Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs

Linjian Ma, Edgar Solomonik
Department of Computer Science, University of Illinois at Urbana-Champaign
{lma16, solomon2}@illinois.edu

Abstract

This work discusses tensor network embeddings, which are random matrices (S) with tensor network structure. These embeddings have been used to perform dimensionality reduction of tensor network structured inputs x and accelerate applications such as tensor decomposition and kernel regression. Existing works have designed embeddings for inputs x with *specific* structures, such as the Kronecker product or Khatri-Rao product, such that the computational cost for calculating Sx is efficient. We provide a systematic way to design tensor network embeddings consisting of Gaussian random tensors, such that for inputs with *more general* tensor network structures, both the sketch size (row size of S) and the sketching computational cost are low.

We analyze general tensor network embeddings that can be reduced to a sequence of sketching matrices. We provide a sufficient condition to quantify the accuracy of such embeddings and derive sketching asymptotic cost lower bounds using embeddings that satisfy this condition and have a sketch size lower than any input dimension. We then provide an algorithm to efficiently sketch input data using such embeddings. The sketch size of the embedding used in the algorithm has a linear dependence on the number of sketching dimensions of the input. Assuming tensor contractions are performed with classical dense matrix multiplication algorithms, this algorithm achieves asymptotic cost within a factor of $O(\sqrt{m})$ of our cost lower bound, where m is the sketch size. Further, when each tensor in the input has a dimension that needs to be sketched, this algorithm yields the optimal sketching asymptotic cost. We apply our sketching analysis to inexact tensor decomposition optimization algorithms. We provide a sketching algorithm for CP decomposition that is asymptotically faster than existing work in multiple regimes, and show the optimality of an existing algorithm for tensor train rounding.

1 Introduction

Sketching techniques, which randomly project high-dimensional data onto lower dimensional spaces while still preserving relevant information in the data [43], have been widely used in numerical linear algebra, including for regression, low-rank approximation, and matrix multiplication [50]. One key step of sketching algorithms is to design an embedding matrix $S \in \mathbb{R}^{m\times n}$ with $m \ll n$, such that for any input (also called data throughout the paper) $x \in \mathbb{R}^n$, the projected vector norm is $(1 \pm \epsilon)$-close to the input vector, $\|Sx\|_2 = (1 \pm \epsilon)\|x\|_2$, with probability at least $1 - \delta$ (defined as (ϵ, δ)-accurate embedding throughout the paper), and the multiplication Sx can be computationally efficient. S is commonly chosen as a random matrix with each element being an i.i.d. Gaussian variable when x is dense, or a random sparse matrix when x is sparse, etc.

In this work, we focus on the case where x has a tensor network structure. A tensor network [35] uses a set of (small) tensors, where some or all of their dimensions are contracted according to some

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

pattern, to implicitly represent a tensor. Tensor network structured data is commonly seen in multiple applications, including kernel based statistical learning [38, 1, 51, 31], machine learning and data mining via tensor decomposition methods [3, 46, 22, 46], and simulation of quantum systems [49, 27, 44, 15, 16]. Commonly used embedding matrices are sub-optimal for sketching many such data. For example, consider the case where $x \in \mathbb{R}^{s^N}$ is a chain of Kronecker products, $x = x_1 \otimes \cdots \otimes x_N$, where $x_i \in \mathbb{R}^s$ for $i \in \{1, \ldots, N\}$. If $S \in \mathbb{R}^{m \times s^N}$ is a Gaussian matrix, the multiplication Sx has a computational cost of $\Omega(ms^N)$, and the exponential dependence on the tensor order N makes the calculation impractical when N or s is large.

The computational cost of the multiplication Sx can be reduced when S has a structure that can be easily multiplied with the target data. One example is when S has a Kronecker product structure, $S = S_1 \otimes \cdots \otimes S_N$ and each $S_i \in \mathbb{R}^{m^{1/N} \times s}$. When $x = x_1 \otimes \cdots \otimes x_N$, Sx can then be calculated efficiently via $(S_1 x_1) \otimes \cdots \otimes (S_N x_N)$, reducing the cost to $O(Nm^{1/N}s + m)$. Another example is when S has a Khatri-Rao product structure, $S = (S_1 \odot \cdots \odot S_N)^T$ and each $S_i \in \mathbb{R}^{s \times m}$. Sx can then be calculated efficiently via $(S_1^T x_1) * \cdots * (S_N^T x_N)$, where $*$ denotes the Hadamard product, which reduces the cost to $O(Nms)$. However, tensor-network-structured embedding matrices that can be easily multiplied with data may not necessarily minimize computational cost, since the sketch size sufficient for accurate embedding can also increase. For example, the sketch size necessary for both Kronecker product and Khatri-Rao product embeddings to be (ϵ, δ)-accurate is at least exponential in N, which is inefficient for large tensor order N [1]. To find embeddings that are both accurate and computationally efficient, it is therefore of interest to investigate tensor network structures that can both yield small sketch size and be multiplied with data efficiently.

Existing works discuss tensor network embeddings with more efficient sketch size than Kronecker and Khatri-Rao product structure, such as tensor train [41] and balanced binary tree [1]. In particular, Ahle et al. [1] designed a balanced binary tree structured embedding and showed that the sketch size sufficient for (ϵ, δ)-accurate embedding can have only linear dependence on N. Using this embedding to sketch Kronecker product structured data yields a sketching cost that only has a polynomial dependence on both N and s. However, for data with other tensor network structures, these embeddings may not be the most computationally efficient.

Our contributions In this work, we design algorithms to efficiently sketch more general data tensor networks such that each dimension to be sketched has size lower bounded by the sketch size and is a dimension of only one tensor. One of such data tensor networks is shown in Fig. 1. In particular, we look at the following question.

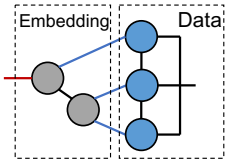


Figure 1: Illustration of target data and embedding. Blue edges have larger weights than the red edge.

For arbitrary data with a tensor network structure of interest, can we automatically sketch the data into one tensor with Gaussian tensor network embeddings that are accurate, have low sketch size, and also minimize the sketching asymptotic computational cost?

Different from existing works [1, 17, 26, 29] that construct the embedding based on fast sketching techniques, including Countsketch [8], Tensorsketch [37], and fast Johnson-Lindenstraus (JL) transform using fast Fourier transform [2], we discuss the case where each tensor in the embedding contains i.i.d. Gaussian random elements. Gaussian-based embeddings yield larger computational cost, but have the most efficient sketch size for both unconstrained and constrained optimization problems [9, 40]. This choice also enables us use a simple computational model to analyze the sketching cost, where tensor contractions are performed with classical dense matrix multiplication algorithms. Note that a related work [29] builds tree tensor network embeddings based on Countsketch for getting low-rank tensor network approximation of an input tensor.

While we allow for the data tensor network to be a hypergraph, we consider only graph embeddings, (detailed definition in Section 2), which include tree embeddings that have been previously studied [1, 10, 4]. Each one of these embeddings consisting of N_E tensors can be reduced to a sequence of N_E sketches (random sketching matrices). In Section 3, we show that if each of these sketches is $(\epsilon/\sqrt{N_E}, \delta)$-accurate, then the embedding is at least (ϵ, δ)-accurate.

In Section 4, we provide an algorithm to sketch input data with an embedding that not only satisfies the (ϵ, δ)-accurate sufficient condition, but is computationally efficient and has low sketch size.

Given a data tensor network and one data contraction tree T_0, this algorithm outputs a sketching contraction tree that is constrained on T_0. This setting is useful for application of sketching to alternating optimization in tensor-related problems, such as tensor decompositions. In alternating optimization, multiple contraction trees of the data x are chosen in an alternating order to form multiple optimization subproblems, each updating part of the variables [39, 25, 28]. Designing embeddings under the constraint can help reuse contracted intermediates across subproblems.

The sketch size of the embedding used in the algorithm has a linear dependence on the number of sketching dimensions of the input. As to the sketching asymptotic computational cost, within all constrained sketching contraction trees with embeddings satisfying the (ϵ, δ)-accurate sufficient condition and only have one output sketch dimension, this algorithm achieves asymptotic cost within a factor of $O(\sqrt{m})$ of the lower bound, where m is the sketch size. When the input data tensor network structure is a graph, the factor improves to $O(m^{0.375})$. In addition, when each tensor in the input data has a dimension to be sketched, such as Kronecker product input and tensor train input, this algorithm yields the optimal sketching asymptotic cost.

At the end of Section 4, we look at cases where the widely discussed tree tensor network embeddings are efficient in terms of the sketching computational cost. We show for input data graphs such that each data tensor has a dimension to be sketched and each contraction in the given data contraction tree T_0 contracts dimensions with size being at least the sketch size, sketching with tree embeddings can achieve the optimal asymptotic cost.

In Section 5, we apply our sketching algorithm to two applications, CANDECOMP/PARAFAC (CP) tensor decomposition [14, 13] and tensor train rounding [36]. We present a new sketching-based alternating least squares (ALS) algorithm for CP decomposition. Compared to existing sketching-based ALS algorithm, this algorithm yields better asymptotic computational cost under several regimes, such as when the CP rank is much lower than each dimension size (the length/number of elements in each dimension) of the input tensor. We also provide analysis on the recently introduced randomized tensor train rounding algorithm [10]. We show that the tensor train embedding used in that algorithm satisfies the accuracy sufficient condition in Section 3 and yields the optimal sketching asymptotic cost, implying that this is an efficient algorithm, and embeddings with other structures cannot achieve lower asymptotic cost.

2 Definitions

We introduce some tensor network notation here, and provide additional definitions and background in Appendix A. The structure of a tensor network can be described by an undirected hypergraph $G = (V, E, w)$, also called tensor diagram. Each hyperedge $e \in E$ may be adjacent to either one or at least two vertices, and we refer to hyperedges with a dangling end (one end not adjacent to any vertex) as uncontracted hyperedges, and those without dangling end as contracted hyperedges. We refer to the cardinality of a hyperedge as its number of ends. An example is shown in Fig. 2. We use w to denote a function such that for each $e \in E$, $w(e) = \log(s)$ is the natural logarithm of the dimension sizes represented by hyperedge e. For a hyperedge set E, we use $w(E) = \sum_{e \in E} w(e)$ to denote the weighted sum of the hyperedge set.

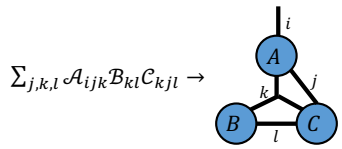


Figure 2: An example of tensor diagram notation.

A tensor network embedding is the matricization of a tensor described by a tensor network, and each embedding can be described by $S = (G_E, \bar{E})$, where $G_E = (V_E, E_E, w)$ shows the embedding graph structure and $\bar{E} \subseteq E_E$ is the edge set connecting data and the embedding. In this work we only discuss the case where G_E is a graph, such that each uncontracted edge in E_E is adjacent to one vertex and contracted edge in E_E is adjacent to two vertices. Let $E_1 \subset E_E$ be the subset of uncontracted edges, S is a matricization such that uncontracted dimensions in $\bar{E}$ are grouped into the column of the matrix, and dimensions in E_1 are grouped into the row. We use $N = |\bar{E}|$ to denote the order of the embedding, and $m = \exp(w(E_1))$ denotes the output sketch size. We use $G_D = (V_D, E_D, w)$ to represent the data tensor network structure, and use $G = (G_D, G_E)$ to denote the overall tensor network structure.

Within the tensor network $G = (V, E, w)$, the contraction between two tensors represented by $v_i, v_j \in V$ is denoted by (v_i, v_j). The contraction between two tensors that are the contraction

outputs of $W_i \subset V$, $W_j \subset V$, respectively, is denoted by (W_i, W_j). A contraction tree on the tensor network $G = (V, E, w)$ is a rooted binary tree $T_B = (V_B, E_B)$ showing how the tensor network is fully contracted. Each vertex in V_B can be represented by a subset of the vertices, $W \subseteq V$, and denotes the contraction output of W. The two children of W, denoted as W_1 and W_2, must satisfy $W_1 \cup W_2 = W$. Each leaf vertex must have $|W| = 1$, and the root vertex is represented by V. Any topological sort of the contraction tree represents a contraction path (order) of the tensor network.

3 Sufficient condition for accurate embedding

We consider the scenario where the data tensor networks have a general hypergraph structure, while the embeddings have a graph structure, thus some embeddings, such as those with a Khatri-Rao product structure [41, 9], are not considered in this work. Such embeddings can be linearized to a sequence of sketches. Let $N_E = |V_E|$ denote the number of vertices in the embedding, in each linearization, each vertex is given an unique index $i \in [N_E]$[1] and denoted v_i. The ith tensor is denoted by $\mathcal{A}_i$, and A_i denotes its matricization where we combine all uncontracted dimensions and contracted dimensions connected to $\mathcal{A}_j$ with $j > i$ into the row, and other dimensions into the column. The embedding can then be represented as a chain of multiplications, $S = M_{N_E} P_{N_E} \cdots M_1 P_1$, where M_i is the Kronecker product of identity matrices with A_i for $i \in [N_E]$, and P_i is a permutation matrix. We illustrate the linearization in Fig. 3 using a fully connected tensor network embedding. We show in Theorem 3.1 a sufficient condition for embeddings to be (ϵ, δ)-accurate.

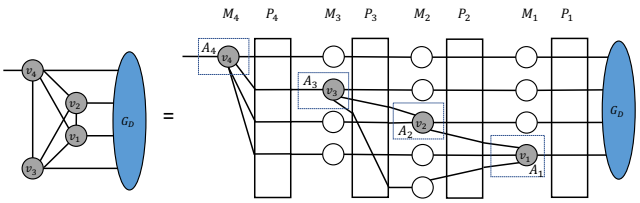


Figure 3: Illustration of embedding linearization. Each gray vertex denotes a tensor of the embedding, each white vertex denotes an identity matrix, and each white box denotes a permutation matrix.

Theorem 3.1 ((ϵ, δ)-accurate sufficient condition)**.** *Consider a Gaussian tensor network embedding where there exists a linearization such that each A_i for $i \in [N_E]$ has row size $\Omega(N_E \log(1/\delta)/\epsilon^2)$. Then the tensor network embedding is (ϵ, δ)-accurate.*

Proof. Based on the composition rules of JL moment [18, 19] in Lemma A.2 and Lemma A.3 in the appendix, in the linearization all $M_i P_i$ satisfy the strong $\left(\frac{\epsilon}{L\sqrt{2N}}, \delta\right)$-JL moment property so S satisfies the strong (ϵ, δ)-JL moment property. This implies the embedding is (ϵ, δ)-accurate. □

Theorem 3.1 is a sufficient (but not necessary) condition for constructing (ϵ, δ)-accurate embedding. It also implies that specific tree embeddings are (ϵ, δ)-accurate, as we show below.

Corollary 3.2. *Consider a Gaussian embedding containing a tree tensor network structure, where there is only one output sketch dimension with size $m = \Theta\big(N_E \log(1/\delta)/\epsilon^2\big)$, and each dimension within the embedding has size m. Then the embedding is (ϵ, δ)-accurate.*

Proof. Consider the linearization such that vertices are labelled based on the reversed ordering of a breath-first search from the vertex adjacent to the edge associated with the output sketch dimension. Each A_i has row size $m = \Theta\big(N_E \log(1/\delta)/\epsilon^2\big)$ thus the embedding satisfies Theorem 3.1. □

One special case of Corollary 3.2 is the tensor train [36] (also called matrix product states (MPS) [45]) embedding, where the embedding tensor network has a 1D structure along with an output dimension adjacent to one of the endpoint tensors. Tensor train is widely used to efficiently represent high dimensional tensors in multiple applications, including numerical PDEs [12, 42], quantum physics [44], high-dimensional data analysis [20, 21] and machine learning [6, 47, 33]. Since the tensor train embedding contains N vertices, Corollary 3.2 directly implies that a sketch size of $m = \Theta\big(N \log(1/\delta)/\epsilon^2\big)$ is sufficient for the MPS embedding to be (ϵ, δ)-accurate. This embedding has already been used in applications including tensor train rounding [10] and low rank approximation of matrix product operators [4].

[1] Throughout the paper we use $[N]$ to denote $\{1, \ldots, N\}$.

Note that the tensor train embedding introduced in this work and [10] adds an output sketch dimension to the standard tensor train, and restricts the tensor train rank to be the sketch size m. This is different from the recent work by Rakhshan and Rabusseau [41], where they construct an embedding consisting of m independent tensor trains, each one with a tensor train rank of R. A sketch size upper bound of $m = \Theta\Big(1/\epsilon^2 \cdot (1+2/R)^N \log^{2N}(1/\delta)\Big)$ is derived for that embedding to be (ϵ, δ)-accurate. However, this bound has an exponential dependence on N.

4 A sketching algorithm with efficient computational cost and sketch size

We find Gaussian tensor network embeddings G_E that both have efficient sketch size and yield efficient computational cost. We are given a specific data tensor network G_D that implicitly represents a matrix $M \in \mathbb{R}^{s_1 s_2 \cdots s_N \times t}$, and want to sketch the row dimension of the matrix. We assume that size of each dimension to be sketched, s_i for $i \in [N]$, is greater than the sketch size m, and each one of these dimensions is adjacent to only one tensor. The goal is to find a Gaussian embedding G_E satisfying the following properties.

