Hardware-Efficient Mixed-Precision CP Tensor Decomposition [*]

Zi Yang, Junnan Shan, and Zheng Zhang

Abstract. Tensor decomposition has been widely used in machine learning and high-volume data analysis. However, large-scale tensor factorization often consumes huge memory and computing cost. Meanwhile, modernized computing hardware such as tensor processing units (TPU) and Tensor Core GPU has opened a new window of hardware-efficient computing via mixed- or low-precision arithmetic representations. In this paper, we exploit the low-precision representation of tensor factorization, and propose a mixed-precision block stochastic gradient descent (SGD) method to reduce the costs of CP tensor decomposition. Our method achieves robust and fast convergence via a two-stage optimization, i.e., SignSGD followed by mixed-precision SGD. Detailed theoretical analysis is provided to prove the convergence of the proposed mixed-precision algorithm. Numerical experiments on both synthetic and realistic tensor data sets show the superior efficiency of our mixed-precision algorithm compared to full-precision CP decomposition. This work can remarkably reduce the memory, computing and energy cost on resource-constraint edge computing devices. We demonstrate this benefit via an FPGA prototype.

Key words. Tensor decomposition, mixed-precision optimization, stochastic gradients, hardware-aware algorithms.

MSC codes. 15A69, 90C15

1. Introduction.

As a higher-order generalization of matrices, tensors [35] have been used to represent and process multi-dimensional arrays in many science and engineering fields, including quantum physics [20, 30, 40, 45], scientific computing [6, 47], uncertainty quantification [18, 55, 56], machine learning [2, 23, 25, 26, 41, 42, 33, 49] and many others. Many successful applications rely on efficient tensor decompositions [8, 15, 22, 46], which represent an original high-order high-volume data array with some low-rank factors to achieve huge memory and computing cost reduction. For instance, tensor decomposition has achieved orders-of-magnitude parameter reduction of deep neural networks [25, 26, 33, 42], enabling their energy-efficient training and deployment on edge devices. As one of the most popular tensor decomposition methods, the CANDECOMP/PARAFAC (CP) decomposition [8] factorizes a large tensor into the summation of some rank-1 tensors. A CP factorization is often obtained via algebraic methods [19, 39] or numerical optimization techniques such as gradient-based optimization [21] and alternating minimization [13]. The former provides excellent theoretical guarantees, but are neither noise-resistant nor scalable to high tensor ranks. The later has better efficiency, but computing the full gradients is expensive for high-volume tensor data sets. Motivated by the success in large-scale machine learning, recent approaches use stochastic gradient descent (SGD) methods [3, 5, 36, 51] to relief the high computation cost in tensor factorization. So far, most (if not all) tensor decomposition algorithms are developed for classical computing platforms (e.g., CPU and conventional GPU) that use double-precision 64-bit or single-precision 32-bit floating-point data representations.

[*]The authors are with Department of Electrical and Computer Engineering, University of California at Santa Barbara, CA. (Emails: ziy@ucsb.edu, junnanshan@ucsb.edu, zzhang01@ucsb.edu).

On the other hand, the recent revolution of artificial intelligence has triggered massive interests in computing hardware that supports mixed-precision and low-precision computation. For instance, Google's Tensor Processing Units (TPUs) [31] can easily handle machine learning tasks with 16-bit floating point representations. NVIDIA's tensor Core GPU supports double-, single- and half-precision floating-point operations, as well as various low-precision integer operations. Reconfigurable computing platforms such as field-programmable gate arrays (FPGA) can support arbitrarily low-precision computation to save energy and hardware utilization. These mixed-precision computing platforms are very suitable for the training and inference of deep learning models [17, 16, 29, 50, 54], due to their error-resilient activation functions or output operators. Interestingly, recently mixed-precision computing has also shown great success in many scientific computing tasks [1, 9, 10, 11, 12, 24, 44] such as LU factorization, Cholesky factorization, least square optimization, GMRES. However, mixed-precision computing has been rarely investigated for tensor computation. We envision that similar memory and runtime benefit can be obtained by developing mixed-precision tensor computation algorithms. As the development of 5G and future 6G networks, more and more (possibly private and sensitive) data needs to be processed on resource-constraint edge devices, where mixed-precision tensor computation will play an increasingly important role.

In this paper, we make the first step of exploring low-precision tensor computation by proposing a novel mixed-precision CP tensor decomposition algorithm. By utilizing low-precision stochastic gradient computation in a two-stage optimization framework, our method can remarkably reduce the computation and energy costs of CP decomposition. Our main contributions are summarized below.

- We propose a computationally efficient CP decomposition via a mixed-precision SGD method. We improve the convergence via a mixed-precision SignSGD initialization. We carefully design the low-precision stochastic gradient computation to maximize the computational efficiency and minimize the accuracy drop via analyzing the sensitivity of each step with respect to the quantization errors.
- We prove the convergence of the proposed mixed-precision CP decomposition. Under some conditions, we firstly show that the CP decomposition problem is locally strongly convex after proper normalization. Then, we prove that SignSGD with mixed-precision gradients converges to a stationary point up to a noise level. Finally, we prove that the mixed-precision SGD has a locally linear convergence rate for our problem.
- Numerical experiments demonstrate that our mixed-precision approach can remarkably reduce the computation cost while attaining similar accuracy to the full-precision algorithm. An FPGA prototype further demonstrates the saving of run-time, hardware resources, and energy on edge computing devices.

We remark that the proposed mixed-precision stochastic gradient can be applied to all SGD-based algorithms for CP tensor decomposition.

2. Preliminary.

Notation. Throughout the paper, lower-case letters (e.g., a) denote scalars; lower-case bold letters (e.g., $\mathbf{a}$) denote vectors; upper case bold letters (e.g., $\mathbf{A}$) denote matrices. We use $\mathbf{1}$ or $\mathbf{0}$ to denote a vector/matrix whose entries are all 1 or 0, respectively. $\mathbf{I}_n$ is an n-by-n identity matrix. We use upper-case calligraphic bold letters (e.g., $\boldsymbol{\mathcal{A}}$) to denote tensors, which are high-dimensional generalizations of matrices. We use $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$. For a vector $\mathbf{v}$, $\|\mathbf{v}\|$ and $\|\mathbf{v}\|_1$ denote its Euclidean norm and 1-norm, respectively. For a matrix $\mathbf{A}$, $\mathbf{A}^T$ denotes its transpose; $\mathrm{tr}(\mathbf{A})$ denotes the trace of $\mathbf{A}$; $\|\mathbf{A}\|$ represents the Frobenius norm, and the spectrum norm $\|\mathbf{A}\|_2$ is the largest singular value of $\mathbf{A}$. We use MATLAB-style indexing to denote submatrices. For instance, $\mathbf{A}(i_1 : i_2, j_1 : j_2)$ denotes the submatrix consisting of the rows from i_1 to i_2 and the columns from j_1 to j_2. The function $\mathrm{sign}(a)$ obtains the sign of a, i.e.,

$$\mathrm{sign}(a) := \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a = 0 \\ -1 & \text{if } a < 0 \end{cases}.$$

The sign function can be used for matrices and tensors by applying the function element-wisely.

For a twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we use $\nabla f \in \mathbb{R}^n$ and $\nabla^2 f \in \mathbb{R}^{n\times n}$ to denote the gradient and the Hessian matrix of f, respectively. The function f is λ-strongly convex for $\lambda > 0$ if the smallest eigenvalue of $\nabla^2 f$ is not less than λ. Equivalently, f is $\lambda-$strongly convex if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{y}-\mathbf{x}\|^2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

For the vector valued function $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_m(\mathbf{x}))$ where $h_i : \mathbb{R}^n \to \mathbb{R}$, the Jacobian matrix $\mathbf{J_h}(\mathbf{x})$ is

$$\mathbf{J_h}(\mathbf{x}) := \big(\nabla h_1(\mathbf{x}), \ldots, \nabla h_m(\mathbf{x})\big)^T.$$

2.1. Tensors. Tensors can be regarded as multi-dimensional data arrays [37]. The space of real tensors with order m and dimension $N_1, N_2, \ldots, N_m$ is denoted by $\mathbb{R}^{N_1\times N_2\times\cdots\times N_m}$. The $(i_1, i_2, \cdots, i_m)$-th element of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1\times N_2\times\cdots\times N_m}$ is denoted as $a_{i_1,\ldots,i_m}$ for $1 \leq i_j \leq N_j$. The mode-k unfolding of $\boldsymbol{\mathcal{A}}$ is the matrix $\mathbf{A}_{[k]} \in \mathbb{R}^{N_k\times(\prod_{i=1}^{m} N_i)/N_k}$ which is obtained by reshaping $\boldsymbol{\mathcal{A}}$ with the kth dimension being the leading dimension. The Frobenius norm of $\boldsymbol{\mathcal{A}}$ is

$$\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}} := \sqrt{\sum_{i_1,\ldots,i_m}^{N_1,\ldots,N_m} a_{i_1,\ldots,i_m}^2}.$$

For vectors $\{\mathbf{u}_i \in \mathbb{R}^{N_i}\}_{i=1}^m$, their outer product forms an order-m rank-1 tensor

$$\boldsymbol{\mathcal{B}} = \mathbf{u}_1 \circ \mathbf{u}_2 \circ \cdots \circ \mathbf{u}_m \iff b_{i_1,\ldots,i_m} = \prod_{k=1}^{m} \mathbf{u}_k(i_k).$$

A tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1\times N_2\times\cdots\times N_m}$ is said to have a rank-r CP decomposition if there exist matrices $\{\mathbf{U}_i \in \mathbb{R}^{N_i\times r}\}_{i=1}^m$ such that

$$\boldsymbol{\mathcal{A}} = [\![\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_m]\!] := \sum_{j=1}^{r} \mathbf{U}_1(:, j) \circ \cdots \circ \mathbf{U}_m(:, j).$$

Type	Bits	Sign	Exponent	Significand	Min	Max
$\texttt{FP}_{16}$	16	1	5	10	6.1×10^{-5}	6.6×10^{4}
$\texttt{FP}_{32}$	32	1	8	23	1.2×10^{-38}	3.4×10^{38}
$\texttt{FP}_{64}$	64	1	11	52	2.2×10^{-308}	1.8×10^{308}

Table 1

Floating Point Representations

(m,k,n)	$\texttt{INT}_8$ time	$\texttt{FP}_{16}$ time	$\texttt{FP}_{32}$ time
$(240,240^2,256)$	232 (**4.47**×)	675 (1.54×)	1037 (1×)
$(60,60^3,64)$	794 (**4.55**×)	2456 (1.47×)	3615 (1×)
$(24,24^4,32)$	1139 (**4.89**×)	3784 (1.47×)	5571 (1×)

Table 2

Time comparisons of matrix multiplications of $m \times k$ and $k \times n$ under various precisions on GPU. The times are measured in microseconds (μs).

The smallest integer r that ensures the above equality is called the CP rank of $\boldsymbol{\mathcal{A}}$, denoted by $\operatorname{rank}(\boldsymbol{\mathcal{A}})$. The Khatri-Rao product of matrices $\mathbf{U}_1, \ldots, \mathbf{U}_m$ is a column-wise Kronecker product, i.e.,

$$\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_m := \left[\otimes_{i=1}^m \mathbf{U}_i(:,1), \ldots, \otimes_{i=1}^m \mathbf{U}_i(:,r)\right],$$

where $\otimes$ denotes the Kronecker product. It holds that $\mathbf{A}_{[k]} = \mathbf{U}_k(\odot_{i=1,i\neq k}^m \mathbf{U}_i)^T$.

2.2. Precision Reprensentations. In practice, numbers are represented and processed as binary strings on digital computing hardware. The binary strings can represent numbers in either fixed-point format or floating-point format. We use $\texttt{INT}_n$ and $\texttt{FP}_n$ to denote an n-bit fixed-point format and an n-bit floating-point format, respectively. The representation format is directly related to the precision of the represented number. Hence, the representation format is also called precision format.

An $\texttt{INT}_n$ data representation uses n bits, where the first bit stores the sign and the other $n-1$ bits store the absolute value. The set of numbers that the $\texttt{INT}_n$ format can represent is $\{-2^{n-1}, -2^{n-1}+1, \ldots, 0, 1, \ldots, 2^{n-1}-1\}$. The $\texttt{FP}_n$ format uses 1 bit to store the sign, N_1 bits to store significand, and N_2 bits to store exponent, where $N_1 + N_2 + 1 = n$. Then, the number is represented by sign × significand × 2^{exponent}. $\texttt{FP}_{16}$ (half precision), $\texttt{FP}_{32}$ (single precision), and $\texttt{FP}_{64}$ (double precision) are most commonly used and are supported by most devices. Their bits for each part and representation ranges are described in Table 1. Floating-point arithmetic operations are much more expensive than fixed-point arithmetic operations with the same number of bits. Clearly, low-bit representations consume less memory and computation resources but cause larger rounding-off errors. Table 2 compares the run-time of matrix multiplications under different precision formats on tensor core GPU. The chosen test sizes are common in computing gradients for the proposed Algorithm 3.1 as in (3.5). We can see that the $\texttt{INT}_8$ multiplications are 4× to 5× faster than $\texttt{FP}_{32}$ multiplications.

