LightGCN: Evaluated and Enhanced

Milena Kapralova[1] **Luca Pantea**[1] **Andrei Blahovici**[1]
[1] University of Amsterdam
{milena.kapralova, luca.pantea, andrei.blahovici}@student.uva.nl

ABSTRACT: This paper analyses LightGCN [3] in the context of graph recommendation algorithms. Despite the initial design of Graph Convolutional Networks for graph classification, the non-linear operations are not always essential. LightGCN enables linear propagation of embeddings, enhancing performance. We reproduce the original findings, assess LightGCN's robustness on diverse datasets and metrics, and explore Graph Diffusion as an augmentation of signal propagation in LightGCN[1].

1 Introduction

The Graph Convolution Network (GCN) has emerged as a powerful method in graph collaborative filtering recommendations. However, GCNs include many non-linear operations essential for collaborative filtering tasks. He et al. [3] proposed LightGCN, a model that removes non-linear activation functions and linear transformations from the Neural Graph Collaborative Filtering (NGCF) [5] architecture, linearly propagating user and item embeddings on the user-item bipartite graph. He et al. [3] empirically validated the new architectural choice by comparing it with NGCF and running an extensive ablation study over the model's hyperparameter space. This work has both reproducibility and extension goals. We verify that simpler GCNs achieve top performance through smoother embeddings. We further test LightGCN on five datasets, considering diversity and fairness, and enhance results using Graph Diffusion.

2 Background

The LightGCN model is based on the NGCF architecture, with the non-linear activation functions and linear projections removed. Thus, the aggregation function in the graph convolutions of LightGCN is:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} \mathbf{e}_i^{(k)} \qquad \mathbf{e}_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} \mathbf{e}_u^{(k)} \tag{1}$$

where $e_u^{(k)}$ and $e_i^{(k)}$ are embedding of user u at layer k and the embedding of item i at layer k, respectively. LightGCN aggregates user and item embeddings from neighboursa at each layer, with only input embeddings being trainable. The final embeddings combine all layers' outputs. Predictions are based on the largest inner product between user and item embeddings:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)} \qquad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)} \qquad \hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i \tag{2}$$

where α_k is the weight associated with the k-th layer embeddings. In He et al. [3], $\forall k : \alpha_k = \frac{1}{K+1}$.

3 Method

Approximate Personalised Propagation of Neural Predictions (APPNP) is a GCN variant and an instance of graph diffusion convolution inspired by Personalised Page Rank (PPR). It derives from topic-sensitive PageRank approximated with power iteration [2], propagating the final embeddings:

$$\mathbf{Z}^{(0)} = \mathbf{E}^{(K)}, \quad \mathbf{Z}^{(k+1)} = \alpha \mathbf{Z}^{(0)} + (1-\alpha)\hat{\hat{\mathbf{A}}}\mathbf{Z}^{(k)}, \quad \mathbf{Z}^{(K)} = \mathrm{s}\left(\alpha \mathbf{Z}^{(0)} + (1-\alpha)\hat{\hat{\mathbf{A}}}\mathbf{Z}^{(K-1)}\right) \tag{3}$$

The final embedding matrix $\mathbf{E}^{(K)}$ serves as the starting vector and teleport set with K denoting power iteration steps. This method maintains graph sparsity without needing extra training parameters and prevents oversmoothing due to its teleport design. The teleport probability, α, adjusts the neighbourhood size. While Gasteiger et al. [2] found optimal alphas between [.05, .2], our grid search on the CiteULike dataset identified the best α as .1. For other datasets, we begin with this alpha, adjusting slightly to optimize test set performance.

[1]Anonymised code repository: `https://anonymous.4open.science/r/LightGCN-1B84/README.md`

4 Experimental setup

Datasets. He et al. [3] use the datasets Gowalla, Yelp2018, and Amazon-Book. To see how LightGCN performs in different domains and dataset sizes, we use five additional datasets (Table 1).

