Multi-Way Data Representation: A Comprehensive Survey on Tensor Decomposition in Machine Learning

Aliona Tatyana
Lomonosov Moscow State University
aliona.tatyana@rector.msu.ru

Abstract

Tensor decomposition has emerged as a fundamental tool in machine learning, enabling efficient representation, compression, and interpretation of high-dimensional data. Unlike traditional matrix factorization methods, tensor decomposition extends to multi-way data structures, capturing complex relationships and latent patterns that would otherwise remain hidden. This paper provides a comprehensive overview of tensor decomposition techniques, including CANDECOMP/PARAFAC (CP), Tucker, and Tensor Train (TT) decompositions, and their applications in various machine learning domains. We explore optimization strategies, computational challenges, and real-world case studies demonstrating the effectiveness of tensor methods in areas such as natural language processing, recommender systems, deep learning compression, and biomedical informatics. Furthermore, we discuss emerging trends and future research directions, including the integration of tensor decomposition with deep learning, scalability improvements, and applications in quantum computing. As machine learning continues to evolve, tensor decomposition is poised to play an increasingly critical role in data-driven discovery and model interpretability.

Keywords: Tensor decomposition, machine learning, CP decomposition, Tucker decomposition, Tensor Train, high-dimensional data, model compression, deep learning, recommender systems, optimization, scalability.

1 Introduction

Machine learning has witnessed remarkable advancements in recent years, driven by the increasing availability of large-scale data and improvements in compu-

tational resources. A key challenge in modern machine learning is the efficient representation, storage, and processing of high-dimensional data [1]. Traditional methods often rely on matrix-based representations; however, many real-world datasets are naturally structured as multi-dimensional arrays, known as tensors. The need to extend conventional linear algebra techniques to multi-way data has led to the growing interest in tensor decomposition, a mathematical framework that generalizes matrix factorization to higher-order structures [2]. Tensors arise in numerous applications, including computer vision, natural language processing, biomedical data analysis, recommender systems, and scientific computing [3]. For instance, in image and video processing, data is inherently multi-dimensional, consisting of spatial and temporal components. In natural language processing, word co-occurrence statistics and contextual embeddings can be effectively represented as tensors [4]. Similarly, in neuroscience, brain imaging data collected through functional magnetic resonance imaging (fMRI) naturally conforms to tensor structures, capturing spatial, temporal, and frequency information [5]. These examples highlight the necessity of tensor-based methods for analyzing and extracting meaningful patterns from complex datasets [6]. Tensor decomposition techniques, such as CANDECOMP/PARAFAC (CP), Tucker decomposition, Tensor Train (TT), and Hierarchical Tucker (HT) decomposition, provide a systematic approach to breaking down high-dimensional tensors into lower-rank components while preserving essential structural information. These methods facilitate efficient storage, reduce computational complexity, and improve interpretability by revealing latent factors underlying multi-way data. Compared to traditional matrix factorization techniques, tensor decompositions capture richer interactions and dependencies across multiple modes, making them particularly suitable for machine learning tasks involving structured data [7]. The integration of tensor decomposition with machine learning has led to significant advancements across various domains. In deep learning, tensor methods have been employed to compress large neural networks by factorizing weight tensors, leading to reduced memory consumption and faster inference times. In unsupervised learning, tensor decomposition has been used for clustering, dimensionality reduction, and feature extraction, providing a more compact and meaningful representation of high-dimensional data. Additionally, in reinforcement learning and graph-based learning, tensor-based approaches have demonstrated effectiveness in modeling multi-agent interactions and structured dependencies. Despite its advantages, tensor decomposition also presents several challenges. The computational complexity of decomposing large-scale tensors remains a significant bottleneck, requiring efficient optimization techniques and scalable algorithms [8]. Furthermore, selecting the appropriate decomposition method and rank estimation is often non-trivial and problem-dependent [9]. Advances in randomized algorithms, GPU acceleration, and distributed comput-

ing have made it possible to apply tensor methods to large datasets, yet there is ongoing research to further improve their scalability and robustness. This paper provides a comprehensive overview of tensor decomposition techniques and their applications in machine learning [10]. We begin with a formal introduction to tensors, their mathematical properties, and commonly used decomposition methods [11]. We then explore various applications of tensor-based approaches in supervised, unsupervised, and deep learning settings [12]. Furthermore, we discuss recent developments, challenges, and future directions in tensor decomposition research [13]. Our goal is to equip researchers and practitioners with the necessary insights to leverage tensor methods effectively in machine learning applications, ultimately enhancing the interpretability, efficiency, and scalability of modern models [14].

2 Mathematical Preliminaries

To understand tensor decomposition and its applications in machine learning, it is essential to establish the necessary mathematical foundations. This section introduces basic tensor notation, key operations, and fundamental concepts used in tensor analysis.

2.1 Notation and Definitions

A tensor is a multi-dimensional array that generalizes vectors (one-dimensional) and matrices (two-dimensional) to higher dimensions [15]. Formally, an N-th order tensor, denoted as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, consists of elements $x_{i_1,i_2,\ldots,i_N}$ indexed by N indices. The number of dimensions (or modes) of a tensor is referred to as its order [16]. Commonly used special cases of tensors include:

- Scalars: A zero-order tensor, represented by a single numerical value, denoted as $x \in \mathbb{R}$.
- Vectors: A first-order tensor, denoted as $\mathbf{x} \in \mathbb{R}^I$.
- Matrices: A second-order tensor, denoted as $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$.

2.2 Tensor Operations

Several tensor operations play a crucial role in tensor decomposition methods [17]. Below, we define key operations:

2.2.1 Tensor Unfolding (Matricization)

Tensor unfolding (or matricization) refers to reshaping a tensor into a matrix by reordering its elements. The mode-n unfolding of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted as $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_N)}$.

