Neurocomputing 682 (2026) 133455

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Survey paper

A survey of latent factorization of tensor-based model compression: Algorithms, toolboxes and future directions

Yaping He[a], Hao Wu[a], Weibo Liu[b,*], Xin Luo[a]

[a] College of Computer and Information Science, Southwest University, Chongqing, 400715, China
[b] Department of Computer Science, Brunel University of London, Uxbridge, Middlesex, UB8 3PH, United Kingdom

ARTICLE INFO

Communicated by R. Yang

Keywords:
Latent factorization of tensor
Model compression
Resource-constrained devices
Convolutional neural network
Recurrent neural network
Transformer

ABSTRACT

Modern neural networks (NNs), while effective at learning representations from given samples and handling downstream pattern recognition tasks, typically contain tens to hundreds of millions of parameters. The growth in NN size motivates ongoing research on effective network compression with the purpose of reducing the computational burden without significantly sacrificing the model performance. It is especially critical when deploying NNs on resource-constrained devices where computation and storage efficiency are of high concern. A promising and currently popular solution to model compression is to replace the NN weight matrix with its low-rank tensor approximation, i.e., implementing an efficient latent factorization of tensors (LFT) process on the NNs parameters. Based on thorough investigations into the state-of-the-art LFT-based model compression methods, this survey 1) provides a comprehensive review of the latest research progress on LFT-based model compression methods for various NNs (e.g., Convolutional NNs, Recurrent NNs, and Transformers); 2) summarizes a number of widely-used LFT toolboxes; 3) evaluates LFT methods for model compression on a variety of main-stream NN backbones; and 4) discusses the development trends of LFT-based model compression techniques. This survey aims to provide a systematic and comprehensive overview of LFT-based model compression methods to artificial intelligence researchers and engineers, thereby promoting further research development in this crucial field.

1. Introduction

During the last few years, machine learning has seen rapid development driven by the maturity of artificial intelligence technologies. Particularly, neural networks (NNs) (especially deep NNs) have achieved remarkable success across various fields such as remote sensing image classification, anomaly detection, natural language processing (NLP), and many other tasks. Modern GPU and TPU clusters make it possible to train ultra-deep models with thousands of layers and billions of model parameters [1]. As the number of model parameters increases, the contradiction between model complexity and model performance becomes increasingly prominent. It is therefore of practical importance to achieve a proper balance between the model complexity and model performance when optimizing a deep learning model. It is worth pointing out that we account for both the model accuracy and resource consumption when deploying NNs on resource-limited devices such as mobile phones and tablets. Consequently, model compression is of great necessity [2].

Various model compression techniques (e.g., knowledge distillation (KD) [3–6], pruning [7,8], quantization [9,10], and Latent Factorization of Tensors (LFT) [11,12]) have been developed to reduce model size and computational cost. Among them, LFT stands out for its strong interpretability and favorable Compression Ratio (CR). The core idea of model compression is to decompose large pre-trained weight tensors or matrices into smaller factor tensors through low-rank approximation. With minimal fine-tuning, LFT can achieve competitive performance while significantly reducing model complexity. Therefore, the development of model compression methods based on LFT to reduce parameters and Floating-Point Operations per Second (FLOPs) holds significant promise. In this paper, we provide a comprehensive summary of existing LFT model compression methods, their structural characteristics, and representative application scenarios, which are presented in Tables 1–3, respectively.

Building upon the above summary of LFT techniques, we next survey the recent developments in model compression for three classic NN families (i.e., CNNs, RNNs, and Transformers), starting with CNNs. CNNs have become ubiquitous across diverse computer vision applications (e.g., 3D facial reconstruction [13], recognition tasks [14], object

* Corresponding author.
Email addresses: hyp15823653754@email.swu.edu.cn (Y. He), haowuf@swu.edu.cn (H. Wu), Weibo.Liu2@brunel.ac.uk (W. Liu), luoxin@swu.edu.cn (X. Luo).

https://doi.org/10.1016/j.neucom.2026.133455
Received 6 November 2025; Received in revised form 25 January 2026; Accepted 24 March 2026
Available online 25 March 2026

Table 1
Summary of model compression methods.

	Category	Subcategory	Detailed models/techniques	Section
Model compression	Compression methods	TSVD	TSVD-CNNs [11,12], OR-CNNs [32], TSVD-free [33], N-CNNs [34], TT-TSVD [35], LRPET [36], LoRKD [37]	III (A)
			S-RNNs [38], LSTM-TSVD [39]	
			NNs-Transformers [40], MPO-Transformers [41], LASER [42], DRONE [43], HASSLE-free [44]	
		CP	MSI-CNNs [45], LP-CNNs [46], F-CNNs [47], I-CNNs [48]	III (B)
			KCP-RNNs [49], HO-LSTM [50]	
			CP-KD-Transformer [51]	
		TK	HALOC-CNNs [52], CORING [53], TK-CNNs [54–57], BATUDE-CNNs [58], EAOA [59], ARSR-CNNs [60], HCF-CNNs [61]	III (C)
			TK-GRU [62], TKDQ [63], HTK-LSTM [64]	
			KDLD-Transformers [65], LFT-SP-Transformer [66]	
		TR	TR-CNNs [67,68], R-CNNs [46], PSTRN [69]	III (D)
			TDL-LSTM [70]	
			TR-Transformer [71], TR-MHSA [72], TR-FFN [72]	
		TT	HOD-CNNs [73], TT-Conv [74,75], VBMF [76], TQ-CNNs [77], LS-CNNs [78], LP-CNNs [79], AD-CNNs [80], NAF-CNNs [81], 3DCNNs [82]	III (E)
			TT-RNNs [83,84], TT-GRU [83], TC-RNNs [85], ETTconv [86]	
			Transformers++ [87], TT-T3SRS [88], CLWM-Transformers [89], DSFormer-LRTC [90]	
	Training schemes	Mixed-precision training	Mixed-accuracy training [91]	IV (A)
		Initialization	Xavier initialization [92], Kaiming initialization [93], Yu initialization [94], weight selection [95], Spectral initialization [96]	IV (B)
		Search ranks	VBMF [76], PSTRN [69], EA [97], RL [97], OR-CNNs [32], BATUDE-CNNs [58], APD-PMC [98]	IV (C)
	Toolboxes	Popular toolboxes for tensor operations	Tensor Toolbox [99], Tensorlab [100], TDALAB [101], Numpy [102], TensorFlow [103], TensorD [104], Tensorly [105], TensorNetwork [106], MXNet [107], TenDeC++ [108], ITensor [109]	V (A)
		Toolboxes for deep networks	Tensorly [105], TedNet [110]	V (B)

Table 2
Compact overview of compressible components, typical rank ranges, and characteristics of different LFT methods across CNNs, RNNs, and transformers.

LFT method	CNN components	RNN components	Transformer components	Rank range	Advantages	Limitations
TSVD	FC, 1×1 Conv	Input-to-Hidden	QKV, FFN	32–256	Simple, stable	Weak for higher-order tensors
CP	Conv kernels	Weight matrices	QKV	5–20	Very high compression	Training instability
TK	Conv kernels	GRU/LSTM matrices	FFN	10–60	Balanced accuracy vs. compression	Core tensor cost
TT	FC, large matrices	Sequential structures	Embeddings, FFN	8–64	Memory efficient	Hard to tune ranks
TR	Conv, FC	RNN full modules	MHSA, FFN	4–32	Strong expressive power	Higher implementation cost

detection [15–18], industrial visual inspection [19], image segmentation [20], and multi-agent perception [21]) due to their exceptional performance. Substantial research has demonstrated that the parameters of NNs exhibit high redundancy. Only a small subset of weights is sufficient to achieve high prediction accuracy, indicating that the NNs are over-parameterized [22]. LFT can be effectively applied to compress both the Fully Connected (FC) and convolutional layers in CNNs. In FC layers, Truncated Singular Value Decomposition (TSVD) and Canonical Polyadic (CP) decomposition have been commonly used to reduce the size of weight matrices while preserving essential feature representations [23]. In convolutional layers, TSVD [24], CP [25], Tucker (TK) [26], Tensor Train (TT) [27], and Tensor Ring (TR) [28] decompositions have been developed to exploit the multi-dimensional structure of convolutional kernels. Among these methods, TK, TT, and TR decompositions provide stronger tensor representation capabilities and can achieve a more favorable balance between CR and model accuracy.

Designed for sequential data analysis, RNNs (such as simple RNNs, LSTM, and GRU) often contain a large number of redundant parameters [29–31]. Various model compression methods (e.g., LFT, TSVD, CP, TK, TT, and TR decompositions) have been widely applied to compress both the hidden-to-hidden and input-to-hidden weight matrices in NLP, computer vision, and other tasks. With the exception of TSVD, the above-mentioned methods require reshaping the weight matrices into higher-order tensors before decomposition. Note that TT and TR decompositions preserve the dependencies between inputs and hidden states because they have a unique structured representation, and they maintain the sequential information inherent in RNNs and their variants. This characteristic makes TT and TR particularly suited for compressing RNNs-based architectures while retaining model performance.

The recent popular NN model, Transformer, has been successfully applied to NLP and computer vision tasks [111–114]. As the backbone of large language models, Transformers contain a great many parameters. To reduce the computational cost, TSVD has been adopted to compress weight matrices, as it provides a direct approach for matrix-based parameters. More recent studies have explored LFT methods (including CP, TK, Tensor TT, and TR decompositions) to further reduce the model size of Transformer-based approaches. Tensor-based methods

Table 3
Application scenarios and features of compression methods.

Method	Scenario	Feature
TSVD	Mobile/edge, small NLP [44]	Simple, low compression
CP	Image/video tasks [45]	High compression, less stable
TK	High-precision CV [52]	Balanced acc., compression
TT	Large NLP, recommender [67]	Efficient, memory-saving
TR	Large recommender/graph [73]	More expressive, complex

benefit from stronger representation capabilities than matrix decomposition, resulting in more effective compression performance. Notably, TT and TR decompositions, similar to their use in RNNs, are also well-suited for sequence modeling tasks due to their capability of preserving the dependencies inherent in sequential data while maintaining model performance.

Several surveys have provided valuable overviews of the applications of LFT for model compression, offering insights into the fundamental principles and methodological categories of LFT [115,116]. However, existing surveys commonly exhibit two critical limitations.

1. First, most of them focus primarily on the conceptual and algorithmic aspects of LFT, with limited systematic discussion of the underlying tensor libraries and stable training strategies that are essential for practical and reproducible implementations. As a result, practitioners often face difficulties in selecting appropriate tools and ensuring reliable training behavior when deploying different LFT methods in real-world scenarios.
2. Second, there is a notable lack of standardized benchmark experiments in the current literature. The absence of consistent baseline evaluations across representative methods makes it challenging to compare the intrinsic performance of LFT-based methods, thereby hindering the establishment of clear performance references for method selection and application.

To address the above-mentioned gaps, this survey offers a methodologically grounded and implementation-oriented perspective on LFT-based model compression. Specifically, we systematically review commonly-used tensor calculation libraries and summarize practical strategies for achieving stable and efficient NN model training, providing researchers and engineers with actionable guidance for further implementation. Furthermore, a set of standardized benchmark experiments is designed and conducted to comprehensively evaluate the baseline performance of representative LFT methods on typical models and tasks. By bridging the gaps between methodological discussion and empirical evaluation, this survey provides a solid foundation for future research and facilitates broader and more reliable applications of LFT in model compression.

The main contributions of this review paper are outlined as follows:

1. **Practical insights for researchers and engineers:** LFT-based model compression methods for three common NN families (including CNNs, RNNs, and Transformers) are highlighted in terms of weight decomposition and sub-layer reconstruction, which help readers understand the latest advances in model compression techniques and provide guidance for further optimization and experimental exploration.
2. **Overview of LFT toolboxes and libraries:** A comprehensive summary of relevant LFT tools and libraries is provided to guide researchers in selecting appropriate model compression methods or in designing novel LFT-based compression schemes tailored to specific requirements.
3. **Performance analysis of existing LFT methods:** A unified qualitative-quantitative evaluation of five LFT methods (i.e., TSVD, CP, TK, TR, and TT) across four datasets is presented, which narrows the gap between methodological discussion and empirical evaluation.
4. **Discussions on challenges and potential research directions:** Existing challenges and prospective topics in LFT-based model compression are examined and outlined, which guide subsequent algorithm design, benchmarking protocols, and resource-aware implementation.

Section 2 elaborates on prevailing NNs. Section 3 provides a review of state-of-the-art model compression methods. In Section 4, we summarize training methodologies. Section 5 introduces commonly used LFT toolboxes. Section 6 presents the empirical studies. In Section 7, challenges and future research topics are discussed. Finally, conclusions are drawn in Section 8.

2. Decomposable NNs

With the advent of deep networks, model compression based on LFT has been given a lease of vitality. Its core is to estimate and decompose the weight information by utilizing matrix or LFT techniques. In this section, we review some of the basic models commonly used for network compression.

2.1. CNNs

Within CNNs, the primary components consist of convolutional layers and FC layers, collectively accounting for 95% of the network's parameters [22]. Therefore, it is a promising solution to reduce redundant parameters and thus contribute to a more effective model. To show more details of compression clearly, the definitions of convolutional and FC layers are provided in this section.

2.1.1. Convolutional layer

Convolutional layers map an input tensor $\mathbf{X} \in \mathbb{R}^{M\times W\times H}$ into an output tensor $\mathbf{Y} \in \mathbb{R}^{N\times W'\times H'}$ with a fourth-order kernel tensor $\mathbf{W} \in \mathbb{R}^{D\times D\times M\times N}$. Convolutional computation is defined as:

$$\mathbf{Y}(w', h', s) = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{m=1}^{M} \mathbf{W}(n, m, i, j)\mathbf{X}(w, h, m), \tag{1}$$

where $w = w' + i - 1$ and $h = h' + j - 1$. The computational structure of the original convolution is illustrated in Fig. 1.

2.1.2. FC layer

The computation of the FC layer is defined as:

$$\mathrm{Y} = \mathrm{W}\boldsymbol{x} + \boldsymbol{b}, \tag{2}$$

In contrast to convolutional layers, the weights of an FC layer are initially represented as a second-order matrix, which is then transposed or

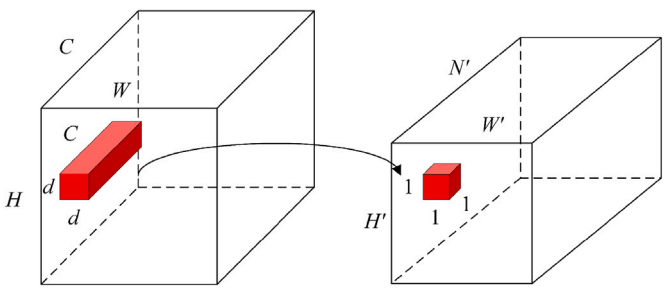


Fig. 1. Schematic diagram of the standard convolution operation. An input tensor $\mathbf{X} \in \mathbb{R}^{M\times W\times H}$ is convolved with a kernel tensor $\mathbf{W} \in \mathbb{R}^{D\times D\times m\times N}$ to produce an output tensor $\mathbf{Y} \in \mathbb{R}^{N\times W'\times H'}$. Each element in the output tensor is computed as the sum of the element-wise multiplication between a kernel filter and the corresponding local region in the input tensor, as defined in (1).

reshaped into a fourth-order tensor as needed. Specifically, the CP decomposition necessitates the decomposition of tensors of order three or higher, whereas the weight matrix of the FC layer remains a second-order matrix. Typically, matrices are reshaped from $\mathbf{W} \in R^{M\times N}$ to $\mathbf{W} \in \mathbb{R}^{1\times 1\times M\times N}$ before being decomposed as a convolutional layer.

2.2. RNNs, LSTMs, GRUs

RNNs constitute a class of NNs specifically engineered to handle sequential data. A fundamental characteristic of RNNs is their capacity to capture temporal information within sequences and sustain memory states within the network, facilitating the handling of tasks reliant on temporal order. RNNs find extensive application in domains including speech recognition, and time series analysis. Among other NNs architectures commonly employed for modeling time-series data are LSTM networks and GRU.

2.2.1. Vanilla RNNs

Generally, the input sequence is represented as $\boldsymbol{x} = (x_1, \ldots, x_T)$, the sequence of hidden vectors as $\boldsymbol{h} = (h_1, \ldots, h_T)$, and the output vector sequence as $\boldsymbol{y} = (y_1, \ldots, y_T)$. The standard RNNs at the t-th time step can be represented as:

$$\begin{cases} \boldsymbol{h}_t = f(\mathbf{W}_{xh}\boldsymbol{x}_t + \mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h), \\ \boldsymbol{y}_t = g(\mathbf{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y), \end{cases} \tag{3}$$

where $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, and $\mathbf{W}_{hy}$ denote weight matrices, $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are bias terms. g is the activation function, typically tanh or sigmoid is used. If it is a multi-categorization problem, g denotes the softmax function [117].

2.2.2. LSTM

LSTM is an advanced variant of RNNs for increasing long-term dependencies by adding some gating units [118]. The hidden layer values of the LSTM at time t are defined by:

$$\begin{cases} \boldsymbol{f}_t = \sigma(\mathrm{W}_f\boldsymbol{x}_t + \mathrm{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f), \\ \boldsymbol{i}_t = \sigma(\mathrm{W}_i\boldsymbol{x}_t + \mathrm{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \\ \boldsymbol{o}_t = \sigma(\mathrm{W}_o\boldsymbol{x}_t + \mathrm{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o), \\ \boldsymbol{g}_t = \tanh(\mathrm{W}_g\boldsymbol{x}_t + \mathrm{U}_g\boldsymbol{h}_{t-1} + \boldsymbol{b}_g), \\ \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t, \\ \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \end{cases} \tag{4}$$

where i_t, f_t, o_t, and c_t represent the input gates, forget gates, output gates, and memory cells, respectively. $\odot$ denotes the element-wise multiplication, and the large model size consumes more resources. The network structure of LSTM is shown in Fig. 2.

