



Master in Computer Vision Barcelona

[<http://pagines.uab.cat/mcv/>]

Xavier Giro-i-Nieto

 [@DocXavi](https://twitter.com/DocXavi)
 xavier.giro@upc.edu

Associate Professor
Universitat Politècnica de Catalunya



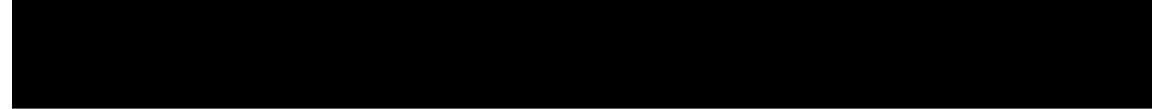
Institut
de Robòtica
i Informàtica
Industrial



UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Module 6 - Day 8 - Lecture 1
The Transformer
31th March 2022

Video-lecture



Lecture 20

The Transformer



DEEP LEARNING FOR ARTIFICIAL INTELLIGENCE
Masters @ UPC TelecomBCN Barcelona (5th edition), Autumn 2021

Instructors:

Xavier Giro-i-Nieto	Veronica Vilanova	Javier Ruiz	Ramon Martos	Lara Torres	Gerard Gallego
---------------------	-------------------	-------------	--------------	-------------	----------------

UPC - UNIVERSITAT POLITÈCNICA DE CATALUNYA
TelecomBCN



Xavier Giro-i-Nieto

Associate Professor
Universitat Politècnica de Catalunya

@DocXavi
xavier.giro@upc.edu

Acknowledgments



Marta R. Costa-jussà

Associate Professor
Universitat Politècnica de Catalunya



Carlos Escolano

PhD Candidate
Universitat Politècnica de Catalunya



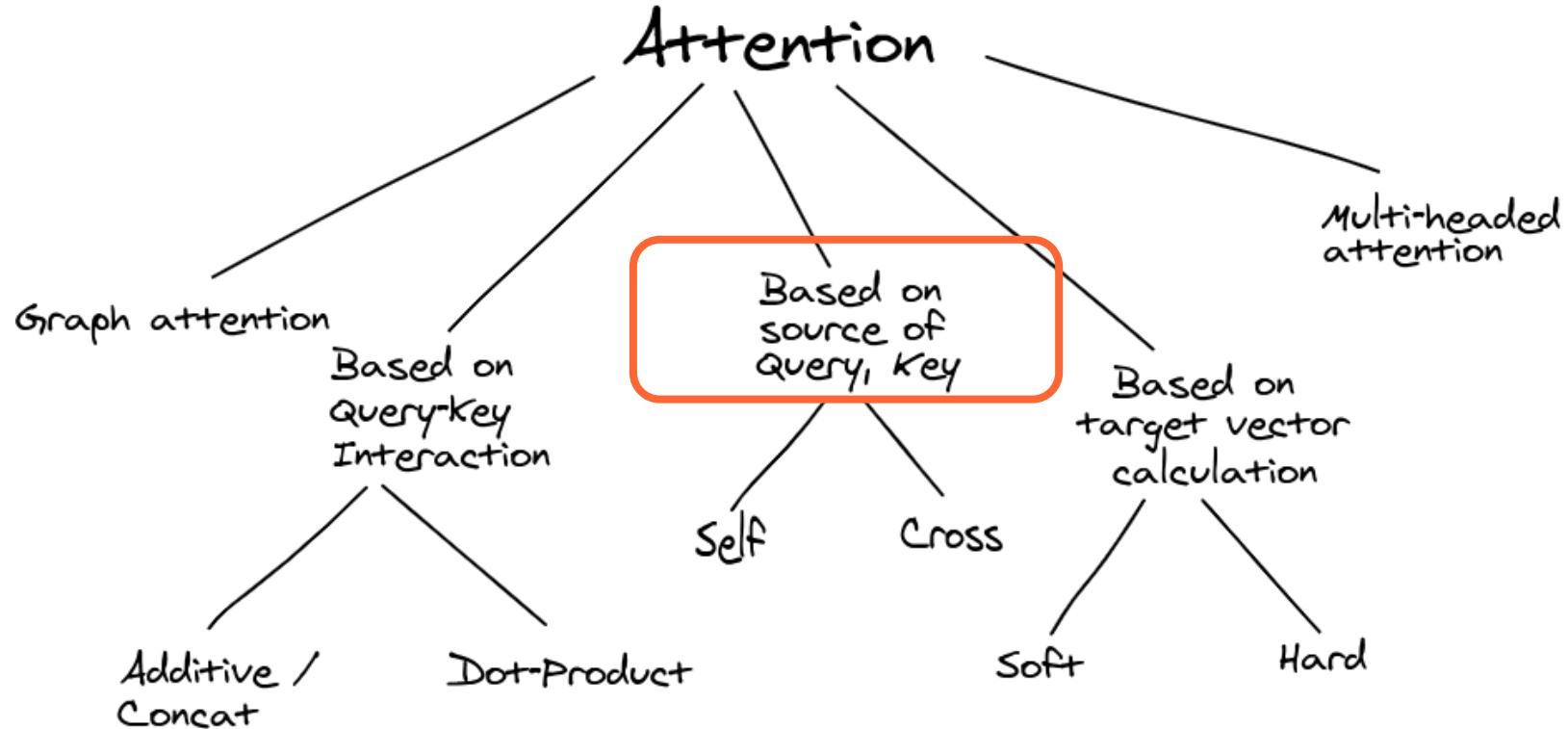
Gerard I. Gállego
PhD Student
Universitat Politècnica de Catalunya
gerard.ion.gallego@upc.edu
[@qeiongallego](https://twitter.com/qeiongallego)



Outline

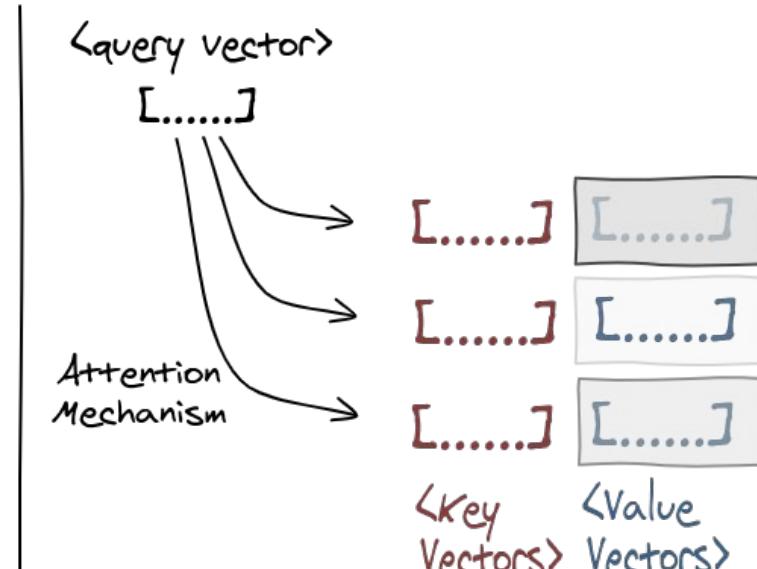
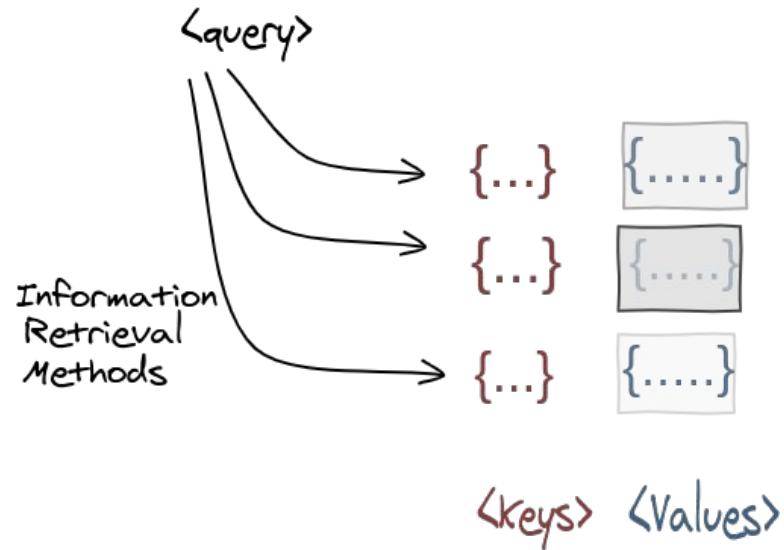
1. Reminders

Reminder

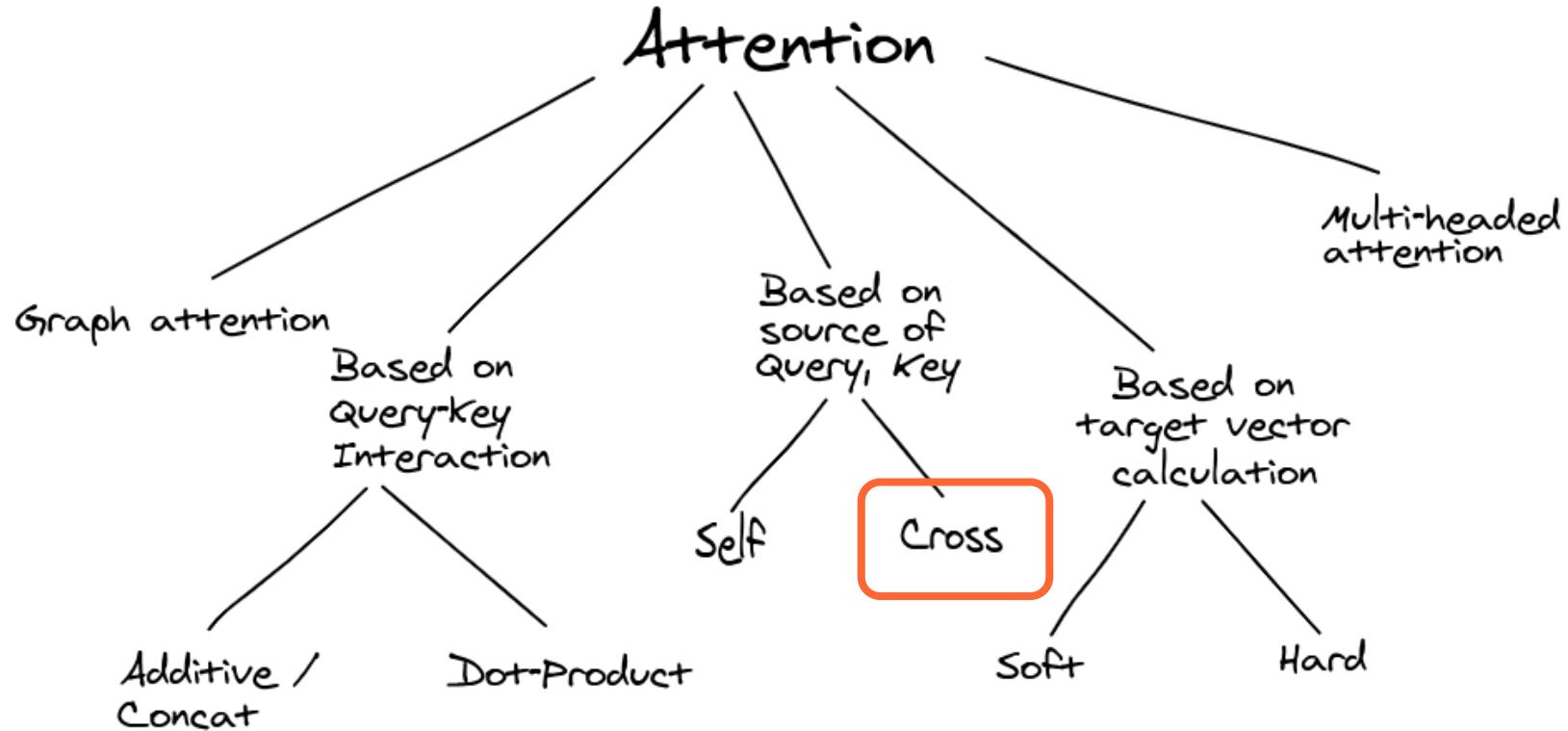


Reminder

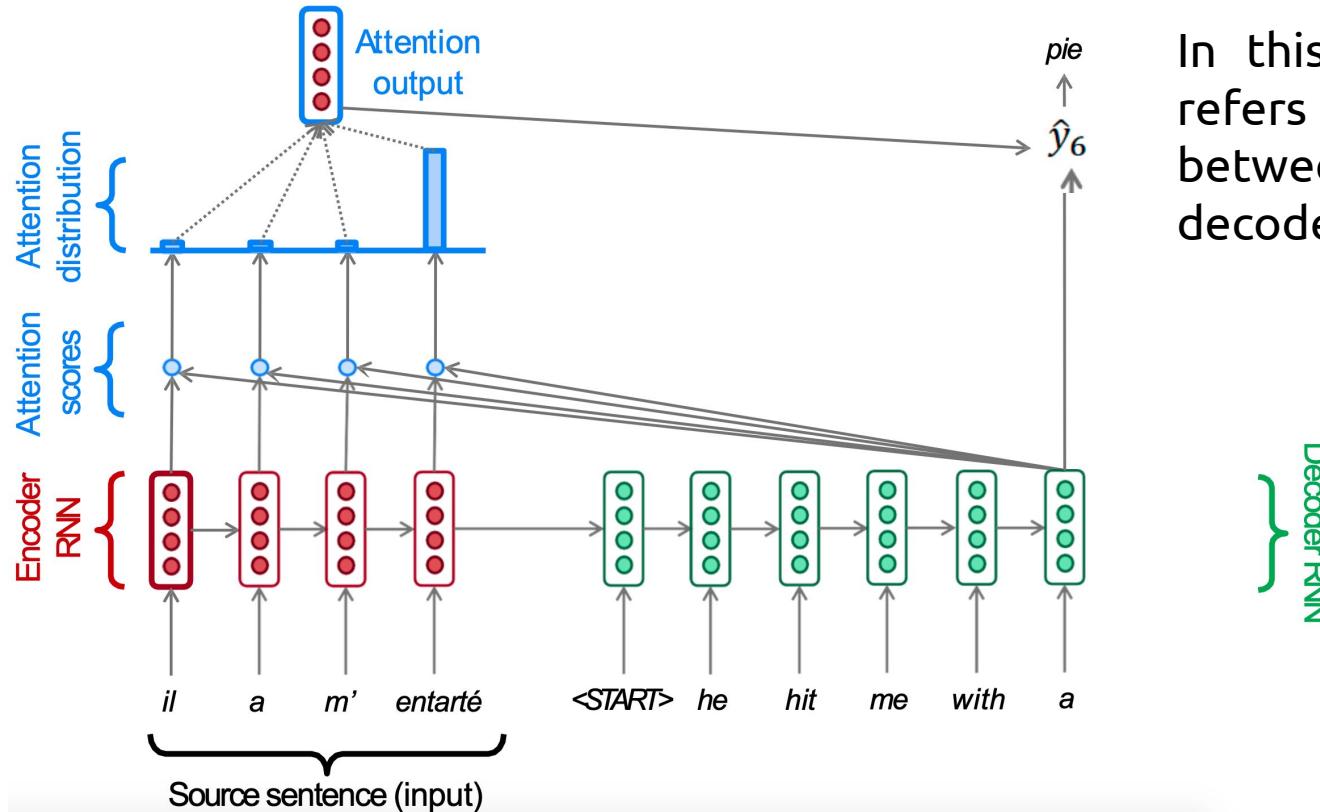
Attention is a mechanism to compute a context vector (c) for a **query (Q)** as a weighted sum of **values (V)**.



