A MULTILAYER LATENT FACTORIZATION OF NONSTANDARD TENSORS MODEL FOR ACCURATELY REPRESENTING HIGHER-ORDER GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Higher-order graphs characterize complex interaction systems involving multiple types of entities and relations, and can be naturally modeled as nonstandard tensors with extreme sparsity and high dimensionality. Tensor decomposition models have been widely used to learn low-dimensional representations of such tensors for higher-order graph inference. Existing tensor decomposition methods typically employ single-layer architectures, limiting their ability to learn expressive representations. To address this limitation, we propose a Multilayer Latent Factorization of Nonstandard tensors (MLFN) model, which enables progressive nonlinear representation learning through multiple latent factorization layers. Specifically, MLFN incorporates three key innovations: 1) a multilayer learning framework that progressively learns low-dimensional representations through layer-by-layer feature learning, 2) a nonlinear activation-based sampling strategy that generates informative training samples to alleviate the extreme sparsity of nonstandard tensor data, and 3) an efficient parallel training scheme using an alternating stochastic gradient descent for scalable model optimization. Experimental results on four types of higher-order graphs demonstrate that MLFN improves the average prediction accuracy by 22.81% over state-of-the-art methods for unobserved-link prediction.

1 INTRODUCTION

Higher-order graphs model complex interaction systems involving multiple entity and relation types, including social, transportation, and bibliographic networks (Tian & Zafarani, 2024). As illustrated in Figure 1, a bibliographic network may contain authors, papers, conferences, and terms, with author relationships induced through higher-order interaction paths such as author-paper-author and author-paper-conference-paper-author. Such higher-order interactions encode rich relational information, and recent higher-order graph embedding studies emphasize the importance of learning low-dimensional representations that preserve complex structural correlations (Fu et al., 2026). Accurate representation is therefore essential for extracting useful knowledge from higher-order graphs and supporting downstream tasks such as relationship prediction. However, real-world higher-order graphs are typically high-dimensional and sparse because only a small fraction of nodes interact. For example, the citation network considered in this study contains 51,530 interactions among 17,411 authors, 17,934 papers, and 12,313 terms, yielding an extremely low link density of 1.34×10^{-8}. Accurately representing such sparse, high-dimensional higher-order graphs is therefore challenging.

Tensors naturally preserve multiway graph structure (Wu et al., 2023), while tensor decomposition maps target tensors into low-dimensional latent spaces. Recent studies have explored tensor-based higher-order graph embedding, dynamic graph representation, spatiotemporal tensor completion, and multiscale tensor representations (Fu et al., 2026; Ghosh et al., 2025; Shu et al., 2024; Zhang et al., 2025). However, most existing tensor decomposition models use a single-layer architecture. As shown in Figure 2, when tensor data density decreases from 80% to 20%, representation accuracy declines while the required training iterations increase, indicating the dependence of single-layer models on sufficiently dense training data (Wang et al., 2025a; Wu et al., 2023). This limitation is particularly severe for nonstandard tensors modeling higher-order graphs. For instance, a tensor modeling two author-citation relations in this study has size $17,411 \times 17,411 \times 2$ but contains

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

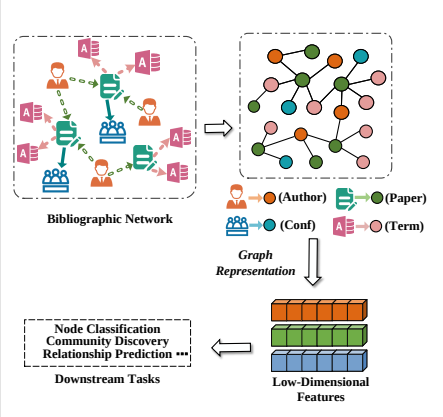


Figure 1: Examples of higher-order graph representation learning.

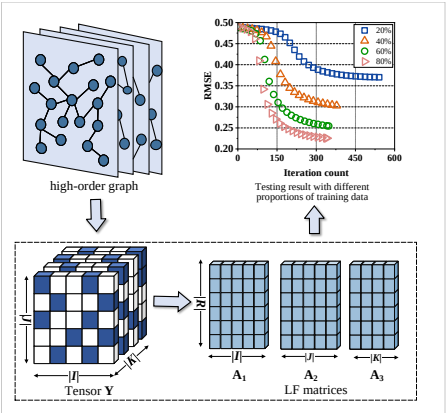


Figure 2: The tensor decomposition-based representation learning model.

only 51,530 observed elements, corresponding to a data density of 0.0085%. Existing approaches often pre-fill missing entries or introduce an additional indicator tensor, increasing computational and storage costs and potentially biasing learning toward imputed data. Consequently, accurately representing such extremely sparse nonstandard tensors remains difficult (Saragadam et al., 2024; Wu et al., 2024).

To address these limitations, we propose a Multilayer Latent Factorization of Nonstandard tensors (MLFN) model that progressively refines latent representations across multiple factorization layers. A nonlinear activation-based sampling mechanism enriches the effective training information at successive layers, while alternating stochastic gradient descent enables efficient optimization. The main contributions are as follows:

1) A progressive multilayer latent factorization paradigm for nonstandard tensors. It hierarchically refines latent representations across layers, extending the representation capacity of conventional single-layer factorization for extremely sparse and high-dimensional tensors.

2) A nonlinear sampling mechanism for progressive representation refinement. It generates informative training samples during multilayer learning, enriching the observations available to subsequent layers and alleviating extreme sparsity.

3) A theoretical convergence guarantee for MLFN. The convergence of the proposed multilayer learning process under alternating stochastic gradient descent is theoretically established.

Experiments on four nonstandard tensors derived from higher-order graphs show that MLFN consistently outperforms state-of-the-art methods in representation accuracy. The remainder of the main paper presents preliminaries, methodology, experiments, and conclusions; detailed proof, related work, and some experiment descriptions are in Appendix.

2 Preliminaries

2.1 A Nonstandard Tensor Modeling a Higher-order Graph

A higher-order graph is defined as $G = (V, E, H, \phi, \varphi)$, where V and E denote the node and edge sets, and H denotes the set of higher-order relations. Each $h \in H$ connects two or more nodes, and $|H| > 1$ indicates the presence of multiple higher-order relations. The mappings $\phi : V \to T_V$ and $\varphi : E \to T_E$ assign node and edge types, respectively.

As shown in Figure 3, higher-order relations between authors in a bibliographic network can be characterized by different relation meta-paths, such as author-paper-author (APA), author-paper-conf-paper-author (APCPA), and author-paper-term-paper-author (APTPA). Each relation defines a subgraph that can be represented by an "Author-Author" adjacency matrix, whose known entries indicate whether two authors are connected by the corresponding relation. Since only a small fraction of author pairs exhibit each relation, these adjacency matrices are highly sparse. For example, an observed entry in the APCPA matrix indicates that two authors publish at the same conference.

By stacking the adjacency matrices of all relation subgraphs, we construct a third-order "Author-Author-Relation" nonstandard tensor $\mathbf{Y} \in \mathbb{R}^{|I|\times|J|\times|K|}$, where $|I| = |J| = |V|$. Each known entry y_{ijk} indicates that authors i and j are connected by relation k. Since each relation-specific adjacency matrix is highly sparse, the resulting tensor $\mathbf{Y}$ also has extremely low data density.

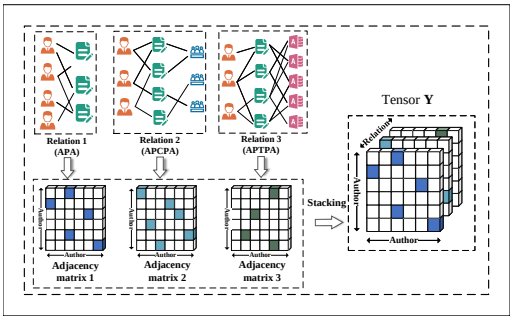


Figure 3: A nonstandard tensor modeling a higher-order graph.

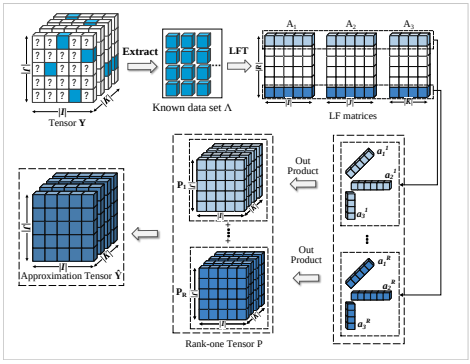


Figure 4: Latent factorization of tensors.

2.2 Latent Factorization of Tensors (LFT)

Given a third-order tensor $\mathbf{X} \in \mathbb{R}^{|I|\times|J|\times|K|}$ with missing elements, an LFT model based on canonical polyadic tensor factorization (CPTF) (Wang et al., 2024; Liao et al., 2025) represents $\mathbf{X}$ using three latent feature (LF) matrices $\mathbf{A}_1 \in \mathbb{R}^{|I|\times R}$, $\mathbf{A}_2 \in \mathbb{R}^{|J|\times R}$, and $\mathbf{A}_3 \in \mathbb{R}^{|K|\times R}$ learned from its known data, as illustrated in Figure 4. For the r-th component, the outer product $\mathbf{P}_r = \mathbf{a}_1^r \circ \mathbf{a}_2^r \circ \mathbf{a}_3^r$ forms a rank-one tensor, whose element is

$$p_{ijk}^r = a_1^{ir} a_2^{jr} a_3^{kr}. \tag{1}$$

Summing R rank-one tensors yields the rank-R approximation $\hat{\mathbf{X}} \in \mathbb{R}^{|I|\times|J|\times|K|}$, whose element is

$$\hat{x}_{ijk} = \sum_{r=1}^{R} p_{ijk}^r = \sum_{r=1}^{R} a_1^{ir} a_2^{jr} a_3^{kr}. \tag{2}$$

The LF matrices are learned by minimizing the reconstruction error between $\mathbf{X}$ and $\hat{\mathbf{X}}$ over the known data set Λ, with L_2 regularization to avoid overfitting (Wang et al., 2025b):

$$\varepsilon = \sum_{x_{ijk}\in\Lambda} \left(\left(x_{ijk} - \sum_{r=1}^{R} a_1^{ir} a_2^{jr} a_3^{kr} \right)^2 + \lambda \sum_{r=1}^{R} \left((a_1^{ir})^2 + (a_2^{jr})^2 + (a_3^{kr})^2 \right) \right). \tag{3}$$

where λ denotes the regularization coefficient. Minimizing Equation 3, e.g., by stochastic gradient descent, yields the LF matrices for downstream tasks such as unobserved-link estimation.

3 An MLFN Model

3.1 Structure of MLFN

Figure 5 illustrates the multilayer architecture of the proposed MLFN model.

At the first layer, the target nonstandard tensor $\mathbf{Y}$ is taken as the input, and its observed entries, denoted by Λ, are used to learn the LF matrices $\mathbf{U}_1$, $\mathbf{S}_1$ and $\mathbf{T}_1$ through the LFT model. Based on the proposed sampling strategy, a set of unobserved entries in $\mathbf{Y}$ is then selected as the samples to be predicted and generated. Their predicted values are calculated from $\mathbf{U}_1$, $\mathbf{S}_1$ and $\mathbf{T}_1$ and subsequently passed through the nonlinear activation function. The activated predictions form the first generated data set Ω_1, which provides additional observations for the subsequent learning layer.

