Causal Incremental Graph Convolution for Recommender System Retraining

Sihao Ding, Fuli Feng, Xiangnan He, Yong Liao, Jun Shi, and Yongdong Zhang, *Senior Member, IEEE*

***Abstract*— The real-world recommender system needs to be regularly retrained to keep with the new data. In this work, we consider how to efficiently retrain graph convolution network (GCN)-based recommender models that are state-of-the-art techniques for the collaborative recommendation. To pursue high efficiency, we set the target as using only new data for model updating, meanwhile not sacrificing the recommendation accuracy compared with full model retraining. This is nontrivial to achieve since the interaction data participates in both the graph structure for model construction and the loss function for model learning, whereas the old graph structure is not allowed to use in model updating. Toward the goal, we propose a *causal incremental graph convolution* (IGC) approach, which consists of two new operators named IGC and *colliding effect distillation* (CED) to estimate the output of full graph convolution. In particular, we devise simple and effective modules for IGC to ingeniously combine the old representations and the incremental graph and effectively fuse the long- and short-term preference signals. CED aims to avoid the out-of-date issue of inactive nodes that are not in the incremental graph, which connects the new data with inactive nodes through causal inference. In particular, CED estimates the causal effect of new data on the representation of inactive nodes through the control of their collider. Extensive experiments on three real-world datasets demonstrate both accuracy gains and significant speed-ups over the existing retraining mechanism.**

***Index Terms*— Casual inference, colliding effect (CE), graph neural network, incremental training, recommender system.**

I. Introduction

RECENT years have witnessed the success of graph convolution network (GCN)-based recommender models, such as PinSAGE [1] and LightGCN [2], which performs node representation learning over the interaction graph and demonstrates promising performance [3]. The core of them is *neighborhood aggregation* that enhances a node's representation with the information from its neighbors. In this way, the graph structure can be explicitly integrated into the embedding space, improving the representations of users and items. In practical usage, a recommender system needs to be periodically (e.g., daily) retrained to keep the model fresh with the new interaction data. In this work, we study the problem of GCN model retraining for the recommendation, which has received relatively little scrutiny.

Given new interactions for refreshing an old GCN model, there are three straightforward strategies.

1) *Full retraining*, which simply merges the old data and new interactions to perform a full model training. This solution retains the most fidelity since all data are used. However, it is very costly in both memory and time since interaction data are usually of a large scale and keep increasing with time.
2) *Fine-Tuning With Old Graph:* Interaction data participate in two parts of a GCN: forming the graph structure to perform graph convolutions and constituting training examples of the loss function. This fine-tuning solution constructs training examples with new interactions only while still using the full graph structure. As such, although this solution costs fewer resources than full retraining, it is still costly due to the usage of the old graph.
3) *Fine-Tuning w/o Old Graph:* This solution uses only the new interactions for model training and graph convolution. The old graph is not used in graph convolution, which saves many computation and storage resources because of the high sparsity of incremental graph (see Fig. 1). However, the new interactions contain only users' short-term preferences, which can differ much from the long-term performances and be much sparser. In addition, it cuts off the connection to the inactive nodes that have no new interactions. It, thus, suffers easily from forgetting and overfitting issues.

Given the pros and cons of the above intuitive strategies, we distill three considerations for effective and efficient GCN recommender retraining: 1) detaching the old graph; 2) reserving the old (long-term) preference signal; and 3) fusing the old and new preference signals. In short, our target is to achieve comparable or even better recommendation accuracy as full retraining with the use of new interactions only. The key lies in how to estimate the output of *full graph convolution* (i.e., on the whole graph for full retraining) based on the *old node representations* and the *incremental graph* (i.e., the graph constructed from the new interactions (cf. Fig. 1) as

Manuscript received August 4, 2021; revised December 23, 2021; accepted February 24, 2022. This work was supported by the National Natural Science Foundation of China under Grant U19A2079, Grant U21B2026, and Grant 62121002. *(Corresponding author: Fuli Feng.)*

Sihao Ding, Fuli Feng, and Yong Liao are with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: dsihao@mail.ustc.edu.cn; fulifeng93@gmail.com; ly@ustc.edu.cn).

Xiangnan He and Yongdong Zhang are with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: hexn@ustc.edu.cn; zhyd73@ustc.edu.cn).

Jun Shi is with the Innovation Laboratory, CETC, Beijing 100043, China (e-mail: junshi@cyberaray.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3156066.

Digital Object Identifier 10.1109/TNNLS.2022.3156066

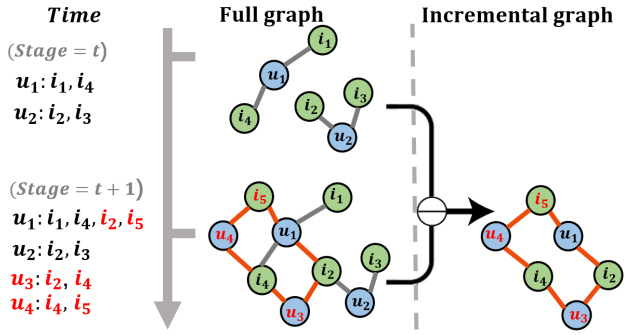


Fig. 1. Illustration of the incremental graph at stage $t+1$ in GCN retraining.

an example). This is, however, nontrivial to achieve for three reasons.

1) In the full graph convolution, the new interactions not only bring new neighbors for a target node but also participate in the normalization weighting of old neighbors. It is known that the normalization weights of neighbors have a large impact on the GCN performance [2] and need to be carefully considered.
2) The old representations are learned over historical data and represent long-term preference. Since user interests may drift, making the new interactions discrepant from long-term preference, blindly fine-tuning old representations could make the model forget the long-term preference.
3) The incremental graph lacks inactive nodes that have no new interactions (e.g., node i_1 in Fig. 1), which calls for extra effort to refresh the representation of such nodes.

Toward our target, we first propose *incremental graph convolution* (IGC), which estimates the full graph convolution of a target node based on its old representation and the incremental graph structure. In particular, we design a *degree synchronizer* to learn the degree-based node normalization weights, so as to approach the normalization weights in full graph convolution. Meanwhile, we devise a *representation aggregator* to compose the old representation with the new neighbors for effectively encoding both long- and short-term signals. We devise the two modules as simple convolutional neural networks (CNNs), which keep the overall complexity of IGC close to fine-tuning w/o old graph with sufficient fidelity.

Meanwhile, we devise *colliding effect distillation* (CED) to refresh the representation of inactive nodes, which estimates the effect of new data on the representation of inactive nodes. In particular, we resort to causal theory [4] and frame the whole incremental training phase as a causal graph (see Fig. 3). The key lies in how to connect the d-separated [4] variables: the representation of inactive nodes and new data. We construct a collider [4] between them to make them conditionally d-connected and estimate their colliding effect (CE) [5] to update the representation of inactive nodes. In particular, CED distills the CE conditioned on the similarity between inactive and active nodes implied by the old representations.

The proposed IGC and CED are universal operators that are applicable to most GCN models. In this work, we equip them on LightGCN [2], a simple GCN model with state-of-the-art performance on the collaborative recommendation. Through extensive experiments on three real-world datasets, we demonstrate the effectiveness of the Causal IGC approach, which outperforms full retraining in recommendation accuracy with overall costs comparable to fine-tuning w/o old graph. The main contributions of this work are summarized as follows.

1) We study the new task of GCN model retraining for recommendation and propose a new *Causal IGC* approach that well supports the efficient and effective retraining of GCN models.
2) We devise two universal operators named IGC and CED that can estimate the full graph convolution for both active and inactive nodes during the incremental training.
3) We instantiate IGC and CED on LightGCN and conduct extensive experiments to demonstrate the effectiveness and efficiency of our approach.

II. Methodology

We represent the temporal interaction data as $\mathcal{I} = \{I_0, I_1, \ldots, I_{t-1}, I_t\}$ and the corresponding user–item graph as $\mathcal{G} = \{G_0, G_1, \ldots, G_{t-1}, G_t\}$. I_t means the new interactions collected in stage t; G_t is the bipartite user–item graph built on the interactions in I_t, which we also term the *incremental graph*. A stage can be of any time period (e.g., one/multiple hours/days), which depends on the expected model freshness and the training cost that we can afford. Toward the target of efficient GCN model retraining, we formulate the task as

$$\boldsymbol{\theta}_{t-1} \xrightarrow[\text{retrain}]{(I_t, G_t)} \boldsymbol{\theta}_t \xrightarrow[\text{serve}]{} I_{t+1} \tag{1}$$

where $\boldsymbol{\theta}_t$ denotes the model parameters learned in stage t.