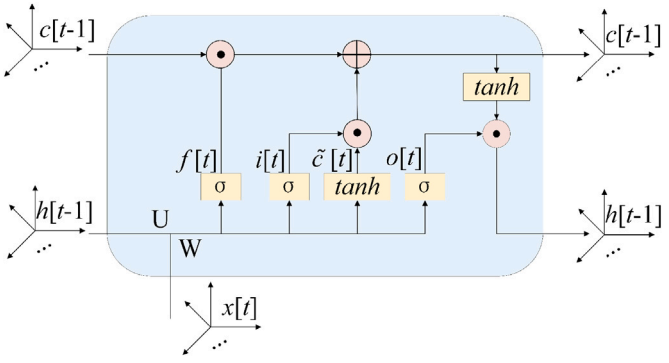


Fig. 2. The figure illustrates the basic components of an LSTM network, including the input gate, forget gate, output gate, and memory cell (cell state). The arrows indicate the data flow within the network and show how each gate processes the input through weighted sums and activation functions, influencing the output and updating the memory cell.

2.2.3. GRU

GRU is another variant of gated RNNs that has been shown to outperform advanced LSTM in certain tasks [119]. Two main differences exist between LSTM and GRU. Firstly, GRU does not segregate the hidden state from the storage unit. Secondly, it comprises only two gating layers: the reset gate and the update gate, as opposed to three in LSTM.

$$\begin{cases} \boldsymbol{r}_t = \sigma(\mathrm{W}_{xr}\boldsymbol{x}_t + \mathrm{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r), \\ \boldsymbol{z}_t = \sigma(\mathrm{W}_{xz}\boldsymbol{x}_t + \mathrm{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z), \\ \boldsymbol{h}_t = f(\mathrm{W}_{xh}\boldsymbol{x}_t + \mathrm{W}_{hh}(\boldsymbol{r}_t\boldsymbol{h}_{t-1}) + \boldsymbol{b}_h), \\ \boldsymbol{h}_t = (1 - z_t)\boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \boldsymbol{h}_t, \end{cases} \tag{5}$$

where $\boldsymbol{r}_t$ and $\boldsymbol{z}_t$ are the outputs of reset and update gates, $\tilde{h}_t$ is the candidate's hidden state, and h_t is the hidden state at the t-th. The update gate governs the amount of information from the preceding time step that should be preserved at the current time step, whereas the reset gate controls the degree to which previous information should be disregarded. GRU enhances the network's ability to capture long-term dependencies by modulating these gates while addressing the issue of vanishing gradients.

Generally, vanilla RNNs, LSTM, and GRU architectures primarily consist of FC layers rather than convolutional layers. This constitutes the primary source of model parameters and also serves as the primary obstacle to model inference.

2.3. Transformers

In recent years, Transformers have attracted considerable interest for their exceptional capacity to capture temporal dependencies [120]. Their performance excels at capturing long-range dependencies compared to RNNs or LSTM. Moreover, to further explore the potential of Transformers in other domains such as vision tasks, Vision Transformer [121] and Swin Transformer [122] have subsequently been developed and achieved outstanding results on benchmark leaderboards.

The core principle of Transformers is to capture the relationships between elements in the input sequence through a self-attention mechanism, without relying on the fixed positions of the elements. Attention is computed as follows:

$$Attention(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = softmax\left(\frac{\mathrm{QK}^T}{\sqrt{d}}\right)V, \tag{6}$$

with Q, K, and V being the representations of the query, key, and value vectors, respectively. The core of the Transformers comprises two parts: Multi-Head Self-Attention (MHSA) and Feed-Forward Network (FFN). Their structures are shown in Fig. 3.

2.3.1. MSHA

MHSA is a fundamental component of the Transformer model, tasked with capturing dependencies among various positions within the input sequence to enhance the model's expressive capability. Overall, MHSA enhances the model's flexibility in sequence processing by simultaneously computing multiple attention heads, enabling it to concentrate on information across various positions, thereby improving its ability to capture relationships and contexts within the sequences. It stands as a cornerstone of the Transformer's success, particularly in domains like. The MHSA is formulated as follows:

$$\begin{cases} MultiHead(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = Concat(head_1, \ldots, head_h)\mathrm{W}^O, \\ head_i = Attention(\mathrm{QW}_i^Q, \mathrm{KW}_i^K, VW_i^V), \end{cases} \tag{7}$$

where W_i^Q, W_i^K, W_i^V, and $\mathrm{W}_i^O \in \mathbb{R}^{d\times d}$ denote the query, key, value, and output matrices, respectively. The role of the projecting matrix W is to keep the input and output dimensions consistent. Let the weights at

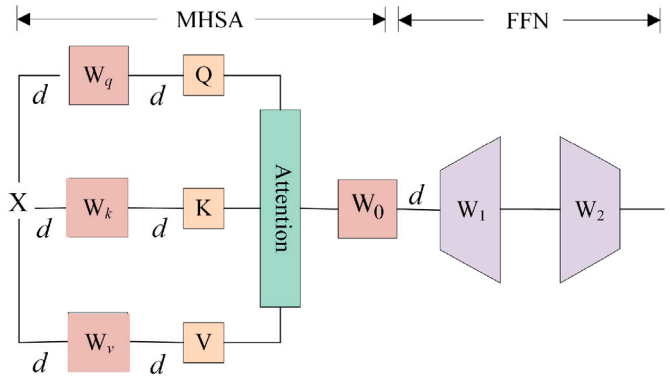


Fig. 3. A standard Transformer Block. The attention computation process includes calculating the Query, Key, and Value, followed by a dot-product to obtain the attention weights, which are then normalized using Softmax. The FFN is applied independently at each position, performing the same feedforward computation for data at each position.

positions s and t be $\text{scores}_{s,t}$ $(s,t=1,\ldots,m)$, where $\text{scores}_{s,t}=\frac{q_s\cdot k_t}{\sqrt{d_k}}$, then, the output x'_s at position s is:

$$\boldsymbol{x}'_s=\sum_{t=1}^{m}\frac{\exp(\text{score}_{s,t})}{\sum_{t'=1}^{m}\exp(\text{score}_{s,t})}v_t. \tag{8}$$

The computation of self-attention in MHSA is realized through a FC layer with distinct neurons. MHSA involves extensive matrix multiplication, resulting in a substantial rise in computation and storage overhead.

2.3.2. FFN

The FFN block, positioned after each attention layer, functions to perform nonlinear transformations and extract features. The intermediate process comprises matrix multiplication and activation operations. In essence, an FFN module comprises two FC layers.

$$\text{FFN}(x)=\text{GELU}\left(max(0,\boldsymbol{x}\text{W}_1+\boldsymbol{b}_1)\right)\text{W}_2+\boldsymbol{b}_2, \tag{9}$$

where W_1, W_2, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$ are the weights and biases of the FC layers in the FFN block, respectively. To simplify the notation, bias terms are excluded, and the FFN is rewritten as follows:

$$\text{FFN}(x)=\text{GELU}\left(max(0,\boldsymbol{x}\text{W}_{up})\right)\text{W}_{down}. \tag{10}$$

In some contexts, the FFN block is considered analogous to multi-head attention. W_{up} and W_{down} are partitioned into four $d\times d$ matrices, denoted as $\{\text{W}_{up}^{(i)}\}_{i=1}^{4}$ and $\{\text{W}_{down}^{(i)}\}_{i=1}^{4}$, respectively. The formulation of the FFN is given as follows:

$$\text{FFN}(\boldsymbol{x})=\sum_{i=1}^{4}\text{GELU}\left(\boldsymbol{x}\text{W}_{up}^{(i)}\right)\text{W}_{down}^{(i)}. \tag{11}$$

In essence, both MHSA and FFN are perceived as FC layers, entailing a substantial expenditure of computational resources during the training phase.

3. Review model compression methods

In this section, we present each specific approach in detail according to the taxonomy proposed in this paper, covering various model compression methods and their research progress. Related works are summarized in Table 1. As shown in Fig. 4, LFT-based compression mainly involves three stages: pre-training, parameter decomposition, and fine-tuning. In essence, the decomposition and fine-tuning steps perform representation learning on the data, enabling the model to capture informative features while reducing parameters.

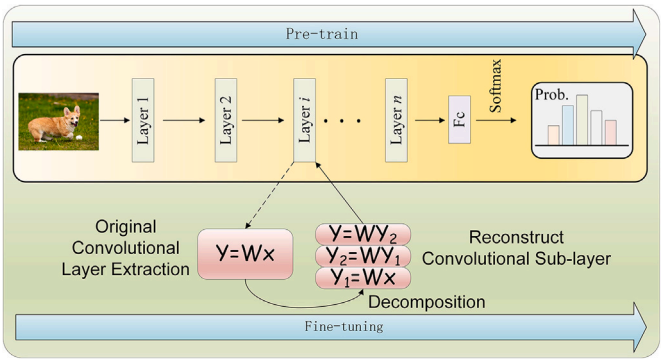


Fig. 4. Framework of low-rank decomposition-based compression. The framework consists of three main stages: pre-training, decomposition, and fine-tuning. Pre-training is used to capture data characteristics, decomposition compresses the model weights via low-rank factorization, and fine-tuning is applied to recover the performance of the compressed model.

3.1. TSVD

Given a matrix $\text{W}\in\mathbb{R}^{M\times N}$, where M and N denote the spatial dimensions of the weight matrix. The form of SVD of the matrix is defined as follows Yu and Bouganis [123]:

$$\text{W}=\text{USV}^{\text{T}}, \tag{12}$$

where $\text{U}\in\mathbb{R}^{M\times M}$, $\text{S}\in\mathbb{R}^{M\times N}$, $\text{S}\in\mathbb{R}^{N\times N}$. U and V denote the left and right singular matrices, both of which are orthogonal matrices. S is a diagonal matrix whose diagonal elements are singular values arranged in descending order.

Theoretical Derivation: The values under consideration can be approximated by retaining the top R dominant components:

$$\widetilde{\text{W}}=\widetilde{\text{U}}\widetilde{\text{S}}\widetilde{\text{V}}^{\text{T}}, \tag{13}$$

where $\widetilde{\text{U}}\in\mathbb{R}^{M\times R}$, $\widetilde{\text{S}}\in\mathbb{R}^{R\times R}$, $\widetilde{\text{V}}\in\mathbb{R}^{R\times N}$, and $R<\min\{M,N\}$.

In CNNs, TSVD is adopted to decompose the weight matrix.

$$\widetilde{\text{W}}=\widetilde{\text{U}}\widetilde{\text{S}}\widetilde{\text{V}}^{\text{T}}=(\widetilde{\text{U}}\widetilde{\text{S}})\widetilde{\text{V}}^{T}, \tag{14}$$

Consequently, an FC layer can be decomposed into two FC layers with fewer neurons each.

$$\begin{cases}\text{Z}^{\text{T}}=\text{X}^{\text{T}}(\widetilde{\text{U}}\widetilde{\text{S}}),\\ \text{Y}^{\text{T}}=\text{Z}^{\text{T}}\text{V}^{\text{T}},\end{cases} \tag{15}$$

where Z^{T} represents the middle layer. The number of parameters of the FC layer is reduced from the original MN to $MR+RN$ when the TSVD ranks are set to R.

In an FC layer, the CR is the same as the speed-up ratio (SR). Therefore, it can be defined as:

$$CR=SR=\frac{MN}{MR+RN}, \tag{16}$$

TSVD-CNNs: Building upon the theoretical foundation of TSVD CNNs (TSVD-CNNs), researchers have proposed a variety of improved methods to further enhance model compression performance. The existing approaches can be broadly divided into two categories:

- **Decomposition and Structural Optimization Methods:** The focus of this group lies in refining Singular Value Decomposition (SVD) and optimizing network structure to achieve greater model compactness and computational efficiency. Orthogonal-regularized CNNs (OR-CNNs) stabilize singular value decomposition, mitigating issues such as gradient vanishing and explosion [32]. Low-rank sparse decomposition of the weight matrix, combined with feature map reconstruction and TSVD-free

optimization, further enhances compression performance [33]. Incorporating nonlinear units after the decomposition kernel (N-CNNs) improves feature representation capability [34]. Tensor-train truncated SVD (TT-TSVD) leverages correlations between tensors to preserve data relevance while improving approximation quality [35].

- **Energy Redistribution and Knowledge Enhancement Methods:** The second group addresses gradient instability and strengthens knowledge representation. Projecting the weight matrix onto a low-rank manifold often reduces energy and leads to gradient vanishing. Low-Rank Projection with Energy Transfer (LRPET) redistributes the energy of the pruned singular values to the remaining ones instead of discarding it, preserving gradient stability [36]. The Low-Rank Knowledge Decomposition (LoRKD) framework introduces low-rank expert modules and efficient knowledge-separating convolutions to decompose domain knowledge in foundational medical models, achieving a balanced trade-off between computational cost and model accuracy [37].

TSVD-RNNs: In RNNs, relying solely on LFT has often failed to produce compressed models with strong performance. Research on TSVD for RNNs (TSVD-RNNs) has mainly focused on two methodological directions:

- **Decomposition and Structural Optimization Methods:** The first direction has emphasized structural refinement and computational efficiency. Sparsification RNNs (S-RNNs) combined with LFT have achieved significant reductions in computational overhead, model size, and execution time [38]. The joint use of sparsification and factorization techniques has led to more efficient parameter utilization while maintaining model capacity.
- **Knowledge Enhancement and Application-Oriented Methods:** The second direction has concentrated on improving model scalability and extending its applicability. KD-based training has enhanced generalization capability and compression performance. The LSTM TSVD (LSTM-TSVD) has removed noise and reduced memory consumption, enabling accurate prediction of remaining useful life in time series tasks [39]. Feature extraction from both time and frequency domains has provided additional support for efficient life cycle management in industrial applications.

TSVD-Transformers Higher-order tensor representations are challenging due to limited effective methods and unclear generality. Using tensor products of word embeddings increases the representational power of Transformers. In NLP, FC layers can contain millions of parameters, creating heavy computational demands. TSVD efficiently reduces parameters in MHSA and FFN while maintaining core functionality.

- **Decomposition and Precision Allocation Methods:** The first category has focused on reducing model complexity through LFT and adaptive precision strategies. A widely used approach has combined quantization or mixed-precision techniques with TSVD to achieve a balance between computational efficiency and model fidelity [40]. Precision allocation within each Transformer layer has been guided by sensitivity analysis, for example by evaluating the nuclear norm of attention maps, enabling the preservation of critical information.
- **Structured Factorization and Hybrid Compression Methods:** The second category has emphasized structured low-rank approximations and hybrid compression techniques for pre-trained large language models (LLMs). Matrix product operator (MPO)-based methods [41] and Layer-Selective Rank Reduction (LASER) [42] have achieved parameter compression without introducing additional components or requiring retraining. DRONE [43] and HASSLE-free [44] have adopted data-aware and hybrid sparse–low-rank strategies, improving reconstruction accuracy and accelerating inference in various NLP tasks.

3.2. CP

CP decomposition can factorize a tensor into the outer product of multiple latent factor matrices. For example, a third-order tensor format of CP decomposition can be defined as follows Chen et al. [124]:

$$\mathbf{T} \approx \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r, \tag{17}$$

where $\boldsymbol{a}_r$, $\boldsymbol{b}_r$, and $\boldsymbol{c}_r$ indicate latent factors from different orders, respectively. $\circ$ represents the outer product operation. Detailed expressions of each element $t_{i,j,k}$ can be obtained in $\mathbf{T}$.

$$t_{i,j,k} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}, \tag{18}$$

where $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$. The factor matrices are defined as the outer product of the vectors from the rank-one components: $\mathrm{A} = [a_1, a_2, \ldots, a_R]$, $\mathrm{B} = [b_1, b_2, \ldots, b_R]$, and $\mathrm{C} = [c_1, c_2, \ldots, c_R]$. For a third-order tensor $\mathbf{T}$, the weak upper bound on the maximal rank is given by:

$$rank(\mathbf{T}) \leq \min\{IJ, IK, JK\}, \tag{19}$$

when the rank is too large, it leads to minimal parameter reduction and poor CR. Conversely, a small rank can cause substantial precision loss that cannot be corrected, even with careful fine-tuning. Therefore, determining the optimal rank is vital to maintain an effective balance in model performance.

Theoretical Derivation: The explanation above clarifies the calculation process of LFT and convolution. The central question is how to integrate the two formulas. More specifically, the kernel tensor $\mathrm{W} \in \mathbb{R}^{D \times D \times M \times N}$ is approximated through a latent factor matrix. Its rank-R CP decomposition, after reshaping, is formulated as follows.

$$\mathbf{W}_{n,m,j,i} = \sum_{r=1}^{R} \mathrm{U}^{(1)}_{r,m} \mathbf{U}^{(2)}_{r,j,i} \mathrm{U}^{(3)}_{n,r}, \tag{20}$$

the components $\mathrm{U}^{(1)}_{r,m}$, $\mathrm{U}^{(2)}_{r,j,i}$, and $\mathrm{U}^{(3)}_{n,r}$ have dimensions $R \times M$, $R \times D \times D$, and $R \times N$, respectively.

CP and convolution formulas can be straightforwardly merged. The convolutional computation can be rewritten as:

$$\mathbf{Y}_{n,w',h'} = \sum_{r=1}^{R} \mathbf{U}^{(3)}_{n,r} \left(\sum_{j=1}^{D} \sum_{i=1}^{D} \mathbf{U}^{(2)}_{r,j,i} \left(\sum_{m=1}^{M} \mathbf{U}^{(1)}_{r,m} \mathbf{X}_{m,w_j,h_i} \right) \right). \tag{21}$$

To speed up the computation, a popular operation is to replace one large convolutional kernel with three smaller ones. The convolution operation in PyTorch has optimizations (e.g., parameter sharing). Therefore, the computational efficiency is dramatically improved compared with performing the matrix operation directly.

$$\begin{cases} \mathbf{Z}_{r,w,h} = \sum_{m=1}^{M} \mathrm{U}^{(1)}_{r,m} \mathbf{X}_{m,w,h}, \\ \mathbf{Z}'_{r,w',h'} = \sum_{j=1}^{D} \sum_{i=1}^{D} \mathrm{U}^{(2)}_{r,j,i} \mathbf{Z}_{r,w,h}, \\ \mathbf{Y}_{n,w',h'} = \sum_{r=1}^{R} \mathrm{U}^{(3)}_{n,r} \mathbf{Z}'_{r,w',h'}, \end{cases} \tag{22}$$

where $\mathrm{Z}_{r,w,h}$ and $\mathrm{Z}'_{r,w',h'}$ denote intermediate feature tensors with dimensions $R \times W \times H$ and $R \times W' \times H'$, respectively. The computational structure of the original convolution is illustrated in Fig. 5.

