LightCCN: Light Cell Complex Networks for Collaborative Filtering

Alexandru Nistor

Delft University of Technology

TU Delft

LightCCN: Light Cell Complex Networks for Collaborative Filtering

by

Alexandru Nistor

born in Galați, România

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 28, 2026 at 16:30.

Student number: 5447550
Project duration: October 2025 – August 2026
Thesis committee:

Dr. E. Isufi,	Faculty EEMCS, TU Delft, chair and thesis advisor
Dr. M. Mansoury,	Faculty EEMCS, TU Delft, daily supervisor
Dr. N. Yorke-Smith,	Faculty EEMCS, TU Delft, committee member

Cover: Generated by Gemini

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Summary

Graph-based collaborative filtering models achieve strong performance by propagating user and item embeddings over the bipartite interaction graph, but they represent only pairwise relations. Any pattern among three or more items has to be recovered through multi-hop propagation rather than modelled directly. Topological deep learning extends graphs with group-level cells, arranged in a hierarchy that connects a group to the pairs and items it is composed of, yet the framework is almost unexplored in recommendation. This thesis introduces LightCCN, a model that gives frequently co-consumed item triples their own learned representation, linked to the pairs and items they contain. Formally, it lifts the user–item interaction data to a cell complex of order 2, propagates embeddings at every order of the complex, and injects the pooled higher-order signal into user representations through a user-side readout. The design is challenging because the structure is not given in advance. Which groups deserve representation and how much structure a dataset supports are modelling choices. On the top-ranked metrics, LightCCN is significantly better than its graph-based counterpart on four of the six datasets and statistically tied on the rest, at almost no additional capacity. Controlled ablations attribute the gains to the readout rather than to higher-order propagation. Depth experiments show maintained or improved accuracy against the same-depth counterpart and no over-smoothing. The amount of admitted structure matters and differs per dataset, so it is exposed as a single control selected on validation data. The selection also removes the structure on the dataset where it hurts. Overall, the results show that higher-order structures improve collaborative filtering when given an explicit path to the user representations, with gains that concentrate at the top of the ranking, and that topological deep learning is an effective and interpretable framework for capturing them.

Contents

Summary i

1 Introduction 1
- 1.1 Research Questions . . . 2
- 1.2 Contributions . . . 2
- 1.3 Thesis Outline . . . 3

2 Background 4
- 2.1 Graph-Based Recommender Systems . . . 4
  - 2.1.1 The Recommendation Problem . . . 4
  - 2.1.2 The Bipartite Interaction Graph . . . 5
- 2.2 Graph Convolutional Networks . . . 5
  - 2.2.1 The General GNN Framework . . . 5
  - 2.2.2 Adjacency Matrix and Symmetric Normalisation . . . 5
  - 2.2.3 Graph Convolutional Networks . . . 6
  - 2.2.4 NGCF: Applying GCNs to Collaborative Filtering . . . 6
  - 2.2.5 LightGCN: A Simplified GCN . . . 6
- 2.3 Cell Complexes . . . 7
  - 2.3.1 From Graphs to Cell Complexes . . . 7
  - 2.3.2 Incidence Matrices . . . 8
  - 2.3.3 Adjacency from Incidence . . . 9
- 2.4 Topological Deep Learning . . . 9
  - 2.4.1 Topological Signals . . . 9
  - 2.4.2 Message Passing on Cell Complexes . . . 10
- 2.5 Conclusion . . . 10

3 Related Work 11
- 3.1 Graph-Based Methods . . . 11
- 3.2 Linear Autoencoders . . . 12
- 3.3 Hypergraph and Motif Methods . . . 13
- 3.4 Topological Deep Learning Methods . . . 14
- 3.5 Discussion . . . 15

4 Methodology 16
- 4.1 Problem Formulation . . . 16
- 4.2 Overview and Design Principles . . . 16
- 4.3 Cell-Complex Construction . . . 17
- 4.4 Propagation Operators . . . 18
- 4.5 Stateful Higher-Order Propagation . . . 19
- 4.6 Higher-Order User Readout . . . 20
- 4.7 Training Loss . . . 21
- 4.8 Computational Complexity . . . 21
- 4.9 Discussion . . . 21

5 Experiments 23
- 5.1 Experimental Setup . . . 23
  - 5.1.1 Data and Protocol . . . 23
  - 5.1.2 Models and Configuration . . . 24
- 5.2 Results . . . 25
  - 5.2.1 RQ1: Does Modelling Higher-Order Relations Improve Recommendation Accuracy? 25

5.2.2 RQ2: Is the Contribution of Higher-Order Structure Attributable to Propagation or to the Readout? . . . 28
5.2.3 RQ3: How Does Propagation Depth Affect the Accuracy of a Model with Higher-Order Structure? . . . 29
5.2.4 RQ4: How Does the Amount of Higher-Order Structure Affect Recommendation Accuracy? . . . 31

6 Discussion **33**
6.1 Synthesis of Findings . . . 33
6.2 Relation to Prior Work . . . 33
6.3 Limitations . . . 34
6.4 Implications and Stakeholders . . . 34

7 Conclusion **36**
7.1 Answers to the Research Questions . . . 36
7.2 Future Work . . . 37

References **38**

A Formal Definitions for Cell Complexes **41**
A.1 Regular Cell Complexes . . . 41
A.2 Boundary Relations and Neighbourhoods . . . 41
A.3 The General Cell-Complex Message-Passing Scheme . . . 42

B Additional Results **43**
B.1 RQ1 at Other Cutoffs . . . 43
B.2 RQ2 Supporting Tables and Figures . . . 46
B.3 RQ3 Supporting Tables and Figures . . . 48
B.4 RQ4 Additional Metrics . . . 49
B.5 LightGCN Results Reported in Its Paper . . . 50

C Ablation: Readout Target **51**

D Ablation: Propagation Backbone under the Readout **52**

E Ablation: User–Item Edges as 1-Cells **53**

F Ablation: Edge Lower Adjacency **55**

1

Introduction

Collaborative filtering is the task of predicting user preferences from historical interaction data. Given a set of users and items and a record of users interacting with items, it recommends to each user new items they are likely to enjoy. We explore the implicit-feedback setting [1], where the model learns only from which interactions occurred, such as clicks, purchases, or check-ins, without using content features or explicit ratings, and the goal is to rank the catalogue for each user with the relevant items near the top.

User–item interactions form a bipartite graph, with user and item nodes connected by an edge whenever an interaction is observed. Classical collaborative filtering uses this structure only indirectly. Neighbourhood methods read similarities off the interaction matrix [2, 3], and matrix factorisation compresses it into latent user and item factors [4]. Graph Neural Networks (GNNs) instead made the graph part of the architecture, propagating embeddings along edges such that multi-hop connectivity, which carries the collaborative signal, is encoded directly in the learned representations [5]. NGCF [6] established this design for recommendation, adapting the graph convolutional network of Kipf and Welling [7] to the bipartite interaction graph. LightGCN [8] subsequently showed that when nodes do not carry content features, removing the feature transformations and non-linear activations from propagation not only simplifies the model but actually improves it.

Graph-based collaborative filtering, however, has a fundamental limitation. The interaction graph itself can express only pairwise relations. An edge joins exactly two nodes, and a group of three or more items never appears in the model as an object of its own. The information is not entirely lost, since a user who consumed three items is connected to all three of them in the bipartite graph, but it enters the model only implicitly, through separate pairwise edges, and never explicitly, as a primitive the model can represent, weight, and learn from. The distinction matters because pairwise co-occurrence does not determine joint consumption. Three items that many users consume jointly and three items whose pairwise overlaps come from unrelated groups of users can produce identical item-to-item co-occurrence counts, yet only in the first case are the three items actually consumed together. Steck and Liang [9] showed that this costs accuracy. Adding an explicit triplet term to a strong linear model yields significant gains, and the learned higher-order weights are predominantly negative, correcting pairwise predictions that overcount the shared reasons items co-occur.

Hypergraph-based collaborative filtering extends deep graph models with group-level structure. DHCF [10] and HCCF [11] connect many nodes through a single hyperedge and report consistent gains over pairwise baselines. Subsequent work refines how the hyperedge structure is obtained, how information propagates over it, and how the auxiliary training objective is constructed [12–14]. The abstraction, however, records that a group exists but not how the group relates to the pairwise interactions it contains. A hyperedge is a flat set of members, with no connection to the edges among its vertices. What is required is a domain in which groups and pairwise relations are represented as distinct objects connected by an explicit relation.

Topological Deep Learning (TDL) provides such domains [15]. Cell complexes extend graphs with

higher-order cells, faces attached to groups of edges, and the boundary relations between orders give the hierarchy an explicit algebraic form, such that message passing can exchange information between a face, the edges on its boundary, and the nodes on theirs [16]. Evidence from other domains indicates that this exchange between adjacent orders is where much of the value lies [17]. In collaborative filtering, however, the framework remains almost unexplored. To the best of our knowledge, only two instantiations exist to date, and neither uses the framework in full. TSP [18] builds a simplicial complex but smooths higher-order signal on top of a frozen pretrained model, while Sheaf4Rec [19] learns its topological structure end-to-end but never leaves order 1, enriching edges instead of adding higher-order cells. Whether higher-order cells whose representations are learned end-to-end inside the model improve recommendation is therefore an open question. Collaborative filtering is a natural setting for this question, since interaction data intrinsically contains higher-order relations, in the form of groups of items consumed jointly by multiple users.

This thesis addresses that question by proposing LightCCN. The model lifts the interaction data itself to a cell complex of order 2. Item triples that enough users have co-consumed become faces. Embeddings are propagated at every order of the complex and trained end-to-end without feature transformations or non-linear activations. The face information enters the model in two ways. It shapes the embeddings during propagation, and a readout injects the pooled cell signal into the user representations.

1.1. Research Questions

The aim of this work is to answer the following research question.

(RQ) *Can topological deep learning improve collaborative filtering by modelling higher-order relations?*

Higher-order relations are present in interaction data, and pairwise models cannot express them. Topological deep learning offers a representation that captures the groups while keeping the hierarchy that connects them to their pairs and items. Yet the framework is almost unexplored in collaborative filtering, and no prior model uses it in full. The question is challenging because the structure is not given. Which item groups deserve representation, how much structure a dataset supports, and whether the added machinery harms training at depth are unknown in advance.

We answer the question through four sub-questions. RQ1 establishes whether the effect exists, comparing a model with higher-order relations against the same model without them. RQ2 attributes the effect to a component of the design, so that the gain can be traced to a mechanism rather than to added capacity. RQ3 tests whether the effect survives the main architectural choice, the propagation depth. RQ4 asks how much structure should be admitted and whether that amount can be chosen reliably per dataset. Together the four cover the existence of the effect, its mechanism, its robustness, and its control. Each sub-question is answered in Chapter 5 and revisited in Chapter 7.

(RQ1) Accuracy. *Does modelling higher-order relations improve recommendation accuracy?*

(RQ2) Mechanism. *Is the contribution of higher-order structure attributable to propagation or to the readout?*

(RQ3) Depth. *How does propagation depth affect the accuracy of a model with higher-order structure?*

(RQ4) Structure amount. *How does the amount of higher-order structure affect recommendation accuracy?*

1.2. Contributions

The main contributions of this work are:

1. **LightCCN**, a collaborative filtering model that lifts the interaction data to a cell complex of order 2 by turning frequently co-consumed item triples into faces, propagates embeddings at every order of the complex, and injects the pooled cell signal into the user representations through a readout. To our knowledge it is the first recommender to use the cell-complex framework in full, and it gives group-level structure to a widely used class of graph-based models at almost no additional capacity or cost (Chapter 4).

2. **Empirical evidence that modelling higher-order relations improves recommendation accuracy.** Adding the structure to a graph-based model yields significant gains on four of six public datasets and no significant loss in the accuracy comparison. The gains concentrate at the top of the ranking, where recommendations matter most (Chapter 5).
3. **An analysis of when higher-order structure helps.** The contribution of the structure is attributable to the user-side readout rather than to propagation, the helpful amount of structure is dataset-specific and controlled by the face-support threshold, and the same selection removes the structure on data where it hurts (Chapters 5 and 6).

1.3. Thesis Outline

The remainder of this thesis is organised as follows. Chapter 2 establishes the technical foundations, from the collaborative filtering problem and the LightGCN model that LightCCN extends to regular cell complexes and message passing on them. Chapter 3 reviews four lines of related work and positions LightCCN among them. Chapter 4 presents the model, with the construction of the cell complex from co-consumption, the higher-order propagation, and the user readout. Chapter 5 specifies the experimental protocol and reports the results, organised by research question. Chapter 6 interprets the findings and their limitations, and Chapter 7 answers the research questions and outlines future work. The appendices collect the formal definitions behind the model, the supporting tables and figures, and the four design ablations.

2

Background

This chapter establishes the technical foundations on which LightCCN is built. Section 2.1 formalises the recommendation problem under the collaborative filtering paradigm and introduces its bipartite-graph representation. Section 2.2 presents graph convolutional networks, from the general message-passing framework to the LightGCN propagation rule on which LightCCN builds. Section 2.3 introduces regular cell complexes, the higher-order topological structure that extends graphs to relations among groups of items rather than pairs. Section 2.4 describes topological deep learning, which generalises message passing to such structures. Section 2.5 synthesises these foundations and bridges to the methodology of Chapter 4.

2.1. Graph-Based Recommender Systems

Graph-based recommender systems cast the collaborative filtering problem as inference on a graph derived from user–item interactions [20, 21].

2.1.1. The Recommendation Problem

$\mathcal{U} = \{u_1, \ldots, u_M\}$ is the set of users and $\mathcal{I} = \{i_1, \ldots, i_N\}$ the set of items, with $M = |\mathcal{U}|$ and $N = |\mathcal{I}|$. The interactions a user has had with items, such as clicks, purchases, or views, are recorded in a binary matrix $X \in \{0, 1\}^{M \times N}$, in which $X_{ui} = 1$ when user u has interacted with item i and $X_{ui} = 0$ otherwise [20]. This is the implicit-feedback setting [21], in which the system observes only the occurrence of an interaction and does not receive a numerical rating. An unobserved entry $X_{ui} = 0$ is not equivalent to dislike, since the user may not have seen item i. An observed entry does not guarantee preference either, since the user may have interacted with i for a reason unrelated to genuine interest [20, 21]. The recommendation task is to predict a real-valued preference score $\hat{y}_{ui} \in \mathbb{R}$ for every unobserved pair (u, i) with $X_{ui} = 0$, and to use these scores to rank the items each user has not yet interacted with [20].

CF models considered in this thesis follow an embedding-based paradigm [6, 20]. Each user u and each item i is associated with a learnable d-dimensional embedding $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$, where d is the embedding dimension. The predicted preference score is the inner product of the two embeddings, $\hat{y}_{ui} = \langle \mathbf{e}_u, \mathbf{e}_i \rangle$.

Matrix factorisation (MF) [4] is the canonical embedding-based CF model and uses this inner-product score at prediction time. It approximates the interaction matrix as $X \approx UV^\top$, where the rows of $U \in \mathbb{R}^{M \times d}$ are the user embeddings $\mathbf{e}_u$ and the rows of $V \in \mathbb{R}^{N \times d}$ are the item embeddings $\mathbf{e}_i$. In the implicit-feedback setting, MF is trained by optimising a ranking loss that places observed user–item pairs above unobserved ones. The Bayesian Personalised Ranking (BPR) loss [1] is the choice adopted by both NGCF [6] and LightGCN [8] and is defined as

$$\mathcal{L}_{\text{BPR}} = -\sum_{(u,i,j) \in O} \ln \sigma\big(\hat{y}_{ui} - \hat{y}_{uj}\big) + \lambda \, \|\Theta\|_2^2, \tag{2.1}$$

where the training set $O = \{(u, i, j) : X_{ui} = 1, \, X_{uj} = 0\}$ pairs each observed interaction (u, i) with a

sampled unobserved pair (u, j), $\sigma(\cdot)$ is the sigmoid function, Θ are the model parameters, and λ is the regularisation coefficient [6, 8].

2.1.2. The Bipartite Interaction Graph

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ that connect pairs of nodes [5]. The interaction matrix X has a natural representation as a bipartite graph, in which the nodes are the users and items and the edges are the observed user–item interactions [6, 22]. The user–item interaction graph is

$$\mathcal{G} = (\mathcal{U} \cup \mathcal{I}, \mathcal{E}), \qquad \mathcal{E} = \{(u, i) : X_{ui} = 1\}, \tag{2.2}$$

where the node set $\mathcal{U} \cup \mathcal{I}$ is the union of the user set and the item set, with $|\mathcal{V}| = M + N$ nodes in total. Edges connect users to items only. The interactions can be represented using an adjacency matrix

$$A = \begin{pmatrix} 0 & X \\ X^\top & 0 \end{pmatrix} \in \mathbb{R}^{(M+N)\times(M+N)}, \tag{2.3}$$

where $A_{vu} = 1$ if there is an edge between nodes v and u, and $A_{vu} = 0$ otherwise [6, 8]. The two off-diagonal blocks contain the interaction matrix X and its transpose, and the two diagonal blocks are zero because there are no user–user or item–item edges.

The neighbourhood of a user u is the set of items u has interacted with, and the neighbourhood of an item i is the set of users that have interacted with it [6, 8], that is,

$$\mathcal{N}_u = \{i \in \mathcal{I} : X_{ui} = 1\}, \qquad \mathcal{N}_i = \{u \in \mathcal{U} : X_{ui} = 1\}. \tag{2.4}$$

If two users u_1 and u_2 share many past interactions, their neighbourhoods $\mathcal{N}_{u_1}$ and $\mathcal{N}_{u_2}$ have a large intersection. This intersection is the bipartite-graph counterpart of the CF principle that behaviourally similar users tend to share preferences [6]. Paths of length greater than one in $\mathcal{G}$ encode multi-hop connectivity. A path $u_1 \to i_1 \to u_2 \to i_2$ indicates that users u_1 and u_2 have both interacted with item i_1, and that u_2 has further interacted with item i_2, which makes i_2 a candidate recommendation for u_1 [6]. MF predicts $\hat{y}_{u_1,i_2}$ from two embeddings learned in isolation, without explicit access to such paths. Graph neural networks inject them into the embeddings during training. NGCF and LightGCN refer to this multi-hop connectivity as high-order connectivity and higher-order proximity, respectively [6, 8]. In this thesis, the concept is referred to as multi-hop connectivity.

2.2. Graph Convolutional Networks

2.2.1. The General GNN Framework

A graph neural network (GNN) processes a graph by iteratively updating an embedding of each node from the embeddings of its neighbours [5, 23]. Let $\mathbf{h}_v^{(\ell)} \in \mathbb{R}^d$ denote the embedding of node v at layer ℓ, where d is the embedding dimension. Let $\mathcal{N}(v) = \{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}$ denote the set of nodes adjacent to v in the graph. Each GNN layer maps the embeddings at layer ℓ to those at layer $\ell + 1$ through

$$\mathbf{h}_v^{(\ell+1)} = \text{UPDATE}\Big(\mathbf{h}_v^{(\ell)}, \text{AGGREGATE}\Big(\{\mathbf{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}\Big)\Big), \tag{2.5}$$

where AGGREGATE is a permutation-invariant pooling function (i.e., a function whose output does not depend on the order of its inputs, such as a sum or a mean) over the neighbour embeddings, instantiated as sum, mean, or max in different GNN variants, and UPDATE combines the node's embedding from layer ℓ with the aggregated neighbourhood embedding [5]. Stacking L such layers extends the receptive field of each node to its L-hop neighbourhood. Different choices of AGGREGATE and UPDATE define different GNN families.

2.2.2. Adjacency Matrix and Symmetric Normalisation

The compact matrix form of Equation 2.5 requires a single operator that performs neighbourhood aggregation across all nodes simultaneously. The adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of an undirected graph $\mathcal{G}$ has entries $A_{uv} = 1$ if $(u, v) \in \mathcal{E}$ and $A_{uv} = 0$ otherwise. For collaborative filtering, A takes the bipartite user–item form of Equation 2.3. The corresponding degree matrix $D \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is diagonal,

with $D_{vv} = |\mathcal{N}(v)|$ and $D_{uv} = 0$ for $u \neq v$. The matrix $\mathbf{H}^{(\ell)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ stacks the node embeddings of layer ℓ as rows, with the row corresponding to node v equal to $\mathbf{h}_v^{(\ell)}$.

The product $A\mathbf{H}^{(\ell)}$ aggregates each node's neighbours by summation. However, using A directly as a propagation operator suffers from two issues [5, 7]. First, nodes with high degree dominate the aggregated signal. Second, the spectral radius of A generally exceeds one, so repeated multiplication by A amplifies signal magnitudes layer by layer and produces unstable representations. Both issues are addressed by replacing A with the symmetric normalisation

$$\hat{A} = \tilde{D}^{-1/2}\,\tilde{A}\,\tilde{D}^{-1/2}, \qquad \tilde{A} = A + I, \quad \tilde{D}_{vv} = \sum_{u \in \mathcal{V}} \tilde{A}_{vu}, \tag{2.6}$$

where $\tilde{A}$ is the adjacency matrix augmented with self-loops and $\tilde{D}$ is the corresponding degree matrix. The matrix $\hat{A}$ is the propagation operator used by the GCN-based models below.

2.2.3. Graph Convolutional Networks

The graph convolutional network (GCN) [7] instantiates the message-passing template of Equation 2.5 with $\hat{A}$ as the aggregation operator and a learnable linear transformation followed by a non-linearity as the update function. The layer-wise propagation rule is

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\hat{A}\,\mathbf{H}^{(\ell)} W^{(\ell)}\right), \tag{2.7}$$

where $W^{(\ell)} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix that linearly transforms the aggregated embeddings, σ is a non-linear activation function, instantiated as ReLU in [7], and $\mathbf{H}^{(0)}$ is the matrix of input node features. In the collaborative filtering setting, $\mathbf{H}^{(0)}$ contains learnable user and item embeddings rather than fixed input features.

2.2.4. NGCF: Applying GCNs to Collaborative Filtering

NGCF [6] adapts the GCN propagation rule to the user–item bipartite graph $\mathcal{G}$ of Section 2.1.2. Each propagation layer refines a user's embedding by aggregating from the items the user has interacted with, and symmetrically for items. The layer-wise rule is

$$\mathbf{e}_u^{(k+1)} = \sigma\left(W_1^{(k)}\,\mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\left(W_1^{(k)}\,\mathbf{e}_i^{(k)} + W_2^{(k)}(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)})\right)\right), \tag{2.8}$$

where $1/\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}$ is the symmetric normalisation, $W_1^{(k)}, W_2^{(k)} \in \mathbb{R}^{d \times d}$ are layer-specific learnable weight matrices, σ is the activation function (LeakyReLU in [6]), and $\mathbf{e}_i \odot \mathbf{e}_u$ is the affinity term, an element-wise product that strengthens the message when user and item embeddings agree on more dimensions [6]. An analogous propagation rule applies to item embeddings. After L layers, NGCF concatenates the per-layer embeddings, $\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \,\|\, \cdots \,\|\, \mathbf{e}_u^{(L)}$, predicts with the inner-product score of Section 2.1.1, and trains with the BPR loss of Equation 2.1 [6].

2.2.5. LightGCN: A Simplified GCN

LightGCN [8] simplifies NGCF by removing four components from the propagation rule of Equation 2.8, namely the per-layer weight matrices $W_1^{(k)}$ and $W_2^{(k)}$, the non-linear activation σ, the affinity term $\mathbf{e}_i \odot \mathbf{e}_u$, and the self-loops added by $\tilde{A} = A + I$ in the GCN normalisation [8]. The update rule is

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\,\mathbf{e}_i^{(k)}, \qquad \mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}}\,\mathbf{e}_u^{(k)}, \tag{2.9}$$

where $\mathcal{N}_u$ and $\mathcal{N}_i$ are the user and item neighbourhoods defined in Equation 2.4, and $\mathbf{e}_v^{(0)}$ is a learnable initial embedding for each user and each item. These initial embeddings are the only trainable parameters in the model [8].

