# Revisiting LightGCN: Unexpected Inflexibility, Inconsistency, and A Remedy Towards Improved Recommendation

Geon Lee
KAIST
Seoul, Republic of Korea
geonlee0325@kaist.ac.kr

Kyungho Kim
KAIST
Seoul, Republic of Korea
kkyungho@kaist.ac.kr

Kijung Shin
KAIST
Seoul, Republic of Korea
kijungs@kaist.ac.kr

## Abstract

Graph Neural Networks (GNNs) have emerged as effective tools in recommender systems. Among various GNN models, LightGCN is distinguished by its simplicity and outstanding performance. Its efficiency has led to widespread adoption across different domains, including social, bundle, and multimedia recommendations. In this paper, we thoroughly examine the mechanisms of LightGCN, focusing on its strategies for scaling embeddings, aggregating neighbors, and pooling embeddings across layers. Our analysis reveals that, contrary to expectations based on its design, LightGCN suffers from inflexibility and inconsistency when applied to real-world data.

We introduce LightGCN++, an enhanced version of LightGCN designed to address the identified limitations. LightGCN++ incorporates flexible scaling of embedding norms and neighbor weighting, along with a tailored approach for pooling layer-wise embeddings to resolve the identified inconsistencies. Despite its remarkably simple remedy, extensive experimental results demonstrate that LightGCN++ significantly outperforms LightGCN, achieving an improvement of up to 17.81% in terms of NDCG@20. Furthermore, state-of-the-art models utilizing LightGCN as a backbone for item, bundle, multimedia, and knowledge-graph-based recommendations exhibit improved performance when equipped with LightGCN++.

## CCS Concepts

• **Information systems → Recommender systems**.

## Keywords

Recommender Systems, Graph Neural Networks, LightGCN

**ACM Reference Format:**
Geon Lee, Kyungho Kim, and Kijung Shin. 2024. Revisiting LightGCN: Unexpected Inflexibility, Inconsistency, and A Remedy Towards Improved Recommendation. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3640457.3688176

## 1 Introduction & Related Work

Recommender systems are crucial in web applications, assisting users navigate the vast amount of available information. They predict user preferences based on interaction patterns between users and items, and recently, their capabilities have been remarkably enhanced by integrating graph neural networks (GNNs) [11].

Among GNN-based recommender systems, LightGCN [6] stands out for its effectiveness and efficiency, enhancing recommendation performance by removing complexities unnecessary for recommendation (spec., feature transformation and nonlinearities). Due to its high efficacy, LightGCN has been widely adopted across various recommendation domains (e.g., social [15, 25, 31, 32], bundle (i.e., itemset) [4, 9, 18, 23], and multimedia [10, 22, 34] recommendations). Furthermore, it has been integrated as a fundamental component in further enhanced methods [12, 16, 17, 24, 30, 33, 36].

Inspired by the broad applicability of LightGCN, we conduct an in-depth investigation into its core mechanisms. We thoroughly examine its strategies for scaling embedding norms, aggregating neighbors, and pooling embeddings across layers. Our analysis reveals that, when applied to real-world data, LightGCN exhibits notable inflexibility and inconsistency in its operations, contrary to expectations based on their formulation. Specifically, we observe that rigidity in embedding norms leads to inflexible near-uniform weighting across neighbors and inconsistent disparities between layers in LightGCN, which may limit its effectiveness.

Building on these insights, we propose LightGCN++, an enhanced version of LightGCN. LightGCN++ offers a simple yet powerful remedy, introducing flexibility in norm scaling and neighbor weighting along with adjustable layer-wise embedding pooling. By addressing the identified inflexibility and inconsistency of LightGCN, our experimental results show that LightGCN++ enhances recommendation performance across diverse datasets. Notably, the versatility of LightGCN++ enables its adoption in diverse state-of-the-art recommendation models across various domains.

In summary, this paper makes the following contributions:

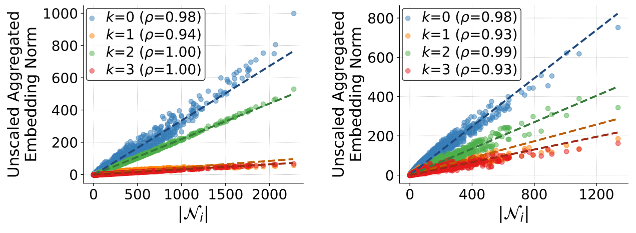
- **Analysis and observation.** We thoroughly analyze LightGCN and uncover unexpected inflexibility and inconsistency which could potentially limit its effectiveness.
- **Remarkably simple yet powerful remedy.** We develop LightGCN++, which introduces flexible norm scaling, neighbor weighting, and adjustable layer-wise pooling for addressing LightGCN's limitations while preserving its inherent strengths.
- **Experiments.** Extensive experiments demonstrate that LightGCN++ mitigates LightGCN's limitations and significantly improves the recommendation performance.

**Code and datasets:** https://github.com/geon0325/LightGCNpp.

## 2 Analysis of LightGCN

In this section, we examine LightGCN [6], which is one of the most successful and widely used GNN-based recommendation models.

