This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

Learning Accurate Representation to Nonstandard Tensors via a Mode-Aware Tucker Network

Hao Wu, *Member, IEEE*, Qu Wang, Xin Luo, *Fellow, IEEE*, and Zidong Wang, *Fellow, IEEE*

***Abstract*—A nonstandard tensor is frequently adopted to model a large-sale complex dynamic network. A Tensor Representation Learning (TRL) model enables extracting valuable knowledge form a dynamic network via learning low-dimensional representation of a target nonstandard tensor. Nevertheless, the representation learning ability of existing TRL models are limited for a nonstandard tensor due to its inability to accurately represent the specific nature of the nonstandard tensor, i.e., mode imbalance, high-dimension, and incompleteness. To address this issue, this study innovatively proposes a Mode-Aware Tucker Network-based Tensor Representation Learning (MTN-TRL) model with three-fold ideas: a) designing a mode-aware Tucker network to accurately represent the imbalanced mode of a nonstandard tensor, b) building an MTN-based high-efficient TRL model that fuses both data density-oriented modeling principle and adaptive parameters learning scheme, and c) theoretically proving the MTN-TRL model's convergence. Extensive experiments on eight nonstandard tensors generating from real-world dynamic networks demonstrate that MTN-TRL significantly outperforms state-of-the-art models in terms of representation accuracy.**

***Index Terms*—Tensor representation learning, nonstandard tensor, tensor network, dynamic network representation.**

I. Introduction

TENSORS are very versatile and powerful tools that is able to natively model large-scale complex dynamic networks such as a telecommunication network, a cryptocurrency trading network, and a traffic network, where the structural information of a dynamic network can be fully preserved and each element of tensor describes a certain interaction between a pair of network nodes [1], [2]. However, as the scale of the dynamic network continues to expand, a resultant tensor is normally high-dimensional and incomplete due to the impossibility of observing full interactions among all nodes at each time slot. Further, such a tensor in practical applications is commonly high imbalanced [3]–[5]. That is to say, such a resultant tensor with high-dimension, incompleteness, and mode imbalance properties is a typical nonstandard tensor. For example, Fig. 1 shows a "node×node×time" third-order nonstandard tensor $\mathbf{Y}$ that models a telecommunication network used in this study, where such a dynamic network involves 1,504,528 interactions among 308,200 terminal nodes (e.g., computers or sensor terminals) at 850 time slots (a time slot is ten minutes). As a result, the dimension of the nonstandard tensor $\mathbf{Y}$ comes to $(308{,}200 \times 308{,}200 \times 850)$ but contains only 1,504,528 known elements, correspondingly, its data density (i.e., known element count over total) is merely 1.86×10^{-8}. Obviously, the dimension of its mode-1 and mode-2 (i.e., the size of node modes), which is the total number of involved nodes, is very large compared to the dimension of mode-3, which is the total number of the time slots. Evidently, a nonstandard tensor contains rich knowledge regarding various characteristics of involved network nodes, like unobserved links and nodes' latent features [6]–[9]. Accurately and effectively acquiring such valuable knowledge is challenging, with the key issue being the accurate representation of a nonstandard tensor [10].

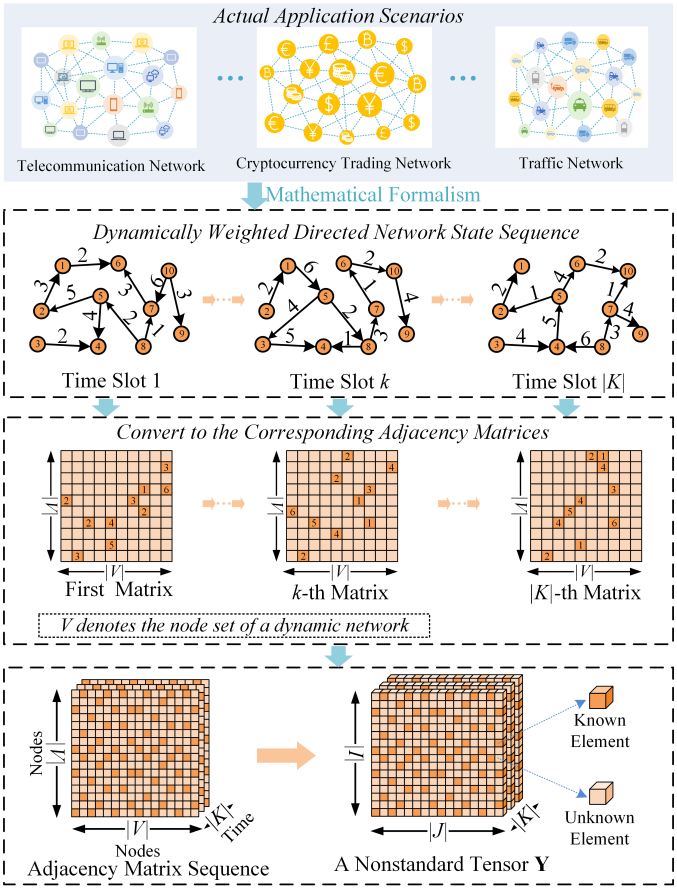

Fig. 1. The process of a nonstandard tensor models a large-scale dynamic network. (The $|K|$ adjacency matrices describing a dynamic network are sequentially connected to construct a nonstandard tensor. Each known element denotes a weighted directed link.)

This work is supported by the National Natural Science Foundation of China under grant 62302402, 62272078, and the Science and Technology Research Program of Chongqing Municipal Education Commission under grant KJQN202403017, KJZD-K202400209. (Corresponding authors: X. Luo).

H. Wu, Q. Wang, and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: haowuf@gmail.com, wangquff@gmail.com, luoxin@swu.edu.cn).

Z. Wang is with the Department of Computer Science, Brunel University London, Uxbridge UB8 3PH, United Kingdom. (E-mail: zidong.wang@brunel.ac.uk.)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

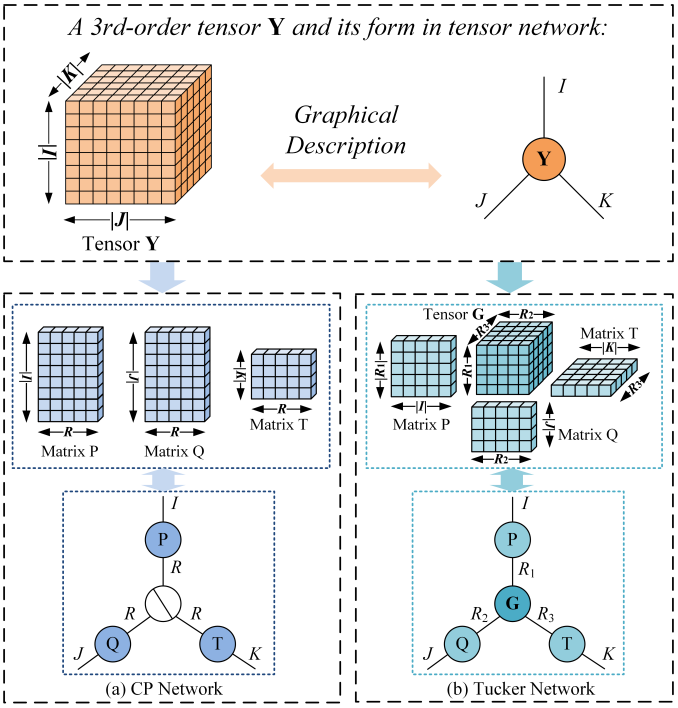

Fig. 2. Tensor network diagrams of CP and Tucker formats.

To date, researchers have proposed a variety of Tensor Representation Learning (TRL) models, which are constructed based on different tensor decomposition methods to achieve accurate representation of a target tensor. Among them, a Canonical Polyadic (CP) decomposition–based TRL model has gained significant popularity because of its concise mathematical form [11]. Notable examples include a Cauchy loss-based nonnegative tensor factorization model [12], a transfer learning-based biased tensor factorization model [13], and a graph regularized sparse CP decomposition model [14]. Specially, CP decomposition is a special form of Tucker decomposition when the Tucker core tensor is a super-diagonal tensor. Compared with CP does, Tucker decomposition normally obtains higher representation ability because it maintains a core tensor that can represents the relationship among latent factors [15], [16]. Accordingly, several Tucker decomposition-based TRL models have been built. Typical models include a L_∞-norm based nonnegative Tucker decomposition model [17], a biased nonnegative Tucker factorization model [18], and a graph regularized smooth nonnegative Tucker factorization model [19]. In addition to this, some other decomposition methods are also used to construct TRL models, e.g., tensor singular value decomposition and tensor training decomposition. However, the above models fail to account for mode imbalance, high-dimension, and incompleteness properties simultaneously, leading to suboptimal representation performance for a nonstandard tensor.

Tensor network is normally countable collections of low-order tensors that are interconnected by tensor contractions [20]–[22]. In particular, CP and Tucker decomposition are commonly used tensor network formats. Tensor network diagram is a straightforward graphical diagram for understanding the interconnected structures of tensor network, where a tensor is denoted graphically by a node with edges (an edge indicates a mode of the tensor and a value on an edge denotes the dimension of the corresponding mode). Fig. 2 illustrates tensor network diagrams of CP and Tucker formats. As shown in Fig. 2, an edge connecting two nodes means a type of tensor contraction operation, e.g., mode-n product. As a result, is it possible to construct a new edge connecting the two nodes, thus designing a new form of tensor network oriented to a nonstandard tensor for enhancing representation ability.

Inspired by the above findings, we innovatively design a Mode-Aware Tucker Network (MTN), which is a novel tensor network format for a nonstandard tensor. Accordingly, an MTN-based tensor representation learning model is developed for efficiently learning accurate representation of a nonstandard tensor, which adopts the following three-fold ideas:

1) Designing a novel mode-aware Tucker network, where a new edge is constructed on a tensor network of Tucker format to connects two latent factor (LF) nodes corresponding to two high-dimensional modes of a nonstandard tensor, thus, two high-dimensional modes are represented by two LF tensors while a low-dimensional mode is represented by a LF matrix.
2) Adopting data density-oriented modeling principle to build an MTN-based TRL model for learning the LFs accurately and efficiently, and implementing hyper-parameters self-adaptation via a differential evolution algorithm.
3) Theoretically proving that an MTN-TRL model can converge to a stationary point of its learning objective on a nonstandard tensor.

By doing so, this paper makes the following clear contributions:

1) A novel tensor network named MTN. It achieves fine-grained representations for a nonstandard tensor via constructing new tensor contraction operation between the latent factors of high-dimensional modes.
2) An MTN-TRL model. It can learn the low-dimensional representations of a nonstandard tensor accurately and efficiently by building learning objective on tensor known element set only.
3) The convergence proof of MTN-TRL. It demonstrates MTN-TRL can converge to a stable stationary point.

Experiments are performed on eight nonstandard tensors generated from real-world large-scale dynamic networks, which confirm that MTN is more accurate than other commonly used tensor network in representing a nonstandard tensor. At the same time, the results show that MTN-TRL has better performance on learning the representation of a nonstandard tensor compared with state-of-the-art models.

