Transformers #1: Introduction to Transformer

RPL CV/DL reading group

Sebastian Bujwid

23 Feb 2021

Transformers theme: schudule

- 23 Feb (today) (Sebastian): Introduction to the Transformer model
- 9 Mar (Federico): Transformer models for Image
- (tentative) 23 Mar (Yonk): Transformer models in different applications/domains
- (tentative) 6 Apr (Sofia): Alternative approaches to Transformer

Agenda

1. Taxonomy: attention, self-attention, Transformer?
   - Why attention?
   - Why Transformer?
   - **Discussion!**
2. Transformer
   - Transformer vs. RNN; Transformer in general
   - Some specifics for text
   - **Discussion!**
3. Some details about the Transformer & results
4. **Discussion!**

Taxonomy

Attention! - definition

"Attention is a technique that mimics cognitive attention" (Wikipedia)

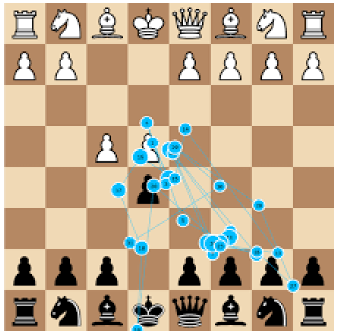
"(Your) Attention is (our) profit" (Instagram) ⚠ Offical sources might deny it.

"The ability to focus on one thing and ignore others" (Alex Graves, 2020) - I might have taken it out of context :/

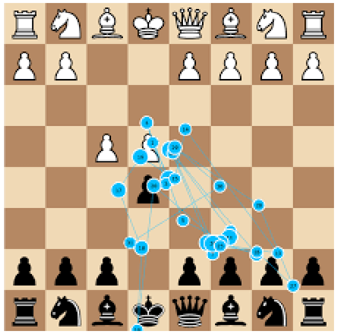
"**Looking at some places more than at others**" (Bujwid, 2021)

Why attention (in neural networks)?

Why attention (in neural networks)?



Why attention (in neural networks)?



