Graph Contrastive Learning with Positional Representation for Recommendation

Zixuan Yi[1], Iadh Ounis[2], Craig Macdonald[2]

University of Glasgow, Glasgow, UK
[1]z.yi.1@research.gla.ac.uk
[2]{firstname.secondname}@glasgow.gla.ac.uk

Abstract. Recently, graph neural networks have become the state-of-the-art in collaborative filtering, since the interactions between users and items essentially have a graph structure. However, a major issue with the user-item interaction graph in recommendation is the absence of the positional information of users/items, which limits the expressive power of graph recommenders in distinguishing the users/items with the same neighbours after propagating several graph convolution layers. Such a phenomenon further induces the well-known over-smoothing problem. We hypothesise that we can obtain a more expressive graph recommender through graph positional encoding (e.g., Laplacian eigenvector) thereby also alleviating the over-smoothing problem. Hence, we propose a novel model named Positional Graph Contrastive Learning (PGCL) for top-K recommendation, which aims to explicitly enhance graph representation learning with graph positional encoding in a contrastive learning manner. We show that concatenating the learned graph positional encoding and the pre-existing users/items' features in each feature propagation layer can achieve significant effectiveness gains. To further have sufficient representation learning from the graph positional encoding, we use contrastive learning to jointly learn the correlation between the pre-exiting users/items' features and the positional information. Our extensive experiments conducted on three benchmark datasets demonstrate the superiority of our proposed PGCL model over existing state-of-the-art graph-based recommendation approaches in terms of both effectiveness and alleviating the over-smoothing problem.

1 Introduction

Personalised recommendation is a widely used technology to improve the quality of information services, which aims to predict a group of items that users might intend to purchase according to their preferences. The effective personalisation of the recommendation results, typically rely on rich available data, in particular the historical user-item interactions [11]. Recent advances in Graph Neural Networks (GNNs) provided a strong and fundamental opportunity to develop effective personalised recommendations [29]. Specifically, GNNs adopt embedding propagation to aggregate neighbourhood embeddings iteratively through connectivities on a bipartite user-item graph. By stacking the multiple propagation layers, each node on the graph can access high-order neighbours' information through the message passing scheme [31], rather than only modelling the direct interactions between users and items. With their advantages in handling structural data and exploring structural information on a graph, graph recommenders have attained a state-of-the-art recommendation performance [6].

Despite the success of the graph recommenders, the current graph feature propagation function only repeatedly aggregates neighbourhood embeddings that are adjacent to the target node. As a consequence, the conventional message passing scheme of the graph recommenders typically fails to differentiate two users with the same interacted items and all user representations converge to a constant after propagating several graph convolution layers [2]. This problem can be further amplified after stacking multiple layers [10], leading to the well-known over-smoothing problem [17]. Indeed, this limitation is now well understood in the context of the equivalence of GNNs with the Weisfeiler-Leman (WL) test [30] for graph isomorphism [22,34], which further confirms the limited expressive power of the current graph recommenders. Consequently, there is a stronger motivation for proposing a new graph recommender that is more expressive in distinguishing the users/items with the same neighbours after graph convolution and hence to further amplify the difference between the users/items further apart. Indeed, many approaches have been proposed to alleviate the limited expressive power of the GNNs, to some extent, by considering the positional encoding (PE) information of nodes for enriching the nodes' features [4,14,25]. Graph positional encoding approaches [3,4,37] typically consider a global positioning or a unique representation of the users/items in the graph, which can encode a graph-based distance between the users/items. To leverage the advantage of positional encoding, in this paper, we also use a graph-specific learned positional encoding as a unique ID for each user/item and inject these positional encodings into each feature propagation layer to improve the expressive power of graph recommenders.

Inspired by recent studies [15,32,33,39], which have shown the superior ability of Contrastive Learning (CL) to construct supervised signals from correlations within raw data, we also investigate in this paper the possibility of leveraging CL to explore the correlations among learned graph positional encodings and address the limited expressive power problem in graph recommenders. A typical approach [32,40] to apply CL to recommendations on graphs is to first augment the user-item bipartite graph with noise or structure perturbations, and then to maximise the agreement of the augmented user/item embeddings via a graph encoder. To address the limited expressive power of graph recommenders, we propose a novel recommendation model named Positional Graph Contrastive Learning (PGCL) for top-K recommendation, which aims to use existing graph positional encoding methods to improve the expressive power of graph recommenders and further enhances the integrated user/item representations through a noise-based augmentation method. To be more specific, PGCL provides additional positional information to existing graph recommenders by injecting the learned graph positional encoding into each feature propagation layer. Furthermore, in order to prevent distorting the users' intents, we apply a noise-based augmentation technique to such position-enhanced user/item embeddings – this aims to maintain the users' intent unchanged while adding distance properties to the learned user/item representations. To summarise, in this paper, we argue that graph recommenders enriched with our proposed graph positional encoding can effectively improve their expressive power while alleviating the over-smoothing problem. Indeed, we show that with the integration of contrastive learning, our PGCL model enforces the divergence of the learned user/item representations resulting in an improved recommendation performance.