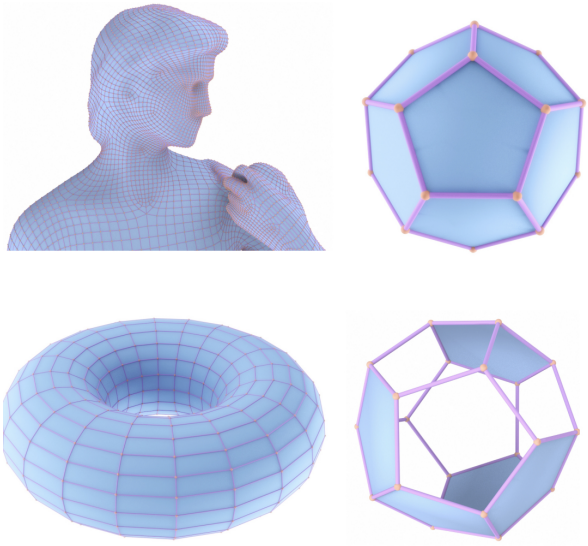

Figure 2.1: Examples of cell complexes of varying dimensions, reproduced from Figure 7 of [15].

After K propagation layers, the final embedding of each node is a weighted sum of its embeddings at every layer:

$$\mathbf{e}_v^* = \sum_{k=0}^{K} \alpha_k \, \mathbf{e}_v^{(k)}, \qquad \alpha_k \geq 0, \tag{2.10}$$

with $\alpha_k = 1/(K+1)$ in [8]. The layer-combination step subsumes the self-loops removed from the propagation operator. A formal proof in [8] shows that a K-layer GCN with self-loops can be expressed as a weighted sum of LightGCN-style propagation layers. Prediction uses the inner-product score of Section 2.1.1 on the final embeddings $\mathbf{e}_u^*$ and $\mathbf{e}_i^*$, and training uses the BPR loss of Equation 2.1 [8]. LightCCN builds on LightGCN for two reasons. First, LightGCN is the standard baseline of graph-based collaborative filtering, and later methods such as SimGCL [24] and HCCF [11] use it as their backbone. Second, its simplicity allows the effect of an extension to be isolated. Each LightGCN layer consists only of degree-normalised averaging over the interaction graph, without transformation matrices or non-linearities. Any difference in accuracy that follows from adding structure can be attributed to the added structure rather than to additional capacity.

2.3. Cell Complexes

Graphs encode pairwise relations, with each edge joining exactly two nodes. Many relations are intrinsically multi-way rather than pairwise [15, 16]. In collaborative filtering, examples include items frequently bought together, playlists of co-listened tracks, and sequences of co-watched videos, all of which describe relations among more than two entities at once. Such a relation connects users through a set of items they share, $u_1 \rightarrow \{i_1, i_2, i_3\} \rightarrow u_2$, rather than through a chain of pairwise links such as $u_1 \rightarrow i_1 \rightarrow u_2 \rightarrow i_2$. This thesis calls the former relations higher-order and the latter multi-hop. Cell complexes are a standard topological structure that extends graphs to such higher-order relations while preserving a hierarchy in which every higher-order cell is bounded by lower-order ones.

2.3.1. From Graphs to Cell Complexes

A cell complex is built from cells, and each cell has an order, also called its dimension or rank [15, 16, 25]. A cell of order k is a k-cell. A 0-cell is a point, a 1-cell is a line segment, an edge, joining two 0-cells, and a 2-cell is a face enclosed by 1-cells [25]. Figure 2.1 illustrates cell complexes of higher order.

A complex is regular when it is built by attaching each cell to cells of lower order that are already present, so that a face is attached to its edges and an edge to its two endpoints [16]. The order of a complex is the maximum order of its cells [25]. A graph is a cell complex of order 1, consisting only of nodes and edges [16, 25, 26]. This thesis works with complexes of order 2 throughout, written $\mathcal{X} = (\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2)$

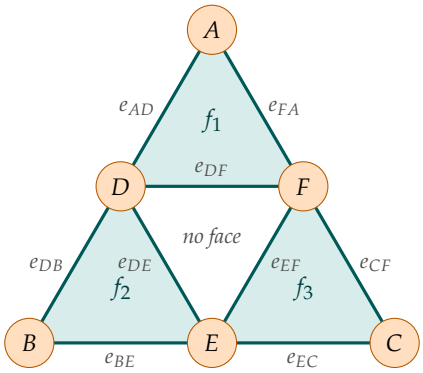

Figure 2.2: A cell complex of order 2 with six 0-cells, the items A to F, nine 1-cells, the item–item edges, and three 2-cells, the faces $f_1 = \{A, D, F\}$, $f_2 = \{D, B, E\}$, and $f_3 = \{F, E, C\}$. The centre triangle $\{D, E, F\}$ is bounded by edges of the complex but is not a face.

with the node set $\mathcal{X}_0$, the edge set $\mathcal{X}_1$, and the face set $\mathcal{X}_2$. Appendix A gives the formal definitions of cell complexes and their neighbourhoods.

A cell complex of order 2 extends a graph by allowing a cycle of edges to be declared a face. A group of nodes, such as a set of items, thereby gains a representation of its own, instead of being represented only by the pairwise links between its members. The relation between a face and its edges is one-directional. By regularity, a face in the complex has all of its edges in the complex. However, the presence of those edges does not imply that the face is in the complex.

Example in recommender systems. Figure 2.2 shows the example used throughout this chapter, a complex of order 2 with six items as its 0-cells, nine item–item edges as its 1-cells, and three faces as its 2-cells. Assume, as one possible rule for building the complex, that a group of items forms a face when the items are frequently consumed together, and that this rule yields the three faces $f_1 = \{A, D, F\}$, $f_2 = \{D, B, E\}$, and $f_3 = \{F, E, C\}$. The item–item pairs within these faces are the edges induced by them and are part of the complex, and by regularity each item of these faces is likewise part of the complex. The centre triple $\{D, E, F\}$ has all three of its edges and all of its items in the complex, since each of its edges bounds one of the three faces, yet it is not itself a face.

The complex in Figure 2.2 is also a simplicial complex. A simplex generalises a triangle or a tetrahedron to arbitrary orders, and a simplex of order zero, one, two, or three is a point, a line segment, a triangle, or a tetrahedron, respectively [15]. A simplicial complex is a collection of simplices that contains, with every simplex, all of its lower-order simplices [15]. Every face in Figure 2.2 is a triangle with its three edges and three items in the complex, which makes the complex simplicial. A cell complex relaxes the shape of its cells and allows a face bounded by a cycle of any length [15].

2.3.2. Incidence Matrices

The structure of a cell complex of order 2 is fully described by two incidence matrices, which generalise the incidence matrix of a graph [15, 25]. The node-to-edge matrix $B_1 \in \{0, 1\}^{|\mathcal{X}_0| \times |\mathcal{X}_1|}$ has one column per edge, with a 1 in the rows of the two nodes the edge joins. The edge-to-face matrix $B_2 \in \{0, 1\}^{|\mathcal{X}_1| \times |\mathcal{X}_2|}$ has one column per face, with a 1 in the rows of the edges that bound the face. In Figure 2.2 the column of B_1 for the edge e_{AD} has 1s in the rows of A and D, and the column of B_2 for the face f_1 has 1s in the rows of e_{AD}, e_{DF}, and e_{FA}. The centre triangle has no column in B_2, since it is not a face. Every column of B_1 thus has exactly two 1s, and every column of B_2 has one 1 per edge of the face.

The incidence matrices are binary, with an entry of 1 when a cell bounds another and 0 otherwise. This is the unsigned boundary matrix of Bodnar et al. [16] and the incidence matrix of Hajij et al. [15], the form on which both message-passing frameworks operate. The incidence relation is often given signs, which encode an orientation of the cells [16]. Liu et al. [25] use entries in $\{-1, 0, +1\}$ such that $B_1^\top$ computes the difference between the values at the two endpoints of an edge and $B_2^\top$ sums the flows along the boundary of a face. Their signals are flows, which change sign when their direction is reversed. The embeddings of items carry no direction, and LightCCN's operators only sum and average the

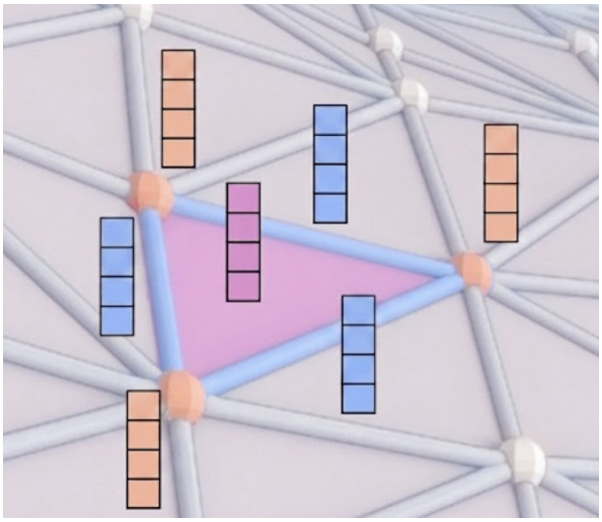

Figure 2.3: Signals on a cell complex. A d-dimensional vector is attached to every cell, shown for the 0-cells (orange), 1-cells (blue), and 2-cells (pink). Adapted from Figure 16 (left) of [15].

embeddings of neighbouring cells. With signed entries, $B_1^\top$ would give an edge the difference of its two endpoint embeddings rather than their sum.

2.3.3. Adjacency from Incidence

Cells of the same order are related through the cells they share, and these relations are products of the incidence matrices [15, 16]. Two edges are lower adjacent when they share a node, and two faces are lower adjacent when they share an edge. Two edges are upper adjacent when they bound the same face, and two nodes are upper adjacent when they are joined by an edge. In matrix form, the entry (i, j) of $B_1^\top B_1$ counts the nodes shared by edges i and j. For two different edges it is non-zero when the edges are lower adjacent, while the diagonal entry (i, i) counts the nodes of edge i and carries no relation between two cells. The same holds for $B_2^\top B_2$ and faces that share an edge, for $B_2 B_2^\top$ and edges that bound a common face, and for $B_1 B_1^\top$ and nodes joined by an edge.

In Figure 2.2, the edges e_{AD} and e_{DF} are lower adjacent through the node D and upper adjacent through the face f_1, whereas e_{DF} and e_{DE} are lower adjacent through the node D but not upper adjacent, since they bound no common face. No two faces in Figure 2.2 share an edge, so $B_2^\top B_2$ has no off-diagonal entries for this example.

2.4. Topological Deep Learning

Topological deep learning (TDL) generalises graph neural networks to higher-order topological domains such as simplicial complexes, cell complexes, hypergraphs, and combinatorial complexes [15]. A graph neural network propagates embeddings between nodes along edges. On a cell complex, every cell of every order carries an embedding, and embeddings are propagated between cells that are incident or adjacent.

2.4.1. Topological Signals

A signal of order k on a cell complex is a function $s_k : \mathcal{X}_k \to \mathbb{R}^d$ that assigns a d-dimensional vector to every k-cell [15, 25]. For $k = 0, 1, 2$, the signal is called the node, edge, and face signal, respectively. Listing these vectors as rows gives the matrix $\mathbf{H}_k \in \mathbb{R}^{|\mathcal{X}_k| \times d}$ with one row per cell. The node embeddings $\mathbf{H}_0$, the edge embeddings $\mathbf{H}_1$, and the face embeddings $\mathbf{H}_2$ generalise the embedding matrix of a graph neural network to every order of the complex. In Figure 2.2, $\mathbf{H}_0$ holds one row per item, $\mathbf{H}_1$ one row per item–item edge such as e_{AD}, and $\mathbf{H}_2$ one row per face such as f_1. Figure 2.3 illustrates such signals on cells of different orders.

The incidence matrices and their products act on these signals as propagation operators [15, 25]. A single incidence matrix moves embeddings between adjacent orders. The product $B_1^\top \mathbf{H}_0$ moves node

embeddings up to the edges, giving each edge the sum of its two endpoints, and $B_1\mathbf{H}_1$ moves edge embeddings down to the nodes, giving each node the sum of the edges it lies on. The product $B_2^\top\mathbf{H}_1$ moves edge embeddings up to the faces, giving each face the sum of its edges, and $B_2\mathbf{H}_2$ moves face embeddings down to the edges, giving each edge the sum of the faces it bounds. A product of an incidence matrix with its transpose moves embeddings within an order. The product $B_2B_2^\top\mathbf{H}_1$ gives each edge the sum of the embeddings of the edges that bound a common face with it, and $B_2^\top B_2\mathbf{H}_2$ gives each face the sum of the embeddings of the faces that share an edge with it. In Figure 2.2, $B_2^\top\mathbf{H}_1$ gives the face f_1 the sum of the embeddings of e_{AD}, e_{DF}, and e_{FA}, and $B_2\mathbf{H}_2$ distributes the embedding of f_1 to each of these three edges. All of these operators are sums. For propagation they are degree-normalised, which divides every non-zero entry by the square roots of its row sum and its column sum. For a matrix Q, let $D_r(Q)$ and $D_c(Q)$ be the diagonal matrices of its row sums and column sums. The normalised operator is

$$\mathrm{N}(Q) = D_r(Q)^{-1/2}\, Q\, D_c(Q)^{-1/2}. \tag{2.11}$$

For a symmetric matrix the two degree matrices coincide, and $\mathrm{N}(\cdot)$ reduces to the symmetric normalisation of Section 2.2.2 without self-loops, as used by LightGCN.

2.4.2. Message Passing on Cell Complexes

Message passing on a cell complex exchanges embeddings along the four local relations of Bodnar et al. [16]. A cell receives messages from the cells that bound it, from the cells it bounds, and from its lower- and upper-adjacent cells. A cell of order k exchanges messages only with cells of orders $k-1$, k, and $k+1$. In a complex of order 2 there is no direct channel between faces and nodes, and information from a face reaches a node only through the edges in between. In Figure 2.2, the face f_1 can inform the item A only through the edges e_{AD} and e_{FA}.

In matrix form, the embeddings of one order are updated from their current value and one message for each of the four relations. For the edges,

$$\mathbf{H}_1^{(\ell+1)} = \mathrm{UPDATE}\Big(\mathbf{H}_1^{(\ell)},\, B_1^\top\mathbf{H}_0^{(\ell)},\, B_2\mathbf{H}_2^{(\ell)},\, B_1^\top B_1\mathbf{H}_1^{(\ell)},\, B_2B_2^\top\mathbf{H}_1^{(\ell)}\Big), \tag{2.12}$$

where ℓ indexes the layers as in Section 2.2, and UPDATE combines the current embeddings with the four messages, following the format of Equation 2.5. The same template applies at the node and face levels with the corresponding products. The products above are an instance of the message-passing scheme of Bodnar et al., chosen for its simplicity. The general scheme computes each message with a learnable message function and aggregates the messages of each relation with any permutation-invariant function [16].

2.5. Conclusion

Three ideas from this chapter underpin the methodology that follows. Section 2.1 fixed the input domain as a binary user–item interaction matrix, equivalent to a bipartite graph $\mathcal{G}$, and formulated the recommendation problem as ranking unobserved entries of that matrix using embeddings learned with a BPR objective. Section 2.2 established LightGCN's design, reducing propagation to symmetric-normalised neighbourhood aggregation without feature transformations or non-linearities, as highly effective for the CF setting. Sections 2.3 and 2.4 extended the input domain beyond pairwise relations. A cell complex of order 2 adds faces to the bipartite graph. Its incidence matrices B_1 and B_2 and their products are the operators that move embeddings between orders and within an order. TDL provides message passing in which information moves only between cells whose orders differ by at most one.

Chapter 4 combines these building blocks. The methodology first specifies a construction that turns the bipartite graph $\mathcal{G}$ into a cell complex of order 2 by attaching faces to frequently co-consumed item triples. It then defines a LightGCN-style propagation scheme that maintains separate embeddings at every order of the complex, together with a readout that carries the pooled higher-order signal to the user embeddings. Both components inherit LightGCN's design philosophy and the locality of cell-complex message passing.

3

Related Work

This chapter places LightCCN in the context of four lines of research. Section 3.1 reviews the graph-based collaborative filtering family, which establishes the propagation philosophy LightCCN inherits. The next three sections each address higher-order structure through a different abstraction. Section 3.2 discusses linear autoencoders that incorporate a higher-order term into a shallow model. Section 3.3 reviews hypergraph and motif-based methods that carry higher-order signal as learnable representations but encode no algebraic relation between higher-order objects and the pairwise interactions they contain. Section 3.4 covers topological deep learning, the only thread that preserves this relation explicitly. Section 3.5 synthesises the four threads and places LightCCN within the resulting landscape. Figure 3.1 provides a taxonomy of the literature surveyed in this chapter.

3.1. Graph-Based Methods

The graph-based family of collaborative filtering models inherits from two earlier approaches, namely neighbourhood methods and matrix factorisation. Neighbourhood methods predict a user's preference for an item by aggregating similarities computed directly from the interaction matrix. The user-based variant of Resnick et al. [2] weights interactions by user–user similarity, and the item-based variant of Sarwar et al. [3] weights them by item–item similarity. Both treat the interaction matrix as a fixed source of similarities rather than as a basis for representation learning. Matrix factorisation [4] replaces explicit similarities with a learned low-dimensional embedding for each user and each item, and predicts interactions as inner products of these embeddings. The two approaches differ in whether they encode collaborative signal as similarities or as latent factors, but neither propagates information along multi-hop paths in the interaction graph. Multi-hop connectivity is therefore recoverable only implicitly through the choice of similarity or scoring function rather than encoded in the architecture itself. This shared limitation is addressed by graph-based methods.

Neural Graph Collaborative Filtering (NGCF) [6] places stacked graph-convolutional propagation layers on the bipartite interaction graph, encoding multi-hop user–item connectivity directly in the embedding function. Each layer aggregates neighbour messages under symmetric normalisation and keeps the feature transformations, non-linear activations, and affinity term of the GCN it adapts [7]. After L layers, the per-layer embeddings are concatenated and scored through inner products under the BPR loss [1]. The reported gains over matrix factorisation establish empirically that graph propagation carries collaborative signal that learned embeddings alone do not.

LightGCN [8] re-examines NGCF's architectural choices through ablation, removing the feature transformation and the non-linear activation independently and jointly. The joint removal not only matches NGCF but improves on it, with a 9.57% relative recall gain across the ablation benchmarks. In the implicit-feedback setting, where each node carries only an ID, layers of non-linear feature transformation cannot extract richer features and instead complicate optimisation. The resulting propagation rule retains only symmetric-normalised neighbourhood aggregation, and the final node embedding is a uniformly weighted sum across propagation layers. This pruned design is the propagation philosophy

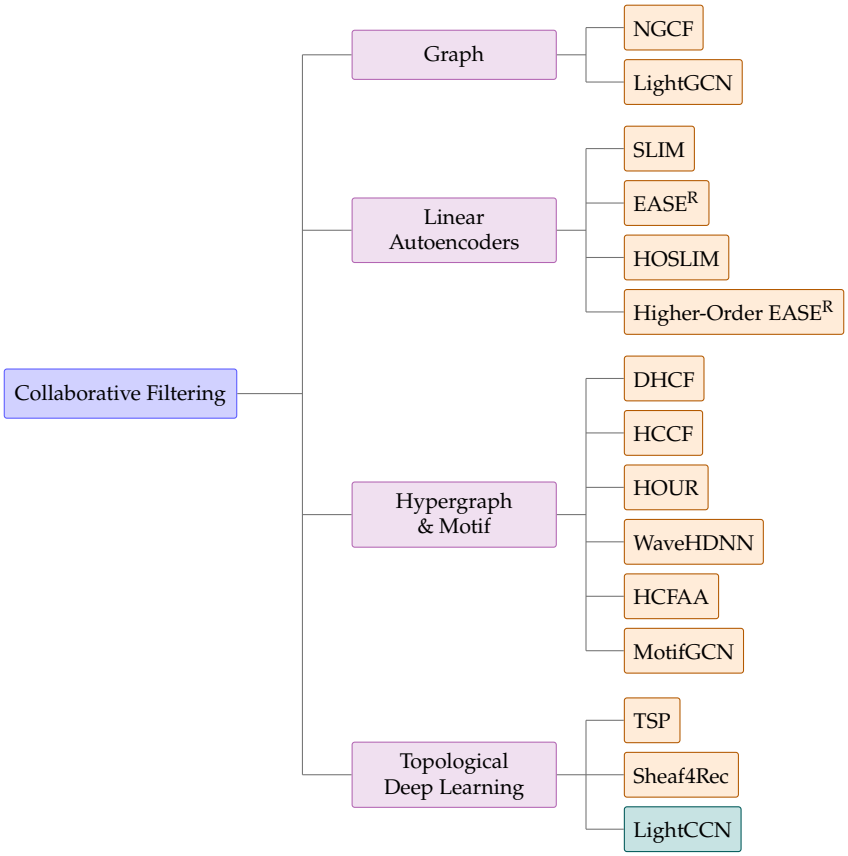

Figure 3.1: A taxonomy of the works reviewed in this chapter. The four branches correspond to Sections 3.1 to 3.4.

that LightCCN inherits.

The limitation shared across this family is that the underlying domain remains pairwise. Information is exchanged only along user–item edges, and any pattern among three or more items is recoverable only through multi-hop traversal of pairwise links rather than as a model primitive.

3.2. Linear Autoencoders

A line of work outside the graph deep-learning paradigm provides early empirical evidence on the contribution of higher-order signal. The architectural form shared by these works is a linear autoencoder, a shallow model in which a learned item–item weight matrix transforms each user's interaction vector into predicted scores over all items.

SLIM [27] learns this matrix B under elastic-net regularisation, with B encoding how much one item indicates a preference for another. EASE^R [28] replaces the elastic-net objective with a closed-form ridge regression and constrains the diagonal of the weight matrix to zero. Despite having no hidden layer, it matches or exceeds many neural alternatives on standard benchmarks. HOSLIM [29] adds a second weight matrix over preselected pairs of items, framing the joint consumption of a pair as a positive bonus to the pairwise score.

Steck and Liang [9] combine HOSLIM's two-matrix construction with the EASE^R objective and drop the non-negativity constraint. The learned higher-order interactions turn out predominantly negative. The pairwise term overcounts the shared reasons for item co-occurrence, and the higher-order term acts as a corrective subtraction rather than an additive bonus. This effect is concentrated among users with low-to-moderate activity. The higher-order term absorbs dense co-occurrence patterns and frees the pairwise term to fit less-active users.

The contribution of this thread lies less in the architectural form than in its empirical claim. Higher-order

item interactions carry collaborative signal that pairwise models miss. The signal is largely corrective rather than additive.

Two properties, however, limit how far this finding transfers to representation learning. First, each higher-order interaction contributes a single scalar weight rather than a vector representation of the items involved. The model thus captures the predictiveness of one item for another, but not the factors that relate them. Second, the set of interactions is fixed before training by selecting frequent item pairs. The model cannot adjust which groups matter during learning. Both limitations separate this thread from the graph deep-learning approaches, which carry higher-order signal within the learnable vector representations.

3.3. Hypergraph and Motif Methods

The hypergraph thread carries higher-order signal as learnable representations and so addresses the architectural limitation of the shallow autoencoders. This comes at the cost of representing multi-entity relations as undifferentiated hyperedges, with no algebraic relation linking higher-order objects to the pairwise interactions they contain.

Dual Channel Hypergraph Collaborative Filtering (DHCF) [10] was one of the first to apply hypergraph convolutional networks to bipartite collaborative filtering. From the interaction graph it derives separate user and item hypergraphs, whose hyperedges are the k-hop reachable sets. DHCF outperforms each of the four pairwise baselines tested (BPR-MF, GC-MC, PinSage, NGCF) across all four datasets. The relative gains in Recall@20 grow with sparsity and reach up to 186% on the sparsest datasets. The structure itself, however, is not learned, with the hyperedges fixed once at preprocessing and unchanged during training. The authors identify learnable weighting of different hyperedge groups as a natural direction for follow-up work.

Hypergraph Contrastive Collaborative Filtering (HCCF) [11] subsequently introduces the contrastive variant that has become a widely followed template within this thread. The model propagates embeddings over two structures in parallel, namely the bipartite graph and a hypergraph learned end-to-end with the rest of the model. The graph provides a local view and the hypergraph a global view of the embeddings, and a contrastive loss aligns the two. HCCF targets the over-smoothing of deep graph-based models and the scarcity of supervision signals, and reports average relative gains in NDCG@20 of 24.7% over LightGCN and 33.8% over DHCF across three benchmarks. Its hyperedges, however, are learned as soft memberships rather than being fixed before training. This adds free parameters, and thus degrees of freedom, to be learned from the supervision signals that HCCF identifies as scarce.

Subsequent work refines this template along three axes, namely the hyperedge structure, the propagation over it, and the auxiliary objective. A parallel motif-based thread replaces hyperedges with motif-weighted adjacencies.

