Attention & Transformers

Ivo Verhoeven | Advanced Topics in Computational Semantics

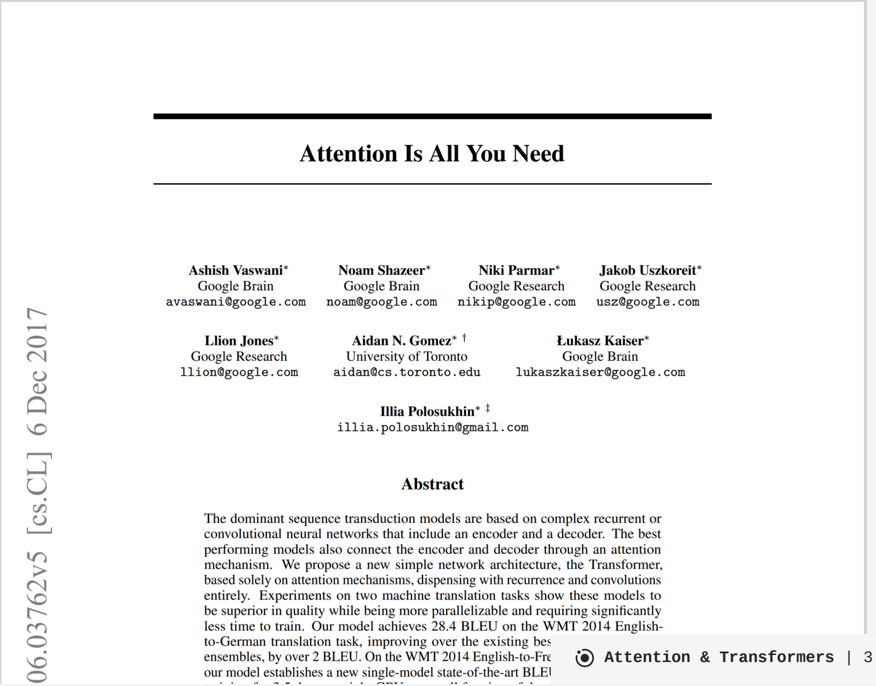
About Me

- 2017 - 2020: BSc. Liberal Arts & Sciences
- 2020 – 2022: MSc. AI at University of Amsterdam
  - Thesis on meta-learning, morphology and translation
  - Took ATCS in 2021
- 2022 - ???: PhD at ILLC
  - Misinformation detection and generalisation with Katia Shutova

Vaswani et al.: Attention is All You Need

- Introduces the Transformer architecture in late 2017
  - Google Brain/Google Research collab

Vaswani et al. (2017). Attention is all you need. Advances
neural information processing systems, 30.



06.03762v5 [cs.CL] 6 Dec 2017

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

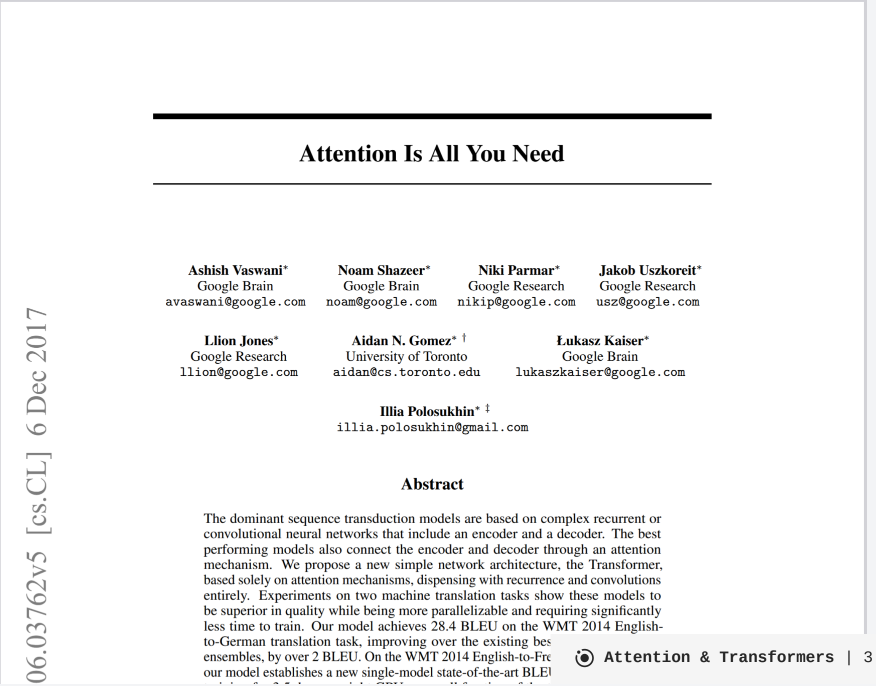
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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 bes
ensembles, by over 2 BLEU. On the WMT 2014 English-to-Fre
our model establishes a new single-model state-of-the-art BLE

Vaswani et al.: Attention is All You Need

- Introduces the Transformer architecture in late 2017
    - Google Brain/Google Research collab

- Paper currently has **169 248** citations
    - Or **~64 citations a day**

`Vaswani et al. (2017). Attention is all you need. Advances neural information processing systems, 30.`



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

06.03762v5 [cs.CL] 6 Dec 2017

Abstract

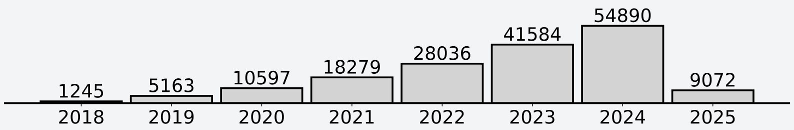
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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 bes ensembles, by over 2 BLEU. On the WMT 2014 English-to-Fre our model establishes a new single-model state-of-the-art BLE

Vaswani et al.: Attention is All You Need

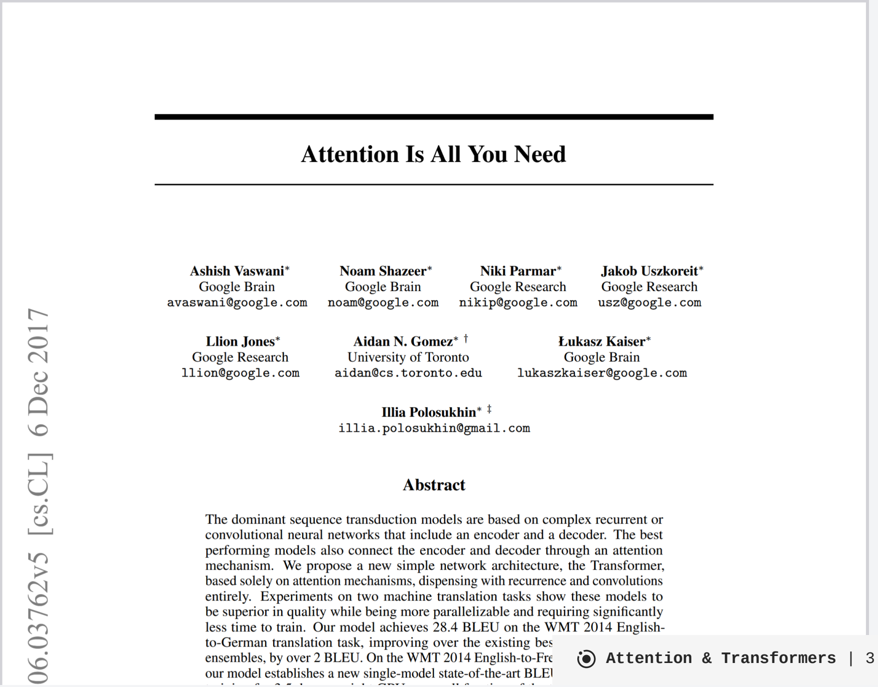
- Introduces the Transformer architecture in late 2017
  - Google Brain/Google Research collab

- Paper currently has **169 248** citations
  - Or **~64 citations a day**

- Number of citations is only accelerating



Vaswani et al. (2017). Attention is all you need. Advances neural information processing systems, 30.



arXiv:1706.03762v5 [cs.CL] 6 Dec 2017

Attention Is All You Need

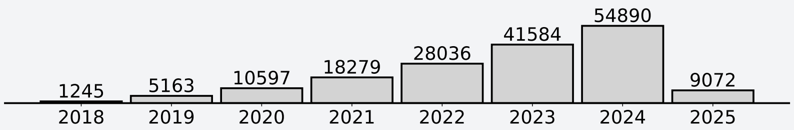
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* 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 bes ensembles, by over 2 BLEU. On the WMT 2014 English-to-Fre our model establishes a new single-model state-of-the-art BLE

Vaswani et al.: Attention is All You Need

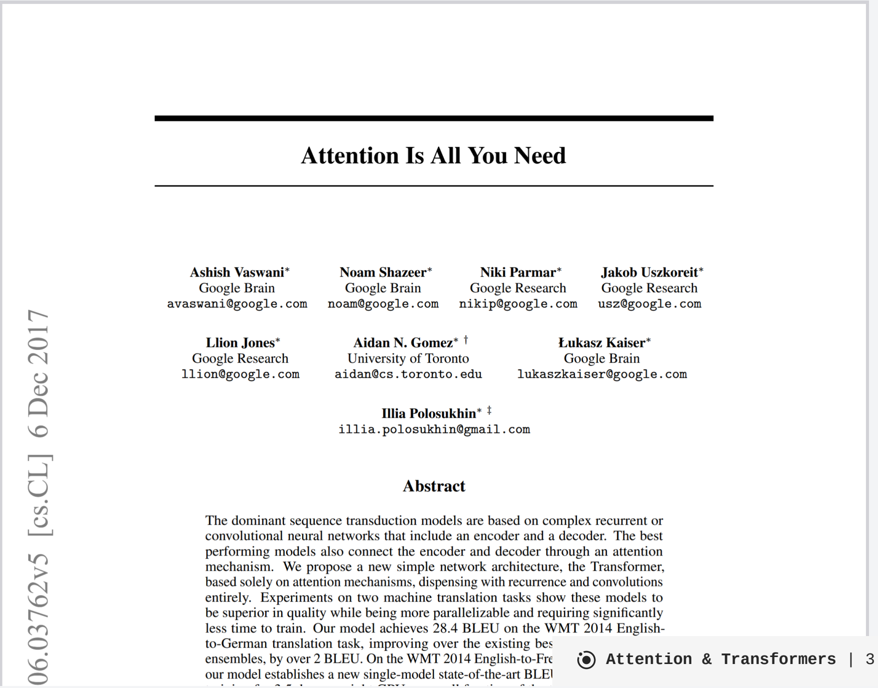
- Introduces the Transformer architecture in late 2017
  - Google Brain/Google Research collab
- Paper currently has **169 248** citations
  - Or **~64 citations a day**
- Number of citations is only accelerating



- Most cited paper ever has **233 829** citations

  `Lowry et al. (1951) Protein measurement with the folin phenol reagent.`

`Vaswani et al. (2017). Attention is all you need. Advances neural information processing systems, 30.`



