Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.085505

ARTICLE

Multi-Level Graph Signal Preservation for Sequential Recommendation with Selective State Spaces

Yitao Yang[1,2], **Peng Wu**[1,2,*], **Xiaoming Zhang**[3,*], **Renjie Xu**[3] **and Yong Zhang**[3]

[1]School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou, China
[2]Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University, Hangzhou, China
[3]National Key Laboratory of Intelligent Parallel Technology, Arms of the Army University, Beijing, China

*Corresponding Authors: Peng Wu. Email: wupeng@zstu.edu.cn; Xiaoming Zhang. Email: xhg.1999@tsinghua.org.cn

Received: 12 May 2026; Accepted: 13 July 2026; Published: 13 August 2026

ABSTRACT: Existing graph-enhanced sequential recommendation methods typically adopt a unidirectional information flow, in which graph embeddings are injected into the sequential encoder only at the input stage, after which the graph signal is progressively diluted through multiple layers of deep processing. In this paper, the graph signal dilution phenomenon is analyzed systematically across three levels—the input, representation, and prediction layers—and the GSPRec model is proposed to address this issue. The core of GSPRec is the Graph-Sequence Collaborative Injection (GSCI) module, comprising three lightweight components: the Graph Confidence Gate (GCG) controls GCN smoothing via dimension-wise bounded interpolation; the Graph Residual Aggregation (GRA) restores diluted signals through a graph skip connection; and the Graph Collaborative Prediction (GCP) injects collaborative signals into prediction scores via a de-meaned shortcut. GSCI introduces only 194 learnable parameters in total and is equipped with a zero-damage initialization guarantee. The sequential encoder is further enhanced with independent dual Mamba instances and an adaptive path router. Experiments on four benchmark datasets—Food, Movie, Book, and Douban—demonstrate that GSPRec outperforms eight baselines across all 16 evaluation metrics, with relative improvements of 0.61%–3.16% over the strongest baseline and less than 4% additional training time. A layer-wise probing analysis directly confirms that the graph signal is diluted within the sequential encoder and that the GSCI components counteract this loss, while the learned injection strengths adapt across datasets.

KEYWORDS: Sequential recommendation; graph convolutional network; state space model; graph signal; collaborative filtering

1 Introduction

Sequential recommendation predicts the next item a user is likely to interact with based on the historical interaction sequence, capturing the dynamic evolution of user preferences—a more expressive formulation than static preference modeling [1,2].

Early sequential recommendation methods relied on Markov chains [3]. With the rise of deep learning, RNNs [4,5] and CNNs [6,7] were employed to model user interaction sequences, later followed by memory networks [8] and Transformer-based approaches [9–12] that leverage self-attention for long-range dependencies. More recently, state space models have attracted attention due to their linear-complexity advantage: SIGMA [13] applied the selective state space model Mamba to sequential recommendation via the G-Mamba block. In parallel, graph neural networks—particularly GCNs such as LightGCN [14]—have demonstrated

strong collaborative filtering capabilities, motivating the integration of graph structures with sequential encoders to capture high-order collaborative signals.

However, an important phenomenon overlooked in existing graph-enhanced methods is identified, as illustrated in Fig. 1: the graph signal is progressively diluted during the multi-layer processing pipeline from the GCN to the final prediction. Representative methods such as EA-GPS [15] adopt a unidirectional information flow, where graph embeddings are injected only at the input stage and must then pass through multiple layers of sequence encoding, aggregation, and prediction transformations. Specifically, this dilution—used broadly to encompass both gradual signal attenuation and complete signal exclusion—manifests at three levels:

(1) *Input-layer dilution*: multi-layer GCN aggregation provides high-order collaborative signals but simultaneously causes over-smoothing [16], and existing methods feed GCN outputs into the sequential encoder without controlling the smoothing degree, losing discriminative information in certain dimensions.
(2) *Representation-layer dilution*: deep nonlinear transformations in the sequential encoder overwrite graph-based collaborative signals with sequential patterns, progressively weakening the original graph-structural information.
(3) *Prediction-layer dilution*: the final prediction layer relies solely on the encoder output for scoring, leaving graph-based collaborative signals unable to participate directly in prediction computation.

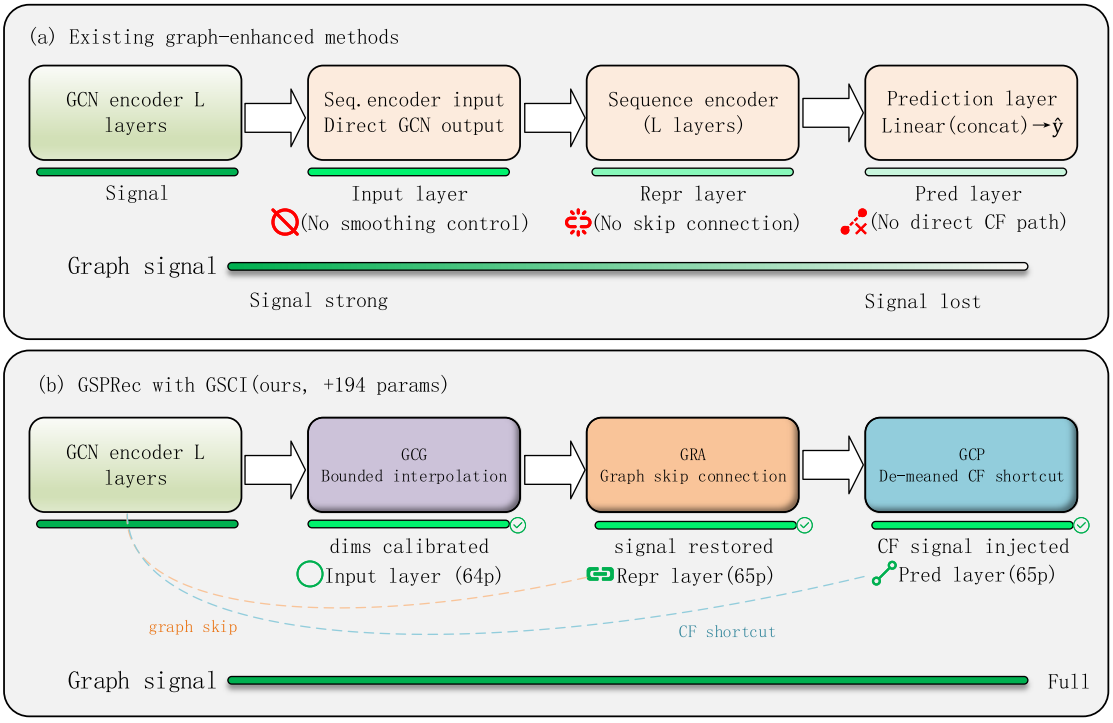

Figure 1: Illustration of graph signal dilution in existing methods (**a**) vs. multi-level graph signal preservation with GSCI in GSPRec (**b**). The signal strength bars show how graph information degrades through processing layers without GSCI, while GSCI preserves it at the input, representation, and prediction layers with only 194 additional parameters.

To overcome these limitations, a multi-level graph signal preservation model, termed GSPRec, is proposed from the perspective of graph–sequence collaboration. Its core innovation is the Graph-Sequence

Collaborative Injection (GSCI) module, which preserves graph signals at the three levels above through complementary components: (i) the *Graph Confidence Gate* (GCG) controls GCN smoothing via dimension-wise bounded interpolation, confining the output within the convex hull of GCN and raw embeddings to prevent harmful perturbations; (ii) the *Graph Residual Aggregation* (GRA) establishes a skip connection that bypasses the sequential encoder and supplements undiluted graph signals into the final interest representation; and (iii) the *Graph Collaborative Prediction* (GCP) injects collaborative signals at the prediction layer via a de-meaned inner product, where de-meaning adapts the injection strength to varying graph density conditions and eliminates popularity bias.

The three components collectively introduce only 194 learnable parameters and are equipped with a zero-damage initialization guarantee—at the beginning of training, the model behavior is nearly identical to that of the baseline. The sequential encoder is further improved by replacing the shared forward–backward Mamba instance with independent dual Mamba instances and by introducing a position-aware Adaptive Path Router in place of fixed-weight three-path fusion.

Experiments on four real-world benchmark datasets demonstrate that GSPRec outperforms all eight baselines across all evaluation metrics. The main contributions of this paper are summarized as follows:

1. The graph signal dilution problem in graph-enhanced sequential recommendation is systematically analyzed across three levels—the input, representation, and prediction layers—providing the design rationale for the GSCI module.
2. The GSCI module is proposed, comprising three lightweight components (GCG, GRA, and GCP) that introduce only 194 parameters in total, possess a zero-damage initialization guarantee, and act in a complementary manner across the three levels.
3. Extensive experiments show that GSPRec outperforms eight baselines across all 16 evaluation metrics, and a direct layer-wise probing analysis confirms that the graph signal is diluted within the sequential encoder and that GSCI counteracts this loss, while the learned injection strengths exhibit dataset-specific adaptation.

2 Preliminaries

This section introduces the problem definition and two key technical backgrounds involved in this work: graph convolutional networks and selective state space models.

2.1 Problem Definition

Let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ denote the user set and $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$ the item set. For a user $u \in \mathcal{U}$, the chronologically ordered interaction sequence is $S^u = \{i_1, i_2, \ldots, i_n\}$, where $i_t \in \mathcal{I}$ is the item interacted with at time step t and n is the sequence length. Sequential recommendation estimates the probability distribution over the next interacted item:

$$p(i_{n+1} = v \mid S^u_{1:n}), \quad v \in \mathcal{I} \tag{1}$$

In practice, the maximum sequence length is set to 50. Sequences exceeding this length are truncated to retain only the most recent interactions, while shorter sequences are left-padded with zero vectors.