In this formulation, we retrain using only the latest stage's interactions I_t and the incremental graph G_t.[1] The model with $\boldsymbol{\theta}_t$ is used to serve for next stage $t+1$; thus, we use the (future) data I_{t+1} to evaluate the retraining effectiveness.

A. GCN-Based Recommender Model

We recap how GCN works for collaborative filtering. Suppose that we construct GCN on the *full graph* up to t, denoted as $G_{0\sim t}$; the graph convolution commonly used is

$$\boldsymbol{e}_{i,t}^{(l+1)} = \sigma\left(\frac{1}{\sqrt{d_{i,0\sim t}}} \overbrace{\sum_{j\in\mathcal{N}_{i,0\sim t}} \frac{1}{\sqrt{d_{j,0\sim t}}} \boldsymbol{e}_{j,t}^{(l)}}^{\text{neighborhood aggregation}}\right) \tag{2}$$

where $\boldsymbol{e}_{j,t}^{(l)}$ denotes the representation of node j at the lth layer, $\mathcal{N}_{i,0\sim t}$ represents the neighbors of node i in the graph $G_{0\sim t}$, and $d_{i,0\sim t}$ (accumulated degree) equals to the number of nodes in $\mathcal{N}_{i,0\sim t}$. The core of graph convolution is *neighborhood aggregation*, which aggregates the representations of neighbor nodes for the target node i. The node degrees play the

[1] Real-world recommender systems will continually collect new data, so the whole training data and the full graph will keep growing. Thus, the existing gap in retraining costs between *fine-tuning with old graph* (or *full-retraining*) and *incremental training* will continue to widen.

role of normalization, exerting a large impact on the GCN performance [2]. The feature transformation function $\sigma(\cdot)$ has various formats, such as linear [3] and bilinear mapping [6]. We focus on the neighborhood aggregation in this work, omitting the feature transformation function $\sigma(\cdot)$ for briefness.

The embeddings of the zeroth layer are the model parameters to learn, which are trained by minimizing the loss function

$$\sum_{(u,i)\in\mathcal{I}/\mathcal{I}^-} L\left(y_{u,i}, \hat{y}_{u,i}\right) + \lambda\|\boldsymbol{\theta}\|^2 \tag{3}$$

where I^- are negative samples, $y_{u,i}$ is the interaction label, and $\hat{y}_{u,i}$ is the corresponding model prediction, such as the inner product of final user and item representations. $L(\cdot)$ specifies the recommendation loss, such as the pairwise BPR [7] and pointwise cross entropy [8], and λ is the hyperparameter for L_2 regularization.

B. Incremental Graph Convolution

The cost of full retraining is high and increasing with time. Since $\mathcal{N}_{i,0\sim t}$ equals to $\mathcal{N}_{i,0\sim t-1} \cup \mathcal{N}_{i,t}$, an efficient retraining means bypassing $\mathcal{N}_{i,0\sim t-1}$ and using only $\mathcal{N}_{i,t}$ to estimate the full graph convolution of (2). Suppose that the old representation $\boldsymbol{e}_{i,t-1}^{(l+1)}$ is trained on $G_{0\sim t-1}$; it then can encode graph convolution experience on $\mathcal{N}_{i,0\sim t-1}$.[2] As such, a smart integration of it with only new neighbors can well approximate the result of full graph convolution. To this end, we propose IGC (see Fig. 2) that estimates the full graph convolution on target node i as

$$\boldsymbol{e}_{i,t}^{(l+1)} = \frac{1}{\sqrt{d'_{i,t}}} \cdot \varphi \times \left(\overbrace{\sqrt{d_{i,0\sim t-1}} \cdot \boldsymbol{e}_{i,t-1}^{(l+1)}}^{\text{old representation, } \textbf{constant}}, \overbrace{\sum_{j\in\mathcal{N}_{i,t}} \frac{1}{\sqrt{d'_{j,t}}} \cdot \boldsymbol{e}_{j,t}^{(l)}}^{\text{new representations, } \textbf{to learn}} \right) \tag{4}$$

where $d'_{j,t} = f(d_{j,0\sim t-1}, d_{j,t})$ denotes the normalization weight estimated by the *degree synchronizer*. In the next, we elaborate the core designs of our IGC.

1) Degree Synchronizer: To retain the most fidelity as the full graph convolution, we carefully set the degree-based normalization weights in IGC. We first scrutinize the first term of old representation

$$\sqrt{d_{i,0\sim t-1}} \cdot \boldsymbol{e}_{i,t-1}^{(l+1)} = \sum_{j\in\mathcal{N}_{i,0\sim t-1}} \frac{1}{\sqrt{d_{j,0\sim t-1}}} \boldsymbol{e}_{j,t-1}^{(l)}. \tag{5}$$

If we replace $t-1$ with t, the right side becomes exactly what we need for full neighborhood aggregation. Since we are not allowed to use $\mathcal{N}_{i,0\sim t-1}$ in IGC, we use $(d_{i,0\sim t-1})^{1/2} \cdot \boldsymbol{e}_{i,t-1}^{(l+1)}$ instead, which has encoded the signal in $\mathcal{N}_{i,0\sim t-1}$. As updating $\boldsymbol{e}_{i,t-1}^{(l+1)}$ in retraining has the risk of losing the signal of old neighbors, we set it as a constant during stage t retraining. For this operation, we only need to store one additional integer for each node—its accumulated degree $d_{i,0\sim t-1}$—the cost of which is negligible.

[2]Iteratively training with our IGC from stage 0 to $t-1$ yields the same effect, so the premise still holds.

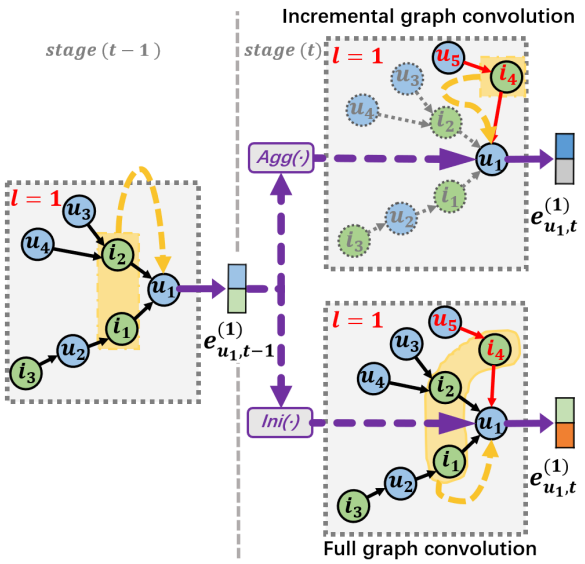


Fig. 2. IGC on the target node u_1 in stage t. The blurred part is the old graph that is not used in stage t. Red edges denote the new interactions in stage t. The yellow region shows the nodes included in graph convolution. Agg(·) and Ini(·) represent the operations to use $\boldsymbol{e}_{u_1,t-1}^{(1)}$ in *IGC* and *full graph convolution*, respectively.

We next move to the second term for new neighbors modeling. At the first sight, we can set $f(d_{j,0\sim t-1}, d_{j,t})$ as $d_{j,0\sim t-1} + d_{j,t}$, which is the one used in full graph convolution. However, the evolution pattern of node degree may be different for datasets of different domains. This provides opportunities to boost the performance of original graph convolution if we can strategically learn $f(\cdot)$ toward the recommendation objective function. To this end, we weighted combine the two degrees

$$f(d_{j,0\sim t-1}, d_{j,t}) = \beta \cdot d_{j,0\sim t-1} + d_{j,t} \tag{6}$$

where β is end-to-end trained to control the impact of old degree. We empirically find this simple function works well, and the optimal β varies significantly for different datasets (cf. Section III-C).

2) Representation Aggregator $\varphi(\cdot)$: The full graph convolution updates parameters associated with both old and new neighbors. However, IGC is only allowed to update parameters of new neighbors for efficiency concerns. This causes significant discrepancy issues: 1) the old representation is off-the-shelf, while the new representation is iteratively updated to optimize the loss function on new interactions and 2) the old representation is learned over historical data and represents long-term preference, which may be discrepant from the new interactions due to interest drift.

