Do We Really Need Graph Convolution During Training? Light Post-Training Graph-ODE for Efficient Recommendation

Weizhi Zhang
wzhan42@uic.edu
University of Illinois Chicago
Chicago, USA

Liangwei Yang
lyang84@uic.edu
University of Illinois Chicago
Chicago, USA

Zihe Song
zsong29@uic.edu
University of Illinois Chicago
Chicago, USA

Henry Peng Zou
pzou3@uic.edu
University of Illinois Chicago
Chicago, USA

Ke Xu
Liancheng Fang
kxu25@uic.edu
lfang87@uic.edu
University of Illinois Chicago
Chicago, USA

Philip S. Yu
psyu@uic.edu
University of Illinois Chicago
Chicago, USA

Abstract

The efficiency and scalability of graph convolution networks (GCNs) in training recommender systems (RecSys) have been persistent concerns, hindering their deployment in real-world applications. This paper presents a critical examination of the necessity of graph convolutions during the training phase and introduces an innovative alternative: the Light Post-Training Graph Ordinary-Differential-Equation (LightGODE). Our investigation reveals that the benefits of GCNs are more pronounced during testing rather than training. Motivated by this, LightGODE utilizes a novel post-training graph convolution method that bypasses the computation-intensive message passing of GCNs and employs a non-parametric continuous graph ordinary-differential-equation (ODE) to dynamically model node representations. This approach drastically reduces training time while achieving fine-grained post-training graph convolution to avoid the distortion of the original training embedding space, termed the embedding discrepancy issue. We validate our model across several real-world datasets of different scales, demonstrating that LightGODE not only outperforms GCN-based models in terms of efficiency and effectiveness but also significantly mitigates the embedding discrepancy commonly associated with deeper graph convolution layers. Our LightGODE challenges the prevailing paradigms in RecSys training and suggests re-evaluating the role of graph convolutions, potentially guiding future developments of efficient large-scale graph-based RecSys.

CCS Concepts

• **Information systems** → **Recommender systems**; **Collaborative filtering**; • **Mathematics of computing** → *Ordinary differential equations.*

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679773

Keywords

Graph Recommendation, Efficient Recommendation, Graph Convolution Network, Ordinary-Differential-Equation

ACM Reference Format:
Weizhi Zhang, Liangwei Yang, Zihe Song, Henry Peng Zou, Ke Xu, Liancheng Fang, and Philip S. Yu. 2024. Do We Really Need Graph Convolution During Training? Light Post-Training Graph-ODE for Efficient Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627673.3679773

1 Introduction

Recommender systems (RecSys) are significant integral parts of many online platforms and web applications, helping users navigate vast amounts of information by providing personalized item recommendations. These systems are essential across various domains such as digital retailing [14, 34], social networking platforms [6, 15], and video-sharing services [39], where they filter and tailor content to align with individual user preferences. Among the techniques [19, 27, 30, 32] used in RecSys, collaborative filtering (CF) [19] is notably effective, and it predicts user preferences based on historical user-item interactions. Essentially, those historical interactions can be represented as a user-item bipartite graph. Inspired by the superior ability of graph convolution networks (GCNs) [17, 23, 40, 43] in modeling on graph-structured data, a large number of GCN-based recommendation models [9, 37, 38, 48] have emerged recently. They share the common idea of learning the node representation via acquiring neighborhood information in the bipartite graph layer by layer, thus capturing the multi-hop connectivity of users/items [41].

Despite the inspiring progress made in graph-based recommendation, these approaches are inherently challenged by the issues of efficiency and scalability. They are intrinsically raised by the computation-intense message-passing of graph convolution in the existing training paradigm of graph-based recommendation. Such problems are further exaggerated in the real-world application of large-scale graphs as the time/computation complexity will grow exponentially with the number of users and items. Recent studies show that simple MLPs as the initialization of graph model [8, 47] or trained with contrastive learning [11], knowledge distillation

[49] demonstrate competitive performance compared with GCN models as long as they share an equivalent weight space. Considering that one can trivially derive a counterpart light graph model [9] given the matrix factorization (MF) [26] weight, we naturally raise a meaningful and significant question: *Do we really need computation-intense graph convolution during training for recommendation?*

To address the inquiry, we first conducted a preliminary experiment to investigate the role of graph convolution. The results reveal that graph convolution has a more pivotal role in testing rather than in training. Notably, the MF model is capable of matching the performance of the GCN when a similar lightweight graph convolution [9] is implemented after training. To uncover the underlying reasons from a training viewpoint, we examined the supervision alignment force when training with MF and LightGCN models, finding that the alignment property [33, 35] of positive user-item pairs is approximate in two distinct training paradigms. This prompted us to further explore the training processes of the MF and GCN models, leading us to conclude that GCN-based training essentially acts as a degree-weighted form of MF training. Intuitively, by following the pairwise alignment force from a depth-first search (DFS) perspective, MF training results in effects akin to GCN training that adopts information aggregation based on breadth-first search (BFS). Given the time demand of these processes, we suggest that graph convolution may not be necessary during training. However, the current graph convolution method is suboptimal, as we empirically find that the increasing number of layers significantly enlarges the difference between embeddings before and after convolution, denoted as the **Embedding Discrepancy**. Assuming the MF model is well-trained, any post-training operations should not significantly alter the original embedding space, whereas the existing convolution strategy with high embedding discrepancy may potentially offset the benefits of higher-order information. Moreover, the existing coarse-grained graph convolution approaches fail to find an optimal convolution depth due to its discrete characteristics. These motivate us to seek a more fine-grained method to integrate higher-order user-item interactions while avoiding computation-intense message passing during training.

In this paper, we introduce Light Post-Training Graph-ODE (LightGODE), a novel graph-based method designed for fine-grained and efficient large-scale RecSys. Specifically, we first propose a novel Post-Training Graph Convolution (PTGC) paradigm that significantly improves training efficiency by skipping the most time-consuming operations, including adjacency matrix normalization and layer-by-layer graph convolutions, making the training process as efficient as for traditional MF models. To tackle the issue of embedding discrepancy, we develop a non-parametric graph convolution that incorporates the self-loop during the information update. This straightforward operation will prioritize the preceding layers, thereby implicitly assigning greater importance to shallow layers, particularly the initial embeddings during graph convolution, which aids in minimizing the variation between the embedding spaces before and after graph convolution. Thus, it helps to reduce the distribution discrepancy between the embedding space. Building on this foundation, we propose a continuous graph ordinary-differential-equation derived from the discrete parameter-free graph convolution. The continuity offers several advantages. First, it characterizes the continuous dynamics of user/item representations within the bipartite graph, making the traditional graph convolution a specific discretization of seamless layer-wise embedding transformation. Additionally, it enables precise and fine-grained graph convolution to achieve the optimal trade-off with the continuous-time value to capture high-order information while balancing the embedding discrepancy. To foster future research and development of LightGODE, we have released it open-source, available at https://github.com/DavidZWZ/LightGODE. Here, we summarize our contributions as follows:

- To the best of our knowledge, we are the first to challenge the long-standing authority in the graph-based recommendation - the necessity of graph convolution, and we empirically and analytically reveal its decisive role in testing rather than training.
- We developed a novel post-training graph convolution framework for extremely efficient training and devised a none-parametric GCN with self-loop, alleviating the embedding discrepancy issue.
- Originally, we proposed a continuous graph ordinary-differential-equation (LightGODE), which allows dynamic modeling of node representations and achieving optimal trade-offs of high-order information and embedding discrepancy.
- We conduct extensive experiments on three real-world datasets to test the effectiveness of LightGODE, demonstrating the highest recommendation performance with the lowest training time.

