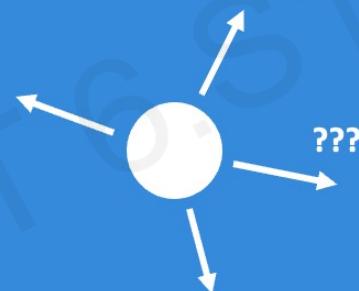


Given an image of a ball,
can you predict where it will go next?



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



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



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



Sequences in the Wild

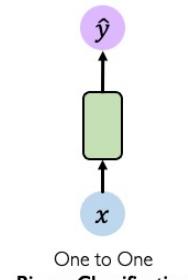


Audio

Sequences in the Wild



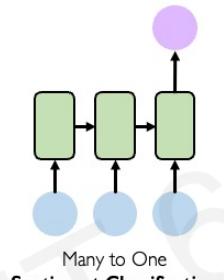
Sequence Modeling Applications



One to One
Binary Classification



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



Many to One
Sentiment Classification

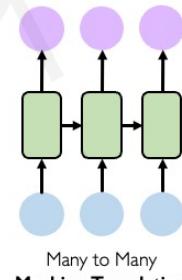
Ivar Hagedoorn
[@IvarHagedoorn](https://twitter.com/IvarHagedoorn)
Follow
The @MIT Introduction to #DeepLearning is definitely one of the best courses of its kind currently available online
introtodeeplearning.com

12:40 PM - 12 Feb 2019



"A baseball player throws a ball."

