Profiling the Design Space for Graph Neural Networks based Collaborative Filtering

Zhenyi Wang*
Beijing University of Posts and Telecommunications
China
zy_wang@bupt.edu.cn

Huan Zhao*
4Paradigm Inc.
China
zhaohuan@4paradigm.com

Chuan Shi†
Beijing University of Posts and Telecommunications
China
shichuan@bupt.edu.cn

ABSTRACT

In recent years, Graph Neural Networks (GNNs) have been widely used in Collaborative Filtering (CF), one of the most popular methods in recommender systems. However, most existing works focus on designing an individual model architecture given a specific scenario, without studying the influences of different design dimensions. Thus, it remains a challenging problem to quickly obtain a top-performing model in a new recommendation scenario. To address the problem, in this work, we make the first attempt to profile the design space of GNN-based CF methods to enrich the understanding of different design dimensions as well as provide a novel paradigm of model design. Specifically, a unified framework of GNN-based CF is proposed, on top of which a design space is developed and evaluated by extensive experiments. Interesting findings on the impacts of different design dimensions on recommendation performance are obtained. Guided by the empirical findings, we further prune the design space to obtain a compact one containing a higher concentration of top-performing models. Empirical studies demonstrate its high quality and strong generalization ability[1].

CCS CONCEPTS

• **Information systems → Collaborative filtering**.

KEYWORDS

Collaborative filtering; Graph neural networks; Empirical evaluation

ACM Reference Format:
Zhenyi Wang, Huan Zhao, and Chuan Shi. 2022. Profiling the Design Space for Graph Neural Networks based Collaborative Filtering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498520

*Equal contribution.
†Corresponding author.
[1]Our code is available at https://github.com/BUPT-GAMMA/Design-Space-for-GNN-based-CF.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498520

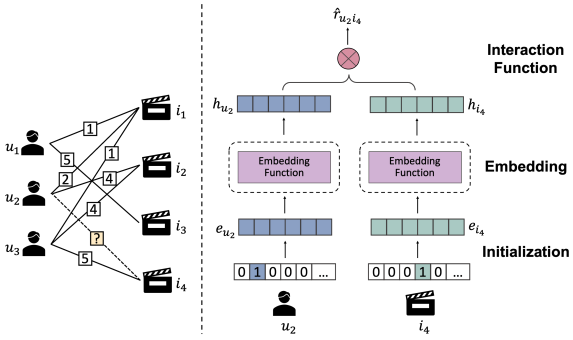


Figure 1: An illustration of user-item bipartite graph (left) and the general embedding-based CF framework taking the rating prediction procedure of u_2 to i_4 as an example (right).

1 INTRODUCTION

Collaborative Filtering (CF) has been one of the most popular recommender system (RS) methods, which aims at predicting users' preferences based on those who share similar behaviors [12, 17, 20, 30]. In the literature, most CF methods follow an embedding-based paradigm as shown in Figure 1 (right), which first learns low-dimensional representations for users and items, and then uses an interaction function to predict the preferences (ratings) of users to items. Classic Matrix Factorization (MF) [16, 24] directly adopts dot product as the interaction function while Neural Collaborative Filtering (NCF) [11] proposes to learn interaction function with neural networks. Very recently, Graph Neural Networks (GNNs) have been incorporated for the CF tasks, since the user-item interactions can be naturally modeled as a bipartite graph as shown in Figure 1 (left) and GNN can learn better user/item representations by capturing the high-order information in the user-item bipartite graph through an iterative message passing (neighborhood aggregation) [39, 41]. Numerous successful GNN-based models have been proposed and shown promising results for CF tasks, such as PinSage [41], NGCF [36], LightGCN [10], and MCCF [38].

Existing GNN based methods are mainly limited to designing a *single* best architecture for a specific scenario, while in real-world applications, the recommendation scenarios are diverse as the datasets differ in their collected domains and properties like scale (large or small) and density (dense or sparse). Such diversity makes it a common practice to design different architectures across different scenarios. To be more specific, the choice of each *design dimension*, e.g., aggregation function or activation function, in the

chosen model varies across recommendation scenarios. For example, in PinSage [41], the non-linear activation function (`ReLU`) is a natural choice in the deployed architecture in Pinterest, while in a following-up work, LightGCN [10], the authors show that the non-linear activation functions do not always benefit the final performance in other benchmark datasets. Besides, despite various new GNN-based models for CF have been developed, little has been done to systematically understand *the influences of different design dimensions of GNN-based CF on recommendation performance*. Therefore, each time given a new scenario, huge efforts including computational resources and human expertise have to be invested in exploring the entire huge space of the possible combinations of all dimensions of GNN models to obtain a top-performing model.

To address the problem, in this work, we propose to profile the *design space*, a Cartesian product of multiple *design dimensions* [42], of existing GNN-based methods for CF by empirical evaluation. It not only provides a deeper understanding of the influences of different dimensions on recommendation performance, but also paves the way for a novel paradigm to efficiently design top-performing GNN methods in different CF scenarios.

Specifically, we firstly propose a unified framework consisting of 4 key modules, i.e., initialization, GNN, multi-component, and interaction, as illustrated in Figure 2(a), which most GNN-based CF methods can fit into. Then on top of this framework, we develop a design space which covers important design dimensions of GNN-based CF models. By incorporating popular choices for each design dimension (Figure 2(b)), we obtain a design space with more than 100,000 different model architectures, even including classic MF methods [16, 24] and Multi-Layer Perceptron (MLP), i.e., NCF [11]. Obviously, it is too costly to train and tune all models in the entire design space. Thus, we adopt the controlled random search [42], which can provide an efficient and effective way to evaluate the impacts of different design dimensions. By training ~3,400 models on 9 real-world datasets of different domains, scale, and density, some interesting findings are obtained to help us better understand the influences of the proposed design dimensions. Taking the activation function as an example again, in our evaluation results, `Sigmoid` tend to perform better than other choices including `Identity`, i.e., no activation function (see Section 4.4 for more details).

Based on the empirical insights, furthermore, we prune the vanilla design space by narrowing down the choices of design dimensions, leading to a more compact design space consisting of only 96 model instances, more than 1,000 times smaller than the vanilla one. To verify the superiority of the pruned search space compared to the vanilla one, we empirically compare the performance distribution of the sampled models in the 2 design spaces [26]. Then extensive experiments are conducted in different popular settings in terms of recommendation scenarios, e.g., different levels of sparsity, different model complexity, and new datasets, to show that the pruned search space contains a higher concentration of top-performing CF methods. Finally, as a case study, we perform a random search on the pruned search space and compare the best searched model with popular CF models, like MF [16], NCF [11], and LighGCN [10]. The experimental results show that the searched model can obtain the best performance, which further demonstrates the quality of the pruned search space and provide a novel paradigm of model design in recommendation scenarios.

