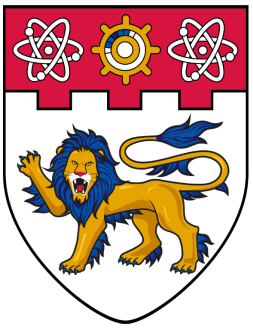


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

A Comparative Analysis of Graph Neural Networks for Recommender Systems

PROJECT 1

AY 2023/24 SEMESTER-1
CZ4032 - Data Analytics and Mining

Delivered by—
Pathak Siddhant (U2023715K)
Chen Wei May (U2020687E)
Mahtolia Ronan (U2023144J)

Submitted to—
Prof Lin Guosheng
School of Computer Science and Engineering, NTU

Table of Contents

1. Abstract

Entertainment platforms have been implementing features to recommend new items to their users based on their previous interactions with similar items. However, with the growing scale of the users and the items on the web, it can be difficult to implement an efficient recommender system on a large scale. Therefore, the aim of this paper is to investigate the different models and architectures used to implement recommender systems and compare them against each other. We experiment on NGCF, LightGCN and UltraGCN. They are compared based on Recall@20 and NDCG@20. Furthermore, an ablation study is carried out on UltraGCN. All experiments have been conducted on the Yelp18 and Amazon-Books datasets. Our experiments indicate that UltraGCN performs the best and it can effectively learn on item-item and item-user graphs to provide good recommendations.

2. Introduction

2.1 Recommender Systems

A recommender system can be viewed as a search ranking system, where the input query is a set of user and contextual information, and the output is a ranked list of items. Given a query, the recommendation task is to find the relevant items in a database and then rank the items based on certain objectives, such as clicks or purchases [1].

With the rise of social media, entertainment, and online shopping platforms such as YouTube, Instagram, and Amazon, recommender systems have become an integral part of such platforms to increase user activity and retention. Good recommender systems can provide a significantly better user experience which is why all these platforms are vying for more efficient and accurate recommender systems.

Recommender systems have been popularly implemented with deep learning techniques such as Graph Convolutional Networks. The core idea behind GCNs is to iteratively aggregate feature information from local graph neighbourhoods using neural networks. Here a single “convolution” operation transforms and aggregates feature information from a node’s one-hop graph neighbourhood, and by stacking multiple such convolutions, information can be propagated across far reaches of a graph [2] [3] [4].

There is a rather big challenge in implementing recommender systems on the web, i.e., scalability. It can prove difficult to implement a recommender system for millions of users

and billions of items on the web. In the past, state-of-the-art models have shown great improvement in performance on recommender system benchmarks.

2.2 Problem Statement

As mentioned above, there is a challenge in implementing recommender systems on a large scale. The number of users and items is ever-expanding, therefore there is a need to use an efficient and accurate model. In this experiment, we will explore the difference in performance: Recall@20 and NDCG@20, between baseline models used to implement recommender systems in the past and our primary algorithm. Additionally, an ablation study is performed to investigate the significance of the components of the model on the results it produces. This experiment aims to determine which model performs better than others out of the ones we have chosen. The ablation study aims to determine which parts are essential to the model along with the assumptions made for different hyper-parameters using a rigorous hyperparameter analysis

3. Methodology

We implement the algorithm introduced as Ultra-GCN [5]. It questions the necessity of an explicit message-passing mechanism commonly used in Collaborative Filtering (CF) tasks.

3.1 Ultra-GCN

The algorithm lays importance on two kinds of learnings from the different nature of the graphs:

1. Learning on User-Item Graphs: Representations of the last two layers of the network remain unchanged since the vector generated from the neighbourhood aggregation is equivalent to the node representation itself. The aim is to approximate such a convergence state directly, in place of message passing. Generally, Collaborative Filtering models recommend items through the utilisation of either pairwise Bayesian Personalized Ranking (BPR) loss or pointwise Binary Cross-Entropy (BCE) loss for optimization [6]. In our approach, we conceptualise CF as a problem of predicting links within a graph in the context of machine learning.

2. Learning on Item-Item Graphs: Traditional methods overlook the importance of item-item graphs and solely focus on user-item mapping. GCN-based algorithms lately are focusing on such relations and implicitly learning them using the same message-passing flow used for user-item relations. UltraGCN does not depend on such methods for the interpretation of these links. This allows manual adjustment of the relative importance of different relationships.

The UltraGCN algorithm constructs a weighted item-item co-occurrence graph matrix by establishing connections between items that exhibit co-occurrences. To uphold training efficiency and preserve the sparsity of item connections, the algorithm selects the top K items based on their edge weights. This selection is grounded on the assumption that the edge weight between two items intuitively represents a robust measure of the proximity or similarity between them.

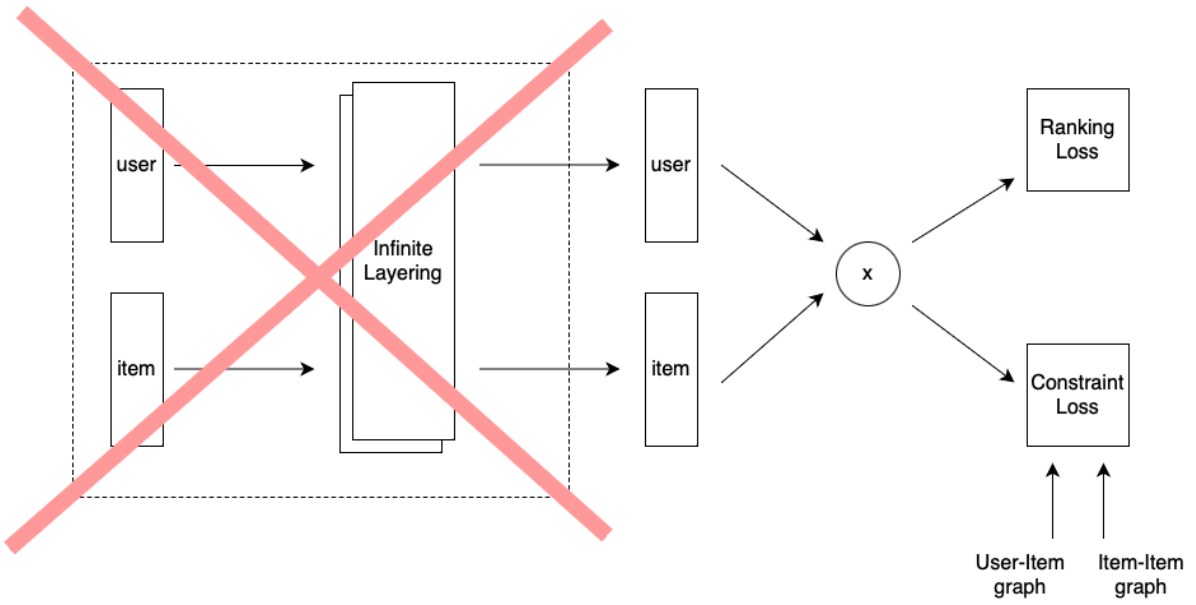


Figure 1. Model Architecture for UltraGCN

Generally, models for CF-based tasks make recommendations by optimizing Bayesian Personalised Loss (BPR). In this study, it is considered as a link prediction problem under the graph domain, thus BCE loss is used for optimization and defined further as L_0. It takes into account negative and positive samples. Negative sampling of links is done randomly [7].

