information

MDPI

Article

EPCF: An Equivariant Positional Propagation Enhanced Graph Neural Network for Collaborative Filtering

Xin Sun [1], Jishen Sun [1], Li Pang [1], Guiling Wang [2], Zhizhong Liu [3], Xin Liu [4] and Jian Yu [1,*]

[1] School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; xin.sun@autuni.ac.nz (X.S.); jishen.sun@autuni.ac.nz (J.S.); li.pang@autuni.ac.nz (L.P.)
[2] School of Artificial Intelligence and Computer Science, North China University of Technology, Beijing 100144, China; wangguiling@ncut.edu.cn
[3] School of Computer and Control Engineering, Yantai University, Yantai 264005, China; zhizhongliu@ytu.edu.cn
[4] School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; liuxin@cqupt.edu.cn
* Correspondence: jian.yu@aut.ac.nz

Abstract

Graph neural networks (GNNs) have shown great advantages in collaborative filtering recommender systems due to their capacity to model user–item relationships through information propagation. However, traditional GNN-based recommenders often fail to distinguish nodes with the same local structure, leading to identical representations after propagation. Some studies address this issue by introducing positional encoding. However, most existing positional encoding approaches break the permutation and orthogonal symmetries of graph representations and degrade generalization ability. To address this limitation, we propose EPCF (equivariant positional collaborative filtering), a novel GNN model for collaborative filtering that introduces an equivariant propagation mechanism for Laplacian positional features. The proposed mechanism preserves equivariance of positional features under orthogonal transformations while maintaining the permutation equivariance inherent to graphs, which can improve generalization. The equivariant positional features are further leveraged to guide node embedding propagation. Our experiments on real-world datasets show that EPCF achieves better average performance than the evaluated baselines, achieving average improvements of 7.01% in Recall@20 and 1.17% in area under the curve (AUC) over the strongest baselines. Furthermore, integrating EPCF as a plug-in mechanism into five different GNN backbone models achieves improvements of 23.13% in Recall@20 and 2.14% in AUC across five datasets, demonstrating its generalization capability.

Keywords: recommender systems; collaborative filtering; graph neural networks; positional encoding; equivariance

Academic Editor: Mohd Shareduwan Mohd Kasihmuddin

Received: 20 May 2026
Revised: 25 June 2026
Accepted: 29 June 2026
Published: 1 July 2026

1. Introduction

Graph neural networks (GNNs) have emerged as a widely used model for collaborative filtering recommender systems by modeling user–item interactions as graphs and learning representations through message passing. By propagating information over interaction graphs, GNNs can effectively capture collaborative signals and achieve strong performance in various recommendation tasks. Existing GNN-based recommender systems often adopt graph convolutional network (GCN), LightGCN, attention-based GNNs,

and other message passing architectures to learn user and item representations from interaction graphs [1–3].

While GNN-based recommender systems are effective in modeling user–item interactions, they often overlook a fundamental limitation that node identities are lost during the intrinsic computation of GNNs [4]: nodes that share the same local subgraph structure will be associated with the same representation by GNNs, and thus they are indistinguishable. For instance, as illustrated in Figure 1a, nodes h and i share the same local structure within two hops from node a and lose their global distinctiveness after propagation. This issue could be more severe for collaborative filtering tasks because many users or items share similar local interaction patterns in recommender systems.

Recognizing these limitations, researchers have turned to global structural information as a complementary signal. For instance, random walks have been introduced as a kind of global structural information for GNNs [5] inspired by the previous work DeepWalk [6]. It captures global information by estimating node distances to a set of randomly selected anchor nodes. However, this approach suffers from slow convergence and delivers only moderate performance. Other methods attempt to capture structural information using heat kernels [7], subgraphs [8,9] and shortest path distances [10]. Nevertheless, these methods are often computationally expensive and memory-intensive.

To seek a more efficient alternative to these heavy structural descriptors, some studies have explored the absolute node positions in the graph as additional node features. A common choice is to use graph Laplacian eigenvectors as absolute positions for positional encoding [11,12]. However, incorporating Laplacian eigenvectors as positional encoding introduces additional ambiguities. Although Laplacian eigenspaces transform consistently under graph relabeling, the eigenvector basis itself is not unique. Eigenvectors associated with simple eigenvalues are defined only up to sign flips, while repeated eigenvalues admit arbitrary orthogonal basis transformations within the corresponding eigenspace. As a result, positional encoding constructed from a fixed eigenvector basis may become unstable under graph relabeling, numerical eigendecomposition, or changes of basis. Therefore, designing propagation mechanisms that remain consistent under such orthogonal transformations remains an important challenge. Graph structured data should respect symmetry properties under group actions on graphs such as node permutations and orthogonal transformations. These symmetries reflect the fact that the semantic meaning of a graph remains unchanged under re-indexing of nodes or transformations of coordinate systems. Accordingly, GNNs are expected to be equivariant to such transformations, which means applying a transformation to the input should result in a consistent transformation of the output representations. For instance, as illustrated in Figure 1c, applying an orthogonal transformation to the positional feature at a previous layer should result in the same orthogonal transformation to the propagated positional features at the current layer. However, absolute positional encoding derived from Laplacian eigenvectors does not satisfy these equivariance requirements. Directly injecting such positional encoding breaks the permutation and orthogonal symmetries that GNNs are expected to respect. This may lead to inconsistent representations and degraded generalization. Therefore, ensuring that positional features are propagated in a manner that remains equivariant to the underlying group actions remains a challenge. Despite the appeal of Laplacian positional encoding, an effective and principled approach for incorporating them into recommender systems in an equivariant manner remains unexplored.

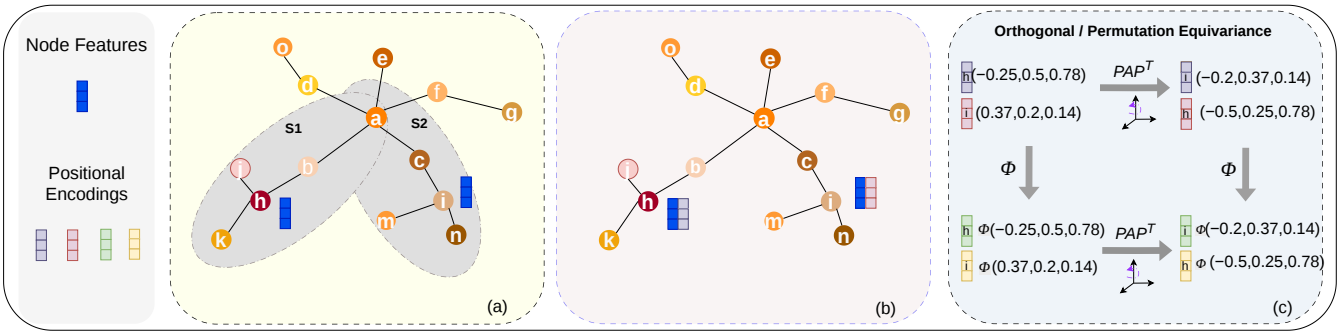

Figure 1. Positional features help distinguish nodes with identical local structure (node embeddings with the same color represent indistinguishable representations): (**a**) To avoid oversmoothing, most GNNs (graph neural networks) use only 2–3 layers of message passing. As a result, in a two-layer GNN, nodes h and i aggregate information only from their two hop neighborhoods. Since $S_1 \cong S_2$, nodes h and i become structurally indistinguishable after message passing. (**b**) By incorporating positional encoding as additional information, nodes h and i can be distinguished. (**c**) An example of using Laplacian eigenvectors as positional encoding for nodes h and i: Permutation matrix P re-indexes the graph and orthogonal transformations rotate or reflect the positional feature space. Under these transformations, the GNN layer ϕ needs to keep the positional features equivariant.

To address this issue, we propose EPCF (equivariant positional collaborative filtering), an equivariant GNN framework for collaborative filtering that leverages Laplacian eigenvectors as positional features and enables their equivariant propagation under permutation and orthogonal transformations. Unlike existing recommendation models that mainly use positional encoding as static structural information, EPCF propagates positional representations throughout message passing while preserving the symmetry properties of graph representations. We further investigate the role of equivariant positional propagation in recommendation and evaluate its effectiveness on multiple benchmark datasets. Our experimental results demonstrate that EPCF consistently improves recommendation performance and can be effectively integrated into different GNN-based recommender backbones, indicating its applicability across diverse recommendation architectures.

We summarize the main contributions of this work as follows:

- We propose an equivariant mechanism for Laplacian positional features in GNN-based collaborative filtering. Based on this mechanism, we develop EPCF, an equivariant graph neural network that propagates positional representations throughout message passing while preserving equivariance under permutation and orthogonal transformations.
- We theoretically analyze the equivariance properties of the proposed propagation mechanism and provide proofs that EPCF preserves equivariance under permutation and orthogonal transformations at each propagation layer. We further discuss how equivariant positional propagation contributes to recommendation performance.
- We conducted experiments on multiple real-world datasets to evaluate the performance of EPCF. Comparisons with several GNN-based methods on three real-world datasets showed that EPCF achieved better performance than the other baselines. To further assess its generality, we integrated EPCF into different types of GNN-based recommender systems. Our experimental results on five real-world datasets demonstrated that EPCF improved the average performance of these backbones, exhibiting its generalization ability as a plug-in mechanism.

The rest of this paper is organized as follows. Section 2 reviews the related literature. Section 3 introduces the preliminaries. Section 4 presents the proposed EPCF framework. Section 5 reports the experimental results. Section 6 provides further discussion and analysis of the proposed method. Finally, Section 7 concludes this work and outlines future research directions.

2. Related Work

In this section, we review the evolution of GNN-based recommender systems, positional encoding methods in recommender systems and equivariant graph neural networks. We then discuss the research gap that motivated our work.

2.1. GNN-Based Recommender Systems

GNNs are widely used in recommender systems because user–item interactions can be naturally represented as a graph where connectivity is explicitly captured through message passing. Early GNN approaches for recommendation typically used the GCN framework [13] to perform message passing by aggregating information from neighboring nodes [1,14,15]. They employed feature transformation, neighborhood aggregation and non-linear activation in each propagation layer. However, recognizing that these operations often made the propagation step very complicated in terms of training and scalability [16], subsequent works [2,17,18] adopted LightGCN [16], which streamlines the propagation by removing feature transformation, activation functions and self-connections. This simplification reduces computational cost while preserving competitive performance. To better capture differences in user preferences for items and the varying influence of neighbors, some approaches have introduced attention mechanisms or Transformer into GNN-based recommenders [3,19,20]. These methods compute attention weights using different strategies to selectively aggregate neighbors' embeddings. For instance, DANSER [21] and SCGRec [22] adopt concatenation-based attention, while DICER [23] and MEGCN [24] rely on similarity-based attention to modulate message propagation.

Besides modeling static user–item interactions, subsequent research has extended GNN architectures to capture temporal evolution and complex high-order dependencies. To model the evolution of user preferences over time, temporal recommender systems have gained attention. GNN-DSR [25] and GNNRec [26] model dynamic user interests using graph recurrent neural network encoders, capturing time-evolving preferences through session-based or temporal sequences. At the same time, hypergraph-based approaches have been proposed to explicitly model group-wise interactions beyond pairwise edges. For instance, DH-HGCN [27] and MHCN [28] focus on connections among hypergraph inputs. These methods use HyperGNN encoders to learn user and item embeddings. Specifically, MHCN [28] introduces three types of triangular motifs, constructing three incidence matrices, each representing a hypergraph induced by a specific motif, and it learns user embeddings using multi-type HyperGNN encoders. Similarly, to model multiple types of nodes and relations in real-world platforms heterogeneous GNNs like HNGCL [29] and MixRec [30] have been developed to encode complex relational structures.

Even with these extensions, most of them still rely on local neighborhood aggregation mechanisms. This inherent reliance restricts their ability to distinguish structurally isomorphic nodes, which limits their representation ability.

2.2. Positional and Structural Encoding in GNN-Based Recommender Systems

Most GNN-based recommender systems focus on neighborhood aggregation. Although positional encoding has become an important component in graph learning, its use in recommendation is still limited. PGCL [31] introduces Laplacian and random walk

positional encoding into graph contrastive recommendation. GFormer [32] incorporates anchor distance information to preserve global topological structure. SHT [33] extends transformer to hypergraph recommendation for capturing higher-order collaborative relations. PGTR [34] designs positional encoding for recommender, including spectral, degree, PageRank and type encoding, then integrates them into a graph Transformer framework for long-range collaborative modeling.

