Stochastic Machine Learning
Chapter 04 - Transformers

Thorsten Schmidt

Abteilung für Mathematische Stochastik

www.stochastik.uni-freiburg.de
thorsten.schmidt@stochastik.uni-freiburg.de

SS 2024

Transformers

- In the paper "Attentions is all you need", Vaswani et al. (2017) (today cited approx 120.000 time) the authors proposed a new architecture outperforming LSTM, the state of the art at that time.
- Instead of recurrence, the architecture relies on attention mechanisms which are easier to train and to parallelize.
- The transformer has an encoder - decoder structure: given an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a a sequence of continuous representations $z = (z_1, \ldots, z_n)$
- The decoder then translates z back to the output symbols $(y_1, \ldots, y_m)$.
- In the following graphic from Vaswani et al. (2017) we have the encoder on the left and the decoder on the right.

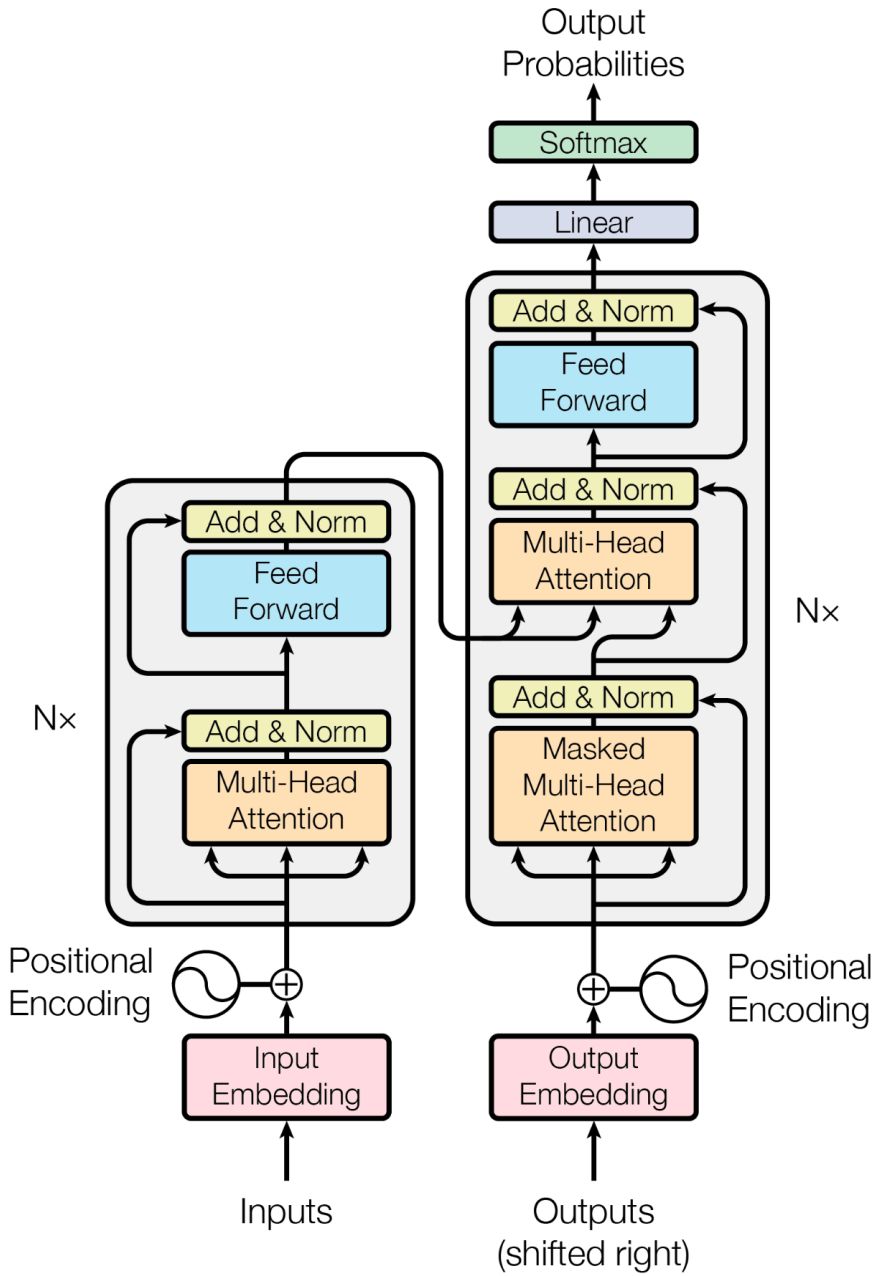


Figure 1: The Transformer - model architecture.

