arXiv:2304.00241v1 [cs.IR] 1 Apr 2023

面向汉明空间高效有效 Top-N 搜索的二部图卷积哈希

Yankai Chen[1], Yixiang Fang[2], Yifei Zhang[1], Irwin King[1]{ykchen, yfzhang, king}@cse.cuhk.edu.hk fangyixiang@cuhk.edu.cn[1]香港中文大学 [2]香港中文大学（深圳）

摘要

二部图上的搜索对于许多现实世界中的 Web 应用至关重要且用途广泛，例如在线推荐、数据库检索和查询-文档搜索。给定一个查询节点，传统方法依赖于在连续欧氏空间中使用向量化节点嵌入进行相似度匹配。为高效处理大量相似度计算，近期针对图结构数据开发哈希技术已成为一个新兴研究方向。尽管在汉明空间中检索效率较高，既有工作却面临灾难性的性能衰退。在本研究中，我们探讨了在二部图上利用图卷积网络进行哈希以实现有效 Top-N 搜索的问题。我们提出了一种端到端的二部图卷积哈希方法，即 BGCH，该方法由三个新颖且有效的模块组成：（1）自适应图卷积哈希，（2）潜在特征分散，以及（3）傅里叶序列化梯度估计。具体而言，前两个模块在哈希编码不可避免的信息损失面前，能够大幅保留结构信息；最后一个模块在频域中对哈希函数进行傅里叶级数分解，主要用于实现更准确的梯度估计。在六个真实世界数据集上开展的大量实验不仅表明，该方法的性能优于其他基于哈希的竞争方法，也证明了其中所有所提模型组件的有效性。

ACM计算分类系统概念

• **计算方法** → **学习潜在表示**；• **信息系统** → **信息检索**。

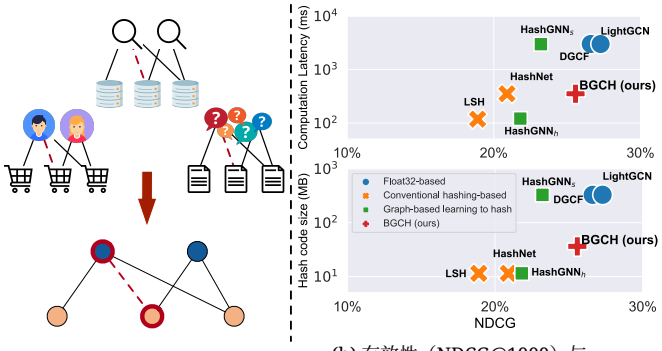
关键词

表示学习；学习哈希；图卷积网络；二部图；汉明空间搜索

ACM参考文献格式：

Yankai Chen，Yixiang Fang[2]，Yifei Zhang[1]，Irwin King[1]。2023。面向汉明空间中高效且有效 Top-N 搜索的二部图卷积哈希。载于《*2023*年*ACM Web*会议论文集》（*WWW '23*），*2023*年*5*月*1*日至*5*日，美国得克萨斯州奥斯汀。ACM，美国纽约州纽约市，11页。
https://doi.org/10.1145/3543507.3583219

允许免费为个人或课堂使用而制作本作品全部或部分内容的数字或纸质副本，但不得为营利或商业利益制作或分发副本，且副本必须保留本声明及首页上的完整引文。本作品中不属于作者所有的内容，其版权必须予以尊重。注明出处后可以摘录。若要以其他方式复制、再版、发布到服务器或分发至列表，须事先获得特别许可和/或支付费用。请向 permissions@acm.org 申请许可。*WWW '23*，*2023*年*5*月*1*日至*5*日，美国得克萨斯州奥斯汀 © 2023 版权所有，版权归所有者/作者所有。出版权已授权给 ACM。ACM 国际标准书号 978-1-4503-9416-1/23/04...$15.00
https://doi.org/10.1145/3543507.3583219



(a) 二部图建模　(b) 有效性（NDCG@1000）与匹配计算延迟（对数刻度）

图1：大众点评数据集上二部图建模及整体模型性能可视化示意图。

1 引言

二部图因便于对各种 Web 应用进行建模而在现实世界中无处不在，例如图1(a)所示的用户-产品推荐 [41, 63] 和在线查询-文档匹配 [66]。一项基础任务是*Top-N* 搜索，即为查询节点筛选出匹配度最高的 N 个图节点，例如在用户-产品图中向目标用户推荐最具吸引力的 Top-N 产品。随着近期机器学习研究的发展，学习向量化表示（也称为嵌入）如今已成为相似度匹配 [10, 16]的标准流程之一。在现有技术中，基于图的神经方法，即图卷积网络（GCN），近年来展现出卓越的模型性能 [17, 19]。由于能够捕获高阶连接信息，GCN 模型可以为节点嵌入注入语义信息。随后，基于学习得到的嵌入，在连续欧氏空间中穷举进行相似度估计。

除了嵌入的信息丰富度外，计算延迟和嵌入内存开销也是实际应用部署中的两项重要指标。随着数据量爆炸式增长，学习哈希 [25, 51] 最近为基于图的模型提供了一种优化模型可扩展性的替代方案。通常，这类方法会学习将连续值构成的向量列表转换为有限长度的二值化哈希码。与使用全精度[1] 嵌入不同，学习得到的哈希码有望在汉明空间中进行 Top-N 对象匹配与检索时，不仅缩减空间占用，还能加速计算。

尽管将图卷积网络与学习哈希相结合具有诸多优势，但简单地叠加这两种技术并不复杂，因此难以达到令人满意的性能，且存在若干不足：

[1]“全精度”通常指单精度和双精度。为便于说明，本文默认使用 float32。

• **粗粒度相似度测量**。 与连续嵌入相比，维度相同的哈希码在汉明空间中的编码排列有限（例如， 2^d 若维度为 d），因此表达能力较弱。这会导致对节点对相似度的估计较为粗略，从而因 Top-N 匹配不准确而显著降低性能。

• **特征侵蚀问题**。近期工作 [38, 44, 45, 49] 通常采用 sign(·) 函数进行 $O(1)$复杂度编码。然而，无论连续嵌入的每一位原本取何种具体值，通过将其转换到汉明空间，使用 sign(·) 进行哈希都不可避免地会削弱嵌入的特征信息。因此，这些学习得到的哈希嵌入中的潜在特征信息量会降低，而现有模型缺乏应对哈希中特征侵蚀的机制。

• **棘手的模型优化**。由于 sign(·) 在 0 处不可微，且在其他位置的导数均为 0，先前的模型通常使用视觉上相似但不一定具有理论相关性的函数（例如 tanh(·)）进行梯度估计。这可能导致模型训练中的优化方向不一致。此外，由于嵌入离散性，相关的损失景观[2] （图 2(a)）陡峭且起伏不平 [1], ，从而进一步增加了优化难度。

本文研究在二部图上使用图卷积网络（GCN）学习哈希，以实现高效的汉明空间 Top-N 搜索。我们提出了一种名为二部图卷积哈希 （BGCH）的模型，包含三个有效模块：（1）自适应图卷积哈希，（2）潜在特征分散，以及（3）傅里叶序列化梯度估计。前两个模块显著增强了学习得到的哈希码的信息量和表达能力，最后一个模块则为模型优化中的前向与反向传播提供了一套协调且易于处理的优化流程。具体而言：

• **自适应图卷积哈希**。第一个模块设计了一种拓扑感知卷积哈希，通过逐层哈希编码（从二部图的低阶到高阶子结构）对具有不同语义的节点特征进行连续二值化。为增强表达能力，该卷积哈希采用了一种有效的嵌入重缩放近似技术，同时不会降低汉明距离计算的效率。直观而言，这两种设计使学习得到的哈希码包含更丰富的信息、具有更强的表达能力，从而能够在汉明空间中保留细粒度相似度。

• **潜在特征扩散**。我们的第二个模块，即特征扩散，旨在弥补数值二值化造成的不可避免的信息损失。然而，在传统的连续嵌入中，主要特征往往集中在嵌入结构的一小块区域内。由于这些向量化的潜在特征难免会受到哈希离散性的平滑作用，因此，与其只分散到极少数嵌入维度，不如将这些关键特征扩散到大多数嵌入维度，以尽可能保留信息。为此，我们提出的模块旨在显式扩散信息丰富的潜在节点特征，使其在探索二部图时能够进一步传播到各个卷积层。

[2]有关可视化构建的详细信息见附录 A。

• **傅里叶序列化梯度估计**。 此外，为了提供准确的梯度估计，BGCH 提出在频域中使用傅里叶级数分解 sign(·) 函数。与现有的梯度估计器相比， [11, 15, 39, 44, 59],该估计器更贴合真实梯度的主要方向，从而实现前向与反向传播过程中协调且易于处理的模型优化。通过使用数量有限的分解项，BGCH 能够在可接受的训练成本内，为 sign(·) 提供更准确的梯度估计。

基于学习得到的哈希码，BGCH 在保持适度资源消耗的同时，显著提升了 Top-N 汉明空间检索的性能。图 1(b) 总结了质量与成本之间的权衡，并在一个包含逾 1000 万条观测边的真实世界二部图上，将 BGCH 与一系列具有代表性的方法（包括基于 *float32* 和基于哈希的方法）进行了比较（实验详情见 § 5.1）。如图 1(b) 右下角所示，右下角代表理想的最优性能；与现有全精度模型相比，BGCH 可实现超过 8× 的计算加速和空间缩减，同时比每一种基于哈希的方法都更有效（§ 5.2 和 § 5.3）。总而言之，我们的主要贡献如下：

• 我们研究了在二部图上使用图卷积网络进行学习哈希的问题。我们提出了一种新颖的方法 BGCH，该方法包含三个有效模块，可在汉明空间中实现高效且有效的 Top-N 搜索（§ 4）。

• 我们在六个真实世界数据集上开展了大量实验，以评估检索质量。同时，我们还从技术和实证角度深入分析了所有所提出模型组件的必要性（§ 5）。

• 我们从理论上证明了模型的有效性，并对时间和空间开销进行了复杂度分析（附录 C）。

2 相关工作

图卷积网络（GCN）。早期研究主要关注谱域中的图卷积，例如拉普拉斯特征分解 [3] 和切比雪夫多项式 [12]。其中一个主要问题是，这些模型通常计算成本高昂。为了解决这一问题，研究人员提出了基于空间的 GCN 模型，通过聚合邻居的嵌入来重新定义图卷积操作，从而优化和更新目标节点的嵌入。由于具有良好的大型图扩展性，基于空间的 GCN 模型被广泛应用于各种应用场景 [17, 19, 52]。例如，为了捕获高阶结构信息，NGCF [52] 和 LightGCN [19] 在二部交互图上学习协同过滤信号，以用于推荐。尽管这些模型能够有效地嵌入图节点的潜在特征，但由于连续嵌入之间相似度计算的高计算成本 [49]，它们通常会受到推理低效问题的影响。为了解决这一问题，学习哈希 提供了可行性。