To summarize, this work makes the following contributions:

- To the best of our knowledge, we make the first attempt to profile the design space of GNN-based CF. It not only deepens the understanding of different dimensions, but also provides a novel paradigm to design GNN methods in recommendation scenarios.
- A unified framework, which can cover extensive popular GNN-based CF models, is proposed. On top of this framework, we develop a design space and evaluate it by extensive experiments. Interesting findings are obtained to provide insights into model design.
- Guided by the insights, we prune the vanilla design space to obtain a compact one containing a higher concentration of top-performing models. Empirical studies demonstrate the high quality and strong generalization ability of the pruned design space.

2 RELATED WORK

2.1 GNN-based Collaborative Filtering

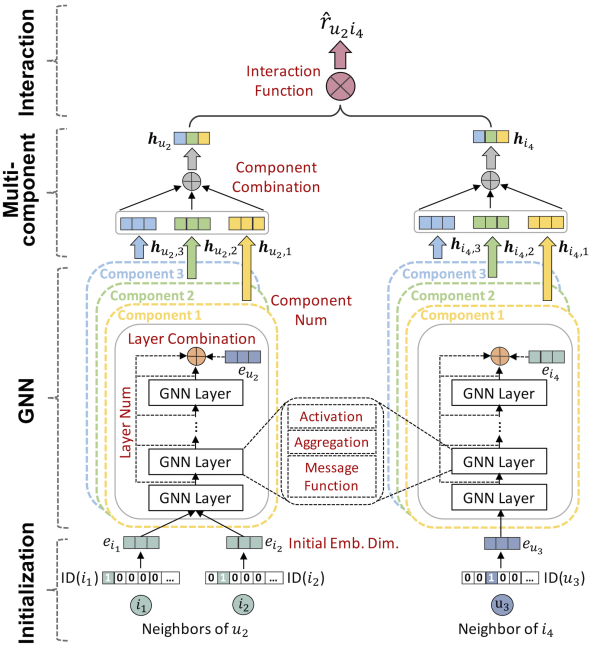
Collaborative Filtering (CF), learning user preferences by parameterizing users and items as embeddings based on the interactions between them, is the fundamental technique in modern recommender systems [12, 17, 20, 30]. Matrix Factorization (MF) [16, 24], one of the most traditional CF techniques, projects the one-hot ID of a user/an item into an embedding vector and then utilizes the dot product of the embedding vectors of users and items to reconstruct the user-item interaction matrix. The recent success of deep learning has motivated a wave of studies focusing on incorporating neural components into CF models [11]. For example, NCF [11] uses Multi-Layer Perceptron (MLP) to replace the conventional dot product as the user-item interaction function.

Recent years have witnessed the great success of Graph Neural Networks (GNNs) in CF for learning effective user/item representations [1, 2, 10, 36–38, 41, 43]. For example, NGCF [36] decides the message propagating on both the graph structure and the affinity between the central node and combines the representations of different layers to get the final node representation. LightGCN [10] argues that the non-linearity and weight matrices are useless for CF without side information, and proposes a simple GNN-based CF model. DGCF [37] disentangles the user/item representations into several components to reflect user preference from multiple aspects. More works on GNN-based recommendation can be checked in a latest survey [39].

Despite various GNN-based models developed, they only design a specific model architecture tuned for a specific scenario while our work focuses on a novel paradigm, i.e., profiling the design space, to facilitate the design of top-performing models in diverse recommendation scenarios.

2.2 Empirical Evaluation of GNN and CF

Recently, there has been an emerging trend of conducting evaluation studies on GNNs and CF, respectively. Towards GNNs, Shchur et al. [31] investigate the influence of dataset split and compare the aggregated evaluation results on different dataset splits. Lv et al. [22] point out the issues in heterogeneous GNNs comparison and propose a heterogeneous graph benchmark. Zhang et



(a) The Unified Framework

(b) Choices in the Design Dimensions

Design Dimension	Choices
Initial Embedding Dimension d	64, 128, 256
Message Function $m(\cdot)$	Identity, Hadamard
Aggregation $f(\cdot)$	None, GCN, GAT, GIN, GraphSAGE
Activation $\sigma(\cdot)$	Identity, Sigmoid, Tanh, ReLU, PReLU, LeakyReLU
Layer Number L	1, 2, 3, 4
Layer Combination $g(\cdot)$	Stack, Concat, Sum, Mean
Component Number K	1, 2, 3, 4
Component Combination $c(\cdot)$	Concat, Mean, Att
Interaction Function $p(\cdot)$	Dot Product, Concat+MLP, Sum+MLP

(c) Controlled Random Search

Group No.	Configuration Space					Experimental Results	
	Message Function	Activation	...	Interaction Function	Dataset	Performance	Ranking
1	Identity	ReLU	...	Dot Product	Yelp	0.8996	2
	Hadamard					0.8665	1
2	Identity	Sigmoid	...	Concat+MLP	Amazon-CDs	0.7636	1
	Hadamard					0.7812	2
...							
5	Identity	Tanh	...	Sum+MLP	MovieLens-1M	0.8231	1(tie)
	Hadamard					0.8231	1(tie)

(d) Ranking Analysis of $m(\cdot)$

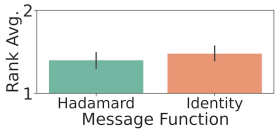


Figure 2: Overview of the design space and the evaluation strategy. (a) An illustration of the unified framework of GNN-based CF suppose that the rating of the same user-item pair (u_2, i_4) as in Figure 1 is to be predicted. It contains 4 modules: initialization, GNN, multi-component, and interaction. On top of the framework, we propose a GNN-based CF design space consisting of 9 design dimensions, which are marked in red. (b) Popular choices in the 9 design dimensions are listed. (c) An example of performing controlled random search to investigate the influence of message function $m(\cdot)$. In each group of configurations, the design choices of $m(\cdot) \in$ {`Identity`, `Hadamard`} are ranked by performance. (d) The average of the rankings corresponding to each design choice is shown for analysis. Lower is better.

al. [44] compare the performance of embedding a network into hyperbolic space with Euclidean space. Previous works in the recommendation domain have also discussed the worrying current situation of reproducibility issue and fair evaluation of CF techniques and several recently proposed complicated models are found to be outperformed by simple and well-optimized baseline algorithms [3, 19, 28, 29, 32, 48].

