Blurring-Sharpening Process Models for Collaborative Filtering

Jeongwhan Choi
Yonsei University
Seoul, South Korea
jeongwhan.choi@yonsei.ac.kr

Seoyoung Hong
Yonsei University
Seoul, South Korea
seoyoungh@yonsei.ac.kr

Noseong Park
Yonsei University
Seoul, South Korea
noseong@yonsei.ac.kr

Sung-Bae Cho
Yonsei University
Seoul, South Korea
sbcho@yonsei.ac.kr

ABSTRACT

Collaborative filtering is one of the most fundamental topics for recommender systems. Various methods have been proposed for collaborative filtering, ranging from matrix factorization to graph convolutional methods. Being inspired by recent successes of graph filtering-based methods and score-based generative models (SGMs), we present a novel concept of blurring-sharpening process model (BSPM). SGMs and BSPMs share the same processing philosophy that new information can be discovered (e.g., new images are generated in the case of SGMs) while original information is first perturbed and then recovered to its original form. However, SGMs and our BSPMs deal with different types of information, and their optimal perturbation and recovery processes have fundamental discrepancies. Therefore, our BSPMs have different forms from SGMs. In addition, our concept not only theoretically subsumes many existing collaborative filtering models but also outperforms them in terms of Recall and NDCG in the three benchmark datasets, Gowalla, Yelp2018, and Amazon-book. In addition, the processing time of our method is comparable to other fast baselines. Our proposed concept has much potential in the future to be enhanced by designing better blurring (i.e., perturbation) and sharpening (i.e., recovery) processes than what we use in this paper. Our code is available at https://github.com/jeongwhanchoi/BSPM.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

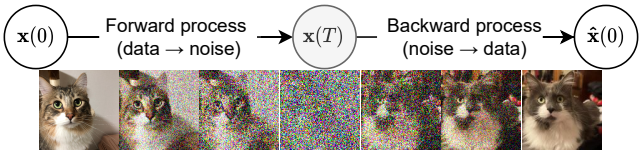
KEYWORDS

Collaborative Filtering, Blurring-Sharpening Process

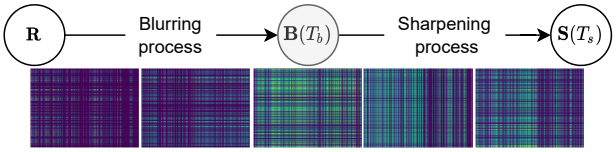
ACM Reference Format:
Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. 2023. Blurring-Sharpening Process Models for Collaborative Filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591645

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591645



(a) Score-based generative models (SGMs) use two stochastic processes, one for the forward perturbation and the other for the backward recovery. Since the recovery process is stochastic, it does not typically converge to the original sample $\boldsymbol{x}(0)$ but to another similar sample. After training, only the recovery process is used to generate fake samples from random noisy vectors $\boldsymbol{x}(T) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$.



(b) Our blurring-sharpening process models (BSPMs) use two deterministic blurring and sharpening processes. Unlike SGMs trained with many images, our BSPMs process only one interaction matrix and therefore, we use the deterministic processes.

Figure 1: The comparison between SGMs and our proposed BSPMs. SGMs, a recently proposed paradigm for deep generative task, outperform generative adversarial networks (GANs), variational autoencoders (VAEs), and many other generative models.

1 INTRODUCTION

Recommender systems are one representative topic of information filtering. These days a non-trivial portion of the revenue of many global information technology (IT) companies is from advertising and recommendation. In this regard, recommender systems are of utmost interest in real-world environments. Among various technologies, collaborative filtering (CF) is one of the most popular approaches of recommender systems, and many CF-based methods have been proposed.

In particular, graph convolution-based CF methods currently show state-of-the-art accuracy [10, 14, 18, 21, 23, 27, 37]. They represent user-item interactions as a bipartite graph and apply the graph convolutional technology. Among various graph convolutional operations, they all use relatively simple linear or low-pass filters. Surprisingly, these approaches now beat other classical and deep learning-based methods.

Table 1: Comparison of existing methods. Our BSPMs not only combine the blurring and the sharpening processes but also interpret them in a continuous time domain.

Model	Blurring	Sharpening
LightGCN	Discrete with heat equation	X
LT-OCF	Continuous with heat equation	X
GF-CF	Discrete with low pass & ideal filters	X
BSPM	Continuous with various filters	Continuous with a filter

In this paper, we propose a novel paradigm of ***B***lurring-***S***harpening ***P***rocess ***M***odel (BSPM) for CF. Our blurring and sharpening processes are formulated as differential equations — we will also show that some of the existing graph convolution-based CF methods are special cases of our model.

Our method is greatly inspired by i) score-based generative models (SGMs [46–48]) which are considered as state-of-the-art methods for deep generative tasks, and ii) GF-CF and its following work [23, 32, 39, 40, 45, 64] which are simple and computationally efficient but show state-of-the-art accuracy. GF-CF does not learn embedding vectors for users/items but directly processes the user-item interaction matrix to derive unknown user-item interactions. Although GF-CF has shown encouraging results, we found that our proposed *perturbation-recovery paradigm*, called BSPM, can significantly outperform it. Similar successes were already made for image generation. For instance, SGMs show the state-of-the-art quality in the domain of image generation. In SGMs, specific types of stochastic differential equations (SDEs) are adopted to describe the forward and the backward processes (cf. Fig. 1 (a)) — the backward process is considered as a generative model.

Our overall model design has a perturbation-recovery architecture, i.e., the blurring process corrupts (or perturbs) original information in the user-item interaction matrix, and the sharpening process tries to recover the original information in conjunction with promising additional information (cf. Fig. 1 (b)). We apply the blurring and the sharpening processes directly to the interaction matrix in a continuous-time manner whereas existing methods, such as GF-CF, apply certain blurring filters to the matrix in a discrete-time manner (cf. Table 1). To our knowledge, we are the first proposing the blurring-sharpening process paradigm for CF.

Therefore, the key in our model is how to define the blurring and the sharpening processes. Both of them are written as ordinary differential equations (ODEs) in our case (cf. Eqs. (8) and (13)). We customize various well-known blurring and sharpening functions proposed in various domains different from CF.

After defining our blurring and sharpening processes, we design two variants of BSPM: BSPM-LM and BSPM-EM. These variants differ from each other in how to connect the blurring and the sharpening processes. We then show that some popular existing methods are special cases of our method.

We conduct experiments with 3 benchmark datasets and 43 baselines. Surprisingly, our method beats all existing popular CF algorithms by large margins. There are no existing methods that are comparable to our method in all datasets.

Moreover, our proposed model can also be properly understood from the perspective of classical graph convolutional processing. Therefore, we emphasize that our proposed model has strong theoretical grounds, and it is not by chance that our model marks the best accuracy. Our contributions can be summarized as follows:

(1) There are two research trends, which inspire us: i) GF-CF and its following work, which are some of the state-of-the-art methods for collaborative filtering, directly process the user-item interaction matrix to reveal unknown user-item interactions without learning embedding vectors, and ii) SGMs adopt the perturbation-recovery paradigm to generate fake images.
(2) We design a perturbation-recovery concept, called ***B***lurring-***S***harpening ***P***rocess ***M***odel (BSPM).
(3) Our BSPMs directly perturb (blur) the user-item interaction matrix, and recover (sharpen) the blurred matrix to derive unknown user-item interactions.
(4) Our method outperforms all existing 43 popular CF methods in the three benchmark datasets.
(5) To our knowledge, we are the first adopting the perturbation-recovery paradigm for CF. Therefore, one can consider that we propose a new paradigm for CF, and we think that it has much potential in the future by discovering better perturbation and recovery processes than ours.

