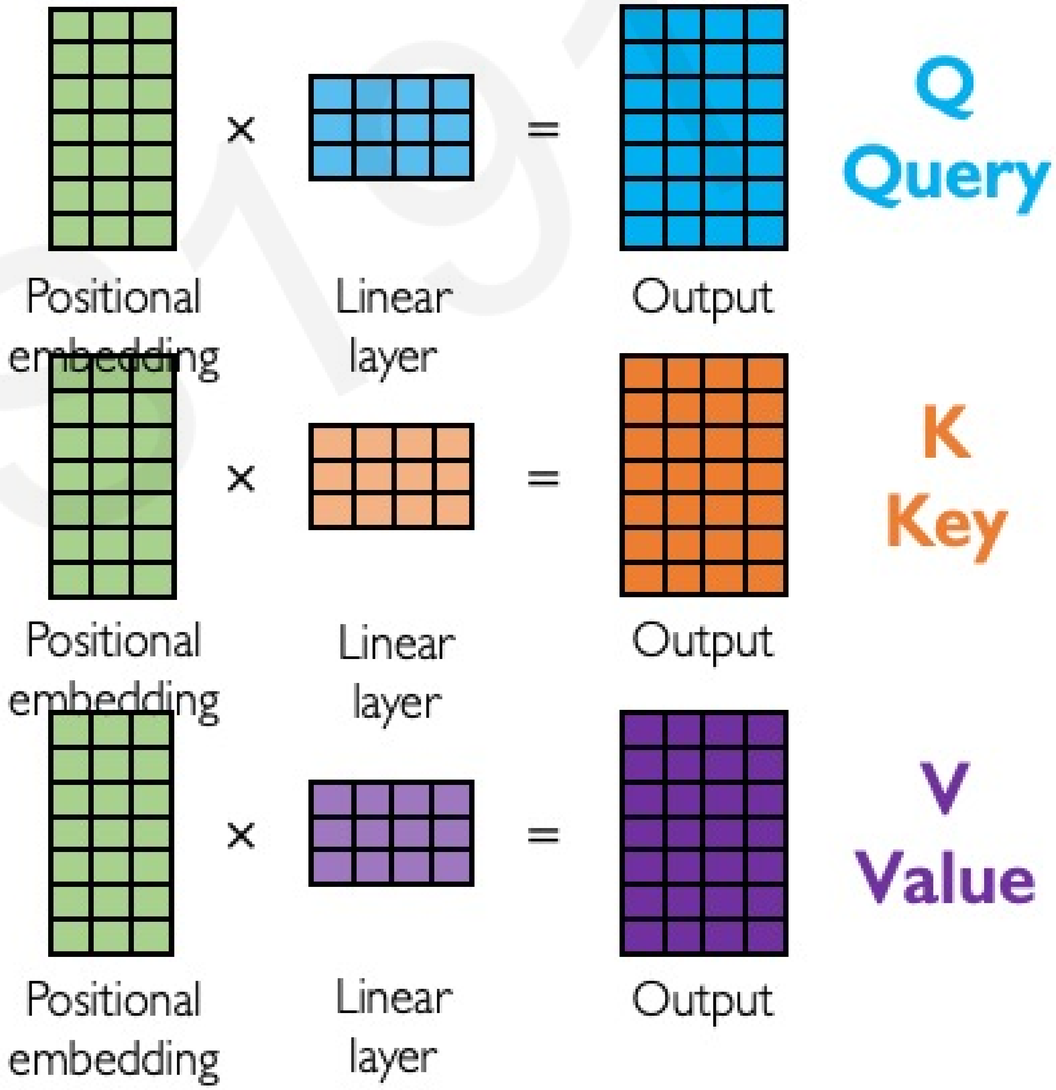
Data is fed in all at once! Need to encode position information to understand order.

MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



MIT 6.S191

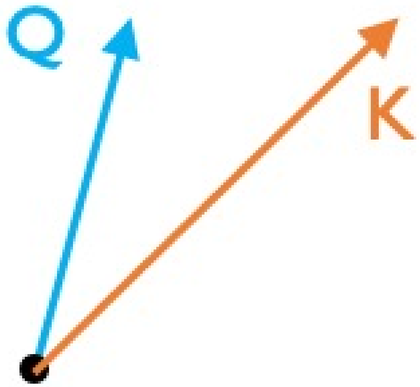
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



Dot product

Scale

$$\frac{Q \cdot K^T}{\text{scaling}}$$

Similarity metric

Also known as the "cosine similarity"

MIT 6.S191

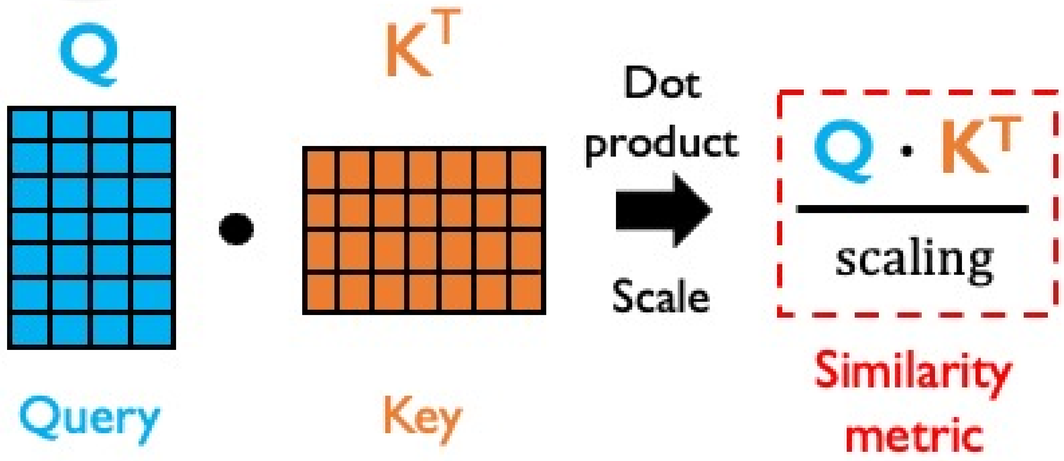
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



Also known as the "cosine similarity"

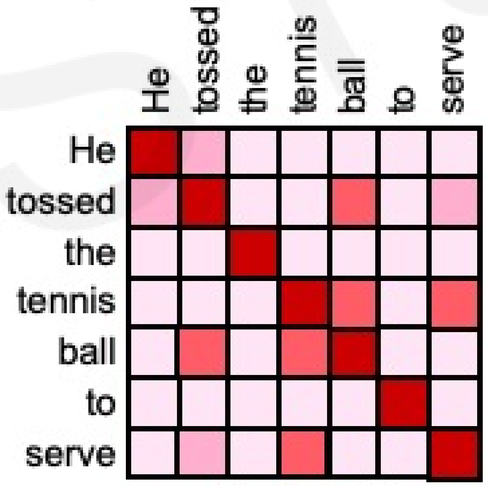
MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Attention weighting: where to attend to! How similar is the key to the query?



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right)$$

Attention weighting

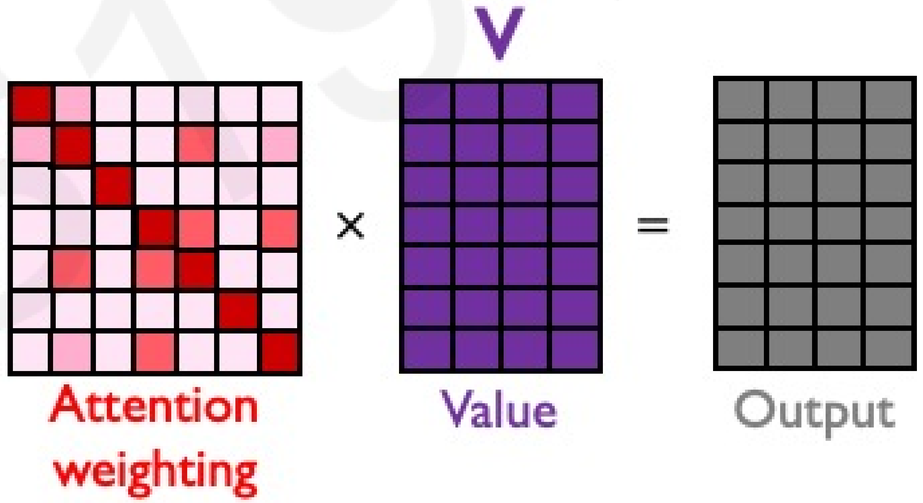
MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q,K,V)$$