The rest is organized as follows: Section II introduces the preliminary knowledge, Section III proposes MTN and designs the MTN-TRL model in detail, Section IV conducts the experiments and analyzes the results, Section V reviews related work, and Section VI concludes this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

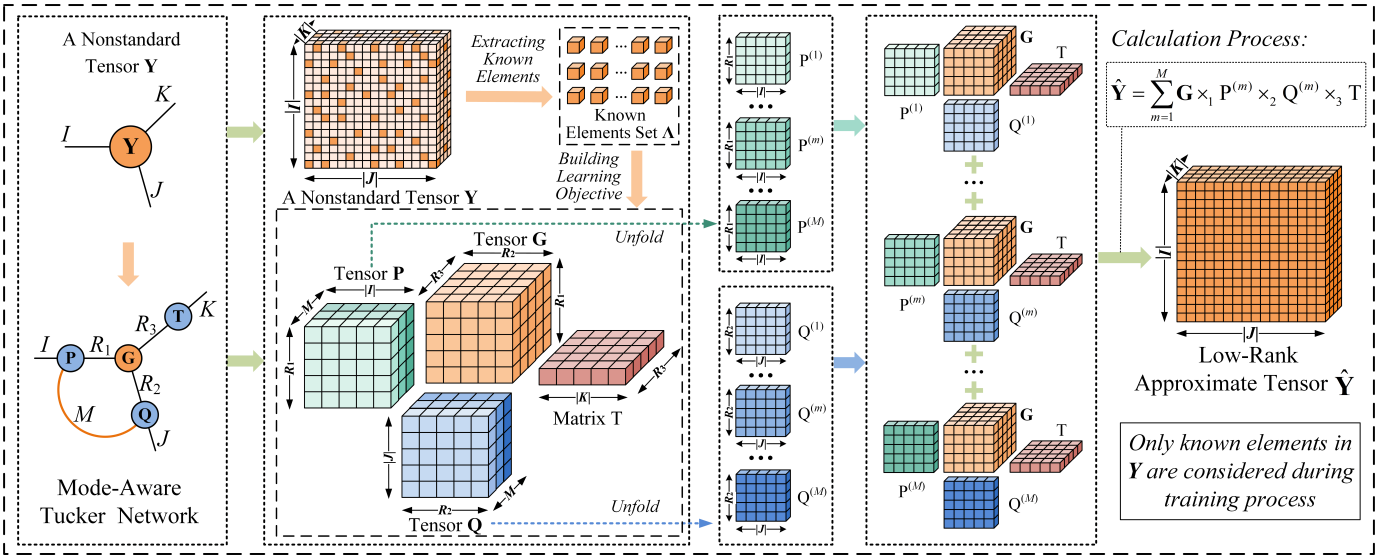

Fig. 3. The MTN diagram and the framework of MTN-TRL model.

TABLE I
SYMBOL AND DESCRIPTIONS

Symbols	Description
I, J, K	Three entity sets.
Λ	Known element entries set.
$\mathbb{R}$	Real number field.
R, R_1, R_2, R_3	Rank of CP and Tucker decomposition.
r_1, r_2, r_3	Index of R_1, R_2, R_3.
$\hat{\mathbf{Y}}$	A low-rank approximation tensor of $\mathbf{Y}$.
y_{ijk}, $\hat{y}_{ijk}$	Single element in tensor $\mathbf{Y}$, $\hat{\mathbf{Y}}$.
M	The size of the expanded dimension.
m, i, j, k	Index of $M, \lvert I\rvert, \lvert J\rvert, \lvert K\rvert$.
$\lvert*\rvert$	The size of the element collection.
$\lvert\lvert*\rvert\rvert_2$	Computes the L_2 norm of a tensor.
$\mathbf{G}$	A core tensor.
$\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$	Three bias vectors.
a_i, b_j, c_k	Single element in $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$.
λ	Regularization coefficient.
Ψ	Training set.
Ω	Validation set.
Φ	Testing set.

II. PRELIMINARIES

A. Notations

In tensor network, a one-edge node indicates a first-order tensor, i.e., a vector $\boldsymbol{y} \in \mathbb{R}^{I_1}$, a two-edge node indicates a second-order tensor, i.e., matrix $\mathrm{Y} \in \mathbb{R}^{I_1 \times I_2}$, and a three-edge node indicates a third-order tensor, i.e., tensor $\mathbf{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Table I summarizes the symbols used in this paper.

B. Tucker Network

Tucker network (i.e., a tensor network of Tucker format) represents a Nth-order tensor into a Nth-order core tensor multiplied by a matrix along each mode [16], [17]. As shown in Fig. 2(b), given a third-order tensor $\mathbf{Y} \in \mathbb{R}^{|I| \times |J| \times |K|}$, Tucker network can be formulated:

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{G} \times_1 \mathrm{P} \times_2 \mathrm{Q} \times_3 \mathrm{T}, \tag{1}$$

where $\mathbf{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ denotes a core tensor and $\mathrm{P} \in \mathbb{R}^{|I| \times R_1}, \mathrm{Q} \in \mathbb{R}^{|J| \times R_2}, \mathrm{T} \in \mathbb{R}^{|K| \times R_3}$ denote three LF matrices, the tensor contraction operators $\times 1$, $\times 2$ and $\times 3$ denote mode-n product, i.e., an edge connecting two nodes in tensor network diagram. Generally, $\hat{\mathbf{Y}}$ is called the rank-$\{R_1, R_2, R_3\}$ approximation of $\mathbf{Y}$, thus, to obtain the core tensor and LF matrices, a TRL model should be built to measure the difference between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, with normally adopted Euclidean distance, its learning objective is formulated as follows:

$$\begin{aligned}\varepsilon &= \left\|\mathbf{Y} - \hat{\mathbf{Y}}\right\|_F^2 = \sum_{i=1}^{|I|}\sum_{j=1}^{|J|}\sum_{k=1}^{|K|}\left(y_{ijk} - \hat{y}_{ijk}\right)^2 \\ &= \sum_{i=1}^{|I|}\sum_{j=1}^{|J|}\sum_{k=1}^{|K|}\left(y_{ijk} - \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{i r_1} q_{j r_2} t_{k r_3}\right)^2.\end{aligned} \tag{2}$$

III. METHODOLOGY

A. Mode-Aware Tucker Network

As shown in Fig. 2(b), a Tucker network fails to represent the mode imbalance of a nonstandard tensor, concretely, the high-dimensional mode and low-dimensional mode are uniformly represented by LF matrices. However, a high-dimensional mode typically contains more knowledge of the corresponding entities, thus requiring a larger representation space. Therefore, as shown in Fig. 3, considering a third-order nonstandard tensor $\mathbf{Y}$ with two high-dimensional modes, we construct a new edge between LF nodes P and Q, intuitively, the LF matrices $\mathrm{P} \in \mathbb{R}^{|I| \times R_1}$ and $\mathrm{Q} \in \mathbb{R}^{|J| \times R_2}$ are extended into two third-order LF tensors $\mathbf{P} \in \mathbb{R}^{M \times |I| \times R_1}$ and $\mathbf{Q} \in \mathbb{R}^{M \times |J| \times R_2}$, thereby granting the high-dimensional mode a larger representation space. As a result, a mode-aware Tucker network is achieved. It adopts LF tensor to represent high-dimensional mode and LF matrix represent low-dimensional mode, and a core tensor is still needed to represent the relationship among latent factors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

Accordingly, the proposed MTN oriented to a third-order nonstandard tensor $\mathbf{Y}$ can be expressed as:

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = [\![\mathbf{G}, \mathbf{P}, \mathbf{Q}, \mathrm{T}]\!] = \sum_{m=1}^{M} \mathbf{G} \times_1 \mathrm{P}^{(m)} \times_2 \mathrm{Q}^{(m)} \times_3 \mathrm{T}, \tag{3}$$

where M represents the dimension size of the shared order after expanding the matrices $\mathrm{P} \in \mathbb{R}^{|I| \times R_1}$ and $\mathrm{Q} \in \mathbb{R}^{|J| \times R_2}$ into tensors $\mathbf{P} \in \mathbb{R}^{M \times |I| \times R_1}$ and $\mathbf{Q} \in \mathbb{R}^{M \times |J| \times R_2}$, and $\mathrm{P}^{(m)}$ and $\mathrm{Q}^{(m)}$ respectively denote the *m*-th frontal slice of LF tensors $\mathbf{P}$ and $\mathbf{Q}$. Hence, each element in the tensor $\hat{\mathbf{Y}}$ is:

$$\hat{y}_{ijk} = \sum_{m=1}^{M} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{mir_1} q_{mjr_2} t_{kr_3}. \tag{4}$$

To acquire the desired representations of a nonstandard tensor, a tensor representation learning model is also necessary. Next, we present MTN-TRL model.

B. Objective Function of MTN-TRL

When learning the representation of a nonstandard tensor, as shown in (2), a traditional TRL model normally needs to pre-fill the unknown elements with artificial values to obtain a complete tensor as model input. Considering the high-dimension and incompleteness properties of a nonstandard tensor, such pre-filling strategy leads to unnecessarily high computational cost, and the artificially filled data may bend the learning process, causing loss of representation accuracy [10], [11]. As previous research [7], [11], by using the data density-oriented modeling principle, the objective function of a TRL model can be defined on the known element set Λ only of target nonstandard tensor $\mathbf{Y}$, which avoids efficiency and accuracy loss caused by the artificial pre-filling values. Therefore, for accurate and efficient learning the LF tensors $\mathbf{P}$ and $\mathbf{Q}$, LF matrix T, and core tensor $\mathbf{G}$, with the MTN, the TRL model's learning objective is obtained as:

$$\begin{aligned} \varepsilon &= \sum_{y_{ijk} \in \Lambda} (y_{ijk} - \hat{y}_{ijk})^2 \\ &= \sum_{y_{ijk} \in \Lambda} \left(y_{ijk} - \sum_{m=1}^{M} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{mir_1} q_{mjr_2} t_{kr_3} \right)^2. \end{aligned} \tag{5}$$

As stated in [18], [19], incorporating linear biases to a TRL model can improve its representation learning ability, as they effectively handle the magnitude differences in nonstandard tensor data. As reveal in [18], linear biases corresponding to the third-order nonstandard tensor $\mathbf{Y}$ can be represented by three linear bias vectors $\boldsymbol{a} \in \mathbb{R}^{|I|}, \boldsymbol{b} \in \mathbb{R}^{|J|}$, and $\boldsymbol{c} \in \mathbb{R}^{|K|}$, by integrating them to the learning objective, (5) is extended as:

$$\begin{aligned} \varepsilon = \sum_{y_{ijk} \in \Lambda} \left(y_{ijk} - \hat{y}_{ijk} \right)^2 = \sum_{y_{ijk} \in \Lambda} \Bigg(y_{ijk} - \Bigg(a_i + b_j + c_k \\ + \sum_{m=1}^{M} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{mir_1} q_{mjr_2} t_{kr_3} \Bigg) \Bigg)^2. \end{aligned} \tag{6}$$

Moreover, it is extremely important to introduce regularization into (6) for avoiding overfitting. With normally adopted L_2 regularization, we achieve the objective function of MTN-TRL model as:

$$\begin{aligned} \varepsilon = & \sum_{y_{ijk} \in \Lambda} \Bigg((y_{ijk} - a_i - b_j - c_k \\ & - \sum_{m=1}^{M} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{mir_1} q_{mjr_2} t_{kr_3} \Bigg)^2 \\ & + \lambda \left(\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3}^2 + a_i^2 + b_j^2 + c_k^2 \right) \\ & + \lambda \left(\sum_{m=1}^{M} \sum_{r_1=1}^{R_1} p_{mir_1}^2 + \sum_{m=1}^{M} \sum_{r_2=1}^{R_2} q_{mjr_2}^2 + \sum_{r_3=1}^{R_3} t_{kr_3}^2 \right) \Bigg) \\ s.t. \quad & \forall i \in I, j \in J, k \in K, m \in \{1, 2, ..., M\}, \\ & r_1 \in \{1, 2, ..., R_1\}, r_2 \in \{1, 2, ..., R_2\}, r_3 \in \{1, 2, ..., R_3\}. \end{aligned} \tag{7}$$

C. Adaptive Parameters Learning Scheme

Although a Stochastic Gradient Descent (SGD) algorithm can solve the MTN-TRL model, it normally suffers from slow convergence, thus leading to high computational cost. A momentum method can accelerate the SGD algorithm, which uses the gradient of the previous iteration to correct the current update direction and step size [23], [24]. Given an objective function $J(\omega)$, the update rule of parameter ω using Momentum-accelerated SGD (MSGD) algorithm as follows:

$$\begin{aligned} v_n &= \gamma v_{n-1} + \eta \nabla_\omega J(\omega_{n-1}), \\ \omega_n &= \omega_{n-1} - v_n, \end{aligned} \tag{8}$$

where γ denotes the momentum parameter, η denotes the learning rate, $\nabla_\omega J(\omega)$ indicates the gradient of the parameter ω, and v_n and v_{n-1} represent the velocity vectors at the *n*-th and (*n*-1)-th iterations respectively. Therefore, the gradients of each parameter in (7) are given:

$$\begin{aligned} \nabla_{g_{r_1 r_2 r_3}} \varepsilon(g_{r_1 r_2 r_3}) &= \lambda g_{r_1 r_2 r_3} - e_{ijk} \sum_{m=1}^{M} p_{imr_1} q_{mjr_2} t_{kr_3}, \\ \nabla_{p_{mir_1}} \varepsilon(p_{mir_1}) &= \lambda p_{imr_1} - e_{ijk} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} q_{mjr_2} t_{kr_3}, \\ \nabla_{q_{mjr_2}} \varepsilon(q_{mjr_2}) &= \lambda q_{mjr_2} - e_{ijk} \sum_{r_1=1}^{R_1} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{imr_1} t_{kr_3}, \\ \nabla_{t_{kr_3}} \varepsilon(t_{kr_3}) &= \lambda t_{kr_3} - e_{ijk} \sum_{m=1}^{M} \sum_{r_1=1}^{R_1} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} p_{imr_1} q_{mjr_2}, \\ \nabla_{a_i} \varepsilon(a_i) &= \lambda a_i - e_{ijk}, \end{aligned} \tag{9}$$

where e_{ijk} represents $y_{ijk} - \hat{y}_{ijk}$, the gradients of a_i, b_j, and c_k are similar, so only the gradient of a_i is shown in (9).

