



AI and Deep Learning

5-3 Transformers

Zhonglei Wang

WISE, SOE, CHOW

XMU

(Version v1)

Contents

1. Motivation
2. Dot-product self-attention
3. Position encoding
4. Multi-head self-attention
5. Transformer layer
6. Transformer model

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 can be parallelized to achieve **high computation efficiency**
 - More efficient to capture long-distance information
 - 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 $\mathbf{x}_1, \dots, \mathbf{x}_7$
4. For simplicity, use $\mathbf{x}_i$, instead of $\mathbf{x}^{<i>}$ for the word vector

Dot-product self-attention

1. The shared model parameters are $\{\mathbf{b}_q, \mathbf{W}_q, \mathbf{b}_k, \mathbf{W}_k, \mathbf{b}_v, \mathbf{W}_v\}$
2. For each word vector $\mathbf{x}_i$, we first compute

- Query:

$$\mathbf{q}_i = \mathbf{b}_q + \mathbf{W}_q \cdot \mathbf{x}_i$$

- Key:

$$\mathbf{k}_i = \mathbf{b}_k + \mathbf{W}_k \cdot \mathbf{x}_i$$

- Value:

$$\mathbf{v}_i = \mathbf{b}_v + \mathbf{W}_v \cdot \mathbf{x}_i$$

3. The dimensions of model parameters are chosen such that
 - The dimensions of $\mathbf{x}_i$ and $\mathbf{v}_i$ are usually the same
 - The dimensions of $\mathbf{q}_i$ and $\mathbf{k}_i$ should be the same, but may be different from $\mathbf{x}_i$

Dot-product self-attention

1. For $i \in \{1, \dots, 7\}$, $\mathbf{x}_i$ is updated by dot-product self-attention

- For $l \in \{1, \dots, 7\}$,

$$a_{il} = a(\mathbf{q}_i, \mathbf{k}_l)$$

▷ $a(\mathbf{x}, \mathbf{y})$: Alignment function (Bahdanau et al., 2015), measuring how well $\mathbf{x}$ and $\mathbf{y}$ match

▷ Transformer uses $a(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \cdot \mathbf{y}$

2. Normalize $\{a_{il} : l = 1, \dots, 7\}$ to obtain

$$w_{il} = \frac{\exp(a_{il})}{\sum_{h=1}^7 \exp(a_{ih})}$$

Dot-product self-attention

1. For $i \in \{1, \dots, 7\}$, $\mathbf{x}_i$ is updated by dot-product self-attention (Cont'd)

- Update $\mathbf{x}_i$ by

$$\mathbf{x}'_i = \sum_{l=1}^7 w_{il} \mathbf{v}_l$$

▷ w_{il} measures how much “attention” we should pay to $\mathbf{v}_l$ when updating $\mathbf{x}_i$

2. Calculate $\{w_{il} : l = 1, \dots, 7\}$ for each i to obtain UPDATED features

$$\{\mathbf{x}'_i : i = 1, \dots, 7\}$$

Illustration

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

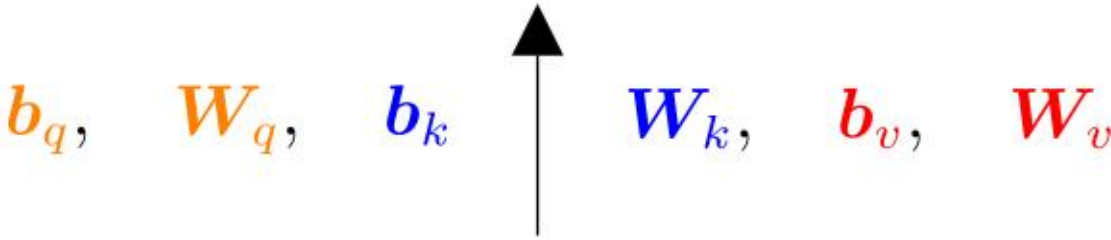
Updated: $\mathbf{x}'_1$ $\mathbf{x}'_2$ $\mathbf{x}'_3$ $\mathbf{x}'_4$ $\mathbf{x}'_5$ $\mathbf{x}'_6$ $\mathbf{x}'_7$

Normalization:

Self-attention:

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

Parameters:



