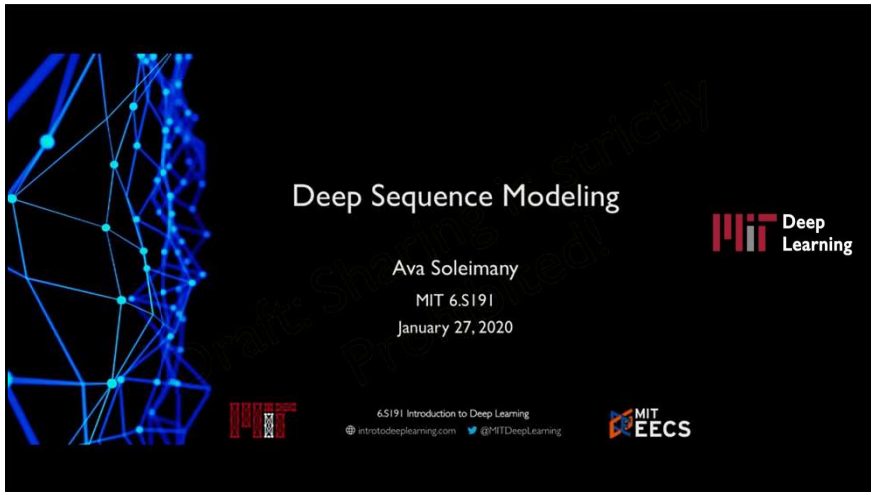
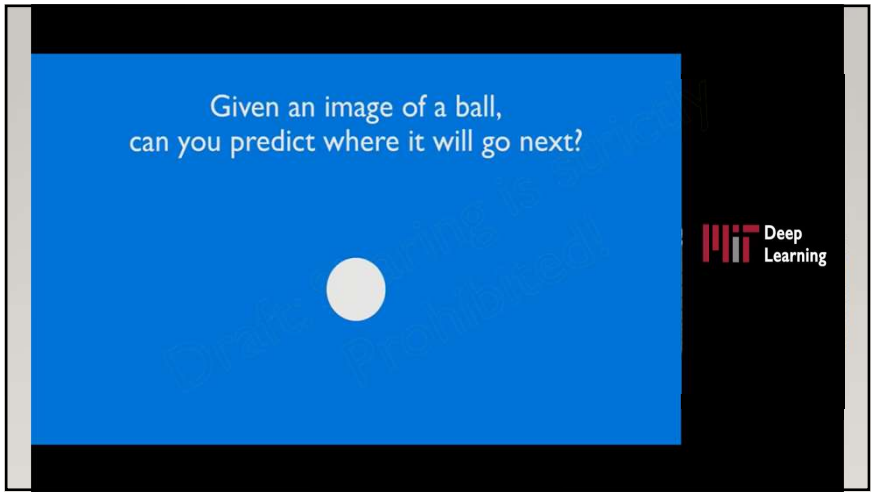
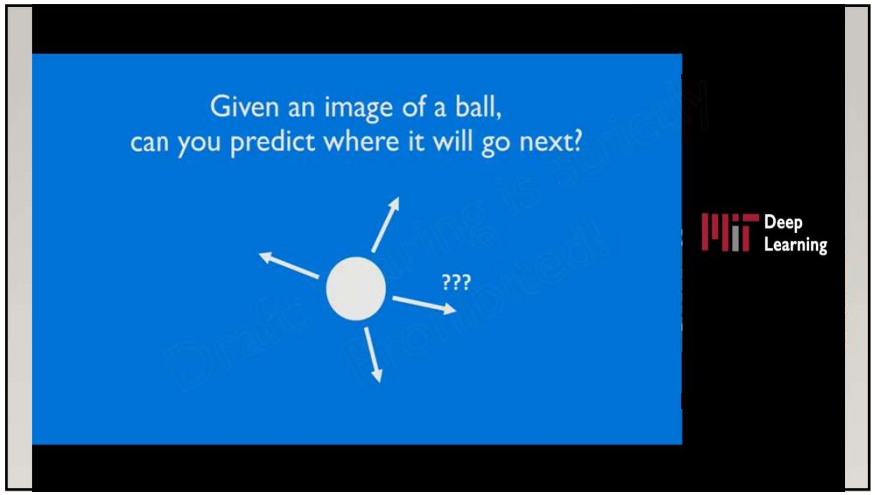
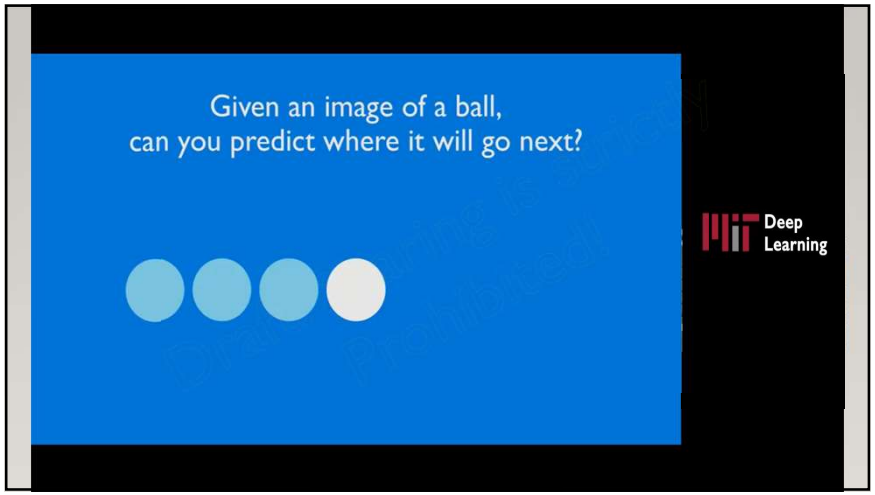

Deep Sequence Modeling
Ava Soleimany
MIT 6.S191
January 27, 2020
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
MIT EECS


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

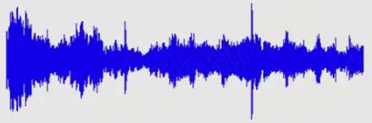

Given an image of a ball,
can you predict where it will go next?
???
Deep Learning


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

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

Deep Learning

Sequences in the Wild



Audio

Deep Learning

Sequences in the Wild

Audio

Deep Learning

Sequences in the Wild

6.S191 Introduction to Deep Learning

Text

Deep Learning

Sequences in the Wild

character: 6 . S I 9 I

word: Introduction to Deep Learning

Text

Deep Learning

A Sequence Modeling Problem: Predict the Next Word

Deep Learning

A Sequence Modeling Problem: Predict the Next Word

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

Deep Learning

A Sequence Modeling Problem: Predict the Next Word

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

given these words

Deep Learning

A Sequence Modeling Problem: Predict the Next Word

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

given these words — predict the next word

Deep Learning

Idea #1: Use a Fixed Window

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

given these two words — predict the next word

Deep Learning

Idea #1: Use a Fixed Window

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

given these two words — predict the next word

One-hot feature encoding: tells us what each word is

[1 0 0 0 0 0 1 0 0 0]

for a

↓

prediction

Deep Learning

Problem #1: Can't 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.

Deep Learning

Idea #2: Use Entire Sequence as Set of Counts

"This morning I took my cat for a"

↓

"bag of words"

[0 1 0 0 1 0 0 ... 0 0 1 1 0 0 0 1]

↓

prediction

Deep Learning

Problem #2: Counts Don't Preserve Order

The food was good, not bad at all.

vs.

