# Deep Learning

## 15 Self-Attention & Transformers - I

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2024

# Motivation

1. Why does one need to think beyond LSTMs?

# Motivation



1. Why does one need to think beyond LSTMs?
2. Sequential processing doesn't allow parallelization
    - Path length $= \mathbb{O}(\mathrm{n})$
    - RNNs need $\mathbb{O}(\mathrm{n})$ steps to process a sentence of length $n$

# Motivation

1. (Despite the LSTM/GRU) RNNs need attention to deal with long-range dependencies

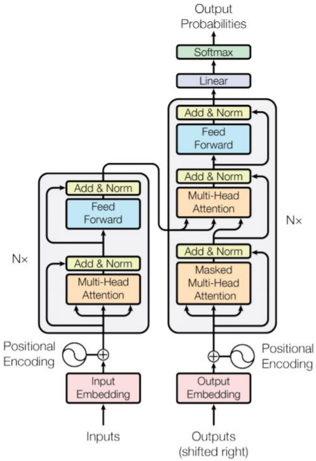
# Motivation



1. (Despite the LSTM/GRU) RNNs need attention to deal with long-range dependencies
2. Since attention enables access to any state, do we need RNNs?

# Transformers

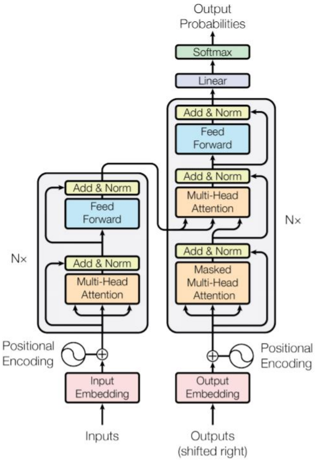
1. Introduced by Vaswani et al. NeurIPS 2017



# Transformers

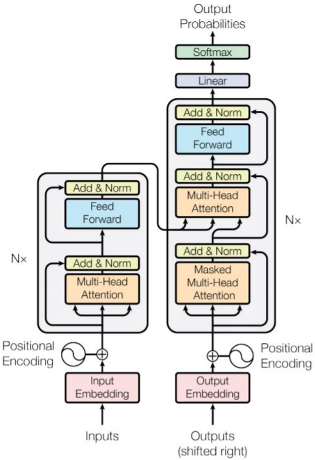


1. Introduced by Vaswani et al. NeurIPS 2017
2. Sequnce to sequence modeling without RNNs



# Transformers

1. Introduced by Vaswani et al. NeurIPS 2017
2. Sequnce to sequence modeling without RNNs
3. Transformer model is built on self-attention (no recurrence or convolutions)

