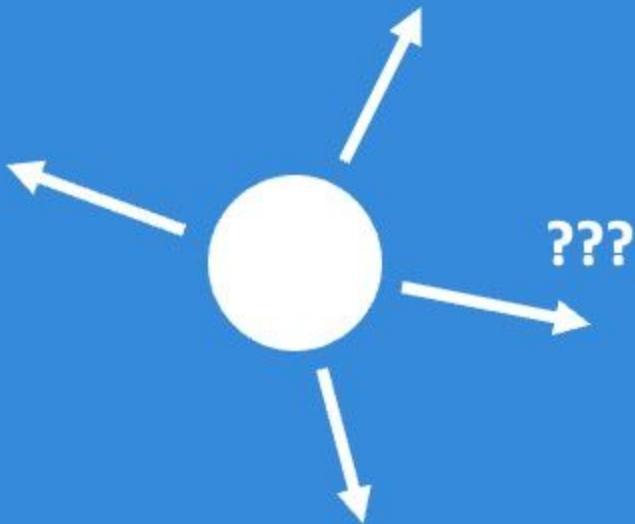


Recurrent Neural Networks, Transformers, and Attention

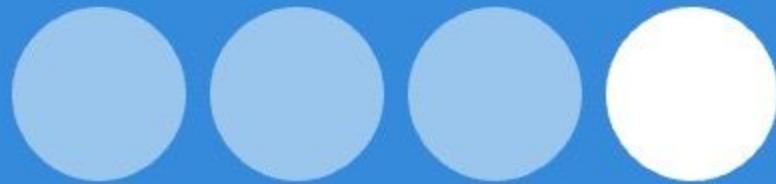
Prof. Dr. Patrick Terrematte (IMD / UFRN)

- Deep sequence modeling
- Intuition and Applications of RNNs
- Drawbacks of RNNs
- Self-Attention and Transformers

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



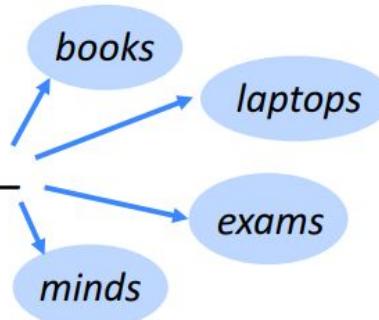
**Given an image of a ball,
can you predict where it will go next?**



Language Modeling

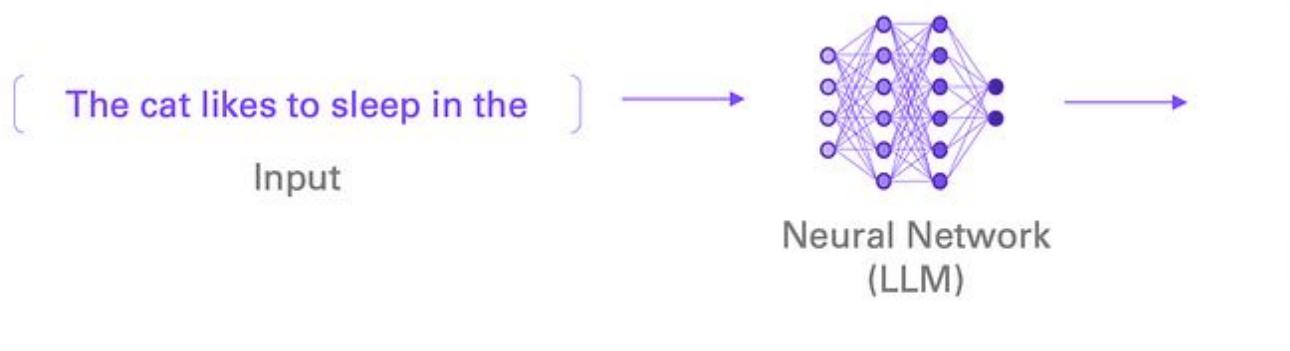
Language modeling is the task of predicting what word comes next.

the students opened their _____



⌚ Language Modeling

Language modeling is the task of predicting what word comes next.



Word	Probability
ability	0.002
bag	0.071
box	0.085
...	...
zebra	0.001

Output

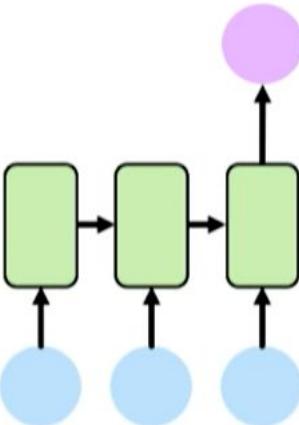
Sequence Modeling



One to One
Binary Classification



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



Many to One
Sentiment Classification

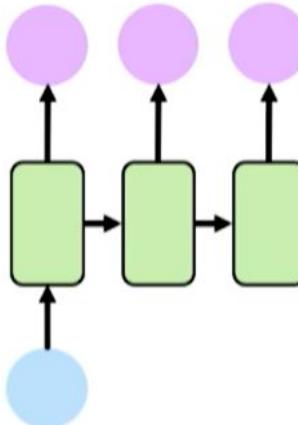


Ivar Hagendoorn
@IvarHagendoorn



The @MIT Introduction to #DeepLearning is definitely one of the best courses of its kind currently available online introtodeeplearning.com

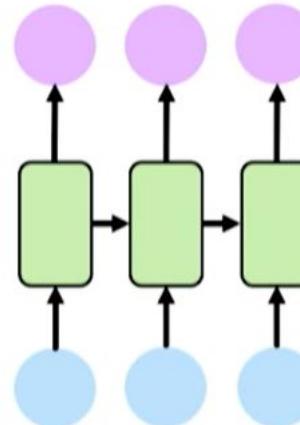
12:45 PM - 12 Feb 2018



One to Many
Image Captioning



"A baseball player throws a ball."



Many to Many
Machine Translation



⌚ Attention Timeline

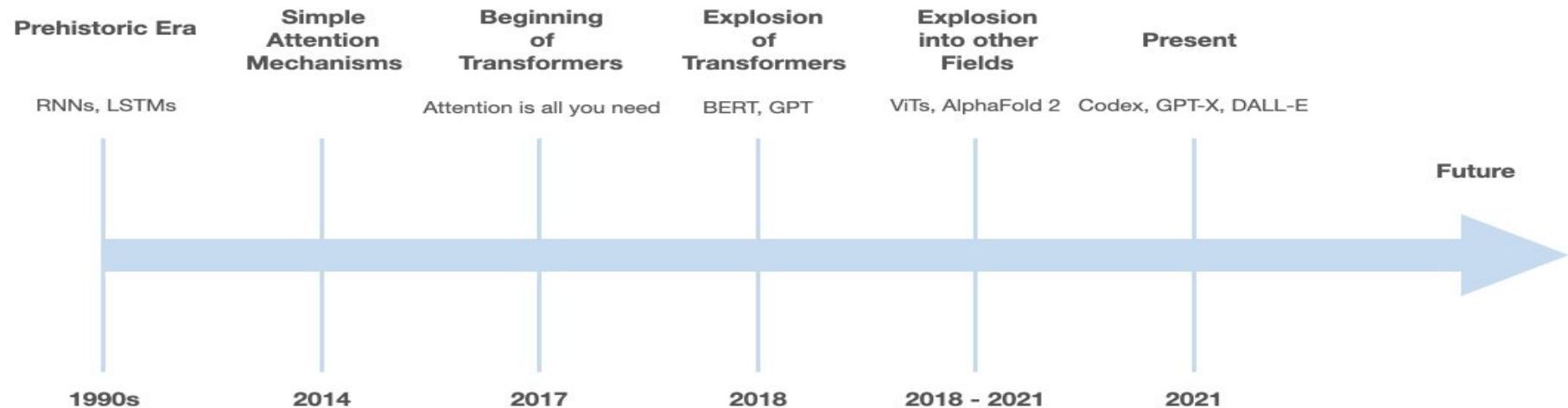
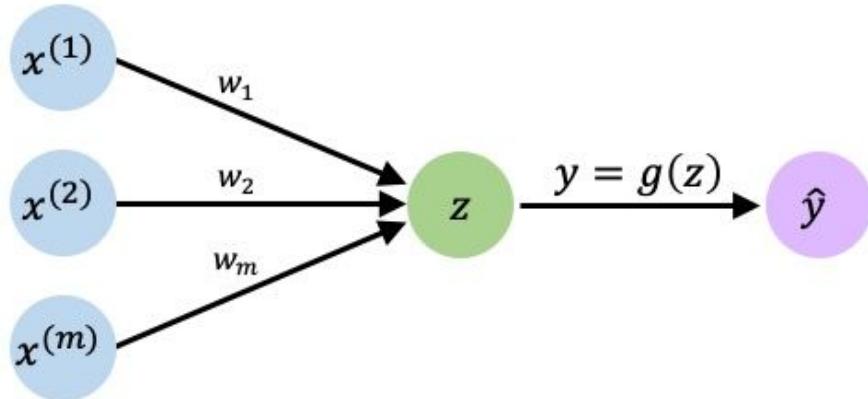
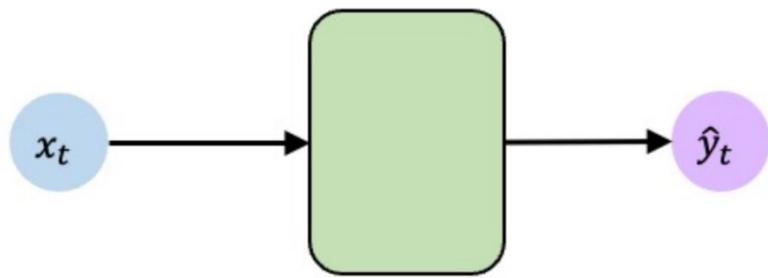


Figure adapted from [Transformer United Course](#) by Stanford.