These studies show that positional and structural information can be beneficial for recommendation. However, existing methods generally use positional encoding as additional features. They do not consider the intrinsic ambiguity of Laplacian positional encoding, where equivalent representations may arise from sign flips or orthogonal transformations of eigenvectors.

2.3. Equivariance in GNNs

Although absolute positional encoding helps distinguish structurally similar nodes, it introduces an additional difficulty. The same graph may admit multiple equivalent positional representations due to node permutations or orthogonal transformations of the positional coordinates. Without equivariance, these equivalent representations can lead to inconsistent model outputs.

Several studies have addressed equivariance through randomized positional representations [12,35]. Srinivasan et al. [35] computed eigenvectors from randomly permuted Laplacian matrices. SAN [12] addressed the sign ambiguity of Laplacian eigenvectors by randomly perturbing their signs during training. Although these methods improve robustness to symmetry ambiguities, they do not guarantee permutation equivariance when the matrix contains multiple eigenvalues. In addition, they were primarily developed for settings where geometric coordinates or predefined positional information are available, which differs from recommender systems where positional information must be inferred from graph structure.

Other works [4,36,37] leverage scalarization-based methods to achieve equivariance. They first translate positional encoding into invariant scalars like relative distances or angles and then utilize processed scalars for the update of equivariant features. EGNN [36] studies equivariance when nodes have physical coordinates given in advance. It updates node features and positional information separately, ensuring that node features remain invariant while positional information remains equivariant under permutation and orthogonal transformations.

More recent studies [38–40] have combined positional encoding with equivariance. EigenMLP [39] learns sign-invariant and basis-invariant representations from Laplacian eigenvectors. It is permutation equivariant but invariant to rotation and reflection. OGE-Aug [41] proposes a method that uses a series of masks to filter Laplacian eigenvectors belonging to different eigenspaces, generating globally expressive graph representations while maintaining orthogonal group equivariance. OAP [42] develops a canonicalization framework that unifies equivariant and invariant learning for eigenvectors, providing a novel perspective on ensuring stability and expressiveness in graph representations. Most of these approaches focus on molecular property prediction tasks in the context of graph classification and have not been adapted for recommender systems or link prediction tasks. EDEN [38] generalizes cosine positional encoding to non-Euclidean spaces via phase propagation and employs mathematical transformations to ensure permutation equivariance. It can be applied to both graph classification and link prediction tasks. PEG [4] introduces positional encoding that are equivariant to permutations, orthogonal transformations and reflections. It re-weights the graph structure using pairwise distances derived from Laplacian eigenvectors. Although PEG has been applied to link prediction

tasks, it does not propagate positional information across layers, which may limit its expressive power.

Most existing equivariant positional encoding methods are designed for molecular or graph classification tasks and often assume geometric coordinates, while recommender systems operate on relational graphs without such information. The integration of positional information with equivariant propagation in recommendation remains unexplored, motivating our work.

3. Preliminaries

3.1. User–Item Bipartite Graph

A bipartite graph is denoted as $G = (U, V, E)$. The node set is partitioned into two disjoint subsets U and V, where $U \cap V = \emptyset$. U represents users and V represents items. Edge $e = (u, v) \in E$ connects a user u to an item v. The interaction matrix $R \in \mathbb{R}^{|U|\times|V|}$ stores user–item interactions. Specifically, $R_{ui} = 1$ indicates that user u has interacted with item i and $R_{ui} = 0$ otherwise.

3.2. Laplacian Positional Encoding

We denote $A \in \mathbb{R}^{N\times N}$ as the adjacency matrix of the graph G. The diagonal degree matrix D is defined by $D_{ii} = \sum_j A_{ij}$. The normalized Laplacian matrix of graph G is defined as $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. The Laplacian L is real symmetric and positive semi-definite. Therefore, L has eigenvalue decomposition $L = U\Lambda U^\top$, where Λ is a real diagonal matrix with the eigenvalues of L, $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$ as its diagonal components. The matrix $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ is orthogonal, where each $\mathbf{u}_i \in \mathbb{R}^N$ is the i-th eigenvector, i.e., $L\mathbf{u}_i = \lambda_i \mathbf{u}_i$.

For connected graphs, the smallest eigenvalue satisfies $\lambda_1 = 0$ and the corresponding eigenvector $\mathbf{u}_1$ is constant over all nodes, carrying no useful positional information. Therefore, LapPE (Laplacian positional encoding) is commonly constructed from the eigenvectors associated with the smallest non-trivial eigenvalues. For disconnected graphs, the multiplicity of the zero eigenvalue equals the number of connected components. Let c denote this multiplicity. In this case, we discard the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_c$ associated with $\lambda_1 = \cdots = \lambda_c = 0$, and we define LapPE as LapPE $= [\mathbf{u}_{c+1}, \mathbf{u}_{c+2}, \ldots, \mathbf{u}_{c+k}]$, where $\lambda_{c+1}, \ldots, \lambda_{c+k}$ are the smallest positive eigenvalues.

The eigendecomposition of the Laplacian is generally not unique. If an eigenvalue λ_i is simple, i.e., $\lambda_i \neq \lambda_j$ for all $j \neq i$ then the corresponding eigenvector is unique only up to sign, meaning that both $\mathbf{u}_i$ and $-\mathbf{u}_i$ are valid eigenvectors. If an eigenvalue has multiplicity m, i.e., $\lambda_i = \lambda_{i+1} = \cdots = \lambda_{i+m-1}$ then the corresponding eigenspace admits multiple valid orthonormal bases. In particular, the basis $[\mathbf{u}_i, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_{i+m-1}]$ can be replaced by $[\mathbf{u}_i, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_{i+m-1}]Q$ for any orthogonal matrix $Q \in O(m)$, while representing the same eigenspace.

The positional feature matrix Z should be interpreted as a representation of the underlying eigenspace rather than a specific eigenvector basis. Throughout this work, orthogonal equivariance is defined with respect to the action $Z \mapsto ZQ$, where $Q \in O(k)$. This formulation naturally accounts for sign flips and arbitrary orthogonal basis transformations within degenerate eigenspaces. Therefore, the equivariance analysis presented in Section 4.2 applies not only to orthogonal transformations of positional coordinates but also to the intrinsic ambiguities of Laplacian eigenvectors.

3.3. Equivariance

We denote $X \in \mathbb{R}^{N\times d}$ as the node embedding matrix and $Z \in \mathbb{R}^{N\times k}$ as the positional feature matrix, where N is the number of nodes, d is the embedding dimension and k is the positional dimension. $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix. A graph propagation function is equivariant if a transformation applied to the input leads to the corresponding transformation of the output.

3.3.1. Permutation Equivariance

For a permutation matrix $P \in \{0,1\}^{N\times N}$, permutation equivariance requires that

$$(PX', PZ') = F(PAP^\top, PX, PZ),$$

where $(X', Z') = F(A, X, Z)$. In other words, reordering the nodes before propagation produces the same result as reordering the propagated representations after propagation.

3.3.2. Orthogonal Equivariance

Orthogonal transformations of positional features are represented by matrices $Q \in O(k)$, where

$$O(k) = \{Q \in \mathbb{R}^{k\times k} \mid Q^\top Q = QQ^\top = I\}.$$

Following the row vector convention, an orthogonal transformation acts on the positional feature matrix as ZQ. Orthogonal equivariance is satisfied if

$$(X', Z'Q) = F(A, X, ZQ).$$

This property ensures that transforming the positional coordinate basis before propagation leads to the same transformation of the propagated positional features.

4. Methodology

In this section, we first introduce the design of our proposed Equivariant Positional Propagation in Section 4.1. We subsequently analyze how EPCF preserves equivariance under orthogonal transformations and permutations on positional encoding in Section 4.2.

4.1. The Equivariant Positional Propagation

Our proposed EPCF builds on Laplacian positional encoding and consists of three core components: edge feature computation, equivariant position propagation and node embedding propagation. Figure 2 illustrates the overall architecture of the proposed EPCF model. We first compute edge features from the Euclidean distances between Laplacian positional encoding and the learned attention coefficients. These edge features are then used for both equivariant position propagation and node embedding propagation.

Following the notation from Section 3, we consider a bipartite graph $G = (U, V, E)$, where U represents the set of users, V represents the set of items and E represents the edges denoting user–item interactions. For a user $u \in U$, an item $i \in V$ and their interaction edge $e_{ui} \in E$, we define user embeddings as $\mathbf{x}_u \in \mathbb{R}^{1\times d}$ and item embeddings as $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ to capture the latent representations of nodes. The adjacency matrix of the user–item bipartite graph is represented as $A = \begin{bmatrix} 0 & R \\ R^\top & 0 \end{bmatrix}$, where $R \in \mathbb{R}^{|U|\times|V|}$ is the user–item interaction matrix, with $R_{ui} = 1$ if $(u, i) \in E$ and 0 otherwise. Positional encoding for users and items in bipartite graph G are initialized from the eigenvectors of the normalized Laplacian matrix L, where the positions are denoted as $\mathbf{z}_u \in \mathbb{R}^{1\times k}$ for users and $\mathbf{z}_i \in \mathbb{R}^{1\times k}$ for items.

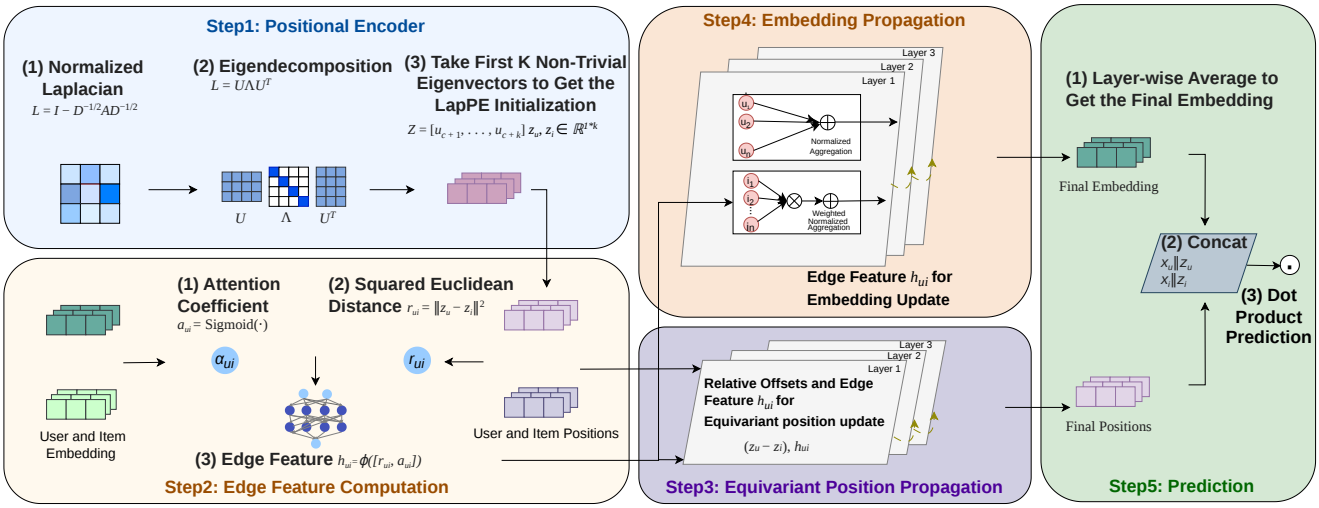

Figure 2. The overall architecture of the proposed model. The positional encoder generates positional features, which are used for edge feature computation. The resulting edge features jointly guide equivariant position propagation and embedding propagation, leading to the final prediction.

