Incremental Graph Convolutional Network for Collaborative Filtering

Jiafeng Xia
Fudan University
Shanghai, China
jfxia19@fudan.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsli@microsoft.com

Hansu Gu*
Seattle, United States
hansug@acm.org

Tun Lu*
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Peng Zhang
Fudan University
Shanghai, China
zhangpeng_@fudan.edu.cn

Ning Gu
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

ABSTRACT

Graph neural networks (GNN) recently achieved huge success in collaborative filtering (CF) due to the useful graph structure information. However, users will continuously interact with items, which causes the user-item interaction graphs to change over time and well-trained GNN models to be out-of-date soon. Naive solutions such as periodic retraining lose important temporal information and are computationally expensive. Recent works that leverage recurrent neural networks to keep GNN up-to-date may suffer from the "catastrophic forgetting" issue, and experience a cold start with new users and items. To this end, we propose the incremental graph convolutional network (IGCN) — a pure graph convolutional network (GCN) based method to update GNN models when new user-item interactions are available. IGCN consists of two main components: 1) a historical feature generation layer, which generates the initial user/item embedding via model agnostic meta-learning and ensures good initial states and fast model adaptation; 2) a temporal feature learning layer, which first aggregates the features from local neighborhood to update the embedding of each user/item within each subgraph via graph convolutional network and then fuses the user/item embeddings from last subgraph and current subgraph via incremental temporal convolutional network. Experimental studies on real-world datasets show that IGCN can outperform state-of-the-art CF algorithms in sequential recommendation tasks.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

collaborative filtering, incremental recommendation, graph neural network

*Corresponding author.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482354

ACM Reference Format:
Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2021. Incremental Graph Convolutional Network for Collaborative Filtering. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482354

1 INTRODUCTION

Collaborative filtering (CF) algorithms are playing a vital role in today's recommender systems due to high accuracy and broad applicability [1, 8]. During recent years, graph neural network (GNN) based CF algorithms have achieved state-of-the-art performance in many kinds of recommendation tasks, e.g., home/news feed [36], social network [10, 35], e-commerce [40], etc. The main advantages of GNNs are: 1) the ability to leverage local neighborhood information among users and/or items [36], in which the behaviors of neighbouring users/items within several hops on the graph can help to facilitate representation learning; and 2) the flexibility to incorporate auxiliary information, e.g., social information [10], knowledge graphs [29], etc., which can help to alleviate the well-known data sparsity issue in collaborative filtering.

In real-world recommender systems, users will continuously interact with items, which makes the user-item interaction graphs to be continuously changing over time, so the well-trained GNN models will soon become out-of-date. One trivial way to solve this issue is to retrain the GNN models after obtaining new interactions. However, retraining is computationally expensive, and more importantly useful temporal information may lose during retraining because recent interactions are often treated as equally important as old interactions. Recent works, such as DeepCoevolvo [7] and JODIE [19], tried to update the user/item embedding vectors with new interactions using recurrent neural networks (RNNs). However, these RNN-based methods may suffer from the "catastrophic forgetting" issue [12], which may achieve inferior performance on users/items with long interaction sequences as demonstrated in our empirical studies.

To this end, this paper proposes an incremental graph convolution network (IGCN), which enables incremental model learning via graph convolutional networks on the user-item interaction graphs. IGCN consists of two key components to learn temporal-aware user/item embedding: one is a historical feature generation

layer, which generates the initial user/item embedding via model agnostic meta-learning to solve the new users/items issue and the data sparsity issue in each subgraph and ensure fast model adaptation, the other one is a temporal feature learning layer, which first aggregates the features from local neighborhood to update the embedding of each user/item within each subgraph via graph convolutional network (GCN) and then fuses the user/item embeddings from last subgraph and current subgraph via incremental temporal convolutional network (iTCN) . After learning user/item embedding, a preference decoder layer is employed to recommend items to users based on the learned embedding. Experimental studies on several real-world datasets demonstrate that IGCN can outperform state-of-the-art CF algorithms in sequential recommendation tasks, including GNN-based sequential recommendation methods and other incremental recommendation methods.

The key contributions of this work are summarized as follows:

- We propose a novel incremental temporal convolutional network (iTCN) which fuses temporal information from last period and current period incrementally and can achieve efficient and accurate temporal-aware feature learning.
- We propose IGCN, which is a novel GNN based incremental recommendation algorithm that uses 1) GCN to extract higher level interaction features on user-item interaction graph, 2) iTCN to learn user and item temporal-aware features and 3) model agnostic meta-learning (MAML) to initialize user and item historical embeddings to alleviate cold start issue and ensure fast model adaptation.
- We conduct extensive experiments on five real-world datasets, and the results show that IGCN can outperform state-of-the-art recommendation methods, including GNN-based sequential methods and other incremental methods.

2 RELATED WORK

Collaborative filtering algorithms have achieved superior performances compared to other recommendation methods, from early simple methods [16, 25], factorization methods [4, 18, 20, 21] to recent deep learning methods [5, 6, 15, 26, 39]. Recently, graph neural network based CF algorithms further pushed the state-of-the-art performances by incorporating useful user-item interaction graph structure information in representation learning [14, 19, 29]. The GCMC method [3] introduced an AutoEncoder into the user-item interaction graph to predict the possible ratings. On top of previous factorization-based CF methods, such as matrix factorization [18] and FISM [17], NGCF [32] further extracted high-level features by building high-order connectivities through GNN, thus improved the performance. LightGCN [14] improved over NGCF by removing two unnecessary feature transformation and nonlinear activation to improve both the efficiency and the accuracy of recommendations. It is worth noting that GNN updates a node's feature by aggregating the information of all its neighboring nodes. With the increasing of GNN layers, each node can obtain the information of neighboring nodes with larger distances, which can potentially improve the accuracy if the oversmoothing issue is properly handled [14].

However, all above GNN-based CF methods suffer from one common issue: a well-trained model may be soon out-of-date due to the graph structure change caused by new interactions. To address this issue, DeepCoevolve [7] and JODIE [19] adopt RNNs to update the node features to tackle the incremental update issue of GNN-based CF methods. However, these RNN-based methods may suffer from the well-known "catastrophic forgetting" issue when dealing with long sequences [12, 13], and experience a cold start with new users and items. TGN [24] proposes a framework for learning continuous-time dynamic graphs, which can accurately capture the changes of nodes' feature over time through memory module and embedding module. Incremental learning is another line of works to solve the above problem when new user-item interactions are available. SPMF [31] is probabilistic matrix factorization (PMF) based method, which employs a reservoir to maintain historical data and uses the data in the reservoir plus the new observations to update the model and make recommendations. IncCTR [33] uses a data module and a feature module to construct training data and handle features respectively, and uses a model module to fine-tune the model parameters with knowledge distillation. SML [38] employs a neural network-based transfer learning component to transforms the old model to a new model during training, and optimizes the recommendation accuracy evaluated in the next time period to learn the transfer learning component. However, none of them was proposed for GNN-based collaborative filtering algorithms.

3 INCREMENTAL GRAPH CONVOLUTIONAL NETWORK

This section first presents IGCN in high-level and then introduces the details of IGCN. Finally, we show how to train IGCN model.