Hence, by combining (8) and (9), an MSGD-based parameter learning scheme is obtained for the MTN-TRL model. However, the hyper-parameters η, λ and γ require manual tuning via using a three-fold grid search, which would take a lot of time and diminish the usefulness of the model. In order to overcome these critical defects, we make these hyper-parameters self-adaptation during the training process via a Differential Evolution Algorithm (DEA) [25], [26], which has a strong global search capability and simple structure. To do

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

so, we first initialize Z vectors as a swarm, and each vector has the following definition:

$$\begin{aligned}\theta_{(z)} &= \left(\gamma_{(z)}, \eta_{(z)}, \lambda_{(z)}\right), \\ \gamma_{(z)} &= \gamma^{\min} + \delta \cdot \left(\gamma^{\max} - \gamma^{\min}\right), \\ \eta_{(z)} &= \eta^{\min} + \delta \cdot \left(\eta^{\max} - \eta^{\min}\right), \\ \lambda_{(z)} &= \lambda^{\min} + \delta \cdot \left(\lambda^{\max} - \lambda^{\min}\right),\end{aligned} \tag{10}$$

where δ is a random value in the range [0, 1], (η_{min}, η_{max}), $(\lambda_{min}, \lambda_{max})$ and $(\gamma_{min}, \gamma_{max})$ denote the range that η, λ and γ can obtain respectively. According to the DEA principle, the following mutation operation is performed on each vector individual:

$$\theta_{(z)}^{n+1} = \tau + \mu\left(\theta_{(z1)}^{n} - \theta_{(z2)}^{n}\right), \tag{11}$$

where θ is the scaling factor, τ is the global best individual, z_1 and z_2 are two randomly selected individuals. Then, each individual performs the following crossover operation:

$$\theta_{m(z)}^{n+1} = \begin{cases}\theta_{m(z)}^{n+1}, \text{if} \quad \sigma \le C_z \quad or \quad m = m^*; \\ \theta_{m(z)}^{n}, \text{else},\end{cases} \tag{12}$$

where m denotes the dimension index of the vector individual $\sigma_{(p)}$, m^* represents a randomly selected dimension, $C_z \in$[0, 1] indicates the crossover probability. Next, the fitness of each individual is determined by the model parameters learning process. Then, the fitness function for the $(n$+1)-th iteration is expressed as:

$$F_{(z)}^{n+1} = \frac{H\left(\theta_{(z)}^{n+1}\right) - H\left(\theta_{(z-1)}^{n+1}\right)}{H\left(\theta_{(Z)}^{n+1}\right) - H\left(\theta_{(Z)}^{n}\right)}, \tag{13}$$

where the function $H(^*)$ is calculated based on the model performance in the current iteration:

$$H\left(\theta_{(z)}\right) = \sqrt{\frac{\sum\limits_{y_{ijk}\in\Omega}\left(y_{ijk} - \hat{y}_{ijk(z)}\right)^2}{|\Omega|}} + \frac{\sum\limits_{y_{ijk}\in\Omega}\left|y_{ijk} - \hat{y}_{ijk(z)}\right|}{|\Omega|}, \tag{14}$$

where Ω represents the validation set, and $|\Omega|$ represents the number of samples in the validation set. If (14) is employed directly as the fitness function, it evaluates only the generalization error of individual particles, which naturally decreases over successive iterations. Consequently, the global optimum may increasingly favor the most recent particle, thereby diminishing the population's exploratory capability. In contrast, equation (13) quantifies each particle's marginal contribution to the reduction in overall generalization error, enhancing both global search effectiveness and convergence speed. By calculating the fitness function F, the global optimal individual vector is selected as follows:

$$\tau^{n+1} = \begin{cases}\theta_{(z)}^{n+1}, \text{if} \quad F_{(z)}^{n+1} > F_{(z-1)}^{n+1}, \\ \tau^{n}, \quad \text{if} \quad F_{(z)}^{n+1} \le F_{(z-1)}^{n+1}.\end{cases} \tag{15}$$

Through (10)-(15), we realize the hyper-parameters self-adaptation via using DEA, thereby obtaining an MSGD-based self-adaptation parameter learning scheme for MTN-TRL model, the detailed update rule is given as follows:

$$\begin{cases} g_{r_1r_2r_3}^{n,z} \leftarrow g_{r_1r_2r_3}^{n-1,z} - v_{g_{r_1r_2r_3}}^{n,z}, \\ v_{g_{r_1r_2r_3}}^{n,z} = \theta_1^{n,z} v_{g_{r_1r_2r_3}}^{n-1,z} + \theta_2^{n,z} \nabla_{g_{r_1r_2r_3}} \varepsilon\left(\theta_3^{n,z} g_{r_1r_2r_3}^{n-1}\right). \\ p_{imr_1}^{n,z} \leftarrow p_{imr_1}^{n-1,z} - v_{p_{imr_1}}^{n,z}, \\ v_{p_{imr_1}}^{n,z} = \theta_1^{n,z} v_{p_{imr_1}}^{n-1,z} + \theta_2^{n,z} \nabla_{p_{mir_1}} \varepsilon\left(\theta_3^{n,z} p_{mir_1}^{n-1}\right). \\ q_{mjr_2}^{n,z} \leftarrow q_{mjr_2}^{n-1,z} - v_{q_{mjr_2}}^{n,z}, \\ v_{q_{mjr_2}}^{n,z} = \theta_1^{n,z} v_{q_{mjr_2}}^{n-1,z} + \theta_2^{n,z} \nabla_{q_{mjr_2}} \varepsilon\left(\theta_3^{n,z} q_{mjr_2}^{n-1}\right). \\ t_{kr_3}^{n,z} \leftarrow t_{kr_3}^{n-1,z} - v_{t_{kr_3}}^{n,z}, \\ v_{t_{kr_3}}^{n,z} = \theta_1^{n,z} v_{t_{kr_3}}^{n-1,z} + \theta_2^{n,z} \nabla_{t_{kr_3}} \varepsilon\left(\theta_3^{n,z} t_{kr_3}^{n-1}\right). \\ a_i^{n,z} \leftarrow a_i^{n-1,z} - v_{a_i}^{n,z}, \\ v_{a_i}^{n,z} = \theta_1^{n,z} v_{a_i}^{n-1,z} + \theta_2^{n,z} \nabla_{a_i} \varepsilon\left(\theta_3^{n,z} a_i^{n-1}\right), \end{cases} \tag{16}$$

where $\theta_1^{n,z}$, $\theta_2^{n,z}$ and $\theta_3^{n,z}$ indicate the value of η, λ and γ in the z-th individual vector during the n-th iteration.

Algorithm 1. MTN-TRL

Input Λ, Ω, I, J, K, R_1, R_2, R_3, M

Operation	Cost
1: Initialize $\mathbf{G}$ with random values;	$\Theta(R_1 \times R_2 \times R_3)$
2: Initialize $\mathbf{P}$ with random values;	$\Theta(R_1 \times \|I\| \times M)$
3: Initialize $\mathbf{Q}$ with random values;	$\Theta(R_2 \times \|J\| \times M)$
4: Initialize T with random values;	$\Theta(R_3 \times \|K\|)$
5: Initialize $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ with random values;	$\Theta(\|I\| + \|J\| + \|K\|)$
6: Initialize vector v_{G}, v_{P}, v_{Q}, v_{T}, v_a, v_b, v_c=0;	
7: Initialize individual vector θ=0, swarm Z=10;	$\Theta(3) + \Theta(1)$
8: Initialize n=1, N=1000;	$\Theta(1)$
9: for each $y_{ijk} \in \Omega$ do	$\times\|\Omega\|$
10: Compute $\hat{y}_{ijk}$ based on (6);	$\Theta(1)$
11: end for	–
12: Compute $H\left(\theta_{(Z)}^{0}\right)$ based on (14);	$\Theta(1)$
13: while $n \le N$ and not converge do	$\times n$
14: for $z = 1$ to Z do	$\times Z$
15: for each $y_{ijk} \in \Lambda$ do	$\times\|\Lambda\|$
16: Compute $\hat{y}_{ijk}$ based on (6);	$\Theta(1)$
17: Update velocity vector v_{G}, v_{P}, v_{Q}, 18: v_{T}, v_a, v_b, v_c based on (16);	$\Theta(7)$
19: Update $\mathbf{G}$, $\mathbf{P}$, $\mathbf{Q}$, T based on (16);	$\Theta(4{\times}M{\times}R_1{\times}R_2{\times}R_3)$
20: Update a, b, c based on (16);	$\Theta(3)$
21: end for	–
22: end for	–
23: Compute $H\left(\theta_{(Z)}^{n}\right)$ based on (14);	$\Theta(1)$
24: for $z = 1$ to Z do	$\times Z$
25: Compute $F\left(\theta_{(Z)}^{n}\right)$ based on (13);	$\Theta(1)$
26: Update τ based on (15);	$\Theta(1)$
27: end for	–
28: $n = n + 1$;	$\Theta(1)$
29: end while	–

Output $\mathbf{G}, \mathbf{P}, \mathbf{Q}, \mathrm{T}, a, b, c$

D. Algorithm Design and Analysis

After the derivation in the previous section, Algorithm 1. MTN-TRL is design. Noting that the time cost of each step is listed, we find that MTN-TRL's computational complexity mainly relies on two parts:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

a) Initialization:

$$\begin{aligned} C_1 = &\Theta\left(R_1 \times R_2 \times R_3\right) + \Theta\left(R_1 \times |I| \times M\right) \\ &+\Theta\left(R_2 \times |J| \times M\right) + \Theta\left(R_3 \times |K|\right) \\ &+\Theta\left(|I| + |J| + |K|\right) \end{aligned} \tag{17}$$

b) Parameters update:

$$C_2 = \Theta\left(n \times Z \times |\Lambda| \times \left(\Theta\left(4 \times M \times R_1 \times R_2 \times R_3\right)\right)\right) \tag{18}$$

Thus, the computational complexity of MTN-TRL is:

$$\begin{aligned} C &= C_1 + C_2 \\ &\approx \Theta\left(n \times M \times R_1 \times R_2 \times R_3 \times |\Lambda|\right). \end{aligned} \tag{19}$$

where the low-order terms and constant coefficients are omitted. Since $|\Lambda|$ is much larger than $\max\{n, M, R_1, R_2, R_3\}$, the computational complexity of MTN-TRL is linear in the number of known elements in the nonstandard tensor.

