Master in
Computer Vision
Barcelona

Module M6 : Video Analysis

Lectures 7: Transformers

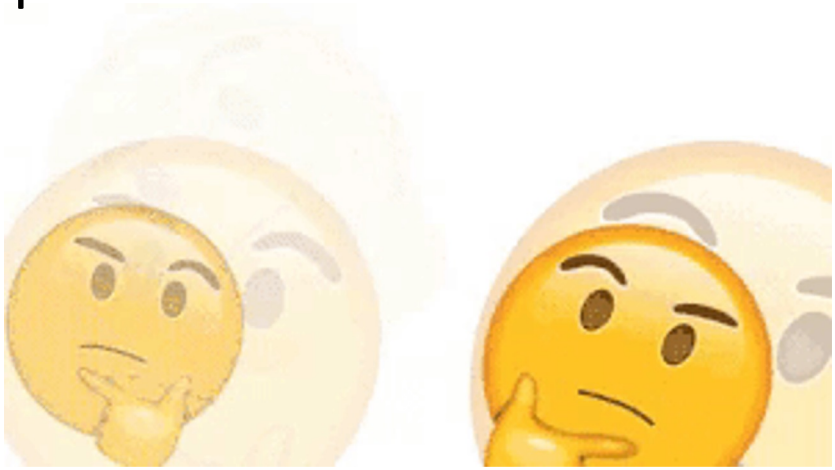
Lecturer: Ramon Morros
Slides: Javier Ruiz Hidalgo

Outline

- Introduction
- Self-attention
- Multi-head attention
- Deep layers
- Positional encoding
- The Transformer

Motivation for transformers

- Despite LSTMs and GRUs, recursive neural networks still need attention mechanism to better deal with dependencies
- But if attention gives us access to any state.. Maybe we don't need the recursion?
- We can go a step forward and ask **is attention all we need?**

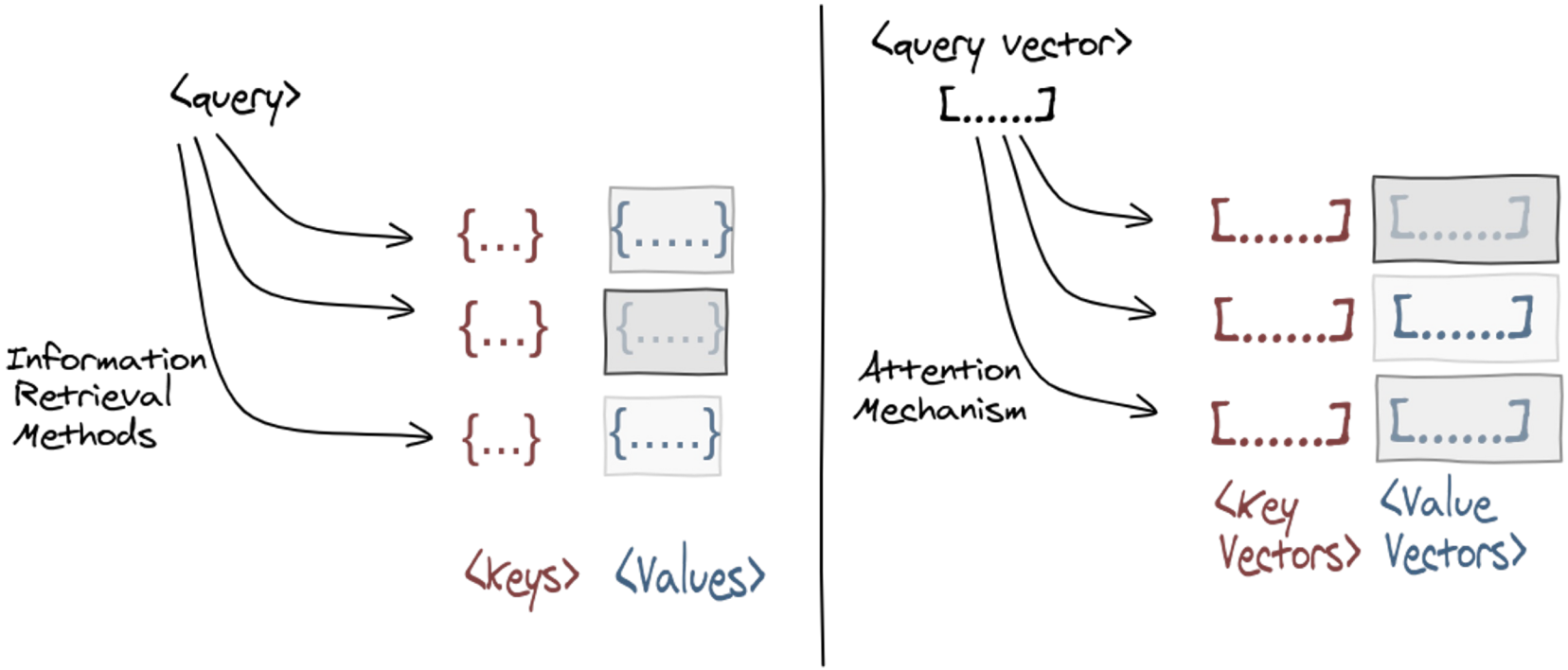


- Spoiler alert! Not quite…

A. Vaswani, et al. Attention is all you need. NeurIPS, 2017.

Attention

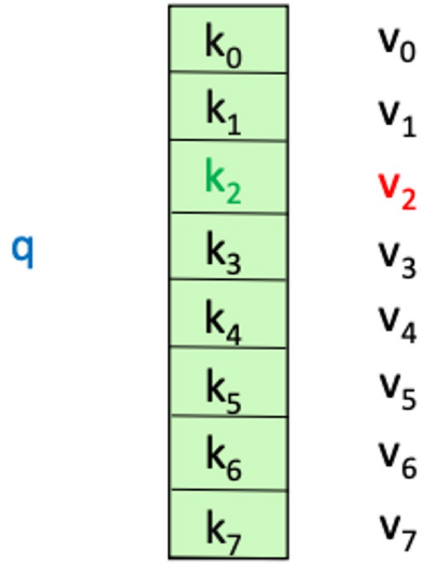
- Attention is a mechanism to compute a **context vector** for a **query** as a weighted sum of **values**
  - Improves the performance of recurrent neural networks

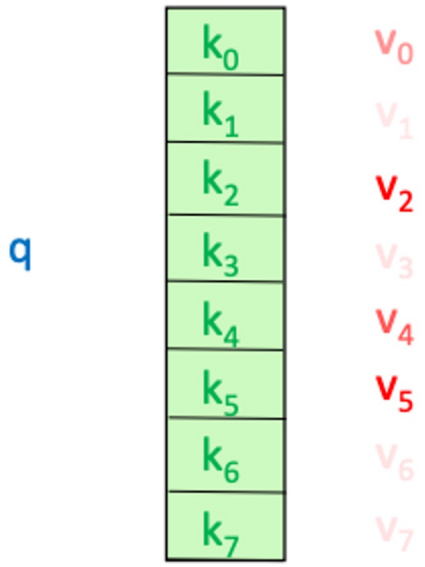


Nikhil Shah, "Attention? An Other Perspective! [Part 1]", 2020.

Attention recap (1)

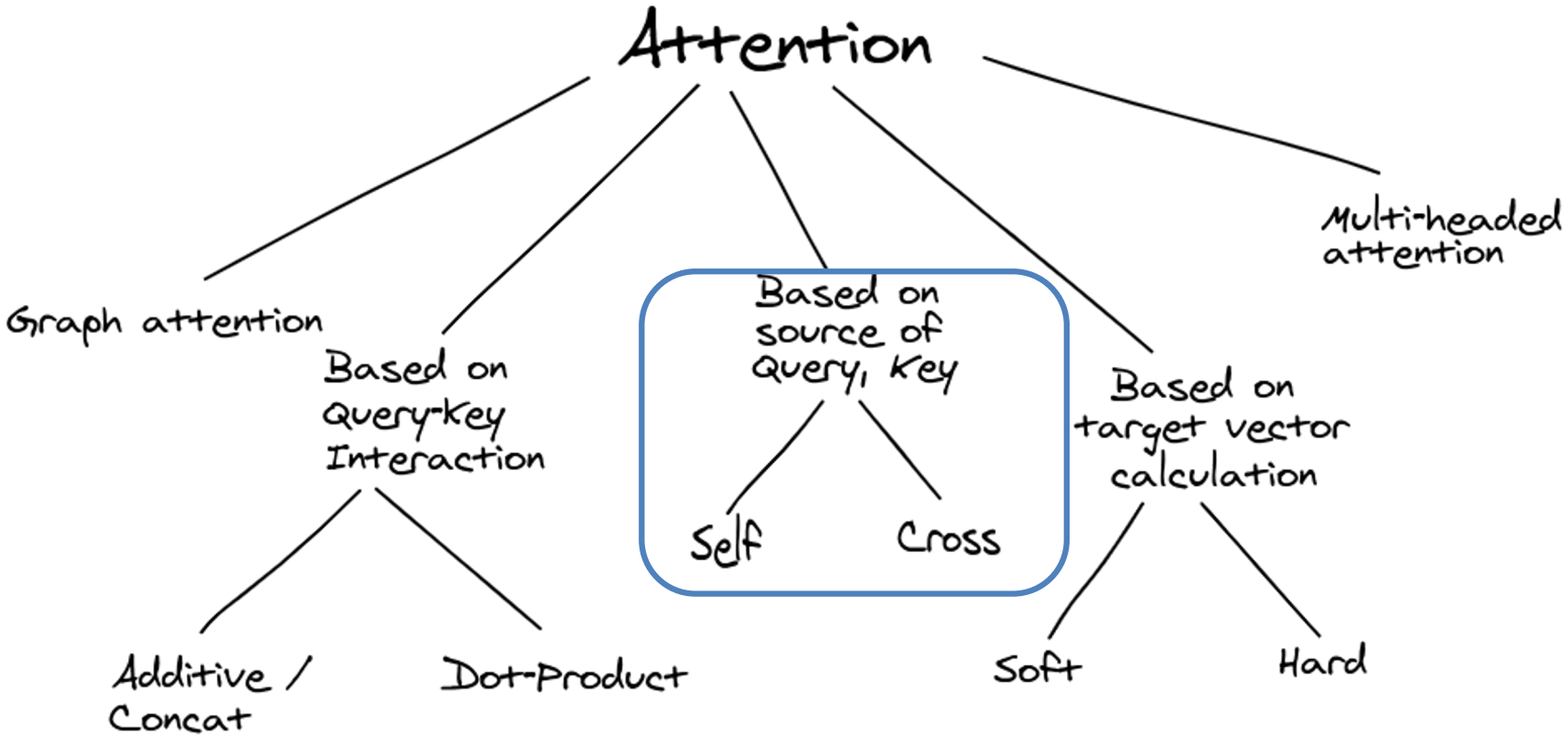
- Let's think of attention as a "fuzzy" or approximate hashtable:
  - To look up a value, we compare a query against keys in a table.
  - In a hashtable (shown on the bottom left):
    - Each query (hash) maps to exactly one key-value pair.
  - In attention (shown on the bottom right):
    - Each query matches each key to varying degrees.
    - We return a sum of values weighted by the query-key match.





Anna Goldie, John Hewitt, Standford cs224n course, 2022.

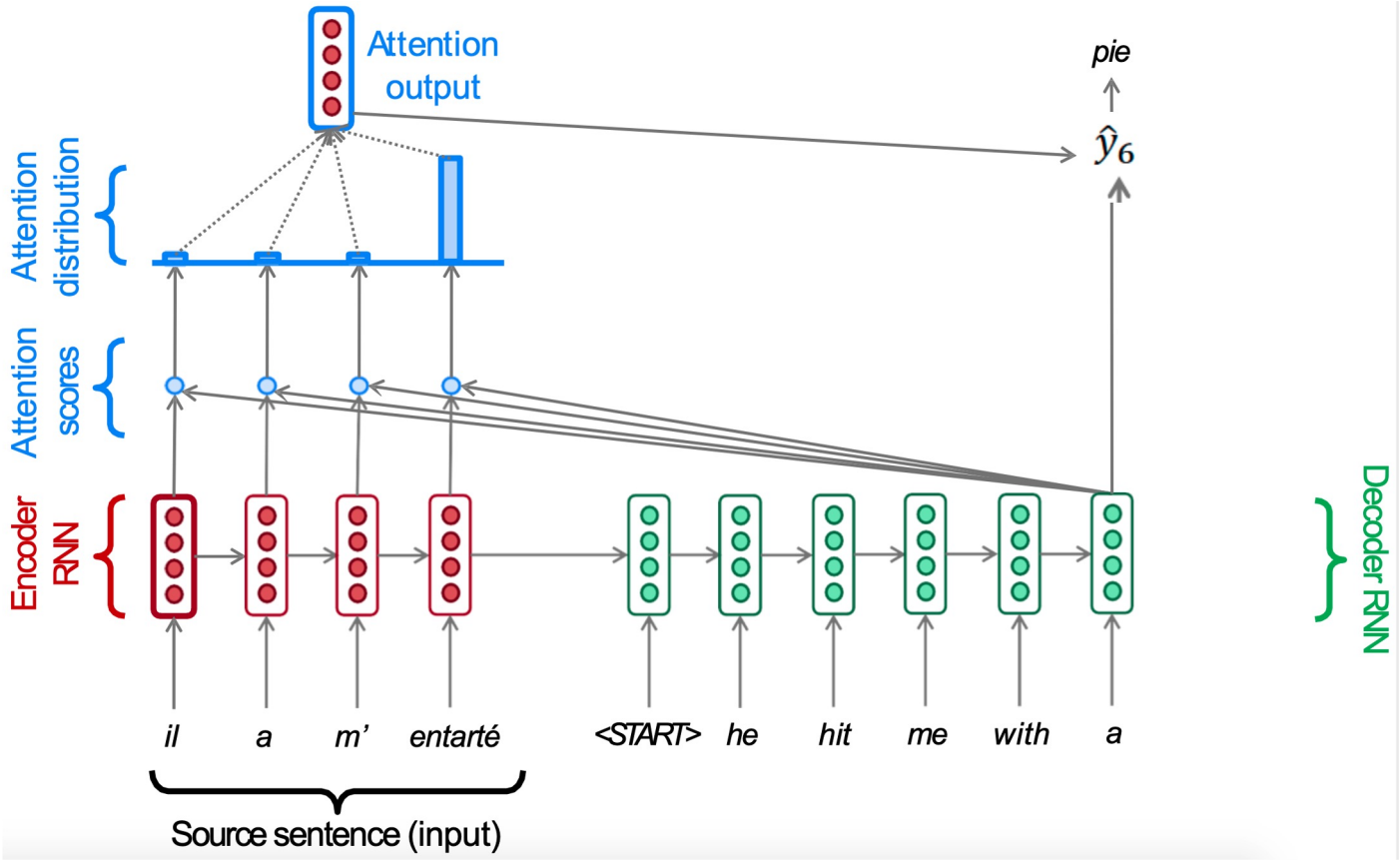
Attention recap (2)



Nikhil Sha, "Attention? An other Perspective!", 2020.

Cross-attention

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

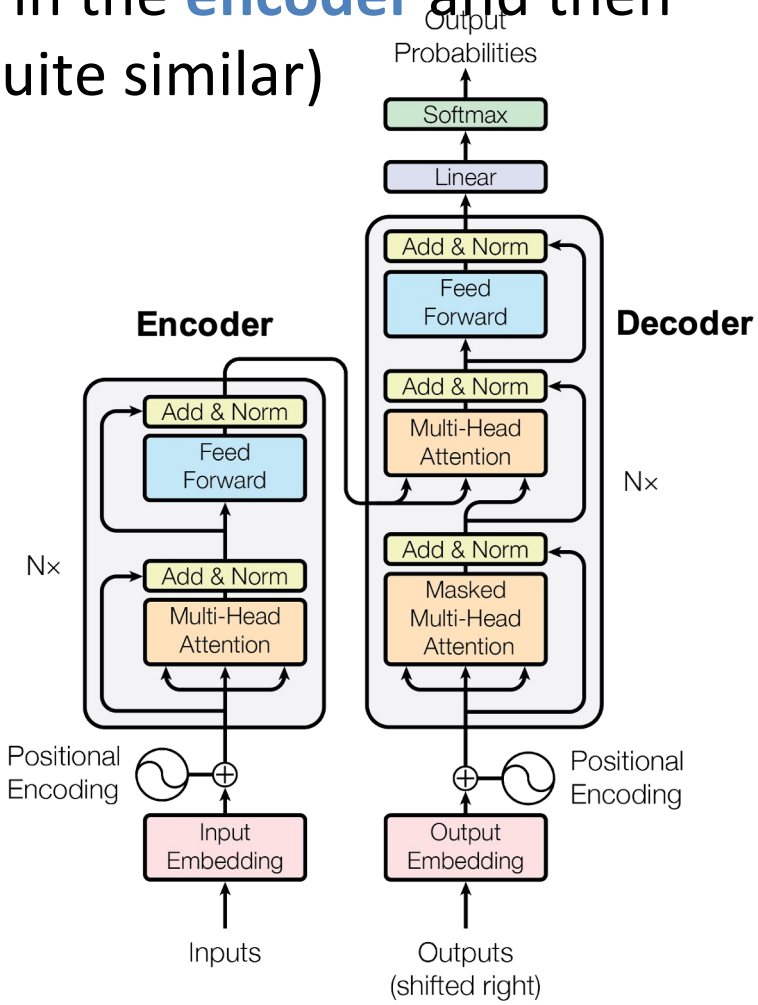


Abigail See, Matthew Lamm, (Stanford CS224N), 2020.

Is attention all you need?

- The **transformer** is a very powerful architecture that exploits attention to the max
- We will first review the main ideas in the **encoder** and then the **decoder** (but fortunately it is quite similar)

Courtesy of Paramount Pictures



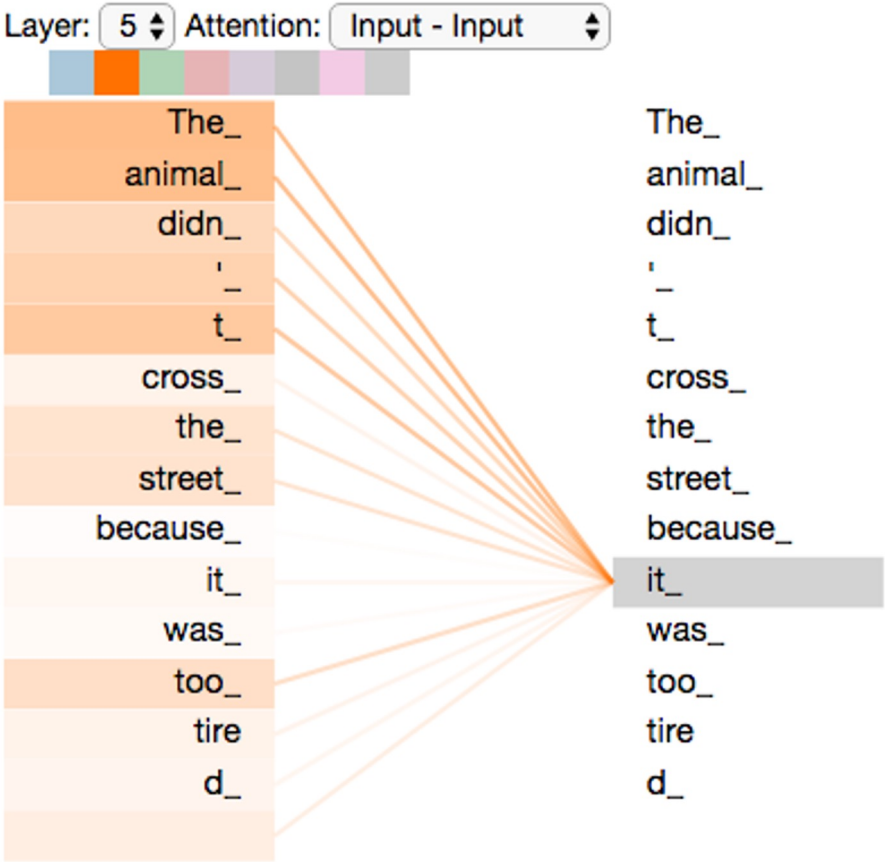
A. Vaswani, et al. Attention is all you need. NeurIPS, 2017.