2 Investigation of the Graph Convolution for Recommendation

In this section, we initially investigate the necessity of graph convolution for recommendation and examine the key reasons behind the unexpectedly superior performance of the MF model enhanced by post-training graph convolution. Subsequently, we pinpoint the trade-offs in designing post-training graph convolution by identifying the embedding discrepancy issues when constructing deeper graph convolution layers.

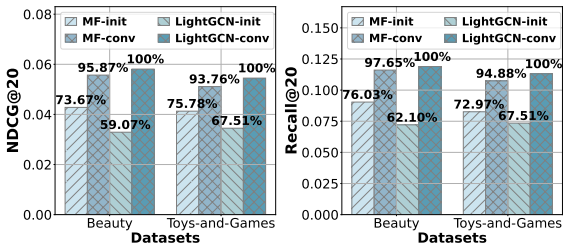

Figure 1: Preliminary study on the role of graph convolution for recommendation in training and testing stages. The MF model with graph convolution after training (MF-conv) achieves competitive results with the LightGCN-conv.

2.1 The Role and Necessity of Graph Convolution during Training

To investigate the necessity of graph convolution for graph-based RecSys, we conduct preliminary experiments on the Amazon-Beauty (denoted as Beauty) and the Amazon-Toys-and-Games (denoted as

Toys-and-Games) datasets to understand the impact of graph convolution in the training/testing stages of recommendation. Specifically, we design four variations of the model with the same amount of embedding parameters, including MF-init, which involves training with traditional matrix factorization and testing with its factorized ID embeddings; MF-conv, which integrates the 2-hops LightGCN convolution after MF training; LightGCN-init, which tests only with initial embeddings from a LightGCN model; and LightGCN-conv, which fully implements a 2-layer LightGCN model architecture.

As illustrated in Figure 1, we establish the LightGCN model (LightGCN-conv) as the benchmark by setting its performance as 100%. To our surprise, MF-conv consistently outperforms both MF-init and Light-conv across the two datasets, achieving an impressive average of over 95% of the performance metrics compared to LightGCN. This clearly highlights the substantial benefits of integrating post-training graph convolution with MF initialization, which significantly reduces computational costs by circumventing the intricate graph convolution process. Furthermore, these results underscore that the improved performance of graph-based RecSys primarily arises from the graph convolution after training, which prompts a thorough reconsideration of the necessity of the graph convolution during the training phase. Meanwhile, we propose a new point of view to understand the underlying reasons behind the unexpected exceptional performance of the MF-conv model even trained without graph convolution.

2.2 The Alignment Force: A DFS Perspective

Recommendation losses typically aim to identify the potential positive interactions via applying a supervised alignment force to positive user-item pairs during training. In this context, we empirically evaluated the alignment property [33, 35] (the average distance between normalized positive embeddings) of four model versions in Section 2.1 across the Beauty and Toys-and-Games datasets.

Table 1: The alignment property of positive pairs in training.

Training	Beauty		Toys-and-Games	
	Initial	Conv.	Initial	Conv.
MF	0.7952	0.6631	0.8100	0.7033
LightGCN	0.8270	0.6594	0.7761	0.6503

In Table 1, the initial ID embeddings for both MF and LightGCN exhibit approximate alignment values in Beauty and Toys-and-Games, suggesting comparable training effects with and without lightweight graph convolution. Similarly, the embeddings of post-training convolution show closely matched values, complying with the experimental findings of their comparable performance levels in Figure 1. These observations prompt us to explore whether the alignment forces exerted on user-item pairs, with and without the light graph convolution, are effectively equivalent.

Analytically, when a graph model is optimized by the same objective as the MF model, the alignment force applied on surrounding neighbors of positive pairs with graph convolution is the degree-weighted version of that alignment directly forced on two clusters of nodes. The assumption and proof are listed in the Appendix C.

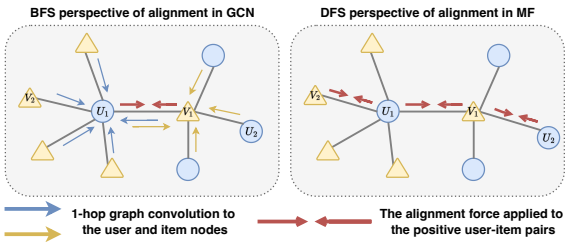

Figure 2: A comparison of alignment force in GCN-based and MF-based models from BFS and DFS, respectively.

Intuitively, in an illustrative scenario where a single-layer graph convolution network is employed for gathering information, as depicted in the left section of Figure 2, one observes the subgraph comprising positive pairs U_1 and V_1 alongside their neighboring nodes. When the alignment force acts upon U_1 and V_1, the representations of their respective adjacent nodes, such as V_2 and U_2, also move closer together. Conversely, in the right portion of Figure 2, an alternative approach is showcased where, instead of employing BFS for neighborhood aggregation, direct connections between U_1 and nodes V_2 and U_2 are established via DFS paths among various user-item pairs, achieving a comparable outcome in terms of aligned representation learning. This conclusion further weakens the necessity of time-intensive graph convolution in training.

2.3 Trade-off in Designing Graph Convolution

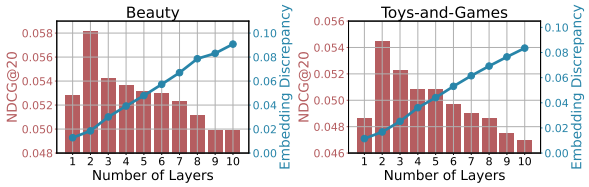

Figure 3: Study of the trade-off of embedding discrepancy and high-order information on Beauty and Toys-and-Games.

