arXiv:2301.05430v1 [cs.IR] 13 Jan 2023

HS-GCN：用于推荐的汉明空间图卷积网络

Han Liu、Yinwei Wei，*IEEE* 会员，Jianhua Yin，*IEEE* 会员，以及 Liqiang Nie，*IEEE* 资深会员

摘要——针对大规模推荐系统，一种高效的解决方案是将用户和物品表示为汉明空间中的二进制哈希码。为此，现有方法倾向于通过建模用户与其历史交互物品之间的汉明相似度来编码用户，本文将其称为一阶相似度。尽管这些方法效率较高，但其表征能力并非最优，因为它们忽略了通过连接多个一阶相似度所建立的相关性，即间接实例之间的关系，这种关系可定义为高阶相似度。为解决这一缺陷，我们提出通过用户-物品二部图，在汉明空间中同时建模一阶相似度和高阶相似度。因此，我们开发了一种新颖的哈希学习框架，即汉明空间图卷积网络（HS-GCN），该框架显式建模汉明相似度，并将其嵌入用户和物品的哈希码中。在三个公开基准数据集上进行的大量实验表明，我们提出的模型显著优于若干最先进的哈希模型，且取得了与实值推荐模型相当的性能。

索引词—哈希、高效推荐、高阶相似度、图卷积网络、汉明空间。

1 引言

THE 推荐系统旨在根据用户的偏好，从海量信息中找到其感兴趣的物品。因此，如何衡量用户与物品之间的相似度，是个性化推荐的核心。为此，现有研究 [1], [2], [3] 往往遵循两阶段流程：先用向量表示用户和物品 [4],，再通过衡量向量之间的相似度来预测它们的交互。尽管取得了显著的性能，这些方法仍面临一个不可避免的问题：随着用户和物品数量的增加，计算量会呈指数级增长 [5]。理论上，要为每位用户推荐排名前k的偏好物品，当有 N 个用户和 M 个物品时，时间复杂度为 $\mathcal{O}(NMK + NM\log k)$，其中由

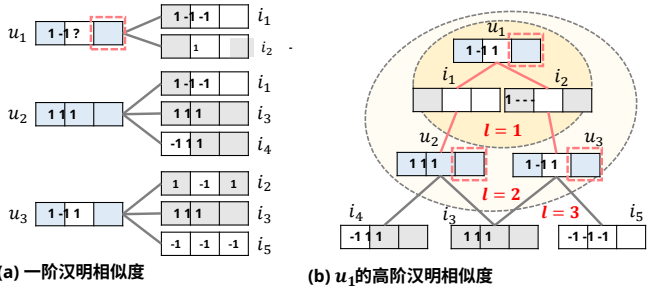


图1. 哈希学习中高阶与一阶汉明相似度的示意图。

潜在空间中的K维嵌入。

深入研究这些方法后，我们很容易发现，问题主要源于用户-物品相似度计算 [6]。然而，设计一种既能计算两个向量之间的相似度、又比内积更高效的新算法几乎是不可能的。因此，人们致力于为用户和物品学习一种新型表示——哈希码，以降低复杂度 [7], [8]。特别是，得益于这种由 ±1比特组成的代码，可以通过 XOR 位运算 [9]加速度量。因此，推荐的时间复杂度显著降低，利用查找表甚至可以实现常数时间搜索 [6]。由于用户-物品相似度的计算效率大幅提升，大规模推荐得以高效执行 [10], [11], ，尤其是在计算资源非常有限的移动应用中 [12], [13]。然而，由于用户和物品都被近似表示为二进制向量，基于哈希的推荐模型往往性能欠佳。为解决这一缺陷，人们提出了多种方法来增强哈希码的表示能力 [14],[15], [16]。例如，HashNet [17] 利用深度神经网络学习用户和物品嵌入，然后将其映射到汉明空间，以计算两个向量之间的汉明距离。最近，受图卷积网络在表示学习方面取得成功的启发，研究者提出了 HashGNN [18] ，将局部结构信息编码到用户-物品交互图中的每个节点，并通过哈希层将其增强后的表示二值化。

尽管这些方法丰富了用户和物品的表示，但我们认为，它们仅利用发生交互的用户-物品对的相似度来学习哈希码，却忽略了隐藏在间接交互中的相似度。为了通过相似度计算区分这些相似度

• *Han Liu*、*Jianhua Yin* 和 *Liqiang Nie* 任职于中国青岛 *266200* 山东大学计算机科学与技术学院。电子邮件：*hanliu.sdu@gmail.com*、*jhyin@sdu.edu.cn*、*nieliqiang@gmail.com*。

• *Yinwei Wei* 任职于新加坡国立大学计算学院。电子邮件：*weiyinwei@hotmail.com*。

• *Liqiang Nie* 是通讯作者。

在欧几里得空间中 [2], ，我们分别将其称为一阶和高阶汉明相似度。为阐明我们的观点，我们连接直接交互以建模间接交互，从而构建出如图 1（b）所示的用户-物品二部图，并将其与图 1（a）所示的直接交互在哈希码学习方面进行比较。具体而言，用户（即 u_1、u_2 和 u_3）和物品（即 i_1、i_2 和 i_3）均由二进制向量表示，边的存在表示用户与物品发生了交互。为了学习用户的哈希码，基于深度学习的模型和基于 GNN 的模型都会对与其发生交互的物品进行编码，并最小化它们的一阶汉明相似度（即，$\langle u_1, i_1 \rangle$ 和 $\langle u_1, i_2 \rangle$），如图 1（a）所示。然而，对于 u_1，我们认为，仅凭一阶汉明相似度很难确定其代码，因为它恰好位于两个物品的中间。相反，如图 1（b）所示，通过纳入高阶汉明相似度，可以通过衡量其高阶汉明相似度来捕获隐藏线索，例如 $\langle u_1, u_2 \rangle$ 和 $\langle u_1, u_3 \rangle$，从而大幅促进哈希码学习。

因此，我们提出在用户-物品二部图上显式建模高阶汉明相似度，从而对用户和物品进行编码。值得注意的是，这与 HashGNN 不同：HashGNN 在欧几里得空间中学习节点的表示，然后将其转换为哈希码。我们认为，哈希码优化的监督信号仅来自历史中的一阶交互，这可能会加剧转换过程中高阶汉明相似度的信息损失。

然而，在汉明空间中执行图卷积操作以捕捉汉明相似度并非易事。按照图卷积网络中的术语，我们将挑战归纳为两个方面:

- 与连续向量上的操作不同，哈希码中的每个元素都被限制为二进制值。因此，对于每个节点，如何在汉明空间中聚合其邻居的信息是我们面临的首要挑战。

- 为了在汉明空间中表示节点，必须保留它们与邻居哈希码之间的汉明相似度。因此，如何将汉明相似度显式编码到每个节点中，是本研究面临的另一项技术挑战。

为应对上述挑战，我们开发了一种新颖的基于哈希的推荐模型，名为汉明空间图卷积网络（**HS-GCN**），由初始层、传播层和预测层组成。具体而言，在初始层中，我们将用户-物品交互视为一个二部图，并使用 [17], 中提出的哈希码生成方法初始化节点，该方法保证了端到端优化的可行性。在构建的图上，我们设计了一个码传播层，以在汉明空间中实现图卷积操作。更具体地说，该层可分为两个组件——哈希码聚合和哈希码编码。前者通过统计逐比特符号（即，找出每个维度中占主导地位的比特值）来聚合一阶相似邻居的信息对应哈希码中的信息。后者通过根据邻居的逐比特符号细化节点哈希码中的比特，将汉明相似度注入节点哈希码。借助堆叠传播层，我们迭代地将学习到的哈希码嵌入节点中。随后，在预测层中，我们可以通过对用户-物品对的亲和度进行评分来预测其交互。为评估所提出的模型，我们在三个公开可用的数据集上开展了广泛实验，并将其性能与若干先进基线方法进行比较。除了优于基于哈希的模型（例如 HashNet 和 HashGNN）之外，我们提出的方法还取得了与基于实值的模型相当的结果，例如 PinSage 和 GraphSAGE。

总的来说，我们的主要贡献可概括为以下三点:

- 据我们所知，这是首次在推荐系统中显式建模用户与物品之间的高阶汉明相似度，从而增强哈希码的表示能力，并相应地优化用户-物品交互预测。

- 我们提出了一种基于GCN的哈希推荐模型，称为汉明空间图卷积网络（HS-GCN），该模型能够显式捕获一阶和高阶汉明相似度。具体而言，我们在汉明空间中设计了新颖的图卷积操作——哈希码聚合和哈希码编码。

- 在三个真实世界数据集上的大量实验结果表明，与若干先进基线方法相比，所提出的模型取得了显著的性能提升。作为一项附带贡献，我们已公开代码，以便其他研究人员使用: https://github.com/hanliu95/HS-GCN。

2 相关工作

本节简要回顾基于图的推荐方法，以及用于推荐的哈希学习方法。

2.1 基于图的推荐方法

图上的机器学习是一项重要任务，其优势在于能够纳入结构信息。作为主要应用之一，利用信息传播在推荐场景中对用户-物品图结构进行表示学习，已得到广泛研究。在这一研究方向上，ItemRank [19] 和 BiRank [20]较早尝试了标签传播。具体而言，这些方法直接在图上传播用户偏好分数（即标签），为用户评估标签相似的相连物品。然而，这些方法本质上基于邻居，无法充分编码图的结构信息。

近年来，图神经网络受到了广泛关注 [21],[22], ，因为它们在建模图结构方面具有独特优势，尤其是能够通过信息传播来指导表示学习。然而，早期基于图神经网络的方法计算成本高昂，因为它们在谱域进行图卷积。随后，图卷积网络利用空间域中的图卷积操作

在空间域中聚合邻居的嵌入，以优化目标节点的嵌入 [3], [23]。受图卷积高效性的启发，GC-MC [24]、PinSage [25], 和 NGCF [2]采用图卷积网络，在用户-物品交互图中捕获潜在协同过滤（CF）信号的传播，用于推荐。然而，后续研究认为图卷积网络过于复杂。例如，LightGCN [26] 通过移除所有冗余参数构建了一个线性模型。实验结果表明，这种简化设计的表现远胜于原始图卷积网络。这说明图卷积网络的非线性对于协同过滤推荐并非必要。

