# HybridGNN-SR: Combining Unsupervised and Supervised Graph Learning for Session-based Recommendation

Kai Deng
*Guizhou University*
Guiyang, China
kai_deng@126.com

Jiajin Huang
*Beijing University of Technology*
Beijing, China
jhuang@bjut.edu.cn

Jin Qin
*Guizhou University*
Guiyang, China
qin_gs@163.com

2020 International Conference on Data Mining Workshops (ICDMW) | 978-1-7281-9012-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDMW51313.2020.00028

***Abstract*—Session-based recommendation aims to predict the next item that a user may visit in the current session. By constructing a session graph, Graph Neural Networks (GNNs) are employed to capture the connectivity among items in the session graph for recommendation. The existing session-based recommendation methods with GNNs usually formulate the recommendation problem as the classification problem, and then use a specific uniform loss to learn session graph representations. Such supervised learning methods only consider the classification loss, which is insufficient to capture the node features from graph structured data. As unsupervised graph learning methods emphasize the graph structure, this paper proposes the HybridGNN-SR model to combine the unsupervised and supervised graph learning to represent the item transition pattern in a session from the view of graph. Specifically, in the part of unsupervised learning, we propose to combine Variational Graph Auto-Encoder (VGAE) with Mutual Information to represent nodes in a session graph; in the part of supervised learning, we employ a routing algorithm to extract higher conceptual features of a session for recommendation, which takes dependencies among items in the session into consideration. Through extensive experiments on three public datasets, we demonstrate that HybridGNN-SR outperforms a number of state-of-the-art methods on session-based recommendation by integrating the strengths of the unsupervised and supervised graph learning methods.**

***Index Terms*—Session-based Recommendation, Capsule Network, Variable Graph Auto-Encoder**

## I. Introduction

A session is an ordered user historical behaviors in a certain period of time. The ordered behaviors may consist of visited items (e.g movies, POIs, Web pages, etc.). Session-based recommendation aims to predict the next item that the user may visit in the current session. Session-based recommendation considers a user preference evolution and can better meet the needs in the practical scenario [26], [27]. Recently, a directed graph structure was used to describe the ordered user historical behaviors in a session. Specifically, in each session graph, the node is an item and the edge between two nodes means a user visits one after other in a session. And then the Session-based Recommendation with Graph Neural Networks (SR-GNN) [31] was proposed to capture the information over adjacent items in the session by using Graph Neural Networks (GNNs) [2], [12].

There are two key problems in session-based recommendation with GNNs. One is how to integrate graph structured data into the recommendation methods and then use GNNs to obtain the representations of all nodes involved in each graph. [31] and [32] used the Gated Graph Sequence Neural Network [10] on each session graph to extract the latent vectors of each node, while [15] used a weighted graph attentional layer on each session graph. The other is how to represent each session by the obtained representations of nodes. [15] and [31] used an attention mechanism, and [32] used a self-attention mechanism to aggregate the item latent vectors together as the session representation.

However, majority of the existing session-based recommendation methods with GNNs only use the information of liking or not liking an item to define the loss functions. Optimizing these loss functions results in various supervised learning methods to represent nodes in each session graph. These supervised methods enforce the consistency of user preference on items, but ignore how to further extract information from the graph structure. Currently, the methods of extracting information from the graph structure are usually based on unsupervised learning methods, and can be divided two categories [22]. One tends to reconstruct adjacency information of the graph, the other tends to use a random walk method on the graph. The loss function of the former is derived from the similarity of corresponding nodes in both the real graph and the representation space, and the loss function of the latter is derived from the proximity information on the graph. Obviously, the loss functions of the unsupervised GNN methods emphasize the graph structure information, and ignore user preferences, which can not be directly used for recommendation. In summary, we argue that a method which only uses a uniform type of loss will limit its ability of integrating the complementary strengths of the supervised and unsupervised graph learning method, which inspires us to integrate the unsupervised learning method into the supervised learning method to generate better results of session-based recommendation with GNNs. For the purpose, we employ a

2375-9259/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDMW51313.2020.00028

Variational Graph Auto-Encoder (VGAE) [4] as a regularizer to reconstruct adjacency information of the graph. Different from the original VGAE, on the one hand, our model adds a probability to balance two operations, namely preserving each node's information about itself and aggregating the information of its' neighborhood nodes; on the other hand, our model adds a negative sampling procedure to make the representations of nodes in a graph more similar to the graph than other graphs.

Furthermore, the existing supervised learning methods for session-based recommendation with GNNs mainly integrate item vectors to represent the session. However, items in a session belong to different categories. For example, if items in a session include 'iphone 11', 'illy coffee bean', 'Down coat', and 'boots', we can see that the session tends to belong to the 'clothes' category. We should recommend an item belonging to 'clothes'. This inspires us to represent a session from the view of categories to get a higher conceptual representation of a session. A dynamic routing process [19] has been used to analyze the user's multiple interests for recommendation [7]. However, [7] only takes each items as a single object, which results in a limited capacity of learning a higher conceptual representation of a session from information over adjacent items. In this paper, we mine items' adjacent information by GNNs and then employ the dynamic routing process to group items of a session into several clusters. At last, the attention mechanism is used to aggregate the cluster features together to represent the session.

The main contributions of this paper are summarized as:

- To leverage graph data for the session-based recommendation tasks, we propose the HybridGNN-SR model to jointly optimize two loss functions derived from the GNN methods: the supervised loss over data relevant to the recommendation task and the unsupervised loss only over graph structured data. The combined loss function integrates the strengths of the supervised and unsupervised learning method to enforce both the user preference on items and graph data structure consistency, which leads to the improvement of recommendation performance.
- In the part of unsupervised method over graph structured data, HybridGNN-SR uses multi-kinds of strategies of preserving graph structure to assist the session-based recommendation task. In the part of supervised method for the recommendation task, HybridGNN-SR represents a session from the higher conceptual level derived from a dynamic routing process. In one word, HybridGNN-SR integrates the complementary strengths of GNN and the dynamic routing process over adjacent items in a session.
- We perform experiments on three real-world datasets to demonstrate the effectiveness of HybridGNN-SR and comprehensively analyze the key components of HybridGNN-SR.