The food was bad, not good at all.

Deep Learning

Idea #3: Use a Really Big Fixed Window

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

given these words — predict the next word

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

morning I took this cat

↓

prediction

Deep Learning

Problem #3: No Parameter Sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter:**

Deep Learning

Problem #3: No Parameter Sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter**:

[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 ...]

this morning

Deep Learning

Problem #3: No Parameter Sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter**:

[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 ...]

this morning

Things we learn about the sequence **won't transfer** if they appear **elsewhere** in the sequence.

Deep Learning

6.S191 Introduction to Deep Learning introtodeeplearning.com @MITDeepLearning H. Suresh, 6.S191 2018. 1/27/20

Sequence Modeling: Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences

Deep Learning

6.S191 Introduction to Deep Learning introtodeeplearning.com @MITDeepLearning 1/27/20

Sequence Modeling: Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies

Deep Learning

6.S191 Introduction to Deep Learning introtodeeplearning.com @MITDeepLearning 1/27/20

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**

Deep Learning

Sequence Modeling: Design Criteria

To model sequences, we need to:

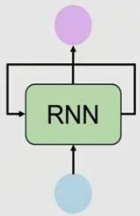
1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence

Deep Learning

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



Today: **Recurrent Neural Networks (RNNs)** as an approach to sequence modeling problems

Deep Learning

Recurrent Neural Networks (RNNs)

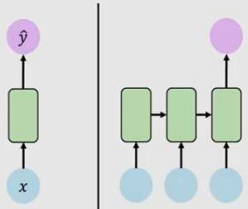
Deep Learning

Standard Feed-Forward Neural Network

One to One
"Vanilla" neural network

Deep Learning

Recurrent Neural Networks for Sequence Modeling

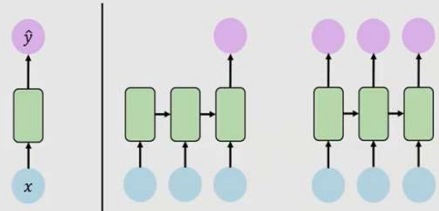


One to One
"Vanilla" neural network

Many to One
Sentiment Classification

Deep Learning

Recurrent Neural Networks for Sequence Modeling



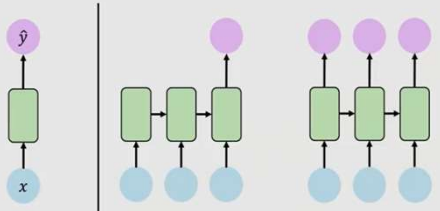
One to One
"Vanilla" neural network

Many to One
Sentiment Classification

Many to Many
Music Generation

Deep Learning

Recurrent Neural Networks for Sequence Modeling



One to One
"Vanilla" neural network

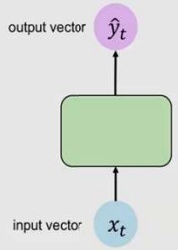
Many to One
Sentiment Classification

Many to Many
Music Generation

… and many other architectures and applications

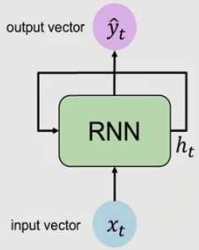
Deep Learning

Standard "Vanilla" Neural Network



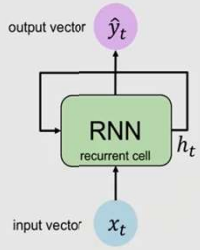
Deep Learning

Recurrent Neural Network (RNN)



Deep Learning

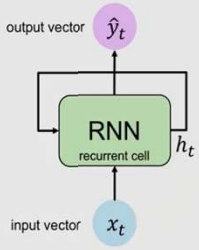
Recurrent Neural Network (RNN)



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

Deep Learning

Recurrent Neural Network (RNN)

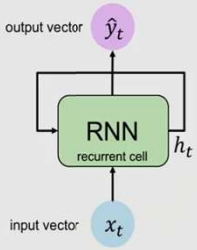


Apply a **recurrence relation** at every time step to process a sequence:

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

Deep Learning

Recurrent Neural Network (RNN)



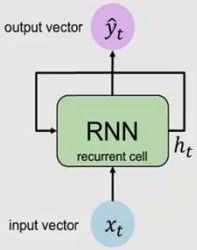
Apply a **recurrence relation** at every time step to process a sequence:

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

cell state | function parameterized by W

Deep Learning

Recurrent Neural Network (RNN)



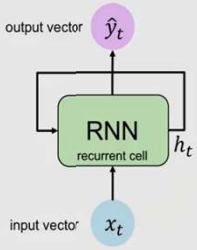
Apply a **recurrence relation** at every time step to process a sequence:

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

cell state | function parameterized by W | old state

Deep Learning

Recurrent Neural Network (RNN)



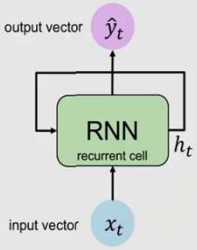
Apply a **recurrence relation** at every time step to process a sequence:

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

cell state | function parameterized by W | old state | input vector at time step t

Deep Learning

Recurrent Neural Network (RNN)



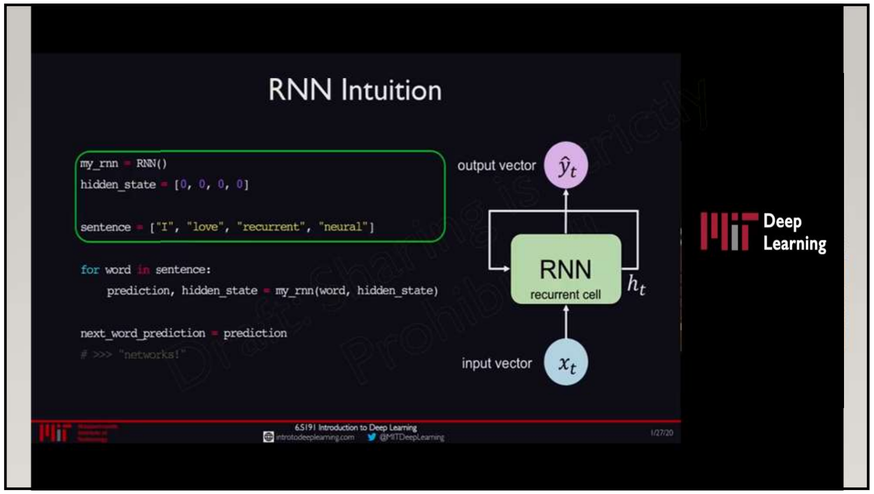
Apply a **recurrence relation** at every time step to process a sequence:

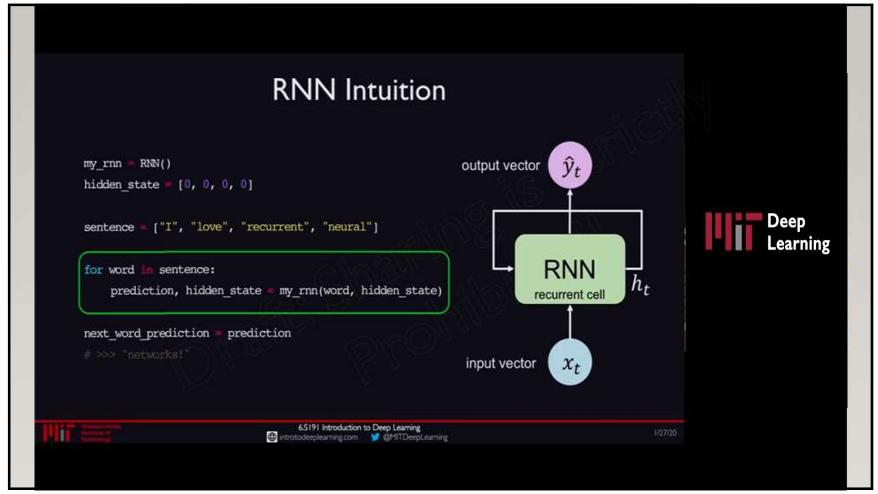
$$h_t = f_W(h_{t-1}, x_t)$$

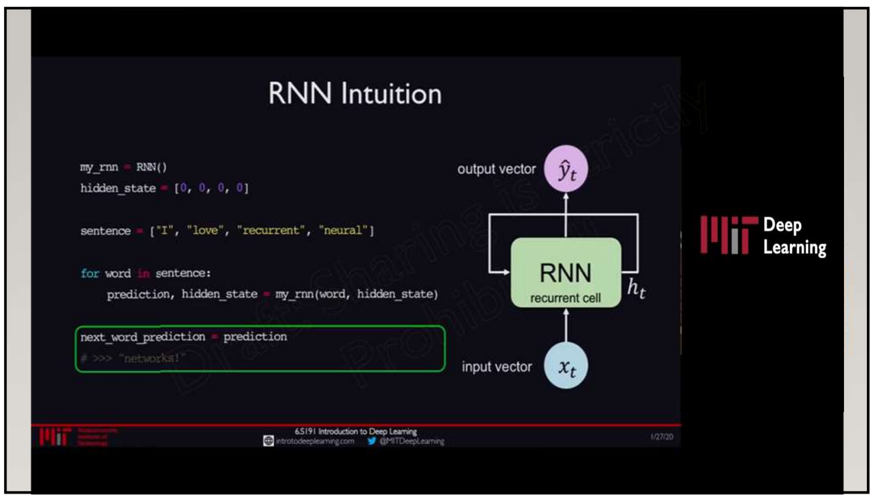
cell state | function parameterized by W | old state | input vector at time step t

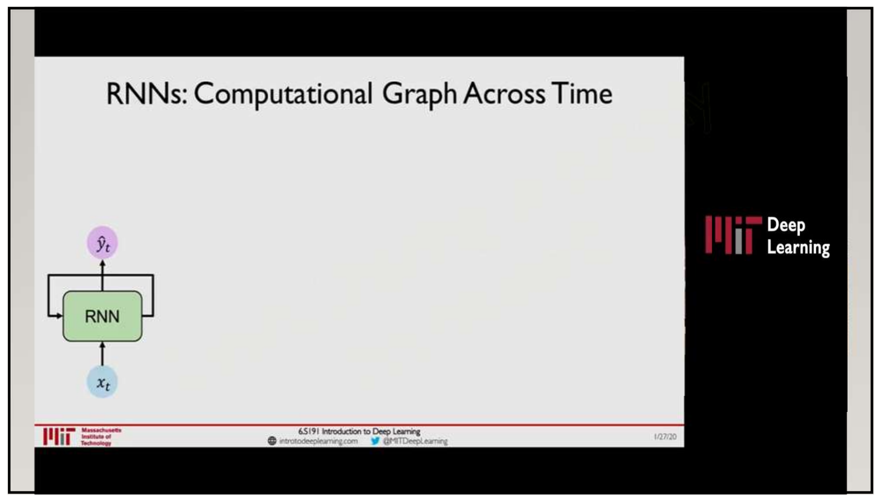
Note: the same function and set of parameters are used at every time step

Deep Learning

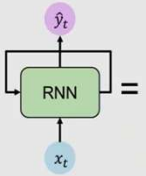

RNN Intuition
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]
sentence = ["I", "love", "recurrent", "neural"]
for word in sentence:
prediction, hidden_state = my_rnn(word, hidden_state)
next_word_prediction = prediction
# >>> "networks!"
output vector
RNN
recurrent cell
input vector
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20


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!"
output vector
RNN
recurrent cell
input vector
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20


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!"
output vector
RNN
recurrent cell
input vector
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20


RNNs: Computational Graph Across Time
RNN
Deep Learning
Massachusetts Institute of Technology
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20

