PLAN ONCE, PREDICT ALL: LATENT PLANNING FOR NON-AUTOREGRESSIVE USER BEHAVIOR FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modeling future user behavior trajectories is crucial for capturing evolving preferences and enabling proactive recommendations. However, most sequential recommenders either focus on next-item prediction or extend to multi-step forecasting via autoregressive rollout, which often incurs error accumulation and high inference cost. To alleviate these problems, we propose LapTop, a latent planning framework for user behavior trajectory prediction that generates multi-step actions through parallel trajectory decoding without autoregressive rollout. LapTop introduces a compact set of latent plan tokens that operate in the hidden space to encode long-term intent and global preference structure, together with learnable future query slots that decode all future steps simultaneously. To capture the inherent uncertainty of user behaviors, LapTop adopts a variational learning scheme that aligns a history-conditioned prior with a future-aware posterior during training, while using only the prior at inference time. This design enables diverse and coherent trajectory generation without iterative decoding. Experiments on real-world benchmarks demonstrate that LapTop consistently outperforms state-of-the-art baselines in multi-step accuracy and trajectory quality, while also reducing inference latency.

1 INTRODUCTION

Sequential recommendation models user preferences from interaction histories, with Transformer-based methods such as SASRec (Kang & McAuley, 2018; Huang et al., 2023a;b) achieving strong performance in next-item prediction. However, many aSeq-NLLications require forecasting not only the next interaction but an entire *future behavior trajectory*, such as the next k session items, short-term purchase paths, or longer-horizon user intents for downstream planning (Ning et al., 2025; Huang et al., 2025a; 2024; 2025b). This task, known as *user behavior trajectory prediction* (UBTP) (Pancha et al., 2022; Huang et al., 2025c), is more challenging than next-item prediction because it must capture long-horizon dependencies and evolving structure across future actions.

A common approach to multi-step prediction is *autoregressive rollout*, which repeatedly aSeq-NLLies a next-item model and feeds predictions back as inputs (Kang & McAuley, 2018; Rajput et al., 2023). Despite its simplicity, rollout suffers from exposure bias and error accumulation, as early mistakes distort later inputs and degrade long-horizon trajectory quality. It also requires k sequential forward passes for k future steps, leading to high inference latency. These limitations motivate more efficient UBTP formulations that generate multiple future steps with stronger long-horizon fidelity.

Recent work has moved toward *trajectory-level* UBTP, especially diffusion-based methods that generate future sequences and use listwise supervision to align predicted trajectories with ground truth (Wei & Fang, 2025; Yang et al., 2023; Huang et al., 2025c). While effective for long-horizon modeling, diffusion-style generation typically relies on iterative denoising or repeated sampling, as shown in Figure 1(c), incurring substantial overhead and limiting deployment scalability. This raises the question: *Can we enable trajectory-level modeling and one-to-many future prediction within a lightweight sequential framework, without relying on iterative generation at inference time?*

In this work, we revisit UBTP from the perspective of *hidden-space reasoning* in standard Transformer-based sequential recommenders. Our key insight is that multi-step futures are often *multi-modal*: the same history may lead to different plausible trajectories depending on latent

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.

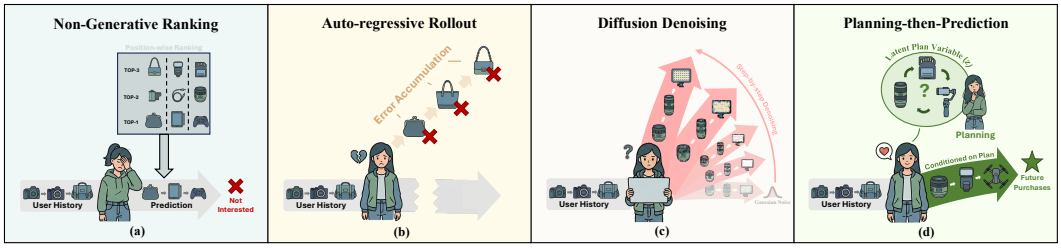


Figure 1: Comparison of paradigms for user behavior trajectory prediction. (a) Direct prediction with non-generative ranking. (b) Auto-regressive rollout. (c) Diffusion-based denoising generation. (d) Planning-then-Prediction (ours).

factors such as intent shifts, exploration versus exploitation, or short-term goals (He & McAuley, 2016; Huang et al., 2025c). Instead of forcing a deterministic rollout or a single-mode trajectory predictor, we introduce a *latent planning* variable that allows the model to first "plan" in a compact latent space and then generate future steps conditioned on that plan.

Concretely, we propose **LapTop**, a latent planning trajectory predictor that integrates (i) *parallel future query slots* to predict k future steps in one parallel decoding pass, and (ii) *latent plan tokens* injected with a variational latent variable z to enable *plan-then-predict* reasoning. During training, we learn a posterior $q_\phi(z \mid x, y)$ conditioned on the history x and the ground-truth future trajectory y, together with a history-only prior $p_\psi(z \mid x)$. At inference time, the model samples (or deterministically uses) the prior plan and produces the entire k-step trajectory in parallel, combining efficiency with multi-modal expressivity. This design offers several practical benefits. First, parallel trajectory prediction mitigates error accumulation and substantially reduces inference cost compared to autoregressive rollout. Second, latent planning provides an explicit mechanism to capture one-to-many futures and supports sampling diverse trajectories when needed. Third, the approach preserves the simplicity and deployability of SASRec-style recommenders (Kang & McAuley, 2018), avoiding iterative generation while still benefiting from trajectory-aware modeling. Extensive experiments on multiple real-world sequential recommendation benchmarks demonstrate that LapTop consistently improves both step-wise ranking metrics and trajectory-level metrics compared to strong Transformer-based baselines (Kang & McAuley, 2018) and diffusion-based UBTP methods (Huang et al., 2025d;c), while significantly reducing inference latency. Our main contributions are summarized as follows:

- We revisit user behavior trajectory prediction within a lightweight formulation and propose a parallel k-step prediction mechanism via future query slots.
- We introduce a posterior-to-prior latent planning mechanism that learns future-aware trajectory modes during training and distills them into a history-conditioned prior for inference. Unlike directly conditioning each future slot on a latent vector, LapTop injects the latent plan through causal plan tokens, which act as a shared interface for coordinating parallel horizon queries.
- We conduct extensive empirical evaluation and analysis, demonstrating consistent gains over strong Transformer sequential recommenders and diffusion-based trajectory predictors, together with substantial inference efficiency improvements.

2 Related Work

User Behavior Modeling. User behavior modeling is a fundamental component of recommender systems. Conventional sequential recommendation methods (Hidasi et al., 2016; Kang & McAuley, 2018; Huang et al., 2023c;a) primarily focus on next-item prediction by leveraging short-term interaction patterns, but often fall short in representing long-term user preferences or complex multi-step behaviors. To address these limitations, recent studies have explored the prediction of user behavior trajectories across multiple future time steps or conversion stages. Graph Multi-Scale Pyramid Networks (GMPN) (Huang et al.) model purchase behavior by capturing temporal dynamics at multiple resolutions as well as hierarchical category dependencies. The Adaptive Intent Transfer Model (AITM) (Xi et al., 2021) characterizes sequential dependencies in multi-stage conversion

funnels through adaptive feature transfer mechanisms. PinnerFormer (Pancha et al., 2022) adopts a Transformer-based architecture to forecast diverse forms of user engagement in a unified manner.

Generative Models in User Behavior Prediction. Unlike earlier generative paradigms such as VAEs and GANs, diffusion models (Ho et al., 2020) generate samples through iterative denoising that gradually inverts a predefined noise injection process. This multi-step refinement enables strong distributional modeling and has shown strong empirical performance across generative tasks. Recently, diffusion-based approaches have become a dominant direction for user behavior modeling and generative recommendation (Wei & Fang, 2025; Yang et al., 2024; Ye et al., 2025). DiffRec (Wang et al., 2023) aSeq-NLLies diffusion processes to multi-hot user interaction vectors for collaborative recommendation. DiffuRec (Li et al., 2024b) uses a Transformer to recover target item representations from noisy histories. DCDR (Lin et al., 2024) introduces discrete diffusion for progressive re-ranking. DreamRec (Yang et al., 2023) generates oracle next-item embeddings reflecting user preferences, but suffers from limited scalability due to the lack of negative sampling. PreferDiff (Liu et al., 2025) adapts the BPR objective for diffusion-based learning, while DCRec (Huang et al., 2025d) improves preference alignment with implicit and explicit conditioning signals. Finally, LPDO (Huang et al., 2025c) extends diffusion-based recommendation beyond next-item prediction by injecting listwise ranking supervision, enabling coherent long-horizon user behavior trajectories that preserve preference consistency.

Although promising, these diffusion-based approaches remain less suitable for user behavior trajectory prediction. Most existing methods are optimized in a largely position-wise or step-wise manner, making it difficult to explicitly capture cross-step dependencies and trajectory-level consistency. Moreover, iterative denoising or sampling incurs substantial inference overhead, especially for long horizons. These limitations motivate a more efficient trajectory-level modeling paradigm that jointly captures global future intent and generates multi-step predictions through non-iterative parallel decoding.

3 Problem Formulation

Following previous work (Huang et al., 2025c), let a user interaction history be denoted by:

$$x = (i_1, i_2, \ldots, i_T), \tag{1}$$

where $i_t \in \{0, 1, \ldots, |\mathcal{I}|\}$ is the item identifier at position t, and $i_t = 0$ denotes padding. Given x, the task of *user behavior trajectory prediction* (UBTP) is to predict a future sequence of length H:

$$y = (i_{T+1}, i_{T+2}, \ldots, i_{T+H}), \tag{2}$$

by learning a conditional distribution $p(y \mid x)$, where dependencies across future steps are modeled through shared representations rather than autoregressive feedback. If the observed future is shorter than H, we pad it with 0 and exclude padded positions from the loss and evaluation. Unlike standard next-item recommendation, which models a single-step conditional distribution $p(i_{T+1} \mid x)$, UBTP evaluates multi-step predictions at the trajectory level, where quality depends on the joint correctness and coherence across all H steps. This setting poses two key challenges. First, the model must capture long-horizon dependencies within the future trajectory, where later actions correlate with (and are constrained by) earlier future actions in the underlying user behavior. Second, the future trajectory is often *inherently multi-modal*: the same history x may lead to multiple plausible future sequences due to latent factors such as intent shifts, exploration behavior, or short-term goals. Therefore, a practical predictor should represent multiple modes under the same history and support generating diverse yet plausible trajectories. Our objective is to learn a trajectory-level predictor that (i) generates coherent H-step futures conditioned on history, (ii) accounts for the multi-modality of possible trajectories, and (iii) enables non-iterative inference for a full H-step trajectory.

4 Methodology

4.1 Overview of LapTop

User behavior forecasting is inherently *one-to-many*: under the same history, multiple *trajectory-level* futures can be plausible due to unobserved intents and short-term goals. Autoregressive rollout further

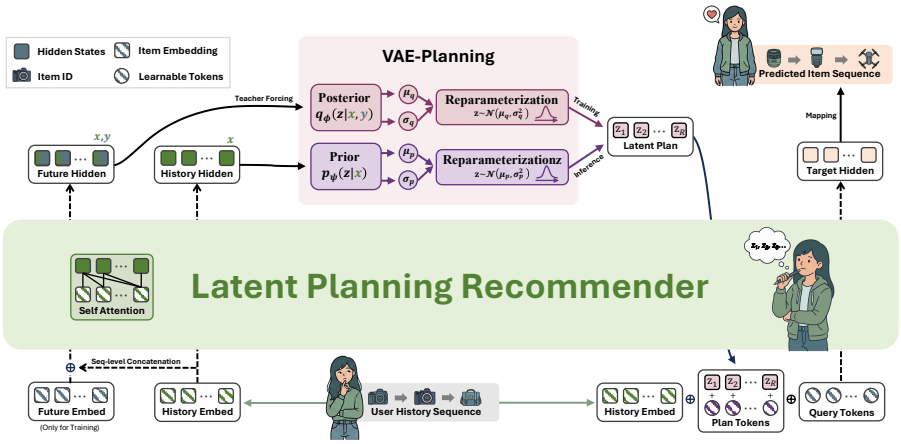


Figure 2: LapTop performs non-iterative H-step trajectory prediction via a plan-then-predict design. It first infers a history-conditioned latent plan and then uses R plan tokens as a shared planning interface to mediate history-to-future information flow across all horizons. Meanwhile, H future query tokens decode all H steps in one parallel trajectory decoding pass through self-attention. Multi-modality is captured by a variational latent plan z learned by aligning a history-conditioned prior with a future-aware posterior; at test time, drawing different z from the prior yields multiple plausible trajectories without autoregressive rollout or iterative denoising.

amplifies this uncertainty by feeding back its own predictions, which not only accumulates errors but also requires H sequential passes to produce a H-step trajectory. To address both, we propose **LapTop**, a SASRec-style Transformer that decodes all H future steps in parallel *through a shared planning interface*, and uses an explicit latent variable to control multimodal future intents.