2 PRELIMINARIES & RELATED WORK

In this section, we review related work and preliminary knowledge: collaborative filter (CF), score-based generative models (SGMs), and ordinary differential equations (ODEs).

2.1 Collaborative Filtering

Let $\boldsymbol{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{V}|}$, where $\mathcal{U}$ is a set of users and $\mathcal{V}$ is a set of items, be an interaction matrix. $\boldsymbol{R}_{u,v}$ is 1 *iff* an interaction (u,v) is observed in data, or otherwise 0. We also define the normalized interaction matrix as $\tilde{\boldsymbol{R}} = \boldsymbol{U}^{-\frac{1}{2}}\boldsymbol{R}\boldsymbol{V}^{-\frac{1}{2}}$, where $\boldsymbol{U} = Diag(\boldsymbol{R}\mathbf{1})$, $\boldsymbol{V} = Diag(\mathbf{1}^\intercal\boldsymbol{R})$, $\mathbf{1}$ means a column vector of ones, and $^\intercal$ means transpose. We also define the normalized item-item adjacency matrix as $\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{R}}^\intercal\tilde{\boldsymbol{R}}$.

Matrix Factorization-based Methods. The most common CF paradigm is to learn latent features (also known as embedding vectors) to represent users and items. The dot product of user and item embedding vectors $\boldsymbol{e}_u^T\boldsymbol{e}_i$ approximates user u's rating on item i, which is denoted by r_{ui}. Earlier CF models focused on low-rank matrix factorization (MF) [28], which aims to approximate the interaction matrix $\boldsymbol{R}_{u,v}$. Singular value decomposition (SVD) was initially proposed to learn the feature matrices, followed by many other MF methods [6, 19, 38, 42, 43, 53, 66]. NCF [19] replaces the dot product with a similarity learned with a multi-layer perceptron (MLP). GRMF [42] smoothes MF through adding a graph Laplacian regularizer. HOP-Rec [66] proposes a unified and efficient method that incorporates both MF and graph-based models for CF. ENMF [6] proposes a simple neural MF method without a negative sampling strategy and uses an MSE loss function. MF-CCL [36] proposes a neural MF model trained with a cosine contrastive loss, and shows that it is superior to existing loss functions.

Graph-based Methods. From the perspective of the user-item interaction graph, the individual interaction history is equivalent to the first-order connectivity of the user. Thus, a natural extension is to mine the higher-order connectivity from the user-item graph structure. For example, the second-order connectivity of a user consists of similar users who have co-interacted with the same items. Fortunately, with the development and success of graph convolutional networks (GCNs) for modeling graph structure data in various machine learning areas, it recently became popular to adopt GCNs for CF [8, 10, 18, 37, 45, 51, 54, 59, 67].

GC-MC [54] is the first work using GCNs for recommendations, which is a graph-based auto-encoder framework for explicit matrix completion. PinSage [67] first applies GCNs on web-scale recommender systems and proposes the combination of efficient random walks and GCNs. NGCF [59] then proposes an interaction encoder to capture the collaboration signal among users and items, using non-linear activations and transformation matrices. NIA-GCN [51] explicitly models the relational information between neighbor nodes and exploits the heterogeneous nature of the user-item bipartite graph. Graph-based recommendation models have achieved remarkable results, but their efficiency remains unsatisfactory when confronted with large-scale recommendation scenarios. Therefore, improving the efficiency of graph-based methods while leaving high performance for recommendations has become a popular research question. Inspired by a simplified GCN (SGC) [62], LightGCN [18] outperforms NGCF [59] by removing the non-linear activation and feature transformation to improve both accuracy and efficiency. Its linear graph convolutional layer definition is as follows:

$$\mathbf{E}(l+1) = \tilde{\mathbf{A}}\mathbf{E}(l), \tag{1}$$

where $\mathbf{E}(0) \in \mathbb{R}^{(|\mathcal{U}|\times|\mathcal{V}|)\times D}$ is the learnable initial embedding matrix of users and items, $\mathbf{E}(l)$ denotes the embedding matrix at l-th layer, and $\tilde{\mathbf{A}}$ is the normalized user-item adjacency matrix. LightGCN learns the initial embedding and uses the layer combination. The model prediction is defined as the dot product of the user's and item's final representation $\boldsymbol{e}_u^T \boldsymbol{e}_i$.

Other variants of LightGCN also achieved competitive performance [15, 23, 29, 36, 37, 60, 63]. For example, SGL-ED [63] contrasts different node views that are generated by randomly masking the edge connections on the graph and incorporating the proposed self-supervised loss into LightGCN. DGCF [60] considers user-item relationships at the finer granularity of user intents and generated disentangled user and item representations. UltraGCN [37] proposes a simplified CF that skips infinite layers of message passing for an efficient recommendation, which generalizes multiple standard linkage scores. SimpleX [36] improves the CF methods with the help of an appropriate negative sampling rate and proposed cosine contrastive loss. LinkProp [15] proposes a new linkage score for link prediction on a bipartite graph. MGDCF [23] generalizes LightGCN and APPNP [26] with the Markov process for distance learning. GTN [14] captures the adaptive reliability of the interactions between users and items using the trend filter.

Recently, researchers argued that linear GCN-based models resemble heat equations, which describe the law of thermal diffusive processes, i.e., Newton's Law of Cooling [10, 24, 61]. LT-OCF [10] redesigned LightGCN as a continuous diffusive process and outperforms LightGCN. LT-OCF also learns an optimal layer combination rather than relying on a pre-defined architecture. The heat equation is directly related to low pass filters and smoothness, which is one of the key operations in graph signal processing. GF-CF [45] was proposed from the perspective of the smoothness of graph signals. It is the special case of existing CF methods: the low-rank matrix factorization corresponds to the ideal low-pass filter, and LightGCN with infinitely embedding dimensionality corresponds to a first-order linear filter. Therefore, GF-CF proposed a simple model combining a linear filter and an ideal low-pass filter as follows:

$$\hat{\boldsymbol{R}} = \boldsymbol{R}\big(\tilde{\boldsymbol{P}} + \beta \boldsymbol{V}^{-\frac{1}{2}}\bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^{\top}\boldsymbol{V}^{\frac{1}{2}}\big), \tag{2}$$

where $\hat{\boldsymbol{R}}$ is an inferred interaction matrix, and $\bar{\boldsymbol{U}}$ is the top-k singular vectors of $\tilde{\boldsymbol{R}}$. GF-CF only needs matrix multiplication operations to calculate the recommendation scores thanks to its non-parametric architecture. Both LT-OCF and GF-CF use blurring processes with the heat equation and the low-pass filter, respectively, but no sharpening processes. In Table 1, we compare recent methods.

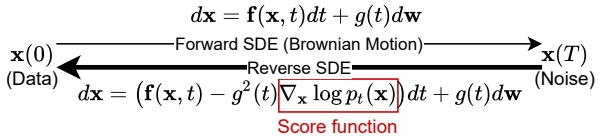


Figure 2: The overall workflow of SGMs, where the score function is approximated by a score network, i.e., $S_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. We note that it means the gradient of the log probability w.r.t. x at time t.