- $G_E \in \mathcal{G}^{(\epsilon,\delta)}$, where $\mathcal{G}^{(\epsilon,\delta)}$ contains all embeddings not only satisfying the (ϵ, δ)-accurate sufficient condition in Theorem 3.1, but also only have one output sketch dimension ($|E_1| = 1$) with size $m = \Theta\big(N_E \log(1/\delta)/\epsilon^2\big)$. This guarantees that the embedding is accurate and the output sketch size is linear w.r.t. the number of vertices in G_E. Note that although the data can be a hypergraph, the embeddings considered in $\mathcal{G}^{(\epsilon,\delta)}$ are defined on graphs.
- To fully contract the tensor network system (G_D, G_E), this embedding yields a contraction tree with the *optimal asymptotic contraction cost* under a fixed data contraction tree. The data contraction tree constraint is useful for application of sketching to alternating optimization algorithms, as we will discuss in Section 5. This can be written as an optimization problem below,

$$\min_{G_E} \min_{T_B} C_a(T_B(G_D, G_E)), \quad \text{s.t. } G_E \in \mathcal{G}^{(\epsilon,\delta)}, \quad T_0(G_D) \subset T_B(G_D, G_E), \tag{4.1}$$

  where $T_B(G_D, G_E)$ denotes a contraction tree of the tensor network (G_D, G_E), C_a denotes the asymptotic computational cost, and $T_0(G_D) \subset T_B(G_D, G_E)$ means the contraction tree T_B is constrained on T_0 (the detailed definition and a simple example are shown in Definition 1 and Appendix B.1, respectively).

Definition 1 (Constrained contraction tree)**.** Given $G = (G_E, G_D)$ and a contraction tree T_0 of G_D, the contraction tree T_B for G is constrained on T_0 if for each contraction $(A, B) \in T_0$, there must exist one contraction $(\hat{A}, \hat{B}) \in T_B$, such that $\hat{A} \cap V_D = A$ and $\hat{B} \cap V_D = B$.

Algorithm We propose an algorithm to sketch tensor network data with an embedding containing two parts, a Kronecker product embedding and an embedding containing a binary tree of small tensor networks. The embedding is illustrated in Fig. 4. The Kronecker product embedding consists of N Gaussian random matrices and is used to reduce the weight of each edge in $\bar{E}$, the set of edges to be sketched. The binary tree structured embedding consists of $N-1$ *small tensor networks*, each represented by one binary tree vertex. Each small tensor network is used to effectively sketch the contraction of pairs of tensors adjacent to edges in $\bar{E}$. The embedding with a binary tree structure may not be a binary tree tensor network, since each tree vertex is not restricted to represent one tensor. The binary tree is chosen to be consistent with the dimension tree of the data contraction tree T_0, which is a directed binary tree showing the way edges in $\bar{E}$ are merged onto the same tensor in T_0. The detailed definition of dimension tree is in Appendix B.1.

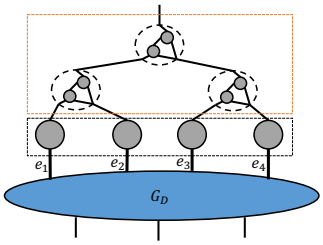


Figure 4: Illustration of the embedding. The black box includes the Kronecker product embedding and the orange box includes the embedding containing a binary tree of small tensor networks.

We first introduce some notation before presenting the algorithm. Consider a given input data tensor network $G_D = (V_D, E_D, w)$ and its given data contraction tree, T_0. Below we let $\bar{E} = \{e_1, e_2, \ldots, e_N\}$ to denote the edges to be sketched. Let $N_D = |V_D|$. Based on the definition we have $N \leq N_D$ and T_0 contains $N_D - 1$ contractions. Let one contraction path

of T_0, which is a topological sort of the contractions in T_0, be expressed as

$$\{(U_1, V_1), \ldots, (U_{N_D-1}, V_{N_D-1})\}, \tag{4.2}$$

where (U_i, V_i) represents the contraction of two intermediate tensors represented by two subset of vertices $U_i, V_i \subset V_D$. The $N_D - 1$ contractions can be categorized into $N + 2$ sets, $\mathcal{D}(e_1), \ldots, \mathcal{D}(e_N), \mathcal{S}, \mathcal{I}$, as follows, and these sets are illustrated with an example in Appendix C.

- Consider contractions (U_i, V_i) such that both U_i and V_i are adjacent to edges in $\bar{E}$. $\mathcal{S}$ contains all contractions with this property.
- Consider contractions (U_i, V_i) such that the only edge in $\bar{E}$ that is adjacent to the contraction output is e_j, $\bar{E}(U_i \cup V_i) = \{e_j\}$. We let $\mathcal{D}(e_j)$ contains contractions with this property. When $\mathcal{D}(e_j)$ is not empty, we let $X(e_j) \subset V$ represent the sub network contracted by $\mathcal{D}(e_j)$. When $\mathcal{D}(e_j)$ is empty, we let $X(e_j) = v_j$, where v_j is the vertex in the data graph adjacent to e_j.
- The remaining contractions in the contraction tree include (U_i, V_i) such that both U_i and V_i are not adjacent to $\bar{E}$, and contractions where U_i or V_i is adjacent to at least two edges in $\bar{E}$, and the other one is not adjacent to any edge in $\bar{E}$. We let $\mathcal{I}$ contain these contractions.

The sketching algorithm is shown in Algorithm 1, and the details are as follows,

Algorithm 1 Sketching algorithm

Input: Input data tensor network G_D, data contraction tree T_0 expressed in (4.2)
for each $e_i \in \bar{E}$ **do**
  // Sketch with Kronecker product embedding
  $W \leftarrow$ contract and sketch $X(e_j)$
  Replace the contraction output of $X(e_j)$ by W in T_0
end for
for each contraction (U_i, V_i) in $\mathcal{S} \cup \mathcal{I}$ **do**
  if $i \in \mathcal{S}$ **then**
    // Sketch with binary tree embedding
    $W_i \leftarrow$ contract and sketch (U_i, V_i) (detailed in Appendix C.1)
  else
    $W_i \leftarrow$ contract(U_i, V_i)
  end if
  Replace the contraction output of (U_i, V_i) by W_i in T_0
end for
return W_{N_D-1}

- One matrix in the Kronecker product embedding is used to sketch the sub data network $X(e_j)$, which guarantees that two sketch dimensions to be merged onto one tensor will both have size $\Theta(N \log(1/\delta)/\epsilon^2)$. For the case where $\mathcal{D}(e_j) = \emptyset$, we directly sketch $X(e_j) = v_j$ using an embedding matrix. For the case where $\mathcal{D}(e_j) \neq \emptyset$, we select $k(e_j) \in \mathcal{D}(e_j)$ and apply the sketching matrix during the contraction $(U_{k(e_j)}, V_{k(e_j)})$. The value of $k(e_j)$ is selected via an exhaustive search over all $|\mathcal{D}(e_j)|$ contractions, so that sketching $X(e_j)$ has the lowest asymptotic cost.
- One small tensor network (denoted as Z_i) represented by a binary tree vertex in the binary tree structured embedding is used to sketch the contraction (U_i, V_i) when $i \in \mathcal{S}$, which means that both U_i and V_i are adjacent to $\bar{E}$. Let $\hat{U}_i, \hat{V}_i$ denote the sketched U_i and V_i formed in previous contractions in the sketching contraction tree T_B, such that $\hat{U}_i \cap V_D = U_i$ and $\hat{V}_i \cap V_D = V_i$, the structure of Z_i is determined so that the asymptotic cost to sketch $(\hat{U}_i, \hat{V}_i)$ is minimized under the constraint that Z_i is in $\mathcal{G}^{(\epsilon/\sqrt{N}, \delta)}$, so that it satisfies the $(\epsilon/\sqrt{N}, \delta)$-accurate sufficient condition and only has one output dimension. In Appendix C.1, we provide an algorithm to construct Z_i containing 2 tensors, so that the output sketch size of Z_i is $\Theta(N \log(1/\delta)/\epsilon^2)$.

The total computational cost of Algorithm 1 consists of three components: the cost of determining the embedding structure, the cost of determining the sketching contraction tree, and the cost of sketching. The first two components are $O(N)$ and are therefore negligible in comparison to the cost of sketching.

Analysis of the algorithm The embedding constructed during Algorithm 1 contains $\Theta(N)$ vertices, and the output sketch size is $m = \Theta(N \log(1/\delta)/\epsilon^2)$. Therefore, the sketching result both has low sketch size and is (ϵ, δ)-accurate. Below we discuss the optimality of Algorithm 1 in terms of the sketching asymptotic computational cost. We first discuss the case when each vertex in the data tensor network is adjacent to an edge in $\bar{E}$.

Theorem 4.1. *For data tensor networks where each vertex is adjacent to an edge in $\bar{E}$, the asymptotic cost of Algorithm 1 is optimal w.r.t. the optimization problem in (4.1).*

We show the detailed proof of the theorem above in Appendix D.1. Therefore, Algorithm 1 is efficient in sketching multiple widely used tensor network data, including tensor train, Kronecker product, and Khatri-Rao product. As we will discuss in Section 5, Algorithm 1 can be used to design efficient sketching-based ALS algorithm for CP tensor decomposition.

Note that the embedding in Algorithm 1 may not be a tree embedding. As we will show in Section 6, for cases including sketching a Kronecker product data, Algorithm 1 is more efficient than sketching with tree embeddings. On the other hand, for some data tensor networks, sketching with a tree embedding also yields the optimal asymptotic cost, which we will show in Theorem 4.3.

For general input data where each data vertex may not adjacent to an edge in $\bar{E}$, Algorithm 1 may not yield the optimal sketching asymptotic cost, but is within a factor of at most $O(\sqrt{m})$ from the cost lower bound. Below we show the theorem, and the detailed proof is in Appendix D.2.

Theorem 4.2. *For any data tensor network G_D, the asymptotic cost of Algorithm 1 (denoted as c) satisfy $c = O(\sqrt{m} \cdot c_{opt})$, where c_{opt} is the optimal asymptotic computational cost for the optimization problem (4.1) and $m = \Theta(N \log(1/\delta)/\epsilon^2)$. When G_D is a graph, $c = O\big(m^{0.375} \cdot c_{opt}\big)$.*

Efficiency of tree tensor network embedding We discuss cases where tree tensor network embeddings can be optimal w.r.t. the optimization problem in (4.1). Tree embeddings, in particular the tensor train embedding, have been widely discussed and used in prior work [41, 10, 4]. We design an algorithm to sketch with tree embeddings. The algorithm is similar to Algorithm 1, and the only difference is that for each contraction (U_i, V_i) with $i \in \mathcal{S}$, such that both U_i and V_i are adjacent to edges in $\bar{E}$, we sketch it with one tensor rather than a small network. Below, we present the optimality of the algorithm in terms of sketching asymptotic cost.

Theorem 4.3. *Consider G_D with each vertex adjacent to an edge to be sketched and its given contraction tree T_0. If each contraction in T_0 contracts dimensions with size being at least the sketch size, then sketching with tree embedding would yield the optimal asymptotic cost for (4.1).*

We present the proof of Theorem 4.3 in Appendix E. As we will show in Section 6, for tensor network data with relatively large contracted dimension sizes such that the condition in Theorem 4.3 is satisfied, sketching with tree embedding yields a similar performance as Algorithm 1. However, for data where the condition in Theorem 4.3 is not satisfied, Algorithm 1 is more efficient. For example, when the data is a vector with a Kronecker product structure, sketching with Algorithm 1 yields a cost of $\Theta(\sum_{j=1}^{N} s_j m + Nm^{2.5})$ and sketching with a tree embedding yields a cost of $\Theta(\sum_{j=1}^{N} s_j m + Nm^3)$. We present the detailed analysis in Appendix C.2 and Appendix E.

5 Applications

Alternating least squares for CP decomposition On top of Algorithm 1, we propose a new sketching-based ALS algorithm for CP tensor decomposition. Throughout analysis we assume the input tensor is dense, and has order N and size $s \times \cdots \times s$, and the CP rank is R. The goal of CP decomposition is to minimize the objective function, $f(A_1, \ldots, A_N) = \left\| \mathcal{X} - \sum_{r=1}^{R} A_1(:, r) \circ \cdots \circ A_N(:, r) \right\|_F^2$, where $A_i \in \mathbb{R}^{s \times R}$ for $i \in [N]$ are called factor matrices, and $\mathcal{X}$ denotes the input tensor. In each iteration of ALS, N subproblems are solved sequentially, and the ith subproblem can be formulated as $A_i = \arg\min_A \left\| L_i A^T - R_i \right\|_F^2$, where $L_i = A_1 \odot \cdots \odot A_{i-1} \odot A_{i+1} \odot \cdots \odot A_N$ consists of a chain of Khatri-Rao products, and $R_i = X_{(i)}^T$ is the transpose of ith matricization of $\mathcal{X}$.

Multiple sketching-based randomized algorithms are proposed to accelerate each subproblem in CP-ALS [5, 24, 30]. The sketched problem can be formulated as $A_i = \underset{A}{\operatorname{argmin}} \left\| S_i L_i A^T - S_i R_i \right\|_F^2$, where S_i is an embedding. The goal is to design S_i such that the sketched subproblem can be solved efficiently and accurately. In Table 1, we summarize two state-of-the-art sketching methods for CP-ALS. Larsen and Kolda [24] propose a method that sketches the subproblem based on (approximate) leverage score sampling (LSS), but both the per-iteration computational cost and the sketch size sufficient for (ϵ, δ)-accurate solution has an exponential dependence on N, which is inefficient for decomposing high order tensors. Malik [30] proposes a method called recursive leverage score

sampling for CP-ALS, where the embedding contains two parts, $S_i = S_{i,1}S_{i,2}$, and $S_{i,2}$ is an embedding with a binary tree structure proposed in [1] with sketch size $\Theta(NR^2/\delta)$, and $S_{i,1}$ performs approximate leverage score sampling on $S_{i,2}L_i$ with sketch size $\tilde{\Theta}(NR/\epsilon^2)$. This sketching method has a better dependence on R in terms of per-iteration cost. For both algorithms, the preparation cost shown in Table 1 denotes the cost to go over all elements in the tensor and initialize factor matrices using randomized range finder. As is shown in [24, 26], randomized range finder based initialization is critical for achieving accurate CP decomposition with sampling-based sketched ALS.

CP-ALS algorithm	Per-iteration cost	Sketch size (m)	Prep cost
Standard ALS	$\Theta(s^N R)$	/	/
LSS [24]	$\tilde{\Theta}(N(R^{N+1} + sR^N)/\epsilon^2)$	$\tilde{\Theta}(R^{N-1}/\epsilon^2)$	$\Theta(s^N)$
Recursive LSS [30]	$\tilde{\Theta}(N^2(R^4 + NsR^3/\epsilon)/\delta)$	$\Theta(NR^2/\delta)$ and $\tilde{\Theta}(R/(\epsilon\delta))$	$\Theta(s^N)$
Algorithm 1	$\tilde{\Theta}(N^2(N^{1.5}R^{3.5}/\epsilon^3 + sR^2)/\epsilon^2)$	$\tilde{\Theta}(NR/\epsilon^2)$	$\Theta(s^N m)$

Table 1: Comparison of asymptotic algorithmic complexity between standard CP-ALS, CP-ALS with leverage score sampling (LSS), CP-ALS with recursive leverage score sampling (recursive LSS), and sketching CP-ALS with Algorithm 1. The third column shows the sketch size sufficient for the sketched linear least squares to be $(1+\epsilon)$-accurate with probability at least $1-\delta$. By using $\tilde{\Theta}$, we neglect logarithmic factors, including $\log(R)$ and $\log(1/\delta)$.

We propose a new sketching algorithm for CP-ALS based on Algorithm 1. Each S_i is generated on top of the data tensor network L_i and its given data contraction tree T_i, with the sketch size being $m = \Theta(NR\log(1/\delta)/\epsilon^2) = \tilde{\Theta}(NR/\epsilon^2)$. The contraction trees T_i for $i \in [N]$ are chosen in a fixed alternating order, such that the resulting embeddings S_i for $i \in [N]$ have common parts and allow reusing contraction intermediates. We leave the detailed analysis in Appendix F.2.

The ALS per-iteration cost is $\Theta(N(m^{2.5}R + smR)) = \tilde{\Theta}(N^2(N^{1.5}R^{3.5}/\epsilon^3 + sR^2)/\epsilon^2)$. We present the detailed sketching algorithm and its cost analysis in Appendix F.3. When performing a low-rank CP decomposition with $s \gg R^{1.5}$ and ϵ is not too small so that $\epsilon = \Theta(1)$[2], the per-iteration cost is dominated by the term $\tilde{\Theta}(N^2sR^2/\epsilon^2)$, which is $\Theta(NR\epsilon/\delta) = \Omega(NR)$ times better than the per-iteration cost of the recursive LSS algorithm. For another case of a high-rank CP decomposition with $R \gg s$, which happens when one wants a high-accuracy CP decomposition of high order tensors, the per-iteration cost of our sketched CP-ALS algorithm is dominated by the term $\tilde{\Theta}(N^{3.5}R^{3.5}/\epsilon^5)$, and the cost ratio between this algorithm and the recursive LSS algorithm is $\tilde{\Theta}(N^{1.5}\delta/(\epsilon^5 R^{0.5}))$. For this case, our algorithm is only preferable when $N^{1.5}\delta/\epsilon^5$ is not too large compared to $R^{0.5}$.