L_C is the constraint loss function explicitly defined to approximate the convergence state of limit of infinite-layer message passing to understand relations in the user-item bipartite graphs, also neglecting any smoothing issues via incorporating negative sampling.

L_I is the constraint loss function explicitly defined to approximate the convergence state of limit of infinite-layer message passing to understand relations in the user-item bipartite graphs, also neglecting any smoothing issues via incorporating negative sampling.

Regularisation penalties are applied for wrongly classified pairs. Based on the two different types of graphs defined in the Section 3.1, we therefore define a multi-level loss function to be optimized:

$$L = L_O + \lambda L_C + \gamma L_I$$

Here, λ and γ are hyper-parameters to reflect the importance of user-item and item-item relationships respectively. For a more detailed explanation and derivation of the loss function expressions, refer to the original paper.

3.1.2 Experiments

We aim to answer two important research questions:
RQ1 How is UltraGCN effective with respect to the conventional CF algorithms?
RQ2 How are the assumptions made for the rationality, and simplicity of UltraGCN justified?

To answer these questions, we perform an experiment to evaluate and benchmark the UltraGCN algorithm against conventional models as baselines. Ablation studies are conducted later for demonstration of effectiveness and justification of the simplicity of our approach.

3.1.3 Experimental Setup

Datasets: For the scope of this study, we make use of the three publicly available datasets - Amazon-Books, Yelp2018 [8], and Movielens-1M [9]. These are used industry-wide and thus allow us to easily benchmark the performance of our model against currently existing State-Of-The-Art architectures.

The Amazon-Books dataset contains a vast collection of product information, customer reviews, and ratings for an extensive range of books available on the Amazon platform.

The Yelp2018 dataset encompasses a comprehensive repository of user-generated reviews, business information, and user demographics, serving as a valuable resource for studying consumer behaviour and business trends.

The Movielens-1M dataset is a classic collection of movie ratings and user data, featuring one million ratings from thousands of users, making it a foundational resource for collaborative filtering and recommendation system research.

Dataset Name	**Number of Users**	**Number of Items**	**Number of Interactions**
Amazon-Books	52643	91599	2984108
Yelp2018	31668	38048	1561406
Movielens-1M	6022	3043	995154

Table 1. Statistics of Dataset

Evaluation Metrics: Following the foundational research paper, we decided to use similar metrics as used by them to ensure reproducibility. We conform to Recall@20 and NDCG@20 due to their increasing popularity in graph-based recommendation algorithms [10] [11].

Parameter Name	**Value**
Learning Rate	Fine tuned from the set {1e-2, 1e-3, 1e-4}
Batch Size	1024
K (size of neighbour set)	10
Regularisation	L_2
Embedding Size	64
λ (Loss for User-Item relations)	Fine-tuned from the set {0, 1e-1, 1e-2, 1e-3, 1e-4}
γ (Loss for Item-Item relations)	Fine-tuned from the set {0, 1e-1, 1e-2, 1e-3, 1e-4}
β (Momentum)	Fine-tuned from a random sample of log-uniform distribution

Table 2. Parameter settings

Parameter Settings: Embedding initialization is done from pure sampling from Gaussian Distribution. Refer to Table 2 for the values chosen for experimentation. This section details the selection of values for different hyper-parameters as well as experimentation results for fine-tuning. We use PyTorch and PyTorch-Geometric [12] to replicate the algorithm. For other baseline models, we represent their results as is, from their original authors. We were unable to reproduce those results due to a lack of computational power and resources.

3.2 Baselines

3.2.1 Neural Graph Collaborative Filtering

Neural Graph Collaborative Filtering (NGCF) uses a user-item bipartite graph structure by propagating embeddings on it to integrate user-item interactions into the embedding process [8]. This enables expressive modelling of high-order connectivity in the user-item graph because of the explicit manner by which collaborative signals are injected into the embedding process (embedding function). The architecture comprises 3 components:

1. An embedding layer for initialization of user embeddings and item embeddings respectively,
2. Multiple embedding propagation layers that refine the user and item embeddings by injective high-order connectivity relations, and
3. A prediction layer that aggregates (concatenates) the embeddings refined from different propagation layers and outputs an affinity score for the user-item pair as the final prediction.

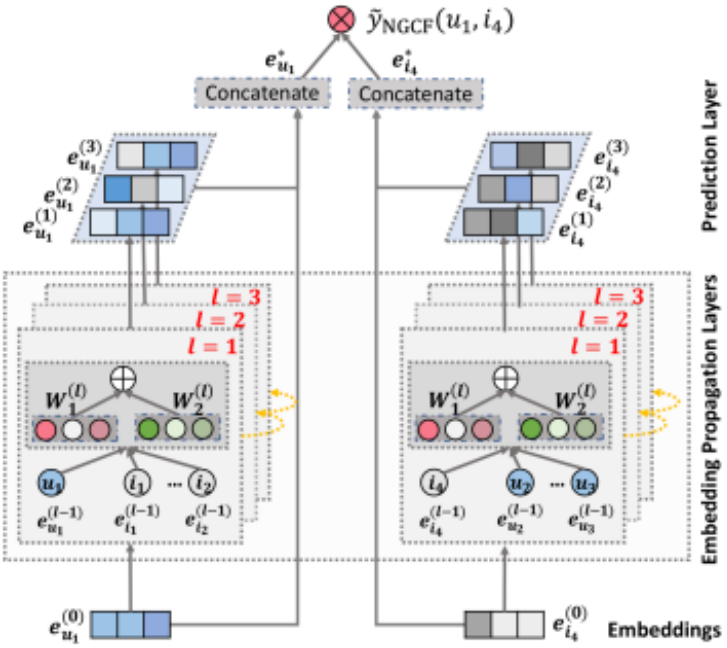


Figure 2. Model Architecture for Neural Graph Collaborative Filtering (NGCF)

3.2.2 Light-GCN

LightGCN hypothesised that neighbourhood aggregation is the most essential component in GCN for CF and that other designs common in GCNs such as feature transformation and nonlinear activation, like NGCF, are extraneous, increasing model training difficulty and degrades recommendation performance [13]. LightGCN aims to make the GCN design more concise and appropriate for recommendation to produce time savings in recommendation model training and prediction. It learns user and item embeddings through linear propagation on the user-item interaction graph for refining them. That is, only the normalised sum of neighbour embeddings is performed towards the next layer and other redundant operations such as feature transformation, self-connection and nonlinear activation are all removed. This is how the GCN is simplified. The weighted sum of the embeddings learned at all layers is used as the final combined embedding for prediction (in the Layer Combination). The rationality behind the simple design of LightGCN is demonstrated through the layer combination where LightGCN subsumes the effect of self-connection, thus removing the need for it to add self-connection in the adjacency matrix.