Reminder

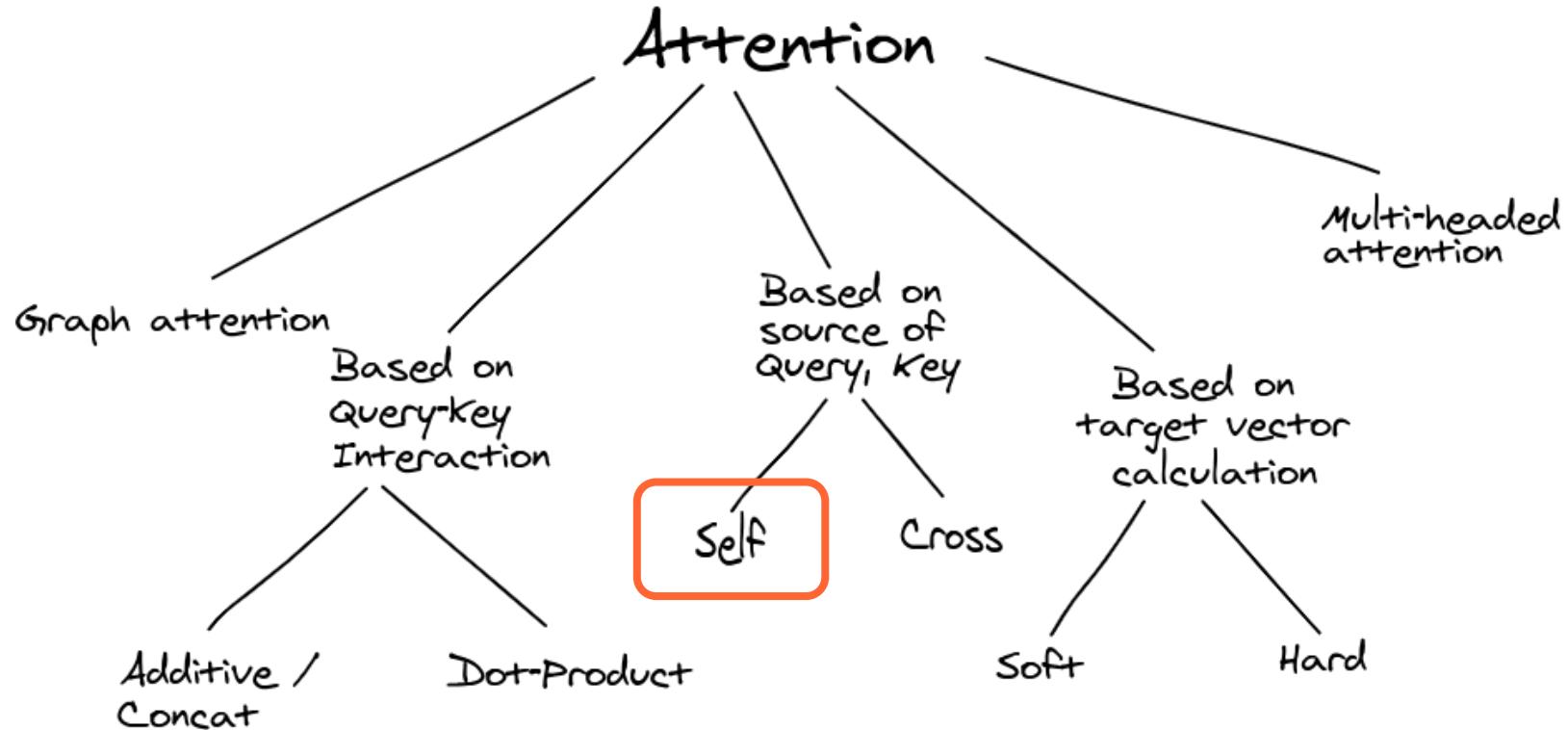


Reminder: Seq2Seq with Cross-Attention



In this case, **cross-attention** refers to the attention between the encoder and decoder states.

What may the term “self” refer to, as a contrast of “cross”-attention ?



Outline

1. Motivation
2. **Self-Attention (SA)**

Self-Attention (or intra-Attention)

Self-attention refers to attending to other elements from the SAME sequence.

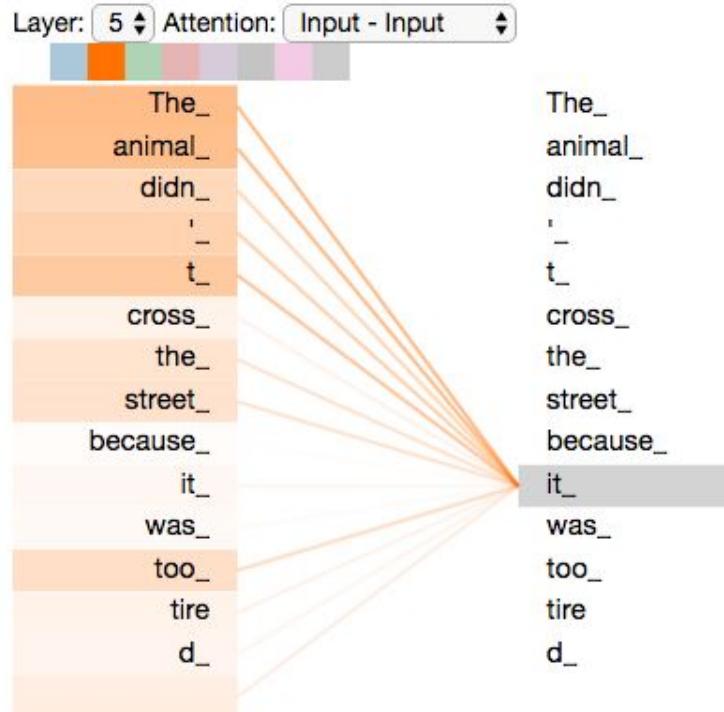
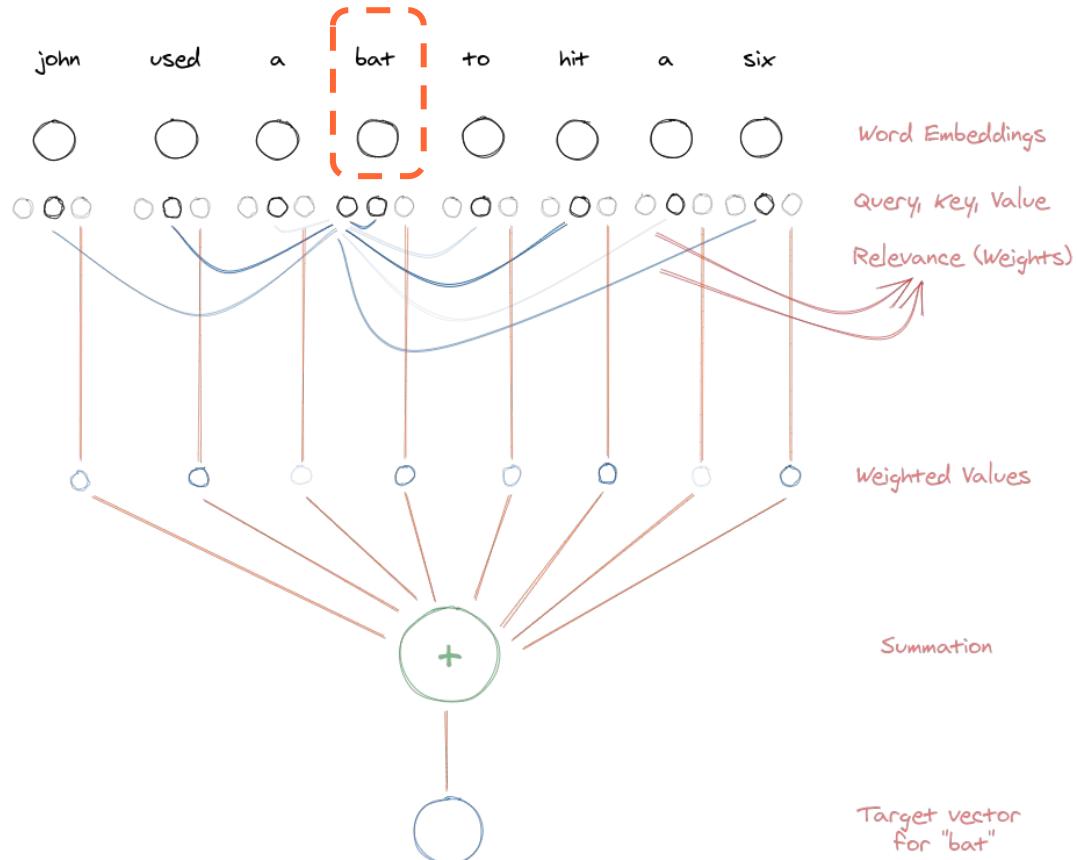


Figure:

Jay Alammar,

["The Illustrated Transformer"](#)

Self-Attention (or intra-Attention)



Query (Q)
 $g(x) = W^Q x$

Key (K)
 $f(x) = W^K x$

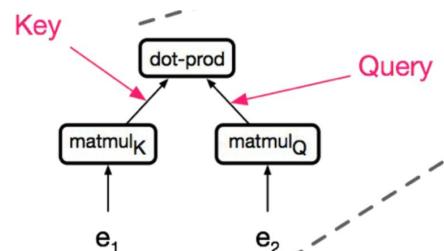
Value (V)
 $h(x) = W^V x$

W^Q , W^K and W^V are **projection layers** shared across all words.

Self-Attention (or intra-Attention)

Which steps are necessary to compute the contextual representation of a word embedding e_2 in a sequences of four words embeddings (e_1, e_2, e_3, e_4)?

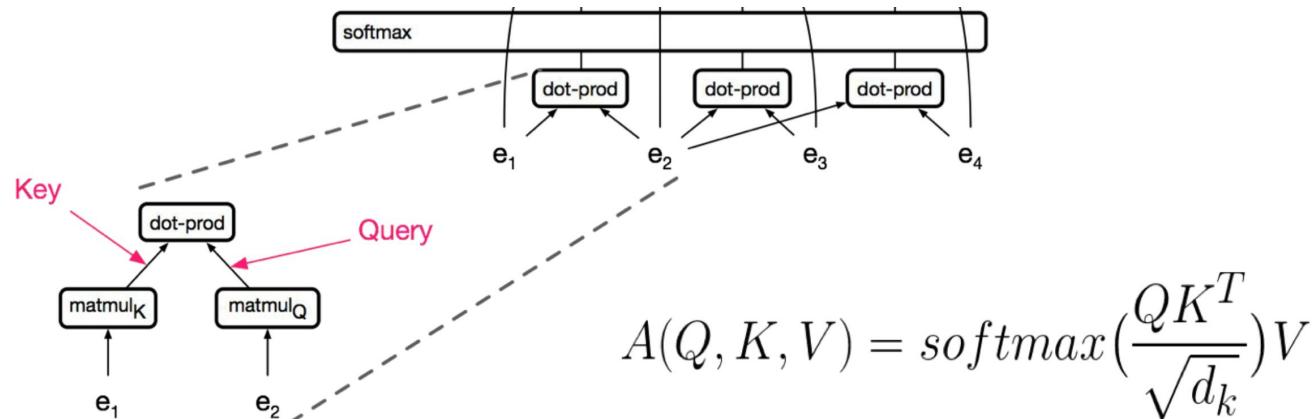
A (scaled) dot-product is computed between each pair of word embeddings (eg. e_1 and e_2)...



Self-Attention (or intra-Attention)

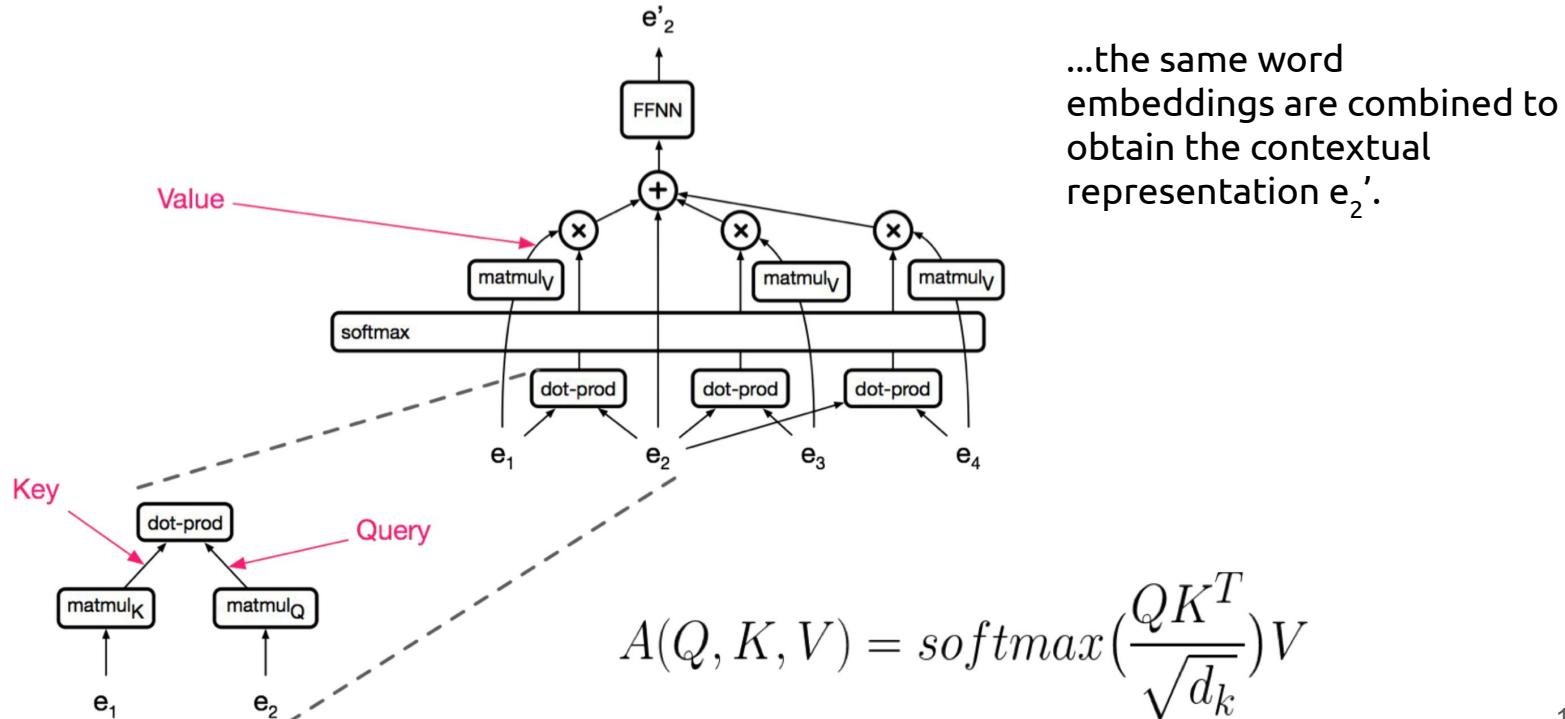
Which steps are necessary to compute the contextual representation of a word embedding e_2 in a sequences of four words embeddings (e_1, e_2, e_3, e_4)?

... a softmax layer normalizes the attention scores to obtain the attention distribution...



Self-Attention (or intra-Attention)

Which steps are necessary to compute the contextual representation of a word embedding e_2 in a sequences of four words embeddings (e_1, e_2, e_3, e_4)?