RNNs: Computational Graph Across Time

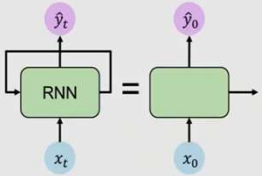


=

Represent as computational graph unrolled across time

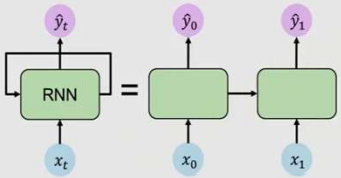
Deep Learning

RNNs: Computational Graph Across Time



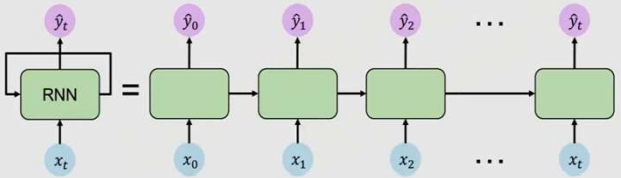
Deep Learning

RNNs: Computational Graph Across Time



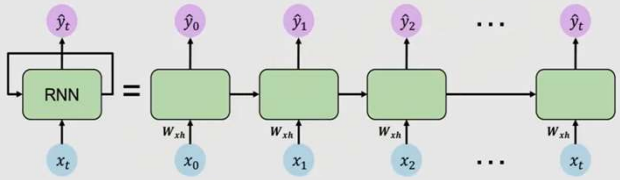
Deep Learning

RNNs: Computational Graph Across Time



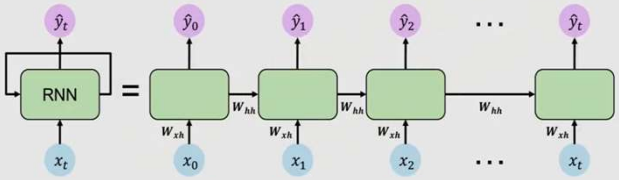
Deep Learning

RNNs: Computational Graph Across Time



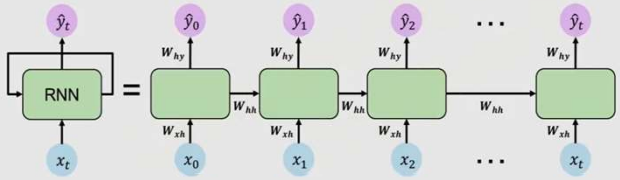
Deep Learning

RNNs: Computational Graph Across Time



Deep Learning

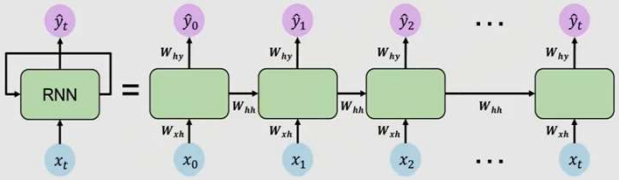
RNNs: Computational Graph Across Time



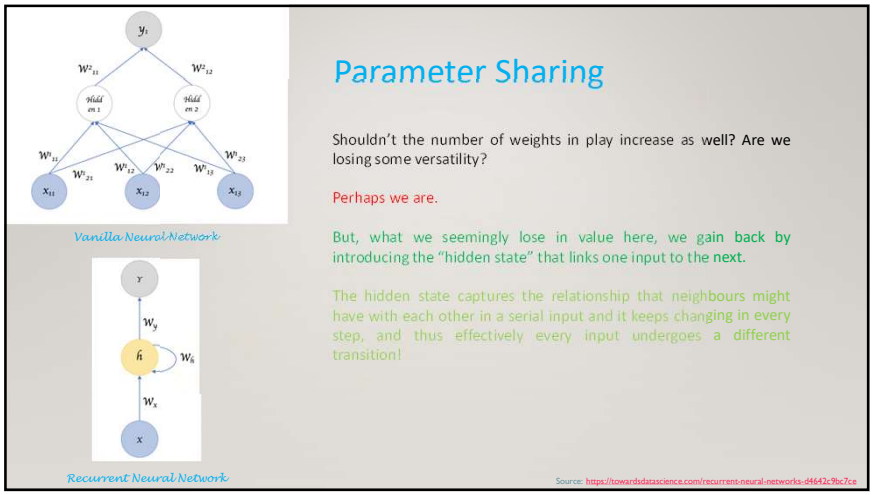
Deep Learning

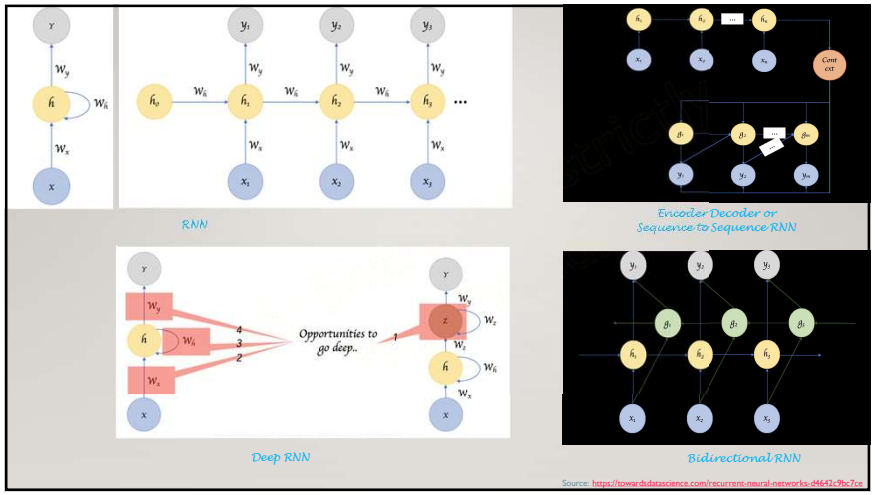
RNNs: Computational Graph Across Time

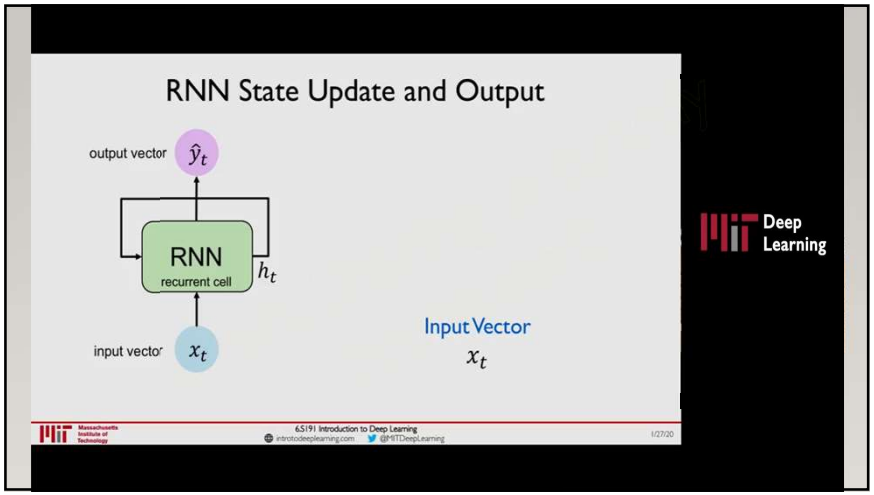
Re-use the **same weight matrices** at every time step

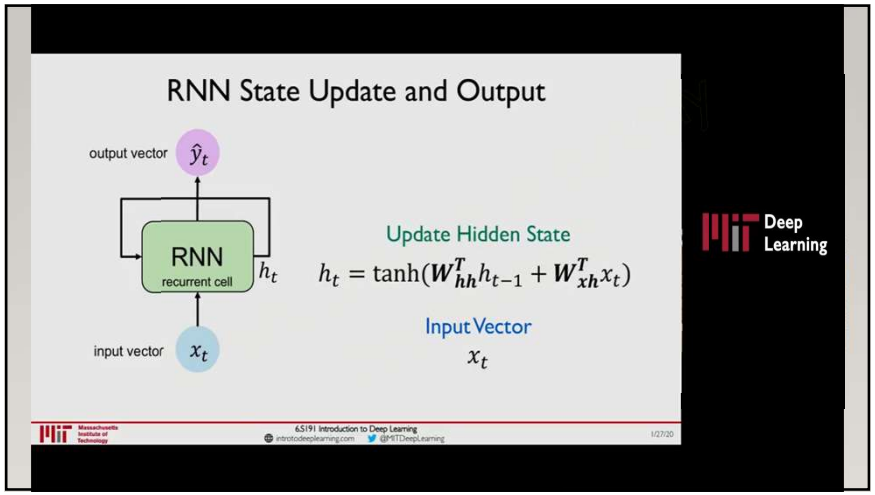


Deep Learning

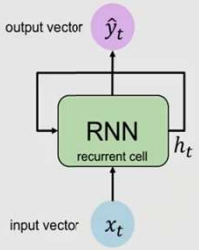

Parameter Sharing
Shouldn't the number of weights in play increase as well? Are we losing some versatility?
Perhaps we are.
But, what we seemingly lose in value here, we gain back by introducing the "hidden state" that links one input to the next.
The hidden state captures the relationship that neighbours might have with each other in a serial input and it keeps changing in every step, and thus effectively every input undergoes a different transition!
Vanilla Neural Network
Recurrent Neural Network
Source: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce


RNN
Deep RNN
Opportunities to go deep..
Encoder Decoder or Sequence to Sequence RNN
Bidirectional RNN
Source: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce


RNN State Update and Output
output vector
RNN
recurrent cell
input vector
Input Vector
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20


RNN State Update and Output
output vector
RNN
recurrent cell
input vector
Update Hidden State
Input Vector
Deep Learning
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20