2.2.2 Tensor Vectorization

The vectorization of a tensor $\mathcal{X}$, denoted as $\text{vec}(\mathcal{X})$, is the process of converting all elements of $\mathcal{X}$ into a column vector.

2.2.3 Mode-n Product

The mode-n product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is denoted as:

$$\mathcal{X} \times_n \mathbf{U} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N} \tag{1}$$

where the multiplication is performed along the n-th mode [18].

2.2.4 Kronecker, Khatri-Rao, and Hadamard Products

Several matrix products are frequently used in tensor decomposition:

- **Kronecker Product:** The Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is given by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \ldots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \ldots & a_{IJ}\mathbf{B} \end{bmatrix}. \tag{2}$$

- **Khatri-Rao Product:** The Khatri-Rao product is the column-wise Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$:

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_K \otimes \mathbf{b}_K]. \tag{3}$$

- **Hadamard Product:** The Hadamard product of two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{I \times J}$ is the element-wise product:

$$(\mathbf{A} \circ \mathbf{B})_{ij} = a_{ij} b_{ij} [19]. \tag{4}$$

2.3 Tensor Rank and Low-Rank Approximations

One of the most fundamental concepts in tensor decomposition is tensor rank, which generalizes the notion of matrix rank [20].

2.3.1 CP Rank

The CP rank of a tensor $\mathcal{X}$ is the minimal number of rank-one tensors that sum to reconstruct $\mathcal{X}$:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{5}$$

where $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ are factor vectors.

2.3.2 Tucker Rank

The Tucker rank of a tensor is a tuple $(R_1, R_2, \ldots, R_N)$ representing the ranks of the mode-wise matricizations of the tensor [21].

2.3.3 Low-Rank Tensor Approximations

Finding low-rank approximations to tensors is crucial for reducing dimensionality and computational complexity [22]. The most common methods include CP decomposition, Tucker decomposition, and Tensor Train decomposition [23]. This section has established the fundamental mathematical concepts necessary to understand tensor decomposition. In the next section, we will explore the core tensor decomposition techniques in detail.

3 Tensor Decomposition Methods

Tensor decomposition methods provide a way to factorize multi-dimensional arrays into lower-dimensional components, enabling efficient storage, computation, and interpretation [24]. These techniques extend classical matrix factorization methods to higher-order tensors and play a crucial role in machine learning applications such as dimensionality reduction, feature extraction, and model compression. In this section, we introduce three widely used tensor decomposition methods: CANDECOMP/PARAFAC (CP) decomposition, Tucker decomposition, and Tensor Train (TT) decomposition.

3.1 CANDECOMP/PARAFAC (CP) Decomposition

The CANDECOMP/PARAFAC (CP) decomposition, also known as the CP model, expresses a tensor as a sum of rank-one component tensors [25]. Given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, its CP decomposition is represented as:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{6}$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ are factor vectors corresponding to the r-th component, and $\circ$ denotes the outer product.

3.1.1 Properties and Applications

- CP decomposition provides an intuitive and interpretable factorization by identifying latent factors in multi-dimensional data.
- It is widely used in applications such as topic modeling, recommendation systems, and biomedical signal processing.
- The rank R must be chosen carefully to balance approximation accuracy and computational efficiency.

3.1.2 Computation

Computing the CP decomposition is typically performed using Alternating Least Squares (ALS) or stochastic gradient-based optimization methods. The cost of computing CP decomposition grows with tensor size and rank, making efficient implementations crucial [26].

3.2 Tucker Decomposition

Tucker decomposition generalizes CP decomposition by introducing a core tensor that captures interactions between factor matrices. A third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be decomposed as:

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}, \tag{7}$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is the core tensor, and $\mathbf{U} \in \mathbb{R}^{I \times R_1}$, $\mathbf{V} \in \mathbb{R}^{J \times R_2}$, and $\mathbf{W} \in \mathbb{R}^{K \times R_3}$ are factor matrices [27].

3.2.1 Properties and Applications

- Tucker decomposition allows more flexible rank constraints compared to CP decomposition, making it useful for compression and dimensionality reduction.
- It is widely used in multi-modal data analysis, image compression, and neuroscience [28].
- The core tensor captures complex interactions between modes, leading to improved interpretability in some applications [29].

3.2.2 Computation

Computing the Tucker decomposition typically involves Higher-Order Singular Value Decomposition (HOSVD) or Higher-Order Orthogonal Iteration (HOOI). These methods rely on singular value decomposition (SVD) of matricized versions of the tensor [30].

3.3 Tensor Train (TT) Decomposition

Tensor Train (TT) decomposition represents a high-order tensor as a sequence of smaller core tensors connected in a chain-like structure. Given an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its TT decomposition is expressed as:

$$\mathcal{X}(i_1, i_2, \ldots, i_N) = \mathbf{G}_1(i_1)\mathbf{G}_2(i_2)\ldots\mathbf{G}_N(i_N), \tag{8}$$

where each $\mathbf{G}_n(i_n) \in \mathbb{R}^{r_{n-1} \times r_n}$ is a core tensor, and $\{r_n\}$ are the TT ranks [31].

3.3.1 Properties and Applications

- TT decomposition scales efficiently with tensor order, making it ideal for handling extremely high-dimensional tensors.
- It has applications in compressing large machine learning models, solving high-dimensional partial differential equations, and efficient deep learning architectures [32].
- The low-rank structure enables reduced memory usage and faster computations compared to CP and Tucker decompositions [33].

3.3.2 Computation

The TT decomposition is typically computed using the Sequential SVD (TT-SVD) algorithm, which iteratively applies SVD to unfoldings of the tensor. This results in an efficient factorization with controlled ranks.

3.4 Comparison of Tensor Decomposition Methods

The choice of tensor decomposition method depends on the application and computational constraints. Table 1 summarizes key differences among CP, Tucker, and TT decompositions [34].

This section has introduced the fundamental tensor decomposition techniques, each of which plays a vital role in machine learning applications. In the next section, we explore how these decompositions are leveraged in various machine learning tasks [35].