Self-Attention (or intra-Attention)

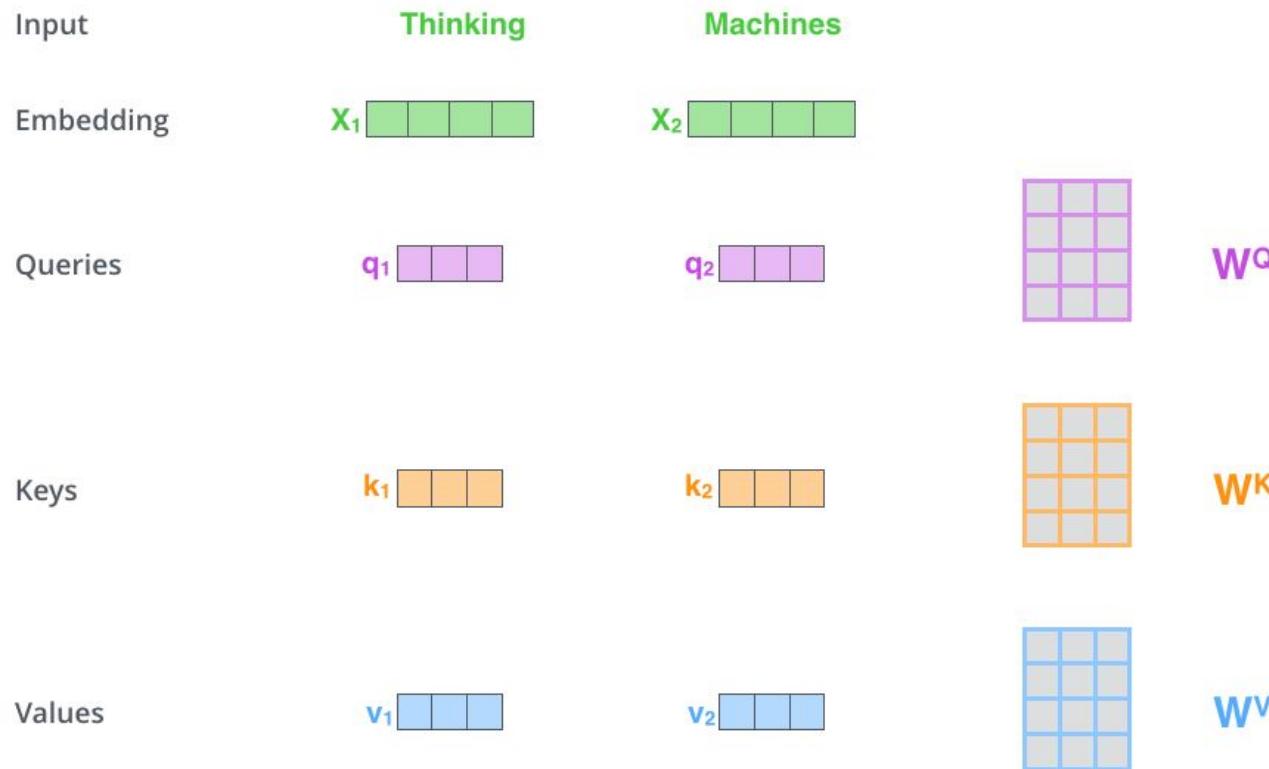
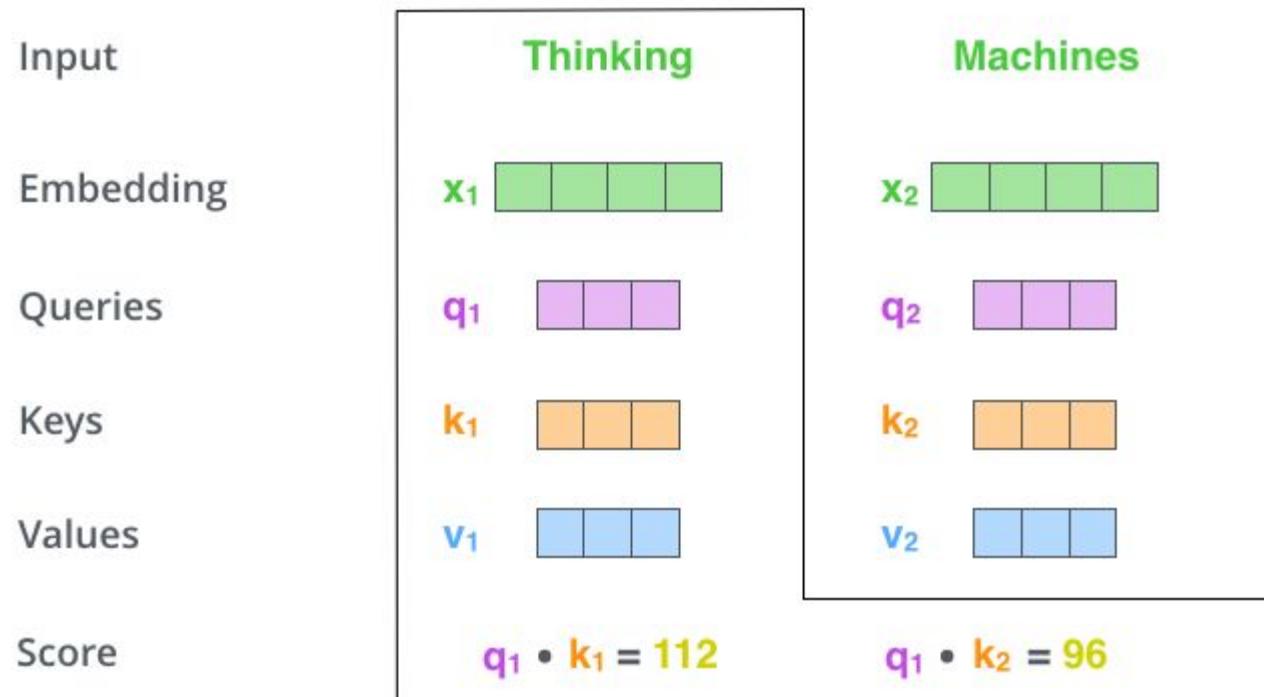


Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Self-Attention (or intra-Attention)



Self-Attention (or intra-Attention)

Input		
Embedding	x_1	
Queries	q_1	
Keys	k_1	
Values	v_1	
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Self-Attention (or intra-Attention)

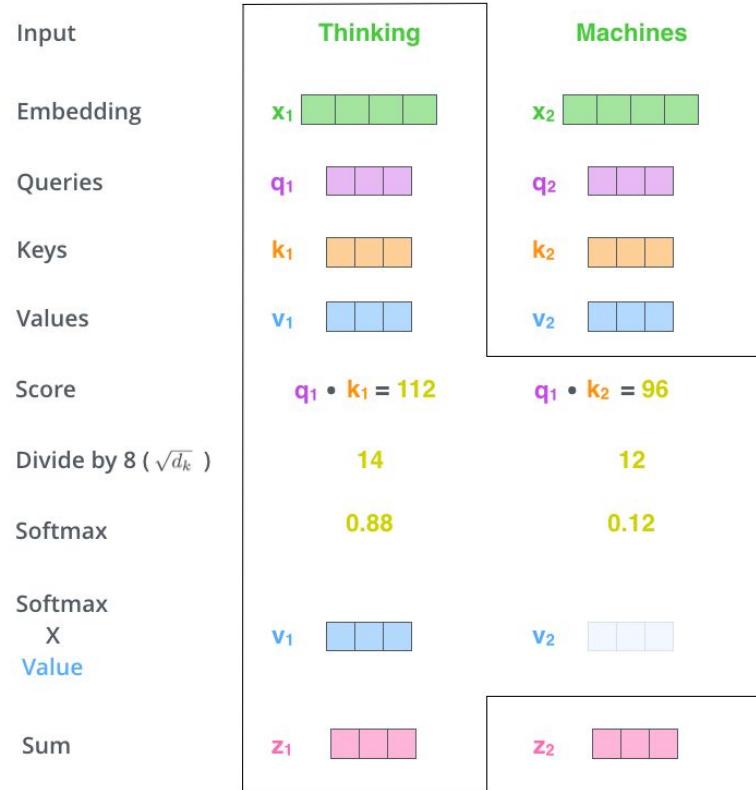


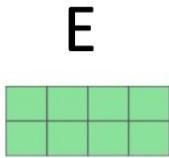
Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Self-Attention (or intra-Attention)

1) This is our
input sentence

2) We embed
each word

Thinking
Machines

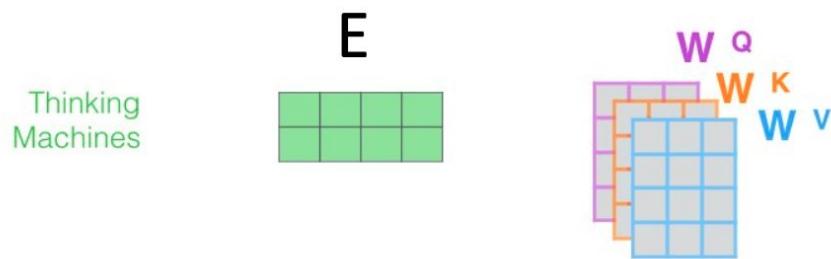


Self-Attention (or intra-Attention)

1) This is our input sentence

2) We embed each word

3) We multiply \mathbf{X} with weight matrices



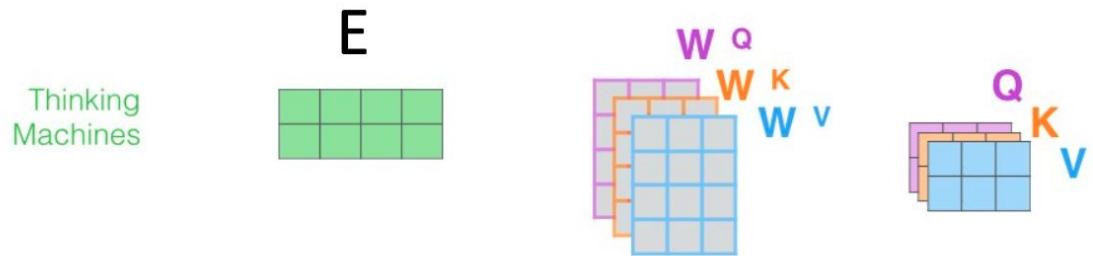
Self-Attention (or intra-Attention)

1) This is our input sentence

2) We embed each word

3) We multiply \mathbf{X} with weight matrices

4) Calculate attention using the resulting $\mathbf{Q}/\mathbf{K}/\mathbf{V}$ matrices



$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} = \mathbf{E}'$$

Self-Attention (or intra-Attention)

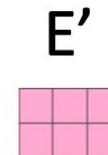
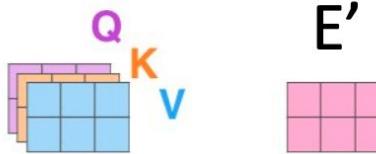
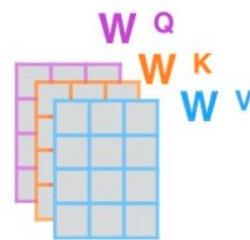
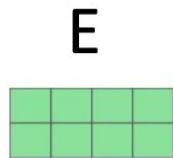
1) This is our input sentence

2) We embed each word

3) We multiply \mathbf{X} with weight matrices

4) Calculate attention using the resulting $\mathbf{Q}/\mathbf{K}/\mathbf{V}$ matrices

Thinking
Machines

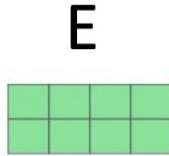


Self-Attention (or intra-Attention)

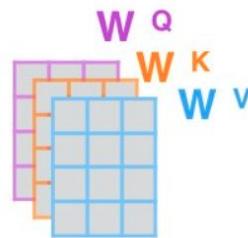
1) This is our input sentence

Thinking
Machines

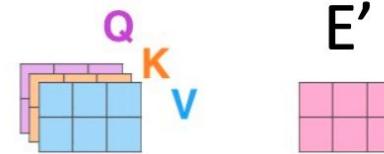
2) We embed each word



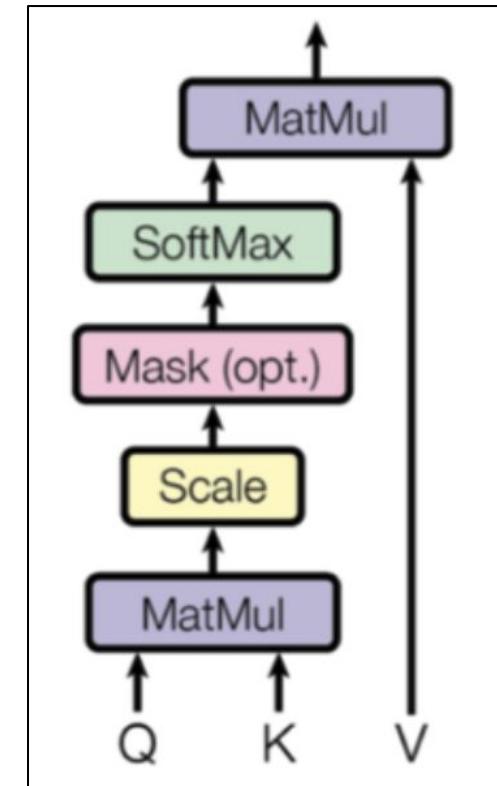
3) We multiply \mathbf{X} with weight matrices



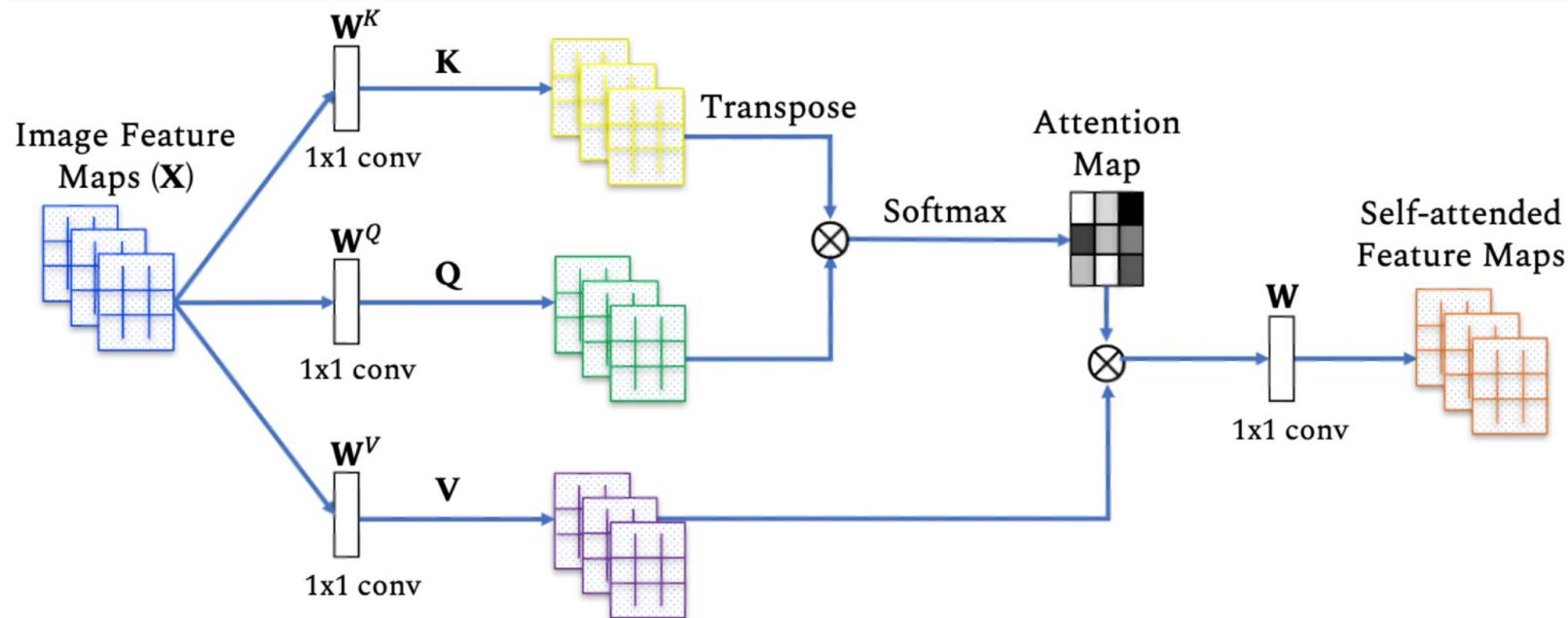
4) Calculate attention using the resulting $Q/K/V$ matrices



Scaled dot-product attention



Self-Attention over 2D Image Feature Maps



#SAGAN Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "[Self-attention generative adversarial networks.](#)" ICML 2019. [\[video\]](#)

Self-Attention in for image generation

Generator (G):

Discriminator (D):

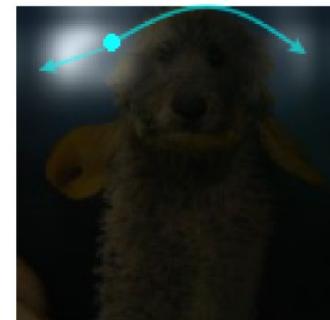
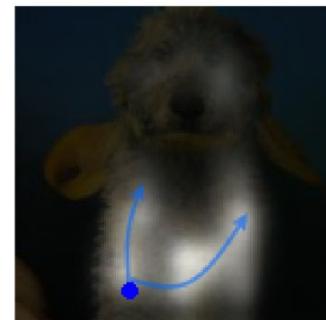
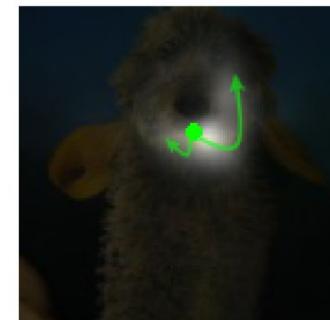
Details can be generated using cues from all feature locations.