However, as a prominent CF technique, empirical evaluations on GNN-based CF have received relatively less scrutiny in existing works. Furthermore, the aforementioned evaluation works focus on comparing the individual model performance, while our approach elevates the study level to design space, i.e., a population of models, which guarantees stronger robustness and generalization. You et al. [42] explores the design space and the task space of GNN and conduct experiments to provide guidelines for better GNN model design. Faced with more domain-specific challenges, e.g., sparsity and diverse user interests, we accordingly customize the evaluation, make a more thorough analysis, and draw more valuable conclusions that enhance research compared to the link prediction task of GraphGym [42]. We further develop a pruned design space containing a higher concentration of top-performing models that can work well in different recommendation settings.

3 DESIGN SPACE OF GNN-BASED CF

3.1 The Unified Framework

As mentioned in the introduction, the existing GNN-based CF could be divided into 4 key modules, i.e., *initialization*, *GNN*, *multi-component*, and *interaction*. In general, as illustrated in Figure 2(a), we propose a unified framework composed of the 4 modules. Taking the representation update procedure on the user side as an example, we elaborate on the framework as follows.

(1) **Initialization** projects the one-hot user/item ID to dense real-valued embeddings by embedding matrix ID-lookup, i.e., $\mathbf{e}_u = lookup\,(ID\,(u))$, $\mathbf{e}_i = lookup\,(ID\,(i))$, where $\mathbf{e}_u$ (resp. $\mathbf{e}_i$) is the initial embedding of user u (resp. item i). It is a standard manner when there are no extra available features such as user profiles or item attributes.

(2) **GNN** feeds the initial embeddings of users/items into GNN, refines the embeddings through the propagation of GNN layers, and obtains the representations by combining the output of each GNN layer (including the initial embedding), as follows,

$$\mathbf{h}_u^{(0)} = \mathbf{e}_u, \mathbf{h}_i^{(0)} = \mathbf{e}_i, \tag{1}$$

$$\mathbf{m}_{u\leftarrow i}^{(l)} = m\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_i^{(l-1)}\right), \tag{2}$$

Table 1: A summary of the popular CF methods that can be instantiated from the design space. Initial embedding dimension d, layer number L, and component number K are omitted since they do not affect the overall model architecture and the last column indicates if multi-component is taken into consideration during model design. The notations are introduced in Figure 2(b).

Category	Model	$m(\cdot)$	$f(\cdot)$	$\sigma(\cdot)$	$g(\cdot)$	$c(\cdot)$	$p(\cdot)$	Single-/Multi-component
Classic	MF [16, 24]	`Identity`	`None`	`Identity`	`Stack`	-	`Dot Product`	`Single`
	LLORMA [18, 46]	`Identity`	`None`	`Identity`	`Stack`	`Att`	`Dot Product`	`Multiple`
MLP-based	NCF [11]	`Identity`	`None`	`ReLU`	`Stack`	-	`Concat+MLP`	`Single`
GNN-based	NGCF [36]	`Hadamard`	`GCN`	`LeakyReLU`	`Concat`	-	`Dot Product`	`Single`
	LightGCN [10]	`Identity`	`GCN`	`Identity`	`Mean`	-	`Dot Product`	`Single`
	LR-GCCF [2]	`Identity`	`GCN`	`Identity`	`Concat`	-	`Dot Product`	`Single`
	SMOG-CF [43]	`Hadamard`	`GCN`	`ReLU`	`Concat`	-	`Dot Product`	`Single`
	PinSage [41]	`Identity`	`GraphSAGE`	`ReLU`	`Stack`	-	`Dot Product`	`Single`
	MCCF [38]	`Identity`	`GAT`	`ReLU`	`Stack`	`Att`	`Concat+MLP`	`Multiple`
	DGCF [37]	`Identity`	`GCN`	`Tanh`	`Sum`	`Concat`	`Dot Product`	`Multiple`

$$\mathbf{h}_u^{(l+1)} = \sigma\left(f\left[\left\{\mathbf{m}_{u\leftarrow i}^{(l)}, \forall i \in \mathcal{N}(u)\right\}, \mathbf{h}_u^{(l)}\right]\right), \tag{3}$$

$$\mathbf{h}_u = g(\mathbf{h}_u^{(0)}, \mathbf{h}_u^{(1)}, \cdots, \mathbf{h}_u^{(L)}), \tag{4}$$

where $\mathbf{h}_u^{(l)}$ (resp. $\mathbf{h}_i^{(l)}$) denotes the refined representation of user u (resp. item i) after l GNN layers, $m(\cdot)$ is the message function to encode the message flow from item i to user u as $\mathbf{m}_{u\leftarrow i}^{(l)}$, σ is the activation function, $\mathcal{N}(u)$ is the neighborhood of user u, $f(\cdot)$ is the neighborhood information aggregation technique in each layer, $g(\cdot)$ is the layer combination function of the $L+1$ representations after propagating L layers.

(3) **Multi-component** learns how to better model diverse user interests from different aspects by disentangling user/item representations into multiple components [37, 38]. Specifically, K independent embedding procedures are performed according to Equation (1)-(4), and K representations corresponding to each component are obtained as $\mathbf{h}_{u,1}, \mathbf{h}_{u,2}, \cdots, \mathbf{h}_{u,K}$, which are combined to obtain the final representation, as follows,

$$\mathbf{h}_u = c(\mathbf{h}_{u,1}, \mathbf{h}_{u,2}, \cdots, \mathbf{h}_{u,K}), \tag{5}$$

where $c(\cdot)$ is the component combination function of the K representations from each component to obtain the final user representation.

(4) **Interaction** performs the user-item matching and predicts the rating value of user-item pair to reflect user preference, as follows,

$$\hat{r}_{ui} = p\left(\mathbf{h}_u, \mathbf{h}_i\right), \tag{6}$$

where $p(\cdot)$ is the interaction function to predict $\hat{r}_{ui}$ as the rating of the user-item pair (u, i).

Due to the symmetry of the user-item bipartite graph, we can get similar formulations on the item side.