2.2 Graph Convolutional Network

Graph convolutional networks (GCNs) learn node representations through message passing over graph structures [17]. In recommender systems, user–item interaction relationships naturally form a bipartite graph, where the adjacency matrix $\mathbf{A}$ encodes the connectivity between nodes. To ensure stable message

passing, $\mathbf{A}$ is symmetrically normalized. Let $\mathbf{D}$ denote the degree matrix; the symmetrically normalized adjacency matrix is defined as:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{2}$$

The propagation at the l-th GCN layer can be expressed as:

$$\mathbf{E}^{(l)} = \tilde{\mathbf{A}}\mathbf{E}^{(l-1)} \tag{3}$$

where $\mathbf{E}^{(0)}$ denotes the initial node embeddings. The above formulation follows the simplified propagation rule of LightGCN [14], which removes feature transformations and nonlinear activations, retaining only neighborhood aggregation. After L layers of propagation, the outputs of all layers are aggregated to obtain the final graph embeddings. This multi-layer propagation enables each node to acquire information from multi-hop neighbors, thereby capturing high-order collaborative filtering signals. However, as the number of propagation layers increases, node representations tend to converge, giving rise to the over-smoothing problem [16]. The specific construction of the graph encoder adopted in this work is detailed in Section 3.2.

2.3 Selective State Space Model

State space models (SSMs) originate from control theory and model sequential data through continuous-time state equations. Mamba [18] proposed the selective state space model, whose core formulation is a discretized state space equation:

$$\mathbf{h}_t = \bar{\mathbf{A}}_t\mathbf{h}_{t-1} + \bar{\mathbf{B}}_t\mathbf{x}_t \tag{4}$$

$$\mathbf{y}_t = \mathbf{C}_t\mathbf{h}_t + D\mathbf{x}_t \tag{5}$$

where $\mathbf{h}_t \in \mathbb{R}^{d_s}$ is the hidden state, $\bar{\mathbf{A}}_t = \exp(\mathbf{A} \cdot \Delta_t)$ and $\bar{\mathbf{B}}_t = \Delta_t\mathbf{B}_t$ are the discretized transition and input matrices, $\mathbf{A} \in \mathbb{R}^{d_s \times d_s}$ is initialized via HiPPO for stability, and D is a skip connection parameter. Note that $\mathbf{A}$ and D here denote the SSM state transition matrix and skip connection parameter, respectively, distinct from the adjacency matrix and degree matrix in Section 2.2.

Mamba's selectivity arises because $\mathbf{B}_t$, $\mathbf{C}_t$, and Δ_t are dynamically generated from the input $\mathbf{x}_t$ through linear projections, allowing the model to adaptively retain or discard information. The full Mamba module has $O(n)$ time complexity, a significant advantage over the $O(n^2)$ of Transformer self-attention. SIGMA [13] applied Mamba to sequential recommendation via the G-Mamba Block, which combines forward Mamba, backward Mamba, and FE-GRU; its improvements in this work are detailed in Section 3.3.

3 Method

This section presents the proposed GSPRec model. Section 3.1 provides an overview of the architecture. Sections 3.2 and 3.3 describe the graph encoder and the improved sequential encoder. Section 3.4 details the GSCI module, and Sections 3.5 and 3.6 present the zero-damage initialization and the training objective.

3.1 Overall Architecture

The overall architecture of GSPRec is illustrated in Fig. 2. The three components of GSCI (GCG, GRA, and GCP) are embedded at the input, representation, and prediction layers of the main pipeline, respectively, to preserve graph signals in an end-to-end manner.

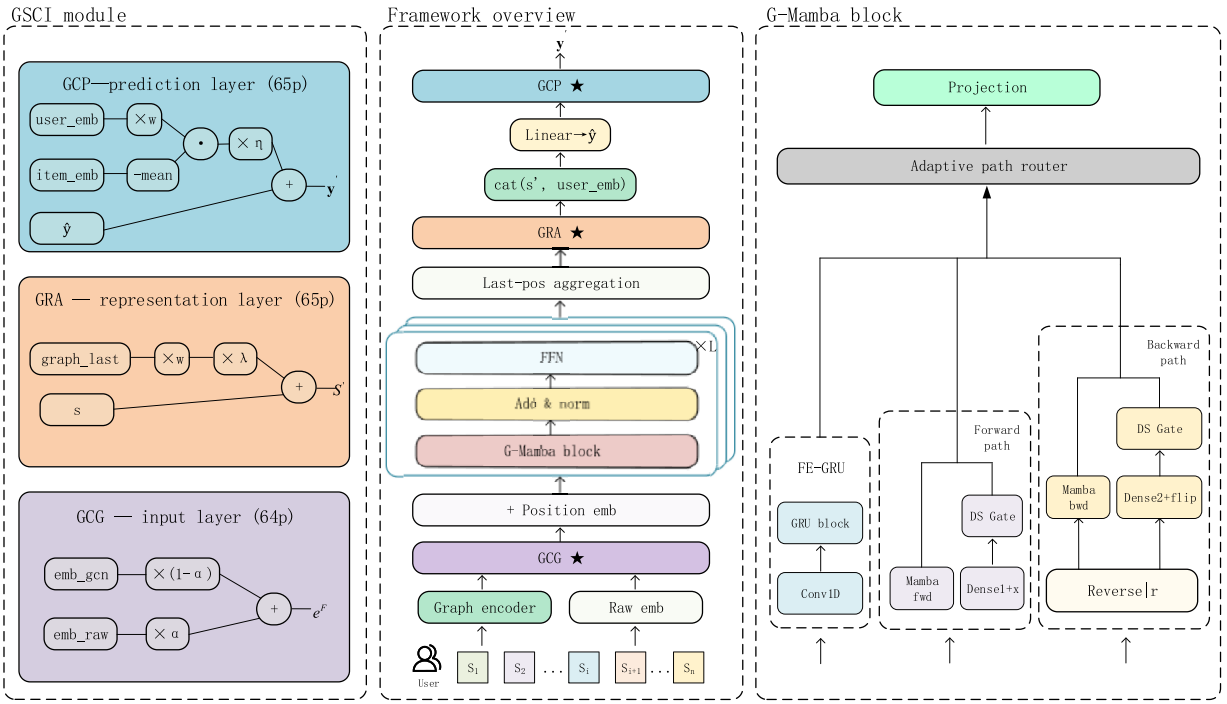

Figure 2: Overall architecture of GSPRec. (**Left**) Internal structure of the three GSCI components (GCG, GRA, GCP). (**Middle**) Framework overview showing the main pipeline with three GSCI injection points (⋆). (**Right**) Internal structure of the improved G-Mamba block with independent dual Mamba instances and adaptive path router.

Given the historical interaction sequence $S^u = \{i_1, i_2, \ldots, i_n\}$ of user u, the forward computation proceeds as follows. The graph encoder first performs multi-layer GCN propagation with external attention on the user–item interaction graph, producing item and user graph embeddings (computed once per epoch and cached). For each item in the sequence, both its graph embedding and its raw embedding are retrieved; GCG fuses them via dimension-wise bounded interpolation. The fused embeddings, combined with learnable positional encodings, are fed into the sequential encoder composed of stacked G-Mamba blocks. Last-position aggregation then extracts the user interest vector, to which GRA adds a graph skip connection from the last valid position of the graph embedding sequence. The enhanced vector is concatenated with the user graph embedding and projected through a linear layer to produce prediction scores, onto which GCP superimposes its de-meaned collaborative filtering shortcut. The training objective is detailed in Section 3.6.

3.2 Graph Encoder

3.2.1 User–Item Interaction Graph

A heterogeneous graph containing item and user nodes is constructed from the training interactions, with three edge types: item–item edges between consecutively interacted items, bidirectional user–item edges, and self-loops. After deduplication, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{(|\mathcal{I}|+|\mathcal{U}|)\times(|\mathcal{I}|+|\mathcal{U}|)}$ is symmetrically normalized to obtain $\tilde{\mathbf{A}}$.

3.2.2 GCN with External Attention

The initial embeddings are formed by concatenating a learnable item embedding matrix $\mathbf{M}_I \in \mathbb{R}^{|\mathcal{I}|\times d}$ and a user embedding matrix $\mathbf{M}_U \in \mathbb{R}^{|\mathcal{U}|\times d}$, where d denotes the embedding dimension: $\mathbf{E}^{(0)} = [\mathbf{M}_I; \mathbf{M}_U]$. The propagation at the l-th GCN layer follows the same rule as Eq. (3):

$$\tilde{\mathbf{E}}^{(l)} = \tilde{\mathbf{A}}\mathbf{E}^{(l-1)} \tag{6}$$

After each layer of GCN propagation, an External Attention (EA) mechanism is introduced to enhance node representations. Unlike standard self-attention, external attention employs two learnable external memory matrices $\mathbf{M}_K \in \mathbb{R}^{d' \times d_e}$ and $\mathbf{M}_V \in \mathbb{R}^{d_e \times d}$ as globally shared key–value pairs, where d_e is the external memory dimension. The computation is defined as:

$$\mathrm{EA}(\mathbf{X}) = \mathrm{softmax}(\mathbf{X}\mathbf{W}_d\mathbf{M}_K)\mathbf{M}_V \tag{7}$$

where $\mathbf{W}_d \in \mathbb{R}^{d \times d'}$ is a projection matrix and the softmax is computed along the d_e dimension. The complexity $O(Nd_e)$ with $N = |\mathcal{I}| + |\mathcal{U}|$ scales linearly with the number of nodes. External attention scores are added via a residual connection followed by layer normalization:

$$\mathbf{E}^{(l)} = \mathrm{LayerNorm}\left(\tilde{\mathbf{E}}^{(l)} + \mathrm{EA}\left(\tilde{\mathbf{E}}^{(l)}\right)\right) \tag{8}$$

After L_g layers of propagation, the outputs of all layers, including the initial embeddings ($l = 0$), are averaged to form the final graph embeddings:

$$\mathbf{E}^G = \frac{1}{L_g + 1}\sum_{l=0}^{L_g}\mathbf{E}^{(l)} \tag{9}$$

The resulting $\mathbf{E}^G$ is then split along the item and user dimensions to obtain $\mathbf{E}_I^G$ and $\mathbf{E}_U^G$, respectively.

3.3 Improved SSM Encoder

The sequential encoder is built upon the G-Mamba Block of SIGMA [13], with two improvements proposed to strengthen its foundational capability.

3.3.1 G-Mamba Block in SIGMA

The G-Mamba Block in SIGMA [13] uses a three-path fusion of forward Mamba (capturing look-ahead signals), backward Mamba via partial flipping (capturing retrospective context), and FE-GRU (capturing local short-term patterns). Each path's output is gated by a DS Gate combining SiLU and sigmoid activations, with a residual connection to its Mamba output.