Method	Rank Structure	Advantages	Challenges
CP	Single rank R	Simple, interpretable	Selecting R, computationally expensive
Tucker	Multi-mode ranks $(R_1, R_2, R_3, \dots)$	Flexible rank constraints	Core tensor size grows rapidly
TT	Chain of low-rank tensors	Efficient for high-order tensors	Requires sequential factorization

Table 1: Comparison of tensor decomposition methods.

4 Applications of Tensor Decomposition in Machine Learning

Tensor decomposition techniques have been widely adopted in machine learning due to their ability to efficiently represent and process high-dimensional data [36]. Their applications span various domains, including deep learning, dimensionality reduction, clustering, recommendation systems, and more [37]. In this section, we explore key areas where tensor decomposition plays a crucial role.

4.1 Dimensionality Reduction and Feature Extraction

High-dimensional data often contain redundant or irrelevant information, making dimensionality reduction essential for improving computational efficiency and model interpretability [38]. Tensor decomposition methods provide a natural way to reduce dimensionality while preserving the multi-way structure of the data [39].

- **Principal Component Analysis (PCA) for Tensors:** Tucker decomposition is often used as a multi-dimensional extension of PCA, where the core tensor captures significant variations while discarding less important components.
- **Sparse and Low-Rank Representations:** CP and Tucker decompositions facilitate feature extraction by identifying latent factors that best represent the data.
- **Multi-View Learning:** In scenarios with multiple data modalities (e.g., images, text, and audio), tensor decompositions help integrate and extract useful features from heterogeneous sources [40].

4.2 Deep Learning Model Compression

Deep neural networks (DNNs) are often over-parameterized, leading to excessive computational and memory requirements. Tensor decomposition provides an effective way to compress neural network models by factorizing weight tensors, thereby reducing redundancy while maintaining accuracy [41].

- **Low-Rank Factorization of Weight Tensors:** In convolutional neural networks (CNNs), tensor decomposition techniques such as CP and Tucker decomposition are used to compress convolutional kernels, reducing the number of parameters [42, 43].
- **Efficient Recurrent Neural Networks (RNNs):** Tensor Train decomposition helps reduce the complexity of weight matrices in RNNs, improving training and inference speed.
- **Tensor-Based Attention Mechanisms:** In transformer-based architectures, tensor decomposition can optimize attention layers, reducing memory and computation costs in large-scale models such as BERT and GPT [44, 45].

4.3 Recommendation Systems

Tensor decomposition methods are widely used in recommendation systems to model multi-way user-item interactions [46]. Unlike traditional matrix factorization techniques, tensor-based approaches can capture higher-order dependencies, leading to improved recommendation accuracy.

- **Personalized Recommendations:** CP decomposition is used to factorize user-item-time tensors, enabling dynamic and context-aware recommendations.
- **Cold-Start Problem Mitigation:** Tensor methods integrate multiple data sources (e.g., user preferences, product metadata, and social network interactions) to improve recommendations for new users and items [47].
- **Scalability Improvements:** Tensor Train decomposition reduces storage and computation costs, making large-scale recommendation systems more efficient.

4.4 Clustering and Unsupervised Learning

Tensor decomposition techniques are effective for discovering latent patterns in multi-dimensional data, making them valuable tools for clustering and unsupervised learning [48].

- **Tensor-Based K-Means Clustering:** Tucker decomposition can be used to extract lower-dimensional representations before applying clustering algorithms.
- **Anomaly Detection:** CP decomposition helps identify anomalies in multimodal data by revealing deviations from normal patterns.

- **Topic Modeling:** Tensor decompositions are used in natural language processing (NLP) to uncover latent topics in document-term-timestamp tensors [49].

4.5 Natural Language Processing (NLP)

Many NLP tasks involve multi-dimensional data representations, making tensor decomposition a natural choice for efficient computation and representation learning [50].

- **Word Embeddings and Contextual Representations:** Tensor decompositions improve word embedding models by reducing the dimensionality of word co-occurrence tensors [51].
- **Text Summarization:** Tensor-based factorization helps extract key information from large text corpora for automatic summarization [52].
- **Sentiment Analysis:** Multi-modal sentiment analysis benefits from tensor decomposition by fusing textual, audio, and visual data.

4.6 Graph Learning and Network Analysis

Graph-structured data is prevalent in social networks, biological networks, and knowledge graphs. Tensor decomposition methods provide efficient ways to model and analyze such data [53].

- **Community Detection:** CP and Tucker decompositions uncover hidden communities in large-scale social networks [54].
- **Link Prediction:** Tensor factorization is used to predict missing links in knowledge graphs, improving search and recommendation systems [55].
- **Graph Neural Networks (GNNs):** Tensor methods optimize GNN architectures by reducing memory and computational overhead.

4.7 Scientific Computing and Healthcare Applications

Tensor decomposition has been extensively used in scientific computing and healthcare to analyze complex multi-modal datasets.

- **Brain Imaging Analysis:** Functional MRI (fMRI) and electroencephalogram (EEG) data are naturally represented as tensors, where tensor decompositions help identify meaningful patterns in neural activity.

- **Genomic Data Analysis:** Tensor-based methods facilitate gene expression analysis, biomarker discovery, and disease classification [56].
- **Drug Discovery:** Tensor factorization is used to analyze chemical compound interactions, aiding in drug repurposing and discovery [57].

4.8 Challenges and Future Directions

While tensor decomposition has demonstrated significant advantages in machine learning, several challenges remain:

- **Computational Complexity:** Large-scale tensor decompositions require efficient algorithms and hardware acceleration techniques to handle high-dimensional data [58].
- **Rank Selection:** Choosing the appropriate rank for tensor decomposition remains a non-trivial problem, often requiring heuristics or domain-specific knowledge [59].
- **Scalability:** As datasets continue to grow, scalable and distributed tensor decomposition methods are needed for real-time and big data applications [60].