Although our proposed sketching algorithm yields better per-iteration asymptotic cost in multiple regimes compared to existing leverage score based sketching algorithms, some preparation computations are needed to sketch right-hand-sides S_iR_i for $i \in [N]$ before ALS iterations, and this cost is non-negligible. On the other hand, this algorithm has better parallelism, since it involves a sequence of matrix multiplications rather than sampling the matrix. We leave the detailed experimental comparison of computational efficiency of different sketching techniques for future work.

Tensor train rounding Given a tensor train, *tensor train rounding* finds a tensor train with a lower rank to approximate the original representation. Throughout analysis we assume the tensor train has order N with the output dimension sizes equal s, the tensor train rank is $R < s$, and the goal is to round the rank to $r < R$. The standard tensor train rounding algorithm [36] consists of a right-to-left sweep of QR decompositions of the input tensor train (also called orthogonalization), and another left-to-right truncated singular value decompositions (SVD) sweep to perform rank reduction. The orthogonalization step is the bottleneck of the rounding algorithm, and costs $\Theta(NsR^3)$. Recently, [10] has introduced a randomized rounding algorithm called "*Randomize-then-Orthogonalize*". Let X denote a matricization of the tensor train data with all except one dimension at the end grouped into the row, the algorithm first sketches X with a tensor train embedding S, then performs a sequence of truncated SVDs on top of SX. The sketch size m of S is r plus some constant, and is assumed to be smaller than R. The bottleneck is to compute SX, which costs $\Theta(NsR^2m)$.

We can recast the problem as finding an embedding satisfying the linearization sufficient condition with sketch size m, such that the asymptotic cost of computing SX is optimal given the data

[2] As is shown in [26], in practice, setting ϵ to be 0.1-0.2 will result in accurate sketched least squares with relative residual norm error less than 0.05.

contraction tree that contracts the tensor train from one end to another. Our analysis (detailed in Appendix G) shows that the sketching cost for the problem is lower bounded by $\Omega(NsR^2m)$, thus the sketching algorithm in [10] attains the asymptotic cost lower bound and is efficient. Note that sketching with Algorithm 1 yields the same asymptotic cost, despite using a different embedding.

6 Experiments

We conduct multiple experiments to demonstrate the efficacy of our proposed embeddings. Below we first justify the theoretical analysis in Theorem 4.1 and Theorem 4.3 via testing the sketching performance on tensor train inputs and Kronecker product inputs. We then perform experiments to demonstrate that the accuracy of our proposed sketching algorithms is comparable to that of state-of-the-art sketching techniques for CP decomposition and tensor train rounding. Our experiments are carried out on an Intel Core i7 2.9 GHz Quad-Core machine using NumPy [34] routines in Python.

Sketching tensor train and Kronecker product inputs We compare the performance of general tensor network embedding used in Algorithm 1 (called TN embedding), tree embedding discussed in Theorem 4.3, and the baseline, tensor train embedding [10], in sketching tensor train input data in Fig. 5. The input tensor train data has order 6, and the dimension size is 500. We test the sketching performance under different tensor train ranks. For a given rank, we randomly generate 25 different inputs, with each element in each tensor being an i.i.d. variable uniformly distributed within $[0, 1]$. Additional experiments with Gaussian-distributed input tensor train data are presented in Appendix H. For each input x and a specific embedding structure, we calculate the relative sketching error twice under different sketch sizes, and record the smallest sketch size such that both of its relative sketching errors are within 0.2, $\frac{\|Sx\|_2}{\|x\|_2} \leq 0.2$. We also calculate the number of floating point operations (FLOPs) for computing Sx under the smallest sketch size based on the classical dense matrix multiplication algorithm. As can be seen, tree and tensor train embeddings are as efficient as TN embedding in terms of number of FLOPs under relatively high tensor train rank (when rank is at least 32), but are less efficient than TN embedding when the tensor train rank is lower than 32. The results are consistent with the theoretical analysis in Theorem 4.3, which shows that tree embeddings yield the optimal asymptotic cost when the input tensor train rank is at least the output sketch size, but the asymptotic cost is not optimal when the tensor train rank is low.

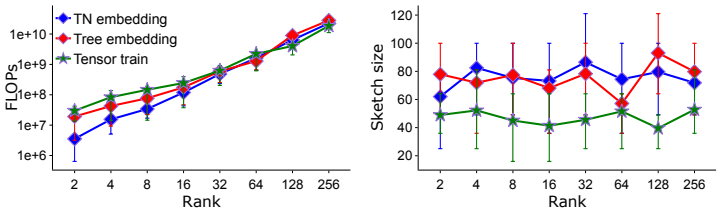


Figure 5: Results for sketching tensor train inputs. Each point denotes the mean value across 25 experiments, and each error bar shows the 25th-75th quartiles.

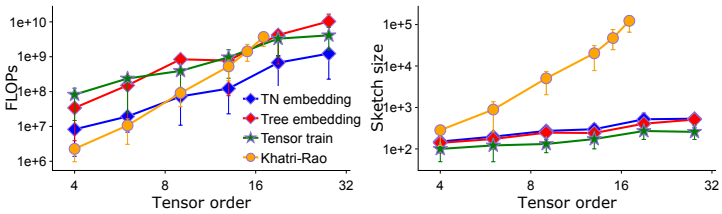


Figure 6: Results for sketching Kronecker product inputs.

We also compare the performance of TN, tree, and two baselines proposed in [41], tensor train and Khatri-Rao product embeddings, in sketching Kronecker product inputs in Fig. 6. Each dimension size of the Kronecker product input is fixed to be 1000, and we test the sketching performance under different tensor orders. For each input x and a specific embedding structure, we record the smallest sketch size such that its relative sketching error is within 0.1. As can be seen, compared to Khatri-Rao product embedding, the sketch size of TN, tree and tensor train embeddings all increase slowly with the increase of tensor order, consistent with the theoretical analysis that these embeddings have efficient sketch size. In addition, the cost in FLOPs of TN embedding is smaller than tree and tensor train embeddings. This is consistent with the analysis in Theorem 4.1 and its following discussions,

showing that TN embedding yields the optimal asymptotic cost for Kronecker product inputs, but tree and tensor train embeddings do not.

CP decomposition and tensor train rounding We perform experiments to demonstrate that the accuracy of our proposed sketching methods for CP-ALS and tensor train rounding is comparable to that of state-of-the-art sketching techniques. For both applications, we evaluate accuracy based on the final fitness f for each algorithm, defined as $f = 1 - \frac{\|\mathcal{T}-\widetilde{\mathcal{T}}\|_F}{\|\mathcal{T}\|_F}$, where $\mathcal{T}$ is the input tensor and $\widetilde{\mathcal{T}}$ is the reconstructed low-rank tensor.

For CP-ALS, we conduct experiments on a Time-Lapse hyperspectral radiance image [32], which is a 3-D tensor with dimensions of $1024 \times 1344 \times 33$. This input data is used to demonstrate the applicability of our method in real-world scenarios. Standard CP-ALS, sketched CP-ALS using Algorithm 1, and sketched CP-ALS with approximate leverage score sampling (LSS) [24] are compared. The output CP decomposition fitness under varying CP ranks and sketch sizes is shown in Table 2. As can be seen, sketching with Algorithm 1 yields comparable fitness with the algorithm that sketches with approximate leverage score sampling. Note that the computational cost of Algorithm 1 is lower that LSS especially when the CP rank is low and the tensor dimension is large, as stated in Table 1.

CP rank	2	5	10
Sketch size	25	64	100
CP-ALS	0.737	0.804	0.838
LSS [24]	0.739	0.773	0.789
Algorithm 1	0.737	0.770	0.801

Table 2: Comparison of the final fitness of different CP decomposition algorithms under different CP ranks and sketch sizes. 10 ALS iterations are performed for all algorithms before the final fitness are calculated.

TT rounding rank	1	4	11	20
Sketch size	4	9	16	25
TT-SVD [36]	0.734	0.862	0.944	0.981
TT embedding [10]	0.573	0.757	0.882	0.951
Algorithm 1	0.527	0.761	0.866	0.948

Table 3: Comparison of the final fitness of different tensor train rounding algorithms under different tensor train rounding ranks and sketch sizes.

We use 9 images from the Time-Lapse hyperspectral radiance image dataset for tensor train rounding, and reshape the input data to an order 6 tensor with size $9 \times 32 \times 32 \times 28 \times 48 \times 33$. We use the TensorLy [23] library to truncate the input tensor to a tensor train with a rank of 30. On top of this tensor train, we evaluate the accuracy of various approaches, including tensor train SVD [36], randomized algorithm using tensor train embedding [10], and randomized algorithm using Algorithm 1. The fitness of the truncated tensor trains are displayed in Table 3 for a variety of rounding rank thresholds and sketch sizes. As can be seen, sketching with Algorithm 1 has comparable accuracy with the baseline algorithm (sketching with tensor train embedding). In addition, both sketching algorithms also have similar complexity as is analyzed in Section 5.

7 Conclusions

We provide detailed analysis of general tensor network embeddings. For input data such that each dimension to be sketched has size greater than the sketch size, we provide an algorithm to efficiently sketch such data using Gaussian embeddings that can be linearized into a sequence of sketching matrices and have low sketch size. Our sketching method is then used to design state-of-the-art sketching algorithms for CP tensor decomposition and tensor train rounding. We leave the analysis for more general embeddings for future work, including those with each tensor representing fast sketching techniques, such as Countsketch and fast JL transform using fast Fourier transform, and those containing structures cannot be linearized, such as Khatri-Rao product embedding. It would also be of interest to look at other tensor-related applications that could benefit from tensor network embedding, including tensor ring decomposition and simulation of quantum circuits. We also leave the high-performance implementation of the algorithm for general tensor networks as future work.

Acknowledgments

This work is supported by the US NSF OAC via award No. 1942995.

References

[1] T. D. Ahle, M. Kapralov, J. B. Knudsen, R. Pagh, A. Velingker, D. P. Woodruff, and A. Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 141–160. SIAM, 2020.

[2] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, 2006.

[3] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[4] K. Batselier, W. Yu, L. Daniel, and N. Wong. Computing low-rank approximations of large-scale matrices with the tensor network randomized SVD. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1221–1244, 2018.

[5] C. Battaglino, G. Ballard, and T. G. Kolda. A practical randomized CP tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

[6] G. Beylkin, J. Garcke, and M. J. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 31(3):1840–1857, 2009.

[7] J. C. Bridgeman and C. T. Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22):223001, 2017.

[8] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[9] K. Chen and R. Jin. Tensor-structured sketching for constrained least squares. *SIAM Journal on Matrix Analysis and Applications*, 42(4):1703–1731, 2021.

[10] H. A. Daas, G. Ballard, P. Cazeaux, E. Hallman, A. Miedlar, M. Pasha, T. W. Reid, and A. K. Saibaba. Randomized algorithms for rounding in the tensor-train format. *arXiv preprint arXiv:2110.04393*, 2021.

[11] C. Damm, M. Holzer, and P. McKenzie. The complexity of tensor calculus. *computational complexity*, 11(1-2):54–89, 2002.

[12] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

[13] R. A. Harshman. Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis. *University of California at Los Angeles Los Angeles, CA*, 1970.

[14] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927.

[15] E. G. Hohenstein, R. M. Parrish, and T. J. Martínez. Tensor hypercontraction density fitting. I. quartic scaling second-and third-order Møller-Plesset perturbation theory. *The Journal of chemical physics*, 137(4):044103, 2012.

[16] F. Hummel, T. Tsatsoulis, and A. Grüneis. Low rank factorization of the Coulomb integrals for periodic coupled cluster theory. *The Journal of chemical physics*, 146(12):124105, 2017.

[17] R. Jin, T. G. Kolda, and R. Ward. Faster Johnson-Lindenstrauss transforms via Kronecker products. *arXiv preprint arXiv:1909.04801*, 2019.

[18] D. Kane, R. Meka, and J. Nelson. Almost optimal explicit Johnson-Lindenstrauss families. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 628–639. Springer, 2011.

[19] D. M. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[20] S. Klus, P. Gelß, S. Peitz, and C. Schütte. Tensor-based dynamic mode decomposition. *Nonlinearity*, 31(7):3359, 2018.

[21] S. Klus, P. Koltai, and C. Schütte. On the numerical approximation of the Perron-Frobenius and Koopman operator. *arXiv preprint arXiv:1512.05997*, 2015.

[22] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[23] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic. Tensorly: Tensor learning in python. *arXiv preprint arXiv:1610.09555*, 2016.

[24] B. W. Larsen and T. G. Kolda. Practical leverage-based sampling for low-rank tensor decomposition. *arXiv preprint arXiv:2006.16438*, 2020.

[25] L. Ma and E. Solomonik. Efficient parallel CP decomposition with pairwise perturbation and multi-sweep dimension tree. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 412–421. IEEE, 2021.

[26] L. Ma and E. Solomonik. Fast and accurate randomized algorithms for low-rank tensor decompositions. In *Advances in Neural Information Processing Systems*, 2021.

[27] L. Ma and C. Yang. Low rank approximation in simulations of quantum algorithms. *Journal of Computational Science*, page 101561, 2022.

[28] L. Ma, J. Ye, and E. Solomonik. AutoHOOT: Automatic high-order optimization for tensors. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, pages 125–137, 2020.

[29] A. V. Mahankali, D. P. Woodruff, and Z. Zhang. Low rank approximation for general tensor networks. *arXiv preprint arXiv:2207.07417*, 2022.

[30] O. A. Malik. More efficient sampling for tensor decomposition. *arXiv preprint arXiv:2110.07631*, 2021.

[31] M. Meister, T. Sarlos, and D. P. Woodruff. Tight dimensionality reduction for sketching low degree polynomial kernels. In *NeurIPS*, pages 9470–9481, 2019.

[32] S. M. Nascimento, K. Amano, and D. H. Foster. Spatial distributions of local illumination color in natural scenes. *Vision research*, 120:39–44, 2016.

[33] A. Obukhov, M. Rakhuba, A. Liniger, Z. Huang, S. Georgoulis, D. Dai, and L. Van Gool. Spectral tensor train parameterization of deep learning layers. In *International Conference on Artificial Intelligence and Statistics*, pages 3547–3555. PMLR, 2021.

[34] T. E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[35] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[36] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[37] R. Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[38] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247, 2013.

[39] A.-H. Phan, P. Tichavskỳ, and A. Cichocki. Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013.

[40] M. Pilanci and M. J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.

[41] B. Rakhshan and G. Rabusseau. Tensorized random projections. In *International Conference on Artificial Intelligence and Statistics*, pages 3306–3316. PMLR, 2020.

[42] L. Richter, L. Sallandt, and N. Nüsken. Solving high-dimensional parabolic PDEs using the tensor train format. In *International Conference on Machine Learning*, pages 8998–9009. PMLR, 2021.

[43] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152. IEEE, 2006.

[44] U. Schollwöck. The density-matrix renormalization group. *Reviews of modern physics*, 77(1):259, 2005.

[45] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of physics*, 326(1):96–192, 2011.

[46] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[47] E. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[48] L. G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[49] G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902, 2003.

[50] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Theoretical Computer Science*, 10(1-2):1–157, 2014.

[51] D. P. Woodruff and A. Zandieh. Leverage score sampling for tensor product matrices in input sparsity time. *arXiv preprint arXiv:2202.04515*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The limitations (assumptions in our theoretical analysis) are summarized multiple places throughout the paper, including Section 1 and Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our work focuses on fast sketching techniques for numerical linear algebra and there is no direct potential negative societal impacts.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] These are stated throughout the paper, including Section 1, Section 7, and sections presenting the detailed theory (Section 3 and Section 4).

(b) Did you include complete proofs of all theoretical results? [Yes] Detailed proofs are presented in the appendix.

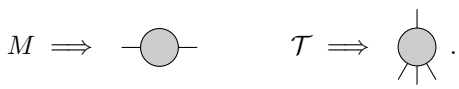
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars are reported in Fig. 5 and Fig. 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The computational resources used is shown in the first paragraph of Section 6.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the NumPy library for experiments and it is cited in the paper.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is included in the supplemental materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Background

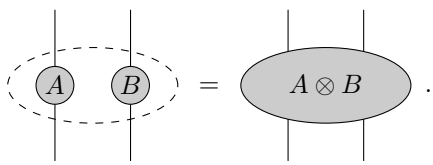
A.1 Tensor algebra and tensor diagram notation

Our analysis makes use of tensor algebra for tensor operations [22]. Vectors are denoted with lowercase Roman letters (e.g., v), matrices are denoted with uppercase Roman letters (e.g., M), and tensors are denoted with calligraphic font (e.g., $\mathcal{T}$). An order N tensor corresponds to an N-dimensional array. For an order N tensor $\mathcal{T} \in \mathbb{R}^{s_1 \times \cdots \times s_N}$, the size of ith dimension is s_i. The ith column of the matrix M is denoted by $M(:, i)$, and the ith row is denoted by $M(i, :)$. Subscripts are used to label different vectors, matrices and tensors (e.g. $\mathcal{T}_1$ and $\mathcal{T}_2$ are unrelated tensors). The Kronecker product of two vectors/matrices is denoted with $\otimes$, and the outer product of two or more vectors is denoted with $\circ$. For matrices $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{n \times k}$, their Khatri-Rao product results in a matrix of size $(mn) \times k$ defined by $A \odot B = [A(:, 1) \otimes B(:, 1), \ldots, A(:, k) \otimes B(:, k)]$. Matricization is the process of unfolding a tensor into a matrix. The dimension-n matricized version of $\mathcal{T}$ is denoted by $T_{(n)} \in \mathbb{R}^{s_n \times K}$ where $K = \prod_{m=1, m \neq n}^{N} s_m$.

