

Towards Effective Top-N Hamming Search via Bipartite Graph Contrastive Hashing

Yankai Chen, Yixiang Fang, Yifei Zhang, Chenhao Ma, Yang Hong, and Irwin King, *Fellow, IEEE*.

Abstract—Searching on bipartite graphs serves as a fundamental task for various real-world applications, such as recommendation systems, database retrieval, and document querying. Conventional approaches rely on similarity matching in continuous Euclidean space of vectorized node embeddings. To handle intensive similarity computation efficiently, hashing techniques for graph-structured data have emerged as a prominent research direction. However, despite the retrieval efficiency in Hamming space, previous studies have encountered *catastrophic performance decay*. To address this challenge, we investigate the problem of hashing with Graph Convolutional Network for effective Top-N search. Our findings indicate the learning effectiveness of incorporating hashing techniques within the exploration of bipartite graph reception fields, as opposed to simply treating hashing as post-processing to output embeddings. To further enhance the model performance, we advance upon these findings and propose **Bipartite Graph Contrastive Hashing (BGCH+)**. BGCH+ introduces a novel dual augmentation approach to both *intermediate information* and *hash code outputs* in the latent feature spaces, thereby producing more expressive and robust hash codes within a dual self-supervised learning paradigm. Comprehensive empirical analyses on six real-world benchmarks validate the effectiveness of our dual feature contrastive learning in boosting the performance of BGCH+ compared to existing approaches.

Index Terms—Graph Convolutional Hashing, Hamming Space Search, Self-supervised Learning, Contrastive Learning.

arXiv:2408.09239v1 [cs.LG] 17 Aug 2024

1 INTRODUCTION

Bipartite graphs are ubiquitous in the real world for the ease of modeling various Web applications, e.g., as shown in Figure 1(a), user-product recommendation [1], [2], and online query-document matching [3]. The fundamental task of *Top-N search* involves selecting the best-matched graph nodes for a given query node, enabling effective information filtering. Machine learning advancements have popularized the use of vectorized representations (*a.k.a.* embeddings) for similarity matching [4], [5], with *Graph Convolutional Networks* (GCNs) standing out for their remarkable performance in capturing high-order connection information and enriching node embeddings [6], [7]. In addition to embedding informativeness, addressing computation latency and memory overhead is crucial for practical application deployment. Recently, *learning to hash* [8], [9] recently provides an alternative option to graph-based models for optimizing the model scalability. Generally, it learns to convert continuous values into the finite binarized hash codes. In lieu of using *full-precision*¹ embeddings, this approach offers space reduction and computation acceleration for Top-N object matching and retrieval in the Hamming space, providing scalability amidst the context of explosive data growth.

- Yankai Chen, Yifei Zhang, Irwin King are with Department of Computer Science and Engineering, The Chinese University of Hong Kong. E-mail: {ykchen,yfzhang,king}@cse.cuhk.edu.hk.
- Yang Hong is with Faculty of Information Science and Engineering, Ocean University of China. E-mail: hongyang@stu.ouc.edu.cn.
- Yixiang Fang, Chenhao Ma are with School of Data Science, The Chinese University of Hong Kong, Shenzhen. E-mail: {fangyixiang,machenhao}@cuhk.edu.cn

1. The term “full-precision” generally refers to single-precision and double-precision. And we use float32 by default throughout this work for illustration.

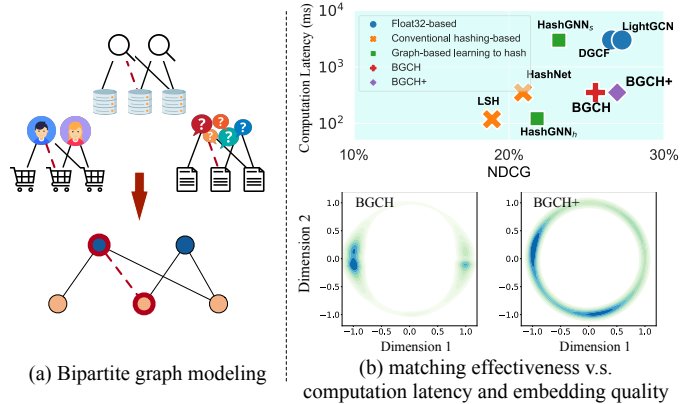


Fig. 1: Bipartite graph modeling and overall model performance in terms of evaluation metrics and embedding distribution visualization (best view in color).

Despite the promising advantages of bridging GCNs and learning to hash, simply stacking these two techniques is trivial and thus falls short of performance satisfaction. Firstly, compared to continuous embeddings, hash codes with the same vector dimension are naturally *less expressive* with finite encoding permutation in Hamming space (e.g., 2^d if the dimension is d). Consequently, this not only leads to a *coarse-grained information encoding* of the graph nodes, but further derives *inaccurate estimation of the pairwise node similarity*. Therefore, the model exhibits a conspicuous performance decay in Top-N matching and ranking. Secondly, for $O(1)$ complexity encoding, $\text{sign}(\cdot)$ function is usually adopted in recent work [10], [11], [12], [13]. Despite the simplicity, optimizing neural networks with $\text{sign}(\cdot)$ is not easy, as $\text{sign}(\cdot)$ is not differentiable at 0 and its derivatives

are 0 anywhere else. Previous models usually use *visually similar* but not necessarily *theoretically relevant* functions, e.g., $\tanh(\cdot)$, for gradient estimation. This may lead to inconsistent optimization directions between forward and backward propagation. Their associated loss landscapes are usually steep and bumping [14], which further increases the difficulty in optimization. These factors jointly lead to an intractable model training process.

To tackle these aforementioned challenges, we have progressively studied the problem of learning to hash with GCN on bipartite graphs in [15]. We identify the critical effect of mining the high-order correlation knowledge in bipartite graph exploration and hashing. In response to this, we have developed an effective learning framework namely BGCH (short for **B**ipartite **G**raph **C**onvolutional **H**ashing) [15]. Generally, the primary objective of BGCH is to enhance the expressivity of the learned hash codes while ensuring an accordant and robust model optimization flow. By leveraging BGCH, the empirical results demonstrate notable performance superiority and competitiveness when compared to both hashing-based and full-precision-based counterparts, respectively. Therefore, BGCH strikes a delicate balance between matching accuracy and computational efficiency, making it a desirable solution for real-world scenarios with limited computation resources.

However, there may still exist two major inadequacy of BGCH [15]. Firstly, due to the discreteness of hash codes, the learning of BGCH [15] in generating informative hash codes is characterized by unpleasant inefficiency, as it typically necessitates substantial training iterations for convergence. Secondly, BGCH currently focuses on alleviating the information loss during the process of graph convolutional hashing, but ignores another fundamental point in learning quality of graph information extraction, especially for sparse graphs. As we include six real-world datasets, their graph densities vary from 0.0419, 0.00084, 0.00267, to 0.0013, 0.00062, and 0.02210. Since the graph convolutions learn and extract the topological information, these sparse graphs may however provide limited structural knowledge as the supervision signal of model learning, resulting in a less effective subsequent graph hashing. To overcome these inadequacies, we seek to absorb the self-supervised learning capability by introducing additional regularization signals to BGCH. Notably, we expect to empower BGCH with the contrastive learning methodology [16], [17], [18] that can extract meaningful and discriminative features from unlabeled data and regularize representations with good generalization capabilities. The conventional way to apply contrastive learning on graphs is to first obtain different views of graph structures by explicit data augmentation, e.g., stochastic dropout of nodes, edges, or subgraphs with a certain probability [18], [19], [20]. Then the learning objective is to maximize the consistency of the same samples under different views and distinguish different samples simultaneously. While it seems straightforward to apply similar augmentation techniques to our topic of interest, such stochastic data manipulation may however introduce undesired variations to the learning process, as graph hashing is more sensitive to such structure variations due to its inherent discreteness property.

To provide a more robust contrastive learning paradigm

for graph hashing, in this work we put forward the model **B**ipartite **G**raph **C**ontrastive **H**ashing (BGCH+). BGCH+ builds upon the topology-aware hashing paradigm of BGCH while proposing a novel **dual feature contrastive learning** framework. Specifically, different from the conventional augmentation to input data, we propose to conduct feature-level augmentations to both *intermediate information* and *binarized hash codes* when training. Via random noise addition, BGCH+ generates perturbation between contrastive views while maintaining the noise with controlled magnitude. This feature-based augmentation approach, as opposed to explicit manipulation of the graph structure, is thus more flexible and efficient to implement. Subsequently, these two streams of augmentations are employed within a dual feature contrastive learning objective, incorporating distinct learning granularities. As shown in the lower section of Figure 1(b), this learning mechanism of BGCH+ eventually enhances the *distribution uniformity* of the resultant target hash codes, thereby improving their robustness when confronted with structural variations.

Based on the well-learned hash codes, BGCH+ substantially improves the performance in Top-N Hamming space retrieval. The quality-cost trade-off is summarized in the upper section of Figure 1(b), where BGCH+ is compared to several representative counterparts, including float32-based and hashing-based models. The evaluation is conducted on a real-world bipartite graph with over 10 million observed edges, and further experimental details can be found in § 5.1. Notably, as the figure lower-right corner indicates the ideal optimal performance, BGCH+ surpasses BGCH and even achieves a comparable prediction accuracy to existing full-precision models, while still maintaining over $8\times$ computation acceleration. To summarize, our early model BGCH [15] has the following contributions:

- BGCH studies the problem of learning to hash on bipartite graphs with graph representation learning, and proposes an effective approach for effective and efficient Top-N search in Hamming space.
- BGCH provides both theoretical and empirical effectiveness on several datasets and demonstrates efficiency in both time and space complexity.

Extending these early findings, our advanced model BGCH+ further presents the new contributions as follows:

- BGCH+ focuses on improving the learning quality of graph convolutional hashing via leveraging self-supervised learning. To the best of our knowledge, BGCH+ is the first to elucidate benefits of contrastive learning for graph hashing.
- BGCH+ proposes a novel dual feature contrastive learning paradigm that operates on feature augmentations of intermediate information and output hash codes, which is different with the conventional manner of complicated structural manipulation.
- We conduct a comprehensive evaluation on six real-world datasets. The empirical analyses demonstrate the efficacy of learning high-quality hash codes and its superiority in surpassing its predecessor BGCH and other counterparts.

We organize this paper as follows. We introduce the preliminaries in § 2 and formally present BGCH+ methodology in § 3. The complexity analysis is conducted in § 4. Then

we detail all experiments and review the related work in § 5 and § 6 with the conclusion in § 7.

2 PRELIMINARIES AND PROBLEM FORMULATION

Graph Convolution Network (GCN). The general idea of GCN is to learn node embeddings by *iteratively propagating and aggregating* latent features of node neighbors via the graph topology [21], [7], [22]:

$$\mathbf{V}_x^{(l)} = \text{AGG} \left(\mathbf{V}_x^{(l-1)}, \{ \mathbf{V}_z^{(l-1)} : z \in \mathcal{N}(x) \} \right), \quad (1)$$

where $\mathbf{V}_x^{(l)} \in \mathbb{R}^d$ denotes node x 's embedding after l -th iteration of graph convolutions, indexed in the embedding matrix \mathbf{V} . $\mathcal{N}(x)$ is the set of x 's neighbors. Function $\text{AGG}(\cdot, \cdot)$ is the information aggregation function, with several implementations in previous work [22], [6], [23], [24], mainly aiming to transform the center node feature and the neighbor features. In this work, we adopt the state-of-the-art graph convolution paradigm [7].

