EECS 182 Deep Neural Networks
Fall 2026 Gireeja Ranade

Discussion 11

1. Query-Key-Value Mechanics in Self-Attention

Self-attention is a core building block of transformers. In this problem, we study query–key–value interactions among three hidden states from an RNN encoder. The hidden states and projection matrices are:

$$h_1 = \begin{bmatrix}5\\1\end{bmatrix} \quad h_2 = \begin{bmatrix}4\\3\end{bmatrix} \quad h_3 = \begin{bmatrix}-2\\6\end{bmatrix} \quad W_q = \begin{bmatrix}1 & 0\\2 & 9\end{bmatrix} \quad W_k = \begin{bmatrix}0 & 1\\6 & 0\end{bmatrix} \quad W_v = \begin{bmatrix}4 & 3\\9 & 1\end{bmatrix}$$

Here, h_i is the hidden state at time step i. Each q_i, k_i, v_i is determined by $q_i = W_q h_i$, $k_i = W_k h_i$, $v_i = W_v h_i$.

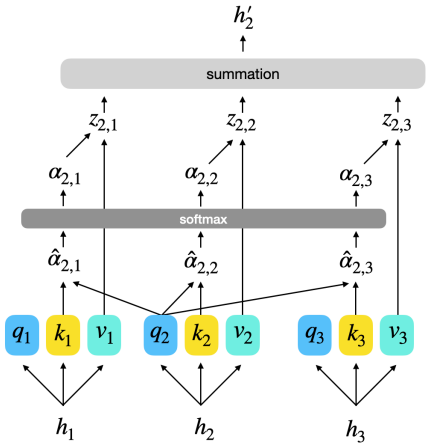


Figure 1: Self-attention at the second sequence position. Each $z_{2,j} = \alpha_{2,j} v_j$; in particular, $z_{2,1} = \alpha_{2,1} v_1$.

Use unscaled dot-product scores $\hat{\alpha}_{i,j} = q_i^T k_j$ in this exercise, omitting the usual $1/\sqrt{d_k}$ factor.

(a) **Compute $\hat{\alpha}_{2,1}$, $\hat{\alpha}_{2,2}$, $\hat{\alpha}_{2,3}$.**

Solution:

$$\hat{\alpha}_{2,1} = q_2^T k_1 = (W_q h_2)^T (W_k h_1) = (\begin{bmatrix}1 & 0\\2 & 9\end{bmatrix}\begin{bmatrix}4\\3\end{bmatrix})^T (\begin{bmatrix}0 & 1\\6 & 0\end{bmatrix}\begin{bmatrix}5\\1\end{bmatrix}) = \begin{bmatrix}4 & 35\end{bmatrix}\begin{bmatrix}1\\30\end{bmatrix} = 1054 \quad (1)$$

$$\hat{\alpha}_{2,2} = q_2^T k_2 = (W_q h_2)^T (W_k h_2) = \begin{bmatrix}4 & 35\end{bmatrix} (\begin{bmatrix}0 & 1\\6 & 0\end{bmatrix}\begin{bmatrix}4\\3\end{bmatrix}) = \begin{bmatrix}4 & 35\end{bmatrix}\begin{bmatrix}3\\24\end{bmatrix} = 852 \quad (2)$$

$$\hat{\alpha}_{2,3} = q_2^T k_3 = (W_q h_2)^T (W_k h_3) = \begin{bmatrix}4 & 35\end{bmatrix} (\begin{bmatrix}0 & 1\\6 & 0\end{bmatrix}\begin{bmatrix}-2\\6\end{bmatrix}) = \begin{bmatrix}4 & 35\end{bmatrix}\begin{bmatrix}6\\-12\end{bmatrix} = -396 \quad (3)$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

(b) Figure 1 shows self-attention at the second sequence position. **Write equations for** $h'_2 = \text{SelfAttention}(h_1, h_2, h_3)$**.** You may define intermediate variables.