In the preceding sections, we highlighted the notable efficacy of the MF-conv model and analyzed the alignment properties to pinpoint key contributors to its approximate performance of LightGCN-conv. However, the MF-conv still slightly lags behind the LightGCN-conv, and it is not yet clear how to design a more effective non-parametric graph convolution in the post-training stage. The hint in the experiments (Figure 1) suggests that incorporating the multi-hop connectivity information during the testing phase proves beneficial. Then, it would be valuable to investigate whether higher-order connectivity continues to be advantageous after training with the MF model. In addition, if a model is optimally trained to fit the user-item interactions, one would expect the training embedding distribution to be ideal for testing. Consequently, any post-training operations should minimally impact the original embedding space. We are particularly interested in exploring how the model's performance correlates with the differences between initial and convolution embeddings, termed as **Embedding Discrepancy**.

We utilize the average Euclidean distance, widely acknowledged for numerical shifts [7], to quantify distribution shift across all users/items in the embedding space during graph convolution.

In Figure 3, we empirically increase the layer number of post-training graph convolution, and the performance peaks at a two-hop convolution. Surprisingly, incorporating more complex, higher-order information mostly leads to a performance decline. Additionally, the discrepancy between initial and convolution embeddings enlarges with more layers, indicating that the existing graph convolution strategy can disrupt the foundational training, potentially causing over-smoothing [29] as layers increase (Figure 3). This suggests that while additional convolution layers introduce more high-order information, they also risk perturbing well-trained embeddings. This could explain the counterexample as increasing the number of convolution layers initially promotes the performance and then continually brings negative effects - the current strategy finds a balance of configuring two layers.

To enhance performance, it is crucial to incorporate higher-order information while maintaining an embedding distribution close to that of the original MF model by adding more layers. This necessitates a more nuanced graph convolution approach that delicately constructs layers to maintain a trade-off between high-order structure information and the embedding discrepancy issue. In a comprehensive view of efficiency and effectiveness, we design a more fine-grained approach to balance the convolution depth and embedding discrepancy, as introduced in the following section.

3 Light Post-Training Graph-ODE for Efficient Recommendation

In this section, we propose the post-training graph convolution framework, including the pre-training user/item embeddings for extremely efficient graph recommendation. To balance the integration of high-order information and the risk of embedding discrepancy, we devise a non-parametric graph convolution with self-loop. Based on the formulation, we propose that LightGODE - a continuous post-training graph convolution based on ordinary-differential-equations aiming to achieve the optimal trade-off. Finally, a detailed time complexity analysis and comparison with other strong GCN baselines are demonstrated.

3.1 Pre-training User/Item Embedding

Here, we outline our overall training pipeline toward the extremely efficient graph-based recommendation. Since the graph convolution proved unnecessary in the training stage as Section 2, we abandoned the graph convolution-related operations and focused solely on training the randomly initialized ID embeddings, as shown in the training section of Figure 4. Regarding the loss computation, we directly optimize the alignment and uniformity properties as in [33] to reach optimal status for MF embedding training as an ideal foundation for the subsequent graph convolution phase. Specifically, the alignment loss minimizes the distance between the normalized embeddings of the positive pairs $(\mathbf{u}_i, \mathbf{v}_j)$ within batch $\mathcal{B}$:

$$\mathcal{L}_{align} = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{u}_i, \mathbf{v}_j) \in \mathcal{B}} \left\| \mathbf{u}_i - \mathbf{v}_j \right\|^2. \tag{1}$$

The uniformity loss $\mathcal{L}_{uniform} = (\mathcal{L}^U_{uniform} + \mathcal{L}^V_{uniform})/2$, and the user-side uniformity is given by:

$$\mathcal{L}^U_{uniform} = \log \frac{1}{|\mathcal{B}_u|^2} \sum_{\mathbf{u}_i \in \mathcal{B}_u} \sum_{\mathbf{u}_{i'} \in \mathcal{B}_u} e^{-2\|\mathbf{u}_i - \mathbf{u}_{i'}\|}, \tag{2}$$

where $\mathcal{B}_u$ is the user batch and $\mathbf{u}_{i'}$ is rest of users in batch. The item side uniformity $\mathcal{L}^U_{uniform}$ follows the same format and final loss becomes $\mathcal{L} = \mathcal{L}_{align} + \gamma \mathcal{L}_{uniform}$ adjusted by weight γ.

3.2 Discrete GCN with Self-Loop

Empirical evidence in [9, 38] and Section 2 suggests that optimal performance is typically achieved when the graph model is configured with two or three layers. However, abruptly discontinuing the convolution process at higher-order layers is inappropriate since neither the preceding shallow layers are distinctively treated nor the subsequent high-order layers are noticed. Such an approach lacks a seamless transition from lower to higher-order graph convolutions, potentially overlooking nuanced differences in structural information embedded in shallow and deep graph relationships. This calls for reconsidering the graph convolution process across different layer depths to better capture the complexity and dynamics of the graph data in recommendation contexts.

One straightforward solution is to integrate the self-loop (SL) into the graph convolution process. This simple operation highlights the importance of the node representations of preceding layers in each message-passing process, contributing to a gradual transition to higher-order connectivity. Suppose we observe a pair of interacted users and items with corresponding initial input ID embeddings $\mathbf{u}_i^0$ and $\mathbf{v}_j^0$, and we design the parameter-free graph convolution based on the smoothed neighborhood aggregation process as in [9]. The graph convolution with SL is finalized as:

$$\begin{aligned} \mathbf{u}_i^k &= \mathbf{u}_i^{k-1} + \sum_{j \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}} \mathbf{v}_j^{k-1}, \\ \mathbf{v}_j^k &= \mathbf{v}_j^{k-1} + \sum_{i \in N_j} \frac{1}{\sqrt{|N_j|}\sqrt{|N_i|}} \mathbf{u}_i^{k-1}, \end{aligned} \tag{3}$$

where $\mathbf{u}_i^{k-1}$ and $\mathbf{v}_j^{k-1}$ are embeddings of user $\mathbf{u}_i$ and item $\mathbf{v}_j$ at layer of $k-1$, respectively. The normalization employs the average degree $\frac{1}{\sqrt{|N_i|}\sqrt{|N_j|}}$ to temper the magnitude of popular nodes after graph convolution. Afterward, the collaborative filtering final embedding is obtained by synthesizing the layer-wise representations:

$$\mathbf{u}_i^{(K)} = \sum_{k=0}^{K} \mathbf{u}_i^k; \quad \mathbf{v}_j^{(K)} = \sum_{k=0}^{K} \mathbf{v}_j^k. \tag{4}$$

3.3 Continuous Graph-ODE

Motivated by [2, 31, 42] deriving continuous differential equations from diffusion process to model the dynamics in the graph, we aim to design a continuous version of our discrete non-parametric graph convolution with self-loops.

