# HOW TO BRIDGE GRAPH AND SEQUENCE PATTERNS IN SESSION-BASED RECOMMENDATION? A SELF-SUPERVISED METHOD

*Xinglong Wu*[⋆]*, Hui He*[⋆]*, Zejun Wang*[⋆]*, Yu Tai*[⋆]*, Sheng Yin*[⋆]*, Hongwei Yang*[⋆]*, Weizhe Zhang*[⋆†]

[⋆] Harbin Institute of Technology
[†] Pengcheng Laboratory

## ABSTRACT

Session-based Recommendation aims to reveal the item distribution patterns in anonymous session sequences. Most existing approaches model the distribution patterns by utilizing either sequential or structural information individually to absorb different pattern knowledge, which can only model the distinct one-sided facet of item distribution in sessions, thus leading to suboptimal performance. Self-supervised learning provides a natural solution as a bridge to fill the gap between different learning paradigms in session-based recommendations, which remains unexplored. In this paper, we regard the distinct learning paradigm as an individual channel and then integrate the sequential and graphical channels with a contrastive bridge architecture. We name the novel framework **DC-Rec**, for **D**ual **C**hannel **Rec**ommendation, to model the comprehensive session characteristics. Extensive experiments conducted on two real-world datasets demonstrate that our model consistently outperforms the state-of-the-art recommendation methods.

***Index Terms***— Session-based Recommendation, Graph Neural Networks, Recurrent Neural Networks, Self-supervised Learning

## 1. INTRODUCTION

Session-based recommendation (SBR) has been prosperous recently in personalized recommendation scenarios due to its anonymity, where user profiles are invisible, and items are recommended solely based on users' historical interacted preferences. Since user profiles are unavailable, obtaining the item distribution in the session sequence becomes the core problem. However, the item sequences in the session are typically short and irregular, so how to reveal the item dependency relationship remains intractable.

Different session-based recommendation methods have been proposed to excavate the distribution pattern beneath the item transition sequence in the session. Current SBR methods can be roughly divided into two categories, i.e., SBRs based on (1) Recurrent Neural Networks (RNNs) and (2) Graph Neural Networks (GNNs). Specifically, RNN-based SBRs [1, 2] process the item transition sequence chronologically and predict the target item utilizing the sequential transition information. On the other hand, GNN-based SBRs [3, 4] capture the item distribution knowledge in the session by transforming the item sequence to session graphs, and thus excavate the structural distribution details and portray intricate user preference.

Focusing on different characteristic perspectives of sessions, GNN- and RNN-based SBR methods exhibit distinct advantages and limitations. Specifically, the RNN-based models reveal mainly the sequential patterns via processing the items in the single session linearly and thus possess advantages at item dependency modeling but neglect the global inter-session information and sophisticated structural distribution patterns. The GNN-based models reveal mainly the structural patterns by regarding the session as a graph and broaden the receptive fields by stacking multiple GNN layers. However, session graphs cannot model to-and-fro item sequential dependencies—a session graph could correspond to several session sequences.

Sequence- and graph-based SBR schemes inhibit each other and only capture one-sided item distribution patterns in the session. However, conventional SBR models only utilize either of the schemes without integrating them. A natural solution to overcome the aforementioned limitations is incorporating both schemes utilizing a self-supervised learning framework. Concretely, by regarding graph- and sequence-based patterns as two individual channels, we integrate the above two channels via a contrastive architecture. Thus, we propose a **D**ual-**C**hannel Framework for **Rec**ommendation, **DC-Rec** for brevity. In DC-Rec model, we absorb both graphical and sequential knowledge in sessions via an inter-channel contrastive bridge module. By maximizing the mutual information between two channels, DC-Rec provides flexibility and capability for recommendation.