06.03762v5 [cs.CL] 6 Dec 2017

Attention Is All You Need

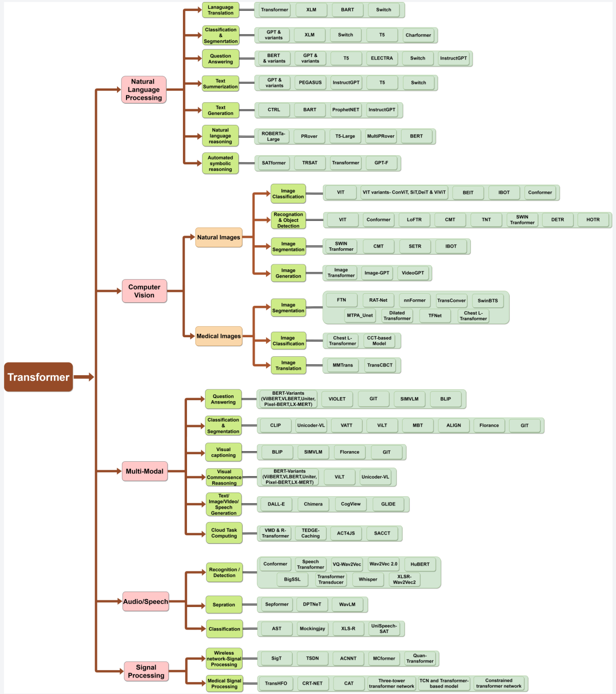
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* 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 bes ensembles, by over 2 BLEU. On the WMT 2014 English-to-Fre our model establishes a new single-model state-of-the-art BLE

Vaswani et al.: Attention is All You Need

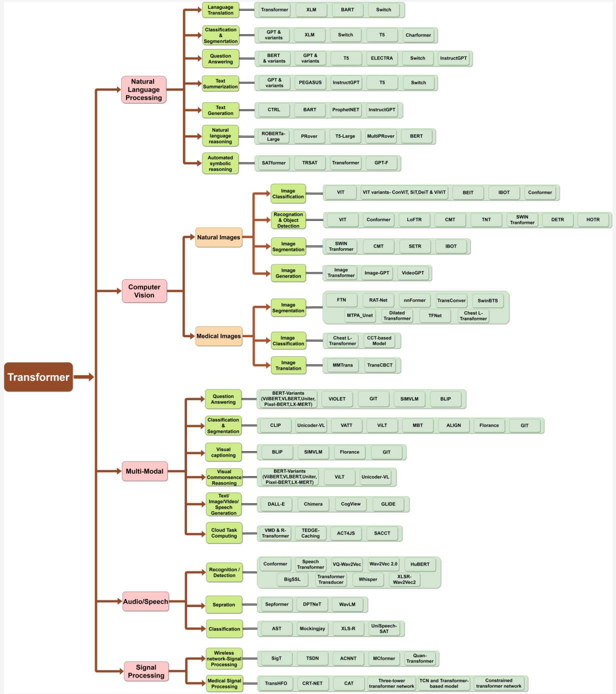
- It’s hard to think of an AI area that hasn’t been affected by the Transformer



Islam, et al. (2023). A Comprehensive Survey on Application Transformers for Deep Learning Tasks. arXiv:2306.07303.

Vaswani et al.: Attention is All You Need

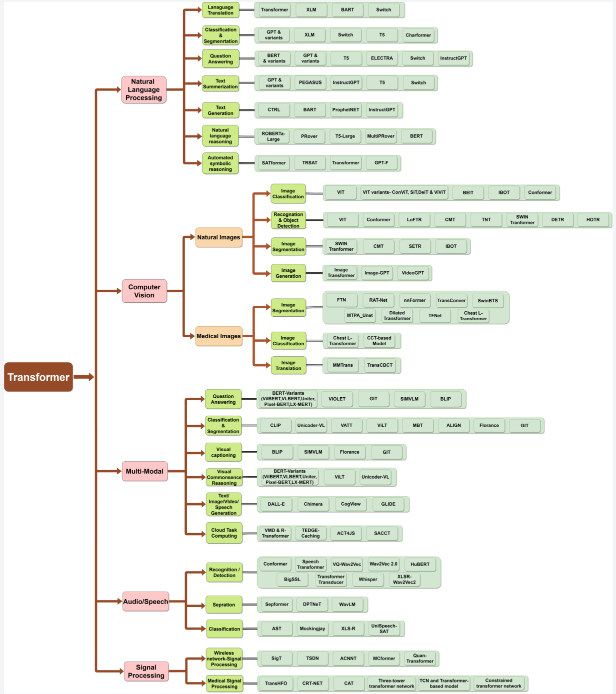
- It's hard to think of an AI area that hasn't been affected by the Transformer
  - **NLP:** Transformer > RNN
    - Seq-to-seq: what it was designed for
    - Classification: encoder-only transformers
    - Generation: decoder-only transformers



Islam, et al. (2023). A Comprehensive Survey on Application
Transformers for Deep Learning Tasks. arXiv:2306.07303.

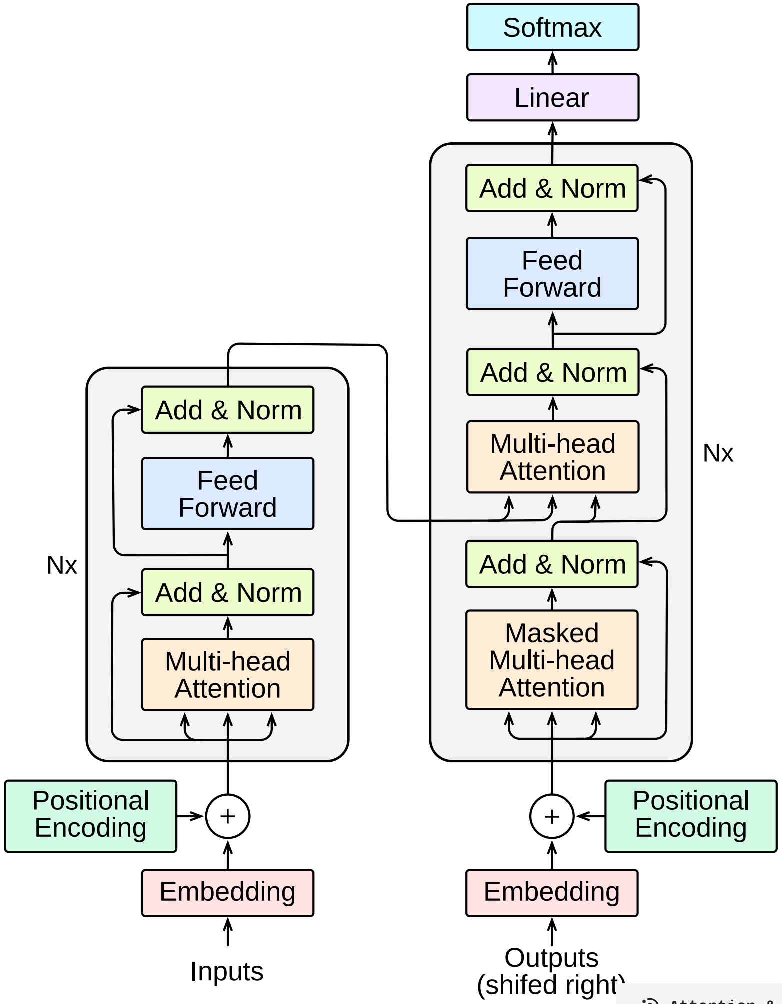
Vaswani et al.: Attention is All You Need

- It's hard to think of an AI area that hasn't been affected by the Transformer
  - **NLP:** Transformer > RNN
    - Seq-to-seq: what it was designed for
    - Classification: encoder-only transformers
    - Generation: decoder-only transformers
  - **CV:** ViT > CNN
  - **Multi-modal:** Transformer > different architectures
  - **Speech:** Transformer > CNN
  - **Graphs:** Transformer/Attention > GCN

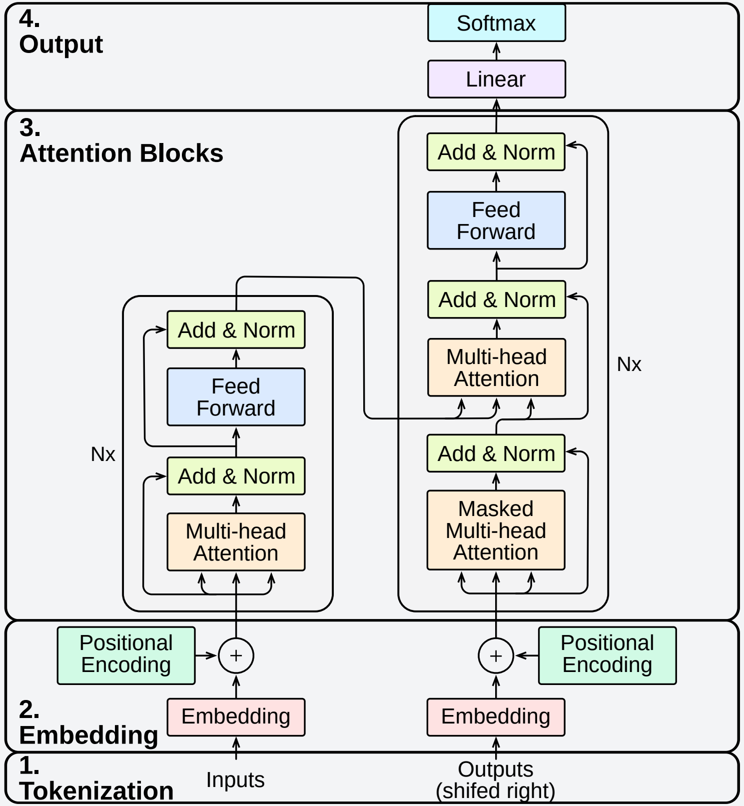


Islam, et al. (2023). A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. arXiv:2306.07303.