Formally, given $\mathbf{h}_0$ as the initial embedding of the users and items, we can rewrite the layer-wise information update Equation 3 in terms of the matrix operations:

$$\mathbf{h}_k = \mathbf{A}\mathbf{h}_{k-1}, \tag{5}$$

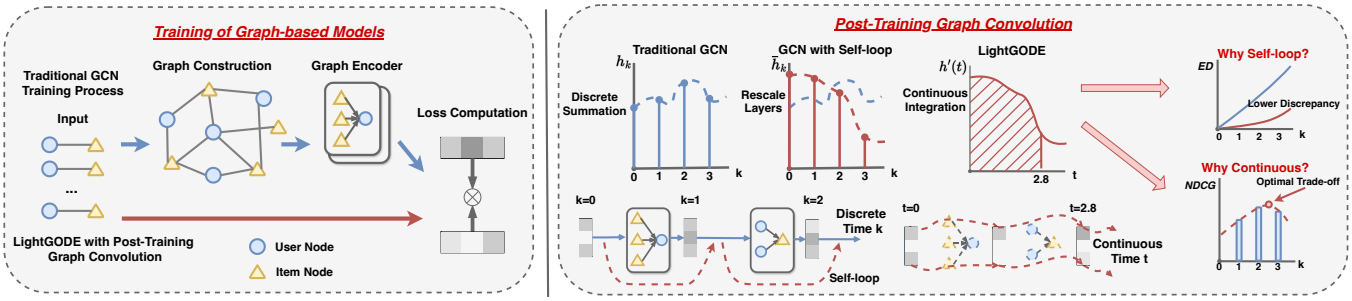

Figure 4: The training pipeline of traditional GCN-based recommendation and our proposed LightGODE with post-training graph convolution (PTGC) framework, where we skip the time-consuming convolution-related operations to speed up the training. In the PTGC stage, the self-loop prioritizes the shallow layers by weighing more on preceding layer representations, thus mitigating the distribution discrepancy problem. Based on the design of discrete non-parametric GCN, we derive LightGODE, a continuous ODE function that implements fine-grained graph convolution to achieve the optimal trade-off in the GCN design.

where $\mathbf{h}_k$ is the node embeddings of k-th layer, aggregating their neighborhood information and fusing with its own representation of the previous layer via self-loop. The matrix $\mathbf{A} = \bar{\mathbf{A}} + \mathbf{I}$ and $\bar{\mathbf{A}}$ is the normalized adjacency matrix.

Consequently, the end result for a K-layer discrete graph convolution network $\mathbf{h}(K)$ can be represented as:

$$\mathbf{h}(K) = \sum_{k=0}^{K} \mathbf{h}_k = \sum_{k=0}^{K} \mathbf{A}^k \mathbf{h}_0. \tag{6}$$

This sum from Equation 6 can be seen as a Riemann sum extending from layer 0 to layer $K \to \infty$, transitioning to a continuous ODE function (proof provided in the Appendix B):

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \ln \mathbf{A}\mathbf{h}(t) + (\mathbf{A} - \ln \mathbf{A})\mathbf{h}_0, \tag{7}$$

which simplifies under a first-order Taylor expansion approximation where $\ln \mathbf{A} = \mathbf{A} - \mathbf{I} = \bar{\mathbf{A}}$, leading to:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \bar{\mathbf{A}}\mathbf{h}(t) + \mathbf{h}_0. \tag{8}$$

The general form of this continuous graph convolution network is obtained by integration from the initial condition as:

$$\mathbf{h}(t) = h_0 + \int_0^t [\bar{\mathbf{A}}\mathbf{h}(s) + \mathbf{h}_0]\mathrm{d}(s). \tag{9}$$

Note that the final integration form could be solved analytically using the integration factor. However, considering that computing the exponential of the matrix in the analytical solution is time-consuming, we resort to the simple and fast Euler solver [3] to approximate the ODE solution.

3.4 Time Complexity Analysis

In this subsection, we analyze the computation complexity of LightGODE and compare it with two prominent GCN benchmark methods, LightGCN [9] and GraphAU [47]. We first define the number of edges in the user-item bipartite graph as $|\mathcal{E}|$. Then, let K represent the number of graph convolution layers and d the size of embeddings. On this basis, we can derive the following facts:

- In the graph construction process, both LightGCN and GraphAU require normalization of the adjacency matrix. This step involves computing $2|\mathcal{E}|$ non-zero elements of the original adjacency matrix. On the contrary, LightGODE alleviates the need for graph construction and adjacency matrix normalization in training.
- In the graph convolution stage, LightGCN and GraphAU both perform linear message-passing through the graph's edges in each layer, which incurs a computational cost of $2|\mathcal{E}|Kd$. Whereas LightGODE does not involve graph convolution in training, significantly facilitating large-scale graph recommendations.
- Regarding the loss computation, LightGCN adopts the BPR loss for optimization, leading to a computational demand of $O(2Bd)$. GraphAU, on the other hand, uses alignment and uniformity loss calculations between users and items in the batch, resulting in a time complexity in the batch of $O(2KBd + 2B^2d)$. LightGODE, focusing only on the alignment loss at the initial embedding, has a time complexity per batch of $O(Bd + 2B^2d)$. It should be noted that all the experiments are implemented on GPU-based parallel computation, which minimizes the relative importance of batch size B in model comparisons. Furthermore, the BPR loss's reliance on negative sampling for each user-item pair in every batch through all epochs makes LightGCN less efficient than LightGODE in handling large-scale graphs.

Table 2: Time complexity comparison of LightGCN, GraphAU, and LightGODE during training.

Stages	LightGCN	GraphAU	LightGODE
Adjency Matrix	$O(2\|\mathcal{E}\|)$	$O(2\|\mathcal{E}\|)$	-
Graph Convolution	$O(2\|\mathcal{E}\|Kd)$	$O(2\|\mathcal{E}\|Kd)$	-
Loss Computation	$O(2Bd)$	$O(2KBd + 2B^2d)$	$O(Bd + 2B^2d)$

4 Experiments

4.1 Datasets

We experiment on three public real-world datasets: Gowalla, Amazon-Beauty (Beauty), and Amazon-Toys-and-Games (Toys-and-Games),

Table 3: Performance comparison on three benchmark datasets in terms of NDCG and Recall.