3.1 Overview of IGCN

In GNN-based CF algorithms, newly observed user-item interactions can be regarded as a subgraph, which only contains the users and items (as nodes) and their interactions (as edges) over a short period of time, e.g., one week, one month. While in the industrial recommendation, a large amount of data will be generated every minute and IGCN can also complete real-time update when new data arrives by adjusting the time interval to a smaller value. In this paper, we focus on the problem of learning from the newly observed subgraphs in an incremental fashion to make GNN-based CF algorithms better in line with real-world recommender systems. As shown in Figure 1, IGCN consists of three key components:

- *A historical feature generation layer*, which generate user/item embedding, named as historical embedding, before the first period with model agnostic meta learning (MAML) [11] to address two challenges: 1) Fast model adaptation. Since user ratings are rare in each time period, MAML-based initialization will help to achieve faster convergence with few ratings in each time period. 2) New users or items. Since there might be users/items that only appear in the test time, initializing their embedding via MAML can achieve much better performance than random initialization [27].
- *A temporal feature learning layer*, which aggregate the features from local neighborhood to update the embedding of each user/item within each subgraph via graph convolutional network (GCN) and then fuse the features from the user/item embedding of last and current subgraphs via incremental temporal convolutional network (iTCN). iTCN

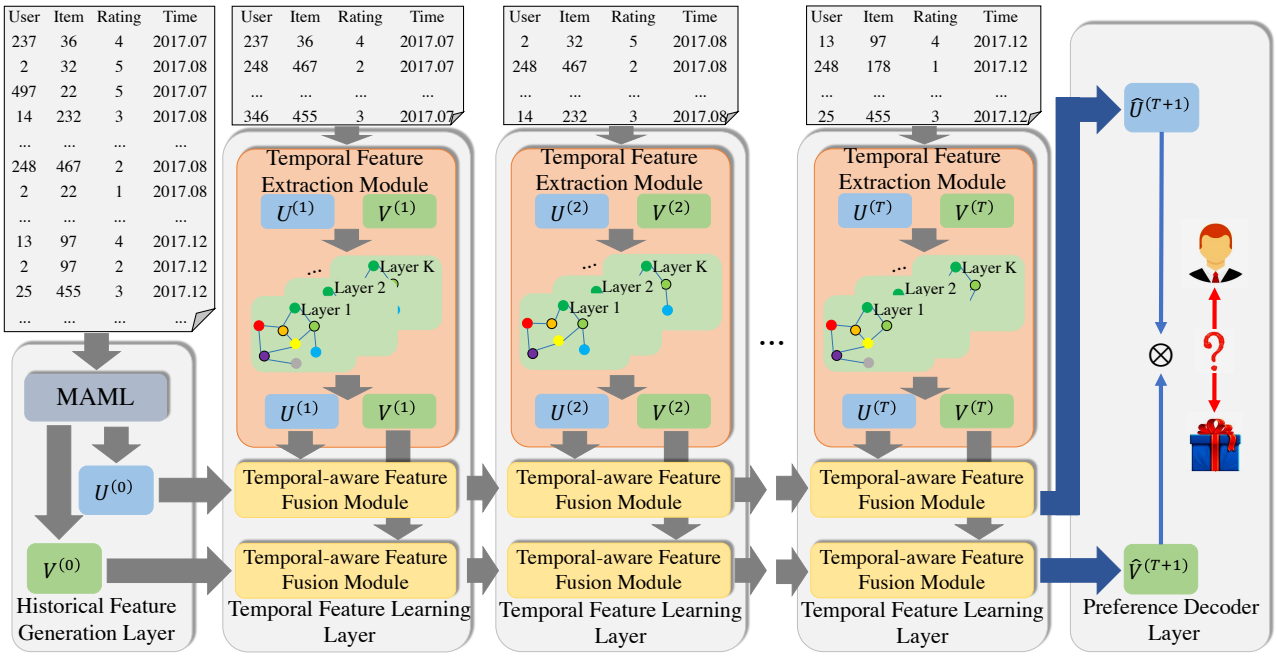

Figure 1: The architecture of the proposed incremental graph convolution network method.

employs dilated causal convolutions to ensure: 1) large receptive field, since hierarchical dilated convolutions can yield exponentially growing receptive field and 2) causality, since the causal convolutions can ensure current features are only relied on all past features [2].

- *A preference decoder layer*, which is used to predict the possibility of interaction between a given user and a given item. We choose dot product as the interaction function due to its superiority in both accuracy and efficiency [23] while MLP and other interaction functions can also work here.

Based on the above novel design, IGCN has the following advantages: 1) incremental training. IGCN only needs to train a temporal feature extraction module on the subgraph formed by recent interactions, then retrain the temporal-aware feature fusion module for users and items respectively to automatically fuse the new features with features from last period of time. Both two modules are only performed on new user-item interactions, i.e., the training of IGCN is incremental; 2) temporal-awareness and global-awareness. After training based on the most recent interactions, the temporal-aware feature fusion module can automatically discover the most important features to predict temporal user behaviors and item attributes (temporal-awareness) by scanning over the entire user and item feature trajectories (global-awareness). Therefore, IGCN can address the temporal information lost issue in model retraining and the "catastrophic forgetting" issue in RNN-based methods.

3.2 Historical Feature Generation Layer

Incremental model updates in a sequential way will raise two challenges: 1) Sparse data in each time period. Recommender systems usually suffer from the data sparsity issue, which will be even more severe when we only consider the ratings in a small period of time; 2) New user or item issue. Users or items that only appear in the test time can not initialize their embedding reasonably and traditional random or constant initialization often leads to poor performance [27]. As shown in Algorithm 1, we propose to leverage model agnostic meta learning (MAML) to initialize the historical embedding when they are unavailable, which can also help to achieve faster model adaptation even when there are very few ratings in each subgraph.

Unlike traditional model training, which divides the dataset into training, validation and test sets, MAML uses a task as the minimum unit of the dataset. For each task in MAML, we randomly select some users with their interactions from the training set, and divide them into a support set and a query set. These two sets are used to calculate support and query losses respectively (line 1–5 in Algorithm 1). The model parameters e_u and e_i are randomly initialized to start the MAML process (line 6), and are updated multiple times until they converge (line 7–16). Specifically, in each update, we first sample multiple tasks as a batch from the task set (line 8). Then for each sampled task, we generate user historical representations $E_u \in \mathbb{R}^{M' \times d}$ and item historical representations $E_i \in \mathbb{R}^{N' \times d}$ by repeating $e_u \in \mathbb{R}^d$ M' times and $e_i \in \mathbb{R}^d$ and N' times respectively, where M' and N' are the number of users and items in that task (line 9). Historical representations E_u and E_i are trained with temporal-aware feature fusion module (which will be introduced later) to learn user/item representations in the next period and calculates the loss to update the parameter e_u and e_i using gradient descent on the support set (line 10–11). We regenerate new user and item historical representations from the updated e'_u and e'_i, train E'_u and E'_i, and calculate the loss on the query set (line 13). Finally, after all tasks have been used to update parameters, we

Algorithm 1 MAML for model parameter pretraining

Input: User-item interaction data D, number of tasks I, size of each task H, number of sampled tasks K, learning rates α and β.
Parameters: user historical representation e_u and item historical representation e_i.
Output: Pretrained e_u and e_i.