如前所述，我们总结了两个要点：1）基于图的方法为建模基于用户-物品图的关系信息传播（例如汉明空间中的高阶相似度）提供了启发；2）简化后的图卷积性能更好，这验证了在基于图的推荐中，非线性并非必需。我们的工作深受这些分析的启发，因为基于图的机制能够有效地建模高阶汉明相似度，而且在汉明空间中所有操作都只是线性的。此外，异构图神经网络被提出用于适配结构复杂的图，以利用其中更丰富的信息。基于这项技术，一些方法通过聚合异构信息网络中不同类型的邻居信息来有效学习表示，例如 HERec [27], MCRec [28],MEIRec [29],和 ie-HGCN [30]。相反，我们的方法处理的是容易收集的简单交互信息。

2.2 推荐中的哈希学习

另一条相关研究路线是面向推荐的哈希学习，主要沿着两个方向展开：无监督哈希和有监督哈希 [31]。前者 [32],[33] 通过在无标签数据上训练，学习将对象编码为二进制码的哈希函数；后者 [34],[35], [36], [37] 则旨在通过探索有标签信号（例如推荐场景中用户与物品之间的反馈），学习判别能力更强的哈希码。总体而言，早期面向推荐的哈希学习方法本质上是两阶段方法 [38]。然而，学习得到的哈希码之间的汉明相似度可能无法对应用户与物品之间原本的相关性，因为在二值化步骤中将实数阈值化为二进制比特时会产生量化偏差。DCF [6]解决了这一具有挑战性的离散优化问题，并直接学习用户和物品的哈希码。因此，学习得到的哈希码能够刻画用户-物品之间的内在相关性。

一种普遍趋势是利用深度学习进行推荐。例如，NFM [39]成功将深度学习引入推荐领域，以增强表示学习和匹配函数建模。受此启发，随后发展出了深度哈希学习，并取得了良好的推荐性能。早期的深度哈希学习也采用两阶段策略：第一阶段利用深度网络学习连续表示，第二阶段则使用 sign(x) 函数将学得的表示二值化为二进制哈希码。这类方法，如 CNNH [40],DNNH [41], 和 DHN [15], ，也存在量化偏差问题。为缓解这一问题，HashNet [17]被提出，它设计了一种单阶段哈希学习技术，以降低二值化的量化偏差。该方法用函数 $\tanh(\beta \mathbf{x})$近似 $\text{sign}(\mathbf{x})$ 函数，其中 β 是一个在训练过程中逐渐增大的缩放参数。无限逼近使偏差可以忽略不计，从而有助于提升推荐性能。

从某种意义上说，上述哈希技术仅考虑哈希码之间的一阶相似度。因此，我们在哈希学习中引入基于图的技术，以捕捉高阶汉明相似度。值得一提的是，近期已有一些研究将图神经网络的思想融入哈希，例如 DGCN-BinCF [42], GCNH [43],和 HashGNN [18]。具体而言，HashGNN [18] 在连续空间中构建图神经网络，随后通过直通估计器 [44] 生成汉明空间中的哈希码。然而，哈希步骤会造成信息损失，从而妨碍对内在高阶汉明相似度的捕捉。为弥补这一不足，我们直接在汉明空间中构建了一个图卷积网络来建模高阶相似度。

不同于通过多个相似度图结合不同相似度度量的扩散模型 [45], [46], [47] [48], [49], [50], [51], ，我们的方法通过单个图将汉明相似度从一阶扩展到高阶，并利用高阶相似度学习表征能力更强的哈希码。此外，不同于利用超图中的多模态数据进行表示学习的超图卷积网络 [52],，我们的方法仅基于用户-物品图中的单模态交互信息，因此更容易获取。

3 预备知识

我们首先给出汉明相似度的定义。汉明相似度是在汉明空间中衡量哈希码的相似度指标，强调其与相同比特的数量有关。随后，我们将阐述本研究要解决的问题。

3.1 汉明相似度

在汉明空间中，用户-物品相似度等价于其对应哈希码之间的相似度，称为汉明相似度。给定用户 u 和物品 i，分别将它们的哈希码记作$\mathbf{h}_u \in \{\pm 1\}^K$ 和$\mathbf{h}_i \in \{\pm 1\}^K$，其中 K 表示代码的长度。因此，它们之间的汉明相似度定义如下：

$$\text{sim}(u,i) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}(h_{uk} = h_{ik}), \tag{1}$$

其中， h_{uk} 和 h_{ik} 分别是$\mathbf{h}_u$ 和 $\mathbf{h}_i$的第 k 个比特， $\mathbb{I}(\cdot)$ 表示指示函数：当命题为真时返回1 ，否则返回0 。此外，可以证明，$\text{sim}(u,i)$ 与$\mathbf{h}_u$ 和 $\mathbf{h}_i$中相同比特的数量成正比。因此，我们将公式 [6] 改写为：

$$\text{sim}(u,i) = \frac{1}{2} + \frac{1}{2K}\mathbf{h}_u{}^\top \mathbf{h}_i \propto \mathbf{h}_u{}^\top \mathbf{h}_i, \tag{2}$$

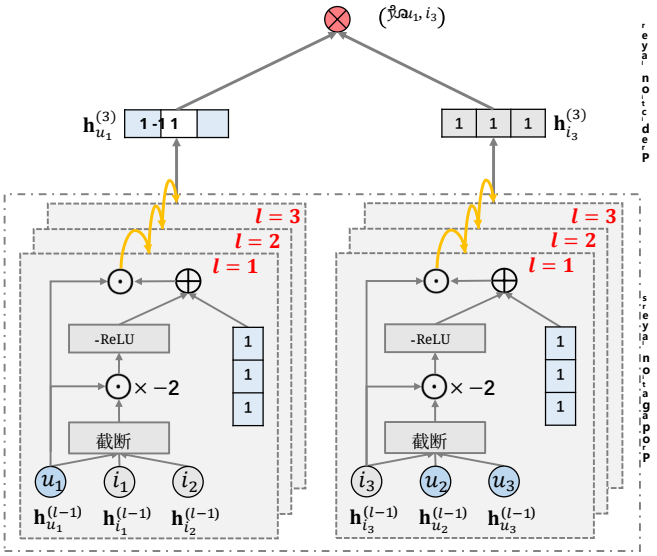


图 2. 我们提出的 HS-GCN 模型示意图（箭头表示信息流）。用户 u_1（左侧）和物品 i_3（右侧）的哈希码通过多个哈希码传播层进行优化。

推导过程从略。基于这一公式，我们直接使用$\mathbf{h}_u$和 $\mathbf{h}_i$ 的内积来衡量 u 和 i之间的相似度。

3.2 问题定义

在本文中，我们旨在解决将用户-物品图中的节点映射为用于推荐的哈希码这一问题。我们将本文要解决的问题定义如下：

- **输入：** 记录历史用户-物品交互的用户-物品二部图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$。其中，节点集由包含 $\mathcal{U}$个 N 用户的用户集和包含 $\mathcal{I}$ 个 M 物品的物品集组成，形式化表示为$\mathcal{V} = \mathcal{U} \cup \mathcal{I}$。此外，边集 $\mathcal{E}$ 包含训练阶段观测到的用户-物品交互。

- **输出：** 所有用户和物品的哈希码，定义为 $\{\mathbf{h}_u^* | u \in \mathcal{U}\} \cup \{\mathbf{h}_i^* | i \in \mathcal{I}\}$。由于$\mathbf{h}_u^{*\top}\mathbf{h}_i^*$表示用户 u 与物品 i之间的相似度分数，因此我们可以据此为给定用户生成物品排序列表，从而解决实际的推荐问题。为了准确衡量相似度分数，我们致力于设计一种用于哈希码传播的图卷积网络，以学习节点哈希码，并捕获用户-物品二部图中蕴含的高阶汉明相似度。

4 方法

本节将详细介绍我们提出的 HS-GCN 模型，如图2所示。具体而言，该模型由三个部分组成：1）初始层，为用户-物品二部图中的节点生成可训练哈希码；2）传播层，显式建模节点对的汉明相似度，并将其注入相应的代码，以增强表示能力；3）预测层，通过计算用户与物品哈希码的内积，对二者的相似度进行评分。此外，我们以矩阵形式给出 HS-GCN 的优化方法，并将 HS-GCN 与一种基于图的哈希学习方法 HashGNN [18]进行比较。

4.1 初始层

初始层用于为用户 u（物品 i）生成初始哈希码$\mathbf{h}_u \in \{\pm 1\}^K$ ($\mathbf{h}_i \in \{\pm 1\}^K$)，其中 K 表示哈希码的长度。显然，由于哈希码的取值是离散的，无法通过梯度下降进行训练。因此，在训练阶段，我们改用代理表示用户和物品。为实现这一目标，我们遵循已有工作 [17] ，将哈希码重新表述为：

$$\mathbf{h}_u = \lim_{\beta \to \infty} \tanh(\beta \mathbf{e}_u), \mathbf{h}_i = \lim_{\beta \to \infty} \tanh(\beta \mathbf{e}_i), \tag{3}$$

其中，$\mathbf{e}_u \in \mathbb{R}^K$ 和$\mathbf{e}_i \in \mathbb{R}^K$ 分别表示用户 u 和物品 i的参数向量。通过缩放 β的值，输出可以近似所需的代码。借助这种代理表示，模型能够解决不适定的梯度问题，并在训练过程中优化表示。当模型达到收敛时，我们可以通过学习到的哈希码，而非其近似值，推断用户与物品之间的交互。

下文中，我们将用户和物品的近似代码作为用户-物品图中节点的初始表示，并详细介绍我们在汉明空间中设计的图卷积操作。为便于记号表示，我们将其记为$\mathbf{h}_u^{(0)}$和$\mathbf{h}_i^{(0)}$ ，分别表示节点在第0层的代码。

4.2 传播层

为了在汉明空间中执行图卷积操作，我们设计了一种新颖的传播层，将这些操作解耦为哈希码聚合和哈希码编码。借助堆叠传播层，我们可以显式建模隐藏在用户-物品图中的一阶和高阶汉明相似度，并将学习到的汉明相似度注入代码，从而优化节点的表示。

本节中，除非特别说明，我们将以用户节点为例详细阐述所提出的模型，并对物品节点执行相同的操作。

4.2.1 哈希码聚合

对于图卷积操作，捕获每个中心节点的局部结构信息至关重要。为此，我们受到先前研究 [2], [3]的启发，提出一种新的聚合方法，在汉明空间中显式建模局部结构信息，从而揭示一阶汉明相似度。形式上，第$(l+1)$个传播层的聚合可定义为：

$$\mathbf{m}_u^{(l)} = f\left(\mathbf{h}_u^{(l)}, \{\mathbf{h}_i^{(l)} | i \in \mathcal{N}_u\}\right), \tag{4}$$

