Reproducibility Report for UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation

FABIAN HARTMANN 11824496, TU Wien, Austria
SEBASTIAN HEPP 01015083, TU Wien, Austria
YIFAN MAYR 01457320, TU Wien, Austria
MATTHIAS MOIK 11810738, TU Wien, Austria

We report on the reproducibility of the results of UltraGCN by Mao et al. [3], published during CIKM '21, November 2021, Virtual Event, QLD, Australia. UltraGCN is an ultra simplified graph convolutional network for recommendation. In this work, the reproducibility of the original paper is assessed in terms of its experimental design, data/code availability. Their metrics are also compared to the results obtained during our replication study. The authors of the original paper provided the necessary code, config-files, packages with version numbers and already split datasets, readily accessible in a GitHub repository. However, statistical significance test for performance between different settings was reported only with p-values; the exact type/name of the test and on what data it was used was not specified. An exact replication study was not possible due to computing power limit, therefore we had to use a subset of some datasets. In conclusion, the original work on UltraGCN could be reproduced to a reasonable extent. This paper is done as part of the course Experiment Design for Data Science at TU Wien. Our code and experiment configurations are publicly available in a GitHub repository[1].

CCS Concepts: • **Information systems** → **Recommender systems**; **Collaborative filtering**.

Additional Key Words and Phrases: Reproducibility

ACM Reference Format:
Fabian Hartmann 11824496, Sebastian Hepp 01015083, Yifan Mayr 01457320, and Matthias Moik 11810738. 2023. Reproducibility Report for UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. 1, 1 (January 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Reproducibility is core to the scientific method. According to the PRIMAD reproducibility model [1], we can modify different aspects of a data-driven experiment to gain insights into the various aspects of such an experiment. Here we probe the paper on UltraGCN by Mao et al. [3] by priming the Platform and Actors to evaluate whether the information provided is sufficient for independent verification.

[1] https://github.com/e11824496/ExpDesign_WS22.git

Authors' addresses: Fabian Hartmann 11824496, TU Wien, Karlsplatz 13, Vienna, Austria, e11824496@student.tuwien.ac.at; Sebastian Hepp 01015083, TU Wien, Karlsplatz 13, Vienna, Austria, e01015083@student.tuwien.ac.at; Yifan Mayr 01457320, TU Wien, Karlsplatz 13, Vienna, Austria, e1457320@student.tuwien.ac.at; Matthias Moik 11810738, TU Wien, Karlsplatz 13, Vienna, Austria, e11810738@student.tuwien.ac.at.

2 ULTRAGCN

Recommendation is the task of matching a user's preferences with available candidate items. Collaborative filtering is one approach where historic data is used to train vector representations of users and items, the similarity between these vector representations of users and items are then used to determine the top k best recommendations.

Graph Convolutional Networks (GCN) are popular for collaborative filtering, it's marked by its message passing system to aggregate neighborhood information. GCN models allow the discovery of higher-order connections between users and items. However, this message passing system requires substantial computation and is not efficient. LightGCN by He et al. [2] reduces GCN models - removing feature transformations and nonlinear activations. UltraGCN was developed to further simplify the process - removing message passing and instead approximating infinite-layer graph convolutions using a constraint loss.

3 EXPERIMENT DESIGN

To enable reproducibility, Mao et al. provided a git repository which includes all required source code for executing the performance analysis, the datasets and for each dataset a configuration-file and a python-notebook. The notebooks can execute the code and also contain the original output with information about loss and run-times for each training epoch and the final performance results. The repository also contains information about the CPU and RAM of the system which was used for the performance evaluation and the concrete versions of python and packages.

The original paper compared UltraGCN with other state of the art collaborative filtering methods with regards to its effectiveness and efficiency. Statistical significance testing for the efficiency metrics between UltraGCN and the baseline model DGCF appears to be done, since p-values are reported. However, it is not stated which and how the statistical test was applied.