Outline

- Introduction
- **Self-attention**
- Multi-head attention
- Positional encoding
- Deep layers
- The Transformer

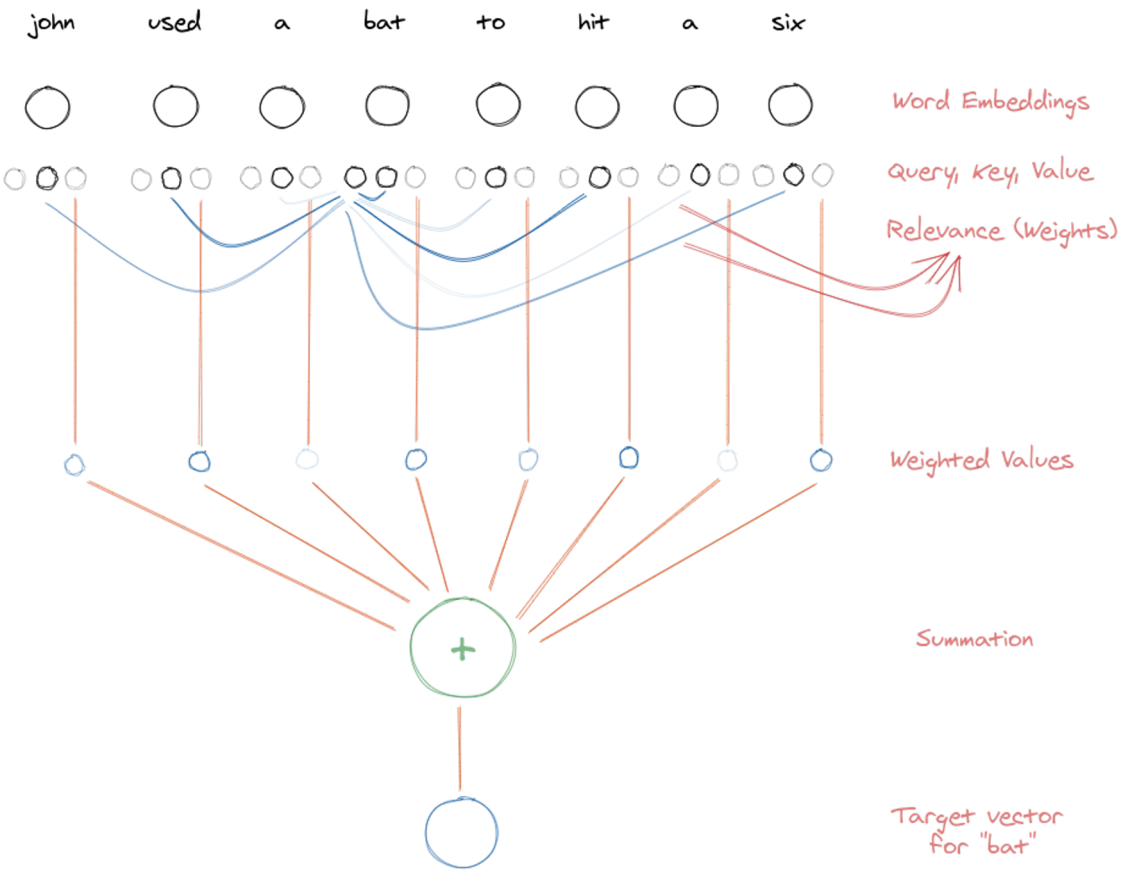
Self-attention (1)

- **Self-attention** refers to attending to other elements from the **same** sequence



Jay Alammar, The Illustrated Transformer, 2018.
Z. Lin, et al. A structured self-attentive sentence embedding, ICLR, 2017.

Self-attention (2)



Nikhil Sha, "Attention? An other Perspective!", 2020.

Queries:

$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q$

Keys:

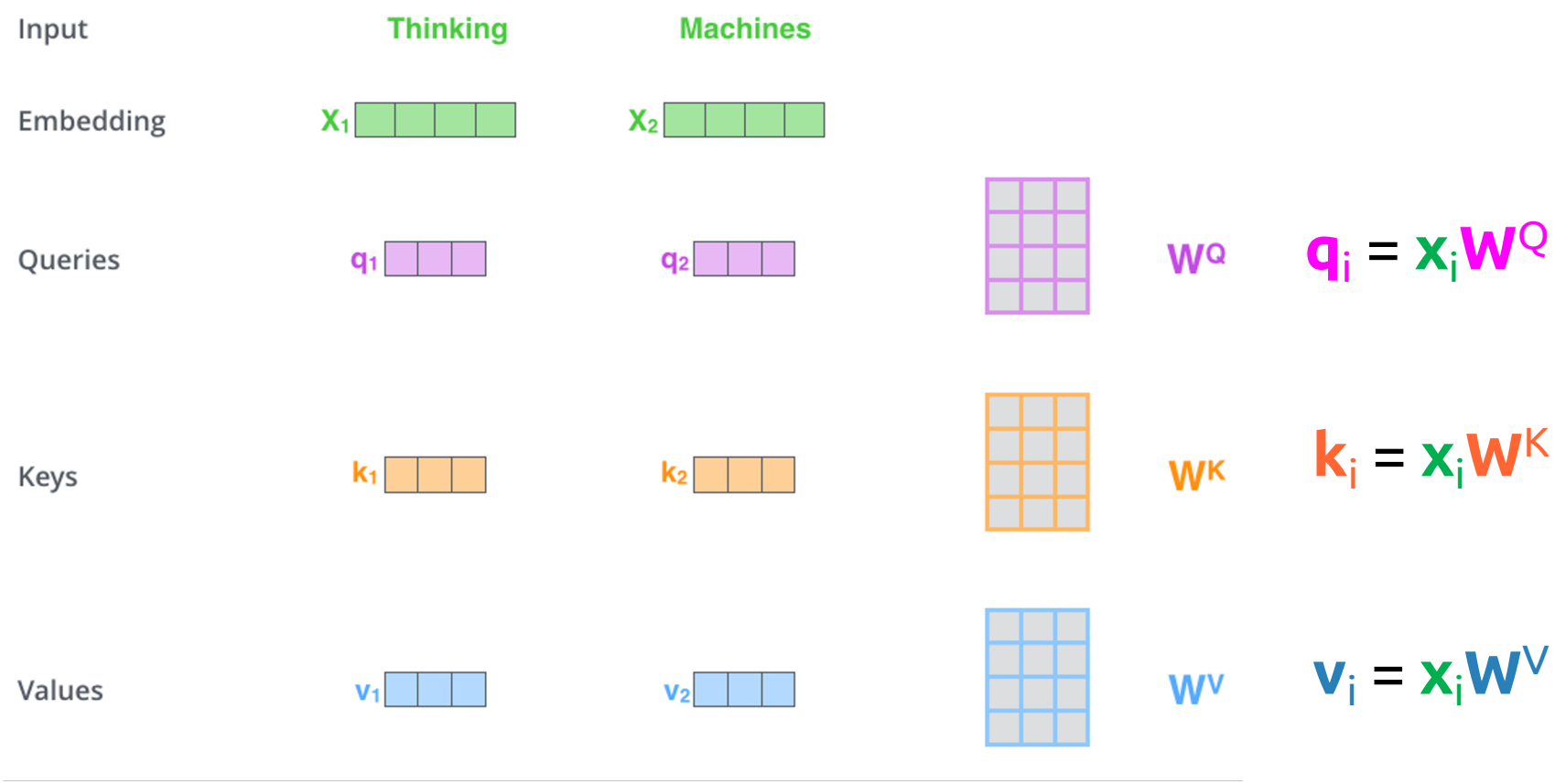
$\mathbf{k}_i = \mathbf{x}_i \mathbf{W}^K$

Values:

$\mathbf{v}_i = \mathbf{x}_i \mathbf{W}^V$

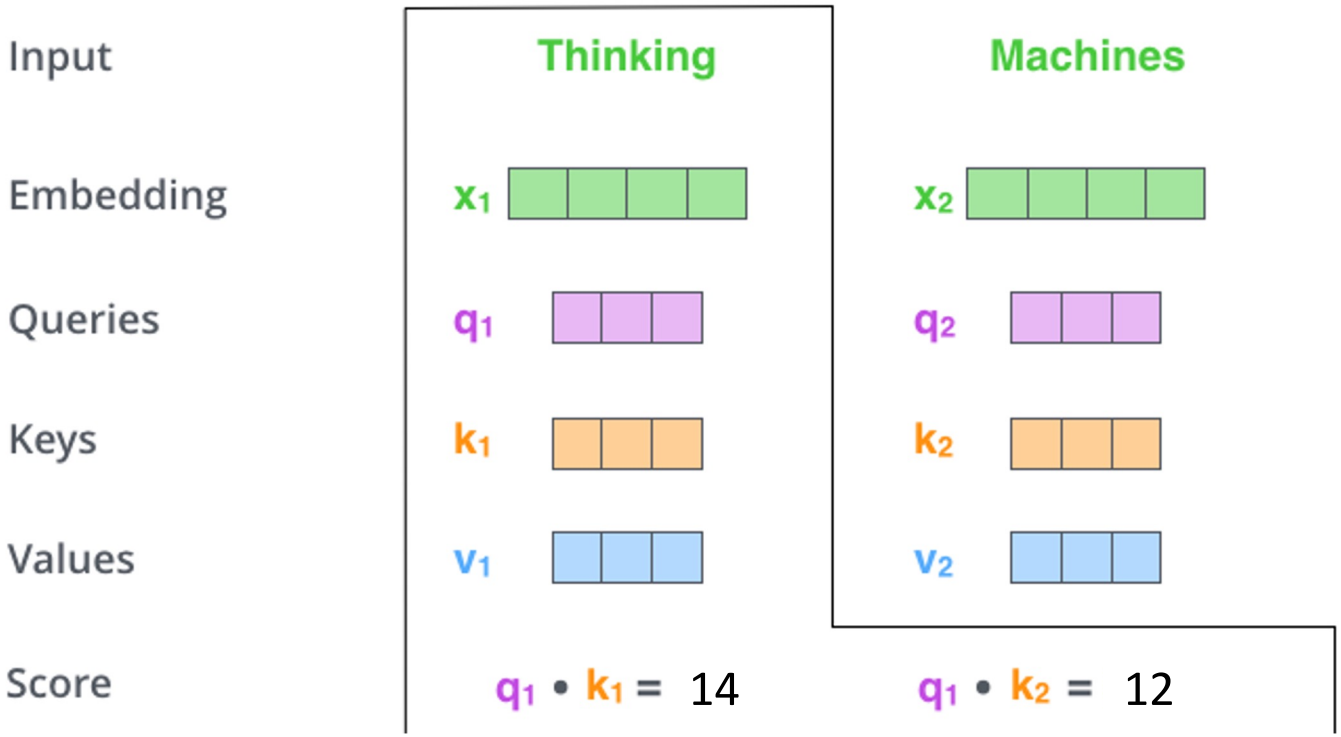
$\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$ are **projection layers shared** across all words

Self-attention (3)



Jay Alammar, The Illustrated Transformer, 2018.

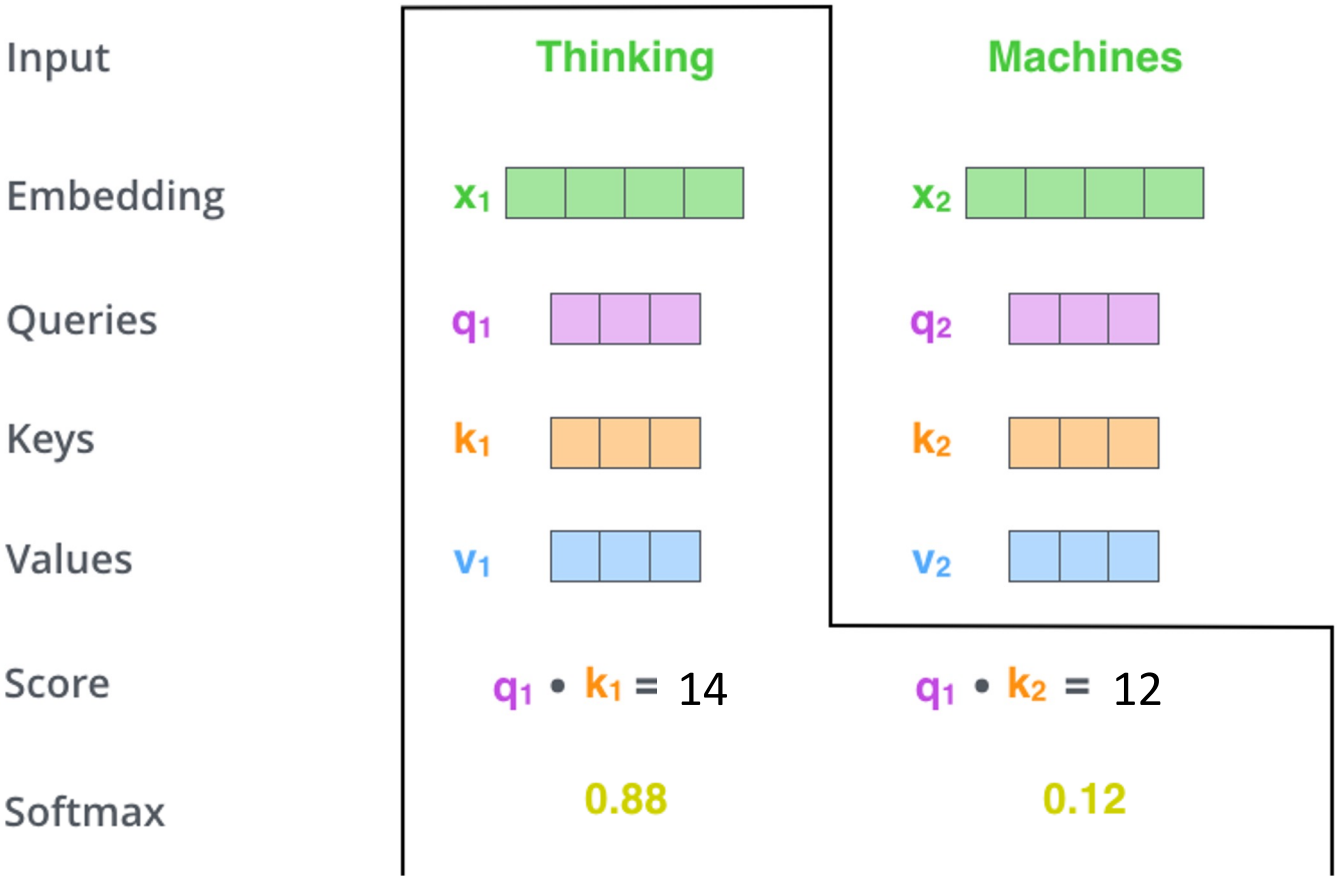
Self-attention (4)



$$e_{ij} = q_i \cdot k_j$$

Jay Alammar, The Illustrated Transformer, 2018.

Self-attention (5)

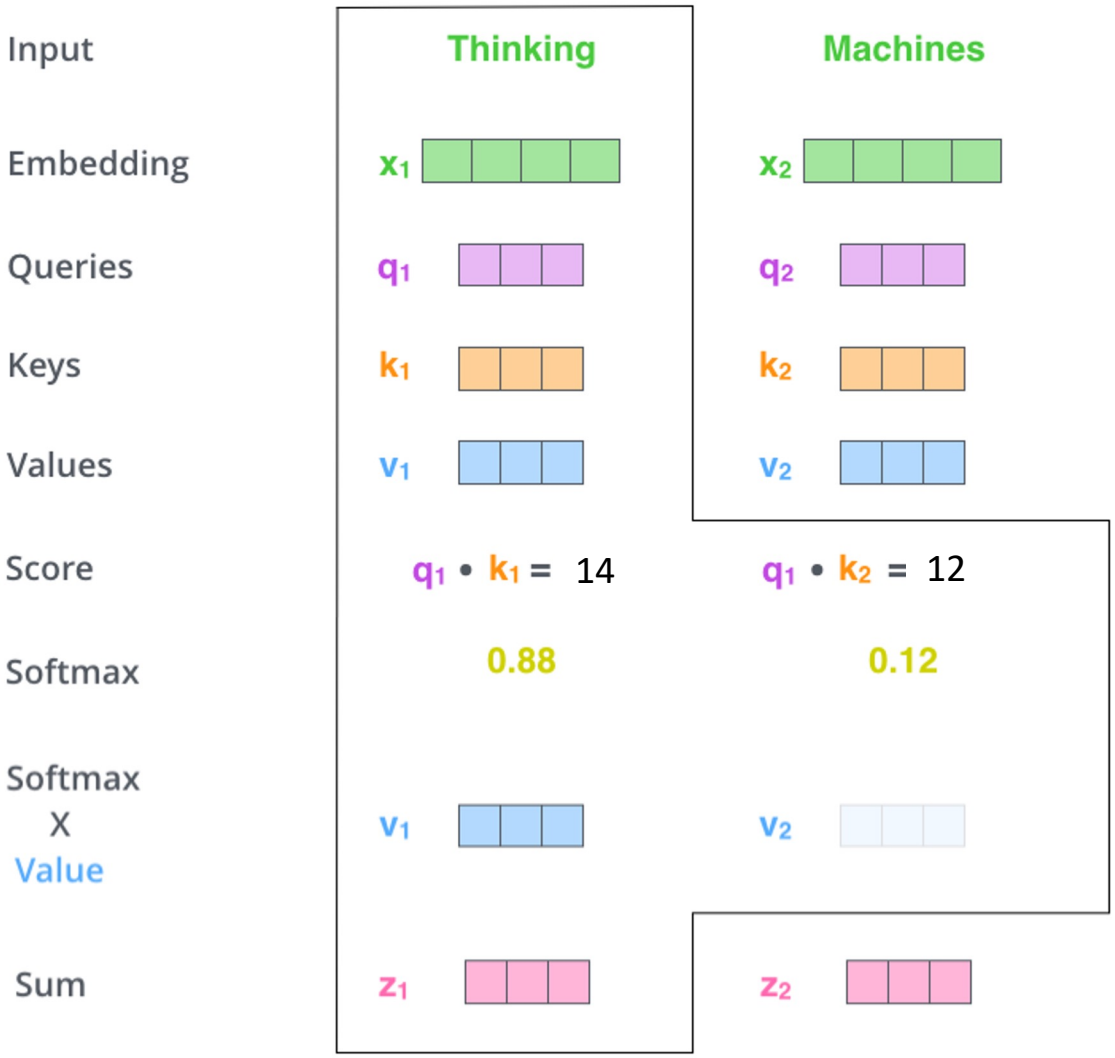


$$e_{ij} = q_i \cdot k_j$$

$$a_{ij} = \text{softmax}(e_{ij})$$

Jay Alammar, The Illustrated Transformer, 2018.

Self-attention (6)



$$e_{ij} = q_i \cdot k_j$$

$$a_{ij} = \text{softmax}(e_{ij})$$

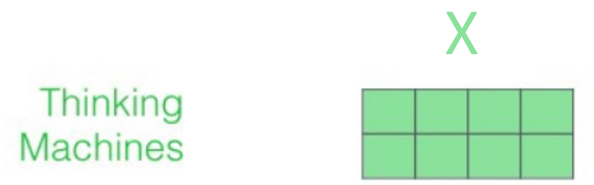
$$z_i = \sum_j a_{ij} v_j$$

Jay Alammar, The Illustrated Transformer, 2018.

Self-attention: vectorized steps (1)

1) This is our input sentence

2) We embed each word



Jay Alammar, The Illustrated Transformer, 2018.