3.2 The Proposed Design Space

Based on the unified framework, 9 *design dimensions* can be extracted, which are split over the 4 modules and marked in red in Figure 2(a), with various possible *design choices* shown in Figure 2(b). The proposed model architectures vary in their design choices. For example, LightGCN [10] and LR-GCCF [2] claim that non-linear activation doesn't benefit CF so remove it from their proposed architectures while it still remains in many other works [36, 38, 41]. Different combinations of design choices generate different model instantiations with different recommendation performance. Therefore, the specific choices in the design dimensions should be taken into serious consideration during model design. To explore the impacts of different design dimensions, we propose to profile the *design space*, defined as the Cartesian product of design dimensions [42], which contains a population of model instantiations. Note that our purpose is not to propose the most extensive design space, but to help to understand the influences of different design dimensions of GNN-based CF and gain insights for designing well-performing models. In fact, the design dimensions and their ranges can be naturally expanded by incorporating more choices.

We elaborate on some important design dimensions as follows. Others can be naturally understood from the table in Figure 2(b).

- **Message function** $m(\cdot)$. In the literature, the common practice is to directly set $m(\cdot) = \mathbf{h}_u^{(l-1)}$ [10, 37, 38, 41], but some works [36, 43] claim that the interaction between the source and the target node should be encoded into message. Therefore, we also set $m(\cdot) = \mathbf{h}_u^{(l-1)} \odot \mathbf{h}_i^{(l-1)}$, where $\odot$ denotes the hadamard multiplication of two vectors. The above two design choices are denoted as `Identity` and `Hadamard`, respectively.
- **Aggregation** $f(\cdot)$. For this design dimension, we consider four common and effective GNN methods as its design choices: `GCN` [15], `GAT` [34], `GIN` [40], and `GraphSAGE` [8]. Particularly, we generalize the design dimension to include the choice of `None`, which represents not exploiting the graph information and refining the user/item representations through MLP, to enlarge the capacity of the design space to include those non-GNN-based models [11, 16].
- **Layer combination** $g(\cdot)$. `Stack` represents directly stacking multiple GNN layers and using the output of the final layer [38, 41] to obtain the representation corresponding to each component. Since the outputs of intermediate layers are also found

useful in previous works [10, 36, 37], we investigate three additional layer combination approaches: `Concat`, `Sum`, and `Mean`.

- **Component combination** $c(\cdot)$. A direct strategy used in existing works, such as DGCF [37], is to get the representations concatenated, denoted as `Concat`. The attention mechanism can also be used [38], denoted as `Att`. And we add an additional design choice `Mean`.
- **Interaction function** $p(\cdot)$. A simple yet effective choice is calculating the dot product of the user and item representations, denoted as `Dot Product`. Neural networks can also be used for learning an interaction function [11]. The user and item representations are first concatenated or summed up, and then fed into an MLP for prediction, which are denoted as `Concat+MLP` and `Sum+MLP`, respectively.

3.3 Relationship with Existing CF Methods

Table 1 shows 10 popular CF methods that can be instantiated from the proposed design space. Specifically, the methods can be categorized into 3 groups: (1) classic methods which mainly refer to MF and its variants; (2) MLP-based methods which incorporate neural networks for CF; (3) GNN-based methods which enhance CF by GNNs. We next choose 3 representative models from each category and briefly explain how they can be instantiated by the design space.

- **MF** [16] is the most common CF method which uses the dot product of the user and item representations to reconstruct the user-item interaction matrix. It can be naturally instantiated by setting the aggregation as `None` and adopting `Dot Product` as the interaction function.
- **NCF** [11] proposes to learn interaction function with neural networks. It can be included by choosing `Concat+MLP` as the interaction function.
- **LightGCN** [10] is a state-of-the-art GNN-based CF method, which can be considered as choosing `GCN` as the aggregation technique, `Identity` function as the activation, i.e., removing the nonlinear computations, and `Mean` as the layer combination function.

We can see that the framework unifies the key design dimensions in popular CF models and the proposed design space is comprehensive enough to include a broad spectrum of model instantiations. In the next section, we then perform an evaluation of the design space to understand the impacts of different design dimensions.

4 EVALUATION OF THE DESIGN SPACE

4.1 Datasets

As discussed in the introduction, the recommendation scenarios in real-world applications are diverse. To make the experimental findings more robust, the evaluation is conducted on 9 real-world datasets, which are diverse in scale (large or small), density (dense or sparse), and the collected domain. The statistics of the datasets are shown in Table 2, whose detailed descriptions and preprocessing strategies can be found in Appendix A.

Table 2: Statistics of the datasets.

Dataset	# of Users	# of Items	# of Interactions	Rating Scale	Density
Yelp[1]	58,069	31,721	1,160,605	[1,5]	0.063%
Amazon-CDs [9]	31,296	24,379	622,163	[1,5]	0.082%
Amazon-Movies [9]	44,439	25,047	1,070,860	[1,5]	0.096%
YahooMusic [5, 25]	1,357	1,363	5,335	[1,100]	0.28%
Amazon-Beauty [9]	7,068	3,570	79,506	[1,5]	0.32%
Flixster[13, 25]	2,341	2,956	26,173	[0.5,5]	0.38%
Douban [23, 25]	2,999	3,000	136,891	[1,5]	1.52%
MovieLens-1M[2]	6,040	3,706	1,000,209	[1,5]	4.47%
MovieLens-100K[3]	943	1,682	100,000	[1,5]	6.31%

[1] https://www.yelp.com/dataset/
[2] https://grouplens.org/datasets/movielens/100k/
[3] https://grouplens.org/datasets/movielens/1m/

Table 3: Choices of the hyperparameter dimensions.

Hyperparameters	Choices
Dropout	`0, 0.5`
Training Epochs	`120, 160, 200`
L_2 Regularization Weight λ	`0.0005, 0.005, 0.05`

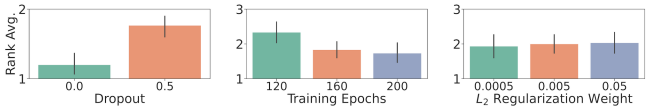


Figure 3: Ranking analysis of the hyperparameter dimensions. Lower is better. The values of these hyperparameters are fixed as the optimal design choices.

4.2 Evaluation Technique

With over 100,000 model architectures in the design space, conducting full grid search to evaluate each design dimension is too costly. To overcome this issue, we adopt *controlled random search* [42] as the design space evaluation strategy.