Moreover, the storage complexity of MTN-TRL primarily consists of three parts:

a) The known elements set Λ and corresponding estimates:

$$S_1 = \Theta\left(|\Lambda|\right) \tag{20}$$

b) The LF matrices and tensors, bias vectors:

$$\begin{aligned} S_2 = &\Theta\left(M \times \left(|I| \times R_1 + |J| \times R_2\right)\right. \\ &\left.+|K| \times R_3 + R_1 \times R_2 \times R_3 + |I| + |J| + |K|\right). \end{aligned} \tag{21}$$

c) The velocity vectors:

$$\begin{aligned} S_3 = &\Theta\left(M \times \left(|I| \times R_1 + |J| \times R_2\right)\right. \\ &\left.+|K| \times R_3 + R_1 \times R_2 \times R_3 + |I| + |J| + |K|\right). \end{aligned} \tag{22}$$

Hence, the MTN-TRL's storage complexity is:

$$\begin{aligned} S &= S_1 + S_2 + S_3 \\ &\approx \Theta\left(|\Lambda| + M \times \left(|I| \times R_1 + |J| \times R_2\right) + |K| \times R_3\right), \end{aligned} \tag{23}$$

where (23) is obtained by reasonably dropping the lower-order and constants terms. From (23), it is evident that the storage complexity of MTN-TRL is linear with the size of the latent factors and the target nonstandard tensor's known element count.

E. Model Convergence Analysis

In the optimization of MTN-TRL, although the full objective (7) is nonconvex due to the interaction of tensors and matrices [27], our alternating optimization reduces each update to a convex least-squares problem with L_2 regularization. When all variables except one block (for example, **P**, **G**, **Q**, T or the bias vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$) are held fixed, the subproblem becomes a quadratic function whose Hessian is positive semidefinite, ensuring convexity. Then, based on the core update formula (16) of MSGD, we use MSGD to approximate the global optimal value of each subproblem and ensure that each iteration has a descending direction. This block-by-block alternation strategy ensures that MTN-TRL can converge to a stable point.This is also confirmed from the convergence iteration curves in Fig. S1. Next we will briefly analyze the convergence of the model, first reviewing the standard MSGD convergence assumptions [28]:

Assumption 1:

1) For any $x, y \in \mathbb{R}^N$,

$$g(x) \geq g(y) + \nabla g(y)(x - y). \tag{24}$$

2) There exists an θ^* that satisfies the following:

$$\nabla_{\theta^*} g(\theta^*) = 0. \tag{25}$$

3) The second derivative of $g(\theta)$ exists.

Lemma 1: If $\{X_n\} \in \mathbb{R}^N$ is martingale difference sequence, the following results can be obtained [29]:

$$\sum_{k=0}^{\infty} E\left(\|X_K\|^2\right) < +\infty \Rightarrow \sum_{k=0}^{\infty} X_K < +\infty. \tag{26}$$

Lemma 2: If $\{X_n\} \in \mathbb{R}^N$ is a sequence of random variables, according to the basic conclusions of probability theory, get

$$\sum_{t=1}^{\infty} E\left(\|X_K\|^2\right) < +\infty \Rightarrow \sum_{t=1}^{\infty} \|X_K\|^2 < +\infty. \tag{27}$$

Lemma 3: Suppose v_t is the sequence generated by (10). If 2) and 3) in Assumption 1 hold [29], then we get:

$$\sum_{t=1}^{n-1} E\left(\|v_{t-1}\|^2\right) < C < +\infty, \tag{28}$$

where C is a constant, and $\sum_{t-1}^{n-1} ||v_t - 1||^2 < +\infty$.

Lemma 4: Assume that $\{\omega_n\}$ is the estimated sequence generated by (10). If 2) and 3) in Assumption 1 hold [30], then

$$\begin{aligned} \sum_{t=1}^{n} \varepsilon_t E\left(\|\nabla \omega_t g(\omega_t)\|^2\right) < B < +\infty, \\ \sum_{t=1}^{n} \varepsilon_t \|\nabla \omega_t g(\omega_t)\|^2 < +\infty, \end{aligned} \tag{29}$$

where $B > 0$ is a constant.

Then, based on the above lemmas, the convergence theorem of MTN-TRL is given below.

Theorem 1: Assume that $\{\theta_t\}$ is the estimated sequence generated by (8). If 1)-3) of Assumption 1 hold, then for $\forall \theta_1 \in \mathbb{R}^N$ and $\forall v_0 \in \mathbb{R}^N$, we have:

$$\omega_t \to \omega^*. \tag{30}$$

Through the above assumptions and lemmas, we prove that MTN-TRL can almost converge to a stationary point. For the sake of brevity, the detailed proof of Theorem 1 is provided in the Supplementary File (SF)[1].

IV. Experiments

A. General Settings

Dataset. The experiments use eight dynamic network datasets, and their properties are summarized in Table II. In more detail, D1-D4 are derived from terminal devices in the real-world Internet of Things, and the traffic transmission between them is recorded in each time period [1]. D5-D8 originate from a cryptocurrency transaction website, which

[1]https://github.com/wangquuu/MTN-SF

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

TABLE II
DESCRIPTION OF DATASET PROPERTIES

Datasets	Nodes	Edges	Time slot	Density
D1	308,200	1,504,528	850	1.86E-08
D2	240,500	658,079	765	1.49E-08
D3	186,600	545,795	640	2.45E-08
D4	128,805	208,894	580	2.17E-08
D5	236,110	677,854	124	9.81E-08
D6	198,863	563,420	124	1.15E-07
D7	201,848	451,554	124	8.94E-08
D8	98,022	206,980	120	1.80E-07

TABLE III
DESCRIPTION OF ALL TESTED MODELS

No.	Competitor
M1	**MTN-TRL:** The model proposed in this paper.
M2	**TW:** A Tensor Wheel (TW) decomposition model trained with SGD algorithm and incorporating L_2 regularization [31].
M3	**Tucker:** A Tucker decomposition model trained with SGD algorithm and incorporating L_2 regularization [21].
M4	**TR:** A Tensor Ring decomposition model trained with SGD algorithm and incorporating L_2 regularization [32].
M5	**GSNTD:** A smooth nonnegative Tucker decomposition model, which uses L_p norm regularization [19].
M6	**SGCP:** A sparse graph-regularized CP decomposition model that adds L_1 and graph regularization to each factor matrix [14].
M7	**BNTucF:** A Tucker decomposition-based model, which designs a nonnegative update scheme [18].
M8	**BCTL:** A biased tensor latent factorization model that incorporates the idea of transfer learning into the training process [13].
M9	**TCA:** A tensor completion method based on CP, which adopts a completion mechanism based on gradient descent [33].
M10	**DNL:** A deep nonnegative latent factorization of tensor model that designs a scheme to jointly scale the depth of nonnegative multiplication learning [34].

records the transaction amount between each account at each point in time [4]. For each dataset, a "node×node×time" third-order nonstandard tensor can be constructed, where each element denotes an interaction weight between two nodes at a certain time slot, e.g., for D1, a nonstandard tensor of size $308,200 \times 308,200 \times 850$ is obtained, and its density is only 1.86×10^{-8}.

Evaluation Metrics. The representation learning ability of a TRL model to a nonstandard model can be intuitively reflected by prediction accuracy for dynamic network missing links (i.e., nonstandard tensor missing elements prediction accuracy), it is measured by widely used Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-squared (R^2):

$$MAE = \sum_{y_{ijk}\in\Phi} |y_{ijk} - \hat{y}_{ijk}|_{abs} \Big/ |\Phi|,$$

$$RMSE = \sqrt{\sum_{y_{ijk}\in\Phi} (y_{ijk} - \hat{y}_{ijk})^2 \Big/ |\Phi|},$$

$$R^2 = 1 - \sum_{y_{ijk}\in\Phi} (y_{ijk} - \hat{y}_{ijk})^2 \Big/ \sum_{y_{ijk}\in\Phi} (y_{ijk} - \tilde{y}_{ijk})^2,$$

where $\tilde{y}_{ijk}$ represents the mean of all y_{ijk} in the testing set Φ. Note that lower MAE and RMSE indicate higher prediction accuracy, whereas higher R^2 indicates better model performance.

Comparison Models. We compare the proposed MTN-TRL model with the nine state-of-the-art models, their briefly introduction is summarized in Table III.

Training Settings. The following training sets are uniformly adopted for all tested models to ensure fairness:

1) In order to evaluate the computational efficiency of all tested models, the total time of each model is recorded (including hyper-parameters adjustment time). Thus, all experiments are conducted on an i7-13700 CPU with 32G RAM, code in JAVA, and run under JDK1.8.
2) For obtaining objective results, for each dataset, 10% of the data is used as the training set Ψ to train the model, 20% of the data is used as the validation set Ω to monitor the training effect of the model, and 70% of the data is used as the test set Φ to verify the performance of the model. Note that the above processes are repeated for 20 times for obtaining 20 different sets of experimental results to eliminate the data biases.
3) The latent factor dimension R of all models is set to 5, e.g., M=R_1=R_2=R_3=5 for MTN-TRL. We strictly adjust the model's hyper-parameters on the validation set, and then verify the model's performance on the testing set after finding the optimal hyper-parameters. The hyper-parameters of the comparison models are searched according to the grid search to obtain their optimal performance and Table S1 records the optimal hyper-parameters for each model.
4) All involved models use the same early stopping strategy. Specifically, a model is considered converged if one of the following two conditions is met: a) the total number of iterations reaches 1000, and b) the iteration error of two consecutive rounds is less than 10^{-5}.

B. Comparison with State-of-the-art Models

To verify the performance of the proposed model, we compare MTN-TRL (M1) with nine TRL models. Table IV and Fig. S2 record the RMSE, MAE and R^2 of all test models on D1-8. Fig. S3 and Table S2 record their total time cost. Table S4 shows the results of Friedman Test, and Table S5 shows the results of Wilcoxon Signed-Ranks Test. Fig. S1 records the convergence iteration curves of MTN-TRL on D1-D8. The following findings can be drawn from these results:

1) **The representation learning ability of MTN-TRL performs best among all tested models**. As shown in Table IV and Fig. S2, MTN-TRL's RMSE, MAE and R^2 are all superior to its peers. For instance, the RMSE obtained by MTN-TRL on D1 is 0.2745, which is 1.12%, 5.90%, 3.68%, 6.78%, 7.91%, 5.06%, 6.56%, 5.32%, and 9.98% lower than that of M2-10 respectively. The MAE obtained by MTN-TRL on D1 is 0.1900, which is 0.21%, 2.26%, 5.42%, 0.42%, 0.79%, 7.05%, 5.68%, 6.47%, and 6.21% lower than that of M2-10 respectively. The R^2 obtained by MTN-TRL on D1 is 0.3717, which is 4.00%, 25.49%, 15.22%, 31.16%, 39.58%, 21.39%, 29.83%, 22.71%, and 55.07% higher than that of M2-10 respectively. In particular, the MTN-TRL outperforms Tucker-based TRL in all cases, which proves the effectiveness of the proposed MTN. On D2-8, as recorded in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

