LOOPGR: PARAMETER-EFFICIENT SCALING FOR GENERATIVE RECOMMENDATION VIA RECURRENT DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation formulates item recommendation as autoregressive generation rather than a discriminative scoring problem. Although empirical studies exhibit favorable scaling behavior in recommendation, unleashing the benefits of scaling remains challenging under the strict model size and inference cost constraints of real-world systems. A central challenge is to combine expressive modeling of user-item interaction histories with flexible computational depth without proportionally increasing model size. We introduce **LoopGR**, a parameter-efficient scaling framework for generative recommendation via recurrent depth. LoopGR employs a tailored hierarchical recurrent depth mechanism that alternates between a shared generative decoder and a lightweight context updater. For efficient inference, we further develop recurrent trajectory self-distillation that distills a low-budget student from a high-budget teacher by aligning both trajectory states and evolution directions. Extensive Experiments show that LoopGR consistently outperforms strong generative recommendation baselines. Compared with a representative encoder–decoder baseline, LoopGR achieves superior performance while reducing model size by up to 52%. These results establish recurrent depth as a new scaling dimension, offering a flexible and resource-efficient scaling path for generative recommendation. Our code is available at `https://anonymous.4open.science/r/LoopGR-620B`

1 INTRODUCTION

Generative recommendation which reconceptualizes the traditional discriminative matching paradigm as a generative task has been a promising paradigm for large-scale recommender systems Wang et al. (2023); Zhou et al. (2025a); Liu et al. (2025). Inspired by tokenization in LLMs and computer vision, recent methods represent each item as a short sequence of discrete semantic IDs (SIDs) and formulate next-item recommendation as autoregressive generation of the target SIDs Rajput et al. (2023); Ju et al. (2026); He et al. (2026). Empirical studies further suggest that increasing model capacity and training computation can improve recommendation performance, suggesting that generative recommenders also benefit from predictable scaling laws Guo et al. (2024); Liu et al. (2026); Hou et al. (2026). However, unlike LLMs, recommender systems are deployed under much tighter computational and serving constraints. The key challenge is not simply to increase computation, but to effectively allocate a limited computational budget between interaction history modeling and target SID generation.

Existing generative recommenders allocate computation differently between history modeling and target generation, as illustrated in Figure 1. This architectural evolution progressively shifts computation from dedicated history modeling toward shared and ultimately target-centric computation Rajput et al. (2023); Zhou et al. (2025b); Yang et al. (2025b). The above architectural designs reveal a common conflict between expressive context modeling and efficient target generation. Rich user-item context is typically obtained by deeply encoding historical interactions before generation, whereas efficient decoding often treats the resulting context as fixed conditioning. Meanwhile, computational depth remains tightly coupled with the number of network layers in existing architectures, such that deeper context refinement requires additional parameters. These observations motivate a central question:

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.

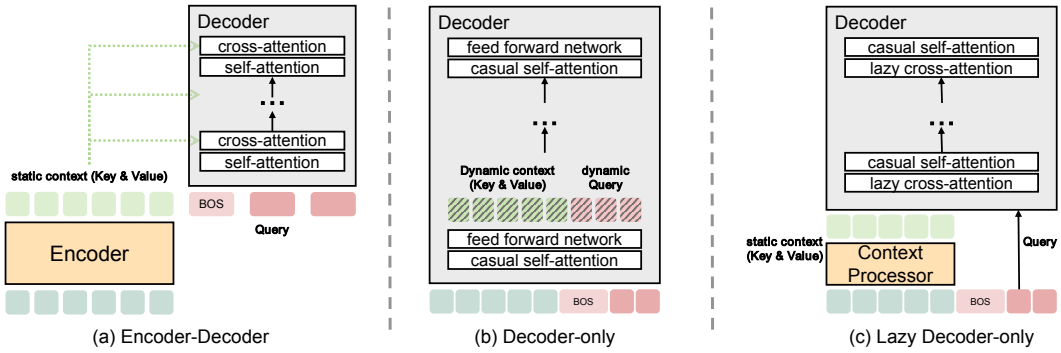


Figure 1: **Backbone comparison of different generative recommendation frameworks.** (a) Encoder-Decoder architectures allocate separate computation to history modeling and target generation. (b) Decoder-Only architectures unify this computation within a shared stack. (c) Lazy Decoder-Only architectures further reallocate the computational budget toward target generation.

How can we design a flexible generative recommendation backbone that dynamically organizes computation between context modeling and target SID generation?

Looped Transformers, which recursively apply a set of shared layers across recurrent steps, offer a promising way to decouple computational depth from parameter size or memory consumption Yang et al. (2025a); Zhu et al. (2025); Geiping et al. (2026). Early studies combined recurrent depth with adaptive computation to iteratively refine token representations Dehghani et al. (2018); Graves (2016), while recent work has revisited this paradigm to improve the reasoning and efficiency capabilities in modern language models Prairie et al. (2026); Wang et al. (2025a); Saunshi et al. (2025). Although recurrent depth improves model efficiency via parameter sharing, it is hard to differentiate two distinct computations over long behavioral histories and target SIDs in generative recommenders. Moreover, additional recurrent steps still incur inference latency, making unrestricted looping impractical under strict serving budgets. This motivates a recommendation-specific recurrent design that separately organizes history refinement and target generation while supporting efficient low-budget inference.

In this paper, we propose **LoopGR**, a parameter-efficient scaling framework for generative recommendation via hierarchical recurrent depth. LoopGR comprises an encoder, a shared decoder and a lightweight shared context updater. After the encoder initializes the behavioral context, high-level decoder cycles and low-level updater cycles are recurrently interleaved. Specifically, the decoder predicts target SID representations from the current context while the updater feeds the emerging target state back to refine the context for subsequent decoding. This design allows user context to evolve during generation while scaling computational depth through shared layers. To support efficient inference under hard serving budgets, we propose a recurrent trajectory self-distillation to match predictive distributions and evolution directions from high-budget to low-budget recurrent trajectories. LoopGR reconciles dynamic context modeling with efficient target generation by decoupling computational depth from parameter count while retaining strong performance under limited recurrent budgets. Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to introduce a recurrent-depth architecture for SID-based generative recommendation, which enables dynamic context refinement during target SID generation while decoupling computational depth from parameter count.
- We propose a recurrent trajectory self-distillation method that aligns both predictive distributions and their evolution directions. It avoids the entanglement between exit selection and trajectory formation and effectively transfers a high-budget teacher into a low-budget student for efficient inference.
- Extensive experiments show that LoopGR outperforms strong baselines with fewer parameters. Beyond overall comparisons, we systematically evaluate recurrent depth, model size and computation scaling. Trajectory-level analyses are conducted to characterize the evolution of predictive representations across recurrent steps.