2.2 Score-based Generative Models (SGMs)

Fig. 1 (a) depicts the basic mechanism behind SGMs [46–48]. The forward process is written as the following stochastic differential equation:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \tag{3}$$

where $\mathbf{f}(\mathbf{x}, t) = f(t)\mathbf{x}$, and its reverse SDE (i.e., backward process) is defined as follows:

$$d\mathbf{x} = \big(\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\big)dt + g(t)d\mathbf{w}, \tag{4}$$

where this reverse SDE process is a generative process. Depending on the types of f and g, various sub-types of SGMs are defined.

In order to solve the reverse SDE, we need to know the gradient of the log-probability of the forward SDE (i.e., $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$). We typically train a neural network, called *score network*, to approximate it (cf. Fig. 2). There exists a well-established theory for training the score network. After training the score network with the data collected during the forward process, one can generate fake data from noisy vectors using only the reverse SDE. We formally compare our BSPMs with SGMs as follows since they share similar processing philosophy:

(1) SGMs are for images. One image dataset includes many images and therefore, the entire process should be described in stochastic differential equations (SDEs).

Table 2: The comparison between SGMs and our BSPMs

	SGM	Our proposed BSPM
Type	SDEs	ODEs
Perturbation	Adding noises to images	Blurring interaction matrix
Recovery	Denoising images	Sharpening blurred matrix
Data	Many images	One interaction matrix
What to Learn	Score function	N/A

(2) BSPMs deal with the user-item interaction matrix. One CF dataset includes only one such matrix and therefore, the entire process can be described by deterministic ordinary differential equations (ODEs).
(3) In both models, we expect that new information is discovered during the recovery process. For instance, the denoising process is a generative process in SGMs and in our case, user-specific items are recommended during the sharpening process.
(4) Except that a series of perturbation-recovery processes are used in both models, however, they differ at many detailed points. In BSPMs, most importantly, there does not exist anything to learn since we directly blur and sharpen the interaction matrix $\boldsymbol{R}$.
(5) In Table 2, we summarize key differences.

2.3 Ordinary Differential Equations (ODEs)

The initial value problem (IVP) of ordinary differential equations can be written as follows:

$$\boldsymbol{x}(T) = \boldsymbol{x}(0) + \int_0^T f(\boldsymbol{x}(t))dt, \tag{5}$$

where $\boldsymbol{x}(0)$ is an initial value at time $t = 0$, and $f : \mathbb{R}^{\dim(\boldsymbol{x})} \to \mathbb{R}^{\dim(\boldsymbol{x})}$ is an ODE function describing the time-derivative of $\boldsymbol{x}$, denoted by $\frac{d\boldsymbol{x}(t)}{dt}$[1]. Therefore, integrating the time-derivative of $\boldsymbol{x}$ until $t = T$ returns a solution $\boldsymbol{x}(T)$ at time $t = T$.

f is typically complicated in real-world applications and it is frequently impossible to find an analytical solution of $\boldsymbol{x}(T)$. We then typically use ODE solvers, such as the Euler method, the Runge-Kutta method, the Dormand–Prince (DOPRI) method, and so on [12]. The Euler method [2] is written as follows:

$$\boldsymbol{x}(t+s) = \boldsymbol{x}(t) + \tau \cdot f(\boldsymbol{x}(t)), \tag{6}$$

where τ is a pre-configured step size.

Other ODE solvers use more complicated methods to update $\boldsymbol{x}(t+\tau)$ from $\boldsymbol{x}(t)$. For instance, the fourth-order Runge–Kutta (RK4) method uses the following method:

$$\boldsymbol{x}(t+\tau) = \boldsymbol{x}(t) + \frac{s}{6}\Big(f_1 + 2f_2 + 2f_3 + f_4\Big), \tag{7}$$

where $f_1 = f(\boldsymbol{x}(t))$, $f_2 = f(\boldsymbol{x}(t) + \frac{\tau}{2}f_1)$, $f_3 = f(\boldsymbol{x}(t) + \frac{\tau}{2}f_2)$, and $f_4 = f(\boldsymbol{x}(t) + \tau f_3)$.

[1]In the case of neural ordinary differential equations (NODEs), f is approximated by a neural network, which means the time-derivative of $\boldsymbol{x}$ is learned from data [node]. In this paper, however, f is not a neural network but a blurring/sharpening function.
[2]Note that Eq. (6) is identical to a residual connection when $s = 1$ and therefore, NODEs are a continuous generalization of residual networks.

In order to solve the above integral problem, therefore, we need to iterate one of the fixed-step ODE solvers $\lceil T/\tau \rceil$ times since each iteration updates $\boldsymbol{x}(t)$ to $\boldsymbol{x}(t+\tau)$. However, the DOPRI method is an adaptive solver, which dynamically adjusts the step-size τ depending on estimated potential errors. Therefore, the number of iterations is not deterministic for DOPRI. In general, DOPRI is considered one of the most advanced solvers. These solvers are already implemented on many deep learning platforms, such as PyTorch and TensorFlow. We test all those solvers for our experiments.

3 PROPOSED METHOD

We describe our BSPMs for CF, which consist of a blurring process and a sharpening process. Our method is greatly inspired by the recent successes of SGMs for deep generative tasks. In fact, there already exists a research trend to use generative models for CF due to the similarity in them [4, 5, 7, 50, 56–58] — revealing hidden interactions between users and items means that we generate new interactions.

3.1 Overall Workflow

Our overall workflow is as simple as i) applying a continuous blurring process to the interaction matrix $\boldsymbol{R}$ to derive its blurred matrix $\boldsymbol{B}(T_b)$, and then ii) applying a continuous sharpening process to the blurred matrix to derive its sharpening matrix $\boldsymbol{S}(T_s)$. After these processes, we can recommend items as we will shortly describe.

We also make it clear that in our method, there does not exist anything to train. During the processes, neural networks are not used at all and we do not learn user/item embedding vectors. The blurring and sharpening functions are all hand-crafted functions (without any trainable parameters) in our method. Therefore, our process is surprisingly simple and the overall computation can be done quickly. However, our method outperforms all existing popular methods by non-trivial margins.

Meaning of Blurring. The blurring process is a core of CF. Many graph-based CF methods use graph convolutional filters that correspond to blurring processes [1, 18, 45]. In general, popular items are recommended to users after this process.

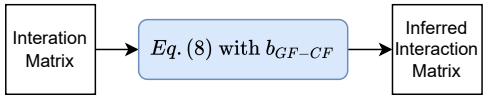
Meaning of Sharpening. The sharpening process is an inverse of the blurring and therefore, it retrieves user-specific items — for general GCNs, similar sharpening processes are used to emphasize differences among node features [3, 9]. As we will show in our experiment section, it actually increases the overall recommendation accuracy while mostly decreasing the degree of recommended items. In other words, less popular items are also recommended to users in conjunction with popular items.

3.2 Blurring Process

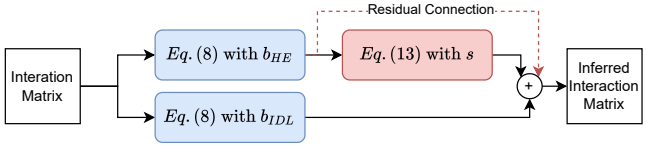
Blurring processes can be written as follows and solved by the ODE solvers we reviewed in Sec. 2.3:

$$\boldsymbol{B}(T_b) = \boldsymbol{B}(0) + \int_0^{T_b} b(\boldsymbol{B}(t))dt, \tag{8}$$

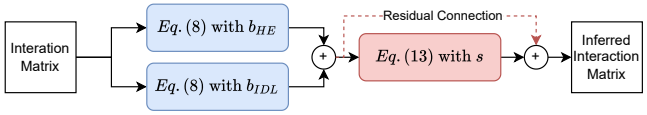
where $b : \mathbb{R}^{\dim(\boldsymbol{B})} \to \mathbb{R}^{\dim(\boldsymbol{B})}$ is a blurring function which approximates $\frac{d\boldsymbol{B}(t)}{dt}$ and $\boldsymbol{B}(0)$ is an interaction matrix $\boldsymbol{R}$ in our setting. Therefore, $\boldsymbol{B}(1)$ means a blurred interaction matrix.