To tackle these issues, we devise a parameterized function $\varphi(\cdot)$, adapting the fusion strategy to the training objective. An existing solution is the Transfer component in [9], which adopts a standard CNN network with nonlinear transformations. However, we find it works poorly in our scenario, partially because of the complexity brought by feature mapping and nonlinear activation. As such, we further simplify the

CNN design, retaining the minimum operations and parameters. Specifically, we stack the old and new representations, passing them into a convolution layer

$$p\left(\left\langle \boldsymbol{w}_f^{(l)}, \left[\sqrt{d_{i,0\sim t-1}}\cdot \boldsymbol{e}_{i,t-1}^{(l+1)}, \sum_{j\in\mathcal{N}_{i,t}} \frac{1}{\sqrt{d'_{j,t}}}\cdot \boldsymbol{e}_{j,t}^{(l)}\right]\right\rangle\right) \quad (7)$$

where $p(\cdot)$ is a pooling operation, $\boldsymbol{w}_f \in \mathbb{R}^{2\times 1}$ denotes the fth filter of the CNN layer, and there can be multiple filters in a layer. In this way, the aggregator adjusts the importance of long- and short-term preferences and aligns the scale of the two representations in the training process. We can stack multiple CNN layers to enhance the expressiveness of the aggregator, whereas, in our experiments, using one layer leads to good performance in most cases (cf. Section III-D).

To summarize, compared with the vanilla graph convolution, our IGC demonstrates three main differences.

1) *Incremental Neighborhood Aggregation:* Instead of aggregating all neighbors, IGC aggregates the new neighbors only, combining with the target node's old representation $\boldsymbol{e}_{i,t-1}^{(l+1)}$. Only the parameters associated with new neighbors are trainable (i.e., the zeroth layer embeddings), which significantly reduces the retraining cost.
2) *Degree Synchronizer:* This module revises the normalization weights for old representation and new neighbors to boost the model performance. It requires saving the accumulated degree for each node, which has a negligible cost equal to increasing the embedding size by 1.
3) *Representation Aggregator $\varphi(\cdot)$:* This learnable function combines the constant $\boldsymbol{e}_{i,t-1}^{(l+1)}$ and the trained new representations with the ability to handle the discrepancy between them. It can also compensate for some estimation error of the old representation and improve model capability by only introducing a simple CNN with very few parameters.

C. Colliding Effect Distillation

IGC is a universal operator that is applicable to most GCN models to calculate user and item representations. We apply IGC to a state-of-the-art collaborative filtering GCN model LightGCN [2] as *I-LightGCN*, where we stack L IGC layers and set the final representations of all nodes as the average of their representations at all layers. Formally,

$$\boldsymbol{r}_{i,t} = \frac{1}{L}\sum_{l=0}^{L} \boldsymbol{e}_{i,t}^{(l)}. \quad (8)$$

We denote the nodes occur in I_t as active nodes and the remaining as inactive nodes. Accordingly, we denote the representations of active and inactive nodes as $\boldsymbol{R}_{\mathrm{ac},t}$ and $\boldsymbol{R}_{\mathrm{In},t}$, respectively. Their old representations are organized into $\boldsymbol{R}_{\mathrm{ac},t-1}$ and $\boldsymbol{R}_{\mathrm{In},t-1}$ with the same criterion.

As we use the new data I_t to train I-LightGCN, IGC mainly refreshes the representation of active nodes, which will, thus, face the out-of-date issue on the inactive nodes. The reason is that the parameters correspond to the inactive nodes (e.g., node embedding) are not involved in the training procedure (e.g., backpropagation). In this light, we consider two directions to properly refresh the representation of inactive nodes: 1) directly injecting the new preference signal from active nodes into the representation of inactive nodes and 2) including the parameters of inactive nodes into the training objective to indirectly push their representations to be updated. Apparently, the key lies in constructing the connection from new data to the representation of the inactive node, which is cut off due to the discarding of data replay in the incremental training setting.

1) Direct Update: Toward the target, we devise CED as

$$\tilde{\boldsymbol{r}}_{m,t} = \gamma_1 \cdot \boldsymbol{r}_{m,t} + \frac{1-\gamma_1}{K}\cdot \sum_{n\in \mathrm{KNN}(\boldsymbol{R}_{\mathrm{ac},t-1}, m, K, \delta)} \boldsymbol{r}_{n,t} \quad (9)$$

where m is an inactive node in stage t and $\tilde{\boldsymbol{r}}_{m,t}$ is the final representation that connects K active nodes. $\mathrm{KNN}(\cdot)$ denotes the nearest neighbor fetching operation that calculates the top-K nearest active nodes to node m according to a distance measure δ (e.g., the Euclidean distance) between the old representations $\boldsymbol{r}_{m,t-1}$ and $\boldsymbol{R}_{\mathrm{ac},t-1}$. $\gamma_1 \in [0, 1]$ is a hyperparameter that controls the influence of active nodes. Intuitively, CED refreshes the representation of an inactive node with the latest status of active nodes that have shown similar properties in previous stages. For instance, we believe that the inactive user m will exhibit similar short-term preference evolution as her/his similar users shown in stage t. We will give a rigorous derivation for the formulation of CED in Section II-D.

2) Indirect Update: While (9) updates the representations of inactive nodes, their parameters are still not touched since we construct the training data from I_t only. CED, thus, also operates the active nodes, which is formulated as

$$\tilde{\boldsymbol{r}}_{n,t} = \gamma_2 \cdot \boldsymbol{r}_{n,t} + \frac{1-\gamma_2}{K}\cdot \sum_{m\in \mathrm{KNN}(\boldsymbol{R}_{\mathrm{In},t-1}, n, K, \delta)} \boldsymbol{r}_{m,t}. \quad (10)$$

In this way, the parameters of inactive nodes are attached to the objective function and updated during the retraining.

Note that CED is also a universal operator applicable to most GCN models. In this work, we apply both IGC and CED to LightGCN, which is named *CI-LightGCN*. Upon the output of CED, we use the inner product of $\tilde{\boldsymbol{r}}_{u,t}$ and $\tilde{\boldsymbol{r}}_{i,t}$ to generate the prediction result of one u, i pair

$$\hat{y}_{u,i} = \langle \tilde{\boldsymbol{r}}_{u,t} \cdot \tilde{\boldsymbol{r}}_{i,t} \rangle. \quad (11)$$

Following the original LightGCN paper, we learn the model parameters $\boldsymbol{\theta}_t$ by optimizing (3) over the new data I_t with the minibatch Adam [10] optimizer. In particular, the parameters to be learned include the embedding of nodes and the CNN filters, and the β in IGC. Note that it is initialized as $\boldsymbol{\theta}_{t-1}$ in the beginning of stage t retraining (random initialization for $t = 0$). The time complexity is close to the standard fine-tuning since only the new interactions I_t are used to construct graph and training examples. Below illustrates the retraining procedure of CI-LightGCN.

In Algorithm 1, line 6 and 10 refer to *IGC*, i.e., the incremental training of the active nodes. Line 3, 7, 11, and 12 refer to *CED*, where line 7 and 12 correspond to **indirect update** and **direct update** of inactive nodes, respectively.

Algorithm 1 Retraining CI-LightGCN

Input: Old parameters $\boldsymbol{\theta}_{t-1}$, accumulated degree $\{d_{i,0\sim t-1}\}$, new interactions I_t
Output: New parameters $\boldsymbol{\theta}_t$, recommender result $\boldsymbol{Y}_t$

$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1}$ ▷ Initialization;
Generate incremental graph G_t from I_t;
Calculate KNN($\boldsymbol{R}_{In,t-1}, n, K, \delta$) of active nodes;
while *Stop condition is not reached* **do**
  Fetch mini-batch data from I_t;
  Feed forward active node embeddings by Eq. (7) and get active node representations by Eq. (8);
  Generate new active node representations with inactive nodes by Eq. (10) for indirect update;
  Update $\boldsymbol{\theta}_t$ by minimizing Eq. (3) with Eq. (11);
end
Calculate representations of all nodes based on optimized $\boldsymbol{\theta}_t$ by Eq. (7) and Eq. (8);
Calculate KNN($\boldsymbol{R}_{ac,t-1}, m, K, \delta$) of inactive nodes;
Generate final inactive node representations by Eq. (9);

D. Casual View of CED

We first conceptually introduce CE [4] with a real-life example. Suppose that a college gives scholarships to two types of students with unusual musical talents or high GPAs. This corresponds to a causal graph: $M \rightarrow S \leftarrow G$ where M, S, and G denote musical talents, scholarship, and GPA; S is a collider between M and G. Ordinarily, musical talent and GPA are independent but become dependent given that a student won a scholarship. In such a case, knowing that the student lacks musical talent, we can infer that the student is likely to have a high GPA. In other words, M and G become dependent by conditioning on the value of their common cause, i.e., the collider S. In this light, we can leverage such CE to enhance the prediction of M or G.