1: TaskSet = $\emptyset$.
2: **for** $i = 1, \cdots, I$ **do**
3: Randomly sample H users with their interactions from D to generate support set S_i and query set Q_i.
4: TaskSet = TaskSet $\cup$ $\{(S_i, Q_i)\}$.
5: **end for**
6: Randomly initialize e_u and e_i.
7: **while** not done **do**
8: Sample K tasks from TaskSet.
9: **for** $k = 1, \cdots, K$ **do**
10: Generate E_u and E_i using e_u and e_i respectively.
11: Train E_u and E_i on support set S_k and compute loss $L(S_k)$.
12: Update parameter using gradient descent : $e'_u = e_u - \alpha \nabla_{e_u} L(S_k)$, $e'_i = e_i - \alpha \nabla_{e_i} L(S_k)$.
13: Train E'_u and E'_i generated from e'_u and e'_i on query set Q_k and compute loss $L(Q_k)$.
14: **end for**
15: Update parameter using gradient descent: $e_u = e_u - \beta \nabla_{e_u} \sum_{k=1}^{K} L(Q_k)$, $e_i = e_i - \beta \nabla_{e_i} \sum_{k=1}^{K} L(Q_k)$.
16: **end while**

sum the loss on the query set across all tasks to complete the final update of parameter e_u and e_i (line 15).

Intuitively, MAML can help IGCN to better initialize user and item representations by encoding some global information, e.g., which items are popular among users. When new users/items appear, their historical representations can be reasonably initialized by MAML. In addition, when new user-item interactions are available, user/item representations can quickly adapt and converge to the optimal ones.

3.3 Temporal Feature Learning Layer

Temporal feature learning layer consists of three modules, including one temporal feature extraction module and two temporal-aware feature fusion modules. Temporal feature extraction module is used to extract temporal information from user-item interactions of current period, while temporal-aware feature fusion modules are used to capture the pattern on how user and item features are changing over time. Next, we introduce temporal feature extraction module and temporal-aware feature fusion module in detail, and then we introduce temporal feature learning layer that combines the above two modules to achieve incremental recommendation. Finally, we analyze the efficiency of the temporal feature learning layer.

3.3.1 Temporal feature extraction module. For each subgraph of user-item interactions over a short period of time (e.g., one month), we employ graph convolutional network to learn the temporal feature of these users and items. The graph convolution layer aims to extract features of interactions between users and items on graph and enrich the representation learning of each node by aggregating features from its neighboring nodes. Since the graph convolutions for different subgraphs are in the same manner, we describe the graph convolution operations for a specific time period t without loss of generality.

To ensure fast model training, we first pretrain the node embedding of the subgraph at time period t with matrix factorization [18]. Let $\mathbf{U}^{(t)} \in \mathbb{R}^{M_t \times d}$ and $\mathbf{V}^{(t)} \in \mathbb{R}^{N_t \times d}$ be the user feature matrix and item feature matrix after pretraining in time period t, respectively, where M_t and N_t are the number of users and items who have interactions during time period t. The user-item interaction data in time period t can form a user-item bipartite graph G_t. To better capture the neighborhood information, we stack multiple graph convolutional layers to achieve more comprehensive feature aggregation. Let $\mathbf{u}_i^l$ and $\mathbf{u}_i^{l+1}$ be user u_i's feature vector in the l-th layer and $l+1$-th layer, respectively, then the $l+1$-th graph convolutional layer works as follows:

$$\mathbf{u}_{N(i)}^l = Aggregate(\{\mathbf{v}_j^l, \forall j \in N(i)\}), \tag{1}$$

$$\mathbf{u}_i^{l+1} = \sigma(W^{l+1} \cdot (\mathbf{u}_{N(i)}^l \oplus \mathbf{u}_i^l) + b^{l+1}), \tag{2}$$

where $\mathbf{u}_{N(i)}^l$ is the aggregated feature vector from u_i's neighbors and $N(i)$ is the set of u_i's neighbors. $\oplus$ is an element-wise addition operation. W^{l+1} and b^{l+1} are the weight matrix and bias term in $l+1$-th layer and $\sigma(\cdot)$ is an activation function. $Aggregate(\cdot)$ is an aggregation function that aggregates features from the neighbors of each node. There could be different options for the aggregation function, such as the simple element-wise sum or mean. However, simple sum or mean may lose the varying importance of different neighbors, so that we propose an importance-aware aggregation function as follows:

$$\mathbf{u}_{N(i)}^l = \sum_{j \in N(i)} c_{ij} \cdot \mathbf{v}_j^l, \tag{3}$$

$$c_{ij} = \frac{\mathbf{R}_{ij}^*}{\sqrt{D(i) \cdot D(j)}}, \ \mathbf{R}_{ij}^* = \mathbf{R}_{ij} - \overline{\mathbf{R}}_i, \tag{4}$$

where $\mathbf{R}_{ij}$ is the rating that user u_i gives to item v_j and $\overline{\mathbf{R}}_i$ is the mean of u_i's ratings. We use $\mathbf{R}_{ij}^*$ to normalize $\mathbf{R}_{ij}$ aiming to make the GCN converge faster. $D(i)$ and $D(j)$ are u_i's degree and v_j's degree, respectively. The design of c_{ij} is based on two intuitions: 1) higher rating indicates stronger correlation between u_i and v_j and vice versa and 2) large node degrees indicates weaker correlation between u_i and v_j and vice versa, i.e., the node with larger degrees should not be easily influenced by only one neighbor. For items, we update their embeddings in the same way:

$$\mathbf{v}_j^{l+1} = \sigma(W^{l+1} \cdot (\mathbf{v}_{N(j)}^l \oplus \mathbf{v}_j^l) + b^{l+1}). \tag{5}$$

$$\mathbf{v}_{N(j)}^l = \sum_{i \in N(j)} c_{ji} \cdot \mathbf{u}_i^l, \tag{6}$$

$$c_{ji} = \frac{\mathbf{R}_{ji}^*}{\sqrt{D(i) \cdot D(j)}}, \ \mathbf{R}_{ji}^* = \mathbf{R}_{ji} - \overline{\mathbf{R}}_j, \tag{7}$$

where $\overline{\mathbf{R}}_j$ is the mean of v_j's ratings. Based on the above graph convolutional operations, we can iteratively update the user/item embedding of a subgraph until convergence.

3.3.2 Temporal-aware feature fusion module. After learning the temporal features of users/items in each subgraph, a feature fusion module is required to aggregate the user/item features from different time periods to achieve accurate recommendation on most recent user-item interactions. Since user interests may drift over time [19], it is important to model the dynamic characteristics of users and items in each period of time, i.e., the fused user/item features should be temporal-aware.

Existing work [12] leverage recurrent neural networks (RNNs) to capture temporal dynamics, which often suffer from the "catastrophic forgetting" issue and may not work well over long sequences of interaction history. Convolutional neural networks (CNNs) have recently shown promising performance on sequential data [28]. However, they cannot be directly applied due to: 1) violation of sequential data by considering future features in the prediction; 2) restricted receptive field that usually not capable of capturing earlier historical information from users and items. TCN [2] is a promising solution to address the above two issues with a specific design of dilated and causal convolutions. As shown in Figure 2, TCN can obtain the feature information of earlier periods and force the causality of the prediction results, which means the prediction of current period is only related to the past time periods, not related to the future time periods.

However, TCN is not optimal in the incremental recommendation problem. Firstly, TCN is an N-to-N structure, which requires the input representations from N periods as a sequence at the same time, and then outputs N predicted representations. While in incremental recommendation, it usually requires a 1-to-1 structure that takes the input representation of the current period and outputs the predicted representation for the next period. Secondly, TCN requires to save all the historical data and the intermediate results of the model, which greatly increases model storage consumption in the training phase. Therefore, we design an incremental temporal convolutional network (iTCN) –an improved version of TCN to achieve incremental and temporal-aware feature fusion.

