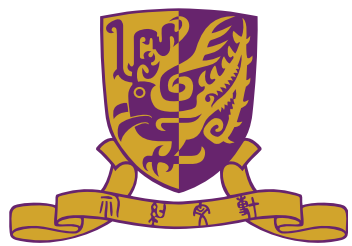
DDA5001 Machine Learning

Transformer (Part I)

Xiao Li

School of Data Science
The Chinese University of Hong Kong, Shenzhen



"Attention is All You Need"

- Transformer is one special but one of the most important architectures of deep learning.
- People come to know it widely because of the following paper:

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 translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Transformer is The Foundation of Language Models

"The transformer, today's dominant AI architecture, ...has gone on to revolutionize the field of AI over the past half-decade ..."
— Rob Toews, Contributor. Sep. 3, 2023. Forbes Magazine.

- Contemporary big AI models (large language models) mostly use transformer architecture.

Attention

Attention in Transformer

- Transformer is inherently designed for natural language processing, but now is also used for vision tasks.
- The key ingredient of a transformer is attention, which captures the semantic structure of a sentence.

Before introducing the mechanism of attention, we first introduce some basic notions of transformer.

Basic Notions

- The input data to a transformer $\{\boldsymbol{x}_i\}_{i=1}^n$ with each $\boldsymbol{x}_i \in \mathbb{R}^d$ are called tokens.

Examples:

- For a sentence like

  `It is raining.`

  A token can be a word in this sentence such as "It". It is transferred to a vector $\{\boldsymbol{x}_i\}_{i=1}^n$ by embedding.
- n is called sequence length / context window, i.e., number of tokens in the sentence. d is embedding dimension.

One can regard $\boldsymbol{x}_i$ as the usual training data point. We call it token instead of training data. Consequently, x_{ij} is the j-th feature of token $\boldsymbol{x}_i$.

In transformers, the data matrix $\boldsymbol{X}$ is constructed in a row-wise manner:

$$\boldsymbol{X} = \left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\right]^\top \in \mathbb{R}^{n \times d}.$$

Basically, you can regard $\boldsymbol{X}$ as one sentence or one paragraph.

Attention: A Linear Combination

Core idea of attention: Transferring the set of input tokens $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ to the transformed tokens $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$, so that $\{\boldsymbol{y}_i\}$ contains richer semantic structure.

To perform such a transform, we impose that $\boldsymbol{y}_i$ should not only depend on $\boldsymbol{x}_i$ but the whole set of input tokens $\{\boldsymbol{x}_i\}$ through a linear relationship:

$$\boldsymbol{y}_i = \sum_{j=1}^{n} a_{ij} \boldsymbol{x}_j.$$

Here, $\{a_{ij}\}$ are called attention coefficients. We often impose a simplex structure to $\{a_{ij}\}$, i.e.,

$$a_{ij} \geq 0 \ (\forall i, j), \quad \sum_{j=1}^{n} a_{ij} = 1 \ (\forall i).$$

$\rightsquigarrow \boldsymbol{y}_i$ is calculated by all $\{\boldsymbol{x}_j\}$, and their importance is based on a_{ij}.

Why Attention?

- It is easy to see that attention is to mix all the token information $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ to generate $\boldsymbol{y}_i$.
- This is because one word is not fully meaningful in a sentence. We often need to understand the whole semantic meaning, and then understand the meaning of a word (token).

