Neural Additive Tensor Decomposition for Sparse Tensors

Dawon Ahn
University of California, Riverside
Riverside, USA
dahn017@ucr.edu

Uday Singh Saini
University of California, Riverside
Riverside, USA
usain001@ucr.edu

Evangelos E. Papalexakis
University of California, Riverside
Riverside, USA
epapalex@cs.ucr.edu

Ali Payani
Cisco Systems Inc.
San Jose, USA
apayani@cisco.com

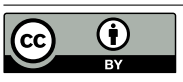
ABSTRACT

Given a sparse tensor, how can we accurately capture complex latent structures inherent in the tensor while maintaining the interpretability of those structures? Tensor decomposition is a fundamental technique for analyzing tensors. Classical tensor models provide multi-linear structures that are easy to interpret, but have limitations in capturing complex structures present in real-world sparse tensors. Recent neural tensor models have extended the capabilities of classical tensor models in capturing complex structures within the data. However, this has come at the cost of interpretability: neural tensor models entangle interactions across and within latent structures in a black-box manner, making it difficult to readily understand the discovered structures. Understanding these structures, however, is crucial in applications such as healthcare, which requires transparency in critical decision-making processes.

To overcome this major limitation and bridge the gap between the classical multi-linear models and neural tensor models, we propose Neural Additive Tensor Decomposition (NeAT), an accurate and interpretable neural tensor model for sparse tensors. The main idea of NeAT is to apply neural networks to each latent component in an additive fashion. This not only captures diverse patterns and complex structures in sparse tensors, but also provides a direct and intuitive interpretation of the structures by being close to the multi-linear tensor model. We conduct extensive experiments on six large real-world sparse tensors. NeAT outperforms the state-of-the-art neural tensor models in link prediction, surpassing a linear tensor model by 10% and the second-best neural tensor model by 4%, in accuracy. Through ablation studies, we explore various model designs for NeAT to identify key factors that impact generalization. Finally, we evaluate qualitatively and quantitatively latent patterns discovered by NeAT, demonstrating how to analyze the discovered latent patterns in real data obtained from NeAT.

CCS CONCEPTS

• **Information systems → Data mining**.



CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10.
https://doi.org/10.1145/3627673.3679833

KEYWORDS

Sparse Tensor Decomposition; Interpretable Neural Tensor Model

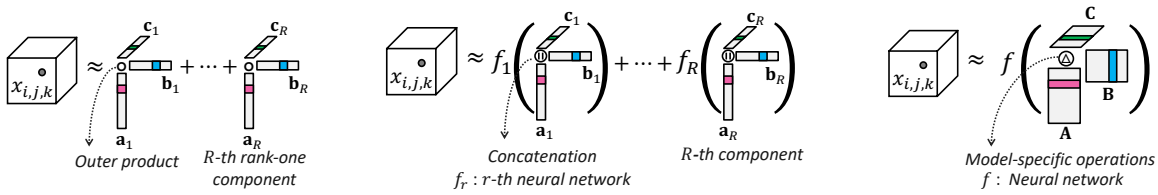
ACM Reference Format:
Dawon Ahn, Uday Singh Saini, Evangelos E. Papalexakis, and Ali Payani. 2024. Neural Additive Tensor Decomposition for Sparse Tensors. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679833

1 INTRODUCTION

A tensor or multi-dimensional array is a natural way to represent higher-order interactions between multi-aspect data. In healthcare domain, an example of such data includes (patients, medical diagnosis, procedure) tuples in Electronic Health Records (EHRs), indicating whether a patient has been diagnosed with a specific medical procedure. Tensor decomposition is a fundamental method for analyzing tensors by extracting latent structures. Canonical Polyadic Decomposition (CPD) and Tucker are classical multi-linear tensor decomposition models. They have been central to a diverse range of applications such as healthcare analysis [2, 15], social networks analysis [4, 27], knowledge base completion [22, 23, 26] and recommendation [7, 40, 41].

CPD [6, 12] specifically has gained popularity due to its simplicity, uniqueness, and interpretability [20, 32, 35]. As depicted in Figure 1(a), CPD reconstructs a tensor as a sum of rank-one components, where each represents unique multi-linear relationships and does not depend on other components. Importantly, this additive nature of the decomposition simplifies understanding of the relationships between latent factors within individual components and identifying the entities that play a crucial role in these components [4, 15, 29]. In the above example, CPD discovers soft co-clustering of (patients, medical diagnosis, procedure) that share similar interaction patterns within each component. The discovered co-clusterings help to improve transparency in clinical decisions and verify the correctness of the extracted patterns.

Even though CPD is preferred for its interpretability, many real-world sparse tensors are better explained via complex non-linear structures, which are often insufficiently represented by the addition of rank-one components. This limitation can lead to lower accuracy of the model in various practical applications. Recent tensor models based on neural networks for sparse tensors have garnered attention, having successfully captured non-linear patterns in sparse tensors [7, 9, 14, 24, 25, 30, 36, 38]. They have shown



(a) CPD (b) NeAT (Proposed) (c) Neural tensor models

Figure 1: Comparison of model designs of CPD, NeAT, and neural tensor models with the third mode tensor. In NeAT, a neural network is applied to each component to achieve interpretability similar to the rank-one component in CPD. In contrast, the neural tensor model applies a neural network to all components simultaneously, which makes all components interact within the neural network.

superior performance in tensor completion tasks compared to the classical tensor models. Even though these approaches enhance the conventional tensor models with the expressive power of neural networks, the way they are designed to use neural networks intermixes components with each other thereby hindering the interpretability of latent components, as illustrated in Figure 1(c).

To address this limitation, we propose Neural Additive Tensor Decomposition (NeAT) to accurately model non-linear latent structures while providing easily interpretable latent components by respecting the separation of distinct components in a similar vein as CPD. As illustrated in Figure 1(b), NeAT learns non-linear structures present in each component by employing individually parameterized neural networks, unlike existing neural tensor models that employ neural networks interacting with all components. Also, NeAT designs the model leveraging the sparsity of tensors to save significant computations by avoiding conventional tensor operations. Our contributions are summarized as follows:

- **Model.** We propose NeAT, a novel neural tensor model that can learn diverse non-linear structures while enhancing their interpretability for sparse tensors.
- **Performance.** Extensive experiments on six large real-world tensors demonstrate that NeAT shows the state-of-the-art performance in accuracy for sparse tensor completion over all baselines and shows its ability to capture meaningful patterns in factors with downstream tasks.
- **Interpretability.** Finally, we show how NeAT can serve as a glass-box model instead of a black-box model and provide meaningful insights into its latent factors.

The rest of this paper is organized as follows. We introduce the preliminaries and related works in Section 2. We propose NeAT in Section 3 and present experimental results in Section 4. We summarize the key points and results of our paper in Section 5. The source code and datasets used in this paper are available at https://github.com/dawonahn/NeAT.

2 PRELIMINARIES & RELATED WORK

We introduce preliminaries of tensor decompositions. We then discuss their interpretability from the perspective of tensor and additive models. We also provide an overview of neural tensor models.