Can check consistency between features in distant portions of the image.

Query locations



Attention maps for different query locations

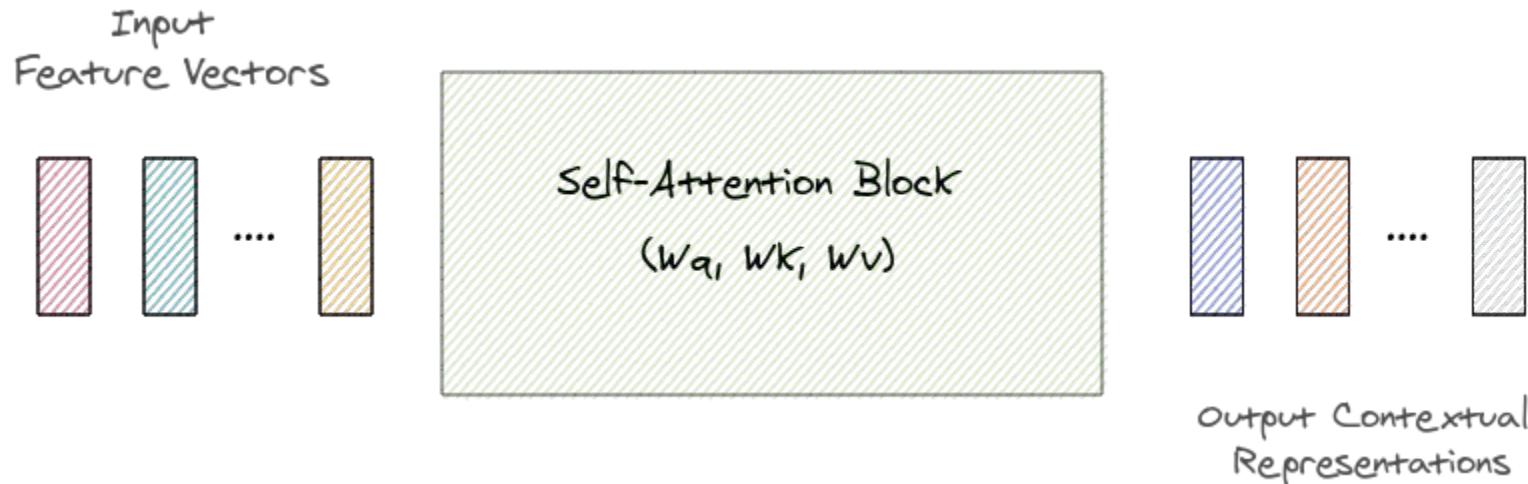


Outline

1. Motivation
2. Self-attention (SA)
3. **Multi-head Self-Attention (MHSA)**

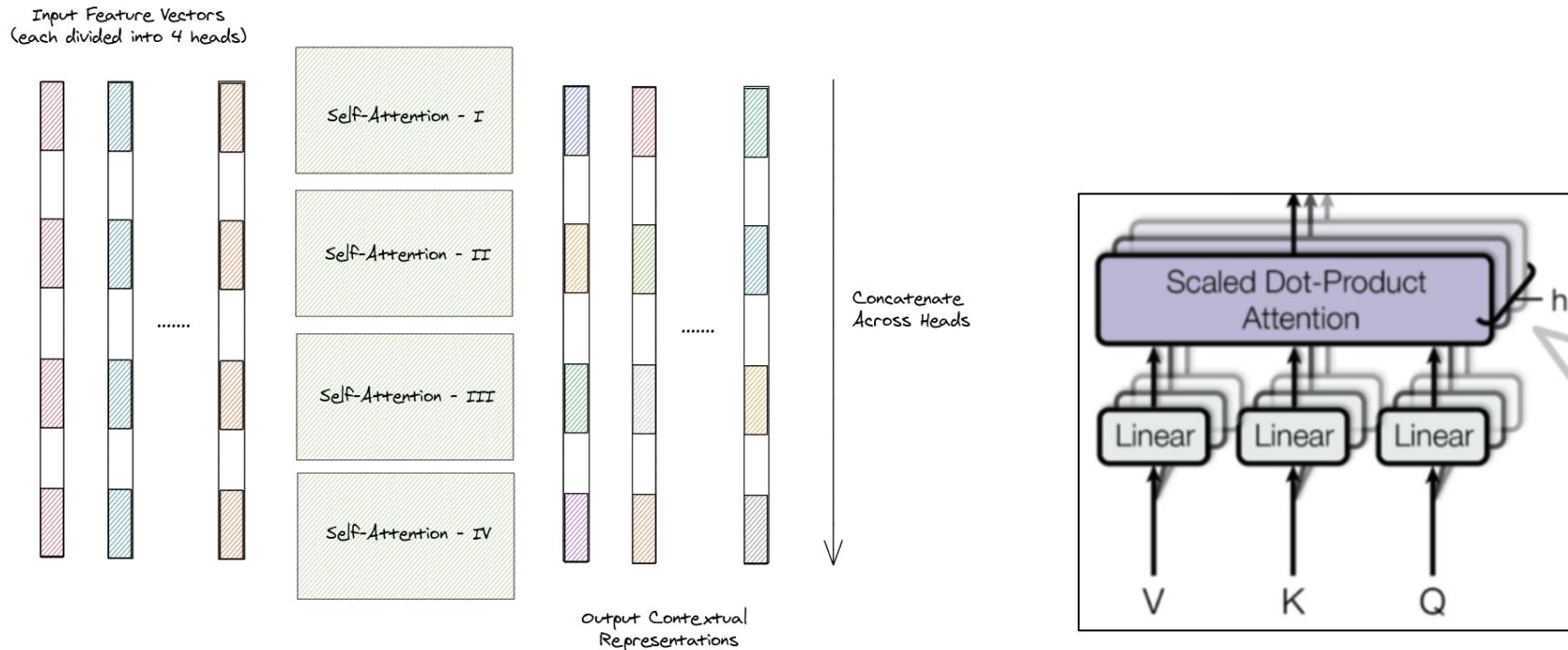
Multi-Head Self-Attention (MHSA)

In vanilla self-attention, a single set of projection matrices W^Q , W^K , W^V is used.



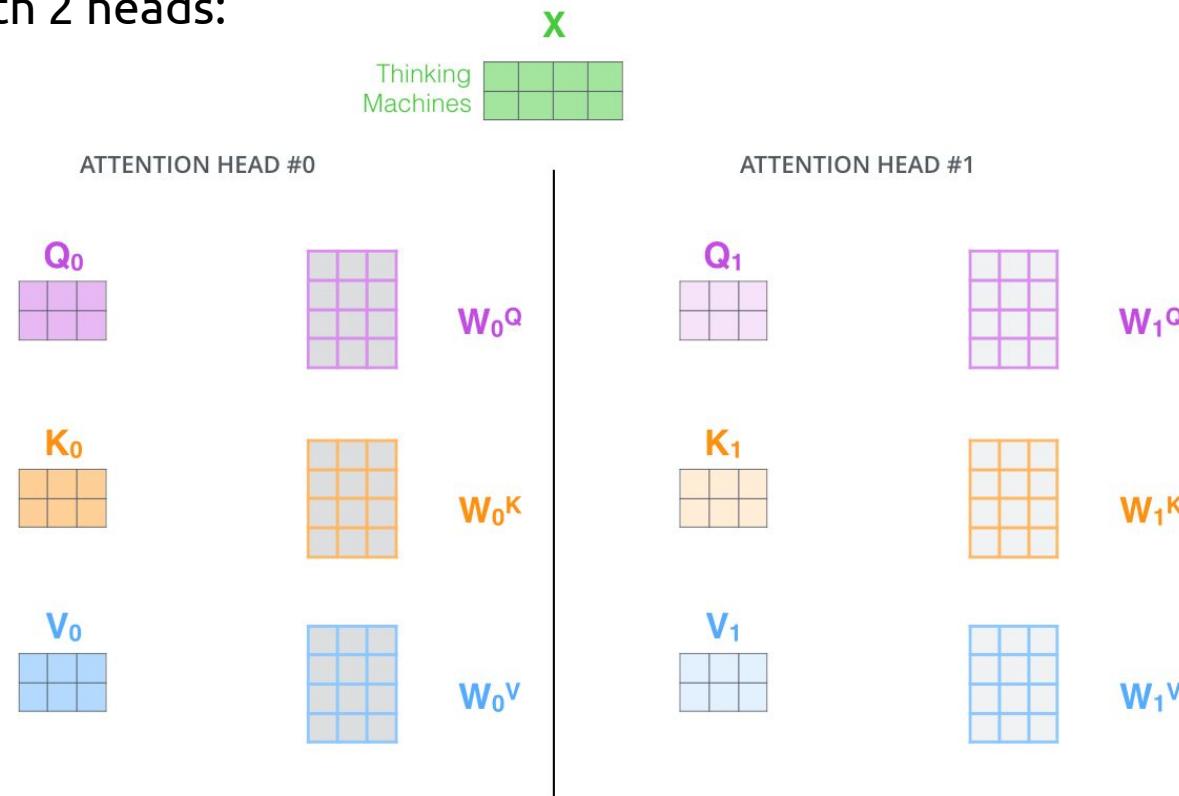
Multi-Head Self-Attention (MHSA)

In multi-head self-attention, multiple sets of projection matrices are used, and can provide different contextual representations for the same input token.



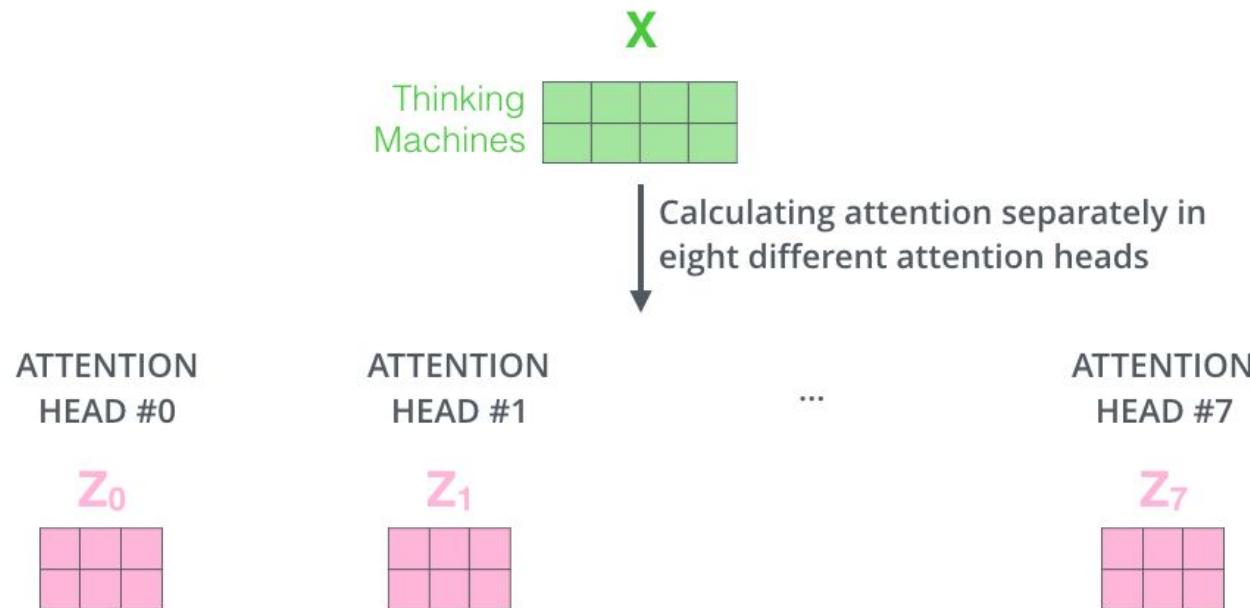
Multi-Head Self-Attention (MHSA)

Example with 2 heads:



Multi-Head Self-Attention (MHSA)

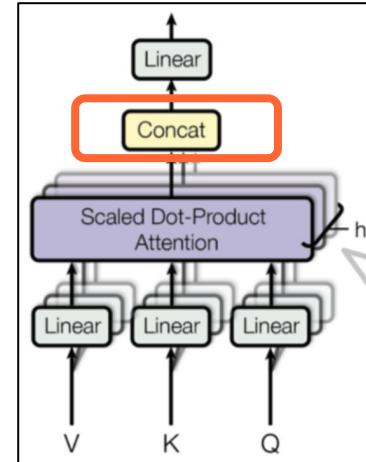
Example with 7 heads:



Multi-Head Self-Attention (MHSA)

The multi-head self-attended z_i are concatenated...

1) Concatenate all the attention heads



Multi-Head Self-Attention (MHSA)

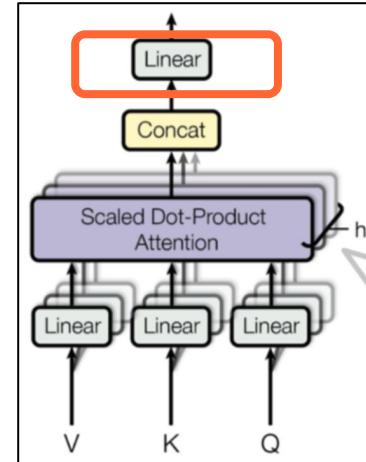
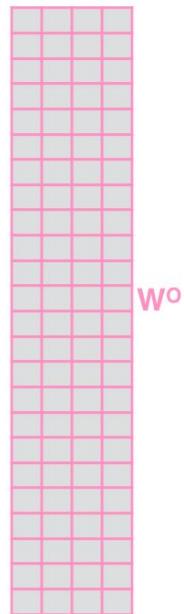
...and projected with a linear layer...

