CoLA-Former: Graph Transformer Using Communal Linear Attention for Lightweight Sequential Recommendation

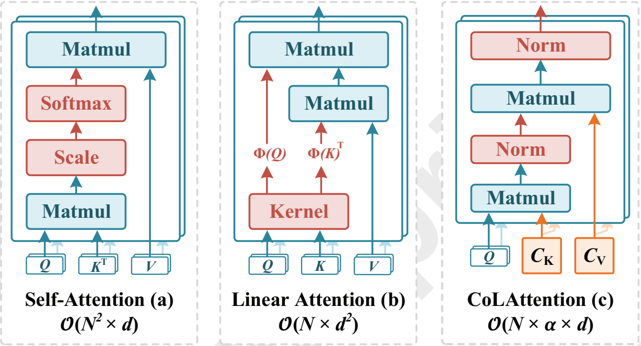
Zhongying Zhao[1], **Jinyu Zhang**[1], **Chuanxu Jia**[1], **Chao Li**[1*], **Yanwei Yu**[2], **Qingtian Zeng**[1*]
[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2] College of Computer Science and Technology, Ocean University of China, Qingdao 266400, China
zzysuin@163.com, jinyuz1996@outlook.com, chuanxujia@outlook.com, lichao@sdust.edu.cn, yuyanwei@ouc.edu.cn, qtzeng@163.com

Abstract

Graph Transformer has shown great promise in capturing the dynamics of user preferences for sequential recommendations. However, the self-attention mechanism within its structure is of quadratic complexity, posing challenges for deployment on devices with limited resources. To this end, we propose a Communal Linear Attention-enhanced Graph TransFormer for lightweight sequential recommendation, namely CoLA-Former. Specifically, we introduce a Communal Linear Attention (CoLAttention) mechanism. It utilizes low-rank yet reusable communal units to calculate the global correlations on sequential graphs. The weights from the units are also made communal across different training batches, enabling inter-batch global weighting. Moreover, we devise a low-rank approximation component. It utilizes weights distillation to reduce the scale of the trainable parameters in the Graph Transformer network. Extensive experimental results on three real-world datasets demonstrate that the proposed CoLA-Former significantly outperforms twelve state-of-the-art methods in accuracy and efficiency. The datasets and codes are available at https://github.com/ZZY-GraphMiningLab/CoLA_Former.

1 Introduction

User behaviors often exhibit continuity, manifesting as historical sequences [Ma *et al.*, 2024]. This promotes the research of Sequential Recommendation (SR) systems which aim to effectively deliver personalized products [Chen *et al.*, 2023], contents [Hu *et al.*, 2024], and services [Zhou *et al.*, 2020] to users. Among various SR methods, Graph-based Sequential Recommender systems (GSRs) excel in capturing intricate relationships between users and items in non-Euclidean spaces [Liu *et al.*, 2024b; Wang *et al.*, 2024]. Numerous advancements have been made to enhance GSRs, including GCN-based approaches [Chang *et al.*, 2021; Zhang *et al.*, 2023b], GAT-based techniques [Zhang *et al.*, 2023a; Wu *et al.*, 2023a], GCL-based methods [Zhang *et al.*, 2024a; Zhang *et al.*, 2022], and those based on Graph Transformers [Luo *et al.*, 2024; Xia *et al.*, 2023].



Self-Attention (a) $\mathcal{O}(N^2 \times d)$ **Linear Attention (b)** $\mathcal{O}(N \times d^2)$ **CoLAttention (c)** $\mathcal{O}(N \times \alpha \times d)$

Figure 1: Difference between Self-Attention, Linear Attention, and our proposed CoLAttention mechanism, where Q, K, and V denote the query, key, and value vectors, respectively; $\Phi(Q)$ and $\Phi(K)$ denote the vectors transformed by the nonlinear kernels; C_K and C_V are the communal units.

As one of the most representative GSRs, Graph Transformer-based methods have received considerable attention due to their capability of apprehending the deep and global correlations among nodes [Fan *et al.*, 2021]. However, the Self-Attention (SA) component within its encoder (as shown in Figure 1 (a)) brings quadratic complexity [Guo *et al.*, 2023]. Some researchers replaced the SA with Linear Attention (LA) [Katharopoulos *et al.*, 2020; Han *et al.*, 2023]. The LA substitutes the Softmax operation with custom kernel functions, allowing it to decouple the computation and change the order to compute $K^\top V$ first. This reduces the complexity from $\mathcal{O}(N^2 \times d)$ to $\mathcal{O}(N \times d^2)$ [Han *et al.*, 2023]. Nevertheless, these methods still face the following challenges:

(1) The recommendation accuracy decreased. In contrast to the Softmax, the kernel functions in LA do not incorporate nonlinear reweighting of the similarities between queries and keys. As a result, LA-based models struggle to accentuate the significant features within the attention map, resulting in a decreased prediction accuracy. Besides, these attentive calculations in Graph Transformer-based SRs overlook the item correlations among different training batches. Such an insufficient global weighting strategy also leads to a

[*] Corresponding Authors.

decrease in the model's performance [Guo *et al.*, 2023].

(2) The parameter scale remains substantial. Although the application of linear attention reduces the computational complexity of global weighting [Katharopoulos *et al.*, 2020], there still remains substantial learnable feature matrices in the stacked encoders of the Graph Transformer. Furthermore, for each encoder layer, the model repeatedly generates a set of Query, Key, and Value vectors for every input feature [Zhang *et al.*, 2023a]. Such a strategy also diminishes the efficiency of the Graph Transformer-based models.

To tackle the above challenges, we propose a **Co**mmunal **L**inear **A**ttention-enhanced Graph Trans**Former** for Sequential Recommendation, namely **CoLA-Former**. Specifically, we introduce a communal linear attention mechanism with dual normalization (i.e., the CoLAttention presented in Figure 1 (c)). It employs low-rank communal units to linearly measure the item correlations on sequential graphs. With the usage of dual normalization, such a mechanism is able to reduce the model's sensitivity to the scale of input features, while focusing on significant features on the attention maps. Meanwhile, the communal units operate independently from the Graph Transformer's encoders and are reusable across various training batches. This design simultaneously enables the model to consider inter-batch item correlations and reduces the parameter scale. Furthermore, we devise a low-rank approximation component. It reduces the size of trainable parameters in the weight matrix via factorization and utilizes weight distillation to retain key features for the Graph Transformer. Experimental results on three real-world datasets indicate that CoLA-Former excels in both accuracy and efficiency over twelve state-of-the-art competitive methods.

The main contributions of this work are listed as follows.

- We propose a lightweight yet efficient Graph Transformer network for sequential recommendation, namely CoLA-Former.
- We present a communal linear attention mechanism with linear computational complexity but only a minor loss of prediction accuracy.
- We optimize the structure of Graph Transformer to consider inter-batch item correlations and improve the parameter efficiency.
- We devise a low-rank approximation component that reduces the parameter scale and distills significant features for CoLA-Former.