Bipartite Graph and Adjacency Matrix. The bipartite graph is denoted as $\mathcal{G} = \{\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}\}$, where \mathcal{V}_1 and \mathcal{V}_2 are two *disjoint* node sets and \mathcal{E} is the set of edges between nodes in \mathcal{V}_1 and \mathcal{V}_2 . We can use $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}_1| \times |\mathcal{V}_2|}$ to indicate the edge transactions, where 1-valued entries, i.e., $\mathbf{Y}_{x,y} = 1$, indicate there is an observed edge between nodes $x \in \mathcal{V}_1$ and $y \in \mathcal{V}_2$, otherwise $\mathbf{Y}_{x,y} = 0$. Then the adjacency matrix \mathbf{A} of the whole graph can be defined as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{Y} \\ \mathbf{Y}^T & \mathbf{0} \end{bmatrix}. \quad (2)$$

Problem Formulation. Given a bipartite graph $\mathcal{G} = \{\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}\}$ along with its adjacency matrix \mathbf{A} , we devote to learn a hashing function:

$$F(\mathbf{A}|\Theta) \rightarrow \mathcal{Q}, \quad (3)$$

where Θ is the set of all learnable parameters and embeddings and F maps nodes into the d -dimensional Hamming space. Given two nodes in the bipartite graph, e.g., $x \in \mathcal{V}_1$ and $y \in \mathcal{V}_2$, their hash codes are \mathcal{Q}_x and \mathcal{Q}_y . Then the probability of edge existence $\hat{\mathbf{Y}}_{x,y}$ between nodes $x \in \mathcal{V}_1$ and $y \in \mathcal{V}_2$ can be effectively and efficiently measured by the hash codes \mathcal{Q}_x and \mathcal{Q}_y , i.e., $\hat{\mathbf{Y}}_{x,y} = f(\mathcal{Q}_x, \mathcal{Q}_y)$ where f is a score function. Intuitively, the larger value $\hat{\mathbf{Y}}_{x,y}$ is, the more likely x and y are matched, i.e., an edge between x and y exists. We use bold uppercase and calligraphy characters for matrices and sets. The non-bolded denote graph nodes or scalars. Explanations of key notations used in this paper are attached in Table 1.

3 BGCH+: METHODOLOGY

3.1 Overview

We hereby formally introduce our BGCH+ model, which incorporates the following key modules: (1) *adaptive graph convolutional hashing* (§ 3.2) provides an effective encoding approach to enhance the expressivity of hashed features significantly while ensuring efficient matching in Hamming space; (2) *dual feature augmentation for contrastive learning* (§ 3.3) constructs effective data augmentations for both intermediate continuous information and target hash codes in dual manner, striving to improve the quality and discriminability of the outputs. (3) *Fourier serialized gradient*

TABLE 1: Notations and meanings.

Notation	Meaning
$\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2, \mathcal{E}$	Bipartite graph with node and edge sets.
\mathbf{A}, \mathbf{D}	Adjacency and diagonal degree matrices.
\mathbf{Y}	Edge transactions where 1 indicates the interaction existence, and 0 otherwise.
$\tilde{\mathbf{Y}}$	Estimated matching scores.
d	Hash code dimension.
L	Numbers of convolutional hashing.
$\mathbf{V}_x^{(l)}$	Node x 's intermediate feature at iteration l .
$\mathbf{Q}_x^{(l)}$	Binary embedding of node x at iteration l .
$\alpha^{(l)}$	x 's rescaling factor computed at iteration l .
$\epsilon_x^{(l)'}, \epsilon_x^{(l)''}, \varrho_x^{(l)}, \varrho_x^{(l)''}$	Perturbation noises.
$\mathbf{V}_x^{(l)'}, \mathbf{V}_x^{(l)''}$	Augmented continuous features of x .
$\mathcal{Q}_x^{(l)'}, \mathcal{Q}_x^{(l)''}$	Augmented hash codes of x .
$\alpha_x^{(l)'}, \alpha_x^{(l)''}$	Augmented binarized embeddings of x .
\mathcal{Q}_x	Final output hash codes of node x .
$\mathcal{L}_{cl}^1, \mathcal{L}_{cl}^2$	Dual contrastive loss terms.
$\mathcal{L}_{cl}, \mathcal{L}_{bpr}, \mathcal{L}$	Two loss terms of final objective \mathcal{L} .
$\eta, \tau, \sigma, H, n, \lambda_1, \lambda_2$	Hyperparameters.

estimation (§ 3.4) introduces the Fourier Series decomposition for $\text{sign}(\cdot)$ in the frequency domain. By leveraging this technique, more accurate gradient approximation can be achieved. Incorporating the learned hash codes, BGCH+ also incorporates efficient online matching using the Hamming distance measurement (§ 3.5). Our model illustration is attached in Figure 2.

3.2 Adaptive Graph Convolutional Hashing

To enhance expressivity and smooth loss landscapes, one approach is to incorporate the *relaxation strategy*. Apart from the topology-aware embedding hashing with $\text{sign}(\cdot)$:

$$\mathbf{Q}_x^{(l)} = \text{sign}(\mathbf{V}_x^{(l)}), \quad (4)$$

we introduce an effective way to increase the flexibility and smoothness of the learned representations, via additionally computing a layer-wise positive rescaling factor for each node as follows, such that $\alpha_x^{(l)} \in \mathbb{R}^+$ and $\mathbf{V}_x^{(l)} \approx \alpha_x^{(l)} \mathbf{Q}_x^{(l)}$:

$$\alpha_x^{(l)} = \frac{1}{d} \|\mathbf{V}_x^{(l)}\|_1. \quad (5)$$

Instead of treating these factors as learnable parameters, such deterministic computation substantially prunes the parameter search space while attaining the adaptive approximation functionality for different graph nodes. We demonstrate this in § 5.5 of experiments.

After L iterations of hashing, we obtain the table of **adaptive hash codes** $\mathcal{Q} = \{\alpha, \mathbf{Q}\}$, where $\alpha \in \mathbb{R}^{(|\mathcal{V}_1|+|\mathcal{V}_2|) \times (L+1)}$ and $\mathbf{Q} \in \mathbb{R}^{(|\mathcal{V}_1|+|\mathcal{V}_2|) \times (L+1) \times d}$. For each node x , its corresponding hash codes are indexed and assembled:

$$\alpha_x = \alpha_x^{(0)} \|\alpha_x^{(1)}\| \cdots \|\alpha_x^{(L)}\|, \text{ and } \mathbf{Q}_x = \mathbf{Q}_x^{(0)} \|\mathbf{Q}_x^{(1)}\| \cdots \|\mathbf{Q}_x^{(L)}\|. \quad (6)$$

Intuitively, table \mathcal{Q} encodes bipartite structural information that is propagated back and forth at different iteration steps l , i.e., from 0 to the maximum step L . It not only tracks the intermediate knowledge hashed for all graph nodes, but also maintains the value approximation to their original continuous embeddings, e.g., $\mathbf{V}_x^{(l)}$. In addition, with the slightly more space cost (complexity analysis in § 4), such detached hash encoding approach still supports the bitwise operations (§ 3.5) for accelerating inference and matching.

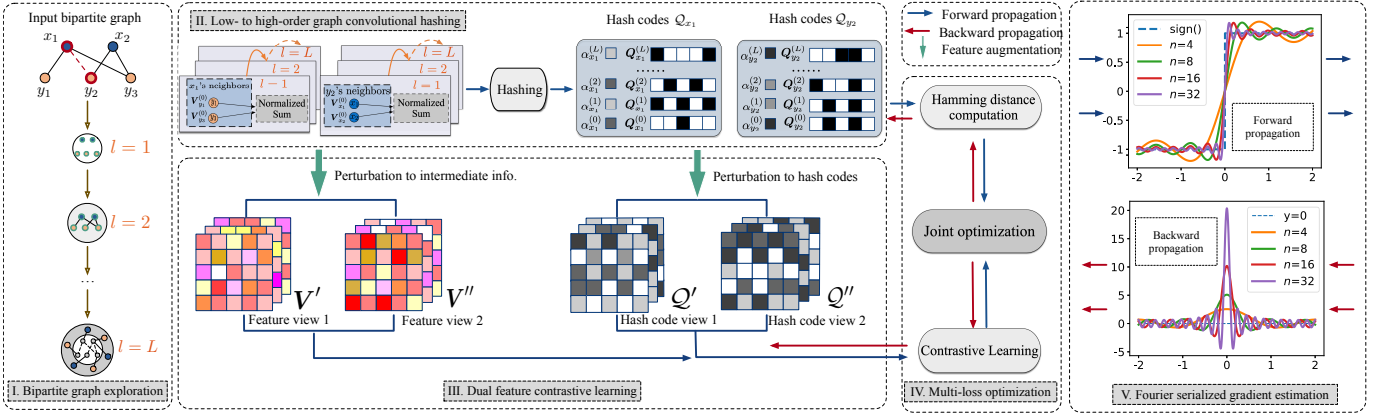


Fig. 2: Workflow illustration of BGCH+ framework (best view in color).

3.3 Dual Feature Contrastive Learning

3.3.1 Dual Feature Augmentation

Diverging from conventional methods that manipulate graph structures, such as dropout of edges and nodes [18], [19], [20], which often prove to be intractable and time-consuming, our research focuses on exploring contrastive learning directly on features within the embedding spaces. Specifically, we introduce a procedure involving the addition of random noises [25], [26] to both the embeddings before and after hashing. Given the node x with the segments of its *continuous embedding* $V_x^{(l)}$ and *binarized hash code* $\alpha_x^{(l)} Q_x^{(l)}$ at the l -th layer of graph convolution, we conduct the augmentation for continuous embedding $V_x^{(l)}$ as:

$$V_x^{(l)'} = V_x^{(l)} + \epsilon_x^{(l)'}, \quad V_x^{(l)''} = V_x^{(l)} + \epsilon_x^{(l)''}. \quad (7)$$

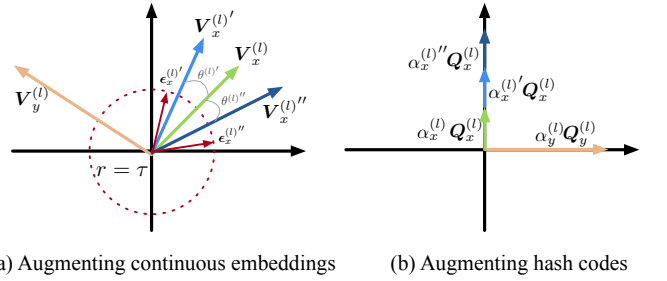
These perturbation vectors $\epsilon_x^{(l)'}, \epsilon_x^{(l)''} \in \mathbb{R}^d$ are drawn by:

$$\|\epsilon^{(l)}\|_2 = \tau, \quad \epsilon^{(l)} = \overline{\epsilon^{(l)}} \odot Q_x^{(l)}, \quad (8)$$

where $\overline{\epsilon^{(l)}} \in \mathbb{R}^d \sim U(0, 1)$ and \odot is the Hadamard product. The hyperparameter τ controls the *embedding uniformity* which will be empirically analyzed later in § 5.4. In Equation (8), the first constraint mainly shapes the magnitude of perturbation vector, e.g., $\epsilon_x^{(l)'}$, and it numerically corresponds to the point on a hypersphere with the radius τ . The second constraint ensures that $V_x^{(l)}$, $\epsilon_x^{(l)'}$, and $\epsilon_x^{(l)''}$ are located in the same region. This requirement constrains them to have the same direction with $Q_x^{(l)}$ and prevents the addition of noises from causing significant deviations in $V_x^{(l)}$. We visually depict the process in Figure 3(a).