## II. Related Work

### A. Session-based Recommendation

In session-based recommendation, sequential data are obtained by tracking users' records over a sequence of time. To handle such sequential data, the earlier method mainly mined item co-occurrence patterns to obtain an item-to-item similarity matrix. Later, FPMC [18] used Markov Chain (MC) to model the sequence relationship for next-item recommendation. STAMP [11] employed a multi-layer neural network with the attentive mechanism to capture users' general and short interests for session-based recommendation.

Recent studies showed that Recurrent Neural Network (RNN) can be used to model such sequential data for session-based recommendation. For instance, GRU4REC [3] used the Gated Recurrent Unit (GRU) as a special form of RNN for session-based recommendation, NARM [9] added the attentive mechanism into RNN for session-based recommendation, SR-GNN [31] integrated the Graph Neural Network into RNN to improve the recommendation quality, and FGNN [15] and GC-SAN [32] introduced the self-attention mechanism into Gated Graph Neural Network to obtain the state-of-the-art recommendation performance. In addition, a user's multi purpose [28] and intention [29] can provide rich information for session-based recommendation.

### B. Graph Neural Network for Recommendation

The Graph Neural Networks (GNNs) aim to learn a node's representation by repeating the procedure of aggregating features of the node' neighbor nodes. In recent years, some typical GNNs have been proposed, such as Graph Convolutional Network (GCN) [5], Graph Attention Networks (GAN) [21], Gated Graph Sequence Neural Network (GG-NN) [10], and so on. These GNN models have also be extended for recommendation [13], [23]. Furthermore, some methods integrate auxiliary information into the GNN models for recommendation, for example, item-item relation [33], Knowledge Graph [16], [24], [25], and social relation [20], [30]. As mentioned before, SR-GNN [31], GC-SAN [32] and FGNN [15] extended GG-NN for session-based recommendation. Different from the existing purely supervised session-based recommendation methods with GNNs, we combine unsupervised and supervised methods to make full use of graph structure for session-based recommendation.

### C. Routing Mechanism for Recommendation

The dynamic routing mechanism is used to model the connections between the relations of features in the form of a vector named a capsule [19]. Based on the routing mechanism, [7] proposed the Multi-Interest Network with Dynamic routing (MIND) to learn user diverse representations for recommendation; [8] employed the capsule network to mine the sentiment for review-based recommendation. Specially, [7] improved the recommendation quality by using the routing mechanism to cluster a user's historical behaviors into several embedding vectors to represent the user, instead of a single embedding vector. However, [7] only took each item as a single object

and did not consider the adjacent items. Our model adds the session graph structure to extract dependent item correlation information and then feeds them into the routing mechanism to mine the user's diversity of interests from a session.

## III. The Proposed Model

### A. Preliminary

The session-based recommendation problem can be formalized as follows. Let $V = \{v_1, v_2, ..., v_m\}$ denote the set of all unique items collected in all the sessions. All $d$-dimensional item latent features are summarized in matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$. A session sequence $s$ can be represented by an ordered list $s = \{v_{s,1}, v_{s,2}, ..., v_{s,|s|}\}$ according to timestamps, where $v_{s,i} \in V$ represents a clicked item of the user in the session $s$ and $|s|$ is the size of session $s$. The goal of the session-based recommendation is to predict the $v_{s,|s|+1}$ which may be visited in the session $s$. A session $s$ can be modeled as a graph $G_s = (V_s, E_s)$ where $V_s \subseteq V$ is a set of nodes which consisting of items in session $s$ and $E_s$ is a set of edges consisting of a set of item pair $(v_{s,i}, v_{s,i+1})$ which means a user visits item $v_{s,i+1}$ after $v_{s,i}$ in session $s$ [31].

Starting from the above definition, the framework of our model HybridGNN-SR is presented in Figure 1. There are two key parts: one is the GNN part to extract the item latent information, the other is the routing part to represent a session. The subsequent sections will give more details.

### B. Preserving Graph Structure

Let us consider two adjacency matrices $\hat{\mathbf{A}}_s^{in} \in \mathbb{R}^{|s| \times |s|}$ and $\hat{\mathbf{A}}_s^{out} \in \mathbb{R}^{|s| \times |s|}$ corresponding to incoming and outgoing edges in graph $G_s$. Following SR-GNN [31], each edge in $\hat{\mathbf{A}}_s^{in}$ is divided by the indegree of the ending node of the edge; each edge in $\hat{\mathbf{A}}_s^{out}$ is divided by the outdegree of the starting node of the edge. Finally, we get two matrices $\mathbf{A}_s^{in} \in \mathbb{R}^{|s| \times |s|}$ and $\mathbf{A}_s^{out} \in \mathbb{R}^{|s| \times |s|}$.