To make the evaluation distributed across different datasets, we first define the Cartesian product of design dimensions and datasets as the *configuration space*, and the controlled random search is performed in the configuration space to draw experimental configurations. As illustrated in Figure 2(c), suppose that we want to evaluate message function $m(\cdot)$, we first draw S experimental configurations by random searching the configuration space, all with $m(\cdot)$ = `Identity`. Then, by setting $m(\cdot)$ = `Hadamard`, while *controlling all the other dimensions*, we draw another S configurations. Now, we obtain S groups, all with 2 configurations that only differ from each other in $m(\cdot)$. Within each group, the 2 design choices of $m(\cdot) \in$ {`Identity`, `Hadamard`} are ranked by performance, where a tie is given if the performance difference is less than 0.0001. The average rankings of different choices over all the S groups are shown via bar plot as illustrated in Figure 2(d). In our experiment, we set $S = 100$ to cut the number of experiments from 103,680 to 3,400, by over 30 times.

4.3 Evaluation Setup

4.3.1 Loss Function and Evaluation Metric. Since our task is predicting ratings for user-item pairs, we adopt the widely-used *Mean*

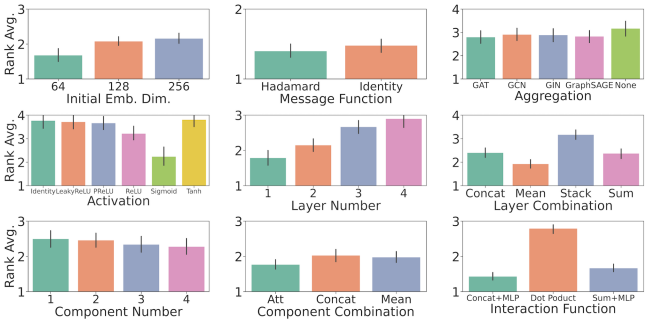


Figure 4: Ranking analysis of the 9 design dimensions. Lower is better.

Square Error (MSE) loss function,[2] which is formulated as follows:

$$L = \frac{\sum_{(u,i)\in O_t} (\hat{r}_{ui} - r_{ui})^2}{|O_t|} + \lambda \|\Theta\|^2, \tag{7}$$

where $\hat{r}_{ui}$ and r_{ui} are the predicted rating and the ground truth, respectively, O_t is the set of observed ratings for training, λ is the hyperparameter controlling the L_2 regularization weight and Θ denotes the model parameters.

As for the evaluation metric, we use the common *Rooted Mean Square Error (RMSE)* for the rating prediction task [16], which is calculated by $RMSE = \sqrt{\frac{\sum_{(u,i)\in O_e} (\hat{r}_{ui} - r_{ui})^2}{|O_e|}}$, where O_e is the set of observed ratings for testing.

4.3.2 Hyperparameters. The hyperparameters of the sampled models should be the same to ensure a fair comparison. The values of 3 hyperparameters, i.e., dropout, training epochs, and L_2 regularization weight λ are set according to the evaluation result of controlled random search on the validation set. Their available choices are listed in Table 3 and the evaluation result is shown in Figure 3. The rest of the hyperparameters are set as common values in practice. We refer the readers to Appendix B.1 for more detailed hyperparameter settings and C.1 for other implementation details.

4.4 Evaluation Results

Results are shown in Figure 4 with 9 bar plots, each depicting the averaged rankings of different choices in each design dimension (we further show the violin plots to decipt the ranking distribution in Appendix D). The experiment quantitatively evaluates the impacts of different design dimensions of GNN-based CF on recommendation performance across a wide range of recommendation scenarios.

Rather than searching for the single best model out of all these configurations, we explore whether there are findings that can enrich the understanding of the design dimensions and help to efficiently design top-performing GNN-based CF models in different recommendation scenarios. Some key experimental findings are enumerated as below:

- `GAT` and `GraphSAGE` slightly outperform the other alternatives. Interestingly, `None` is comparable with GNN-based aggregators, indicating that simply using MF [16] or MLP-based CF methods [11] can achieve competitive or even better performance in some scenarios. The interesting finding reveals that incorporating graph information can not always enhance CF.
- `Sigmoid` clearly stands out among all the 6 activations. This finding differs from that in [10], which finds out that the non-linear activation can not benefit CF. A possible explanation is that the tasks studied in the two works are different: item ranking in [10] while rating prediction in ours, where non-linear activation can help to increase the expressive power of the neural networks for such a regression task.
- When taking multi-component into consideration, it is more favorable to set component number as `4`, which aligns with the finding of previous work [37] that the user interests are diverse in different aspects. And it is preferable to combine representations of different components with `Att` mechanism.
- Adopting neural interaction function is superior to adopting `Dot Product`, which is not consistent with the finding in [28] that `Dot Product` is a better choice to MLP for predicting the ratings for user-item pairs. We suppose it may be caused by the different settings of the two works. In [28], the evaluation is performed on specific model architectures with 2 datasets, while in our setting, we evaluate thousands of model architectures on 9 different datasets, and thus, different conclusions are drawn.

The above findings not only enrich our understanding of the impacts of different design dimensions but further provide valuable insights for effectively designing top-performing models. Specifically, we can observe that there exists some redundancy in the design space. For example, the initial embedding dimension can be fixed as 64 since it significantly outperforms the other 2 alternatives. It motivates us that the vanilla design space can be further pruned to improve its quality, which will boost the searching efficiency of top-performing models.

5 EVALUATION OF PRUNED DESIGN SPACE

Following the insights provided in Section 4.4, we prune the vanilla design space by narrowing down the choices of design dimensions. The motivation is that we only remain those favorable design choices, which are empirically more likely to generate well-performing models. Thus, the pruned design space contains a higher concentration of top-performing models which will facilitate model searching.

5.1 The Pruned Design Space

Table 4 introduces the choices in the design dimensions of the pruned design space. We briefly explain the pruning in 3 dimensions, and for the other dimensions, preferable design choices are remaining, which can be naturally understood from Figure 4. For aggregation, `GraphSAGE` remains as the representative of graph information aggregator for its smaller training consumption than `GAT`, and `None` also remains to make the pruned design space have a capacity of non-GNN-based models. As for the activation, two preferable non-linear functions and `Identity` remain for better containment of linear and non-linear models. The optimal component number 4 remains, and we also keep the choice of 1 for investigating single-component models.

[2] In this work, we mainly study the rating prediction task under MSE loss and leave the study of the influence of different types of loss function as future work.

Table 4: Choices in the design dimensions of the pruned design space.