As illustrated in Figure 2, LapTop comprises three components: (i) **parallel future query slots** $F_{1:H}$, namely H learnable query tokens that read out logits for H future steps in one parallel decoding pass; although decoded in parallel, these slots interact via self-attention over the augmented sequence rather than assuming conditional independence across horizons; (ii) **latent plan tokens** $Z_{1:R}$, i.e., R learnable tokens inserted between the history and future queries that act as a shared hidden-space *mediator* to aggregate history evidence and broadcast global intent to all horizons, improving long-horizon coherence without autoregressive feedback; and (iii) a **variational latent plan** $z \in \mathbb{R}^d$, with a history-conditioned prior $p_\psi(z \mid x)$ for inference and a future-aware posterior $q_\phi(z \mid x, y)$ for training. The KL alignment transfers future information into the prior, so sampling from $p_\psi(z \mid x)$ yields diverse yet plausible trajectories at test time. At inference, we either use $z = \mu_p$ for deterministic decoding or sample $z \sim p_\psi(z \mid x)$ to generate multiple trajectories, where each trajectory is produced by one parallel trajectory decoding pass after obtaining the latent plan, without autoregressive rollout or iterative refinement.

4.2 Embeddings and Encoder

Following SASRec (Kang & McAuley, 2018), we represent each token (an item ID or padding) with a d-dimensional embedding. Let $\mathcal{I}$ be the item set. We use an embedding table $\mathbf{E} \in \mathbb{R}^{(|\mathcal{I}|+1)\times d}$, where row 0 corresponds to the padding token and rows $1{:}|\mathcal{I}|$ correspond to items. We also adopt positional embeddings $\mathbf{P} \in \mathbb{R}^{L_{\max}\times d}$, where $L_{\max} = T_{\max} + R + H$ is the maximum length that covers the truncated/padded history ($T \le T_{\max}$) plus the additional plan and future tokens.

Backbone encoder. Our design targets deployment-friendly trajectory prediction. We therefore keep the backbone identical to a standard SASRec Transformer encoder f_θ (a stack of multi-head self-attention blocks), and implement planning entirely through token-level interfaces and a lightweight latent module. This choice isolates the contribution to the planning mechanism while retaining the efficiency and robustness of a widely used sequential backbone. Given an input token sequence s of length L, we form the initial hidden states as $\mathbf{H}^{(0)} = \text{Embed}(s) + \mathbf{P}_{0:L-1}$, where $\text{Embed}(\cdot) \in \mathbb{R}^{L\times d}$ returns the corresponding token embeddings and $\mathbf{P}_{0:L-1}$ denotes the first L positional vectors. We index positions locally as $0, \ldots, L-1$ for each encoding pass, which matches our implementation

and avoids dependence on absolute sequence length. Let $\mathbf{m} \in \{0,1\}^L$ be the attention mask, where $m_\ell = 1$ indicates a valid token and $m_\ell = 0$ indicates padding. The encoder outputs $\mathbf{H} = f_\theta\big(\mathbf{H}^{(0)}, \mathbf{m}\big) \in \mathbb{R}^{L\times d}$.

4.3 Parallel H-Step Prediction with Plan Tokens

Instead of generating future items step-by-step and feeding predictions back as inputs, we allocate H dedicated output slots in the sequence. This enables trajectory prediction in one parallel decoding pass and avoids compounding errors caused by prediction feedback. Although each slot corresponds to one horizon, the slots interact through self-attention over the augmented sequence and share information via the plan tokens.

Prediction sequence. Given a history $x = (i_1, \ldots, i_T)$, we construct:

$$s^{(\text{pred})} = \big[x,\ \tilde{Z}_{1:R}(z),\ F_{1:H}\big], \tag{3}$$

where $F_{1:H}$ are H learnable future query tokens. The base plan tokens $Z_{1:R}$ are learnable and are transformed into $\tilde{Z}_{1:R}(z)$ by injecting the latent plan z (Section 4.4). The padding mask is:

$$\mathbf{m}^{(\text{pred})} = \big[\mathbf{m}_x,\ \mathbf{1}_R,\ \mathbf{1}_H\big], \tag{4}$$

where mask value 1 indicates a valid token and 0 indicates padding. Plan tokens and future query tokens are always valid, while history positions follow the input padding pattern.

Token placement and information flow. We insert the plan tokens between the history and the future queries to regulate information flow across the augmented sequence. The plan tokens summarize history evidence and broadcast global intent to all horizons, while each future query focuses on horizon-specific decoding. This reduces redundant long-range extraction from the history and empirically improves long-horizon coherence and trajectory-level accuracy.

Parallel H-step logits. Let $L = T + R + H$. We compute $\mathbf{H}^{(0)}_{\text{pred}} = \text{Embed}(s^{(\text{pred})}) + \mathbf{P}_{0:L-1}$ and obtain $\mathbf{H}^{(\text{pred})} = f_\theta(\mathbf{H}^{(0)}_{\text{pred}}, \mathbf{m}^{(\text{pred})})$. Since future query tokens are encoded in the same causal self-attention graph, later future queries can condition on earlier future query representations, together with the history and plan tokens, without using autoregressive item feedback. We read the hidden states at the H future query positions:

$$\mathbf{U} = \mathbf{H}^{(\text{pred})}_{\{T+R,\ldots,T+R+H-1\}} \in \mathbb{R}^{H\times d}, \tag{5}$$

Then, we compute logits by tying the output projection with the item embeddings:

$$\text{logits} = \mathbf{U}\mathbf{E}^\top \in \mathbb{R}^{H\times(|\mathcal{I}|+1)}, \tag{6}$$

which reduces parameters and maintains a consistent geometry between input representations and output scores.

4.4 Variational Latent Planning

Without an explicit stochastic variable, parallel decoding tends to collapse to a single mode under the same history, limiting one-to-many trajectory generation. We therefore introduce a continuous latent plan z that captures trajectory-level modes (e.g., short-term goals or exploration vs. exploitation tendencies), and condition prediction on z through the plan tokens. This design decouples "which future mode to follow" (controlled by z) from "how to realize it across horizons" (implemented by the parallel query slots). Sampling different z values yields different full H-step trajectories, each produced by one parallel trajectory decoding pass.

Masked mean pooling. For any hidden sequence $\mathbf{H} \in \mathbb{R}^{L\times d}$ with mask $\mathbf{m} \in \{0,1\}^L$ (where m_ℓ=1 indicates a valid token), we use masked mean pooling:

$$\text{Pool}(\mathbf{H}, \mathbf{m}) = \frac{\sum_{\ell=0}^{L-1} m_\ell \mathbf{H}_\ell}{\sum_{\ell=0}^{L-1} m_\ell + \epsilon} \in \mathbb{R}^d, \tag{7}$$

Compared with using a single position (e.g., the last item), mean pooling provides a robust sequence summary under variable-length padding.

Posterior $q_\phi(z \mid x, y)$**.** During training, we observe the ground-truth future $y = (i_{T+1}, \ldots, i_{T+H})$ (padded with 0 if shorter than H) and encode:

$$s^{(\text{post})} = [x, y], \qquad \mathbf{m}^{(\text{post})} = [\mathbf{m}_x, \mathbf{m}_y], \tag{8}$$

Let $L_{\text{post}} = T + H$ and $\mathbf{H}^{(0)}_{\text{post}} = \text{Embed}(s^{(\text{post})}) + \mathbf{P}_{0:L_{\text{post}}-1}$. We obtain:

$$\mathbf{H}^{(\text{post})} = f_\theta\Big(\mathbf{H}^{(0)}_{\text{post}}, \mathbf{m}^{(\text{post})}\Big), \qquad \mathbf{g}_{\text{post}} = \text{Pool}\Big(\mathbf{H}^{(\text{post})}, \mathbf{m}^{(\text{post})}\Big), \tag{9}$$

A lightweight MLP predicts diagonal Gaussian parameters:

$$(\mu_q, \log \sigma_q^2) = \text{MLP}_\phi(\mathbf{g}_{\text{post}}), \qquad q_\phi(z \mid x, y) = \mathcal{N}(\mu_q, \text{diag}(\sigma_q^2)), \tag{10}$$

We sample using reparameterization:

$$z = \mu_q + \sigma_q \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{11}$$

Prior $p_\psi(z \mid x)$**.** At inference, only history is available. We encode $s^{(\text{prior})} = x$ with mask $\mathbf{m}_x$. Let $L_{\text{prior}} = T$ and $\mathbf{H}^{(0)}_{\text{prior}} = \text{Embed}(x) + \mathbf{P}_{0:L_{\text{prior}}-1}$. We obtain:

$$\mathbf{H}^{(\text{prior})} = f_\theta\Big(\mathbf{H}^{(0)}_{\text{prior}}, \mathbf{m}_x\Big), \qquad \mathbf{g}_{\text{prior}} = \text{Pool}(\mathbf{H}^{(\text{prior})}, \mathbf{m}_x), \tag{12}$$

and predict:

$$(\mu_p, \log \sigma_p^2) = \text{MLP}_\psi(\mathbf{g}_{\text{prior}}), \qquad p_\psi(z \mid x) = \mathcal{N}(\mu_p, \text{diag}(\sigma_p^2)), \tag{13}$$

Conditioning the prior on x ensures that sampled plans remain compatible with the user's observed history, while Gaussian sampling enables exploring multiple plausible trajectory modes under the same x. To make the sampled plan z influence all horizons in a unified manner, we inject it into the shared plan-token interface. Specifically, we maintain R learnable plan tokens $Z_{1:R} \in \mathbb{R}^d$ and aSeq-NLLy a linear transformation of z as an additive shift:

$$\tilde{Z}_r(z) = Z_r + W_z z, \qquad r = 1, \ldots, R, \tag{14}$$

where $W_z \in \mathbb{R}^{d \times d}$. The shared shift $W_z z$ acts as a global mode-specific bias, while the R distinct plan token embeddings allow the model to specialize different aspects of the plan through self-attention. This injection is lightweight and enables the parallel horizon queries to exploit the sampled plan without iterative decoding.

4.5 Optimization Objective

We optimize a variational objective that combines a masked multi-step prediction loss with a KL term that transfers future-aware planning signals into a history-only prior, enabling controllable sampling.

Masked multi-step cross entropy. For each horizon $j \in \{1, \ldots, H\}$, the model produces logits_j over items and we supervise it with the ground-truth target y_j. To handle padded futures, we compute a masked average:

$$\mathcal{L}_{\text{CE}} = \frac{1}{\sum_{j=1}^{H} \mathbb{I}(y_j \neq 0)} \sum_{j=1}^{H} \mathbb{I}(y_j \neq 0) \cdot \text{CE}\big(\text{logits}_j,\ y_j\big), \tag{15}$$

where $y_j = 0$ indicates padding and is excluded from the loss.