TABLE IV
THE RMSE, MAE AND R^2 OF ALL TESTED MODELS ON D1-8

Models	D1	D2	D3	D4	D5	D6	D7	D8	Win/Loss*
				RMSE↓					
M1	**0.2745$_{\pm 2.1E-04}$**	**0.2904$_{\pm 5.8E-05}$**	**0.3041$_{\pm 1.7E-04}$**	**0.2662$_{\pm 1.6E-04}$**	**0.5745$_{\pm 3.2E-05}$**	**0.6207$_{\pm 2.4E-03}$**	**0.6350$_{\pm 7.9E-04}$**	**0.6806$_{\pm 6.4E-03}$**	–
M2	0.2776$_{\pm 1.2E-04}$	0.2946$_{\pm 2.1E-04}$	0.3086$_{\pm 1.0E-04}$	0.2701$_{\pm 3.5E-04}$	0.5935$_{\pm 1.0E-03}$	0.6357$_{\pm 3.0E-03}$	0.6459$_{\pm 1.9E-03}$	0.6918$_{\pm 9.1E-04}$	8/0
M3	0.2907$_{\pm 5.8E-05}$	0.3069$_{\pm 1.2E-04}$	0.3164$_{\pm 1.7E-04}$	0.2790$_{\pm 5.8E-05}$	0.6228$_{\pm 4.2E-04}$	0.6667$_{\pm 1.5E-04}$	0.6801$_{\pm 9.5E-04}$	0.7419$_{\pm 3.2E-04}$	8/0
M4	0.2846$_{\pm 1.7E-04}$	0.2970$_{\pm 2.5E-04}$	0.3146$_{\pm 1.5E-04}$	0.2716$_{\pm 3.1E-04}$	0.5925$_{\pm 3.1E-04}$	0.6321$_{\pm 1.5E-04}$	0.6392$_{\pm 3.0E-04}$	0.6860$_{\pm 6.0E-04}$	8/0
M5	0.2931$_{\pm 2.6E-05}$	0.3110$_{\pm 1.0E-04}$	0.3241$_{\pm 2.5E-04}$	0.2859$_{\pm 2.7E-04}$	0.6537$_{\pm 2.4E-04}$	0.6874$_{\pm 4.2E-04}$	0.6900$_{\pm 7.0E-04}$	0.7544$_{\pm 4.0E-04}$	8/0
M6	0.2962$_{\pm 3.5E-04}$	0.3181$_{\pm 1.0E-03}$	0.3302$_{\pm 9.6E-04}$	0.2876$_{\pm 7.8E-04}$	0.6810$_{\pm 7.1E-04}$	0.7070$_{\pm 4.4E-03}$	0.7093$_{\pm 1.3E-02}$	0.7606$_{\pm 2.0E-03}$	8/0
M7	0.2884$_{\pm 1.2E-04}$	0.3034$_{\pm 1.0E-04}$	0.3161$_{\pm 2.0E-04}$	0.2760$_{\pm 5.8E-05}$	0.6215$_{\pm 3.6E-04}$	0.6678$_{\pm 6.7E-04}$	0.6852$_{\pm 5.4E-03}$	0.7431$_{\pm 5.5E-04}$	8/0
M8	0.2925$_{\pm 4.6E-04}$	0.3085$_{\pm 1.3E-03}$	0.3217$_{\pm 5.9E-04}$	0.2851$_{\pm 1.2E-03}$	0.6134$_{\pm 2.1E-04}$	0.6651$_{\pm 3.1E-04}$	0.6737$_{\pm 5.7E-04}$	0.7398$_{\pm 2.2E-03}$	8/0
M9	0.2891$_{\pm 1.5E-04}$	0.3071$_{\pm 5.5E-04}$	0.3210$_{\pm 5.8E-04}$	0.2834$_{\pm 5.9E-04}$	0.6566$_{\pm 2.9E-04}$	0.6980$_{\pm 3.2E-04}$	0.6949$_{\pm 1.2E-03}$	0.7726$_{\pm 5.2E-04}$	8/0
M10	0.3019$_{\pm 2.5E-04}$	0.3179$_{\pm 3.5E-04}$	0.3350$_{\pm 1.2E-03}$	0.2926$_{\pm 7.8E-04}$	0.6820$_{\pm 1.3E-03}$	0.7118$_{\pm 4.0E-04}$	0.7175$_{\pm 1.8E-03}$	0.7916$_{\pm 4.6E-03}$	8/0
				MAE↓					
M1	**0.1900$_{\pm 2.3E-04}$**	**0.1957$_{\pm 1.5E-04}$**	**0.2077$_{\pm 1.2E-04}$**	**0.1905$_{\pm 2.1E-04}$**	**0.3929$_{\pm 4.5E-04}$**	**0.4194$_{\pm 1.3E-03}$**	**0.4605$_{\pm 1.2E-03}$**	**0.5124$_{\pm 3.0E-03}$**	–
M2	0.1904$_{\pm 1.5E-04}$	0.1971$_{\pm 5.8E-05}$	0.2101$_{\pm 1.7E-04}$	0.1922$_{\pm 2.5E-04}$	0.4029$_{\pm 3.4E-03}$	0.4215$_{\pm 4.8E-03}$	0.4801$_{\pm 4.8E-03}$	0.5169$_{\pm 7.5E-04}$	8/0
M3	0.1943$_{\pm 1.5E-04}$	0.2050$_{\pm 1.2E-04}$	0.2140$_{\pm 1.0E-04}$	0.1993$_{\pm 1.7E-04}$	0.4104$_{\pm 1.5E-04}$	0.4363$_{\pm 3.1E-04}$	0.4786$_{\pm 4.6E-04}$	0.5429$_{\pm 1.5E-04}$	8/0
M4	0.2003$_{\pm 1.0E-04}$	0.2014$_{\pm 1.5E-04}$	0.2200$_{\pm 1.2E-04}$	0.1943$_{\pm 2.3E-04}$	0.4036$_{\pm 3.8E-04}$	0.4247$_{\pm 2.5E-04}$	0.4624$_{\pm 2.3E-03}$	0.5131$_{\pm 6.1E-04}$	8/0
M5	0.1908$_{\pm 5.6E-05}$	0.2007$_{\pm 5.8E-05}$	0.2116$_{\pm 1.1E-04}$	0.1994$_{\pm 2.5E-04}$	0.4213$_{\pm 2.9E-04}$	0.4482$_{\pm 1.5E-04}$	0.4782$_{\pm 4.0E-04}$	0.5471$_{\pm 2.9E-04}$	8/0
M6	0.1915$_{\pm 2.5E-04}$	0.2055$_{\pm 1.5E-04}$	0.2142$_{\pm 6.0E-04}$	0.1999$_{\pm 2.8E-04}$	0.4356$_{\pm 1.1E-03}$	0.4526$_{\pm 3.7E-03}$	0.4834$_{\pm 6.8E-03}$	0.5507$_{\pm 1.5E-03}$	8/0
M7	0.2034$_{\pm 5.8E-05}$	0.2051$_{\pm 1.0E-04}$	0.2193$_{\pm 5.8E-05}$	0.1991$_{\pm 5.8E-05}$	0.4323$_{\pm 2.1E-04}$	0.4550$_{\pm 2.3E-04}$	0.5003$_{\pm 3.0E-03}$	0.5546$_{\pm 8.3E-04}$	8/0
M8	0.2008$_{\pm 1.5E-04}$	0.2070$_{\pm 5.8E-05}$	0.2207$_{\pm 1.5E-04}$	0.2049$_{\pm 1.2E-04}$	0.4188$_{\pm 1.0E-04}$	0.4473$_{\pm 2.3E-04}$	0.4863$_{\pm 2.5E-04}$	0.5549$_{\pm 2.0E-03}$	8/0
M9	0.2023$_{\pm 5.0E-04}$	0.2083$_{\pm 3.2E-04}$	0.2173$_{\pm 4.6E-04}$	0.2026$_{\pm 3.6E-04}$	0.4767$_{\pm 1.0E-04}$	0.4975$_{\pm 4.0E-04}$	0.5018$_{\pm 1.1E-03}$	0.5621$_{\pm 4.6E-04}$	8/0
M10	0.2018$_{\pm 5.8E-05}$	0.2124$_{\pm 3.2E-04}$	0.2217$_{\pm 7.2E-04}$	0.2084$_{\pm 6.4E-04}$	0.4450$_{\pm 5.2E-04}$	0.4675$_{\pm 4.7E-04}$	0.4987$_{\pm 9.2E-04}$	0.5728$_{\pm 1.8E-03}$	8/0
				R^2↑					
M1	**0.3717$_{\pm 9.6E-04}$**	**0.2764$_{\pm 3.6E-04}$**	**0.2660$_{\pm 8.4E-04}$**	**0.2913$_{\pm 8.3E-04}$**	**0.4323$_{\pm 9.5E-05}$**	**0.3837$_{\pm 4.7E-03}$**	**0.3305$_{\pm 1.8E-03}$**	**0.3352$_{\pm 1.3E-02}$**	–
M2	0.3574$_{\pm 6.1E-04}$	0.2551$_{\pm 9.9E-04}$	0.2451$_{\pm 4.2E-04}$	0.2702$_{\pm 1.5E-03}$	0.3938$_{\pm 1.7E-03}$	0.3537$_{\pm 5.9E-03}$	0.3072$_{\pm 4.0E-03}$	0.3128$_{\pm 1.9E-03}$	8/0
M3	0.2962$_{\pm 5.1E-04}$	0.1930$_{\pm 7.8E-04}$	0.2055$_{\pm 9.2E-04}$	0.2219$_{\pm 2.9E-04}$	0.3328$_{\pm 9.7E-04}$	0.2889$_{\pm 5.0E-04}$	0.2322$_{\pm 2.1E-03}$	0.2098$_{\pm 6.8E-04}$	8/0
M4	0.3226$_{\pm 7.9E-04}$	0.2417$_{\pm 1.4E-03}$	0.2133$_{\pm 7.5E-04}$	0.2597$_{\pm 1.6E-03}$	0.3916$_{\pm 7.4E-04}$	0.3493$_{\pm 3.6E-04}$	0.3135$_{\pm 6.0E-04}$	0.3183$_{\pm 1.2E-03}$	8/0
M5	0.2834$_{\pm 1.5E-04}$	0.1700$_{\pm 6.0E-04}$	0.1662$_{\pm 1.3E-03}$	0.1824$_{\pm 1.6E-03}$	0.2649$_{\pm 5.4E-04}$	0.2443$_{\pm 9.3E-04}$	0.2096$_{\pm 1.7E-03}$	0.1829$_{\pm 8.6E-04}$	8/0
M6	0.2663$_{\pm 1.8E-03}$	0.1283$_{\pm 5.8E-03}$	0.1343$_{\pm 6.6E-03}$	0.1701$_{\pm 4.7E-03}$	0.1959$_{\pm 2.8E-03}$	0.1851$_{\pm 1.0E-02}$	0.1545$_{\pm 3.0E-02}$	0.1604$_{\pm 6.5E-03}$	8/0
M7	0.3062$_{\pm 3.8E-04}$	0.2100$_{\pm 8.1E-04}$	0.2069$_{\pm 7.0E-04}$	0.2379$_{\pm 2.1E-04}$	0.3356$_{\pm 8.3E-04}$	0.2867$_{\pm 1.5E-03}$	0.2205$_{\pm 1.2E-02}$	0.2072$_{\pm 1.2E-03}$	8/0
M8	0.2863$_{\pm 2.1E-03}$	0.1830$_{\pm 6.8E-03}$	0.1785$_{\pm 3.0E-03}$	0.1870$_{\pm 6.9E-03}$	0.3528$_{\pm 5.5E-04}$	0.2925$_{\pm 6.2E-04}$	0.2464$_{\pm 1.3E-03}$	0.2143$_{\pm 4.8E-03}$	8/0
M9	0.3029$_{\pm 7.0E-04}$	0.1904$_{\pm 3.0E-03}$	0.1823$_{\pm 2.9E-03}$	0.1966$_{\pm 3.2E-03}$	0.2584$_{\pm 6.7E-04}$	0.2207$_{\pm 7.0E-04}$	0.1984$_{\pm 2.9E-03}$	0.1430$_{\pm 1.2E-03}$	8/0
M10	0.2397$_{\pm 1.3E-03}$	0.1327$_{\pm 1.9E-03}$	0.1094$_{\pm 6.6E-03}$	0.1437$_{\pm 4.4E-03}$	0.1999$_{\pm 3.1E-03}$	0.1895$_{\pm 8.5E-04}$	0.1453$_{\pm 4.3E-03}$	0.1003$_{\pm 1.0E-02}$	8/0

*Win/Loss represents the number of MTN-TRL compared to other models to win or lose cases on D1-8.

Table IV, similar results are also obtained. The above experimental results are consistent with expectations. This superior performance is mainly attributed to the design of MTN, which improves the traditional Tucker network by introducing a tensor contraction operation. As a result, the latent feature space is expanded to represent higher dimensions, which can more accurately represent the mode imbalance in nonstandard tensors.