Method	Gowalla				Beauty				Toys-and-Games			
	N@20	R@20	N@50	R@50	N@20	R@20	N@50	R@50	N@20	R@20	N@50	R@50
BiasMF	0.0406	0.0700	0.0507	0.1122	0.0428	0.0904	0.053	0.1404	0.0413	0.0826	0.0503	0.1271
NeuMF	0.0487	0.0952	0.0637	0.1597	0.0343	0.0746	0.043	0.1173	0.0301	0.0632	0.0375	0.0994
NGCF	0.0501	0.0923	0.0644	0.1535	0.0438	0.0943	0.0559	0.1537	0.0379	0.0827	0.0486	0.1356
DGCF	0.0553	0.0967	0.0692	0.1556	0.0516	0.1081	0.0624	0.1610	0.0485	0.1007	0.0589	0.1515
SimpleX	0.0451	0.0876	0.0611	0.1555	0.0502	0.1104	0.0623	0.1697	0.0521	0.1092	0.0632	0.1640
LightGCN	0.0683	0.1224	0.0860	0.1974	0.0581	0.1189	0.0709	0.1816	0.0555	0.1131	0.0669	0.1696
ODE-CF	0.0680	0.1220	0.0854	0.1960	0.0537	0.1158	0.0661	0.1760	0.0516	0.1075	0.0633	0.1656
DirectAU	0.0768	0.1437	0.0978	0.2319	0.0555	0.1149	0.0673	0.1725	0.0571	0.1184	0.0677	0.1714
GraphAU	0.0811	0.1461	0.1017	0.2346	0.0662	0.1398	0.0782	0.2116	0.0622	0.1324	0.0725	0.1952
LightGODE	**0.0929**	**0.1678**	**0.1150**	**0.2628**	**0.0714**	**0.1452**	**0.0852**	**0.2130**	**0.0673**	**0.1371**	**0.0794**	**0.1983**

varying in scales and domains. The Gowalla [1] is a location-based social networking dataset obtained from users' checking-in. Beauty and Toys-and-Games are crawled from real-world data in Amazon [2] according to the product category. We follow the 5-core setting in [33, 51] by removing the users/items with node degrees less than five to ensure the data quality for testing. We split all datasets into training (80%), validation (10%), and testing (10%), and the statistical information of the three datasets after filtering is summarized in Table 4. More details about implementations, evaluations, and baseline can be found in Appendix A.

Table 4: The statistics of the datasets.

Dataset	# Users	# Items	# Interactions	Sparsity
Gowalla	64,116	164,533	2,018,421	99.9809%
Beauty	22,364	12,102	198,502	99.9267%
Toys-and-Games	19,413	11,925	167,597	99.9276%

4.2 Overall Performance Comparison

In this comprehensive experiment, we compare the performance of several state-of-the-art recommendation algorithms on three diverse datasets: Gowalla, Beauty, and Toys-and-Games, using NDCG20, NDCG@50, Recall@20, and Recall@50. Here, we highlight the main observations as follows:

- Noticeably, LightGODE achieves the highest scores in NDCG and Recall across all datasets, demonstrating its effectiveness in different recommendation tasks. It should highlighted that in the large-scale dataset Gowalla (with 64,116 users and 164,533 items), LightGODE surpasses all the other baseline methods by large margins with more than 10% improvement over the strongest baseline, emphasizing its potential to be deployed in the large-scale graphs in real-world applications.
- Among all, DirectAU and GraphAU emerge as the most competitive baselines in all three datasets, demonstrating the effectiveness of the alignment and uniformity [35] in optimization.
- Most of the graph-based recommender systems consistently outperform the traditional MF models. This suggests the importance of graph convolution for capturing the multi-hop information. Though leveraging contrastive learning loss, SimpleX performs poorly in the context of sparse dataset Gowalla, whereas LightGCN and ODE-CF are more robust across all datasets.

4.3 Ablation Study

Ablation studies on LightGODE are conducted to validate the rationality and effectiveness of our design choices. From Table (5), it is evident that the full version of the LightGODE model achieves the best scores across all metrics and datasets, showcasing the efficacy of the continuous ODE function. Furthermore, using only our parameter-free graph convolution with self-loop (w/o ODE) still results in higher NDCG and Recall and lower embedding discrepancy compared to traditional lightweight graph convolution (w/o SL), indicating more consistent embeddings. The model without post-training graph convolution (w/o Conv) exhibits the lowest performance. Therefore, each component within our LightGODE contributes significantly to the final recommendation performance.

Table 5: Abalation study on different components. The embedding discrepancy (ED) is the Euclidean distance between initial and convolution embeddings; the lower the better.

Dataset	Metrics	Light-GODE	w/o ODE	w/o SL	w/o Conv
Gowalla	NDCG	**0.0929**	0.0833	0.0801	0.0768
	Recall	**0.1678**	0.1537	0.1481	0.1437
	ED	**0.0066**	0.0158	0.0282	-
Beauty	NDCG	**0.0714**	0.0700	0.0686	0.0555
	Recall	**0.1452**	0.1450	0.1428	0.1149
	ED	**0.0022**	0.0049	0.0082	-
Toys-and-Games	NDCG	**0.0673**	0.0644	0.0641	0.0571
	Recall	**0.1371**	0.1343	0.1337	0.1184
	ED	**0.0011**	0.0045	0.0067	-

[1] https://snap.stanford.edu/data/loc-gowalla.html
[2] https://jmcauley.ucsd.edu/data/amazon/links.html

4.4 Efficiency Analysis

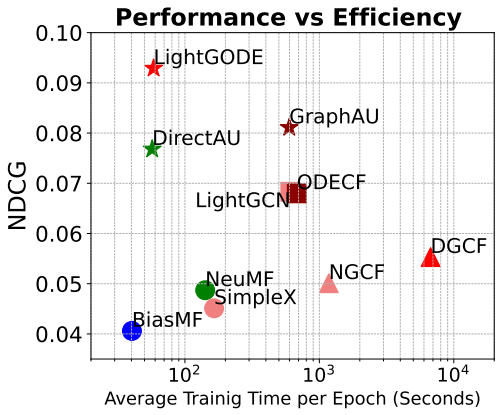

Figure 5: Trade-off between the performance and the efficiency on the Gowalla dataset. The left upper direction indicates stronger performance and more efficient training.

4.4.1 ***Trade-off between the Performance and the efficiency.*** Figure 5 illustrates the overall comparison of performance and efficiency on the large Gowalla dataset. LightGODE markedly outperforms all benchmarks while maintaining high efficiency, underscoring its potential for effective, large-scale recommendation systems. Early works that leverage GCN encoders, such as NGCF and DGCF, fall behind in average training times per epoch and NDCG. More advanced GCN-based approaches, including LightGCN, ODECF, and GraphAU, show substantial improvements in NDCG scores yet are still much slower than simpler MF models in speed. Conversely, BiasMF, NeuMF, and SimpleX directly utilize user-item interactions, achieving notably low training times but exhibiting poor ranking scores. Only DirectAU manages a balanced trade-off but still lags behind LihgtGODE regarding NDCG.