(a) This model corresponds to GF-CF when we use i) the Euler method with $T_b = 1$ and $\tau = 1$ to solve Eq. (8), and ii) a heat capacity of $k = 1$.



(b) The late-merge, denoted BSPM-LM, merges the ideal low-pass filter at the last moment.



(c) The early-merge, denoted BSPM-EM, merges the ideal low-pass filter before the sharpening process begins.

Figure 3: Blue boxes mean blurring processes, and red boxes mean sharpening processes. The red dotted path means the residual connection, which is optional in our method. The residual connection enhances the recommendation accuracy in one dataset in our experiments.

The exact blurring process depends on how we define the function b. In various domains, similar blurring functions have been defined for various purposes. We introduce some key definitions among them that are widely used in various domains.

We first articulate that all the aforementioned notations can be naturally extended after considering the temporal nature of our proposed blurring-sharpening process. For instance, $\boldsymbol{B}(t)$ means a blurred matrix of the original interaction matrix after t following our proposed blurring process *iff* $\boldsymbol{B}(0) = \boldsymbol{R}$.

Heat equation. The heat equation means the Newton's law of cooling, which describes the rate of heat loss in a body. This concept is frequently used in image processing for blurring images. We use the following definition of b:

$$b_{HE}(\boldsymbol{B}(t)) = k\boldsymbol{B}(t)\big(\tilde{\boldsymbol{P}} - \boldsymbol{I}\big), \tag{9}$$

where $k \in \mathbb{R}$ is a coefficient called heat capacity. This is a hyperparameter in our framework. This definition of b has a resemblance to the low-pass filter in the field of graph convolutions.

Ideal low-pass filter. In the field of graph convolutions, the following ideal low-pass filter is frequently used:

$$b_{IDL}(\boldsymbol{B}(t)) = \boldsymbol{B}(t)\big(\boldsymbol{V}^{-\frac{1}{2}}\bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^{\top}\boldsymbol{V}^{\frac{1}{2}} - \boldsymbol{I}\big), \tag{10}$$

where $\bar{\boldsymbol{U}}$ is the top-k singular vectors of $\tilde{\boldsymbol{R}}$.

Our framework has a flexibility that one can also combine them, for instance, as follows:

$$b_{GF-CF}(\boldsymbol{B}(t)) = k\boldsymbol{B}(t)\big(\tilde{\boldsymbol{P}} + \beta\boldsymbol{V}^{-\frac{1}{2}}\bar{\boldsymbol{U}}\bar{\boldsymbol{U}}^{\top}\boldsymbol{V}^{\frac{1}{2}} - \boldsymbol{I}\big), \tag{11}$$

where β is a coefficient to (de-)emphasize the ideal low-pass filter.

In particular, Eq. (8) with b_{GF-CF} reduces to GF-CF [45] which can be written as follows and solved by the ODE solvers we reviewed in Sec. 2.3:

$$\hat{\boldsymbol{R}} = \boldsymbol{B}(0) + \int_0^{T_b} b_{GF-CF}(\boldsymbol{B}(t))dt, \tag{12}$$

where $\boldsymbol{B}(0) = \boldsymbol{R}$, $k = 1$, $T_b = 1$, $\hat{\boldsymbol{R}}$ is an inferred interaction matrix, and we use the Euler method with $\tau = 1$. Therefore, one can consider that GF-CF is a CF method based only on the blurring process (cf. Fig. 3 (a)).

However, our work shows that it is sub-optimal to use only the blurring process. The following sharpening process is able to further enhance the recommendation accuracy. To our knowledge, we are the first proposing the blurring-sharpening process-based CF method.

3.3 Sharpening Process

Sharpening processes can also be written as follows and solved by the ODE solvers we reviewed in Sec. 2.3:

$$\boldsymbol{S}(T_s) = \boldsymbol{S}(0) + \int_0^{T_s} s(\boldsymbol{S}(t))dt, \tag{13}$$

where $s : \mathbb{R}^{\dim(S)} \to \mathbb{R}^{\dim(S)}$ is a sharpening function which approximates $\frac{d\boldsymbol{S}(t)}{dt}$, and $\boldsymbol{S}(1)$ is a sharpened matrix from the input matrix $\boldsymbol{S}(0)$. The sharpening function s can be defined as follows:

$$s(\boldsymbol{S}(t)) = -\boldsymbol{S}(t)\tilde{\boldsymbol{P}}, \tag{14}$$

where the negative sign is added to emphasize the difference from neighbors, i.e., sharpening.

3.4 Blurring-Sharpening Process Model (BSPM)

Using the blurring and sharpening processes, we can define a couple of variations of BSPM. In BSPM-LM in Fig. 3 (b), we use the two blurring processes but apply the sharpening process only to the heat equation-based blurring outcome. We then merge the sharpened interaction matrix with the matrix perturbed by the ideal low-pass filter. This variant can be written as follows and solved by the ODE solvers we reviewed in Sec. 2.3:

$$\begin{aligned}
\boldsymbol{B}_{HE}(T_b) &= \boldsymbol{B}(0) + \int_0^{T_b} b_{HE}(\boldsymbol{B}(t))dt, \\
\boldsymbol{B}_{IDL}(T_b) &= \boldsymbol{B}(0) + \int_0^{T_b} b_{IDL}(\boldsymbol{B}(t))dt, \\
\boldsymbol{S}(T_s) &= \boldsymbol{S}(0) + \int_0^{T_s} s(\boldsymbol{S}(t))dt, \\
\hat{\boldsymbol{R}} &= \begin{cases} \boldsymbol{S}(T_s) + \boldsymbol{B}_{IDL}(T_b) + \boldsymbol{B}_{HE}(T_b), & \text{if residual,} \\ \boldsymbol{S}(T_s) + \boldsymbol{B}_{IDL}(T_b), & \text{otherwise,} \end{cases}
\end{aligned} \tag{15}$$

where $\boldsymbol{B}(0) = \boldsymbol{R}$, $\boldsymbol{S}(0) = \boldsymbol{B}_{HE}(T_b)$, and $\hat{\boldsymbol{R}}$ is an inferred interaction matrix. Adding $\boldsymbol{B}_{HE}(T_b)$ to $\hat{\boldsymbol{R}}$ is optional in our method and is called as *residual connection*.

In BSPM-EM in Fig. 3 (c), we merge the heat equation-based and the ideal low-pass filter-based blurring outcomes as early as before the sharpening process begins. We then apply the sharpening

process. This can be written as follows and solved by the ODE solvers we reviewed in Sec. 2.3:

$$\begin{aligned} \boldsymbol{B}_{HE}(T_b) &= \boldsymbol{B}(0) + \int_0^{T_b} b_{HE}(\boldsymbol{B}(t))dt, \\ \boldsymbol{B}_{IDL}(T_b) &= \boldsymbol{B}(0) + \int_0^{T_b} b_{IDL}(\boldsymbol{B}(t))dt, \\ \boldsymbol{S}(T_s) &= \boldsymbol{S}(0) + \int_0^{T_s} s(\boldsymbol{S}(t))dt, \\ \hat{\boldsymbol{R}} &= \begin{cases} \boldsymbol{S}(T_s) + \boldsymbol{S}(0), & \text{if residual,} \\ \boldsymbol{S}(T_s), & \text{otherwise,} \end{cases} \end{aligned} \tag{16}$$

where $\boldsymbol{B}(0) = \boldsymbol{R}$, and $\boldsymbol{S}(0) = \boldsymbol{B}_{HE}(T_b) + \boldsymbol{B}_{IDL}(T_b)$. Adding $\boldsymbol{S}(0)$ to $\hat{\boldsymbol{R}}$ is a residual connection.