其中，$\mathbf{h}_u^{(l)} \in \{\pm 1\}^K$ 和$\mathbf{h}_i^{(l)} \in \{\pm 1\}^K$ 分别递归表示用户 u 和物品 i 在第 l层传播后的哈希码。此外， $\mathcal{N}_u$ 表示与用户 u直接发生过交互的物品邻居集合。利用函数 $f(\cdot)$，我们可以捕获信息信号$\mathbf{m}_u^{(l)}$；该信号是通过将节点与其邻居结合而提炼得到的。

然而，这一函数在汉明空间中的实现并不直接。在本研究中，我们分两步实现它。具体而言，考虑到

加权矩阵的变换在汉明空间中并不适用，我们首先采用非加权求和来整合哈希代码的逐位信号。因此，我们将汉明空间中的聚合改写为：

$$\mathbf{m}_u^{(l)} = f(\mathbf{h}_u^{(l)} + \sum_{i \in \mathcal{N}_u} \mathbf{h}_i^{(l)}). \tag{5}$$

接下来，由于哈希码受二进制值（即 +1和 -1）的约束，我们提取中心节点及其邻居的哈希码中逐比特占主导的值。这与欧几里得空间中的图操作不同，例如均值池化和最大池化。我们将此归因于哈希码服从二项分布。因此，我们旨在为聚合后的代码设计逐比特函数 $f(\cdot)$，并将 $\mathbf{m}_u^{(l)}$中的每一比特作为符号，以反映与邻居相关联的 u 的代码中对应比特的占主导值。由此，为了兼顾函数的连续性，我们采用区间受限的截断函数实现$f(\cdot)$，如下所示：

$$\mathbf{m}_u^{(l)} = \text{clamp}(\mathbf{x}) = \begin{cases} 1, \text{ if } \mathbf{x} > 1 \\ \mathbf{x}, \text{ if } -1 \leq \mathbf{x} \leq 1 \\ -1, \text{ if } \mathbf{x} < -1 \end{cases}, \tag{6}$$

其中 $\mathbf{x} = (\mathbf{h}_u^{(l)} + \sum_{i \in \mathcal{N}_u} \mathbf{h}_i^{(l)})$。借助截断函数的特性，我们可以保证 $\mathbf{m}_u^{(l)} \in \{-1, 0, 1\}^K$。对于每个节点，该函数旨在反映其邻居哈希码的分布。它能够确定中心节点的值，使其与邻居相似。以哈希码中的一比特为例，当邻居哈希码中对应比特的大多数为 -1 时，该函数会在这一比特上输出 -1，反之亦然。总之，我们提出了一种汉明空间中的聚合操作，将中心节点及其邻居的代码整合起来，并提炼出能够反映哈希码逐比特值分布的信号。

4.2.2 哈希码编码

从局部结构中获得聚合后的代码后，我们将信息编码到节点中，以保留这些节点与相连节点之间的一阶汉明相似度。对于每个节点，尽可能减少其代码与邻居代码中对应比特不同的总数至关重要。因此，我们首先比较用户 u （即，$\mathbf{h}_u^{(l)}$）的代码与获得的符号（即，$\mathbf{m}_u^{(l)}$），以计算不同比特的数量，如下所示：

$$\mathbf{d}_u^{(l)} = \mathbf{h}_u^{(l)} \odot \mathbf{m}_u^{(l)}, \tag{7}$$

其中，$\odot$ 表示逐元素乘积，$\mathbf{d}_u^{(l)} \in \{-1, 0, 1\}^K$是一个由逐比特比较结果组成的向量。具体而言，输出中的值 -1 表示对应比特与大多数邻居的比特不同；而值 1表示该节点的比特值与其周围大多数节点相同。此外，当值等于 0时，表示对应比特处于中间状态。

然后，我们可以根据所得向量中的每个值来细化用户 u 的代码。考虑到向量中存在 0 这一元素，通过对两个向量进行逐元素乘积来实现这一操作并不容易，因为除二进制值之外，该操作还可能输出零（即，$+1$ 和 -1）。因此，我们设计了一个新函数，以避免值0带来的负面影响。其形式化定义如下：

$$\mathbf{c}_u^{(l)} = -\text{ReLU}(-2 \times \mathbf{d}_u^{(l)}) + ones, \tag{8}$$

其中，$ones = \{1\}^K$ 是根据二元约束调整输出的向量。基于这一变换函数，数值会在 $\{-1, 1\}$内进行调整。请注意，我们将数值0 转换为1 ，而不是 -1，使得$\mathbf{c}_u^{(l)}$中的-1 值能够准确表示节点与其邻居之间明确无疑的比特差异。这将提升后续逐位细化的可靠性。

利用调整后的向量，我们可以通过最大化哈希码$\mathbf{h}_u^{(l)}$与其邻居的相似度来细化该哈希码。由于$\mathbf{c}_u^{(l)}$表示其邻居中多数节点所对应的不同比特，我们利用$\mathbf{c}_u^{(l)}$通过逐元素乘积来细化$\mathbf{h}_u^{(l)}$的比特，具体而言：

$$\mathbf{h}_u^{(l+1)} = \mathbf{c}_u^{(l)} \odot \mathbf{h}_u^{(l)}, \tag{9}$$

其中，$\mathbf{h}_u^{(l+1)}$表示在第（l+1）个传播层学习到的用户 u 的哈希码。类似地，我们可以通过传播自身及其相连用户的哈希码，得到物品 i 的哈希码$\mathbf{h}_i^{(l+1)}$ 。总之，所设计的哈希码传播层有助于利用一阶相似度信息，在汉明空间中关联用户和物品的哈希码。

4.2.3 矩阵形式中的哈希码传播规则

为了全面展示哈希码传播架构并便于批量实现，我们将逐层传播规则表示为矩阵形式：

$$\mathbf{H}^{(l+1)} = \Big(-\text{ReLU}\big(-2 \times \text{clamp}\big((\mathbf{A} + \mathbf{I})\mathbf{H}^{(l)}\big) \odot \mathbf{H}^{(l)}\big) + Ones\Big) \odot \mathbf{H}^{(l)}, \tag{10}$$

其中，$\mathbf{H}^{(l)} \in \{\pm 1\}^{(N+M) \times K}$ 表示经过 l 步传播后获得的用户和物品的哈希码。$\mathbf{H}^{(0)}$ 表示初始层之后的初始哈希码矩阵，其中 $\mathbf{h}_u^{(0)} = \mathbf{h}_u$ 和 $\mathbf{h}_i^{(0)} = \mathbf{h}_{i+}$ $\mathbf{I}$ 表示单位矩阵，$Ones \in \{1\}^{(N+M) \times K}$ 表示全 $\mathbf{1}$ 矩阵。$\mathbf{A}$表示用户-物品图的邻接矩阵，其定义如下：

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix}, \tag{11}$$

其中，$\mathbf{R} \in \{0, 1\}^{N \times M}$ 表示用户-物品交互矩阵，$\mathbf{0}$ 表示全零矩阵。通过实现矩阵形式的传播规则，我们可以以相当高效的方式同时更新所有用户和物品的哈希码。此外，该规则还帮助我们省略节点采样过程；为了使图卷积网络适用于大规模图，该过程通常是必需的。

4.3 预测层

我们堆叠多个传播层，以探索哈希码之间的高阶相似性，这对于升级哈希码并最终估计用户-物品对之间的相关性至关重要。具体而言，我们将 L 定义为

最终传播层的深度，用户 u 的哈希码递归定义如下：

$$\begin{cases} \mathbf{h}_u^{(1)} = \mathbf{c}_u^{(0)} \odot \mathbf{h}_u, \\ \mathbf{h}_u^{(2)} = \mathbf{c}_u^{(1)} \odot \mathbf{h}_u^{(1)} = \mathbf{c}_u^{(1)} \odot \mathbf{c}_u^{(0)} \odot \mathbf{h}_u, \\ \cdots\cdots \\ \mathbf{h}_u^{(L)} = \mathbf{c}_u^{(L-1)} \odot \mathbf{h}_u^{(L-1)} = \mathbf{c}_u^{(L-1)} \odot \cdots \odot \mathbf{c}_u^{(0)} \odot \mathbf{h}_u. \end{cases} \tag{12}$$

这表明，我们的模型通过将初始哈希码与一系列细化向量逐元素相乘，得到最终哈希码。随后，这些细化向量逐层调制用户 u哈希码中的比特信号。因此，高阶汉明相似度被融入哈希码学习过程。请注意，根据公式（12），每一层的哈希码长度均为 K 。类似地，我们可以得到物品 i 在第L层的哈希码。

经过 L 个传播层后，我们得到用户 u的多个哈希码，记为 $\{\mathbf{h}_u^{(1)}, \cdots, \mathbf{h}_u^{(L)}\}$。值得注意的是，我们没有将它们拼接起来作为用户的最终表示，而基于图的推荐模型 [2], [26]中普遍采用了这种做法。原因在于，拼接后代码的长度可能会损害计算效率并增加内存空间占用。作为替代，我们采用最后一层的哈希码（即$\mathbf{h}_u^{(L)}$和$\mathbf{h}_i^{(L)}$）作为用户 u 和物品 i 的表示来预测二者的交互，形式化地，

$$\mathbf{h}_u^* = \mathbf{h}_u^{(L)}, \quad \mathbf{h}_i^* = \mathbf{h}_i^{(L)}. \tag{13}$$

最后，我们通过内积估计目标用户与物品的哈希代码之间的相似度，这也可以等价地视为用户对该物品的偏好：

$$\hat{y}_{ui} = \mathbf{h}_u^{*\top} \mathbf{h}_i^*. \tag{14}$$

4.4 优化

在本文中，我们基于用户的隐式反馈（如观察记录和购买行为） [53], [54]来优化所提出的模型。与显式评分相比，隐式反馈更容易收集，但由于缺少负反馈，在探索用户偏好方面更具挑战性。为了通过最大似然估计（MLE）优化模型，我们将似然函数 $P(r_{ui}|\mathbf{h}_u^*, \mathbf{h}_i^*)$ 表述为：给定用户 u 和物品 i 的可训练代码 $\mathbf{h}_u^*$ 和 $\mathbf{h}_i^*$，二者发生交互的概率。因此，其最优目标可具体表示为以下成对逻辑函数：

$$P(r_{ui}|\mathbf{h}_u^*, \mathbf{h}_i^*) = \begin{cases} \sigma(\hat{y}_{ui}), & r_{ui} = 1, \\ 1 - \sigma(\hat{y}_{ui}), & r_{ui} = 0, \end{cases} \tag{15}$$