At the intermediate layers, taking the second layer as an example, the generated data set Ω_1 is first filled into the corresponding unobserved entries of $\mathbf{Y}$ to form the second-layer input tensor $\mathbf{Y}_1$. The observed data for the second layer consist of the original observed set Λ and the newly generated

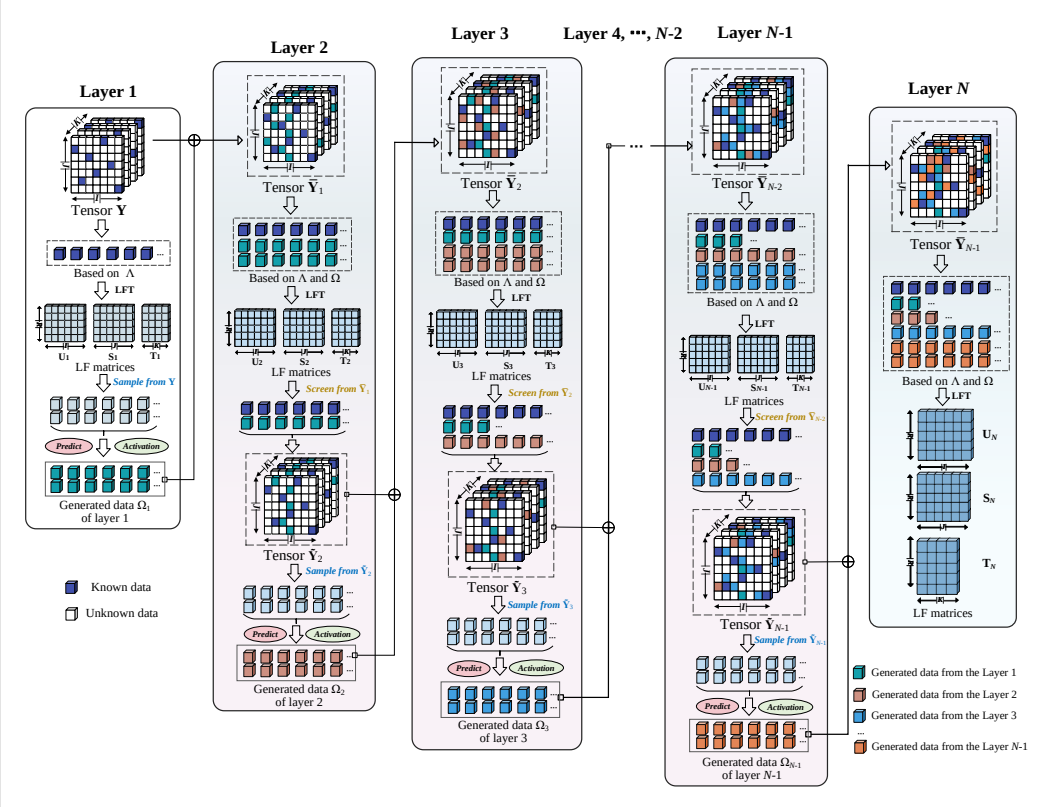


Figure 5: Structure of MLFN.

samples in Ω_1, based on which the LFT model learns the LF matrices $\mathbf{U}_2$, $\mathbf{S}_2$ and $\mathbf{T}_2$. Next, the low-confidence elements in Ω_1 are removed from $\bar{\mathbf{Y}}_1$ according to the proposed sampling strategy, yielding a refined tensor $\tilde{\mathbf{Y}}_2$. A new set of unobserved entries is then selected from $\tilde{\mathbf{Y}}_2$, and the corresponding values are predicted using $\mathbf{U}_2$, $\mathbf{S}_2$ and $\mathbf{T}_2$. After nonlinear activation, the predictions form the second generated data set Ω_2. The third to the $N-1$ layers follow the same procedure, through which reliable generated samples are progressively incorporated into the input tensor $\bar{\mathbf{Y}}_n$ and the effective data density is gradually increased.

At the N-th layer, $\bar{\mathbf{Y}}_{N-1}$ is taken as the input tensor. Its observed entries consist of the original observed set Λ and the reliable generated samples accumulated from the preceding layers. The LFT model is finally applied to these observations to obtain the LF matrices $\mathbf{U}_N$, $\mathbf{S}_N$ and $\mathbf{T}_N$, which serve as the final latent features of the target nonstandard tensor $\mathbf{Y}$ for higher-order graph representation.

3.2 Loss function

At the n-th layer, the LF matrices are jointly optimized over the original known data set Λ and the generated data set Ω using reconstruction, fitting, and regularization terms.

Reconstruction term. The reconstruction error on the original known data set Λ is

$$\varepsilon_{\text{recon}}^{n} = \frac{1}{2}\sum\nolimits_{\langle i,j,k\rangle\in\Lambda}\left(y_{ijk} - \sum\nolimits_{r=1}^{R} u_{ir}^{n} s_{jr}^{n} t_{kr}^{n}\right)^{2}. \tag{4}$$

This term preserves the information contained in the original observations.

Fitting term. The fitting error on the generated data set Ω is weighted by the hyperparameter α:

$$\varepsilon_{\text{synth}}^{n} = \frac{1}{2}\alpha\sum\nolimits_{\langle i,j,k\rangle\in\Omega}\left(\bar{y}_{ijk} - \sum\nolimits_{r=1}^{R} u_{ir}^{n} s_{jr}^{n} t_{kr}^{n}\right)^{2}. \tag{5}$$

Here, α balances the contribution of generated data against the original known data.

Regularization term. An L_2-norm regularization term is imposed on the LF matrices to prevent overfitting and improve numerical stability:

$$\varepsilon_{\text{reg}}^{n} = \frac{\lambda}{2} \sum\nolimits_{r=1}^{R} \left((u_{ir}^{n})^2 + (s_{jr}^{n})^2 + (t_{kr}^{n})^2 \right). \tag{6}$$

The composite loss for the n-th layer is therefore

$$\varepsilon_n = \varepsilon_{\text{recon}}^{n} + \varepsilon_{\text{synth}}^{n} + \varepsilon_{\text{reg}}^{n}. \tag{7}$$

3.3 Sampling Strategy

The sampling strategy is described in detail with reference to Figs. 5 and 6. The subsequent strategy is categorized into two cases according to the layer n.

In the case of sampling at the first layer ($n = 1$), the blank index y_{ijk} is selected sequentially starting from each row of the frontal slice of the $\mathbf{Y}$. If no observed data exists in the current row, an index is chosen randomly. Otherwise, the blank positions between two known data are selected in order. Then the trained U_1, S_1, and T_1 are adopted to predict the selected blank index. The predicted values $\hat{y}_{ijk}$ are further mapped via the activation function in Equation 8, corresponding to the activation module in Fig. 5. The newly generated data after mapping is then added to the generated dataset Ω_1.

$$\bar{y}_{ijk} = \begin{cases} \hat{y}_{ijk} = \sum\nolimits_{r=1}^{R} u_{ir}^{n} s_{jr}^{n} t_{kr}^{n}, & y_{\min} \le \hat{y}_{ijk} \le y_{\max}, \\ y_{\min} + \max(0, \min(1, 0.2\hat{y}_{ijk} + 0.5)), & \hat{y}_{ijk} < y_{\min}, \\ y_{\max} \cdot \max(0, \min(1, 0.2\hat{y}_{ijk} + 0.5)), & \hat{y}_{ijk} > y_{\max}, \end{cases} \tag{8}$$

The design idea of Equation 8 is derived from the Hard-Sigmoid activation function Glorot & Bengio (2010), and its typical form is $\max(0, \min(1, az + b))$. In this formula, we adopt the classic parameter combination of $a = 0.2$ and $b = 0.5$ in engineering. The bias term b ensures that when the original network output $\hat{y}_{ijk} = 0$, the activated value is at the midpoint of the interval, the scaling factor a extends the effective linear response region of the function and avoids the gradient saturation problem. Based on this core mapping, Equation 8 further integrates de-normalization and boundary clipping to restore the compressed intermediate value to the target physical interval, and constructs an adaptive output module combined with the idea of linear decoding. After traversing all sliced matrices, the augmented generated data together with the original observed data constitute a new $\bar{\mathbf{Y}}_1$, as shown in the latter part of Fig. 6, which is adopted for the training of the second layer. Notably, the generation of sampled data is not unlimited. The generation process at the current layer terminates once the number of generated samples reaches that of the observed entries in the current tensor.

When the sampling is conducted at the n layers ($1 < n < N$), this paper carries out the following operations. The generated data from the previous layer is filtered (screen operation in Figs. 5 and 6), and the confidence indicator $C(x)$ is introduced to measure the consistency between the values of $\bar{y}_{ijk}$ in Ω and the predicted values of $\hat{y}_{ijk}$ at the corresponding positions generated using U_n, S_n, and T_n. The closer $C(x)$ is to 1, the higher the confidence is, and the corresponding generated data is also more consistent with the real interaction pattern in the network.

$$C(x) = \frac{1}{1 + |\hat{y}_{ijk} - \bar{y}_{ijk}|}. \tag{9}$$

A dynamic threshold δ is used to judge the value of $C(x)$. This threshold is adaptively calculated based on the distribution of confidence score Guastavino et al. (2024) within the generated dataset Ω. If $C(x) \ge \delta$, the generated data can continue to be used for the training of the next layer, otherwise the data is removed. As shown in Layer 2 in Fig. 5, the threshold for a given layer is computed as:

$$\delta = C_{\max}(x_1) - 0.2\left(C_{\max}(x_1) - C_{\min}(x_2)\right). \tag{10}$$

where x_1 and x_2 represent the specific data points that attain the maximum and minimum confidence values $C_{max}(x_1)$ and $C_{min}(x_2)$, respectively. The threshold is intrinsically anchored to the upper extreme of the observed confidence distribution $C_{max}(x_1)$. The subtraction of a fraction (20%) of the total confidence range ensures that the threshold dynamically adjusts to the dispersion of the data. This mechanism guarantees that the screening remains highly selective, consistently retaining

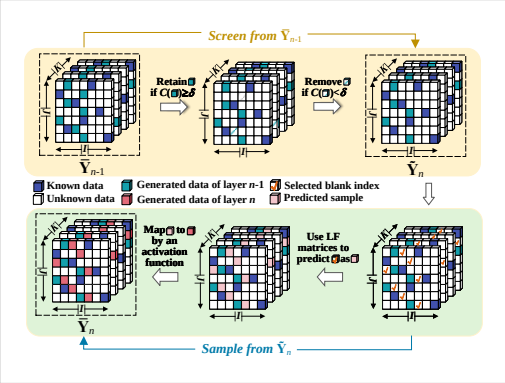


Figure 6: Example of generating synthetic data at the n-th layer.

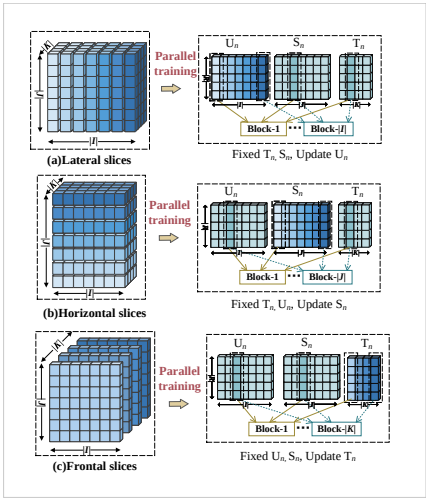


Figure 7: Parallelizing each training epoch of each layer of MLFN: fixing $\mathbf{U}_n$, $\mathbf{S}_n$ to parallel train $\mathbf{T}_n$.

only the most reliable predictions located in the top portion of the confidence spectrum. Therefore, a refined tensor $\tilde{\mathbf{Y}}_n$ is obtained by removing the low-confidence generated data in Ω of the current layer. After the screen operation for tensor $\tilde{\mathbf{Y}}_n$, the sample operation similar to Layer 1 and the activation function mapping operation are continued to obtain the generated data of this layer.

3.4 Training the Nth layer with ASGD

According to update rules of standard SGD, we formulate the update rules of MLFN on known or generated data sets. This method is conducive to the independent update of the LF matrices $\mathbf{U}_n$, $\mathbf{S}_n$ and $\mathbf{T}_n$. The core update mechanism for any parameter θ in three LF matrices follows the gradient descent principle:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \varepsilon_n. \tag{11}$$

where η is the learning rate and ε_n is the instantaneous loss from Equation 7. From the formula, we can see that the three LF matrices of each layer in the MLFN model are iteratively trained. The training process of LF matrices $\mathbf{U}_n$, $\mathbf{S}_n$ and $\mathbf{T}_n$ are interdependent. The ASGD algorithm can be used to perform efficient parallelization of LFT (Shi et al., 2022; Luo et al., 2012). It actually fixes two of $\mathbf{U}_n$, $\mathbf{S}_n$ and $\mathbf{T}_n$ alternately to train the other one on each iteration of the training task.