RNN State Update and Output



Output Vector

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

Update Hidden State

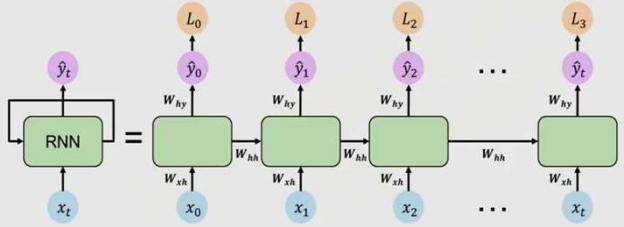
$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

$$x_t$$

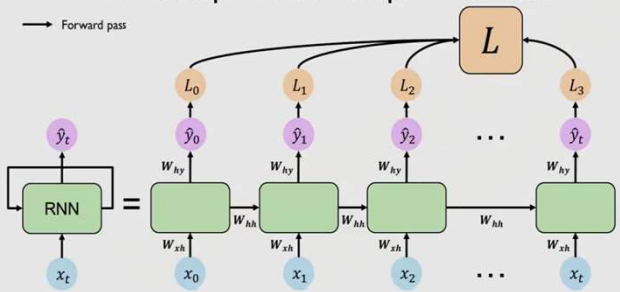
Deep Learning

RNNs: Computational Graph Across Time



Deep Learning

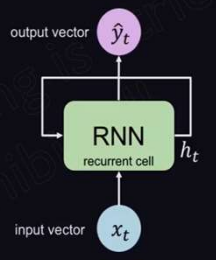
RNNs: Computational Graph Across Time



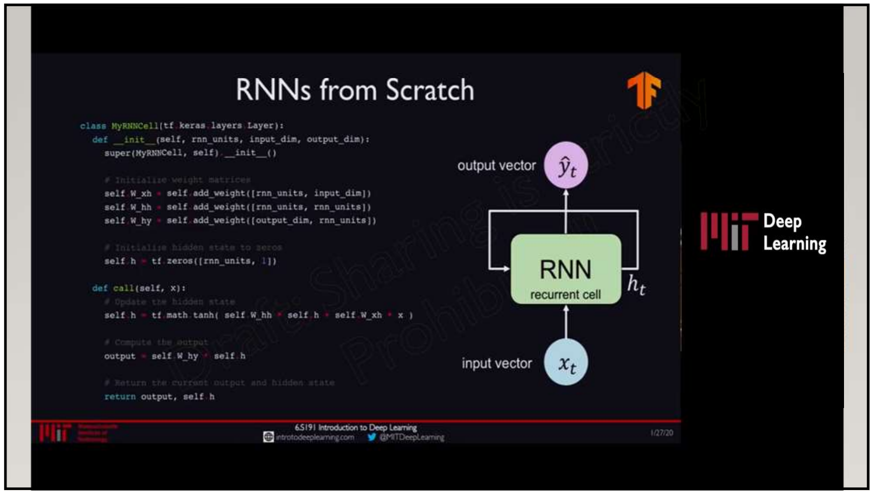
Deep Learning

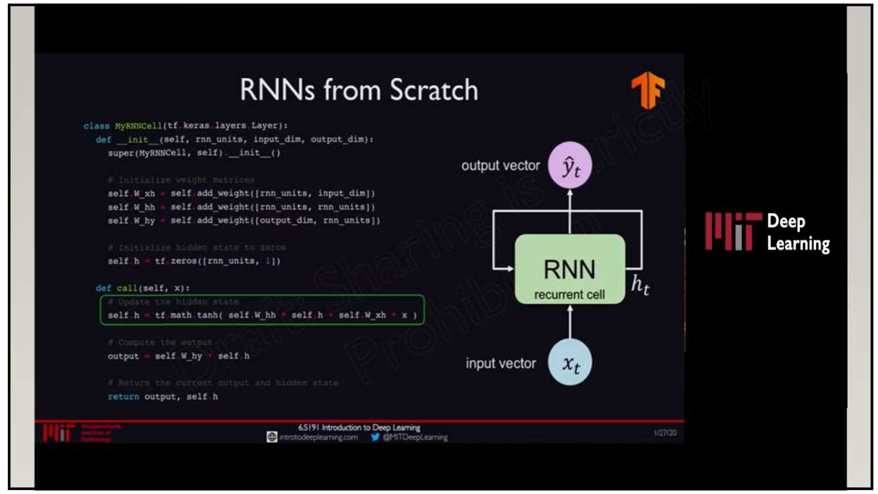
RNNs from Scratch

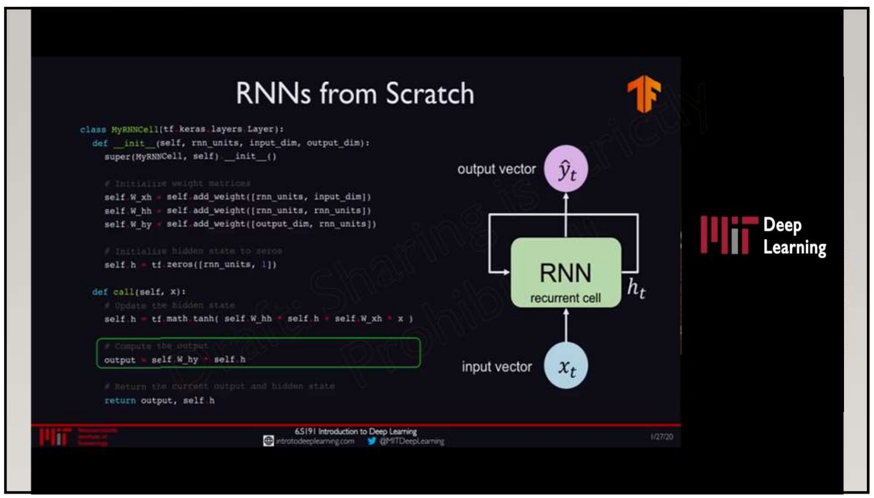
```
class MyRNNCell(tf.keras.layers.Layer):
  def __init__(self, rnn_units, input_dim, output_dim):
    super(MyRNNCell, self).__init__()

    # 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])
```

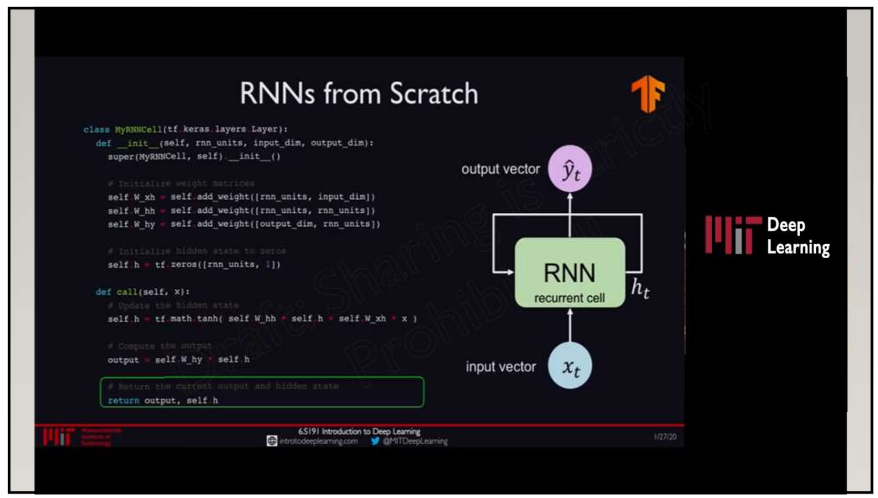


Deep Learning


RNNs from Scratch
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
output vector
$\hat{y}_t$
RNN
recurrent cell
h_t
input vector
x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20
Deep Learning