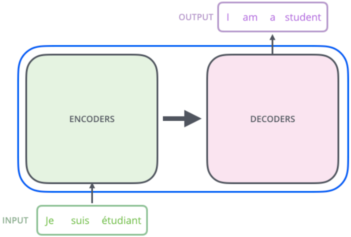


# Transformers


Credits: Jay Alammar

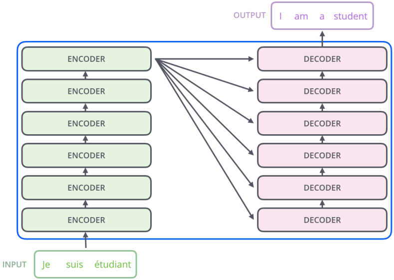
# Transformers


Credits: Jay Alammar




Credits: Jay Alammar

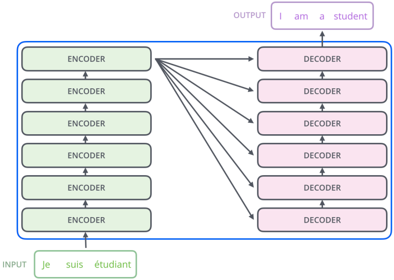
# Transformers




Credits: Jay Alammar

1. The Encoding module has a stack of encoders

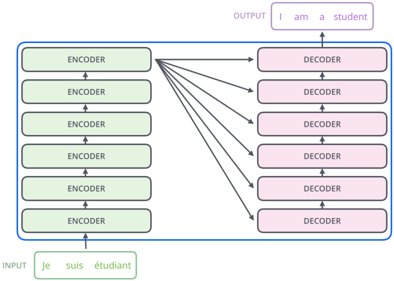
# Transformers




Credits: Jay Alammar

1. The Encoding module has a stack of encoders
2. Same structure different parameters

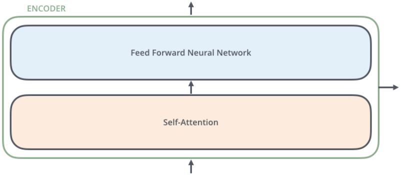
# Transformers




Credits: Jay Alammar

1. The Encoding module has a stack of encoders
2. Same structure different parameters
3. Similarly, the decoding module

# Transformers

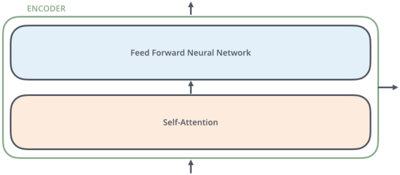



Credits: Jay Alammar

1. Encoder first has a self-attention layer

# Transformers

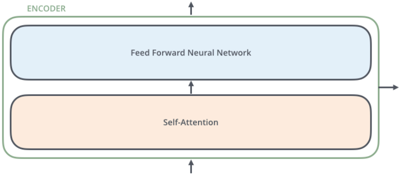





Credits: Jay Alammar

1. Encoder first has a self-attention layer
2. Looks at the other words while encoding a specific word

# Transformers




Credits: Jay Alammar

1. Encoder first has a self-attention layer
2. Looks at the other words while encoding a specific word
3. Next a (same) feed-forward NN is applied at all positions

# Attention vs Self-attention

1. Encoder-Decoder Attention
   looks

# Attention vs Self-attention

1. Encoder-Decoder Attention looks
2. **From**: a decoder (current) state

# Attention vs Self-attention

1. Encoder-Decoder Attention looks
2. **From**: a decoder (current) state
3. **To**: all the encoder states

# Attention vs Self-attention



1. Encoder-Decoder Attention looks
2. **From**: a decoder (current) state
3. **To**: all the encoder states

1. Self-Attention looks

# Attention vs Self-attention

1. Encoder-Decoder Attention looks
2. **From**: a decoder (current) state
3. **To**: all the encoder states

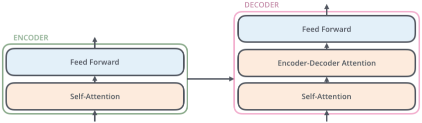
1. Self-Attention looks
2. **From**: each state from a set of states

# Attention vs Self-attention

1. Encoder-Decoder Attention looks
2. **From**: a decoder (current) state
3. **To**: all the encoder states

1. Self-Attention looks
2. **From**: each state from a set of states
3. **To**: all other states in the same set

# Transformers

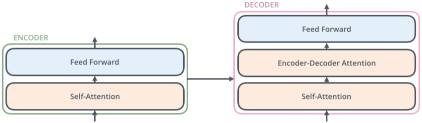



Credits: Jay Alammar

1. Decoder has both the layers (self-attention and shared feed-forward)

# Transformers






Credits: Jay Alammar

1. Decoder has both the layers (self-attention and shared feed-forward)
2. But, in the middle it has an **encoder-decoder attention layer**

# Why the name 'Transformer'?

1. Transforms a set of vectors in some representation space into a corresponding set of vectors (same dimensionality) in some new space

# Why the name 'Transformer'?

1. Transforms a set of vectors in some representation space into a corresponding set of vectors (same dimensionality) in some new space
2. Goal: new space will have a richer internal representation that is better suited to solve the downstream task

# Transformers-Encoding

1. Start with turning each word into a vector at the bottom-most encoder


Credits: Jay Alammar

# Transformers-Encoding

1. Start with turning each word into a vector at the bottom-most encoder
2. Others receive a list of vectors from the encoder immediately below


Credits: Jay Alammar

# Self-attention in Encoder vs. Decoder



1. Who is doing: all source tokens

# Self-attention in Encoder vs. Decoder



1. Who is doing: all source tokens
2. What are they doing (repeat)
    - look at each other
    - update representations

# Self-attention in Encoder vs. Decoder



1. Who is doing: all source tokens
2. What are they doing (repeat)
   - look at each other
   - update representations

1. Decoder

# Self-attention in Encoder vs. Decoder



1. Who is doing: all source tokens
2. What are they doing (repeat)
   - look at each other
   - update representations

1. Decoder
2. Who is doing: target token at each time step

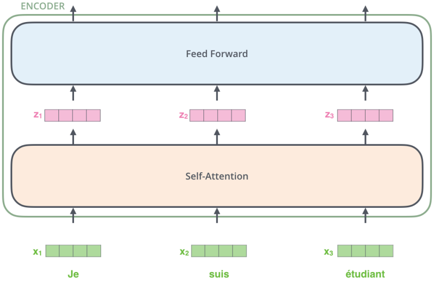
# Self-attention in Encoder vs. Decoder

1. Who is doing: all source tokens
2. What are they doing (repeat)
   - look at each other
   - update representations

1. Decoder
2. Who is doing: target token at each time step
3. What are they doing (repeat)
   - looks at previous target tokens (self-attention)
   - looks at source representations (encoder-decoder attention)
   - update representation