3.5 Direct Inference without Training

We note that our proposed BSPM does not include any training phase, which drastically reduces the total computation time. Since we do not learn any embedding vectors but directly process the interaction matrix $\boldsymbol{R}$, there is no training process. Solving Eq. (15) or (16) is enough to infer unknown user-item interactions, and there exist many ODE solvers which can solve Eqs. (15) and (16) efficiently. As a matter of fact, our method is one of the fastest CF methods. In addition, our method shows the best accuracy in almost all cases for our experiments.

3.6 Comparison with Other Methods

We already showed that GF-CF in Eq. (12) is a special case of BSPM. We will show that other popular CF algorithms are also special cases of our model: i) LightGCN is one of the most influential algorithms for linear graph-based CF. Shen et al. [45] already proved that LightGCNs with infinite-dimensional embeddings are theoretically the same as a one-step heat equation process, which is equivalent to our blurring process with $T_b = 1$, the Euler method with a step size of 1. LightGCN also does not have any sharpening processes. ii) LT-OCF is a continuous generalization of LightGCN. Therefore, our ODE-based blurring process with the heat equation conceptually corresponds to the key idea of LT-OCF although it also has several other contributions. No sharpening processes are used in LT-OCF. iii) It is obvious that GF-CF is equivalent to the blurring process with b_{GF-CF} and the Euler method of $T_b = 1$ to solve it.

4 EXPERIMENTS

In this section, we describe our experimental environments and results. The following software and hardware environments were used for all experiments: Ubuntu 18.04 LTS, Python 3.6.6, PyTorch 1.9.0, Numpy 1.18, Scipy 1.5, sparsesvd 0.2.2, torchdiffeq 0.2.2, CUDA 11.4, NVIDIA Driver 470.42, i9 CPU, and RTX A6000.

4.1 Experimental Environments

4.1.1 Datasets and Baselines. In our experiments, we use the three benchmark datasets that are the most frequently used in the literature: Gowalla, Yelp2018, and Amazon-book [8, 18, 59]. We summarize the dataset statistics in Table 3. Fig. 4 shows the long tail characteristic of the datasets. We compare our proposed BSPM with the following baseline models of different groups:

Table 3: Statistics of datasets

Dataset	#Users	#Items	#Interactions	Density
Gowalla	29,858	40,981	1,027,370	0.084%
Yelp2018	31,668	38,048	1,561,406	0.130%
Amazon-book	52,643	91,599	2,984,108	0.062%

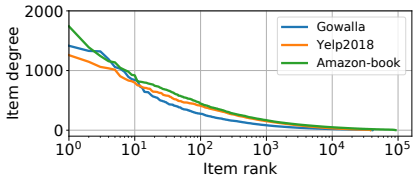


Figure 4: The long-tail characteristic in all datasets.

(1) In the first group of baselines, we consider popular MF-based methods and its variants: MF-BPR [43], Neu-MF [19], HOP-Rec [66], GRMF [42], ENMF [6], and MF-CCL [36].
(2) The second group includes autoencoder-based methods for CF: Mult-VAE [30], Macrid-VAE [35], and EASE^R [49].
(3) The third group includes popular network embedding methods: DeepWalk [41], LINE [52], Node2Vec [17], and Item2Vec [2].
(4) These three models are based on various deep learning paradigms: YoutubeNet [11] is an MLP-based method, CML [22] is a metric learning-based method, and CMN [13] is a memory network-based model.
(5) The fifth group includes general GCN methods: GAT [55], JKNet [65], APPNP [26], DisenGCN [34], and DropEdge [44].
(6) The last group includes GCN-based CF methods: GC-MC [54], NGCF [59], LR-GCCF [8], LightGCN [18], NIA-GCN [51], DeosGCF [33], IMP-GCN [31], SGL-ED [63], HMLET [27], DGCF [60], IA-GCN [68], BUIR$_{\text{NB}}$ [29], UltraGCN [37], SimpleX [36], LT-OCF [10], GF-CF [45], LinkProp [15], MGDCF [23], and GTN [14].

4.1.2 Evaluation Metrics and Hyperparameters. We adopt the two widely used ranking metrics: Recall@20 and NDCG@20 [25]. All items that do not have any interactions with a user are recommendation candidates for the user. To keep the comparison fair with previous studies, we use the same datasets and the same train/test splits.

For other baselines, we use the recommended hyperparameters and for our method, we test the following hyperparameters:

- For solving the integral problems of the blurring/sharpening processes, we consider the following ODE solvers: the Euler method, RK4, and DOPRI. However, we found that RK4 and DOPRI produce almost the same results in our preliminary experiments so we test only the Euler method and RK4.
- For the blurring process, the number of steps $\frac{T_b}{\tau}$ for solvers is in {1,2,3,4}, and the terminal time T_b is set to 1 to 5.
- For the sharpening process, the number of steps $\frac{T_s}{\tau}$ is in {1,2,3,4}, and the terminal time T_s is set to 1 to 5.
- The size of β is in $\{0.0, 0.1, \cdots, 1.0\}$.
- The heat capacity k is in $\{0.1, \cdots, 1.0\}$.

Among the test configurations, the best configuration set in each data is as follows: In Gowalla, $\frac{T_b}{\tau} = 1$, $T_b = 1$, $\frac{T_s}{\tau} = 1$, $T_s = 2.5$, $k = 1.0$, and $\beta = 0.2$. In Yelp2018, $\frac{T_b}{\tau} = 1$, $T_b = 1$, $\frac{T_s}{\tau} = 1$, $T_s = 1.2$, $k = 1.0$, and $\beta = 0.3$. In Amazon-book, $\frac{T_b}{\tau} = 1$, $T_b = 1$, $\frac{T_s}{\tau} = 2$, $T_s = 2.2$, $k = 1.0$, and $\beta = 0$. In general, it is the best to use the Euler method for the blurring process and RK4 for the sharpening process. However, for Yelp2018, it is the best to use the Euler method for sharpening.

4.2 Experimental Results

In Table 4, we summarize the overall accuracy in terms of Recall@20 and NDCG@20. Our specific choices of baselines cover almost all representative CF methods, and the three datasets are widely used in the literature. As reported, our method clearly marks the best accuracy in all cases. In particular, BSPM-LM is the best method and performs better than the LinkProp-Multi by 3.74% on NDCG@20 for Amazon-books. BSPM-EM is the best method for Yelp2018 and Gowalla. In many cases, BSPM-LM and BSPM-EM mark the best and the second-best methods, respectively, and their differences are not significant.

Among the tested baselines, LinkProp-Multi, SimpleX, and GF-CF work well in some cases. However, only LinkProp-Multi is comparable to our method for Gowalla and Amazon-book — however, the accuracy gap between our method and LinkProp-Multi is still non-trivial. For Yelp2018, SimpleX, GF-CF, and MGDCF show good scores. For Amazon-book, GF-CF, LinkProp-Multi, and EASE^{R} show high performance. However, no existing methods are comparable to our proposed method in all datasets. Therefore, we consider that our proposed concept of BSPM opens a new era of CF.