4.1.1. Edge Feature Computation

The edge features h_{ui} between user u and item i are scalars derived from their positional relative squared Euclidean distance r_{ui} and the attention coefficient $\tilde{a}_{ui}$, given by

$$h_{ui} = \phi\left(\begin{bmatrix} r_{ui} & \tilde{a}_{ui} \end{bmatrix}\right). \tag{1}$$

The relative squared distance is defined as $r_{ui} = \|\mathbf{z}_u - \mathbf{z}_i\|^2$, where $\|\cdot\|$ denotes the Euclidean norm. The function $\phi : \mathbb{R}^2 \to \mathbb{R}$ is implemented as an MLP (multilayer perceptron) that maps the 2-dimensional input to a scalar. The attention coefficient $\tilde{a}_{ui}$ is computed as

$$\mathbf{s}_u = \mathbf{x}_u W_1, \quad \mathbf{s}_i = \mathbf{x}_i W_2, \quad \tilde{a}_{ui} = \sigma(\text{LeakyReLU}([\mathbf{s}_u \parallel \mathbf{s}_i]\mathbf{a})), \tag{2}$$

where W_1 and W_2 are learnable weight matrices, $\mathbf{a}$ is a parameter vector, σ is the sigmoid activation and $\parallel$ denotes concatenation.

The edge feature h_{ui} later serves as the propagation weight in both positional and embedding updates. Since r_{ui} is invariant under orthogonal transformations of positional features and $\tilde{a}_{ui}$ depends only on node embeddings, h_{ui} is invariant under orthogonal transformations of Z. Under type-preserving permutations, r_{ui}, $\tilde{a}_{ui}$, and h_{ui} are re-indexed consistently.

4.1.2. Equivariant Position Propagation

Based on the learned edge features, we design a propagation mechanism for updating node positions. To strictly preserve equivariance to orthogonal transformations and permutations, each user or item position is propagated by adding the weighted sum of all relative differences $(\mathbf{z}_u - \mathbf{z}_i)$ to the previous layer's position. The propagation weight h_{ui} is not constrained to be positive. Consequently, the positional update may either attract neighboring nodes when $h_{ui} < 0$ or repel neighboring nodes when $h_{ui} > 0$. This design allows the model to adaptively learn both similarity-preserving and separation-promoting structural relations from the graph. The scalar edge feature h_{ui} defined in Equation (1) is

directly used as the propagation weight in Equations (3) and (4). The positional feature $\mathbf{z}$ at layer $l+1$ is updated as

$$\mathbf{z}_u^{(l+1)} = \mathbf{z}_u^{(l)} + \frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} h_{ui} \left(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)} \right), \tag{3}$$

$$\mathbf{z}_i^{(l+1)} = \mathbf{z}_i^{(l)} + \frac{1}{|\mathcal{N}_i|} \sum_{u \in \mathcal{N}_i} h_{ui} \left(\mathbf{z}_i^{(l)} - \mathbf{z}_u^{(l)} \right), \tag{4}$$

where $\mathcal{N}_u$ is the set of neighbors of user u, and where $\mathcal{N}_i$ is the set of neighbors of item i. This formulation ensures that the position propagation is equivariant under permutations and orthogonal transformations. The analysis is provided in Section 4.2.

4.1.3. Node Embedding Propagation

Embedding propagation leverages edge features for enhanced aggregation. The user embeddings $\mathbf{x}_u$ and the item embeddings $\mathbf{x}_i$ at layer $l+1$ are updated as

$$\mathbf{x}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{x}_i^{(l)}, \tag{5}$$

$$\mathbf{x}_i^{(l+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \left(\sum_{t \in \mathcal{N}_u} \frac{h_{ut}}{|\mathcal{N}_t|} \mathbf{x}_t^{(l)} \right), \tag{6}$$

where the user embedding $\mathbf{x}_u^{(l+1)}$ aggregates information from its neighboring items $i \in \mathcal{N}_u$. For item propagation, each item i receives information from its neighboring users $u \in \mathcal{N}_i$ and u's neighboring items $t \in \mathcal{N}_u$ with the positional edge feature h_{ut}. This gives a two-hop propagation path and allows item representations to be refined by other items that are connected through shared users.

We apply the edge feature h only in the item propagation step to avoid repeatedly injecting the same positional signal within one layer. After item representations are updated with the positional edge features, the updated item embeddings will be propagated to users in the next layer through Equation (5). Thus, the positional information can still influence both user and item representations across layers. The same design can also be applied by swapping the roles of users and items, but in our implementation we choose the item-side update because recommendation prediction is ultimately performed by matching users with candidate items. The degree normalization terms $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ and $1/|\mathcal{N}_t|$ are used to reduce the influence of highly connected users or items and make the aggregation more stable on imbalanced bipartite graphs.

4.1.4. Layer-Wise Embedding Averaging and Prediction

The final embedding $\mathbf{x}'$ is obtained by averaging the propagated embeddings across layers

$$\mathbf{x}' = \frac{1}{L} \sum_{l=1}^{L} \left(\mathbf{x}^{(l)} \right), \tag{7}$$

where L is the number of layers. The predicted preference score is computed as the dot product

$$\hat{y}_{ui} = (\mathbf{x}'_u \parallel \mathbf{z}_u)(\mathbf{x}'_i \parallel \mathbf{z}_i)^\top. \tag{8}$$

4.1.5. Loss Function

We employ the Bayesian Personalized Ranking (BPR) loss. This is a pairwise loss function that encourages the prediction score of a positive item to be higher than that of a

negative item. For a batch of triplets $\{(u, i, j)\}$, where i is a positive item and j a negative item, the BPR loss terms are

$$\mathcal{L}_{\text{BPR}} = \frac{1}{B} \sum_{(u,i,j)} \text{softplus}(\hat{y}_{uj} - \hat{y}_{ui}), \tag{9}$$

where B denotes the batch size.

4.2. Equivariance Analysis of EPCF

We analyze how EPCF preserves equivariance under permutations and orthogonal transformations. In the context of Laplacian positional encoding, orthogonal transformations include sign flips of individual eigenvectors and arbitrary orthogonal basis transformations within eigenspaces associated with repeated eigenvalues. Therefore, proving orthogonal equivariance also establishes robustness against the non-uniqueness of Laplacian eigenvector representations. Using the notation introduced in Section 3.3, let $X \in \mathbb{R}^{N \times d}$ denote the node embedding matrix and $Z \in \mathbb{R}^{N \times k}$ denote the positional feature matrix. Let $(X^{(l+1)}, Z^{(l+1)}) = EPCF(A, X^{(l)}, Z^{(l)})$ represent one propagation layer of EPCF.

To preserve the equivariance properties of graph representations, EPCF is expected to satisfy the following equivariance conditions:

$$(PX^{(l+1)}, PZ^{(l+1)}Q) = EPCF(PAP^{\top}, PX^{(l)}, PZ^{(l)}Q), \quad \forall P \in \Pi(N),\ \forall Q \in O(k), \tag{10}$$

where P is a permutation matrix and Q is an orthogonal matrix. $P = P_U \oplus P_I$ is a type-preserving permutation matrix, with P_U and P_I acting on user and item nodes respectively. Since EPCF employs distinct transformations W_1 and W_2 for users and items, the equivariance analysis is restricted to type-preserving permutations that do not exchange user and item nodes. According to the properties of h_{ui} discussed in Section 4.1.1, the edge feature h_{ui} is invariant under orthogonal transformations of positional features and is re-indexed consistently under type-preserving permutations.

For the node embeddings X, we use the edge feature h_{ui} for embedding aggregation from their neighbors as defined in Equation (5). Under a type-preserving permutation $P = P_U \oplus P_I$, the node embeddings and graph structure are re-indexed consistently while preserving user and item identities. Therefore, the propagation of node embeddings is permutation-equivariant.

For the positional features Z, since h_{ui} is invariant under type-preserving permutations and orthogonal transformations as discussed above, Equations (3) and (4) compute $\mathbf{z}_i^{(l+1)}$ as a weighted sum of relative offsets $(\mathbf{z}_u - \mathbf{z}_i)$, which is then added to the positional feature at layer l. Under a type-preserving permutation, these relative offsets are re-indexed consistently with the graph structure, resulting in the transformed output PZ'. Under an orthogonal transformation $Z \mapsto ZQ$, the relative offsets transform as $(\mathbf{z}_u - \mathbf{z}_i)Q$, while the scalar weights h_{ui} remain unchanged. Therefore, the updated positional features transform as $Z'Q$, preserving both permutation and orthogonal equivariance. The mathematical derivation is provided in Appendix A.

5. Experiments

We conducted two types of experiments to evaluate our method: horizontal comparison and vertical enhancement. We first describe the experimental settings in Section 5.1. We conducted a horizontal comparison with LightGCN and other representative equivariant GNN baselines, as outlined in Section 5.2. We evaluated vertical enhancement by applying our method as a mechanism and integrating it into different GNN backbones, as outlined

in Section 5.3. We present a confidence interval (CI) analysis in Section 5.4. We performed ablation studies and hyperparameter analysis, as outlined in Sections 5.5 and 5.6.

5.1. Experimental Settings

In this work, we conducted our experiments on a graphics processing unit (GPU) with 32 GB of random access memory (RAM) and a 12-core virtual central processing unit (vCPU) Intel(R) Xeon(R) Platinum 8352V processor @ 2.10 GHz. The implementation was developed in Python 3.12 using PyTorch 2.3.0.

5.1.1. Datasets

As outlined in Section 5.2, we conducted our experiments on three widely used public benchmark datasets: LastFM, Douban-Book and Tmall. As outlined in Section 5.3, we used Twitch, Douban-Book, ProgrammableWeb and Yelp datasets to evaluate the performance of the proposed method. LastFM: A music recommendation dataset that captures user interactions with artists or songs. Douban-Book: Collected from a Chinese social platform, this dataset records user ratings and interactions with books and may also include social relations among users. Tmall: Derived from a Chinese e-commerce platform, it contains user interactions with products, representing a typical online shopping recommendation scenario. Twitch: A live streaming platform dataset capturing user interactions with channels. In our experiments, we specifically used two regional subsets: the ES (Spanish-language) dataset and the RU (Russian-language) dataset. ProgrammableWeb: A dataset of services and APIs drawn from the ProgrammableWeb directory. Yelp: A business review platform dataset capturing users interacting with service providers. Table 1 summarizes the statistics of all datasets used in our experiments, including the number of users, items, interactions and graph density.

Table 1. Statistics and sources of the datasets used in our experiments.

Dataset	# Users	# Items	# Interactions	Density	Source
LastFM	1892	17,632	92,834	0.00278	HetRec 2011 LastFM Dataset
Douban-Book	12,859	22,294	598,420	0.00209	Douban-Book Dataset
Tmall	47,939	41,390	2,357,450	0.00030	Tmall Recommendation Dataset
Twitch-RU	43,904	6398	312,779	0.00111	Twitch Social Network Dataset (RU)
Twitch-ES	46,048	6016	353,770	0.00128	Twitch Social Network Dataset (ES)
ProgrammableWeb	13,545	13,077	2,063,341	0.01165	ProgrammableWeb Mashup Dataset
Yelp	29,601	24,734	1,517,326	0.00051	Yelp Open Dataset

The datasets were obtained from publicly available benchmark repositories commonly used in recommender systems research. The sources are provided in Table 1. To initialize node representations, we employed random feature embeddings instead of predefined node features, enabling the model to effectively learn latent user–item relationships in the absence of explicit feature data. For datasets containing explicit ratings, all observed interactions were binarized and treated as positive feedback. We split user–item interactions into training and test sets with a ratio of 8:2. All our reported results were evaluated on the test set.

5.1.2. Baselines

As outlined in Section 5.2, we chose three baselines for performance comparison: LightGCN [16], EGNN [36], PEG [4]. LightGCN focuses solely on neighborhood aggregation for collaborative filtering tasks in recommender systems. EGNN and PEG are two equivariant GNNs. EGNN uses separate channels to update the original node features and positional features to keep position features equivariant through propagation. PEG

adopts positional encoding techniques and utilizes a weighted GCN framework to impose equivariance for link prediction tasks. These baselines include both recommender systems and equivariant GNNs that leverage positional features.

As outlined in Section 5.3, we apply our method as a mechanism integrated into five representative GNN backbones: GCN [13], GAT [43], FAGCN [44], XSimGCL [45], and Differmer [46]. These backbones span a diverse range of GNN architectures, including spectral convolution (GCN, FAGCN), attention-based propagation (GAT), graph contrastive learning (XSimGCL), and diffusion-inspired message passing (Differmer). By covering this breadth of architectures, our evaluation ensured that the proposed method is not restricted to a particular model type: it enabled a comprehensive investigation into how positional encoding and equivariant propagation interact with spectral, attention, contrastive and diffusion-based GNN methods.