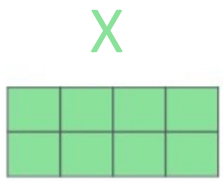
Self-attention: vectorized steps (2)

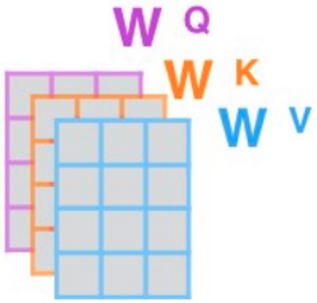
1) This is our input sentence

2) We embed each word

3) We multiply X with weight matrices

Thinking
Machines





Jay Alammar, The Illustrated Transformer, 2018.

Self-attention: vectorized steps (3)

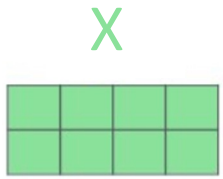
1) This is our input sentence

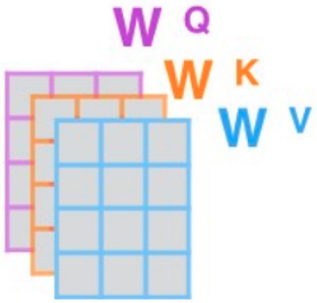
2) We embed each word

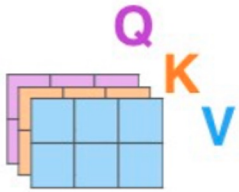
3) We multiply X with weight matrices

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

Thinking
Machines







Jay Alammar, The Illustrated Transformer, 2018.

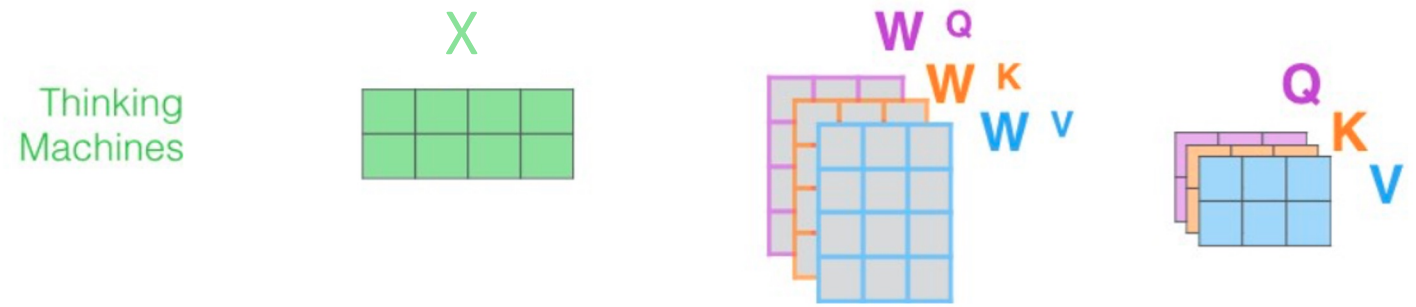
Self-attention: vectorized steps (4)

1) This is our input sentence

2) We embed each word

3) We multiply X with weight matrices

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

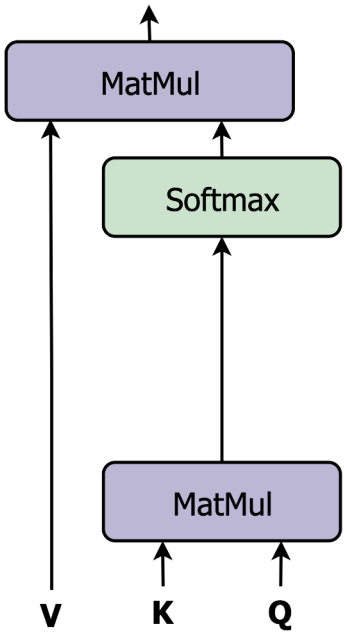


5) Calculate outputs (context)

Jay Alammar, The Illustrated Transformer, 2018.

Self-attention in (vectorized) equations

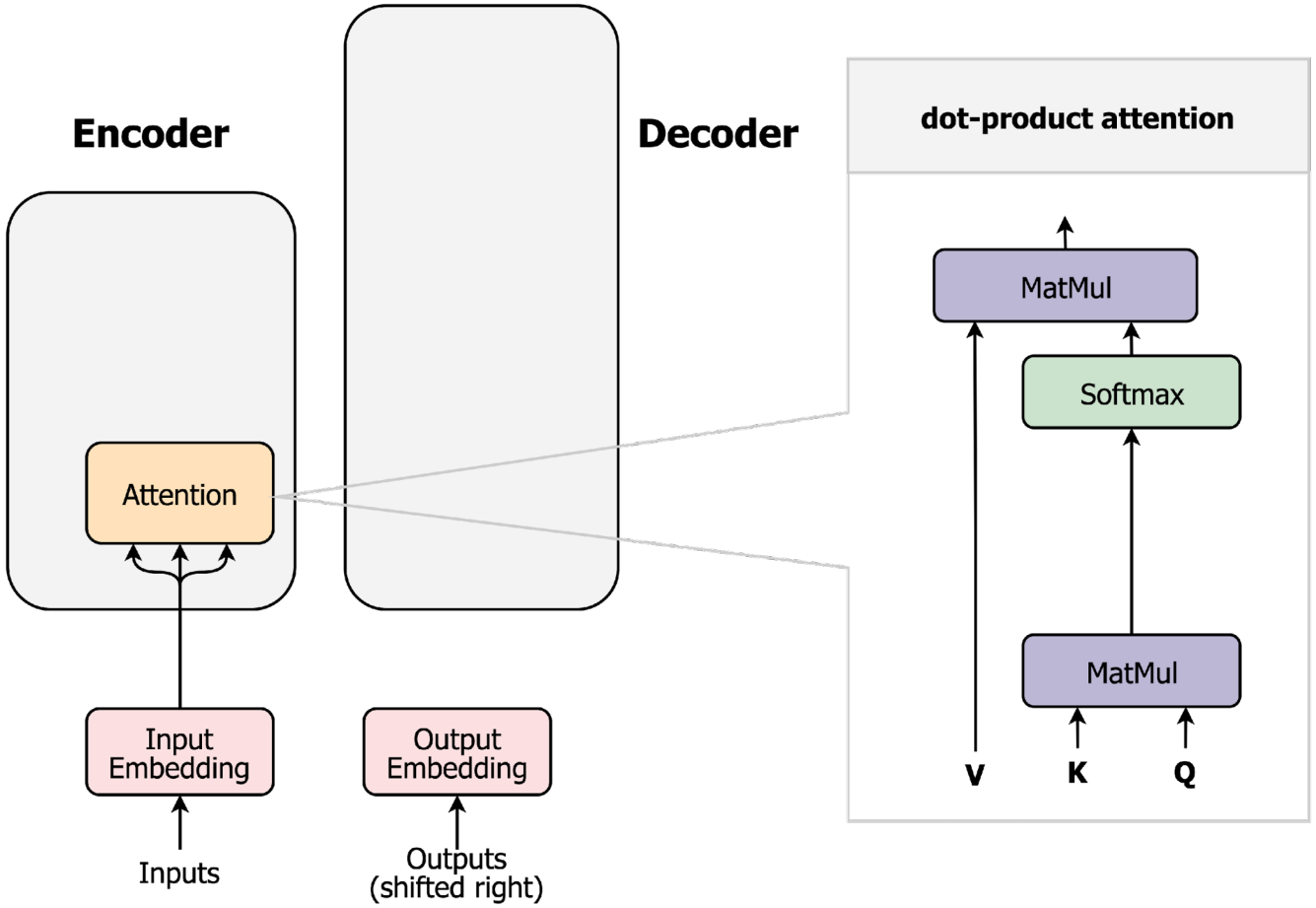
$$\boldsymbol{Z} = \text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T\right)\boldsymbol{V} = \text{softmax}\left(\boldsymbol{X}\boldsymbol{W}^Q\boldsymbol{W}^{KT}\boldsymbol{X}^T\right)\boldsymbol{X}\boldsymbol{W}^V$$



$$\boldsymbol{W}^Q \in \mathbb{R}^{d_x \times d_q}$$
$$\boldsymbol{W}^K \in \mathbb{R}^{d_x \times d_k} \quad d_q = d_k$$
$$\boldsymbol{W}^V \in \mathbb{R}^{d_x \times d_v}$$

Jay Alammar, The Illustrated Transformer, 2018.

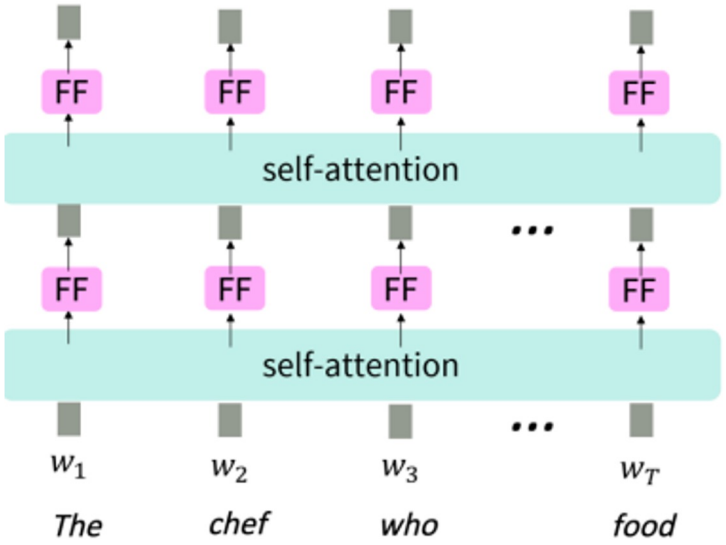
Self-attention in transformer (1)



Self-attention in transformer (2)

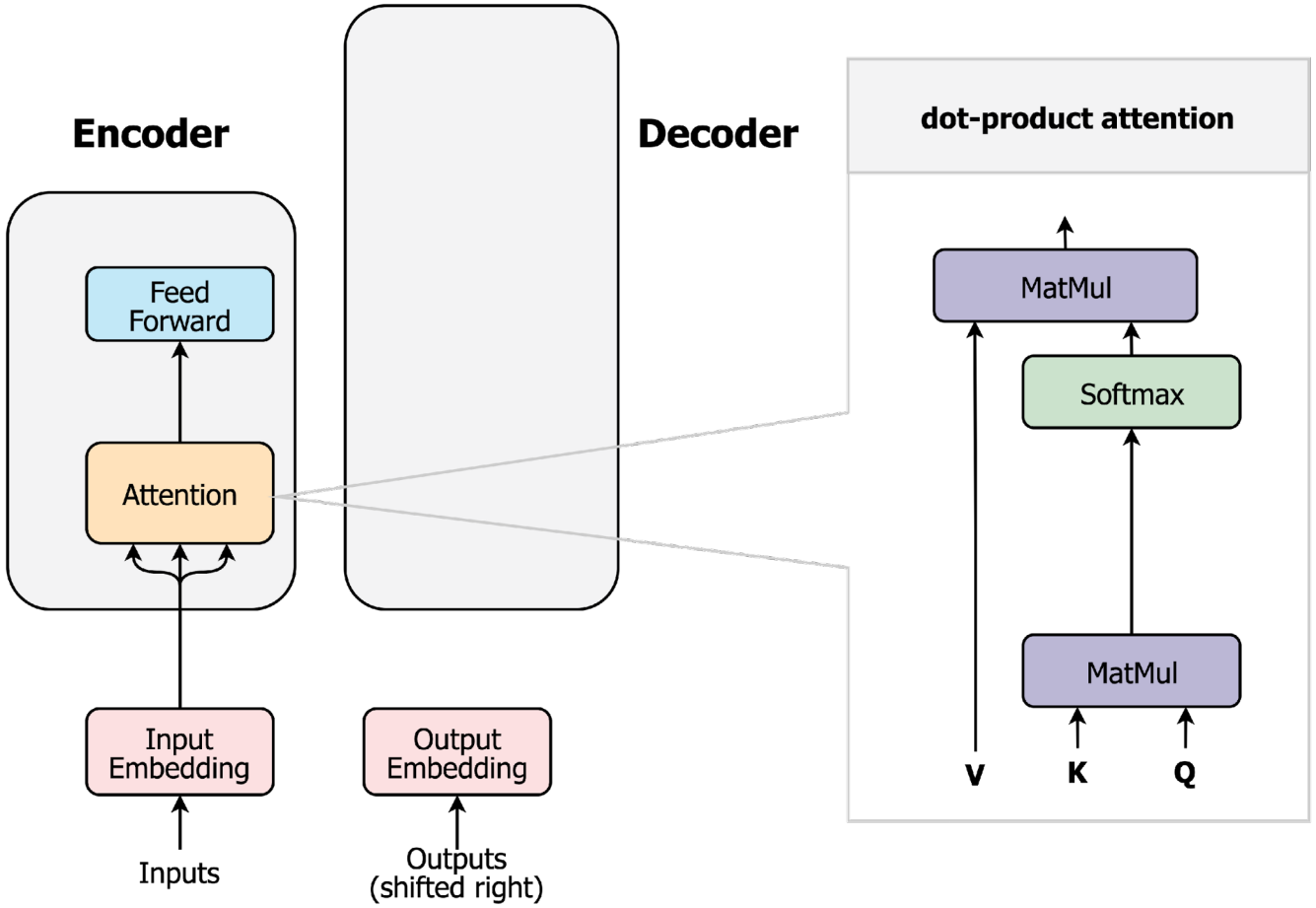
- **Problem**: Since there are no element-wise non-linearities, self-attention is simply performing a re-averaging of the value vector
- **Solution**: Apply a feed forward layer to each output of attention (each row of the matrix Z), providing non-linear activation and additional expressive power

$$\boldsymbol{c}_i = \mathrm{MLP}\left(\boldsymbol{z}_i\right) = \boldsymbol{W}_2 \mathrm{f}\left(\boldsymbol{W}_1 \boldsymbol{z}_i + \boldsymbol{b}_1\right) + \boldsymbol{b}_2$$



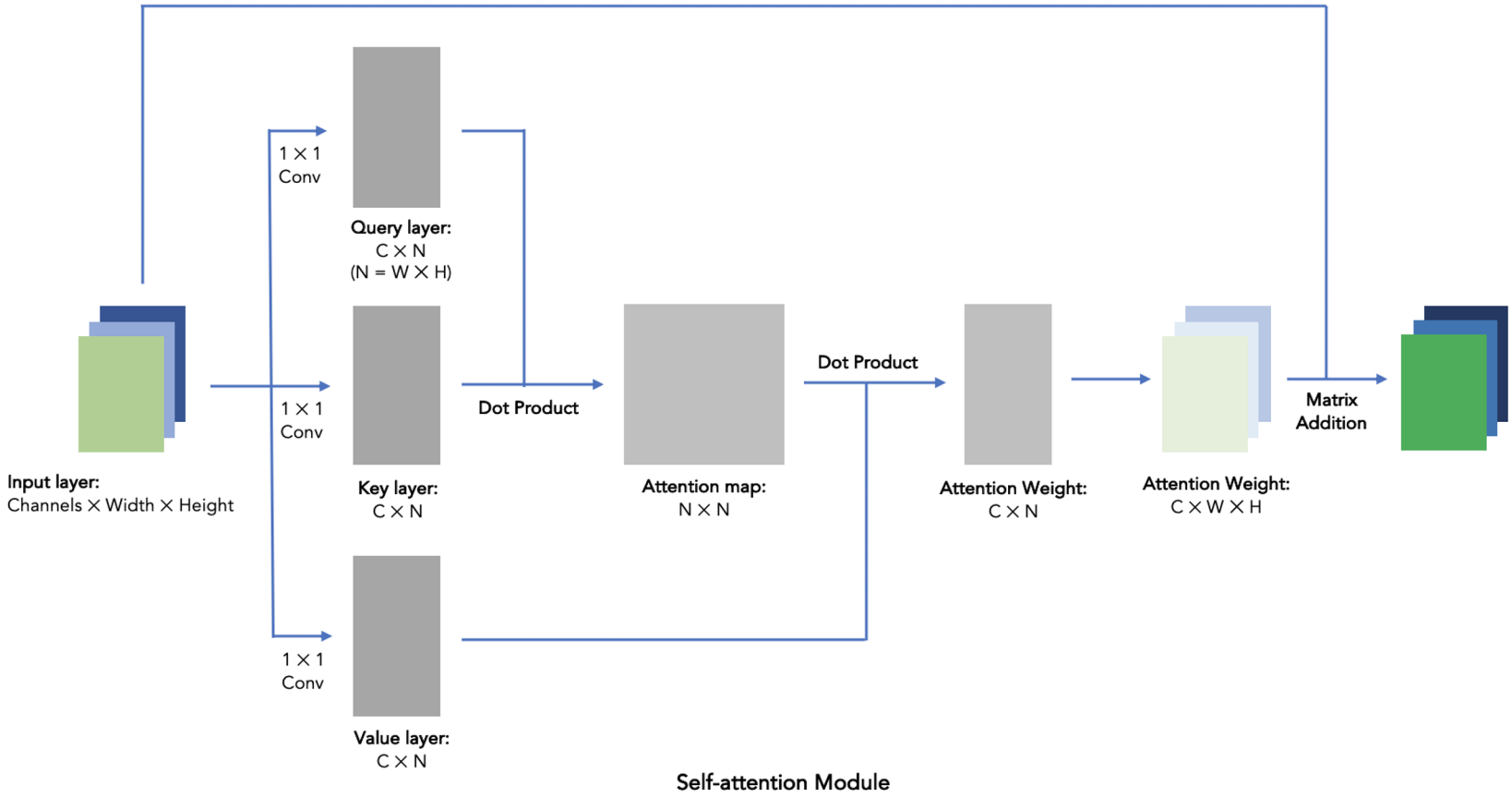
Anna Goldie, John Hewitt, Standford cs224n course, 2022.