The design of CED is inspired by Hu *et al.* [5], which first models and leverages the CE in class incremental learning. We use three causal graphs [11] (see Fig. 3) to elaborate on the causal theory behind CED. We model the representation calculation of I-LightGCN [see Fig. 3(a)] and then introduce the collider to make inactive nodes and new data conditionally d-connected [see Fig. 3(b)], followed by the causal graph of CI-LightGCN [see Fig. 3(c)]. The nodes in a causal graph denote variables, and we scrutinize the meaning of each variable as follows.

1) $\boldsymbol{R}_{\text{In},t}$ and $\boldsymbol{R}_{\text{ac},t}$ denote the node representations of inactive nodes and active nodes, respectively.
2) $\boldsymbol{R}_{\text{In},t-1}$ and $\boldsymbol{R}_{\text{ac},t-1}$ denote the corresponding old node representations in stage $t-1$.
3) $\boldsymbol{I}_t$ is the new interaction data that is collected in stage t.
4) $\boldsymbol{S}_t$ denotes the pairwise distance between nodes in $\boldsymbol{R}_{\text{In},t}$ and $\boldsymbol{R}_{\text{ac},t}$.

The edges in the causal graph describe the causal relations between variables, where black arrows correspond to the operations in I-LightGCN, red dotted arrows correspond to the calculation of node distance, and double arrows denote the causal relations as conditioned on a collider. In particular,

1) $(\boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t) \rightarrow \boldsymbol{R}_{\text{ac},t}$: As to the active nodes, IGC aggregates the old representations $\boldsymbol{R}_{\text{ac},t-1}$ and the incremental graph constructed from $\boldsymbol{I}_t$.
2) $\boldsymbol{R}_{\text{In},t-1} \rightarrow \boldsymbol{R}_{\text{In},t}$: As to inactive nodes, IGC only encodes their old representations $\boldsymbol{R}_{\text{In},t-1}$.
3) $(\boldsymbol{R}_{\text{ac},t}, \boldsymbol{R}_{\text{In},t}) \rightarrow \boldsymbol{S}_t$: The calculation of $\boldsymbol{S}_t$ is based on the new representations of active and inactive nodes. Note that $\boldsymbol{S}_t$ is a collider between variable pairs $\boldsymbol{R}_{\text{ac},t}$ and $\boldsymbol{R}_{\text{In},t}$.
4) $\boldsymbol{R}_{\text{ac},t} \leftrightarrow \boldsymbol{R}_{\text{In},t}$: When conditioned on the collider as $\boldsymbol{S}_t = s_{t-1}$, the causal path between its parent nodes is built up [5].

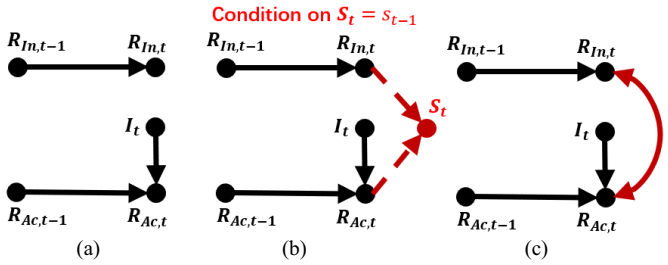


Fig. 3. (a) Causal graph of I-LightGCN to generate node representations in stage t. (b) adds an auxiliary variable, the distance $\boldsymbol{S}_t$ between active and inactive nodes. (c) Causal graph of CI-LightGCN conditioned on $\boldsymbol{S}_t = s_{t-1}$.

Recall that the key to refreshing the representation of inactive nodes is connecting $\boldsymbol{R}_{\text{In},t}$ to $\boldsymbol{I}_t$. As shown in Fig. 3(c), $\boldsymbol{I}_t$ has CE on $\boldsymbol{R}_{\text{In},t}$ through the path $(\boldsymbol{I}_t, \boldsymbol{R}_{\text{ac},t-1}) \rightarrow \boldsymbol{R}_{\text{ac},t} \leftrightarrow \boldsymbol{R}_{\text{In},t}$ as conditioned on $\boldsymbol{S}_t = s_{t-1}$.

s_{t-1} represents the old similarity between active and inactive nodes calculated from $\boldsymbol{R}_{\text{ac},t-1}$ and $\boldsymbol{R}_{\text{In},t-1}$. The condition on $\boldsymbol{S}_t = s_{t-1}$ means that the updated representations in stage t maintain a similar relative distance between nodes as the previous stage. In this light, we additionally consider the CE $\text{CE}_{\boldsymbol{I}_t, \boldsymbol{R}_{\text{ac},t-1}}$, which equals to

$$P\left(\boldsymbol{R}_{\text{In},t} | \boldsymbol{R}_{\text{In},t-1}, \boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t, \boldsymbol{S}_t = s_{t-1}\right) - P\left(\boldsymbol{R}_{\text{In},t} | \boldsymbol{R}_{\text{In},t-1}, \boldsymbol{R}_{\text{ac},t-1} = \boldsymbol{0}, \boldsymbol{I}_t = \emptyset, \boldsymbol{S}_t = s_{t-1}\right) \quad (12)$$

which denotes the change of $\boldsymbol{R}_{\text{In},t}$ as $\boldsymbol{R}_{\text{ac},t-1}$ and $\boldsymbol{I}_t$ changes from a reference status ($\boldsymbol{R}_{\text{ac},t-1} = \boldsymbol{0}$ and $\boldsymbol{I}_t = \emptyset$) to the factual status. We omit the second term since it can be treated as constant. This is because $\boldsymbol{R}_{\text{In},t}$ will not be updated if the new data are empty. By extending the first term according to the total probability formula, we derive $\text{CE}_{\boldsymbol{I}_t, \boldsymbol{R}_{\text{ac},t-1}}$ as

$$\begin{aligned}
&\sum_{\boldsymbol{R}_{\text{ac},t}} P\left(\boldsymbol{R}_{\text{In},t} | \boldsymbol{R}_{\text{In},t-1}, \boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t, \boldsymbol{S}_t = s_{t-1}, \boldsymbol{R}_{\text{ac},t}\right) \\
&\qquad \times P\left(\boldsymbol{R}_{\text{ac},t} | \boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t\right) \\
&= \sum_{\boldsymbol{R}_{\text{ac},t}} \underline{P\left(\boldsymbol{R}_{\text{In},t} | \boldsymbol{I}_t, \boldsymbol{S}_t = s_{t-1}, \boldsymbol{R}_{\text{ac},t}\right)} P\left(\boldsymbol{R}_{\text{ac},t} | \boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t\right) \\
&= \sum_{\boldsymbol{R}_{\text{ac},t}} W\left(\boldsymbol{I}_t, \boldsymbol{R}_{\text{In},t}, s_{t-1}, \boldsymbol{R}_{\text{ac},t}\right) P\left(\boldsymbol{R}_{\text{ac},t} | \boldsymbol{R}_{\text{ac},t-1}, \boldsymbol{I}_t\right). \quad (13)
\end{aligned}$$

Note that we can omit the variables $\boldsymbol{R}_{\text{In},t-1}$, $\boldsymbol{R}_{\text{ac},t-1}$ for briefness (the second step) when conditioned on $\boldsymbol{S}_t = s_{t-1}$, which

is calculated by $\boldsymbol{R}_{\mathrm{ac},t-1}$ and $\boldsymbol{R}_{\mathrm{In},t-1}$. By abstracting the underlined term as a weighting function $W(\boldsymbol{I}_t, \boldsymbol{R}_{\mathrm{In},t}, s_{t-1}, \boldsymbol{R}_{\mathrm{ac},t})$, we can understand $\mathrm{CE}_{\boldsymbol{I}_t, \boldsymbol{R}_{\mathrm{ac},t-1}}$ as a weighted adjustment of the conditional probability distribution of $\boldsymbol{R}_{\mathrm{ac},t}$. The value of W reflects the important of an active node $\boldsymbol{r}_{i,t} \in \boldsymbol{R}_{\mathrm{ac},t}$ to maintain its old similarity (s_{t-1}) to an inactive node $\boldsymbol{r}_{m,t} \in \boldsymbol{R}_{\mathrm{In},t}$.