By introducing a stochastic latent variable matrix $\mathbf{Z}_s \in \mathbb{R}^{|s| \times d}$ and a sub-matrix of $\mathbf{X}_s \in \mathbb{R}^{|s| \times d}$ which corresponds items in session $s$, we choose Variational Graph Autoencoder (VGAE) [4] to extract the latent feature representation to avoid the influence of noisy data. For the adjacency matrices $\mathbf{A}_s^{in}$, we have

$$q(\mathbf{Z}_s^{in} | \mathbf{X}_s, \mathbf{A}_s^{in}) = \prod_{i=1}^{|s|} q(\mathbf{Z}_{s,i:}^{in} | \mathbf{X}_s, \mathbf{A}_s^{in}) \tag{1}$$

where $q(\mathbf{Z}_{s,i:}^{in} | \mathbf{X}_s, \mathbf{A}_s^{in}) \sim N(\mathbf{Z}_{s,i:}^{in} | \boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2)$ and $\mathbf{Z}_{s,i:}^{in}$ is the sampled vectors for the $i$-th item in session $s$ which is equal to the $i$-th row of $\mathbf{Z}_s^{in}$. In the normalization distribution $N(\mathbf{Z}_{s,i:}^{in} | \boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2)$, the mean $\boldsymbol{\mu}_{s,i}$ and the logarithmic variance $\ln \boldsymbol{\sigma}_{s,i}^2$ can be obtained by a GNN which takes adjacency matrices $\mathbf{A}_s^{in}$ and the item embeddings $\mathbf{X}_s$ as input. For simplification, let $\mathbf{A}_{s,i:}^{in}$ be the $i$-th row of $\mathbf{A}_s^{in}$, we can use the GNN with one layer as $\boldsymbol{\mu}_{s,i} = \mathbf{A}_{s,i:}^{in} \mathbf{X}_s \mathbf{W}_{\mu}^{in}$ and $\ln \boldsymbol{\sigma}_{s,i}^2 = \mathbf{A}_{s,i:}^{in} \mathbf{X}_s \mathbf{W}_{\sigma}^{in}$ where $\mathbf{W}_{\mu}^{in} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_{\sigma}^{in} \in \mathbb{R}^{d \times d}$ are learnable neural parameters. When $\boldsymbol{\mu}_{s,i}$ and $\boldsymbol{\sigma}_{s,i}^2$ are available, we can follow the re-parameterization trick to transform $\mathbf{Z}_{s,i:}^{in} \sim N(\mathbf{Z}_{s,i:}^{in} | \boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2)$ into $\mathbf{Z}_{s,i:}^{in} = \boldsymbol{\mu}_{s,i} + \boldsymbol{\sigma}_{s,i} \odot \boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{1})$.

Furthermore, different from the original VGAE, we consider the mean $\boldsymbol{\mu}_{s,i}$ can also be used to represent the features of node $v_i$ passed to the GNN, and add a trade-off parameter $\alpha \in [0, 1]$ to balance the contributions of features mined by the GNN and the original features. So the final $\boldsymbol{\mu}_{s,i}$ can be modified by

$$\boldsymbol{\mu}_{s,i} = (1 - \alpha) \mathbf{A}_{s,i:}^{in} \mathbf{X}_s \mathbf{W}_{\mu}^{in} + \alpha \mathbf{X}_{s,i:}. \tag{2}$$

where $\mathbf{X}_{s,i:}$ is the $i$-th row of $\mathbf{X}_s$. $\alpha$ in Equ. (2) could be explained as a probability of being back to the original features of the node [6]. Finally, the loss function can be

$$L_{VGAE}^{in} = \sum_{s, v_{s,i} \in s} ||f(\mathbf{Z}_{s,i:}^{in} (\mathbf{Z}_s^{in})^T) - \hat{\mathbf{A}}_{s,i:}^{in}||_2^2 + KL(\boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2) \tag{3}$$

where $f$ is a sigmoid function and the first part of $L_{VGAE}^{in}$ is the reconstruction loss of a session graph. The second part $KL(\boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2)$ is Kullback-Leibler divergence between $N(\boldsymbol{\mu}_{s,i}, \boldsymbol{\sigma}_{s,i}^2)$ and $N(\mathbf{0}, \mathbf{1})$ to balance the two normalization distributions.

For the adjacency matrices $\mathbf{A}_s^{out}$, we also have

$$q(\mathbf{Z}_s^{out} | \mathbf{X}_s, \mathbf{A}_s^{out}) = \prod_{i=1}^{|s|} q(\mathbf{Z}_{s,i:}^{out} | \mathbf{X}_s, \mathbf{A}_s^{out}) \tag{4}$$

where $q(\mathbf{Z}_{s,i:}^{out} | \mathbf{X}_s, \mathbf{A}_s^{out}) \sim N(\mathbf{Z}_{s,i:}^{out} | \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\sigma}}_{s,i}^2)$. Similarly, by introducing learnable neural parameters $\mathbf{W}_{\mu}^{out} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_{\sigma}^{out} \in \mathbb{R}^{d \times d}$, we have the $\hat{\boldsymbol{\mu}}_{s,i} = (1 - \alpha) \mathbf{A}_{s,i:}^{out} \mathbf{X}_s \mathbf{W}_{\mu}^{out} + \alpha \mathbf{X}_{s,i:}$ and $\ln \hat{\boldsymbol{\sigma}}_{s,i}^2 = \mathbf{A}_{s,i:}^{out} \mathbf{X}_s \mathbf{W}_{\sigma}^{out}$. So we have the loss function

$$L_{VGAE}^{out} = \sum_{s, v_{s,i} \in s} ||f(\mathbf{Z}_{s,i:}^{out} (\mathbf{Z}_s^{out})^T) - \hat{\mathbf{A}}_{s,i:}^{out}||_2^2 + KL(\hat{\boldsymbol{\mu}}_{s,i}, \hat{\boldsymbol{\sigma}}_{s,i}^2) \tag{5}$$

We can see the total loss can be obtained by

$$L_{VGAE} = L_{VGAE}^{in} + L_{VGAE}^{out} \tag{6}$$

Furthermore, the simple sum operation over all node features can be used to represent the graph from the structural view as

$$R(\mathbf{Z}_s^{in}) = \sum_{i=1}^{|s|} \mathbf{Z}_{s,i:}^{in}$$
$$R(\mathbf{Z}_s^{out}) = \sum_{i=1}^{|s|} \mathbf{Z}_{s,i:}^{out} \tag{7}$$