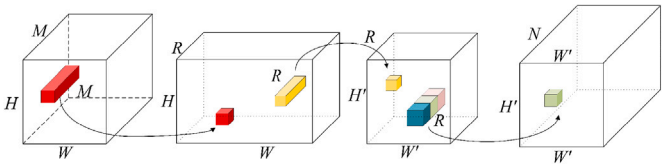


Fig. 5. Convolution with CP decomposition. The convolution process consists of three distinct stages. Initially, the input feature map undergoes convolution with smaller convolution kernels. Subsequently, the resulting feature map is convolved with an additional set of convolution kernels. Finally, the output feature map is generated through convolution with a final set of kernels.

In the previous calculation, the output from the first convolutional layer is the input to the second convolutional layer. Similarly, the output of the second layer is fed into the third. The intermediate layer uses group convolution to minimize model parameters. Three convolutional layers are connected sequentially. The CR and SR of the network are defined as follows:

$$\begin{cases} CR = \dfrac{MND^2}{RM + RD^2 + NR}, \\ SR = \dfrac{MND^2W'H'}{RMWH + RD^2W'H' + NRW'H'}. \end{cases} \tag{23}$$

CP-CNNs: CP decomposition for CNNs (CP-CNNs) is one of the earliest LFT techniques for model compression, popular due to its simple structure [125]. CP decomposition has replaced the convolution kernel with LFT methods, enabling efficient compression of CNNs. Despite its advantages, obtaining reliable low-rank representations has posed challenges, which has motivated several specialized methods.

- **LFT and Sparsity Methods:** Minimal Sensitivity and Intensity CNNs (MSI-CNNs) have provided stable CP representations by addressing degeneracy issues inherent in low-rank kernel approximation [45]. Low-rank Perceptual CNNs (LP-CNNs) have introduced sparsity into the weight matrix through regularization, creating a low-rank space that facilitates both CP decomposition and channel pruning [46]. In addition, Fast CP-Compression (F-CNNs) has enabled rapid compression of convolutional and FC layers without the need for fine-tuning, offering practical guidance for end-to-end model compression [47].
- **Interpretability and Design-Guided Compression Methods:** Interpretable CNNs (I-CNNs) have enhanced the theoretical understanding of deep CNNs, guiding the design of compressed networks in deeper architectures where conventional methods face limitations [48]. These approaches emphasize the relationship between network structure, low-rank representations, and compression performance, supporting more reliable and effective model compression.

CP-RNNs: Reflecting on prior research, LFT methods have been ranked in priority as TT<TK<TR in terms of accuracy and space complexity, i.e., TR has achieved the highest accuracy with fewer parameters, while TT has been the worst. However, none of the above LFT formats have provided both space-saving and computational efficiency.

- **Low-Rank Factorization and Parallel Efficiency Methods:** A compression method based on Kronecker CP-RNNs (KCP-RNNs) [49] has proven to be very promising based on LFT. Two efficient algorithms for LFT weight multiplication with inputs have been proposed. A CR of 278,219 times has been achieved using low-rank KCP. Additionally, KCP has exhibited excellent potential for parallel computing, enhancing computational speed.
- **Hybrid Optimization and Knowledge-Based Methods:** Zhang et al. [50] have investigated the hybrid optimization of several techniques, including tensor decomposition (TD), global average pooling (GAP) with self-taught KD, iterative structured sparsity (ISS) in LSTM (HO-LSTM), and 8-bit quantization. The feasibility of these approaches has been verified and has provided a basis for subsequent hybrid compression studies.

CP-Transformers: The primary advantage of CP decomposition is its extraordinarily large CR compared with other LFT methods. However, accuracy loss remains evident due to the limited capability of the latent factor matrix to capture expressive information. Compressing Transformer models with CP decomposition has therefore posed a significant challenge.

- **KD and Accuracy Recovery Methods:** CP-KD Transformers (CP-KD-Transformers) [51] have employed CP decomposition to reduce model complexity and storage space, followed by two-stage KD to recover part of the accuracy loss. The strategy has achieved a balance between efficiency and accuracy.
- **Novel LFT Methods:** It is worth noting that the use of CP decomposition for Transformer compression has not been fully explored, and tensor network designs based on CP remain particularly scarce. Fortunately, CP decomposition has demonstrated promising potential in RNNs, leading to noticeable improvements in model performance [49]. Therefore, CP-based compression for Transformers can draw inspiration from these existing RNN designs.

3.3. TK

TK decomposition can decompose an N-order tensor $\mathbf{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into an outer product of matrices and vectors. TK decomposition is formulated as [52]:

$$\begin{aligned} \mathbf{T} &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} C_{r_1 \cdots r_N} \left(\mathrm{U}_{r_1}^{(1)} \circ \mathrm{U}_{r_2}^{(2)} \cdots \mathrm{U}_{r_N}^{(N)} \right) \\ &= \mathbf{C} \times_1 \mathrm{U}^{(1)} \times_2 \mathrm{U}^{(2)} \times_3 \cdots \times_N \mathrm{U}^{(N)} \\ &= [\mathbf{C}, \mathrm{U}^{(1)}, \mathrm{U}^{(2)}, \ldots, \mathrm{U}^{(N)}], \end{aligned} \tag{24}$$

where $\mathbf{C} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor, and $\mathrm{U}^{(i)}$ is the i-th factor matrix. The outer product is used to construct the tensor. $\times_N$ denotes the N-mode product of tensors. $R_1 \times R_2 \times \cdots \times R_N$ are the TK ranks. We derive the explicit expression for each element of $\mathbf{T}$.

$$\mathbf{T}_{i_1 i_2 \cdots i_N} = \sum_{r_1}^{R_1} \cdots \sum_{r_N}^{R_N} \mathbf{C}_{r_1 \cdots r_N} \left(\mathrm{U}_{(i_1, r_1)}^{(1)} \cdots \mathrm{U}_{(i_N, r_N)}^{(N)} \right), \tag{25}$$

To apply TK decomposition to NNs, it is crucial to introduce the TK model, where the weight tensor is represented in the TK format.

Theoretical Derivation: TK decomposition can decompose a fourth-order tensor $\mathbf{W} \in \mathbb{R}^{D \times D \times M \times N}$ into a set of matrices and vectors [53]. Therefore, the TK format can be simplified as follows:

$$\mathbf{W}_{i,j,m,n} = \sum_{r_1}^{R_1} \cdots \sum_{r_4}^{R_4} \mathbf{C}_{r_1 \cdots r_4} \left(\mathrm{U}_{i,r_1}^{(1)} \circ \mathrm{U}_{j,r_2}^{(2)} \circ \mathrm{U}_{m,r_3}^{(3)} \circ \mathrm{U}_{n,r_4}^{(4)} \right), \tag{26}$$

where $\mathbf{C}$ is the core tensor, and $\mathrm{U}_{i,r_1}^{(1)}$, $\mathrm{U}_{j,r_2}^{(2)}$, $\mathrm{U}_{m,r_3}^{(3)}$, $\mathrm{U}_{n,r_4}^{(4)}$ are the latent factor matrices. In the FC layer, we usually reshape from $\mathrm{W} \in \mathbb{R}^{M \times N}$ to $\mathbf{W} \in \mathbb{R}^{1 \times 1 \times M \times N}$, where M and N are the input and output neurons, respectively. The tensor representation is given by the following expression:

$$\mathrm{W} = \mathbf{C} \times \mathrm{U}_{D \times R_1}^{(1)} \times \mathrm{U}_{D \times R_2}^{(2)} \times \mathrm{U}_{M \times R_3}^{(3)} \times \mathrm{U}_{N \times R_4}^{(4)}. \tag{27}$$

In practical network design, large kernel convolution introduces additional parameters that affect model inference. Consequently, the convolution kernels of 1×1 or 3×3 are usually adopted to compact

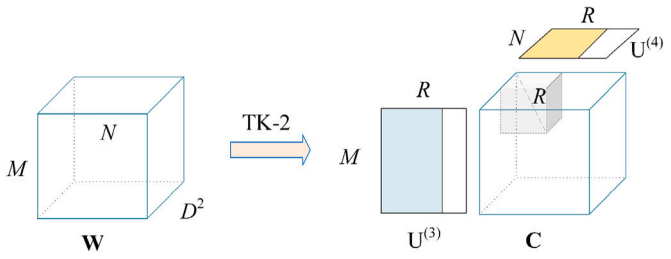


Fig. 6. The application of TK decomposition in convolutional layers. The figure demonstrates how the original fourth-order convolution kernel is transformed into three components in TK format. Through this decomposition, model parameters can be reduced while maintaining model performance, thereby enhancing the efficiency of the network.

the network structure. Decomposing the convolution kernel is unnecessary when D is quite small compared to M and N. The decomposition method is referred to as TK-2 decomposition, through which the weight tensor can be represented as:

$$\mathbf{W} = \mathbf{C}_{D,D,R_3,R_4} \times \mathbf{U}^{(3)}_{M\times R_3} \times \mathbf{U}^{(4)}_{N\times R_4}, \tag{28}$$

where $\mathbf{C} \in \mathbb{R}^{D\times D\times R_3\times R_4}$ is the core tensor, with R_3 and R_4 denoting the ranks of the final two dimensions. Meanwhile, $\mathrm{U}^{(3)}$ and $\mathrm{U}^{(4)}$ are the second-order matrices with sizes $M \times R_3$ and $N \times R_4$, respectively. TK-2 decomposition is illustrated in Fig. 6 when $R_3 = R_4 = R$.

Original convolutions can be represented as a series of small convolutions with the TK-2 format. Therefore, the output features are obtained by:

$$\begin{cases} \mathbf{Z}_{w,h,r_3} = \sum_{m=1}^{M} \mathbf{X}_{w,h,m} \mathrm{U}^{(3)}_{m,r_3}, \\ \mathbf{Z}'_{w',h',r_4} = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{r_3=1}^{R_3} \mathbf{Z}_{w,h,r_3} \mathbf{C}_{i,j,r_3,r_4}, \\ \mathbf{Y}_{n,w',h'} = \sum_{r_4=1}^{R_4} \mathbf{Z}'_{w',h',r_4} \mathrm{U}^{(4)}_{n,r_4}, \end{cases} \tag{29}$$

where $\mathbf{Z} \in \mathbb{R}^{W\times H\times R_3}$ and $\mathbf{Z}' \in \mathbb{R}^{W'\times H'\times R_4}$. More details about TK-2 decomposition can be found in [58,126]. In contrast to CP decomposition, TK decomposition replaces the convolution of the intermediate layers with a standard convolution rather than a group convolution. Therefore, the parameters of the convolutional layer are reduced from the original D^2MN to $MR_3 + D^2R_3R_4 + NR_4$ by decomposing the original weight tensor. The computational process of the original convolution is shown in Fig. 7.

TK-CNNs: Recent years have witnessed a surge of significant contributions focusing on TK decomposition for CNNs (TK-CNNs) [54–57]. Overall, research in this area has evolved along two major directions: algorithmic optimization for automatic rank selection and hardware–software co-design to improve both efficiency and practicality.

- **Optimization and Automatic Rank Selection Methods:** The choice of rank is critical: an inappropriate rank can degrade compressed model performance, and even with online learning [127], class-imbalance [128], reinforcement learning [129,130] or other robust training methods [131,132], accuracy is hard to recover. To begin with, BATUDE-CNNs [58] have introduced a budget-aware ADMM optimization algorithm based on TK decomposition, which has enabled efficient tensor rank computation within a single training session. Building upon this, the Efficient Alternating Optimization Algorithm (EAOA) [59] has further improved compression effectiveness by iteratively solving both network training and TK decomposition subproblems, thereby selecting appropriate TK ranks for multiple layers during training without requiring extensive fine-tuning. Furthermore, ARSR-CNNs [60] have advanced this line of research by combining adversarial robustness training with ADMM, allowing automatic and layer-specific rank selection. As a result, the method has avoided the tedious process of manual tuning while still maintaining high model accuracy.
- **Hardware-Aware and Structured Compression Methods:** In parallel with algorithmic developments, increasing attention has been paid to hardware-aware design. For instance, HCF-CNNs [61] have incorporated explicit hardware constraints into a co-optimization framework and optimized it using the ADMM algorithm, thus ensuring that compression strategies align with practical deployment needs. Similarly, HALOC-CNNs [52] have introduced a hardware-aware framework for automatic low-rank compression, leveraging architectural search to determine the optimal hierarchical rank in a differentiable and hardware adaptive manner. Moreover, CORING [53] has complemented these approaches by leveraging tensor decomposition, specifically Higher-Order SVD (HOSVD), to preserve the multidimensional structure of filters while providing low-rank approximations. In addition, it has integrated a direct and efficient filter selection strategy based on a similarity matrix, thereby significantly reducing computational complexity.

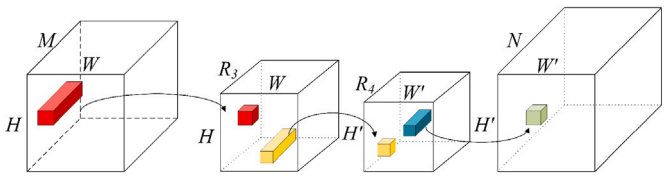


Fig. 7. TK-2 convolution operation. The first step involves a convolutional layer that transforms m feature maps into r_3 feature maps using a 1×1 patch. The second step applies a convolutional layer that maps r_3 feature maps to r_4 feature maps with a $D \times D$ patch. Finally, the third step uses a convolutional layer to convert r_4 feature maps into N feature maps with a 1×1 patch.

TK-RNNs: In recent years, there has been growing interest in applying TK decomposition to RNNs (TK-RNNs) to overcome the limitations of traditional RNNs in modeling high-order traffic flow data. Overall, research in this field has progressed along two main directions: improving prediction accuracy through compact model design and enhancing compression efficiency through hybrid strategies.

- **Compact Model Design for High-Order Data:** Traditional RNNs have struggled to handle high-order traffic flow data and have failed to capture the inherent structural relationships, which has led to inaccurate multi-modal prediction services [133]. To address this limitation, researchers have combined TK decomposition with GRU to construct a compact TK-GRU model, thereby significantly improving the accuracy of traffic flow prediction.
- **Hybrid Compression and Robustness Enhancement Methods:** Building upon compact model design, hybrid strategies have been developed to further reduce parameter size and computation cost. For example, a combination of KD and TK decomposition [62] has proven to be a highly cost-effective compression strategy. Specifically, a large teacher model has transferred knowledge to a compact student model, which is then further compressed using TK decomposition. As a result, minimal accuracy loss and tens of times inference speedup have been achieved. Furthermore, integrating TK decomposition with quantization (TKDQ) [63]

has enabled the development of a robust framework capable of handling noise and missing values while further improving the CR.

- **Hierarchical Decomposition and High-Accuracy Methods:** Hierarchical TK decomposition (HTK) [64] is a hierarchical variant of the TK decomposition that has traded off higher storage costs for excellent accuracy in compact RNN models. Unlike existing LFT methods, which have only decomposed the inputs of RNNs to the hidden layers, the HT decomposition method has comprehensively compressed the entire RNN model while preserving extremely high accuracy. In general, HT decomposition has exhibited lower space complexity on tensor data of equal size with the same number of selected ranks. Leveraging this theoretical advantage, large-scale RNN models have been represented in HT format with a reduced number of parameters.

TK-Transformers: Q, K, and V can be mapped into three-factor matrices, where each latent factor matrix contains three sets of orthogonal basis vectors in transformers. A new attention mechanism can be constructed by initializing a third-order diagonal tensor [134]. The TK decomposition of individual attention is calculated as follows:

$$Atten_{TD}(\mathrm{C};\mathrm{Q},\mathrm{K},\mathrm{V}) = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M}\mathbf{C}_{ijm}\circ\mathrm{Q}_i\circ\mathrm{K}_j\circ\mathrm{V}_m, \tag{30}$$

where $\mathbf{C}$ is the core tensor, and i, j, and m are the indices of the core tensor. Q_i, K_j, and V_m are the column vectors of matrices Q, K, and V, respectively. In practice, $I = J = M = R$. The core tensor is defined as:

$$\mathbf{C}_{ijk} = \begin{cases} rand(0,1), & \text{if } i = j = m, \\ 0, & \text{otherwise}, \end{cases} \tag{31}$$

where $\mathrm{rand}(0,1)$ generates random numbers uniformly distributed between 0 and 1.

Extreme Compression in Transformers: To boost both the effectiveness and efficiency of compression, recent research has focused on exploring extreme compression strategies for Transformer models. These efforts have primarily followed two directions: ultra-compact LFT methods and hybrid compression approaches combining LFT with complementary techniques.

- **Ultra-Compact LFT Methods:** Some researchers have achieved extreme compression of BERT through LFT, obtaining performance comparable to the original model while reducing the number of parameters to only one-seventh of the original size. In this way, the models have retained high accuracy despite significant parameter reduction.
- **Hybrid Knowledge-Distillation and Structured Compression Methods:** Synergistic approaches combining KD with LFT (KDLD-Transformers) [65] have created new compression opportunities. Additionally, LFT-SP-Transformer [66] has integrated LFT with sparse pruning, leveraging both coarse and fine-grained structural information for compression. These hybrid methods have presented appealing alternatives to relying solely on pure LFT.

3.4. TR

Given an N-mode tensor $\mathbf{T} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ with $\prod_{i=1}^{N} I_i$ degrees of freedom, a TR decomposition decomposes $\mathbf{T}$ into N independent third-order tensors, $\mathrm{U}^{(1)},\ldots,\mathrm{U}^{(N)}$, such that each entry within the tensor $\mathbf{T}$ is represented as [67]:

$$\mathbf{T}_{i_1,\ldots,i_t} = \sum_{r_1,\ldots,r_t}\mathrm{U}^{(1)}_{r_t,i_t,r_1}\mathrm{U}^{(2)}_{r_1,i_2,r_2}\cdots\mathrm{U}^{(t)}_{r_{t-1},i_t,r_t}, \tag{32}$$

where $\mathrm{U}^{(i)} \in \mathbb{R}^{R\times I_i\times R}$ represents the factor matrix of each mode, and R is the TR rank.