KL alignment between posterior and prior. We regularize the latent plan by aligning the future-aware posterior with the history-conditioned prior:

$$\mathcal{L}_{\text{KL}} = \text{KL}\big(q_\phi(z \mid x, y) \parallel p_\psi(z \mid x)\big), \tag{16}$$

Importantly, this term is not merely a generic regularizer: the posterior leverages y to infer a trajectory mode during training, and the KL term distills this information into the prior so that sampling from $p_\psi(z \mid x)$ yields meaningful and diverse plans at inference. For diagonal Gaussians, the KL admits a closed form:

$$\text{KL}(q \| p) = \frac{1}{2} \sum_{h=1}^{d} \left[\log \frac{\sigma_{p,h}^2}{\sigma_{q,h}^2} + \frac{\sigma_{q,h}^2 + (\mu_{q,h} - \mu_{p,h})^2}{\sigma_{p,h}^2} - 1 \right], \tag{17}$$

Thus, the final objective function, where β controls the strength of plan alignment, is defined as:

$$\mathcal{L}_{LapTop} = \mathcal{L}_{\text{CE}} + \beta\, \mathcal{L}_{\text{KL}}, \tag{18}$$

4.6 Inference and Trajectory Generation

At inference time, we first compute the history-conditioned prior $p_\psi(z \mid x)$ and obtain a latent plan z, either by using the prior mean or by sampling from the prior. Conditioned on this plan, we build the augmented prediction sequence $s^{(\text{pred})} = [x, \tilde{Z}_{1:R}(z), F_{1:H}]$ and perform one parallel trajectory decoding pass to obtain $\text{logits}_{1:H}$. A trajectory is produced by taking the per-horizon $\arg\max$ from $\text{logits}_{1:H}$, without feeding predictions back into the model. To generate multiple plausible futures, we reuse the history-conditioned prior and repeat the parallel decoding stage with different draws of z from $p_\psi(z \mid x)$.

5 Experiment

In this section, we aim to answer these research questions (RQs):

- **RQ1**: How does LapTop perform on the UBTP task compared to state-of-the-art baselines?
- **RQ2**: How does LapTop benefit from variational latent planning comparing to other methods?
- **RQ3**: How do the key design components of LapTop affect prediction quality?
- **RQ4**: What are the inference cost and model complexity of LapTop compared to baselines?

5.1 Experiment Settings

Baselines. Following the prior work LPDO (Huang et al., 2025c), we evaluate LapTop against a wide range of representative baselines to assess its performance on the UBTP task. These include conventional sequential models (FPMC (Rendle et al., 2010), SASRec (Kang & McAuley, 2018), and STOSA (Fan et al., 2022)) and recent diffusion-based generative approaches (DiffuRec (Li et al., 2024b), DreamRec (Yang et al., 2023), PreferDiff (Liu et al., 2025), and DCRec (Huang et al., 2025d)). All baseline models are modified to support UBTP by optimizing the loss for every item in the predicted list. We also compare with SASRec-AR, which extends SASRec with an auto-regressive decoding strategy to generate sequences step-by-step.

Implementation Details. We tune common hyperparameters for all models on the validation set, including learning rate, dropout, and early-stopping criteria. For method-specific hyperparameters not included in our search space, we follow the settings recommended in the original papers. All models are trained and evaluated on an NVIDIA V100 GPU, and the training stopped after 5 evaluations without improvement to prevent overfitting. More details are provided in Section J.

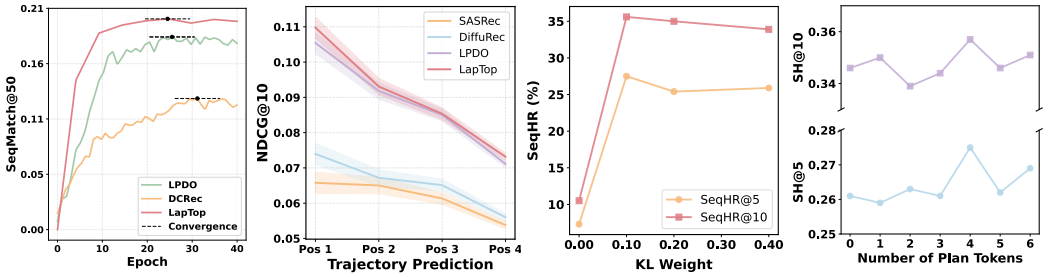
Evaluation Metrics. Following LPDO (Huang et al., 2025c), we adopt Sequence Hit Rate (SeqHR@K, SH@K), Sequence Normalized Discounted Cumulative Gain (SeqNDCG@K, SN@K) and Sequence Match (SeqMatch@K, SM@K) to enforce strict consistency across the entire predicted trajectory. Additionally, we utilize Perplexity (Seq-NLL) (Radford et al., 2019), scaled by a logarithmic transformation for numerical stability, to quantify the uncertainty of the trajectory predictions. For more details, please refer to Section I.

5.2 Performance Comparison (RQ1)

Table 1 summarizes the performance comparison across datasets and trajectory lengths. Diffusion-based methods equipped with ranking-oriented objectives, such as DiffuRec and DCRec, generally outperform non-diffusion baselines. By contrast, diffusion models without explicit ranking supervision, such as DreamRec and PreferDiff, struggle on trajectory prediction, yielding near-zero values on most ranking metrics and substantially higher Seq-NLL values; for example, PreferDiff obtains Seq-NLL values above 500.0 in several settings. We attribute this performance degradation partly to embedding collapse, which is further analyzed in Appendix M. Overall, LapTop consistently outperforms all baselines on both step-wise ranking metrics (SH@K and SN@K) and trajectory-level metrics (SM@50 and Seq-NLL). The improvements are particularly pronounced for longer prediction horizons. For example, on MovieLens-1M and LastFM with $K{=}10$, LapTop achieves substantial relative improvements in SH@K and SeqMatch@50 over the strongest baseline, while also achieving lower perplexity, indicating more coherent and distribution-aligned trajectory generation. Compared to diffusion-based approaches, LapTop delivers stronger and more stable performance without relying on iterative denoising or repeated sampling. These results demonstrate that latent planning with

Table 1: Overall performance across datasets and prediction horizons. Standard deviations are shown as subscripts for LapTop only. For all metrics except PPL, higher values indicate better performance. **Bold** results indicate the best results, while underlined results denote the second-best. *Improve.* denotes the relative improvement of LapTop over the strongest baseline.

Data	Metric	FPMC	SASRec	SASRec$_{AR}$	STOSA	DiffuRec	DreamRec	PreferDiff	DCRec	LPDO	**LapTop**	*Improve.*
Beauty ($H=3$)	SH@5	0.0236	0.0219	0.0218	0.0238	0.0274	0.0002	0.0025	0.0250	0.0306	$\mathbf{0.0371}_{\pm 0.0012}$	21.2%
	SH@10	0.0376	0.0466	0.0455	0.0407	0.0442	0.0006	0.0055	0.0388	0.0512	$\mathbf{0.0592}_{\pm 0.0015}$	15.6%
	SN@5	0.0129	0.0114	0.0114	0.0120	0.0149	0.0001	0.0015	0.0135	0.0174	$\mathbf{0.0262}_{\pm 0.0019}$	50.6%
	SN@10	0.0158	0.0199	0.0195	0.0171	0.0188	0.0003	0.0025	0.0159	0.0258	$\mathbf{0.0283}_{\pm 0.0010}$	9.7%
	SM@50	0.0253	0.0257	0.0238	0.0177	0.0363	0.0000	0.0000	0.0432	0.0523	$\mathbf{0.0599}_{\pm 0.0032}$	14.5%
	Seq-NLL	37.56	51.25	52.59	54.89	36.89	89.55	>500.0	38.90	34.32	$\mathbf{31.20}_{\pm 0.28}$	9.1%
ML-1M ($H=5$)	SH@5	0.0157	0.0895	0.0795	0.0865	0.0894	0.0028	0.0004	0.1107	0.1201	$\mathbf{0.1266}_{\pm 0.0021}$	5.4%
	SH@10	0.0371	0.1365	0.1366	0.1476	0.1540	0.0054	0.0165	0.1823	0.1939	$\mathbf{0.2100}_{\pm 0.0026}$	8.3%
	SN@5	0.0098	0.0434	0.0445	0.0493	0.0506	0.0016	0.0049	0.0621	0.0601	$\mathbf{0.0699}_{\pm 0.0014}$	12.6%
	SN@10	0.0179	0.0573	0.0574	0.0627	0.0649	0.0023	0.0073	0.0759	0.0839	$\mathbf{0.0890}_{\pm 0.0015}$	6.1%
	SM@50	0.0160	0.1045	0.1049	0.1074	0.1168	0.0000	0.0000	0.1458	0.1572	$\mathbf{0.1676}_{\pm 0.0040}$	6.6%
	Seq-NLL	38.43	33.62	33.58	33.47	32.47	165.59	>500.0	31.42	29.95	$\mathbf{29.16}_{\pm 0.20}$	2.6%
ML-1M ($H=10$)	SH@5	0.0058	0.0581	0.0560	0.0476	0.0624	0.0022	0.0077	0.0717	0.0851	$\mathbf{0.1001}_{\pm 0.0020}$	17.6%
	SH@10	0.0140	0.1033	0.0985	0.0864	0.1075	0.0043	0.0101	0.1277	0.1423	$\mathbf{0.1625}_{\pm 0.0027}$	14.2%
	SN@5	0.0036	0.0331	0.0320	0.0276	0.0356	0.0012	0.0043	0.0412	0.0456	$\mathbf{0.0538}_{\pm 0.0013}$	18.0%
	SN@10	0.0068	0.0446	0.0426	0.0377	0.0456	0.0018	0.0006	0.0550	0.0613	$\mathbf{0.0693}_{\pm 0.0015}$	13.1%
	SM@50	0.0000	0.0299	0.0261	0.0214	0.0311	0.0000	0.0000	0.0381	0.0449	$\mathbf{0.0591}_{\pm 0.0022}$	31.6%
	Seq-NLL	85.55	68.99	70.41	73.92	68.35	305.19	>500.0	68.58	66.14	$\mathbf{62.26}_{\pm 0.55}$	5.9%
LastFM ($H=5$)	SH@5	0.1636	0.1816	0.1829	0.1537	0.1793	0.0433	0.0035	0.1931	0.2576	$\mathbf{0.2720}_{\pm 0.0037}$	5.6%
	SH@10	0.2311	0.2523	0.2564	0.2184	0.2458	0.0531	0.0065	0.2681	0.3312	$\mathbf{0.3511}_{\pm 0.0045}$	6.0%
	SN@5	0.0830	0.0917	0.0908	0.0760	0.0906	0.0205	0.0023	0.0972	0.1209	$\mathbf{0.1332}_{\pm 0.0025}$	10.2%
	SN@10	0.0897	0.0953	0.0972	0.0836	0.0921	0.0185	0.0028	0.1015	0.1131	$\mathbf{0.1204}_{\pm 0.0023}$	6.5%
	SM@50	0.1636	0.1578	0.1636	0.1463	0.1638	0.0189	0.0000	0.1723	0.2337	$\mathbf{0.2503}_{\pm 0.0047}$	7.1%
	Seq-NLL	39.29	28.26	28.20	30.65	30.55	149.98	>500.0	29.28	27.68	$\mathbf{25.75}_{\pm 0.21}$	7.0%
LastFM ($H=10$)	SH@5	0.1080	0.1163	0.1148	0.0888	0.1209	0.0290	0.0000	0.1311	0.1703	$\mathbf{0.1801}_{\pm 0.0031}$	5.8%
	SH@10	0.1703	0.1606	0.1654	0.1495	0.1715	0.0407	0.0074	0.1881	0.2332	$\mathbf{0.2489}_{\pm 0.0036}$	6.7%
	SN@5	0.0598	0.0608	0.0597	0.0471	0.0625	0.0141	0.0000	0.0673	0.0869	$\mathbf{0.0939}_{\pm 0.0018}$	8.1%
	SN@10	0.0686	0.0610	0.0645	0.0624	0.0665	0.0159	0.0033	0.0730	0.0880	$\mathbf{0.0956}_{\pm 0.0017}$	8.6%
	SM@50	0.0713	0.0635	0.0566	0.0449	0.0586	0.0098	0.0000	0.0557	0.0779	$\mathbf{0.0900}_{\pm 0.0026}$	15.5%
	Seq-NLL	83.16	61.69	61.56	66.41	65.29	283.50	>500.0	61.32	59.31	$\mathbf{56.19}_{\pm 0.49}$	5.3%



(a) Training process comparison (b) Position-wise comparison (c) Study of KL weight β (d) Study of Plan Token

Figure 3: Ablation and analysis of LapTop. (a) Training process comparison on ML-1M (len=5). (b) Position-wise comparison of HR@5 across different models on ML-1M (len=5). (c) Impact of KL weight β of $\mathcal{L}$ on LastFM (len=5). (d) Effect of plan token r on LastFM (len=5).

parallel multi-step prediction effectively captures long-horizon dependencies, while maintaining the efficiency of SASRec-style recommenders.

