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
Introduction/Preview

- Area of expertise: language modeling and machine translation
- Previously solved by RNNs
- Transformer: a network with no recurrence
Why bother?

- RNNs had many problems:
  - Hard to parallelize
  - Difficulty to learn long-range dependencies
- Transformer’s solution to these problems: self-attention
- Outcome: state of the art performance on English-to-German and English-to-French translation
- Faster training time
Neural Machine Translation

- Map a sequence to another sequence
- **encoder-decoder**
- Encoder takes sequence of words -> converts it to intermediate representation (fixed length vector) -> decoder decodes it to output sequence
RNN example
Shortcomings of RNN

- Works great with short sentences, but takes a hit with longer sentences
- Intermediate representation needs to hold all the information
- **decoder** needs different information at different timesteps
The problem of long sequences

Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.
Solution: attention

- Attention let’s decoder to look back
- It provides decoder access to all encoders hidden states
- Decoder gets to choose which hidden states are more relevant
- (But it still is hard to parallelize!)
Attention model intuition

[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]
Attention model intuition

[Bahdanau et. al., 2014, Neural machine translation by jointly learning to align and translate]
Attention model intuition

[Bahdanau et al., 2014. Neural machine translation by jointly learning to align and translate]
Attention model intuition

[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]
Attention model

[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]
Attention model

\[ a^{(t,t')} = \text{amount of "attention" } y^{(t')} \text{ should pay to } a^{(t)} \]

\[ a^{(t,t')} = (a^{(t,t')}, a^{(t,t')}) \]

\[ \sum_{t'} a^{(t,t')} = 1 \]

\[ c^{(t,t')} = \sum_{t'} a^{(t,t')} a^{(t,t')} \]

[Bahdanau et al., 2014. Neural machine translation by jointly learning to align and translate]
Computing attention $\alpha^{<t,t'>}$

$$\alpha^{<t,t'>} = \text{amount of attention } y^{<t>} \text{ should pay to } \alpha^{<t'>}$$

$$\Rightarrow \alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

[Xu et. al., 2015. Show attention and tell: neural image caption generation with visual attention]
Dependencies is Neural Machine Translation

- Dependencies between:
  a. input and output tokens
  b. input tokens
  c. output tokens

- RNN + Attention only focused on (a)
Self-Attention (intra-attention)

- Relates different positions of a single sequence
- Constant path length between any two positions
Three ways of attention

Encoder-Decoder Attention

Encoder Self-Attention

Masked Decoder Self-Attention
The Transformer - model architecture

- encoder - decoder
- No recurrent layers

Figure 1: The Transformer - model architecture.
Multi-Head Attention

Attention\( (Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \)
Self-Attention

Multi-Head Attention
Encoder

- Self-attention layer, where all the keys, values and queries come from output of previous layer
- Each position in the encoder can attend to all positions in the previous layer of the encoder
Decoder

- self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position
## Complexity

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity per Layer</th>
<th>Sequential Operations</th>
<th>Maximum Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(n^2 \cdot d)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Recurrent</td>
<td>$O(n \cdot d^2)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>$O(k \cdot n \cdot d^2)$</td>
<td>$O(1)$</td>
<td>$O(log_k(n))$</td>
</tr>
<tr>
<td>Self-Attention (restricted)</td>
<td>$O(r \cdot n \cdot d)$</td>
<td>$O(1)$</td>
<td>$O(n/r)$</td>
</tr>
</tbody>
</table>

n - sequence length        k - kernel size        d - depth
Training

- Training data
  - English-German dataset: 4.5 million sentence pairs
  - English-French dataset: 36 million sentence pairs

- Training time on 8 NVIDIA P100 GPUs:
  - Base model 100 000 steps = 12h
  - Big model 300 000 steps = 3.5 days

- Adam optimizer with varied learning rate: increased learning rate for 4000 \textit{warmup_steps} and then decreased it proportionally

- Residual Dropout and Label Smoothing

<table>
<thead>
<tr>
<th></th>
<th>$N$</th>
<th>$d_{model}$</th>
<th>$d_{ff}$</th>
<th>$h$</th>
<th>$d_k$</th>
<th>$d_v$</th>
<th>$P_{drop}$</th>
<th>$\epsilon_{ls}$</th>
<th>train steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
<td>512</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>100K</td>
</tr>
<tr>
<td>big</td>
<td>6</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td>64</td>
<td></td>
<td>0.3</td>
<td>0.1</td>
<td>300K</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td><strong>BLEU</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ByteNet [15]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [32]</td>
<td>24.6</td>
<td>1.0 \cdot 10^{20}</td>
</tr>
<tr>
<td>GNMT + RL [31]</td>
<td>25.16</td>
<td>9.6 \cdot 10^{18}</td>
</tr>
<tr>
<td>ConvS2S [8]</td>
<td>26.03</td>
<td>2.0 \cdot 10^{19}</td>
</tr>
<tr>
<td>MoE [26]</td>
<td>26.03</td>
<td>1.2 \cdot 10^{20}</td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [32]</td>
<td>26.30</td>
<td>1.8 \cdot 10^{20}</td>
</tr>
<tr>
<td>GNMT + RL Ensemble [31]</td>
<td>26.36</td>
<td>7.7 \cdot 10^{19}</td>
</tr>
<tr>
<td>ConvS2S Ensemble [8]</td>
<td>26.36</td>
<td>1.2 \cdot 10^{21}</td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>3.3 \cdot 10^{18}</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>28.4</td>
<td>2.3 \cdot 10^{19}</td>
</tr>
</tbody>
</table>
Other fields

- Example of Wikipedia article generated by Transformer