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^o that was trained jointly with the model

X



Multi-Head Self-Attention (MHSA)

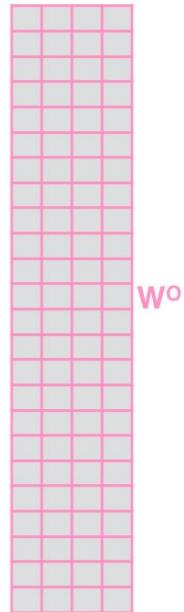
...to build a new single representation z .

1) Concatenate all the attention heads



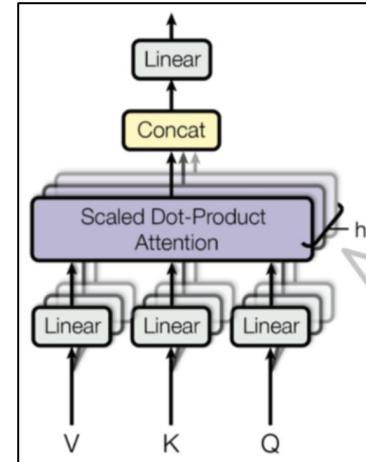
2) Multiply with a weight matrix W^o that was trained jointly with the model

x



3) The result would be the z matrix that captures information from all the attention heads. We can send this forward to the FFNN

$$= \begin{matrix} z \\ \hline \end{matrix} \quad \begin{matrix} \text{---} \\ \hline \end{matrix} \quad \begin{matrix} \text{---} \\ \hline \end{matrix}$$



Multi-Head Self-Attention (MHSA)

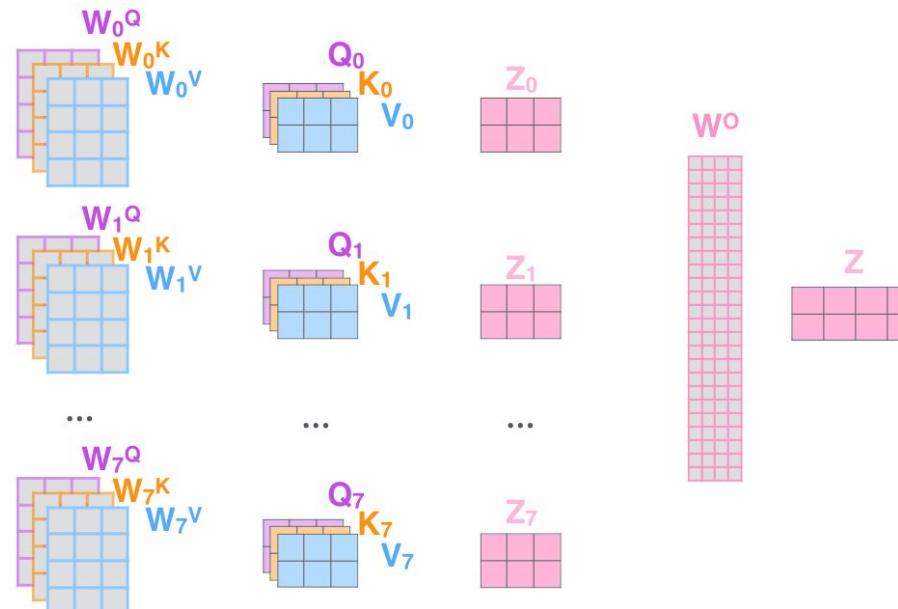
The overall MHSA scheme:

1) This is our input sentence*
2) We embed each word*

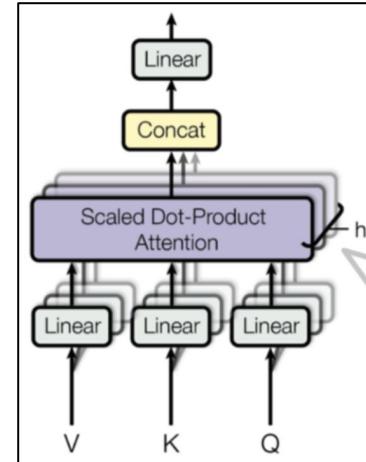
3) Split into 8 heads.
We multiply X or R with weight matrices

4) Calculate attention using the resulting $Q/K/V$ matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer

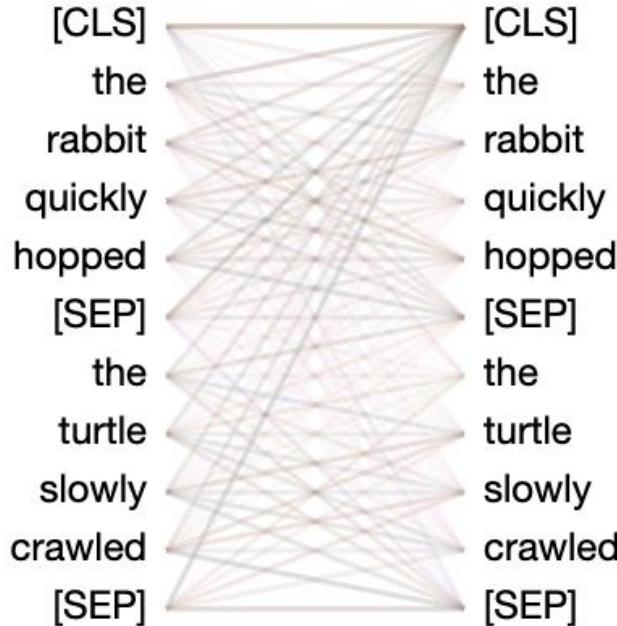
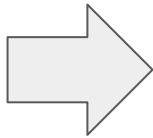


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

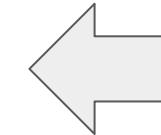


Multi-head Self-Attention: Visualization

Each colour corresponds to a head.



Blue: First head only.



Multi-color: Multiple heads.

Self-Attention and Convolutional Layers

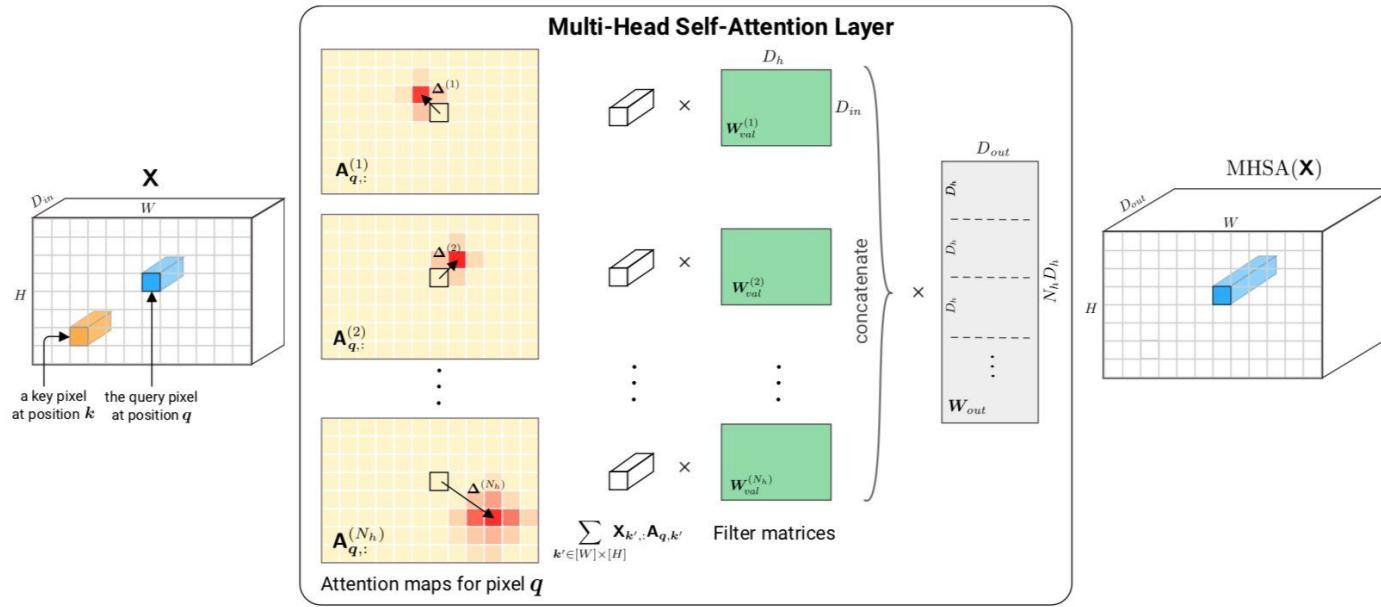


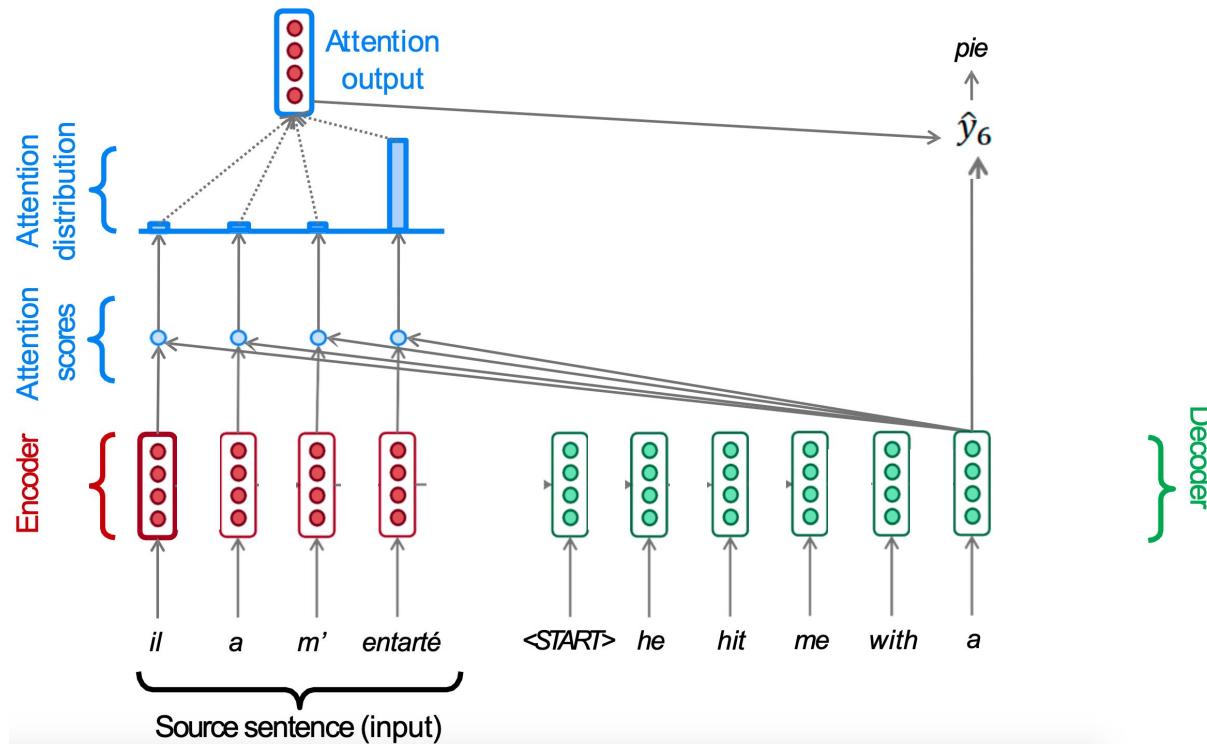
Figure 1: Illustration of a Multi-Head Self-Attention layer applied to a tensor image \mathbf{X} . Each head h attends pixel values around shift $\Delta^{(h)}$ and learn a filter matrix $\mathbf{W}_{val}^{(h)}$. We show attention maps computed for a query pixel at position q .

Outline

1. Motivation
2. Self-attention
3. Multi-head Attention
- 4. Positional Encoding**

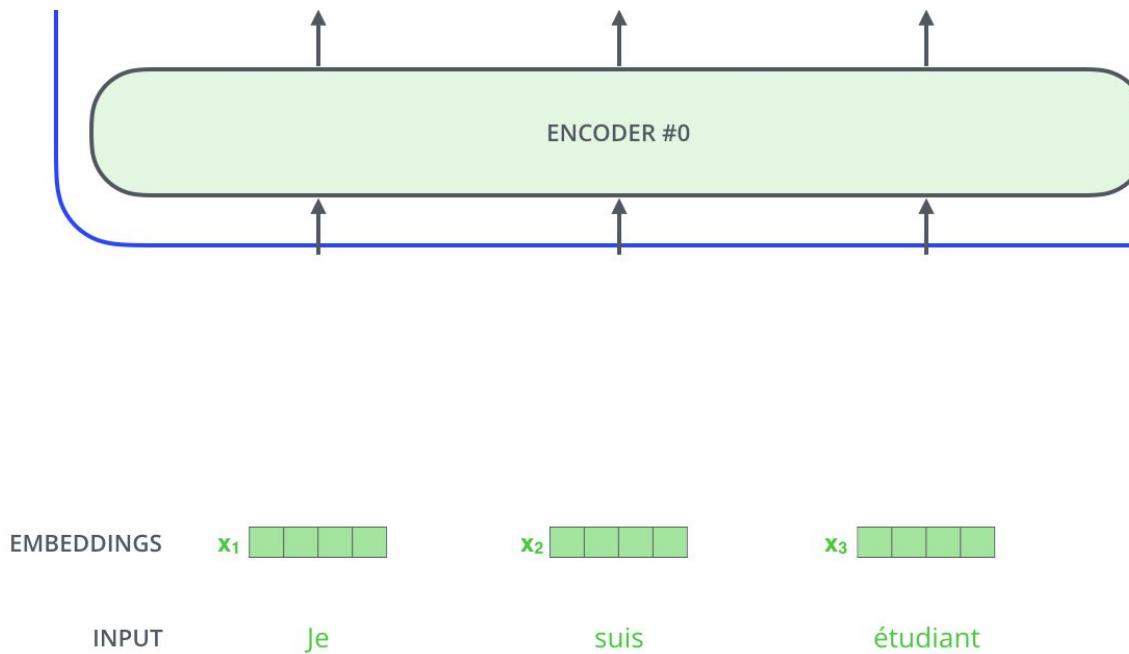
Positional Encoding

Given that the attention mechanism allows accessing all input (and output) tokens, we no longer need a memory through recurrent layers.



