Elastic Graph Neural Networks for Session Recommendation

Anonymous ACL submission

Abstract

Session-based recommendation aims to predict users' next actions from anonymous interaction sessions. Existing approaches commonly adopt L2-based smoothing to enforce global stability and convergence; however, such strategies often overlook fine-grained local variations within session sequences and fail to fully exploit the adaptive modeling capacity of graph neural networks (GNNs). To overcome these limitations, we propose SR-EGNN, an Elastic Graph Neural Network for session-based recommendation that explicitly models adaptivity at both local and global levels. Specifically, the proposed framework decomposes adaptivity into two complementary components: local adaptivity, which captures subtle and dynamic variations within individual sessions, and global adaptivity, which preserves overall smoothness and training stability across the session graph. By jointly modeling these two aspects, SR-EGNN achieves a more effective trade-off between local sensitivity and global consistency than conventional smoothing-based methods. Concretely, SR-EGNN first encodes session representations using GNNs equipped with multi-head attention, and then applies dedicated local and global smoothing mechanisms to refine the representations. Extensive experiments on two real-world benchmark datasets demonstrate that SR-EGNN consistently outperforms strong baselines, validating the effectiveness of the proposed elastic modeling framework. The implementation is publicly available at https://gitcode.com/Zane507/SR-EGNN

1 Introduction

With the exponential growth of online information, recommendation systems have become essential for alleviating information overload and enabling users to efficiently access relevant content. Conventional recommendation approaches typically rely on user profiles and long-term behavioral histories to deliver personalized recommendations. However, in many real-world scenarios, user identity information is unavailable, and only short-term interaction data within a single session can be observed. This setting gives rise to the session-based recommendation task, which aims to predict a users next action solely based on anonymous click sequences in the current session. Despite the lack of persistent user identifiers, session-based recommendation is of substantial practical importance, as it supports effective personalization under severe data sparsity and privacy constraints.

A wide range of methods have been proposed for session-based recommendation. Early approaches such as POP and S-POP rank items by their occurrence frequency in the training data. Item-KNN (Davidson et al., 2010) applies item-based collaborative filtering by constructing item-to-item similarity matrices using cosine similarity. Neural sequence models, exemplified by GRU4Rec (Hidasi et al., 2015b), leverage recurrent neural networks to capture sequential dependencies for next-item prediction. NARM (Li et al., 2017) introduces an attention mechanism to emphasize informative items and infer the main user intent within a session. SR-GNN (Wu et al., 2019) models session sequences as graphs and employs gated graph neural networks with self-attention over the last clicked item. More recently, Amazon-M2 (Jin et al., 2024) provides a large-scale multilingual and multi-locale shopping session dataset, facilitating comprehensive empirical evaluation. SRGI-CM (Wang et al., 2024b) enhances dynamic item-transition modeling by incorporating global item-transition information, while Re2LLM (Wang et al., 2025) explores large language models guided by lightweight retrieval agents for session-based recommendation.

Despite their strong empirical performance, existing methods still face fundamental challenges.

First, accurately modeling user preferences from highly sparse session data remains difficult. Sessions are anonymous and typically contain only a few implicit behavioral signals, making it challenging to construct reliable user representations or effective retrieval agents. Most neural approaches encode user intent implicitly through hidden representations (e.g., SR-GNN), while large-model-based methods depend on auxiliary retrieval mechanisms (e.g., Re2LLM); both are constrained by the limited information available within a single session. Second, although prior studies have shown that information enhancement plays a crucial role in session-based recommendation (Wang et al., 2024b, 2025), existing approaches primarily focus on global item interaction enhancement or task-specific feedback modeling, while largely overlooking the elastic adaptability of session representationsnamely, the ability to flexibly adjust representation structures in response to diverse and evolving session dynamics.

To address these limitations, we propose SR-EGNN, an Elastic Graph Neural Network framework for session-based recommendation that explicitly models elasticity adaptability within sessions. Elastic Graph Neural Networks (ElasticGNNs) (Liu et al., 2021) are designed to learn flexible graph representations and have demonstrated strong performance in various domains, including natural language processing and computer vision, such as text embedding (Daneshfar et al., 2024), image classification (Qin and Qian, 2024), and edge prediction (Li et al., 2023). Inspired by these advances, we introduce elastic modeling mechanisms into session-based recommendation for the first time. By enhancing both local and global elasticity in graph-based session representations, SR-EGNN captures structural variations that are difficult to model using conventional methods such as NARM, SR-GNN, and Re2LLM, enabling more accurate next-item prediction under sparse and anonymous session settings.

Figure 1 illustrates the overall framework of SR-EGNN. Each session is modeled as a directed graph, consisting of input and output graphs along with an association matrix. The graphs are processed by graph neural networks to obtain node representations, which are aggregated via a multi-head attention mechanism. A soft attention module integrates session-level information, followed by a local elastic operation to enhance intra-session adaptability. Subsequently, a global elastic operation is performed using the association matrix to capture cross-session elasticity. Finally, SR-EGNN predicts the click probabilities of candidate items for the next interaction. Extensive experiments on real-world datasets demonstrate that SR-EGNN consistently outperforms strong baselines and achieves state-of-the-art performance.