*RecSys '24, October 14–18, 2024, Bari, Italy*
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688176



**(a) MovieLens** **(b) Amazon**

**Figure 1: The norm of the unscaled aggregated embedding ($\sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{-0.5}\mathbf{e}_u^{(k)}$) tends to be proportional to the number of neighbors $|\mathcal{N}_i|$ for $k \geq 0$. The symbol $\rho$ represents the Pearson correlation coefficient. More results are in [13].**

## 2.1 Review: Definition of LightGCN

We first briefly review two main components of LightGCN.

**Aggregation.** LightGCN adopts simple neighbor aggregation by removing feature transformation and non-linear activations. Its aggregation at each $k^{\text{th}}$ layer is as follows:

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_u|}}\mathbf{e}_u^{(k)}. \tag{1}$$

For brevity, we focus on the aggregation rule for item embeddings (note its symmetry with that for user embeddings). Note that the symmetric normalization term $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ is used in Eq. (1).

**Pooling.** After aggregating embeddings through $K$ layers, the intermediate embeddings from each layer are combined to construct the final user/item embeddings, which are utilized for making predictions. Specifically, LightGCN pools embeddings as follows:

$$\mathbf{e}_u = \sum\nolimits_{k=0}^{K} \omega_k \mathbf{e}_u^{(k)}, \text{ where } \omega_k = \frac{1}{K+1}, \forall k \in \{0, 1, \cdots, K\}. \tag{2}$$

This means that mean pooling (i.e., equal importance for each layer's embedding) is used to aggregate the embeddings.

## 2.2 Basic Analysis: Dual Effects of Normalization

Now, we closely examine neighbor aggregation at each layer. The aggregation rule for each item $i$ in Eq. (1) can be rewritten as:

$$\mathbf{e}_i^{(k+1)} = \frac{1}{\sqrt{|\mathcal{N}_i|}} \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)}, \tag{3}$$

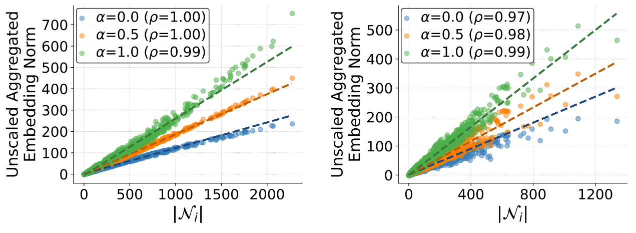
which we divide its normalization term into two components: $1/\sqrt{|\mathcal{N}_i|}$ (i.e., left term) and $1/\sqrt{|\mathcal{N}_u|}$ (i.e., right term). Our analysis reveals their distinct roles.

**Role of the left term.** The left term, $1/\sqrt{|\mathcal{N}_i|}$, plays a role in scaling the norm of the aggregated embedding $\mathbf{e}_i^{(k+1)}$. The norm of the embedding is derived as $\frac{1}{\sqrt{|\mathcal{N}_i|}}\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)} \right\|$, i.e., the norm of the scaled aggregated embedding is the product of $1/\sqrt{|\mathcal{N}_i|}$ and the norm of the unscaled aggregated embedding.

**Role of the right term.** The right term, $1/\sqrt{|\mathcal{N}_u|}$, which is applied individually to each neighboring user $u \in \mathcal{N}_i$, determines the "influence" of each neighbor $u$ on $i$. While one might presume that this *explicit* term $1/\sqrt{|\mathcal{N}_u|}$ solely determines the influence of $u$, we argue that such a view overlooks the fact that the norms of user embeddings (e.g., $\|\mathbf{e}_u^{(k)}\|$) may vary across users, which *implicitly* affect the influence. We can rewrite Eq. (3) as follows:

$$\mathbf{e}_i^{(k+1)} = \frac{1}{\sqrt{|\mathcal{N}_i|}} \sum_{u \in \mathcal{N}_i} \frac{\|\mathbf{e}_u^{(k)}\|}{\sqrt{|\mathcal{N}_u|}} \frac{\mathbf{e}_u^{(k)}}{\|\mathbf{e}_u^{(k)}\|}.$$

Essentially, the actual influence of each neighbor $u$ on $i$ is more accurately described as $\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|}$, which we refer to as the *effective weight* of a neighbor $u$. It accounts for both explicit (i.e., $1/\sqrt{|\mathcal{N}_u|}$) and implicit (i.e., $\|\mathbf{e}_u^{(k)}\|$) influences of the neighbor.