3.3.2 Independent Dual Mamba

In the original SIGMA, the forward and backward paths share a single Mamba instance and thus use the same selective parameters $(\mathbf{B}, \mathbf{C}, \Delta)$. However, the two paths serve distinct semantic tasks—the forward path predicts the next interaction, while the backward path summarizes historical context—so parameter sharing may limit direction-specific pattern capture. To address this, independent Mamba modules are instantiated for each path:

$$\mathbf{O}_{\mathrm{fwd}} = \mathrm{Mamba}_{\mathrm{fwd}}(\mathbf{X}), \quad \mathbf{O}_{\mathrm{bwd}} = \mathrm{Mamba}_{\mathrm{bwd}}(\mathrm{PFlip}(\mathbf{X})) \tag{10}$$

where $\mathrm{PFlip}(\cdot)$ denotes the partial flip operation, which preserves the last r positions of the sequence unchanged and reverses the preceding positions.

3.3.3 Adaptive Path Router

The original SIGMA fuses the three paths using global scalar weights shared across all positions, overlooking that different positions may benefit from different signal mixes—e.g., later positions may rely more on short-term patterns from FE-GRU, while earlier positions may depend more on the long-range context from the backward Mamba. A position-aware Adaptive Path Router is designed to generate independent fusion weights for each position:

$$\mathbf{W}_{\text{route}} = \text{softmax}(\text{MLP}(\mathbf{X})) \in \mathbb{R}^{B\times L\times 3} \tag{11}$$

$$\mathbf{O}_{\text{fused}} = w_1 \odot \mathbf{O}_{\text{fwd}} + w_2 \odot \mathbf{O}_{\text{bwd}} + w_3 \odot \mathbf{O}_{\text{gru}} \tag{12}$$

where B is the batch size, L the padded sequence length, and w_1, w_2, w_3 correspond to the three channels of $\mathbf{W}_{\text{route}}$. The MLP uses two linear layers with ReLU activation. A linear projection produces the final output $\mathbf{O} = \mathbf{W}_{\text{proj}}\mathbf{O}_{\text{fused}}$, $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d\times d}$. When the MLP learns a constant mapping, the router reduces to the fixed-weight scheme of the original SIGMA, so the expressive lower bound is preserved.

3.3.4 Encoder Architecture

The complete sequential encoder comprises input Dropout, LayerNorm, and L_s stacked encoder layers. Each encoder layer consists of the improved G-Mamba Block, a residual connection, LayerNorm, and a feed-forward network (FFN):

$$\tilde{\mathbf{H}}^{(l)} = \text{LayerNorm}\left(\text{Dropout}\left(\text{GMamba}\left(\mathbf{H}^{(l-1)}\right)\right) + \mathbf{H}^{(l-1)}\right) \tag{13}$$

$$\mathbf{H}^{(l)} = \text{FFN}\left(\tilde{\mathbf{H}}^{(l)}\right) \tag{14}$$

where the FFN consists of two linear layers with GELU activation, incorporating an internal residual connection and LayerNorm. The encoder input is formed by adding the GCG-filtered item embeddings with positional embeddings: $\mathbf{H}^{(0)} = \mathbf{E}^F + \mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{n\times d}$ is a learnable positional embedding matrix. The encoder output $\mathbf{H}^{(L_s)}$ is passed through an aggregator to extract the user interest vector. By default, last-position aggregation is adopted:

$$\mathbf{s} = \mathbf{H}_n^{(L_s)} \tag{15}$$

where n is the actual (unpadded) sequence length of user u.

3.4 Graph-Sequence Collaborative Injection (GSCI)

GSCI constitutes the core innovation of this work, injecting graph signals at three distinct levels—the input, representation, and prediction layers (see Fig. 1b)—to systematically counteract graph signal dilution. The three components collectively contain only $3d + 2$ learnable parameters.

The choice of these three injection levels is not arbitrary: they are exactly the points at which graph signal is lost along the GCN-to-prediction pipeline analysed in Section 1, namely over-smoothing at the input, overwriting by nonlinear transformations inside the encoder, and the lack of a direct collaborative path at prediction. Each GSCI component is placed as the minimal intervention at the level where the corresponding loss occurs, and the layer-wise measurement in Section 4.7 confirms that representation-layer dilution is substantial. Section 4.3 further compares this complete multi-level scheme against injecting at only a subset of levels.

The *form* of injection at each level is also deliberately constrained rather than free. GCG interpolates between the raw and GCN embeddings through a bounded gate instead of an unconstrained transformation,

which preserves the embedding geometry and underlies the zero-damage initialization. GRA re-injects the original graph embedding through a gated skip connection operating in the same representation space, avoiding the space mismatch of an element-wise sequence–graph product and the parameter cost of attention-based modulation. GCP adds a de-meaned collaborative term directly to the prediction scores, providing a direct user–item path that representation-level mechanisms cannot offer.

3.4.1 GCG: Graph Confidence Gate

Multi-layer GCN message passing introduces a smoothing effect on item embeddings that is non-uniform across dimensions: moderate smoothing facilitates collaborative signal propagation, while excessive smoothing erases item distinctiveness. GCG learns the optimal degree of smoothing for each dimension through dimension-wise bounded interpolation.

For the t-th item in the sequence, let $\mathbf{e}_t^G$ and $\mathbf{e}_t^R \in \mathbb{R}^d$ denote its GCN embedding and raw (non-propagated) embedding. GCG is defined as:

$$\boldsymbol{\alpha} = \sigma(\boldsymbol{\alpha}_{\text{logit}}), \quad \boldsymbol{\alpha}_{\text{logit}} \in \mathbb{R}^d \tag{16}$$

$$\mathbf{e}_t^F = (1 - \boldsymbol{\alpha}) \odot \mathbf{e}_t^G + \boldsymbol{\alpha} \odot \text{sg}[\mathbf{e}_t^R] \tag{17}$$

where $\odot$ denotes element-wise multiplication (Hadamard product, used hereafter) and $\text{sg}[\cdot]$ the stop-gradient operator ($\text{sg}[\mathbf{x}] = \mathbf{x}$ in the forward pass, $\partial\, \text{sg}[\mathbf{x}]/\partial \mathbf{x} = 0$ in the backward pass), which detaches $\mathbf{e}_t^R$ so that gradients flow only through $\boldsymbol{\alpha}_{\text{logit}}$ and the graph encoder. Each $\alpha_j \in (0, 1)$ controls whether dimension j retains the GCN embedding ($\alpha_j \approx 0$) or reverts to the raw embedding ($\alpha_j \approx 1$). Since $\mathbf{e}_t^F$ is a convex combination of $\mathbf{e}_t^G$ and $\mathbf{e}_t^R$, the output is strictly confined within their convex hull, eliminating harmful perturbations that unconstrained transformations may introduce. GCG has d parameters; $\boldsymbol{\alpha}_{\text{logit}}$ is initialized to -5 (giving $\boldsymbol{\alpha} \approx 0.007$), so the initial output is approximately the GCN embedding.

3.4.2 GRA: Graph Residual Aggregation

After multi-layer nonlinear transformations in the sequential encoder, the collaborative signals carried by graph embeddings are progressively diluted. GRA addresses this by establishing a skip connection from the graph embeddings to the encoder output, directly supplementing undiluted graph signals into the user interest representation. Since the aggregator adopts a last-position strategy, GRA correspondingly extracts the representation at the last valid position from the graph embedding sequence:

$$\mathbf{g} = \mathbf{E}_{\text{seq},n}^G \in \mathbb{R}^d \tag{18}$$

$$\mathbf{r} = \mathbf{g} \odot \mathbf{w}_{\text{diag}}, \quad \mathbf{w}_{\text{diag}} \in \mathbb{R}^d \tag{19}$$

$$\mathbf{s}' = \mathbf{s} + \lambda \cdot \mathbf{r} \tag{20}$$

where $\mathbf{E}_{\text{seq}}^G \in \mathbb{R}^{n \times d}$ is the GCN embedding matrix for the sequence, n is the unpadded sequence length, $\mathbf{w}_{\text{diag}}$ is a dimension-wise importance weight, and λ is a learnable injection scalar. GRA uses the original graph embeddings rather than the GCG-filtered $\mathbf{E}^F$ to avoid introducing GCG bias and gradient coupling between the two components.

A gradient deadlock phenomenon is identified in naive zero-initialization. If both $\lambda = 0$ and $\mathbf{w}_{\text{diag}} = 0$, then $\mathbf{r} = \mathbf{g} \odot \mathbf{0} = 0$, yielding $\partial \mathcal{L}/\partial \lambda = (\partial \mathcal{L}/\partial \mathbf{s}')^\top \mathbf{r} = 0$ and $\partial \mathcal{L}/\partial \mathbf{w}_{\text{diag}} = \lambda \cdot (\partial \mathcal{L}/\partial \mathbf{s}') \odot \mathbf{g} = 0$: both gradients mutually require the other to be nonzero, forming a deadlock that Adam cannot escape. The proposed solution initializes λ to $\lambda_0 = 0.01$ while keeping $\mathbf{w}_{\text{diag}} = 0$, giving $\mathbf{w}_{\text{diag}}$ a nonzero gradient while preserving the zero-damage output $\mathbf{s}' = \mathbf{s}$. GRA has $d + 1$ parameters.

3.4.3 GCP: Graph Collaborative Prediction

After traversing the deep path from graph encoder to linear layer, the direct collaborative filtering affinity signal between users and items may be diluted by multiple layers of nonlinear transformations. GCP provides a shallow shortcut from graph embeddings to the output logits, computing inner products between the user graph embedding and all item graph embeddings, with de-meaning applied to the item embeddings:

$$\bar{\mathbf{e}} = \frac{1}{|\mathcal{I}|} \sum_i \mathbf{e}_i^G \tag{21}$$

$$\hat{\mathbf{e}}_i = \mathbf{e}_i^G - \bar{\mathbf{e}} \tag{22}$$

$$s_{\text{cf}}(u, i) = \frac{(\mathbf{e}_u^G \odot \mathbf{w}_{\text{gcp}})^\top \hat{\mathbf{e}}_i}{\sqrt{d}} \tag{23}$$

$$\hat{\mathbf{y}}' = \hat{\mathbf{y}} + \eta \cdot \mathbf{s}_{\text{cf}} \tag{24}$$

where $\mathbf{w}_{\text{gcp}} \in \mathbb{R}^d$ is a globally shared dimension-wise weight (initialized to ones), η is a learnable scalar, $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{I}|}$ is the main-path prediction logit, and $\mathbf{s}_{\text{cf}}$ is the vector of per-item scores from Eq. (23). The $\sqrt{d}$ normalization controls the magnitude of collaborative scores relative to the main-path logits. The de-meaning operation makes the injection magnitude data-adaptive: when item embeddings retain their individuality and deviate from the mean, GCP contributes meaningful signals; when over-smoothing drives embeddings toward the mean, $\hat{\mathbf{e}}_i \approx 0$ and GCP's contribution automatically vanishes. De-meaning also counteracts the norm inflation of high-frequency items, eliminating popularity bias. GCP has $d + 1$ parameters.