5.3 Benefit of LapTop (RQ2)

Faster convergence. LapTop converges faster than other diffusion-based models employing different ranking losses. As illustrated in Figure 3 (a), LapTop converges within approximately 24 epochs, whereas its counterpart DCRec, which adopts a cross-entropy ranking loss, requires around 33 epochs to reach convergence.

Higher position-wise performance. Figure 3 (b) provides a position-wise comparison of SASRec, DiffuRec, LPDO, and LapTop on trajectory prediction tasks. The x-axis represents the different positions of the trajectory, and the y-axis represents the prediction accuracy measured by HR@5. The shaded areas represent the standard deviation. The results clearly show that LapTop consistently achieves higher HR@5 scores across all prediction positions compared to the other methods, indicating its superior ability to capture cross-horizon dependencies and the evolution of user preferences over time. Notably, LapTop exhibits a significant performance advantage, particularly at earlier positions, further highlighting its effectiveness in UBTP tasks.

5.4 Ablation Study of Hyperparameters in LapTop (RQ3)

Effect of the number of plan tokens r. Figure 3 analyzes the impact of the number of plan tokens r. We observe that performance improves as r increases from 0 to a small number, peaking around r=4, after which additional plan tokens provide diminishing or negative returns. This suggests that a compact set of plan tokens is sufficient to encode high-level future intent, while excessive tokens introduce redundancy without further benefits. More details refer to Appendix N.

Effect of KL regularization strength β. Figure 3 and Figure 7 show that LapTop is sensitive to the KL weight β. When β=0, the model degenerates into a purely deterministic parallel predictor, leading to inferior performance. Increasing β substantially improves both SHR and SNDCG, with optimal performance achieved at a moderate value (e.g., β=0.1). Further increasing β results in slight performance degradation, suggesting that overly strong regularization constrains the latent space and limits expressive capacity. This behavior is consistent across datasets, indicating that a balanced prior–posterior alignment is crucial for effective latent planning. More details refer to Appendix N.

Table 2: Architectural ablation study of different components in LapTop. All values are reported in percentage scale ($\times$100), and higher values indicate better performance.

Model	ML-1M (len=10)				LastFM (len=10)			
	SH@10↑	SN@10↑	SM@50↑	SM@100↑	SH@10↑	SN@10↑	SM@50↑	SM@100↑
V1: PQ	15.52	6.53	5.21	12.47	25.14	9.38	9.86	12.89
V2: PQ+PT	15.89	6.70	5.32	12.70	25.33	9.41	9.89	13.29
V3: PQ+$z\rightarrow F$	15.59	6.54	5.01	12.52	25.88	9.50	9.77	14.45
V4: PQ+DP	15.42	6.48	5.38	12.48	25.79	9.63	8.69	12.99
V5: PQ+CVAE	15.70	6.61	5.13	13.00	25.72	9.45	9.28	13.96
V6: Full	**16.85**	**7.07**	**5.98**	**13.42**	**26.16**	**9.71**	**10.35**	**14.77**

PQ: parallel future queries only. **+PT**: plan tokens. **+$z\rightarrow F$**: directly injecting the CVAE latent variable into future tokens. **+DP**: deterministic planning. **+CVAE**: CVAE latent planning without plan tokens. **Full**: the complete LapTop architecture.

Architectural ablation. Table 2 studies the contribution of different architectural components in LapTop. The parallel-query-only variant already provides a strong non-autoregressive baseline, showing the effectiveness of decoding multiple future horizons in one pass. Adding plan tokens or latent conditioning individually improves certain ranking or trajectory-level metrics, but their effects are not strictly monotonic across all variants. In contrast, the full LapTop consistently achieves the strongest overall performance on both ML-1M and LastFM, obtaining the best results on all reported metrics in ML-1M and on most metrics in LastFM. These results suggest that parallel future queries, latent planning, and plan-token mediation are complementary: the latent variable captures trajectory-level uncertainty, while plan tokens provide a compact interface for broadcasting the sampled plan to future query slots.

Inference efficiency (RQ4). We further compare inference cost and model complexity in Appendix L. LapTop achieves latency comparable to SASRec while substantially outperforming all baselines on trajectory-level metrics.

6 Conclusion

This paper investigated user behavior trajectory prediction and proposed LapTop, a latent planning framework that enables efficient parallel multi-step recommendation with non-iterative parallel trajectory decoding. By introducing a variational latent plan and a small set of plan tokens, LapTop explicitly models long-horizon dependencies and multi-modal futures while avoiding autoregressive rollout and iterative generative sampling. Extensive experiments across multiple benchmarks show that LapTop consistently outperforms strong sequential and generative baselines on both step-wise and trajectory-level metrics, with particularly strong gains for longer prediction horizons, while maintaining inference efficiency.

7 GENERATIVE AI DISCLOSURE

In this work, we used generative AI tools only to polish and improve the clarity of the manuscript writing. We did not use generative AI tools for code writing, experimental design, experiment execution, result analysis, or evaluation. All technical implementation, experimental design, experiments, and evaluations were conducted by the authors. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical accuracy and consistency with the original research. We take responsibility for the final content of this work, including any text produced with the aid of generative AI.

REFERENCES

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011.

Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. Sequential recommendation via stochastic self-attention. In *Proceedings of the ACM web conference 2022*, pp. 2036–2047, 2022.

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

Ruining He and Julian J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE 16th International Conference on Data Mining, December 12-15, 2016, Barcelona, Spain*, pp. 191–200. IEEE Computer Society, 2016.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the Advances in Neural Information Processing Systems*, 2020.

Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V. Chawla. Online purchase prediction via multi-scale modeling of behavior dynamics. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2613–2622. ACM.

Chengkai Huang, Shoujin Wang, Xianzhi Wang, and Lina Yao. Dual contrastive transformer for hierarchical preference modeling in sequential recommendation. In *Proceedings of the 46th international acm sigir conference on research and development in information retrieval*, pp. 99–109, 2023a.

Chengkai Huang, Shoujin Wang, Xianzhi Wang, and Lina Yao. Modeling temporal positive and negative excitation for sequential recommendation. In *Proceedings of the ACM Web Conference 2023*, pp. 1252–1263, 2023b.

Chengkai Huang, Shoujin Wang, Xianzhi Wang, and Lina Yao. Dual contrastive transformer for hierarchical preference modeling in sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 99–109. ACM, 2023c.

Chengkai Huang, Tong Yu, Kaige Xie, Shuai Zhang, Lina Yao, and Julian J. McAuley. Foundation models for recommender systems: A survey and new perspectives. *CoRR*, abs/2402.11143, 2024.

Chengkai Huang, Hongtao Huang, Tong Yu, Kaige Xie, Junda Wu, Shuai Zhang, Julian J. McAuley, Dietmar Jannach, and Lina Yao. A survey of foundation model-powered recommender systems: From feature-based, generative to agentic paradigms. *CoRR*, abs/2504.16420, 2025a.

Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A. Rossi, Branislav Kveton, Dongruo Zhou, Julian J. McAuley, and Lina Yao. Towards agentic recommender systems in the era of multimodal large language models. *CoRR*, abs/2503.16734, 2025b.

Hongtao Huang, Chengkai Huang, Junda Wu, Tong Yu, Julian McAuley, and Lina Yao. Listwise preference diffusion optimization for user behavior trajectories prediction. *The Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025c.

Hongtao Huang, Chengkai Huang, Tong Yu, Xiaojun Chang, Wen Hu, Julian McAuley, and Lina Yao. Dual conditional diffusion models for sequential recommendation. *Proceedings of the 19th ACM International Conference on Web Search and Data Mining*, 2025d.

Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. In *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104. Springer, 2021.

Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation . In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Jiayu Li, Hanyu Li, Zhiyu He, Weizhi Ma, Peijie Sun, Min Zhang, and Shaoping Ma. Rechorus2. 0: A modular and task-flexible recommendation library. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pp. 454–464, 2024a.

Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(3):66:1–66:28, 2024b.

Xiao Lin, Xiaokai Chen, Chenyang Wang, Hantao Shu, Linfeng Song, Biao Li, and Peng Jiang. Discrete conditional diffusion for reranking in recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 161–169. ACM, 2024.

Shuo Liu, An Zhang, Guoqing Hu, Hong Qian, and Tat-Seng Chua. Preference diffusion for recommendation. In *Proceedings of the Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025.

Julian J. 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. ACM, 2015.

Lin Ning, Luyang Liu, Jiaxing Wu, Neo Wu, Devora Berlowitz, Sushant Prakash, Bradley Green, Shawn O'Banion, and Jun Xie. User-llm: Efficient LLM contextualization with user embeddings. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 1219–1223. ACM, 2025.

Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. Pinnerformer: Sequence modeling for user representation at pinterest. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3702–3712. ACM, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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.

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*, pp. 811–820, 2010.

Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 832–841, 2023.

Ting-Ruen Wei and Yi Fang. Diffusion models in recommendation systems: A survey. *arXiv preprint arXiv:2501.10548*, 2025.

Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3745–3755. ACM, 2021.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, 56(4):105:1–105:39, 2024.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Xiaoxin Ye, Chengkai Huang, Hongtao Huang, and Lina Yao. Gaussian mixture flow matching with domain alignment for multi-domain sequential recommendation, 2025. URL `https://arxiv.org/abs/2510.21021`.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 1097–1100, 2018.

SuSeq-NLLementary Material

A LIMITATIONS

LapTop is evaluated under standard offline fixed-horizon protocols on public ID-based sequential recommendation benchmarks. This setting enables controlled and reproducible comparisons with Transformer-based and diffusion-based baselines while isolating the effect of latent planning. The framework can be naturally extended to richer item features, side information, or multimodal encoders by augmenting the input representation. In practical systems, prediction horizons and candidate sets may vary across aSeq-NLLications. Future work may explore adaptive horizon selection and system-specific serving strategies, without changing LapTop's core non-autoregressive decoding framework. In addition, while this paper uses stochastic decoding mainly to analyze multiple plausible future modes, downstream systems can further tailor how sampled trajectories are ranked, aggregated, or selected for diversity- or exploration-oriented recommendation.

B PRELIMINARY: VARIATIONAL AUTOENCODER

Variational Autoencoders (VAEs) (Kingma & Welling, 2019) provide a principled framework for learning latent variables that capture underlying generative factors while supporting efficient inference. In recommendation and sequential modeling, VAEs are commonly adopted to represent user intent, preference uncertainty, or high-level behavioral patterns that cannot be directly observed from interaction logs.

A VAE consists of two components: an inference network and a generative network. During training, the inference network estimates a latent representation conditioned on observed data, while the generative network reconstructs the target outputs given the latent variable. To ensure that latent representations can be sampled at inference time without access to future observations, a prior distribution conditioned only on the available context is learned jointly with the inference network. Training encourages the inferred latent distribution to remain close to this prior, enabling consistent behavior between training and inference. A key property of VAEs is their ability to model uncertainty and one-to-many mappings. Instead of committing to a single deterministic representation, the latent variable captures multiple plausible explanations of the data, allowing the model to generate diverse yet coherent outputs. This property is particularly important for user behavior modeling, where future actions are inherently ambiguous and may follow different trajectories even under similar historical contexts.

In our UBTP task, VAEs have been aSeq-NLLied to represent latent user intent or session-level factors, typically focusing on next-item prediction or global preference modeling. However, most existing approaches use latent variables to influence a single prediction step or rely on autoregressive decoding, limiting their effectiveness in long-horizon forecasting. In this work, we leverage the VAE framework as a foundation for latent planning, where the latent variable encodes high-level future intent and guides parallel multi-step trajectory prediction, bridging the gap between uncertainty modeling and efficient long-term forecasting.