2 Related Works

Sequential Recommendation. Recently, researchers have applied various approaches for the SR task, including RNN-based techniques [Liu *et al.*, 2024a; Yue *et al.*, 2024], GNN-oriented methods [Zhang *et al.*, 2024a; Zhang *et al.*, 2023b], Transformer-inspired architectures [Du *et al.*, 2024; Shi *et al.*, 2024], and self-supervised learning strategies [Zhao *et al.*, 2024; Liu *et al.*, 2024b]. As one of the most representative graph-based SR methods, Graph Transformer [Luo *et al.*, 2024; Li *et al.*, 2023] has achieved remarkable performance. It enables the global weighting when learning node representations on the sequential graphs [Xia *et al.*, 2023; Fan *et al.*, 2021]. However, these methods have quadratic complexity, hindering their deployment to the resource-constrained scenarios [Zhang *et al.*, 2023a].

Lightweight Recommendation. Explorations of lightweight recommendation [Zhou *et al.*, 2023; Xu *et al.*, 2024; Zhang *et al.*, 2024b] could be roughly classified into two categories. One is to compress the original collaborative matrices [Li *et al.*, 2021; Zhou *et al.*, 2023]. Another one is to optimize the model structures [He *et al.*, 2020; Yue *et al.*, 2024; Gao *et al.*, 2024]. As for the Graph Transformer-based methods, one practical lightweight solution is to replace the Self-Attention (SA) with the Linear Attention (LA) mechanism [Katharopoulos *et al.*, 2020; Han *et al.*, 2023]. The LA significantly reduces the computational complexity by utilizing custom kernel functions [Liu *et al.*, 2023]. However, these LA-based lightweight solutions inevitably decrease the recommendation accuracy of Graph Transformers, and their parameter scales remain substantial as well.

3 Methodology

3.1 Preliminaries

Notations

Let $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$ be the set of all items, while $\mathcal{U} = \{u_1, u_2, \dots, u_k, \dots, u_m\}$ denotes the set of all users, where u_k represents the k-th user. The set of all sequences is defined as $\mathcal{S} = \{S_1, S_2, \dots, S_k, \dots, S_m\}$, where S_k denotes the historical interaction sequence of u_k. Subsequently, we specify the definition of S_k as $S_k = \{I_1^k, I_2^k, \dots, I_j^k, \dots, I_{t_k}^k\}$, where I_j^k denotes the j-th interacted item in the sequence, t_k is the length of the sequence. Furthermore, we define $P_k = \{p_1^k, p_2^k, \dots, p_j^k, \dots, p_{t_k}^k\}$ as the set recording the exact position of each interaction within the sequence S_k. $P_k \in \mathcal{P}$, and $\mathcal{P}$ denotes the set of all positional information.

Problem Definition

Given u_k, S_k, and P_k, the SR task aims to suggest the subsequent item $I_{t_k+1}^k$ that u_k is likely to interact with. The probability of each candidate for recommendation is denoted as:

$$P(I_{t_k+1}^k | u_k, S_k, P_k) \sim f(u_k, S_k, P_k), \tag{1}$$

where $P(I_{t_k+1}^k | u_k, S_k, P_k)$ is the probability of recommending item $I_{t_k+1}^k$ to u_k, $f(u_k, S_k, P_k)$ denotes the approximation function for estimating the probability.

Sequential Graph Construction

The user behavioral sequences are treated as inputs of the model. We then utilize them to construct the sequential graph for CoLA-Former. Two types of associations are taken into consideration during the graph construction, i.e., user-item interactive relations and item-item sequential dependencies. The sequential graph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of nodes, and $\mathcal{E}$ represents the set of edges. The adjacency matrix $\mathcal{A} \in \mathbb{R}^{(m+n)\times(m+n)}$ of the graph $\mathcal{G}$ is denoted as:

$$\mathcal{A} = \begin{bmatrix} \mathbf{R}_{\mathcal{I}\to\mathcal{I}} & \mathbf{R}_{\mathcal{U}\to\mathcal{I}} \\ \mathbf{R}_{\mathcal{I}\to\mathcal{U}} & \mathbf{0} \end{bmatrix}, \tag{2}$$

where $\mathbf{R}_{\mathcal{I}\to\mathcal{I}} \in \mathbb{R}^{n\times n}$ is the adjacency matrix carrying the sequential dependencies among items, each entry R_{ij} is 1 if

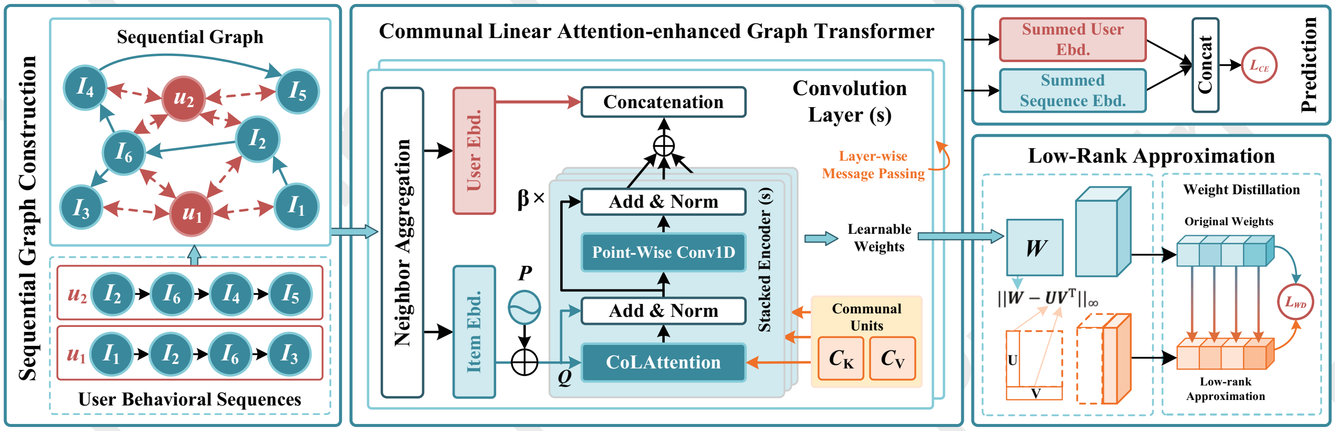


Figure 2: Framework of CoLA-Former, where u_1 and u_2 represent two different users, $\{I_1, I_2, \ldots, I_6\}$ denote the interacted items that compose the behavioral sequences for the users.

item I_j is the successor of item I_i, otherwise 0; $\mathbf{R}_{\mathcal{U}\to\mathcal{I}} \in \mathbb{R}^{n\times m}$ represents the user-item interaction matrix, each entry R_{ki} is 1 if there is an interaction between user u_k and item I_i, otherwise 0; $\mathbf{R}_{\mathcal{I}\to\mathcal{U}} \in \mathbb{R}^{m\times n}$ denotes the transpose matrix of $\mathbf{R}_{\mathcal{U}\to\mathcal{I}}$.

