T3SRS: Tensor Train Transformer for Compressing Sequential Recommender Systems

Hao Li[a], Jianli Zhao[a,*], Huan Huo[b], Sheng Fang[a], Jianjian Chen[a,c], Lutong Yao[a], Yiran Hua[a]

[a] *College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China*
[b] *Faculty of Engineering and Information Technology, University of Technology Sydney, Australia*
[c] *Shandong Provincial Academy of Educational Recruitment and Examination, Jinan, China*

Abstract

In recent years, attention mechanisms have gained popularity in sequential recommender systems (SRSs) due to obtaining dynamic user preferences efficiently. However, over-parameterization of these models often increases the risk of overfitting. To address this challenge, we propose a Transformer model based on tensor train networks. Initially, we propose a tensor train layer (TTL) to accommodate the original weight matrix, thus reducing the space complexity of the mapping layer. Based on the TTL, we reconfigure the multi-head attention module and the position-wise feed-forward network. Finally, a tensor train layer replaces the output layer to complete the overall compression. According to the experimental results, the proposed model compresses SRSs parameters effectively, achieving compression rates of $76.2\% - 85.0\%$, while maintaining or enhancing sequence recommendation performance. To our knowledge, the Tensor Train Transformer is the first model compression approach for Transformer-based SRSs, and the model is broadly applicable.

Keywords: Sequential recommender systems, Model compression, Transformer, Tensor train network

*Corresponding author.
Email addresses: lihao95@sdust.edu.cn (Hao Li), jlzhao@sdust.edu.cn (Jianli Zhao), Huan.Huo@uts.edu.au (Huan Huo), fangs99@126.com (Sheng Fang), e37_chenjj@sdzk.cn (Jianjian Chen), lutongyao@sdust.edu.cn (Lutong Yao), Nariy_Auh@outlook.com (Yiran Hua)

1. Introduction

Recommender systems (RSs) have become effective tools for tackling the problem of information overload and are widely applied in various fields, such as e-commerce and social media. Among them, sequential recommender systems (SRSs), which model user behavior sequences and simulate their interest changes using neural networks such as Recurrent neural network (RNN) (Jannach & Ludewig, 2017), Convolutional neural network (CNN) (Tang & Wang, 2018), Memory network (Chen et al., 2018), and Attention network (Kang & McAuley, 2018; Sun et al., 2019), has become a hot research topic in the recommendation field due to its ability to capture dynamic preference. RNN is commonly used for sequence modeling, CNN can capture union-level patterns in sequences, memory networks introduce memory modules to store sequence interaction information, and attention networks consider the importance of user sequence behavior interaction and long-distance correlation to capture users' dynamic preferences more accurately, which has gained widespread recognition.

Although attention models are widely popular, their complex structure is inevitable, especially in the case of Transformer models. As user interaction behavior becomes increasingly complex, model sizes grow larger and require stacking multiple layers of network structures involving numerous learnable parameters, leading to over-parameterization[1] and parameter redundancy issues. Moreover, training SRSs is not an easy task. For example, the BERT4Rec (Sun et al., 2019) model involves over 9 million parameters on the MovieLens-20M dataset, and training the complete model takes approximately three hours. This also limits the effective deployment of the model under limited resources. Therefore, to mitigate over-parameterization and reduce the number of model

[1]Over-parameterization usually means the situation in which the amount of information is insufficient to estimate a large number of parameters of deep networks, which leads to inaccuracy and high cost for the model's inference.(Fan et al., 2021)

parameters, our primary objective is to compress Transformer-based SRSs.

In the field of model compression research, various techniques have emerged (Novikov et al., 2015; Ma et al., 2019; Pan et al., 2019; Zafrir et al., 2019; Shen et al., 2020) that can reduce the scale of neural networks with limited loss in accuracy. However, compressing Transformer-based SRSs presents unique challenges. Firstly, the Transformer-based models are more complex and have more parameters compared to CNN-based and RNN-based models. The Transformer-based models are composed of multiple network layers, including input layer, Transformer layer, and output layer. The Transformer layer can be divided into multi-head attention (MHA) and position-wise feed-forward (PWFF) networks. MHA can alse be considered a nonlinear function (Ma et al., 2019), further increasing the models' complexity. The second challenge lies in ensuring end-to-end training. To effectively model user preferences, SRSs typically have multiple network components. However, the structure of these components is not uniform. After compressing the model, it becomes essential to retrain these components to ensure their accuracy.

Among the many compression methods, tensor decomposition (Ma et al., 2019) is a parameter fitting method that can fit the original parameters with limited parameters. Tensor train decomposition has been adopted by lower space complexity (Yin et al., 2021). Our goal is to reduce the size of the model parameters. To do this, we analyze the model and locate its parameter positions. A few parameters are used to fit the original parameters and end-to-end training is ensured to reduce the loss of accuracy.

To address the aforementioned challenges, this paper proposes a Tensor Train Transformer model for SRSs, namely T3SRS. Firstly, the Transformer model architecture comprises two key components, MHA and PWFF, whose learnable parameters mainly concentrate on linear layers, with the scaled dot-product attention only involving attention matrix computation. Thus, we design tensor train layer (TTL) to handle these parameters, using multiple learnable fourth-order tensor train factors to fit the weights and adding bias weights. Next, multi-head attention with tensor train layers (TT-HMA) is proposed to replace

the traditional MHA module. TT-MHA integrates TTLs into the multi-head attention to effectively reduce parameters of MHA. In the PWFF network, a similar method is adopted, and position-wise feed-forward with tensor train layers (TT-PWFF) is proposed. In the SRSs, the final classification layer is also replaced with a TTL, and the replaced module can be connected with other model structures for joint training. In summary, the contributions of this paper are as follows.

1. In addressing the challenge of parameter redundancy in SRSs, we propose the Tensor Train Transformer. By utilizing tensor train decomposition, this model represents the parameter matrices in the form of a tensor network. To our knowledge, this work is the first to apply tensor train networks specifically for compressing SRSs, and fills the gap in the research on compression of Transformer-based SRSs.
2. Through a systematic and comprehensive examination of the space complexity, we propose a tensor train layer. Unlike traditional tensor decomposition approaches, within the Transformer framework, we innovate by substituting the original third-order TT factors with several fourth-order tensors. This format effectively compresses storage and computational complexity and maintains higher accuracy compared to the original model, as verified in Section 5.2.
3. To address the challenges associated with the training process in compressed models, we design the TT-MHA and TT-PWFF components to ensure end-to-end training. In addition, we replace the traditional classification layer of SRSs with the tensor train layer to further reduce the space complexity. Compared with similar compression methods, our method has a larger compression scope, as verified in Section 5.6.

The remaining structure of this paper is as follows. Related work will be presented in Section 2. Section 3 will mainly describe the background and preparation work. Section 4 will describe the compressed model and specific details. The experiment and the analysis of the results will be described in

Section 5. Finally, we will summarize the advantages and disadvantages of this work in Section 6, and describe the future work.

2. Related work

2.1. Sequential recommendation

In recent years, various SRSs based on deep neural networks have emerged. RNN (Jannach & Ludewig, 2017) was the first to be applied in SRSs. Hidasi et al. (2016b) used RNNs to extract high-quality features and proposed the parallel RNN framework, which utilizes different RNN architectures to build user feature vectors and predict the next user behavior. However, when there is no strong correlation between the previous and next sequences, the performance of the RNN model is not satisfactory. Therefore, CNN has also been applied to sequence recommendation tasks. The Caser model (Tang & Wang, 2018) uses convolutional kernels to capture the local features of item embeddings and provides the ability to model general preferences and sequential preferences.

The model mentioned above does not consider the importance of past interactions. To overcome this, attention mechanisms have been employed in sequential data modeling, yielding impressive results (Kang & McAuley, 2018). For instance, the hierarchical attention mechanism (Du et al., 2022) is used to capture the dependencies of high-level items and, through weighting, infer the main objective of sequence recommendation. The STAMP model (Liu et al., 2018) introduces a limited short-term attention/memory model, which learns the unified embedding space of the entire session item. These models all integrate attention mechanisms into the original model. With the Transformer model composed entirely of attention mechanisms (Vaswani et al., 2017), the attention model's sequence extraction ability was fully utilized. Subsequently, the BERT model's (Devlin et al., 2019) introduction achieved cutting-edge outcomes in numerous natural language processing tasks. In sequence recommendation, there also exist entirely attention-based systems (Kang & McAuley, 2018; Sun et al., 2019; Wu et al., 2020; Li et al., 2020; Liu et al., 2021; Fan et al., 2021;

Hou et al., 2022). SASRec is a self-attention based SRS proposed by Kang & McAuley (2018), and BERT4Rec (Sun et al., 2019) is a two-way sequential recommender system trained through Cloze tasks. While these attention-based sequential recommendation models have demonstrated significant effects, model compression research has become critical due to many training parameters and the high training costs.

2.2. Model compression

Generally, the current model compression approaches can be classified into pruning (Lin et al., 2022), quantization (Bartol et al., 2015; Shen et al., 2020), weight sharing (Dehghani et al., 2019; Lan et al., 2019), and low rank fitting (Novikov et al., 2015; Ma et al., 2019; Pan et al., 2019). The application of tensor networks to neural networks and the creation of tensor networks is an emerging field because of the superior ability of tensor networks to fit original weights with a small number of parameters (Wang et al., 2023). For instance, in literature (Novikov et al., 2015; Qiang & Ji, 2022), researchers successfully employed tensor decomposition methods to compress convolutional layer neural networks. The Tensorizing Neural Network (Novikov et al., 2015) represents the weight matrix in a convolutional neural network using tensor train decomposition. Wang et al. (2022b) utilized CP decomposition to compress convolutional neural networks, while Aggarwal et al. (2018) constructed the tensor ring network using tensor ring decomposition to compress the full connectivity layer and convolution layer. RNN (Xu et al., 2021) also adopts this form of weight matrix representation. Although these techniques effectively reduce the parameters of neural networks, their suitability for attention models remains unexplored.