2.1 Tensors

Tensors are defined as multi-dimensional arrays that generalize one-dimensional arrays (or vectors) and two-dimensional arrays (or matrices) to higher dimensions. Sparse tensors indicate tensors where the majority of their entries are missing. Traditionally, the dimension of a tensor is referred to as its order or the number of modes; the size of each mode is called "dimensionality". We use boldface Euler script letters (e.g., $\mathcal{X}$) to denote tensors, boldface capitals (e.g., A) to denote matrices, boldface lower cases (e.g., a) to denote vectors. We denote the ith row vector as $\mathrm{a}_{i,:}$, rth column vector as a_r and (i,r)th entry as a_{ir}. The Frobenius norm of a tensor $\mathcal{X}$ is given by $||\mathcal{X}||_F = \sqrt{\sum_{\alpha\in\Omega} x_\alpha^2}$, where x_α is the $\alpha = (i_1, \cdots, i_N)$th entry of $\mathcal{X}$ and Ω is the set of indices of entries.

2.2 Tensor Decomposition

We explain two classical multi-linear tensor models, CPD and Tucker decomposition. CPD [6, 12] approximates an N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ as the sum of R rank-one components as:

$$\mathcal{X} \approx [\![\mathrm{A}^{(1)}, \ldots, \mathrm{A}^{(N)}]\!] = \sum_{r=1}^{R} \mathrm{a}_r^{(1)} \circ \cdots \circ \mathrm{a}_r^{(N)}, \tag{1}$$

where $\circ$ denotes an outer product and $\mathrm{a}_r^{(n)} \in \mathbb{R}^{I_n}$ is the rth column or component of the nth factor matrix $\mathrm{A}^{(n)} \in \mathbb{R}^{I_n\times R}$ for $n = 1, \cdots, N$. Each rank-one component (e.g., $\mathrm{a}_r^{(1)} \circ \cdots \circ \mathrm{a}_r^{(N)}$) corresponds to the unique latent pattern that simultaneously clusters entities across N modes. Elementwise, Equation (1) is written as

$$x_\alpha \approx \sum_{r=1}^{R} a_{i_1 r}^{(1)} a_{i_2 r}^{(2)} \cdots a_{i_N r}^{(N)}, \tag{2}$$

where x_α indicates the $\alpha = (i_1, \cdots, i_N)$th entry of $\mathcal{X}$ and $a_{i_n r}^{(n)}$ indicates (i_n, r)th element of $\mathrm{A}^{(n)}$.

Tucker Decomposition [37] approximates an N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ as a core tensor $\mathcal{G} \in \mathbb{R}^{R_1\times\cdots\times R_N}$ multiplied by a factor matrix $\mathrm{A}^{(n)} \in \mathbb{R}^{I_n\times R_n}$ along each mode n as follows:

$$\mathcal{X} \approx [\![\mathcal{G}; \mathrm{A}^{(1)}, \ldots, \mathrm{A}^{(N)}]\!] = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1\cdots r_N} \mathrm{a}_r^{(1)} \circ \cdots \circ \mathrm{a}_r^{(N)}, \tag{3}$$

where R_n denotes the number of components in the nth factor matrix $\mathrm{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. Note that the entries of the core tensor $\mathcal{G}$ indicate the level of interactions between different components. Elementwise, Equation (3) is written as

$$x_\alpha \approx \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1 \cdots r_N} a^{(1)}_{i_n r_1} \cdots a^{(N)}_{i_n r_N}. \quad (4)$$

Note that CPD can be written as a constrained instance of the Tucker decomposition where the core tensor is super-diagonal and R_n is identical for all n.

The optimization problem of both tensor models is to find the best factor matrices and a core tensor by minimizing the difference between the given tensor and the reconstructed tensor, which is formulated as:

$$\min_{\mathcal{G},\mathrm{A}^{(1)},\cdots,\mathrm{A}^N} \|\mathcal{X} - \tilde{\mathcal{X}}\|_F \Leftrightarrow \min_{\mathcal{G},\mathrm{A}^{(1)},\cdots,\mathrm{A}^N} \sum_{\alpha \in \Omega} (x_\alpha - \tilde{x}_\alpha)^2, \quad (5)$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\tilde{\mathcal{X}}$ or $\tilde{x}_\alpha$ is the reconstruction using either CPD in Equations (1) and (2) or Tucker in Equations (3) and (4). Note that Ω denotes a set of indices of all entries. Alternating least square (ALS) or gradient descent methods are common approaches to solve this problem [20, 21]. A widely used application of tensor models is tensor completion, where missing entries of a tensor are predicted using factor matrices trained with observed entries, also known as link prediction or missing entry prediction [8, 10, 16, 17, 28, 34].

2.3 Interpretability

2.3.1 Tensor Models. In the realm of tensor decomposition and in the context of this paper, interpretability refers to discovering hidden patterns in latent components related to original data more readily. To understand the hidden patterns, we examine the most influential entities in the components, where higher magnitude values indicate greater impact. In CPD, each latent component corresponds directly to a unique hidden pattern and the contribution of entities to each hidden pattern is easy to identify. On the other hand, Tucker decomposition considers interactions between components due to the core tensor, which complicates the interpretation. Inspired by the additive nature of CPD, we design the proposed model to achieve interpretability. To further improve the interpretability of latent components other than the model design, various constraints such as non-negativity and sparsity can be applied to the components [2, 15].

2.3.2 Additive Models. Generalized Additive Models (GAMs) [13] model non-linear relationships between features and outputs by summing smooth functions of each feature. This additivity lends interpretability to the relationships between features and outputs. More recently, works like Neural Additive Models (NAMs) [3] extend GAMs to capture non-linear behavior in an additive manner with neural networks. From the perspective of additive models, CPD can be thought of as a multi-modal extension of GAMs for multi-aspect data, and NeAT can be seen as a multi-modal extension of NAMs. The key difference between the additive models and tensor models is that tensor models train features (factor matrices) corresponding to entities.

2.4 Neural Tensor Models

Traditionally, CPD and Tucker successfully fit low-rank linear structures while they often fail to fit tensors including non-linear latent structures [25]. Thus, numerous methods replace multi-linear operations with neural networks to capture complex structures. NCF [14] is a matrix factorization model that employs Multi-Layer Perceptrons (MLPs) to learn multi-linear and non-linear interactions between users and items. NeuralCP [24] is a Bayesian tensor decomposition learning MLP where its input is a long concatenation of row factors. NTF [24] exploits a Long Short-Term Memory (LSTM) network for temporal interactions and MLP to model non-linear interactions between components for predictive tasks in dynamic relational data. CoSTCo [25] leverages two Convolutional Neural Networks (CNNs) to capture nonlinear interaction across modes and ranks, and then uses MLPs for aggregating the output of CNNs. It claimed and empirically showed that MLP-based tensor models are prone to overfit to sparse tensors due to their dense connections while CNNs avoid this problem. NTM [7] combines two neural networks to learn multi-linear and non-linear relations by considering the inner and outer products for recommendation tasks. POND [36] is a probabilistic tensor decomposition leveraging Gaussian processes for capturing complex interactions and CNN to complete a given entry. M^{2}DMTF [9] is a multi-mode nonlinear deep tensor factorization where each factor matrix is modeled with two-mode non-linear deep matrix factorization for tensor completion. JULIA [30] is a framework for a tensor decomposition model to jointly capture linear and non-linear interactions in the tensors by combining multi-linear and neural tensor models. However, the way these models rely on the interaction of different latent dimensions, makes it complicated to understand concepts of those dimensions. In contrast, NeAT not only simplifies the discovery of patterns in its latent dimensions as CPD does but also captures diverse complex structures within components. Furthermore, we show that, via NeAT, even using MLPs produces the best generalization if an appropriate design with regularization techniques is used in contrast to previous work [25].