HOUR [12] replaces HCCF's end-to-end-learned hypergraph with one derived deterministically from non-negative matrix factorisation of the interaction matrix. Each latent factor becomes a hyperedge connecting its top users and items, ranked by their values on that factor. A graph convolution with feature transformations and non-linear activations then propagates over graphs derived from the hypergraph. The reported gains range from 3% to 12% over the next-best baseline across four datasets. Ablations indicate that the learned higher-order embeddings carry signal beyond what pairwise propagation captures. Group membership, however, remains binary, with no algebraic relation between a hyperedge and the pairwise interactions whose endpoints define it.

Two subsequent works in the contrastive-hypergraph paradigm address concerns orthogonal to hypergraph construction. WaveHDNN [13] targets heterophilic interactions and over-smoothing through two complementary encoders. The first uses an equivariant operator to differentiate the messages passed to heterogeneous nodes, and the second uses wavelet transforms to tune the spread of information across the hypergraph. WaveHDNN surpasses the second-best model by 3.9% to 7.2% in NDCG@20 across three benchmarks. HCFAA [14] addresses a known limitation of random graph augmentation in contrastive collaborative filtering [30]. Indiscriminate dropout removes the key nodes and edges of the graph topology and significantly degrades performance. HCFAA replaces this random

augmentation with a centrality-guided scheme that preserves these elements, and reports average gains of 17.3% over LightGCN and 25.1% over DHCF.

MotifGCN [31] pursues a route parallel to the hypergraph thread and applies network motifs to bipartite collaborative filtering. The model defines motif adjacency matrices for the seven motifs of up to four nodes and propagates with a LightGCN backbone over the original graph combined with one motif adjacency. The reported gains over LightGCN range from about 60% on two benchmarks to more than 100% on the other two. The result confirms that local higher-order patterns expose collaborative signal beyond multi-hop propagation alone. The motif type and its combination coefficient, however, are hyperparameters fixed before training rather than learned. The authors defer learning these coefficients to future work. Moreover, no entity at the motif level is created, so outputs remain node-level with no higher-order representation that could be examined directly.

The common limitation across this thread is structural. Hyperedges and motif-weighted adjacencies both extend the pairwise graph framework, without an algebraic operation that links higher-order objects to the pairwise edges they contain. A hyperedge is a flat set of node memberships, and motif co-occurrence is folded into a reweighted node–node adjacency. The relationship between a group and its constituent pairwise interactions therefore enters the model only implicitly, through their effects on node embeddings. The next thread encodes this relationship explicitly through topology rather than through reweighted adjacencies.

3.4. Topological Deep Learning Methods

Topological deep learning generalises graph neural networks from graphs to higher-order domains such as simplicial and cell complexes [15], with recent work exploring representation learning on these structures [32]. Two works apply this framework to collaborative filtering and leverage it at different points in its design space. TSP [18] preserves the distinction in order between pairwise edges and higher-order cells but applies the simplicial complex as post-hoc smoothing on top of a pretrained model. Sheaf4Rec [19] stays at order 1 but enriches the algebraic structure attached to each cell, learning it end-to-end with the rest of the model.

TSP targets popularity bias and operates on top of a frozen pretrained LightGCN backbone. The interaction graph is first augmented with edges between the nodes whose pretrained embeddings are most similar. This densifies the graph sufficiently for triangles and higher-order simplices to form. The simplicial complex is built on this augmented graph, and each simplex receives the sum of the pretrained embeddings of its nodes. The embeddings at each order are smoothed over several layers, mapped back to the nodes, and concatenated with the pretrained node embedding for scoring. TSP demonstrates that simplicial complexes can capture higher-order signal in collaborative filtering, with gains concentrated on tail items and overall improvements over the backbone on five datasets.

Three properties of the design, however, limit what TSP shows about the role of topology in recommendation. First, propagation is order-restricted. Within each smoothing iteration the embedding at order k stays at order k, and the per-order views are mixed only at the final concatenation. Independent evidence from the simplicial-network literature [17] indicates that explicit message passing between cells of adjacent order produces measurable gains on tasks such as simplex prediction. Second, because the bipartite user–item graph is triangle-free, the simplicial complex requires the similarity-based augmentation. This introduces a similarity-threshold hyperparameter and a dependence on the pretrained encoder. Third, higher-order structure does not inform training. TSP operates on frozen pretrained node embeddings with no learnable higher-order embeddings. The topological signal acts as post-hoc smoothing rather than participating in representation learning.

Sheaf4Rec, in contrast to TSP, trains its sheaf structure as part of the model, with the architecture built on cellular sheaves over the graph. Each node and each edge is associated with a vector space called a stalk, and adjacent stalks are connected by linear restriction maps. User and item embeddings are represented as signals on the nodes and preference scores as signals on the edges. Propagation follows the Neural Sheaf Diffusion of Bodnar et al. [33], with the sheaf and the layer weights trained together with the rest of the model. The model performs best at five sheaf-diffusion layers, which the authors highlight as evidence of robustness to over-smoothing. Improvements over the strongest of five baselines, including NGCF and LightGCN, reach up to 11.29% in NDCG@10 across three benchmarks.

Two limitations follow from this approach. First, propagation remains at order 1. The model enriches each node and edge with a vector space rather than adding cells of higher order. Second, the authors report that the sheaf Laplacian cannot be applied directly to a bipartite graph. The model operates instead on a projection of the graph, which retains its properties without being strictly bipartite.

3.5. Discussion

The four threads above converge on the same empirical observation. Higher-order structure carries collaborative signal that pairwise models miss. They diverge in how that signal is encoded. The graph-based family operates on a pairwise domain and treats any higher-order regularity as a phenomenon to be recovered through multi-hop propagation rather than represented directly. The linear autoencoder thread demonstrates the existence of corrective higher-order signal in a shallow setting and characterises its concentration on less-active users. The hypergraph and motif works carry higher-order signal as learnable representations, but encode no algebraic relation between higher-order objects and the pairwise interactions they contain. The topological thread preserves the framework's algebraic primitives, but neither of its two collaborative-filtering instantiations uses the framework in full. TSP adds higher-order simplices but only after training, and Sheaf4Rec learns its structure end-to-end but stays at order 1.

LightCCN is positioned at the intersection of these lines of work. It adopts the linear-autoencoder premise that higher-order item interactions carry information that pairwise models lose. It shares with the hypergraph works the representation of higher-order signal in learnable vector spaces, and with LightGCN the removal of feature transformations and non-linearities from propagation. It follows the topological thread in encoding higher-order structure hierarchically rather than as flat hyperedges. The chapter that follows describes how LightCCN builds the higher-order structure from item co-occurrence and integrates it into a LightGCN backbone.

4

Methodology

This chapter presents LightCCN, a collaborative filtering model that extends LightGCN with the higher-order structure of a cell complex of order 2. Section 4.1 states the problem, and Section 4.2 the design principles. Section 4.3 describes how the cell complex is constructed from the interaction data. Section 4.4 assembles the propagation operators of the complex. Section 4.5 defines the stateful propagation scheme that carries embeddings at every order of the complex, and Section 4.6 defines the higher-order readout that injects the pooled cell embeddings into the user representations before scoring. Section 4.7 states the training objective, Section 4.8 analyses computational complexity, and Section 4.9 discusses the strengths and limitations of these design choices. Figure 4.1 shows the higher-order structure the model operates on, a cell complex of order 2 built from the interaction data.

4.1. Problem Formulation

Let $\mathcal{U}$ denote the set of M users and $\mathcal{I}$ the set of N items. The available data is implicit feedback, a binary interaction matrix $X \in \{0,1\}^{M \times N}$ with $x_{ui} = 1$ if user u interacted with item i and $x_{ui} = 0$ otherwise, with no content features and no rating values (Section 2.1.1). The task is top-K recommendation, where for every user the items the user has not yet interacted with are ranked and the highest-ranked ones are recommended. Each user and item is assigned a learned embedding, and a user–item pair is scored by the inner product of the final embeddings. Training uses the pairwise BPR objective (Equation 2.1), which trains each observed interaction to score higher than a sampled unobserved one.

Within this setting, the problem is to learn representations for groups of jointly consumed items while keeping the hierarchy that relates them to the pairs and items they are composed of. Chapter 1 established that the interaction graph cannot represent such groups as objects of their own, and that pairwise co-occurrence cannot accurately tell whether a larger group of items is consumed together by the same users or only in pairs by different users. The rest of the chapter presents LightCCN, which meets this requirement with a cell complex of order 2 built from X.

4.2. Overview and Design Principles

LightCCN follows three principles. First, it inherits the design philosophy that made LightGCN effective for collaborative filtering [8]: no feature transformation matrices, no non-linear activations inside propagation, symmetric degree normalisation, and averaged layers. The only trainable parameters beyond the LightGCN embedding table are nine scalars: seven channel-mixing weights and two readout gates. Second, higher-order relations enter the model as cells of a cell complex of order 2 (Section 2.3) rather than as reweighted pairwise edges, so the hierarchy between an item group, the item pairs it contains, and the items themselves is recorded by the incidence matrices B_1 and B_2 (Section 2.3.2). Third, propagation follows the message passing of Section 2.4.2, in which a cell of order k exchanges embeddings only with cells of order $k-1$, k, and $k+1$.

The model has two configurations that differ in exactly one component. **LightCCN-prop** performs higher-order propagation only. **LightCCN-full** additionally applies the higher-order user readout of

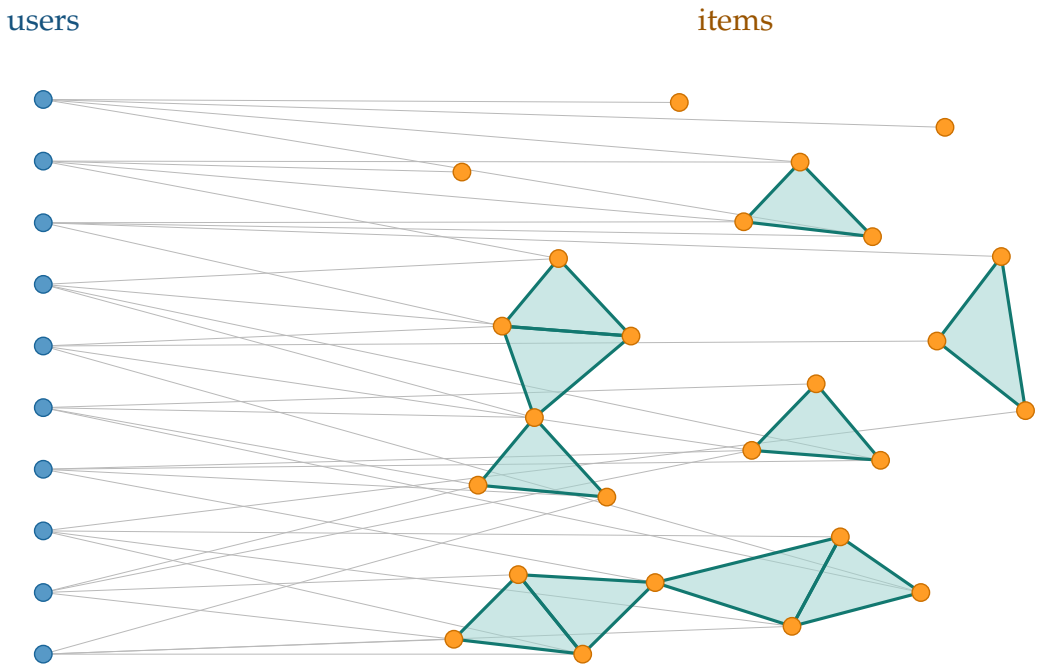

Figure 4.1: The cell complex of order 2 built from the interaction data. Users are on the left, items on the right, and thin grey lines are observed interactions. Filled triangles are the faces, each one an item triple, and the thicker lines are the item–item edges induced by those faces. Some faces share an edge, some share a single item, and items that lie on no face carry no thicker lines.

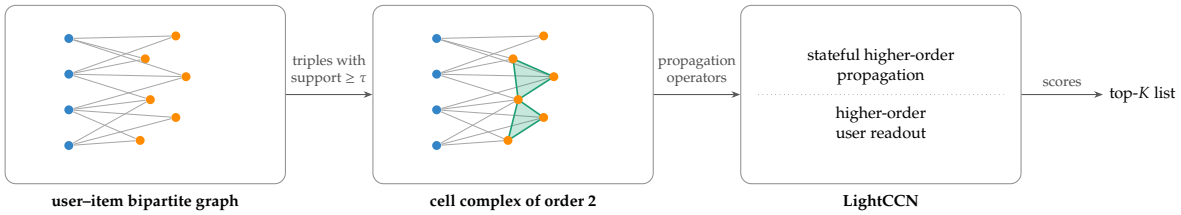

Figure 4.2: The three stages of the model. The observed interactions form a user–item bipartite graph. The support threshold τ then admits item triples as faces and induces the item–item edges between them, which turns the graph into a cell complex of order 2. LightCCN propagates over that complex and applies the higher-order user readout before scoring. The third stage is expanded in Figure 4.3.

Section 4.6. Because the readout gates are initialised at zero, LightCCN-full starts training as exactly LightCCN-prop and opens the readout only if the ranking loss rewards it. This pairing gives the ablation structure used to answer RQ2: comparing LightCCN-prop with LightGCN isolates the contribution of higher-order propagation, and comparing LightCCN-full with LightCCN-prop isolates the contribution of the readout.

Figure 4.2 outlines the pipeline of the model. The model runs in three stages. The interaction data is first turned into a cell complex. Embeddings are then propagated at every order of the complex. Finally, the readout injects the pooled cell signal into the user representations before scoring.

4.3. Cell-Complex Construction

The complex is built once from the user–item interaction graph, using the training interactions only, and is fixed thereafter.

Nodes and edges. The 0-cells are the users and the items, $\mathcal{X}_0 = \mathcal{U} \cup \mathcal{I}$. The edge set is $\mathcal{X}_1 = \mathcal{E} \cup \mathcal{E}_{\mathcal{I}}$, where $\mathcal{E}$ is the set of observed user–item interactions of Equation 2.2, each bounded by a user and an

Table 4.1: Propagation operators of LightCCN. All are fixed. N(·) is the normalisation of Equation 2.11 and off(·) removes the diagonal.

operator	size	definition	role
$\hat{A}_0$	$(M+N)\times(M+N)$	$\mathrm{N}(A)$	nodes → nodes along $\mathcal{E}$ (LightGCN channel)
$\hat{B}_1^{\uparrow}$	$\lvert\mathcal{E}_{\mathcal{I}}\rvert\times(M+N)$	$\mathrm{N}(B_1^{\top})$	nodes → edges
$\hat{B}_1^{\downarrow}$	$(M+N)\times\lvert\mathcal{E}_{\mathcal{I}}\rvert$	$\mathrm{N}(B_1)$	edges → nodes
$\hat{B}_2^{\uparrow}$	$\lvert\mathcal{X}_2\rvert\times\lvert\mathcal{E}_{\mathcal{I}}\rvert$	$\mathrm{N}(B_2^{\top})$	edges → faces
$\hat{B}_2^{\downarrow}$	$\lvert\mathcal{E}_{\mathcal{I}}\rvert\times\lvert\mathcal{X}_2\rvert$	$\mathrm{N}(B_2)$	faces → edges
$\hat{A}_1$	$\lvert\mathcal{E}_{\mathcal{I}}\rvert\times\lvert\mathcal{E}_{\mathcal{I}}\rvert$	$\mathrm{N}(\mathrm{off}(B_2B_2^{\top}))$	edges → edges that bound a common face
$\hat{A}_2$	$\lvert\mathcal{X}_2\rvert\times\lvert\mathcal{X}_2\rvert$	$\mathrm{N}(\mathrm{off}(B_2^{\top}B_2))$	faces → faces that share an edge

item, and $\mathcal{E}_{\mathcal{I}}$ is a set of item–item pairs, each bounded by two items. The set $\mathcal{E}_{\mathcal{I}}$ is induced by the faces, which are defined in the next paragraph.

Faces. For an item triple $\{i, j, k\} \subseteq \mathcal{I}$, define its *support* as the number of users that interacted with all three items,

$$\mathrm{supp}(i, j, k) \;=\; \left|\mathcal{N}_i \cap \mathcal{N}_j \cap \mathcal{N}_k\right|, \tag{4.1}$$

where $\mathcal{N}_i$ is the user neighbourhood of item i from Equation 2.4. Given a support threshold $\tau \in \mathbb{N}$, the face set is

$$\mathcal{X}_2 \;=\; \left\{\{i, j, k\} : \mathrm{supp}(i, j, k) \geq \tau\right\}. \tag{4.2}$$

The support of a triple measures how much evidence the data provides that its three items are consumed together. The threshold τ therefore trades the number of faces against their support. A low τ admits many faces, each backed by few users, whereas a high τ keeps only the groups whose co-consumption is confirmed by many users. Increasing τ reduces the number of faces. For two thresholds $\tau_1 < \tau_2$, the face set obtained at τ_2 is a subset of the face set obtained at τ_1. How τ is chosen per dataset is part of the experimental design (Section 5.1.2), and its effect on accuracy is the subject of RQ4. Each face is bounded by its three item pairs, which form the item–item edge set

$$\mathcal{E}_{\mathcal{I}} \;=\; \left\{\{i, j\} : \{i, j\} \subset F \text{ for some } F \in \mathcal{X}_2\right\}. \tag{4.3}$$

A pair is a 1-cell when it is part of at least one face. Every face has its three bounding edges in the complex and every edge its two endpoint nodes, satisfying the regularity condition of Section 2.3.1.

Every face is an item triple, which makes the complex a simplicial complex. The construction and the operators of the following sections use only the incidence matrices of Section 2.3.2, which are defined for faces bounded by any number of edges. Item triples are the smallest groups beyond pairs and are the only group size evaluated in this thesis. Since the support of a group can only decrease as the group grows, larger groups are backed by fewer users and are left for future work.

4.4. Propagation Operators

The two kinds of 1-cells enter propagation through different operators. The interactions $\mathcal{E}$ enter through the adjacency matrix A of Equation 2.3, normalised as $\hat{A}_0$, which is the propagation operator of LightGCN. The item–item 1-cells enter through the incidence matrices of Section 2.3.2. $B_1 \in \{0,1\}^{(M+N)\times|\mathcal{E}_{\mathcal{I}}|}$ has one column per item–item edge, marking its two endpoint items, and $B_2 \in \{0,1\}^{|\mathcal{E}_{\mathcal{I}}|\times|\mathcal{X}_2|}$ has one column per face, marking its three bounding edges. The item–item 1-cells receive an embedding matrix of their own, while the user–item 1-cells do not. A user–item 1-cell bounds no face and shares no face with any edge. A state on it could only carry its two endpoints back to the nodes, which the operator $\hat{A}_0$ already does. All operators are fixed and normalised with N(·) of Equation 2.11. Table 4.1 lists them.

The incidence operators $\hat{B}_1^{\uparrow}$, $\hat{B}_1^{\downarrow}$, $\hat{B}_2^{\uparrow}$, and $\hat{B}_2^{\downarrow}$ move embeddings between adjacent orders as in Section 2.4.1, and $\hat{A}_1$ and $\hat{A}_2$ are the upper adjacency of the edges and the lower adjacency of the faces of Section 2.3.3. Two edges are neighbours in $\hat{A}_1$ when they bound a common face, and two faces are neighbours in $\hat{A}_2$

when they share an edge. The diagonals of $B_2 B_2^\top$ and $B_2^\top B_2$ are removed before normalisation, mirroring the absence of self-loops in LightGCN.

4.5. Stateful Higher-Order Propagation

The model maintains one embedding matrix per order (Section 2.4.1): node states $\mathbf{H}_0^{(\ell)} \in \mathbb{R}^{(M+N)\times d}$, edge states $\mathbf{H}_1^{(\ell)} \in \mathbb{R}^{|\mathcal{E}_I|\times d}$, and face states $\mathbf{H}_2^{(\ell)} \in \mathbb{R}^{|\mathcal{X}_2|\times d}$. Edge and face states are seeded once from the items they contain and are then carried and refined across layers, rather than being recomputed from node embeddings at every layer. Cells thereby accumulate their own representation of the group they describe, at no cost in learned parameters. Figure 4.3 traces the resulting information flow, both for a propagation layer and for the readout of Section 4.6.

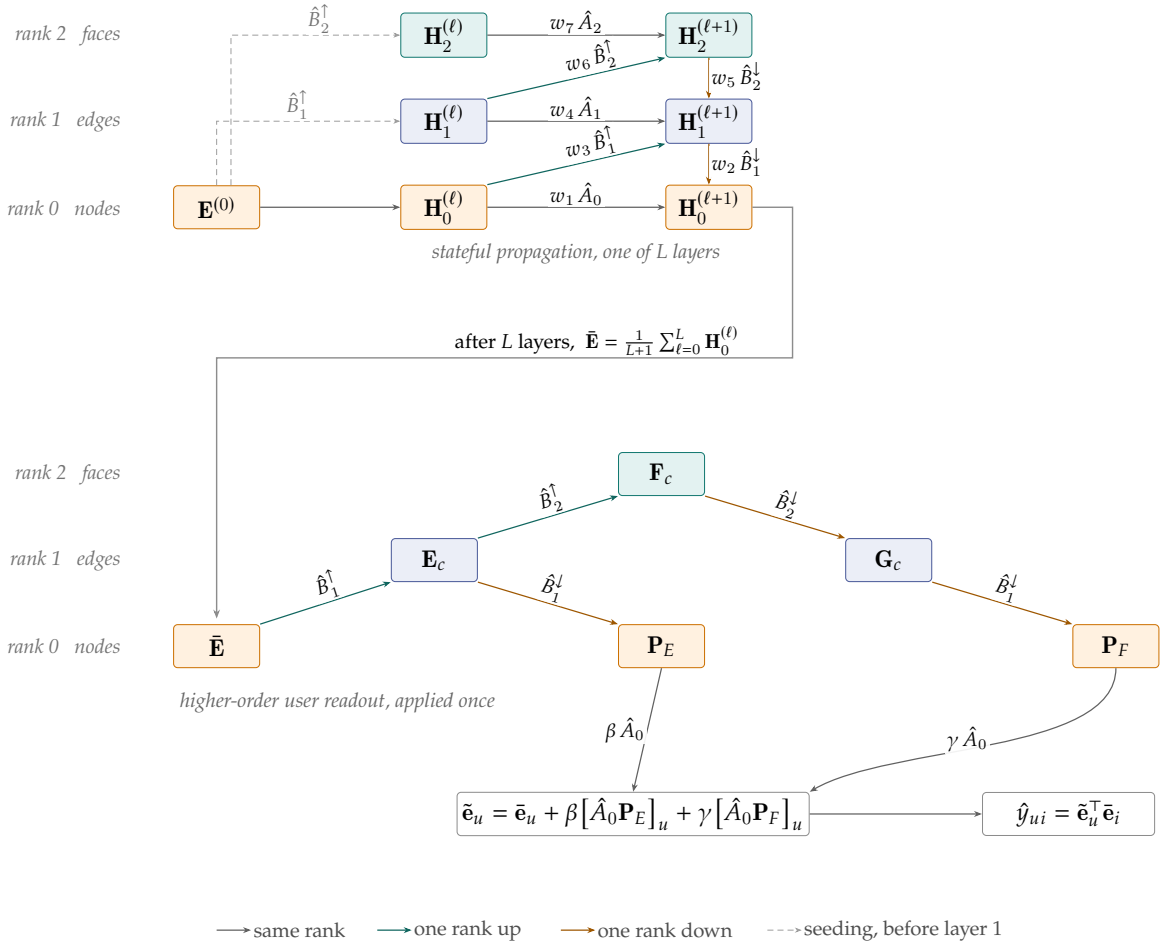

Figure 4.3: Information flow in LightCCN, on the three orders of the complex. The upper row is one propagation layer, and the lower rows are the readout applied after the layer average, where $\mathbf{E}_c$ and $\mathbf{F}_c$ are edge and face summaries of $\bar{\mathbf{E}}$, $\mathbf{G}_c$ is the edge-level result of pushing $\mathbf{F}_c$ down one order, and $\mathbf{P}_E$ and $\mathbf{P}_F$ are the item-level pools that the gates β and γ carry to the users.

Initialisation. The node embedding matrix is the learnable embedding table, $\mathbf{H}_0^{(0)} = \mathbf{E}^{(0)}$, exactly as in LightGCN. The edge and face states are seeded through the incidence operators:

$$\mathbf{H}_1^{(0)} = \hat{B}_1^{\uparrow}\mathbf{H}_0^{(0)}, \qquad \mathbf{H}_2^{(0)} = \hat{B}_2^{\uparrow}\mathbf{H}_1^{(0)}, \tag{4.4}$$

so an edge starts as the normalised average of its two items and a face as the normalised average of its three edges.