**(a) MovieLens** **(b) Amazon**

**Figure 2: The norm of the unscaled aggregated embedding ($\sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{-\alpha}\mathbf{e}_u^{(k)}$) tends to be proportional to the number of neighbors $|\mathcal{N}_i|$ for various $\alpha$'s. The symbol $\rho$ represents a Pearson correlation coefficient. More results are in [13].**

## 2.3 Primary Empirical Observation

When applied to real-world datasets, we observe a near-linear relationship between the norms of the unscaled aggregated embeddings and the numbers of neighbors, as shown in Figure 1, i.e., for $k \geq 0$,

$$\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)} \right\| \mathrel{\overset{\sim}{\propto}} |\mathcal{N}_i|, \tag{4}$$

where $\mathrel{\overset{\sim}{\propto}}$ denotes a strong positive linear correlation. This property is further extended to the generalized right term, $1/|\mathcal{N}_u|^{\alpha}$. Specifically, $\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{\alpha}}\mathbf{e}_u^{(k)} \right\| \mathrel{\overset{\sim}{\propto}} |\mathcal{N}_i|$ for various $\alpha$ values, as shown in Figure 2. We theoretically explore the potential rationale behind these observations in [13].

## 2.4 In-Depth Analysis of LightGCN

In this deeper analysis, we uncover unexpected inflexibility and inconsistency in the embedding behavior of LightGCN.

**How LightGCN scales embedding norms.** Based on our primary observation (Eq. (4)), we derive the norm of the scaled aggregated embedding $\mathbf{e}_i^{(k+1)}$, for any $k \geq 0$, as follows:

$$\left\|\mathbf{e}_i^{(k+1)}\right\| = \frac{1}{\sqrt{|\mathcal{N}_i|}}\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)} \right\| \mathrel{\overset{\sim}{\propto}} \frac{1}{\sqrt{|\mathcal{N}_i|}}|\mathcal{N}_i| = \sqrt{|\mathcal{N}_i|}. \tag{5}$$

This is empirically confirmed in Figure 3 where the embedding norms $\|\mathbf{e}_i^{(k)}\|$ exhibit strong positive linear correlation with $\sqrt{|\mathcal{N}_i|}$ for $k \geq 1$. Notably, this property does not hold when $k = 0$, as the initial embeddings $\mathbf{e}_i^{(0)}$ are not subject to normalization or neighbor aggregation. This results in **inconsistency** in norm scaling between embeddings at $k = 0$ and $k \geq 1$, which should be carefully addressed when pooling embeddings across layers, as discussed later.

**How LightGCN aggregates neighbors.** As discussed in Section 2.2, the effective weight of a neighboring user $u \in \mathcal{N}_i$ is $\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|}$.

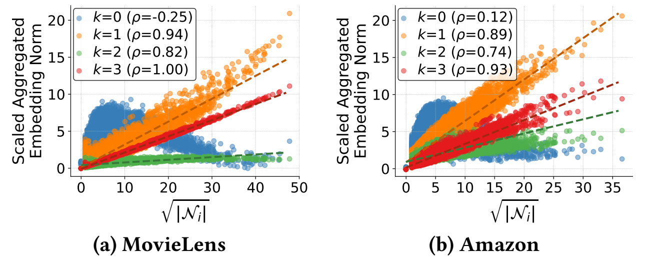


**Figure 3: LightGCN exhibits inconsistency in norm scaling. The norm of the scaled aggregated embedding ($|\mathcal{N}_i|^{-0.5}\sum_{u\in\mathcal{N}_i}|\mathcal{N}_u|^{-0.5}\mathbf{e}_u^{(k)}$) tends to be proportional to $\sqrt{|\mathcal{N}_i|}$ when $k \geq 1$, but not when $k = 0$. The symbol $\rho$ represents a Pearson correlation coefficient. More results are in [13].**

Interestingly, LightGCN's norm scaling property (Eq. (5)) leads to a near-uniform effective weight across neighbors when $k \geq 1$:

$$\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|} \propto \sqrt{|\mathcal{N}_u|}/\sqrt{|\mathcal{N}_u|} = 1.$$

This near-uniformity, empirically confirmed in Figure 4, is surprising since one might expect that a higher degree neighbor $u$ would have less influence due to the term $1/\sqrt{|\mathcal{N}_u|}$. However, when $k \geq 1$, all neighbors, regardless of their degrees, are aggregated with nearly identical effective weights. This **inflexibility** implies that LightGCN may not properly account for the varying importance of neighbors, which can be crucial for improving recommendations. In contrast, when $k = 0$, there is no discernible pattern between $\|\mathbf{e}_u^{(k)}\|$ and $\sqrt{|\mathcal{N}_u|}$, and due to the denominator $\sqrt{|\mathcal{N}_u|}$, the effective weights tend to decrease with the degree of neighbors.

**How LightGCN pools embeddings.** LightGCN applies identical weights to embeddings across layers, meaning it combines embeddings at $k = 0$ and $k \geq 1$ with a fixed weight ratio of $1 : K$ (Eq. (2)). However, recall the inconsistent norm scaling properties of embeddings at $k = 0$ and $k \geq 1$. This notable disparity indicates that employing an inflexible ratio in their combination may result in suboptimal recommendation performance.

## 3 Proposed Remedy: LightGCN++

Based on our understanding of LightGCN, we present LightGCN++ which retains the core advantages of LightGCN while addressing its inflexibility and inconsistency encountered in real-world data.