In the Transformer model, the Sparse Transformer (Child et al., 2019) applies sparse technology to attention matrix computation and reduces the number of parameters. This technique utilizes a sparse attention matrix to decrease model parameters by selecting certain positional information on the attention matrix, but the number of reduced parameters is limited. Ma et al. (2019) proposed a novel attention model that utilizes Block Term Tensor Composition (BTD),

achieving good results in natural language processing tasks. Pham Minh et al. (2022) proposed a compressed Transformer, based on tensor train decomposition in image classification task. However, the BTD-Transformer and TT-ViT models only deal with original multi-head attention modules. In addition, the TT-ViT model uses a TT matrix-by-vector product, which fails to preserve the originally existing bias term. Inspired by Tucker decomposition representations, Tuformer (Liu et al., 2022) designed a data-driven trainable header Transformer with a theoretically reliable framework. Nevertheless, most of these methods focus on language models, and model compression in the recommendation field, especially attention models, has not been extensively studied. In the recommendation field, Hrinchuk et al. (2020) proposed using tensor train decomposition to process the embedded layer, which can be applied to various model structures. TT-rec (Yin et al., 2021) replaces large embedded tables in the deep learning recommendation model with a series of matrix products and introduces a new initialization method. However, Tensorized Embedding (Hrinchuk et al., 2020) and TT-Rec compress only the Embedding layer and do not consider the main structure of the sequential recommender systems, such as the Transformer layer. CpRec (Sun et al., 2020) applies low-rank decomposition and weight sharing to compress the parameter amount of the NextItNet model. While CpRec compresses the feature interaction layer and the Embedding layer, it operates within a multi-layered CNN architecture, rendering it ill-suited for the intricacies of the Transformer architecture. Despite being inspired by models in the field of natural language processing, some attention models in sequential recommender systems, such as SASRec and BERT4Rec, still face challenges in model compression. In this paper, we propose a Tensor Train Transformer for SRSs, namely T3SRS. T3SRS takes advantage of the powerful fitting ability of tensor train networks and the rationality of approximating higher-order tensors to reduce the space consumption of SRSs. Unlike the above compression methods, the T3SRS model has a stronger compression capability, which realizes the parameter compression of the multi-head attention layer, the position-wise feed-forward network, and the classification layer. At the same time, the T3SRS

model is able to retain the original bias terms present in the network, as detailed in Section 4.2.

3. Background and preliminaries

In this section, for ease of understanding, we will provide some background and preliminary work related to tensors, including definitions of symbols, some operations of tensors, and tensor train networks.

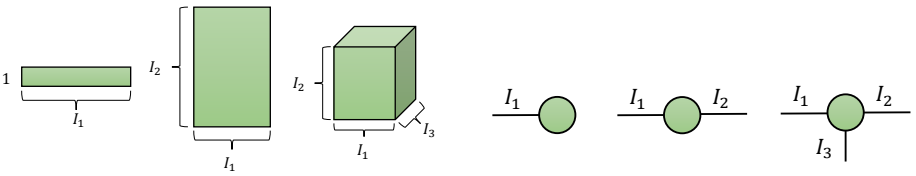
3.1. Tensor and tensor contraction

Following the paper (Kolda & Bader, 2009), a tensor is a multidimensional array. As shown in Table 1, an Nth-order tensor is represented by $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. A vector is a first-order tensor, represented by bold lowercase letter $\mathbf{x} \in \mathbb{R}^{I_1}$. A matrix is a second-order tensor, represented by bold uppercase letter $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. Similarly, the elements of tensor $\mathcal{X}$ are denoted by handwritten uppercase letter $\mathcal{X}_{i_1,\ldots,i_N}$, with indices $(i_1, i_2, \ldots, i_N)$.

Table 1: Tensor notations

Symbol	Explanation
$\mathbf{x} \in \mathbb{R}^{I_1}$	vector
$\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$	matrix
$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Nth-order tensor
I	dimensionality
$\circ$	outer product operation

A tensor network is a countable collection of small-scale tensors that are interconnected through tensor contractions (Qiang & Ji, 2022). A fundamental tensor diagram is depicted in Figure 1, where tensors are visualized as nodes and edges. Each node represents an independent algebraic object, and each edge represents a mode (or index) of the algebraic object.



(a) Graphical representation (b) Tensor diagram representation

Figure 1: Two modes of representation Vector $\mathbf{x} \in \mathbb{R}^{I_1}$, Matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$.

As depicted in Figure 2, the connection of two algebraic objects on the same mode represents summation over that mode. Figure 2 (a) shows the contraction of two matrices $\sum_{I_2} \mathbf{A}_{I_1,I_2}\mathbf{B}_{J_1,J_2}$, where $I_2 = J_1$. Figure 2 (b) shows the contraction of two vectors, which is the inner product operation $\sum_{I_2} \mathbf{a}_{I_2}\mathbf{b}_{J_1}$. Figures 2 (c) and (d) illustrate more complex tensor operations.

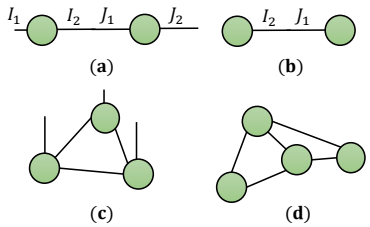


Figure 2: Graphical representation of tensor operations.

Tensor contraction refers to the process of merging two tensors on one or more modes to form a new tensor. It requires that the dimensions of the contracted modes be consistent between the two tensors, as shown in Figure 3. Tensor contraction can be viewed as a higher-order extension of matrix multiplication in the field of tensors. For example, given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times J_3 \times J_4 \times J_5}$, where $I_5 = J_3$ and $I_3 = J_2$, contracting the two tensors on the common indices I_5, I_3 yields tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times I_2 \times I_4 \times J_1 \times J_3 \times J_4}$, whose elements are computed according to Eq. 1.

$$\mathcal{C}_{i_1,i_2,i_4,j_1,j_3,j_4} = \sum_{i_3,i_5} \mathcal{A}_{i_1,i_2,i_3,i_4,i_5}\mathcal{B}_{j_1,j_2,j_3,j_4,j_5} \tag{1}$$

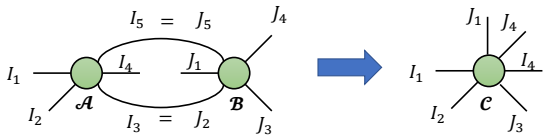


Figure 3: Diagram for tensor contraction.

3.2. Tensor train network

The traditional tensor train (TT) decomposition aims to decompose a higher-order tensor into several third-order tensors. For a N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$, the TT decomposition is given by

$$\mathcal{X} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \mathcal{G}^{(1)}(r_1, :, r_2) \circ \mathcal{G}^{(2)}(r_2, :, r_3) \circ \cdots \circ \mathcal{G}^{(N)}(r_N, :, r_{N+1}) \quad (2)$$

where $\mathcal{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$, $n = 1, \ldots, N$ are the core factors. $\{R_n\}_{n=1}^{N+1}$ represent the ranks of TT decomposition factors, with $R_1 = R_{N+1} = 1$. Figure 4 illustrates the TT decomposition form of an N-th order tensor.

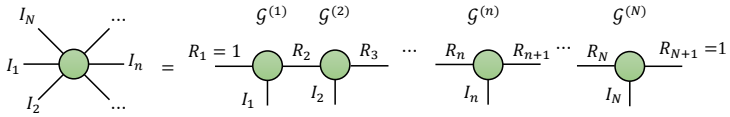


Figure 4: Tensor train decomposition.

In this work, we will use several fourth-order tensors instead of the original third-order TT factors, as shown in Figure 5. We will represent higher-order tensors using a network composed of fourth-order tensor factors and their contraction operations. Compared with other matrix decomposition techniques, the tensor train format does not require tensors to have specific structures or attributes, and tensor train decomposition can significantly reduce storage and computational requirements, but it still maintains relatively high approximation accuracy.

The formula for this representation is as follows:

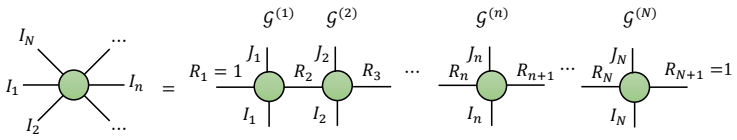


Figure 5: Tensor train network.

$$\mathcal{X} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \mathcal{G}^{(1)}(r_1, :, : r_2) \circ \mathcal{G}^{(2)}(r_2, :, : r_3) \circ \cdots \circ \mathcal{G}^{(N)}(r_N, :, : r_{N+1}) \quad (3)$$

where $\mathcal{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times J_n \times R_{n+1}}, n = 1, \ldots, N$ are fourth-order tensor factors. $\{R_n\}_{n=1}^{N+1}$ represent the ranks of tensor train network factors, with $R_1 = R_{N+1} = 1$.

4. Proposed model

This section provides a detailed overview of the proposed model. First, we articulate the framework and underlying motivation for the design of the model. Subsequently, we describe each part of the model. Finally, a theoretical analysis of the complexity of the model is presented.