The main contributions of this work are summarized as follows:

- We are the first to introduce elastic graph neural networks into session-based recommendation, explicitly modeling elasticity adaptability throughout the recommendation process.
- We propose a novel message-passing framework that jointly captures both local and global elasticity in session modeling.
- Extensive experiments on benchmark datasets demonstrate the effectiveness of SR-EGNN, yielding consistent and significant improvements over state-of-the-art methods.

2 Related Work

In this section, we will introduce works related to our method, including session recommendation, graph neural networks, elasticity mechanism.

2.1 Session Recommendation

In session recommendation, it is common to predict items of interest to users based on the order in which they click on them. PME (Wu et al., 2013) proposed a personalized Markov embedding model that maps users and items to Euclidean space, where the user item distance and item item distance reflect their respective importance. GRU4Rec (Hidasi et al., 2015a) introduced recurrent neural networks into session-based recommendation tasks for the first time, addressing the inaccuracy of matrix factorization methods.Each item in the session has different importance for predicting future behavior, so many of the recommended methods use attentional mechanisms to assign weight. (Wang et al., 2018) proposes an effective attention based transaction embedding model that strengthens relevant items and dilutes items unrelated to the next choice. (Liu et al., 2018) proposed a new short-term memory priority model to address the issue of user interest drift caused by accidental clicks in long-term conversations. This model captures the overall interest of

users in long-term conversations and considers the current interest based on the short-term memory of the last click. (Pan et al., 2020) designed a self-attention mechanism to correct the importance of items in a sequence and predict users' long-term preferences.(Jin et al., 2024) proposes a multilingual dataset to enhance the personalization and understanding of user preferences and benchmarks a series of algorithms. (Zhang et al., 2024) identifies and distinguishes the impact of ID and modality on modeling user behavior. Although these methods have proven effective and have advanced research in the session-based recommendation domain, they all overlook the elasticity adaptability in session modeling. This oversight can lead to models overfitting the noise in the data, preventing them from achieving optimal performance.

2.2 Graph Neural Networks

In recent years, Graph Neural Networks (GNNs) have played a significant role in various fields, such as natural language processing (Mikolov et al., 2013), image segmentation (Felzenszwalb and Huttenlocher, 2004), citation networks, knowledge graphs (Kipf and Welling, 2022), and recommendation systems (He et al., 2020). However, due to the heterogeneity between node classification tasks and session-based recommendation tasks, some research efforts have been proposed in the field of session-based recommendation. SRGNN (Wu et al., 2019) was the first to apply graph neural network algorithms to the field of session-based recommendation, aiming to capture the complex transition relationships between items. (Xu et al., 2019) used Graph Neural Networks (GNNs) to capture rich local dependencies and constructed a Broadly Connected Sequence (BCS) graph to connect different sequences, proposing a new Full Graph Neural Network (FGNN) to learn complex item dependencies. (Yu et al., 2020) proposed a sequence-based model that adaptively activates different user interests for different predicted items. (Wang et al., 2024a) proposed a knowledge graph-based session adaptive propagation algorithm that adaptively aggregates the neighbor information of items, addressing the issue that the intent of each session cannot adaptively change. Although GNNs have strong capabilities in handling graph data and have been widely applied in the field of session-based recommendation, existing methods have overlooked the elasticity adaptability of GNNs in session modeling. Therefore, our method is based on GNN modeling and considers the elasticity adaptability in session modeling.

3 Proposed Method

We propose elastic graph neural network for session recommendation. First, following (Wu et al., 2019), we construct the information within and between nodes of session s_i, including the output information of nodes $\mathbf{A}_{s_i}^{\text{out}}$ and the input information of nodes $\mathbf{A}_{s_i}^{\text{in}}$. Unlike previous methods, we introduce the information between nodes and edges Δ_{s_i}, which is often overlooked in historical session modeling. Second, we learn the session embedding through graph neural networks and a multi-head attention mechanism. Then, we perform a local elasticity operation on the items within the session. Subsequently, we execute a global elasticity operation. Finally, we combine the results of global and local elasticity to recommend the next item. We present the architecture of our model in Figure 1.

3.1 Notations

We use S to denote all session sequences. The i-th session sequence $s_i \in S$ is represented as $\{v_1^{s_i}, v_2^{s_i}, \cdots, v_m^{s_i}\}$, where m denotes the number of items in s_i. $\mathbf{A}^{\text{in}}$ represents the relationships where nodes are pointed to by other nodes within the session, and $\mathbf{A}^{\text{out}}$ represents the relationships where nodes point to other nodes.