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

Vectorization

1. Denote

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_7) \in \mathbb{R}^{d \times 7}$$

$$\mathbf{V} = \mathbf{b}_q \cdot \mathbf{1}^T + \mathbf{W}_q \cdot \mathbf{X}$$

$$\mathbf{K} = \mathbf{b}_k \cdot \mathbf{1}^T + \mathbf{W}_k \cdot \mathbf{X}$$

$$\mathbf{V} = \mathbf{b}_v \cdot \mathbf{1}^T + \mathbf{W}_v \cdot \mathbf{X}$$

$$\mathbf{X}' = (\mathbf{x}'_1, \dots, \mathbf{x}'_7) \in \mathbb{R}^{d \times 7}$$

2. Vectorization for the dot-product self-attention is

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

3. It fails to concentrate on the **ORDER** of the words

I arrived at **Beijing** from **Xiamen**. I arrived at **Xiamen** from **Beijing**.

Positional encoding

1. Vaswani et al. (2017) to add a matrix $\mathbf{P}$ to $\mathbf{X}$

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

- $\mathbf{P}$: Encodes position information
- $\mathbf{X}_p$: Updated embedding matrix

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

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

- i : Dimension index
- j : Position index

Scaled dot-product self-attention

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

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

- d_q : Dimension of “query” as well as “key”
- Statistically, 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

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

- $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$: h th set of “queries”, “keys” and “values”
4. The updated feature matrix is obtained by concatenating information from those “heads”

$$\mathbf{X}' = \mathbf{W}_o \cdot (\mathbf{X}'_1, \dots, \mathbf{X}'_H)^T$$

- $\mathbf{W}_o$: 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:

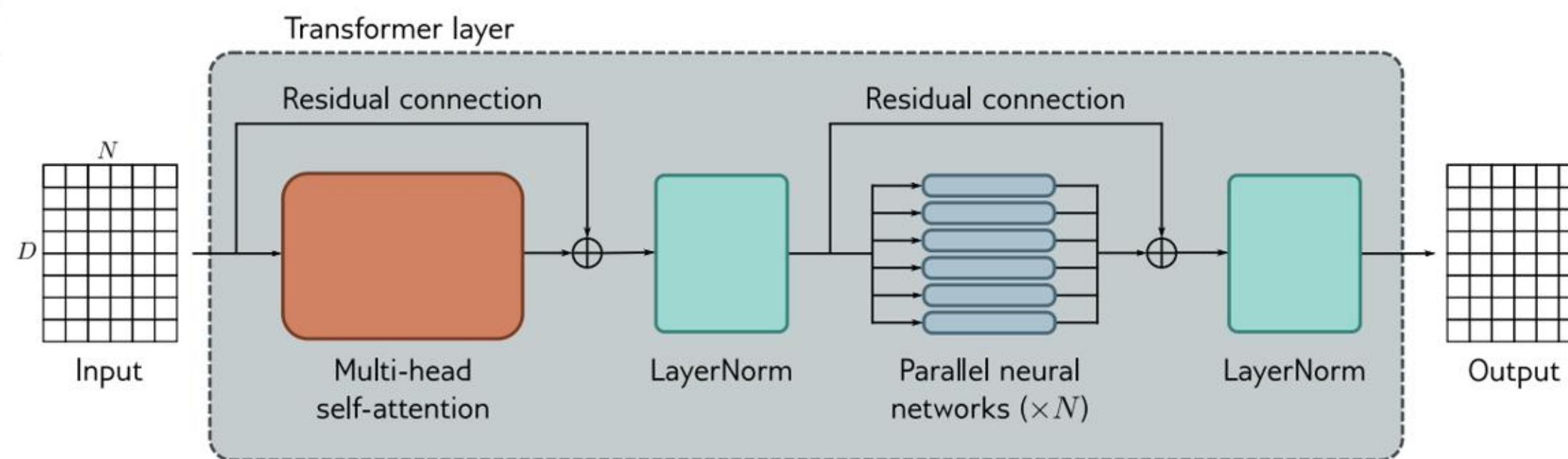
- **Computational Cost:** For long sequence with n tokens, it requires computing pairwise interactions between all input tokens which causes a time and memory complexity of $O(n^2)$
- **Lack of Local Context:** It focuses on global dependencies across all tokens but it may not effectively capture local 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