学习哈希。学习哈希模型有望为通用信息检索和处理任务实现计算加速并减少存储开销 [5, 13, 19, 22, 23, 36, 40]。除了减少冲突 [33], ，相似性保持哈希还将高维稠密向量映射到低维汉明空间，以便高效处理下游任务。一个代表性模型是局部敏感哈希（LSH） [14] ，它使用

随机投影作为哈希函数。近期研究着重于集成深度神经网络架构以改进模型 [51]。这些研究启发了一系列针对不同任务的后续工作，例如快速检索图像 [4, 38, 44], 、文档 [9, 35], 、类别信息 [27],和电子商务产品 [6, 65]。

为将哈希技术与图卷积网络结合，当前最先进的方法 HashGNN [49] 研究了面向在线匹配和推荐的学习哈希。具体而言，HashGNN 依次将 GraphSage [17] 作为嵌入编码器，并结合学习哈希方法，随后获得相应的二进制编码。其哈希编码过程仅在多层图卷积结束时进行，即使用 GraphSage 的聚合输出进行表示二值化。然而，这种方法未能捕获节点不同感受野层级中的中间语义 [30]。HashGNN 的另一个问题是使用直通估计器（*STE*） [2]，在反向传播时假设 sign(·) 的所有梯度均为 1。然而，1 的积分是某个线性函数，而非 sign(·)，这可能导致模型训练中的优化方向不一致。为解决这些问题，我们提出了 BGCH 模型，并在 § 5 中论证其有效性。

3 预备知识与问题表述

图卷积网络（GCN）。GCN 的基本思想是通过图拓扑 [19, 30, 56] 迭代传播并聚合节点邻居的潜在特征，从而学习节点嵌入：

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

其中，$\boldsymbol{V}_{(l)x} \in \mathbb{R}^d$表示节点 x经过第 l轮图卷积后的嵌入，该嵌入在嵌入矩阵 $\boldsymbol{V}$中索引。$\mathcal{N}(x)$是 x的邻居集合。函数 $AGG(\cdot,\cdot)$是信息聚合函数；以往工作中有多种实现方式 [17, 30, 50, 57],，其主要目的是变换中心节点特征和邻居特征。本文采用当前最先进模型 LightGCN [19]中的图卷积范式。

二部图与邻接矩阵。二部图记为 $G = \{V_1, V_2, E\}$，其中 V_1 和 V_2 是两个不相交的节点集合，E 是 V_1 和 V_2中节点之间的边集合。我们可以用 $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{V}_1| \times |\mathcal{V}^2|}$表示边交易，其中取值为 1 的元素，即 $Y_{x,y} = 1$，表示节点 $x \in V_1$与 $y \in V_2$之间存在一条观测边，否则为 $Y_{x,y} = 0$。于是，整个图的邻接矩阵 $\boldsymbol{A}$可定义如下：

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{Y} \\ \boldsymbol{Y}^T & \boldsymbol{0} \end{bmatrix}. \tag{2}$$

问题定义。给定一个二部图 $G = \{V_1, V_2, E\}$及其邻接矩阵 $\boldsymbol{A}$，我们致力于学习一个哈希函数：

$$F(\boldsymbol{A}|\Theta) \to \boldsymbol{Q}, \tag{3}$$

其中，Θ是所有可学习参数的集合。给定二部图中的两个节点，例如 $x \in V_1$和 $y \in V_2$，它们的哈希码分别为 Q_x和 Q_y。随后，可通过哈希码 Q_x和 Q_y有效且高效地衡量节点$x \in V_1$与 $y \in V_2$ 之间存在边的概率$\widehat{Y}_{x,\,y}$，即$\widehat{Y}_{x,\,y} = f(Q_x,\ Q_y)$，其中 f是评分函数。直观而言，$\widehat{Y}_{x,\,y}$的值越大，x与 y匹配的可能性就越高，也就是说，x与 y之间存在一条边。本文所用关键符号的说明见附录 B。

4 BGCH：方法

4.1 概述

我们正式介绍 BGCH 模型。注意，由于特征扩散模块的作用会随着卷积哈希传播，因此我们按以下顺序介绍这些模块：(1) 潜在特征分散 (§ 4.2)旨在将嵌入特征分散到更广泛的嵌入结构中，以弥补哈希过程中不可避免的信息损失；(2) 自适应图卷积哈希 (§ 4.3)提供了一种有效的编码方法，可显著提升哈希特征表达能力，同时保持汉明空间中的匹配效率；(3) 傅里叶序列化梯度估计 (§ 4.4)在频域中对 sign(·)进行傅里叶级数分解，从而提供更准确的梯度近似。基于学习得到的哈希码，BGCH 通过汉明距离度量实现高效的在线匹配(§ 4.5)。模型示意图见图 2(b)。

4.2 潜在特征扩散

为了解决特征侵蚀问题，我们尝试扩散嵌入特征，以此作为一种有效策略，来弥补数值二值化不可避免造成的信息损失。从奇异值分解（SVD）的角度来看，奇异值及其对应的奇异向量可以重构原始矩阵；通常，较大的奇异值可理解为与矩阵 [54] 的主要特征结构相关。由于我们希望避免将信息丰富的特征压缩并聚集到（相对较小的）嵌入子结构中，因此，通过处理这些奇异值来实现目标是自然的选择。基于这一思路，我们旨在对奇异值进行归一化，使其各自对构成潜在特征的贡献趋于均衡。为此，幂归一化 [31, 68] 是解决特征不平衡 [32]等相关问题的方法之一。受近期近似方法 [61], 的启发，我们接下来介绍一种轻量级的图卷积特征扩散技术，具体如下。

具体而言，令 $\boldsymbol{I}$ 表示单位矩阵，我们首先生成一个标准正态随机向量 $\boldsymbol{p} \sim \boldsymbol{0}\,\boldsymbol{I}$ $\boldsymbol{p} \in {}_{(0)}\mathrm{N}(,)$, $^{\mathbb{R}^c}$其中$_{(0)}$。基于嵌入矩阵进行特征扩散，例如，令 $\boldsymbol{V} = \boldsymbol{V}_{(0)}$，我们通过迭代执行 $\boldsymbol{p}_{(k)} = \boldsymbol{V}_{\mathrm{T}}\boldsymbol{V}\boldsymbol{p}_{(k-1)}$来计算所需的**离散向量** $\boldsymbol{p}_{(k)}$。生成离散向量的迭代过程独立于图卷积迭代[3]。我们通过以下方式得到 $\boldsymbol{p}_{(K)}$ 的投影矩阵 $\boldsymbol{P}$：

$$\boldsymbol{P} = \frac{\boldsymbol{p}^{(K)}\boldsymbol{p}^{(K)^{\mathrm{T}}}}{||\boldsymbol{p}^{(K)}||_2^2}. \tag{4}$$

然后，我们得到带有超参数 $\epsilon \in (0,1)$ 的特征扩散表示矩阵如下：

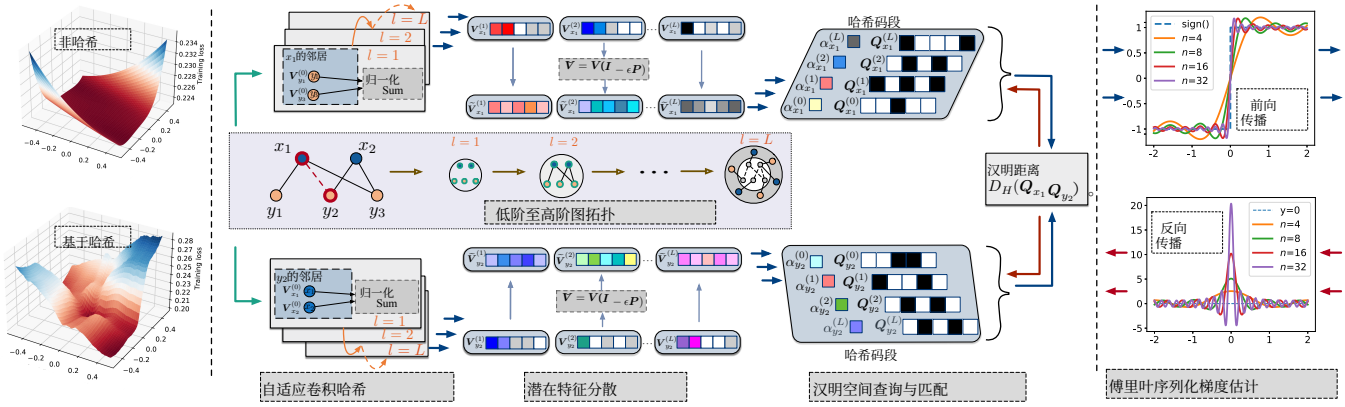
$$\widetilde{\boldsymbol{V}} = \boldsymbol{V}(\boldsymbol{I} - \epsilon\boldsymbol{P}). \tag{5}$$

~因此，结合扩散矩阵 $\boldsymbol{V}$，我们得到**特征扩散**图卷积如下：

$$\widetilde{\boldsymbol{V}}^{(l+1)} = (\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})\widetilde{\boldsymbol{V}}^{(l)}, \text{ where } \widetilde{\boldsymbol{V}}^{(0)} = \boldsymbol{V}^{(0)}(\boldsymbol{I} - \epsilon\boldsymbol{P}). \tag{6}$$

请注意，我们仅在初始步骤，即 $\widetilde{\boldsymbol{V}}_{(0)}$，显式执行一次此特征扩散操作；更重要的是，这种特征扩散可以通过多层图

[3]在我们的工作中，我们将 $K \le L$ 设置为，主要是为了确保离散向量生成所涉及的复杂度上界不超过图卷积的复杂度。



(a) 损失景观可视化　(b) BGCH 框架示意图　(c) 梯度估计

图 2：(a) 可视化损失景观对比；(b) BGCH模型框架（彩色效果最佳）；(c) 前向传播和反向传播中的傅里叶序列化梯度估计。

卷积从 0 扩散到 L。与未经处理的嵌入对应项（例如 $V_{(l)}$）相比，每一层的嵌入矩阵 $\widetilde{V}_{(l)}$ 都呈现出扩散后的特征结构，其奇异值的分布在预期上更加均衡。我们对此作如下形式化说明：