Theoretical Derivation: The TR decomposition is applied to the weight tensor $\mathbf{W} \in \mathbb{R}^{M\times N}$ or $\mathbf{W} \in \mathbb{R}^{D\times D\times M\times N}$ in TR networks. For instance, a fourth-order tensor is decomposed into four third-order tensors:

$$\mathbf{W}_{i,j,m,n} = \sum_{r_1,r_2,r_3=1}^{R}\mathbf{U}^{(1)}_{r_1,i,j,r_2}\mathbf{U}^{(2)}_{r_2,m,r_3}\mathbf{U}^{(3)}_{r_3,n,r_1}, \tag{33}$$

If M and N are large, the tensors $\mathbf{U}^{(2)}$ and $\mathbf{U}^{(3)}$ can be further decomposed into smaller tensors.

The fusion of kernel LFT with the convolution operation can be equivalently computed in three steps:

$$\begin{cases} \mathbf{Z}_{h,w,r_2,r_3} = \sum_{m=1}^{M}\mathbf{X}_{h,w,m}\mathbf{U}^{(2)}_{r_2,m,r_3}, \\ \mathbf{Z}'_{h',w',r_3,r_1} = \sum_{i,j=1}^{D}\sum_{r_2=1}^{R}\mathbf{Z}_{h,w,r_2,r_3}\mathbf{U}^{(1)}_{r_1,i,j,r_2}, \\ \mathbf{Y}_{h',w',n} = \sum_{r_1}\sum_{r_3}\mathbf{Z}'_{h',w',r_3,r_1}\mathbf{U}^{(3)}_{r_3,n,r_1}, \end{cases} \tag{34}$$

where the first term maps M feature maps to R feature maps with a 1×1 patch, the second term maps R feature maps to R feature maps with a $D\times D$ patch, and the last term maps R^2 feature maps to N feature maps with a 1×1 patch.

Based on the above analysis, CR and SR are commonly used to quantify the efficiency of TR networks:

$$\begin{cases} CR = \dfrac{MND^2}{R^2M + R^2D^2 + NR^2}, \\ SR = \dfrac{MND^2W'H'}{R^2MWH + R^3D^2W'H' + R^2NW'H'}. \end{cases} \tag{35}$$

If the compressed objects are FC layers, a similar equivalent process is applied. More details can be found in TR-CNNs [67].

TR-CNNs: TR decomposition for CNNs (TR-CNNs) has demonstrated advantages over TSVD and CP decomposition by preserving information integrity more effectively for higher-order tensors. Studies have indicated that TR is more expressive than TT for the same intermediate rank, which facilitates generalization [68]. Consequently, basic TR decomposition has outperformed other compression schemes in specific scenarios or datasets [67]. However, standard TR decomposition has faced challenges in balancing storage complexity and accuracy, motivating the development of targeted optimization strategies.

- **Regularization and Low-Rank Induction Methods:** Regularization techniques have been frequently employed to induce weight sparsity and create a low-rank space in CNNs (R-CNNs) [46]. These methods have facilitated latent factorization and channel pruning in compressed perceptual blocks, effectively improving compression efficiency while preserving model performance.
- **Evolutionary and Rank Search Methods:** Evolutionary algorithms have proven effective in searching for suitable TR ranks. For example, the Asymptotic Search TR Network (PSTRN) [69] has employed genetic algorithms to efficiently identify the optimal rank, converging rapidly to the region of interest and achieving strong performance through its evolutionary and progressive optimization phases.

TR-RNNs: To better align the input data with the model, it is necessary to adjust the shapes of the input vector $\mathbf{x}$ and the weight matrix $\mathbf{W}$ using the reshape operation. The computational process from the input vector $\mathbf{x}$ to the output vector $\mathbf{y}$ is represented as a decomposition layer:

$$\boldsymbol{y} = TDL(\mathbf{W},\boldsymbol{x}), \tag{36}$$

where TDL denotes the LFT layer. In RNNs, a model based on LFT can be obtained by replacing the multiplication between the weight matrix

and the input vector with TDL. The hidden state at time t is expressed as:

$$\boldsymbol{h}_t = \sigma(TDL(\mathbf{W}_{hx}, \boldsymbol{x}_t) + \mathbf{U}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}), \tag{37}$$

where $\mathbf{W}_{hx}$ and $\mathbf{U}_{hh}$ are the input-to-hidden and hidden-to-hidden weight matrices, respectively.

Next, TDL is incorporated into the standard LSTM, which is a high-level variant of RNNs. The TDL-LSTM model is formulated as:

$$\begin{cases} \boldsymbol{f}_t = \sigma(TDL(\mathrm{W}_f, \boldsymbol{x}_t) + \mathrm{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f), \\ \boldsymbol{i}_t = \sigma(TDL(\mathrm{W}_i, \boldsymbol{x}_t) + \mathrm{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \\ \boldsymbol{o}_t = \sigma(TDL(\mathrm{W}_o, \boldsymbol{x}_t) + \mathrm{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o), \\ \boldsymbol{g}_t = \tanh(TDL(\mathrm{W}_g, \boldsymbol{x}_t) + \mathrm{U}_g\boldsymbol{h}_{t-1} + \boldsymbol{b}_g), \\ \boldsymbol{c}_t = \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t\boldsymbol{g}_t, \\ \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \end{cases} \tag{38}$$

According to the aforementioned calculations, the proposed TDL-LSTM reduces redundant information in the high-dimensional inputs while achieving performance comparable to that of the standard LSTM [70].

TR-Transformers: TR decomposition for Transformers has demonstrated advantages in capturing relationships between data, particularly when the data exhibits ring or periodic characteristics. This property has contributed to improved model performance in certain tasks. Research has primarily focused on two directions: lightweight multi-scale design and higher-order tensor representation for efficient computation.

- **Lightweight Multi-scale Design Methods:** A multiscale lightweight TR-Transformer has been proposed to predict product lifetime in life cycle management. The model constructs two levels of weights using low-rank matrices, mapping tensors with larger dimensions to smaller ones to significantly reduce computational effort. Although multimodal inputs have enhanced model robustness, they have also increased computational and storage requirements [71].
- **Higher-Order Tensor Representation Methods:** The TR decomposition has been applied to the MHSA (TR-MHSA) and FFN (TR-FFN) layers, enabling the model to acquire approximate representations of higher-order tensors. This network has facilitated the transmission of multimodal low-level intercorrelations to compact high-level representations in a recursive manner [72].

3.5. TT

Given a tensor $\mathbf{T} \in \mathbb{R}^{N_1 \times \cdots \times N_d}$, it can be decomposed into a series of third-order tensors and second-order matrices via TT decomposition. It is defined as [135]:

$$\begin{aligned} \mathbf{T}_{(i_1,i_2,\ldots,i_t)} &= \mathrm{U}^{(1)}_{(:,i_1,:)} \cdots \mathrm{U}^{(d)}_{(:,i_t,:)} \\ &= \sum_{\alpha_0,\alpha_1\cdots\alpha_t}^{r_0,r_1,\cdots r_t} \mathrm{U}^{(1)}_{(\alpha_0,i_1,\alpha_1)} \cdots \mathrm{U}^{(d)}_{(\alpha_{t-1},i_t,\alpha_t)}, \end{aligned} \tag{39}$$

where $\mathbf{U}^{(k)} \in \mathbb{R}^{r_{k-1}\times n_k \times r_k}$ are referred to as TT-cores, and r_k are the TT-ranks that determine the storage complexity of the TT-format tensor. Similarly, the weight tensor can be transformed into a series of smaller convolutions using TT decomposition. An example is illustrated in Fig. 8.

Theoretical Derivation: TT decomposition can be applied to both convolutional and FC layers of CNNs. While both implementations share similarities, there are some differences in certain details. In a TT-FC layer, consider a standard FC layer with weight matrix $\mathrm{W} \in \mathbb{R}^{M\times N}$ and input vector $\boldsymbol{x} \in \mathbb{R}^N$, where $M = \prod_{k=1}^t m_k$ and $N = \prod_{k=1}^t n_k$. The output $\boldsymbol{y} \in \mathbb{R}^M$ is obtained by $\boldsymbol{y} = \mathrm{W}\boldsymbol{x}$ [136]. To obtain a standard TT-FC layer, it is common to tensorize W into a tensor $\mathbf{W} \in \mathbb{R}^{(m_1\times n_1)\times\cdots\times(m_t\times n_t)}$

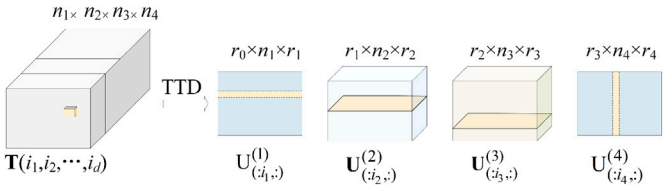


Fig. 8. Illustration of TT decomposition for a 4-order Tensor. The figure illustrates how a fourth-order tensor can be decomposed into a series of third-order tensors (TT-cores) and second-order matrices. Each TT-core is represented as $\mathrm{U}^{(k)}$, where K denotes the index of the core, and r_k refers to the TT-ranks, which determine the storage complexity of the tensor in TT format.

by reshaping or transposing. Thus, $\mathbf{W}$ is decomposed into TT-format:

$$\mathbf{W}_{((i_1,j_1),\ldots,(i_t,j_t))} = \mathrm{U}^{(1)}_{(:,i_1,j_1,:)} \cdots \mathrm{U}^{(t)}_{(:,i_t,j_t,:)}, \tag{40}$$

where each TT-core $\mathbf{U}_k \in \mathbb{R}^{r_{k-1}\times m_k\times n_k\times r_k}$ is a four-order tensor, which is one dimension higher than the standard TT-core due to the separate division of the output and input dimensions of W. Therefore, the forward propagation can be computed in TT-format as follows:

$$\mathbf{Y}_{(i_1,\ldots,i_t)} = \sum_{j_1,\ldots,j_t} \mathrm{U}^{(1)}_{(:,i_1,j_1,:)} \cdots \mathrm{U}^{(t)}_{(:,i_t,j_t,:)} \mathbf{X}_{(j_1,\ldots,j_t)}, \tag{41}$$

where $\mathbf{X} \in \mathbb{R}^{m_1\times\cdots\times m_t}$ and $\mathbf{Y} \in \mathbb{R}^{n_1\times\cdots\times n_t}$ are the tensorized input and output corresponding to $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. More details are provided in [135].

In a TT convolutional (TT-Conv) layer, the forward computation involves performing a convolution operation between a third-order input tensor $\tilde{\mathbf{X}} \in \mathbb{R}^{W\times H\times N}$ and a fourth-order weight tensor $\tilde{\mathbf{W}} \in \mathbb{R}^{D\times D\times M\times N}$ to obtain an output tensor $\tilde{\mathbf{Y}} \in \mathbb{R}^{(W-D+1)\times(H-D+1)\times M}$. In a TT-Conv layer, the input tensor $\tilde{\mathbf{X}}$ is reshaped to a tensor $\mathbf{X} \in \mathbb{R}^{D\times D\times n_1\times\cdots\times n_t}$, while the weight tensor $\tilde{\mathbf{W}}$ is reshaped and transposed to a tensor $\mathbf{W} \in \mathbb{R}^{(D\times D)\times(m_1\times n_1)\times\cdots\times(m_t\times n_t)}$ and decomposed into TT-format:

$$\mathbf{W}_{((d_1,d_2),(i_1,j_1),\cdots,(i_t,j_t))} = \mathrm{U}^{(0)}_{(d_1,d_2)} \cdots \mathrm{U}^{(t)}_{(:,i_t,j_t,:)}, \tag{42}$$

where $M = \prod_{k=1}^t m_k$ and $N = \prod_{k=1}^t n_k$. Similar to the TT-FC layer, $\mathrm{U}_k \in \mathbb{R}^{r_{k-1}\times m_k\times n_k\times r_k}$ is a four-order tensor except for $\mathbf{U}_0 \in \mathbb{R}^{D\times D}$. The output tensor $\mathbf{Y} \in \mathbb{R}^{(W-D+1)\times(H-D+1)\times m_1\times\cdots\times m_t}$ is obtained as:

$$\begin{aligned} \mathbf{Y}_{(w,h,i_1,\ldots,i_t)} = \sum_{d_1=1}^{D}\sum_{d_2=1}^{D}\sum_{j_1,\ldots,j_t} & \mathbf{X}_{(d_1+w-1,d_2+h-1,j_1,\ldots,j_t)} \\ & \mathrm{U}^{(0)}_{(d_1,d_2)} \mathrm{U}^{(1)}_{(:,i_1,j_1,:)} \cdots \mathrm{U}^{(t)}_{(:,i_t,j_t,:)}, \end{aligned} \tag{43}$$

The TT-Conv layer has been detailed in [73–75]. Similarly, the weight tensor can be decomposed using TT decomposition, transforming it into a series of smaller convolutions. By decomposing the weight tensor, the parameters of the convolutional layer are reduced from the original D^2MN to $R^2D^2 + R^2M + R^2N$. The computation process of TT-Conv is depicted in Fig. 9.

TT-CNNs: TT decomposition for CNNs (TT-CNNs) has provided a more expressive representation compared to SVD by capturing higher-dimensional features and offering greater flexibility in balancing complexity and accuracy. Research efforts have primarily concentrated on two directions: decomposition and compression strategies, as well as stability and nonlinear enhancements.

- **Adaptive Decomposition and Co-Optimization Methods:** A high-order decomposition scheme for CNNs (HOD-CNNs) has been developed through systematic exploration of the underlying causes of computational inefficiencies and corresponding mitigation strategies, leading to simultaneous reductions in computational and storage costs [73]. Variable Bayesian Matrix

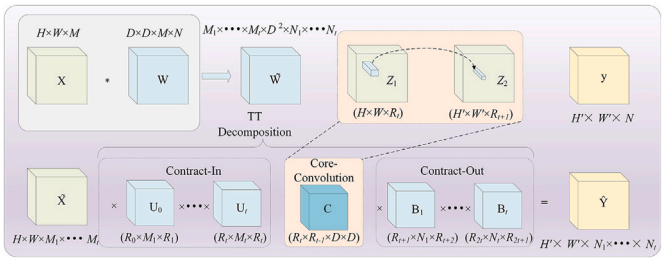


Fig. 9. The overall decomposition format and execution scheme of TT-Conv. It depicts a decomposition process that involves multiple successive convolutional layers, which are transformed into a series of smaller convolutional operations through low-rank approximation.

Factorization (VBMF) has enabled probabilistic estimation of ranks by integrating Bayesian statistics with matrix decomposition, allowing dynamic rank updates as data is observed [76]. Furthermore, the expansion of the CNN compression space has involved co-optimization strategies such as LFT and quantization (TQ-CNNs) [77], low-rank and sparsity (LS-CNNs) [78], LFT and pruning (LP-CNNs) [79], as well as algorithm–hardware co-design (AD-CNNs) [80,137].

- **Stability and Nonlinear Enhancement Methods:** TT decomposition has often suffered from vanishing gradients during three successive convolutional computations. To address this limitation, researchers have introduced nonlinear activation functions after convolutional layers to enhance the model's nonlinear representation capability. This strategy has stabilized the gradient flow during backpropagation and improved learning efficiency in CNNs (NAF-CNNs) [81]. The structure of an LFT with an activation layer is shown in Fig. 10.
- **Optimization-Guided High-Fidelity Compression Methods:** Despite the progress of existing techniques, most approaches generally involve lossy compression, particularly under large CR. To address this issue, a common strategy integrates optimization algorithms to achieve lossless or near-lossless compression at lower compression rates (CR), and to preserve high accuracy even under more aggressive compression. For example, incorporating a tensor rank constraint into the loss function and decomposing it into two subproblems for separate optimization enhances compression model performance [135]. Moreover, TT decomposition has been applied to compress 3DCNNs to accelerate inference, demonstrating promising applications in video processing, 3D point cloud recognition, and related fields [82].

TT-RNNs: TT decomposition has become one of the most widely used LFT methods in sequence modeling. Its ability to represent large weight matrices in a compact TT format enables substantial parameter reduction without severely sacrificing performance. Meanwhile, various enhancement strategies have been developed to further boost compression efficiency and model generalization. Overall, existing research on TT-RNNs can be broadly categorized into two directions: core compression methods and enhanced structural optimization strategies.

- **TT-based Core Compression in Sequential Models:** TT decomposition has been among the most commonly employed methods for reducing the parameters of sequence models [138,139]. Training RNNs with large input matrices has traditionally been challenging [117,140]. The tensorized LSTM model is depicted in Fig. 11. TT-RNNs alleviate this issue by factorizing large weight matrices into compact TT, significantly reducing model parameters while retaining temporal dependencies. Higher-order convolutional LSTM models efficiently learn spatio-temporal correlations by approximating convolutional kernel sequences as lower-order

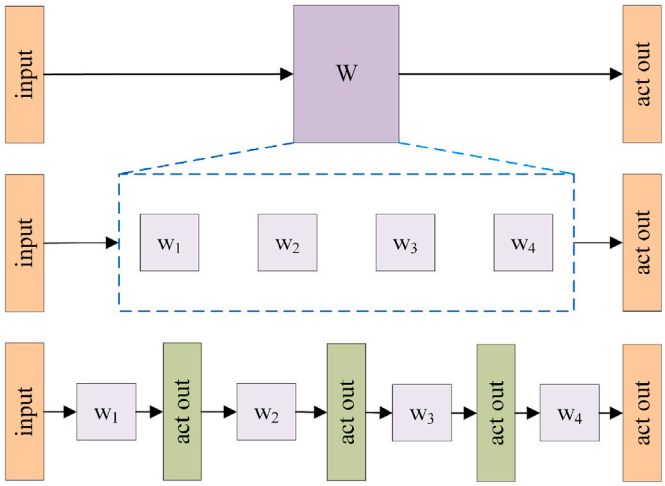


Fig. 10. LFT with an activation layer. W represents the pre-trained weights of a specific layer, while W1 to W4 denote the decomposed weight matrices or tensors. Nonlinear activation functions are introduced based on LFT to enhance the model's ability to represent nonlinearity. Incorporating activation functions after consecutive convolutional layers ensures that gradients do not vanish during backpropagation, allowing the CNNs to learn efficiently.