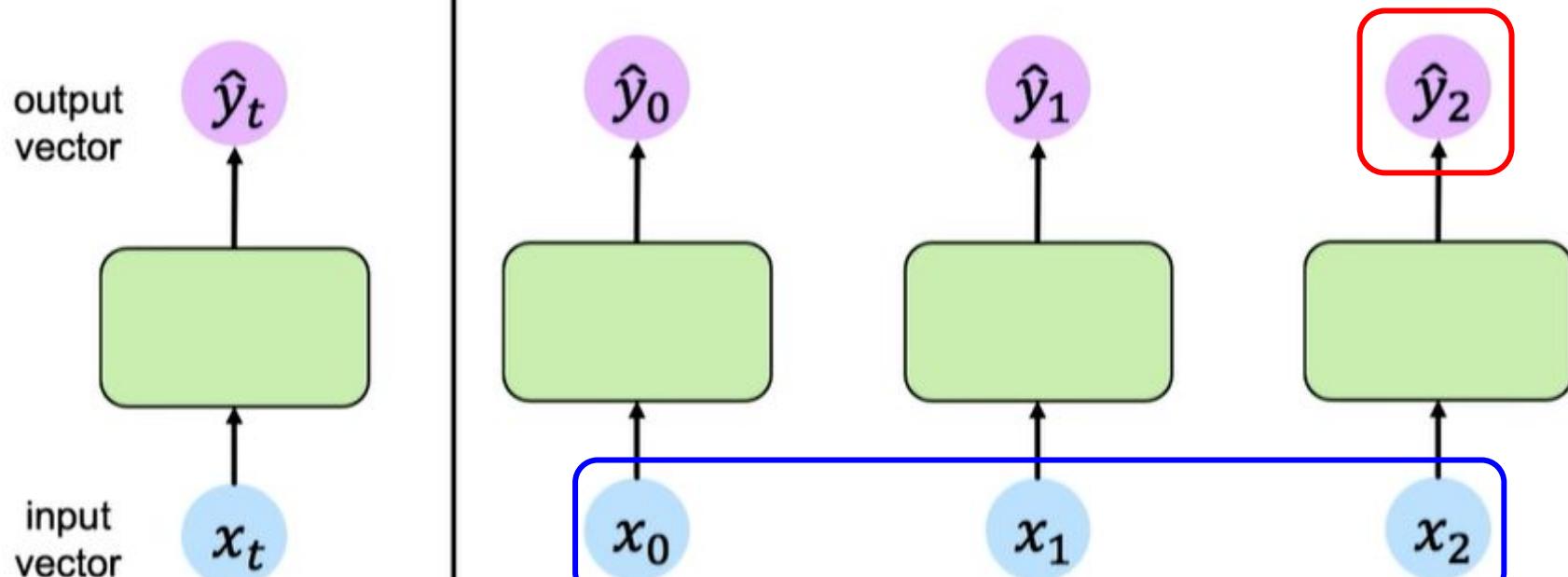




$$\boldsymbol{x}_t \in \mathbb{R}^m$$

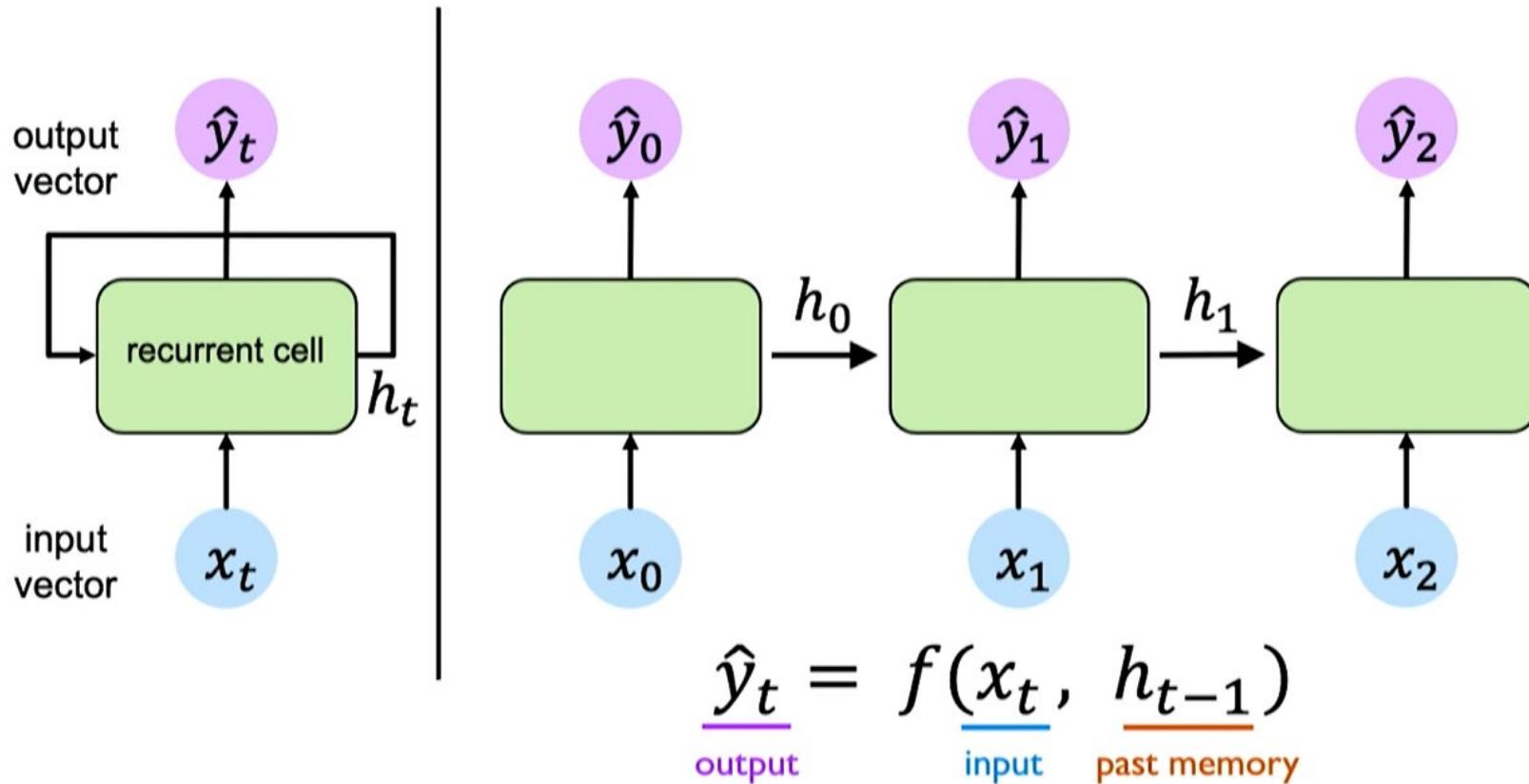
$$\hat{\boldsymbol{y}}_t \in \mathbb{R}^n$$

⌚ Time steps



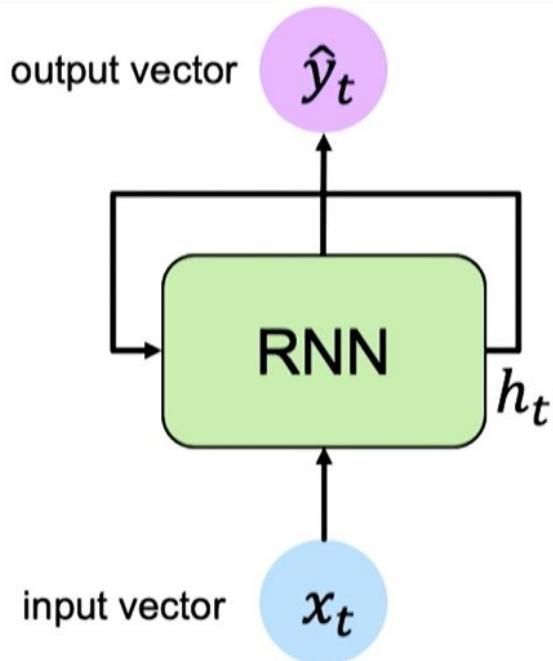
$$\hat{y}_t = f(x_t)$$

⟳ Neuron with Recurrence



Recurrent Neural Networks (RNNs)

Recurrent Neural Networks



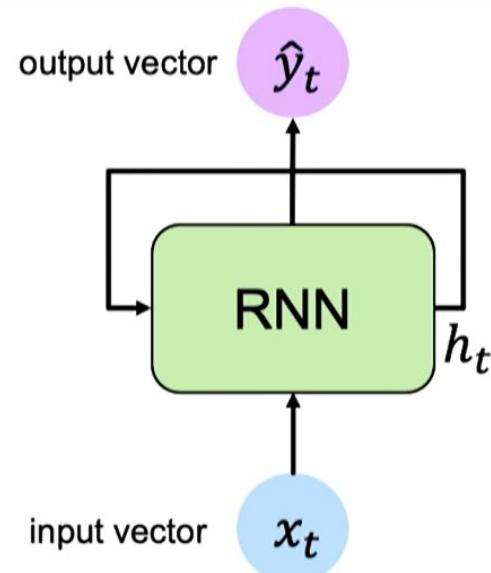
$$h_t = f_W(x_t, h_{t-1})$$

cell state function with weights W
 input old state

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!"
```

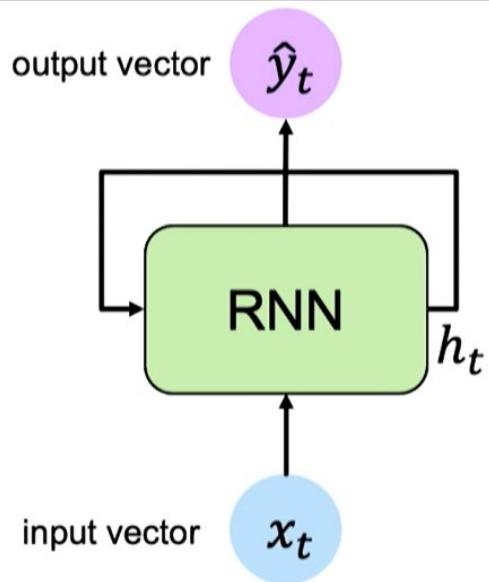


Perplexity measures language model quality

- "It's raining outside"
- "It's raining bananas"
- "It's raining piouw;kcj pwepoiut"

- Inverse of the geometric mean of the number of actual choices we have when deciding which token to choose next.
 - Best case, the model always perfectly estimates the probability of the target token as 1.
 - Worst case, the model always predicts the probability of the target token as 0. In this situation, the perplexity is positive infinity.
- If the model predicts a uniform distribution over all available tokens in the vocabulary, the perplexity is equal to the number of unique tokens in the vocabulary.

$$\exp \left(-\frac{1}{n} \sum_{t=1}^n \log P(x_t \mid x_{t-1}, \dots, x_1) \right)$$



Output Vector

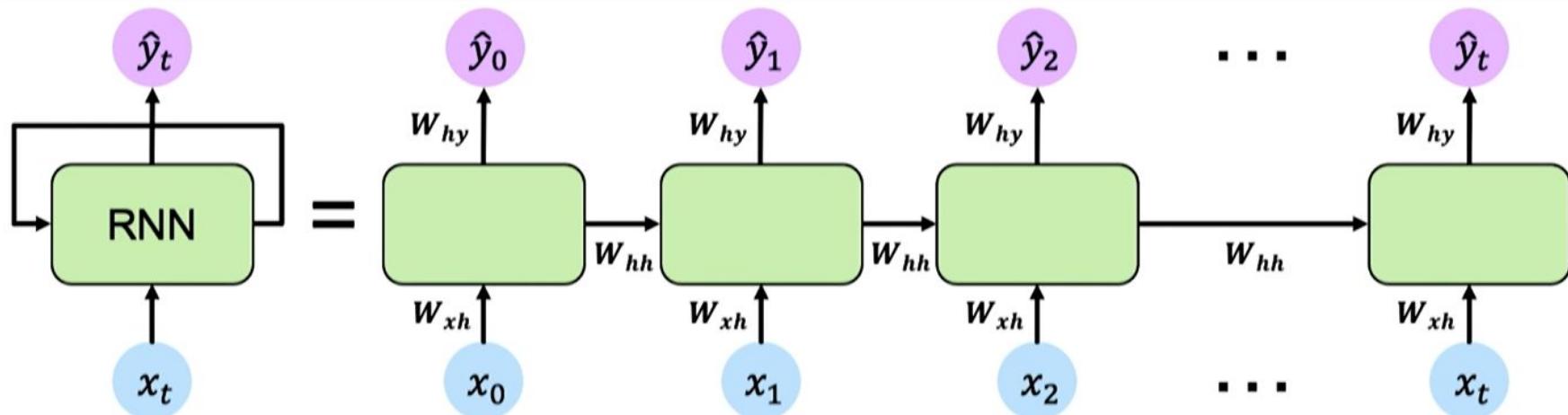
$$\hat{y}_t = \mathbf{W}_{hy}^T h_t$$

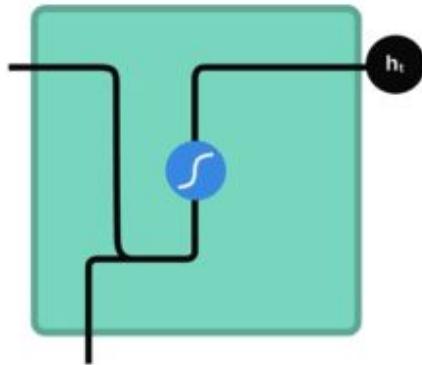
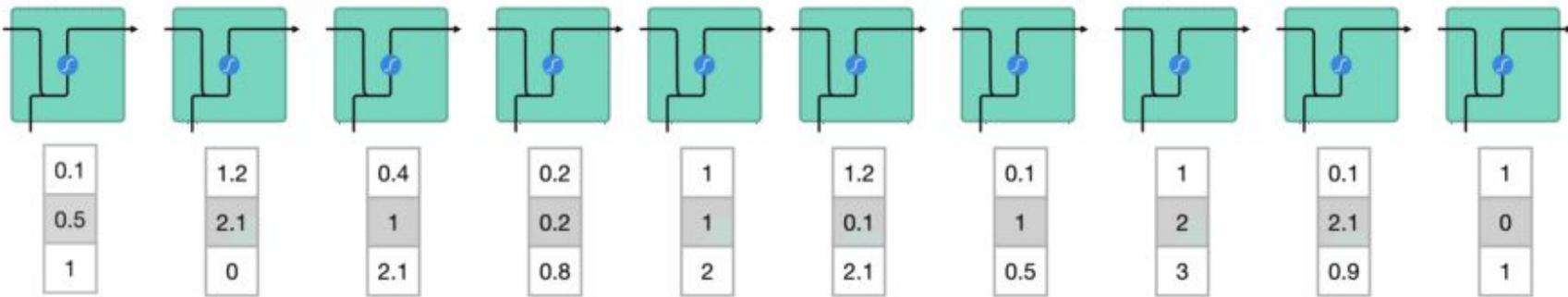
Update Hidden State

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

Input Vector

$$x_t$$

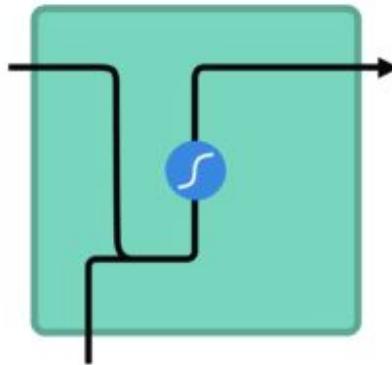


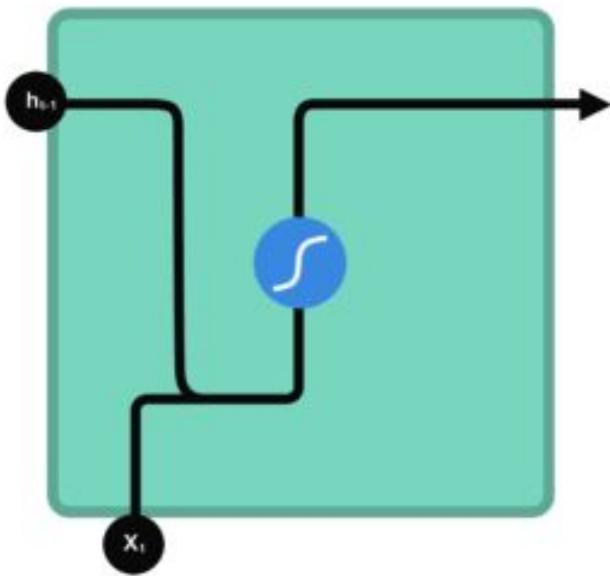


Tanh function



hidden state (memory)





Tanh function



new hidden state



previous hidden state

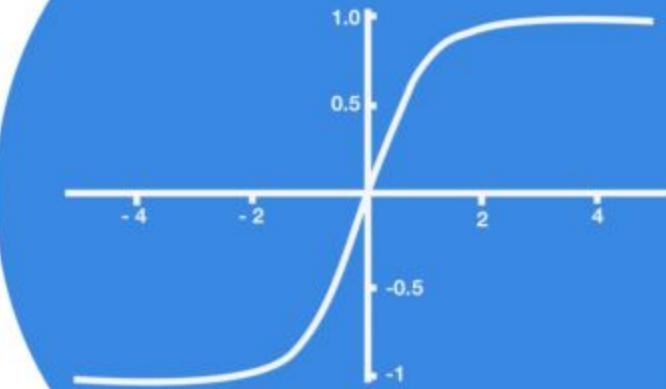


input

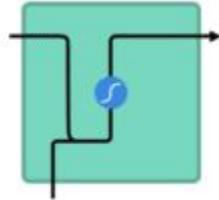
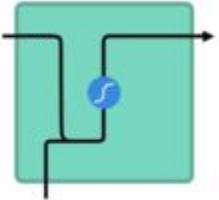
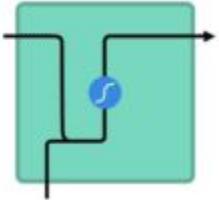
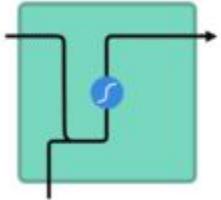


concatenation

5
0.1
-0.5



5
0.01
-0.5