Self-attention in transformer (3)



Study case: Self-attention in image generation (1)

- Details can be generated using cues from all feature locations.
- Can check consistency between features in distant portions of the image

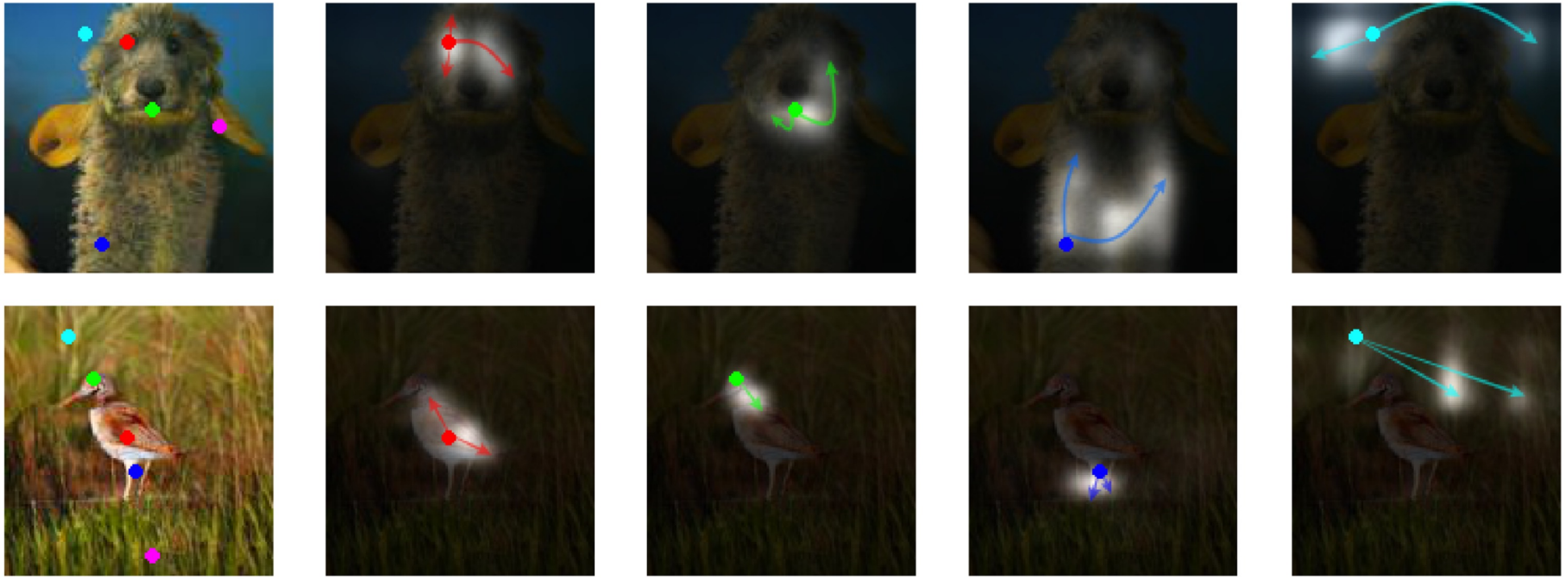


Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." ICML 2019. [video]

Study case: Self-attention in image generation (2)

Query locations

Attention maps for different query locations



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

Outline

- Introduction
- Self-attention
- **Multi-head attention**
- Deep layers
- Positional encoding
- The Transformer

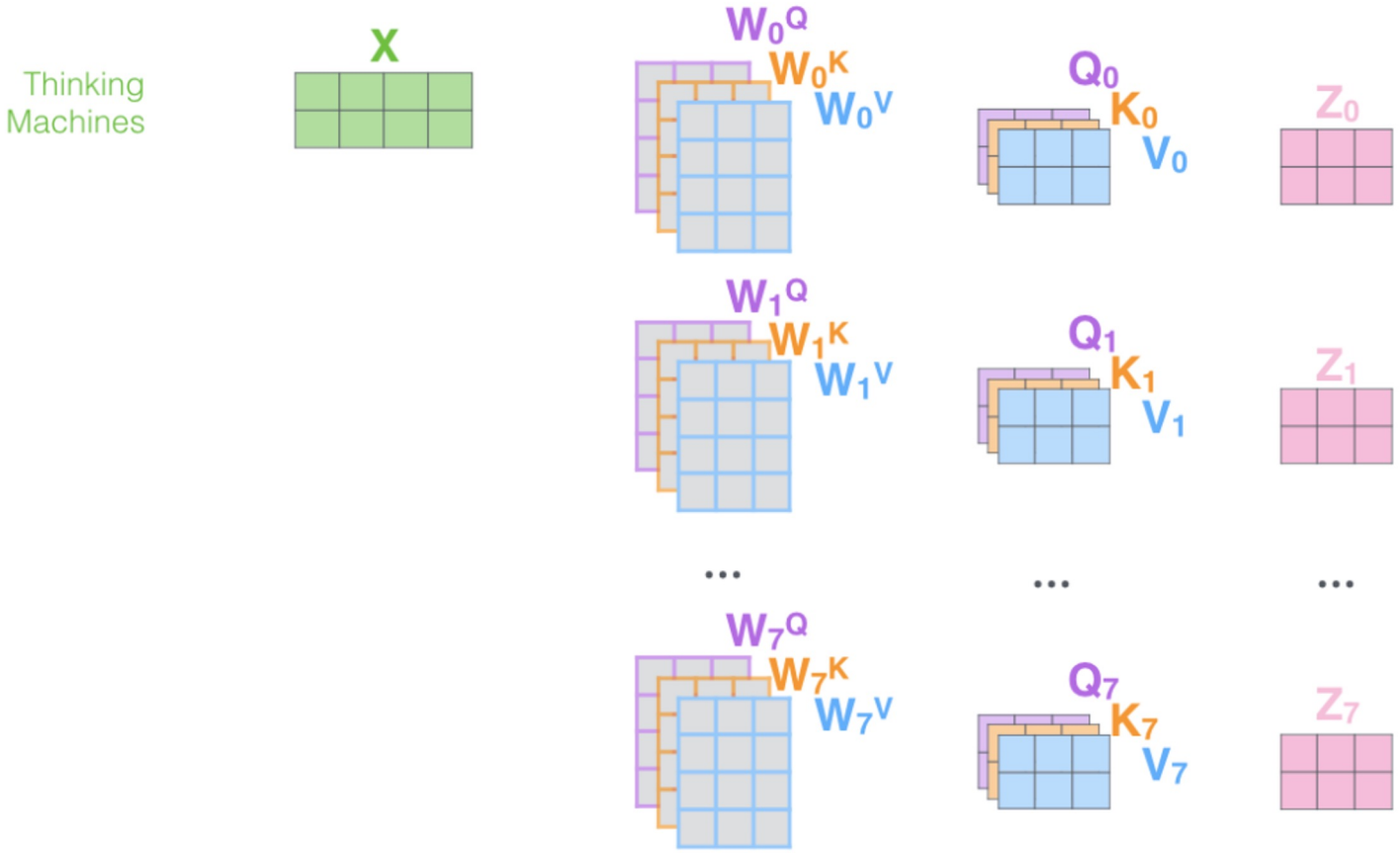
Multi-head attention (1)

- **H** heads is better than 1!!
- **Main idea**: perform self-attention multiple times in parallel and combine the results

Wizards of the Coast, Artist: Todd Lockwood

Multi-head attention (2)

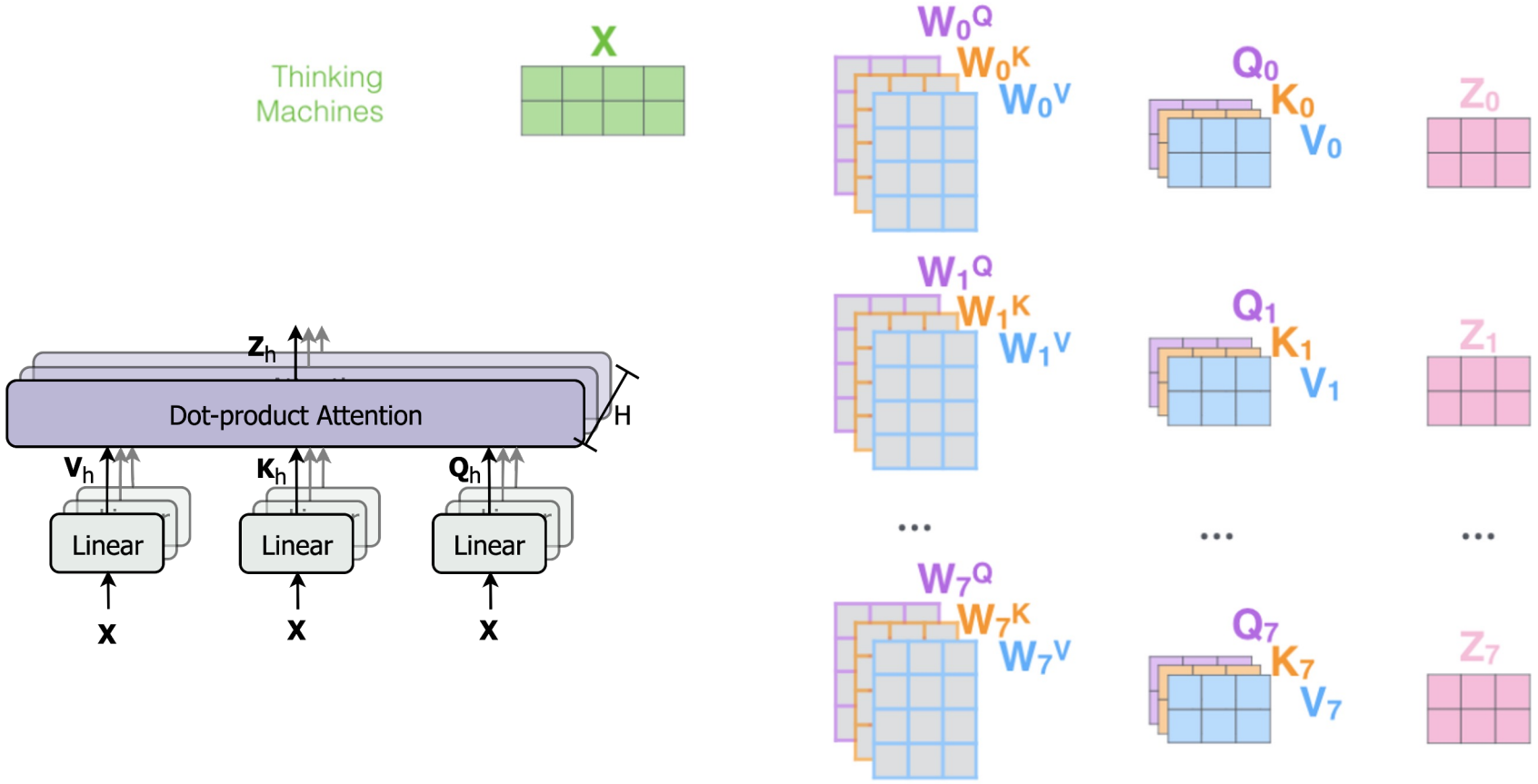
Multiple sets of projection matrices to provide **different contextual representations** for the same input token



Jay Alammar, The Illustrated Transformer, 2018.

Multi-head attention (3)

Multiple sets of projection matrices to provide **different contextual representations** for the same input token



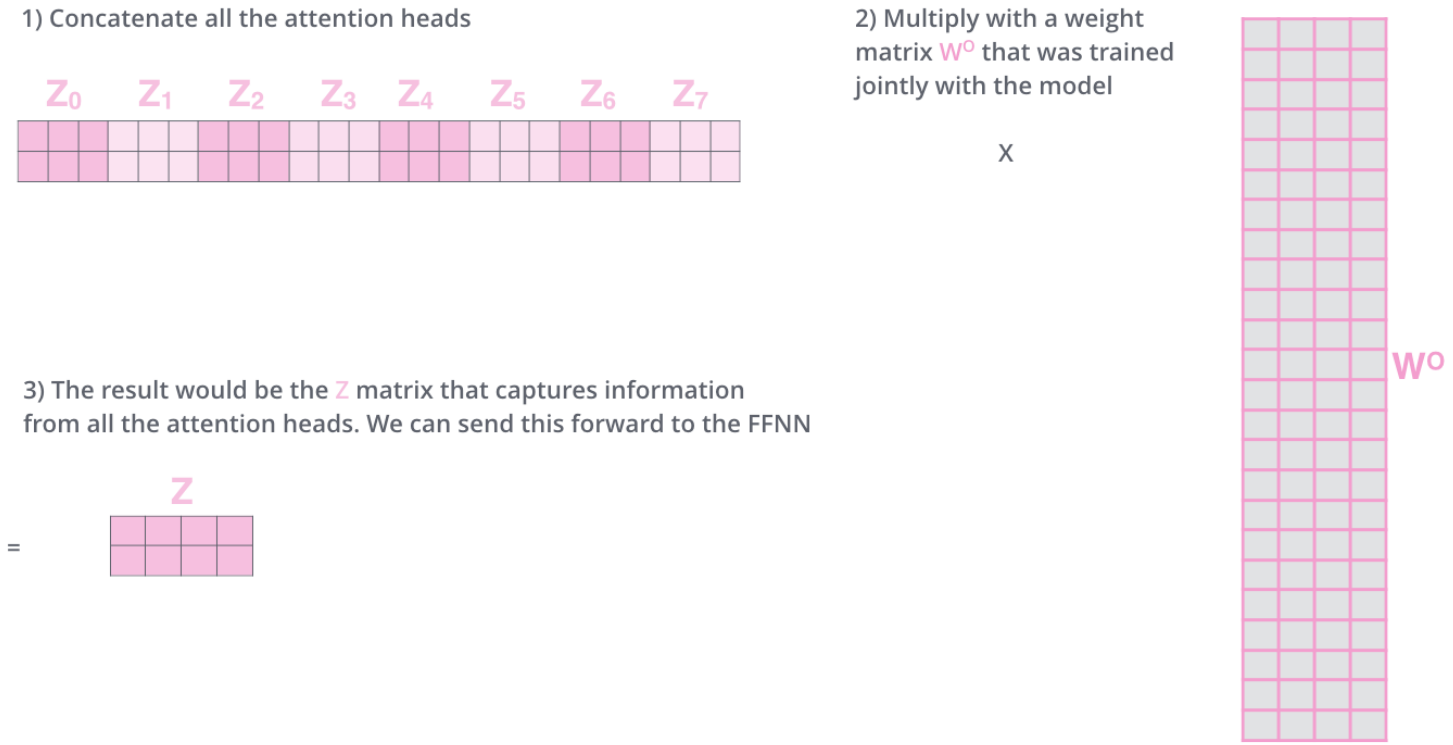
$$Z_h = \text{softmax}\left(XW_h^Q W_h^{K\,T} X^T\right) XW_h^V$$

Jay Alammar, The Illustrated Transformer, 2018.

Multi-head attention (5)

How to combine all H heads?

- All Z_h matrices are concatenated and combined in a new Z



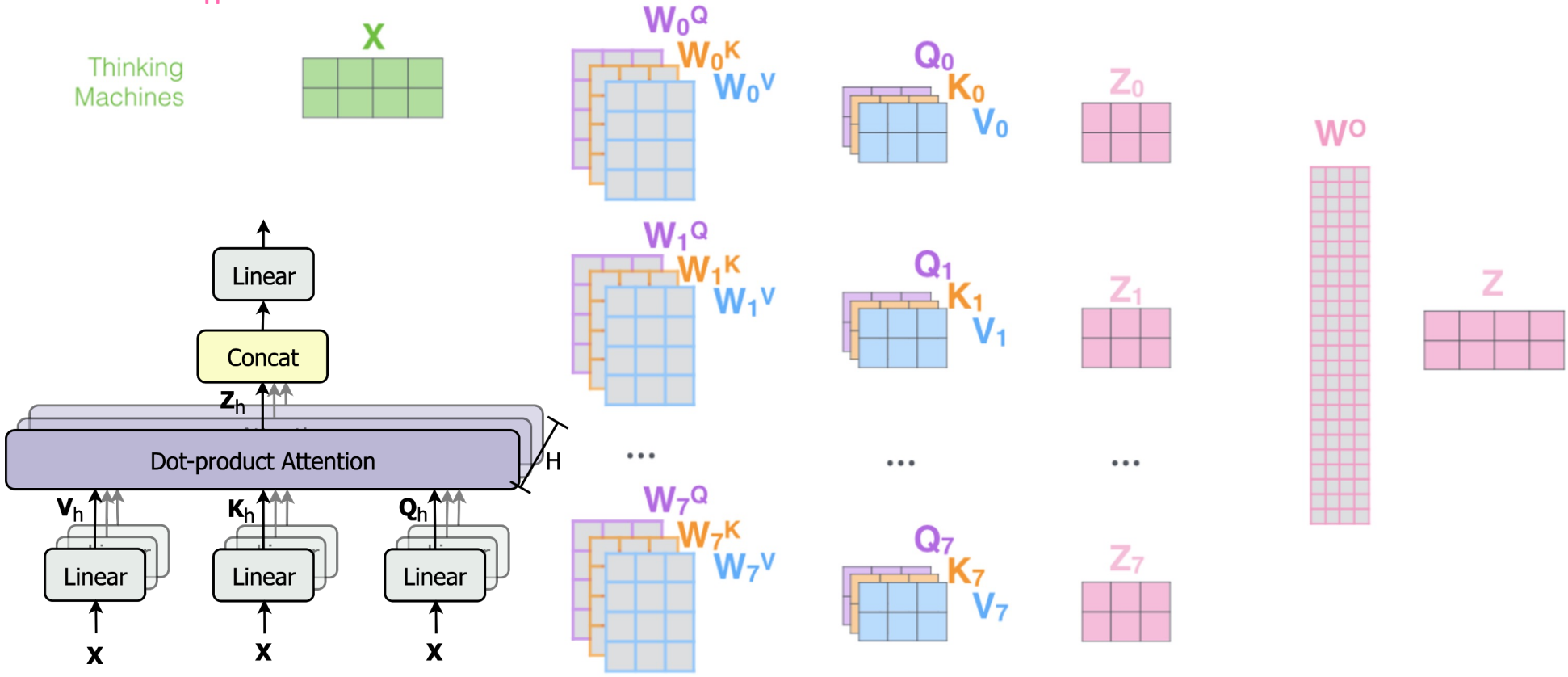
$$Z = [Z_0, Z_1, \ldots, Z_H]\, W^0$$

Jay Alammar, The Illustrated Transformer, 2018.

Multi-head attention (5)

How to combine all H heads?

- All Z_h matrices are concatenated and combined in a new Z



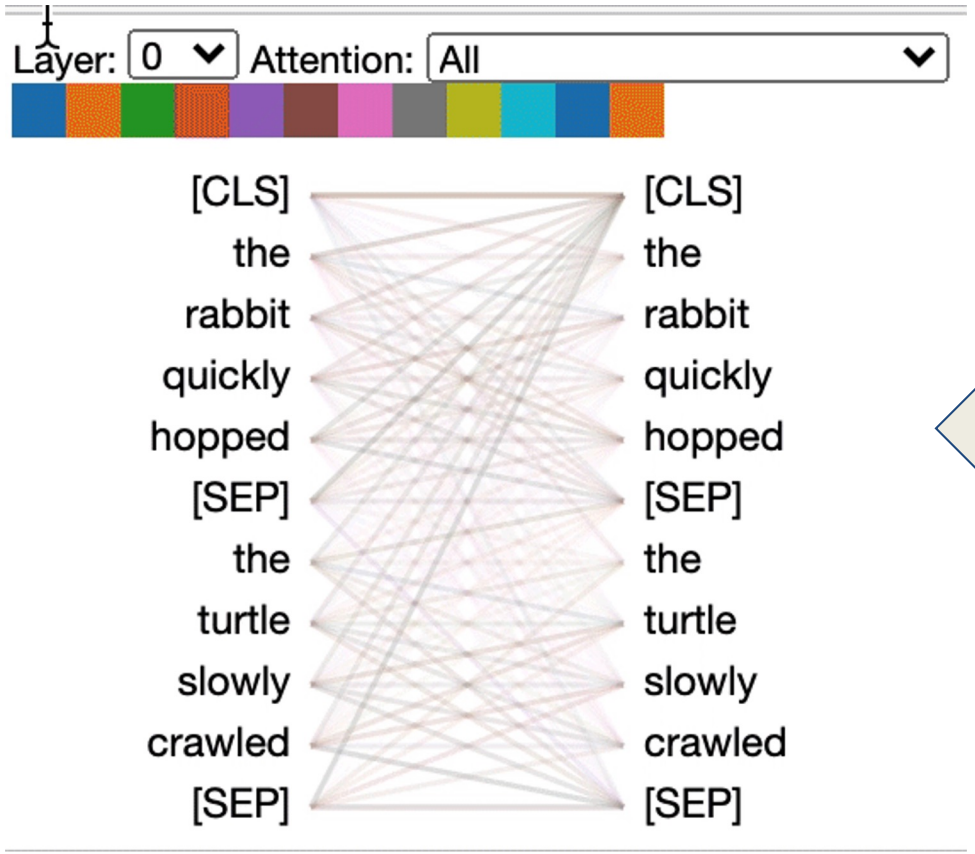
$$Z = [Z_0, Z_1, \ldots, Z_H]\, W^0$$

Jay Alammar, The Illustrated Transformer, 2018.

Multi-head attention (6)

Visualization

Each colour corresponds to a head.