Solution:

$$\begin{aligned} h'_2 &= \textit{SelfAttention}(h_1, h_2, h_3) \\ &= z_{2,1} + z_{2,2} + z_{2,3} = \alpha_{2,1} v_1 + \alpha_{2,2} v_2 + \alpha_{2,3} v_3 \end{aligned}$$

where $\boldsymbol{\alpha_2} = \text{softmax}(\hat{\boldsymbol{\alpha}}_\mathbf{2})$, $\hat{\boldsymbol{\alpha}}_\mathbf{2} = (\hat{\alpha}_{2,1}, \hat{\alpha}_{2,2}, \hat{\alpha}_{2,3})^T$, and $z_{2,j} = \alpha_{2,j} v_j$. The logits were computed in part (a).

(c) For simplicity, replace softmax with **hard attention**, an operation that takes the three scores $(\hat{\alpha}_{2,1}, \hat{\alpha}_{2,2}, \hat{\alpha}_{2,3})$ and outputs a one-hot vector: 1 at the position of the largest input and 0 at the other two positions. What is h'_2 in this case?

Solution:

With this one-hot argmax rule, $\boldsymbol{\alpha_2}$ will simply be $[1, 0, 0]$ and thus $h'_2 = z_{2,1} = v_1 = W_v h_1$ which is equal to:

$$\begin{bmatrix} 4 & 3 \\ 9 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 23 \\ 46 \end{bmatrix}$$

(d) In practice, it is common to have multiple self-attention operations happening in one layer. Each self-attention block is referred to as a *head*, and thus the entire block is typically called *Multi-head Self-Attention (MSA)* in papers. **What's the benefit of having multiple heads?** (*Hint*: why do we want multiple kernel filters in ConvNets?)

Solution: Multiple heads have separate query, key, and value projections, so they can learn different attention patterns and different features. For example, one head can attend to a subject while another attends to a relevant modifier. Their outputs are concatenated and projected to combine this information. A single head uses one attention distribution for all of its value features; several heads can therefore represent different relationships in parallel. This is analogous to using multiple convolutional filters to detect different features.

 All Rights Reserved. This may not be publicly shared without explicit permission.

2. NoPE (No Positional Encoding)

In models with a decoder-only architecture (such as GPT) using causal attention, it might be possible for the attention layer itself to learn to encode positional information.

Suppose we have a sequence of tokens $\{< beg >, x_2, \ldots, x_n\}$ where $< beg >$ is a special token indicating the beginning of the sequence. Additionally, suppose we learn an embedding where $< beg >$ is represented by d-dimensional vector $[1, 1, e_{b,3}, \ldots, e_{b,d}]$ and each other token x_i is represented by $[1, 0, e_{i,3}, \ldots, e_{i,d}]$ for $i = 2, 3, \ldots, n$. Here, $e_{b,j}$ and $e_{i,j}$ can be arbitrary when $j \geq 3$.

Additionally, suppose we have an attention head with query, key, and value matrices $W^Q \in \mathbb{R}^{d\times d}$, $W^K \in \mathbb{R}^{d\times d}$, $W^V \in \mathbb{R}^{d\times d}$ with the following learned values:

$$W^K = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad W^V = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

and W^Q is arbitrary.

Use row-vector token embeddings, with weight matrices multiplied on the right.

(a) **Write the embedding matrix $X \in \mathbb{R}^{n\times d}$ for the token sequence.**

Solution: Given the token sequence $\{< beg >, x_2, \ldots, x_n\}$, the learned embeddings are given by $X \in \mathbb{R}^{n\times d}$ where each row corresponds to the embedding of a token:

$$X = \begin{bmatrix} 1 & 1 & e_{b,3} & \cdots & e_{b,d} \\ 1 & 0 & e_{2,3} & \cdots & e_{2,d} \\ 1 & 0 & e_{3,3} & \cdots & e_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & e_{n,3} & \cdots & e_{n,d} \end{bmatrix}.$$

(b) **Compute the key, query, and value matrices $K = XW^K$, $Q = XW^Q$, and $V = XW^V$.**