Layer update. Each layer updates the three embedding matrices in a cascade from the faces down to the nodes, such that the updated higher-order information flows to lower orders within the same layer:

$$\mathbf{H}_2^{(\ell+1)} = w_6\,\hat{B}_2^{\uparrow}\mathbf{H}_1^{(\ell)} \;+\; w_7\,\hat{A}_2\,\mathbf{H}_2^{(\ell)}, \tag{4.5}$$

$$\mathbf{H}_1^{(\ell+1)} = w_3\,\hat{B}_1^{\uparrow}\mathbf{H}_0^{(\ell)} \;+\; w_4\,\hat{A}_1\,\mathbf{H}_1^{(\ell)} \;+\; w_5\,\hat{B}_2^{\downarrow}\mathbf{H}_2^{(\ell+1)}, \tag{4.6}$$

$$\mathbf{H}_0^{(\ell+1)} = w_1\,\hat{A}_0\,\mathbf{H}_0^{(\ell)} \;+\; w_2\,\hat{B}_1^{\downarrow}\mathbf{H}_1^{(\ell+1)}. \tag{4.7}$$

Faces aggregate from their bounding edges and from faces that share an edge (Eq. 4.5). Edges aggregate from their endpoint items, from edges that co-bound a face, and from the just-updated faces they bound (Eq. 4.6). Nodes aggregate through the LightGCN channel and from the just-updated edges incident to them (Eq. 4.7). Restricted to Equation 4.7 with $w_2 = 0$, the scheme is exactly LightGCN (Eq. 2.9). Closing the higher-order channels recovers the backbone. Of the four local relations in the general update of Equation 2.12, the edge update omits one, the lower adjacency $B_1^\top B_1$ that links edges sharing an endpoint. Bodnar et al. prove this message redundant for the expressive power of cellular Weisfeiler–Lehman refinement, since what it transmits in one step travels in two steps through the boundary messages the model already carries, while noting that it may still encode a useful inductive bias for certain tasks [16]. The user–item interactions are also not lifted to 1-cells, so the model stores edge embeddings only for the item–item edges of the complex. The ablation of Appendix F tests both choices. Adding the channel over the item–item edges leaves accuracy statistically unchanged relative to LightCCN-full on every dataset, and adding it together with every interaction as a 1-cell is significantly worse on five of the six datasets and never better.

Mixing weights. The scalars (w_1, w_2), (w_3, w_4, w_5), and (w_6, w_7) are softmax outputs of learnable logits, one softmax per order. They are therefore positive and sum to one within each order, so each update is a convex combination of its incoming channels and the scale of the embeddings is preserved across layers without normalisation layers. The learned values of these weights are themselves an object of analysis in RQ2, since they reveal how much of each channel the model chooses to use.

Layer combination. After L layers, the final node embeddings average the node embedding matrix over layers, exactly as in Equation 2.10:

$$\bar{\mathbf{E}} = \frac{1}{L+1}\sum_{\ell=0}^{L}\mathbf{H}_0^{(\ell)}. \tag{4.8}$$

The edge and face states serve propagation only and do not participate in the layer average.

4.6. Higher-Order User Readout

Equation 4.7 delivers edge and face information to items, and users receive it only indirectly through subsequent user–item hops. The readout adds one explicit, parameter-free path from the higher-order cells to the users, gated by two scalars.

From the layer-combined embeddings $\bar{\mathbf{E}}$, the readout first summarises each cell by its final content and pools the summaries back to the item level:

$$\mathbf{P}_E = \hat{B}_1^{\downarrow}\big(\hat{B}_1^{\uparrow}\bar{\mathbf{E}}\big), \qquad \mathbf{P}_F = \hat{B}_1^{\downarrow}\hat{B}_2^{\downarrow}\big(\hat{B}_2^{\uparrow}\hat{B}_1^{\uparrow}\bar{\mathbf{E}}\big). \tag{4.9}$$

$\mathbf{P}_E$ assigns to every item the normalised average of the edge summaries of the edges that contain it, and $\mathbf{P}_F$ the corresponding face-level quantity. Both are structurally zero on user rows and on items that lie on no edge or face. One hop through the interaction graph then carries the pooled signal to the users that consume those items, and two gates control its magnitude:

$$\tilde{\mathbf{e}}_u = \bar{\mathbf{e}}_u \;+\; \beta\big[\hat{A}_0\mathbf{P}_E\big]_u \;+\; \gamma\big[\hat{A}_0\mathbf{P}_F\big]_u, \tag{4.10}$$

applied to user rows only. Item embeddings remain $\tilde{\mathbf{e}}_i$. The gates β (edges) and γ (faces) are the only parameters of the readout and are initialised at zero. Hence, LightCCN-full starts exactly at

LightCCN-prop and the higher-order injection grows only as far as the ranking loss pushes it. Scores remain plain inner products, as in the inner-product score of Section 2.1.1:

$$\hat{y}_{ui} = \tilde{\mathbf{e}}_u^\top \bar{\mathbf{e}}_i. \tag{4.11}$$

The readout is personalised rather than a global shift. Since $[\hat{A}_0 \mathbf{P}_E]_u$ is a weighted average over the items user u interacted with, two users receive different corrections unless their histories coincide on the covered items.

4.7. Training Loss

All parameters are trained end-to-end with the BPR objective of Equation 2.1 on scores from Equation 4.11, following LightGCN [8]. For a mini-batch $\mathcal{B}$ of triples (u, i, j), where i is an observed and j a sampled unobserved item of user u, the loss is

$$\mathcal{L} \;=\; -\frac{1}{|\mathcal{B}|} \sum_{(u,i,j)\in\mathcal{B}} \ln \sigma\big(\hat{y}_{ui} - \hat{y}_{uj}\big) \;+\; \frac{\lambda}{2\,|\mathcal{B}|} \sum_{(u,i,j)\in\mathcal{B}} \Big(\|\mathbf{e}_u^{(0)}\|_2^2 + \|\mathbf{e}_i^{(0)}\|_2^2 + \|\mathbf{e}_j^{(0)}\|_2^2 \Big). \tag{4.12}$$

The ℓ_2 term regularises only the layer-0 embeddings of the users and items in the mini-batch. The nine scalars, the mixing logits and the readout gates, are not regularised. The trainable parameters are the node embedding table $\mathbf{E}^{(0)}$, the seven mixing logits, and the two readout gates. LightCCN-full therefore has exactly nine scalar parameters more than LightGCN at equal embedding size.

4.8. Computational Complexity

Every operator application multiplies a fixed sparse matrix with the dense embedding matrix of an order. Its cost is the number of non-zero entries of the operator times the embedding dimension d, since each non-zero entry contributes one multiplication and one addition per embedding dimension. One propagation layer costs

$$\mathcal{O}\Big(\big(\mathrm{nnz}(X) + 2|\mathcal{E}_I| + 3|\mathcal{X}_2| + \mathrm{nnz}(\hat{A}_1) + \mathrm{nnz}(\hat{A}_2)\big)\, d \Big), \tag{4.13}$$

where $\mathrm{nnz}(X)$ is the number of interactions (the LightGCN cost), $2|\mathcal{E}_I|$ and $3|\mathcal{X}_2|$ count the node–edge and edge–face incidences, two per edge and three per face, and the adjacency terms count the pairs of edges that share a face and the pairs of faces that share an edge. The readout adds a constant number of the same products once per forward pass. Memory adds the carried edge and face embeddings, $(|\mathcal{E}_I| + |\mathcal{X}_2|)\, d$ floats, with no additional parameters. Both overheads scale with the size of the complex, which the threshold τ controls directly. At the operating points selected in this thesis (Table 5.1), the complex has fewer cells than the graph has interactions on five of the six datasets, by factors ranging from roughly two (Beidian) and three (Epinions) to roughly one thousand (Gowalla). On Foursquare-NYC it is about three times larger, since that dataset selects the abundant end of its sweep.

4.9. Discussion

Design rationale.

- **Stateful cells rather than recomputed cells.** Recomputing edge and face embeddings from node embeddings at every layer would make higher-order cells a deterministic function of the nodes and prevent them from accumulating group-level information. Carrying their state gives each cell a persistent representation at no cost in learned parameters. Empirically, the backbone with stateful edges and faces is the only variant that is never significantly worse: dropping the faces, or recomputing cell states instead of carrying them, each fails on exactly one dataset (Appendix D).
- **Convex channel mixing.** Softmax mixing keeps every update a convex combination, which preserves the embedding scale without normalisation layers. It also remains faithful to LightGCN's transformation-free design and makes the learned weights interpretable as channel shares.
- **A user-side readout.** The structural channels of Equations 4.5–4.7 touch item rows only, so without a readout the user tower is updated exclusively by the LightGCN channel. The readout closes this

asymmetry along the incidence hierarchy, face → edge → item → user, moving information one order at a time instead of jumping from faces to users directly. The target choice is validated by ablation. Redirecting the same readout to the item tower never helps and is significantly worse on four datasets, with the item-side gates remaining close to zero or turning negative. Injecting into both towers helps on exactly one dataset while hurting on two (Appendix C).

- **Zero-initialised gates.** Starting the readout closed guarantees that LightCCN-full can never start worse than its propagation-only counterpart and turns the learned gate trajectory into evidence of whether the ranking loss values the higher-order signal.
- **Interactions without embeddings.** The user–item 1-cells enter propagation through $\hat{A}_0$ only (Section 4.4). Giving them embeddings of their own lets the edge channel dominate the node-level softmax and is significantly worse on all six datasets (Appendix E).
- **A fixed complex as inductive bias.** The complex is fixed before training, and which triples become faces is decided by support counting rather than learned. This structure is the inductive bias of the model. It constrains propagation to the observed co-consumption structure, which prevents over-parameterisation [25] and suits the sparse data of implicit feedback.

Limitations. The mixing weights are global scalars per channel, so all cells of an order share the same channel allocation. Finer mixing is expressible in the same framework at the cost of more parameters. Examples are a separate weight for every cell, different weights for users and for items, or weights that depend on item popularity. A second limitation is that faces are restricted to item triples, and larger groups appear only as overlapping triples. Finally, the quality of the higher-order signal depends on τ. As RQ4 shows, an ill-chosen τ can make the structural channels useless or harmful, which is why τ selection is part of the experimental protocol rather than a fixed constant.

5

Experiments

This chapter presents the experiments conducted to evaluate LightCCN. Section 5.1 outlines the experimental setup, covering the datasets, the evaluation protocol, and the compared models. Section 5.2 reports the results, with one subsection per research question. Supporting tables at the remaining cutoffs are presented in the appendix.

5.1. Experimental Setup

5.1.1. Data and Protocol

Experiments use six public implicit-feedback datasets spanning movie reviews, product opinions, social e-commerce, and location check-ins: CiaoDVD [34], Epinions [35], Beidian [12], Foursquare-NYC and Foursquare-TKY [36], and Gowalla [8, 37]. All interactions are binarised. An observed user–item pair counts as positive feedback regardless of any rating value, following the implicit-feedback formulation of Section 2.1.1. Table 5.1 reports the statistics together with the cell complex each dataset trains on at its selected threshold τ.

Table 5.1: Dataset statistics and the cell complex at the selected support threshold τ. Density is $|\mathcal{E}|/(M \cdot N)$ over training interactions. train/item is the average number of training interactions per item. Faces and edges are the 2-cells and induced item–item 1-cells of the complex used by the RQ1–RQ3 experiments.

dataset	users	items	train	test	density	train/item	τ	faces	edges
CiaoDVD	16,043	12,787	34,352	11,244	0.017%	2.7	5	46	96
Epinions	37,952	119,015	386,406	101,333	0.009%	3.2	3	95,436	47,053
Beidian	3,773	4,544	31,920	3,660	0.186%	7.0	4	12,773	4,701
Foursquare-NYC	1,083	38,333	72,816	18,208	0.175%	1.9	2	169,625	82,327
Foursquare-TKY	2,293	61,858	169,569	42,386	0.120%	2.7	55	535	307
Gowalla	29,858	40,981	810,128	217,242	0.066%	19.8	60	416	330

Five datasets are sparse, with an item receiving on average at most seven interactions. This is the setting where the linear autoencoders of Section 3.2 found higher-order signal most valuable, with gains concentrated among users with low-to-moderate activity. Gowalla is the contrasting regime, with an item receiving on average about twenty interactions. The number of faces spans four orders of magnitude across the selected operating points, from 46 on CiaoDVD to 1.7×10^5 on Foursquare-NYC. This range already shows the control that τ provides, which RQ4 studies.

Validation split. Each dataset comes with a train/test split, which is not altered. For model selection, a validation set is carved out of the training data by leave-one-out [20]. For every user with at least two training interactions, one randomly chosen training interaction is moved to the validation set. Beidian is the only dataset that provides an official validation split, which is used instead. The complex and all

propagation operators are built from the remaining training interactions only, so no validation or test information reaches the model structure.

Epoch selection. The released implementations of NGCF and LightGCN early-stop and select epochs on the *test* set. The papers mention a validation split, but the official code and benchmark data provide none, and training uses early stopping on test Recall@20. Dacrema et al. [38] and Sun et al. [39] identify this practice as a source of optimistic bias. All experiments in this thesis instead early-stop and select on the validation split. Training runs for at most 600 epochs and is evaluated every 10. It stops after 40 epochs without improvement in validation Recall@20. Every model is then reported at the epoch where its validation Recall@20 peaked, with all reported metrics computed on the official test split at that single epoch. The validation split serves selection only. The test set never influences training, structure building, the selection of the face-support threshold τ, or epoch choice.

Metrics. Evaluation is full-catalogue top-K ranking. For each user, the model ranks all items absent from the user's training data. The user's items in the official test split form the relevance set R_u. Three metric families are reported at cutoffs $K \in \{3, 5, 10, 20, 50\}$. In the following, $\text{rank}(r)$ denotes the position of held-out item r in the user's ranking.

$$\text{Recall@}K(u) = \frac{\left|\{r \in R_u : \text{rank}(r) \le K\}\right|}{|R_u|}, \tag{5.1}$$

$$\text{NDCG@}K(u) = \frac{\sum_{r \in R_u,\, \text{rank}(r) \le K} 1/\log_2\big(\text{rank}(r)+1\big)}{\sum_{p=1}^{\min(K,|R_u|)} 1/\log_2(p+1)}, \tag{5.2}$$

$$\text{ARHR@}K(u) = \frac{1}{|R_u|} \sum_{r \in R_u,\, \text{rank}(r) \le K} \frac{1}{\text{rank}(r)}. \tag{5.3}$$

Recall@K [8] measures how many of the held-out items are retrieved. NDCG@K [40] discounts hits logarithmically by position. ARHR@K [41] weights each hit by the reciprocal of its rank, placing the strongest emphasis on the very top of the list. It is the metric RecWalk [42] reports for the same reason. All metrics are computed per user and averaged over the users with a non-empty relevance set. The per-user values are also the unit of observation for the significance tests described below. The main text reports $K = 5$, and the appendix the remaining cutoffs.

Significance testing. Every comparison between two models is tested with a two-sided paired t-test on the per-user metric values, pairing the two models user by user on the identical evaluation set [43]. A difference is called significant at $p < 0.05$, marked with a star (*) in tables and rendered as a filled marker in figures. The design tests whether the average user's metric changed, and pairing removes the variance between users from the comparison. Unless stated otherwise, stars always refer to the comparison named in the surrounding table or caption.

5.1.2. Models and Configuration

Three models are compared throughout, differing in exactly one component at a time:

- **LightGCN** [8]: the backbone, trained and selected under the same protocol. It is our main graph-based collaborative filtering baseline (Section 3.1) and the model LightCCN reduces to when the higher-order channels are removed.
- **LightCCN-prop**: higher-order propagation only (Section 4.5), no readout.
- **LightCCN-full**: LightCCN-prop extended with the higher-order user readout (Section 4.6).

This controlled ladder attributes every accuracy difference to a specific mechanism. Comparing LightCCN-prop with LightGCN isolates propagation, and setting LightCCN-full against LightCCN-prop isolates the readout. Measuring LightCCN-full against LightGCN gives the full model (RQ1). All three share the embedding size, initialisation, loss, and selection protocol, so the comparison is architectural.

In addition to the controlled ladder, four further baselines position LightCCN in the wider landscape:

- **MF** [1, 4]: matrix factorisation trained with the same BPR objective, scoring by inner products with no propagation. It shows what the shared embedding capacity achieves without propagation.
- **NGCF** [6]: the graph-convolutional predecessor of LightGCN (Section 3.1), which keeps feature transformations and non-linear activations inside propagation.
- **SimGCL** [24]: a contrastive extension of the LightGCN backbone that perturbs embeddings with uniform noise and adds an InfoNCE objective to the BPR loss. It represents the contrastive class of graph collaborative filtering.
- **HCCF** [11]: hypergraph contrastive collaborative filtering, which learns one hypergraph over the users and one over the items as a global view and contrasts it with the local view of the interaction graph. It represents the hypergraph class of higher-order collaborative filtering (Section 3.3).

Hyperparameters. Table 5.2 lists the settings, held identical across models and datasets. Their embedding size, learning rate, regularisation, and depth are the values the LightGCN paper reports as best [8], so LightGCN runs at its own reported optimum. The propagation depth is $L = 3$ for the main results, with $L = 1$ to 6 studied in RQ3. LightCCN inherits these settings unchanged and tunes only τ, its only structural hyperparameter. Any difference from the backbone is attributable to the structure and not to a tuning advantage. The threshold is selected empirically, per dataset, on validation data, as the τ with the best validation ARHR@10. Ties within 0.1% are resolved by validation Recall@20. The selected values are those of Table 5.1.

Table 5.2: Training hyperparameters, identical for MF, LightGCN, and the LightCCN variants across all datasets. NGCF, SimGCL, and HCCF deviate only in the settings they tune per dataset.

setting	value
embedding dimension d	64
propagation layers L (main results)	3
optimiser	Adam [44], learning rate 10^{-3}
ℓ_2 regularisation λ	10^{-4}
BPR negative sampling	1 uniform negative per positive
mini-batch size	2048
maximum epochs / evaluation cadence	600 / every 10
early stopping	40 epochs without validation Recall@20 gain
reported epoch	best validation Recall@20
random seed (initialisation, sampling, validation split)	2020

Baseline tuning. MF runs in the same fixed setup as LightGCN. NGCF, SimGCL, and HCCF are tuned per dataset within the search ranges of their own papers. NGCF is tuned over the learning rate and the ℓ_2 coefficient. SimGCL is tuned over the contrastive weight, with its noise magnitude and temperature fixed at the paper's values. HCCF is tuned over the contrastive weight, the temperature, the number of hyperedges, and the dropout rate. The tuned configuration is selected per dataset on validation Recall@20, the same rule used for epoch selection.

Implementation and hardware. All models are implemented in Python with PyTorch [45], sharing the training loop, evaluator, and data pipeline.[1] Experiments ran on NVIDIA A100 and L4 GPUs. A full training run takes minutes on the smaller datasets and up to a few hours on Gowalla. Every reported number is derived from the files each run writes, namely the per-epoch metrics and the per-user rank lists. This makes the pipeline from raw data to every table and figure reproducible end-to-end.

5.2. Results

5.2.1. RQ1: Does Modelling Higher-Order Relations Improve Recommendation Accuracy?

Table 5.3 reports the three featured metrics at $K = 5$ for LightGCN and LightCCN-full on all six datasets, and Table B.1 in the appendix expands the comparison to every cutoff. Figure 5.1 plots NDCG@K across

[1] The implementation, the experiment scripts, and the results are available in our GitHub repository.

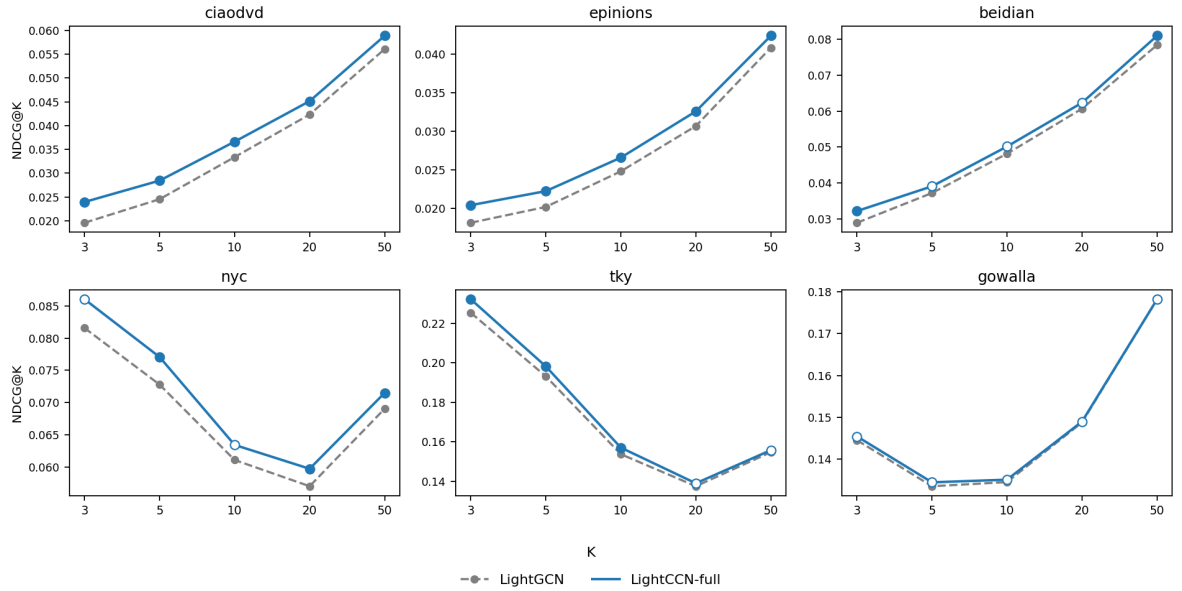

Figure 5.1: NDCG@K for $K = 3$ to 50, per dataset. Gray dashed: LightGCN; blue: LightCCN-full. Filled markers indicate cutoffs at which LightCCN-full is significantly better under the per-user paired t-test ($p < 0.05$); red markers would indicate significantly worse. Recall@K and ARHR@K versions are in the appendix (Figures B.1 and B.2).

cutoffs.

Table 5.3: RQ1 results at $K = 5$. Each dataset block shows the raw LightGCN and LightCCN-full values and the relative change. Stars mark per-user paired t-test significance of the change at $p < 0.05$. The full cutoff grid is in Table B.1, and the $K = 3$ and $K = 10$ versions in Tables B.2 and B.3.

dataset	model	NDCG@5	Recall@5	ARHR@5
CiaoDVD	LightGCN	0.0245	0.0370	0.0179
	LightCCN-full	0.0284*	0.0402	0.0216*
	Δ%	**+15.9%***	**+8.6%**	**+20.8%***
Epinions	LightGCN	0.0202	0.0217	0.0114
	LightCCN-full	0.0223*	0.0236*	0.0125*
	Δ%	**+10.1%***	**+8.5%***	**+9.8%***
Beidian	LightGCN	0.0372	0.0593	0.0300
	LightCCN-full	0.0390	0.0607	0.0319
	Δ%	**+5.0%**	**+2.3%**	**+6.5%**
Foursquare-NYC	LightGCN	0.0728	0.0213	0.0117
	LightCCN-full	0.0771*	0.0224	0.0125
	Δ%	**+5.9%***	**+5.0%**	**+6.6%**
Foursquare-TKY	LightGCN	0.1932	0.0507	0.0295
	LightCCN-full	0.1982*	0.0525*	0.0305*
	Δ%	**+2.6%***	**+3.7%***	**+3.3%***
Gowalla	LightGCN	0.1335	0.0858	0.0514
	LightCCN-full	0.1344	0.0861	0.0516
	Δ%	**+0.7%**	**+0.2%**	**+0.5%**

Answer. Yes. At $K = 5$, LightCCN-full is significantly better than LightGCN on four of the six datasets (CiaoDVD, Epinions, and Foursquare-TKY, with Foursquare-NYC significant on NDCG). Beidian becomes significant at $K = 3$ (Table B.2), and Gowalla is statistically tied throughout. Over the full grid of Table B.1, 6 datasets × 3 metrics × 5 cutoffs, the model is significantly better in 47 of 90 measurements,

statistically tied everywhere else, and never significantly worse. All 54 measured changes at cutoffs $K \leq 10$ are positive. At these cutoffs LightCCN-full is ahead of LightGCN on every dataset and every metric, whether or not the difference is significant. The higher-order signal acts as a re-ranking of items just below the cutoff rather than as retrieval of items from deep in the ranking (Table 5.4).

