Attention & Transformers

Ivo Verhoeven | Advanced Topics in Computational Semantics

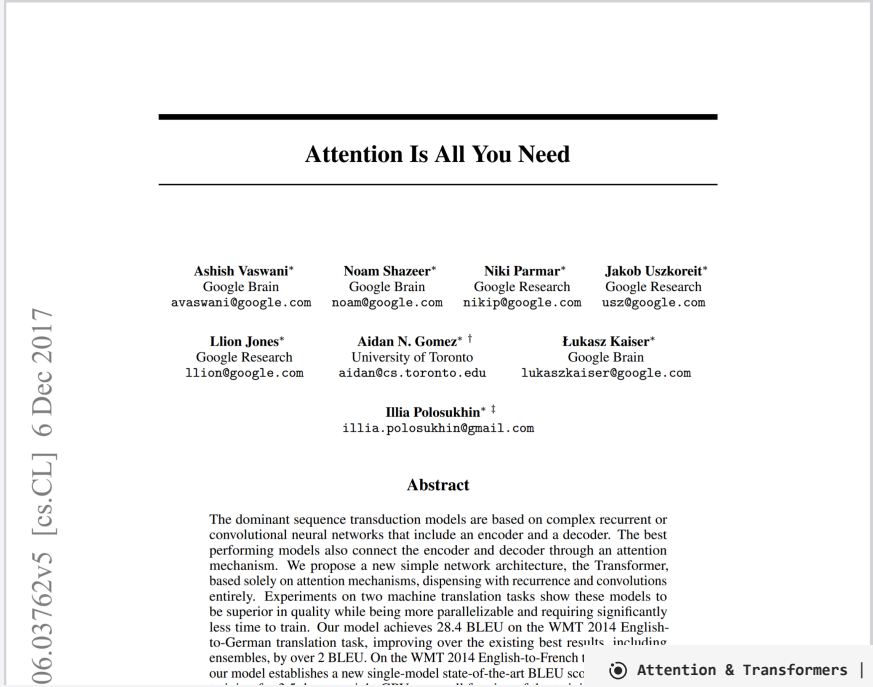
About Me

- 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 in 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

Abstract

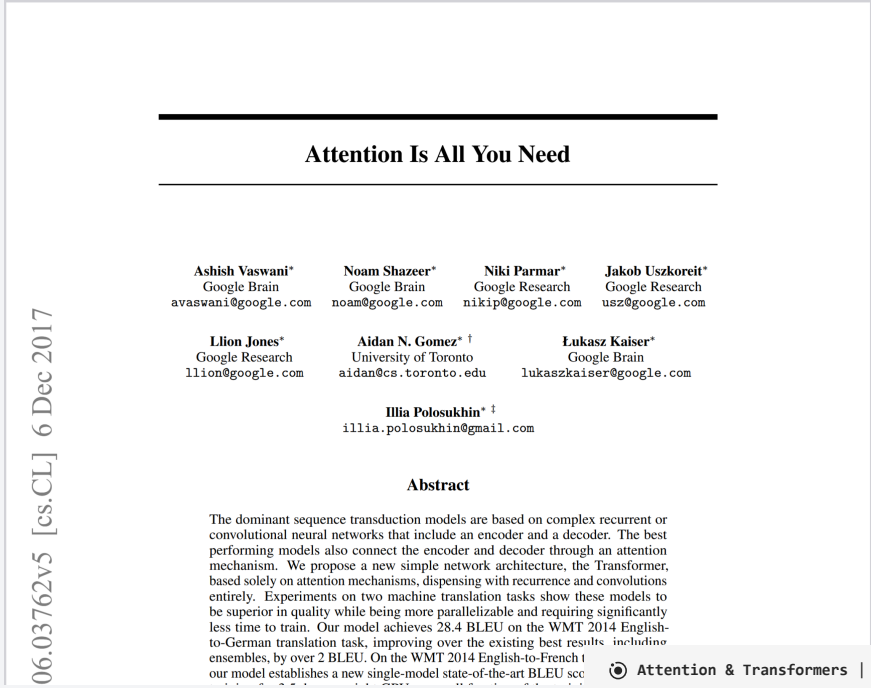
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,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

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 in 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

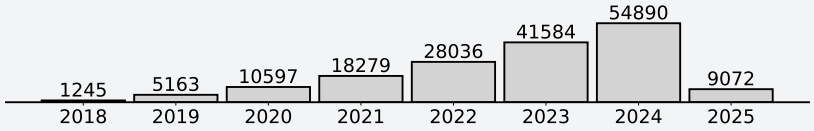
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,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

Vaswani et al.: Attention is All You Need

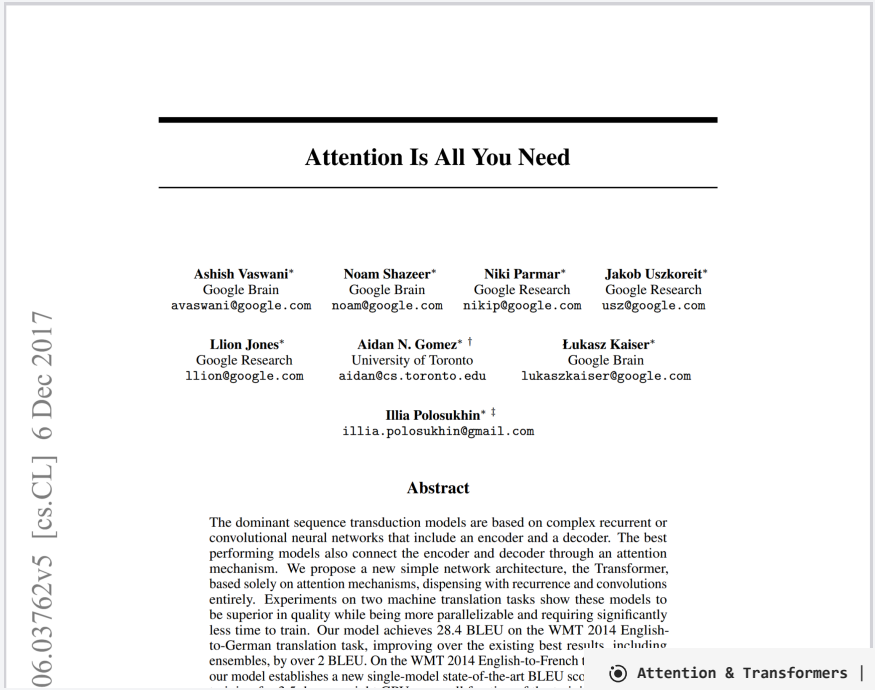
- Introduces the Transformer architecture in late 2017
  - Google Brain/Google Research collab

- 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 in 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

Abstract

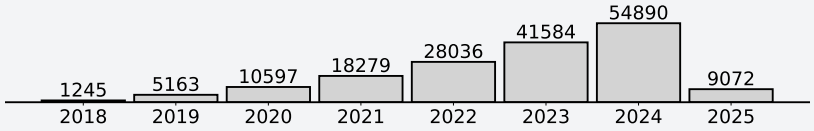
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,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

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**

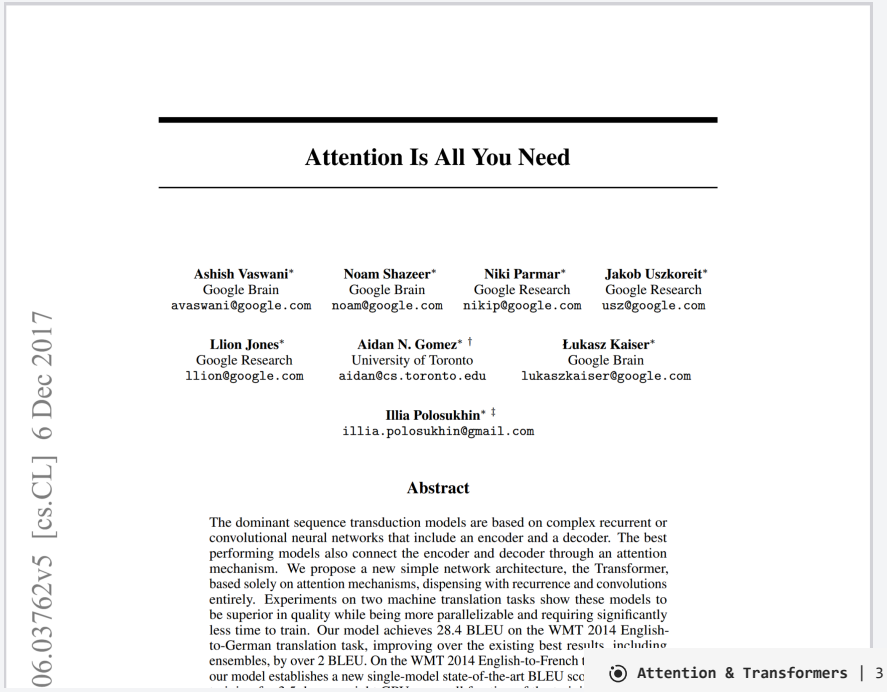
- Number of citations is only accelerating



- 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 in neural information processing systems, 30.



Attention Is All You Need

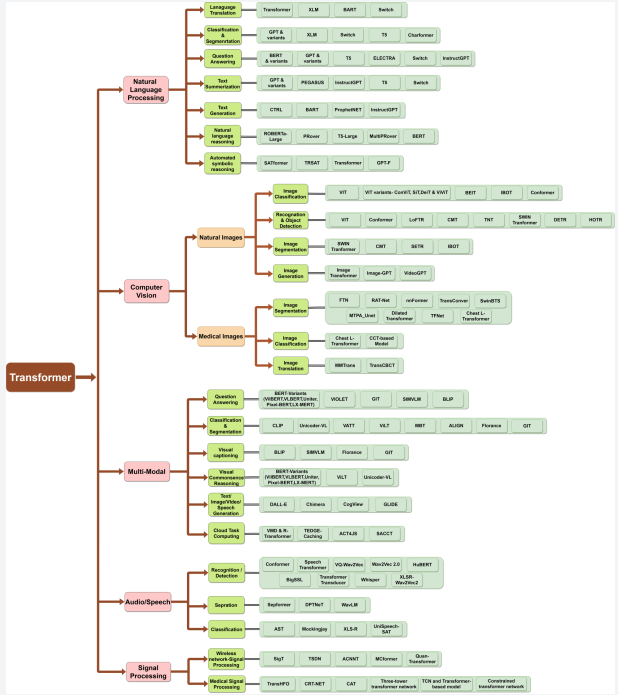
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

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

Vaswani et al.: Attention is All You Need

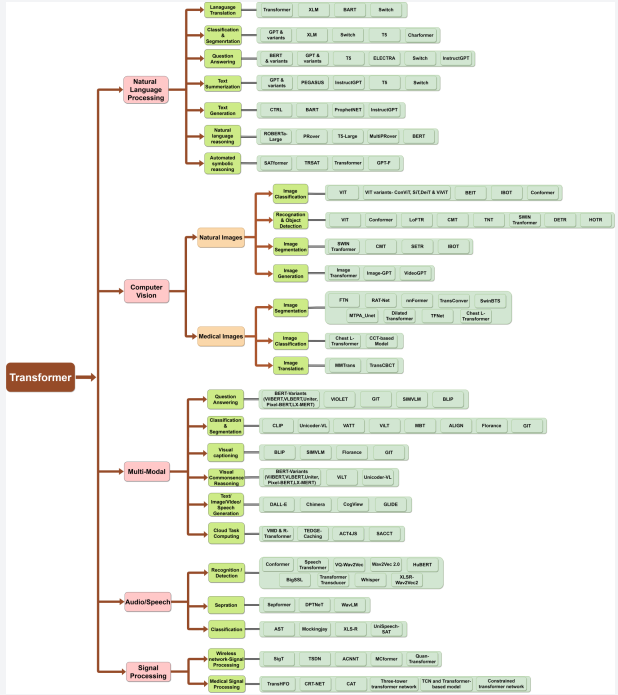
- It’s hard to think of an AI area that hasn’t been affected by the Transformer



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

Vaswani et al.: Attention is All You Need

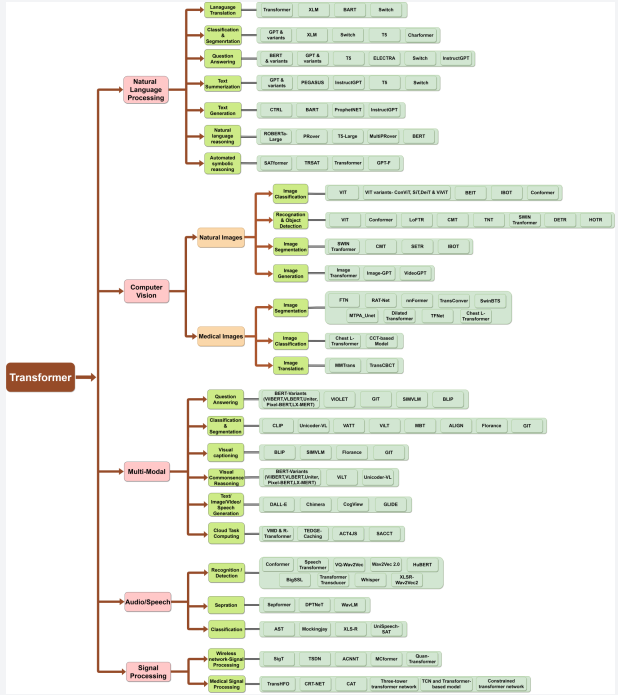
- It’s hard to think of an AI area that hasn’t been affected by the Transformer
  - **NLP:** Transformer > RNN
    - Seq-to-seq: what it was designed for
    - Classification: encoder-only transformers
    - Generation: decoder-only transformers



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

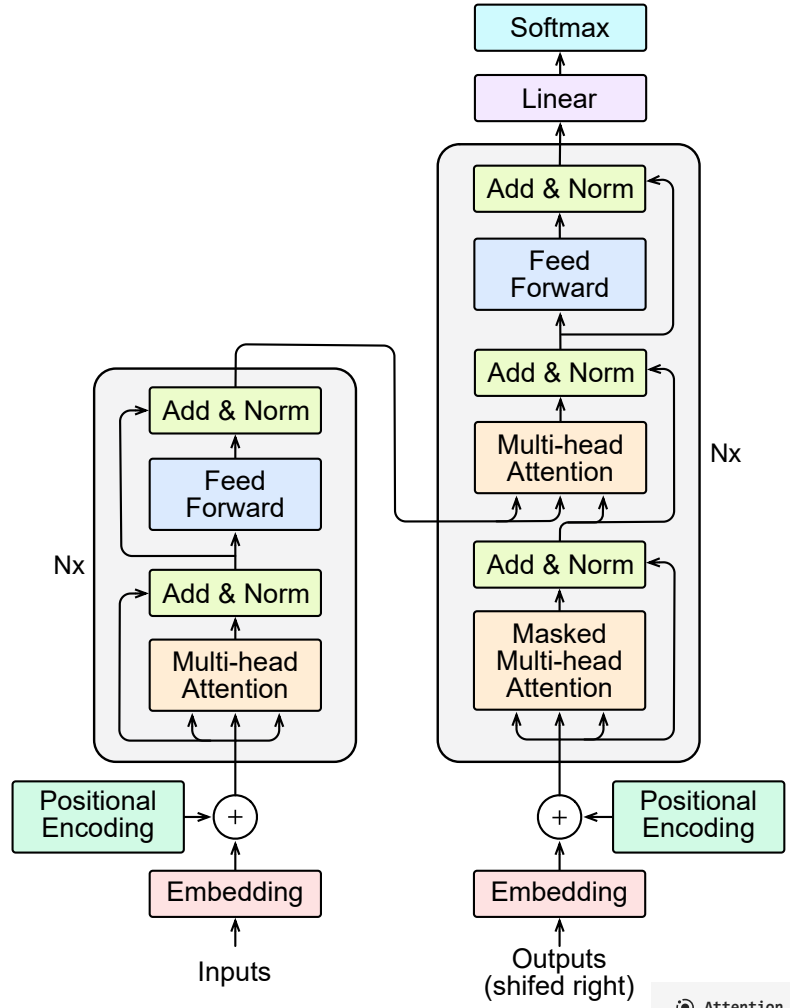
Vaswani et al.: Attention is All You Need

- 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

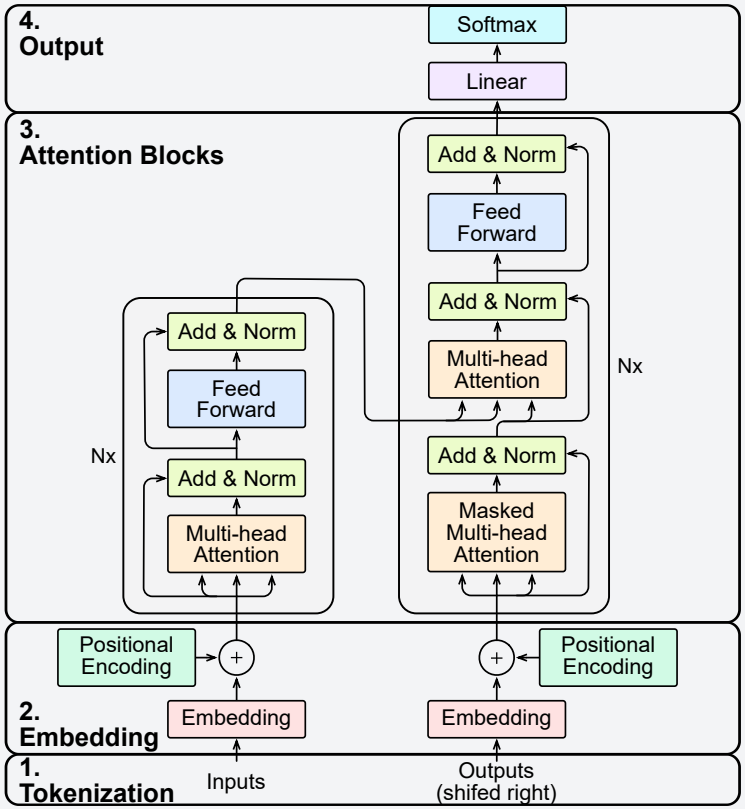


Islam, et al. (2023). A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. arXiv:2306.07303.