3.2 Constructing session graph

To clearly express the internal relationships between nodes, we provide a case. In S, the transition relationships between nodes can be described as $\mathbf{A}$, where $\mathbf{A}_{case} \in \mathbf{A}$.

$$\mathbf{A}_{case} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{1}$$

$\mathbf{A}_{case}$ is a case we present, and it is further divided into $\mathbf{A}_{case}^{\text{in}}$ and $\mathbf{A}_{case}^{\text{out}}$.

$$\mathbf{A}_{case}^{\text{in}} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \end{bmatrix} \mathbf{A}_{case}^{\text{out}} = \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \tag{2}$$

where the term $\mathbf{A}_{case}^{\text{in}}$ represents the relationships where node is pointed to by other nodes within a

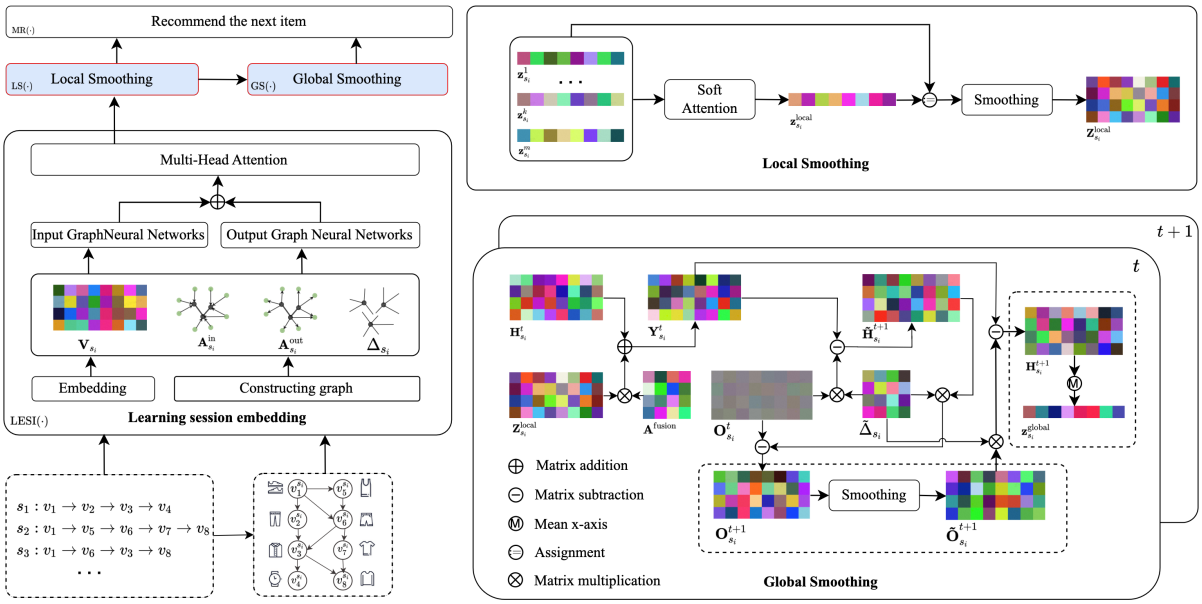


Figure 1: The workflow of the proposed SR-EGNN method.

session, while $\mathbf{A}_{case}^{\text{out}}$ represents the relationships where node points to other nodes.

In the session, not only does it include the relationships between nodes, but it also includes the relationships between nodes and edges. We use Δ to represent the relationships between edges and nodes, where $\Delta_{case} \in \Delta$.

$$\Delta_{case} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

The term Δ_{case} represents the relationships between different edges and nodes in $\mathbf{A}_{case}$ after removing bidirectional connections and self-loops.

3.3 Learning session embedding

We initialize the embedding of items in each session to obtain $\{\mathbf{v}_{s_i}^1, \mathbf{v}_{s_i}^2, \cdots, \mathbf{v}_{s_i}^m\} \in \mathbf{V}_{s_i}$. Then, we use graph neural network to learn the spatial structural features of the input graph $\mathbf{A}_{s_i}^{\text{input}}$ and the spatial structural features of the output graph $\mathbf{A}_{s_i}^{\text{out}}$ for $\mathbf{V}_{s_i}$.

$$\mathbf{G}_{s_i} = \mathbf{A}_{s_i}^{\text{in}} \mathbf{V}_{s_i} \mathbf{W}_1 + \mathbf{A}_{s_i}^{\text{out}} \mathbf{V}_{s_i} \mathbf{W}_2 + \mathbf{b} \tag{4}$$

where $\mathbf{A}_{s_i}^{\text{in}} \in \mathbb{R}^{m \times m}$, $\mathbf{A}_{s_i}^{\text{out}} \in \mathbb{R}^{m \times m}$, $\mathbf{V}_{s_i} \in \mathbb{R}^{m \times d}$, $\mathbf{W} \in \mathbb{R}^{d \times 3d}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{W}$ and bepresent learnable parameters, and d denotes the dimension of the hidden layer.

$$\mathbf{Z}_{s_i} = \text{MultiHeadAttention}(\mathbf{G}_{s_i}) \tag{5}$$

where $\mathbf{G}_{s_i} \in \mathbb{R}^{d \times 3d}$, $\mathbf{Z}_{s_i} \in \mathbb{R}^{d \times d}$.

3.4 Local elasticity

By learning the embedding of the s_i items, we obtain $\{\mathbf{z}_{s_i}^1, \mathbf{z}_{s_i}^2, \cdots, \mathbf{z}_{s_i}^m\} \in \mathbf{Z}_{s_i}$. Then, we learn the local embeddings of the items and perform elasticity operation.