The details of the iTCN unit are illustrated in the middle part of Figure 2. Compared with the TCN, iTCN simplifies the network architecture, where the residual connection is removed and hierarchical multi-layer convolutions are simplified to a single convolution. The simplification not only learns more accurate temporal-aware features, but also reduces the storage consumption of the training process. More detailed accuracy and efficiency comparison between iTCN and TCN will be presented in Section 4.4. Similar to TCN, iTCN ensures causality (future interactions are not leverage in current time period) and large receptive field by dilated and causal convolutions. In order to achieve 1-to-1 incremental recommendation, we design a storage unit for iTCN, which can be used to store the historical data of a given length, based on the setting of the current iTCN. Let d and ks be the dilation and kernel size of current iTCN, and the storage length is $d * (ks - 1)$. With the designed storage unit, iTCN only needs to store a small amount of historical data, therefore achieving much higher storage efficiency than TCN.

Formally, iTCN in time period t and layer l works as follows:

$$\mathbf{H}_t^l = Conv(concat(\{\mathbf{X}_{t-d\cdot i}^l | i = 0, 1, \cdots, ks-1\})), \tag{8}$$

$$\mathbf{X}_t^{l+1} = Dropout(\sigma(\mathbf{H}_t^l)), \tag{9}$$

where $\mathbf{X}_t^{l+1}$ and $\mathbf{X}_t^l$ are the output and input of iTCN in time period t and layer l, and in the first layer of iTCN (i.e. $l = 1$), $\mathbf{X}_t^l$ can be $\mathbf{U}^{(t)}$ for user's temporal-aware feature fusion module or $\mathbf{V}^{(t)}$ for item's temporal-aware feature fusion module. $Dropout(\cdot)$ and $Conv(\cdot)$ are dropout and convolutional layers, respectively. $\sigma(\cdot)$ is an activation function, which is chosen as ReLU in this paper. When the training for time period t is completed, $\mathbf{X}_t^l$ will be added to the storage unit, and $\mathbf{X}_{t-d\cdot(ks-1)}^l$ will be removed out of the storage unit, so the data $\mathbf{X}_{t-d\cdot(ks-1)+1:t}^l$ in the storage unit will be used for the training in time period $t+1$.

The temporal-aware feature fusion module shown in the right part of Figure 2 is composed by stacking one or more iTCNs. An identity mapping component is added to help to address the vanishing gradient problem when stacking multiple iTCNs. Thus, the final output of the fusion module can be described as follows:

$$\mathbf{O}_t = \sigma(\mathbf{X}_t^L + \mathbf{X}_t), \tag{10}$$

where $\mathbf{X}_t$ and $\mathbf{O}_t$ are the input and output of the temporal-aware feature fusion layer in period t, $\mathbf{X}_t^L$ is the output of the iTCN in the last layer (i.e. $l = L$) and $\sigma(\cdot)$ is an activation function.

3.3.3 Efficiency Analysis. In order to achieve incremental recommendation, the main body of IGCN is made by recurring temporal feature learning layer multiple times, each recurrence corresponds to a time period. In terms of data storage, IGCN does not require to store all the historical data to complete the training, but only stores a small amount of historical data. When taking interactions from a new time period, IGCN will update the stored historical data with the new data at the same time. It is worth noting that we set different W and b for different periods in GCN in temporal feature extraction module, instead of sharing the same parameters among periods, so that the features extracted from the user-item interaction network in each period with GCN are not affected by the other periods, which can better represent the features of users and items in that period. Regarding time efficiency, for each new time period, IGCN only needs to execute one matrix factorization operation, one graph convolution operation and a few convolution operations to complete the interaction prediction in the next period, so the training process is rather efficient. We will analyze the time efficiency of IGCN in detail in Section 4.4.

3.4 Preference Decoder Layer

After fusing the user/item features in different time periods, we use the dot product to predict user ratings on items using their feature vectors in previous period as follows:

$$\hat{r}_{ij}^{(t)} = \mathbf{u}_i^{(t-1)} \cdot \mathbf{v}_j^{(t-1)^T}. \tag{11}$$

$\mathbf{u}_i^{(t-1)}$ and $\mathbf{v}_j^{(t-1)}$ are u_i's and v_j's feature vectors in time period $t-1$. respectively. Then, $\hat{r}_{ij}^{(t)}$ is the predicted rating in time period t that u_i may give to v_j. Other preference decoders, e.g., MLP, can also be adopted, but we choose dot product due to higher efficiency and comparable accuracy [23].

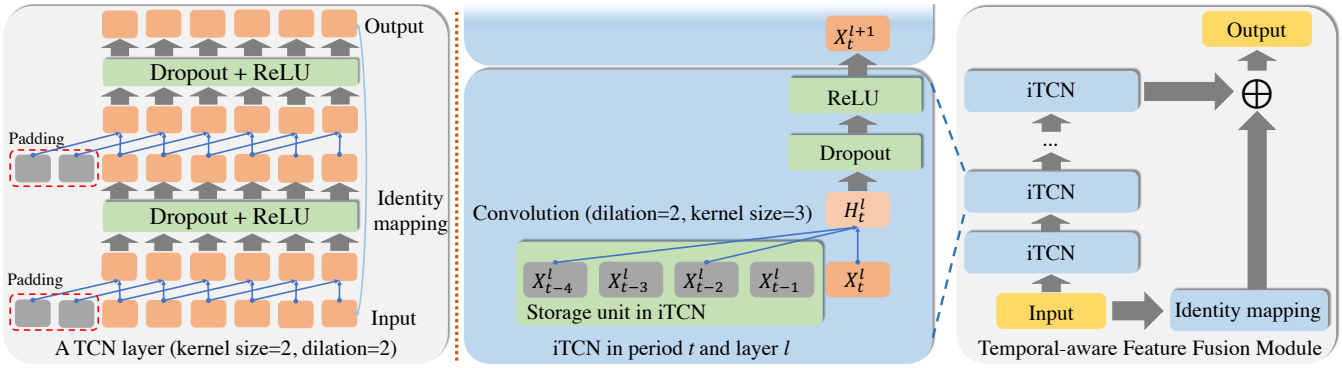

Figure 2: The architecture of TCN layer (left part), iTCN (middle part) and temporal-aware feature fusion module (right part).

3.5 Model Training

3.5.1 Loss Function. For explicit feedback datasets, we choose the commonly used mean square loss as follows:

$$L = \frac{1}{N} \sum_{t=1}^{T} \sum_{i,j \in O^{(t)}} (\hat{r}_{ij}^{(t)} - r_{ij}^{(t)})^2. \tag{12}$$

N is the total number of ratings in the training data. $O^{(t)}$ is the set of observed ratings in time period t. $r_{ij}^{(t)}$ is the true rating that u_i gives to v_j in time period t.

For implicit feedback datasets, we choose the commonly used cross entropy loss as follows:

$$L = -\frac{1}{N} \sum_{t=1}^{T} \sum_{i,j \in O^{(t)}} (r_{ij}^{(t)} \log \hat{r}_{ij}^{(t)} + (1 - r_{ij}^{(t)}) \log(1 - \hat{r}_{ij}^{(t)})). \tag{13}$$

3.5.2 Training Process. The training process of IGCN works as follows. In each time period t, we use the interactions observed during time period t to train the model for multiple epochs. Rather than directly using the parameters from the last epoch where model may suffer overfitting problem, we select the parameters with the lowest loss on the validation data and set them as the initial values of the parameters in the training of time period $t + 1$.