This section has outlined the broad impact of tensor decomposition in machine learning applications. In the next section, we discuss optimization techniques and computational strategies for efficient tensor decomposition [61].

5 Optimization Techniques and Computational Strategies for Tensor Decomposition

Efficient computation of tensor decompositions is crucial for their practical applications in machine learning. Given the large size and high dimensionality of real-world tensors, naive implementations can be computationally expensive and memory-intensive [62]. This section explores key optimization techniques and computational strategies for tensor decomposition, including gradient-based optimization, alternating minimization methods, randomized techniques, and hardware acceleration [37].

5.1 Gradient-Based Optimization Methods

Gradient-based optimization techniques are widely used for computing tensor decompositions by minimizing reconstruction error or maximizing statistical likelihood [63]. These methods are particularly useful for large-scale and sparse tensors [64].

5.1.1 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a widely used optimization method for tensor factorization, particularly in large-scale settings [65]. Given a loss function $\mathcal{L}(\mathcal{X}, \hat{\mathcal{X}})$ that quantifies the difference between the original tensor $\mathcal{X}$ and its decomposition $\hat{\mathcal{X}}$, the parameters are updated iteratively as:

$$\Theta_{t+1} = \Theta_t - \eta \nabla_\Theta \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}), \tag{9}$$

where Θ represents the factor matrices or core tensors, η is the learning rate, and $\nabla_\Theta \mathcal{L}$ is the gradient of the loss function [66].

5.1.2 Adaptive Optimization Methods

Variants of SGD such as Adam, Adagrad, and RMSprop improve convergence speed and stability by using adaptive learning rates. These methods are particularly effective for decomposing tensors in deep learning applications [67].

5.2 Alternating Least Squares (ALS)

Alternating Least Squares (ALS) is a widely used optimization technique for tensor decomposition, particularly for CP and Tucker decompositions. In ALS, the optimization problem is solved iteratively by fixing all but one factor matrix and solving a least-squares problem for the remaining factor.

5.2.1 ALS for CP Decomposition

For a given CP decomposition, ALS iteratively updates each factor matrix while keeping others fixed:

$$\mathbf{A} = \arg\min_{\mathbf{A}} \|\mathcal{X}_{(1)} - (\mathbf{B} \odot \mathbf{C})\mathbf{A}^T\|_F^2, \tag{10}$$

where $\mathcal{X}_{(1)}$ is the mode-1 unfolding of tensor $\mathcal{X}$, and $\odot$ represents the Khatri-Rao product [56].

5.2.2 ALS for Tucker Decomposition

In Tucker decomposition, ALS is used to iteratively update the core tensor and factor matrices:

$$\mathbf{U}_i = \arg\min_{\mathbf{U}_i} \|\mathcal{X} \times_i \mathbf{U}_i^\dagger - \mathcal{G}\|_F^2, \tag{11}$$

where $\mathcal{G}$ is the core tensor, and $\times_i$ denotes mode-i multiplication [68].

5.3 Randomized Tensor Decomposition

Randomized algorithms offer an efficient alternative to deterministic tensor decomposition methods, providing approximate solutions with reduced computational complexity.

5.3.1 Randomized SVD for Tensor Decomposition

Randomized Singular Value Decomposition (SVD) is used in Tucker and CP decompositions to accelerate factorization by projecting high-dimensional tensors into lower-dimensional subspaces:

$$\mathbf{Q} = \mathcal{X}\mathbf{\Omega}, \tag{12}$$

where $\mathbf{\Omega}$ is a randomly generated matrix, and $\mathbf{Q}$ captures a low-rank approximation of $\mathcal{X}$.

5.3.2 Sketching-Based Methods

Sketching techniques, such as CountSketch and TensorSketch, reduce memory and computational requirements by compressing tensors before applying decomposition algorithms [69].

5.4 Hardware Acceleration: GPU and Distributed Computing

Due to the high computational cost of tensor decomposition, leveraging modern hardware such as Graphics Processing Units (GPUs) and distributed computing frameworks can significantly improve performance [70].

5.4.1 GPU-Accelerated Tensor Decomposition

GPUs offer massive parallelization capabilities, making them well-suited for tensor decomposition [71]. Libraries such as TensorLy, cuTensor, and PyTorch provide GPU implementations for CP, Tucker, and Tensor Train decompositions [72].

5.4.2 Distributed Tensor Computation

Large-scale tensor decomposition can be efficiently handled using distributed frameworks such as Apache Spark, Dask, and MPI-based implementations. Parallel ALS and distributed SGD methods enable scalable computation for high-dimensional tensors [73].

5.5 Challenges and Open Problems

Despite significant progress, several challenges remain in optimizing tensor decomposition methods:

- **Scalability:** Handling extremely large and sparse tensors requires further advancements in parallel and distributed algorithms [74].
- **Convergence Issues:** ALS methods may suffer from slow convergence or get stuck in local minima, requiring better initialization strategies.
- **Automated Rank Selection:** Choosing the optimal tensor rank remains an open problem, necessitating adaptive rank selection techniques [75].

This section has provided an overview of optimization techniques and computational strategies for tensor decomposition [76]. In the next section, we explore real-world case studies demonstrating the impact of tensor decomposition in machine learning applications [77].

6 Case Studies and Real-World Applications of Tensor Decomposition

Tensor decomposition has demonstrated its effectiveness across various domains, from natural language processing to healthcare and scientific computing [78]. In this section, we present several real-world case studies where tensor decomposition techniques have played a crucial role in advancing machine learning applications [79].

6.1 Case Study 1: Tensor-Based Topic Modeling for Document Analysis

6.1.1 Problem Statement

Traditional topic modeling techniques, such as Latent Dirichlet Allocation (LDA), rely on matrix factorization and may fail to capture higher-order dependencies in textual data. Tensor decomposition provides a more structured way to model word co-occurrence patterns over multiple contexts, such as time, authorship, or geographic location.