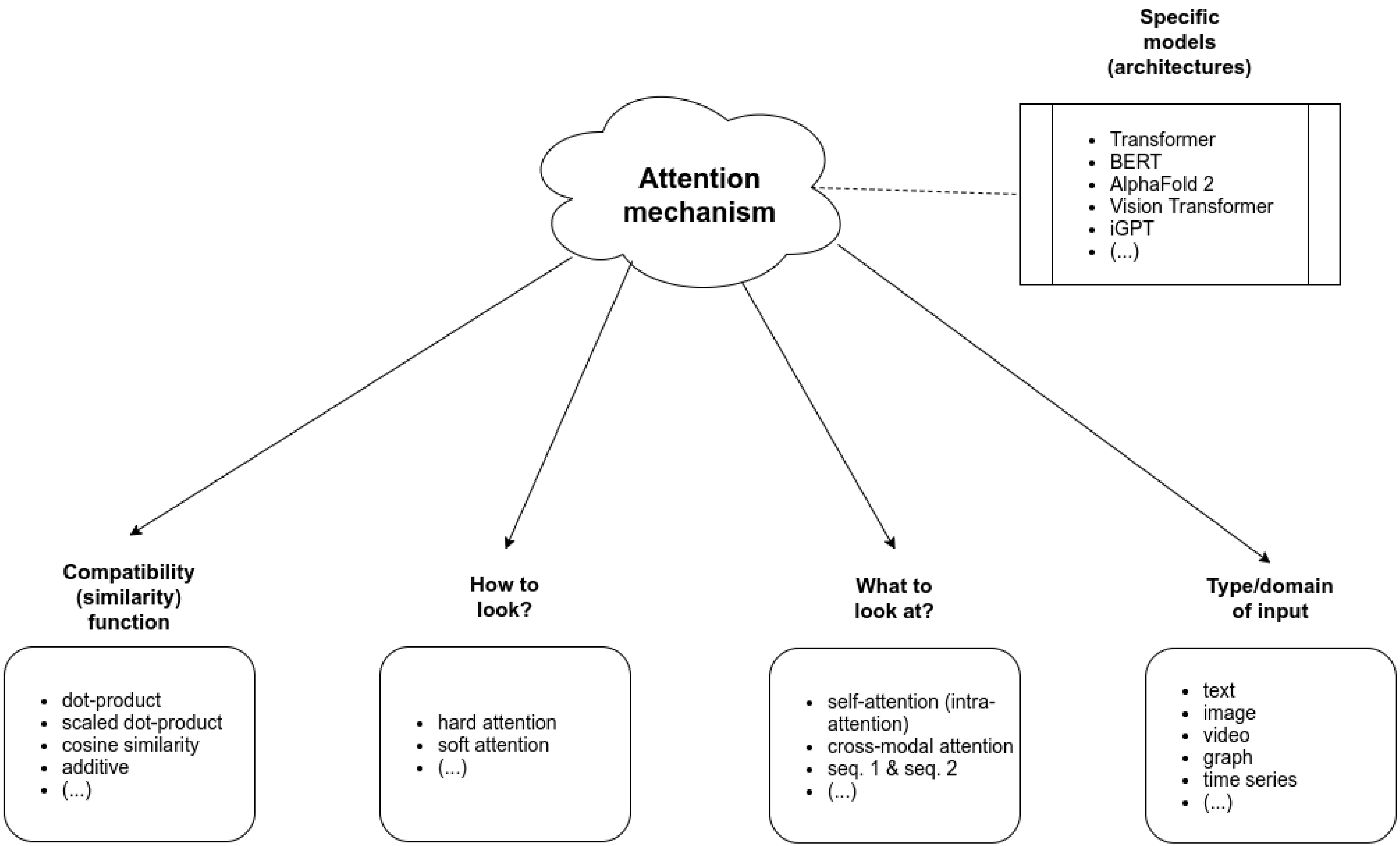
Anna är orolig för miljön så hon använder sin cykel året runt

(...) (...)

Anna is concerned about the environment so she uses ? bike all year round

(her)

Taxonomy

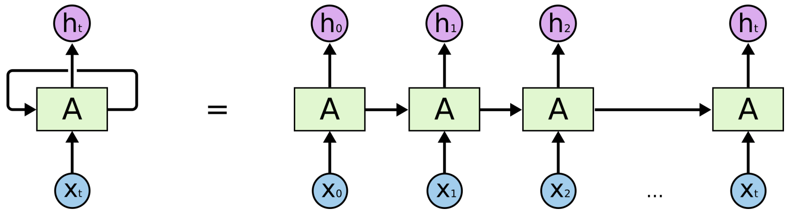

Specific models (architectures)
Transformer
BERT
AlphaFold 2
Vision Transformer
iGPT
(...)
Attention mechanism
Compatibility (similarity) function
dot-product
scaled dot-product
cosine similarity
additive
(...)
How to look?
hard attention
soft attention
(...)
What to look at?
self-attention (intra-attention)
cross-modal attention
seq. 1 & seq. 2
(...)
Type/domain of input
text
image
video
graph
time series
(...)

Why Transformer [1]?

Let's look at the issues with RNNs first!

Issue #1: Computational efficiency

This structure is not efficient on GPUs!



(Image from: http://colah.github.io/)

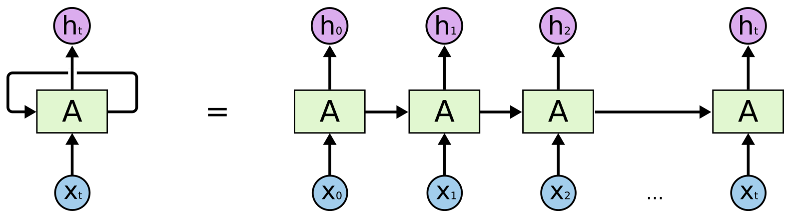
[1] Vaswani et al., "Attention Is All You Need," NeurIPS 2017

Why Transformer [1]?