其中 $\sigma(\cdot)$ 是标准的 Sigmoid函数。并且，$\hat{y}_{ui} = \mathbf{h}_u^{*\top} \mathbf{h}_i^*$ 是$\mathbf{h}_u^*$和$\mathbf{h}_i^*$之间估计的汉明相似度。在这种情况下，哈希码越相似，条件概率 $P(1|\mathbf{h}_u^*, \mathbf{h}_i^*)$ 就越大，反之亦然。因此，公式(15)是基于哈希码对进行分类的逻辑回归分类器的合理扩展——问题。此外，我们可以通过最小化以下损失函数来优化参数：

$$\mathcal{L}_{cross} = -\sum_{r_{ui} \in \mathbf{R}} r_{ui} \log(\sigma(\hat{y}_{ui})) + (1 - r_{ui}) \log(1 - \sigma(\hat{y}_{ui})), \tag{16}$$

其中，$\mathbf{R}$ 是前文所述的用户-物品交互矩阵，用于记录真实隐式反馈。采用该损失函数后，最优目标转为在训练阶段重构已观测交互。

通常，公式（16）能够有效学习用户和物品的有用哈希码，并保证发生交互的用户-物品对具有相似的哈希码。然而，候选物品之间的排序往往比其绝对分数更为重要，因为推荐系统主要依据物品的排序向用户推荐物品。对于隐式反馈数据，可以通过从交互矩阵 $\mathbf{R}$ [1]中采样来获得排序关系。具体而言，我们假设，能够更好反映用户偏好的已观测交互，其估计值应高于未观测交互。受此启发，我们引入了如下排序顺序增强损失函数：

$$\mathcal{L}_{rank} = \sum_{(u,i,j) \in \mathcal{D}} \max(0, -\sigma(\hat{y}_{ui}) + \sigma(\hat{y}_{uj}) + \alpha), \tag{17}$$

其中，$\mathcal{D} = \{(u, i, j) \mid (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ 表示三元组训练数据，$\mathcal{R}^+$ 表示已观测交互，$\mathcal{R}^-$ 表示未观测交互；α 表示边界参数，用于控制已观测交互和未观测交互。通过最小化 $\mathcal{L}_{rank}$，发生交互的物品将在汉明空间中比未发生交互的物品更紧密地聚集在用户 u 周围 u 。

综合上述因素，待最小化的总体损失函数表示如下：

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{cross} + \lambda_1 \mathcal{L}_{rank} + \lambda_2 \|\mathbf{E}\|_2^2 \\ &= -\sum_{r_{ui} \in \mathbf{R}} r_{ui} \log(\sigma(\hat{y}_{ui})) + (1 - r_{ui}) \log(1 - \sigma(\hat{y}_{ui})) \\ &\quad + \lambda_1 \sum_{(u,i,j) \in \mathcal{D}} \max(0, -\sigma(\hat{y}_{ui}) + \sigma(\hat{y}_{uj}) + \alpha) + \lambda_2 \|\mathbf{E}\|_2^2, \end{aligned} \tag{18}$$

其中，λ_1 是权衡参数，用于平衡熵与排序损失之间的比例。注意，成对排序损失和逐点重构损失对于模型优化都是必不可少的；$\mathbf{E} \in \mathbb{R}^{(N+M) \times K}$ 表示初始层中的实值参数矩阵，作为可训练的模型参数；λ_2 控制 L_2 正则化强度，以防止过拟合。此外，我们采用小批量Adam [55]优化模型并更新模型参数。具体而言，对于一批随机采样的三元组$(u, i, j) \in \mathcal{D}$，我们获得它们的最终哈希码$\mathbf{h}_u^*$、$\mathbf{h}_i^*$和$\mathbf{h}_j^*$，然后利用损失函数的梯度更新模型参数。

4.5 与HashGNN的比较

我们将哈希码传播层与 HashGNN [18], 进行比较，后者采用欧几里得空间中的嵌入传播来学习哈希码。HashGNN 设计了

图卷积操作并非作用于哈希码 $\mathbf{h}_u^{(l)}$（或 $\mathbf{h}_i^{(l)}$），而是作用于其实值参数近似 $\mathbf{e}_u^{(l)}$（或 $\mathbf{e}_i^{(l)}$）。显然，二者之间存在量化偏差，即 $\boldsymbol{e}_u^{(l)} = \mathbf{h}_u^{(l)} + \epsilon_u^{(l)}$。因此，HashGNN 的传播层可表述为：

$$\mathbf{e}_u^{(l+1)} = \tanh\big(\mathbf{W}^{(l)} \cdot \text{mean}\big(\mathbf{h}_u^{(l)} + \epsilon_u^{(l)} + \sum_{i \in \mathcal{N}_u} (\mathbf{h}_i^{(l)} + \epsilon_i^{(l)})\big)\big), \tag{19}$$

其中，量化偏差会随着聚合而累积。非线性双曲正切函数和权重矩阵 $\mathbf{W}^{(l)} \in \mathbb{R}^{K \times K}$ 会在$\mathbf{e}_u^{(l+1)}$与

$\mathbf{h}_u^{(l+1)}$之间引入新的偏差，导致最终哈希码无法获得高阶汉明相似度。与 HashGNN 相比，我们提出的 HS-GCN 模型在不引入量化偏差的情况下实现了哈希码的传播，从而有效捕获哈希码之间的高阶汉明相似度。

5 实验设置

本节首先介绍评估数据集，然后介绍实验设置，最后详细阐述基线方法。

5.1 数据集

为了验证所提出模型的有效性，我们在六个公开数据集上进行了实验：MovieLens[1]，、Yelp[2]，、亚马逊[3]，、Gowalla、Pinterest 和 Net- flix[4]。表1 总结了所评估数据集的详细统计信息。

MovieLens： 这是一个广泛使用的电影评分数据集，我们在实验中采用了 1M 版本。与 [56]，类似，我们将评分转换为隐式反馈，使每个用户-物品对的标签为1或0，表示用户是否对该电影进行过评分。其他数据集也采用类似方式处理。

Yelp： Yelp 是一个知名的商业在线评论平台，涵盖餐厅、酒吧和水疗中心等。我们选取了最新版本中的数据集，并采用 20核心设置，以获得更稠密的数据集。

亚马逊： Amazon-review 是一个广受欢迎的商品推荐数据集 [57]。我们从该数据集中选取了 Amazon-book。同样，我们采用20核心设置，以确保每个用户和物品至少有 20 次交互。**Gowalla**：这是从 Gowalla 获取的签到数据集 [58]，用户通过签到分享自己的位置。为确保数据集质量，我们采用 10 核心设置，即保留至少有十次交互的用户和物品，与 [2]类似。

Pinterest：此隐式反馈数据集由 [59] 构建，用于评估图像推荐。我们采用了由 [60]分享的处理后版本。

Netflix：这是 Netflix 挑战赛所使用的大规模电影评分数据集。我们采用了 20核心设置，以获得更稠密的数据集。

1. https://grouplens.org/datasets/movielens/。 2. https://www.yelp.com/dataset。 3. http://jmcauley.ucsd.edu/data/amazon/。 4. http://www.netflixprize.com/。

表 1 数据集统计信息。

数据集	#用户	#物品	#交互	密度
MovieLens	6,040	3,706	1,000,209	4.47%
Yelp	40,500	58,755	2,024,283	0.09%
亚马逊	36,783	77,379	2,402,416	0.08%
Gowalla	29,585	40,981	1,027,370	0.09%
Pinterest	55,187	9,916	1,500,809	0.27%
Netflix	429,584	17,764	99,884,887	1.31%

对于每个数据集，我们将其隐式反馈随机划分为两部分：70% 用于训练，其余30% 用于测试。此外，我们从训练集中随机选取10% 的交互作为验证集，用于超参数调优。对于训练集中的每个已观测到的用户-物品交互，我们将其视为一个正样本，然后采用负采样策略，为其配对五个用户在训练集中未交互过的负物品。

5.2 实验设置

评估指标。对于测试集中的每位用户，我们将所有未与其发生交互的物品视为负样本。随后，推荐方法会预测该用户对除训练集中物品之外的所有物品的偏好分数（例如，汉明相似度）。为了衡量 Top-K 推荐和偏好排序的效果，我们采用 HR@K （命中率） [60] 和 NDCG@K （归一化折损累计增益）作为评估指标。其中，NDCG@K 的正式定义如下：

$$\text{NDCG@}K = \frac{\text{DCG}_K}{\text{IDCG}_K}, \text{ and } \text{DCG}_K = \sum_{i=1}^{K} \frac{2^{r_{ui}} - 1}{\log_2(1+i)},$$

其中，IDCG 是理想 DCG，r_{ui} 表示第 i个推荐物品的交互状态。默认情况下，我们设置$K = 50$和100。我们报告了测试集中所有用户的平均指标。

模型实现。我们使用开发工具 PyTorch[5] 和 PyTorch Geometric[6]实现了 HS-GCN 模型。我们将 HS-GCN L 的深度设为 2，以建模二阶汉明相似度。哈希码长度设为64。我们使用 Adam优化器，以 3,000 的批大小优化模型。此外，我们采用广受欢迎的 Xavier [61] 方法初始化模型中的所有参数。我们根据验证集的结果，通过网格搜索调整超参数：学习率在 {1e-4、3e-4、1e-3、3e-3}中进行搜索，最终设为 3e-4；权衡系数 λ_1 在 0.1 至 1 之间以 0.1 为步长进行搜索，最终设为 0.1； L_2 归一化的系数 λ_2 在 {1e-8、1e-7、 $\cdots$、1}中进行搜索，最优值为 1e-7。间隔参数 α 固定为0.2。此外，如果验证数据上的 HR@K 连续 10 个训练轮次没有提升，则采用早停。

5. https://pytorch.org/. 6. https://pytorch-geometric.readthedocs.io/en/latest/.