The Transformer

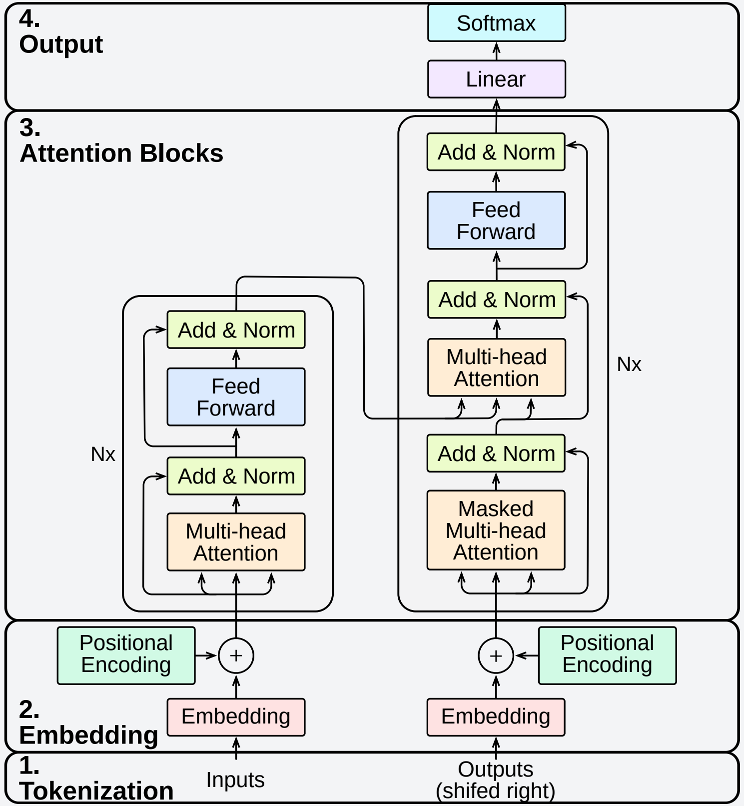


Breaking the Transformer into modules



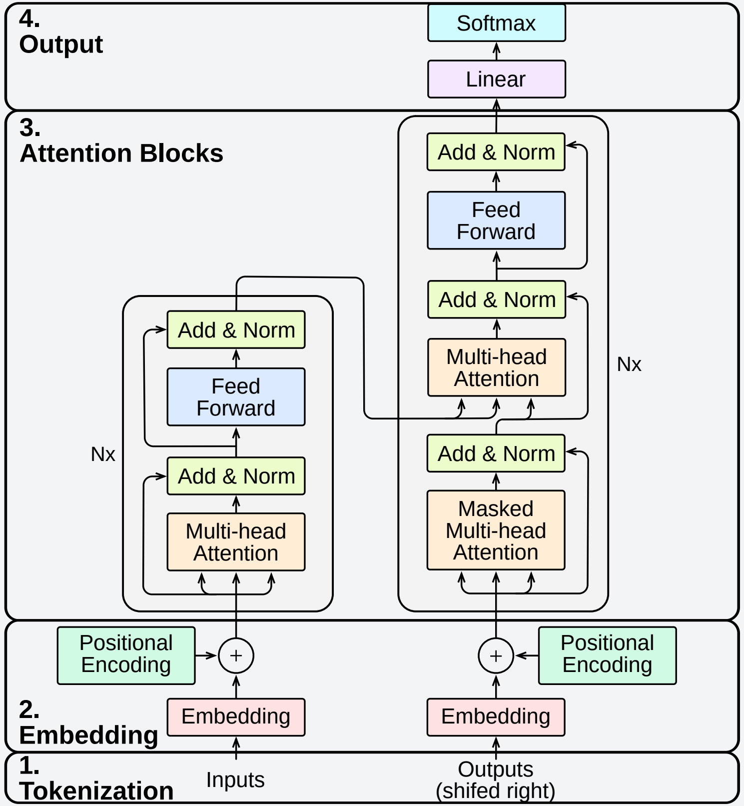
Breaking the Transformer into modules

4. Output
    - Softmax
    - Linear



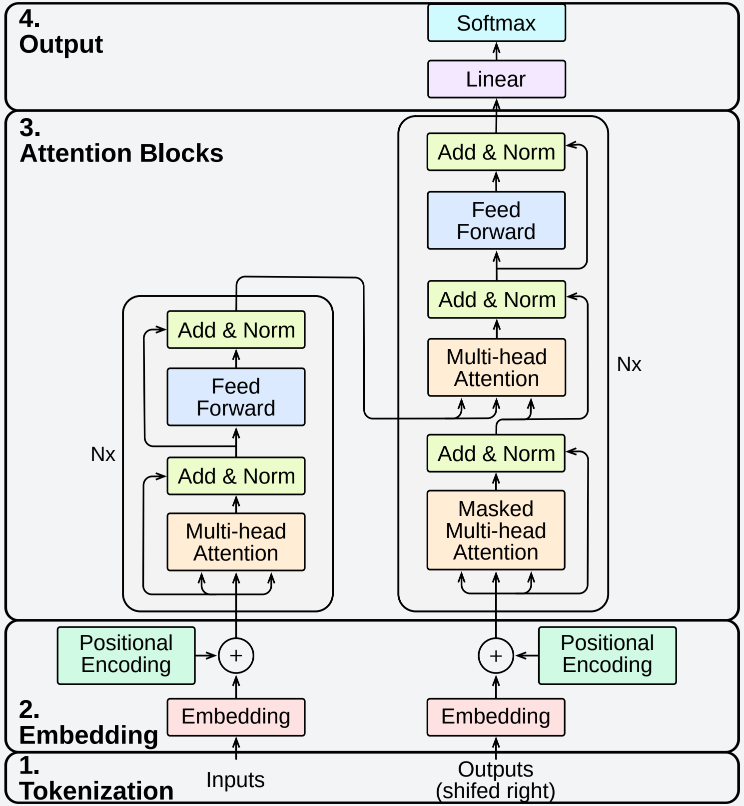
Breaking the Transformer into modules

4. Output
   - Softmax
   - Linear
3. Attention Blocks
   - Multi-head Attention
   - Add & Norm
   - Feed Forward



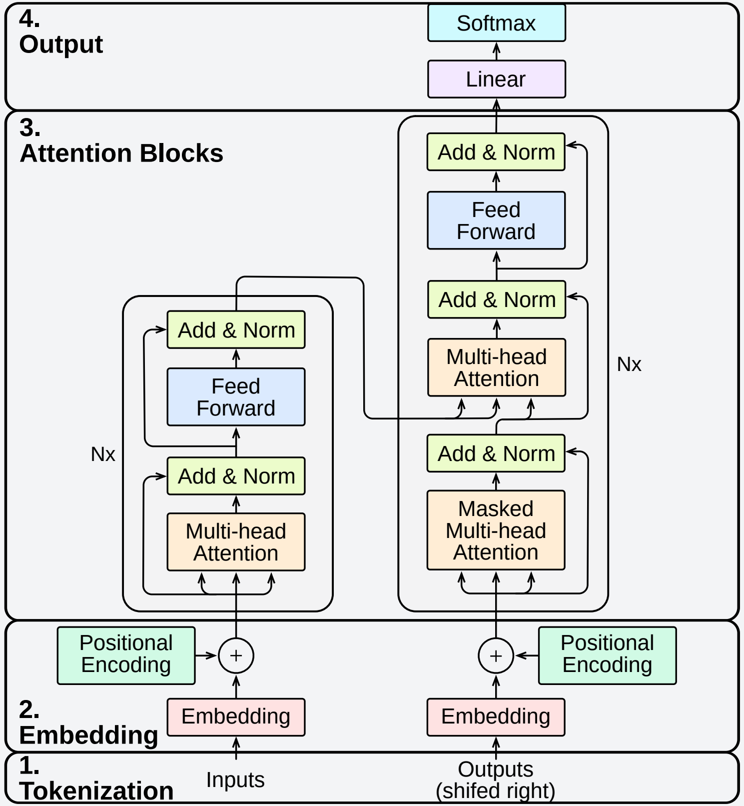
Breaking the Transformer into modules

4. Output
   - Softmax
   - Linear
3. Attention Blocks
   - Multi-head Attention
   - Add & Norm
   - Feed Forward
2. Embedding
   - Token Embedding
   - Positional Encoding



Breaking the Transformer into modules

4. Output
    - Softmax
    - Linear
3. Attention Blocks
    - Multi-head Attention
    - Add & Norm
    - Feed Forward
2. Embedding
    - Token Embedding
    - Positional Encoding
1. Tokenization
    - (Not pictured)



Breaking the Transformer into modules

4. ~~Output~~
    - ~~Softmax~~
    - ~~Linear~~
3. Attention Blocks
    - Multi-head Attention
    - Add & Norm
    - Feed Forward
2. Embedding
    - ~~Token Embedding~~
    - Positional Encoding
1. Tokenization
    - (Not pictured)

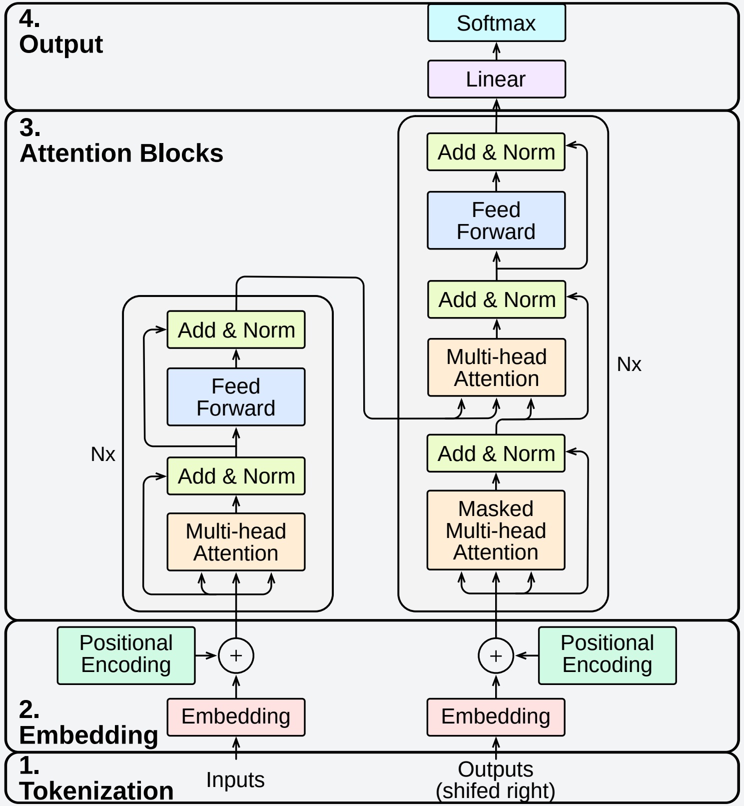
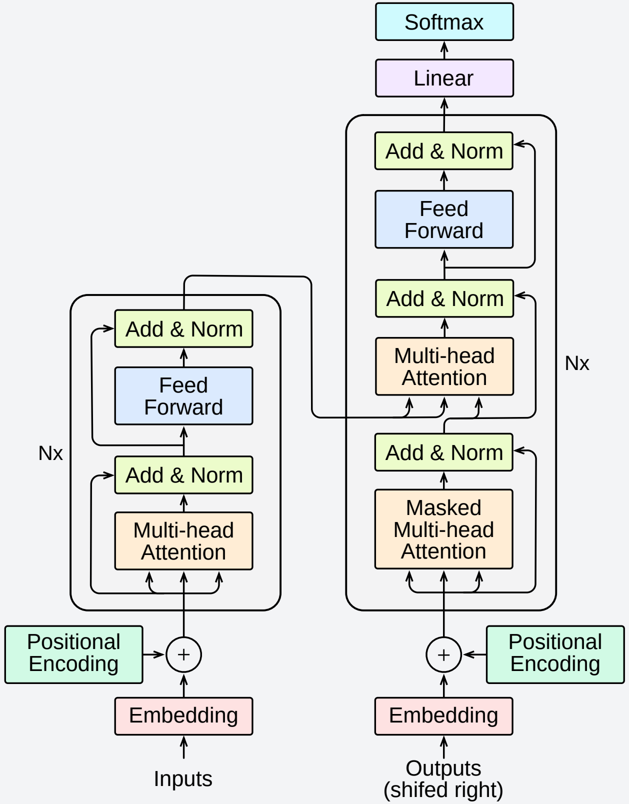


Table of Contents



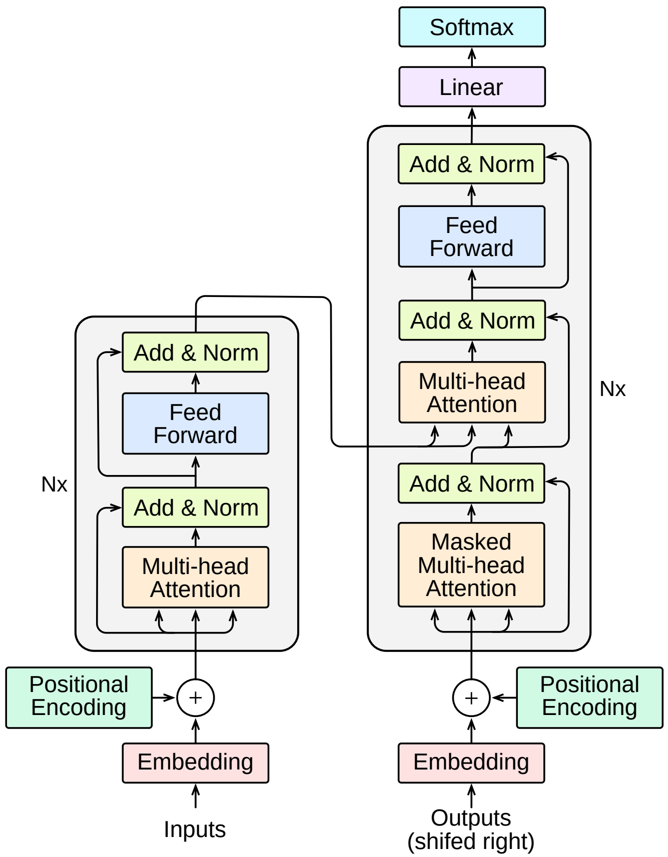
Encoders & Decoders

Text comes in, text goes out

Attention Blocks

What makes the Transformer what it is — and where it came from

Multi-head Attention



Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of (word) vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_V

$$\texttt{Attention}(?, ?, \mathbf{V}) = \mathbf{A}\mathbf{V}$$

$$\mathbf{A} \in (0, 1)^{[t_V \times t_V]}$$

$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_V]}$$

Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of (word) vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_V

$$\texttt{Attention}(?, ?, \mathbf{V}) = \mathbf{A}\mathbf{V}$$

$$\mathbf{A} \in (0, 1)^{[t_V \times t_V]}$$

$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_V]}$$

- $\texttt{Attention}$ is just a matrix product of $\mathbf{V}$ with an attention matrix $\mathbf{A}$
  - $\mathbf{A}$ is a square matrix of size $t_V \times t_V$
  - It's elements are all between $(0, 1)$
  - It's rows sum to 1

Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of (word) vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_V
- $\texttt{Attention}$ is just a matrix product of $\mathbf{V}$ with an attention matrix $\mathbf{A}$
  - $\mathbf{A}$ is a square matrix of size $t_V \times t_V$
  - It's elements are all between $(0, 1)$
  - It's rows sum to 1

$$\texttt{Attention}(?, ?, \mathbf{V}) = \mathbf{A}\mathbf{V}$$

$$\mathbf{A} \in (0, 1)^{[t_V \times t_V]}$$

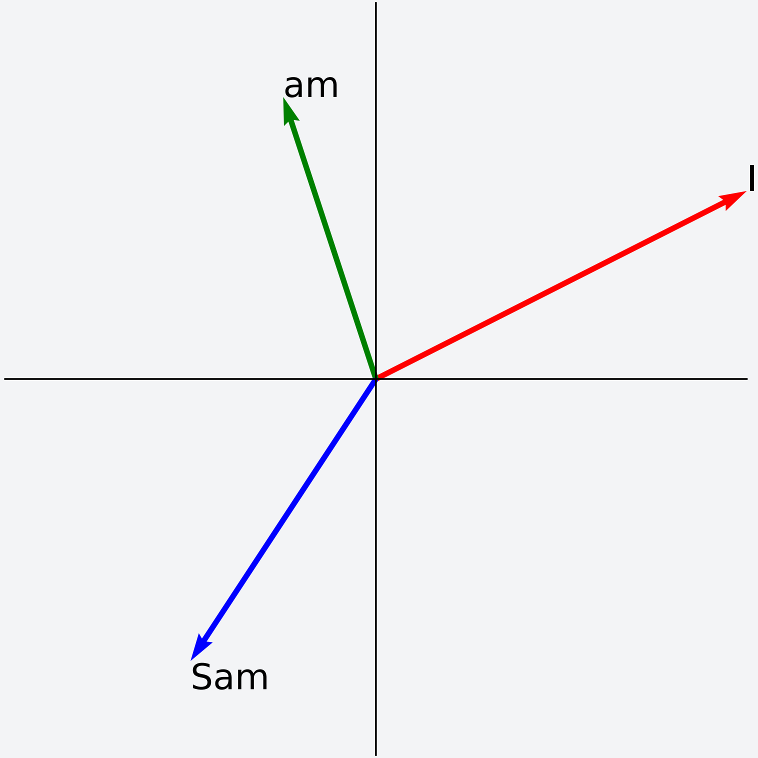
$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_V]}$$

Definition & Properties

Multi-head Attention

- The result of **`Attention`** is just a convex combination of $\mathbf{V}$

$$
\begin{array}{cc}
\mathbf{A} & \mathbf{V} & \\
\begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.1 & 0.7 \end{bmatrix} &
\begin{bmatrix} 2.0 & 1.0 \\ -0.5 & 2.0 \\ -1.0 & -0.5 \end{bmatrix} &
\begin{array}{c} \text{I} \\ \text{am} \\ \text{Sam} \end{array}
\end{array}
$$



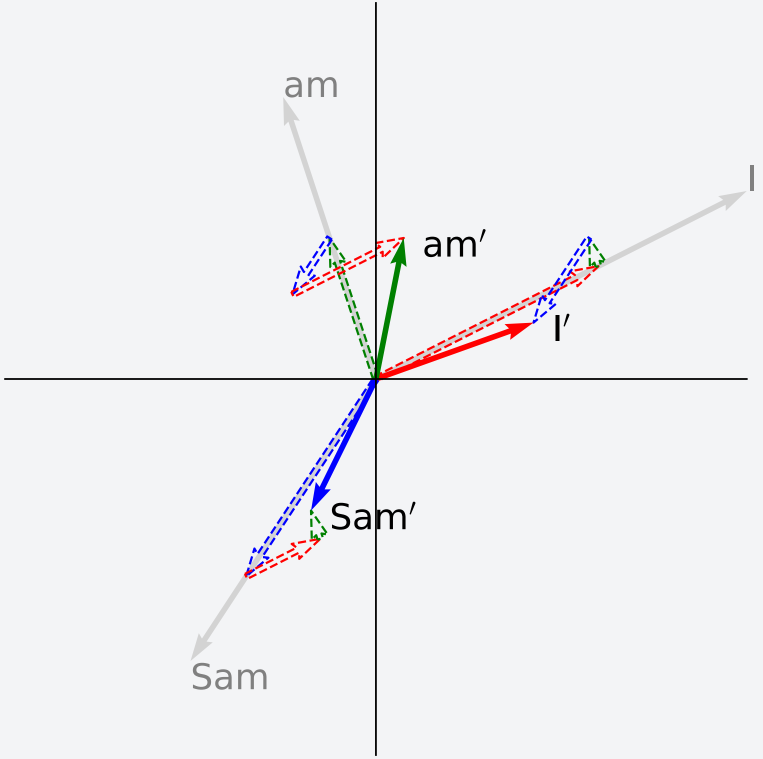
Definition & Properties

Multi-head Attention

- The result of **`Attention`** is just a convex combination of $\mathbf{V}$

$$\overset{\mathbf{A}}{\begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.1 & 0.7 \end{bmatrix}} \overset{\mathbf{V}}{\begin{bmatrix} 2.0 & 1.0 \\ -0.5 & 2.0 \\ -1.0 & -0.5 \end{bmatrix}} \begin{matrix} \color{red}{\mathrm{I}} \\ \color{green}{\mathrm{am}} \\ \color{blue}{\mathrm{Sam}} \end{matrix}$$

$$= \begin{bmatrix} 0.6 * \color{red}{\mathrm{I}} + 0.1 * \color{green}{\mathrm{am}} + 0.3 * \color{blue}{\mathrm{Sam}} \\ 0.3 * \color{red}{\mathrm{I}} + 0.5 * \color{green}{\mathrm{am}} + 0.2 * \color{blue}{\mathrm{Sam}} \\ 0.2 * \color{red}{\mathrm{I}} + 0.1 * \color{green}{\mathrm{am}} + 0.7 * \color{blue}{\mathrm{Sam}} \end{bmatrix}$$

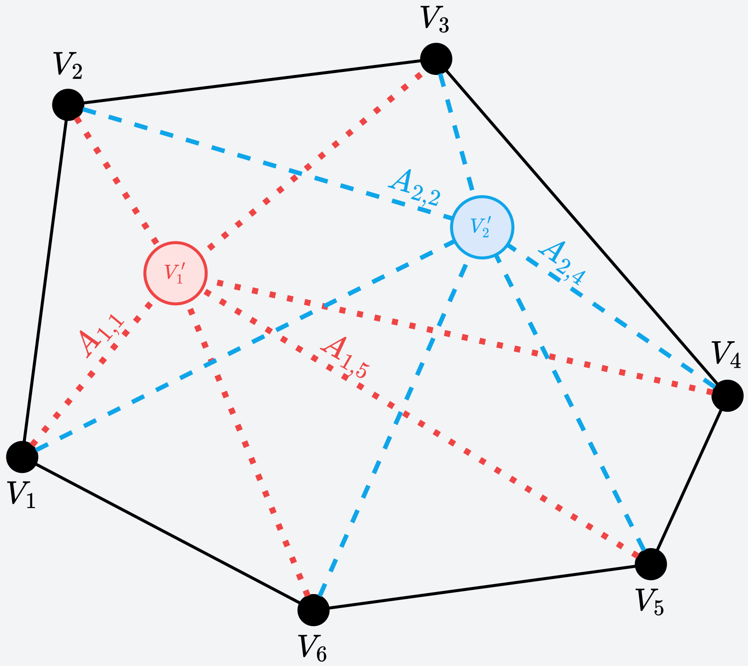


Definition & Properties

Multi-head Attention

Convex Combination

The elements of V' will lie inside the convex hull of all of the elements in V



Definition & Properties

Multi-head Attention

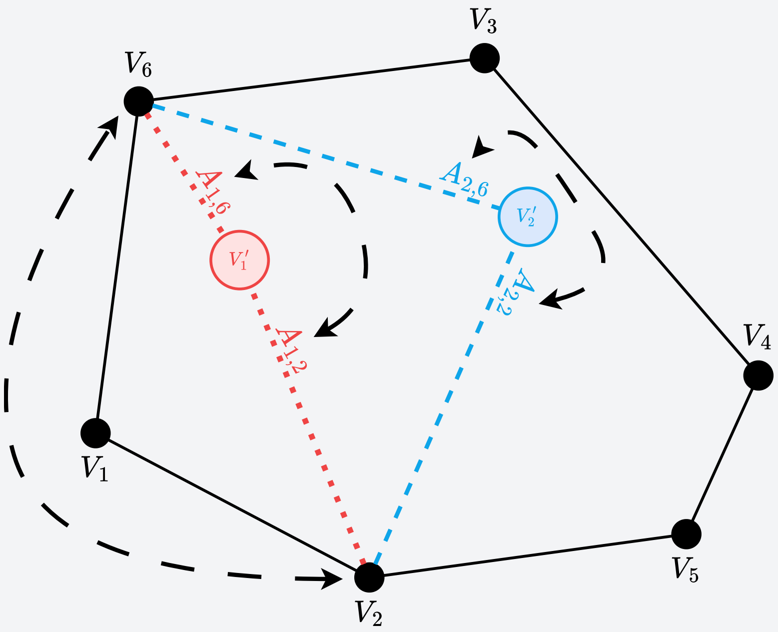
Convex Combination

The elements of V' will lie inside the convex hull of all of the elements in V

Permutation Equivariance

The elements of $\mathbf{V}'$ are *equivariant* to a change in the order of the columns of $\mathbf{A}$ and the rows of $\mathbf{V}$

- Attention does not care about word order
  - 'I am Sam' ~ 'Sam I am'



Definition & Properties

Multi-head Attention

So is `Attention` just a linear map?

- Not quite

Linear maps are:

Definition & Properties

Multi-head Attention

So is `Attention` just a linear map?

- Not quite

Linear maps are:

- Inflexible in terms of sequence length

Definition & Properties

Multi-head Attention

So is `Attention` just a linear map?

- Not quite

Linear maps are:

- Inflexible in terms of sequence length
- Parameter inefficient

Definition & Properties

Multi-head Attention

So is `Attention` just a linear map?