We introduce the graph representation for tensors, which is also called tensor diagram [7]. A tensor is represented by a vertex with hyperedges adjacent to it, each corresponding to a tensor dimension. A matrix M and an order four tensor $\mathcal{T}$ are represented as follows,



The Kronecker product of two matrices A and B can be expressed as



Connecting two edges means two tensor dimensions are contracted or summed over. One example is shown in Fig. 7.

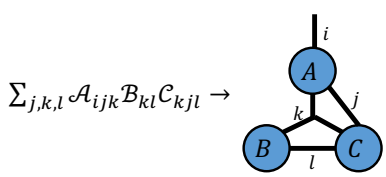


Figure 7: An example of tensor diagram notation.

A.2 Background on sketching

In this section, we introduce definitions for sketching used throughout the paper.

Definition 2 (Gaussian embedding)**.** A matrix $S = \frac{1}{\sqrt{m}} M \in \mathbb{R}^{m \times n}$ is a Gaussian embedding if each element of M is a normalized Gaussian random variable, $M(i, j) \sim N(0, 1)$.

One key property we would like the tensor network embedding to satisfy is the (ϵ, δ)-accurate property. To achieve this, one central property each tensor in the tensor network embedding needs to satisfy is the Johnson-Lindenstrauss (JL) moment property. The JL moment property captures a bound on the moments of the difference between the vector Euclidean norm and the norm after sketching. We introduce both definitions below.

Definition 3 ((ϵ, δ)-accurate embedding)**.** A random matrix $S \in \mathbb{R}^{m \times n}$ has the (ϵ, δ)-accurate embedding property if for every $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$,

$$\Pr_S \left(\left| \|Sx\|_2^2 - 1 \right| > \epsilon \right) < \delta.$$

Definition 4 ((ϵ, δ, p)-JL moment [18, 19]). A random matrix $S \in \mathbb{R}^{m\times n}$ has the (ϵ, δ, p)-JL moment property if for every $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$,

$$\mathop{\mathbb{E}}_{S}\left|\|Sx\|_2^2 - 1\right|^p < \epsilon^p\delta \quad \text{and} \quad \mathbb{E}\left[\|Sx\|_2^2\right] = 1.$$

Definition 5 (Strong (ϵ, δ)-JL moment [18, 1]). A random matrix $S \in \mathbb{R}^{m\times n}$ has the strong (ϵ, δ)-JL moment property if for every $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$, and every integer $p \in [2, \log(1/\delta)]$,

$$\mathop{\mathbb{E}}_{S}\left|\|Sx\|_2^2 - 1\right|^p < \left(\frac{\epsilon}{e}\right)^p \left(\frac{p}{\log(1/\delta)}\right)^{p/2} \tag{A.1}$$

and $\mathbb{E}\left[\|Sx\|_2^2\right] = 1$.

Note that the strong (ϵ, δ)-JL moment property directly reveals the $(\epsilon, \delta, \log(1/\delta))$-JL moment property, since letting $p = \log(1/\delta)$, (A.1) becomes

$$\mathop{\mathbb{E}}_{S}\left|\|Sx\|_2^2 - 1\right|^{\log(1/\delta)} < \left(\frac{\epsilon}{e}\right)^{\log(1/\delta)} = \epsilon^p\delta.$$

Both the strong (ϵ, δ)-JL moment property and the (ϵ, δ, p)-JL moment property directly imply (ϵ, δ)-accurate embedding via Markov's inequality,

$$\Pr_{S}\left(\left|\|Sx\|_2^2 - 1\right| > \epsilon\right) < \frac{\mathbb{E}\left|\|Sx\|_2^2 - 1\right|^p}{\epsilon^p} < \delta.$$

The lemmas below show that Gaussian embeddings can be used to construct embeddings with the JL moment property.

Lemma A.1 (Strong JL moment of Gaussian embeddings [18]). *Gaussian embeddings with $m = \Omega(\log(1/\delta)/\epsilon^2)$ satisfy the (ϵ, δ)-strong JL moment property.*

Below we review the composition rules of JL moment properties introduced in [1], which are used to prove the (ϵ, δ)-accurate sufficient condition in Theorem 3.1.

Lemma A.2 (JL moment with Kronecker product). *If a matrix S has the (ϵ, δ, p)-JL moment property, then the matrix $M = I_i \otimes S \otimes I_j$ also has the (ϵ, δ, p)-JL moment property for identity matrices I_i and I_j with any size. This relation also holds for the strong (ϵ, δ)-JL moment property.*

Lemma A.3 (Strong JL moment with matrix product). *There exists a universal constant L, such that for any constants $\epsilon, \delta \in [0, 1]$ and any integer k, if $M_1 \in \mathbb{R}^{d_2\times d_1}, \cdots, M_k \in \mathbb{R}^{d_{k+1}\times d_k}$ are independent random matrices, each having the strong $\left(\frac{\epsilon}{L\sqrt{k}}, \delta\right)$-JL moment property, then the product matrix $M = M_k \cdots M_1$ satisfies the strong (ϵ, δ)-JL moment property.*

B Definitions and basic properties of tensor network embedding

In this section, we introduce definitions and basic properties of tensor network embeddings. These properties will be used in Appendix C and Appendix D for detailed computational cost analysis. The notation defined in the main text is summarized in Table 4, which is also used in later analysis.

B.1 Graph notation for tensor network and tensor contraction

We use undirected hypergraphs to represent tensor networks. For a given hypergraph $G = (V, E, w)$, V represents the vertex set, E represents the set of hyperedges, and w is a function such that $w(e)$ is the natural logarithm of the tensor dimension size represented by the hyperedge $e \in E$. We use $E(u, v)$ to denote the set of hyperedges adjacent to both u and v, which includes the edge (u, v) and hyperedges adjacent to u, v. We use $E(A, B)$ to denote the set of hyperedges connecting two subsets A, B of V with $A \cap B = \emptyset$. We use $E(A, *)$ to denote all uncontracted edges only adjacent to A, $E(A, *) = \{(u) \in E : u \in A\}$. we illustrate $E(A, B), E(A, *)$ in Fig. 8. For any set $A \subseteq V$, we let

$$E(A) = E(A, V \setminus A) \cup E(A, *). \tag{B.1}$$

Notations	Meanings
S, S_i	Embedding matrix
m	Sketch size
$G_E = (V_E, E_E, w)$	Embedding tensor network
$G_D = (V_D, E_D, w)$	Input data tensor network
$\bar{E} = \{e_1, \ldots, e_N\}$	Set of edges to be sketched
s_i	Size of e_i in $\bar{E}$
T_0	Given data contraction tree
$\mathcal{D}(e_1), \ldots, \mathcal{D}(e_N), \mathcal{S}, \mathcal{I}$	Subsets of contractions in T_0
$X(e_i)$	Sub network contracted by $\mathcal{D}(e_i)$

Table 4: Notations used throughout the paper.

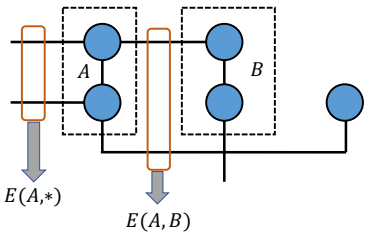


Figure 8: An example of $E(A, *)$ and $E(A, B)$, where both A, B are subset of vertices.

A tensor network implicitly represents a tensor with a set of (small) tensors and a specific contraction pattern. We use $G[A] = (A, E_A, w)$ to denote a sub tensor network defined on $A \subseteq V$, where E_A contains all hyperedges in E adjacent to any $v \in A$.

Our analysis also use directed graphs to represent tensor network linearizations. We use $E(u, v)$ to denote the edge from u to v, and similarly use $E(A, B)$ to denote the set of edges from A to B.

When representing the contraction tree, we use (v_1, v_2) to denote the contraction of v_1, v_2. This notation is also used to represent multiple contractions. For example, we use $(((v_1, v_4), (v_2, v_5)), v_3)$ to represent the contraction tree shown in Fig. 9. The computational cost of a contraction tree is the summation of each contraction's cost. In the discussion throughout the paper, we assume that all tensors in the network are dense. Therefore, the contraction of two general dense tensors $\mathcal{A}$ and $\mathcal{B}$, represented as vertices v_a and v_b in $G = (V, E, w)$, can be cast as a matrix multiplication, and the overall asymptotic cost is

$$\Theta(\exp(w(E(v_a)) + w(E(v_b)) - w(E(v_a, v_b))))$$

with classical matrix multiplication algorithms. In general, contracting tensor networks with arbitrary structure is #P-hard [11, 48].

Here is an example of constrained contraction tree, which is defined in Definition 1. Consider a tensor network with three tensors, v_1, v_2, v_3, with a given contraction tree T_0 that is $((v_1, v_2), v_3)$, which indicates that v_1 first contracts with v_2 and subsequently with v_3. Consider an additional tensor network consisting of v_1, v_2, v_3 and another tensors u. Then the contraction tree $(((v_1, v_2), u), v_3)$, $(((v_1, u), v_2), v_3)$ and $(((v_1, v_2), v_3), u)$ are all constrained on T_0, since the contraction ordering of v_1, v_2, v_3 remains unchanged. However, the contraction tree $(((v_1, v_3), u), v_2)$ is not constrained on T_0.

For a given data G_D and its given contraction tree T_0, its dimension tree is a directed binary tree showing the way edges in $\bar{E}$ are merged onto the same tensor. Each vertex in the dimension tree is a subset $E' \subseteq \bar{E}$, and for any two vertices E'_1, E'_2 of the dimension tree with the same parent, there is

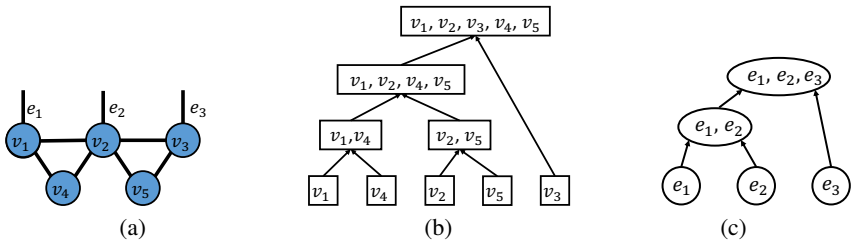


Figure 9: Example of a tensor network, its contraction tree and the corresponding dimension tree.

a contraction in T_0 such that the two input tensors are incident to E_1', E_2', respectively. One example is shown in Fig. 9.

B.2 Definitions used in the analysis of tensor network embedding

In this section, we introduce definitions that will be used in later analysis. For a (hyper)graph $G = (V, E, w)$ and two subsets of V denoted as A, B, we define $\mathsf{cut}_G(A, B) = \sum_{e \in E(A,B)} w(e)$. Similarly, we define $\mathsf{cut}_G(A, *) = \sum_{e \in E(A,*)} w(e)$, and define $\mathsf{cut}_G(A) = \sum_{e \in E(A)} w(e)$, where $E(A)$ is expressed in (B.1). When G is a directed hypergraph, $\mathsf{cut}_G(A, B)$ denotes the sum of the weights of edges from A to B. When G is an undirected graph, $\mathsf{cut}_G(A, B)$ denotes the sum of the weights of hyperedges connecting A and B.

For two tensors represented by two subsets $A, B \subset V$ and $A \cap B = \emptyset$, the logarithm of the contraction cost between a tensor represented by A and a tensor represented by B, (A, B), is

$$\mathsf{cost}_G(A, B) = \mathsf{cut}_G(A) + \mathsf{cut}_G(B) - \mathsf{cut}_G(A, B).$$

Note that the function cost is only defined on undirected hypergraphs.

Consider a given input data $G_D = (V_D, E_D, w)$ and an embedding $G_E = (V_E, E_E, w)$. Below we let $V = V_E \cup V_D$, $E = E_E \cup E_D$, and $G = (V, E, w)$ denote the hypergraph including both the embedding and the input data. We use $L = (V, E_E, w)$ to denote the graph including V and all edges in the embedding, and use $R = (V, E \setminus E_E, w)$. Note that in this work we focus on the case where L is a graph, and R can be a general hypergraph. We illustrate G, G_D, G_E, L, R in Fig. 10. For any $A, B \subset V$ and $A \cap B = \emptyset$, we have

$$\mathsf{cut}_G(A) = \mathsf{cut}_L(A) + \mathsf{cut}_R(A), \tag{B.2}$$

and

$$\mathsf{cut}_G(A, B) = \mathsf{cut}_L(A, B) + \mathsf{cut}_R(A, B). \tag{B.3}$$

Based on (B.2) and (B.3), we have

$$\mathsf{cost}_G(A, B) = \mathsf{cost}_L(A, B) + \mathsf{cost}_R(A, B).$$

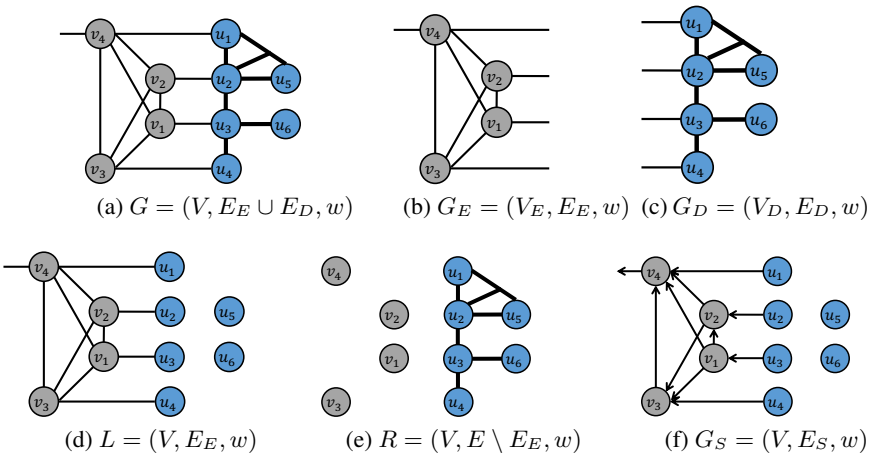
Our analysis of tensor network embedding is based on the linearization of the tensor network graph. Linearization casts an undirected graph into a *directed acyclic graph* (DAG). We define linearization formally below, then specify linearizations of the data and embedding graphs that our analysis considers.

Definition 6 (Linearization DAG)**.** A linearization of the undirected graph $G = (V, E, w)$ is defined by the DAG $G' = (V, E', w)$ induced by a given choice of vertex ordering in V. For each contracted edge in E, E' contains an same-weight edge directing towards the higher indexed vertex. For each uncontracted edge in E, E' contains an edge with the same weight that is directed outward from the vertex it is adjacent to.

Based on Definition 6, we define the *sketching linearization DAG*, $G_S = (V, E_S, w)$, as a DAG defined on top of the graph $L = (V, E_E, w)$, which includes all vertices in both the embedding and the data and all embedding edges. For a given vertex ordering of embedding vertices, G_S is the linearization of L based on the ordering with all data vertices being ordered ahead of embedding vertices.

As discussed in Section 3, for a given sketching linearization, the sketching accuracy of each tensor $\mathcal{A}_i$ at v_i is dependent on the row size of its matricization A_i, which is the weighted size of the edge set adjacent to v_i containing all uncontracted edges and contracted edges also adjacent to v_j with $j > i$, which is called *effective sketch dimension* of v_i throughout the paper. Based on the definition, when $v \in V_E$, $\mathsf{cut}_{G_S}(v)$ equals the effective sketch dimension size of v. When $v \in V_D$, $\mathsf{cut}_{G_S}(v)$ represents the size of the sketch dimension adjacent to v. We look at embeddings G_E not only satisfying the (ϵ, δ)-accurate sufficient condition in Theorem 3.1, but also only have one output sketch dimension ($|E_1| = 1$) with the output sketch size $m = \Theta\big(N_E \log(1/\delta)/\epsilon^2\big)$. For each one of these embeddings, there must exist a linearization G_S such that for all $v \in V_E$, we have

$$\mathsf{cut}_{G_S}(v) = \Omega(\log(m)). \tag{B.4}$$



(a) $G = (V, E_E \cup E_D, w)$ (b) $G_E = (V_E, E_E, w)$ (c) $G_D = (V_D, E_D, w)$

(d) $L = (V, E_E, w)$ (e) $R = (V, E \setminus E_E, w)$ (f) $G_S = (V, E_S, w)$

Figure 10: Illustration of graphs and hypergraphs used throughout the paper.