3 PROPOSED METHOD

We propose Neural Additive Tensor Decomposition (NeAT) to accurately learn non-linear structures present in sparse tensors and to make those easy to interpret. We describe the details of the model in the following sections.

3.1 Model

A key challenge in designing our model arises from the question: How can we efficiently neuralize each rank-one component? In other words, how can we efficiently apply neural networks to each rank-one component? Given an N-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, applying a neural network to each rank-one component equals to dealing with computations proportional to the size of the given tensor. This is especially impractical for real-world sparse tensors that have high dimensionality, due to the high computational costs.

To address this, NeAT leverages the sparsity of the tensor, considering only observed entries (nonzeros) instead of the entire tensor; it replaces a linear product to neural network operations between elements of each component corresponding to indices of

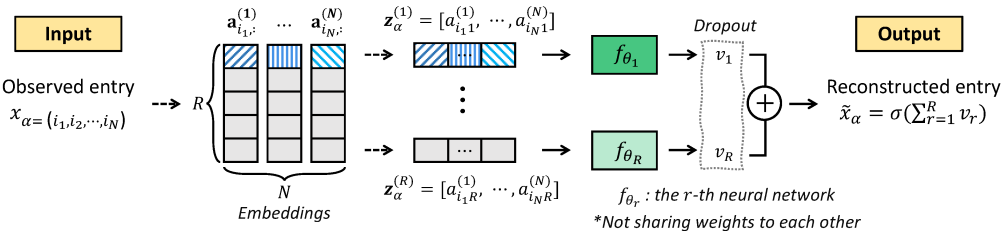


Figure 2: Illustration of a model architecture of NeAT. NeAT reconstructs observed entries by summing outputs of individually parameterized networks. While making components easy to interpretable, NeAT jointly captures various patterns in components and learning complex interactions between them with neural networks.

nonzeros. This approach is scalable to large tensors since it is independent of tensor size and more accurate than the one of treating unobserved entries as 0 for sparse tensors [1, 31]. As illustrated in Figure 2, given a nonzero $x_{\alpha=(i_1,\cdots,i_N)}$, NeAT gathers N embeddings of dimension R corresponding to an index α. Then the rth neural network uses only the rth entries of embeddings to learn non-linear interactions between them. NeAT reconstructs the nonzero by summing all outputs from neural networks.

The formulation of NeAT is defined as follows. Given an N-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with the set of the observed indices Ω and a rank R, NeAT aims to learn a set of factor matrices $\{\mathrm{A}^{(n)} \in \mathbb{R}^{I_n \times R} \mid 1 \le n \le N\}$ and a set of neural networks $\{f_{\theta_r} \mid 1 \le r \le R\}$, by reconstructing observed entries $x_\alpha (\forall \alpha \in \Omega)$ as follows:

$$x_\alpha \approx f_{\theta_1}(\mathrm{z}^{(1)}_\alpha) + \cdots + f_{\theta_r}(\mathrm{z}^{(r)}_\alpha) + \cdots + f_{\theta_R}(\mathrm{z}^{(R)}_\alpha), \tag{6}$$

where $\mathrm{z}^{(r)}_\alpha = \left[a^{(1)}_{i_1 r}, a^{(2)}_{i_2 r}, \ldots, a^{(N)}_{i_N r}\right] \in \mathbb{R}^{1 \times N}$ is a concatenation of the rth element $a^{(n)}_{i_n r}$ of embeddings corresponding to the α. Each individual neural network f_{θ_r} operates on the rth concatenated component $\mathrm{z}^{(r)}_\alpha$, and returns the rth contribution to a given entry x_α. Then, their all outputs are then aggregated to reconstruct to x_α. It is important to note that each f_{θ_r} is individually parameterized and does not share parameters with other neural networks. This allows each component to act as an individual model, capturing diverse patterns and interactions and thereby accurately reconstructing a tensor. In Sections 4.2 and 4.4.1, Table 2 and Figures 4(a) and 4(b) shows NeAT increase their performance when the rank increases.

We employ MLPs to learn non-linear interactions between the elements in the r-th component, follows as:

$$f_{\theta_r}(\mathrm{z}^{(r)}_\alpha) = \mathrm{h}^{(l-1)}_r \mathrm{W}^{(L)}_r + b^{(L)}_r, \tag{7}$$

where $\mathrm{W}^{(L)}_r \in \mathbb{R}^{d_{L-1} \times 1}$ and $\mathrm{b}^{(L)}_r \in \mathbb{R}^{1 \times 1}$ are a weight and bias of the last Lth layer. In the first layer, $\mathrm{h}^{(1)}_r \in \mathbb{R}^{1 \times d_1}$ is computed as:

$$\mathrm{h}^{(1)}_r = g(\mathrm{z}^{(r)}_\alpha \mathrm{W}^{(0)}_r + \mathrm{b}^{(0)}_r), \tag{8}$$

where $\mathrm{W}^{(0)}_r \in \mathbb{R}^{N \times d_1}$ and $\mathrm{b}^{(0)}_r \in \mathbb{R}^{1 \times d_1}$, and for the subsequent layers ($l \ge 2$), we have:

$$\mathrm{h}^{(l-1)}_r = g(\mathrm{h}^{(l-2)}_r \mathrm{W}^{(l-1)}_r + \mathrm{b}^{(l-1)}_r), \tag{9}$$

where $\mathrm{W}^{(l-1)}_r \in \mathbb{R}^{d_{l-1} \times d_l}$ and $\mathbb{R}^{1 \times d_l}$. Note that g denotes an activation function, l ($1 \le l \le L$) denotes a depth of layers, and d_l denotes the dimension size of the lth layer. We use the Rectified Linear Unit (ReLU) as an activation function for each internal layer, except for the last layer. In Section 4, we show that NeAT with 2-layer MLPs excels compared to all the baselines; however, we note that other types of neural networks can possibly replace the MLPs to further improve the accuracy, and we leave it as a future work. Furthermore, if we replace f_{θ_r} as the function that returns the product of all entries in a vector, NeAT in Equation (6) is exactly the same as CPD in Equation (2), therefore making NeAT a generalizable extension of CPD. NeAT is able to express non-linear interactions by employing neural networks while CPD is able to only capture multi-linear interactions by computing an outer product between factor matrices.

