Softmax OP

For an input vector $x \in \mathbb{R}^N$ and grab only first B.

1. $m(x) := \max_i x_i$ — find the largest score in this block B
   (we need this to subtract to in order to prevent overflow)

2. exponentiated Differences vector: $f(x) := [e^{x_1 - m(x)} \dots e^{x_B - m(x)}]$
   ↓ $x_i - m(x)$ is ≤ 0, so safely in the range of $(0, 1]$

3. Sum of exponents (Denominator) $\ell(x) := \sum_i f(x)_i$

4. $\text{softmax}(x) := \frac{f(x)}{\ell(x)}$ Local softmax (incorrect for softmax for now)

5. For vectors $x^1, x^2 \in \mathbb{R}^B$, decompose the softmax of the concat $x = [x^1, x^2] \in \mathbb{R}^{2B}$ as

$m(x) = m([x^1\ x^2]) = \max(m(x^1), m(x^2))$: find the max between B1's local max and B2's local max

$f(x) = [\, e^{m(x^1) - m(x)} f(x^1) \quad e^{m(x^2) - m(x)} f(x^2) \,]$

$e^{m(x^1) - m(x)} \cdot e^{x_i^1 - m(x^1)} \Downarrow e^{x_i^1 - m(x)}$

6. $\ell(x) = \ell([x^1, x^2]) = e^{m(x^1) - m(x)} \ell(x^1) + e^{m(x^2) - m(x)} \ell(x^2)$, $\text{Softmax}(x) = \frac{f(x)}{\ell(x)}$

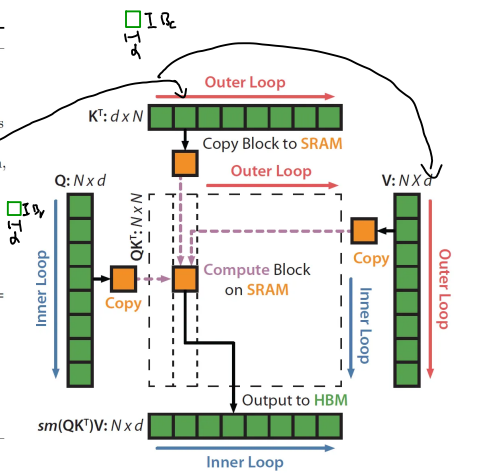
logic continues to $\frac{N}{B}$ of blocks with dim B

Alg 1 assumes Batchsize of 1.

Algorithm 1 FLASHATTENTION

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, on-chip SRAM of size M.

1: Set block sizes $B_c = \lceil \frac{M}{4d} \rceil$, $B_r = \min(\lceil \frac{M}{4d} \rceil, d)$.
2: Initialize $\mathbf{O} = (0)_{N \times d} \in \mathbb{R}^{N \times d}$, $\ell = (0)_N \in \mathbb{R}^N$, $m = (-\infty)_N \in \mathbb{R}^N$ in HBM.
3: Divide $\mathbf{Q}$ into $T_r = \lceil \frac{N}{B_r} \rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \lceil \frac{N}{B_c} \rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.
4: Divide $\mathbf{O}$ into T_r blocks $\mathbf{O}_i, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide ℓ into T_r blocks $\ell_i, \dots, \ell_{T_r}$ of size B_r each, divide m into T_r blocks $m_1, \dots, m_{T_r}$ of size B_r each.
5: **for** $1 \le j \le T_c$ **do**
6: Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.
7: **for** $1 \le i \le T_r$ **do**
8: Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.
9: On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
10: On chip, compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.
11: On chip, compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$, $\ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.
12: Write $\mathbf{O}_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1} (\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij} \mathbf{V}_j)$ to HBM.
13: Write $\ell_i \leftarrow \ell_i^{\text{new}}$, $m_i \leftarrow m_i^{\text{new}}$ to HBM.
14: **end for**
15: **end for**
16: Return $\mathbf{O}$.



1. Ceil($\frac{M}{4d}$), because we need to allocate $Q, K, V, O \in \mathbb{R}^{Nd}$
2. In HBM: $O \in \mathbb{R}^{Nd}$ (output matrix)
   $\ell \in \mathbb{R}^N$ (normalizer, initialized to all 0)
   $m \in \mathbb{R}^N$ (row max vector, initialized to $-\infty$)
3. Divide Q into $T_r = \lceil \frac{N}{B_r} \rceil$ blocks $Q_1 \dots Q_{T_r}$, $Q_i \in \mathbb{R}^{B_r \times d}$
   Divide K, V into $T_c = \lceil \frac{N}{B_c} \rceil$ each of $K_i, V_i \in \mathbb{R}^{B_c d}$
4. Divide O into T_r blocks $O_1 \dots O_{T_r}$ of size $O_i \in \mathbb{R}^{B_r \times d}$
   Divide ℓ into T_r blocks $\ell_1, \dots, \ell_{T_r}$ of size B_r, also m as well

Side: we can load ℓ and m into SRAM cuz they are so small

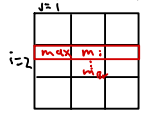
B_r (row Block Size)
↳ and two Buckets, we are still allocating space for every Q, K, V, O
B_c (col Block size), but if B_c is bigger than d, K and V take advantage of $B_c (B_c \cdot d + B_c \cdot d)$ whereas Q and O fall back to d to save resources