Our main contributions are summarized as follows: 1) We highlight the significance of complementing the single channel knowledge and maximizing the mutual information between sequential and graphical patterns. 2) Regarding graph and sequence patterns as two individual channels, we propose a novel self-supervised framework called Dual-Channel Framework for Recommendation (DC-Rec). 3) We apply the

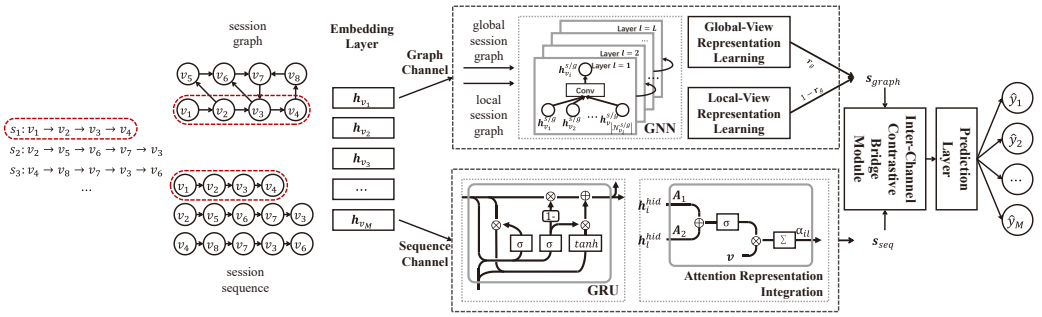



**Fig. 1**: Architecture of DC-Rec.

pre-trained item embeddings to reserve the single channel characteristic and boost the overall performance, simultaneously. 4) The extensive experiments conducted on 2 public datasets demonstrate that our model consistently outperforms other state-of-the-art SBR models.

## 2. METHODOLOGY

In this section, we introduce our DC-Rec[1] model that bridges the structural and sequential patterns in session-based recommendations. We first present the SBR problem statement, followed by detailed implementations (as shown in Fig. 1).

**Problem Statement:** Let $V = \{v_1, v_2, \cdots, v_M\}$ be the set of items that has appeared in all the sessions in recommendation, where $M$ is the number of items. A session $s = \{v_1^s, v_2^s, \cdots, v_l^s\}$ can be represented as a chronological list of items that is ordered by the user interaction timestamp, where $v_i^s \in V$ denotes the $i$-th item that the user has interacted with in the session $s$, and the length of the session $s$ is $l$. Given a session $s$, the goal of SBR is to predict the top-$N$ ($1 \leq N \leq M$) items that are most likely to be clicked by the user in the session $s$.

### 2.1. Graph Channel

**Local Session Graph Construction.** Each session can be modeled as a directed session graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where the vertex set $\mathcal{V}_s \subseteq V$ represents the candidate item set in session $s$, and the edge set $\mathcal{E}_s = \left\{e_{ij}^s\right\}$ represents the item transition relationships, with $e_{ij}^s$ connecting two chronologically neighbored items $\left(v_i^s, v_j^s\right)$, i.e., the user interacts with item $v_i^s$ and $v_j^s$ consecutively.

**Global Session Graph Construction.** Focusing on inter-session transition knowledge, we integrate all pairwise item transitions among sessions. We define the global-view session graph as $\mathcal{G}_g = (\mathcal{V}_g, \mathcal{E}_g)$, where $\mathcal{V}_g \subseteq V$ denotes the global-view item vertices, and $\mathcal{E}_g = \left\{e_{ij}^g\right\}$ denotes the set of global-view edges, with $e_{ij}^g$ corresponding to pairwise item tuple $\left(v_i^g, v_j^g\right)$ in all sessions. For any tuple $\left(v_i^g, v_j^g\right)$, we set the edge weight as frequency that the tuple has occurred, and only keep the Top-$N$ edges with the highest edge weights.

[1]We have released the Appendix, codes, and datasets publicly at https://www.github.com/WuXinglong-HIT/DC-Rec/.

#### 2.1.1. *Local-View Representation Learning (**LVRL**)*

We apply a lightweight GNN implementation as the aggregation method to obtain the target node representation:

$$\mathbf{h}_{v_i}^{s,(l)} = \sum_{v_j \in \mathcal{N}_{v_i}^s} \frac{1}{\sqrt{|\mathcal{N}_{v_i}^s||\mathcal{N}_{v_j}^s|}} \mathbf{h}_{v_j}^{s,(l-1)}, \tag{1}$$

where $\mathcal{N}_{v_i}^s$ denotes the neighbors of node $v_i$ in session $s$, and $\mathbf{h}_{v_i}^{s,(l)} \in \mathbb{R}^d$ represents the refined local embedding of node $v_i$ in session $s$ at the $l$th hop. With the maximum propagation scope $L$, we directly utilize the last layer representation $\mathbf{h}_{v_i}^s = \mathbf{h}_{v_i}^{s,(L)}$ to represent the local-view characteristics.