To properly encode structural similarities of different nodes in the graph, we sample a negative graph over the original session graph by randomly selecting some nodes to put them on the original graph topology. The negative graph passes the VGAE encoder to obtain the representations of nodes on the negative graph, namely $\tilde{\mathbf{Z}}_s^{in}$ and $\tilde{\mathbf{Z}}_s^{out}$. Obviously, each row of $\mathbf{Z}_s^{in}$ should be closer to $R(\mathbf{Z}_s^{in})$ than $\tilde{\mathbf{Z}}_s^{in}$. Maximizing the mutual information (i.e. minimizing the negative mutual information) between $\mathbf{Z}_{s,i:}^{in}$ and $R(\mathbf{Z}_s^{in})$ can be used to embody

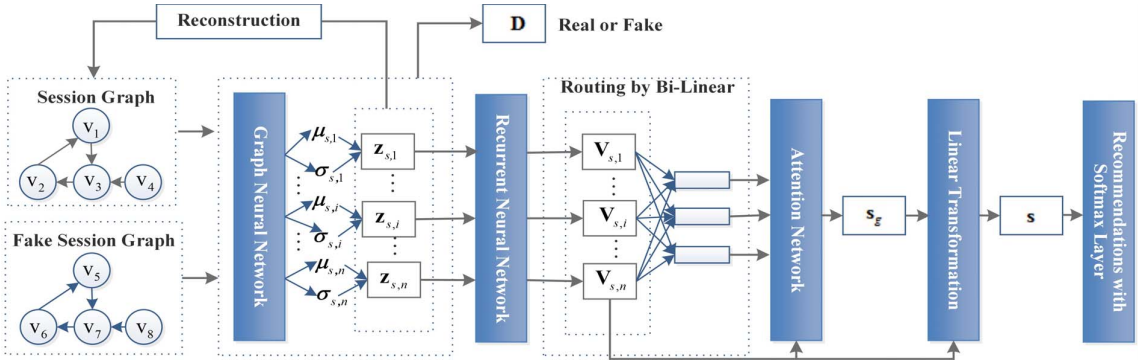


Fig. 1. An illustration of HybridGNN-SR model

such intuitive observation [22]. Let $f$ be a sigmoid function and $f(\mathbf{Z}^{in}_{s,i:}(R(\mathbf{Z}^{in}_s))^T)$ be the probability of $\mathbf{Z}^{in}_{s,i:}$ being closer to $R(\mathbf{Z}^{in}_s)$, we have

$$L^{in}_D = -\sum_{s,v_{s,i:}\in s} \log f(\mathbf{Z}^{in}_{s,i:}(R(\mathbf{Z}^{in}_s))^T) + \log(1 - f(\tilde{\mathbf{Z}}^{in}_{s,i:}(R(\mathbf{Z}^{in}_s))^T)) \tag{8}$$

It is worth noting that Equ. (8) uses randomly sampled negative samples, which is different from the loss of Generative Adversarial Networks (GAN) for VGAE [14] whose negative samples are generated by a special generator network with learnable parameters. Obviously, removing a generator reduces the number of learnable parameters in a model. [22] also adopted such random negative samples and then used a bi-linear discriminator to discriminate the positive and negative side. In our experiments, we find that the linear similarity measurement between each node and the graph can obtain a good result. Compared with the bi-linear discriminator with more learnable parameters, the linear similarity measurement in Equ. (8) does not to introduce any learnable parameters and simplifies the complexity of the model.

Similarly, each row of $\mathbf{Z}^{out}_s$ should be closer to $R(\mathbf{Z}^{out}_s)$ than $\tilde{\mathbf{Z}}^{out}_s$, so we have

$$L^{out}_D = -\sum_{s,v_{s,i:}\in s} \log f(\mathbf{Z}^{out}_{s,i:}(R(\mathbf{Z}^{out}_s))^T) + \log(1 - f(\tilde{\mathbf{Z}}^{out}_{s,i:}(R(\mathbf{Z}^{out}_s))^T)) \tag{9}$$

Combined Equ. (8) with (9), we have a loss function with sampled negative graphs as

$$L_D = L^{in}_D + L^{out}_D \tag{10}$$

In a word, in the graph-based VGAE encoder, we let the extracted item features have a chance to return original session graph, and also be closer to the original session graph but far way from the non-original session graph. The two operations can enhance the graph structure when we extract the item features from the session graph. As we do not use the label information of the graph, we can say the above operations are unsupervised methods. At the same time, as we aim to predict the next item which a user may visit, we also need to use clicked item information to supervise the learning process.

### C. Learning Preference

After the forward computation of the above GNN layer, we can get the representation of the $i$-th item in the session $s$. In order to learn the order of the item transition pattern, following [3], [10], [31], we feed items of each session graph into a Gated Recurrent Units (GRU) [1]. Passing the GRU, items in session $s$ can be represented by a matrix $\mathbf{V}_s = GRU(\mathbf{Z}_s)$. As items in each session can belong to different categories, we need to use their category information to represent the session. Let the category set be $C = \{c_1, c_2, \cdots, c_{|C|}\}$ ($|C|$ is the size of $C$) and each category $c_j$ can be represented by a vector $\mathbf{c}_{s,j} \in \mathbb{R}^{1\times d}$ for a session $s$ to be learned which is initialized randomly. In the session $s$, the connection $b_{ij}$ between the low-level item $v_{s,j}$ and high-level category $c_j$ is calculated by

$$b_{ij} = \mathbf{c}_{s,j}\mathbf{U}(\mathbf{V}_{s,i:})^T \tag{11}$$

where $\mathbf{U} \in \mathbb{R}^{d\times d}$ denotes the bilinear mapping matrix to be learned which is initialized randomly [7]. $\mathbf{V}_{s,i:}$ is the $i$-th row of $\mathbf{V}_s$ which integrates the dependent correlation information over adjacent items by the GNN model.