Attribution. Table 5.4 shows where the gains reside when the ranking of LightCCN-full is compared with that of LightGCN. Only 6% to 24% of the test items that LightCCN-full newly places in the top 5 were ranked beyond 20 by LightGCN. The rest were at ranks 6 to 20, with ranks 6 to 10 being the largest source on every dataset (46% to 74%). The gain at small K is thus a re-ranking of items just below the cutoff. Among the items both models place in the top 5, more move up than down on every dataset (17% to 36% up against 13% to 28% down), so their ARHR increases everywhere. The number of test items ranked first rises on every dataset as well, by 1% to 40%. The number of top-5 hits changes for 2% to 21% of the users, and on every dataset more of these users gain than lose, by 6% to 50%.

Table 5.4: Decomposition of the gains of LightCCN-full over LightGCN at $K = 5$, all values in percent. *Overall*: *net hits* is the change in the number of top-5 hits relative to LightGCN, *rank 1* the change in the number of test items ranked first, *users changed* the share of users whose number of top-5 hits differs between the two models, and *gain vs lose* how many more of these users gain than lose. *New hits by rank* splits the test items that LightCCN-full newly places in the top 5 by their rank under LightGCN. *Shared hits* are the items both models place in the top 5, *up* and *down* are the shares of them that LightCCN-full ranks higher or lower, and *ARHR* is the change in their mean reciprocal rank.

	overall				new hits by rank			shared hits		
dataset	net hits	rank 1	users changed	gain vs lose	6–10	11–20	>20	up	down	ARHR
CiaoDVD	9.9	39.5	3.3	34.4	55	32	13	36	23	10.9
Epinions	10.9	9.0	4.4	34.6	46	30	24	31	28	2.3
Beidian	2.3	5.4	1.7	17.9	67	21	12	28	21	3.1
Foursquare-NYC	6.5	4.7	8.8	50.0	74	20	6	21	15	2.6
Foursquare-TKY	2.5	1.9	21.3	22.8	70	22	8	17	13	1.6
Gowalla	0.7	1.2	12.5	5.5	70	22	8	20	19	0.6

Magnitude of the gains. Significant improvements range from +2.2% (Foursquare-TKY, NDCG@10) to +25.7% (CiaoDVD, ARHR@3). The position-weighted families carry most of the significant measurements, ARHR 19 and NDCG 18 of 30 each against Recall's 10, and the largest gains occur on CiaoDVD, the dataset with the most selective complex (46 faces, Table 5.1). Table B.1 and Figure 5.1 show a consistent shrinkage as K grows. Epinions ARHR falls from $+12.2^*$ at $K = 3$ to $+6.7^*$ at $K = 50$, and Foursquare-TKY Recall from $+4.6^*$ to -0.2 (not significant). CiaoDVD Recall is significant only at $K = 3$, while its ARHR stays significant at every cutoff.

Exceptions. Beidian is significant at $K = 3$ on all three metrics ($+10.6^*$ ARHR, $+11.3^*$ NDCG, $+12.7^*$ Recall) and, with the single exception of NDCG@50 ($+3.3^*$), not at larger cutoffs. Its gains appear on very short lists. Foursquare-NYC shows the opposite pattern, with significance arriving only at $K \geq 10$ on ARHR. With about 1.1k evaluated users, its small-cutoff comparisons are underpowered. Their values (+5 to +7%) match the significant deeper cells, so this indicates a power limit of the test rather than an absence of effect. Gowalla is statistical parity in every measurement. This is the expected outcome rather than a failure. Higher-order structure hurts on this dataset (Section 5.2.4), so its τ was selected to admit almost none of it. Section 5.2.2 confirms that the mechanism is correspondingly inactive there.

Comparison with baselines. Table 5.5 places LightCCN-full next to the baselines of Section 5.1.2. LightCCN-full is above LightGCN in all 18 columns, consistent with the paired comparisons above. The 18 best values split evenly between LightCCN-full and SimGCL. LightCCN-full has the best value in all nine columns on Foursquare-NYC, Foursquare-TKY, and Gowalla, and SimGCL in the nine on CiaoDVD, Epinions, and Beidian, with margins over LightCCN-full between roughly 4 and 21 percent. Where LightCCN-full leads, SimGCL is within about 3 percent on Foursquare-TKY and Gowalla, while on Foursquare-NYC it is 11 to 19 percent below and also below LightGCN. MF and NGCF are below both in every column. On CiaoDVD, Epinions, and Beidian the contrastive objective of SimGCL thus adds more than the higher-order structure does. The two additions are independent, since SimGCL keeps LightGCN's propagation, and their combination is left for future work. HCCF, the hypergraph

baseline, lies between MF and LightGCN in all 18 columns. As a reference point, Table B.8 lists the LightGCN values reported in its paper next to ours on Gowalla, the only dataset shared with the paper, and on Yelp2018, which was run for this comparison only. The paper applies early stopping on the test set, whereas this thesis applies it on the validation split.

Table 5.5: Overall performance at $K = 5$ on the official test split at the validation-selected epoch, values in percent. R, N, and A denote Recall@5, NDCG@5, and ARHR@5. The best value per column is in green, and the LightCCN-full row is in bold.

	CiaoDVD			Epinions			Beidian			Foursquare NYC			Foursquare TKY			Gowalla		
method	R	N	A	R	N	A	R	N	A	R	N	A	R	N	A	R	N	A
MF	1.76	1.27	0.94	0.92	0.86	0.48	3.33	2.26	1.90	1.02	3.31	0.55	4.01	15.06	2.30	5.58	8.05	3.11
NGCF	3.04	2.15	1.58	1.84	1.68	0.94	4.84	3.06	2.48	1.70	5.65	0.91	4.55	17.04	2.62	6.78	10.41	3.96
SimGCL	4.19	3.05	2.34	2.82	2.60	1.51	6.48	4.29	3.57	1.82	6.53	1.11	5.23	19.56	2.97	8.56	13.23	5.04
HCCF	2.85	1.90	1.36	1.95	1.85	1.03	4.40	2.76	2.23	1.04	3.58	0.55	4.66	17.55	2.74	7.72	11.90	4.56
LightGCN	3.70	2.45	1.79	2.17	2.02	1.14	5.93	3.72	3.00	2.13	7.28	1.17	5.07	19.32	2.95	8.58	13.35	5.14
LightCCN-full	**4.02**	**2.84**	**2.16**	**2.36**	**2.23**	**1.25**	**6.07**	**3.90**	**3.19**	**2.24**	**7.71**	**1.25**	**5.25**	**19.82**	**3.05**	**8.61**	**13.44**	**5.16**

5.2.2. RQ2: Is the Contribution of Higher-Order Structure Attributable to Propagation or to the Readout?

RQ2 decomposes the RQ1 gains with the controlled ladder of Section 5.1.2. The prop column of Table 5.6 compares LightCCN-prop with LightGCN (propagation alone), and the readout column compares LightCCN-full with LightCCN-prop (the readout's own contribution). Figure B.3 in the appendix shows the three variants side by side.

Table 5.6: RQ2 decomposition at $K = 5$, as relative changes in percent: full = LightCCN-full vs LightGCN, prop = LightCCN-prop vs LightGCN, readout = LightCCN-full vs LightCCN-prop. Each star is the per-user paired t-test of exactly that comparison at $p < 0.05$. The full grid at $K = 5$ and $K = 10$ is in Table B.4.

	NDCG@5 Δ%			Recall@5 Δ%		
dataset	full	prop	readout	full	prop	readout
CiaoDVD	+15.9*	+10.9*	+4.5	+8.6	+5.1	+3.3
Epinions	+10.1*	+2.0	+7.9*	+8.5*	+2.5	+5.9*
Beidian	+5.0	-3.6	+9.0*	+2.3	-6.5	+9.4*
Foursquare-NYC	+5.9*	+0.5	+5.3*	+5.0	+2.8	+2.2
Foursquare-TKY	+2.6*	-0.3	+2.9*	+3.7*	+0.7	+2.9*
Gowalla	+0.7	+0.1	+0.6	+0.2	-0.2	+0.4

Answer. The readout is the active ingredient. On five of the six datasets, LightCCN-prop is statistically indistinguishable from LightGCN, and adding the readout produces the significant improvement. The readout column is significant on Beidian, Epinions, Foursquare-NYC, and Foursquare-TKY, while the corresponding prop comparisons are not (Table 5.6). CiaoDVD is the exception. Its 46 faces of exceptionally high support already act through propagation alone (+10.9* NDCG@5 without the readout), and the readout's addition on top is positive but not significant. Gowalla shows the mechanism inactive. With almost no faces to pool, all three comparisons are non-significant. Notably, Beidian's readout effect is significant at $K = 5$ even though its full-vs-LightGCN comparisons reach significance only at $K = 3$. LightCCN-prop and LightCCN-full differ only in the readout, so the per-user differences between these two models vary less than differences against LightGCN. The paired test then detects smaller effects. Three further design ablations support the mechanism and the model's design choices. Redirecting the readout away from the user tower hurts, each of the three alternative backbones (edges without faces, recomputed edges, and recomputed edges and faces) is significantly worse on exactly one dataset, and adding user–item edges to the complex hurts everywhere (Appendices C to E).

The learned weights explain the split. Table 5.7 reports the softmax channel shares and readout gates at the reported epoch, and Figures B.4 and B.5 in the appendix show their trajectories over training.

Three observations follow. First, the node-level entry of the structural signal remains near zero. The share w_2 of the edge channel in node updates is at most 6.6% across all six datasets, so node-level propagation is dominated by the user–item channel and propagation alone adds little higher-order signal to the node embeddings, which is what the prop column of Table 5.6 measures. Second, structure is absorbed one level up. The face share w_5 of the edge update lies between 17% and 58%, and the trajectory of w_5 falls early and then grows to a plateau on every dataset, so training actively raises the face share at the edge level while closing the node-level entry. Third, the readout gates open from zero on every dataset, to β between 0.76 and 7.32 and γ between 0.74 and 6.33 at the reported epoch, and keep growing beyond it (Figure B.5). The higher-order signal thus reaches users through the readout and not through propagation. CiaoDVD is the one exception, where propagation carries it. The learned weights are direct evidence for this answer, since training itself closes the propagation path into the node embeddings and opens the readout path.

Table 5.7: Learned channel shares and readout gates at the reported epoch. w_2 is the softmax share of the edge channel in node updates and w_5 the share of the face channel in edge updates (Equations 4.7 and 4.6); β and γ are the readout gates of Equation 4.10. The full seven-weight table is in the appendix (Table B.5).

dataset	edges→nodes w_2	faces→edges w_5	readout β	readout γ
CiaoDVD	6.62%	25.4%	0.76	0.74
Epinions	0.02%	38.2%	5.38	4.40
Beidian	3.26%	23.5%	0.91	0.85
Foursquare-NYC	3.72%	17.2%	1.56	1.49
Foursquare-TKY	0.00%	57.8%	4.82	4.72
Gowalla	0.00%	36.8%	7.32	6.33

Face consistency. To examine the mechanism further, Table 5.8 tests whether the face channel acts by pulling the embeddings of a face's items together. It reports the mean pairwise cosine similarity within each face's three items, within triples sampled from the items each user co-consumed, and within random item triples. In both models the items of a face are far more similar than random triples and more similar than a user's own triples. LightCCN-full does not tighten the faces further. Its within-face cosine is 5% to 22% below LightGCN's on every dataset. The gain is thus not produced by making the items of a face more similar. A possible reading connects this to the linear autoencoders of Section 3.2, whose learned higher-order weights are predominantly negative. In a similar vein, the higher-order signal here may act as a correction that could make the embeddings of jointly consumed items richer and less similar, rather than smoother.

Table 5.8: Mean pairwise cosine similarity within the three items of each face, within triples sampled from the items each user co-consumed (ten per user, faces excluded), and within the same number of random item triples, for both models at the selected τ. The percentage in parentheses is the change of the within-face similarity relative to LightGCN, and the star marks its significance under a two-sided paired t-test over the faces at $p < 0.001$.

	faces		user triples		random triples	
dataset	LightGCN	LightCCN	LightGCN	LightCCN	LightGCN	LightCCN
CiaoDVD	0.93	0.83* (−10%)	0.70	0.63	0.01	0.02
Epinions	0.70	0.54* (−22%)	0.52	0.46	0.02	0.03
Beidian	0.78	0.73* (−5%)	0.52	0.50	0.01	0.00
Foursquare-NYC	0.71	0.62* (−13%)	0.63	0.57	0.02	0.00
Foursquare-TKY	0.92	0.82* (−11%)	0.55	0.49	0.01	0.00
Gowalla	0.88	0.70* (−20%)	0.48	0.48	0.00	0.00

5.2.3. RQ3: How Does Propagation Depth Affect the Accuracy of a Model with Higher-Order Structure?

RQ3 trains both models at every depth $L \in \{1, \ldots, 6\}$ and compares them at equal depth. Figure 5.2 plots NDCG@5 against depth, and Figure 5.3 reports the over-smoothing analysis. The per-depth values are in the appendix (Tables B.6 and B.7).

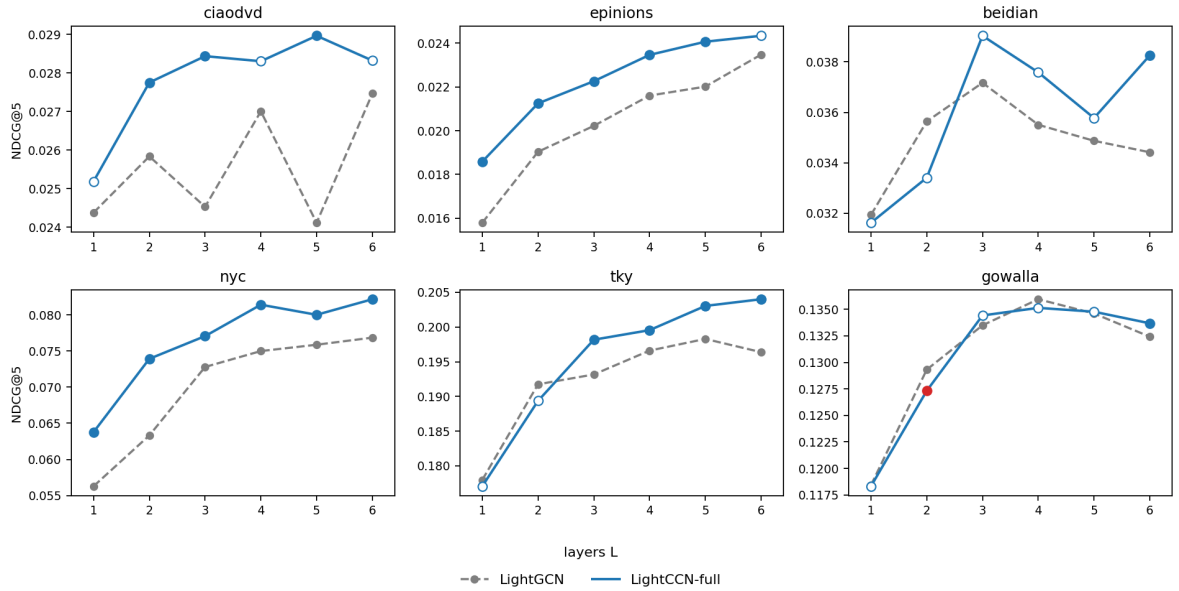

Figure 5.2: NDCG@5 against propagation depth L, per dataset. Gray dashed: LightGCN; blue: LightCCN-full. Filled markers: LightCCN-full significantly better than LightGCN at the same depth ($p < 0.05$); red markers: significantly worse. The ARHR@5 version is in the appendix (Figure B.7).

Answer. More layers do not hurt. On five of the six datasets there is no significant regression at any depth on any metric (Figure 5.2 and Table B.6). Depth can amplify the advantage. Epinions is significantly ahead at $L = 1$ to 5, Foursquare-NYC at every depth, and Foursquare-TKY grows from $+2.6^*$ NDCG@5 at $L = 3$ to $+3.9^*$ at $L = 6$. On Gowalla, the dataset whose τ removes almost all structure, the two models are statistically tied at every depth except two. The grid's only significant loss appears at $L = 2$ (-1.5^* NDCG@5), and a small significant gain at $L = 6$ ($+1.0^*$ NDCG@5), where the two models' smoothness converges in Figure 5.3. The advantage is present at every depth where the complex admits structure and absent on Gowalla, where it admits almost none. Smoothness does not explain it, as the over-smoothing analysis below shows.

Over-smoothing. A standard concern with deeper propagation is over-smoothing, the collapse of embeddings toward indistinguishable vectors as depth grows [46]. To measure it, every run logs the smoothness measure the LightGCN paper uses [8]: for users,

$$S_U = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} c_{v \to u} \left\| \frac{\mathbf{e}_u}{\|\mathbf{e}_u\|} - \frac{\mathbf{e}_v}{\|\mathbf{e}_v\|} \right\|_2^2, \qquad c_{v \to u} = \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_v|}} \sum_{i \in \mathcal{N}_u \cap \mathcal{N}_v} \frac{1}{|\mathcal{N}_i|}, \tag{5.4}$$

with the item-side S_I defined symmetrically. Lower values mean smoother embeddings. The sum runs over ordered pairs, so absolute values are twice the unordered-pair convention. All comparisons use the same convention on both models, computed at the reported epoch on the training graph. Figure 5.3 shows that both models end $L = 6$ smoother than they start at $L = 1$, on both the user and the item side. The two models also remain in the same smoothness regime throughout. LightCCN is the smoother model on most datasets, and the relative gap stays below 17% when compared with LightGCN. The exception is CiaoDVD. The LightGCN baseline there is the noisiest across depths, and LightCCN is up to 32% less smooth at $L = 5$ while remaining the more accurate model. There is no depth at which LightCCN's embeddings collapse faster than the baseline's, so the over-smoothing objection to adding higher-order structure at depth is not supported. At the same time, smoothness does not explain the gains. If smoother embeddings caused the accuracy improvements, LightCCN should win exactly where it is the smoother model and lose where it is not, but the two do not align. CiaoDVD is the clearest counterexample. LightCCN is less smooth than LightGCN there yet more accurate.

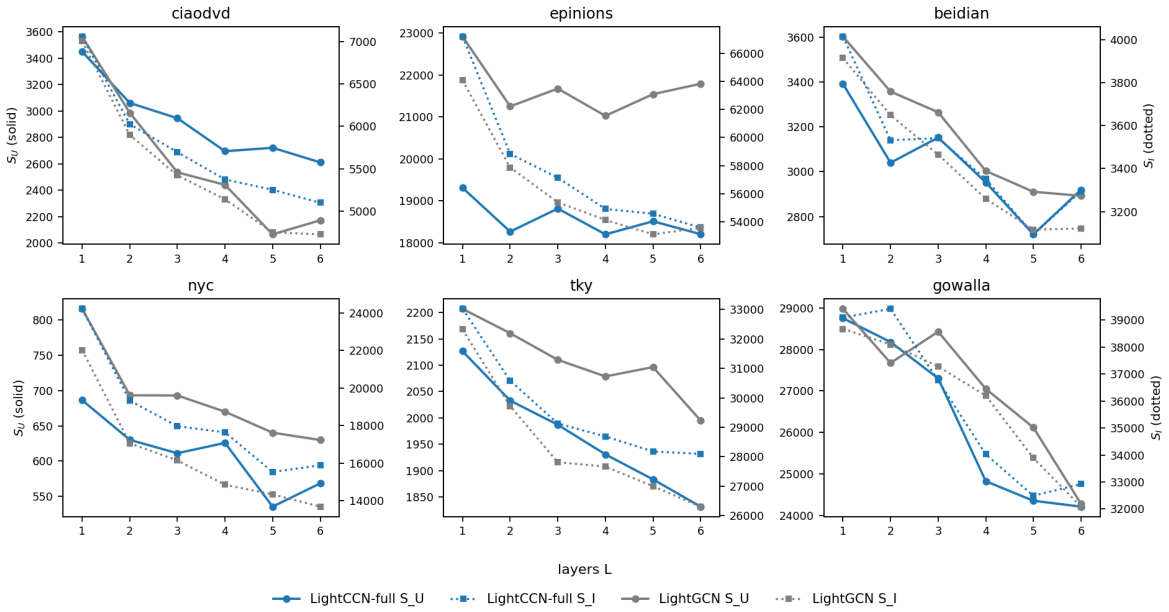

Figure 5.3: Smoothness against depth per dataset, computed with Equation 5.4 at the reported epoch (lower = smoother). Solid curves show S_U on the left axis and dotted curves S_I on the right axis, with LightCCN-full in blue and LightGCN in gray.

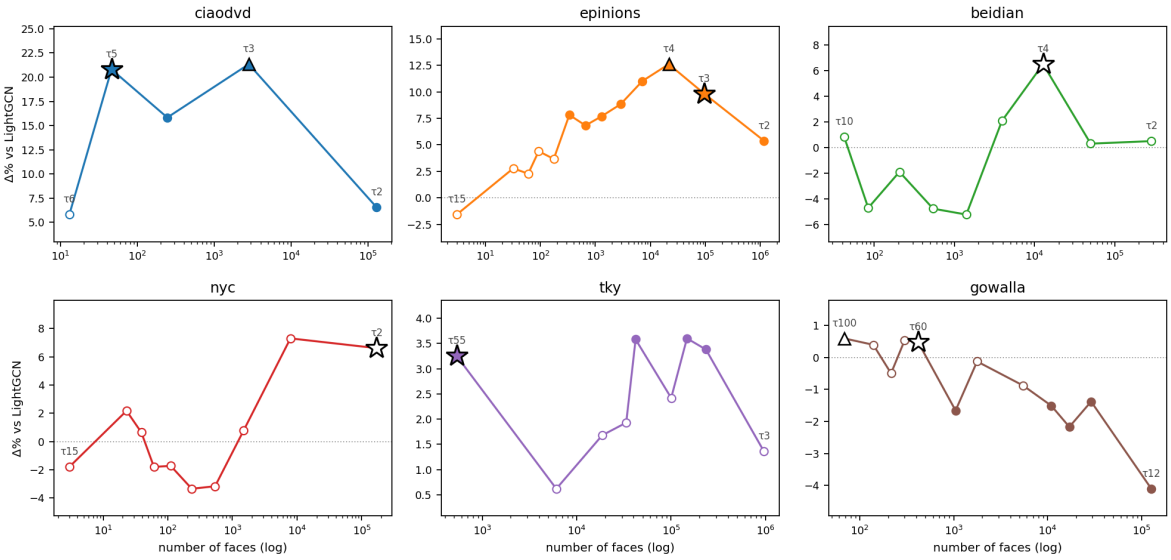

Figure 5.4: ARHR@5 change over LightGCN, in percent, against the number of faces (log scale), per dataset. Star: the validation-selected τ; triangle: the τ with the best net significance count where it differs from the star; circles: other τ values. Filled markers: significant vs LightGCN under the per-user paired t-test ($p < 0.05$); empty: not significant. NDCG@5 and Recall@5 versions are in the appendix (Figures B.8 and B.9).

5.2.4. RQ4: How Does the Amount of Higher-Order Structure Affect Recommendation Accuracy?

RQ4 varies the support threshold τ on every dataset, from complexes with abundant faces to complexes with almost none, training one model per value of τ. Raising τ can only remove faces (Section 4.3), so the sweep produces a nested family of complexes. Across the swept values, the number of faces moves across three to six orders of magnitude per dataset. Figure 5.4 plots the resulting sweep curves. Each panel shows the change in ARHR@5 over LightGCN against the number of faces.

Answer. The amount of higher-order structure is directly controllable through τ, and its impact is strong but dataset-shaped. The panels of Figure 5.4 show four qualitatively different response shapes. CiaoDVD, Beidian, and Epinions have interior optima, where an intermediate amount of structure beats both a large and a small complex. CiaoDVD peaks on a wide plateau between 46 and 2.8k faces (up to +21.3* ARHR@5) and collapses to parity with LightGCN when fewer faces remain. Epinions stays significant from over a million faces down to a few hundred, a range of more than three orders of magnitude, and collapses once fewer than roughly 200 faces remain. Foursquare-NYC behaves as an abundance threshold, with gains appearing only at its largest complexes. Foursquare-TKY is a precision tail. Its 535 faces, each supported by at least 55 co-visitors, carry the entire effect, significant on 18 of 25 metric measurements, while much larger complexes achieve no more. Gowalla shows monotone harm that τ eliminates when increased. Losses are significant at the abundant end and deepen to -4.1^* ARHR@5 (significant on 25 of 25 measurements) at its largest complex of 127k faces. They shrink monotonically as faces are removed and reach exact parity when almost no faces remain.