#### 2.1.2. *Global-View Representation Learning (**GVRL**)*

We utilize the unnormalized neighbor embedding aggregation to integrate global embeddings as follows:

$$\mathbf{h}_{v_i}^g = \sum_{v_j \in \mathcal{N}_{v_i}^g} \omega_{ij} \mathbf{h}_{v_j}^g, \tag{2}$$

where $\mathcal{N}_{v_i}^g$ indicates the one-hop adjacent vertices of $v_i$ in global-view session graph, and $\omega_{ij}$ denotes the weight of edge $e_{ij}^g$. We apply the weighted sum of the adjacent neighbors because we have normalized the global adjacency matrix $\mathcal{G}_g$ and applied a Top-$N$ strategy to keep the scale of different node neighbors consistent during the global session graph construction.

#### 2.1.3. *Inter-View Representation Integration*

We integrate local- and global-view item representation attentively by utilizing a self-adaptive query vector. First, we apply a gating mechanism to adjust the proportion of embeddings from different views adaptively:

$$\begin{aligned} \mathbf{h}_{v_i}' &= \mathbf{r}_g \odot \mathbf{h}_{v_i}^g + (1 - \mathbf{r}_g) \odot \mathbf{h}_{v_i}^s, \\ \mathbf{r}_g &= \sigma\left(\mathbf{W}_g \mathbf{h}_{v_i}^g + \mathbf{W}_s \mathbf{h}_{v_i}^s\right), \end{aligned} \tag{3}$$

where $\mathbf{r}_g \in \mathbb{R}^d$ denotes the gating vector for measuring the proportion, and $\mathbf{W}_g, \mathbf{W}_s \in \mathbb{R}^{d\times d}$ denote the linear transformation matrices. For any inter-view item representation $\mathbf{h}'_{v_i}$ in session $s$, we formulate the query vector by applying the mean-pooling operation as follows:

$$\mathbf{h}_s^q = \frac{1}{l}\sum_{i=1}^{l}\mathbf{h}'_{v_i}, \quad \alpha_i^s = \mathbf{h}_q^{s\top}\mathbf{h}'_{v_i}. \tag{4}$$

We utilize the mean-pooled inter-view representations of all items in session $s$ as the query vector to calculate the attention coefficient of each candidate item. We then attentively integrate the candidate item embeddings to acquire the graph-channel session representation $\mathbf{s}_{graph}$: $\mathbf{s}_{graph} = \sum_{i=1}^{l}\alpha_i^s\mathbf{h}'_{v_i}$, where $\alpha_i^s$ denotes the attention score of item $v_i$ over the session $s$, and we utilize an inner-product operation between the query vector $\mathbf{h}_q^s$ and candidate item representation $\mathbf{h}'_{v_i}$ to calculate the attention score.

#### *2.1.4. Graph Prediction Module*

Notably, by taking the graph channel as an independent end-to-end recommender system, we obtain graph channel predictions for all sessions and formulate the single graph channel prediction objective:

$$\begin{aligned}\hat{\mathbf{y}}_i &= softmax\left(\mathbf{s}_{graph}^{\top}\mathbf{h}'_{v_i}\right),\\ \mathcal{L}_g &= -\sum_{i=1}^{M}\mathbf{y}_i\log\left(\hat{\mathbf{y}}_i\right) + \left(1-\mathbf{y}_i\right)\log\left(1-\hat{\mathbf{y}}_i\right)\end{aligned} \tag{5}$$