3.2 Framework of CoLA-Former

The framework of CoLA-Former is shown in Figure 2. The model consists of four key components, i.e., the neighbor aggregation (Section 3.3), the stacked encoders (Section 3.4), the prediction layer (Section 3.5), and the low-rank approximation (Section 3.6). We elaborate on the above components in subsequent sections.

3.3 Neighbor Aggregation

The CoLA-Former model treats the sequential graph $\mathcal{G}$ as its input and initializes the node embeddings for items and users randomly at the initial layer. Specifically, the item embeddings $\mathbf{E}_I^{(0)}$ are initialized in $\mathbb{R}^{n\times d}$, and the user embeddings $\mathbf{E}_u^{(0)}$ are initialized in $\mathbb{R}^{m\times d}$. The architecture of the model comprises L graph convolution layers, where L is a hyper-parameter that determines the depth of the network. Inspired by LightGCN [He *et al.*, 2020], CoLA-Former omits both feature transformation and nonlinear activation functions within the graph convolution operations, retaining only the neighbor aggregation mechanism for node representation learning. Due to the sequential nature of the input graph $\mathcal{G}$, it requires additional consideration of the sequential dependencies between items during the message aggregation stage. In terms of other components, such as Laplacian regularization and the layer-wise message passing strategy, CoLA-Former maintains consistency with LightGCN's methodology.

Taking user u_k as an example, the embedding of u_k is denoted as $\mathbf{e}_{u_k}$; the embedding of u_k's interaction sequence S_k is represented as $\mathbf{e}_{S_k}$, and the j-th item embedding within the sequence is denoted as $\mathbf{e}_{I_j^k}$. Then, the graph convolution operation at the l-th ($0 < l < L$) layer in CoLA-Former is defined as:

$$\mathbf{e}_{u_k}^{(l)} = \sum_{I_j^k \in S_k} \frac{1}{\sqrt{|S_k||\mathcal{N}_u|}} \mathbf{e}_{I_j^k}^{(l-1)}; \tag{3}$$

$$\mathbf{e}_{I_j^k}^{(l)} = \sum_{u_q\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||S_k|}}\mathbf{e}_{u_q}^{(l-1)} + \sum_{I_{j-1}^k\in\mathcal{N}_I}\frac{1}{\sqrt{|\mathcal{N}_I||S_k|}}\mathbf{e}_{I_{j-1}^k}^{(l-1)}, \tag{4}$$

where $\mathcal{N}_u$ denotes the set of users that interact with item I_j^k, u_q represents q-th user within $\mathcal{N}_u$; $\mathcal{N}_I$ denotes the set of neighbor items of I_j^k within multiple sequences, I_{j-1}^k represents one of the neighbor items to I_j^k.

The node representations for all users $\mathbf{E}_u^{(l)} \in \mathbb{R}^{m\times d}$ and items $\mathbf{E}_I^{(l)} \in \mathbb{R}^{n\times d}$ at the l-th layer are denoted as:

$$\mathbf{E}_u^{(l)} = \text{Concat}\left[\mathbf{e}_{u_1}^{(l)}, \mathbf{e}_{u_2}^{(l)}, \ldots, \mathbf{e}_{u_i}^{(l)}, \ldots, \mathbf{e}_{u_m}^{(l)}\right] \cdot \mathbf{W}_U; \tag{5}$$

$$\mathbf{E}_I^{(l)} = \text{Concat}\left[\mathbf{e}_{I_1}^{(l)}, \mathbf{e}_{I_2}^{(l)}, \ldots, \mathbf{e}_{I_j}^{(l)}, \ldots, \mathbf{e}_{I_n}^{(l)}\right] \cdot \mathbf{W}_I, \tag{6}$$

where $\mathbf{W}_U$ and $\mathbf{W}_I$ denote the linear transformation matrix that aligns the dimensions of input features and the outputs.

3.4 Stacked Encoders

After the neighbor aggregation, we follow the principle of Graph Transformer [Xia *et al.*, 2023; Sun *et al.*, 2019] to refine the item embeddings via multiple encoders. As illustrated in Figure 2, β encoders are stacked at each graph convolution layer of the CoLA-Former. However, without any recurrence or located methods, the encoders are not aware of the order of the input sequence. To make use of the sequential information, we inject learnable positional embeddings $\mathbf{E}_P$ into the input item embeddings $\mathbf{E}_I$ at the bottoms of the stacks.

Learnable Positional Encoding

Since an item in the sequential graph belongs to multiple sequences with distinct positional information, the overlapping position encodings may interfere with one another. To this end, we introduce a learnable position encoding. It treats the position of each item as learnable vectors and maintains an external position matrix C_P to preserve the vectors. Such a design allows the model to iteratively update the positional representations during the model training, reflecting the typical positions of items in the majority of users' sequences. For each item, we initialize its learnable position encoding by computing the average of its relative positions in varying sequences. Taking item $I_j^k \in S_k$ as an example, its positional embedding $\mathbf{e}_{p_j^k} \in \mathbb{R}^{1\times d}$ is initialized as:

$$\mathbf{e}_{p_j^k} = \frac{1}{|\mathcal{N}_S|} \sum_{S_k \in \mathcal{N}_S} p_j^k, \tag{7}$$

where $\mathcal{N}_S$ is the set of all sequences that contain item I_j, p_j^k is the absolute position of I_j^k in the sequence S_k.

Then, the position embeddings $\mathbf{e}_{P_k} \in \mathbb{R}^{t_k \times d}$ of sequence S_k are detailed as $\mathbf{e}_{P_k} = [\mathbf{e}_{p_1^k}, \mathbf{e}_{p_2^k}, \ldots, \mathbf{e}_{p_j^k}, \ldots, \mathbf{e}_{p_{t_k}^k}]$. Finally, we obtain the **Q**uery vector $\mathbf{Q}_{S_k}$ for sequence S_k by adding $\mathbf{e}_{P_k}$ to $\mathbf{e}_{S_k}$, where $\mathbf{e}_{S_k}$ is the representation of S_k.

CoLAttention

After obtaining the query vectors for all sequences, the encoder utilizes CoLAttention to consider the global correlations within sequences. Different from the self-attention [Chen *et al.*, 2023] and linear attention [Liu *et al.*, 2023], the key vectors and value vectors in CoLAttention are derived from two low-rank communal units, i.e., $\mathbf{C}_K \in \mathbb{R}^{d\times\alpha}$ and $\mathbf{C}_V \in \mathbb{R}^{d\times\alpha}$. As the dimensions of the communal units are much lower than that of the input sequence representations (i.e., $\alpha \ll t_k$), the complexity of CoLAttention approximates linearity, i.e., $\mathcal{O}(n \times \alpha \times d)$. Besides, these communal units are reusable and shared across multiple encoders and training batches. This design improves the parameter efficiency and makes the significant features from the attention maps transmissible.