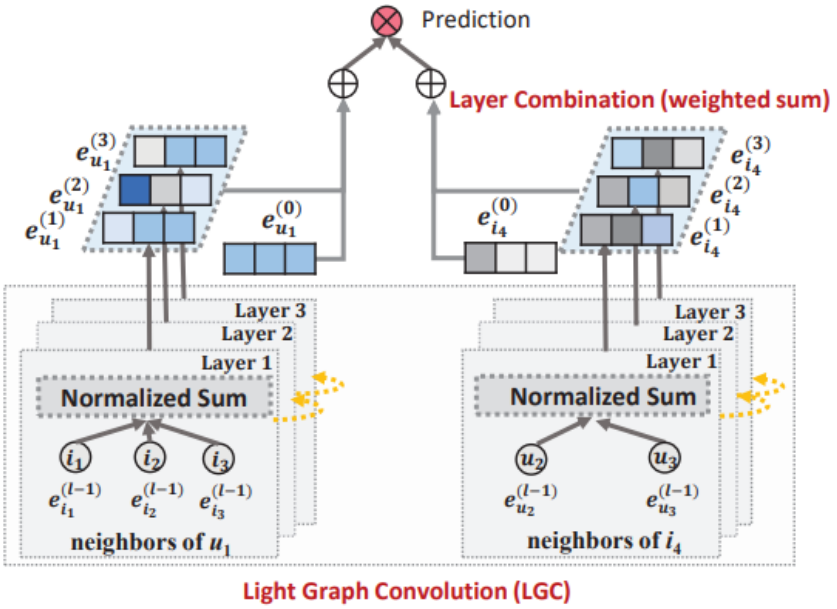


Figure 3. Model Architecture for LightGCN

4. Results and Discussion

4.1 Results

The Table (shown below) shows the performance of recommendation by the methods described in Section 2. The highest measures are highlighted in red and the second-highest measures are underlined for reference, with improvements in bold.

UltraGCN refers to the model trained using the base parameters provided in [5]. As per the Table 3 below, UltraGCN performed better at the recommendation tasks for all three datasets on both the evaluation metrics on the test set.

The Amazon-Books and Yelp2018 datasets are sparse datasets (with densities of 0.062% and 0.130% respectively). Our simplified UltraGCN algorithm outranks the existing NGCF and LightGCN by 71% and 4.3% on average for these two datasets. The Movielens-1M dataset is a relatively denser dataset (density 5.431%), thus providing our model with more positive user-item relations as well as item-item relations. Our GCN model outperforms the other two but not by a significant margin, suggesting the robustness of older models only on dense datasets. This demonstrates the ability of our GCN model to adapt to both dense and sparse datasets. Refer to Appendix A for detailed training graphs.

Model	**Amazon-Books**		**Yelp2018**		**Movielens-1M**	
	Recall@20	*NDCG@20*	*Recall@20*	*NDCG@20*	*Recall@20*	*NDCG@20*
NGCF [8]	0.0344	0.0263	0.0579	0.0477	0.2513	0.2511
LightGCN [13]	0.0411	0.0315	0.0649	0.0530	0.2576	0.2427
UltraGCN	0.0682	0.0557	0.0674	0.0555	0.2783	0.2641
Improvement (%)	**65.94**	**76.83**	**3.85**	**4.72**	**8.04**	**8.82**

Table 3. Benchmark UltraGCN against baselines

4.2 Ablation Study

Ablation studies on Amazon-Book and Yelp2018 datasets were performed to show that the design of UltraGCN is effective in being able to flexibly and separately learn the user-item relationships and item-item relationships for improved recommendation performance. The following variants of UltraGCN were compared to show the effectiveness in the designs in UltraGCN:

1. $\gamma = 0, \lambda = 0$: Setting UltraGCN with gamma and lambda to be 0 reduces UltraGCN to simple Matrix Factorization [14] training with Binary Cross Entropy loss that does not leverage graph information and cannot capture higher-order collaborative signals.

2. $\gamma = 0$: Setting UltraGCN with $\gamma=0$ is identical to $\text{UltraGCN}_{\text{base}}$ which only learns on the user-item graph and lacks more effective learning for item-item relationships

3. $\lambda = 0$: Setting UltraGCN with $\lambda=0$ lacks the graph convolution ability for learning on the user-item graph to mine the collaborative information more deeply.

Results in Figure 4 below shows that UltraGCN($\lambda=0$) and UltraGCN($\gamma=0$) perform better than UltraGCN($\lambda = 0, \gamma = 0$). Thus this demonstrates that the designs of UltraGCN can effectively learn on both the item-item and user-item graph to improve recommendations. Furthermore, it is observed that $\text{UltraGCN}_{\text{base}}$ performs much better than the other three variants, which suggests that the proposed solution of UltraGCN to disassemble various relationships, eliminate uninformative aspects which may hinder model learning as well as carry out multi-task learning more clearly, effectively overcomes the limitations of other previous GCN-based CF models such as NGCF and LightGCN.