Deterministic rounding and stochastic rounding methods can be used to round a high-precision number to a lower precision. For a given precision format p, let $\mathcal{R}(p)$ be the set of numbers that can be represented by the format p. The ceiling and floor functions with precision p are defined as

$$\lceil y \rceil_p := \min\{v \in \mathcal{R}(p) \cup \{+\infty\} | v \geq y\},$$
$$\lfloor y \rfloor_p := \max\{v \in \mathcal{R}(p) \cup \{-\infty\} | v \leq y\}.$$

When the precision p is not specified, we use $\mathcal{R}(p) = \mathbb{N}$ by default. The quantization function $\mathtt{Q}^D_{p,\delta}$, with precision p, scaling factor δ, and deterministic rounding, is defined as

$$\mathtt{Q}^D_{p,\delta}(x) = \begin{cases} \delta\lceil x/\delta \rceil_p & \text{if } x/\delta \geq \frac{1}{2}\left(\lceil x/\delta \rceil_p + \lfloor x/\delta \rfloor_p\right) \\ \delta\lfloor x/\delta \rfloor_p & \text{if } x/\delta < \frac{1}{2}\left(\lceil x/\delta \rceil_p + \lfloor x/\delta \rfloor_p\right). \end{cases}.$$

The quantization function $\mathtt{Q}^S_{p,\delta}$ with stochastic rounding is

$$\mathtt{Q}^S_{p,\delta}(x) = \begin{cases} \delta\lceil x/\delta \rceil_p & \text{with probability } \frac{x/\delta - \lfloor x/\delta \rfloor_p}{\lceil x/\delta \rceil_p - \lfloor x/\delta \rfloor_p} \\ \delta\lfloor x/\delta \rfloor_p & \text{with probability } \frac{\lceil x/\delta \rceil_p - x/\delta}{\lceil x/\delta \rceil_p - \lfloor x/\delta \rfloor_p} \end{cases}.$$

The stochastic rounding ensures that the quantization is unbiased, i.e., $\mathbb{E}(\mathtt{Q}^S_{p,\delta}(x)|x) = x$.

3. Proposed Algorithm.

This section presents a mixed-precision SGD-type algorithm to reduce the memory and computation cost of CP tensor decomposition. This method has a linear convergence rate when it gets close to the optimal solution. A mixed-precision SignSGD method is utilized at the beginning to improve the convergence of the whole framework.

3.1. Mixed-Precision CP Decomposition.

Given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times \cdots \times N_m}$, the rank-$r$ CP tensor decomposition can be formulated as the optimization problem

$$\min_{\boldsymbol{\Theta}} f(\boldsymbol{\Theta}) = \|\boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1, \cdots, \mathbf{U}_m]\!]\|_{\mathrm{F}}^2, \text{ with } \boldsymbol{\Theta} = \left\{\mathbf{U}_i \in \mathbb{R}^{N_i \times r}\right\}_{i=1}^m. \tag{3.1}$$

This problem can be rewritten as

$$\min_{\boldsymbol{\Theta}} f := \frac{1}{N} \sum_{i_1=1}^{N_1} \cdots \sum_{i_m=1}^{N_m} \left(a_{i_1 \dots i_m} - [\![\mathbf{U}_1(i_1,:), \cdots, \mathbf{U}_m(i_m,:)]\!]\right)^2, \tag{3.2}$$

where $N := N_1 \cdots N_m$. Since the cost function is the summation of N functions, we can naturally apply an SGD-type method to solve the optimization.

Instead of using standard SGD [7], we present a mixed-precision SGD-type algorithm to solve Problem (3.1). Let $\mathbf{U}_1^s, \dots, \mathbf{U}_m^s$ be the tensor factor matrices in the s-th iteration and $\mathtt{Q}(\tilde{g}_i^s)$ be the quantized stochastic gradient with respect to $\mathbf{U}_i^s$. Corollary 4.2 shows that Problem (3.1) is locally strongly convex around the true decomposition if the leading rows of $\mathbf{U}_i(1,:)$ are fixed for $i = 2, \dots, m$. We propose to update variables as

$$\mathbf{U}_1^{s+1} = \mathbf{U}_1^s - \alpha_s \mathtt{Q}(\tilde{\mathbf{g}}_1^s),$$

$$\mathbf{U}_i^{s+1}(2:N_i,:) = \mathbf{U}_i^s(2:N_i,:) - \alpha_s \mathtt{Q}(\tilde{\mathbf{g}}_i^s)(2:N_i,:).$$

Algorithm 3.1 Mixed-Precision Stochastic Gradient Algorithm for Tensor Decomposition

```
1: Input: tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1\times\cdots\times N_m}$, rank $r$, initialization $\{\mathbf{U}_i^0 \in \mathbb{R}^{N_i\times r}\}_{i=1}^m$, initial learning rates $\alpha_0^{\text{sign}}, \alpha_0^{\text{SGD}} > 0$, thresholds $\epsilon_1 > \epsilon_2 > 0$, quantizations $\mathtt{Q}_1, \mathtt{Q}_2$, sample sizes $\{n_i\}_{i=1}^m$, learning rate update intervals $K^{\text{sign}}, K^{\text{SGD}}$, learning rate update constants $\eta^{\text{sign}}, \eta^{\text{SGD}}$.
2: Let $s = 0$.
3: $\alpha_0 = \alpha_0^{\text{sign}}$.
4: while $\|\boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1^s, \cdots, \mathbf{U}_m^s]\!]\|/\|\boldsymbol{\mathcal{A}}\| > \epsilon_1$ do
5:     Compute the mixed-precision gradient $\{\mathtt{Q}(\tilde{\mathbf{g}}_i^s)\}_{i=1}^m$ as in Algorithm 3.2.
6:     $\mathbf{U}_i^{s+1} = \mathbf{U}_i^s - \alpha_s \text{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s))$.
7:     $\alpha_{s+1} = \begin{cases} \eta^{\text{sign}}\alpha_s, & \text{if } ((s+1) \bmod K^{\text{sign}}) = 0 \\ \alpha_s, & \text{otherwise} \end{cases}$.
8:     $s = s + 1$.
9: end while
10: Let $\alpha_s = \alpha_0^{\text{SGD}}$, $s^{\text{sign}} = s$.
11: while $\|\boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1^s, \cdots, \mathbf{U}_m^s]\!]\|/\|\boldsymbol{\mathcal{A}}\| > \epsilon_2$. do
12:     Compute the mixed-precision gradient $\{\mathtt{Q}(\tilde{\mathbf{g}}_i^s)\}_{i=1}^m$ as in Algorithm 3.2.
13:     $\mathbf{U}_1^{s+1} = \mathbf{U}_1^s - \alpha_s \mathtt{Q}(\tilde{\mathbf{g}}_1^s)$.
14:     $\mathbf{U}_i^{s+1}(2:N_i,:) = \mathbf{U}_i^s(2:N_i,:) - \alpha_s \mathtt{Q}(\tilde{\mathbf{g}}_i^s)(2:N_i,:)$, $i = 2, \ldots, m$
15:     $\alpha_{s+1} = \begin{cases} \eta^{\text{SGD}}\alpha_s, & \text{if } ((s - s^{\text{sign}} + 1) \bmod K^{\text{SGD}}) = 0 \\ \alpha_s, & \text{otherwise} \end{cases}$.
16:     $s = s + 1$.
17: end while
18: Output: factor matrices $\{\mathbf{U}_i^s\}_{i=1}^m$.
```

Problem (3.1) has many stationary points, and the mixed-precision SGD can easily converge to a local optimizer without a good initialization point. We propose to use mixed-precision SignSGD to find a good initialization for SGD, which updates variables as follows:

$$\mathbf{U}_i^{s+1} = \mathbf{U}_i^s - \alpha_s \text{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)).$$

The mixed-precision SignSGD only uses the sign of the gradient to update parameters. Consequently, it is more robust against non-convexity and quantization errors. In practice, we find that SignSGD is unlikely to be trapped by a stationary point. This motivates us to firstly run mixed-precision SignSGD for a number of iterations. When the error becomes small, we switch to mixed-precision SGD for better accuracy and faster convergence.

The learning rate is updated as $\alpha_{s+1} = \eta\alpha_s$ every K iterations for some constant $1 > \gamma > 0$. It is the multi-stage update rule. The complete mixed-precision CP decomposition (3.1) is presented in Algorithm 3.1.

3.2. Mixed-Precision Block Stochastic Gradient.

Gradient computation is often the most expensive part in SGD-type algorithms. This subsection describes how to efficiently compute the mixed-precision stochastic gradient used in Algorithm 3.1.

Problem (3.2) is well-structured, therefore we use block sampling to maximize the usage of parallel computing. In each iteration, we uniformly sample a subset of indices $\mathcal{I}_i \subset [N_i]$

Algorithm 3.2 Compute Mixed-Precision Stochastic Gradient

```
1: Input: tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times \cdots \times N_m}$, rank $r$, factor matrices $\{\mathbf{U}_i^0 \in \mathbb{R}^{N_i \times r}\}_{i=1}^m$, quantization
   functions $\mathtt{Q}_1, \mathtt{Q}_2$, sample sizes $\{n_i\}_{i=1}^m$.
2: Randomly sample $\mathcal{I}_i \subset [n_i]$ with $|\mathcal{I}_i| = n_i$ for $i \in [m]$.
3: Compute $\boldsymbol{\mathcal{M}} = -\boldsymbol{\mathcal{A}}(\mathcal{I}_1, \ldots, \mathcal{I}_m) + [\![\mathtt{Q}_1(\mathbf{U}_1(\mathcal{I}_1, :)), \cdots, \mathtt{Q}_1(\mathbf{U}_m(\mathcal{I}_m, :))]\!]$.
4: Compute $\mathbf{V}_i = \odot_{j=1, j\neq i}^m \mathtt{Q}_1(\mathbf{U}_j(\mathcal{I}_j, :))$ for $i \in [m]$.
5: Compute $\mathtt{Q}(\tilde{\mathbf{g}}_i)(\mathcal{I}_i, :) = \mathtt{Q}_2(\mathbf{M}_{[i]})\mathtt{Q}_2(\mathbf{V}_i)$.
6: Output: Mixed-precision stochastic gradient $\{\mathtt{Q}(\tilde{\mathbf{g}}_i)\}_{i=1}^m$.
```

with $|\mathcal{I}_i| = n_i$ for $i = 1, \ldots, m$. Then, we consider the cost function

$$f_{\mathcal{I}} := \frac{1}{n} \|\boldsymbol{\mathcal{A}}(\mathcal{I}_1, \ldots, \mathcal{I}_m) - [\![\mathbf{U}_1(\mathcal{I}_1, :), \cdots, \mathbf{U}_m(\mathcal{I}_m, :)]\!]\|_{\mathrm{F}}^2, \text{ with } n = n_1 \cdots n_m.$$

The gradient of $f_{\mathcal{I}}$ with respect to $\mathbf{U}_i(\mathcal{I}_i, :)$ is

$$\nabla_{\mathbf{U}_i(\mathcal{I}_i, :)} f_{\mathcal{I}} = -\frac{2}{n} \left(\boldsymbol{\mathcal{A}}(\mathcal{I}_1, \ldots, \mathcal{I}_m) - [\![\mathbf{U}_1(\mathcal{I}_1, :), \cdots, \mathbf{U}_m(\mathcal{I}_m, :)]\!]\right)_{[i]} \odot_{j=1, j\neq i}^m \mathbf{U}_j(\mathcal{I}_j, :). \tag{3.3}$$

Therefore, the stochastic gradient $\tilde{\mathbf{g}}_i := \nabla_{\mathbf{U}_i} f_{\mathcal{I}} \in \mathbb{R}^{N_i \times r}$ is given as

$$\tilde{\mathbf{g}}_i(j_i, :) = \begin{cases} \nabla_{\mathbf{U}_i(j_i, :)} f_{\mathcal{I}} & \text{if } j_i \in \mathcal{I}_i \\ 0 & \text{if } j_i \notin \mathcal{I}_i \end{cases}. \tag{3.4}$$

We regard $\mathcal{I} := (\mathcal{I}_1, \ldots, \mathcal{I}_m)$ as a random variable. Each $\mathcal{I}_i$ is sampled uniformly, hence it holds that $\mathbb{E}_{\mathcal{I}}[\tilde{\mathbf{g}}_i] = \mathbf{g}_i$ for $i = 1, \ldots, m$, where $\mathbf{g}_i := \nabla_{\mathbf{U}_i} f$.

We compute the quantized value of the block stochastic gradient $\tilde{\mathbf{g}}_i$ (3.4) as follows:

$$\boldsymbol{\mathcal{M}} := -\boldsymbol{\mathcal{A}}(\mathcal{I}_1, \ldots, \mathcal{I}_m) + [\![\mathtt{Q}_1(\mathbf{U}_1(\mathcal{I}_1, :)), \cdots, \mathtt{Q}_1(\mathbf{U}_m(\mathcal{I}_m, :))]\!], \tag{3.5a}$$

$$\mathbf{V}_i := \odot_{j=1, j\neq i}^m \mathtt{Q}_1(\mathbf{U}_j(\mathcal{I}_j, :)), \tag{3.5b}$$

$$\mathtt{Q}(\tilde{\mathbf{g}}_i)(\mathcal{I}_i, :) := \mathtt{Q}_2(\mathbf{M}_{[i]})\mathtt{Q}_2(\mathbf{V}_i), \tag{3.5c}$$

where $\mathtt{Q}_1, \mathtt{Q}_2$ are two quantization functions as described in Section 2.2. The steps for computing the stochastic gradient in mixed-precision are summarized in Algorithm 3.2.