3.5 Zero-Damage Initialization

The three GSCI components share a *Zero-Damage Floor* initialization strategy: each is configured to degenerate to an identity operation at the start of training, so that the Full and Base models behave (near-) identically in the first forward pass—GCG outputs the GCN embedding, GRA adds a zero residual, and GCP adds a zero logit. Empirically, the first-epoch loss difference between Full and Base is less than 1% across all four datasets, confirming that subsequent gains arise from the graph signal injections learned by GSCI rather than from initialization perturbations.

3.6 Training and Complexity

The training objective combines a cross-entropy loss $\mathcal{L}_{CE}$ with an auxiliary alignment loss $\mathcal{L}_{\text{align}}$ that encourages the GRA-enhanced interest vector $\mathbf{s}'$ to be directionally consistent with the graph embedding of the target item:

$$\mathcal{L}_{CE} = -\log \frac{\exp(\hat{y}'_{g^*})}{\sum_i \exp(\hat{y}'_i)} \tag{25}$$

$$\mathcal{L}_{\text{align}} = 1 - \frac{\mathbf{s}'}{\|\mathbf{s}'\|} \cdot \frac{\mathbf{e}_{g^*}^G}{\|\mathbf{e}_{g^*}^G\|} \tag{26}$$

$$\mathcal{L} = \mathcal{L}_{CE} + \mu \, \mathcal{L}_{\text{align}} \tag{27}$$

where g^* is the ground-truth next item and μ is the balancing coefficient (Section 4.1.3).

GSCI introduces $3d + 2 = 194$ parameters when $d = 64$. Its complexities are $O(nd)$ for GCG, $O(d)$ for GRA, and $O(|\mathcal{I}|d)$ for GCP—the latter of the same order as the model's output layer. GSCI therefore adds negligible parameter and time cost; the GCP term does, however, raise inference peak memory in proportion

to $|\mathcal{I}|$, as analyzed empirically in Section 4.4. For industrial-scale item sets ($|\mathcal{I}| > 10^6$), this GCP computation can be restricted to a candidate subset or accelerated via approximate nearest neighbor search.

4 Experiment

The experiments address six research questions:

- **RQ1:** How does GSPRec perform compared to current state-of-the-art sequential recommendation methods?
- **RQ2:** How do the individual components of GSCI affect the performance of GSPRec?
- **RQ3:** What is the additional computational overhead introduced by the GSCI module?
- **RQ4:** Do the parameters learned by each GSCI component validate the design motivation?
- **RQ5:** How sensitive is GSPRec to key hyperparameters (alignment weight μ, number of GCN layers, embedding dimension)?
- **RQ6:** Does a direct, layer-wise measurement of graph signal retention provide empirical evidence for the dilution phenomenon and GSCI's preservation effect?

4.1 Experimental Setup

4.1.1 Datasets

GSPRec is evaluated on four real-world benchmark datasets that differ in domain, scale, and sparsity. Preprocessing follows prior work: training and test sets are split 8:2, sequences are segmented by year, and users with fewer than 3 interactions and items with fewer than 5 interactions are filtered out.

Dataset statistics are listed in Table 1. Food, Movie, and Book are food-, movie-, and book-domain datasets from Amazon Review; Douban is a comprehensive recommendation dataset from the Douban platform.

Table 1: Statistics of the datasets after preprocessing. Avg.Len denotes the average input sequence length per training sample.

Dataset	#Users	#Items	#Train	#Test	Avg.Len	Sparsity
Food	6582	14,636	29,444	12,618	10.6	99.9563%
Movie	13,724	67,161	81,455	34,909	14.7	99.9874%
Book	13,724	126,547	76,146	32,633	15.2	99.9937%
Douban	9204	61,362	84,029	21,008	8.3	99.9814%

4.1.2 Baselines

GSPRec is compared against eight baselines covering classical sequential methods, graph-enhanced methods, and the direct baseline:

(a) **GRU4Rec** [4]: an RNN-based method using gated recurrent units.
(b) **SASRec** [9]: a Transformer-based method using unidirectional self-attention.
(c) **BERT4Rec** [19]: a bidirectional Transformer with a cloze-task training objective.
(d) **FMLPRec** [20]: an MLP-based method with learnable filter-enhanced layers.
(e) **CL4SRec** [21]: a contrastive learning method using data augmentation.
(f) **LRURec** [22]: a method based on linear recurrent units.
(g) **EA-GPS** [15]: a graph-enhanced method using external attention-enhanced GCN embeddings as input to the sequential encoder.

(h) **SIGMA** [13]: a Mamba-based method using the G-Mamba block with forward–backward scanning and FE-GRU.

EA-GPS and SIGMA are selected as the graph-enhanced and SSM-based representatives because GSPRec directly adopts their graph encoder and sequential encoder architectures, enabling controlled comparisons that isolate the contribution of GSCI.

4.1.3 Implementation and Hyperparameter Settings

All models are implemented in PyTorch. Key hyperparameters: embedding dimension $d = 64$, maximum sequence length $n_{\max} = 50$, GCN layers 2, encoder layers 1, Mamba state dimension 16, convolution kernel size 4, expansion factor 2, and partial flip retention $r = 5$. Training uses Adam (learning rate 1×10^{-3}, weight decay 1×10^{-5}), batch size 2048, and cosine annealing. The alignment loss weight is $\mu = 0.1$. Early stopping with patience 3 is applied on Recall@5, evaluated every 5 epochs. Baselines are tuned following their original papers.

4.1.4 Evaluation Metrics

Standard top-K ranking metrics are adopted: Recall@K and MRR@K, with $K \in \{5, 10\}$. Following prior work [23], the full-ranking protocol is used, computing rankings over the entire item set without negative sampling.

4.2 Overall Performance Comparison (RQ1)

Table 2 reports the results of all methods across four datasets.

The following observations can be drawn from Table 2. Among classical baselines, LRURec achieves the best or second-best baseline performance on Movie, Book, and Douban, demonstrating the competitiveness of linear recurrent models. The graph-enhanced EA-GPS shows the advantage of graph-based collaborative filtering on Recall metrics (e.g., R@5 = 0.7847 on Food, surpassing all purely sequential baselines), but its MRR metrics on Movie, Book, and Douban are consistently inferior to LRURec, suggesting that single-point graph signal injection may not fully exploit the collaborative potential of graph embeddings—an observation that motivates the multi-level injection strategy of GSCI. The base model, which combines the improved encoder with the graph encoder, surpasses all external baselines on 14 of the 16 metrics; on the two Book MRR metrics it trails the strongest sequential baselines (e.g., LRURec), and GSPRec closes this gap and improves over base on every metric through GSCI.

Most importantly, GSPRec achieves the best performance across all $4 \times 4 = 16$ evaluation metrics, confirming the effectiveness of GSCI in preserving graph signals. Over five runs, the standard deviations are consistently small across all metrics (maximum 0.0024 for GSPRec), confirming the stability of these improvements. Book exhibits the largest gains (R@10 +3.16%, M@5 +0.81%), with consistent improvements on Food, Movie, and Douban. The modest improvement on Douban is consistent with its already-high baseline scores and short average sequence length of 8.3 (the shortest among the four datasets), which together leave limited headroom for further gains. Although several of these improvements are modest in absolute terms, they remain practically relevant. Under the full-ranking protocol the model ranks over the entire item set, which reaches 126,547 items on Book, so even sub-percent gains in Recall and MRR correspond to non-trivial changes in the top-K exposure that ultimately drives user-facing recommendations. The gains also require only 194 additional parameters and less than 4% extra training time, giving a favorable accuracy–cost trade-off for deployment on top of an existing encoder.

Table 2: Performance comparison across four datasets. Best score in each row is **bold**. Improv. is relative to the best external baseline (excluding base and GSPRec). Mean ± std over five runs.

