

Assignment 2: Efficient model computation

42 points + 4 optional points

You may use a result requested in an earlier part even if you have not proved it.

Notation. Use the multiply-add convention from Lecture 6: one accumulation $z \leftarrow z + xy$ costs one multiply-add, so multiplying an $a \times b$ matrix by a $b \times c$ matrix costs abc multiply-adds. Treat transposes as free. $\text{diag}(a_1, \dots, a_r)$ denotes the diagonal matrix with diagonal entries $a_1, \dots, a_r$.

1 Conditional computation with MoE (10 points)

A mixture-of-experts (MoE) layer evaluates a few experts per token [1]. Following Lecture 4 and GShard [2], we batch tokens by expert and combine the expert outputs for each token.

There are T token vectors $h_t \in \mathbb{R}^d$ and E experts with separate parameters. Expert e is a two-layer multilayer perceptron (MLP):

$$F_e(h) = W_{e,2}\phi(W_{e,1}h), \quad W_{e,1} \in \mathbb{R}^{m \times d}, \quad W_{e,2} \in \mathbb{R}^{d \times m},$$

where $\phi(z) = \max\{z, 0\}$ acts entrywise. The router computes $s_t = W_r h_t$, with $W_r \in \mathbb{R}^{E \times d}$, and selects the k largest scores, where $1 \leq k \leq E$. Let S_t be the selected expert indices, breaking ties by smaller index. The output is

$$w_{t,e} = \frac{e^{s_{t,e}}}{\sum_{j \in S_t} e^{s_{t,j}}}, \quad y_t = \sum_{e \in S_t} w_{t,e} F_e(h_t).$$

Every selected token–expert pair is evaluated.¹

- (a) **(3 points)** Count the parameters in the router and experts. For the whole batch, compute the multiply-add counts for the router and for the selected experts. Exclude weighted output aggregation. By what factor does selecting experts *before* evaluating them reduce expert computation compared with evaluating all E experts on every token? With T, d, m, k fixed, which of these counts still grow with E ?
- (b) **(4 points)** The router's sets S_t and weights $w_{t,e}$ are available. Complete the following batched implementation.
 - (i) **(2 points)** Write pseudocode that first counts $N_e = |\{t : e \in S_t\}|$ and then fills an array of N_e assignment records for each expert. What must each record store to identify the destination token and apply its router weight? Prove that grouping the records takes $O(Tk + E)$ operations.²

¹There is no limit on the number of tokens assigned to an expert. The layer has no biases, and no residual term h_t is added to y_t .

²Each index or scalar read, write, or arithmetic operation has unit cost. This bound concerns records; it excludes copying token vectors and the computations in (ii)–(iii).

- (ii) **(1 point)** If expert e 's records contain token indices $t_1, \dots, t_{N_e}$, define its input matrix H_e . Express its output matrix Z_e using two matrix products and ϕ . Give the dimensions of H_e , $W_{e,1}H_e$, and Z_e . Skip experts with $N_e = 0$.
- (iii) **(1 point)** Starting from $y_t = 0$, give the update using column j of Z_e and its record. Prove that the procedure computes the stated y_t .
- (c) **(3 points)** Each expert has its own device. All devices start together, and expert e takes τN_e time, where $\tau > 0$. Assume T is a multiple of E . Compare balanced loads $N_e = Tk/E$ for all e with concentrated loads $N_e = T$ for $e \leq k$ and $N_e = 0$ otherwise. Compute the batch completion time and total expert multiply-add count for each case. Give the ratio of concentrated to balanced completion time, and prove that no assignment of k distinct experts per token can finish faster than the balanced case.

2 Latent attention and an indexer (12 points)

Lectures 5–6 combine multi-head latent attention (MLA) with position selection in DeepSeek Sparse Attention (DSA) [3]. We use a simplified indexer below.