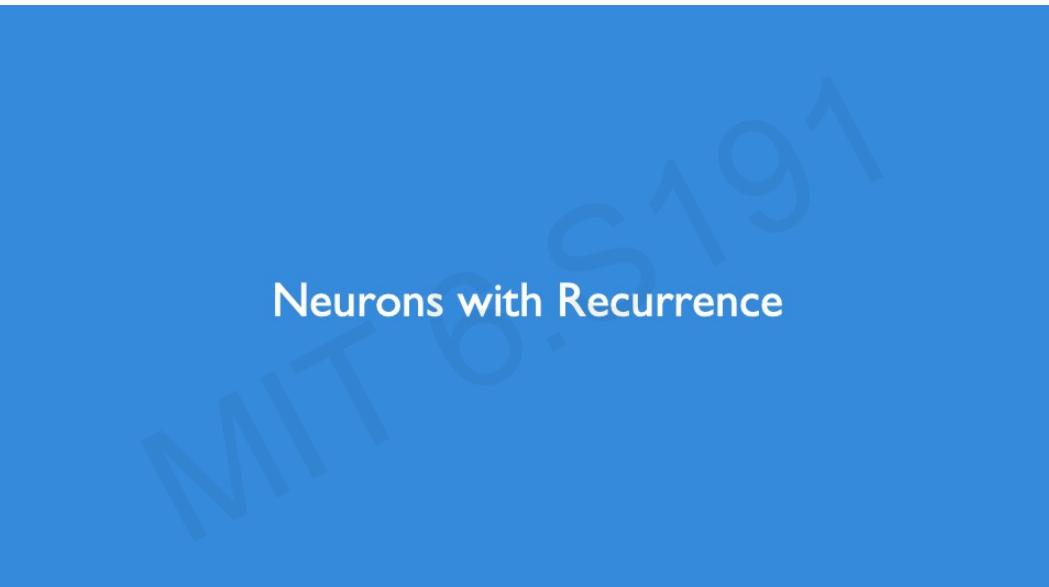
One to Many
Image Captioning



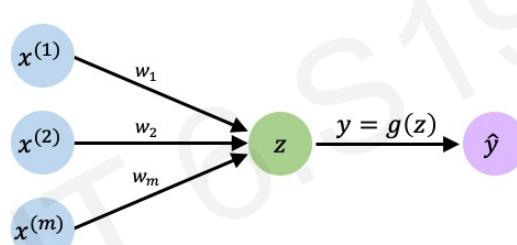
Many to Many
Machine Translation



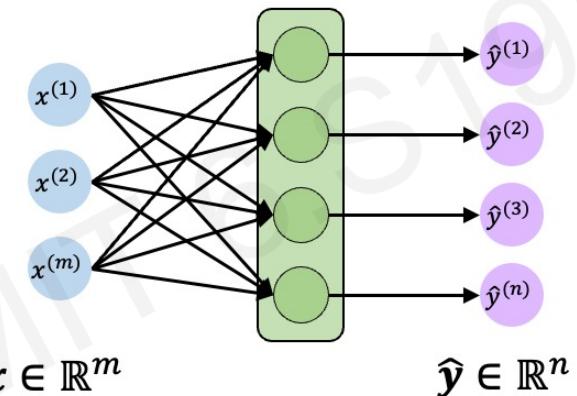
Neurons with Recurrence



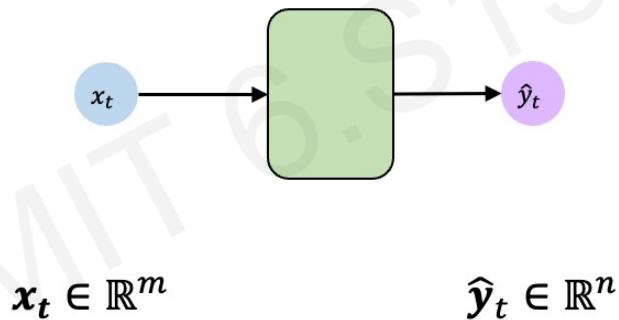
The Perceptron Revisited



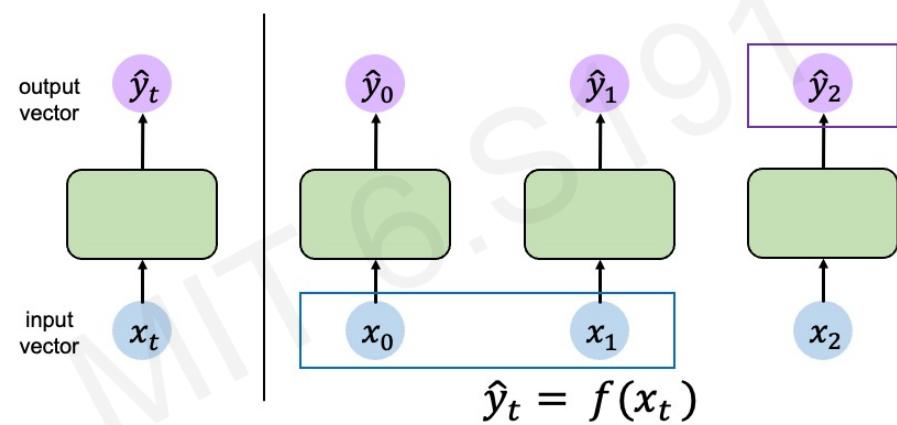
Feed-Forward Networks Revisited



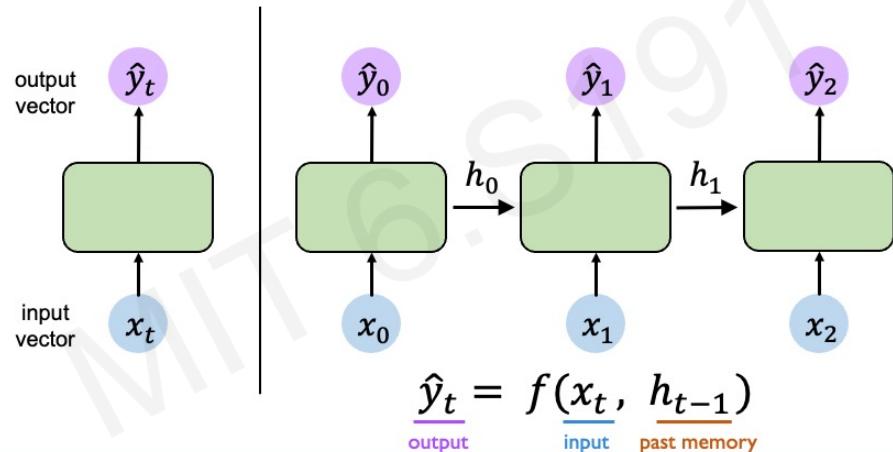
Feed-Forward Networks Revisited



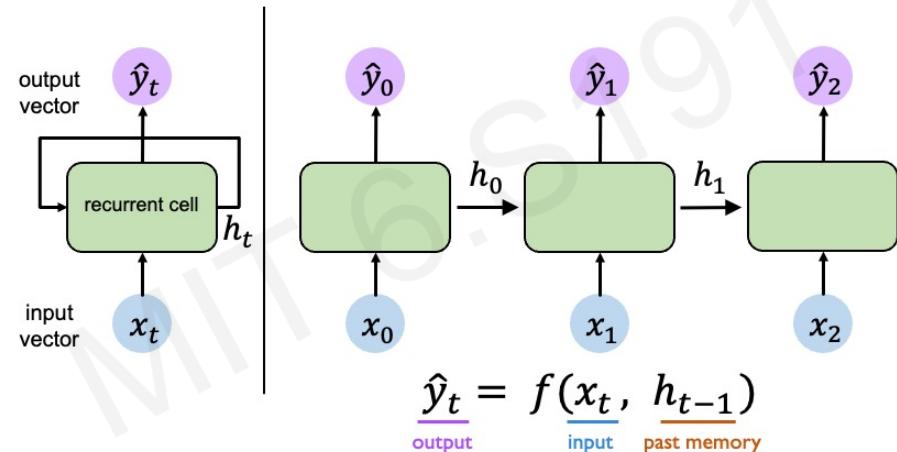
Handling Individual Time Steps



Neurons with Recurrence

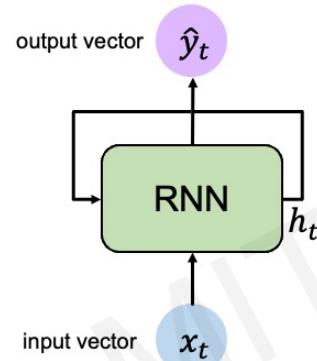


Neurons with Recurrence



Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs)



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

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

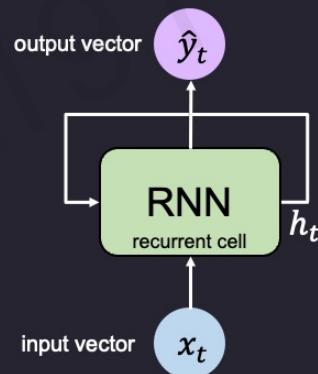
cell state function with weights w input old state

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

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

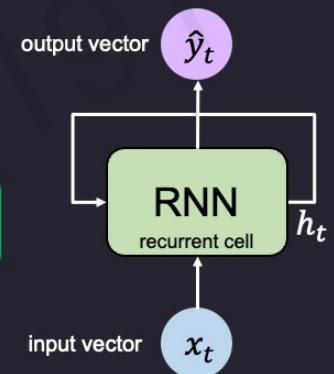
RNN Intuition

```
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!"
```



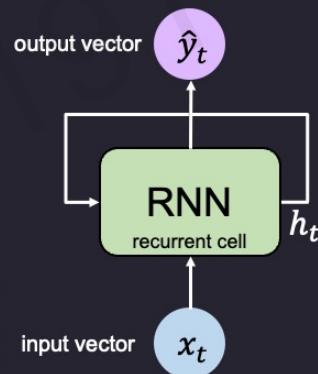
RNN Intuition

```
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!"
```

