5-3 Transformers

Zhonglei Wang

WISE and SOE, XMU, 2025

Contents

Motivation

1. Existing RNNs achieve great success for modeling sequential data
   - Using gates is inefficient to capture long-distance information
   - They are difficult to be parallelized, so their computation efficiency is limited
2. Vaswani et al. (2017) proposed a multi-head self-attention framework for translation
   - It does not rely on the RNN architecture
   - It uses self-attention to extract information from contexts
   - It can be parallelized to achieve high computation efficiency
   - It is widely used in deep learning models to extract information from sequential data

Transformers

1. Transformer is a new architecture
   - It uses shared parameters to deal with long input passages of different lengths
   - It detects connections between words represented by embeddings
2. It achieves the desired properties by *dot-product self-attention*
3. Consider a sentence "I have arrived at Beijing from Xiamen."
   - Assume it has been tokenized, and its embeddings are $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_8$
4. For simplicity, use $\boldsymbol{x}_i$, instead of $\boldsymbol{x}^{<i>}$ for the word vector

Dot-product self-attention

1. For each word vector $\boldsymbol{x}_i$, we first compute
   - Query:
   $$\boldsymbol{q}_i = \boldsymbol{b}_q + \boldsymbol{W}_q \boldsymbol{x}_i$$
   - Key:
   $$\boldsymbol{k}_i = \boldsymbol{b}_k + \boldsymbol{W}_k \boldsymbol{x}_i$$
   - Value:
   $$\boldsymbol{v}_i = \boldsymbol{b}_v + \boldsymbol{W}_v \boldsymbol{x}_i$$
2. The shared model parameters are $\{\boldsymbol{b}_q, \boldsymbol{W}_q, \boldsymbol{b}_k, \boldsymbol{W}_k, \boldsymbol{b}_v, \boldsymbol{W}_v\}$
3. The dimensions of model parameters are chosen such that
   - The dimensions of $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ are usually the same
   - The dimensions of $\boldsymbol{q}_i$ and $\boldsymbol{k}_i$ should be the same, but may be different from $\boldsymbol{x}_i$

Dot-product self-attention

1. For $i \in \{1, \ldots, 8\}, \boldsymbol{x}_i$ is updated by dot-product self-attention
   - For $l \in \{1, \ldots, 8\}$,
     $$a_{il} = a(\boldsymbol{q}_i, \boldsymbol{k}_l)$$
     - ▷ $a(\boldsymbol{x}, \boldsymbol{y})$: an alignment function (Bahdanau et al., 2015), measuring how well $\boldsymbol{x}$ and $\boldsymbol{y}$ match
     - ▷ Consider $a(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{y}$ for practical use
2. Normalize $\{a_{il} : l = 1, \ldots, 8\}$ to obtain
   $$w_{il} = \frac{\exp(a_{il})}{\sum_{h=1}^{8} \exp(a_{ih})}$$

Dot-product self-attention

1. For $i \in \{1, \ldots, 8\}$, $\boldsymbol{x}_i$ is updated by dot-product self-attention (Cont'd)
   - Update $\boldsymbol{x}_i$ by

$$\boldsymbol{x}_i' = \sum_{l=1}^{8} w_{il} \boldsymbol{v}_l$$

     - ▷ w_{il} measures how much "attention" we should pay to $\boldsymbol{v}_l$ when updating $\boldsymbol{x}_i$

2. Calculate $\{w_{il} : l = 1, \ldots, 8\}$ for each i to obtain UPDATED features

$$\{\boldsymbol{x}_i' : i = 1, \ldots, 8\}$$

Illustration

To calculate UPDATED feature for $\boldsymbol{x}_7$

Updated: $\boldsymbol{x}'_1$ $\boldsymbol{x}'_2$ $\boldsymbol{x}'_3$ $\boldsymbol{x}'_4$ $\boldsymbol{x}'_5$ $\boldsymbol{x}'_6$ $\boldsymbol{x}'_7$

Normalization:

Self-attention:

$\boldsymbol{q}_1, \boldsymbol{k}_1, \boldsymbol{v}_1$ $\boldsymbol{q}_2, \boldsymbol{k}_2, \boldsymbol{v}_2$ $\boldsymbol{q}_3, \boldsymbol{k}_3, \boldsymbol{v}_3$ $\boldsymbol{q}_4, \boldsymbol{k}_4, \boldsymbol{v}_4$ $\boldsymbol{q}_5, \boldsymbol{k}_5, \boldsymbol{v}_5$ $\boldsymbol{q}_6, \boldsymbol{k}_6, \boldsymbol{v}_6$ $\boldsymbol{q}_7, \boldsymbol{k}_7, \boldsymbol{v}_7$

Parameters: $\boldsymbol{b}_q$, $\boldsymbol{W}_q$, $\boldsymbol{b}_k$ $\boldsymbol{W}_k$, $\boldsymbol{b}_v$, $\boldsymbol{W}_v$

Embedding: $\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_3$ $\boldsymbol{x}_4$ $\boldsymbol{x}_5$ $\boldsymbol{x}_6$ $\boldsymbol{x}_7$

Vectorization

1. Denote

$$\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_7)^{\mathrm{T}} \in \mathbb{R}^{d\times 7}$$
$$\boldsymbol{V} = \boldsymbol{b}_q \mathbf{1}^{\mathrm{T}} + \boldsymbol{W}_q \boldsymbol{X}$$
$$\boldsymbol{K} = \boldsymbol{b}_k \mathbf{1}^{\mathrm{T}} + \boldsymbol{W}_k \boldsymbol{X}$$
$$\boldsymbol{V} = \boldsymbol{b}_v \mathbf{1}^{\mathrm{T}} + \boldsymbol{W}_v \boldsymbol{X}$$
$$\boldsymbol{X}' = (\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_7)^{\mathrm{T}} \in \mathbb{R}^{d\times 7}$$

2. Then, the dot-product self-attention is

$$\boldsymbol{X}' = \boldsymbol{V} \cdot \mathrm{Softmax}(\boldsymbol{K}^{\mathrm{T}} \cdot \boldsymbol{Q})$$

Positional encoding

1. It fails to concentrate on the ORDER of the words
2. Vaswani et al. (2017) to add a matrix $\boldsymbol{P}$ to $\boldsymbol{X}$

$$\boldsymbol{X}_p = \boldsymbol{X} + \boldsymbol{P}$$

- $\boldsymbol{P}$: encodes position information
- $\boldsymbol{X}_p$: the updated embedding matrix

3. For $\boldsymbol{P}$, we may consider to use sin and cos functions with different frequencies

$$\boldsymbol{P}_{(2i),j} = \sin(j/10000^{2i/d}), \quad \boldsymbol{P}_{(2i+1),j} = \cos(j/10000^{2i/d})$$

- i: dimension index
- j: position index

Scaled dot-product self-attention

1. There may be large values in $\boldsymbol{K}^{\mathrm{T}} \cdot \boldsymbol{Q}$
2. Then, a certain "value" may dominate
3. The model may be inefficient
4. To prevent such inconvenience, we consider

$$\boldsymbol{X}' = \boldsymbol{V} \cdot \mathrm{Softmax}\left(\frac{\boldsymbol{K}^{\mathrm{T}} \cdot \boldsymbol{Q}}{\sqrt{d_q}}\right)$$

- d_q : the dimension of "query" as well as "key"
- Statisticall, we are dividing the "standard deviation" to make the result stable

Multi-head self-attention

1. It is limited to only use a single set of “queries”, “keys” and “values”
2. We may consider H such sets
3. Denote

$$\boldsymbol{X}_h' = \boldsymbol{V}_h \cdot \mathrm{Softmax}\left(\frac{\boldsymbol{K}_h^{\mathrm{T}} \cdot \boldsymbol{Q}_h}{\sqrt{d_q}}\right) \quad (h = 1, \ldots, H)$$

- $\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h$: the hth set of “queries”, “keys” and “values”
- Different set of model parameters are used for different “head”

4. The updated feature matrix is obtained by concatenating informtion from those “head

$$\boldsymbol{X}' = \boldsymbol{W}_o(\boldsymbol{X}_1^{\mathrm{T}}, \ldots, \boldsymbol{X}_H^{\mathrm{T}})^{\mathrm{T}}$$

- $\boldsymbol{W}_o$: weight matrix to combine information from different heads

Remarks

1. Advantages:
   - Parallelization: Unlike sequential models, it allows for full parallel processing which speeds up training.
   - Long-Range Dependencies: It provides direct access to distant elements making it easier to model complex structures and relationships across long sequences.
   - Contextual Understanding: Each token's representation is influenced by the entire sequence which integrates global context and improves accuracy.
   - Interpretable Weights: Attention maps can show which parts of the input were most influential in making decisions.