```

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])

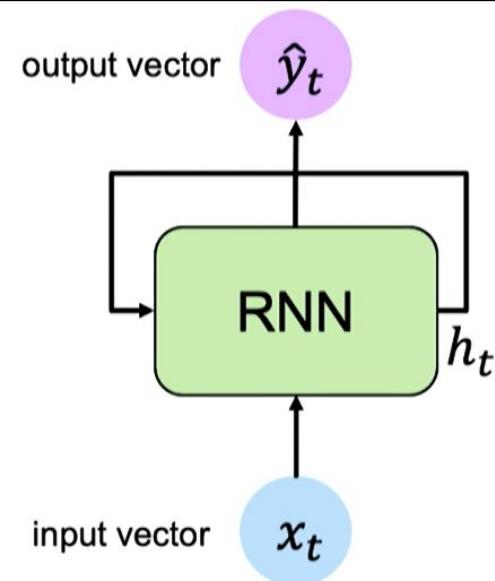
        # Initialize hidden state to zeros
        self.h = tf.zeros([rnn_units, 1])

    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

```

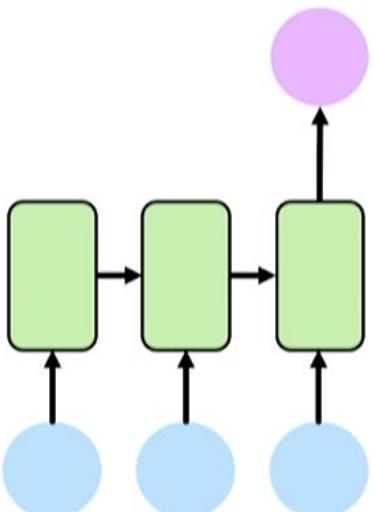


`tf.keras.layers.SimpleRNN(rnn_units)`

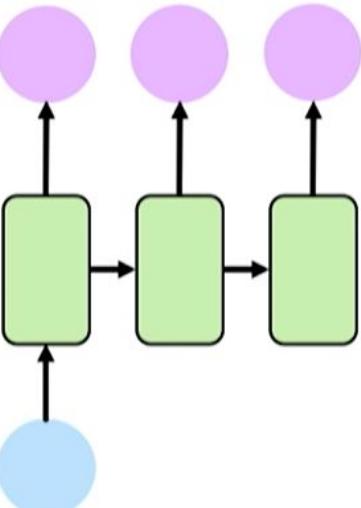
Sequence Modeling



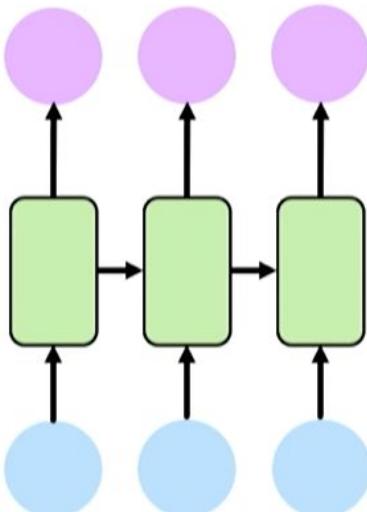
One to One
"Vanilla" NN
Binary classification



Many to One
Sentiment Classification



One to Many
Text Generation
Image Captioning

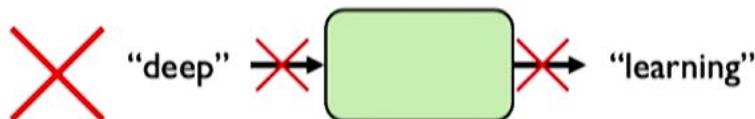


Many to Many
Translation & Forecasting
Music Generation

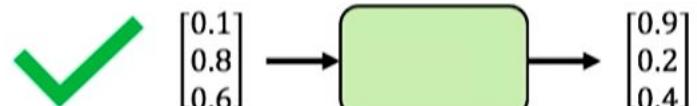
1. Handle variable-length sequences
2. Track long-term dependencies
3. Maintain information about order
4. Share parameters across the sequence
5. Capture differences in Sequence Order.



Representing Language to a Neural Network



Neural networks cannot interpret words



Neural networks require numerical inputs

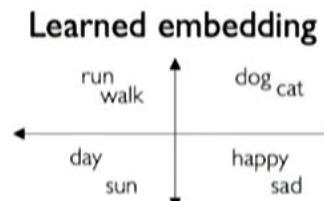
Embedding

Transform indexes into a vector of fixed size.

this cat for
my took I
a I walk
morning

a → 1
cat → 2
... ...
walk → N

One-hot embedding
"cat" = [0, 1, 0, 0, 0, 0]
i-th index

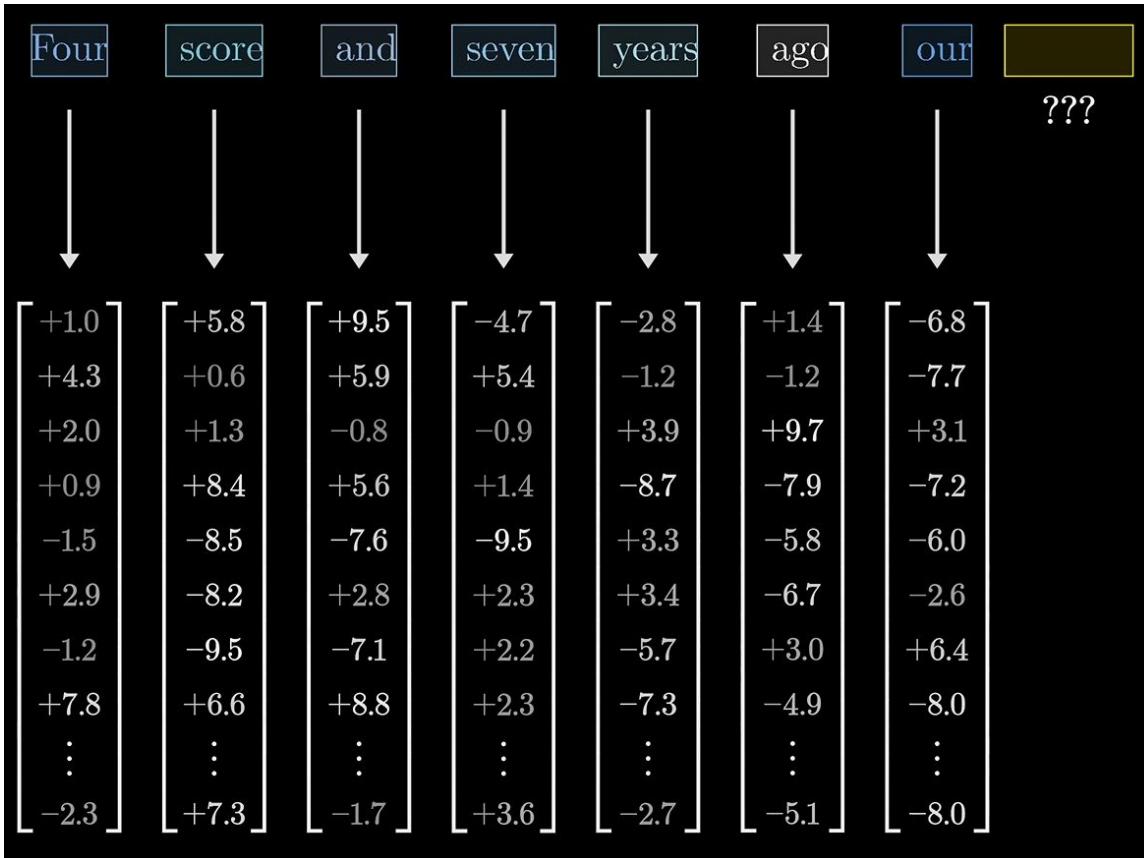
Learned embedding


1. Vocabulary:
Corpus of words

2. Indexing:
Word to index

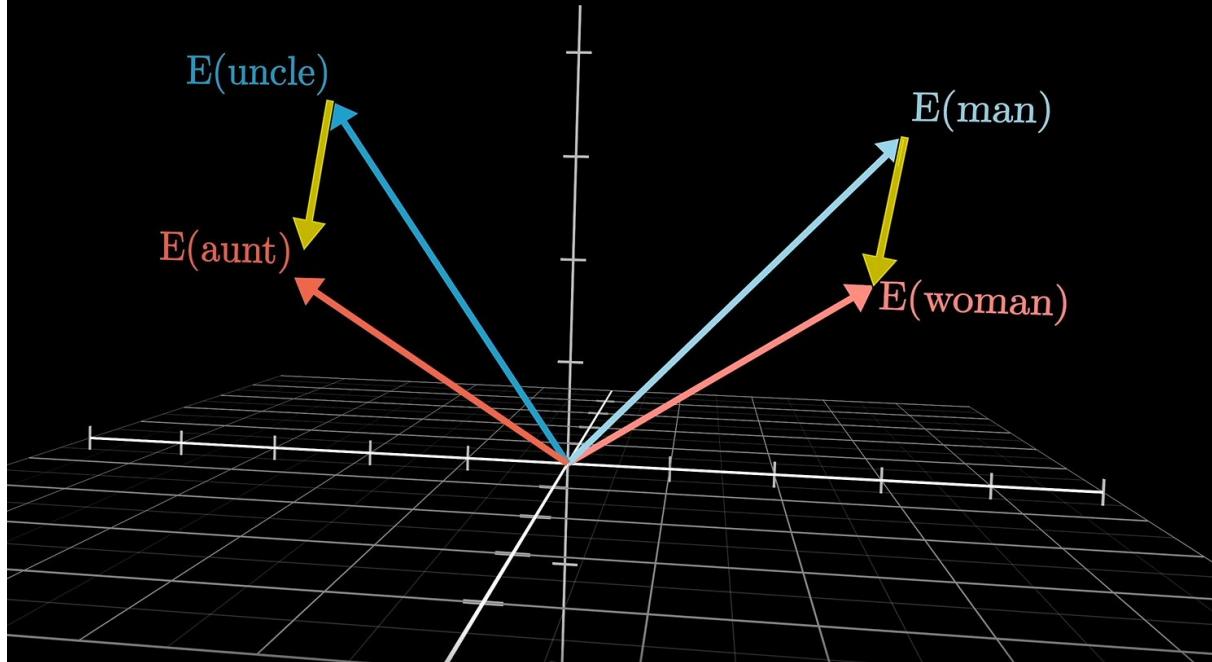
3. Embedding:
Index to fixed-sized vector

Embedding



Embedding

$$E(\text{aunt}) - E(\text{uncle}) \approx E(\text{woman}) - E(\text{man})$$



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

Short distance

One mole of carbon dioxide



Long distance

Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework but was forced to do it in secret, in the dead of night. And he also happened to be a wizard.

It was nearly midnight, and he was lying on his stomach in bed, the blankets drawn right over his head like a tent, a flashlight in one hand and a large leather-bound book (*A History of Magic* by Bathilda Bagshot) propped open against the pillow. Harry shoved the tip of his eagle-feather quill down the page, frowning as he looked for something that would help him write his essay, "Witch Burning in the Fourteenth Century Was Completely Pointless discuss."

The quill paused at the top of a likely-looking paragraph. Harry pushed his round glasses up the bridge of his nose, moved his flashlight closer to the book, and read:

Non-magic people (more commonly known as Muggles) were particularly afraid of magic in medieval times, but not very good at recognizing it. On the rare occasion that they did catch a real witch or wizard, burning had no effect whatsoever. The witch or wizard would perform a basic Flame Freezing Charmer and then pretend to shriek with pain while enjoying a gentle, tickling sensation. Indeed, Wendelin the Weird enjoyed being burned so much that she allowed herself to be caught no less than fortyseven times in various disguises.

Harry put his quill between his teeth and reached underneath his pillow for his ink bottle and a roll of parchment. Slowly and very carefully he unscrewed the ink bottle, dipped his quill into it, and began to write, pausing every now and then to listen, because if any of the Dursleys heard the scratching of his quill on their way to the bathroom, he'd probably find himself locked in the cupboard under the stairs for the rest of the summer.

The Dursley family of number four, Privet Drive, was the reason that Harry never enjoyed his summer holidays. Uncle Vernon, Aunt Petunia, and their son, Dudley, were Harry's only living relatives. They were Muggles, and they had a very medieval attitude toward magic. Harry's dead parents, who had been a witch and a wizard themselves, were never mentioned under the Dursleys' roof for years. Aunt Petunia and Uncle Vernon had hoped that if they kept Harry as bewitched as possible, they would be able to squash the magic out of him. And, to their fury, they had been unsuccessful. These days they lived in terror of anyone finding out that Harry had spent most of the last two years at Hogwarts School of Witchcraft and Wizardry. The most they could do, however, was to lock away Harry's spellbooks, wand, cauldron, and broomstick at the start of the summer break, and forbid him to talk to the neighbors.

Capture differences in Sequence Order.



The food was good, not bad at all.

vs.

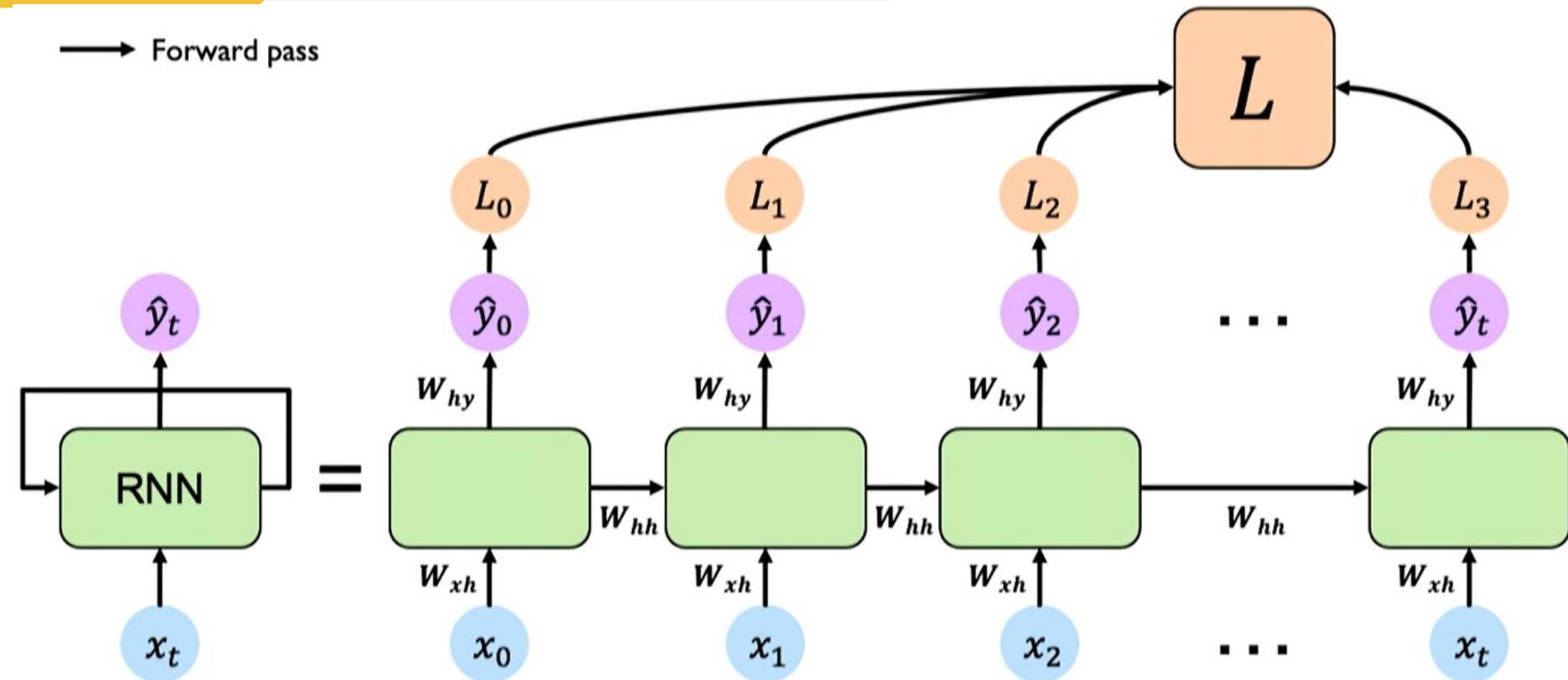
The food was bad, not good at all.



Backpropagation Through Time

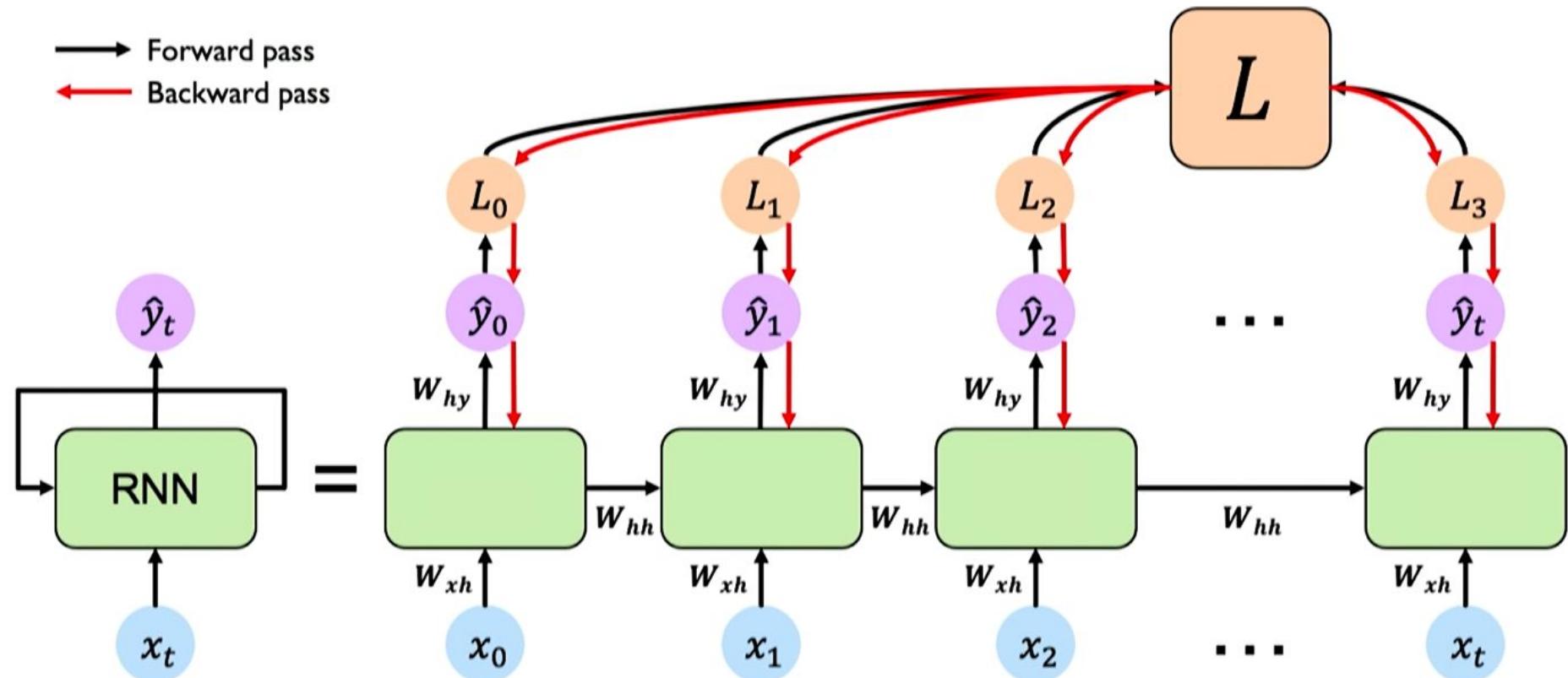
Backpropagation Through Time

→ Forward pass



Backpropagation Through Time

→ Forward pass
← Backward pass



Exploding and Vanishing

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

Why are vanishing gradients a problem?

Multiply many **small numbers** together



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

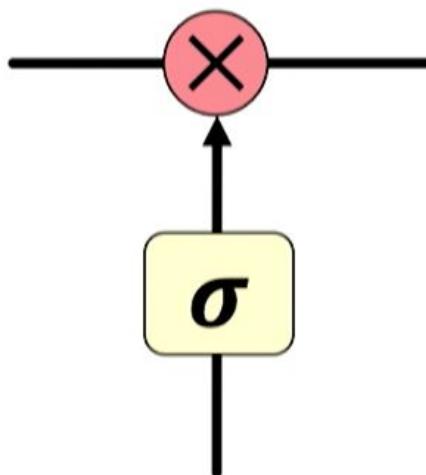


Bias parameters to capture short-term
dependencies