By introducing the routing logits $b_{ij}$ which measures the connection of items and categories in a session, we can use a routing mechanism in [19] to learn the category vector. The candidate vector for high-level category $\mathbf{h}_j$ is computed as weighted sum of all low-level items in a session.

$$\mathbf{h}_j = \sum_{i=1}^{|s|} w_{ij}\mathbf{U}(\mathbf{V}_{s,i:})^T \tag{12}$$

where $w_{ij}$ denotes the weight for connecting low-level item $v_{s,i}$ and high-level category $c_j$, which can be calculated by a softmax function relevant to $b_{:j}$ as

$$w_{ij} = \frac{exp(b_{ij})}{\sum_i exp(b_{ij})} \tag{13}$$

Finally, a non-linear squash function is applied to obtain the vector of category $c_{s,j}$ for session $s$ as

$$\mathbf{c}_{s,j} = \frac{||\mathbf{h}_j||^2}{1 + ||\mathbf{h}_j||^2}\frac{\mathbf{h}^T_j}{||\mathbf{h}_j||} \tag{14}$$

By recurrently calculating from Equ. (11) to (12), we can obtain each $\mathbf{c}_{s,j}$ for session $s$. It is worth noting that $\mathbf{c}_{s,j}$ may have different contents for different sessions because we calculate $\mathbf{c}_{s,j}$ for each session by Equ. (14).

Considering each category has different priority level for a session and the last item in a session can be considered as the current most interested item of a user, we can use the method of [7] to represent a session by a weighted summarization mechanism which takes the similarity between the last item and each $\mathbf{c}_{s,j}$ as a weight. However, in our experiments, we find that it is better to calculate the weight by defining a new general representation for each category. In detail, we define $\mathbf{W}_g \in R^{d\times|C|}$ as the general category representation. Each column of $\mathbf{W}_g$ is the general representation of a category for any sessions. Now, we can calculate the similarity between the last item and the general representation of a category to measure the contribution of the category. Equ. (15) shows a soft-attention mechanism of considering the embedding $\mathbf{s}_g$ of the session graph by aggregating the category representation in the graph.

$$\begin{aligned} a_j &= softmax(\mathbf{V}_{s,|s|:}\mathbf{W}_g) \\ \mathbf{s}_g &= \sum_{j=1}^{|C|} a_j \mathbf{c}_{s,j} \end{aligned} \tag{15}$$

where $\mathbf{V}_{s,|s|:}$ is the vector of last item in session $s$.

As for the final session representation, we take the strategy similar to [31] in which the $\mathbf{s}_g$ and the vector of last item in the session are concatenated by

$$\mathbf{s} = [\mathbf{V}_{s,|s|:}; \mathbf{s}_g]\mathbf{W} \tag{16}$$

where $\mathbf{W} \in R^{2d\times d}$.

For the session $s$, we need to output the recommendation score of the corresponding item. By ranking items in descent order based on the scores, we select the top-$K$ items for recommendation. The score of item $i$ is calculated by the inner product of the item embedding $\mathbf{X}_i$ and the session embedding $\mathbf{s}$. Furthermore, based on the scores, for each session graph, the predicted probability $\hat{y}_{s,i}$ of the item $i$ appearing to be next in session $s$ can be calculated by a softmax function over all items as shown in the following.

$$\hat{y}_{s,i} = \frac{exp(\mathbf{X}_{i:}\mathbf{s}^T)}{\sum_{v_i\in V} exp(\mathbf{X}_{i:}\mathbf{s}^T)} \tag{17}$$

Let $y_{s,i}$ be the ground truth for session $s$ on item $i$. The loss function is defined as the cross entropy of the prediction and the ground truth as

$$L_{pre} = -\sum_{s,i} y_{s,i}\log(\hat{y}_{s,i}) + (1-y_{s,i})\log(1-\hat{y}_{s,i}) \tag{18}$$

### D. Training

A straightforward way to train our model is to minimize the loss which combines the supervised loss and the unsupervised loss:

$$L = L_{pre} + \lambda_1 L_{VGAE} + \lambda_2 L_D + \lambda_3 \Omega \tag{19}$$

TABLE I
DATASET INFORMATION

| Dataset | Yoochoose1/64 | Yoochoose1/4 | Diginetica |
|---|---|---|---|
| Number of all the clicks | 557,248 | 8,326,407 | 982,961 |
| Number of train sessions | 369,859 | 5,917,745 | 719,470 |
| Number of test sessions | 55,898 | 55,898 | 60,858 |
| Number of all the items | 16,766 | 29,618 | 43,097 |
| Average length | 6.16 | 5.71 | 5.12 |

where $\lambda_1$ and $\lambda_2$ control the contribution of the corresponding $L_{VGAE}$ and $L_D$, respectively. $\Omega$ is the regularizer and $\lambda_3$ is the coefficient of the regularizer.

The whole process can be divided two tasks. They are to learn neural parameters by using unsupervised and supervised methods, respectively. We alternatively learn the two tasks to obtain the model parameters. The alternatively strategy is shown in Algorithm 1.