Considering that $\boldsymbol{R}_{\mathrm{In},t}$ is directly affected by $\boldsymbol{R}_{\mathrm{In},t-1}$ (denoted as $\mathrm{DE}_{\boldsymbol{R}_{\mathrm{In},t-1}}$), we estimate their total effect to update the representation of inactive nodes, which is formulated as $(\gamma_1 \cdot \mathrm{DE}_{\boldsymbol{R}_{\mathrm{In},t-1}} + (1-\gamma_1) \cdot \mathrm{CE}_{\boldsymbol{I}_t, \boldsymbol{R}_{\mathrm{ac},t-1}})$. Given an inactive node m, we can infer $\mathrm{DE}_{\boldsymbol{R}_{\mathrm{In},t-1}}$ from the output of I-LightGCN (i.e., $\boldsymbol{r}_{m,t}$). As to $\mathrm{CE}_{\boldsymbol{I}_t, \boldsymbol{R}_{\mathrm{ac},t-1}}$, we follow Hu *et al.* [5] to infer a representation from the weighted distribution according to the K-nearest neighbors of m in s_{t-1}, which is formulated as

$$\sum_{n \in \mathrm{KNN}(\boldsymbol{R}_{\mathbf{ac},t-1}, m, K, \delta)} W\left(\boldsymbol{r}_{m,t}, s_{t-1}, \boldsymbol{r}_{n,t}\right) \cdot \boldsymbol{r}_{n,t}. \tag{14}$$

We omit $\boldsymbol{I}_t$ since no co-occurrence of node m and n in $\boldsymbol{I}_t$. As the simplest average weighting can achieve competitive performance [5], we set W as $1/K$. We, thus, obtain the formulation of CED in (9). Similarly, by considering the CE from $\boldsymbol{R}_{\mathrm{In},t-1}$ to $\boldsymbol{R}_{\mathrm{ac},t}$, we obtain (10).

III. Experiments

In this section, we evaluate *CI-LightGCN* on three real-world datasets to answer the following questions.

1) *RQ1:* How is the performance of CI-LightGCN compared with the existing retraining methods?
2) *RQ2:* How do the *CED* and *IGC* operators affect the recommendation performance?
3) *RQ3:* What factors (e.g., hyperparameters) significantly affect the recommendation performance of *CI-LightGCN*?

A. Experimental Settings

1) Datasets: We select three widely used datasets for recommendation: Yelp, Gowalla, and Adressa: 1) Yelp is adopted from the 2019 Yelp Challenge, which records the interactions between customers and local business in a period of more than ten years; 2) Gowalla[3] includes the check-in records in one year on go.gowalla.com, where users share their locations by checking in; and 3) Adressa is a news articles' clicks record dataset with historical interactions on Adressa in three weeks. For Yelp and Adressa, we adopt the version[4] preprocessed by Zhang *et al.* [9], where the interactions are chronologically split into stages. For Gowalla, we follow the same preprocessing procedure and split the interactions into 40 stages. We summarize detailed statistics of the datasets in Table I. The densities of Yelp, Gowalla, and Adressa are 0.042%, 0.084%, and 0.037%, respectively. We evaluate the effectiveness of top-K recommendation[5] by reporting the average recall@5, 20 (R@5 and R@20) and ndcg@5, 20 (N@5 and N@20) over interactions in the testing stages.

[3]http://snap.stanford.edu/data/loc-gowalla.html

[4]https://github.com/zyang1580/SML/

[5]The code of our method CI-LightGCN is available at https://github.com/Dingseewhole/CI_LightGCN_master/

TABLE I
Statistics of the Three Used Datasets

Dataset	#Users	#Items	#Interactions	#Train	#Validation	#Test
Yelp	122,816	59,082	3,014,421	[0-30) stage	[30-33) stage	[33-39] stage
Gowalla	29,858	40,981	1,027,370	[0-30) stage	[30-33) stage	[33-39] stage
Adressa	478,612	20,875	3,664,225	[0-48) stage	[48-53) stage	[53-62] stage

2) Compared Methods: We consider five LightGCN models with different retraining methods.

1) *Full-Retrain LightGCN:* This method retrains LightGCN with all interactions and full graph. We search the L2-norm coefficient in $[0, 0.001]$ at a multiplicative ratio of $10\times$ and training epochs in $[400, 1,000]$ with the step of 100.
2) *Fine-Tune LightGCN:* This method updates LightGCN with new interactions and the incremental graph.[6] We search the L2-norm coefficient in $[0, 0.001]$ at a multiplicative ratio of $10\times$ and training epochs in $\{200, 300, 400, 500\}$.
3) *SML + LightGCN-O:* This method retrains LightGCN with SML [9], which first trains a LightGCN over the incremental graph and then combines the old and new representations with a Transfer. We search the L2-norm coefficient in $[0, 0.01]$ at a multiplicative ratio of $10\times$; training epochs of LightGCN and Transfer in $[50, 600]$ with the step of 50; and learning rate in $[1e{-}5, 1e{-}2]$ at $10\times$ multiplicative ratio.
4) *SML + LightGCN-E:* It also retrains LightGCN with SML, which combines the parameters (i.e., the zeroth layer's node embedding) of the new model and old model. We search the same hype-parameters as **SML + LightGCN-O**.
5) *LightGCN + EWC:* This method retrains LightGCN with elastic weight consolidation (EWC) [12] over the incremental graph and new interactions, which is a regularization-based method in continual learning. We search the EWC ratio in $[0.01, 1]$ at a multiplicative ratio of $10\times$.

We also include some sequential recommendation baselines.

1) *GRU4Rec [13]:* It pioneered the usage of recurrent neural network (RNN) to serve the session-based recommender system. By building an RNN for each user's interaction sequence, it can capture the interest drift of users. We use the whole history to construct the user's interaction sequence and retrain the model with a full-retrain strategy. We search the hidden layer size of GRU in $\{64, 128, 256\}$.
2) *Caser [14]:* This method uses CNN to capture the interest evolution of users and the collaboration signal between items of the most L recent interactions. We tune L in the range of $[1, 5]$ with step 1, and other hyperparameters follow the optimal setting, as reported in this article.

[6]We omit the version of fine-tuning with old graph, which is less effective than full retraining and less efficient than fine-tuning w/o old graph.

TABLE II

RECOMMENDATION PERFORMANCE ON YELP AND GOWALLA. THE BEST PERFORMANCE AND BEST BASELINE IN EACH COLUMN ARE HIGHLIGHTED WITH BOLD FONT AND UNDERLINE, RESPECTIVELY. RI DENOTES **CI-LIGHTGCN**'S RELATIVE PERFORMANCE GAIN *w.r.t.* R@5

Methods	Yelp					Gowalla				
	R@5	R@20	N@5	N@20	RI	R@5	R@20	N@5	N@20	RI
GRU4Rec	0.1706	0.4158	0.1080	0.1771	92.5%	0.1016	0.2978	0.0623	0.1169	215.8%
Caser	0.2195	0.4565	0.1440	0.2117	49.6%	0.1143	0.3622	0.1111	0.3585	180.8%
SPMF	0.1725	0.3635	0.1136	0.1677	90.4%	0.1595	0.3759	0.0993	0.1606	101.2%
SML-MF	0.2251	0.4748	0.1485	0.2194	45.9%	0.1761	0.3428	0.1233	0.1701	82.2%
Fine-tune LightGCN	0.2338	0.4320	0.1619	0.2185	40.5%	0.2142	0.3860	0.1505	0.1993	49.8%
Full-retrain LightGCN	0.2923	0.5247	0.2074	0.2727	12.4%	0.3103	0.5276	0.2212	0.2819	3.4%
SML-LightGCN-O	0.1895	0.4197	0.1246	0.1897	73.3%	0.2152	0.4275	0.1505	0.2103	49.1%
SML-LightGCN-E	0.1771	0.3971	0.1159	0.1782	85.4%	0.2153	0.4515	0.1476	0.2144	49.0%
LightGCN+EWC	0.2365	0.4441	0.1736	0.2463	38.9%	0.2117	0.3942	0.1498	0.1987	51.6%
CI-LightGCN	**0.3284**	**0.5695**	**0.2294**	**0.2956**	-	**0.3209**	**0.5421**	**0.2272**	**0.2908**	-

3) *SPMF [15]:* It applies a sample-based retraining method on matrix factorization (MF), which samples some historical interactions to be added into new data to update the old model. We search the best reservoir size in $\{7,000, 15,000, 30,000, 70,000, 150,000\}$.
4) *SML-MF [9]:* This method applies SML on MF where the Transfer combines the old embedding and new embedding. We quote the result from SML [9].

B. Performance Comparison (RQ1)

Table II reports the performance comparison results. From the table, we have the following observations.