4.1. Framework and motivation

Figure 6 illustrates the original Transformer, weight matrix tensorization, and Tensor Train Transformer. Primarily, to circumvent model overparameterization and achieve compression, it is essential to analyze the parameter configurations and complexity of the original model. Upon analyzing the model and observing the parameter positions, it becomes apparent that the MHA and PWFF components entail a significant number of parameters. When the model's channel number is d, the parameters of these two components are approximately $12d^2$, mainly consisting of linear mapping weight matrices, as shown in the left panel of Figure 6.

Upon completing the analysis of model parameters, our objective is to reduce the parameter count while preserving the recommendation efficacy of the model.

The tensor train network introduces an effective parametrization paradigm that encapsulates the model's weight matrix within tensor train factors. Inspired by this, we propose the Tensor Train Transformer, as illustrated in the right panel of Figure 6. This framework decomposes the original large parameter matrix into a series of smaller tensors, significantly reducing the model's parameters while maintaining its inherent expressive power.

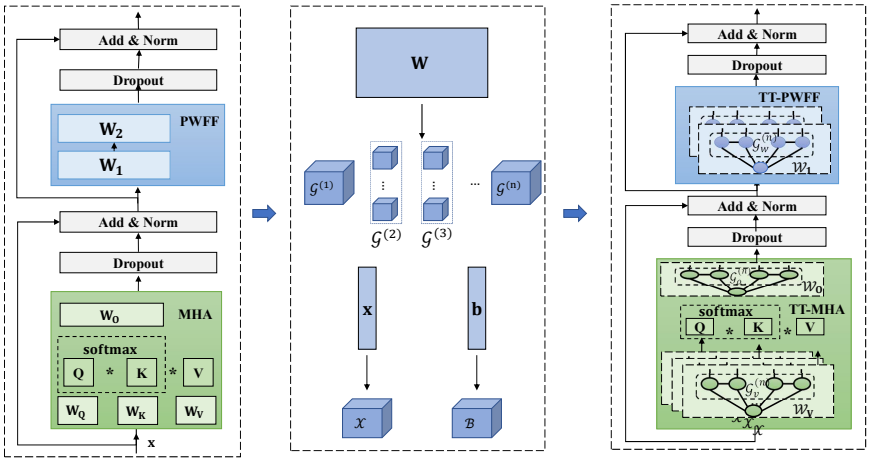


Figure 6: Schematic diagram of Tensor Train Transformer model. The left panel depicts the original Transformer model, the middle panel illustrates the weight tensorization process, and the right panel showcases the Tensor Train Transformer.

Specifically, we use the tensor train network to approximate the weight matrices and reduce the model's space complexity. Firstly, in order to apply tensor train layers, we tensorize the high-dimensional interaction sequence into a higher-order tensor, then use tensor train layers to extract local and inter-factor features while retaining self-attention mechanisms to capture global sequence patterns. Furthermore, the factor weights of the tensor train layer's input and output are shared, enabling the model to learn implicit knowledge from the input while improving performance on compressed models.

The Tensor Train Transformer effectively circumvents the inherent over-parameterization dilemma present in traditional models. Our experimental findings corroborate that, even with a substantial reduction in parameters, this

model consistently matches or surpasses conventional models across various performance metrics.

4.2. Tensor train layer

In this section, we introduce the tensor train layer (TTL), which represents weight matrices as a contraction of multiple fourth-order tensors, thereby reducing the number of model parameters. Unlike the traditional TT decomposition, our approach utilizes multiple fourth-order tensors for weight matrix fitting, as detailed in Section 3.2. When compressing Transformer-based SRSs, the fourth-order tensor is more conducive to maintaining the performance of the compressed model.

First, the input vector $\mathbf{x} \in \mathbb{R}^I$ of the linear layer is tensorized into a higher-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ using a pre-defined tensor train network. The weight matrix $\mathcal{W} \in \mathbb{R}^{I \times J}$ is also tensorized into a higher-order tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_N}$, where $\prod_{n=1}^{N} I_n = I$ and $\prod_{n=1}^{N} J_n = J$. Then, multiple fourth-order tensor factors are contracted to approximate the weight matrix tensorized as a higher-order tensor $\mathcal{W}$, which can be expressed as:

$$\mathcal{W}_{i_1,\ldots,i_N,j_1,\ldots,j_N} = \mathcal{G}^{(1)}(:, i_1, j_1, :) \cdots \mathcal{G}^{(N)}(:, i_N, j_N, :) \tag{4}$$

where $\mathcal{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times J_n \times R_{n+1}}$, $n \in 1, \ldots, N$, and N is the number of tensor factors. $\{R_n\}_{n=1}^{N+1}$ represent the ranks of tensor train factors, with $R_1 = R_{N+1} = 1$.

TTL operations can be expressed as:

$$\mathcal{Y}(j_1, \ldots, j_N) = \sum_{i_1,\ldots,i_N} \mathcal{X}(i_1, \ldots, i_N)\mathcal{W}(i_1, \ldots, i_N, j_1, \ldots, j_N) + \mathcal{B}(j_1, \ldots, j_N) \tag{5}$$

where $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$. represents the output tensor of the TTL, whose order is consistent with the higher-order tensor obtained from the input.

Finally, to facilitate data transfer with other modules in the network, we vectorize the output tensor $\mathcal{Y}$ into a high-dimensional vector $\mathbf{y} \in \mathbb{R}^J$ and represent the tensor train layer as $\mathbf{y} = TTL(\mathcal{G}^{(n)}, \mathbf{x}, \mathcal{B})$. The structure of a TTL

is illustrated in Figure 7. The parameter comparison between the original fully connected layer and the TTL can be expressed as $\frac{I\times J+J}{\sum_{n=1}^{N} R_n I_n J_n R_{n+1}+J}$.

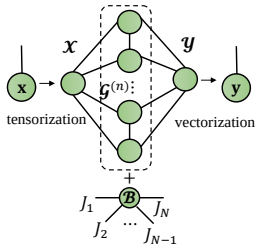


Figure 7: Schematic diagram of TTL.

4.3. Tensor Train Transformer

In the original Transformer layer, the main components include the MHA layer and the PWFF layer. In this section, TTL is used to replace the linear layer in the two parts, as described below.

4.3.1. TTL for MHA Layer

In Transformer, MHA is composed of multiple scaled dot product attention. The input includes matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, representing queries, keys, and values, respectively. Attention calculation can be expressed as Eq. 6.

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \tag{6}$$

where d_k is the scaling factor. In Vaswani et al. (2017), the calculation of MHA is expressed as:

$$MHA(\mathbf{Q}', \mathbf{K}', \mathbf{V}') = Concat(head_1, \ldots, head_h)\mathbf{W}^O \tag{7}$$

where $head_i = Attention(\mathbf{Q}'\mathbf{W}_i^Q, K'\mathbf{W}_i^K, V'\mathbf{W}_i^V)$ and $\mathbf{W}_i^Q \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d\times d_v}$, $\mathbf{W}^O \in \mathbb{R}^{hd_h\times d}$. In practice, $d_v = d_k = d_h = d/h$. Multiple sets of weights, $(\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}^O)$, lead to weight redundancy and large spatial resource consumption issues for MHA.

By replacing the linear layer in the original MHA with TTL, we obtain TT-MHA. Compared to the original MHA, TT-MHA can learn more compact representations and reduce the number of internal parameters. In the case of h attention heads, the representation of TT-MHA is as follows:

$$TTMHA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = TTL(\mathcal{G}_O^*, (head_1, \ldots, head_h), \mathcal{B}_O) \tag{8}$$

where $head_i = Attention(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$. $\mathbf{Q}_i$, $\mathbf{K}_i$ and $\mathbf{V}_i$ denote the representation of each head obtained by TTL. The attention calculation function is represented by Eq. 9.

$$\begin{cases} Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \\ \mathbf{Q} = TTL(\mathcal{G}_Q^*, \mathbf{X}, \mathcal{B}_Q) \\ \mathbf{K} = TTL(\mathcal{G}_K^*, \mathbf{X}, \mathcal{B}_K) \\ \mathbf{V} = TTL(\mathcal{G}_V^*, \mathbf{X}, \mathcal{B}_V) \end{cases} \tag{9}$$

where $\mathcal{G}_Q^*, \mathcal{G}_K^*, \mathcal{G}_V^*, \mathcal{G}_O^*$ represent the learnable tensor train factorization set, $\mathcal{G}^*$ denotes $\left\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\right\}$. Let $hd_h = I_1 \times \ldots \times I_N$ and $d = J_1 \times \ldots \times J_N$, then the space consumption is reduced from $hd_h \times d + d$ to $\sum_{n=1}^{N} I_n J_n R_n R_{n+1}$.

4.3.2. TTL for PWFF layer

The PWFF layer is designed to capture non-linear characteristics and interactions between different dimensions. It consists of two fully connected layers and an activation function, with the input being the output of the MHA $\mathbf{O}^{MHA}$. Its calculation can be expressed as Eq. 10.

$$\mathbf{O}^{FF_2} = \sigma(\mathbf{O}^{MHA}\mathbf{W}_{f_1} + \mathbf{b}_{f_1})\mathbf{W}_{f_2} + \mathbf{b}_{f_2} \tag{10}$$

where $\mathbf{W}_{f_1}$ and $\mathbf{b}_{f_1}$ are the weights and biases of the first forward layer, and$\mathbf{W}_{f_2}$ and $\mathbf{b}_{f_2}$ are the weights and biases of the second forward layer, with σ representing the activation function. Typically, $\mathbf{W}_{f_1} \in \mathbb{R}^{d \times 4d}$ and $\mathbf{W}_{f_2} \in \mathbb{R}^{4d \times d}$ have large parameter sizes. By replacing the fully connected layers in the PWFF network with TTL, the calculation becomes:

$$\mathbf{O}^{FF_1} = TTL(\mathcal{G}^*_{f_1}, \mathbf{O}^{MHA}, \mathcal{B}_{f_1}) \tag{11}$$

$$\mathbf{O}^{FF_2} = TTL(\mathcal{G}^*_{f_2}, \sigma(TTL(\mathcal{G}^*_{f_1}, \mathbf{O}^{MHA}, \mathcal{B}_{f_1})), \mathcal{B}_{f_2}) \tag{12}$$

where $\mathbf{O}^{FF_1}$ is the output of the first forward layer, and $\mathbf{O}^{FF_2}$ is the output of the second forward layer. $\mathcal{G}^*_{f_1}$ and $\mathcal{G}^*_{f_1}$ represent the tensor train factors in the forward network.