The importance of individual parameters was assessed using an ablation study on the Amazon-books dataset. Mao et al. compared variants of UltraGCN with enabled or disabled learning on the item-item graph (gamma=0 to exclude) and/or graph convolution on the user-item graph (lambda=0 to exclude). The effects of model parameters (neighborhood sizes and weights of constraint loss) on performance were analyzed by sweeping a range of values.

In our study, we tried to reproduce the experiments of the effectiveness, ablation study, and impact of the model parameter K for UltraGCN. Efficiency analysis is not reasonable due to differences in hardware. Also the influence of the weights of the constraint losses was not reproduced due to time reasons. The same data processing and split strategies were applied for all models in the original paper and used in our replication study, but we had to use a subsampled dataset for most of the tests, because of limited RAM.

3.1 Datasets

Four publicly available datasets were used in the original paper for benchmarking: Amazon-Books, Yelp2018, Gowalla, and Movielens-1M. These datasets were commonly used to evaluate many GCN-based collaborative filtering methods. Interestingly, the original repository contained benchmarking results for two additional datasets, although they were not mentioned in the paper. The datasets are already preprocessed and split into train- and test-sets. The datasets were taken from another paper, where the preprocessing is explained. Therefore, reproducing preprocessing and sampling is possible by following the cited paper.

In our replication, we focused on the datasets Movielens-1M and Amazon-Books, where a subsample had to be used for the latter one due to limited computing power (Table 1). The subsampled set was created by randomly selecting 40%

Table 1. Statistics of datasets

Dataset	#Users	#Items
Movielens-1M	6,022	3,043
Amazon-Books	52,643	91,599
Amazon-Books-subsample	21,057	36,640

of users and 40% of items, where only users are included that are left with at least 5 items for training and 2 or more for testing.

The dataset Movielens was taken from another repository and already preprocessed by parsing integer-values from .dat-file, selecting specific columns (user-id, movie-id) and reshaping the data matrix. The training-test split was approximately 88:12.

3.2 Evaluation metrics

The effectiveness of the methods were assessed as Recall@20 and NDCG@20. Recall measures the proportion of relevant items found in the top 20 recommendations; NDCG measures the weighted relevance of the recommended items, normalized over all queries. The metrics were calculated based on the top 20 recommended items and averaged over all users for each metric.

The efficiency of UltraGCN compared to other methods was measured with the number of epochs and training time needed to achieve the model's best performance. Additionally, all model were trained for 75 epochs and the achieved effectiveness metrics were calculated.

3.3 Code

The authors provided the Python code in a public GitHub repository - all code is in the file "`main.py`". Also configuration files named "`dataset_name_config.ini`" are available for each dataset. The hardware(CPU and RAM) as well as the software environment requirements (versions of necessary packages) was specified. Since the datasets are relatively large and a lot of time is needed to run the tests, we ported the notebooks to Google Colab and ran the experiments there. About 12GB RAM and 107GB disk space were available for use.

The code required minor changes to run, as not all necessary tensors were transferred to the GPU, leading to errors within Pytorch. A slight modification within the config files was necessary to set the right GPU (from id 3 to 0). These changes were all minor but necessary to run the code in our setting.

The random seed that was used to sample the data into batches during training was not provided. This might be a source of some differences during replication for the results. In addition, each model was evaluated only once - based on this one training-split. With this approach, the resulting evaluation metrics are likely to be biased by this one split and may deviate from the results that would be obtained using cross-validation or repeated train-test-splits. Also, for the early stop and best epoch selection, no validation set is used and instead the evaluation is always done on the final test set. This also skews the results, as it tends to overfit on the test set.

Table 2. UltraGCN performance

Dataset	Recall@20		NDCG@20		#Epoch	
	Original	Reproduced	Original	Reproduced	Original	Reproduced
Movielens-1M	0.2787	0.277811	0.2642	0.26396	136	137
Movielens-1M + Validation set	-	0.25706	-	0.2329	-	137
Amazon-Books-subsample	0.0681	0.0574	0.0556	0.0373	86	85

4 RESULTS

4.1 Effectiveness study

For the dataset Movieliens-1M, the reproduced performance results are very close to the reported results with Recall@20 lower by 0.3%, NDCG@20 lower by 0.09% and one training epoch less. We assume the only reason the results are not exactly the same is because of the unfixed random seed and therefore different data samples in each training batch.