4.4.2 ***Training Time Comparison.*** To delve deeper into the efficiency and scalability analysis in terms of the training, we provide a comprehensive training time comparison featuring the average time per epoch, the number of required epochs, and the total training cost shown in Table 6. On the large dataset Gowalla, NGCF and DGCF consume longer times per epoch for training, taking tens of hours in total to reach the optimal status. LightGCN and ODECF demonstrate shorter epoch duration but demand a greater number of epochs to complete training. Although GraphAU exhibits the fastest training speed per epoch among the baseline methods in the Beauty and Toys-and-Games dataset, it is almost as slow as LightGCN and ODECF, especially on the large Gowalla dataset. LightGODE significantly reduces the overall training time to less than one hour on Gowalla. These observations highlight the efficiency and scalability of LightGODE towards industrial RecSys.

4.4.3 ***Performance Curve and Convergence Speed.*** In Figure 6, we present the training curves of performance against epochs on the tree datasets. Overall, NGCF and DGCF exhibit low-performance peaks, while LightGCN and ODECF converge slowly. By enforcing alignment and uniformity in the representation hyperspace, the performances of GraphAU and LightGODE ensure convergence at early training stages. Our method requires fewer epochs to converge and consistently results in high recommendation scores.

Table 6: Training time comparison of GCN-based models on Gowalla, Beauty, and Toys-and-Games datasets. It includes the average training time per epoch, the number of epochs, and the total training time. For abbreviation, we denote seconds as s, minutes as m, and hours as h.

Dataset	Method	Time/Epoch	# Epochs	Total Time
Gowalla	NGCF	1175.79s	84	27.44h
	DGCF	6720.88s	43	80.28h
	LightGCN	608.69s	105	17.75h
	ODECF	679.25s	79	14.91h
	GraphAU	597.11s	91	15.09h
	LightGODE	58.46s	61	**0.99h**
Beauty	NGCF	10.79s	48	8.63m
	DGCF	204.95s	72	245.94m
	LightGCN	6.76s	83	9.35m
	ODECF	8.34s	108	15.01m
	GraphAU	8.72s	41	5.96m
	LightGODE	3.16s	68	**3.58m**
Toys-and -Games	NGCF	11.19s	57	10.63m
	DGCF	116.69s	61	118.64m
	LightGCN	5.01s	119	9.94m
	ODECF	6.60s	147	16.17m
	GraphAU	7.09s	41	4.84m
	LightGODE	2.65s	76	**3.36m**

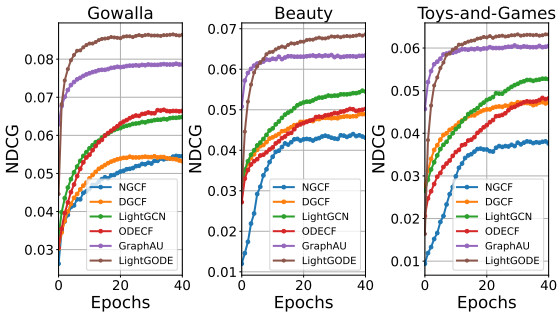

Figure 6: Performance curve in the first 40 epochs.

4.5 Comparison with GODE

To evaluate the effectiveness of our continuous ODE function and self-loop operations tailored for post-training graph convolution, we compare LightGODE with post-training graph convolution and GODE with pre-training graph convolution. LightGODE consistently emerges as the superior performer across all metrics in all three datasets, especially in the large-scale dataset Gowalla. This demonstrates that our innovative design for post-training graph convolution not only allows our simple model to exceed the performance of GCN models trained with traditional graph convolution but also significantly speeds up the training strategy.

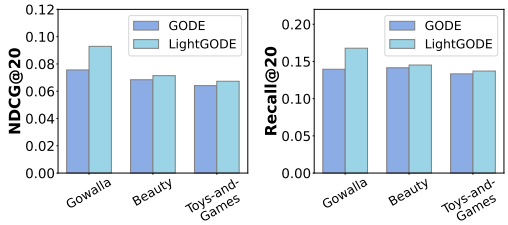

Figure 7: Performance Comparison of LightGODE and GODE.

4.6 Hyperparameter Analysis

4.6.1 ***Impact of the Time*** t. We evaluate how the continuous time t affects the performance of LightGODE. As observed in Figure 8, it suggests that an appropriate time t is generally associated with the scale of the datasets. In Gowalla, performance peaked at around 3 as a larger time t enables the model to long-distance neighborhood aggregations. Whereas in Beauty and Toys-and-Games, t is chosen at 1.8 and 0.8, indicating a smaller receptive field to achieve the optimal convolution depth.

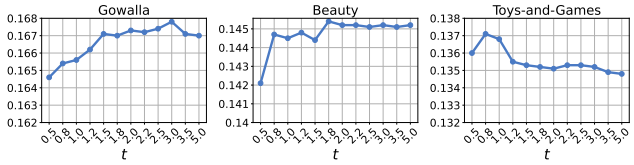

Figure 8: Impact the time t on Recall.

4.6.2 ***Impact of the uniformity weight*** γ. Another hyperparameter is the weight of the uniformity loss γ. From the curves in Figure 9, a smaller uniformity weight (0.5) achieves the highest recall on the Beauty and Toys-and-Games datasets. In contrast, larger values of γ are detrimental to the recommendation performance in Gowalla datasets. As for large-scale datasets, the user and item representations should be more evenly distributed so as to make the user/item embeddings more representative for distinction.

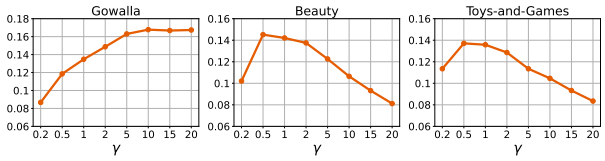

Figure 9: Impact of the uniformity weight γ on Recall.

5 Related Work

5.1 Graph Convolution Network for RecSys

Collaborative filtering (CF) is widely used to provide recommendations based on user-item interactions. Recent developments in graph convolution networks (GCNs) have reformed CF from conventional matrix factorization CF to Graph-based CF, incorporating social networks [6, 21, 46], knowledge graph [1, 20, 36], and user-item interactions [9, 25, 37, 38, 45, 48, 50]. One of the early attempts is NGCF [37], which incorporates the importance of high-order connectivity in the user-item bipartite graph. Another early work, PinSAGE [48], utilizes random-walk to sample subgraphs and scales up RecSys industrial level. DGCF [38] disentangles latent intentions of users and diversifies item recommendations. LightGCN [9] omits linear transformations and nonlinear activations in GCN layers and drastically improves the efficiency of graph recommendation. To further simplify the graph training, UltraGCN [25] adjusts the relative importance of nodes to aggregate the embeddings by weights and directly approximates the converged state of message passing, which accelerates LightGCN by more than multiple times. GraphAU [47] identifies the inefficiency of DirectAU [33] on graph-based recommendations and proposes high-order representation alignment for the linear scale of computation for additional layers. All previous work aims to improve the training efficiency from the graph convolution process, whereas we innovatively challenge the necessity of time-intensive graph convolution and propose the extremely efficient post-training graph convolution framework.