2 Related Works

Generative Recommendation. Generative recommendation has emerged as a new paradigm that directly generates target items from user-item interaction history Hou et al. (2026); Li et al. (2025). With the rise of Large Language Models (LLMs), one line of studies adapts pretrained LLMs to recommendation by aligning their rich linguistic priors with recommendation-specific objectives Zheng et al. (2024); Team et al. (2026); Zhang et al. (2026). For example, OneRec-Think Liu et al. (2025) introduces itemic alignment, reasoning activation and reasoning enhancement to further equip recommenders with both explicit reasoning capabilities and personalized recommendation. A complementary line forgoes linguistic priors in LLMs and trains a recommendation-tailored architecture from scratch Rajput et al. (2023); Wang et al. (2025b; 2024). OneRec-V2 Zhou et al. (2025b) proposes a lazy decoder-only architecture to reallocate computation toward the target SID generation and avoid repeatedly applying expensive sequence encoding. Despite these advances, most existing methods retain fixed-depth Transformer backbones, where increasing representational capacity typically entails introducing additional layers and parameters. Such parameter-centric scaling is difficult to reconcile with the stringent memory and latency requirements of recommender systems.

Looped Transformers. Looped Transformers increase effective computational depth by repeatedly applying a set of shared layers to decouple the computational depth from model size Bai et al. (2019); Lan et al. (2019). As an earlier study, Universal Transformers Dehghani et al. (2018) proposes a self-attentive recurrent sequence model with Adaptive Computation Time (ACT) Graves (2016) to iteratively refine the symbol representations. More recently, recurrent depth has been revisited in LLMs to improve parameter-efficient scaling and reasoning Yang et al. (2024); Giannou et al. (2023); Fan et al. (2025). As an exemplary implementation, Ouro Zhu et al. (2025) repeatedly reuses a Transformer stack and systematically studies the scaling behavior of looped language models. Huginn Geiping et al. (2026) adopts a different architecture in which the iterative body is tied between prelude and coda stacks. In recommender systems, recurrent depth remains largely underexplored. LoopCTR Tang et al. (2026) uses a parameter-shared prefix-attention module as the recurrent block for CTR prediction. However, its recurrence is tailored to discriminative CTR prediction rather than autoregressive item generation. Deeper recurrence introduces additional sequential computation, making the balance between recurrent depth and inference efficiency an important direction for recommender systems.

3 Preliminaries

Semantic ID-based Generative Recommendation. Let $\mathcal{I} = \{1, \ldots, N\}$ denote the item set and $\mathcal{S}_u = (i_1, \ldots, i_n)$ the user-item interaction history of user u. The goal of next-item recommendation is to predict the subsequent item $i_{n+1} \in \mathcal{I}$. Instead of assigning each item an atomic identifier, semantic ID-based generative recommendation represents an item i by a hierarchical code sequence $\mathbf{c}_i = (c_i^1, \ldots, c_i^H)$, where H is the number of codebooks and c_i^h denotes its code at hierarchy h. Following prior work Lee et al. (2022), these identifiers can be constructed via a residual-quantized variational autoencoder (RQ-VAE), which recursively quantizes item semantic representations into discrete codes. Accordingly, next-item prediction over semantic IDs can be formulated as follows:

$$p_\theta(i_{n+1} \mid \mathcal{S}_u) \equiv p_\theta(\mathbf{c}_{i_{n+1}} \mid \mathcal{S}_u) = \prod_{h=1}^{H} p_\theta\Big(c_{i_{n+1}}^h \mid \mathbf{c}_{i_{n+1}}^{<h}, \mathcal{S}_u\Big), \tag{1}$$

where θ denotes the parameters of the generative recommendation model.

Recurrent Depth. Looped architectures increase computational depth by repeatedly applying parameter-shared transformations. Given an initial hidden representation $\mathbf{H}^{(0)}$, the iterative computation can be written as

$$\mathbf{H}^{(t+1)} = \mathcal{F}_\phi\Big(\mathbf{H}^{(t)}; \mathbf{E}\Big), \qquad t = 0, \ldots, T-1, \tag{2}$$

where $\mathbf{E}$ denotes the conditioning context. $\mathcal{F}\phi$ denotes a recurrent transformation whose parameters ϕ are shared across iterations. T denotes the number of recurrent steps, referred to as the loop budget. The intermediate states naturally form a trajectory in the latent space $\tau = \big(\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \ldots, \mathbf{H}^{(T)}\big)$.

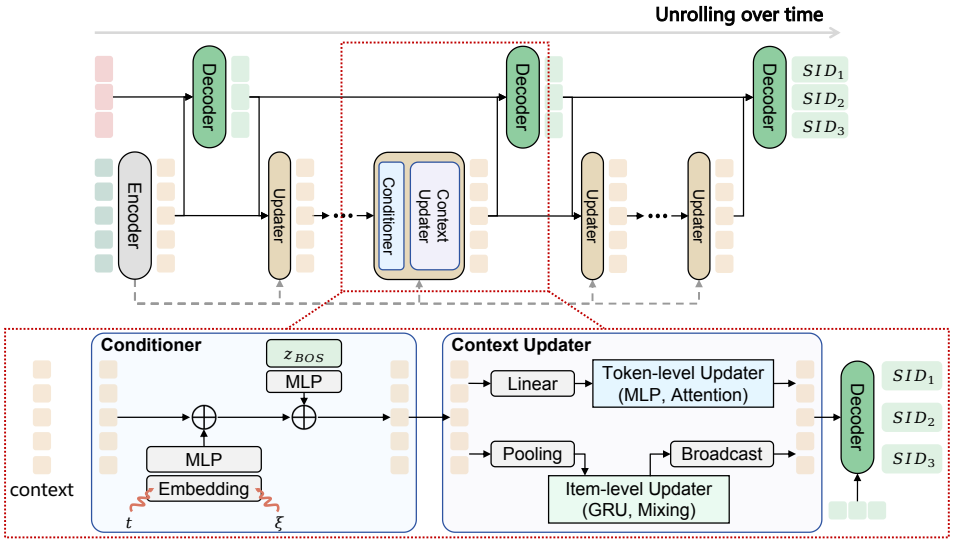


Figure 2: **Overview of LoopGR.** LoopGR alternates a shared decoder with a shared context updater to jointly refine target SID predictions and context representations over hierarchical recurrent depth.

4 METHODOLOGY

In this section, we formally define LoopGR from two complementary perspectives, architectural design and training objective. As illustrated in Figure 2, LoopGR encodes the historical interactions into a contextual memory and then alternates parameter-shared decoding and lightweight context updates to progressively predict target SIDs along recurrent depth. This recurrent computation naturally forms a multi-step prediction trajectory, which we further exploit through a tailored training objective that first optimizes a high-budget teacher and then distills its trajectory into a low-budget student. Consequently, LoopGR defines how a trajectory is formed, while the objective determines how this trajectory is exploited under different inference budgets for generative recommendation.