6.1.2 Approach

A third-order tensor is constructed where each entry represents the frequency of a word in a document across different time periods [80]. CP decomposition is applied to extract latent topics, with each rank-one component representing a meaningful topic-word distribution [81].

6.1.3 Results

The tensor-based approach improves topic coherence and provides insights into the evolution of topics over time [82]. Compared to LDA, tensor decomposition leads to more interpretable topics and better separation between distinct themes.

6.2 Case Study 2: Tensor Decomposition in Recommender Systems

6.2.1 Problem Statement

Traditional recommendation algorithms, such as matrix factorization, only consider user-item interactions, neglecting additional contextual information (e.g., time, location, or social influence) [83]. Tensor decomposition extends recommendation models by incorporating multi-way interactions.

6.2.2 Approach

A user-item-context tensor is constructed, where the third dimension represents additional contextual factors [84]. CP decomposition is employed to factorize this tensor, identifying latent factors that influence user preferences [85].

6.2.3 Results

The tensor-based model outperforms matrix-based methods in predicting user preferences, especially in cold-start scenarios [86]. It enhances personalization by capturing contextual influences on user behavior [87].

6.3 Case Study 3: Tensor-Based Compression in Deep Learning

6.3.1 Problem Statement

Deep neural networks, especially large-scale transformer models, contain billions of parameters, making them computationally expensive for deployment on edge devices [88]. Tensor decomposition offers a way to reduce model size while preserving performance [89].

6.3.2 Approach

Weight tensors of convolutional and fully connected layers are factorized using Tucker decomposition, reducing redundancy in parameter representations. The decomposed tensors are used to reconstruct approximate weight matrices for inference.

6.3.3 Results

The compressed models achieve up to 50% reduction in storage requirements with minimal loss in accuracy [90]. Tensor decomposition enables real-time deployment of deep learning models on resource-constrained devices [91].

6.4 Case Study 4: Tensor-Based Brain Signal Analysis in Healthcare

6.4.1 Problem Statement

Electroencephalography (EEG) and functional Magnetic Resonance Imaging (fMRI) generate high-dimensional brain activity data. Traditional analysis techniques struggle to extract meaningful patterns from multi-channel signals.

6.4.2 Approach

EEG data is represented as a three-way tensor (channels $\times$ time $\times$ trials), and Tucker decomposition is applied to extract interpretable brain activity patterns.

The decomposition identifies key frequency components and their temporal evolution [92].

6.4.3 Results

Tensor decomposition improves the classification of neurological disorders such as epilepsy and Alzheimer's disease [93]. It helps in identifying biomarkers that are not easily detectable using standard machine learning techniques [94].

6.5 Case Study 5: Tensor-Based Link Prediction in Knowledge Graphs

6.5.1 Problem Statement

Knowledge graphs store relational data in the form of triplets (subject, predicate, object) [95]. Predicting missing links in these graphs is essential for improving search engines and recommendation systems [96].

6.5.2 Approach

A knowledge graph is represented as a three-way tensor, where each slice corresponds to a relation type [97]. CP decomposition is used to extract latent entity representations, which are then leveraged for link prediction.

6.5.3 Results

Tensor-based models outperform traditional embedding-based methods by capturing higher-order relationships in knowledge graphs. The approach enhances reasoning and inference in knowledge-based systems [98].

6.6 Challenges and Future Prospects in Real-World Applications

While tensor decomposition has shown promising results in multiple domains, several challenges remain:

- **Scalability:** Handling extremely large tensors requires further research in distributed and parallelized tensor computation.
- **Noise Sensitivity:** Real-world data often contain noise and missing values, which can affect the accuracy of tensor decomposition methods.

- **Automated Model Selection:** Determining the optimal rank and decomposition strategy for different applications remains an open research problem [99].

Despite these challenges, tensor decomposition continues to evolve, with applications in cutting-edge technologies such as quantum computing, autonomous systems, and biomedical informatics [100]. The next section will discuss potential research directions and future advancements in tensor decomposition for machine learning [101].

7 Future Directions and Research Challenges in Tensor Decomposition for Machine Learning

Tensor decomposition has proven to be a powerful tool for representing and processing high-dimensional data in machine learning [102]. However, several open research challenges and emerging trends warrant further exploration [103]. This section highlights promising directions for future research in tensor decomposition, including algorithmic advancements, scalability, interpretability, and novel applications.

7.1 Scalability and Efficient Computation

One of the most pressing challenges in tensor decomposition is its computational complexity, particularly for large-scale, high-dimensional tensors [104]. Several approaches can improve scalability:

- **Parallel and Distributed Tensor Decomposition:** Designing scalable tensor factorization methods that leverage distributed computing frameworks, such as Apache Spark and TensorFlow, will be crucial for handling massive datasets [105].
- **Low-Rank Approximation Methods:** Developing adaptive rank selection algorithms can reduce computational overhead by dynamically adjusting tensor rank during factorization [106].
- **Hardware-Accelerated Computation:** Optimizing tensor decomposition algorithms for Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) can significantly improve performance.

7.2 Automated Rank Selection and Model Order Determination

Choosing the appropriate rank for tensor decomposition remains a fundamental challenge [107]. Overestimating rank leads to computational inefficiency, while underestimating it results in loss of useful information [108]. Future research should focus on:

- **Bayesian and Probabilistic Approaches:** Bayesian tensor decomposition models can incorporate prior knowledge to infer optimal rank automatically [109].
- **Information-Theoretic Methods:** Techniques based on minimum description length (MDL) and Bayesian Information Criterion (BIC) can help determine the best tensor rank [110].
- **Data-Driven Adaptive Methods:** Developing learning-based approaches that dynamically adjust rank based on data complexity.