5.2 Graph Ordinary Differential Equation

Neural ordinary-differential-equations (NODE) [4] propose a new paradigm that models continuous dynamics through the derivative of the neural network's hidden state. Motivated by this, graph ordinary-differential-equations (GDE) [28] combines the concepts with GCN and directly treats the GCN layer as a continuous vector field. Derived from the diffusion process, continuous graph model CGNN [42] characterizes the dynamics of node representations using a continuous message-passing layer. Concurrently, in the continuous time data, ODE with a graph encoder [12, 13, 22] are developed for modeling interacting dynamics. For RecSys, LT-OCF [5] first introduces ordinary-differential-equations into graph collaborative filtering and learns the optimal layer combination. Then ODECF [44] condenses multiple GCN layers into one continuous layer for better performance and efficiency. In comparison, instead of using a deep neural network to parametrize the ODE derivative, we derive the continuous graph ODE based on the discrete non-parametric graph convolution for efficient recommendation.

6 Conclusion

In this study, we critically challenge the conventional reliance on graph convolution in the training of graph RecSys by demonstrating that their primary benefits are realized during the testing phase. We propose the Light Post-Training Graph-ODE (LightGODE), which innovatively skips traditional resource-heavy convolution processes, and devise a novel continuous graph ordinary-differential-equation model to mitigate the embedding discrepancy for optimal convolution depth. Our empirical evaluations across three real-world datasets, especially on the large-scale dataset Gowalla, show that LightGODE significantly outperforms traditional CF models in both recommendation performance and computational efficiency. This work not only questions existing training paradigms but also pinpoints potentially new research directions for efficient and large-scale graph RecSys.

Acknowledgments

This work is supported in part by NSF under grants III-2106758, and POSE-2346158

A Experimental Setup

A.1 Baselines

- **BiasMF** [18] is a matrix factorization technique that integrates bias vectors for both users and items for enhanced prediction.
- **NeuMF** [10] leverages deep neural networks to model the complex and non-linear interactions between users and items.
- **NGCF** [37] introduces GCN models with collaborative filtering to exploit the rich user-item interaction graph structure.
- **DGCF** [38] utilizes a disentangled representation learning approach to exploit distinct factors of user-item interactions.
- **SimpleX** [24] propose a novel cosine contrastive loss function to be integrated with simple collaborative filtering models.
- **LightGCN** [9] simplifies the GCN architecture for a recommendation via obviating the complex non-linear operation.
- **ODECF** [44] presents a neural ODE-based model that can skip multiple GCN layers to reach the final representation.
- **DirectAU** [33] explores to directly optimize the alignment and uniformity of latent representations in collaborative filtering.
- **GraphAU** [47] extend the alignment loss layer-wise and tailor for graph encoders for efficient graph recommendation.

A.2 Evaluation Metrics

For evaluating performance metrics, we use NDCG@K and Recall@K to ensure a fair comparison among all baseline methods in the top-K recommendation tasks. In all experiments, K is set to 20 by default unless specified. We employ the full-ranking strategy [52] for all experiments, meaning that all candidate items not previously interacted by the user will be ranked during testing.

A.3 Implementation Details

Our implementation of LightGODE and all baseline models are carried out using RecBole [53]. For the baseline training, we meticulously search their hyperparameters for various datasets following respective original papers to ensure a fair comparison. The batch size and the embedding size are standardized at 256 and 64, respectively. All models employ the Adam optimizer [16], with a learning rate set at 1e-3. To prevent overfitting, we utilize an early stopping mechanism that terminates training if there is a consistent decline in the performance metric NGCG@20 over 10 epochs. Specifically, for our method LightGODE, we tune the uniformity weight γ within the set [0.2, 0.5, 1, 2, 5, 10, 15, 20] and search time t in the range of [0.5, 0.8, 1.0, 1.2, 1.5, 1.8, 2.0, 2.2, 2.5, 3.0, 3.5, 5.0] for optimal performance. To maintain impartiality in our efficiency evaluations, each model is trained independently using a single GPU.

B Derivation of Continuous ODE

One can view final representations in Equation 6 as a Riemann sum:

$$\mathbf{h}^{(K)} = \sum_{k=1}^{K+1} \mathbf{A}^{(k-1)\Delta t} h_0 \Delta t, \tag{10}$$

with $\Delta t = \frac{t+1}{K+1}$. In this discrete setup, $t = K$ and thus $\Delta t = 1$ for the discrete graph convolution networks. Now let $K \to \infty$, the equation transitions to its continuous form:.

$$\mathbf{h}(t) = \int_0^{t+1} \mathbf{A}^{\mathbf{s}} \mathbf{h_0} \, \mathrm{d}s. \tag{11}$$

Differentiating this, we have:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \mathbf{A}^{t+1}\mathbf{h}_0. \tag{12}$$

Given the challenges in computing $\mathbf{A}^{t+1}$ for non-integer values of t, it is reformulated into an ODE using the second derivative:

$$\frac{\mathrm{d}^2\mathbf{h}(t)}{\mathrm{d}t^2} = \ln \mathbf{A}\mathbf{A}^{t+1}\mathbf{h}_0 = \ln \mathbf{A}\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} \tag{13}$$

The ODE integrates to:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \ln \mathbf{A}\mathbf{h}(t) + X_{const}, \tag{14}$$

Applying $t = 0$ to Equation 12 and Equation 14 gives:

$$\left.\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t}\right|_{t=0} = \mathbf{A}\mathbf{h}_0 = \ln \mathbf{A}\mathbf{h}_0 + X_{const}. \tag{15}$$

Therefore,

$$X_{const} = (\mathbf{A} - \ln \mathbf{A})\mathbf{h}_0, \tag{16}$$

resulting in the ODE formulation for the graph convolution as:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \ln \mathbf{A}\mathbf{h}(t) + (\mathbf{A} - \ln \mathbf{A})\mathbf{h}_0. \tag{17}$$

C Analysis on the Alignment Force

Definition C.1 (Perfect Alignment). A pair of observed user-item pair is perfectly aligned if $e_u = e_v$ and $(u, v) \sim P_{pos}$