# Transformers-Encoding

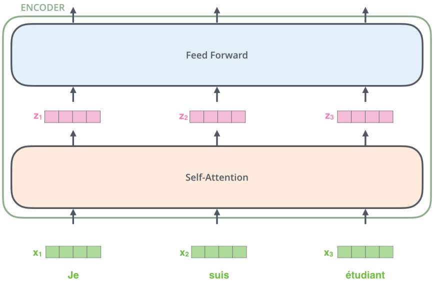
1. Each word flows through the two layers of the encoder through its own path




Credits: Jay Alammar

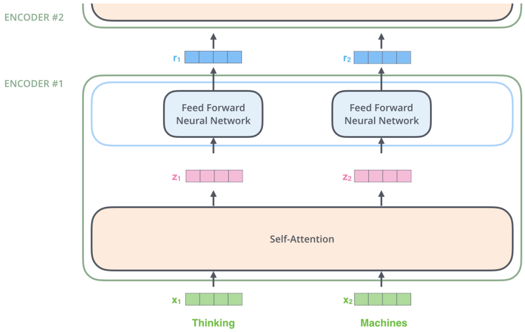
# Transformers-Encoding

1. Each word flows through the two layers of the encoder through its own path
2. Self-attention layer has dependencies among them. However, the path length is $\mathbb{O}(1)$




Credits: Jay Alammar

# Transformers-Encoding




Credits: Jay Alammar

# Self-Attention

1. The animal didn't cross the street because it was too tired
2. The animal didn't cross the street because it was too wide

# Self-Attention



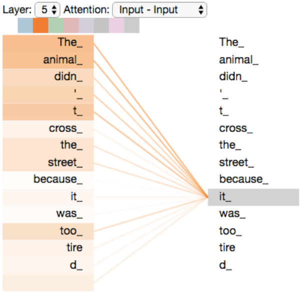
1. `The animal didn't cross the street because it was too tired`
2. `The animal didn't cross the street because it was too wide`
3. What does 'it' refer to?

# Self-Attention

1. `The animal didn't cross the street because it was too tired`
2. `The animal didn't cross the street because it was too wide`
3. What does 'it' refer to?
4. Easy for humans, but not so much for the traditional Seq2Seq models

# Self-Attention

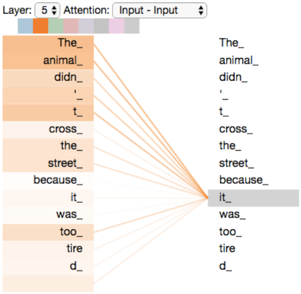
1. As the model processes each word, self-attention attends other positions in the i/p sequence to encode better




Credits: Jay Alammar

# Self-Attention

1. As the model processes each word, self-attention attends other positions in the i/p sequence to encode better
2. Unlike RNNs, we don't keep hidden states from previous positions here!




Credits: Jay Alammar

# Attention weights

1. Input tokens $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_N}$

# Attention weights



1. Input tokens $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_N}$
2. Output tokens $\mathbf{y_1}, \mathbf{y_2}, \ldots \mathbf{y_N}$

# Attention weights



1. Input tokens $\mathbf{x_1, x_2, \ldots x_N}$
2. Output tokens $\mathbf{y_1, y_2, \ldots y_N}$
3. $\mathbf{y_n = \sum_{m=1}^{N} a_{nm} \cdot x_m}$

# Attention weights

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Input tokens $\mathbf{x_1, x_2, \ldots x_N}$
2. Output tokens $\mathbf{y_1, y_2, \ldots y_N}$
3. $\mathbf{y_n = \sum_{m=1}^{N} a_{nm} \cdot x_m}$
4. $a_{mn} \geq 0$ and $\sum_{m=1}^{N} a_{mn} = 1$ Why?