In s_i, we first define the attention coefficient for the j-th term as follows:

$$\alpha_j = \mathbf{q}^\top \sigma\left(\mathbf{W}_3 \mathbf{z}_{s_i}^m + \mathbf{W}_4 \mathbf{z}_{s_i}^j + \mathbf{c}\right) \tag{6}$$

where $j \in [1, m]$, $\mathbf{W}_3$, $\mathbf{W}_4 \in \mathbb{R}^{d \times d}$, and $\mathbf{q}$, $\mathbf{c} \in \mathbb{R}^{\mathbf{d}}$ represents learnable weight parameters.

Local embedding is defined as the embedding operation on the last k items clicked within s_i.

$$\mathbf{z}_{s_i}^{\text{local}} = \sum_{j=m-k}^{m} \alpha_j \mathbf{z}_{s_i}^j \tag{7}$$

Then, we assign $\mathbf{z}_{s_i}^{\text{local}}$ to the last clicked item.

$$\mathbf{z}_{s_i}^{\text{m}} := \mathbf{z}_{s_i}^{\text{local}} \tag{8}$$

where $:=$ denotes the reassignment of the value to $\mathbf{z}_{s_i}^{\text{m}}$. Then, we obtain $\{\mathbf{z}_{s_i}^1, \mathbf{z}_{s_i}^2, \cdots, \mathbf{z}_{s_i}^{\text{local}}\} \in \mathbf{Z}_{s_i}^{\text{local}}$. Finally, we perform elasticity operation using the l norm.

$$\mathbf{z}_{s_i}^{\text{local}} := \begin{cases} \text{sign}(\mathbf{z}_{s_i}^{\text{local}}) \cdot \min(|\mathbf{z}_{s_i}^{\text{local}}|, \lambda_1) & (\textbf{OptionI}: l_1 \text{ norm}) \\ \dfrac{\mathbf{z}_{s_i}^{\text{local}}}{||\mathbf{z}_{s_i}^{\text{local}}||_2} \cdot \min(||\mathbf{z}_{s_i}^{\text{local}}||_2, \lambda_1) & (\textbf{OptionII}: l_{21} \text{ norm}) \end{cases} \tag{9}$$

where the reassigned $\mathbf{z}_{s_i}^{\text{local}}$ represents the result of local elasticity using either the l_1 norm or the l_{21} norm.

3.5 Global elasticity

After performing local elasticity to obtain $\mathbf{Z}_{s_i}^{\text{local}}$, we proceed with a global elasticity operation. The global elasticity operation integrates the relationships between nodes as well as between nodes and edges. We initialize $\mathbf{H}_{s_i}^0 = \mathbf{Z}_{s_i}^{\text{local}}$, and then

$$\mathbf{Y}_{s_i}^t = \gamma \mathbf{H}_{s_i}^0 + (1-\gamma)\mathbf{A}_{s_i}^{\text{fusion}} \mathbf{H}_{s_i}^t \tag{10}$$

where $\mathbf{A}_{s_i}^{\text{fusion}} = \mathbf{A}_{s_i}^{\text{in}} + \mathbf{A}_{s_i}^{\text{out}}$, $\mathbf{A}_{s_i}^{\text{fusion}} \in \mathbb{R}^{m \times m}$, $\mathbf{H}_{s_i}^t \in \mathbb{R}^{m \times d}$, γ represents a hyperparameter.

$$\tilde{\mathbf{H}}_{s_i}^{t+1} = \mathbf{Y}_{s_i}^t - \gamma \tilde{\Delta}_{s_i}^{\mathrm{T}} \mathbf{O}_{s_i}^t \tag{11}$$

where $\tilde{\Delta}_{s_i} \in \tilde{\Delta}$, $\tilde{\Delta} = \Delta \widehat{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\Delta}_{ij}$, $\mathbf{O}_{s_i}^0 = \mathbf{0}^{m \times d}$, $\mathbf{0}^{m \times d}$ represents all matrices whose values are 0.

$$\mathbf{O}_{s_i}^{t+1} = \mathbf{O}_{s_i}^t - \beta \tilde{\Delta}_{s_i} \tilde{\mathbf{H}}_{s_i}^{t+1} \tag{12}$$

where β is a hyperparameter. Then, we perform the elasticity operation using the l-norm.

$$\tilde{\mathbf{O}}_{s_i}^{t+1} = \begin{cases} \operatorname{sign}(\mathbf{O}_{s_i}^{t+1}) \cdot \min(|\mathbf{O}_{s_i}^{t+1}|, \lambda_1) & (\textbf{OptionI}: l_1 \text{ norm}) \\ \dfrac{\mathbf{O}_{s_i}^{t+1}}{||\mathbf{O}_{s_i}^{t+1}||_2} \cdot \min(||\mathbf{O}_{s_i}^{t+1}||_2, \lambda_1) & (\textbf{OptionII}: l_{21} \text{ norm}) \end{cases} \tag{13}$$

where $\tilde{\mathbf{O}}$ represents the result after elasticity using either the l_1 norm or the l_{21} norm.

$$\mathbf{H}_{s_i}^{t+1} = \mathbf{Y}_{s_i}^t - \gamma \tilde{\Delta}_{s_i}^{\mathrm{T}} \tilde{\mathbf{O}}_{s_i}^{t+1} \tag{14}$$

The global elasticity is iteratively executed T times, ultimately resulting in $\mathbf{H}_{s_i}^T$.

$$\mathbf{z}_{s_i}^{\text{global}} = \text{MEAN}(\mathbf{H}_{s_i}^T, \text{axis} = 0) \tag{15}$$

where $\text{MEAN}(\cdot, \text{axis} = 0)$ indicates taking the mean of $\mathbf{H}_{s_i}^T$ along the x-axis, where $\mathbf{H}_{s_i}^T \in \mathbb{R}^{m \times d}$, and $\mathbf{z}_{s_i}^{\text{global}} \in \mathbb{R}^d$.

