GRAPH PROCESS NEURONS FOR SEQUENTIAL RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural network (GNN)-based methods have achieved remarkable performance on sequential recommendation tasks. They compress collaborative relations and item transitions into compact node representations and propagate them to neighboring nodes, thereby learning user interests from collaborative signals. However, such representations are static summaries of historical interactions. User preference is inherently time-varying, and this compression overlooks how interest rises and fades between observed interactions. To address this limitation, we propose a **G**raph **Pr**ocess **N**euron (GPrN) for sequential recommendation. It represents the user's history as a continuous preference process over real time and lets each candidate induce its own reading of it. Specifically, we first organize the visible interactions into a query-specific preference process along an observable temporal coordinate, which characterizes how user preference evolves over time. Then, each candidate induces branch-specific scalar reading kernels over complementary temporal views of this process, allowing candidate-dependent evidence to contribute at different scales. A factorized implementation over a finite cosine basis supports matrix-based full ranking. The final score augments structural and sequential preferences with a candidate-specific process residual controlled by a bounded query-adaptive gate. Extensive experiments on three real-world datasets show that GPrN outperforms nine state-of-the-art baselines on sequential recommendation tasks. The code and datasets are available at `https://anonymous.4open.science/r/GPrN`.

1 INTRODUCTION

User interest rarely stays constant over time (Li et al., 2026). After buying a phone, a user may want a matching case within days, and the interest fades once the case is bought. Capturing such dynamics is essential for sequential recommendation (SR), which predicts a user's next interaction from historical interactions and supports many online services (Zhang et al., 2025b; Zhao et al., 2025). In practice, each interaction is recorded with a real-world timestamp, so the temporal gaps between successive interactions are observable and informative (Zhao et al., 2026).

In the early explorations of SR, researchers utilized Recurrent Neural Networks (RNNs) (Hidasi et al., 2016) to encode user histories. However, recurrent state compression makes long-range dependencies difficult to preserve in lengthy histories. Subsequently, self-attention attracted significant research interest. SASRec (Kang & McAuley, 2018) and BERT4Rec (Sun et al., 2019) capture sequential patterns directly, and FEARec (Du et al., 2023) further refines the attention with frequency-enhanced representations. Beyond interaction order, time-aware models inject temporal intervals into sequential scoring (Li et al., 2020), and target-aware models adapt the historical representation to each candidate (Zhou et al., 2018). Nevertheless, these methods primarily model each interaction sequence independently and do not explicitly exploit collaborative structures across users and items. Recently, graph-based SR methods construct sequential graphs over user-item interactions and item-item transitions (Chang et al., 2021; Xia et al., 2023), and such methods have achieved remarkable performance on SR tasks (Zhang et al., 2022). The underlying principle is to compress each node's history into one embedding vector and propagate it to neighboring nodes, so that user interests are learned from collaborative signals (Wu et al., 2019).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

On such sequential graphs, a graph neural network (GNN) (Kipf & Welling, 2017) updates each node representation by aggregating messages from its neighbors (Zhang et al., 2025a). The historical information is thus compressed into a fixed-dimensional node representation, which serves as a shared structural summary for candidate scoring. However, user preference is inherently time-varying: interest rises and fades between observed interactions, and different candidates may value different parts of the same history. Such pair-specific dependence is unavailable to a single pooled history vector. We therefore ask: "Can we model the node's historical information as a continuous preference process over real time, rather than pooling it into a static vector?" Our answer is to organize the visible interaction history into a query-specific preference process on the temporal axis. Each candidate reads this process with its own kernel and extracts candidate-specific temporal evidence from the same history. To this end, we propose the **G**raph **Pr**ocess **N**euron (GPrN), which introduces candidate-conditioned process reading over graph-aware interaction histories into SR. Then, we face the following two challenges:

(1) Process construction from irregularly timed interactions. Real-world logs record discrete interactions (i_j, t_j) with irregular temporal gaps, rather than dense observations of a continuous signal. A process-based model, however, needs the preference to be defined over a continuous temporal coordinate. The model must therefore reconstruct a smooth process from a sparse and uneven interaction sequence. Moreover, different queries arrive at different times and expose different history prefixes. Besides, the construction should be query-specific and use only observable timestamps.

(2) Candidate-conditioned functional reading under full ranking. At inference time, an SR model scores every item as a potential next interaction. Since different candidates may emphasize different temporal regions of the same history, the reading of the process should vary across candidates. A straightforward solution is to learn a separate neural reader for each user-item pair. Such a solution is computationally prohibitive because the number of user-item pairs grows with the catalog size. The model therefore needs candidate-specific reading that remains compatible with matrix-based full ranking, i.e., scoring all items with shared matrix operations.

GPrN addresses these two challenges with the following design. Specifically, it first constructs a sequential graph to learn structural context from collaborative interactions and global item transitions. Then, GPrN employs a lightweight GRU-based sequential encoder to summarize the order-sensitive visible history. Subsequently, we organize the same history into complementary coarse, selective, and local process views along the observable temporal coordinate. Each view is equipped with a query- and candidate-conditioned scalar reading kernel to capture temporal evidence at a different scale. The resulting process scores are adaptively fused and injected as a bounded residual over the structural-sequential backbone. Finally, we expand the reading kernels with a finite orthonormal cosine basis and factorize their coefficients, retaining compatibility with matrix-based full ranking.

Our main contributions are summarized as follows:

- We formulate sequential recommendation from a process-valued perspective and present GPrN, where each candidate performs functional readings over complementary views of the same query-specific preference process.
- We devise a multi-scale residual process reader, which reads coarse, selective, and local process views with candidate-conditioned kernels and injects the branch scores as a bounded residual.
- We establish key properties of the functional reading, including exact coefficient contraction, perturbation stability, and ideal-kernel approximation.
- Extensive experiments on three real-world datasets show that GPrN outperforms nine state-of-the-art baselines on SR tasks.

2 Related Work

Sequential Recommendation. Early explorations utilized Markov chains (Rendle et al., 2010) and recurrent neural networks (Hidasi et al., 2016) to capture transition patterns. Subsequently, self-attention became the dominant sequence encoder (Kang & McAuley, 2018; Sun et al., 2019), and contrastive regularization further refines the learned sequence representations (Qiu et al., 2022). Beyond interaction order, time-aware variants incorporate temporal intervals between interactions

(Li et al., 2020), and target-aware variants condition the historical aggregation on each candidate (Zhou et al., 2018; 2019). However, these methods learn each interaction sequence in isolation, so the collaborative structures across users and items remain unexploited. Recently, GNN-based SR methods propagate messages on sequential graphs built over user-item interactions and item-item transitions (Chang et al., 2021; Zhang et al., 2022). For example, Wu et al. (2019) represent session transitions as graphs, and Zhang et al. (2025a) construct a unified sequential graph. Xia et al. (2023) further model multi-behavior temporal dependencies with a graph transformer, and Liu et al. (2026) employ self-supervised hypergraph learning to capture intent tendencies and sequential dependencies. However, message passing still pools each node's history into one static vector before scoring, so the time-varying dependence between the user and each candidate is lost.

Process Neural Networks. Process neural networks (PNNs) aim to model systems whose inputs and weights are both time-varying functions. He & Xu (2010) integrate time-varying inputs against learnable process weight functions and reduce the resulting functional learning problem through basis expansion, following functional data analysis (Ramsay & Silverman, 2005). Subsequently, Conditional Neural Processes learn distributions over functions from context observations (Garnelo et al., 2018), and operator-learning models such as DeepONet (Lu et al., 2021) and the Fourier Neural Operator (Li et al., 2021) learn mappings between function spaces. More recently, Boschi et al. (2024) combine functional representations with graph learning for longitudinal multimodal data. However, these models pursue generic function estimation with query-agnostic processes and fixed readings. GPrN instead constructs a query-specific preference process from graph-aware interaction representations and lets each candidate parameterize its own reading kernel over it.

3 METHODOLOGY

3.1 PRELIMINARIES

Notations. Let $\mathcal{U}$ and $\mathcal{I}$ be the sets of users and items, respectively. Each user $u \in \mathcal{U}$ is associated with a chronological interaction sequence $\mathcal{S}_u = [(i_1, t_1), \ldots, (i_{N_u}, t_{N_u})]$, where $i_j \in \mathcal{I}$ denotes the j-th interacted item and t_j records its timestamp. Furthermore, we define the visible prefix at the q-th prediction as $\mathcal{S}_u^{(q)} = [(i_1, t_1), \ldots, (i_{q-1}, t_{q-1})]$, containing the first $q-1$ interactions of $\mathcal{S}_u$.

Problem Definition. Given the visible prefix $\mathcal{S}_u^{(q)}$, the task of sequential recommendation is to predict the next item i_q that user u is most likely to interact with. The score of each candidate item i is estimated by a scoring function $f(\cdot)$:

$$\hat{y}_{ui}^{(q)} = f\left(\mathcal{S}_u^{(q)},\, i\right), \tag{1}$$

where all candidate items are ranked by their scores. In GPrN, $f(\cdot)$ is implemented with a structural-sequential backbone and a candidate-conditioned process reader.