### 2.2. Sequence Channel

In Sequence Channel, we treat the session as a session sequence and utilize a Gated Recurrent Unit (GRU) to model the sequential transition patterns:

$$\begin{aligned}\mathbf{r}_i &= \sigma\left(\mathbf{W}_r\mathbf{h}_{v_i} + \mathbf{U}_r\mathbf{h}_{i-1}^{hid}\right),\\ \mathbf{z}_i &= \sigma\left(\mathbf{W}_z\mathbf{h}_{v_i} + \mathbf{U}_z\mathbf{h}_{i-1}^{hid}\right),\\ \widetilde{\mathbf{h}}_i &= \tanh\left(\mathbf{W}\mathbf{h}_{v_i} + \mathbf{U}\left(\mathbf{r}_i\odot\mathbf{h}_{i-1}^{hid}\right)\right),\\ \mathbf{h}_i^{hid} &= \left(1-\mathbf{z}_i\right)\odot\mathbf{h}_{i-1}^{hid} + \mathbf{z}_i\odot\widetilde{\mathbf{h}}_i,\end{aligned} \tag{6}$$

where $\mathbf{W}_r, \mathbf{W}_z, \mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{U}_r, \mathbf{U}_z, \mathbf{U} \in \mathbb{R}^{d\times d}$ are transformation weight matrices. We calculate the affinity coefficient factor between candidate items and the last one:

$$\alpha_{il} = att\left(\mathbf{h}_i^{hid}, \mathbf{h}_l^{hid}\right) = \mathbf{v}^{\top}\sigma\left(\mathbf{A}_1\mathbf{h}_i^{hid} + \mathbf{A}_2\mathbf{h}_l^{hid}\right), \tag{7}$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{d\times d}$ and $\mathbf{v} \in \mathbb{R}^d$ are learnable weights. We integrate $\mathbf{h}_i^{hid}$ attentively to acquire the sequential session embedding and obtain the prediction. The single sequence channel loss $\mathcal{L}_s$ is obtained analogously following Eq. (5).

$$\mathbf{s}_{seq} = \sum_{i=1}^{l}\alpha_{il}\mathbf{h}_i^{hid}, \quad \hat{\mathbf{y}}_i = softmax\left(\mathbf{s}_{seq}^{\top}\mathbf{h}_{v_i}\right). \tag{8}$$

### 2.3. Inter-Channel Integration Module

In order to absorb distinct pattern knowledge from different aspects of views, we resort to Self-Supervised Learning (SSL) and deploy a contrastive integration module to bridge independent channels. In detail, we apply InfoNCE [5] to minimize the distance between distinct channel representations of the same session and maximize the distance of different sessions, simultaneously:

$$\mathcal{L}_c = -\sum_{s\in\mathcal{B}}\log\left(\frac{\exp\left(\mathbf{s}_{graph}^{\top}\mathbf{W}_c\mathbf{s}_{seq}\right)}{\sum_{s'\in\mathcal{B}}\exp\left(\mathbf{s}_{graph}^{\top}\mathbf{W}_c\mathbf{s}'_{graph}\right)}\right), \tag{9}$$

where we define the discriminator with a bi-linear similarity function $f\left(\cdot,\cdot\right): \mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}$ with a trainable weight matrix $\mathbf{W}_c \in \mathbb{R}^{d\times d}$ capturing the correlations between channel embeddings, and $\mathcal{B}$ denotes the mini-batch data.

### 2.4. Model Training

We employ the primary & auxiliary loss architecture to combine the main prediction objective and contrastive auxiliary loss term, and utilize the decay factor $\beta$ to balance the weight: $\mathcal{L} = \mathcal{L}_g + \beta\mathcal{L}_c$. We utilize the pre-training mechanism to maintain the independent characteristic of each channel. Concretely, we train each channel independently with the corresponding loss ($\mathcal{L}_g$ or $\mathcal{L}_s$), ignoring the counterpart channel. We dump the pre-trained representations, and load them in the integrated DC-Rec model to further optimize the performance.