Consider one decoding step with n cached positions and H heads of dimension d . Each position has a shared latent vector $c_i \in \mathbb{R}^{d_c}$. The key and value for head h are

$$k_{h,i} = U_{K,h}^\top c_i, \quad v_{h,i} = U_{V,h}^\top c_i, \quad U_{K,h}, U_{V,h} \in \mathbb{R}^{d_c \times d}.$$

For a query $q_h \in \mathbb{R}^d$, the full attention output is

$$s_{h,i} = \frac{q_h^\top k_{h,i}}{\sqrt{d}}, \quad a_{h,i} = \frac{e^{s_{h,i}}}{\sum_{j=1}^n e^{s_{h,j}}}, \quad o_h = \sum_{i=1}^n a_{h,i} v_{h,i}.$$

Count matrix products needed to compute o_h from q_h and the cached vectors specified in each part, including query transformations. The key-value (KV) cache contains vectors retained for future steps; its size excludes model matrices and temporary arrays.³

- (a) **(2 points)** A full KV cache stores $k_{h,i}$ and $v_{h,i}$ for every head and position. Compute its size in scalars and the multiply-add counts for all scores and weighted value sums.
- (b) **(4 points)** Suppose the cache stores only $c_1, \dots, c_n$. Use associativity, as in Assignment 1, Problem 1(a), to avoid reconstructing the keys and values.
- (i) **(2 points)** Find $\tilde{q}_h \in \mathbb{R}^{d_c}$ and $z_h \in \mathbb{R}^{d_c}$ such that $s_{h,i} = \tilde{q}_h^\top c_i / \sqrt{d}$ and $o_h = U_{V,h}^\top z_h$. Derive both identities. (*Hint:* Transform the query once per head; for the output, sum the weighted latent vectors before applying $U_{V,h}^\top$.)

³Exclude computing the input queries, constructing or appending cache entries, and any projection after o_h . The problem has no positional terms. Exclude score scaling, softmax, and selection from the matrix-product counts.

- (ii) **(1 point)** Count the multiply-adds in these four stages: query transformation, scores, weighted latent sum, and output projection. Compute the total over H heads and the cache size.
- (iii) **(1 point)** Compared with (a), when does this cache store strictly fewer scalars? When does the procedure use strictly more multiply-adds? Give an inequality in H, d, d_c, n for each comparison.
- (c) **(6 points)** An indexer selects K positions for all heads, where $1 \leq K \leq n$. The cache now stores c_i and a search key $k_i^I \in \mathbb{R}^{d_I}$ for every position. From an input query $q^I \in \mathbb{R}^{d_I}$, compute $I_i = (q^I)^\top k_i^I$ and select the K largest scores, breaking ties by smaller position index. Call this set S . Each head uses the procedure in (b), computing scores and applying softmax only on S , and summing latent vectors over S . Keep all n latent vectors and search keys for future steps.
- (i) **(2 points)** Compute the cache size in scalars and the total multiply-add count $C(K)$, including the indexer and the products $U_{K,h} q_h$. Exclude forming q^I and the cached search keys. Does selecting fewer positions reduce this cache size?
- (ii) **(3 points)** Take $H = 4, d = 10, d_c = 20, d_I = 2$, and $n = 100$. Under a budget of 5000 multiply-adds per step, find the largest allowed K and verify that $K + 1$ exceeds the budget. Compute the cache size in scalars and cost per step for full KV, MLA over all positions, and indexed MLA at this K . Which procedures meet both this compute budget and a cache budget of 2500 scalars? Compare the indexer's score cost with the main-attention work saved by selecting K positions.
- (iii) **(1 point)** For $H = d = d_c = 1, n = 2, q = 0, U_K = U_V = 1, c_1 = 0, c_2 = 1$, and $S = \{1\}$, compute the full output o using both positions and the output o^S obtained by applying softmax and summing values only over S .
- (d) **(2 optional points)** Concentrated attention weights, as observed in H₂O (Heavy-Hitter Oracle) [4], motivate selecting a set that retains a large share of one head's attention weight. Fix one head's full weights $a_i > 0$, with $\sum_i a_i = 1$. For a size- K set S , let $m(S) = \sum_{i \in S} a_i$ and define $a_i^S = a_i / m(S)$ on S , and $a_i^S = 0$ elsewhere. Using the KL divergence from Assignment 1, Problem 3, prove

$$\text{KL}(a^S \| a) = -\log m(S) \leq -\log(1 - \varepsilon) \quad \text{if } m(S) \geq 1 - \varepsilon, \quad 0 \leq \varepsilon < 1.$$