5.3 基线比较

为展示所提方法的有效性，我们将其性能与一系列最先进的、基于哈希学习的模型进行了比较（即 DHN、Hash ste、HashNet 和 HashGNN）。除这些方法外，我们还引入了几种实值推荐模型（即矩阵分解、BiNE、PinSage 和 GraphSAGE），以验证 HS-GCN 能够取得具有竞争力的结果。

- **DCF** [6]：这是首个基于评分矩阵直接优化哈希码的方法。在我们的实验中，我们对 DCF 进行了调整，以处理交互矩阵。

- **DFM** [5]：这是一种采用离散参数化的评分预测因子分解机。我们对其进行了调整，使其能够在隐式数据上进行训练。

- **DHN** [15]：这是一种用于相似图像推荐的流行深度哈希方法。它通过控制量化偏差来学习高质量的哈希码。我们将原始 AlexNet [62]框架替换为图卷积网络（图卷积网络），从而将其调整为适用于用户-物品推荐。

- **Hash_ste**：这是一种基于直通估计器 [44], 的有效端到端哈希学习方法，可通过直接复制近似梯度来优化离散变量。

- **HashNet** [17]：这是一种最先进的深度哈希方法，它设计了一种连续可微函数来近似符号函数。类似地，我们使用图卷积网络替代了 HashNet 中的 AlexNet。

- **GCNH** [43]：这是一种基于图的哈希方法，它引入了一种新颖且高效的非对称图卷积网络，随后通过二值化步骤学习保持相似性的哈希码。我们将用户和物品 ID 信息作为输入特征，使其适用于推荐任务。

- **DGCN-BinCF** [42]：这是一种基于图的哈希方法。该方法放宽了二元约束，使连续优化成为可能，从而将图卷积网络中的排序信息提炼到哈希码中。

- **HashGNN** [18]：这是一种基于图的哈希模型，其中利用图神经网络生成高质量的哈希码。

- **MF** [63]：矩阵分解是推荐任务中最常见的嵌入模型，它仅利用用户-物品直接交互进行推荐。

- **BiNE** [64]：该嵌入模型基于二部图采用有偏随机游走进行表示学习，并使用优化框架处理显式评分和隐式反馈。

- **PinSage** [25]：PinSage 是一种图嵌入模型。它结合了高效的随机游走和物品-物品图上的图卷积，同时利用图结构和节点特征信息。在本文中，我们将其应用于用户-物品交互图。

- **GraphSage** [65]：作为一种著名的图嵌入模型，它是一种通用的归纳式框架，通过对节点局部邻域中的特征进行采样和聚合来学习嵌入。

基线方法实现。为确保公平比较，除公开提供实现的 DCF[7],DFM[8],GCNH[9],和 BiNE[10]外，所有基线方法均使用 PyTorch 和 PyTorch Geometric 实现。若无特别说明，嵌入或哈希码的默认维度为 64。对于所有基于图的基线方法，我们采用由 Xavier 初始化的两层图卷积网络，并使用经过精心选择的批大小进行小批量训练，采用 Adam优化器进行模型优化。学习率从 {1e-4、3e-4、1e-3、3e-3}中调优。对于 HashGNN，我们按照原论文 [18]设置图层架构和超参数。对于 HashNet，我们将双曲正切(βx)的尺度参数 β 初始化为 1，并按照文献 [17]的建议，在经过固定训练轮次后以指数方式增大该参数。对于 DGCN-BinCF、PinSage 和 Graph-Sage，我们根据原论文 [25], [42], [64], [65], 中的默认架构实现，并根据模型在验证集上的表现调优其超参数。

6 实验结果

为验证所提出方法的有效性，我们开展了全面实验，从性能、效率和稀疏性问题等方面将所提出的 HS-GCN 模型与基线方法进行比较。此外，为评估 HS-GCN 各组件的有效性，我们针对传播层、排序损失和丢弃技术等关键组件进行了消融实验。

6.1 总体实验

我们首先比较了所有方法的性能，随后分析了这些方法的效率。最后，我们探究了在稀疏场景下对高阶相似度进行建模的有效性。

6.1.1 性能比较

表 2 展示了我们的方法和基线方法在所实验数据集上的结果。此外，表 2 报告了我们提出的方法与表现最强的哈希基线方法（以下划线标出）之间的性能提升和统计显著性检验结果。从上到下观察表2所示结果，我们得出了以下主要发现：

- DCF 和 DFM 在隐式反馈数据集上的表现不佳。这反映了基于显式反馈设计的哈希方法存在局限性。DHN 在所有数据集上的表现都较差。这表明，两阶段哈希方法不足以捕捉汉明空间中用户与物品之间的一阶相似度，这是由量化偏差造成的。Hash ste 在所有情况下的表现都优于 DHN，显示了直接哈希学习策略的优势。然而，Hash ste 未能完全解决直通估计器与符号函数之间的量化偏差。

–

- 与 DHN 和 Hash ste 相比，HashNet 有效消除了量化偏差，其更优的性能验证了 HashNet 能够捕捉一阶

7. https://github.com/hanwangzhang/Discrete-Collaborative-Filtering。8. https://github.com/hanliu95/DFM。9. https://github.com/zxJohnFly/GCN。10. https://github.com/clhchtcjj/BiNE。

表2 六个数据集上的整体性能比较。提升百分比和p值分别表示HS-GCN相较于HashGNN的相对提升（%）和t检验结果。

方法	MovieLens		Yelp		亚马逊		Gowalla		Pinterest		Netflix	
	HR@50	HR@100	HR@50	HR@100	HR@50	HR@100	HR@50	HR@100	HR@50	HR@100	HR@50	HR@100
DCF	0.0469	0.0910	0.0143	0.0281	0.0145	0.0288	0.1376	0.1471	0.0723	0.0840	0.0667	0.1143
DFM	0.0549	0.1068	0.0163	0.0310	0.0171	0.0336	0.1753	0.1820	0.0849	0.0931	0.0819	0.1330
DHN	0.0782	0.1393	0.0228	0.0430	0.0238	0.0448	0.2281	0.2415	0.1200	0.1328	0.1084	0.1813
Hash ste	0.0955	0.1684	0.0349	0.0620	0.0327	0.0557	0.3268	0.3606	0.1125	0.2032	0.1211	0.2007
HashNet	0.1359	0.2232	0.0364	0.0651	0.0353	0.0599	0.3370	0.3830	0.1539	0.2730	0.1463	0.2337
GCNH	0.1552	0.2488	0.0389	0.0714	0.0394	0.0636	0.3979	0.4209	0.1847	0.3129	0.1515	0.2402
DGCN-BinCF	0.1554	0.2546	0.0358	0.0666	0.0373	0.0623	0.3420	0.3853	0.1834	0.3112	0.1744	0.2883
HashGNN	0.1598	0.2557	0.0447	0.0774	0.0363	0.0611	0.3932	0.4329	0.1947	0.3214	0.2058	0.3185
HS-GCN	**0.2052**	**0.3169**	**0.0497**	**0.0883**	**0.0523**	**0.0885**	**0.4084**	**0.4480**	**0.2066**	**0.3322**	**0.2192**	**0.3432**
MF	0.1332	0.2042	0.0319	0.0513	0.0349	0.0566	0.3360	0.3675	0.1400	0.2472	0.1981	0.3042
BiNE	0.1308	0.1868	0.0345	0.0546	0.0335	0.0560	0.3603	0.3812	0.1526	0.2582	0.2097	0.3346
PinSage	0.1587	0.2501	0.0503	0.0812	0.0402	0.0656	0.3791	0.4131	0.1837	0.3008	0.2279	0.3585
GraphSage	0.2132	0.3326	0.0581	0.0942	0.0408	0.0672	0.3894	0.4305	0.1919	0.3162	0.2334	0.3616
提升百分比	28.41%	23.93%	11.19%	14.08%	44.08%	44.84%	3.87%	3.49%	6.11%	3.36%	6.51%	7.76%
p值	2.77e-4	7.01e-4	5.62e-3	1.95e-4	1.04e-3	1.27e-4	1.81e-4	1.39e-4	1.37e-3	5.08e-4	1.35e-4	2.04e-4

方法	MovieLens		Yelp		亚马逊		Gowalla		Pinterest		Netflix	
	N@50	N@100	N@50	N@100	N@50	N@100	N@50	N@100	N@50	N@100	N@50	N@100
DCF	0.0641	0.0785	0.0097	0.0137	0.0113	0.0173	0.2066	0.2599	0.0756	0.1109	0.0862	0.1014
DFM	0.0767	0.0937	0.0105	0.0165	0.0131	0.0187	0.2616	0.3041	0.0912	0.1242	0.1019	0.1294
DHN	0.1038	0.1251	0.0149	0.0226	0.0176	0.0264	0.3440	0.4080	0.1212	0.1710	0.1373	0.1660
Hash ste	0.1305	0.1608	0.0225	0.0322	0.0263	0.0354	0.4692	0.5114	0.1248	0.1748	0.1515	0.1817
HashNet	0.1940	0.2233	0.0232	0.0335	0.0281	0.0378	0.4825	0.5245	0.1441	0.1877	0.1710	0.1900
GCNH	0.1443	0.2283	0.0294	0.0450	0.0319	0.0408	0.4957	0.5490	0.1616	0.2105	0.1723	0.2049
DGCN-BinCF	0.2298	0.2688	0.0268	0.0345	0.0302	0.0390	0.4646	0.5234	0.1436	0.2072	0.2230	0.2745
HashGNN	0.2335	0.2630	0.0308	0.0430	0.0273	0.0374	0.5155	0.5512	0.1693	0.2198	0.3056	0.3502
HS-GCN	**0.3081**	**0.3376**	**0.0336**	**0.0483**	**0.0436**	**0.0585**	**0.5351**	**0.5696**	**0.1786**	**0.2325**	**0.3883**	**0.4501**
MF	0.2023	0.2216	0.0240	0.0312	0.0303	0.0392	0.4977	0.5221	0.1029	0.1514	0.2469	0.2883
BiNE	0.1370	0.1765	0.0231	0.0320	0.0278	0.0370	0.5104	0.5346	0.1169	0.1858	0.2584	0.2999
PinSage	0.2316	0.2517	0.0380	0.0496	0.0337	0.0438	0.5136	0.5447	0.1536	0.2063	0.2923	0.3261
GraphSage	0.3195	0.3490	0.0434	0.0570	0.0349	0.0457	0.5368	0.5581	0.1643	0.2124	0.3703	0.4114
提升百分比	31.95%	28.37%	9.09%	12.33%	59.71%	56.42%	3.80%	3.34%	5.49%	5.78%	27.1%	28.5%
p值	3.28e-3	2.87e-3	1.41e-2	8.47e-4	3.51e-4	1.44e-4	2.21e-3	4.09e-4	4.11e-3	1.69e-3	4.15e-3	3.72e-3

汉明相似度更加准确。然而，这些方法都没有在哈希码学习过程中明确建模高阶汉明相似度，因此导致结果并非最优。

•基于图神经网络的哈希方法显著优于基于深度学习的哈希方法。这是合理的，因为图神经网络在二值化步骤之前提升了用户和物品连续表示的质量，从而间接丰富了最终哈希码。在基于图神经网络的方法中，HashGNN 表现更优，因为它是真正意义上的端到端哈希学习方法，能够通过梯度复制解决符号函数的反向传播问题。相比之下，DGCN-BinCF 和 GCNH 都将符号函数作为额外的二值化步骤，因此不可避免地会产生量化损失。