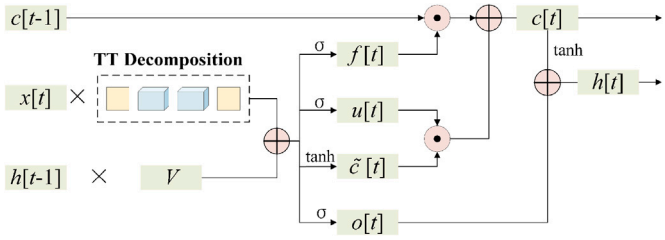


Fig. 11. Tensorized LSTM Model. The weight matrix is factored into multiple consecutive sub-layers using TT decomposition.

tensor sequences. This makes higher-order ConvLSTM more compact than its first-order counterpart and induces a regularizing effect [118]. TT-RNNs have demonstrated strong performance in multimodal prediction tasks, including a 25.29% accuracy improvement in metro traffic flow forecasting [83]. Moreover, TT-GRU achieves CR of 200–780×, and has been successfully applied to financial prediction tasks, offering enhanced interpretability through LFT [84].

- **Enhanced Compression via Structural and Regularization Strategies:** In addition to TT decomposition, tensor contraction has been used to compress parameters between layers in RNNs (TC-RNNs) [85]. Further improvements were achieved through an efficient tensorized convolutional layer (ETTConv), which tensorizes convolution kernels and maps them to smaller kernels, significantly reducing both parameters and computation [86]. Regularization and global average pooling are also incorporated to further enhance compression and generalization. These strategies expand the compression space beyond pure TT decomposition, improving both model compactness and robustness.

TT-Transformers: Transformer architectures often contain extremely large weight matrices, which makes deployment challenging on resource-constrained platforms. Leveraging TT decomposition to tensorize and compress these weights has become a promising solution for enhancing both storage and computational efficiency. Over the past few years, TT-based techniques have been widely applied in various

components of Transformers, ranging from fine-tuning schemes to structural compression. Overall, related research can be broadly divided into two major categories: TT-enhanced fine-tuning strategies and TT-based structural compression frameworks.

- **TT-based Incremental Parameter Compression:** Fine-tuning is a widely used approach for adapting pre-trained Transformer models to downstream tasks with minimal additional parameters. To make this process more efficient, a tensorization-based fine-tuning framework has been proposed to decompose only the *weight increments* rather than the entire weight matrices. This allows only the matrices representing potential factors to be updated and stored during adaptation, greatly reducing overhead [87]. Similarly, in NLP and image processing tasks, TT low-rank features have been integrated into the embedding layers of Transformers, enabling end-to-end training while supporting larger batch sizes thanks to lower memory consumption [89]. Moreover, low-bandwidth quantization has been combined with TT decomposition to further minimize model storage, as demonstrated in CLWM-Transformers [141].
- **TT-based Structural Compression Frameworks:** Beyond fine-tuning, TT decomposition has been extensively employed to restructure core Transformer components. For example, in the Industrial Internet context, Transformers++ introduces a plug-and-play lightweight MHSA and FFN structure based on TT decomposition, making it better suited for deployment on resource-constrained embedded devices [88]. In the DSFormer-LRTC model, TT decomposition has been applied specifically to the FFN modules, achieving substantial compression without compromising performance [90]. More recently, TTCL [142] has leveraged TT compression in combination with hardware optimization to reduce both storage and computational overhead in large language models, facilitating efficient edge deployment.

4. Training schemes

The aforementioned compression algorithms have demonstrated strong performance on various datasets. Developing more stable and efficient training schemes has remained an important research focus. Next, we cover stabilization training programs, which include mixed-precision training, initialization, and rank search strategies.

4.1. Mixed-precision training

Deep models are typically trained using the float32 data type instead of float64, primarily due to considerations of storage requirements and computational efficiency. Tensor compression, as patterns increase linearly, produces data streams exhibiting exponential scales, manifested in forward-propagating features and backward-propagating gradients [143]. A practical approach to mitigate these numerical issues is to employ the full-precision float64 format to represent large weights, which alleviates the numerical issues to some extent. Nonetheless, full-precision training consumes more time and storage space compared to lower-precision formats such as float16 and float32. Current practice adopts mixed-precision training to reduce memory usage and accelerate training while maintaining accuracy [91]. The dynamic training strategy effectively reduces memory usage and enhances the stability of the model training process.

Initialization Strategies: Initialization methods critically affect the convergence speed, performance, and training stability of NNs. Existing initialization approaches can be broadly divided into two categories:

- **Classical and Unified Initialization Strategies:** Classical methods such as Xavier initialization [92] and Kaiming initialization [93] are widely used to accelerate training and mitigate gradient vanishing or exploding issues. However, these methods often fail to accurately compute the scale of NNs and neglect interactions in tensor contraction. Yu initialization has addressed these limitations by unifying the Xavier paradigm and improving adaptability for LFT models [94]. Weight selection has also emerged as a novel initialization approach, often viewed as a pruning technique [95]. This method selects parameters from large pre-trained models to initialize smaller models and can be combined with KD to reduce training time and enhance accuracy.
- **Spectral Initialization for Stability Enhancement:** Spectral initialization remains underexplored in compressed networks. It involves eigenvalue decomposition of the weight matrix, followed by scaling of the eigenvectors to stabilize gradient flow [96]. The strategy effectively mitigates gradient vanishing and explosion, providing a promising approach for improving training stability. In practice, the choice of initialization method depends on network architecture, activation function, and task characteristics, and empirical evaluation is often conducted to select the optimal strategy.

4.2. Tensor rank search strategies

The complexity and CR of a model are directly influenced by the tensor rank, making the selection of an appropriate rank crucial [144]. Finding the optimal rank is an NP-hard problem, and manually setting it has been tedious in previous studies. Existing approaches for rank selection can be broadly divided into two categories:

- **Bayesian and Evolutionary Search Methods:** Some optimization methods have been developed to efficiently identify suitable tensor ranks. Variational Bayesian matrix factorization (VBMF) combines the flexibility of Markov chain Monte Carlo methods with the efficiency of variational Bayesian inference for rank estimation [76]. Progressive Search TR Network (PSTRN), a novel genetic algorithm, employs both evolutionary and progressive phases to rapidly converge towards optimal ranks, achieving strong performance [69]. These approaches provide systematic and reliable solutions for rank estimation, significantly easing the compression of CNNs.
- **Reinforcement Learning-based Rank Search:** Reinforcement learning (RL) methods treat rank search as a game within an irregular search space, exploring appropriate ranks for trained CNNs [97]. While RL-based approaches automate rank selection, their performance can be affected by the underlying tensor format, influencing overall efficiency and effectiveness in model compression.
- **Singular Value Selection:** Selecting the rank based on singular values is another commonly used and effective strategy. This approach typically determines the decomposition rank by applying an energy threshold, thereby retaining the dominant components while removing redundant information. For example, the OR-CNNs [32] obtain the decomposition rank through direct truncation of singular values. In contrast, the BATUDE-CNNs [58] and APD-PMC [98] explicitly account for the dimensional differences across different modes in Tucker decomposition and achieve a favorable trade-off between model accuracy and compression ratio through careful algorithmic design.

5. Toolboxes

For researchers or NN engineers, it is essential to have a handle on basic network compression tools due to their high efficiency, especially when used frequently. Hence, this section introduces some popular toolboxes for tensor operations and toolboxes for deep networks. An overview of several LFT tools is illustrated in Fig. 12.

5.1. Popular toolboxes for core tensor operations

Toolboxes have been developed to implement a variety of typical LFT methods. Existing toolboxes can be broadly divided into two categories:

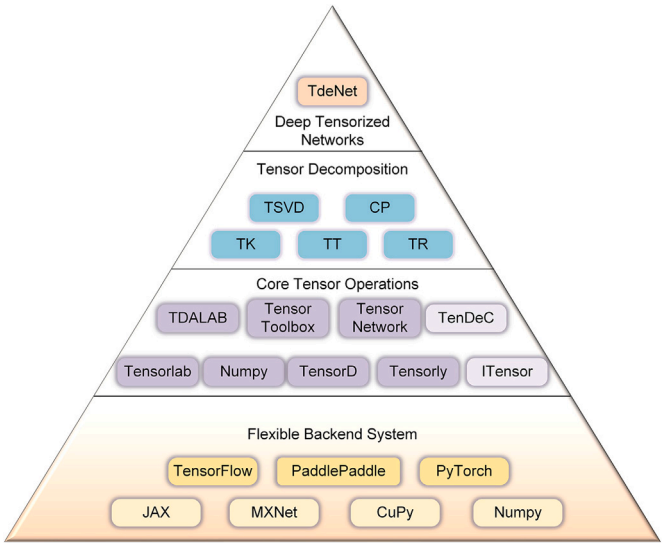


Fig. 12. Common tensor processing toolboxes.

- **MATLAB and Python Toolboxes:** MATLAB toolboxes, including Tensor Toolbox [99], Tensorlab [100], and TDALAB [101], provide extensive functions for tensor manipulation and decomposition of high dimensional data. For Python, Tensortools, implemented with Numpy [102], focuses on CP decomposition, whereas T3F [103] is tailored for TT decomposition within TensorFlow [103]. TensorD [104] supports multiple LFT formats, including CP and TK decomposition. General-purpose libraries such as Tensorly [105] and TensorNetwork [106] support multiple LFT formats and are compatible with Python frameworks including PyTorch, TensorFlow, and MXNet [107], enhancing flexibility for diverse workflows.
- **C++ Toolboxes:** High-performance C++ toolboxes (such as TenDeC++ [108] and ITensor [109]) provide efficient tensor operations using pointer techniques and optimized memory management. These tools are particularly suitable for large-scale computations or industrial applications requiring maximum performance.

5.2. Toolboxes for deep neural networks

Toolkits for NNs play an essential role in accelerating the development of tensorized architectures. Existing toolkits can be grouped into two categories:

- **Tensorly-based Toolkits:** Tensorly [104] provides decomposition and initialization of network layers but lacks a direct API for constructing complete network layers. Tensorly-Torch addresses this limitation by enabling tensorized layer construction within the PyTorch framework. It offers a high-quality API for tensor operations, LFT techniques, and regression functionalities, with a flexible backend supporting NumPy, PyTorch, TensorFlow, JAX, MXNet, PaddlePaddle [145] and CuPy. In addition, Tntorch [146], t3f [103], and HOTTBOX [147] can also perform basic tensor decomposition and manipulation.
- **Deep Tensorized Networks and Compression Toolkits:** TedNet [110] facilitates end-to-end construction of deep models with minimal code modifications, enabling more convenient model compression and deployment. It focuses on simplifying the workflow for building complete tensorized networks while maintaining flexibility and efficiency.

Table 4
Statistical information for different datasets.

Datasets	Class	Train	Test	Size
D1	10	5×10^4	1×10^4	28×28
D2	10	5×10^4	1×10^4	32×32
D3	100	5×10^4	1×10^4	32×32
D4	10	73257	26032	32×32

6. Empirical studies

This section presents the benchmark studies, including data preparation, evaluation metrics, experimental settings, comparison results and analysis, and summary. Whether it is CNNs, RNNs, or Transformers, the principle of compression based on LFT is remarkably similar. Thus, this paper only provides the experimental results of tensorized CNNs as a reference.

6.1. Data preparation

For fair experimental comparison, we will assess the performance of different tensorized models using the ResNet-20 backbone on publicly available datasets.[1] The utilized datasets are described as follows:

- *MNIST (D1)*. It is a frequently employed dataset for handwritten digit recognition tasks in both machine learning and deep learning. Comprising grayscale images measuring 28×28 pixels, it represents digits from 0 to 9. The dataset includes a total of 60,000 images, with 50,000 for training and 10,000 for testing. Each image has 784 pixels, with grayscale values indicating the intensity of each pixel.
- *CIFAR-10 (D2)*. This widely utilized computer vision dataset comprises 60,000 color images distributed across 10 categories, with 6000 images per category. The training set consists of 50,000 images, while the test set contains 10,000 images.
- *CIFAR-100 (D3)*. This widely utilized computer vision dataset comprises 60,000 color images distributed across 10 categories, with 6000 images per category. The training set consists of 50,000 images, while the test set contains 10,000 images.
- *SVHN (D4)*. It is used for number recognition tasks and primarily consists of house numbers extracted from Google Street View images. It contains a substantial collection of color images, including 73,257 training images, 26,032 test images, and 531,131 additional unlabeled images, making it one of the larger datasets for number recognition. The statistical details of the various datasets are presented in Table 4.

6.2. Evaluation metrics

This section provides a brief overview of the commonly used evaluation metrics for compression models, such as Top-1 Accuracy, Top-5 Accuracy, CR, and SR.

Top-1 and Top-5 Accuracy. Top-1 accuracy measures whether the model's most probable prediction aligns with the true label, while Top-5 accuracy assesses if the true label is included within the top five predicted categories. The formulas are as follows:

$$\begin{cases} \text{Top-1} = \dfrac{n_{\text{Top1}}}{N}, \\ \text{Top-5} = \dfrac{n_{\text{Top5}}}{N}, \end{cases} \tag{44}$$

where $n_{\text{top-1}}$ represents the number of correctly classified samples, $n_{\text{Top-5}}$ denotes the number of samples where the true label is among the top five

[1] MNIST: https://yann.lecun.com/exdb/mnist/; CIFAR-10/100: https://www.cs.toronto.edu/ kriz/cifar.html; SVHN: http://ufldl.stanford.edu/housenumbers/.

predictions, and N is the total number of samples. Compared to Top-1 accuracy, Top-5 accuracy provides a more lenient evaluation, making it particularly useful for multi-class classification tasks.

CR and **SR**. In NNs, CR indicates the reduction in the number of model parameters, thereby making the model more lightweight. The CR can be calculated using the following formula:

$$\begin{cases} CR = \dfrac{P_{original}}{P_{compressed}}, \\ SR = \dfrac{F_{original}}{F_{compressed}}, \end{cases} \tag{45}$$

where P_{original} and F_{original} denote the total number of parameters and FLOPs of the original NNs' weights and biases, respectively. $P_{\text{compressed}}$ and $F_{\text{compressed}}$ represent the corresponding quantities after LFT compression. The CR and SR reflect the algorithm's capability in parameter reduction and computational efficiency improvement, respectively.

In summary, a slight loss of precision is anticipated when the model undergoes significant compression. When designing compact networks, it is crucial to consider Top-1, Top-5, and CR simultaneously. In essence, a balance between these metrics should be sought to achieve an optimal performance.

6.3. Experimental settings

The experiments on neural networks are performed using the PyTorch framework with the Tensorly toolbox. All experiments are conducted on a desktop computer with a 2.50-GHz Intel CPU and 12-GB RAM. The batch size is set to 128, and the initial learning rate is 0.1, which is reduced to 0.01 at the 100th epoch. We use stochastic gradient descent (SGD) with a momentum of 0.9 as the optimizer and a weight decay of 1e-4. In this experiment, the original models (ResNet20) are first trained from scratch on four datasets and then the decomposed models are fine-tuned for additional epochs. Compressed low-rank format models are also fine-tuned on the four datasets with a learning rate of 0.1, reduced to 0.01 at the 50th epoch. Therefore, the training time of the compression model consists of three parts: pre-training, decomposition, and fine-tuning. Since the decomposition operation is similar across different compression methods for neural networks, we only choose ResNet20 as a representative example to compare the performance of various compression methods across the four datasets.

6.4. Comparison results and analysis

In this section, we train an original model and five LFT compression variants on four datasets using the classical ResNet-20 to evaluate the performance of different decomposition methods.

To provide a clearer analysis of the models' performance, the parameters of each model are compressed by half, resulting in a CR of approximately 2. As illustrated in Table 5, whether the input and output channels are equal determines the rank. A comparison of model parameters and FLOPs among different LFT methods is shown in Table 6. The experimental results of the original and decomposed models on the four datasets are presented in Table 7. Consequently, we observe that:

- **ResNet-20 on D1.** TK achieves the highest classification accuracy in both Top-1 and Top-5 while consuming fewer training resources. However, other LFT methods have also shown excellent results besides CP decomposition. The discrepancy could be attributed to the limited feature expressiveness of the latent factor matrix resulting from CP decomposition compared to tensors, while TT, and TR utilize tensors to achieve higher-order representation of features to obtain accuracy gains. In other words, complex feature information may be difficult to efficiently express through low-order matrices. Consequently, some crucial information might be lost during the training process, leading to a decrease in accuracy.

Table 5
Setting of rank for different compression methods. For each method, the smallest mode dimension is first determined and then multiplied by a scaling factor to achieve the target compression ratio.

Model	r	ratio	R
Original	–	–	–
TSVD	$\min(M,N)\times ratio$	0.40	r
CP	$\min(M,N)\times ratio$	1.92	r or $2r$
TK	$\min(M,N)\times ratio$	0.58	$[r,r]$ or $[r,2r]$
TT	$\min(M,N)\times ratio$	0.58	$[1,r,r,r,1]$ or $[1,r,r,2r,1]$
TR	$\min(M,N)\times ratio$	0.58	$[3,r/3,r,r,3]$ or $[3,r/3,r,2r,3]$

Table 6
Comparison of model parameters and FLOPs for different LFTs.

Methods	Parameters	FLOPs
Original	D^2MN	$D^2MNW'H'$
Full Rank	$D^2R(M+N)$	$D^2RW'H'(M+N)$
TSVD	$D^2R(M+N)$	$D^2RW'H'(M+N)$
CP	$R(M+D^2+N)$	$R(WHM+W'H'(D^2+N))$
TK	$R(M+D^2R+N)$	$R(WHM+W'H'(D^2R+N))$
TT	$R(M+D^2R+N)$	$R(WHM+W'H'(D^2R+N))$
TR	$R^2(M+N+D^2)$	$R^2(WHM+W'H'(N+D^2))$

Table 7
ResNet-20 on D1-D4 datasets.