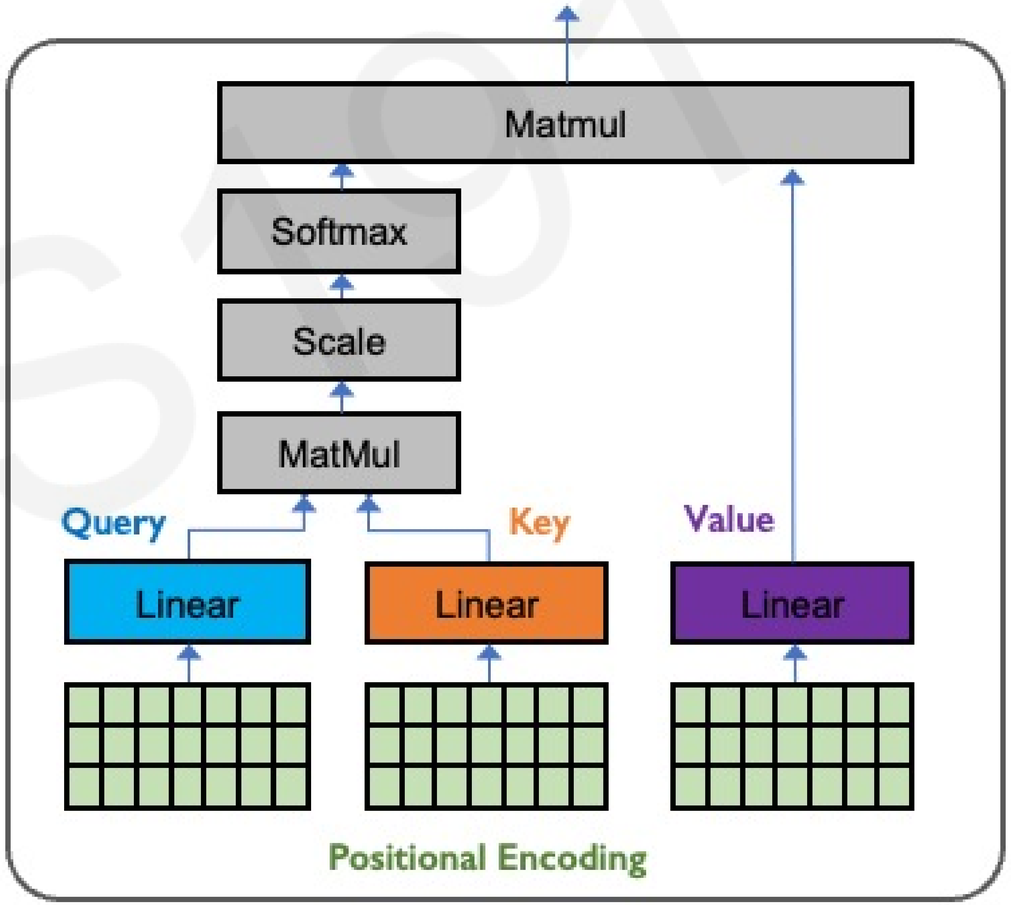
MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

These operations form a self-attention head that can plug into a larger network.
Each head attends to a different part of input.