# How to compute the Attention weights?

1. A simple way is to use the 'dot product' self-attention

# How to compute the Attention weights?

1. A simple way is to use the 'dot product' self-attention
2. $a_{nm} = \frac{exp(\mathbf{x_n^T x_m})}{\sum_{m'=1}^{N} exp(\mathbf{x_n^T x'_m})}$

# How to compute the Attention weights?



1. A simple way is to use the 'dot product' self-attention
2. $a_{nm} = \frac{exp(\mathbf{x_n^T x_m})}{\sum_{m'=1}^{N} exp(\mathbf{x_n^T x'_m})}$
3. $\mathbf{Y} = \text{Softmax}[\mathbf{X X^T}]\mathbf{X}$

# How to compute the Attention weights?

1. A simple way is to use the 'dot product' self-attention
2. $a_{nm} = \frac{exp(\mathbf{x_n^T x_m})}{\sum_{m'=1}^{N} exp(\mathbf{x_n^T x'_m})}$
3. $\mathbf{Y} = \text{Softmax}[\mathbf{X}\mathbf{X}^\mathbf{T}]\mathbf{X}$
4. The transformation from $\mathbf{X}$ to $\mathbf{Y}$ is fixed; has no capacity to learn from the data

# How to compute the Attention weights?

1. A simple way is to use the 'dot product' self-attention
2. $a_{nm} = \frac{exp(\mathbf{x_n^T x_m})}{\sum_{m'=1}^{N} exp(\mathbf{x_n^T x'_m})}$
3. $\mathbf{Y} = \text{Softmax}[\mathbf{X}\mathbf{X}^\mathbf{T}]\mathbf{X}$
4. The transformation from $\mathbf{X}$ to $\mathbf{Y}$ is fixed; has no capacity to learn from the data
5. Each of the feature values in a token plays an equal role in determining the attention weights

# Some terminology from Information Retrieval



1. Consider a streaming platform and the problem of choosing which movie to watch

# Some terminology from Information Retrieval

1. Consider a streaming platform and the problem of choosing which movie to watch
2. One approach would be

# Some terminology from Information Retrieval

1. Consider a streaming platform and the problem of choosing which movie to watch
2. One approach would be
3. Associate each movie with a list of attributes (genre, actors, technicians, length, year, etc.) $\rightarrow$ Key

# Some terminology from Information Retrieval

1. Consider a streaming platform and the problem of choosing which movie to watch
2. One approach would be
3. Associate each movie with a list of attributes (genre, actors, technicians, length, year, etc.) $\rightarrow$ Key
4. These form a catalog of movies to be searched for

# Some terminology from Information Retrieval

1. Consider a streaming platform and the problem of choosing which movie to watch
2. One approach would be
3. Associate each movie with a list of attributes (genre, actors, technicians, length, year, etc.) $\rightarrow$ Key
4. These form a catalog of movies to be searched for
5. The movie file itself is the Value

# Some terminology from Information Retrieval

1. Consider a streaming platform and the problem of choosing which movie to watch
2. One approach would be
3. Associate each movie with a list of attributes (genre, actors, technicians, length, year, etc.) $\rightarrow$ Key
4. These form a catalog of movies to be searched for
5. The movie file itself is the Value
6. User's input of desired attributes $\rightarrow$ Query

# Self-Attention

1. $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathbf{q})}$

# Self-Attention

1. $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathbf{q})}$
2. $\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathbf{k})}$

# Self-Attention



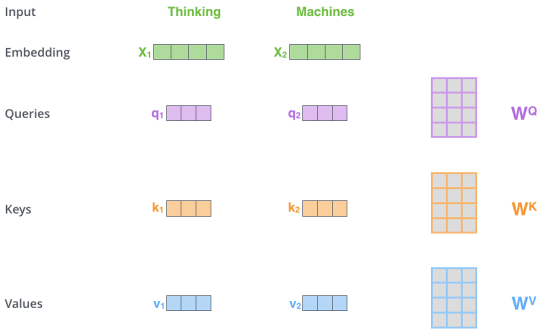
1. $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathbf{q})}$
2. $\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathbf{k})}$
3. $\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathbf{v})}$

# Self-Attention

1. $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathbf{q})}$
2. $\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathbf{k})}$
3. $\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathbf{v})}$
4. $\mathbf{Y} = \text{Softmax}[\mathbf{Q}\mathbf{K}^{\mathbf{T}}]\mathbf{V}$