Solution: Applying the key, query, and value matrices yields $K \in \mathbb{R}^{n\times d}$, $Q \in \mathbb{R}^{n\times d}$, and $V \in \mathbb{R}^{n\times d}$ where

$$K = XW^K = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \quad Q = XW^Q, \quad V = XW^V = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

(c) **Calculate the masked scores $\alpha = \text{Mask}(QK^T)$ and the attention weights after applying softmax to each row.**

Solution: Our (unnormalized) causal attention scores are given by

$$\alpha = \text{Mask}(QK^T) = \text{Mask}\left(\begin{bmatrix} s_1 & s_1 & s_1 & \cdots & s_1 \\ s_2 & s_2 & s_2 & \cdots & s_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_n & s_n & s_n & \cdots & s_n \end{bmatrix}\right) = \begin{bmatrix} s_1 & -\infty & -\infty & \cdots & -\infty \\ s_2 & s_2 & -\infty & \cdots & -\infty \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_n & s_n & s_n & \cdots & s_n \end{bmatrix},$$

where $s_t = \sum_{i=1}^{d} Q_{t,i}$ and $\text{Mask}(\cdot)$ applies the causal mask.
After applying softmax to each row, we get the attention weights

$$\text{Softmax}(\alpha) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}.$$

(d) **Apply attention to V and compute the output for token x_t.**

Solution: The output of the attention head is given by

$$\begin{aligned} \text{Attention}(Q, K, V) &= \text{Softmax}(\alpha)V \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \frac{1}{2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

Each row of this output corresponds to one token; the output for token x_t is given by the row vector $[\frac{1}{t}, 0, \ldots, 0]$.

(e) **Explain why this output encodes the position t and how it can pass positional information to subsequent layers.**

Solution: The first coordinate is $\frac{1}{t}$, which uniquely determines the token's position t. The output of this attention head can be passed to subsequent layers in the model, so subsequent layers can learn functions of the position t. This can be advantageous because strategies like RoPE can fail on context lengths if the parameters ω are not set appropriately.
Note that this is just an *existence* argument showing that it is theoretically possible for the attention layer to learn positional information.

Reference: https://arxiv.org/abs/2305.19466.

 All Rights Reserved. This may not be publicly shared without explicit permission.

3. Transformer Architecture

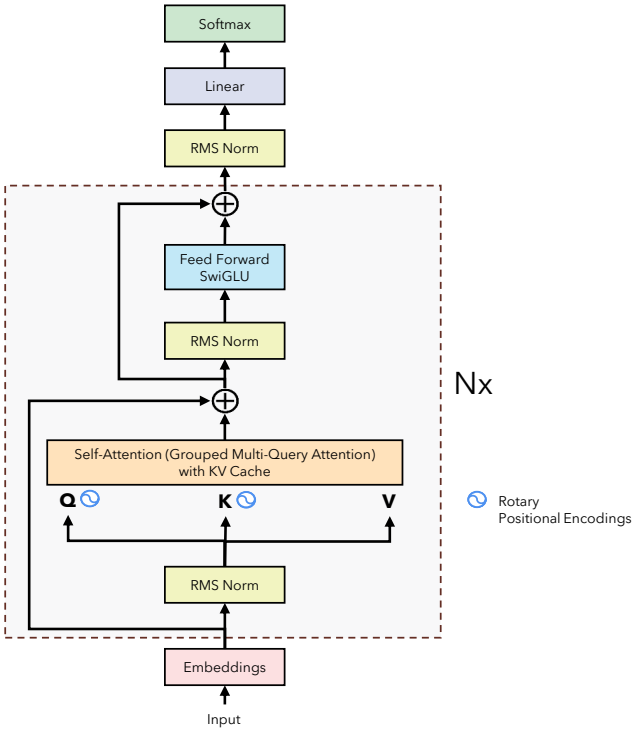


Figure 2: LLaMA architecture diagram. Many modern transformers use rotary positional embeddings (RoPE) and the SwiGLU activation function (source: https://github.com/hkproj/pytorch-llama-notes/).