- The approach starts by the following steps:
- first, the input (text) is converted into tokens
- Then it follows an input embedding layer, which converts tokens and positions of tokens into vector representations
- The key are transformer layers, which repeatedly carry out transformations on the vector representations, extracting more and more information
- The transformer layers consist of alternating attention and feedforward layers.
- While in the original paper the normalization was after the attention layer, in a 2020 paper it was found out that putting them before the attention layer improves the performance
- At the very end there is an optional unembedding layer, which converts the output of the softmax back to a **probability distribution** over the tokens.

- While the original paper proposed different architectures for encoding and decoding, later papers (BERT and GPT) only use decoder-layers.
- The encoding layers process the input iteratively one layer after another
- The deocding layers are more complex. They contain to attention sublayers:
- first, a cross-attention for incorporating the output of encoder (contextualized input token representations),
- second, a self-attention for "mixing" information among the input tokens to the decoder (i.e., the tokens generated so far during inference time)
- the transformer building blocks are attention layers based on scalar products. Each attention layer consists of three matrices: query weights W_Q, key weights W_K and value weights W_V.

The attention layer

- For each token i the vector representation x_i this results in

$$q_i = W_Q x_i$$
$$k_i = W_K x_i$$
$$v_i = W_V x_i.$$

- The attention weights a_{ij} are computed as

$$\tilde{a}_{ij} = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

where d_k is the dimension of the key vectors. The $\tilde{a}_{ij}$ are passed through a softmax function

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_k e^{x_j}}$$

which converts the values of x into a probability distribution (see next slide).

- Note that the attention mechanism is non-symmetric
- The output of the attention unit for token i is given by the weighted tokens,