4.1 LOOPGR ARCHITECTURE

Overall Design. LoopGR organizes target generation as a hierarchical recurrent process consisting of an Encoder, a Context Updater and a Decoder. The hierarchical recurrent computation can be summarized as

$$\mathcal{F} = \mathrm{DEC} \circ \underbrace{\mathcal{B} \circ \cdots \circ \mathcal{B}}_{\text{high-level cycles}} \circ \mathrm{ENC}, \qquad \mathcal{B} := \underbrace{\mathrm{UPD} \circ \cdots \circ \mathrm{UPD}}_{\text{low-level cycles}} \circ \mathrm{DEC}\,. \tag{3}$$

where $\circ$ denotes function composition. The high-level recurrence repeatedly revisits target prediction, while each high-level transition contains a nested low-level recurrence for progressively refining its conditioning context.

Encoder. We represent each historical item by its SIDs and obtain the corresponding token sequence $\mathcal{C}_u = (\mathbf{c}_{i_1}, \ldots, \mathbf{c}_{i_n})$. After embedding these tokens into $\mathbf{X}_u \in \mathbb{R}^{L_u \times d}$, a Transformer encoder constructs the initial contextual memory:

$$\mathbf{M}^{(0)} = \mathrm{ENC}\left(\mathbf{X}_u\right), \tag{4}$$

where L_u denotes the length of the history sequence and d is the hidden dimension.

Context Updater. The context updater is introduced between decoding cycles to progressively refine historical representations via the previous prediction state. Specifically, at the t_H-th high-level cycle and the t_L-th low-level cycle, the updater constructs a cycle-dependent condition and extracts a prediction-aware query from the intermediate hidden states produced by the Decoder as follows:

$$\mathbf{e}^{(t_H,t_L)} = \mathrm{MLP}\left(\mathrm{Emb}(t_H, t_L, \boldsymbol{\xi})\right), \qquad \mathbf{q}^{(t_H)} = \mathrm{MLP}\left(\mathbf{z}_{\mathrm{BOS}}^{(t_H)}\right), \tag{5}$$

where $\mathbf{z}_{\text{BOS}}^{(t_H)}$ denotes the hidden state of the BOS token after the t_H-th decoding cycle, while $\boldsymbol{\xi}$ denotes the hierarchies of SID in the historical sequence. These signals are injected into the context from the previous low-level step, and the resulting conditioned representation is further refined by a lightweight sequence operator:

$$\widetilde{\mathbf{M}}^{(t_H,t_L+1)} = \text{UPD}\left(\widehat{\mathbf{M}}^{(t_H,t_L)}\right) = \text{UPD}\left(\mathbf{M}^{(t_H,t_L)}, \mathbf{e}^{(t_H,t_L)}, \mathbf{q}^{(t_H)}\right), \tag{6}$$

$$\mathbf{M}^{(t_H,t_L+1)} = \text{Norm}\left(\mathbf{M}^{(0)} + \widetilde{\mathbf{M}}^{(t_H,t_L+1)}\right), \tag{7}$$

In our implementation, the updater $\text{UPD}(\cdot)$ shares parameters across all low-level cycles and is instantiated by default as a lightweight GRU-based operator. Notably, the updater is model-agnostic, and we compare different instantiations in Section 5.5. Besides, the initial memory $\mathbf{M}^{(0)}$ provides a stable anchor for recurrent refinement, which facilitates optimization over multiple low-level cycles.

Decoder. The decoder generates recurrent target SID representations from the current historical context. At each autoregressive generation step, the same decoding procedure is repeated for T_H high-level cycles before the resulting prediction is used for subsequent generation. Each decoding cycle is explicitly:

$$\widehat{\mathbf{Z}}^{(t_H)} = \text{Norm}\left(\mathbf{Z}^{(t_H)} + \mathbf{e}^{(t_H)}\right), \tag{8}$$

$$\mathbf{Z}^{(t_H+1)} = \text{DEC}\left(\widehat{\mathbf{Z}}^{(t_H)}, \mathbf{M}^{(t_H)}\right). \tag{9}$$

where $\mathbf{Z}$ denotes the decoder input formed by the BOS token and the target prefix. The condition $\mathbf{e}^{(t_H)}$ encodes the high-level timestep t_H together with the SID type $\boldsymbol{\xi}$ as the conditioning mechanism introduced above. $\mathbf{M}^{(t_H)}$ denotes the contextual memory used at the t_H-th high-level cycle.

4.2 Recurrent Trajectory Self-Distillation

Increasing recurrent depth allows LoopGR to refine its predictions without increasing the parameter count, but deeper unrolling inevitably introduces additional sequential inference cost. Therefore, we propose recurrent trajectory self-distillation to transfer the of a high-budget LoopGR to a low-budget counterpart. Let T_H^{tr} and T_H^{stu} denote the numbers of high-level cycles used by the teacher and student, where $T_H^{stu} < T_H^{tr}$. We first train a high-budget teacher via the last step supervision. This allows the recurrent trajectory to form without explicitly treating intermediate states as candidate exits. We initialize the student from the trained teacher, $\theta_S^{(0)} \leftarrow \theta_T^\star$, and reduce the recurrent budget to T_H^{stu} cycles. The above training objective can be formulated as follows:

$$\underbrace{\theta_T^\star = \arg\min_{\theta_T} \mathcal{L}_{\text{task}}^T,}_{\text{Stage I}} \qquad \underbrace{\theta_S^\star = \arg\min_{\theta_S} \left(\mathcal{L}_{\text{task}}^S + \alpha\mathcal{L}_{\text{Traj}} + \beta\mathcal{L}_{\text{Der}}\right),}_{\text{Stage II}} \tag{10}$$

We align the predictive distributions of the student with the corresponding states along the high-budget teacher trajectory as follows:

$$\mathcal{L}_{\text{Traj}} = \frac{1}{T_H^{\text{stu}} H} \sum_{s=1}^{T_H^{\text{stu}}} \sum_{h=1}^{H} \mathcal{D}_{\text{KL}}\left(\mathbf{p}_{\text{tr}}^{(a(s),h)} \,\|\, \mathbf{p}_{\text{stu}}^{(s,h)}\right). \tag{11}$$

where $a(s) = \left\lceil \frac{sT_H^{tr}}{T_H^{stu}} \right\rceil$. It maps each student cycle to the corresponding state along the teacher trajectory. Following prior trajectory distillation Gong et al. (2025), we further minimize the directional discrepancy based on the first-order change:

$$\mathcal{L}_{\text{Der}} = \frac{1}{(T_H^{\text{stu}} - 1)H} \sum_{s=2}^{T_H^{\text{stu}}} \sum_{h=1}^{H} \mathcal{D}_{\cos}\left(\Delta\mathbf{z}_{\text{tr}}^{(a(s),h)}, \Delta\mathbf{z}_{\text{stu}}^{(s,h)}\right), \tag{12}$$

The finite differences in log-probability space are defined as $\Delta\mathbf{z}_S^{(s,h)} = \log\mathbf{p}_S^{(s,h)} - \log\mathbf{p}_S^{(s-1,h)}$ and $\Delta\mathbf{z}_T^{(a(s),h)} = \log\mathbf{p}_T^{(a(s),h)} - \log\mathbf{p}_T^{(a(s-1),h)}$. Consequently, the student learns to approximate the high-budget refinement process with substantially fewer loop steps.