Let us consider the following two sentences

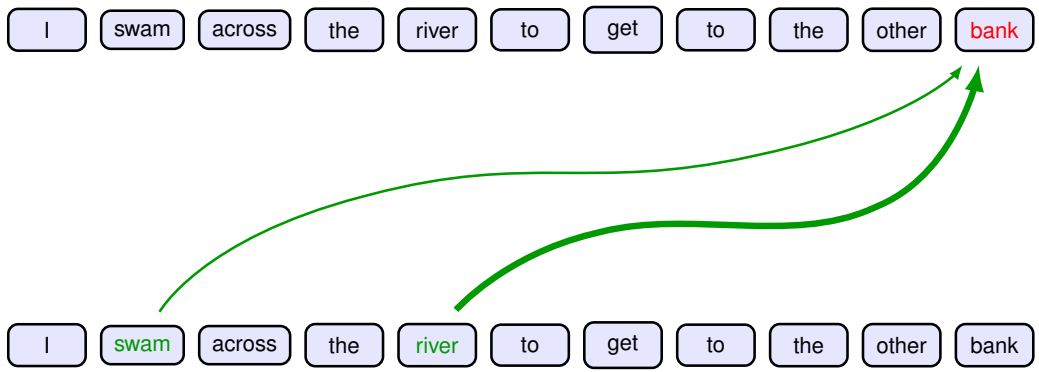
I swam across the river to get to the other bank.
I walked across the road to get cash from the bank.

Suppose the the word "bank" is a token.

- It is the same word "bank" in both two sentences. Do they have the same meaning? Obviously not.
- We need to put the word in the context of the whole sentence, i.e., other words (tokens) also contribute to "bank".

Why Attention?

Hence, attention is to represent the relationship between tokens.



Observations:

- All tokens are related to each other.
- Some tokens are more important to interpret "bank".

$\rightsquigarrow$ Justifies why we need attention mechanism to mix the info of all tokens.

Self-Attention

Self-Attention

However, it is nontrivial to calculate the attention coefficients a_{ij}.

- Since we need to understand the relationship between different tokens.
- We also want to determine the importance of the pairwise relationship. Stronger relation should have more similarity.

This motivates us to compute the similarity using inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$. Then, the attention coefficients can be constructed using softmax:

$$a_{ij} = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{x}_j)}{\sum_{j'=1}^n \exp(\boldsymbol{x}_i^\top \boldsymbol{x}_{j'})}.$$

Such a computed $\{a_{ij}\}$ will satisfy $a_{ij} \geq 0$ $(\forall i, j)$, $\sum_{j=1}^n a_{ij} = 1$ $(\forall i)$.

- If for token $\boldsymbol{x}_i$, we have $a_{ij} > a_{ij'}$ for all other $j' \neq j$, it means $\boldsymbol{x}_j$ is the most important token in the sentence in terms of interpreting token $\boldsymbol{x}_i$.
- This way of determining attention coefficient using tokens themselves is called self-attention.

Self-Attention in Matrix Notation

If we stack the output vectors $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ to be a output matrix $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]^\top.$$

Then, we can have the following matrix form of self-attention

$$\boldsymbol{Y} = \text{softmax}(\boldsymbol{X}\boldsymbol{X}^\top)\boldsymbol{X}.$$

- Here, softmax$(\boldsymbol{L})$ is first take exponential to all entries of $\boldsymbol{L}$.
- Then, it normalized each row sum to one.

Namely,

$$[\text{softmax}(\boldsymbol{X}\boldsymbol{X}^\top)]_{ij} = a_{ij}.$$

Trainable Self-Attention

- The attention mechanism is insightful for capturing semantic information. However, the derived self-attention is fixed and has no trainable parameters.

We have three data matrix $\boldsymbol{X}$ in the self-attention formula. Indeed, we can manually add trainable parameters to each data matrix as

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{(q)},\ \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{(k)},\ \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{(v)}.$$

Here, $\boldsymbol{W}^{(q)} \in \mathbb{R}^{d\times d_q}, \boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d_k}, \boldsymbol{W}^{(v)} \in \mathbb{R}^{d\times d_v}$ are the trainable network parameters.