定理 *1*（特征扩散）。设 $V(l) = U_1 \Sigma U T2$，其中 U_1和 U_2 为酉矩阵，降序奇异值矩阵 $\Sigma = diag(\sigma_1, \sigma_2, \cdots \ \sigma_c)$。则 $\mathbb{E}(\widetilde{V}(l))$ $= U_1 \Sigma \Sigma_\mu U T2$，其中 $\Sigma_\mu = diag(\mu_1, \mu_2, \cdots \ \mu_c)^{0<\mu_{1\cdots c}<1}$ 按升序排列。

直观而言，在给定相同正交归一基的情况下，与 $V_{(l)}$ 相比，当信息特征分散在更大的矩阵子结构中时，重构 $\widetilde{V}^{(l)}$ 的期望难度更高。这最终能够在数值二值化过程中弥补信息损失。我们在附录 C 中给出了定理证明，并将在 § 5.4 中评估该模块的有效性。

4.3 自适应图卷积哈希

一种提升表达能力并平滑损失景观的可行方案是采用松弛策略。因此，除了使用 sign(·) 进行拓扑感知的嵌入二值化之外：

$$Q_x^{(l)} = \text{sign}(\widetilde{V}_x^{(l)}), \tag{7}$$

我们的模型 BGCH 还会为每个节点逐层计算一个正的重缩放因子，例如 $\alpha_{(l)x} \in \mathbb{R}^+$，使得 $\widetilde{V}_{(l)x} \approx \alpha_{(l)x} Q_{(l)x}$。在这项工作中，我们提出了一种简单而有效的方法，直接计算重缩放因子，具体如下：

$$\alpha_x^{(l)} = \frac{1}{d} ||\widetilde{V}_x^{(l)}||_1. \tag{8}$$

与将这些因子设为可学习参数不同，这种确定性计算在实现针对不同图节点的自适应近似功能的同时，大幅缩减了参数搜索空间。我们在实验的 § 5.4 中对此进行了展示。

经过 L 轮特征传播和哈希后，我们得到**自适应哈希码**表 Q $= \ \boldsymbol{\alpha} \ Q \quad \boldsymbol{\alpha} \in \{, \}$，其中 $\mathbb{R}^{|\ _1|+|\ _2| \times L+1} \quad |\ _1|+|\ _2| \times d$ $(\mathcal{V}\mathcal{V})()$ 且 $Q \in \mathbb{R}^{(\mathcal{V}\mathcal{V})}\}$。对于每个节点，其对应的哈希码经过索引并组合：

$$\boldsymbol{\alpha}_x = \alpha_x^{(0)} || \alpha_x^{(1)} || \cdots || \alpha_x^{(L)}, \text{ and } \boldsymbol{Q}_x = Q_x^{(0)} || Q_x^{(1)} || \cdots || Q_x^{(L)}. \tag{9}$$

直观而言，哈希码表 Q 表示在不同迭代步骤中来回传播的二部结构信息 l，即从 0 到最大步骤 L。它不仅追踪了所有图节点经哈希编码后的中间知识，还保留了对其原始连续嵌入的近似值，例如 $\widetilde{v}_x^{(l)}$。此外，尽管空间开销略有增加（复杂度分析见附录 C），这种分离式哈希编码方法仍支持位运算（§ 4.5），从而加速推理和匹配。

4.4 傅里叶级数展开的梯度估计

为了对哈希函数 sign(·) 提供相应的梯度估计，我们在频域中引入其傅里叶级数分解来近似该函数。具体而言，sign(·) 可视为周期方波函数 $t(x)$ 在长度 $2H$ 内的一个特例，即 $\text{sign}(\phi) = t(\phi)$，$|\phi| < H$。由于 $t(x)$ 可以展开为傅里叶级数，因此我们有：

$$\text{sign}(\phi) = \frac{4}{\pi} \sum_{i=1,3,5,\cdots}^{+\infty} \frac{1}{i} \sin(\frac{\pi i \phi}{H}), \text{ where } |\phi| < H. \tag{10}$$

sign( ) 的无限项傅里叶级数分解是一种无损变换 [46]。因此，如图 2(c) 所示，我们可以设置有限的展开项 n，得到其近似形式如下：

$$\text{sign}(\phi) \doteq \frac{4}{\pi} \sum_{i=1,3,5,\cdots}^{n} \frac{1}{i} \sin(\frac{\pi i \phi}{H}). \tag{11}$$

相应的导数可据此推导如下：

$$\frac{\partial \text{sign}(\phi)}{\partial \phi} \doteq \frac{4}{H} \sum_{i=1,3,5,\cdots}^{n} \cos(\frac{\pi i \phi}{H}). \tag{12}$$

与双曲正切类函数等其他梯度估计器不同，使用傅里叶级数逼近 sign(·) 函数，不会在模型优化过程中扭曲事实梯度的主要方向 [58]。这有助于实现从连续值到节点表示对应二值化的协调转换，从而显著保留二值化表示的可辨别性，并

相应地获得更高的检索准确率。我们在实验的 § 5.6 中展示了这一性能比较。总而言之，如公式 (13) 所示，为了学习和优化图节点的二值化嵌入，我们在前向传播中使用严格的 sign(·) 函数，并使用 $\partial \text{sign}(\phi)/\partial\phi$ 估计反向传播的梯度。

$$\begin{cases} Q^{(l)} = \text{sign}(\phi), & \text{Forward propagation.} \\ \dfrac{\partial Q^{(l)}}{\partial \phi} \doteq \dfrac{4}{H} \displaystyle\sum_{i=1,3,5,\cdots}^{n} \cos\left(\dfrac{\pi i \phi}{H}\right). & \text{Backward propagation.} \end{cases} \tag{13}$$

4.5 评分预测与模型优化

4.5.1 匹配分数预测。 给定两个节点 $x \in V_1$ 和 $y \in V_2$，实现评分函数的一种自然方式是内积，主要由于其简单性，如下所示：

$$\widehat{Y}_{x,y} = (\alpha_x Q_x)^{\mathsf{T}} \cdot (\alpha_y Q_y). \tag{14}$$

然而，公式（14）中的内积仍然是在（连续的）欧几里得空间中使用全精度运算进行的。为了建立内积与汉明距离度量之间的联系，我们引入定理 2 如下：

定理 2（**汉明距离匹配**）。给定两个哈希码，有 $(\alpha_x Q_x)^{\mathsf{T}} \cdot (\alpha_y Q_y) = \alpha_x \alpha_y \left(d - 2D_H(Q_x, Q_y)\right)$。

$D_H(\cdot,\cdot)$ 表示两个输入之间的汉明距离。根据定理 2，我们将分数计算转换为汉明距离匹配。这样，我们可以将原始分数计算公式（公式 (14)）中的大部分浮点运算次数（#FLOPs）减少为高效的汉明距离匹配。这可以实现显著的计算加速，具体分析见附录 C。

4.5.2 多损失目标函数。 我们的目标函数由两个部分组成，即图重构损失 L_{rec} 和 BPR 损失 L_{bpr}。通常，这两个损失函数对彼此具有正则化效应。这种设计的直观理解是：

• L_{rec} 重构观测到的二部图拓扑；• L_{bpr} 对根据哈希码计算得到的匹配分数进行排序。

具体而言，我们使用交叉熵损失实现 L_{rec}：

$$\mathcal{L}_{rec} = \sum_{x \in \mathcal{V}_1} \Big(\sum_{y \in \mathcal{N}(x)} \ln \sigma\Big((V_x^{(0)})^{\mathsf{T}} \cdot V_y^{(0)}\Big) + \sum_{y' \notin \mathcal{N}(x)} \ln\Big(1 - \sigma((V_x^{(0)})^{\mathsf{T}} \cdot V_{y'}^{(0)})\Big)\Big), \tag{15}$$

其中，σ 是激活函数，例如 Sigmoid。L_{rec} 基于图卷积之前的初始连续嵌入，例如 $V_{(0)x}$，为拓扑重构提供最基础的信息。对于 L_{bpr}，我们采用贝叶斯个性化排序（BPR）损失，具体如下：

$$\mathcal{L}_{bpr} = -\sum_{x \in \mathcal{V}_1} \sum_{\substack{y \in \mathcal{N}(x) \\ y' \notin \mathcal{N}(x)}} \ln \sigma(\widehat{Y}_{x,y} - \widehat{Y}_{x,y'}). \tag{16}$$

L_{bpr} 鼓励观测边的预测得分高于其未观测对应项 [19]。令 Θ 表示由参数 λ_2 正则化、以避免过拟合的可训练嵌入集合。我们的最终目标函数最终定义如下：

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_1 \mathcal{L}_{bpr} + \lambda_2 ||\Theta||_2^2. \tag{17}$$

表 1：数据集统计信息。

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

至此，我们已介绍 BGCH 的所有技术部分，并在附录 B 中附上伪代码。我们在附录 C 中给出了所有定理证明和复杂度分析。

5 实验评估

我们评估 BGCH，以回答以下研究问题：

• **研究问题 1.** BGCH 在 Top-N 汉明空间检索中的表现如何，与最先进的基于哈希模型相比如何?

• **研究问题 2.** 就长列表检索质量而言，BGCH 与全精度模型之间的性能差距有多大?

• **研究问题3**。 BGCH中所提出的组件有哪些优势?

• **研究问题4**。 BGCH的实际资源消耗是多少？ • **研究问题5**。傅里叶级数分解在关于检索准确率和训练效率方面表现如何?