Table 1: Performance comparison among baselines and LoopGR

Model	Beauty				Sports and Outdoors				CDs and Vinyl			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
GRU4Rec	0.0164	0.0099	0.0283	0.0137	0.0129	0.0086	0.0204	0.0110	0.0195	0.012	0.0353	0.0171
BERT4Rec	0.0203	0.0124	0.0347	0.0170	0.0115	0.0075	0.0191	0.0099	0.0326	0.0201	0.0547	0.0271
SASRec	0.0387	0.0249	0.0605	0.0208	0.0233	0.0154	0.0350	0.0192	0.0351	0.0177	0.0619	0.0263
FDSA	0.0267	0.0163	0.0407	0.0208	0.0182	0.0122	0.0288	0.0156	0.0226	0.0137	0.0378	0.0186
S3-Rec	0.0387	0.0244	0.0647	0.0327	0.0251	0.0161	0.0385	0.0204	0.0213	0.013	0.0375	0.0182
VQ-Rec	0.0434	0.0311	0.0741	0.0372	0.0181	0.0132	0.0251	0.0154	0.0314	0.0209	0.0485	0.0264
TIGER	0.0454	0.0321	0.0648	0.0384	0.0264	0.0181	0.0400	0.0225	0.0492	0.0329	0.0748	0.0411
HSTU	0.0469	0.0314	0.0704	0.0389	0.0258	0.0165	0.0414	0.0215	0.0417	0.0275	0.0638	0.0346
ActionPiece	0.0511	0.0340	0.0775	0.0424	0.0316	0.0205	**0.0500**	0.0264	**0.0544**	0.0359	**0.0830**	**0.0451**
RPG	0.0550	0.0381	0.0809	0.0464	0.0314	0.0216	0.0463	0.0263	0.0498	0.0338	0.0735	0.0415
LoopGR	**0.0563**	**0.0389**	0.0799	**0.0465**	**0.0333**	**0.0223**	0.0495	**0.0275**	0.0537	**0.0363**	0.0799	0.0447
LoopGR-RTSD	0.0561	0.0383	**0.0813**	0.0464	0.0325	0.0213	0.0484	0.0264	0.0516	0.0346	0.0789	0.0434

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. We conduct experiments on three widely used categories from the Amazon Reviews 2014 dataset McAuley et al. (2015): *Sports and Outdoors* (Sports), *Beauty* (Beauty) and *CDs and Vinyl* (CDs). We treat user reviews as implicit interactions and sort them chronologically to construct user behavior sequences. For the training set, we construct next-item prediction instances using sliding windows over each user's interaction history Rajput et al. (2023). More details about the datasets are provided in Appendix A.1.

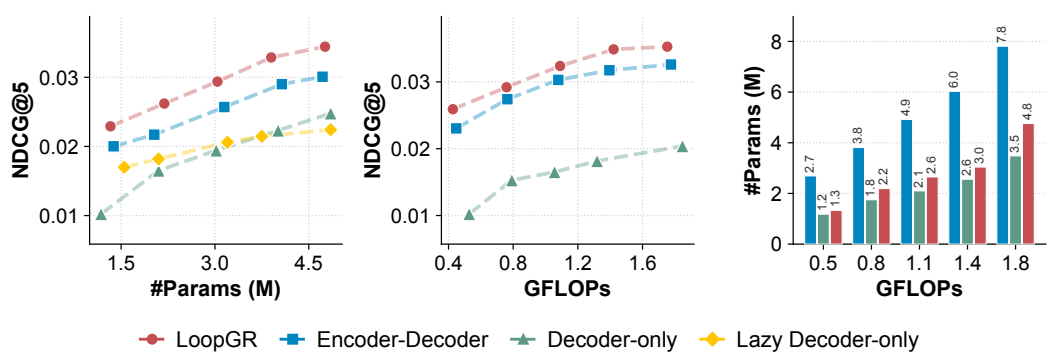
Baselines. We compare LoopGR with representative recommendation methods from two categories. (i) Conventional recommendation methods include GRU4REC (Hidasi et al., 2015), SASREC (Kang & McAuley, 2018), BERT4REC (Sun et al., 2019), FDSA (Zhang et al., 2019), S^3-REC (Zhou et al., 2020) and VQ-REC (Hou et al., 2023). (ii) Generative recommendation methods include TIGER (Rajput et al., 2023), HSTU (Zhai et al., 2024), ACTIONPIECE (Hou et al., 2025b) and RPG (Hou et al., 2025a). We report the best-performing full-budget LOOPGR, together with LOOPGR-RTSD, its distilled student operating at half of the recurrent budget.

Implementation Details. All experiments are conducted on four NVIDIA H20 GPUs. For SID construction, we generate the semantic embedding of items by a pre-trained Sentence-T5 model Ni et al. (2022) and quantize the resulting item embeddings with an RQ-VAE Lee et al. (2022). The RQ-VAE produces a three-digit hierarchical SID for each item. Besides, we append a fourth anti-collision digit to ensure that every item has a unique SID. For the LoopGR backbone, we instantiate the Encoder with T5 Encoder Raffel et al. (2020) and employ a plain Transformer decoder Vaswani et al. (2017) as the parameter-shared Decoder.

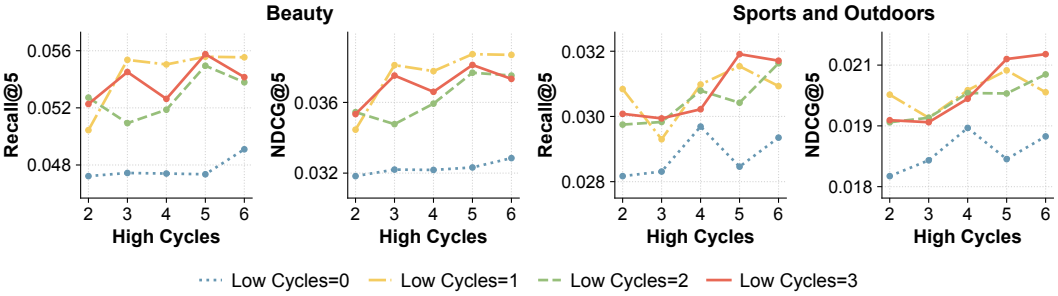
5.2 OVERALL PERFORMANCE

Table 1 compares LoopGR with conventional sequential recommenders and recent generative recommendation methods. Overall, generative approaches achieve stronger performance across the three datasets though several conventional methods remain competitive on individual metrics. By contrast, generative methods exhibit stronger and more consistent performance, demonstrating the benefit of directly generating compositional item identifiers. **LoopGR and LoopGR (RTSD) achieve the best result on 8 of the 12 evaluation settings.** In particular, LoopGR improves over the strongest baseline by 2.4% in Recall@5 and 2.1% in NDCG@5 for Beauty, and by 5.4% in Recall@5 and 4.2% in NDCG@10 for Sports and Outdoors.