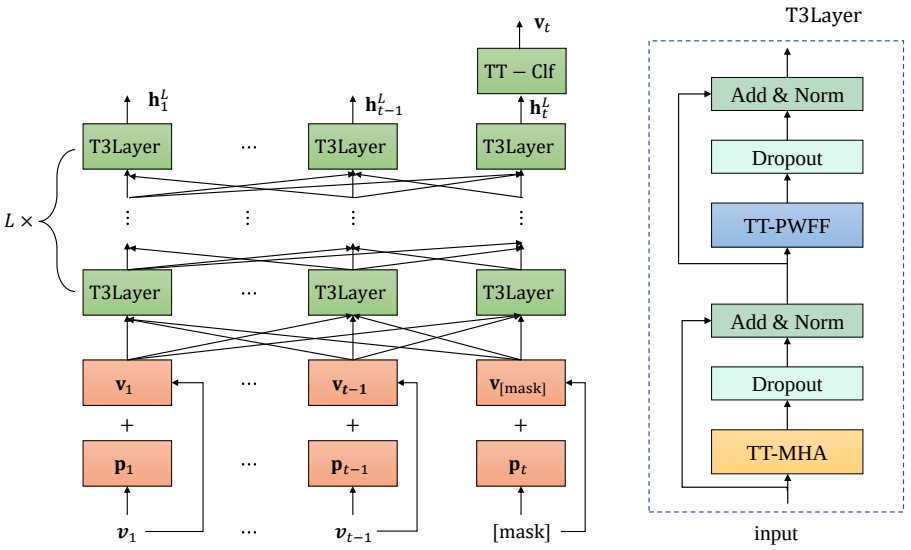
4.4. Tensor Train Transformer for BERT4Rec

Next, the application of the proposed Tensor Train Transformer will be demonstrated based on the sequence recommendation model BERT4Rec. The model framework is shown in Figure 8, where the feature extraction layer consists of L Tensor Train Transformer layers (denoted as T3Layer). Figure 8 (b) shows the specific structure of T3Layer. Compared to the original BERT4Rec model, this model uses Tensor Train Transformer and employs tensor train layers in the classification layer, which significantly reduces the number of parameters.

Following Sun et al. (2019), let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ represent the set of users, $V = \{v_1, v_2, \ldots, v_{|V|}\}$ represent the set of items, and list $S_u = \{v_1^{(u)}, \ldots, v_t^{(u)}, \ldots, v_{n_u}^{(u)}\}$ represent the interaction sequence of user $u \in U$ in chronological order, where $v_t^{(u)} \in V$ is the item that user u interacts with at time step t, and n_u is the length of user u's interaction sequence. Given the interaction history S_u, sequence recommendation aims to predict the item that user u will interact with at time step $n_u + 1$.

The model takes user interaction history as input, including fixed-length item embeddings and learnable position embeddings. For an item v_i, its input representation is obtained by adding its item embedding vector $v_i \in \mathbb{R}^{d_e}$ and position embedding vector $p_i \in \mathbb{R}^{d_e}$. Through multiple layers of stacked T3Layer and the final TT-Clf classifier, the model predicts items.

In the T3Layer section, a proposed Tensor Train Transformer is used to improve the MHA and PWFF networks of the Transformer, as shown in Figure 8



(a) T3-BERT4Rec

(b) Tensor Train Transformer layer

Figure 8: Architecture diagram of BERT4Rec model based on Tensor Train Transformer.

(b). Multiple T3Layer process the input and obtain hidden representations $h_i^l \in \mathbb{R}^d$ for each layer l, where $h_i^0 = v_i + p_i$. Due to the computation of attention functions for all positions simultaneously in practice, h_i^l is stacked into a matrix $\mathbf{H}^l \in \mathbb{R}^{t\times d}$.

$$\begin{aligned} TTMHA(\mathbf{H}^l) &= TTL(\mathcal{G}_a^*, (head_1, \ldots, head_h), \mathcal{B}_a) \\ head_i &= Attention(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \end{aligned} \tag{13}$$

$$\begin{cases} Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \\ \mathbf{Q} = TTL(\mathcal{G}_Q^*, \mathbf{H}^l, \mathcal{B}_Q) \\ \mathbf{K} = TTL(\mathcal{G}_K^*, \mathbf{H}^l, \mathcal{B}_K) \\ \mathbf{V} = TTL(\mathcal{G}_V^*, \mathbf{H}^l, \mathcal{B}_V) \end{cases} \tag{14}$$

To obtain nonlinear characteristics and interactions across different dimensions, the output matrix $\mathbf{H}^l$ from the attention sublayer is further fed into the TT-PWFF network, where a Gaussian Error Linear Unit (GELU) activation

function is employed. The calculation formula is as follows:

$$\begin{aligned} TTPWFF(\mathbf{H}^l) &= TTL(\mathcal{G}^*_{f_2}, GELU(TTL(\mathcal{G}_{f_1})^*, \mathbf{H}^l, \mathcal{B}_{f_1}), \mathcal{B}_{f_2}) \\ GELU(x) &= x\Phi(x) \end{aligned} \tag{15}$$

The overall computation of the T3Layer layer is as follows:

$$\mathbf{H}^l = T3Layer(\mathbf{H}^{l-1}), l \in [1, \ldots, L] \tag{16}$$

$$T3Layer(\mathbf{H}^{l-1}) = LN(\mathbf{A}^{l-1} + Dropout(TTPWFF(\mathbf{A}^{l-1})) \tag{17}$$

$$\mathbf{A}^{l-1} = LN(\mathbf{H}^{l-1} + Dropout(TTMHA(\mathbf{H}^{l-1}))) \tag{18}$$

After passing through L layers that can interact hierarchically, we obtain the final output $\mathbf{H}^l$ for all items in the input sequence. Then, the linear layer in the prediction layer that outputs the distribution of items is replaced with a tensor train layer, which results in the final output layer.

$$P(v) = softmax(GELU(TTL(\mathcal{G}^*_P, \mathbf{h}^L_t, \mathcal{B}_P))\mathbf{E}^\top + \mathbf{b}^O) \tag{19}$$

where $\mathcal{G}^*_P$ represents a learnable mapping tensor factor, while $\mathcal{B}_P$ and $\mathbf{b}^O$ denote biases. Additionally, $\mathbf{E} \in \mathbb{R}^{|V| \times d}$ is the embedding matrix for the item set V, which is shared between the input and output layers to alleviate overfitting.

It is essential to predefine the structure of tensor train layers before the model is trained. We determine an appropriate number of tensor train factors and predefine the tensor train layers. Subsequently, model parameter training is conducted using the training dataset, in accordance with the procedure outlined in Algorithm 1. This algorithm provides an end-to-end model training process, with the parameters of the tensor train layers being randomly initialized and the loss function being chosen from the original model.

4.5. Space and computational complexity analysis

For a single linear layer $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{I \times J}$ and $\mathbf{b} \in \mathbb{R}^J$, the storage requirement is $I \times J + J$ and the computational complexity is $O(IJ)$.

Algorithm 1 Overall training process of T3-BERT4Rec.

Input:

Embedding matrix $\mathbf{E}$ of item set $\mathcal{V}$;

The number of attention heads h;

The number of Transformer layers L;

The number of tensor train factors N;

The ranks of tensor train factors $\{R_n\}_{n=1}^{N+1}$;

Initialized parameter $\mathcal{G}^*$;

Output:

The final embedding matrix $\mathbf{E}$;

According to quantity N and ranks $\{R_n\}_{n=1}^{N+1}$ predefined tensor train layers;

for $l = 1 \dots L$ **do**

  Calculate TTMHA layer;

  /* As shown in Eq. 13 and Eq. 14 */

  Calculate TTPWFF layer;

  /* As shown in Eq. 15 */

  Calculate T3-Layer;

  /* As shown in Eq. 16, 17 and 18 */

  Calculate Output layer;

  /* As shown in Eq. 19 */

end for

Minimize the loss function of the original model;

Back propagation and update parameters $\mathbf{E}$, $\mathcal{G}^*$;

return $\mathbf{E}$;

For a TTL, the storage requirement is $\sum_{n=1}^{N} R_n I_n J_n R_{n+1} + J$, and the computational complexity is $O(N\tilde{I}\tilde{R}^2\tilde{J}^N) = O(N\tilde{I}\tilde{R}^2 J)$, where $\tilde{I} = \max_{n\in[1,\dots,N]} I_n$, $\tilde{J} = \max_{n\in[1,\dots,N]} J_n$, and $\tilde{R} = \max_{n\in[1,\dots,N]} R_n$.