5.1 实验设置

数据集和评估指标。 我们在表 1 [7, 8, 19, 52, 60, 中纳入了六个真实世界的二部图，这些图得到了广泛评估。我们采用 Recall@N 和 NDCG@N 评估协议来衡量 Top-N 汉明空间排序能力。数据集详情和评估流程见附录 D。

基线方法。 我们纳入了以下具有代表性的基于哈希的模型：（1）用于一般对象检索的模型（LSH [14]）；（2）用于图像搜索的模型（HashNet [4]）；以及（3）用于推荐的 Top-N候选生成模型（Hash_Gumbel [24, 42], CIGAR [28] 和 HashGNN [49]）。我们还纳入了若干当前最先进的全精度[4] 推荐模型，即 NeurCF [20], NGCF [52], DGCF [53], LightGCN [19],，用于比较长列表排序质量。模型介绍见附录 D。早期的哈希方法，例如 SH [55], RMMH [26], LCH [62],，主要由于上述竞争模型 [4, 28] 已经验证了相对于它们的性能优势，因此不再纳入。

5.2 汉明空间 Top-N 查询（RQ1）

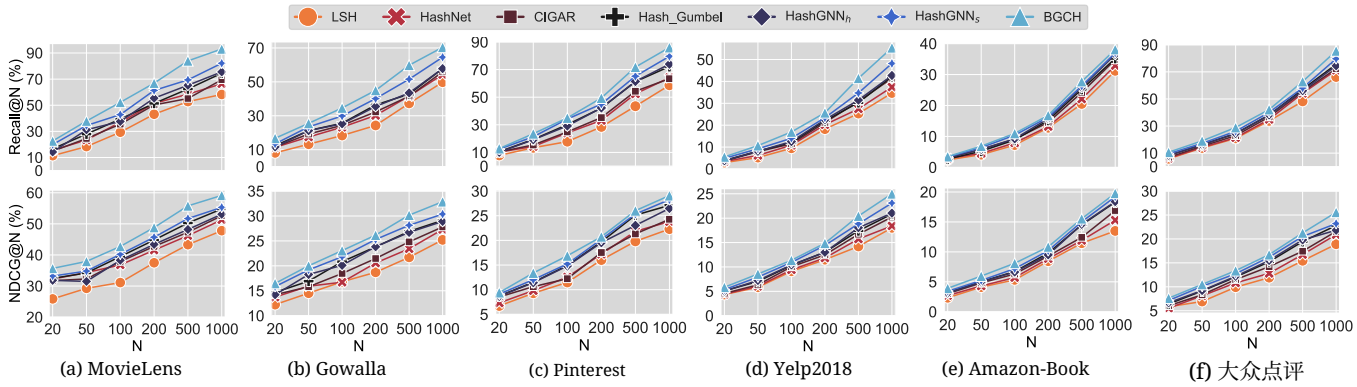
为评估**由粗到细** Top-N 排序能力，我们设置 N=1000。我们首先在表 2中报告 Top-1000 搜索的 Recall@20_{1000} 和 NDCG@20_{1000}[5] 结果，然后在图3中绘制 Top-1000 搜索中 {20, 50, 100,200, 500, 1000} 对应的整体召回率和 NDCG 指标曲线。为公平比较，我们将 BGCH 和基线方法的卷积迭代次数设为 2，嵌入维度设为 256。

• **实验结果表明，BGCH模型优于先前的基于哈希的模型**。(1) 如表2所示，当前最先进的模型HashGNN优于传统的基于哈希的基线方法，例如LSH、HashNet和CIGAR。这表明，与基于图的模型相比，直接改编传统的（即非基于图的）哈希方法可能

[4]由于实验中使用 float32 实现这些方法，因此将其记为 FT32。[5]若不存在歧义，则使用简写 Recall@20、NDCG@20。

表 2：Top-1000检索中 Recall@20 和 NDCG@20 的结果：(1)"R"和"N"分别表示召回率和 NGCG；(2)粗体表示 BGCH，下划线表示性能最佳的模型；(3) 标记 * 表示 Wilcoxon 符号秩检验表明，与次优模型相比，模型取得了统计显著的提升，置信水平超过 95% 的情形。

数据集	MovieLens (%)		Gowalla (%)		Pinterest (%)		Yelp2018 (%)		AMZ-Book (%)		大众点评 (%)	
指标	$R@20_{1000}$	$N@20_{1000}$	$R@20_{1000}$	$N@20_{1000}$	$R@20_{1000}$	$N@20_{1000}$	$R@20_{1000}$	$N@20_{1000}$	$R@20_{1000}$	$N@20_{1000}$	$R@20_{1000}$	$N@20_{1000}$
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分布	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	**22.86***	**36.26***	**16.73***	**16.48***	**12.78***	**9.42***	**5.51***	**5.84***	**3.48***	**3.92***	**10.66***	**7.63***
增益百分比	15.05%	9.18%	24.39%	10.83%	3.23%	3.40%	13.37%	9.36%	4.19%	13.62%	11.39%	7.01%



(a) MovieLens (b) Gowalla (c) Pinterest (d) Yelp2018 (e) Amazon-Book (f) 大众点评

图3：N取值为{20, 50, 100, 200, 500, 1000}时的Top-N 检索质量（彩色显示效果最佳）。

难以取得可比的性能，主要原因在于，图卷积架构能够有效捕获二部图拓扑中的潜在信息，从而为哈希编码做好准备。(2) 得益于我们提出的模型组件，例如自适应图卷积哈希，BGCH在所有数据集上始终优于HashGNN，在Recall@20和NDCG@20方面分别提升3.23%~24.39%和3.40%~ 13.62%。(3) 此外，我们对BGCH进行了Wilcoxon符号秩检验。结果验证了，在95%的置信水平下，BGCH相较于次优模型的所有改进均具有统计显著性。(4) 为解释这些结果，我们提出的拓扑感知图卷积哈希方法有效丰富了图节点嵌入。我们提出的特征扩散进一步缓解了数值二值化导致的特征侵蚀问题。最后但同样重要的是，我们提出的傅里叶序列化梯度估计对于为模型优化提供准确梯度也至关重要。我们将在§ 5.4中进行消融研究。

• **通过将 N 从 20 调整到 1000，与基线方法相比，BGCH 始终展现出有竞争力的性能**。Recall@N 表示 Top-N 检索结果中相关对象所占的比例，而 NDCG@N 衡量相对排序的能力。如图3所示：（1）与 HashGNN 的近似版本 $HashGNN_s$相比，在全部六个基准数据集上，当 N 从 20 变化到 1000 时，BGCH 的召回率和 NDCG 指标通常都取得了稳定且显著的提升。（2）

表 3：基于 float32 的模型的 NDCG@1000 结果。

	Movie	Gowalla	Pinterest	Yelp2018	AMZ-Book	大众点评
NeurCF	58.76	32.07	28.79	24.69	19.83	25.54
NGCF	60.28	32.13	29.78	25.23	20.37	25.76
DGCF	62.41	34.97	31.47	26.28	21.74	26.87
LightGCN	62.88	35.26	31.32	26.55	21.92	27.28
BGCH	**59.16**	**32.87**	**29.09**	**25.01**	**19.79**	**25.57**
% 容量	94.08%	93.22%	92.44%	94.20%	90.28%	93.73%

除了更高的检索质量外，相较于 $HashGNN_s$，BGCH 的另一项优势是仍支持位运算，即汉明距离匹配，从而实现推理加速。这是因为，为了提高预测准确率，$HashGNN_s$ 采用伯努利随机变量，提供以原始连续值替换哈希码中特定数位的概率，因此无法进行按位计算。正如我们在 § 5.5 中所示，与 $HashGNN_s$相比，BGCH 实现了超过 8× 的推理加速，这对于基于查询的在线匹配和检索应用尤其有前景。

5.3 与基于 FT32 的模型进行比较（RQ2）

在本节中，我们还将 BGCH 与若干全精度（基于 FT32）模型进行比较，以评估长列表搜索质量。从表 3 中可以看出，我们得出以下分析。（1）我们注意到，在所有数据集上，我们的模型 BGCH 整体表现与较早的全精度模型（如 NeurCF 和 NGCF）相比具有竞争力。至于最先进的模型 LightGCN，我们的模型

表 4：消融研究。

变体	MovieLens		Gowalla		Pinterest		Yelp2018		AMZ-Book		大众点评	
	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 FD	22.82 (-0.17%)	35.87 (-1.08%)	15.92 (-4.84%)	15.79 (-4.19%)	12.25 (-4.15%)	9.07 (-3.72%)	5.16 (-6.35%)	5.49 (-2.49%)	3.26 (-6.32%)	3.57 (-8.93%)	10.46 (-1.88%)	7.50 (-1.70%)
w/o AH-TA	19.54(-14.52%)	29.17(-19.55%)	13.49(-19.37%)	12.38(-24.88%)	12.24 (-4.23%)	8.86 (-5.94%)	4.77(-13.43%)	5.18(-11.30%)	2.49(-28.45%)	2.86(-27.04%)	9.83 (-7.79%)	6.87 (-9.96%)
w/o AH-RF	16.73(-26.82%)	26.97(-25.62%)	11.24(-32.82%)	11.29(-31.49%)	10.18(-20.34%)	7.33(-22.19%)	3.76(-31.76%)	4.30(-26.37%)	3.27 (-6.03%)	3.64 (-7.14%)	8.33(-21.86%)	6.93(-9.17%)
w/in LF	21.06 (-7.87%)	34.59 (-4.61%)	15.48 (-7.47%)	15.38 (-6.67%)	11.94 (-6.57%)	8.89 (-5.63%)	4.86(-11.80%)	5.17(-11.47%)	3.14 (-9.77%)	3.62 (-7.65%)	9.40(-11.82%)	7.27(-4.72%)
去除 $\mathcal{L}^{bpr}$	21.42 (-6.30%)	34.83 (-3.94%)	15.87 (-5.14%)	15.66 (-4.98%)	12.33 (-3.52%)	9.17 (-2.65%)	5.31 (-3.63%)	5.61 (-3.94%)	3.35 (-3.74%)	3.77 (-3.83%)	10.21 (-4.22%)	7.38 (-3.28%)
不含 $\mathcal{L}_{rec}$	17.01(-25.59%)	27.16(-25.10%)	12.27(-26.66%)	12.63(-23.36%)	10.81(-15.41%)	7.86(-16.56%)	3.93(-28.68%)	4.37(-25.17%)	3.19 (-8.33%)	3.73 (-4.85%)	8.82(-17.26%)	7.26(-4.85%)
BGCH	**22.86**	**36.26**	**16.73**	**16.48**	**12.78**	**9.42**	**5.51**	**5.84**	**3.48**	**3.92**	**10.66**	**7.63**

通常能够达到其 90% 以上的 Top-1000 排名能力。（2）BGCH 的表现证明了它在保证长列表 Top-N 检索质量方面的有效性。这对某些工业应用很有用，例如推荐系统，它们通常包含两个主要阶段：候选生成 和重排序。因此，显然，候选生成的高质量会直接降低下一阶段重排序的复杂度，因为搜索空间得到了大幅缩减。（3）考虑到这些学习得到的哈希码在汉明空间检索方面的效率 以及更低的空间开销，我们认为，BGCH 可以作为这些全精度模型的备选方案，尤其适用于计算资源有限的场景。