For generative baselines, ActionPiece and RPG employ more elaborate identifier or decoding designs. ActionPiece constructs context-dependent action tokens, whereas RPG adopts long unordered SIDs together with parallel prediction and graph-constrained decoding. In contrast, LoopGR adopts a standard SID construction using RQ-VAE, while improving model capacity through a tailored hierarchical recurrent depth for generative recommendation. LoopGR outperforms RPG on 11 of



(a) Model scaling under varying parameter and computation budgets on CDs.



(b) Effect of high-level cycles T_H and low-level cycles T_L cycles on Beauty and Sports.

Figure 3: Architectural comparison across model size, computation and recurrent depth scaling.

12 metrics and ActionPiece on 10 of 12 metrics. Moreover, LoopGR (RTSD) remains competitive while using fewer refinement steps and obtains the best Hit@10 on Beauty. It validates the effectiveness of trajectory self-distillation in iterative reasoning to a lower-computation model.

5.3 Architecture Comparison

For further investigating model size and recurrent depth scaling, we compare LoopGR with three representative generative backbones, including Encoder-Decoder, Decoder-only and Lazy Decoder-only baselines. For parameter scaling, we fix $T_H = 3$ and $T_L = 1$ to compare LoopGR with the baselines at comparable parameter counts. For computation scaling, we search over encoder/decoder depths and T_H/T_L to report the best configuration at comparable GFLOPs. Lazy Decoder-only is excluded from computation scaling because its computation is substantially lower than that of the other architectures. For hierarchical recurrence analysis, we fix both encoder and decoder to one layer and vary T_H and T_L. More details about the inference latency are provided in Appendix A.2.

Parameter Scaling. As shown in Figure 3 (a), LoopGR achieves higher NDCG@5 than the alternative architectures at similar parameter counts. Its performance continues to improve as the model size increases, while recurrent computation provides an additional dimension for increasing effective depth without introducing new parameters at every step. Consequently, its performance continues to improve beyond Encoder-Decoder as the parameter budget increases. In contrast, Lazy Decoder-only exhibits a flatter scaling curve because the historical context remains fixed and additional parameters are devoted primarily to target decoding. These results suggest that hierarchical recurrence provides a more parameter-efficient scaling strategy for SID-based generative recommendation.

Computation Scaling. LoopGR achieves the highest NDCG@5 across the different computation levels while requiring substantially fewer parameters than Encoder-Decoder. At approximately 0.5 GFLOPs, LoopGR uses only 1.3M parameters, less than half of the 2.7M parameters required by Encoder-Decoder. Notably, with a parameter budget close to Decoder-only, LoopGR still outper-

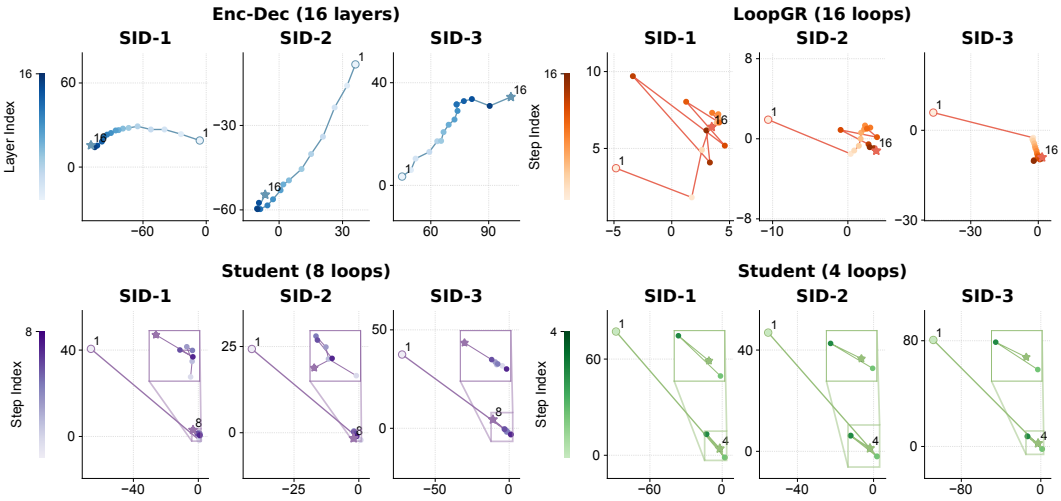


Figure 4: Latent space trajectories for different SIDs.

forms the considerably larger Encoder-Decoder model. This demonstrates that additional computation can be introduced through parameter reuse rather than proportional model expansion, yielding a more favorable trade-off among accuracy, computation and model size.

Effect of Hierarchical Recurrent Depth. We further vary the high-level depth T_H and low-level depth T_L to examine their respective roles. Figure 3 (b) shows that removing low-level recurrence $T_L = 0$ significantly degrades performance, while introducing even a small number of updater cycles improves both Recall@5 and NDCG@5. It highlights the importance of dynamically refining the historical context between successive decoding cycles. Increasing T_H generally brings further gains, although the gains become non-monotonic at larger depths. On Sports, increasing both T_H and T_L yields more pronounced performance gains. This indicates that deeper hierarchical recurrence becomes increasingly beneficial as the recommendation scale grows. Overall, these results support the hierarchical design of LoopGR, where high-level cycles progressively refine target prediction while low-level cycles adapt the conditioning context between decoding stages.

5.4 Analysis of Loop Dynamics

In this section, we visualize intermediate representation trajectories in latent space across recurrent steps. We focus on two questions: **(Q1)** how do the latent dynamics of LoopGR differ from those of a conventional Encoder-Decoder? and **(Q2)** how does trajectory distillation reshape these dynamics under a limited loop budget? We collect the intermediate representations at different SID positions from a 16-layer Encoder-Decoder, a 16-loop LoopGR teacher and distilled students with 8 and 4 loops. For each SID position, we project the intermediate states onto the first two principal components to trace their evolution. Figure 4 provides a visualization of these trajectories.

Q1: How do the latent dynamics of LoopGR differ from a conventional counterpart? As depicted in Figure 4, the Encoder-Decoder exhibits relatively smooth layer-wise evolution. Its adjacent representations progressively move along a largely continuous trajectory. In contrast, LoopGR undergoes substantially larger transitions in its early cycles and subsequently enters a more localized refinement regime. In the experiments, LoopGR exhibits clear position-dependent dynamics that are characterized by stronger oscillations at earlier SID positions and faster convergence at later SID positions. These non-monotonic patterns resemble the structured latent trajectories observed in previous Looped Transformers Geiping et al. (2026); Wang et al. (2025a). Unlike the progressive layer-wise transformations of the Encoder-Decoder, LoopGR repeatedly revises its latent state across cycles, yielding a more dynamic and position-dependent refinement process.

Q2: What does trajectory distillation learn? Figure 4 shows that reducing the loop budget through distillation does not simply truncate the teacher trajectory. The 8-loop student exhibits a

Table 2: Ablation study of different updater architectures and training objectives. The best and second-best results in each column are highlighted in bold and underlined, respectively.