Figure 7 illustrates this parallelization scheme. For example, when updating $\mathbf{U}_n$, the input tensor $\tilde{\mathbf{Y}}_{n-1}$ is sliced according to the number of slices 1–$|I|$. The rows of $\mathbf{U}_n$ are subsequently partitioned into blocks that align with these segments, permitting their concurrent update. This block-wise parallel processing is similarly applied to $\mathbf{S}_n$ and $\mathbf{T}_n$.

The specific parallel update rules are derived by applying the gradient descent principle to the instantaneous loss functions. Taking the update of an element u^n_{ir} in $\mathbf{U}_n$ as an example (with $\mathbf{S}_n$ and $\mathbf{T}_n$ fixed), the rules are as follows. For a known entry in Λ:

$$\begin{cases} u^n_{1r} \leftarrow u^n_{1r} + \eta s^n_{jr} t^n_{kr} \left(y_{1jk} - \sum_{q=1}^{R} u^n_{1q} s^n_{jq} t^n_{kq} \right) - \eta \lambda u^n_{1r}, & \{1,j,k\} \in \Lambda, \\ u^n_{ir} \leftarrow u^n_{ir} + \eta s^n_{jr} t^n_{kr} \left(y_{ijk} - \sum_{q=1}^{R} u^n_{iq} s^n_{jq} t^n_{kq} \right) - \eta \lambda u^n_{ir}, & \{i,j,k\} \in \Lambda, \\ u^n_{|I|r} \leftarrow u^n_{|I|r} + \eta s^n_{jr} t^n_{kr} \left(y_{|I|jk} - \sum_{q=1}^{R} u^n_{|I|q} s^n_{jq} t^n_{kq} \right) - \eta \lambda u^n_{|I|r}, & \{|I|,j,k\} \in \Lambda. \end{cases} \tag{12}$$

For a synthetic entry in Ω:

$$\begin{cases} u_{1r}^n \leftarrow u_{1r}^n + \eta s_{jr}^n t_{kr}^n \alpha \left(\bar{y}_{1jk} - \sum_{q=1}^{R} u_{1q}^n s_{jq}^n t_{kq}^n \right) - \eta \lambda u_{1r}^n, & \{1, j, k\} \in \Omega, \\ u_{ir}^n \leftarrow u_{ir}^n + \eta s_{jr}^n t_{kr}^n \alpha \left(\bar{y}_{ijk} - \sum_{q=1}^{R} u_{iq}^n s_{jq}^n t_{kq}^n \right) - \eta \lambda u_{ir}^n, & \{i, j, k\} \in \Omega, \\ u_{|I|r}^n \leftarrow u_{|I|r}^n + \eta s_{jr}^n t_{kr}^n \alpha \left(\bar{y}_{|I|jk} - \sum_{q=1}^{R} u_{|I|q}^n s_{jq}^n t_{kq}^n \right) - \eta \lambda u_{|I|r}^n, & \{|I|, j, k\} \in \Omega. \end{cases} \tag{13}$$

Noting that after $\mathbf{U}_n$ is trained on all the data, we train $\mathbf{T}_n$ and $\mathbf{S}_n$ respectively in parallel by fixing the other matrices at the same epoch. With the above update rules, the MLFN model is obtained. The complete workflow and algorithmic details of MLFN are provided in Appendix B.

3.5 Convergence Analysis

To establish the theoretical reliability of MLFN, we analyze the convergence of its ASGD-based training and state the following theorem.

Theorem 1. *Given a nonstandard tensor $\mathbf{Y}$, under the smoothness and strong-convexity assumptions adopted in Appendix D and the $\alpha = 1$ setting used in the convergence analysis, the alternating ASGD updates of the three LF matrices at each layer of MLFN converge when minimizing the layer-wise objective over the original observed set Λ and generated set Ω; consequently, the corresponding layer loss converges.*

Proof. Since each layer is optimized independently, it is sufficient to establish convergence for an arbitrary n-th layer. The detailed proof shows that the instantaneous loss ε_n is L-smooth and strongly convex under the adopted assumptions. Applying the ASGD update then makes the expected distance between the updated and optimal parameters decrease iteratively and converge to zero. Consequently, the parameters and the corresponding loss converge. The complete proof of Theorem 1 is provided in Appendix D.

4 Experimental Results and Discussion

4.1 General Settings

Dataset. We conduct experiments on four widely used higher-order graphs: Cora (D1), Aminer (D2), Douban Book (D3), and Yelp (D4), each modeled as a "Node-Node-Relation" third-order nonstandard tensor with extremely low data density. Detailed dataset construction, entity statistics, relation meta-paths, and tensor statistics are provided in Appendix C.1 and Table 1.

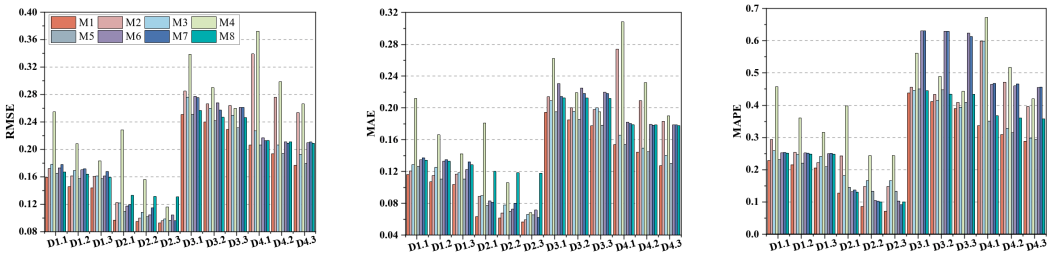
Evaluation Metrics. Prediction accuracy for unobserved relations is evaluated using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). Their definitions are provided in Equation 17 in Appendix C.2; lower values indicate higher prediction accuracy.

Comparison Models. We compare MLFN (M1) with HDOP (M2) (Wang et al., 2019), CTF (M3) (Ye et al., 2021), TCA (M4) (Su et al., 2021), DNL (M5) (Luo et al., 2021), HCCF (M6) (Xie et al., 2022a), SHT (M7) (Xie et al., 2022b), and MGDCF (M8) (Hu et al., 2024).

Experimental Setup. For fairness, each dataset is randomly split into ten disjoint subsets under training-validation-testing ratios of 3:1:6, 5:1:4, and 7:1:2, producing D1.1–D1.3 and analogously D2–D4; experiments are repeated 10 times to reduce data bias. Hyper-parameters are selected by grid search, and the latent dimension R of all tensor-decomposition models is set to 20 following prior research (Wu & Mi, 2025). MLFN uses 20 layers and 16 parallel threads. All models share the same early-stopping rule: training stops at 1000 iterations or when the error difference between two consecutive iterations is below 10^{-5}.

4.2 Comparison with State-of-the-art Models

We compare MLFN (M1) with seven SOTA models to evaluate representation accuracy. Figure 8 visualizes the RMSE, MAE, and MAPE of M1–M8 over the 12 testing cases. Complete numerical results, including Win/Loss counts and Friedman ranks, are reported in Table 3 in Appendix C.3; Wilcoxon Signed-Ranks Test results are in Table 2 in Appendix C.2. The results yield three findings.



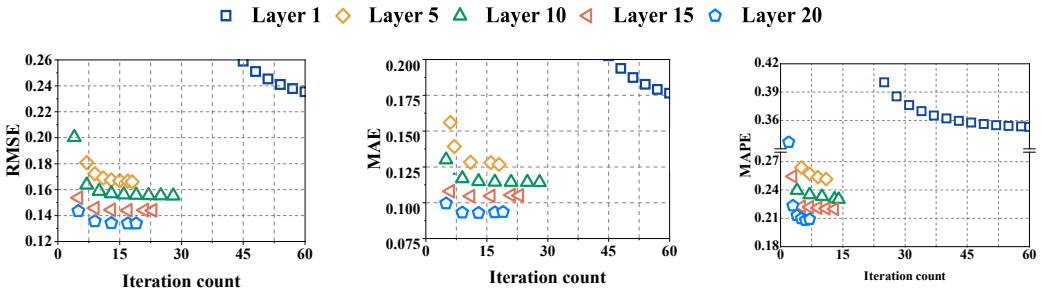
(a) RMSE of M1-8 on D1-4 (b) MAE of M1-8 on D1-4 (c) MAPE of M1-8 on D1-4

Figure 8: Prediction accuracy of M1–M8 on D1–D4

MLFN achieves the highest prediction accuracy among the tested models. As visualized in Figure 8 and quantified in Table 3 in Appendix C.3, M1 obtains lower RMSE, MAE, and MAPE in most cases. On D1.1, its RMSE is 0.1592, 3.83%–60.11% lower than M2–M8, while its MAE is 0.1163, 3.70%–82.20% lower. On D3.1, MLFN obtains a MAPE of 0.4379, 1.71%–44.05% lower than the competing models. These results support the multilayer design, which increases effective data density for subsequent latent-factor learning.

MLFN maintains its advantage as the number of training samples increases. Figure 8 shows the overall trend, while the exact values are reported in Table 3 in Appendix C.3. On D4, the RMSE decreases from 0.2061 on D4.1 to 0.1936 on D4.2 and 0.1769 on D4.3 as the training proportion increases. On D4.3, its MAE is 0.1272, 2.28%–49.53% lower than M2–M8. Similar trends on other datasets and metrics indicate that increased data density benefits performance while MLFN remains competitive.

The performance improvement is statistically significant. MLFN wins all 36 comparisons and achieves the lowest Friedman rank in Table 3 in Appendix C.3. The Wilcoxon Signed-Ranks Tests in Appendix C.2 yield $p = 0.0039$ and $R^- = 0$ for M2–M8 versus M1, confirming statistically significant improvements in prediction accuracy.



(a) RMSE on D1.3 (b) MAE on D1.3 (c) MAPE on D1.3

Figure 9: MLFN training on D1.3 across layers in terms of RMSE, MAE, and MAPE.

4.3 Impact of Multilayer Structure

We evaluate MLFN as the number of layers increases under the 7:1:2 split. Representative D1.3 training curves for RMSE, MAE, and MAPE are shown in Figure 9, while the complete D1.3–D4.3 results are provided in Appendix C.4. These results yield two observations.

Deeper layers accelerate convergence. For example, on D1.3, the required iterations decrease from 143 at Layer 1 to 23 at Layer 10 and 12 at Layer 20, while RMSE, MAE, and MAPE decrease until convergence. Appendix C.4 shows the same trend on other datasets, indicating that synthetic samples across layers improve training efficiency and stability.

Additional layers improve representation accuracy with diminishing returns. For example, on D1.3, RMSE decreases from 0.2081 at the first layer to 0.1628 at the second layer, while MAE decreases from 0.1625 to 0.1244, followed by further gradual improvements. Appendix C.4 shows that 20th-layer RMSE, MAE, and MAPE remain substantially below first-layer values in most cases, with diminishing gains as depth increases.

4.4 Ablation Study with Sampling Strategy

To evaluate the core components of MLFN, we construct MLFN-1, which uses predictive sampling without screening, and MLFN-2, which uses neither sampling nor screening. A representative D1–D2 result is shown in Figure 10; the remaining ablation panels are provided in Appendix C.6.

Across experiments, MLFN consistently outperforms MLFN-1, which in turn outperforms MLFN-2. On D1.3, MLFN reduces RMSE from 0.1601 to 0.1439, MAE from 0.1181 to 0.1039, and MAPE from 23.47% to 20.59% relative to MLFN-1, corresponding to improvements of 11.26%, 13.67%, and 13.99%, respectively. Appendix C.6 confirms the same pattern: sampling improves accuracy, while screening further refines generated samples for later layers.