C ALGORITHM

In this section, we provide the pseudo-code of the training and inference process of LapTop in in Algorithm 1 and Algorithm 2.

D VARIATIONAL INTERPRETATION OF THE OBJECTIVE

In this section, we provide a variational interpretation of the training objective of LapTop. Given a user history $\mathbf{x}$ and a target future trajectory $\mathbf{y}$, LapTop introduces a latent plan variable $\mathbf{z}$ to capture trajectory-level uncertainty. The conditional likelihood of the future trajectory can be written by

Algorithm 1 Training of LapTop

Require: User history sequence $\mathbf{x}$, target trajectory $\mathbf{y}$, history mask $\mathbf{m}_x$, future mask $\mathbf{m}_y$, trajectory length H, number of plan tokens R, KL weight β, trainable parameters Θ, learning rate η

repeat
  $\mathbf{s}^{\text{post}} \leftarrow \text{Concat}(\mathbf{x}, \mathbf{y})$, $\mathbf{m}^{\text{post}} \leftarrow \text{Concat}(\mathbf{m}_x, \mathbf{m}_y)$ ▷ Build future-aware posterior input.
  $\mathbf{H}^{\text{post}} \leftarrow f_\theta(\text{Embed}(\mathbf{s}^{\text{post}}) + \mathbf{P}, \mathbf{m}^{\text{post}})$ ▷ Encode history and future.
  $\mathbf{g}^{\text{post}} \leftarrow \text{Pool}(\mathbf{H}^{\text{post}}, \mathbf{m}^{\text{post}})$
  $(\boldsymbol{\mu}_q, \log \boldsymbol{\sigma}_q^2) \leftarrow \text{MLP}_\phi(\mathbf{g}^{\text{post}})$ ▷ Posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$.
  $\mathbf{H}^{\text{prior}} \leftarrow f_\theta(\text{Embed}(\mathbf{x}) + \mathbf{P}, \mathbf{m}_x)$ ▷ Encode history only.
  $\mathbf{g}^{\text{prior}} \leftarrow \text{Pool}(\mathbf{H}^{\text{prior}}, \mathbf{m}_x)$
  $(\boldsymbol{\mu}_p, \log \boldsymbol{\sigma}_p^2) \leftarrow \text{MLP}_\psi(\mathbf{g}^{\text{prior}})$ ▷ Prior $p_\psi(\mathbf{z}|\mathbf{x})$.
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{z} \leftarrow \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon}$ ▷ Sample latent plan from posterior.
  $\widetilde{\mathbf{Z}}_{1:R} \leftarrow \mathbf{Z}_{1:R} + \mathbf{W}_z \mathbf{z}$ ▷ Inject latent plan into plan tokens.
  $\mathbf{s}^{\text{pred}} \leftarrow \text{Concat}(\mathbf{x}, \widetilde{\mathbf{Z}}_{1:R}, \mathbf{F}_{1:H})$ ▷ Build prediction sequence.
  $\mathbf{m}^{\text{pred}} \leftarrow \text{Concat}(\mathbf{m}_x, \mathbf{1}_R, \mathbf{1}_H)$
  $\mathbf{H}^{\text{pred}} \leftarrow f_\theta(\text{Embed}(\mathbf{s}^{\text{pred}}) + \mathbf{P}, \mathbf{m}^{\text{pred}})$ ▷ One-pass trajectory prediction.
  $\mathbf{U} \leftarrow \text{Extract}(\mathbf{H}^{\text{pred}}, H)$ ▷ Extract hidden states of future query slots.
  $\mathbf{O}_{1:H} \leftarrow \mathbf{U}\mathbf{E}^\top$ ▷ Compute item logits for all future steps.
  $\mathcal{L}_{\text{CE}} \leftarrow \frac{1}{\sum_{j=1}^{H} \mathbb{I}(y_j \neq 0)} \sum_{j=1}^{H} \mathbb{I}(y_j \neq 0) \cdot \text{CE}(\mathbf{O}_j, y_j)$ ▷ Masked multi-step prediction loss.
  $\mathcal{L}_{\text{KL}} \leftarrow \text{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \,\|\, p_\psi(\mathbf{z}|\mathbf{x})\big)$ ▷ Align posterior and prior.
  $\mathcal{L}_{LapTop} \leftarrow \mathcal{L}_{\text{CE}} + \beta \mathcal{L}_{\text{KL}}$
  $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}_{LapTop}$ ▷ Gradient descent update.
until convergence
return optimized Θ

Algorithm 2 Inference of LapTop

Require: User history sequence $\mathbf{x}$, history mask $\mathbf{m}_x$, trajectory length H, number of plan tokens R, trained parameters Θ, generation mode $g \in \{\text{mean}, \text{sample}\}$

$\mathbf{H}^{\text{prior}} \leftarrow f_\theta(\text{Embed}(\mathbf{x}) + \mathbf{P}, \mathbf{m}_x)$ ▷ Encode history for prior inference.
$\mathbf{g}^{\text{prior}} \leftarrow \text{Pool}(\mathbf{H}^{\text{prior}}, \mathbf{m}_x)$
$(\boldsymbol{\mu}_p, \log \boldsymbol{\sigma}_p^2) \leftarrow \text{MLP}_\psi(\mathbf{g}^{\text{prior}})$ ▷ Infer $p_\psi(\mathbf{z}|\mathbf{x})$.
if $g = \text{mean}$ **then**
  $\mathbf{z} \leftarrow \boldsymbol{\mu}_p$ ▷ Use deterministic latent plan.
else
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{z} \leftarrow \boldsymbol{\mu}_p + \boldsymbol{\sigma}_p \odot \boldsymbol{\epsilon}$ ▷ Sample stochastic latent plan.
end if
$\widetilde{\mathbf{Z}}_{1:R} \leftarrow \mathbf{Z}_{1:R} + \mathbf{W}_z \mathbf{z}$ ▷ Inject latent plan into plan tokens.
$\mathbf{s}^{\text{pred}} \leftarrow \text{Concat}(\mathbf{x}, \widetilde{\mathbf{Z}}_{1:R}, \mathbf{F}_{1:H})$ ▷ Construct augmented prediction sequence.
$\mathbf{m}^{\text{pred}} \leftarrow \text{Concat}(\mathbf{m}_x, \mathbf{1}_R, \mathbf{1}_H)$
$\mathbf{H}^{\text{pred}} \leftarrow f_\theta(\text{Embed}(\mathbf{s}^{\text{pred}}) + \mathbf{P}, \mathbf{m}^{\text{pred}})$ ▷ Parallel trajectory decoding.
$\mathbf{U} \leftarrow \text{Extract}(\mathbf{H}^{\text{pred}}, H)$ ▷ Extract future query hidden states.
$\mathbf{O}_{1:H} \leftarrow \mathbf{U}\mathbf{E}^\top$ ▷ Compute item logits.
$\hat{y}_j \leftarrow \arg\max_{i \in \mathcal{I}} \mathbf{O}_{j,i}$, for $j = 1, \ldots, H$ ▷ Decode all horizons in parallel.
$\hat{\mathbf{y}} \leftarrow (\hat{y}_1, \ldots, \hat{y}_H)$
return predicted trajectory $\hat{\mathbf{y}}$

marginalizing over the latent plan:

$$p_{\theta,\psi}(\mathbf{y} \mid \mathbf{x}) = \int p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) p_\psi(\mathbf{z} \mid \mathbf{x}) d\mathbf{z}, \tag{19}$$

where $p_\psi(\mathbf{z} \mid \mathbf{x})$ is the history-conditioned prior and $p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z})$ is implemented by the one-pass trajectory decoder.

Since the exact marginal likelihood is generally intractable, we introduce a future-aware posterior distribution $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$. By Jensen's inequality, the log-likelihood admits the following evidence

lower bound:

$$\log p_{\theta,\psi}(\mathbf{y} \mid \mathbf{x}) = \log \int q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \frac{p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) p_\psi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y})} d\mathbf{z} \tag{20}$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} \left[\log p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \,\|\, p_\psi(\mathbf{z} \mid \mathbf{x})\right). \tag{21}$$

The first term corresponds to reconstructing the observed future trajectory from the sampled latent plan. In LapTop, the decoder produces logits for all future horizons in a single forward pass. With padded positions masked out, we model the trajectory likelihood as:

$$\log p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) = \sum_{j=1}^{H} \mathbb{I}(y_j \neq 0) \log p_\theta(y_j \mid \mathbf{x}, \mathbf{z}, j), \tag{22}$$

where j denotes the future horizon. Therefore, maximizing the expected log-likelihood is equivalent to minimizing the masked multi-step cross-entropy:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{\sum_{j=1}^{H} \mathbb{I}(y_j \neq 0)} \sum_{j=1}^{H} \mathbb{I}(y_j \neq 0) \log p_\theta(y_j \mid \mathbf{x}, \mathbf{z}, j). \tag{23}$$

The second term aligns the future-aware posterior with the history-conditioned prior:

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \,\|\, p_\psi(\mathbf{z} \mid \mathbf{x})\right). \tag{24}$$

This alignment is important because the posterior has access to the ground-truth future trajectory during training, while only the prior is available at inference time.

Thus, minimizing the negative ELBO leads to:

$$\mathcal{L}_{\text{N-ELBO}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{KL}}. \tag{25}$$

In practice, we use a weighted KL term:

$$\mathcal{L}_{LapTop} = \mathcal{L}_{\mathrm{CE}} + \beta \mathcal{L}_{\mathrm{KL}}, \tag{26}$$

where β controls the strength of prior–posterior alignment. This can be viewed as a β-VAE style objective that balances trajectory reconstruction and latent-plan regularization. When β is too small, the learned posterior may not transfer sufficient future-aware planning information to the prior. When β is too large, the latent plan may be over-regularized and lose expressive capacity. This interpretation is consistent with the sensitivity analysis of the KL weight in the main experiments.

E Probabilistic View of Parallel Trajectory Decoding

Although LapTop predicts all future positions in a single non-autoregressive decoding pass, it does not treat future horizons as independent isolated representations. Given a history $\mathbf{x}$ and a latent plan $\mathbf{z}$, LapTop constructs the augmented prediction sequence as belows:

$$\mathbf{s}^{\mathrm{pred}} = [\mathbf{x}, \widetilde{\mathbf{Z}}_{1:R}(\mathbf{z}), \mathbf{F}_{1:H}], \tag{27}$$

where $\widetilde{\mathbf{Z}}_{1:R}(\mathbf{z})$ are latent-conditioned plan tokens and $\mathbf{F}_{1:H}$ are learnable future query slots. The Transformer encoder computes as belows:

$$\mathbf{H}^{\mathrm{pred}} = f_\theta(\mathrm{Embed}(\mathbf{s}^{\mathrm{pred}}) + \mathbf{P}, \mathbf{m}^{\mathrm{pred}}), \tag{28}$$

where $\mathbf{m}^{\mathrm{pred}}$ includes the padding mask and the causal self-attention mask. Under this causal mask, the hidden state of the j-th future query slot can attend to the non-padded history tokens, the latent-conditioned plan tokens, and earlier future query slots $\mathbf{F}_{<j}$, but not to later future query slots $\mathbf{F}_{>j}$. Thus, future queries can exchange horizon-level information in a causal manner while avoiding autoregressive item feedback.

The conditional likelihood used for training can be written as:

$$p_\theta(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) = \prod_{j=1}^{H} p_\theta(y_j \mid \mathbf{x}, \mathbf{z}, j), \tag{29}$$

where each factor is parameterized by the corresponding future-query hidden state:

$$p_\theta(y_j \mid \mathbf{x}, \mathbf{z}, j) = \mathrm{Softmax}(\mathbf{u}_j \mathbf{E}^\top)_{y_j}, \tag{30}$$

Here, $\mathbf{u}_j$ is not computed by an independent horizon-specific network. Instead, it is produced by a shared Transformer pass over the augmented sequence, where causal self-attention allows information to flow from the history and latent plan tokens to all horizons, and from earlier future query slots to later ones. Therefore, the factorized likelihood should be understood as an efficient output parameterization rather than an assumption that future horizons are encoded independently. This view clarifies how LapTop avoids autoregressive feedback over predicted items while still enabling causal cross-horizon coordination through shared attention and latent plan tokens.