Model	Sports and Outdoors				CDs and Vinyl			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
LoopGR	0.0333	**0.0223**	0.0495	0.0275	**0.0537**	**0.0363**	**0.0799**	**0.0447**
w. Identity	0.0311	0.0197	0.0489	0.0254	0.0502	0.0324	0.0730	0.0400
w. Mixing	**0.0342**	<u>0.0222</u>	**0.0531**	**0.0283**	0.0527	0.0352	<u>0.0792</u>	0.0436
w. Linear Attn	<u>0.0338</u>	0.0219	0.0516	0.0276	**0.0537**	0.0354	0.0789	0.0435
w. Vanilla Attn	0.0333	0.0217	<u>0.0520</u>	<u>0.0277</u>	<u>0.0533</u>	<u>0.0362</u>	0.0768	<u>0.0437</u>
w. Per-Step Supervision	0.0280	0.0186	0.0424	0.0232	0.0516	0.0348	0.0771	0.0430
w. Gated Exit	0.0264	0.0174	0.0406	0.0220	0.0502	0.0338	0.0754	0.0420
w. ILSD	0.0290	0.0192	0.0427	0.0236	0.0512	0.0347	0.0752	0.0424
w. RTSD	0.0325	0.0213	0.0484	0.0264	0.0516	0.0346	0.0789	0.0434

much larger transition at the beginning of recurrent computation, followed by progressively smaller updates toward its terminal state. This tendency becomes even more pronounced for the 4-loop student, whose trajectory is dominated by the initial transition. Here, we propose an intuitive interpretation that the distillation objective encourages a low-budget model to approximate a longer refinement process with fewer recurrent transitions rather than reproducing every intermediate teacher state. Consequently, the teacher and student exhibit complementary computational behaviors. The long-budget teacher develops a richer refinement trajectory, whereas the distilled student learns a more direct trajectory toward the terminal prediction under a limited inference budget.

5.5 Ablation Study

We conduct ablation studies on two key design choices of LoopGR, namely the context updater and the training objective. For the updater, we replace the default GRU with three sequence modeling methods, including token mixing, linear attention, vanilla attention and an Identity variant that leaves the context representations unchanged. Furthermore, we compare per-step supervision, gated exit, Intra-Loop Self-Distillation (ILSD) Goyal et al. (2026) and Recurrent Trajectory Self-Distillation (RTSD). At inference, all alternative training objectives use the full budget, and RTSD distills LoopGR into a student operating at half of the teacher's budget.

Table 2 shows that the Mixing baseline achieves the best Recall@5, Recall@10 and NDCG@10 on Sports. In contrast, replacing the updater with Identity consistently degrades performance, with an average drop of about 8% relative to the learnable updater variants. This confirms that the gain primarily comes from dynamically updating the conditioning context rather than from a particular sequence operator, making the updater design largely architecture-agnostic. Averaged over the eight reported metrics, per-step supervision, gated exit and ILSD fall 9.7%, 13.2% and 9.4% below the full-budget LoopGR, respectively. RTSD reduces this gap to only 3.2%. We suggest that direct supervision at intermediate cycles can induce representation stagnation. It causes subsequent updates to diminish before the full recurrent budget is exploited. RTSD instead separates trajectory learning from compression and better preserves performance under a reduced budget.

6 Conclusion

We introduced **LoopGR**, a parameter-efficient framework for generative recommendation that decouples computational depth from parameter growth through hierarchical recurrent computation. By recurrently reusing a decoder and a context updater, LoopGR increases effective depth while dynamically refining the historical context during SID generation. We further develop recurrent trajectory self-distillation to better balance recommendation performance and inference efficiency under hard recurrent budgets. Extensive experiments show that LoopGR consistently outperforms strong baselines and achieves better performance than representative Encoder-Decoder architectures with up to 52% fewer parameters. These results demonstrate the potential of recurrent depth as a complementary scaling dimension beyond model size for generative recommendation.

AI USE STATEMENT

In this work, we used generative AI tools only to assist with language editing and polishing of the manuscript, including improving grammar, clarity and readability. We have not used generative AI tools for research ideation, hypothesis formulation, methodology or experiment design, method implementation, data generation or processing, result analysis or interpretation, theoretical development or proof construction. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical accuracy and consistency with the authors' intended meaning. The research ideas, methods, experiments, analyses and conclusions were developed and carried out by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, volume 2025, pp. 14502–14520, 2025.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2026.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Guoqiang Gong, Jiaxing Wang, Jin Xu, Deping Xiang, Zicheng Zhang, Leqi Shen, Yifeng Zhang, JunhuaShu JunhuaShu, ZhaolongXing ZhaolongXing, Zhen Chen, et al. Beyond logits: Aligning feature dynamics for effective knowledge distillation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23067–23077, 2025.

Sahil Goyal, Swayam Agrawal, Gautham Govind Anil, Prateek Jain, Sujoy Paul, and Aditya Kusupati. Elt: Elastic looped transformers for visual generation. *arXiv preprint arXiv:2604.09168*, 2026.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Wei Guo, Hao Wang, Luankang Zhang, Jin Yao Chin, Zhongzhou Liu, Kai Cheng, Qiushi Pan, Yi Quan Lee, Wanqi Xue, Tingjia Shen, et al. Scaling new frontiers: Insights into large recommendation models. *arXiv preprint arXiv:2412.00714*, 2024.

Yingzhi He, Yan Sun, Junfei Tan, Yuxin Chen, Xiaoyu Kong, Chunxu Shen, Xiang Wang, An Zhang, and Tat-Seng Chua. Reasoning over semantic ids enhances generative recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 1638–1649, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3818122. URL https://doi.org/10.1145/3770855.3818122.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Min Hou, Le Wu, Yuxin Liao, Yonghui Yang, Zhen Zhang, Yu Wang, Changlong Zheng, Han Wu, and Richang Hong. A survey on generative recommendation: Data, model, and tasks, 2026. URL https://arxiv.org/abs/2510.27157.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pp. 1162–1171, 2023.

Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 956–966, 2025a.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian McAuley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. *arXiv preprint arXiv:2502.13581*, 2025b.

Clark Mingxuan Ju, Tong Zhao, Leonardo Neves, Liam Collins, Bhuvesh Kumar, Jiwen Ren, Lili Zhang, Wenfeng Zhuo, Vincent Zhang, Xiao Bai, Jinchao Li, Karthik Iyer, Zihao Fan, Yilun Xu, Yiwen Chen, Peicheng Yu, Manish Malik, and Neil Shah. Semantic ids for recommender systems at snapchat: Use cases, technical challenges, and design choices, 2026. URL `https://arxiv.org/abs/2604.03949`.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11513–11522. IEEE, 2022.

Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. From matching to generation: A survey on generative information retrieval. *ACM Transactions on Information Systems*, 43(3):1–62, 2025.

Jingzhe Liu, Liam Collins, Jiliang Tang, Tong Zhao, Neil Shah, and Mingxuan Ju. Understanding generative recommendation with semantic ids from a model-scaling view. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3234–3245, 2026.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, et al. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*, 2025.