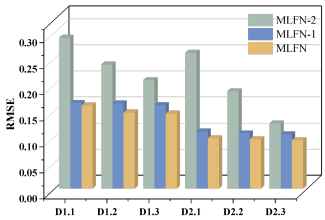
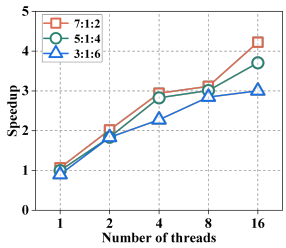
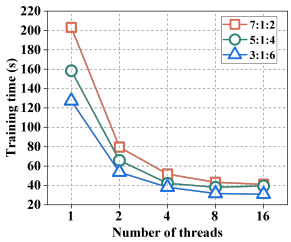


Figure 10: Representative RMSE comparison of MLFN and its variants on D1–D2.



(a) Speedup on D1



(b) Training time on D1

Figure 11: Parallel training on D1: (a) speedup and (b) training time.

4.5 Impact of Parallel Training Method

We evaluate parallel-training efficiency using the speedup

$$\text{Speedup} = \frac{T_{\text{serial}}}{T_t}. \tag{14}$$

where T_{serial} and T_t denote serial and t-thread training times, respectively. We evaluate $t = 1$–16; Figure 11 shows representative D1 results, while Appendix C.5 reports D2–D4.

Figure 11 shows that more threads increase speedup and reduce training time. On D1, the 16-thread speedup rises from 3.102 on D1.1 to 4.156 on D1.3 as data density increases. The same trend appears on D2–D4 (Appendix C.5); for example, on D3.3, speedups with 2, 4, 8, and 16 threads are 2.277, 2.968, 4.059, and 6.898, while training time drops from 208.617 s to 68.863 s. Thus, thread parallelism and higher data density improve MLFN's computational performance.

5 Conclusion

This paper proposes MLFN for higher-order graph representation through progressive multilayer factorization of extremely sparse, high-dimensional nonstandard tensors. Nonlinear activation-based sampling and ASGD enable effective and efficient multilayer learning with theoretically established convergence. Experiments on four nonstandard tensors show higher unobserved-link prediction accuracy than state-of-the-art methods. Future work will explore multilayer frameworks with other tensor decomposition methods and GPU-based optimization.

REFERENCES

F. Al-Obeidat, A. Rocha, M. S. Khan, et al. Parallel tensor factorization for relational learning. *Neural Computing and Applications*, 34(11):8455–8464, 2022.

Yiyang Fu, Huiling Fu, Yuwu Lu, and Ming Zhao. Deep non-convex tensor and higher-order graph embedding for multi-source domain adaptation. *Pattern Recognition*, 176:113170, 2026. doi: 10.1016/j.patcog.2026.113170.

Y. Gao, Y. Feng, S. Ji, and R. Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2023.

M. Ghosh, R. Halder, and J. Chandra. Treat: Temporal and relational attention-based tensor representation learning for ethereum phishing users. *IEEE Transactions on Services Computing*, 2025. DOI: 10.1109/TSC.2025.3568277.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

S. Guastavino, M. Piana, and F. Benvenuto. Bad and good errors: Value-weighted skill scores in deep ensemble learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2): 1993–2002, 2024.

J. Hu, B. Hooi, S. Qian, Q. Fang, and C. Xu. Mgdcf: distance learning via markov graph diffusion for neural collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3281–3296, 2024.

H. Huang and J. Yang. A unified framework for graph and hypergraph neural networks. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2563–2569, 2023.

X. Huang, Y. Hu, L. Chen, W. Chen, and Y. Huang. Structural change point detection in temporal networks via mdl-guided tensor decomposition. *Pattern Recognition*, 177:113323, 2026a.

Xingyue Huang, Mikhail Galkin, Michael Bronstein, and Ismail I. Ceylan. HYPER: A foundation model for inductive link prediction with knowledge hypergraphs. In *International Conference on Learning Representations*, 2026b.

J. G. Jang and U. Kang. Static and streaming tucker decomposition for dense tensors. *ACM Transactions on Knowledge Discovery from Data*, 17(5):1–34, 2023.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and Ismail I. Ceylan. Flock: A knowledge graph foundation model via learning on random walks. In *International Conference on Learning Representations*, 2026.

X. Liao, H. Wu, and X. Luo. A novel tensor causal convolution network model for highly-accurate representation to spatio-temporal data. *IEEE Transactions on Automation Science and Engineering*, 22:19525–19537, 2025.

H. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu. A parallel matrix factorization-based recommender by alternating stochastic gradient descent. *Engineering Applications of Artificial Intelligence*, 25(7): 1403–1412, 2012.

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 Transactions on Automation Science and Engineering*, 18(4):2142–2155, 2021.

Emanuele Mule, Stefano Fiorini, Antonio Purificato, Federico Siciliano, Stefano Coniglio, and Fabrizio Silvestri. Directional sheaf hypergraph networks: Unifying learning on directed and undirected hypergraphs. In *International Conference on Learning Representations*, 2026.

M. Pang, K. M. Ting, P. Zhao, and Z. H. Zhou. Improving deep forest by confidence screening. In *Proceedings of the 2018 IEEE International Conference on Data Mining*, pp. 1194–1199, 2018.

V. Saragadam, R. Balestriero, A. Veeraraghavan, and R. G. Baraniuk. Deeptensor: Low-rank tensor decomposition with deep network priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10337–10348, 2024.

X. Shi, Q. He, X. Luo, Y. Bai, and M. Shang. Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems. *IEEE Transactions on Big Data*, 8(2):420–431, 2022.

H. Shu, H. Wang, J. Peng, and D. Meng. Low-rank tensor completion with 3-d spatiotemporal transform for traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, 25(11):18673–18687, 2024.

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 Transactions on Network and Service Management*, 18(3):3843–3857, 2021.

H. Tian and R. Zafarani. Higher-order networks representation and learning: a survey. *ACM SIGKDD Explorations Newsletter*, 26(1):1–18, 2024.

H. Wang, Y. Chen, M. Yao, W. Liu, J. Peng, and X. Fu. Tensor completion framework by graph refinement for incomplete multi-view clustering. *IEEE Transactions on Multimedia*, 2025a. DOI: 10.1109/TMM.2025.3613125.

L. Wang, X. Wu, Y. Yu, Y. Qiu, S. Xie, and Q. Zhao. Nonnegative latent factorization of tensors with swish regularization for water quality data imputation. In *Proceedings of the 12th International Conference on Communication and Broadband Networking*, pp. 201–207, 2024.

Q. Wang, H. Wu, and X. Luo. A convolution bias-incorporated non-negative latent factorization of tensors model for accurate representation learning to dynamic directed graphs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2025b. DOI: 10.1109/TSMC.2025.3611792.

S. G. Wang, Y. Ma, B. Cheng, F. C. Yang, and R. N. Chang. Multi-dimensional qos prediction for service recommendations. *IEEE Transactions on Services Computing*, 12(1):47–57, 2019.

H. Wu and J. Mi. A cauchy loss-incorporated nonnegative latent factorization of tensors model for spatiotemporal traffic data recovery. *Neurocomputing*, 2025.

H. Wu, X. Wu, and X. Luo. *Dynamic Network Representation Based on Latent Factorization of Tensors*. Springer, 2023.

H. Wu, Y. Qiao, and X. Luo. A fine-grained regularization scheme for nonnegative latent factorization of high-dimensional and incomplete tensors. *IEEE Transactions on Services Computing*, 2024. DOI: 10.1109/TSC.2024.3486171.

L. H. Xie, C. Huang, Y. Xu, J. S. Zhao, D. W. Yin, and J. Huang. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 70–79, 2022a.

L. H. Xie, C. Huang, and C. Zhang. Self-supervised hypergraph transformer for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2100–2109, 2022b.

F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang. Outlier-resilient web service qos prediction. In *Proceedings of The Web Conference 2021*, pp. 3099–3110, 2021.

Y. Yu, G. Zhou, N. Zheng, Y. Qiu, S. Xie, and Q. Zhao. Graph-regularized nonnegative tensor-ring decomposition for multiway representation learning. *IEEE Transactions on Cybernetics*, 53(5): 3114–3127, 2023.

C. Zhang et al. Scale-driven tensor representation-based multi-view clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 2025. DOI: 10.1109/TNNLS.2025.3558613.

H. Zhao and S. Ram. Constrained cascade generalization of decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):727–739, 2004.

A Extended Related Work

To date, a variety of higher-order graph representation learning models have been proposed. These models can be broadly categorized into three main types: graph neural network-based models, random walk-based models, and tensor decomposition-based models.

Graph neural network (GNN)-based models extend traditional GNNs to capture high-order structures through various architectural innovations. For example, a general hypergraph neural network (Gao et al., 2023) generalizes convolution operations to hypergraphs, enabling effective feature propagation across hyperedges. Directional Sheaf Hypergraph Networks (DSHN) (Mule et al., 2026) integrate sheaf theory with directed hypergraphs to model asymmetric higher-order interactions while unifying learning on directed and undirected hypergraphs. Furthermore, a UniGNN framework (Huang & Yang, 2023) proposes a generalized message passing scheme that unifies the modeling of pairwise and higher-order relations, effectively bridging the gap between standard graph neural networks and hypergraph architectures. More recently, HYPER (Huang et al., 2026b) develops a foundation model for inductive link prediction on knowledge hypergraphs, supporting generalization to unseen entities and unseen relation types of varying arities. In summary, recent graph and hypergraph representation models have demonstrated strong capabilities for modeling higher-order interactions, although training efficiency and generalization remain important challenges for large-scale and sparse higher-order graphs.

Random walk-based models learn graph representations by sampling traversal paths that expose structural dependencies among nodes and relations. Flock (Kim et al., 2026) extends this paradigm to knowledge-graph foundation models by iteratively sampling random walks, encoding them as sequences, and aggregating the resulting node and relation representations for transferable link-level reasoning. This recent formulation illustrates how random-walk information can be combined with sequence modeling to capture rich graph structure; however, such approaches still rely on explicit path sampling and traversal, which may introduce additional computational cost on large-scale graphs.

Tensor decomposition-based models aim to capture multidimensional interactions among nodes, relations, or temporal patterns by factorizing high-order tensors constructed from graph data. A RESCAL tensor decomposition model (Al-Obeidat et al., 2022) provides an efficient parallel implementation scheme for processing large scale relational data. A Tucker decomposition model (Jang & Kang, 2023) significantly improves the efficiency of mining hidden relationships from dense tensors by random SVD and optimizing the calculation order. A CP tensor decomposition model (Huang et al., 2026a) learns latent representations of temporal networks by factorizing third-order network tensors into node and temporal factors. Yu et al. (2023) propose a non-negative tensor-ring decomposition model, which adopts the internal geometric structure of the data by incorporating graph regularization for multiway representation learning. In general, tensor decomposition-based models are widely used for learning low-dimensional representations of target graphs, their single-layer structural limitations make it challenging to achieve accurate representations for nonstandard tensors with extremely low data density.

B Algorithm Design and Computational Complexity

The complete workflow of MLFN is illustrated in Figure 12, followed by the detailed algorithm design and computational-complexity analysis.

Based on the above analyses, we design Algorithm MLFN in Algorithm 1. Note that it usually has $|\Lambda| \gg \max\{|I|, |J|, |K|\}$ for a tensor. Note that the training update process of MLFN includes three procedures, that is, Update $\mathbf{U}_n$, Update $\mathbf{S}_n$, and Update $\mathbf{T}_n$ for training LFs. Since their principle are basically to fix the other two matrices to update, we only give the procedure Update $\mathbf{U}_n$ as in Algorithm 2. If the model has only one layer, it is actually degraded to the original LFT model.