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

Pointwise multiplication

Sigmoid neural net layer

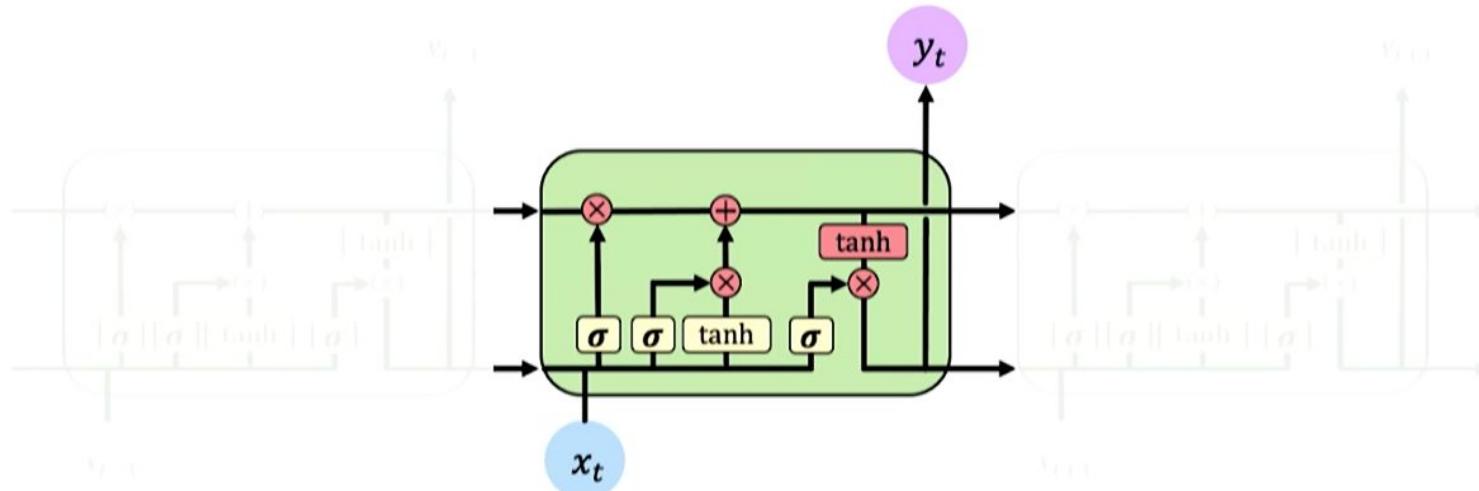


gated cell
LSTM, GRU, etc.

Gates optionally let information through the cell

Gated LSTM cells **control information flow**:

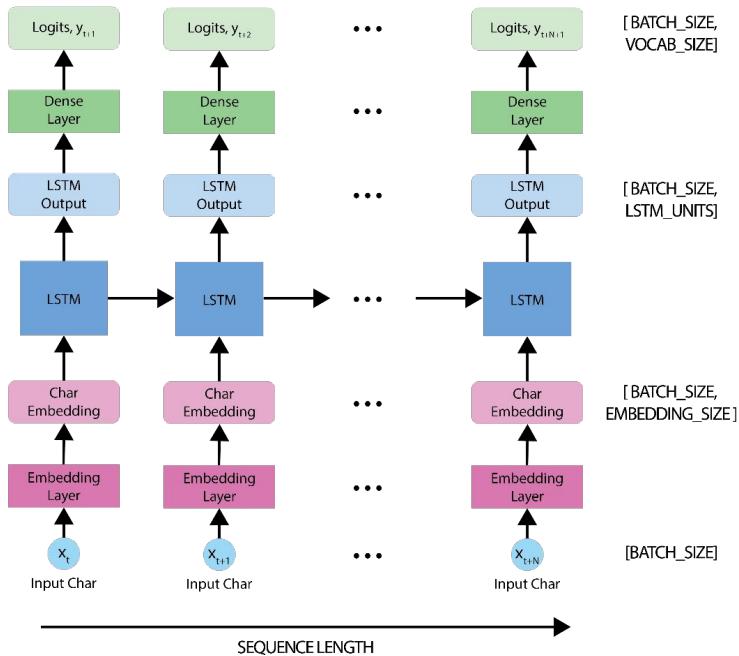
- 1) Forget
- 2) Store
- 3) Update
- 4) Output

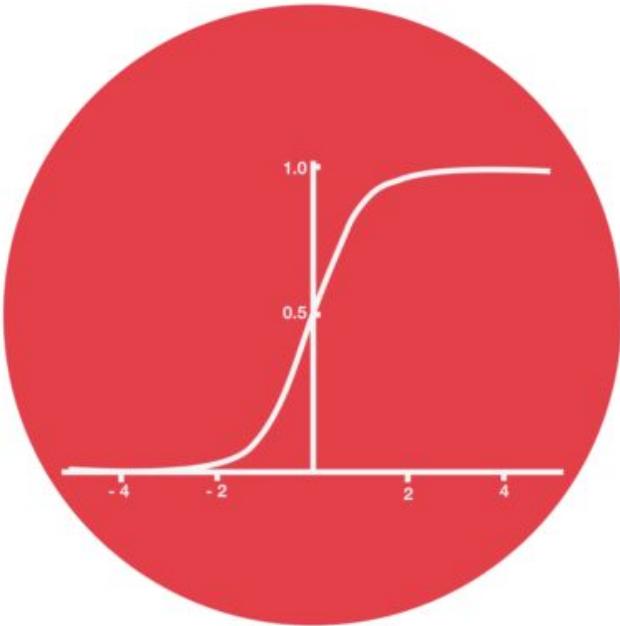


LSTM cells are able to track information throughout many timesteps

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

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



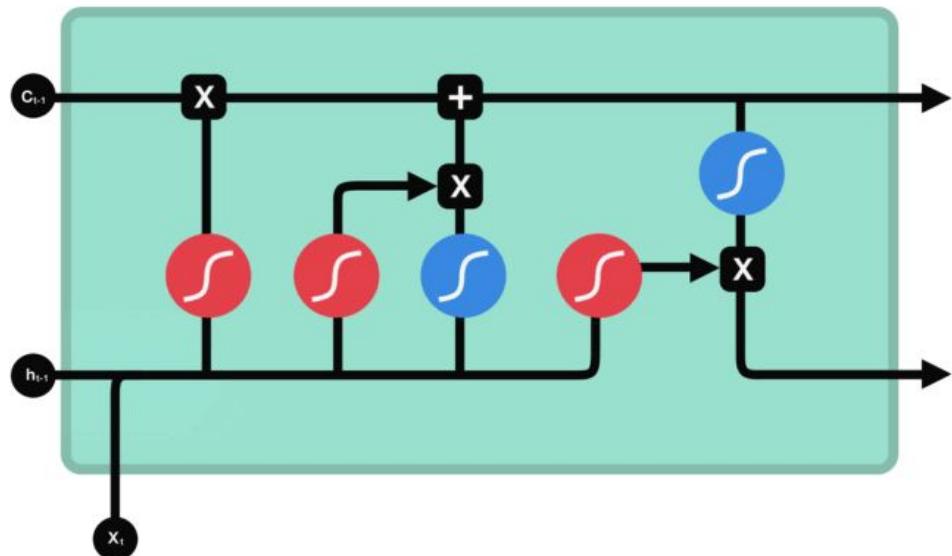


Tanh compresses values between -1 and 1

Sigmoid compresses values between 0 and 1

Useful for updating or forgetting data

Forget Gate



c_{t-1} previous cell state

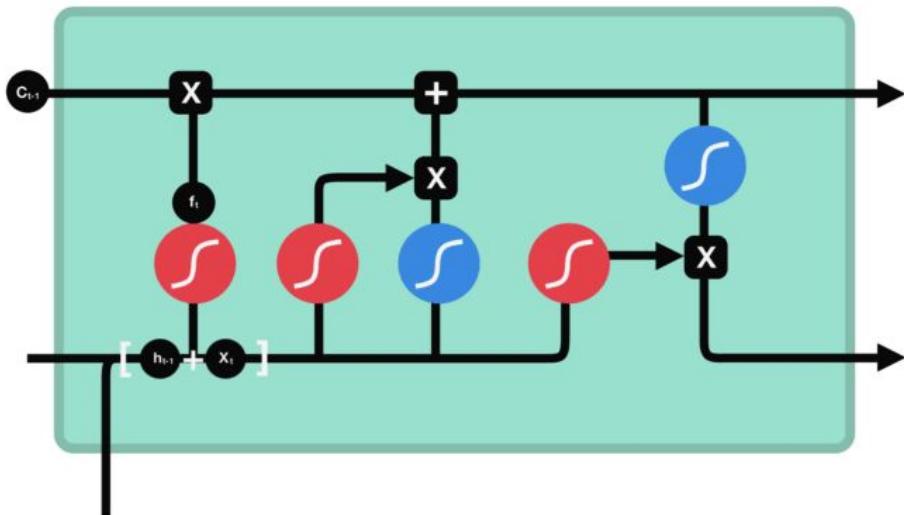
f_t forget gate output

This gate decides which information should be discarded or kept.

The information from the previous hidden state and the information from the current input are passed through the sigmoid function.

The values are between 0 and 1. Near 0 it forgets and near 1 it keeps

Input Gate



c_{t-1} previous cell state

f_t forget gate output

i_t input gate output

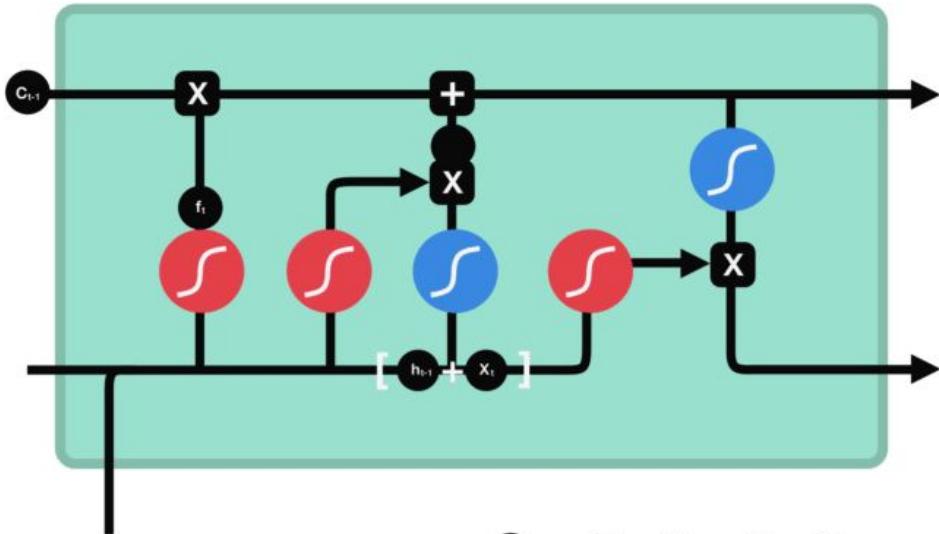
\hat{c}_t candidate

The input gate update the cell state

Previous hidden state and the current input to a sigmoid function.

Hidden state and the current input to the tanh function to compress values between -1 and 1 to help regulate the network. Then you multiply the tanh output by the sigmoid output.

Cell State

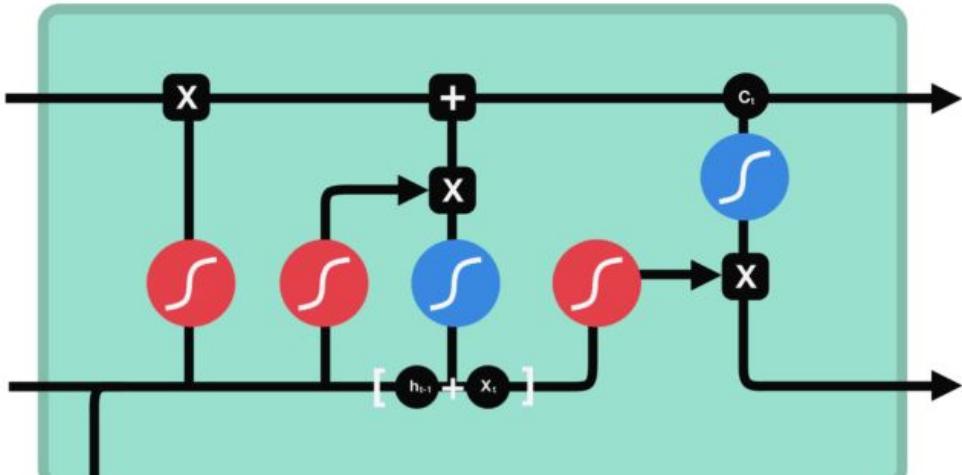


- c_{t-1} previous cell state
- f_t forget gate output
- i_t input gate output
- \tilde{c}_t candidate
- c_t new cell state

The input gate update the cell state
Previous hidden state and the current input to a sigmoid function.
Hidden state and the current input to the tanh function to compress values between -1 and 1 to help regulate the network. Then you multiply the tanh output by the sigmoid output.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

Output Gate



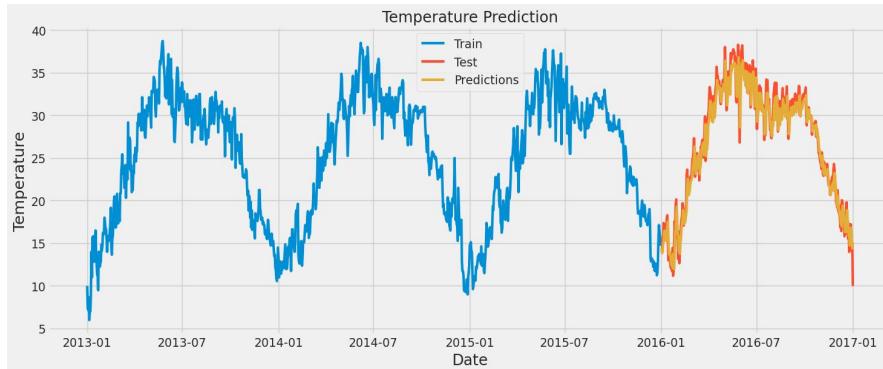
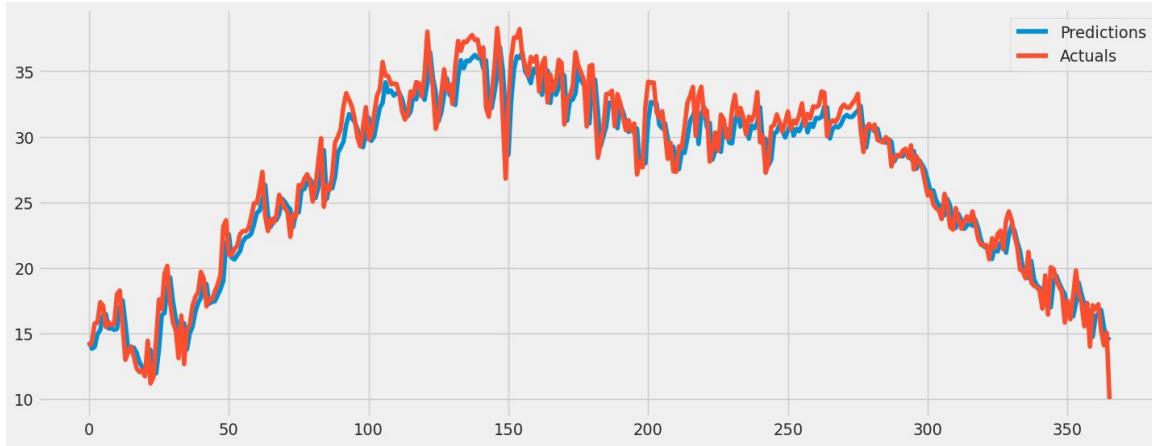
- c_{t-1} previous cell state
- f_t forget gate output
- i_t input gate output
- \tilde{c}_t candidate
- c_t new cell state
- o_t output gate output
- h_t hidden state

The output gate decides what the next hidden state should be.

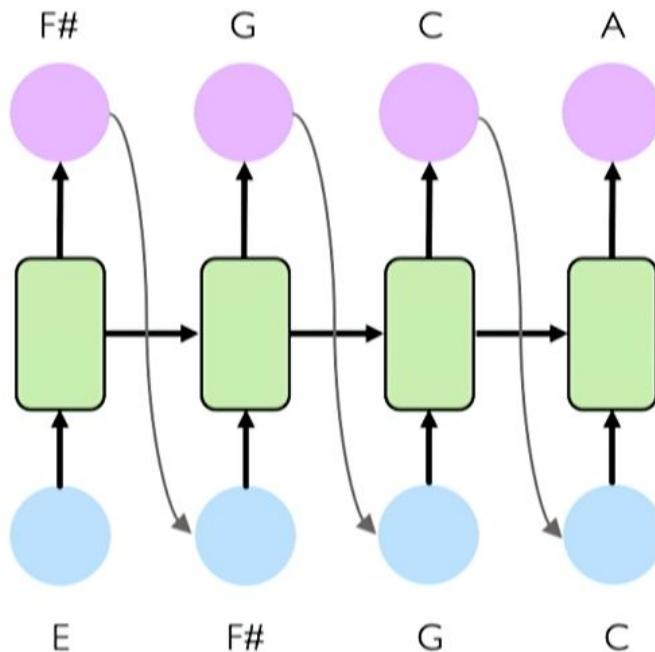
The hidden state contains information about previous inputs.

