Exercise Sheet 9

Due: 12.01.26 14h00

Christof Gehrig, Gerrit Gerhartz

General Regulations.

- Please hand in your solutions in groups of two (preferably from the same tutorial group).
- Your solutions to theoretical exercises can be either handwritten notes (scanned), or typeset using LaTeX. For scanned handwritten notes, please ensure they are legible and not blurry.
- For the practical exercises, always provide the (commented) code as well as the output, and don't forget to explain/interpret the latter.
- Please hand in a **single PDF** that includes both the exported notebook and your solutions to the theoretical exercises. Submit the PDF to the Übungsgruppenverwaltung once per group, making sure to include the names of both group members in the submission.
- You can find all the data in the GitHub Repository.

1 Positional Encoding

In transformers, the features $X \in \mathbb{R}^{p\times n}$ are combined with the positional embeddings E of the tokens. We will consider the simplest case of self-attention where the attention weights are computed as the soft(arg)max over the "scores" K^TQ with (i) $K = Q = X + E$ in case features and positional embeddings are combined by adding them up and (ii) $K = Q = \text{concat}(X, E)$ in case they are concatenated. Let the positional embeddings $E \in \mathbb{R}^{p\times n}$ be defined as

$$E_{(2k),i} = \sin\left(i \cdot \exp\left(-\frac{2k \cdot \log(10000)}{p}\right)\right)$$
$$E_{(2k+1),i} = \cos\left(i \cdot \exp\left(-\frac{2k \cdot \log(10000)}{p}\right)\right),$$

for $k \in \{0, 1, 2, \ldots, \frac{p}{2} - 1\}$, taken from "Attention Is All You Need". The index i runs over the samples (which are assumed to be ordered). log is the natural logarithm.

(a) Expand the scores in X and E both for addition and concatenation. Discuss the different resulting terms. (2 pts)

(b) Implement the positional encoding as defined above and plot E^TE for 64 tokens with an embedding dimension of 256. What do you observe? (2 pts)

(c) Plot K^TQ for some random features (with the same variance as the positional embedding) for both addition and concatenation and discuss what you see. (2 pts)

(d) **Bonus:** Assume that both the positional encodings and the features of all tokens lie in two (comparatively) low-dimensional subspace of the high-dimensional Euclidean space. Argue why, under these assumptions, adding the positional encodings will lead to similar results as concatenating them with the features. (2 bonus pts)

2 Multi-Head Latent Attention

DeepSeek presented Multi-Head Latent Attention (MLA), a scheme to compress key and value matrices and thereby save storage. Ignoring positional embeddings for simplicity, MLA works as follows:

$$\mathbf{z}_t = C\mathbf{x}_t$$
$$\hat{\mathbf{k}}_t = D^K\mathbf{z}_t$$
$$\hat{\mathbf{v}}_t = D^V\mathbf{z}_t$$

Here, $\mathbf{x}_t \in \mathbb{R}^p$ is an input at time t, $C \in \mathbb{R}^{l \times p}$ with $l < p$ is a compression matrix projecting onto an l-dimensional subspace, and $D^K, D^V \in \mathbb{R}^{p \times l}$ are "decompression" matrices for keys and values, respectively. $\mathbf{x}_t$ is discarded, and only $\mathbf{z}_t$ is stored.

(a) Why did DeepSeek not propose the following?

$$\begin{aligned}\mathbf{z}_t &= C\mathbf{x}_t \\ \hat{\mathbf{k}}_t &= W^K D\mathbf{z}_t \\ \hat{\mathbf{v}}_t &= W^V D\mathbf{z}_t\end{aligned}$$

(2 pts)

(b) What would be a suitable training-time objective to find C, D in the equations proposed in a) that lead to minimal loss of information? (1 pt)

(c) DeepSeek was trained by fine-tuning Meta's Llama, and so it was convenient to recover $\hat{\mathbf{k}}_t, \hat{\mathbf{v}}_t$ with the original dimensions $p \times 1$. Other than compatibility with the original Llama, do you see any advantage in linearly projecting back to the original dimension p? That is, does $\hat{\mathbf{k}}_t$ hold more information than $\mathbf{z}_t$? (2 pts)

Note: This exercise is taken verbatim from last year's exam.