Selectivity against abundance. On five of the six datasets the best operating point is not at the abundant end of the axis. Gowalla peaks at complexes of a few hundred faces or fewer, CiaoDVD on its plateau between 46 and 2.8k faces, Foursquare-TKY's 535-face complex matches its much larger mid-range optimum, and Epinions and Beidian peak at intermediate sizes. Foursquare-NYC is the one dataset that needs abundance. Given these different response curves, no fixed τ suits every dataset, so the threshold is selected per dataset on validation (Section 5.1.2). Nothing in the summary statistics of a dataset predicts which end of the axis it prefers, since Foursquare-NYC and CiaoDVD are equally sparse yet want opposite ends. At the selected operating points, 30 of the 54 measurements at $K \leq 10$, counted across all six datasets and the three metric families, are significantly positive, and none is significantly negative anywhere in the $K = 3$ to 50 grid. This matches the RQ1 result. The NDCG@5 and Recall@5 versions (Figures B.8 and B.9) show the same shapes and the same best regions, with only the exact peak position shifting with the metric on Foursquare-NYC and Foursquare-TKY.

Admitting and removing structure. On the gain datasets, selecting τ admits the amount of structure that helps. On Gowalla, it removes nearly all structure and with it the harm, at no cost relative to LightGCN. The threshold acts as a safety mechanism as much as a tuning parameter. A dataset on which higher-order structure is counterproductive is handled by the same selection procedure that finds the helpful amount elsewhere, without changing the model.

6

Discussion

This chapter interprets the results as a whole. Section 6.1 synthesises the findings across the four research questions, Section 6.2 relates them to the literature of Chapter 3, Section 6.3 states the limitations of the study, and Section 6.4 reflects on implications and stakeholders.

6.1. Synthesis of Findings

Higher-order structure improves collaborative filtering when two conditions are met, and the experiments identify both. First, the admitted amount of structure must fit the dataset. The sweep curves of RQ4 take four shapes. CiaoDVD, Beidian, and Epinions have interior optima, Foursquare-NYC an abundance threshold, Foursquare-TKY a precision tail, and Gowalla monotone harm. On the five datasets other than Foursquare-NYC the best operating point lies away from the abundant end, and Gowalla performs best when almost no faces are admitted. For this dataset, abundant faces cause significant losses, and the model returns to parity with LightGCN once the threshold removes nearly all of them (RQ4). Second, the structure must reach the user representations. Propagation alone injects the higher-order signal directly only into the item embeddings, and none of it reaches the user representations. The learned weights show the model keeping the node-level entry nearly closed ($w_2 \leq 6.6\%$) while the readout gates open from zero on every dataset (RQ2). The readout, a parameter-free pooling of the edge and face embeddings of a user's items into the user representation, gated by two learned scalars, turns the higher-order signal into ranking gains on four of the five datasets that show gains, with CiaoDVD the exception, where propagation alone already delivers them.

Under these conditions, the gains concentrate at the top of the ranking. Position-weighted metrics improve more, and to deeper cutoffs, than set-inclusion metrics (RQ1), consistent with a mechanism that re-ranks lists rather than retrieving new items. Increasing the propagation depth does not diminish the advantage. There is no significant regression at any depth on the five gain datasets, and on Foursquare-TKY the gains grow with depth. Measured with LightGCN's own smoothness measure, LightCCN stays in the same smoothness regime as the baseline at every depth (RQ3). The model achieves this with nine scalar parameters beyond the backbone and a preprocessing step measured in seconds to minutes. The gains are obtained essentially without additional model capacity.

Notably, the propagation over the complex does not bring the embeddings of a face's items closer. Their within-face similarity is lower than under LightGCN on every dataset (RQ2). The higher-order signal may act as a correction rather than a smoothing, in line with the predominantly negative higher-order weights of the linear autoencoders of Section 3.2.

6.2. Relation to Prior Work

The findings sharpen several claims from Chapter 3. The linear-autoencoder thread [9, 29] established that higher-order item interactions carry signal that pairwise models miss. The present results confirm that this signal exists in a deep, ranking-oriented setting as well. On five of the six datasets, restricting the structure to well-supported groups beats abundance, and the signal acts through re-ranking. Steck

and Liang observed that higher-order terms act as corrections concentrated on specific user populations [9]. In LightCCN, the correction reaches each user through one interaction hop. Items that lie on faces of jointly consumed items receive a pooled higher-order summary, and each user aggregates the summaries of the items in their own history. The edge and face signal personalises the user embedding without additional per-user parameters.

The ablations of the hypergraph thread [10–12] toggle whole modules, so the contribution of the higher-order structure is not separated from the extra parameters and auxiliary objectives those modules bring. The controlled ladder of this thesis attributes the gain to a specific, minimal mechanism. It also shows that most of the added capacity, the propagation channels, is not where the value lies. Against the topological thread, the results answer the question left open by TSP and Sheaf4Rec [18, 19]. Higher-order cell representations can be learned end-to-end in collaborative filtering, on the bipartite data directly. The order hierarchy of the cell complex (faces to edges to items to users) is the path along which the useful signal travels. Finally, the τ sweep asks a question that none of these threads does. It questions not whether higher-order structure helps, but how much of it a given dataset benefits from.

6.3. Limitations

Structure is fixed and triple-based. Faces are item triples selected by support counting before training, and the structure itself receives no gradient. The model can weight the channels that read the structure but cannot revise the structure. A harmful face can only be removed by raising τ for the whole dataset, and a group that support counting misses can never enter the complex. Groups of four or more items appear only as overlapping triples rather than as cells of their own, so a coherent larger group and a set of accidentally overlapping triples look identical to the model.

Global mixing weights and gates. The seven mixing weights and the two readout gates are global scalars, so all cells of an order share one channel allocation and all users receive the same gate strength. The corrective signal can therefore not be stronger for some user populations than for others. The linear higher-order literature suggests that such population-dependent strength would help [9].

Binarised feedback. All interactions are binarised, so on the rating-valued datasets a low rating counts as the same positive evidence as a high one. Face support therefore measures co-interaction rather than co-preference, and a triple of items that many users interacted with but disliked is indistinguishable from a genuinely co-liked group.

Coverage of the higher-order signal. The pooled readout is structurally zero for items that lie on no face, and high-support faces form predominantly among popular items. The correction therefore reaches only users whose histories touch the covered items, while the rest of the catalogue is at best unaffected.

The complex must be kept up to date. For LightGCN, new interactions only update the interaction graph and its normalisation. LightCCN additionally depends on a structure derived from that graph. As interactions accumulate, triple supports change and faces appear and disappear. The complex, its operators, and possibly the selected τ then have to be recomputed. A deployed system would therefore carry a maintenance step that the backbone does not have.

Scope of data. Six public implicit-feedback datasets of small to medium scale. The limiting factor for scale is cost. A single Gowalla run already takes up to a few hours, so the largest standard benchmarks did not fit the time budget of this thesis. Rating-valued feedback, sequential feedback, and domains where item groups are given explicitly, such as playlists or baskets, are also out of scope, and each would require a different method to construct the cell complex.

6.4. Implications and Stakeholders

For enterprises, the practical content of the thesis is that a higher-order extension of a production-grade recommender can be obtained at negligible parameter cost, with one interpretable parameter, and with a built-in off switch. On data where group structure is uninformative, the selection procedure returns a

near-empty complex and the model degrades gracefully to its backbone. The sweep curves also function as a diagnostic, showing directly whether a catalogue's co-consumption structure is informative enough to be worth modelling.

For end users, the mechanism has a benefit and a risk. The readout re-embeds items that sit on reliable faces into the representations of the users who interacted with them. These items are disproportionately popular, so the accuracy gains partly come from strengthening well-evidenced, head-anchored patterns. This is the classic accuracy side of the accuracy–diversity tension [47]. The same mechanism that lifts precision at the top of the list could, if deployed without care, narrow exposure for niche items and reinforce popularity bias. A responsible deployment would monitor tail-item exposure alongside accuracy, and the interpretable gates offer a direct control. The gates β and γ can be capped or regularised to bound how much head-anchored correction the model applies.

For the research community, the most transferable contribution is methodological. It combines validation-based selection in a field where test-set selection remains common [38, 39], per-user significance testing on every claim, and ablations designed into the architecture. These practices cost little and would make comparisons across the higher-order recommendation literature considerably more informative.

7

Conclusion

This thesis investigated whether topological deep learning can improve collaborative filtering by modelling higher-order relations. It introduced LightCCN, which lifts the user–item interaction data to a cell complex of order 2 by turning frequently co-consumed item triples into faces. The model propagates embeddings at every order of the complex under LightGCN's transformation-free design and injects the pooled higher-order signal into user representations through a user-side readout.

7.1. Answers to the Research Questions

(RQ1) *Does modelling higher-order relations improve recommendation accuracy?*

Yes. LightCCN-full is significantly better on four of the six datasets at cutoff five, on Beidian at cutoff three, and statistically tied on Gowalla. Over the full grid of 90 dataset–metric–cutoff measurements it is significantly better in 47 and significantly worse in none (Section 5.2.1). The gains are largest at the top of the ranking and on position-weighted metrics, with the largest significant improvement of +25.7% in ARHR at cutoff three on CiaoDVD, and they decrease as the cutoff grows. This identifies the improvement as re-ranking of retrieved items rather than retrieval of new ones. The gain is a short-range re-ranking. Most of the newly hit items sat just below the cutoff under LightGCN and are promoted across it (Section 5.2.1).

(RQ2) *Is the contribution of higher-order structure attributable to propagation or to the readout?*

The readout. Propagation alone is statistically indistinguishable from LightGCN on five of the six datasets, and adding the readout produces the significant jump on four of the five gain datasets (Section 5.2.2). The learned weights explain why. Training keeps the node-level share of the higher-order channel below 6.6% everywhere, while folding face information into edge states and opening the readout gates from zero on every dataset. CiaoDVD, whose 46 faces have exceptionally high support, is the one dataset where propagation alone already carries the signal.

(RQ3) *How does propagation depth affect the accuracy of a model with higher-order structure?*

Increasing depth does not hurt accuracy, and it often helps. At depths one to six there is no significant regression against same-depth LightGCN on any of the five gain datasets, and the advantage is maintained at every depth, growing on Foursquare-TKY. Gowalla contributes the single significant loss, at depth two, and a significant gain at depth six (Section 5.2.3). On the smoothness measure of the LightGCN paper, both models get smoother with depth on both the user and the item side and remain in the same smoothness regime at every depth, with LightCCN up to 17% smoother. However, smoothness and accuracy do not move together. The gains appear whether or not LightCCN is the smoother model, and the growing smoothness never comes with an accuracy loss.

(RQ4) *How does the amount of higher-order structure affect recommendation accuracy?*

The amount is directly controllable through the face-support threshold, and its impact is strong but dataset-shaped (Section 5.2.4). The sweep curves show interior optima on CiaoDVD, Beidian, and Epinions, an abundance threshold on Foursquare-NYC, a precision tail of 535 highly supported faces carrying the whole effect on Foursquare-TKY, and monotone harm on Gowalla that raising the threshold eliminates at no cost. On five of the six datasets the best operating point lies away from the abundant end. Because the response curves differ across datasets, no fixed threshold suits all of them, and it is selected per dataset on validation. The selected operating points have 30 of 54 measurements at cutoffs up to ten significantly positive and none negative.

(Main RQ) *Can topological deep learning improve collaborative filtering by modelling higher-order relations?*

Yes, under the conditions identified by the sub-questions. Higher-order relations improve collaborative filtering when the admitted amount of structure fits the dataset and when the architecture gives it an explicit path to the user representations. Under those conditions the gains are significant, concentrated at the top of the ranking, and stable in depth. Topological deep learning is an effective framework for this. The cell complex's order hierarchy is not notation but the actual causal path of the useful signal, from faces through edges to items to users, and its algebraic structure is what makes the mechanism interpretable, from the learned weights to the readout gates.

7.2. Future Work

Four directions follow from the limitations of Section 6.3. The first direction is learning the structure. Replacing the fixed support threshold with a differentiable or iterative face-selection mechanism would let the model revise which groups it trusts. A related direction is larger groups. The design is general for cell complexes, but with faces built from item triples every tested complex is simplicial. Faces attached to cycles of more than three items would capture larger consumption groups natively. The second direction is richer parameterisation within the same design budget. Per-cell channel mixing, separate weights for users and items, and readout gates conditioned on user activity would test whether the corrective signal should be stronger for some populations, as the linear higher-order literature suggests [9]. The third direction is preference-aware construction. On rating-valued data the support of a triple could count co-liking instead of co-interaction alone, and the same idea extends to sequential data, with faces over temporally close consumption, and to domains with explicit groups, where the complex is given rather than counted. The fourth direction goes beyond accuracy. Measuring tail-item exposure and diversity under the readout, and constraining the gates accordingly, would connect the mechanism to the accuracy–diversity trade-off [47].

References

[1] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[2] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

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

[5] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[6] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174, 2019.

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

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[9] Harald Steck and Dawen Liang. Negative interactions for improved collaborative filtering: Don't go deeper, go higher. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 34–43, 2021.

[10] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2020–2029, 2020.

[11] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 70–79, 2022.

[12] Jingwen Wang, Yuguang Yan, Ruichu Cai, Michael K. Ng, and Zhifeng Hao. Learning high-order user-item relation via hyperedge for recommender system. *Neurocomputing*, 676:133004, 2026.

[13] Darnbi Sakong, Thanh Trung Huynh, and Jun Jo. Hypergraph diffusion for high-order recommender systems. *arXiv preprint arXiv:2501.16722*, 2025.

[14] Jian Wang, Jianrong Wang, Di Jin, and Xinglong Chang. Hypergraph collaborative filtering with adaptive augmentation of graph data for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2640–2651, 2025.

[15] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep learning: Going beyond graph data. *arXiv preprint arXiv:2206.00606*, 2023.

[16] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and Lehman go cellular: CW networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640, 2021.

[17] Maosheng Yang and Elvin Isufi. Convolutional learning on simplicial complexes. *arXiv preprint arXiv:2301.11163*, 2023.

[18] Yanbiao Ji, Yue Ding, Chang Liu, Yuxiang Lu, Xin Xin, and Hongtao Lu. Topology-aware popularity debiasing via simplicial complexes. *arXiv preprint arXiv:2411.13892*, 2024.

[19] Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. Sheaf4rec: Sheaf neural networks for graph-based recommender systems. *ACM Transactions on Recommender Systems*, 2024.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

[21] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

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

[23] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[24] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1294–1303, 2022.

[25] Chengen Liu, Rohan Money, Ting Gao, Mohammad Sabbaqi, Baltasar Beferull-Lozano, and Elvin Isufi. Topological Kalman filtering on cell complexes. *arXiv preprint arXiv:2605.15955*, 2026.

[26] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002.

[27] Xia Ning and George Karypis. SLIM: Sparse linear methods for top-N recommender systems. In *IEEE International Conference on Data Mining*, pages 497–506, 2011.

[28] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257, 2019.

[29] Evangelia Christakopoulou and George Karypis. HOSLIM: Higher-order sparse linear method for top-N recommender systems. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 38–49, 2014.

[30] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735, 2021.

[31] Yuqi Zhang, Jian Yu, Zhizhong Liu, Guiling Wang, Minh Nguyen, Quan Z. Sheng, and Nancy Wang. Improving graph collaborative filtering with network motifs. *Neural Computing and Applications*, 37:9413–9432, 2025.

[32] Alexander Möllers, Alexander Immer, Vincent Fortuin, and Elvin Isufi. Hodge-aware contrastive learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[33] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, 2022.

[34] Guibing Guo, Jie Zhang, Daniel Thalmann, and Neil Yorke-Smith. ETAF: An extended trust antecedents framework for trust prediction. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 540–547, 2014.

[35] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys)*, pages 17–24, 2007.

[36] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.

[37] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1082–1090, 2011.

[38] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*, pages 101–109, 2019.

[39] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. Are we evaluating rigorously? Benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys)*, pages 23–32, 2020.

[40] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[41] Mukund Deshpande and George Karypis. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[42] Athanasios N. Nikolakopoulos and George Karypis. RecWalk: Nearly uncoupled random walks for top-N recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 150–158, 2019.

[43] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962, 2009.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 8024–8035, 2019.

[46] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3538–3545, 2018.

[47] Elvin Isufi, Matteo Pocchiari, and Alan Hanjalic. Accuracy-diversity trade-off in recommender systems via graph convolutions. *Information Processing & Management*, 58(2):102459, 2021.

A

Formal Definitions for Cell Complexes

This appendix collects the formal definitions behind the matrix presentation of Chapter 2. This appendix is not required to follow the method. LightCCN uses only the incidence matrices, their products, and their normalised forms. The definitions are given for the reader who wants the rigorous grounding, and follow Hajij et al. [15] and Bodnar et al. [16].

A.1. Regular Cell Complexes

Definition A.1 (Regular cell complex [15])**.** *A regular cell complex is a topological space $\mathcal{X}$ together with a partition $\{x_\alpha\}_{\alpha \in P_\mathcal{X}}$ into disjoint subspaces called cells, where $P_\mathcal{X}$ is an index set. The partition satisfies the following three conditions:*

1. $\mathcal{X} = \bigcup_{\alpha \in P_\mathcal{X}} \text{int}(x_\alpha)$*, where* $\text{int}(x_\alpha)$ *denotes the interior of cell* x_α*.*
2. *For each $\alpha \in P_\mathcal{X}$, there exists a homeomorphism $\psi_\alpha : x_\alpha \to \mathbb{R}^{n_\alpha}$ for some $n_\alpha \in \mathbb{N}$, called the dimension of cell x_α.*
3. *For each cell x_α, its boundary ∂x_α is a union of finitely many cells of dimension strictly less than n_α.*

The first condition states that every point of $\mathcal{X}$ belongs to the interior of exactly one cell, so the cells cover the space without overlapping and without leaving gaps. The second condition assigns every cell a dimension by identifying it with a piece of standard shape. A 0-cell is a single point, a 1-cell is homeomorphic to an open line segment, and a 2-cell is homeomorphic to an open disk. The third condition, the regularity condition, enforces that higher-order cells attach only to cells of strictly lower dimension that are already present, which gives the complex its strict hierarchy.

Face poset. The regularity condition implies that the entire structure of the complex is captured by a finite combinatorial object. The index set $P_\mathcal{X}$ can be equipped with a partial order that records which cells lie on the boundaries of which other cells. Two cells α and β are ordered $\alpha \leq \beta$ when $x_\alpha \subseteq \overline{x_\beta}$, where $\overline{x}$ denotes the closure of cell x. This partial order is the face poset of $\mathcal{X}$ [15].

Inductive construction. A regular cell complex is constructed by hierarchical gluing [16], following the standard inductive attachment procedure of algebraic topology [26]. Starting from a set of 0-cells, 1-cells are attached by gluing the endpoints of closed line segments to existing 0-cells, producing a graph. 2-cells are then attached by gluing the boundary of a closed disk, a circle, to a simple cycle of 1-cells already present. Cells of higher order are attached analogously.

A.2. Boundary Relations and Neighbourhoods

Definition A.2 (Boundary relation [16])**.** *For two cells $\alpha, \beta \in P_\mathcal{X}$, the boundary relation $\alpha \prec \beta$ holds if and only if $\alpha < \beta$ in the face poset and no cell γ sits between them, that is, there is no γ with $\alpha < \gamma < \beta$.*

The relation $\alpha \prec \beta$ means that α lies on the boundary of β and is at the next-lower order. Hajij et al. call the same notion incidence [15].

Definition A.3 (Boundary and co-boundary adjacent cells [16])**.** *For a cell $\sigma \in P_X$, the boundary $\mathcal{B}(\sigma)$ is the set of all cells immediately below σ in order, and the co-boundary $\mathcal{C}(\sigma)$ is the set of all cells immediately above σ in order:*

$$\mathcal{B}(\sigma) = \{\tau \in P_X : \tau \prec \sigma\}, \qquad \mathcal{C}(\sigma) = \{\tau \in P_X : \sigma \prec \tau\}.$$

Definition A.4 (Lower and upper adjacency [16])**.** *Two k-cells σ and τ are lower adjacent when they share a common $(k-1)$-cell on their boundary, and upper adjacent when they both lie on the boundary of a common $(k+1)$-cell:*

$$\mathcal{N}_{\downarrow}(\sigma) = \{\tau : \exists \delta \text{ such that } \delta \prec \sigma \text{ and } \delta \prec \tau\},$$
$$\mathcal{N}_{\uparrow}(\sigma) = \{\tau : \exists \delta \text{ such that } \sigma \prec \delta \text{ and } \tau \prec \delta\}.$$

The cell δ is the bridge cell that connects σ and τ. In the terminology of Hajij et al., lower adjacency is called coadjacency and upper adjacency is called adjacency [15].

Relation to the incidence matrices. With the cells of each order enumerated, the boundary relation is encoded by the incidence matrices B_k of Section 2.3.2. Representing a k-cell σ by the indicator vector $\mathbf{1}_\sigma \in \mathbb{R}^{|X_k|}$, the product $B_k \mathbf{1}_\sigma$ marks the cells of $\mathcal{B}(\sigma)$ and $B_k^\top \mathbf{1}_\tau$ marks the cells of $\mathcal{C}(\tau)$. The off-diagonal support of $B_k^\top B_k$ is the lower adjacency relation and that of $B_{k+1} B_{k+1}^\top$ the upper adjacency relation, which is the form used in the main text.

A.3. The General Cell-Complex Message-Passing Scheme

The message-passing scheme of Bodnar et al. [16] lets a k-cell σ receive four kinds of messages, one per neighbourhood type. Denote by $\mathbf{h}_\sigma^{(\ell)} \in \mathbb{R}^d$ the embedding of σ at layer ℓ. The messages are

$$\mathbf{m}_{\mathcal{B}}^{(\ell+1)}(\sigma) = \mathrm{AGG}_{\mathcal{B}}\left\{ M_{\mathcal{B}}\left(\mathbf{h}_\sigma^{(\ell)}, \mathbf{h}_\tau^{(\ell)}\right) : \tau \in \mathcal{B}(\sigma) \right\}, \tag{A.1}$$

$$\mathbf{m}_{\mathcal{C}}^{(\ell+1)}(\sigma) = \mathrm{AGG}_{\mathcal{C}}\left\{ M_{\mathcal{C}}\left(\mathbf{h}_\sigma^{(\ell)}, \mathbf{h}_\tau^{(\ell)}\right) : \tau \in \mathcal{C}(\sigma) \right\}, \tag{A.2}$$

$$\mathbf{m}_{\downarrow}^{(\ell+1)}(\sigma) = \mathrm{AGG}_{\downarrow}\left\{ M_{\downarrow}\left(\mathbf{h}_\sigma^{(\ell)}, \mathbf{h}_\tau^{(\ell)}, \mathbf{h}_\delta^{(\ell)}\right) : \tau \in \mathcal{N}_{\downarrow}(\sigma),\ \delta \in \mathcal{B}(\sigma) \cap \mathcal{B}(\tau) \right\}, \tag{A.3}$$

$$\mathbf{m}_{\uparrow}^{(\ell+1)}(\sigma) = \mathrm{AGG}_{\uparrow}\left\{ M_{\uparrow}\left(\mathbf{h}_\sigma^{(\ell)}, \mathbf{h}_\tau^{(\ell)}, \mathbf{h}_\delta^{(\ell)}\right) : \tau \in \mathcal{N}_{\uparrow}(\sigma),\ \delta \in \mathcal{C}(\sigma) \cap \mathcal{C}(\tau) \right\}, \tag{A.4}$$

where the M are message functions, the AGG are permutation-invariant aggregators over multisets, and δ ranges over the bridge cells. The cell's embedding is then updated from its previous state and the four messages. Equation 2.12 of the main text is the matrix form of this update. The matrix form of Section 2.4.2 is the special case with identity message functions and sum aggregation, in which each message becomes one product of an incidence matrix, or of two incidence matrices, with the embedding matrix of an order.

Bodnar et al. show that, for the expressive power of cellular Weisfeiler-Lehman colour refinement, the co-boundary and lower-adjacency messages are redundant given the boundary and upper-adjacency ones [16]. They note that this theoretical redundancy does not make the corresponding channels useless in practice, since they can carry useful inductive bias for tasks other than isomorphism testing. LightCCN omits the lower adjacency of edges, the channel this result declares removable, and keeps the remaining ones, letting the softmax weights of Chapter 4 decide how much each is used. Appendix F confirms empirically that adding it changes nothing.