B.3 Properties of tensor network embedding

We now derive properties that are used in the sketching computational cost analysis. In Lemma B.1, we show relations between cuts in the graph L and cuts in the graph G_S. In Lemma B.2, we show relations between costs in the graph L and cuts in the graph G_S. Lemma B.2 along with cut lower bounds (B.4) is used to derive lower bounds for cost_L and cost_G in Appendix D.

Lemma B.1. *Consider an embedding $G_E = (V_E, E_E, w)$ and a data tensor network $G_D = (V_D, E_D, w)$, and a given sketching linearization $G_S = (V, E_S, w)$, where $V = V_E \cup V_D$. For any $A, B \subset V$ and $A \cap B = \emptyset$, the following relations hold,*

$$\mathsf{cut}_L(A) = \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(V \setminus A, A), \tag{B.5}$$

$$\mathsf{cut}_L(A, B) = \mathsf{cut}_{G_S}(A, B) + \mathsf{cut}_{G_S}(B, A), \tag{B.6}$$

$$\begin{aligned}\mathsf{cut}_{G_S}(A \cup B) &= \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) - \mathsf{cut}_{G_S}(A, B) - \mathsf{cut}_{G_S}(B, A) \\ &= \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) - \mathsf{cut}_L(A, B).\end{aligned} \tag{B.7}$$

Proof. (B.5) and (B.6) hold directly based on the definition of the linearization DAG. For (B.7), based on (B.1), we have

$$\begin{aligned}\mathsf{cut}_{G_S}(A \cup B) &= \mathsf{cut}_{G_S}(A \cup B, V \setminus (A \cup B)) + \mathsf{cut}_{G_S}(A \cup B, *) \\ &= \mathsf{cut}_{G_S}(A, V \setminus (A \cup B)) + \mathsf{cut}_{G_S}(B, V \setminus (A \cup B)) + \mathsf{cut}_{G_S}(A \cup B, *) \\ &= \mathsf{cut}_{G_S}(A, V \setminus A) - \mathsf{cut}_{G_S}(A, B) + \mathsf{cut}_{G_S}(B, V \setminus B) - \mathsf{cut}_{G_S}(B, A) \\ &+ \mathsf{cut}_{G_S}(A, *) + \mathsf{cut}_{G_S}(B, *) \\ &= \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) - \mathsf{cut}_{G_S}(A, B) - \mathsf{cut}_{G_S}(B, A).\end{aligned} \tag{B.8}$$

Note that the second and third equalities in (B.8) hold since A and B are disjoint sets. This finishes the proof. □

Lemma B.2. *Consider any data $G_D = (V_D, E_D, w)$ and embedding $G_E = (V_E, E_E, w)$, and a sketching linearization $G_S = (V, E_S, w)$, where $V = V_D \cup V_E$. For any two subsets $A, B \in V$ such that $A \cap B = \emptyset$, the contraction of two tensors that are the contraction outputs of A and B has a logarithm cost of*

$$\mathsf{cost}_L(A, B) = \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) + \mathsf{cut}_{G_S}(V \setminus (A \cup B), A \cup B).$$

Proof. Based on Lemma B.1, we have

$$\mathsf{cut}_L(A) \overset{(B.5)}{=} \mathsf{cut}_{G_S}(V \setminus A, A) + \mathsf{cut}_{G_S}(A), \tag{B.9}$$

$$\mathsf{cut}_L(B) \overset{(B.5)}{=} \mathsf{cut}_{G_S}(V \setminus B, B) + \mathsf{cut}_{G_S}(B). \tag{B.10}$$

Based on (B.6), we have

$$\begin{aligned}
&\mathsf{cut}_{G_S}(V \setminus A, A) + \mathsf{cut}_{G_S}(V \setminus B, B) - \mathsf{cut}_L(A, B) \\
&= \mathsf{cut}_{G_S}(V \setminus A, A) + \mathsf{cut}_{G_S}(V \setminus B, B) - \mathsf{cut}_{G_S}(A, B) - \mathsf{cut}_{G_S}(B, A) \\
&= \mathsf{cut}_{G_S}(V \setminus (A \cup B), A) + \mathsf{cut}_{G_S}(V \setminus (A \cup B), B) \\
&= \mathsf{cut}_{G_S}(V \setminus (A \cup B), A \cup B).
\end{aligned} \tag{B.11}$$

Based on (B.9),(B.10), (B.11), we have

$$\begin{aligned}
\mathsf{cost}_L(A, B) &= \mathsf{cut}_L(A) + \mathsf{cut}_L(B) - \mathsf{cut}_L(A, B) \\
&= \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) + \mathsf{cut}_{G_S}(V \setminus (A \cup B), A \cup B).
\end{aligned}$$

This finishes the proof. □

Lemma B.3. *Consider any data $G_D = (V_D, E_D, w)$ and an embedding $G_E = (V_E, E_E, w)$, and a sketching linearization $G_S = (V, E_S, w)$ such that the embedding is (ϵ, δ)-accurate. Then for any $U \subseteq V$ such that there exists $v \in U$ and $\mathsf{cut}_{G_S}(v) \geq \log(m)$, we have $\mathsf{cut}_{G_S}(U) \geq \log(m)$.*

Proof. When U is a subset of the data vertices, $U \subseteq V_D$, this holds directly since

$$\mathsf{cut}_{G_S}(U) = \sum_{u \in U} \mathsf{cut}_{G_S}(u) \geq \mathsf{cut}_{G_S}(v) \geq \log(m).$$

Next we consider the case where $U \cap V_E \neq \emptyset$. Let $A = U \cap V_E$ and $B = U \cap V_D$. Based on the definition of DAG, there is no directed cycle in the subgraph $G_S[A]$. Therefore, there exists one vertex $s \in A$, such that $\mathsf{cut}_{G_S}(s, A \setminus \{s\}) = 0$. Based on Lemma B.1, we have

$$\begin{aligned}
\mathsf{cut}_{G_S}(A) &\overset{(B.7)}{=} \mathsf{cut}_{G_S}(s) + \mathsf{cut}_{G_S}(A \setminus \{s\}) - \mathsf{cut}_{G_S}(s, A \setminus \{s\}) - \mathsf{cut}_{G_S}(A \setminus \{s\}, s) \\
&\geq \mathsf{cut}_{G_S}(s) - \mathsf{cut}_{G_S}(s, A \setminus \{s\}) \\
&= \mathsf{cut}_{G_S}(s) \overset{(B.4)}{\geq} \log(m),
\end{aligned}$$

In addition, we have $\mathsf{cut}_{G_S}(A, B) = 0$ since $A \subseteq V_E$ and $B \subseteq V_D$. Thus we have

$$\begin{aligned}
\mathsf{cut}_{G_S}(U) = \mathsf{cut}_{G_S}(A \cup B) &\overset{(B.7)}{=} \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) - \mathsf{cut}_{G_S}(A, B) - \mathsf{cut}_{G_S}(B, A) \\
&= \mathsf{cut}_{G_S}(A) + \mathsf{cut}_{G_S}(B) - \mathsf{cut}_{G_S}(B, A) \\
&\geq \mathsf{cut}_{G_S}(A) \geq \log(m).
\end{aligned}$$

This finishes the proof. □

C Computationally-efficient sketching algorithm

In this section, we introduce the detail of the computationally-efficient sketching algorithm in Algorithm 1. Consider a given data tensor network $G_D = (V_D, E_D, w)$ and a given data contraction tree, T_0. Also let $N_D = |V_D|$, and let $\bar{E} \subseteq E_D$ denote the set of edges to be sketched, and $N = |\bar{E}|$. Below we let $\bar{E} = \{e_1, e_2, \ldots, e_N\}$, and let each e_i has weight $\log(s_i) > \log(m)$. Based on the

definition we have $N \leq N_D$. Let one contraction path representing T_0 be expressed as a sequence of $N_D - 1$ contractions,

$$\{(U_1, V_1), \ldots, (U_{N_D-1}, V_{N_D-1})\}. \tag{C.1}$$

Above we use (U_i, V_i) to represent the contraction of two intermediate tensors represented by two subset of vertices $U_i, V_i \subset V_D$. Below we let

$$\begin{aligned} a_i &= \exp(\mathsf{cut}_R(U_i) - \mathsf{cut}_R(U_i, V_i)), \\ c_i &= \exp(\mathsf{cut}_R(V_i) - \mathsf{cut}_R(U_i, V_i)), \\ d_i &= \exp(\mathsf{cut}_R(U_i \cup V_i))/(a_i c_i), \\ b_i &= \exp(\mathsf{cut}_R(U_i, V_i))/d_i. \end{aligned} \tag{C.2}$$

Note that d_i represents the size of uncontracted dimensions adjacent to both U_i and V_i, and b_i represents the size of contracted dimensions between U_i and V_i. We also have $\mathsf{cost}_R(U_i, V_i) = \log(a_i b_i c_i d_i)$, and $\mathsf{cut}_R(U_i \cup V_i) = \log(a_i c_i d_i)$. a_i, b_i, c_i, d_i are visualized in Fig. 11.

In Section 4, the $N_D - 1$ contractions are categorized into $N + 2$ sets, $\mathcal{D}(e_1), \ldots, \mathcal{D}(e_N), \mathcal{S}, \mathcal{I}$, where $\mathcal{D}(e_i)$ contains contractions such that e_i is the only data edge adjacent to the contraction output and in $\bar{E}$, $\mathcal{S}$ contains contractions (U_i, V_i) such that both U_i and V_i are adjacent to edges in $\bar{E}$, and $\mathcal{I}$ includes (U_i, V_i) such that both U_i and V_i are not adjacent to $\bar{E}$. An illustration of these sets is provided below.

Consider an input data consisting of five tensors, v_1, v_2, v_3, v_4, v_5, where v_1 is adjacent to the edge e_1, v_2 is adjacent to e_2, v_3 is adjacent to e_3, and e_1, e_2, e_3 are the edges to be sketched. There are no edges to be sketched adjacent to v_4, v_5. Consider the contraction tree $(((v_1, v_4), (v_2, v_5)), v_3)$, where v_1 contracts with v_4 and outputs $v_{1,4}$, v_2 contracts with v_5 and outputs $v_{2,5}$, and then $v_{1,4}, v_{2,5}$ contract together into $v_{1,2,4,5}$, and $v_{1,2,4,5}$ contracts with v_3. We have $\mathcal{D}(e_1) = \{(v_1, v_4)\}$ and $\mathcal{D}(e_2) = \{(v_2, v_5)\}$, since v_1, v_2 are adjacent to e_1, e_2, respectively. We also have $\mathcal{I} = \emptyset$ and $\mathcal{D}(e_3) = \emptyset$, since each contraction is adjacent to at least one edge in $\{e_1, e_2, e_3\}$, and there is no contraction such that e_3 is the only data edge in the output. All the remaining contractions are in $\mathcal{S}$, so $\mathcal{S} = \{(v_{1,4}, v_{2,5}), (v_{1,2,4,5}, v_3)\}$.

C.1 Sketching with the embedding containing a binary tree of small tensor networks

We now present the details of applying the embedding containing a binary tree of small tensor networks. In Section 4, we define $\mathcal{S}$ as the set containing contractions (U_i, V_i) such that both U_i and V_i are adjacent to edges in $\bar{E}$. For each contraction $i \in \mathcal{S}$, one small embedding tensor network (denoted as Z_i) is applied to the contraction. Let $\hat{U}_i, \hat{V}_i$ denote the sketched U_i and V_i formed in previous contractions in the sketching contraction tree T_B, such that $\hat{U}_i \cap V_D = U_i$ and $\hat{V}_i \cap V_D = V_i$. The structure of Z_i is determined so that the asymptotic cost to sketch $(\hat{U}_i, \hat{V}_i)$ is minimized, under the constraint that Z_i is $(\epsilon/\sqrt{N}, \delta)$-accurate and only has one output sketch dimension.

The structure of Z_i is illustrated in Fig. 11. For the case $a_i \leq c_i$, the structure is shown in Fig. 11a, and sketching is performed via the contraction sequence of contracting $\hat{U}_i$ and v_1 first, then with $\hat{V}_i$, and then with v_2 (also denoted as a contraction sequence of $(((\hat{U}_i, v_1), \hat{V}_i), v_2)$). For the case $a_i > c_i$, the structure of Z_i is shown in Fig. 11b, and the sketching is performed via the contraction sequence of $(((\hat{V}_i, v_2), \hat{U}_i), v_1)$. With this algorithm, sketching $(\hat{U}_i, \hat{V}_i)$ yields a computational cost proportional to

$$y_i = a_i b_i c_i d_i m^2 + m^2 d_i \sqrt{a_i b_i c_i m} \cdot \min(\sqrt{a_i}, \sqrt{c_i}). \tag{C.3}$$

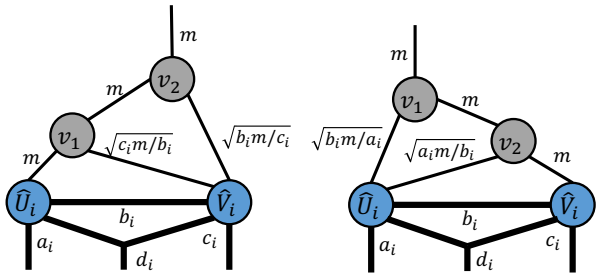
We show in Lemma D.6 that the asymptotic cost lower bound to sketch (U_i, V_i) is also $\Omega(y_i)$.

C.2 Computational cost analysis

We provide the computational cost analysis of Algorithm 1 in this section.

Theorem C.1. *Algorithm 1 has an asymptotic computational cost of*

$$\Theta\left(\sum_{j=1}^{N} t(e_j) + \sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{I}} z_i\right), \tag{C.4}$$



(a) (b)

Figure 11: Illustration of the small network in the binary tree structured embedding. For each edge e, we show the dimension size of that edge (exponential in $w(e)$).

where $t(e_j)$ is the optimal asymptotic cost to sketch the sub tensor network $X(e_j)$ (defined in Table 4) with a matrix in the Kronecker product embedding, y_i is expressed in (C.3), and

$$z_i = a_i b_i c_i d_i \cdot \min(\exp(\mathsf{cut}_{G_S}(U_i \cup V_i)), m), \tag{C.5}$$

where a_i, b_i, c_i, d_i are expressed in (C.2).

Proof. The terms $\sum_{j \in N} t(e_j) + \sum_{i \in \mathcal{S}} y_i$ can be easily verified based on the analysis in Section 4 and Appendix C.1.

Consider the contractions in $\mathcal{I}$, which include (U_i, V_i) such that both U_i and V_i are not adjacent to $\bar{E}$, and contractions where U_i or V_i is adjacent to at least two edges in $\bar{E}$. The first type of contractions in $\mathcal{I}$ would have a cost of $\Theta(a_i b_i c_i d_i)$, and not be affected by previous sketching steps. For the second type, application of the Kronecker product and binary tree embeddings to U_i and V_i would reduce all adjacent edges in $\bar{E}$ to a single dimension of size m. Consequently, the contraction cost would be $\Theta(a_i b_i c_i d_i \cdot m)$. Summarizing both cases prove the cost in (C.5). The cost in (C.4) follows from combining the terms $\sum_{j \in N} t(e_j) + \sum_{i \in \mathcal{S}} y_i$ and $\sum_{i \in \mathcal{I}} z_i$. □

For the special case where each vertex in the data tensor network is adjacent to an edge to be sketched, we have $\mathcal{D}(e_j) = \emptyset$ for all $j \in [N]$ and $\mathcal{I} = \emptyset$, thus all the contractions are in the set $\mathcal{S}$. Therefore, sketching each e_j has an asymptotic cost of $\Theta(t(e_j)) = \Theta(\exp(\mathsf{cut}_G(v_j)) \cdot m)$, where v_j is the vertex in the data graph adjacent to e_j, and Theorem C.1 implies that the sketching cost would be

$$\Theta\left(\sum_{j=1}^{N} t(e_j) + \sum_{i \in \mathcal{S}} y_i + \sum_{i \in \mathcal{I}} z_i\right) = \Theta\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_G(v_j)) \cdot m + \sum_{j=1}^{N-1} y_i\right). \tag{C.6}$$

As we will show in Theorem D.1, this cost matches the asymptotic cost lower bound, when the embedding satisfies the (ϵ, δ)-accurate sufficient condition and only has one output sketch dimension.

When the data has a Kronecker product structure, we have $\mathsf{cut}_G(v_j) = w(e_j) = \log(s_j)$, and $a_i, b_i, c_i, d_i = 1$ for all $i \in \{1, \ldots, N-1\}$ for all contraction trees. Therefore,

$$y_i = a_i b_i c_i d_i m^2 + m^2 d_i \sqrt{a_i b_i c_i m} \cdot \min(\sqrt{a_i}, \sqrt{c_i}) = m^2 + m^{2.5},$$

and the sketching cost is

$$\Theta\left(\sum_{j=1}^{N} s_j m + N m^{2.5}\right). \tag{C.7}$$

As we will show in Appendix E, sketching with tree tensor network embeddings yield an asymptotic cost of $\Theta\left(\sum_{j=1}^{N} s_j m + N m^3\right)$. Therefore, Algorithm 1 is more efficient to sketch Kronecker product input data.