The calculation of CoLAttention is depicted in Figure 1 (c), which is formulated as:

$$a_{i,j} = \text{Norm}\left(\frac{\mathbf{Q}_{S_k}\mathbf{C}_K}{\sqrt{d}}\right); \tag{8}$$

$$\hat{\mathbf{e}}_{S_k} = \text{Norm}\left(a_{i,j} \cdot \mathbf{C}_V^{\top}\right), \tag{9}$$

where $a_{i,j} \in \mathbb{R}^{t_k\times\alpha}$ represents the low-rank attention map of CoLAttention, $\hat{\mathbf{e}}_{S_k} \in \mathbb{R}^{t_k\times d}$ is the refined sequence representations, α denotes the hyper-parameter that controls the dimension of communal units.

Instead of using the softmax function which is sensitive to the scale of input features, we adopt a double normalization [Guo *et al.*, 2023] from the field of computer vision to the CoLAttention. Such a mechanism separately normalizes columns and rows of the attention map $a_{i,j}$, mitigating the sensitivity and enhancing model performance. After detailing the calculation for each single attention head, we define the sequence representations weighted by multiple heads as $\{\hat{\mathbf{e}}_{S_k}^{(1)}, \hat{\mathbf{e}}_{S_k}^{(2)}, \ldots, \hat{\mathbf{e}}_{S_k}^{(v)}, \ldots, \hat{\mathbf{e}}_{S_k}^{(h)}\}$, where h is the head number. Then, the refined sequence embeddings $\bar{\mathbf{e}}_{S_k}$ are formulated as:

$$\bar{\mathbf{e}}_{S_k} = \sum_{v=1}^{h} \hat{\mathbf{e}}_{S_k}^{(v)} \cdot \mathbf{W}_M, \tag{10}$$

where $\mathbf{W}_M$ is the learnable weighting matrix.

Subsequently, we perform an residual link to obtain the updated representations $\tilde{\mathbf{e}}_{S_k}$.

Point-wise Conv1D

The multi-head attention structure in CoLAttention is adept at capturing long-range dependencies. However, the model still struggles to capture the local features and short-range dependencies within sequences [Kreuzer *et al.*, 2021]. To this end, we develop a lightweight local weighting layer via a point-wise Conv1D layer, which operates on each individual item embedding of $\tilde{\mathbf{e}}_{S_k}$ and their immediate neighbors. This localized operation allows the model to focus on short-range dependencies and patterns. Specifically, both the kernel size and the stride of the point-wise Conv1D are set to 1 for linearly local weightings [Wu *et al.*, 2023b]. The calculation is denoted as:

$$\mathbf{z}_{S_k} = \tilde{\mathbf{e}}_{S_k} * \mathbf{W}_c + \mathbf{b}_c, \tag{11}$$

where $*$ denotes the one-dimensional convolutions, $\mathbf{W}_c \in \mathbb{R}^{d\times d}$ represents the convolution kernel, $\mathbf{b}_c \in \mathbb{R}^{1\times d}$ is the bias term.

By incorporating point-wise Conv1D layers into the stacked encoders, CoLA-Former strikes a balance between capturing both local and global information. We perform another residual link to generate the output representations $\tilde{\mathbf{z}}_{S_k}$. Then, the refined representations of all items $\mathbf{E}_I^{(t)}$ in the t-th $(0 < t < \beta)$ encoder are derived as:

$$\mathbf{E}_I^{(t)} = \text{Concat}\left[\tilde{\mathbf{z}}_{S_1}, \ldots, \tilde{\mathbf{z}}_{S_k}, \ldots, \tilde{\mathbf{z}}_{S_{|\mathcal{S}|}}\right] \cdot \mathbf{W}_S, \tag{12}$$

where $\mathbf{W}_S$ denotes the learnable transformation matrix.

The refined item representations from the t-th encoder are treated as the input of the $(t+1)$-th encoder. We define the output item embeddings of the β encoder as the output item representations $\hat{\mathbf{E}}_I^{(l)} \in \mathbb{R}^{n\times d}$ for the l-th CoLA-Former layer.

3.5 Prediction Layer

Follow the Multiple Layer Aggregation Protocol (MLAP) in LightGCN [He *et al.*, 2020], the summed representations for all users and sequences are resulted by aggregating embeddings from multiple convolution layers:

$$\mathbf{E}_U = \sum_{l=0}^{L} \frac{1}{1+L}\mathbf{E}_u^{(l)}; \qquad \mathbf{E}_S = \sum_{l=0}^{L} \frac{1}{1+L}\hat{\mathbf{E}}_I^{(l)}, \tag{13}$$

where $\mathbf{E}_U$ and $\mathbf{E}_S$ denote the resulted node representations for users and sequences, respectively.

After obtaining the final sequence representation $\mathbf{E}_S$, we concatenate it with the user representations $\mathbf{E}_U$, and then feed them into the prediction layer (as shown in Eqn. (14)).

$$P(I_{t+1}|\mathcal{U}, \mathcal{S}, \mathcal{P}) = \text{Softmax}\left(\mathbf{W}_f[\mathbf{E}_S, \mathbf{E}_U]^{\top} + \mathbf{b}_f\right), \tag{14}$$

where $\mathbf{W}_f \in \mathbb{R}^{n\times 2d}$ represents the weighting matrix that transforms the dimensions of predictions to the number of all candidates, $\mathbf{b}_f \in \mathbb{R}^{n\times 1}$ is the bias term that adjusts the activation threshold.

Then, we minimize the cross-entropy loss function $\mathcal{L}_{CE}$ to optimize the learnable parameters, which is formalized as:

$$\mathcal{L}_{CE} = -\frac{1}{|\mathcal{S}|}\sum_{I_{t+1}\in\mathcal{I}} \log P(I_{t+1}|\mathcal{U},\mathcal{S},\mathcal{P}). \tag{15}$$

3.6 Low-Rank Approximation

The integration of CoLAttention has successfully diminished computational complexity and augmented parameter efficiency. Nevertheless, the stacked encoders still retain a significant quantity of learnable weighting matrices. Inspired by the success of low-rank approximation in Computer Vision [Guo *et al.*, 2024], we propose to apply this technique to the CoLA-Former to further reduce the parameter scale.