5.1.3. Parameter Settings

As outlined in Section 5.2, we first tested the performance of our model and LightGCN when the embedding dimensions were in the range of 8, 16, 32, 64. We then conducted a comparison with other baselines with an embedding dimension of 32. We used 32-dimensional Laplacian eigenvectors as positional features. We used the default learning rate of 0.001 and the default mini-batch size of 1024. The L2 regularization coefficient λ was searched in the range $\{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$. We used 3 layers on the LastFM and Tmall datasets and 2 layers on the Douban-Book dataset. As outlined in Section 5.3, we conducted comparisons using 16-dimensional embeddings and 16-dimensional Laplacian eigenvectors, using two layers and 100 epochs for all the datasets. The other parameter settings remain the same as in Section 5.2. One negative item was uniformly sampled from the unobserved items of each user during training. We trained for 100 epochs until convergence. Each experiment was repeated five times using different random seeds. We report the mean performance and standard deviation over these runs.

5.1.4. Evaluation Metrics

We evaluated the models' performance using multiple metrics, including Recall@20 and area under the curve (AUC). Recall focuses on how many of the user's preferred items are successfully predicted, aiming to ensure that the user's interests are effectively captured. AUC measures whether the items that users prefer are consistently scored higher than the ones they are not interested in. These two evaluation metrics are widely used in recommender systems. For AUC, we adopted a ranking-based protocol. For each test user, training positives were filtered from the candidate set before ranking. We computed AUC using the top-K ranked items, where $K = \max(\texttt{topks})$: items in the top-K list were assigned decreasing scores according to their ranks and all other candidate items were assigned zero scores. No sampled negative evaluation set was used and the same candidate construction was applied to all the models.

5.2. Horizontal Comparison with Other Methods

We first compared the performance with LightGCN across embedding dimensions (8, 16, 32, 64). We then compared EPCF with two equivariant GNN baselines at an embedding dimension of 32.

5.2.1. Comparison with LightGCN Under Different Embedding Dimensions

We conducted the comparison with LightGCN by evaluating performance under different embedding dimensions (8, 16, 32, 64). We present the Recall@20 and AUC results in Figure 3; the key observations are summarized as follows:

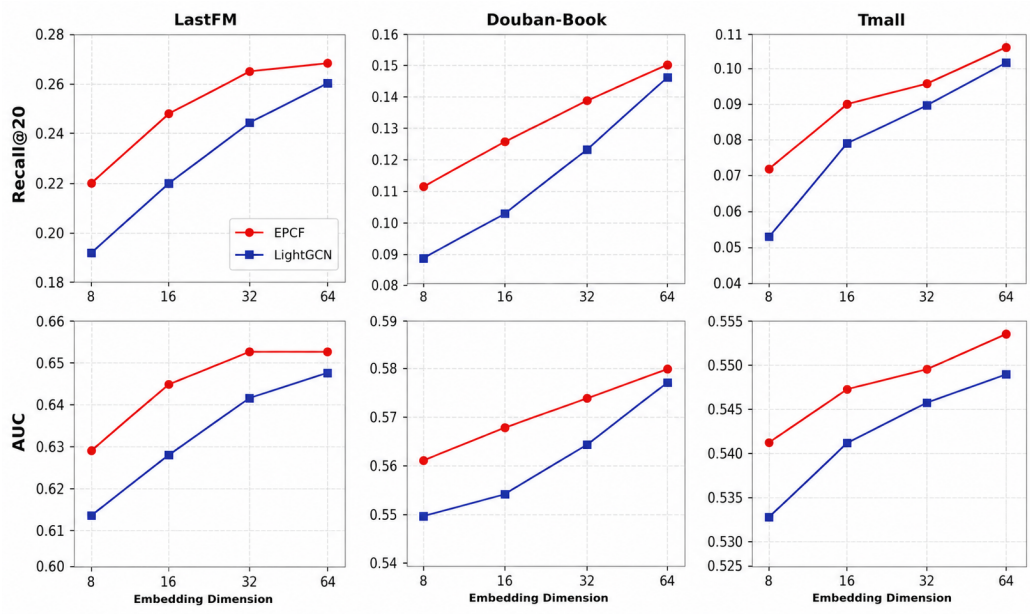

Figure 3. Performance comparison of EPCF and LightGCN under different embedding dimensions.

EPCF achieved higher mean Recall@20 and AUC than LightGCN in these settings. This suggests that EPCF captures richer user–item information and achieves better recommendation performance. The performance advantage of our model over LightGCN was most pronounced at 8 embedding dimensions. As the embedding dimension increased, the performance of both models improved, while the advantage of our model diminished. The Recall advantage of our model over LightGCN on the LastFM dataset dropped from 14.99% at 8 dimensions to just 2.57% at 64 dimensions, on the Douban-Book dataset from 25.8% to 1.45%, and on the Tmall dataset from 37.38% to 0.39%. At 32 dimensions, our model still maintained a clear advantage, while at 64 dimensions the advantage became smaller. This suggests that our model is more useful when the embedding dimension is relatively small, although it can still contribute useful information at 64 dimensions.

5.2.2. Comparison with Other Equivariant GNN Baselines

We compared our proposed model with two equivariant baselines, EGNN and PEG. EGNN provides a representative equivariant message-passing architecture that explicitly updates both node features and positional representations while preserving geometric equivariance. Although originally designed for geometric learning tasks, EGNN served as a relevant baseline because the main contribution of EPCF is also an equivariant positional propagation mechanism. PEG is designed for link prediction tasks and incorporates positional encoding while preserving permutation and orthogonal equivariance. Since recommendation can be viewed as a link prediction task on user–item bipartite graphs, PEG served as a relevant baseline for assessing the impact of equivariant positional information. These baselines covered both the collaborative filtering methods (LightGCN) and the representative equivariant graph learning models (EGNN and PEG). This allowed us to evaluate EPCF from both recommendation and equivariant perspectives. For EGNN and PEG, we adapted the original models to the user–item recommendation setting following the same protocol as EPCF. User and item IDs were initialized with trainable embeddings of dimension 32. Laplacian positional encoding with the same dimensionality were used as positional inputs whenever required by the original model design. For recommendation, the final user and item representations were scored using an inner product and optimized with the same BPR loss and negative sampling strategy used by EPCF. The number of propagation layers, embedding dimension, learning rate, regularization range, and training

epochs were tuned under the same search space as EPCF to ensure a fair comparison. Official implementations were used whenever available, with only specific task modifications required for bipartite recommendation.

We conducted the comparison using the 32-dimensional embedding setting. As summarized in Table 2, EPCF achieved higher Recall@20 and AUC than EGNN and PEG on the three datasets. For Recall@20, the relative improvements over the strongest baseline were 8.20% on LastFM, 8.61% on Douban-Book, and 4.23% on Tmall, giving an average improvement of 7.01%. For AUC, the corresponding improvements were 1.86%, 1.61% and 0.37%, with an average of 1.17%.

EGNN uses MLPs to update node features, which limits its capacity to aggregate relational information between nodes in recommender systems. PEG employs positional differences as weights to propagate embeddings to maintain equivariance, whereas it does not update the positional features across layers. Our model updates positional information equivariantly at each layer and propagates relational information between nodes using weights that depend on positional features, enabling richer representations that better capture complex relational patterns within the graph while preserving equivariance.

Table 2. Performance comparison on LastFM, Douban-Book and Tmall under the 32-dimensional embedding setting. Results are presented in terms of Recall@20 and AUC (mean and standard deviation). Bold values indicate the best performance.

Method	LastFM		Douban-Book		Tmall	
	Recall@20	**AUC**	**Recall@20**	**AUC**	**Recall@20**	**AUC**
LightGCN	0.2451 ± 0.0018	0.6408 ± 0.0004	0.1277 ± 0.0016	0.5652 ± 0.0003	0.0921 ± 0.0005	0.5478 ± 0.0004
EGNN	0.1301 ± 0.0027	0.5817 ± 0.0010	0.0731 ± 0.0024	0.5415 ± 0.0014	0.0411 ± 0.0016	0.5223 ± 0.0008
PEG	0.1439 ± 0.0039	0.5909 ± 0.0007	0.0970 ± 0.0008	0.5516 ± 0.0006	0.0475 ± 0.0019	0.5255 ± 0.0007
EPCF	**0.2652** ± 0.0015	**0.6527** ± 0.0006	**0.1387** ± 0.0011	**0.5743** ± 0.0008	**0.0960** ± 0.0012	**0.5498** ± 0.0005

5.3. Vertical Enhancement: Applying EPCF to Other GNN Backbones

To further explore the generalization ability of EPCF in recommendation, we integrated EPCF into several representative GNN backbones, including GCN, GAT, FAGCN, XSimGCL, and Differmer, and we investigated how positional encoding and equivariant propagation influenced recommendation performance across different architectural paradigms.

As shown in Table 3, EPCF improved Recall@20 by an average of 23.13% and AUC by 2.14% over the original backbones across all the models and datasets. Overall, EPCF improved the performance of most of the backbone models. GCN, FAGCN, XSimGCL and Differmer achieved performance improvements after integrating EPCF. The improvements were most evident on GCN and FAGCN, which achieved gains across all the datasets after integrating equivariant positional propagation. These models mainly rely on graph convolution and neighborhood aggregation, therefore the additional positional information provided by EPCF can complement their original propagation process. Differmer + EPCF and XSimGCL + EPCF achieved performance improvements over the original backbones on four out of the five datasets, demonstrating that EPCF provides benefits across diverse propagation mechanisms in GNNs. For GAT, the situation is different. GAT already incorporates adaptive weighting mechanisms when aggregating information from neighbors. It explicitly computes structure-aware neighbor weights through the attention mechanism, and the additional positional information introduced by EPCF may therefore introduce interference. As a result, the positional signals propagated by EPCF may overlap with the information captured by the original propagation mechanism, leading to performance degradation on some datasets.

Table 3. Performance comparison on five backbones, report in Recall@20 and AUC (mean and standard deviation). **Bold** values indicate better performance between each backbone and its EPCF version. The last two columns report the average relative improvement of EPCF over the corresponding backbone across the five datasets.

Method	Twitch-RU		Twitch-ES		Douban-Book		ProgrammableWeb		Yelp		Avg. Imp.	
	Recall@20	AUC	Recall@20	AUC	Recall@20	AUC	Recall@20	AUC	Recall@20	AUC	Recall@20	AUC
GCN	0.1154 ± 0.0003	0.5626 ± 0.0004	0.1387 ± 0.0018	0.5897 ± 0.0009	0.0556 ± 0.0013	0.5274 ± 0.0002	0.2684 ± 0.0025	0.6393 ± 0.0010	0.0603 ± 0.0008	0.5699 ± 0.0004		
GCN + EPCF	**0.1506** ± 0.0028	**0.5806** ± 0.0011	**0.1541** ± 0.0014	**0.5974** ± 0.0002	**0.0713** ± 0.0003	**0.5352** ± 0.0017	**0.4573** ± 0.0022	**0.7380** ± 0.0009	**0.0623** ± 0.0004	**0.5709** ± 0.0002	**+28.71%**	**+4.32%**
GAT	0.0903 ± 0.0015	0.5500 ± 0.0002	0.1044 ± 0.0026	0.5725 ± 0.0007	0.0433 ± 0.0005	0.5212 ± 0.0003	0.2272 ± 0.0011	0.6175 ± 0.0006	0.0430 ± 0.0031	0.5613 ± 0.0014		
GAT + EPCF	0.0714 ± 0.0009	0.5404 ± 0.0005	0.0978 ± 0.0008	0.5654 ± 0.0004	**0.0456** ± 0.0004	**0.5224** ± 0.0002	**0.2474** ± 0.0009	**0.6294** ± 0.0005	0.0307 ± 0.0006	0.5551 ± 0.0004	−8.33%	−0.39%
FAGCN	0.0460 ± 0.0017	0.5276 ± 0.0003	0.0740 ± 0.0029	0.5535 ± 0.0012	0.0555 ± 0.0036	0.5273 ± 0.0020	0.1365 ± 0.0010	0.5725 ± 0.0014	0.0546 ± 0.0005	0.5671 ± 0.0009		
FAGCN + EPCF	**0.1034** ± 0.0022	**0.5566** ± 0.0004	**0.1315** ± 0.0006	**0.5861** ± 0.0013	**0.0843** ± 0.0034	**0.5417** ± 0.0006	**0.3723** ± 0.0021	**0.6939** ± 0.0011	**0.0798** ± 0.0013	**0.5797** ± 0.0005	**+94.65%**	**+7.51%**
XSimGCL	0.1273 ± 0.0016	0.5686 ± 0.0008	0.1379 ± 0.0014	0.5892 ± 0.0007	0.0669 ± 0.0010	0.5330 ± 0.0008	0.2474 ± 0.0027	0.6280 ± 0.0014	0.0425 ± 0.0009	0.5610 ± 0.0008		
XSimGCL + EPCF	**0.1298** ± 0.0013	**0.5700** ± 0.0007	**0.1387** ± 0.0016	**0.5897** ± 0.0006	0.0601 ± 0.0005	0.5296 ± 0.0007	**0.2566** ± 0.0022	**0.6379** ± 0.0010	**0.0454** ± 0.0008	**0.5625** ± 0.0006	**+0.58%**	**+0.31%**
Differmer	0.0902 ± 0.0015	0.5500 ± 0.0006	0.1214 ± 0.0017	0.5810 ± 0.0009	0.0544 ± 0.0009	0.5267 ± 0.0003	0.3364 ± 0.0038	0.6735 ± 0.0016	0.0585 ± 0.0010	0.5893 ± 0.0008		
Differmer + EPCF	**0.0982** ± 0.0013	**0.5538** ± 0.0007	**0.1279** ± 0.0014	**0.5842** ± 0.0002	**0.0595** ± 0.0004	**0.5293** ± 0.0006	0.2384 ± 0.0042	0.6232 ± 0.0023	**0.0619** ± 0.0009	**0.5907** ± 0.0007	**+0.06%**	−1.10%