- Not quite

Linear maps are:

- Inflexible in terms of sequence length
- Parameter inefficient
- Invariant to the input content

Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of **value** vectors
  - It has a sequence length of T_V
  - It has a dimensionality of d_V
- Let $\mathbf{K}$ be a matrix of **key** vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_K
- Let $\mathbf{Q}$ be a matrix of **query** vectors
  - It has a sequence length of t_Q
  - It has a dimensionality of d_K

$$\mathtt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \underbrace{\mathtt{softmax}\left(f\left(\mathbf{Q}, \mathbf{K}\right)\right)}_{\mathbf{A}}\mathbf{V}$$

$$\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$$

$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_v]}$$

$$\mathbf{K} \in \mathbb{R}^{[t_V \times d_k]}$$

$$\mathbf{Q} \in \mathbb{R}^{[t_Q \times d_k]}$$

Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of **value** vectors
  - It has a sequence length of T_V
  - It has a dimensionality of d_V
- Let $\mathbf{K}$ be a matrix of **key** vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_K
- Let $\mathbf{Q}$ be a matrix of **query** vectors
  - It has a sequence length of t_Q
  - It has a dimensionality of d_K
- Let $f(\mathbf{Q}, \mathbf{K})$ be some kernel function
  - Read: similarity function

$$\texttt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \underbrace{\texttt{softmax}\left(f\left(\mathbf{Q}, \mathbf{K}\right)\right)}_{\mathbf{A}}\mathbf{V}$$

$$\mathbf{A} \in (0,1)^{[t_Q \times t_V]}$$

$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_v]}$$

$$\mathbf{K} \in \mathbb{R}^{[t_V \times d_k]}$$

$$\mathbf{Q} \in \mathbb{R}^{[t_Q \times d_k]}$$

Definition & Properties

Multi-head Attention

- Let $\mathbf{V}$ be a matrix of **value** vectors
  - It has a sequence length of T_V
  - It has a dimensionality of d_V
- Let $\mathbf{K}$ be a matrix of **key** vectors
  - It has a sequence length of t_V
  - It has a dimensionality of d_K
- Let $\mathbf{Q}$ be a matrix of **query** vectors
  - It has a sequence length of t_Q
  - It has a dimensionality of d_K
- Let $f(\mathbf{Q}, \mathbf{K})$ be some kernel function
  - Read: similarity function

$$\texttt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \underbrace{\texttt{softmax}\left(f\left(\mathbf{Q}, \mathbf{K}\right)\right)}_{\mathbf{A}}\mathbf{V}$$

$$\mathbf{A} \in (0,1)^{[t_Q \times t_V]}$$

$$\mathbf{V} \in \mathbb{R}^{[t_V \times d_v]}$$

$$\mathbf{K} \in \mathbb{R}^{[t_V \times d_k]}$$

$$\mathbf{Q} \in \mathbb{R}^{[t_Q \times d_k]}$$

Non-Transformer Examples

Multi-head Attention

- $\mathbf{V}$ contains information
- $\mathbf{K}$ contains information about information (i.e, metadata)
- $\mathbf{Q}$ contains metadata about what we want from $\mathbf{V}$
- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$

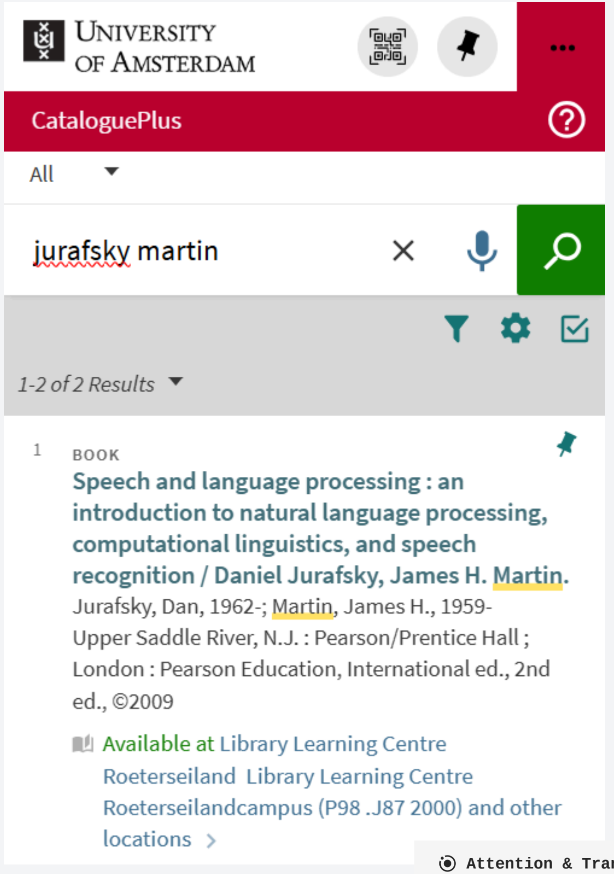
Non-Transformer Examples

Multi-head Attention

- $\mathbf{V}$ contains information
- $\mathbf{K}$ contains information about information (i.e, metadata)
- $\mathbf{Q}$ contains metadata about what we want from $\mathbf{V}$
- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$

E Soft lookup

We want to find a textbook about NLP in the library ($\mathbf{V}$). We search for titles ($\mathbf{K}$) with "jurafsky" and "martin" as authors ($\mathbf{Q}$). The computer returns books with similar titles (f)



Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$
- The output of f must a matrix of size $\mathbf{A} \in (0,1)^{[T_Q \times T_V]}$

Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$
- The output of f must a matrix of size $\mathbf{A} \in (0, 1)^{[T_Q \times T_V]}$

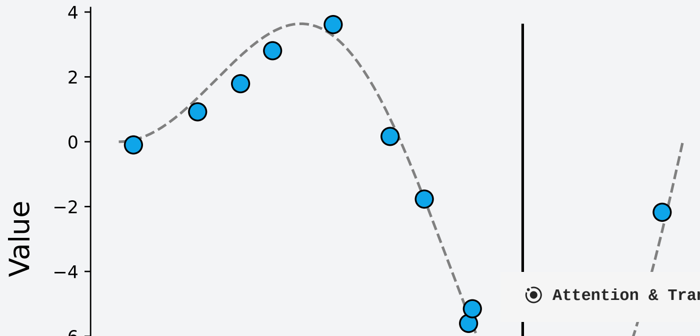
E Nadaraya-Watson Kernel Regression

We have some sequence of values
$\mathcal{D} = [(1.36, 1.79), (3.40, -1.77) \ldots, (6.05, -2.17)]$

We want to predict a new sample at $x = 4.25$

We compute the negative Euclidean distance of our new sample with all training samples (f). We normalize the outputs to lie between $(0, 1)$

We compute our predicted value as the mean of the seen values, weighted by the computed similarities



Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$
- The output of f must a matrix of size $\mathbf{A} \in (0,1)^{[T_Q \times T_V]}$

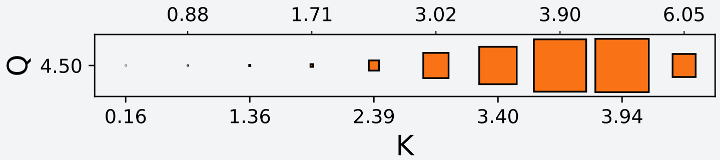
E Nadaraya-Watson Kernel Regression

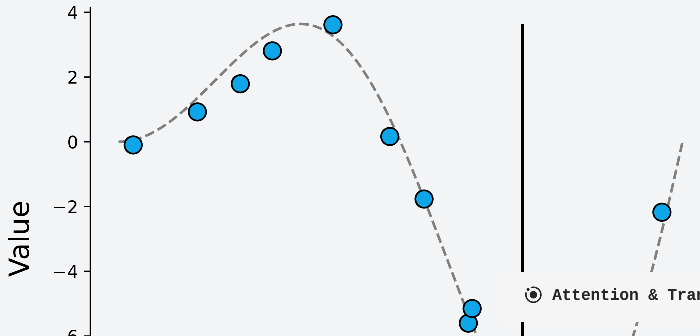
We have some sequence of values
$\mathcal{D} = [(1.36, 1.79), (3.40, -1.77) \ldots, (6.05, -2.17)]$

We want to predict a new sample at $x = 4.25$

We compute the negative Euclidean distance of our new sample with all training samples (f). We normalize the outputs to lie between $(0,1)$

We compute our predicted value as the mean of the seen values, weighted by the computed similarities





Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$
- The output of f must a matrix of size $\mathbf{A} \in (0,1)^{[T_Q \times T_V]}$

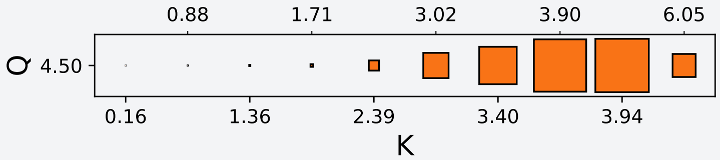
E **Nadaraya-Watson Kernel Regression**

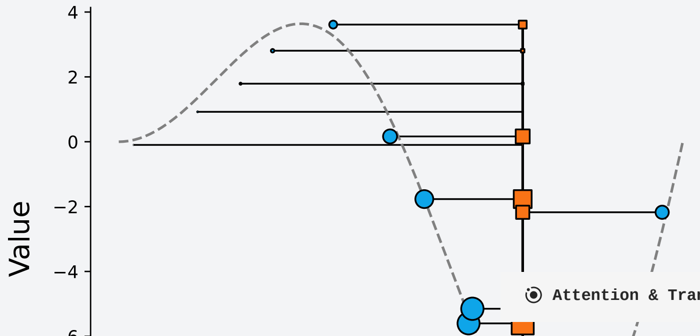
We have some sequence of values
$\mathcal{D} = [(1.36, 1.79), (3.40, -1.77)\ldots, (6.05, -2.17)]$

We want to predict a new sample at $x = 4.25$

We compute the negative Euclidean distance of our new sample with all training samples (f). We normalize the outputs to lie between $(0,1)$

We compute our predicted value as the mean of the seen values, weighted by the computed similarities





Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when $\mathbf{Q}$ is similar to $\mathbf{K}$
- The output of f must a matrix of size $\mathbf{A} \in (0,1)^{[T_Q \times T_V]}$

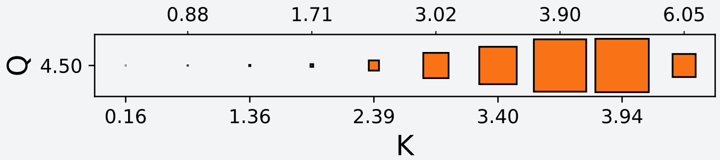
E Nadaraya-Watson Kernel Regression

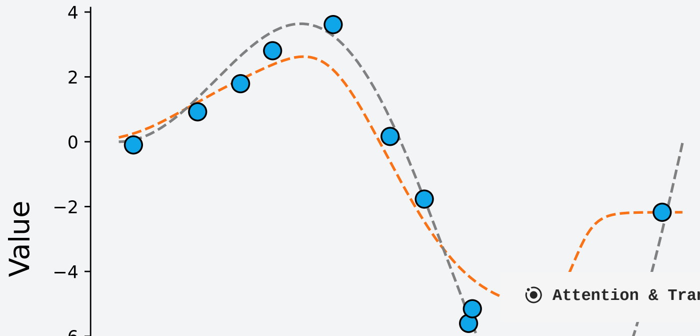
We have some sequence of values
$\mathcal{D} = [(1.36, 1.79), (3.40, -1.77) \ldots, (6.05, -2.17)]$