**Aggregation.** To address the inflexibility in norm scaling and neighbor weighting of LightGCN, we design LightGCN++'s neighbor aggregation rule as follows:

$$\mathbf{e}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|^{\alpha}} \sum_{u\in\mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{\beta}} \frac{\mathbf{e}_u^{(k)}}{\|\mathbf{e}_u^{(k)}\|}, \tag{6}$$

where $\alpha$ and $\beta$ are controllable hyperparameters. Compared to $1/\sqrt{|\mathcal{N}_i|}$ in LightGCN, the left term $1/|\mathcal{N}_i|^{\alpha}$ offers more flexibility in norm scaling. Recall that the norm of the unscaled aggregated embedding is proportional to the number of neighbors, regardless of the weight of each neighbor (Figure 2). Thus, the norm of the scaled aggregated embedding is controllable by $\alpha$, i.e., $\|\mathbf{e}_i^{(k)}\| \propto |\mathcal{N}_i|^{1-\alpha}$ for $k \geq 1$. Each neighbor embedding is normalized before aggregation, and thus each neighbor $u$ is assigned an effective weight of $1/|\mathcal{N}_u|^{\beta}$. The non-zero term $\beta$ biases the weighting either toward low-degree neighbors ($\beta > 0$) or high-degree neighbors ($\beta < 0$), addressing the inflexible near-uniform weighting issue in LightGCN. Moreover, the effective weight remains $1/|\mathcal{N}_u|^{\beta}$ for both $k = 0$ and $k \geq 1$, ensuring consistency across all layers. This flexible and controllable neighbor weighting allows weight distributions to be finely customized for a target dataset.

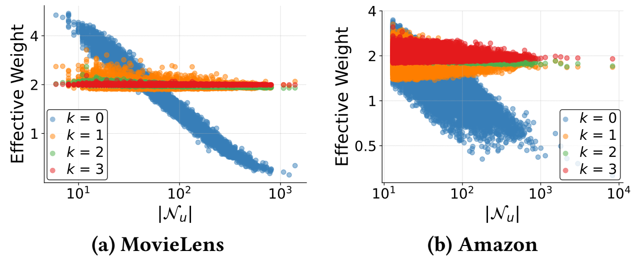


**Figure 4: LightGCN exhibits inflexibility in neighbor weighting. The effective weight $\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|}$ of every neighbor $u$ tends to be near-uniform for $k \geq 1$. More results are in [13].**

**Pooling.** To account for the aforementioned inconsistency in norm scaling between embeddings at $k = 0$ and $k \geq 1$ (Figure 3), LightGCN++ adjusts the layer-wise weights for pooling as follows:

$$\mathbf{e}_i = \gamma\mathbf{e}_i^{(0)} + (1-\gamma)\frac{1}{K}\sum_{k=1}^{K}\mathbf{e}_i^{(k)}, \tag{7}$$

where $\gamma \in [0, 1]$ is a adjustable hyperparameter. By adaptively controlling $\gamma$, we aim to better determine the relative importance of embeddings at each layer. While LightGCN fixes $\gamma$ to $1/(K+1)$, we show in Section 4 that this may not be the optimal choice.

**Comparison with LightGCN enhancements.** We compare LightGCN++ with recent enhancements to LightGCN [26, 28, 35], which primarily aim to address the inflexibility in norm scaling of LightGCN, for example, by introducing $\alpha$ in Eq. (6). Commonly, these methods do not pay attention to the effective weights of neighbors, which can be expressed as $\|\mathbf{e}_u^{(k)}\|/|\mathcal{N}_u|^{\beta'}$ (specifically, ALGCN [28] fixes $\beta'$ to 1, AdjNorm [35] constrains $\beta'$ to $1-\alpha$, and SSM-GNN [26] allows flexibility with $\beta'$). **Firstly**, compared to the effective weight of $1/|\mathcal{N}_u|^{\beta}$ in LightGCN++, which is explicitly determined by the neighbor popularity $|\mathcal{N}_u|$, the effective weight of $\|\mathbf{e}_u^{(k)}\|/|\mathcal{N}_u|^{\beta'}$ includes $\|\mathbf{e}_u^{(k)}\|$, which implicitly affects the influence and thus results in less controllable effective weights. It is important to note that, ALGCN [28] and AdjNorm [35] exhibit inflexible near-uniform effective weights, similar to LightGCN, as shown empirically in [13]. **Secondly**, the effective weights of all these methods are inconsistent at layers $k = 0$ and $k \geq 1$, especially due to the arbitrary effective weights at $k = 0$. **Thirdly**, they all adopt mean pooling of layer-wise embeddings, and thus do not address the inconsistency in norm scaling between layers at $k = 0$ and $k \geq 1$. **Lastly, and most importantly**, all these methods are outperformed by LightGCN++, which effectively addresses these limitations, as empirically demonstrated in Section 4.

**Complexity Analysis** We analyze the complexity of LightGCN++. Since it does not introduce any extra trainable parameters compared to LightGCN, its $O(|\mathcal{V}|Kd)$ space complexity remains unchanged. The time complexity of each phase of LightGCN++ is:

- The time complexity of computing the degree of every node (i.e., $|\mathcal{N}_i|$'s and $|\mathcal{N}_u|$'s) is $O(|\mathcal{E}|)$.