Sequential Graph Construction. The interactions of all users are organized into a training-only sequential graph $\mathcal{G}_s = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$. The graph contains bidirectional user-item edges for collaborative relations and directed $i_j \to i_{j+1}$ edges for global item transitions.

3.2 FRAMEWORK OF GPRN

The framework of GPrN is shown in Figure 1. GPrN consists of four key components, i.e., the structural-sequential backbone, the multi-scale query-specific preference process, the candidate-conditioned graph process neuron, and prediction and optimization. In general, the backbone encodes collaborative structure and the order-sensitive visible history. The latter two components form a residual process reader, which organizes the same timestamped history into coarse, selective, and local continuous preference processes and reads them with candidate-conditioned kernels. We elaborate on the above components in subsequent sections.

3.3 STRUCTURAL-SEQUENTIAL BACKBONE

The structural-sequential backbone takes $\mathcal{G}_s$ and the visible prefix as input and learns node-level context for scoring. It consists of two parts, i.e., graph propagation and sequential state modeling.

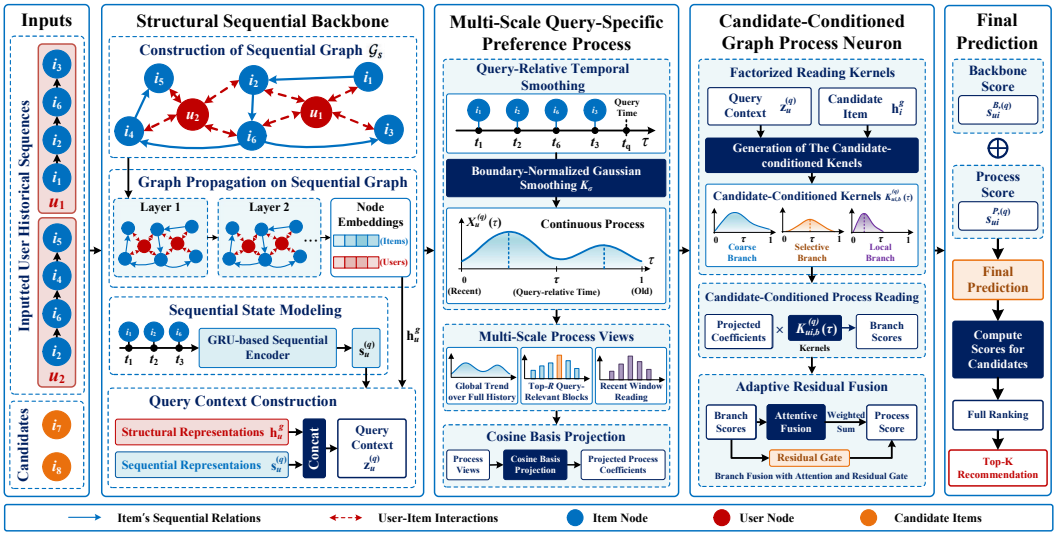

Figure 1: Overall framework of Graph Process Neuron (GPrN), where u_1, u_2 are two different users, i_1 to i_6 are interacted items, $\{i_7, i_8\}$ are two candidate items.

Graph Propagation. GPrN employs LightGCN (He et al., 2020) to encode collaborative structure over $\mathcal{G}_s$. The node representations are propagated by neighborhood aggregation:

$$\mathbf{H}^{(l+1)} = \hat{\mathbf{A}}\mathbf{H}^{(l)}, \qquad \mathbf{h}_v^g = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{h}_v^{(l)}, \tag{2}$$

where $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, L is the propagation depth, and $\mathbf{h}_v^g \in \mathbb{R}^d$ is the graph-aware representation of node v averaged over all layers.

For example, a user node aggregates messages from its interacted items, and an item node aggregates messages from its interacting users and its sequential successors.

Sequential State Modeling. Since the graph propagation does not preserve the exact order of the current prefix, we then feed $\{\mathbf{h}_{i_j}^g\}_{j<q}$ into a lightweight GRU (Hidasi et al., 2016). The recurrent update summarizes the growing prefix into a single state in one pass, which preserves the interaction order and adds little overhead. The capacity for query-specific modeling is thus concentrated in the process reader rather than in the sequential encoder. Let $\mathbf{g}_u^{(q)}$ be its last output. The sequential state and query context are:

$$\begin{aligned}\mathbf{s}_u^{(q)} &= \mathrm{LN}\left(\mathbf{g}_u^{(q)} + \eta\mathbf{h}_{i_{q-1}}^g\right), \qquad \eta = 2\ \mathrm{sigmoid}(\xi),\\ \mathbf{z}_u^{(q)} &= \left[\mathbf{h}_u^g \,\|\, \mathbf{s}_u^{(q)}\right] \in \mathbb{R}^{2d},\end{aligned} \tag{3}$$

where $\mathrm{LN}(\cdot)$ denotes layer normalization, and ξ is a learnable scalar whose sigmoid rescaling $\eta \in (0, 2)$ controls the last-item residual.

Such a simple yet efficient backbone encodes both collaborative structure and order-sensitive sequential context. The following two components then organize the same timestamped history into continuous preference processes.

3.4 Multi-Scale Query-Specific Preference Process

This component organizes the visible history into query-specific preference processes over real time. It proceeds in three steps, i.e., query-relative temporal smoothing, multi-scale view construction, and cosine basis projection.

Query-Relative Temporal Smoothing. We anchor each query at its latest visible interaction $t_u^{(q)} = t_{q-1}$ and map every historical interaction onto a shared query-relative coordinate:

$$\tau_{u,j}^{(q)} = \min\left\{ \frac{\log\left(1 + t_u^{(q)} - t_j\right)}{\log(1 + T_{\max})}, 1 \right\} \in [0, 1], \tag{4}$$

where $T_{\max}$ is estimated from training interactions. Thus, $\tau = 0$ represents the most recent observation and larger values represent increasingly older interactions.

We convert discrete interactions into a continuous process using the boundary-normalized Gaussian smoothing function:

$$\kappa_\sigma(\tau; \mu) = \frac{\exp\left(-\frac{(\tau-\mu)^2}{2\sigma^2}\right)}{\int_0^1 \exp\left(-\frac{(s-\mu)^2}{2\sigma^2}\right) ds}, \tag{5}$$

where $\mu \in [0, 1]$ is an interaction location and σ controls temporal smoothness.

Multi-Scale Process Views. A single temporal scale may miss either long-range or query-specific evidence. GPrN therefore constructs three views from the same visible history. Specifically, we divide the prefix into contiguous blocks $\{\mathcal{B}_m\}_{m=1}^M$ of at most B interactions. Each block is summarized by its mean representation $\bar{\mathbf{h}}_m = |\mathcal{B}_m|^{-1} \sum_{j \in \mathcal{B}_m} \mathbf{h}_{i_j}^g$ and mean coordinate $\bar{\tau}_m = |\mathcal{B}_m|^{-1} \sum_{j \in \mathcal{B}_m} \tau_{u,j}^{(q)}$. The selective branch computes a query-block similarity $e_m = \mathrm{norm}(\mathbf{W}_{\mathrm{sel}} \mathbf{z}_u^{(q)})^\top \mathrm{norm}(\bar{\mathbf{h}}_m)$, where $\mathrm{norm}(\cdot)$ denotes ℓ_2-normalization and keeps the top-R blocks $\mathcal{T}_R$, whose softmax-normalized weights are denoted by π_m. The local branch reads $\mathcal{L}_W$, which contains the most recent $\min(W, q-1)$ interactions. The three process views are:

$$\begin{aligned}
\mathbf{X}_{u,\mathrm{c}}^{(q)}(\tau) &= \frac{1}{M} \sum_{m=1}^{M} \kappa_\sigma(\tau; \bar{\tau}_m) \bar{\mathbf{h}}_m, \\
\mathbf{X}_{u,\mathrm{s}}^{(q)}(\tau) &= \sum_{m \in \mathcal{T}_R} \frac{\pi_m}{|\mathcal{B}_m|} \sum_{j \in \mathcal{B}_m} \kappa_\sigma(\tau; \tau_{u,j}^{(q)}) \mathbf{h}_{i_j}^g, \\
\mathbf{X}_{u,\mathrm{l}}^{(q)}(\tau) &= \frac{1}{|\mathcal{L}_W|} \sum_{j \in \mathcal{L}_W} \kappa_\sigma(\tau; \tau_{u,j}^{(q)}) \mathbf{h}_{i_j}^g,
\end{aligned} \tag{6}$$

where c, s, and l denote coarse, selective, and local views, respectively.

Specifically, the coarse branch preserves global temporal structure, the selective branch retains query-relevant blocks at interaction resolution, and the local branch emphasizes the recent neighborhood.

Cosine Basis Projection. We represent each continuous process with K coefficients on an orthonormal cosine basis. The basis functions are $\phi_1(\tau) = 1$ and $\phi_k(\tau) = \sqrt{2}\cos((k-1)\pi \cdot \tau)$ for $k \geq 2$:

$$\mathbf{a}_{u,b,k}^{(q)} = \int_0^1 \phi_k(\tau) \mathbf{X}_{u,b}^{(q)}(\tau) d\tau, \qquad b \in \mathcal{R} = \{\mathrm{c}, \mathrm{s}, \mathrm{l}\}, \tag{7}$$

where $\mathbf{a}_{u,b,k}^{(q)} \in \mathbb{R}^d$ is the k-th process coefficient.