We want to predict a new sample at $x = 4.25$

We compute the negative Euclidean distance of our new sample with all training samples (f). We normalize the outputs to lie between $(0, 1)$

We compute our predicted value as the mean of the seen values, weighted by the computed similarities



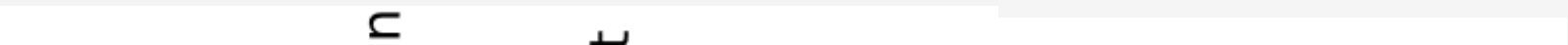


Non-Transformer Examples

Multi-head Attention

Non-Transformer Examples

Multi-head Attention



Attention in Transformers

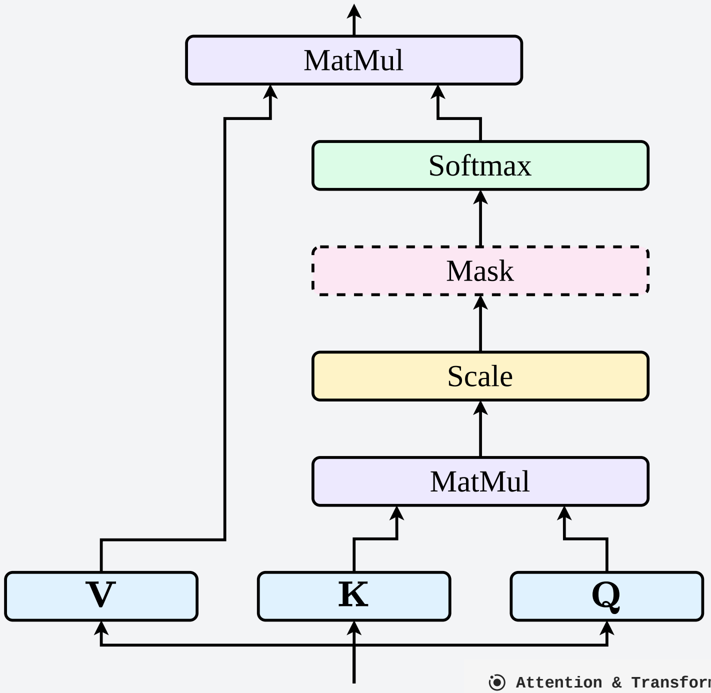
Multi-head Attention

- Transformer attention uses a scaled dot-product kernel function

$$f(\mathbf{Q}, \mathbf{K}) = \frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}$$

- $\mathbf{Q}$ is of size $t_Q \times d_K$
- $\mathbf{K}$ is of size $t_V \times d_K$
- Attention matrix is thus of size $t_Q \times t_V$

$$\mathtt{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathtt{softmax}\left(f(\mathbf{Q}, \mathbf{K})\right) \mathbf{V}$$



Attention in Transformers

Multi-head Attention

- Transformer attention uses a scaled dot-product kernel function

$$f(\mathbf{Q}, \mathbf{K}) = \frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}$$

- Why scale?
  - Assume the elements in $\mathbf{Q}$ and $\mathbf{K}$ come from *independent* normal distributions:

$$\mathbf{q}, \mathbf{k} \sim \mathcal{N}(0, 1)$$

  - The distribution of their dot-product is:

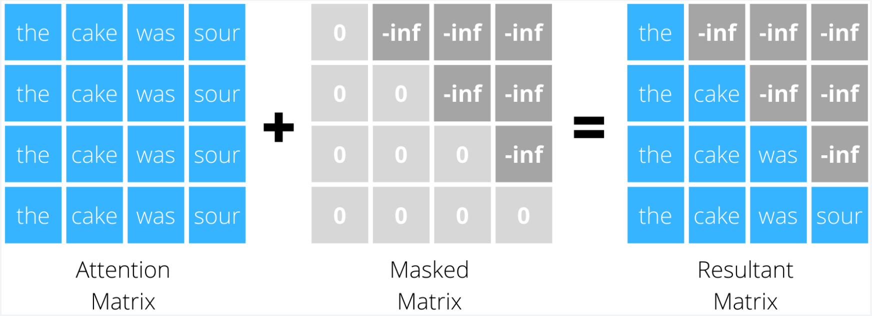
$$\mathbf{q}^\intercal\mathbf{k} \sim \mathcal{N}(0, \sqrt{d_k})$$

$$\begin{aligned}
\mathtt{var}\left[\mathbf{q}^\intercal\mathbf{k}\right] &= \mathtt{var}\left[\sum_i^{d_k} q_i k_i\right] \\
&= \sum_i^{d_k} \mathtt{var}\left[q_i k_i\right] \\
&= \sum_i^{d_k} \mathtt{var}\left[q_i\right] \mathtt{var}\left[k_i\right] \\
&= \sum_i^{d_k} 1 \cdot 1 \\
&= d_k
\end{aligned}$$

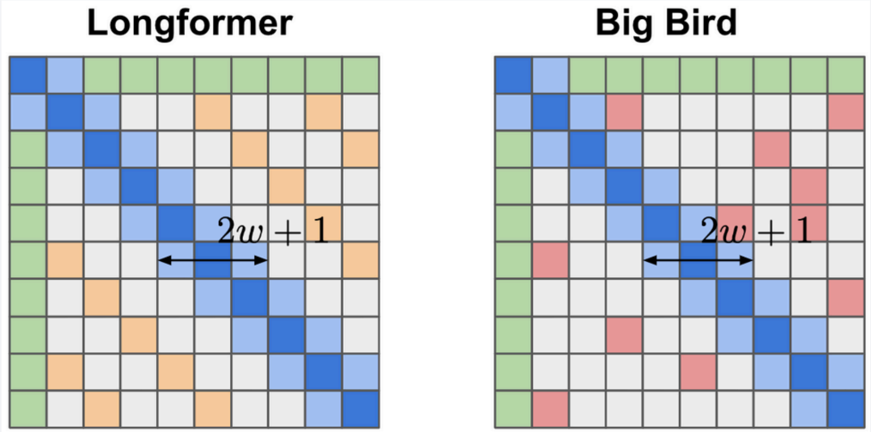
Attention in Transformers

Multi-head Attention

- Why mask?
  - Currently all tokens are treated equally
  - **Causal masking**: decoder tokens should never attend to future tokens, only to the past
  - **Local masking**: sometimes local attention is all you need



https://krypticmouse.hashnode.dev/attention-is-all-you-need

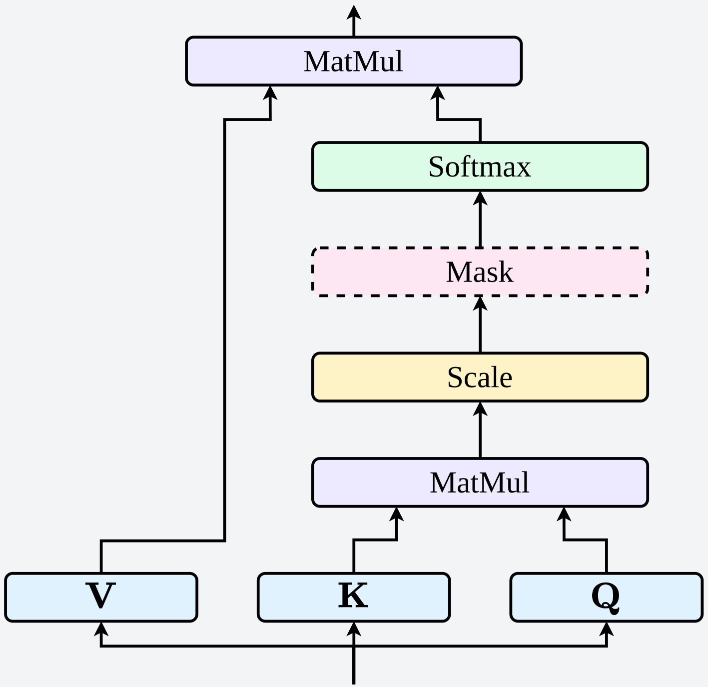


https://lilianweng.github.io/posts/2023-01-27-the-transformer-family-v2/

Attention in Transformers

Multi-head Attention

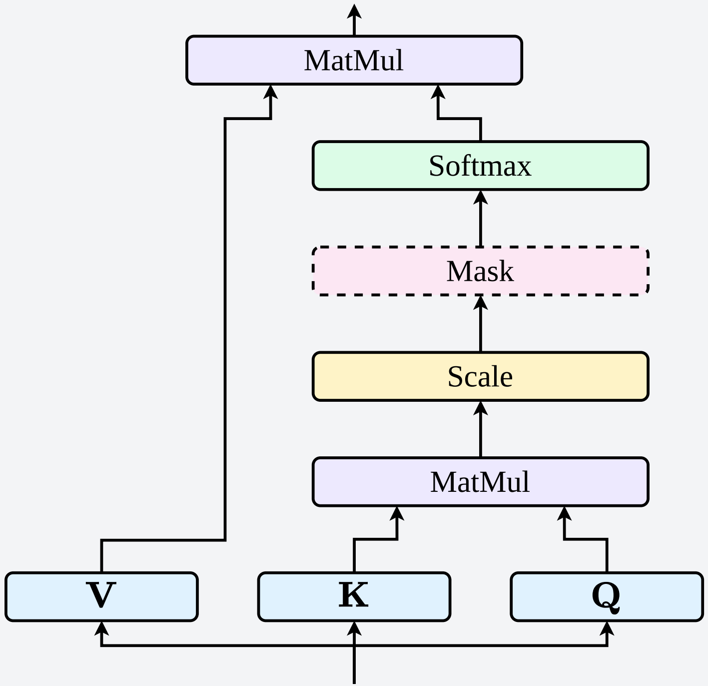
- Where do $\mathbf{V}$, $\mathbf{K}$, $\mathbf{Q}$ come from?