Here log is natural. With all a_i available, which size- K set minimizes this divergence? Prove your answer, breaking ties by smaller index. Explain why ranking the attention scores $s_{h,i}$ gives the same set without computing softmax.⁴

⁴The minimizing set is for one head's attention scores $s_{h,i}$. The shared indexer may choose a different set.

3 Allocating precision in quantization (10 points)

Mixed-precision quantization assigns more bits to some weights than to others. We consider a row allocation problem motivated by HAWQ-V3 [5].

Let $W \in \mathbb{R}^{p \times d}$ be a linear layer's weight matrix and let $X \in \mathbb{R}^{d \times m}$ contain m calibration inputs. The error of a quantized matrix Q is

$$D(Q) = \frac{1}{m} \|(Q - W)X\|_F^2, \quad \|A\|_F^2 = \sum_{i,j} A_{ij}^2.$$

Fix grids $\mathcal{Q}_L \subset \mathcal{Q}_H$ with 2^{b_L} and 2^{b_H} values, where $1 \leq b_L < b_H$ are integers. Round each entry of W to the nearest value in each grid, breaking ties toward the smaller value, to obtain the low-precision matrix Q^L and high-precision matrix Q^H . For a set of rows S , let $Q(S)$ use row i of Q^H when $i \in S$ and row i of Q^L otherwise. Choose S to minimize $D(Q(S))$ subject to the storage budget in (b).

- (a) **(2 points)** Write w_i, q_i^L, q_i^H for the transposes of row i of W, Q^L, Q^H , and define

$$D_i^L = \frac{1}{m} \|X^\top (q_i^L - w_i)\|_2^2, \quad D_i^H = \frac{1}{m} \|X^\top (q_i^H - w_i)\|_2^2.$$

Find Δ_i in terms of these errors so that

$$D(Q(S)) = D(Q^L) - \sum_{i \in S} \Delta_i.$$

Prove the identity by summing squared errors over output rows, without any assumption on correlations in X .

- (b) **(5 points)** A low-precision weight uses b_L bits and a high-precision weight uses b_H bits. In addition to the weights, store both grids and one bit per row indicating its precision. The grids and row indicators use C bits in total, independent of S . The storage budget is B bits. Assume it covers this cost and all pd weights at low precision, i.e., $B \geq C + pdb_L$.⁵
- (i) **(1 point)** Find the extra storage c for each row that is changed to high precision and the largest number $L \leq p$ of such rows allowed by the budget.
 - (ii) **(2 points)** Write pseudocode that starts with $S = \emptyset$ and traverses all p rows in decreasing order of Δ_i , breaking ties by smaller index. Add a row if and only if it strictly reduces D and the new set fits the budget. Express both conditions using $\Delta_i, |S|$, and L , with $|S|$ measured before adding the row. Return S after the traversal.

⁵Count the stored grids, row indicators, and quantized weights only. Do not round storage up to whole bytes or machine words, or count temporary memory used during calibration.

- (iii) **(2 points)** Prove that the returned set S_{alg} minimizes $D(Q(S))$ over all sets allowed by the budget. (*Hint:* Every row replacement costs the same number of bits. Use an exchange argument and the identity in (a).)

- (c) **(3 points)** To examine the role of calibration data, take

$$W = \begin{pmatrix} 2/5 & 0 \\ 0 & 1/5 \end{pmatrix}, \quad \mathcal{Q}_L = \{0, 1\}, \quad \mathcal{Q}_H = \{0, 1/3, 2/3, 1\}, \quad B = C + 6.$$

For $X_A = I_2$ and $X_B = \text{diag}(1, 3)$, compute Q^L, Q^H , the two benefits, and the unique optimal set S under the budget. Can any deterministic rule that receives W , the grids, and the storage parameters, but not X , choose an optimal set for both calibration matrices? Justify your answer.