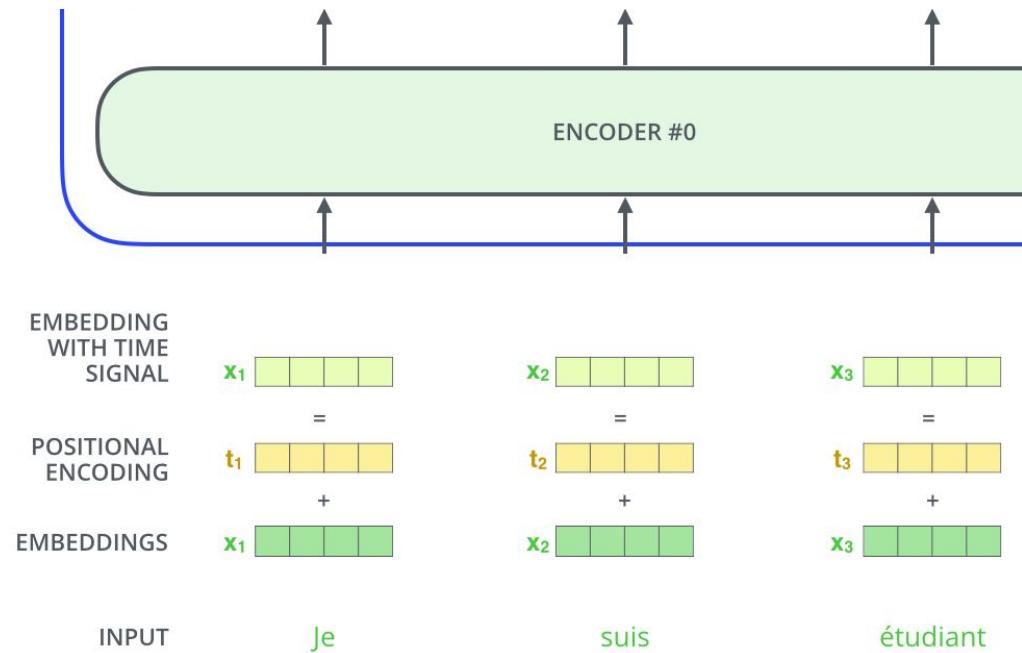
Positional Encoding

Where is the relative relation in the sequence encoded ?



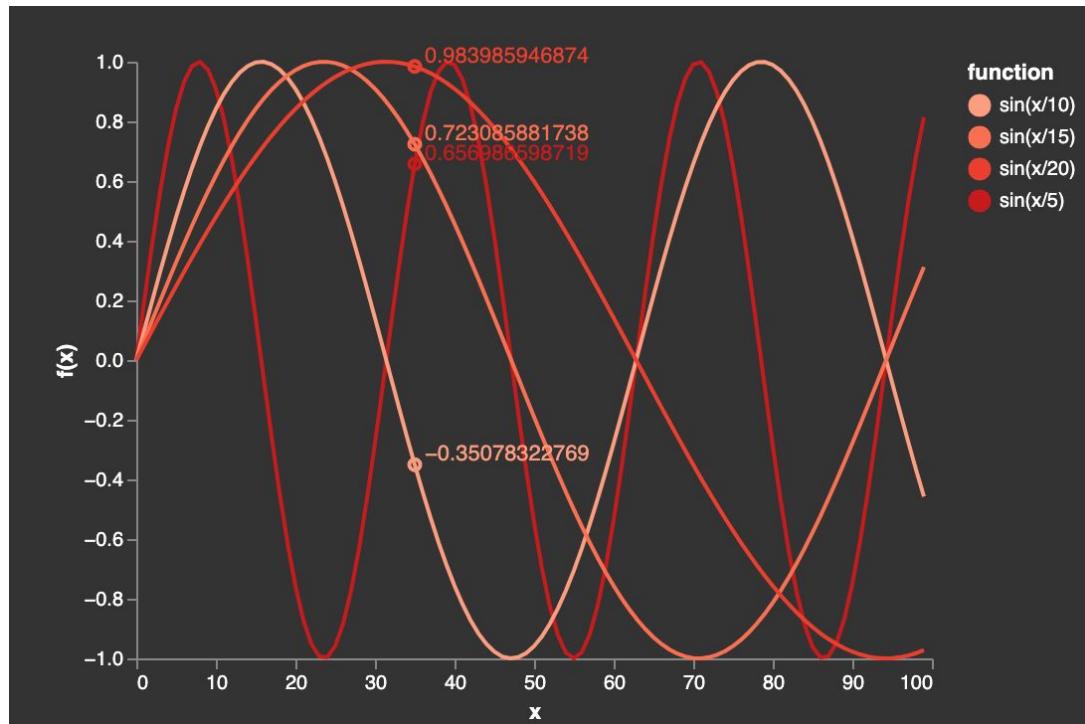
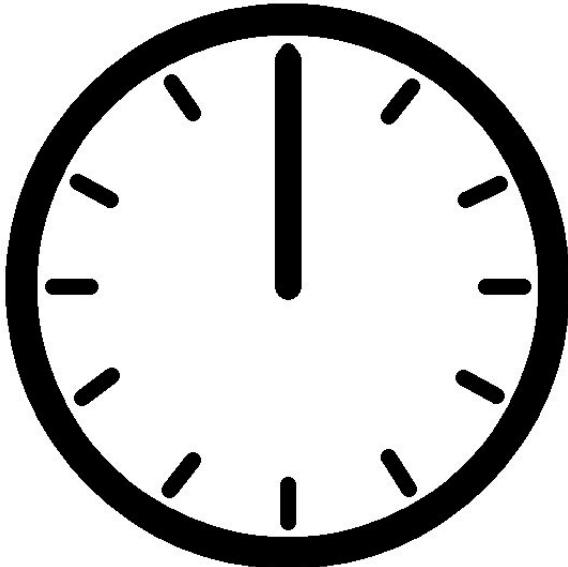
Positional Encoding

Where is the relative relation in the sequence encoded ?

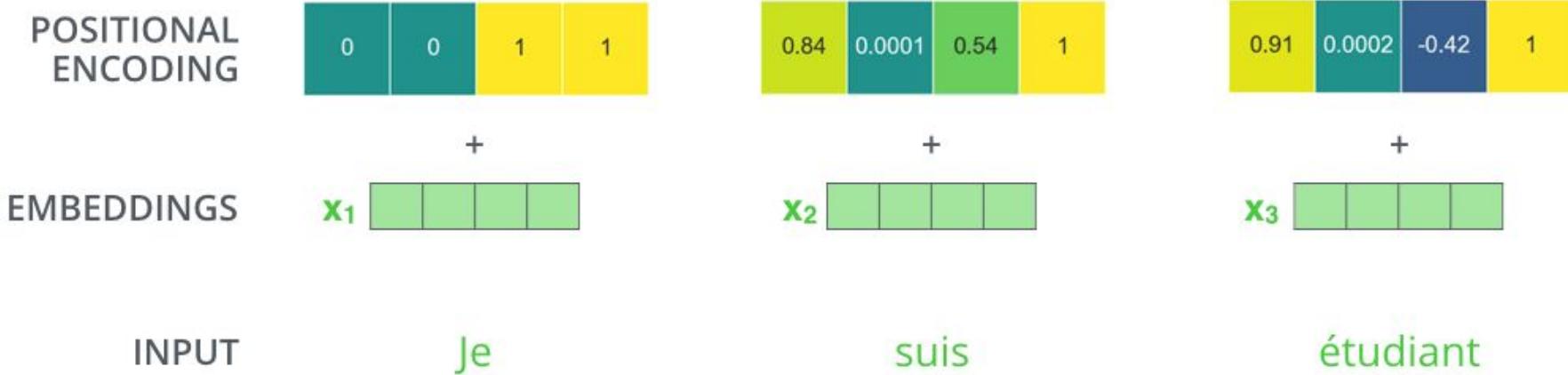


Positional Encoding

Sinusoidal functions are typically used to provide positional encodings.



Positional Encoding

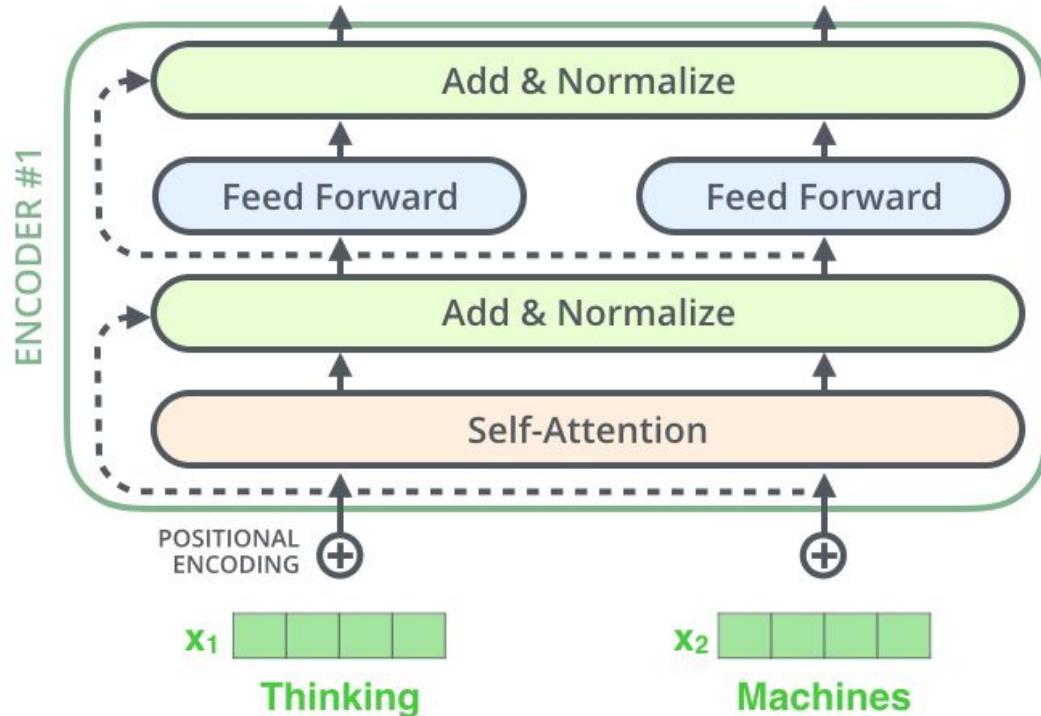


Outline

1. Motivation
2. Self-attention
3. Multi-head Attention
4. Positional Encoding
5. **Stacks of Encoder & Decoder Layers**

Non-linear & Residual Layers

- The output of the self-attention is fed into a non-linear Feed-Forward layer.
- Both SA & FFD layers include residual connections.
- Data is also normalized with layer-normalization.



Stacking Layers

Multiple stacks of SA + FF layers can be stacked in both encoder and decoder.
Example: 2 Stacked Encoders and Decoders:

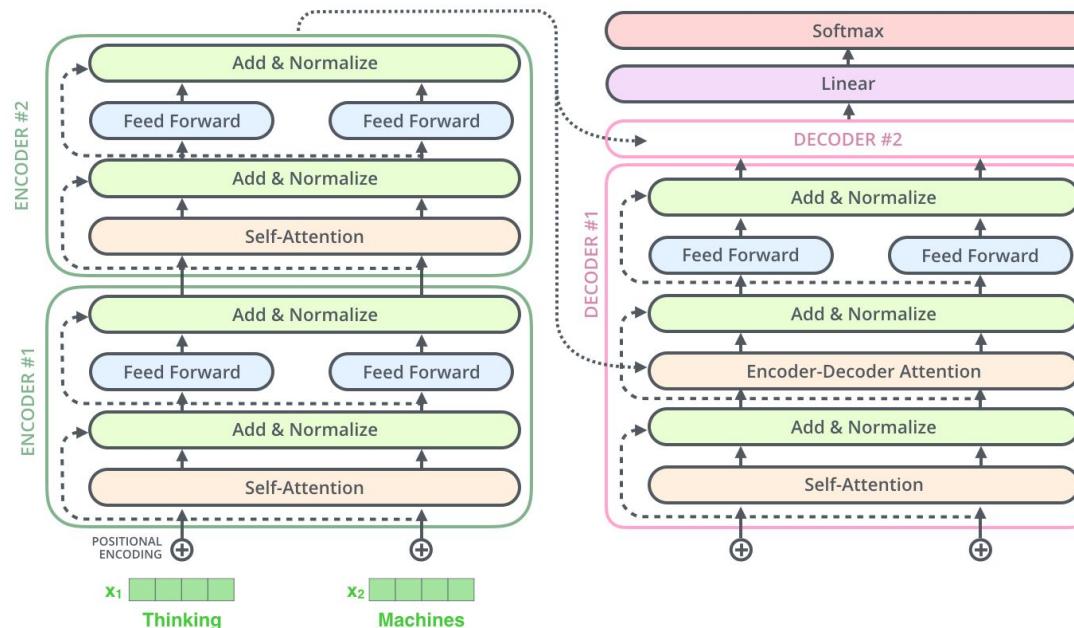


Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Stacks of Encoder & Decoder Layers

Multiple stacks of SA + FF layers can be stacked in both encoder and decoder.
Example: 2 Stacked Encoders and Decoders:

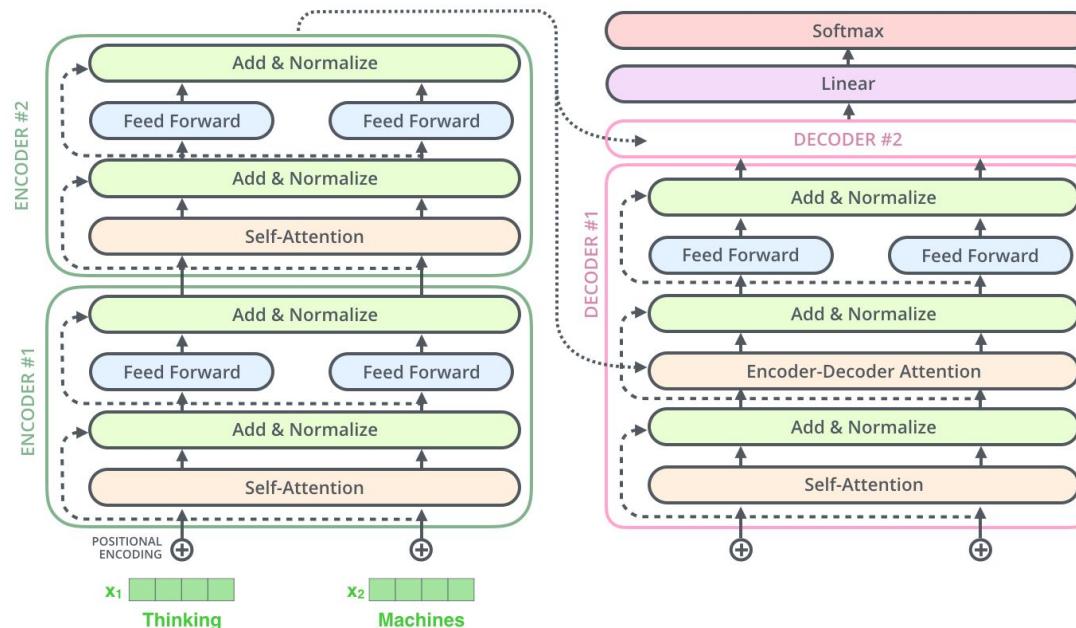


Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Stacks of Encoder & Decoder Layers