3.2 Training

In our experiments, we primarily focus on the completion of binary tensors, also known as a link prediction, and as a result, we opt for a binary cross-entropy loss function:

$$\mathcal{L}(\Theta) = -\frac{1}{|\Omega|} \sum_{\forall \alpha \in \Omega} (x_\alpha \log \tilde{x}_\alpha + (1 - x_\alpha) \log(1 - \tilde{x}_\alpha)) + \lambda R(\Theta), \tag{10}$$

where Θ includes all learnable parameters, factor matrices, and weights of MLPs. Note that x_α and $\tilde{x}_\alpha$ indicates an observed entry and reconstruction entry corresponding to α, respectively. We apply the Sigmoid function to the entry $\tilde{x}_\alpha$ to predict the probability of observed entries for the link prediction task. $\lambda R(\Theta)$ is a regularization term for all parameters followed as:

$$R(\Theta) = \sum_{n=1}^{N} \|\mathrm{A}^{(n)}\|_F^2 + \sum_{r=1}^{R} \sum_{\mathrm{W}^{(l)}_r \in \theta_r} \|\mathrm{W}^{(l)}_r\|_F^2, \tag{11}$$

where λ indicates a weight decay, $\mathrm{A}^{(n)}$ indicates the nth factor matrix, and $\mathrm{W}^{(l)}_r$ indicates lth weights in each rth neural network. In the case of real-valued tensors, we would accordingly modify our loss such as least squares.

We optimize Equation (10) of NeAT based on Adam [18] using backpropagation and jointly train factor matrices and MLPs. Although in principle a joint model like ours should be easier to train with backpropagation, we observe that only specific components are in utilization for loss minimization while others are not, resulting in poor performance. To avoid this, we apply dropout [33] to all final outputs $f_{\theta_r}(\mathrm{z}^{(r)}_\alpha)$ of MLPs. Dropout helps components to be trained appropriately by randomly selecting different subsets of components to reconstruct a tensor, preventing overfitting [19].

Table 1: Summary of six real-world sparse tensors.

Name	Dimensionality	Nonzeros	Sparsity
DBLP [1]	4,057 × 14,328 × 7,723	94,022	2.1e-07
MovieLens [2]	610 × 9,724 × 4,110	100,836	4.1e-06
YELP [3]	70,818 × 15,580 × 109	335,022	7.1e-07
FS-NYC [4]	1,084 × 38,334 × 7,641	225,701	2.8e-06
FS-TKY [4]	2,294 × 61,859 × 7,641	570,743	5.3e-07
Yahoo-M [5]	82,309 × 82,308 × 168	785,749	6.9e-07

[1] https://github.com/Jhy1993/HAN
[2] https://files.grouplens.org/datasets/movielens/ml-latest-small-README.html
[3] https://www.yelp.com/dataset
[4] https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r

Also, Table 3 and Figure 6 in Section 4.6 exhibits that applying dropout is highly effective in better generalization. We further normalize inputs for MLPs and apply dropout to each layer inside neural networks, excluding the final one for further stable training.

3.3 Complexity Analysis

We analyze space and time complexities of our model. The total dimensionality of an N-mode tensor is denoted as $I = I_1 + \cdots I_N$, where I_n represents the size of the nth mode, and the rank is denoted as R. The maximum size of the hidden dimension of neural networks is denoted as $D = max(d_1, \cdots, d_l)$ and the depth is denoted as L. The total number of parameters in NeAT is $\mathcal{O}(IR + RLD^2)$. The number of parameters for factor matrices is IR and for MLPs is $\mathcal{O}(RLD^2)$ where each has approximately equal to LD^2 parameters ($Nd_1 + d_1d_2 + \cdots + d_{l-1}d_l + d_l$). The time complexity of NeAT is linear with regard to the number of observed entries and independent to the size of a given tensor. Thus, the time complexity for all observed entries, is $\mathcal{O}(|\Omega|RLD^2)$, when forwarding a single tensor entry through each neural network, it takes $\mathcal{O}(RLD^2)$ computational cost.

We employ shallow networks (e.g., two layers) for most of experiments, showing the best link prediction performance. With the two-layer MLPs, computational costs for processing each entry becomes much smaller such as $\mathcal{O}(RD)$. Also, the parameters of neural networks are much less than the factors $\mathcal{O}(IR)$ since I is much larger than D.

4 EXPERIMENTS

We conduct experiments to answer the following questions.

- **Q1 Performance (Section 4.2).** How accurately does NeAT perform in link prediction?
- **Q2 Interpretability (Section 4.3).** Is NeAT interpretable?
- **Q3 Pattern Discovery (Section 4.4).** Can NeAT learn meaningful patterns in components?
- **Q4 Ablation Study (Section 4.5).** How do model designs affect performance?
- **Q5 Hyper-Parameter Study (Section 4.6).** How do hyperparameter settings affect performance?

4.1 Experimental Setting

We conduct experiments on a machine equipped with an AMD Ryzen CPU and an NVIDIA RTX A6000. We provide the details of the experimental setup in the following paragraphs.

4.1.1 Datasets. We use six real-world sparse tensors to evaluate the performance of the proposed method to the baselines. The datasets are summarized in Table 1. MovieLens [11], YELP, and Yahoo-M are movie, business, and music rating datasets consisting of (user, item, timestamp). FS-NYC [39] and FS-TKY [39] are check-in datasets consisting of (user, venue, timestamp), collected by Foursquare in New York and Tokyo, respectively. Each entry $x_{i,j,k}$ of each tensor is binary indicating if a user i is associated with an item j (e.g., movie, venue) at the timestamp k. DBLP is a computer science bibliography network consisting of (author, paper, terminology) , representing whether an author i published a paper j including a terminology k. We split a tensor into training, validation, and test datasets with an 8:1:1 ratio. We randomly sample negative samples with the same number of observed entries as in the split dataset.

4.1.2 Baselines. We compare NeAT to five baselines which consist of multi-linear and neural tensor decomposition models.

- CPD is a standard CPD with an L2 regularization.
- TuckER [5][1] is a Tucker decomposition model without orthogonal constraints on factor matrices for knowledge graph completion.
- NCF [14][2] is a collaborative filtering model extended for tensors, learning linear and non-linear structures with MLP.
- CoSTCo [25][3] is a tensor completion model learning non-linear structures with two 1-d CNNs and 2-layer MLPs.
- NTM [7] is a tensor decomposition model that combines the inner product and the neuralized outer product via MLP to learn both linear and non-linear structures.

4.1.3 Training. We employ Adam to optimize all models. We train all baselines except for CPD with a binary cross entropy as specified in Equation (10) since CPD shows better performance with a reconstruction loss. We select all hyper-parameters via a combination of grid search and Bayesian optimization based on early stopping. We find learning rates from $\{10^{-2}, 10^{-3}, 10^{-4}\}$, weight decays from $\{10^{-3}, 10^{-4}, 10^{-5}\}$, and ranks from $\{8, 16, 32, 64, 128\}$ for all models. We also find dimension sizes of layers from $\{8, 16, 32, 64, 128\}$, and layer depths from two to four, and batch sizes from $\{512, 1024\}$ for all neural tensor models.