In Transformer, when $d_{ff} = 4d$, it will occupy $12L(d^2 + d)$ and its computational complexity is $O(Ld^2)$. Tensor Train Transformer, on the other hand, will occupy $12L(\sum_{n=1}^{N} R'_n J_n J_n R'_{n+1} + d)$, which is significantly smaller than the

original. Its computation complexity is $O(N\tilde{J}\tilde{R'}^2\tilde{J}^N) = O(N\tilde{J}\tilde{R'}^2 d)$, where $\tilde{R'}$ represents $\max_{n\in[1,\ldots,N]} R'_n$.

At the output layer, where d_e represents the embedding dimension of the items, the space required is $dd_e + d_e$, resulting in a computational complexity equivalent to that of a single fully connected layer, i.e., $O(dd_e)$. The space and computational complexities of each component are shown in Table 2. Since $d > d_e \gg I_n, J_n, R_n, R'_n$, the Tensor Train Transformer can effectively compress sequence recommendation models and exhibit excellent reusability, as detailed in the experimental section.

Table 2: Comparison of spatial and computational complexity between BERT4Rec and T3-BERT4Rec

Modules	Space		Computational	
	BERT4Rec	T3-BERT4Rec	BERT4Rec	T3-BERT4Rec
MHA	$4L(d^2 + d)$	$4L(\sum_{n=1}^{N} R'_n J_n J_n R'_{n+1} + d)$	$O(Ld^2)$	$O(LN\tilde{J}\tilde{R'}^2 d)$
PWFF	$8L(d^2 + d)$	$8L(\sum_{n=1}^{N} R'_n J_n J_n R'_{n+1} + d)$	$O(Ld^2)$	$O(LN\tilde{J}\tilde{R'}^2 d)$
Output Layer	$dd_e + d_e$	$\sum_{n=1}^{N} R_n I_n J_n R_{n+1} + d_e$	$O(dd_e)$	$O(N\tilde{J}\tilde{R}^2 d_e)$

5. Experiments

In this section, we first describe the experimental setup, including the dataset, baseline models, implementation details, and evaluation metrics. In order to clearly reflect the research objectives, we design the experiment by answering the following research questions. Then, we present the experimental results and answer the research questions raised by analyzing and discussing the results.

- **RQ1**: Is there an effective reduction in the size of typical sequence recommendation models, such as SASRec and BERT4Rec? If so, does Tensor Train Transformer have an advantage in terms of accuracy?
- **RQ2**: What is the impact of different modules on the efficiency and accuracy of T3SRS?

- **RQ3**: Does Tensor Train Transformer have a certain degree of reusability? Can it be applied in other sequence recommendation models, such as TiSASRec, SSE-PT, LightSANs, and CORE models?
- **RQ4**: What is the effect of the ranks of the tensor train factor on the model?
- **RQ5**: Is our proposed model competitive compared to other popular model compression approaches?

5.1. Experimental setup

Datasets. The proposed model is evaluated on three real-world recommendation datasets: Steam[2], MovieLens-1M[3], and MovieLens-20M[4]. The Steam dataset is collected from the large online video game architecture. MovieLens-1M (ML-1M) and MovieLens-20M (ML-20M) are two widely used baseline datasets. The data preprocessing follows the procedure in Seol et al. (2022), which removes items with interaction counts less than or equal to 5. Table 3 presents the statistical information of the preprocessed datasets. For each user, we use the last item as the test set, the second most frequent item as the validation set, and the remaining items as the training set.

Table 3: Statistics of datasets after data preprocessing.

Datasets	#users	#items	#actions	Density
Steam	6,330	4,331	49,163	0.18%
ML-1M	1,196	3,327	158,498	3.98%
ML-20M	23,404	12,239	1,981,866	0.69%

Baseline models. To evaluate the effectiveness of our proposed model, we selected multiple SRSs with a Transformer structure and replaced them with the

[2]https://cseweb.ucsd.edu/∼jmcauley/datasets.html#steam_data
[3]https://grouplens.org/datasets/movielens/1m/
[4]https://grouplens.org/datasets/movielens/20m/

Tensor Train Transformer. We then compared the recommendation performance and model size between the replaced model and the original one.

- GRU4Rec (Hidasi et al., 2016a): It uses an RNN-based approach by modeling the whole session for session-based recommendations.
- NextItNet (Yuan et al., 2019): A simple and effective generation model, with a network structure consisting of multiple convolutional layers stacked together, can effectively increase receptive fields without relying on pooling operations.
- SASRec (Kang & McAuley, 2018): It uses a Transformer model to capture user sequence behavior. The model includes an item embedding layer, a position encoding layer, multiple head attention layers, feed-forward layers, and an output layer.
- BERT4Rec (Sun et al., 2019): It also uses a Transformer structure to model user behavior sequences, similar to the SASRec model. To obtain bidirectional contextual information, the model is trained using a Cloze task and achieves excellent performance in sequence recommendation.

Implementation details. We implemented the Tensor Train Transformer using PyTorch and replaced the Transformer structure of existing SRSs with it. All parameters were initialized randomly, and the tensor train factors were initialized using normal distribution $\mathcal{N}(0, 0.1)$. The model was trained using the Adam optimizer (learning rate 1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.999$) on an RTX A5000 GPU, and the loss function used was the same as the original model's loss function. All hyper-parameters refer to the parameters of the original papers and then use grid search to record the best results. The hyper-parameters for the data are shown in Table 4, and the embedding size for all datasets was set to 512. Due to the need for the tensor train layers to be predefined based on the number of factors, we uniformly set the number of factors to 3. Due to the TT ranks being a more sensitive hyper-parameter, we carefully selected the appropriate

ranks and conducted experiments on their sensitivity. The experimental results can be found in Section 5.5.

All source code, including model implementation, dataset preprocessing, compression, and recommendation experiments is released to the public in https://github.com/jlzhaoteam/T3SRS.

Table 4: Hyper-parameters of the dataset.

Hyper-parameters	Datasets		
	Steam	ML-1M	ML-20M
Maximum sequence length	15	200	100
Hidden dim	256	256	256
Mask probability	0.4	0.2	0.2
Number of attention head	2	2	4
Dropout rate	0.2	0.2	0.2
Batch size	256	64	64

Evaluation metrics. Recall@10 and NDCG@10 are commonly employed as evaluation metrics for recommendation systems, while the size of the model is expressed in terms of trainable parameters. In terms of sampling, 100 items with no prior user interactions are randomly selected as negative samples for each user. Following Seol et al. (2022), random sampling introduces bias, therefore, negative sampling based on item popularity is implemented.

*5.2. Overall performance comparison (**RQ1**)*

To answer research question RQ1 and obtain objective and comprehensive experimental results, we selected GRU4Rec, NextItNet, SASRec, and BERT4Rec as models to evaluate recommendation performance. To gain a deeper understanding of the impact of the Tensor Train Transformer on model space occupation, we documented the parameter counts of SASRec and BERT4Rec and replaced their original Transformer layers with the Tensor Train

Transformer. We conducted negative sampling based on item popularity, compared the Recall@10 and NDCG@10 metrics. The experimental results are presented in Table 5.

Table 5: Overall performance comparison of SASRec and BERT4Rec. Bolded text indicates a relatively better result. Improvements over baselines are statistically significant with $p < 0.05$.

Dataset	Model	R@10 Pop.	N@10 Pop.	Para. all	Para. w/o embed.	Ratio of compressible Para.	Compression ratio	Ranks
Steam	MostPop	0.1053	0.1112	\	\	\	\	\
	GRU4Rec	0.3589	0.2845	1.1M	0.1M	\	\	\
	NextItNet	0.4266	0.3772	4.9M	3.9M	\	\	\
	SASRec	0.5523	0.5007	5.3M	4.2M	\	\	\
	BERT4Rec	0.5600	0.5196	5.3M	4.2M	\	\	\
	T3-SASRec	**0.5541**	**0.5049**	**2.1M**	**1.0M**	79.2%	76.2%	35
	T3-BERT4Rec	**0.5670**	**0.5251**	**1.9M**	**0.8M**	79.2%	80.9%	35
ML-1M	MostPop	0.0710	0.1268	\	\	\	\	\
	GRU4Rec	0.2493	0.1467	1.0M	0.1M	\	\	\
	NextItNet	0.3614	0.2521	4.8M	3.9M	\	\	\
	SASRec	0.4189	0.2674	4.9M	4.0M	\	\	\
	BERT4Rec	0.4983	0.3209	4.9M	4.0M	\	\	\
	T3-SASRec	**0.4233**	**0.2783**	**1.7M**	**0.8M**	81.6%	80.0%	35
	T3-BERT4Rec	**0.5702**	**0.3758**	**1.5M**	**0.6M**	81.6%	85.0%	30
ML-20M	MostPop	0.0828	0.1477	\	\	\	\	\
	GRU4Rec	0.1903	0.1746	3.2M	0.1M	\	\	\
	NextItNet	0.3559	0.2310	7.0M	3.9M	\	\	\
	SASRec	0.4370	0.2839	9.4M	6.3M	\	\	\
	BERT4Rec	0.4548	0.2909	9.4M	6.3M	\	\	\
	T3-SASRec	**0.4517**	**0.2944**	**4.4M**	**1.2M**	67.0%	80.9%	35
	T3-BERT4Rec	**0.4678**	**0.3004**	**4.3M**	**1.1M**	67.0%	82.5%	40

Table 5 presents a comprehensive comparison of the overall performance and parameter reduction of two Transformer-based sequence recommendation models, SASRec and BERT4Rec, and their counterparts with the Tensor Train Transformer layer, T3-model, on three datasets (Steam, ML-1M, ML-20M). Upon observing Table 5, sequence recommendation models based on the Transformer architecture significantly outperform GRU4Rec and NextItNet in terms of performance. However, these models also have a comparatively larger number of parameters. Consequently, our objective is to enhance the space efficiency of these models without compromising their performance.