5.4 消融研究 (RQ3)

我们使用 Top-20 搜索指标评估各模型组件的必要性，并在表 4 中报告结果。

特征分散的影响。我们首先分析所提出的特征分散方法在缓解哈希编码中的特征侵蚀方面的作用。我们引入一个模型变体，记为 *w/o FD*，将 η 设为 0 即可直接禁用该方法。如表 4 所示，*w/oFD* 与 BGCH 之间的性能差距，充分证明了在嵌入二值化以进行哈希之前分散潜在特征的有效性，这一结论适用于这六个数据集。此外，令表 1 中汇总的密度为 $\frac{|\mathcal{V}_1|\times|\mathcal{V}|}{|\mathcal{E}|^2}$。在稀疏数据集，即 Gowalla (0.00084)、Pinterest (0.00267)、Yelp2018 (0.00130) 和 AMZ-Book (0.00062) 中，BGCH 与 *w/o FD* 之间的性能衰减远大于另外两个数据集，即 MovieLens (0.04190) 和大众点评 (0.02210)。这是因为稀疏数据集对哈希更为敏感，因为它们可能没有足够的训练边来弥合与未哈希版本之间的差距。解决数据稀疏性问题的另一种有前景的方法是数据增强 [67]，我们将其留待未来工作。

自适应图卷积哈希的效果。随后，我们通过设置两个变体来研究该模型组件，其中：（1）*w/oAH-TA* 仅禁用哈希的拓扑感知，并在所有图卷积之后将其设为最终编码器（类似于传统方式 [4, 49]）；（2）*w/o AH-RF* 移除重缩放因子。根据表 4 的结果，我们得到以下观察：

（1）变体 *w/o AH-TA* 始终劣于 BGCH。这表明，仅使用 GCN框架的最终输出嵌入可能无法充分建模用于哈希的独特潜在节点特征，尤其是不同图深度中丰富的结构信息。而在 BGCH 中，通过捕获中间信息进行表示增强，拓扑感知哈希能够有效缓解离散哈希码表达能力有限的问题。

（2）除了拓扑感知哈希之外，促成性能提升的另一个关键因素是我们在公式（8）中引入的重缩放因子。从 BGCH 中移除该因子后，变体*w/o AH-RF* 的性能会大幅下降。尽管这些因子是直接计算得到的，且在理论上可能并非最优，但它们反映了嵌入的数值独特性，有助于后续的哈希编码，从而显著提升 BGCH 的预测能力。下一节将研究因子计算的确定性设计。

可学习重缩放设计。我们加入了另一个变体，即*w/in LF*，表示使用可学习重缩放因子的模型版本。如表 4 所示，*w/in LF* 中的可学习重缩放因子设计未能取得预期的良好性能。一种解释是，我们提出的模型目前没有对可学习因子施加足够强的数学约束（α_x），例如，$\alpha_{(l)x} = \text{argmin}(\tilde{\mathbf{v}}_{(l)x}, \alpha_{(l)x} \mathcal{Q}_{(l)x})$，主要是因为这会增加额外的训练复杂度；而纯粹依赖随机优化，例如随机梯度下降（SGD），可能很难达到最优解。考虑到该正则化设计引入的额外搜索空间，我们认为，我们的确定性重缩放方法在实践中简单且有效。

多重损失对优化的影响。最后，为了研究 BPR 损失 L^{bpr} 和图重构损失 L_{rec}的影响，我们设置了两个变体，分别称为 不使用 L^{bpr} 和不使用 L_{rec}，并分别对 BGCH 进行优化。如表 4 所示，在保留所有其他模型组件的情况下，仅使用 L^{bpr}和 L_{rec}中的任意一个都无法达到预期性能。这证实了我们所提出的多重损失设计的有效性：L^{bpr} 会学习为观测边（即 $Y_{x,\ y} = 1$）赋予高于未观测节点对的预测值，而 L_{rec} 则在训练过程中利用原始嵌入将图重构问题转化为分类任务。通过共同优化这两个损失函数，我们的模型 BGCH 能够从 L_{rec}中学习到精确的中间嵌入，并相应地生成由 L^{bpr} 正则化、具有高质量相对顺序信息的目标哈希码。

5.5 资源消耗分析（RQ4）

由于六个数据集上的取值范围各不相同，我们在图4中简要报告了BGCH相对于当前最先进的基于哈希的模型HashGNN$_s$ 的数值比率。

模型训练时间成本。如图4中的指标“*T*-时间”所示，我们注意到，训练HashGNN$_s$ 比训练我们提出的模型更耗时。主要原因是

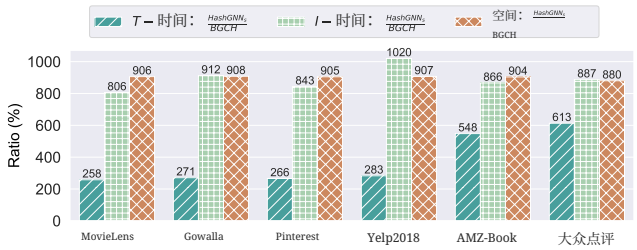
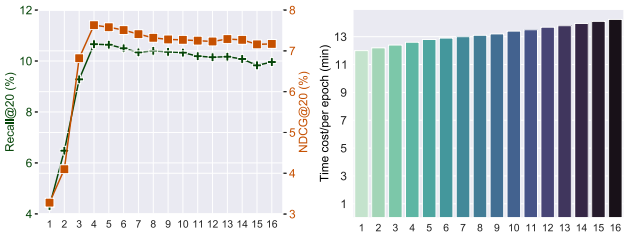


图4：资源消耗比。



（a）Top-20 检索指标 （b）训练时间成本

图5：BGCH 中的傅里叶级数分解项 n 。

HashGNN采用了较早的GCN框架 [17] 作为模型骨干，而我们的模型采用了最新的框架 [19]，以去除自连接、特征变换和非线性激活等操作。此外，在最大的两个数据集AMZ-Book和大众点评上，训练成本的比例进一步增加到约 5~6 倍。这是因为为了使训练过程可行，我们不得不减小HashGNN$_s$ 的批大小。

在线推理时间开销。 我们随机生成 1,000 个查询，并评估计算耗时。为确保比较公平，我们使用开源工具包[6]禁用了所有并行算术技术（如 MKL、BLAS）。如图4中的"*I-Time*"所示，在所有数据集上，我们采用汉明距离匹配的模型相比 HashGNN$_s$ 通常实现了超过 8× 的计算加速。这是因为，正如我们在 § 5.2 中所述，HashGNN$_s$会随机将哈希码替换为其原始连续嵌入以进行松弛，并采用浮点运算来提升性能，同时牺牲了位运算带来的计算加速。

哈希码内存占用。 二值化嵌入能够大幅降低内存空间消耗。与最先进的基于哈希模型 HashGNN$_s$相比，我们的 BGCH 进一步将哈希码的内存空间占用减少了约 9× 。正如我们刚才所述，由于 HashGNN$_s$ 使用随机实数位来表示哈希码，因此需要额外开销来区分二进制位与全精度位。相反，BGCH 将二值化编码部分与相应的重缩放因子分开存储，从而有利于优化空间开销。

5.6 傅里叶梯度估计研究（研究问题5）

我们以规模最大的大众点评数据集为例进行说明，这一分析也普遍适用于其他数据集。

分解项的影响 n。 我们将分解项 n 从 1 调整到 16。如图5所示，我们有两点观察：（1）不同的分解项必然会影响最终的检索质量；从理论上讲，n 越大，估计出的梯度就越准确。然而在实践中，n 的值过大可能会带来过拟合风险，这意味着保持适中的 n，例如，

n=4 在图5(a)中已经能够最大化模型性能。(2) 当将 n 从 1 调整至 16 时，BGCH 每次迭代的训练时间缓慢增加。这总体上与我们在附录 C 中的复杂度分析一致，其中大部分训练成本集中在我们的特征扩散和图卷积哈希上，如 $O(\ ^{2cs}\ (K+L)\ |\ |^2\mathcal{E}) \gg O(snd|\mathbb{E}|)$。

[6] https://www.lfd.uci.edu/~gohlke/pythonlibs/

表5：Recall@20 上的梯度估计器比较。

	Movie	Gowalla	Pinterest	Yelp2018	AMZ-Book	大众点评
STE	20.93 (-8.44%)	14.85(-11.24%)	12.35 (-3.36%)	5.24 (-4.90%)	3.12(-10.34%)	10.34 (-3.00%)
Tanh	21.75 (-4.86%)	15.06 (-9.98%)	12.36 (-3.29%)	5.43 (-1.45%)	3.21 (-7.76%)	10.41 (-2.34%)
SignSwish	22.13 (-3.19%)	15.62 (-6.63%)	12.44 (-2.66%)	5.50 (-0.18%)	3.34 (-4.02%)	10.43 (-2.16%)
Sigmoid	22.08 (-3.41%)	15.21 (-9.09%)	12.52 (-2.03%)	5.53 (+0.03%)	3.18 (-8.62%)	10.38 (-2.63%)
PBE	21.68 (-5.16%)	15.05 (-10.04%)	12.32 (-3.60%)	5.35 (-2.90%)	3.13 (-10.06%)	10.47 (-1.78%)
BGCH	**22.86**	**16.73**	**12.78**	**5.51**	**3.48**	**10.66**

与其他梯度估计器的比较。我们纳入了近期提出的几种梯度估计器，即 类*Tanh* [15, 44], *Sign-Swish* [11], *Sigmoid* [59], 和基于投影的估计器 [39]（记为 PBE）。(1) 表5中汇总的结果充分证明了我们提出的傅里叶级数分解在梯度估计方面优于 sign(·) 函数。正如我们简要解释的那样，现有的大多数估计器都采用与 视觉上相似的函数来近似 sign(·)；与 STE 相比，它们通常能够提供更好的梯度估计。(2) 然而，对于那些稀疏性较高的二部图，例如 Gowalla（0.00084）和 AMZ-Book（0.00062），基于图的模型可能难以收集足够的结构信息，以有效训练哈希码。基于有限的训练样本，这些理论上无关的估计器可能无法有效纠正优化偏差，因此与我们提出的傅里叶序列化估计器相比，表现出明显的性能差距。