**Table 1: LightGCN++ consistently and significantly outperforms LightGCN in terms of Recall@20 and NDCG@20. State-of-the-art methods enhanced with LightGCN++ (i.e., NCL++, SimGCL++, and XSimGCL++) also outperform their counterparts with LightGCN. For each dataset, the best performance is in bold and the second-best is underlined.**

| Dataset | LastFM | | MovieLens | | Gowalla | | Yelp | | Amazon | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| BPRMF [20] | 0.2014 | 0.1891 | 0.2098 | 0.2692 | 0.1335 | 0.1040 | 0.0367 | 0.0301 | 0.0296 | 0.0223 |
| NeuMF [7] | 0.2182 | 0.2064 | 0.2167 | 0.2737 | 0.1231 | 0.0991 | 0.0375 | 0.0296 | 0.0236 | 0.0178 |
| NGCF [21] | 0.2334 | 0.2190 | 0.2280 | 0.2894 | 0.1364 | 0.1081 | 0.0462 | 0.0369 | 0.0318 | 0.0236 |
| LR-GCCF [2] | 0.1980 | 0.1919 | 0.1651 | 0.2252 | 0.0967 | 0.0829 | 0.0392 | 0.0322 | 0.0167 | 0.0136 |
| HCCF [27] | 0.2242 | 0.2128 | 0.2198 | 0.2881 | 0.1328 | 0.1152 | 0.0627 | 0.0510 | 0.0338 | 0.0256 |
| UltraGCN [19] | 0.2494 | 0.2528 | 0.2551 | 0.3172 | 0.1702 | 0.1408 | 0.0625 | 0.0507 | 0.0367 | 0.0278 |
| LightGCL [1] | 0.2525 | 0.2428 | 0.2344 | 0.2951 | 0.1703 | 0.1425 | 0.0623 | 0.0506 | 0.0418 | 0.0316 |
| ALGCN [28] | 0.2408 | 0.2318 | 0.2404 | 0.3050 | 0.1635 | 0.1376 | 0.0538 | 0.0435 | 0.0349 | 0.0259 |
| AdjNorm [35] | 0.2532 | 0.2434 | 0.2419 | 0.3054 | 0.1635 | 0.1344 | 0.0554 | 0.0448 | 0.0379 | 0.0282 |
| SSM-GNN [26] | 0.2612 | 0.2523 | 0.2515 | 0.3124 | 0.1639 | 0.1380 | 0.0602 | 0.0491 | 0.0373 | 0.0278 |
| LightGCN [6] | 0.2523 | 0.2427 | 0.2392 | 0.3010 | 0.1683 | 0.1426 | 0.0553 | 0.0449 | 0.0367 | 0.0274 |
| LightGCN++ | 0.2715$^{**}$ | 0.2624$^{**}$ | 0.2616$^{**}$ | 0.3275$^{**}$ | 0.1739$^{**}$ | 0.1469$^{**}$ | 0.0650$^{**}$ | 0.0529$^{**}$ | 0.0394$^{**}$ | 0.0294$^{**}$ |
| Improvement | 7.60% | 8.11% | 9.36% | 8.80% | 3.32% | 3.01% | 17.54% | 17.81% | 7.35% | 7.29% |
| NCL [16] | 0.2548 | 0.2453 | 0.2401 | 0.3027 | 0.1704 | 0.1430 | 0.0584 | 0.0475 | 0.0393 | 0.0293 |
| NCL++ | 0.2721$^{**}$ | 0.2632$^{**}$ | **0.2621**$^{**}$ | **0.3285**$^{**}$ | **0.1759**$^{**}$ | **0.1478**$^{**}$ | **0.0678**$^{**}$ | **0.0553**$^{**}$ | 0.0424$^{**}$ | 0.0315$^{**}$ |
| Improvement | 6.78% | 7.29% | 9.16% | 8.52% | 1.87% | 1.81% | 16.09% | 16.42% | 7.88% | 7.50% |
| SimGCL [33] | 0.2602 | 0.2494 | 0.2584 | 0.3217 | 0.1703 | 0.1424 | 0.0650 | 0.0528 | 0.0415 | 0.0314 |
| SimGCL++ | 0.2723$^{**}$ | 0.2617$^{**}$ | 0.2615$^{**}$ | 0.3276$^{**}$ | 0.1704 | 0.1431 | 0.0657$^{**}$ | 0.0536$^{**}$ | 0.0444$^{**}$ | 0.0334$^{**}$ |
| Improvement | 4.65% | 4.93% | 1.19% | 1.83% | 0.05% | 0.49% | 1.07% | 1.70% | 6.98% | 6.36% |
| XSimGCL [30] | 0.2614 | 0.2508 | 0.2600 | 0.3245 | 0.1678 | 0.1400 | 0.0651 | 0.0528 | 0.0397 | 0.0298 |
| XSimGCL++ | **0.2738**$^{**}$ | **0.2638**$^{**}$ | 0.2613 | 0.3270$^{*}$ | 0.1705$^{**}$ | 0.1432$^{**}$ | 0.0674$^{**}$ | 0.0549$^{**}$ | **0.0454**$^{**}$ | **0.0342**$^{**}$ |
| Improvement | 4.74% | 5.18% | 0.50% | 0.77% | 1.60% | 2.28% | 3.53% | 3.97% | 14.35% | 14.76% |