Although EPCF did not improve every backbone on every dataset, it achieved better performance in most settings, indicating that the proposed equivariant positional propagation mechanism is compatible with different GNN architectures.

5.4. Confidence Interval Analysis

To further examine the stability of the results in Sections 5.2 and 5.3, we computed the 95% confidence intervals based on the five independent runs. The confidence interval was calculated as

$$CI = \bar{x} \pm t_{0.975,4} \frac{s}{\sqrt{5}},$$

where $\bar{x}$ denotes the mean performance, s is the standard deviation, and $t_{0.975,4} = 2.776$ is the critical value of the distribution with four degrees of freedom.

For our horizontal comparison results shown in Table 4, the confidence intervals of EPCF and the strongest baseline LightGCN did not overlap on any dataset for either Recall@20 or AUC. This indicates that the improvements were larger than the variation caused by different random initializations and suggests that the performance gains were statistically stable in our horizontal comparison.

Table 4. 95% confidence intervals of EPCF and the strongest baseline for the horizontal comparison.

Dataset	Metric	LightGCN 95% CI	EPCF 95% CI	Overlap
LastFM	Recall@20	[0.2429, 0.2473]	[0.2633, 0.2671]	No
	AUC	[0.6403, 0.6413]	[0.6520, 0.6534]	No
Douban-Book	Recall@20	[0.1257, 0.1297]	[0.1373, 0.1401]	No
	AUC	[0.5648, 0.5656]	[0.5733, 0.5753]	No
Tmall	Recall@20	[0.0915, 0.0927]	[0.0945, 0.0975]	No
	AUC	[0.5473, 0.5483]	[0.5492, 0.5504]	No

For the vertical comparison results shown in Table 5, most comparisons between the original backbone and the version with EPCF did not overlap 95% confidence intervals. Specifically, 24 out of 25 Recall@20 comparisons and 21 out of 25 AUC comparisons had non-overlapping confidence intervals. AUC considered all the items rather than only the top-ranked recommendations. Thus the differences between the methods were smaller and most overlaps occurred in AUC. Only a few results involving XSimGCL and Difformer had overlapping intervals. The average improvements of these backbones were smaller than those of GCN and FAGCN.

We then examined these cases using 90% confidence intervals. For XSimGCL on Twitch-RU AUC, the confidence intervals became [0.5678, 0.5694] and [0.5693, 0.5707], which no longer overlapped. For XSimGCL on Yelp AUC, the confidence intervals became [0.5602, 0.5618] and [0.5619, 0.5631], which also did not overlap. For XSimGCL on Twitch-RU Recall@20, the overlap became very small.

The confidence interval analysis suggests that the improvements brought by EPCF were stable across most of the datasets and backbones and that the amount of improvement varied across different backbone architectures.

Table 5. 95% confidence intervals for the vertical enhancement experiments.

Backbone	Dataset	Recall@20			AUC		
		Base CI	EPCF CI	Overlap	Base CI	EPCF CI	Overlap
GCN	Twitch-RU	[0.1150, 0.1158]	[0.1471, 0.1541]	No	[0.5621, 0.5631]	[0.5792, 0.5820]	No
	Twitch-ES	[0.1365, 0.1409]	[0.1524, 0.1558]	No	[0.5886, 0.5908]	[0.5972, 0.5976]	No
	Douban-Book	[0.0540, 0.0572]	[0.0709, 0.0717]	No	[0.5272, 0.5276]	[0.5331, 0.5373]	No
	ProgrammableWeb	[0.2653, 0.2715]	[0.4546, 0.4600]	No	[0.6381, 0.6405]	[0.7369, 0.7391]	No
	Yelp	[0.0593, 0.0613]	[0.0618, 0.0628]	No	[0.5694, 0.5704]	[0.5707, 0.5711]	No
GAT	Twitch-RU	[0.0884, 0.0922]	[0.0703, 0.0725]	No	[0.5498, 0.5502]	[0.5398, 0.5410]	No
	Twitch-ES	[0.1012, 0.1076]	[0.0968, 0.0988]	No	[0.5716, 0.5734]	[0.5649, 0.5659]	No
	Douban-Book	[0.0427, 0.0439]	[0.0451, 0.0461]	No	[0.5208, 0.5216]	[0.5222, 0.5226]	No
	ProgrammableWeb	[0.2258, 0.2286]	[0.2463, 0.2485]	No	[0.6168, 0.6182]	[0.6288, 0.6300]	No
	Yelp	[0.0392, 0.0468]	[0.0300, 0.0314]	No	[0.5596, 0.5630]	[0.5546, 0.5556]	No
FAGCN	Twitch-RU	[0.0439, 0.0481]	[0.1007, 0.1061]	No	[0.5272, 0.5280]	[0.5561, 0.5571]	No
	Twitch-ES	[0.0704, 0.0776]	[0.1308, 0.1322]	No	[0.5520, 0.5550]	[0.5845, 0.5877]	No
	Douban-Book	[0.0510, 0.0600]	[0.0801, 0.0885]	No	[0.5248, 0.5298]	[0.5410, 0.5424]	No
	ProgrammableWeb	[0.1353, 0.1377]	[0.3697, 0.3749]	No	[0.5708, 0.5742]	[0.6925, 0.6953]	No
	Yelp	[0.0540, 0.0552]	[0.0782, 0.0814]	No	[0.5660, 0.5682]	[0.5791, 0.5803]	No
XSimGCL	Twitch-RU	[0.1253, 0.1291]	[0.1293, 0.1314]	No	[0.5676, 0.5696]	[0.5691, 0.5709]	Yes
	Twitch-ES	[0.1362, 0.1396]	[0.1367, 0.1407]	Yes	[0.5883, 0.5901]	[0.5890, 0.5904]	Yes
	Douban-Book	[0.0657, 0.0681]	[0.0595, 0.0607]	No	[0.5320, 0.5340]	[0.5287, 0.5305]	No
	ProgrammableWeb	[0.2440, 0.2508]	[0.2539, 0.2593]	No	[0.6263, 0.6297]	[0.6367, 0.6391]	No
	Yelp	[0.0414, 0.0436]	[0.0444, 0.0464]	No	[0.5600, 0.5620]	[0.5618, 0.5632]	Yes
Differmer	Twitch-RU	[0.0883, 0.0921]	[0.0966, 0.0998]	No	[0.5493, 0.5507]	[0.5529, 0.5547]	No
	Twitch-ES	[0.1193, 0.1235]	[0.1262, 0.1296]	No	[0.5799, 0.5821]	[0.5840, 0.5844]	No
	Douban-Book	[0.0533, 0.0555]	[0.0590, 0.0600]	No	[0.5263, 0.5271]	[0.5286, 0.5300]	No
	ProgrammableWeb	[0.3317, 0.3411]	[0.2332, 0.2436]	No	[0.6715, 0.6755]	[0.6203, 0.6261]	No
	Yelp	[0.0573, 0.0597]	[0.0608, 0.0630]	No	[0.5883, 0.5903]	[0.5898, 0.5916]	Yes

5.5. Ablation Studies

We conducted an ablation study to investigate the impact of different components in our architecture, in order to gauge their contribution to recommendation performance:

- **w/o Edge Feature**. To verify the effectiveness of the edge feature in the node embedding propagation module, we conducted experiments where we removed this edge feature;
- **w/o Equivariant Position Propagation**. To investigate the effect of the equivariant position propagation module, we conducted an experiment in which this module was removed;
- **w/o Position Concatenation**. To evaluate the pivotal role of the position concatenation module, we conducted an ablation study by excluding this module;
- **w/o Layer-Wise Averaging**. In order to further validate the effectiveness of the layer-wise embedding averaging, we conduct an ablation study by removing this module.

Table 6 summarizes the AUC performance of our model across ablation variants on the LastFM and Douban-Book datasets. The full model achieved the best AUC performance (0.6527 on LastFM and 0.5743 on Douban-Book). Removing edge features reduced AUC by 2.33% on LastFM and by 1.08% on Douban-Book. Without equivariant position propagation, AUC dropped by 0.52% on LastFM and by 0.77% on Douban-Book. Removing position concatenation reduced AUC by 1.55% on LastFM and by 2.18% on Douban-Book. Excluding layer-wise embedding averaging led to a decrease of 1.7% on LastFM and 3.03% on Douban-Book. These results indicate that all the components contributed to the overall performance. In particular, edge features, position concatenation and the layer-wise embedding averaging had an impact on AUC across both datasets. The impact of equivariant position propagation

was comparatively smaller. This is consistent with the fact that the positional signals already encode rich structural information, while equivariant position propagation mainly acts as a complementary component that further enhances the model's generalization ability.

Table 6. Ablation study of EPCF evaluated by AUC (mean and standard deviation). Relative performance drop w.r.t. the full model is reported in parentheses. ↓ denotes the percentage decrease compared with the full model.

Method Variant	LastFM	Douban-Book
Full Model	0.6527 ± 0.0006	0.5743 ± 0.0008
w/o Edge Feature	0.6375 ± 0.0010 (↓2.33%)	0.5681 ± 0.0006 (↓1.08%)
w/o Equivariant Position Propagation	0.6493 ± 0.0009 (↓0.52%)	0.5699 ± 0.0007 (↓0.77%)
w/o Position Concatenation	0.6426 ± 0.0008 (↓1.55%)	0.5618 ± 0.0012 (↓2.18%)
w/o Layer-Wise Averaging	0.6416 ± 0.0005 (↓1.70%)	0.5569 ± 0.0007 (↓3.03%)

5.6. Hyperparameter Analysis

We conducted a sensitivity analysis on three key hyperparameters: embedding dimension, positional feature dimension and the number of layers. We report Recall@20 and AUC on the LastFM, Tmall and Douban-Book datasets.

5.6.1. Embedding Dimension

We evaluated the impact of the embedding dimension by testing values in the set {8, 16, 32, 64}. As shown in Figure 4, increasing the embedding dimension generally improved both Recall@20 and AUC across all three datasets. The largest gains were observed when increasing the dimension from 8 to 32. Further increasing the dimension from 32 to 64 yielded only marginal improvements. Therefore, an embedding dimension of 32 provides a good balance between model performance and computational efficiency.