4.1.4 Metrics. To evaluate the interpretability of NeAT and baselines, we examine the homogeneity of extracted patterns in components. If each component consists of homogenous patterns, it is easier to distinguish it from the others, making the extracted pattern more interpretable. In our experiment, we use auxiliary information (e.g., labels) to evaluate homogeneity of components; we use the labels of the top K entities to create a top-K label distribution. We derive two interpretability metrics: 1) AE (average entropy of the top-K label distribution) and 2) JSD (average Jenson-Shannon distance between the top-K label distributions). The AE evaluates

[1] https://github.com/ibalazevic/TuckER
[2] https://github.com/guoyang9/NCF
[3] https://github.com/USC-Melady/KDD19-CoSTCo

Table 2: Accuracy of NeAT and baselines in link prediction. NeAT is superior to all baselines in six real-world sparse tensors across different rank sizes. Note that the best-performing method is denoted in bold.

	DBLP					MovieLens					YELP				
Model \ Rank	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**
CPD	0.918	0.926	0.925	0.917	0.903	0.911	0.918	0.923	0.923	0.916	0.781	0.772	0.765	0.761	0.760
TuckER	0.834	0.837	0.946	0.966	0.965	0.902	0.919	0.934	0.941	0.944	0.829	0.831	0.831	0.833	0.826
NCF	0.834	0.836	0.821	0.937	0.879	**0.981**	**0.985**	0.987	0.989	0.988	**0.840**	**0.846**	0.849	0.849	0.852
CoSTCo	0.933	0.948	0.956	0.939	0.932	0.978	0.983	0.986	0.989	0.990	0.838	0.846	0.849	**0.857**	0.858
NTM	0.885	0.911	0.887	0.882	0.828	0.953	0.948	0.954	0.958	0.959	0.796	0.798	0.807	0.827	0.830
NeAT (Proposed)	**0.942**	**0.960**	**0.969**	**0.973**	**0.976**	**0.981**	0.984	**0.988**	**0.990**	**0.991**	0.835	0.836	**0.854**	**0.857**	**0.864**
	FS-NYC					**FS-TKY**					**Yahoo-M**				
Model \ Rank	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**	**8**	**16**	**32**	**64**	**128**
CPD	0.803	0.816	0.824	0.821	0.787	**0.869**	**0.876**	0.870	0.863	0.809	0.802	0.794	0.820	0.811	0.804
TuckER	0.805	0.811	0.816	0.829	0.838	0.852	0.854	0.870	0.876	0.881	0.866	0.879	0.886	0.891	0.908
NCF	0.794	0.796	0.817	0.818	0.825	0.854	0.860	0.859	0.861	0.875	0.826	0.851	0.845	0.870	0.888
CoSTCo	0.806	0.816	0.824	0.833	0.833	0.862	0.869	0.875	0.876	0.878	0.841	0.806	0.819	0.845	0.838
NTM	0.762	0.776	0.778	0.796	0.805	0.833	0.834	0.839	0.843	0.844	0.836	0.833	0.828	0.846	0.857
NeAT (Proposed)	**0.811**	**0.830**	**0.845**	**0.851**	**0.849**	0.861	0.873	**0.881**	**0.887**	**0.887**	**0.917**	**0.927**	**0.925**	**0.918**	**0.915**

whether each component per mode represents a distinct latent pattern. Lower entropy indicates higher occurrence of certain labels, making components more distinguishable. The JSD evaluates how well the rth component across the mode clusters together based on specific labels. The lower the distance is, the better components clusters. The AE is defined as follows.

$$AE = \frac{1}{N \times R} \sum_{n=1}^{N} \sum_{r=1}^{R} H(\mathrm{p}_r^{(n)}), \tag{12}$$

where $H(\mathrm{p}_r^{(n)}) = -\sum_{m=1}^{M} \mathrm{p}_r^{(n)}(m) \log \mathrm{p}_r^{(n)}(m)$ and N, R, and M denotes the number of modes, the rank, and the number of labels. Here,

$$\mathrm{p}_r^{(n)} \in \mathbb{R}^{M \times 1} = \frac{1}{K}[p_{r1}^{(n)} \cdots p_{rM}^{(n)}] \tag{13}$$

indicates the top-K label distribution for rth column in the nth factor matrix, where $p_{rm}^{(n)} = |\{i_n \in I_n^{'} | l_{i_n} = L_m\}|$ indicates the number of indices having the label L_m and $I_n^{'}$ indicates a set of indices of top-k values in $\mathrm{a}_r^{(n)}$. The JSD is defined as follows.

$$JSD = \frac{1}{R} \sum_{r=1}^{R} \sum_{n \in N, n \neq n'} D(\mathrm{p}_r^{(n)} || \mathrm{p}_r^{(n')}), \tag{14}$$

where D indicates Jenson-Shannon distance defined as

$$D(\mathrm{p}_r^{(n)} || \mathrm{p}_r^{(n')}) = \frac{1}{2} D(\mathrm{p}_r^{(n)} || Q) + \frac{1}{2} D(\mathrm{p}_r^{(n')} || Q), \tag{15}$$

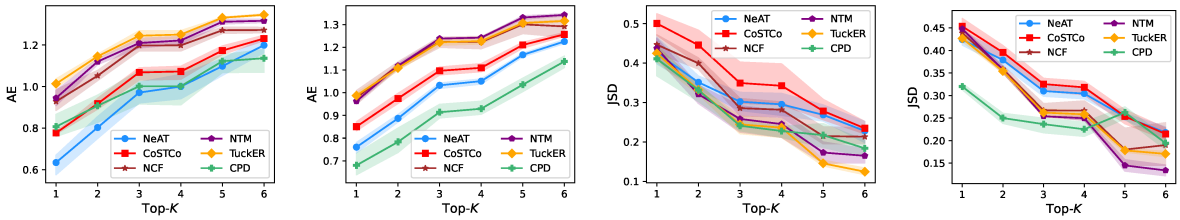
and $Q = \frac{1}{2}(\mathrm{p}_r^{(n)} + \mathrm{p}_r^{(n')})$.

4.2 Link Prediction

We evaluate NeAT and baselines on the link prediction task in terms of accuracy with six real-world sparse tensors across different ranks. For all models, we repeat experimental results averaged over three runs with optimal hyper-parameters and a fixed rank.