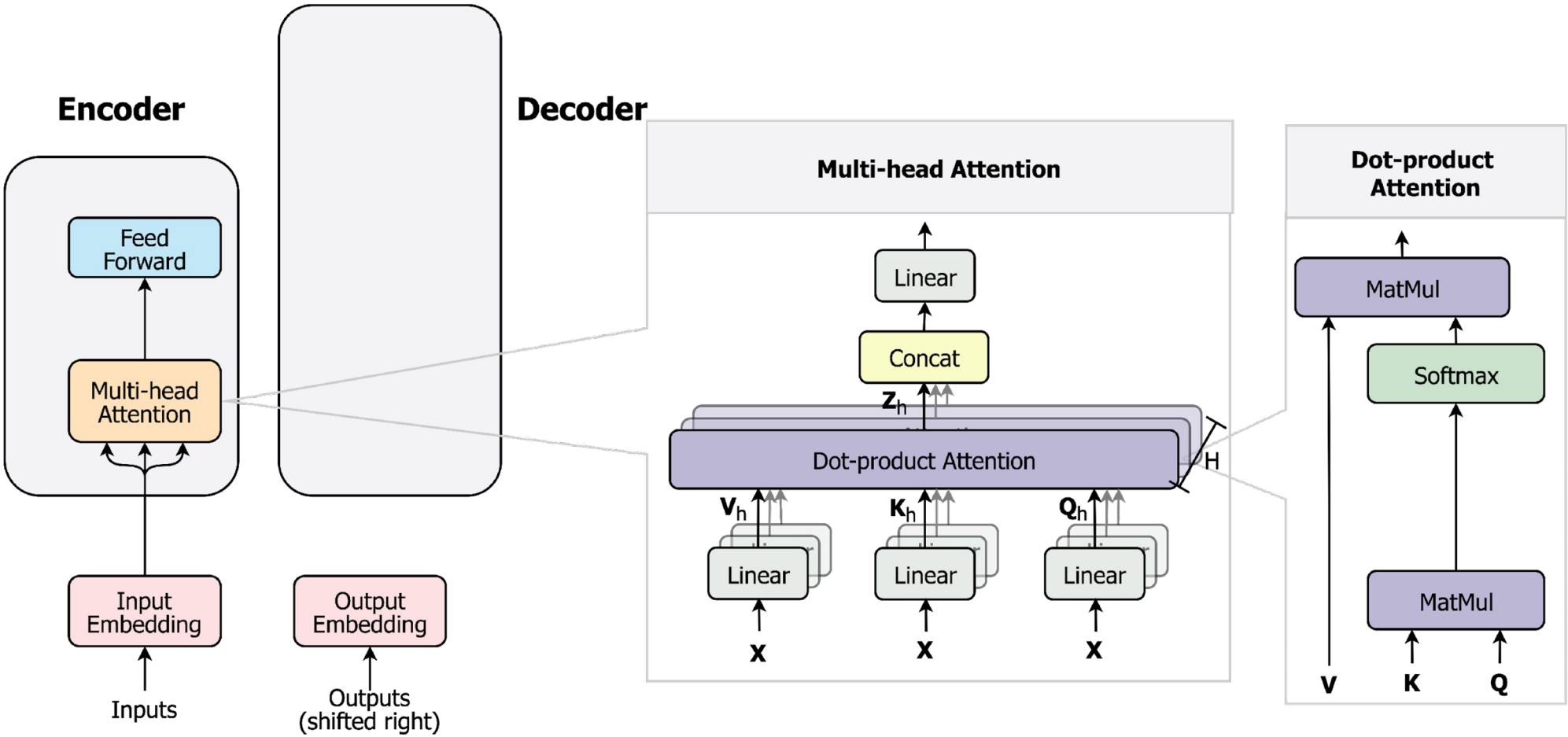


Blue: First head only.

Multi-color: Multiple heads.

Vig, Jesse. "A multiscale visualization of attention in the transformer model." ACL 2019. [code] [tweet]

Multi-head attention in transformer



Outline

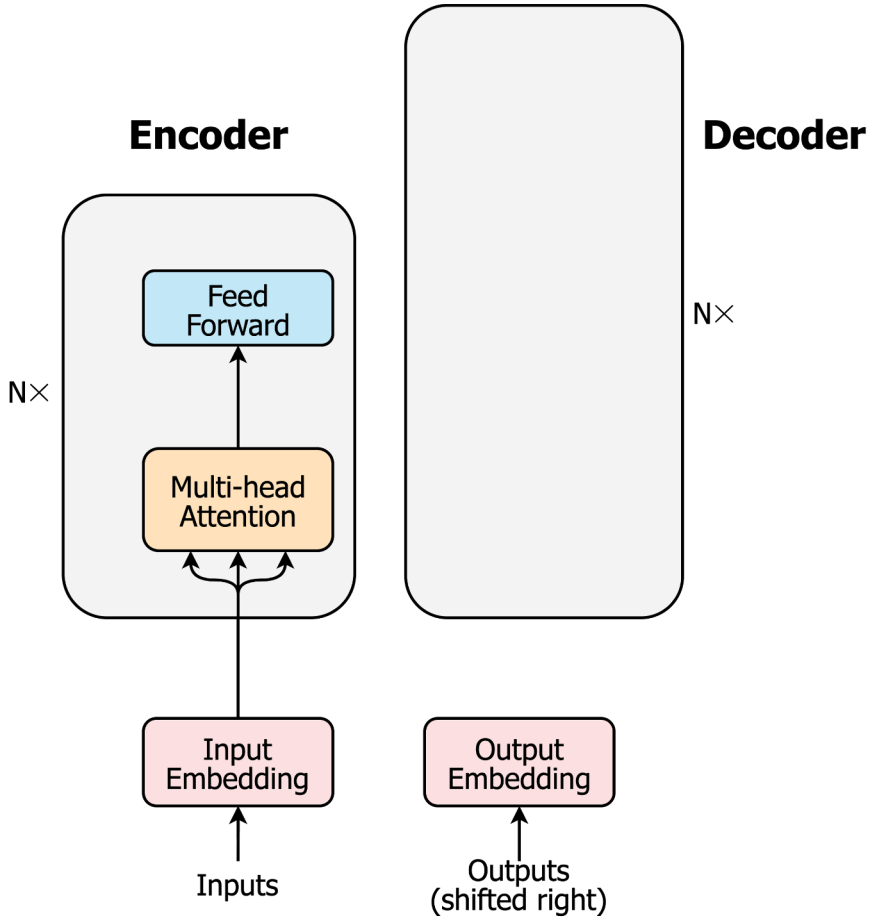
- Introduction
- Self-attention
- Multi-head attention
- **Deep layers**
- Positional encoding
- The Transformer

Let's go deep (1)

Add **N layers** (N=6 in the original paper) of multi-head attentions and feed-forward networks.

Repeat this structure N times:

- First layer's input are word embeddings
- Further layers' input is the context vector output of the previous layer

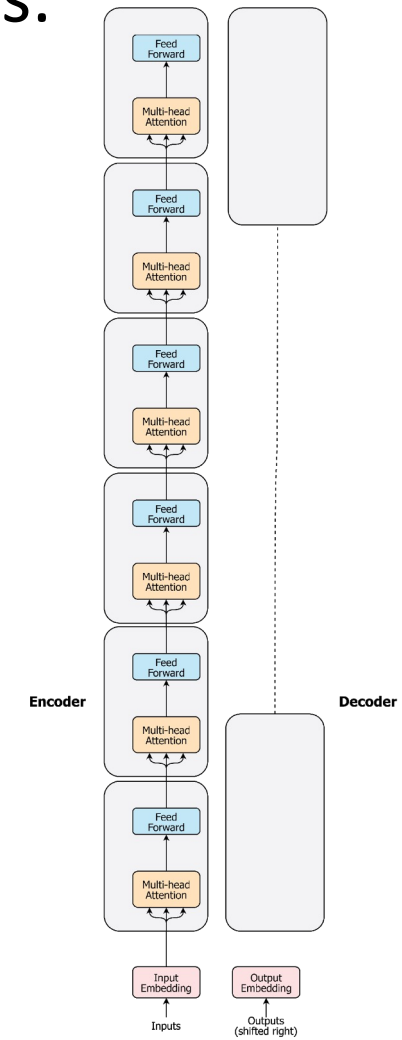


Let’s go deep (2)

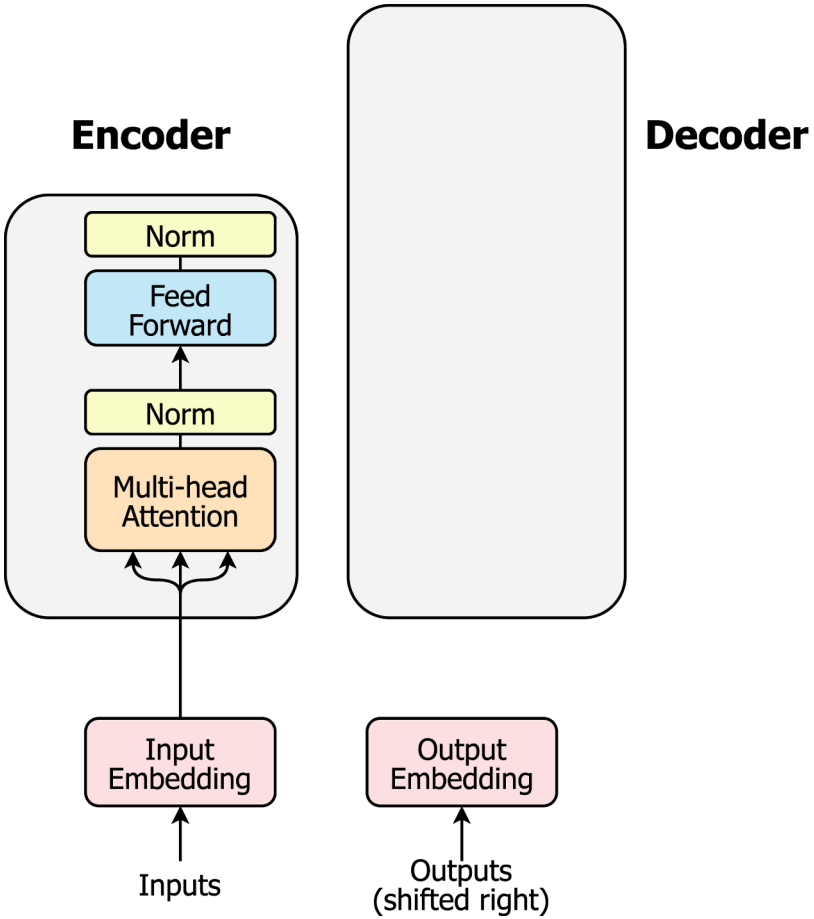
Add N layers (N=6 in the original paper) of multi-head attentions and feed-forward networks.

Repeat this structure N times:

- First layer’s input are word embeddings
- Further layers’ input is the context vector output of the previous layer



Layer normalization



- **Problem**: Difficult to train the parameters of a given layer because its input from the layer beneath keeps shifting
- **Solution**: Reduce uninformative variation by normalizing to zero mean and standard deviation of one within each layer [Ba et al., 2016]

Mean: $\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l$ Standard Deviation: $\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2}$

$$x^{\ell\prime} = \frac{x^\ell - \mu^\ell}{\sigma^\ell + \epsilon}$$

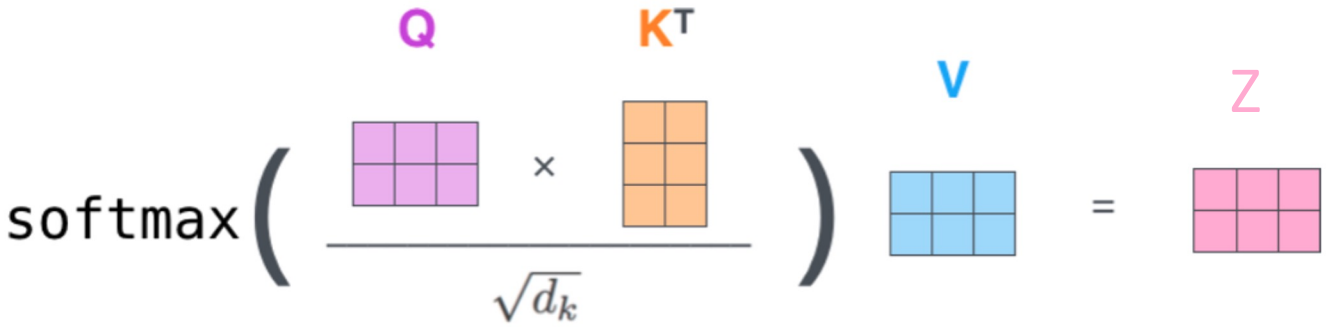
Scaled dot-product attention (1)

- After Layer **Normalization**, the mean and variance of vector elements is 0 and 1, respectively
- However, the dot product still tends to take on extreme values, as its variance scales with dimensionality d_k

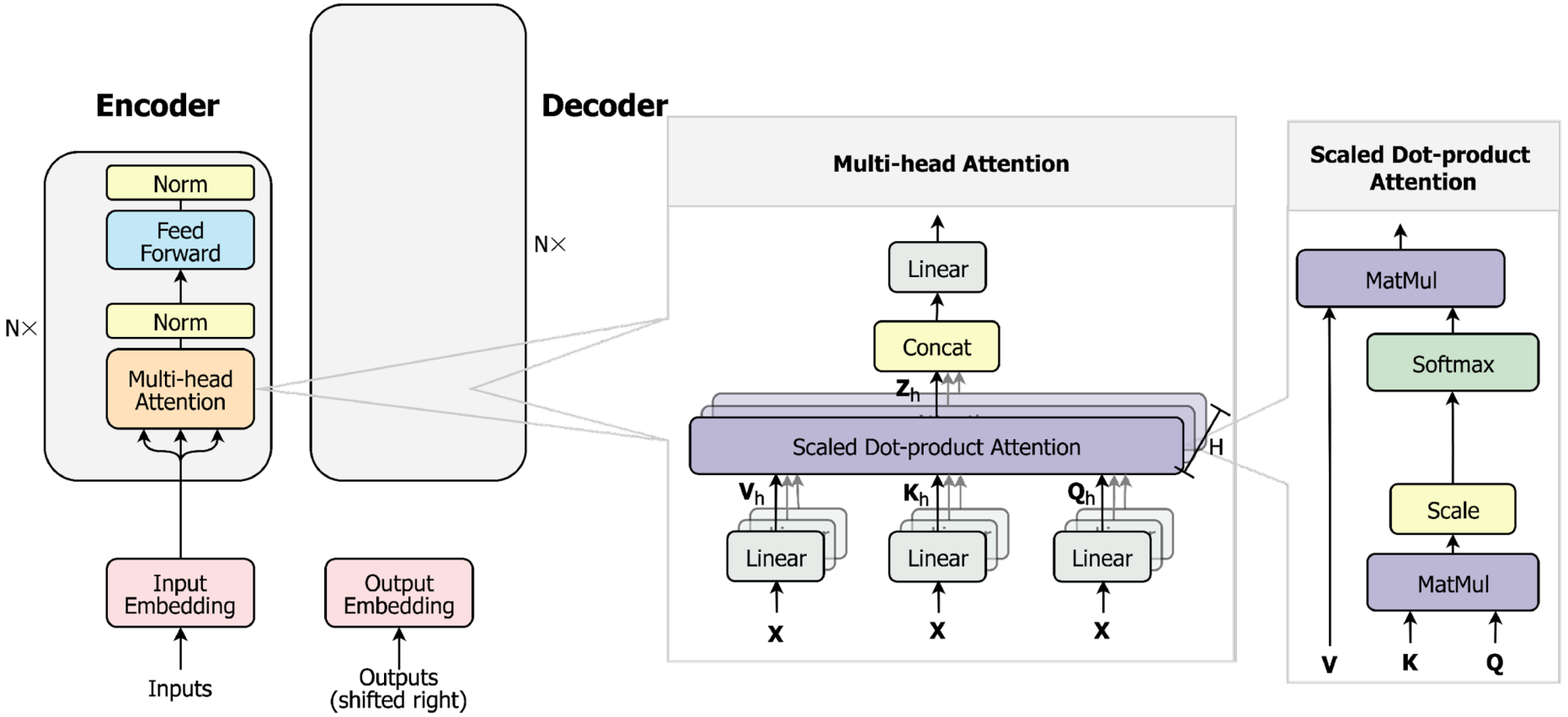
Quick Statistics Review:

- Mean of sum = sum of means = $d_k * 0 = 0$
- Variance of sum = sum of variances = $d_k * 1 = d_k$
- To set the variance to 1, simply divide by $\sqrt{d_k}$!

5) Updated self-attention equation (context)



Scaled dot-product attention (2)



$$Z = \mathrm{MHSA}(X) = [Z_0, \ldots, Z_H]\, W^0$$

$$Z_h = \mathrm{softmax}\left(\frac{Q_h {K_h}^T}{\sqrt{d_k}}\right) V_h = \mathrm{softmax}\left(\frac{X W_h^Q W_h^{K\,T} X^T}{\sqrt{d_k}}\right) X W_h^V$$

Residual connections (1)

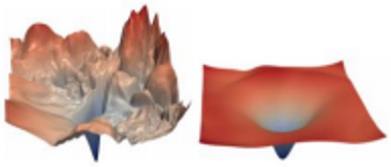
Residual connections are a simple but powerful technique [He at al., 2016]

- Deep networks are surprisingly bad at learning the identity function
- Directly passing "raw" embeddings to the next layer can actually be very helpful

$$x_l = F(x_{l-1}) + x_{l-1}$$

- This prevents the network from "forgetting" or distorting important information as it is processed by many layers.

Residual connections are also thought to smooth the loss landscape and make training easier!