•显然，在哈希方法中，HS-GCN 的性能最佳。具体而言，与最强的哈希基线方法 HashGNN 相比，HS-GCN 在 HR@K 上的平均提升幅度分别为 26.17%、12.64%、44.46%、3.68%、4.74%和 7.14%；在 NDCG@K 上的平均提升幅度分别为 30.16%、10.71%、58.07%、3.57%、5.64%和 27.8%，对应 MovieLens、Yelp、亚马逊、Gowalla、Pinterest 和 Netflix。其原因在于，HS-GCN 通过直接在汉明空间中构建图卷积网络，能够捕捉哈希码的高阶相似度，而 HashGNN 仅利用一阶相似度来指导哈希学习。此外，我们进行了交叉验证 t 检验，p值 < 0.05 表明，HS-GCN 相较于最强哈希基线方法的性能提升具有统计显著性。

• HS-GCN 的表现显著优于基于嵌入的基线方法 MF 和 BiNE。此外，与图嵌入模型 GraphSage 和 PinSage 相比，我们的模型取得了相当的性能。GraphSage 的表现优于大多数基于哈希的方法，因为它利用图卷积网络学习用户和物品的实值嵌入，其表征能力优于二进制码。值得一提的是，我们在

亚马逊上的表现优于表现最佳的图嵌入模型 GraphSage。这也验证了捕获高阶相似度对于哈希学习的重要性。

6.1.2 效率比较

本节分析了 HS-GCN 与基线方法相比的效率。具体而言，我们重点考察了这些模型在训练和测试阶段的效率，并记录了其运行时间。为确保公平比较，所有模型均在配备 NVIDIA TITAN Xp（12GB 显存）和 Python3.7 的 Ubuntu 16.04.5 系统上训练，直至收敛；并在配备 Intel Core i7、16GB RAM 和 Python3.7 的 Windows 10 系统上进行测试。在训练过程中，所有模型的超参数（例如训练轮次、批大小和学习率）均采用最优设置。尽管我们对所有方法都采用线性扫描作为测试技术来推荐物品，但值得一提的是，多级索引等技术 [66] 可以进一步加速哈希方法的测试。表3总结了不同规模亚马逊数据集上的结果，其中“1/8Amazon”是从完整数据集中随机抽取八分之一的交互构建而成，其余规模的数据集也以同样方式构建。其他数据集上也有类似的效率比较，因此为节省篇幅，我们省略了相关结果。关于这些实验，我们发现：

•我们的模型所需的训练时间远少于其他方法。与矩阵分解相比，HS-GCN 和其他基于图的模型在训练时更加高效，因为它们在实验中能够快速收敛。这是合理的，因为在小批量中优化交互对时，图结构会将间接连接的用户和物品纳入其中。与其他基于图的模型相比，HS-GCN 在训练过程中表现得更加高效，因为它具有最为简洁的模型大小，与矩阵分解相同。具体而言，除初始实值矩阵 $\mathbf{E}$ 外，HS-GCN 不引入任何额外参数，而其他模型会在每个传播层中引入可训练权重矩阵作为传递参数。

•与大多数哈希模型类似，在测试时间方面，与连续推荐模型相比，HS-GCN 实现了显著的加速。通过联合分析表 2 和表 3，我们可以看到，HS-GCN 不仅能够取得与最先进的连续模型相比具有竞争力的性能，而且具备较高的时间效率。这验证了 HS-GCN 是一种适用于大规模网络服务的可操作方案，能够大幅降低其推荐系统的计算成本 [67]。

• 在不同规模的数据集上，HS-GCN 的训练效率比相较于 HashGNN 稳定在约 2.4 倍，测试效率比相较于 GraphSage 稳定在约 4.9 倍。从表3可以看出，随着数据集规模逐渐增大，我们的方法在训练和测试方面的效率比均呈上升趋势。

6.1.3 不同交互稀疏度级别下的性能比较

稀疏性问题通常会限制推荐系统的表达能力，因为不活跃用户缺乏足够的

表3 采用64 比特时，不同规模亚马逊数据集上的训练/测试时间。结果分别以小时和秒为单位。训练/测试效率比是通过比较 HS-GCN 与 HashGNN/GraphSage 计算得出的，其中 HashGNN/GraphSage 是表现最佳的基于图的哈希/嵌入基线。

方法	1/8 亚马逊		1/4 亚马逊		1/2 亚马逊		亚马逊	
	训练	测试	训练	测试	训练	测试	训练	Test
HS-GCN	1.3	3.0	2.4	6.1	4.6	13.3	9.1	24.8
DHN	1.4	3.0	2.7	6.1	5.4	13.2	11.2	24.9
Hash ste	1.5	3.0	3.0	6.0	6.1	13.3	12.6	24.7
HashNet	2.1	3.0	4.3	6.2	9.0	13.1	18.3	24.8
GCNH	1.9	3.1	3.6	6.1	7.2	13.3	14.8	25.0
DGCN-BinCF	1.6	3.1	3.2	6.1	6.2	13.3	12.6	24.8
HashGNN	2.7	3.0	5.6	6.1	11.1	13.2	23.3	24.9
MF	2.3	14.1	4.3	30.5	9.1	66.2	18.9	127.3
BiNE	1.4	14.2	2.6	30.2	5.4	66.2	10.9	127.0
PinSage	1.6	14.1	3.2	30.4	6.5	66.0	13.1	127.2
GraphSage	1.5	14.1	2.9	30.1	6.0	65.9	12.1	127.0
效率比	2.1	4.7	2.3	4.9	2.4	5.0	2.6	5.1

交互来生成高质量的表示。这尤其阻碍了仅基于一阶汉明相似度的学习哈希方法。在本节中，我们尝试回答这样一个问题：探索高阶汉明相似度是否有助于缓解这一问题。

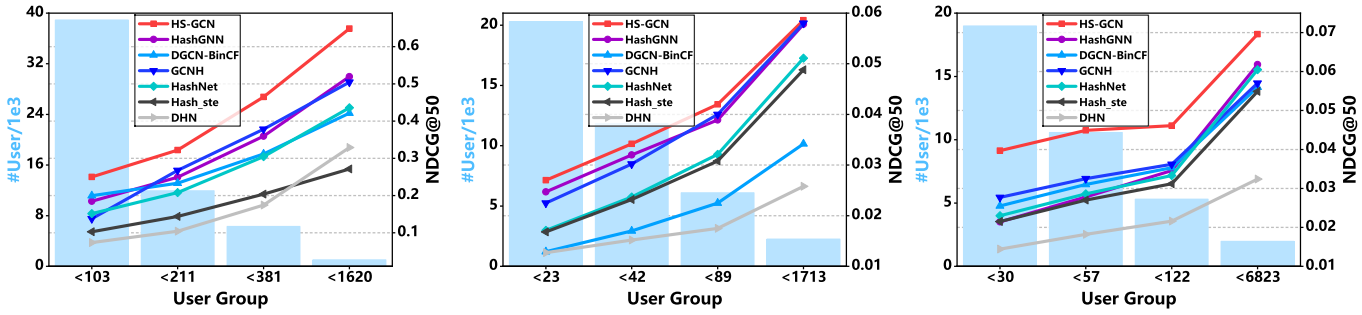
为此，我们首先将用户划分为不同的稀疏度分组。具体而言，根据训练集中每位用户的交互次数，我们将测试集划分为四组，每组的交互总数相同。以 Yelp 数据集为例，每位用户的交互次数分别少于 23、42、89 和 1,713。图3 展示了 MovieLens、Yelp 和亚马逊中不同用户组的 NDCG@50 实验结果。HR@50 也呈现出类似的性能趋势 *w.r.t.* ，因此为节省篇幅，我们省略了这部分内容。关于实验结果，我们发现：

• HS-GCN 在所有用户组上的表现始终优于其他基于哈希的基线方法。这表明，利用高阶汉明相似度可以提升活跃用户和不活跃用户的哈希码学习效果。

•通过分析图3，我们发现，HS-GCN 在前两个用户组中取得的提升（例如，在 Yelp 上分别对 < 23 和 < 42 提升例如，9。31% 和6。88% ，优于最佳基线方法）大于其他用户组（例如，在 <1，713 Yelp 用户组中提升例如，0。52%）。这证实，捕捉高阶汉明相似度对不活跃用户尤其有益，因为除了一阶相似度之外，他们的哈希码学习还利用了更充分的相似信息。因此，利用高阶相似度有望解决基于哈希的推荐系统中的稀疏性问题。

6.2 消融实验

本节研究了所提出模型中关键组件的作用，包括传播层、排序损失和 Dropout。由于哈希码传播层在模型中发挥着重要作用，我们首先考察了层数对模型性能的影响。随后



(a) MovieLens (b) Yelp (c) 亚马逊

图3. 三个不同数据集上用户组稀疏度分布的性能比较。其中，背景直方图表示每个组涉及的用户数量，折线展示了相对于*w.r.t.*NDCG@50的性能。

表4 哈希码传播层数的影响（L）。

方法	MovieLens		Yelp		亚马逊	
	HR@50	N@50	HR@50	N@50	HR@50	N@50
HS-GCN-1	0.1822	0.2774	0.0452	0.0316	0.0407	0.0340
HS-GCN-2	0.2052	0.3081	0.0497	0.0336	0.0523	0.0436
HS-GCN-3	0.1840	0.2798	0.0482	0.0329	0.0500	0.0421

我们分析了模型优化中的排序增强损失。此外，我们还联合分析了节点丢弃率和比特丢弃率的影响。

6.2.1 层数的影响

为探究哈希码传播层的最佳数量，我们改变了模型深度。具体而言，我们在 $\{1, 2, 3\}$ 范围内搜索层数。表 4 总结了实验结果，其中 HS-GCN-2 表示具有两个传播层的模型，其他模型也采用类似的命名方式。结合分析表 2 和表 4，我们得出以下观察结果:

- 显然，在所有数据集上，HS-GCN-2 和 HS-GCN-3 的表现始终优于 HS-GCN-1。原因在于，HS-GCN-1 仅考虑一阶汉明相似度，而 HS-GCN-2 和 HS-GCN-3 均利用了高阶汉明相似度。
- HS-GCN-2 的表现最佳。进一步叠加传播层时，我们发现 HS-GCN-3 在所有数据集上都出现了过拟合。我们模型的最佳层数与此前的 HashGNN [18] 一致。因此，对于哈希学习而言，二阶相似度已足够。

•在改变传播层数量时，HS-GCN 在三个数据集上始终优于学习哈希基线方法。这证实，直接捕获高阶汉明相似度有助于提升哈希码的质量。

6.2.2 排序损失的影响

为捕捉哈希学习中的相对物品排序，我们在模型优化过程中引入了排序增强损失。在实验中，我们发现排序损失既能作用于初始层，也能作用于预测层，其中排序损失分别根据初始状态 $\{\mathbf{h}_u, \mathbf{h}_i, \mathbf{h}_j\}$ 和最终哈希码 $\{\mathbf{h}_u^*, \mathbf{h}_i^*, \mathbf{h}_j^*\}$计算。为研究这两类