The Transformer

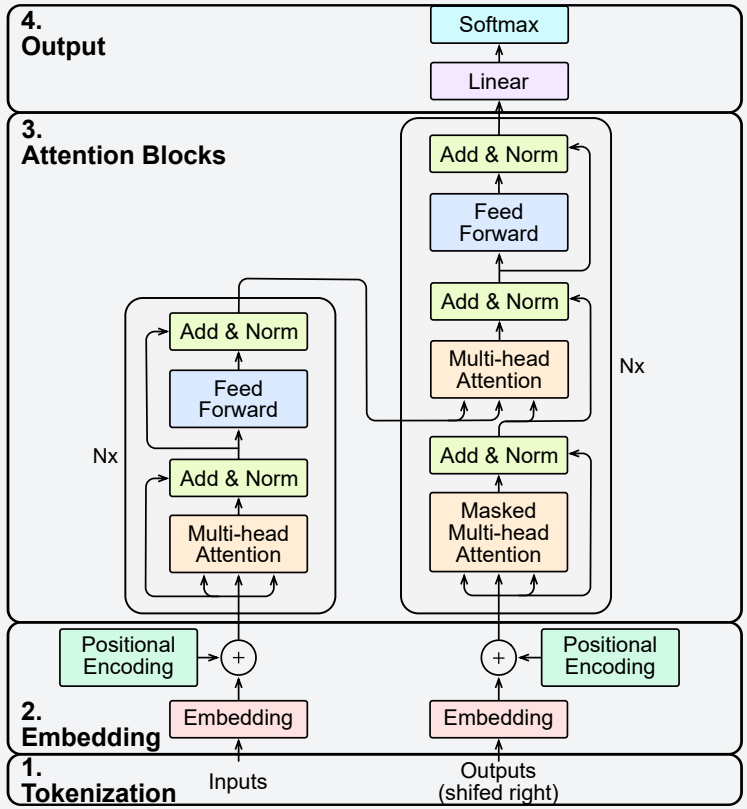


Breaking the Transformer into modules



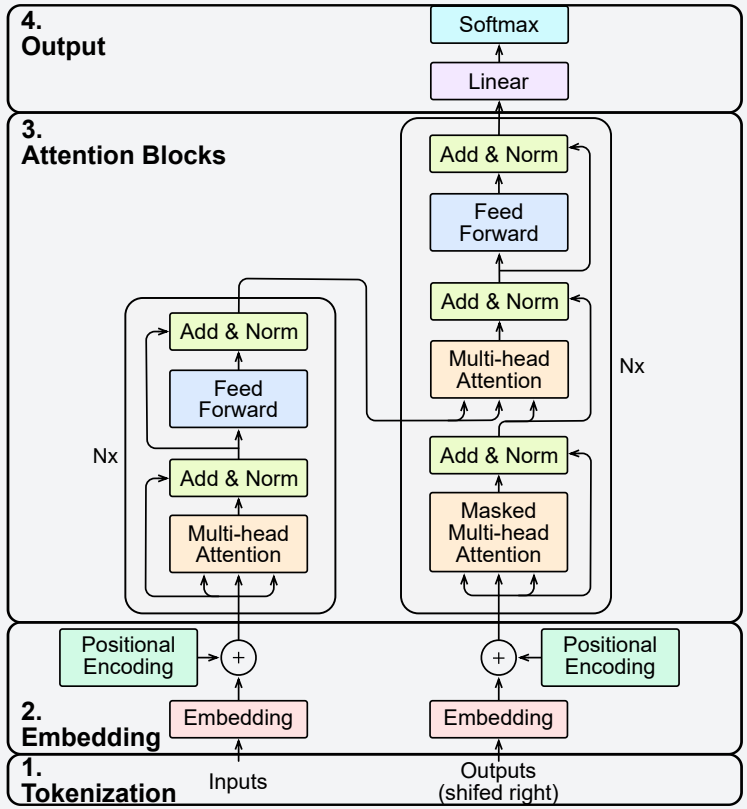
Breaking the Transformer into modules

4. Output
   - Softmax
   - Linear



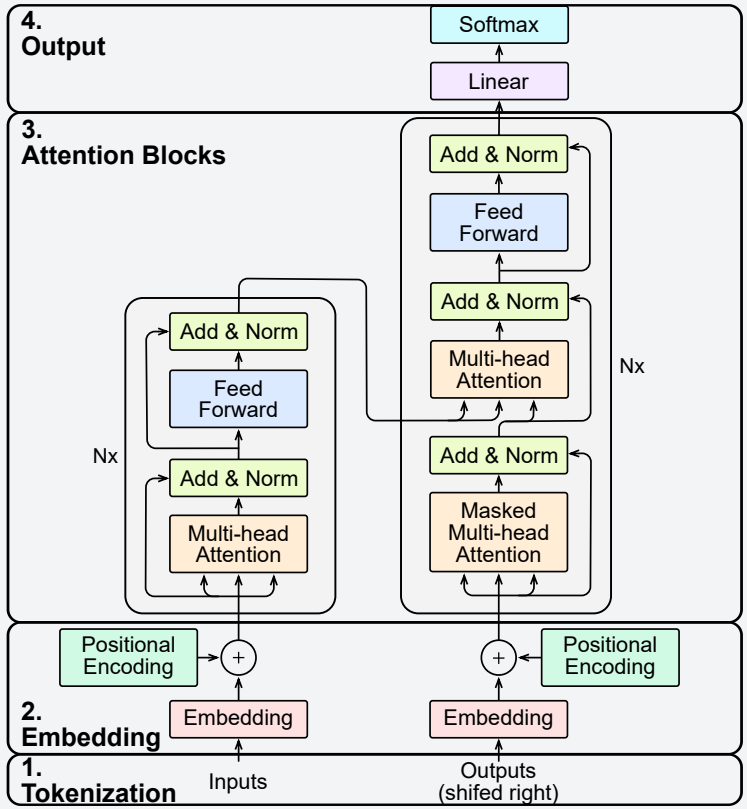
Breaking the Transformer into modules

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



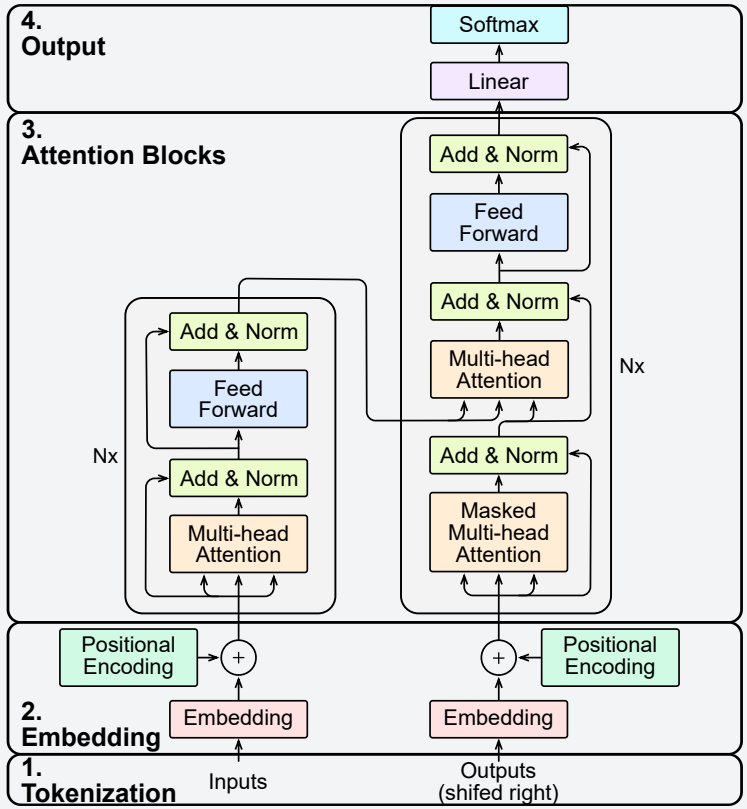
Breaking the Transformer into modules

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)

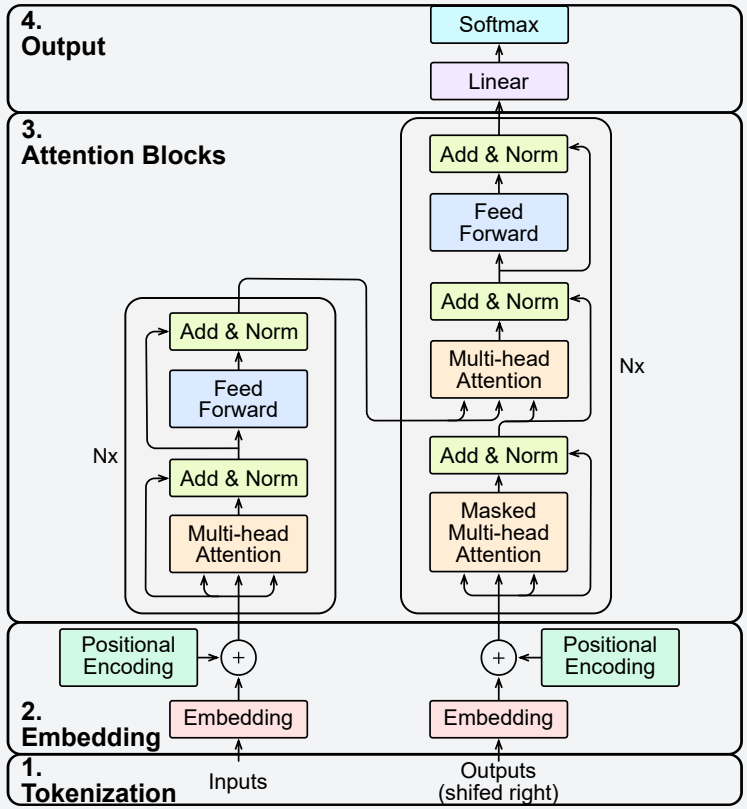
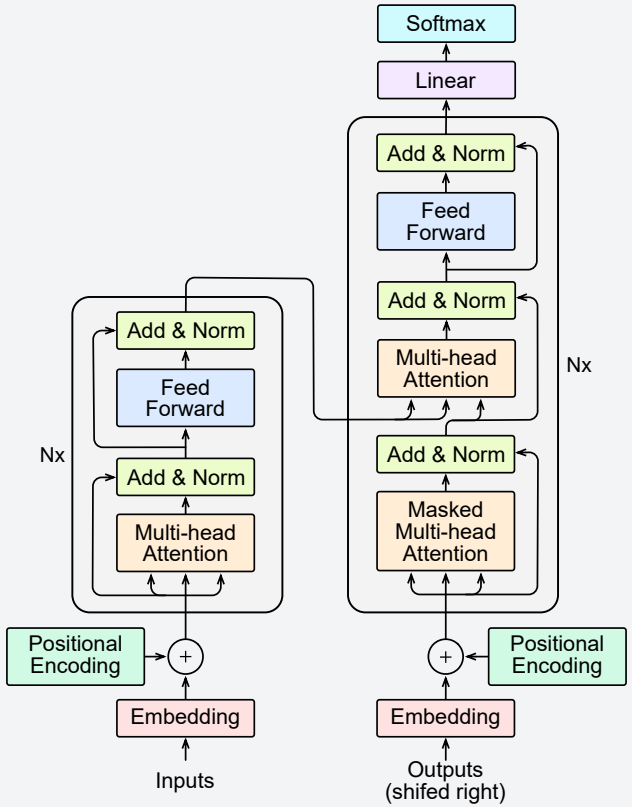
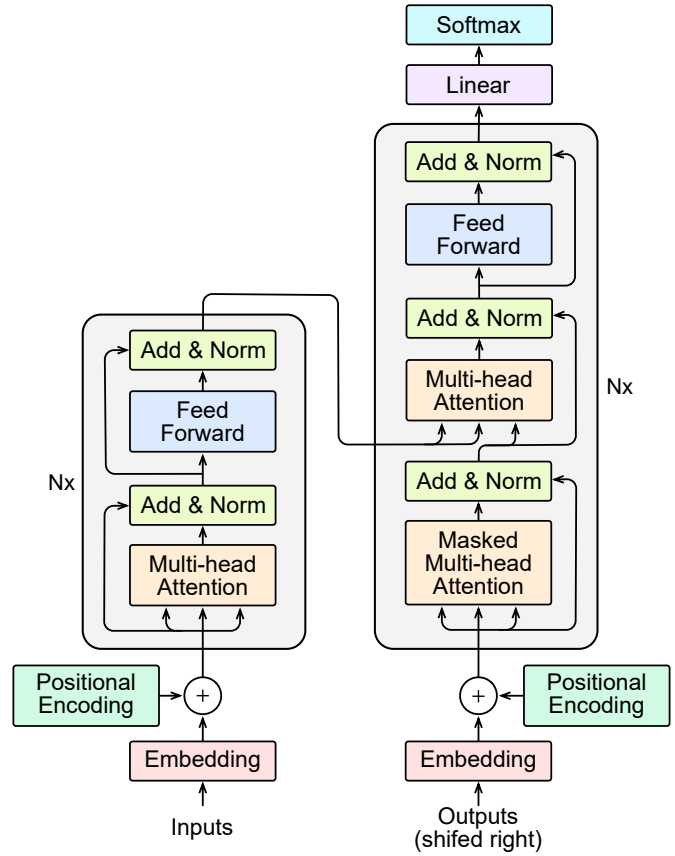


Table of Contents



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

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

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

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

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

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

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

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

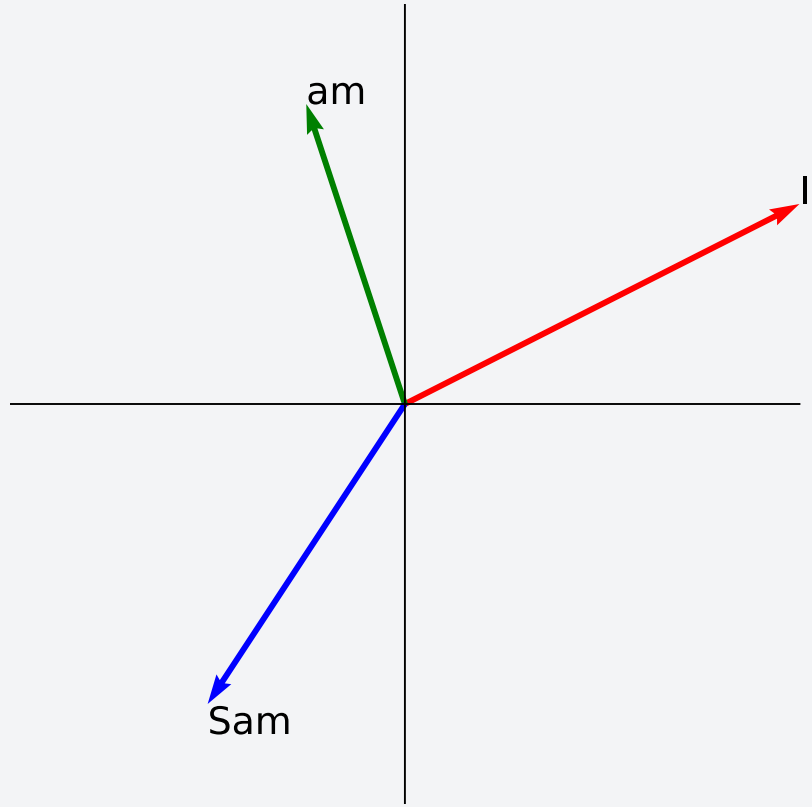
- $\texttt{Attention}$ is just a matrix product of $\mathbf{V}$ with an attention matrix $\mathbf{A}$
  - $\mathbf{A}$ is a square matrix of size $t_V \times t_V$
  - It's elements are all between $(0, 1)$
  - It's rows sum to 1