Factorization

We assume that all the learnable weights in the stacked encoders are denoted as $\mathbf{W}_E \in \mathbb{R}^{d_{in}\times d_{out}}$. Then, we factorize the matrix as:

$$\begin{aligned}\mathbf{W}_E^\top \mathbf{X} &\approx \left(\mathbf{U}_E \mathbf{V}_E^\top\right)^\top \mathbf{X} \\ &= \mathbf{V}_E^\top \left(\mathbf{U}_E^\top \mathbf{X}\right),\end{aligned} \tag{16}$$

where $\mathbf{X}$ denotes the input features that are multiplied by the learnable feature matrices within the stacked encoder architecture, $\mathbf{U}_E \in \mathbb{R}^{d_{in}\times\gamma}$ and $\mathbf{V}_E \in \mathbb{R}^{d_{out}\times\gamma}$ are low-rank matrices, γ controls the rank.

Referring to the setting of PELA [Guo *et al.*, 2024], we seek the low-rank approximation of the original matrix and deliberately choose a smaller γ=$\frac{1}{4}\min(d_{in}, d_{out})$. Then, we exploit the well-known SVD approach [Wu *et al.*, 2022] to perform the low-rank approximation as:

$$\mathrm{SVD}(\mathbf{W}_E) = \mathbf{U}^*\mathbf{\Sigma}\mathbf{V}^*, \tag{17}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{d_{in}\times d_{out}}$ is the singular value matrix, $\mathbf{U}^* \in \mathbb{R}^{d_{in}\times d_{in}}$ and $\mathbf{V}^* \in \mathbb{R}^{d_{out}\times d_{out}}$ are complex unitary matrices. Then, the low-rank matrices are formalized via the following transformation (i.e., truncated singular value decomposition):

$$\mathbf{U}_E = \mathbf{U}^*_{[:,:\gamma]}\mathbf{\Sigma}^{\frac{1}{2}}_{[:\gamma,:\gamma]}; \quad \mathbf{V}_E = \left(\mathbf{\Sigma}^{\frac{1}{2}}_{[:\gamma,:\gamma]}\mathbf{V}^*_{[:\gamma,:]}\right)^\top. \tag{18}$$

The resulting approximated weight matrix $\hat{\mathbf{W}}_E$ is denoted as:

$$\hat{\mathbf{W}}_E = \mathbf{U}_E \cdot \mathbf{V}_E^\top. \tag{19}$$

Weight Distillation

Although the application of truncated Singular Value Decomposition successfully reduces the scale of the learnable matrix, it inherently leads to the loss of features. To this end, we further propose a weight distillation component to align the weights of the approximated weights $\hat{\mathbf{W}}_E$ with the original weights $\mathbf{W}_E$. For β layer stacked encoders, the weight distillation loss is defined as follows:

$$\mathcal{L}_{WD} = \frac{1}{2\beta}\sum_{l=1}^{\beta}\|\mathcal{M}(\mathbf{W}_E) - \mathcal{M}(\hat{\mathbf{W}}_E)\|^2, \tag{20}$$

where $\mathcal{M}$ is the identity mapping method that transfers the feature to the target feature space. In this way, the output features from each encoder layer of the low-rank model are expected to share a similar distribution with that of the original one.

4 Experiments

In this section, we conduct experiments to demonstrate the superiority of CoLA-Former in terms of the prediction accuracy and model efficiency. We aim to answer the following four **R**esearch **Q**uestions.

- **RQ1:** What is the performance of CoLA-Former in terms of the prediction accuracy on the SR tasks?
- **RQ2:** How is the training efficiency and parameter scale of CoLA-Former compared with other state-of-the-art methods?
- **RQ3:** What is the contribution of key components to CoLA-Former for the recommendation performance?
- **RQ4:** How do the hyper-parameters affect the performance of CoLA-Former?

4.1 Experimental Setups

Datasets

We evaluate the performance of CoLA-Former on three real-world datasets, i.e., Amazon-Food, Douban and MovieLens. Specifically, Amazon-Food dataset is a collection of food product reviews from Amazon, encompassing a vast array of items and customer feedback recommendation research [Xia *et al.*, 2023]. MovieLens dataset is a widely-used collection of movie ratings provided by the GroupLens Research Project, offering a rich resource for collaborative filtering and recommendation system studies [Gao *et al.*, 2024]. Douban dataset contains book reviews and ratings from the Chinese social media platform Douban, which is utilized for sentiment analysis and recommendation research [Zhang *et al.*, 2023a].

Evaluation Metrics

In experiments, two common evaluation metrics are adopted to assess the performance of CoLA-Former, i.e., Top-N Recall (RC@N) [Yue *et al.*, 2023] and Top-N Mean Reciprocal Rank (MRR@N) [Zhang *et al.*, 2023a], where N={5, 20}.

Implementation Details

CoLA-Former is implemented with Pytorch 2.1.2. The model training is accelerated via NVIDIA® RTX 3090 (24GB) GPU. We employ the Xavier [Glorot and Bengio, 2010] for the initialization of the learnable parameters and treat Adam [Kingma and Ba, 2015] as the optimizer. For the model training, we set the batch size as 256 and the dropout rate as 0.2. The learning rate is set to 0.003 for Amazon-Food, 0.001 for MovieLens, and 0.001 for Douban. Besides, we uniformly set the embedding size to 64 for CoLA-Former and other competitive baselines to ensure the fairness of experimental results. As for other hyper-parameters of baselines, we follow the optimal setup reported in their papers and fine-tune them on each dataset.

Dataset	Amazon-Food				MovieLens				Douban			
Metric	**RC@5**	**RC@10**	**MRR@5**	**MRR@10**	**RC@5**	**RC@10**	**MRR@5**	**MRR@10**	**RC@5**	**RC@10**	**MRR@5**	**MRR@10**
SASRec (2018)	73.29	74.96	71.23	72.48	2.64	4.85	1.53	1.65	56.79	58.55	56.31	57.47
S^3-Rec (2020)	73.58	75.42	72.07	72.88	2.71	4.92	1.59	1.66	56.78	58.50	56.34	57.51
MLSI (2024)	76.52	78.54	75.25	75.86	3.85	6.21	2.08	2.30	64.12	65.89	61.56	62.42
RDCRec (2024)	75.98	78.24	75.30	75.88	3.96	6.22	2.07	2.33	64.99	66.04	62.64	63.18
LSAN (2021)	73.62	77.42	72.80	73.11	2.66	4.89	1.58	1.63	62.87	64.00	58.92	60.15
LRURec (2024)	75.58	78.11	74.79	75.53	2.89	4.96	1.61	1.75	64.28	65.59	59.12	61.44
SMLP4Rec (2024)	76.00	78.32	75.64	75.97	2.95	4.95	1.66	1.72	64.32	65.74	61.37	62.66
SURGE (2021)	73.12	74.44	71.63	72.26	2.64	4.67	1.56	1.61	57.21	59.32	55.48	57.74
TGSRec (2021)	76.48	78.33	75.57	76.08	3.82	6.27	2.10	2.13	64.25	66.00	60.52	61.06
GCL4SR (2022)	76.32	78.54	74.38	74.68	3.66	6.20	1.99	2.11	65.81	66.02	62.91	63.08
TGT (2023)	76.99	78.45	74.75	75.12	3.95	6.37	2.12	2.16	65.99	66.38	63.22	63.54
MG-Former (2024)	77.45	78.56	76.10	76.21	4.13	7.29	2.12	2.40	65.94	66.90	63.52	63.78
CoLA-Former	**79.39**	**81.12**	**76.69**	**76.93**	**4.47**	**7.92**	**2.17**	**2.61**	**67.11**	**68.06**	**64.94**	**65.24**

Table 1: Prediction performance (%) of different methods on three real-world datasets (i.e., Amazon-Food, MovieLens and Douban). The best results are indicated in bold while the sub-optimal results are underlined.