Table 2 describes that NeAT consistently demonstrates superior accuracy compared to all baselines including multi-linear models, CPD and TuckER, and neural tensor models, NCF, CoSTCo, and NTM, across all six datasets. NeAT consistently shows a stable and increasing trend in performance as the rank increases. This can be attributed to two factors: 1) the sum of individually parameterized components prevents the model from becoming overly expressive while capturing diverse patterns, and 2) dropout on components prevents reliance on specific components when they are training, resulting in a better generalization. For MovieLens and YELP, neural tensor models show better accuracy over multi-linear tensor models even at lower ranks (e.g., 8 or 16), indicating the presence of complex latent patterns in these datasets. This demonstrates that NeAT is able to capture complex structures without underfitting, as other neural tensor models do. For DBLP and FS-TKY, CPD performs well at lower ranks compared to neural tensor models, and neural tensor models trained with even higher ranks show marginal improvements in performance over multi-linear models. This indicates these datasets present less complex latent patterns. However, NeAT achieves better performance for all competitors, which indicates that NeAT does not overfit tensors when even inherent structures are not very complex. For FS-NYC and Yahoo-M, NeAT shows the best accuracy while neural tensor models and multi-linear tensor models show comparable performance with each other.

According to the study [25], MLP-based neural tensor models are prone to overfitting sparse tensors due to MLP's excessive over-parameterization in the form of redundant connections. CoSTCo generally performs better than neural tensor models based on MLP, NCF and NTM, due to CNNs, learning informative patterns using convolutional filters. Whereas, NeAT achieves both expressiveness based on MLP and generalization due to its additivity and proper regularization techniques. Overall, NeAT avoids two common pitfalls: it effectively captures complex structures in tensors without oversimplifying in contrast to classical tensor models, and it avoids



(a) Rank=8 (b) Rank=32 (c) Rank=8 (d) Rank=32

Figure 3: Comparison of NeAT and baselines in terms of AE and JSD of top-K label distribution. The lower, the better. NeAT shows the second-best lower AE among baselines across the different K and lower JSD between rth component across the mode. This indicates that the components extracted in NeAT are easier to interpret since they are easier to separate from each other. Note that each xtick indicates a mode-wise top-K settings and K values increase toward the right of the x-axis.

the tendency to overfit simpler structures, a challenge often faced by neural tensor models.

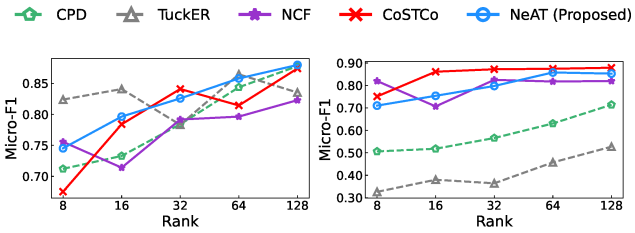
4.3 Interpretability

In our experiment, we evaluate interpretability of baselines by using a DBLP dataset consisting of (author, paper, conference) tuples. Authors, papers, and conferences are labeled with one of four study areas of study: Information Retrieval (IR), Database (DB), Data Mining (DM), and Artificial Intelligence (AI). We varies K according to the size of each mode and use six settings of top-K such as [5, 10, 3], [15, 30, 3], [30, 50, 5], [30, 100, 5], [50, 100, 10], and [100, 1000, 10]. Figure 3 demonstrates the comparison of NeAT with the baselines in terms of AE and JSD of top-K label distribution over ranks 8 to 32. NeAT shows the second-best AE and a lower JSD compared to CoSTCo, which shows the best AE among neural tensor baselines; a lower value of AE indicates that for NeAT, just like CPD, each rth component is well separated with respect to true labels and a lower value of JSD indicates that rth components across the modes learns similar distribution. For baselines such as NTM, NCF and TuckER, a higher AE and lower JSD indicates that distributions learned by respective models in each component are not separable. As signaled by higher AE, but the lower JSD indicated that differences between label distributions across rth components are small. This implies that even though all models learn similar clusters over rth components , i.e. co-clusters, those co-clusters aren't easily identifiable based on the labels for high entropy methods like NTM, NCF and TuckER.

4.4 Pattern Discovery

We evaluate if NeAT is able to capture meaningful patterns in factors, rather than having neural networks absorb all available information, through the quantitative and qualitative analyses.

4.4.1 Downstream task. We evaluate NeAT and baselines for a downstream task using a DBLP dataset consisting of (author, paper, conference) tuples. The task is to classify the research area of authors using author embeddings obtained from tensor models. Authors are labeled with one of four study areas of study: IR, DB, DM, and AI. There are two settings: a transductive and inductive setting. Under the transductive setting, an input tensor includes both

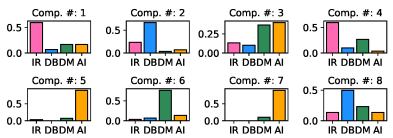


(a) Transductive (b) Inductive

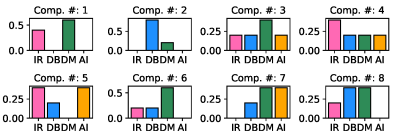
Figure 4: Comparison of NeAT and baselines in terms of Micro-F1 for the downstream task. NeAT presents better generalization compared to multi-linear models for both inductive and transductive settings since it captures diverse patterns in factors and complex interactions with MLPs.

training and test author samples. We simultaneously obtain training and test embeddings from tensor models. In the inductive setting, tensor models are trained with only a training tensor, consisting of training authors. Test embeddings for new authors are inferred using the trained models whose parameters are frozen; test embeddings are trained with the frozen trained models by reconstructing a test tensor until they converge. Note that the learnable parameters of multi-linear models are factor matrices and a core tensor, and the ones of neural tensor models are factor matrices and all weights of neural networks. To focus on evaluating the embeddings themselves, we employ a linear classifier rather than advanced classifiers.

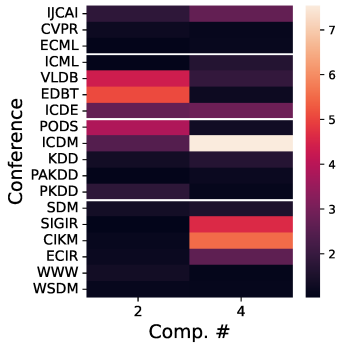
Figure 4(a) exhibits stable classification performance with embeddings obtained from NeAT in a transductive setting, indicating that the embeddings are able to capture meaningful patterns. Figure 4(b) shows that NeAT and neural tensor models have better classification performance than multi-linear tensor models in an inductive setting. This indicates that trained neural networks accurately learn complex interactions between latent patterns. Also, as rank increases, the classification performance also increases, which indicates that NeAT learns diverse patterns in factors. Therefore, NeAT is able to capture both meaningful diverse patterns and complex interactions by jointly training MLP and factor matrices.



(a) Label dist. of top-k authors



(b) Label dist. of top-k conferences



(c) Conference factor matrix

Author (comp. #2)	Label	Author (comp. #4)	Label
Jan Paredaens	DB	Barry Smyth	IR
Maurizio Lenzerini	DB	Ting Chen	DB
Anand Rajaraman	DB	Andreas Henrich	IR
Nancy A. Lynch	DB	Panos K. Chrysanthis	DB
Nicole Schweikardt	DB	Luo Si	IR
Mihalis Yannakakis	DB	Emine Yilmaz	IR
Michel de Rougemont	DB	Chris Buckley	IR
Dirk Van Gucht	DB	Koji Eguchi	IR
Thomas Schwentick	DB	Abdur Chowdhury	IR
Patricia G. Selinger	DB	Nivio Ziviani	IR

(d) Top-k authors

Figure 5: Pattern discoveries obtained from NeAT exhibit coherent co-clusters. (a) and (b) indicates the label distributions of top-k authors and conferences. (c) indicates a conference factor matrix where its y-axis is sorted in the order of AI, DB, DM, and IR labels. (d) highlights the top-k authors in the second and the fourth components.