Definition & Properties

Multi-head Attention

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

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



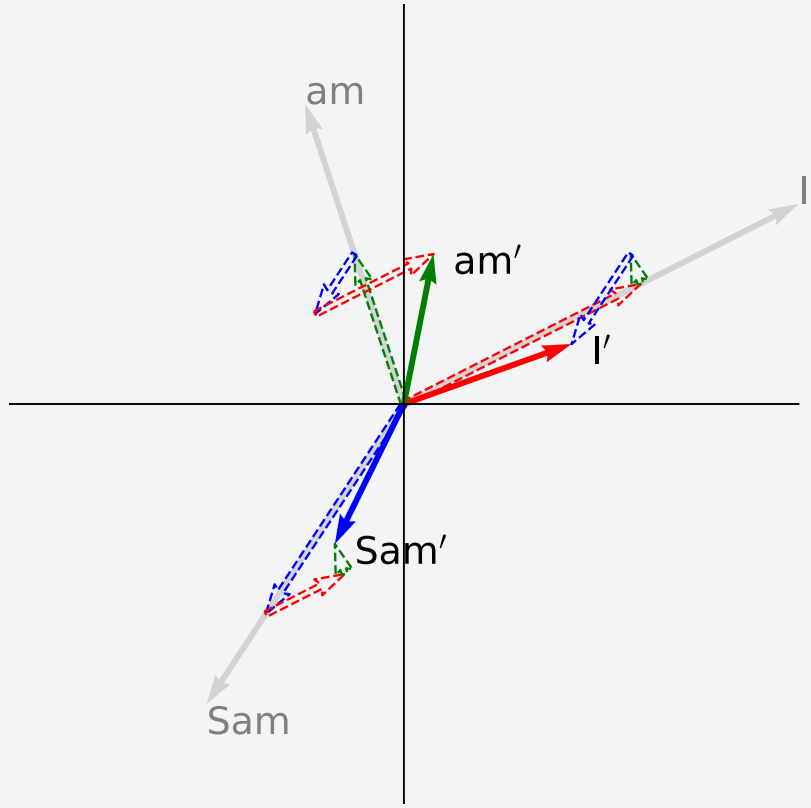
Definition & Properties

Multi-head Attention

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

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

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

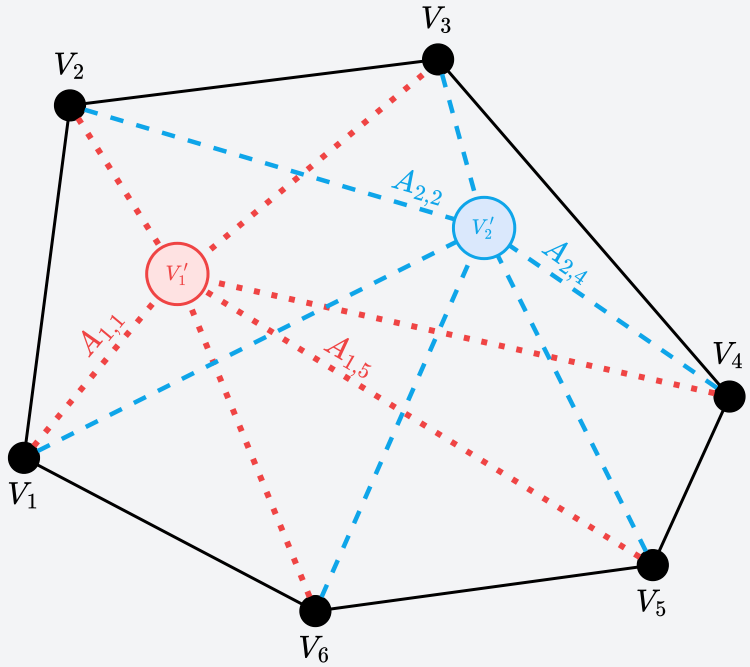


Definition & Properties

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

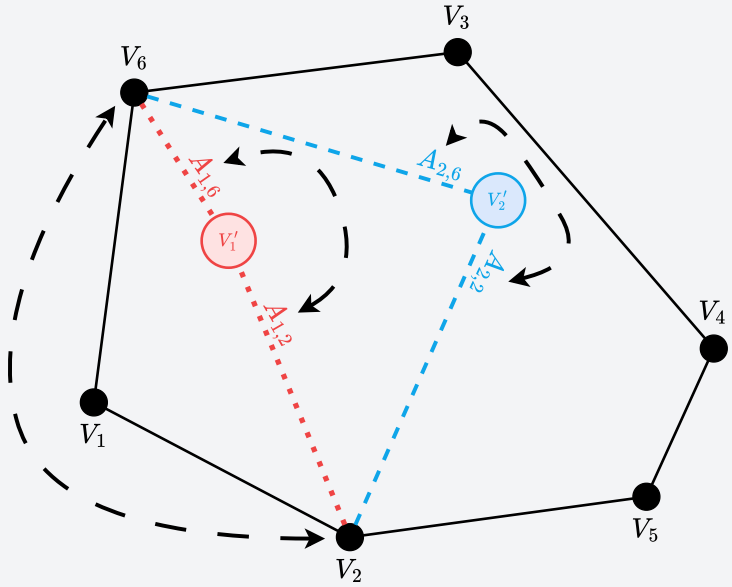
Convex Combination

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

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

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

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

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

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

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

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

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

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

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

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

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 $\mathbf{K}$

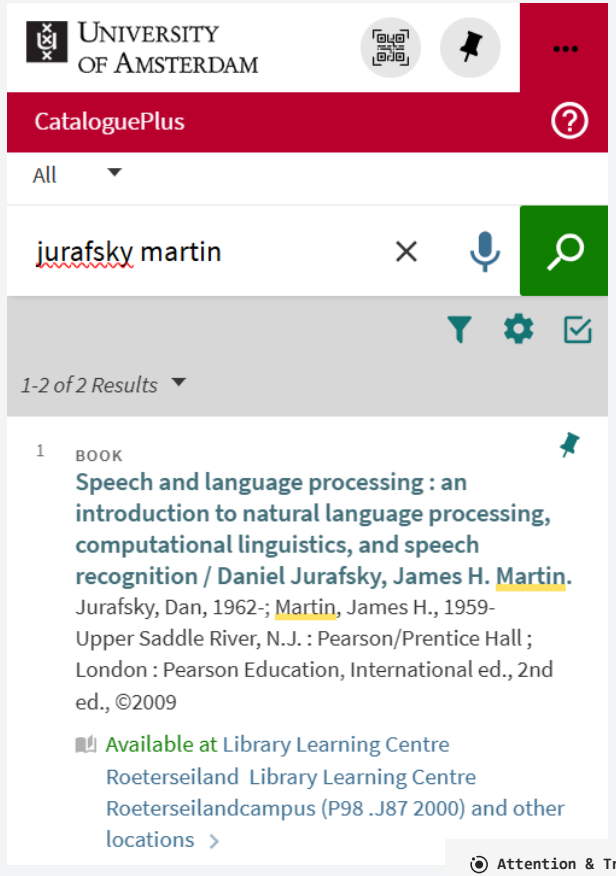
Non-Transformer Examples

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

- $\mathbf{Q}$ and $\mathbf{V}$ do not need to have the same sequence length
- The output of f is *always* a matrix of size $\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$

Non-Transformer Examples

Multi-head Attention

- $\mathbf{Q}$ and $\mathbf{V}$ do not need to have the same sequence length
- The output of f is *always* a matrix of size $\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$

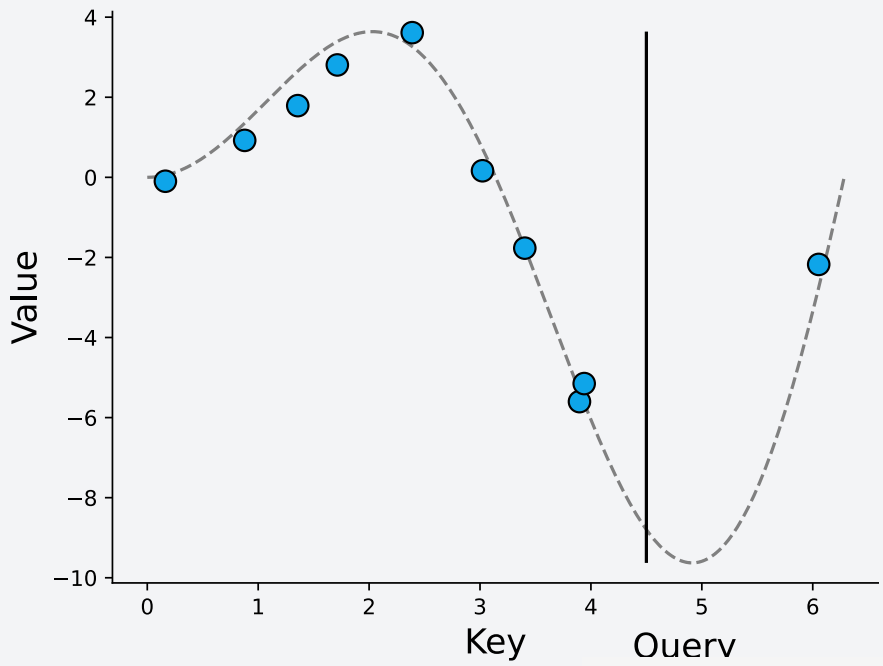
Nadaraya-Watson Kernel Regression

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

- $\mathbf{Q}$ and $\mathbf{V}$ do not need to have the same sequence length
- The output of f is *always* a matrix of size $\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$

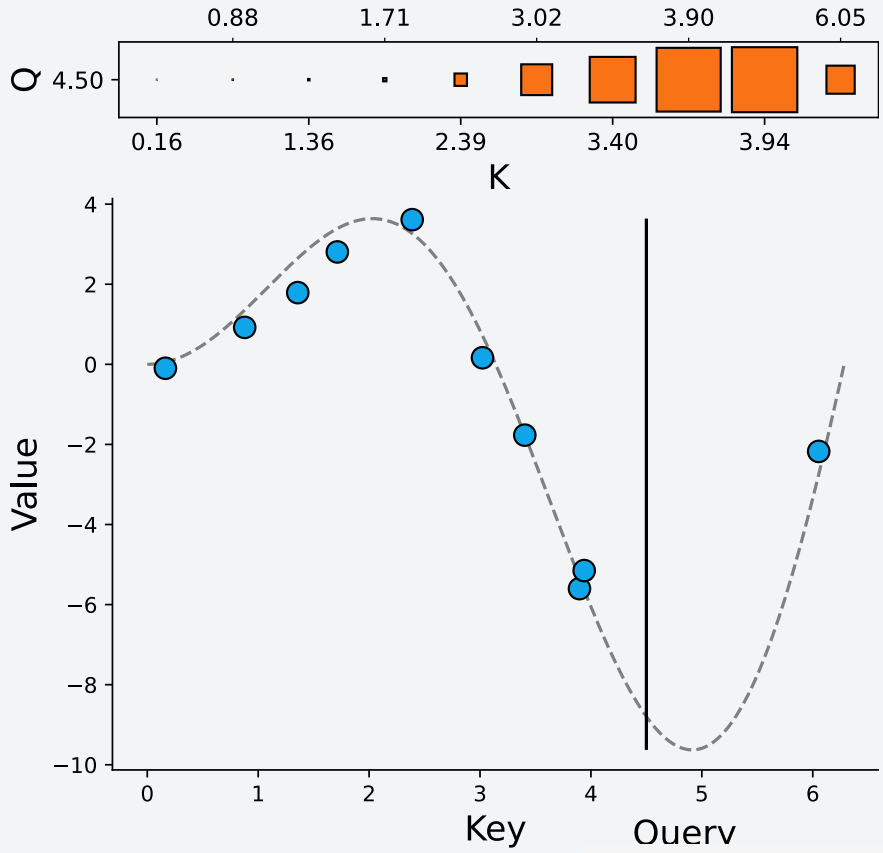
E **Nadaraya-Watson Kernel Regression**

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

- $\mathbf{Q}$ and $\mathbf{V}$ do not need to have the same sequence length
- The output of f is *always* a matrix of size $\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$

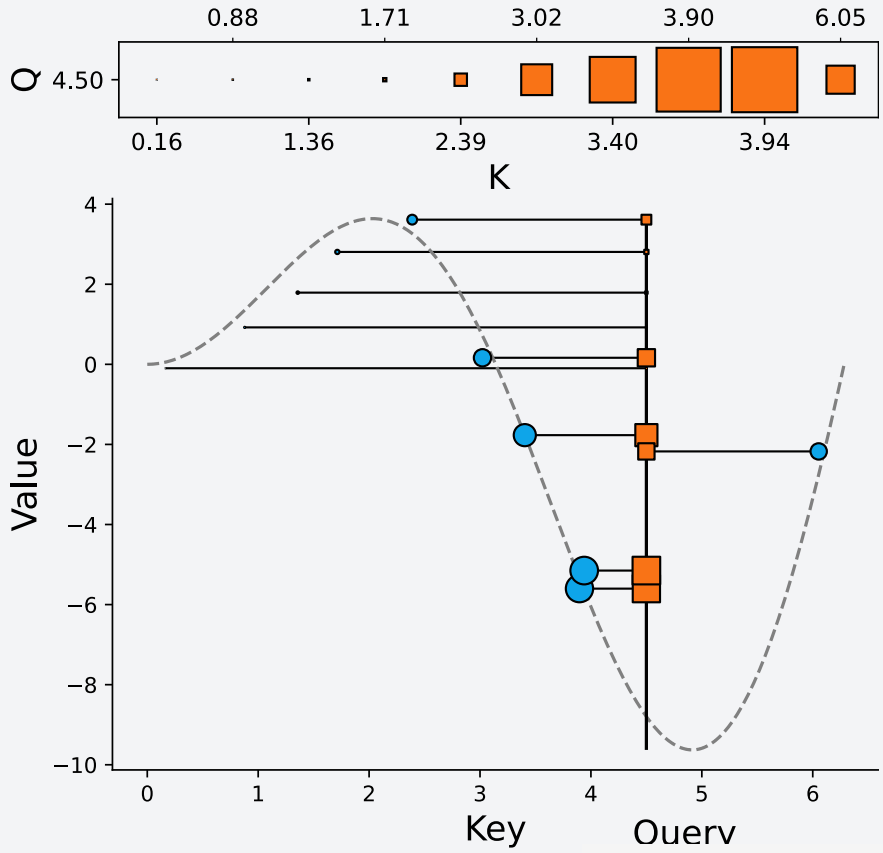
E **Nadaraya-Watson Kernel Regression**

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

- $\mathbf{Q}$ and $\mathbf{V}$ do not need to have the same sequence length
- The output of f is *always* a matrix of size $\mathbf{A} \in (0, 1)^{[t_Q \times t_V]}$

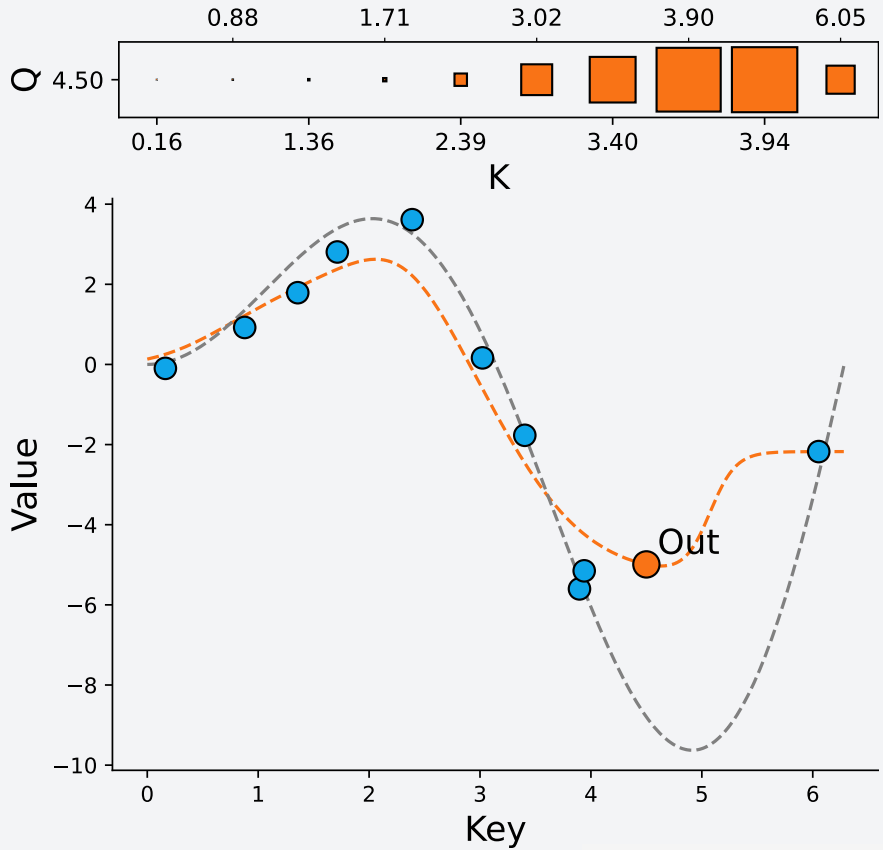
E **Nadaraya-Watson Kernel Regression**

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 q_i is similar to k_j
- Attention matrix tells us how important v_j is to q_i

Non-Transformer Examples

Multi-head Attention

- $f(\mathbf{Q}, \mathbf{K})$ is high when q_i is similar to k_j
- Attention matrix tells us how important v_j is to q_i

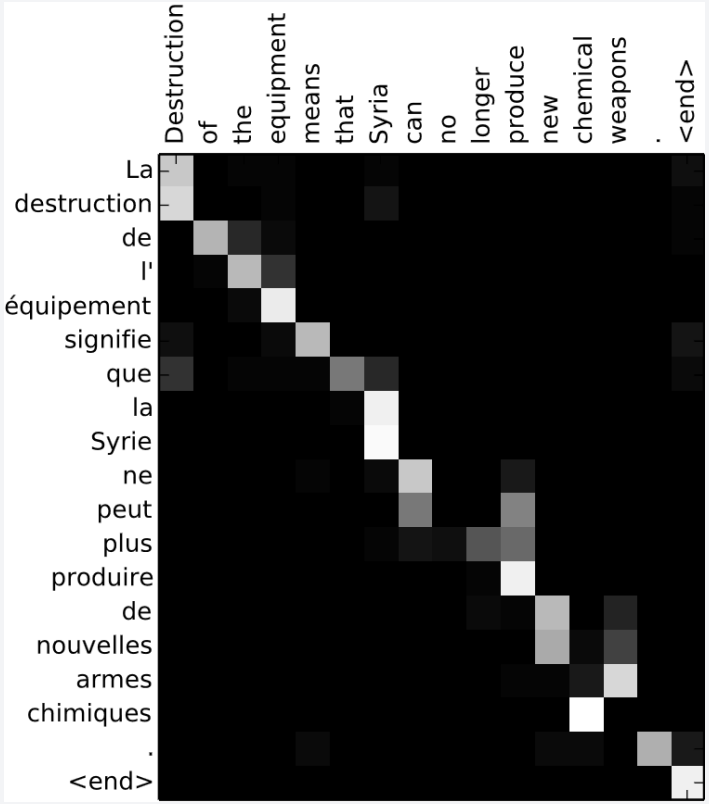
E Bahdanau et al. Alignment

In Neural Machine Translation (NMT) the encoder generates a representation of the input language