F Dataset

Table 3: Statistics of Datasets.

Dataset	# Sequence	# Items	# Actions	Average Lens
Beauty	22,363	12,101	198,502	8.53
MovieLens-1M	6,040	3,416	999,611	165.50
LastFM	891	1,000	1,293,103	1254.49

Table 4: Statistics of Datasets after Processing.

Dataset	Trajectory Length	# Sequence (Before)	# Sequence (After)
Beauty	3	22,363	1927
MovieLens-1M	5	6,040	6040
MovieLens-1M	10	6,040	5231
LastFM	5	891	829
LastFM	10	891	784

We evaluate our approach on three sequential recommendation datasets: Amazon Beauty (McAuley et al., 2015), MovieLens-1M (Harper & Konstan, 2015), and LastFM (Bertin-Mahieux et al., 2011). Following previous work (Huang et al., 2025c), we adopt the standard *Leave-One-Out* data-splitting strategy commonly used in sequential recommendation tasks, where the *"One"* in our setting refers to a user trajectory with a predefined sequence length. The trajectory length is set to 3 for Amazon Beauty, and 5 and 10 for MovieLens-1M and LastFM, respectively. Empirically, we follow the Leave-One-Out rule to split the dataset. Specifically, for a trajectory length H, only sequences in the original dataset with lengths greater than $1 + 3 \times H$ are considered valid. Tables 3 and 4 report the dataset statistics before and after preprocessing.

G Environment Details

All experiments were conducted on a Linux server equipped with one NVIDIA V100 GPU. We used Python 3.10.19 and PyTorch 2.9.0 (Imambi et al., 2021) with CUDA 12.8 and cuDNN enabled. Unless otherwise specified, the same hardware and software configuration was used for both memory construction and inference.

H Baseline Comparison

To comprehensively evaluate the effectiveness of our proposed latent planning framework, we conduct extensive benchmarking across a diverse set of recommendation models, including three widely adopted conventional models and five generative models.

- **FPMC** (Rendle et al., 2010): A classical non-Transformer recommendation model that combines the strengths of matrix factorization and Markov chains.

- **SASRec** (Kang & McAuley, 2018): A pioneering Transformer-based recommender that leverages self-attention to capture both short-term and long-term dependencies in user interaction sequences. Additionally, we extend SASRec to support auto-regressive generation, denoted as SASRec-AR.
- **STOSA** (Fan et al., 2022): An extension of Transformer-based recommendation models that explicitly models uncertainty in users' historical action sequences.
- **DiffuRec** (Li et al., 2024b): A widely adopted generative recommender that aSeq-NLLies diffusion models to recommender systems by probabilistically modeling items and user interests.
- **DreamRec** (Yang et al., 2023): A diffusion-based recommender that introduces a learning-to-generate paradigm for next-item recommendation.
- **PreferDiff** (Liu et al., 2025): A user preference-aware diffusion recommender that focuses on optimizing the training objective.
- **DCRec** (Huang et al., 2025d): A recently proposed diffusion-based model that integrates both implicit and explicit conditioning signals.
- **LPDO** (Huang et al., 2025c): A diffusion model for the user behavior trajectory prediction task, designed to address the challenge of modeling global preferences across item sequences.

For fair comparison, all baselines and LapTop are implemented as ID-based recommendation models. For baselines originally designed for next-item prediction, we extend their training objectives to predict the next H items and carefully tune hyperparameters to perform best on validation. To be specific, for pointwise sequential baselines (Kang & McAuley, 2018; Rendle et al., 2010; Fan et al., 2022), we attach H horizon-specific query/output heads and optimize the masked cross-entropy over all valid future positions. For autoregressive SASRec-AR (Kang & McAuley, 2018), predictions are generated step by step by appending the previously predicted item. For diffusion-based baselines (Li et al., 2024b; Yang et al., 2023; Liu et al., 2025; Huang et al., 2025d;c), we follow their original generation procedure and supervise/evaluate the generated H-step trajectory under the same UBTP metrics.

I Evaluation Metrics

Commonly used evaluation metrics for next-item prediction tasks, such as HR@K and NDCG@K, are not well-suited for UBTP tasks because they mainly evaluate single-step ranking quality and cannot fully capture the consistency of long-horizon trajectory prediction. Similarly, sequence-based metrics from NLP tasks, such as BLEU and ROUGE, are also less suitable for UBTP because they emphasize strict token-level overlap, which is difficult to satisfy in sparse user behavior data where multiple future trajectories may be plausible.

Following previous work (Huang et al., 2025c), let $\mathcal{D}_{\text{test}}$ denote the test set. For each user sequence $u \in \mathcal{D}_{\text{test}}$, the ground-truth future trajectory is denoted as $\mathbf{y}_u = (y_{u,1}, \ldots, y_{u,H})$, where H is the prediction horizon. For each horizon j, the model produces a ranked list over candidate items, and we use $A^K_{u,j}$ to denote the Top-K item set at horizon j. Let $r_{u,j}$ denote the rank of the ground-truth item $y_{u,j}$ in the predicted list. If $y_{u,j} = 0$, the position is padding and is excluded from evaluation. We define the valid-position mask as:

$$m_{u,j} = \mathbb{I}(y_{u,j} \neq 0). \tag{31}$$

We further define the set of valid horizons as:

$$\mathcal{J} = \left\{ j \in \{1, \ldots, H\} \mid \sum_{u \in \mathcal{D}_{\text{test}}} m_{u,j} > 0 \right\}. \tag{32}$$

Sequence Hit Rate. For each horizon j, the hit indicator is defined as:

$$h^K_{u,j} = \mathbb{I}\left(y_{u,j} \in A^K_{u,j}\right). \tag{33}$$

We first compute the position-wise Hit Rate at each valid horizon:

$$\text{HR}^K_j = \frac{\sum_{u \in \mathcal{D}_{\text{test}}} m_{u,j} h^K_{u,j}}{\sum_{u \in \mathcal{D}_{\text{test}}} m_{u,j}}. \tag{34}$$

Then, SeqHR@K is computed as the geometric mean of position-wise Hit Rates over all valid horizons:

$$\text{SeqHR@K} = \left(\prod_{j \in \mathcal{J}} \text{HR}_j^K \right)^{\frac{1}{|\mathcal{J}|}}. \tag{35}$$

Sequence NDCG. SeqNDCG@K further considers the ranking position of the ground-truth item. For each horizon, the discounted gain is:

$$\text{NDCG}_{u,j}^K = \mathbb{I}(r_{u,j} \leq K) \cdot \frac{1}{\log_2(r_{u,j}+1)}. \tag{36}$$

We first compute the position-wise NDCG at each valid horizon:

$$\text{NDCG}_j^K = \frac{\sum_{u \in \mathcal{D}_{\text{test}}} m_{u,j} \text{NDCG}_{u,j}^K}{\sum_{u \in \mathcal{D}_{\text{test}}} m_{u,j}}. \tag{37}$$

The sequence-level NDCG is then computed as the geometric mean of position-wise NDCG values over all valid horizons:

$$\text{SeqNDCG@K} = \left(\prod_{j \in \mathcal{J}} \text{NDCG}_j^K \right)^{\frac{1}{|\mathcal{J}|}}. \tag{38}$$

Sequence Match. SeqHR@K and SeqNDCG@K measure average horizon-level ranking quality, but they do not require the entire trajectory to be correctly covered. To evaluate strict trajectory-level consistency, we adopt SeqMatch@K, which checks whether every non-padded target item in the future trajectory appears in the corresponding Top-K prediction list:

$$\text{SeqMatch@K} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{u \in \mathcal{D}_{\text{test}}} \mathbb{I} \left(\bigwedge_{j=1}^{H} \left[m_{u,j} = 0 \ \vee \ y_{u,j} \in A_{u,j}^K \right] \right). \tag{39}$$

Here, padded positions are ignored through $m_{u,j}$. Therefore, a trajectory is counted as matched only when all valid future items are successfully retrieved at their corresponding horizons.

Perplexity. We further use perplexity (Seq-NLL) (Radford et al., 2019; Huang et al., 2025c) to measure the uncertainty of trajectory prediction. Let $p_\theta(y_{u,j} \mid \mathbf{x}_u, j)$ denote the probability assigned by the model to the ground-truth item at horizon j. In this paper, for numerical stability, Seq-NLL is scaled using a logarithmic transformation and is defined as follows:

$$\text{Seq-NLL} = -\frac{1}{\sum_{u \in \mathcal{D}_{\text{test}}} \sum_{j=1}^{H} m_{u,j}} \sum_{u \in \mathcal{D}_{\text{test}}} \sum_{j=1}^{H} m_{u,j} \log p_\theta(y_{u,j} \mid \mathbf{x}_u, j). \tag{40}$$

Lower Seq-NLL indicates that the model assigns a higher likelihood to the ground-truth future trajectory. All padded positions are excluded from the computation of SeqHR@K, SeqNDCG@K, SeqMatch@K, and Seq-NLL.

J Implementation Details

For all baselines, we carefully tune their hyperparameters to achieve the best performance on the validation set. For our LapTop, we set r=4 and β=0.1 for all datasets. For all models, we fix the embedding dimension to 128 and the dropout rate to 0.1. All models are trained and evaluated on an NVIDIA Tesla V100 GPU, and the training stopped after 5 evaluations without improvement to prevent overfitting. Moreover, we implement all models in PyTorch and adopt consistent training configurations across baselines to ensure fair comparison. For both the baselines and our proposed LapTop, we fix the embedding dimension to 128 and the dropout rate to 0.1. For sequential models,

the maximum sequence length is fixed at 50. All models are trained using the Adam optimizer (Kingma & Ba, 2015) with a batch size of 256 and a learning rate selected from $\{0.01, 0.005, 0.001\}$ based on validation performance. For the transformer-based model, the number of blocks is set to 4. Model training is conducted for up to 1,000 epochs with early stopping after 5 epochs of no improvement. To adapt all models to the UBTP task (i.e., next-H item prediction), we modify the output layer to predict multiple future items rather than only the immediate next one. Specifically, their training objective is reformulated from single-item prediction to trajectory-level prediction. Except for the aforementioned configurations, models with public implementations by previous works, including SASRec (Kang & McAuley, 2018), STOSA (Fan et al., 2022), DiffuRec (Li et al., 2024b), DreamRec (Yang et al., 2023), and PreferDiff (Liu et al., 2025), are trained following their respective default settings. For FPMC, we adopt the implementation from ReChorus (Li et al., 2024a). For DCRec (Huang et al., 2025d), we reproduce the model and set its loss balance factor to $\lambda = 0.1$. For LPDO, we follow the original setting used in the previous paper, with $\lambda = 0.1$ and $\gamma = 0.3$. All reported results are averaged over five independent runs with different random seeds. For the main comparison metrics, statistical significance is assessed using paired t-tests across the five runs, with $p < 0.05$ considered significant.

K Inference Complexity Analysis

Let

$$C(n) = O(n^2 d + n d^2), \tag{41}$$

denote the computational cost of one Transformer encoder pass with sequence length n and hidden dimension d. For a history length T and prediction horizon H, an autoregressive rollout model requires H sequential forward passes, since each predicted item is appended back to the input before the next prediction. Its total cost is approximately:

$$\sum_{j=1}^{H} C(T + j), \tag{42}$$

Diffusion-based trajectory predictors require multiple denoising or refinement steps. If the number of denoising steps is S, their inference cost can be written as:

$$S \cdot C(T + H), \tag{43}$$

up to model-specific conditioning overhead.