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*, pp. 43–52, 2015.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the association for computational linguistics: ACL 2022*, pp. 1864–1874, 2022.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025.

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*, pp. 1441–1450, 2019.

Jiakai Tang, Runfeng Zhang, Weiqiu Wang, Yifei Liu, Chuan Wang, Xu Chen, Yeqiu Yang, Jian Wu, Yuning Jiang, and Bo Zheng. Loopctr: Unlocking the loop scaling power for click-through rate prediction. *arXiv preprint arXiv:2604.19550*, 2026.

OneRec Team, Biao Yang, Boyang Ding, Chenglong Chu, Dunju Zang, Fei Pan, Han Li, Hao Jiang, Honghui Bao, Huanjie Wang, Jian Liang, Jiangxia Cao, Jiao Ou, Jiaxin Deng, Jinghao Zhang, Kun Gai, Lu Ren, Peiru Du, Pengfei Zheng, Rongzhou Zhang, Ruiming Tang, Shiyao Wang, Siyang Mao, Siyuan Lou, Teng Shi, Wei Yuan, Wenlong Xu, Xingchen Liu, Xingmei Wang, Xinqi Jin, Yan Sun, Yan Wang, Yifei Hu, Yingzhi He, Yufei Ye, Yuhao Wang, Yunhao Zhou, Yuqin Dai, Zhao Liu, Zhipeng Wei, Zhixin Ling, Ziming Li, Zixing Zhang, Ziyuan Liu, An Zhang, Changxin Lao, Chaoyi Ma, Chengru Song, Defu Lian, Fan Yang, Guowang Zhang, Hao Peng, Jiayao Shen, Jie Chen, Jun Xu, Junmin Chen, Kun Zhang, Kuo Cai, Mingxing Wen, Minmao Wang, Minxuan Lv, Qi Zhang, Qiang Luo, Sheng Yu, Shijie Li, Shijie Yi, Shuang Yang, Shugui Liu, Shuni Chen, Tinghai Zhang, Tingting Gao, Xiang Wang, Xiangyu Wu, Xiangyu Zhao, Xiao Lv, Xiaoyou Zhou, Xuming Wang, Yong Du, Zejian Zhang, Zhaojie Liu, Zhiyang Zhang, Zhuang Zhuang, Ziqi Wang, and Ziyi Zhao. Onereason technical report, 2026. URL https://arxiv.org/abs/2606.06260.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025a. URL https://arxiv.org/abs/2506.21734.

Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation, 2025b. URL https://arxiv.org/abs/2405.07314.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, and Zhenhua Dong. Eager: Two-stream generative recommender with behavior-semantic collaboration, 2024. URL https://arxiv.org/abs/2406.14017.

Chen Yang, Guangyue Peng, Jiaying Zhu, Ran Le, Ruixiang Feng, Tao Zhang, Wei Ruan, Xiaoqi Liu, Xiaoxue Cheng, Xiyun Xu, et al. Nanbeige4-3b technical report: Exploring the frontier of small language models. *arXiv preprint arXiv:2512.06266*, 2025a.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International conference on learning representations*, volume 2024, pp. 42195–42214, 2024.

Yuhao Yang, ZhI JI, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, shuanglong li, and LIU LIN. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 93746–93770. Curran Associates, Inc., 2025b. doi: 10.52202/085713-3135. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/86ba836d4c5dd859d795a172911745e2-Paper-Conference.pdf.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. Feature-level deeper self-attention network for sequential recommendation. In *Ijcai*, pp. 4320–4326, 2019.

Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced latent reasoning for llm-based recommendation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 128449–128470, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d0c6bc641a56bebee9d985b937307367-Paper-Conference.pdf.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 2024.

Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. Onerec technical report. *arXiv preprint arXiv:2506.13695*, 2025a.

Guorui Zhou, Hengrui Hu, Hongtao Cheng, Huanjie Wang, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Lu Ren, Liao Yu, Pengfei Zheng, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Ruiming Tang, Shiyao Wang, Shujie Yang, Tao Wu, Wuchao Li, Xinchen Luo, Xingmei Wang, Yi Su, Yunfan Wu, Zexuan Cheng, Zhanyu Liu, Zixing Zhang, Bin Zhang, Boxuan Wang, Chaoyi Ma, Chengru Song, Chenhui Wang, Chenglong Chu, Di Wang, Dongxue Meng, Dunju Zang, Fan Yang, Fangyu Zhang, Feng Jiang, Fuxing Zhang, Gang Wang, Guowang Zhang, Han Li, Honghui Bao, Hongyang Cao, Jiaming Huang, Jiapeng Chen, Jiaqiang Liu, Jinghui Jia, Kun Gai, Lantao Hu, Liang Zeng, Qiang Wang, Qidong Zhou, Rongzhou Zhang, Shengzhe Wang, Shihui He, Shuang Yang, Siyang Mao, Sui Huang, Tiantian He, Tingting Gao, Wei Yuan, Xiao Liang, Xiaoxiao Xu, Xugang Liu, Yan Wang, Yang Zhou, Yi Wang, Yiwu Liu, Yue Song, Yufei Zhang, Yunfeng Zhao, Zhixin Ling, and Ziming Li. Onerec-v2 technical report, 2025b. URL https://arxiv.org/abs/2508.20900.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1893–1902, 2020.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Appendix

A.1 Experimental Details

We conduct experiments on three public Amazon review datasets, including Beauty, Sports and Outdoors and CDs and Vinyl. As summarized in Table 3, the datasets span different scales and interaction densities. Beauty contains 22,363 users, 12,101 items and 176,139 interactions, while Sports and Outdoors includes 18,357 users, 35,598 items and 260,739 interactions. CDs and Vinyl is the largest dataset with 75,258 users, 64,443 items and over one million interactions. The average interaction lengths are 8.87, 8.32 and 14.58 for Beauty, Sports and Outdoors and CDs and Vinyl, respectively.

Table 3: Statistics of the datasets.

Dataset	**#Users**	**#Items**	**#Interactions**	**Avg.** t
Beauty	22,363	12,101	176,139	8.87
Sports	18,357	35,598	260,739	8.32
CDs	75,258	64,443	1,022,334	14.58