Our contributions in this paper are as follows: (1) We propose a personalised graph-based recommendation model for top-K recommendation, which leverages the learned graph positional encoding to facilitate a new message passing scheme for existing graph recommenders; (2) We apply noise-based augmentation on position-enhanced user/item embeddings and examine the impact of the resulting PGCL model using different ranking metrics; (3) We conduct extensive experiments on three benchmark datasets and demonstrate the effectiveness of PGCL in comparison to the existing state-of-the-art graph recommenders; (4) By comparing with the existing baselines, we show that PGCL is more expressive by stacking multiple layers and can alleviate the over-smoothing problem by reducing the over-smoothness of user/item embeddings.

2 Related Work

In this section, we discuss the related methods and techniques to our conducted study, namely graph-based recommendation, graph positional encoding and graph contrastive learning for recommendation.

Graph-based Recommendation: Graph-based recommenders [10,20,29] typically exploit the message passing scheme in the user-item graph by propagating information from local neighbours and integrating the collaborative signals into a user/item representation. However, the existing approaches (e.g., LightGCN [10]) follow the original message passing scheme, which is known to suffer from over-smoothing due to its repeated aggregation of local information. As a result, the existing graph recommenders only propagate homogeneous features (e.g., IDs) from the original neighbours, which are not expressive enough to distinguish the users/items with the same neighbours after stacking several graph convolution layers. Unlike prior works, we leverage the positional representation of each user/item that relies on positional features (e.g., Laplacian eigenvectors) and inject the learned positional encodings into each feature propagation layer of the existing graph recommenders so as to enhance their expressive power.

Graph Positional Encoding: The notion of positional encodings (PEs) in graphs is not a trivial concept, as there exists no canonical way of ordering nodes [14]. Various studies [3,5,14,18,8,28,36] have exploited positional encodings on graphs to improve the expressiveness of GNNs. Many earlier studies [21,23] used index positional encoding to enhance conventional GNNs in terms of their associated model expressiveness. For example, GRP [23] devised a positional encoding by assigning to each node an identifier that depends on the index or-

dering. This approach can be computationally expensive as it needs to account for all $n!$ node permutations to guarantee a higher expressiveness. Therefore, some prior studies (e.g., [16,37]) have applied a more efficient distance positional encoding to enhance the model expressiveness of GNNs. For example, P-GNN [37] enhanced the model expressiveness by projecting the distances between a target node and randomly sampled nodes into a position-aware embedding. However, a large number of sampled nodes will include most of the nodes on the graph, thus leading to insufficient positional embeddings. DEGNN [16] modeled a distance positional encoding by capturing distances between nodes using landing probabilities of random walks. However, this approach cannot scale to large-scale graphs because of the cost of computing the power matrices. Alternatively, Laplacian eigenvectors [1] have been shown to be good candidates for graph positional encoding, since Laplacian eigenvectors form a meaningful local coordinate system while preserving the global graph structure. In particular, we can pre-compute the Laplacian eigenvectors/eigenvalues and provide a unique ID for each node, which solves the scalability issue on the user-item graph in a recommender system and further enhances the pre-existing node features by merging Laplacian eigenvectors/eigenvalues. Another alternative approach used by APPNP [13] provided an improved graph feature propagation scheme with Personalised PageRank [9], which particularly addresses the over-smoothing problem in a random walk manner. In this paper, we leverage the Laplacian eigenvectors and the random walk operator to define a new relative positional encoding in recommender systems. Unlike the above prior works, we allocate the learned positional encodings to a separate message passing function to generate the user/item positional embedding from the neighbours' positional information.
Graph Contrastive Learning for Recommendation: Recently, graph-based recommendation approaches [19,32,35,38] have benefited from contrastive learning, because its ability to extract contrastive signals from the raw data is well-aligned with the recommender systems' needs for more collaborative filtering signals. SGL [32] adopted different augmentation operators such as edge dropout and node dropout, which aim to capture the essential information of the original user-item bipartite. The authors of SimGCL [40] claimed that graph augmentations highly distort the original graph and applied a more effective noise-based augmentation on a user/item representation level. As discussed above, we aim to inject the learned positional encodings into the node features to enhance the final user/item embeddings. Hence, applying a representation-level augmentation is more reasonable than perturbing the graph structure. To the best of our knowledge, our proposed PGCL model is the first graph-based recommendation model to enhance user/item representations by contrasting augmented user/item embeddings with a learned graph positional encoding.

3 Model Architecture

In this section, we first present the personalised recommendation task in Section 3.1. We describe our proposed PGCL model, the architecture of which is