No.	Model	Best accuracy		CR	SR	time(h)
		Top-1(%)	Top-5(%)			
D1	Origin	99.29	100	1.00	–	0.83
	TSVD	99.12	99.98	1.99	1.99	0.87
	CP	98.88	99.99	1.98	1.76	0.89
	TK	99.17	99.99	2.01	1.90	0.87
	TR	99.17	99.99	2.01	1.99	0.88
	TT	99.16	99.99	2.00	1.90	0.88
D2	Origin	91.25	99.72	1.00	–	0.83
	TSVD	86.32	99.46	1.99	1.99	0.87
	CP	84.39	98.97	1.98	1.76	0.89
	TK	88.37	99.57	2.01	1.90	0.87
	TR	88.42	99.60	2.01	1.90	0.88
	TT	88.11	99.61	2.00	1.90	0.88
D3	Origin	65.40	90.33	1.00	–	1.31
	TSVD	57.10	85.85	1.99	1.99	1.39
	CP	55.67	83.28	1.98	1.76	1.40
	TK	59.17	86.98	2.01	1.90	1.39
	TR	59.23	87.06	2.01	1.90	1.39
	TT	59.16	86.94	2.01	1.90	1.39
D4	Origin	95.77	99.53	1.00	–	1.64
	TSVD	87.78	99.09	1.99	1.99	1.73
	CP	85.93	98.65	1.98	1.76	1.75
	TK	88.57	99.11	2.01	1.90	1.74
	TR	88.51	99.06	2.01	1.90	1.74
	TT	88.46	99.04	2.00	1.90	1.74

- **ResNet-20 on D2**. TK, TR, and TT decompositions achieve similar results, both in terms of accuracy, training time, and inference FLOPs. Meanwhile, TSVD still achieves excellent performance with a fixed CR. However, CP decomposition consumes more training resources and FLOPs while obtaining worse classification accuracy.
- **ResNet-20 on D3**. Because of the excessive number of categories, the classification accuracy suffers a significant degradation compared to other datasets, especially with the TSVD and CP decompositions. Fortunately, it is noteworthy that TK, TT, and TR decompositions exhibit robust performance in balancing accuracy and training time, even in the face of more complex classification tasks.

- **ResNet-20 on D4**. The richer SVHN dataset and utilization of color images enable various potential factor decomposition models to extract features more effectively, resulting in higher accuracy. TK, TR, and TT have comparable numbers of parameters and FLOPs, leading to insignificant differences in training time.

Note that although low-rank decomposition reduces the number of model parameters, the training time may increase. This is because each original layer is split into multiple smaller sub-layers or tensor operations, which increases the number of forward and backward computations. Additional tensor operations such as reshaping, permutation, and matrix multiplications also introduce extra computational and memory overhead. Moreover, the decomposition can increase the effective network depth and the difficulty of optimization, requiring more iterations to achieve convergence. Therefore, the observed increase in training time is an inherent characteristic of low-rank decomposition-based model compression.

To achieve excellent model compression performance, two key conditions must be met:

- **Avoid excessive depth increase**. For instance, naive CP decomposition splits an original single-layer convolution into four sub-layers, significantly deepening the network. Increased depth complicates optimization and ultimately degrades model performance.
- **Ensure the expressive power of tensor factors**. Compared to matrix decomposition, TK, TT, and TR decompositions better preserve crucial information in convolutional layers by leveraging the high-order expressiveness of core tensors. The inferior performance of CP decomposition can be attributed to its excessive increase in network depth and its limited ability to exploit tensor structures for retaining the original high-order features.

In summary, TK, TT, and TR decomposition strike a better balance between CR and model performance by maintaining a reasonable network depth and employing higher order tensors to effectively represent the essential information of the original convolutional layers.

6.5. Training dynamics of LFT models

Fig. 13 presents the performance evaluation results of the LFT model across different datasets. The experiments led to the following conclusions.

- **ResNet-20 on D1**. All methods, including CP and TSVD, demonstrate rapid convergence with negligible differences in the training loss curves. The results imply that on simpler datasets with a limited number of classes, methods like CP, which usually struggle with complex data, are capable of performing adequately.
- **ResNet-20 on D2**. Convergence differences become more noticeable. While TK, TT, and TR methods maintain faster convergence, CP decomposition shows a much slower reduction in training loss, reflecting its difficulty in capturing complex patterns in this dataset compared to the tensor-based approaches.
- **ResNet-20 on D3**. The number of classes increases, making convergence much more challenging. Here, TK, TT, and TR decompositions outperform both TSVD and CP regarding convergence speed and final training loss. CP decomposition, in particular, converges more slowly and achieves higher final training loss, highlighting its limitations in handling larger and more diverse datasets.
- **ResNet-20 on D4**. Due to the dataset's richer feature representation (color images), TK, TT, and TR maintain robust convergence behaviors, achieving comparable training losses. CP, however, continues to lag, exhibiting a slower decrease in training loss.

For more complex datasets, LFT-based methods like TK, TT, and TR not only converge faster but also achieve better final performance. CP decomposition, while effective in reducing parameters, faces challenges in achieving low training loss, especially for high-dimensional or complex data. These findings indicate that higher-order tensor decompositions are more suitable for applications requiring both rapid convergence and strong performance.

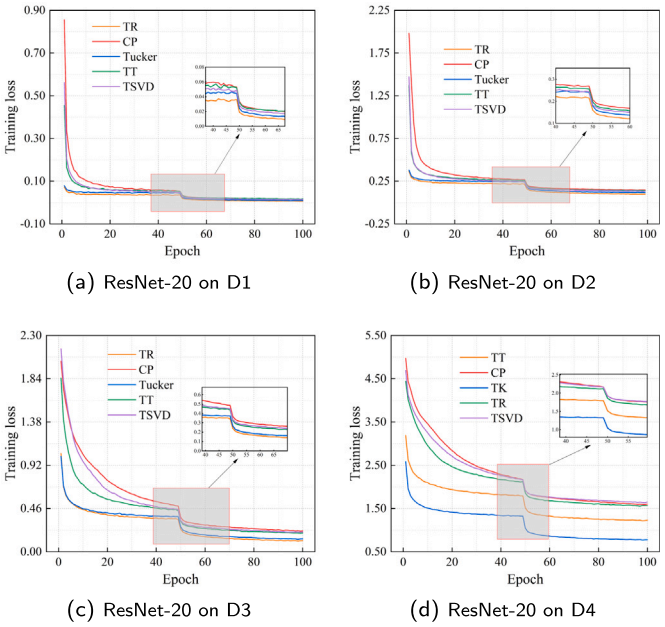


Fig. 13. The training curves of different LFT methods on various datasets.

Across all datasets, LFTs like TK, TT, and TR exhibit faster convergence compared to CP and TSVD. These methods consistently reach lower training loss values at earlier epochs, indicating that they can extract useful features more efficiently during training.

6.6. Summary

The experiments led to the following conclusions.

- For certain straightforward datasets, there is negligible variance in accuracy among different decomposition methods, whether considering Top-1 or Top-5. However, as the number of categories increases, it's advisable to consider TK decomposition to balance accuracy against training resources.
- As demonstrated in Table 5, when the input and output channels of a convolutional layer are not equal, different ranks should be used for decomposition. For example, if a layer has 64 input and 128 output channels, $R_{\text{in}} = 10$ and $R_{\text{out}} = 20$ can be chosen, whereas when both are 64, a single rank $R = 10$ is sufficient. The adaptive rank setting helps balance compression and accuracy.
- In the case of CP decomposition, the model's parameters decrease by half, but the FLOPs experience the least reduction across the four mentioned datasets. The discrepancy arises primarily due to the utilization of grouped convolution in implementing the convolutional layer with CP decomposition, which effectively reduces computational parameters. Furthermore, to maintain a CR of 2, CP decomposition opts for a larger rank compared to TK decomposition. Consequently, the FLOPs for CP decomposition are calculated as $RMWH + RD^2W'H' + NRW'H'$, whereas for TK decomposition, they are $RMWH + R^2D^2W'H' + NRW'H'$.
- CP decomposition exhibits a clear advantage in parameter reduction. However, its FLOPs experience the least reduction, leading to considerably longer fine-tuning times compared to other decomposition methods. Although TK and TT share similar parameters and FLOPs, their principles and implementation nuances differ significantly, resulting in notable variations in performance.

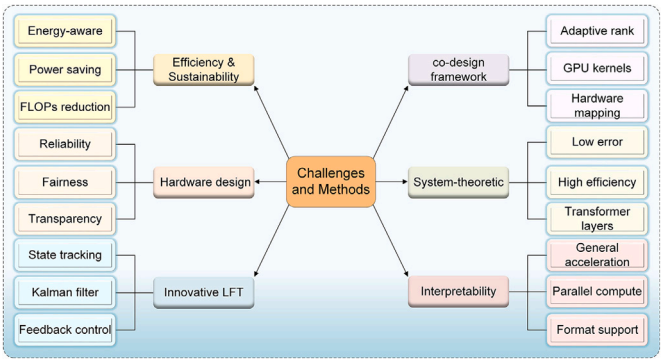


Fig. 14. Potential research directions.

7. Future research directions in LFT-based model compression

The previous sections have provided a comprehensive overview of the background, fundamental concepts, and representative methods of LFT-based model compression. Despite the significant progress made in recent years, several critical challenges remain to be addressed to further advance the field. This section discusses these key challenges and outlines promising future research directions to inspire further exploration and innovation. An overview of the potential research directions is presented in Fig. 14.

- **Algorithm and hardware-aware co-design framework:** Highly accurate tensor-format compression models can be trained using existing optimization techniques; however, achieving the desired device speedup remains challenging. The challenge arises because the model architecture of the decomposition is determined by empirically set target ranks, making it difficult to adapt to all hardware devices. Future research should incorporate hardware constraints into the algorithm design process, followed by the construction of an LFT format with a high-performance GPU kernel to guide the selection of execution parameters [148]. Iterative learning and scheduling mechanisms developed for networked systems under random access protocols [149] may further inspire multi-stage and adaptively scheduled compression and deployment strategies.
- **Design of innovative LFT methods:** LFT has demonstrated remarkable potential in reducing model size; however, most existing approaches still suffer from relatively large approximation errors, which limit their effectiveness in balancing the CR and model accuracy. Therefore, it is crucial to explore innovative LFT techniques with lower reconstruction error and higher representational efficiency [150]. In this context, ideas from nonlinear signal enhancement and Bayesian sparse learning, such as stochastic resonance [151] and nonparametric Bayesian group sparsity [152], may offer useful inspiration for designing adaptive and error-resilient LFT schemes. In addition, beyond their widespread use in convolutional and recurrent networks, LFT methods can be further extended to the FC layers of Transformer architectures, where substantial redundancy exists. Such innovations may unlock new opportunities for achieving efficient yet accurate compression across diverse NNs. Leveraging these structural properties could yield a potent and appealing compression scheme [153].
- **Acceleration based on hardware design:** Current algorithms effectively reduce model complexity and storage space in theory; however, achieving the desired speedup in real-world hardware deployments remains challenging due to the large number of matrix multiplication operations and insufficient parallelism. Existing tensor acceleration hardware and software structures are often tailored to specific LFT schemes or rely on parallel matrix acceleration. Therefore, there is a need to develop generalized acceleration schemes that can support various LFT formats [154].
- **Energy-efficient and sustainable compression:** With the increasing scale of deep NNs, the energy consumption and carbon footprint associated with model training and deployment have grown substantially. While LFT-based compression can reduce model size and computational cost, most existing methods primarily focus on accuracy and CR, neglecting their impact on energy consumption. Future research should explicitly incorporate energy efficiency and environmental sustainability into the compression objectives, particularly for industrial and robotic systems that demand high accuracy and robustness under resource constraints [155,156], as well as continuous and contactless sensing applications such as radar-based ECG monitoring [157,158]. This includes developing energy-aware compression algorithms, measuring real energy savings during deployment, and designing green-friendly LFT strategies that minimize both FLOPs and power usage. Such efforts could make LFT-based compression more aligned with sustainable AI development [159].
- **Trustworthy and explainable compression:** LFT-based compression can effectively reduce model size and computational cost; however, its impact on model trustworthiness and interpretability remains largely underexplored. Compression may alter the internal representation and decision boundaries of NNs, potentially leading to unpredictable behavior or degraded robustness. Future research should focus on developing trustworthy compression frameworks that preserve or even enhance model reliability, robustness, and fairness after compression. In addition, integrating explainability techniques into the compression process can help researchers and practitioners better understand how LFT affects information flow and decision-making [160]. Such efforts would not only increase transparency but also build greater confidence in deploying compressed models in safety-critical or high-stakes scenarios.
- **System-theoretic and control-based compression frameworks:** LFT-based compressed neural networks can be viewed as dynamical systems, where layer-wise activations or tensor factors act as system states [161,162]. From this perspective, state estimation and filtering techniques developed for networked and resource-constrained systems, such as Tobit filtering [163], recursive estimation under communication constraints or adversarial conditions [164–166], and secure or event-triggered estimation frameworks [167–170], offer principled tools to monitor and infer the behavior of compressed models. Moreover, estimation methods for energy-limited and distributed systems [171–173] can inspire adaptive and resource-aware compression strategies. Integrating these system-theoretic ideas with feedback control may enable adaptive rank selection and robust performance maintenance in LFT-based model compression [174].

8. Conclusions

With the successful application of deep NNs, model compression techniques based on LFT have experienced rapid development. Leveraging the high CR offered by LFT techniques, the deep NNs can be readily applied to resource-constrained devices, garnering widespread attention. This paper has provided a thorough review of the latest developments in LFT methods for model compression on various NNs. Specifically, this review has summarized various LFT approaches used for compressing the model parameters of CNNs, RNNs, and Transformers. The progression of prominent training strategies and widely adopted LFT tools has been outlined. Several popular LFT models including TSVD, CP, TK, TR, and TT have been empirically validated on four datasets to demonstrate their performance. Future research directions for LFT-based model compression have been pointed out to give readers a clear view of future trends in model compression.

CRediT authorship contribution statement

Yaping He: Writing – original draft, Visualization, Software, Methodology. **Hao Wu:** Writing – review & editing, Methodology, Conceptualization. **Weibo Liu:** Writing – review & editing, Conceptualization. **Xin Luo:** Writing – original draft, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Science and Technology Innovation Key R&D Program of Chongqing under Grant CSTB2025TIAD-STX0032, the National Key Research and Development Program of China under Grant 2024YFF0908200, the Chongqing Technology Innovation and Application Development Special Key Project under Grant CSTB2024TIAD-KPX0018, the Royal Society of the UK under Grant IES\R3\243021, and the Southwest University Graduate Student Research Innovation Grant SWUB24051.

Data availability

No data was used for the research described in the article.