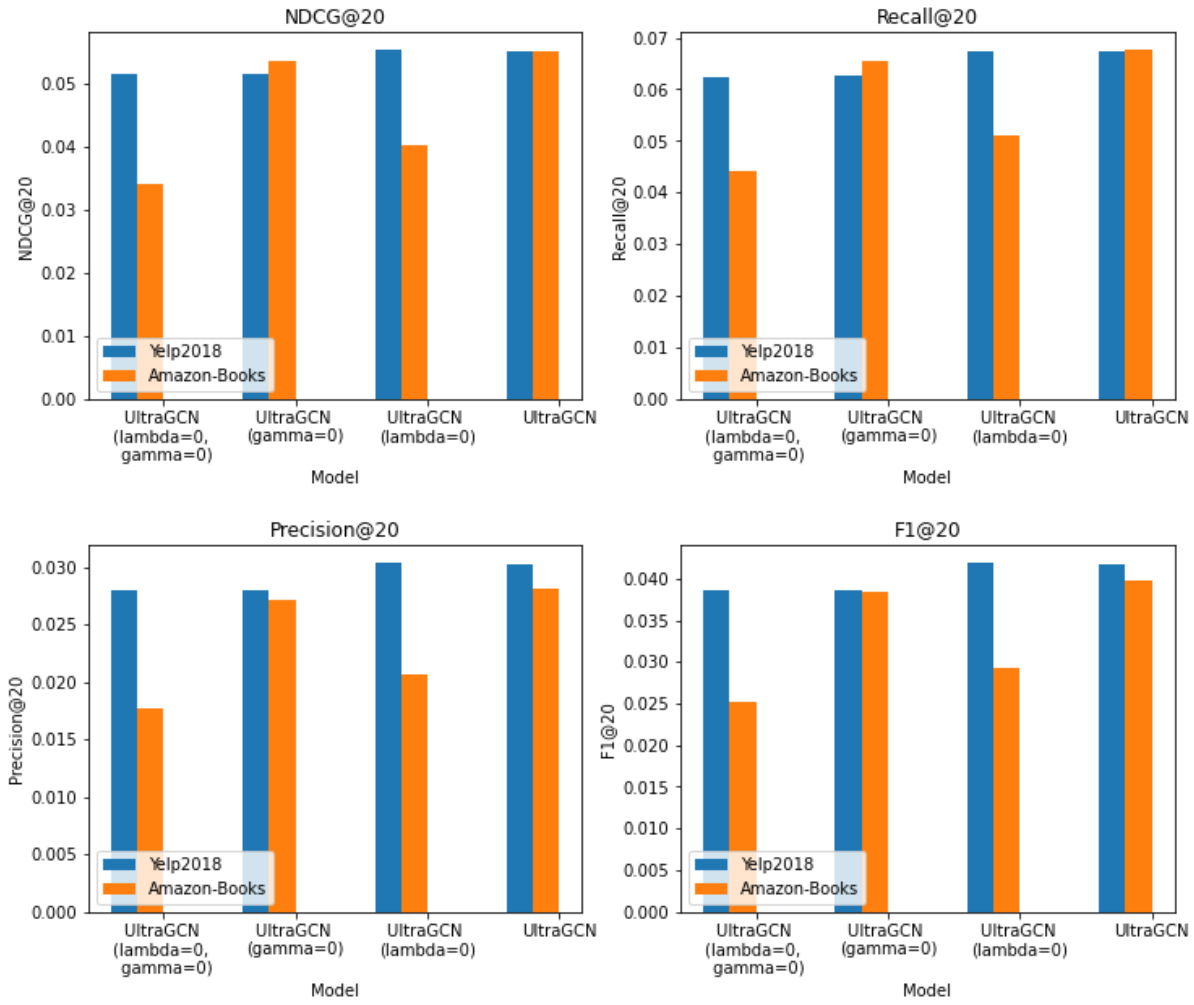


Figure 4. Ablation Study

4.3 Strength of UltraGCN over Baselines

In this section, we focus on discussing the merits and the research gaps laid out in the baseline algorithms and how UltraGCN tackles those problems without the use of convolution and infinite message-passing.

4.3.1 The research gap filled by NGCF

Traditional user and item embeddings lack collaborative signals, limiting their effectiveness in capturing collaborative filtering effects. NGCF addresses this by using a graph neural network to explicitly encode collaborative signals, making recommendation models more powerful.

4.3.2 Evaluation of NGCF Design

However, NGCF's designs can be complex and burdensome due to some operations inherited from GCN without clear justification for their usefulness in collaborative filtering tasks. In ablation studies conducted on NGCF [8], it was identified that common design features in GCNs such as feature transformation and nonlinear activation have no contribution to NGCF's effectiveness but significantly negatively impacts model training time and that the removal of such features leads to significant improvements in model prediction accuracy.

4.3.3 Comparison between LightGCN and NGCF

Compared to other GCNs for recommender systems such as NGCF, LightGCN has a simple, elegant linear, and neat model that is far easier to implement and train. In a study, it was found that under the same experimental setting, LightGCN exhibited a substantial improvement of about 16.0% relative improvement on average over NGCF, a state-of-the-art GCN-based recommender model. This is due to its removal or non-inclusion of the redundant design features (feature transformation and nonlinear activation) of operations or components of GCNs which are not useful for the CF task, which significantly improves model training time and prediction accuracy.

1. Simplifying model architecture: Specifically, while NGCF has a more computationally intensive and complex model architecture that involves graph convolution operations as well as user-item CF, LightGCN simplifies the architecture by focusing on only the CF aspect, thus removing the need for user-item feature embeddings and graph convolutional layers.

2. Ensures uniform information propagation: Additionally, the different layers of graph convolution in NGCF having different weights leads to uneven non-uniform propagation of information across layers, which may hinder convergence and increase the difficulty of hyperparameter tuning. However, LightGCN addresses this issue through the use of uniform propagation weights for all layers in order to ensure stable and consistent information flow through the network.

3. Improved parameter efficiency: Furthermore, NGCF's high number of parameters due to the multi-layer graph convolutional architecture may lead to overfitting and require significantly high training time. The simplified architecture of LightGCN with

uniform propagation weights reduces the number of parameters significantly so that the training can be more parameter-efficient.

4. Enhanced scalability: Moreover, the improved simplicity and parameter efficiency of the LightGCN model enables it to be more scalable than the NGCF. Where NGCF is computationally expensive, LightGCN is significantly less demanding of computational resources.

In summary, LightGCN addresses the problems in NGCF by simplifying the model architecture, ensuring uniform information propagation, improving parameter efficiency, and enhancing scalability. These improvements/modifications make LightGCN a promising recommendation model that retains the effectiveness of NGCF while being more efficient and easier to train.

4.3.4 Comparison between LightGCN and UltraGCN

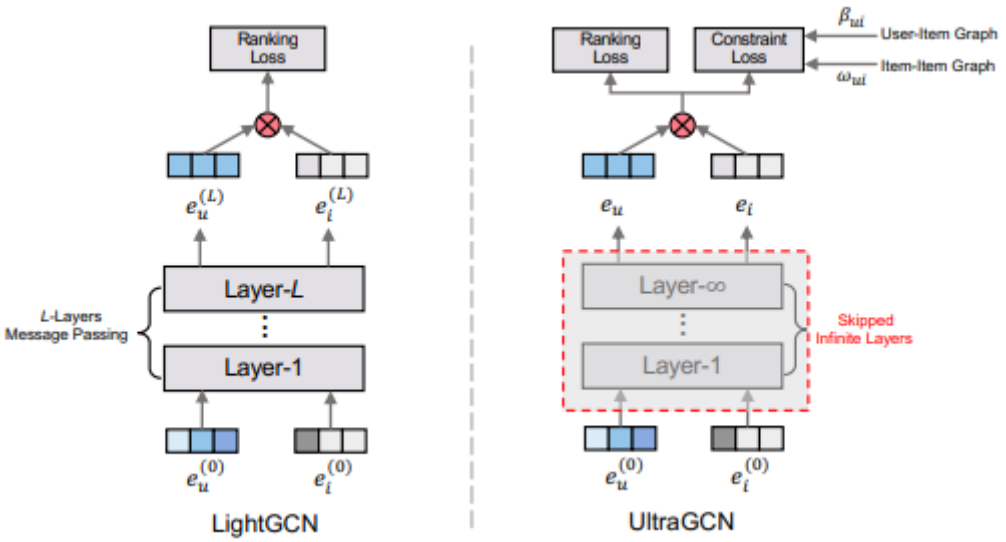


Figure 5. Illustrations of training of LightGCN (left) and UltraGCN (right).

Although the LightGCN as mentioned earlier model has already been simplified for training, the message-passing operations still dominate their training. Ultra-GCN aims to further enhance the efficiency of the recommender system training by bypassing the message-passing operations. Specifically, UltraGCN's direct approximation of the limit of infinite-layer graph convolutions using a constraint loss addresses three limitations of LightGCN.

First, the message-passing mechanism involves weights assigned to edges, which may not be appropriate for CF. Hence, by bypassing the message-passing operations, UltraGCN gets around this issue.