The subtraction in (3.5a) and the Khatri–Rao product in (3.5b) are both sensitive to quantization errors, and extremely low-precision quantization function $\mathtt{Q}_1$ will cause bad convergence behavior. Therefore, we use precision $\mathtt{FP}_{16}$ and scale $\delta = 1$ for $\mathtt{Q}_1$, i.e., $\mathtt{Q}_1 = \mathtt{Q}_{\mathtt{FP16},1}$. The last matrix multiplication (3.5c) is more robust against errors. Consequently, the quantization functions $\mathtt{Q}_2$ can use an extremely low precision. Practically, $\mathtt{INT}_4$ and $\mathtt{INT}_8$ always work well, and $\mathtt{INT}_2$ can work when the tensor rank r is small. For the specific quantization $\mathtt{Q}_{\mathtt{INT}_b, \delta}(\mathbf{X})$ for a matrix $\mathbf{X}$, the scaling factor δ depends on $\mathbf{X}$ and the precision $\mathtt{INT}_b$. We typically set δ slightly less than $\frac{\max\{\mathbf{X}\}}{2^{b-1}-1}$. This ensures most entries of $\mathbf{X}$ lie in the representation range of $\mathtt{INT}_b$ while preserving low quantization errors.

	$\texttt{INT}_8$	$\texttt{INT}_4$	$\texttt{INT}_2$
$m = 3$	21.9%	17.2%	14.8%
$m = 4$	20.0%	15.0%	12.5%
$m = 5$	18.75%	13.5%	10.9%

Table 3

Normalized computation cost compared with full-precision for various orders and precisions.

Complexity Analysis. The sub-tensor $\boldsymbol{\mathcal{M}}$ in (3.5a) is only computed once for all $i \in [m]$, and the computation requires around $2nr$ arithmetic operations. Computing each $\mathbf{V}_i$ in (3.5b) needs $\frac{N}{N_i}r$ arithmetic operations, so the total number of operations of step (3.5b) is $\sum_{i=1}^{m} \frac{n}{n_i}r$. Step (3.5c) involves a tensor unfolding along its ith dimension. The matrix multiplication (3.5c) for each i requires about $2nr$ operations. In total, we will do m such multiplications and the total number of operations is $2mnr$. Therefore, the most expensive step in (3.5) is the matrix multiplications (3.5c). Fortunately, (3.5c) is robust against quantization noises, and its cost can be reduced significantly by using ultra low-precision quantization functions. Suppose that each arithmetic operation of precision p costs c_p computation resources. Computing the mixed-precision gradient as in (3.5) requires $C(p_1, p_2) = c_{p_1}\left(2nr + \sum_{i=1}^{m} \frac{n}{n_i}r\right) + 2c_{p_2}mnr$ resources, where p_1, p_2 are the precision formats used by $\texttt{Q}_1, \texttt{Q}_2$ respectively. In practice, we typically choose p_1 as $\texttt{FP}_{16}$ and p_2 as some low-bit fixed-point format. The computation resource consumed by a specific representation format is proportional to the number of bits. On modern hardware, fixed-point operations typically use less resources and are much faster than floating-point operations. More specifically, fixed-point operations use less than half resources of floating-point operations with the same number of bits [27]. Therefore, we have the estimation $c_{\texttt{FP}_{16}} \approx \frac{1}{2}c_{\texttt{FP}_{32}}$, $c_{\texttt{INT}_b} \approx \frac{b}{64}c_{\texttt{FP}_{32}}$. Then, the estimated costs of (3.5) under full-precision and low-precision are

$$C(\texttt{FP}_{32}, \texttt{FP}_{32}) \approx (2 + 2m + \sum_{i=1}^{m} \frac{1}{n_i})nrc_{\texttt{FP}_{32}} \approx (2 + 2m)nrc_{\texttt{FP}_{32}},$$

$$C(\texttt{FP}_{16}, \texttt{INT}_b) \approx (1 + \frac{b}{32}m + \sum_{i=1}^{m} \frac{1}{n_i})nrc_{\texttt{FP}_{32}} \approx (1 + \frac{b}{32}m)nrc_{\texttt{FP}_{32}}.$$

The computation saving of using mixed-precision is

$$C(\texttt{FP}_{16}, \texttt{INT}_b)/C(\texttt{FP}_{32}, \texttt{FP}_{32}) \approx \frac{1 + \frac{b}{32}m}{2 + 2m}.$$

The cost reduction of our proposed mixed-precision gradient is more obvious for high-order tensors and smaller number of bits. Table 3 shows the normalized computational cost for orders $m = 3, 4, 5$ and precision $\texttt{INT}_8, \texttt{INT}_4, \texttt{INT}_2$, respectively.

4. Convergence Analysis.

This section presents the convergence result of Algorithm 3.1. Under some generic conditions, we prove that the tensor decomposition problem (3.1) is locally strongly convex after proper normalization. We prove that the mixed-precision SignSGD converges to some stationary points up to some noise caused by stochasticity and quantization errors. We also prove that the mixed-precision SGD has a locally linear convergence rate around the global minimizer.

4.1. Locally Strong Convexity. This subsection shows the locally strong convexity of the problem (3.1) after proper normalization. Note that Problem (3.1) itself is non-convex and it does not have local convexity as well. Suppose that the tensor $\boldsymbol{\mathcal{A}}$ has the CP decomposition $\boldsymbol{\mathcal{A}} = [\![\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_m]\!]$. Then, it holds that

$$\boldsymbol{\mathcal{A}} = \sum_{j=1}^{r} c_{1,j}\mathbf{U}_1(:,j) \circ \cdots \circ c_{m,j}\mathbf{U}_m(:,j), \tag{4.1}$$

for any $c_{i,j}$'s as long as $\Pi_{i=1}^{m} c_{1,j} = 1$. Therefore, the CP decomposition problem (3.1) has an infinite number of minimizers, but many solutions differ only with scaling factors. Therefore, we fix the elements $\{\mathbf{U}_i(1,:)\}_{i=2}^{m}$ and assume that $\mathbf{U}_i(1,:) = \mathbf{1}^T$ for $2 \leq i \leq m$ without loss of generality. The CP decomposition problem (3.1) now becomes

$$\min_{\tilde{\boldsymbol{\Theta}}} \tilde{f}(\tilde{\boldsymbol{\Theta}}) := \left\| \boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1, [\mathbf{1}^T; \tilde{\mathbf{U}}_2], \cdots, [\mathbf{1}^T; \tilde{\mathbf{U}}_m]]\!] \right\|^2, \tag{4.2}$$

where $\tilde{\boldsymbol{\Theta}} := (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \ldots, \tilde{\mathbf{U}}_m)$ and $\mathbf{U}_1 \in \mathbb{R}^{N_1 \times r}$, $\tilde{\mathbf{U}}_i \in \mathbb{R}^{(N_i-1)\times r}$ for $i = 2, \ldots, m$.

It can be shown that the normalized problem (4.2) is strongly convex around its global minimizers. For the tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_m}$ with $N_1 \geq N_2 \geq \cdots \geq N_m$, we define the largest rank r_m such that the problem (4.2) is locally strongly convex. Let

$$r_3 := \tilde{N}_{m-2} \lfloor \frac{\tilde{N}_{m-1}\tilde{N}_m}{\tilde{N}_{m-2} + \tilde{N}_{m-1} + \tilde{N}_m - 2} \rfloor, \tag{4.3}$$

where $\tilde{N}_{m-2}, \tilde{N}_{m-1}$ and $\tilde{N}_m$ are the largest integers such that (i) $\tilde{N}_{m-2}$ is even, (ii) $\tilde{N}_{m-2} \geq \tilde{N}_{m-1} \geq \tilde{N}_m$, and (iii) $N_i \geq \tilde{N}_i$ for $i = m-2, m-1, m$. Then, the upper bound r_m is computed recursively by

$$r_k := N_{m-k+1} \min\{r_{k-1}, \lfloor \frac{N_{m-k+2} \cdots N_m}{N_{m-k+1} + \cdots + N_m - k + 1} \rfloor\}, \; k = 4, \ldots, m. \tag{4.4}$$

The upper bound r_m is around $\frac{N_1 \cdots N_m}{N_1 + \cdots + N_m - m + 1}$ when $N_1, \ldots, N_m$ are large.

The locally strong convexity holds generically when $r \leq r_m$. We say a property is generic if it is true on the whole space except a subset with zero measure [14]. The rigorous result is presented in Theorem 4.1.

Theorem 4.1. *Suppose that $N_1 \geq N_2 \geq \cdots \geq N_m$ and $r \leq r_m$ for r_m in (4.4). Let*

$$\boldsymbol{\mathcal{A}} := [\![\mathbf{U}_1^*, [\mathbf{1}^T; \tilde{\mathbf{U}}_2^*], \cdots, [\mathbf{1}^T; \tilde{\mathbf{U}}_m^*]]\!],$$

where $\mathbf{U}_1^* \in \mathbb{R}^{N_1 \times r}$, $\tilde{\mathbf{U}}_i^* \in \mathbb{R}^{(N_i-1)\times r}, 2 \leq i \leq m$. Then, for generic $\tilde{\boldsymbol{\Theta}}^* := (\mathbf{U}_1^*, \ldots, \tilde{\mathbf{U}}_m^*)$, the Hessian matrix $\nabla^2 \tilde{f}(\tilde{\boldsymbol{\Theta}}^*)$ is positive definite.

Proof. See the proof in the Appendix A. ■

Problem (4.2) scales the leading rows to all one vectors, which simplifies the theoretical analysis. In practice, the leading rows can be scaled to arbitrary non-zero vectors. Consider the problem

$$\min_{\tilde{\boldsymbol{\Theta}}} \tilde{f}_A(\tilde{\boldsymbol{\Theta}}) := \|\boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1, [\mathbf{u}_2^T; \tilde{\mathbf{U}}_2], \cdots, [\mathbf{u}_m^T; \tilde{\mathbf{U}}_m]]\!]\|_{\mathrm{F}}^2, \tag{4.5}$$

where $\mathbf{u}_i \in \mathbb{R}^r$ and $(\mathbf{u}_i)_j \neq 0$, $\forall j \in [r]$ and $i \in [m]$. Problem (4.2) can be converted Problem (4.5) via some invertible transformations. The invertibility preserves the positive definiteness of the Hessian. Therefore, Problem (4.5) preserves the locally strong convexity.

Corollary 4.2. *Suppose $N_1 \geq N_2 \geq \cdots \geq N_m$ and $r \leq r_m$ in (4.4). Let $\mathbf{u}_2, \ldots, \mathbf{u}_m$ be vectors in $\mathbb{R}^r$ whose elements are all nonzero and*

$$\boldsymbol{\mathcal{A}} := [\![\mathbf{U}_1^*, [\mathbf{u}_2^T; \tilde{\mathbf{U}}_2^*], \cdots, [\mathbf{u}_m^T; \tilde{\mathbf{U}}_m^*]]\!],$$

where $\mathbf{U}_1^* \in \mathbb{R}^{N_1 \times r}, \tilde{\mathbf{U}}_i^* \in \mathbb{R}^{(N_i-1)\times r}, 2 \leq i \leq m$. Then, for generic $\tilde{\boldsymbol{\Theta}}^* := (\mathbf{U}_1^*, \ldots, \tilde{\mathbf{U}}_m^*)$, the Hessian $\nabla^2 \tilde{f}_A(\tilde{\boldsymbol{\Theta}}^*)$ is positive definite and there exists an open set O containing $\tilde{\boldsymbol{\Theta}}^*$ and a constant $\lambda > 0$ such that the function $\tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ in (4.5) is λ-strongly convex in O.

Proof. To simplify the descriptions, here we regard $\tilde{\boldsymbol{\Theta}}$ as a vector including all optimization variables. There exists a nonsingular matrix $\mathbf{D}$ such that $\tilde{f}_A(\tilde{\boldsymbol{\Theta}}) = \tilde{f}(\mathbf{D}\tilde{\boldsymbol{\Theta}})$. The Hessian $\nabla^2 \tilde{f}(\mathbf{D}\tilde{\boldsymbol{\Theta}}^*)$ is positive definite by Theorem 4.1. It holds that

$$\nabla^2 \tilde{f}_A(\tilde{\boldsymbol{\Theta}}^*) = \mathbf{D}(\nabla^2 \tilde{f}(\mathbf{D}\tilde{\boldsymbol{\Theta}}^*))\mathbf{D}^T.$$

Thus, the Hessian $\nabla^2 \tilde{f}_A(\tilde{\boldsymbol{\Theta}}^*)$ is positive definite. Since the eigenvalues of a matrix are continuous with respect to all matrix elements [28], there exists a constant $\lambda > 0$ and an open set O containing $\tilde{\boldsymbol{\Theta}}^*$ such that the smallest eigenvalue of $\nabla^2 \tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ is not less than λ in O. In other words, $\tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ is λ-strongly convex in O. ■

Based on Corollary 4.2, we can prove that Algorithm 3.1 has a local convergence rate in O after switching to mixed-precision SGD.

4.2. Convergence of Algorithm 3.1. We show that (1) the mixed-precision SignSGD in Algorithm 3.1 converges to a stationary point up to some noise, (2) the mixed-precision SGD in Algorithm 3.1 has a linear convergence rate around the true CP decomposition.