4 EXPERIMENT

4.1 Experimental Setup

4.1.1 Dataset. In the experiments, we use five well-known datasets to evaluate the performance of IGCN. In the MovieLens 100K dataset, we split all ratings into 8 time periods and use last 2 time periods as test data. The Netflix Prize dataset is huge so that we select a 4 months subset and keep top 1,000 users and their ratings, then we use ratings from the first three month in training and the last month for test. For the MovieLens 1M dataset, we choose a dense 8 months period containing about 90% of ratings and then use the ratings from the first 7 months in training and the rest in test. In the Amazon Book dataset, we first choose the most dense 9 months period and keep the top 4,000 users and items, then we use the ratings from the first 7 months in training and the rest in test. In the Yelp dataset, we select a 7 years subset and keep top 15,000 users and 25,000 items, then we use ratings from the first six year in training and the last year for test. Table 1 describes the detailed statistics of the five datasets used in the experiments. It should be noted that due to the sparsity of datasets, we set a large time interval for each dataset. It is not necessary to set the interval to a small value because the update of the model is too small. However, in the industrial recommendation, a large amount of interactions is generated every minute. In that case, we can specify a small value for the time interval.

4.1.2 Metrcis. We compare IGCN with other methods in top-N recommendation task, so that We choose three popular ranking metrics to evaluate the model performance: 1) F1-score, which is the harmonic mean of precision and recall; 2) Mean Reciprocal Rank (MRR), which evaluates the performance by the ranking of the first real item in the recommendation lists; and 3) Normalized Discounted Cumulative Gain (NDCG), which evaluates the gap between recommended item lists and optimally ranked item lists. Note that we only use four-star or five-star ratings as positive ones in top-N recommendation.

4.1.3 Baselines. IGCN is compared against the following eleven state-of-the-art CF methods:

- BPR [22] is a classical CF algorithm, which minimizes a pairwise loss over positive and unlabeled data. In BPR, we use MF to initialize users' and items' embedding.
- LambdaFM [37] combines LambdaRank and FM for recommendation with pairwise loss function.
- IRGAN [30] expands GAN to discrete data in information retrieval by introducing policy gradient and achieves good performance on several tasks.
- LightGCN [14] is a GNN based CF algorithm which simplify the structure of graph convolutional network by removing feature transformation and nonlinear activation.
- RRN [34] uses two RNNs to model users' and items' temporal feature respectively. In RRN, we omit stationary components such as user profile and item genre for fairness comparison.
- GCMCRNN [9] uses GCMC [3] to learn spatial structure of user-item subgraphs in all periods and then uses RNN to extract temporal feature. There are two variants — GCMCRNN (sep) and GCMCRNN (inc) where data in each period are split separately or incrementally respectively.

Table 1: The statistics of the five datasets. Here, time span is described by year and month, e.g., 97.09 means September 1997.

	MovieLens 100K	Netflix	MovieLens 1M	Amazon Books	Yelp
# Users	943	1,000	6,040	4,000	15,000
# Items	1,682	13,104	3,706	4,000	25,000
# Ratings	100,000	605,108	892,982	73,408	1,025,124
Training time span	97.09–98.02	05.09–05.11	00.05–00.11	13.01–13.07	13.01–18.12
Test time span	98.03–98.04	05.12	00.12	13.08–13.09	19.01–19.12
# Training ratings	77,836	524,859	777,387	59,807	891,914
# Test ratings	22,164	80,321	115,595	13,601	133,210
Density	0.063	0.046	0.045	0.005	0.003

Table 2: Summary of the eleven compared methods: non-deep learning method (ND), deep learning method (D), non-temporal method(NT), temporal method (T), non-graph based method (NG), graph based method (G), non-incremental method (NI) and incremental method(I).

Baseline	ND	D	NT	T	NG	G	NI	I
BPR	✓		✓		✓		✓	
LambdaFM	✓		✓		✓		✓	
IRGAN		✓	✓		✓		✓	
LightGCN		✓	✓			✓	✓	
RRN		✓		✓	✓		✓	
GCMCRNN		✓		✓		✓	✓	
DeepCoevolve		✓		✓		✓	✓	
JODIE		✓		✓		✓	✓	
SPMF	✓			✓	✓			✓
IncCTR		✓		✓	✓			✓
SML		✓		✓	✓			✓

- DeepCoevolve [7] uses RNN to model the intensity function in point process to capture feature evolving over time on the interaction graph. Note that we disabled their user/item embedding updates in test phase due to fair comparison.
- JODIE [19] is a graph based algorithm that employs two RNNs to learns the embedding trajectories of users and items respectively. Note that we disabled the user/item embedding updates in test phase due to fair comparison.
- SPMF [31] employs a reservoir to maintain historical data and uses the data in the reservoir plus the new observations to update the recommendation.
- IncCTR [33] uses a data module and a feature module to construct training data and handle features respectively, and uses a model module to fine-tune the model parameters.
- SML [38] is a state-of-the-art incremental CF method that employs a neural network-based transfer component to transforms the old model to a new model during training.

Table 2 further summaries the categories of the compared state-of-the-art collaborative filtering algorithms in the experiments.

4.1.4 Implementation Details. We use Adam optimizer with the initial learning rates from [0.0005, 0.001, 0.005] in the training. For all datasets, initial embedding size is fixed to 60. In meta-learning, we tune the number of tasks from [5, 7, 10] and task size from [150, 200, 250, 300] for MovieLens and Netflix, and tune the number of tasks from [10, 12, 14] and task size from [400, 500] for Amazon Books and Yelp. In the GCN part of the temporal feature extraction module, the number of layers is fixed to 3 and the embedding size in each layer is fixed to 60. For iTCN, the kernel size is fixed to 3, and dropout rate is searched from [0.45, 0.5, 0.55]. In the training process of IGCN, the hyper-parameters in each period are kept the same. We tune the hyper-parameters of all the methods (including IGCN and the compared methods) according to the performance on the validation set in the last time period of the training phase.

4.2 Accuracy Comparison

Table 3 compares the performance of IGCN with eleven state-of-the-art CF methods in all five datasets, which shows that IGCN significantly outperforms all the compared methods. Besides, we have the following observations from the results.

1) Sequential methods outperform non-sequential methods. The main reason is that non-sequential methods use all ratings to learn static features while ignoring the dynamic features of user/items. On the contrary, sequential methods can learn dynamic features across different time periods, which can better capture the user preference drifts and thus achieve better performance.

2) Adopting graph structure can help to improve recommendation accuracy as shown in the results, i.e., GCMCRNN, LightGCN, DeepCoevolove and JODIE outperform the other baseline methods in most cases. The main reason is that graph structure provides additional information to CF algorithms, which can help users/items to enhance their representations from their neighborhoods.

3) IGCN consistently outperforms all the state-of-the-art methods on all datasets. The main reasons are: a) the temporal-aware feature fusion is achieved via CNN rather than RNNs, which can address the "catastrophic forgetting" issue in RRN, GCMCRNN, DeepCoevolove and JODIE; b) the MAML-based parameter initialization can further improve the performance especially on user/item with few or even no ratings. Compared with RNN, iTCN enables each user/item to enrich its own feature learning by extracting information from its previous features in earlier time periods without any forgetting mechanism, which can help to improve the performance.