By incorporating these scaled noise vectors to $V_x^{(l)}$, we essentially rotate them by two small angles, denoted as $\theta^{(l)'}$ and $\theta^{(l)''}$. Each rotation corresponds to a modification in $V_x^{(l)}$ and produces a pair of augmented representations, i.e., $V_x^{(l)'}$ and $V_x^{(l)''}$. Due to the small rotation magnitude, the augmented embeddings capture both the essential features of the original information and introduce slight and acceptable variations.

Likewise, we implement the feature augmentation on the output hash codes. However, due to the discreteness of Hamming space, it would introduce significant variations



(a) Augmenting continuous embeddings

(b) Augmenting hash codes

Fig. 3: Dual feature augmentation for contrastive learning.

to the binary embeddings, i.e., $Q_x^{(l)}$, if the noise is simply perturbed on it. Therefore, as illustrated in Figure 3(b), we opt to add noise to the scalar value $\alpha_x^{(l)}$ as follows:

$$\alpha_x^{(l)'} = \alpha_x^{(l)} + \varrho_x^{(l)'}, \quad \alpha_x^{(l)''} = \alpha_x^{(l)} + \varrho_x^{(l)''}, \quad (9)$$

where $\varrho_x^{(l)} \in \mathbb{R} \sim U(0, 1)$. Instead of directly perturbing $Q_x^{(l)}$, this approach allows us to introduce controlled perturbations without affecting the binarization nature of hashing.

3.3.2 Dual Feature Contrastive Learning Objectives

After iterative graph convolutions, we achieve two views of feature-based augmentations. For continuous intermediate information, we have the contrastive optimization term as:

$$\mathcal{L}_{cl}^1 = \sum_{x \in \mathcal{B}} -\log \frac{\exp(\frac{V_x'^T V_x''}{\sigma})}{\sum_{y \in \mathcal{B}} \exp(\frac{V_x'^T V_y''}{\sigma})}, \quad (10)$$

where \mathcal{B} denotes a training batch and the hyperparameter $\sigma > 0$. V_x' and V_x'' are concatenated representations from:

$$V_x' = V_x^{(0)'} \| V_x^{(1)'} \| \dots \| V_x^{(L)'}, \quad V_x'' = V_x^{(0)''} \| V_x^{(1)''} \| \dots \| V_x^{(L)'}. \quad (11)$$

On the other hand, for the hash code Q_x , we can similarly obtain two augmented views Q_x' and Q_x'' and have:

$$\mathcal{L}_{cl}^2 = \sum_{x \in \mathcal{B}} -\log \frac{\exp(\frac{\alpha_x' \alpha_x''}{\sigma} \| Q_x \|^2)}{\sum_{y \in \mathcal{B}} \exp(\frac{\alpha_x' \alpha_y''}{\sigma} Q_x^T Q_y)}, \quad (12)$$

where $\|Q_x\|^2$ and $Q_x^T Q_y$ can be efficiently computed in Hamming space (introduced later in § 3.5).

Generally, these two contrastive loss terms encourage consistency between the augmented representations of the same node x , e.g., Q_x' and Q_x'' , while maximizing the dis-

agreement between embeddings of different nodes, e.g., x and y . And our proposed dual contrastive feature objective can further strengthen the optimization effect to both *continuous intermediate information* as well as the *target hash codes*, eventually producing a fine-grained learning of hash codes with good “uniformity”. We provide a detailed empirical study to this design in § 5.4.

3.4 Fourier Serialized Gradient Estimation

To provide the accordant gradient estimation for hash function $\text{sign}(\cdot)$, we approximate it by utilizing its Fourier Series decomposition in the frequency domain. Specifically, $\text{sign}(\cdot)$ can be regarded as a special instance of the periodical Square Wave Function $t(x)$ within the interval of length $2H$, i.e., $\text{sign}(\phi) = t(\phi)$, $|\phi| < H$. By decomposing $t(x)$ into its Fourier Series form, we shall have:

$$\text{sign}(\phi) = \frac{4}{\pi} \sum_{i=1,3,5,\dots}^{+\infty} \frac{1}{i} \sin\left(\frac{\pi i \phi}{H}\right), \quad \text{where } |\phi| < H. \quad (13)$$

Fourier Series decomposition of $\text{sign}(\cdot)$ with infinite terms is a lossless transformation [27]. Therefore, as depicted in Figure 2(c), we can set the finite expanding term n to obtain its approximation version as follows:

$$\text{sign}(\phi) \doteq \frac{4}{\pi} \sum_{i=1,3,5,\dots}^n \frac{1}{i} \sin\left(\frac{\pi i \phi}{H}\right). \quad (14)$$

The corresponding derivatives can be obtained as:

$$\frac{\partial \text{sign}(\phi)}{\partial \phi} \doteq \frac{4}{H} \sum_{i=1,3,5,\dots}^n \cos\left(\frac{\pi i \phi}{H}\right). \quad (15)$$

Unlike other gradient estimators such as tanh-alike [28], [10] and SignSwish [29], approximating the $\text{sign}(\cdot)$ function with its Fourier Series does not distort the main direction of the true gradients during model optimization [30]. This characteristic is advantageous as it facilitates a coordinated transformation from continuous values to their corresponding binarizations for node representations. Consequently, it effectively preserves the discriminability of hash codes and leads to improved retrieval accuracy. We present this performance comparison in § 5.7 of experiments. In summary, as indicated in Equation (16), we utilize the strict $\text{sign}(\cdot)$ during forward propagation to encode hashing embeddings, while estimating the gradients $\frac{\partial \text{sign}(\phi)}{\partial \phi}$ for backward propagation.

$$\begin{cases} \mathbf{Q}^{(l)} = \text{sign}(\phi), & \text{Forward propagation.} \\ \frac{\partial \mathbf{Q}^{(l)}}{\partial \phi} \doteq \frac{4}{H} \sum_{i=1,3,5,\dots}^n \cos\left(\frac{\pi i \phi}{H}\right). & \text{Backward propagation.} \end{cases} \quad (16)$$

3.5 Model Prediction and Optimization

3.5.1 Matching score prediction

Given two nodes $x \in \mathcal{V}_1$ and $y \in \mathcal{V}_2$, one natural manner to implement the score function is *inner-product*, mainly for its simplicity as:

$$\hat{\mathbf{Y}}_{x,y} = (\alpha_x \mathbf{Q}_x)^\top \cdot (\alpha_y \mathbf{Q}_y). \quad (17)$$

However, the inner product in Equation (17) is still conducted in the (continuous) Euclidean space with *full-precision arithmetics*. To bridge the connection between the

inner product and Hamming distance measurement, we introduce the theorem:

Theorem 1 (Hamming Distance Matching). *Given two hash codes, we have $(\alpha_x \mathbf{Q}_x)^\top \cdot (\alpha_y \mathbf{Q}_y) = \alpha_x \alpha_y (d - 2D_H(\mathbf{Q}_x, \mathbf{Q}_y))$.*

Proof.

$$\begin{aligned} & \mathbf{Q}_x^\top \cdot \mathbf{Q}_y \\ &= |\{i | (\mathbf{Q}_x)_i = (\mathbf{Q}_y)_i = 1\}| + |\{i | (\mathbf{Q}_x)_i = (\mathbf{Q}_y)_i = -1\}| \\ &\quad - |\{i | (\mathbf{Q}_x)_i \neq (\mathbf{Q}_y)_i\}| \\ &= d - 2 \cdot |\{i | (\mathbf{Q}_x)_i \neq (\mathbf{Q}_y)_i\}| = \underline{d - 2D_H(\mathbf{Q}_x, \mathbf{Q}_y)}, \end{aligned} \quad (18)$$

which completes the proof. \square

$D_H(\cdot, \cdot)$ is Hamming distance between two inputs. By applying Theorem 1, we transform the score computation to the Hamming distance matching. This transformation allows to significantly reduce the number of floating-point operations (#FLOPs) in the original score computation formulation (Equation (17)) to efficient Hamming distance matching. Consequently, this can develop substantial computation acceleration, as further analyzed in § 4.

3.5.2 Multi-loss Objective Function

Our objective function comprises two components, i.e., BPR loss \mathcal{L}_{bpr} and contrastive learning loss \mathcal{L}_{cl} . Generally, these two loss functions harness the regularization effect to each other. The rationale behind this design is:

- \mathcal{L}_{bpr} ranks the matching scores computed from the hash codes of a given pair of graph nodes.
- \mathcal{L}_{cl} measures the consistency of nodes under different augmentation views represented by both continuous and binarized hash codes.

Concretely, we implement \mathcal{L}_{bpr} with *Bayesian Personalized Ranking* (BPR) loss as follows:

$$\mathcal{L}_{bpr} = - \sum_{x \in \mathcal{V}_1} \sum_{\substack{y \in \mathcal{N}(x) \\ y' \notin \mathcal{N}(x)}} \ln \sigma(\hat{\mathbf{Y}}_{x,y} - \hat{\mathbf{Y}}_{x,y'}). \quad (19)$$

\mathcal{L}_{bpr} encourages the predicted score of an observed edge to be higher than its unobserved counterparts [7]. As for \mathcal{L}_{cl} , we take summation of both loss terms, i.e., \mathcal{L}_{cl}^1 and \mathcal{L}_{cl}^2 introduced in § 3.3, for dual feature contrastive learning:

$$\mathcal{L}_{cl} = \mathcal{L}_{cl}^1 + \mathcal{L}_{cl}^2. \quad (20)$$

Let λ_1 be a hyperparameter and Θ denote the set of trainable embeddings regularized by the parameter λ_2 to avoid overfitting. Our final objective function is defined as:

$$\mathcal{L} = \mathcal{L}_{bpr} + \lambda_1 \mathcal{L}_{cl} + \lambda_2 \|\Theta\|_2^2. \quad (21)$$

While \mathcal{L}_{bpr} focuses on pairwise preferences and rankings, the dual contrastive learning loss helps in capturing fine-grained similarities and differences between node representations. By incorporating these loss terms, the model not only prioritizes the learning of ranking information embedded in the outputs, but also produces high-quality hash codes with better performances. The ablation study of such multi-loss optimization design is conducted in § 5.5.

So far, we have introduced all technical parts of BGCH+ and attached the pseudocodes in Algorithm 1. To better understand the model scalability of BGCH+, we provide the complexity analysis in the following section.