In this way, the self-attention is generalized to

$$\boldsymbol{Y} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}.$$

In transformer, $\boldsymbol{Q}$ is called query, $\boldsymbol{K}$ is called key, and $\boldsymbol{V}$ is called value. Note that $d_q = d_k$ must be ensured. One typical choice is

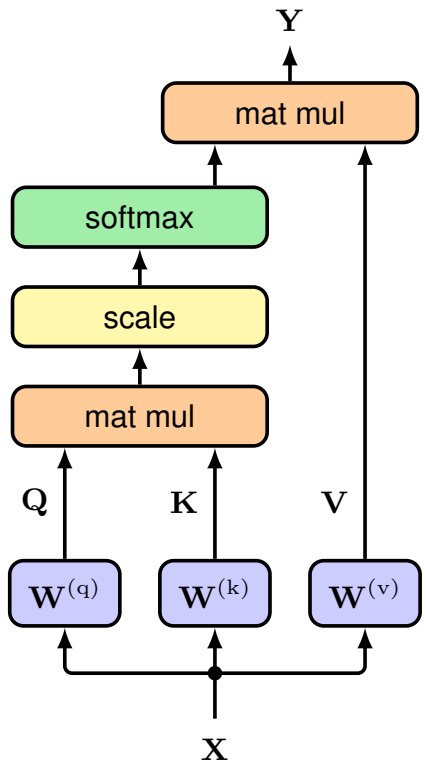
$$d_q = d_k = d_v = d.$$

Scaled Self-Attention

We often re-scale the attention so that gradients of the network will be more stable, giving

$$\boldsymbol{Y} = \text{attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}.$$

Illustration of self-attention:



Multi-head Attention

Multi-head Attention

- There might be different patterns of attention. Some patterns may be related to tense, some may be related to vocabulary, etc.
- If we just use one attention, it may learn an averaged information of these multiple patterns.
- Thus, we need multiple attention heads to learn these multiple patterns. Each attention head has its own query, key, value trainable parameters. $\rightsquigarrow$ multi-head attention.

Suppose we have H heads defined as

$$\boldsymbol{H}_h = \text{attention}(\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h), \quad h = 1, \ldots, H.$$

With

$$\boldsymbol{Q}_h = \boldsymbol{X}\boldsymbol{W}_h^{(q)}, \ \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_h^{(k)}, \ \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_h^{(v)}.$$

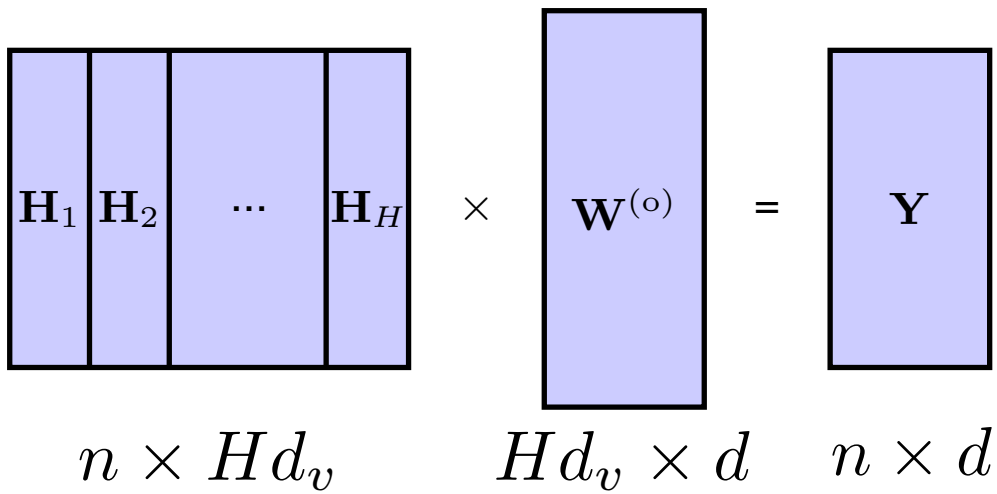
If each $\boldsymbol{W}_h^{(v)}$ has dimension $d \times d_v$, then each $\boldsymbol{H}_h$ has dimension $n \times d_v$.