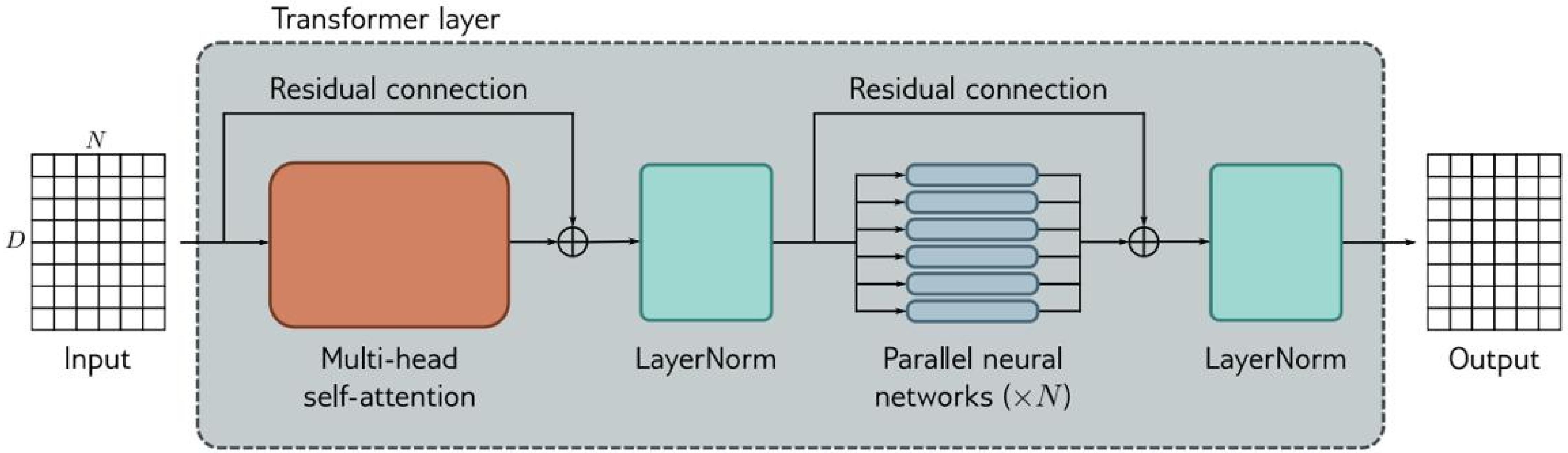
Remarks

1. Disadvantages:

   - Computational Cost: Especially for long sequence with n tokens, it requires computing pairwise interactions between all input tokens which causes a time and memory complexity of $O(n^2)$.
   - Memory Usage: Large number of pairwise calculations in self-attention uses high memory while working with very long sequences or large batch sizes.
   - Lack of Local Context: It focuses on global dependencies across all tokens but it may not effectively capture local patterns. This can cause inefficiencies when local context is more important than global context.
   - Overfitting: Due to its ability to model complex relationships it can overfit when it is trained on small datasets.

Transformer layer

1. Its building blocks consist of
   - Multi-head dot-product self-attention (for word interaction)
   - Fully connected NN (for each word with shared parameters)
   - Residual connection
   - Layer normalization
2. Figure 12.7 of Prince (2024)



Transformer layer

1. Calculation is

$$
\begin{aligned}
\boldsymbol{X} &\leftarrow \boldsymbol{X} + \mathrm{MhSa}(\boldsymbol{X}) \\
\boldsymbol{X} &\leftarrow \mathrm{LayerNorm}(\boldsymbol{X}) \\
\boldsymbol{x}_i &\leftarrow \boldsymbol{x}_i + \mathrm{MLP}\boldsymbol{x}_i \\
\boldsymbol{X} &\leftarrow \mathrm{LayerNorm}(\boldsymbol{X})
\end{aligned}
$$