Since each process is a weighted sum of smoothing functions, the required scalar integrals $\int_0^1 \phi_k(\tau) \kappa_\sigma(\tau; \mu) d\tau$ depend only on μ for fixed σ. We precompute them on a dense grid and use linear interpolation during training and ranking.

3.5 Candidate-Conditioned Graph Process Neuron

The graph process neuron scores each candidate by reading the preference processes from its own perspective. It proceeds in three steps, i.e., kernel factorization, candidate-conditioned process reading, and adaptive residual fusion.

Factorized Reading Kernels. For each branch b, GPrN induces a candidate-conditioned scalar reading kernel through factorized query-side and item-side coefficients:

$$\begin{aligned} \mathbf{p}_{u,b}^{(q)} &= \tanh\Big(\mathbf{W}_b^Q \mathbf{z}_u^{(q)} + \mathbf{b}_b^Q\Big), \\ \mathbf{r}_{i,b} &= \tanh\big(\mathbf{W}_b^I \mathbf{h}_i^g + \mathbf{b}_b^I\big), \\ \mathcal{K}_{ui,b}^{(q)}(\tau) &= \sum_{k=1}^{K} p_{u,b,k}^{(q)} r_{i,b,k} \phi_k(\tau), \end{aligned} \tag{8}$$

where $\mathbf{p}_{u,b}^{(q)}, \mathbf{r}_{i,b} \in \mathbb{R}^K$ are the query-side and item-side factors generated from $\mathbf{z}_u^{(q)}$ and $\mathbf{h}_i^g$, respectively.

The $\tanh$ generators retain signed coefficients and bound every basis coefficient by one.

Candidate-Conditioned Process Reading. The branch-specific functional reading is denoted as:

$$\begin{aligned} \mathbf{m}_{u|i,b}^{(q)} &= \int_0^1 \mathcal{K}_{ui,b}^{(q)}(\tau) \mathbf{X}_{u,b}^{(q)}(\tau) d\tau \\ &= \sum_{k=1}^{K} p_{u,b,k}^{(q)} r_{i,b,k} \mathbf{a}_{u,b,k}^{(q)}, \\ s_{ui,b}^{(q)} &= \Big(\mathbf{m}_{u|i,b}^{(q)}\Big)^\top \mathbf{h}_i^g, \end{aligned} \tag{9}$$

where $\mathbf{m}_{u|i,b}^{(q)} \in \mathbb{R}^d$ is the candidate-conditioned process message and $s_{ui,b}^{(q)}$ is the branch score.

The second equality is exact for the modeled finite-basis kernel once the process coefficients are obtained. Moreover, the factorization separates query-side and item-side terms. All candidates in each branch are therefore scored by one matrix multiplication with cached item-side factors.

Adaptive Residual Fusion. The three branch scores may have different numerical scales. We first standardize each branch over the candidate set:

$$\bar{s}_{ui,b}^{(q)} = \frac{s_{ui,b}^{(q)} - \mu_{u,b}^{(q)}}{\sqrt{v_{u,b}^{(q)} + \epsilon}}, \tag{10}$$

where $\mu_{u,b}^{(q)}$ and $v_{u,b}^{(q)}$ are the query-wise mean and variance, and ϵ is a small constant.

A query-dependent gate then combines the three branches:

$$\begin{aligned} \boldsymbol{\omega}_u^{(q)} &= \mathrm{softmax}\Big(\mathbf{W}_g \mathbf{z}_u^{(q)} + \mathbf{b}_g\Big), \\ s_{ui}^{P,(q)} &= \sum_{b \in \mathcal{R}} \omega_{u,b}^{(q)} \bar{s}_{ui,b}^{(q)}, \end{aligned} \tag{11}$$

where $\boldsymbol{\omega}_u^{(q)}$ is the branch-weight vector.

We further bound the total process contribution with a query-dependent residual coefficient:

$$\alpha_u^{(q)} = \alpha_{\max} \,\mathrm{sigmoid}\Big(\mathbf{w}_\alpha^\top \mathbf{z}_u^{(q)} + b_\alpha\Big), \tag{12}$$

where $0 < \alpha_u^{(q)} < \alpha_{\max}$.

This adaptive residual design lets GPrN vary both the preferred temporal scale and the strength of process evidence across queries.

3.6 Prediction and Optimization

The backbone score combines graph-based user preference and the order-sensitive sequential state:

$$s_{ui}^{B,(q)} = (\mathbf{h}_u^g)^\top \mathbf{h}_i^g + (\mathbf{s}_u^{(q)})^\top \mathbf{h}_i^g. \tag{13}$$

The final score adds the residual process signal,

$$\hat{y}_{ui}^{(q)} = s_{ui}^{B,(q)} + \lambda_p \alpha_u^{(q)} s_{ui}^{P,(q)}, \tag{14}$$

where λ_p is the global process weight.

Setting $\lambda_p = 0$ yields the matched structural-sequential backbone. We optimize full-softmax cross-entropy over the item set:

$$\mathcal{L} = -\sum_{\text{training queries}} \log \frac{\exp(\hat{y}_{ui^\star}^{(q)})}{\sum_{i \in \mathcal{I}} \exp(\hat{y}_{ui}^{(q)})}, \tag{15}$$

where $i^\star$ is the ground-truth next item.

During ranking, GPrN caches the graph representations and item-side process factors, so full ranking remains matrix-based. The additional ranking cost of the process reader is $\mathcal{O}(|\mathcal{R}|Kd|\mathcal{I}|)$ per query.

4 Experiments

We conduct experiments to answer four **R**esearch **Q**uestion**s** (RQs): how GPrN performs compared with state-of-the-art sequential and graph-based recommenders (**RQ1**); how the multi-scale process branches and their adaptive fusion contribute to GPrN (**RQ2**); how GPrN responds to its process-reader hyperparameters and how they shape the accuracy trade-off (**RQ3**); and what overhead the residual process reader introduces in parameters, training time, and full-ranking latency (**RQ4**).

4.1 Experimental Setup

Datasets. We use Amazon-Beauty and Amazon-Sports from the 2014 Amazon Product Data (McAuley et al., 2015), together with MovieLens-100K (ML-100K) (Harper & Konstan, 2015). All three datasets provide original interaction timestamps, which are required to construct the observable query-relative temporal coordinate in Eqn. (4). We treat each rating or review as an implicit interaction and preprocess all datasets with the same timestamp-preserving pipeline. Beauty and Sports mainly represent sparse short-history regimes, with average sequence lengths of 8.88 and 8.33, respectively, whereas ML-100K provides substantially longer histories with an average length of 104.98.

Baselines. We compare GPrN with nine baselines, including GRU4Rec (Hidasi et al., 2016), SASRec (Kang & McAuley, 2018), BERT4Rec (Sun et al., 2019), FEARec (Du et al., 2023), TiSASRec (Li et al., 2020), SURGE (Chang et al., 2021), TGT (Xia et al., 2023), EA-GPS (Zhang et al., 2025a), and the recent state-of-the-art method S^2HyRec (Liu et al., 2026).

Evaluation Metrics. We adopt chronological leave-two-out evaluation. The last interaction of each user is used for testing, the penultimate interaction is used for validation, and all earlier interactions form the training prefix. The sequential graph is constructed only from training user-item interactions and train-only consecutive item transitions. We report Recall@N and MRR@N with $N \in \{5, 10\}$ under full ranking without sampled negatives. For each query, items already present in the visible history are masked from the candidate set, except that the ground-truth next item is retained when it has appeared previously. The same mask is applied to all compared methods.

Implementation Details. We set the embedding dimension to $d = 64$, use $L = 2$ graph propagation layers, and use $K = 16$ cosine basis functions. In the process reader, the process weight is selected for each dataset using validation MRR@10; RQ3 illustrates its sensitivity on Beauty. We optimize the full-softmax cross-entropy objective with Adam (Kingma & Ba, 2015).

4.2 Experimental Results & Analysis

Overall Performance (RQ1). Table 1 reports the full-ranking results on the three datasets. Our main observations are as follows: 1) GPrN ranks first on all twelve columns. The relative gains over the runner-up in Recall@5 reach 25.0% and 31.0% on Beauty and Sports, and 18.1% on ML-100K, far exceeding the per-seed variability of each column. 2) GPrN outperforms the time-aware baseline (TiSASRec) on all datasets. This shows that the observable query-relative temporal coordinate with

candidate-conditioned functional reading is more effective than treating timestamps as input features. 3) GPrN outperforms S^2HyRec, the strongest graph-based baseline in Recall@10 on all three datasets. Its matched backbone-only control ($\text{GPrN}_{w/o\text{Reader}}$ in Table 2) also outperforms S^2HyRec on Beauty and Sports, while the process reader further improves this backbone.