7.3 Tensor Decomposition for Explainable AI (XAI)

As AI models grow in complexity, interpretability becomes increasingly important [111]. Tensor decomposition can enhance model explainability by:

- **Identifying Latent Factors:** Decomposing neural network weight tensors can reveal underlying structures in learned representations.
- **Feature Attribution:** Tensor-based methods can help attribute model decisions to specific features in multi-modal data.
- **Uncertainty Quantification:** Probabilistic tensor decompositions can provide confidence estimates for model predictions.

7.4 Integration with Deep Learning and Neural Networks

The synergy between tensor decomposition and deep learning presents exciting research opportunities [112]. Key areas of integration include:

- **Tensorized Neural Networks:** Replacing traditional weight matrices with tensor representations in neural networks to improve efficiency and generalization [13, 113].

- **Hybrid Tensor-Deep Learning Models:** Combining tensor factorization with deep learning architectures, such as transformers and graph neural networks (GNNs), for improved performance in structured data.
- **Continual and Lifelong Learning:** Exploring tensor-based memory models for lifelong learning applications where knowledge retention and transfer are critical.

7.5 Robust Tensor Decomposition for Noisy and Incomplete Data

Real-world datasets often contain missing values, noise, and outliers, which can degrade tensor decomposition performance [114]. Future research should explore:

- **Robust Tensor Factorization:** Developing algorithms resilient to noise and missing data, such as robust CP and Tucker decompositions [115].
- **Tensor Completion Methods:** Leveraging low-rank tensor completion techniques to recover missing entries in multi-modal data.
- **Hybrid Statistical-ML Approaches:** Combining probabilistic modeling with machine learning techniques for improved robustness.

7.6 Emerging Applications of Tensor Decomposition

Beyond traditional machine learning applications, tensor decomposition is poised to impact several emerging fields:

- **Quantum Computing:** Tensor networks provide efficient representations of quantum states, facilitating advancements in quantum machine learning.
- **Autonomous Systems:** Tensor decomposition can improve decision-making and sensor fusion in robotics and autonomous vehicles.
- **Biomedical Informatics:** Multi-omics data analysis and personalized medicine can benefit from tensor-based feature extraction and integration [116].
- **Social Network Analysis:** Tensor models can uncover complex relationships in large-scale social graphs, enhancing link prediction and community detection [117].

7.7 Challenges and Open Problems

While tensor decomposition has achieved significant progress, several challenges remain:

- **Theoretical Understanding:** A deeper theoretical understanding of tensor decomposition's convergence properties and uniqueness conditions is needed [118].
- **Scalability Trade-offs:** Balancing efficiency, accuracy, and interpretability in large-scale tensor applications remains an open problem.
- **Generalization Across Domains:** Adapting tensor decomposition methods to different application domains with minimal manual tuning is an ongoing challenge [119].

7.8 Conclusion and Future Outlook

Tensor decomposition continues to evolve as a fundamental tool in machine learning, with numerous opportunities for improvement and expansion. Advances in scalable computation, rank selection, deep learning integration, and robustness will shape the future of tensor decomposition research. As emerging applications demand more efficient and interpretable machine learning models, tensor methods are expected to play an increasingly important role in next-generation AI systems [120]. In summary, addressing the computational and theoretical challenges outlined in this section will drive further adoption of tensor decomposition across diverse scientific and industrial domains [121]. The intersection of tensor methods with deep learning, quantum computing, and explainable AI presents exciting possibilities for future research and innovation.

8 Conclusion

Tensor decomposition has emerged as a powerful mathematical framework for analyzing high-dimensional data in machine learning. By extending traditional matrix factorization techniques to multi-way data structures, tensor methods have enabled significant advancements in various domains, including natural language processing, recommender systems, deep learning model compression, biomedical informatics, and knowledge graph reasoning.

In this paper, we explored the fundamental concepts of tensor decomposition, including CP, Tucker, and Tensor Train decompositions, and discussed their applications in machine learning. We examined optimization techniques for efficient tensor factorization, highlighting gradient-based methods, alternating least squares,

randomized approaches, and hardware-accelerated implementations. Through case studies, we demonstrated the impact of tensor decomposition on real-world problems, showcasing its ability to extract latent patterns, improve model efficiency, and enhance interpretability.

Despite its numerous advantages, tensor decomposition presents several challenges, such as scalability, rank selection, robustness to noise, and integration with deep learning. As research in tensor methods continues to evolve, future advancements will focus on improving computational efficiency, developing automated rank selection techniques, and exploring novel applications in emerging fields such as quantum computing and autonomous systems.

In conclusion, tensor decomposition remains a crucial tool in machine learning, offering promising directions for both theoretical research and practical applications. As data continues to grow in complexity, tensor methods will play an increasingly vital role in shaping the future of AI and large-scale data analysis.

References

[1] Juan Carrasquilla, Giacomo Torlai, Roger G Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

[2] Zi Cai and Jinguo Liu. Approximating quantum many-body wave functions using artificial neural networks. *Physical Review B*, 97(3):035116, 2018.

[3] A. V. Uvarov, A. S. Kardashin, and J. D. Biamonte. Machine learning phase transitions with a quantum processor. *Phys. Rev. A*, 102:012415, Jul 2020. doi: 10.1103/PhysRevA.102.012415. URL `https://link.aps.org/doi/10.1103/PhysRevA.102.012415`.

[4] Justin A Reyes and E Miles Stoudenmire. Multi-scale tensor network architecture for machine learning. *Machine Learning: Science and Technology*, 2 (3):035036, 2021.

[5] Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. *NeurIPS*, 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on t classification. In *ICCV*, 2015.

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

[8] Chuan-Yung Tsai, Andrew M Saxe, and David Cox. Tensor switching networks. In *NeurIPS*, pages 2038–2046, 2016.

[9] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] S. Sachdev. Viewpoint: Tensor networks—a new tool for old problems. *Physics*, 2:90, 2009. doi: 10.1103/Physics.2.90.