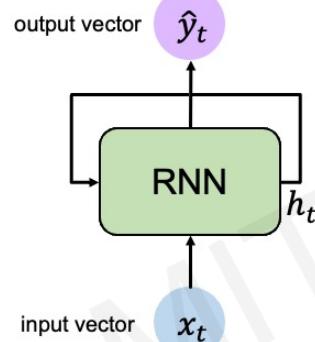


RNN Intuition

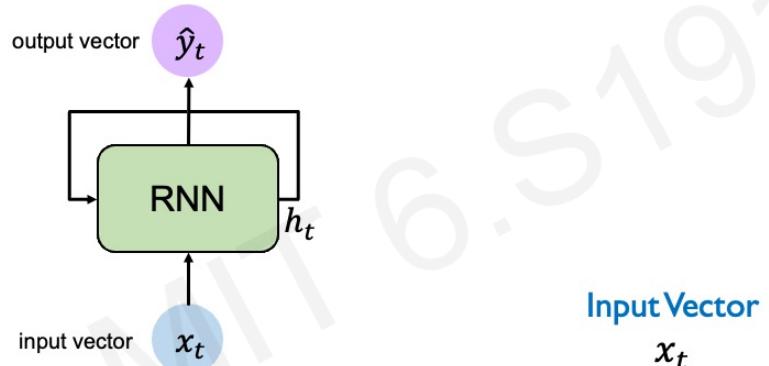
```
my_rnn = RNN()  
hidden_state = [0, 0, 0, 0]  
  
sentence = ["I", "love", "recurrent", "neural"]  
  
for word in sentence:  
    prediction, hidden_state = my_rnn(word, hidden_state)  
  
next_word_prediction = prediction  
# >>> "networks!"
```