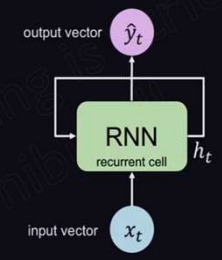

RNNs from Scratch
class MyRNNCell(tf.keras.layers.Layer):
def __init__(self, rnn_units, input_dim, output_dim):
super(MyRNNCell, self).__init__()
# Initialize weight matrices
self.W_xh = self.add_weight([rnn_units, input_dim])
self.W_hh = self.add_weight([rnn_units, rnn_units])
self.W_hy = self.add_weight([output_dim, 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
output vector
$\hat{y}_t$
RNN
recurrent cell
h_t
input vector
x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20
Deep Learning


RNNs from Scratch
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
output vector
$\hat{y}_t$
RNN
recurrent cell
h_t
input vector
x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20
Deep Learning


RNNs from Scratch
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
output vector
$\hat{y}_t$
RNN
recurrent cell
h_t
input vector
x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com
@MITDeepLearning
1/27/20
Deep Learning

RNN Implementation in TensorFlow

```
tf.keras.layers.SimpleRNN(rnn_units)
```



Deep Learning

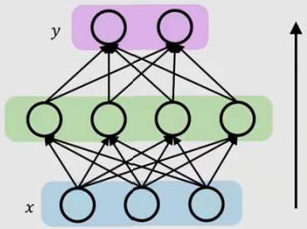
Backpropagation Through Time (BPTT)

Deep Learning

Recall: Backpropagation in Feed Forward Models

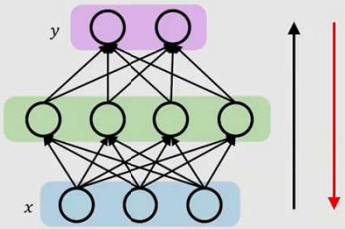
Deep Learning

Recall: Backpropagation in Feed Forward Models



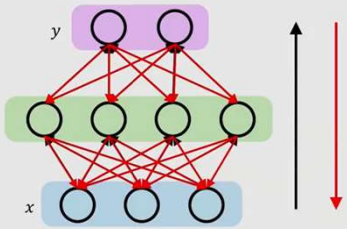
Deep Learning

Recall: Backpropagation in Feed Forward Models



Deep Learning

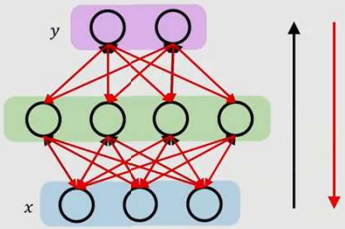
Recall: Backpropagation in Feed Forward Models



Backpropagation algorithm:

Deep Learning

Recall: Backpropagation in Feed Forward Models

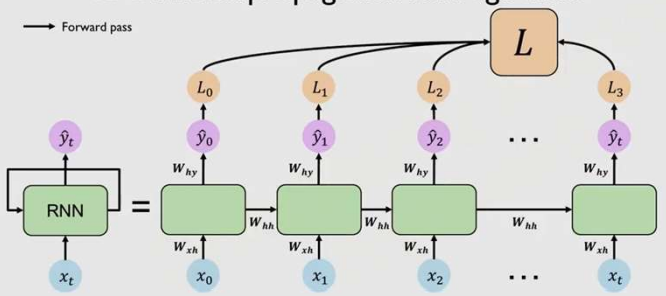


Backpropagation algorithm:

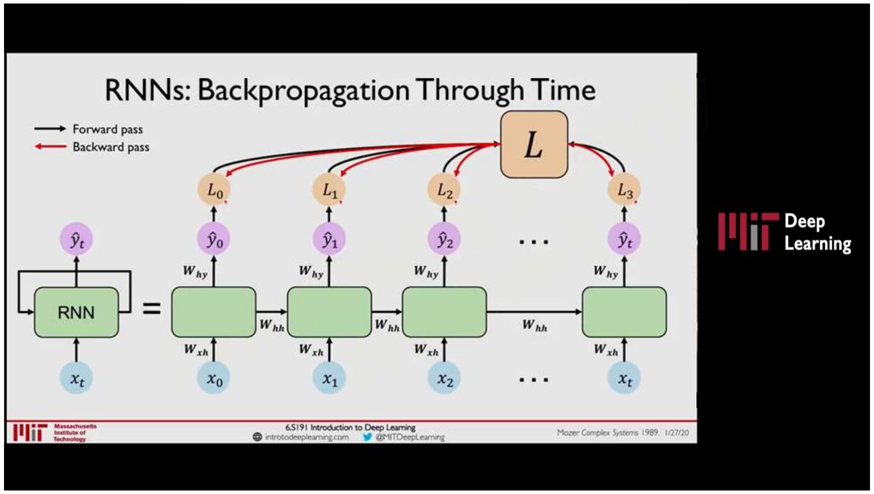
1. Take the derivative (gradient) of the loss with respect to each parameter
2. Shift parameters in order to minimize loss