$$\mathbf{X} \leftarrow \mathbf{X} + \text{MhSa}(\mathbf{X})$$

$$\mathbf{X} \leftarrow \text{LayerNorm}(\mathbf{X})$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \text{MLP}(\mathbf{x}_i)$$

$$\mathbf{X} \leftarrow \text{LayerNorm}(\mathbf{X})$$

- $\mathbf{X}$: Matrix after embedding and positional encoding for a SINGLE input sequence
- $\mathbf{x}_i$: i th column of $\mathbf{X}$ (*different from design matrix*)
- $\text{MhSa}(\mathbf{X})$: Multi-head dot-product self-attention
- $\text{MLP}(\mathbf{X})$: Fully connected NN (with shared parameters)
- $\text{LayerNorm}(\mathbf{X})$: **Layer normalization**

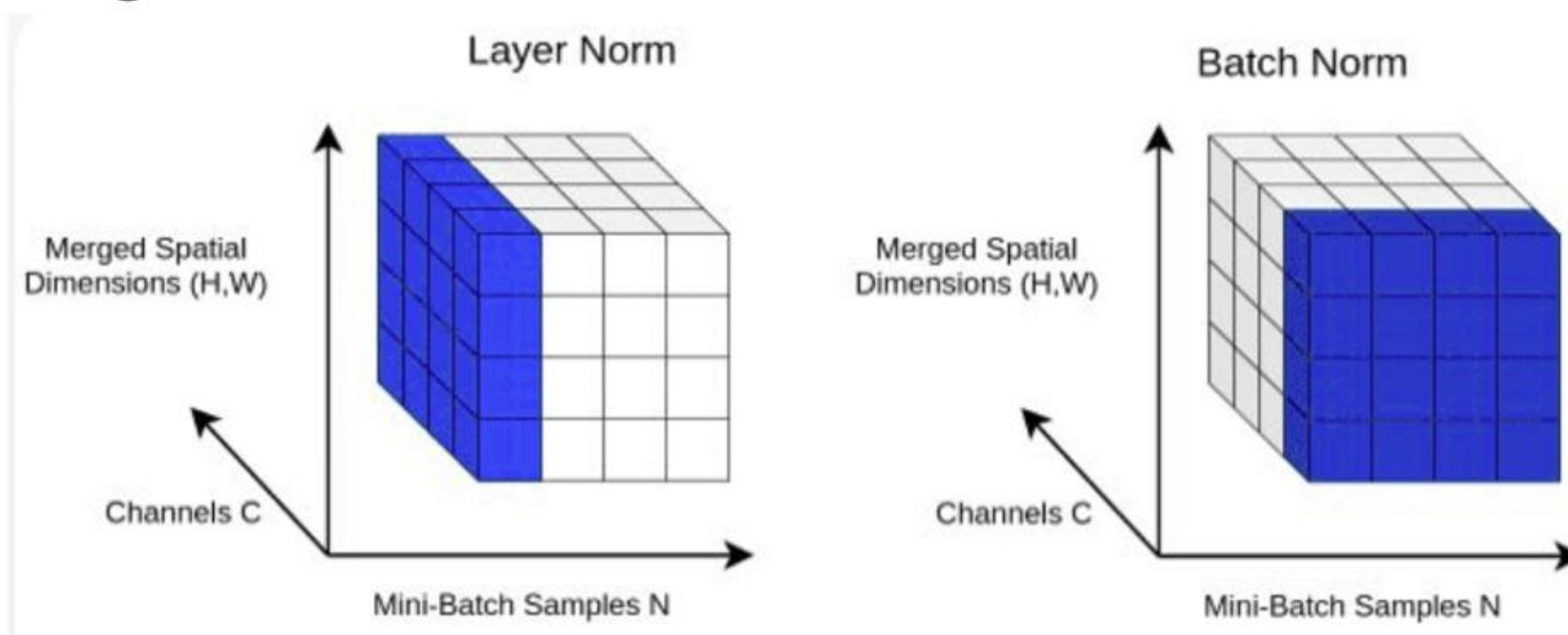
Transformer layer

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 : Size of a mini-batch or input sequences
 - C : Number of features or 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