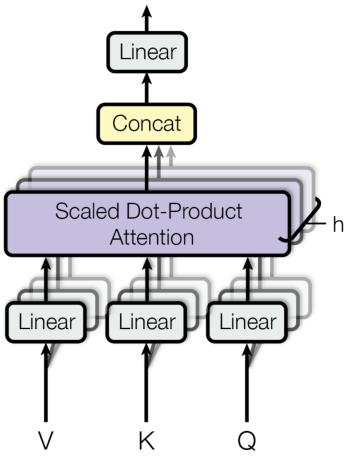
3 Mixture of Experts and backpropagation

DeepSeek uses a "mixture of experts" (MoE) at every layer. The following excerpt is from the DeepSeek paper, where "FFN" stands for "feed-forward network", a multi-layer perceptron:

> For FFNs, we employ the DeepSeekMoE architecture (Dai et al., 2024). DeepSeekMoE has two key ideas: segmenting experts into finer granularity for higher expert specialization and more accurate knowledge acquisition, and isolating some shared experts for mitigating knowledge redundancy among routed experts. With the same number of activated and total expert parameters, DeepSeekMoE can outperform conventional MoE architectures like GShard (Lepikhin et al., 2021) by a large margin.
>
> Let $\mathbf{u}_t$ be the FFN input of the t-th token, we compute the FFN output $\mathbf{h}'_t$ as follows:
>
> $$\mathbf{h}'_t = \mathbf{u}_t + \sum_{i=1}^{N_s} \mathrm{FFN}_i^{(s)}(\mathbf{u}_t) + \sum_{i=1}^{N_r} g_{i,t}\,\mathrm{FFN}_i^{(r)}(\mathbf{u}_t), \tag{20}$$
>
> $$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise}, \end{cases} \tag{21}$$
>
> $$s_{i,t} = \mathrm{Softmax}_i\left(\mathbf{u}_t{}^T \mathbf{e}_i\right), \tag{22}$$
>
> where N_s and N_r denote the numbers of shared experts and routed experts, respectively; $\mathrm{FFN}_i^{(s)}(\cdot)$ and $\mathrm{FFN}_i^{(r)}(\cdot)$ denote the i-th shared expert and the i-th routed expert, respectively; K_r denotes the number of activated routed experts; $g_{i,t}$ is the gate value for the i-th expert; $s_{i,t}$ is the token-to-expert affinity; $\mathbf{e}_i$ is the centroid of the i-th routed expert in this layer; and $\mathrm{Topk}(\cdot, K)$ denotes the set comprising K highest scores among the affinity scores calculated for the t-th token and all routed experts.

Hint: The centroids $\mathbf{e}_i$ are learnable parameters with the same dimensionality as $\mathbf{u}_t$.

(a) Create a schematic sketch of the MoE described in equation (20), concentrating on the paths that $\mathbf{u}_t$ takes. You can represent each FFN and the router $g_{\cdot,t}(\mathbf{u}_t)$ by a single box and disregard computational efficiency. Use a level of detail similar to the attached plot (though not necessarily a similar layout): (3 pts)



The following questions look at the gradient flow across this MoE.

(b) Give the formula for the adjoint in backpropagation / reverse-mode automatic differentiation at node v_a of a computational graph with children $\mathcal{C}(a)$. (1 pt)

(c) In the backward pass of back propagation, what message does node v_a send to its parents in the computational graph if v_a represents a sum of two terms $c + d$? (1 pt)

(d) In the backward pass of back propagation, what message does node v_a send to its parents in the computational graph if v_a represents a product of two factors $c \cdot d$? (1 pt)

(e) At train time and for a given input $\mathbf{u}_t$, do all of the $\text{FFN}^{(s)}$ receive gradient flow during backpropagation? And do all of the $\text{FFN}^{(r)}$ receive gradient flow during backpropagation? Explain with reference to your previous answers. (2 pts)

(f) Suggest a drop-in replacement for Eq. (21) which guarantees gradient flow through all FFNs. (1 pt)

(g) **Bonus:** Suggest a regularization side loss, favoring balanced average scores $\bar{s}_i = \frac{1}{n}\sum_{t=1}^{n} s_{i,t}$ across a large training corpus. Justify your choice. (2 bonus pts)

(h) **Bonus:** Suggest a regularization side loss that does not merely favor balanced scores $\bar{s}_i$ but explicitly balances the *usage* of experts across a large training corpus when using Top-k routing. Justify your choice. (2 bonus pts)

Note: This exercise is taken verbatim from last year's exam, except that none of the tasks were marked as "bonus". The adjoint was not presented explicitly this year, but we reproduce the exercise "as is" from a previous year when it was.