Table 3: Accuracy comparison between IGCN and the state-of-the-art methods in five datasets. Note that JODIE cannot finish training within a reasonable time on Yelp, so that the results are not reported here. The results on the other four datasets can confirm the superior performance of IGCN compared to JODIE. Bold face indicates the highest accuracy.

Models	MovieLens 100K		Netflix		MovieLens 1M		Amazon Books		Yelp	
	F1-score@5	F1-score@10	F1-score@5	F1-score@10	F1-score@5	F1-score@10	F1-score@5	F1-score@10	F1-score@5	F1-score@10
BPR	0.047 ± 0.001	0.064 ± 0.005	0.012 ± 0.002	0.016 ± 0.002	0.019 ± 0.000	0.027 ± 0.001	0.013 ± 0.000	0.013 ± 0.001	0.003 ± 0.000	0.004 ± 0.000
LambdaFM	0.035 ± 0.002	0.035 ± 0.001	0.008 ± 0.000	0.010 ± 0.000	0.020 ± 0.001	0.027 ± 0.002	0.003 ± 0.000	0.003 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
IRGAN	0.054 ± 0.008	0.073 ± 0.010	0.012 ± 0.002	0.018 ± 0.002	0.035 ± 0.001	0.049 ± 0.001	0.006 ± 0.001	0.007 ± 0.000	0.002 ± 0.000	0.003 ± 0.000
LightGCN	0.075 ± 0.012	0.108 ± 0.011	0.016 ± 0.001	0.025 ± 0.001	0.041 ± 0.001	0.061 ± 0.001	0.016 ± 0.001	0.018 ± 0.000	0.006 ± 0.000	0.008 ± 0.000
RRN	0.135 ± 0.016	0.140 ± 0.005	0.036 ± 0.004	0.045 ± 0.002	0.045 ± 0.004	0.050 ± 0.008	0.014 ± 0.000	0.012 ± 0.000	0.004 ± 0.000	0.005 ± 0.000
GCMCRNN(sep)	0.134 ± 0.010	0.130 ± 0.002	0.020 ± 0.001	0.028 ± 0.002	0.052 ± 0.003	0.079 ± 0.004	0.010 ± 0.000	0.013 ± 0.001	0.005 ± 0.000	0.006 ± 0.000
GCMCRNN(inc)	0.172 ± 0.013	0.142 ± 0.011	0.019 ± 0.001	0.028 ± 0.003	0.054 ± 0.001	0.078 ± 0.005	0.009 ± 0.000	0.014 ± 0.000	0.006 ± 0.000	0.006 ± 0.000
DeepCoevolve	0.080 ± 0.004	0.078 ± 0.006	0.012 ± 0.003	0.021 ± 0.002	0.064 ± 0.005	0.091 ± 0.010	0.006 ± 0.000	0.006 ± 0.001	0.006 ± 0.000	0.005 ± 0.000
JODIE	0.100 ± 0.002	0.092 ± 0.001	0.025 ± 0.001	0.039 ± 0.001	0.067 ± 0.002	0.092 ± 0.003	0.015 ± 0.003	0.016 ± 0.002	–	–
SPMF	0.083 ± 0.001	0.092 ± 0.001	0.013 ± 0.000	0.014 ± 0.000	0.023 ± 0.001	0.021 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.003 ± 0.000	0.003 ± 0.000
IncCTR	0.091 ± 0.003	0.102 ± 0.001	0.012 ± 0.000	0.018 ± 0.000	0.021 ± 0.002	0.032 ± 0.004	0.003 ± 0.000	0.003 ± 0.000	0.002 ± 0.000	0.003 ± 0.000
SML	0.121 ± 0.003	0.123 ± 0.002	0.015 ± 0.003	0.026 ± 0.002	0.032 ± 0.002	0.050 ± 0.005	0.005 ± 0.000	0.006 ± 0.000	0.005 ± 0.000	0.006 ± 0.000
IGCN (Ours)	**0.189 ± 0.001**	**0.185 ± 0.000**	**0.039 ± 0.002**	**0.048 ± 0.002**	**0.071 ± 0.002**	**0.101 ± 0.003**	**0.021 ± 0.002**	**0.023 ± 0.001**	**0.010 ± 0.000**	**0.012 ± 0.000**

Models	MovieLens 100K		Netflix		MovieLens 1M		Amazon Books		Yelp	
	MRR@5	MRR@10	MRR@5	MRR@10	MRR@5	MRR@10	MRR@5	MRR@10	MRR@5	MRR@10
BPR	0.176 ± 0.002	0.154 ± 0.008	0.073 ± 0.002	0.071 ± 0.005	0.115 ± 0.004	0.121 ± 0.005	0.025 ± 0.001	0.026 ± 0.002	0.011 ± 0.000	0.012 ± 0.000
LambdaFM	0.160 ± 0.007	0.131 ± 0.007	0.074 ± 0.005	0.071 ± 0.004	0.121 ± 0.007	0.116 ± 0.003	0.008 ± 0.001	0.008 ± 0.001	0.004 ± 0.000	0.004 ± 0.000
IRGAN	0.226 ± 0.034	0.205 ± 0.019	0.106 ± 0.005	0.111 ± 0.003	0.218 ± 0.012	0.203 ± 0.014	0.022 ± 0.003	0.020 ± 0.004	0.009 ± 0.000	0.011 ± 0.000
LightGCN	0.261 ± 0.020	0.239 ± 0.019	0.100 ± 0.004	0.104 ± 0.003	0.187 ± 0.003	0.184 ± 0.003	0.030 ± 0.001	0.032 ± 0.001	0.019 ± 0.001	0.022 ± 0.001
RRN	0.294 ± 0.032	0.265 ± 0.026	0.153 ± 0.018	0.139 ± 0.003	0.088 ± 0.006	0.101 ± 0.022	0.027 ± 0.001	0.023 ± 0.002	0.011 ± 0.001	0.014 ± 0.000
GCMCRNN(sep)	0.307 ± 0.040	0.216 ± 0.021	0.140 ± 0.011	0.147 ± 0.009	0.216 ± 0.003	0.200 ± 0.003	0.021 ± 0.001	0.025 ± 0.001	0.016 ± 0.000	0.016 ± 0.000
GCMCRNN(inc)	0.321 ± 0.068	0.239 ± 0.053	0.145 ± 0.014	0.141 ± 0.005	0.211 ± 0.008	0.202 ± 0.003	0.020 ± 0.000	0.029 ± 0.000	0.018 ± 0.000	0.019 ± 0.000
DeepCoevolve	0.200 ± 0.017	0.164 ± 0.024	0.069 ± 0.004	0.077 ± 0.002	0.155 ± 0.022	0.169 ± 0.020	0.017 ± 0.001	0.017 ± 0.004	0.016 ± 0.001	0.013 ± 0.000
JODIE	0.256 ± 0.003	0.268 ± 0.001	0.158 ± 0.001	0.163 ± 0.003	0.177 ± 0.003	0.209 ± 0.003	0.031 ± 0.009	0.030 ± 0.002	–	–
SPMF	0.187 ± 0.005	0.193 ± 0.004	0.076 ± 0.004	0.069 ± 0.002	0.105 ± 0.002	0.067 ± 0.001	0.007 ± 0.000	0.008 ± 0.000	0.018 ± 0.001	0.014 ± 0.000
IncCTR	0.215 ± 0.002	0.216 ± 0.003	0.074 ± 0.003	0.081 ± 0.002	0.094 ± 0.009	0.100 ± 0.011	0.009 ± 0.000	0.012 ± 0.000	0.013 ± 0.000	0.013 ± 0.000
SML	0.234 ± 0.009	0.203 ± 0.004	0.097 ± 0.003	0.108 ± 0.006	0.156 ± 0.004	0.154 ± 0.009	0.014 ± 0.000	0.016 ± 0.001	0.019 ± 0.003	0.017 ± 0.001
IGCN (Ours)	**0.371 ± 0.010**	**0.320 ± 0.015**	**0.176 ± 0.008**	**0.171 ± 0.005**	**0.232 ± 0.002**	**0.229 ± 0.005**	**0.035 ± 0.001**	**0.036 ± 0.002**	**0.030 ± 0.002**	**0.035 ± 0.003**

