Tensor Attention Training: Provably Efficient Learning of Higher-order Transformers

Yingyu Liang, Zhenmei Shi, Zhao Song, Yufa Zhou

香 港 大 學 THE UNIVERSITY OF HONG KONG · WISCONSIN UNIVERSITY OF WISCONSIN–MADISON · Berkeley UNIVERSITY OF CALIFORNIA · Penn UNIVERSITY of PENNSYLVANIA

SCAN ME

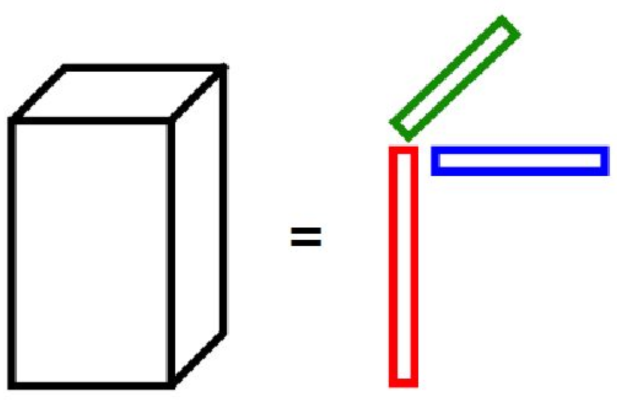
Background

$\otimes$ Kronecker product and $\oslash$ Kathri-Rao product

3-Way Outer Product

$$\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$$

$$x_{ijk} = a_i b_j c_k$$



Rank-1 Tensor

Review: Matrix Kronecker Product

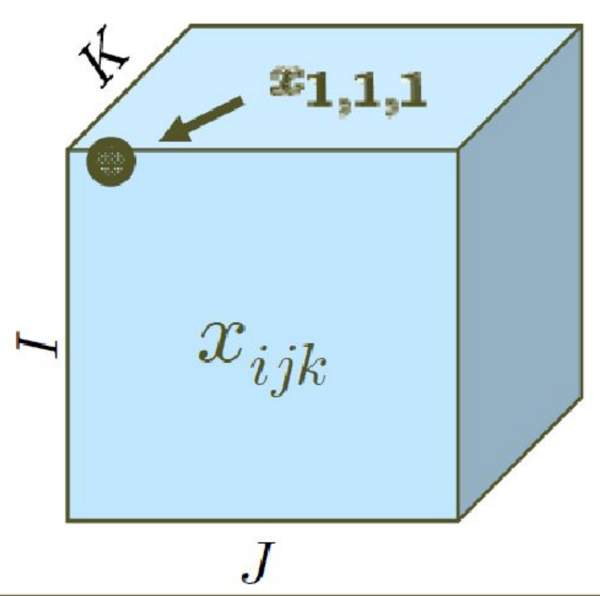
$$\underset{M \times N}{\mathbf{A}} \otimes \underset{P \times Q}{\mathbf{B}} = \underset{MP \times NQ}{\begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1N}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2N}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1}\mathbf{B} & a_{M2}\mathbf{B} & \cdots & a_{MN}\mathbf{B} \end{bmatrix}}$$

$$= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_N \otimes \mathbf{b}_Q \end{bmatrix}$$

Matrix Khatri-Rao Product

$$\underset{M \times R}{\mathbf{A}} \oslash \underset{N \times R}{\mathbf{B}} = \underset{MN \times R}{\begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_R \otimes \mathbf{b}_R \end{bmatrix}}$$