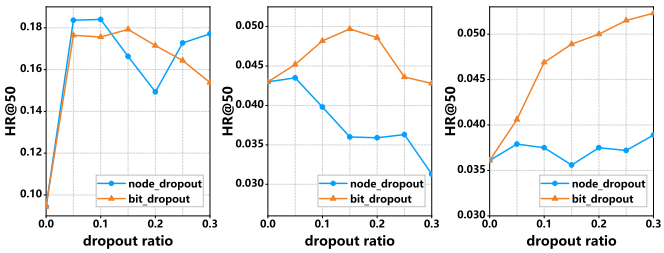
表5 不同损失函数下模型的性能比较。

方法	MovieLens		Yelp		亚马逊	
	HR@50	N@50	HR@50	N@50	HR@50	N@50
HS-GCN	0.2052	0.3081	0.0497	0.0336	0.0523	0.0436
无初始排序	0.1773	0.2747	0.0468	0.0313	0.0509	0.0428
无最终排序	0.1769	0.2566	0.0418	0.0281	0.0460	0.0385
不使用 CR-熵	0.1564	0.2426	0.0156	0.0100	0.0289	0.0239

排序损失的影响，我们进行了消融研究，以验证它们各自对模型整体性能的贡献。

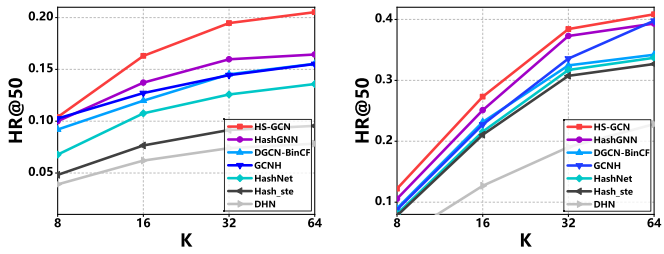
表5记录了在三个数据集上的消融研究结果。其中，HS-GCN 是我们的完整模型，包含两种排序损失，分别基于初始状态 $\{\mathbf{h}_u, \mathbf{h}_i, \mathbf{h}_j\}$ 和最终哈希码$\{\mathbf{h}_u^*, \mathbf{h}_i^*, \mathbf{h}_j^*\}$的三元组计算而得。W/o Initial Rank 表示通过移除初始哈希码三元组上的排序损失得到的 HS-GCN 消融变体，而 W/o Final Rank 则表示通过移除最终哈希码上的排序损失得到的变体。从表中可以看出，忽略用于预测的排序损失会导致 HS-GCN 的表现变差。这表明排序损失能够提升哈希效果。此外，我们发现，当排序损失作用于初始层时，对模型性能的提升更为显著，因为它将顺序信息注入了初始状态。这也表明，初始化能够提升最终哈希码的质量，这与先前研究的发现一致 [6], [11]。

沿用 HashGNN [18],中的操作，我们引入交叉熵损失，对用户与其此前消费过的物品之间的交互进行监督。我们认为，最大化交叉熵损失能够提供保留二部图结构的信号，而二部图结构是图神经网络的基础。由于这并非由我们首创，因此我们没有在论文中声称这一设计是我们的贡献。为评估各项设计的有效性，我们在多个数据集上进行了消融研究。具体而言，我们从最终损失函数中移除了二元交叉熵损失，并在表5中将其记为 W/o CR-熵；实验观察到，该变体的表现不如完整模型。可能的原因是，二元交叉熵损失能够监督哈希码重建用户与物品之间的交互，从而优化它们的哈希编码。



(a) MovieLens (b) Yelp (c) 亚马逊

图 4. 节点丢弃率和比特丢弃率的影响。



(a) MovieLens (b) Gowalla

图 5. 哈希码长度的影响。

6.2.3 Dropout 的影响

尽管图卷积网络具有很强的表示能力，但通常会面临过拟合问题。Dropout 是防止模型过拟合的有效方法 [68]。遵循以往基于图卷积网络的研究，并结合我们的模型架构，我们尝试采用两种丢弃技术来提升 HS-GCN 的性能：节点丢弃和比特丢弃。节点丢弃会在哈希码聚合过程中随机丢弃特定的邻居节点。对于第 l 个传播层，我们随机丢弃邻接矩阵中的 $p_1 \times (N+M)$ 个节点，其中 p_1 为丢弃率。我们还采用了比特丢弃，在预测前以 p_2 的概率随机丢弃用户和物品最终哈希码中的少数比特。

请注意，Dropout 仅用于训练过程，在测试过程中应禁用。图4 展示了不同数据集上，在 HR@50 评估指标下节点丢弃率 p_1 和比特丢弃率 p_2 的影响。对于这两种丢弃策略，比特丢弃的表现要好得多，尤其是在稀疏数据集上。节点丢弃甚至会对 Yelp 数据集产生负面影响。一个可能的原因是，丢弃邻居节点会损害 HS-GCN 在传播过程中执行的哈希码聚合操作，尤其是在邻居节点稀疏时。比特丢弃更为有效，这意味着它可以成为一种有效策略，解决图卷积网络上哈希学习的过拟合问题。

6.2.4 哈希码长度的影响

为了探究哈希码长度对 HS-GCN 和其他基线方法的影响，我们将它们的哈希码长度设为 $\{8, 16, 32, 64\}$ 范围内的不同值，并重复了实验。图 5 展示了这些模型在 MovieLens 和 Gowalla 数据集上使用不同长度 K 时的性能。可以观察到，哈希码长度对哈希模型有积极影响，但随着长度增加，这种影响逐渐减弱

随着长度 K 增加，其他数据集上也观察到了类似趋势。

7 结论与未来工作

在这项工作中，我们将高阶汉明相似度明确融入哈希码学习。具体而言，我们设计了一种新颖的框架 HS-GCN，通过在用户-物品图结构上传播相似度信息来生成二进制哈希码。HS-GCN 是首个提出的汉明空间图卷积网络，其传播层基于汉明空间运算构建，从而直接捕捉高阶汉明相似度。在三个真实世界数据集上进行的大量实验表明，捕捉高阶相似度用于哈希学习是合理且有效的。

未来，我们计划通过融入多形式知识来增强 HS-GCN，例如表格中的属性 [69],以及矩阵中的名人 [70]，。此外，我们还对构建微视频推荐系统感兴趣 [71], [72]。鉴于微视频不断大量涌现，因此需要更高效、准确的推荐。另一个新兴的研究方向是探索基于可解释哈希的推荐 [73]。

致谢

本研究得到国家自然科学基金（中国）项目（编号：U1936203）、山东省重点研发计划（重大科技创新项目）（编号：2020CXGC010111）以及北京人工智能研究院（BAAI）的支持。

参考文献

[1] S. Rendle、C. Freudenthaler、Z. Gantner 和 L. Schmidt-Thieme，《基于隐式反馈的贝叶斯个性化排序》，发表于 *UAI*，2012年，第 452–461 页。[2] X. Wang、X. He、M. Wang、F. Feng 和 T.-S. Chua，《神经图协同过滤》，发表于 *SIGIR*，2019年，第 165–174 页。[3] Y. Wei、X. Wang、L. Nie、X. He、R. Hong 和 T.-S. Chua，《用于微视频个性化推荐的多模态图卷积网络》，发表于 *MM*，2019年，第 1437–1445 页。[4] Z. Li、J. Tang 和 T. Mei，《用于社交图像理解的深度协同嵌入》，*TPAMI*，第 41 卷，第 9 期，第 2070–2083 页，2018年。[5] H. Liu、X. He、F. Feng、L. Nie、R. Liu 和 H. Zhang，《面向快速基于特征的推荐的离散因子分解机》，发表于 *IJCAI*，2018年，第 3449–3455 页。[6] H. Zhang、F. Shen、W. Liu、X. He、H. Luan 和 T.-S. Chua，《离散协同过滤》，发表于 *SIGIR*，2016年，第 325–334 页。[7] Y. Zhang、H. Wang、D. Lian、I. W. Tsang、H. Yin 和 G. Yang，《基于自步学习的离散排序矩阵分解》，发表于 *KDD*，2018年，第 2758–2767 页。[8] J. Wang、S. Kumar 和 S.-F. Chang，《面向大规模搜索的半监督哈希》，*TPAMI*，第 34 卷，第 12 期，第 2393–2406 页，2012年。[9] Y. Weiss、A. Torralba 和 R. Fergus，《谱哈希》，发表于 *NIPS*，2009年，第 1753–1760 页。[10] W.-C. Kang 和 J. McAuley，《利用二进制码进行大规模 Top-N 推荐的候选生成》，发表于 *CIKM*，2019年，第 1523–1532 页。[11] D. Lian、R. Liu、Y. Ge、K. Zheng、X. Xie 和 L. Cao，《离散内容感知矩阵分解》，发表于 *KDD*，2017年，第 325–334 页。[12] T. Chen、H. Yin、Y. Zheng、Z. Huang、Y. Wang 和 M. Wang，《用于定制设备端推荐系统的弹性嵌入学习》，发表于*KDD*，2021年，第 138–147 页。[13] Q. Wang、H. Yin、T. Chen、Z. Huang、H. Wang、Y. Zhao 和 N. Q. Viet Hung，《资源受限移动设备上的下一兴趣点推荐》，发表于*WWW*，2020年，第 906–916 页。