Let $\boldsymbol{\Theta}^s := (\mathbf{U}_1^s, \ldots, \mathbf{U}_m^s)$ denote the factor matrices at the s-th iteration. Suppose that $\{\boldsymbol{\Theta}^s\}_{s=0}^{S_1}$ and $\{\boldsymbol{\Theta}^s\}_{s=S_1+1}^{S_2}$ are generated by mixed-precision SignSGD and mixed-precision SGD respectively in Algorithm 3.1. Recall that f is the objective function defined in (3.1). We make the following assumptions.

Assumption 4.3. *Assume that $\|\nabla^2 f(\boldsymbol{\Theta}^s)\|_2 \leq L$, $s = 0, \ldots, S_2$.*

Assumption 4.4. *Let $\tilde{\mathbf{g}}^s$ be the stochastic gradient at the s-th iteration. Assume that for $s \in [S_2]$ and $i \in [m]$, it holds*

$$\mathbb{E}(\|\tilde{\mathbf{g}}_i^s(j,k) - \mathbf{g}_i^s(j,k)\|^2) \leq \sigma_g^2,\ \mathbb{E}(\|\mathcal{Q}(\tilde{\mathbf{g}}_i^s)(j,k) - \tilde{\mathbf{g}}_i^s(j,k)\|^2) \leq \sigma_Q^2,\ \forall j \in [N_i], k \in [r].$$

Assumption 4.3 assumes the Hessian matrices are bounded, which is widely used in the convergence analysis of SGD methods. Assumption 4.4 ensures the variance of the stochastic gradient and the quantization error are both bounded. Under Assumption 4.4, the quantized stochastic gradient can be bounded as

$$\begin{aligned}\mathbb{E}(\|\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)-\mathbf{g}_i^s(j,k)\|) &\le \mathbb{E}(\|\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)-\tilde{\mathbf{g}}_i^s(j,k)\|)+\mathbb{E}(\|\tilde{\mathbf{g}}_i^s(j,k)-\mathbf{g}_i^s(j,k)\|)\\ &\le \sqrt{\mathbb{E}(\|\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)-\tilde{\mathbf{g}}_i^s(j,k)\|^2)}+\sqrt{\mathbb{E}(\|\tilde{\mathbf{g}}_i^s(j,k)-\mathbf{g}_i^s(j,k)\|^2)}\\ &\le \sigma_Q+\sigma_g.\end{aligned}$$

4.2.1. Convergence of Mixed-Precision SignSGD. We show the convergence of the mixed-precision SignSGD in Algorithm 3.1. Our proof is partially motivated by [4].

Theorem 4.5. *Let $\{\boldsymbol{\Theta}^s\}_{s=0}^{S_1}$ be the sequence generated by the SignSGD update in Algorithm 3.1. Under Assumption 4.3 and Assumption 4.4, we have*

$$\sum_{s=0}^{S_1-1}\sum_{i=1}^{m}\frac{n_i}{N_i}\alpha_s\|\mathbf{g}_i^s\|_1 \le f(\boldsymbol{\Theta}^0)+\frac{1}{2}rL\sum_{s=0}^{S_1-1}\sum_{i=1}^{m}n_i\alpha_s^2+2r(\sigma_g+\sigma_Q)\sum_{s=1}^{S_1-1}\sum_{i=1}^{m}n_i\alpha_s. \tag{4.6}$$

Proof. Under Assumption 4.3, it holds that

(4.7)

$$\begin{aligned}f(\boldsymbol{\Theta}^{s+1}) \le& f(\boldsymbol{\Theta}^s)+\sum_{i=1}^{m}\mathrm{tr}\big((\mathbf{g}_i^s)^T(\mathbf{U}_i^{s+1}-\mathbf{U}_i^s)\big)+\sum_{i=1}^{m}\frac{L}{2}\|\mathbf{U}_i^{s+1}-\mathbf{U}_i^s\|_{\mathrm{F}}^2\\ =& f(\boldsymbol{\Theta}^s)-\alpha_s\sum_{i=1}^{m}\mathrm{tr}\big((\mathbf{g}_i^s(\mathcal{I}_i^s,:))^T\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(\mathcal{I}_i^s,:))\big)+\sum_{i=1}^{m}\frac{L}{2}\|\alpha_s\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(\mathcal{I}_i^s,:))\|^2\\ =& f(\boldsymbol{\Theta}^s)+\frac{1}{2}\alpha_s^2rL\sum_{i=1}^{m}n_i-\alpha_s\sum_{i=1}^{m}\|\mathbf{g}_i^s(\mathcal{I}_i^s,:)\|_1\\ &+2\alpha_s\sum_{i=1}^{m}\sum_{j\in\mathcal{I}_i^s}\sum_{k=1}^{r}|\mathbf{g}_i^s(j,k)|I\left(\mathrm{sign}\left(\mathbf{g}_i^s\left(j,k\right)\right)\neq\mathrm{sign}\left(\mathtt{Q}\left(\tilde{\mathbf{g}}_i^s\left(j,k\right)\right)\right)\right),\end{aligned}$$

where I is the indicator function such that $I(\text{true})=1, I(\text{false})=0$. Considering the part $I(\mathrm{sign}(\mathbf{g}_i^s(j,k))\neq\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s(j,k)))$ in the above, we have

$$\begin{aligned}\mathbb{E}[I(\mathrm{sign}(\mathbf{g}_i^s(j,k))\neq\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)] &= \mathbb{P}[I\big(\mathrm{sign}(\mathbf{g}_i^s(j,k))\neq\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k))\big)]\\ &\le \mathbb{P}[\|\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)-\mathbf{g}_i^s(j,k)\|\ge|\mathbf{g}_i^s(j,k)|]\\ &\le \frac{\mathbb{E}[\|\mathtt{Q}(\tilde{\mathbf{g}}_i^s)(j,k)-\mathbf{g}_i^s(j,k)\|]}{|\mathbf{g}_i^s(j,k)|}\\ &\le \frac{\sigma_g+\sigma_Q}{|\mathbf{g}_i^s(j,k)|}\end{aligned}$$

It implies that

$$\mathbb{E}[\sum_{j\in\mathcal{I}_i^s}\sum_{k=1}^{r}|\mathbf{g}_i^s(j,k)|I(\mathrm{sign}(\mathbf{g}_i^s(j,k))\neq\mathrm{sign}(\mathtt{Q}(\tilde{\mathbf{g}}_i^t)(j,k))]\le\sum_{j\in\mathcal{I}_i^s}\sum_{k=1}^{r}(\sigma_g+\sigma_Q)=n_ir(\sigma_g+\sigma_Q).$$

Then, we take expectation on both sides of (4.7) and get

$$f(\boldsymbol{\Theta}^{s+1}) \leq f(\boldsymbol{\Theta}^s) + \frac{1}{2}\alpha_s^2 rL \sum_{i=1}^m n_i - \alpha_s \sum_{i=1}^m \frac{n_i}{N_i}\|\mathbf{g}_i^s\|_1 + 2\alpha_s r(\sigma_g + \sigma_Q) \sum_{i=1}^m n_i.$$

After summing up both sides for $s = 0, \ldots, S_1 - 1$ and rearrangement, we have

$$\sum_{s=0}^{S_1-1} \sum_{i=1}^m \frac{n_i}{N_i}\alpha_s\|\mathbf{g}_i^s\|_1 \leq f(\boldsymbol{\Theta}^0) - f(\boldsymbol{\Theta}^{S_1}) + \frac{1}{2}rL \sum_{s=0}^{S_1-1} \sum_{i=1}^m n_i\alpha_s^2 + 2r(\sigma_g + \sigma_Q) \sum_{s=0}^{S_1-1} \sum_{i=1}^m n_i\alpha_s.$$

It implies the result (4.6) since $f(\boldsymbol{\Theta}^{S_1}) \geq 0$. ■

In practice, we usually choose a relatively large constant learning rate $\alpha_s = \alpha$ to accelerate the convergence at the beginning. We prove in Corollary 4.6 that a constant learning rate provides $O(\frac{1}{T_1})$ convergence rate up to some noise.

Corollary 4.6. *Under conditions of Theorem 4.5, if $\alpha_s = \alpha$, then*

$$\min_{s=0,\ldots,S_1-1} \|\mathbf{g}_i^s\|_1 \leq O(\frac{1}{S_1}) + \frac{1}{\gamma}(\frac{\alpha L}{2} + 2\sigma_g + 2\sigma_Q)r \sum_{i=1}^m n_i,$$

where $\gamma = \min_{i\in[m]} \frac{n_i}{N_i}$.

Proof. Equation (4.6) implies that

$$\begin{aligned}
\min_{s=0,\ldots,S_1-1} \|\mathbf{g}_i^s\|_1 &\leq \frac{1}{\gamma}\frac{f(\boldsymbol{\Theta}^0)}{\sum_{s=0}^{S_1-1}\alpha_s} + \frac{1}{\gamma}\left(\frac{1}{2}rL\sum_{i=1}^m n_i \frac{\sum_{t=0}^{T_1-1}\alpha_s^2}{\sum_{t=0}^{T_1-1}\alpha_s} + 2r(\sigma_g + \sigma_Q)\sum_{i=1}^m n_i\right) \\
&= \frac{f(\boldsymbol{\Theta}^0)}{S_1\gamma\alpha} + \frac{1}{\gamma}(\frac{\alpha L}{2} + 2\sigma_g + 2\sigma_Q)r\sum_{i=1}^m n_i \\
&= O(\frac{1}{S_1}) + \frac{1}{\gamma}(\frac{\alpha L}{2} + 2\sigma_g + 2\sigma_Q)r\sum_{i=1}^m n_i.
\end{aligned}$$

4.2.2. Convergence of Mixed-Precision SGD. In this subsection, we show the locally linear convergence rate of the mixed-precision SGD in Algorithm 3.1. We make the following extra assumption.

Assumption 4.7. *Assume that for some $\theta \in [0, 1)$, it holds*

$$\|\mathbb{E}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s) - \tilde{\mathbf{g}}_i^s)\| \leq \theta\|\mathbf{g}_i^s\|, \quad i \in [m],\ s = S_1 + 1, \ldots, S_2.$$

Assumption 4.7 assumes the quantized stochastic gradient is a good descent direction in expectation. If the quantization function Q uses independent stochastic rounding, then Assumption 4.7 is true for $\theta = 0$ since $\mathbb{E}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s)) = \mathbb{E}(\tilde{\mathbf{g}}_i^s) = \mathbf{g}_i^s$. Assumption 4.7 still holds for deterministic rounding as long as the quantization error is not large.

Let $\mathbf{u}_i := \mathbf{U}_i^{S_1}(1,:) \in \mathbb{R}^r$ for $i = 2, \ldots, m$. Suppose $\boldsymbol{\mathcal{A}} := [\![\mathbf{U}_1^*, [\mathbf{u}_2^T; \tilde{\mathbf{U}}_2^*], \cdots \circ [\mathbf{u}_m^T; \tilde{\mathbf{U}}_m^*]]\!]$, where $\mathbf{U}_1^* \in \mathbb{R}^{N_1\times r}$, $\tilde{\mathbf{U}}_i^* \in \mathbb{R}^{(N_i-1)\times r}, 2 \leq i \leq m$. The objective function now becomes $\tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ as in (4.5), which is locally λ-strongly convex by Corollary 4.2. Consequently, the convergence result of SGD for strongly convex functions can be applied.

Theorem 4.8. *Suppose the tensor $\boldsymbol{\mathcal{A}}$ satisfies the conditions of Corollary 4.2. Let $\{\boldsymbol{\Theta}^s\}_{s=S_1}^{S_2}$ be the sequence generated by the mixed-precision SGD in Algorithm 3.1 and $\alpha_s = \alpha$ be the learning rate. If $\tilde{\boldsymbol{\Theta}}^s$ is in the set O as in Corollary 4.2 and $\mathbb{E}(\|\mathtt{Q}(\tilde{\mathbf{g}}^s)\|) \le G$ for $s = S_1, \ldots, S_2$, then under Assumption 4.3, Assumption 4.4, and Assumption 4.7, it holds that*

$$\mathbb{E}(\|f(\boldsymbol{\Theta}^{S_2})\|^2) \le \frac{\alpha LG}{2\lambda(1-\theta)} + (1-\alpha\lambda(1-\theta))^{S_2-S_1}(f(\boldsymbol{\Theta}^{S_1}) - \frac{\alpha LG}{2\lambda(1-\theta)})$$

where λ is the strong convexity parameter in Corollary 4.2.