4 Computing Muon's matrix normalization (10 points)

Muon (MomentUm Orthogonalized by Newton–Schulz, Lecture 3) approximately replaces the nonzero singular values of a momentum matrix by 1 [6]. We analyze an iteration for this normalization, starting from the momentum matrix.

Let $M \in \mathbb{R}^{m \times n}$, with $m \geq n$, have full column rank. Write its thin singular value decomposition (SVD) as

$$M = U \Sigma V^\top, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \sigma_i > 0, \quad U^\top U = V^\top V = I_n,$$

where $U \in \mathbb{R}^{m \times n}$ and $V \in \mathbb{R}^{n \times n}$. The ideal normalized matrix is the polar factor $P = UV^\top$. Consider the classical cubic Newton–Schulz iteration

$$X_0 = \frac{M}{\|M\|_F}, \quad X_{j+1} = \frac{1}{2} X_j (3I_n - X_j^\top X_j), \quad j = 0, 1, \dots$$

We use the cubic iteration to analyze convergence. The Muon implementation in [6] uses a different polynomial and a fixed iteration count.

- (a) **(2 points)** For $M = \text{diag}(10, 1)$, compute $M/\|M\|_F$ and P . Let $v_1 = (1, 0)^\top$, $v_2 = (0, 1)^\top$, and $\eta > 0$. Define the updates

$$\Delta_{\text{scalar}} = -\eta M / \|M\|_F, \quad \Delta_{\text{polar}} = -\eta P.$$

For each update, compute $\|\Delta v_1\|_2$, $\|\Delta v_2\|_2$, and their ratio $\|\Delta v_1\|_2 / \|\Delta v_2\|_2$. Which normalization gives equal update magnitudes in these two directions?

- (b) **(3 points)** Show that the iterates can be written using the same matrices U, V as

$$X_j = U \operatorname{diag}(s_{1,j}, \dots, s_{n,j}) V^\top.$$

Derive the scalar recurrence for $s_{i,j}$ and its initial value. Show that every initial value lies in $(0, 1]$.

- (c) **(3 points)** Prove that a scalar sequence following the recurrence in (b), starting in $(0, 1]$, stays in $(0, 1]$, is nondecreasing, and converges to 1. Deduce that $X_j \rightarrow P$. (*Hint:* Let f be the scalar update function derived in (b). For $0 < s \leq 1$, examine $f(s) - s$ and $1 - f(s)$, then identify possible limits.)
- (d) **(2 points)** Now let $M \neq 0$ have rank $r < n$, with the same thin SVD notation but $\sigma_1, \dots, \sigma_r > 0$ and $\sigma_{r+1} = \dots = \sigma_n = 0$.⁶ Determine the limit of X_j . Does it have orthonormal columns? Explain why an initially zero singular value remains zero.
- (e) **(2 optional points)** For a positive singular-value sequence in (b), let $e_j = 1 - s_j$. Derive a recurrence for e_j . Show that, once $e_j \leq 1/2$, the next error is at most $\frac{3}{2}e_j^2$. For $0 < s_0 < 1$, compute $\lim_{j \rightarrow \infty} e_{j+1}/e_j^2$ to establish quadratic convergence. If $s_0 = 1$, give closed-form expressions for s_j and e_j for every $j \geq 0$.

Sources

- [1] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. [Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer](#). *ICLR*, 2017. Sections 2–4.
- [2] Dmitry Lepikhin, Hyoungho Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. [GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding](#). *ICLR*, 2021. Section 2.
- [3] DeepSeek-AI. [DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models](#). 2025.
- [4] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. [H₂O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models](#). *NeurIPS*, 2023. Section 3.
- [5] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, and Kurt Keutzer. [HAWQ-V3: Dyadic Neural Network Quantization](#). *ICML*, 2021. Section 3.4.
- [6] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. [Muon: An optimizer for hidden layers in neural networks](#). Technical blog post, 2024.

⁶For $M = 0$, define the normalized matrix and optimizer update to be zero.