1) CI-LightGCN outperforms Full-retrain LightGCN and Fine-tune LightGCN in all cases. It validates the rationality of combining the old representation and incremental graph to approach the full graph convolution. The performance gain of CI-LightGCN is attributed to the IGC and CED, which properly fuses the long- and short-term preferences for all users and items.
2) CI-LightGCN also outperforms SML-LightGCN-O and SML-LightGCN-E, which combines the old and new representations with the complex Transfer model. We postulate the reasons are twofold.
   a) The complex Transfer model hurts the performance of LightGCN due to its feature transformation and nonlinear activation [2].
   b) The SML-based methods use degree within incremental graph only for normalization weighting, which ignores the accumulated degree. This result indicates the importance of normalization weighting in the GCN model and the importance of synchronizing the weights for GCN retraining.
3) LightGCN + EWC beats Fine-tune LightGCN since EWC alleviates the forgetting issue of the old preference signal. Nevertheless, CI-LightGCN further achieves performance gain over LightGCN + EWC. The reason for this is due to the IGC and CED operators. IGC transfers old representations into new representations, whereas EWC just treats it as a regularizer, and CED updates the representation of inactive nodes, but EWC is unable to do that.

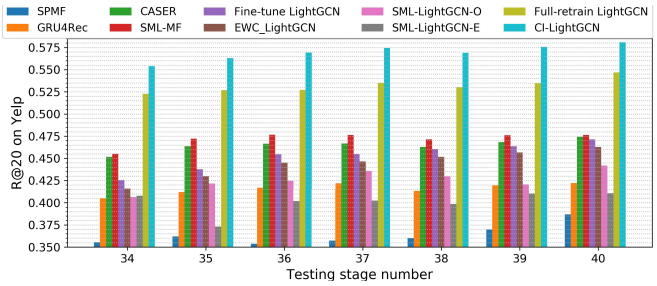


Fig. 4. Stagewise performance on Yelp.

4) On all datasets, LightGCN-based methods largely outperform the non-LightGCN ones in most cases, including SML-MF equipped with advanced retraining strategy. It is consistent with the result in [3], validating the effectiveness of GCN in the recommendation.

1) Stagewise Performance: Fig. 4 shows the detailed recommendation performance *w.r.t.* by R@10 at each testing stage of Yelp. To save space, we omit the results of other metrics and the results on Yelp, which shows the same trend. From the figure, we can see that CI-LightGCN stably outperforms the baselines across the stages.

2) Speed-Up: Recall that our target is the efficient retraining of the GCN model; we compare the time cost of LightGCN-based methods. Fig. 5 shows their training time on the same server with one RTX-3090 GPU. From the figure, we can see that: 1) CI-LightGCN speeds up the retraining with more than 30 times compared with full-retraining and 2) the running time of CI-LightGCN is slightly longer than fine-tuning. The results justify that IGC and CED enable the fast retraining of the GCN model, which is highly valuable in practice.

C. Ablation Study (RQ2)

1) Study on CED: To reveal the effect of CED and IGC, we further evaluate two variants of CI-LightGCN: 1) I-LightGCN, which only applies IGC on LightGCN, i.e., removing CED and 2) CI-LightGCN(T), which only uses CED in training, i.e., removing Steps 11 and 12 of Algorithm 1. Table III shows their recommendation performance on the two datasets. From the table, we can see that the recommendation

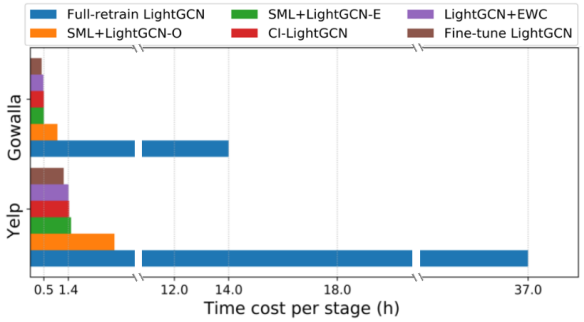


Fig. 5. Training time of different LightGCN-based methods.

TABLE III
PERFORMANCE OF CI-LIGHTGCN AND ITS VARIANTS

Datasets	Methods	R@5	R@20	N@5	N@20
Yelp	I-LightGCN	0.3222	0.5566	0.2261	0.2933
	CI-LightGCN(T)	0.3249	0.5663	0.2262	0.2954
	CI-LightGCN	**0.3284**	**0.5695**	**0.2294**	**0.2956**
Gowalla	I-LightGCN	0.3173	0.5369	0.2248	0.2878
	CI-LightGCN(T)	0.3179	0.5370	0.2260	0.2890
	CI-LightGCN	**0.3209**	**0.5421**	**0.2272**	**0.2908**

TABLE IV
PERFORMANCE OF CI-LIGHTGCN AND ITS VARIANTS ON INACTIVE USERS

Datasets	Methods	R@5	R@20	N@5	N@20
Yelp	I-LightGCN	0.2712	0.4724	0.1925	0.2507
	CI-LightGCN(T)	0.2783	0.4864	0.1945	0.2542
	CI-LightGCN	**0.2852**	**0.4959**	**0.1994**	**0.2598**
Gowalla	I-LightGCN	0.2807	0.4510	0.2026	0.2515
	CI-LightGCN(T)	0.2839	0.4549	0.2063	0.2543
	CI-LightGCN	**0.2870**	**0.4599**	**0.2091**	**0.2587**

performance of the three methods exhibits a clear increasing trend, which validates the rationality of updating inactive nodes during incremental training and the effectiveness of CED. Moreover, I-LightGCN still outperforms the baselines in Table II, which justifies the effectiveness of IGC. Furthermore, we evaluate the detailed performance of inactive users as the target of CED is to refresh their representations. Across Tables III and IV, we observe that CI-LightGCN(T) and CI-LightGCN achieve larger improvement over I-LightGCN on inactive users. It validates that CED can boost the recommendation accuracy of inactive users.

2) Study on IGC: We study how the components of IGC influence its effectiveness by comparing two variants of I-LightGCN without degree synchronizer (I-LightGCN w/o DS) and without representation aggregator as well (I-LightGCN w/o RA & DS). Meanwhile, we compare the I-LightGCN-MLP RA that enables parameter sharing across dimensions in the representation aggregator. Fig. 6 shows the performance comparison between vanilla I-LightGCN and its variants. From the figure, we can see that: 1) the performance of vanilla I-LightGCN (i.e., I-LightGCN - Conv RA), I-LightGCN w/o DS, and I-LightGCN w/o RA & DS shows a clear decrease trend in most cases, which justifies the effectiveness and the necessity of the two modules in IGC and 2) I-LightGCN-MLP RA performs much worse than the

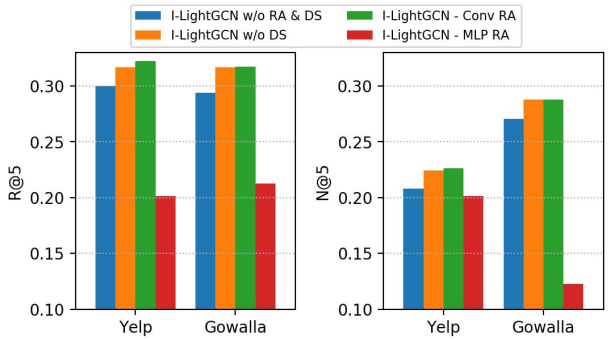


Fig. 6. Performance of I-LightGCN and its variants.

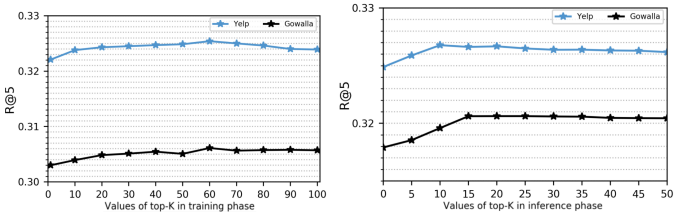


Fig. 7. Performance of CI-LightGCN as changing the top-K in the training phase (left) and the inference phase (right).

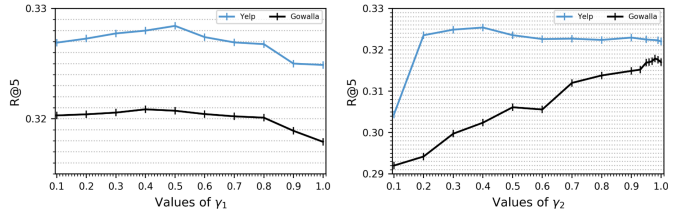


Fig. 8. Performance of CI-LightGCN as changing the value of γ_1 (left) and γ_2 (right).

one with CNN implementation across the two datasets, which demonstrates the rationality of restricting the complexity of the representation aggregator.