2) **The MTN-TRL's total time cost is very competitive compared with its peers**. For a tensor representation learning model without hyper-parameters adaptation, the tuning time before it reaches optimal performance needs to be considered. Generally speaking, an effective solution for manually tuning hyper-parameters is to perform a grid search. Considering M2-10, without hyper-parameters self-adaptation, they normally have two or three main hyper-parameters, their optimal values are usually in range of $[2^{-0}, 2^{-12}]$, so it is necessary to search a 13×13 hyper-parameters grid on average. Table S1 record the optimal hyper-parameters values of M2-10 obtained by grid search. Table S2 records the total time cost in obtaining the optimal RMSE, MAE and R^2 of M1-M10. From it, we can obviously see that with hyper-parameters self-adaptation using DEA, the total time cost of M1 is highly competitive. For example, on D1, the MTN-TRL's total time cost in RMSE is 5.8min, which is 1.88% of M2's 308.1min, 1.89% of M3's 306.7min, 11.49% of M4's 50.5min, 14.39% of M5's 40.3min, 8.61% of M7's 67.4min, 4.41% of M9's 131.5min, 2.61% of M10's 222.1min, but is only high than that of M6 and M8. Similar situations can be observed on D2-8. In general, MTN-TRL has such an advantage in time cost due to the following two aspects: a) the adopted MSGD algorithm reduces the number of convergence iterations to a certain extent, thereby accelerating convergence; b) the designed self-adaptation parameters learning scheme based on the DE algorithm makes it unnecessary to include parameter adjustment time during training.

Remark. Note that MTN-TRL already has a lower training time cost than other baselines, but when deployed on an industrial scale, it is usually pursued to have a lower training time cost. To address this problem, we designed a parallel computing method for MTN-TRL, as shown in Fig. S4. This method divides the target tensor into multiple sub-blocks according to the time dimension, and divides the time factor matrix T in the same way for multi-threaded parallel training. We performed 10-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

thread parallel training on MTN-TRL on D5, and the experimental results are shown in Table S3. We find that it needs to take 33 minutes to converge on the original R^2, but now it only takes 15.2 minutes to converge, and it hardly affects the representation accuracy.

3) **The MTN-TRL's performance is statistically significantly improved**. Statistical tests help to reflect the experimental results more intuitively. Thus, we perform the commonly used Friedman Test and Wilcoxon Signed-Ranks Test on Table S4 and S5 [1]. First, the Friedman test is a nonparametric test used to compare the overall differences of three or more related samples under different treatment conditions. Note that in terms of accuracy, RMSE, MAE, and R^2 of each model on D1-8 have a total of 24 cases, and efficiency also has 24 cases. As shown in Table S4, the Friedman Test is performed on the accuracy and efficiency of all tested models and two sets of F-rank are obtained, where the smaller the F-rank, the better the performance. Specifically, MTN-TRL achieved an F-rank of 1.00 and 2.25 in accuracy and efficiency, respectively, both ahead of other models. We also find that since MTN-TRL uses the proposed self-adaptation parameter learning scheme, the F-rank of 2.25 obtained in efficiency is leading compared to other models. This is consistent with our experimental results.
   In addition, the Wilcoxon signed-rank test is a nonparametric paired test used to assess the significance of the difference between the medians of the same sample under two treatment conditions. Table S5 records the results of Wilcoxon Signed-Ranks Test, which mainly include three indicators: *R*+, *R*-, *p*-value. Among them, *R*+ represents the ranking of winning cases, *R*- is the opposite, and *p*-value represents the significance level, if it is less than 0.1, it is judged that the performance of M1 has been improved compared to the comparison model. In all comparisons on accuracy, M1 obtain a *p*-value less than 0.1. And in terms of efficiency, M1 is on par with M6, but far exceeds other models. The results achieved are statistically significant.

C. Scalability of the MTN-TRL

In order to verify the scalability of the MTN-TRL model, We conduct two extension experiments with different training set ratios and two new nonstandard tensors. First, the proportion of the training set in one dataset is gradually increased from 10% to 80% in 10% increments. Then, two new nonstandard tensors (D9 and D10) with higher-dimension and more extreme mode imbalance are added in the empirical study, where the dimension of D9 is (352,678×352,678×50) and its density is 1.57×10^{-8}, and the dimension of D10 comes to (409,025×409,025×30) and its density is 2.21×10^{-8}. Figs. S5 and S6 show the representation accuracy, based on two experimental results, we obtain the following findings:

1) **The MTN-TRL model has better performance than its peers in different training set ratios.** To validate the performance of the proposed MTN-TRL model on training sets of different proportions, we gradually increase the training set ratio from 10% to 80% on D4. As shown in Fig. S5, under different training set ratios, M1 consistently achieves the highest representation accuracy. For example, when the training set ratio reaches 20% (i.e., the ratio of training set-validation set-testing set is 20%-10%-70%), the RMSE values for M1-10 are 0.3330, 0.3375, 0.3456, 0.3384, 0.3616, 0.3536, 0.3545, 0.3552, 0.3557, 0.3668, and the MAE values are 0.3256, 0.3318, 0.3423, 0.3261, 0.3556, 0.3553, 0.3441, 0.3489, 0.3502, 0.3657, with M1 exhibiting the lowest RMSE and MAE. Furthermore, as the training set ratio increase, the representation accuracy of M1 also steadily improve. For instance, the R^2 of M1 consistently rises from 0.2913 to 0.4272. This continued advantage highlights the advantages of MTN-TRL in extracting salient features and modeling temporal relationships.
2) **MTN-TRL still leads in performance on a nonstandard with higher-dimension and more extreme mode imbalance.** The experimental results in Fig. S6 show that MTN-TRL has lower RMSE and MAE than other tested models. Specifically, on D9, MTN-TRL achieves an RMSE of 0.3330, which is 1.33%, 3.65%, 1.60%, 7.91%, 5.83%, 6.06%, 6.25%, 6.38%, and 9.21% lower than those of M2-M9, respectively. In addition, the MAE achieves by MTN-TRL is 0.2247, which is 1.71%, 4.10%, 2.01%, 8.02%, 2.09%, 4.75%, 5.59%, 9.40%, and 7.03% lower than those of M2-9, respectively. Considering R^2, M1 still achieves the highest R^2-value compared with the benchmark models. Similar findings can be obtained on D10. The above results prove that MTN demonstrates excellent scalability when representing nonstandard tensors.

D. Hyper-parameters Sensitivity Analysis

According to Section III, the proposed MTN-TRL's hyper-parameters, i.e., γ, η and λ, affect its performance. Thus, we perform a hyper-parameter sensitivity analysis on D1-D8 via adopting manual grid search. First, the γ is set to 0.8 according to the empirical value, and then a double grid search is performed on η and λ with a step size of 2^{-1} and a range of $[2^{-0}, 2^{-12}]$. Then, under the optimal η and λ, γ is tuned with a step size of 0.1 and a range of $[0.1, 0.9]$. According to the experimental results depicted in Figs. S7-S9, the following conclusions can be obtained.

1) **The representation accuracy of MTN-TRL exhibits sensitivity to η , λ, and γ.** As shown in Fig. S7(a), on D1, when γ is fixed, the highest RMSE obtained by MTN-TRL is 0.3207 with η=2^{-5} and λ=2^{-8}, and its lowest RMSE stands at 0.2749 with η=2^{-11} and λ=2^{-7}, the gap between the highest and the lowest RMSE reaches 14.28%. As shown in Fig. S7(b), the difference between the highest MAE at 0.2284 with η=2^{-5} and λ=2^{-8}, and the lowest MAE at 0.1905 with η=2^{-11} and λ=2^{-5} comes 16.59%. Considering R^2, as shown

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

in Fig. S7(c), its highest and lowest values are 0.3727 with η=2^{-12} and λ=2^{-6} and 0.1459 with η=2^{-5} and λ=2^{-7} respectively, resulting in an 60.85% difference. Moreover, on D8, as shown in Fig. S9(v), when η and λ are fixed, MTN-TRL achieves a minimum RMSE of 0.6748 at γ=0.7 and a maximum RMSE of 0.6949 at γ=0.1. Similarly, as depicted in Fig. S9(w), when γ=0.1 and γ=0.7, MTN-TRL achieves the highest and lowest MAE values of 0.5396 and 0.5125, respectively. As depicted in Fig. S9(x), the highest R^2 is 0.3341 with γ=0.7 the lowest R^2 is 0.3025 with γ=0.1. A similar situation occurs in other datasets, as shown in Figs. S7-S9.

2) **The convergence iteration count of the MTN-TRL model is primarily influenced by η and γ.** As shown in Fig. S7(d), on D1, when λ=2^{-7}, as η increases from 2^{-12} to 2^0, the number of iterations required to converge to the minimum RMSE decreases from 112 to 3. Similarly, as shown in Fig. S7(e) when λ=2^{-5}, as η increases from 2^{-12} to 2^0, the number of iterations required to converge to the minimum MAE decreases from 136 to 2. As shown in Fig. S7(f), when λ=2^{-6}, as η increases from 2^{-12} to 2^0, convergence iterations drop from 133 to 2 at the highest R^2. When η and λ are fixed, with γ increasing from 0.1 to 0.9, MTN-TRL's convergence iteration count shows a continuous decline. As depicted in Fig. S9(v), on D8, when γ=0.1 and 0.9, MTN-TRL takes 811 and 275 iterations respectively to converge to the lowest RMSE, and similar trends can be observed for both MAE and R^2. Similar phenomena are also encountered on other datasets, as in Figs. S7-S9.

Overall, the large swings in representation accuracy and convergence iteration counts reveal that manually tuning hyper-parameters is both inefficient and poorly adaptable across datasets. By incorporating an adaptive hyperparameter adjustment strategy, we can address these issues directly, boosting the model's robustness and real-world applicability.

E. Impact of Adaptive Hyper-parameters

To explore the impact of hyper-parameters adaptation schemes on the MTN-TRL, we conduct a series of experiments on D1-8, using both manual and adaptive hyper-parameters tuning. Table S6 records in detail the RMSE, MAE, R^2, and total time of MTN-TRL under these two schemes. The following conclusions can be drawn:

1) **The hyper-parameters self-adaptation scheme is beneficial for MTN-TRL's representation accuracy gain**. As shown in Table S6, there are a total of 24 cases including RMSE, MAE, and R^2 on D1-8. MTN-TRL's representation accuracy after hyper-parameters adaptation wins 17 cases compared with manual-tuning, among which the largest accuracy gains are improved by 0.38%, 0.41% and 3.00% in RMSE, MAE, and R^2 on D2, respectively. Since the hyper-parameters self-adaptation scheme optimally selects the current hyper-parameters based on previous iteration errors during the learning process, compared with manual-tuning scheme, the hyper-parameters adaptation scheme can dynamically adjust the learning process and find the optimal solution more easily. Therefore, the accuracy gain obtained by MTN-TRL is reasonable.
2) **The hyper-parameters self-adaptation scheme can significantly reduce the total time cost of MTN-TRL**. For instance, as shown in Table S6, the MTN-TRL's total time cost in RMSE, MAE, and R^2 with adaptive hyper-parameters on D1 is 5.8min, 8.1 min, 5.8min, which is 96.50%, 96.29%, 96.55% less than the 165.8min, 218.3min, 168.0min with manual-tuning. Similar results can be observed on other datasets as well, and the time cost savings are significant. Accordingly, this efficiency improvement is reasonable. As mentioned above, MTN-TRL's hyper-parameters γ, η and λ need to be optimal through three-fold grid search, which requires 13×13×10 training cycles. On the contrary, with hyper-parameters adaptation, MTN-TRL can automatically find optimal hyper-parameters in a single training process, thereby significantly reducing the total time cost.

Through the above experiments and analysis, we found that the adaptive hyper-parameters learning scheme based on DEA can not only save a lot of time, but also obtain certain accuracy benefits.

F. Model Performance Analysis with Different Parameters Learning Schemes

In this section, we mainly evaluate the performance of the MTN-TRL model under different parameter learning schemes. Considering the differential evolution strategy, we use formula (14) as the fitness function to construct the model MTN-DE1, and replace the mutation formula (11) of differential evolution with $\theta_{(z)}^{n+1} = \theta_{(z1)}^{n} + \mu\left(\theta_{(z2)}^{n} - \theta_{(z3)}^{n}\right)$ to obtain MTN-DE2. In addition, two commonly used optimization algorithms (SGD and Adam) are used to learn the representations, and two variants of the MTN-TRL model are constructed based on these two optimization algorithms, namely MTN-SGD and MTN-Adam. The above models are then compared with the MTN-TRL model. Figs. S10 and S11 show the representation accuracy and total training time of the five test models, respectively. From these results, we can observe that:

1) **The adaptive parameter learning scheme based on the newly designed differential evolution strategy in this study demonstrates better representation accuracy and computational efficiency compared with other differential evolution strategies.** As shown in Figs. S10 and S11, there are a total of 96 cases on D1-8, including RMSE, MAE, R^2 and their corresponding time costs. It can be clearly seen that MTN-TRL has lower RMSE/MAE and highest R^2 than MTN-DE1 and MTN-DE2. For example, as shown in Figs. S10(a) and S11(a), the RMSE of MTN-TRL in D2 decreases by 1.06% and 0.75% compared with that of MTN-DE1 and MTN-DE2, respectively, and the corresponding time cost decreases by 45.28% and 62.82%. This is because the fitness function of MTN-DE1 will result in the last

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

particle in the iteration process being selected each time, while the fitness function (13) we designed considers the particle with the greatest contribution in the evolution process. In addition, the mutation strategy used by MTN-DE2 considers random particles for mutation, while the mutation strategy (11) we use considers the global optimal particle for mutation. For the above reasons, MTN-TRL naturally has higher representation accuracy and computational efficiency.