We multiply the tanh output by the sigmoid output to decide what information the hidden state should carry.

⌚ Time series



LSTM: Music Generation

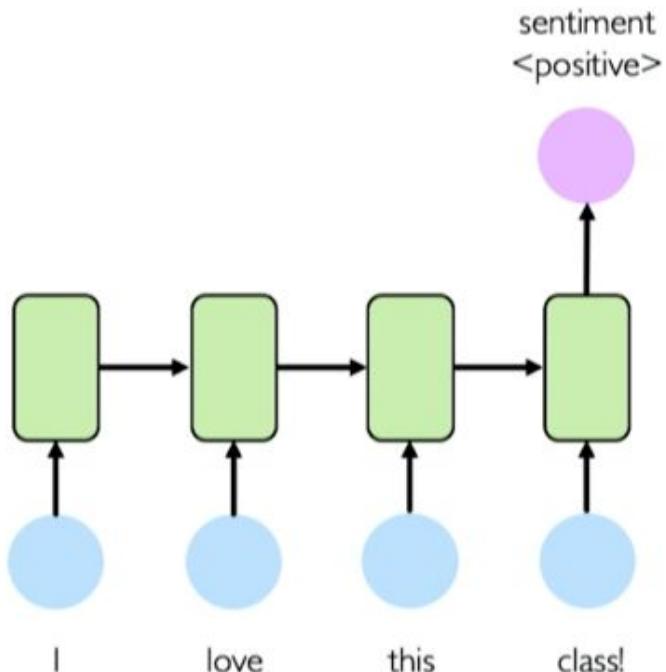


MIT 6.S191 Lab 1: Intro to TensorFlow and Music Generation with RNNs



<https://github.com/aamini/introtodeeplearning/tree/master/lab1>

Sentiment Classification



Tweet sentiment classification

 Ivar Hagendoorn
@Ivar-Hagendoorn

Follow



The [@MIT](#) Introduction to [#DeepLearning](#) is definitely one of the best courses of its kind currently available online introtodeeplearning.com

12:45 PM - 12 Feb 2018

 Angels-Cave
@AngelsCave

Follow



Replying to [@Kazuki02048](#)

I wouldn't mind a bit of snow right now. We haven't had any in my bit of the Midlands this winter! :(

2:19 AM - 25 Jan 2019

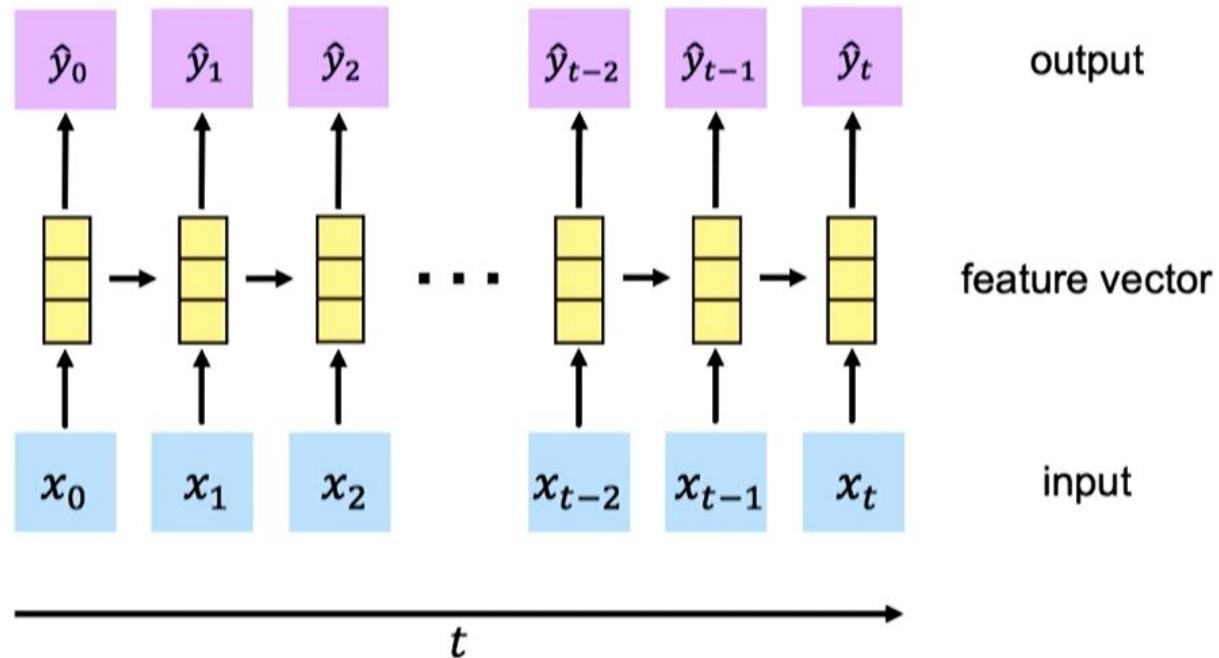
⌚ Limitations of RNN

Limitations of RNNs

EncodingException bottleneck

🕒 Slow, no parallelization

🧠 Not long memory



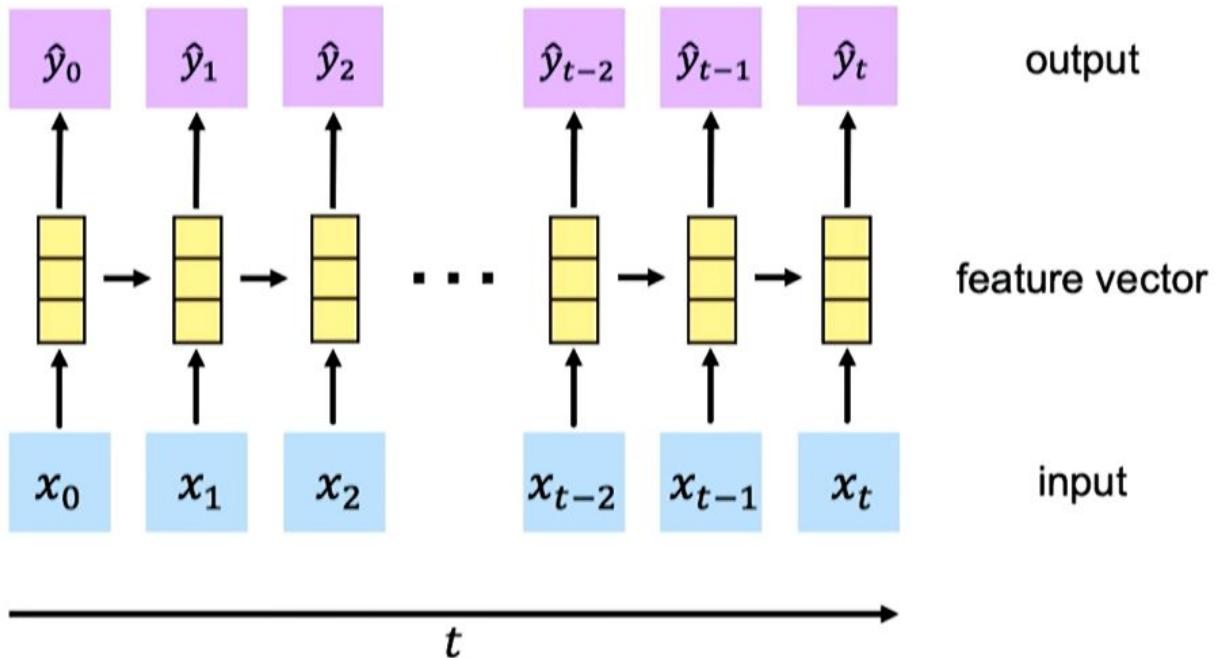
⌚ Limitations of RNN

Desired Capabilities

🌊 Continuous stream

↗️ Parallelization

🧠 Long memory



A large yellow triangle is positioned in the bottom-left corner, pointing towards the center. Inside its base, a white triangle points upwards.

Attention Is All You Need

Attention Is All You Need

Ashish Vaswani*

Google Brain

avaswani@google.com

Noam Shazeer*

Google Brain

noam@google.com

Niki Parmar*

Google Research

nikip@google.com

Jakob Uszkoreit*

Google Research

usz@google.com

Llion Jones*

Google Research

llion@google.com

Aidan N. Gomez* †

University of Toronto

aidan@cs.toronto.edu

Łukasz Kaiser*

Google Brain

lukaszkaiser@google.com

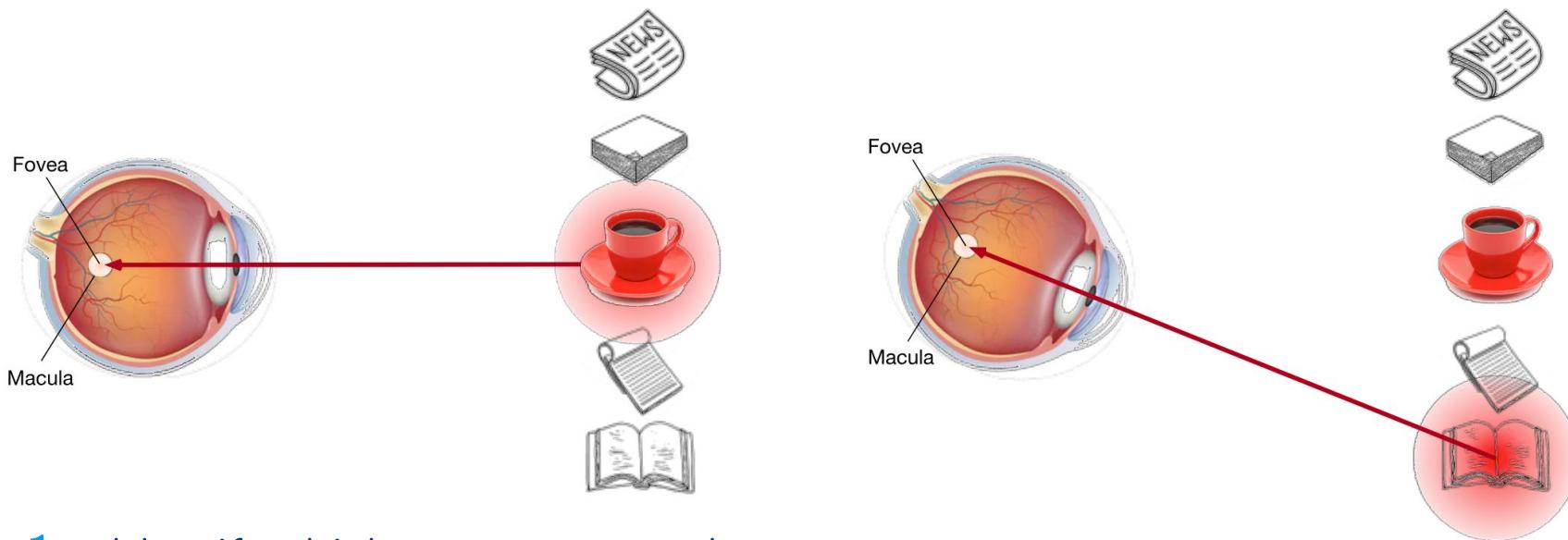
Illia Polosukhin* †

illia.polosukhin@gmail.com

Abstract

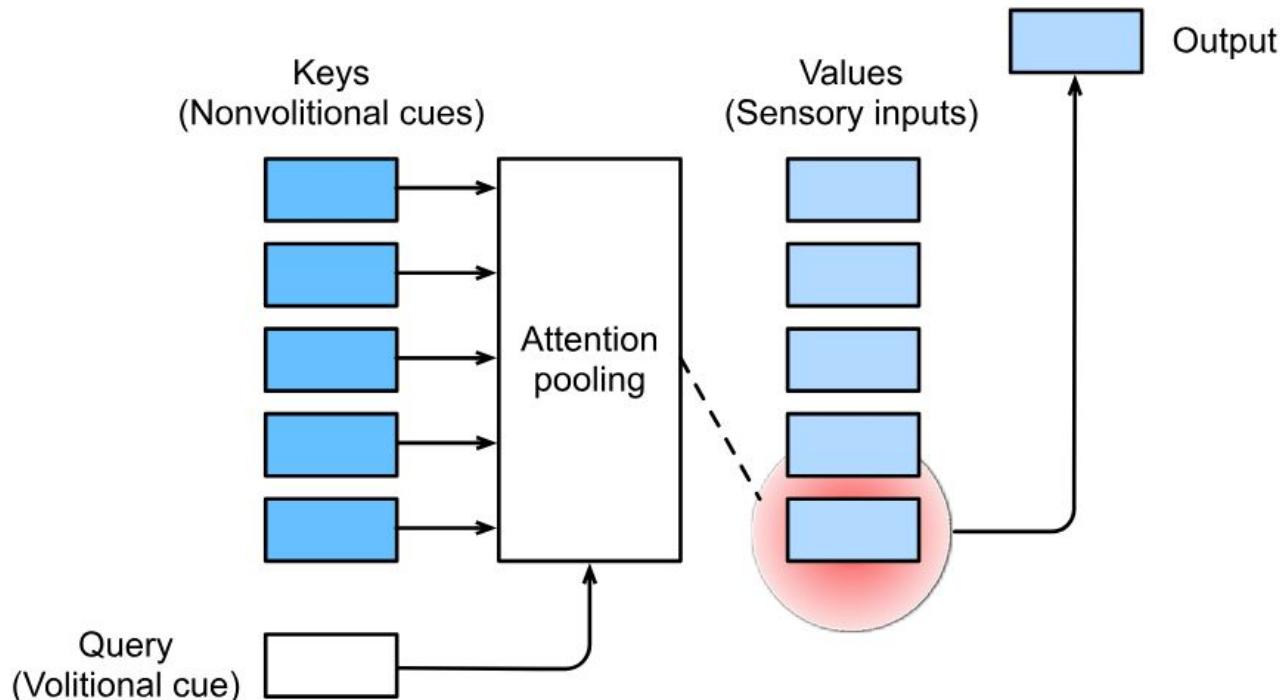
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Attention Is All You Need



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

Attention Is All You Need



$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i$$

YouTube search results for "deep learning":

- Query (Q)**: GIANT SEA TURTLES • AMAZING CORAL REEF FISH • 12 HOURS OF THE BEST RELAX MUSIC (5.4M views • 4 years ago)
- Key (K_1)**: MIT 6.S191 (2020): Introduction to Deep Learning (1M views • 1 year ago)
- Key (K_2)**: The Kobe Bryant Fadeaway Shot (235K views • 1 year ago)
- Key (K_3)**: Postup #Footwork #PartOne No copyright infringement is intended, all audio and video clips property of their ... (16.45s)

How similar is the key to the query?