The decoder needs to generate in a target language

Token in input language != token in output language

Solution: have each token in the target language ($\mathbf{Q}$) attend back to all input language tokens ($\mathbf{K}$, $\mathbf{V}$)



Bahdanau, Cho & Bengio (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Non-Transformer Examples

Multi-head Attention

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

Non-Transformer Examples

Multi-head Attention

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

Model	$f(\mathbf{Q}, \mathbf{K})$
Gaussian	$\log \exp\left(\frac{(\mathbf{q}-\mathbf{k})^2}{\sigma^2}\right), \quad \forall \mathbf{q}, \mathbf{k} \in \mathbf{Q}, \mathbf{K}$
Cosine	$\frac{\mathbf{Q}\mathbf{K}^\intercal}{\|\|\mathbf{Q}\|\|\|\mathbf{K}\|\|}$
Additive	$\mathbf{v}^\intercal \tanh\left(\mathbf{W}\left[\mathbf{q}\|\|\mathbf{k}\right]\right), \quad \forall \mathbf{q}, \mathbf{k} \in \mathbf{Q}, \mathbf{K}$
General	$\mathbf{Q}\mathbf{W}\mathbf{K}^\intercal$

Attention in Transformers

Multi-head Attention

- Transformer Attention uses a **masked 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 $t_V \times d_K$

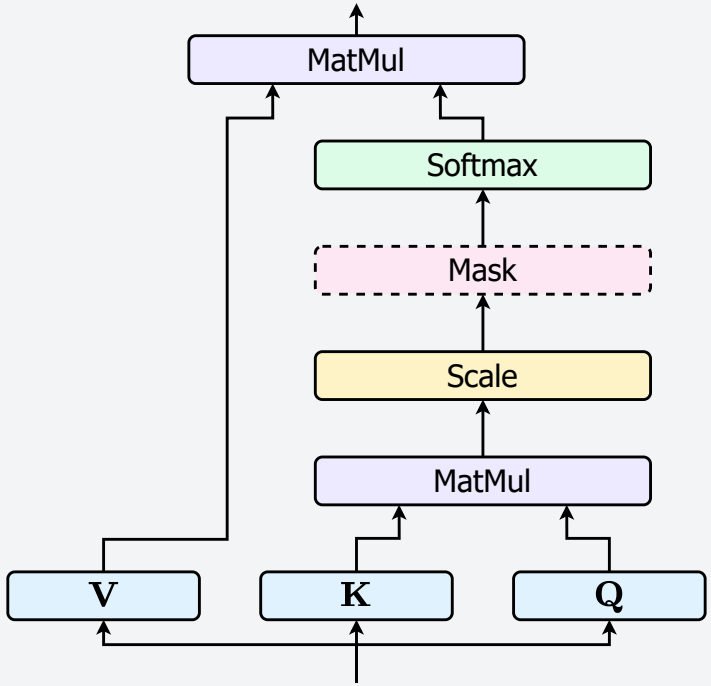
Attention in Transformers

Multi-head Attention

- Transformer Attention uses a **masked 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 in Transformers

Multi-head Attention

- Transformer Attention uses a **masked 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

- Transformer Attention uses a **masked scaled dot-product** kernel function

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

- Why mask?
  - Currently all tokens are treated equally

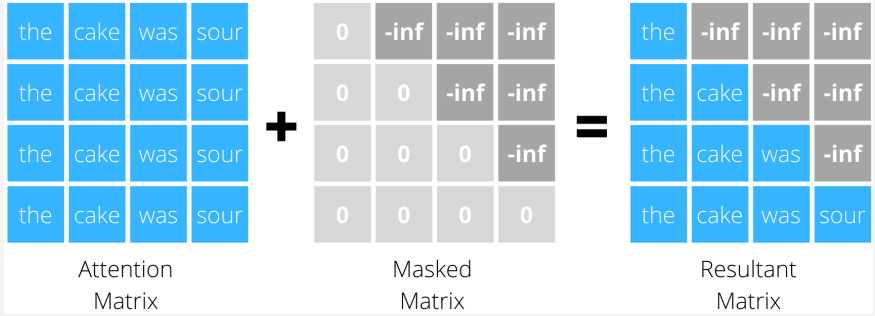
Attention in Transformers

Multi-head Attention

- Transformer Attention uses a **masked scaled dot-product** kernel function

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

- Why mask?
  - Currently all tokens are treated equally
  - **Causal masking**: decoder tokens should never attend to future tokens, only to the past



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

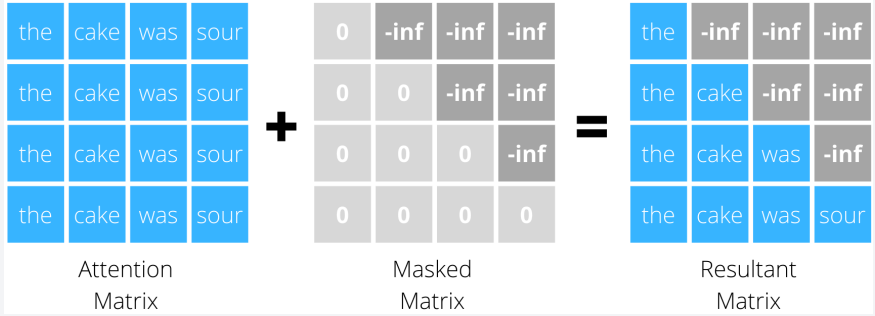
Attention in Transformers

Multi-head Attention

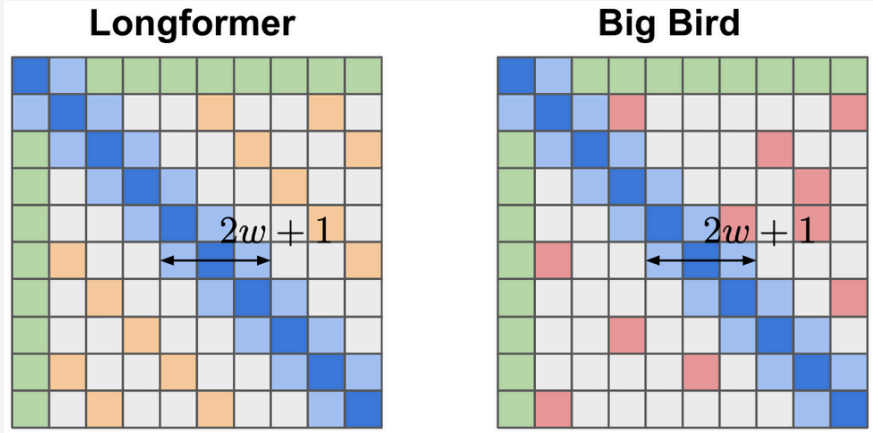
- Transformer Attention uses a **masked scaled dot-product** kernel function

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

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



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

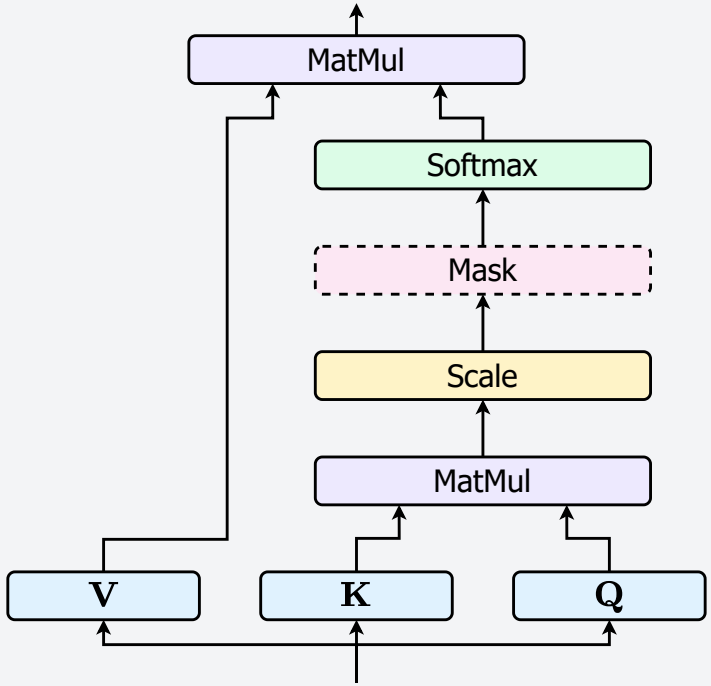


https://lilianweng.github.io/posts/2023-01-27-the-transformer-family-v2/

Attention in Transformers

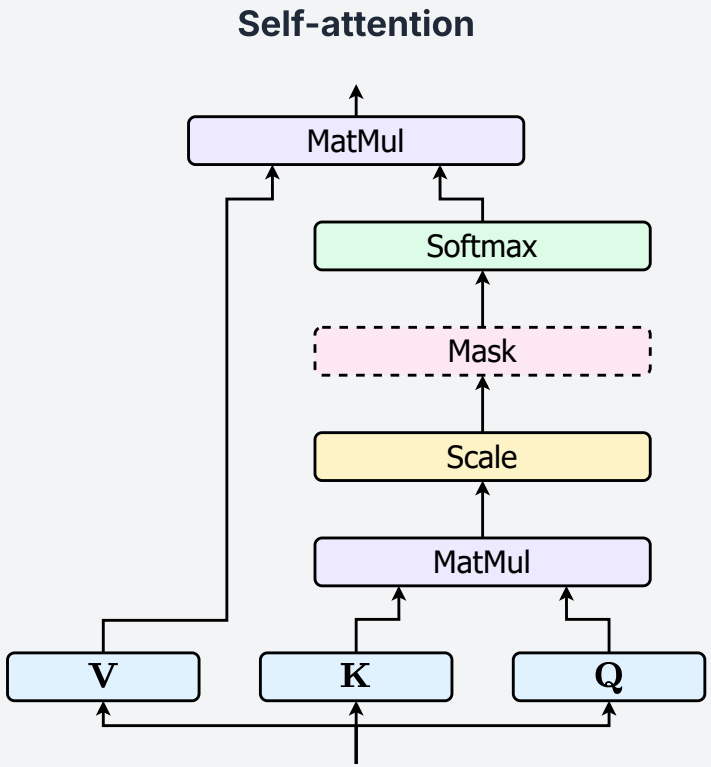
Multi-head Attention

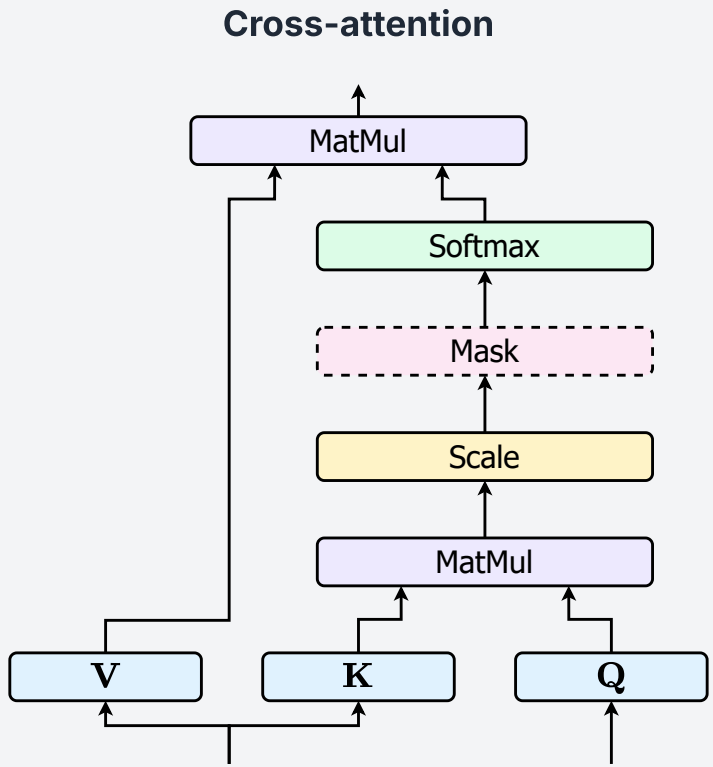
- Where do $\mathbf{V}$, $\mathbf{K}$, $\mathbf{Q}$ come from?



Attention in Transformers

Multi-head Attention



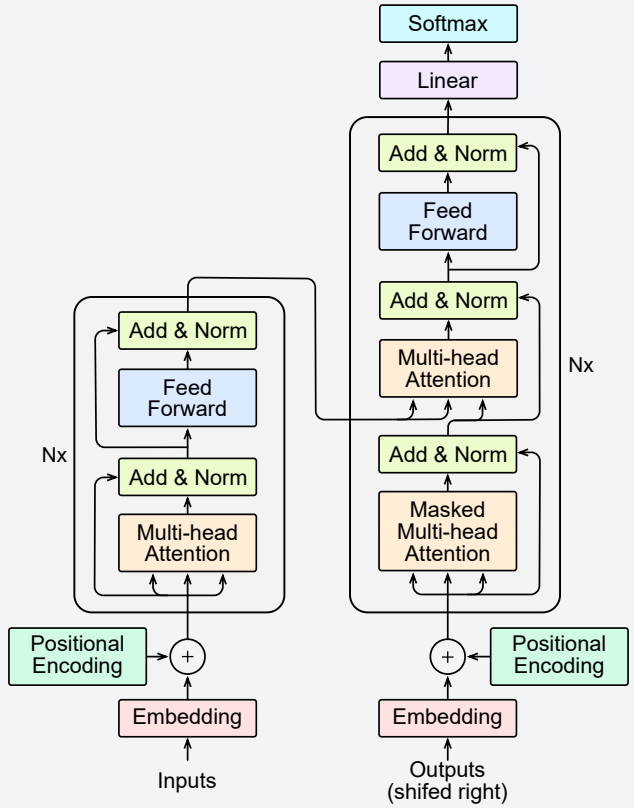


Attention in Transformers

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



Attention in Transformers

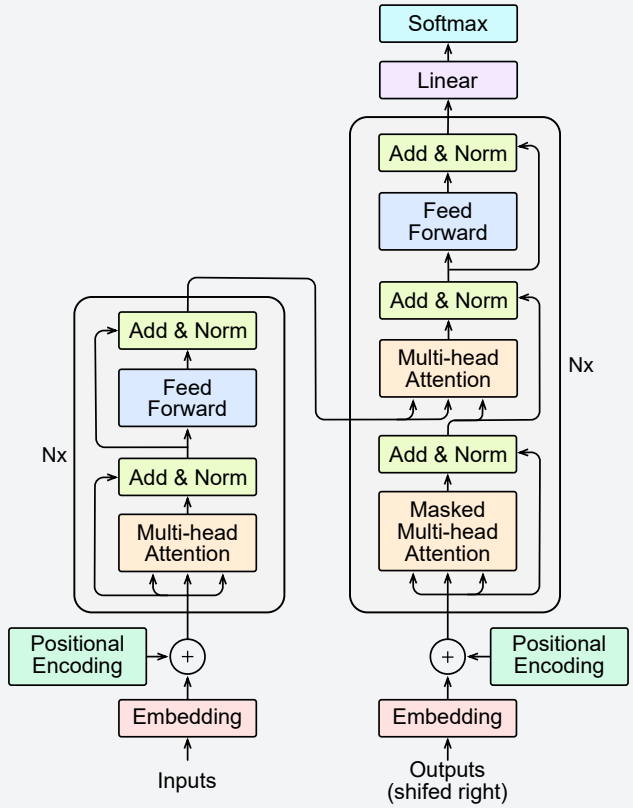
Multi-head Attention

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 $\underbrace{\mathbf{Q} = \mathbf{Y}\mathbf{W}_Q}_{\text{Cross-attention}}$