Proof. Under Assumption 4.7, the gradient $\tilde{\mathbf{g}}_i^s$ satisfies

$$\begin{aligned}\mathbb{E}((\mathbf{g}_i^s)^T\mathtt{Q}(\tilde{\mathbf{g}}_i^s)) &= (\mathbf{g}_i^s)^T\mathbb{E}(\tilde{\mathbf{g}}_i^s) + (\mathbf{g}_i^s)^T\mathbb{E}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s - \tilde{\mathbf{g}}_i^s)\\ &\ge \|\mathbf{g}_i^s\|^2 - \|\mathbf{g}_i^s\|\|\mathbb{E}(\mathtt{Q}(\tilde{\mathbf{g}}_i^s - \tilde{\mathbf{g}}_i^s)\|\\ &\ge (1-\theta)\|\mathbf{g}_i^s\|^2.\end{aligned}$$

By Corollary 4.2, the function $\tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ is λ-strongly convex in O containing $\tilde{\boldsymbol{\Theta}}^*$. Algorithm 3.1 is minimizing the function $\tilde{f}_A(\tilde{\boldsymbol{\Theta}})$ after switching to mixed-precision SGD. It also holds that $f(\boldsymbol{\Theta}^{S_2}) = f_A(\tilde{\boldsymbol{\Theta}}^{S_2})$. Therefore, the result is a direct conclusion of Theorem 4.6 in [7]. ■

5. Numerical Experiments.

5.1. Implementation Details.

Recall that the block stochastic gradient is computed as in Algorithm 3.2. The quantization function $\mathtt{Q}_1$ use $\mathtt{FP}_{16}$, scale factor $\delta = 1$, and deterministic rounding, i.e., $\mathtt{Q}_1 = \mathtt{Q}^D_{\mathtt{FP}_{16},1}$. The quantization function $\mathtt{Q}_2$ use $\mathtt{INT}_b$ precision and deterministic rounding. When quantizing the matrix $\mathbf{X}$, we use the scale factor $\delta = \frac{\max|\mathbf{X}|}{c}$, where $c \ge 2^{b-1} - 1$. Specifically, we use $c = 10, 30, 200$ for $\mathtt{INT}_2$, $\mathtt{INT}_4$, and $\mathtt{INT}_8$ respectively. In this section, the precision of Algorithm 3.1 always means the precision of $\mathtt{Q}_2$.

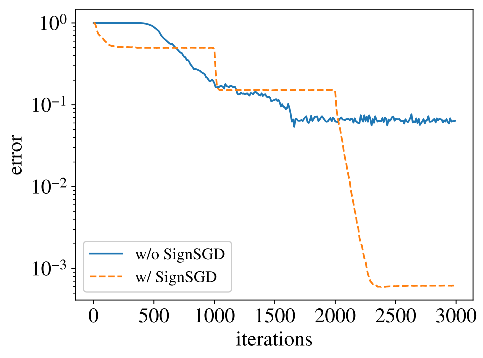
Our implementation uses the Python package CuPy [43]. For fair comparisons between different precisions, we implement the matrix multiplication by CUTLASS kernels [32]. However, due to the lack of support for extremely low-bit fixed-point integer representations in Python, we only compare the running time between $\mathtt{INT}_8$ and $\mathtt{FP}_{32}$ on GPU. The learning rate for the mixed-precision SignSGD in Algorithm 3.1 is set as 0.5 initially and is updated as $\alpha = 0.3\alpha$ every 1000 iterations. The mixed-precision SGD stage uses the constant learning rate $\alpha = 0.01$. For tensors of orders $3, 4, 5$, we use the sample sizes $|\mathcal{I}_i| = 0.2N_i, 0.3N_i, 0.4N_i$ respectively. The size of the sampled sub-tensor is roughly 1% of the original tensor.

Suppose that Algorithm 3.1 outputs the factor matrices $\{\mathbf{U}_i\}_{i=1}^m$ for the input tensor $\boldsymbol{\mathcal{A}}$. We use a relative error to measure the qualify of our results, which is defined as

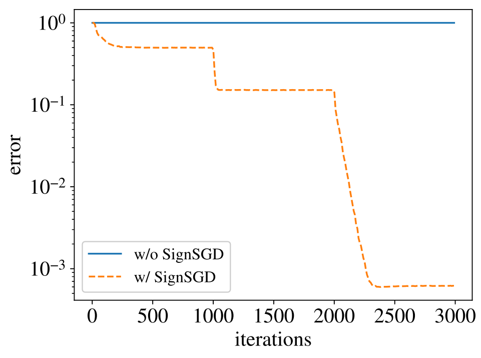
$$\text{error} = \frac{\|\boldsymbol{\mathcal{A}} - [\![\mathbf{U}_1, \cdots, \mathbf{U}_m]\!]\|_{\mathrm{F}}}{\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}}}.$$

5.2. Synthetic Examples.

We first test the runtime and convergence of Algorithm 3.1 under various precisions on some synthetic tensor benchmarks.



(a) Initialization with max 1.0



(b) Initialization with max 0.1 and 0.01

Figure 1. *Performance of Algorithm 3.1 with and without SignSGD initialization.*

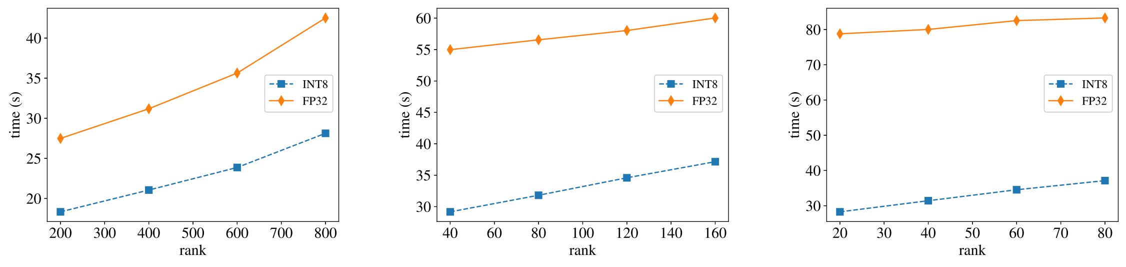
Dimension	Sample size	Rank	$\texttt{INT}_8$ time (s)	$\texttt{FP}_{32}$ time (s)	Speed up
(1200,1200,1200)	(240,240,240)	256	17.96	28.15	**1.56**$\times$
(200,200,200,200)	(60,60,60,60)	64	29.90	56.44	**1.88**$\times$
(60,60,60,60,60)	(24,24,24,24,24)	32	41.71	114.89	**2.75**$\times$

Table 4

Time comparison between $\texttt{FP}_{32}$ and $\texttt{INT}_8$ of Algorithm 3.1 for various dimensions

5.2.1. Role of SignSGD Initialization. This section runs the experiment in full precision to show the influence of SignSGD initialization to the convergence of the whole algorithm. The results with different initialization methods are shown in Figure 1. The “max” in Figure 1 is the maximum absolute value of each $\{\mathbf{U}_i\}_{i=1}^m$. Algorithm 3.1 without SignSGD is trapped by a stationary point and fails to converge with max $= 1.0$ as shown in Figure 1a. After we decrease max to 0.1 and 0.01, Algorithm 3.1 without SignSGD stays at zero, which is a stationary point. In contrast, Algorithm 3.1 with SignSGD converges well for max$=1.0, 0.1, 0.01$. The result demonstrates that the SignSGD initialization can greatly improve the convergence of Algorithm 3.1.

5.2.2. Time Comparison in Different Precisions. We test the runtime of Algorithm 3.1 to reach the same relative error 10^{-3} under different precisions. We specifically compare the runtime of Algorithm 3.1 with $\texttt{INT}_8$ and $\texttt{FP}_{32}$ respectively, and the result is summarized in Table 4. Figure 2a, 2b, 2c show that the runtime increases linearly as the tensor rank increases for both low-precision and full-precision. The reduction ratio remains the same as the rank changes. We can observe significant time savings when using the $\texttt{INT}_8$ format for all sizes, ranks, and orders. The time saving is also more remarkable as the tensor order increases. This is because m large-size matrix multiplications are computed in low precision in Algorithm 3.1 for tensors with order m. Therefore, higher order m brings in more time savings. The detailed complexity analysis is in Section 3.2.



(a) Size (1200,1200,1200) (b) Size (200,200,200,200) (c) Size (60,60,60,60,60)

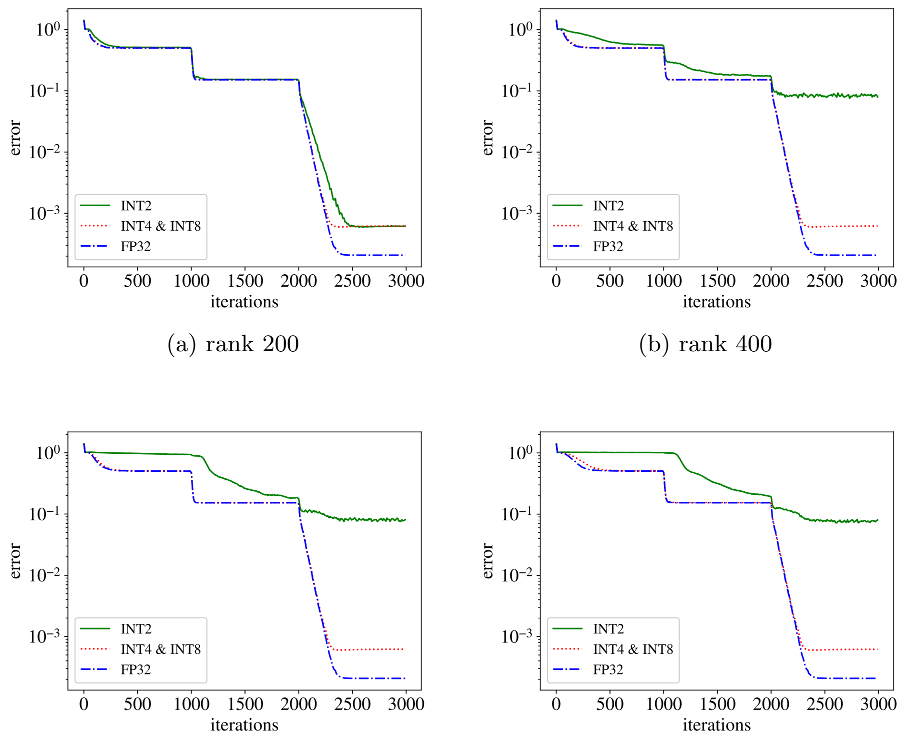
Figure 2. *Time comparison between $\mathtt{FP}_{32}$ and $\mathtt{INT}_8$ of Algorithm 3.1 for various ranks*

5.2.3. Convergence Comparison in Different Precisions. We further evaluate the convergence of Algorithm 3.1 under various precisions. We compare precisions $\mathtt{INT}_2$, $\mathtt{INT}_4$, $\mathtt{INT}_8$, and $\mathtt{FP}_{32}$, where the computation of $\mathtt{INT}_2$ and $\mathtt{INT}_4$ is simulated by $\mathtt{FP}_{32}$. The simulation simply rounds the scaled number into the nearest integer and then clamps it into the representation range. The convergence of $\mathtt{INT}_4$ and $\mathtt{INT}_8$ precision are almost the same as the convergence of $\mathtt{FP}_{32}$, so they are combined in Figure 3. The final relative error of low-precision Algorithm 3.1 is slightly worse than the full-precision version due to the quantization error. The quantization error also causes the slow convergence for rank 200 and the divergence for higher ranks of $\mathtt{INT}_2$ precision. The noisy ball term in Corollary 4.6 for SignSGD depends on the rank r and the quantization error σ_Q. Therefore, a large rank r and large quantization error σ_Q may lead to bad convergence due to the large noisy ball. The mixed-precision SGD part starts at around the 2000th iteration. Figure 3 shows that the mixed-precision SGD has a linear convergence rate which matches the theoretical result in Theorem 4.8. The slower convergence of the mixed-precision SGD part of $\mathtt{INT}_2$ precision in Figure 3a is caused by the large θ in Assumption 4.7 due to the quantization error.

5.3. Real Datasets.

5.3.1. Coil-100 Dataset. The Coil-100 dataset [38] contains the images of 100 objects in 72 different poses. Each image has size $128 \times 128 \times 3$, where 128×128 is the number of pixels and 3 represents the 3 RGB channels. Thus, the size of the formed tensor $\boldsymbol{\mathcal{A}}$ is $128 \times 128 \times 3 \times 7200$. The CP decomposition is applied for dimension reduction. The fourth-factor matrix $\mathbf{U}_4 \in \mathbb{R}^{7200\times r}$ can be used as features for clustering and classification tasks.

We run Algorithm 3.1 on the tensor $\boldsymbol{\mathcal{A}}$ with precisions $\mathtt{INT}_2$, $\mathtt{INT}_4$, $\mathtt{INT}_8$, and $\mathtt{FP}_{32}$. The test employs rank $r = 16$ and sample size $(32, 32, 3, 1440)$. As shown in Figure 4, the convergence trends are similar for all precisions. Higher numerical precisions produce smaller relative errors in the final solution, but the difference is insignificant. All of our relative errors are better than the best reported result in [3], which is 0.314. Regarding the running time, $\mathtt{INT}_8$ takes 11 seconds for 1000 iterations while $\mathtt{FP}_{32}$ takes 31 seconds. Algorithm 3.1 of the $\mathtt{INT}_8$ precision is **2.8** times faster. The experiment demonstrates that our proposed mixed-precision algorithm can effectively reduce the computation cost of CP tensor decomposition on real-world datasets with negligible accuracy loss.