Design Dimension	Choices
Initial Embedding Dimension d	`64`
Message Function $m(\cdot)$	`Identity`, `Hadamard`
Aggregation $f(\cdot)$	`None`, `GraphSAGE`
Activation $\sigma(\cdot)$	`Identity`, `Sigmoid`, `ReLU`
Layer Number L	`1`, `2`
Layer Combination $g(\cdot)$	`Mean`
Component Number K	`1`, `4`
Component Combination $c(\cdot)$	`Att`
Interaction Function $h(\cdot)$	`Concat+MLP`, `Sum+MLP`

After pruning, there are only 96 candidate models in the design space, compared to 103,680 in the vanilla one. The scale of the design space is reduced by three orders of magnitude (1,080x). Compared to the vanilla one, the superiority of the pruned design space lies in that it simplifies the combinations of design dimensions by ruling out the sub-optimal choices, and thus contains a higher concentration of top-performing models which work well across scenarios for efficient searching. The pruned design space consistently shows high quality in different recommendation settings, which indicates its strong generalization ability. In the remainder of this section, we then conduct a series of evaluation studies to verify the superiority of the pruned design space.

5.2 Evaluation

5.2.1 Evaluation Technique. We use the RMSE *empirical distribution function* (EDF) [26] to quantify the quality of a design space by characterizing the RMSE distribution generated by sampling and training n models from that design space. We empirically observe that sampling $n = 100$ samples is sufficient. For more discussions on EDF and the detailed selection process for n, please refer to Appendix E.

5.2.2 Generalization Evaluation. To show that the pruned design space can facilitate model design across different scenarios, we conduct the following empirical studies to evaluate its generalization. Specifically, we compare the EDFs in 3 different popular settings in terms of recommendation scenarios. The experimental results demonstrate that the quality of the pruned design space consistently outperforms that of the vanilla one.

Different levels of density. Interaction density is an important property to characterize a recommendation dataset and it diversifies in different recommendation scenarios. As shown in Table 2, datasets can be divided into 3 groups according to the order of magnitude of their interaction density. We select Yelp, Amazon-Beauty, and MovieLens-1M as the representatives of each group, and the EDF comparison results on the 3 datasets are shown in Figure 5.

Under all the 3 levels of density, the performance distribution of the pruned design space concentrates within a compact and high-valued range. Specifically, on Yelp and Amazon-Beauty, the pruned design space contains the models with the best performance in the vanilla design space; and in MovieLens-1M, although the models

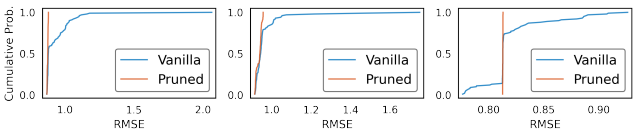


Figure 5: Comparison results of design space generalization to different levels of density. Yelp (left), Amazon-Beauty (middle), and MovieLens-1M (right).

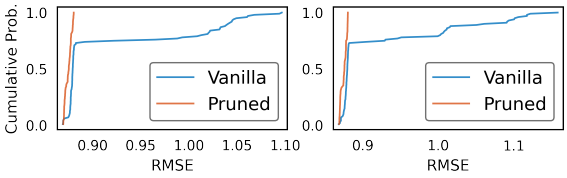


Figure 6: Comparison results of design space generalization to different model complexity on Yelp. Low (left) and high model complexity (right).

of the pruned design space may be outperformed by about 10% of those in the vanilla one, we still significantly narrow down the performance range of the models to a high-valued area, making it more efficient to find models with satisfactory performance.

Different model complexity. Recently, there is a growing interest in deploying customized neural architectures on diverse hardware devices with different computational resource constraints [33]. Therefore, the requirements of model complexity vary across different platforms. By regarding the number of trainable parameters as the indicator of model complexity, we compare the quality of the vanilla and the pruned design space under the constraints of low and high model complexity, respectively. Specifically, we use a model with $d = 64, L = 1, K = 1, f(\cdot) =$ `None` as reference to set the low complexity constraint; and the other with $d = 256, L = 4, K = 4$, $f(\cdot) =$ `GraphSAGE` to set the high complexity constraint (both with 64 hidden dimensions). The number of hidden dimensions of the sampled architecture is adjusted to match the model complexity constraints. We choose Yelp as the experimental dataset and the comparison results are shown in Figure 6.

We can observe that model performance in the pruned design space distributes in a small but top-valued area under both levels of model complexity, demonstrating its superior performance compared to the vanilla design space.

Generalization to new datasets. To further show the generalization ability to new recommendation scenarios, we sample and train the models from each design space on 2 new datasets: Epinions and Amazon-Sports, whose statistics are shown in Table 5. The EDFs of the 2 design spaces on the new datasets are compared, respectively, in Figure 7.

Again, we can observe that the EDF of the pruned design space is improved substantially on both of the new datasets, which demonstrates the stronger robustness and generalization of the pruned design space given new recommendation scenarios.

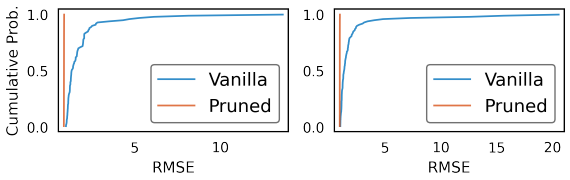


Figure 7: Comparison results of design space generalization to new datasets. Epinions (left) and Amazon-Sports (right).

Table 5: Statistics of the 2 new datasets.

Dataset	# of Users	# of Items	# of Interactions	Rating Scale	Density
Epinions[1]	40,163	139,738	664,824	[1,5]	0.012%
Amazon-Sports [9]	11,435	5,405	108,004	[1,5]	0.17%

[1] http://www.trustlet.org/downloaded_epinions.html

Table 6: The best searched model architectures with random sample number m=10 on Epinions and Amazon-Sports. Only the variable design dimensions in Table 4 are listed, and the choices in the remaining design dimensions remain the same as those in Table 4.

Dataset	$m(\cdot)$	$f(\cdot)$	$\sigma(\cdot)$	L	K	$p(\cdot)$
Epinions	Hadamard	None	ReLU	1	1	Sum+MLP
Amazon-Sports	Identity	GraphSAGE	Sigmoid	1	4	Sum+MLP

To summarize, in all the above 3 settings, the pruned design space consistently holds better quality than that of the vanilla design space without signs of overfitting, which demonstrates its strong generalization to various new settings.

5.3 Case Study: Random Search

A case study is conducted to further verify if the pruned design space with a higher concentration of top-performing models can enhance model searching in new scenarios. Since our focus is the design space, the searching can be performed with exiting graph neural architecture search algorithm [4, 7, 35, 45, 47], for which we leave this as future work. Here, we simply adopt random search. Specifically, we randomly search and train $m = 10$ models in the pruned search space on the 2 new datasets, i.e., Epinions and Amazon-Sports, and adopt the one with the best test performance as our final choice. Table 6 shows the best searched model architectures.