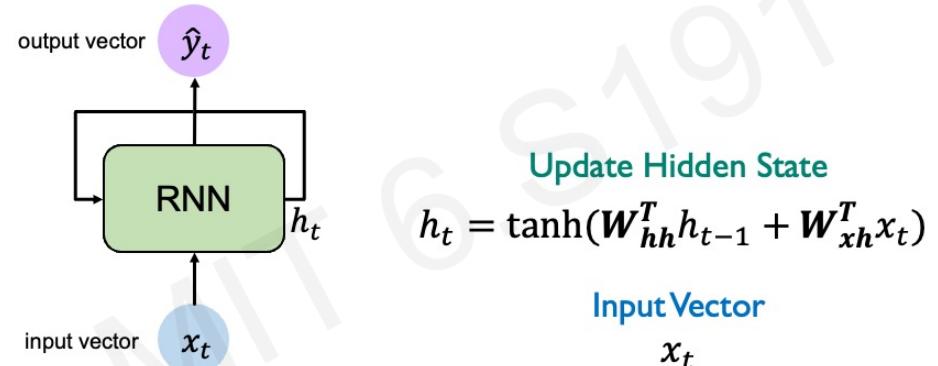
RNN State Update and Output



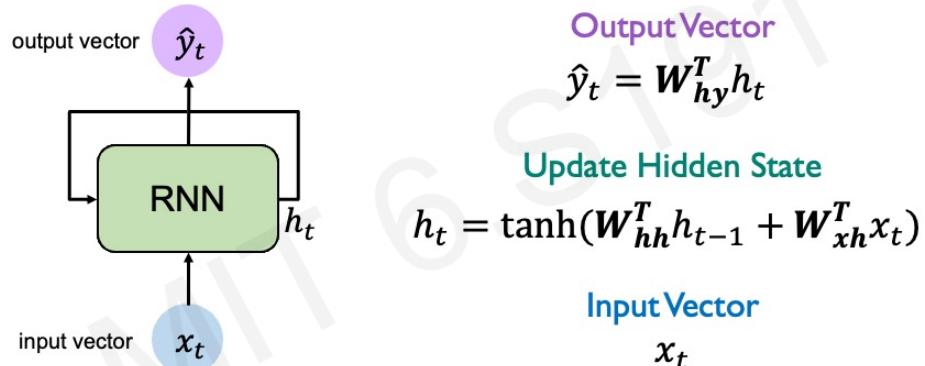
RNN State Update and Output



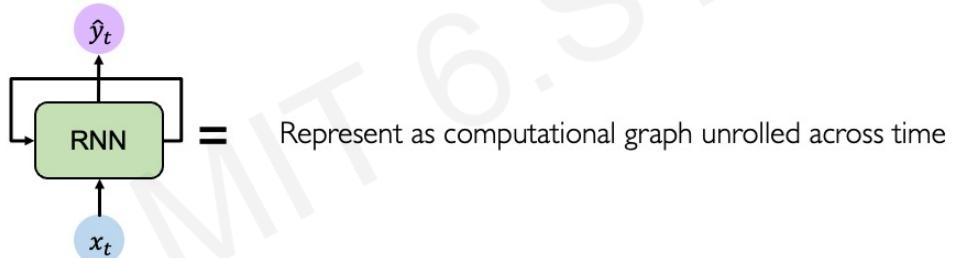
RNN State Update and Output



RNN State Update and Output

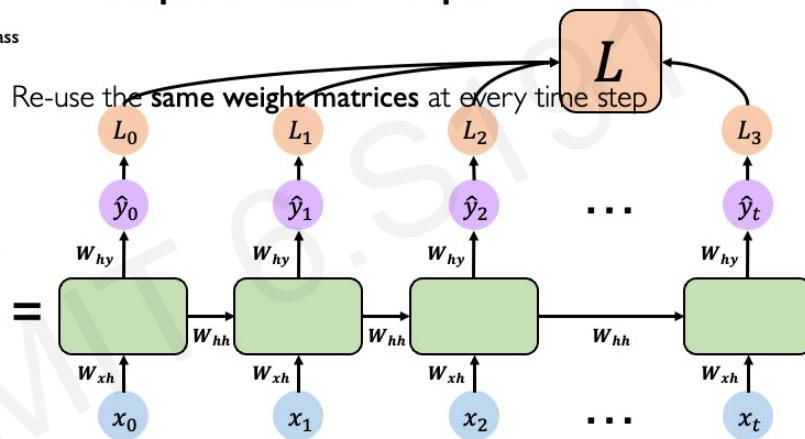


RNNs: Computational Graph Across Time

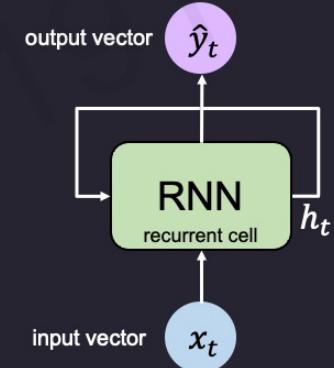


RNNs: Computational Graph Across Time

→ Forward pass



RNNs from Scratch in TensorFlow



```
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
```

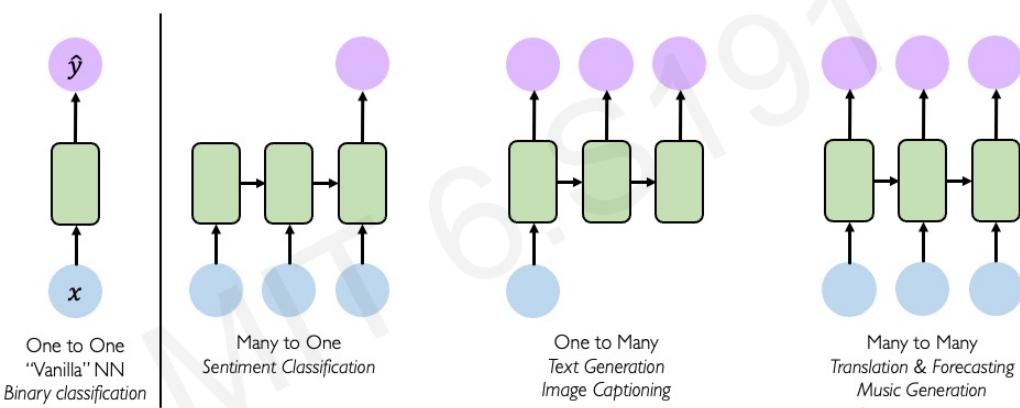
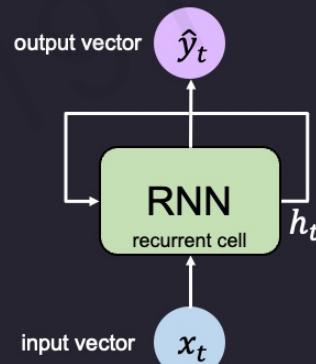