Data	Metric	GRU4Rec	SASRec	BERT4Rec	FMLPRec	CL4SRec	LRURec	EA-GPS	SIGMA	Base	GSPRec	Improv.
Food	R@5	0.7558	0.7702	0.7710	0.7695	0.7734	0.7687	0.7847	0.7705	0.7880 ± 0.0003	**0.7933** ± 0.0005	+1.09%
	R@10	0.7672	0.7771	0.7776	0.7774	0.7803	0.7733	0.7940	0.7784	0.7987 ± 0.0009	**0.8057** ± 0.0006	+1.48%
	M@5	0.7343	0.7583	0.7596	0.7543	0.7601	0.7610	0.7606	0.7560	0.7756 ± 0.0002	**0.7769** ± 0.0007	+2.09%
	M@10	0.7358	0.7592	0.7605	0.7553	0.7611	0.7616	0.7618	0.7570	0.7770 ± 0.0003	**0.7786** ± 0.0006	+2.20%
Movie	R@5	0.3952	0.4084	0.4089	0.4051	0.4031	0.4149	0.4071	0.4068	0.4192 ± 0.0003	**0.4204** ± 0.0002	+1.33%
	R@10	0.4038	0.4110	0.4116	0.4087	0.4069	0.4164	0.4111	0.4103	0.4236 ± 0.0004	**0.4260** ± 0.0005	+2.29%
	M@5	0.3759	0.4019	0.4025	0.3969	0.3934	0.4122	0.3697	0.3943	0.4149 ± 0.0004	**0.4159** ± 0.0003	+0.88%
	M@10	0.3770	0.4023	0.4029	0.3974	0.3939	0.4124	0.3703	0.3947	0.4155 ± 0.0003	**0.4166** ± 0.0002	+1.01%
Book	R@5	0.4028	0.4033	0.4033	0.3878	0.3969	0.4046	0.4003	0.4040	0.4102 ± 0.0003	**0.4113** ± 0.0017	+1.66%
	R@10	0.4046	0.4040	0.4040	0.3929	0.3996	0.4050	0.4023	0.4048	0.4147 ± 0.0008	**0.4178** ± 0.0024	+3.16%
	M@5	0.3940	0.3999	0.4000	0.3652	0.3874	0.4027	0.3620	0.3969	0.3981 ± 0.0041	**0.4060** ± 0.0014	+0.81%
	M@10	0.3942	0.4000	0.4001	0.3659	0.3878	0.4028	0.3622	0.3970	0.3987 ± 0.0041	**0.4068** ± 0.0015	+1.01%
Douban	R@5	0.6865	0.6854	0.6855	0.6747	0.6832	0.6869	0.6828	0.6849	0.6897 ± 0.0005	**0.6915** ± 0.0006	+0.67%
	R@10	0.6869	0.6860	0.6861	0.6776	0.6845	0.6870	0.6848	0.6859	0.6915 ± 0.0005	**0.6959** ± 0.0010	+1.30%
	M@5	0.6741	0.6794	0.6800	0.6445	0.6767	0.6840	0.6441	0.6751	0.6876 ± 0.0003	**0.6882** ± 0.0003	+0.61%
	M@10	0.6741	0.6795	0.6801	0.6449	0.6769	0.6840	0.6444	0.6752	0.6878 ± 0.0003	**0.6888** ± 0.0003	+0.70%

4.3 Ablation Studies (RQ2)

Ablation experiments are conducted at two levels. At the GSCI level, a rigorous Leave-One-Out strategy is adopted to evaluate the necessity of each component. Five variants are examined: base (without GSCI), w/o GCG (GSCI excluding the Graph Confidence Gate), w/o GRA (GSCI excluding the Graph Residual Aggregation), w/o GCP (GSCI excluding the Graph Collaborative Prediction), and the Full model. At the encoder level, base is compared with three degradation variants: sigma_original (the default SIGMA encoder), +Dual Mamba only, and +Adaptive Router only. Table 3 presents the full results (note that Table 3 reports a representative run, distinct from the 5-run means in Table 2); a visual comparison of the GSCI ablation appears in Fig. 3.

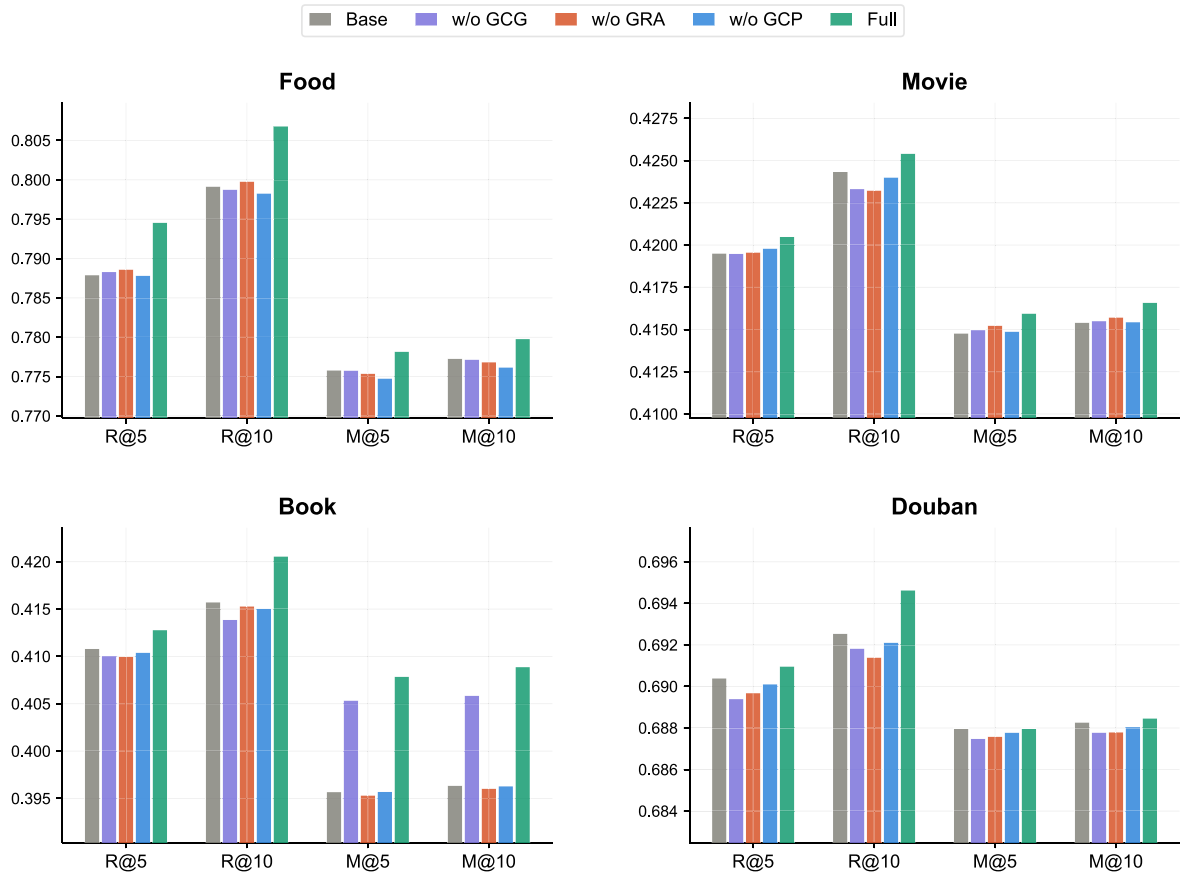

Figure 3: Performance comparison of the Leave-One-Out ablation study across four datasets. The variants include the naive Base, w/o GCG, w/o GRA, w/o GCP, and the Full model. The full model achieves the strongest overall performance in this representative run.

At the encoder level, *base* outperforms *sigma_original* on 15/16 metrics, with each modification individually surpassing *sigma_original* on most metrics—12/16 for Dual Mamba and 11/16 for the Adaptive Router. The magnitude of this encoder-level gain, however, is much smaller than the gain from GSCI. On Book R@10, for instance, the two encoder modifications together raise the score from 0.4151 to 0.4157, an absolute gain of +0.0006, whereas GSCI raises it further to 0.4207, an absolute gain of +0.0050—about an order of magnitude larger. This separation confirms that the two contributions are decoupled: the encoder improvements provide a small, complementary boost to the backbone, while the principal performance gains of GSPRec originate from the multi-level graph signal preservation mechanism of GSCI.

Table 3: Ablation studies. Upper: encoder ablation. Lower: GSCI component ablation. Best results in **bold**. Δ_1: parameters from dual Mamba; Δ_2: from the adaptive router MLP. Scores are from a representative run; see Table A1 for significance tests over five runs.

Method	#Params	Food				Movie				Book				Douban			
		R@5	R@10	M@5	M@10	R@5	R@10	M@5	M@10	R@5	R@10	M@5	M@10	R@5	R@10	M@5	M@10
sigma_original	baseline	0.7882	0.7978	0.7755	0.7768	0.4188	0.4233	0.4144	0.4150	0.4102	0.4151	0.3956	0.3963	0.6891	0.6915	0.6868	0.6871
+Dual Mamba	$+\Delta_1$	0.7878	0.7979	0.7755	0.7768	0.4196	0.4235	0.4145	0.4151	0.4108	0.4155	0.3964	0.3970	0.6894	0.6913	0.6871	0.6874
+Adaptive Router	$+\Delta_2$	0.7884	0.7987	0.7756	0.7770	0.4188	0.4230	0.4145	0.4150	0.4099	0.4150	0.3962	0.3969	0.6904	0.6925	0.6879	0.6882
Base	$\Delta_1+\Delta_2$	0.7879	0.7990	0.7758	0.7773	0.4195	0.4243	0.4148	0.4154	0.4108	0.4157	0.3957	0.3964	0.6904	0.6925	0.6880	0.6883
w/o GCG	+130	0.7883	0.7988	0.7758	0.7772	0.4195	0.4233	0.4150	0.4155	0.4100	0.4139	0.4054	0.4059	0.6894	0.6918	0.6875	0.6878
w/o GRA	+129	0.7886	0.7998	0.7754	0.7769	0.4196	0.4232	0.4152	0.4157	0.4100	0.4153	0.3953	0.3961	0.6897	0.6914	0.6876	0.6878
w/o GCP	+129	0.7878	0.7983	0.7748	0.7762	0.4198	0.4240	0.4149	0.4155	0.4104	0.4151	0.3957	0.3963	0.6901	0.6921	0.6878	0.6881
Full	+194	**0.7943**	**0.8085**	**0.7773**	**0.7791**	**0.4208**	**0.4272**	**0.4159**	**0.4167**	**0.4135**	**0.4207**	**0.4078**	**0.4087**	**0.6917**	**0.6957**	**0.6884**	**0.6890**

At the GSCI level, a rigorous Leave-One-Out ablation strategy is adopted to verify the necessity of each component. As shown in Table 3, the Full model attains the best result on all 16 metrics, and in the vast majority of cases removing any single component reduces performance. For example, on Food, removing GCP lowers Recall@5 from 0.7943 to 0.7878, roughly the *base* level, and on Douban the variant without GCG reaches only 0.6894, below the naive *base* model's 0.6904. These reductions indicate that the three components are complementary rather than redundant and jointly preserve the graph signal across the input, representation, and prediction layers. Statistical validation over five runs is provided in Table A1.

4.4 Efficiency Analysis (RQ3)

Table 4 compares the training time, memory footprint, and inference cost of base and GSPRec.

Table 4: Efficiency comparison between base and GSPRec (reported as base→GSPRec with relative overhead). GSCI adds 194 parameters (<0.006%) on every dataset. Inference cost is measured under the full-ranking protocol at batch size 2048.