## 3. EXPERIMENTS

**Table 1**: Overall Performance Comparison.

| Dataset | YooChoose | | | | | | NowPlaying | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | MRR@K | | | NDCG@K | | | MRR@K | | | NDCG@K | | |
| K | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| GRU4REC | 24.03 | 25.64 | 26.36 | 27.87 | 31.77 | 34.36 | 4.79 | 4.93 | 5.01 | 5.18 | 5.51 | 5.78 |
| GRU4REC+ | 23.85 | 25.72 | 26.55 | 28.45 | 32.97 | 35.94 | 6.14 | 6.59 | 6.90 | 7.01 | 8.12 | 9.26 |
| NARM | 24.46 | 26.35 | 27.16 | 29.06 | 33.61 | 36.57 | 6.12 | 6.61 | 6.93 | 7.03 | 8.22 | 9.41 |
| STAMP | 25.44 | 27.20 | 27.97 | 29.99 | 34.26 | 37.03 | 7.06 | 7.44 | 7.68 | 7.91 | 8.83 | 9.70 |
| FGNN | 26.77 | 28.53 | 29.28 | 31.31 | 35.55 | 38.27 | 5.95 | 6.37 | 6.66 | 6.73 | 7.80 | 8.91 |
| HIDE | 27.42 | 29.25 | 30.02 | 32.19 | 36.62 | 39.39 | 6.39 | 7.16 | 7.58 | 7.62 | 9.48 | 11.03 |
| SR-GNN | 27.96 | 29.69 | 30.42 | 32.69 | 36.84 | 39.49 | 6.96 | 7.44 | 7.76 | 7.85 | 9.06 | 10.27 |
| CORE | 23.77 | 25.77 | 26.62 | 28.60 | 33.43 | 36.50 | 6.18 | 6.97 | 7.41 | 7.46 | 9.37 | 10.99 |
| DC-Rec | **28.66** | **30.46** | **31.22** | **33.35** | **37.70** | **40.43** | **7.28** | **7.97** | **8.39** | **8.58** | **10.27** | **11.79** |
| Impv%[1] | 2.49 | 2.60 | 2.62 | 2.03 | 2.33 | 2.39 | 3.11 | 7.15 | 8.19 | 8.47 | 8.29 | 6.91 |

[1] The improvement compares DC-Rec (marked with boldface) with the best competitor (underlined)

### 3.1. Experimental Settings

**Dataset.** To evaluate the performance of DC-Rec model, we conduct experiments on two real-world datasets, i.e., YooChoose1/64 and NowPlaying. We conduct the same dataset processing procedure on these datasets following [3, 6]. The dataset statistics after processing is shown in Table 3 in the Appendix.

**Metric.** We adopt the widely-used evaluation protocols $MRR@K$ and $NDCG@K$ as the metrics. We set $K =$

$5, 10, 20$ to test the model performance under different evaluation lengths.

**Compared Methods.** We compare the experimental results with state-of-the-art SBR models. These SOTAs are generally divided into two groups—RNN-based (including GRU4REC [7], GRU4REC+ [8], NARM [1], and STAMP [9]) and GNN-based methods (including SR-GNN [3], FGNN [10], and HIDE [11]). We also include a model-agnostic method (CORE [12]) to make a comprehensive comparison.

**Parameter Settings.** The parameter settings are in §B in the Appendix.

### 3.2. Performance Comparison

Table 1 shows the overall performance comparison with baseline models. We can see DC-Rec exhibits a consistent outperformance across all cases. In specific, DC-Rec achieves a mean improvement of $2.57\%$ and $2.25\%$ in terms of MRR@K and NDCG@K on YooChoose, respectively; $6.15\%$ and $7.89\%$ in terms of MRR@K and NDCG@K on NowPlaying, respectively. This empirically demonstrates the consistent superiority of our DC-Rec model and the effectiveness of the dual-channel contrastive proposal for SBRs. DC-Rec consistently outperforms CORE by a large gap, exhibiting the robustness of our model. Besides, the performance of DC-Rec exhibits a continuous enhancement as the value of $K$ increases, demonstrating the rising optimization for a longer target sequence.