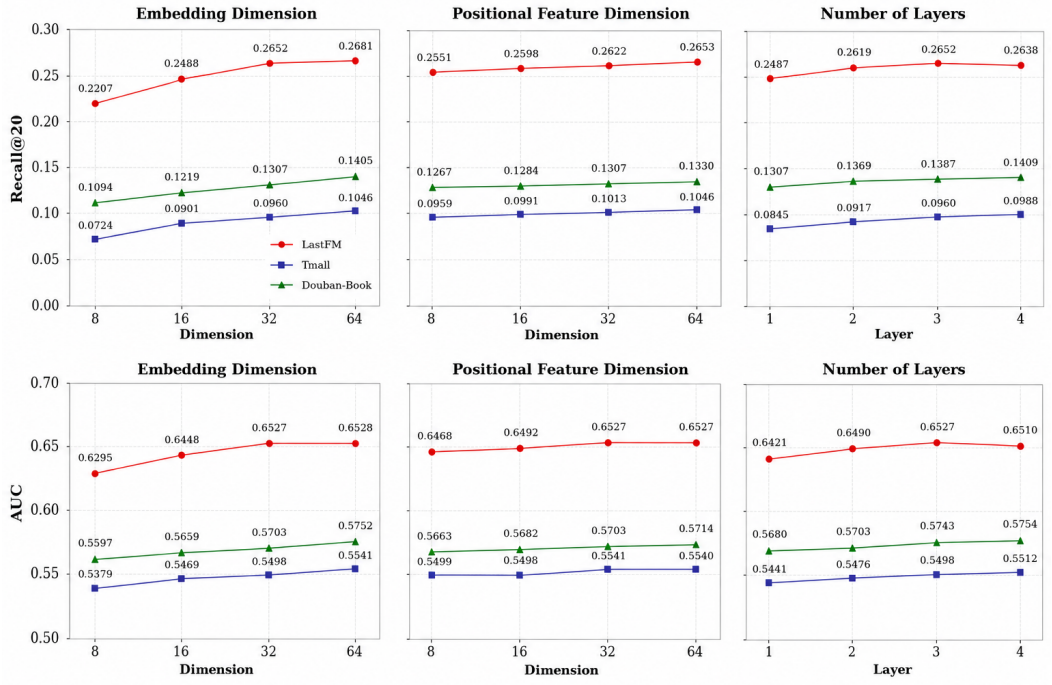

Figure 4. Parameter sensitivity analysis of EPCF under different embedding dimensions, positional feature dimensions and numbers of layers. Results are reported in terms of Recall@20 and AUC on the LastFM, Tmall and Douban-Book datasets.

5.6.2. Positional Feature Dimension

The dimensionality of positional features generated by Laplacian eigenvectors k determines the geometric information of nodes. We evaluated k from the set {8, 16, 32, 64} when the embedding dimension was 32. As shown in Figure 4, the results indicate that a higher dimension generally leads to better performance. Although the best performance was generally achieved at 64 dimensions, the improvements over 32 dimensions were relatively small. Considering the additional computational cost of eigendecomposition and memory consumption, we used 32-dimensional positional features in the main experiments.

5.6.3. Number of Layers

We investigated the effect of model depth by varying the number of layers from one to four. As shown in Figure 4, on the LastFM dataset Recall@20 reached its highest value at three layers and slightly decreased at four layers. On the Tmall and Douban-Book datasets, performance generally improved as the number of layers increased. Similar trends can also be observed for AUC. Overall, three layers provided a good balance between recommendation performance and model complexity.

6. Discussion

Our experimental results demonstrate that EPCF can improve recommendation performance across different datasets and GNN backbones. In this section, we further provide a geometric interpretation to explain why equivariant positional propagation effectively enhances collaborative filtering performance, and we provide the complexity analysis of Laplacian eigenvector computation.

6.1. Effect of Equivariant Positional Propagation

From the perspective of graph representation learning, traditional message-passing GNNs mainly aggregate local neighborhood information and may fail to distinguish nodes with similar local structures. In contrast, the proposed equivariant positional propagation introduces additional relative positional information during aggregation while preserving the symmetry properties of graph representations. This may help the model capture additional structural information and improve node discrimination. The discussion below is intended as an analogy rather than a geometric assumption on recommender graphs. It is only used to illustrate that the proposed positional propagation aggregates weighted relative offsets from neighboring nodes, similar to differential coordinates.

We begin by revisiting the symmetric Laplacian matrix, which is defined as $L_s = D - A$, where D is the degree matrix and A is the adjacency matrix. For a vector $\mathbf{x}$ containing the x-coordinates of all vertices (the same analysis applies to other basis components such as y and z) the Laplacian operation can be written as $L_s\mathbf{x} = D\delta(\mathbf{x})$, where $\delta(\mathbf{x})$ represents the differential coordinates.

From the perspective of differential geometry, the differential coordinates δ can be viewed as a discrete approximation of the continuous Laplace–Beltrami operator, assuming that the mesh $\mathcal{M}$ is a piecewise linear approximation of a smooth surface. The differential coordinate vector at vertex v_i is defined as

$$\delta_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (\mathbf{v}_i - \mathbf{v}_j), \tag{11}$$

where $\mathcal{N}_i$ denotes the set of neighbors of vertex v_i. We illustrate this discrete differential coordinate in Figure 5a. As shown in Figure 5b, the above summation is a discrete analog of the following curvilinear integral:

$$\frac{1}{|\gamma|}\int_{\mathbf{v}\in\gamma}(\mathbf{v}_i-\mathbf{v})\,dl, \tag{12}$$

where γ is a closed curve around $\mathbf{v}_i$, and $|\gamma|$ is its length.

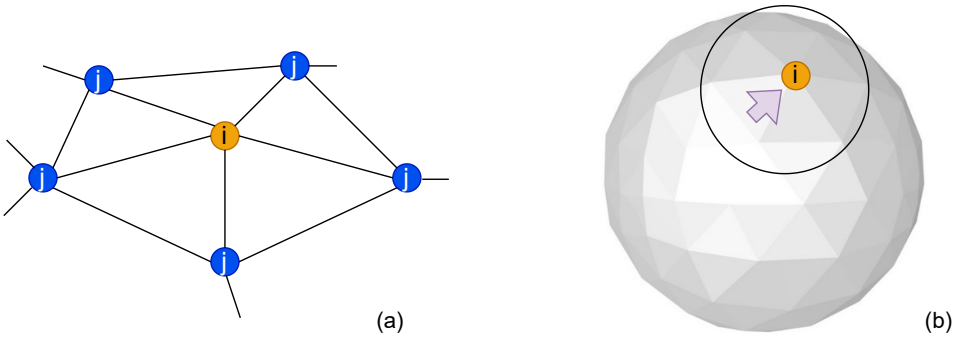

Figure 5. (**a**) The differential coordinate δ_i captures the aggregate of relative offsets to neighboring vertices; (**b**) The vector of the differential coordinates at a vertex approximates the local shape characteristics of the surface.

In our model, the position propagation in Equations (3) and (4) adopts a similar form to the differential coordinate vector. Specifically, we perform a weighted sum of position differences $(\mathbf{z}_u - \mathbf{z}_i)$ for position propagation. This operation aggregates the relative positional information of neighboring nodes into the current node, enriching the information captured during aggregation. By incorporating this additional positional information, our model improves recommendation performance.

6.2. Complexity Analysis

The main additional cost of EPCF comes from computing Laplacian positional encoding. Let N and E denote the numbers of nodes and edges. The first k non-trivial eigenvectors are computed once before training using a sparse eigensolver, with complexity approximately $O(k|E|)$ and memory cost $O(|E| + Nk)$.

The positional encoding remain fixed throughout training and is not recomputed. Therefore, the additional training overhead is limited to positional propagation and edge feature computation.

7. Conclusions

In this paper, we propose EPCF, an equivariant graph neural network framework for collaborative filtering that incorporates Laplacian positional encoding through an equivariant positional propagation mechanism. The framework jointly propagates positional features and node embeddings, aiming to introduce additional structural information into recommendation while preserving permutation and orthogonal equivariance.

From a theoretical perspective, we analyzed the equivariance properties of the proposed propagation mechanism and proved that the positional updates remained equivariant under both permutation and orthogonal transformations. From a practical perspective, the proposed EPCF can be incorporated into existing GNN-based recommender systems with some architectural changes. It can serve as an additional component for improving representation learning in recommendation tasks. Our experimental results show that EPCF achieves better performance than the baselines and improves several GNN backbones when used as an additional component. These results suggest that equivariant positional propagation can provide useful structural information beyond local neighborhood aggregation in an equivariant manner. However, EPCF relies on Laplacian eigendecomposition to

obtain positional features and is evaluated in a transductive setting. In addition, the performance gains vary across different backbone architectures and the interaction between equivariant positional propagation and different recommendation models remains to be further investigated.

In future work, we plan to explore dynamic positional encoding for temporal recommender systems, investigate more scalable positional representations and extend the proposed framework to other graph learning tasks.

Author Contributions: Conceptualization, X.S. and J.Y.; methodology, X.S. and J.Y.; software, X.S.; validation, X.S.; formal analysis, X.S.; investigation, X.S.; resources, J.Y. and J.S.; data curation, X.S.; writing—original draft preparation, X.S.; writing—review and editing, X.S., J.S., J.Y., L.P., G.W., Z.L. and X.L.; visualization, X.S.; supervision, J.Y.; project administration, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All datasets used in this study are publicly available benchmark datasets. The source code and implementation of EPCF are publicly available at: https://github.com/xinxin-mia/EPCF, accessed on 28 June 2026.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Proof of Equivariance of EPCF

In this section, we provide the detailed proof of the equivariance property of our proposed model EPCF.

Appendix A.1. Equivariance Under Orthogonal Transformations

We first consider orthogonal transformations acting on the positional feature space. Such transformations include both coordinate orthogonal transformations and the basis ambiguities arising from Laplacian eigendecomposition. Let $Q \in \mathbb{R}^{k\times k}$ be an orthogonal matrix satisfying $Q^\top Q = QQ^\top = I$.

We aim to prove that the EPCF positional propagation satisfies

$$\left(X^{(l+1)}, Z^{(l+1)}Q\right) = EPCF(A, X^{(l)}, Z^{(l)}Q). \tag{A1}$$

where $EPCF(\cdot)$ denotes one propagation layer of our proposed model.

According to the positional update method in Section 4, Equations (3) and (4) (we demonstrate the proof using Equation (3) as an example), we now prove

$$\mathbf{z}_u^{(l+1)}Q = \mathbf{z}_u^{(l)}Q + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}\left(\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\right)h_{ui}^{(l)}. \tag{A2}$$

Starting from the right-hand side, we have

$$\begin{aligned}&\mathbf{z}_u^{(l)}Q + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}\left(\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\right)h_{ui}^{(l)}\\ &\quad= \mathbf{z}_u^{(l)}Q + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}\left(\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\right)\phi\Big(\Big[\|\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\|^2 \quad \tilde{a}_{ui}\Big]\Big).\end{aligned} \tag{A3}$$

Since orthogonal transformations are linear transformations, the difference of transformed vectors equals the transformation of the difference

$$\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q = \left(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)}\right)Q. \tag{A4}$$

Moreover, the Euclidean distance is invariant under orthogonal transformations:

$$\|\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\|^2 = \|(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)})Q\|^2 = \|\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)}\|^2. \tag{A5}$$

We get

$$\begin{aligned}
&\mathbf{z}_u^{(l)}Q + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}(\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q)\,\phi\Big(\Big[\|\mathbf{z}_u^{(l)}Q - \mathbf{z}_i^{(l)}Q\|^2 \quad \tilde{a}_{ui}\Big]\Big) \\
&= \mathbf{z}_u^{(l)}Q + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)})\,h_{ui}^{(l)}Q \\
&= \mathbf{z}_u^{(l)}Q + \left(\frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)})\,h_{ui}^{(l)}\right)Q \\
&= \left(\mathbf{z}_u^{(l)} + \frac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u}(\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)})\,h_{ui}^{(l)}\right)Q \\
&= \mathbf{z}_u^{(l+1)}Q.
\end{aligned} \tag{A6}$$

Therefore, we have shown that applying an orthogonal transformation to the positional representation at layer l results in the same transformation applied to the updated representation at layer $l+1$. This includes both coordinate orthogonal transformations and orthogonal basis changes within degenerate eigenspaces. Hence,

$$\left(X^{(l+1)}, Z^{(l+1)}Q\right) = EPCF(A, X^{(l)}, Z^{(l)}Q). \tag{A7}$$

Thus, EPCF is equivariant under orthogonal transformations.