↳ for QK^T

Total SRAM: $2 \cdot (B_r \cdot d) + 2(B_c \cdot d) + (B_r \cdot B_c)$
if $B_c < d$, then B_c which means: $4B_c \cdot d + B_c \cdot B_c$

9. $S_{ij} = Q_i K_j^T \in \mathbb{R}^{B_r \times B_c}$
   we don't need the $N \times N$ size they are made up of tiny grid, so we only need to handle one grid at a time

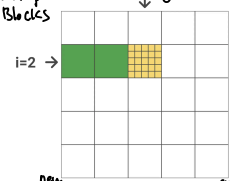
11. $m_i^{new} = \max_{j=i}(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$ $\quad \ell_i^{new} = e^{m_i - m_i^{new}} \ell_i + e^{\tilde{m}_{ij} - m_i^{new}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$



$B_r \times d$ max for each row in the small grid but also the bigger

rescaled sum from past green Blocks

rescaled from new yellow



12. $O_i \leftarrow \text{diag}(\ell_i^{new})^{-1} (\text{diag}(\ell_i) e^{m_i - m_i^{new}} O_i + e^{\tilde{m}_{ij} - m_i^{new}} \tilde{P}_{ij} V_j)$

green yellow

Green:

$\text{diag}(\ell_i) e^{m_i - m_i^{new}} O_i$

$O_i \in \mathbb{R}^{B_r \times d}$: current val of O_i for previous $j = 1, 2$

$\text{diag}(\ell_i)$: $B_r \times B_r$ diagonal matrix

$\text{diag}(\ell_i) O_i$ undivides O_i -> goes back to $\sum$

dim d

$O_i[k] = [o_1, \ldots o_d]$

$O_i[k] = \frac{1}{\ell[k]} [\text{raw } o_1, \ldots \text{raw } o_d]$

$\text{diag}(\ell_i)[k] \cdot O_i[k] = [\text{raw } o_1, \ldots \text{raw } o_d]$

$\text{raw}_c = \sum_{\text{past keys } p} e^{S_{kp} - m_i} \cdot V_{pc}$

$\left[e^{m_i - m_i^{new}} \sum_p e^{S_{kp} - m_i} \cdot V_{p1} \ldots e^{m_i - m_i^{new}} \sum e^{S_{kp} - m_i} V_{pd} \right]$

Yellow: $e^{\tilde{m}_{ij} - m_i^{new}} \tilde{P}_{ij} V_j$

$\tilde{P}_{ij} \in \mathbb{R}^{B_r \times B_c}$ ↓ score $Q_i K_j^T$

$\tilde{P}_{ij}[k] = [e^{S_{k,1} - \tilde{m}_{ij}}, e^{S_{k,2} - \tilde{m}_{ij}}, \ldots, e^{S_{k,B_c} - \tilde{m}_{ij}}]$

$V_j \in \mathbb{R}^{B_c \times d}$

$\tilde{P}_{ij} V_j[k] = [\sum_{q=1}^{B_c} e^{S_{kq} - \tilde{m}_{ij}} V_{q1}, \sum_{q=1}^{B_c} e^{S_{kq} - \tilde{m}_{ij}} V_{q2} \ldots]$

$e^{\tilde{m}_{ij} - m_i^{new}} \left(\sum_{q=1}^{B_c} e^{S_{kq} - \tilde{m}_{ij}} V_{q1} \right) = \sum_{q=1}^{B_c} e^{S_{kq} - m_i^{new}} V_{q1}$

Green + Yellow

$\text{part } A[k] = [\sum_{\text{past } p} e^{S_{kp} - m_i^{new}} V_{p1} \ldots, \sum_{\text{past } p} e^{S_{kp} - m_i^{new}} V_{pd}]$

$\text{part } B[k] = [\sum_q e^{S_{kq} - m_i^{new}} V_{q1} \ldots, \sum_q e^{S_{kq} - m_i^{new}} V_{qd}]$

$A + B = \sum_{p+q} e^{S_{k,all} - m_i^{new}} V_{q,1} \in \mathbb{R}^{B_r \times d}$

Final $\left[\frac{1}{\ell_1} \ddots \frac{1}{\ell_{Br}} \right] \times (\text{part } A + \text{part } B) = O_i^{new}$

$B_r \times B_r$ $\quad B_r \times d$ $\quad B_r \times d$

Complexity Analysis

$Q, K, V, O \in \mathbb{R}^{N \times d}$, l and $m(N)$ in HBM

Total HBM mem: $4Nd + 2N$

Space complexity $= O(N)$ note d is much smaller than N

IO (HBM Access) Complexity: $O(N^2 d^2 M^{-1})$

1. Num of Blocks $T = \frac{N}{B} = \frac{N}{M/4d} = \frac{4Nd}{M}$
2. Total iter: $T_r \times T_c \sim \frac{16N^2d^2}{M^2}$
3. Data transferred per iter: $B \cdot 4d = M$
4. Total HBM Access = Total iter x data per iter $= \frac{16N^2d^2}{M}$