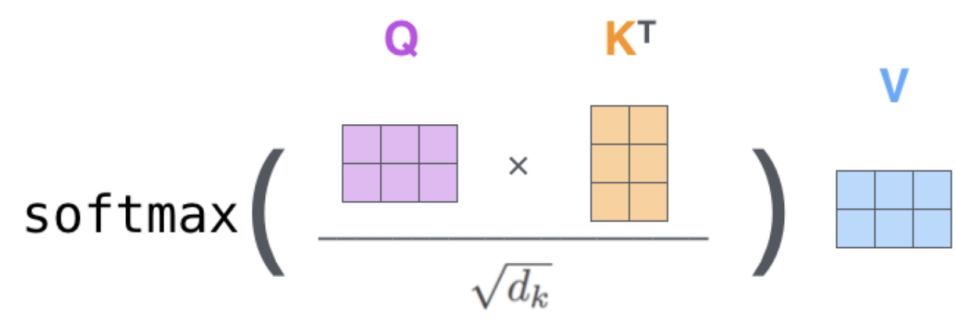


FlashAttention

Attention



Softmax

```
def softmax(x):
    x_max = x.max()
    x_exp = torch.exp(x - x_max)
    x_exp_sum = x_exp.sum()
    return x_exp / x_exp_sum
```

1. `sub block softmax + x_max() + x_exp_sum()`
2. `max() exp_sum()`
3. `elementwise softmax softmax`
 1. `sum max`
 2. `exp exp`
 3. `sum`

1 qk 3 v softmax qkv

softmax 3 1. max 2. sum 3

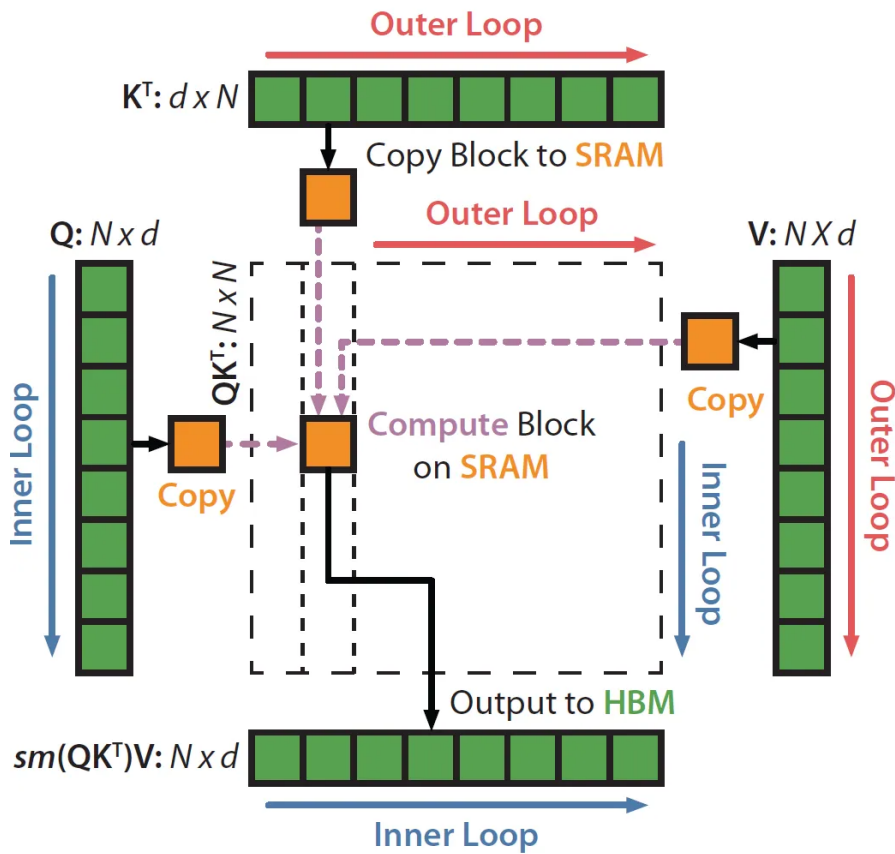
Flash Attention

Algorithm 2 FLASHATTENTION Forward Pass

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, on-chip SRAM of size M , softmax scaling constant $\tau \in \mathbb{R}$, masking function MASK, dropout probability p_{drop} .

- 1: Initialize the pseudo-random number generator state \mathcal{R} and save to HBM.
- 2: Set block sizes $B_c = \lceil \frac{M}{4d} \rceil, B_r = \min(\lceil \frac{M}{4d} \rceil, d)$.
- 3: 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.
- 4: 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} into $T_c = \lceil \frac{N}{B_c} \rceil$ 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.
- 5: 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.
- 6: **for** $1 \leq j \leq T_c$ **do**
- 7: Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.
- 8: **for** $1 \leq i \leq T_r$ **do**
- 9: Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.
- 10: On chip, compute $\mathbf{S}_{ij} = \tau \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
- 11: On chip, compute $\mathbf{S}_{ij}^{\text{masked}} = \text{MASK}(\mathbf{S}_{ij})$.
- 12: On chip, compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}^{\text{masked}}) \in \mathbb{R}^{B_r}, \tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij}^{\text{masked}} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.
- 13: On chip, compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}, \ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.
- 14: On chip, compute $\tilde{\mathbf{P}}_{ij}^{\text{dropped}} = \text{dropout}(\tilde{\mathbf{P}}_{ij}, p_{\text{drop}})$.
- 15: Write $\mathbf{O}_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1} (\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij}^{\text{dropped}} \mathbf{V}_j)$ to HBM.
- 16: Write $\ell_i \leftarrow \ell_i^{\text{new}}, m_i \leftarrow m_i^{\text{new}}$ to HBM.
- 17: **end for**
- 18: **end for**
- 19: Return $\mathbf{O}, \ell, m, \mathcal{R}$.

- 1-5 Outer Loop
- 6-7 \mathbf{K} \mathbf{V} Outer Loop
- 8 \mathbf{Q} (Inner Loop)
- 9 \mathbf{QKV} SRAM (Copy Block to SRAM)
- 10 \mathbf{S}_{ij} (Compute Block on SRAM)
- 11 \mathbf{S}_{ij} mask (Compute Block on SRAM)
- 12 m, ℓ (Compute Block on SRAM)
- 13 m, ℓ (Compute Block on SRAM)
- 14 dropout (Compute Block on SRAM)
- 15 \mathbf{O}_i HBM (Output to HBM)
- 16 ℓ_i, m_i HBM (Output to HBM)



FlashAttention3

1. ☐ Hopper☐ wgmma☐ cuda cores☐ tensor cores☐ 1D☐ 2D☐

Revision #7

Created 8 March 2025 10:18:23 by Colin

Updated 9 March 2025 07:25:26 by Colin