6 结论

我们研究了二部图上的图卷积哈希，以实现高效的汉明空间搜索，并提出了包含三个有效模块的 BGCH。大量实验表明，该模型优于传统方法，并验证了所提出各模块的有效性。未来，我们计划研究主要因其实践中的普遍性而采用的半监督图设置 [47, 48]。此外，另一个有前景的方向是将 BGCH 升级为归纳学习 [17],，使其能够针对结构更新的演化图进行自适应匹配和预测。

致谢

我们感谢匿名审稿人提出的真知灼见和建议。Yankai Chen、Yifei Zhang 和 Irwin King 得到了中国国家重点研发计划（编号：2018AAA0100204）以及中国香港特别行政区研究资助局（CUHK 2410021、研究影响基金，编号：R5034-18）的资助。Yixiang Fang 得到了国家自然科学基金委员会项目（62102341）的资助。

参考文献

[1] Haoli Bai、Wei Zhang、Lu Hou、Lifeng Shang、Jing Jin、Xin Jiang、Qun Liu、Michael Lyu 和 Irwin King。2020。突破 BERT 量化的极限。arXiv（2020）。[2] Yoshua Bengio、Nicholas Léonard 和 Aaron Courville。2013。通过随机神经元估计或传播梯度以实现条件计算。arXiv（2013）。[3] Joan Bruna、Wojciech Zaremba、Arthur Szlam 和 Yann LeCun。2013。图上的谱网络和局部连接网络。arXiv（2013）。[4] Zhangjie Cao、Mingsheng Long、Jianmin Wang 和 Philip S Yu。2017。Hashnet：通过延拓进行深度学习哈希。载于 ICCV。5608–5617。[5] Minyu Chen、Guoqiang Li、Chen Ma、Jingyang Li 和 Hongfei Fu。2022。通过对 GitHub 仓库进行稠密检索来回答编程问题。载于 COLING。[6] Yankai Chen、Huifeng Guo、Yingxue Zhang、Chen Ma、Ruiming Tang、Jingjie Li 和 Irwin King。2022。通过多方面量化强化学习二值化图表示，用于 Top-k 推荐。载于 SIGKDD。[7] Yankai Chen、Menglin Yang、Yingxue Zhang、Mengchen Zhao、Ziqiao Meng、Jianye Hao 和 Irwin King。2022。利用双曲几何建模无标度图，实现知识感知推荐。载于 WSDM。94–102。[8] Yankai Chen、Yaming Yang、Yujing Wang、Jing Bai、Xiangchen Song 和 Irwin King。2022。通过协同引导实现个性化推荐的注意力知识感知图卷积网络。载于 ICDE。[9] Yankai Chen、Yifei Zhang、Huifeng Guo、Ruiming Tang 和 Irwin King。2022。一种高效的训练后嵌入二值化方法，用于快速在线 Top-K 段落匹配。载于 AACL-IJCNLP。102–108。[10] Zhiyong Cheng、Ying Ding、Lei Zhu 和 Mohan Kankanhalli。2018。方面感知潜在因子模型：结合评分和评论进行评分预测。载于 WWW。[11] Sajad Darabi、Mouloud Belbahri、Matthieu Courbariaux 和 Vahid Partovi Nia。2018。Bnn+：改进的二值网络训练。（2018）。[12] Michaël Defferrard、Xavier Bresson 和 Pierre Vandergheynst。2016。采用快速局部谱滤波的图卷积神经网络。NeurIPS 29（2016）。[13] Yifan Gao、Chien-Sheng Wu、Jingjing Li、Shafiq Joty、Steven CH Hoi、Caiming Xiong、Irwin King 和 Michael Lyu。2020。Discern：用于对话机器阅读的话语感知蕴含推理网络。载于 EMNLP。2439–2449。[14] Aristides Gionis、Piotr Indyk、Rajeev Motwani 等。1999。通过哈希在高维空间中进行相似度搜索。载于 PVLDB，第 99 卷。518–529。[15] Ruihao Gong、Xianglong Liu、Shenghu Jiang、Tianxiang Li、Peng Hu、Jiazhen Lin、Fengwei Yu 和 Junjie Yan。2019。可微软量化：连接全精度与低比特神经网络。载于 ICCV。4852–4861。[16] Aditya Grover 和 Jure Leskovec。2016。node2vec：面向网络的可扩展特征学习。载于 SIGKDD。855–864。[17] William L Hamilton、Rex Ying 和 Jure Leskovec。2017。大型图上的归纳式表示学习。载于 NeurIPS。1025–1035。[18] Ruining He 和 Julian McAuley。2016。利用单类协同过滤建模时尚趋势的视觉演变。载于 WWW。507–517。[19] Xiangnan He、Kuan Deng、Xiang Wang、Yan Li、Yongdong Zhang 和 Meng Wang。2020。LightGCN：简化并增强用于推荐的图卷积网络。载于 SIGIR。639–648。[20] Xiangnan He、Lizi Liao、Hanwang Zhang、Liqiang Nie、Xia Hu 和 Tat-Seng Chua。2017。神经协同过滤。载于 WWW。173–182。[21] Xiangnan He、Hanwang Zhang、Min-Yen Kan 和 Tat-Seng Chua。2016。利用隐式反馈进行在线推荐的快速矩阵分解。载于 ICCV。[22] Xuming Hu、Fukun Ma、Chenyao Liu、Chenwei Zhang、Lijie Wen 和 Philip S Yu。2021。通过增量元自训练进行半监督关系抽取。载于 EMNLP。[23] Xuming Hu、Lijie Wen、Yusong Xu、Chenwei Zhang 和 Philip S. Yu。2020。Self-ORE：用于开放关系抽取的自监督关系特征学习。载于 EMNLP。3673–3682。[24] Eric Jang、Shixiang Gu 和 Ben Poole。2017。采用 Gumbel-Softmax 的分类重参数化。（2017）。[25] Herve Jegou、Matthijs Douze 和 Cordelia Schmid。2010。用于最近邻搜索的乘积量化。TPAMI 33，1（2010），117–128。[26] Alexis Joly 和 Olivier Buisson。2011。随机最大间隔哈希。载于 CVPR。IEEE，873–880。[27] Wang-Cheng Kang、Derek Zhiyuan Cheng、Tiansheng Yao、Xinyang Yi、Ting Chen、Lichan Hong 和 Ed H Chi。2021。无需嵌入表即可学习嵌入类别特征以用于推荐。载于 SIGKDD。840–850。[28] Wang-Cheng Kang 和 Julian McAuley。2019。使用二进制编码为大规模 Top-N 推荐生成候选项。载于 CIKM。1523–1532。[29] Diederik Kingma 和 Jimmy Ba。2015。随机优化方法。ICLR。[30] Thomas N Kipf 和 Max Welling。2017。利用图卷积网络进行半监督分类。（2017）。[31] Piotr Koniusz、Fei Yan、Philippe-Henri Gosselin 和 Krystian Mikolajczyk。2016。词袋的高阶出现池化：视觉概念检测。TPAMI 39，2（2016），313–326。[32] Piotr Koniusz、Hongguang Zhang 和 Fatih Porikli。2018。深入探讨幂归一化。载于 CVPR。5774–5783。