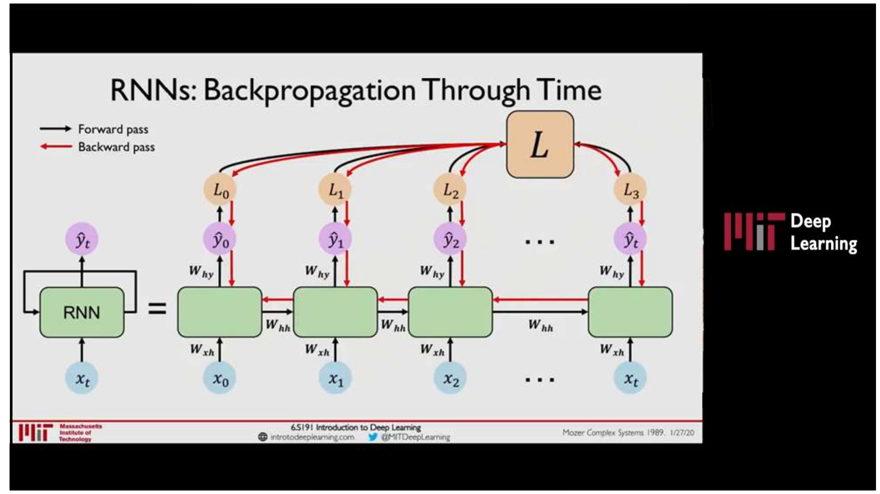
Deep Learning

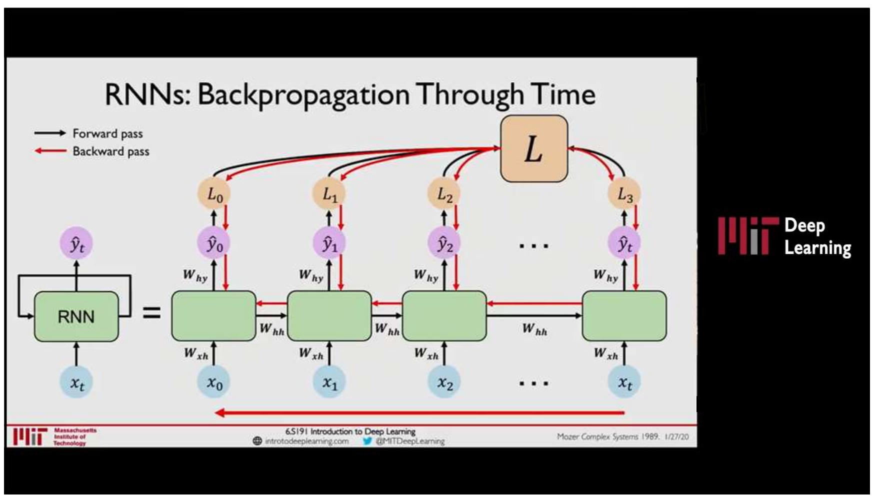
RNNs: Backpropagation Through Time

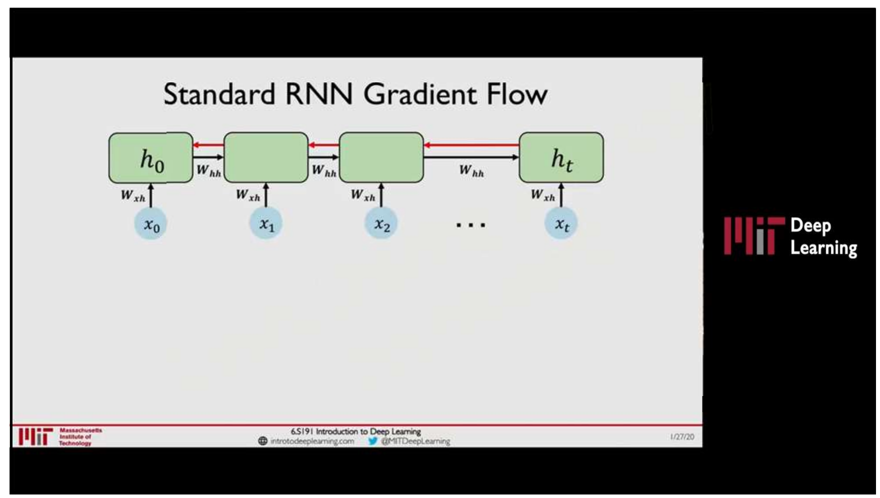


Deep Learning

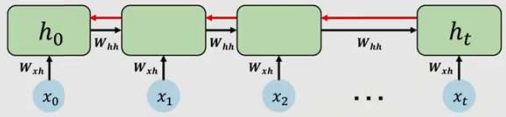

RNNs: Backpropagation Through Time
Forward pass
Backward pass
L
L_0 L_1 L_2 L_3
$\hat{y}_t$ $\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_t$
W_{hy}
RNN =
W_{hh}
W_{xh}
x_t x_0 x_1 x_2 ... x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com @MITDeepLearning
Mozer Complex Systems 1989. 1/27/20
Deep Learning


RNNs: Backpropagation Through Time
Forward pass
Backward pass
L
L_0 L_1 L_2 L_3
$\hat{y}_t$ $\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_t$
W_{hy}
RNN =
W_{hh}
W_{xh}
x_t x_0 x_1 x_2 ... x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com @MITDeepLearning
Mozer Complex Systems 1989. 1/27/20
Deep Learning


RNNs: Backpropagation Through Time
Forward pass
Backward pass
L
L_0 L_1 L_2 L_3
$\hat{y}_t$ $\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_t$
W_{hy}
RNN =
W_{hh}
W_{xh}
x_t x_0 x_1 x_2 ... x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com @MITDeepLearning
Mozer Complex Systems 1989. 1/27/20
Deep Learning


Standard RNN Gradient Flow
h_0 h_t
W_{hh}
W_{xh}
x_0 x_1 x_2 ... x_t
6.S191 Introduction to Deep Learning
introtodeeplearning.com @MITDeepLearning
1/27/20
Deep Learning

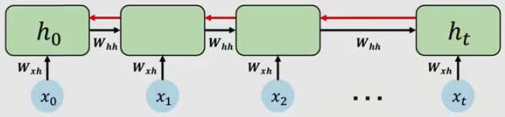
Standard RNN Gradient Flow



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

Deep Learning

Standard RNN Gradient Flow: Exploding Gradients

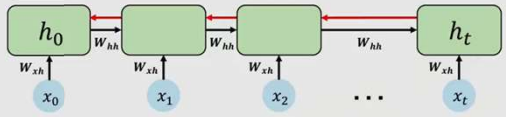


Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation!**

Many values > 1:
exploding gradients

Deep Learning

Standard RNN Gradient Flow: Vanishing Gradients

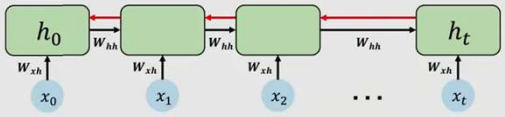


Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation!**

Many values < 1:
vanishing gradients

Deep Learning

Standard RNN Gradient Flow: Vanishing Gradients



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

Many values < 1:
vanishing gradients

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

Deep Learning

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

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

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 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

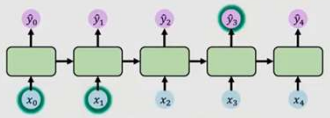
↓

Bias parameters to capture short-term dependencies

Deep Learning

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

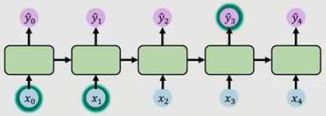
↓

Bias parameters to capture short-term dependencies

Deep Learning

The Problem of Long-Term Dependencies

"The clouds are in the ___"



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

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

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

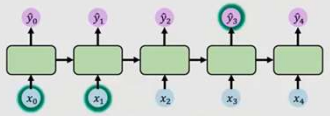
↓

Bias parameters to capture short-term dependencies

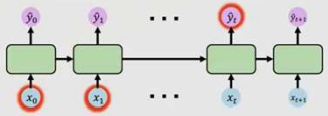
Deep Learning

The Problem of Long-Term Dependencies

"The clouds are in the ___"



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



Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

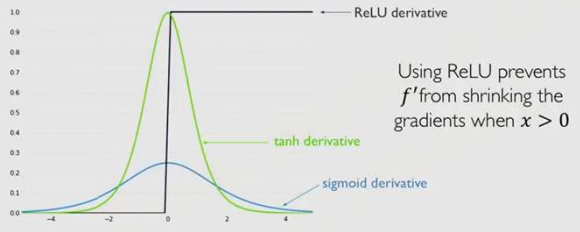
Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

Deep Learning

Trick #1: Activation Functions



Using ReLU prevents f' from shrinking the gradients when $x > 0$

Deep Learning

Trick #2: Parameter Initialization

Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

Deep Learning

Solution #3: Gated Cells

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

gated cell
LSTM, GRU, etc.

Deep Learning

Solution #3: Gated Cells

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

gated cell
LSTM, GRU, etc.