B

Additional Results

This appendix collects the supporting evidence referenced from Chapter 5, the RQ1 tables at the remaining cutoffs, the full RQ2 grid and weight table, and the additional metric versions of the main-text figures. Conclusions are unchanged relative to the featured cutoff and metrics.

B.1. RQ1 at Other Cutoffs

Table B.1: Relative change of LightCCN-full over LightGCN, in percent, for every metric family and cutoff. Stars mark per-user paired t-test significance at $p < 0.05$.

dataset	metric	K=3	K=5	K=10	K=20	K=50
CiaoDVD	ARHR	+25.7*	+20.8*	+16.0*	+13.9*	+13.0*
	NDCG	+22.3*	+15.9*	+9.7*	+6.6*	+5.0*
	Recall	+16.3*	+8.6	+2.5	-0.4	-0.3
Epinions	ARHR	+12.2*	+9.8*	+8.2*	+7.7*	+6.7*
	NDCG	+12.5*	+10.1*	+7.1*	+6.2*	+3.8*
	Recall	+14.1*	+8.5*	+4.1*	+3.3*	+0.0
Beidian	ARHR	+10.6*	+6.5	+5.6	+4.9	+5.0
	NDCG	+11.3*	+5.0	+4.2	+2.7	+3.3*
	Recall	+12.7*	+2.3	+2.3	+0.6	+2.1
Foursquare-NYC	ARHR	+7.7	+6.6	+6.2*	+6.2*	+5.6*
	NDCG	+5.5	+5.9*	+3.8	+4.7*	+3.6*
	Recall	+7.3	+5.0	+4.1	+4.7*	+2.5
Foursquare-TKY	ARHR	+3.6*	+3.3*	+2.9*	+2.4*	+2.2*
	NDCG	+3.0*	+2.6*	+2.2*	+1.1	+0.7
	Recall	+4.6*	+3.7*	+2.7*	+0.3	-0.2
Gowalla	ARHR	+0.6	+0.5	+0.5	+0.4	+0.3
	NDCG	+0.6	+0.7	+0.4	+0.1	-0.0
	Recall	+0.3	+0.2	+0.1	-0.1	-0.3

Table B.2: RQ1 results at $K = 3$, in the layout of Table 5.3. Beidian's significant cells are at this cutoff.

dataset	model	NDCG@3	Recall@3	ARHR@3
CiaoDVD	LightGCN	0.0196	0.0245	0.0151
	LightCCN-full	0.0239*	0.0285*	0.0189*
	Δ%	**+22.3%***	**+16.3%***	**+25.7%***
Epinions	LightGCN	0.0182	0.0145	0.0097
	LightCCN-full	0.0204*	0.0166*	0.0109*
	Δ%	**+12.5%***	**+14.1%***	**+12.2%***
Beidian	LightGCN	0.0289	0.0388	0.0255
	LightCCN-full	0.0321*	0.0437*	0.0281*
	Δ%	**+11.3%***	**+12.7%***	**+10.6%***
Foursquare-NYC	LightGCN	0.0816	0.0150	0.0103
	LightCCN-full	0.0860	0.0161	0.0111
	Δ%	**+5.5%**	**+7.3%**	**+7.7%**
Foursquare-TKY	LightGCN	0.2253	0.0374	0.0265
	LightCCN-full	0.2322*	0.0391*	0.0275*
	Δ%	**+3.0%***	**+4.6%***	**+3.6%***
Gowalla	LightGCN	0.1445	0.0651	0.0467
	LightCCN-full	0.1454	0.0653	0.0469
	Δ%	**+0.6%**	**+0.3%**	**+0.6%**

Table B.3: RQ1 results at $K = 10$, in the layout of Table 5.3.

dataset	model	NDCG@10	Recall@10	ARHR@10
CiaoDVD	LightGCN	0.0334	0.0633	0.0214
	LightCCN-full	0.0366*	0.0649	0.0248*
	Δ%	**+9.7%***	**+2.5%**	**+16.0%***
Epinions	LightGCN	0.0248	0.0362	0.0133
	LightCCN-full	0.0266*	0.0377*	0.0143*
	Δ%	**+7.1%***	**+4.1%***	**+8.2%***
Beidian	LightGCN	0.0481	0.0934	0.0345
	LightCCN-full	0.0501	0.0956	0.0364
	Δ%	**+4.2%**	**+2.3%**	**+5.6%**
Foursquare-NYC	LightGCN	0.0611	0.0321	0.0131
	LightCCN-full	0.0635	0.0334	0.0139*
	Δ%	**+3.8%**	**+4.1%**	**+6.2%***
Foursquare-TKY	LightGCN	0.1537	0.0726	0.0325
	LightCCN-full	0.1570*	0.0745*	0.0334*
	Δ%	**+2.2%***	**+2.7%***	**+2.9%***
Gowalla	LightGCN	0.1345	0.1227	0.0562
	LightCCN-full	0.1351	0.1228	0.0565
	Δ%	**+0.4%**	**+0.1%**	**+0.5%**

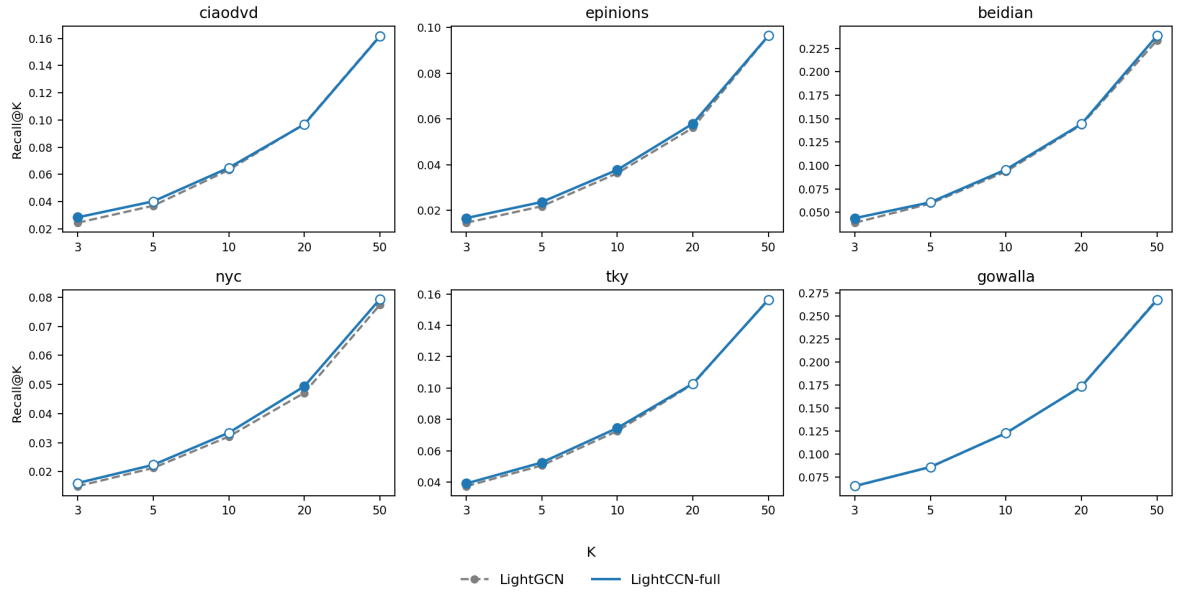

Figure B.1: Recall@K per dataset, companion to Figure 5.1.

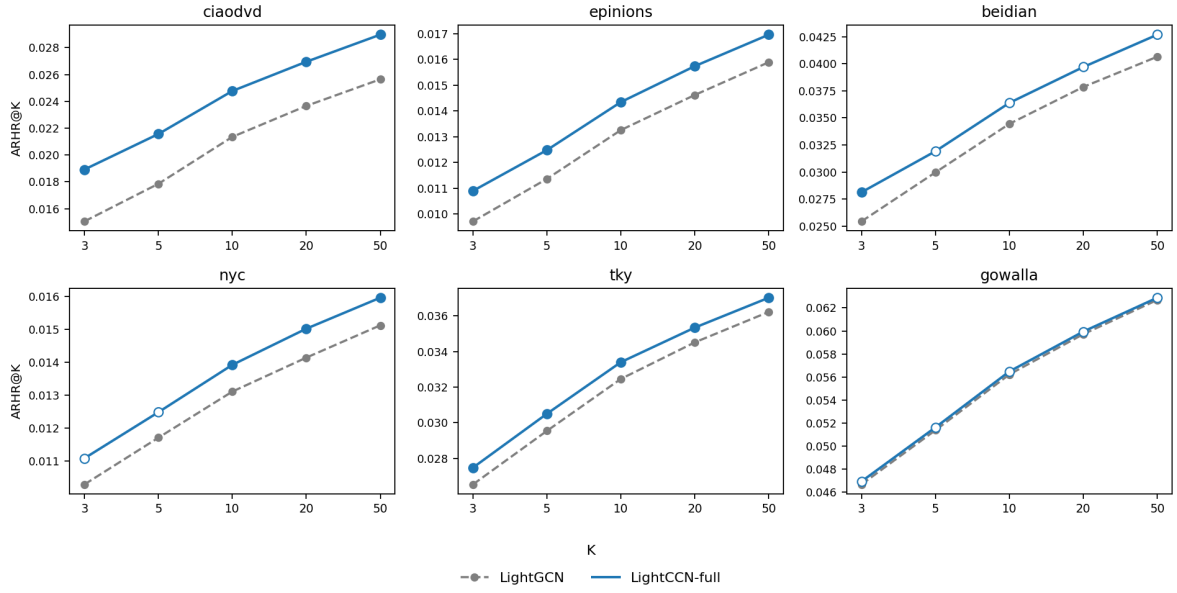

Figure B.2: ARHR@K per dataset, companion to Figure 5.1.

B.2. RQ2 Supporting Tables and Figures

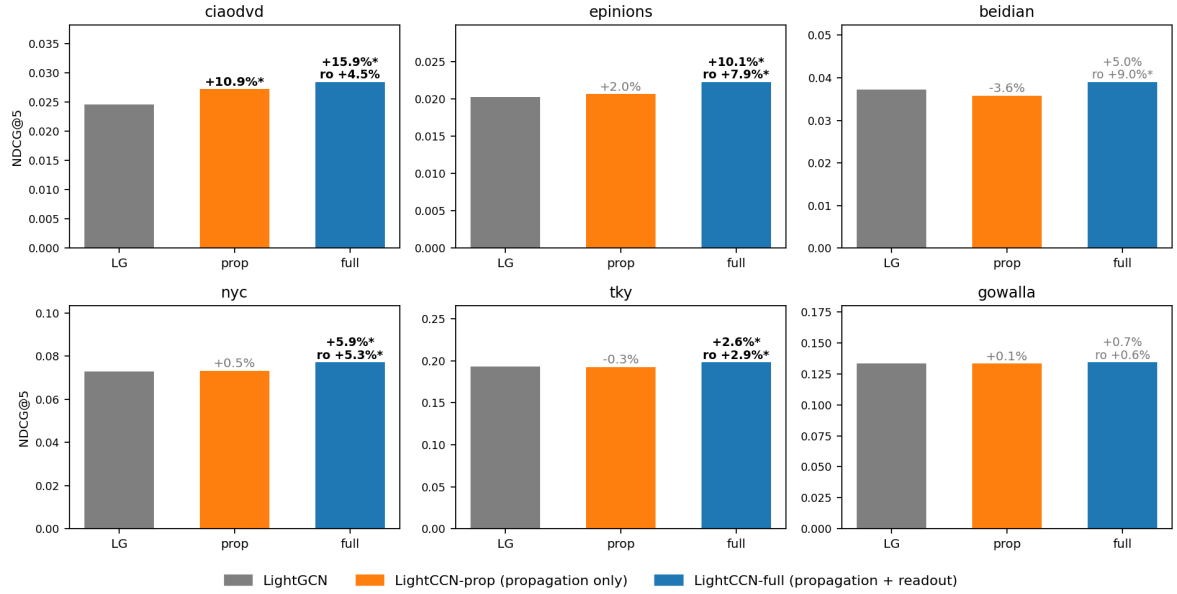

Figure B.3: NDCG@5 of the three variants per dataset, with LightGCN in gray, LightCCN-prop in orange, and LightCCN-full in blue. Bar labels give the change over LightGCN, with stars for significance. The second line above the full bar (ro) is the readout contribution, LightCCN-full over LightCCN-prop, with its own star. The Recall@5 version is in Figure B.6.

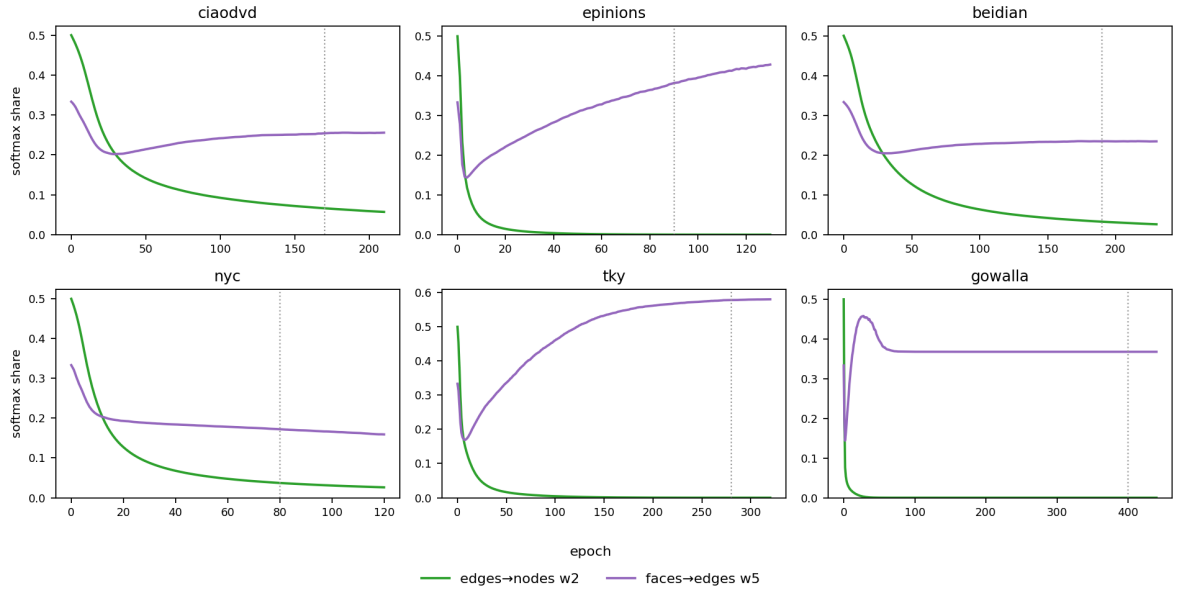

Figure B.4: Propagation shares w_2 (green) and w_5 (purple) over training epochs, per dataset. The dotted vertical line marks the reported epoch.

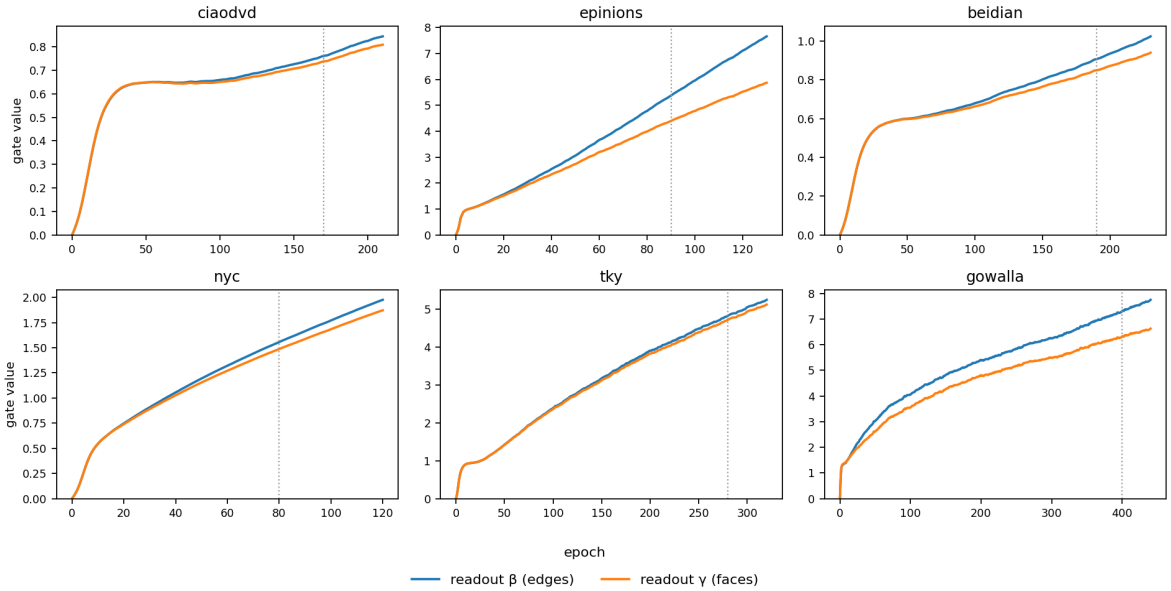

Figure B.5: Readout gates β (blue) and γ (orange) over training epochs, per dataset. Both start at exactly zero. The dotted vertical line marks the reported epoch.

Table B.4: Full RQ2 grid, with relative changes in percent for all three comparisons at $K = 5$ and $K = 10$ (A = ARHR, N = NDCG, R = Recall). Stars mark per-user paired t-test significance of the named comparison at $p < 0.05$.

dataset	comparison	A@5	N@5	R@5	A@10	N@10	R@10
CiaoDVD	full vs LightGCN	+20.8*	+15.9*	+8.6	+16.0*	+9.7*	+2.5
	prop vs LightGCN	+14.6*	+10.9*	+5.1	+10.9*	+5.3*	-0.3
	full vs prop	+5.4	+4.5	+3.3	+4.7	+4.2	+2.8
Epinions	full vs LightGCN	+9.8*	+10.1*	+8.5*	+8.2*	+7.1*	+4.1*
	prop vs LightGCN	+1.9	+2.0	+2.5	+1.8	+1.2	+1.7
	full vs prop	+7.8*	+7.9*	+5.9*	+6.3*	+5.9*	+2.3
Beidian	full vs LightGCN	+6.5	+5.0	+2.3	+5.6	+4.2	+2.3
	prop vs LightGCN	-2.0	-3.6	-6.5	-1.1	-1.9	-2.9
	full vs prop	+8.7*	+9.0*	+9.4*	+6.8*	+6.2*	+5.4*
Foursquare-NYC	full vs LightGCN	+6.6	+5.9*	+5.0	+6.2*	+3.8	+4.1
	prop vs LightGCN	+1.1	+0.5	+2.8	+0.7	-0.8	+0.8
	full vs prop	+5.5*	+5.3*	+2.2	+5.5*	+4.6*	+3.2
Foursquare-TKY	full vs LightGCN	+3.3*	+2.6*	+3.7*	+2.9*	+2.2*	+2.7*
	prop vs LightGCN	-0.3	-0.3	+0.7	+0.1	+0.6	+1.9*
	full vs prop	+3.5*	+2.9*	+2.9*	+2.8*	+1.5*	+0.8
Gowalla	full vs LightGCN	+0.5	+0.7	+0.2	+0.5	+0.4	+0.1
	prop vs LightGCN	-0.2	+0.1	-0.2	-0.1	+0.1	+0.1
	full vs prop	+0.7	+0.6	+0.4	+0.6	+0.4	+0.0

Table B.5: Complete propagation weight table at the reported epoch, with softmax shares per order (Equations 4.5–4.7) and readout gates.

dataset	node w_1/w_2	edge $w_3/w_4/w_5$	face w_6/w_7	β	γ
CiaoDVD	0.934 / 0.0662	0.440 / 0.306 / 0.254	0.686 / 0.314	0.76	0.74
Epinions	1.000 / 0.0002	0.296 / 0.322 / 0.382	0.893 / 0.107	5.38	4.40
Beidian	0.967 / 0.0326	0.545 / 0.220 / 0.235	0.748 / 0.252	0.91	0.85
Foursquare-NYC	0.963 / 0.0372	0.357 / 0.471 / 0.172	0.834 / 0.166	1.56	1.49
Foursquare-TKY	1.000 / 0.0000	0.032 / 0.390 / 0.578	0.687 / 0.313	4.82	4.72
Gowalla	1.000 / 0.0000	0.019 / 0.614 / 0.368	0.796 / 0.204	7.32	6.33

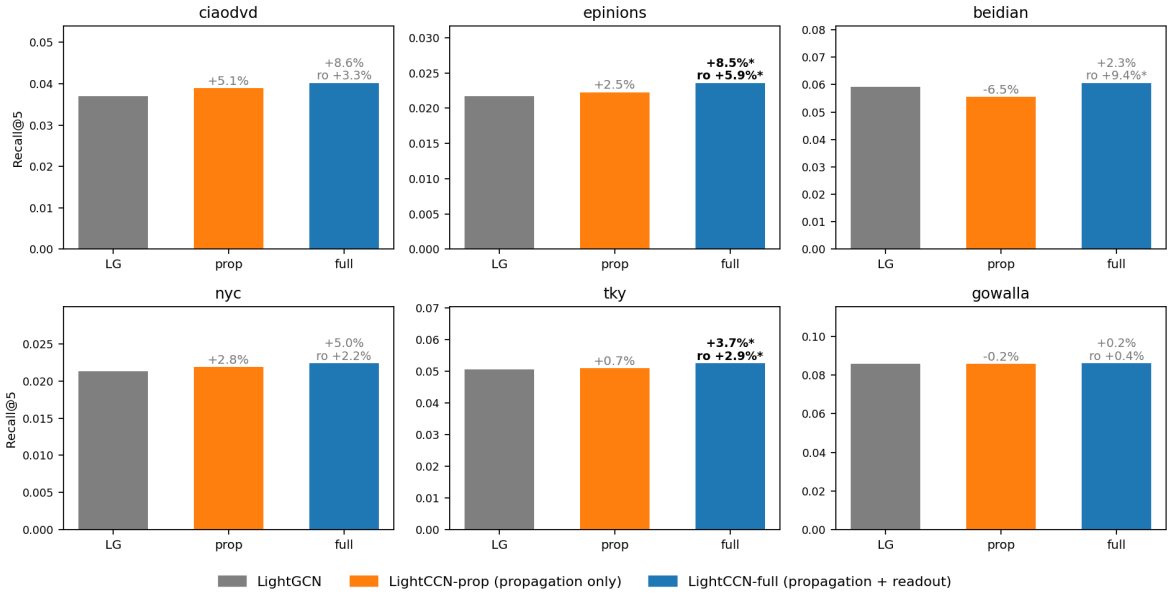

Figure B.6: Recall@5 of the three variants per dataset, companion to Figure B.3.