We then compare the best searched model with popular CF baselines. The goal of this section is not to pursue state-of-the-art performance on the 2 datasets, but to evaluate the quality of the pruned search space, i.e., higher concentration of top-performing models, and how it benefits model design in new recommendation scenarios simply with a random search strategy. Thus, we only include 3 popular GNN-based CF methods as well as MF and NCF as comparisons and do not heavily tune the baselines as well as our searched models. We run all the methods for 10 times with different random seeds, and report the average and the standard deviation of the RMSE performance results in Table 7. The detailed hyperparameter settings and more implementation details can be found in Appendix B.2 and C.2, respectively.

We can observe that the randomly searched models can outperform all the baselines on the 2 datasets, which shows that the pruned design space with high quality can help to efficiently design top-performing models for new recommendation scenarios. Besides, we note that there is not a single baseline model which can consistently beat its competitors, which shows that designing top-performing models for diverse new scenarios is not a trivial task and emphasizes the significance of exploring design space for higher model design efficiency.

Table 7: Performance comparisons with baselines on Epinions and Amazon-Sports. RS-10 denotes the randomly searched model under 10 samples. Lower is better. The best results are bold and the second-best are underlined.

Model	Epinions	Amazon-Sports
MF [16]	0.9945 ± 0.0000	0.9882 ± 0.0007
NCF [11]	1.0070 ± 0.0055	<u>0.9342 ± 0.0008</u>
NGCF [36]	1.1437 ± 0.0240	1.0668 ± 0.0038
LightGCN [10]	<u>0.9926 ± 0.0001</u>	0.9705 ± 0.0003
DGCF [37]	1.6800 ± 0.2272	0.9894 ± 0.0000
RS-10	**0.8729 ± 0.0014**	**0.9327 ± 0.0006**

The insights obtained from design space profiling in Section 4.4 guide us to perform design space pruning for a higher concentration of top-performing models. Compared to the common practice that designing an *individual* model for a specific setting, the novel paradigm of profiling the design space focuses on a *population* of models, and thus stronger robustness and generalization are guaranteed. The evaluation results demonstrate its strong generalization ability and the case study further shows its effectiveness in quickly finding well-performing models. Therefore, we conclude that the novel paradigm of exploring design space paves the way for efficiently designing top-performing GNN methods in different recommendation scenarios.

6 CONCLUSION

In this work, we propose to profile the design space of GNN-based CF methods, which have been widely researched in recent years. By covering existing GNN-based CF methods in a unified framework, a novel design space is developed and a controlled random search is adopted to efficiently and effectively evaluate the influences of different dimensions on recommendation performance. Furthermore, based on the empirical findings, design space pruning is performed by ruling out some design choices which are shown to be sub-optimal in the evaluation. Then empirical studies in different settings demonstrate the high quality and the strong generalization ability of the pruned design space. Finally, as a case study, we show that it can quickly obtain top-performing model architectures on 2 new datasets simply by random searching the pruned design space, compared to popular CF methods. In the future, we will further explore how the proposed design space can benefit more complex recommendation scenarios, e.g., social recommendation [6].

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. U20B2045, 61772082, 61702296, 62002029). We thank all anonymous reviewers for their constructive comments.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[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 *Proceedings of the AAAI Conference on Artificial Intelligence*. 27–34.

[3] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the Conference on Recommender Systems*. 101–109.

[4] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. 2021. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 279–288.

[5] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. 2012. The yahoo! music dataset and kdd-cup'11. In *Proceedings of KDD Cup 2011*. 3–18.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the International Conference on World Wide Web*. 417–426.

[7] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graphnas: Graph neural architecture search with reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1403–1409.

[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1025–1035.

[9] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the International Conference on World Wide Web*. 507–517.

[10] 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 International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International Conference on World Wide Web*. 173–182.

[12] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 230–237.

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

[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

[15] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations*.

[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 426–434.

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

[18] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *Proceedings of International Conference on Machine Learning*. 82–90.

[19] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. 2012. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193* (2012).

[20] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[21] Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of International Conference on Learning Representations*.

[22] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[23] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the International Conference on Web Search and Data Mining*. 287–296.

[24] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.

[25] Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*. 3697–3707.

[26] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. 2019. On network design spaces for visual recognition. In *Proceedings of the International Conference on Computer Vision*. 1882–1890.

[27] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing network design spaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 10428–10436.

[28] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the Conference on Recommender Systems*. 240–248.

[29] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).

[30] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the International Conference on World Wide Web*. 285–295.

[31] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[32] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the Conference on Recommender Systems*. 23–32.

[33] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2820–2828.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations*.

[35] Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. 2021. Explainable automated graph representation learning with hyperparameter importance. In *Proceedings of International Conference on Machine Learning*. 10727–10737.

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[37] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1001–1010.

[38] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2020. Multi-component graph convolutional collaborative filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 6267–6274.

[39] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260* (2020).

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *Proceedings of International Conference on Learning Representations*.

[41] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 974–983.

[42] Jiaxuan You, Rex Ying, and Jure Leskovec. 2020. Design Space for Graph Neural Networks. In *Advances in Neural Information Processing Systems*. 17009–17021.

[43] Hengrui Zhang and Julian McAuley. 2020. Stacked Mixed-Order Graph Convolutional Networks for Collaborative Filtering. In *Proceedings of the International Conference on Data Mining*. 73–81.

[44] Sixiao Zhang, Hongxu Chen, Xiao Ming, Lizhen Cui, Hongzhi Yin, and Guandong Xu. 2021. Where are we in embedding spaces? A Comprehensive Analysis on Network Embedding Approaches for Recommender Systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[45] Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2021. Automated Machine Learning on Graphs: A Survey. *Proceedings of the International Joint Conference on Artificial Intelligence*, 4704–4712.

[46] Huan Zhao, Quanming Yao, James T Kwok, and Dik Lun Lee. 2017. Collaborative filtering with social local models. In *Proceedings of the International Conference on Data Mining*. 645–654.

[47] Huan Zhao, Quanming Yao, and Weiwei Tu. 2021. Search to aggregate neighborhood for graph neural network. In *Proceedings of the International Conference on Data Engineering*. 552–563.

[48] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, et al. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the International Conference on Information and Knowledge Management*.