LightGCN is worse than recent methods such as LT-OCF and GF-CF, but its value is that it is the first method showing that simple linear convolutions work better than non-linear ones. Being inspired by it, many methods have been proposed, including ours. Our blurring and sharpening processes are all linear operations. One can adopt non-linear sharpening processes, but in general, linear operations show reliable recommendations.

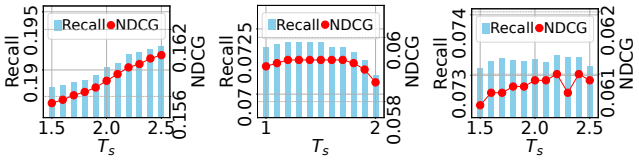
4.3 Ablation and Sensitivity Studies

We report some selected key sensitivity and ablation study results. It is the case that our model is not significantly sensitive to a hyperparameter if not reported in this subsection. In general, our model is sensitive to the hyperparameters of the sharpening process and we focus on them.

4.3.1 Sensitivity on T_s. By varying the terminal integral time T_s of the sharpening process, we investigate how the model accuracy changes in Fig. 5. For Gowalla and Amazon-book, T_s around 2.4 produces the best outcomes. After a certain point, however, the model accuracy drastically decreases in all datasets. It is obvious that applying the sharpening too much (i.e., T_s is too large) is not helpful in the perspective of CF since the sharpening process emphasizes user-specific information (rather than collaborative information). In general, the blurring process can be considered as a collaborative step, where a user's interactions with items are mixed with its neighbors.

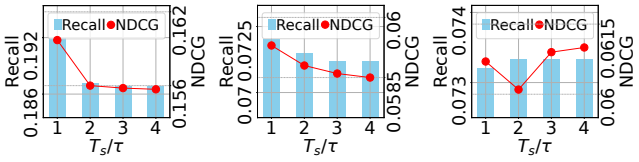
Table 4: Overall performance comparison. *Relative improvement* stands for the improvement of BSPM against the second-best baseline.

Model	Gowalla		Yelp2018		Amazon-book	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF-BPR	0.1291	0.1109	0.0433	0.0354	0.0250	0.0196
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-Norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
NeuMF	0.1399	0.1212	0.0451	0.0363	0.0258	0.0200
HOP-Rec	0.1399	0.1214	0.0517	0.0428	0.0309	0.0232
ENMF	0.1523	0.1315	0.0624	0.0515	0.0359	0.0281
MF-CCL	0.1837	0.1493	0.0698	0.0572	0.0559	0.0447
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
Macrid-VAE	0.1618	0.1202	0.0612	0.0495	0.0383	0.0295
EASE^{R}	0.1765	0.1467	0.0657	0.0552	0.0710	0.0567
YouTubeNet	0.1754	0.1473	0.0686	0.0567	0.0502	0.0388
CMN	0.1405	0.1221	0.0475	0.0369	0.0267	0.0218
CML	0.1670	0.1292	0.0622	0.0536	0.0522	0.0428
DeepWalk	0.1034	0.0740	0.0476	0.0378	0.0346	0.0264
LINE	0.1335	0.1056	0.0549	0.0446	0.0410	0.0318
Node2Vec	0.1019	0.0709	0.0452	0.0350	0.0402	0.0309
Item2Vec	0.1325	0.1057	0.0503	0.0411	0.0326	0.0251
GAT	0.1401	0.1236	0.0543	0.0431	0.0326	0.0235
JKNet	0.1622	0.1391	0.0608	0.0502	0.0268	0.0343
DropEdge	0.1627	0.1394	0.0614	0.0506	0.0342	0.0270
APPNP	0.1708	0.1462	0.0635	0.0521	0.0384	0.0299
DisenGCN	0.1356	0.1174	0.0558	0.0454	0.0329	0.0254
GC-MC	0.1395	0.1204	0.0462	0.0379	0.0288	0.0224
PinSage	0.1380	0.1196	0.0471	0.0393	0.0282	0.0219
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
NIA-GCN	0.1359	0.1106	0.0599	0.0491	0.0369	0.0287
LR-GCCF	0.1701	0.1452	0.0604	0.0498	0.0375	0.0296
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315
SGL-ED	0.1835	0.1539	0.0675	0.0555	0.0478	0.0379
DeosGCF	0.1784	0.1477	0.0626	0.0504	0.0410	0.0316
IMP-GCN	0.1845	0.1567	0.0653	0.0531	0.0460	0.0357
BUIR_{NB}	0.1575	0.1301	0.0647	0.0526	0.0439	0.0346
DGCF	0.1842	0.1561	0.0654	0.0534	0.0422	0.0324
IA-GCN	0.1839	0.1562	0.0659	0.0537	0.0472	0.0373
UltraGCN	0.1862	0.1580	0.0683	0.0561	0.0681	0.0556
SimpleX	0.1872	0.1557	0.0701	0.0575	0.0583	0.0468
LT-OCF	0.1875	0.1574	0.0671	0.0549	0.0442	0.0341
GF-CF	0.1849	0.1518	0.0697	0.0571	0.0710	0.0584
HMLET	0.1874	0.1589	0.0675	0.0557	0.0482	0.0371
LinkProp	0.1814	0.1477	0.0676	0.0559	0.0684	0.0559
LinkProp-Multi	0.1908	0.1573	0.0690	0.0571	0.0721	0.0588
MGDCF	0.1864	0.1589	0.0696	0.0572	0.0490	0.0378
GTN	0.1870	0.1588	0.0679	0.0554	0.0450	0.0346
Only Blurring (HE)	0.1682	0.1331	0.0684	0.0565	0.0710	0.0584
Only Blurring (IDL)	0.1776	0.1489	0.0668	0.0549	0.0395	0.0316
Only Blurring (GF-CF)	0.1854	0.1518	0.0701	0.0575	0.0710	0.0584
BSPM-LM	0.1901	0.1570	0.0713	0.0584	**0.0733**	**0.0610**
BSPM-EM	**0.1920**	**0.1597**	**0.0720**	**0.0593**	**0.0733**	0.0609
Relative Improvement	0.63%	0.50%	2.71%	3.13%	1.66%	3.74%



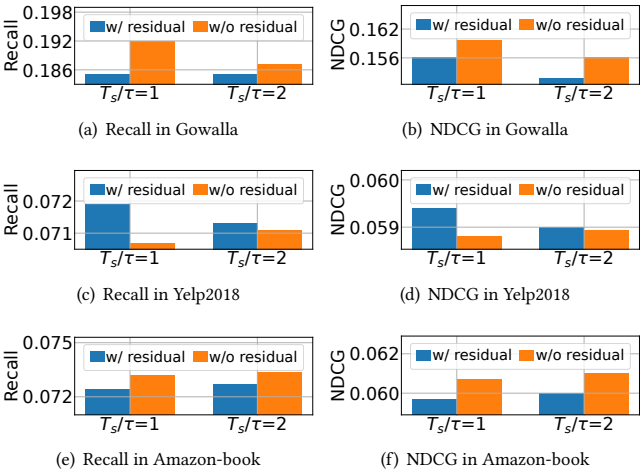
(a) Gowalla (b) Yelp2018 (c) Amazon-book