3.6 Recommend the next item

After applying local elasticity and global elasticity, we obtained $\mathbf{z}_{s_i}^{\text{local}}$ and $\mathbf{z}_{s_i}^{\text{global}}$, respectively. We then fuse the local and global elasticity using learnable weight parameters.

$$\mathbf{z}_{s_i}^{\text{lg}} = \mathbf{W}_5 \text{CONCAT}(\mathbf{z}_{s_i}^{\text{local}}; \mathbf{z}_{s_i}^{\text{global}}) \tag{16}$$

where $\mathbf{W}_5 \in \mathbb{R}^{d \times 2d}$ represents learnable weight parameters, and $\text{CONCAT}(\cdot)$ denotes the vector concatenation operation.

After obtaining $\mathbf{z}_{s_i}^{\text{lg}}$, we calculate the score for each item.

$$\hat{\mathbf{z}}_{s_i}^j = \mathbf{z}_{s_i}^{\text{lg}} [\mathbf{v}_{s_i}^j]^{\mathrm{T}} \tag{17}$$

where $\mathbf{z}_{s_i}^{\text{lg}} \in \mathbb{R}^d$, $\mathbf{v}_{s_i}^{\text{j}} \in \mathbb{R}^d$. Then, we use the softmax function to obtain the final output vector of the model.

$$\hat{\mathbf{y}} = \text{softmax}(\hat{\mathbf{z}}_{s_i}) \tag{18}$$

where $\hat{\mathbf{z}}_{s_i} \in \mathbb{R}^c$, c represents the number of candidate prediction items. Then, we use the cross-entropy loss function to calculate the loss between the candidate prediction items and the true values.

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{c} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \tag{19}$$

It is worth noting that the detailed experimental parameters are documented in the experimental details.

4 Experiments

In this section, we introduce the datasets used to validate the effectiveness of our method, the evaluation metrics for the experiments, the baseline methods for comparison, and the parameter settings of our method. Additionally, we provide a detailed analysis of the performance metrics of our proposed method, including improvements in the P@k and MRR@k metrics. Furthermore, we conduct an in-depth analysis of the hyperparameters of our method. Finally, we present the results of the ablation study for the proposed method.

4.1 Datasets

Yoochoose and Diginetica are two commonly used real-world datasets based on session recommendation. The two datasets are derived from the clickstream of users to e-commerce websites within six months of the release of the 2015 RecSys Challenge and transaction data of user clicks recorded within six months of the 2016 CIKM Cup platform. For a more fair comparison, according to (Li et al., 2017; Liu et al., 2018), we filtered out all items with a length of 1 and appearing less than 5 times in both datasets. The statistical data of the dataset is summarized in Table 1. Reference to (Tan et al., 2016), We generate sequences and corresponding labels by splitting the input sequences. For example, there is a sequence $\mathrm{S} = \{v_1, v_3, v_5, v_2\}$, The sequence we generate is labeled $(\{v_1\}, v_3)$,$(\{v_1, v_3\}, v_5)$,$(\{v_1, v_3, v_5\}, v_2)$, and the label represents the item that the user clicks next.

Table 1: Statistics of datasets used in the study.

Statistics	Yoochoose 1/64	Diginetica
clicks	557,248	982,961
training sessions	369,859	719,470
test sessions	55,898	60,858
items	16,766	43,097
average length	6.16	5.12

4.2 Evaluation Metrics

Both P@k and MRR@k are widely used as evaluation metrics. The value of k is set to 5, 10 and 20.

P@k: We use P@k (Precision) score as a measure of precision of prediction. P@k represents the proportion of all test cases which has correctly recommended items amongst the top-k items.

$$\mathrm{P@k} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} c_i \tag{20}$$

where N denotes the number of test data, c_i is 0 or 1. If the predicted result has appeared in the top-k items, the value of c_i is 1. Otherwise the value of c_i is 0.

MRR@k: Another evaluation metric is MRR@k (Mean Reciprocal Rank). It is the average of reciprocal ranks of all desire items. We set the reciprocal rank to 0 if the rank exceeds k. It is important in the field of recommendation systems because it takes into account the ranking order of recommendation list.

$$\mathrm{MRR@k} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} \frac{1}{\mathrm{Rank}(i)} \tag{21}$$

where N denotes the number of test data. If the predicted result has appeared in the top-k items, the value of $\mathrm{Rank}(i)$ is the position index corresponding to the item. Otherwise the value of $\frac{1}{\mathrm{Rank}(i)}$ is 0.

4.3 Baseline methods

POP and S-POP : Recommend the sequence item by its frequency in the training set, and select N items with the highest frequency as the recommendation item.

item-knn (Davidson et al., 2010): uses item-based collaborative filtering algorithm, and establishes item-to-item similarity matrix by cosine distance. Sequence recommendation based on item similarity.