**Table 2: LightGCN++ enhances recommendation performance across various domains, specifically, bundle, multimedia, and knowledge graph recommendations, demonstrating its versatility and applicability.**

**(a) Bundle Recommendation**

| Dataset | Youshu | NetEase | iFashion |
|---|---|---|---|
| CrossCBR [18] | 0.1584 | 0.0359 | 0.0778 |
| CrossCBR++ | **0.1625**$^{**}$ | **0.0361** | **0.0847**$^{**}$ |
| Improv. | 2.58% | 0.55% | 8.88% |

**(b) Multimedia Recommendation**

| Dataset | Clothing | Sports | Baby |
|---|---|---|---|
| LATTICE [34] | 0.0316 | 0.0428 | 0.0364 |
| LATTICE++ | **0.0322** | **0.0463**$^{**}$ | **0.0402**$^{**}$ |
| Improv. | 1.90% | 8.18% | 10.44% |

**(c) Knowledge Graph Recommendation**

| Dataset | Yelp | Amazon | MIND |
|---|---|---|---|
| KGCL [29] | 0.0470 | 0.0737 | 0.0519 |
| KGCL++ | **0.0475** | **0.0782**$^{**}$ | **0.0551**$^{**}$ |
| Improv. | 1.06% | 6.11% | 6.17% |

- While LightGCN takes $O(|\mathcal{E}|Kd)$ time for neighborhood aggregation and $O(|\mathcal{V}|Kd)$ time for layer-wise pooling. LightGCN++ additionally requires embedding normalization at each layer, with a time complexity of $O(|\mathcal{V}|Kd)$ in total.
- The time complexity of computing the BPR loss is $O(|\mathcal{E}|d)$.

## 4 Experimental Results

In this section, we review our experiments, whose results support the effectiveness of LightGCN++.

### 4.1 Experimental Settings

**Datasets.** We use five benchmark datasets: LastFM [8], MovieLens [5], Gowalla [3, 14], Yelp [6, 16], and Amazon [6, 16]. Each is divided into training, validation, and test sets using a ratio of 7:1:2.
**Baselines.** We evaluate LightGCN++ and various recommender systems, including state-of-the-art methods: BPRMF [20], NeuMF [7], NGCF [21], LR-GCCF [2], HCCF [27], UltraGCN [19], LightGCL [1], ALGCN [28], AdjNorm [35], SSM-GNN [26], LightGCN [6], NCL [16], SimGCL [33], and XSimGCL [30]. Specifically, NCL, SimGCL, and XSimGCL employ LightGCN as their backbone.
**Implementations.** LightGCN++ is implemented based on the official PyTorch source code of LightGCN. The embedding dimension is set to 64, the batch size is set to 2,048, and the learning rate is set to 0.001 with a regularization coefficient of 0.0001. We tune $\alpha$, $\beta$, and $\gamma$ for each dataset using validation sets. For GNN-based models, including LightGCN++, we set the number of layers $K$ to 2.
**Evaluation metrics.** We use two metrics, NDCG@$N$ and Recall@$N$ to evaluate top-$N$ recommendations. We set $N$ to 20, unless otherwise stated, and results for $N = 10$ and $N = 40$ are in [13]. In each setting, we conduct ten trials and report the average performance.
**Details.** Refer to [13] for more detailed experimental settings.

### 4.2 Experimental Results

We conduct comparative analyses on five datasets to evaluate the recommendation performance of LightGCN++ and its integration with state-of-the-art methods. In Tables 1 and 2, * and ** indicates that $p < 0.01$ and $p < 0.001$, respectively, for a one-tailed t-test.
**Accuracy.** First, as shown in Table 1, LightGCN++ consistently and significantly outperforms LightGCN across all datasets. This indicates that LightGCN++, by effectively addressing the empirical inflexibility and inconsistency of LightGCN, is capable of yielding more accurate recommendations. Additionally, we evaluate the performance of state-of-the-art methods that use LightGCN as their backbone model after replacing it with LightGCN++. They are named NCL++, SimGCL++, and XSimGCL++. Integrating LightGCN++ leads to improved performance for these models across all datasets. In addition, LightGCN++ and the methods integrate it consistently secure the top two ranks among the compared methods, demonstrating the effectiveness of LightGCN++.