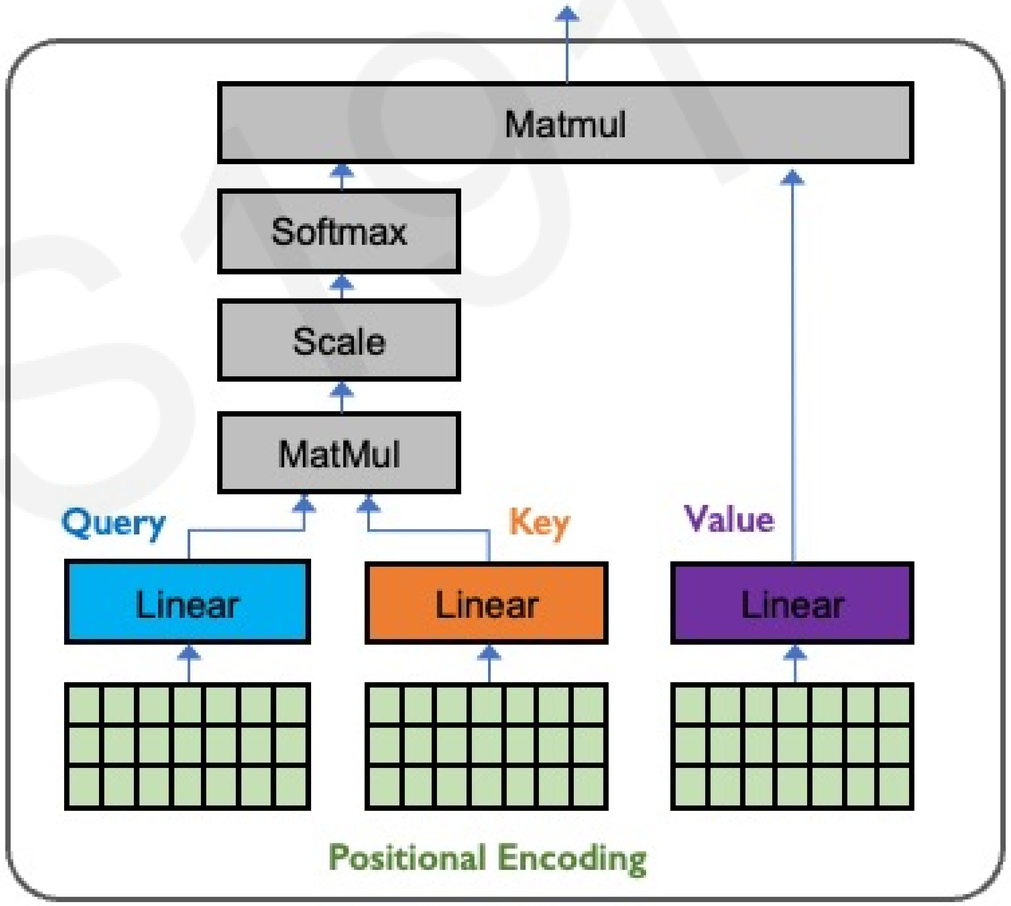
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$

MIT 6.S191

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



Attention is the foundational building block of the <u>Transformer</u> architecture.