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

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i$$

Query (Q)

Key (K_1)

Key (K_2)

Value (V)

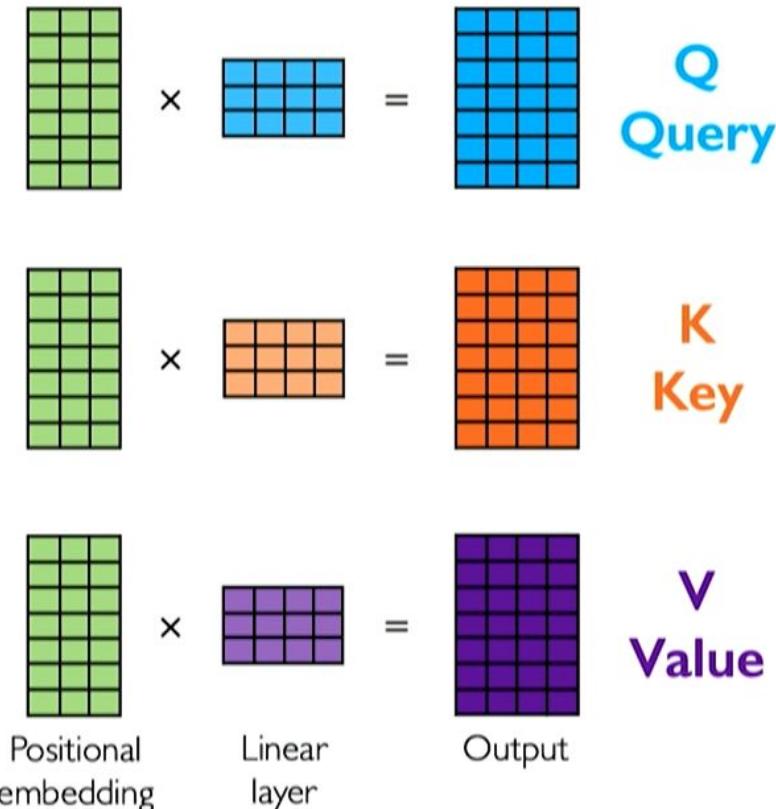
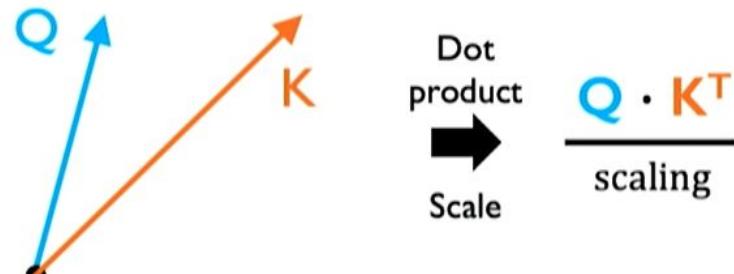
Key (K_3)

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

⌚ Self-Attention

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

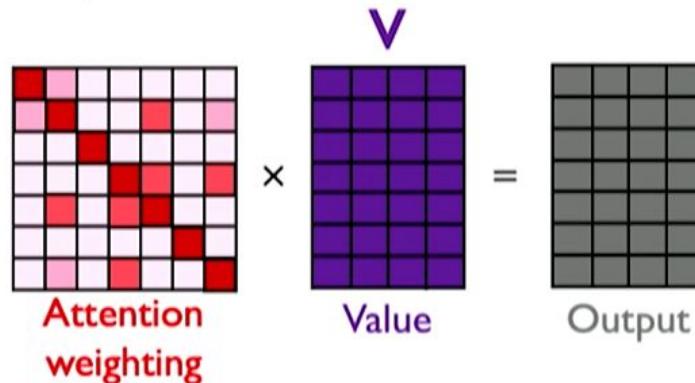
1. Encode **position** information
2. Extract **query**, **key**, **value** for search



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**

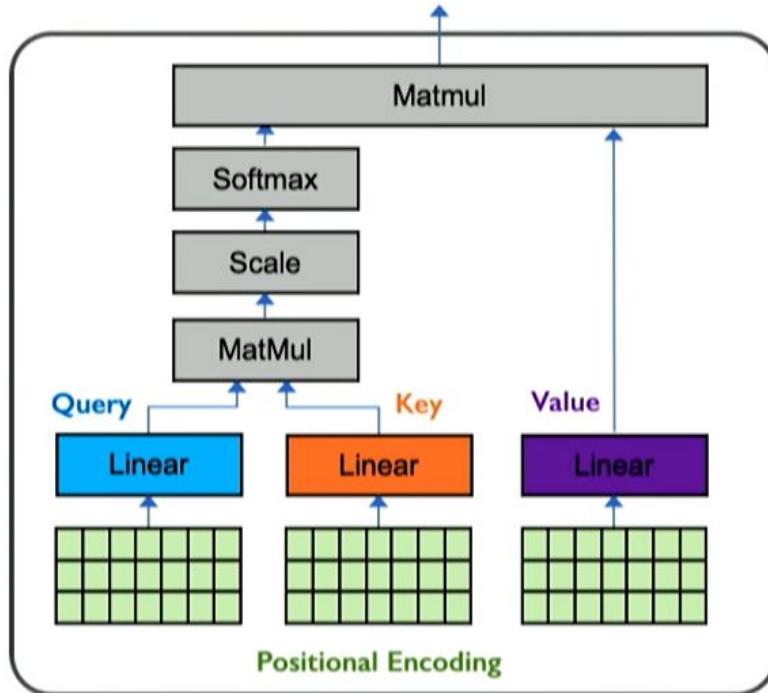
Last step: self-attend to extract features



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

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

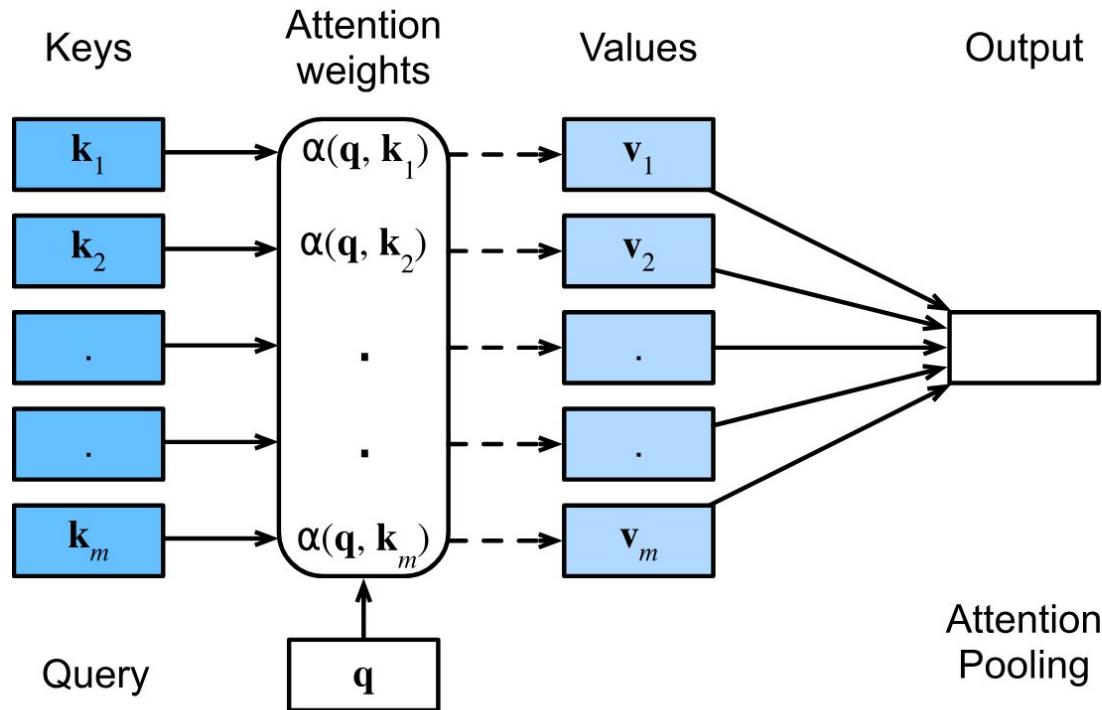
1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



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

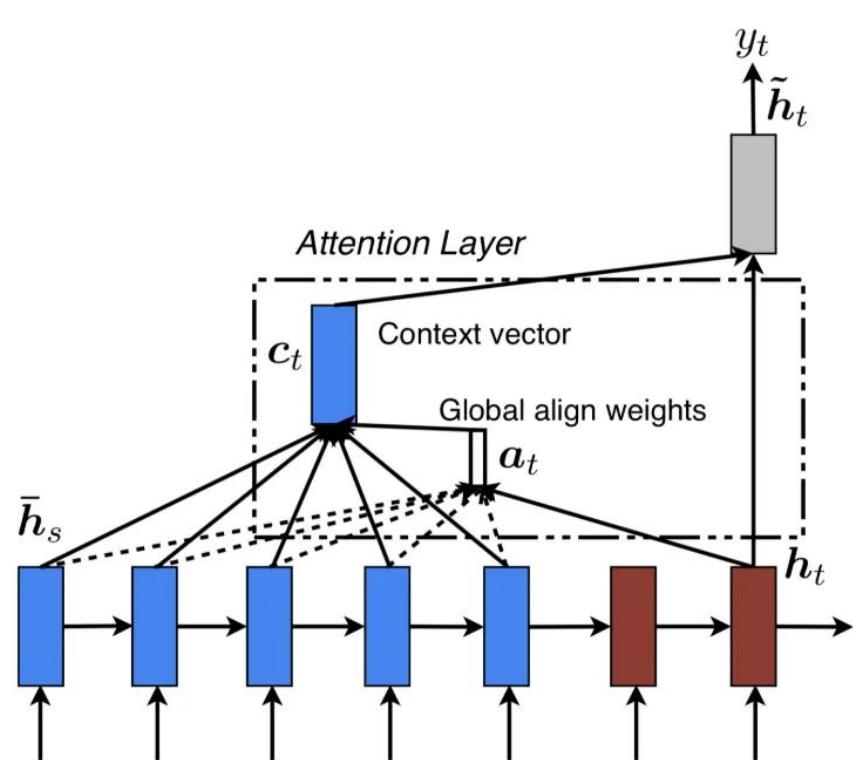
Queries, Keys, and Values

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i$$

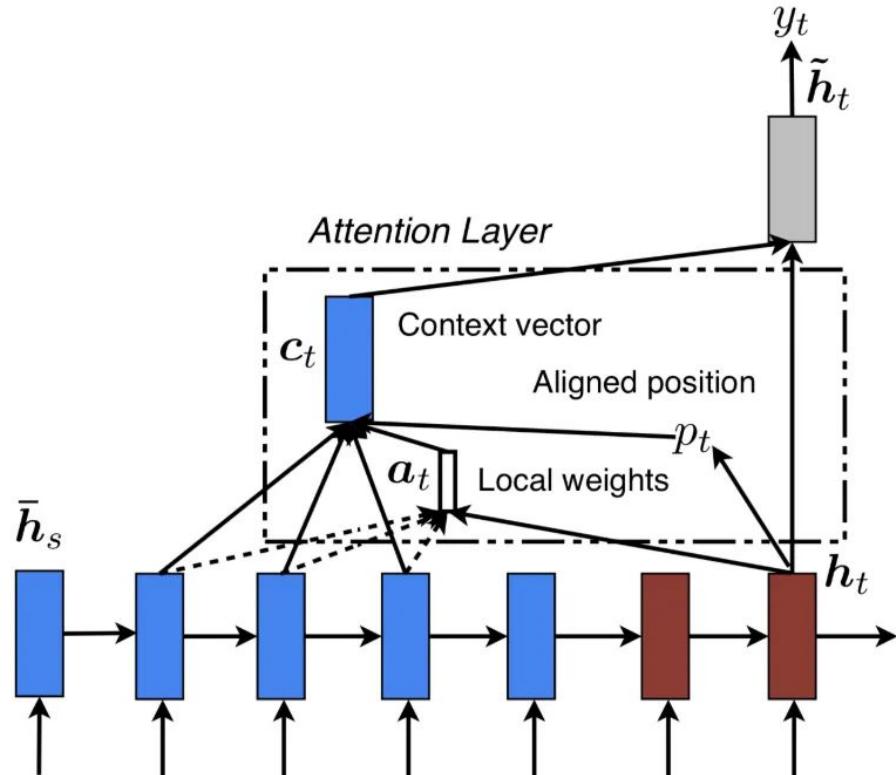


$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_j \exp(a(\mathbf{q}, \mathbf{k}_j))}$$

Global vs local attention



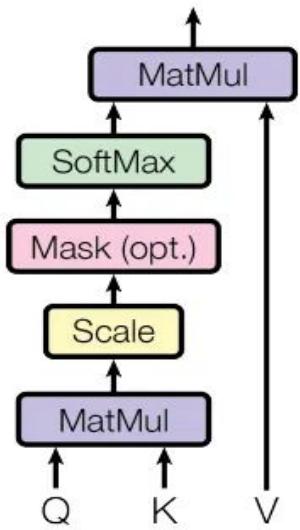
Global Attention Model



Local Attention Model

⌚ Self-Attention

Scaled Dot-Product Attention



Multi-Head Attention

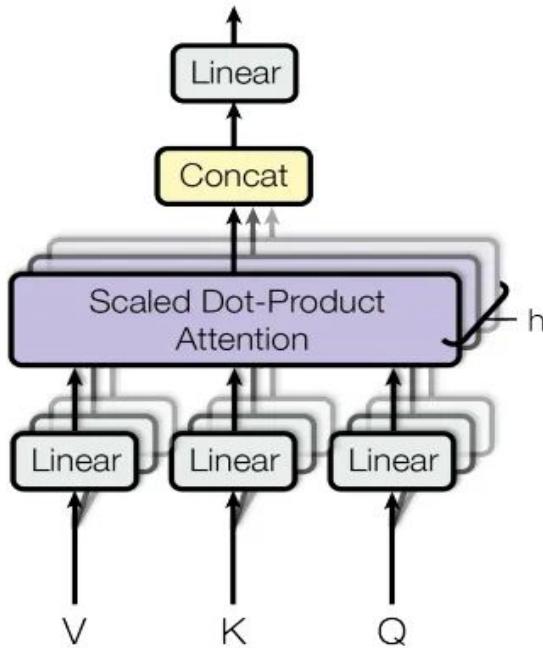


Figure source: Vaswani et al. 2017

Illustrated: Self-Attention

A step-by-step guide to self-attention with illustrations and code



Raimi Karim · Follow

Published in Towards Data Science · 9 min read · Nov 18, 2019

<https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a>

<https://colab.research.google.com/drive/1rPk3ohrmVclqhH7uQ7qys4oznDdAhpzF>

The Transformer model by Vaswani et al. 2017 stacks six encoder blocks followed by six decoder blocks.

Each encoder block has:

- a self-attention layer followed by a feedforward layer to capture non-linearities
- each of these is also followed by a layer norm
- there are residual connections between encoder blocks

Each decoder is similar to the encoder, except that it has a 3rd layer that performs multi-head attention on the output from the encoders and the previous layers of the decoder. Masking is used here to prevent positions from looking ahead during self-attention.

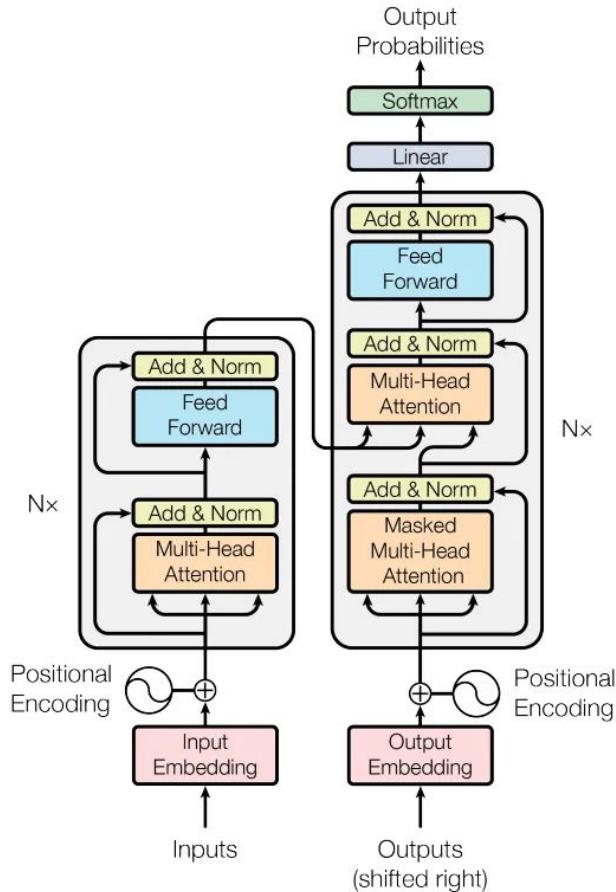
Encoder-Decoder Architecture

Advantages:

- constant path length between any two positions in the sequence (hence are better to model context in long sequences);