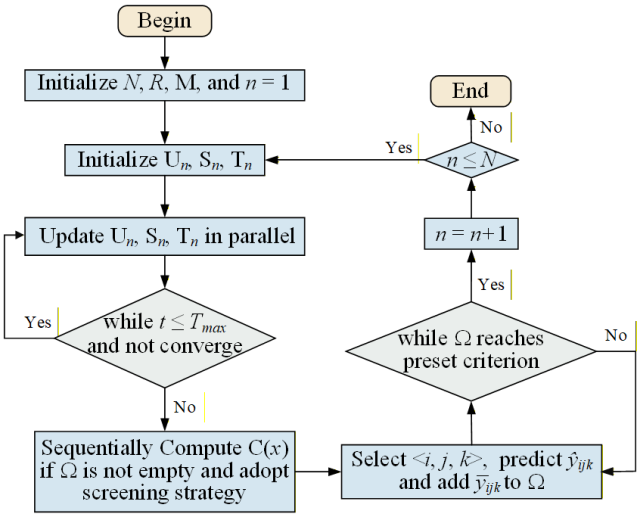


Figure 12: Flowchart of MLFN.

The computational complexity of MLFN is divided into two parts: time complexity and space complexity. Firstly, the time complexity is as follows.

$$\begin{aligned} C_{\text{time}} &= \Theta((|I|+|J|+|K|)R) \\ &\quad + \Theta\left(\frac{1}{M}\left[T_{\max}R|\Lambda| + \cdots + T_{\max}R|\Lambda|(1+\beta)^{n-1} + \cdots + T_{\max}R|\Lambda|(1+\beta)^{N-1}\right]\right) \\ &\approx \Theta\left(\frac{T_{\max}R|\Lambda|(1+\beta)^{N}}{\beta M}\right). \end{aligned} \tag{15}$$

Note that it usually has $|\Lambda| \gg \max\{|I|, |J|, |K|\}$ for a tensor. Where β is the generation rate of the synthetic data for each layer, that is, $\beta = |\text{synthetic data}|/(|\Lambda| + |\Omega|)$. MLFN model employs ASGD to train in parallel, and M threads are set up in the experiment.

Secondly, the space complexity of MLFN is as follows.

$$\begin{aligned} C_{\text{space}} &= \Theta(|I|R) + \Theta(|J|R) + \Theta(|K|R) \\ &\quad + \Theta\left(|\Lambda| + \cdots + |\Lambda|(1+\beta)^{n-1} + \cdots + |\Lambda|(1+\beta)^{N-1}\right) \\ &\approx \Theta\left(R\max\left\{|I|, |J|, |K|, \frac{|\Lambda|(1+\beta)^{N}}{\beta R}\right\}\right). \end{aligned} \tag{16}$$

The formula considers the following factors: 1) it caches $\mathbf{U}$, $\mathbf{S}$, $\mathbf{T}$ and Λ; 2) it caches input data with $|\Lambda|(1+\beta)^{n-1}$ for training. In summary, the MLFN model exhibits linear scalability in both time and space, making it well-suited for nonstandard tensors.

ALGORITHM 1: MLFN

	Input: $\Lambda, I, J, K, N, M, R, \lambda, \eta, \alpha$	**Cost**
1	Initialize $\Omega = \emptyset$, $T_{\max}$, M numbers of threads	$\Theta(1)$
2	**for** $n = 1$ to N	$\times N$
3	Initialize $\mathbf{U}_n, \mathbf{S}_n, \mathbf{T}_n$ randomly with non-negative random values in $[0, 0.05]$	$\Theta((\|I\| + \|J\| + \|K\|)R)$
4	Initialize M number of threads for parallel	$\Theta(M)$
5	**while** $t \leq T_{\max}$ and not converge	$\times T_{\max}$
6	Train $\mathbf{U}_n$ with fixed $\mathbf{S}_n$ and $\mathbf{T}_n$ in parallel	$\Theta(\text{Update } U_n)$
7	Train $\mathbf{S}_n$ with fixed $\mathbf{U}_n$ and $\mathbf{T}_n$ in parallel	$\Theta(\text{Update } S_n)$
8	Train $\mathbf{T}_n$ with fixed $\mathbf{S}_n$ and $\mathbf{U}_n$ in parallel	$\Theta(\text{Update } T_n)$
9	$t = t + 1$	$\Theta(1)$
10	**end while**	–
11	**while** traverse all of Ω	$\Theta(\|\Omega\|)$
12	Sequentially compute $C(x)$	$\Theta(1)$
13	Remove x if $C(x) < \delta$	$\Theta(1)$
14	**end while**	–
15	**while** generated synthetic data do not reach the pre-set criterion	$\Theta(\|\Lambda\|(1+\beta)^{n-1})$
16	Select an unobserved index from $\bar{\mathbf{Y}}_{n-1}$, obtain its predicted value $\hat{y}_{ijk}$ from $\hat{\mathbf{Y}}_n$, and input $\hat{y}_{ijk}$ into the activation function to obtain $\bar{y}_{ijk}$	–
17	**end while**	–
18	**end for**	–

ALGORITHM 2: PROCEDURE UPDATE $\mathbf{U}_n$

	Input: $\Lambda, \Omega, R, M, \mathbf{U}_n, \mathbf{S}_n, \mathbf{T}_n, \lambda, \eta, \alpha$	**Cost**
1	Separate Λ and Ω of $\bar{\mathbf{Y}}_{n-1}$ into M blocks by the I-th dimension	–
2	**for** $m = 1$ to M	$\times M$
3	Dispatch a thread to compute	–
4	**for each** $\langle i, j, k \rangle$ in Λ or Ω	$\times(\|\Lambda\| + \|\Omega\|)$
5	**for** $r = 1$ to R	$\times R$
6	Update U_{ir} according to Equation 12 for Λ and Equation 13 for Ω	$\Theta(1)$
7	**end for**	–
8	**end for**	–
9	Close the dispatched thread	–
10	**end for**	–

C ADDITIONAL EXPERIMENTAL RESULTS

This appendix provides dataset details, evaluation definitions, statistical analysis, and additional experimental results that complement the main paper.

C.1 DATASET CONSTRUCTION AND STATISTICS

Following the organization used in the main experiments, four real-world higher-order graphs are modeled as third-order "Node-Node-Relation" nonstandard tensors.

D1 (Cora; `https://people.cs.umass.edu/~mccallum/data.html`). This citation network contains 17,411 authors (A), 17,934 papers (P), and 12,313 terms (T). An "Author-Author-Relation" tensor is constructed from the APA and APTPA relation meta-paths.

D2 (Aminer; `https://www.aminer.cn/data`). This academic network contains 8,052 authors (A), 20,201 papers (P), and 4 conferences (C). An "Author-Author-Relation" tensor is constructed from the APA and APCPA relation meta-paths.

D3 (Douban Book; `http://book.douban.com`). This review network contains 13,024 users (U) in 2,936 groups (G), 22,347 books (B), and 10,805 authors (A). A "User-User-Relation" tensor is constructed from the UBU, UGU, and UBABU relation meta-paths.

D4 (Yelp; `https://www.yelp.com/dataset/challenge`). This review network contains 1,286 users (U) rating 2,614 businesses (B), associated with 2 services (S), 3 categories (C), 2 reservations (R), and 9-star levels (L). A "User-User-Relation" tensor is constructed from the UBU, UBRBU, UBCBU, UBSBU, and UBSLBU relation meta-paths.

Detailed tensor statistics are summarized in Table 1.

Table 1: Dataset details.

Dataset	Nodes	Relations	Known data	Data density
D1	17411	2	51530	0.0085%
D2	8052	2	36620	0.0282%
D3	13024	3	50884	0.0099%
D4	1286	5	3531	0.0427%

C.2 Evaluation Metrics and Statistical Analysis

The definitions of RMSE, MAE, and MAPE used in the main paper are

$$\begin{aligned} \mathrm{RMSE} &= \sqrt{\frac{\sum_{y_{ijk}\in\Psi}(y_{ijk}-\hat{y}_{ijk})^2}{|\Psi|}}, \\ \mathrm{MAE} &= \frac{\sum_{y_{ijk}\in\Psi}|y_{ijk}-\hat{y}_{ijk}|}{|\Psi|}, \\ \mathrm{MAPE} &= \frac{1}{|\Psi|}\sum_{y_{ijk}\in\Psi}\frac{|y_{ijk}-\hat{y}_{ijk}|}{y_{ijk}}. \end{aligned} \tag{17}$$

where Ψ denotes the testing set. Lower RMSE, MAE, and MAPE indicate higher prediction accuracy. Detailed Wilcoxon Signed-Ranks Test results are reported in Table 2.

Table 2: Wilcoxon Signed-Ranks Test results.

Comparison	R^+	R^-	p-value*
M2 vs M1	36	0	0.0039
M3 vs M1	36	0	0.0039
M4 vs M1	36	0	0.0039
M5 vs M1	36	0	0.0039
M6 vs M1	36	0	0.0039
M7 vs M1	36	0	0.0039
M8 vs M1	36	0	0.0039

*$p < 0.05$ indicates significant difference.

C.3 Complete Comparison Results

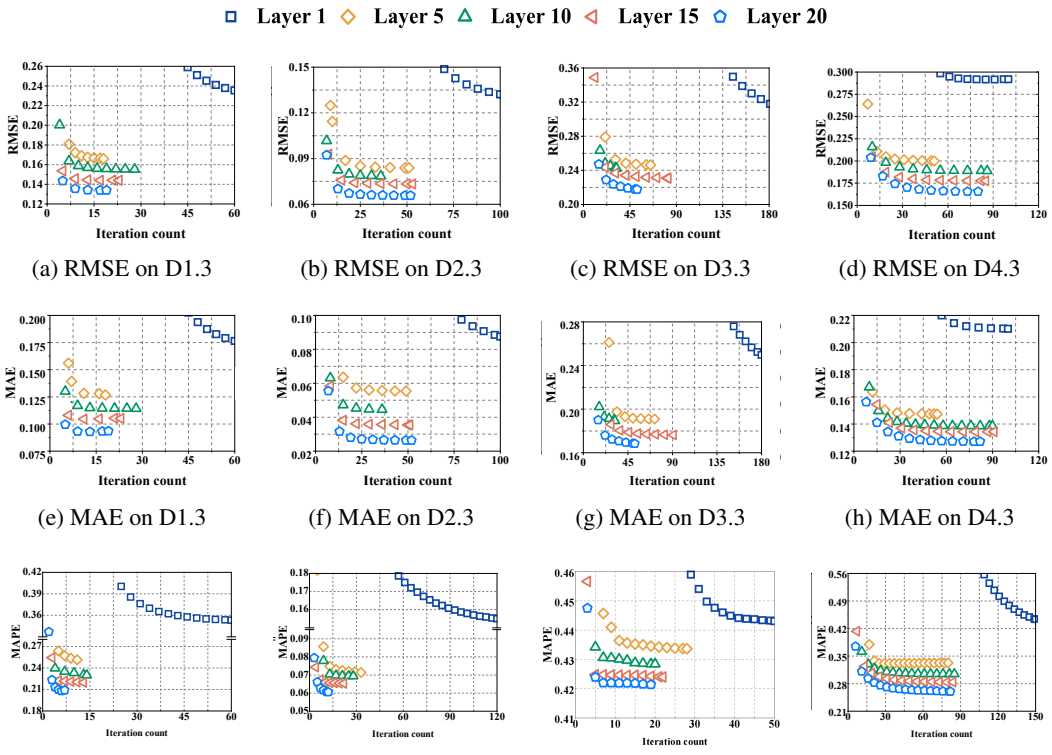
Table 3 reports the complete numerical comparison corresponding to Figure 8 in the main paper, including the Win/Loss counts and Friedman ranks.

Table 3: The comparison results on prediction accuracy. M1 values are bolded according to the revision comment.