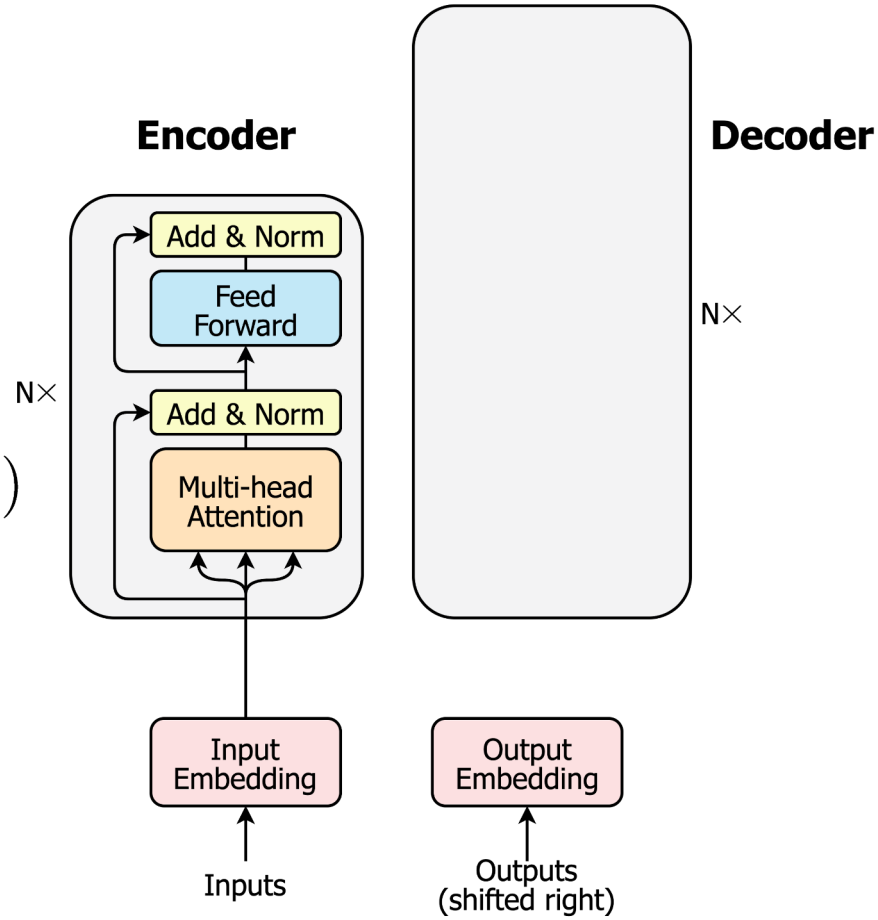
[no residuals] [residuals]

[Loss landscape visualization, Li et al., 2018, on a ResNet]

Residual connections (2)

$$C = \mathrm{MLP}(Z) + Z$$

$$Z = \mathrm{LayerNorm}(\mathrm{MHSA}(X) + X)$$

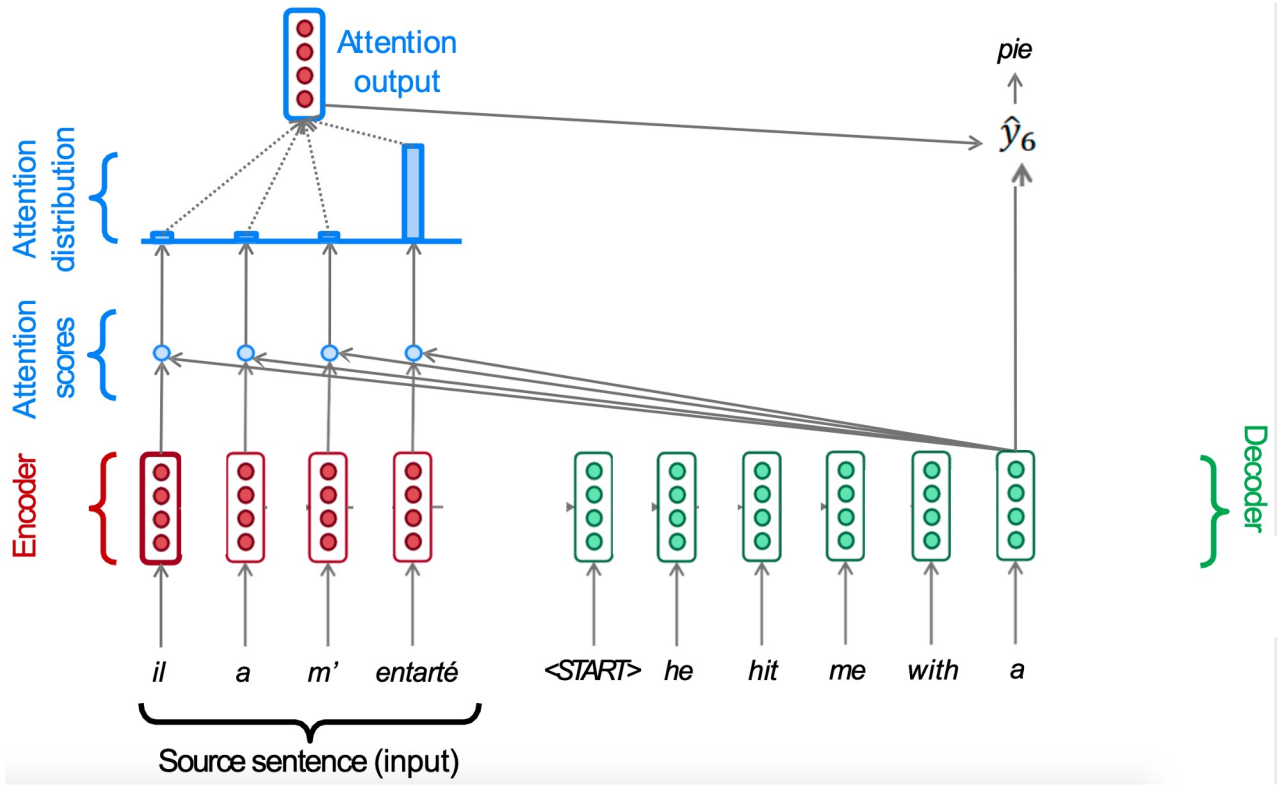


Outline

- Introduction
- Self-attention
- Multi-head attention
- Deep layers
- **Positional encoding**
- The Transformer

Positional encoding (1)

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



Positional encoding (2)

- Consider the following two sequences:

elephants hate mice

x_1 x_2 x_3

Mice hate elephants

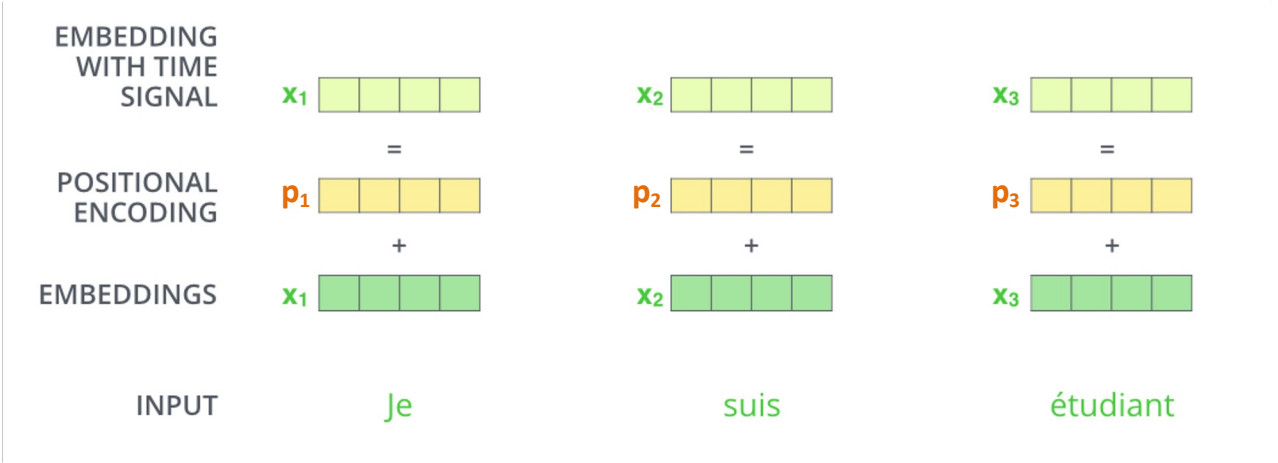
x_1 x_2 x_3

Are self-attention scores different in both cases?

Positional encoding (3)

- Since self-attention does not build in order information, we need to encode the order in the keys, queries and values
- Consider representing each **sequence index** as a **vector** (positional encoding): $\boldsymbol{p_i} \in \mathbb{R}^{d_x} \quad \text{for} \quad i \in \{1, 2, \ldots, T\}$
- **Add** the positional encoding to the word embedding (concatenating also a possibility)

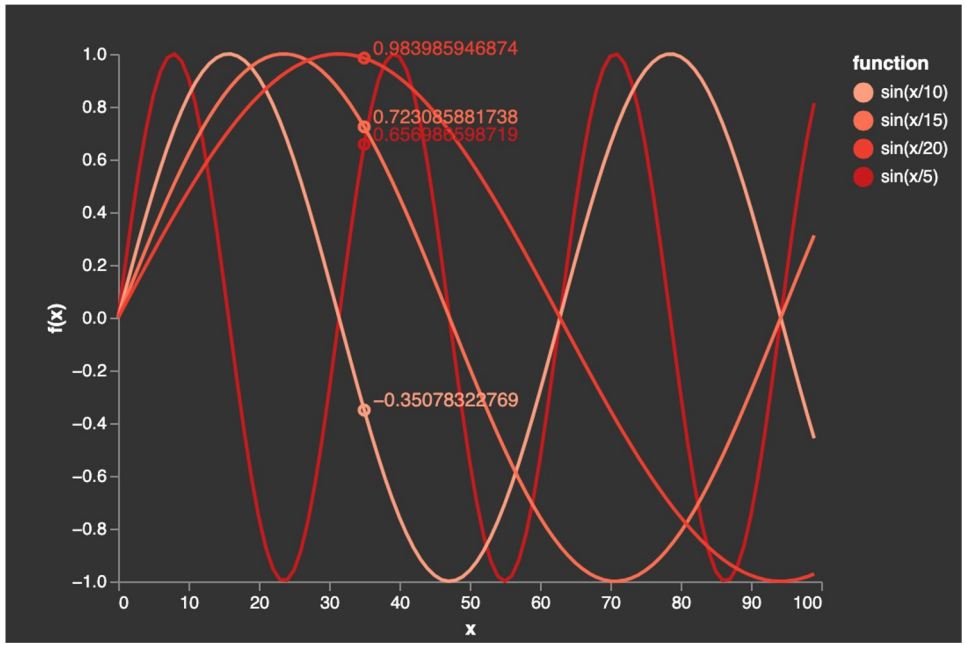
$$\boldsymbol{x_i} = \tilde{\boldsymbol{x}}_i + \boldsymbol{p_i}$$



Positional encoding (4)

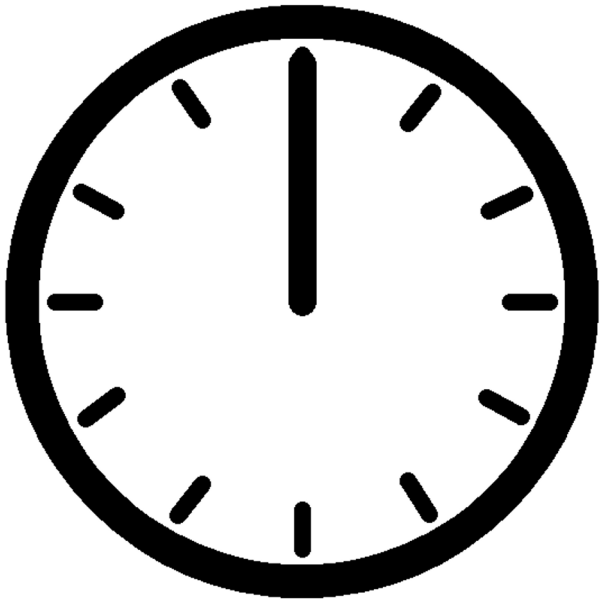
- **Sinusoidal position representations**: concatenate sinusoidal functions of varying periods:

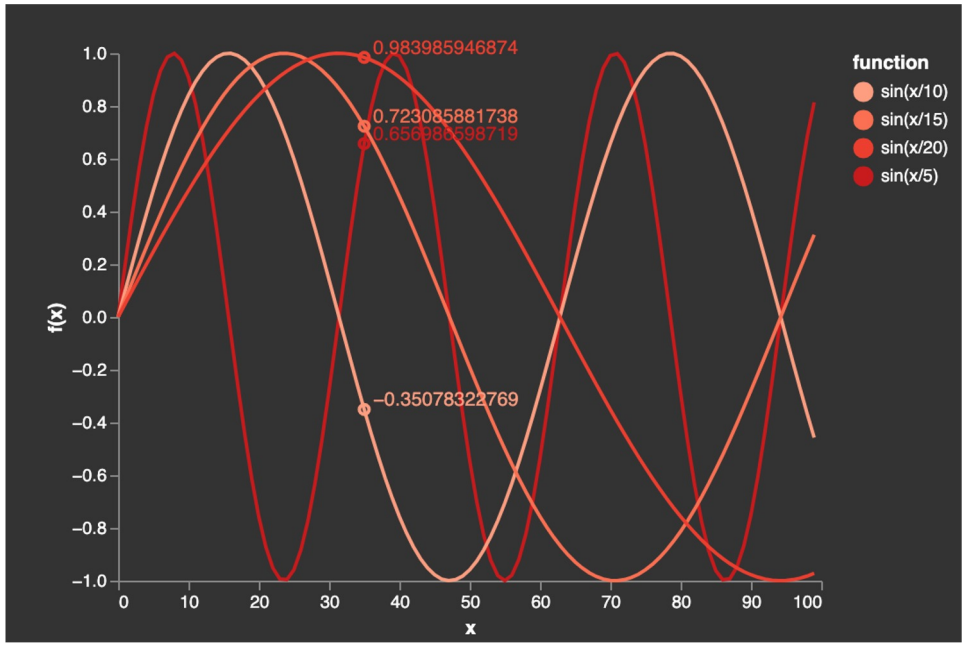
$$\boldsymbol{p_i} = \begin{pmatrix} \sin\left(\frac{i}{10000^{2\frac{1}{d_x}}}\right) \\ \cos\left(\frac{i}{10000^{2\frac{1}{d_x}}}\right) \\ \vdots \\ \sin\left(\frac{i}{10000^{2\frac{d_x/2}{d_x}}}\right) \\ \cos\left(\frac{i}{10000^{2\frac{d_x/2}{d_x}}}\right) \end{pmatrix}$$



Positional encoding (5)

- **Sinusoidal position representations**: concatenate sinusoidal functions of varying periods:





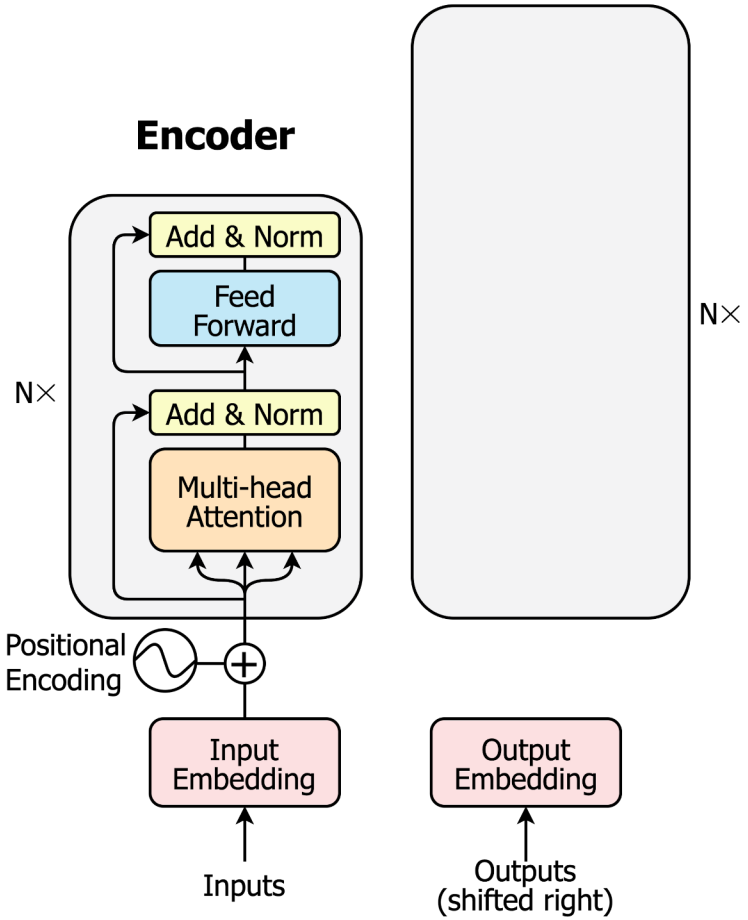
- **Advantages:**
  - Periodicity indicates that maybe "absolute position" isn't as important
  - Maybe can extrapolate to longer sequences as periods restart
- **Disadvantages**:
  - Not learnable (also, the extrapolation doesn't really work)

Positional encoding (6)

- **Learned absolute position representations**: Let all $\boldsymbol{p}_i$ be learnable parameters
  - Learn a matrix $\boldsymbol{P} \in \mathbb{R}^{T \times d_x}$ and let each $\boldsymbol{p}_i$ be a row of that matrix

- **Advantages:**
  - Flexibility: each position gets to be learned to fit the data
- **Disadvantages**:
  - Definitely can't extrapolate to indices outside $1, \ldots, T$
- Most systems use this

Positional encoding in transformer

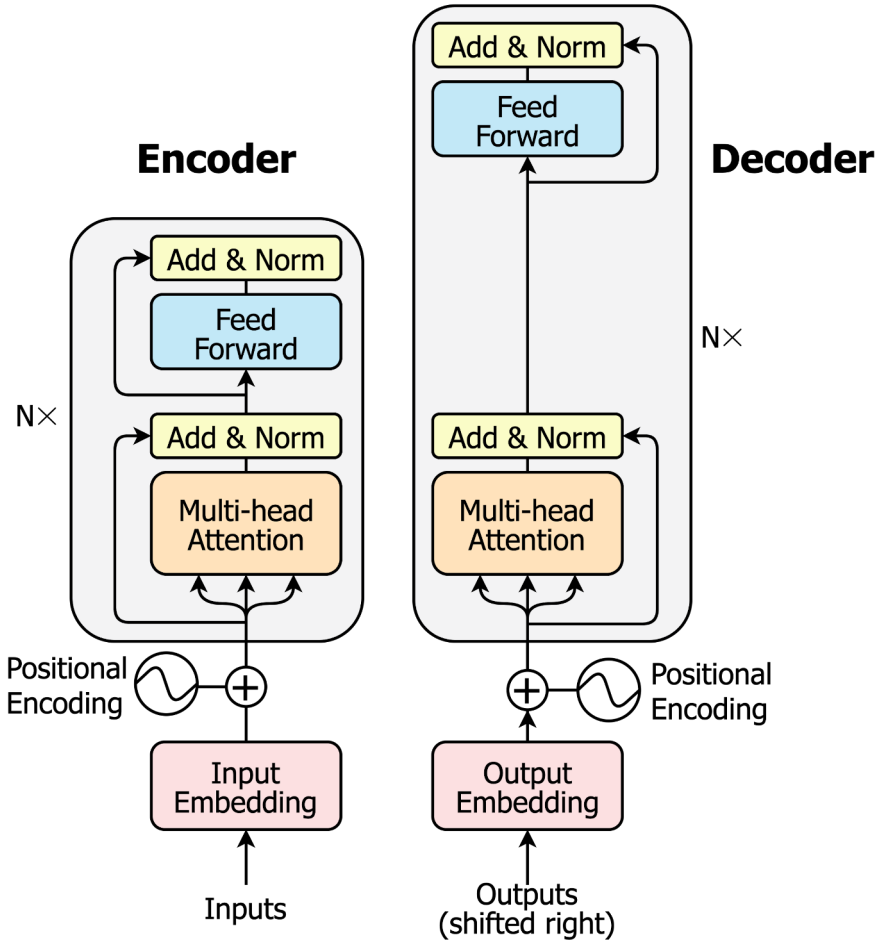


Outline

- Introduction
- Self-attention
- Multi-head attention
- Deep layers
- Positional encoding
- **The Transformer**

Transformer: decoder

- We can also exploit self-attention in the decoder



Self-attention in the decoder

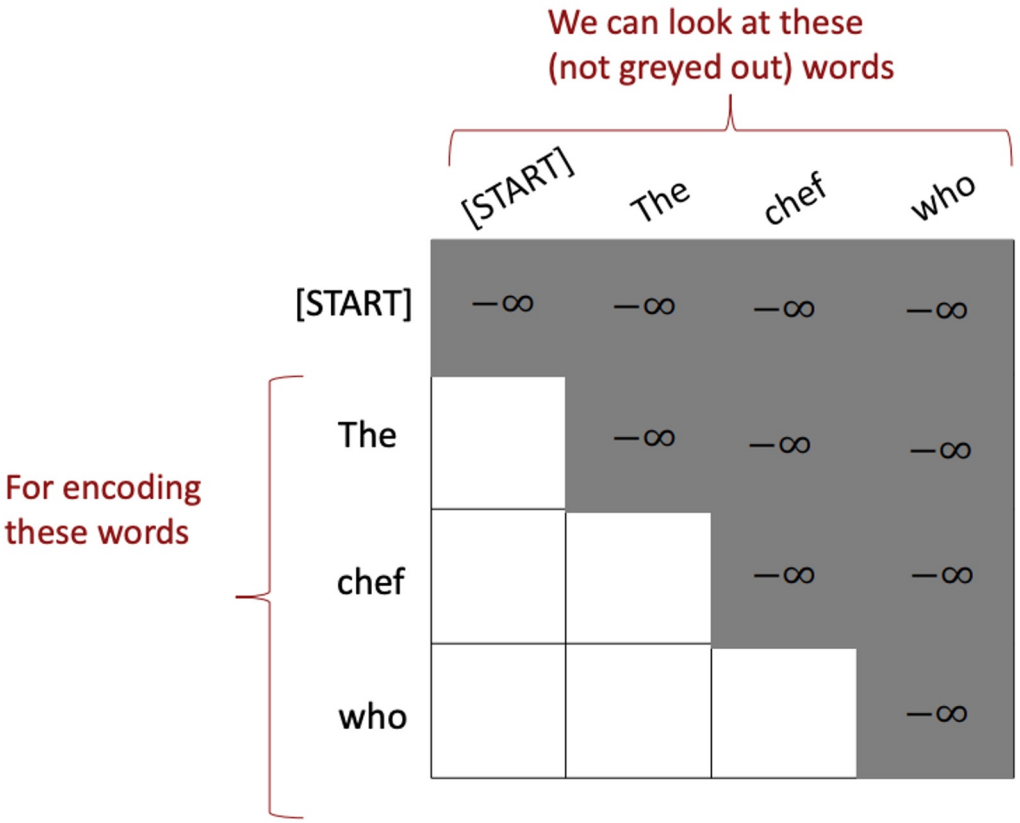
However!