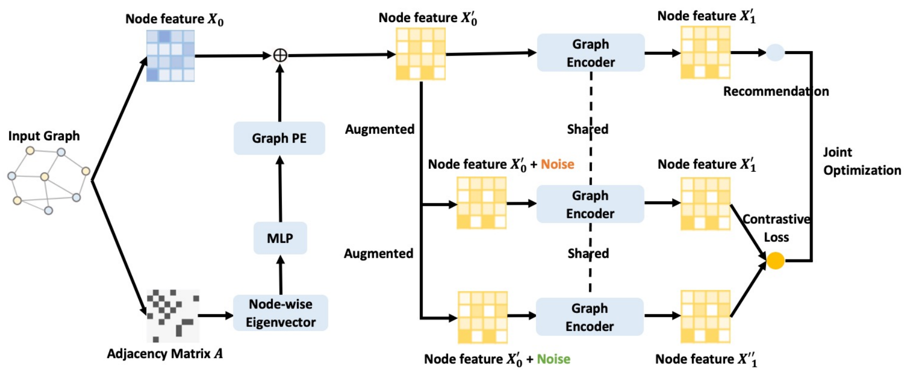

Fig. 1: The architecture of our PGCL model

illustrated in Figure 1. In Section 3.2, we define the graph positional encoding. Next, we illustrate the motivation of decoupling the positional encoding from the conventional message passing function in Section 3.3. In Section 3.4, we apply contrastive learning for effectively learning the positional encoding and optimise our PGCL model jointly with a pairwise ranking loss.

3.1 Preliminaries

In this paper, we focus on addressing the ranking-based recommendation task. Conceptually, we consider a recommender system with a user set $\mathcal{U}$ and an item set $\mathcal{I}$. In order to facilitate the description of graph recommenders, we use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote an interaction graph, where the node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ includes all users and items. $\mathcal{E}$ is the set of edges. e_u denotes the user feature for user u and e_i denotes the item feature for item i. In addition, p_u and p_i denote the positional feature of the user and item, respectively. The layers are indexed by ℓ, where $\ell = 0$ denotes the input layer. For a given user u or item i, there is a positional feature p_u or p_i on an interaction graph $\mathcal{G}$. We aim to estimate the users' preferences through a graph encoder f, which can recommend the top-K items for a target user u.

3.2 Definition of Initial Graph Positional Encoding

As mentioned Section 2, we aim to use the Laplacian eigenvectors and the random walk operator to define the positional encoding (PE) in recommendation. In this section, we define the initial positional encoding of a given user u or item i.

Laplacian PE: Laplacian PE (LapPE) is a spectral technique that embeds graphs into an Euclidean space, and is defined via the factorisation of the graph's Laplacian $\Delta = \mathrm{I} - D^{-1/2}AD^{-1/2} = U^T \Lambda U$, where I is the identity matrix, A is the adjacency matrix, D is the degree matrix, and matrices Λ and U correspond to the Laplacian eigenvalues and Laplacian eigenvectors of a graph, respectively.

In this work, we consider the Laplacian eigenvector as the initial graph positional encoding, which is defined as follows:

$$p_i^{\mathrm{LapPE}} = [U_{i1}, U_{i2}, \cdots, U_{ik}] \in \mathbb{R}^k \tag{1}$$

As a consequence, LapPE is expected to provide a unique ID for each user/item representation and is distance-sensitive w.r.t. the Euclidean norm.

Random Walk PE: Apart from LapPE, we also investigate the random walk-based method [5] to generate the graph positional encoding. Hence, we use Random Walk PE (RWPE), which is a method based on the random walk diffusion process. Formally, RWPE is defined with k-steps of random walks as follows:

$$p_i^{\mathrm{RWPE}} = \left[\mathrm{RW}_i, \mathrm{RW}_i^2, \cdots, \mathrm{RW}_i^k\right] \in \mathbb{R}^k \tag{2}$$

where $\mathrm{RW} = AD^{-1}$ is the random walk operator. As such, RWPE provides a unique node representation under the condition that each user/item has a unique k-hop topological neighbourhood [16] for a sufficiently large k.

Finally, the initial graph PE of the network is obtained by projecting LapPE or RWPE into a d-dimensional feature vector with a Multi-Layer Perceptron (MLP) network:

$$p_i^{\ell=0} = \mathrm{MLP}\left(p_i^{\mathrm{PE}}\right) = W^0 p_i^{\mathrm{PE}} + b^0 \in \mathbb{R}^d, \tag{3}$$

where $W^0 \in \mathbb{R}^{d \times k}$ and $b^0 \in \mathbb{R}^d$ are the learned parameters of the MLP network. As illustrated in Figure 1, we leverage LapPE or RWPE to generate the initial positional encoding through a MLP network.

3.3 Feature Propagation with Learned Positional Encoding

Figure 1 illustrates how PGCL concatenates the graph PE and the pre-existing node feature X' which is generated by user/item IDs. As discussed in Section 2, we aim to decouple the graph PE from the conventional message passing function. Hence, we propose a message passing function for the graph PEs. The layer update equations is defined as follows:

$$p_u^{\ell+1} = \mathrm{Tanh}\left(\mathrm{AGG}\left(p_u^{\ell}, \{p_i^{\ell}\}_{i \in \mathcal{N}_u}\right)\right), \tag{4}$$

where Tanh is the activation function, and AGG is an aggregation function that combines the positional information of the adjacent item nodes. Since the graph positional encoding is interpreted as a unique positional ID of a given node (see Section 2), we expect to use the message passing scheme to exploit high-order positional information of the positional features through the aggregation operation. Next, we aim to integrate the graph PE p_i into user representations. Analogously, we can obtain the updated positional embeddings of the items.