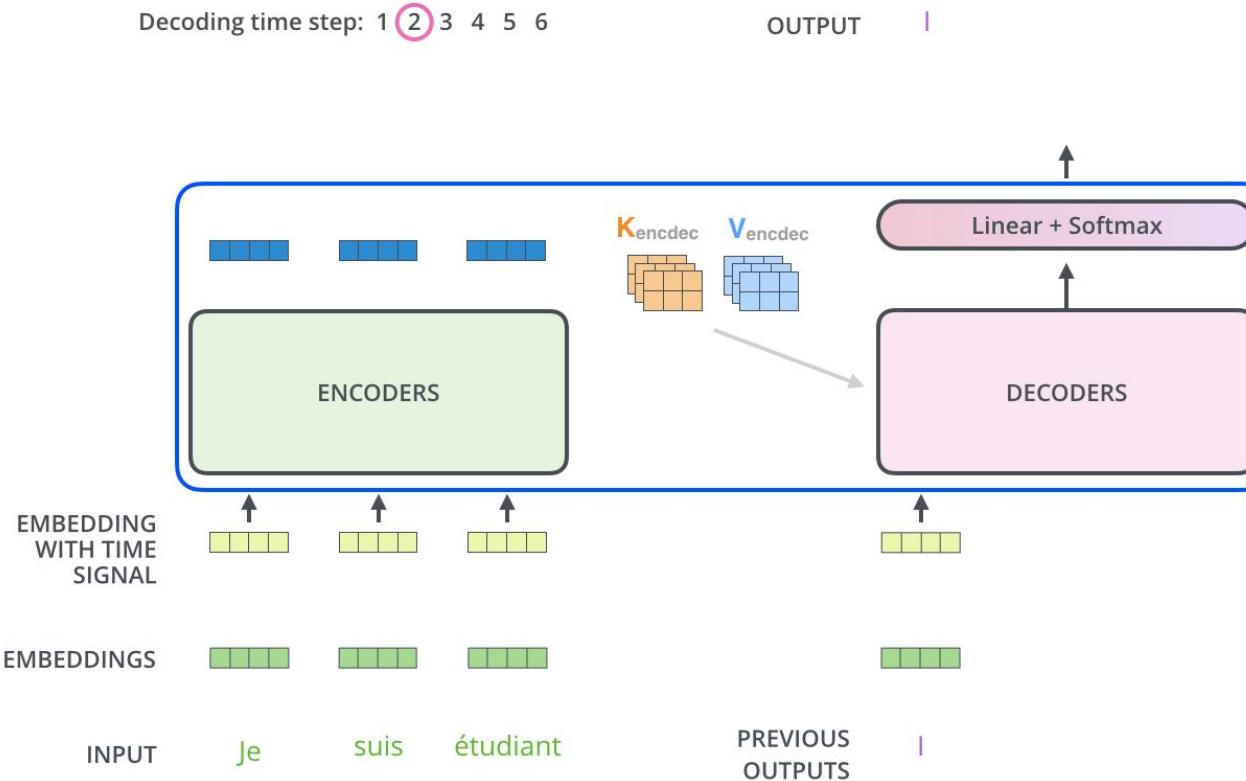


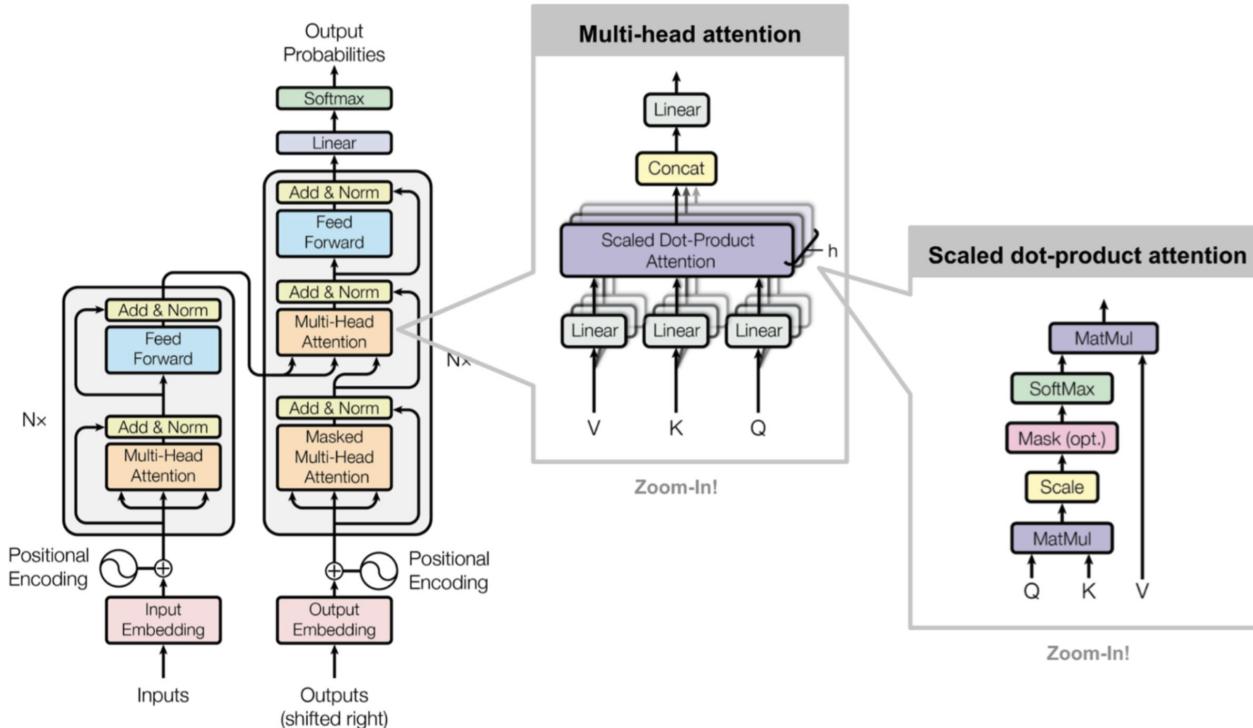
Figure: Jay Alammar, "[The illustrated Transformer](#)" (2018)

Outline

1. Motivation
2. Self-attention
3. Multi-head Attention
4. Positional Encoding
5. Stacks of Encoder & Decoder Layers
6. **The Transformer**

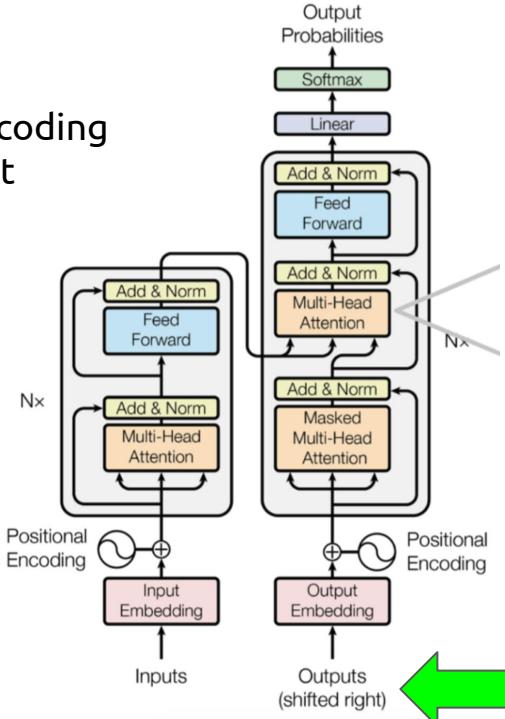
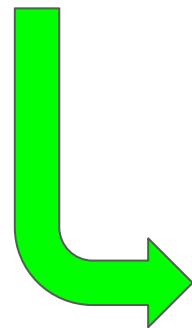
The Transformer

The Transformer removed the recurrence mechanism thanks to self-attention.



The Transformer

Positional Encoding over the input sequence.



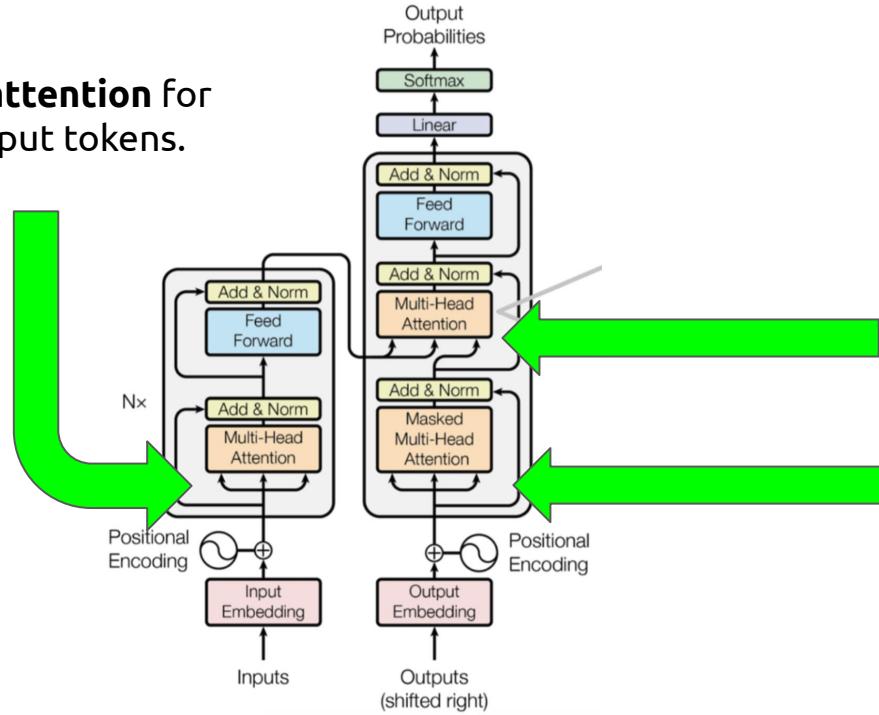
Positional Encoding over the output sequence.



Auto-regressive (at test).

The Transformer

Self-attention for the input tokens.

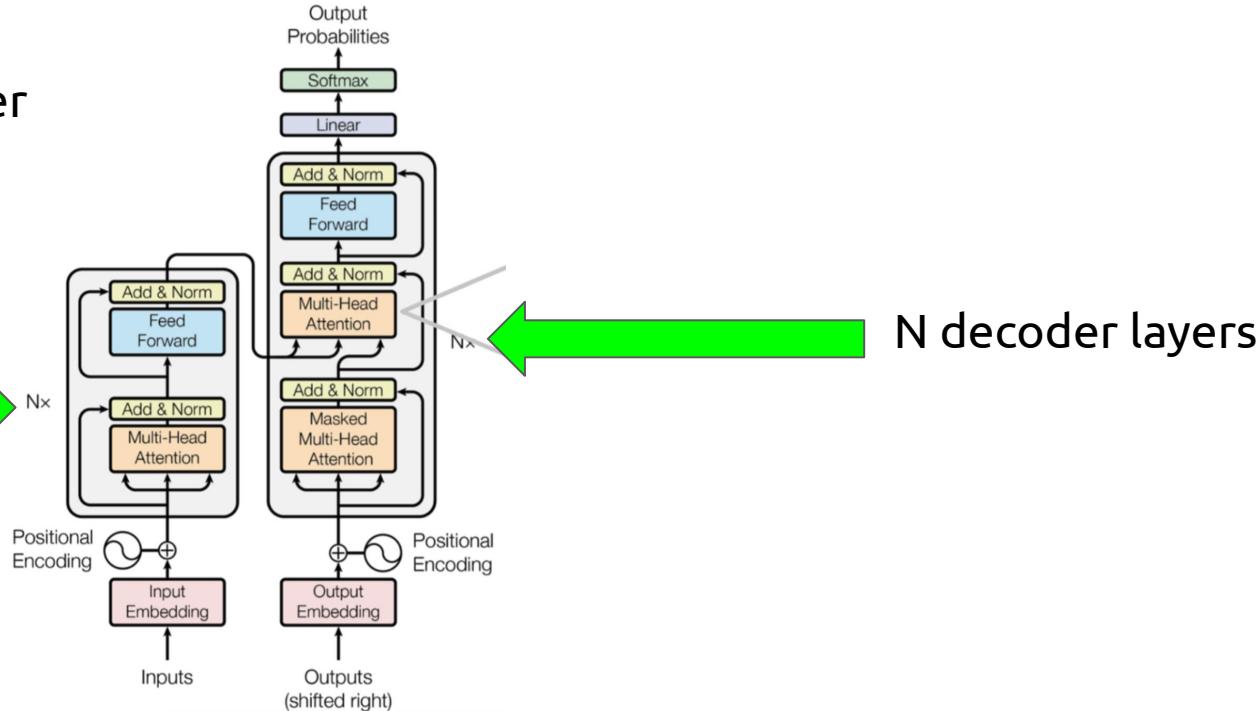
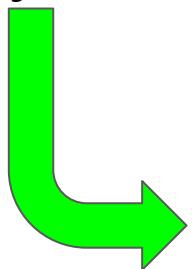


Cross-Attention (or inter-attention) between input and output tokens

Self-attention for the output tokens.

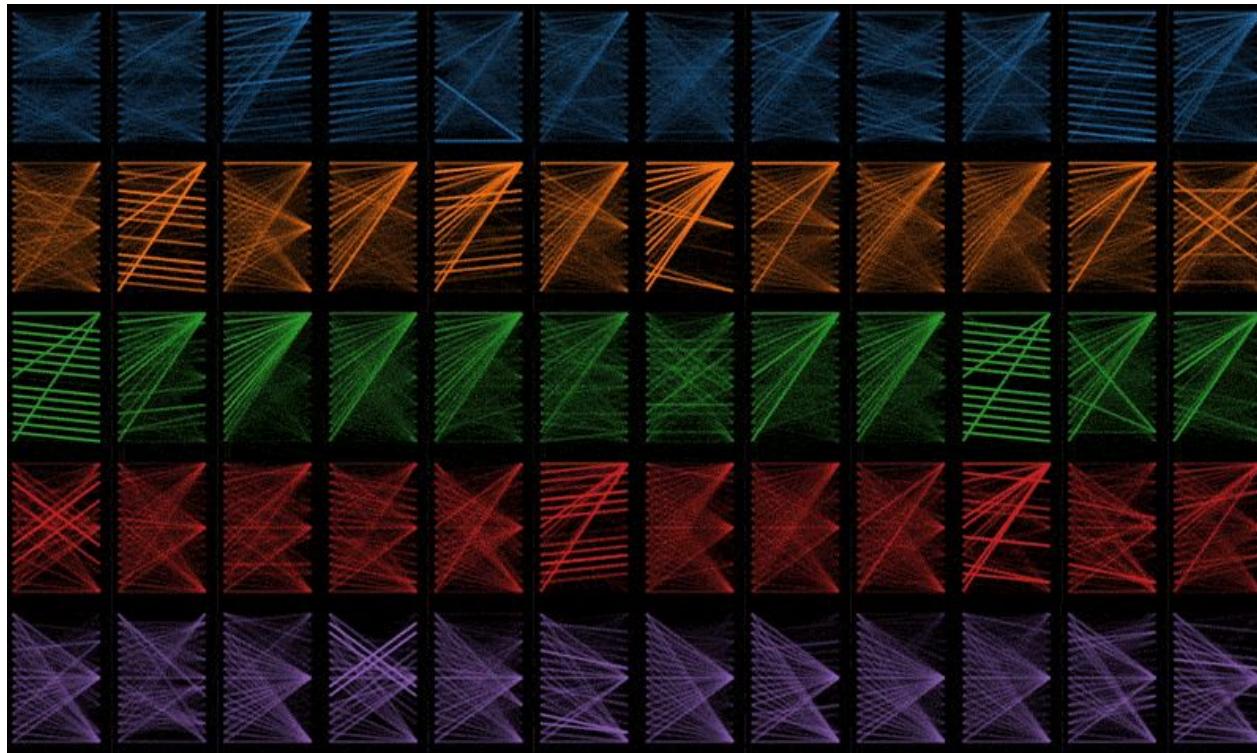
The Transformer: Layers

N encoder
layers

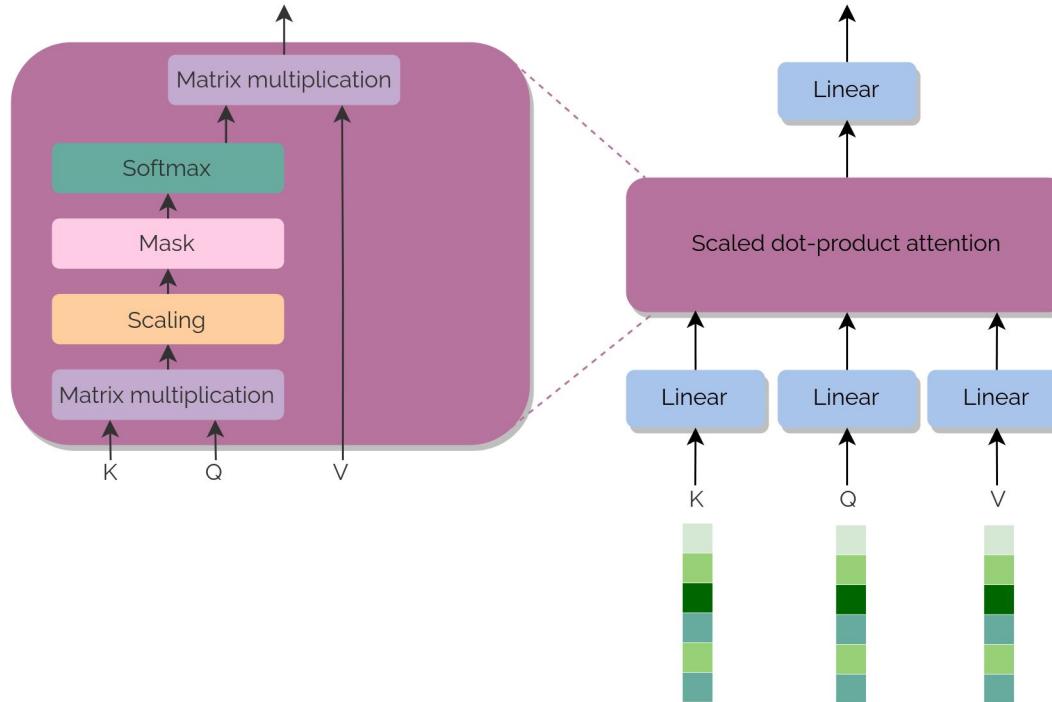


The Transformer: Layers

A birds-eye view of attention across all of the model's layers and heads



The Transformer: Visualization



Are Transformers for Language only ? NO !!