4.4.2 Visualization. We evaluate the discovered latent patterns by visualizing them in factor matrices obtained from NeAT trained with a rank of eight on DBLP used in Section 4.4.1. Authors and conferences are categorized into four areas of study. Thanks to the simplicity of NeAT, we can easily explore factor matrices to discover patterns by considering only each column (component) of factor matrices without interactions between components. We note that other neural tensor models [7, 25, 38] and linear tensor models [5] with the exception of CPD are not amenable to the same co-clustering-based interpretation since those models interact with all dimensions of embeddings.

To explore latent patterns, we consider the top-k highest valued factors in each column of factor matrices; we set k to 100 for authors and 5 for conferences, respectively, and count the labels in those top-k to identify their distribution. We use Inverse document frequency (IDF) scores [29] to consider top-k entities that appear in only a few columns, to further enhance the interpretability. Figure 5 present coherent pattern discoveries. By examining the top-k authors and conferences in each component, we observe that each component exhibits homogeneity based on the majority votes for specific labels. For example, Figures 5(a) and 5(b) reveal that the second and fourth components are softly clustered based on DB and IR labels, respectively. This indicates that we can discover the meaningful patterns with NeAT using the same straightforward analysis procedures as in CPD. Furthermore, we display the entire conference factor matrix and top-k ranked authors.

4.5 Ablation Study

We perform an ablation study of the NeAT design to identify key design elements that affect performance. It has three main features: additivity, weight sharing, and Dropout. Additivity means that there is no interaction between the components and those components add together to reconstruct the tensor. To eliminate additivity (to learn rank-wise interaction), we apply MLP to outputs of MLPs of all components in NeAT. Weight sharing means using the same MLP instead of using individually parameterized MLPs. Dropout refers to the dropout applied on components while dropout for inner layers are added with $p = 0.5$ by default. The (+) and (-) in front of a feature indicate whether the respective feature is included or not. We fixed the rank at 32 and used a two-layer MLP with a maximum dimension of 32. Note that NeAT includes features: additivity, no sharing weights, and dropout.

Table 3 shows the results of an ablation study for the three main features considered in the model design with respect to the link prediction task. There are three findings. 1) Removing additivity does not improve performance. This indicates rank-wise interactions do not help in capturing non-linear structures but might overfit. 2) NeAT performs better than the one with shared weights, indicating that more complex interactions can be captured with individual MLPs. 3) applying dropout to components greatly improves generalization.

4.6 Hyper-Parameter Study

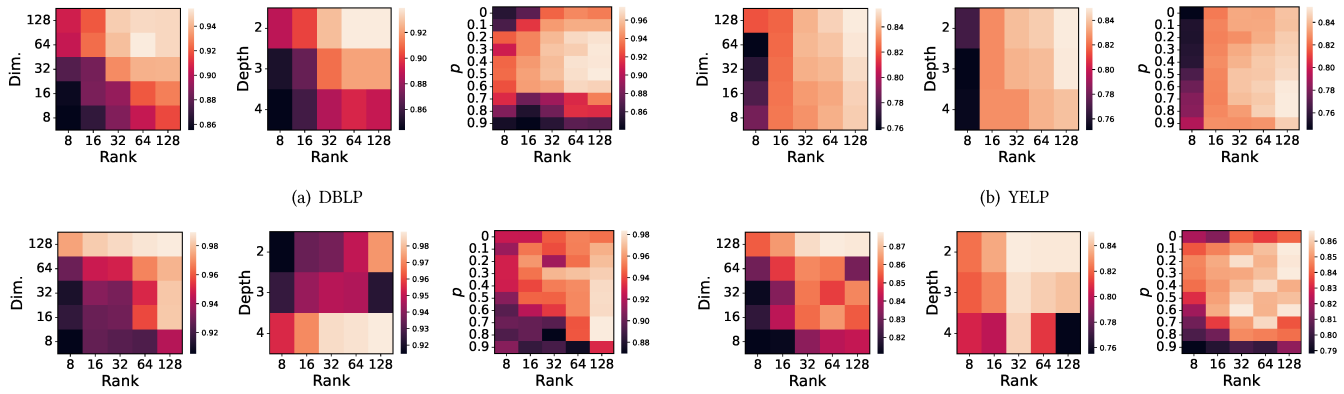
We investigate the impact of major hyper-parameters in NeAT on link prediction accuracy, focusing on dimensions of layers, layer depth, and dropout ratio across different ranks. We increase the the rank and dimension size within the range 8, 16, 32, 64, 128, increase the depth from two to four layers, and vary the dropout ratio ranging from 0 to 0.9. The first two heatmaps presented in Figure 6 reveals that increasing the dimension size generally leads to improved accuracy compared to varying the depth of layers. The third heatmaps, sensitivity analysis regarding dropout ratio, demonstrates consistent patterns across all datasets. Higher dropout ratios provide significant accuracy enhancements, particularly for larger ranks, indicating that dropout contributes to better generalization by ensuring balanced training of all components.

5 CONCLUSION

We propose NeAT, a neural tensor model that neuralizes each latent component with MLP in an additive fashion. NeAT discovers various patterns and complex interactions in real-world sparse tensors, and lends themselves to direct and intuitive interpretations. Experimental results demonstrate the generalization of NeAT for link prediction and validate the applicability of its factor matrices to downstream tasks. Through visualization of the discovered latent patterns, we illustrate their coherent co-clusters and provide

Table 3: Ablation study on combinations of different features of a model design. The feature of model designs are additivity between components, dropout on components, and weight sharing between neural networks. Note that (+) and (-) indicates each feature is included or not. NeAT shows the best performance among the combinations of features for link prediction.