RNN Implementation: TensorFlow & PyTorch

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

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

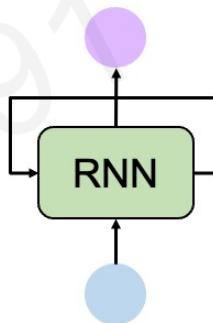
```



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

A Sequence Modeling Problem: Predict the Next Word

“This morning I took my cat for a walk.”

“This morning I took my cat for a walk.”
given these words

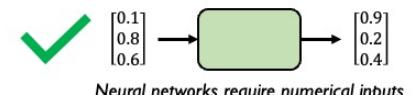
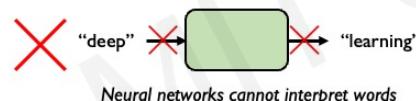
A Sequence Modeling Problem: Predict the Next Word

A Sequence Modeling Problem: Predict the Next Word

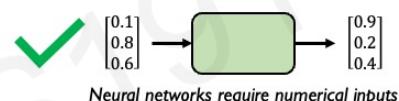
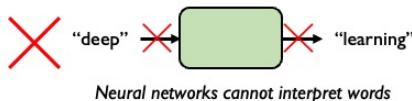
"This morning I took my cat for a walk."
given these words predict the next word

"This morning I took my cat for a walk."
given these words predict the next word

Representing Language to a Neural Network



Encoding Language for a Neural Network

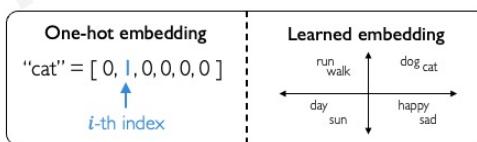


Embedding: transform indexes into a vector of fixed size.

this	cat	for
my	took	
a	I	walk

a	→ 1
cat	→ 2
...	...
walk	→ N

1. Vocabulary:
Corpus of words



3. Embedding:
Index to fixed-sized vector

Handle Variable Sequence Lengths

The food was great

vs.

We visited a restaurant for lunch

vs.

We were hungry but cleaned the house before eating

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.

Capture Differences in Sequence Order



The food was good, not bad at all.

vs.

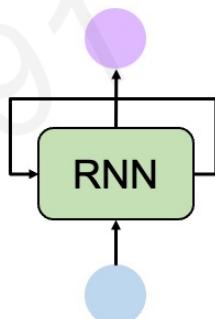


The food was bad, not good at all.

Sequence Modeling: Design Criteria

To model sequences, we need to:

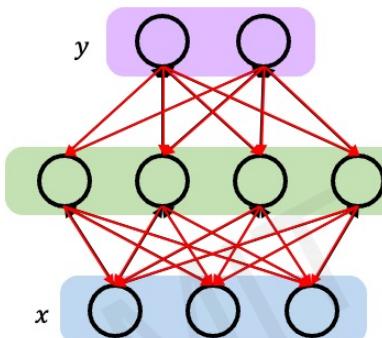
1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. Share **parameters** across the sequence



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

Backpropagation Through Time (BPTT)

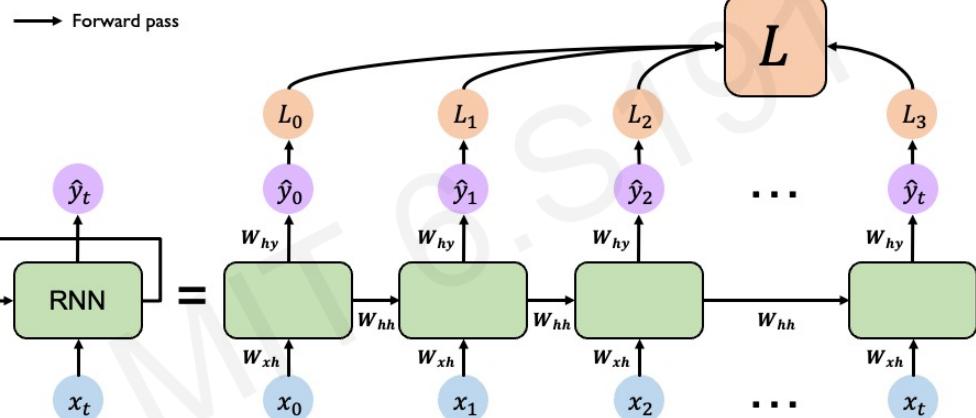
Recall: Backpropagation in Feed Forward Models



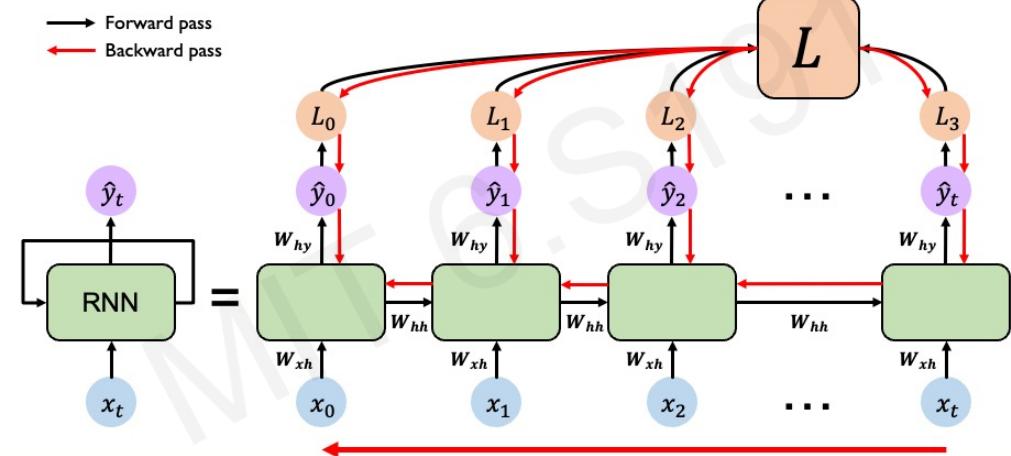
Backpropagation algorithm:

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