Appendix A.2. Equivariance Under Permutations

We now prove that the positional propagation in Equation (3) is equivariant to node permutations. Let $P \in \mathbb{R}^{N\times N}$ be a permutation matrix. Under the permutation, the adjacency matrix and positional feature matrix become

$$A' = PAP^\top, \qquad Z'^{(l)} = PZ^{(l)}. \tag{A8}$$

We aim to prove that EPCF satisfies

$$\left(PX^{(l+1)}, PZ^{(l+1)}\right) = EPCF(PAP^\top, PX^{(l)}, PZ^{(l)}). \tag{A9}$$

where $EPCF(\cdot)$ denotes one propagation layer of the proposed model.

For a node v, let $p(v)$ denote its index after permutation. Since $A' = PAP^\top$, the neighborhood and degree are re-indexed consistently:

$$\mathcal{N}'_{p(v)} = \{\, p(w) \mid w \in \mathcal{N}_v \,\}, \qquad |\mathcal{N}'_{p(v)}| = |\mathcal{N}_v|. \tag{A10}$$

Since $Z'^{(l)} = PZ^{(l)}$, the positional vectors are only reordered:

$$\mathbf{z}'^{(l)}_{p(u)} = \mathbf{z}_u^{(l)}, \qquad \mathbf{z}'^{(l)}_{p(i)} = \mathbf{z}_i^{(l)}. \tag{A11}$$

Therefore, the radial distance is preserved under the permutation:

$$r'_{p(u)p(i)} = \left\|\mathbf{z}'^{(l)}_{p(u)} - \mathbf{z}'^{(l)}_{p(i)}\right\|^2 = \left\|\mathbf{z}_u^{(l)} - \mathbf{z}_i^{(l)}\right\|^2 = r_{ui}. \tag{A12}$$

https://doi.org/10.3390/info17070644

Similarly, the attention coefficient is re-indexed consistently:

$$\tilde{a}'_{p(u)p(i)} = \tilde{a}_{ui}. \tag{A13}$$

Thus,

$$h'_{p(u)p(i)} = \phi\Big(\begin{bmatrix} r'_{p(u)p(i)} & \tilde{a}'_{p(u)p(i)} \end{bmatrix}\Big) = \phi\Big(\begin{bmatrix} r_{ui} & \tilde{a}_{ui} \end{bmatrix}\Big) = h_{ui}. \tag{A14}$$

Starting from the right-hand side, the positional update for node $p(u)$ is

$$\begin{aligned} \mathbf{z}'^{(l+1)}_{p(u)} &= \mathbf{z}'^{(l)}_{p(u)} + \frac{1}{|\mathcal{N}'_{p(u)}|} \sum_{p(i)\in\mathcal{N}'_{p(u)}} \Big(\mathbf{z}'^{(l)}_{p(u)} - \mathbf{z}'^{(l)}_{p(i)}\Big) h'^{(l)}_{p(u)p(i)} \\ &= \mathbf{z}^{(l)}_{u} + \frac{1}{|\mathcal{N}_u|} \sum_{i\in\mathcal{N}_u} \Big(\mathbf{z}^{(l)}_{u} - \mathbf{z}^{(l)}_{i}\Big) h^{(l)}_{ui} \\ &= \mathbf{z}^{(l+1)}_{u}. \end{aligned} \tag{A15}$$

Thus, the updated positional vectors are re-indexed in the same way as the input:

$$Z'^{(l+1)} = PZ^{(l+1)}. \tag{A16}$$

The positional update therefore remains equivariant under node permutations.

For the node embedding update, a permutation does not change the Euclidean distance $r_{ui} = \|\mathbf{z}_u - \mathbf{z}_i\|^2$, and thus the edge feature h_{ui} is re-indexed consistently. Since the neighborhoods and node embeddings are also re-indexed by the same permutation, the embedding output is transformed as $PX^{(l+1)}$.

Combining the positional and embedding updates yields

$$\Big(PX^{(l+1)}, PZ^{(l+1)}\Big) = EPCF(PAP^\top, PX^{(l)}, PZ^{(l)}). \tag{A17}$$

Thus, EPCF satisfies permutation equivariance. Let $\hat{Y}$ denote the final user–item score matrix. Since the propagated embeddings and positional features are re-indexed consistently under permutation, the predicted scores are re-indexed accordingly. For a bipartite graph with $P = P_U \oplus P_I$,

$$\hat{Y}' = P_U \hat{Y} P_I^\top. \tag{A18}$$

Appendix B. Implementation Details of Backbone Models with EPCF

In this section, we provide the layer-wise embedding propagation formulas for all the backbone models integrated with EPCF used in our experiments in Section 5.3. Our method updates node embeddings and positional encoding separately. Thus, the equivariant positional propagation and the final position embedding concatenation remained identical across all the backbones. Layer-wise embedding averaging was preserved only for the models that originally adopted it, ensuring a fair comparison. For each backbone, the same hyperparameter settings were used before and after integrating EPCF, and no additional tuning was performed. For all the backbone experiments, the final prediction score was computed using the same scoring function as EPCF.

Here, we only present the node embedding propagation formulas.

Recall from Section 4.1 that our enhanced embedding update depends on the positional affinity h_{ui} between user u and item i, defined as

$$h_{ui} = \phi([\, r_{ui}, \tilde{a}_{ui} \,]), \tag{A19}$$

where r_{ui} denotes the relative squared distance term and $\tilde{a}_{ui}$ denotes the attention coefficient derived from our equivariant graph module.

Since all experiments were conducted on user–item bipartite graphs, each backbone with EPCF instantiation specified how the user embeddings $\mathbf{x}_u$ and item embeddings $\mathbf{x}_i$ were propagated at each layer.

Appendix B.1. EPCF with GCN

The original GCN propagates information from node neighbors [13]

$$X^{(l+1)} = \sigma\Big(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(l)}W^{(l)}\Big), \tag{A20}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the undirected graph G with added self-connections. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $X^{(l)}$ is the node embedding matrix at layer l, $W^{(l)}$ is the trainable transformation matrix, and $\sigma(\cdot)$ denotes the activation function.

To integrate EPCF, we modify the aggregation term by injecting the positional affinity h_{ui} into user–item interactions while preserving GCN's normalized propagation structure. The user embeddings $\mathbf{x}_u$ and item embeddings $\mathbf{x}_i$ are propagated as follows:

$$\mathbf{x}_u^{(l+1)} = \sigma\left(\sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\mathbf{x}_i^{(l)}W^{(l)}\right), \tag{A21}$$

$$\mathbf{x}_i^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\Big(\sum_{t\in\mathcal{N}_u}\frac{h_{ut}}{|\mathcal{N}_t|}\mathbf{x}_t^{(l)}\Big)W^{(l)}\right). \tag{A22}$$

Appendix B.2. EPCF with GAT

GAT [43] learns attention weights to aggregate neighbor information:

$$\mathbf{x}_u^{(l+1)} = \sigma\left(\sum_{i\in\mathcal{N}_u}\alpha_{ui}^{(l)}\mathbf{x}_i^{(l)}W\right), \tag{A23}$$

with coefficients α_{ui} computed by

$$\alpha_{ui}^{(l)} = \frac{\exp\left(\text{LeakyReLU}([\mathbf{x}_u^{(l)}W\|\mathbf{x}_i^{(l)}W]\mathbf{a})\right)}{\sum_{t\in\mathcal{N}_u}\exp\left(\text{LeakyReLU}([\mathbf{x}_u^{(l)}W\|\mathbf{x}_t^{(l)}W]\mathbf{a})\right)}, \tag{A24}$$

where W and $\mathbf{a}$ are learnable parameters, $(\cdot)^\top$ denotes matrix transposition and $\|$ represents the concatenation operator.

With EPCF, we preserve the attention mechanism while injecting aggregation guided by positional information into the item representations. The user embeddings $\mathbf{x}_u$ and item embeddings $\mathbf{x}_i$ are propagated as

$$\mathbf{x}_u^{(l+1)} = \sigma\left(\sum_{i\in\mathcal{N}_u}\alpha_{ui}^{(l)}\mathbf{x}_i^{(l)}W\right), \tag{A25}$$

$$\mathbf{x}_i^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}_i}\alpha_{iu}^{(l)}\Big[\sum_{t\in\mathcal{N}_u}\frac{h_{ut}}{|\mathcal{N}_t|}\mathbf{x}_t^{(l)}\Big]W\right). \tag{A26}$$

Appendix B.3. EPCF with FAGCN

FAGCN models the trade-off between low-frequency and high-frequency signals [44]. Its original propagation is

$$\mathbf{x}_u^{(l+1)} = \epsilon \mathbf{x}_u^{(l)} + \sum_{i \in \mathcal{N}_u} \left(\frac{\alpha_{ui}^L - \alpha_{ui}^H}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \right) \mathbf{x}_i^{(l)}, \tag{A27}$$

where ϵ is a scaling hyperparameter limited in $[0, 1]$, α_{ui}^L, and where α_{ui}^H represent the proportions of node i's low-frequency and high-frequency signals to node u and $\alpha_{ui}^L + \alpha_{ui}^H = 1$.

To incorporate EPCF, we apply the positional affinity h_{ui} on the neighbor messages while keeping the original frequency weighting scheme. The FAGCN+EPCF propagation is given by

$$\mathbf{x}_u^{(l+1)} = \epsilon \mathbf{x}_u^{(l)} + \sum_{i \in \mathcal{N}_u} \left(\frac{\alpha_{ui}^L - \alpha_{ui}^H}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \right) h_{ui} \mathbf{x}_i^{(l)}, \tag{A28}$$

$$\mathbf{x}_i^{(l+1)} = \epsilon \mathbf{x}_i^{(l)} + \sum_{u \in \mathcal{N}_i} \left(\frac{\alpha_{iu}^L - \alpha_{iu}^H}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \right) h_{iu} \mathbf{x}_u^{(l)}. \tag{A29}$$

Appendix B.4. EPCF with XSimGCL

XSimGCL injects uniform noise into embeddings to improve contrastive uniformity [45]. Its original propagation rule is

$$X^{(l+1)} = \tilde{A} X^{(l)} + \Delta^{(l+1)}, \tag{A30}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the undirected graph G with added self-connections. $\mathbf{X}$ denotes the node feature matrix; $\mathbf{x}_u$ and $\mathbf{x}_i$ denote the embedding vectors of individual user and item nodes; Δ is the noise augmentation term, consisting of different uniform random noises that are added to the aggregated embeddings at each layer.

$$\Delta = \omega \odot \operatorname{sign}(\mathbf{x}_i), \qquad \omega \in \mathbb{R}^d \sim U(0, 1). \tag{A31}$$

When integrating EPCF, we keep the same noise augmentation term while injecting positional affinity into the deterministic graph propagation term. The XSimGCL+EPCF propagation is

$$\mathbf{x}_u^{(l+1)} = \tilde{A} \left(\frac{1}{|\mathcal{N}_u|} \sum_{i \in \mathcal{N}_u} h_{ui} \mathbf{x}_i^{(l)} \right) + \Delta^{(l+1)}, \tag{A32}$$

$$\mathbf{x}_i^{(l+1)} = \tilde{A} \mathbf{x}_i^{(l)} + \Delta^{(l+1)}. \tag{A33}$$

Appendix B.5. EPCF with Differmer

Differmer performs diffusion based on pairwise embedding distances [46]. Its original propagation is

$$\mathbf{x}_u^{(l+1)} = \left(1 - \tfrac{\tau}{2} \sum_i \left(S_{ui}^{(l)} + \tilde{A}_{ui} \right) \right) \mathbf{x}_u^{(l)} + \tfrac{\tau}{2} \sum_i \left(S_{ui}^{(l)} + \tilde{A}_{ui} \right) \mathbf{x}_i^{(l)}, \tag{A34}$$

where $\tilde{A}$ is the symmetrically normalized adjacency matrix, and where $S_{ui}^{(j)}$ is the diffusivity coefficient of Differmer at layer j. It is computed by mapping the embedding distance between nodes u and i through a function $f(\cdot)$, followed by normalization over all candidate

nodes of u. This coefficient controls the strength of information diffusion from node i to node u.

$$S_{ui}^{(l)} = \frac{f\left(\|\mathbf{x}_u^{(l)} - \mathbf{x}_i^{(l)}\|_2^2\right)}{\sum_{t=1}^{N} f\left(\|\mathbf{x}_u^{(l)} - \mathbf{x}_t^{(l)}\|_2^2\right)}, \qquad 1 \le u, i \le N. \tag{A35}$$

To integrate EPCF, we modulate the diffusion process with positional affinities and apply them consistently to both the self and neighbor aggregation. The Difformer+EPCF propagation is given by

$$\mathbf{x}_u^{(l+1)} = \left(1 - \tfrac{\tau}{2}\sum_i (S_{ui}^{(l)} + \tilde{A}_{ui}h_{ui})\right)\left(\tfrac{1}{|\mathcal{N}_u|}\sum_{i\in\mathcal{N}_u} h_{ui}\mathbf{x}_i^{(l)}\right) + \tfrac{\tau}{2}\sum_i (S_{ui}^{(l)} + \tilde{A}_{ui}h_{ui})\mathbf{x}_i^{(l)}, \tag{A36}$$

$$\mathbf{x}_i^{(l+1)} = \left(1 - \tfrac{\tau}{2}\sum_u (S_{ui}^{(l)} + \tilde{A}_{ui}h_{ui})\right)\mathbf{x}_i^{(l)} + \tfrac{\tau}{2}\sum_u (S_{ui}^{(l)} + \tilde{A}_{ui}h_{ui})\left(\tfrac{1}{|\mathcal{N}_t|}\sum_{t\in\mathcal{N}_u} h_{ut}\mathbf{x}_t^{(l)}\right). \tag{A37}$$

Appendix C. Notation Summary

For clarity, Table A1 summarizes the main symbols used throughout the paper.