BRP-MF (Rendle et al., 2012): This is a Bayesian personalized ranking algorithm. Many algorithms do not directly optimize the personalized ranking. BRF-MF algorithm proposes a general personalized ranking optimization standard and applies it to matrix factorization and adaptive KNN algorithm.

FPMC(Rendle et al., 2010): combines matrix-based methods with Markov chains. In addition, Bayesian personalized ranking is introduced when learning model parameters.

GRU4Rec(Hidasi et al., 2015b): This is a deep learning method for sequence prediction. Sequence is a kind of sequential information, so recurrent neural networks can learn this kind of sequential information. In GRU4Rec, the recurrent neural network is applied to sequence prediction.

NARM(Li et al., 2017): Based on the previous sequence prediction, which only considers the sequence information of sequence prediction, the important items in sequence are not considered. Therefore, NARM algorithm considers the importance of sequence items through attention mechanism to capture the main purpose of users and make recommendations.

STAMP(Liu et al., 2018): User click prediction STAMP considers the user's last click item and integrates the user's last click item with the whole interest.

STAN(Garg et al., 2019): This method considers the position of an item in the current sequence, the most recent sequence should be recommended, and the consideration of sequence prediction in neighboring sequence.

SRGNN(Wu et al., 2019): This method uses gated GNN to get item embeddings and self-attention with the last item in the sequence.

MTD(Huang et al., 2021): This approach leverages an automatic and hierarchical approach to jointly learn the dynamics of intra-sequence and inter-sequence item transitions.

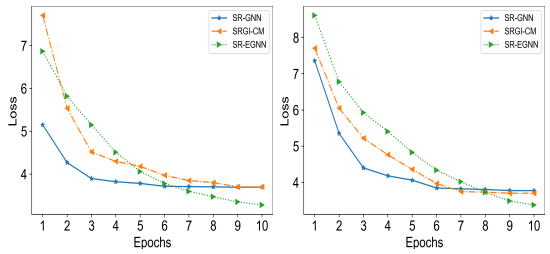
SRGI-CM(Wang et al., 2024b): This method introduces global item-transition information to enhance the modeling of dynamic item transitions.

4.4 Parameter settings

We use pytorch to implement our model. The latent vector dimension d of the two datasets is set to 100. We specify the regularization penalty $\lambda = 10^{-6}$. Adam optimizer was used to optimize all models, and the batch size and learning rate

Table 2: Performance of SR-EGNN and other baseline methods on two datasets.

Method	Yoochoose 1/64						Diginetica					
	P@5	MRR@5	P@10	MRR@10	P@20	MRR@20	P@5	MRR@5	P@10	MRR@10	P@20	MRR@20
POP	5.39	1.27	7.17	1.44	6.71	1.65	0.58	0.02	0.75	0.03	0.89	0.20
S-POP	20.57	16.62	26.82	17.63	30.44	18.35	11.06	11.02	16.21	12.71	21.06	13.68
Item-KNN	35.18	19.97	44.64	20.56	51.60	21.81	18.62	9.28	26.93	10.67	35.75	11.57
BPR-MF	21.72	11.08	26.86	11.74	31.31	12.08	3.27	1.11	4.26	1.70	5.24	1.98
FPMC	31.25	13.49	39.80	14.46	45.62	15.01	14.09	5.18	20.29	5.97	26.53	6.95
GRU4REC	41.09	20.80	52.53	22.13	60.64	22.89	15.94	2.36	21.84	2.92	29.45	8.33
NARM	46.23	25.58	58.84	27.53	68.32	28.63	26.31	13.69	37.22	14.53	49.70	16.17
STAMP	46.67	27.12	59.20	28.27	68.74	29.67	24.04	11.53	34.01	12.59	45.64	14.32
STAN	46.86	25.79	60.27	27.90	69.45	28.74	26.03	13.89	37.32	15.34	50.13	16.73
SR-GNN	47.55	28.44	60.49	30.18	70.57	30.94	26.28	14.97	37.37	16.44	50.73	17.59
MTD	48.57	**29.02**	61.44	**31.11**	70.37	31.02	28.29	**15.66**	40.22	**17.58**	53.92	**19.20**
SRGI-CM	**50.12**	29.13	**63.51**	31.20	**70.31**	**30.27**	**29.13**	17.13	**41.32**	18.23	**54.71**	19.43
SR-EGNN	**55.21**	**38.72**	**64.12**	**41.13**	**71.61**	**41.37**	**40.68**	**29.51**	**47.6**	**31.68**	**55.2**	**32.77**



(a) Yoochoose 1/64 (b) Diginetica

Figure 2: The convergence of SR-EGNN with baselines on the Yoochoose 1/64 and Diginetica.

were 100 and 0.001 respectively, where the learning rate was decayed to 1/10 of the original after every 3 epochs. All parameters were initialized using a Gaussian distribution with mean 0 and standard deviation 0.1. We set the number of eopch that is 10. During global elasticity, the number of iterations $T \in [1, 15]$. Apart from this, we directly use the entire training set to train the model. After the model training is completed, we select the model that converges best on the training set as the optimal model and test it on the test set.