As illustrated in Figure 1, we concatenate the graph PE – which is generated by Equation (4) – with the existing node feature X, similar to the Transformers [27] network structure:

$$e_u^{\ell+1} = \mathrm{AGG}\left(\begin{bmatrix} e_u^\ell \\ p_u^\ell \end{bmatrix}, \left\{\begin{bmatrix} e_i^\ell \\ p_i^\ell \end{bmatrix}\right\}_{i \in \mathcal{N}_u}\right) \tag{5}$$

where e_u and e_i denote the representations of user u and item i, respectively, $\mathcal{N}_u$ is the neighbourhood of the user u, and p_u & p_i denote the position representations of user u and item i, respectively. We use LightGCN [10] to aggregate the concatenated result of the pre-exiting item feature e_i and with the item positional feature p_i. Hence, the feature propagation equation is defined as follows:

$$\mathbf{e}_u^{\ell+1} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \begin{bmatrix} e_i^\ell \\ p_i^\ell \end{bmatrix}, \tag{6}$$

$$\text{with } p_i^\ell = \mathrm{Tanh}\left(W_1 p_i^{\ell-1} + \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \left(W_1 p_u^{\ell-1} + W_2\left(p_u^{\ell-1} \odot p_i^{\ell-1}\right)\right)\right) \tag{7}$$

where W_1 and W_2 are trainable weight matrices. The main difference of our feature propagation layer with the standard graph recommenders is a separated message passing function for the graph PE, which injects the graph PE into each propagation layer. For this reason, we expect PGCL to provide less over-smoothed user/item embeddings, thereby alleviating the over-smoothing problem of the existing graph recommenders.

3.4 Self-augmented Learning

As discussed in Section 2, since a graph perturbation has the possibility to distort the user-item bipartite graph, applying a representation-level augmentation on the learned graph PE is more rational than perturbing the graph structure. Following Yu et al. [40], we apply a noise-based augmentation on the representation level for both the integrated user and item embeddings. For example, given a user embedding e_u, which integrates its graph positional feature p_u, we can generate an augmented user representation by adding a noise vector Δ_u as follows:

$$\mathbf{e}_u' = \mathbf{e}_u + \Delta_u', \mathbf{e}_u'' = \mathbf{e}_u + \Delta_u'',\ e_u \in \mathbb{R}^d \tag{8}$$

$$\text{with } \Delta_u = e_x \odot \mathrm{sign}\left(\mathbf{e}_u\right) \odot \epsilon,\ e_x \in \mathbb{R}^d \sim U(0,1) \tag{9}$$

where $\mathbf{e}_u'$ and $\mathbf{e}_u''$ are two augmented user representations, e_x is a vector that is generated by random numbers from a uniform distribution, and ϵ is a hyperparameter to control the strength of the user representation perturbation with a range in $[0,1]$. Goodfellow et al. [7] have also shown that a linear perturbation in high-dimensional spaces can generate sufficient samples. As such, in addition to applying the noise-based augmentation, we also aim to enforce that the integrated user/item representations further spread out in the entire embedding space so as to fully exploit the expressive power of the embedding space. In the graph contrastive recommendation scenario [32,40], the target is to generate a better user/item representation via data augmentations. Hence we use InfoNCE [24], to maximise the agreement of two augmented representations:

$$\mathcal{L}_{cl}^{user} = -\log \frac{\exp\left({\mathbf{e}'_u}^{\top}\mathbf{e}''_u/\boldsymbol{\tau}\right)}{\sum_{i=1}^{n}\exp\left({\mathbf{e}'_u}^{\top}\mathbf{e}_n/\boldsymbol{\tau}\right)} \tag{10}$$

where $\mathbf{e}_n$ is the embedding of a different user, and τ is a hyper-parameter that adjusts the dynamic range of the resulting loss value. Analogously, we can calculate the contrastive loss of a target item $\mathcal{L}_{cl}^{item}$. Therefore, we obtain a combined contrastive loss that acts as an auxiliary loss for top-K recommendation tasks as follows: $\mathcal{L}_{cl} = \mathcal{L}_{cl}^{user} + \mathcal{L}_{cl}^{item}$. To better mine the user/item representations in recommendation, we adopt a multi-task training strategy to jointly optimise the widely used pair-wise ranking objective, namely Bayesian Personalised Ranking (BPR) [26], and the contrastive learning objective $\mathcal{L}_{cl}$:

$$\mathcal{L} = \lambda_1\mathcal{L}_{cl} + \sum_{(u,i,j)\in D_s} \ln\sigma(y_{ui} - \boldsymbol{e_u}^{\mathrm{T}}\boldsymbol{e_i}) + \lambda_2\left\|\Theta\right\|_2^2 \tag{11}$$

where the second term is the BPR loss, $\boldsymbol{e_u}$ is the user embedding, $\boldsymbol{e_i}$ denotes the positive item embedding and y_{ui} is the ground truth value, which indicates whether the paired user and item have interacted, $D_s = \{(u,i,j)|(u,i) \in R^+, (u,j) \in R^-\}$ is the set of the training data, R^+ indicates the observed interactions and R^- indicates the unobserved interactions; $\sigma(\cdot)$ is the sigmoid function, Θ is the set of model parameters in the BPR loss, while λ_1 and λ_2 are hyper-parameters to control the strengths of the contrastive learning and L_2 regularisation, respectively. Through propagating the integrated user/item representations in multiple feature propagation layers with Equation (6), we obtain multiple user/item embeddings from each layer, then we concatenate each user/item embedding e_u^{ℓ}, so that the final embedding collectively contains information from each layer. Hence, we can estimate the relevant score between a user and item by minimising the multi-task learning loss in Equation (11).