Multi-head Attention

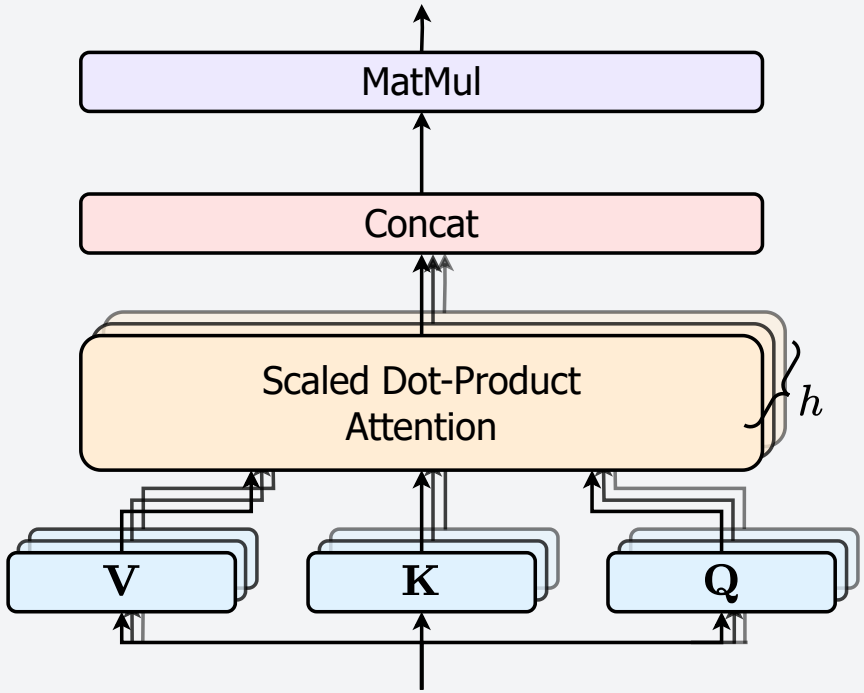
Multi-head Attention

- 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

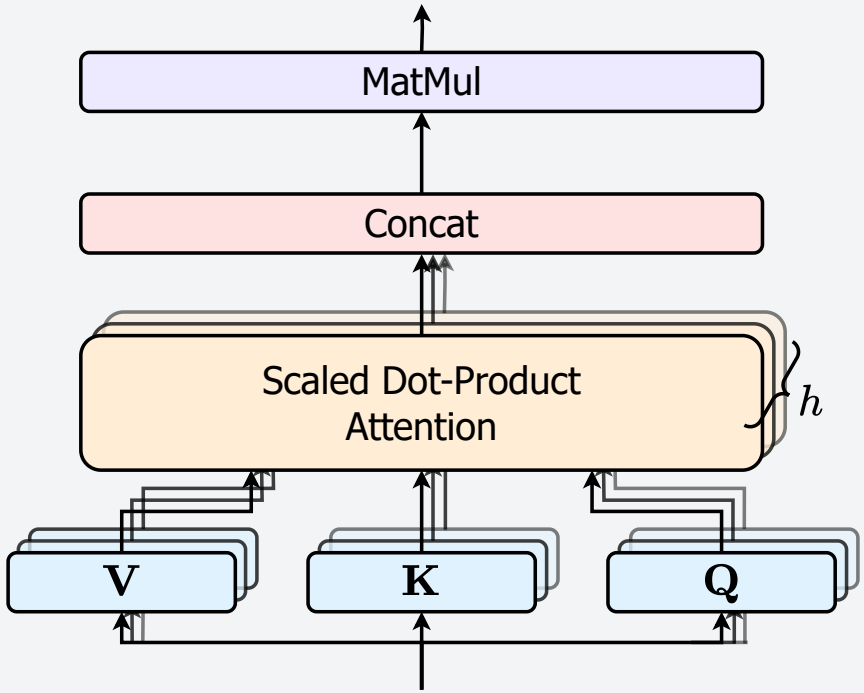
- Currently we use 1 set of attention weights
  - Can only process 1 query type
- With h attention heads, we learn h concepts
  - To reduce cost, reduce dimensionality $d_{K,V}/h$

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

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. Advances in neural information processing systems, 30. (p. 5 & 15)

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. Advances in neural information processing systems, 30. (p. 5 & 15)

Multiple heads, multiple different queries processed in parallel

- Positional heads
- Syntactic heads
- Rare words?

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

Multi-head Attention

Multi-head Attention

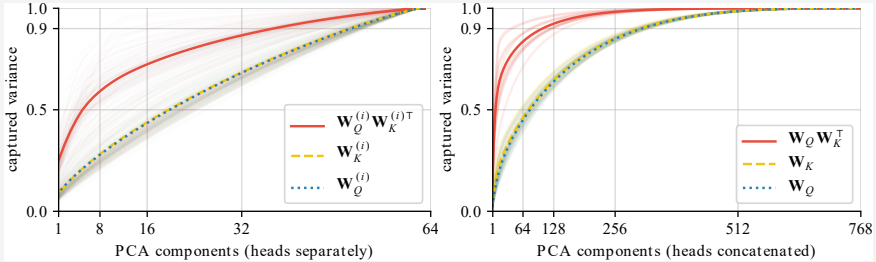
Do different heads attend to different concepts?

Multi-head Attention

Do different heads attend to different concepts?

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

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

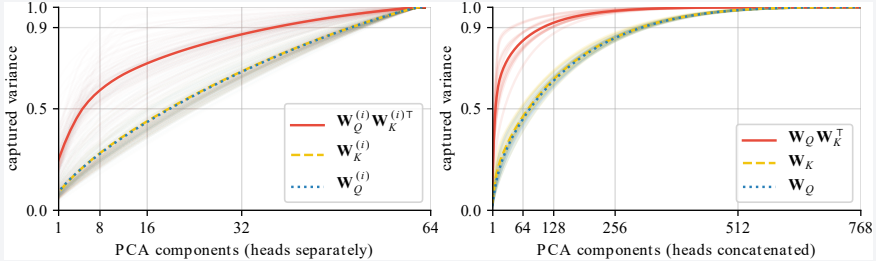


Multi-head Attention

Do different heads attend to different concepts?