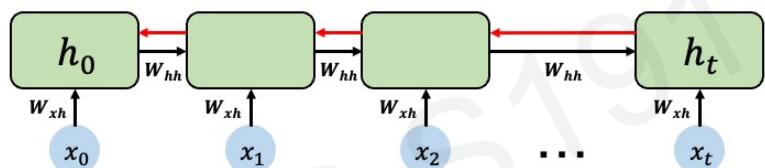
RNNs: Backpropagation Through Time



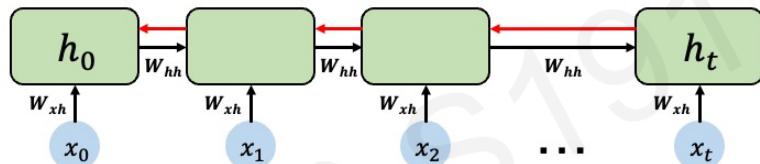
RNNs: Backpropagation Through Time



Standard RNN Gradient Flow

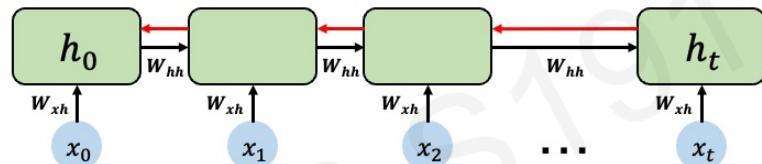


Standard RNN Gradient Flow



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

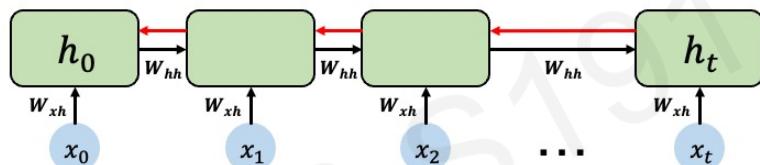
Standard RNN Gradient Flow: Exploding Gradients



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

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

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 ___"

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

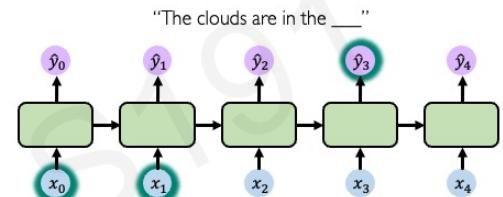
Multiply many **small numbers** together



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

Why are vanishing gradients a problem?

Multiply many **small numbers** together

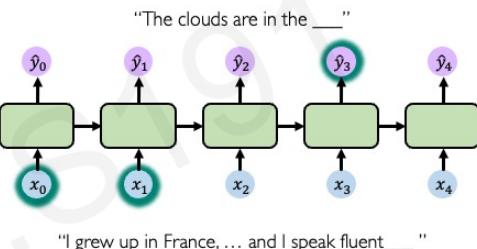


Errors due to further back time steps have smaller and smaller gradients



Bias parameters to capture short-term dependencies

"The clouds are in the ___"



The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

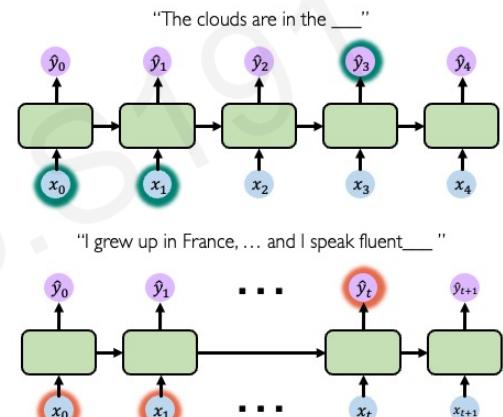
Multiply many **small numbers** together



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

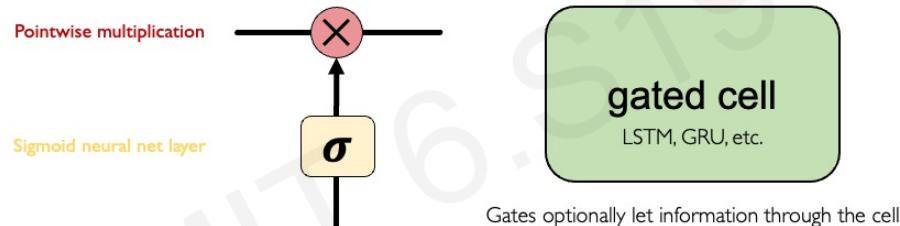


Bias parameters to capture short-term dependencies



Gating Mechanisms in Neurons

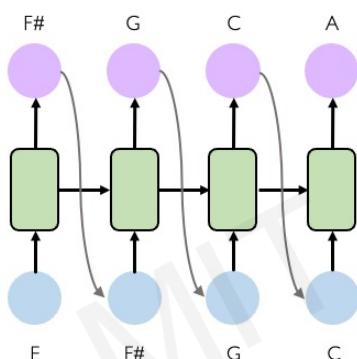
Idea: use **gates** to selectively **add or remove** information within **each recurrent unit with**



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

RNN Applications & Limitations

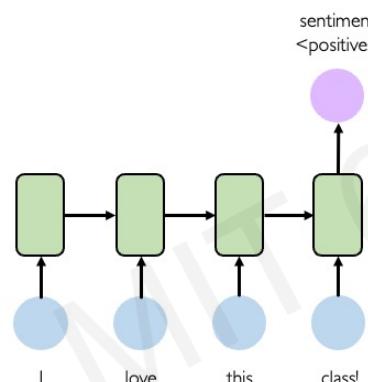
Example Task: Music Generation



Input: sheet music
Output: next character in sheet music



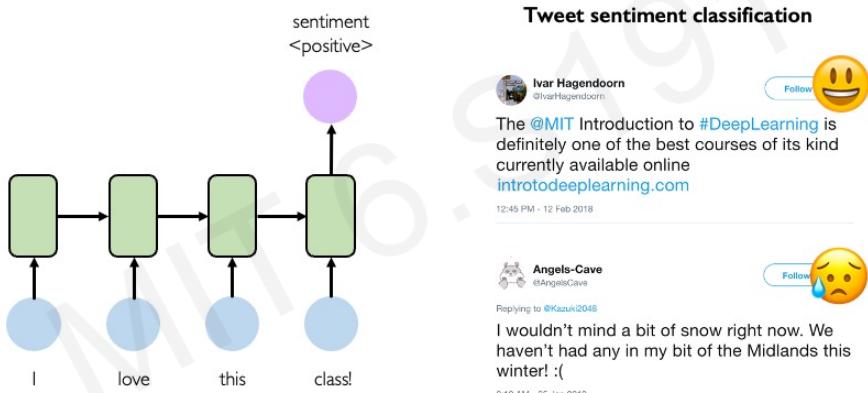
Example Task: Sentiment Classification



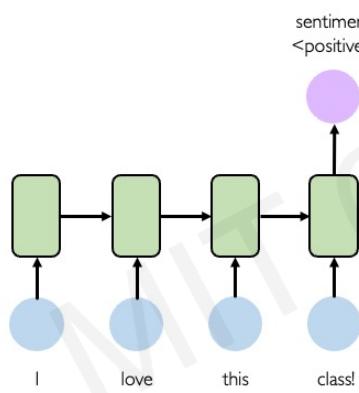
Input: sequence of words
Output: probability of having positive sentiment

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

Example Task: Sentiment Classification



Limitations of Recurrent Models