4 Experiments

We now examine the performance of PGCL through experiments on three real-world datasets, in comparison to four existing state-of-the-art graph recommendation models. To demonstrate the effectiveness of PGCL, we conduct experiments to answer the following three research questions:

RQ1: How does the PGCL model perform in top-K recommendation compared with existing baselines?
RQ2: How do different positional encodings and augmentation methods impact the recommendation performance?
RQ3: Is our PGCL model more expressive than LightGCN thereby alleviating the over-smoothing problem compared to the baselines based on LightGCN?

Table 1: Statistics of the used datasets.

Dataset	Users	Items	Interactions	Density
Gowalla	39, 657	31, 211	1,072,325	0.087%
Yelp2018	28,361	43,142	1,481,472	0.121%
Amazon-Kindle	116,417	72,439	1,643,646	0.019%

4.1 Datasets and Evaluation Protocol

We evaluate our PGCL model using three real-world datasets, namely *Yelp2018*,[1] *Gowalla*[2] and *Amazon-Kindle*.[3] Table 1 shows the statistics of these datasets. Following He et al. [10] and Wang et al. [29], we randomly split the above datasets into training, validation, and testing sets with a 7:1:2 ratio. We use two commonly used evaluation metrics: Recall@K and NDCG@K to evaluate the performance of top-K recommendation. We follow [40] in setting K = 20 and report the average performance achieved for all users in the testing set. We use the Adam [12] optimiser in both our PGCL model and the four baseline models. We apply early-stopping during training, terminating the training when the validation loss does not decrease for 50 epochs. To determine the hyper-parameters in both PGCL and the baseline models, we apply a grid search on the validation set. Specifically, we tune our PGCL model by varying the learning rate in $\{10^{-2}, 10^{-3}, 10^{-4}\}$. The learning rates of the baseline models are also tuned according to the suggested ranges in [10], for a fair comparison. Similarly, we also tune each of λ_1, λ_2 and ϵ within the range of $\{0, 0.1, 0.2, ..., 1.0\}$. A detailed analysis of the models' performance with different layer settings is shown in Section 4.5.

4.2 Baselines

We compare the effectiveness of PGCL[4] with four existing strong baselines. In the following, we briefly describe these baselines: **(1) NGCF [29]** is a classical GNN-based model that first captures the high-order connectivity information in the embedding function by stacking multiple embedding propagation layers. **(2) LightGCN [10]** is another GNN-based model that has evolved from NGCF. It simplifies the design in the feature propagation by removing the non-linear activation and the transformation matrices. This approach has been widely used as a strong graph recommender for top-K recommendation [6]. **(3) SGL [32]** leverages contrastive learning for GNN-based models. With LightGCN as the encoder of the users/items, SGL adopts different augmentation operators such as edge dropout and node dropout, on the pre-existing features of the users/items. This approach can implicitly identify the important nodes from different augmentations [41]. **(4) SimGCL [40]** is effective in improving LightGCN with different augmentations, which is similar to SGL. It removes the

[1] `https://www.yelp.com/dataset`
[2] `https://snap.stanford.edu/data/loc-gowalla.html`
[3] `https://jmcauley.ucsd.edu/data/amazon/`
[4] Source code is available at: `https://github.com/zxy-ml84/PGCL`

dropout-based augmentations from SGL and devises a noise-based augmentation on the user/item representation level with an increased recommendation performance. In addition, to examine the effectiveness of the Laplacian positional encoding (see Equation (1)), we compare PGCL to a variant called $\mathbf{PGCL}_{w/o\,CL}$. Different from PGCL, $\mathrm{PGCL}_{w/o\,CL}$ only concatenates the positional encoding (from Equation (7)) with the pre-existing users/items' features from LightGCN, without applying contrastive learning (Equation (10)).

4.3 Performance Comparison with Baselines (RQ1)

Table 2 compares our proposed PGCL model with four used baselines. We particularly compare PGCL to the strongest baseline, whose performance is highlighted with an underline in the table. From the table, we observe that for all three datasets, PGCL outperforms all the baseline models on all metrics, and statistically significantly in most cases according to the paired t-test with Holm-Bonferroni correction. This result demonstrates the rationality and effectiveness of injecting graph PE to the graph feature propagation layer and incorporating the augmented positional and pre-existing node features (i.e. IDs). For a given GNN-based method (NGCF, LightGCN, $\mathrm{PGCL}_{w/o\,CL}$, PGCL), we evaluate the usefulness of leveraging the graph PE in enriching the user/item representations. Comparing NGCF, LightGCN and $\mathrm{PGCL}_{w/o\,CL}$, we observe that $\mathrm{PGCL}_{w/o\,CL}$ performs generally better than both NGCF and LightGCN on all three used datasets. This result demonstrates the benefit of injecting the learned positional encoding into the pre-exiting users/items' features to estimate the users' preferences. On the other hand, as can be observed in Table 2, $\mathrm{PGCL}_{w/o\,CL}$ performs worse than PGCL on all three used datasets. This result illustrates the importance of contrastive learning in providing additional supervised signals during training. For the contrastive learning method, we also evaluate the usefulness of different augmentations by comparing our PGCL model with SGL and SimGCL. Table 2 shows that the noise-based methods (SimGCL, PGCL) markedly outperform the dropout-based method (SGL) on both the Yelp2018 and Amazon-Kindle datasets. This result shows the marginal effect of graph perturbation and the effectiveness of using noise-based augmentation. Moreover, Table 2 also shows that PGCL outperforms SimGCL by a large margin on all metrics (significantly on Gowalla), which demonstrates that graph PE can enrich the user/item representations as an additional feature. Hence, in answer to RQ1, we conclude that our proposed PGCL model can effectively leverage both the graph positional feature and the augmented user/item representations, thereby enhancing the existing graph recommender models with significant performance improvements.