To simplify the derivation process, we consider the given user-item pair (u, v) perfectly aligned, and the number of users is less than the number of items. The lower bound of the alignment force for the MF model is held as:

$$\begin{aligned}
\mathcal{L}_{align-mf} &= \|e_u^0 - e_v^0\|^2 + \sum_{i \in N_u} \|e_i^0 - e_u^0\|^2 + \sum_{j \in N_v} \|e_v^0 - e_j^0\|^2 \\
&\geq \sum_{i \in N_u} \|e_i^0 - e_u^0\|^2 + \sum_{j \in N_v} \|e_v^0 - e_j^0\|^2 \\
&\geq \left\| \sum_{i \in N_u} (e_i^0 - e_u^0) + \sum_{j \in N_v} (e_v^0 - e_j^0) \right\|^2 \\
&\geq \left\| \sum_{i \in N_u} e_i^0 - \sum_{j \in N_v} e_j^0 \right\|^2,
\end{aligned} \tag{18}$$

where e_u^0 and e_v^0 represents the initial embedding of user u and item v, with i and j being their neighboring nodes. Assuming the GCN model employs light convolution as in [9] for neighborhood aggregation, the alignment force for a 1-layer graph convolution for the user-item pair (u, v) can be described as:

$$\begin{aligned}
\mathcal{L}_{align-gcn} &= \left\| e_u^1 - e_v^1 \right\|^2 \\
&= \left\| \sum_{i \in N_u} \frac{e_i^0}{\sqrt{|N_u|}\sqrt{|N_i|}} - \sum_{j \in N_v} \frac{e_j^0}{\sqrt{|N_u|}\sqrt{|N_i|}} \right\|^2.
\end{aligned} \tag{19}$$

Therefore, comparing Equation 18 and Equation 19, the alignment force applied on surrounding neighbors of positive pairs using graph convolution is the weighted version of the direct alignment force applied on two clusterings of nodes.

References

[1] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*. 151–161.

[2] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. 2021. Grand: Graph neural diffusion. In *International Conference on Machine Learning*. PMLR, 1407–1418.

[3] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems* (2018).

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems* (2018), 6571–6583.

[5] Jeongwhan Choi, Jinsung Jeon, and Noseong Park. 2021. LT-OCF: Learnable-time ode-based collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 251–260.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[7] Igor Goldenberg and Geoffrey I Webb. 2019. Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems* 60, 2 (2019), 591–615.

[8] Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, and Neil Shah. 2023. MLPInit: Embarrassingly Simple GNN Training Acceleration with MLP Initialization. In *The Eleventh International Conference on Learning Representations*.

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[11] Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. 2021. Graph-mlp: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051* (2021).

[12] Zijie Huang, Yizhou Sun, and Wei Wang. 2020. Learning continuous system dynamics from irregularly-sampled partial observations. *Advances in Neural Information Processing Systems* 33 (2020), 16177–16187.

[13] Zijie Huang, Yizhou Sun, and Wei Wang. 2021. Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 705–715.

[14] Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. 2018. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications* 28 (2018), 94–101.

[15] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*. 135–142.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] Yehuda Koren, Steffen Rendle, and Robert Bell. 2021. Advances in collaborative filtering. *Recommender systems handbook* (2021), 91–142.

[20] Xiaolong Liu, Liangwei Yang, Zhiwei Liu, Mingdai Yang, Chen Wang, Hao Peng, and Philip S Yu. 2024. Knowledge Graph Context-Enhanced Diversified Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 462–471.

[21] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2022. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–24.

[22] Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. 2023. Hope: High-order graph ode for modeling interacting dynamics. In *International Conference on Machine Learning*. PMLR, 23124–23139.

[23] Jing Ma, Liangwei Yang, Qiong Feng, Weizhi Zhang, and Philip S Yu. 2023. Graph-based Village Level Poverty Identification. In *Proceedings of the ACM Web Conference 2023*. 4115–4119.

[24] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A simple and strong baseline for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1243–1252.

[25] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 1253–1262.

[26] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[27] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 325–341.

[28] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. 2021. Graph Neural Ordinary Differential Equations. arXiv. https://doi.org/10.48550/arXiv.1911.07532 arXiv:1911.07532 [cs, stat].

[29] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. 2023. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993* (2023).

[30] Poonam B Thorat, Rajeshwari M Goudar, and Sunita Barve. 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications* 110, 4 (2015), 31–36.

[31] Matthew Thorpe, Tan Minh Nguyen, Heidi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. 2022. GRAND++: Graph neural diffusion with a source term. In *International Conference on Learning Representation (ICLR)*.

[32] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. *Advances in neural information processing systems* 26 (2013).

[33] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 1816–1825.

[34] Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee, and Liangjie Hong. 2020. Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 645–653.

[35] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*. PMLR, 9929–9939.

[36] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[38] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1001–1010.

[39] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-Modal Self-Supervised Learning for Recommendation. In *Proceedings of the ACM Web Conference 2023*. 790–800.

[40] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[41] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[42] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. 2020. Continuous graph neural networks. In *International conference on machine learning*. PMLR, 10432–10441.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[44] Ke Xu, Yuanjie Zhu, Weizhi Zhang, and S Yu Philip. 2023. Graph Neural Ordinary Differential Equations-based method for Collaborative Filtering. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1445–1450.

[45] Chenxiao Yang, Qitian Wu, Jiahua Wang, and Junchi Yan. 2023. Graph Neural Networks are Inherently Good Generalizers: Insights by Bridging GNNs and MLPs. In *The Eleventh International Conference on Learning Representations*.

[46] Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, and Philip S Yu. 2021. Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation. In *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*. 2141–2145.

[47] Liangwei Yang, Zhiwei Liu, Chen Wang, Mingdai Yang, Xiaolong Liu, Jing Ma, and Philip S Yu. 2023. Graph-based alignment and uniformity for recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4395–4399.

[48] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[49] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2021. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *International Conference on Learning Representations*.

[50] Weizhi Zhang, Liangwei Yang, Yuwei Cao, Ke Xu, Yuanjie Zhu, and S Yu Philip. 2023. Dual-Teacher Knowledge Distillation for Strict Cold-Start Recommendation. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 483–492.

[51] Weizhi Zhang, Liangwei Yang, Zihe Song, Henry Peng Zou, Ke Xu, Yuanjie Zhu, and Philip S Yu. 2024. Mixed Supervised Graph Contrastive Learning for Recommendation. *arXiv preprint arXiv:2404.15954* (2024).

[52] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting alternative experimental settings for evaluating top-n item recommendation algorithms. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2329–2332.

[53] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *CIKM*.