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

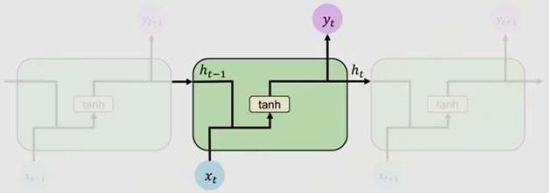
Deep Learning

Long Short Term Memory (LSTM) Networks

Deep Learning

Standard RNN

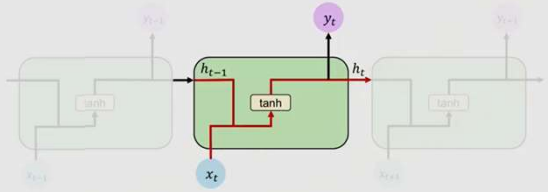
In a standard RNN, repeating modules contain a **simple computation node**



Deep Learning

Standard RNN

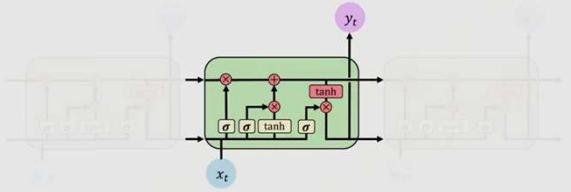
In a standard RNN, repeating modules contain a **simple computation node**



Deep Learning

Long Short Term Memory (LSTMs)

LSTM modules contain **computational blocks** that **control information flow**



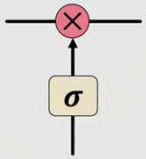
LSTM cells are able to track information throughout many timesteps

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

Deep Learning

Long Short Term Memory (LSTMs)

Information is **added** or **removed** through structures called **gates**



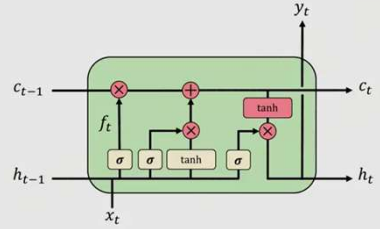
Gates optionally let information through, for example via a sigmoid neural net layer and pointwise multiplication

Deep Learning

Long Short Term Memory (LSTMs)

How do LSTMs work?

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

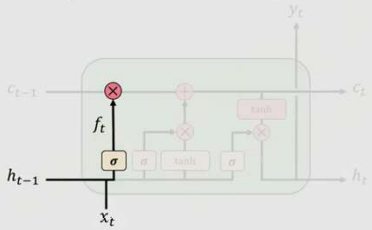


Deep Learning

Long Short Term Memory (LSTMs)

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

LSTMs **forget irrelevant** parts of the previous state

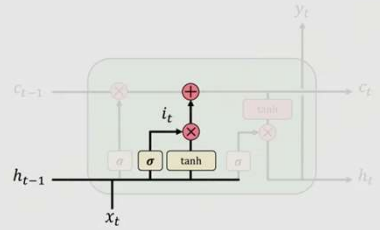


Deep Learning

Long Short Term Memory (LSTMs)

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

LSTMs **store relevant** new information into the cell state

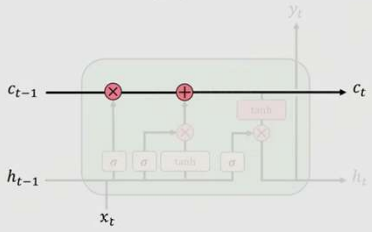


Deep Learning

Long Short Term Memory (LSTMs)

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

LSTMs **selectively update** cell state values

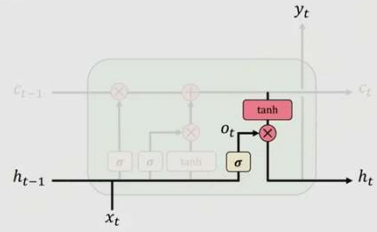


Deep Learning

Long Short Term Memory (LSTMs)

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

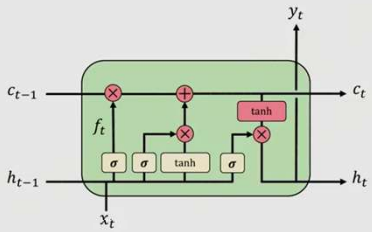
The **output gate** controls what information is sent to the next time step



Deep Learning

Long Short Term Memory (LSTMs)

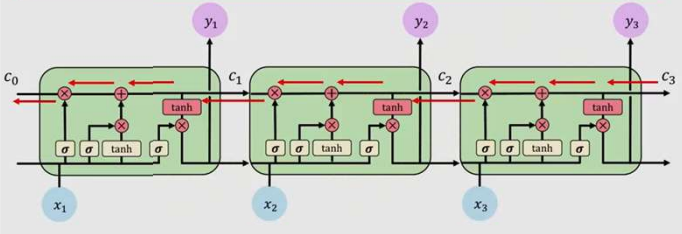
1) Forget 2) Store 3) Update 4) Output



Deep Learning

LSTM Gradient Flow

Uninterrupted gradient flow!



Deep Learning

LSTMs: Key Concepts

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
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

Deep Learning

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
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

Deep Learning

LSTMs: Key Concepts

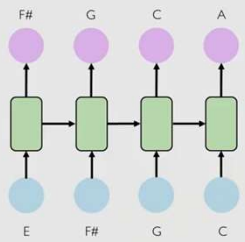
1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input
   - Selectively **update** cell state
   - **Output** gate returns a filtered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

Deep Learning

RNN Applications

Deep Learning

Example Task: Music Generation

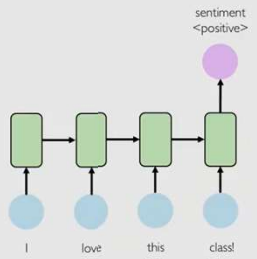


Input: sheet music

Output: next character in sheet music

Deep Learning

Example Task: Sentiment Classification



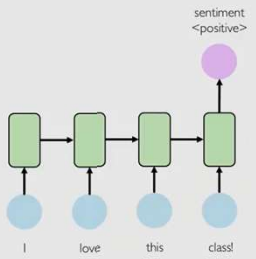
Input: sequence of words

Output: probability of having positive sentiment

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


Deep Learning

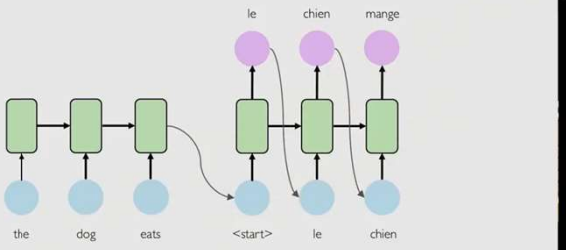
Example Task: Sentiment Classification



Tweet sentiment classification

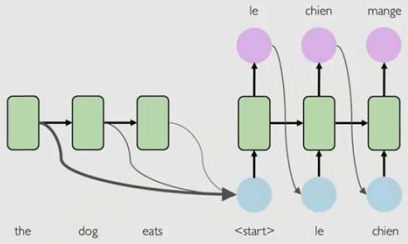
Deep Learning

Example Task: Machine Translation



Deep Learning

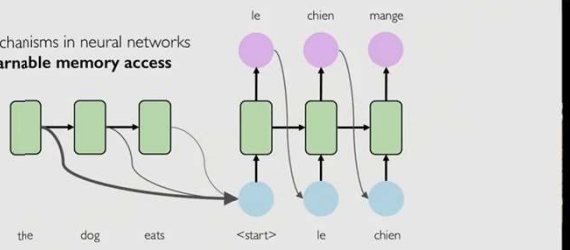
Attention Mechanisms



Deep Learning

Attention Mechanisms

Attention mechanisms in neural networks provide **learnable memory access**



Deep Learning

Deep Learning for Sequence Modeling: Summary

1. RNNs are well suited for **sequence modeling** tasks

Deep Learning

Deep Learning for Sequence Modeling: Summary

1. RNNs are well suited for **sequence modeling** tasks
2. Model sequences via a **recurrence relation**

Deep Learning

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**

Deep Learning

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. Gated cells like **LSTMs** let us model **long-term dependencies**

Deep Learning

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. Gated cells like **LSTMs** let us model **long-term dependencies**
5. Models for **music generation**, classification, machine translation, and more

Deep Learning

THANK YOU!