D Lower bound analysis

In this section, we discuss the *asymptotic* computational lower bound for sketching with embeddings satisfying the (ϵ, δ)-accurate sufficient condition and only have one output sketch edge. In Appendix D.1, we discuss the case where the data has uniform sketch dimensions. In this case, each vertex in the data tensor network is adjacent to an edge to be sketched. In Appendix D.2, we discuss the sketching computational lower bound for a more general case, when the data tensor network can have arbitrary graph structure, and vertices not adjacent to sketch edges are allowed. For both cases, we assume that the size of each dimension to be sketched is greater than the sketch size.

D.1 Sketching data with uniform sketch dimensions

We now discuss the sketching asymptotic cost lower bound when the data $G_D = (V_D, E_D, w)$ has uniform sketch dimensions, where each $v \in V_D$ is adjacent to an edge to be sketched with size lower bounded by the target sketch size, m. We have $N = |\bar{E}| = |V_D|$, and we let the size of each $e_i \in \bar{E}$ be denoted $s_i > m$. We let $V = V_E \cup V_D$ denote the set of all vertices in both the data and the embedding. Below, we show the main theorem using lemmas and notations introduced in Appendix B.

Theorem D.1. *For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition and only has one output sketch dimension, and any contraction tree T_B of (G_D, G_E) constrained on the data contraction tree T_0 expressed in (4.2), the sketching asymptotic cost is lower bounded by*

$$\Omega\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_G(v_j)) \cdot m + \sum_{j=1}^{N-1} y_i\right), \tag{D.1}$$

where $m = \Omega(N \log(1/\delta)/\epsilon^2)$ represents the embedding sketch size, v_j is the vertex in V_D adjacent to e_j, and $\exp(\mathsf{cut}_G(v_j))$ denotes the size of the tensor at v_j, and y_i is expressed in (C.3).

We present the proof of Theorem D.1 at the end of Appendix D.1. Note that the first term in (D.1), $\sum_{v \in V_D} \exp(\mathsf{cut}_G(v)) \cdot m$, is a term independent of the data contraction tree, while the second term is dependent of the data contraction tree.

Proof of Theorem 4.1. The asymptotic cost of of Algorithm 1 in (C.6) matches the lower bound shown in Theorem D.1, thus proving the statement. □

Theorem D.1 also yields an asymptotic lower bound for sketching data with a Kronecker product structure. We state the results below.

Corollary D.2. *Consider an input data G_D representing a vector with a Kronecker product structure and each v_j for $j \in [N]$ is adjacent to an edge to be sketched with size s_j. For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension and any contraction tree T_B of (G_D, G_E), the asymptotic cost must be lower bounded by*

$$\Omega\left(\sum_{j=1}^{N} s_j m + N m^{2.5}\right),$$

where $m = \Omega(N \log(1/\delta)/\epsilon^2)$.

Below, we present some lemmas needed to prove Theorem D.1.

Lemma D.3. *Consider an (ϵ, δ)-accurate embedding $G_E = (V_E, E_E, w)$ with a sketching linearization $G_S = (V, E_S, w)$. Then for any subset of the embedding and data graph vertex set, $W \subseteq V$, we have $\mathsf{cut}_{G_S}(W) \geq \log(m)$.*

Proof. Since each vertex in the data graph is adjacent to an edge to be sketched, and the edge dimension size is greater than m, we have $\mathsf{cut}_{G_S}(w) \geq \log(m)$ for all $w \in V_D$. Since the embedding satisfies the (ϵ, δ)-accurate sufficient condition, we have $\mathsf{cut}_{G_S}(w) \geq \log(m)$ for all $w \in V_E$. Therefore, $\mathsf{cut}_{G_S}(w) \geq \log(m)$ for all $w \in V$. Based on Lemma B.3, $\mathsf{cut}_{G_S}(W) \geq \log(m)$ for all $W \subseteq V$. □

Lemma D.4. *Consider an (ϵ, δ)-accurate embedding G_E with a sketching linearization $G_S = (V, E_S, w)$. Consider any contraction tree T_B for (G_D, G_E). If there exists a contraction output of $U \subset V$ formed in T_B and $\mathsf{cut}_{G_S}(U) > \log(m)$, then the asymptotic cost for the contraction tree T_B must be lower bounded by $\Omega(\exp(\mathsf{cut}_R(U) + \mathsf{cut}_{G_S}(U)) \cdot m)$.*

Proof. Since $\mathsf{cut}_{G_S}(U) > \log(m)$, there must exist a contraction $(U, W) \in T_B$ with W containing some vertices in $V \setminus U$. Based on Lemma D.3, $\mathsf{cut}_{G_S}(W) \geq \log(m)$. Based on Lemma B.2, we have

$$\begin{aligned}\mathsf{cost}_G(U, W) &= \mathsf{cost}_R(U, W) + \mathsf{cost}_L(U, W) \\ &= \mathsf{cost}_R(U, W) + \mathsf{cut}_{G_S}(U) + \mathsf{cut}_{G_S}(W) + \mathsf{cut}_{G_S}(V \setminus (U \cup W), U \cup W).\end{aligned}$$

Further, since $\mathsf{cost}_R(U, W) \geq \mathsf{cut}_R(U)$,

$$\begin{aligned}\mathsf{cost}_G(U, W) &\geq \mathsf{cut}_R(U) + \mathsf{cut}_{G_S}(U) + \mathsf{cut}_{G_S}(W) \\ &\geq \mathsf{cut}_R(U) + \mathsf{cut}_{G_S}(U) + \log(m).\end{aligned}$$

This proves the lemma since the contraction cost is $\Theta(\exp(\mathsf{cost}_G(U, W)))$. □

In Lemma D.6, we show that when the data contraction tree T_0 contains the contraction (U_i, V_i), then any contraction tree T_B of (G_D, G_E) that is constrained on T_0 will yield a contraction cost of $\Omega(y_i)$. To show that, we first discuss the case where T_B also contains the contraction (U_i, V_i) in Lemma D.5. The more general case where the contraction (U_i, V_i) need not be in T_B is discussed in Lemma D.6.

Lemma D.5. *Consider a specific contraction tree T_0 for G_D, where the contraction (U_i, V_i) is in T_0. For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension and any contraction tree T_B of (G_D, G_E) constrained on T_0, if (U_i, V_i) is also in T_B, the sketching asymptotic cost must be lower bounded by $\Omega\big(a_i b_i c_i d_i m^2 + a_i c_i d_i m^3\big)$, where a_i, b_i, c_i, d_i are defined in (C.2).*

Proof. Consider any sketching linearization $G_S = (V, E_S, w)$ such that the embedding satisfies the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension. Based on Lemma D.3, we have $\mathsf{cut}_{G_S}(U_i), \mathsf{cut}_{G_S}(V_i) \geq \log(m)$. Based on Lemma B.2, we have

$$\begin{aligned}\mathsf{cost}_G(U_i, V_i) &= \mathsf{cost}_R(U_i, V_i) + \mathsf{cost}_L(U_i, V_i) \\ &= \mathsf{cost}_R(U_i, V_i) + \mathsf{cut}_{G_S}(U_i) + \mathsf{cut}_{G_S}(V_i) + \mathsf{cut}_{G_S}(V \setminus (U_i \cup V_i), U_i \cup V_i) \\ &\geq \mathsf{cost}_R(U_i, V_i) + \mathsf{cut}_{G_S}(V_i) + \mathsf{cut}_{G_S}(V_i) \\ &\geq \log(a_i b_i c_i d_i) + 2\log(m).\end{aligned}$$

Thus this contraction has a cost of $\Omega\big(a_i b_i c_i d_i \cdot m^2\big)$. In addition, since U_i, V_i are subsets of the data vertices, $\mathsf{cut}_L(U_i, V_i) = 0$. Therefore, based on (B.7),

$$\mathsf{cut}_{G_S}(U_i \cup V_i) = \mathsf{cut}_{G_S}(U_i) + \mathsf{cut}_{G_S}(V_i) \geq 2\log(m).$$

Based on Lemma D.4, the cost needed to sketch $U_i \cup V_i$ is $\Omega\big(a_i c_i d_i m^3\big)$. Thus the overall asymptotic cost is lower bounded by $\Omega\big(a_i b_i c_i d_i m^2 + a_i c_i d_i m^3\big)$. This finishes the proof. □

Lemma D.6. *Consider a specific contraction tree T_0 for G_D, where the contraction (U_i, V_i) is in T_0. For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension and any contraction tree T_B of (G_D, G_E) constrained on T_0, the sketching asymptotic cost must be lower bounded by*

$$\Omega(y_i) = \Omega\Big(a_i b_i c_i d_i m^2 + m^2 d_i \sqrt{a_i b_i c_i m} \cdot \min(\sqrt{a_i}, \sqrt{c_i})\Big), \tag{D.2}$$

where a_i, b_i, c_i, d_i are defined in (C.2), and y_i is defined in (C.3).

Proof. Consider any sketching linearization $G_S = (V, E_S, w)$ such that the embedding satisfies the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension. We first consider the

case where the contraction (U_i, V_i) exists in T_B. Based on Lemma D.5, the overall asymptotic cost is lower bounded by $\Omega\big(a_i b_i c_i d_i m^2 + a_i c_i d_i m^3\big)$. Since

$$a_i b_i c_i d_i m^2 + a_i c_i d_i m^3 = a_i c_i d_i m^2 (b_i + m) \geq 2 a_i c_i d_i m^2 \sqrt{b_i m},$$

the overall asymptotic cost is lower bounded by

$$\Omega\Big(a_i b_i c_i d_i m^2 + a_i c_i d_i m^2 \sqrt{b_i m}\Big) = \Omega\Big(a_i b_i c_i d_i m^2 + a_i m^2 d_i \sqrt{b_i c_i m}\Big),$$

and hence it satisfies (D.2).

We next consider the other case where the contraction (U_i, V_i) is not performed directly in T_B. Since T_B is constrained on T_0, there must exist a contraction $(\hat{U}_i, \hat{V}_i) \in T_B$ with either $\hat{U}_i$ or $\hat{V}_i$ containing embedding vertices, and $\hat{U}_i \cap V_D = U_i$, $\hat{V}_i \cap V_D = V_i$. Let x be the last embedding vertex (based on the linearization order) applied in T_B to $\hat{U}_i \cup \hat{V}_i$, so that $\mathsf{cut}_{G_S}(x, (\hat{U}_i \cup \hat{V}_i) \setminus \{x\}) = 0$. For the case where $x \in \hat{U}_i \setminus U_i$, we show below that the sketching asymptotic cost is lower bounded by

$$\Omega\Big(a_i b_i c_i d_i m^2 + a_i m^2 d_i \sqrt{b_i c_i m}\Big). \tag{D.3}$$

For the other case where $x \in \hat{V}_i \setminus V_i$, we have the cost is lower bounded by $\Omega\big(a_i b_i c_i d_i m^2 + c_i m^2 d_i \sqrt{a_i b_i m}\big)$ by symmetry. Together, these two results prove the lemma.

Detailed proof of (D.3) Since $|\hat{U}_i| > |U_i|$, there must exist a contraction (Y_1, Y_2), for which the output is $\hat{U}_i = Y_1 \cup Y_2$. Based on Lemma B.2, we have

$$\begin{aligned}
\mathsf{cost}_G(Y_1, Y_2) &= \mathsf{cost}_R(Y_1, Y_2) + \mathsf{cost}_L(Y_1, Y_2) \\
&= \mathsf{cost}_R(Y_1, Y_2) + \mathsf{cut}_{G_S}(Y_1) + \mathsf{cut}_{G_S}(Y_2) + \mathsf{cut}_{G_S}(V \setminus (Y_1 \cup Y_2), Y_1 \cup Y_2) \\
&\geq \mathsf{cost}_R(Y_1, Y_2) + \mathsf{cut}_{G_S}(Y_1) + \mathsf{cut}_{G_S}(Y_2) + \mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) \\
&\geq \log(a_i b_i d_i) + 2\log(m) + \mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i).
\end{aligned}$$

Thus, the cost of the contraction (Y_1, Y_2) is lower bounded by

$$\Omega\Big(a_i b_i d_i m^2 \cdot \exp\Big(\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i)\Big)\Big). \tag{D.4}$$

In addition, since

$$\begin{aligned}
\mathsf{cost}_G(\hat{U}_i, \hat{V}_i) &= \mathsf{cost}_R(\hat{U}_i, \hat{V}_i) + \mathsf{cut}_{G_S}(\hat{U}_i) + \mathsf{cut}_{G_S}(\hat{V}_i) + \mathsf{cut}_{G_S}(V \setminus (\hat{U}_i \cup \hat{V}_i), \hat{U}_i \cup \hat{V}_i) \\
&\geq \log(a_i b_i c_i d_i) + 2\log(m),
\end{aligned}$$

the contraction $(\hat{U}_i, \hat{V}_i)$ yields a cost lower bounded by

$$\Omega\big(a_i b_i c_i d_i \cdot m^2\big). \tag{D.5}$$

Combining (D.4) and (D.5), we have that the contractions (Y_1, Y_2) and $(\hat{U}_i, \hat{V}_i)$ have a cost of

$$\Omega\Big(a_i b_i d_i m^2 \cdot \exp\Big(\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i)\Big) + a_i b_i c_i d_i \cdot m^2\Big). \tag{D.6}$$

When $\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) = \log(m)$, (D.6) implies that the overall asymptotic cost is lower bounded by $\Omega(a_i b_i c_i d_i m^2 + a_i b_i d_i m^3)$. Since

$$a_i b_i c_i d_i m^2 + a_i b_i d_i m^3 = a_i b_i d_i m^2 (c_i + m) \geq 2 a_i b_i d_i m^2 \sqrt{c_i m},$$

the overall asymptotic cost is lower bounded by

$$\Omega\big(a_i b_i c_i d_i m^2 + a_i b_i d_i m^2 \sqrt{c_i m}\big) = \Omega\Big(a_i b_i c_i d_i m^2 + a_i m^2 d_i \sqrt{b_i c_i m}\Big).$$

When $\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) < \log(m)$, based on Lemma B.1, the effective sketch dimensions of $\hat{U}_i \cup \hat{V}_i$ satisfy

$$\begin{aligned}
\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) &= \Big(\mathsf{cut}_{G_S}(\hat{U}_i) - \mathsf{cut}_{G_S}(\hat{U}_i, \hat{V}_i)\Big) + \mathsf{cut}_{G_S}(\hat{V}_i) - \mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) \\
&\geq \mathsf{cut}_{G_S}(x) + \mathsf{cut}_{G_S}(\hat{V}_i) - \mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) \\
&\geq 2\log(m) - \mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i),
\end{aligned} \tag{D.7}$$

where the first inequality holds since

$$\begin{aligned}\mathsf{cut}_{G_S}(\hat{U}_i) - \mathsf{cut}_{G_S}(\hat{U}_i, \hat{V}_i) &= \mathsf{cut}_{G_S}(\hat{U}_i, V \setminus (\hat{U}_i \cup \hat{V}_i)) + \mathsf{cut}_{G_S}(\hat{U}_i, *) \\ &\geq \mathsf{cut}_{G_S}(x, V \setminus (\hat{U}_i \cup \hat{V}_i)) + \mathsf{cut}_{G_S}(x, *) = \mathsf{cut}_{G_S}(x),\end{aligned}$$

and the second inequality in (D.7) holds since $\mathsf{cut}_{G_S}(x), \mathsf{cut}_{G_S}(\hat{V}_i) \geq \log(m)$ based on Lemma D.3. Based on the condition $\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i) < \log(m)$ as well as (D.7), we have $\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) > \log(m)$.

Based on Lemma D.4, since $\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) > \log(m)$, there must exist another contraction in T_B to sketch $\hat{U}_i \cup \hat{V}_i$ with a cost of

$$\begin{aligned}\Omega\Big(\exp\Big(\mathsf{cut}_R(\hat{V}_i \cup \hat{U}_i) + \mathsf{cut}_{G_S}(\hat{V}_i \cup \hat{U}_i)\Big) \cdot m\Big) &= \Omega\Big(a_i c_i d_i \cdot \exp\Big(\mathsf{cut}_{G_S}(\hat{V}_i \cup \hat{U}_i)\Big) \cdot m\Big) \\ &\overset{(D.7)}{=} \Omega\left(a_i c_i d_i \cdot \frac{m^3}{\exp(\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i))}\right).\end{aligned}$$

Let $\alpha = \exp\Big(\mathsf{cut}_{G_S}(\hat{V}_i, \hat{U}_i)\Big)$, the asymptotic cost is then lower bounded by

$$\Omega\left(a_i b_i c_i d_i m^2 + a_i b_i d_i m^2 \cdot \alpha + a_i c_i d_i m^3 \cdot \frac{1}{\alpha}\right) = \Omega\Big(a_i b_i c_i d_i m^2 + a_i m^2 d_i \sqrt{b_i c_i m}\Big).$$

This finishes the proof.

□

Proof of Theorem D.1. Based on Lemma D.6, the cost of $\Omega(y_i)$ is needed to sketch the contraction (U_i, V_i). Since T_0 contains contractions (U_i, V_i) for $i \in [N-1]$, the asymptotic cost of T_B must be lower bounded by $\Omega\Big(\sum_{i=1}^{N-1} y_i\Big)$. In addition, in the analysis of Lemma D.6, at least one embedding vertex is needed to sketch each contraction (U_i, V_i), thus $N_E = \Omega(N)$ and $m = \Omega(N \log(1/\delta)/\epsilon^2)$ for the lower bound $\Omega\Big(\sum_{i=1}^{N-1} y_i\Big)$ to hold.