**Algorithm 1** Learning for HybridGNN-SR

**input**: A session set
**output**: neural parameters
**for** Each Epoch **do**
  **for** Each Iteration **do**
    1. Sample a batch of sessions of size $B$
    2. Update model parameters by optimizing $L_{pre}+\lambda_3\Omega$
  **end for**
  **for** Each Iteration **do**
    1. Sample a batch of real sessions size $B$
    2. Disorganize the order of items in each session as the fake sessions.
    3. Update model parameters by optimizing $\lambda_1 L_{VGAE} + \lambda_2 L_D + \lambda_3\Omega$
  **end for**
**end for**

## IV. EXPERIMENTS

### A. Datasets and Evaluation Metrics

**Datasets**. We validate the effectiveness of our proposed method HybridGNN-SR on three real-world datasets, i.e., Yoochoose1/64, Yoochoose1/4[1] and Diginetica[2]. The detailed information of the datasets can be found in [3], [15], [31]. Following their preprocessing process, we also filter out items appearing less than 5 times and then remove all sessions with fewer than 2 items on three datasets. We set the session data of last week as the test data, and the remaining for training. After preprocessing, the statistics of the datasets are shown in Table I.

**Evaluation Metrics**. To evaluate the recommendation performance of all models, we rank all possible items for all sessions in descent order according to scores calculated by all models, and then adopt two common metrics, Precision ($P$ for short) and Mean Reciprocal Rank ($MRR$ for short ). $P@20$ and $MRR@20$ measures predictive accuracy and the

[1]http://2015.recsyschallenge.com/challege.html
[2]http://cikm2016.cs.iupui.edu/cikm-cup

correct recommendation order in the top 20 of the ranking list, respectively. The larger are their values, the better is the recommendation performance.

### B. Baselines

In order to demonstrate the advantage of our model, we compare HybridGNN-SR with other recommendation methods. POP, S-POP and Item-KNN recommend top rank items of the most popular item, the most popular items in sessions, and items similar to items that in sessions, respectively; BPR-MF [17] uses the mean of latent vectors of items in a session for recommendation; FPMC [18] combines matrix factorization and the most recent item for recommendation. GRU4REC [3], NARM [9] and STAMP [11] are three typical session-based recommendation with RNN. SR-GNN [31] and FGNN [15] are two state-of-the-art methods of session-based recommendation with GNNs.

### C. Parameter Setup

The size of the latent feature vectors of the items are set to 100 for every layer including the initial embedding layer. We use the Adam optimizer for mini-batch optimization with the initial learning rate 0.001 and the batch size 100. All parameters of the HybridGNN-SR are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. In HybridGNN-SR, $\lambda_1$ is set to 10, $\lambda_2$ is 150, L2 penalty $\lambda_3$ is set $10^{-9}$ to avoid overfitting, and the category number is set to 7. For $\alpha$, we set 0.65 in Yoochoose1/64 , 0.85 in Yoochoose1/4 , and 0.25 in Diginetica. Section IV-D demonstrates the comparison results and section IV-E discusses the effects of hyperparameters in HybridGNN-SR.

### D. Comparison with Baselines

Table II presents the comparison results of our model HybridGNN-SR with baselines. The best results are highlighted in boldface. As in [9], [15], [31], the result of FPMC on Yoochoose1/4 is not reported because of the hardware limitation. From the table, we can see methods combining RNN with GNN, such as SR-GNN, FGNN, and our model HybridGNN-SR, outperform other methods. HybridGNN-SR performs better than SR-GNN and FGNN in terms of Precision and MRR on both Yoochoose1/64 and Diginetica. On Yoochoose1/4, HybridGNN-SR performs better than SR-GNN and FGNN in terms of Precision and weaker than SR-GNN and FGNN in terms of MRR. In one word, HybridGNN-SR beats all baselines in terms of Precision, and obtains acceptable MRR in most cases.

As SR-GNN, FGNN and HybridGNN-SR adopt the same soft-attention mechanism to represent a session, the competitive performance of HybridGNN-SR comes from the graph-preserving mechanism and the high-level representation of a session. The next session will demonstrate the effect of each component of HybridGNN-SR.

### E. Parameter Study

*1) Impact of Each Component:* We now investigate the contribution of each component to the proposed model by eliminating the contribution from HybridGNN-SR in turn:

- HybridGNN-SR\GS: Eliminating the impact of graph structure by removing the session graph. In this case, HybridGNN-SR degrades the session-based recommendation with the routing mechanism similar to [7].
- HybridGNN-SR\VGAE: Eliminating the impact of $L_{VGAE}$ by setting $\lambda_1$ = 0 in Equ. (19) and directly feeding the outputs of GNN into the GRU.
- HybridGNN-SR\CN: Eliminating the impact of session representation from the higher level by removing the routing mechanism.
- HybridGNN-SR\D: Eliminating the impact of $L_D$ by setting $\lambda_2$ = 0 in Equ. (19).

Table III shows results of HybridGNN-SR and its variants with the same parameters. In summary, the performance degrades when we remove any of components in HybridGNN-SR.

In detail, HybridGNN-SR\GS performs worst on the three datasets. This observation demonstrates the importance of the session graph by which we can make full use of information over adjacent items.

When either VGAE or the routing mechanism is eliminated, the performance also degrades. At the same time, when we combine $L_D$ with $L_{VGAE}$ to preserve the graph structure, the performance will be further improved. These observations suggest that the graph-preserving mechanism and the high-level representation of a session can help HybridGNN-SR improve the recommendation quality.

*2) Impact of hyper-parameters:* In the VGAE part of HybridGNN-SR, we set $\alpha$ to balance the contributions of features mined by the GNN and the original features. Figure 2 indicates results of different $\alpha$ by fixing $\lambda_1 = 10$ and $\lambda_2 = 150$. We observe that the optimum differs slightly for every dataset due to their different neighborhood structures. In detail, $\alpha$ tends to have a higher value to perform better on Yoochoose1/64 and Yoochose1/4, and have a smaller value to perform better on Diginetica. In summary, the results of $\alpha$ in $[0.25, 0.85]$ are better than either $\alpha = 0$ or $\alpha = 1$, which proves the importance of adding $\alpha$ into HybridGNN-SR.