A.2 Efficiency under Compressed Recurrent Trajectories

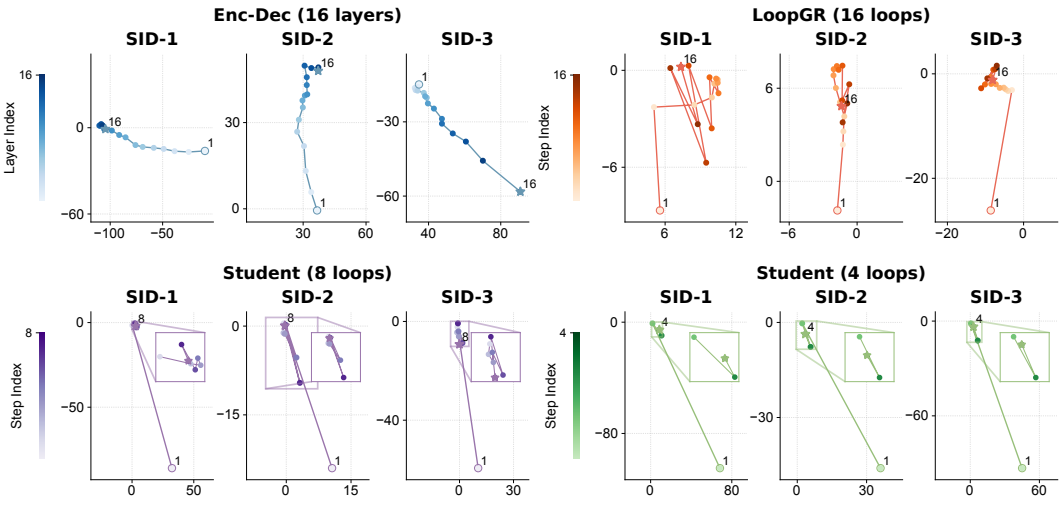
We further evaluate whether RTSD can preserve recommendation quality when the recurrent trajectory is compressed to fewer cycles. Reducing the teacher from 16 cycles to 8 and 4 cycles decreases computation from 17.408 GFLOPs to 8.415 and 3.918 GFLOPs, respectively. The corresponding inference latency drops from 211.08 ms to 104.01 ms and 49.82 ms under batch size 1, providing approximately 2.0× and 4.2× speedups. Despite the substantially shorter trajectories, both students retain comparable recommendation accuracy. Notably, the 4-cycle student surpasses the 16-cycle teacher on all four metrics for Sports and remains competitive on Beauty, while the 8-cycle student achieves the best Hit@10 and NDCG@10 on Beauty. These results show that RTSD can compress a long recurrent trajectory into substantially fewer cycles without proportionally sacrificing predictive quality, enabling a more favorable trade-off between recurrent depth and inference efficiency.

Table 4: Performance–efficiency comparison between the full-budget LoopGR teacher and RTSD students with reduced recurrent cycles. Inference latency is measured with batch size 1.

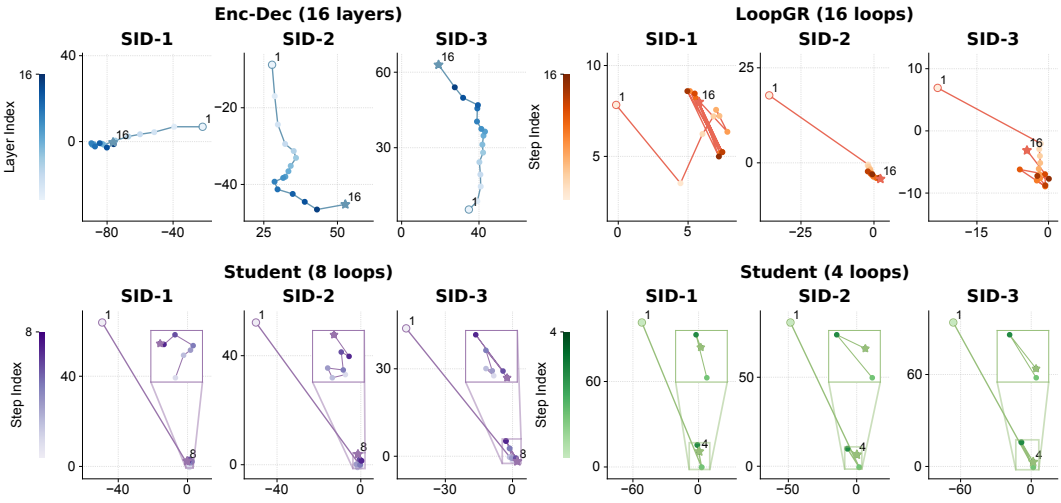
Model	**GFLOPs** ↓	**Latency (ms)** ↓	**Sports and Outdoors**				**Beauty**			
			H@5	**N@5**	**H@10**	**N@10**	**H@5**	**N@5**	**H@10**	**N@10**
LoopGR (Teacher, 16-loop)	17.408	211.08	0.0304	0.0199	0.0456	0.0248	0.0531	**0.0369**	0.0775	0.0448
LoopGR-RTSD (8-loop)	8.415	104.01	0.0295	0.0195	0.0465	0.0250	0.0530	**0.0369**	**0.0791**	**0.0453**
LoopGR-RTSD (4-loop)	**3.918**	**49.82**	**0.0315**	**0.0209**	**0.0479**	**0.0261**	**0.0534**	0.0368	0.0789	0.0450

A.3 Additional Visualization of Recurrent Trajectories

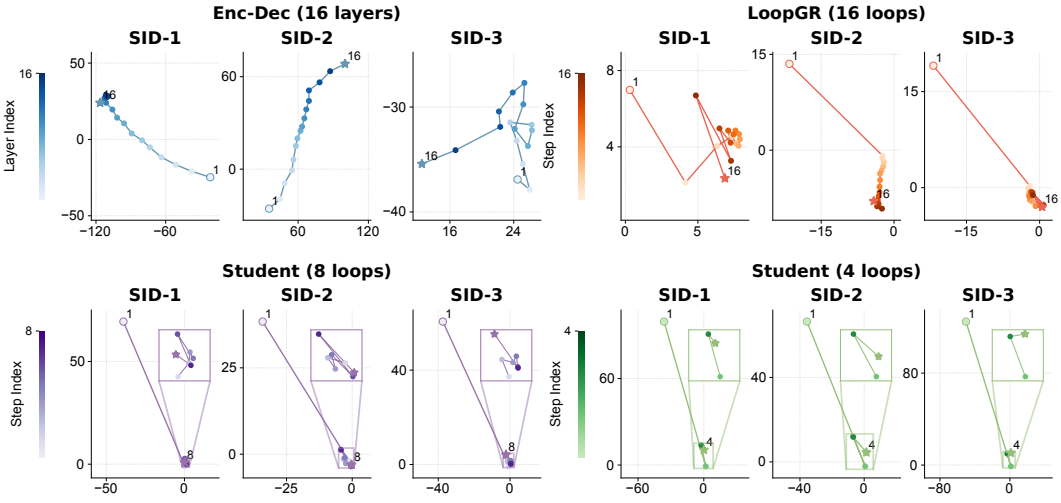
We provide additional trajectory visualizations to complement the analysis in the main text. Figure 5 shows representative examples across the three SID positions, comparing the 16-layer Encoder–Decoder, the 16-loop LoopGR teacher, and the distilled students with 8 and 4 recurrent cycles. Across different samples, LoopGR exhibits iterative refinement throughout recurrent computation, while the distilled students reach similar terminal regions through substantially shorter trajectories. These examples further illustrate that RTSD compresses the recurrent refinement process rather than simply truncating the teacher trajectory.



(a) Sample 0



(b) Sample 1



(c) Sample 2

Figure 5: Visualization of recurrent trajectories for different recommendation samples.