[14] Y. Zhang、H. Yin、Z. Huang、X. Du、G. Yang 和 D. Lian，《面向快速内容感知推荐的离散深度学习》，发表于 WSDM，2018年，第717–726 页。[15] H. Zhu、M. Long、J. Wang 和 Y. Cao，《用于高效相似性检索的深度哈希网络》，发表于 AAAI，2016年，第 2415–2421 页。[16] H. Liu、R. Wang、S. Shan 和 X. Chen，《用于快速图像检索的深度监督哈希》，发表于 CVPR，2016年，第 2064–2072 页。[17] Z. Cao、M. Long、J. Wang 和 P. S. Yu，《HashNet：基于延续法的深度哈希学习》，发表于 ICCV，2017年，第 5608–5617 页。[18] Q. Tan、N. Liu、X. Zhao、H. Yang、J. Zhou 和 X. Hu，《利用图神经网络为推荐系统学习哈希》，发表于 WWW，2020年，第 1988–1998 页。[19] M. Gori、A. Pucci、V. Roma 和 I. Siena，《ItemRank：一种基于随机游走的推荐引擎评分算法》，发表于 IJCAI，2007年，第 2766–2771 页。[20] X. He、M. Gao、M.-Y. Kan 和 D. Wang，《BiRank：迈向二部图排序》，TKDE，第 29 卷，第 1 期，第 57–71 页，2016年。[21] Y. Wei、X. Wang、X. He、L. Nie、Y. Rui 和 T.-S. Chua，《用于多媒体推荐的分层用户意图图网络》，TMM，2021年。[22] Y. Wei、X. Wang、L. Nie、X. He 和 T.-S. Chua，《结合隐式反馈的多媒体推荐图精炼卷积网络》，发表于 MM，2020年，第 3541–3549 页。[23] F. Liu、Z. Cheng、L. Zhu、Z. Gao 和 L. Nie，《用于推荐的兴趣感知消息传递图卷积网络》，发表于 WWW，2021年，第 1296–1305 页。[24] Y. Wu、H. Liu 和 Y. Yang，《用于二部图边预测的图卷积矩阵补全》，发表于 IC3K，2018年，第 49–58 页。[25] R. Ying、R. He、K. Chen、P. Eksombatchai、W. L. Hamilton 和 J. Leskovec，《面向网络规模推荐系统的图卷积神经网络》，发表于 KDD，2018年，第 974–983 页。[26] X. He、K. Deng、X. Wang、Y. Li、Y. Zhang 和 M. Wang，《LightGCN：简化并增强用于推荐的图卷积网络》，发表于 SIGIR，2020年，第 639–648 页。[27] C. Shi、B. Hu、W. X. Zhao 和 S. Y. Philip，《用于推荐的异构信息网络嵌入》，TKDE，第 31 卷，第 2 期，第 357–370 页，2018年。[28] B. Hu、C. Shi、W. X. Zhao 和 P. S. Yu，《利用基于元路径的上下文，通过神经协同注意力模型实现 Top-N 推荐》，发表于 KDD，2018年，第 1531–1540 页。[29] S. Fan、J. Zhu、X. Han、C. Shi、L. Hu、B. Ma 和 Y. Li，《用于意图推荐的元路径引导异构图神经网络》，发表于 KDD，2019年，第 2478–2486 页。[30] Y. Yang、Z. Guan、J. Li、W. Zhao、J. Cui 和 Q. Wang，《可解释且高效的异构图卷积网络》，TKDE，2021年。[31] Z. Li、J. Tang、L. Zhang 和 J. Yang，《用于社交图像检索的弱监督语义引导哈希》。IJCV，第128 卷，第 8/9 期，第 2265–2278 页，2020年。[32] H. Jegou、M. Douze 和 C. Schmid，《用于最近邻搜索的乘积量化》，TPAMI，第33 卷，第 1 期，第 117–128 页，2010年。[33] W. Liu、J. Wang、S. Kumar 和 S.-F. Chang，《基于图的哈希》，发表于 ICML，2011年，第 1–8 页。[34] B. Kulis 和 T. Darrell，《利用二进制重构嵌入进行哈希学习》，发表于 NIPS，2009年，第 1042–1050 页。[35] W. Liu、J. Wang、R. Ji、Y.-G. Jiang 和 S.-F. Chang，《基于核的监督哈希》，发表于 CVPR，2012年，第 2074–2081 页。[36] M. Norouzi 和 D. J. Fleet，《面向紧凑二进制码的最小损失哈希》，发表于 ICML，2011年，第 353–360 页。[37] F. Shen、C. Shen、W. Liu 和 H. Tao Shen，《监督离散哈希》，发表于 CVPR，2015年，第 37–45 页。[38] Y. Gong、S. Lazebnik、A. Gordo 和 F. Perronnin，《迭代量化：一种用于学习大规模图像检索二进制码的普罗克鲁斯忒斯方法》，TPAMI，第 35 卷，第 2929、2012年 12 2916 期，第 – 页。[39] X. He 和 T.-S. Chua，《用于稀疏预测分析的神经因子分解机》，发表于 SIGIR，2017年，第 355–364 页。[40] R. Xia、Y. Pan、H. Lai、C. Liu 和 S. Yan，《通过图像表征学习进行图像检索的监督哈希》，发表于 AAAI，2014年，第 2156–2162 页。[41] H. Lai、Y. Pan、Y. Liu 和 S. Yan，《利用深度神经网络联合进行特征学习与哈希编码》，发表于 CVPR，2015年，第 3270–3278 页。[42] H. Wang、D. Lian 和 Y. Ge，《通过蒸馏图卷积网络实现二值化协同过滤》，arXiv 预印本 arXiv:1906.01829，2019年。[43] X. Zhou、F. Shen、L. Liu、W. Liu、L. Nie、Y. Yang 和 H. T. Shen，“图卷积网络哈希”，TC，第50卷，第4期，第1460–1472页，2018年。[44] Y. Bengio、N. L´eonard 和 A. Courville，“通过随机神经元进行条件计算时估计或传播梯度”，arXiv预印本 arXiv:1308.3432，2013年。[45] Y. Zhou、X. Bai、W. Liu 和 L. J. Latecki，“通过扩散融合实现鲁棒视觉跟踪”，载于 NIPS，2012年，第2978–2986页。[46] F. Yang、R. Hinami、Y. Matsui、S. Ly 和 S. Satoh，“通过将扩散解耦为在线处理和离线处理实现高效图像检索”，载于 AAAI，2019年，第9087–9094页。[47] B. Wang、A. M. Mezlini、F. Demir、M. Fiume、Z. Tu、M. Brudno、B. Haibe-Kains 和 A. Goldenberg，“用于基因组尺度数据类型聚合的相似性网络融合”，《自然方法》，第11卷，第3期，第333–337页，2014年。[48] H. Liu、R. Ji、Y. Wu、F. Huang 和 B. Zhang，“通过融合相似性哈希进行跨模态二进制码学习”，载于 CVPR，2017年，第7380–7388页。[49] S. Bai、Z. Zhou、J. Wang、X. Bai、L. Jan Latecki 和 Q. Tian，“用于检索的集成扩散”，载于 ICCV，2017年，第774–783页。[50] Y. Wang、Y. Wang、J. Yang 和 Z. Lin，“剖析线性图卷积网络中的扩散过程”，arXiv预印本 arXiv:2102.10739，2021年。[51] B. Jiang、D. Lin、J. Tang 和 B. Luo，“利用图扩散嵌入网络进行数据表示与学习”，载于 CVPR，2019年，第10 414–10 423页。[52] Y. Feng、H. You、Z. Zhang、R. Ji 和 Y. Gao，“超图神经网络”，载于 AAAI，2019年，第3558–3565页。[53] Y. Wei、X. Wang、Q. Li、L. Nie、Y. Li、X. Li 和 T.-S. Chua，“面向冷启动推荐的对比学习”，载于 MM，2021年，第5382–5390页。[54] X. Yu、T. Gan、Y. Wei、Z. Cheng 和 L. Nie，“面向二手交易平台的个性化物品推荐”，载于 MM，2020年，第3478–3486页。[55] Z. Zhang，“用于深度神经网络的改进 Adam 优化器”，载于 IWQoS，2018年，第1–2页。[56] X. He、H. Zhang、M.-Y. Kan 和 T.-S. Chua，“利用隐式反馈实现在线推荐的快速矩阵分解”，载于 SIGIR，2016年，第549–558页。[57] X. Wang、X. He、L. Nie 和 T.-S. Chua，“物品丝绸之路：从信息领域向社交用户推荐物品”，载于 SIGIR，2017年，第185–194页。[58] D. Liang、L. Charlin、J. McInerney 和 D. M. Blei，“推荐中的用户曝光建模”，载于 WWW，2016年，第951–961页。[59] X. Geng、H. Zhang、J. Bian 和 T.-S. Chua，“学习图像和用户特征以在社交网络中进行推荐”，载于 ICCV，2015年，第4274–4282页。[60] X. He、L. Liao、H. Zhang、L. Nie、X. Hu 和 T.-S. Chua，“神经协同过滤”，载于 WWW，2017年，第173–182页。[61] X. Glorot 和 Y. Bengio，“理解深度前馈神经网络训练的难点”，载于 AISTATS，2010年，第249–256页。[62] A. Krizhevsky、I. Sutskever 和 G. E. Hinton，“使用深度卷积神经网络进行 ImageNet 分类”，《ACM通讯》，第60卷，第6期，第84–90页，2017年。[63] A. Mnih 和 R. R. Salakhutdinov，“概率矩阵分解”，载于 NIPS，2008年，第1257–1264页。[64] M. Gao、L. Chen、X. He 和 A. Zhou，“BiNE：二部网络嵌入”，载于 SIGIR，2018年，第715–724页。[65] W. Hamilton、Z. Ying 和 J. Leskovec，“大型图上的归纳式表示学习”，载于 NIPS，2017年，第1024–1034页。[66] M. Norouzi、A. Punjani 和 D. J. Fleet，“利用多索引哈希在汉明空间中快速搜索”，载于 CVPR，2012年，第3108–3115页。[67] J. Liu、Z. Li、J. Tang、Y. Jiang 和 H. Lu，“面向社交网站照片的个性化地理位置特定标签推荐”，TMM，第16卷，第3期，第588–600页，2014年。[68] N. Srivastava、G. Hinton、A. Krizhevsky、I. Sutskever 和 R. Salakhutdinov，“Dropout：防止神经网络过拟合的简单方法”，JMLR，第15卷，第1929–1958页，2014年。[69] F. Liu、Z. Cheng、C. Sun、Y. Wang、L. Nie 和 M. S. Kankanhalli，“通过多模态注意力度量学习建模用户的多样化偏好”，载于 MM，2019年，第1526–1534页。[70] X. Wang、X. He、Y. Cao、M. Liu 和 T.-S. Chua，“KGAT：用于推荐的知识图谱注意力网络”，载于 KDD，2019年，第950–958页。

[71] X. Song、F. Feng、X. Han、X. Yang、W. Liu 和 L. Nie，“通过注意力知识蒸馏进行神经兼容性建模”，载于 *SIGIR*，2018年，第5–14页。[72] C. Sun、X. Song、F. Feng、W. X. Zhao、H. Zhang 和 L. Nie，“监督式分层跨模态哈希”，载于 *SIGIR*，2019年，第725–734页。[73] Y. Wei、X. Wang、W. Guan、L. Nie、Z. Lin 和 B. Chen，“面向微视频理解的神经多模态协同学习”，*TIP*，第29卷，第1–14页，2019年。[74] Y. Zhang、Y. Zhang、M. Zhang 和 C. Shah，“EARS 2019：第二届可解释推荐与搜索国际研讨会”，载于 *SIGIR*，2019年，第1438–1440页。