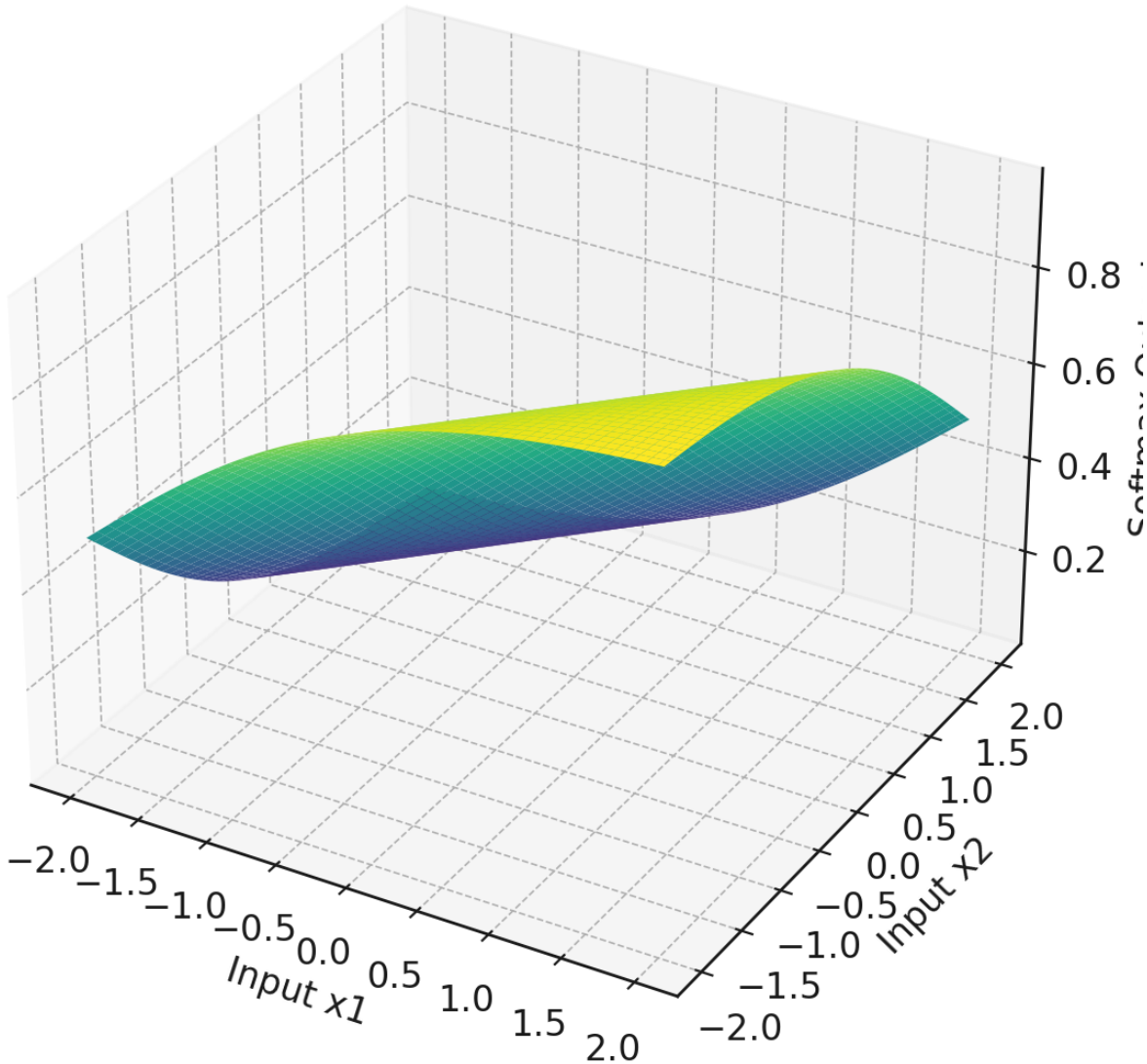
$$\sum_j a_{ij} v_j$$

- Overall we obtain

$$Attention(W_Q, Q_X, W_V) = V\sigma\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)$$

where Q, K and V are the matrices of the (q_i), (k_i) and (v_i)

Two-Dimensional Softmax Function



Multi-headed attention

- While one attention layer focusses on a single aspect, **multi-headed attention** concatenates various attention layers to be able to focus on multiple effects.
- The concatenation of $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$ is simply

$$\text{concat}(x, y) = (x_1, \ldots, x_n, y_1, \ldots, y_m)$$

- and thus, the multi-headed attention is

$$MHAttention = W_O \, \text{concat}_{i=1}^{n}(Attention(XW_Q^i, XW_K^i, XW_V^i))$$

with output projection W_O and the concatenation of the word embeddings X

Masked attention

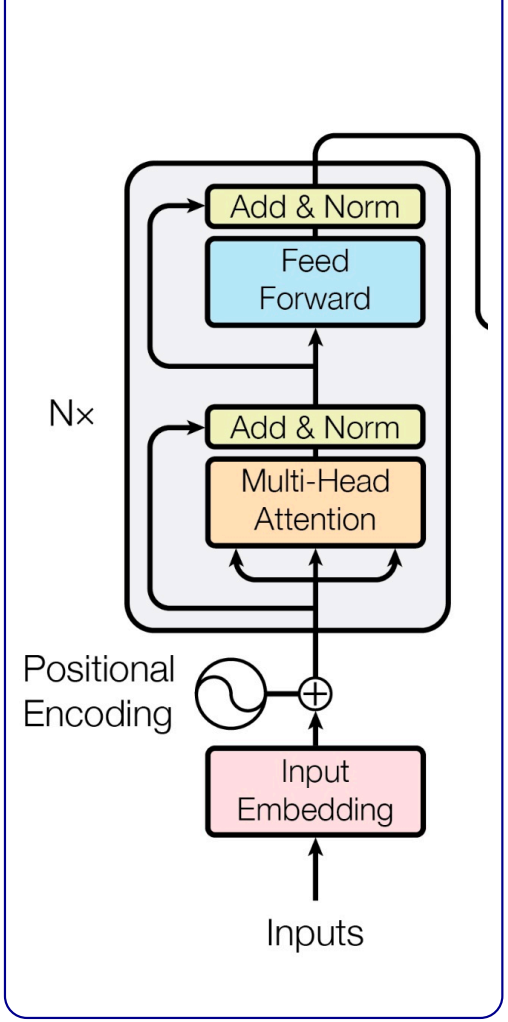
- It may be necessary to cut the attention for several tokens. For example, one wants to avoid that attention looks "into the future"
- This can be achied by adding the matrix

$$M = \begin{pmatrix} 0 & -\infty & \dots & -\infty \\ 0 & \ddots & \dots & -\infty \\ \vdots & & \vdots & \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

such that

$$MaskedAttention = V\sigma\Big(M + \frac{QK^\top}{\sqrt{d_k}}\Big)$$

A detailed view on the encoder



- Each encoder has two main components: a multi-head self-attention component and a classical feed-forward network.
- The attention component receives encodings from the previous encoders and generates an output encoding.
- This is passed into a feed-forward network, after normalization and adding of the previous inputs.
  The FFN processes the output encoding individually.
- The output is passed to the next encoder as input **and** to the the decoder.
- Only the first encoder receives information only from the input vectors and positional information.
- The input encoder is bi-directional (not masked), i.e. attention can be placed on tokens before and after the current token.