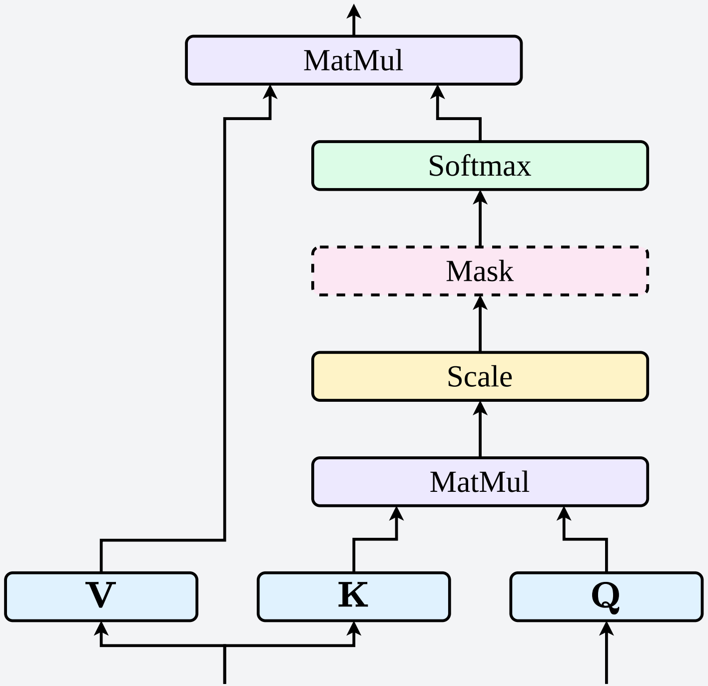
Attention in Transformers

Multi-head Attention

Self-attention



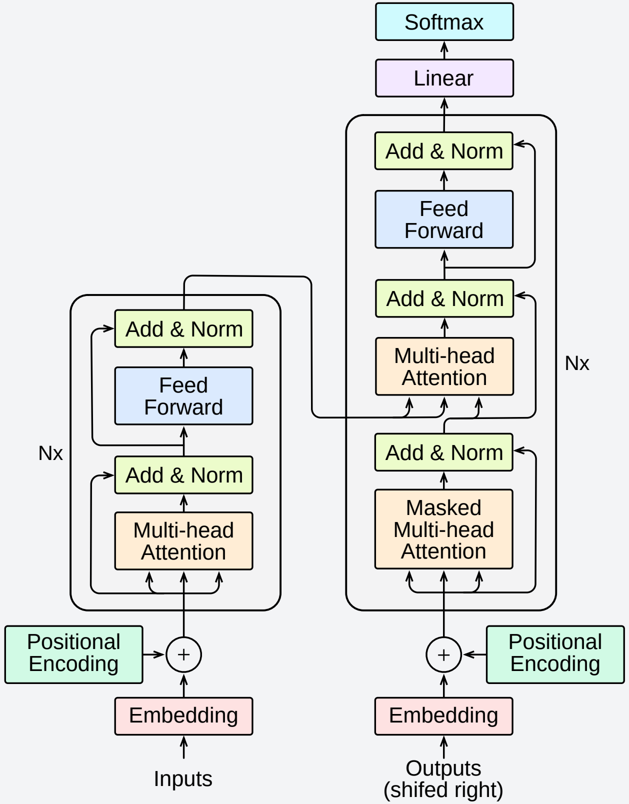
Cross-attention



Attention in Transformers

Multi-head Attention

- Where do $\mathbf{V}$, $\mathbf{K}$, $\mathbf{Q}$ come from?
  - **Self-attention**: everything comes from the same sequence
  - **Cross-attention**: $\mathbf{V}$, $\mathbf{K}$ come from source sequence, $\mathbf{Q}$ comes from target sequence
  - All components constructed from a projection of the token embeddings
    1. $\mathbf{V} = \mathbf{X}\mathbf{W}_V$
    2. $\mathbf{K} = \mathbf{X}\mathbf{W}_K$
    3. $\underbrace{\mathbf{Q} = \mathbf{X}\mathbf{W}_Q}_{\text{Self-attention}}$ or $\underbrace{\mathbf{Q} = \mathbf{Y}\mathbf{W}_Q}_{\text{Cross-attention}}$



Attention in Transformers

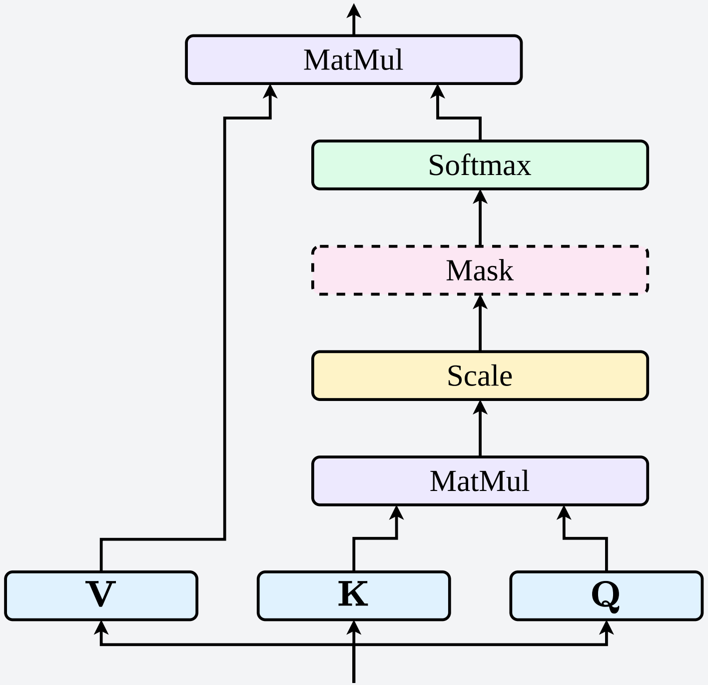
Multi-head Attention

- Even in self-attention, attention matrix is **not** symmetric

$$\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}} = \frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\intercal}{\sqrt{d_k}} = \frac{\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\intercal\mathbf{X}^\intercal}{\sqrt{d_k}}$$

Asymmetry

The contribution of token $\mathbf{x}_i$ to $\mathbf{x}_j$, is **not** the same as the contribution of tokn $\mathbf{x}_j$ to $\mathbf{x}_i$



Attention in Transformers

Multi-head Attention

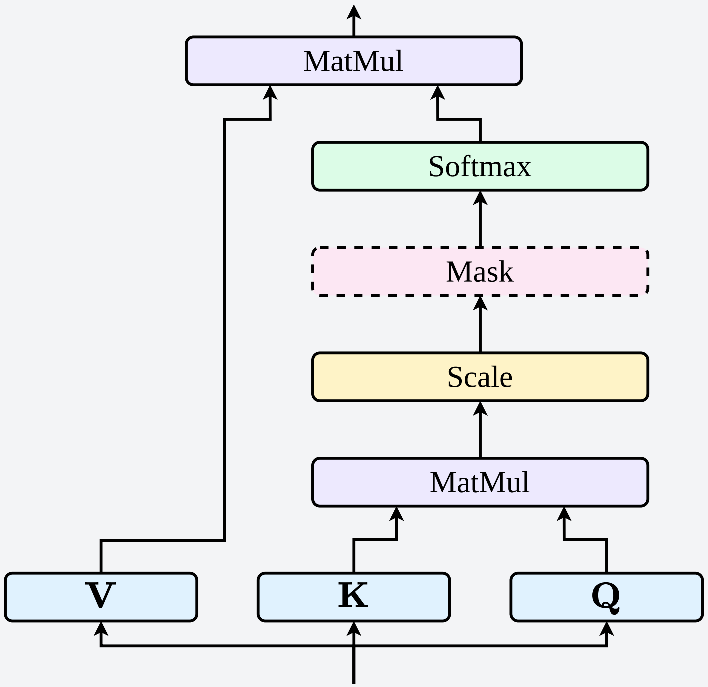
- Transformer attention between two sequences, $\mathbf{X}$ and $\mathbf{Y}$ has a computational cost of (excluding projections):

$$\mathcal{O}\left(\underbrace{t_x \cdot t_y \cdot d_k}_{\text{MatMul 1}} + \underbrace{t_x \cdot t_y \cdot d_v}_{\text{MatMul 2}}\right)$$

- But RNNs have linear time complexity...

$$\mathcal{O}\left(t_x \cdot d_k^2 + t_x \cdot d_q^2\right)$$

- RNNs are serial, Attention is parallel
  - GPUs *looove* parallelism



Jakob Uszkoreit (August 31, 2017). Transformer: A Novel Neural Network Architecture for Language Understanding. https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/

Multi-head Attention

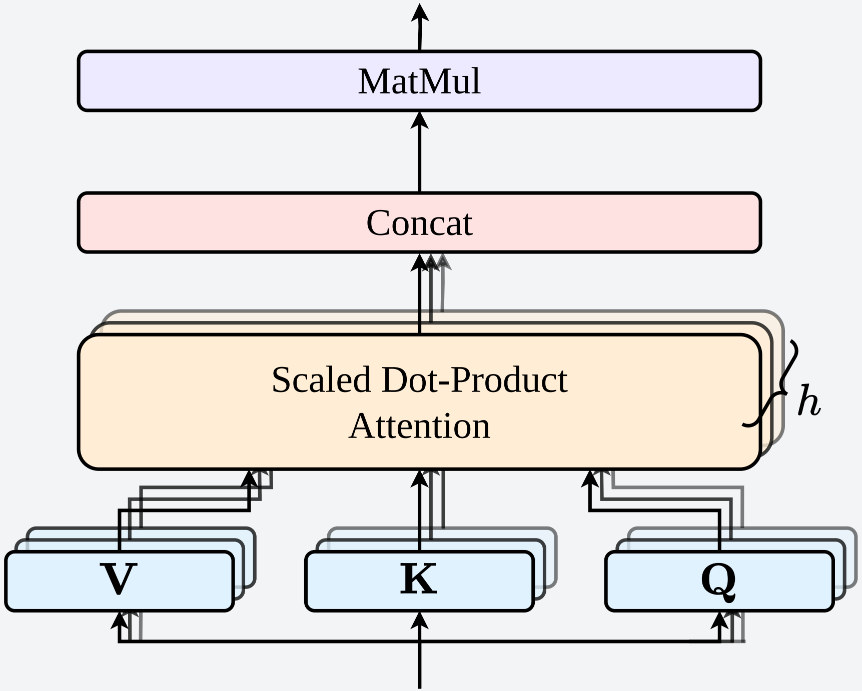
Multi-head Attention

- Currently we use 1 set of attention weights
  - Can only process 1 query type

Multi-head Attention

Multi-head Attention

- Currently we use 1 set of attention weights
  - Can only process 1 query type
- With h attention heads, we learn h concepts
  - To reduce cost, reduce dimensionality $d_{K,V}/h$



Multi-head Attention

Multi-head Attention

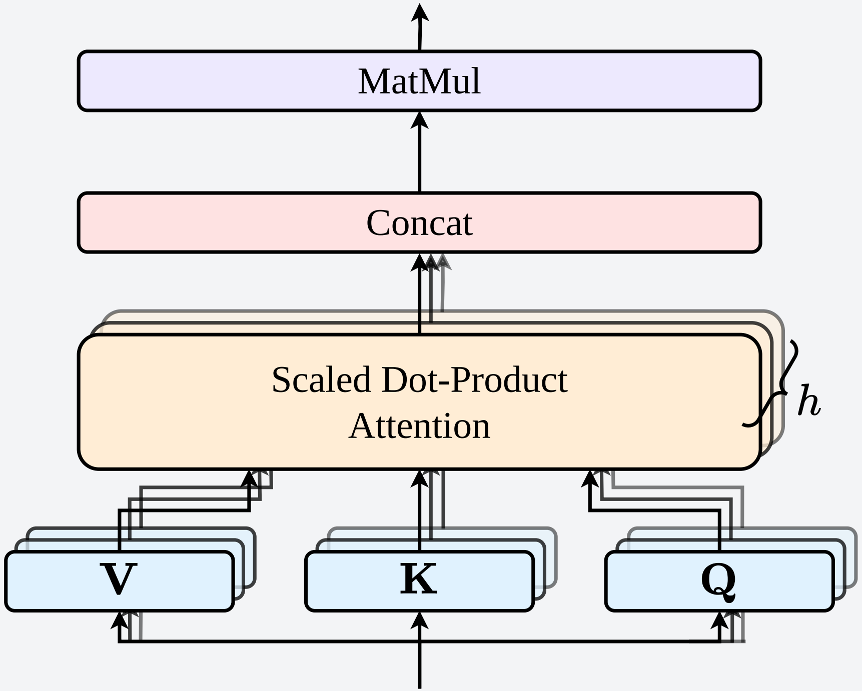
- Currently we use 1 set of attention weights
  - Can only process 1 query type
- With h attention heads, we learn h concepts
  - To reduce cost, reduce dimensionality $d_{K,V}/h$

```
self.attention_heads = [
  AttentionHead(d=self.d // self.h) for i in range(self
  ]

self.mha_proj = nn.Linear(self.d, self.d)

mha = torch.concat([
  attention_heads[i](x) for i in range(self.h)
  ])

out = self.mha_proj(mha)
```