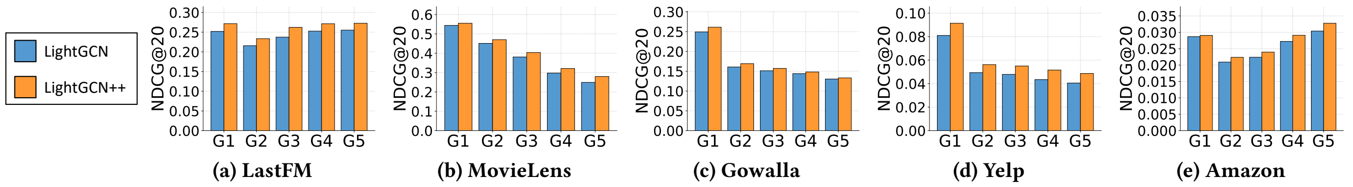


**Figure 5: LightGCN++ provides more accurate recommendations for users with varying levels of sparsity, compared to LightGCN.**

**Table 3: Each design choice in LightGCN++ contributes to its improved performance (in terms of NDCG@20).**

| Dataset | LastFM | MovieLens | Gowalla | Yelp | Amazon |
|---|---|---|---|---|---|
| LightGCN | 0.2427 | 0.3010 | 0.1426 | 0.0449 | 0.0274 |
| $\text{LightGCN}_{++}^{\text{w/o }\alpha}$ | 0.2580 | **0.3275** | 0.1435 | 0.0498 | 0.0291 |
| $\text{LightGCN}_{++}^{\text{w/o }\beta}$ | 0.2536 | **0.3275** | 0.1463 | 0.0516 | 0.0280 |
| $\text{LightGCN}_{++}^{\text{w/o }\gamma}$ | 0.2614 | 0.3217 | 0.1436 | 0.0507 | 0.0282 |
| LightGCN++ | **0.2624** | **0.3275** | **0.1469** | **0.0529** | **0.0294** |

**Versatility.** We evaluate LightGCN++ across different recommendation domains: bundle, multimedia, and knowledge graph. By replacing LightGCN used in CrossCBR [18], LATTICE [34], and KGCL [29], with LightGCN++, we enhance them to CrossCBR++, LATTICE++, and KGCL++. As shown in Table 2, LightGCN++ enhances the performance of these models in their respective domains, demonstrating the versatility and applicability of LightGCN++. Furthermore, following [1], we classify users into five groups (G1 to G5) based on their degrees. Each group has the same degree sum, with G1 representing users with the highest degrees and G5 representing those with the lowest. As shown in Figure 5, LightGCN++ consistently outperforms LightGCN across all user groups. This indicates that LightGCN++ is proficient at recommending items to users with varying levels of interactions.

**Effectiveness.** To verify the effectiveness of each design choice of LightGCN++, we assess the performance of the following variants:

- $\textbf{LightGCN}_{++}^{\textbf{w/o }\alpha}$ uses a fixed value of $\alpha = 0.5$ in Eq. (6), i.e., without flexible embedding norm scaling.
- $\textbf{LightGCN}_{++}^{\textbf{w/o }\beta}$ uses a fixed value of $\beta = 0$ in Eq. (6), i.e., without flexible neighbor weighting.
- $\textbf{LightGCN}_{++}^{\textbf{w/o }\gamma}$ uses a fixed value of $\gamma = 1/(K+1)$, i.e., mean-pooling, without flexible layer-wise embedding pooling.

As shown in Table 3, LightGCN++ outperforms all its variants, confirming the efficacy of LightGCN++'s design choices.

**Parameter analysis.** We examine the influence of the controllable hyperparameters $\alpha$, $\beta$, and $\gamma$ on the performance of LightGCN++. In Figure 6, we evaluate the performance of LightGCN++ for $\alpha \in \{0.0, 0.1, \cdots, 1.0\}$, $\beta \in \{-0.25, -0.2, \cdots, 0.25\}$, and $\gamma \in \{0.0, 0.1, \cdots 1.0\}$. Our observations indicate that the impact of the values of $\alpha$, $\beta$, and $\gamma$ varies across datasets, and thus their flexible and proper adjustment is crucial for improved recommendation. In [13], we offer more results and potential rationales for the chosen hyperparameters in relation to the graph structure.

**Speed.** We evaluate the speed of LightGCN++. As shown in Table 4, LightGCN++ is marginally slower than LightGCN, with an increase in runtime ranging from 0.08% to 5.29%, as we can expect from our complexity analysis in Section 3. [1]

[1]Experiments were conducted on a computer with RTX 3090Ti GPUs.

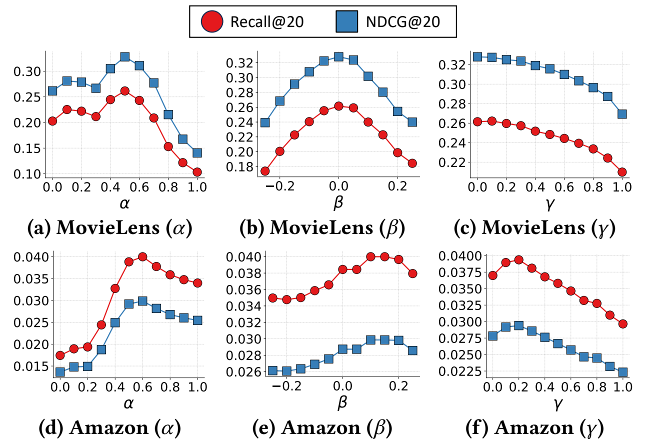


**Figure 6: The controllable hyperparameters $\alpha$, $\beta$, and $\gamma$ have varying impacts on different datasets, indicating the importance of their flexible and adaptive adjustment.**