Baselines

To validate the effectiveness of CoLA-Former, we compare its performance with 12 competitive baselines: **1) Sequential Recommendations (SRs).** SASRec [Kang and McAuley, 2018]; S^3-Rec [Zhou *et al.*, 2020]; MLSI [Hu *et al.*, 2024]; RDCRec [Zhao *et al.*, 2024]; **2) Lightweight Sequential Recommendations (LSRs).** LSAN [Li *et al.*, 2021]; LRURec [Yue *et al.*, 2024]; SMLP4Rec [Gao *et al.*, 2024]; **3) Graph-based Sequential Recommendations (GSRs).** SURGE [Chang *et al.*, 2021]; TGSRec [Fan *et al.*, 2021]; GCL4SR [Zhang *et al.*, 2022]; TGT [Xia *et al.*, 2023]; MG-Former [Luo *et al.*, 2024].

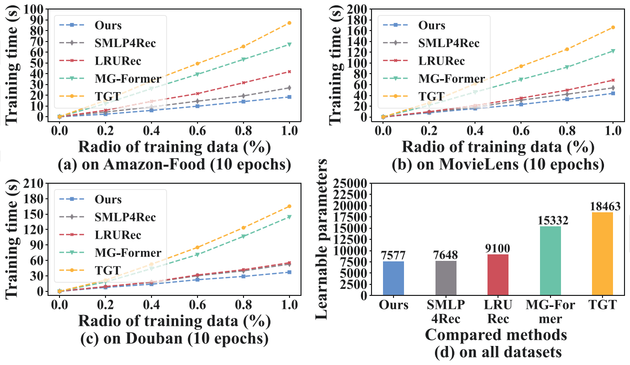
4.2 Prediction Performance (RQ1)

Table 1 shows the comparison of prediction accuracy between CoLA-Former and the other twelve competitive methods. The observations and analysis are given as follows: 1) Most GSRs (i.e., TGSRec, GCL4SR, TGT, MG-Former, and CoLA-Former) exhibit superior predictive performance compared to other sequential recommendation methods. This observation indicates that GSRs excel in capturing the intricacies of user interactions and the dependencies inherent in sequential data. 2) CoLA-Former achieves the best performance on each evaluation metric across all datasets, demonstrating the effectiveness of our proposed model in addressing the SR tasks. 3) CoLA-Former outperforms the recently proposed LSRs (i.e., LSAN, LRURec, SMLP4Rec). This observation indicates that CoLA-Former achieves an optimal trade-off between lightweight efficiency and predictive precision compared to other lightweight approaches. 4) CoLA-Former exhibits superior performance compared to other Graph Transformer-based GSRs (i.e., TGSRec, TGT, and MG-Former). This observation indicates that our lightweight design has not decreased the recommendation accuracy of the Graph Transformer but has even improved it.

4.3 Lightweight Performance (RQ2)

In this section, we validate the lightweight design performance of CoLA-Former by comparing its time consumption and parameter scale with the most competitive baselines (i.e., LRURec, SMLP4Rec, TGT, MG-Former).

Time Consumption. We vary the proportion of the input data to be {0.2, 0.4, 0.6, 0.8, 1.0} on three datasets to assess the training time consumption of different methods. As shown in Figure 3 (a) to (c), CoLA-Former requires less training time than recently proposed LSRs (i.e., LRURec and SMLP4Rec) and Graph Transformer-based GSRs (i.e., TGT and MG-Former), demonstrating improved training efficiency. Moreover, the time consumption of CoLA-Former increases at a much slower pace than the other baseline methods as the dataset size grows, highlighting its superior scalability for large-scale data.



(a) on Amazon-Food (10 epochs)
(b) on MovieLens (10 epochs)
(c) on Douban (10 epochs)
(d) on all datasets

Figure 3: Comparison of time consumption and parameter scale between CoLA-Former and competitive baselines.

Parameter Scale. We also compare the parameter scale of CoLA-Former with the other competitive methods. As depicted in Figure 3 (d), CoLA-Former requires fewer parameters than other Graph Transformer-based GSRs (i.e., TGT and MG-Former). This observation demonstrates that the lightweight design within CoLA-Former effectively diminishes the parameter scale of the Graph Transformer, resulting in enhanced lightweight performance. Besides, CoLA-Former requires notably fewer learnable parameters than the other LSR methods, making it more suitable for resource-constrained SR scenarios.

4.4 Ablation Studies (RQ3)

In this section, we perform a series of ablation studies across two datasets (i.e., MovieLens and Douban) to assess the contributions of each component within CoLA-Former. The details of variant methods are as follows: 1) $\text{CoLA}_{\text{r/p SA}}$ denotes the variant method that replaces the CoLAttention with self-attention. 2) $\text{CoLA}_{\text{r/p LA}}$ represents the variant that replaces the CoLAttention with linear attention. 3) $\text{CoLA}_{\text{w/o P}}$ denotes the variant that removes the learnable positional encoding. 4) $\text{CoLA}_{\text{w/o C1D}}$ is the variant that removes the Point-wise Conv1D. 5) $\text{CoLA}_{\text{w/o LRA}}$ denotes the variant that disables the low-rank approximation. 6) $\text{CoLA}_{\text{w/o ALL}}$ is another variant that disables all components in CoLA-Former.

Dataset	**MovieLens**				**Douban**			
Metric	**RC**		**MRR**		**RC**		**MRR**	
	@5	**@10**	**@5**	**@10**	**@5**	**@10**	**@5**	**@10**
$\text{CoLA}_{\text{w/o ALL}}$	3.80	6.21	2.07	2.11	63.96	65.58	60.12	60.93
$\text{CoLA}_{\text{r/p LA}}$	3.97	6.28	2.06	2.10	63.99	65.92	60.02	60.56
$\text{CoLA}_{\text{w/o C1D}}$	4.01	6.45	2.08	2.21	64.32	65.98	62.04	62.73
$\text{CoLA}_{\text{w/o P}}$	4.20	7.19	2.08	2.19	65.89	67.09	63.82	64.14
$\text{CoLA}_{\text{r/p SA}}$	4.22	7.38	2.14	2.49	66.21	67.17	63.84	63.98
$\text{CoLA}_{\text{w/o LRA}}$	**4.49**	**7.95**	**2.24**	**2.65**	**67.31**	**68.26**	**65.05**	**65.33**
CoLA-Former	4.47	7.92	2.17	2.61	67.11	68.06	64.94	65.24

Table 2: Prediction performance $(\%)$ of different variant methods on two real-world datasets (i.e., MovieLens and Douban).