MIT 6.S191

Applying Multiple Self-Attention Heads

Attention weighting

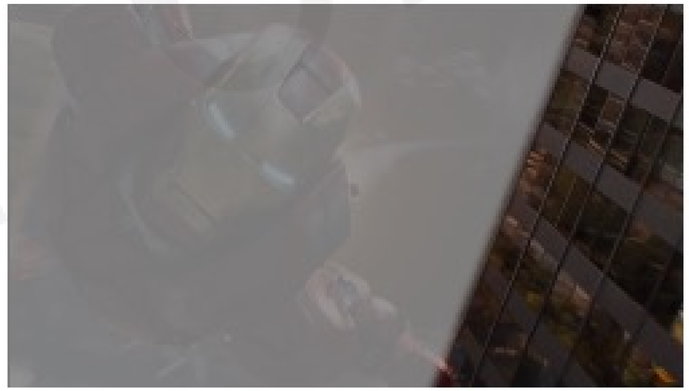
×

Value

=

Output

Output of attention head 1



Output of attention head 2

Output of attention head 3

Self-Attention Applied

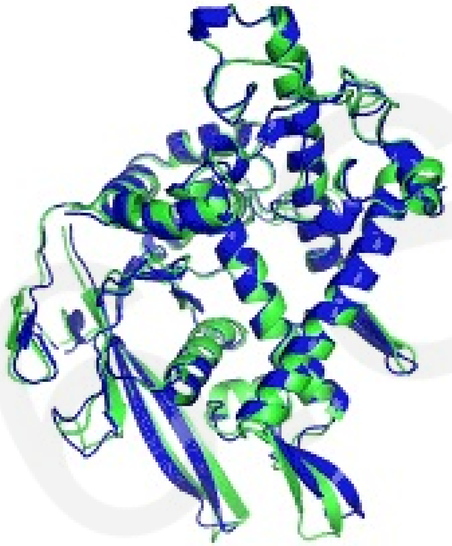
Language Processing

An armchair in the shape of an avocado

Transformers: BERT, GPT

Devlin et al., *NAACL* 2019
Brown et al., *NeurIPS* 2020

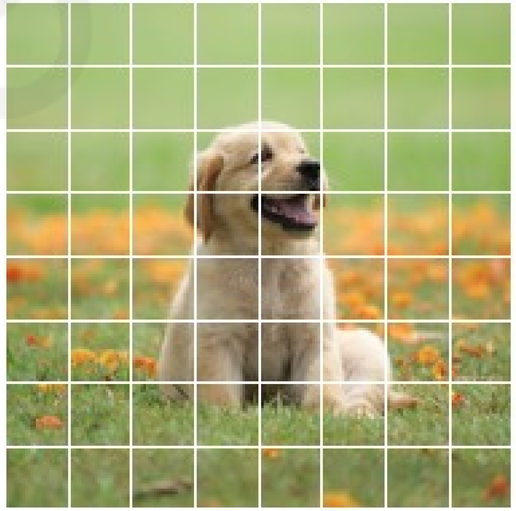
Biological Sequences



Protein Structure Models

Jumper et al., *Nature* 2021
Lin et al., *Science* 2023

Computer Vision



Vision Transformers

Dosovitskiy et al., *ICLR* 2020

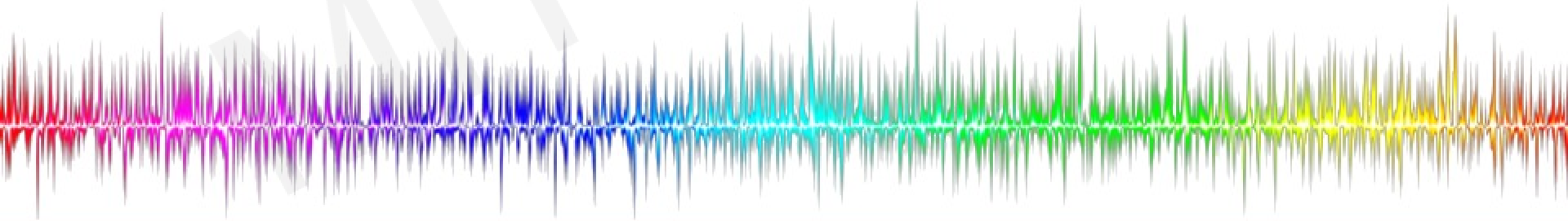
6.S191 Lab and Lectures!

MIT 6.S191

Deep Learning for Sequence Modeling: Summary

1. RNNs are well suited for **sequence modeling** tasks
2. Model sequences via a **recurrence relation**
3. Training RNNs with **backpropagation through time**
4. Models for **music generation**, classification, machine translation, and more
5. Self-attention to model **sequences without recurrence**
6. Self-attention is the basis for many **large language models** – stay tuned!

MIT 6.S191



6.S191: Introduction to Deep Learning

Lab 1: Deep Learning in Python and Music Generation with RNNs

Link to download labs:
http://introtodeeplearning.com#schedule

1. Open the lab in Google Colab
2. Start executing code blocks and filling in the #TODOs
3. Need help? Find a TA/instructor!