Multi-head Attention

Then, we can introduce a mixing trainable matrix $\boldsymbol{W}^{(o)}$ to mix all the information as

$$\boxed{\boldsymbol{Y}(\boldsymbol{X}) = [\boldsymbol{H}_1, \ldots, \boldsymbol{H}_H]\boldsymbol{W}^{(o)}.}$$

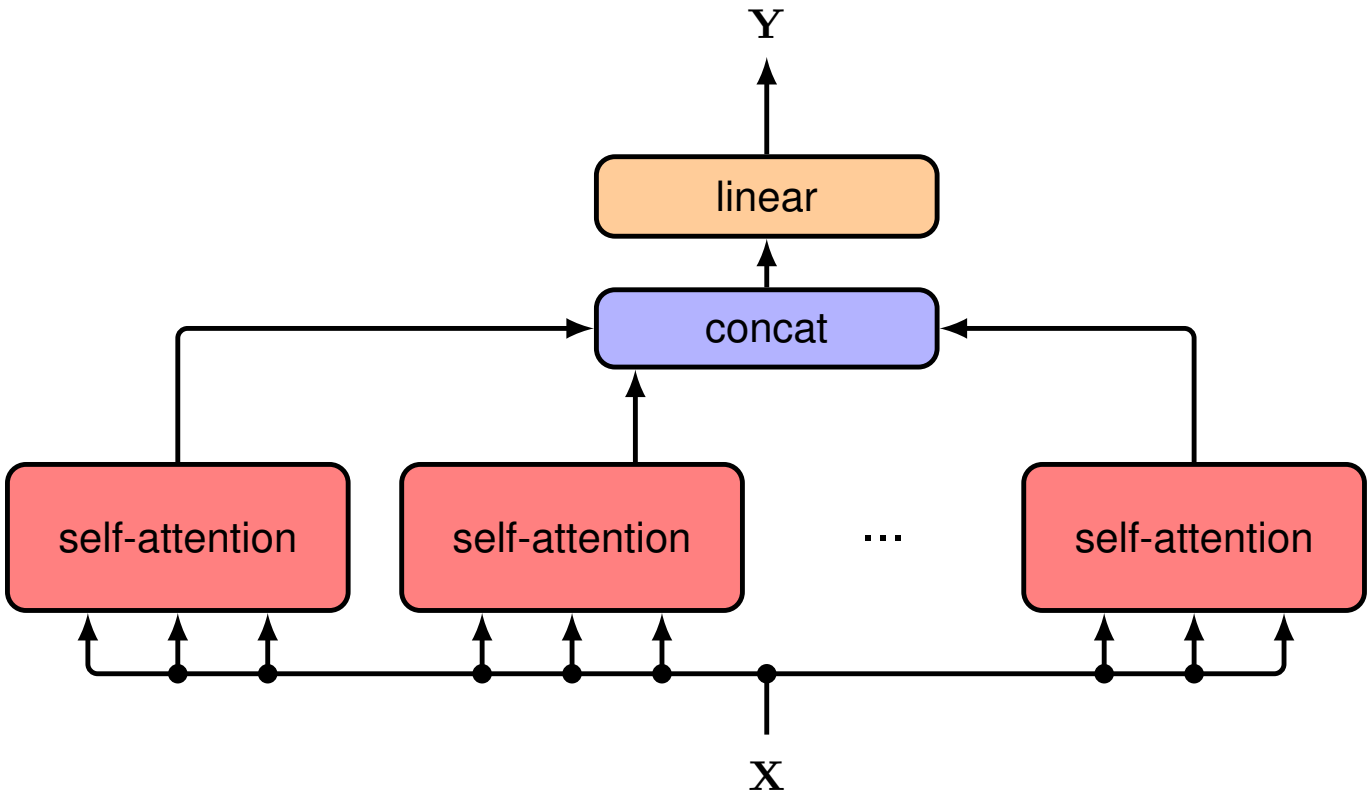
The dimension of the multi-head attention process is illustrated as



Since we want each head to have even dimension distribution over each head, we typically choose

$$Hd_v = d.$$

Illustration of Multi-head Attention



Attention Masks

Attention Masks: Motivation

In attention, every token could attend to every other token. However:

- Some positions are invalid (e.g., padding) and must not influence the result.
- Autoregressive decoding must be causal (no information leakage from the future).