Models	MovieLens 100K		Netflix		MovieLens 1M		Amazon Books		Yelp	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
BPR	0.203 ± 0.005	0.202 ± 0.011	0.087 ± 0.002	0.098 ± 0.005	0.130 ± 0.004	0.149 ± 0.005	0.031 ± 0.001	0.039 ± 0.002	0.014 ± 0.000	0.018 ± 0.000
LambdaFM	0.194 ± 0.006	0.184 ± 0.011	0.090 ± 0.004	0.099 ± 0.004	0.148 ± 0.009	0.160 ± 0.006	0.010 ± 0.001	0.012 ± 0.000	0.005 ± 0.000	0.008 ± 0.001
IRGAN	0.257 ± 0.028	0.277 ± 0.023	0.130 ± 0.008	0.155 ± 0.005	0.245 ± 0.007	0.256 ± 0.009	0.025 ± 0.003	0.028 ± 0.003	0.012 ± 0.000	0.016 ± 0.000
LightGCN	0.308 ± 0.026	0.342 ± 0.024	0.121 ± 0.005	0.141 ± 0.003	0.218 ± 0.003	0.233 ± 0.001	0.039 ± 0.001	0.051 ± 0.000	0.024 ± 0.001	0.034 ± 0.001
RRN	0.336 ± 0.037	0.339 ± 0.025	0.190 ± 0.019	0.202 ± 0.005	0.115 ± 0.004	0.141 ± 0.025	0.034 ± 0.001	0.034 ± 0.001	0.015 ± 0.001	0.020 ± 0.001
GCMCRNN(sep)	0.350 ± 0.031	0.302 ± 0.017	0.165 ± 0.012	0.190 ± 0.009	0.249 ± 0.003	0.259 ± 0.004	0.028 ± 0.001	0.038 ± 0.001	0.018 ± 0.000	0.019 ± 0.000
GCMCRNN(inc)	0.387 ± 0.049	0.338 ± 0.036	0.177 ± 0.014	0.195 ± 0.009	0.241 ± 0.010	0.256 ± 0.009	0.027 ± 0.000	0.044 ± 0.000	0.020 ± 0.000	0.024 ± 0.000
DeepCoevolve	0.233 ± 0.011	0.222 ± 0.019	0.089 ± 0.006	0.115 ± 0.003	0.198 ± 0.024	0.242 ± 0.024	0.020 ± 0.000	0.023 ± 0.004	0.020 ± 0.000	0.019 ± 0.001
JODIE	0.332 ± 0.002	0.344 ± 0.002	0.183 ± 0.002	0.204 ± 0.003	0.238 ± 0.002	0.247 ± 0.005	0.042 ± 0.012	0.054 ± 0.008	–	–
SPMF	0.227 ± 0.004	0.252 ± 0.003	0.089 ± 0.003	0.101 ± 0.003	0.145 ± 0.009	0.119 ± 0.004	0.009 ± 0.000	0.011 ± 0.000	0.018 ± 0.000	0.017 ± 0.001
IncCTR	0.235 ± 0.007	0.284 ± 0.011	0.096 ± 0.003	0.118 ± 0.003	0.123 ± 0.006	0.149 ± 0.004	0.012 ± 0.001	0.012 ± 0.001	0.015 ± 0.000	0.017 ± 0.000
SML	0.241 ± 0.022	0.278 ± 0.007	0.104 ± 0.004	0.131 ± 0.003	0.183 ± 0.005	0.197 ± 0.004	0.020 ± 0.000	0.024 ± 0.001	0.023 ± 0.001	0.023 ± 0.002
IGCN (Ours)	**0.433 ± 0.006**	**0.409 ± 0.013**	**0.202 ± 0.006**	**0.211 ± 0.003**	**0.265 ± 0.004**	**0.288 ± 0.003**	**0.046 ± 0.001**	**0.056 ± 0.001**	**0.037 ± 0.003**	**0.049 ± 0.004**

Table 4: Ablation study on IGCN. "Seen" or "Unseen" means users are observed or not during training, respectively.

Setting	Models	MovieLens 1M		
		F1-score@Top5	MRR@Top5	NDCG@Top5
Seen	IGCN w/o GCN	0.054 ± 0.007	0.143 ± 0.009	0.180 ± 0.006
	IGCN w/o MAML	0.057 ± 0.003	0.203 ± 0.001	0.234 ± 0.002
	IGCN w/ GRU	0.065 ± 0.000	0.225 ± 0.002	0.257 ± 0.001
	IGCN	**0.071 ± 0.002**	**0.232 ± 0.002**	**0.265 ± 0.004**
Unseen	IGCN w/o MAML	0.050 ± 0.001	0.633 ± 0.002	0.657 ± 0.003
	IGCN w/ MAML	**0.052 ± 0.001**	**0.653 ± 0.001**	**0.683 ± 0.003**

4.3 Ablation Study

4.3.1 The importance of GCN. We first conduct ablation study to verify the importance of GCN to IGCN on MovieLens 1M dataset, and the results are shown in Table 4. Due to the space limitation, we only show the top5 recommendation results, and the top10 recommendation results have the same trends and thus are omitted. From the results, we find that IGCN (with GCN) achieves the best performance, which shows that GCN is important for IGCN. GCN enables users and items to enrich their own representations from their neighborhoods. Meanwhile, with the help of GCN, the user features and item features can complement with each other, which can help to improve the recommendation accuracy. Therefore, IGCN with GCN can achieve better accuracy than IGCN w/o GCN.

4.3.2 The importance of MAML. We verify the importance of MAML to IGCN through two different settings on MovieLens 1M dataset: 1) parameter initialization (seen), in which we use e_u, e_i trained from MAML to initialize the parameters of IGCN; 2) new users (unseen), in which we sample 680 new users who don't appear in the training phase for test. For the new users, 5% of their interactions are used

Table 5: Accuracy comparison between IGCN w/ TCN and IGCN w/ iTCN.

Models	MovieLens 1M		
	F1-score@Top5	MRR@Top5	NDCG@Top5
IGCN w/ TCN	0.070 ± 0.003	0.226 ± 0.005	0.259 ± 0.007
IGCN w/ iTCN	**0.071 ± 0.002**	**0.232 ± 0.002**	**0.265 ± 0.004**

to fine-tune users' historical embedding and the rest interactions are used to verify the effectiveness of MAML. The first setting can verify whether MAML can help to achieve better representation learning on seen users while the second setting can verify whether MAML can address the new user issue in test time.