In addition, each $v_j \in V_D$ for $j \in [N]$ is adjacent to e_j and each $w(e_j) > \log(m)$. Based on Lemma D.4, the asymptotic cost must be lower bounded by

$$\Omega\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_R(v_j) + \mathsf{cut}_{G_S}(v_j)) \cdot m\right) = \Omega\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_G(v_j)) \cdot m\right).$$

The above holds since v_j is a vertex in the data graph, thus $\mathsf{cut}_{G_S}(v_j) = \mathsf{cut}_L(v_j)$ and $\mathsf{cut}_R(v_j) + \mathsf{cut}_{G_S}(v_j) = \mathsf{cut}_G(v_j)$. This finishes the proof. □

D.2 Sketching general data

In this section, we look at general tensor network data G_D, where each vertex in G_D can either be adjacent to an edge to be sketched with weight greater than $\log(m)$ or not adjacent to any edge to be sketched. Below we consider any data contraction tree T_0 containing $\mathcal{D}(e_1), \ldots, \mathcal{D}(e_N), \mathcal{S}, \mathcal{I}$ defined in Section 4. We also let $X(e_j) \subset V$ represent the sub network contracted by $\mathcal{D}(e_j)$. We present the asymptotic sketching lower bound in Theorem D.9.

Lemma D.7. *Consider G_D with a data contraction tree T_0 containing $\mathcal{D}(e_j)$, which is a set containing contractions such that e_j is the only data edge adjacent to the contraction output and in $\bar{E}$ (set of data edges to be sketched). For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension and any contraction tree T_B of (G_D, G_E) constrained on the data contraction tree T_0, the sketching asymptotic cost must be lower bounded by $\Omega(t(e_j))$, where $t(e_j)$ is the optimal asymptotic cost to sketch the sub tensor network $X(e_j)$ (defined in Table 4) with an adjacent matrix in the Kronecker product embedding.*

Proof. When $\mathcal{D}(e_j) = \emptyset$, $X(e_j) = \{v_j\}$, where v_j is the vertex in the data graph adjacent to e_j. As is analyzed in the proof of Theorem D.1, the asymptotic cost must be lower bounded by

$\Omega(\exp(\mathsf{cut}_G(v_j)) \cdot m)$, which equals the asymptotic cost to contract v_j with the adjacent embedding matrix.

Now we discuss the case where $\mathcal{D}(e_j) \neq \emptyset$. We first consider the case where there is a contraction $(X(e_j), W)$ in T_B. We show that under this case, the cost is lower bounded by $\Omega(t(e_j))$. We then show that for the case where there is no contraction $(X(e_j), W)$ in T_B, meaning that some sub network of $X(e_j)$ is sketched, the cost is also lower bounded by $\Omega(t(e_j))$. Summarizing both cases prove the lemma.

Consider the case where there exists a contraction $(X(e_j), W)$ in T_B. Contracting $X(e_j)$ yields a cost of $\Omega(\sum_{i\in\mathcal{D}(e_j)} a_i b_i c_i d_i \cdot s_j)$. Next we analyze the contraction cost of $(X(e_j), W)$. Since $X(e_j)$ is the contraction output of $\mathcal{D}(e_j)$, W must either contain embedding vertices, or contain some data vertex adjacent to edges in $\bar{E}$ (edges to be sketched). Therefore, W contains some vertex v with $\mathsf{cut}_{G_S}(v) \geq \log(m)$. Based on Lemma B.3, we have $\mathsf{cut}_{G_S}(W) \geq \log(m)$. Therefore,

$$\begin{aligned}\mathsf{cost}_L(X(e_j), W) &= \mathsf{cut}_{G_S}(X(e_j)) + \mathsf{cut}_{G_S}(W) + \mathsf{cut}_{G_S}(V \setminus (X(e_j) \cup W), X(e_j) \cup W) \\ &\geq \log(s_j) + \log(m).\end{aligned}$$

Let $l \in \mathcal{D}(e_j)$ denote the last contraction in $\mathcal{D}(e_j)$, then we have $\mathsf{cut}_R(X(e_j)) = \log(a_l c_l d_l)$. Thus, we have

$$\mathsf{cost}_G(X(e_j), W) = \mathsf{cost}_L(X(e_j), W) + \mathsf{cost}_R(X(e_j), W) \tag{D.8}$$

$$\geq \mathsf{cost}_L(X(e_j), W) + \mathsf{cut}_R(X(e_j)) \geq \log(a_l c_l d_l s_j m), \tag{D.9}$$

making the cost lower bounded by $\Omega\Big(\sum_{i\in\mathcal{D}(e_j)} a_i b_i c_i d_i \cdot s_j + a_l c_l d_l s_j m\Big)$. Note that contracting $X(e_j)$ and sketching the contraction output with a matrix in the Kronecker product embedding yields a cost of $\Theta\Big(\sum_{i\in\mathcal{D}(e_j)} a_i b_i c_i d_i \cdot s_j + a_l c_l d_l s_j m\Big)$, which upper-bounds the value of $t(e_j)$ based on definition. Thus the sketching cost is lower bounded by $\Omega(t(e_j))$.

Below we analyze the case where there is no contraction $(X(e_j), W)$ in T_B. Without loss of generality, for each contraction (U_i, V_i) with $i \in \mathcal{D}(e_j)$, assume that U_i is adjacent to e_j. When $X(e_j)$ is not formed in T_B, there must exist U_k with $k \in \mathcal{D}(e_j)$, and a contraction (U_k, X) with $X \subset V_E$ is in T_B. All contractions before k yield a cost of

$$\Omega\left(\sum_{i\in\mathcal{D}(e_j), i<k} a_i b_i c_i d_i \cdot s_j\right). \tag{D.10}$$

Similar to the analysis for the contraction $(X(e_j), W)$ in (D.8), the contraction (U_k, X) yields a cost of

$$\Omega(a_k b_k d_k s_j m). \tag{D.11}$$

For other contractions in T_0, (U_i, V_i) with $i \in \mathcal{D}(e_j), i \geq k$, there must exist some contractions $(\hat{U}_i, \hat{V}_i)$ in T_B with $\hat{U}_i \cap V_D = U_i$, $\hat{V}_i \cap V_D = V_i$, since T_B is constrained on T_0. Therefore, we have

$$\begin{aligned}\mathsf{cost}_G(\hat{U}_i, \hat{V}_i) &= \mathsf{cost}_R(\hat{U}_i, \hat{V}_i) + \mathsf{cost}_L(\hat{U}_i, \hat{V}_i) = \mathsf{cost}_R(U_i, V_i) + \mathsf{cost}_L(\hat{U}_i, \hat{V}_i) \\ &\geq \mathsf{cost}_R(U_i, V_i) + \mathsf{cut}_{G_S}(\hat{U}_i) \geq \log(a_i b_i c_i d_i m).\end{aligned} \tag{D.12}$$

In the last inequality in (D.12) we use the fact that there exists a vertex $v \in U_i \subseteq \hat{U}_i$ with $\mathsf{cut}_{G_S}(v) = \log(s_j) \geq \log(m)$, then based on Lemma B.3, $\mathsf{cut}_{G_S}(\hat{U}_i) \geq \log(m)$.

Combining (D.10), (D.11) and (D.12), we have the cost is lower bounded by

$$\Omega(f(k)) = \Omega\left(\sum_{i\in\mathcal{D}(e_j), i<k} a_i b_i c_i d_i \cdot s_j + a_k b_k d_k s_j m + \sum_{i\in\mathcal{D}(e_j), i\geq k} a_i b_i c_i d_i \cdot m\right),$$

where $f(k)$ represents the asymptotic cost to contract $X(e_j)$ with an embedding matrix adjacent to e_j, when sketching is performed at kth contraction with $k \in \mathcal{D}(e_j)$. Based on the definition of $t(e_j)$, we have $f(k) = \Omega(t(e_j))$, thus finishing the proof. □

Lemma D.8. *Consider any data G_D. For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition with only one output sketch dimension and any contraction tree T_B of (G_D, G_E), the sketching asymptotic cost must be lower bounded by $\Omega(Nm^{2.5})$, where $m = \Omega(N \log(1/\delta)/\epsilon^2)$.*

Proof. Let G'_D be the data with the same set of sketching edges ($\bar{E}$) as G_D, but G'_D is a Kronecker product data. For any given contraction tree T_B of (G_D, G_E), there must exist a contraction tree of (G'_D, G_E) whose asymptotic cost is upper bounded by the cost of T_B. Therefore, the asymptotic cost lower bound to contract (G'_D, G_E) must also be the asymptotic cost lower bound to contract (G_D, G_E). Based on Corollary D.2, the asymptotic cost of T_B must be lower bounded by

$$\Omega\left(\sum_{j=1}^{N} s_j m + Nm^{2.5}\right) = \Omega(Nm^{2.5}).$$

□

Theorem D.9. *For any embedding G_E satisfying the (ϵ, δ)-accurate sufficient condition and any contraction tree T_B of (G_D, G_E) constrained on the data contraction tree T_0 expressed in (4.2), the sketching asymptotic cost must be lower bounded by*

$$\Omega\left(\sum_{j=1}^{N} t(e_j) + \sum_{i\in\mathcal{S}} a_i b_i c_i d_i m^2 + Nm^{2.5} + \sum_{i\in\mathcal{I}} z_i\right), \tag{D.13}$$

where $m = \Omega(N \log(1/\delta)/\epsilon^2)$, a_i, b_i, c_i, d_i are expressed in (C.2), $t(e_j)$ is the optimal asymptotic cost to sketch the sub tensor network $X(e_j)$ (the sub network contracted by $\mathcal{D}(e_j)$, also defined in Table 4) with an adjacent matrix in the Kronecker product embedding, and z_i is expressed in (C.5).

Proof. The term $\sum_{j=1}^{N} t(e_j)$ can be proven based on Lemma D.7, and the term $Nm^{2.5}$ with $m = \Omega(N \log(1/\delta)/\epsilon^2)$ can be proven based on Lemma D.8. Below we show the asymptotic cost is also lower bounded by $\Omega(\sum_{i\in\mathcal{S}} a_i b_i c_i d_i m^2 + \sum_{i\in\mathcal{I}} z_i)$, thus finishing the proof.

For each contraction (U_i, V_i) in T_0 with $i \in \mathcal{S} \cup \mathcal{I}$, there must exist a contraction $(\hat{U}_i, \hat{V}_i)$ in T_B, and $\hat{U}_i \cap V_D = U_i$, $\hat{V}_i \cap V_D = V_i$. For the case where $i \in \mathcal{S}$, since both U_i and V_i contain edges to be sketched, we have $\mathsf{cut}_{G_S}(\hat{U}_i) \geq \log(m)$ and $\mathsf{cut}_{G_S}(\hat{V}_i) \geq \log(m)$ based on Lemma B.3. Therefore, we have

$$\begin{aligned}
\sum_{i\in\mathcal{S}} \mathsf{cost}_G(\hat{U}_i, \hat{V}_i) &= \sum_{i\in\mathcal{S}} \mathsf{cost}_R(\hat{U}_i, \hat{V}_i) + \mathsf{cost}_L(\hat{U}_i, \hat{V}_i) \\
&= \sum_{i\in\mathcal{S}} \mathsf{cost}_R(U_i, V_i) + \mathsf{cost}_L(\hat{U}_i, \hat{V}_i) \\
&\geq \sum_{i\in\mathcal{S}} \mathsf{cost}_R(U_i, V_i) + \mathsf{cut}_{G_S}(\hat{U}_i) + \mathsf{cut}_{G_S}(\hat{V}_i) \\
&\geq \sum_{i\in\mathcal{S}} \log(a_i b_i c_i d_i m^2),
\end{aligned} \tag{D.14}$$

where the first inequality above holds based on Lemma B.2. This shows the cost is lower bounded by $\Omega(\sum_{i\in\mathcal{S}} a_i b_i c_i d_i m^2)$.

Now consider the case where $i \in \mathcal{I}$. In this case, either $\mathsf{cut}_{G_S}(U_i \cup V_i) = 0$ or $\mathsf{cut}_{G_S}(U_i \cup V_i) \geq \log(m)$. When $\mathsf{cut}_{G_S}(U_i \cup V_i) = 0$, we have $\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) \geq \mathsf{cut}_{G_S}(U_i \cup V_i)$. When $\mathsf{cut}_{G_S}(U_i \cup V_i) \geq \log(m)$, based on Lemma B.3, we have $\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) \geq \log(m)$. To summarize, we have

$$\mathsf{cut}_{G_S}(\hat{U}_i \cup \hat{V}_i) \geq \min(\mathsf{cut}_{G_S}(U_i \cup V_i), \log(m)),$$

thus

$$\begin{aligned}\sum_{i\in\mathcal{I}} \mathsf{cost}_G(\hat{U}_i,\hat{V}_i) &= \sum_{i\in\mathcal{I}} \mathsf{cost}_R(\hat{U}_i,\hat{V}_i) + \mathsf{cost}_L(\hat{U}_i,\hat{V}_i)\\ &\geq \sum_{i\in\mathcal{I}} \mathsf{cost}_R(U_i,V_i) + \mathsf{cut}_{G_S}(\hat{U}_i) + \mathsf{cut}_{G_S}(\hat{V}_i)\\ &\geq \sum_{i\in\mathcal{I}} \mathsf{cost}_R(U_i,V_i) + \mathsf{cut}_{G_S}(\hat{U}_i\cup\hat{V}_i)\\ &\geq \sum_{i\in\mathcal{I}} \log(a_ib_ic_id_i) + \min(\mathsf{cut}_{G_S}(U_i\cup V_i),\log(m))\\ &= \sum_{i\in\mathcal{I}} \log(z_i). \end{aligned} \tag{D.15}$$

This shows the sketching cost is lower bounded by $\Omega\big(\sum_{i\in\mathcal{I}} z_i\big)$, thus finishing the proof. □

Proof of Theorem 4.2. Based on Theorem C.1, the computational cost of Algorithm 1 is

$$\alpha = \Theta\left(\sum_{j=1}^{N} t(e_j) + \sum_{i\in\mathcal{S}} y_i + \sum_{i\in\mathcal{I}} z_i\right).$$

Let β equals the expression in (D.13). We have

$$\begin{aligned}\frac{\alpha}{\beta} &= \frac{\Theta\Big(\sum_{j=1}^{N} t(e_j) + \sum_{i\in\mathcal{S}} y_i + \sum_{i\in\mathcal{I}} z_i\Big)}{\Omega\Big(\sum_{j=1}^{N} t(e_j) + \sum_{i\in\mathcal{S}}(a_ib_ic_id_im^2 + m^{2.5}) + \sum_{i\in\mathcal{I}} z_i\Big)} = O\left(\frac{\sum_{i\in\mathcal{S}} y_i}{\sum_{i\in\mathcal{S}}(a_ib_ic_id_im^2+m^{2.5})}\right)\\ &= O\left(\max_{i\in\mathcal{S}} \frac{y_i}{a_ib_ic_id_im^2+m^{2.5}}\right) = O\left(\max_{i\in\mathcal{S}} \frac{a_ib_ic_id_im^2 + m^2d_i\sqrt{a_ib_ic_im}\cdot\min(\sqrt{a_i},\sqrt{c_i})}{a_ib_ic_id_im^2+m^{2.5}}\right)\\ &= O(1) + O\left(\max_{i\in\mathcal{S}} \frac{m^2d_i\sqrt{a_ib_ic_im}\cdot\min(\sqrt{a_i},\sqrt{c_i})}{a_ib_ic_id_im^2+m^{2.5}}\right).\end{aligned}$$

Below we derive asymptotic upper bound of the term $\theta = \frac{m^2d_i\sqrt{a_ib_ic_im}\cdot\min(\sqrt{a_i},\sqrt{c_i})}{a_ib_ic_id_im^2+m^{2.5}}$. We analyze the case below with $a_i \leq c_i$, and the other case with $a_i > c_i$ can be analyzed in a similar way based on the symmetry of a_i, c_i in θ.

When $a_i \leq c_i$, we have $m^2d_i\sqrt{a_ib_ic_im}\cdot\min(\sqrt{a_i},\sqrt{c_i}) = a_im^2d_i\sqrt{b_ic_im}$. We consider two cases, one satisfies $\sqrt{b_ic_im} \leq b_ic_i$ and the other satisfies $\sqrt{b_ic_im} > b_ic_i$.

When $\sqrt{b_ic_im} \leq b_ic_i$, we have $\theta \leq 1$, thus $\frac{\alpha}{\beta} = O(1)$, thus satisfying the theorem statement.