4.5 Performance results

In Table 2, we present the results of the SR-EGNN model with the best convergence performance after 10 iterations ($E = 10$). When performance is optimal, the values of T are 5 and 8 for the Yoochoose 1/64 and Diginetica datasets, respectively. At this point, our method shows new improvements in both P@k and MRR@k, with significant enhancements particularly in the MRR@k. On Yoochoose 1/64, when k is set to 5, 10, and 20, SR-EGNN improves by 11.4%, 2.6%, and 0.3% over the best baseline, respectively. Notably, in terms of the MRR metric, it improves by 33.43%, 32.21%, and 31.17%, respectively. On Diginetica, when k is set to 5, 10, and 20, SR-EGNN shows improvements of 42.84%, 16.44%, and 2.3% in the P@k over the best baseline, respectively. Specifically, in terms of the MRR@k, it improves by 95.95%, 80.2%, and 70.7%, respectively.

Generally, some baseline methods focus more on the P@20 metric, such as SRGNN and MTD. However, items ranked higher are more likely to be clicked. On the Yoochoose 1/64 and Diginectica datasets, SR-EGNN provides experimental results for the P@5, P@10, and P@20 metrics, showing further improvement compared to baseline methods. The higher ranking of session recommendation items and the higher precision when k is smaller contribute to the improvements in the P@5 and P@10 metrics, which are attributed to the enhancement in the MRR performance metric. SR-EGNN achieves further improvement in MRR@k, thanks to its embedding learning for session items, local elasticity mechanism, and global elasticity mechanism.

4.6 Convergence analysis

To understand the differences between our proposed method and the baseline methods, we conducted a convergence analysis. The analysis results demonstrate the convergence of SR-EGNN compared to SR-GNN and MTD on the training datasets Yoochoose 1/64 and Diginetica Loss. As shown in Figure 2, SR-EGNN exhibits a more linear convergence trend compared to SR-GNN and MTD, achieving the best convergence result for the training set loss at 10 epochs.

Table 3: Performance of SR-EGNN with different k-value.

Data	k=2		k=3		k=4		k=5		k=6		k=8	
	P@5	MRR@5	P@5	MRR@5	P@5	MRR@5	P@5	MRR@5	P@5	MRR@5	P@5	MRR@5
Yoochoose 1/64	56.37	39.48	**56.21**	**39.83**	55.69	39.59	55.37	38.83	55.56	38.54	55.16	38.26
Diginetica	25.07	15.22	**41.1**	**29.9**	40.85	29.43	40.85	29.46	40.64	29.81	29.17	21.91
	P@10	MRR@10	P@10	MRR@10	P@10	MRR@10	P@10	MRR@10	P@10	MRR@10	P@10	MRR@10
Yoochoose 1/64	64.88	42.16	**64.97**	**41.94**	64.63	41.33	64.29	41.32	64.49	41.03	64.13	40.81
Diginetica	34.71	16.42	**47.84**	**32.11**	47.87	31.55	47.91	31.99	47.67	31.86	33.11	23.66
	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20
Yoochoose 1/64	72.35	42.54	**72.53**	**42.45**	72.21	41.79	72.03	41.39	71.87	41.51	71.65	41.15
Diginetica	45.53	17.05	55.09	33.63	**55.47**	**32.98**	55.44	33.3	55.15	33.24	37.69	24.93

Table 4: Performance metric of SR-EGNN with different ablation module.

Datasets	Ablation Module			Metric					
	LESI$(\cdot)$	LS$(\cdot)$	GS$(\cdot)$	P@5	MRR@5	P@10	MRR@10	P@20	MRR@20
Yoochoose 1/64	✓	✓	×	41.37	26.43	53.06	26.93	63.41	26.74
	✓	×	✓	7.42	4.61	12.15	4.69	20.96	4.6
	×	✓	✓	42.4	27.07	54.95	26.78	65.79	26.67
Diginetica	✓	✓	×	1.83	0.71	2.95	0.89	4.51	1.16
	✓	×	✓	2.36	1.48	4.08	1.51	6.6	1.76
	×	✓	✓	21.0	13.25	31.34	13.13	43.17	13.27

4.7 Parameter analysis

4.7.1 The local elasticity k-value analysis

In the local elasticity, the value of k represents the last k items clicked by the user in session recommendation task. In Table 3, we analyze the impact of different values of k on the experimental results. Compared to the performance where k is set to 7 in Table 2, there is a further improvement in P@k and MRR@k in Table 3. On one hand, this indicates that the SR-EGNN can further enhance performance metrics by adjusting hyperparameters. On the other hand, it highlights the importance of modeling the last k items.

4.8 Ablation studies

We conducted ablation experiments on the different modules of our proposed method, with the experiments still carried out on the Yoochoose 1/64 and Diginetica datasets. The modules subjected to ablation include the session embedding learning module LESI$(\cdot)$, the local elasticity module LS$(\cdot)$, and the global elasticity module GS$(\cdot)$. In Table 4, a ✓ indicates that the corresponding module is used in the modeling, while a × indicates the opposite. Compared to the experimental results in Table 2, in Table 4, removing any one of the modules LESI$(\cdot)$, LS$(\cdot)$, or GS$(\cdot)$ results in SR-EGNN exhibiting poor performance, highlighting the importance of each module. Specifically, when the LS$(\cdot)$ module is removed, the experimental results show poor performance on both the Yoochoose 1/64 and Diginetica datasets. However, when the LESI$(\cdot)$ module is removed, the performance metrics are not as poor. Therefore, the LS$(\cdot)$ module might deserve more attention in modeling compared to the LESI$(\cdot)$ module.