2) **Compared with MTN-SGD and MTN-Adam, MTN-TRL exhibits higher computational efficiency and highly competitive representation accuracy.** As shown in Fig. S11(a), on D1, when converging to the lowest RMSE, MTN-TRL takes only 5.8 minutes, while MTN-SGD and MTN-Adam require 33.7 minutes and 194.4 minutes, which are 5.8 times and 33.5 times the time taken by MTN-TRL, respectively. Considering the MAE, MTN-TRL takes 8.1 minutes to converge to the lowest MAE, while MTN-SGD and MTN-Adam take 12.7 times and 20.8 times the time of MTN-TRL, respectively. When the metric is R^2, MTN-TRL converges to the optimal R^2 in only about 1/6 of the time taken by MTN-SGD and about 1/30 of the time taken by MTN-Adam. Similar results can be obtained on D2-8. In term of representation accuracy, as shown in Fig. S10, on D1-8, based on the RMSE, MAE, and R^2 metrics, MTN-SGD and MTN-Adam generate a total of 48 cases to measure representation accuracy, with M1 winning in 45 cases. For example, as shown in Fig. S10(a), on D2, MTN-TRL's RMSE is 0.2765, which is 0.50% and 1.29% lower than those of MTN-SGD and MTN-Adam, respectively. These results indicate that the MSGD-based adaptive parameter learning scheme proposed in this study is essential and necessary for efficiently and accurately learning the representation of nonstandard tensors.

In summary, the proposed adaptive parameter learning scheme based on MSGD and DE algorithm is crucial for building an efficient and accurate MTN-TRL model. It enables MTN-TRL to achieve good representation accuracy while significantly reducing the time cost.

G. Impact of Dimension M

In MTN, two LF matrices $\mathrm{P} \in \mathbb{R}^{|I| \times R_1}$ and $\mathrm{Q} \in \mathbb{R}^{|J| \times R_2}$ are extended into two third-order LF tensors $\mathbf{P} \in \mathbb{R}^{M \times |I| \times R_1}$ and $\mathbf{Q} \in \mathbb{R}^{M \times |J| \times R_2}$ for representing two high-dimensional mode of the nonstandard tensor, as a result, the dimension M of LF tensors' newly constructed mode has a directed effect on the representation learning ability of MTN-TRL. Therefore, in this part, we explore the impact of the dimension M to MTN-TRL, and verify the effectiveness of MTN via comparing with a traditional Tucker network (i.e., M=1). As shown in Fig. S12, we gradually increase the value of dimension M from the range $[1, 10]$ on D1-D8, and record the RMSE, MAE, and R^2 of MTN-TRL. Based on these results, the following findings can be drawn:

1) **The representation learning ability of MTN-TRL can be significantly enhanced by the newly constructed mode of LF tensors**. As mentioned above, when M=1, MTN degenerates into a Tucker network. As shown in Fig. S12, on D1-D8, when M=1, the maximum RMSE and MAE are obtained, and the minimum R^2 is obtain. Concretely, when M=1, the RMSE of MTN-TRL are 0.2993, 0.3229, 0.331, 0.2949, 0.6247, 0.672, 0.692, and 0.7567 on D1-8, respectively. When M=2, i.e., LF matrices are extended into third-order LF tensors (a new mode is constructed), the RMSE of MTN-TRL respectively are 0.2877, 0.3082, 0.3168, 0.2798, 0.6052, 0.6497, 0.6617, and 0.7153 on D1-8, which are reduced by 3.88%, 4.55%, 4.29%, 5.12%, 3.12%, 3.32%, 4.38%, and 5.47%, respectively. Considering MAE and R^2, similar results are also obtained on D1-8.
2) **The dimension M of LF tensor affects the performance of MTN-TRL**. As shown in Fig. S12, as the value of dimension M increases, the RMSE and MAE show a downward trend, while R^2 shows the opposite trend. Specifically, the MTN-TRL obtains the minimum RMSE of 0.2757 on D1 when M=10, which is 4.17% lower than 0.2877 when M=2, at the same time, MAE decreases by 2.16% and R^2 increases by 18.06%. Moreover, the MTN-TRL's MAE on D3 and RMSE on D6 have a trend of first decreasing and then rising, e.g., the minimum MAE is 0.2089 when M=5 on D3, that is to say, the optimal M for LF tensor has been obtained with MAE as metric. In general, when the value of M is larger, the performance of MTN-TRL will gradually improve, but there is an optimal value of M, which is normally data-dependent.

H. Ablation Study

In order to verify whether a high-dimensional mode of a nonstandard tensor need to adopt an LF tensor representation, in other words, verifying the rationality of constructing a new edge between latent factor nodes representing two high-dimensional modes in MTN. As shown in Fig. S13, two new MTN is designed, i.e., an MTN-2 by expanding matrices Q and T (i.e., constructing a new edge between nodes Q and T), and an MTN-3 by expanding matrices P and T (i.e., constructing a new edge between nodes P and T). Fig. S14 records the RMSE, MAE, and R^2 of TRL adopting three different MTN on D1-8. As illustrated in Fig. S14, the TRL with MTN demonstrates better representation learning ability than TRL with MTN-2 and MTN-3 on D1-8. For example, the RMSE obtained by MTN-TRL on D1 is 0.2745, which is 5.67% lower than 0.2910 of MTN-2 and 4.05% lower than 0.2861 of MTN-3. Likewise, similar findings can be obtained in other cases. Through experimental verification, we conclude that a larger representation space can better represent high-dimensional mode of a nonstandard tensor.

I. The Generalization of MTN-TRL on $4th$-order Nonstandard Tensors

To assess the representation capability of MTN to higher-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

order nonstandard tensors, two fourth-order nonstandard tensors (Tensor1 and Tensor2) are adopted in this experiment. Tensor1 is a "node×node×time×feature" fourth-order nonstandard tensor which models a telecommunication network, and each element describes a interaction feature value (e.g., byte count, packet count, latency, or packet loss rate between two nodes). It has the dimension of (1392×1392×100×4) and its data density is 0.04%. Tensor2 models a road traffic network, where each element represents the average speed of one road monitored by a sensor. It has the dimension of (214×144×7×8) and its data density is 10%, where 214 denotes the number of speed sensors, 144 and 7 represents the time slots of one day and the number of consecutive monitoring days, and 8 indicates monitoring cycles. For such a fourth-order non-standard tensor, according to the design idea of MTN, we construct a new edge between two LF nodes with higher-dimensional modes. Three common tensor networks, i.e., Tucker, Tensor Ring (TR), and Tensor Wheel (TW), are used as benchmarks, and the experimental results are shown in Table S7. From the results, we can find that MTN has comprehensive advantages over the other three common tensor networks in the fourth-order nonstandard tensor representation. As recorded on Table S7, on Tensor1, the RMSE of MTN is 0.8389, which is 7.94% lower than Tucker's 0.9113, 7.84% lower than TR's 0.9103, and 3.71% lower than TW's 0.8712. Similarly, on Tensor2, the RMSE of MTN is 0.0850, which is 1.62% lower than Tucker's 0.0864, 2.86% lower than TR's 0.0875, and 1.16% lower than TW's 0.0860. It also shows similar trends in MAE and R^2. These results indicate that MTN not only represents third-order nonstandard tensors effectively but also exhibits excellent representation capabilities for higher-order nonstandard tensors, demonstrating its strong generalization ability owing to its mode-aware design philosophy.

J. Engineering Application Scenes

In this section we apply MTN-TRL to a real-world engineering application, i.e., traffic congestion prediction. Specifically, traffic congestion prediction can be achieved through road network link prediction, where each road is represented as a weighted link based on vehicle speed data collected by sensors. By predicting the future weights of these links, the level of congestion can be effectively assessed. However, in the actual process of data collection, some factors (i.e., network interruption or sensor anomaly) may lead to the loss of data acquisition, making it difficult to accurately predict future speed data when directly using predictors like LSTM. As shown in Fig. S15, a road network can be modeled as a third-order "road×interval×day" nonstandard tensor, where an element of the tensor represents the average speed on one road during a specific time interval within one day. Hence, we first employ the MTN-TRL model to learn representations of the non-standard speed tensor, i.e., latent factors of road mode, interval mode and day mode, then impute missing historical speed data based on the low-dimensional representations. Subsequently, the completed tensor is fed as input to a predictor (e.g., LSTM) to predict road average speed data for each interval over the next N days. Concretely, this study employs a road network of 214 links, capturing 60 consecutive days of vehicle speed data at a temporal resolution of 144 intervals per day. Correspondingly, the road network is modeled as a 214×144×60 nonstandard tensor, and it is used to predicted the speed data on the 61st day. M2 (TW model) serves as the benchmark model, as it demonstrates the best performance among the nine state-of-the-art models in Section VI.B. Accordingly, the MTN-TRL and TW models are utilized to learn the low-dimensional representations of target nonstandard tensor, which serve as the basis for predicting vehicle speeds on Day 61. The prediction results show that MTN-TRL achieves RMSE and MAE values of 0.1801 and 0.1294, respectively, while the TW model yields 0.1865 and 0.1326. The experimental results indicate that MTN-TRL provides robust support for downstream tasks.

K. Summary

Based on the above experimental results, we draw the following summary:

1) Compared with other commonly used tensor networks, MTN-TRL has better performance owing to its excellent represent ability to a nonstandard tensor with mode unbalance, high-dimension, and incompleteness properties.
2) The hyper-parameter self-adaptation mechanism is necessary for MTN-TRL model because the model's performance is highly sensitive to hyper-parameter configurations.
3) The MSGD-based adaptive parameters learning scheme plays a pivotal role in constructing MTN-TRL model owing to its computational efficiency.
4) The MTN-TRL possesses remarkable scalability and generalization capabilities owing to its superior representation learning ability when handling higher-dimensional tensors with more extreme mode unbalance and higher-order nonstandard tensors.

V. Related Work

Tensor representation learning has performed well in many fields, such as traffic prediction, network embedding, image completion, etc. To date, researchers design various tensor network formats to build a TRL model. Li *et al*. [22] use tensor train subspace to represent features, and solve the problem of structural damage and excessive parameters caused by vectorization. Liu *et al*. [35] estimates the rank of a tensor ring through group sparse constraints, and introduces total variational components to enhance local consistency. Wu *et al*. [31] develop a tensor wheel decomposition by introducing a core tensor into TR and established the connection between multiple latent tensors. Yang *et al*. [36] propose a generalized transformed tensor representation learning model for the concurrent recovery of corrupted tensor data. Zhang *et al*. [37] efficiently handle third-order tensor data with partial and noisy observations by minimizing the maximum likelihood estimate.

The above methods all consider complete tensors, but real-world data are usually incomplete [11]–[15], [38]. Thus, Zhang

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TKDE.2025.3617894,

et al. [39] propose a sparse tensor decomposition model, which decomposes a tensor via sparse coding and learns its compact dictionary and sparse core tensor. Chen *et al*. [40] design a tensor low-rank learning model to explore the local and global patterns of high-dimensional data by constructing a sparse similarity matrix. Zhang *et al*. [41] develop a novel low-rank tensor regularized view recovery by reconstructing missing views and learning multi-level graphs to discover the consistency and complementarity between views. Bi *et al*. [42] use simplified graph convolution, prior convolution operators and local enhancement pooling schemes to efficiently obtain hidden information in incomplete data.