Table 1: Statistics of the experimented datasets

Dataset	**Gowalla**	**Yelp2018**	**A-Book**	**CiteULike***	**A-Movies***	**A-Electro***	**A-CDs***	**A-Beauty***
User #	29,858	31,668	52,643	3,276	44,438	1,434	43,168	7,068
Item #	40,981	38,048	91,599	16,807	25,046	1,522	35,647	3,569
Inter. #	1,027,370	1,561,406	2,984,108	178,062	1,070,860	35,931	777,426	79,506
Density	0.00084	0.00130	0.00062	0.00323	0.00096	0.01645	0.00051	0.00315

Note: "A" indicates Amazon. *Additional datasets not implemented in He et al.[3]

Metrics. We assess NDCG, recall, precision, diversity and fairness. We calculate *diversity* by the ILD [1]. Users who make repeat purchases receive better recommendations than those exploring new items or with few interactions [4]. To calculate *fairness*, we bin users by the number of interacted items. Similar recommendation performance across bins implies more fairness.

Implementation details. We follow He et al. [3] in training the models for 1000 epochs (for diffusion experiments, we use 600 epochs), using Adam optimizer ($lr = .001$), the BPR loss and $\lambda = .0001$.

5 Replication study

Claim 1: LightGCN Surpasses SOTA GCN Recommenders - *Correct*

Table 2 indicates the replicated LightGCN underperforms compared to the original, with Gowalla showing improvement with more layers, while Yelp2018 and Amazon-Book have varied outcomes. Table 3 examines the 3-layer LightGCN with different normalizations. Consistent with the original paper, square root normalization on both sides is best, while altering one side diminishes results. Using L_1 normalization symmetrically reduces performance for Gowalla and Yelp2018 but enhances it for Amazon-Book. Despite variations, the replicated values surpass NGCF.

Table 2: Replicated (**Rep.**) LightGCN. The relative difference to the original results is in %.

Dataset		**Gowalla**		**Yelp2018**		**Amazon-Book**	
Layer #	**Method**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**
1 Layer	LightGCN (**Rep.**)	0.1666 (-5.08%)	0.1399 (-6.34%)	0.0557 (-13.87%)	0.0452 (-12.14%)	0.0383 (-0.26%)	0.0293 (-1.64%)
2 Layers	LightGCN (**Rep.**)	0.1774 (-0.17%)	0.1512 (-0.80%)	0.0613 (-1.44%)	0.0503 (-0.20%)	0.0409 (-0.97%)	0.0316 (+0.32%)
3 Layers	LightGCN (**Rep.**)	0.1807 (-0.88%)	0.1537 (-1.07%)	0.0643 (+0.63%)	0.0528 (+0.62%)	0.0416 (+1.46%)	0.0322 (+2.13%)
4 Layers	LightGCN (**Rep.**)	0.1806 (-1.31%)	0.1528 (-1.20%)	0.0653 (+0.62%)	0.0537 (+1.28%)	0.0414 (+1.79%)	0.0319 (+2.12%)

Claim 2: Evaluating Embedding Smoothing in LightGCN's Performance Gain - *Unreproducible*

The step needs $O(M + N)^2$ operations (M - users, N - items); parallelization isn't viable due to set intersection. With an estimated 1507-hour runtime, we didn't replicate this study aspect.

Table 3: 3-layer LightGCN with different normalization schemes in graph convolution.

Method	**Gowalla**		**Yelp2018**		**Amazon-Book**		**Replication Rel. Diff. Interval**
	Recall@20	**NDCG@20**	**Recall@20**	**NDCG@20**	**Recall@20**	**NDCG@20**	
LightGCN-L_1-L	0.1700	0.1392	0.0623	0.0507	0.0424	0.0324	$[-1.55\%, 1.25\%]$
LightGCN-L_1-R	0.1517	0.1223	0.0553	0.0445	0.0321	0.0246	$[-9.28\%, -3.87\%]$
LightGCN-L_1	0.1564	0.1300	0.0570	0.0462	0.0360	0.0276	$[-1.64\%, 0.36\%]$
LightGCN-L	0.1767	0.1501	0.0642	0.0524	0.04102	0.03216	$[-0.07\%, 14.01\%]$
LightGCN-R	0.173	0.1436	0.06189	0.0499	0.03695	0.02817	$[-0.46\%, 24.44\%]$
LightGCN	0.1807	0.1537	0.0643	0.0528	0.0416	0.0322	$[-1.25\%, 2.22\%]$

Notation: $-L$ and $-R$ the side norm used, $-L_1$ means the L_1 norm is used instead of the L_2 norm.