5 Conclusion

In this paper, we propose SR-EGNN, an elastic graph neural network framework for session-based recommendation. Extensive experiments on the Yoochoose 1/64 and Diginetica datasets demonstrate that SR-EGNN consistently outperforms strong baselines in terms of both P@k and MRR@k. In particular, when $k \in \{5, 10, 20\}$, SR-EGNN achieves substantial improvements in MRR@k over the best baseline, with gains of 33.43%, 32.21%, and 31.17% on Yoochoose 1/64, and 95.95%, 80.20%, and 70.70% on Diginetica, respectively. By explicitly modeling both local and global elasticity, SR-EGNN captures fine-grained structural variations in session sequences while learning effective session representations. The progressive learning enabled by the elastic modules allows the model to adapt to diverse session dynamics, leading to superior recommendation performance. Furthermore, convergence analysis, hyperparameter studies, and ablation experiments consistently validate the effectiveness and robustness of the proposed framework.

6 Limitations

Despite its strengths, SR-EGNN has several limitations. First, it may struggle with very short sessions, as capturing fine-grained local variations in limited interactions is challenging. Second, the computational complexity introduced by multi-head attention and graph neural networks (GNNs) could hinder scalability. Additionally, the model is vulnerable to cold-start and data sparsity issues, which restrict its performance for new users or items. Finally, SR-EGNN's interpretability is limited by its complex architecture, making it difficult to understand how specific session patterns drive predictions.

References

Fatemeh Daneshfar, Sayvan Soleymanbaigi, Ali Nafisi, and Pedram Yamini. 2024. Elastic deep autoencoder for text embedding clustering by an improved graph regularization. *Expert Systems with Applications*, 238:121780.

J. Davidson, B. Livingston, D. Sampath, B. Liebald, J. Liu, P. Nandy, T Van Vleet, U. Gargi, S. Gupta, and Y. He. 2010. The youtube video recommendation system. In *Acm Conference on Recommender Systems*.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2004. Efficient graph-based image segmentation. *Int. J. Comput. Vis.*, 59(2):167–181.

Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. Sequence and time aware neighborhood for session-based recommendations: Stan. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1069–1072.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. *CoRR*, abs/2002.02126.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2015a. Session-based recommendations with recurrent neural networks. *Computer ence*.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015b. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Chao Huang, Jiahui Chen, Lianghao Xia, Yong Xu, Peng Dai, Yanqing Chen, Liefeng Bo, Jiashu Zhao, and Jimmy Xiangji Huang. 2021. Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4123–4130.

Wei Jin, Haitao Mao, Zheng Li, Haoming Jiang, Chen Luo, Hongzhi Wen, Haoyu Han, Hanqing Lu, Zhengyang Wang, Ruirui Li, Zhen Li, Monica Cheng, Rahul Goutam, Haiyang Zhang, Karthik Subbian, Suhang Wang, Yizhou Sun, Jiliang Tang, Bing Yin, and Xianfeng Tang. 2024. Amazon-m2: a multilingual multi-locale shopping session dataset for recommendation and text generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Thomas N Kipf and Max Welling. 2022. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428.

Yanan Li, Haitao Yuan, Zhe Fu, Xiao Ma, Mengwei Xu, and Shangguang Wang. 2023. Elastic: edge workload forecasting based on collaborative cloud-edge deep learning. In *Proceedings of the ACM Web Conference 2023*, pages 3056–3066.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 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*, pages 1831–1839.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849. PMLR.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking item importance in session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1837–1840.

Yalan Qin and Li Qian. 2024. Fast elastic-net multi-view clustering: A geometric interpretation perspective. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 10164–10172.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In

Proceedings of the 19th international conference on World wide web, pages 811–820.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 17–22.

Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Yu Wang, Amin Javari, Janani Balaji, Walid Shalaby, Tyler Derr, and Xiquan Cui. 2024a. Knowledge graph-based session recommendation with session-adaptive propagation. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 264–273.

Ziyan Wang, Yingpeng Du, Zhu Sun, Haoyan Chua, Kaidong Feng, Wenya Wang, and Jie Zhang. 2025. Re2llm: Reflective reinforcement large language model for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12827–12835.

Ziyang Wang, Wei Wei, Ding Zou, Yifan Liu, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2024b. Exploring global information for session-based recommendation. *Pattern Recognition*, 145:109911.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353.

Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 137–140.

Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph contextualized self-attention network for session-based recommendation. In *IJCAI*, volume 19, pages 3940–3946.

Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. Tagnn: target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1921–1924.

Xiaokun Zhang, Bo Xu, Zhaochun Ren, Xiaochen Wang, Hongfei Lin, and Fenglong Ma. 2024. Disentangling id and modality effects for session-based recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 18831892, New York, NY, USA. Association for Computing Machinery.