**Table 4: Runtime (in seconds) of LightGCN and LightGCN++ per epoch. LightGCN++ is marginally slower than LightGCN.**

| Dataset | LightGCN | LightGCN++ | Increase |
|---|---|---|---|
| LastFM | 0.9137 | 0.9345 | 2.27 % |
| MovieLens | 10.0144 | 10.0232 | 0.08 % |
| Gowalla | 13.2507 | 13.3094 | 0.44 % |
| Yelp | 21.2809 | 22.0242 | 3.49 % |
| Amazon | 46.2871 | 48.7372 | 5.29 % |

## 5 Conclusions

In this paper, we conduct an in-depth analysis of LightGCN. Our analysis reveals that, contrary to expectations based on its mathematical formulation, LightGCN suffers from notable inflexibility and inconsistency when applied to real-world data. In response, we develop LightGCN++, a remedy that incorporates flexible adjustments in embedding norm scaling, neighbor weighting, and layer-wise pooling. Through extensive experiments, we demonstrate that LightGCN++ significantly outperforms LightGCN and enhances LightGCN-based models across multiple domains.

**Acknowledgements.** This research was supported in part by the NAVER-Intel Co-Lab. The work was conducted by KAIST and reviewed by both NAVER and Intel. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)).

## References

[1] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2022. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *ICLR*.

[2] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI*.

[3] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and Mobility: User Movement In Location-Based Social Networks. In *KDD*.

[4] Xiaoyu Du, Kun Qian, Yunshan Ma, and Xinguang Xiang. 2023. Enhancing Item-level Bundle Representation for Bundle Recommendation. *ACM Transactions on Recommender Systems* (2023).

[5] F. Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[8] Cantador Iván, Brusilovsky Peter, and Kuflik Tsvi. 2011. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems. In *RecSys*.

[9] Kyungho Kim, Sunwoo Kim, Geon Lee, and Kijung Shin. 2024. Towards Better Utilization of Multiple Views for Bundle Recommendation. In *CIKM*.

[10] Taeri Kim, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2022. MARIO: Modality-Aware Attention and Modality-Preserving Decoders for Multimedia Recommendation. In *CIKM*.

[11] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[12] Geon Lee, Kyungho Kim, and Kijung Shin. 2024. Post-Training Embedding Enhancement for Long-Tail Recommendation. In *CIKM*.

[13] Geon Lee, Kyungho Kim, and Kijung Shin. 2024. Revisiting LightGCN: Unexpected Inflexibility, Inconsistency, and A Remedy Towards Improved Recommendation (Supplementary Document). https://github.com/geon0325/LightGCNpp/blob/main/supplementary_document.pdf.

[14] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling User Exposure in Recommendation. In *WWW*.

[15] Jie Liao, Wei Zhou, Fengji Luo, Junhao Wen, Min Gao, Xiuhua Li, and Jun Zeng. 2022. SocialLGN: Light graph convolution network for social recommendation. *Information Sciences* 589 (2022), 595–607.

[16] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*.

[17] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware message-passing gcn for recommendation. In *WWW*.

[18] Yunshan Ma, Yingzhi He, An Zhang, Xiang Wang, and Tat-Seng Chua. 2022. CrossCBR: cross-view contrastive learning for bundle recommendation. In *KDD*.

[19] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *CIKM*.

[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[21] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.

[22] Yinwei Wei, Wenqi Liu, Fan Liu, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. 2023. Lightgt: A light graph transformer for multimedia recommendation. In *SIGIR*.

[23] Yinwei Wei, Xiaohao Liu, Yunshan Ma, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. 2023. Strategy-aware bundle recommender system. In *SIGIR*.

[24] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *MM*.

[25] Jiahao Wu, Wenqi Fan, Jingfan Chen, Shengcai Liu, Qing Li, and Ke Tang. 2022. Disentangled Contrastive Learning for Social Recommendation. In *CIKM*.

[26] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2022. On the effectiveness of sampled softmax loss for item recommendation. *ACM Transactions on Information Systems* 42, 4 (2022), 1–26.

[27] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *SIGIR*.

[28] Ronghai Xu, Haijun Zhao, Zhi-Yuan Li, and Chang-Dong Wang. 2023. ALGCN: Accelerated Light Graph Convolution Network for Recommendation. In *DASFAA*.

[29] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *SIGIR*.

[30] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2024. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 36, 2 (2024), 913–926.

[31] Junliang Yu, Hongzhi Yin, Min Gao, Xin Xia, Xiangliang Zhang, and Nguyen Quoc Viet Hung. 2021. Socially-aware self-supervised tri-training for recommendation. In *SIGIR*.

[32] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhancing social recommendation with adversarial graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3727–3739.

[33] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*.

[34] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. 2021. Mining latent structures for multimedia recommendation. In *MM*.

[35] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. 2022. Investigating accuracy-novelty performance for graph-based collaborative filtering. In *SIGIR*.

[36] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling user interest and conformity for recommendation with causal embedding. In *WWW*.