Algorithm 1: BGCH+ training algorithm.

```

1 while model does not converge do
2   for  $l = 0, \dots, L - 1$  do
3      $\mathbf{Q}^{(l+1)} \leftarrow \text{sign}(\mathbf{V}^{(l+1)})$ ;  $\triangleright$  Eq. (4)
4      $\alpha \leftarrow$  calculate the rescaling factors;  $\triangleright$  Eq. (5)
5     for  $x \in \mathcal{V}_1, y \in \mathcal{N}(x)$  do
6        $\mathbf{V}_x^{(l+1)'}, \mathbf{V}_x^{(l+1)''}, \mathbf{V}_y^{(l+1)'}$ ,
7        $\alpha_x^{(l+1)}, \alpha_x^{(l+1)''}, \alpha_y^{(l+1)'}$   $\leftarrow$  construct dual
8       feature augmentation;  $\triangleright$  Eq. (7-9)
9        $\mathcal{L}_{cl}^1, \mathcal{L}_{cl}^2 \leftarrow$  contrastive loss;  $\triangleright$  Eq. (10-12)
10       $\hat{\mathbf{Y}}_{x,y} \leftarrow \alpha_x \alpha_y (d - 2D_H(\mathbf{Q}_x, \mathbf{Q}_y))$ ;  $\triangleright$  Eq. (17)
11       $\mathcal{L}_{bpr} \leftarrow$  BPR ranking loss;  $\triangleright$  Eq. (19)
12     $\mathcal{L} \leftarrow$  accumulate loss for optimization;  $\triangleright$  Eq. (21)
13  Function Gradient_estimator( $\mathcal{L}$ ):
14     $\frac{\partial \mathcal{L}}{\partial \mathbf{V}} \leftarrow \frac{\partial \mathcal{L}}{\partial \mathbf{Q}} \cdot \frac{4}{H} \sum_{i=1,3,5,\dots}^n \cos(\frac{\pi i \mathbf{V}}{H})$ ;  $\triangleright$  Eq. (15)
```

TABLE 2: Traing time complexity.

Graph Norm.	Conv. & Hash.	BPR Loss	CL loss	Grad. Est.
$O(2 \mathcal{E})$	$O(\frac{2dsL \mathcal{E} ^2}{ \mathcal{B} })$	$O(2sd \mathcal{E})$	$O(2 \mathcal{B} sd \mathcal{E})$	$O(snd \mathcal{E})$

4 COMPLEXITY ANALYSIS

Training time complexity. $|\mathcal{E}|$, $|\mathcal{B}|$, s , and n are the edge number, batch size, numbers of train iterations and Fourier Series decomposition terms, respectively. As shown in Table 2, we derive that: (1) The time complexity of the graph normalization to the original adjacency matrix, i.e., $\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, is $O(2|\mathcal{E}|)$. (2) In graph convolution and hashing, the time complexity is $O(\frac{2dsL|\mathcal{E}|^2}{|\mathcal{B}|})$, where $L \leq 4$ is a common setting [7], [31], [22], [6] to avoid *over-smoothing* [32]. (3) BGCH+ takes $O(2sd|\mathcal{E}|)$ to compute \mathcal{L}_{bpr} loss. As for \mathcal{L}_{cl} loss, for each node x , we need to conduct random noise addition to all other nodes in \mathcal{B} , as shown in Equations (10-12), which is known as the widely-used in-batch sampling [17]. Thus the total loss complexity for our dual feature contrastive learning would be $O(2|\mathcal{B}|sd|\mathcal{E}|)$. (4) Lastly, BGCH+ takes $O(snd|\mathcal{E}|)$ to estimate the gradients for the d -dimension hash codes. Thus, the total complexity in total is quadratic to the graph edge numbers, i.e., $|\mathcal{E}|$. This is common in GCN frameworks and actually acceptable for large bipartite graphs, which may dispel the concerns of large training cost in practice.

Hash codes space cost. As shown in Table 3, the total space cost of hash codes is $O(|\mathcal{V}_1 \cup \mathcal{V}_2|(d + 32(L + 1)))$ bits, supposing that we use float32 for those rescaling factors in $L + 1$ iterations. Compared to the continuous embedding size, i.e., $32|\mathcal{V}_1 \cup \mathcal{V}_2|d$ bits, the size reduction ratio thus is:

$$\begin{aligned}
 \text{ratio} &= \frac{32|\mathcal{V}_1 \cup \mathcal{V}_2|d}{|\mathcal{V}_1 \cup \mathcal{V}_2|(d + 32(L + 1))} = \frac{32d}{d + 32(L + 1)} \\
 &= \frac{1}{1 + 32(\frac{L}{d} + \frac{1}{d})}.
 \end{aligned} \quad (22)$$

Based on our previous explanation, it has been noted that excessively stacking iteration layers can lead to a decline in performance [32]. Therefore, if $L \leq 4$, BGCH+ has the potential to achieve a significant increase in the size compression ratio, if the embedding dimension is increased.

Online matching. The original score formulation in

TABLE 3: Complexity of space and computation.

	Space cost	#FLOP	#BOP
float32-based	$32 \mathcal{V}_1 \cup \mathcal{V}_2 d$	$O(\mathcal{V}_1 \cdot \mathcal{V}_2 d)$	-
BGCH+	$ \mathcal{V}_1 \cup \mathcal{V}_2 (d + 32(L + 1))$	$O(4 \mathcal{V}_1 \cdot \mathcal{V}_2)$	$O(\mathcal{V}_1 \cdot \mathcal{V}_2 d)$

Equation (17) contains d floating-point operations (#FLOPs). As shown in Table 3, using Hamming distance matching can convert the most of floating-point arithmetics to binary operations (#BOPs), with a few #FLOPs for scalar computations, i.e., $4 \ll d$.

5 EXPERIMENTAL EVALUATION

We evaluate BGCH+ with the aim of answering the following research questions:

- **RQ1.** How does BGCH+ compare to the state-of-the-art hashing-based models in Top-N Hamming retrieval?
- **RQ2.** What is the performance gap between BGCH+ and full-precision models *w.r.t.* long-list retrieval quality?
- **RQ3.** How does our proposed dual feature contrastive learning contribute to the performance of BGCH+?
- **RQ4.** What are advantages of other model components?
- **RQ5.** What is the resource consumption of BGCH+?
- **RQ6.** How does Fourier Series decomposition perform in terms of retrieval accuracy and training efficiency?

5.1 Experiment Setup

Datasets and Evaluation Metrics. We include six real-world bipartite graphs in Table 4 that are widely evaluated [7], [33], [34], [35], [31], [36] as follows:

- 1) **MovieLens**² is a widely adopted benchmark for movie review. Similar to the setting in [13], if the user x has rated item y , we set the edge $\mathbf{Y}_{x,y} = 1, 0$ otherwise.
- 2) **Gowalla**³ [31], [13], [7], [37] is the dataset [38] between *customers* and *check-in locations* collected from Gowalla.
- 3) **Pinterest**⁴ is an open dataset for image recommendation between *users* and *images*. Edges represent the pins over images initiated by users.
- 4) **Yelp2018**⁵ is from Yelp Challenge 2018 Edition, bipartitely modeling between *users* and *local businesses*.
- 5) **AMZ-Book**⁶ is the bipartite graph between *readers* and *books*, organized from Amazon-review [39].
- 6) **Dianping**⁷ is a commercial dataset between *users* and *local businesses* recording their diverse interactions, e.g., clicking, saving, and purchasing.

Evaluation Metrics. To assess the model performance in Hamming space retrieval over bipartite graphs, we use the hash codes to find the Top-N answers for a given query node based on the closest Hamming distances. We then evaluate the ranking capability using two commonly used evaluation protocols, i.e., Recall@N and NDCG@N.

Baselines. In addition to BGCH, we include the following representative hashing-based models for (1) general

2. <https://grouplens.org/datasets/movielens/1m/>

3. <https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/gowalla>

4. https://sites.google.com/site/xueatalphabeta/dataset-1/pinterest_iccv

5. <https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018>

6. <https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/amazon-book>

7. <https://www.dianping.com/>

TABLE 4: The statistics of datasets.

	MovieLens	Gowalla	Pinterest	Yelp2018	AMZ-Book	Dianping
$ \mathcal{V}_1 $	6,040	29,858	55,186	31,668	52,643	332,295
$ \mathcal{V}_2 $	3,952	40,981	9,916	38,048	91,599	1,362
$ \mathcal{E} $	1,000,209	1,027,370	1,463,556	1,561,406	2,984,108	10,000,014
Density	0.04190	0.00084	0.00267	0.00130	0.00062	0.02210

TABLE 5: Hyperparameter settings.

	MovieLens	Gowalla	Pinterest	Yelp2018	AMZ-Book	Dianping
η	1×10^{-2}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-4}
λ_1	1×10^{-2}	5×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-2}	5×10^{-3}
λ_2	1×10^{-5}	1×10^{-5}	1×10^{-4}	1×10^{-4}	1×10^{-5}	1×10^{-5}
τ	0.1	0.1	0.1	0.1	0.1	0.1
σ	0.2	0.2	0.2	0.2	0.1	0.2
n	8	16	8	4	8	4
H	1	1	1	1	1	1

object retrieval (LSH [40]), (2) image search (HashNet [41]), and (3) Top-N candidate generation for recommendation (Hash_Gumbel [42], [43], CIGAR [44] and HashGNN [13]). We also include several state-of-the-art full-precision (i.e., denoted by FT32 as implemented with float32 in experiments) recommender models, i.e., NeurCF [45], NGCF [31], DGCF [37], LightGCN [7], for the long-list ranking quality comparison:

- 1) **LSH** [40] is a classical hashing method. LSH is proposed to approximate the similarity search for massive high-dimensional data and we introduce it for Top-N object search by following the adaptation in [13].
- 2) **HashNet** [41] is a representative deep hashing method that is originally proposed for multimedia retrieval tasks. Similar to [13], we adapt it for graph data by modifying it with the general graph convolutional network.
- 3) **CIGAR** [44] is a state-of-the-art neural-network-based framework for fast Top-N candidate generation in recommendation. CIGAR can be further followed by a full-precision re-ranking algorithm. And we only use its hashing part for fair comparison.
- 4) **Hash_Gumbel** is a variance of BGCH+ with Gumbel-softmax for hash encoding and gradient estimation [42], [43]. Specifically, we first expand each embedding bit to a size-two one-hot encoding. Then it utilizes the Gumbel-softmax trick to replace $\text{sign}(\cdot)$ as relaxation for binary hash code generation.
- 5) **HashGNN** [13] is the state-of-the-art learning to hash based method with GCN framework. We use HashGNN_{*h*} to denote the vanilla version with *hard encoding* proposed in [13], where each element of user-item embeddings is strictly binarized. We use HashGNN_{*s*} to denote its proposed approximated version.
- 6) **BGCH** [15] is the vanilla graph-base method with no feature augmentations for bipartite graph hashing.
- 7) **NeurCF** [45] is one representative deep neural network model for collaborative filtering in recommendation.
- 8) **NGCF** [31] is one of the representative graph-based models with collaborative filtering methodology.
- 9) **DGCF** [37] is a graph-based model that learns disentangled user intents with state-of-the-art performance.
- 10) **LightGCN** [7] is another state-of-the-art GCN-based rec-

ommender model that has been widely evaluated.

The early hashing methods such as SH [46], RMMH [47], and LCH [48] are not considered in our evaluation due to the established performance superiority of the competing models [41], [44] over them.

Implementations and Hyperparameter Settings. We implemented our models using Python 3.6 and PyTorch 1.14.0 on a Linux machine equipped with 4 Nvidia V100 GPUs and 4 Intel Core i7-8700 CPUs. For the baselines, we followed the official hyperparameter settings or conducted a grid search if the settings were not available. In the following sections, we set the dimension d to 256 and graph convolutions L to 2. The learning rate η and the coefficients λ_1, λ_2 were tuned among $\{5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-4}, 10^{-4}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-2}, 10^{-2}\}$. We initialize and optimize all models with default normal initializer and Adam optimizer [49]. All hyperparameters are reported in Table 5.