(a) rank 200 (b) rank 400

(c) rank 600 (d) rank 800

Figure 3. *Convergence comparison of various precision on the tensor of dimension (1200,1200,1200).*

5.3.2. MRI Dataset. Magnetic Resonance Imaging (MRI) is widely used in brain science and clinic diagnosis. Low-rank tensor decomposition can be applied to denoise practical MR images [52]. This experiment uses the data from the NYU fastMRI Initiative database [34, 53]. The original data is in a Fourier space and forms a complex tensor $\boldsymbol{\mathcal{K}}$ of size $16 \times 640 \times 320$. The tensor $\boldsymbol{\mathcal{K}}$ in real-world is typically corrupted by noises. In our test, we intentionally corrupt $\boldsymbol{\mathcal{K}}$ by the noise tensor $\boldsymbol{\mathcal{N}}$. The real part and imaginary part of $\boldsymbol{\mathcal{N}}$ both obey the normal distribution with mean 0 and variance τ^2. Let the corrupted tensor be $\widehat{\boldsymbol{\mathcal{K}}} = \boldsymbol{\mathcal{K}} + \boldsymbol{\mathcal{N}}$, then the inverse Fourier transform is applied to $\widehat{\boldsymbol{\mathcal{K}}}$ to get $\widehat{\boldsymbol{\mathcal{A}}} \in \mathbb{C}^{16\times 640\times 320}$. Next, we use Algorithm 3.1 to find a low-rank approximation $\boldsymbol{\mathcal{A}}$ of the noisy tensor $\widehat{\boldsymbol{\mathcal{A}}}$ for noise removal. Finally, the gray-scale image $\mathbf{M} \in \mathbb{R}^{640\times 320}$ is reconstructed as

$$\mathbf{M}_{i,j} = \sqrt{\sum_{k=1}^{16} |\boldsymbol{\mathcal{A}}(k,i,j)|^2}.$$

The image $\mathbf{M}$ is further cropped into size 320×320 by selecting $\mathbf{M}(161:480,:)$. Let $\mathbf{M}_{\text{truth}}$ be the ground-truth of the image, then the relative error is computed as

$$\text{error} = \frac{\|\mathbf{M} - \mathbf{M}_{\text{truth}}\|}{\|\mathbf{M}_{\text{truth}}\|}.$$

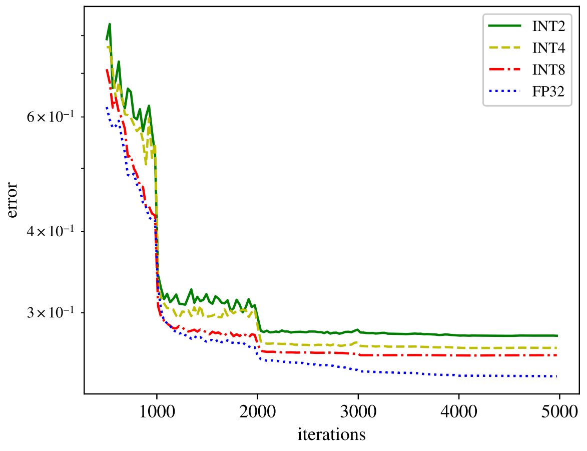


Figure 4. *Converge curves of Algorithm 3.1 in various precisions on Coil-100 data*

τ	w/o CP	$\mathtt{INT}_4$	$\mathtt{INT}_8$	$\mathtt{FP}_{32}$
1.0e-5	0.379	0.115	0.109	0.109
1.5e-5	0.639	0.173	0.164	0.165
2.0e-5	0.913	0.246	0.236	0.236

Table 5
Relative errors before and after removing noises of MRI by Algorithm 3.1

This experiment applied Algorithm 3.1 to the complex tensors. We would like to remark that Algorithm 3.1 is designed for real tensors, but we can extend the algorithm to complex tensors by considering the real part and the imaginary part separately.

Our experiment uses rank $r = 200$, sample size $(16, 64, 64)$ and noise tensors $\boldsymbol{\mathcal{N}}$ of standard deviations $\tau = 1.0 \times 10^{-5}, 1.5 \times 10^{-5}, 2.0 \times 10^{-5}$, respectively. We test the performance of Algorithm 3.1 in various precisions. The test results are presented in Table 5 and Figure 5. Table 5 lists the relative errors before and after CP decompositions. Algorithm 3.1 successfully removes the noises and reduces the errors. Moreover, the performance of Algorithm 3.1 in $\mathtt{INT}_4$ and $\mathtt{INT}_8$ is similar to $\mathtt{FP}_{32}$. Figure 5 compares the recovered images and the ground-truth image. The images obtained via $\mathtt{INT}_4$, $\mathtt{INT}_8$, $\mathtt{FP}_{32}$ mixed-precision CP decomposition are visually identical and vastly superior to those without noise removal. The results demonstrate that the mixed-precision Algorithm 3.1 is capable of producing accurate decomposition for noisy MRI datasets.

5.4. FPGA Demonstration for Edge Computing. The proposed mixed-precision CP decomposition can reduce the computing cost on both cloud and edge devices. Here we implement Algorithm 3.1 on a Field Programmable Gate Array (FPGA) to demonstrate its benefit on resource-constrained edge devices. FPGAs are widely used for edge computing due to their energy efficiency, flexible reconfigurability, and fast time-to-market [48]. However, FP-

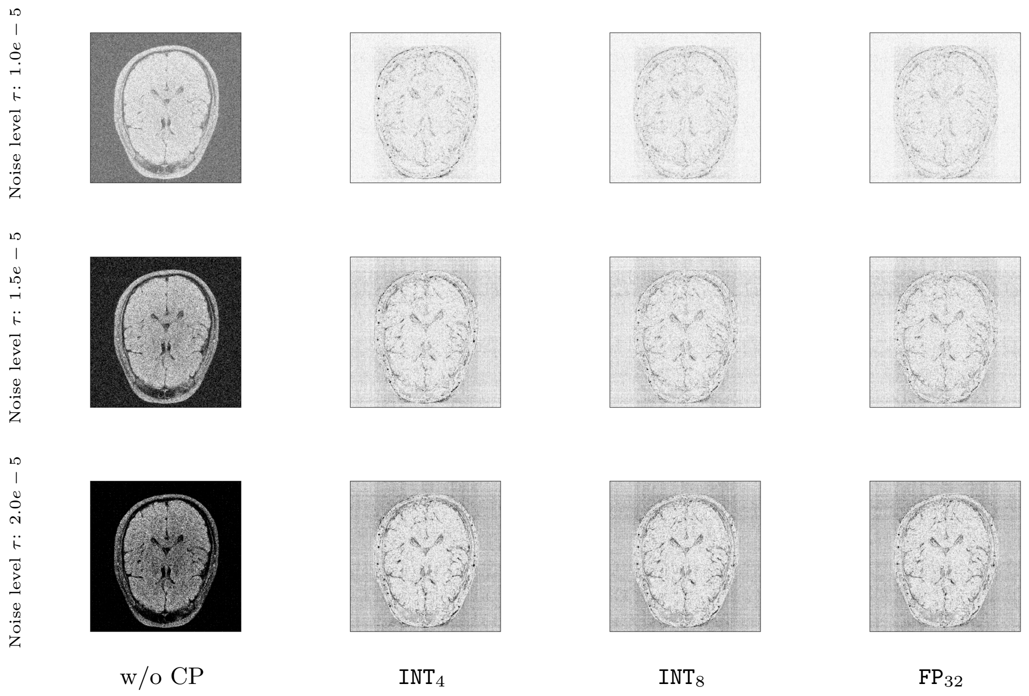

Noise level τ: 1.0e − 5
Noise level τ: 1.5e − 5
Noise level τ: 2.0e − 5
w/o CP
INT$_4$
INT$_8$
FP$_{32}$

Figure 5. *Error images before and after removing noises of MRI by Algorithm 3.1*

Precision	Time (s)	BRAM	FF	Power (W)	Energy (J)
INT$_2$	**3.08**	**12**	**86025**	**8.4**	**25.87**
INT$_8$	**3.08**	15	197868	9.5	29.26
FP$_{32}$	8.61	250	624416	13.3	114.4

Table 6
FPGA implementations of Algorithm 3.1 in different precisions

GAs have very limited memory and computing resources, therefore, it is desired to use low numerical precision to save the hardware cost in massive engineering applications.

We consider a rank-20 tensor with size $(100, 100, 100)$ and sample size $(20, 20, 20)$. Table 6 shows the performance of Algorithm 3.1 on FPGA in different precisions. The integer operations accelerate Algorithm 3.1 about **2.8** times compared to FP$_{32}$. The running time of Algorithm 3.1 in INT$_2$ and INT$_8$ are dominated by higher precision parts in the algorithm. Consequently, INT$_2$ and INT$_8$ have the similar running time. The usage of BRAM (block random-access memory) of INT$_2$ and INT$_8$ is about **20** times less than FP$_{32}$. Due to the reduced hardware resource requirements, the power consumption of INT$_2$ and INT$_8$ is also reduced. The energy cost of FP$_{32}$ is 4 times more than the energy consumed by INT$_2$ and INT$_8$. The number of FF (Flip-Flop) used by INT$_2$ is **2.3** times less than INT$_8$ and **7.2** times less than FP$_{32}$, respectively. In summary, the reduction of time and resources on FPGAs successfully demonstrates the effectiveness of our proposed mixed-precision algorithm on resource-constrained devices.

6. Conclusion.

This paper has proposed a mixed-precision stochastic gradient method for the CP tensor decomposition problem. First, the stochastic gradient is computed in mixed-precision to reduce the runtime and computation cost. Then, we develop a two-stage optimization algorithm to solve the CP decomposition problem using mixed-precision gradients. The convergence of the proposed algorithm has been proved. We have shown that the CP decomposition problem is locally strongly convex after proper normalization. Consequently, the mixed-precision SGD stage in our algorithm can have a linear convergence rate. A set of numerical experiments on GPUs and on edge devices have successfully demonstrated that our mixed-precision algorithm can significantly reduce the computation costs and latency compared to the full-precision algorithm while maintaining high accuracy. The proposed mixed-precision stochastic gradient method can be applied to many gradient-based CP decomposition algorithms. It will be an interesting future topic to study the applications of mixed-precision gradients to other optimization algorithms.

Appendix A. Proof of Theorem 4.1.

In the appendix, we will give the concrete proof of Theorem 4.1. We first show the Hessian is positive definite if and only the Jacobian matrix has full column rank. Then, we prove that the Jacobian matrix has a full column rank in general.

We consider the following vector-valued function

$$\tilde{\mathbf{h}}(\tilde{\mathbf{\Theta}}) = \text{vec}\left([\![\mathbf{U}_1, [\mathbf{1}^T; \tilde{\mathbf{U}}_2], \cdots, [\mathbf{1}^T; \tilde{\mathbf{U}}_m]]\!] \right), \tag{A.1}$$

where vec$(\cdot)$ is the vectorization function. It holds that $\tilde{f}(\tilde{\mathbf{\Theta}}) = \|\tilde{\mathbf{h}}(\tilde{\mathbf{\Theta}}) - \text{vec}(\boldsymbol{\mathcal{A}})\|^2$.

Lemma A.1. *Suppose that $\tilde{f}(\tilde{\mathbf{\Theta}}) = 0$, then $\nabla^2 \tilde{f}(\tilde{\mathbf{\Theta}})$ is positive definite if and only if the Jacobian $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank.*

Proof. $\tilde{f}(\tilde{\mathbf{\Theta}}) = 0$ implies that $\tilde{\mathbf{h}}(\tilde{\mathbf{\Theta}}) - \text{vec}(\boldsymbol{\mathcal{A}}) = \mathbf{0}$. Then, we have

$$\nabla \tilde{f}(\tilde{\mathbf{\Theta}}) = 2(\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}}))^T \tilde{\mathbf{h}}(\tilde{\mathbf{\Theta}}),\ \nabla^2 \tilde{F}(\tilde{\mathbf{\Theta}}) = 2(\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}}))^T \mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}}).$$

As a result, $\nabla^2 \tilde{f}(\tilde{\mathbf{\Theta}})$ is positive definite if and only $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank. ■

In the following, we first prove $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ generally has full column rank for the order $m = 3$ and the even N_1, then we extend the result to general orders and dimensions.

Proposition A.2. *When $m = 3$, N_1 is even, $N_1 \geq N_2 \geq N_3 \geq 3$, and $r \leq N_1 \lfloor \frac{n_2 n_3}{N_1+N_2+N_3-2} \rfloor$, the Jacobian matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank for generic $\tilde{\mathbf{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \tilde{\mathbf{U}}_3)$.*

Proof. It suffices to only consider $r = N_1 \lfloor \frac{n_2 n_3}{N_1+N_2+N_3-2} \rfloor$. The Jacobian matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank if and only $(\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}}))^T \mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has non-zero determinant, which is a polynomial function p in terms of variables $\tilde{\mathbf{\Theta}}$. The conclusion is then equivalent to that $p(\tilde{\mathbf{\Theta}})$ is nonzero for generic $\tilde{\mathbf{\Theta}}$. Thus, it suffices to show $p(\tilde{\mathbf{\Theta}})$ is not the constant 0 polynomial [14]. In the following, we will construct the specific $\tilde{\mathbf{\Theta}}$ such that $p(\tilde{\mathbf{\Theta}}) \neq 0$.

Here we denote $\mathbf{U}_i = [\mathbf{1}^T; \tilde{\mathbf{U}}_i]$ for convenience. Let $k = \frac{N_1}{2}$ and $R = 2\lfloor \frac{n_2 n_3}{N_1+N_2+N_3-2} \rfloor$. We evenly split $\{1, \ldots, r\}$ into k groups such that each group has $R = 2\lfloor \frac{n_2 n_3}{N_1+N_2+N_3-2} \rfloor$ elements.

We denote the groups by $\text{group}_1, \dots, \text{group}_k$. For $i = 1, \dots, k$, let

$$\mathbf{J}_i = \Big[\big[[\mathbf{e}_{2i-1}, \mathbf{e}_{2i}] \otimes \mathbf{U}_2(:,j) \otimes \mathbf{U}_3(:,j) \big]_{j=1}^{r},\ \big[\mathbf{U}_1(:,j) \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{U}_3(:,j) \big]_{j \in \text{group}_i}, \\ \big[\mathbf{U}_1(:,j) \otimes \mathbf{U}_2(:,j) \otimes \tilde{\mathbf{I}}_{N_3} \big]_{j \in \text{group}_i} \Big],$$

where $\tilde{\mathbf{I}}_n := \mathbf{I}_n(2:n,:)$. The Jacobian $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})$ can be written as $\big[\mathbf{J}_1, \dots, \mathbf{J}_k\big]$.