Table 1: Prediction performance (%) on three datasets. All rows report mean±std over three random seeds. The best results are in **bold** and the sub-optimal are underlined. Significant improvements are marked with † (paired t-test, $p < 0.05$).

Dataset	Beauty				Sports				ML-100K			
Model	R@5	R@10	MRR@5	MRR@10	R@5	R@10	MRR@5	MRR@10	R@5	R@10	MRR@5	MRR@10
GRU4Rec	4.68±.03	6.62±.04	2.74±.02	3.00±.01	2.27±.02	3.54±.03	1.26±.01	1.42±.02	11.88±.20	20.04±.16	6.49±.08	7.56±.14
SASRec	5.27±.02	7.31±.06	3.20±.02	3.46±.01	2.71±.03	4.22±.02	1.58±.02	1.78±.02	12.30±.08	19.41±.10	6.09±.05	7.02±.07
BERT4Rec	4.59±.06	6.94±.03	2.67±.02	2.98±.01	2.32±.01	3.64±.04	1.35±.01	1.52±.01	11.13±.07	19.41±.10	5.80±.07	6.93±.03
FEARec	3.02±.03	4.52±.05	1.76±.01	1.96±.01	1.03±.01	1.70±.01	0.52±.01	0.61±.01	6.36±.11	11.77±.10	2.71±.02	3.44±.05
TiSASRec	3.76±.02	5.34±.02	2.25±.01	2.46±.01	1.71±.03	2.77±.02	0.92±.01	1.05±.01	9.65±.06	18.45±.13	4.84±.03	5.99±.04
SURGE	4.69±.05	6.77±.07	2.74±.01	3.01±.02	2.22±.01	3.49±.03	1.20±.01	1.37±.01	11.35±.14	17.92±.19	5.84±.02	6.70±.03
TGT	2.87±.02	4.37±.06	1.54±.01	1.74±.01	1.11±.01	1.76±.02	0.58±.01	0.66±.01	10.50±.09	16.33±.07	5.62±.03	6.39±.11
EA-GPS	2.29±.02	3.68±.03	1.20±.01	1.38±.01	0.69±.01	1.19±.01	0.33±.01	0.40±.01	11.03±.09	17.18±.08	5.46±.04	6.26±.09
S^2HyRec	5.47±.03	7.63±.02	3.47±.04	3.76±.01	2.77±.01	4.07±.02	1.68±.02	1.85±.01	11.35±.05	19.94±.09	6.30±.07	7.44±.08
GPrN	**6.84±.07**†	**9.66±.05**†	**4.10±.04**†	**4.47±.03**†	**3.63±.02**†	**5.44±.01**†	**2.07±.02**†	**2.31±.01**†	**14.53±.09**†	**22.91±.07**†	**7.71±.11**†	**8.81±.06**†

Ablation Study (RQ2). In this section, we isolate the contribution of each component of the GPrN. Specifically, we compare the following variants: 1) $\text{GPrN}_{w/o\text{Reader}}$ sets $\lambda_p = 0$ and reduces GPrN exactly to its backbone. 2) $\text{GPrN}_{\text{Coarse}}$, $\text{GPrN}_{\text{Selective}}$, and $\text{GPrN}_{\text{Local}}$ retain a single process branch at a time. 3) $\text{GPrN}_{w/o\text{Gate}}$ replaces the query-adaptive branch gate in Eqn. (11) with equal branch weights. 4) $\text{GPrN}_{w/o\text{Norm}}$ disables the query-wise branch-score standardization in Eqn. (10). Table 2 reports the ablation results on Beauty and Sports. Then, we have the following observations: 1) Removing the reader entirely ($\text{GPrN}_{w/o\text{Reader}}$) degrades all four metrics consistently on both datasets. This observation indicates that process reading adds consistent gains over the already competitive graph and GRU backbone. 2) The full reader outperforms every single-branch variant. $\text{GPrN}_{\text{Local}}$ is the strongest single branch on both datasets, which confirms recency as the dominant signal. 3) Both $\text{GPrN}_{w/o\text{Gate}}$ and $\text{GPrN}_{w/o\text{Norm}}$ degrade the full model, supporting the adaptive fusion and the branch-score standardization designs.

Table 2: Ablation results (%) for the multi-scale residual process reader.

Variant	Beauty				Sports			
	R@5	R@10	MRR@5	MRR@10	R@5	R@10	MRR@5	MRR@10
$\text{GPrN}_{w/o\text{Reader}}$	6.20±.05	8.83±.16	3.82±.03	4.19±.03	3.51±.03	5.19±.03	2.02±.03	2.25±.03
$\text{GPrN}_{\text{Coarse}}$	6.46±.06	9.00±.04	3.81±.04	4.15±.08	3.54±.03	5.21±.04	2.05±.03	2.29±.02
$\text{GPrN}_{\text{Selective}}$	6.42±.03	8.98±.03	3.70±.03	4.05±.03	3.62±.05	5.21±.02	2.06±.03	2.29±.02
$\text{GPrN}_{\text{Local}}$	6.51±.06	9.15±.07	3.88±.05	4.23±.02	3.62±.06	5.41±.04	2.06±.04	2.30±.04
$\text{GPrN}_{w/o\text{Gate}}$	6.46±.02	9.08±.08	3.73±.03	4.08±.03	3.53±.03	5.32±.04	2.05±.04	2.30±.02
$\text{GPrN}_{w/o\text{Norm}}$	6.38±.03	9.01±.06	3.82±.03	4.17±.03	3.51±.03	5.25±.07	1.97±.02	2.23±.03
GPrN	**6.84±.07**	**9.66±.05**	**4.10±.04**	**4.47±.03**	**3.63±.02**	**5.44±.01**	**2.07±.02**	**2.31±.01**

Hyperparameter Study (RQ3). In this section, we examine how the four process-reader hyperparameters affect accuracy on Beauty: the process weight λ_p, the smoothing bandwidth σ, the number of cosine basis functions K, and the residual cap $\alpha_{\max}$. We vary one factor at a time around the initial configuration ($\lambda_p = 1.0$, $\sigma = 0.08$, $K = 16$, $\alpha_{\max} = 0.30$) and keep all remaining hyperparameters at the values stated in the Implementation Details. Each configuration is trained with three random seeds, and the final configuration is selected by the three-seed mean of validation MRR@10. Figure 2 summarizes the results: 1) **Process weight λ_p:** the curve rises steeply out of the backbone regime, peaks at an interior value, and decays gradually. The process term should therefore be present but bounded; the selected $\lambda_p = 2.0$ also attains the smallest seed variance among all tested values. 2) **Smoothing bandwidth σ:** the curves are essentially flat over the tested range, so GPrN is insensitive to the smoothing bandwidth. 3) **Basis size K:** accuracy saturates beyond a small basis set. Additional basis functions contribute capacity rather than signal, consistent with the exact-reading property. 4) **Residual cap $\alpha_{\max}$:** with $\lambda_p = 1.0$ fixed in the plotted sweep, performance peaks at

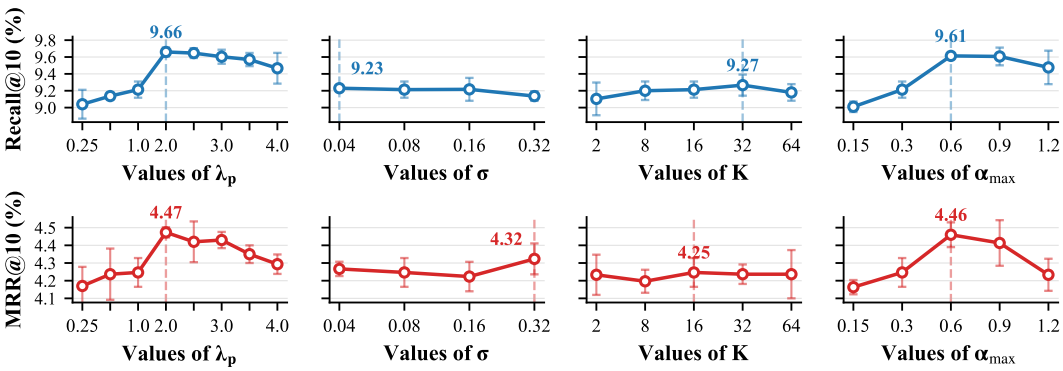

Figure 2: Hyperparameter sensitivity on Beauty under full ranking (test set). Each point averages three random seeds with ± 1 std error bars.

$\alpha_{\max} = 0.60$. At the selected $\lambda_p = 2.0$, the additional comparison in Appendix E.4 instead favors $\alpha_{\max} = 0.30$.

Efficiency Analysis (RQ4). In this section, we quantify the practical cost of the residual process reader. We measure the trainable parameter count, the wall-clock training time per epoch, and the full-ranking test latency per query on Beauty, running every model through the same training and evaluation protocol. We compare GPrN with the four most competitive baselines from Table 1 (S^2HyRec, SASRec, BERT4Rec, and SURGE) and with its matched backbone ($\text{GPrN}_{w/o\text{Reader}}$).