B.3. RQ3 Supporting Tables and Figures

Table B.6: Per-depth relative change of LightCCN-full over same-depth LightGCN, shown as ARHR@5 in percent with NDCG@5 in parentheses. Stars mark per-user paired t-test significance at $p < 0.05$.

dataset	L=1	L=2	L=3	L=4	L=5	L=6
CiaoDVD	+3.4 (+3.3)	+7.9* (+7.4*)	+20.8* (+15.9*)	+5.8 (+4.8)	+23.2* (+20.1*)	+2.9 (+3.1)
Epinions	+18.1* (+17.7*)	+12.4* (+11.6*)	+9.8* (+10.1*)	+8.4* (+8.6*)	+8.5* (+9.3*)	+3.2 (+3.7)
Beidian	-1.4 (-1.0)	-5.9 (-6.3)	+6.5 (+5.0)	+7.8 (+5.9)	+1.4 (+2.6)	+11.3* (+11.2*)
Foursquare-NYC	+12.9* (+13.3*)	+21.6* (+16.8*)	+6.6 (+5.9*)	+11.0* (+8.6*)	+5.8* (+5.5*)	+7.1* (+6.9*)
Foursquare-TKY	-0.5 (-0.5)	-0.7 (-1.2)	+3.3* (+2.6*)	+2.0* (+1.5*)	+3.0* (+2.4*)	+4.0* (+3.9*)
Gowalla	-0.3 (-0.1)	-0.8 (-1.5*)	+0.5 (+0.7)	-0.5 (-0.6)	+0.5 (+0.1)	+1.1* (+1.0*)

Table B.7: User-side smoothness S_U per depth, shown as LightCCN-full / LightGCN.

dataset	L=1	L=2	L=3	L=4	L=5	L=6
CiaoDVD	3449 / 3563	3060 / 2986	2945 / 2535	2695 / 2440	2721 / 2065	2610 / 2172
Epinions	19312 / 22921	18267 / 21255	18816 / 21679	18202 / 21029	18513 / 21546	18206 / 21794
Beidian	3392 / 3602	3040 / 3357	3152 / 3265	2953 / 3003	2721 / 2911	2919 / 2892
Foursquare-NYC	686 / 816	630 / 693	611 / 693	626 / 670	535 / 640	568 / 630
Foursquare-TKY	2127 / 2207	2033 / 2161	1987 / 2110	1931 / 2079	1884 / 2096	1832 / 1995
Gowalla	28756 / 28981	28179 / 27675	27304 / 28430	24822 / 27051	24351 / 26117	24209 / 24278

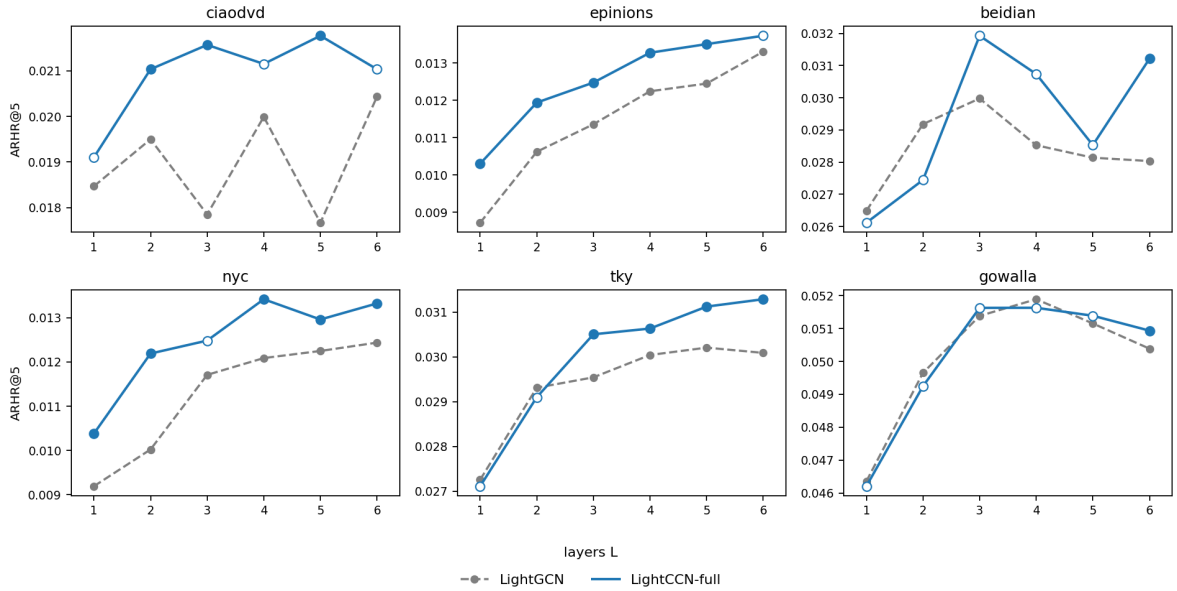

Figure B.7: ARHR@5 against depth per dataset, companion to Figure 5.2.

B.4. RQ4 Additional Metrics

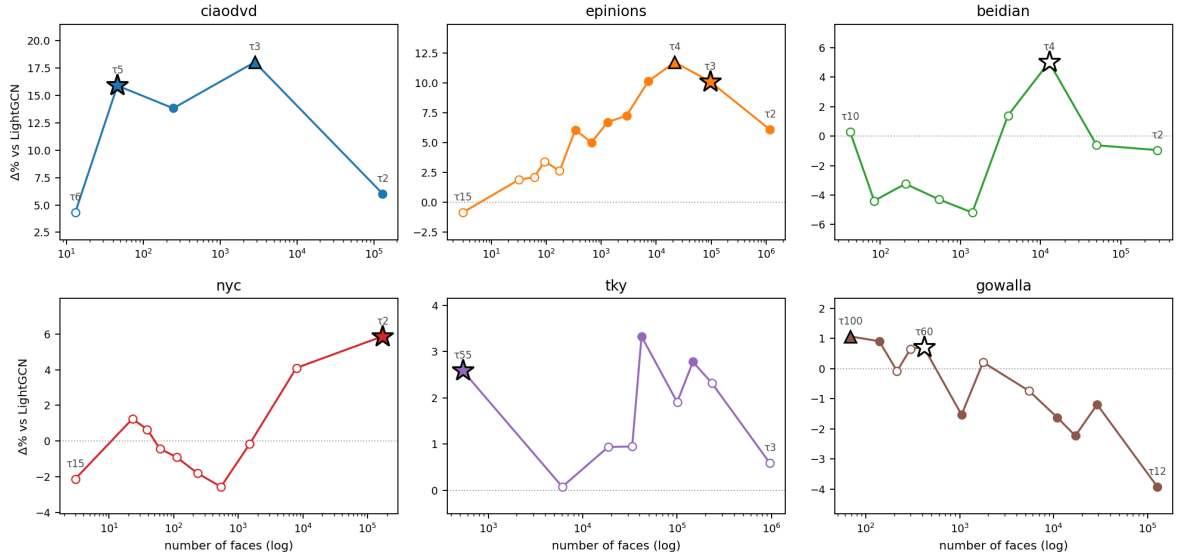

Figure B.8: NDCG@5 change over LightGCN, in percent, against the number of faces, companion to Figure 5.4.

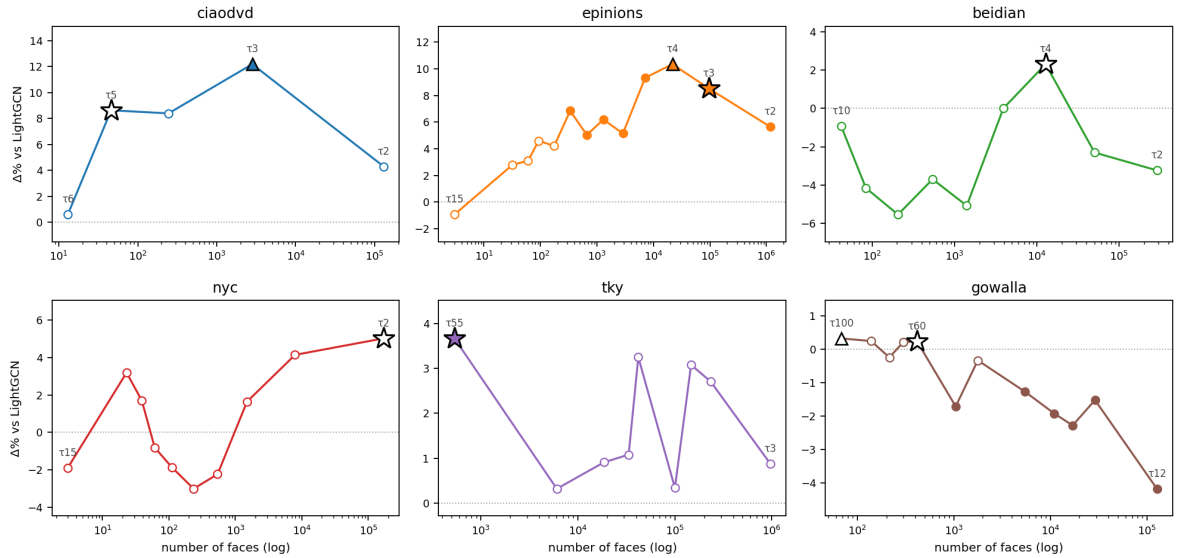

Figure B.9: Recall@5 change over LightGCN, in percent, against the number of faces, companion to Figure 5.4.

B.5. LightGCN Results Reported in Its Paper

Table B.8 places the LightGCN results reported in its paper [8] next to those obtained with the protocol of this thesis, on two of the three datasets the paper reports. Gowalla is part of our test benchmarks for LightCCN, and Yelp2018 is used only for this comparison. The published values were selected by early stopping on the test set, as described under epoch selection in Section 5.1.1, and are marked with * since they are not directly comparable to validation-selected values.

Table B.8: LightGCN at $K = 20$ and three layers, values in percent. Published* values are those reported in the LightGCN paper [8], which selects the epoch on the test set. The values of this thesis are selected on validation and reported on the test split.

	Recall@20		NDCG@20	
dataset	published*	this thesis	published*	this thesis
Gowalla	18.23	17.39	15.55	14.86
Yelp2018	6.39	6.22	5.25	5.07

C

Ablation: Readout Target

The higher-order readout of Section 4.6 injects the pooled cell embeddings into the *user* tower. This ablation redirects the identical machinery to the item tower (item readout) and to both towers (user+item readout), leaving the backbone unchanged, under the exact protocol of the main campaigns. Table C.1 reports the results. Stars mark per-user paired t-test significance ($p < 0.05$) of the comparison in the respective column.

Table C.1: Readout-target ablation at $K = 5$, values in percent. Variant cells show the raw value and, in parentheses, the change vs LightGCN; the vs-full column is the change vs LightCCN-full, tested directly against it.

				item readout		user+item readout	
dataset	metric	LightGCN	LightCCN-full	value (vs LightGCN)	vs full	value (vs LightGCN)	vs full
CiaoDVD	NDCG@5	2.45	2.84* (+15.9*)	2.46 (+0.2)	-13.5*	2.83* (+15.5*)	-0.3
	Recall@5	3.70	4.02 (+8.6)	3.67 (-0.8)	-8.7*	4.09* (+10.4*)	+1.6
	ARHR@5	1.79	2.16* (+20.8*)	1.80 (+0.6)	-16.8*	2.12* (+19.0*)	-1.5
Epinions	NDCG@5	2.02	2.23* (+10.1*)	1.91* (-5.6*)	-14.3*	2.02 (+0.0)	-9.2*
	Recall@5	2.17	2.36* (+8.5*)	2.05* (-5.5*)	-12.9*	2.16 (-0.6)	-8.4*
	ARHR@5	1.14	1.25* (+9.8*)	1.07* (-5.7*)	-14.1*	1.14 (+0.5)	-8.5*
Beidian	NDCG@5	3.72	3.90 (+5.0)	3.95* (+6.4*)	+1.3	4.30* (+15.6*)	+10.1*
	Recall@5	5.93	6.07 (+2.3)	6.23 (+5.1)	+2.7	6.34 (+6.9)	+4.5
	ARHR@5	3.00	3.19 (+6.5)	3.21* (+7.2*)	+0.7	3.63* (+20.9*)	+13.6*
Foursquare-NYC	NDCG@5	7.28	7.71* (+5.9*)	7.39 (+1.5)	-4.1*	7.64 (+4.9)	-0.9
	Recall@5	2.13	2.24 (+5.0)	2.16 (+1.3)	-3.5	2.21 (+3.8)	-1.2
	ARHR@5	1.17	1.25 (+6.6)	1.23 (+4.9)	-1.6	1.28* (+8.9*)	+2.1
Foursquare-TKY	NDCG@5	19.32	19.82* (+2.6*)	19.30 (-0.1)	-2.6*	18.02* (-6.7*)	-9.1*
	Recall@5	5.07	5.25* (+3.7*)	5.15 (+1.6)	-2.0	4.85* (-4.2*)	-7.6*
	ARHR@5	2.95	3.05* (+3.3*)	3.01 (+1.8)	-1.4	2.80* (-5.4*)	-8.3*
Gowalla	NDCG@5	13.35	13.44 (+0.7)	13.47* (+0.9*)	+0.2	13.44* (+0.6*)	-0.0
	Recall@5	8.58	8.61 (+0.2)	8.63 (+0.6)	+0.3	8.62 (+0.4)	+0.2
	ARHR@5	5.14	5.16 (+0.5)	5.18 (+0.7)	+0.2	5.17 (+0.6)	+0.1

The user readout is the right target. The item readout never beats the full model and is significantly worse than it on CiaoDVD (-13.5^* NDCG@5), Epinions (-14.3^* NDCG@5), Foursquare-NYC (-4.1^* NDCG@5) and Foursquare-TKY (-2.6^* NDCG@5). On Epinions it falls significantly below LightGCN itself. Training supports the same conclusion from the inside. The item-readout gates barely open or turn negative (CiaoDVD $-0.08/-0.09$, Epinions $\gamma = -0.53$), so the optimiser itself declines the item-side injection. The user+item variant ties with the full model on CiaoDVD, Foursquare-NYC and Gowalla and is significantly worse on Epinions (-9.2^* NDCG@5) and Foursquare-TKY (-9.1^* NDCG@5). The one positive deviation is Beidian, where the user+item readout is $+10.1^*$ NDCG@5 over the full model and $+15.6^*$ NDCG@5 over LightGCN, the best Beidian configuration. We report it as a dataset-specific finding rather than a change of default.

D

Ablation: Propagation Backbone under the Readout

This ablation keeps the user readout and replaces the stateful edges+faces backbone of Section 4.5 with three alternatives, stateful edges only (no face order), derived edges (edge states recomputed from node embeddings at every layer), and derived edges and faces. Table D.1 reports the results.

Table D.1: Backbone ablation at $K = 5$, values in percent, in the layout of Table C.1.

				stateful edges		derived edges		derived edges+faces	
dataset	metric	LightGCN	LightCCN-full	value (vs LG)	vs full	value (vs LG)	vs full	value (vs LG)	vs full
CiaoDVD	NDCG@5	2.45	2.84* (+15.9*)	2.83* (+15.5*)	-0.3	2.84* (+15.7*)	-0.2	2.43 (-0.9)	-14.5*
	Recall@5	3.70	4.02 (+8.6)	4.04 (+9.2)	+0.6	4.04 (+9.0)	+0.4	3.56 (-4.0)	-11.6*
	ARHR@5	1.79	2.16* (+20.8*)	2.14* (+20.1*)	-0.6	2.16* (+21.0*)	+0.2	1.78 (-0.2)	-17.4*
Epinions	NDCG@5	2.02	2.23* (+10.1*)	2.15* (+6.5*)	-3.3*	2.21* (+9.4*)	-0.7	2.21* (+9.1*)	-0.9
	Recall@5	2.17	2.36* (+8.5*)	2.32* (+6.8*)	-1.6	2.36* (+8.5*)	+0.0	2.35* (+8.1*)	-0.4
	ARHR@5	1.14	1.25* (+9.8*)	1.21* (+6.9*)	-2.6	1.26* (+10.5*)	+0.6	1.24* (+9.5*)	-0.3
Beidian	NDCG@5	3.72	3.90 (+5.0)	3.86 (+3.9)	-1.0	3.73 (+0.4)	-4.4*	3.92 (+5.6)	+0.5
	Recall@5	5.93	6.07 (+2.3)	6.07 (+2.3)	+0.0	5.82 (-1.8)	-4.1*	6.15 (+3.7)	+1.4
	ARHR@5	3.00	3.19 (+6.5)	3.14 (+4.8)	-1.6	3.05 (+1.6)	-4.6*	3.20 (+6.7)	+0.2
Foursquare-NYC	NDCG@5	7.28	7.71* (+5.9*)	7.52 (+3.3)	-2.4	7.51 (+3.1)	-2.6	7.55 (+3.8)	-2.0
	Recall@5	2.13	2.24 (+5.0)	2.23 (+4.8)	-0.2	2.24 (+5.1)	+0.1	2.21 (+3.8)	-1.2
	ARHR@5	1.17	1.25 (+6.6)	1.22 (+4.0)	-2.4	1.22 (+4.0)	-2.5	1.25 (+6.8)	+0.1
Foursquare-TKY	NDCG@5	19.32	19.82* (+2.6*)	19.55 (+1.2)	-1.4	19.57 (+1.3)	-1.3	19.57 (+1.3)	-1.3
	Recall@5	5.07	5.25* (+3.7*)	5.17* (+2.1*)	-1.5	5.19* (+2.5*)	-1.1	5.17 (+2.1)	-1.6
	ARHR@5	2.95	3.05* (+3.3*)	3.00* (+1.6*)	-1.6*	3.00* (+1.7*)	-1.5*	3.01* (+1.9*)	-1.3
Gowalla	NDCG@5	13.35	13.44 (+0.7)	13.48* (+1.0*)	+0.3	13.38 (+0.2)	-0.5	13.40 (+0.4)	-0.3
	Recall@5	8.58	8.61 (+0.2)	8.59 (+0.1)	-0.2	8.58 (-0.0)	-0.2	8.56 (-0.3)	-0.5
	ARHR@5	5.14	5.16 (+0.5)	5.17 (+0.6)	+0.1	5.14 (+0.1)	-0.4	5.15 (+0.3)	-0.2

The stateful edges+faces backbone is the only one that is never significantly worse. On Foursquare-NYC, Foursquare-TKY and Gowalla all three alternatives are statistically tied with it, consistent with RQ2's finding that the readout carries most of the value. Each alternative, however, fails on exactly one dataset. Stateful edges only is -3.3^* NDCG@5 on Epinions (the one dataset where faces matter in propagation), derived edges is -4.4^* NDCG@5 on Beidian, and derived edges+faces is -14.5^* NDCG@5 on CiaoDVD, where it collapses to LightGCN parity. The choice of the stateful backbone is therefore a robustness argument rather than a dominance argument. It is the configuration with no failure case.

E

Ablation: User–Item Edges as 1-Cells

By construction, the complex of Section 4.3 is item-side only. Faces are item triples and the 1-cells are the item pairs inside them, so users lie on no cell. This ablation adds every training interaction as a 1-cell whose boundary is its user and its item. Users then lie on edges of the complex, with three consequences. B_1 gains one row per user, the edge channel of the node update (w_2) delivers edge states to users and not only to items, and the pooled edge signal of the readout, $\mathbf{P}_E$, includes the interaction edges. The edge lower adjacency is not used in this variant. Item–item edges exchange information through their faces and their endpoint items, while the interaction edges do so only through their endpoints. Appendix F's second variant repeats this augmentation with the edge lower adjacency added. Table E.1 reports the results and Table E.2 the learned-parameter shift.

Table E.1: User–item-edges ablation at $K = 5$, values in percent, in the layout of Table C.1.

dataset	metric	LightGCN	LightCCN-full	with UI edges (vs LightGCN)	vs full
CiaoDVD	NDCG@5	2.45	2.84* (+15.9*)	2.55 (+4.1)	-10.2*
	Recall@5	3.70	4.02 (+8.6)	3.70 (-0.2)	-8.1*
	ARHR@5	1.79	2.16* (+20.8*)	1.89 (+5.8)	-12.4*
Epinions	NDCG@5	2.02	2.23* (+10.1*)	1.85* (-8.5*)	-16.9*
	Recall@5	2.17	2.36* (+8.5*)	1.98* (-9.0*)	-16.2*
	ARHR@5	1.14	1.25* (+9.8*)	1.04* (-8.7*)	-16.9*
Beidian	NDCG@5	3.72	3.90 (+5.0)	3.69 (-0.8)	-5.6*
	Recall@5	5.93	6.07 (+2.3)	5.87 (-0.9)	-3.2
	ARHR@5	3.00	3.19 (+6.5)	2.97 (-0.9)	-6.9*
Foursquare-NYC	NDCG@5	7.28	7.71* (+5.9*)	7.40 (+1.7)	-3.9*
	Recall@5	2.13	2.24 (+5.0)	2.22 (+4.1)	-0.9
	ARHR@5	1.17	1.25 (+6.6)	1.20 (+2.5)	-3.8
Foursquare-TKY	NDCG@5	19.32	19.82* (+2.6*)	16.38* (-15.2*)	-17.4*
	Recall@5	5.07	5.25* (+3.7*)	4.41* (-12.9*)	-16.0*
	ARHR@5	2.95	3.05* (+3.3*)	2.60* (-12.1*)	-14.9*
Gowalla	NDCG@5	13.35	13.44 (+0.7)	10.98* (-17.8*)	-18.3*
	Recall@5	8.58	8.61 (+0.2)	7.14* (-16.9*)	-17.1*
	ARHR@5	5.14	5.16 (+0.5)	4.25* (-17.3*)	-17.7*

Table E.2: Channel shares and readout gates at the reported epoch, before → after adding user–item edges to the complex.

dataset	w_1 (LightGCN channel)	w_2 (edge channel)	β	γ	UI edges added
CiaoDVD	0.934 → 0.797	0.066 → 0.203	0.84 → 1.06	0.81 → 0.76	30,363
Epinions	1.000 → 0.026	0.000 → 0.974	7.66 → 6.10	5.87 → 0.76	360,281
Beidian	0.967 → 0.659	0.033 → 0.341	1.02 → 1.66	0.94 → 0.68	31,920
Foursquare-NYC	0.963 → 0.819	0.037 → 0.181	1.98 → 1.86	1.87 → 1.38	71,733
Foursquare-TKY	1.000 → 0.023	0.000 → 0.977	5.25 → 5.05	5.12 → 0.97	167,276
Gowalla	1.000 → 0.000	0.000 → 1.000	7.75 → 10.02	6.63 → 6.29	780,270

Adding user–item edges hurts on all six datasets, from -3.9^* (Foursquare-NYC) to -18.3^* (Gowalla) NDCG@5 against the plain full model, and on Epinions, Foursquare-TKY and Gowalla the variant falls significantly below LightGCN as well. Table E.2 shows the mechanism. With user rows in B_1, the node-level softmax is dominated by the edge channel. The share w_2 rises from 0.03 to 0.07 into the range 0.18 to 0.34 on the small datasets and to 0.974 (Epinions), 0.977 (Foursquare-TKY) and 1.000 (Gowalla), so the node update becomes a re-encoding of the interaction graph through edge states and displaces the directly normalised LightGCN channel (w_1 drops toward 0.02). At the same time, the few hundred face-incident edges drown in edge tables of up to 780k interaction edges, diluting the pooled readout, while the gates move only mildly. The damage is done in propagation, not in the readout. Gowalla provides the cleanest control. Higher-order structure is inert there by construction, yet user–item edges still cost -18.3^* NDCG@5, so the harm requires no face interaction at all. The complex therefore stays item-side, exactly as the closure argument of Section 4.3 constructs it.

F

Ablation: Edge Lower Adjacency

Two edges of the complex are lower adjacent when they share an item. The general update of Equation 2.12 includes this channel through $B_1^\top B_1$, and LightCCN's edge update omits it. The first variant below adds the symmetrically normalised lower adjacency of the item–item edges as a fourth edge channel in the softmax. The second additionally gives every user–item interaction its own 1-cell. The lower adjacency then connects any two edges that share an endpoint, so an item–item edge also receives the interaction edges at its two items, and interaction edges receive one another through shared users and items. Both are trained under the protocol of Section 5.1.1 at the selected τ. Table F.1 reports the results.

Table F.1: Edge lower adjacency at $K = 5$, values in percent. Stars mark a per-user paired t-test against LightCCN-full at $p < 0.05$.

dataset	model	Recall@5	NDCG@5	ARHR@5
CiaoDVD	LightGCN	3.70	2.45	1.79
	LightCCN-full	4.02	2.84	2.16
	+ lower adjacency	4.05	2.86	2.17
	+ lower adj. and u–i edges	3.89	2.69	1.99*
Epinions	LightGCN	2.17	2.02	1.14
	LightCCN-full	2.36	2.23	1.25
	+ lower adjacency	2.37	2.22	1.24
	+ lower adj. and u–i edges	1.99*	1.86*	1.05*
Beidian	LightGCN	5.93	3.72	3.00
	LightCCN-full	6.07	3.90	3.19
	+ lower adjacency	5.90	3.81	3.12
	+ lower adj. and u–i edges	6.01	3.81	3.08
Foursquare-NYC	LightGCN	2.13	7.28	1.17
	LightCCN-full	2.24	7.71	1.25
	+ lower adjacency	2.24	7.73	1.25
	+ lower adj. and u–i edges	2.22	7.40*	1.19*
Foursquare-TKY	LightGCN	5.07	19.32	2.95
	LightCCN-full	5.25	19.82	3.05
	+ lower adjacency	5.22	19.76	3.04
	+ lower adj. and u–i edges	4.42*	16.40*	2.59*
Gowalla	LightGCN	8.58	13.35	5.14
	LightCCN-full	8.61	13.44	5.16
	+ lower adjacency	8.62	13.48	5.16
	+ lower adj. and u–i edges	7.16*	11.02*	4.27*

Adding the lower adjacency changes nothing. The variant is statistically indistinguishable from

LightCCN-full on every dataset and metric, with the smallest p-value at 0.26, although the softmax does assign the channel weight. This is the redundancy that Bodnar et al. prove for lower-adjacency messages, whose one-step content is the composition of the two boundary messages the model already carries [16]. With user–item edges added, the variant is significantly worse than LightCCN-full on five of the six datasets and never better. This is the edge-level view of the collapse of Appendix E.