# Self-Attention

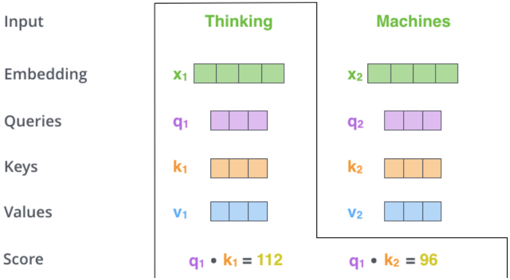
భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad




Credits: Jay Alammar

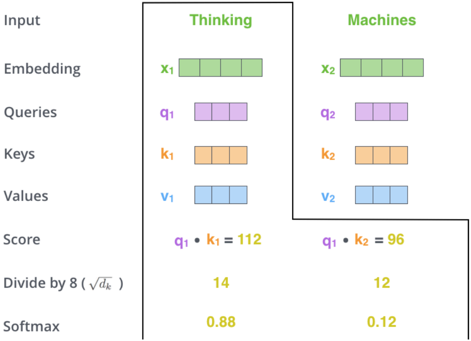
# Self-Attention





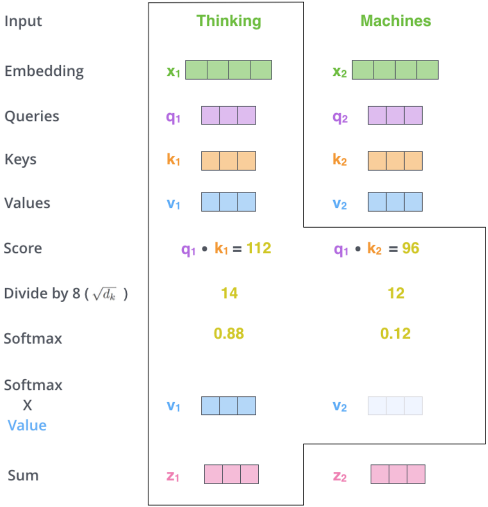

Credits: Jay Alammar

# Self-Attention



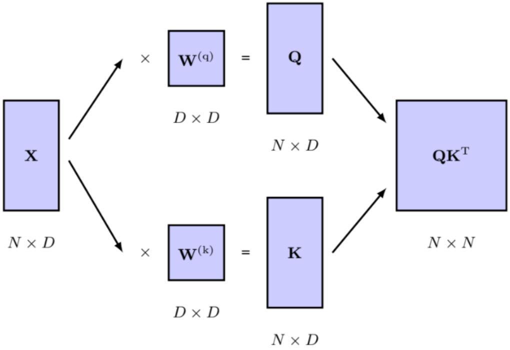

Credits: Jay Alammar

# Self-Attention



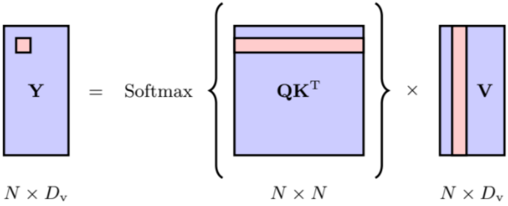

Credits: Jay Alammar

# Self-Attention




Credits: The Bishop's book

# Self-Attention




Credits: The Bishop's book

# Scaled Self-Attention



1. Gradients of the softmax become exponentially small for large input magnitudes

# Scaled Self-Attention

1. Gradients of the softmax become exponentially small for large input magnitudes
2. To prevent this, the $\mathbf{QK^T}$ is scaled before the softmax

# Scaled Self-Attention

1. Gradients of the softmax become exponentially small for large input magnitudes
2. To prevent this, the $\mathbf{QK^T}$ is scaled before the softmax
3. If the elements of $q$ and $v$ vectors are independent $N(0,1)$ distributed, the variance of the dot product $\rightarrow D_k$

# Scaled Self-Attention



1. Gradients of the softmax become exponentially small for large input magnitudes
2. To prevent this, the $\mathbf{QK^T}$ is scaled before the softmax
3. If the elements of $q$ and $v$ vectors are independent $N(0,1)$ distributed, the variance of the dot product $\rightarrow D_k$
4. Hence, normalize the product by the standard deviation
   $\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}[\frac{\mathbf{QK^T}}{\sqrt{D_k}}]\mathbf{V}$

# Multi-headed Self-Attention

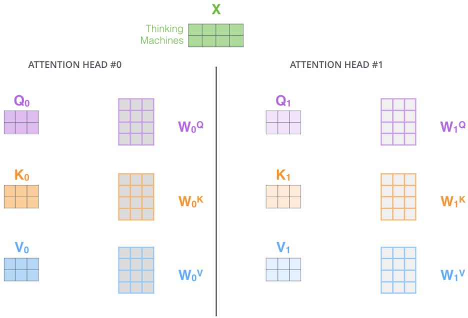
1. There may be multiple patterns of attention that are relevant at the same time

# Multi-headed Self-Attention



1. There may be multiple patterns of attention that are relevant at the same time
2. E.g., some patterns relevant to the ‘tense’ while others might be associated with the ‘vocabulary.’