Dataset	Case	Metric	M1	M2	M3	M4	M5	M6	M7	M8
D1	D1.1	RMSE	**0.1592±2.4E-4**	0.1719±2.9E-4	0.1783±7.4E-4	0.2549±1.4E-4	0.1653±3.1E-3	0.1730±3.6E-4	0.1777±6.3E-4	0.1669±1.9E-5
		MAE	**0.1163±5.3E-4**	0.1206±7.9E-5	0.1288±7.6E-4	0.2119±9.0E-5	0.1267±2.9E-3	0.1349±2.1E-3	0.1374±6.2E-4	0.1344±4.3E-6
		MAPE	**0.2287±5.0E-3**	0.2941±1.2E-3	0.2592±3.4E-3	0.4575±7.3E-4	0.2311±6.4E-3	0.2531±2.1E-4	0.2538±6.2E-4	0.2519±4.6E-5
	D1.2	RMSE	**0.1457±9.4E-4**	0.1614±3.2E-4	0.1693±4.2E-3	0.2084±1.7E-4	0.1575±7.9E-4	0.1703±1.1E-3	0.1718±5.0E-4	0.1639±5.0E-6
		MAE	**0.1072±1.1E-3**	0.1150±4.5E-4	0.1255±4.5E-3	0.1662±1.7E-4	0.1107±2.8E-4	0.1330±7.5E-4	0.1351±4.7E-4	0.1330±1.5E-7
		MAPE	**0.2157±1.8E-4**	0.2546±1.0E-3	0.2475±7.5E-4	0.3612±5.5E-4	0.2205±1.9E-4	0.2523±7.3E-4	0.2517±4.5E-4	0.2492±1.4E-5
	D1.3	RMSE	**0.1439±3.2E-4**	0.1607±2.2E-5	0.1612±1.4E-3	0.1834±2.2E-4	0.1572±1.4E-3	0.1612±1.3E-4	0.1677±1.7E-3	0.1595±1.2E-7
		MAE	**0.1039±4.2E-4**	0.1167±2.4E-4	0.1186±1.4E-3	0.1425±8.3E-5	0.1106±1.3E-3	0.1227±1.2E-4	0.1323±4.2E-4	0.1285±5.3E-8
		MAPE	**0.2059±4.4E-3**	0.2224±1.3E-3	0.2417±2.3E-4	0.3158±1.3E-4	0.2092±2.7E-3	0.2501±2.3E-4	0.2506±3.1E-4	0.2485±2.8E-5
D2	D2.1	RMSE	**0.0968±9.7E-4**	0.1224±1.3E-5	0.1220±4.1E-3	0.2284±2.7E-4	0.1093±1.3E-3	0.1175±5.0E-4	0.1195±3.5E-4	0.1331±1.9E-5
		MAE	**0.0634±8.9E-4**	0.0890±1.4E-4	0.0899±4.4E-3	0.1810±2.1E-4	0.0776±1.6E-3	0.0832±7.7E-4	0.0813±2.9E-4	0.1205±6.6E-6
		MAPE	**0.1280±3.9E-3**	0.2433±5.9E-3	0.1820±2.9E-3	0.3985±3.7E-4	0.1453±1.8E-3	0.1321±2.1E-3	0.1374±8.2E-4	0.1309±3.4E-5
	D2.2	RMSE	**0.0949±4.9E-4**	0.0995±3.2E-4	0.1078±1.9E-3	0.1561±1.6E-4	0.1016±2.3E-3	0.1047±1.4E-3	0.1148±5.2E-4	0.1312±8.9E-6
		MAE	**0.0617±1.2E-3**	0.0681±2.9E-4	0.0782±3.3E-3	0.1059±1.4E-4	0.0701±2.1E-3	0.0726±7.6E-4	0.0803±5.2E-4	0.1187±1.4E-5
		MAPE	**0.0865±2.3E-3**	0.1481±1.3E-3	0.1662±7.0E-5	0.2437±1.6E-4	0.1339±2.2E-3	0.1050±3.5E-3	0.1031±5.6E-4	0.1004±2.5E-5
	D2.3	RMSE	**0.0927±2.4E-3**	0.0958±4.3E-4	0.0982±1.6E-3	0.1160±1.6E-4	0.0963±1.1E-3	0.1042±7.3E-4	0.0959±7.1E-4	0.1306±2.4E-7
		MAE	**0.0568±1.4E-3**	0.0593±2.3E-4	0.0661±9.9E-4	0.0682±3.2E-4	0.0657±4.8E-4	0.0715±3.1E-4	0.0623±1.4E-3	0.1179±1.1E-7
		MAPE	**0.0715±9.8E-4**	0.1481±1.3E-3	0.1662±7.0E-5	0.2437±1.6E-4	0.1339±2.2E-3	0.1033±2.1E-4	0.0918±3.3E-4	0.1004±2.5E-5
D3	D3.1	RMSE	**0.2509±3.9E-5**	0.2851±6.0E-4	0.2763±1.9E-3	0.3387±1.5E-4	0.2511±5.4E-4	0.2774±1.9E-3	0.2756±3.9E-6	0.2566±1.2E-8
		MAE	**0.1944±2.0E-4**	0.2140±5.0E-4	0.2090±4.0E-3	0.2623±9.8E-5	0.1953±1.4E-3	0.2306±1.2E-4	0.2145±6.3E-5	0.2128±3.8E-8
		MAPE	**0.4379±1.3E-3**	0.4556±6.2E-4	0.4467±6.8E-3	0.5618±4.8E-4	0.4504±1.4E-3	0.6308±4.5E-3	0.6303±6.8E-3	0.4454±2.8E-4
	D3.2	RMSE	**0.2399±6.1E-4**	0.2665±4.2E-4	0.2597±3.7E-3	0.2901±6.9E-5	0.2416±2.3E-3	0.2680±7.1E-4	0.2572±1.4E-3	0.2468±1.1E-5
		MAE	**0.1850±3.5E-4**	0.2004±3.1E-5	0.1958±1.2E-3	0.2191±1.3E-4	0.1854±5.4E-4	0.2252±3.1E-4	0.2184±7.4E-4	0.2126±9.9E-6
		MAPE	**0.4117±3.6E-5**	0.4335±7.8E-3	0.4152±6.1E-3	0.4899±5.4E-4	0.4474±3.4E-3	0.6296±2.6E-3	0.6294±1.2E-3	0.4349±5.4E-5
	D3.3	RMSE	**0.2291±7.2E-4**	0.2641±1.1E-4	0.2497±7.3E-3	0.2600±7.3E-4	0.2314±2.8E-3	0.2610±1.6E-3	0.2612±3.4E-4	0.2464±2.2E-7
		MAE	**0.1774±1.2E-3**	0.1980±4.5E-5	0.2002±6.2E-4	0.1952±3.3E-4	0.1784±1.9E-3	0.2199±9.2E-4	0.2181±8.3E-4	0.2119±8.1E-7
		MAPE	**0.3899±8.4E-3**	0.4091±4.6E-3	0.3938±4.6E-3	0.4433±4.6E-4	0.4088±5.6E-3	0.6235±2.3E-4	0.6131±5.3E-3	0.4334±1.3E-5
D4	D4.1	RMSE	**0.2061±5.1E-5**	0.3391±1.6E-4	0.2270±1.5E-3	0.3724±8.1E-4	0.2064±3.7E-3	0.2171±8.7E-4	0.2127±1.7E-4	0.2130±2.4E-7
		MAE	**0.1539±2.9E-4**	0.2740±3.0E-4	0.1656±3.7E-3	0.3084±7.9E-4	0.1542±2.9E-3	0.1822±1.0E-3	0.1804±5.8E-4	0.1791±2.1E-6
		MAPE	**0.3369±7.1E-4**	0.5991±5.1E-3	0.3453±9.3E-3	0.6723±9.9E-4	0.3511±1.6E-3	0.4647±2.7E-4	0.4679±9.3E-4	0.3678±3.4E-5
	D4.2	RMSE	**0.1936±4.9E-4**	0.2761±3.8E-4	0.2064±2.3E-3	0.2989±7.0E-5	0.1941±2.3E-3	0.2110±1.6E-5	0.2091±2.5E-4	0.2111±2.1E-4
		MAE	**0.1446±8.4E-4**	0.2093±1.7E-4	0.1498±2.0E-3	0.2319±2.5E-4	0.1449±1.6E-3	0.1796±7.6E-4	0.1781±2.1E-5	0.1788±9.0E-5
		MAPE	**0.3091±6.0E-4**	0.4719±2.8E-3	0.3288±7.9E-3	0.5179±1.4E-3	0.3151±6.4E-4	0.4602±4.5E-4	0.4665±5.7E-4	0.3605±2.6E-5
	D4.3	RMSE	**0.1769±2.3E-3**	0.2538±2.1E-3	0.1925±1.4E-3	0.2662±1.2E-3	0.1795±2.4E-3	0.2098±1.8E-4	0.2107±5.2E-4	0.2086±4.9E-5
		MAE	**0.1272±3.0E-3**	0.1831±1.1E-3	0.1402±2.8E-3	0.1902±1.1E-3	0.1301±7.6E-4	0.1785±4.2E-4	0.1785±4.6E-4	0.1780±5.2E-5
		MAPE	**0.2885±4.0E-3**	0.3962±6.3E-4	0.2989±8.1E-3	0.4195±9.8E-4	0.2945±6.8E-3	0.4552±2.2E-4	0.4566±6.4E-4	0.3575±5.8E-5
Win/Loss			–	36/0	36/0	36/0	36/0	36/0	36/0	36/0
F-rank*			1	5.03	4.25	7.42	2.64	5.61	5.42	4.56

C.4 Additional Multilayer-Structure Results

Figure 13 reports the complete multilayer-training results on D1.3–D4.3, complementing the representative D1.3 results in Figure 9 of the main paper.



(a) RMSE on D1.3 (b) RMSE on D2.3 (c) RMSE on D3.3 (d) RMSE on D4.3

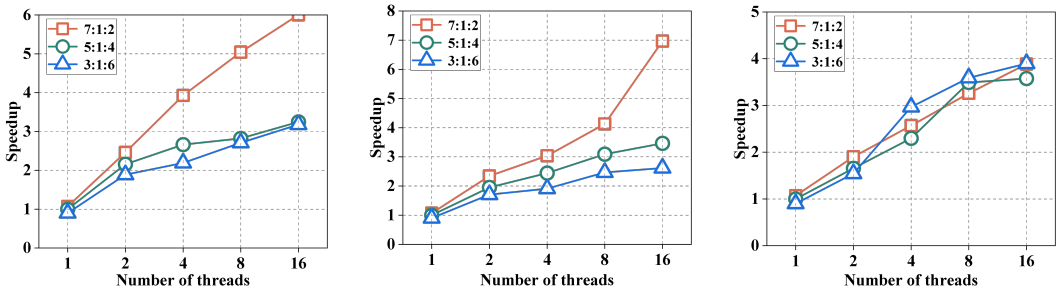
(e) MAE on D1.3 (f) MAE on D2.3 (g) MAE on D3.3 (h) MAE on D4.3

(i) MAPE on D1.3 (j) MAPE on D2.3 (k) MAPE on D3.3 (l) MAPE on D4.3

Figure 13: The training process of MLFN regarding RMSE, MAE and MAPE at different layers on all the datasets.

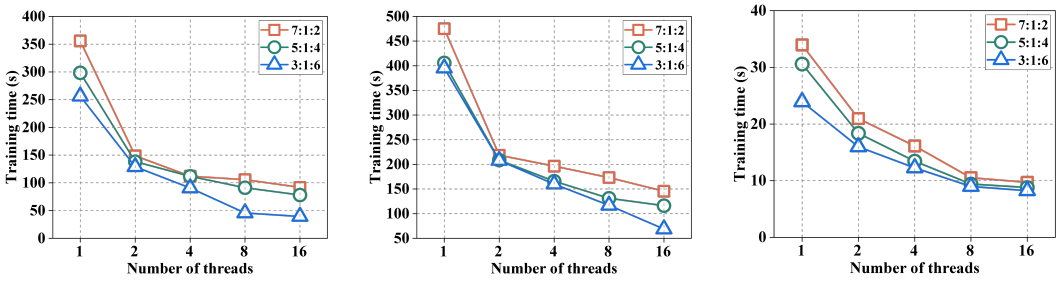
C.5 Additional Parallel-Training Results

Figures 14 and 15 report the speedup and training-time results on D2–D4, complementing the D1 results in Figure 11 of the main paper.