The results are summarized in table 4, in which we only present the results for top 5 recommendation due to space limitation and the top 10 recommendation results have the same trends. Note that IGCN w/o MAML means IGCN use random initialization instead of using MAML. We have the following observations from the results: 1) the initialization from MAML is significantly better than random initialization; 2) the pretrained user embedding from MAML can indeed help to address the new users/items issue in test time. Overall, the two experiments confirm that meta learning can help to address both the parameter initialization issue and the new user/item issue in recommendation tasks.

It should be noted that the comparison should only be done within the same setting. It is reasonable for the MRR and NDCG in the "Unseen" setting to be higher than those in the "Seen" setting, because the rated items of each user in the "Unseen" setting is more than those in the "Seen" setting, resulting in higher chance of being ranked high. However, F1-score is not high due to small recalls.

4.3.3 The importance of iTCN. In this experiment, we verify the effectiveness of iTCN on the MovieLens 1M dataset. We replace iTCN to GRU in IGCN and keep IGCN w/ GRU incremental. By comparing IGCN w/ GRU and IGCN w/ iTCN, we can verify the effectiveness of iTCN. From the results shown in Table 4, we can see that IGCN w/ iTCN has better prediction accuracy than IGCN w/ GRU, which shows that the proposed iTCN can achieve better performance than RNN-based method in our incremental recommendation task. By reasonably setting the values of kernel size and dilation, iTCN can remember more information than GRU and ensure high accuracy by temporal-aware feature learning.

4.4 iTCN vs. TCN

To verify if iTCN is superior to TCN, we first compare the accuracy of IGCN using iTCN and TCN, respectively. As shown in Table 5, IGCN with iTCN achieves higher accuracy than IGCN with TCN in all cases. The main reason is that iTCN only adopts one convolution layer to extract feature from the raw input but TCN adopts hierarchical convolution layers so that part of the raw information may be lost after multiple layers. In addition, TCN requires zero padding at each convolution layer, which may also introduce information loss, but iTCN requires only one padding. In summary, compared to TCN, iTCN can avoid the information loss caused by excessive padding and stacking multiple convolutional layers.

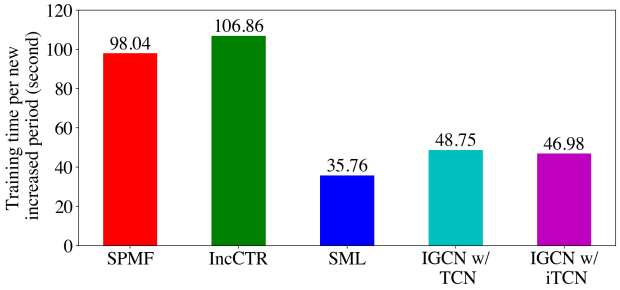

Figure 3: Efficiency comparison between IGCN and other incremental CF algorithms. Here, IGCN w/ TCN means that we replace the iTCN from IGCN with TCN.

We also compare the efficiency between iTCN and TCN. Note that we implement TCN in an incremental form in this experiment, but keep the original TCN structure. As shown in Figure 3, iTCN is faster than TCN due to simplified structure. Although the overall efficiency improvement seems small (~3.6%), the time reported here actually contains the time for training all components. If we only compare the computation of iTCN and TCN, the improvement of iTCN is as high as 53.4%. Due to higher accuracy and efficiency, we believe iTCN is superior to TCN in our task.

4.5 Efficiency of Incremental Training

To compare the efficiency of IGCN against other incremental CF algorithms, we conduct the efficiency analysis experiment on the MovieLens 1M dataset, and show the training time of IGCN and the other incremental algorithms for model updating in each new time period in Figure 3. As we can see from the results, SML achieves the highest efficiency. The training time of IGCN is comparable with that of SML, which is over 2X more efficient than IncCTR and SPMF. In addition, IGCN can achieve much higher recommendation accuracy compared with the other incremental CF algorithms as shown in Table 3. Thus, we claim that IGCN is more desirable than the other state-of-the-art incremental CF algorithms due to competitive efficiency and higher accuracy.

5 CONCLUSION

Incremental training of GNN models are challenging in collaborative filtering and existing retraining or RNN-based methods cannot optimally address this issue. This paper proposes the incremental graph convolutional network, which leverages both GCN and incremental temporal convolutional networks to perform incremental GNN training with high accuracy and efficiency. MAML is adopted to initialize the user/item embedding to speedup model adaptation and alleviate cold-start issue. Experiments on five real-world datasets show that IGCN can outperform state-of-the-art CF methods in sequential recommendation tasks. In addition, IGCN exhibits competitive efficiency compared with incremental CF methods.

ACKNOWLEDGMENTS

The research was supported by National Natural Science Foundation of China (NSFC) under the Grant No. 61932007.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271* (2018).

[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[4] Chao Chen, Dongsheng Li, Junchi Yan, Hanchi Huang, and Xiaokang Yang. 2021. Scalable and Explainable 1-Bit Matrix Completion via Graph Signal Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7011–7019.

[5] Chao Chen, Dongsheng Li, Junchi Yan, and Xiaokang Yang. 2021. Modeling Dynamic User Preference via Dictionary Learning for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 7–10.

[7] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[8] Debashis Das, Laxman Sahoo, and Sujoy Datta. 2017. A survey on recommendation system. *International Journal of Computer Applications* 160, 7 (2017).

[9] Samuel G Fadel and Ricardo da S Torres. 2018. Link Prediction in Dynamic Graphs for Recommendation. *arXiv preprint arXiv:1811.07174* (2018).

[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 417–426.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. (2017), 1126–1135.

[12] Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3, 4 (1999), 128–135.

[13] Albert Gu, Caglar Gulcehre, Thomas Paine, Matt Hoffman, and Razvan Pascanu. 2020. Improving the gating mechanism of recurrent neural networks. In *International Conference on Machine Learning*. PMLR, 3800–3809.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 639–648.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182.

[16] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, California, USA) *(SIGIR '99)*. Association for Computing Machinery, New York, NY, USA, 230–237.

[17] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for Top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chicago, Illinois, USA) *(KDD '13)*. Association for Computing Machinery, New York, NY, USA, 659–667.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1269–1278.

[20] Dongsheng Li, Chao Chen, Tun Lu, Stephen M. Chu, and Ning Gu. 2021. Mixture Matrix Approximation for Collaborative Filtering. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2021), 2640–2653.

[21] Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen M. Chu. 2016. Low-Rank Matrix Approximation with Stability. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (New York, NY, USA) *(ICML'16)*. JMLR.org, 295–303.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*. 452–461.

[23] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 240–248.

[24] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. Association for Computing Machinery, New York, NY, USA, 285–295.

[26] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112.

[27] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. PMLR, 1139–1147.

[28] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 565–573.

[29] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3307–3313.

[30] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 515–524.

[31] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 525–534.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 165–174.

[33] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A Practical Incremental Method to Train Deep CTR Models. *arXiv preprint arXiv:2009.02147* (2020).

[34] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 495–503.

[35] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2091–2102.

[36] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 974–983.

[37] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. LambdaFM: Learning Optimal Ranking with Factorization Machines Using Lambda Surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. ACM, New York, NY, USA, 227–236.

[38] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1479–1488.

[39] Zhenghao Zhang, Tun Lu, Dongsheng Li, Peng Zhang, Hansu Gu, and Ning Gu. 2021. SANS: Setwise Attentional Neural Similarity Method for Few-Shot Recommendation.. In *DASFAA (3)*. 69–84.

[40] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. ACM, New York, NY, USA, 2347–2357.