References

[1] J. Gou, B. Yu, S.J. Maybank, D. Tao, Knowledge distillation: a survey, Int. J. Comput. Vis. 129 (2021) 1789–1819.
[2] Y. Wang, H. Zhang, X. Zeng, B. Wang, W. Li, W. Ding, Binary lightweight neural networks for arbitrary scale super-resolution of remote sensing images, IEEE Trans. Geosci. Remote Sens. (2025), https://doi.org/10.1109/TGRS.2025.3529696
[3] D. Ji, F. Zhao, H. Lu, F. Wu, J. Ye, Structural and statistical texture knowledge distillation and learning for segmentation, IEEE Trans. Pattern Anal. Mach. Intell. (2025), https://doi.org/10.1109/TPAMI.2025.3536481
[4] H. Zhang, L. Liu, Y. Zhang, X. Lei, F. Hui, B. Wen, DenseKD: dense knowledge distillation by exploiting region and sample importance, IEEE Trans. Neural Netw. Learn. Syst. (2025), https://doi.org/10.1109/TNNLS.2025.3525737
[5] Y. Qian, X. Wang, F. Sun, L. Pan, Compressing transfer: mutual learning-empowered knowledge distillation for temporal knowledge graph reasoning, IEEE Trans. Neural Netw. Learn. Syst. (2025), https://doi.org/10.1109/TNNLS.2025.3525699
[6] Z. Tu, W. Zhou, X. Qian, W. Yan, Hybrid knowledge distillation network for RGB-D co-salient object detection, IEEE Trans. Syst. Man Cybern. Syst. (2025), https://doi.org/10.1109/TSMC.2025.3526234
[7] D. Ren, W. Li, J. Huo, L. Wang, H. Pan, Y. Gao, Leveraging frequency analysis for image denoising network pruning, IEEE Trans. Image Process. 34 (2025) 1660–1671.
[8] Y. Feng, B. Ma, D. Liu, Y. Zhang, W. Cai, Y. Xia, Contrastive neuron pruning for backdoor defense, IEEE Trans. Image Process. 34 (2025) 1234–1245.
[9] X. Chu, L. Li, B. Zhang, Make RepVGG greater again: a quantization-aware approach, in: Proc. AAAI Conf. Artif, Intell, 2024, pp. 11624–11632.
[10] Y. Shang, Z. Yuan, B. Xie, B. Wu, Y. Yan, Post-training quantization on diffusion models, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit, 2023, pp. 1972–1981.
[11] H. Zhu, H. Ni, S. Liu, G. Xu, L. Deng, TNLRS: target-aware non-local low-rank modeling with saliency filtering regularization for infrared small target detection, IEEE Trans. Image Process. 29 (2020) 9546–9558.
[12] E.L. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, Adv. Neural Inf. Process. Syst. 27 (2014).
[13] S. Zhang, Y. Sun, D. Ding, H. Yu, SMHNet: self-supervised multiscale hierarchical network for high fidelity 3-D face reconstruction, IEEE Trans. Hum.-Mach. Syst. 56 (1) (2026) 114–123.
[14] W. Tan, H. Zhang, Y. Wang, W. Wen, L. Chen, H. Li, X. Gao, N. Zeng, SEDA-EEG: a semi-supervised emotion recognition network with domain adaptation for cross-subject EEG analysis, Neurocomputing 622 (2025) 129315.
[15] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, X. Xie, J. Han, Towards large-scale small object detection: survey and benchmarks, IEEE Trans. Pattern Anal. Mach. Intell. 45 (2023) 13467–13488.
[16] S. Wang, X. Zheng, Y. Zhang, R. Lan, Y. Zheng, MDFusion: a multistage dynamic fusion framework for multimodal 3D object detection with leveraging cross-modal feature complementarity, Expert Syst. Appl. 297 (2026) 129444.
[17] Y. Liu, Y. Zhang, R. Lan, X. Cui, L. Xie, Z. Wu, Adaptive temporal fusion network with depth supervision and modulation for robust three-dimensional object detection in complex scenes, Eng. Appl. Artif. Intell. 144 (2025) 109988.
[18] P. Wu, W. Wen, H. Li, Z. Li, N. Zeng, AI-driven automation of aviation equipment inspection: insights from a complex adaptive systems perspective, The Innovation 7 (1) (2026) 10184.
[19] H. Chen, P. Wu, W. Wen, N. Zeng, DLA-Net: a dynamically learnable attention network for intelligent surface visual inspection of aero-engine blades, IEEE Trans. Instrum. Meas. 74 (2025) 3532114.
[20] V.I. Butoi, J.J.G. Ortiz, T. Ma, M.R. Sabuncu, J. Guttag, A.V. Dalca, UniverSeg: universal medical image segmentation, in: Proc. IEEE/CVF Int. Conf. Comput, Vis, 2023, pp. 21438–21451.
[21] L. Zhang, B. Wang, Y. Zhao, Y. Yuan, T. Zhou, Z. Li, Collaborative multimodal fusion network for multiagent perception, IEEE Trans. Cybern. 55 (2025) 486–498.
[22] R. Zhang, S. Zhang, Rethinking influence functions of neural networks in the over-parameterized regime, in: Proc. AAAI Conf. Artif, Intell, 2022, pp. 9082–9090.
[23] P. Wu, H. Li, L. Hu, J. Ge, N. Zeng, A local-global attention fusion framework with tensor decomposition for medical diagnosis, in: IEEE/CAA J. Autom. Sinica, vol. 11, 2024, pp. 1536–1538.
[24] S. Li, J. Xu, P. Liu, X. Li, P. Wang, X. Jin, S. Yao, Truncated Lanczos-TSVD: an effective dimensionality reduction algorithm for detecting DDoS attacks in large-scale networks, IEEE Trans. Netw. Sci. Eng. 11 (2024) 4689–4703.
[25] J. Huang, L. Cui, Tensor singular spectrum decomposition: multisensor denoising algorithm and application, IEEE Trans. Instrum. Meas. 72 (2023) 1–15.
[26] T. Liu, J. Yang, B. Li, Y. Wang, W. An, Infrared small target detection via nonconvex tensor tucker decomposition with factor prior, IEEE Trans. Geosci. Remote Sens. 61 (2023) 1–17.
[27] M. Wang, D. Hong, Z. Han, J. Li, J. Yao, L. Gao, B. Zhang, J. Chanussot, Tensor decompositions for hyperspectral data processing in remote sensing: a comprehensive review, IEEE Geosci. Remote Sens. Mag. 11 (2023) 26–72.
[28] M. Feng, W. Chen, Y. Yang, Q. Shu, H. Li, Y. Huang, Hyperspectral anomaly detection based on tensor ring decomposition with factors TV regularization, IEEE Trans. Geosci. Remote Sens. 61 (2023) 1–14.
[29] Y. Lin, G. Hue, L. Wang, Q. Li, J. Zhu, A multi-AGV routing planning method based on deep reinforcement learning and recurrent neural network, in: IEEE/CAA J. Autom. Sin, vol. 11, 2023, pp. 1720–1722.
[30] H. Tian, T. Deng, H. Yan, Driving as well as on a sunny day? Predicting driver's fixation in rainy weather conditions via a dual-branch visual model, in: IEEE/CAA J. Autom. Sin, vol. 9, 2022, pp. 1335–1338.
[31] Y. Wang, K. Li, Z. Chen, Battery full life cycle management and health prognosis based on cloud service and broad learning, in: IEEE/CAA J. Autom. Sin, vol. 9, 2022, pp. 1540–1542.
[32] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. Li, Y. Chen, Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2020, pp. 678–679.
[33] X. Yu, T. Liu, X. Wang, D. Tao, On compressing deep models by low rank and sparse decomposition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit, 2017, pp. 7370–7379.
[34] X. Zhang, J. Zou, K. He, J. Sun, Accelerating very deep convolutional networks for classification and detection, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2015) 1943–1955.
[35] D. Liu, L.T. Yang, P. Wang, R. Zhao, Q. Zhang, TT-TSVD: a multi-modal tensor train decomposition with its application in convolutional neural networks for smart healthcare, ACM Trans. Multim. Comput. Commun. Appl. 18 (2022) 1–17.
[36] K. Guo, Z. Lin, C. Chen, X. Xing, F. Liu, X. Xu, Compact model training by low-rank projection with energy transfer, IEEE Trans. Neural Netw. Learn. Syst. 36 (2024) 6708–6722.
[37] Y. Zhou, H. Li, S. Du, J. Yao, Y. Zhang, Y. Wang, Low-rank knowledge decomposition for medical foundation models, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit, 2024, pp. 11611–11620.
[38] S. Rajbhandari, H. Shrivastava, Y. He, AntMan: sparse low-rank compression to accelerate RNN inference, arXiv preprint arXiv:1910.01740, 2019.
[39] X. Wang, L.T. Yang, E. Cao, L. Guo, L. Ren, M.J. Deen, A tensor-based T-SVD-LSTM remaining useful life prediction model for industrial intelligence, IEEE Trans. Ind. Inform. (2022), https://doi.org/10.1109/TII.2022.3220854
[40] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, W. Gao, Post-training quantization for vision transformer, Adv. Neural Inf. Process. Syst. 34 (2021) 28092–28103.
[41] P. Liu, Z.-F. Gao, W.X. Zhao, Z.-Y. Xie, Z.-Y. Lu, J.-R. Wen, Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators, arXiv preprint arXiv:2106.02205, 2021.
[42] P. Sharma, J.T. Ash, D. Misra, The truth is in there: improving reasoning in language models with layer-selective rank reduction, arXiv preprint arXiv:2312.13558, 2023.
[43] P. Chen, H.-F. Yu, I. Dhillon, C.-J. Hsieh, DRONE: data-aware low-rank compression for large NLP models, Adv. Neural Inf. Process. Syst. 34 (2021) 29321–29334.
[44] M. Makni, K. Behdin, Z. Xu, N. Ponomareva, R. Mazumder, Hassle-free: a unified framework for sparse plus low-rank matrix decomposition for LLMs, arXiv preprint arXiv:2502.00899, 2025.
[45] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavsk'y, V. Glukhov, I. Oseledets, A. Cichocki, Stable low-rank tensor decomposition for compression of convolutional neural network, in: Eur. Conf. Comput. Vis, Springer, 2020, pp. 522–539.
[46] X. Ruan, Y. Liu, C. Yuan, B. Li, W. Hu, Y. Li, S. Maybank, EDP: an efficient decomposition and pruning scheme for convolutional neural network compression, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 4499–4513.
[47] Y. Ji, Q. Wang, Fast CP-compression layer: tensor CP-decomposition to compress layers in deep learning, IET Image Process. 16 (2022) 2535–2543.

[48] Y. Wang, G. Weihong, X. Yue, CPAC-Conv: CP-decomposition to approximately compress convolutional layers in deep learning, arXiv preprint arXiv:2005.13746, 2020.

[49] D. Wang, B. Wu, G. Zhao, M. Yao, H. Chen, L. Deng, T. Yan, G. Li, Kronecker CP decomposition with fast multiplication for compressing RNNs, IEEE Trans. Neural Netw. Learn. Syst. 34 (2021) 2205–2219.

[50] Q. Zhang, M. Zhang, M. Wang, W. Sui, C. Meng, J. Yang, W. Kong, X. Cui, W. Lin, Efficient deep learning inference based on model compression, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, 2018, pp. 1695–1702.

[51] J. Gu, B. Keller, J. Kossaifi, A. Anandkumar, B. Khailany, D.Z. Pan, arXiv preprint arXiv:2211.16749, 2022.

[52] J. Xiao, C. Zhang, Y. Gong, M. Yin, Y. Sui, L. Xiang, D. Tao, B. Yuan, Haloc: hardware-aware automatic low-rank compression for compact neural networks, in: Proc. AAAI Conf. Artif, Intell, 2023, pp. 10464–10472.

[53] V.T. Pham, Y. Zniyed, T.P. Nguyen, Efficient tensor decomposition-based filter pruning, Neural Netw. 178 (2024).

[54] Z. Zhong, F. Wei, Z. Lin, C. Zhang, ADA-Tucker: compressing deep neural networks via adaptive dimension adjustment tucker decomposition, Neural Netw. 110 (2019) 104–115.

[55] J. Gusak, M. Kholiavchenko, E. Ponomarev, L. Markeeva, P. Blagoveschensky, A. Cichocki, I. Oseledets, Automated multi-stage compression of neural networks, in: Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops, 2019, pp. 1–8.

[56] B.-S. Chu, C.-R. Lee, Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization, arXiv preprint arXiv:2112.03690, 2021.

[57] G.G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao, D.P. Mandic, Compression and interpretability of deep neural networks via tucker tensor layer: from first principles to tensor valued back-propagation, arXiv preprint arXiv:1903.06133, 2019.

[58] M. Yin, H. Phan, X. Zang, S. Liao, B. Yuan, Batude: budget-aware neural network compression based on Tucker decomposition, in: Proc. AAAI Conf. Artif, Intell, 2022, pp. 8874–8882.

[59] C. Liu, K. Xie, J. Wen, G. Xie, K. Li, An accuracy-preserving neural network compression via tucker decomposition, IEEE Trans. Sustain. Comput. 10 (2025) 262–273.

[60] H. Phan, M. Yin, Y. Sui, B. Yuan, S. Zonouz, Cstar: towards compact and structured deep neural networks with adversarial robustness, in: Proc. AAAI Conf. Artif, Intell, 2023, pp. 2065–2073.

[61] L. Xiang, M. Yin, C. Zhang, A. Sukumaran-Rajam, P. Sadayappan, B. Yuan, D. Tao, TDC: towards extremely efficient CNNs on GPUs via hardware-aware tucker decomposition, in: Proc. ACM SIGPLAN Annu. Symp. Princ. Pract. Parallel Program, 2023, pp. 260–273.

[62] H. Ding, K. Chen, Q. Huo, Compressing CNN-DBLSTM models for OCR with teacher-student learning and tucker decomposition, Pattern Recognit. 96 (2019) 106957.

[63] Y. Ouyang, K. Xie, J. Wen, G. Xie, K. Li, A robust low-rank tensor decomposition and quantization based compression method, in: IEEE Int. Conf. Data Eng, IEEE, 2024, pp. 995–1008.

[64] M. Yin, S. Liao, X.-Y. Liu, X. Wang, B. Yuan, Towards extremely compact RNNs for video recognition with fully decomposed hierarchical tucker structure, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit, 2021, pp. 12085–12094.

[65] Y. Ren, B. Wang, L. Shang, X. Jiang, Q. Liu, Exploring extreme parameter compression for pre-trained language models, arXiv preprint arXiv:2205.10036, 2022.

[66] C. Hawkins, H. Yang, M. Li, L. Lai, V. Chandra, arXiv preprint arXiv:2111.01697, 2021.

[67] W. Wang, Y. Sun, B. Eriksson, W. Wang, V. Aggarwal, Wide compression: tensor ring nets, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit, 2018, pp. 9329–9338.

[68] W. Wang, V. Aggarwal, S. Aeron, Efficient low rank tensor ring completion, in: Proc. IEEE Int. Conf. Comput, Vis, 2017, pp. 5697–5705.

[69] N. Li, Y. Pan, Y. Chen, Z. Ding, D. Zhao, Z. Xu, Heuristic rank selection with progressively searching tensor ring network, Complex Intell. Syst. (2021) 1–15.

[70] Y. Pan, J. Xu, M. Wang, J. Ye, F. Wang, K. Bai, Z. Xu, Compressing recurrent neural networks with tensor ring for action recognition, in: Proc. AAAI Conf. Artif, Intell, 2019, pp. 4683–4690.

[71] W. Zou, Z. Lu, Z. Hu, L. Mao, Remaining useful life estimation of bearing using deep multiscale window-based transformer, IEEE Trans. Instrum. Meas. 72 (2023) 1–11.

[72] J. Tang, K. Li, M. Hou, X. Jin, W. Kong, Y. Ding, Q. Zhao, MMT: multi-way multi-modal transformer for multimodal learning, in: Int. Jt. Conf. Artif. Intell., 2022, pp. 3458–3465.

[73] M. Yin, Y. Sui, W. Yang, X. Zang, Y. Gong, B. Yuan, HODEC: towards efficient high-order decomposed convolutional neural networks, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit, 2022, pp. 12299–12308.

[74] A. Novikov, D. Podoprikhin, A. Osokin, D.P. Vetrov, Tensorizing neural networks, Adv. Neural Inf. Process. Syst. 28 (2015).

[75] T. Garipov, D. Podoprikhin, A. Novikov, D. Vetrov, Ultimate tensorization: compressing convolutional and FC layers alike, arXiv preprint arXiv:1611.03214, 2016.

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

[77] O.A. Ademola, P. Eduard, L. Mairo, Ensemble of tensor train decomposition and quantization methods for deep learning model compression, in: Int. Joint Conf. Neural Netw, IEEE, 2022, pp. 1–6.

[78] Z. Zha, B. Wen, X. Yuan, S. Ravishankar, J. Zhou, C. Zhu, Learning nonlocal sparse and low-rank models for image compressive sensing: nonlocal sparse and low-rank modeling, IEEE Signal Process. Mag. 40 (2023) 32–44.

[79] Y. Sui, M. Yin, Y. Gong, B. Yuan, Co-exploring structured sparsification and low-rank tensor decomposition for compact DNNs, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 6642–6654.

[80] Y. Gong, M. Yin, L. Huang, J. Xiao, Y. Sui, C. Deng, B. Yuan, ETTE: efficient tensor-train-based computing engine for deep neural networks, in: Proc. 50th Annu. Int. Symp. Comput. Archit, 2023, pp. 1–13.

[81] D. Wang, G. Zhao, H. Chen, Z. Liu, L. Deng, G. Li, Nonlinear tensor train format for deep neural network compression, Neural Netw. 144 (2021) 320–333.

[82] D. Wang, G. Zhao, G. Li, L. Deng, Y. Wu, Compressing 3DCNNs based on tensor train decomposition, Neural Netw. 131 (2020) 215–230.

[83] Q. Wu, Z. Jiang, K. Hong, H. Liu, L.T. Yang, J. Ding, Tensor-based recurrent neural network and multi-modal prediction with its applications in traffic network management, IEEE Trans. Netw. Serv. Manag. 18 (2021) 780–792.

[84] Y.L. Xu, G.G. Calvi, D.P. Mandic, Tensor-train recurrent neural networks for interpretable multi-way financial forecasting, in: Int. Joint Conf. Neural Netw, IEEE, 2021, pp. 1–5.

[85] Z. Bai, Y. Li, M. Woźniak, M. Zhou, D. Li, DecomVQANet: decomposing visual question answering deep network via tensor decomposition and regression, Pattern Recognit. 110 (2021) 107538.

[86] T.-Y. Ma, H.-C. Li, Y.-B. Zheng, Q. Du, A. Plaza, Fully tensorized lightweight ConvLSTM neural networks for hyperspectral image classification, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 14213–14227.

[87] S. Jie, Z.-H. Deng, Fact: factor-tuning for lightweight adaptation on vision transformer, in: Proc. AAAI Conf. Artif, Intell, 2023, pp. 1060–1068.

[88] H. Li, J. Zhao, H. Huo, S. Fang, J. Chen, L. Yao, Y. Hua, T3SRS: tensor train transformer for compressing sequential recommender systems, Expert Syst. Appl. 238 (2024) 122260.

[89] O. Hrinchuk, V. Khrulkov, L. Mirvakhabova, E. Orlova, I. Oseledets, Tensorized embedding layers for efficient model compression, arXiv preprint arXiv:1901.10787, 2019.

[90] J. Zhao, F. Zhuo, Q. Sun, Q. Li, Y. Hua, J. Zhao, DSFormer-LRTC: dynamic spatial transformer for traffic forecasting with low-rank tensor compression, IEEE Trans. Intell. Transp. Syst. 25 (2024) 16323–16335.

[91] Y. Panagakis, J. Kossaifi, G.G. Chrysos, J. Oldfield, M.A. Nicolaou, A. Anandkumar, S. Zafeiriou, Tensor methods in computer vision and deep learning, in: Proc. IEEE, vol. 109, 2021, pp. 863–890.

[92] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. 13th Int. Conf. Artif. Intell. Stat., J. Mach. Learn. Res. Workshop, 2010, pp. 249–256.

[93] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: Proc. IEEE Int. Conf. Comput. Vis, 2015, pp. 1026–1034.

[94] Y. Pan, Z. Su, A. Liu, W. Jingquan, N. Li, Z. Xu, A unified weight initialization paradigm for tensorial convolutional neural networks, in: Int. Conf. Mach. Learn, Proc. Mach. Learn, Res, 2022, pp. 17238–17257.

[95] Z. Xu, Y. Chen, K. Vishniakov, Y. Yin, Z. Shen, T. Darrell, L. Liu, Z. Liu, Initializing models with larger ones, arXiv preprint arXiv:2311.18823, 2023.

[96] M. Khodak, N. Tenenholtz, L. Mackey, N. Fusi, Initialization and regularization of factorized neural layers, arXiv preprint arXiv:2105.01029, 2021.

[97] Z. Cheng, B. Li, Y. Fan, Y. Bao, A novel rank selection scheme in tensor ring decomposition based on reinforcement learning for deep neural networks, in: IEEE Int. Conf. Acoust. Speech Signal Process, IEEE, 2020, pp. 3292–3296.

[98] Y. He, H. Wu, X. Luo, Adaptive tucker decomposition-based progressive model compression for convolutional neural networks, Expert Syst. Appl. (2026) 131153.

[99] B.M. Damon, Z. Ding, M.T. Hooijmans, A.W. Anderson, X. Zhou, C.L. Coolbaugh, M.K. George, B.A. Landman, A matlab toolbox for muscle diffusion-tensor MRI tractography, J. Biomech. 124 (2021) 110540.

[100] N. Vervliet, O. Debals, L. De Lathauwer, Tensorlab 3.0—numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization, in: Asilomar Conf. Signals Syst. Comput, IEEE, 2016, pp. 1733–1738.

[101] T. Yokota, N. Lee, A. Cichocki, Robust multilinear tensor rank estimation using higher order singular value decomposition and information criteria, IEEE Trans. Signal Process. 65 (2016) 1196–1206.