(a) D2 (b) D3 (c) D4

Figure 14: Additional speedup results of MLFN on D2–D4 as thread count increases.



(a) D2 (b) D3 (c) D4

Figure 15: Additional training-time results of MLFN on D2–D4 as thread count increases.

C.6 Additional Ablation Results

Figure 16 collects the remaining ablation panels, including the second and third D1–D2 comparison panels moved from the main paper and the additional results on D3–D4.

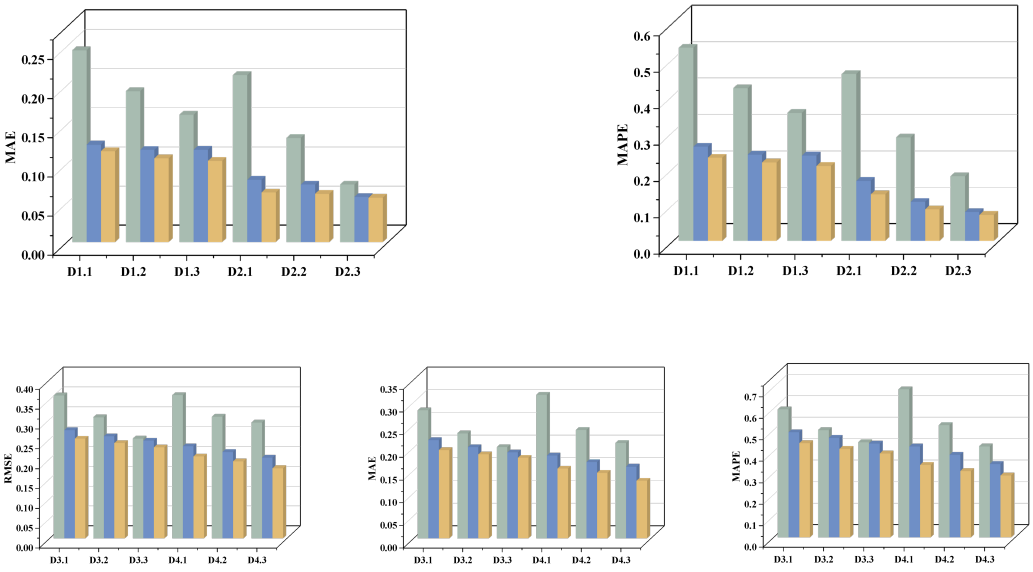
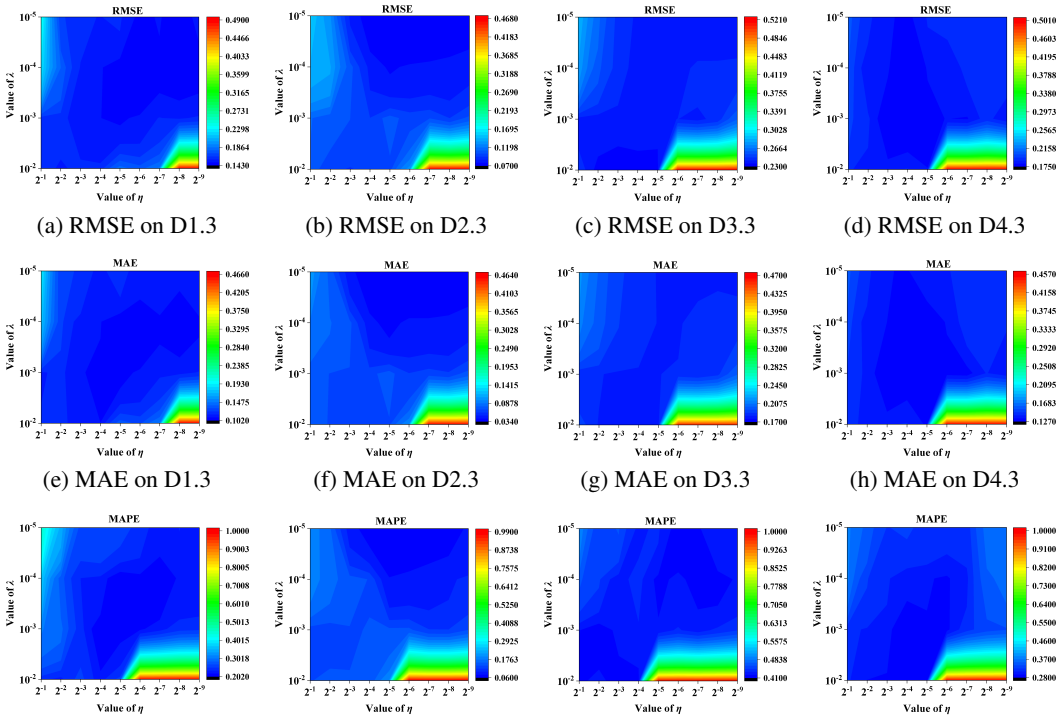


Figure 16: Additional performance comparison of MLFN and its variants, including the remaining D1–D2 panels and results on D3–D4.

C.7 HYPER-PARAMETERS SENSITIVITY TEST

As shown in Section 3, MLFN is affected by the controlling parameter α, learning rate η, and regularization coefficient λ. We conduct parameter-sensitivity tests under the 7:1:2 data split and perform a grid search with $\eta \in [2^{-9}, 2^{-1}]$ and $\lambda \in [10^{-5}, 10^{-2}]$.



(a) RMSE on D1.3 (b) RMSE on D2.3 (c) RMSE on D3.3 (d) RMSE on D4.3

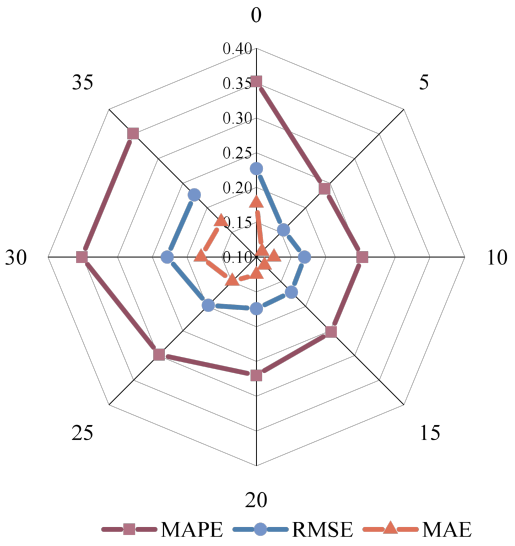
(e) MAE on D1.3 (f) MAE on D2.3 (g) MAE on D3.3 (h) MAE on D4.3

(i) MAPE on D1.3 (j) MAPE on D2.3 (k) MAPE on D3.3 (l) MAPE on D4.3

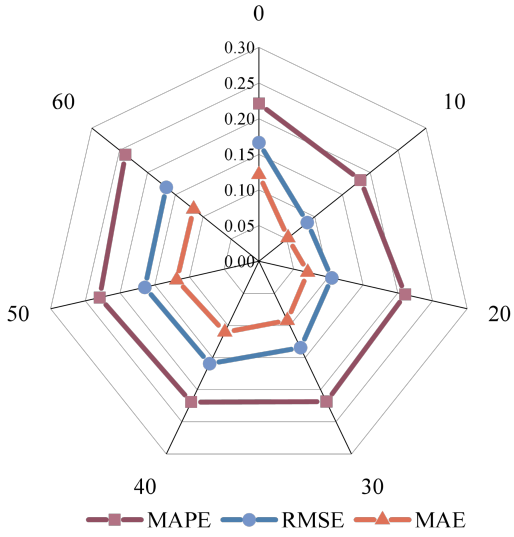
Figure 17: Performance of MLFN as η and λ vary on D1.3–D4.3.

Impact of λ and η. Figure 17 shows that MLFN is sensitive to η and λ. When $\eta = 2^{-7}$ and $\lambda = 10^{-5}$, RMSE reaches its lowest value of 0.2362, compared with the highest value of 0.5209 at $\eta = 2^{-9}$ and $\lambda = 10^{-2}$, a 54.65% difference. In Figure 17(e), MAE ranges from 0.1775 at $\eta = 2^{-6}$ and $\lambda = 10^{-2}$ to 0.4605 at $\eta = 2^{-9}$ and $\lambda = 10^{-2}$. In Figure 17(i), MAPE reaches its lowest value of 0.2032 at $\eta = 2^{-6}$ and $\lambda = 10^{-4}$, while the highest value is 0.9984 at $\eta = 2^{-9}$ and $\lambda = 10^{-2}$. The same sensitivity is observed across the other datasets.

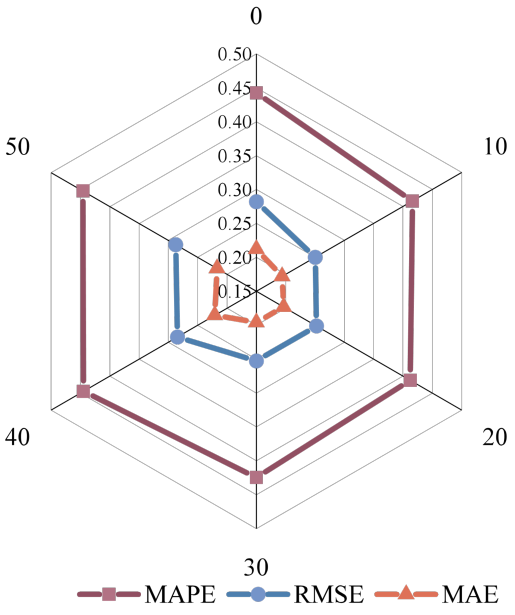
Impact of α. The parameter α controls the contribution of generated data and achieves optimal accuracy at an intermediate value. Following the multilayer principle that outputs from one layer enrich the input to the next (Zhao & Ram, 2004; Pang et al., 2018), we evaluate RMSE, MAE, and MAPE as α increases. Figure 18 shows that the metrics initially decrease, reach their best range, and then gradually increase after α exceeds a threshold. For example, on D3.3, RMSE decreases from 0.2821 to 0.2503 as α increases from 0 to 10, after which it fluctuates and slowly increases. This confirms that generated synthetic data can improve MLFN's representation learning when appropriately weighted.



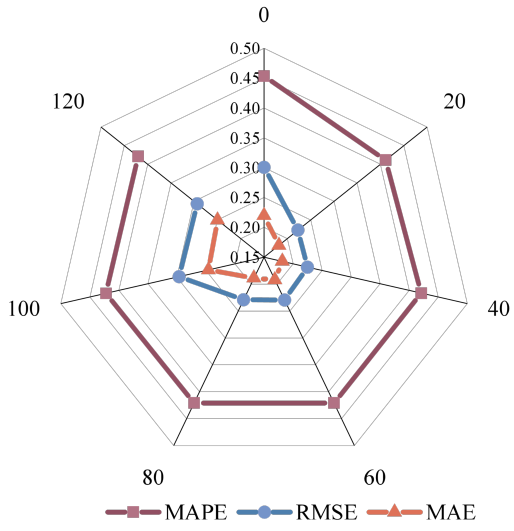
(a) D1.3



(b) D2.3



(c) D3.3



(d) D4.3

Figure 18: Changes of RMSE/MAE/MAPE of MLFN as α increases on the four datasets.

D DETAILED CONVERGENCE PROOF OF MLFN

This appendix presents the convergence proof of the proposed MLFN model from the supplied supplementary material.

Proof of Theorem 1. From Section 3.2, we see that each layer of MLFN is individual in effect. Therefore, if it is shown that the general n-th layer of MLFN is convergent, then the MLFN model is convergent. For this purpose, we analyze the convergence of objective function (7). Since (7) is the sum of the instant loss and nonconvex, (7) converges due to some relaxations.