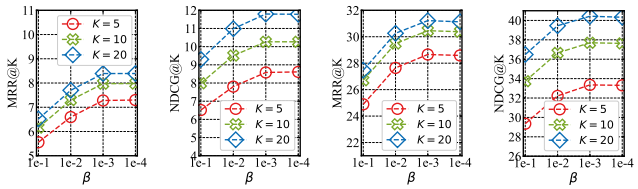
### 3.3. Ablation Experiments

**Table 2**: The Performance of Ablation Experiments.

| Dataset | Model | M@5 | M@10 | M@20 | N@5 | N@10 | N@20 |
|---|---|---|---|---|---|---|---|
| YooChoose | w/o seq | 25.30 | 27.00 | 27.76 | 29.54 | 33.63 | 36.38 |
| | w/o graph | 26.65 | 28.43 | 29.18 | 31.10 | 35.38 | 38.10 |
| | w/o ssl | 27.56 | 29.32 | 30.04 | 32.12 | 36.35 | 38.93 |
| | DC-Rec | 28.66 | 30.46 | 31.22 | 33.35 | 37.70 | 40.43 |
| NowPlaying | w/o graph | 6.22 | 6.63 | 6.90 | 7.04 | 8.05 | 9.04 |
| | w/o ssl | 6.83 | 7.21 | 7.47 | 7.59 | 8.54 | 9.47 |
| | w/o seq | 7.20 | 7.61 | 7.88 | 8.01 | 9.00 | 9.99 |
| | DC-Rec | 7.28 | 7.97 | 8.39 | 8.58 | 10.27 | 11.79 |

We conduct ablation experiments on key components of DC-Rec model to verify the effectiveness of different channels and self-supervised integration. DC-Rec w/o graph and DC-Rec w/o seq are trained with the single corresponding channel and DC-Rec w/o ssl replace SSL integration with SUM pooling operation.

As can be seen in Table 2, DC-Rec demonstrates a consistent superiority over other ablation variants. Moreover, on YooChoose, the sequence channel plays a more vital role. Besides, integrating the graph channel also significantly improves the model performance. Replacing contrastive channel integration operation with SUM pooling leads to a degradation, which verifies the advancement of our Inter-Channel Integration Module design. The performance on NowPlaying reveals the prioritization as: graph channel $>$ contrastive integration $>$ sequence channel. We argue that this might be because with more items in NowPlaying, sessions in NowPlaying possess more structural and inter-session information, thus graph channel and integration module embody more important roles.




(a) $Recall@K$ on NowPlaying (b) $NDCG@K$ on NowPlaying (c) $Recall@K$ on YooChoose (d) $NDCG@K$ on YooChoose

**Fig. 2**: Effect of SSL Coefficient $\beta$.

### 3.4. Parameter Analysis

We accomplish DC-Rec SSL architecture following the *primary & auxiliary* framework, which utilizes the SSL coefficient $\beta$ to control the ratio of SSL loss term. In order to investigate the influence of SSL coefficient $\beta$ on DC-Rec's performance, we conduct experiments w.r.t. a set of $\beta$ values $\{1e-1, 1e-2, 1e-3, 1e-4\}$. Fig. 2 demonstrates the impact of different $\beta$ values. DC-Rec realizes a performance improvement with the increase of $\beta$ while $\beta > 1e-3$, and tends to degrade after $\beta < 1e-3$. Thus, setting $\beta = 1e-3$ leads to a consistent optimal performance.

Analysis experiments targeting other key hyper-parameters including embedding dimension $d$ and GNN depth $L$ are also conducted. Due to space limitation, the experimental results and analyses are left to §C in the Appendix.

## 4. CONCLUSION AND FUTURE WORK

Revealing the intricate patterns of item distribution beneath a chronological interaction sequence is at the core of anonymous session-based recommendation. Current literature models the interaction dependency from the aspect of either sequential or graphical scheme. However, the two distribution patterns impede each other for comprehensive modeling. Therefore, in this paper, we propose a Self-Supervised Learning-based integration framework—**DC-Rec**, for **D**ual-**C**hannel **Rec**ommendation. In DC-Rec, we capture local and global structure patterns via the graph channel, and attentive sequential patterns via the sequence channel. We deploy a contrastive inter-channel integration module to enhance the semantic content of the individual channels. Moreover, we utilize the pre-trained single channel representation to reserve the original knowledge from each channel. Extensive experiments conducted on two real-world datasets demonstrate the consistent superiority of our DC-Rec framework.

## 5. REFERENCES


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

[2] Hidasi Balázs, Quadrana Massimo, Karatzoglou Alexandros, and Tikk Domonkos, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. 2016, pp. 241–248, ACM.

[3] Wu Shu, Tang Yuyuan, Zhu Yanqiao, Wang Liang, Xie Xing, and Tan Tieniu, "Session-based recommendation with graph neural networks," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 346–353.