[33] Tim Kraska、Alex Beutel、Ed H Chi、Jeffrey Dean 和 Neoklis Polyzotis。2018。论学习型索引结构的价值。载于 SIGMOD。489–504。[34] Guohao Li、Matthias Muller、Ali Thabet 和 Bernard Ghanem。2019。DeepGCNs：图卷积网络能像卷积神经网络一样加深吗？载于 ICCV。9267–9276。[35] Hao Li、Wei Liu 和 Heng Ji。2014。用于快速文档检索的两阶段哈希。载于 ACL。495–500。[36] Jingjing Li、Zichao Li、Lili Mou、Xin Jiang、Michael Lyu 和 Irwin King。2020。通过从搜索中学习进行无监督文本生成。NeurIPS。[37] Dawen Liang、Laurent Charlin、James McInerney 和 David M Blei。2016。对推荐中的用户曝光进行建模。载于 WWW。951–961。[38] Xiaofan Lin、Cong Zhao 和 Wei Pan。2017。迈向精确的二值卷积神经网络。（2017）。[39] Chunlei Liu、Wenrui Ding、Xin Xia、Yuan Hu、Baochang Zhang、Jianzhuang Liu、Bohan Zhuang 和 Guodong Guo。2019。用于提升 1位深度卷积神经网络性能的整流二值卷积网络。arXiv（2019）。[40] Chen Ma、Peng Kang 和 Xue Liu。2019。用于序列推荐的分层门控网络。载于 SIGKDD。[41] Chen Ma、Liheng Ma、Yingxue Zhang、Ruiming Tang、Xue Liu 和 Mark Coates。2020。基于自适应间隔的概率度量学习，用于 Top-K 推荐。载于 SIGKDD。[42] Chris J Maddison、Andriy Mnih 和 Yee Whye Teh。2017。Concrete 分布：离散随机变量的连续松弛。（2017）。[43] Yury Nahshan、Brian Chmiel、Chaim Baskin、Evgenii Zheltonozhskii、Ron Banner、Alex M Bronstein 和 Avi Mendelson。2021。损失感知的训练后量化。《机器学习》（2021），1–18。[44] Haotong Qin、Ruihao Gong、Xianglong Liu、Mingzhu Shen、Ziran Wei、Fengwei Yu 和 Jingkuan Song。2020。为精确的二值神经网络保留前向和后向信息。载于 CVPR。2250–2259。[45] Mohammad Rastegari、Vicente Ordonez、Joseph Redmon 和 Ali Farhadi。2016。使用二值卷积神经网络进行 ImageNet 分类。载于 ECCV。Springer，525–542。[46] BRYAN RUST。2013。傅里叶级数的收敛。（2013）。[47] Zixing Song、Ziqiao Meng、Yifei Zhang 和 Irwin King。2021。面向图结构数据的半监督多标签学习。载于 CIKM。[48] Zixing Song、Yifei Zhang 和 Irwin King。2022。迈向用于鲁棒半监督节点分类的最优非对称图结构。载于 SIGKDD。[49] Qiaoyu Tan、Ninghao Liu、Xing Zhao、Hongxia Yang、Jingren Zhou 和 Xia Hu。2020。利用图神经网络为推荐系统学习哈希。载于 WWW。1988–1998。[50] Petar Velickovic、Guillem Cucurull、Arantxa Casanova、Adriana Romero、Pietro Liò 和 Yoshua Bengio。2018。图注意力网络。载于 ICLR。[51] Jingdong Wang、Ting Zhang、Nicu Sebe、Heng Tao Shen 等。2017。学习哈希综述。TPAMI 40，4（2017），769–790。[52] Xiang Wang、Xiangnan He、Meng Wang、Fuli Feng 和 Tat-Seng Chua。2019。神经图协同过滤。载于 SIGIR。165–174。[53] Xiang Wang、Hongye Jin、An Zhang、Xiangnan He、Tong Xu 和 Tat-Seng Chua。2020。解耦图协同过滤。载于 ICLR。1001–1010。[54] Xing Wei、Yue Zhang、Yihong Gong、Jiawei Zhang 和 Nanning Zheng。2018。将 Grassmann 池化作为用于细粒度视觉分类的紧凑齐次双线性池化。载于 ECCV。355–370。[55] Yair Weiss 和 Antonio Torralba。2008。谱哈希。NeurIPS（2008）。[56] Felix Wu、Amauri Souza、Tianyi Zhang、Christopher Fifty、Tao Yu 和 Kilian Weinberger。2019。简化图卷积网络。载于 ICML。[57] Keyulu Xu、Weihua Hu、Jure Leskovec 和 Stefanie Jegelka。2019。图神经网络的能力有多强？（2019）。[58] Yixing Xu、Kai Han、Chang Xu、Yehui Tang、Chunjing Xu 和 Yunhe Wang。2021。为二值神经网络学习频域近似。arXiv（2021）。[59] Jiwei Yang、Xu Shen、Jun Xing、Xinmei Tian、Houqiang Li、Bing Deng、Jianqiang Huang 和 Xian-Sheng Hua。2019。量化网络。载于 CVPR。7308–7316。[60] Menglin Yang、Min Zhou、Jiahong Liu、Defu Lian 和 Irwin King。2022。HRCF：通过双曲几何正则化增强协同过滤。载于 WWW。2462–2471。[61] Tan Yu、Yunfeng Cai 和 Ping Li。2020。通过秩一更新实现更快、更简单的矩阵归一化。载于 ECCV。Springer，203–219。[62] Dell Zhang、Jun Wang、Deng Cai 和 Jinsong Lu。2010。词项和文档的拉普拉斯协同哈希。载于 ECIR。Springer，577–580。[63] Jiani Zhang、Xingjian Shi、Shenglin Zhao 和 Irwin King。2019。用于推荐系统的堆叠重构图卷积网络。IJCAI。[64] Xinni Zhang、Yankai Chen、Cuiyun Gao、Qing Liao、Shenglin Zhao 和 Irwin King。2022。用于冷启动推荐的知识感知神经网络与个性化特征引用。arXiv（2022）。[65] Yan Zhang、Defu Lian 和 Guowu Yang。2017。基于隐式反馈的快速协同过滤离散个性化排序。载于 AAAI，第 31 卷。[66] Yifei Zhang 和 Hao Zhu。2019。Doc2hash：学习用于文档检索的离散潜变量。载于 NAACL。2235–2240。[67] Yifei Zhang、Hao Zhu、Zixing Song、Piotr Koniusz 和 Irwin King。2022。COSTA：用于图对比学习的协方差保持特征增强。载于 SIGKDD。[68] Yifei Zhang、Hao Zhu、Zixing Song、Piotr Koniusz 和 Irwin King。2022。用于图对比学习及其他任务的谱特征增强。载于 arXiv。

表6：符号及含义。

符号	含义
G、V_1、V_2、E	包含节点集合和边的二部图。
c, d	卷积维度和哈希码维度。
$V_x^{(l)}$	节点 x 在第 l次迭代时的嵌入。
Y	边交易，其中 $Y_{x,\ y}=1$ 表示交互存在于节点 x 和 y之间，否则为 $Y_{x,\ y}=0$。
A, D	邻接矩阵及其对应的对角度矩阵。
$p^{(k)}, P$	第 k 次迭代时的离散向量和投影矩阵。
$\widetilde{V}$	特征离散嵌入矩阵。
$\widehat{Y}$	估计匹配分数。
$Q_x^{(l)}$	第 l次迭代时节点 x 的哈希码段。
$\alpha(l)$	x在第 l次卷积时计算出的重缩放因子。
Q_x	节点 x的目标哈希码。
L, K	卷积哈希和离散生成的次数。
$\mathcal{L}_{rec}$、$\mathcal{L}_{bpr}$、$\mathcal{L}$	最终目标函数$\mathcal{L}$的两个损失项。
η、H、n、λ_1、λ_2	超参数。

算法 1： BGCH算法。

```
当 模型尚未收敛时
    对于 k = 0, ..., K − 1 执行
        p^(k+1) ← (V(0))^T V(0) p^(k);
    P ← 获得投影矩阵;                                   ▷ 公式（4）
    Ṽ^(0) ← 获得特征分散嵌入;   ▷ 公式（5）
    对于 l = 0, ··· L − 1 执行
        Ṽ^(l+1) ← (D^{-1/2} A D^{-1/2}) Ṽ^(l);          ▷ 式（6）
        Q^(l+1) ← 符号 (Ṽ^(l+1));                       ▷ 式（7）
        α ← 计算重缩放因子;                             ▷ 公式（8）
    对于 x ∈ V1, y ∈ N(x) 执行
        Ŷ_{x,y} ← α_x α_y (d − 2 D_H(Q_x, Q_y));        ▷ 公式（14）和定理2
    L ←计算损失并优化模型;   ▷ 公式（15-17）
函数梯度_估计器(L):
    ∂L/∂Ṽ ← ∂L/∂Q · (4/H) Σ_{i=1,3,5,···} cos(πi/H);    ▷ 式（12）
```

A 损失景观可视化

我们模拟可学习嵌入的优化轨迹，并在图 2(a) 中直观比较非哈希版本和哈希版本的损失景观。具体而言，我们手动为 MovieLens 数据集上的嵌入赋予扰动 [1, 43]，如下所示：$V_x^{(l)} = V_x^{(l)} \pm p \cdot |V_x^{(l)}| \cdot \mathbf{1}^{(l)}$。其中，$|V_x^{(l)}|$表示 $V_x^{(l)}$中各项绝对值的均值，扰动幅度 p取自 $\{0.01, \cdots 0.50\}$。$\mathbf{1}$是全 1 向量。对于成对的受扰动节点嵌入，我们相应地绘制其损失分布。可以看出，非哈希版本产生了平坦的损失曲面，呈现出局部凸性。相反，哈希版本的损失景观则起伏不平且复杂。

B 符号表与 BGCH 伪代码

我们使用粗体大写字母和花体字母表示矩阵和集合。非粗体字母表示图节点或标量。表 6 和算法 1 对主要符号和伪代码作了解释。

C 理论证明与分析

定理 *1*（特征扩散）。设 $V(l) = U_1 \Sigma U T2$，其中 U_1和 U_2 为酉矩阵，降序奇异值矩阵 $\Sigma = diag(\sigma_1, \sigma_2, \cdots \sigma_c)$。则 $\mathbb{E}(\widetilde{V}(l)) = U_1 \Sigma \Sigma_\mu U T2$，其中 $\Sigma_\mu = diag(\mu_1, \mu_2, \cdots \mu_c)_{0<\mu_1 \cdots c<1}$ 按升序排列。

证明。我们重点比较 $(\widetilde{V}(0), V(0))$，该结论可以轻松推广到任意卷积层，即 $(\widetilde{V}(l), V(l))$。对 $V(0)$ 进行奇异值分解，我们得到 $V(0) = U_1 \Sigma U T2$，其中 U_1 和 U_2 为由奇异向量构成的酉矩阵。根据 $p^{(k)} = V(0)^{\mathrm{T}} V(0) p^{(k-1)}$，可得 $p^{(k)} = (V(0)^{\mathrm{T}} V(0))^k p^{(0)}$。将$V^{(0)}$ 替换为其奇异值分解，可得到如下等式：

$$p^{(k)} = (U_2 \Sigma^{2k} U_2^{\mathrm{T}}) p^{(0)}. \tag{18}$$

接下来，我们将公式（4）中的投影矩阵变换如下：

$$\begin{aligned} P = \frac{p^{(k)} p^{(k)\mathrm{T}}}{p^{(k)\mathrm{T}} p^{(k)}} &= \frac{(U_2 \Sigma^{2k} U_2^{\mathrm{T}}) p^{(0)} p^{(0)\mathrm{T}} (U_2 \Sigma^{2k} U_2^{\mathrm{T}})}{p^{(0)\mathrm{T}} (U_2 \Sigma^{2k} U_2^{\mathrm{T}}) (U_2 \Sigma^{2k} U_2^{\mathrm{T}}) p^{(0)}} \\ &= U_2 \Sigma^{2k} \frac{U_2^{\mathrm{T}} p^{(0)} p^{(0)\mathrm{T}} U_2}{p^{(0)\mathrm{T}} U_2 \Sigma^{4k} U_2^{\mathrm{T}} p^{(0)}} \Sigma^{2k} U_2^{\mathrm{T}}. \end{aligned} \tag{19}$$

令 $t = U T2\, p_{(0)}$，我们可以进一步将上式简化为：

$$P = U_2 \Sigma^{2k} \frac{t t^{\mathrm{T}}}{t^{\mathrm{T}} \Sigma^{4k} t} \Sigma^{2k} U_2^{\mathrm{T}}, \text{ where scalar } t^{\mathrm{T}} \Sigma^{4k} t = \sum_{j=1}^{c} t_j^2 \sigma_j^{4k}. \tag{20}$$

回顾 $\widetilde{V}_{(0)} = V_{(0)}(I - \epsilon P)$、$\mathbb{E}(\widetilde{V}_{(0)}) = V_{(0)} - \epsilon \cdot \mathbb{E}(V_{(0)} P)$。接下来，我们考察项 $\mathbb{E}(V_{(0)} P)$，如下所示：

$$\mathbb{E}(V^{(0)} P) = \frac{1}{t^{\mathrm{T}} \Sigma^{4k} t} U_1 \Sigma^{2k+1} \cdot \mathbb{E}(t t^{\mathrm{T}}) \cdot \Sigma^{2k} U_2^{\mathrm{T}} \tag{21}$$