Second, the failure to capture the varying importance of the relationship pairs (user-user, item-item, and user-item) in LightGCN is addressed by UltraGCN because the loss-based design allows manual adjustment of the relative importance of the different types of relationships.

Thirdly, LightGCN has an over-smoothing issue which limits the usage of too many message-passing layers, which UltraGCN's loss-based design also helps to avoid by negative sampling. Experimental results indicate that UltraGCN outperforms the state-of-the-art GCN models, attains up to 76.6% improvement in NDCG@20, and also achieves more than 10x speedup over LightGCN.

With reference to Figures 1 and 3, the difference between the model architectures of LightGCN and UltraGCN is the absence of infinite message-passing layers in the latter.

4.4 Parameter Analysis

The model's accuracy is contingent upon the settings of these hyperparameters. Therefore, selecting the best configuration for these parameters would yield the highest level of accuracy, whereas a less-than-ideal choice could lead to less accurate results. Some of the key hyperparameters that should be adjusted include the weights of the two constraint losses (λ and γ), the learning rate, and the momentum of the Adam optimizer (β). The Yelp2018 dataset was used for experimentation and results are shown in a graphical manner.

4.4.1 Impact of User-Item Relations

This parameter is indicative of the relevance of user-item relations in terms of representation learning. We test with different λ from the set {0, 1e-1, 1e-2, 1e-3, 1e-4}. We see a lot of unseasonal variations for validation loss during the training for each λ. Losses for $\lambda = 0$ and $\lambda = 1e\text{-}1$ reach the global minimum earliest among the other λ, but they increase with more variation, indicative of the model not learning the representations well, and focusing solely on improving the NDCG@20. For $\lambda = 1e\text{-}3$, we observe that the NDCG@20 metric is very similar to others; however, the loss follows a more stable curve, with less variation, highlighting its capacity to capture the user-item relations with the right relevance parameter.

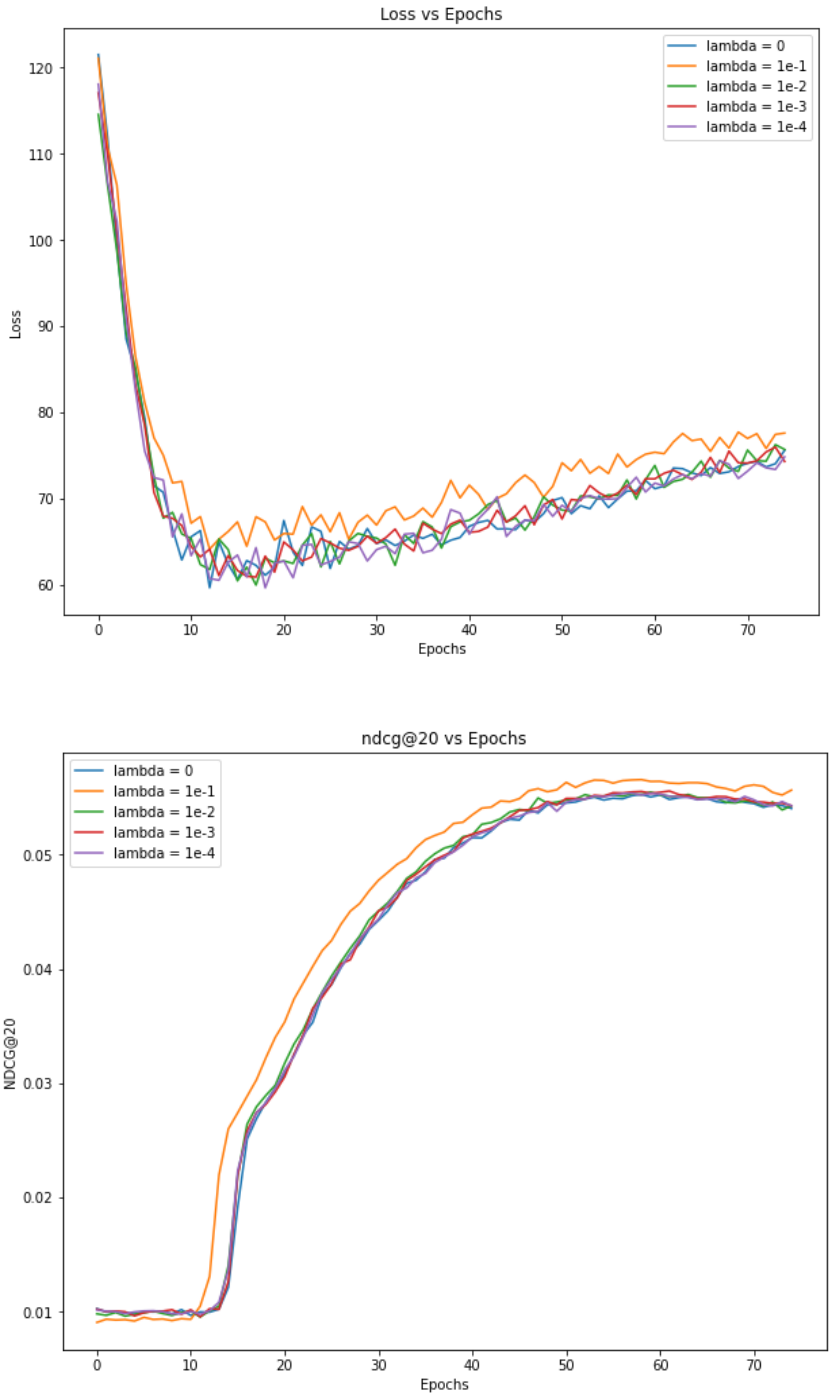


Figure 4. Loss and NCDG@20 curves for different lambda

4.4.2 Impact of Item-Item Relations

This parameter is indicative of the relevance of item-item relations in terms of representation learning. A similar approach is taken for tuning γ. The set for fine-tuning γ chosen is {0, 1e-1, 1e-2, 1e-3, 1e-4}. The number of neighbors K is set to 10 for all these experiments. We can see the impact and degree of variation in γ. For γ = {1e-1, 1e-2), we can see that running

losses over the training jump drastically on the scale with respect to other γ. This showcases the relatively low importance of item-item relations. Therefore, only γ = {1e-3, 1e-4} are the only relevant values of the datasets. However, after investigating the NDCG@20 for these two values, we find out that γ = 1e-4 produces the lowest running loss as well as an increasing trend for NDCG@20.