For the dataset Amazon-Books, the efficiency metrics are not completely comparable. The efficiency metrics achieved by UltraGCN from our study are lower than those from the original paper (Table 2). Our Recall@20 was 16% lower than the original, the NDCG@20 was 33% below the one reported in the paper. Our run needed 1 fewer epoch than the original result to reach the model best performance. Again, due to the reduced dataset, these results are not comparable.

4.2 Validation Set

One issue mentioned earlier, is the lack of a validation set to determine which epoch we want to use. Furthermore it seemed the authors performed their hyper-parameter tuning on the test-set as well. To combat this, we extracted a validation set from the training data. A random subset of 10% of the items were used as validation data. We then performed epoch selection using the validation set and evaluated the best model on the final test set. The results deviate more than previously, most probably due to a reduced training set. Applying this procedure avoids overfitting on test data, but as a result the differences increased, the recall lowered by 8% and NDCG lowered by 12%(Table 2).

4.3 Ablation study

The ablation study show that the full UltraGCN model outperformed the other variants, in both our study and the original paper (Figure 1, Figure 2). That means, the model is more effective with learning from item-item relationships and graph convolution during learning on the user-item graph. For both metrics, the other variants do not seem to differ much in their performance, although the most basic model appears to be slightly worse. This trend was similarly reported in the original paper, but with larger magnitude of differences between the variants. To test for a significant difference between the reported and our reproduced results a Mann-Whitney U test was performed, which tests the measured results for having the same underlying distribution. The test does not find a significant difference for either recall or NDCG (reporting a p-value of 0.89 resp. 0.47 with a significance level of 0.05).

4.4 Parameter influence

The impact of parameter K, the number of selected neighbors, was assessed in the original paper (Figure 3b) and reproduced here (Figure 3a) for Amazon-books dataset. The performance was tested with $K \in \{5, 10, 20, 50\}$. In our case, as K increases, the effectiveness decreases. However, in the original paper, an initial increase of effectiveness is reported when K went from 5 to 10. The effectiveness metrics we observed are somewhat lower than reported in the original

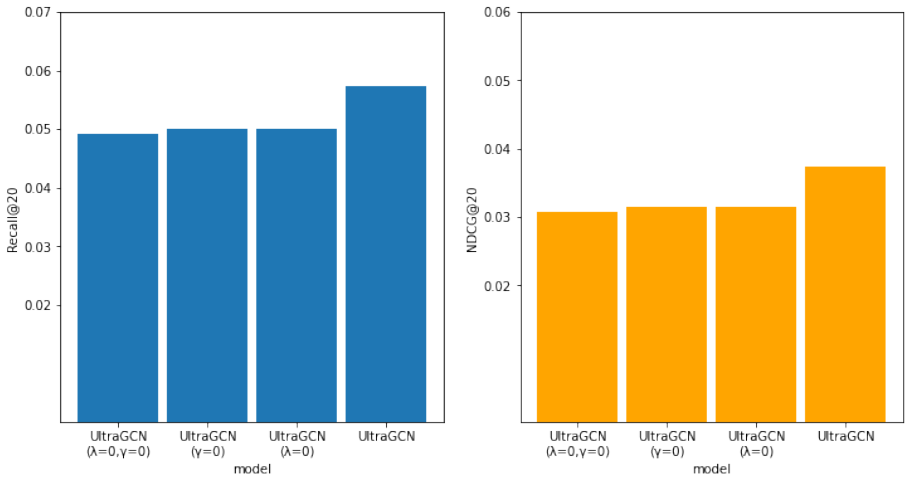


Fig. 1. Ablation reproduced: performance of variants of UltraGCN. Left: performance measured by Recall@20. Right: performance measured by NDCG@20. Three variants are compared to the full UltraGCN model.