 - i.e., every token in the sequence is looking at every other token in the sequence; so now if you have a token that depends on another that is way far behind in the sequence, you don't have the issue of losing context.
- better parallelization because of how the computations are done; no sequential computation within a layer.

Drawbacks:

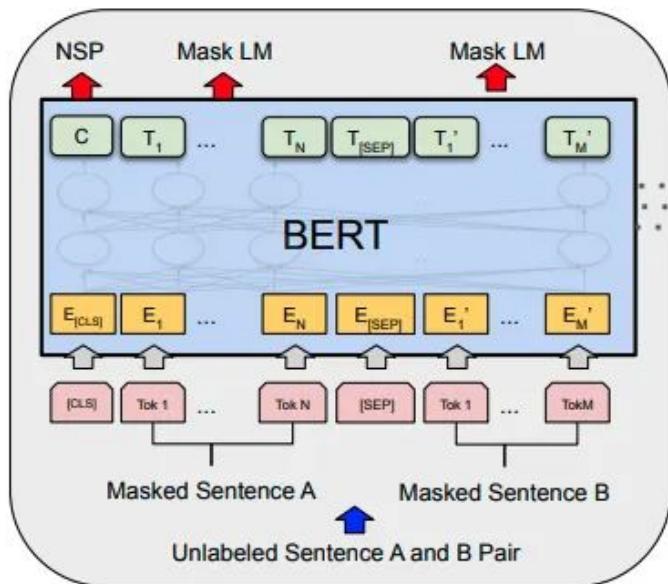
- Self-attention takes quadratic time and space (each token attends to all other tokens); scaling, which is now very popular and effective, has now become very



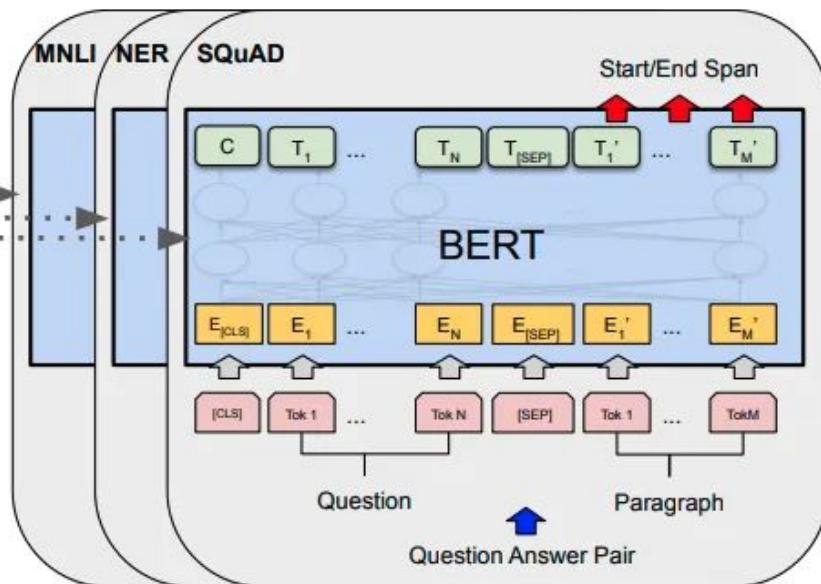
<https://aclanthology.org/N19-1423/>

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- consists of only the encoder blocks from transformers.



Pre-training



Fine-Tuning

⟳ Self-Attention Applied

Language Processing

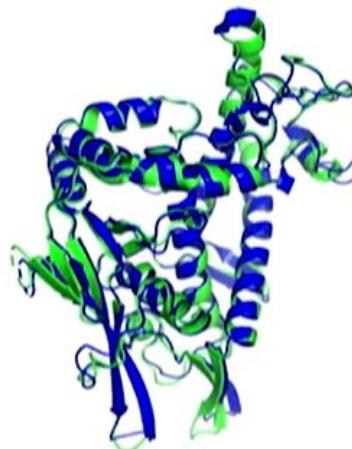


An armchair in the shape of an avocado

BERT, GPT-3

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

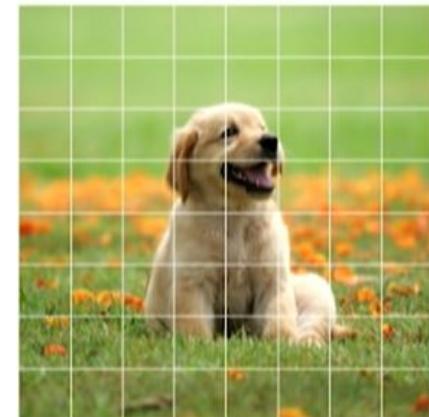
Biological Sequences



AlphaFold2

Jumper et al., Nature 2021

Computer Vision



Vision Transformers

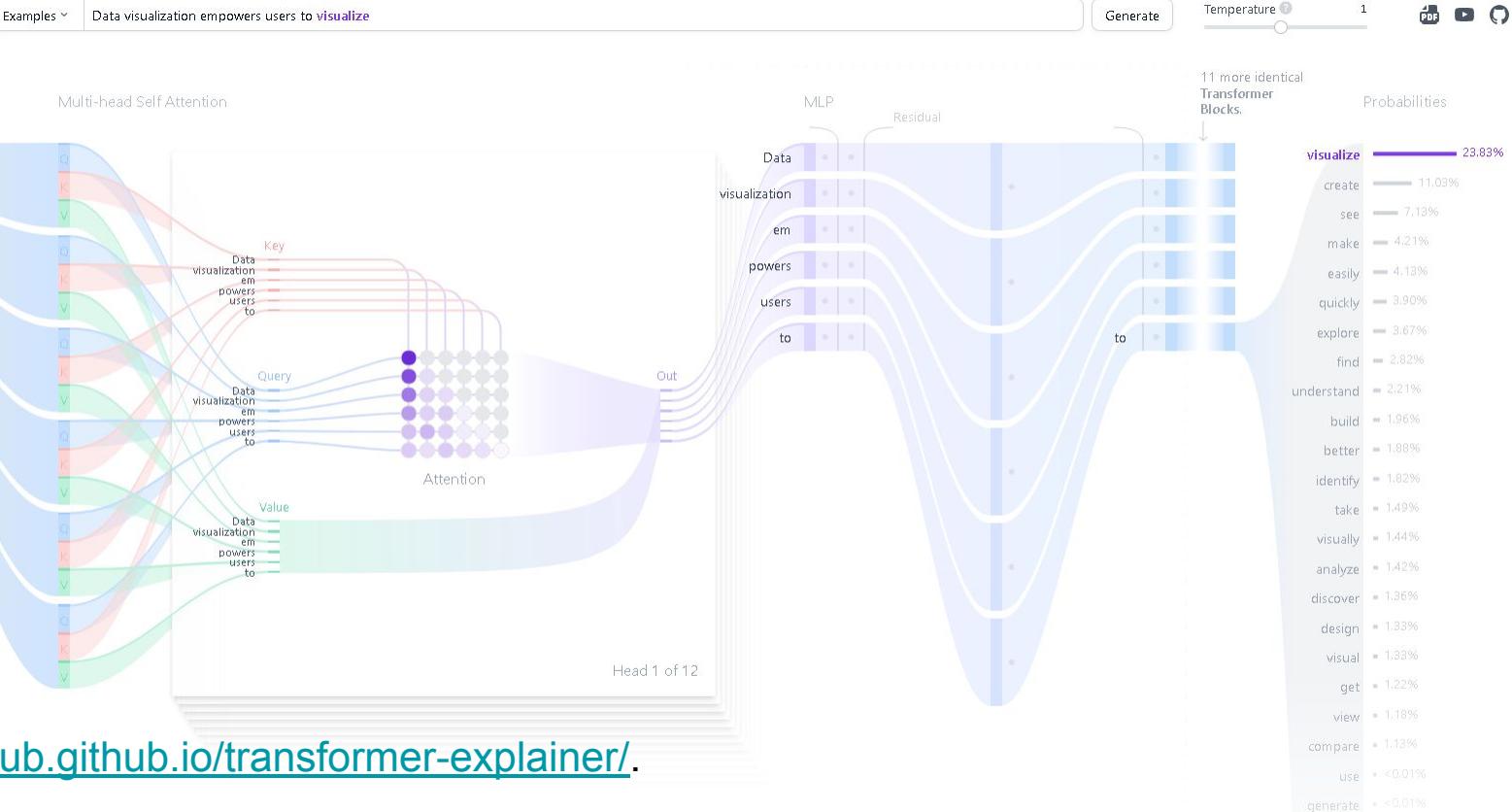
Dosovitskiy et al., ICLR 2020

1. RNNs are suited for sequence modeling
2. Model sequences via a recurrence relation
3. Backpropagation through time
4. LSTM
5. Self-attention to model sequences without recurrence



References

TRANSFORMER EXPLAINER



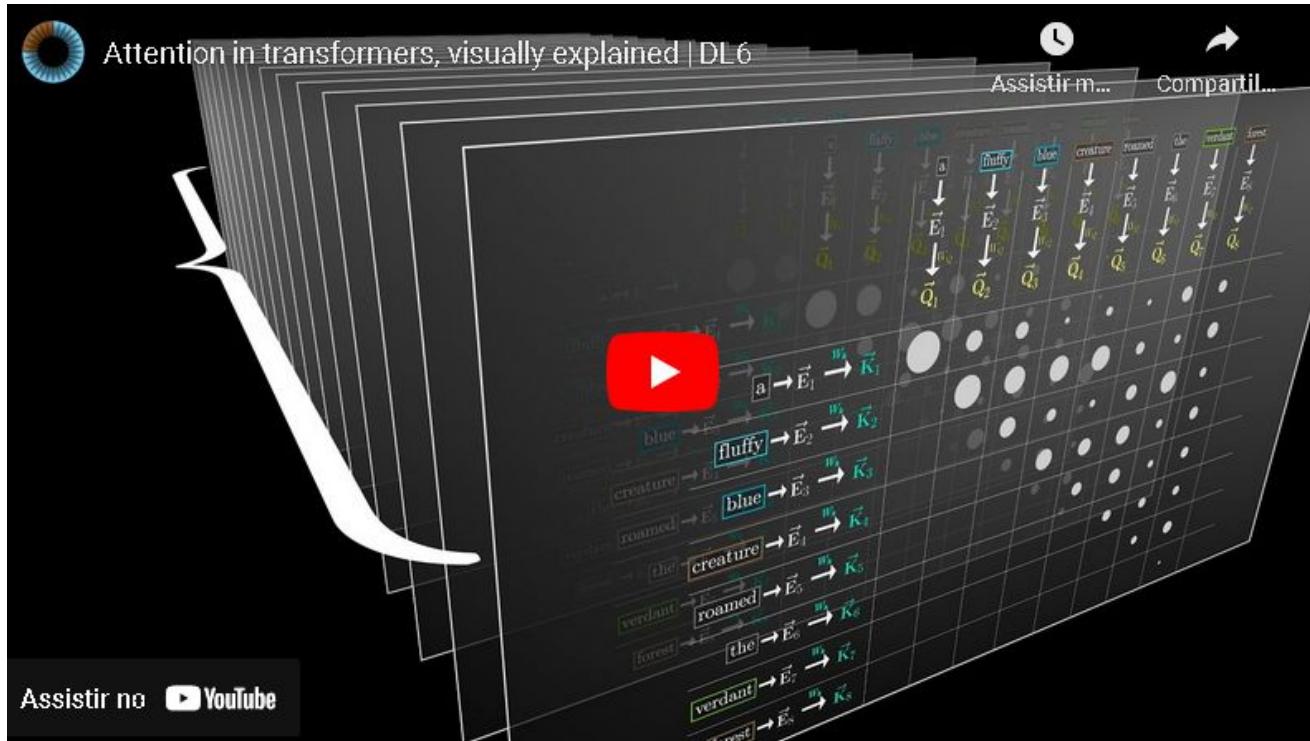
<https://poloclub.github.io/transformer-explainer/>

The Illustrated Transformer



<https://jalammar.github.io/illustrated-transformer/>

References



<https://www.youtube.com/watch?v=eMlx5fFNoYc>

TensorFlow notebook



- <https://colab.research.google.com/github/tensorflow/text/blob/master/docs/tutorials/transformer.ipynb>

Hugging Face's notebooks



- <https://huggingface.co/docs/transformers/en/notebooks>

- https://d2l.ai/chapter_recurrent-neural-networks/index.html
- https://d2l.ai/chapter_recurrent-neural-networks/language-model.html
- https://d2l.ai/chapter_recurrent-modern/lstm.html
- <https://dair-ai.notion.site/Lecture-2-RNNs-and-Transformers-71fb3ba2a24f4b6c8cc77281fc19cfab>
- <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- <https://www.kaggle.com/code/hadeerismail/daily-climate-time-series-analysis-lstm/notebook>
- <https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data/data>



patrick.terrematte@ufrn.br

 **Discord** patrickterrematte



Prof. Dr. Patrick Terrematte