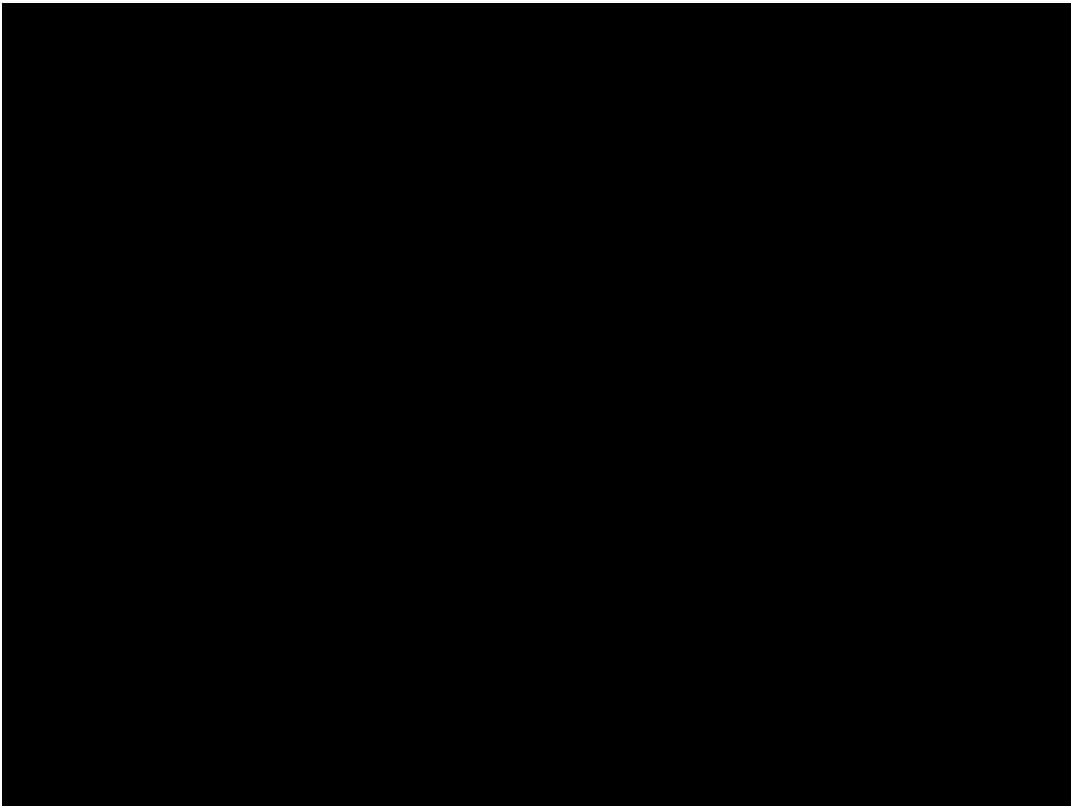


#ViT Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani et al. ["An image is worth 16x16 words: Transformers for image recognition at scale."](#) ICLR 2021. [\[blog\]](#) [\[code\]](#)

Outline

1. Motivation
2. Self-attention
3. Multi-head Attention
4. Positional Encoding
5. Stacks of Encoder & Decoder Layers
6. The Transformer

Teaser of Kostas Derpanis lecture



[Kostas Derpanis, York University](#) (2022) - [@CSProfKGD](#)

(extra) PyTorch Lab on Google Colab

The screenshot shows a Google Colab interface with the following details:

- Title:** lab_transformer_todo.ipynb
- File menu:** File, Edit, View, Insert, Runtime, Tools, Help
- Toolbar:** + Code, + Text, Copy to Drive
- Section:** The Transformer
- Text:** Created by [Gerard I. Gallego](#) for the Postgraduate Course in Artificial Intelligence with Deep Learning (UPC School, 2021).
- Text:** In this lab we will learn about the [Transformer](#), a popular architecture that revolutionized Deep Learning a few years ago.
- Text:** This architecture was firstly designed for Machine Translation. In this field, Recurrent Neural Networks (e.g. LSTM) had been the state-of-the-art since the [introduction of the Attention mechanism](#) in 2015. The Transformer surpassed them by introducing a key idea: getting rid of any recurrence and mainly using the ([Self-](#))Attention mechanism, as the title of the paper states: [Attention is All you Need](#).
- Text:** Actually, Transformer-based architectures are used in many fields beyond Machine Translation. First, many models arised for other text-related tasks (e.g. [BERT](#) or [GPT-3](#)), but now they're also used in other fields, like [Speech processing](#) and [Computer Vision](#).
- Text:** Throughout this notebook, we will build our own Transformer, and you'll understand module by module how this architecture works. Once it's finished, we will train it with a dummy dataset and we will try to interpret the results. Finally, we will see how to use pre-trained Transformers from [Hugging Face](#), for other tasks, like Sentiment Analysis and Text Generation.
- Code:**

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import DataLoader

import math
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```
- Section:** Building your own Transformer



Gerard Gallego
gerard.ion.gallego@upc.edu

PhD Candidate
Universitat Politècnica de Catalunya
Technical University of Catalonia



The promotional page features the following elements:

- Title:** DEEP LEARNING FOR ARTIFICIAL INTELLIGENCE
- Text:** 3rd Master Course UPC ETSETB TelecomBCN Barcelona. Autumn 2019.
- Image:** Aerial view of Barcelona cityscape.
- Section:** Instructors
- Profile Pictures:** Xavier Gil-Ortego, Marta R. Costa-Jussà, Noé Casas, Verònica Vilaplana, Ramon Morros, Javier Ruiz, Albert Pumarola, Jordi Torres.
- Section:** Organizers
- Logos:** UPC, Universitat Politècnica de Catalunya, BARCELONATECH, telecomBCN, Google Cloud, GitHub Education.
- Text:** + info: <http://bit.ly/dlai2019>

- DL resources from UPC Telecos:
- [Lectures](#) (with Slides & Videos)
- [Labs](#)

Software

- [Transformers](#) in HuggingFace.
- [GPT-Neo](#) by EleutherAI
 - Similar results to GPT-3, but smaller and open source.
- Andrej Karpathy, [minGPT](#) (2020).



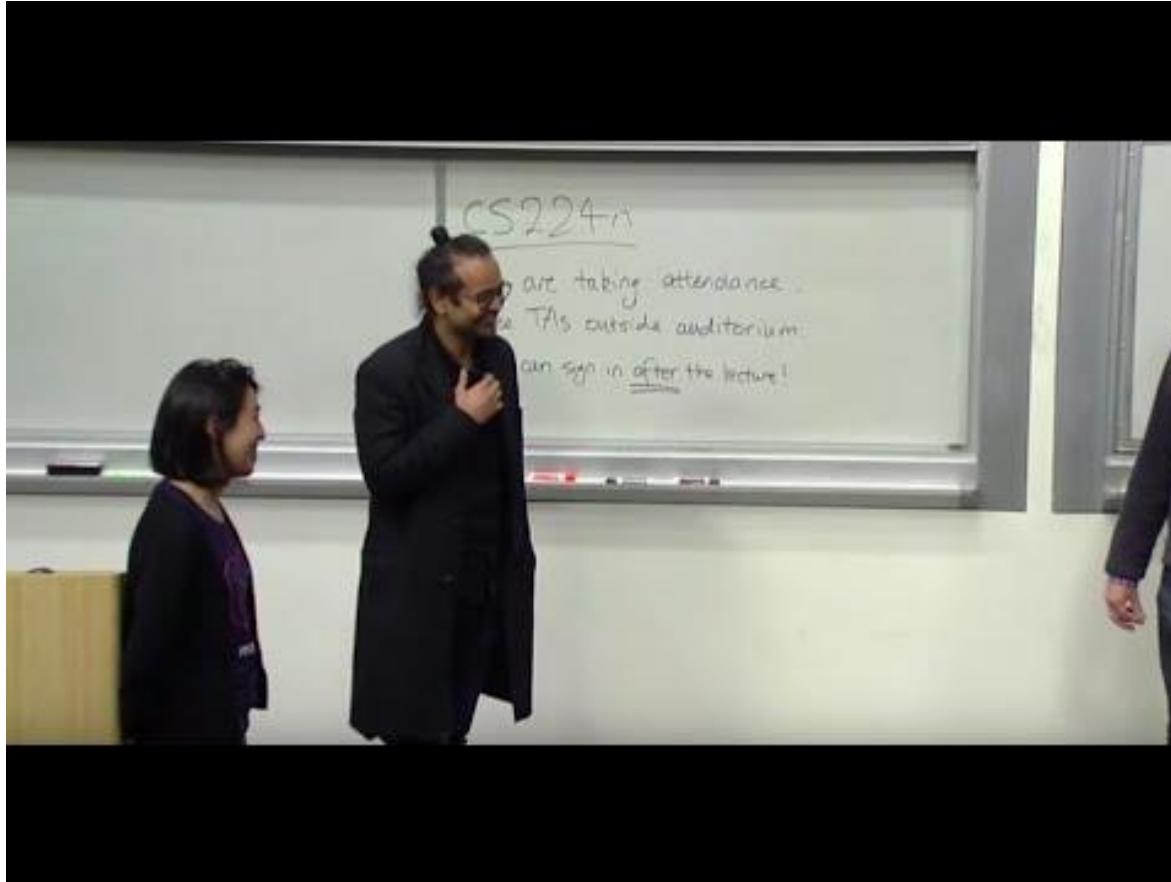
available GPT implementations



minGPT



Learn more



Ashish Vaswani, Stanford CS224N 2019.

Learn more

- Tutorials
 - Lilian Weg, [The Transformer Family](#). Lil'Log 2020
- Twitter threads
 - [Christian Wolf](#) (INSA Lyon)
- Scientific publications
 - **#Perceiver** Jaegle, Andrew, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. ["Perceiver: General perception with iterative attention."](#) arXiv preprint arXiv:2103.03206 (2021).
 - **#Longformer** Beltagy, Iz, Matthew E. Peters, and Arman Cohan. ["Longformer: The long-document transformer."](#) arXiv preprint arXiv:2004.05150 (2020).
 - Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. [Transformers are RNNs: Fast autoregressive transformers with linear attention](#). ICML 2020.
 - Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, Razvan Pascanu, ["Multiplicative Interactions and Where to Find Them"](#). ICLR 2020. [[tweet](#)]
 - Self-attention in language
 - Cheng, J., Dong, L., & Lapata, M. (2016). [Long short-term memory-networks for machine reading](#). arXiv preprint arXiv:1601.06733.
 - Self-attention in images
 - Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., & Tran, D. (2018). [Image transformer](#). ICML 2018.
 - Wang, Xiaolong, Ross Girshick, Abhinav Gupta, and Kaiming He. ["Non-local neural networks."](#) In CVPR 2018.

Learn more

jupyter transformer_language_modeling Last Checkpoint: 3 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 [pykernel] O

Lab 01: Language Modeling with Transformers - Demo

```
In [2]: # For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
# find automatically the path of the folder containing "file_name" :
file_name = "transformer_language_modeling.ipynb"
# change current path to the folder containing "file_name"
os.chdir(path_to_file)
# change current path to the folder containing "file_name"
os.chdir(path_to_file)
!pwd

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab01_language_model/
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab01_language_model

In [3]: import torch
import torch.nn.functional as F
import torch.nn as nn
import math
import time
import utils
```

GPU

It is recommended to run this code on GPU:

- Time for 1 epoch on GPU: 48 sec w/ Google Colab Tesla P100-PCIE-16GB

```
In [4]: device= torch.device("cuda")
device= torch.device("cpu")
print(device)

if torch.cuda.is_available():
    print('cuda available with GPU:',torch.cuda.get_device_name(0))
```

jupyter transformer_translation Last Checkpoint: 3 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 [pykernel] O

Lab 02: Sequence-To-Sequence with Transformers - Demo

The task is to learn to memorize an input sequence of length 100 and output the same sequence of length 100 but shifted by one word in the future.

For example, the input sequence is "some analysts expect oil prices to remain relatively" and the output sequence is "analysts expect oil prices to remain relatively high".

```
In [50]: # For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
# find automatically the path of the folder containing "file_name" :
file_name = "transformer_translation.ipynb"
# change current path to the folder containing "file_name"
path_to_file = subprocess.check_output('find . -type f -name ' + str(file_name), shell=True).decode("utf-8")
path_to_file = path_to_file.replace(file_name,'').replace('\n','')
# if previous search failed or too long, comment the previous line and simply write down manually the path below :
# path_to_file = '/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation/'
# change current path to the folder containing "file_name"
os.chdir(path_to_file)
!pwd

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation/
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation
```

```
In [51]: import torch
import torch.nn.functional as F
import torch.nn as nn
import math
import time
import utils
```

GPU

It is recommended to run this code on GPU:

- Time for 1 epoch on GPU: xxx sec w/ Google Colab Tesla P100-PCIE-16GB

```
In [52]: device= torch.device("cuda")
#device= torch.device("cpu")
print(device)
```

cuda

Questions ?

Undergradese

What undergrads ask vs. what they're REALLY asking

"Is it going to be an open book exam?"

Translation: "I don't have to actually memorize anything, do I?"

"Hmm, what do you mean by that?"

Translation: "What's the answer so we can all go home."

"Are you going to have office hours today?"

Translation: "Can I do my homework in your office?"

"Can i get an extension?"

Translation: "Can you re-arrange your life around mine?"

"Is this going to be on the test?"

Translation: "Tell us what's going to be on the test."

"Is grading going to be curved?"

Translation: "Can I do a mediocre job and still get an A?"