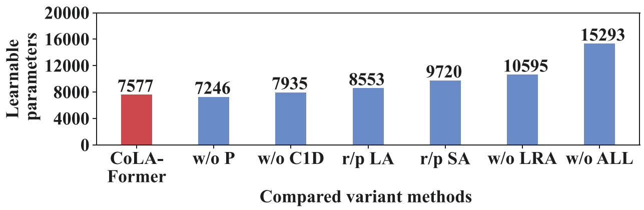


Figure 4: Comparison of learnable parameter scale between CoLA-Former and its variants.

From Table 2 and Figure 4, we have the following observations: 1) CoLA-Former and $\text{CoLA}_{\text{w/o LRA}}$ significantly outperform $\text{CoLA}_{\text{r/p SA}}$ and $\text{CoLA}_{\text{r/p LA}}$ in both prediction accuracy and parameter consumption, showcasing the superiority of the CoLAttention. 2) CoLA-Former and $\text{CoLA}_{\text{w/o LRA}}$ achieve better prediction performance than $\text{CoLA}_{\text{w/o P}}$ and $\text{CoLA}_{\text{w/o C1D}}$, demonstrating the effectiveness of the learnable positional encoding and the point-wise Conv1D. 3) CoLA-Former has similar predictive performance with $\text{CoLA}_{\text{w/o LRA}}$, but with a significantly reduced parameter scale. This observation indicates that our low-rank approximation component achieves a reasonable trade-off between prediction accuracy and parameter scale.

4.5 Hyper-parameters Analysis (RQ4)

1) The hyper-parameter α controls the dimension of the communal units. Figures 5 (a) to (c) present the performance of CoLA-Former for different α values in $\{16, 32, 64, 128, 256\}$ across three diverse datasets. The experimental results show that varying α values do not significantly affect the model's predictive capabilities. In other words, CoLA-Former is able to function with a lower alpha value (even the α is set to 16), highlighting its lightweight efficiency. 2) The hyper-parameter L controls the depth of the convolutional layer. As depicted in Figures 5 (d) to (f), CoLA-Former exhibits optimal performance at the layer depth of 2, which is in line with the majority of graph convolution-based models. 3) β and h are two significant hyper-parameters that control the number of encoder layers and attention heads, respectively. As shown in Figures 5 (g) to (i), CoLA-Former reaches the best performance on all datasets when β=3 and h=2. This observation indicates that the model's efficacy in sequential recommendation tasks is not directly proportional to the number of attention heads or encoders. Optimal performance is achieved with a judiciously selected minimal number of these components.

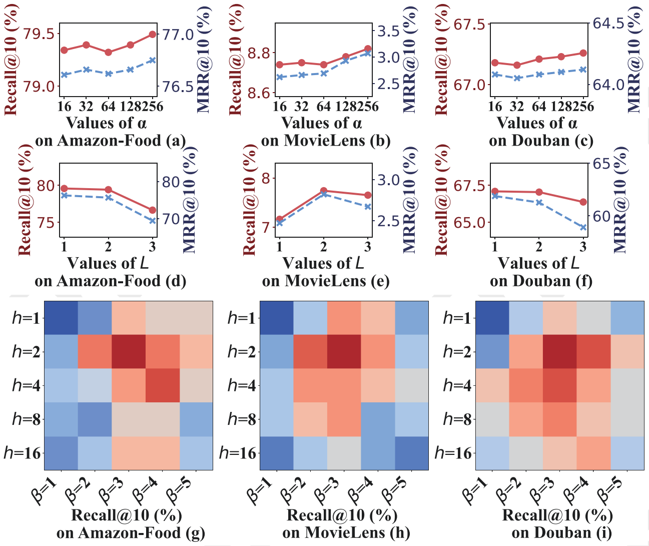


Figure 5: Impact of hyper-parameters α, β, h and L on three datasets.

5 Conclusion

In this work, we introduce a lightweight Graph Transformer-based GSRs called CoLA-Former. In its encoder, we design the CoLAttention to linearly consider the inter-batch global correlations among items. This design achieves linear complexity of global weighting but with less loss of the prediction accuracy. Moreover, we present a low-rank approximation component. It factorizes the high-dimensional weight matrix into two smaller matrices and preserves the significant features via weight distillation. Such a component significantly reduces the scale of the learnable weights.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 62472263, 52374221, 62072288, 62176243), the National Key R&D Program of China (Grant No. 2022ZD0119501), the Taishan Scholar Program of Shandong Province, Shandong Youth Innovation Team, the Natural Science Foundation of Shandong Province (Grant No. ZR2024MF034, ZR2022MF268).

References

[Chang *et al.*, 2021] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *SIGIR*, pages 378–387, Virtual Event, Canada, 2021. ACM.

[Chen *et al.*, 2023] Huiyuan Chen, Kaixiong Zhou, Zhimeng Jiang, Chin-Chia Michael Yeh, Xiaoting Li, Menghai Pan, Yan Zheng, Xia Hu, and Hao Yang. Probabilistic masked attention networks for explainable sequential recommendation. In *IJCAI*, pages 2068–2076, Macao, SAR, China, 2023. ijcai.org.

[Du *et al.*, 2024] Hanwen Du, Huanhuan Yuan, Pengpeng Zhao, Deqing Wang, Victor S. Sheng, Yanchi Liu, Guanfeng Liu, and Lei Zhao. Feature-aware contrastive learning with bidirectional transformers for sequential recommendation. *TKDE*, 36(12):8192–8205, 2024.

[Fan *et al.*, 2021] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *CIKM*, pages 433–442, Queensland, Australia, 2021. ACM.

[Gao *et al.*, 2024] Jingtong Gao, Xiangyu Zhao, Muyang Li, Minghao Zhao, Runze Wu, Ruocheng Guo, Yiding Liu, and Dawei Yin. Smlp4rec: An efficient all-mlp architecture for sequential recommendations. *TOIS*, 42(3):86:1–86:23, 2024.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, Sardinia, Italy, 2010. JMLR.org.

[Guo *et al.*, 2023] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *TPAMI*, 45(5):5436–5447, 2023.

[Guo *et al.*, 2024] Yangyang Guo, Guangzhi Wang, and Mohan S. Kankanhalli. PELA: learning parameter-efficient models with low-rank approximation. In *CVPR*, pages 15699–15709, Seattle, WA, USA, 2024. IEEE.

[Han *et al.*, 2023] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *ICCV*, pages 5938–5948, Paris, France, 2023. IEEE.

[He *et al.*, 2020] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pages 639–648, Virtual Event, 2020. ACM.