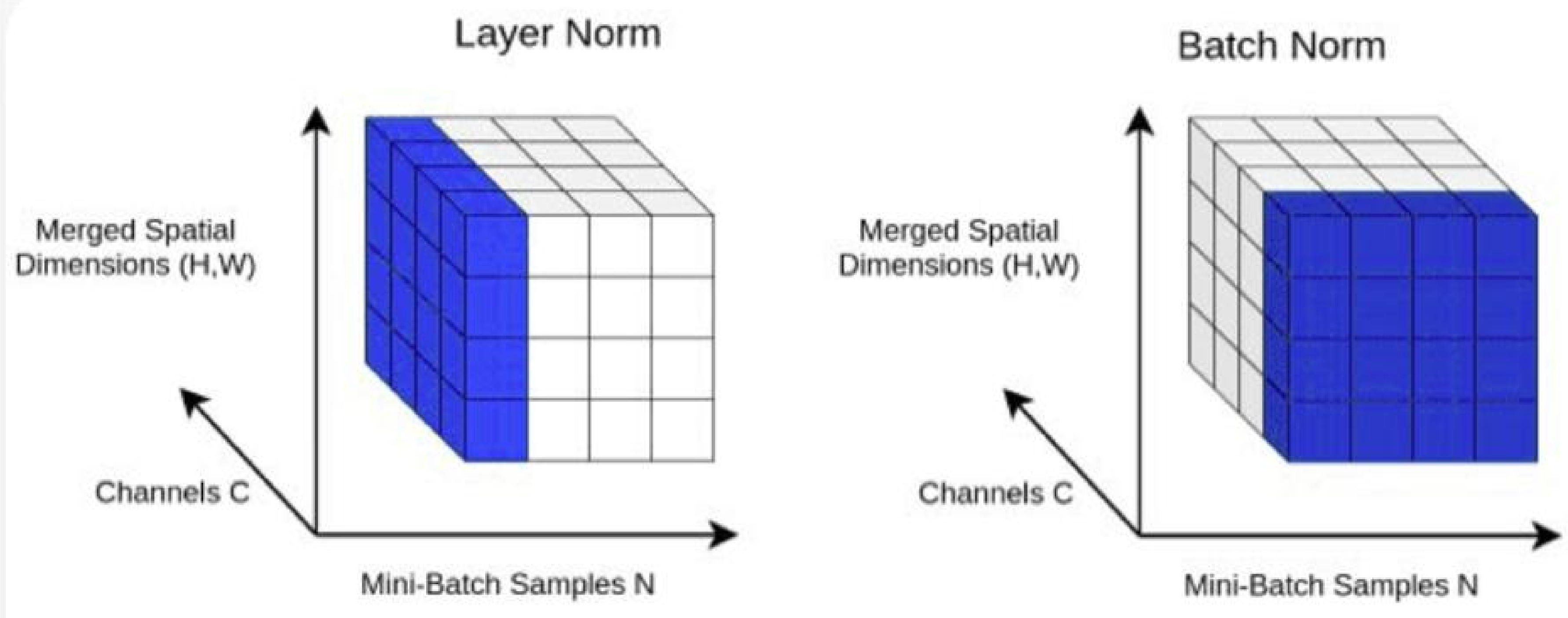
- $\boldsymbol{X}$: matrix after embedding and positional encoding for a SINGLE input sequence
- $\boldsymbol{x}_i$: the ith column of $\boldsymbol{X}$
- $\mathrm{MhSa}(\boldsymbol{X})$: multi-head dot-product self-attention
- $\mathrm{MLP}(\boldsymbol{X})$: fully connected NN (with shared parameters)
- $\mathrm{LayerNorm}(\boldsymbol{X})$: layer normalization

Transformer layer

1. Recall batch normalization
   - Applied to EACH feature of ALL of all examples in the same mini-batch
   - Not applicable to NLP since input sequences may have different lengths

Transformer layer

1. Layer normalization is applied across tokens within the SAME input sequence
2. The figure below is from https://theaisummer.com/normalization/
   - N : the size of a mini-batch or input sequences
   - C : number of features of length of input sequences
   - H, W : vectorized feature of embedding



Transformer model

1. It mainly consists of several transformer layers
2. Encoder-decoders are used in sequence-to-sequence tasks
   - Encoder (text to numbers): transforms the text embeddings into a representation that can support a variety of tasks
   - Decoder (numbers to text): predicts the next token to continue the input text
3. Examples
   - BERT: an encoder model
   - ChatGPT: a decoder model
4. See Chapter 12 of Prince (2024) for details