Two parameters $\lambda_1$ and $\lambda_2$ can be used to control the contribution of $L_{VGAE}$ and $L_D$, respectively. We set the large values for both $\lambda_1$ and $\lambda_2$ to balance the the scales of supervised and unsupervised losses. Table IV shows the results under different settings for $\lambda_1$ and $\lambda_2$. From the table, we can observe that different combination of $\lambda_1$ and $\lambda_2$ results in different recommendation performance. In a word, different weights of $L_{VGAE}$ and $L_D$ could perform better. With the help of Table III in which $L_D$ has a little impact on improvement of recommendation results, we tend to select a relatively bigger $\lambda_2$ and relatively smaller $\lambda_1$ to balance the contribution of $L_D$ and $L_{VGAE}$ .

TABLE II
PERFORMANCE COMPARED WITH BASELINES.

| Method | Yoochoose1/64 | | Yoochoose1/4 | | Diginetica | |
|---|---|---|---|---|---|---|
| | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ |
| POP | 6.71 | 1.65 | 1.33 | 0.30 | 0.89 | 0.20 |
| S-POP | 30.44 | 18.35 | 27.08 | 17.75 | 21.06 | 13.68 |
| Item-KNN | 51.60 | 21.81 | 52.31 | 21.70 | 35.75 | 11.57 |
| BPR-MF | 31.31 | 12.08 | 3.40 | 1.57 | 5.24 | 1.98 |
| FPMC | 45.62 | 15.01 | - | - | 26.53 | 6.95 |
| GRU4REC | 60.64 | 22.89 | 59.53 | 22.60 | 29.45 | 8.33 |
| NARM | 68.32 | 28.63 | 69.73 | 29.33 | 49.70 | 16.17 |
| STAMP | 68.74 | 29.67 | 70.44 | 30.00 | 45.64 | 14.32 |
| SR-GNN | 70.57 | 30.94 | 71.36 | 31.89 | 50.73 | 17.59 |
| FGNN | 71.12 | 31.68 | 71.97 | **32.54** | 51.36 | 18.47 |
| HybridGNN-SR | **71.68** | **31.89** | **72.10** | 31.65 | **54.01** | **18.56** |

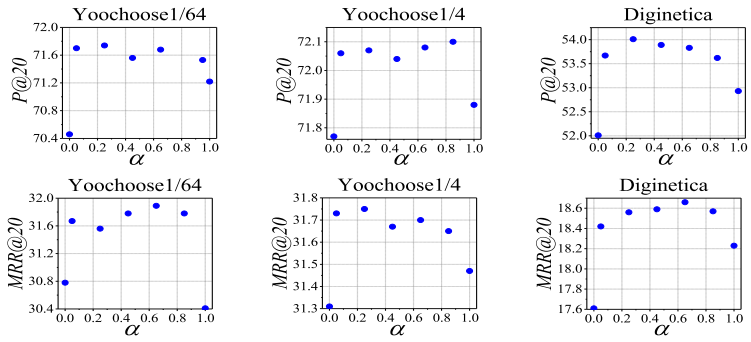


Fig. 2. Impact of $\alpha$ ($\lambda_1 = 10$, $\lambda_2 = 150$).

TABLE III
COMPARISON RESULTS OF HYBRIDGNN-SR AND ITS VARIANTS.

| Method | Yoochoose1/64 | | Yoochoose1/4 | | Diginetica | |
|---|---|---|---|---|---|---|
| | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ |
| HybridGNN-SR | **71.68** | **31.89** | **72.10** | **31.65** | **54.01** | **18.56** |
| \GS | 70.26 | 31.19 | 71.13 | 31.36 | 50.22 | 16.82 |
| \VGAE | 71.44 | 31.70 | 71.68 | 31.47 | 53.25 | 18.32 |
| \CN | 71.42 | 31.13 | 71.77 | 31.42 | 53.47 | 18.39 |
| \D | 71.63 | 31.64 | 71.91 | 31.59 | 53.65 | 18.45 |

TABLE IV
IMPACT OF TWO PARAMETERS $\lambda_1$ AND $\lambda_2$

| Method | | Yoochoose1/64 | | Yoochoose1/4 | | Diginetica | |
|---|---|---|---|---|---|---|---|
| $\lambda_1$ | $\lambda_2$ | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ | $P@20$ | $MRR@20$ |
| 10 | 10 | 71.55 | 31.82 | 72.01 | **31.83** | 54.00 | 18.51 |
| 10 | 150 | **71.68** | **31.89** | **72.10** | 31.65 | 54.01 | 18.56 |
| 150 | 10 | 71.67 | 31.65 | 72.00 | 31.74 | 53.89 | **18.61** |
| 150 | 150 | 71.60 | 31.65 | 72.04 | 31.72 | **54.13** | 18.50 |

## V. CONCLUSIONS

This paper proposes the HybridGNN-SR model to enhance the session-based recommendation quality from two aspects. On the one hand, we use the method of reconstructing adjacency information of each session graph to assist the recommendation task. On the other hand, we represent a session from the view of higher concepts by considering the adjacent items in a session. The experimental results on the three real-world datasets show the effectiveness of HybridGNN-SR. Further experiments demonstrate the impact of each component and the key hyperparameters. In future, we will add attributes of items into HypbridGNN-SR for performance improvements.

## ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China (61562009), and Program of Guizhou Provincial Science and Technology Department (No.[2019]2502).