4.4 Ablation Study (RQ2)

To investigate the impact of each component of our PGCL model and different graph positional encodings (PE), Table 3 shows how the performance of PGCL changes when we start with LightGCN as the basic graph encoder and apply contrastive positional encoding on top of it so as to conclude on the effectiveness

Table 2: Experimental results for PGCL in comparison to other baselines. The best performance is highlighted in bold and the second best result is highlighted with underline. * denotes a significant difference compared to the result of PGCL using the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Dataset	Yelp2018		Gowalla		Amazon-Kindle	
Methods	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
NGCF	0.0502*	0.0417*	0.0889*	0.0592*	0.1893*	0.1285*
LightGCN	0.0542*	0.0437*	0.0996*	0.0635*	0.2230*	0.1644*
SGL	0.0563*	0.0449*	<u>0.1071</u>	<u>0.0671</u>	0.2331*	0.1726*
SimGCL	<u>0.0577</u>	<u>0.0466</u>	0.1068*	0.0664*	<u>0.2425</u>	<u>0.1801</u>
PGCL$_{w/o\,CL}$	0.0581*	0.0477*	0.1059*	0.0675*	0.2420*	0.1803*
PGCL	**0.0608**	**0.0501**	**0.1122**	**0.0699**	**0.2572**	**0.1934**
%Improv.	5.37%	7.51%	4.76%	4.17%	6.06%	7.38%

of graph PE and contrastive learning. Table 3 shows that the PGCL$_{LapPE}$ variant, which uses the Laplacian eigenvalue and a representation level augmentation achieves the best performance on all datasets. This promising result is due to the addition of the unique learned positional features of the users/items and an effective representation learning on the users/items' embeddings. Specifically, we observe from Table 3 that both LightGCN$_{RWPE}$ and LightGCN$_{LapPE}$ achieve an effectiveness gain compared with LightGCN. This result demonstrates the effectiveness of graph PE. One possible reason is that the graph PE denotes a unique positional information to the user/item embedding in each feature propagation layer. For the PGCL$_{RWPE}$ and the LightGCN$_{RWPE}$ variants, which use a random walk operator in Table 3, there is a performance reduction compared with PGCL$_{LapPE}$ and LightGCN$_{LapPE}$, which indicates that a global ID (LapPE) is more beneficial than a local ID (RWPE) for the user-item interaction data in recommender systems. Moreover, we also observe that there is an effectiveness improvement from LightGCN$_{LapPE}$ to PGCL$_{LapPE}$. This suggests that both the PE and the pre-existing users/items' features provide an additional supervised signal through contrastive learning. Hence, in answer to RQ2, we conclude that PGCL successfully leverages graph positional encodings to learn effective user/item representations in a contrastive learning scheme.

4.5 The Over-smoothing Problem (RQ3)

After showing that PGCL is effective in improving LightGCN, we now study the characteristics of graph PE in terms of their usefulness against over-smoothing. In this section, we investigate the over-smoothing problem by comparing PGCL and LightGCN with different layer settings in Table 4. As shown in Table 4, both PGCL and LightGCN reach their best effectiveness within 5 graph layers. In addition, all PGCL variants are effective in improving LightGCN under different layer settings on all used datasets. The largest improvements are observed on the Amazon-Kindle dataset where PGCL can remarkably improve LightGCN by 15.6% on Recall and 17.9% on NDCG with a 4-layer setting. Specifically,

Table 3: PGCL performance in terms of Recall@20 and NDCG@20 on the used datasets. * denotes a significant difference compared to the result of PGCL using the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Dataset	Yelp2018		Gowalla		Amazon-Kindle	
Methods	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
LightGCN	0.0542*	0.0437*	0.0996*	0.0635*	0.2230*	0.1644*
LightGCN$_{RWPE}$	0.0556*	0.0458*	0.1014*	0.0631*	0.2303*	0.1713*
LightGCN$_{LapPE}$	0.0581*	0.0477*	0.1059*	0.0675*	0.2420*	0.1803*
PGCL$_{RWPE}$	0.0583*	0.0486*	0.1068*	0.0674*	0.2451*	0.1815*
PGCL$_{LapPE}$	**0.0608**	**0.0501**	**0.1122**	**0.0699**	**0.2572**	**0.1934**

Table 4: Performance comparison between PGCL and LightGCN at different layers. The peak performance for each method is highlighted in bold.