A DATASETS

- **Yelp** is a business recommendation dataset. 10-core filtering is applied to ensure that each user/item has at least 10 interactions.
- **Amazon** is a large e-commerce dataset introduced in [9], and we take the subsets from 3 categories to form *Amazon-CDs*, *Amazon-Movies* and *Amazon-Beauty* and apply 10-core filtering.
- **MovieLens** is a widely-used movie rating dataset for evaluating recommender systems, and we choose both the 100K and 1M versions in our experiment.
- **YahooMusic** [5] is a music rating dataset, and **Flixster** [13], and **Douban** [23] are both movie rating datasets. For these 3 datasets, we use the preprocessed subsets provided by [25].

For MovieLens-100K/1M, YahooMusic, Flixster, and Douban, we follow the customized split provided together with the dataset; and for the rest of the datasets, we randomly split them into 0.8/0.1/0.1 train/validation/test sets. Testing performance in the best epoch of validation set is reported.

The 9 datasets contain ratings with different scales. For a more fair comparison and fast convergence of models, we adopt standardization to the initial ratings using $R' = \frac{R-\mu_{train}}{\sigma_{train}}$, where μ_{train} and σ_{train} are the mean and standard variation of the rating value in the training set, respectively.

B HYPERPARAMETERS

B.1 Design Dimension Evaluation

In hyperparameter evaluation, $S = 30$ groups of experimental configurations are sampled for each of the 3 selected hyperparameters. The values of these hyperparameters remain the same as the optimal choices in the evaluation result shown in Figure 3, i.e., dropout is fixed as `0`, the number of training epochs is fixed as `200`, and L_2 regularization weight is fixed as `0.0005`. The hidden dimension of GNN layer is set as 64. All models are optimized using Adam [14] optimizer. The base learning rate is set as 0.01 and is scheduled using the cosine annealing method of SGDR [21].

B.2 Case Study

For our randomly searched models, we perform rough hyperparameter tuning; and for the baselines, their hyperparameters are initialized according to the suggestions in the original papers and also roughly tuned by several steps.

For brevity, we will denote some variables. Suppose hidden dimension of (GNN) layer as d_h, number of (GNN) layers as L, L_2 regularization weight as λ, learning rate as lr. For training epochs of baselines, we use early stop mechanism based on the evaluation on validation set.

B.2.1 MF. We set $d_h = 64$, $\lambda = 10^{-2}$ and $lr = 10^{-3}$ for both datasets.

B.2.2 NCF. We set $d_h = 64$, $\lambda = 10^{-2}$ and $lr = 10^{-3}$ for both datasets.

B.2.3 NGCF. We set $d_h = 64$, $L = 3$, $\lambda = 10^{-2}$ and $lr = 10^{-3}$ for both datasets.

B.2.4 LightGCN. We set $d_h = 64$, $L = 3$, $\lambda = 10^{-5}$ and $lr = 10^{-3}$ for both datasets.

B.2.5 DGCF. We set $d_h = 64$, $L = 1$, the number of latent factors as 4, the number of iterations as 2, $\lambda = 10^{-3}$ and $lr = 10^{-3}$ for both datasets.

B.2.6 RS-10. We set the number of training epochs as 160 for Epinions, and $\lambda = 5 \times 10^{-3}$ for Amazon-Sports. Other hyperparameters remain the same as our former setting.

C IMPLEMENTATION DETAILS

C.1 Design Dimension Evaluation

We utilize Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz and GeForce RTX 3090 as the experimental environment. A small number of experimental configurations are skipped because of OOM on a single GPU. Our customized evaluation is performed on the GraphGym platform [42], which supports parallel experiment launch. To make the results convincing as reliable, the averaged results across 3 runs with different random seeds are reported.

C.2 Case Study

The baseline methods are evaluated using the RecBole framework [48]. For fair comparisons, all the methods are optimized with the MSE loss calculated as Equation (7).

D ADDITIONAL RESULTS

In this section, we provide additional experimental results of design space profiling. The violin plot indicates the smoothed distribution of the ranking of each design choice aggregated over the sampled experimental configurations (Figure 8). It provides us with information on how the ranking results distribute across different values.

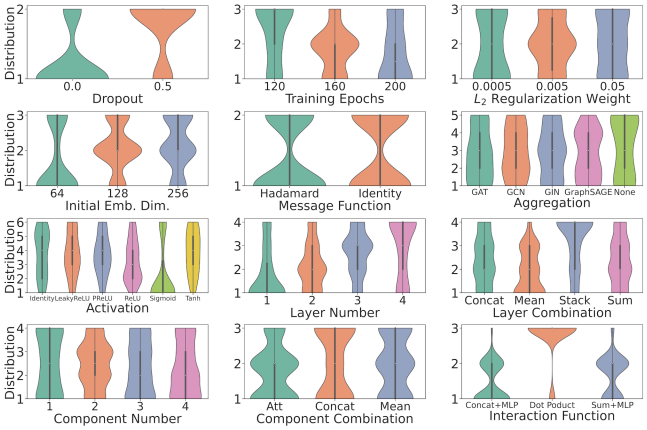


Figure 8: The ranking distribution analysis.

E DISCUSSION ON EDF

Suppose $\mathbf{1}$ as the indicator function, n as the number of sampled models, each with RMSE x_i. The RMSE EDF is given by:

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left[x_i < x\right]. \tag{8}$$

$F(x)$ gives the fraction of models with RMSE less than x.

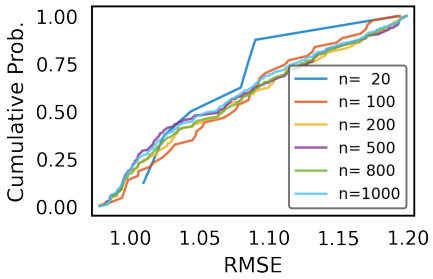


Figure 9: EDF curves w.r.t. varying sample numbers on Amazon-Sports.

The key intuition behind using EDF as the indicator of a design space is that comparing distributions can help to obtain more robust and informative conclusions than comparing the best found models from the two design spaces [27].

To explore the appropriate value for n, we plot the EDF curves of the vanilla design space on the Amazon-Sports dataset, whose statistics are introduced in Table 5, with the sample number n ranging from 20 to 1000 in Figure 9. Since only the performance distribution on the same dataset is comparable, the dataset is fixed when comparing EDF curves in the same illustration. We can observe that the resulting EDF curves are close to each other after n reaching 100, which suggests that 100 samples may be sufficient to give a reasonable estimate of the accurate EDF. Therefore, for the vanilla design space, we will show the EDFs for $n = 100$ sampled models; and for the pruned design space, we set $n = 96$ since it is the total number of the models it contains.