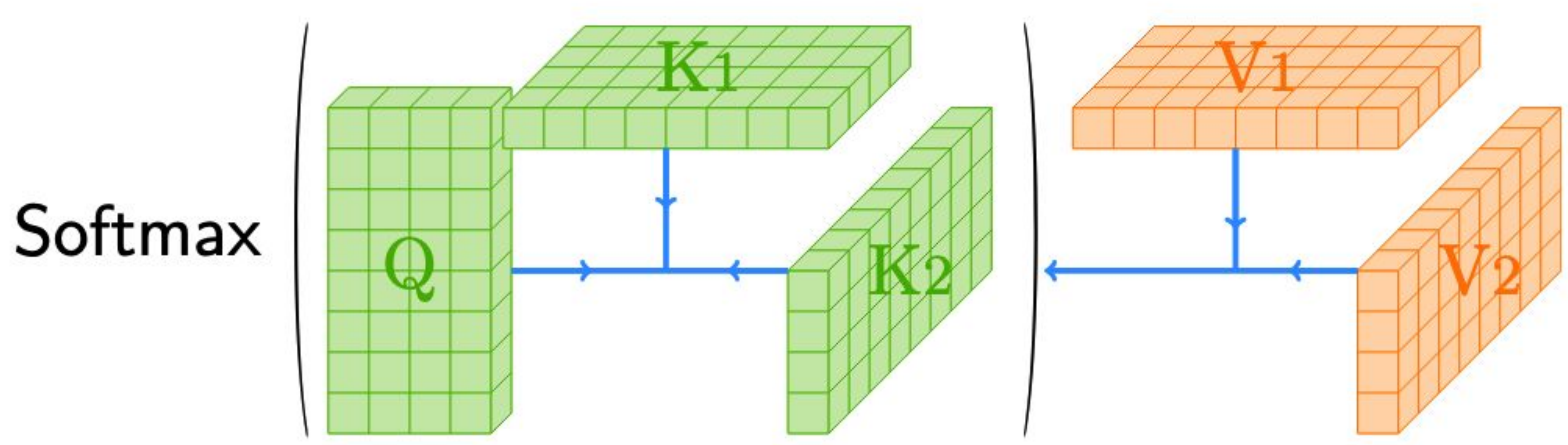
An $I \times J \times K$ tensor



3rd order tensor
mode 1 has dimension I
mode 2 has dimension J
mode 3 has dimension K

Source: CSE 6363 Machine Learning from UT Arlington

Motivation



Tensor Attention, defined as $\mathsf{Softmax}(Q(K_1 \oslash K_2)^\top)(V_1 \oslash V_2)$, is a higher-order generalization of matrix attention that can capture high-order/multi-view information intrinsically. Meanwhile, it faces a cubic computational complexity bottleneck. Therefore, in this work, we pose the following question:

Can we achieve almost linear time for gradient computation in Tensor Attention Training?

Problem Setup

Definition 1 (Tensor attention optimization)

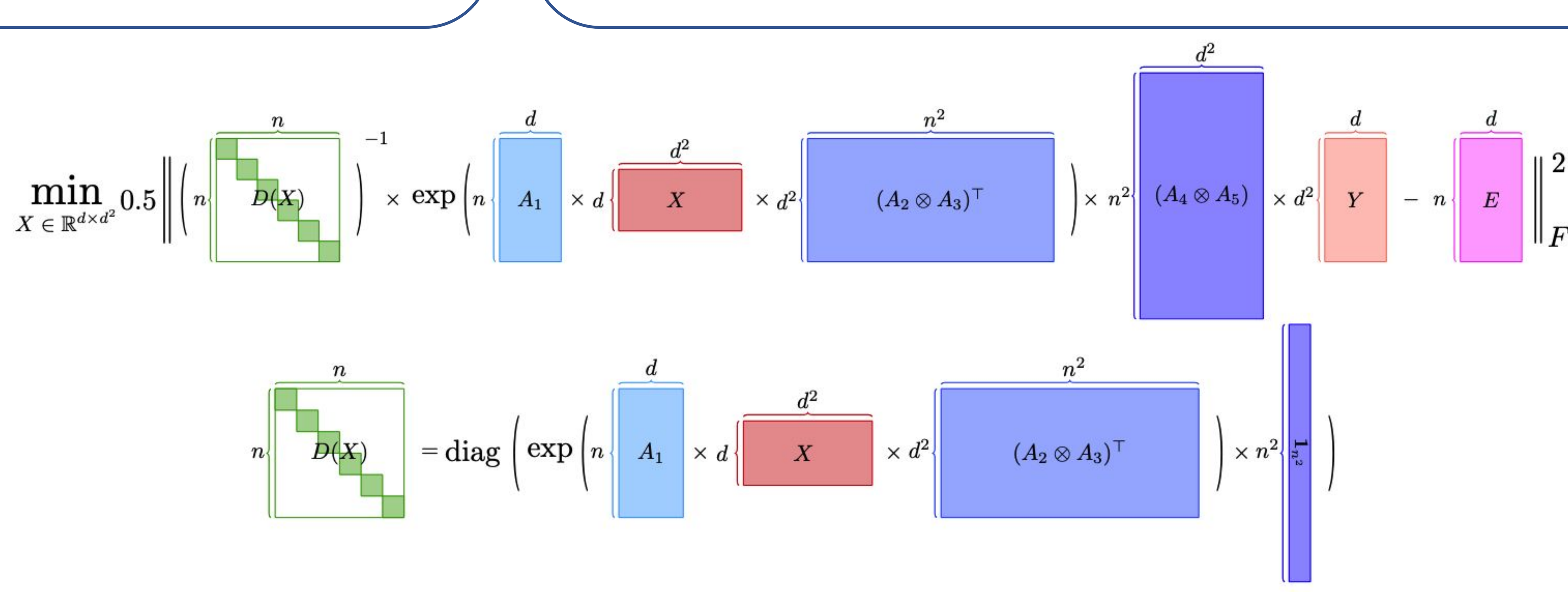
Suppose $A_1, A_2, A_3, A_4, A_5, E \in \mathbb{R}^{n \times d}$ and $Y_1, Y_2 \in \mathbb{R}^{d \times d}$ are given. Let $D(X) = \mathrm{diag}(\exp(A_1 X (A_2 \otimes A_3)^\top / d)\mathbf{1}_{n^2}) \in \mathbb{R}^{n \times n}$ and $Y = Y_1 \oslash Y_2 \in \mathbb{R}^{d^2 \times d}$. We formulate the attention optimization problem as:

$$\min_{X \in \mathbb{R}^{d \times d^2}} \mathsf{Loss}(X) := 0.5 \| D(X)^{-1} \exp(A_1 X (A_2 \otimes A_3)^\top / d)(A_4 \otimes A_5) Y - E \|_F^2.$$

Definition 2 (Approximate Tensor Attention Loss Gradient Computation ($\mathsf{ATAttLGC}(n, d, B, \epsilon)$))

Suppose $A_1, A_2, A_3, A_4, A_5, E \in \mathbb{R}^{n \times d}$ and $X_1, X_2, X_3, Y_1, Y_2 \in \mathbb{R}^{d \times d}$. Let $X = X_1 \cdot (X_2 \oslash X_3)^\top \in \mathbb{R}^{d \times d^2}$. Let $\epsilon, B > 0$. Assume that $\max\{\|A_1 X_1\|_\infty, \|A_2 X_2\|_\infty, \|A_3 X_3\|_\infty, \|A_4 Y_1\|_\infty, \|A_5 Y_2\|_\infty\} \leq B$. Let us assume that any numbers in the previous matrices are in the $\log(n)$ bits model. Then, our target is to output a matrix $\widetilde{g} \in \mathbb{R}^{d \times d^2}$ to approximate the gradient of the loss function in **Definition 1**, satisfying

$$\|\widetilde{g} - \frac{\mathrm{d}\mathsf{Loss}(X)}{\mathrm{d}X}\|_\infty \leq \epsilon.$$



Main Results

Theorem 1 (Fast gradient computation)

Assume that any numbers in the matrices are in the $\log(n)$ bits model. Then, there exist an algorithm that runs in almost linear time $n^{1+o(1)}$ to solve

$$\mathsf{ATAttLGC}(n, d = O(\log n), B = o(\sqrt[3]{\log n}), \epsilon = 1/\mathrm{poly}(n)).$$

Theorem 2 (Hardness)

Assume Strong Exponential Time Hypothesis (**SETH**). Let $\gamma : \mathbb{N} \to \mathbb{N}$ be any function with $\gamma(n) = o(\log n)$ and $\gamma(n) = \omega(1)$. For any constant $\delta > 0$, when $E = 0$, $Y = I_d$, $X = \lambda I_d$ for some scalar $\lambda \in [0, 1]$, it is impossible in $O(n^{3-\delta})$ time to solve

$$\mathsf{ATAttLGC}(n, d = \Theta(\log n), B = \Theta(\sqrt[3]{\gamma(n) \cdot \log n}), \epsilon = O(1/(\log n)^4)).$$