The results indicate that T3-model outperforms the original models on all datasets, improving Recall@10 and NDCG@10. For example, on the ML-1M

dataset, T3-BERT4Rec model showed a 14.4% improvement in Recall@10 and a 17.1% improvement in NDCG@10 compared to the BERT4Rec model. The proposed model achieves better recommendation performance is not an uncommon phenomenon (Pan et al., 2019; Yang et al., 2017; Ye et al., 2020). Our analysis attributes this advantage to the tensor train network being adeptly crafted to concisely encapsulate the weight matrices within the Transformer layers. Leveraging this, the model can reduce the number of parameters without significant information loss. In addition, as the number of parameters decreases, the model is less prone to overfitting and avoids the impact of over-parameterization, especially on sparse sequence recommendation datasets. This demonstrates that Tensor Train Transformer can effectively compress and reduce the parameters of the original sequential recommender system while maintaining or even improving the performance.

Furthermore, T3-model significantly reduces the number of parameters, with compression rates ranging from 76.2% to 85.0%, and a high proportion of compressible parameters (67.0% to 81.6%). For instance, on the Steam dataset, T3-SASRec model reduced the total number of parameters by 60.4%, the number of parameters excluding the embedding layer by 76.2%, and compressible parameters accounted for 79.2% of the total parameter count. Variations in the dataset can influence the changes in the compression rate. This is attributable to the necessity of fine-tuning hyper-parameters, such as maximum sequence length and hidden dimensions, to achieve optimal recommendation performance, which, in turn, influences the model's parameter count. Notably, the ranks are pivotal in tensor train network-based models, substantially impacting the compression rate. Properly calibrating the ranks for each dataset is imperative to guarantee the peak performance of the tensor train networks, which subsequently affects the overall parameter reduction ratio. This presents that Tensor Train Transformer can significantly reduce the model's space usage, saving storage resources.

Additionally, the ranks used vary across different datasets, typically ranging from 30 to 40. For example, the T3-BERT4Rec model used ranks of 40 on the

ML-20M dataset and ranks of 30 on the ML-1M dataset. A diminutive rank denotes fewer parameters and, consequently, a diminished reservoir of retained information. This indicates that Tensor Train Transformer can adjust the ranks used in the tensor train network based on the characteristics of different datasets and tasks to balance the model compression and performance.

In light of the detailed results presented above, it becomes evident that the integration of the Tensor Train Transformer layer into Transformer-based sequence recommendation models has substantial benefits. Our T3-model, as showcased, consistently surpasses the likes of SASRec and BERT4Rec in terms of performance across the Steam, ML-1M, and ML-20M datasets. Beyond performance, the parameter reduction attributes of the T3-model are noteworthy, achieving impressive compression rates between 76.2% and 85.0%. Moreover, the variable tensor train ranks employed in different datasets underscore the model's ability to effectively adapt and balance between model compression and performance.

*5.3. Alation study (**RQ2**)*

To answer research question QR2, we conducted ablation tests on three aspects: (1) multi-head attention layer, (2) position-wise feed-forward network, and (3) output layer. Table 6 displays the experimental results of different settings. The baseline is the original BERT4Rec model, while model (a) T3-BERT4Rec uses the Tensor Train Transformer in the BERT4Rec model. Model (b) T3-BERT4Rec-MHA uses the original multi-head attention layer in the T3-BERT4Rec, model (c) T3-BERT4Rec-FF uses the original position-wise feed-forward network in the T3-BERT4Rec model, and model (d) T3-BERT4Rec-O uses the original output layer in the T3-BERT4Rec model.

The table indicates that, on all datasets, whether using random or popularity-based sampling, models that use the Tensor Train Transformer outperform the baseline model in terms of Recall@10 and NDCG@10 metrics, while also reducing the number of parameters to some extent. This suggests that the Tensor Train Transformer has advantages in improving sequence recommenda-

Table 6: Results of ablation experiment. Bolded text indicates the best result, and underlined text indicates the second-best result.

Dataset	Model	R@10 Ran.	N@10 Ran.	R@10 Pop.	N@10 Pop.	Para. w/o embed.
Steam	Baseline	0.7887	0.6815	0.56	0.5196	4.2M
	(a)	0.7898	0.6864	0.567	0.5251	**0.8M**
	(b)	**0.799**	**0.6948**	0.5756	0.5264	1.5M
	(c)	0.7943	0.6893	**0.581**	**0.5382**	2.7M
	(d)	0.7898	0.6864	0.56	0.5251	1.7M
ML-1M	Baseline	0.7299	0.5031	0.4983	0.3209	4.0M
	(a)	0.7391	0.5268	0.5489	0.3527	**0.6M**
	(b)	0.7357	0.5093	0.5259	0.3485	1.5M
	(c)	0.7257	0.5067	0.5075	0.3487	2.7M
	(d)	**0.7441**	**0.5382**	**0.5568**	**0.3659**	1.4M
ML-20M	Baseline	0.8972	0.6931	0.4548	0.2909	6.3M
	(a)	0.9054	0.6933	0.4678	0.3004	**1.1M**
	(b)	0.9042	0.6947	**0.4721**	**0.3098**	1.8M
	(c)	0.9032	0.6941	0.4683	0.3043	3.1M
	(d)	**0.9059**	**0.6956**	0.4598	0.2967	3.9M

tion performance and reducing parameter count. Additionally, the table reveals that, among all variant models, model (a) has the least number of parameters. On datasets with longer user interaction sequences, such as ML-20M, the output layer will have a large number of parameters, whereas on datasets with shorter sequences, such as ML-1M and Steam, the position-wise feed-forward network has the most parameters of all components.

*5.4. Adaptive experiments (**RQ3**)*

To answer research question RQ3, we evaluated the reusability of the Tensor Train Transformer model by selecting TiSASRec, LightSANs, SSE-PT, and CORE models, which have more complex structures, as baseline models, and conducted experiments on the RecBole (Zhao et al., 2021) framework. The experimental results are shown in Table 7.

- SSE-PT (Wu et al., 2020): It proposes a personalized Transformer model for recommendations, as the SASRec model cannot provide personalized

recommendations. The model uses stochastic shared embedding to regularize randomly replaced user and item embeddings.

- TiSASRec (Li et al., 2020): It considers the time interval between sequences and proposes a time interval-aware attention model based on Transformer.

- LightSANs (Fan et al., 2021): It introduces two linear mappings to scale user historical sequences proportionally in the multi-head attention layer, using latent interests as feature representation. In addition, the position encoding is decoupled to obtain more accurate item order relationships.

- CORE (Hou et al., 2022): It addresses the problem that session embeddings and item embeddings are not in the same representation space by designing a representation-consistent encoder, where the Transformer is used as the encoder and the linear combination of input item embeddings is used as the session embedding.

Observing Table 7, it can be seen that the Tensor Train Transformer can be applied in multiple complex SRSs and achieve similar recommendation results to the original models. In terms of parameter count, it can still be effective in reducing the parameters of the Transformer structure, achieving compression rates of 69.91%-73.91%. Therefore, the Tensor Train Transformer has good applicability.

*5.5. Experiments on tensor train ranks (**RQ4**)*

To answer RQ4, we conducted a series of experiments to explore the relationship between sequence recommendation results and model parameters. Specifically, we conducted experiments with different TT ranks, ranging from 5 to 50 in step of 5, to gain a deeper understanding of the effect of TT ranks on sequence recommendation and to find the optimal TT ranks parameter value for better recommendation performance. The experimental results are shown in Figure 9.

Table 7: Results of more complex SRSs on Steam and ML-1M.

Dataset	Model	R@10 Ran.	N@10 Ran.	Para. all	Para. w/o embed.	Compression ratio
Steam	TiSASRec	0.2470	0.1962	1.12 M	100.10 K	/
	T3-TiSASRec	**0.2491**	**0.1984**	**1.05 M**	**26.11 K**	73.91%
	LightSANs	0.2554	0.2012	1.11 M	118.27 K	/
	T3-LightSANs	**0.2572**	**0.2037**	**1.03 M**	**35.58 K**	69.91%
	SSE-PT	0.2472	0.2057	329.90 M	265.22 K	/
	T3-SSE-PT	**0.2516**	**0.2095**	**329.71 M**	**72.19 K**	72.78%
	CORE	0.2337	0.1796	1.09 M	100.16 K	/
	T3-CORE	**0.2335**	**0.1812**	**1.01 M**	**26.18 K**	73.87%
ML-1M	TiSASRec	0.2205	0.1126	0.37 M	100.10 K	/
	T3-TiSASRec	**0.2234**	**0.1142**	**0.30 M**	**26.11 K**	73.91%
	LightSANs	0.2682	0.1476	0.35 M	118.27 K	/
	T3-LightSANs	**0.2719**	**0.1493**	**0.27 M**	**35.58 K**	69.91%
	SSE-PT	**0.2657**	**0.1432**	1.28 M	265.22 K	/
	T3-SSE-PT	0.2442	0.1219	**1.08 M**	**72.19 K**	72.78%
	CORE	0.1512	0.0681	0.34 M	100.16 K	/
	T3-CORE	**0.1537**	**0.0711**	**0.30 M**	**26.18 K**	73.87%

From Figure 9, it can be observed that the model's recommendation performance improves significantly when the tensor ranks increase from 5 to 20. However, when the ranks continues to increase to 35, the model's performance begins to decline. Therefore, larger tensor ranks can usually improve the model's performance, but they can also make the model more complex, leading to increased space consumption.