We consider the instant loss ε_{ijk}^n instead of the sum of loss $\varepsilon(\mathbf{U}_n, \mathbf{S}_n, \mathbf{T}_n)$, because we take the form of a single-LF-dependent for each training, which corresponds to a single element $\forall\langle i,j,k\rangle \in \Lambda$ or Ω. A portion of the non-convex term is fixed so that the instant loss ε_{ijk}^n is convex, i.e., $\mathbf{s}_j^n$ and $\mathbf{t}_k^n$ is used as a constant to express the convergence of training with $\mathbf{u}_i^n$. Note that the training rules are symmetrical for $\mathbf{u}_i^n$, $\mathbf{s}_j^n$, $\mathbf{t}_k^n$. Therefore, convergence can be achieved in the same way through training $\mathbf{s}_j^n$ and $\mathbf{t}_k^n$. Then, we define the L-smooth and strong convex function $f(\xi)$.

Definition 1 (L-smooth Function)**.** *$f(\xi)$ is L-smooth if*

$$\forall \xi_1, \xi_2 \in \mathbb{R}^d \text{ s.t. } \|\nabla f(\xi_1) - \nabla f(\xi_2)\|_2 \le L\|\xi_1 - \xi_2\|_2. \tag{S1}$$

Definition 2 (Strong Convex $f(\xi)$)**.** *$f(\xi)$ is strong convex if there exists a constant $\delta > 0$ satisfying*

$$\forall \xi_1, \xi_2 \in \mathbb{R}^d \text{ s.t. } f(\xi_1) \ge f(\xi_2) + \nabla f(\xi_2)(\xi_1 - \xi_2)^T + \frac{1}{2}\delta\|\xi_1 - \xi_2\|_2^2. \tag{S2}$$

Lemma 1. *The instant loss ε_{ijk}^n is L-smooth when L is the maximum singular value for matrix $\left((\mathbf{s}_j^n)^T\mathbf{s}_j^n(\mathbf{t}_k^n)^T\mathbf{t}_k^n + \lambda\mathbf{E}_d\right)$ and $\mathbf{E}_d$ is $R \times R$ identity matrix.*

Proof. Assuming that $\mathbf{u}_{\sigma,\cdot}$ and $\mathbf{u}_{\varphi,\cdot}$ are two arbitrary and independent row vectors of $\mathbf{U}_n$, we conclude (S3):

$$\begin{cases} \langle i,j,k\rangle \in \Lambda : \\ \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) \\ \quad = (\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\left((\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d\right), \\ \langle i,j,k\rangle \in \Omega : \\ \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) \\ \quad = (\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\left(\alpha(\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d\right). \end{cases} \tag{S3}$$

In order to simply (S3) for convergence analysis, we can make $\alpha = 1$. Thus, we achieved that

$$\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})\|_2 = \|(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\mathbf{H}\|_2, \tag{S4}$$

According to the L_2-norm properties of a matrix, we have

$$\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})\|_2 \le \left\|(\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d\right\|_2 \|\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot}\|_2. \tag{S5}$$

where $\left\|(\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d\right\|_2$ represents the largest singular value of the same matrix. Based on the above inferences, we conclude that this norm is L. Hence, Lemma 1 holds. □

Lemma 2. *The instant loss ε_{ijk}^n is of strong convexity when δ is the minimum singular value for matrix $\left((\mathbf{s}_j^n)^T\mathbf{s}_j^n(\mathbf{t}_k^n)^T\mathbf{t}_k^n + \lambda\mathbf{E}_d\right)$.*

Proof. Given arbitrary vectors $\mathbf{u}_{\sigma,\cdot}$ and $\mathbf{u}_{\varphi,\cdot}$, we expand the state of ε_{ijk}^n at $\mathbf{u}_{\varphi,\cdot}$. Follow the principle of Taylor series, as shown below:

$$\begin{aligned} \varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) &\approx \varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) + \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T \\ &\quad + \frac{1}{2}(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\nabla^2\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T, \\ \Rightarrow \quad \varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) &= \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T \\ &\quad + \frac{1}{2}(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\nabla^2\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T. \end{aligned} \tag{S6}$$

According to Definition 2, if ε_{ijk}^n is strong convex, we can achieve that

$$\varepsilon_{ijk}^n(\mathbf{u}_{\sigma,\cdot}) - \varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) \geq \nabla\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T + \frac{1}{2}\delta\|\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot}\|_2^2. \tag{S7}$$

Then, Lemma 2 is equivalent to select δ to make the following inequality true:

$$(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\nabla^2\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot})(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T \geq \delta\|\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot}\|_2^2. \tag{S8}$$

From the expression of ε_{ijk}^n, we can obtain that

$$\nabla^2\varepsilon_{ijk}^n(\mathbf{u}_{\varphi,\cdot}) = (\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d. \tag{S9}$$

Combining (S1) and (S2), we only need to prove that

$$(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\left((\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d\right)(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T \geq \delta\|\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot}\|_2^2. \tag{S10}$$

(S10) is also equivalent to

$$(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})\left((\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d - \delta\mathbf{E}_d\right)(\mathbf{u}_{\sigma,\cdot} - \mathbf{u}_{\varphi,\cdot})^T \geq 0. \tag{S11}$$

By the properties of the matrix, (S11) is equivalent to prove that $((\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d - \delta\mathbf{E}_d)$ is a positive semidefinite matrix. As Zhang reveals, this is a positive semidefinite matrix when δ is the minimum singular value of $((\mathbf{s}_{j,\cdot}^n)^T\mathbf{s}_{j,\cdot}^n(\mathbf{t}_{k,\cdot}^n)^T\mathbf{t}_{k,\cdot}^n + \lambda\mathbf{E}_d)$, which satisfies positive semidefiteness. Hence, Lemma 2 holds. □

Considering the t-th iteration of MLFN, we have the following training rules for $\mathbf{u}_{i,\cdot}^n$ on a single entry $\forall\langle i,j,k\rangle \in \Lambda$ or Ω:

$$\mathbf{u}_{i,\cdot}^n(w) \leftarrow \mathbf{u}_{i,\cdot}^n(w-1) - \eta\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1)). \tag{S12}$$

where $\mathbf{u}_{i,\cdot}^n(w)$ and $\mathbf{u}_{i,\cdot}^n(w-1)$ represent the state of $\mathbf{u}_{i,\cdot}^n$ trained by the w-th and $(w-1)$th entries in the t-th iteration, respectively. If $\mathbf{u}_{i,\cdot}^{n,*}$ is the optimal state of $\mathbf{u}_{i,\cdot}^n$, we have

$$\begin{aligned}\|\mathbf{u}_{i,\cdot}^n(w) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 &= \|\mathbf{u}_{i,\cdot}^n(w-1) - \eta\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1)) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \\ &= \|\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 - 2\eta\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))(\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*})^T \\ &\quad + \eta^2\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))\|_2^2.\end{aligned} \tag{S13}$$

Based on Lemma 2, we obtain that

$$\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^{n,*}) - \varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1)) \geq \nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))(\mathbf{u}_{i,\cdot}^{n,*} - \mathbf{u}_{i,\cdot}^n(w-1))^T + \frac{1}{2}\delta\|\mathbf{u}_{i,\cdot}^{n,*} - \mathbf{u}_{i,\cdot}^n(w-1)\|_2^2. \tag{S14}$$

Since $\mathbf{u}_{i,\cdot}^{n,*}$ is the optimal state of $\mathbf{u}_{i,\cdot}^n$, we have that

$$\begin{cases}\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^{n,*}) = 0, \\ \varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^{n,*}) < \varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1)).\end{cases} \tag{S15}$$

By substituting (S15) into (S14), we see that

$$\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))(\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*})^T \geq \frac{1}{2}\delta\|\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2. \tag{S16}$$

Thus, (S13) is equivalent to

$$\|\mathbf{u}_{i,\cdot}^n(w) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \leq (1-\eta\delta)\|\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 + \eta^2\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))\|_2^2. \tag{S17}$$

We take the expectation of (S17), we have

$$\mathbb{E}\|\mathbf{u}_{i,\cdot}^n(w) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \leq (1-\eta\delta)\mathbb{E}\|\mathbf{u}_{i,\cdot}^n(w-1) - \mathbf{u}_{i,\cdot}^{n,*}\|_2^2 + \eta^2\mathbb{E}\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))\|_2^2. \tag{S18}$$

Assume that there is a positive number z such that

$$\mathbb{E}\|\nabla\varepsilon_{ijk}^n(\mathbf{u}_{i,\cdot}^n(w-1))\|_2^2 \leq z^2. \tag{S19}$$

Thus, (S19) is equivalent to

$$\mathbb{E}\|\mathbf{u}_{i,\cdot}^{n}(w)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \leq (1-\eta\delta)\mathbb{E}\|\mathbf{u}_{i,\cdot}^{n}(w-1)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2+\eta^2 z^2. \tag{S20}$$

We take the learning rate $\eta=\frac{\beta}{\delta t}$ with $\beta>1$, (S20) can be convert to

$$\mathbb{E}\|\mathbf{u}_{i,\cdot}^{n}(w)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \leq \left(1-\frac{\beta}{t}\right)\mathbb{E}\|\mathbf{u}_{i,\cdot}^{n}(w-1)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2+\frac{1}{t^2}\left(\frac{\beta z}{\delta}\right)^2. \tag{S21}$$

By expanding the expression of (S21) by induction, we obtain a bound

$$\mathbb{E}\|\mathbf{u}_{i,\cdot}^{n}(w)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2 \leq \frac{1}{t}\max\left\{\|\mathbf{u}_{i,\cdot}^{n}(1)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2, \frac{\beta^2 z^2}{\delta\beta-1}\right\}. \tag{S22}$$

where $\mathbf{u}_{i,\cdot}^{n}(1)$ denote the initial state of $\mathbf{u}_{i,\cdot}^{n}$ at the t-th iteration. According to Lemma 1, ε_{ijk}^{n} is L-smooth and we can achieve that

$$\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n}(w))-\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n,*}) \leq \frac{L}{2}\|\mathbf{u}_{i,\cdot}^{n}(w)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2. \tag{S23}$$

Then we take the expectation of (S23), we can obtain that

$$\mathbb{E}[\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n}(w))-\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n,*})] \leq \frac{L}{2t}\Theta(\beta). \tag{S24}$$

where we set

$$\Theta(\beta)=\max\left\{\|\mathbf{u}_{i,\cdot}^{n}(1)-\mathbf{u}_{i,\cdot}^{n,*}\|_2^2, \frac{\beta^2 z^2}{\delta\beta-1}\right\}. \tag{S25}$$

Then we expand (S24) on all the known entries of Λ and Ω:

$$\mathbb{E}\left[\sum\nolimits_{\langle i,j,k\rangle\in\Lambda\cup\Omega}\left(\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n}(w))-\varepsilon_{ijk}^{n}(\mathbf{u}_{i,\cdot}^{n,*})\right)\right] \leq (|\Lambda|+|\Omega|)\frac{L}{2t}\Theta(\beta). \tag{S26}$$

where $t\to\infty$, and we have $(|\Lambda|+|\Omega|)\frac{L}{2t}\Theta(\beta)\to 0$. The same situation happens to us when $\mathbf{u}_{i,\cdot}^{n}$ is treated as a constant. Although the learning objective (7) is nonconvex, $\mathbf{u}_{i,\cdot}^{n}$, $\mathbf{s}_{j,\cdot}^{n}$, $\mathbf{t}_{k,\cdot}^{n}$ can be trained alternatively by ASGD. Moreover, SGD requires $\eta\leq\frac{1}{\delta t}$, where δ the minimum singular value of the matrix $((\mathbf{s}_{j,\cdot}^{n})^T\mathbf{s}_{j,\cdot}^{n}(\mathbf{t}_{k,\cdot}^{n})^T\mathbf{t}_{k,\cdot}^{n}+\lambda\mathbf{E}_d)$. Besides, the regularization does not affect the convergence, which means that (7) is convergent. Hence, Theorem 1 holds. □