Model \ Dataset	DBLP	MovieLens	YELP	FS-NYC	FS-TKY	Yahoo-M
Benchmark (NeAT)	**0.964**	0.986	**0.848**	**0.843**	**0.877**	**0.909**
(-)Dropout	0.914 (-5.0%)	0.981 (-0.5%)	0.826 (-2.2%)	0.823 (-2.0%)	0.826 (-5.1%)	0.870 (-3.9%)
(-)Additivity	0.941 (-2.3%)	0.986 (0.0%)	0.836 (-1.2%)	0.821 (-2.2%)	0.861 (-1.6%)	0.879 (-3.0%)
(-)Additivity, (-)Dropout	0.903 (-6.1%)	0.983 (-0.3%)	0.813 (-3.5%)	0.795 (-4.8%)	0.847 (-3.0%)	0.875 (-3.4%)
(+)Weight Sharing	0.963 (-0.1%)	**0.989** (0.3%)	0.839 (-0.9%)	0.830 (-1.3%)	0.874 (-0.3%)	0.883 (-2.6%)
(+)Weight Sharing, (-)Dropout	0.947 (-1.7%)	0.979 (-0.7%)	0.821 (-2.7%)	0.791 (-5.2%)	0.847 (-3.0%)	0.839 (-7.0%)
(-)Additivity, (+)Weight Sharing	0.953 (-1.1%)	0.983 (-0.3%)	0.834 (-1.4%)	0.822 (-2.1%)	0.864 (-1.3%)	0.870 (-3.9%)
(-)Additivity, (+)Weight Sharing, (-)Dropout	0.926 (-3.8%)	0.979 (-0.7%)	0.826 (-2.2%)	0.809 (-3.4%)	0.855 (-2.2%)	0.867 (-4.2%)



(a) DBLP (b) YELP

(c) MovieLens (d) FS-TKY

Figure 6: The impact of major hyper-parameters in NeAT on link prediction accuracy. From left to right, the heatmap describes accuracy regarding dimensions of layers, layer depth, and dropout ratio across different ranks. NeAT improves accuracy with larger ranks, dimension, and higher dropout ratio.

insights into understanding each component. Additionally, the ablation study and hyper-parameter study highlight the importance of proper model design and regularization techniques for generalization. In future work, we aim to study the generalization of the proposed method in terms of ensemble learning. Also, we will extend NeAT to handle various types of tensors including counts, and probability measures, which are commonly encountered in real-world tensor data. Furthermore, we intend to enhance the interpretability of interactions in MLP.

ACKNOWLEDGMENTS

The first author was a summer intern at CISCO. Research at UC Riverside was supported in part by the National Science Foundation under CAREER grant no. IIS 2046086 and CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) grant no. 2112650, by the Agriculture and Food Research Initiative Competitive Grant no. 2020-69012-31914 from the USDA National Institute of Food and Agriculture, the University Transportation Center for Railway Safety (UTCRS) at UTRGV through the USDOT UTC Program under Grant No. 69A3552348340, and by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

REFERENCES

[1] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. 2011. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems* 106, 1 (2011), 41–56.

[2] Ardavan Afshar, Kejing Yin, Sherry Yan, Cheng Qian, Joyce Ho, Haesun Park, and Jimeng Sun. 2021. Swift: Scalable wasserstein factorization for sparse nonnegative tensors. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 6548–6556.

[3] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey Hinton. 2021. Neural Additive Models: Interpretable Machine Learning with Neural Nets. arXiv:2004.13912 [cs.LG]

[4] Saba Al-Sayouri, Ekta Gujral, Danai Koutra, Evangelos E Papalexakis, and Sarah S Lam. 2020. t-pine: Tensor-based predictable and interpretable node embeddings. *Social Network Analysis and Mining* 10 (2020), 1–11.

[5] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590* (2019).

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

[7] Huiyuan Chen and Jing Li. 2020. Neural tensor model for learning multi-aspect factors in recommender systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 2020.

[8] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5, 2 (2011), 1–27.

[9] Jicong Fan. 2021. Multi-mode deep matrix and tensor factorization. In *international conference on learning representations*.

[10] Shikai Fang, Xin Yu, Zheng Wang, Shibo Li, Mike Kirby, and Shandian Zhe. 2023. Functional Bayesian Tucker Decomposition for Continuous-indexed Tensor Data. *arXiv preprint arXiv:2311.04829* (2023).

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[12] Richard A Harshman et al. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis. (1970).

[13] Trevor Hastie and Robert Tibshirani. 1990. *Generalized additive models*. Wiley Online Library.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[15] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. 2014. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 115–124.

[16] Bo Hui and Wei-Shinn Ku. 2022. Low-rank Nonnegative Tensor Decomposition in Hyperbolic Space. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 646–654.

[17] Jun-Gi Jang and U Kang. 2020. D-tucker: Fast and memory-efficient tucker decomposition for dense tensors. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1850–1853.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Arinbjörn Kolbeinsson, Jean Kossaifi, Yannis Panagakis, Adrian Bulat, Animashree Anandkumar, Ioanna Tzoulaki, and Paul M Matthews. 2021. Tensor dropout for robust learning. *IEEE Journal of Selected Topics in Signal Processing* 15, 3 (2021), 630–640.

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

[21] Tamara G Kolda and David Hong. 2020. Stochastic gradients for large-scale tensor decomposition. *SIAM Journal on Mathematics of Data Science* 2, 4 (2020), 1066–1095.

[22] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926* (2020).

[23] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*. PMLR, 2863–2872.

[24] Bin Liu, Lirong He, Yingming Li, Shandian Zhe, and Zenglin Xu. 2018. Neuralcp: Bayesian multiway data analysis with neural tensor decomposition. *Cognitive Computation* 10 (2018), 1051–1061.

[25] Hanpeng Liu, Yaguang Li, Michael Tsang, and Yan Liu. 2019. Costco: A neural tensor completion model for sparse tensors. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 324–334.

[26] Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation schema induction using tensor factorization with side information. *arXiv preprint arXiv:1605.04227* (2016).

[27] Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. 2013. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*. IEEE, 899–905.

[28] Moonjeong Park, Jun-Gi Jang, and Lee Sael. 2021. VeST: Very sparse tucker factorization of large-scale tensors. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 172–179.

[29] Namyong Park, Byungsoo Jeon, Jungwoo Lee, and U Kang. 2016. Bigtensor: Mining billion-scale tensor made easy. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2457–2460.

[30] Cheng Qian, Kejun Huang, Lucas Glass, Rakshith S Srinivasa, and Jimeng Sun. 2022. Julia: Joint multi-linear and nonlinear identification for tensor completion. *arXiv preprint arXiv:2202.00071* (2022).

[31] Kijung Shin, Lee Sael, and U Kang. 2016. Fully scalable methods for distributed tensor factorization. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 100–113.

[32] Nicholas D Sidiropoulos and Rasmus Bro. 2000. On the uniqueness of multilinear decomposition of N-way arrays. *Journal of Chemometrics: A Journal of the Chemometrics Society* 14, 3 (2000), 229–239.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[34] Zerui Tao, Toshihisa Tanaka, and Qibin Zhao. 2024. Efficient Nonparametric Tensor Decomposition for Binary and Count Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15319–15327.

[35] Jos MF ten Berge. 2000. The typical rank of tall three-way arrays. *Psychometrika* 65 (2000), 525–532.

[36] Conor Tillinghast, Shikai Fang, Kai Zhang, and Shandian Zhe. 2020. Probabilistic neural-kernel tensor decomposition. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 531–540.

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

[38] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V Chawla. 2019. Neural tensor factorization for temporal interaction learning. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 537–545.

[39] Dingqi Yang, Daqing Zhang, Vincent. W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2015), 129–142.

[40] Lina Yao, Quan Z Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. 2015. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 1007–1010.

[41] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1153–1162.