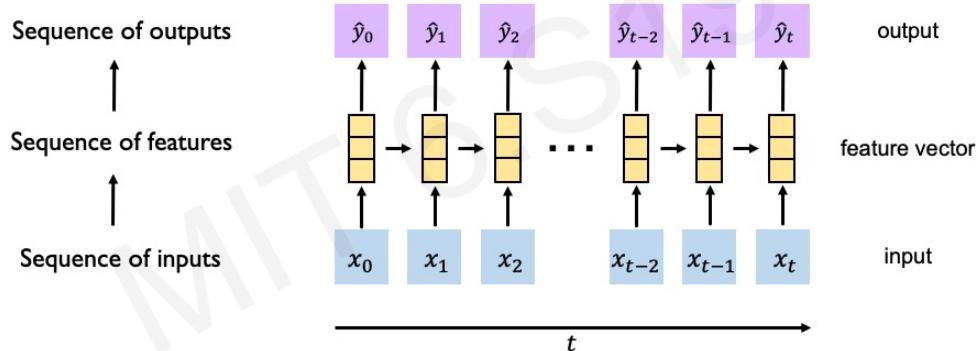


Limitations of RNNs

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

Goal of Sequence Modeling

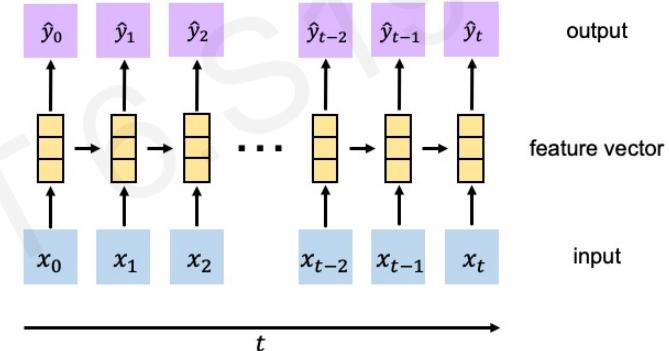
RNNs: recurrence to model sequence dependencies



Goal of Sequence Modeling

RNNs: recurrence to model sequence dependencies

- ### Limitations of RNNs
- ⌚ Encoding bottleneck
 - ⌚ Slow, no parallelization
 - 🧠 Not long memory



Goal of Sequence Modeling

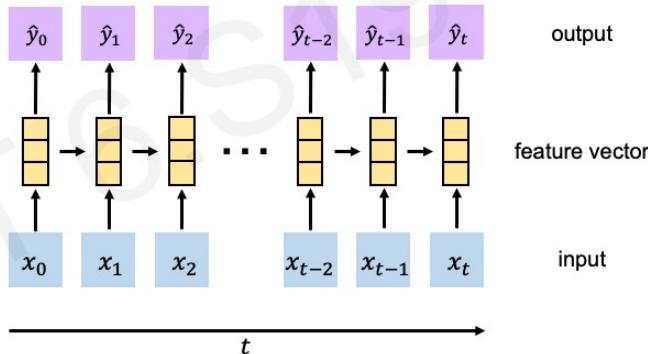
Can we eliminate the need for recurrence entirely?

Desired Capabilities

Continuous stream

Parallelization

Long memory



Goal of Sequence Modeling

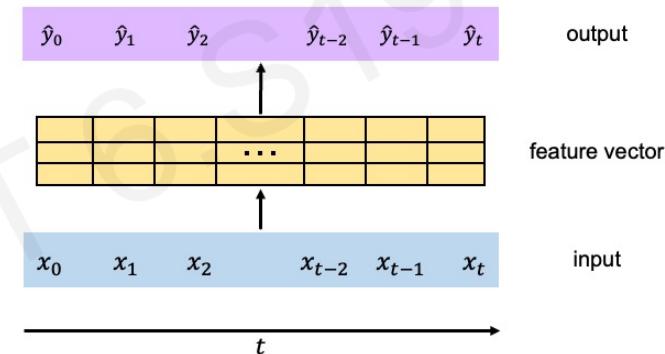
Can we eliminate the need for recurrence entirely?

Desired Capabilities

Continuous stream

Parallelization

Long memory



Goal of Sequence Modeling

Can we eliminate the need for recurrence entirely?

Idea I: Feed everything into dense network

✓ No recurrence

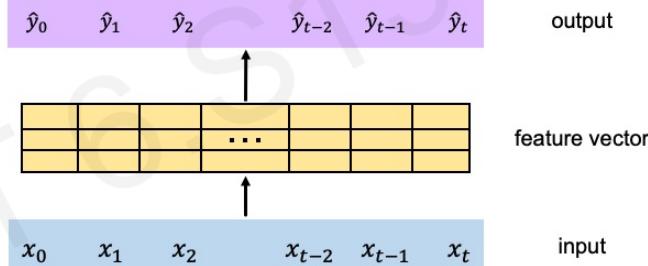
✗ Not scalable

✗ No order

✗ No long memory



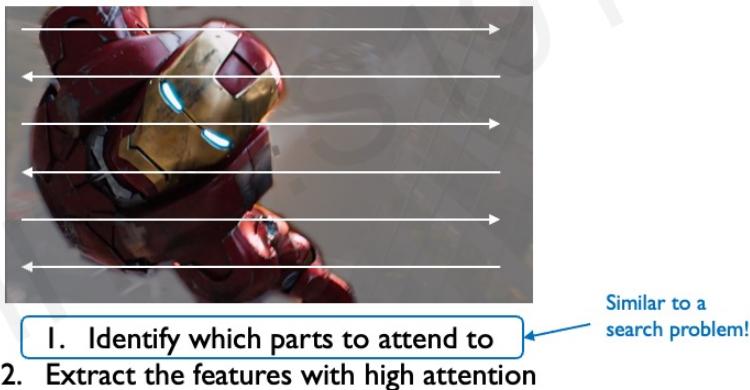
Idea: Identify and attend to what's important



Attention Is All You Need

Intuition Behind Self-Attention

Attending to the most important parts of an input.

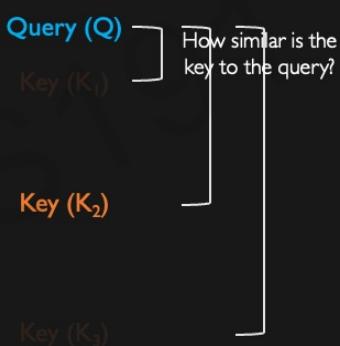


A Simple Example: Search



Understanding Attention with Search

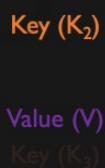
A screenshot of a YouTube search results page for "deep learning". The top result is a video titled "MIT 6.S191 (2020): Introduction to Deep Learning" by Alexander Amini. The video thumbnail shows a man speaking in front of a network diagram. Below the main result are other video thumbnails, including one for "The Kobe Bryant Fadeaway Shot".



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

Understanding Attention with Search

A screenshot of a YouTube search results page for "deep learning". The top result is a video titled "MIT 6.S191 (2020): Introduction to Deep Learning" by Alexander Amini. The video thumbnail shows a man speaking in front of a network diagram. Below the main result are other video thumbnails, including one for "The Kobe Bryant Fadeaway Shot".