Dataset		Yelp2018		Gowalla		Amazon-Kindle	
Layers	Methods	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
1 Layer	LightGCN	0.0531	0.0433	0.0982	0.0622	0.2214	0.1635
	PGCL	0.0570 (+7.3%)	0.0477 (+10.2%)	0.109 (+11.0%)	0.0677 (+8.8%)	0.2553 (+15.3%)	0.1917 (+17.2%)
2 Layers	LightGCN	0.0519	0.0421	0.0993	0.0630	0.2225	0.1641
	PGCL	0.0582 (+12.1%)	0.0493 (+17.1%)	0.1106 (+11.4%)	0.0686 (+8.9%)	0.2561 (+15.1%)	0.1921 (+17.1%)
3 Layers	LightGCN	0.0536	0.0435	**0.0996**	**0.0635**	**0.2230**	**0.1644**
	PGCL	0.0580 (+8.2%)	0.0488 (+12.2%)	0.1112 (+11.6%)	0.0692 (+9.0%)	0.2565 (+15.0%)	0.1927 (+17.2%)
4 Layers	LightGCN	**0.0542**	**0.0437**	0.0991	0.0632	0.2224	0.1640
	PGCL	0.0595 (+9.8%)	0.0496 (+13.5%)	0.1115 (+12.5%)	0.0697 (+10.3%)	**0.2572 (+15.6%)**	**0.1934 (+17.9%)**
5 Layers	LightGCN	0.0538	0.0427	0.0987	0.0630	0.2217	0.1637
	PGCL	**0.0608 (+13.0%)**	**0.0501 (+14.6%)**	**0.1122 (+13.7%)**	**0.0699 (+11.0%)**	0.2562 (+15.6%)	0.1925 (+17.6%)

PGCL continues to reach a higher recommendation performance on the Gowalla and Amazon-Kindle datasets with more layers while LightGCN already reaches its peak performance at 3-layer. This result indicates that injecting the learned graph positional encoding (PE) can benefit the general message passing scheme by encoding the graph PE as additional features and can improve the expressive power of LightGCN with an increased models' depth. On the other hand, as can be seen in Table 4, PGCL does not reach its peak performance at the highest layer on the Amazon-Kindle dataset. This observation indicates that our PGCL model tends to suffer from over-smoothing when using a higher number of layers. We leave the investigation of the over-smoothing problem as a future work direction.

To further examine the effectiveness of the graph PE, we conduct a further analysis on the over-smoothness values for both the 2-layer PGCL and all 2-layer baselines. We use the over-smoothness of second-order embedding to evaluate the PGCL's capability of alleviating the over-smoothing problem. Following He et al. [10], we calculate the users' over-smoothness that have an overlap on the interacted items. In particular, as in [10], we use a smoothness metric to evaluate the over-smoothness of the users/items. A higher value indicates less over-smoothing. Similarly we can also obtain the over-smoothness for the item embeddings. Table 5 shows the over-smoothness values of PGCL and the various used baseline models. The results show that our PGCL model obtains the largest O-Smoothness$_u$ and O-Smoothness$_i$ values, which indicate that more effective user/item embeddings are generated with the learned graph PE. Comparing LightGCN and PGCL$_{w/o\,CL}$, we note that the graph positional encoding exhibits a large gain on O-Smoothness$_u$ and O-Smoothness$_i$ while improving the recommendation performance at the same time. We also compare the impact of using the noise-based augmentation in addressing the over-smoothing problem. According to the results in Table 5, PGCL outperforms PGCL$_{w/o\,CL}$ both in over-smoothness and recommendation performance by a large margin, which demonstrates the effectiveness of mining augmented user/item embeddings through contrastive learning. Hence, in answer to RQ3, we conclude that PGCL successfully alleviates the over-smoothing problem by injecting the learned graph positional encoding to each feature propagation layer. This further shows that a graph positional encoding learned with a separate message passing function can lead to a more expressive graph recommender.

Table 5: Over-smoothness comparison of the 2-layer user/item embeddings between PGCL and the baselines. O-Smoothness$_u$ and O-Smoothness$_i$ represent the over-smoothness of users/items, respectively. A higher over-smoothness value indicates less over-smoothing (i.e. a higher value is better).

Dataset	Yelp2018			Gowalla		
Methods	O-Smoothness$_u$↑	O-Smoothness$_i$↑	Recall@20↑	O-Smoothness$_u$↑	O-Smoothness$_i$↑	Recall@20↑
LightGCN	10747.4	8318.5	0.0542	14634.6	6314.2	0.0996
SimGCL	12187.5	9932.3	0.0577	15043.1	6939.1	0.1068
PGCL$_{w/o\,CL}$	13317.8	10177.4	0.0581	15257.4	7192.5	0.1063
PGCL	13978.4	11748.1	**0.0608**	16462.3	7936.8	**0.1122**