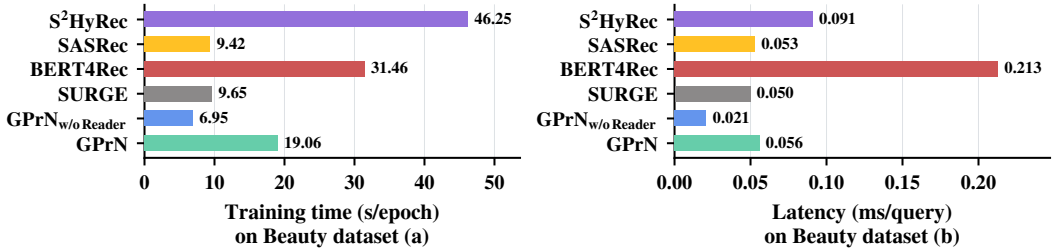

Figure 3: Training time per epoch (a) and full-ranking latency per query (b) on Beauty, measured under identical hardware and batch settings.

Figure 3 reports the results. In capacity, the reader is lightweight and adds only a small fraction of trainable parameters. Its compute cost is visible but bounded: training is slower than the matched backbone, yet remains faster than the heavier baselines. At inference, the matrix-based full-ranking protocol absorbs most of the reader's cost and keeps GPrN on par with the fastest baselines. The reader therefore trades a bounded training-time overhead for candidate-conditioned process reading while preserving inference efficiency.

5 Conclusion & Future Work

In this work, we study sequential recommendation from a process-valued perspective and present Graph Process Neuron (GPrN). GPrN augments a sequential graph and a lightweight order-sensitive encoder with a multi-scale residual process reader over observable timestamps. The reader forms coarse, selective, and local process views, and its bounded residual injection keeps the contribution separable from the backbone and preserves matrix-based full ranking. Extensive experiments on three real-world datasets demonstrate the effectiveness of GPrN. In future, we plan to extend GPrN to richer kernel families and apply it to broader scenarios beyond sequential recommendation.

AI USE STATEMENT

Generative AI tools were used to assist language editing. The authors reviewed and validated all AI-assisted technical content. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

All three datasets are public. Beauty and Sports are derived from the timestamped 2014 Amazon Product Data (McAuley et al., 2015), and ML-100K is provided by GroupLens (Harper & Konstan, 2015). We also release our source code, processed datasets and baseline methods at the anonymous repository: `https://anonymous.4open.science/r/GPrN`.

REFERENCES

Tobia Boschi, Francesca Bonin, Rodrigo Ordonez-Hurtado, Cécile Rousseau, Alessandra Pascale, and John Dinsmore. Functional graph convolutional networks: A unified multi-task and multi-modal learning framework to facilitate health and social-care insights. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 7188–7196, 2024. doi: 10.24963/ijcai.2024/795.

Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 378–387, 2021. doi: 10.1145/3404835.3462968.

Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S. Sheng. Frequency enhanced hybrid attention network for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 78–88, 2023. doi: 10.1145/3539618.3591689.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713, 2018.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015. doi: 10.1145/2827872.

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 (SIGIR)*, pp. 639–648, 2020. doi: 10.1145/3397271.3401063.

Xin-Gui He and Shao-Hua Xu. *Process Neural Networks: Theory and Applications*. Springer, Berlin, 2010.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the Fourth International Conference on Learning Representations (ICLR)*, 2016.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, 2018. doi: 10.1109/ICDM.2018.00035.

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

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017. OpenReview.net.

Jiacheng Li, Yujie Wang, and Julian J. McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 322–330, 2020. doi: 10.1145/3336191.3371786.

Zilong Li, Jia Zhu, Chenglei Huang, Zhangze Chen, Hanghui Guo, Guoqing Ma, and Jianxia Ling. Capturing dynamic user interests under modality imbalance for multimodal sequential recommendation. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 15198–15206, Singapore, 2026. AAAI Press. doi: 10.1609/AAAI.V40I18.38544.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*, 2021.

Yuchen Liu, Kunyu Ni, Zhongying Zhao, Guoqing Chao, and Yanwei Yu. S^2HyRec: Self-supervised hypergraph sequential recommendation. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 15394–15402, 2026. doi: 10.1609/aaai.v40i18.38566.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 43–52, 2015. doi: 10.1145/2766462.2767755.

Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 813–823, 2022. doi: 10.1145/3488560.3498433.

James O. Ramsay and Bernard W. Silverman. *Functional Data Analysis*. Springer, New York, 2nd edition, 2005.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 811–820, 2010. doi: 10.1145/1772690.1772773.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1441–1450, 2019. doi: 10.1145/3357384.3357895.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):346–353, 2019. doi: 10.1609/aaai.v33i01.3301346.

Lianghao Xia, Chao Huang, Yong Xu, and Jian Pei. Multi-behavior sequential recommendation with temporal graph transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35 (6):6099–6112, 2023. doi: 10.1109/TKDE.2022.3175094.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 23078–23097, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/V1/2025.ACL-LONG.1126.

Jinyu Zhang, Chao Li, and Zhongying Zhao. Lightweight yet efficient: An external attentive graph convolutional network with positional prompts for sequential recommendation. *ACM Transactions on Information Systems*, 43(3):79:1–79:25, 2025a. doi: 10.1145/3719343.

Yixin Zhang, Yong Liu, Yonghui Xu, Hao Xiong, Chenyi Lei, Wei He, Lizhen Cui, and Chunyan Miao. Enhancing sequential recommendation with graph contrastive learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2398–2405, 2022. doi: 10.24963/ijcai.2022/333.

Zeyang Zhang, Xin Wang, Haibo Chen, Haoyang Li, and Wenwu Zhu. Disentangled dynamic graph attention network for out-of-distribution sequential recommendation. *ACM Transactions on Information Systems*, 43(1):19:1–19:42, 2025b. doi: 10.1145/3701988.

Wenxu Zhao, Yongkang Li, Yuheng Wu, Yuan Cui, and Xiaona Xia. Popularity bias mitigation based on time interval-aware data augmentation for sequential recommendation. *ACM Transactions on Intelligent Systems and Technology*, 17(3):72:1–72:24, 2026. doi: 10.1145/3799241.

Zhongying Zhao, Jinyu Zhang, Chuanxu Jia, Chao Li, Yanwei Yu, and Qingtian Zeng. CoLA-Former: Graph transformer using communal linear attention for lightweight sequential recommendation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3689–3697, 2025. doi: 10.24963/ijcai.2025/410.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1059–1068, 2018. doi: 10.1145/3219819.3219823.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pp. 5941–5948, 2019. doi: 10.1609/aaai.v33i01.33015941.

TECHNICAL APPENDIX

A NOTATION

Table 3 summarizes the main symbols used in this paper.

Table 3: Summary of notations.

Symbol	Description
$\mathbf{A}$, $\mathbf{D}$, $\hat{\mathbf{A}}$	Adjacency, degree, and normalized adjacency matrices
L, $\mathbf{H}^{(l)}$	Propagation depth and layer-l node representations
$\mathbf{h}_v^g$, d	Graph-aware representation of node v; embedding dimension
$\mathbf{g}_u^{(q)}$, $\mathbf{s}_u^{(q)}$	Last GRU output and the sequential state
η, ξ	Learnable last-item gate and its underlying scalar
$\mathbf{z}_u^{(q)}$	Query context $[\mathbf{h}_u^g \,\|\, \mathbf{s}_u^{(q)}]$
$t_u^{(q)}$, $T_{\max}$	Query anchor timestamp and maximum estimated lag
$\tau_{u,j}^{(q)}$	Query-relative coordinate of interaction j
$\kappa_\sigma(\tau;\mu)$, μ, σ	Gaussian smoother with interaction location μ and bandwidth σ
$\mathcal{B}_m$, B, M	The m-th block, maximum block size, and number of blocks
$\bar{\mathbf{h}}_m$, $\bar{\tau}_m$	Block mean representation and mean coordinate
$\mathbf{W}_{\text{sel}}$, e_m	Selector weight and query-block similarity
$\mathcal{T}_R$, R, π_m	Selected top-R blocks and their softmax weights
$\mathcal{L}_W$, W	Recent local window and its size
$\mathbf{X}_{u,b}^{(q)}(\tau)$	Process view of branch b
$\mathcal{R} = \{\text{c}, \text{s}, \text{l}\}$, b	Branch set (coarse, selective, local) and branch index
$\phi_k(\tau)$, K	The k-th cosine basis function and the basis size
$\mathbf{a}_{u,b,k}^{(q)}$	The k-th process coefficient
$\mathbf{W}_b^Q$, $\mathbf{b}_b^Q$, $\mathbf{W}_b^I$, $\mathbf{b}_b^I$	Query- and item-side factor generators
$\mathbf{p}_{u,b}^{(q)}$, $\mathbf{r}_{i,b}$	Query- and item-side factors
$\mathcal{K}_{ui,b}^{(q)}(\tau)$	Candidate-conditioned scalar reading kernel
$\mathbf{m}_{u\|i,b}^{(q)}$, $s_{ui,b}^{(q)}$	Process message and branch score
$\mu_{u,b}^{(q)}$, $v_{u,b}^{(q)}$, $\bar{s}_{ui,b}^{(q)}$	Branch-wise mean, variance, and standardized score
$\boldsymbol{\omega}_u^{(q)}$, $\mathbf{W}_g$, $\mathbf{b}_g$	Branch-weight vector and its gate parameters
$s_{ui}^{P,(q)}$, $s_{ui}^{B,(q)}$	Fused process score and backbone score
$\alpha_u^{(q)}$, $\alpha_{\max}$	Residual coefficient and its cap
$\mathbf{w}_\alpha$, b_α, λ_p	Residual gate parameters and global process weight

