Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 25.11.2024

Deep Learning and Artificial Intelligence

WS 2024/25

Exercise 6: Attention & Transformer

Exercise 6-1 Implementation

In Moodle you find a Jupyter notebook in which you should implement the multihead attention mechanism in PyTorch.

Exercise 6-2 Additional Questions

(a) Discuss the relation between self-attention and cross-attention in terms of their similarities and dissimilarities, and where they are used in the transformer architecture.

Detailed Solution

In both self-attention and cross-attention, a set of queries attends to a set of key-value pairs with the same mathematical operations.

However, in self-attention, the queries and key-value pairs represent the same instances (i.e., they are generated from the same tokens), whereas in cross-attention the queries may represent different instances than the key-values pairs. Therefore, there is a one-to-one relation between queries and key-value pairs in self-attention, while the number of queries and key-value pairs in cross-attention may be different.

In other words, self-attention is the special case of QKV-attention when the queries, keys, and values come from the same input features and cross-attention is the complementary case where different sets of features are used for query and key-value generation.

In encoder-decoder architectures such as transformers, the encoder layers therefore typically use self-attention to encode and contextualize input features. In the decoder, we typically see a mixture of self-attention and cross-attention layers. The self-attention layers in the decoder are used to relate the output tokens to each other and the cross-attention layers are used to ”inform” the output tokens of the input features.

Furthermore, cross-attention is often used to bring features from different data modalities together, e.g., text prompts and image features in Stable Diffusion.

(b) The figure below depicts the original transformer architecture as proposed in Attention Is All You Need. Which of the attention layers use self-attention and which ones use cross-attention?

Detailed Solution

The two orange boxes in the bottom represent self-attention layers, whereas the upper orange box in the decoder is a cross-attention layer, as the keys and values come from the encoder outputs and not the decoder queries (output embeddings/tokens).

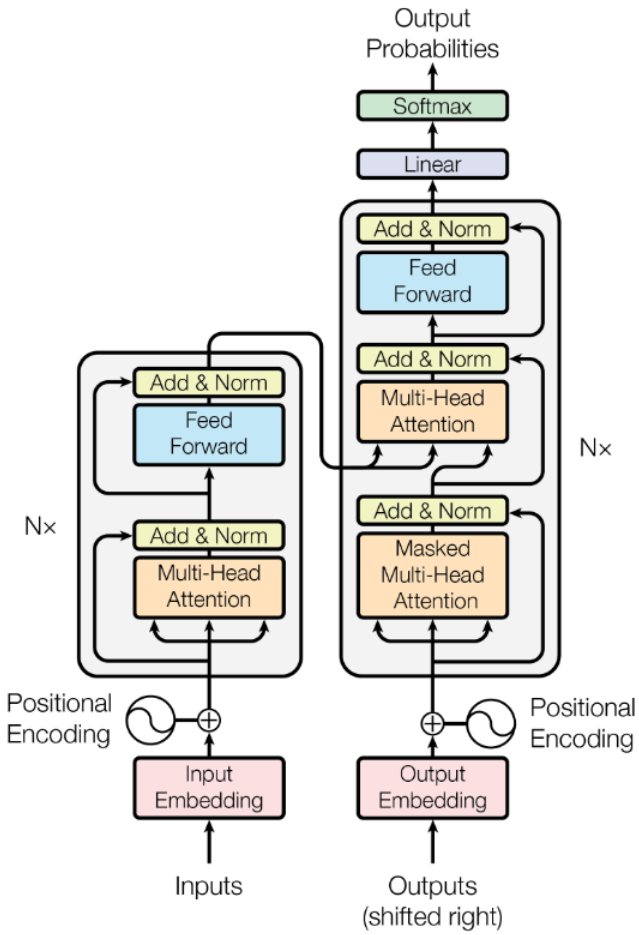

Figure 1: **Transformer architecture** from Attention Is All You Need.

(c) What are positional encodings? Why do we need them? What two kinds of positional encodings exist?

Detailed Solution

The attention mechanism combines different tokens via a weighted sum of all value tokens where the weights are produced by pairwise dot-products. Thus, it has no access to the order or position of input tokens. However, in many domains like NLP and CV, we know that positioning and locality are important, which is why we would like to include this inductive bias in transformers as well. To this end, we add so-called positional encodings to the token representation, making it localizable to the model. Also, this makes it easier for the model to exploit locality, as positional encodings for close positions are designed to be similar. In general, we distinguish between fixed positional encodings (mostly based on trigonometric functions) and learnable positional encodings that are trained just like any other learnable parameter of a model (see Attention Is All You Need).

(d) Compare convolutional, recurrent, and self-attention layers in terms of their computational complexity.

Detailed Solution

- Convolutional: $\mathcal{O}(k \cdot n \cdot d)$
- Recurrent: $\mathcal{O}(n \cdot d^2)$
- Attention: $\mathcal{O}(n^2 \cdot d)$

k: kernel size, n: sequence length, d: input dimensionality

Furthermore, note that the operations in recurrent layers have to be computed sequentially, while convolutional and attention operations can be parallelized.

(e) Compare transformers and RNNs in terms of their capabilities for handling long sequences.

Detailed Solution

Because of the global attention over all input tokens in transformers, they are well-suited for capturing long-range dependencies. On the other hand, RNNs often struggle to preserve information over a long time/range as the hidden state is updated with every new input.

In terms of computation, however, the quadratic complexity of the attention mechanism represents represents a downside of the global attention.