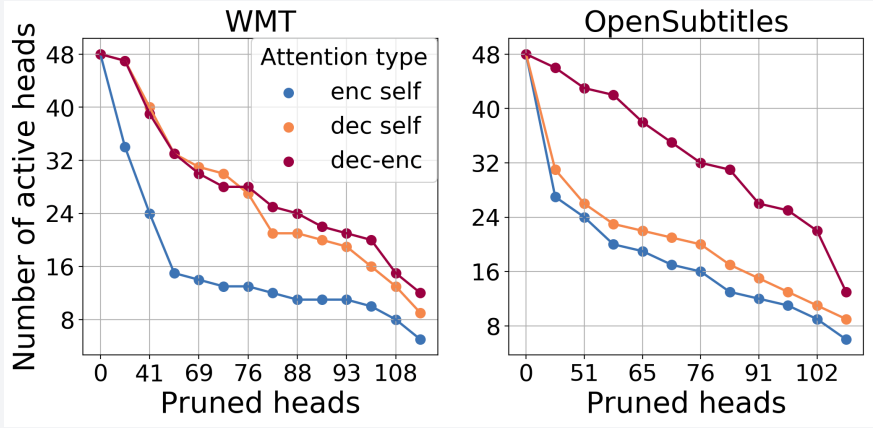
- Individual heads = high rank, concatenated heads = low rank

```
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

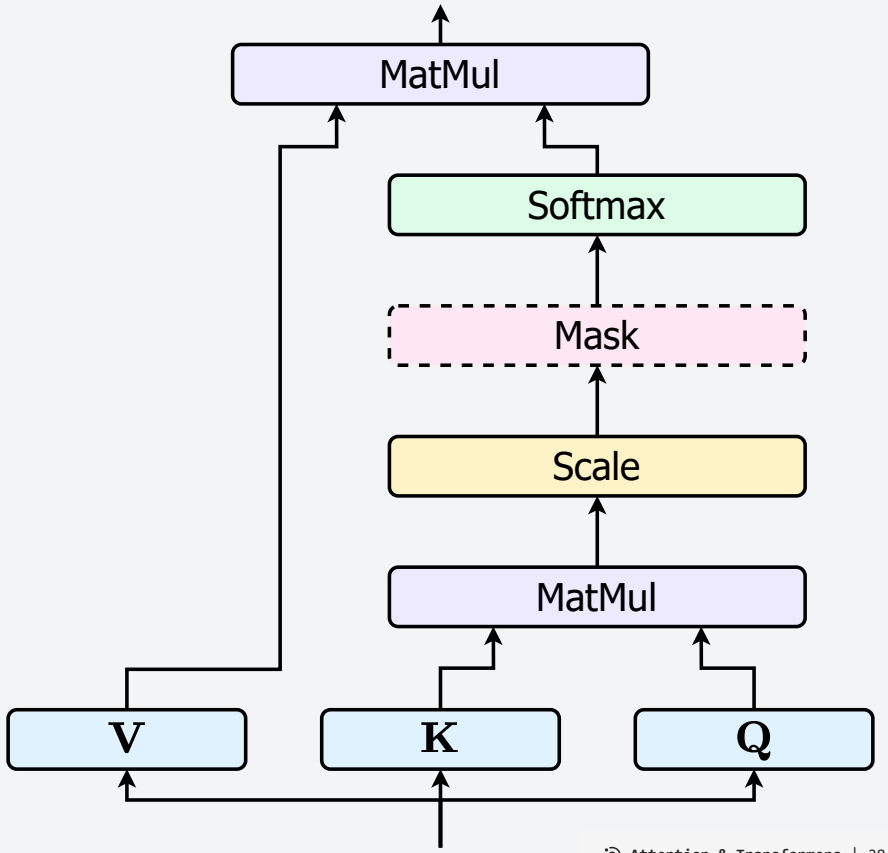
```
Voita et al. (2019). Analyzing Multi-Head Self-Attention:
Specialized Heads Do the Heavy Lifting, the Rest Can
Be Pruned. Association for Computational Linguistics.
```



Why Attention?

Multi-head Attention

Summary

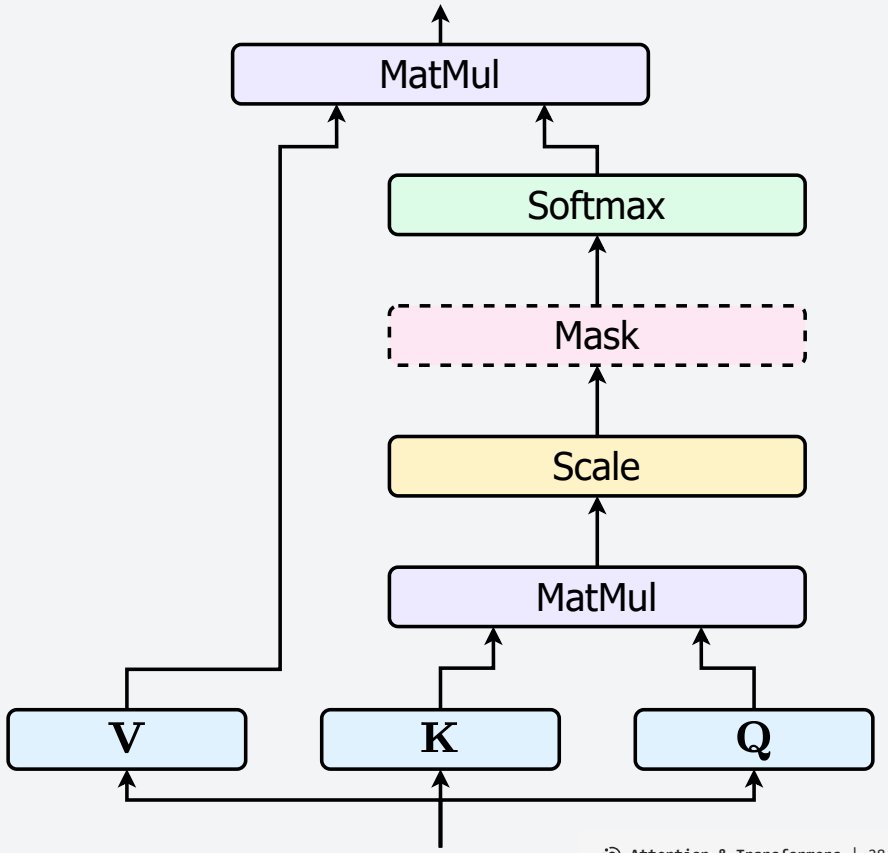


Why Attention?

Multi-head Attention

Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f(\mathbf{Q}, \mathbf{K})$

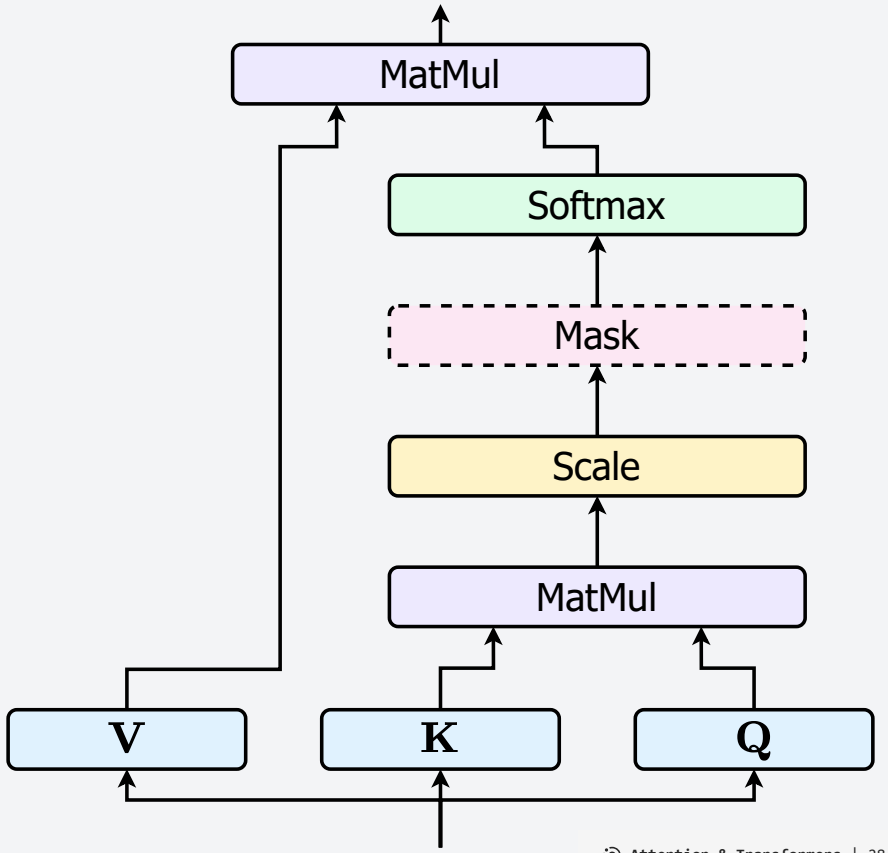


Why Attention?

Multi-head Attention

Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f(\mathbf{Q}, \mathbf{K})$
2. The values of $\mathbf{A}$ are all in $(0, 1)$, making $\mathbf{AV}$ a convex combination/weighted mean of $\mathbf{V}$

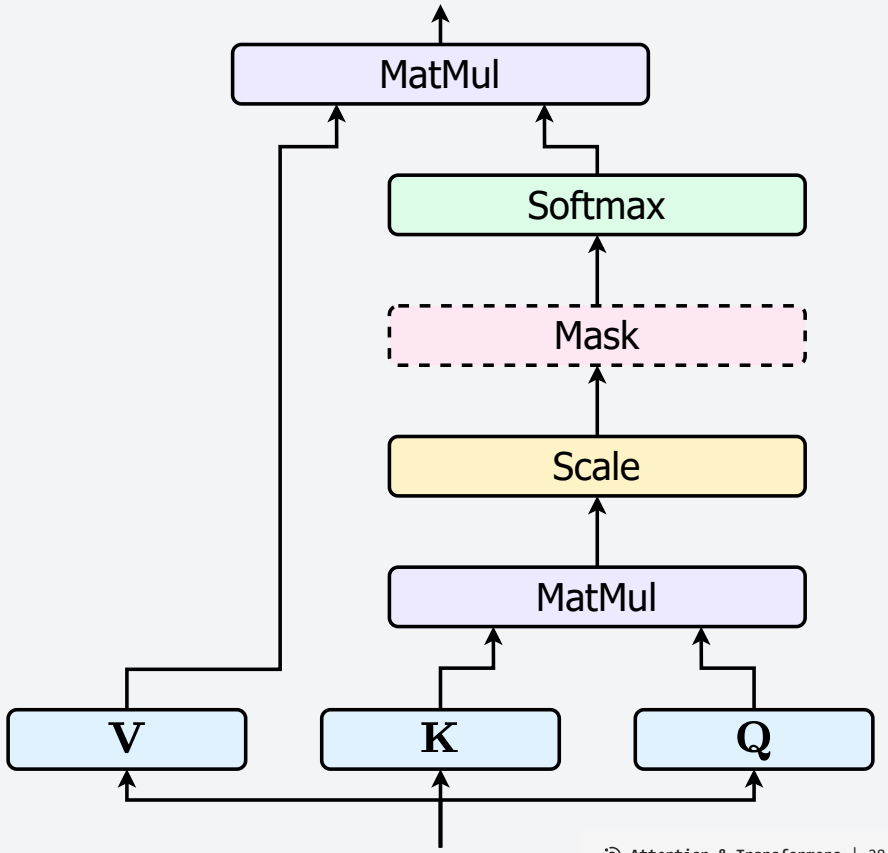


Why Attention?

Multi-head Attention

Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f(\mathbf{Q}, \mathbf{K})$
2. The values of $\mathbf{A}$ are all in $(0, 1)$, making $\mathbf{AV}$ a convex combination/weighted mean of $\mathbf{V}$
3. Attention is does not understand word order

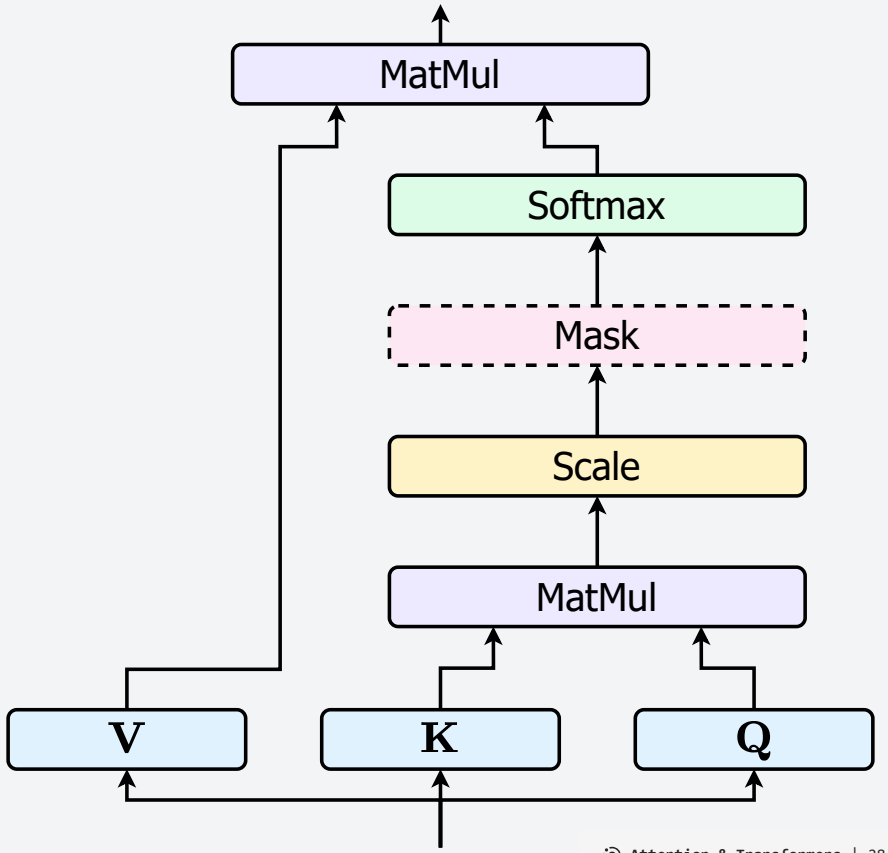


Why Attention?

Multi-head Attention

Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f\left(\mathbf{Q}, \mathbf{K}\right)$
2. The values of $\mathbf{A}$ are all in $(0, 1)$, making $\mathbf{AV}$ a convex combination/weighted mean of $\mathbf{V}$
3. Attention is does not understand word order
4. In Transformers, Attention is used to :
   - add context from self (self-attention)
   - add context from others (cross-attention)

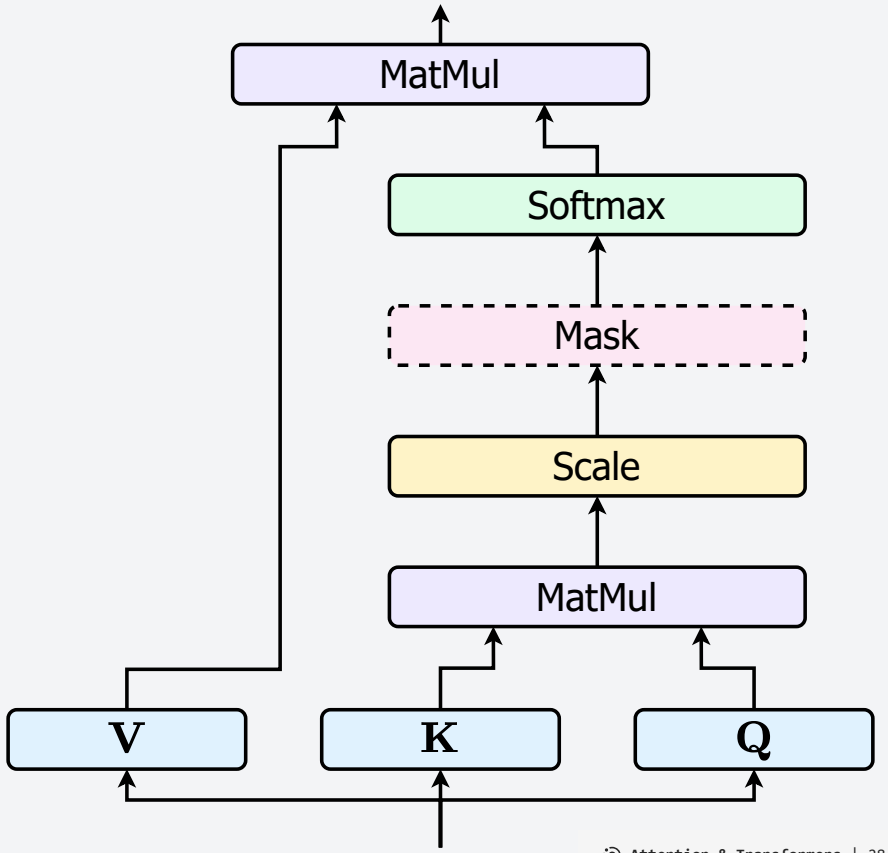


Why Attention?

Multi-head Attention

Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f(\mathbf{Q}, \mathbf{K})$
2. The values of $\mathbf{A}$ are all in $(0, 1)$, making $\mathbf{AV}$ a convex combination/weighted mean of $\mathbf{V}$
3. Attention is does not understand word order
4. In Transformers, Attention is used to :
   - add context from self (self-attention)
   - add context from others (cross-attention)
5. Attention cost scales quadratically with sequence length

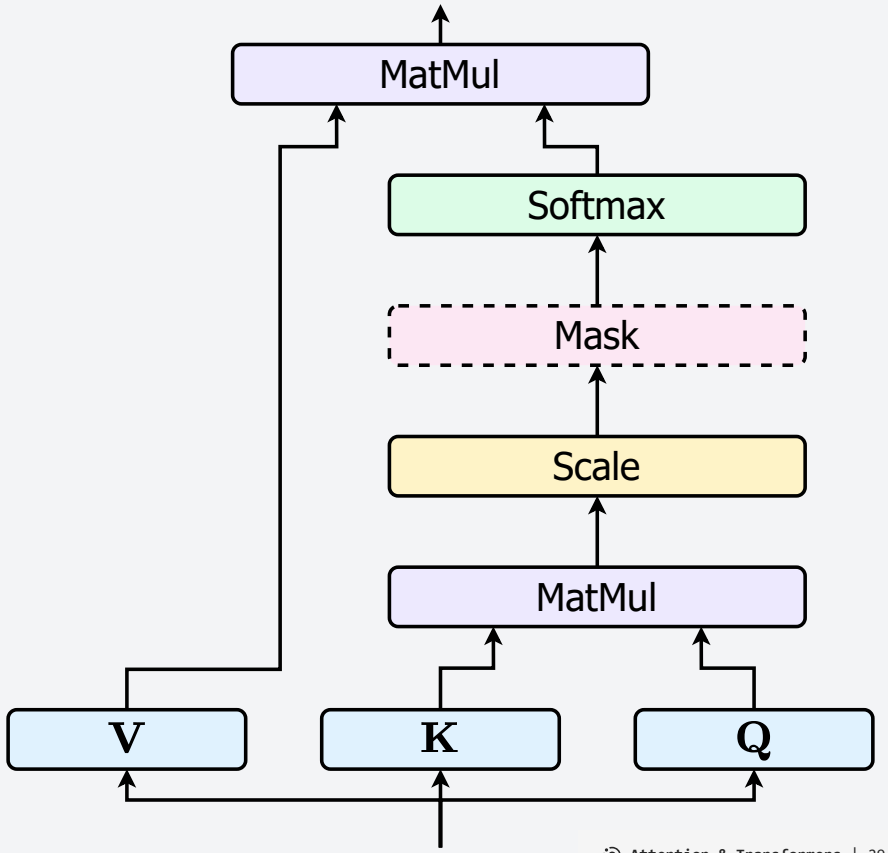


Why Attention?

Multi-head Attention

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

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



Why Attention?

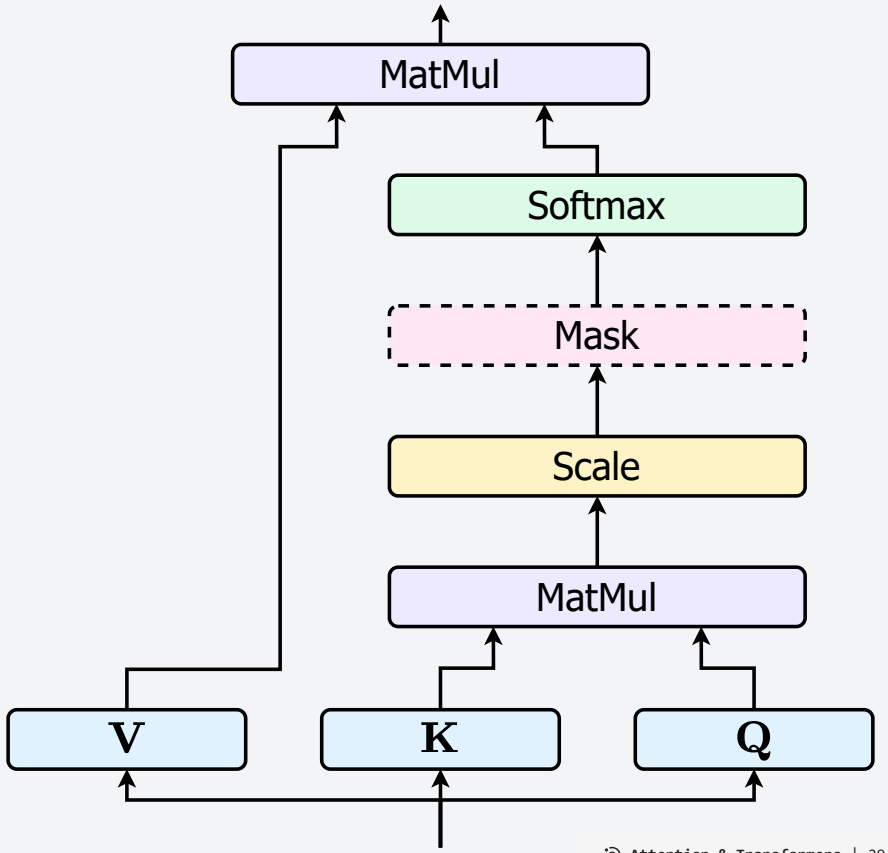
Multi-head Attention

- 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...

$$\mathcal{O}\left(\underbrace{t_x \cdot d_x \cdot d_h}_{x_t} + \underbrace{t_x \cdot d_h^2}_{h_{t-1}}\right)$$

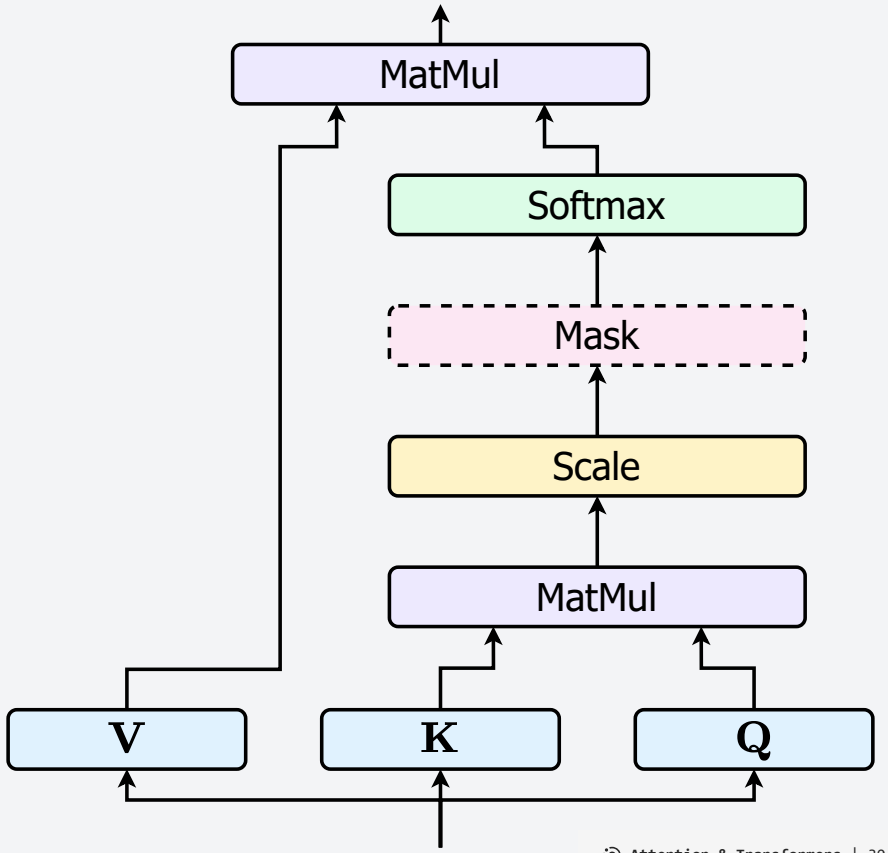


Why Attention?

Multi-head Attention

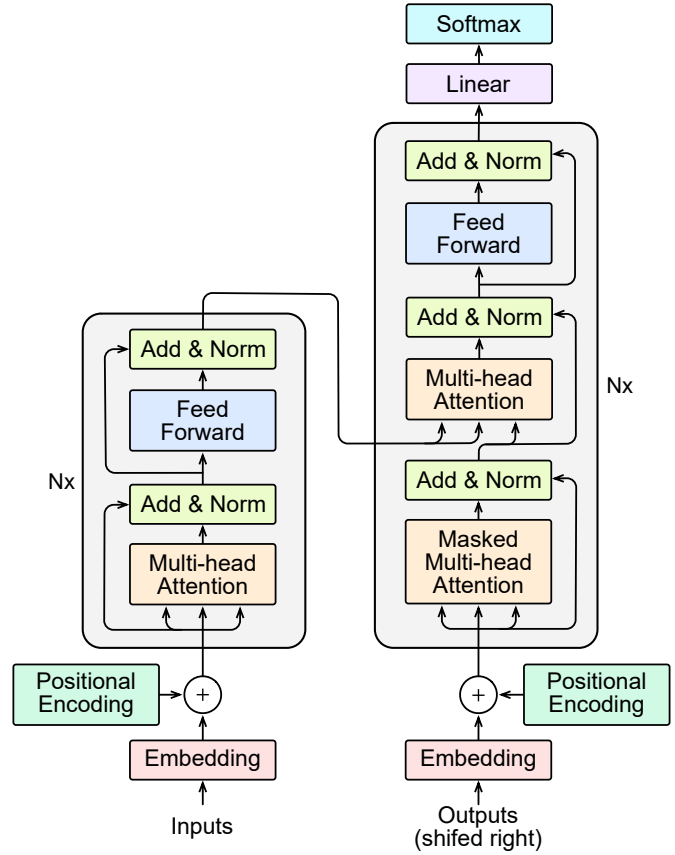
Summary

1. Attention is a **linear map** $\mathbf{AV}$ where $\mathbf{A}$ is dynamically constructed from $f(\mathbf{Q}, \mathbf{K})$
2. The values of $\mathbf{A}$ are all in $(0, 1)$, making $\mathbf{AV}$ a convex combination/weighted mean of $\mathbf{V}$
3. Attention is does not understand word order
4. In Transformers, Attention is used to:
   - add context from self (self-attention)
   - add context from others (cross-attention)
5. Attention cost scales quadratically with sequence length
6. **Attention is parallelizable**



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/

Feed Forward



Non-linearities

Feed Forward

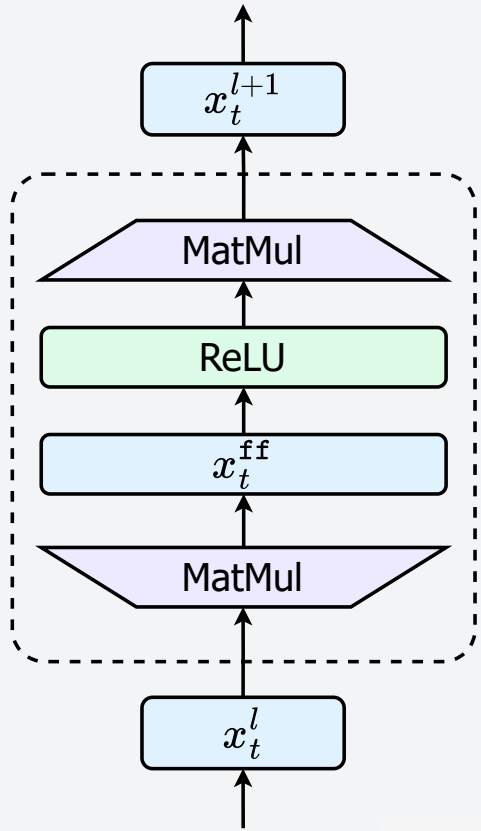
- Attention is just a fancy linear map*
  - Neural nets need non-linear operations

* More or less

Non-linearities

Feed Forward

- Attention is just a fancy linear map*
  - Neural nets need non-linear operations
- Add pointwise feed-forward nets for non-linear expressiveness
  - Just an MLP applied to each time step

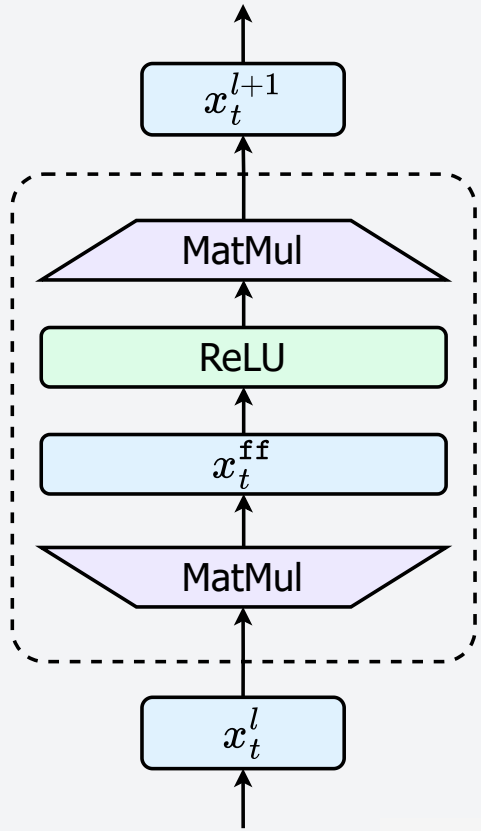


* More or less

Non-linearities

Feed Forward