D. In-Depth Analysis (RQ3)

1) Study on CED: We investigate how hyperparameters influence the effectiveness of CED. We select four hyperparameters: the value of K and γ (i.e., γ_1 and γ_2) in (10) and (9). Figs. 7 and 8 show the performance of CI-LightGCN as changing the value of K and γ. From Fig. 7, we can observe a similar trend of performance (increasing then stable) across the four curves, i.e., CED is insensitive to K, and we can simply set K as a relatively small value (e.g., 15). Fig. 8 shows that CI-LightGCN performs worst as $\gamma_1 = 1.0$, which reveals the benefit of CED on inactive nodes. Moreover, the performance largely decreases as setting γ_2 with small values. This is reasonable since smaller γ_2 leads to less update of the active nodes during training.

2) Study on IGC: For IGC, we select two hyperparameters in the representation aggregator, the number of CNN layers and the number of CNN filters, to study the sensitivity of IGC. Fig. 9 shows the performances of I-LightGCN with one-layer CNN and two-layer CNN on Gowalla and Yelp as changing the number of filters from 1 to 20. We can see that:

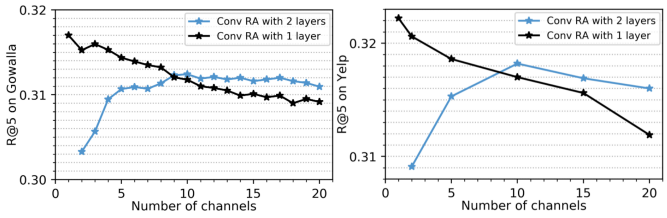


Fig. 9. Performance of I-LightGCN as changing the RA.

TABLE V

RECOMMENDATION PERFORMANCE OF LIGHTGCN-BASED METHODS ON ADRESSA. THE BEST AND RUNNER-UP IN EACH COLUMN ARE HIGHLIGHTED WITH BOLD FONT AND UNDERLINE, RESPECTIVELY

Methods	R@5	R@20	N@5	N@20
I-LightGCN	**0.254**	**0.415**	**0.181**	**0.227**
Full-retrain LightGCN	0.028	0.094	0.018	0.227
Fine-tune LightGCN	<u>0.253</u>	<u>0.412</u>	<u>0.181</u>	<u>0.227</u>
LightGCN + EWC	0.252	0.393	0.177	0.218
SML-LightGCN-O	0.243	0.411	0.170	0.216
SML-LightGCN-E	0.243	0.412	0.168	0.217

1) the one-layer CNN performs better than the two-layer CNN in most cases and 2) adding more filters will hurt the performance of one-layer CNN. This result suggests setting the aggregator with only one filter, which means that a weighted combination is sufficient for combining the old and new representations in practical usage.

3) Study on Time Sensitivity: As shown in [9], retraining methods can perform distinctly on datasets with different time sensitivities. We, thus, further test the LightGCN-based methods on Adressa, which is highly time-sensitive. Table V shows the performance of these methods. Due to the time-sensitivity, full-retrain LightGCN and LightGCN + EWC, which highlight the long-term preference signal, perform much worse than Fine-tune LightGCN. Nevertheless, I-LightGCN achieves performance comparable to Fine-tune LightGCN due to the flexibility brought by the representation aggregator and degree synchronizer in IGC, which enables the model to also approach Fine-tune LightGCN by down weighting the old representation and degree. This result demonstrates the robustness of IGC for different datasets.

IV. RELATED WORK

A. GCN-Based Recommendation

In recent years, GCN has become the cutting-edge technique for recommendation [16]–[22]. A surge of attention has been dedicated on designing GCN models to learn comprehensive user and item representations from the interaction graph for collaborative filtering [2], [3], [16]. Beyond the user–item interactions, a line of research explores GCN models to encode more types of relations, such as item relation, social network [23]–[25], and knowledge graph [26], [27]. However, the existing studies focus on model efficacy and largely ignore the efficiency of model training, which is very important in practical usage. In this work, we focus on efficient retraining of the GCN model, which is in an orthogonal direction to the existing research.

B. Dynamic GCN

The graph is a constantly changing data structure in the real world, such as social networks, academic networks, and point cloud. In response to this issue, plenty of previous work explores Dynamic GCN to capture the temporal feature of graph data [28]–[33], which are mainly in two categories according to viewing the graph evolution in stages or a random process. For instance, DySAT [34] arranges the snapshots of the graph into a sequence with the temporal order and uses two decoupled modules to encode the structure feature and the temporal feature within the sequence. DyRep [35] models the occurrence of an edge as a point process and uses a neural network to parameterize the intensity function.

There are also attempts to use Dynamic GCN for recommendation [24], [36], so as to capture the temporal pattern of user–item interactions. For instance, JODIE [36] couples an RNN with GCN to learn the dynamic representation of users/items from a sequence of interaction graphs. GraphSAIL [37] can retrain the graph-based recommender model with global and local structure distillations. Despite the success of Dynamic GCN in capturing the short- and long-term preference, it requires long interaction history for training, which is memory and time-costly by nature. In this work, we focus on the efficient retraining of the GCN model, where only the incremental graph is accessible in our setting. The Dynamic GCN models are, thus, not applicable here, which are omitted in the experiments for comparison. Although GraphSAIL [37] can incremental train the old model, it performs worse than intuitive full-retraining method (named *named Fu_batch* in [37, Tab. 3]). Since our target is to design a retraining method that is efficient as fine-tuning but without sacrificing the performance, and our proposed IGC and CED always perform better than full-retraining, we omitted this method in the experiments.

C. Causal Recommendation

A surge of attention has been dedicated to incorporating causal inference techniques into various machine learning applications [38]–[44]. These methods are also successfully adapted to recommendation systems for resolving bias issues and enhancing model reliability. For instance, DecRS [45], MACR [46], and KDCRec [47] analyze the biases of recommendation from causal view and use counterfactual learning techniques to debias. Similarly, intervention techniques are also leveraged to debias [48]–[50]. Existing works focus on leveraging causal inference techniques for debiasing. Our work is in an orthogonal direction as we focus on model retraining and CED, which is a new causal inference technique.

V. CONCLUSION

In this work, we highlighted the importance of GCN model retraining for the recommendation. To achieve effective and efficient retraining, our analysis enlightens that the key lies in detaching the old graph from neighborhood aggregation, meanwhile reserving the long-term preference signal and refreshing the inactive nodes. Toward this end, we proposed a causal IGC method that consists of two new operators named

IGC and CED. We instantiated IGC and CED based on LightGCN and conducted extensive experiments on three real-world datasets. The results show that CI-LightGCN outperforms full retraining with a speed-up of more than 30 times, validating the effectiveness and the rationality of our proposal.

This work opens up a new research direction about GCN model retraining and highlights a promising perspective about causal inference. In the future, we are interested in applying different model parameter updating techniques in IGC, such as metalearning [9]. In addition, we will test IGC and CED in more graph learning applications, such as user profiling and targeted advertising.

REFERENCES

[1] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 974–983.

[2] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. SIGIR*, 2020, pp. 639–648.

[3] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 165–174.

[4] J. Pearl, M. Glymour, and N. P. Jewell, *Causal Inference Statistics: A Primer*. Hoboken, NJ, USA: Wiley, 2016.

[5] X. Hu, K. Tang, C. Miao, X.-S. Hua, and H. Zhang, "Distilling causal effect of data in class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 3957–3966.

[6] H. Zhu *et al.*, "Bilinear graph neural network with neighbor interactions," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1452–1458.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. UAI*, 2009, pp. 452–461.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. WWW*, 2017, pp. 173–182.

[9] Y. Zhang *et al.*, "How to retrain recommender System?: A sequential meta-learning method," in *Proc. SIGIR*, 2020, pp. 1479–1488.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

[11] R. Shanmugam, "Causality: Models, reasoning, and inference," *Neurocomputing*, vol. 41, nos. 1–4, pp. 189–190, 2001.

[12] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," 2016, *arXiv:1612.00796*.

[13] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-K gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 843–852.

[14] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 565–573.

[15] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen, "Streaming ranking based recommender systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 525–534.

[16] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," in *Proc. ICML*, 2020, pp. 10936–10945.

[17] T. Chen and R. C.-W. Wong, "Handling information loss of graph neural networks for session-based recommendation," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 72–80.

[18] J. Jin *et al.*, "An efficient neighborhood-based interaction model for recommendation on heterogeneous graph," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 75–84.

[19] J. Xu, Z. Zhu, J. Zhao, X. Liu, M. Shan, and J. Guo, "Gemini: A novel and universal heterogeneous graph information fusing framework for online recommendations," in *Proc. KDD*, 2020, pp. 3356–3365.

[20] X. Yang, X. Du, and M. Wang, "Learning to match on graph for fashion compatibility modeling," in *Proc. AAAI*, 2020, pp. 287–294.