[13] Dian Wu, Riccardo Rossi, Filippo Vicentini, and Giuseppe Carleo. From tensor network quantum states to tensorial recurrent neural networks. *arXiv preprint arXiv:2206.12363*, 2022.

[14] IV Oseledets, S Dolgov, V Kazeev, D Savostyanov, O Lebedeva, P Zhlobich, T Mach, and L Song. TT-toolbox, 2016.

[15] RA HARSHMAN. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[16] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, 2016.

[17] Ingrid Russell. Neural networks module. *Neural Networks Module*, 2012.

[18] Yujun Zhou, Erin Lentz, Hope Michelson, Chungmann Kim, and Kathy Baylis. Machine learning for food security: Principles for transparency and usability. *Applied Economic Perspectives and Policy*, 44(2):893–910, 2022.

[19] Tianqi Chen, Mu Li, and et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274, 2015*, 2015.

[20] Qiuchi Li, Benyou Wang, and Massimo Melucci. CNM: An interpretable complex-valued network for matching. In *NAACL*, 2019.

[21] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32, 2019.

[22] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[23] Shaowu Chen, Jiahao Zhou, Weize Sun, and Lei Huang. Joint matrix decomposition for deep convolutional neural networks compression. *arXiv preprint arXiv:2107.04386*, 2021.

[24] Mark van Breugel, Ivan Rosa e Silva, and Antonina Andreeva. Structural validation and assessment of AlphaFold2 predictions for centrosomal and centriolar proteins and their complexes. *Communications Biology*, 5(1):1–10, 2022.

[25] Farhana Sultana, A Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. *arXiv preprint arXiv:1905.03288*, 2019.

[26] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3 (3):218–229, 2021.

[27] Alwin Stegeman and Pierre Comon. Subtracting a best rank-1 approximation may increase tensor rank. *Linear Algebra and its Applications*, 433(7):1276–1300, 2010.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[29] Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. Towards multimodal sentiment analysis: harvesting opinions from the web. In *ICMI*, 2011.

[30] Ivan Glasser, Nicola Pancotti, and J Ignacio Cirac. Supervised learning with generalized tensor networks. *arXiv preprint arXiv:1806.05964*, page 50, 2018.

[31] Alexander Jahn and Jens Eisert. Holographic tensor network models and quantum error correction: a topical review. *Quantum Science and Technology*, 6(3):033002, jun 2021. doi: 10.1088/2058-9565/ac0293. URL `https://doi.org/10.1088/2058-9565/ac0293`.

[32] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[33] Pierre Comon. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, 2014.

[34] Maolin Wang, Chenbin Zhang, Yu Pan, Jing Xu, and Zenglin Xu. Tensor ring restricted Boltzmann machines. In *IJCNN*, 2019.

[35] JD Explore Academy. Tensor-network enhanced distributed quantum, 2022. URL `https://github.com/JDEA-Quantum-Lab/TeD-Q`.

[36] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, 1994.

[37] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on signal processing*, 65(13):3551–3582, 2017.

[38] Jens Eisert. Entanglement and tensor network states. *arXiv preprint arXiv:1308.3318*, 2013.

[39] G. Venkatesh, E. Nurvitadhi, and D. Marr. Accelerating deep convolutional networks using low-precision and sparsity. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2861–2865, March 2017. doi: 10.1109/ICASSP.2017.7952679.

[40] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(8):2220–2233, Aug 2017. doi: 10.1109/TVLSI.2017.2688340.

[41] Nir Ofek, Soujanya Poria, Lior Rokach, Erik Cambria, Amir Hussain, and Asaf Shabtai. Unsupervised commonsense knowledge enrichment for domain-specific sentiment analysis. *Cognitive Computation*, 8(3):467–477, 2016.

[42] Zenglin Xu, Feng Yan, and Yuan (Alan) Qi. Bayesian nonparametric models for multiway data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):475–487, 2015.

[43] Yassine Zniyed, Thanh Phuong Nguyen, et al. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024.

[44] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[45] Ling Liang, Jianyu Xu, Lei Deng, Mingyu Yan, Xing Hu, Zheng Zhang, Guoqi Li, and Yuan Xie. Fast search of the optimal contraction sequence in tensor networks. *IEEE Journal of Selected Topics in Signal Processing*, 15 (3):574–586, 2021.

[46] André Viebke, Suejb Memeti, Sabri Pllana, and Ajith Abraham. Chaos: a parallelization scheme for training convolutional neural networks on intel xeon phi. *The Journal of Supercomputing*, pages 1–31, 2017.

[47] Chanwook Park, Sourav Saha, Jiachen Guo, Xiaoyu Xie, Satyajit Mojumder, Miguel A Bessa, Dong Qian, Wei Chen, Gregory J Wagner, Jian Cao, et al. Engineering software 2.0 by interpolating neural networks: unifying training, solving, and calibration. *arXiv preprint arXiv:2404.10296*, 2024.

[48] John Lighton Synge and Alfred Schild. *Tensor calculus*. University of Toronto Press, 2020.

[49] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[50] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[51] M. Alwani, H. Chen, M. Ferdman, and P. Milder. Fused-layer cnn accelerators. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016. doi: 10.1109/MICRO.2016.7783725.

[52] Y. Chen, T. Yang, J. Emer, and V. Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, June 2019. doi: 10.1109/JETCAS.2019.2910232.

[53] Jens M Melenk and Ivo Babuška. The partition of unity finite element method: basic theory and applications. *Computer methods in applied mechanics and engineering*, 139(1-4):289–314, 1996.

[54] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[55] Chunhua Deng, Siyu Liao, Yi Xie, Keshab K Parhi, Xuehai Qian, and Bo Yuan. Permdnn: Efficient compressed dnn architecture with permuted diagonal matrices. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 189–202. IEEE, 2018.

[56] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[57] Jiapeng Huang, Linghe Kong, Xiao-Yang Liu, Wenhao Qu, and Guihai Chen. A C++ library for tensor decomposition. In *IPCCC*, 2019.