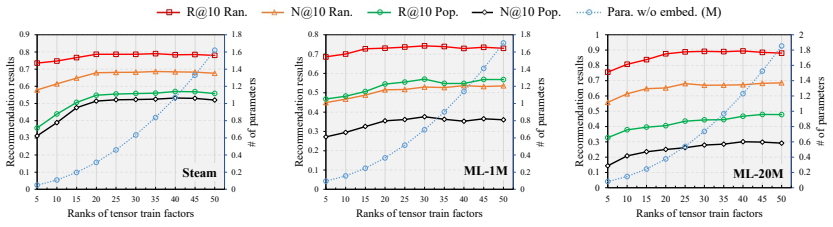


Figure 9: Effect of TT ranks on sequence recommendation.

*5.6. Comparison with other compression approaches (**RQ5**)*

In order to comprehensively analyze the advantages and disadvantages of the model, we chose other popular model compression approaches in the recommendation field for comparison. Since there is no compression method for the Transformer-based SRS, we selected several compression approaches for the Transformer to analyze their compression performance.

- TT-Rec (Yin et al., 2021): A deep learning recommendation model using Tensor Train Decomposition.
- CpRec (Sun et al., 2020): A deep convolution model using block-wise adaptive decomposition and parameter sharing for sequence recommendation tasks.
- BTD-Transformer (Ma et al., 2019): A novel self-attention model with Block-Term Tensor Decomposition for natural language processing tasks.
- TT-ViT (Pham Minh et al., 2022): A tensor train decomposition-based Vision Transformer for image classification tasks.
- Hypoformer (Li et al., 2022): A Transformer model using Hybrid Tensor Train Decomposition for neural machine translation tasks.
- Tucker-Transformer (Wang et al., 2022a): A Pre-trained language model using Tucker decomposition, matrix factorization, and TT decomposition.

These selected models should be tested according to the hyper-parameters set in the original papers as much as possible. Because BTD transformer, Hypoformer, and Tucker-Transformer are methods in the field of natural language processing, sequence recommendation task experiments are not carried out.

Compared with the compression method of the Transformer model, T3SRS compresses the network MHA, PWFF, and output layers, showing a larger compression range and smaller spatial complexity. From the perspective of compressibility, the BTD-Transformer is the smallest and can only compress MHA

Table 8: Comparison of compression methods with Transformer model. Note for clarity only the space complexity in the Transformer layer is shown in the Space Complexity Column.

Model	Compressed module	Space complexity
T3SRS	MHA, PWFF, Output layer	$12L(Nd^{\frac{2}{N}}R^2)$
BTD-Transformer	MHA	$8Ld^2 + 8LdR$
TT-ViT	MHA	$8Ld^2 + 4LNd^{\frac{2}{N}}R^2$
Hypoformer	Embedding layer, MHA, PWFF	$12L(\alpha d^2 + N(1-\alpha)^{\frac{1}{N}}d^{\frac{2}{N}}R^2)$
Tucker-Transformer	MHA, PWFF	$lR^2 + 12Ll + 2dR$

modules, followed by TT-ViT, Tucker-Transformer, and the compressibility of Hypoformer and TSRs models is the largest. Let the space complexity of the original Transformer layer be $12ld^2$, l is the number of layers, D is the parameter dimension, and R is the rank of tensor decomposition. The space complexity of Hypoformer is notably higher than that of T3SRS, primarily attributed to the fact that Hypoformer retains a portion of parameters, specifically $\alpha * 100\%$ [5], uncompressed to ensure accuracy. Regarding space complexity, the Tucker-Transformer model has the lowest space complexity, followed by the T3SRS model, while the BTD-transformer model has the highest space complexity. Tucker-Transformer combines parameters of the Transformer layer into tensors, then decomposes them using Tucker decomposition, matrix factorization, TT decomposition, or even discards some parameters. Therefore, the compression ratio is the highest among these models. However, the model will suffer a huge loss of accuracy [6], forcing the need for other methods to maintain accuracy.

Comparing with TT-Rec and CpRec in the recommendation domain, it can be found that the T3SRS model presents higher recommendation performance and high compression rate. TT-Rec acts on the Embedding layer, which accounts for about 20% of all parameters. In Yin et al. (2021), the targeted model of TT-Rec had an embedding layer with a dimension of 16 and showed good

[5] Parameter α is a tunable hyper-parameter within the range of 0 to 1.

[6] On SST-2, the accuracy rate drops from 93.1 % to 49.9%.

Table 9: comparison with model compression techniques within the domain of recommendation. Bolded text indicates a relatively better result.

Dataset	Model	R@10 Pop.	N@10 Pop.	Para. all
Steam	BERT4Rec	0.5600	0.5196	5.3M
	T3-BERT4Rec	**0.5670**	**0.5251**	**1.9M**
	TT-Rec	0.5331	0.5074	4.5M
	CpRec	0.4835	0.4584	4.1M
ML-1M	BERT4Rec	0.4983	0.3209	4.9M
	T3-BERT4Rec	**0.5702**	**0.3758**	**1.5M**
	TT-Rec	0.4814	0.3168	4.2M
	CpRec	0.4727	0.2979	3.9M

performance. However, the dimension of the embedding layer in the current case has been increased to 512, and TT-Rec faces the challenge of setting the tensor rank to find the appropriate TT rank. In contrast, TSRS does not compress the embedding layer and does not have this concern. Therefore, T3RSR shows more advantages and outperforms TT-Rec in recommendation performance and data compression capability. Although the CpRec model compresses both the Embedding layer and the intermediate layer, since the target model of the CpRec model is NextItNet (Yuan et al., 2019), the intermediate layer is a multilayer convolutional layer superposition, the number of parameters is higher than that of T3-BERT4Rec. In summary, the T3SRS model compresses the Transformer layers and output layer, and retains the embedding layer, resulting in less information loss and better recommendation performance.

5.7. Model properties analysis

5.7.1. Convergence

In order to study the convergence of the algorithm 1, we tested and verified the convergence to the compressed model through numerical experiments. In Figure. 10, we display the curve of Algorithm 1 with respect to iteration on the Steam dataset. We can observe the numerical convergence of the algorithm.

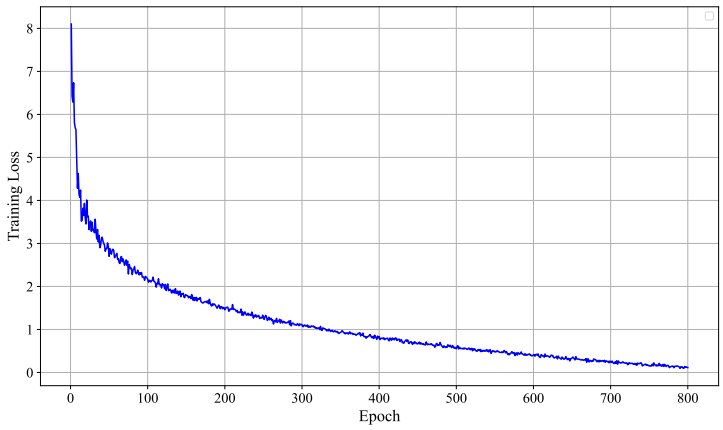


Figure 10: Training loss of T3-BERT4Rec on Steam.

5.7.2. Generalizability

In Section 5.4, we tested the applicability of the T3SRS model to be able to be applied to different sequence recommendation models, however, the generalization ability in the face of different datasets is still questionable. We validate it on TikTok [7] dataset and Beauty [8] dataset using T3-BERT4Rec. The hyper-parameters are strictly following Sun et al. (2019) and Sun et al. (2020). Table 10 shows that the recommendation performance and compression ratio of BERT4Rec and T3-BERT4Rec. We can observe that the T3SRS model can still be effective on different datasets, proving its good generalization.

Table 10: Experimental results on TikTok dataset and Beauty dataset.

Dataset	Model	R@10 Pop.	N@10 Pop.	Para. all	Ranks
TikTok	BERT4Rec	0.4285	0.2764	46.1M	\
	T3-BERT4Rec	0.4293	0.2853	42.6M	40
Beauty	BERT4Rec	0.2197	0.1727	5.2M	\
	T3-BERT4Rec	0.2214	0.1933	3.5M	35

[7] https://www.tiktok.com/en/

[8] http://jmcauley.ucsd.edu/data/amazon/

6. Conclusion

This paper aims to address the issues of over-parametrization and overfitting in Transformer-based SRSs by proposing a Tensor Train-based Transformer model. The model introduces the tensor train network and proposes a tensor train layer to fit weight matrices with lower space complexity. Based on this, a TTL-based MHA module and TTL-based PWFF network are designed and implemented, and parameter compression for multiple modules is achieved by compressing the output layer. Experimental results show that the proposed model can effectively compress parameters of the sequential recommender system with a compression rate of 76.2%-85.0% while ensuring recommendation performance. Furthermore, substitution experiments on multiple models demonstrate the model's good applicability, providing new ideas and methods for compressing SRSs.

Though the model exhibits commendable compression rates and recommendation efficacy, the design of the tensor train network profoundly impacts the outcomes. Both the number of factors, N, and the TT ranks, R_n, play pivotal roles as parameters. In future work, we intend to delve into heuristic optimization techniques(Gao et al., 2022), identifying promising hyper-parameter configurations that can be consistently applied and fine-tuned across varied datasets or scenarios, thereby potentially mitigating experimentation's computational strain and temporal demands.

Additionally, while the T3SRS model performs exceptionally well in the domain of Transformer-based SRSs, its specific design and optimization make it challenging to apply directly to other neural network architectures. The primary reason is that our approach fully leverages the multi-head attention and position-wise feed-forward network intrinsic to the Transformer structure, making it suited for Transformer-based SRSs. In future work, we will explore ways to modify or adapt this method to enhance its universality.