B DERIVATIONS AND FUNCTIONAL PROPERTIES

This section records the derivations behind the reader design and the formal properties of the branch-wise functional reader. For branch b, define the functional

$$\mathcal{M}_{ui,b}^{(q)}[\mathbf{X}] = \int_0^1 \mathcal{K}_{ui,b}^{(q)}(\tau)\mathbf{X}(\tau)d\tau. \tag{16}$$

Coefficient accumulation. Substituting the view definitions in Eqn. (6) into Eqn. (7) and exchanging the order of summation and integration gives

$$\mathbf{a}_{u,b,k}^{(q)} = \sum_j w_{b,j}\, w_k(\mu_j)\, \mathbf{h}_j, \qquad w_k(\mu) = \int_0^1 \phi_k(\tau)\kappa_\sigma(\tau;\mu)d\tau, \tag{17}$$

where $w_{b,j}$ and μ_j are the branch-specific event weights and interaction locations (e.g., $w_{\text{c},m} = 1/M$ with $\mu_m = \bar{\tau}_m$ for the coarse branch). Each scalar integral $w_k(\mu)$ depends only on μ for fixed σ and k, so we precompute it on a dense grid over μ and evaluate it by linear interpolation during training and ranking.

Proposition 1 (Exact contraction given process projections). Let $\mathcal{V}_K = \mathrm{span}\{\phi_1, \ldots, \phi_K\}$ and let Π_K denote the orthogonal projection onto $\mathcal{V}_K$. Since $\mathcal{K}^{(q)}_{ui,b} \in \mathcal{V}_K$,

$$\mathcal{M}^{(q)}_{ui,b}[\mathbf{X}] = \mathcal{M}^{(q)}_{ui,b}[\Pi_K \mathbf{X}] = \sum_{k=1}^{K} c^{(q)}_{ui,b,k} \mathbf{a}_k. \tag{18}$$

Proof. For each output dimension, $\mathcal{K}^{(q)}_{ui,b} \in \mathcal{V}_K$ and $\mathbf{X} - \Pi_K \mathbf{X}$ lies in the orthogonal complement of $\mathcal{V}_K$, so their L^2 inner product vanishes and $\mathcal{M}^{(q)}_{ui,b}[\mathbf{X}] = \mathcal{M}^{(q)}_{ui,b}[\Pi_K \mathbf{X}]$. Expanding $\Pi_K \mathbf{X} = \sum_{k=1}^{K} \mathbf{a}_k \phi_k(\tau)$ with $\mathbf{a}_k = \int_0^1 \phi_k(\tau) \mathbf{X}(\tau) d\tau$, the orthonormality $\int_0^1 \phi_j(\tau)\phi_k(\tau) d\tau = \delta_{jk}$ yields $c^{(q)}_{ui,b,k} = \int_0^1 \mathcal{K}^{(q)}_{ui,b}(\tau)\phi_k(\tau) d\tau = p^{(q)}_{u,b,k} r_{i,b,k}$, which proves the second equality. □

Proposition 2 (Process perturbation stability). For any square-integrable processes $\mathbf{X}$ and $\widetilde{\mathbf{X}}$,

$$\left\| \mathcal{M}^{(q)}_{ui,b}[\mathbf{X}] - \mathcal{M}^{(q)}_{ui,b}[\widetilde{\mathbf{X}}] \right\|_2 \leq \|\mathcal{K}^{(q)}_{ui,b}\|_{L^2} \|\mathbf{X} - \widetilde{\mathbf{X}}\|_{L^2}. \tag{19}$$

Proof. The bound follows from the Cauchy-Schwarz inequality applied componentwise. For the norm of the kernel, Parseval's identity with respect to the orthonormal basis gives $\|\mathcal{K}^{(q)}_{ui,b}\|^2_{L^2} = \sum_{k=1}^{K} \left(p^{(q)}_{u,b,k} r_{i,b,k} \right)^2 \leq K$, where the inequality uses $|p^{(q)}_{u,b,k}|, |r_{i,b,k}| \leq 1$ from the $\tanh$ generators. Hence $\|\mathcal{K}^{(q)}_{ui,b}\|_{L^2} \leq \sqrt{K}$. □

Proposition 3 (Finite-basis approximation). Let $\mathcal{K}^{\star}_{ui,b} \in L^2([0,1])$ be an ideal branch-specific kernel and $\mathcal{K}^{\star}_{ui,b,K} = \Pi_K \mathcal{K}^{\star}_{ui,b}$. Then

$$\left\| \mathcal{M}^{\star}_{ui,b}[\mathbf{X}] - \mathcal{M}^{\star}_{ui,b,K}[\mathbf{X}] \right\|_2 \leq \|\mathcal{K}^{\star}_{ui,b} - \mathcal{K}^{\star}_{ui,b,K}\|_{L^2} \|\mathbf{X}\|_{L^2}. \tag{20}$$

Proof. The difference is $\mathcal{M}^{\star}_{ui,b}[\mathbf{X}] - \mathcal{M}^{\star}_{ui,b,K}[\mathbf{X}] = \int_0^1 \left(\mathcal{K}^{\star}_{ui,b} - \Pi_K \mathcal{K}^{\star}_{ui,b} \right) \mathbf{X} \, d\tau$, and the Cauchy-Schwarz inequality yields the claim. The cosine basis is complete in $L^2([0,1])$, so the projection error $\|\mathcal{K}^{\star}_{ui,b} - \mathcal{K}^{\star}_{ui,b,K}\|_{L^2}$ vanishes as K increases for any square-integrable target kernel. □

Factorized full ranking. For full ranking, the factorized kernel permits branch-specific query and item vectors

$$\begin{aligned} \tilde{\mathbf{h}}^{(q)}_{u,b} &= \left[p^{(q)}_{u,b,1} \mathbf{a}^{(q)}_{u,b,1}; \ldots; p^{(q)}_{u,b,K} \mathbf{a}^{(q)}_{u,b,K} \right], \\ \tilde{\mathbf{h}}_{i,b} &= \left[r_{i,b,1} \mathbf{h}^{g}_{i}; \ldots; r_{i,b,K} \mathbf{h}^{g}_{i} \right], \end{aligned} \tag{21}$$

and then

$$s^{(q)}_{ui,b} = \left(\tilde{\mathbf{h}}^{(q)}_{u,b} \right)^{\top} \tilde{\mathbf{h}}_{i,b} = \left(\mathbf{m}^{(q)}_{u|i,b} \right)^{\top} \mathbf{h}^{g}_{i}. \tag{22}$$

Hence, item-side folded matrices can be cached and reused across queries.

Remark. Replacing a branch-specific candidate-conditioned kernel by a fixed shared function recovers the linear functional core of a classical process neuron (He & Xu, 2010). Setting $\lambda_p = 0$ in Eqn. (14) removes the process reader and retains the sequential graph and GRU backbone.

C Controlled Mechanism Validation

The ablations in Table 2 establish the contribution of the process reader as a whole. Here, we separate two questions: whether the reader benefits from measured temporal gaps beyond interaction order, and whether its reading kernel needs to depend on the candidate item. Both controls retain the graph and GRU backbone, the three process branches, score normalization, and residual fusion.

Real-time gaps versus interaction order. In the ordinal-time control, we replace only the query-relative coordinate in Eqn. (4) with

$$\tau^{\mathrm{ord},(q)}_{u,j} = \frac{q - 1 - j}{\max\{q - 2, 1\}}, \qquad 1 \leq j < q, \tag{23}$$

where j is the position of an interaction in the visible prefix. The latest observation remains at zero, and earlier observations are evenly spaced by their order. Thus, the control retains the same items and sequence order but discards the measured gaps between their timestamps. All three process views use this coordinate; the graph, sequential encoder, and candidate-conditioned kernels remain unchanged.

Candidate-conditioned versus shared reading. In the shared-kernel control, we replace the item-specific factor in Eqn. (8) by a common factor for every candidate:

$$\begin{aligned} \bar{\mathbf{h}}^g &= \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\mathbf{h}_i^g, \qquad \bar{\mathbf{r}}_b = \tanh\!\left(\mathbf{W}_b^I\bar{\mathbf{h}}^g + \mathbf{b}_b^I\right), \\ \mathcal{K}_{u,b}^{\mathrm{shared},(q)}(\tau) &= \sum_{k=1}^{K} p_{u,b,k}^{(q)}\bar{r}_{b,k}\phi_k(\tau), \end{aligned} \tag{24}$$

where $\bar{\mathbf{h}}^g$ is the mean graph-aware item representation and $\bar{\mathbf{r}}_b$ is shared across all candidates in branch b. This control retains the item-side generator parameters and still scores each candidate using its own $\mathbf{h}_i^g$ in Eqn. (9). It removes only the candidate dependence of the temporal reading kernel.