5 Conclusions

In this work, we proposed the PGCL model to tackle the over-smoothing problem of graph recommenders by leveraging graph positional encoding. Specifically, we used Laplacian eigenvector as graph positional encoding to endow the user/item embedding in each feature propagation layer. In particular, we updated the learned graph positional encoding with a separated message passing function and merged it with the pre-existing users/items' features. We further encoded users/items' preferences by contrasting the augmented user/item representations with the learned graph positional encodings. Our results on three benchmark datasets showed that PGCL effectively leverages graph positional encoding along with the commonly-used graph recommenders and provides a significant improvement in comparison with the existing baselines. Moreover, we conducted an ablation study to investigate the effect of using different positional encodings for our PGCL model and concluded that the Laplacian eigenvector is more beneficial for the user-item interaction data. Furthermore, we showed that PGCL is more expressive because it can stack more layers with an improved recommendation performance while reducing the over-smoothness of user/item embeddings compared to the baselines.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6) (2003)
2. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: Proceedings of the 37th International Conference on Machine Learning (2020)
3. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699 (2020)
4. Dwivedi, V.P., Joshi, C.K., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982 (2020)
5. Dwivedi, V.P., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Graph neural networks with learnable structural and positional representations. In: Proceedings of the 10th International Conference on Learning Representations (2022)
6. Gao, C., Wang, X., He, X., Li, Y.: Graph neural networks for recommender system. In: Proceedings of the 15th ACM International Conference on Web Search and Data Mining (2022)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. Proceedings of the 4th International Conference on Learning Representations (2015)
8. Grégoire, M., Dexiong, C., Margot, S., Julien, M.: Graphit: Encoding graph structure in transformers. arXiv preprint arXiv:2106.05667 (2021)
9. Haveliwala, T.H.: Topic-sensitive PageRank. In: Proceedings of the 11th International Conference on World Wide Web (2002)
10. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: 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 (2020)
11. KG, S., Sadasivam, G.S.: A survey on personalized recommendation techniques. International Journal on Recent and Innovation Trends in Computing and Communication **2**(6) (2014)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (2014)
13. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018)
14. Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention. Advances in Neural Information Processing Systems **34** (2021)
15. Lee, D., Kang, S., Ju, H., Park, C., Yu, H.: Bootstrapping user and item representations for one-class collaborative filtering. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021)
16. Li, P., Wang, Y., Wang, H., Leskovec, J.: Distance encoding–design provably more powerful GNNs for structural representation learning. arXiv preprint arXiv:2009.00142 (2020)
17. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the 32th AAAI Conference on Artificial Intelligence (2018)
18. Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., Jegelka, S.: Sign and basis invariant networks for spectral graph representation learning. arXiv preprint arXiv:2202.13013 (2022)

19. Liu, S., Ounis, I., Macdonald, C.: An MLP-based algorithm for efficient contrastive graph recommendations. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022)
20. Liu, S., Ounis, I., Macdonald, C., Meng, Z.: A heterogeneous graph neural model for cold-start recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020)
21. Loukas, A.: What graph neural networks cannot learn: depth vs width. arXiv preprint arXiv:1907.03199 (2019)
22. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the 33th AAAI conference on artificial intelligence (2019)
23. Murphy, R., Srinivasan, B., Rao, V., Ribeiro, B.: Relational pooling for graph representations. In: Proceedings of the 36th International Conference on Machine Learning (2019)
24. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
25. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. In: Proceedings of the 36th Conference on Neural Information Processing Systems (2022)
26. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (2009)
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems **30** (2017)
28. Wang, X., Ounis, I., Macdonald, C.: Leveraging review properties for effective recommendation. In: Proceedings of the 30th International Conference on World Wide Web (2021)
29. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (2019)
30. Weisfeiler, B., Leman, A.: The reduction of a graph to canonical form and the algebra which appears therein. NTI, Series **2**(9) (1968)
31. Welling, M., Kipf, T.N.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations (2016)
32. Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., Xie, X.: Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021)
33. Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., Cui, B.: Contrastive learning for sequential recommendation. In: Proceedings of the 38th IEEE International Conference on Data Engineering (2022)
34. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: Proceedings of the 6th International Conference on Learning Representations (2018)
35. Yi, Z., Wang, X., Ounis, I., Macdonald, C.: Multi-modal graph contrastive learning for micro-video recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022)
36. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.Y.: Do transformers really perform badly for graph representation? Advances in Neural Information Processing Systems **34** (2021)

37. You, J., Ying, R., Leskovec, J.: Position-aware graph neural networks. In: Proceedings of the 36th International Conference on Machine Learning (2019)
38. Yu, J., Yin, H., Li, J., Gao, M., Huang, Z., Cui, L.: Enhance social recommendation with adversarial graph convolutional networks. IEEE Transactions on Knowledge and Data Engineering **34** (2020)
39. Yu, J., Yin, H., Li, J., Wang, Q., Hung, N.Q.V., Zhang, X.: Self-supervised multi-channel hypergraph convolutional network for social recommendation. In: Proceedings of the 30th International Conference on World Wide Web (2021)
40. Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., Nguyen, Q.V.H.: Are graph augmentations necessary? simple graph contrastive learning for recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022)
41. Yu, J., Yin, H., Xia, X., Chen, T., Li, J., Huang, Z.: Self-supervised learning for recommender systems: A survey. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022)