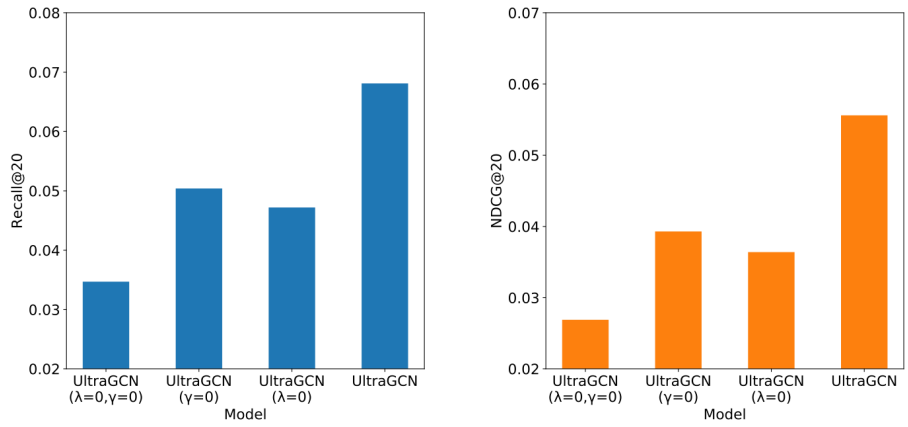
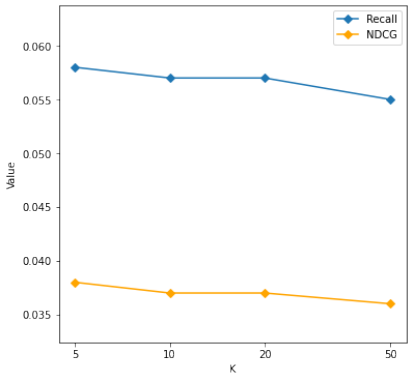


Fig. 2. Ablation original: performance of variants of UltraGCN. Left: performance measured by Recall@20. Right: performance measured by NDCG@20. Three variants are compared to the full UltraGCN model.

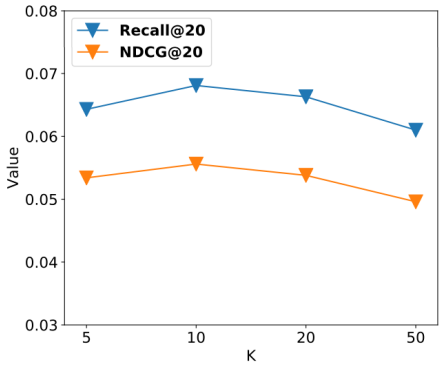
paper. The difference of the optimal number of neighbors might also be explained with the usage of a reduced dataset. To test for significant difference again a Mann-Whitney U test was performed. Here, the test does find a significant difference for both recall and NDCG (reporting a p-value of 0.03 for both with a significance level of 0.05).

5 CONCLUSION

Overall the original paper on UltraGCN offered significant resources for replication - readily available datasets, code for the complete model, clear environment requirements. However, some aspects were lacking - the statistical testing procedure was unclear and random seeds were not provided. Where our computational power was sufficient for the whole dataset, the results were very close. However, when we used only a subset of the dataset, it was not possible to



(a) Impact of K reproduced: performance of UltraGCN under different K parameter.



(b) Impact of K original: performance of UltraGCN under different K parameter.

produce matching results, which is not very surprising. Also the original paper benchmarked UltraGCN against other models. Providing access to the other models would have enabled us to reproduce a relative performance comparison.

REFERENCES

[1] Juliana Freire, Norbert Fuhr, and Andreas Rauber. 2016. Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). *Dagstuhl Reports* 6, 1 (2016), 108–159. https://doi.org/10.4230/DagRep.6.1.108

[2] 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* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 639–648. https://doi.org/10.1145/3397271.3401063

[3] 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 amp; Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 1253–1262. https://doi.org/10.1145/3459637.3482291