5.2 Top-N Hamming Space Query (RQ1)

To assess the **fine-to-coarse** Top-N ranking capability, we fix $N=1000$. We initially present the results of Recall@20₁₀₀₀ and NDCG@20₁₀₀₀⁸ for the Top-1000 search in Table 6. Additionally, we plot the holistic Recall and NDCG metric curves for the $\{20, 50, 100, 200, 500, 1000\}$ of Top-1000 ranking in Figure 4. For fair comparison, we ensure that both BGCH+ and the baselines had the convolution iteration number of 2 and the embedding dimension of 256.

- **The results clearly establish the superiority of our BGCH+ over previous hashing-based models.** (1) Firstly, as shown in Table 6, HashGNN outperforms traditional hashing-based baselines such as LSH, HashNet, CIGAR. This suggests that directly adapting conventional non-graph-based hashing methods may struggle to achieve comparable performance due to the effectiveness of *graph convolutional* architecture in capturing latent information within the bipartite graph topology for hash encoding preparation. (2) Secondly, both BGCH and BGCH+ consistently outperform HashGNN over all datasets, thanks to the proposed *adaptive graph convolutional hashing*. BGCH+ further achieves improvement over its vanilla version BGCH by 2.75%~20.19%, and 2.62%~19.00% *w.r.t.* Recall@20 and NDCG@20, respectively. This highlights the effectiveness of our newly proposed dual feature contrastive learning paradigm. A comprehensive analysis will be conducted in § 5.4. (3) Thirdly, we conduct the Wilcoxon signed-rank tests on BGCH+. The results confirm that all improvements of BGCH+ over BGCH are statistically significant at a 95% confidence level. Considering the performance improvement of BGCH+ and BGCH over all other methods, this demonstrates the effectiveness of all proposed modules contained therein. Detailed ablation study will be introduced in § 5.5.
 - **By varying N from 20 to 1000, both BGCH and BGCH+ consistently demonstrate competitive performance compared to the baselines.** The observations from Figure 4 are as follows: (1) Compared to the approximated version of HashGNN, i.e., HashGNN_{*s*}, both BGCH and BGCH+ consistently exhibit stable and significant
8. We use notation Recall@20, NDCG@20 if no ambiguity is caused.

TABLE 6: Results of Recall@20 and NDCG@20 in Top-1000 retrieval: (1) “R” and “N” denote the Recall and NDCG; (2) the bold indicate BGCH+ and the underline represents the best and second-best performing models; (3) * denotes scenarios where Wilcoxon signed-rank tests indicate statistically significant improvements with over 95% confidence level.

Dataset Metric	MovieLens (%)		Gowalla (%)		Pinterest (%)		Yelp2018 (%)		AMZ-Book (%)		Dianping (%)	
	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20
LSH	11.38	25.87	8.14	12.23	7.88	6.71	2.91	4.35	2.41	2.34	5.85	5.84
HashNet	15.43	32.23	11.38	13.74	10.27	7.33	3.37	4.41	2.86	2.71	6.24	5.59
CIGAR	14.84	31.73	11.57	14.21	10.34	8.53	3.65	4.57	3.05	3.03	6.91	6.03
Hash_Gumbel	16.62	32.48	12.26	14.68	10.53	8.74	3.85	5.12	2.69	3.24	8.29	6.43
HashGNN _h	14.21	31.83	11.63	14.21	10.15	8.67	3.77	5.04	3.09	3.15	8.34	6.68
HashGNN _s	19.87	33.21	13.45	14.87	12.38	9.11	4.86	5.34	3.34	3.45	9.57	7.13
BGCH	<u>22.86</u>	<u>36.26</u>	<u>16.73</u>	<u>16.48</u>	<u>12.78</u>	<u>9.42</u>	<u>5.51</u>	<u>5.84</u>	3.48	3.92	<u>10.66</u>	<u>7.63</u>
BGCH+	24.37*	37.21*	17.19*	17.10*	15.36*	11.21*	5.96*	6.17*	3.78*	4.35*	12.05*	8.81*
% Gain	6.61%	2.62%	2.75%	3.76%	20.19%	19.00%	8.17%	5.65%	8.62%	10.97%	13.04%	15.47%

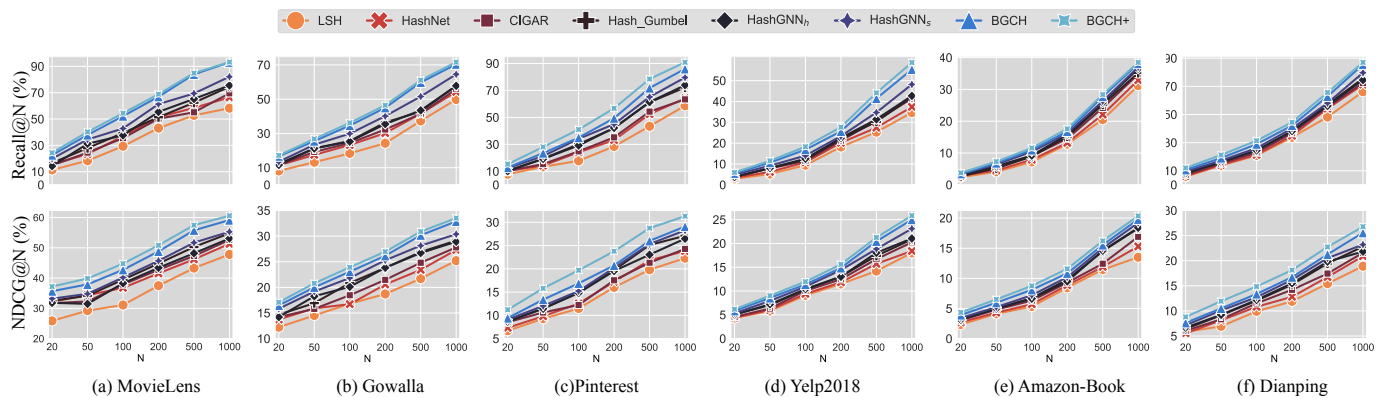


Fig. 4: Top-N retrieval quality with N in $\{20, 50, 100, 200, 500, 1000\}$ (best view in color).

improvements in both Recall and NDCG metrics across all six benchmarks for N ranging from 20 to 1000. (2) In addition to achieving higher retrieval quality, BGCH and BGCH+ possess another advantage over HashGNN_s: they retain support for bitwise operations, specifically hamming distance matching, which facilitates accelerated inference. However, HashGNN_s adopts a Bernoulli random variable to determine the probability of replacing certain digits in the hash codes with original continuous values, which disables bitwise computation. As detailed in § 5.6, such design achieves over $8\times$ inference acceleration compared to HashGNN_s, which is particularly promising for query-based online matching and retrieval applications.

5.3 Comparing to FT32-based Models (RQ2)

In this section, we compare both BGCH and BGCH+ with several full-precision (FT32-based) models to evaluate the long-list search quality. The observations from Table 7 are as follows: (1) Both BGCH and BGCH+ consistently deliver competitive performance compared to early full-precision models, e.g., NeurCF and NGCF, across all datasets. In terms of the state-of-the-art model LightGCN, both BGCH and BGCH+ achieve over 87% of the Recall@1000 and 90% of NDCG@1000 capability. (2) We notice that, compared to NDCG@1000 metric, the Recall@1000 results are generally with larger numerical values. This is because the recall metric focuses on how many ground-truth items are re-

trieved, and thus these models perform quite well. On the other hand, the NDCG metric cares more about the ranking capability, which usually leads to a relatively smaller values. (3) The performance of BGCH and BGCH+ highlights their effectiveness in ensuring high-quality long-list Top-N retrieval with competitive Recall@1000 and NDCG@1000 metrics, where BGCH+ further improves its performance across all datasets. This is particularly valuable in industrial applications such as recommender systems, which involve two major stages: *candidate generation* and *re-ranking*. Having a high-quality candidate generation significantly reduces the complexity of the subsequent re-ranking stage as the search space is substantially pruned. (4) Considering the *efficiency in Hamming space retrieval* and the *reduced space cost* of those learned hash codes, BGCH and BGCH+ can be seen as viable alternatives to these full-precision models to balance retrieval performance and computational efficiency, especially in scenarios with limited computation resources.

5.4 Study of Dual Feature Contrastive Learning (RQ3)

In this section, we conduct a comprehensive empirical analysis to examine the impact of our dual feature contrastive learning on the quality of hashed representations.

Structure Manipulation V.S. Feature Augmentation.

The conventional contrastive learning usually requires explicit graph structure manipulation [50] such as:

- *Node dropout (denoted by ND)*: with a certain probability, the graph node and its connected edges are discarded.

TABLE 7: Results of Float32-based models.

Dataset Metric	MovieLens (%)		Gowalla (%)		Pinterest (%)		Yelp2018 (%)		AMZ-Book (%)		Dianping (%)	
	R@1000	N@1000	R@1000	N@1000	R@1000	N@1000	R@1000	N@1000	R@1000	N@1000	R@1000	N@1000
NeurCF	96.90	58.76	73.28	32.07	91.29	28.79	58.83	24.69	40.29	19.83	89.39	25.54
NGCF	97.32	60.28	76.16	32.13	92.93	29.78	59.97	25.23	41.22	20.37	90.92	25.76
DGCF	98.48	62.41	76.90	34.97	96.52	31.47	62.18	26.28	42.71	21.74	92.66	26.87
LightGCN	98.27	62.88	77.74	35.26	96.59	31.32	62.31	26.55	43.89	21.92	94.37	27.28
BGCH	90.44	59.16	70.45	32.87	86.30	29.09	56.11	25.01	38.27	19.79	88.26	25.57
% Capability	91.84%	94.08%	90.62%	93.22%	89.07%	92.44%	90.05%	94.20%	87.20%	90.28%	93.53%	93.73%
BGCH+	92.43	60.60	72.22	33.52	91.29	31.31	58.28	25.85	39.38	20.36	89.83	26.77
% Capability	93.86%	96.37%	92.90%	95.07%	94.51%	99.49%	93.53%	97.36%	89.72%	92.88%	95.19%	98.13%

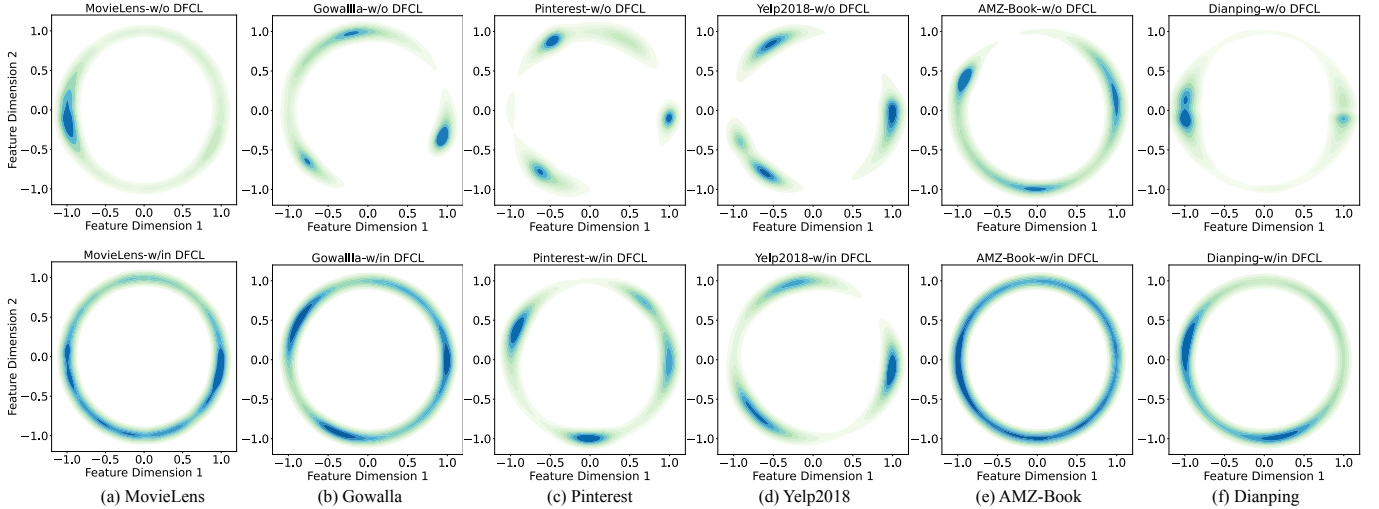


Fig. 5: Distribution illustration of learned hash codes between BGCH and BGCH+ using Gaussian kernel density estimation (KDE) over six datasets with (1) bandwidth as 0.1 and (2) number of contour levels as 10 (best view in color).