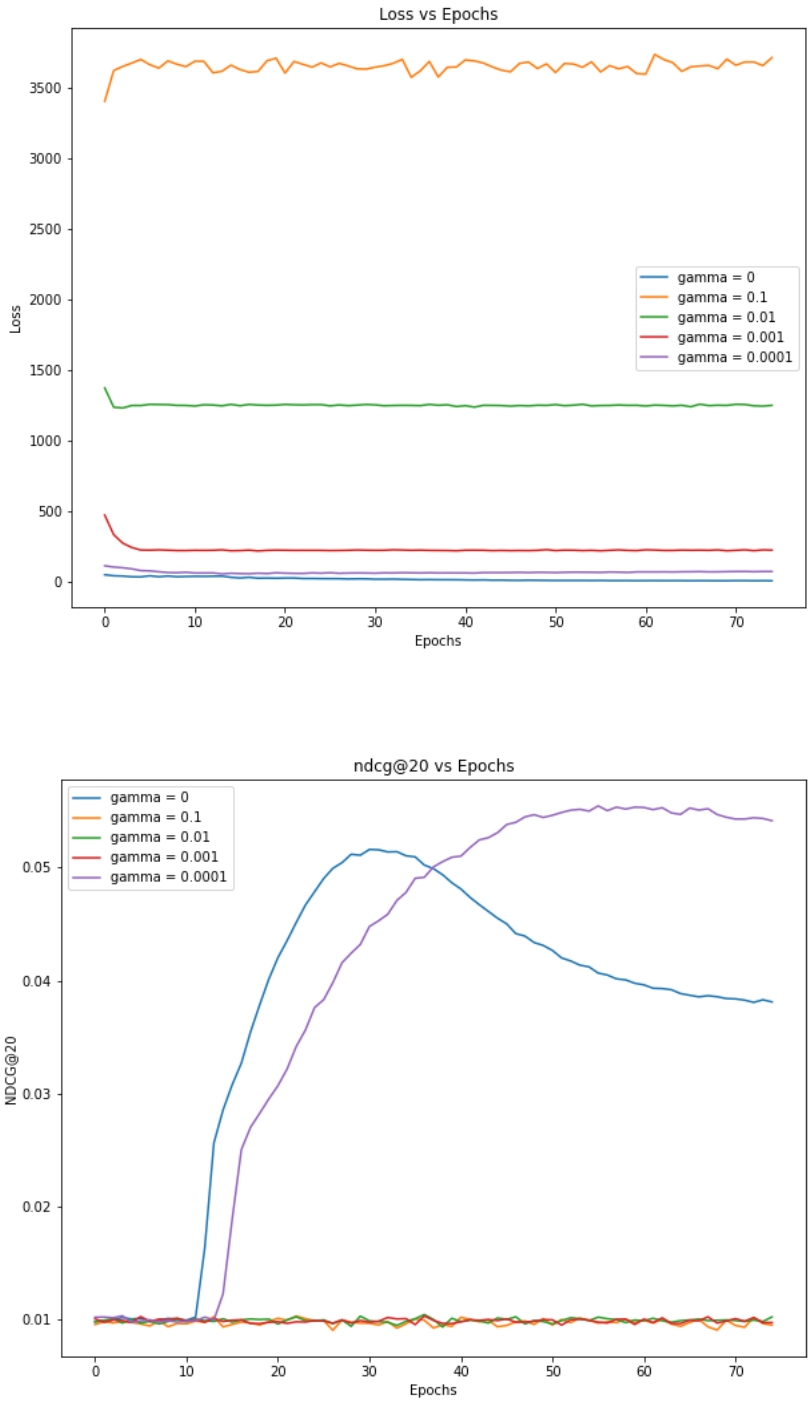


Figure 5. Loss and NCDG@20 curves for different gamma

4.4.3 Impact of Momentum

Momentum is responsible for helping the neural network overcome the shallow local minima in search of optimal loss settings, so that a more important global minima is found. It scales gradients and resists rapid variation in parameter values. Figure 6 below shows the effect of the momentum of Adam Optimizer on validation loss. Value of β was taken from a range of randomly sampled values from a log-normal distribution.

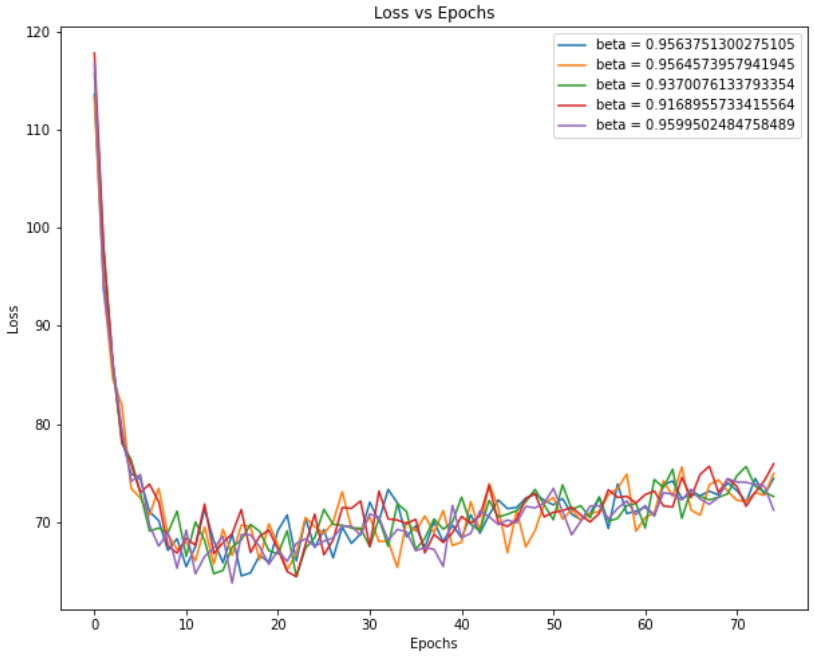


Figure 6. Loss and NCDG@20 curves for different beta (momentum)

4.4.4 Impact of Learning Rate

The learning rate plays a vital role in tweaking the model's parameters after each training cycle. When the learning rate is very low, the parameter adjustments are tiny, meaning it takes longer to reach the best results, which can be problematic when computational resources are scarce. We test with a different learning rate from the set {1e-2, 1e-3, 1e-4}. From Figure 7, we observe that 1e-4 is a relatively slower approach, as it takes time to reach asymptotic values, compared to other values.