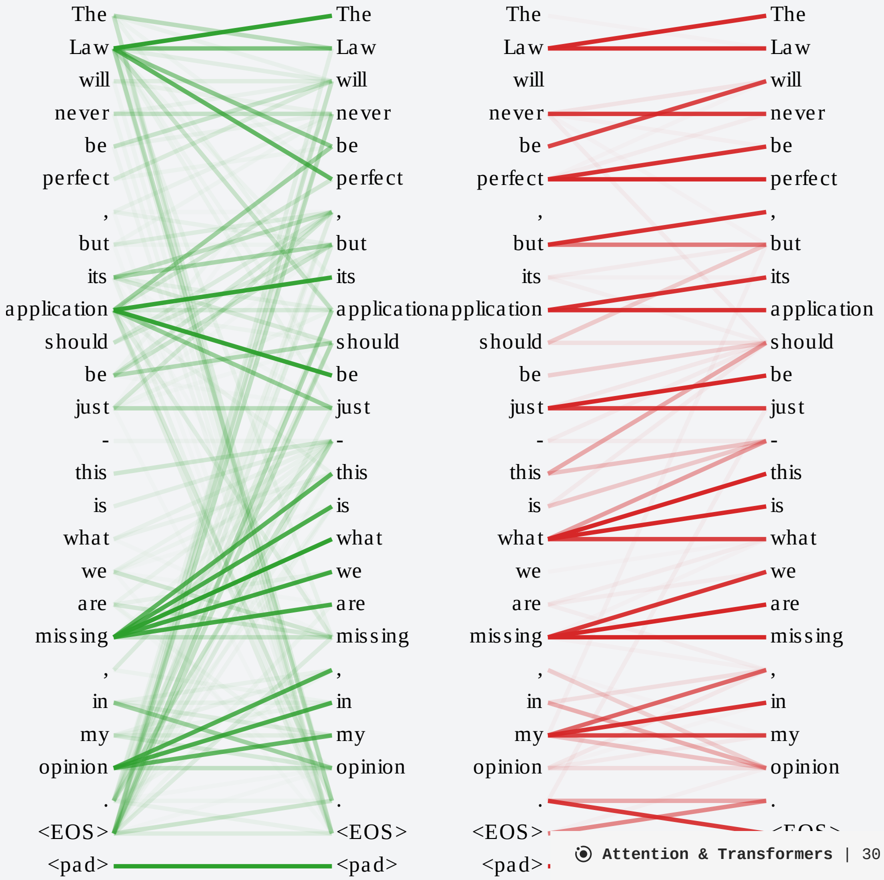


Multi-head Attention

Multi-head Attention

> Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. [One] attention head, averaging inhibits this.

Vaswani et al. (2017). Attention is all you need. Advar
in neural information processing systems, 30. (p. 5 & 1



Multi-head Attention

Multi-head Attention

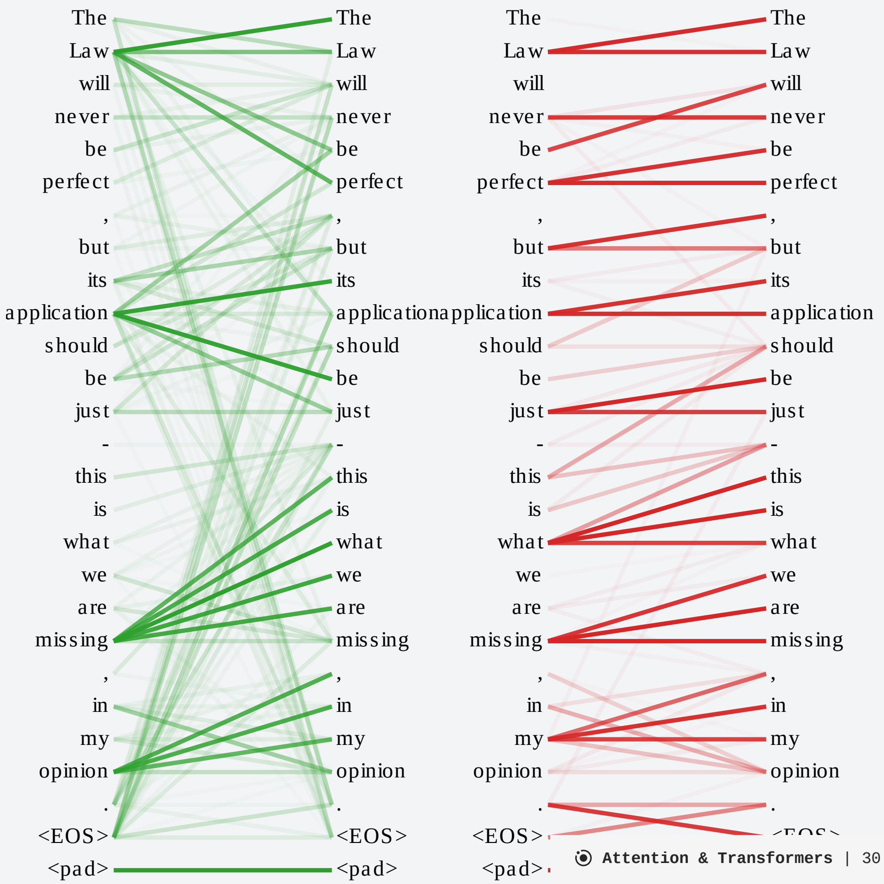
> Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. [One] attention head, averaging inhibits this.

Vaswani et al. (2017). Attention is all you need. Advan
in neural information processing systems, 30. (p. 5 & 1

Multiple heads, multiple different queries processed in parallel

- Positional heads
- Syntactic heads
- Rare words?

Voita et al. (2019). Analyzing Multi-Head Self-Attentic
Specialized Heads Do the Heavy Lifting, the Rest Can
Be Pruned. Association for Computational Linguistics.



Multi-head Attention

Multi-head Attention

Do different heads attend to different concepts?

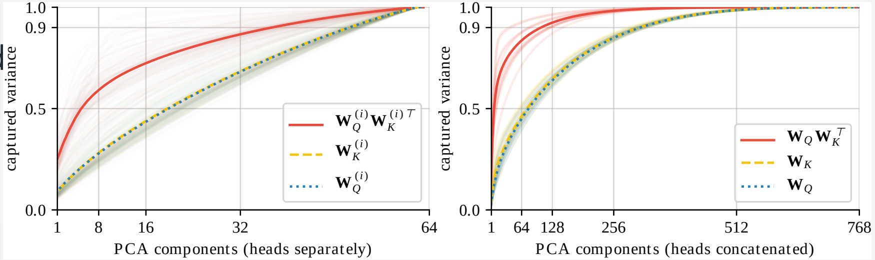
Multi-head Attention

Multi-head Attention

Do different heads attend to different concepts?

- Individual heads = high rank, concantenated head = low rank

  Cordonnier, Loukas & Jaggi (2020). Multi-head attention: Collaborate instead of concatenate. arXiv:2006.16362.



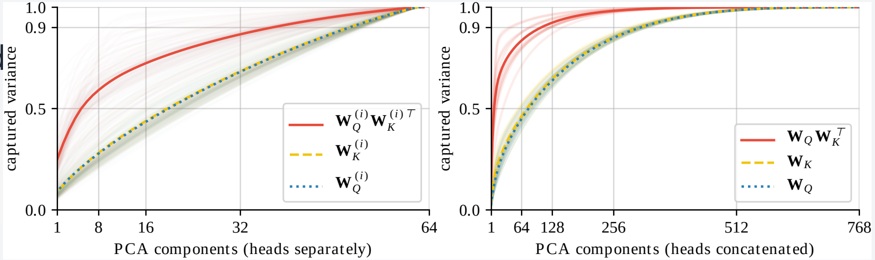
Multi-head Attention

Multi-head Attention

Do different heads attend to different concepts?

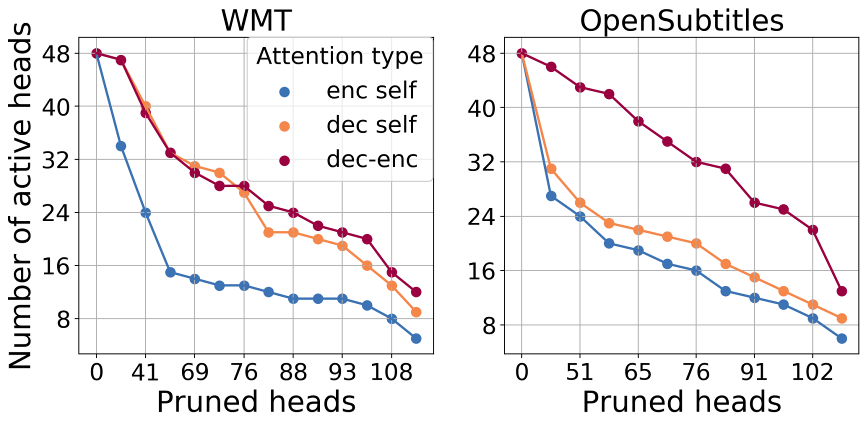
- Individual heads = high rank, concantenated heads = low rank

  Cordonnier, Loukas & Jaggi (2020). Multi-head attention: Collaborate instead of concatenate. arXiv:2006.16362.

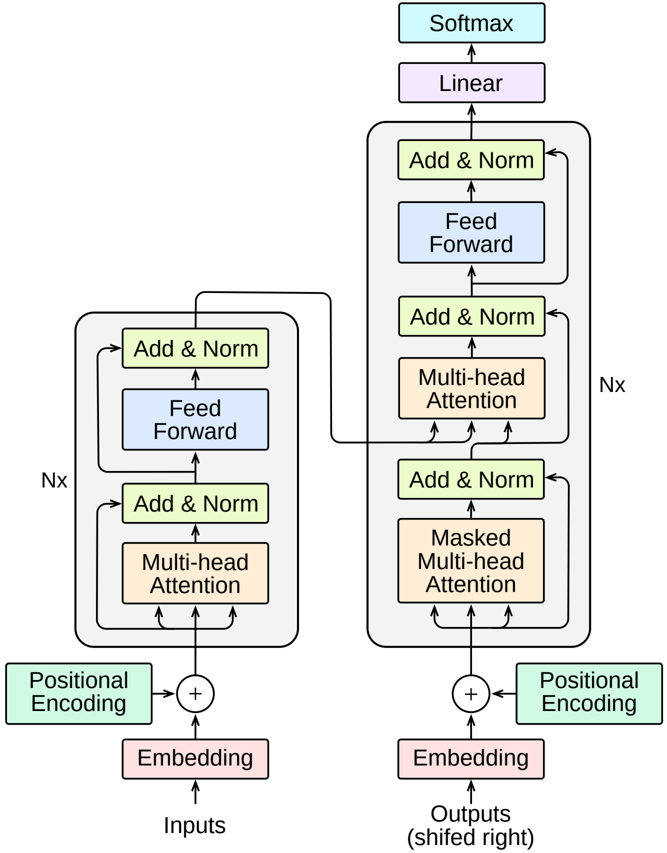


- Most heads can be pruned away
- Enc-Dec heads are more important than Enc-Enc heads

  Voita et al. (2019). Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. Association for Computational Linguistics.



Add & Norm


Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-head Attention
Nx
Add & Norm
Masked Multi-head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-head Attention
Nx
Positional Encoding
Positional Encoding
Embedding
Embedding
Inputs
Outputs (shifed right)

Add & Norm

Residual Connections

Add & Norm

LayerNorm

Add & Norm

These are the equations

$$
\begin{aligned}
\mathbf{X}_l \quad = \ & \texttt{LayerNorm}(\\
& \quad \mathbf{X}_{l-1} + \texttt{SubLayer}\left(\mathbf{X}_{l-1}\right) \\
&)
\end{aligned}
$$

Feed Forward

Embedding

Position Encoding

Tokenization

Training Transformers

The End