[4] Li Yujia, Zemel Richard, Brockschmidt Marc, and Tarlow Daniel, "Gated graph sequence neural networks," in *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[5] Aäron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[6] Wang Ziyang, Wei Wei, Cong Gao, Li Xiao-Li, Mao Xian-Ling, and Qiu Minghui, "Global context enhanced graph neural networks for session-based recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 169–178.

[7] Hidasi Balázs, Karatzoglou Alexandros, Baltrunas Linas, and Tikk Domonkos, "Session-based recommendations with recurrent neural networks," in *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[8] Yong Kiam Tan, Xinxing Xu, and Yong Liu, "Improved recurrent neural networks for session-based recommendations," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 2016, pp. 17–22, ACM.

[9] Liu Qiao, Zeng Yifu, Mokhosi Refuoe, and Zhang Haibin, "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*. 2018, pp. 1831–1839, ACM.

[10] Qiu Ruihong, Li Jingjing, Huang Zi, and Yin Hongzhi, "Rethinking the item order in session-based recommendation with graph neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 579–588.

[11] Yinfeng Li, Chen Gao, Hengliang Luo, Depeng Jin, and Yong Li, "Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1997–2002, ACM.

[12] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao, "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*. 2022, pp. 1796–1801, ACM.

[13] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al., "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 4653–4664, ACM.

## A. DATASET PRE-PROCESSING AND DESCRIPTION

To evaluate the performance of DC-Rec model, we conduct experiments on two real-world datasets, i.e., YooChoose1/64[2] and NowPlaying[3]. We conduct the same dataset processing procedure on these datasets following [3, 6]. Specifically, we filter out sessions with length of 1 and items that appear less than 5 times over all sessions. Moreover, we divide the dataset according to the chronological order—we set sessions of the last week/day as the test set, and set the remaining data as the training set. We serially generate item sequence data and corresponding target item labels via a session splitting data augmentation process. For the input session sequence $s = v_1^s, v_2^s, \cdots, v_l^s$, we generate $l-1$ (data, label) tuplets, i.e., $([v_1^s], v_2^s), ([v_1^s, v_2^s], v_3^s), \cdots, ([v_1^s, v_2^s, \cdots, v_{l-1}^s], v_l^s)$. The statistics of datasets after processing is shown in Table 3.

**Table 3**: Dataset Statistics.

| Dataset | YooChoose | NowPlaying |
|---|---|---|
| # of training sessions | $369,859$ | $825,304$ |
| # of test sessions | $55,898$ | $89,824$ |
| # of items | $16,766$ | $60,417$ |
| # of clicks | $557,248$ | $1,367,963$ |
| Average length | $6.16$ | $7.42$ |

## B. PARAMETER SETTINGS

We implement our model by PyTorch and release our codes publicly. We search the optimal embedding dimension $d$ in the range of $\{32, 64, 128, 256\}$, and in most cases setting $d = 128$ leads to the best performance. The whole model training can be divided into pre-training process and fine-tuning process. In the pre-training process, we separately train the single channel model and initialize the corresponding embedding vectors with normal distribution with mean of 0 and standard deviation of $0.1$. In the fine-turning process, we initialize embedding vectors in DC-Rec with the dumped embedding values. We set the learning rate as $1e-3$ and batch size as 300. We use the Adam as the optimizer. For graph pre-processing, we set the value of $N$ for Top-$N$ sampling in global graph construction as 12. We search the optimal layer depth $L$ in the **GVRL** module in the range of $\{1, \cdots, 4\}$. We set $Recall@20$ as the evaluation metric on the validation set, and early-stop the training process for 3 epochs if the evaluation metric stops improving.