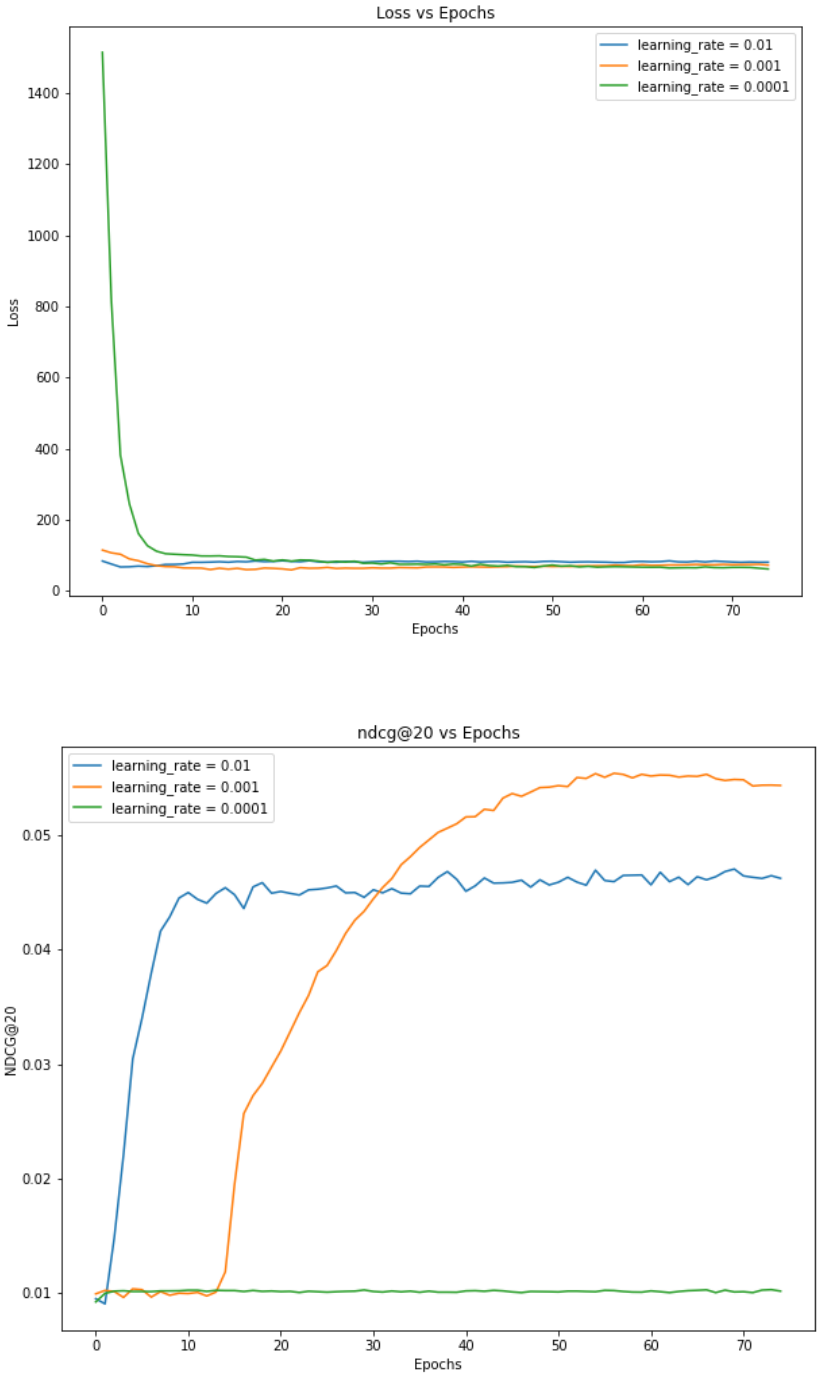


Figure 7. Loss and NCDG@20 curves for different learning rates

5. Conclusion

In our research, we introduce a highly streamlined version of Graph Convolutional Networks (GCN) called UltraGCN. UltraGCN bypasses the conventional message-passing step and instead approximates the outcome of an infinite number of message-passing layers directly. Our extensive experiments showcase remarkable advancements in accuracy and efficiency compared to state-of-the-art Collaborative Filtering (CF) models, highlighting the potential of UltraGCN as a powerful approach in recommendation systems.

The way that the discussed models further improve upon one another can be summarised by the following. NGCF produces state-of-the-art results but is computationally intensive partly due to the fact that it contains redundant components that hinder training and extend model training time. LightGCN addresses the problems in NGCF by simplifying the model architecture through the removal of redundant components. UltraGCN further improves upon the efficiency of LightGCN by bypassing the message-passing operations.

As a future avenue of research, we propose the exploration of neuron count optimization within the layers of Graph Neural Networks (GNNs) using PyTorch's Optuna Tuner [15]. This approach holds the promise of achieving superior model performance by striking the right balance between complexity and accuracy. By systematically fine-tuning the neuron configurations, we anticipate the discovery of optimal GNN architectures, ensuring improved predictive power while guarding against overfitting. Incorporating time intervals into GNNs represents a promising frontier for advancing recommender systems [16]–[18]. This future research direction aims to enhance the temporal representation learning aspect, enabling the model to capture the dynamic nature of user preferences and item interactions over time. By effectively integrating time as a key feature within GNN architectures, we anticipate a significant boost in recommendation accuracy.

6. References

[1] H.-T. Cheng *et al.*, “Wide & Deep Learning for Recommender Systems.” arXiv, Jun. 24, 2016. Accessed: Sep. 26, 2023. [Online]. Available: http://arxiv.org/abs/1606.07792

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph Convolutional Neural Networks for Web-Scale Recommender Systems,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018, pp. 974–983. doi: 10.1145/3219819.3219890.

[3] X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” *Adv. Artif. Intell.*, vol. 2009, pp. 1–19, Oct. 2009, doi: 10.1155/2009/421425.

[4] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, “Graph Neural Networks in Recommender Systems: A Survey,” *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, May 2023, doi: 10.1145/3535101.

[5] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, “UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 1253–1262. doi: 10.1145/3459637.3482291.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian Personalized Ranking from Implicit Feedback,” 2012, doi: 10.48550/ARXIV.1205.2618.

[7] R. Miao *et al.*, “Negative samples selecting strategy for graph contrastive learning,” *Inf. Sci.*, vol. 613, pp. 667–681, Oct. 2022, doi: 10.1016/j.ins.2022.09.024.

[8] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, “Neural Graph Collaborative Filtering,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Paris France: ACM, Jul. 2019, pp. 165–174. doi: 10.1145/3331184.3331267.

[9] F. M. Harper and J. A. Konstan, “The MovieLens Datasets: History and Context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016, doi: 10.1145/2827872.

[10] S. Muwanei, S. D. Ravana, W. L. Hoo, and D. Kunda, “Prediction of the high-cost normalised discounted cumulative gain (nDCG) measure in information retrieval evaluation.” Accessed: Sep. 26, 2023. [Online]. Available: https://informationr.net/ir/27-2/paper928.html

[11] O. Jeunen, I. Potapov, and A. Ustimenko, “On (Normalised) Discounted Cumulative Gain as an Offline Evaluation Metric for Top-n Recommendation.” arXiv, Jul. 27, 2023. Accessed: Sep. 26, 2023. [Online]. Available: http://arxiv.org/abs/2307.15053

[12] M. Fey and J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric,” 2019, doi: 10.48550/ARXIV.1903.02428.

[13] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, “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*, Virtual Event China: ACM, Jul. 2020, pp. 639–648. doi: 10.1145/3397271.3401063.