由于 $p^{(0)} \sim \mathrm{N}(0, I)$ 且 U_2是酉矩阵，因此 $t \sim \mathrm{N}(0, I)$。这表明，t的每个元素（例如 $t_j \in t$）都是独立同分布随机变量。因此，对于 $j \neq k$，$\mathbb{E}(t_j \cdot t_k) = 0$，且 $\mathbb{E}(t t^{\mathrm{T}})$ 是对角矩阵，即 $\mathbb{E}(t t^{\mathrm{T}}) = \mathrm{diag}(t_1^2, t_2^2, \cdots t_c^2)$。于是有：

$$\mathbb{E}(V^{(0)} P) = U_1 \cdot \mathrm{diag}\left(\frac{\sigma_1 t_1^2 \sigma_1^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}, \cdots, \frac{\sigma_c t_c^2 \sigma_c^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}\right) \cdot U_2^{\mathrm{T}}. \tag{22}$$

Th因此，

$$\mathbb{E}(\widetilde{V}^{(0)}) = U_1 \cdot \mathrm{diag}\left(\sigma_1 - \epsilon \frac{\sigma_1 t_1^2 \sigma_1^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}, \cdots, \sigma_c - \epsilon \frac{\sigma_c t_c^2 \sigma_c^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}\right) \cdot U_2^{\mathrm{T}}. \tag{23}$$

设 $\mu_k = 1 - \epsilon \frac{t_k^2 \sigma_k^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}$，其中 $\epsilon \in (0, 1)$，显然，$0 < \mu_k < 1$。此外，$\forall k_1 \geq k_2$，我们有：

$$\mu_{k_1} - \mu_{k_2} = \epsilon \mathbb{E}\left(\frac{t_{k_1}^2 \sigma_{k_1}^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}} - \frac{t_{k_2}^2 \sigma_{k_2}^{4k}}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}\right) \geq \epsilon \sigma_{k_1}^{4k} \cdot \mathbb{E}\left(\frac{t_{k_1}^2 - t_{k_2}^2}{\sum_{j=1}^{c} t_j^2 \sigma_j^{4k}}\right) = 0, \tag{24}$$

当 $\sigma_{k_2}^{4k} \geq \sigma_{k_1}^{4k}$ 时，t_{k_1} 和 t_{k_2} 是 独立同分布的随机变量，且服从相同的正态分布。公式 (24) 证明了 μ_k 在期望意义下是 单调非递减的，从而完成证明。□

定理 2 (汉明距离匹配)。给定两个哈希码，我们有$(\alpha_x Q_u)^{\mathrm{T}} \cdot (\alpha_y Q_y) = \alpha_x \alpha_y (d - 2 D_H(Q_x, Q_y))$。

Proof.

$$\begin{aligned} Q_x^{\mathrm{T}} \cdot Q_y &= \left|\{i | (Q_x)_i = (Q_y)_i = 1\}\right| + \left|\{i | (Q_x)_i = (Q_y)_i = -1\}\right| \\ &\quad - \left|\{i | (Q_x)_i \neq (Q_y)_i\}\right| \\ &= d - 2 \cdot \left|\{i | (Q_x)_i \neq (Q_y)_i\}\right| = d - 2 D_H(Q_x, Q_y), \end{aligned} \tag{25}$$

从而完成证明。□

训练时间复杂度。如表7所示，|E|、B、s和n 分别表示边数、批大小、训练迭代次数以及

表7：训练时间复杂度。

图归一化	特征分散	卷积与哈希	损失计算	梯度估计
$O(2\text{东经})$	$O(\frac{2csK\mathrm{E}_B\ ^2}{})$	$O(\frac{2csL\text{东}}{\ ^2})$	$O(2s(c+d)\ \mathrm{E})$	$O(snd\text{东经})$

表 8：空间和计算复杂度。

	空间开销	#浮点运算	#BOP
基于 float32	$32\mathrm{V}_1 \cup \mathrm{V}_2 d$	$O(\mathrm{V}_1 \cdot \mathrm{V}_2 d)$	-
BGCH	$\mathrm{V}_1 \cup \mathrm{V}_2 (d + 32(L+1))$	$O(4\mathrm{V}_1 \cdot \mathrm{V}_2)$	$O(\mathrm{V}_1 \cdot \mathrm{V}_2 d)$

傅里叶级数分解项。(1) 图归一化的时间复杂度，即 $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$，为 $O(2\mathrm{E})$。(2) 在图卷积之前，我们首先仅对初始节点嵌入进行特征扩散，例如 $\boldsymbol{V}(0)x$，其复杂度为 $O(\ ^{2csK}|\mathcal{E}_B\ ^{|^2})$。在我们的实验中，超参数为 $K \leq 3$。(3) 在图卷积中，时间复杂度为 $O(\ ^{2csL}|\mathcal{E}_B\ ^{|^2})$，其中 $L \leq 4$ 是常用设置 [17, 19, 30, 52]，用于避免过平滑 [34]。(4) 对于损失计算，BGCH 需要 $O(2sc|\mathrm{E}|)$ 来计算 L_{rec}，并需要 $O(2sd|\mathrm{E}|)$ 来计算 L_{bpr}。(5) 最后，BGCH 需要 $O(snd|\mathrm{E}|)$ 来估计 d 维哈希码的梯度。因此，总体时间复杂度与图的边数呈二次关系，即 $|\mathrm{E}|$，这在图卷积网络框架中很常见。

哈希码的空间开销。 如表 8 所示，假设在 $L+1$ 次迭代中对重缩放因子使用 float32，则哈希码的总空间开销为 $O(|\mathcal{V}_1 \cup \mathcal{V}_2|(d + 32(L+1)))$ 位。与连续嵌入的尺寸（即 $32|\mathrm{V}_1 \cup \mathrm{V}_2|d$ 位）相比，理论上的尺寸缩减比为：

$$ratio = \frac{32|\mathcal{V}_1 \cup \mathcal{V}_2|d}{|\mathcal{V}_1 \cup \mathcal{V}_2|(d + 32(L+1))} = \frac{32d}{d + 32(L+1)}. \quad (26)$$

正如我们刚才解释的，堆叠过多的迭代层会导致性能下降 [34]。因此，如果 $L \leq 4$ 和 $d = 1024$，BGCH 可以实现显著的尺寸压缩。

在线匹配。 公式（14）中的原始评分计算包含 d 次浮点运算（#FLOPs）。如表 8 所示，使用汉明距离匹配可以将大部分浮点运算转换为二进制运算（#BOPs），同时标量计算所需的 #FLOPs 略有增加，即 $4 \ll d$。

D 实验设置详情

数据集。 我们在以下六个数据集上评估模型：
(1) **MovieLens**[7] 是用户与电影之间广泛采用的基准数据集。与 [21, 49], 中的设置类似，如果用户 x 对物品 y 评分，我们就设置边 $Y_{x,\ y} = 1$；否则设置 $Y_{x,\ y} = 0$。(2) **Gowalla**[8] [19, 49, 52, 53] 是根据 Gowalla 收集的顾客与其签到地点之间的数据集 [37] 。(3) **Pinterest**[9] 是一个用于用户与图像之间图像推荐的开放数据集。边表示用户在图像上创建的图钉。(4) **Yelp2018**[10] 来自 Yelp Challenge 2018 版，以二部图形式建模用户与本地商家之间的关系。(5) **AMZ-Book**[11] 是读者与图书之间的二部图，整理自 Amazon-review [18] 的图书集合。(6) **Dianping**[12] 是一个记录用户与本地商家之间多种互动行为（如点击、收藏和购买）的商业数据集。

评估指标。 为评估模型在二部图上的汉明空间检索性能，我们直接将模型 BGCH 部署于基础的用户-物品推荐场景中。具体而言，给定一个查询节点，我们利用哈希码匹配与查询节点具有最小汉明距离的 Top-N 答案，并采用广泛使用的两种评估协议 Recall@N 和 NDCG@N 来衡量排序能力。

实现细节。我们使用 Python 3.6 和 PyTorch 1.14.0，在配备 1 块 Nvidia GeForce RTX 3090 GPU、4 个 Intel Core i7-8700 CPU、32 GB 内存且主频为 3.20 GHz 的 Linux 机器上实现模型。对于所有基线方法，我们均遵循官方的超参数设置。若缺少推荐的模型设置，则采用网格搜索。维度在 {32，64，128，256，512} 中搜索。学习率 η在 $\{10^{-3},\ 5\times10^{-3},\ 10^{-2},\ 5\times10^{-2}\}$ 范围内调优，系数 λ则在 $\{10^{-5},\ 10^{-4},\ 10^{-3}\}$ 中调优。我们使用默认的正态初始化器初始化所有模型，并使用 Adam优化器 [29]进行优化。

基线方法。本文研究的所有基线方法介绍如下：
(1) **LSH**[14]是一种经典的基于随机投影的相似度搜索，我们遵循[49]中的适配方法将其引入Top-N对象搜索。

（2）**HashNet** [4] 是一种具有代表性的深度哈希方法，最初为多媒体检索任务提出。与 [49], 类似，我们通过采用通用图卷积网络对其进行修改，使其适用于图数据。（3）**CIGAR** [28] 是一种基于神经网络的先进框架，用于推荐中的快速Top-N候选生成。CIGAR之后还可以进一步采用全精度重排序算法。为进行公平比较，我们仅使用其哈希部分。（4）**Hash_Gumbel** 是BGCH的一种变体，使用Gumbel-softmax进行哈希编码和梯度估计 [24, 42]。具体而言，我们首先将每个嵌入位扩展为大小为二的一位有效编码。然后，它利用Gumbel-softmax技巧替代sign(·)，作为二进制哈希码生成的松弛方法。（5）**HashGNN** [49] 是一种基于GCN框架的先进学习哈希方法。我们使用HashGNN$_h$ 表示 [49],中提出的采用硬编码 的原始版本，其中用户-物品嵌入的每个元素都被严格二值化。我们使用HashGNN$_s$ 表示其提出的近似版本。（6）**NeurCF** [20]是推荐中用于协同过滤的一种具有代表性的深度神经网络模型。（7）**NGCF** [52] 是采用协同过滤方法的代表性图推荐模型之一。（8）**DGCF** [53] 是一种先进的图模型，通过学习解耦用户意图来实现更好的推荐。（9）**LightGCN** [19] 是另一种最新的先进GCN推荐模型，已得到广泛评估。

[7] https://grouplens.org/datasets/movielens/1m/ [8] https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/gowalla [9] https://sites.google.com/site/xueatalphabeta/dataset-1/pinterest_iccv [10] https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018

[11] https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/amazon-book [12] https://www.dianping.com/