[58] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Yu Gong, Miao Yin, Lingyi Huang, Chunhua Deng, and Bo Yuan. Algorithm and hardware co-design of energy-efficient lstm networks for video recognition with hierarchical tucker tensor decomposition. *IEEE Transactions on Computers*, 71(12):3101–3114, 2022.

[61] Maolin Wang, Zeyong Su, Xu Luo, Yu Pan, Shenggen Zheng, and Zenglin Xu. Concatenated tensor networks for deep multi-task learning. In *ICONIP*, 2020.

[62] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

[63] A Cayley. On the theory of groups as depending on the symbolic equation $\theta^n = 1$. *Philosophical Magazine, 7 (42): 40-47*, 1854.

[64] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[65] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[66] Xinyu Chen, Zhaocheng He, and Lijun Sun. A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation research part C: emerging technologies*, 98:73–84, 2019.

[67] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[68] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[69] Hannes Schulz and Sven Behnke. Deep learning. *KI-Künstliche Intelligenz*, 26(4):357–363, 2012.

[70] Claudius Hubig, Ian P McCulloch, Ulrich Schollwöck, and F Alexander Wolf. Strictly single-site DMRG algorithm with subspace expansion. *Physical Review B*, 91(15):155115, 2015.

[71] Guanglei Qi, Yanfeng Sun, Junbin Gao, Yongli Hu, and Jinghua Li. Matrix variate restricted Boltzmann machine. In *IJCNN*, 2016.

[72] Dario Amodei and Danny Hernandez. openai, May 2018. URL `https://openai.com/blog/ai-and-compute/`.

[73] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[74] Kazuhito Ito and Keshab K. Parhi. Determining the minimum iteration period of an algorithm. *Journal of VLSI signal processing systems for signal, image and video technology*, 11(3):229–244, Dec 1995. ISSN 0922-5773. doi: 10.1007/BF02107055.

[75] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[76] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy. A hardware–software blueprint for flexible deep learning specialization. *IEEE Micro*, 39(5):8–16, Sep. 2019. ISSN 1937-4143.

[77] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066, 2008.

[78] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li. Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 553–564, Feb 2017. doi: 10.1109/HPCA.2017.29.

[79] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pages 947–955, 2016.

[80] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706. ACM, 2007.

[81] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.

[82] Frank L Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

[83] David Cyganski and John A Orr. Applications of tensor theory to object recognition and orientation determination. *IEEE Trans. PAMI*, (6):662–673, 1985.

[84] Xilinx. Accelerating DNNs with Xilinx Alveo accelerator cards. Technical report, 2018.

[85] Farnaz Sedighin, Andrzej Cichocki, Tatsuya Yokota, and Qiquan Shi. Matrix and tensor completion in multiway delay embedded space using tensor train, with application to signal reconstruction. *IEEE Signal Processing Letters*, 27:810–814, 2020.

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

[87] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[88] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[89] Enrui Zhang, Ming Dao, George Em Karniadakis, and Subra Suresh. Analyses of internal structures and defects in materials using physics-informed neural networks. *Science advances*, 8(7):eabk0644, 2022.

[90] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *the Association for the Advance of Artificial Intelligence*, volume 14, pages 1112–1119, 2014.

[91] Tamara G Kolda and Brett W Bader. Matlab tensor toolbox. Technical report, Sandia National Laboratories (SNL), 2006.

[92] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[93] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[94] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *Proceedings of the ACM/IEEE International Symposium on Computer Architecture*, ISCA '17, pages 27–40, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4892-8. doi: 10.1145/3079856.3080254. URL `http://doi.acm.org/10.1145/3079856.3080254`.

[95] Ashley Milsted and Guifre Vidal. Geometric interpretation of the multi-scale entanglement renormalization ansatz. *arXiv preprint arXiv:1812.00529*, 2018.

[96] Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. Quantum entanglement in deep learning architectures. *Physical Review Letters*, 122 (6):065301, 2019.

[97] Joseph B Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18, 1989.

[98] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted Boltzmann machines and tensor network states. *Physical Review B*, 97(8):085104, 2018.

[99] Alexey Grigorevich Ivakhnenko. Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, (4):364–378, 1971.

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

[101] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[102] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, 1943.

[103] Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *CVPR*, 2019.

[104] Ye Liu and Michael K. Ng. Deep neural network compression by Tucker decomposition with nonlinear response. *Knowledge-Based Systems*, 2022.

[105] Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan Oseledets, Masashi Sugiyama, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

[106] Yankai Lin, Zhiyuan Liu, Xuan Zhu, Xuan Zhu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[107] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018.

[108] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. PAMI*, 35(8):1798–1828, 2013.

[109] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Trans. PAMI*, 37(9):1751–1763, 2015.

[110] Lukasz Cincio, Jacek Dziarmaga, and Marek M Rams. Multiscale entanglement renormalization ansatz in two dimensions: Quantum ising model. *Physical Review Letters*, 100(24):240603, 2008.

[111] Ilya Sutskever, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In *NeurIPS*, pages 1821–1828, 2009.

[112] Jonathan Romero and Alán Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, 2021.

[113] Yassine Zniyed, Thanh Phuong Nguyen, et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[114] Genevera Allen. Sparse higher-order principal components analysis. In *Artificial Intelligence and Statistics*, pages 27–36, 2012.

[115] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[116] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proc. ISWC*, pages 11–15, 2007.

[117] Fujiao Ju, Yanfeng Sun, Junbin Gao, Michael Antolovich, Junliang Dong, and Baocai Yin. Tensorizing restricted Boltzmann machine. *ACM Transactions on Knowledge Discovery from Data*, 13(3):1–16, 2019.

[118] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

[119] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[120] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326:96–192, January 2011. doi: 10.1016/j.aop.2010.09.012.

[121] K. K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, Hoboken, NJ, USA, 1999.