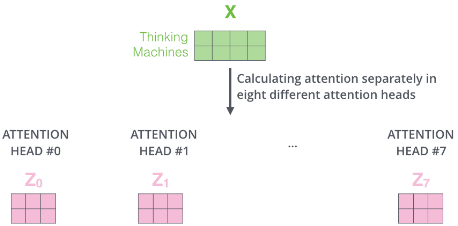
# Multi-headed Self-Attention




Credits: Jay Alammar

# Multi-headed Self-Attention






Credits: Jay Alammar

# Multi-headed Self-Attention



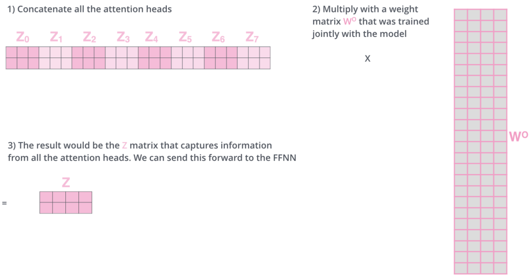
1. Expands the model's ability to focus on different relevant positions in the i/p

# Multi-headed Self-Attention



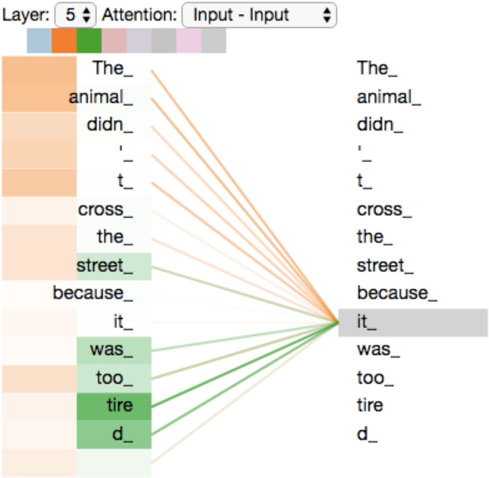
1. Expands the model's ability to focus on different relevant positions in the i/p
2. Enables different 'representational subspace'

# Multi-headed Self-Attention




Credits: Jay Alammar

# Multi-headed Self-Attention




Credits: Jay Alammar

# Transformer Layers

1. Neural nets benefit greatly from the depth

# Transformer Layers

1. Neural nets benefit greatly from the depth
2. $\rightarrow$ stack multiple self-attention layers

# Transformer Layers



1. Neural nets benefit greatly from the depth
2. $\rightarrow$ stack multiple self-attention layers
3. To improve the training efficiency, introduce residual connections (requires to maintain the dimensionality)

# Transformer Layers

1. Neural nets benefit greatly from the depth
2. $\rightarrow$ stack multiple self-attention layers
3. To improve the training efficiency, introduce residual connections (requires to maintain the dimensionality)
4. Followed by Layer normalization
   $\mathbf{Z} = \text{LayerNorm}[\mathbf{Y}(\mathbf{X}) + \mathbf{X}]$

# Transformer Layers



1. Output vectors are constrained to lie in the subspace spanned by the i/p vectors

# Transformer Layers

1. Output vectors are constrained to lie in the subspace spanned by the i/p vectors
2. Enhance the expressive capability/flexibility by post-processing using a nonlinear neural net (MLP)

# Transformer Layers



1. Output vectors are constrained to lie in the subspace spanned by the i/p vectors
2. Enhance the expressive capability/flexibility by post-processing using a nonlinear neural net (MLP)
3. This should not affect the transformer's ability to process variable length i/p

# Transformer Layers

1. Output vectors are constrained to lie in the subspace spanned by the i/p vectors
2. Enhance the expressive capability/flexibility by post-processing using a nonlinear neural net (MLP)
3. This should not affect the transformer's ability to process variable length i/p
4. Same share net applies to all the o/p tokens (followed by residual connection and normalization)
   $\tilde{\mathbf{X}} = \textbf{LayerNorm}[\text{MLP}[\mathbf{Z}] + \mathbf{Z}]$