Furthermore, in order to achieve the effective representation of dynamic graphs with imbalanced modes, Hoang *et al*. [43] enhance the tensor by 2D index encoding and factorized it using the concept of tensor train for efficient completion of imbalanced and sparse low-rank tensors. Van Belle *et al*. [44] adopt a nearest neighbor search sampler, a heterogeneous graph neural network layer, and an aggregation function for category imbalance to solve the problems of large data scale, graph heterogeneity, and category imbalance faced in dynamic graph inductive learning. Qian *et al*. [45] use dynamic graphs and hybrid contrastive learning to enhance the expression of minority class features and improve the learning performance of unbalanced graphs. Sun *et al*. [46] design dynamic graph attention mechanism to accurately model label associations, and adopt double sampling and collaborative training strategies to effectively overcome the unbalanced label distribution in multi-label time series.

However, the above representation learning methods are confronted with high computational and storage complexity, making it difficult to efficiently represent large-scale dynamic graphs. Therefore, this study models a large-scale dynamic graph as a nonstandard tensor, correspondingly, the MTN is specially designed to represent such a nonstandard tensors,and an efficient MTN-TRL model is developed to learn the representation.

VI. Conclusions

Aiming to learning the accurate representation of the nonstandard tensor, this paper designs a Mode-Aware Tucker Network (MTN) and builds an MTN-based tensor representation learning model. Specifically, in MTN, a newly generating LF tensor is adopted to represent the high-dimensional mode of a nonstandard tensor. Furthermore, with data density-oriented modeling principle, an MTN-TRL model is built to learn the representations of nonstandard tensor efficiently and accurately, where an adaptive fast gradient descent algorithm is designed by fusing differential evolution and momentum methods. Finally, the experimental results on eight nonstandard tensors demonstrate that the proposed MTN-TRL performs better than other state-of-the-art models. Moreover, we also verified that MTN is more suitable for nonstandard tensor representation than other commonly used tensor network. Considering the future work,

1) It is highly meaningful to integrate neural network framework and MTN-TRL model (such as introducing activation functions in MTN) to construct more sophisticated model for capturing nonlinearity in a nonstandard tensor.
2) It is of great significance to design a GPU parallel computing framework compatible with the MTN-TRL model to enhance the efficiency of both model training and inference.

We plan to address the above issues in the future.

References

[1] X. Luo, H. Wu, and Z. Li, "Neulft: A novel approach to nonlinear canonical polyadic decomposition on high-dimensional incomplete tensors," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6148–6166, 2022.
[2] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, "A survey on embedding dynamic graphs," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–37, 2021.
[3] X. Li, K. Xie, X. Wang, G. Xie, K. Li, J. Cao, D. Zhang, and J. Wen, "Tripartite graph aided tensor completion for sparse network measurement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 1, pp. 48–62, 2023.
[4] Y. Zhou, X. Luo, and M. Zhou, "Cryptocurrency transaction network embedding from static and dynamic perspectives: An overview," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 5, pp. 1105–1121, 2023.
[5] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A contemporary and comprehensive survey on streaming tensor decomposition," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 10 897–10 921, 2023.
[6] X. Yang, H. Che, and M.-F. Leung, "Tensor-based unsupervised feature selection for error-robust handling of unbalanced incomplete multi-view data," *Inf. Fusion*, vol. 114, p. 102693, 2025.
[7] Y. Liu, J. Chen, Y. Lu, W. Ou, Z. Long, and C. Zhu, "Adaptively topological tensor network for multi-view subspace clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 5562–5575, 2024.
[8] Y. Gao, L. T. Yang, J. Yang, D. Zheng, and Y. Zhao, "Jointly low-rank tensor completion for estimating missing spatiotemporal values in logistics systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1814–1822, 2022.
[9] Y. Shen, C. Jin, J. Hua, and D. Huang, "Ttpnet: A neural network for travel time prediction based on tensor decomposition and graph embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4514–4526, 2020.
[10] C.-Y. Zhang, Z.-L. Yao, H.-Y. Yao, F. Huang, and C. P. Chen, "Dynamic representation learning via recurrent graph neural networks," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, no. 2, pp. 1284–1297, 2022.
[11] H. Wu, Y. Qiao, and X. Luo, "A fine-grained regularization scheme for non-negative latent factorization of high-dimensional and incomplete tensors," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3006–3021, 2024.
[12] F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang, "Outlier-resilient web service qos prediction," in *Proceedings of the Web Conference (WWW)*, 2021, pp. 3099–3110.
[13] J. Dong, Y. Song, M. Li, and H. Rao, "Biased collective latent factorization of tensors with transfer learning for dynamic qos data predicting," *Digit. Signal Process.*, vol. 146, p. 104360, 2024.
[14] H. Che, B. Pan, M.-F. Leung, Y. Cao, and Z. Yan, "Tensor factorization with sparse and graph regularization for fake news detection on social networks," *IEEE Trans. Comput. Soc. Syst.*, vol. 11, no. 4, pp. 4888–4898, 2024.
[15] T. Liu, J. Yang, B. Li, Y. Wang, and W. An, "Infrared small target detection via nonconvex tensor tucker decomposition with factor prior," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, pp. 1–17, 2023.
[16] H. Li, Z. Li, K. Li, J. S. Rellermeyer, L. Chen, and K. Li, "Sgd-tucker: A novel stochastic optimization strategy for parallel sparse tucker decomposition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1828–1841, 2020.
[17] B. Chen, J. Guan, Z. Li, and Z. Zhou, "Robust feature extraction via l_∞-norm based nonnegative tucker decomposition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 12, pp. 7144–7155, 2023.
[18] P. Tang, T. Ruan, H. Wu, and X. Luo, "Temporal pattern-aware qos prediction by biased non-negative tucker factorization of tensors," *Neurocomputing*, vol. 582, p. 127447, 2024.
[19] Q. Liu, L. Lu, and Z. Chen, "Non-negative tucker decomposition with graph regularization and smooth constraint for clustering," *Pattern Recognit.*, vol. 148, p. 110207, 2024.

[20] M. Wang, Y. Pan, Z. Xu, X. Yang, G. Li, and A. Cichocki, "Tensor networks meet neural networks: A survey and future perspectives," *arXiv preprint arXiv:2302.09019*, 2023.
[21] Y. Liu, J. Liu, Z. Long, and C. Zhu, *Tensor computation for data analysis*. Springer, 2022.
[22] G. Li, P. Xu, S. Peng, C. Wang, Y. Cai, and S. Yu, "Ttsr: Tensor-train subspace representation method for visual domain adaptation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7229–7241, 2024.
[23] M. Li, Y. Song, D. Ding, and R. Sun, "Triple factorization-based snlf representation with improved momentum-incorporated agd: A knowledge transfer approach," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 12, pp. 9448–9463, 2024.
[24] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 8, pp. 1754–1766, 2020.
[25] G.-G. Wang, D. Gao, and W. Pedrycz, "Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8519–8528, 2022.
[26] Z. Pan, S. Fang, and H. Wang, "Lightgbm technique and differential evolution algorithm-based multi-objective optimization design of ds-apmm," *IEEE Trans. Energy Convers.*, vol. 36, no. 1, pp. 441–455, 2020.
[27] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *Proceedings of the Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1724–1732.
[28] X. He, L. Chen, D. Cheng, V. Gupta *et al.*, "Revisit last-iterate convergence of msgd under milder requirement on step size," in *Proceedings of the 16th Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, 2022, pp. 36 559–36 570.
[29] Y. Miao and S. Xu, "Almost sure convergence of weighted sums," *Miskolc Math. Notes*, vol. 14, no. 1, pp. 173–181, 2013.
[30] R. Jin and X. He, "Convergence of momentum-based stochastic gradient descent," in *Proceedings of the 16th Int. Conf. Control Autom. (ICCA)*, 2020, pp. 779–784.
[31] Z.-C. Wu, T.-Z. Huang, L.-J. Deng, H.-X. Dou, and D. Meng, "Tensor wheel decomposition and its tensor completion application," in *Proceedings of the 16th Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, 2022, pp. 27 008–27 020.
[32] M. Wang, D. Hong, Z. Han, J. Li, J. Yao, L. Gao, B. Zhang, and J. Chanussot, "Tensor decompositions for hyperspectral data processing in remote sensing: A comprehensive review," *IEEE Geosci. Remote Sens. Mag.*, vol. 11, no. 1, pp. 26–72, 2023.
[33] X. Su, M. Zhang, Y. Liang, Z. Cai, L. Guo, and Z. Ding, "A tensor-based approach for the qos evaluation in service-oriented environments," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3843–3857, 2021.
[34] X. Luo, M. Chen, H. Wu, Z. Liu, H. Yuan, and M. Zhou, "Adjusting learning depth in nonnegative latent factorization of tensors for accurately modeling temporal patterns in dynamic qos data," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 2142–2155, 2021.
[35] J. Liu, C. Zhu, and Y. Liu, "Smooth compact tensor ring regression," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4439–4452, 2022.
[36] J.-H. Yang, C. Chen, H.-N. Dai, M. Ding, Z.-B. Wu, and Z. Zheng, "Robust corrupted data recovery and clustering via generalized transformed tensor low-rank representation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8839–8853, 2024.
[37] X. Zhang and M. K. Ng, "Sparse nonnegative tensor factorization and completion with noisy observations," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2551–2572, 2022.
[38] H. Wu, X. Wu, and X. Luo, *Dynamic Network Representation Based on Latent Factorization of Tensors*. Springer, 2023.
[39] J. Zhang, J. Chen, H. Yu, D. Yang, B. Liang, and M. Xing, "Polarization image demosaicking via nonlocal sparse tensor factorization," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–10, 2021.
[40] J. Chen, Z. Wang, H. Mao, and X. Peng, "Low-rank tensor learning for incomplete multiview clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 11 556–11 569, 2022.
[41] C. Zhang, H. Li, C. Chen, X. Jia, and C. Chen, "Low-rank tensor regularized views recovery for incomplete multiview clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9312–9324, 2024.
[42] F. Bi, T. He, Y.-S. Ong, and X. Luo, "Graph linear convolution pooling for learning in incomplete high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, pp. 1–14, 2024.
[43] P. M. Hoang, H. D. Tuan, T. T. Son, H. V. Poor, and L. Hanzo, "Learning unbalanced and sparse low-order tensors," *IEEE Trans. Signal Process.*, vol. 70, pp. 5624–5638, 2022.
[44] R. Van Belle and J. De Weerdt, "Shine: A scalable heterogeneous inductive graph neural network for large imbalanced datasets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 9, pp. 4904–4915, 2024.
[45] L. Qian, Q. Zuo, D. Li, and H. Zhu, "Dgmscl: A dynamic graph mixed supervised contrastive learning approach for class imbalanced multivariate time series classification," *Neural Networks*, p. 107131, 2025.
[46] L. Sun, C. Li, Y. Ren, and Y. Zhang, "A multitask dynamic graph attention autoencoder for imbalanced multilabel time series classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 11 829–11 842, 2024.

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.

Qu Wang (*Student Member*, IEEE) received the B.S. degree in Digital Media Technology from Chengdu University of Technology, Chengdu, China, in 2021. He received the M.S. degree in Computer Technology from the School of Computer and Information Science, Southwest University, Chongqing in 2025, and is currently pursuing the Ph.D. degree in computer science and technology there. His research interests include tensor networks and big data analysis.

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 Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 370 papers (including over 150 IEEE Transactions/Journal papers) in the areas of Artificial Intelligence and Data Science.

Zidong Wang (*Fellow*, IEEE) was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics from Suzhou University, Suzhou, China, in 1986, and the M.Sc. degree in applied mathematics and the Ph.D. degree in electrical engineering, both from Nanjing University of Science and Technology, Nanjing, China, in 1990 and 1994, respectively. He is currently a Professor of Dynamical Systems and Computing with the Department of Computer Science, Brunel University London, Uxbridge, U.K. He has published more than 700 articles in international journals. His research interests include Dynamical Systems, Signal Processing, Data Science, Bioinformatics, Control Theory and their applications.