- Attention is just a fancy linear map*
  - Neural nets need non-linear operations
- Add pointwise feed-forward nets for non-linear expressiveness
  - Just an MLP applied to each time step
- Assume $d_{\texttt{ff}} \gg d_V$
  - Depth is serial
  - Width is parallel



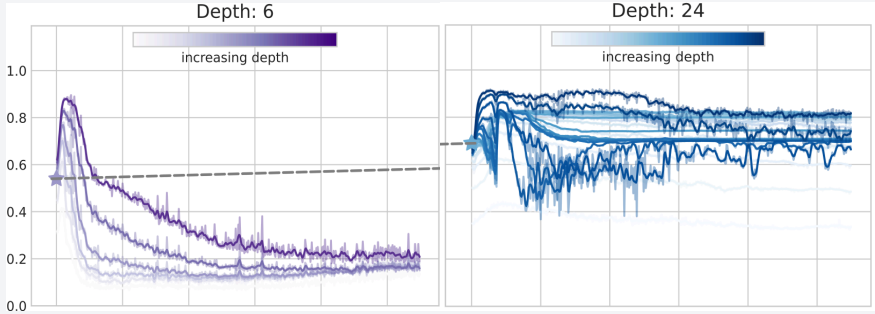
* More or less

Necessity of Feed Forward Nets

Feed Forward

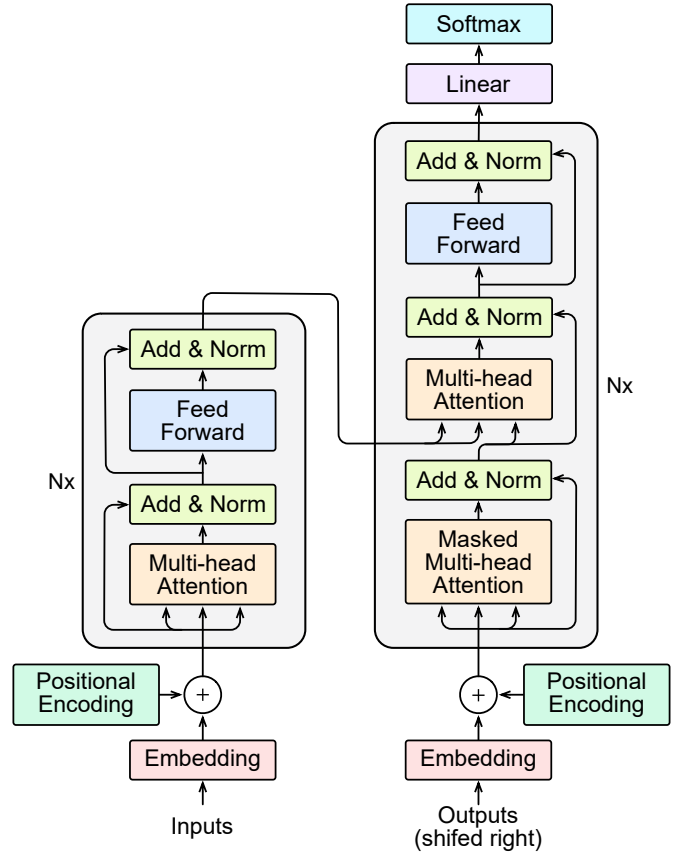
- Is attention all you need?
- Without feed forward nets, transformer token representations collapse
  - Occurs *doubly exponential* with depth (very, very fast)

Dong, Cordonnier, & Loukas (2021). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. PMLR



Noci et al. (2022). Signal Propagation in Transformers: Theoretical Perspectives and the Role of Rank Collapse. arXiv:2206.03126.

Add & Norm


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

Residual Connections

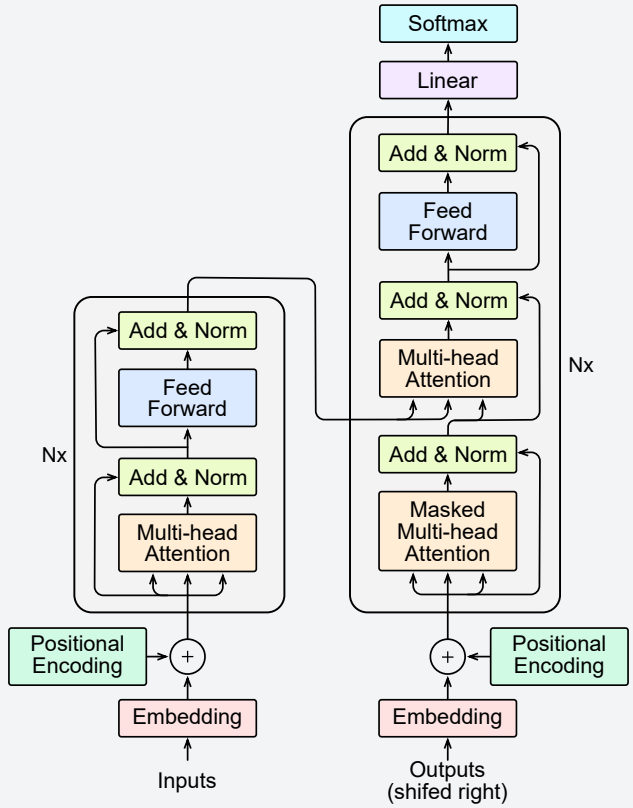
Add & Norm

- After each operation, the input is added back in

$$x^{l+1} = \texttt{SubLayer}(x^l) + x^l$$

- Makes deep architectures optimisable

He et al. (2015). Deep residual learning for image recognition. arXiv:1512.03385



Layer Norm

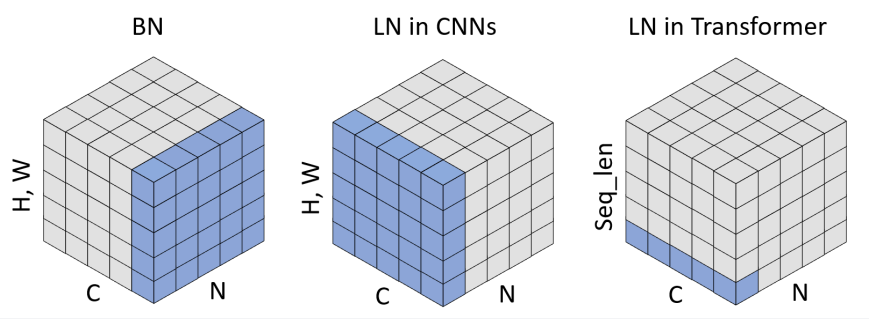
Add & Norm

- After each operation and residual connection, normalize

$$x^{l+1} = \texttt{LayerNorm}(\texttt{SubLayer}(x^l) + x^l)$$

- LayerNorm in Transformers is per token
- Makes neural nets converge faster

Ba, Kiros & Hinton (2016). Layer normalization. arXiv:1607.06450



Yao et al. (2021). Leveraging batch normalization for vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 413-422).

Embeddings

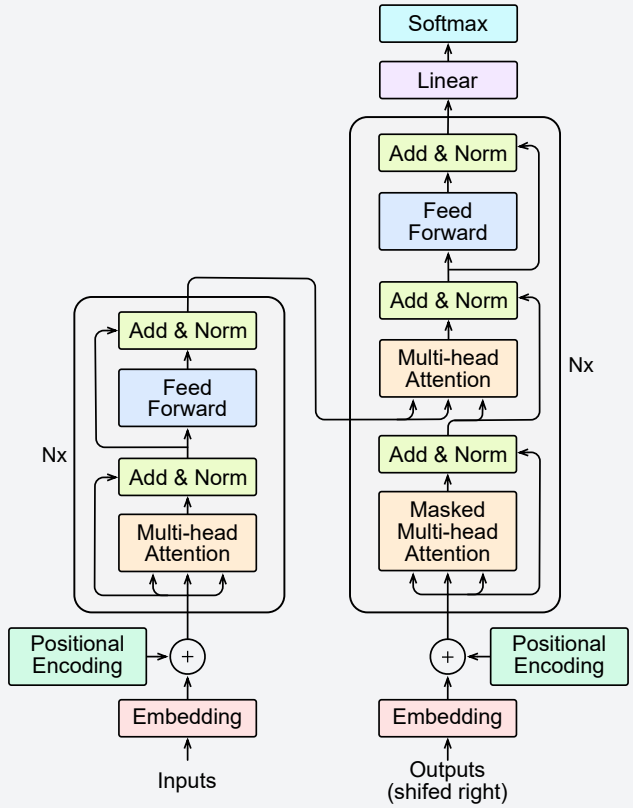
From words to vectors and back

Embedding Types

Embeddings

Transformers apply two embeddings:

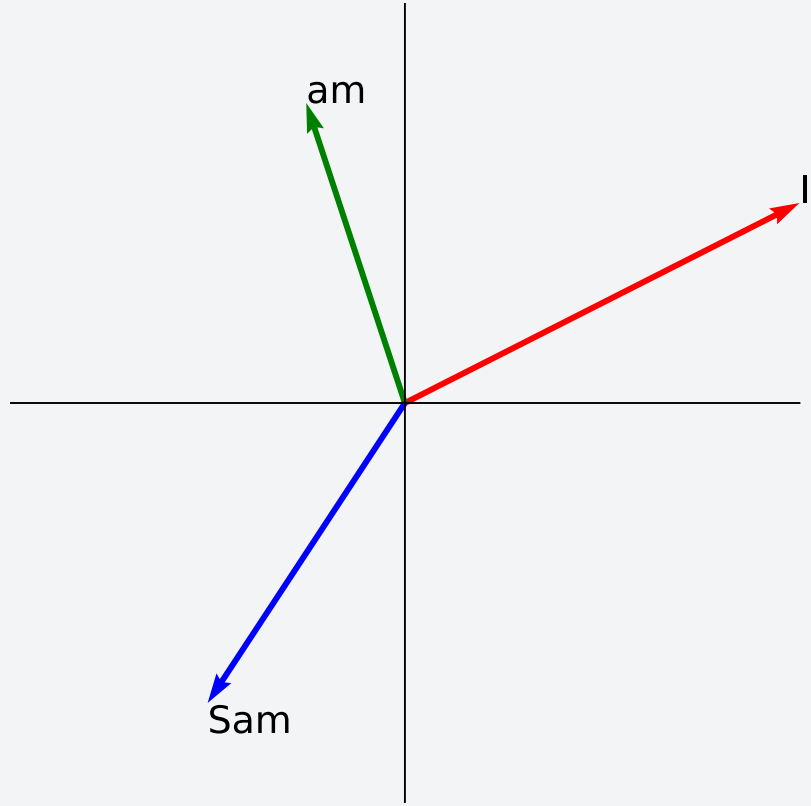
1. Token Embedding transform token IDs into vectors
2. Position Encodings add information about token location



Token to vector

Token Embedding

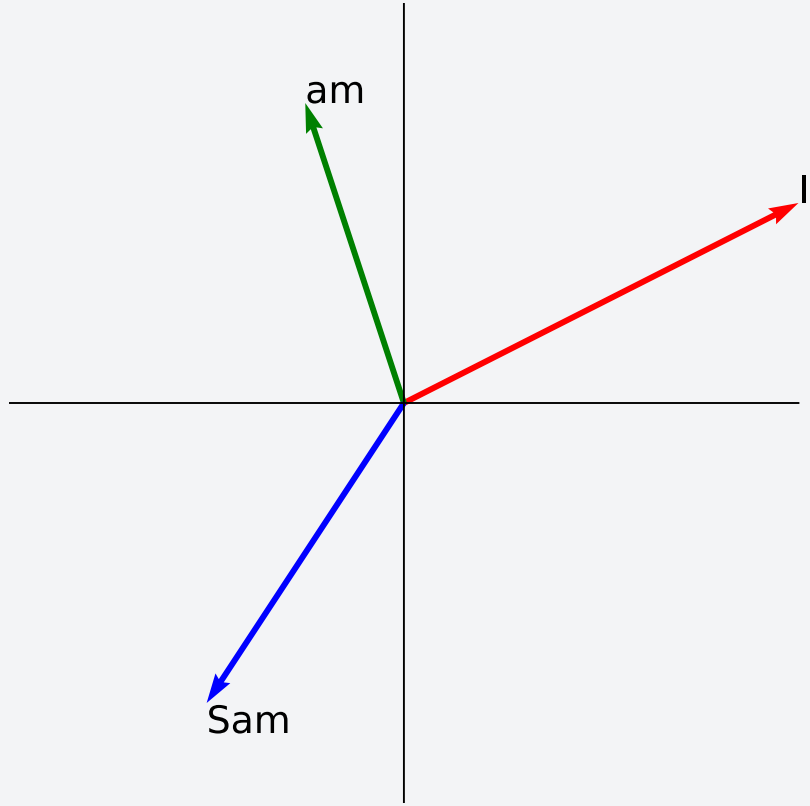
- Convert strings to dense representations
  - Typically done using dictionary lookup



Token to vector

Token Embedding

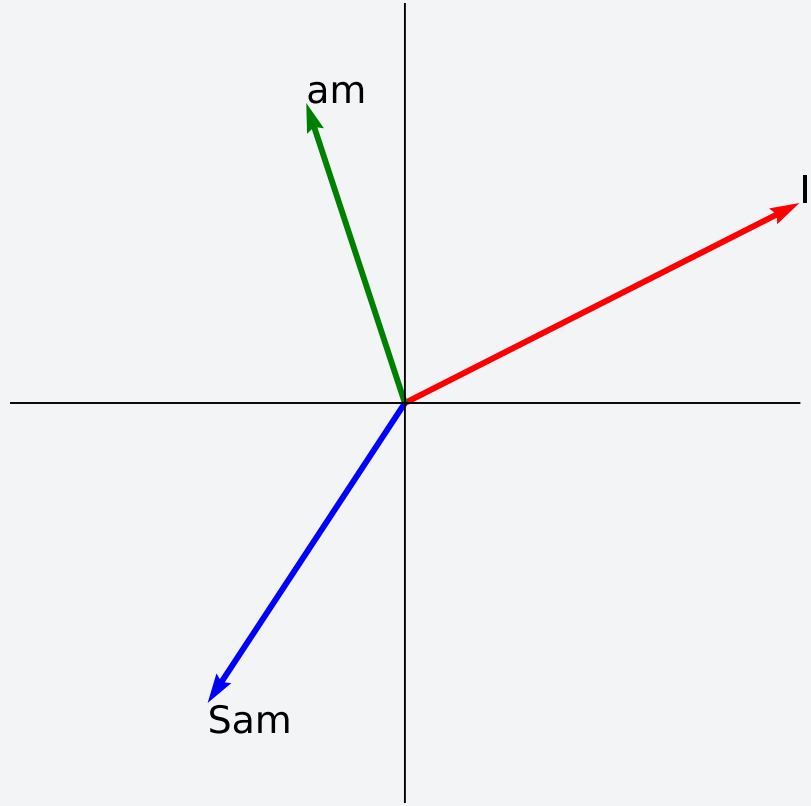
- Convert strings to dense representations
  - Typically done using dictionary lookup
- Memory cost: $\mathcal{O}(|\mathcal{V}| \cdot d_v)$
  - Usually most expensive operation
  - Can save cost by have embedding matrices share weights



Token to vector

Token Embedding

- Convert strings to dense representations
  - Typically done using dictionary lookup
- Memory cost: $\mathcal{O}(|\mathcal{V}| \cdot d_v)$
  - Usually most expensive operation
  - Can save cost by have embedding matrices share weights
- Typically *not* pre-trained



Permutation Equivariance Revisited

Positional Encoding

Permutation Equivariance

Attention is permutation *equivariant* to a changes in the word order of its input

- From NLP1: word order is pretty important for modelling language...

'Only I love you'

=

'I only love you'

=

'I love only you'

=

'I love only you, only...'

Permutation Equivariance Revisited

Positional Encoding

Permutation Equivariance

Attention is permutation *equivariant* to a changes in the word order of its input

- From NLP1: word order is pretty important for modelling language...
- Solution: add token position information to token embedding

$$\texttt{embed}(\texttt{string}[t]) + \texttt{position}(t)$$

'Only I love you'

=

'I only love you'

=

'I love only you'

=

'I love only you, only...'

Learning Position Embeddings

Positional Encoding

Proposal 1: learn per-position offsets manually

Learning Position Embeddings

Positional Encoding

Proposal 1: learn per-position offsets manually

1. Deterministic [✓]
2. Distinct at all time-steps [?]

Learning Position Embeddings

Positional Encoding

Proposal 1: learn per-position offsets manually

1. Deterministic [✓]
2. Distinct at all time-steps [?]