List of abbreviations

The abbreviations covered in the article and their full names are shown in Table 11.

Table 11: List of abbreviations and their full names.

Abbreviation	Full name
SRS	sequential recommender system
TT	tensor train
TTL	tensor train layer
RS	recommender system
RNN	recurrent neural network
CNN	convolutional neural network
MHA	multi-head attention
PWFF	position-wise feed-forward
TT-MHA	multi-head attention with tensor train layers
TT-PWFF	position-wise feed-forward with tensor train layers

Acknowledgements

This paper is supported by Natural Science Foundation of Shandong Province (No.ZR2021MF104, No.ZR2021MF113), National Natural Science Foundation (No. 62072288), Key R&D Projects of Qingdao Science and Technology Plan (No.21-1-2-19-xx), Qingdao West Coast New District Science and Technology Plan (No.2020-1-6).

References

Aggarwal, V., Wang, W., Eriksson, B., Sun, Y., & Wang, W. (2018). Wide Compression: Tensor Ring Nets. In *2018 IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition (pp. 9329–9338). doi:10.1109/CVPR.2018.00972 27.

Bartol, T. M., Bromer, C., Kinney, J. P., Chirillo, M., Bourne, J. N., Harris, K. M., & Sejnowski, T. J. (2015). Hippocampal spine head sizes are highly precise. *bioRxiv*, . 21.

Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., & Zha, H. (2018). Sequential Recommendation with User Memory Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (pp. 108–116). Marina Del Rey CA USA: ACM. doi:10.1145/3159652.3159668 3.

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *ArXiv*, *abs/1904.10509*. 29.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, L. (2019). Universal transformers. In *International Conference on Learning Representations*. 23.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, *abs/1810.04805*. 16.

Du, Y., Peng, Z., Niu, J., & Yan, J. (2022). A unified hierarchical attention framework for sequential recommendation by fusing long and short-term preferences. *Expert Systems with Applications*, *201*, 117102. doi:10.1016/j.eswa.2022.117102. 13.

Fan, X., Liu, Z., Lian, J., Zhao, W. X., Xie, X., & Wen, J.-R. (2021). Lighter and Better: Low-Rank Decomposed Self-Attention Networks for Next-Item Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '21 (pp. 1733–1737). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3404835.3462978 20.

Gao, H., Li, Z., Yu, X., & Qiu, J. (2022). Hierarchical Multiobjective Heuristic for PCB Assembly Optimization in a Beam-Head Surface Mounter. *IEEE Transactions on Cybernetics*, *52*, 6911–6924. doi:10.1109/TCYB.2020.3040788.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016a). Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939.

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016b). Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 241–248). Boston Massachusetts USA: ACM. doi:10.1145/2959100.2959167 12.

Hou, Y., Hu, B., Zhang, Z., & Zhao, W. X. (2022). CORE: Simple and Effective Session-based Recommendation within Consistent Representation Space. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1796–1801). Madrid Spain: ACM. doi:10.1145/3477495.3531955 35.

Hrinchuk, O., Khrulkov, V., Mirvakhabova, L., Orlova, E., & Oseledets, I. (2020). Tensorized Embedding Layers for Efficient Model Compression. *arxiv*, *abs/1901.10787*. doi:10.48550/arXiv.1901.10787.

Jannach, D., & Ludewig, M. (2017). When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (pp. 306–310). Como Italy: ACM. doi:10.1145/3109859.3109872 1.

Kang, W.-C., & McAuley, J. (2018). Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 197–206). IEEE. 4.

Kolda, T. G., & Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, *51*, 455–500. doi:10.1137/07070111X. 34.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, *abs/1909.11942*. 22.

Li, J., Wang, Y., & McAuley, J. (2020). Time Interval Aware Self-Attention for Sequential Recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining* WSDM '20 (pp. 322–330). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3336191.3371786 18.

Li, S., Zhang, P., Gan, G., Lv, X., Wang, B., Wei, J., & Jiang, X. (2022). Hypoformer: Hybrid Decomposition Transformer for Edge-friendly Neural Machine Translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 7056–7068). Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. doi:10.18653/v1/2022.emnlp-main.475.

Lin, M., Zhang, Y., Li, Y., Chen, B., Chao, F., Wang, M., Li, S., Tian, Y., & Ji, R. (2022). 1xN Pattern for Pruning Convolutional Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (pp. 1–11). doi:10.1109/TPAMI.2022.3195774. 9.

Liu, H., Dai, Z., So, D., & Le, Q. V. (2021). Pay Attention to MLPs. In *Advances in Neural Information Processing Systems* (pp. 9204–9215). Curran Associates, Inc. volume 34. 19.

Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H. (2018). STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1831–1839). London United Kingdom: ACM. doi:10.1145/3219819.3219950 14.

Liu, X., Su, J., & Huang, F. (2022). Tuformer: Data-driven Design of Transformers for Improved Generalization or Efficiency. In *International Conference on Learning Representations*. 30.

Ma, X., Zhang, P., Zhang, S., Duan, N., Hou, Y., Zhou, M., & Song, D. (2019). A tensorized transformer for language modeling. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 32. 6.

Novikov, A., Podoprikhin, D., Osokin, A., & Vetrov, D. P. (2015). Tensorizing neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 28. 8.

Pan, Y., Xu, J., Wang, M., Ye, J., Wang, F., Bai, K., & Xu, Z. (2019). Compressing Recurrent Neural Networks with Tensor Ring for Action Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 4683–4690. doi:10.1609/aaai.v33i01.33014683. 7.

Pham Minh, H., Nguyen Xuan, N., & Tran Thai, S. (2022). TT-ViT: Vision Transformer Compression Using Tensor-Train Decomposition. In N. T. Nguyen, Y. Manolopoulos, R. Chbeir, A. Kozierkiewicz, & B. Trawiński (Eds.), *Computational Collective Intelligence* (pp. 755–767). Cham: Springer International Publishing volume 13501. doi:10.1007/978-3-031-16014-1_59.

Qiang, W., & Ji, Y. (2022). TP: tensor product layer to compress the neural network in deep learning. *Applied Intelligence*, *52*, 17133–17144. doi:10.1007/s10489-022-03260-6. 25.

Seol, J. J., Ko, Y., & Lee, S.-g. (2022). Exploiting Session Information in BERT-based Session-aware Sequential Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '22 (pp. 2639–2644). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3477495.3531910 36.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., &

Keutzer, K. (2020). Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*, 8815–8821. doi:10.1609/aaai.v34i05.6409. 10.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 1441–1450). Beijing China: ACM. doi:10.1145/3357384.3357895 5.

Sun, Y., Yuan, F., Yang, M., Wei, G., Zhao, Z., & Liu, D. (2020). A Generic Network Compression Framework for Sequential Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '20 (pp. 1299–1308). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3397271.3401125 33.

Tang, J., & Wang, K. (2018). Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (pp. 565–573). Marina Del Rey CA USA: ACM. doi:10.1145/3159652.3159656 2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 30. 15.

Wang, B., Ren, Y., Shang, L., Jiang, X., & Liu, Q. (2022a). Exploring extreme parameter compression for pre-trained language models. In *International Conference on Learning Representations*.

Wang, M., Pan, Y., Yang, X., Li, G., & Xu, Z. (2023). Tensor networks meet neural networks: A survey. *ArXiv*, *abs/2302.09019*. 24.

Wang, Y., Guo, W. G., & Yue, X. (2022b). Tensor decomposition to compress convolutional layers in deep learning. *IISE TRANSACTIONS*, *54*, 481–495. doi:10.1080/24725854.2021.1894514. 26.

Wu, L., Li, S., Hsieh, C.-J., & Sharpnack, J. (2020). SSE-PT: Sequential Recommendation Via Personalized Transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems* RecSys '20 (pp. 328–337). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3383313.3412258 17.

Xu, Y. L., Calvi, G. G., & Mandic, D. P. (2021). Tensor-Train Recurrent Neural Networks for Interpretable Multi-Way Financial Forecasting. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–5). Shenzhen, China: IEEE. doi:10.1109/IJCNN52387.2021.9534120 28.

Yang, Y., Krompass, D., & Tresp, V. (2017). Tensor-train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* ICML'17 (pp. 3891–3900). Sydney, NSW, Australia: JMLR.org.

Ye, J., Li, G., Chen, D., Yang, H., Zhe, S., & Xu, Z. (2020). Block-term tensor neural networks. *Neural Networks*, *130*, 11–21. doi:10.1016/j.neunet.2020.05.034.

Yin, C., Acun, B., Wu, C.-J., & Liu, X. (2021). TT-Rec: Tensor train compression for deep learning recommendation models. In A. Smola, A. Dimakis, & I. Stoica (Eds.), *Proceedings of Machine Learning and Systems* (pp. 448–462). volume 3. 32.

Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., & He, X. (2019). A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (pp. 582–590). Melbourne VIC Australia: ACM. doi:10.1145/3289600.3290975.

Zafrir, O., Boudoukh, G., Izsak, P., & Wasserblat, M. (2019). Q8bert: Quantized 8bit BERT. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. IEEE. doi:10.1109/emc2-nips53020.2019.00016 11.

Zhao, W. X., Mu, S., Hou, Y., Lin, Z., Chen, Y., Pan, X., Li, K., Lu, Y., Wang, H., Tian, C., Min, Y., Feng, Z., Fan, X., Chen, X., Wang, P., Ji, W., Li, Y., Wang, X., & Wen, J.-R. (2021). RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (pp. 4653–4664). Virtual Event Queensland Australia: ACM. doi:10.1145/3459637.3482016 37.