We construct the matrix $\mathbf{U}_1$, where for $j \in \text{group}_i$

$$\mathbf{U}_1(l,j) = 0, \text{ if } l \neq 2i-1, 2i.$$

For such $\mathbf{U}_1$, the Jocabian $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})$ has full column rank if and only if each $\mathbf{J}_i$ has full column rank. In the following, we will construct $\tilde{\boldsymbol{\Theta}}$ such that $\mathbf{J}_1$ has full column rank.

Since $N_1 \geq N_2 \geq N_3 \geq 3$, it holds that

$$R = 2\lfloor \frac{n_2 n_3}{N_1 + N_2 + N_3 - 2} \rfloor \leq \frac{2n_2 n_3}{2n_2 + N_3 - 2} \leq \frac{2n_2 n_3}{2n_2 - 1} < \frac{2n_2 n_3}{2n_2} = N_3.$$

Thus, we have $R < N_3 \leq N_2$. Let $\mathbf{a}_1, \dots, \mathbf{a}_r$ be pairwisely independent vectors, $\mathbf{b}_1, \dots, \mathbf{b}_{N_2}$ and $\mathbf{c}_1, \dots, \mathbf{c}_{N_3}$ be orthonormal basis of $\mathbb{R}^{N_2}$ and $\mathbb{R}^{N_3}$, respectively. The orthonormal basis can be chosen such that all leading entries are nonzero. Denote the matrix

$$\mathbf{P} := \Big[\big[\mathbf{I}_2 \otimes \mathbf{b}_j \otimes \mathbf{c}_j, \mathbf{a}_j \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_j, \mathbf{a}_j \otimes \mathbf{b}_j \otimes \tilde{\mathbf{I}}_{N_3} \big]_{j=1}^{R}, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3 \Big],$$

where

$$\mathbf{Q}_1 = \big[\mathbf{I}_2 \otimes \mathbf{b}_i \otimes \mathbf{c}_j \big]_{R+1 \leq i \leq N_2, R+1 \leq j \leq N_3},$$

$$\mathbf{Q}_2 = \big[\mathbf{I}_2 \otimes \mathbf{b}_i \otimes (\mathbf{c}_{2j-1} + \mathbf{c}_{2j}) \big]_{R+1 \leq i \leq N_2, 1 \leq j \leq R/2},$$

$$\mathbf{Q}_3 = \big[\mathbf{I}_2 \otimes (\mathbf{b}_{2i-1} + \mathbf{b}_{2i}) \otimes \mathbf{c}_j \big]_{1 \leq i \leq R/2, R+1 \leq j \leq N_3}.$$

Next, we show the matrix $\mathbf{P}$ has full column rank. It is equivalent to proving that $\mathbf{P}\mathbf{x} = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{0}$. We first prove the coefficients for $\mathbf{Q}_1$ is zero. For some $R+1 \leq i \leq N_2, R+1 \leq j \leq N_3$, it holds that

$$\big(\mathbf{I}_2 \otimes \mathbf{b}_i^T \otimes \mathbf{c}_j^T \big) \mathbf{P}\mathbf{x} = \mathbf{I}_2 \boldsymbol{\lambda} = \mathbf{0} \Rightarrow \boldsymbol{\lambda} = \mathbf{0},$$

where $\boldsymbol{\lambda}$ is the coefficient vector corresponding to $\mathbf{I}_2 \otimes \mathbf{b}_i \otimes \mathbf{c}_j$ in $\mathbf{x}$. Thus the coefficient for $\mathbf{Q}_1$ is zero. Then we show the coefficient for $\mathbf{Q}_2$ is zero. For some $R+1 \leq i \leq N_2, 1 \leq j \leq R/2$, it holds

$$\big(\mathbf{I}_2 \otimes \mathbf{b}_i^T \otimes \mathbf{c}_{2j-1}^T \big) \mathbf{P}\mathbf{x} = \big(\mathbf{a}_{2j-1} \otimes \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2}, \mathbf{I}_2 \big) \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\mu} \end{bmatrix} = \mathbf{0},$$

$$\big(\mathbf{I}_2 \otimes \mathbf{b}_i^T \otimes \mathbf{c}_{2j}^T \big) \mathbf{P}\mathbf{x} = \big(\mathbf{a}_{2j} \otimes \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2}, \mathbf{I}_2 \big) \begin{bmatrix} \boldsymbol{\lambda}_2 \\ \boldsymbol{\mu} \end{bmatrix} = \mathbf{0},$$

where $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\mu}$ are coefficients corresponding to $\mathbf{a}_{2j-1} \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_{2j}, \mathbf{a}_{2j} \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_{2j}$ and $\mathbf{I}_2 \otimes \mathbf{b}_i \otimes (\mathbf{c}_{2j-1} + \mathbf{c}_{2j})$ respectively. By above equations, we know

$$\boldsymbol{\mu} = -(\mathbf{a}_{2j-1} \otimes \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2})\boldsymbol{\lambda}_1 = -(\mathbf{a}_{2j} \otimes \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2})\boldsymbol{\lambda}_2.$$

Thus, $\mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2} \boldsymbol{\lambda}_1 = \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2} \boldsymbol{\lambda}_2 = \mathbf{0}$ since $\mathbf{a}_{2j-1}, \mathbf{a}_{2j}$ are linearly independent. It implies that $\boldsymbol{\mu} = 0$. Therefore, the coefficient for $\mathbf{Q}_2$ is zero. Similarly, we can prove the coefficient for $\mathbf{Q}_3$ is zero by using exactly the same technique.

Next, we show the coefficient of $\mathbf{a}_j \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_j$ is zero. For some $1 \le i \le N_2$ and $i \neq j$, it holds

$$\left(\mathbf{I}_2 \otimes \mathbf{b}_i^T \otimes \mathbf{c}_j^T\right)\mathbf{P}\mathbf{x} = \left(\mathbf{a}_j \otimes \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2}, \mathbf{a}_i \otimes \mathbf{c}_j^T \tilde{\mathbf{I}}_{N_3}\right) \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda}_j \end{bmatrix} = \mathbf{0} \Rightarrow \mathbf{b}_i^T \tilde{\mathbf{I}}_{N_2} \boldsymbol{\mu} = \mathbf{c}_j^T \tilde{\mathbf{I}}_{N_3} \boldsymbol{\lambda}_j = 0,$$

where $\boldsymbol{\mu}, \boldsymbol{\lambda}_j$ are coefficients corresponding to $\mathbf{a}_j \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_j, \mathbf{a}_i \otimes \mathbf{b}_i \otimes \tilde{\mathbf{I}}_{N_3}$ respectively. The above equation holds for every i such that $1 \le i \le N_2$ and $i \neq j$. Therefore, we have

$$\left[\tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_1, \ldots, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{j-1}, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{j+1}, \ldots, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{N_2}\right]^T \boldsymbol{\mu} = \mathbf{0} \Rightarrow \boldsymbol{\mu} = \mathbf{0}.$$

The above holds because $\left[\tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_1, \ldots, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{j-1}, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{j+1}, \ldots, \tilde{\mathbf{I}}_{N_2}^T \mathbf{b}_{N_2}\right]^T \in \mathbb{R}^{(N_2-1)\times(N_2-1)}$ is nonsingular. It proves that the coefficient for $\mathbf{a}_j \otimes \tilde{\mathbf{I}}_{N_2} \otimes \mathbf{c}_j$ is zero. Similarly, we can prove the coefficient for $\mathbf{a}_j \otimes \mathbf{b}_j \otimes \tilde{\mathbf{I}}_{N_3}$ is zero.

The only remaining part in $\mathbf{P}$ is $\left[\mathbf{I}_2 \otimes \mathbf{b}_j \otimes \mathbf{c}_j\right]_{j=1}^R$. $\left[\mathbf{I}_2 \otimes \mathbf{b}_j \otimes \mathbf{c}_j\right]_{j=1}^R$ has full column rank since $\mathbf{b}_1, \ldots, \mathbf{b}_R$ are linearly independent. Thus, the coefficients corresponding to $\left[\mathbf{I}_2 \otimes \mathbf{b}_j \otimes \mathbf{c}_j\right]_{j=1}^R$ are also zero. It finishes the proof that $\mathbf{P}\mathbf{x} = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{0}$. Thus, $\mathbf{P}$ has full column rank.

Let

$$\mathbf{Q} := \Bigg[\left[\frac{\mathbf{b}_i}{(\mathbf{b}_i)_1} \otimes \frac{\mathbf{c}_j}{(\mathbf{c}_j)_1}\right]_{R+1\le i\le N_2, R+1\le j\le N_3}, \left[\frac{\mathbf{b}_i}{(\mathbf{b}_i)_1} \otimes \frac{\mathbf{c}_{2j-1} + \mathbf{c}_{2j}}{(\mathbf{c}_{2j-1})_1 + (\mathbf{c}_{2j})_1}\right]_{R+1\le i\le N_2, 1\le j\le R/2}, \\ \left[\frac{\mathbf{b}_{2i-1} + \mathbf{b}_{2i}}{(\mathbf{b}_{2i-1})_1 + (\mathbf{b}_{2i})_1} \otimes \frac{\mathbf{c}_j}{(\mathbf{c}_j)_1}\right]_{1\le i\le R/2, R+1\le j\le N_3}\Bigg].$$

The number of columns of $\mathbf{Q}$ is

$$c = (N_2 - R)(N_3 - R) + (N_2 - R)\frac{R}{2} + (N_3 - R)\frac{R}{2} = n_2 n_3 - \frac{R}{2}(N_2 + N_3).$$

It holds that,

$$c - (r - R) = n_2 n_3 - \frac{R}{2}(N_2 + N_3 - 2) - \frac{n_1 R}{2} = n_2 n_3 - \frac{R}{2}(N_1 + N_2 + N_3 - 2) \ge 0.$$

Let $\mathbf{U}_1(1:2, j) = \mathbf{a}_j, \tilde{\mathbf{U}}_2(:, j) = \mathbf{b}_j(2:)/(\mathbf{b}_j)_1, \tilde{\mathbf{U}}_3(:, j) = \mathbf{c}_j(2:)/(\mathbf{c}_j)_1$ for $j = 1, \ldots, R$ and $\tilde{\mathbf{U}}_2(:, j), \tilde{\mathbf{U}}_3(:, j)$ be vectors such that

$$\left[[1; \tilde{\mathbf{U}}_2(:, j)] \otimes [1; \tilde{\mathbf{U}}_3(:, j)]\right]_{R+1\le j\le r} = \mathbf{Q}(:, 1 : r - R).$$

Then, all columns of $\mathbf{J}_1$ are from $\mathbf{P}$. We have shown that $\mathbf{P}$ has full column rank, so $\mathbf{J}_1$ must have full column rank. The same technique can be applied to $\mathbf{J}_1, \ldots, \mathbf{J}_k$.

Now we have proven that $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})$ is has full column rank for some $\tilde{\boldsymbol{\Theta}}$. Therefore, the Jacobian $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})$ has full column rank for generic $\tilde{\boldsymbol{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \tilde{\mathbf{U}}_3)$. ■

Proposition A.2 proves the case when N_1 is even. If N_1 is odd, we may simply consider $N_1 - 1$ to make the largest dimension even. The result is stated in the following corollary.

Corollary A.3. *If $m = 3$, $N_1 \geq N_2 \geq N_3 \geq 3$, and $r \leq r_3$ for r_3 in (4.3), then the Jacobian matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank for generic $\tilde{\mathbf{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \tilde{\mathbf{U}}_3)$.*

Proof. When N_1 is even, it is the result of Proposition A.2.

When N_1 is odd, we consider the dimension $(\tilde{N}_1, \tilde{N}_2, \tilde{N}_3)$, where $\tilde{N}_1, \tilde{N}_2, \tilde{N}_3$ are largest integers such that $\tilde{N}_1$ is even, $\tilde{N}_1 \geq \tilde{N}_2 \geq \tilde{N}_3$, and $N_i \geq \tilde{N}_i$ for $i = 1, 2, 3$. Let $\mathbf{V}_1 := \mathbf{U}_1(1 : \tilde{N}_1, :), \tilde{\mathbf{V}}_2 := \tilde{\mathbf{U}}_2(1 : \tilde{N}_2 - 1, :), \tilde{\mathbf{V}}_3 := \tilde{\mathbf{U}}_3(1 : \tilde{N}_3 - 1, :)$ By Proposition A.2, the matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\mathbf{V}_1, \tilde{\mathbf{V}}_2, \tilde{\mathbf{V}}_3)$ has full column rank for generic $\mathbf{V}_1, \tilde{\mathbf{V}}_2, \tilde{\mathbf{V}}_3$. The matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\mathbf{V}_1, \tilde{\mathbf{V}}_2, \tilde{\mathbf{V}}_3)$ only consists of some rows in $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$. Thus, $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank if $\mathbf{J}_{\tilde{\mathbf{h}}}(\mathbf{V}_1, \tilde{\mathbf{V}}_2, \tilde{\mathbf{V}}_3)$ has full column rank. It proves $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank for generic $\tilde{\mathbf{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \tilde{\mathbf{U}}_3)$. ■

We have shown the local convexity for order-3 tensors. Finally, we prove the higher order cases by induction on the order m with base case $m = 3$.