Evaluation protocol. We evaluate both controls on Beauty, Sports, and ML-100K using the same data splits, three seeds, and training-only graph as GPrN. The graph is constructed once from training interactions and shared across training prefixes, following the full-model protocol. Validation and test target interactions are excluded from the graph. We retain the full model's dataset-specific hyperparameters and optimization settings. Checkpoints are selected by validation MRR@10. All variants use the same full-ranking candidate mask. Tables 4 and 5 report the aggregate and paired per-seed results, respectively.

Table 4: Controlled validation under full ranking (%). Results are mean±std over three random seeds.

	Beauty		Sports		ML-100K	
Variant	R@10	MRR@10	R@10	MRR@10	R@10	MRR@10
$\text{GPrN}_{\text{Ordinal}}$	9.36±0.07	4.29±0.09	5.28±0.02	2.26±0.02	22.06±0.09	8.46±0.08
$\text{GPrN}_{\text{Shared}}$	9.52±0.07	4.37±0.07	5.37±0.02	2.28±0.02	22.29±0.10	8.58±0.08
GPrN	9.66±0.05	4.47±0.03	5.44±0.01	2.31±0.01	22.91±0.07	8.81±0.06

Table 5: Per-seed results of the mechanism controls (%). Seeds 42, 3407, and 2024 are those used for the main results; all variants share them on each dataset.

		Beauty		Sports		ML-100K	
Variant	Seed	R@10	MRR@10	R@10	MRR@10	R@10	MRR@10
$\text{GPrN}_{\text{Ordinal}}$	42	9.28	4.19	5.27	2.25	22.02	8.48
	3407	9.42	4.35	5.31	2.28	21.99	8.38
	2024	9.37	4.34	5.27	2.24	22.16	8.53
$\text{GPrN}_{\text{Shared}}$	42	9.44	4.30	5.37	2.27	22.30	8.57
	3407	9.56	4.43	5.39	2.30	22.19	8.50
	2024	9.55	4.38	5.35	2.27	22.39	8.66
GPrN	42	9.62	4.45	5.44	2.30	22.94	8.84
	3407	9.71	4.51	5.45	2.32	22.83	8.74
	2024	9.64	4.46	5.43	2.30	22.95	8.86

Table 4 compares the two controls across three datasets. Replacing measured temporal gaps with evenly spaced ordinal positions reduces Recall@10 by 0.30, 0.16, and 0.85 points on Beauty, Sports, and ML-100K, respectively. Sharing the temporal reading kernel reduces Recall@10 by 0.14, 0.07, and 0.62 points. MRR@10 follows the same ordering on all three datasets. Moreover, GPrN exceeds both controls in both metrics for every matched seed in Table 5. This observation indicates that measured temporal gaps provide information beyond interaction order. It also supports a further,

smaller gain from candidate-specific kernel factors. The ordinal control has the larger effect on each dataset, although the magnitude of both effects varies across datasets.

D Dataset Details

This section complements the dataset description in Section 4.1 with the full statistics of the three datasets. Table 6 reports the number of users, items, and interactions after the unified timestamp-preserving preprocessing, together with the numbers of training, validation, and test interactions. Under the chronological leave-two-out protocol, each user contributes exactly one validation and one test interaction, so the #Valid and #Test columns coincide with the number of users. The #Train column counts the remaining interactions that form the training prefixes and the training-only graph.

Table 6: Statistics of the three datasets after the unified timestamp-preserving preprocessing.

Dataset	#Users	#Items	#Interactions	#Train	#Valid	#Test
Beauty	22,363	12,101	198,502	153,776	22,363	22,363
Sports	35,598	18,357	296,337	225,141	35,598	35,598
ML-100K	943	1,341	98,996	97,110	943	943

E Detailed Experimental Analysis

This appendix records the preprocessing and training protocol behind Section 4.1, together with the detailed per-observation analysis, including the exact numbers (mean$\pm$std over three random seeds) on which each observation is based.

E.1 Preprocessing and Training Protocol

Preprocessing. We treat each rating or review as an implicit interaction, remove exact duplicate (u, i, t) interactions, and iteratively apply 5-core filtering so that every retained user and item has at least five interactions. For each user, interactions are stably sorted by timestamp; ties preserve their order in the raw file.

Evaluation protocol. At validation and test time, the target item and its timestamp are never visible to the model. The validation interaction becomes an observed historical interaction when predicting the final test item, but the graph itself remains training-only. The graph is constructed once from training interactions and remains fixed across training queries; validation and test targets never enter the graph, and query-relative times are computed from visible timestamps only.

Training configuration. Training queries are generated from all valid prefixes of the training sequence, and the observable reference time is always the timestamp of the last interaction in the corresponding prefix. The residual coefficient is initialized around 0.05. Sequence truncation, when required by memory-intensive baselines, is applied in the DataLoader rather than during dataset preprocessing, and item/timestamp histories are truncated synchronously. During evaluation, GPrN caches the three branch-specific item-side folded matrices and reuses them across queries. All remaining hyperparameters are fixed at the values stated in Section 4.1, and the search spaces are provided in the released configuration files. The complete efficiency measurements of all nine baselines, together with the same analysis on Sports, are provided in the released logs.

E.2 Overall Performance (RQ1)

(1) Overall superiority. GPrN ranks first on all twelve columns of Table 1. Relative to the strongest baseline in Recall@10 on each dataset (S^2HyRec on Beauty, SASRec on Sports, and GRU4Rec on ML-100K), GPrN improves Recall@10 by 2.03, 1.22, and 2.87 points (26.6%, 28.9%, and 14.3% relative gains), and improves MRR@10 by 0.71, 0.46, and 1.25 points (18.9%, 24.9%, and 16.5%). Every margin far exceeds the per-seed variability of its column (e.g., the 0.46 MRR@10 margin on Sports exceeds the largest std of 0.02 in that column).

(2) Comparison with time-aware methods. GPrN outperforms the time-aware baseline TiSASRec by 80.9%, 96.4%, and 24.2% in Recall@10 on Beauty, Sports, and ML-100K. TiSASRec compresses the history into a single candidate-independent sequence representation before scoring candidates, so the temporal evidence is read once rather than per candidate. GPrN instead re-reads the history functionally for every candidate at multiple temporal scales, which allows the same history to support different candidates through different readings; we attribute this to the candidate-conditioned reading in Eqn. (9).

(3) Comparison with graph-based baselines. Among the graph-based baselines, S^2HyRec obtains the highest Recall@10 on all three datasets. GPrN improves upon it by 26.6%, 33.7%, and 14.9% on Beauty, Sports, and ML-100K, respectively. The matched $\text{GPrN}_{w/o\text{Reader}}$ control in Table 2 already exceeds S^2HyRec on both Amazon datasets (8.83 vs. 7.63 Recall@10 on Beauty and 5.19 vs. 4.07 on Sports). This observation indicates that the process reader provides further gains over GPrN's graph and GRU backbone.

E.3 Ablation Study (RQ2)

(1) The residual process reader improves the backbone. Removing the reader entirely ($\text{GPrN}_{w/o\text{Reader}}$) degrades all four metrics consistently: on Beauty, Recall@5, Recall@10, MRR@5, and MRR@10 drop by 0.64, 0.83, 0.28, and 0.28 (9.4%, 8.6%, 6.8%, and 6.3% relative), and on Sports by 0.12, 0.25, 0.05, and 0.06 (3.3%, 4.6%, 2.4%, and 2.6%). This observation confirms that functional reading adds consistent gains beyond the already competitive graph and GRU backbone.

(2) The three temporal scales are complementary. On Beauty the full reader outperforms the strongest single branch (*Local only*, 9.15 Recall@10) by 0.51 Recall@10 and every single-branch variant by at least 0.51; on Sports the advantage shrinks to at most 0.03 Recall@10, and the full model is marginally ahead on all four metrics. The pattern matches the temporal structure of the data: Beauty's sparse histories contain slow preference drift that the coarse and selective scales capture on top of the local evidence, whereas Sports is more recency-concentrated. Notably, *Local only* is the strongest single reader on both datasets, confirming recency as the dominant signal, and the multi-scale mixture adds value exactly where additional temporal scales exist in the data.

(3) Adaptive fusion and score normalization are both necessary. Replacing the query-adaptive gate with uniform weights costs 0.58 and 0.12 Recall@10 on Beauty and Sports, showing that different queries require different branch weights. Disabling branch-score standardization also consistently degrades the full model on both datasets, supporting the design of Eqn. (10): without query-wise standardization, branch scores live on incomparable scales and the bounded residual in Eqn. (12) is dominated by whichever branch happens to have the largest raw magnitude.