Figure 5: Performance by varying T_s



(a) Gowalla (b) Yelp2018 (c) Amazon-book

Figure 6: Performance by varying $\frac{T_s}{\tau}$



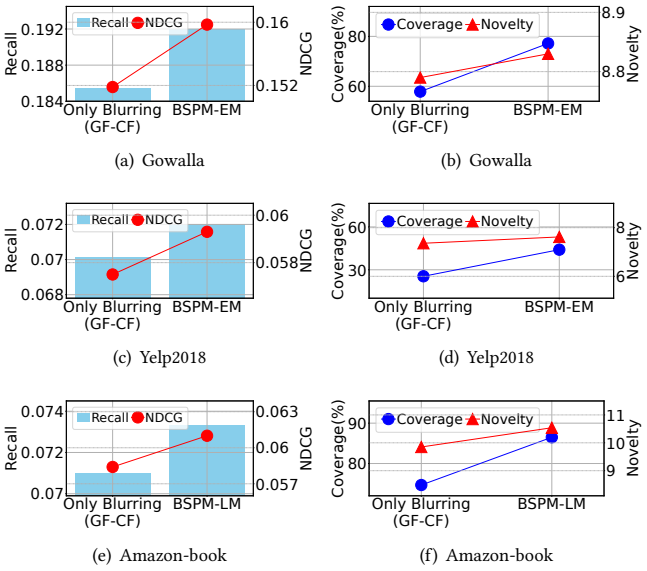
(a) Recall in Gowalla (b) NDCG in Gowalla
(c) Recall in Yelp2018 (d) NDCG in Yelp2018
(e) Recall in Amazon-book (f) NDCG in Amazon-book

Figure 7: Performance by the residual connection

4.3.2 Sensitivity on $\frac{T_s}{\tau}$. By varying the number of steps $\frac{T_s}{\tau}$ for ODE solvers, we test our model. Fig 6 shows that we do not need to use many steps in solving the integral problem of the sharpening process, i.e., Eq. (13), which makes the overall runtime short. In most cases, it shows the best outcomes when $\frac{T_s}{\tau}$ is set to 1 or 2, i.e., τ is set to T_s or $\frac{T_s}{2}$.

4.3.3 Ablation study. As ablation study models, we test the models with only a blurring process, denoted 'Only Blurring (HE)', 'Only Blurring (IDL)', and 'Only Blurring (GF-CF)' in Table 4, depending on the used blurring function type. As shown, they do not show reliable performance in comparison with our main model.

We also test whether the residual connection is helpful in each dataset. As reported in Fig 7, it increases the accuracy in Yelp2018. For other datasets, it is best not to use the residual connection.



(a) Gowalla (b) Gowalla
(c) Yelp2018 (d) Yelp2018
(e) Amazon-book (f) Amazon-book

Figure 8: Accuracy and beyond-accuracy metrics comparison of Only Blurring (GF-CF) and BSPM

4.4 Efficacy of the Sharpening Process

As mentioned earlier, we are the first proposing to use the sharpening process for CF. Therefore, we conduct in-depth analyses on how the sharpening process contributes. Other metrics for CF to measure the quality of recommendation, called *beyond-accuracy metrics*. For these analyses, we use the following beyond-accuracy metrics: novelty [69] and item coverage [20]. The item coverage refers to the extent of items a recommender system can predict. A novel item for a user is one the user has little or no knowledge about it [16]. The novelty measures the unexpectedness of recommended items relative to their global popularity. Using these metrics provides a broader picture of the contribution by the sharpening process. The results are presented in Figs. 8 (b), (d), (f), the blurring-sharpening process has higher novelty and coverage scores than the blurring process in all datasets.

It's worth mentioning that coverage increased by 73.20% on Yelp2018, and the sharpening process improves the ability of the method to recommend long-tail items that users purchase relatively infrequently. In the case of novelty, it can be seen that the metric is improved by 7.01% compared to that using only the blurring process in Amazon-book. In Figs. 8 (a), (c), (e), we can see that the sharpening process is able to enhance Recall and NDCG.

4.5 Runtime Analyses

We also report our pre-processing and inference time in Tables 5 and 6. Since our method does not include any training step, it is much faster than other methods — however, our method requires a pre-processing step to calculate $\tilde{\boldsymbol{R}}$, $\tilde{\boldsymbol{P}}$, and so on. Among various baselines, LightGCN is one of the simplest and most influential graph convolutional methods, but its training time is several orders of magnitude worse than our pre-processing time. GF-CF also does

Table 5: The training time of LightGCN, and the pre-processing time of our method. Since our method does not have any training step, we compare our pre-processing time with LightGCN's training time. GF-CF also requires the same types of pre-processing and therefore has the same pre-processing time as ours.

Model	Gowalla	Yelp2018	Amazon-book
Training of LightGCN	1.0×10^4s	1.5×10^4s	9.7×10^4s
Training of LT-OCF	3.6×10^4s	4.8×10^4s	2.3×10^5s
Pre-processing of BSPM	35.4s	42.3s	101.6s

Table 6: The inference time of BSPM and GF-CF with a mini-batch size of 2048 users

Model	Gowalla	Yelp2018	Amazon-book
GF-CF	9.8s	10.6s	40.1s
BSPM-EM (Euler)	10.5s	11.0s	53.0s
BSPM-EM (RK4)	17.9s	19.8s	125.5s
BSPM-LM (Euler)	10.6s	11.2s	59.3s
BSPM-LM (RK4)	17.6s	19.6s	127.2s

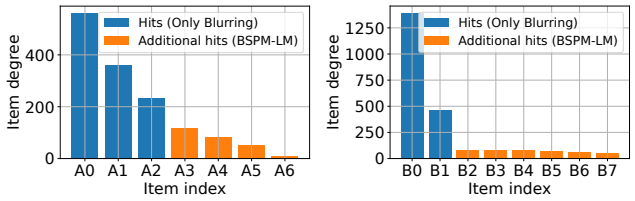
not have any training step and its pre-processing time is the same as our method. However, our model is more complicated than GF-CF (cf. Eq. (12) for GF-CF vs. Eq. (15) or (16) for our method) and has longer inference times. However, current official ODE solver implementations on PyTorch do not support sparse matrices. Our method will become much faster with sparse matrix computations.

4.6 Case Studies

We compare BSPM-EM and BSPM-LM with and without the sharpening process in terms of Hits@20 to see how the sharpening process changes recommended items. After the sharpening process, more items are accurately recommended for all datasets, including Gowalla, Yelp2018, and Amazon-book. Fig. 9 shows case studies for several users on Amazon-book. After the sharpening process, a few more items are accurately recommended for users A and B.

Interestingly, those additional hits, highlighted in orange in Fig. 9, after the sharpening process has relatively low node degrees. In Fig. 9 (b), the recommended items' degrees are high when only the blurring process is performed. In contrast, the degrees of the extra hits after the sharpening process are as low as less than 85, i.e., those additional hits are not popular items but specific to the user. On Amazon-book, the average node degree of hits is 48.20 when only the blurring process is used, but it drops to 33.10 when the sharpening process is added. Fig. 10 shows that the ratio of hits with low item degrees increases after the sharpening process. These results suggest that the model with the sharpening process accurately recommends less popular but user-specific items.

The long-tail problem, i.e., how to recommend more user-specific items, is a major challenge in the recommender system. Benchmark data also has long-tail characteristics, as shown in Fig. 4, making it difficult to recommend user-specific items. Our case study results show that the sharpening process allows for user-specific item recommendations while also improving accuracy.