Let's look at the issues with RNNs first!

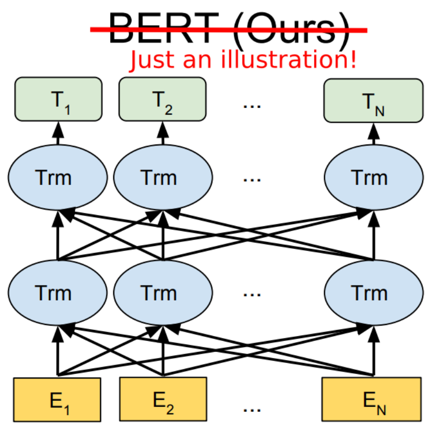
Issue #1: Computational efficiency

This structure is not efficient on GPUs!



(Image from: http://colah.github.io/)

Solution #1: An architecture without sequential operations



[1] Vaswani et al., "Attention Is All You Need," NeurIPS 2017

Issue #2: Long-term dependencies

Issue #2: Long-term dependencies

sv: Viskleken | *eng:* Telephone (game)

Issue #2: Long-term dependencies

sv: Viskleken | *eng:* Telephone (game)

Our unpublished results:

Doesn't work if too many kids.
(Sebastian, 199...)

Issue #2: Long-term dependencies

sv: Viskleken | *eng:* Telephone (game)

Our unpublished results:

Doesn't work if too many kids.
(Sebastian, 199...)

Solution #2: Always look at the whole sequence (or rather sub-sequence)

Transformer

Transformer: scaled dot-product self-attention

T - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values

Transformer: scaled dot-product self-attention

T - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values

Self-attention when all Q, K, and V are a function of the same input sequence X.

- $Q = f_q(X)$, $K = f_k(X)$, $V = f_v(X)$
- Typically all linear functions

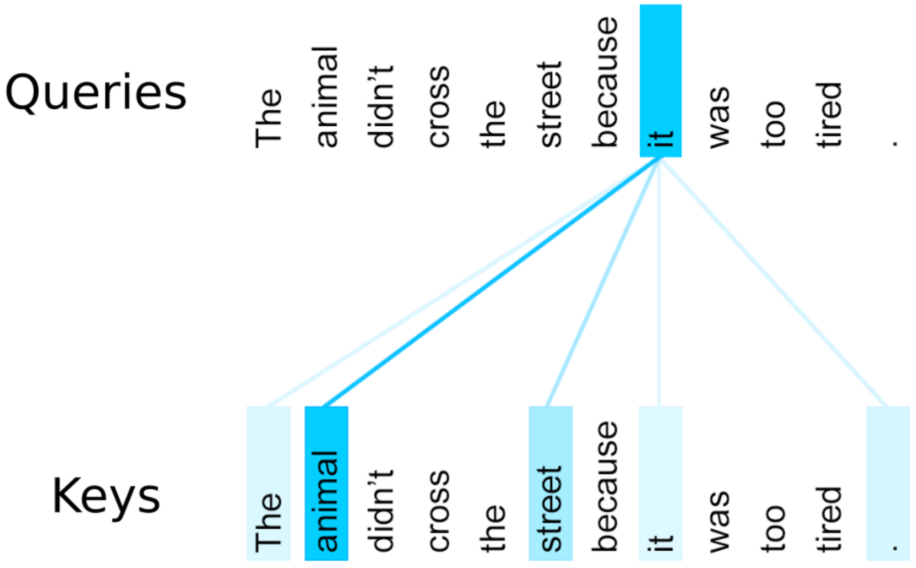
Transformer: scaled dot-product self-attention

T - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values

For each token (query!) compute the "compatibility" with other tokens (keys!):

$$\frac{QK^T}{\sqrt{d}} \qquad [\mathbb{R}^{T \times T}]$$

- Or any other $f(Q, K) \to \mathbb{R}^{T \times T}$

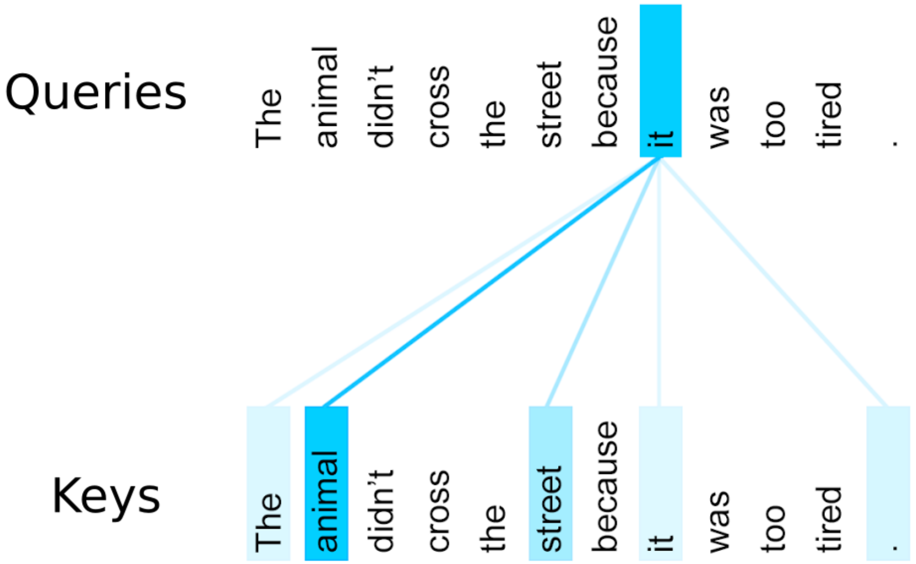


Transformer: scaled dot-product self-attention

T - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values

Normalize the "compatibility" for each query

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

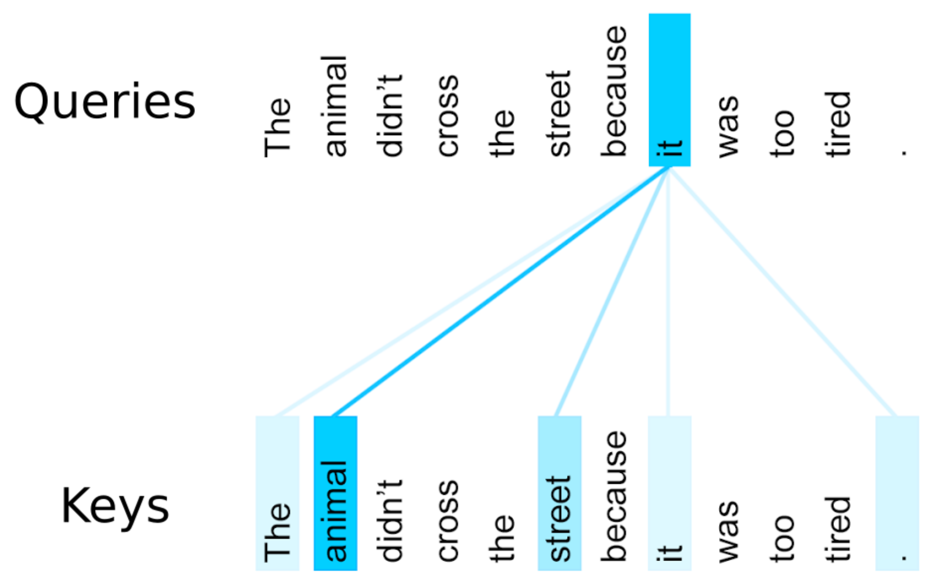


Transformer: scaled dot-product self-attention

T - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values

Compute new values:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V$$



Transformer: architecture

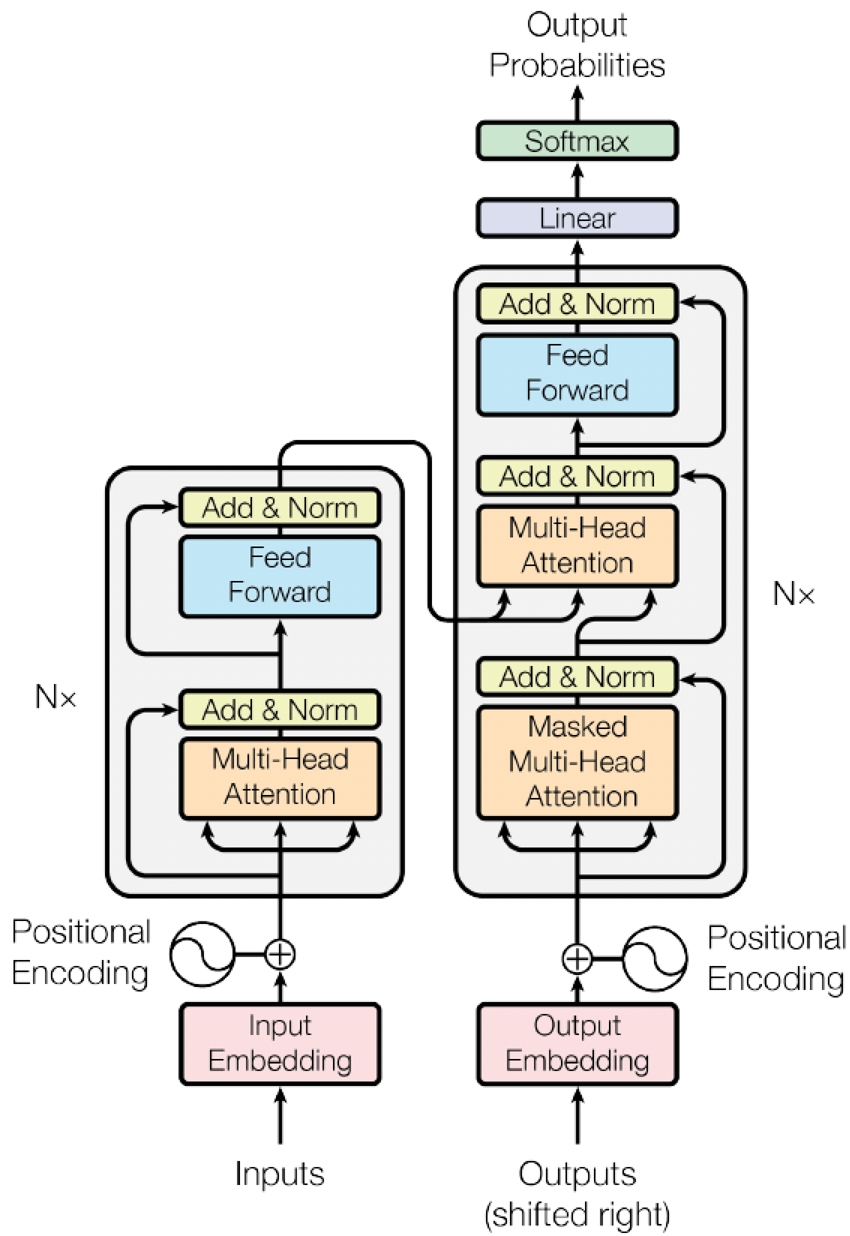


Figure 1: The Transformer - model architecture.

Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

track do the we ? keep of How order

Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

track do the we ? keep of How order

- Input encoded positions, e.g. based on the `sin` function
- Learn input position embeddings

Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

track do the we ? keep of How order

- Input encoded positions, e.g. based on the `sin` function
- Learn input position embeddings

Multi-head attention

- More effective (controlling for #params)

Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

track do the we ? keep of How order

- Input encoded positions, e.g. based on the `sin` function
- Learn input position embeddings

Multi-head attention

- More effective (controlling for #params)

Interpretability?

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Thank you!

Discussion!