[21] T. Wei *et al.*, "Fast adaptation for cold-start collaborative filtering with meta-learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 661–670.

[22] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing GCN for recommendation," in *Proc. Web Conf.*, Apr. 2021, pp. 1296–1305.

[23] X. Wang, Z. Liu, N. Wang, and W. Fan, "Relational metric learning with dual graph attention networks for social recommendation," in *Proc. PAKDD*, 2020, pp. 104–117.

[24] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 555–563.

[25] W. Fan *et al.*, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 417–426.

[26] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. SIGKDD*, 2019, pp. 950–958.

[27] R. Sun *et al.*, "Multi-modal knowledge graphs for recommender systems," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1405–1414.

[28] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Companion Web Conf. Web Conf. (WWW)*, 2018, pp. 969–976.

[29] W. Qiu, H. Chen, and B. An, "Dynamic electronic toll collection via multi-agent deep reinforcement learning with edge-based graph convolutional networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4568–4574.

[30] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1515–1524.

[31] A. Pareja *et al.*, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI*, 2020, pp. 5363–5370.

[32] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, "Combining graph neural networks with expert knowledge for smart contract vulnerability detection," *IEEE Trans. Knowl. Data Eng.*, early access, Jul. 7, 2021, doi: 10.1109/TKDE.2021.3095196.

[33] Y. Zhuang, Z. Liu, P. Qian, Q. Liu, X. Wang, and Q. He, "Smart contract vulnerability detection using graph neural network," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3283–3290.

[34] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. WSDM*, 2020, pp. 519–527.

[35] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *Proc. ICLR*, 2019, pp. 1–25.

[36] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1269–1278.

[37] Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, and M. Coates, "Graphsail: Graph structure aware incremental learning for recommender systems," in *Proc. CIKM*, 2020, pp. 2861–2868.

[38] T. Wang, J. Huang, H. Zhang, and Q. Sun, "Visual commonsense R-CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10760–10770.

[39] K. Tang, J. Huang, and H. Zhang, "Long-tailed classification by keeping the good and removing the bad momentum causal effect," in *Proc. NeurIPS*, 2020, pp. 1513–1524.

[40] X. Yang, F. Feng, W. Ji, M. Wang, and T.-S. Chua, "Deconfounded video moment retrieval with causal intervention," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1–10.

[41] X. Yang, H. Zhang, G. Qi, and J. Cai, "Causal attention for vision-language tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9847–9857.

[42] F. Feng, J. Zhang, X. He, H. Zhang, and T.-S. Chua, "Empowering language understanding with counterfactual reasoning," in *Proc. ACL*, 2021, pp. 2226–2236. [Online]. Available: https://dblp.uni-trier.de/rec/conf/acl/FengZHZC21.html?view=bibtex

[43] L. Chen, X. Yan, J. Xiao, H. Zhang, S. Pu, and Y. Zhuang, "Counterfactual samples synthesizing for robust visual question answering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10800–10809.

[44] Y. Niu, K. Tang, H. Zhang, Z. Lu, X.-S. Hua, and J.-R. Wen, "Counterfactual VQA: A cause-effect look at language bias," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12700–12710.

[45] W. Wang, F. Feng, X. He, X. Wang, and T.-S. Chua, "Deconfounded recommendation for alleviating bias amplification," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1717–1725.

[46] T. Wei, F. Feng, J. Chen, Z. Wu, J. Yi, and X. He, "Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1791–1800.
[47] D. Liu, P. Cheng, Z. Dong, X. He, W. Pan, and Z. Ming, "A general knowledge distillation framework for counterfactual recommendation via uniform data," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 831–840.
[48] Y. Zhang *et al.*, "Causal intervention for leveraging popularity bias in recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 11–20.
[49] Y. Wang, D. Liang, L. Charlin, and D. M. Blei, "The deconfounded recommender: A causal inference approach to recommendation," 2018, *arXiv:1808.06581*.
[50] Y. Wang, D. Liang, L. Charlin, and D. M. Blei, "Causal inference for recommender systems," in *Proc. 14th ACM Conf. Recommender Syst.*, Sep. 2020, pp. 426–431.

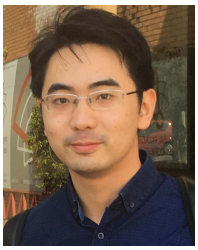
Sihao Ding received the B.E. degree from the School of Electronic Engineering, Xidian University, Xi'an, China, in 2018. He is currently pursuing the Ph.D. degree in cyberspace science and technology with the University of Science and Technology of China (USTC), Hefei, China.

His research interests lie in graph learning, recommender systems, and causal inference.

Fuli Feng received the Ph.D. degree in computer science from the National University of Singapore (NUS), Singapore, in 2019.

He is currently a Research Fellow with the School of Computing, NUS. He has over 40 publications appeared in several top conferences, such as International Conference on Research on Development in Information Retrieval (SIGIR), International Conference on Research on Development in Information Retrieval (WWW), and ACM International Conference on Multimedia (MM), and journals, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE) and *ACM Transactions on Information Systems* (TOIS). His research interests include information retrieval, data mining, and multimedia processing.

Dr. Feng has been serving as a PC Member for several top conferences, including SIGIR, WWW, ACM International Conference on Web Search and Data Mining (WSDM), Annual Conference on Neural Information Processing Systems (NeurIPS), AAAI Conference on Artificial Intelligence (AAAI), Annual Meeting of the Association for Computational Linguistics (ACL), and MM, and an Invited Reviewer for prestigious journals, such as TOIS, TKDE, TNNLS, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), and IEEE TRANSACTIONS ON MULTIMEDIA (TMM). His work has received the Best Paper Award Honorable Mention in SIGIR 2021 and the Best Poster Award in WWW 2018.



Xiangnan He received the Ph.D. degree in computer science from the National University of Singapore (NUS), Singapore, in 2016.

He is currently a Professor with the University of Science and Technology of China (USTC), Hefei, China. He has over 80 publications that appeared in several top conferences, such as International Conference on Research on Development in Information Retrieval (SIGIR), International Conference on Research on Development in Information Retrieval (WWW), and ACM International Conference on Multimedia (MM), and journals, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), *ACM Transactions on Information Systems* (TOIS), and IEEE TRANSACTIONS ON MULTIMEDIA (TMM). His research interests span information retrieval, data mining, and multimedia analytics.

Dr. He has served as an Senior Program Committee (SPC)/Program Committee (PC) Member for several top conferences, including SIGIR, WWW, ACM Knowledge Discovery and Data Mining (KDD), MM, ACM International Conference on Web Search and Data Mining (WSDM), and International Conference on Machine Learning (ICML). His work has received the Best Paper Award Honorable Mention in WWW 2018 and ACM SIGIR 2016. He is on the editorial board of journals, including *Frontiers in Big Data* and *AI Open*. He has served as the PC Chair of IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS) 2019 and a Regular Reviewer for journals, including TKDE, TOIS, and TMM.

Yong Liao received the bachelor's degree from the University of Science and Technology of China, Hefei, China, in 2001, the master's degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 2004, and the Ph.D. degree from the University of Massachusetts Amherst, Amherst, MA, USA, in 2010.

He was a Senior Member of Technical Staff with Narus, a Boeing company, Chicago, USA, from 2011 to 2014. He is currently an Adjunct Professor with the School of Cyberspace Science and Technology, University of Science and Technology of China, Hefei, China. He has over 60 publications and 12 awarded U.S. patents. His research interests include machine learning, and network security and privacy.

Jun Shi received the bachelor's and master's degrees from the Harbin Engineering University, Harbin, China, in 2010 and 2013, respectively, and the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2017.

She is currently the Director of the Innovation Laboratory in a wholly-owned subsidiary of CETC, Beijing. She has over 15 publications and five awarded China patents. Her research interests include machine learning, data analytics, and network security and privacy.

Yongdong Zhang (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from Tianjin University, Tianjin, China, in 2002.

He is currently a Professor with the School of Information Science and Technology, University of Science and Technology of China, Hefei, China. He has authored more than 100 refereed journal articles and conference papers. His research interests include multimedia content analysis and understanding, and multimedia content security.

Prof. Zhang also serves as an Editorial Board Member of the *Multimedia Systems* journal and the TMM. He was a recipient of the Best Paper Awards in Pacific-Rim Conference on Multimedia (PCM) 2013, International Conference on Internet Multimedia Computing and Service (ICIMCS) 2013, and IEEE International Conference on Multimedia and Expo (ICME) 2010, and the Best Paper Candidate in ICME 2011.