E.4 Hyperparameter Study (RQ3)

(1) Process weight λ_p. The curve rises steeply out of the backbone regime and peaks at an interior point: starting from the $\lambda_p \to 0$ backbone (8.83 Recall@10, Table 2), the first increment to $\lambda_p = 0.25$ already recovers 9.04, the initial value 1.0 reaches 9.21, and the mean performance peaks at 2.0 (9.66 Recall@10, 4.47 MRR@10) before decaying gradually toward 4.0 (9.47). The peak is stable rather than a favorable draw: $\lambda_p = 2.0$ attains the smallest seed variance in MRR@10 (± 0.03) among all tested values, and validation MRR@10 selects the same value. The reader therefore delivers most of its benefit as soon as the process term is present, and the interior optimum indicates that an overly strong process term starts to interfere with the backbone; the improvement from 1.0 to 2.0 ($+0.45$ Recall@10, $+0.22$ MRR@10) carries over to the main results in Table 1.

(2) Smoothing bandwidth σ. The curves are essentially flat: the full range of mean Recall@10 over $[0.04, 0.32]$ (0.09) is smaller than the largest single-point standard deviation (0.14), and the same holds for MRR@10. The nominal test peak at $\sigma = 0.32$ (4.32 MRR@10) thus reflects seed noise rather than a genuine preference (i.e., it is the weakest configuration in validation MRR@10, 5.57 vs. 5.62 at the default). We keep $\sigma = 0.08$, which is best in validation MRR@10. GPrN is therefore insensitive to the smoothing bandwidth within the tested range, indicating that performance is governed by the multi-scale reading structure rather than by the precise kernel sharpness.

(3) Basis size K. Recall@10 degrades visibly at $K = 2$ (9.10) and the remaining values stay within one standard deviation of one another (9.20–9.27 for $K \in \{8, 16, 32, 64\}$); MRR@10 peaks at the

initial $K = 16$ (4.25), which is also the best value in validation MRR@10. Doubling K beyond 16 thus enlarges the admissible kernel family without a matching accuracy return, consistent with the exact-reading property: once the dominant temporal patterns are representable, additional basis functions contribute capacity rather than signal. We keep the expressive yet efficient $K = 16$.

(4) Residual cap $\alpha_{\max}$. The cap sweep plotted in Figure 2 fixes the initial $\lambda_p = 1.0$: relaxing the cap from 0.15 to 0.60 improves both metrics (9.01 → 9.61 Recall@10, 4.16 → 4.46 MRR@10), after which performance flattens at 0.90 and degrades at 1.20 (9.48). A separate comparison at the adopted $\lambda_p = 2.0$ instead favors the tight cap $\alpha_{\max} = 0.30$; enlarging it degrades Recall@10 (9.52 → 9.41 → 9.37 for 0.60/0.90/1.20). The same pattern holds at the $\lambda_p = 3.0$ anchor (9.67 vs. 9.31/9.23/9.24). Thus, the cap that performs best depends on the process weight: a larger λ_p calls for a tighter cap to control the residual contribution in Eqn. (12). We retain $\alpha_{\max} = 0.30$, the cap under which the process weight was selected.

E.5 Efficiency Analysis (RQ4)

In capacity, the reader is lightweight: it adds only 18K trainable parameters (0.8% of the model size on Beauty). Its compute cost is visible but bounded: training runs 2.7× slower than the matched backbone (19.1 vs. 7.0 s/epoch), yet remains faster than BERT4Rec and S^2HyRec. At inference, the matrix-based full-ranking protocol absorbs most of the reader's cost: per-query latency increases by only 0.035 ms over the backbone, which keeps GPrN within 0.003 ms of SASRec and 0.006 ms of SURGE, 1.6× faster than S^2HyRec, and 3.8× faster than BERT4Rec, and the peak memory stays below 0.9 GB including the cached branch item matrices. The reader therefore trades a bounded training-time overhead for candidate-conditioned process reading and keeps inference on par with the fastest baselines.

F Choice of the Sequential State Encoder

The sequential state encoder summarizes the order-sensitive visible history before process reading. To examine whether this choice materially affects the model, we conduct a controlled preliminary study on Beauty by replacing the GRU with a SASRec-style self-attention encoder (Kang & McAuley, 2018) and an NSA-style sparse attention encoder (Yuan et al., 2025), and keep the remaining GPrN configuration and evaluation protocol unchanged. Table 7 reports the results.

Table 7: Comparison of sequential state encoders on Beauty. All results report mean±std over three random seeds under full ranking. † marks statistically significant improvements over the two attention encoders (paired t-test, $p < 0.05$).

Encoder	R@5	MRR@5	R@10	MRR@10
GPrN_{SA}	5.92±.09	3.56±.03	8.39±.09	3.89±.06
GPrN_{NSA}	6.02±.14	3.61±.02	8.70±.18	3.96±.04
GPrN_{GRU}	**6.84±.07**†	**4.10±.04**†	**9.66±.05**†	**4.47±.03**†

As shown in Table 7, the GRU encoder achieves the best ranking performance among the three alternatives. In particular, it improves Recall@10 and MRR@10 by 15.14% and 14.91% over self-attention, respectively, and by 11.03% and 12.88% over NSA. This observation indicates that a relatively simple recurrent encoder is sufficient for constructing the order-sensitive sequential state required by GPrN.

A possible explanation lies in the division of modeling roles within GPrN. The sequential encoder is not expected to capture all temporal dependencies by itself. Instead, it provides a compact sequential context, whereas long-range, selective, and recent temporal evidence is further modeled by the process-reading module. Consequently, a more expressive attention-based encoder may provide limited additional benefit and partially overlap with the temporal modeling of the process reader. Based on this controlled study, we adopt the GRU as the default sequential state encoder throughout the paper.

G Algorithm and Complexity Analysis

Algorithm 1 summarizes the training and inference procedure of GPrN. We also record the per-stage computational complexity that supports the efficiency analysis in Section 4.2.

Algorithm 1 Training and inference of GPrN

Require: Training queries $\{(\mathcal{S}_u^{(q)}, i^\star)\}$; sequential graph $\mathcal{G}_s$; hyperparameters σ, K, B, R, W, λ_p, $\alpha_{\max}$

1: Precompute the normalized adjacency $\hat{\mathbf{A}}$ and the projection table $[w_k(\mu)]$ in Eqn. (17) on a dense grid over μ.
2: **for** each training batch **do**
3: Propagate $\mathbf{H}^{(l+1)} = \hat{\mathbf{A}}\mathbf{H}^{(l)}$ and form the query context $\mathbf{z}_u^{(q)}$ by Eqs. (2) and (3).
4: Compute $\tau_{u,j}^{(q)}$ by Eqn. (4) and the branch coefficients $\mathbf{a}_{u,b,k}^{(q)}$ by Eqn. (17).
5: **for** $b \in \{\mathrm{coarse}, \mathrm{selective}, \mathrm{local}\}$ **do**
6: Generate the factorized kernels $(\mathbf{p}_{u,b}^{(q)}, \mathbf{r}_{i,b})$ by Eqn. (8) and compute the branch scores $\{s_{ui,b}^{(q)}\}$ by Eqn. (9).
7: **end for**
8: Standardize the branch scores and fuse them through the query-adaptive residual gate by Eqs. (10)–(12).
9: Assemble $\hat{y}_{ui}^{(q)}$ by Eqn. (14) and update all trainable parameters by minimizing the full-softmax cross-entropy loss in Eqn. (15).
10: **end for**
11: **Offline (once):** after training, cache the graph representations $\{\mathbf{h}_i^g\}$ and the item-side factors $\{\mathbf{r}_{i,b}\}$.
12: **Online (per query):** form the query-side factors, score all candidates by matrix multiplication, and return the ranking.

Training complexity. In each training step, the L sparse propagations over $\mathcal{G}_s$ cost time linear in the number of recorded interactions and are shared by all queries in the batch. For each query, the relative coordinates $\tau_{u,j}^{(q)}$ in Eqn. (4) cost $O(|\mathcal{S}_u^{(q)}|)$, and each branch accumulates its coefficients in $O(Kd|\mathcal{S}_u^{(q)}|)$ through table lookups with linear interpolation. Each branch score in Eqn. (9) reduces to a Kd-dimensional inner product between a query-side stack $p_{u,b,k}^{(q)}\mathbf{a}_{u,b,k}^{(q)}$ and an item-side stack $r_{i,b,k}\mathbf{h}_i^g$, where the item-side stack depends only on the item. Scoring the full catalog therefore costs one matrix product of size $O(Kd|\mathcal{I}|)$ per branch. The two backbone terms in Eqn. (13) can be combined on the query side, so backbone scoring requires one matrix multiplication of size $O(d|\mathcal{I}|)$. The total scoring cost per query is therefore $O((3K+1)d|\mathcal{I}|)$, plus $O(3Kd|\mathcal{S}_u^{(q)}|)$ for coefficient accumulation.

Inference complexity. After training, the graph representations $\{\mathbf{h}_v^g\}$ and the item-side stacks are computed once and cached, with memory linear in the number of nodes and in $Kd|\mathcal{I}|$. Each query then follows the same scoring path with the item side precomputed, which underlies the measured latency in Section 4.2. In capacity, the reader adds only 18K trainable parameters (0.8% of the model size on Beauty), and the projection table is a fixed-size lookup computed once.