When $\sqrt{b_ic_im} > b_ic_i$, which means that $m > b_ic_i$, we have

$$\theta = \frac{a_im^2d_i\sqrt{b_ic_im}}{a_ib_ic_id_im^2+m^{2.5}} \leq \min\left(\frac{\sqrt{m}}{\sqrt{b_ic_i}}, a_id_i\sqrt{b_ic_i}\right) \leq \sqrt{m},$$

thus $\frac{\alpha}{\beta} \leq O(\sqrt{m})$. In addition, when G_D is a graph, we have $d_i = 1$ for all i. Therefore,

$$\begin{aligned}\theta &\leq \min\left(\frac{\sqrt{m}}{\sqrt{b_ic_i}}, a_id_i\sqrt{b_ic_i}\right) = \min\left(\frac{\sqrt{m}}{\sqrt{b_ic_i}}, a_i\sqrt{b_ic_i}\right)\\ &\leq \min\left(\frac{\sqrt{m}}{\sqrt{b_ic_i}}, c_i\sqrt{b_ic_i}\right) \leq \min\left(\frac{\sqrt{m}}{(b_ic_i)^{1/2}}, (b_ic_i)^{3/2}\right) \leq m^{0.375}.\end{aligned}$$

Therefore, in this case we have $\frac{\alpha}{\beta} \leq O\big(m^{0.375}\big)$, thus finishing the proof. □

E Analysis of tree tensor network embeddings

In this section, we provide detailed analysis of sketching with tree embeddings. The algorithm to sketch with tree embedding is similar to Algorithm 1, and the only difference is that for each

contraction (U_i, V_i) with $i \in \mathcal{S}$, such that both U_i and V_i are adjacent to edges in $\bar{E}$, we sketch it with one embedding tensor z_i rather than a small network. Let $\hat{U}_i, \hat{V}_i$ denote the sketched U_i and V_i formed in previous contractions in the sketching contraction tree T_B, such that $\hat{U}_i \cap V_D = U_i$ and $\hat{V}_i \cap V_D = V_i$, we sketch $(\hat{U}_i, \hat{V}_i)$ via the contraction path $((\hat{U}_i, \hat{V}_i), z_i)$. For the case where each vertex in the data tensor network is adjacent to an edge to be sketched, the sketching cost would be

$$\Theta\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_G(v_j)) \cdot m + \sum_{j=1}^{N-1} (a_i b_i c_i d_i m^2 + a_i c_i d_i m^3)\right), \tag{E.1}$$

where v_j is the vertex in the data graph adjacent to e_j, a_i, b_i, c_i, d_i are defined in (C.2), and we replace the term $y_i = a_i b_i c_i d_i m^2 + m^2 d_i \sqrt{a_i b_i c_i m} \cdot \min(\sqrt{a_i}, \sqrt{c_i})$ in (C.6) with $a_i b_i c_i d_i m^2 + a_i c_i d_i m^3$.

Proof of Theorem 4.3. Since each contraction in T_0 contracts dimensions with size being at least the sketch size m, we have $b_i \geq m$ for $i \in \mathcal{S}$. Therefore,

$$m^2 d_i \sqrt{a_i b_i c_i m} \cdot \min(\sqrt{a_i}, \sqrt{c_i}) \leq a_i m^2 d_i \sqrt{b_i c_i m} \leq a_i b_i d_i m^2 \sqrt{c_i} \leq a_i b_i c_i d_i m^2,$$

and the asymptotic cost in (C.6) would be

$$\Theta\left(\sum_{j=1}^{N} \exp(\mathsf{cut}_G(v_j)) \cdot m + \sum_{j=1}^{N-1} a_i b_i c_i d_i m^2\right). \tag{E.2}$$

Based on Theorem D.1, (E.2) matches the sketching asymptotic cost lower bound for this data. Since $a_i c_i d_i m^3 \leq a_i b_i c_i d_i m^2$ so (E.1) equals (E.2), sketching with tree embeddings also yield the optimal asymptotic cost. □

When the data has a Kronecker product structure, sketching with tree tensor network embedding is less efficient compared to Algorithm 1. As is shown in (E.3), Algorithm 1 yields a cost of $\Theta\left(\sum_{j=1}^{N} s_j m + N m^{2.5}\right)$ to sketch the Kronecker product data. However, for tree embeddings, the asymptotic cost (E.1) is equal to

$$\Theta\left(\sum_{j=1}^{N} s_j m + N m^3\right). \tag{E.3}$$

F Computational cost analysis of sketched CP-ALS

In this section, we provide detailed computational cost analysis of the sketched CP-ALS algorithm based on Algorithm 1. We are given a tensor $\mathcal{X} \in \mathbb{R}^{s \times \cdots \times s}$, and aim to decompose that into N factor matrices, $A_i \in \mathbb{R}^{s \times R}$ for $i \in [N]$. Let $L_i = A_1 \odot \cdots \odot A_{i-1} \odot A_{i+1} \odot \cdots \odot A_N$ and $R_i = X_{(i)}^T$. In each iteration, we aim to update A_i via solving a sketched linear least squares problem, $A_i = \underset{A}{\operatorname{argmin}} \left\| S_i L_i A^T - S_i R_i \right\|_F^2$, where S_i is an embedding constructed based on Algorithm 1.

Below we first discuss the sketch size of S_i sufficient to make each sketched least squares problem accurate. We then discuss the contraction trees of L_i, on top of which embedding structures are determined. We select contraction trees such that contraction intermediates can be reused across subproblems. Finally, we present the detailed computational cost analysis of the sketched CP-ALS algorithm.

F.1 Sketch size sufficient for accurate least squares subproblem

Since the tensor network of L_i contains N output dimensions and L_i contains R columns, we show below that a sketch size of $\Theta(NR\log(1/\delta)/\epsilon^2) = \tilde{\Theta}(NR/\epsilon^2)$ is sufficient for the least squares problem to be (ϵ, δ)-accurate.

Theorem F.1. *Consider the sketched linear least squares problem* $\min_A \left\| S_i L_i A^T - S_i R_i \right\|_F^2$. *Let* S_i *be an embedding constructed based on Algorithm 1, with the sketch size* $m = \Theta(NR\log(1/\delta)/\epsilon^2)$, *solving the sketched least squares problem gives us an* $(1+\epsilon)$*-accurate solution with probability at least* $1-\delta$.

Proof. Algorithm 1 outputs an embedding with $\Theta(N)$ vertices. Based on Theorem 3.1, a sketched size of $\Theta(NR\log(1/\delta)/\epsilon^2)$ will make the embedding $(\epsilon, \frac{\delta}{e^R})$-accurate. Based on the ϵ-net argument [50], S is the (ϵ, δ)-accurate subspace embedding for a subspace with dimension R. Therefore, we can get an $(1+\epsilon)$-accurate solution with probability at least $1-\delta$ for the least squares problem. □

F.2 Data contraction trees and efficient embedding structures

The structures of embeddings $S_1, \ldots, S_N$ also depend on the data contraction trees for $L_1, \ldots, L_N$. We denote the contraction tree of L_i as T_i. We construct T_i for $i \in [N]$ such that resulting embeddings $S_1, \ldots, S_N$ have common parts, which yields more efficient sketching computational cost via reusing contraction intermediates.

Let the vertex v_i represent the matrix A_i. We also let $V_L^{(i)} = \{v_1, \ldots, v_i\}$ denote the set of all first i vertices, and let $V_R^{(i)} = \{v_i, \ldots, v_N\}$ denote the set of vertices from v_i to v_N. In addition, we let $\mathcal{C}_L^{(i)}$ denote a contraction tree to fully contract $V_L^{(i)}$, from v_1 to v_i. Let $\mathcal{C}_L^{(1)} = \emptyset$, we have for all $i \geq 1$, $\mathcal{C}_L^{(i+1)} = \mathcal{C}_L^{(i)} \cup \left\{ (V_L^{(i)}, v_{i+1}) \right\}$. Similarly, we let $\mathcal{C}_R^{(i)}$ denote a contraction tree to fully contract $V_R^{(i)}$, from v_N to v_i. Let $\mathcal{C}_R^{(N)} = \emptyset$, we have for all $i \leq N$, $\mathcal{C}_R^{(i-1)} = \mathcal{C}_R^{(i)} \cup \left\{ (V_R^{(i)}, v_{i-1}) \right\}$.

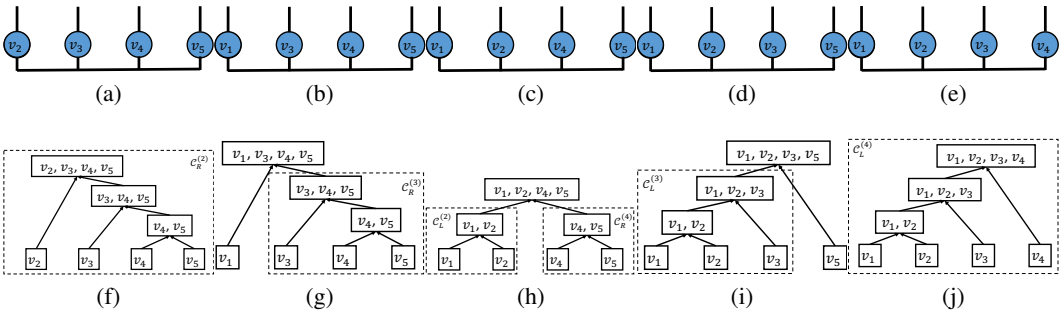


Figure 12: (a)-(e): Representations of $L_1, \ldots, L_5$ for the CP decomposition of an order 5 tensor. (f)-(j): Data dimension trees $T_1, \ldots, T_5$.

Note that the vertex set of the tensor network of L_i is $V_L^{(i-1)} \cup V_R^{(i+1)}$. Each T_i is constructed so that $V_L^{(i-1)}, V_R^{(i+1)}$ are first contracted via the contraction trees $\mathcal{C}_L^{(i-1)}, \mathcal{C}_R^{(i+1)}$, respectively, then a contraction of $(V_L^{(i-1)}, V_R^{(i+1)})$ is used to contract them into a single tensor. We illustrate T_i for the CP decomposition of an order 5 tensor in Fig. 12.

These tree structures allow us to reuse contraction intermediates during sketching. On top of T_1, sketching L_1 using Algorithm 1 yields a cost of $\Theta(N(smR + m^{2.5}R))$, where the term $\Theta(NsmR)$ comes from sketching with the Kronecker product embedding, and the term $\Theta(Nm^{2.5}R)$ comes from sketching each data contraction in $\mathcal{C}_R^{(2)}$. Since $\mathcal{C}_R^{(2)} = \mathcal{C}_R^{(3)} \cup \left\{ (V_R^{(3)}, v_2) \right\}$, all contractions in $\mathcal{C}_R^{(3)}$ are sketched, and we obtain the sketching output of $V_R^{(3)}$, which is denoted as $\hat{V}_R^{(3)}$ below.

We use $\hat{V}_R^{(3)}$ formed during sketching L_1 to sketch L_2. Since T_2 contains contractions

$$T_2 = \mathcal{C}_R^{(3)} \cup \mathcal{C}_L^{(1)} \cup \left\{ (V_L^{(1)}, V_R^{(3)}) \right\} = \mathcal{C}_R^{(3)} \cup \left\{ (v_1, V_R^{(3)}) \right\},$$

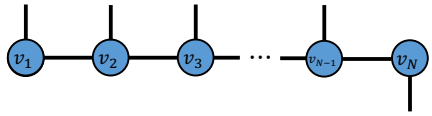


Figure 13: Illustration of the matricization of the tensor train (X) to be sketched. The $N-1$ uncontracted edges adjacent to $v_1, \ldots, v_{N-1}$ are to be sketched.

through reusing $\hat{V}_R^{(3)}$, we only need to sketch $(v_1, \hat{V}_R^{(3)})$ to compute $S_2 L_2$, which only costs $\Theta(smR + m^{2.5}R)$. Similarly, sketching each L_i for $i \geq 2$ only costs $\Theta(smR + m^{2.5}R)$, thus making the overall cost of sketching $L_1, \ldots, L_N$ being $\Theta(N(smR + m^{2.5}R))$.

F.3 Detailed algorithm and the overall computational cost

Algorithm 2 Sketched-ALS: Sketched ALS for CP decomposition

1: **Input:** Input tensor $\mathcal{X}$, initializations $A_1, \ldots, A_N$, maximum number of iterations $I_{\max}$
2: $G_D(L_i) \leftarrow$ structure of the data $L_i = A_1 \odot \cdots \odot A_{i-1} \odot A_{i+1} \odot \cdots \odot A_N$ for $i \in [N]$
3: $T_i \leftarrow$ contraction tree of $G_D(L_i)$ for $i \in [N]$ constructed based on Appendix F.2
4: Build tensor network embeddings S_i on $G_D(L_i)$ and T_i based on Algorithm 1 for $i \in [N]$
5: Compute $\hat{R}_i \leftarrow S_i X_{(i)}^T$ for $i \in [N]$
6: **for** $t \in [I_{\max}]$ **do**
7: **for** $i \in [N]$ **do**
8: Compute $\hat{L}_i \leftarrow S_i L_i$
9: $A_i \leftarrow \underset{X}{\operatorname{argmin}} \left\| \hat{L}_i X - \hat{R}_i \right\|_F^2$
10: **end for**
11: **end for**
12: **return** $A_1, \ldots, A_N$

We present the detailed sketched CP-ALS algorithm in Algorithm 2. Here we analyze the overall computational cost of the algorithm.

Line 5 yields a preparation cost of the algorithm. Note that we construct S_i based on Appendix F.2, where they share common tensors. Contracting $S_1 X_{(1)}^T$ yields a cost of $\Theta(s^N m)$. On top of that, contracting $S_i X_{(i)}^T$ for $i \geq 2$ also only yields a cost of $\Theta(s^N m)$, making the overall preparation cost $\Theta(s^N m)$.

Within each ALS iteration (Lines 7-10), based on Appendix F.2, computing $S_i L_i$ for $i \in [N]$ costs $\Theta(N(smR + m^{2.5}R))$. For each $i \in [N]$, line 9 costs $\Theta(mR^2)$, making the cost of per-iteration least squares solves $\Theta(NmR^2)$. Based on Appendix F.1, a sketch size of $m = \tilde{\Theta}(NR/\epsilon^2)$ is sufficient for the least squares solution to be $(1+\epsilon)$-accurate with probability at least $1-\delta$. Overall, the per-iteration cost is $\Theta(N(smR + m^{2.5}R)) = \tilde{\Theta}(N^2(N^{1.5}R^{3.5}/\epsilon^3 + sR^2)/\epsilon^2)$.

G Computational cost analysis of sketching for tensor train rounding

We provide the computational cost lower bound analysis of computing SX, where X denotes a matricization of the tensor train data shown in Fig. 13. This step is the computational bottleneck of the tensor train randomized rounding algorithm proposed in [10]. As is discussed in Section 5, we assume the tensor train has order N with the output dimension sizes equal s, the tensor train rank is $R < s$, and the goal is to round the rank to $r < R$. The sketch size m of S is r plus some constant, and is assumed to be smaller than R. The lower bound is derived within all embeddings satisfying the sufficient condition in Theorem 3.1 and only have one output sketch dimension with size m.

For the data contraction tree that contracts the tensor train shown in Fig. 13 from left to right, we have $a_i = 1, b_i = R, c_i = R, d_i = 1$ for $i \in [N-2]$, where a_i, b_i, c_i, d_i are expressed in (C.2).

Based on Theorem D.9, the sketching asymptotic cost lower bound is

$$\Omega\left(\sum_{j=1}^{N-1} t(e_j) + \sum_{i\in\mathcal{S}} a_i b_i c_i d_i m^2 + Nm^{2.5} + \sum_{i\in\mathcal{I}} z_i\right) = \Omega\left(\sum_{j=1}^{N-1} t(e_j) + \sum_{i\in\mathcal{S}} a_i b_i c_i d_i m^2\right)$$
$$= \Omega\left(\sum_{j=1}^{N-1} \exp(\mathsf{cut}_G(v_j)) \cdot m + \sum_{j=1}^{N-2} a_i b_i c_i d_i m^2\right)$$
$$= \Omega\big(NsR^2m + NR^2m^2\big) = \Omega\big(NsR^2m\big).$$

Above we use the fact that $\exp(\mathsf{cut}_G(v_1)) = sR$, and for $j \in \{2, \ldots, N-1\}$, we have $\exp(\mathsf{cut}_G(v_j)) = sR^2$. Sketching with Algorithm 1, tree embedding and tensor train embedding all would yield this optimal asymptotic cost.

H Additional experiments

	Uniform	Gaussian
Tensor network embedding (Algorithm 1)	85.0	78.1
Tree embedding (Theorem 4.3)	75.4	68.1
Tensor train embedding [10]	49.3	45.4

Table 5: Comparison of the mean sketch sizes with different input data distribution when sketching a tensor train input. The tensor train order is chosen to be 6 and the dimension size is chosen to be 500. Each reported sketch size is the mean of 25 experiments. Variables in the uniform distribution are within the interval of $[0, 1]$, and variables in the Gaussian distribution have the same mean as the uniform distribution and have the unit variance.

In this section, we experimentally verify that the sketch size of embeddings to get the same sketching accuracy trends similarly for both uniform and Gaussian input tensor distributions. We compare the performance of general tensor network embedding used in Algorithm 1, tree embedding discussed in Theorem 4.3, and the baseline, tensor train embedding [10], in sketching tensor train input data in Table 5. For each input tensor train x and a specific embedding structure, we calculate the relative sketching error twice under different sketch sizes, and record the smallest sketch size such that both of its relative sketching errors are within 0.2, $\frac{\|Sx\|_2}{\|x\|_2} \leq 0.2$. As can be seen in the table, for both distributions, tensor network embedding produces a slightly larger sketch size than tree embedding, and tensor train embedding yields the lowest sketch size.