The positional encoding

- The positional encoding helps the transformer to bring attention to relevant parts of the data in the close-by neighbourhood or even to places very far away.
- The goal of the authors was to be numerically fast and quite flexible
- In the paper the authors chose $f : \mathbb{R} \to \mathbb{R}^d$ with $d \in 2\mathbb{N}$ with

$$\big(f(t)_{2k}, f(t)_{2k+1}\big) = \big(\sin \frac{t}{r^k}, \cos \frac{t}{r^k}\big),$$

  with $r = N^{2/d}$ and a large $N = 10.000$.
- Let us write this in complex notation:

$$f(t) = \big(e^{it/r^k}\big)_{k=0,\dots,d/2-1}.$$

- Then it becomes clear, that shifts can be implemented by simple linear transformations, i.e.

$$f(t + \Delta t) = \operatorname{diag} f(\Delta t) f(t) = \big(e^{i(t+\Delta t)/r^k}\big)_{k=0,\dots,d/2-1}.$$

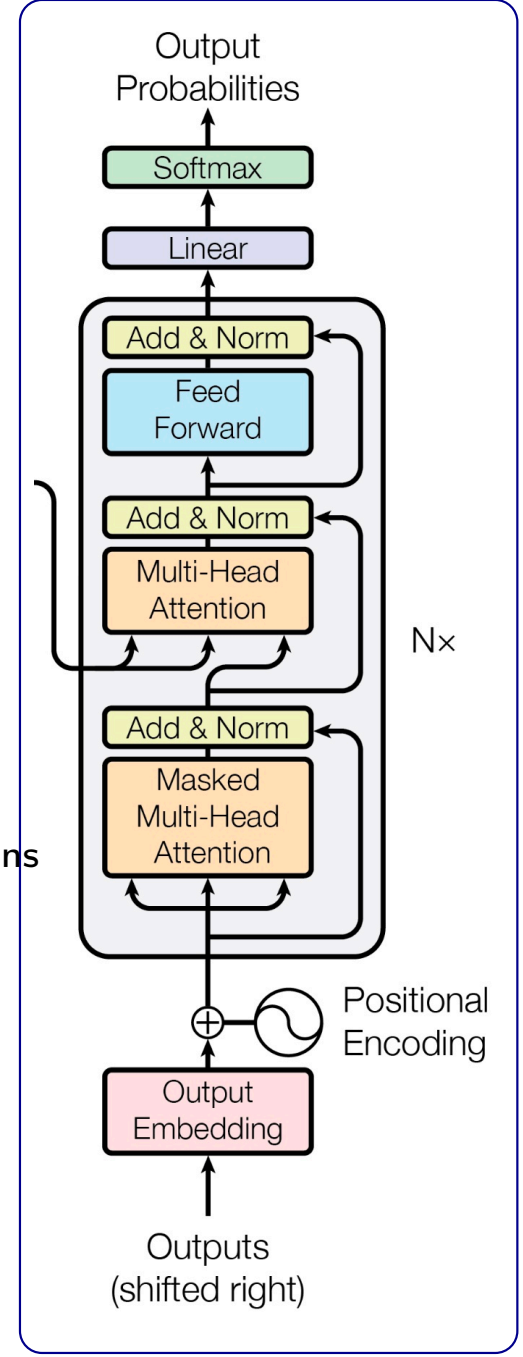
- Also convolutions can be implemented as linear transformations:

$$\sum_j c_j f(t + t_j) = \Big(\sum_k c_k \operatorname{diag} f(t_j)\Big) f(t).$$

- This mimics convolutional neural networks and allows to bring attention to relative positions.

The decoder

- The decoder has three components: a self-attention mechanism (like the encoder), an attention mechanism taking into account the encodings (this is the additional part of the decoder), and a feed-forward neural network (like in the encoder).
- For the decoder one has to take care that no future information is taken into account (for the prediction of the future outcome) and that no information is placed into future outputs, so masking takes place here.
- First, all attention mechanisms following tokens are masked used the masked attention mechanism described above
  An alternative is an "autoregressive" mechanism where all attentions from one token to following tokens are set to zero.
- Also the output sequence is partially masked
- The last decoder undergoes final linear transformation followed by a softmax layer which returns a probability distribution as final output.
- Of course, also here further models are developed.



 Vaswani, Ashish et al. (2017). „Attention is all you need“. In: **Advances in neural information processing systems** 30.