Theorem A.4. *If $r \leq r_m$ for r_m in (4.4), then the Jacobian matrix $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})$ has full column rank for generic $\tilde{\mathbf{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \dots, \tilde{\mathbf{U}}_m)$.*

Proof. We will prove the result by induction on the order m. Proposition A.2 proves the conclusion for $m = 3$. Suppose that the result holds for the order $m - 1$, then we show it also holds for m. Similar to the proof of Theorem A.2, it suffices to prove there exists some $\tilde{\mathbf{\Theta}}$ such that the Jacobian is has full column rank [14].

Denote $L := \min\{r_{m-1}, \lfloor \frac{n_2 n_3 \cdots N_m}{N_1 + \cdots + N_m - m + 1} \rfloor\}$ and $\mathbf{Q}_1 = \odot_{i=2}^m [\mathbf{1}^T; \tilde{\mathbf{U}}_i(:, 1 : L)]$,

$$\mathbf{Q}_k = \left[\otimes_{i=2}^{k-1} [\mathbf{1}^T; \tilde{\mathbf{U}}_i(:, j)] \otimes \tilde{\mathbf{I}}_{n_k} \otimes_{i=k+1}^m [\mathbf{1}^T; \tilde{\mathbf{U}}_i(:, j)] \right]_{j=1}^L, \; k = 2, \dots, m.$$

Let $\mathbf{U}_1(:, i) = \mathbf{e}_j$ for $(j-1)L + 1 \leq i \leq jL$, then (after omitting zero columns) it holds that

$$\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\mathbf{\Theta}})(1 : \Pi_{i=2}^m N_i, :) = \left[\odot_{i=2}^m [\mathbf{1}^T; \tilde{\mathbf{U}}_i(:, L+1 : r)], \, \mathbf{Q}_1, \dots, \mathbf{Q}_m \right].$$ ■

Let $\mathbf{S} := [\mathbf{Q}_1, \dots, \mathbf{Q}_m]$. The matrix $\mathbf{S}$ has the same column space as $\mathbf{J}_{\tilde{\mathbf{h}}}([\mathbf{1}^T; \tilde{\mathbf{U}}_2(:, 1 : L)], \dots, \tilde{\mathbf{U}}_m(:, 1 : L))$. By the induction assumption, we know $\mathbf{S}$ generically has full column rank since $L \leq r_{m-1}$. The number of columns of $\mathbf{S}$ is $c = (N_2 + \cdots + N_m - m + 2)L$, then

$$\begin{aligned}
\Pi_{i=2}^m N_i - c &\geq \Pi_{i=2}^m N_i - (N_2 + \cdots + N_m - m + 2) \lfloor \frac{\Pi_{i=2}^m N_i}{N_1 + \cdots + N_m - m + 1} \rfloor \\
&\geq \Pi_{i=2}^m N_i - \Pi_{i=2}^m N_i \frac{N_2 + \cdots + N_m - m + 2}{N_1 + \cdots + N_m - m + 1} \\
&= \Pi_{i=2}^m N_i \left(1 - \frac{N_2 + \cdots + N_m - m + 2}{N_1 + \cdots + N_m - m + 1}\right) \\
&= \Pi_{i=2}^m N_i \frac{N_1 - 1}{N_1 + \cdots + N_m - m + 1} \\
&\geq (N_1 - 1) \lfloor \frac{\Pi_{i=2}^m N_i}{N_1 + \cdots + N_m - m + 1} \rfloor \\
&= r - L.
\end{aligned}$$

Thus, we could choose columns $\{\tilde{\mathbf{U}}_i(:, L+1:r)\}_{i=1}^m$ such that $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})(1:\Pi_{i=2}^m N_i, :)$ is linearly independent. It proves that $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})(1:\Pi_{i=2}^m N_i, :)$ has full column rank generically. The same proof can be applied to $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})((j-1)\Pi_{i=2}^m N_i + 1 : j\Pi_{i=2}^m N_i, :), j = 1, \ldots, N_1$. Therefore, $\mathbf{J}_{\tilde{\mathbf{h}}}(\tilde{\boldsymbol{\Theta}})$ has full column rank for generic $\tilde{\boldsymbol{\Theta}} = (\mathbf{U}_1, \tilde{\mathbf{U}}_2, \ldots, \tilde{\mathbf{U}}_m)$.

Now, we are ready to prove Theorem 4.1.

Proof. Under the assumption of Theorem 4.1, it holds that $\tilde{f}(\tilde{\boldsymbol{\Theta}}^*) = 0$. Therefore, Lemma A.1 and Theorem A.4 imply that the Hessian $\nabla^2 \tilde{f}(\tilde{\boldsymbol{\Theta}}^*)$ is positive definite for generic $\tilde{\boldsymbol{\Theta}}^*$. ■

REFERENCES

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, et al., *A survey of numerical linear algebra methods utilizing mixed-precision arithmetic*, The International Journal of High Performance Computing Applications, 35 (2021), pp. 344–369.

[2] 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 (2014), pp. 2773–2832.

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

[4] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, *signSGD: Compressed optimisation for non-convex problems*, in International Conference on Machine Learning, PMLR, 2018, pp. 560–569.

[5] A. Beutel, P. P. Talukdar, A. Kumar, C. Faloutsos, E. E. Papalexakis, and E. P. Xing, *Flexifact: Scalable flexible factorization of coupled tensors on hadoop*, in Proceedings of the 2014 SIAM international conference on data mining, SIAM, 2014, pp. 109–117.

[6] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, *Spectral tensor-train decomposition*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2405–A2439.

[7] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, Siam Review, 60 (2018), pp. 223–311.

[8] R. Bro, *PARAFAC. tutorial and applications*, Chemometrics and intelligent laboratory systems, 38 (1997), pp. 149–171.

[9] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek, and J. Kurzak, *Mixed precision iterative refinement techniques for the solution of dense linear systems*, The International Journal of High Performance Computing Applications, 21 (2007), pp. 457–466.

[10] E. Carson and N. J. Higham, *Accelerating the solution of linear systems by iterative refinement in three precisions*, SIAM Journal on Scientific Computing, 40 (2018), pp. A817–A847.

[11] E. Carson, N. J. Higham, and S. Pranesh, *Three-precision GMRES-based iterative refinement for least squares problems*, SIAM Journal on Scientific Computing, 42 (2020), pp. A4063–A4083.

[12] E. Carson and N. Khan, *Mixed precision iterative refinement with sparse approximate inverse preconditioning*, arXiv preprint arXiv:2202.10204, (2022).

[13] P. Comon, X. Luciani, and A. L. De Almeida, *Tensor decompositions, alternating least squares and other tales*, Journal of Chemometrics: A Journal of the Chemometrics Society, 23 (2009), pp. 393–405.

[14] D. Cox, J. Little, and D. OShea, *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, Springer Science & Business Media, 2013.

[15] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278.

[16] C. De Sa, M. Feldman, C. Ré, and K. Olukotun, *Understanding and optimizing asynchronous low-precision stochastic gradient descent*, in Proceedings of the 44th annual international symposium on computer architecture, 2017, pp. 561–574.

[17] C. De Sa, M. Leszczynski, J. Zhang, A. Marzoev, C. R. Aberger, K. Olukotun, and C. Ré, *High-accuracy low-precision training*, arXiv preprint arXiv:1803.03383, (2018).

[18] S. Dolgov, B. N. Khoromskij, A. Litvinenko, and H. G. Matthies, *Polynomial chaos expansion of random coefficients and the solution of stochastic partial differential equations in the tensor train format*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 1109–1135.

[19] I. Domanov and L. De Lathauwer, *Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm*, Linear Algebra and its Applications, 513 (2017), pp. 342–375.

[20] M. Dressler, J. Nie, and Z. Yang, *Separability of hermitian tensors and psd decompositions*, Linear and Multilinear Algebra, (2021), pp. 1–28.

[21] R. Ge and T. Ma, *On the optimization landscape of tensor decompositions*, Advances in Neural Information Processing Systems, 30 (2017).

[22] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM journal on matrix analysis and applications, 31 (2010), pp. 2029–2054.

[23] B. Guo, J. Nie, and Z. Yang, *Learning diagonal gaussian mixture models and incomplete tensor decompositions*, Vietnam Journal of Mathematics, 50 (2022), pp. 421–446.

[24] A. Haidar, H. Bayraktar, S. Tomov, J. Dongarra, and N. J. Higham, *Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems*, Proceedings of the Royal Society A, 476 (2020), p. 20200110.

[25] C. Hawkins, X. Liu, and Z. Zhang, *Towards compact neural networks via end-to-end training: A Bayesian tensor approach with automatic rank determination*, SIAM Journal on Mathematics of Data Science, 4 (2022), pp. 46–71.

[26] C. Hawkins and Z. Zhang, *Bayesian tensorized neural networks with automatic rank selection*, Neurocomputing, 453 (2021), pp. 172–180.

[27] D. L. N. Hettiarachchi, V. S. P. Davuluru, and E. J. Balster, *Integer vs. floating-point processing on modern fpga technology*, in 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2020, pp. 0606–0612.

[28] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge university press, 2012.

[29] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, *Quantized neural networks: Training neural networks with low precision weights and activations*, The Journal of Machine Learning Research, 18 (2017), pp. 6869–6898.

[30] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, *Towards quantum machine learning with tensor networks*, Quantum Science and technology, 4 (2019), p. 024001.

[31] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al., *In-datacenter performance analysis of a tensor processing unit*, in Proceedings of the 44th annual international symposium on computer architecture, 2017, pp. 1–12.

[32] A. Kerr, H. Wu, M. Gupta, D. Blasig, P. Ramini, D. Merrill, A. Shivam, P. Majcher, P. Springer, M. Hohnerbach, J. Wang, and M. Nicely, *CUTLASS*, 4 2022, https://github.com/NVIDIA/cutlass.

[33] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, *Compression of deep convolutional neural networks for fast and low power mobile applications*, arXiv preprint arXiv:1511.06530, (2015).

[34] F. Knoll, J. Zbontar, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, et al., *fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning*, Radiology. Artificial intelligence, 2 (2020).

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

[36] T. G. Kolda and D. Hong, *Stochastic gradients for large-scale tensor decomposition*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1066–1095.

[37] J. M. Landsberg, *Tensors: geometry and applications*, Representation theory, 381 (2012), p. 3.

[38] S. A. Nene, S. K. Nayar, H. Murase, et al., *Columbia object image library (coil-100)*, (1996).

[39] J. Nie, *Generating polynomials and symmetric tensor decompositions*, Foundations of Computational Mathematics, 17 (2017), pp. 423–465.

[40] J. Nie and Z. Yang, *Hermitian tensor decompositions*, SIAM Journal on Matrix Analysis and Applications, 41 (2020), pp. 1115–1144.

[41] J. Nie, Z. Yang, and X. Zhang, *A complete semidefinite algorithm for detecting copositive matrices*

and tensors, SIAM Journal on Optimization, 28 (2018), pp. 2902–2921.

[42] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, *Tensorizing neural networks*, Advances in neural information processing systems, 28 (2015).

[43] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, *CuPy: A NumPy-compatible library for NVIDIA GPU calculations*, in Proc. Workshop on Machine Learning Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS), 2017.

[44] R. Olivares-Amaya, M. Watson, R. Edgar, L. Vogt, Y. Shao, and A. Aspuru-Guzik, *Accelerating correlated quantum chemistry calculations using graphical processing units and a mixed precision matrix multiplication library*, Journal of chemical theory and computation, 6 (2010), pp. 135–144.

[45] R. Orús, *Tensor networks for complex quantum systems*, Nature Reviews Physics, 1 (2019), pp. 538–550.

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

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

[48] J. Shan, M. R. Casu, J. Cortadella, L. Lavagno, and M. T. Lazarescu, *Exact and heuristic allocation of multi-kernel applications to multi-FPGA platforms*, in Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6.

[49] 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 (2017), pp. 3551–3582.

[50] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, *Ultra-low precision 4-bit training of deep neural networks*, Advances in Neural Information Processing Systems, 33 (2020), pp. 1796–1807.

[51] N. Vervliet and L. De Lathauwer, *A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors*, IEEE Journal of Selected Topics in Signal Processing, 10 (2015), pp. 284–295.

[52] B. Yaman, S. Weingärtner, N. Kargas, N. D. Sidiropoulos, and M. Akçakaya, *Low-rank tensor models for improved multidimensional MRI: Application to dynamic cardiac t_1 mapping*, IEEE transactions on computational imaging, 6 (2019), pp. 194–207.

[53] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, et al., *fastMRI: An open dataset and benchmarks for accelerated MRI*, arXiv preprint arXiv:1811.08839, (2018).

[54] K. Zhang, C. Hawkins, X. Zhang, C. Hao, and Z. Zhang, *On-FPGA training with ultra memory reduction: A low-precision tensor method*, arXiv preprint arXiv:2104.03420, (2021).

[55] Z. Zhang, T.-W. Weng, and L. Daniel, *Big-data tensor recovery for high-dimensional uncertainty quantification of process variations*, IEEE Transactions on Components, Packaging and Manufacturing Technology, 7 (2016), pp. 687–697.

[56] Z. Zhang, X. Yang, I. V. Oseledets, G. E. Karniadakis, and L. Daniel, *Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34 (2014), pp. 63–76.