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

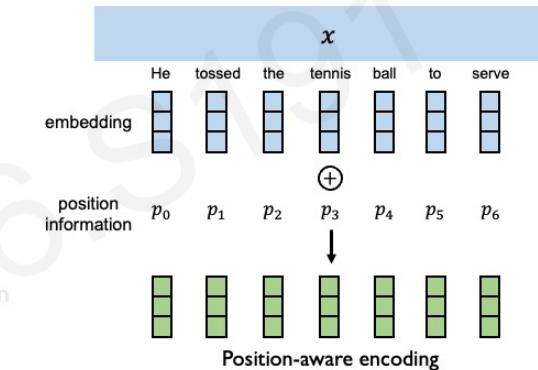


Data is fed in all at once! Need to encode position information to understand order.

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

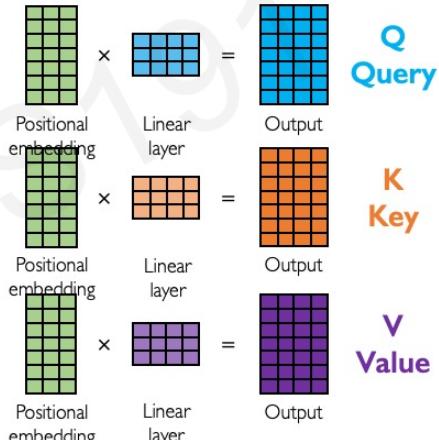


Data is fed in all at once! Need to encode position information to understand order.

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



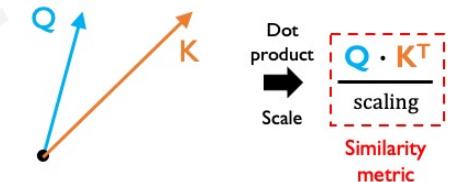
Learning Self-Attention with Neural Networks

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?



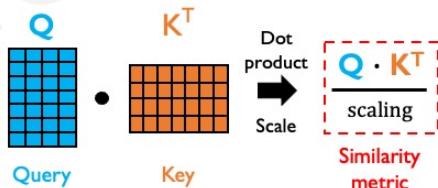
Learning Self-Attention with Neural Networks

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?



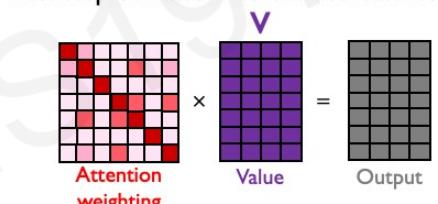
Also known as the "cosine similarity"

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



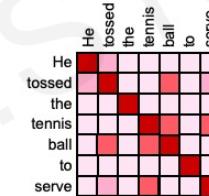
$$\text{softmax}\left(\frac{Q \cdot K^T}{\text{scaling}}\right) \cdot V = A(Q, K, V)$$

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?



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

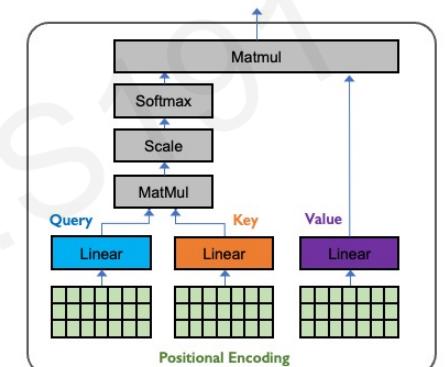
Attention weighting

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.

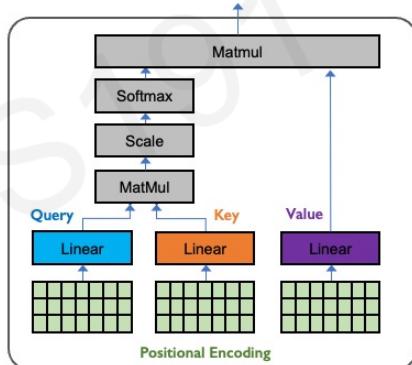


$$\text{softmax}\left(\frac{Q \cdot K^T}{\text{scaling}}\right) \cdot V$$

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 Transformer architecture.

Applying Multiple Self-Attention Heads



Self-Attention Applied

Language Processing



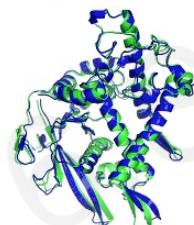
An armchair in the shape of an avocado

Transformers: BERT, GPT

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

★ 6.S191 Lab and Lectures!

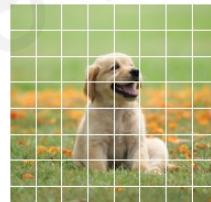
Biological Sequences



Protein Structure Models

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

Computer Vision



Vision Transformers

Dosovitskiy et al., ICLR 2020

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!