Dataset	Train (s/epoch)	Train Mem (MB)	Infer (ms/query)	Infer Peak Mem (MB)
Food	2.47→2.50 (+1.1%)	3627→3932 (+8.4%)	0.0262→0.0259 (−1.2%)	401→573 (+42.9%)
Movie	15.44→15.88 (+2.8%)	9693→10,931 (+12.8%)	0.0365→0.0432 (+18.4%)	1026→2433 (+137%)
Book	15.73→15.97 (+1.5%)	11,088→13,288 (+19.8%)	0.0359→0.0406 (+13.1%)	1355→4378 (+223%)
Douban	12.46→12.91 (+3.6%)	7748→8862 (+14.4%)	0.0331→0.0351 (+6.0%)	799→2185 (+173%)

Beyond training time, Table 4 separates GSCI's cost into four parts. The parameter overhead is negligible (194 parameters, <0.006%). Training peak memory rises by 8%–20% and inference latency stays very low in absolute terms (at most 0.043 ms per query, equivalently ≥23k queries/s), with relative latency overhead between −1.2% and +18.4%. The one substantial cost is inference peak memory, which grows by 43%–223% on the larger item sets: under the full-ranking protocol GCP materializes a $[B, |\mathcal{I}|]$ collaborative-score matrix alongside the main-path logits—the $O(|\mathcal{I}|d)$ term identified in Section 3.6, which scales with the item set. This cost is controllable. In a production retrieval setting the GCP scores are restricted to a candidate subset, or computed via approximate nearest-neighbor search, both of which shrink $|\mathcal{I}|$ in this term; and even simply reducing the inference batch size shrinks the matrix directly—at a batch size of 256 the inference-memory overhead on Book falls from 223% to 1.1%. Training peak memory is, by contrast, insensitive to batch size (16.8% at batch 256 vs. 19.8% at 2048), so trainability is unaffected. GSCI introduces no new hyperparameters—all injection strengths are learned via gradient-based optimization.

4.5 Analysis of Learned GSCI Parameters (RQ4)

The parameter values learned after training are analyzed to understand the behavior of each GSCI component. Table 5 summarizes the key scalar parameters, Fig. 4 visualizes cross-dataset trends, and Fig. 5 shows the per-dimension distributions.

GCG learns differentiated $\boldsymbol{\alpha}$ values across datasets: $\alpha \approx 0.012$ on Food, 0.023 on Movie, 0.040 on Douban, and 0.075 on Book—a ~6-fold variation. The trend shows GCG automatically adjusts its reliance on GCN outputs based on the degree of graph smoothing: trusting GCN embeddings when α remains small (Food) and relying more on raw embeddings where α is larger (Book). All values remain small (<0.1), so GCG acts as fine-grained dimension-level calibration (Fig. 4a); Fig. 5a further reveals per-dimension differentiation within each dataset.

Table 5: Learned GSCI parameters across four datasets. Initialization: GCG $\boldsymbol{\alpha}_{\text{logit}} = -5$, GRA inject_scale ($\lambda$) = 0.01, GRA $\mathbf{w}_{\text{diag}} = 0$, GCP calib_scale ($\eta$) = 0, GCP $\mathbf{w}_{\text{GCP}}$ mean = 1.

Dataset	GCG $\boldsymbol{\alpha}_{\text{logit}}$	GCG $\boldsymbol{\alpha}$	GRA inject_scale	GRA $\mathbf{w}_{\text{diag}}$ Mean	GCP calib_scale	GCP $\mathbf{w}_{\text{GCP}}$ Mean
Food	−4.437	0.012	0.661	0.637	0.383	1.520
Movie	−3.736	0.023	1.257	1.234	0.770	2.021
Book	−2.515	0.075	1.440	1.419	0.759	2.043
Douban	−3.180	0.040	1.194	1.175	0.651	1.900

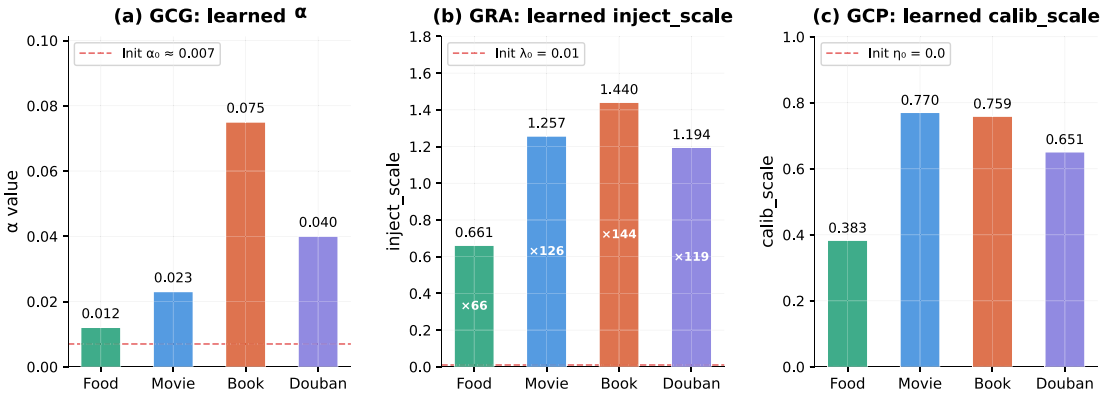

Figure 4: Learned GSCI parameters across four datasets. (**a**) GCG: $\boldsymbol{\alpha}$ increases from 0.012 (food) to 0.075 (book). (**b**) GRA: inject_scale grows 66–144× from $\lambda_0 = 0.01$. (**c**) GCP: calib_scale is positive on all datasets (0.38–0.77).

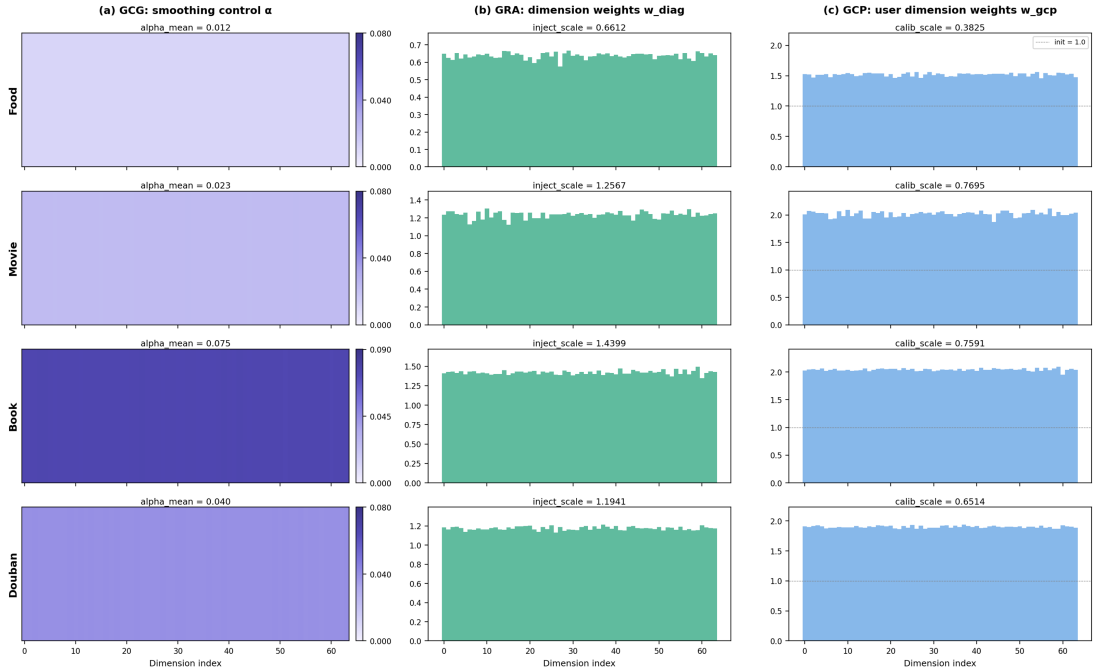

Figure 5: Per-dimension visualization of learned GSCI parameters across four datasets. (**a**) GCG smoothing control $\boldsymbol{\alpha}$—darker shading indicates stronger reliance on raw embeddings (book highest). (**b**) GRA dimension weights $\mathbf{w}_{\text{diag}}$—all dimensions learn positive weights with low variance. (**c**) GCP user dimension weights $\mathbf{w}_{\text{gcp}}$—weights grow uniformly from initialization (1.0) to 1.5–2.0.

The inject_scale of GRA grows from 0.01 to 0.66–1.44 (66–144×), showing that GRA moves well away from its zero-damage initial value and is actively used. It peaks on Book (1.44) (Fig. 4b). The growth from zero also validates the gradient deadlock analysis of Section 3.4.2: the initialization $\lambda_0 = 0.01$ successfully breaks the deadlock, allowing $\mathbf{w}_{\text{diag}}$ (mean 0.64–1.42, std. 0.02–0.04; Fig. 5b) to learn uniformly positive dimension-wise weights.

The calib_scale of GCP grows from 0 to 0.38–0.77, positive on all datasets and largest on Movie (0.77) and Book (0.76), while $\mathbf{w}_{\text{gcp}}$ mean grows from 1.0 to 1.52–2.04, confirming the effectiveness of the de-meaned collaborative shortcut and learned per-dimension scaling of user representations.

Across datasets, all three components learn non-trivial, dataset-specific strengths that move well away from their zero-damage initial values—GRA inject_scale grows 66–144× from 0.01, and GCP calib_scale becomes positive (0.38–0.77) on every dataset. This confirms that the injections are actively used rather than initialized away, i.e., training consistently finds them beneficial. The strengths vary by dataset (for example, GCG relies on raw embeddings most on Book, while GCP injects most strongly on Book and Movie), reflecting per-dataset adaptation; we do not, however, interpret these magnitudes as a cross-dataset measure of dilution severity. The direct, layer-wise evidence in Section 4.7 instead shows where dilution occurs and how each component counteracts it.

4.6 Hyperparameter Sensitivity (RQ5)

To verify that the conclusions are stable across reasonable configurations rather than artifacts of a particular setting, we vary three key hyperparameters one at a time while keeping the others at their defaults: the alignment loss weight $\mu \in \{0, 0.05, 0.1, 0.2, 0.5\}$, the number of GCN layers $L_g \in \{1, 2, 3, 4\}$, and the embedding dimension $d \in \{32, 64, 128\}$.