(a) User A (b) User B

Figure 9: Node degrees of hits (correctly recommended items) by BSPM-LM on Amazon-book. Blue denotes hits when the sharpening process is skipped. Orange means additional hits after the sharpening process is performed.

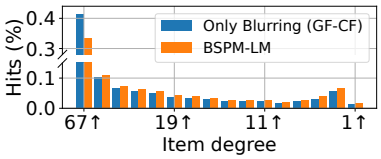


Figure 10: Histogram of Amazon-book's node degrees of correctly recommended items (i.e., hits). In the Y-axis, the number of hits for each node degree bin is divided by the total number of hits.

5 CONCLUSION & FUTURE WORK

We presented a novel paradigm of blurring-sharpening process model (BSPM) for CF. Our work is greatly inspired by the recent two research breakthroughs: i) graph filtering-based methods, e.g., GF-CF, and ii) SGMs for generating fake images. As in SGMs, we adopt the perturbation-recovery paradigm to discover new information. As in GF-CF, we do not learn embedding vectors but directly process the interaction matrix. After defining our BSPM paradigm, we also design a couple of variants to enhance the recommendation accuracy further. In addition, our BSPM is one of the fastest methods ever designed for CF since it does not include any training phase but directly infers unknown user-item interactions. In our experiments with 43 baselines and 3 benchmark datasets, our method marks the best accuracy by large margins. Since our method is not only the most accurate but also one of the fastest methods, it has a significant impact on real-world CF applications.

In the future, we hope that it can be further improved by adopting better blurring and sharpening processes since we have focused on designing the overall architecture by customizing popular filters. In particular, we think that it is promising to learn optimal blurring and sharpening processes from data.

ACKNOWLEDGMENTS

Noseong Park is the corresponding author. This work was supported by an IITP grant funded by the Korean government (MSIT) (No.2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University)) and an ETRI grant funded by the Korean government (23ZS1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System).

REFERENCES

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*.

[2] Oren Barkan and Noam Koenigstein. 2016. ITEM2VEC: Neural item embedding for collaborative filtering. *IEEE 26th International Workshop on Machine Learning for Signal Processing* (2016), 1–6.

[3] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. In *AAAI*.

[4] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating Augmentation with Generative Adversarial Networks towards Accurate Collaborative Filtering. In *TheWebConf (former WWW)*.

[5] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework Based on Generative Adversarial Networks. In *CIKM*.

[6] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Trans. Inf. Syst.* 38, 2, Article 14 (2020), 28 pages.

[7] Honglong Chen, Shuai Wang, Nan Jiang, Zhe Li, Na Yan, and Leyi Shi. 2021. Trust-aware generative adversarial network with recurrent neural network for recommender systems. *International Journal of Intelligent Systems* 36, 2 (2021), 778–795.

[8] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*.

[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[10] Jeongwhan Choi, Jinsung Jeon, and Noseong Park. 2021. LT-OCF: Learnable-Time ODE-based Collaborative Filtering. In *CIKM*.

[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[12] J.R. Dormand and P.J. Prince. 1980. A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.* 6, 1 (1980), 19 – 26.

[13] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR*.

[14] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph Trend Filtering Networks for Recommendation. In *SIGIR*. 112–121.

[15] Hao-Ming Fu, Patrick Poirson, Kwot Sin Lee, and Chen Wang. 2022. Revisiting Neighborhood-based Link Prediction for Collaborative Filtering. In *TheWebConf (former WWW) Workshop on Geometrical and Topological Representation Learning*.

[16] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *RecSys*.

[17] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *KDD*. 855–864.

[18] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-seng Chua. 2017. Neural Collaborative Filtering. In *TheWebConf (former WWW)*.

[20] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53.

[21] Seoyoung Hong, Minju Jo, Seungji Kook, Jaeeun Jung, Hyowon Wi, Noseong Park, and Sung-Bae Cho. 2022. TimeKit: A Time-series Forecasting-based Upgrade Kit for Collaborative Filtering. In *IEEE BigData*.

[22] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *TheWebConf (former WWW)*. 193–201.

[23] Jun Hu, Shengsheng Qian, Quan Fang, and Changsheng Xu. 2022. MGDCF: Distance Learning via Markov Graph Diffusion for Neural Collaborative Filtering. *arXiv preprint arXiv: Arxiv-2204.02338* (2022).

[24] Jeehyun Hwang, Jeongwhan Choi, Hwangyong Choi, Kookjin Lee, Dongeun Lee, and Noseong Park. 2021. Climate Modeling with Neural Diffusion Equations. In *ICDM*. 230–239.

[25] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[26] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

[27] Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park, and Sang-Wook Kim. 2022. Linear, or Non-Linear, That is the Question!. In *WSDM*. 517–525.

[28] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[29] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. In *SIGIR*. 317–326.

[30] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *TheWebConf (former WWW)*.

[31] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-Aware Message-Passing GCN for Recommendation. In *TheWebConf (former WWW)*. 1296–1305.

[32] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Parameter-free Dynamic Graph Embedding for Link Prediction. In *NeurIPS*.

[33] Zhiwei Liu, Lin Meng, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2020. Deoscillated Graph Collaborative Filtering. *arXiv preprint arXiv:2011.02100* (2020).

[34] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *ICML*, Vol. 97. 4212–4221.

[35] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *NeurIPS*, Vol. 32.

[36] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In *CIKM*. 1243–1252.

[37] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *CIKM*.

[38] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic Matrix Factorization. In *NeurIPS*, Vol. 20.

[39] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is More: Reweighting Important Spectral Graph Features for Recommendation. In *SIGIR*. 1273–1282.

[40] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation. In *CIKM*. 1625–1634.

[41] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*. 701–710.

[42] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative Filtering with Graph Information: Consistency and Scalable Methods. In *NeurIPS*.

[43] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*.

[44] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

[45] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B. Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *CIKM*.

[46] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. 2021. Maximum Likelihood Training of Score-Based Diffusion Models. In *NeurIPS*.

[47] Yang Song and Stefano Ermon. 2020. Improved Techniques for Training Score-Based Generative Models. In *NeurIPS*.

[48] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*.

[49] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *TheWebConf (former WWW)*. 3251–3257.

[50] Changfeng Sun, Han Liu, Meng Liu, Zhaochun Ren, Tian Gan, and Liqiang Nie. 2020. *LARA: Attribute-to-Feature Adversarial Learning for New-Item Recommendation.*

[51] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor Interaction Aware Graph Convolution Networks for Recommendation. In *SIGIR*.

[52] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-Scale Information Network Embedding. In *TheWebConf (former WWW)*. 1067–1077.

[53] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In *TheWebConf (former WWW)*.

[54] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. In *KDD*.

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[56] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. In *AAAI*.

[57] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*.

[58] Qinyong Wang, Hongzhi Yin, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2019. Enhancing Collaborative Filtering with Generative Augmentation. In *KDD*.

[59] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.

[60] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *SIGIR*.
[61] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In *NeurIPS*.
[62] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*.
[63] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *SIGIR*. 726–735.
[64] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast Incremental Recommendation with Graph Signal Processing. In *TheWebConf (former WWW)*. 2360–2369.
[65] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*. 5453–5462.
[66] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*.
[67] 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 *KDD*.
[68] Yinan Zhang, Pei Wang, Xiwei Zhao, Hao Qi, Jie He, Junsheng Jin, Changping Peng, Zhangang Lin, and Jingping Shao. 2022. IA-GCN: Interactive Graph Convolutional Network for Recommendation. *arXiv preprint arXiv: Arxiv-2204.03827* (2022).
[69] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4511–4515.