Examples:

- Padding different length sequences to the same length:

```
I swam across the river to get to the other bank .
It is raining .  pad pad pad pad pad pad pad pad
```

  The padding tokens is not the real signal.

- The second case is causality. Self-attention at position t can attend to the future tokens $\boldsymbol{x}_{>t}$ as we will input the whole sequence. Then, It will minimize loss by copying signals from the future rather than learning real conditional structure.

$\rightsquigarrow$ Attention masks.

Definition of Attention Mask

Masked attention: Introduce a matrix $\boldsymbol{M}$ and re-write attention as

$$\text{attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{M}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}} + \boldsymbol{M}\right)\boldsymbol{V},$$

where

$$\boldsymbol{M}_{ij} = \begin{cases} 0, & \text{allowed} \\ -\infty, & \text{blocked} \end{cases}$$

- An attention mask $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ encodes hard visibility constraints. It is not learned; it is constructed from sequence structure (padding, causality, etc).
- 0 entry means allowed, i.e., allowing i-th token attending to j-th token.
- $-\infty$ simply ensures $\exp(-\infty) = 0$ in softmax, i.e., no attention of i to j.

Example: Padding Mask

Scenario: Variable-length sequences are batched by padding shorter sequences with a PAD token.

- Define a boolean padding indicator vector $\boldsymbol{p} \in \{0,1\}^n$:

$$p_j = \begin{cases} 1, & \text{position } j \text{ is padding} \\ 0, & \text{otherwise} \end{cases}$$

- Construct $\boldsymbol{M}^{\text{pad}} \in \mathbb{R}^{n \times n}$:

$$\boldsymbol{M}^{\text{pad}}_{ij} = \begin{cases} -\infty, & p_j = 1 \\ 0, & p_j = 0 \end{cases} \quad \Longleftrightarrow \quad \boldsymbol{M}^{\text{pad}} = \mathbf{1}\,\boldsymbol{p}^\top \cdot (-\infty),$$

  where $\mathbf{1} \in \mathbb{R}^n$ is an all-ones vector and $0 \cdot -\infty = 0$.

- Apply:

$$\boldsymbol{Y} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}} + \boldsymbol{M}^{\text{pad}}\right)\boldsymbol{V}.$$

  This prevents attention to padded keys/values from any query position i.

Example: Causal Mask

Scenario: Maintain causality, token i must not attend to future token $j > i$.

- Define a lower-triangular causal mask $\boldsymbol{M}^{\text{causal}} \in \mathbb{R}^{n \times n}$:

$$\boldsymbol{M}_{ij}^{\text{causal}} = \begin{cases} 0, & j \leq i \\ -\infty, & j > i \end{cases}$$

- Apply (combine with padding mask):

$$\boldsymbol{Y} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}} + \boldsymbol{M}^{\text{causal}} + \boldsymbol{M}^{\text{pad}}\right)\boldsymbol{V}.$$

Then, all bad attentions will be blocked.

Complexity

Complexity of Attention

Consider the just mentioned masked attention.

- Score matrix: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{K}^\top \in \mathbb{R}^{n \times n}$ with cost $O(n^2 d)$.
- Masking + softmax on $\boldsymbol{S}$: Store and softmax $n \times n$ scores with cost $O(n^2)$ (softmax) and memory $O(n^2)$ (score matrix and mask).
- Weighted sum: $\boldsymbol{Y} = \mathrm{softmax}(\boldsymbol{S})\boldsymbol{V}$ with cost $O(n^2 d)$.

Observations:

- Overall, computation and memory are both quadratic in sequence length n.
- Attention is not in linear complexity. One of the major drawback of Transformers.

$\rightsquigarrow$ Next lecture: Transformer architecture using attention mechanism.