(a) **What is the advantage of multi-head attention?** Give some examples of linguistic structures that can be found using multi-head attention (e.g. subject-verb agreement, translation between languages).

Solution: Separate heads can attend to different positions and representation subspaces simultaneously. For example, one head may track a subject–verb relationship while another resolves a pronoun's referent. In translation, this lets the model combine information from several input words when choosing an output token or its grammatical form.

(b) Consider hard attention with one-hot argmax choices and residual connections. Fix the argmax choices. **After n layers, what is the maximum number of input positions contributing to a node through the selected value edges and residual paths?** Let the sequence length be L.
With a single head? With two heads? With h heads?

Solution: A node receives at most h selected value inputs plus its residual input, giving a branching factor of at most $h+1$. Thus the bounds are $\min\{L, 2^n\}$ for one head, $\min\{L, 3^n\}$ for two heads, and $\min\{L, (h+1)^n\}$ for h heads.

Overlapping paths can reduce these counts. The bounds condition on fixed argmax selections; other positions' keys can still affect which selections are made.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(c) For input sequences of length M and output sequences of length N, **what are the complexities of (1) Non-Causal Self-Attention[1] (on the input sequence), (2) Non-Causal Self-Attention (on the input sequence), but with multi-query attention, (3) Cross Attention (from the output sequence to the input sequence), (4) Causal Self-Attention (on the output sequence).** Let d be the hidden dimension of the network (both the encoder and the decoder part), and h be the number of heads.

Recall: Normal multi-head attention is:

$$\text{MultiHeadedAttention}(X, X, X) = \text{Concat}_{i \in [\#\text{ heads}]} \left(\text{Attention} \left(XW_i^Q, XW_i^K, XW_i^V \right) \right) W^O$$

For this question, we assume W_i^Q, W_i^K, W_i^V are of shape $d \times d$, $W^O \in \mathbb{R}^{hd \times d}$, and X has shape $M \times d$ where each row of X is a token embedding. For cross-attention, the query sequence has N rows and the key/value sequence has M rows. We also assume that we use naive matrix multiplication, that is, if A, B are of shape $m \times n, n \times k$, then AB takes mnk multiply-accumulate operations.

With multi-query attention, there is just one W^K, W^V, thus:

$$\text{MultiQueryAttention}(X, X, X) = \text{Concat}_{i \in [\#\text{ heads}]} \left(\text{Attention} \left(XW_i^Q, XW^K, XW^V \right) \right) W^O$$

Solution:

i. $\mathcal{O}((Md + M^2)dh)$
ii. $\mathcal{O}((Md + M^2)dh)$
iii. $\mathcal{O}((Md + Nd + MN)hd)$
iv. $\mathcal{O}((Nd + N^2)dh)$

As one example, the encoder-self-attention takes $3Md^2$ operations in total for XW_i^Q, XW_i^K, and XW_i^V in one head. Computing the attention matrix takes M^2d operations, and $O(M^2)$ for computing its softmax. Multiplying the attention with value matrix takes M^2d operations. This is repeated for h heads. Finally the outputs of all the heads are linearly projected at a cost of Mhd^2 operations. Together:

$$\mathcal{O}((3Md^2 + M^2d + M^2 + M^2d)h + Mhd^2) = \mathcal{O}(Md^2h + M^2dh)$$

(d) **How many operations does the multi-query attention save compared to multi-head attention? What are some other advantages?**

Solution: In multi-query attention, XW^K and XW^V are computed only once instead of h times. For each of these computations, we compute only one matrix multiplication instead of h, saving $2Md^2(h-1)$ operations in total.

Another advantage during inference time is that we only need to store one set of key and value projections instead of h sets, which reduces memory usage.

Contributors: Matteo Guarrera, Mert Cemri, Sizhe Chen, Gireeja Ranade, Kevin Li, Lance Mathias, Anant Sahai, Olivia Watkins, Jerome Quenum, Saagar Sanghavi, Yuxi Liu, Qiyang Li.

[1] This is the "normal" self-attention from lecture.

 All Rights Reserved. This may not be publicly shared without explicit permission.