- At training time, the decoder is feed the complete output sequence
- **How do we keep the decoder from cheating (not being able to look-ahead)?**
- **Solution: Masked attention**
  - Hide information about future tokens from the model

The little black dog [masked] [masked] [masked] [masked]

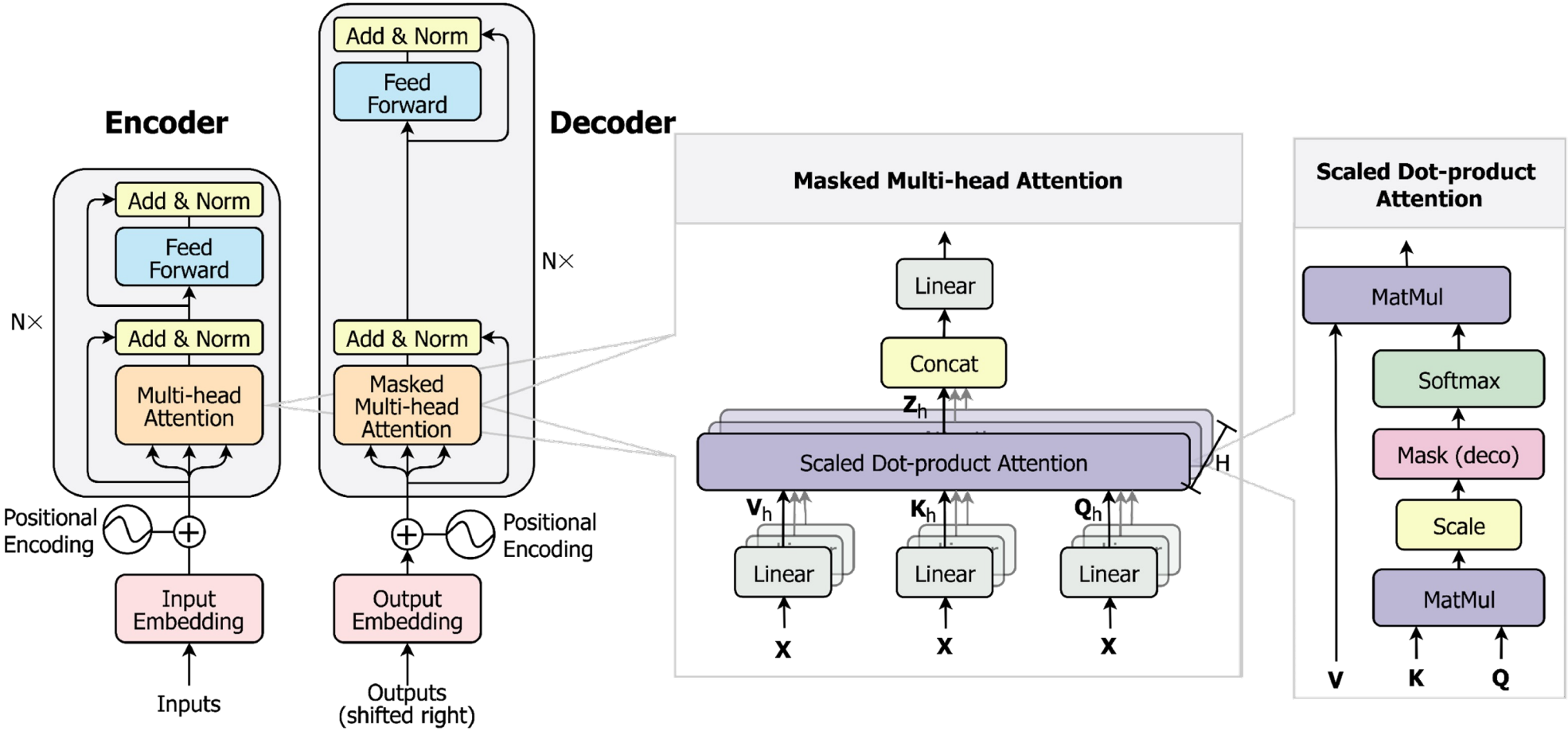
Masked self-attention (1)

- To enable parallelization, mask attention to future tokens by setting attention scores to -∞



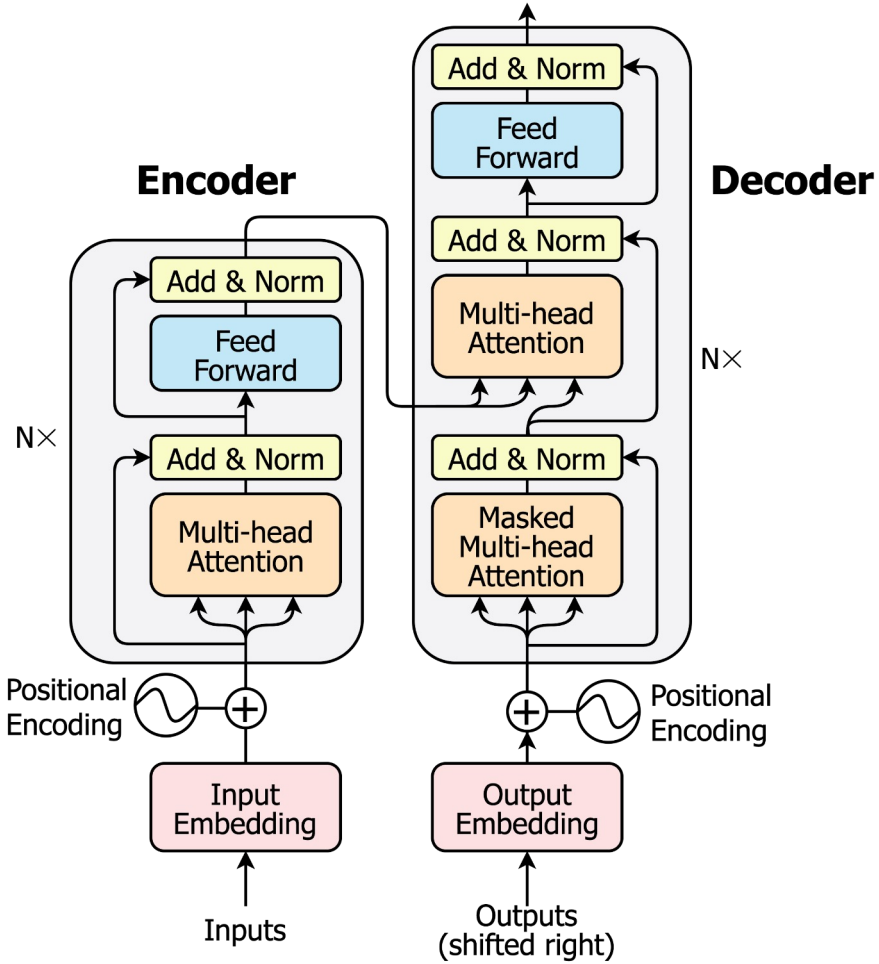
$$e_{ij} = \begin{cases} q_i \cdot k_j & j < i \\ -\infty & j \geq i \end{cases}$$

Masked self-attention (2)



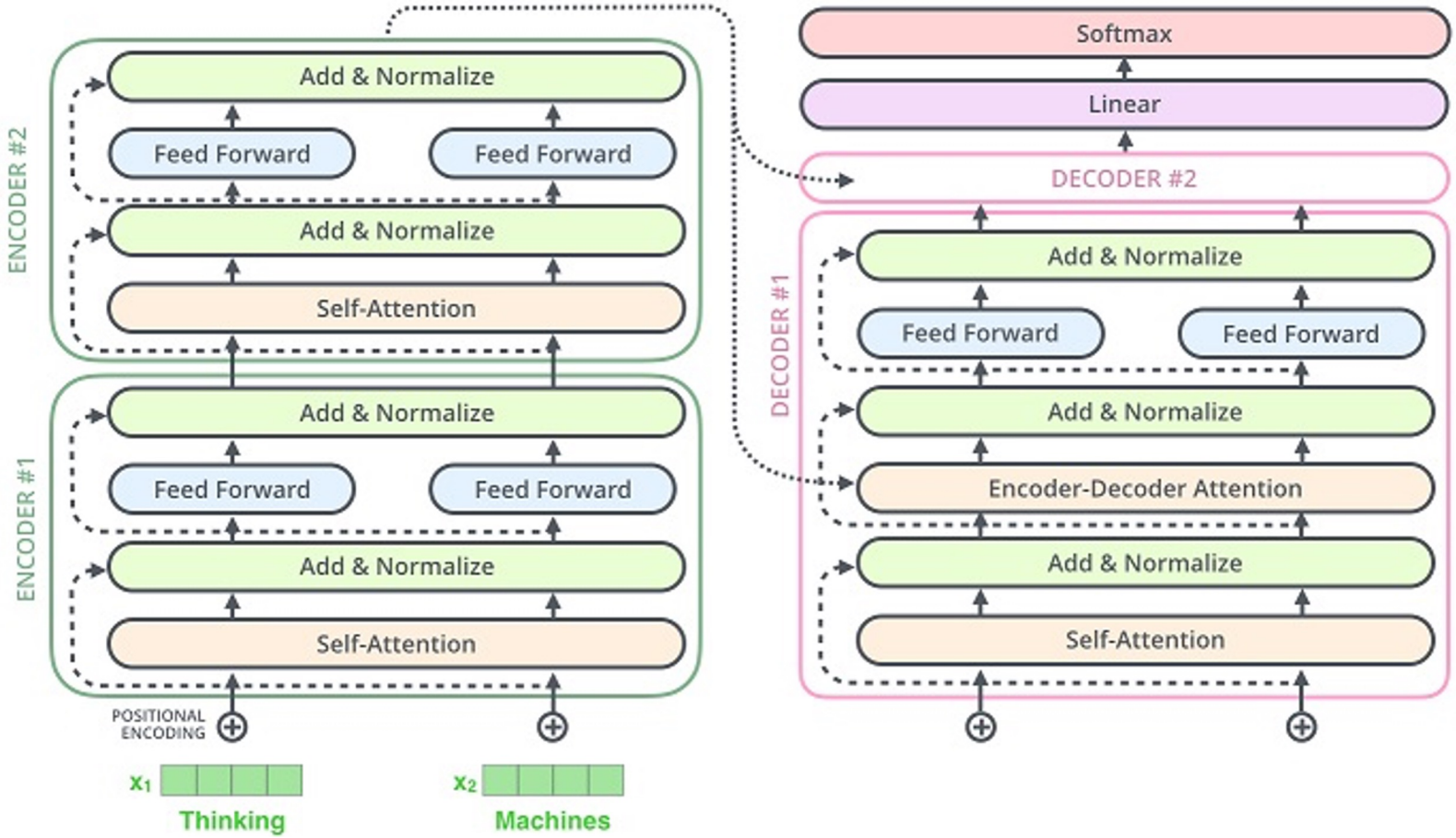
Encoder-Decoder attention (1)

- Use cross-attention in the decoder, where:
  - Keys and values are obtained from the encoder (like a memory)
  - Queries are obtained from the decoder

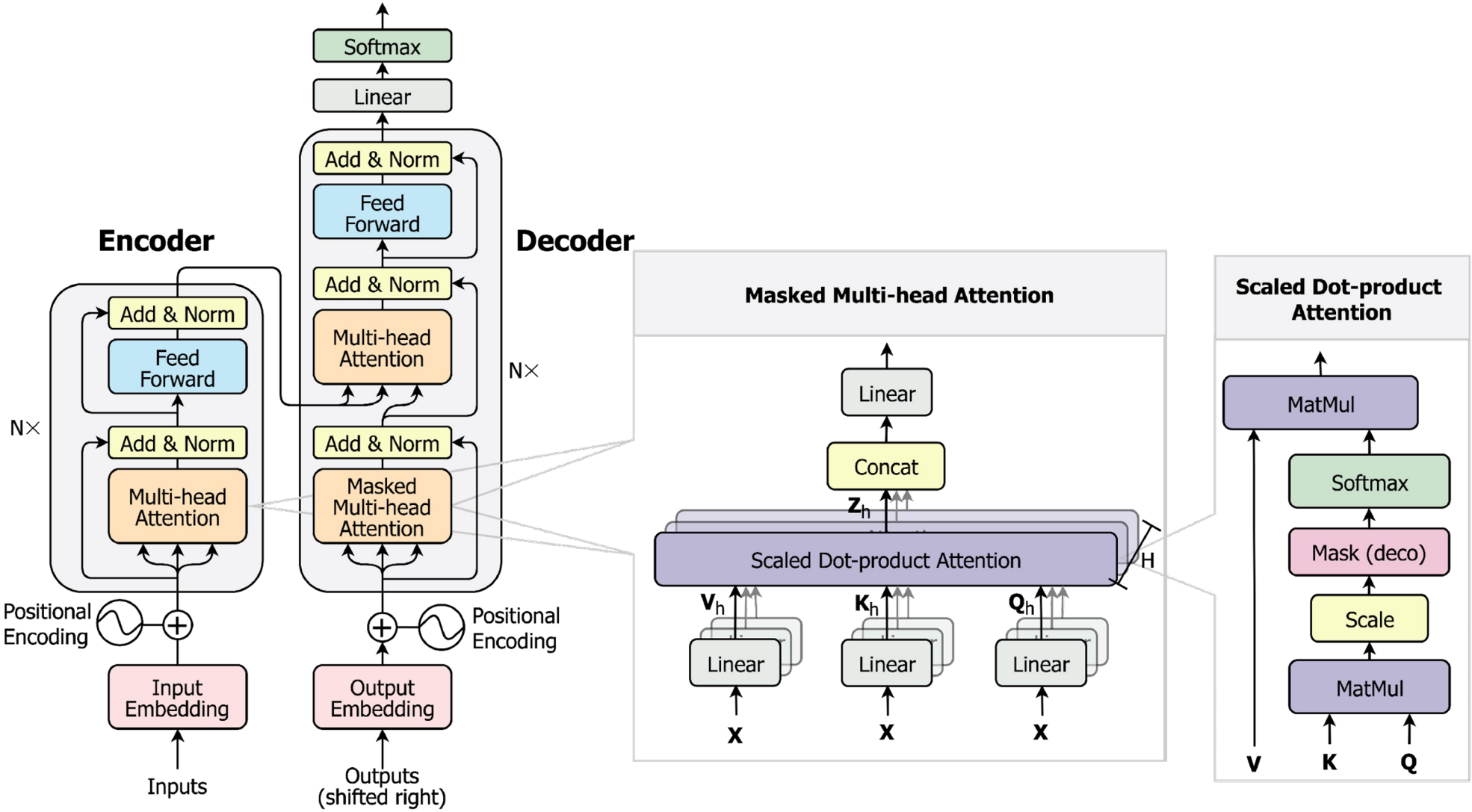


Encoder-Decoder attention (2)

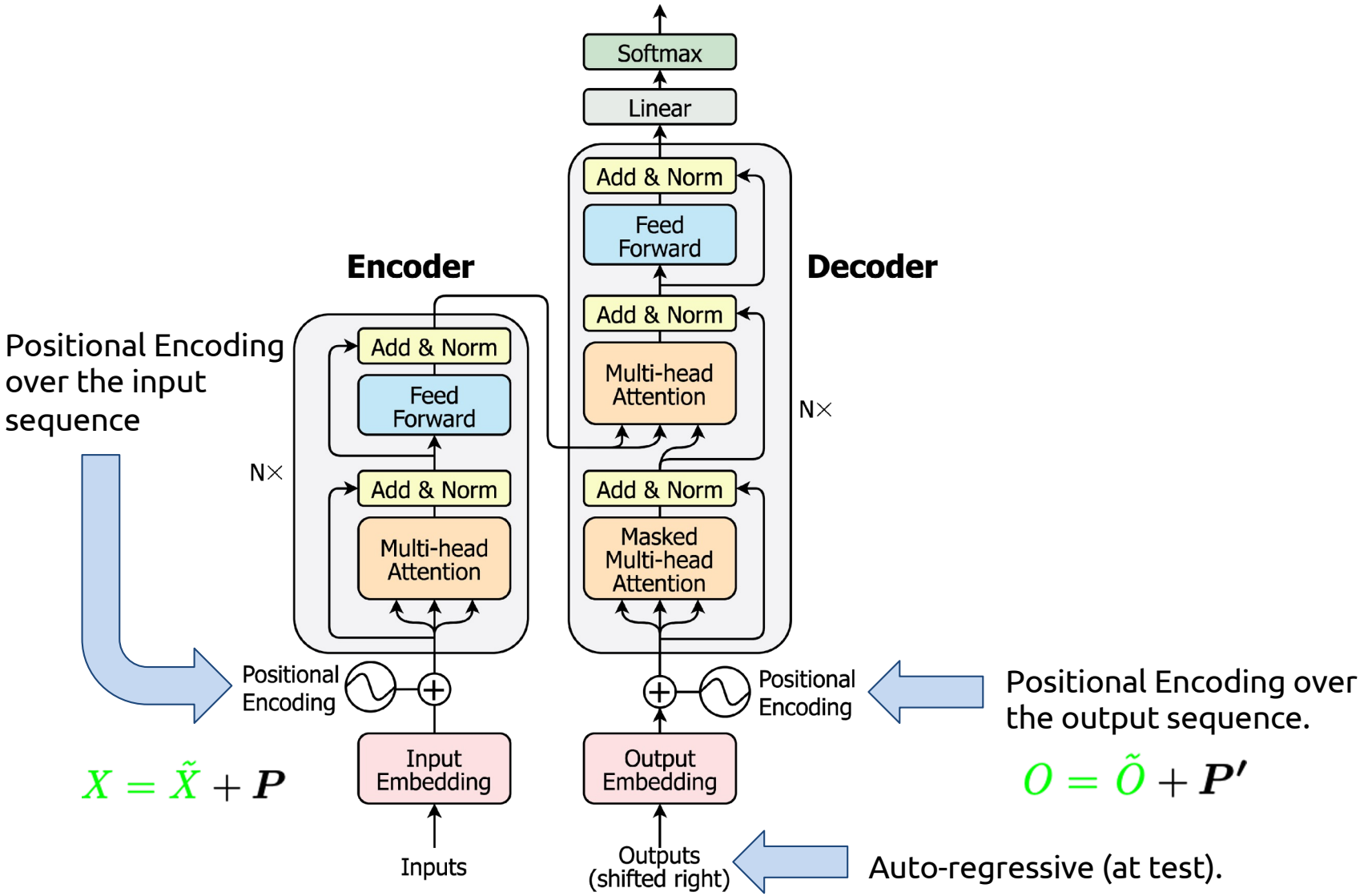
- The last output of the encoder is fed into all decoder layers