3. Extends to different sequence lengths [X]
4. Encodes relative positions [?]

Learning Position Embeddings

Positional Encoding

Proposal 1: learn per-position offsets manually

1. Deterministic [✓]
2. Distinct at all time-steps [?]
3. Extends to different sequence lengths [X]
4. Encodes relative positions [?]

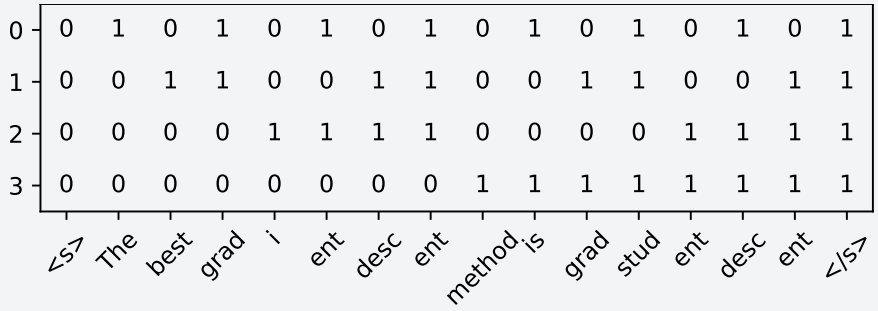
Not a terrible idea

Model	PPL (dev)	BLEU (test)	Params $\times 10^6$
Base	4.92	25.8	65
Learned Positional Encoding	4.92	25.7	65

Bits as Position Encoding

Positional Encoding

Proposal 2: add the offsets in bit representations to the embeddings

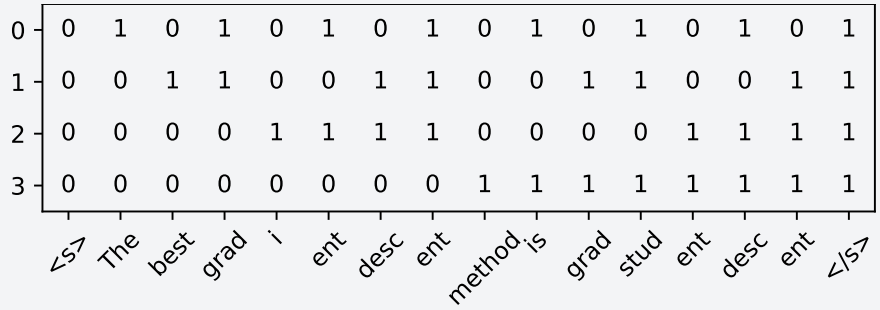


Bits as Position Encoding

Positional Encoding

Proposal 2: add the offsets in bit representations to the embeddings

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]*
4. Encodes relative positions [X]

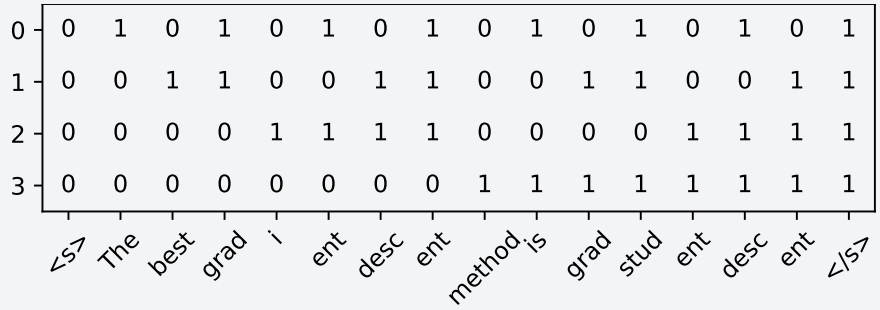


Bits as Position Encoding

Positional Encoding

Proposal 2: add the offsets in bit representations to the embeddings

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]*
4. Encodes relative positions [X]
5. Elegant [X]



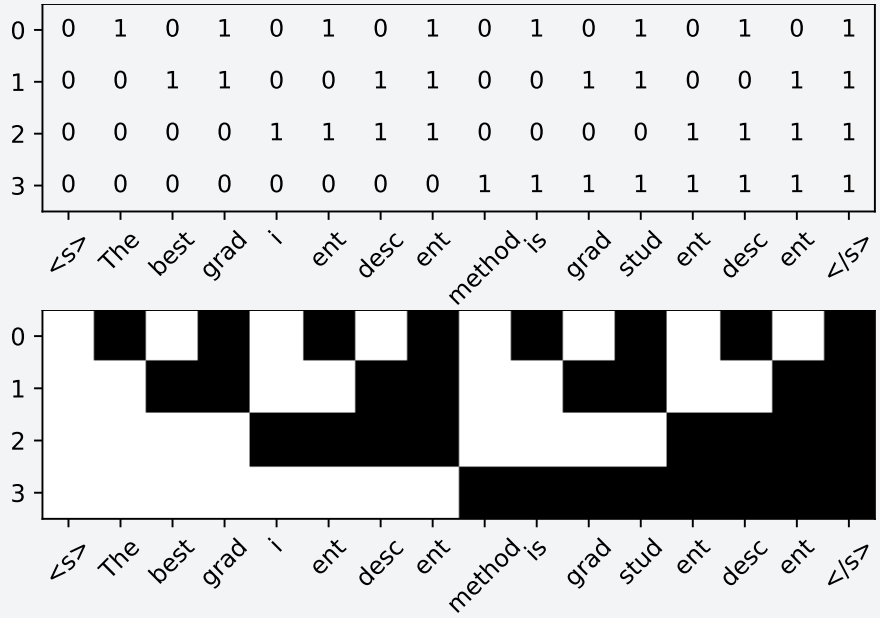
* Maximum sequence length is now $2^{d_{\text{model}}}$

Bits as Position Encoding

Positional Encoding

Proposal 2: add the offsets in bit representations to the embeddings

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]*
4. Encodes relative positions [X]
5. Elegant [X]



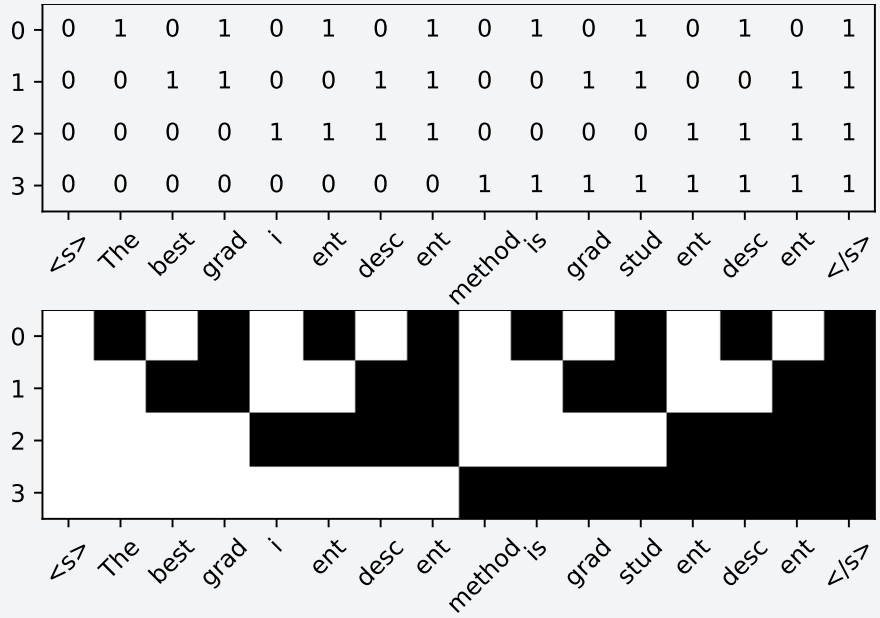
* Maximum sequence length is now $2^{d_{\text{model}}}$

Bits as Position Encoding

Positional Encoding

Proposal 2: add the offsets in bit representations to the embeddings

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]*
4. Encodes relative positions [X]
5. Elegant [X]



* Maximum sequence length is now $2^{d_{\text{model}}}$

Sinusoidal Position Encoding

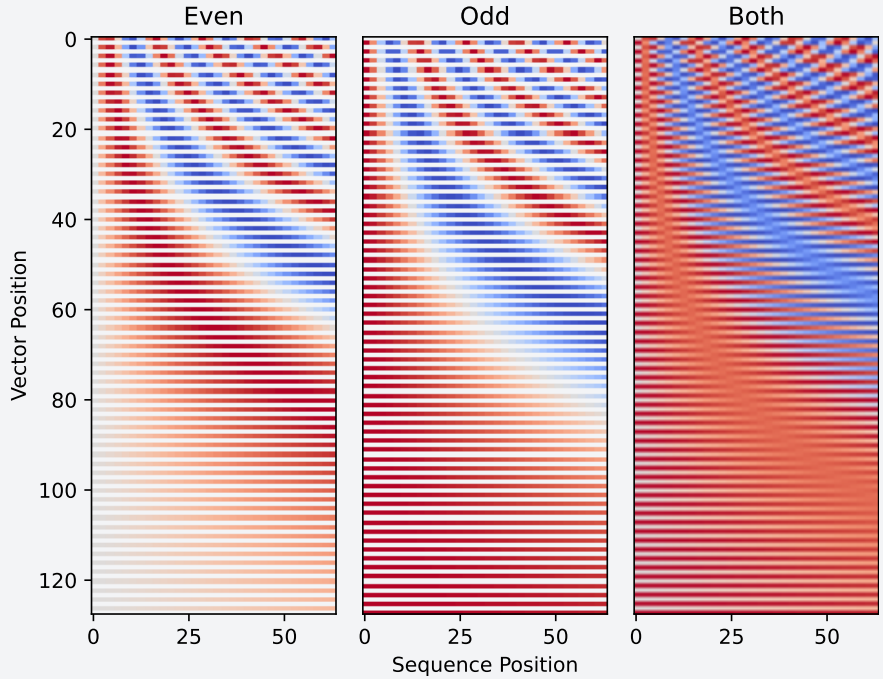
Positional Encoding

Proposal 3: use sinusoidal position encoding

$$\texttt{Position}(t) = \begin{cases} \sin\left(\omega_i \cdot t\right), & \mod\left(i, 2\right) = 0 \\ \cos\left(\omega_i \cdot t\right), & \mod\left(i, 2\right) \neq 0 \end{cases}$$

$$\omega_i = k^{-2i/d_{\text{model}}}$$

- First dimensions get very quickly oscillating sinusoid
- Last dimensions get very slow oscillating sinusoid



Sinusoidal Position Encoding

Positional Encoding

Relative positions can now be represented as a linear mapping, dependent on the offset

Neural nets can learn this offset

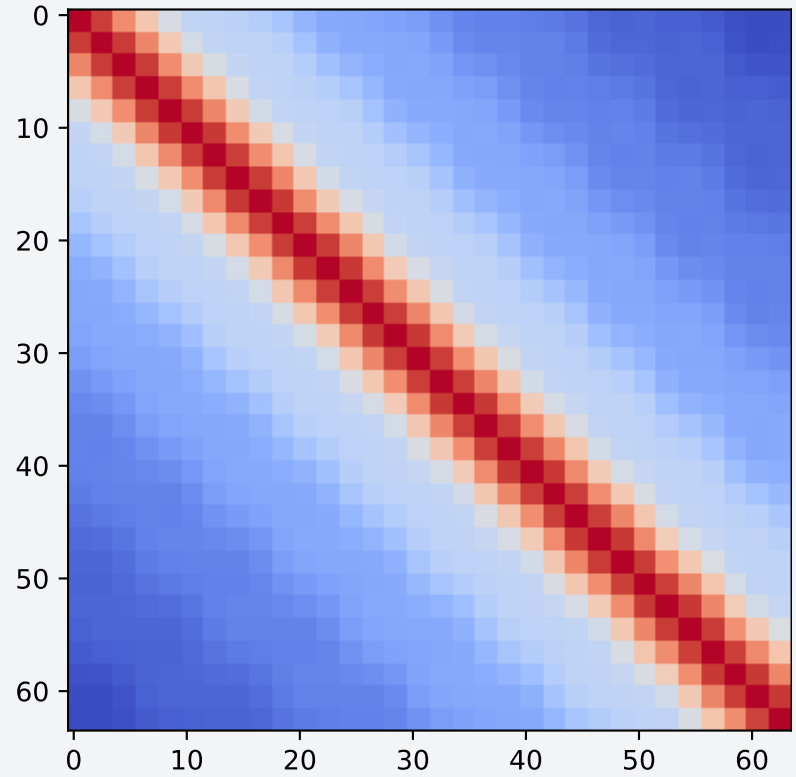
$$
\begin{aligned}
\texttt{Position}(t+\Delta t) &= \begin{bmatrix} \sin\left(\omega_i \cdot (t+\Delta t)\right) \\ \cos\left(\omega_i \cdot (t+\Delta t)\right) \end{bmatrix} \\
&= \begin{bmatrix} \cos\Delta t & \sin\Delta t \\ -\sin\Delta t & \cos\Delta t \end{bmatrix} \begin{bmatrix} \sin\left(\omega_i \cdot t\right) \\ \cos\left(\omega_i \cdot t\right) \end{bmatrix} \\
&= \begin{bmatrix} \cos\Delta t & \sin\Delta t \\ -\sin\Delta t & \cos\Delta t \end{bmatrix} \texttt{Position}(t) \\
&= \mathbf{A}_{\Delta t}\texttt{Position}(t)
\end{aligned}
$$

Sinusoidal Position Encoding

Positional Encoding

Cosine distance between time-steps decreases with offset magnitude

For proofs & more, see here

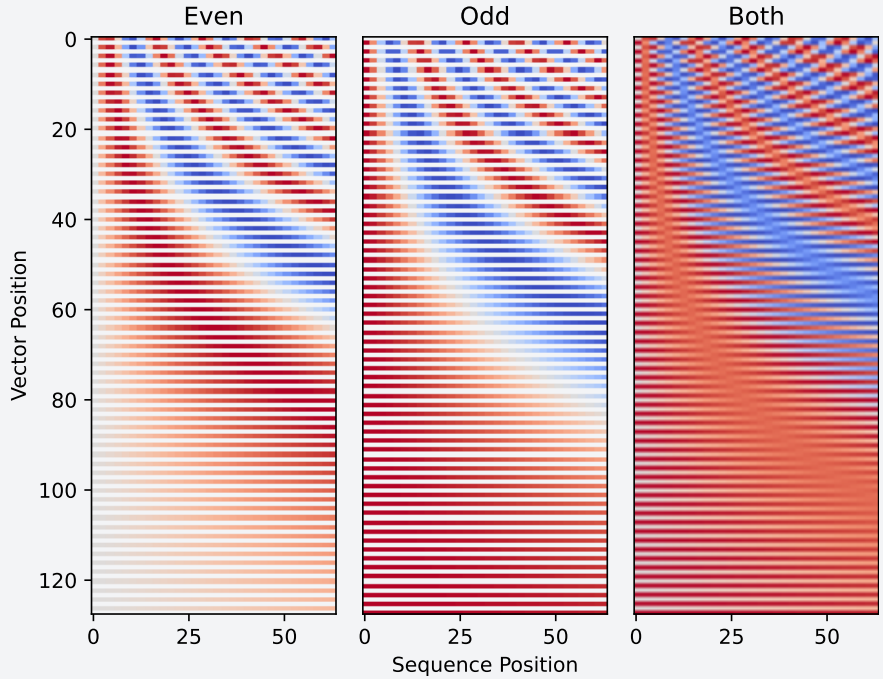


Sinusoidal Position Encoding

Positional Encoding

Proposal 3: use sinusoidal position encoding

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]
4. Encodes relative positions [✓]
5. Elegant [✓]



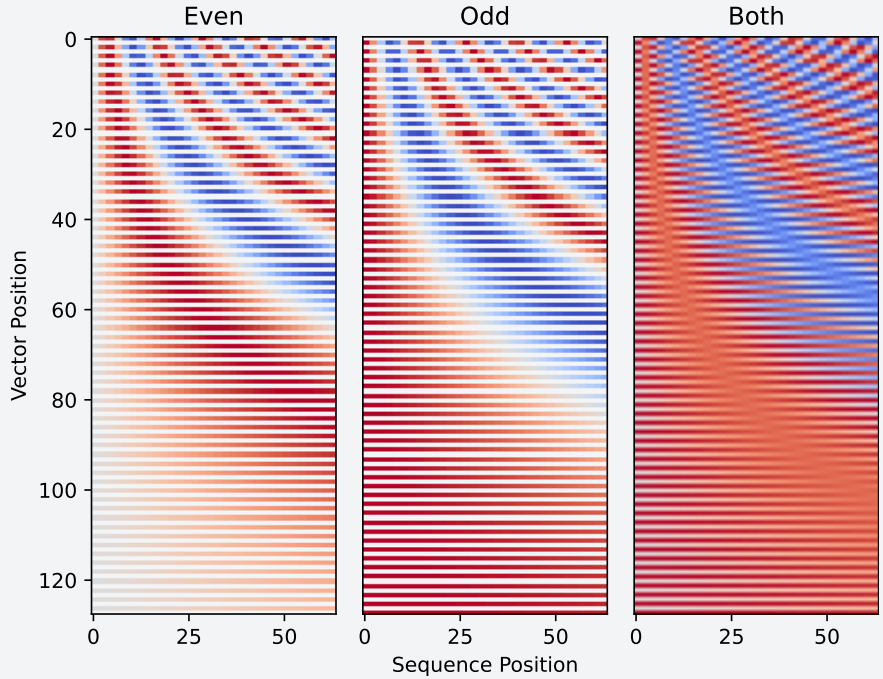
Sinusoidal Position Encoding

Positional Encoding

Proposal 3: use sinusoidal position encoding

1. Deterministic [✓]
2. Distinct at all time-steps [✓]
3. Extends to different sequence lengths [✓]
4. Encodes relative positions [✓]
5. Elegant [✓]
6. Optimal [???]
7. Necessary [???]

See this HF blog for more good ideas



Tokenization

Training Transformers

The End