6 Extended work

Additional datasets In examining various datasets, the 4-layer configuration of the LightGCN variant consistently outperformed the other configurations. For the **CiteULike** dataset, the 4-layer setup improved performance by roughly 44-50% over the 1-layer configuration. In the **Amazon-Movies** dataset, the enhancement was around 30%. **Amazon-Electronics** showcased an increase of 31.8-44.4%, while **Amazon-CDs** and **Amazon-Beauty** revealed improvements of 11.5-16.3%. Notably, datasets with reduced density, e.g. Amazon-Electronics, tend to yield weaker performance outcomes compared to denser datasets.

Additional metrics. In Figure 1, we compare LightGCN and APPNP on NDCG@20 and **diversity**. APPNP outperforms in NDCG@20 but not in diversity, stressing the need for comprehensive evaluation. Limited propagation iterations in APPNP may hinder diverse information diffusion. For **fairness** (Figure 1), both models favour users with more interaction data, with APPNP being less biased.

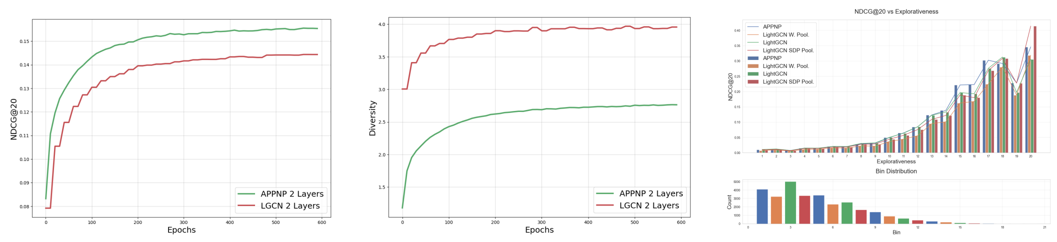

Figure 1: NDCG, diversity and fairness with APPNP and LightGCN, on the Gowalla dataset.

Graph diffusion. Using propagation on LightGCN embeddings enhances early training stability (Figure 2) without significantly improving maximum performance compared to LightGCN alone. Notably, LightGCN's performance declines during training on the Amazon-Electronics dataset, in contrast to APPNP, which maintains stability. Similar trends are observed across other datasets.

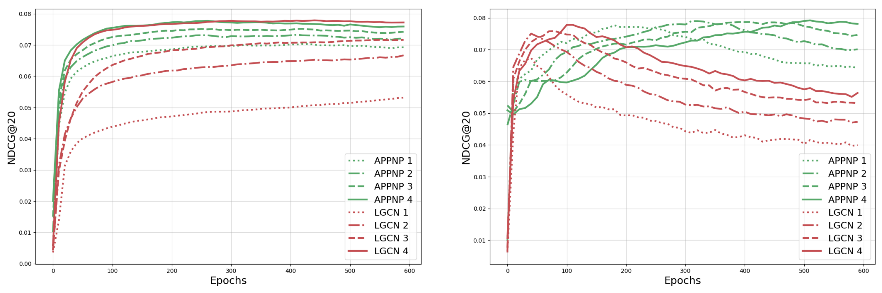

Figure 2: Model performance with (APPNP) and without (LGCN) propagation of embeddings, on diverse datasets. For both APPNP models, number of propagation steps=10. α=.1.

7 Discussion and Conclusion

This study mostly replicated He et al.'s [3] findings, even though time constraints limited full replication. LightGCN excels with high-interaction users but is generally outperformed by diffusion-augmented embeddings. However, combining LightGCN with APPNP can prolong training and requires parameter tuning. Future work can consider approaches like Dual LightGCN for graded interaction predictions, integrating graph attention networks for complex relationship capture and understanding how dataset properties such as sparsity and graph size impact diffusion parameters.

References

[1] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable multi-interest framework for recommendation, 2020.

[2] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank, 2022.

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

[4] Ming Li, Sami Jullien, Mozhdeh Ariannezhad, and Maarten de Rijke. A next basket recommendation reality check. *ACM Trans. Inf. Syst.*, 41(4), apr 2023.

[5] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. *CoRR*, abs/1905.08108, 2019.