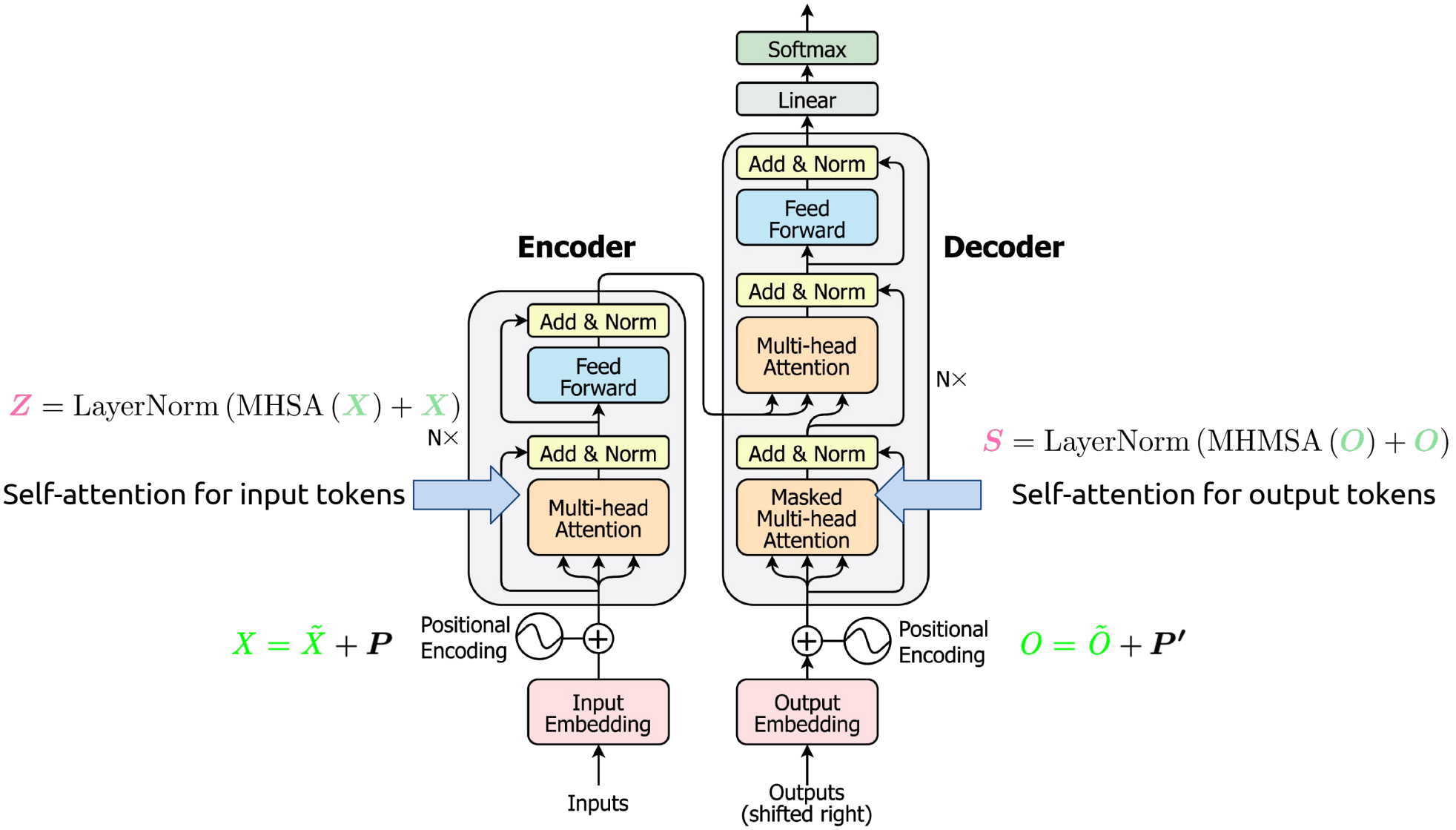
Complete transformer block (1)



Complete transformer block (2)



Complete transformer block (3)

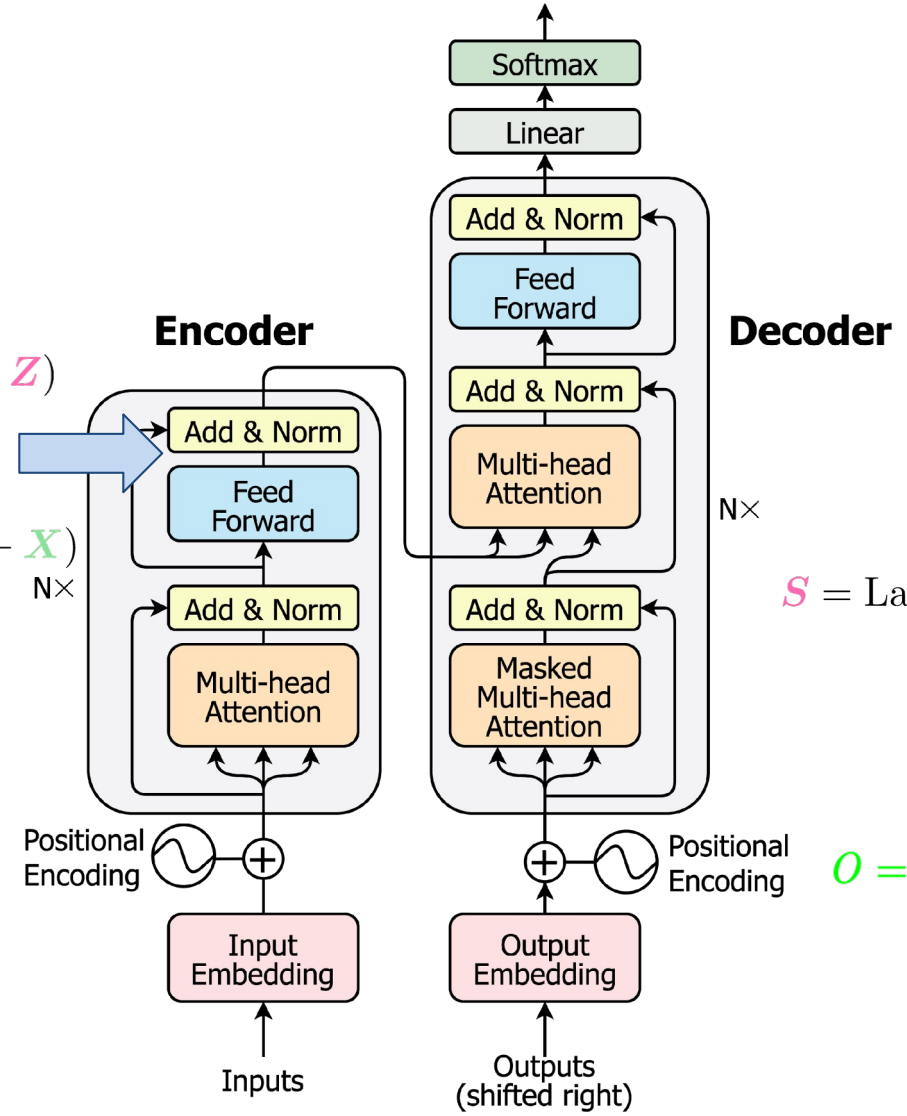


Complete transformer block (4)

$C = \text{LayerNorm}\left(\text{MLP}\left(Z\right) + Z\right)$

Feed forward network

$Z = \text{LayerNorm}\left(\text{MHSA}\left(X\right) + X\right)$

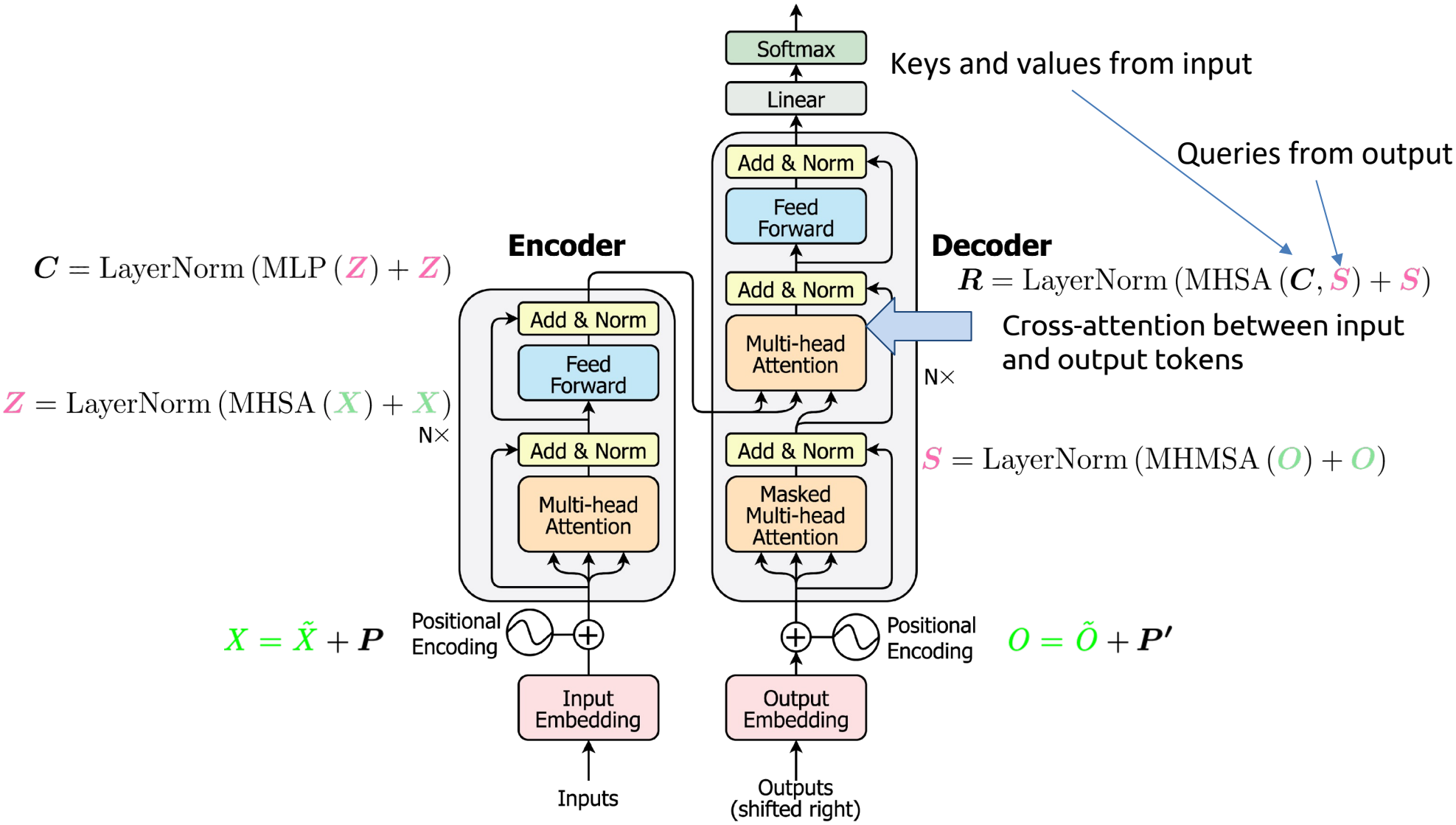


$S = \text{LayerNorm}\left(\text{MHMSA}\left(O\right) + O\right)$

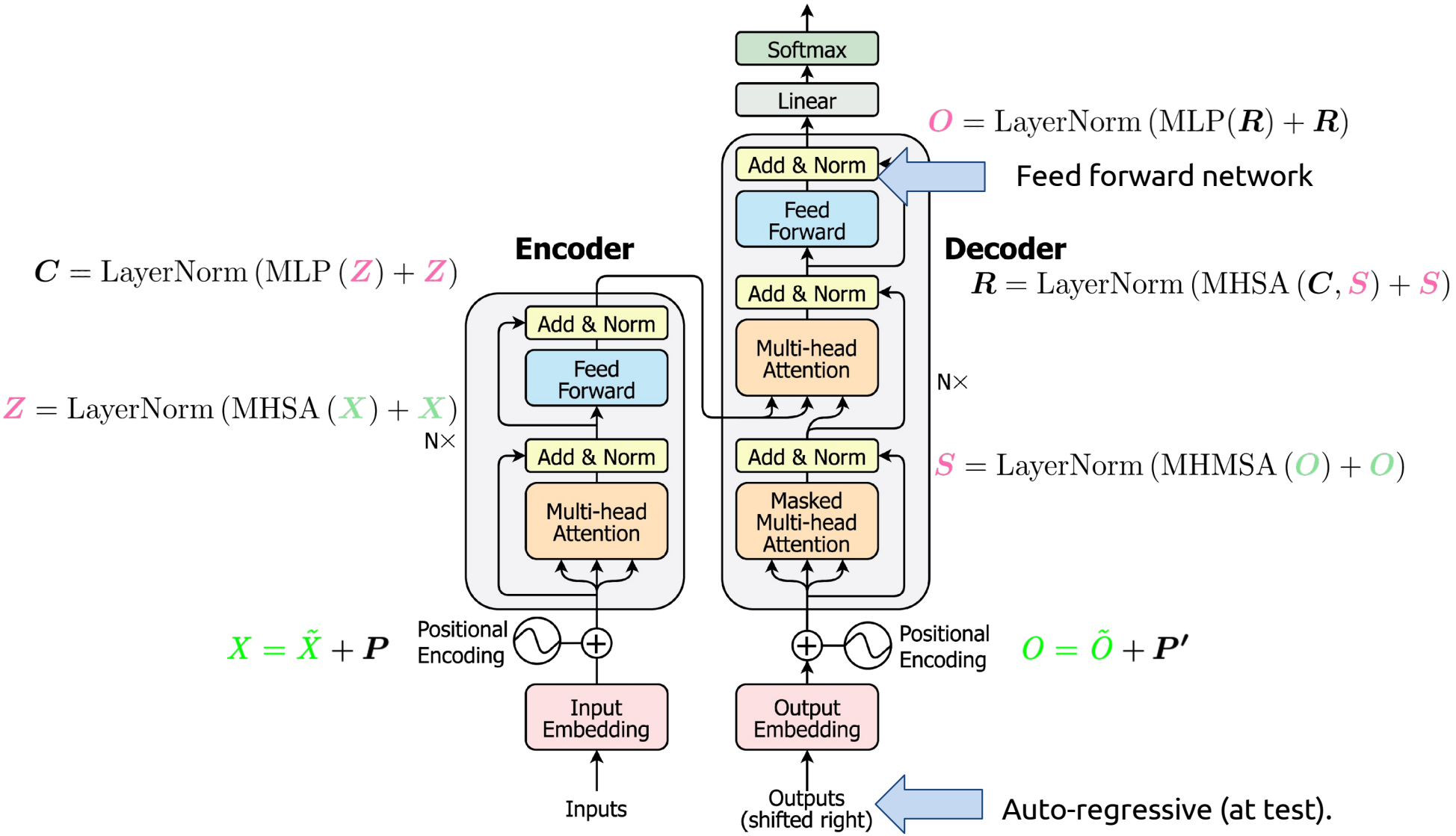
$X = \tilde{X} + \boldsymbol{P}$

$O = \tilde{O} + \boldsymbol{P'}$

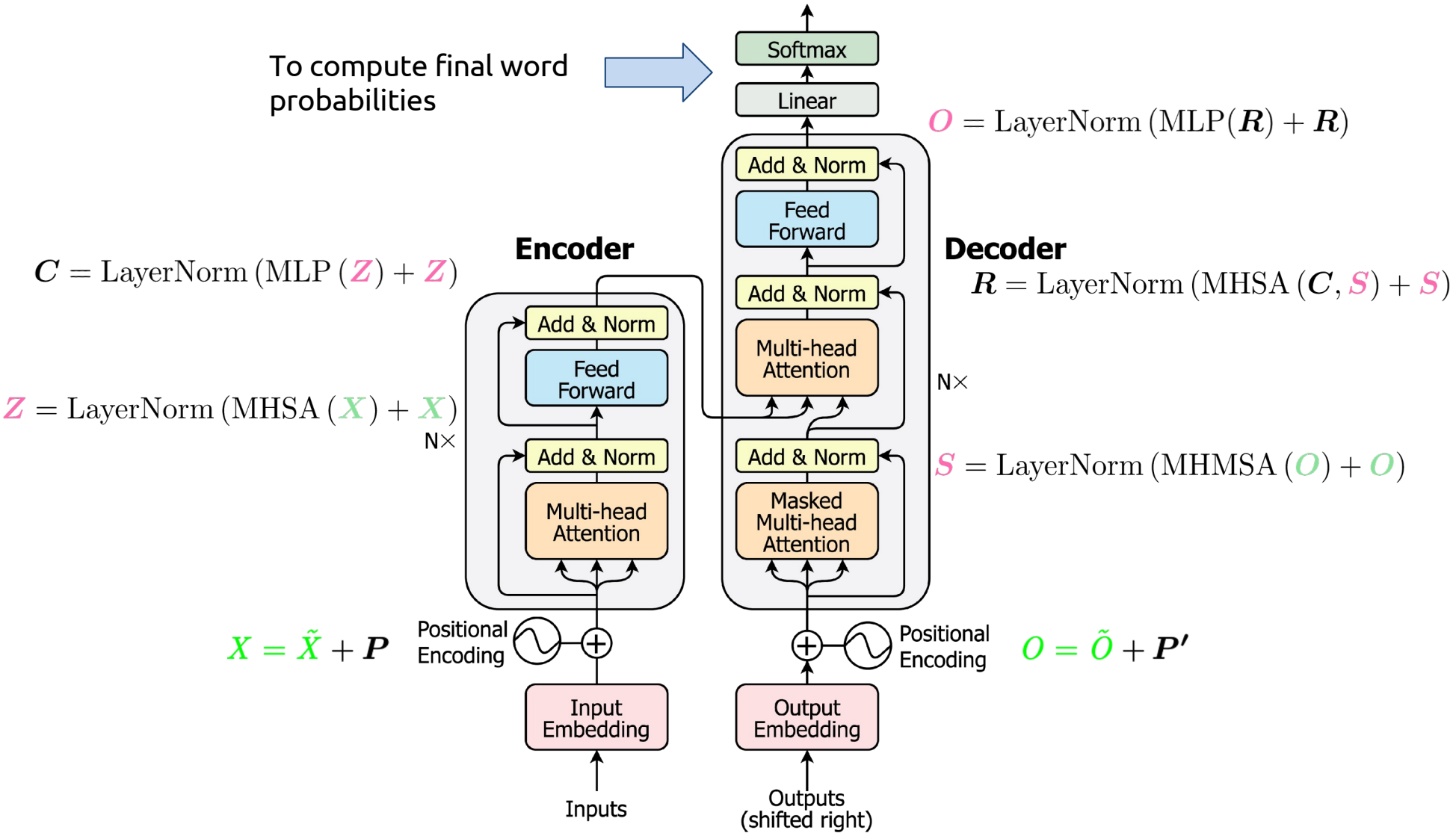
Complete transformer block (5)



Complete transformer block (6)



Complete transformer block (7)



Summary

- Transformers are a **powerful tool** that exploits attention:
  - Each "word" interacts with each other, so maximum interaction distance is **O(1)**
- Their architecture has been ported to **many domains**:
  - language, audio, vision
- **Interpretability** is easier with attention
- They are **hard to optimize** and do not play well with other blocks
- Computing all pairs of interactions means computation **grows quadratically** with sequence length
- More hyperparameters to optimize (num_heads, q/k/v embedding size, etc.)
- The use of **position** representation is still **weak**

Want to know more?

- Tutorials
  - Sebastian Ruder, Deep Learning for NLP Best Practices # Attention, 2017
  - Chris Olah, et al. Attention and Augmented Recurrent Neural Networks distill.pub 2016
  - Lilian Weg, The Transformer Family. Lil'Log 2020
- Scientific publications
  - **#Perceiver** A. Jaegle, et al. Perceiver: General perception with iterative attention, arXiv:2103.03206, 2021
  - **#Longformer** Beltagy, et al. Longformer: The long-document transformer." arXiv preprint arXiv:2004.05150, 2020
  - A. Katharopoulos, et al. Transformers are RNS: Fast autoregressive transformers with linear attention, ICML, 2020.
  - M. Siddhant et al. Multiplicative Interactions and Where to Find Them". ICLR, 2020
  - Self-attention in language
    - J. Cheng, et al. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733, 2016
  - Self-attention in images
    - N. Parmar, et al. Image transformer. ICML, 2018
    - X. Wang, et al. Non-local neural networks, CVPR, 2018

Questions?

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Department of Signal Theory and Communications
Image Processing Group