[Hu *et al.*, 2024] Haibing Hu, Kai Han, Zhizhuo Yin, and Defu Lian. Spatial and temporal user interest representations for sequential recommendation. *TCSS*, 11(5):6087–6097, 2024.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pages 197–206, Singapore, 2018. IEEE Computer Society.

[Katharopoulos *et al.*, 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pages 5156–5165, Virtual Event, 2020. PMLR.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, pages 1–15, San Diego, CA, US, 2015. openreview.org.

[Kreuzer *et al.*, 2021] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *NeurIPS*, pages 21618–21629, virtual, 2021.

[Li *et al.*, 2021] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. Lightweight self-attentive sequential recommendation. In *CIKM*, pages 967–977, Queensland, Australia, 2021. ACM.

[Li *et al.*, 2023] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. Graph transformer for recommendation. In *SIGIR*, pages 1680–1689, Taipei, Taiwan, 2023. ACM.

[Liu *et al.*, 2023] Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, Zitao Liu, and Qing Li. Linrec: Linear attention mechanism for long-term sequential recommender systems. In *SIGIR*, pages 289–299, Taipei, Taiwan, 2023. ACM.

[Liu *et al.*, 2024a] Chengkai Liu, Jianghao Lin, Hanzhou Liu, Jianling Wang, and James Caverlee. Behavior-dependent linear recurrent units for efficient sequential recommendation. In *CIKM*, pages 1430–1440, Boise, ID, USA, 2024. ACM.

[Liu *et al.*, 2024b] Yuxi Liu, Lianghao Xia, and Chao Huang. Selfgnn: Self-supervised graph neural networks for sequential recommendation. In *SIGIR*, pages 1609–1618, Washington DC, USA, 2024. ACM.

[Luo *et al.*, 2024] Tianze Luo, Yong Liu, and Sinno Jialin Pan. Collaborative sequential recommendations via multiview gnn-transformers. *TOIS*, 42(6):1–27, 2024.

[Ma *et al.*, 2024] Haokai Ma, Ruobing Xie, Lei Meng, Yimeng Yang, Xingwu Sun, and Zhanhui Kang. Seedrec: Sememe-based diffusion for sequential recommendation. In *IJCAI*, pages 2270–2278, Jeju, South Korea, 2024. ijcai.org.

[Shi *et al.*, 2024] Chaoyu Shi, Pengjie Ren, Dongjie Fu, Xin Xin, Shansong Yang, Fei Cai, Zhaochun Ren, and Zhumin

Chen. Diversifying sequential recommendation with retrospective and prospective transformers. *TOIS*, 42(5):132:1–132:37, 2024.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450, Beijing, China, 2019. ACM.

[Wang *et al.*, 2024] Zhaobo Wang, Yanmin Zhu, Chunyang Wang, Xuhao Zhao, Bo Li, Jiadi Yu, and Feilong Tang. Graph diffusion-based representation learning for sequential recommendation. *TKDE*, 36(12):8395–8407, 2024.

[Wu *et al.*, 2022] Zhebin Wu, Lin Shu, Ziyue Xu, Yaomin Chang, Chuan Chen, and Zibin Zheng. Robust tensor graph convolutional networks via T-SVD based graph augmentation. In *KDD*, pages 2090–2099, Washington, DC, USA, 2022. ACM.

[Wu *et al.*, 2023a] Bin Wu, Xiangnan He, Le Wu, Xue Zhang, and Yangdong Ye. Graph-augmented co-attention model for socio-sequential recommendation. *TSMC*, 53(7):4039–4051, 2023.

[Wu *et al.*, 2023b] Bin Wu, Xiangnan He, Qi Zhang, Meng Wang, and Yangdong Ye. Gcrec: Graph-augmented capsule network for next-item recommendation. *TNNLs*, 34(12):10164–10177, 2023.

[Xia *et al.*, 2023] Lianghao Xia, Chao Huang, Yong Xu, and Jian Pei. Multi-behavior sequential recommendation with temporal graph transformer. *TKDE*, 35(6):6099–6112, 2023.

[Xu *et al.*, 2024] Yang Xu, Lei Zhu, Jingjing Li, Fengling Li, and Heng Tao Shen. Temporal social graph network hashing for efficient recommendation. *TKDE*, 36(7):3541–3555, 2024.

[Yue *et al.*, 2023] Guowei Yue, Rui Xiao, Zhongying Zhao, and Chao Li. AF-GCN: attribute-fusing graph convolution network for recommendation. *TBD*, 9(2):597–607, 2023.

[Yue *et al.*, 2024] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian J. McAuley, and Dong Wang. Linear recurrent units for sequential recommendation. In *WSDM*, pages 930–938, Merida, Mexico, 2024. ACM.

[Zhang *et al.*, 2022] Yixin Zhang, Yong Liu, Yonghui Xu, Hao Xiong, Chenyi Lei, Wei He, Lizhen Cui, and Chunyan Miao. Enhancing sequential recommendation with graph contrastive learning. In *IJCAI*, pages 2398–2405, Vienna, Austria, 2022. ijcai.org.

[Zhang *et al.*, 2023a] Jinyu Zhang, Huichuan Duan, Lei Guo, Liancheng Xu, and Xinhua Wang. Towards lightweight cross-domain sequential recommendation via external attention-enhanced graph convolution network. In *DASFAA*, volume 13944, pages 205–220, TianJin, China, 2023. Springer.

[Zhang *et al.*, 2023b] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. Dynamic graph neural networks for sequential recommendation. *TKDE*, 35(5):4741–4753, 2023.

[Zhang *et al.*, 2024a] Shengzhe Zhang, Liyi Chen, Chao Wang, Shuangli Li, and Hui Xiong. Temporal graph contrastive learning for sequential recommendation. In *AAAI*, pages 9359–9367, Vancouver, Canada, 2024. AAAI Press.

[Zhang *et al.*, 2024b] Weizhi Zhang, Liangwei Yang, Zihe Song, Henry Peng Zou, Ke Xu, Liancheng Fang, and Philip S. Yu. Do we really need graph convolution during training? light post-training graph-ode for efficient recommendation. In *CIKM*, pages 3248–3258, Boise, ID, USA, 2024. ACM.

[Zhao *et al.*, 2024] Mankun Zhao, Aitong Sun, Jian Yu, Xuewei Li, Dongxiao He, Ruiguo Yu, and Mei Yu. Reliable data augmented contrastive learning for sequential recommendation. *TBD*, 10(6):694–705, 2024.

[Zhou *et al.*, 2020] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, pages 1893–1902, Ireland, 2020. ACM.

[Zhou *et al.*, 2023] Hao Zhou, Geng Yang, Yang Xiang, Yunlu Bai, and Weiya Wang. A lightweight matrix factorization for recommendation with local differential privacy in big data. *TBD*, 9(1):160–173, 2023.