- *Edge dropout (denoted by ED)*: it drops out the edges in a graph with a dropout ratio.
- *Graph random walk (denoted by GRW)*: it essentially operates as the multi-layer edge dropouts.

We implement these structural manipulation strategies and show the comparison results in Table 8. We can clearly observe that: (1) Edge dropout (ED) generally achieves the most competitive performance among all structure augmentation strategies. (2) Our model BGCH+ incorporates dual feature augmentation in the embedding space for contrastive learning, further improving performance across all datasets compared to the ED variant. This highlights the effectiveness of our proposed approach. (3) To provide a more fine-grained comparison, we further combine these strategies. As shown in Table 8, we observe that variants with ED generally perform well, compared to the other; however, even with all these strategies integrated, it still consistently under-performs our model BGCH+. (4) Considering the heavy time cost of all these explicit structural manipulation, our feature-wise augmentation offers more flexibility when training BGCH+ in batch on the fly. This makes it better suited for handling larger bipartite graphs.

Regularization Effect of Representation Uniformity. Previous work [51] identifies that contrastive learning can help to improve the *uniformity* of image representations. To study its effect in learning hash codes, we visualize the

TABLE 8: Comparison of structure-manipulation-based variants and BGCH+ in terms of Recall@20.

	MovieLens	Gowalla	Pinterest	Yelp2018	AMZ-Book	Dianping
ND	22.80(-6.44%)	16.48(-4.13%)	14.91(-2.93%)	5.54(-7.05%)	3.73(-1.32%)	11.75(-2.49%)
ED	23.58(-3.24%)	16.88(-1.80%)	14.85(-3.32%)	5.51(-7.55%)	3.78 (0.00%)	11.89(-1.33%)
GRW	22.39(-8.12%)	16.14(-6.11%)	14.77(-3.84%)	5.60(-6.04%)	3.76(-0.53%)	11.84(-1.74%)
ND+ED	23.87(-2.05%)	17.08(-0.64%)	15.11(-1.63%)	5.82(-2.35%)	3.77(-0.26%)	11.98(-0.58%)
ND+GRW	22.97(-5.74%)	16.43(-4.42%)	14.98(-2.47%)	5.71(-4.19%)	3.72(-1.59%)	11.89(-1.33%)
ED+GRW	23.51(-3.53%)	16.92(-1.57%)	15.07(-1.89%)	5.79(-2.85%)	3.77(-0.26%)	11.94(-0.91%)
ND+ED+GRW	24.13(-0.98%)	17.06(-0.76%)	15.14(-1.43%)	5.87(-1.51%)	3.81(+0.79%)	12.01(-0.33%)
BGCH+	24.37	17.19	15.36	5.96	3.78	12.05

feature distributions of learned hash codes between BGCH and BGCH+ as follows. Firstly, we conduct the dimension reduction using T-SNE [52] to project the acquired representations into the 2-dimensional space. The projected representations are then normalized onto the unit hypersphere, ensuring a radius of 1. Next, we employ nonparametric Gaussian kernel density estimation (KDE) [53] to depict the distributions in Figure 5.

In Figure 5, the upper row presents BGCH with no dual feature contrastive learning, denoted by *w/o DFCL*; and the lower row corresponds to BGCH+ with the proposed module, i.e., *w/in DFCL*. We observe notably different feature distributions between the two scenarios. In the *w/o DFCL* scenario, the features tend to be highly clustered in specific areas. Besides, the *w/in DFCL* scenario however

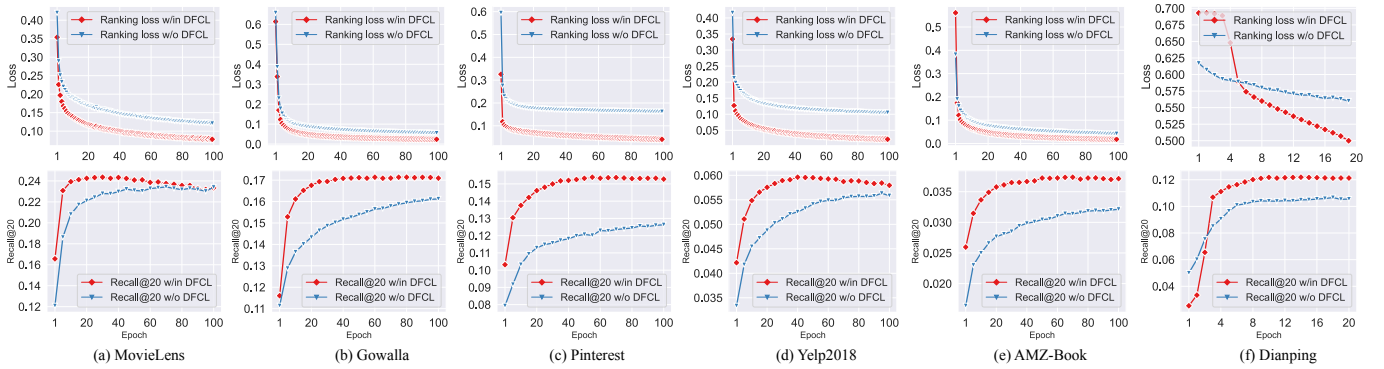


Fig. 6: Illustration of model convergence in terms of (1) ranking loss (upper row) and (2) Recall metric (lower row).

TABLE 9: Results of different feature augmentation strategies in terms of Recall@20.

	MovieLens	Gowalla	Pinterest	Yelp2018	AMZ-Book	Dianping
$w/o \mathcal{L}_{cl}^1$	24.23(-0.57%)	16.85(-1.98%)	14.73(-4.10%)	5.82(-2.35%)	3.67(-2.91%)	11.89(-1.33%)
$w/o \mathcal{L}_{cl}^2$	24.56(+0.78%)	16.88(-1.80%)	14.98(-2.47%)	5.93(-0.50%)	3.64(-3.70%)	11.94(-0.91%)
BGCH+	24.37	17.19	15.36	5.96	3.78	12.05

exhibits more uniform distributions, with features residing on wider arcs across all six datasets. The observed differences in feature distributions align with the performance improvements shown in Tables 6-7. This indicates that the uniformity of learned hashed codes plays a decisive role in the quality of graph hashing and ultimately impacts the prediction performance.

Effect of Dual Feature Contrastive Learning. To investigate the contribution of two feature augmentations, we set two variants via disabling the contrastive learning on different stages: (1) disabling contrastive learning on intermediate continuous embeddings, denoted as $w/o \mathcal{L}_{cl}^1$, and (2) disabling learning on output hash codes, denoted as $w/o \mathcal{L}_{cl}^2$. As presented in Table 9, we observe that both loss terms are jointly important for achieving desired performance.

However, for the MovieLens dataset, the variant $w/o \mathcal{L}_{cl}^2$ showed slightly better performance. This can be attributed to the high density of the MovieLens graph compared to other datasets. Specifically, let the density be defined by $\frac{|V_1| \times |V_2|}{|\mathcal{E}|}$. Sparse datasets with smaller densities, such as Gowalla (0.00084), Pinterest (0.00267), Yelp2018 (0.00130), and AMZ-Book (0.00062), are significantly influenced by our feature augmentation. On the other hand, denser datasets like MovieLens (0.04190) and Dianping (0.02210) have more graph edges for model training and are therefore less sensitive to the augmentations. These findings highlight the impact of our dual feature augmentations, with sparser graphs benefiting more from such techniques.

Convergence Analysis. Lastly, we discuss the model convergence speed over all datasets and made two key observations. We collect the metrics within the first 10% of the whole training epochs and depict results in Figure 6. Firstly, our proposed dual feature contrastive learning in BGCH+ contributes to a faster convergence to the ranking loss. Secondly, the faster convergence facilitated by our DFCL design enables BGCH+ to achieve its opti-

mal performance much earlier compared to BGCH without dual feature contrastive learning. For instance, in MovieLens dataset, BGCH+ already achieves its best performance with only about 20 epochs. Similarly, for other datasets, BGCH+ reaches its peak performance after approximately the 40th epoch, while BGCH without DFCL is still several epochs distant from its best performance. In summary, these observations demonstrate that our DFCL design in BGCH+ can effectively accelerate the model’s convergence, enabling BGCH+ to converge faster and achieve superior performance compared to its counterpart without the dual feature contrastive learning.

5.5 Ablation Study (RQ4)

We evaluate the necessity of model components with Top-20 search metrics and report the results in Table 10.

Effect of Adaptive Graph Convolutional Hashing. We study this model component by setting two variants, where: (1) $w/o AH-TA$ only disables the *topology-awareness of hashing* and uses the final encoder after all graph convolutions (similar to conventional approaches [13], [41]); (2) $w/o AH-RF$ removes the *rescaling factors*. The results from Table 10 results produce to the following observations:

- 1) The variant $w/o AH-TA$ underperforms BGCH+. This indicates that solely relying on the final output embeddings from the Graph Convolutional Network (GCN) framework may not adequately capture the unique latent node features necessary for effective hashing, especially considering the rich structural information present at different graph depths. In contrast, BGCH+ leverages intermediate information to enrich the representations, resulting in topology-aware hashing that effectively addresses the limited expressivity of discrete hash codes.
- 2) In addition to topology-aware hashing, the inclusion of *rescaling factors* (as introduced in Equation (5)) plays a crucial role in performance improvement. The removal of these factors from BGCH+ (variant $w/o AH-RF$) leads to significant performance decay. Although the computation of these factors is based on direct calculations and may not be theoretically optimal, they capture the numerical uniqueness of embeddings for subsequent hash encoding, which substantially enhances BGCH+’s prediction capability. The *determinacy* design of such factor computation is explored in detail in the following section.

TABLE 10: Ablation study.