## REFERENCES

[1] K. Cho and B. van Merrienboer and Ç. Gülçehre and D. Bahdanau and F. Bougares and H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in EMNLP, pp.1724–1734, 2014.
[2] M. Gori and G. Monfardini and F. Scarselli, "A new model for learning in graph domains," in IJCNN, pp. 729–734, 2005.
[3] B. Hidasi and A. Karatzoglou and L. Baltrunas and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks," in ICLR, 2016.
[4] T.N. Kipf and M. Welling, "Variational Graph Auto-Encoders," vol. abs/1611.07308, 2016.
[5] T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in ICLR, 2017.
[6] J. Klicpera and A. Bojchevski and S. Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," in ICLR, 2019.
[7] C. Li and Z.Y. Liu and M.M. Wu and Y.C. Xu and H. Zhao and P.P. Huang and G.L. Kang and Q.W. Chen and W. Li and D.L. Lee, "Multi-Interest Network with Dynamic Routing for Recommendation at Tmall," in CIKM, pp.2615–2623, 2019.
[8] C.L. Li and C. Quan and L. Peng and Y.W. Qi and Y.M. Deng and L.B. Wu, "A Capsule Network for Recommendation and Explaining What You Like and Dislike," in SIGIR, pp.275–284, 2019.

[9] J. Li and P.J. Ren and Z.M. Chen and Z.C. Ren and T. Lian and J. Ma, "Neural Attentive Session-based Recommendation," in CIKM, pp.1419–1428, 2017.
[10] Y.J. Li and D. Tarlow and M. Brockschmidt and R.S. Zemel, "Gated Graph Sequence Neural Networks," in ICLR, 2016
[11] Q. Liu and Y.F. Zeng and R. Mokhosi and H.B. Zhang, "STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation," in KDD, pp.1831–1839, 2018.
[12] A. Micheli, "Neural network for graphs: A contextual constructive approach, "in IEEE Transactions on Neural Networks, pp.498–511, 2009.
[13] F. Monti and M. Bronstein and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in NIPS, pp.3697–3707, 2017.
[14] S.R. Pan, R.Q. Hu, G.D. Long, J. Jiang, L. Yao, C.Q. Zhang, "Adversarially Regularized Graph Autoencoder for Graph Embedding", In IJCAI 2018, pages 2609-2615.
[15] R.H. Qiu and J.J. Li and Z. Huang and H.Z. Yin, "Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks," in CIKM, pp.579–588, 2019.
[16] Y.R. Qu and T. Bai and W.N. Zhang and J.Y. Nie and J. Tang, "An End-to-End Neighborhood-based Interaction Model for Knowledge-enhanced Recommendation," in DLPKDD, pp.555–563, 2019.
[17] S. Rendle and C. Freudenthaler and Z. Gantner and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," in UAI, pp.452–461, 2009.
[18] S. Rendle and C. Freudenthaler and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in WWW, pp.811–820, 2010.
[19] S. Sabour and N. Frosst and G.E. Hinton, "Dynamic Routing Between Capsules," in NeurIPS, pp. 3856–3866, 2017.
[20] W.P. Song and Z.P. Xiao and Y.F. Wang and L. Charlin and M. Zhang and J. Tang, "Session-Based Social Recommendation via Dynamic Graph Attention Networks," in WSDM, pp.555–563, 2019.
[21] P. Velickovic and G. Cucurull and A. Casanova and A. Romero and P. Liò and Y. Bengio, "Graph Attention Networks" in ICLR, 2018.
[22] P. Velickovic and W. Fedus and W.L. Hamilton and P. Liò and Y. Bengio and R.D. Hjelm, "Deep Graph Infomax," in ICLR, 2019.
[23] X. Wang and X.N. He and M. Wang and F.L. Feng and T.S. Chua, "Neural Graph Collaborative Filtering," in SIGIR, pp.165–174, 2019.
[24] H.W. Wang and M. Zhao and X. Xie and W.J. Li and M.Y. Guo, "Knowledge Graph Convolutional Networks for Recommender Systems," in KDD, pp.3307–3313, 2019.
[25] X. Wang and X.N. He and Y.X. Cao and M. Liu and T.S. Chua, "KGAT: Knowledge Graph Attention Network for Recommendation," in KDD, pp.950–958, 2019.
[26] S.J. Wang and L.B. Cao and Y. Wang, "A Survey on Session-based Recommender Systems," vol. abs/1902.04864, 2019.
[27] S.J. Wang and L. Hu and Y. Wang and L.B. Cao and Q. Z. Sheng and M.A. Orgun, "Sequential recommender systems: challenges, progress and prospects," In IJCAI, arXiv:2001.04830, pp. 6332-6338, 2020.
[28] S.J. Wang and L. Hu and Y. Wang and L.B. Cao and Q. Z. Sheng and M. Orgun and L.B. Cao, "Modeling Multi-Purpose Sessions for Next-Item Recommendations via Mixture-Channel Purpose Routing Networks," In IJCAI, pp.3771–3777, 2019.
[29] S.J. Wang and L. Hu and Y. Wang and L.B. Cao and Q. Z. Sheng and M. Orgun, "Intention2Basket: A Neural Intention-driven Approach for Dynamic Next-basket Planning," In IJCAI, pp.2333-2339, 2020.
[30] L. Wu and P.J. Sun and Y.J. Fu and R.C. Hong and X.T. Wang and M. Wang, "A Neural Influence Diffusion Model for Social Recommendation," in SIGIR, pp.235–244, 2019.
[31] S. Wu and Y.Y. Tang and Y.Q. Zhu and L. Wang and X. Xie and T.N. Tan, "Session-Based Recommendation with Graph Neural Networks," in AAAI, pp.346–353, 2019.
[32] C.F. Xu and P.P. Zhao and Y.C. Liu and V.S. Sheng and J.J. Xu and F.Z. Zhuang and J.H. Fang and X.F. Zhou, "Graph Contextualized Self-Attention Network for Session-based Recommendation," in IJCAI, pp.3940–3946, 2019.
[33] R. Ying and R. He and K. Chen and P. Eksombatchai and W. L. Hamilton and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems,"in KDD, pp.452–461, 2018.