Fig. 6 reports the Recall@10 of base and GSPRec as each hyperparameter is varied. Three conclusions emerge. First, GSPRec improves over base across most of the swept range, and the default configuration ($\mu = 0.1$, $L_g = 2$, $d = 64$) lies in a stable region rather than at a fragile optimum. Second, performance is largely insensitive to the alignment weight μ: across the entire range each metric varies by at most about 0.004, and GSPRec stays above base on 13 to 16 of the 16 dataset–metric combinations, with only the largest value ($\mu = 0.5$) slightly degrading Book and Douban. Third, performance rises with both GCN depth and embedding dimension, so the defaults $L_g = 2$ and $d = 64$ trade a small amount of accuracy for efficiency—a larger d also inflates the inference memory discussed in Section 4.4.

Two boundary cases are worth noting for transparency. At a single GCN layer ($L_g = 1$), GSCI provides little benefit and occasionally falls just below base (for example, Book and Douban Recall@10): shallow propagation produces little over-smoothing and hence little diluted signal to preserve, which is consistent with the dilution mechanism this work targets. On Book at the largest dimension ($d = 128$), the high-capacity base already captures the collaborative signal well, and GSCI no longer adds value there. In both regimes GSPRec remains close to base, and the gains it provides emerge precisely where graph-signal dilution is non-trivial, which is the operating regime of the default configuration.

4.7 Direct Evidence of Graph Signal Dilution (RQ6)

To move beyond the inferential argument and examine the dilution phenomenon directly, we probe how much of the original graph signal is retained along the pipeline. For each test sample we compute the cosine similarity between the last-position representation at three stages—the input (after GCG), the encoder output, and after GRA—and the original GCN embedding of the corresponding item; we also report the relative magnitude of the prediction-layer GCP shortcut (gcp_ratio, the norm of the injected collaborative

term relative to the main-path logits). Table 6 summarizes the results over all test samples (input-stage retention is 1.0 by construction and is omitted).

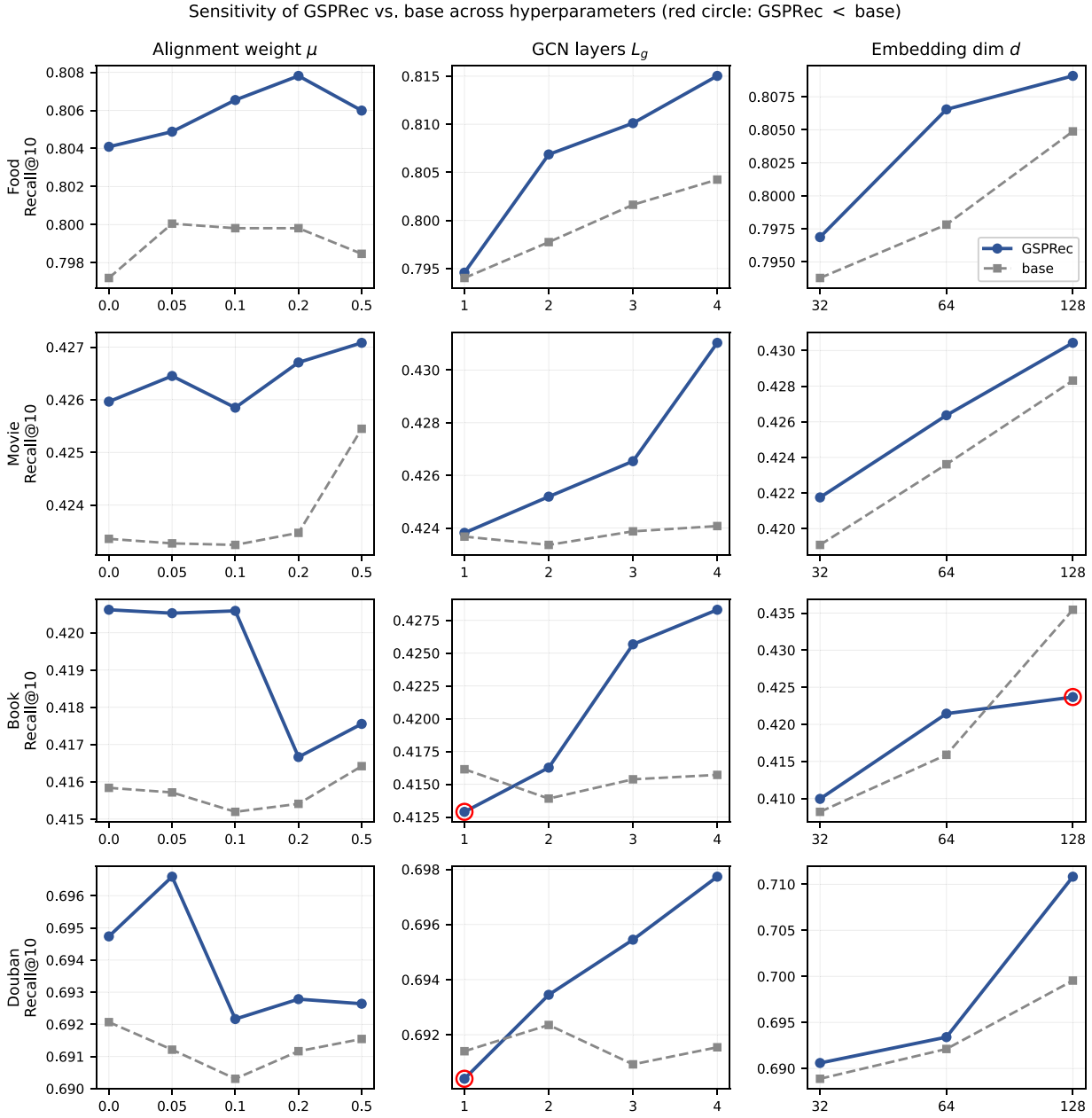

Figure 6: Sensitivity of base and GSPRec (Recall@10) to the alignment weight μ, the number of GCN layers L_g, and the embedding dimension d. Red circles mark the few settings where GSPRec falls below base.

Three observations follow. First, representation-layer dilution is real and directly measured: in the base model the retention drops from 1.0 at the input to 0.78–0.96 at the encoder output, with the largest drop on Food (1.0 → 0.781). Second, GRA restores the diluted signal: in GSPRec the after-GRA retention exceeds the base level on all four datasets, confirming that GRA actively re-injects graph signal. Third, GCP opens a direct collaborative path at the prediction layer that is entirely absent in the base model (gcp_ratio = 0 for base vs. 0.12–0.65 for GSPRec). Notably, on Douban the GSPRec encoder output is already higher than

base (0.968 vs. 0.940), so GRA there builds on an encoder representation that is itself less diluted. Together these results directly confirm that the targeted dilution occurs and that the GSCI components counteract it at their respective levels. We emphasize that the learned injection strengths (Section 4.5) are not a simple function of this representation-layer retention—they reflect how much each dataset's prediction benefits from collaborative signal—so we draw no single cross-dataset "dilution-severity" ordering, relying instead on this direct, layer-wise evidence.

Table 6: Layer-wise graph signal retention (cosine similarity to the original GCN embedding) and the relative magnitude of the GCP prediction-layer shortcut, averaged over all test samples. Input-stage retention is 1.0 by construction and is omitted.

Dataset	Base (enc. Out)	GSPRec (enc. Out)	GSPRec (After GRA)	gcp_ratio (Base)	gcp_ratio (GSPRec)
Food	0.781	0.767	0.825	0	0.117
Movie	0.921	0.908	0.959	0	0.470
Book	0.955	0.951	0.985	0	0.655
Douban	0.940	0.968	0.983	0	0.334

5 Related Work

5.1 Sequential Recommendation

Sequential recommendation methods predict users' future preferences from historical interaction sequences [24]. Early approaches relied on Markov chains, later supplanted by deep models: GRU4Rec [4] employed gated recurrent units; SASRec [9] and BERT4Rec [19] adopted unidirectional and bidirectional self-attention, respectively; and TiSASRec [25] incorporated time-interval awareness. To improve efficiency and representation quality, FMLPRec [20] replaced self-attention with filter-based layers, LRURec [22] introduced linear recurrent units, and methods such as DuoRec [26], FEARec [27], and BSARec [28] leveraged contrastive learning or frequency-domain signals. More recently, Mamba [18] enabled linear-complexity sequence modeling via input-dependent selectivity. Mamba4Rec [29] was among the first to introduce selective state space models into sequential recommendation, and SIGMA [13] subsequently applied them through the G-Mamba block. However, these Mamba-based recommenders focus on the sequence encoder alone; in particular SIGMA shares a single Mamba instance between its forward and backward paths and uses globally fixed fusion weights—limitations addressed in this work via independent dual Mamba instances and an adaptive path router, while our broader contribution lies in graph–sequence collaboration rather than sequence modeling in isolation.

5.2 Graph-Enhanced Sequential Recommendation

Graph-enhanced sequential recommendation leverages graph structures to model global collaborative signals [30]. LightGCN [14] established a simplified foundation by retaining only neighborhood aggregation; SR-GNN [31] applied gated graph neural networks to session-based recommendation; SGL [32] introduced graph augmentation and contrastive learning; and EA-GPS [15] used external attention-enhanced GCN embeddings as input to a sequential encoder. However, all these methods inject graph embeddings only at the input stage, after which the signal is progressively diluted during multi-layer processing—a problem that the proposed GSCI module is specifically designed to address. A closely related challenge is over-smoothing in GCNs [16], typically mitigated by modifying propagation (e.g., residual connections [33], DropEdge [34], JKNet [35]). In contrast, the proposed GCG addresses it from the *consumption side*—adaptively controlling

how GCN-smoothed signals are utilized at the encoder input—making GSCI compatible with arbitrary GCN architectures. A complementary line of recent work adapts the graph itself to context. CAGNN [36] introduces a context-adaptive attention mechanism that jointly incorporates spatial, temporal, and categorical factors during graph propagation, and aligns the graph- and sequence-based representations through a KL-divergence mutual-enhancement objective for next-POI recommendation. The present work is orthogonal in both target and mechanism: rather than modifying how the graph is constructed or propagated, GSPRec leaves the GCN encoder unchanged and focuses on *preserving* the resulting graph signal as it traverses the downstream sequential pipeline, injecting it at three explicit levels through bounded, low-parameter operations. Moreover, whereas CAGNN couples the two branches via a single distribution-alignment objective, GSCI additionally establishes a representation-level skip connection and a prediction-level collaborative shortcut, so that graph signals participate directly at the representation and prediction stages rather than only through an auxiliary loss.