Variant	MovieLens		Gowalla		Pinterest		Yelp2018		AMZ-Book		Dianping	
	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20
w/o <i>AH-TA</i>	21.23(-12.88%)	31.49(-15.37%)	15.91(-7.45%)	14.26(-16.61%)	13.77(-10.35%)	9.50(-15.25%)	5.12(-14.09%)	5.66(-8.27%)	2.98(-21.16%)	3.22(-25.98%)	11.02(-8.55%)	7.45(-15.44%)
w/o <i>AH-RF</i>	18.38(-24.58%)	28.13(-24.40%)	13.19(-23.27%)	12.77(-25.32%)	13.05(-15.04%)	9.11(-18.73%)	4.71(-20.97%)	4.97(-19.45%)	3.24(-14.29%)	3.82(-12.18%)	9.89(-17.93%)	7.48(-15.10%)
w/in <i>LF</i>	22.24(-8.74%)	35.66(-4.17%)	16.24(-5.53%)	16.17(-5.44%)	13.38(-12.89%)	9.55(-14.81%)	5.23(-12.25%)	5.53(-10.37%)	3.45(-8.73%)	3.89(-10.57%)	10.56(-12.37%)	7.91(-10.22%)
w/o \mathcal{L}_{bpr}	22.51(-7.63%)	35.31(-5.11%)	15.22(-11.46%)	15.94(-6.78%)	13.46(-12.37%)	9.92(-11.51%)	5.58(-6.38%)	5.72(-7.29%)	3.50(-7.41%)	3.93(-9.66%)	10.84(-10.04%)	7.76(-11.92%)
w/o \mathcal{L}_{cl}	22.43(-7.96%)	35.55(-4.46%)	14.76(-14.14%)	15.25(-10.82%)	13.79(-10.22%)	9.88(-11.86%)	5.66(-5.03%)	5.84(-5.35%)	3.20(-15.34%)	3.69(-15.17%)	11.33(-5.98%)	7.90(-10.33%)
BGCH+	24.37	37.21	17.19	17.10	15.36	11.21	5.96	6.17	3.78	4.35	12.05	8.81

Design of Learnable Rescaling. To learn the performance of *learnable rescaling factors*, we include another variant namely *w/in LF*. However, as shown in Table 10, the design of learnable rescaling factors in *w/in LF* does not achieve good performance as expected. One possible explanation for this outcome is that our current model does not impose strong mathematical constraints to the learnable factors (α_x), e.g., $\alpha_x^{(l)} = \text{argmin}(\mathbf{V}_x^{(l)}, \alpha_x^{(l)} \mathbf{Q}_x^{(l)})$, mainly because of its additional training complexity. Consequently, relying solely on stochastic optimization methods, such as stochastic gradient descent (SGD), may struggle to find the optimal values for these factors. Considering the additional search space introduced by the incorporation of learnable rescaling factors and the limitations of stochastic optimization, we argue that our deterministic rescaling method is a simple yet effective approach in practice. It strikes a balance between computational efficiency and performance, making it a preferable choice for our proposed model.

Effect of Multi-loss in Optimization. To investigate the impact of BPR loss \mathcal{L}_{bpr} and contrastive learning loss \mathcal{L}_{cl} , we set two variants, termed by *w/o \mathcal{L}_{bpr}* and *w/o \mathcal{L}_{cl}* , to optimize BGCH+ separately. These variants are applied while keeping all other model components intact. As shown in Table 10, partially using each one of \mathcal{L}_{bpr} and \mathcal{L}_{cl} can not yield the expected performance. This finding validates the effectiveness of our proposed multi-loss design. While \mathcal{L}_{bpr} guides the model to assign higher prediction values to observed edges, i.e., $\mathbf{Y}_{x,y} = 1$, than the unobserved node pair counterparts, \mathcal{L}_{cl} helps to alleviate the data sparsity issue and promotes the uniformity of output representations. helps address data sparsity issues and promotes the uniformity of output representations. By jointly optimizing these two loss functions, our model BGCH+ can learn high-quality binarized embeddings from \mathcal{L}_{cl} , and maintain rich relative order information regularized by \mathcal{L}_{bpr} accordingly. Hence, our multi-loss framework enables BGCH+ to achieve superior performance in terms of both representation quality and ranking capability.

5.6 Resource Consumption Analysis (RQ5)

Due to the various value ranges over all six datasets, we compactly report the value ratios of BGCH+ over the state-of-the-art hashing-based model HashGNN_s in Figure 7.

Model Training Time Cost. The training time cost, represented by the metric “*T-Time*” in Figure 7, reveals that training HashGNN_s is more time-consuming compared to our proposed model BGCH+. This difference can be attributed to the architectural disparities between the two models. HashGNN_s utilizes the earlier Graph Convolutional Network (GCN) framework [6] as the model back-

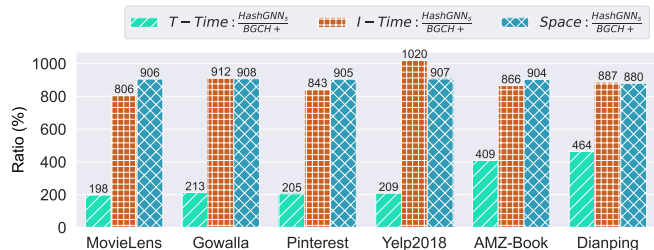


Fig. 7: Resource consumption ratios.

bone, which involves additional operations such as self-connection, feature transformation, and nonlinear activation. On the other hand, our model, BGCH+, follows the latest GCN framework [7] and eliminates these operations, resulting in reduced computational complexity and faster training. Furthermore, on the two largest datasets, AMZ-Book and Dianping, the training cost ratio becomes even more pronounced, reaching approximately 4 to 4.6 times higher for HashGNN_s compared to BGCH+. This is because of the need to decrease the batch size of HashGNN_s to ensure a manageable training process.

Online Inference Time Cost. We randomly generate 1,000 queries and evaluate the computation time cost. To ensure a fair comparison, we disable all parallel arithmetic techniques, such as MKL and BLAS, by using an open-source toolkit⁹. Indicated by “*I-Time*” in Figure 7, our model with Hamming distance matching generally achieves over 8× computation acceleration over HashGNN_s on all datasets. This is because, as we have explained in § 5.2, HashGNN_s randomly replaces the hash codes with their original continuous embeddings for relaxation purposes and relies on floating-point arithmetics, which sacrifices the computation acceleration provided by bitwise operations.

Hash Codes Memory Footprint. Embedding binarization can largely reduce memory space consumption. Compared to the state-of-the-art model HashGNN_s, our BGCH+ achieves about 9× of memory space reduction for the hash codes. As we have just explained, since HashGNN_s interprets hash codes with random real-value digits, it thus requires additional cost to distinguish binary digits from full-precision ones. In contrast, BGCH+ separates the storage of binarized encoding parts and corresponding rescaling factors. This separation allows for optimized space overhead and efficient storage of the binarized embeddings.

9. <https://www.lfd.uci.edu/~gohlke/pythonlibs/>

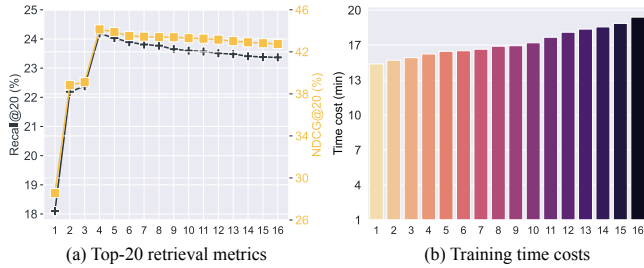
Fig. 8: Fourier Series decomposition term n in BGCH+.

TABLE 11: Gradient estimator comparison on Recall@20.

	MovieLens	Gowalla	Pinterest	Yelp2018	AMZ-Book	Dianping
STE	22.14(-9.15%)	15.34(-10.76%)	13.57(-11.65%)	5.42(-9.06%)	3.41(-9.79%)	11.39(-5.48%)
Tanh	22.66(-7.02%)	15.82(-7.97%)	14.45(-5.92%)	5.82(-2.35%)	3.43(-9.26%)	11.74(-2.57%)
SignSwish	23.14(-5.05%)	16.70(-2.85%)	14.52(-5.47%)	5.67(-4.87%)	3.42(-9.52%)	11.57(-3.98%)
Sigmoid	23.44(-3.82%)	15.89(-7.56%)	14.28(-7.03%)	5.79(-2.85%)	3.56(-5.82%)	11.89(-1.33%)
PBE	22.57(-7.39%)	16.27(-5.35%)	14.20(-7.55%)	5.48(-8.05%)	3.67(-2.91%)	11.55(-4.15%)
BGCH+	24.37	17.19	15.36	5.96	3.78	12.05

5.7 Study of Fourier Gradient Estimation (RQ6)

We take our largest dataset Dianping for illustration and the analysis can be generally popularized to the other datasets.

Effect of Decomposition Term n . We vary the decomposition term n from 1 to 16. As shown in Figure 8, we have two observations: (1) The choice of the decomposition term has a significant impact on the retrieval quality. Theoretically, larger values of n can provide more accurate gradient estimations. However, in practice, excessively large n may lead to overfitting. Therefore, it is advisable to choose a moderate value, such as $n = 4$ in Figure 8(a), to maximize model performance. (2) As we vary n from 1 to 16, the training time per iteration of BGCH+ gradually increased. This observation aligns with our complexity analysis in § 4, where we identified that the training cost is primarily associated with other modules like graph convolutional hashing, rather than the gradient estimation process.

Comparison with Other Gradient Estimators. We include several recent gradient estimators, i.e., *Tanh-like* [10], [28], *SignSwish* [29], *Sigmoid* [54], and *projected-based estimator* [55] (denoted as PBE). (1) The results summarized in Table 11 clearly demonstrate the superiority of our method over $\text{sign}(\cdot)$ function approximation in gradient estimation. As we have briefly explained, most existing estimators, which employ the *visually similar* function approximation to $\text{sign}(\cdot)$, generally provide better gradient estimation than Straight-Through Estimator (STE). (2) However, for bipartite graphs with high sparsity, e.g., Gowalla (0.00084) and AMZ-Book (0.00062), graph-based models may struggle to collect sufficient structural information for effective training of hash codes. With limited training samples, these *theoretically irrelevant* estimators may fail to rectify optimization deviations effectively, leading to noticeable performance gaps compared to our Fourier Series decomposition estimator.

6 RELATED WORK

Graph Convolution Network (GCN). Early research primarily studies the graph convolutions in the *spectral domain*, such as Laplacian eigen-decomposition [56] and Chebyshev

polynomials [57]. One major issue is that they usually suffer from high computationally expensive. To tackle this problem, *spatial-based* GCN models are proposed to re-define the graph convolution operations by aggregating the embeddings of neighbors to refine and update the target node’s embedding. Due to its scalability to large graphs, spatial-based GCN models are successfully applied in various applications [7], [6], [58], [59], [60], [61], [62]. Despite their effectiveness in embedding latent features for graph nodes, these models usually suffer from inference inefficiency due to the high computational cost associated with calculating similarities between continuous embeddings [13]. To address this issue, *learning to hash* has emerged as a viable solution.

Learning to Hash. Learning to hash models have shown great promise in achieving computational acceleration and storage reduction. By employing similarity-preserving hashing techniques, they can efficiently map high-dimensional dense vectors to a low-dimensional Hamming space, facilitating downstream tasks. Locality Sensitive Hashing (LSH) [40], [63] uses a series of hash projections to collect similar data points to the same or nearby “buckets” with high probability. More recent research has focused on integrating deep neural network architectures to improve the model performance [8]. This has led to a series of follow-up studies for various asks such as fast retrieval of images [10], [12], [41], documents [64], [65], [66], categorical information [67], and e-commerce products [68], [69].