[102] S.V.D. Walt, S.C. Colbert, G. Varoquaux, The NumPy array: a structure for efficient numerical computation, Comput. Sci. Eng. 13 (2011) 22–30.

[103] A. Novikov, P. Izmailov, V. Khrulkov, M. Figurnov, I. Oseledets, Tensor train decomposition on tensorflow (T3F), J. Mach. Learn. Res. 21 (2020) 1–7.

[104] L. Hao, S. Liang, J. Ye, Z. Xu, TensorD: a Tensor decomposition library in TensorFlow, Neurocomputing 318 (2018) 196–200.

[105] J. Kossaifi, Y. Panagakis, A. Anandkumar, M. Pantic, TensorLy: tensor learning in Python, J. Mach. Learn. Res. 20 (2019) 1–6.

[106] C. Roberts, A. Milsted, M. Ganahl, A. Zalcman, B. Fontaine, Y. Zou, J. Hidary, G. Vidal, S. Leichenauer, TensorNetwork: a library for physics and machine learning, arXiv preprint arXiv:1905.01330, 2019.

[107] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems, arXiv preprint arXiv:1512.01274, 2015.

[108] J. Huang, L. Kong, X.-Y. Liu, W. Qu, G. Chen, A C++ library for tensor decomposition, in: IEEE Int. Perform. Comput. Commun. Conf, IEEE, 2019, pp. 1–2.

[109] M. Fishman, S. White, E.M. Stoudenmire, The ITensor software library for tensor network calculations, SciPost Phys. Codebases (2022) 4, https://doi.org/10.21468/SciPostPhysCodeb.4

[110] Y. Pan, M. Wang, Z. Xu, TedNet: a pytorch toolkit for tensor decomposition networks, Neurocomputing 469 (2022) 234–238.

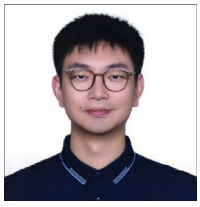
[111] W. Xu, C. Zhao, J. Cheng, Y. Wang, Y. Tang, T. Zhang, Z. Yuan, Y. Lv, F.-Y. Wang, Transformer-based macroscopic regulation for high-speed railway timetable rescheduling, in: IEEE/CAA J. Autom. Sin, vol. 10, 2023, pp. 1822–1833.
[112] Z. Zhang, Z. Lei, M. Omura, H. Hasegawa, S. Gao, Dendritic learning-incorporated vision transformer for image recognition, in: IEEE/CAA J. Autom. Sin, vol. 11, 2024, pp. 539–541.
[113] Y. Liu, B. Tian, Y. Lv, L. Li, F.-Y. Wang, Point cloud classification using content-based transformer via clustering in feature space, in: IEEE/CAA J. Autom. Sin, vol. 11, 2023, pp. 231–239.
[114] J. Ma, L. Tang, F. Fan, J. Huang, X. Mei, Y. Ma, SwinFusion: cross-domain long-range learning for general image fusion via swin transformer, in: IEEE/CAA J. Autom. Sin, vol. 9, 2022, pp. 1200–1217.
[115] L. Deng, G. Li, S. Han, L. Shi, Y. Xie, Model compression and hardware acceleration for neural networks: a comprehensive survey, in: Proc. IEEE, vol. 108, 2020, pp. 485–532.
[116] X. Liu, K.K. Parhi, Tensor decomposition for model reduction in neural networks: a review [feature], IEEE Circuits Syst. Mag. 23 (2023) 8–28.
[117] B. Zhao, X. Li, X. Lu, TTH-RNN: tensor-train hierarchical recurrent neural network for video summarization, IEEE Trans. Ind. Electron. 68 (2020) 3629–3637.
[118] J. Su, W. Byeon, J. Kossaifi, F. Huang, J. Kautz, A. Anandkumar, Convolutional tensor-train LSTM for spatio-temporal learning, Adv. Neural Inf. Process. Syst. 33 (2020) 13714–13726.
[119] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555, 2014.
[120] S. Hazmoune, F. Bougamouza, Using transformers for multimodal emotion recognition: taxonomies and state of the art review, Eng. Appl. Artif. Intell. 133 (2024) 108339.
[121] D. Han, X. Pan, Y. Han, S. Song, G. Huang, FLatten transformer: vision transformer using focused linear attention, in: Proc. IEEE/CVF Int. Conf. Comput, Vis, 2023, pp. 5961–5971.
[122] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, W. Furu, G. Baining, Swin transformer V2: scaling up capacity and resolution, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit, 2022, pp. 12009–12019.
[123] Z. Yu, C.-S. Bouganis, SVD-NAS: coupling low-rank approximation and neural architecture search, in: Proc. IEEE/CVF Winter Conf. Appl. Comput, Vis, 2023, pp. 1503–1512.
[124] B. Chen, J. Guan, Z. Li, Unsupervised feature selection via graph regularized non-negative CP decomposition, IEEE Trans. Pattern Anal. Mach. Intell. 45 (2022) 2582–2594.
[125] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, V. Lempitsky, Speeding-up convolutional neural networks using fine-tuned CP-decomposition, arXiv preprint arXiv:1412.6553, 2014.
[126] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, arXiv preprint arXiv:1511.06530, 2015.
[127] S. Hu, Z. Liu, M. Li, X. He, CADM+: confusion-based learning framework with drift detection and adaptation for real-time safety assessment, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 5126–5139.
[128] J. Dou, Y. Song, H. Yu, Hierarchical oversampling based on Cohen's criterion for imbalanced data with missing information, IEEE Trans. Comput. Social Syst. 12 (2025) 3143–3155.
[129] W. Wang, H. Liu, W. Xu, W. Yu, Multiagent distributional reinforcement learning with dynamic hyper policy network for residential microgrid load scheduling, IEEE Trans. Ind. Informat. 22 (3) (2026) 2232–2242.
[130] X. Ma, Y. Yuan, L. Guo, Hierarchical reinforcement learning for UAV-PE game with alternative delay update method, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 4639–4651.
[131] J. Fang, Z. Wang, W. Liu, N. Zeng, Y. He, Y. Cao, L. Chen, X. Liu, Learning with noisy labels for industrial time series outlier detection: a transformer-embedded contrastive learning framework, IEEE Trans. Ind. Informat. (2025). early access, https://doi.org/10.1109/TII.2025.3616850
[132] Z. Han, Y. Zhou, Y. Jiang, K. Bao, W. Yue, Distributed adaptive consensus control for nonlinear network systems with event-based switching mechanism against malicious attacks, IEEE Trans. Autom. Sci. Eng. 22 (2025) 8329–8340.
[133] J. Mi, H. Wu, W. Li, X. Luo, Spatio-temporal traffic data recovery via latent factorization of tensors based on tucker decomposition, in: IEEE Int. Conf. Syst. Man Cybern, IEEE, 2023, pp. 1512–1517.
[134] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, D. Song, A tensorized transformer for language modeling, Adv. Neural Inf. Process. Syst. 32 (2019).
[135] V. Bharadwaj, B. Toloueirakhshan, O.A. Malik, G. Rabusseau, Efficient leverage score sampling for tensor train decomposition, Adv. Neural Inf. Process. Syst. 37 (2024) 73726–73744.
[136] H. Huang, H. Yu, LTNN: a layerwise tensorized compression of multilayer neural network, IEEE Trans. Neural Netw. Learn. Syst. 30 (2018) 1497–1511.
[137] C. Deng, F. Sun, X. Qian, J. Lin, Z. Wang, B. Yuan, TIE: energy-efficient tensor train-based inference engine for deep neural network, in: Proc. 46th Int. Symp. Comput. Archit, 2019, pp. 264–278.
[138] A.M. Grachev, D.I. Ignatov, A.V. Savchenko, Compression of recurrent neural networks for efficient language modeling, Appl. Soft Comput. 79 (2019) 354–362.
[139] A. Tjandra, S. Sakti, S. Nakamura, Tensor decomposition for compressing recurrent neural network, Int. Jt. Conf. Neural Netw. (2018) 1–8.
[140] Y. Yang, D. Krompass, V. Tresp, Tensor-train recurrent neural networks for video classification, in: Int. Conf. Mach. Learn, PMLR, 2017, pp. 3891–3900.
[141] P. Zhen, Z. Gao, T. Hou, Y. Cheng, H.-B. Chen, Deeply tensor compressed transformers for end-to-end object detection, in: Proc. AAAI Conf. Artif, Intell, 2022, pp. 4716–4724.
[142] S. Huang, T. Wang, A. Li, A. Shen, K. Li, K. Jiang, M. Huang, H. Yu, A tensor-train decomposition based compression of llms on group vector systolic accelerator, arXiv preprint arXiv:2501.19135, 2025.
[143] M. Wang, Y. Pan, Z. Xu, X. Yang, G. Li, A. Cichocki, Tensor networks meet neural networks: a survey and future perspectives, arXiv preprint arXiv:2302.09019, 2023.
[144] W. Zhang, Y. Fan, Y. Song, K. Tang, B. Li, A generalized two-stage tensor denoising method based on the prior of the noise location and rank, Expert Syst. Appl. 255 (2024) 124809.
[145] Y. Ma, D. Yu, T. Wu, H. Wang, PaddlePaddle: an open-source deep learning platform from industrial practice, Front. Data Comput. 1 (2019) 105–115.
[146] M. Usvyatsov, R. Ballester-Ripoll, K. Schindler, tntorch: tensor network learning with PyTorch, J. Mach. Learn. Res. 23 (2022) 1–6.
[147] I. Kisil, G.G. Calvi, B.S. Dees, D.P. Mandic, arXiv preprint arXiv:2111.15662, 2021.
[148] S. Saha, L. Xu, Vision transformers on the edge: a comprehensive survey of model compression and acceleration strategies, Neurocomputing (2025) 130417.
[149] W. Xiong, X. Wang, L. Zou, G. Chen, Designing iterative learning schemes for cooperative-antagonistic systems with random access communication protocols, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 12004–12015.
[150] Y. Gu, W. Zhou, G. Iacovides, D. Mandic, TeRA: vector-based random tensor network for high-rank adaptation of large language models, arXiv preprint arXiv:2509.03234, 2025.
[151] Y. Wang, H. Geng, K. Chen, Z. Liu, B. Xu, Bistable stochastic resonance for weak signal detection: constructing a new asymmetric unsaturated potential function, IEEE Trans. Instrum. Meas. 75 (2026) 1–13.
[152] Y. Liang, Z. Liu, X. Tang, Y. Cheng, H. Geng, Nonparametric Bayesian learning driven dynamic group sparse regularization for transient signal enhancement, IEEE Trans. Instrum. Meas. 74 (2025) 6506512.
[153] S. Huang, T. Wang, A. Li, A. Shen, K. Li, K. Jiang, M. Huang, H. Yu, A tensor-train decomposition based compression of llms on group vector systolic accelerator, arXiv preprint arXiv:2501.19135, 2025.
[154] A. D'sky, I. Syafalni, N. Sutisna, R. Mulyawan, N. Ahmadi, T. Adiono, Parallel R(2+1)D CNN acceleration cores using Winograd for fish appetite detection, IEEE Trans. AgriFood Electron. 3 (2025) 449–462.
[155] X. Luo, Z. Li, W. Yue, S. Li, A calibrator fuzzy ensemble for highly-accurate robot arm calibration, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 2169–2181.
[156] J. Xu, M. Zhong, L. Li, Y. Wu, B. Song, A fuzzy H_i/H_∞ optimization approach to fault detection of high-speed train traction motor systems, IEEE Trans. Ind. Informat. 21 (2025) 3655–3665.
[157] Y. Zhang, R. Guan, L. Li, R. Yang, Y. Yue, E.G. Lim, radarODE: an ODE-embedded deep learning model for contactless ECG reconstruction from millimeter-wave radar, IEEE Trans. Mobile Comput. 24 (2025) 9806–9821.
[158] Y. Zhang, R. Yang, Y. Yue, E.G. Lim, radarODE-MTL: a multitask learning framework with eccentric gradient alignment for robust radar-based ECG reconstruction, IEEE Trans. Instrum. Meas. 74 (2025) 4008315.
[159] J. Dai, J. Li, Z. Zhao, Z. Yang, Z. Zhang, M. Shikh-Bahaei, Energy efficient multi-modal probabilistic semantic communication (PSCom), IEEE Trans. Green Commun. Netw. (2025), https://doi.org/10.1109/TGCN.2025.3559014
[160] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, J. Pei, Trustworthy graph neural networks: aspects, methods, and trends, in: Proc. IEEE, vol. 112, 2024, pp. 97–139.
[161] K. Zhu, Z. Wang, D. Ding, J. Hu, H. Dong, Cloud-based collision avoidance adaptive cruise control for autonomous vehicles under external disturbances with token bucket shapers, IEEE Trans. Ind. Informat. 21 (2025) 8759–8769.
[162] Y. Wang, Z. Wang, L. Zou, Q. Ge, H. Dong, Asynchronous PID control for T-S fuzzy systems over Gilbert-Elliott channels utilizing detected channel modes, IEEE Trans. Fuzzy Syst. 33 (2025) 1555–1567.
[163] S. Yang, J. Hu, R. Caballero-Aguila, C. Jia, Tobit filtering for nonlinear systems with fading observations under periodic protocol via uniform dynamic encoding-decoding scheme, Commun. Nonlinear Sci. Numer. Simul. 152 (2026) 109111.
[164] J. Li, R. Caballero-Aguila, J. Hu, J. Linares-Perez, Recursive estimation with compensation strategies under random access protocol and deception attacks, IEEE Trans. Signal Process. 73 (2025) 1954–1965.
[165] W. Song, Z. Wang, Z. Li, Q.-L. Han, D. Yue, Maximum correntropy filtering for complex networks with uncertain dynamical bias: enabling componentwise event-triggered transmission, IEEE Trans. Neural Netw. Learn. Syst. 35 (2024) 17330–17343.
[166] W. Song, Z. Wang, Z. Li, Q.-L. Han, Particle-filter-based state estimation for delayed artificial neural networks: when probabilistic saturation constraints meet redundant channels, IEEE Trans. Neural Netw. Learn. Syst. 35 (2024) 4354–4362.
[167] L. Zou, Z. Wang, J. Hu, H. Gao, On H_∞ finite-horizon filtering under stochastic protocol: dealing with high-rate communication networks, IEEE Trans. Autom. Control 62 (2017) 4884–4890.
[168] L. Zou, Z. Wang, B. Shen, H. Dong, Encryption-decryption-based state estimation with multirate measurements against eavesdroppers: a recursive minimum-variance approach, IEEE Trans. Autom. Control 68 (2023) 8111–8118.
[169] W. Song, Z. Wang, Z. Li, J. Wang, Q.-L. Han, Nonlinear filtering with sample-based approximation under constrained communication: progress, insights and trends, IEEE/CAA J. Autom. Sinica 11 (2024) 1539–1556.
[170] L. Zou, Z. Wang, B. Shen, H. Dong, G. Lu, Encrypted finite-horizon energy-to-peak state estimation for time-varying systems under eavesdropping attacks: tackling secrecy capacity, in: IEEE/CAA J. Autom. Sinica, vol. 10, 2023, pp. 985–996.

[171] M. Shi, L. Ma, X. Yi, Recursive state estimation for complex networks with energy harvesting constraints and decode-and-forward relays, IEEE Trans. Syst. Man Cybern. Syst. 55 (2025) 5491–5502.
[172] C. Liang, D. He, C. Xu, Y. Chen, Distributed moving horizon estimation over energy harvesting wireless sensor networks: a switching topology approach, IEEE Trans. Circuits Syst. I, Reg. Papers 72 (2025) 8109–8119.
[173] T. Chen, C. Zhang, W. Jing, E.Y.S. Foo, L. Sun, N. Zeng, Distributed multi-agent fusion state estimation method based on finite-time average consensus for large-scale power systems, Inf. Fusion 127 (2026) 103753.
[174] M. Sheng, S. Chen, W. Liu, J. Mao, X. Liu, A differential evolution with adaptive neighborhood mutation and local search for multi-modal optimization, Neurocomputing 489 (2022) 309–322.

Author biography

Yaping He is currently pursuing the Ph.D. degree in Computer Science at the College of Computer and Information Science, Southwest University, Chongqing, China. His research interests include model compression and tensor low-rank representation learning.

Hao Wu (Member, IEEE) received the B.S. degree in Information Security from the Hefei University of Technology, Hefei, China, in 2014, the M.S. degree in Computer Science from the Chongqing University, Chongqing, China, in 2017, and the Ph. D. degree in Computer Science from the University of Chinese Academy of Sciences, Beijing, China, in 2022. He is currently an Associate Professor of Data Science at the College of Computer and Information Science, Southwest University, Chongqing, China. His research interests include Big Data Analytics and Tensor Methods.



Weibo Liu (Member, IEEE) received the B.Eng. degree in electrical engineering from the Department of Electrical Engineering & Electronics, University of Liverpool, Liverpool, U.K, in 2015, and the Ph.D. degree in artificial intelligence in 2020 from the Department of Computer Science, Brunel University of London, Uxbridge, U.K. He is currently a Lecturer in the Department of Computer Science, Brunel University London, Uxbridge, U.K. His research interests include intelligent data analysis, evolutionary computation, machine learning, deep learning and transfer learning. He serves as an Associate Editor for the Journal of Ambient Intelligence and Humanized Computing, the Journal of Cognitive Computation, the Journal of IEEE Transactions on Instrumentation and Measurement, and an Editorial Board Member for the Journal of Scientific Reports. He is a member of program committee for many international conferences and a very active reviewer for many international journals and conferences.

Xin Luo (Fellow, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Distinguished Professor of Data Science and Computational Intelligence, and serving as the Dean of the College of Computer and Information Science, and School of Software, Southwest University, Chongqing, China. He has authored or coauthored over 400 papers (including over 190 IEEE Transactions/Journal papers) in the areas of Artificial Intelligence and Data Science, receiving 20,000+ Google Scholar citations with the H-Index of 87. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE Access in 2018, IEEE/CAA Journal of Automatica Sinica in 2020, and from IEEE Transactions on Neural Networks and Learning Systems in 2022–2024. He is currently serving as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems, and IEEE/CAA Journal of Automatica Sinica. His Google Scholar page is given at the link https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-CN.