6 Conclusion

This paper addresses the problem of graph signal dilution in graph-enhanced sequential recommendation and proposes the GSPRec model. Its core innovation, the GSCI module, preserves graph signals at the input, representation, and prediction layers through three lightweight components: GCG controls GCN smoothing via dimension-wise bounded interpolation, GRA restores diluted signals through a graph skip connection, and GCP injects collaborative signals via a de-meaned shortcut. With only 194 additional parameters and a zero-damage initialization guarantee, GSCI achieves consistent improvements across four benchmark datasets and 16 metrics, while a layer-wise probing analysis directly confirms the targeted dilution and its counteraction by the three complementary components. Additionally, encoder ablation confirms that the proposed dual Mamba instances and adaptive path router outperform the original SIGMA configuration on the majority of metrics, serving a complementary role. The main practical cost is the inference-time memory of the prediction-layer injection, which scales with the item set; restricting GCP to a candidate subset or using approximate nearest-neighbor retrieval keeps GSPRec scalable to large catalogs, which we leave to future deployment studies.

This work has three limitations. First, the three GSCI components inject graph signals at fixed levels; future work may explore mechanisms for automatically discovering optimal injection positions. Second, the experiments are primarily conducted on medium-scale datasets; scalability on larger industrial-scale datasets remains to be validated. Finally, graph signal dilution is not exclusive to sequential recommendation and may also arise in other graph–sequence joint modeling tasks; the cross-domain transferability of GSCI will be explored in future work.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Methodology and writing—original draft preparation, Yitao Yang; investigation and data curation, Peng Wu; writing—review and editing, Xiaoming Zhang; supervision, Renjie Xu; validation, Yong Zhang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data used in this work come mainly from public datasets. Specifically: the Food, Book, and Movie datasets are accessible at https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html; The Douban dataset can be downloaded from https://www.kaggle.com/datasets/fengzhujoey/douban-datasetratingreviewside-information.

Ethics Approval: The user data used in this study were sourced from publicly available datasets and comply with relevant privacy protection regulations.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Per-Metric Ablation Significance

To complement the aggregate counts in Section 4.3, Table A1 reports the per-metric paired t-test of the Full model against each leave-one-out variant over five runs.

Table A1: Per-metric paired t-test of the Full model vs. each leave-one-out variant over five runs. $^{*}/^{**}/^{***}$: Full significantly higher at $p < 0.05/0.01/0.001$; "–": not significant; ↓: the variant is significantly higher than Full in the five-run average. Full significantly outperforms the *w/o* GCP, *w/o* GRA, and *w/o* GCG variants on 12, 5, and 5 of the 16 metrics, respectively.

Dataset	Metric	vs. w/o GCP	vs. w/o GRA	vs. w/o GCG
Food	R@5	***	–	*
	R@10	***	–	–
	M@5	**	↓	–
	M@10	**	↓	–
Movie	R@5	**	–	–
	R@10	***	↓	–
	M@5	**	***	–
	M@10	**	***	–
Book	R@5	↓	–	–
	R@10	–	–	–
	M@5	–	**	–
	M@10	–	**	–
Douban	R@5	*	*	*
	R@10	**	–	**
	M@5	*	–	*
	M@10	**	–	*

References

1. Huang W, Li Z. Recommendation systems: a comparative study of traditional models and pre-trained model approaches. Softw Guide. 2025;24(2):204–10.
2. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;42(8):30–7. doi:10.1109/mc.2009.263.
3. He R, Kang WC, McAuley J. Translation-based recommendation: a scalable method for modeling sequential behavior. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI); 2018 Jul 13–19; Stockholm, Sweden. p. 5264–8. doi:10.24963/ijcai.2018/734.
4. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks. arXiv:1511.06939. 2015.
5. Choe B, Kang T, Jung K. Recommendation system with hierarchical recurrent neural network for long-term time series. IEEE Access. 2021;9:72033–9. doi:10.1109/access.2021.3079922.
6. Chen M, Ma T, Zhou X. CoCNN: co-occurrence CNN for recommendation. Expert Syst Appl. 2022;195(4):116595. doi:10.1016/j.eswa.2022.116595.

7. Alrashidi M, Ibrahim R, Selamat A. Hybrid CNN-based recommendation system. Baghdad Sci J. 2024;21(2):40. doi:10.21123/bsj.2024.9756.
8. Chen X, Xu H, Zhang Y, Tang J, Cao Y, Qin Z, et al. Sequential recommendation with user memory networks. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM); 2018 Feb 5–9; Marina Del Rey, CA, USA. p. 108–16. doi:10.1145/3159652.3159668.
9. Kang WC, McAuley J. Self-attentive sequential recommendation. In: Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM); 2018 Nov 17–20; Singapore. p. 197–206. doi:10.1109/icdm.2018.00035.
10. Zhou K, Wang H, Zhao WX, Zhu Y, Wang S, Zhang F, et al. S3-rec: self-supervised learning for sequential recommendation with mutual information maximization. Proceedings of the 29th ACM International Conference on Information & Knowledge Management; 2020 Oct 19–23; Virtual. p. 1893–902. doi:10.1145/3340531.3411954.
11. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;30:6000–10. doi:10.65215/ctdc8e75.
12. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT); 2019 Jun 2–7; Minneapolis, MN, USA. p. 4171–86.
13. Liu Z, Liu Q, Wang Y, Wang W, Jia P, Wang M, et al. SIGMA: selective gated mamba for sequential recommendation. Proc AAAI Conf Artif Intell. 2025;39(12):12264–72. doi:10.1609/aaai.v39i12.33336.
14. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. 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; 2020 Jul 25–30; Virtual. p. 639–48. doi:10.48550/arxiv.2002.02126.
15. Zhang J, Li C, Zhao Z. Lightweight yet efficient: an external attentive graph convolutional network with positional prompts for sequential recommendation. ACM Trans Inf Syst. 2025;43(3):1–25. doi:10.1145/3719343.
16. Li Q, Han Z, Wu XM. Deeper insights into graph convolutional networks for semi-supervised learning. Proc AAAI Conf Artif Intell. 2018;32(1):3538–45. doi:10.1609/aaai.v32i1.11604.
17. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907. 2016.
18. Gu A, Dao T. Mamba: linear-time sequence modeling with selective state spaces. arXiv:2312.00752. 2023.
19. Sun F, Liu J, Wu J, Pei C, Lin X, Ou W, et al. BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; 2019 Nov 3–7; Beijing, China. p. 1441–50. doi:10.1145/3357384.3357895.
20. Zhou K, Yu H, Zhao WX, Wen JR. Filter-enhanced MLP is all you need for sequential recommendation. In: Proceedings of the ACM Web Conference 2022; 2022 Apr 25–29; Virtual. p. 2388–99. doi:10.1145/3485447.3512111.
21. Xie X, Sun F, Liu Z, Wu S, Gao J, Zhang J, et al. Contrastive learning for sequential recommendation. In: Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE); 2022 May 9–12; Kuala Lumpur, Malaysia. p. 1259–73. doi:10.1109/icde53745.2022.00099.
22. Yue Z, Wang S, Shao Y, Nguyen QVH, Yin H. Linear recurrent units for sequential recommendation. arXiv:2310.02367. 2023.
23. Boka TF, Niu Z, Neupane RB. A survey of sequential recommendation systems: techniques, evaluation, and future directions. Inf Syst. 2024;125(5):102427. doi:10.1016/j.is.2024.102427.
24. Wang S, Hu L, Wang Y, Cao L, Sheng QZ, Orgun MA. Sequential recommender systems: challenges, progress and prospects. arXiv:2001.04830. 2019.
25. Li J, Wang Y, McAuley J. Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th International Conference on Web Search and Data Mining; 2020 Feb 3–7; Houston, TX, USA. p. 322–30. doi:10.1145/3336191.3371786.
26. Qiu R, Huang Z, Yin H, Wang Z. Contrastive learning for representation degeneration problem in sequential recommendation. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining; 2022 Feb 21–25; Virtual. p. 813–23. doi:10.1145/3488560.3498433.

27. Du X, Yuan H, Zhao P, Qu J, Zhuang F, Liu G, et al. Frequency enhanced hybrid attention network for sequential recommendation. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2023 Jul 23–27; Taipei, Taiwan. p. 78–88. doi:10.1145/3539618.3591689.
28. Shin Y, Choi J, Wi H, Park N. An attentive inductive bias for sequential recommendation beyond the self-attention. Proc AAAI Conf Artif Intell. 2024;38(8):8984–92. doi:10.1609/aaai.v38i8.28747.
29. Liu C, Lin J, Wang J, Liu H, Caverlee J. Mamba4Rec: towards efficient sequential recommendation with selective state space models. arXiv:2403.03900. 2024.
30. Liu Y, Xia L, Huang C. SelfGNN: self-supervised graph neural networks for sequential recommendation. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2024 Jul 14–18; Washington, DC, USA. p. 1609–18. doi:10.1145/3626772.3657716.
31. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T. Session-based recommendation with graph neural networks. Proc AAAI Conf Artif Intell. 2019;33(1):346–53. doi:10.1609/aaai.v33i01.3301346.
32. Wu J, Wang X, Feng F, He X, Chen L, Lian J, et al. Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2021 Jul 11–15; Virtual. p. 726–35. doi:10.1145/3404835.3462862.
33. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. p. 770–8. doi:10.1109/cvpr.2016.90.
34. Xie Y, Li J, Zhang S. Adaptive node similarity for DropEdge. Neurocomputing. 2025;626(9):129574. doi:10.1016/j.neucom.2025.129574.
35. Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi KI, Jegelka S. Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th International Conference on Machine Learning (ICML); 2018 Jul 10–15; Stockholm, Sweden. p. 5453–62.
36. Lei Y, Shen L, Sun Z, He T. Context-adaptive graph neural networks for next POI recommendation. Electron Commer Res Appl. 2026;77(3):101597. doi:10.1016/j.elerap.2026.101597.