To leverage hashing techniques with GCNs, the recent work HashGNN [13] investigates learning to hash for on-line matching and recommendation. Specifically, HashGNN combines the GraphSage [6] as the embedding encoder and applies learning to hash methods to obtain binary encodings. Its hash encoding process only proceeds at the end of multi-layer graph convolutions, i.e., using the aggregated output of GraphSage for embedding binarization. However, this fails to capture intermediate semantics from nodes’ different layers of receptive fields [22]. HashGNN utilizes the Straight-Through Estimator (STE) [70] to assume all gradients of $\text{sign}(\cdot)$ as 1 during backpropagation. But the integral of 1 is a certain linear function other than the $\text{sign}(\cdot)$, whereas this may lead to inconsistent optimization directions in the model training. To tackle these issues, our model BGCH+ is proposed.

Graph Contrastive Learning. Graph contrastive learning has recently emerged as a prominent research direction, drawing inspiration from the success of contrastive learning in visual representation tasks [16], [17]. In the context of graph data, contrastive learning typically requires the explicit application of data augmentation techniques. Traditionally, graph data augmentation methods have focused on manipulating the graph structures themselves, employing strategies such as node dropout, edge dropout, and graph random walk. However, these approaches may introduce biases that can affect the quality and integrity of the learned representations [71]. To mitigate these challenges, an alternative approach is to perform data augmentation in the feature space rather than directly modifying the input space [72]. This can be achieved through techniques

such as feature perturbation [71], [25]. By augmenting the feature representations, graph contrastive learning aims to maximize the agreement between two augmented views of the same graph in the latent space, while simultaneously ensuring the differentiation of representations for different nodes. Notably, recent studies have demonstrated the superior performance of graph contrastive learning in various graph-related tasks [73], [25], [18], [74], which motivates us to further study the problem of graph contrastive hashing.

7 CONCLUSION

In this paper, we revisit the learning to hash for efficient Hamming space search over graph structure data and propose BGCH+ for performance improvement. Compared to its predecessor, BGCH+ is further equipped with a novel dual feature contrastive learning paradigm, which operates on the latent features instead of the input graphs. Such design well enhances the robustness of learned hash codes against variations and thereby promotes the extraction of graph semantics in hash encoding. The empirical analyses over six real-world datasets demonstrate that the proposed method consistently outperforms existing hashing-based models while providing an alternative to full-precision models in scenarios with limited resources.

8 ACKNOWLEDGMENTS

The research presented in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF 2151185). Yixiang Fang was supported by NSFC under Grant 62102341, Guangdong Talent Program under Grant 2021QN02X826, and Shenzhen Science and Technology Program under Grants JCYJ20220530143602006, ZDSYS 20211021111415025. Chenhao Ma was supported by NSFC under Grant 62302421, Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280, and the Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen.

REFERENCES

- [1] C. Ma, L. Ma, Y. Zhang, R. Tang, X. Liu, and M. Coates, "Probabilistic metric learning with adaptive margin for top-k recommendation," in *SIGKDD*, 2020.
- [2] J. Zhang, X. Shi, S. Zhao, and I. King, "STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems," in *IJCAI*, 2019, pp. 4264–4270.
- [3] Y. Zhang and H. Zhu, "Doc2hash: Learning discrete latent variables for documents retrieval," in *NAACL*, 2019, pp. 2235–2240.
- [4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
- [5] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: Rating prediction with ratings and reviews," in *WWW*, 2018.
- [6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1025–1035.
- [7] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *SIGIR*, 2020, pp. 639–648.
- [8] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *TPAMI*, vol. 40, no. 4, pp. 769–790, 2017.
- [9] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2010.
- [10] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, "Forward and backward information retention for accurate bnns," in *CVPR*, 2020, pp. 2250–2259.
- [11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, vol. 9908, 2016, pp. 525–542.
- [12] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," 2017.
- [13] Q. Tan, N. Liu, X. Zhao, H. Yang, J. Zhou, and X. Hu, "Learning to hash with gnns for recsys," in *WWW*, 2020, pp. 1988–1998.
- [14] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. R. Lyu, and I. King, "Binarybert: Pushing the limit of BERT quantization," in *ACL/IJCNLP*, 2021, pp. 4334–4348.
- [15] Y. Chen, Y. Fang, Y. Zhang, and I. King, "Bipartite graph convolutional hashing for effective and efficient top-n search in hamming space," in *WWW*, 2023, pp. 3164–3172.
- [16] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020, pp. 9729–9738.
- [17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*. PMLR, 2020, pp. 1597–1607.
- [18] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *SIGIR*, 2021, pp. 726–735.
- [19] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *NeurIPS*, vol. 33, pp. 5812–5823, 2020.
- [20] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *WWW*, 2021, pp. 2069–2080.
- [21] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019.
- [22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.
- [23] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
- [24] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019.
- [25] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *SIGIR*, 2022, pp. 1294–1303.
- [26] J. Yu, X. Xia, T. Chen, L. Cui, N. Q. V. Hung, and H. Yin, "Xsimgl: Towards extremely simple graph contrastive learning for recommendation," *IEEE TKDE*, vol. 36, no. 2, pp. 913–926, 2023.
- [27] B. RUST, "Convergence of fourier series," 2013.
- [28] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *ICCV*, 2019, pp. 4852–4861.
- [29] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "BNN+: improved binary network training," *CoRR*, vol. abs/1812.11800, 2018.
- [30] Y. Xu, K. Han, C. Xu, Y. Tang, C. Xu, and Y. Wang, "Learning frequency domain approximation for bnns," *arXiv*, 2021.
- [31] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019, pp. 165–174.
- [32] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcn: Can gnns go as deep as cnns?" in *ICCV*, 2019, pp. 9267–9276.
- [33] Y. Chen, M. Yang, Y. Zhang, M. Zhao, Z. Meng, J. Hao, and I. King, "Modeling scale-free graphs with hyperbolic geometry for knowledge-aware recommendation," in *WSDM*, 2022, pp. 94–102.
- [34] Y. Chen, Y. Yang, Y. Wang, J. Bai, X. Song, and I. King, "Attentive knowledge-aware graph convolutional networks with collaborative guidance for personalized recommendation," in *ICDE*, 2022.
- [35] M. Yang, M. Zhou, J. Liu, D. Lian, and I. King, "Hrcf: Enhancing collaborative filtering via hyperbolic geometric regularization," in *WWW*, 2022, pp. 2462–2471.
- [36] X. Zhang, Y. Chen, C. Gao, Q. Liao, S. Zhao, and I. King, "Knowledge-aware neural networks with personalized feature referencing for cold-start recommendation," *arXiv*, 2022.
- [37] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *ICLR*, 2020, pp. 1001–1010.
- [38] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, "Modeling user exposure in recommendation," in *WWW*, 2016, pp. 951–961.
- [39] R. He and J. McAuley, "Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *WWW*, 2016, pp. 507–517.
- [40] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *PVLDB*, vol. 99, 1999, pp. 518–529.

- [41] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *JCCV*, 2017, pp. 5608–5617.
- [42] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.
- [43] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2017.
- [44] W.-C. Kang and J. McAuley, "Candidate generation with binary codes for large-scale top-n recommendation," in *CIKM*, 2019, pp. 1523–1532.
- [45] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
- [46] Y. Weiss and A. Torralba, "Spectral hashing," *NeurIPS*, 2008.
- [47] A. Joly and O. Buisson, "Random maximum margin hashing," in *CVPR*. IEEE, 2011, pp. 873–880.
- [48] D. Zhang, J. Wang, D. Cai, and J. Lu, "Laplacian co-hashing of terms and documents," in *ECIR*. Springer, 2010, pp. 577–580.
- [49] D. Kingma and J. Ba, "Method for stochastic optimization," 2015.
- [50] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and S. Y. Philip, "Graph self-supervised learning: A survey," *TKDE*, vol. 35, no. 6, pp. 5879–5900, 2022.
- [51] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *ICML*, 2020, pp. 9929–9939.
- [52] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *NeurIPS*, vol. 15, 2002.
- [53] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," 2010.
- [54] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X. Hua, "Quantization networks," in *CVPR*, 2019, pp. 7308–7316.
- [55] C. Liu, W. Ding, X. Xia, Y. Hu, B. Zhang, J. Liu, B. Zhuang, and G. Guo, "Rectified binary convolutional networks for enhancing the performance of 1-bit dcnn," *arXiv*, 2019.
- [56] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv*, 2013.
- [57] M. Defferrard, X. Bresson, and P. Vandergheynst, "Cnns on graphs with fast localized spectral filtering," *NeurIPS*, vol. 29, 2016.
- [58] Y. Chen, T. Truong, X. Shen, M. Wang, J. Li, J. Chan, and I. King, "Topological representation learning for e-commerce shopping behaviors," 2023.
- [59] Y. Chen, Y. Fang, Q. Wang, X. Cao, and I. King, "Deep structural knowledge exploitation and synergy for estimating node importance value on heterogeneous information networks," in *AAAI*, vol. 38, no. 8, 2024, pp. 8302–8310.
- [60] Y. Wu, Y. Chen, Z. Yin, W. Ding, and I. King, "A survey on graph embedding techniques for biomedical data: Methods and applications," *Information Fusion*, vol. 100, p. 101909, 2023.
- [61] Y. Chen, T. Truong, X. Shen, J. Li, and I. King, "Shopping trajectory representation learning with pre-training for e-commerce customer understanding and recommendation," in *SIGKDD*, 2024.
- [62] X. Zhang, Y. Chen, C. Ma, Y. Fang, and I. King, "Influential exemplar replay for incremental learning in recommender systems," in *AAAI*, vol. 38, no. 8, 2024, pp. 9368–9376.
- [63] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380–388.
- [64] H. Li, W. Liu, and H. Ji, "Two-stage hashing for fast document retrieval," in *ACL*, 2014, pp. 495–500.
- [65] Y. Chen, Y. Zhang, H. Guo, R. Tang, and I. King, "An effective post-training embedding binarization approach for fast online top-k passage matching," in *AAACL-IJCNLP*, 2022, pp. 102–108.
- [66] Z. Qiu, J. Liu, Y. Chen, and I. King, "Hihpq: Hierarchical hyperbolic product quantization for unsupervised image retrieval," *AAAI*, 2024.
- [67] W.-C. Kang, D. Z. Cheng, T. Yao, X. Yi, T. Chen, L. Hong, and E. H. Chi, "Learning to embed categorical features without embedding tables for recommendation," in *SIGKDD*, 2021, pp. 840–850.
- [68] Y. Zhang, D. Lian, and G. Yang, "Discrete personalized ranking for fast collaborative filtering from implicit feedback," in *AAAI*, vol. 31, no. 1, 2017.
- [69] Y. Chen, H. Guo, Y. Zhang, C. Ma, R. Tang, J. Li, and I. King, "Learning binarized graph representations with multi-faceted quantization reinforcement for top-k recommendation," in *SIGKDD*, 2022.
- [70] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv*, 2013.
- [71] Y. Zhang, Y. Chen, Z. Song, and I. King, "Contrastive cross-scale graph knowledge synergy," in *SIGKDD*, 2023, pp. 3422–3433.
- [72] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for nlp," *arXiv preprint arXiv:2105.03075*, 2021.
- [73] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *arXiv preprint arXiv:2006.04131*, 2020.
- [74] Y. Zhang, H. Zhu, Y. Chen, Z. Song, P. Koniusz, and I. King, "Mitigating the popularity bias of graph collaborative filtering: a dimensional collapse perspective," in *NeurIPS*, 2023, pp. 67533–67550.