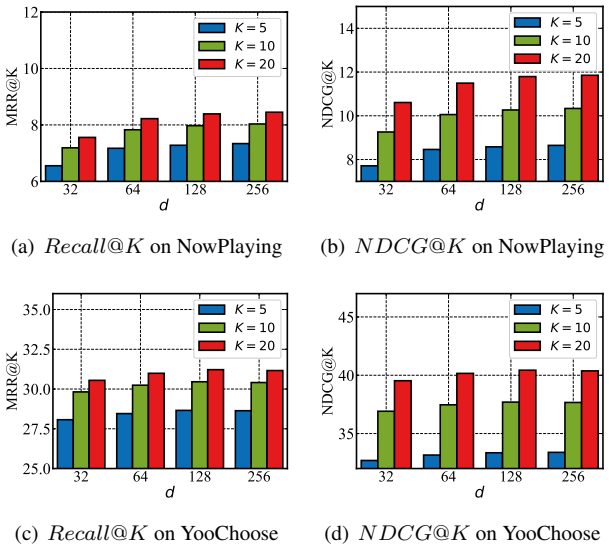
We implement and fine-tune baseline models based on either their official codes or RecBole framework [13]. To make a fair comparison, we set the embedding dimensions of all baseline models as 128. Furthermore, to accelerate the convergence and ensure the baseline performance while making a fair comparison, we set the learning rate as $1e-3$ and set the value of patience steps in early-stop as 5. We fine-tune all baseline models, and search for the optimal parameter assignment. Specifically, regarding SR-GNN, we opt for BPR loss as the loss term, employ a one-layer GRU module, and assign the dropout ratio of the model as $0.5$. For NARM, we set the batch size as 300 for fine-tuning. For other hyper-parameters of all baselines, we adhere to their optimized setting values in their papers.

[2]http://2015.recsyschallenge.com/challege.html
[3]http://dbis-nowplaying.uibk.ac.at/#nowplaying

## C. MORE DETAILED PARAMETER ANALYSES

We conduct analysis experiments on the key hyper-parameters of DC-Rec. Besides SSL loss ratio $\beta$, we also analyze hyper-parameters including embedding dimension $d$ and graph neural network layers $L$.

### C.1. Impact of Embedding Dimension $d$




(a) $Recall@K$ on NowPlaying (b) $NDCG@K$ on NowPlaying

(c) $Recall@K$ on YooChoose (d) $NDCG@K$ on YooChoose

**Fig. 3**: Effect of Dimension $d$

We uniformly set the dimension of embeddings in DC-Rec as $d$ to reduce computation complexity. To investigate the influence of different embedding size, we search for the optimal value of embedding dimension $d$ between $32, 64, 128, 256$. Fig. 3 exhibits the performance comparison of different embedding dimension $d$. We have the following observations: 1) DC-Rec performs better with the increase of embedding dimension $d$. 2) DC-Rec tends to perform steady after $d > 128$. 3) We set $d = 128$ uniformly to balance the model performance and storage capacity.

### C.2. Impact of Layer $L$

**Table 4**: Impact of GNN Layer Depth $L$

| Dataset | L | M@5 | M@10 | M@20 | N@5 | N@10 | N@20 |
|---|---|---|---|---|---|---|---|
| YooChoose | L=1 | 28.63 | 30.41 | 31.17 | 33.35 | 37.67 | 40.37 |
| | L=2 | 28.66 | 30.46 | 31.22 | 33.35 | 37.70 | 40.43 |
| | L=3 | 28.62 | 30.42 | 31.18 | 33.31 | 37.66 | 40.39 |
| NowPlaying | L=1 | 7.31 | 8.00 | 8.41 | 8.61 | 10.28 | 11.80 |
| | L=2 | 7.28 | 7.97 | 8.39 | 8.58 | 10.27 | 11.79 |
| | L=3 | 7.26 | 7.97 | 8.39 | 8.53 | 10.26 | 11.79 |

We also conduct comparative experiments on DC-Rec with different convolution layers. Graph channel performance can benefit from stacking multiple convolution layers since capturing wider scopes and model node embedding propagation and aggregation through multi-hops provides the model more powerful expressiveness.

Towards better graph channel performance, we denote DC-Rec with different graph convolution depth $L$ as DC-Rec-$L$ and conduct experiments with representative layer depths, i.e., $L = \{1, 2, 3\}$. Table 4 exhibits the performance of different layer depths (we abbreviate $MRR@K$ as $M@K$ and $NDCG@K$ as $N@K$ for simplicity). As can be seen in the table, on YooChoose, $L = 2$ leads to a consistent outperformance compared with other variants, and on NowPlaying, $L = 1$ leads to the best performance. Jointly analyzing Table 4 with Table 1, we can see that DC-Rec-$L$ consistently outperforms the state-of-the-art methods under every circumstances.