In contrast, LapTop separates inference into a non-iterative prior inference stage and a parallel trajectory decoding stage. First, it encodes the history once to obtain the history-conditioned prior $p_\psi(\mathbf{z} \mid \mathbf{x})$, which costs:

$$C(T), \tag{44}$$

plus a lightweight MLP head for Gaussian parameterization. Given a deterministic or sampled latent plan $\mathbf{z}$, LapTop then constructs the augmented prediction sequence:

$$[\mathbf{x}, \widetilde{\mathbf{Z}}_{1:R}(\mathbf{z}), \mathbf{F}_{1:H}], \tag{45}$$

whose length is $T + R + H$, and predicts all H future positions in one parallel decoding pass. The decoding cost is:

$$C(T + R + H), \tag{46}$$

Thus, the total deterministic inference cost of LapTop is:

$$C(T) + C(T + R + H) + O(d^2), \tag{47}$$

where the $O(d^2)$ term comes from the lightweight prior/posterior parameterization and latent-plan injection.

When generating N sampled trajectories, the history-conditioned prior can be computed once and reused across samples. The total cost becomes:

$$C(T) + N \cdot C(T + R + H) + O(N d^2), \tag{48}$$

Since R is small and fixed, the additional cost introduced by plan tokens and latent injection is minor compared with repeated autoregressive feedback or iterative diffusion refinement. Therefore, LapTop's efficiency advantage comes from avoiding the H sequential rollout passes of autoregressive methods and the S denoising steps of diffusion-based methods, while still supporting trajectory-level prediction through non-iterative parallel decoding.

L Model Inference Cost and Complexity Analysis (RQ4)

Table 5: Comparison of prediction performance with inference cost and complexity. The complexity column reports asymptotic Transformer cost with respect to sequence length n; LapTop uses one history-prior encoding pass and one parallel decoding pass, which preserves the same asymptotic order but changes the constant factor.

Model	MovieLens-1M (len=5)			MovieLens-1M (len=10)			Complexity
	Time Cost↓	SeqMatch@50↑	Seq-NLL↓	Time Cost↓	SeqMatch@50↑	Seq-NLL↓	
SASRec	**3.34s**	0.1045	33.62	3.29s	0.0299	68.99	$\mathcal{O}(nd^2 + dn^2)$
SASRec-AR	4.01s	0.1049	33.58	5.30s	0.0261	70.41	$\mathcal{O}(nd^2 + dn^2)$
DiffuRec	24.68s	0.1168	32.47	21.51s	0.0311	68.35	$\mathcal{O}(nd^2 + dn^2)$
DCRec	12.83s	0.1458	31.42	11.18s	0.0381	68.58	$\mathcal{O}(nd^2 + dn^2)$
LPDO	13.46s	0.1572	29.95	12.26s	0.0449	66.14	$\mathcal{O}(nd^2 + dn^2)$
LapTop	3.46s	**0.1676**	**29.16**	**3.26s**	**0.0591**	**62.26**	$\mathcal{O}(nd^2 + dn^2)$

All results are measured on an NVIDIA Tesla V100. d and n denote the representation dimension and sequence length.

Compared to autoregressive rollout with H sequential decoding passes, LapTop generates H-step predictions through a non-iterative prior inference stage followed by one parallel trajectory decoding pass, avoiding sequential prediction feedback. The additional cost of latent planning is limited to lightweight MLP heads and a small number of plan tokens. Table 5 shows that LapTop achieves inference latency comparable to SASRec and SASRec-AR, while consistently outperforming them in trajectory-level metrics such as SeqMatch@50 and Seq-NLL. In contrast, diffusion-based and iterative generative methods (e.g., DiffuRec and LPDO) incur substantially higher inference costs due to repeated sampling or refinement, resulting in significant slowdowns for longer horizons. Despite processing a slightly longer input sequence, LapTop maintains the same asymptotic complexity as standard transformer-based recommenders, i.e., $O(nd^2 + dn^2)$. Overall, LapTop provides an effective trade-off between inference efficiency and long-horizon trajectory quality, making it suitable for practical deployment in real-world recommender systems.

M Embedding Analysis

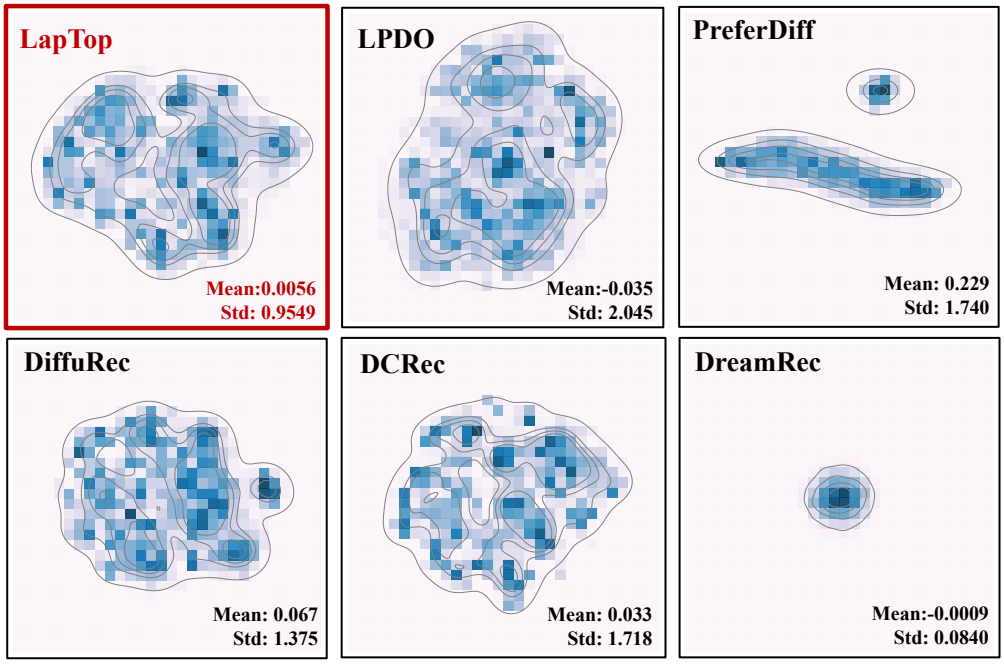


Figure 4: Embedding visualization and comparison on UBTP task. Visualization results from t-SNE on the MovieLens-1M dataset and the target trajectory length is set to be 5.

To further understand how different models capture trajectory-level representations, we visualize the learned trajectory embeddings using t-SNE on the MovieLens-1M dataset with target length K=5 (Figure 4). Compared to diffusion-based and preference-diffusion baselines, LapTop exhibits a more compact and structured embedding distribution, characterized by moderate dispersion and clear local organization. In contrast, PreferDiff and LPDO produce highly dispersed embeddings with large variance, suggesting unstable or overly diffuse trajectory representations, while DreamRec collapses into a narrow region with extremely low variance, indicating limited expressiveness. DiffuRec and DCRec fall between these extremes but still show less coherent global structure than LapTop. Quantitatively, LapTop achieves a balanced embedding variance, reflecting its ability to capture diverse yet consistent future trajectories. These observations suggest that latent planning enables LapTop to organize future behaviors in a well-structured hidden space, effectively balancing diversity and coherence, which aligns with its superior trajectory-level performance reported in Figure 4.

N Sensitivity Analysis

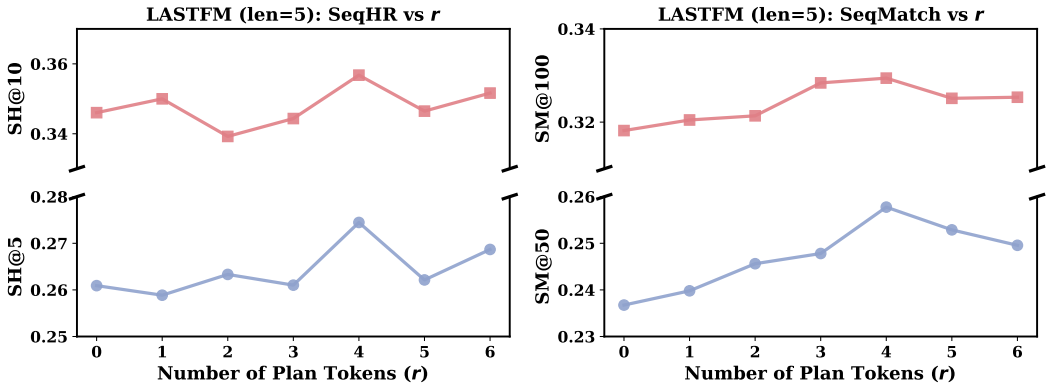


Figure 5: Effect of the number of plan tokens r on SeqHR and SeqMatch on the LastFM dataset with trajectory length H=5. Performance improves as r increases from 0 to a small number, peaking around r=4, after which additional plan tokens yield diminishing returns.

We conduct ablation studies on the LastFM dataset with trajectory length H=5 to examine the contribution of LapTop's key components and the sensitivity to major hyperparameters. Figure 5, Figure 7, and Figure 6 illustrate the performance trends with respect to the number of plan tokens r and the KL weight β.

Effect of the number of plan tokens. Figure 5 analyzes the impact of the number of plan tokens r. We observe that performance improves as r increases from 0 to a small number, peaking around r=4, after which additional plan tokens provide diminishing or negative returns. This suggests that a compact set of plan tokens is sufficient to encode high-level future intent, while excessive tokens introduce redundancy without further benefits.

Effect of KL regularization strength. Figure 6 and Figure 7 show that LapTop is sensitive to the KL weight β. When β=0, the model degenerates into a purely deterministic parallel predictor, leading to inferior performance. Increasing β substantially improves both SHR and SNDCG, with optimal performance achieved at a moderate value (e.g., β=0.1). Further increasing β results in slight performance degradation, suggesting that overly strong regularization constrains the latent space and limits expressive capacity. This behavior is consistent across datasets, indicating that a balanced prior–posterior alignment is crucial for effective latent planning.

Overall, these ablation results validate the design of LapTop: latent planning is indispensable for modeling multi-modal future trajectories, plan tokens serve as an effective hidden-space reasoning buffer for parallel prediction, and moderate hyperparameter settings yield robust performance. Together, these findings explain LapTop's strong long-horizon accuracy and stability observed in the full comparison results.

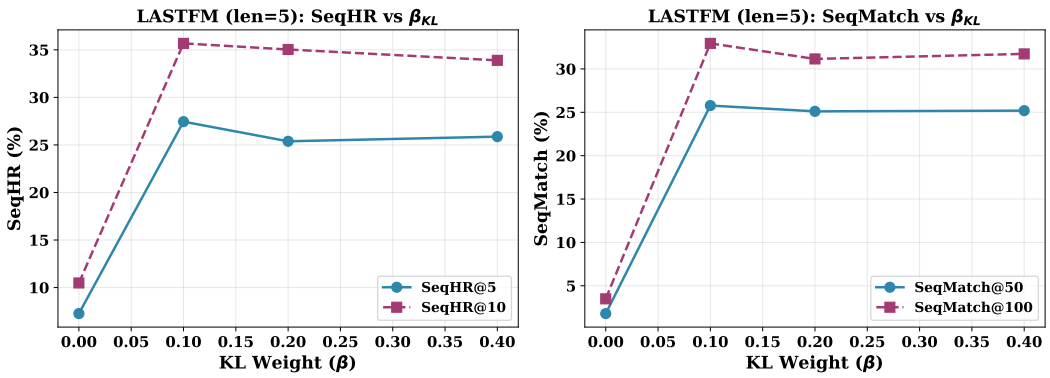


Figure 6: Sensitivity analysis of the KL regularization weight β on the LastFM (trajectory length K=5). When β=0, the model degenerates into a deterministic parallel predictor and exhibits significantly degraded performance. Moderate KL regularization ($\beta \approx 0.1$) yields the best results, while overly large values slightly hurt performance.