[14] B.-H. Kim, A. Yedla, and H. D. Pfister, "Message-Passing Inference on a Factor Graph for Collaborative Filtering." arXiv, Apr. 07, 2010. Accessed: Sep. 26, 2023. [Online]. Available: http://arxiv.org/abs/1004.1003

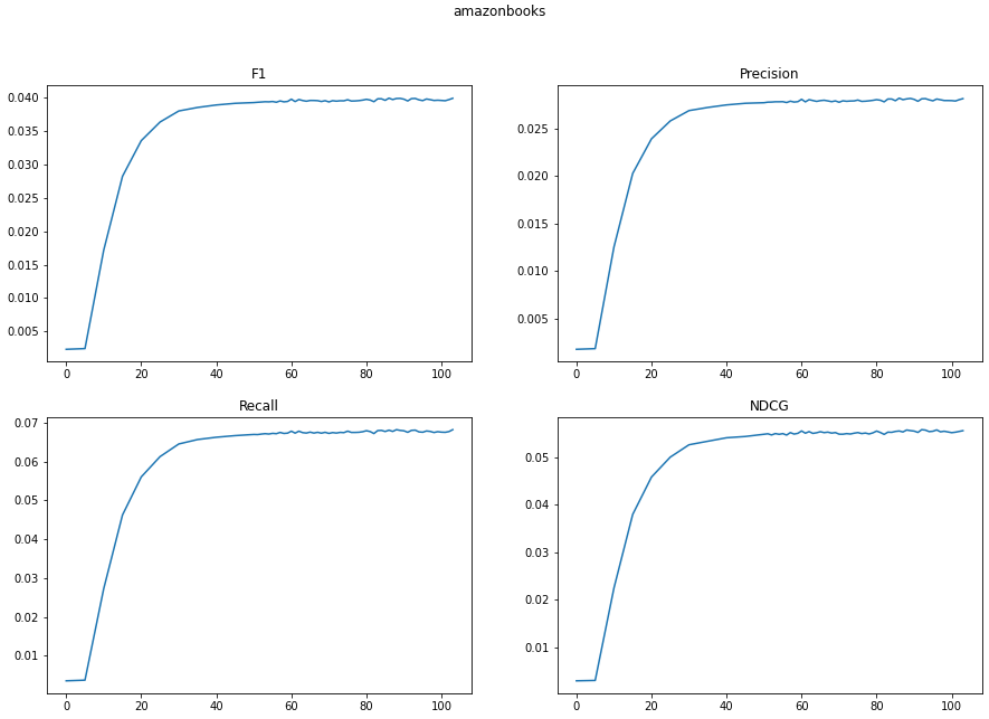
[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.

[16] Z. Fan, Z. Liu, J. Zhang, Y. Xiong, L. Zheng, and P. S. Yu, "Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer," 2021, doi: 10.48550/ARXIV.2108.06625.

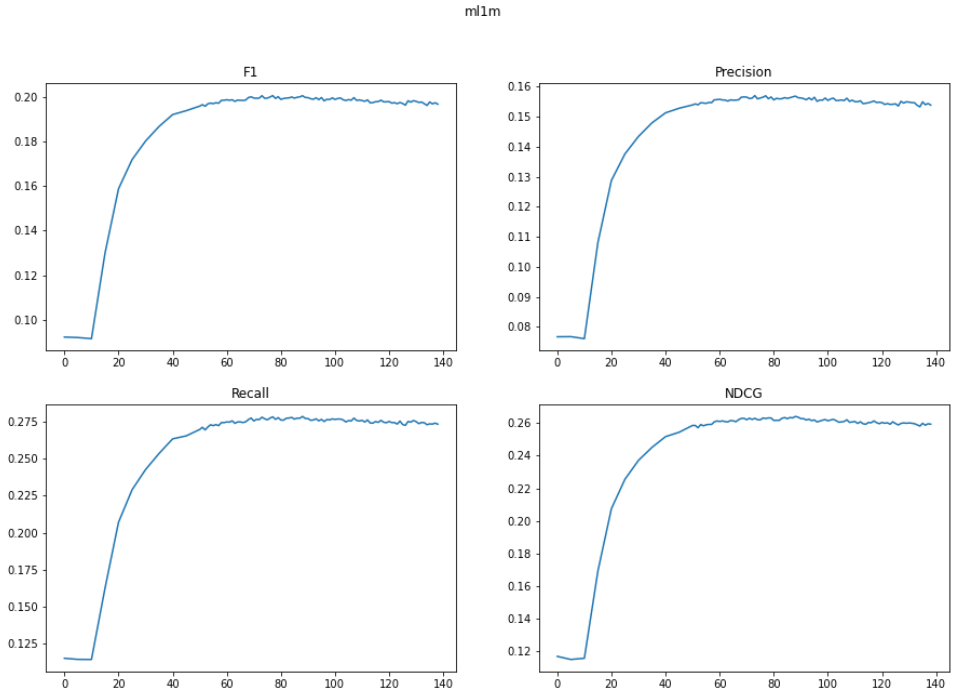
[17] J. Li, Y. Wang, and J. McAuley, "Time Interval Aware Self-Attention for Sequential Recommendation," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, Houston TX USA: ACM, Jan. 2020, pp. 322–330. doi: 10.1145/3336191.3371786.

[18] S. Kumar, X. Zhang, and J. Leskovec, "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 2019, pp. 1269–1278. doi: 10.1145/3292500.3330895.

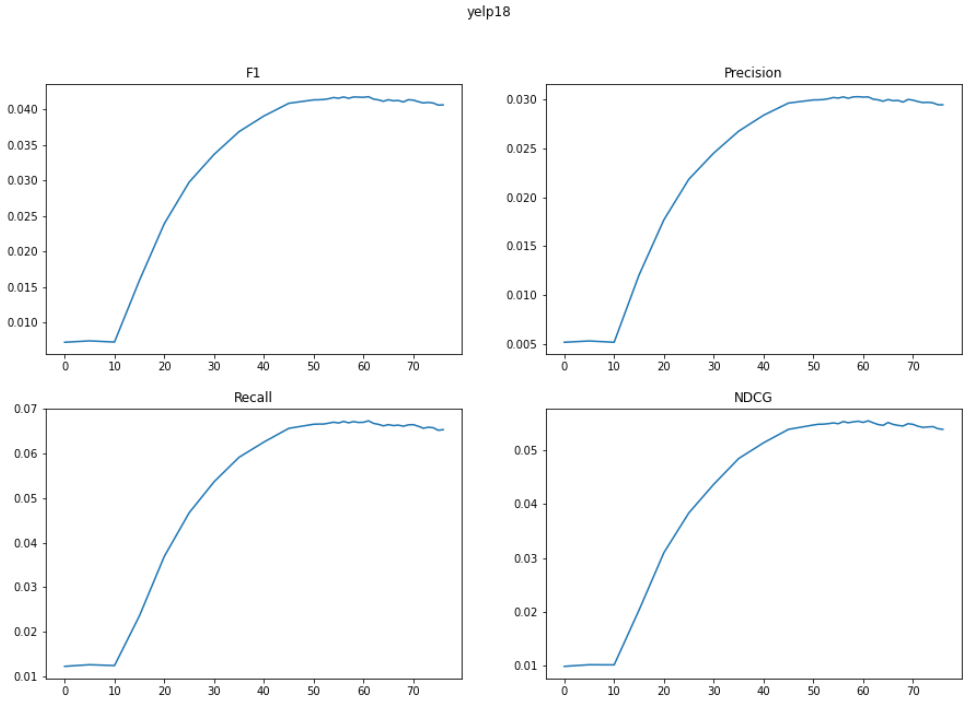
7. Appendix A: Training Graphs



(amazon-books dataset)



(Movielens-1M dataset)



(Yelp2018 dataset)

Figure 8. Training Metrics for all three datasets

We have provided the training metrics (F1 score, Precision, Recall, NDCG) for all three datasets.