Table A1. Summary of main notations.

Symbol	Description
$G = (U, V, E)$	User–item bipartite graph
$U,\ V$	User set and item set
u	User index
i	Item index or positive item in BPR loss
j	Negative item in BPR loss
t	Neighboring item index used in aggregation
N	Total number of nodes
$A \in \mathbb{R}^{N\times N}$	Adjacency matrix
$R \in \mathbb{R}^{\lvert U\rvert\times\lvert V\rvert}$	User–item interaction matrix
$\mathbf{x}_u, \mathbf{x}_i \in \mathbb{R}^{1\times d}$	Row embedding vectors of user and item nodes
$X \in \mathbb{R}^{N\times d}$	Node embedding matrix
$\mathbf{z}_u, \mathbf{z}_i \in \mathbb{R}^{1\times k}$	Row positional vectors of user and item nodes
$Z \in \mathbb{R}^{N\times k}$	Positional feature matrix
d	Embedding dimension
k	Positional feature dimension
$\mathcal{N}_u, \mathcal{N}_i$	Neighborhood sets
h_{ui}	Edge feature/propagation weight
P	Permutation matrix
Q	Orthogonal transformation matrix

References

1. Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; Wang, M. A Neural Influence Diffusion Model for Social Recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Paris, France, 21–25 July 2019; pp. 235–244.
2. Sha, X.; Sun, Z.; Zhang, J. Disentangling Multi-Facet Social Relations for Recommendation. *IEEE Trans. Comput. Soc. Syst.* **2021**, *9*, 867–878. [CrossRef]
3. Chen, Y.; Qian, W.; Liu, D.; Zhou, M.; Su, Y.; Han, J.; Li, R. Your Social Circle Affects Your Interests: Social Influence Enhanced Session-based Recommendation. In Proceedings of the International Conference on Computational Science, London, UK, 21–23 June 2022; pp. 769–781.
4. Wang, H.; Yin, H.; Zhang, M.; Li, P. Equivariant and Stable Positional Encoding for More Powerful Graph Neural Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 25–29 April 2022.
5. You, J.; Ying, R.; Leskovec, J. Position-Aware Graph Neural Networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 7134–7143.

6. Perozzi, B.; Al-Rfou, R.; Skiena, S. DeepWalk: Online Learning of Social Representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), New York, NY, USA, 24–27 August 2014; pp. 701–710.
7. Mialon, G.; Chen, D.; Selosse, M.M.; Mairal, J. GraphiT: Encoding Graph Structure in Transformers. *arXiv* **2021**, arXiv:2106.05667.
8. Bouritsas, G.; Frasca, F.; Zafeiriou, S.; Bronstein, M.M. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 657–668. [CrossRef] [PubMed]
9. Zhao, L.; Jin, W.; Akoglu, L.; Shah, N. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 25–29 April 2022.
10. 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? In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual Event, 6–14 December 2021.
11. Rampásek, 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 Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 14501–14515.
12. Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; Tossou, P. Rethinking Graph Transformers with Spectral Attention. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 21618–21629.
13. Kipf, T.N.; Welling, M. Semi-supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
14. Xiao, Y.; Yao, L.; Pei, Q.; Wang, X.; Yang, J.; Sheng, Q.Z. MGNN: Mutualistic Graph Neural Network for Joint Friend and Item Recommendation. *IEEE Intell. Syst.* **2020**, *35*, 7–17. [CrossRef]
15. Guo, Z.; Wang, H. A Deep Graph Neural Network-Based Mechanism for Social Recommendations. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2776–2783. [CrossRef]
16. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.D.; Wang, M. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Virtual Event, 25–30 July 2020; pp. 639–648.
17. Tao, Y.; Li, Y.; Zhang, S.; Hou, Z.; Wu, Z. Revisiting Graph Based Social Recommendation: A Distillation Enhanced Social Graph Network. In Proceedings of the ACM Web Conference (WWW), Lyon, France, 25–29 April 2022; pp. 2830–2838.
18. Wu, J.; Fan, W.; Chen, J.; Liu, S.; Li, Q.; Tang, K. Disentangled Contrastive Learning for Social Recommendation. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), Atlanta, GA, USA, 17–21 October 2022; pp. 4570–4574.
19. Bi, Z.; Jing, L.; Shan, M.; Dou, S.; Wang, S. Hierarchical Social Recommendation Model Based on a Graph Neural Network. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 9107718. [CrossRef]
20. Zhang, P.; Yan, Y.; Zhang, X.; Li, C.; Wang, S.; Huang, F.; Kim, S. TransGNN: Harnessing the collaborative power of transformers and graph neural networks for recommender systems. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington, DC, USA, 14–18 July 2024; pp. 1285–1295.
21. Wu, Q.; Zhang, H.; Gao, X.; He, P.; Weng, P.; Gao, H.; Chen, G. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In Proceedings of the 2019 World Wide Web Conference (WWW), San Francisco, CA, USA, 13–17 May 2019; pp. 2091–2102.
22. Yang, L.; Liu, Z.; Wang, Y.; Wang, C.; Fan, Z.; Yu, P.S. Large-scale Personalized Video Game Recommendation via Social-aware Contextualized Graph Neural Network. In Proceedings of the ACM Web Conference (WWW), Lyon, France, 25–29 April 2022; pp. 337–347.
23. Fu, B.; Zhang, W.; Hu, G.; Dai, X.; Huang, S.; Chen, J. Dual Side Deep Context-aware Modulation for Social Recommendation. In Proceedings of the ACM Web Conference (WWW), Ljubljana, Slovenia, 19–23 April 2021; pp. 3374–3384.
24. Jin, B.; Cheng, K.; Zhang, L.; Fu, Y.; Yin, M.; Jiang, L. Partial Relationship Aware Influence Diffusion via a Multi-channel Encoding Scheme for Social Recommendation. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), Virtual Event, 19–23 October 2020; pp. 845–854.
25. Lin, J.; Chen, S.; Wang, J. Graph Neural Networks with Dynamic and Static Representations for Social Recommendation. In Proceedings of the International Conference on Database Systems for Advanced Applications, Virtual Event, 11–14 April 2022; pp. 294–308.
26. Liu, C.; Li, Y.; Lin, H.; Zhang, C. GNNRec: Gated Graph Neural Network for Session-Based Social Recommendation Model. *J. Intell. Inf. Syst.* **2023**, *60*, 137–156.
27. Han, J.; Tao, Q.; Tang, Y.; Xia, Y. DH-HGCN: Dual Homogeneity Hypergraph Convolutional Network for Multiple Social Recommendations. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Madrid, Spain, 11–15 July 2022; pp. 2190–2194.

28. 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 ACM Web Conference (WWW), Ljubljana, Slovenia, 19–23 April 2021; pp. 413–424.
29. Sang, L.; Zhang, C.; Huang, M.; Mu, L.; Zhang, Y.; Wu, X. Heterogeneous Neighborhood-Enhanced Graph Contrastive Learning for Recommendation. *IEEE Trans. Comput. Soc. Syst.* **2026**, *13*, 1324–1340. [CrossRef]
30. Xia, L.; Xie, M.; Xu, Y.; Huang, C. MixRec: Heterogeneous graph collaborative filtering. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, Hannover, Germany, 10–14 March 2025; pp. 136–145.
31. Yi, Z.; Ounis, I.; Macdonald, C. Graph contrastive learning with positional representation for recommendation. In *Proceedings of the European Conference on Information Retrieval, Dublin, Ireland, 2–6 April 2023*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 288–303.
32. Li, C.; Xia, L.; Ren, X.; Ye, Y.; Xu, Y.; Huang, C. Graph Transformer for Recommendation. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, 23–27 July 2023; pp. 1680–1689. [CrossRef]
33. Xia, L.; Huang, C.; Zhang, C. Self-Supervised Hypergraph Transformer for Recommender Systems. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2100–2109.
34. Chen, J.; Wu, J.; Chen, J.; Gao, C.; Li, Y.; Wang, X. Position-aware Graph Transformer for Recommendation. *arXiv* **2024**, arXiv:2412.18731. [CrossRef]
35. Srinivasan, B.; Ribeiro, B. On the Equivalence Between Positional Node Embeddings and Structural Graph Representations. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 26 April–1 May 2020.
36. Garcia Satorras, V.; Hoogeboom, E.; Welling, M. E(n) Equivariant Graph Neural Networks. In Proceedings of the International Conference on Machine Learning (ICML), Virtual Event, 18–24 July 2021; pp. 9323–9332.
37. Han, J.; Cen, J.; Wu, L.; Li, Z.; Kong, X.; Jiao, R.; Yu, Z.; Xu, T.; Wu, F.; Wang, Z.; et al. A Survey of Geometric Graph Neural Networks: Data Structures, Models and Applications. *Front. Comput. Sci.* **2025**, *19*, 1911375. [CrossRef]
38. Liu, C.; Jin, B.; Liu, J. EDEN: Learning Deterministic Node Position with Equivariant Distance Encoding. *arXiv* **2023**, arXiv:2306.01677.
39. Bo, D.; Fang, Y.; Liu, Y.; Shi, C. Graph Contrastive Learning with Stable and Scalable Spectral Encoding. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10–16 December 2023.
40. Kanatsoulis, C.I.; Choi, E.; Jegelka, S.; Leskovec, J.; Ribeiro, A. Learning Efficient Positional Encodings with Graph Neural Networks. *arXiv* **2025**, arXiv:2502.01122.
41. Zhou, J.; Zhou, C.; Wang, X.; Li, P.; Zhang, M. Towards Stable, Globally Expressive Graph Representations with Laplacian Eigenvectors. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 10–15 December 2024.
42. Ma, G.; Wang, Y.; Lim, D.; Jegelka, S.; Wang, Y. A Canonicalization Perspective on Invariant and Equivariant Learning. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 10–15 December 2024.
43. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2017**, arXiv:1710.10903.
44. Bo, D.; Wang, X.; Shi, C.; Shen, H. Beyond Low-frequency Information in Graph Convolutional Networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Virtual Event, 2–9 February 2021; Volume 35, pp. 3950–3957.
45. Yu, J.; Xia, X.; Chen, T.; Cui, L.; Hung, N.Q.V.; Yin, H. XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 913–926.
46. Wu, Q.; Yang, C.; Zhao, W.; He, Y.; Wipf, D.; Yan, J. Difformer: Scalable (Graph) Transformers Induced by Energy-constrained Diffusion. *arXiv* **2023**, arXiv:2301.09474.