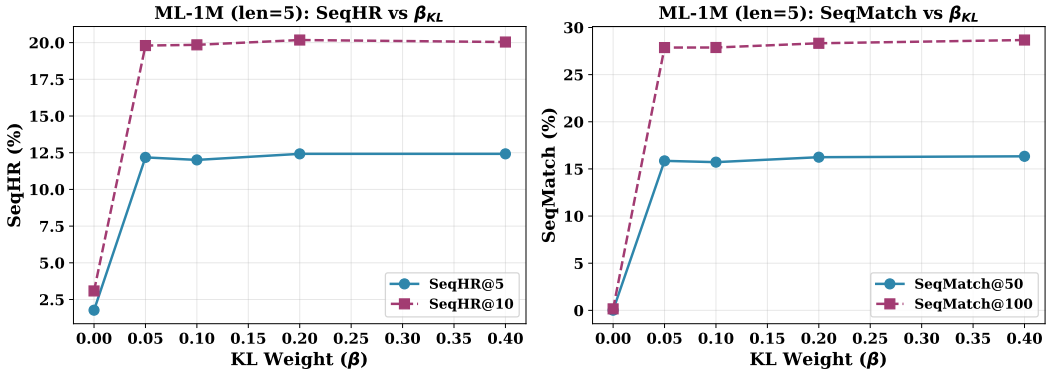


Figure 7: Effect of the KL weight β on SeqHR and SeqMatch on the MovieLens-1M dataset with trajectory length K=5. The trends are consistent with those observed on LastFM: introducing KL regularization substantially improves performance over the deterministic setting, and moderate values of β lead to stable and robust results across metrics.

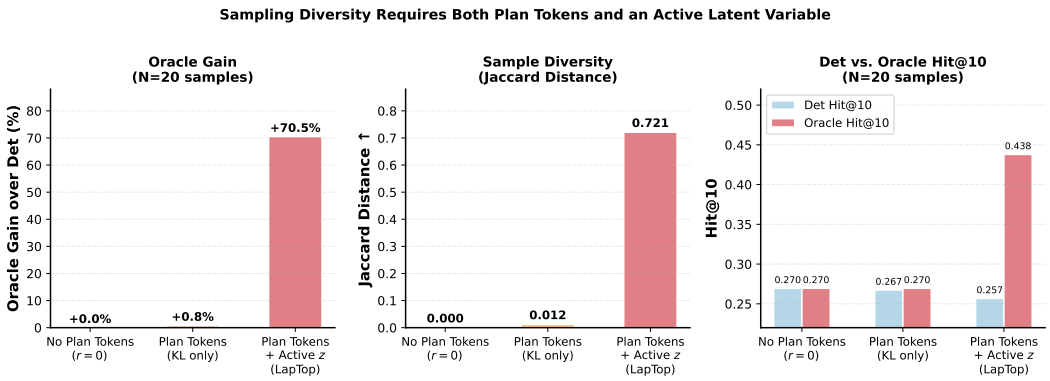


Figure 8: Diagnostic analysis of stochastic trajectory sampling on ML-1M with trajectory length 5 and $N = 20$ samples. The results show that meaningful trajectory diversity requires both an active latent variable and plan-token-mediated latent injection.

O Extend to Long Seq Dataset

Table 6: Statistics of datasets used in long sequence experiment.

Dataset	#Users	#Items	#Interactions	Avg. Length	Pred. Horizon
ML-10M	69,878	10,196	9,998,816	143.1	5
ML-20M	138,493	18,345	19,984,024	144.3	5

To further evaluate the scalability and robustness of LapTop on larger-scale datasets with longer user interaction histories, we conduct additional experiments on MovieLens-10M and MovieLens-20M. The dataset statistics are summarized in Table 6. Both datasets contain substantially more users, items, and interactions than the main benchmark datasets, while the target trajectory length is fixed to 5. These experiments are intended as a scalability stress test; we include SASRec (Kang & McAuley, 2018) and LPDO (Huang et al., 2025c) as representative efficient Transformer-based and trajectory-level diffusion baselines.

As shown in Table 7, LapTop achieves the best performance on both ML-10M and ML-20M across all reported metrics. On ML-10M, compared with the strongest baseline, LapTop improves SH@10 by 11.94%, SN@10 by 10.01%, and Match@50 by 15.11%, while reducing Seq-NLL by 1.07%. On ML-20M, LapTop also demonstrates consistent advantages, improving SH@10 by 20.84%, SN@10 by 19.31%, and Match@50 by 20.40%, while reducing Seq-NLL by 3.36%. These results indicate that the proposed latent planning mechanism remains effective on larger-scale and longer-sequence recommendation settings, improving both sequence-level ranking accuracy and trajectory-level consistency.

P Sampling Quality Analysis

We further evaluate whether stochastic trajectory generation produces not only accurate but also diverse future trajectories. For each user, we generate N candidate trajectories by sampling different latent plans for LapTop, or different noise variables for diffusion-based baselines. We assess sampling quality from two perspectives: best-of-N potential and cross-sample diversity.

For accuracy-oriented evaluation, *Det Hit* measures the performance of a single deterministic prediction, while *Oracle Hit* reports the best Hit score achieved among the N sampled trajectories for each user. The relative improvement from Det Hit to Oracle Hit is reported as *Oracle Gain*, which quantifies the additional benefit brought by stochastic sampling. For diversity-oriented evaluation, we report Self-BLEU (Zhu et al., 2018), which measures the overlap among trajectories generated for the same user, where lower values indicate greater diversity. We also compute the Jaccard distance over the sets of recommended items across samples, where higher values indicate broader exploration of the item space.

Table 7: Performance on large-scale MovieLens datasets with target length 5. All metrics except Seq-NLL are reported in percentage scale ($\times 100$). For all metrics except Seq-NLL, higher is better; for Seq-NLL, lower is better. Bold and underlined values denote the best and second-best results, respectively.

Metric	ML-10M			ML-20M		
	SASRec	LPDO	LapTop	SASRec	LPDO	LapTop
SH@5	9.12	<u>10.14</u>	**11.79**	<u>10.02</u>	9.15	**12.53**
SN@5	5.08	<u>5.65</u>	**6.45**	<u>5.61</u>	5.11	**6.96**
SH@10	15.08	<u>16.67</u>	**18.66**	<u>16.65</u>	15.29	**20.12**
SN@10	6.29	<u>6.89</u>	**7.58**	<u>6.94</u>	6.38	**8.28**
SH@20	23.32	<u>25.39</u>	**27.78**	<u>25.71</u>	23.63	**30.11**
SN@20	7.34	<u>7.90</u>	**8.53**	<u>8.00</u>	7.39	**9.28**
Match@50	12.20	<u>12.71</u>	**14.63**	<u>15.44</u>	11.31	**18.59**
Match@100	21.17	<u>22.15</u>	**24.30**	<u>25.14</u>	19.49	**29.29**
Seq-NLL(↓)	32.97	<u>31.78</u>	**31.44**	<u>31.56</u>	32.71	**30.50**

Table 8: Sampling quality comparison on ML-1M with trajectory length 5.

Metric	SASRec	DCRec	LPDO	**LapTop**
Det Hit ↑	0.1588	0.2396	0.2591	0.2566
Best-of-N (Oracle Hit) ↑	0.1588	0.2603	0.2825	0.4070
Oracle Gain ↑	0%	8.6%	9.5%	58.6%
Self-BLEU ↓	1.0000	0.9028	0.8910	0.4325
Jaccard Distance ↑	0.0000	0.0972	0.1090	0.5675

Table 8 reports the results on ML-1M with trajectory length 5 and $N = 10$ sampled trajectories. Diffusion-based samplers such as DCRec and LPDO achieve strong deterministic accuracy, but their oracle improvements are relatively limited, with gains of only 8.6% and 9.5%, respectively. This suggests that repeated sampling from these models tends to produce highly similar ranked lists, which is also reflected by their high Self-BLEU scores, close to 0.9, and low Jaccard distances.

In contrast, LapTop achieves a substantially higher oracle upper bound. Although its deterministic Hit score is comparable to LPDO, sampling from the latent plan increases Oracle Hit from 0.2566 to 0.4070, corresponding to a relative gain of 70.4%. This improvement is accompanied by much lower cross-sample overlap, with Self-BLEU decreasing to 0.4325 and Jaccard distance increasing to 0.5675. These results indicate that the latent planning mechanism enables LapTop to explore multiple plausible future modes rather than repeatedly generating near-identical trajectories. Finally, SASRec obtains zero oracle gain and a Self-BLEU score of 1.0, as expected from a deterministic model without stochastic trajectory sampling. We do not claim that oracle selection is directly available in deployment; rather, Best-of-N measures the coverage potential of the learned latent space. Deterministic decoding remains the default setting for accuracy-oriented serving, while sampled trajectories provide candidates for downstream reranking or exploration.

To further diagnose where the sampling diversity comes from, we conduct an ablation study in Figure 8 with $N = 20$ sampled trajectories. We compare three variants of LapTop: removing plan tokens ($r = 0$), using plan tokens with only standard KL regularization (*KL only*), and the full model with an active latent variable. The *KL only* variant uses the standard Gaussian KL objective without free bits, which tends to cause posterior collapse and produces nearly deterministic samples. The *Active z* variant (LapTop) follows the free-bits strategy in variational inference (Kingma et al., 2016) to encourage the latent plan to remain informative.

The results show that plan tokens and an informative latent variable are both necessary for effective stochastic trajectory exploration. When plan tokens are removed, sampling different latent variables has no effect on the predicted trajectories, resulting in zero Oracle Gain and zero Jaccard Distance. When plan tokens are retained but the latent variable collapses under standard KL regularization, the model still produces only marginal diversity. In contrast, the full model achieves substantial

oracle improvement ($+70.5\%$) and much higher sample diversity (Jaccard Distance $= 0.721$). These findings confirm that the latent variable alone is insufficient: the sampled plan must be transmitted through the plan-token interface in order to influence future prediction slots and generate diverse yet plausible trajectories.

Q Causal Attention and Information Flow

We use the standard causal self-attention mask in the SASRec Transformer backbone (Kang & McAuley, 2018). For the augmented prediction sequence $[x, \widetilde{Z}_{1:R}(z), F_{1:H}]$, each token can only attend to previous valid tokens under the causal mask. Therefore, a future query slot F_j can attend to all non-padded history tokens, all latent-conditioned plan tokens, and earlier future query slots $F_{<j}$, but cannot attend to later future query slots $F_{>j}$. This design allows horizon-wise future queries to exchange information in a causal manner while preventing access to future positions.

During training, the ground-truth future trajectory y is used only for inferring the posterior latent plan $q_\phi(z \mid x, y)$ and for computing the prediction loss. It is never included in the prediction sequence that produces item logits. Specifically, the posterior path encodes $[x, y]$ to infer z, whereas the prediction path encodes $[x, \widetilde{Z}_{1:R}(z), F_{1:H}]$, where $F_{1:H}$ are learnable query tokens independent of y. The posterior and prior use separate MLP heads to parameterize $q_\phi(z \mid x, y)$ and $p_\psi(z \mid x)$, while sharing the same Transformer backbone with the prediction path. This prevents label leakage and ensures that, at inference time, trajectories are generated using only the history-conditioned prior.

Table 9: Model performance on MovieLens-1M with target trajectory length $H = 5$.

Backbone	SHR@5 ↑	SNDCG@5 ↑	SHR@10 ↑	SNDCG@10 ↑	Seq-NLL ↓
Bidirectional	0.1142	0.0628	0.1884	0.0786	33.74
Prefix	0.1226	0.0680	0.2009	0.0834	30.87
Causal	0.1277	0.0706	0.2091	0.0866	29.85

The results in Table 9 further justify our choice of causal attention for the prediction path. Compared with the bidirectional variant, the causal backbone achieves consistently better performance across all sequence-level ranking metrics, improving SHR@5 from 0.1142 to 0.1277 and SHR@10 from 0.1884 to 0.2091, while also reducing Seq-NLL from 33.74 to 29.85. The prefix variant also improves over the bidirectional variant, suggesting that restricting information flow during trajectory decoding is beneficial for future prediction. However, the causal variant achieves the best overall results, indicating that allowing each future query to attend to the history, latent plan tokens, and earlier future queries provides a more suitable inductive bias for trajectory generation. These results support our design that causal horizon-wise information flow can coordinate parallel future prediction while avoiding access to later future positions.

R First-Horizon Recommendation Performance

To complement the trajectory-level evaluation, we report conventional next-item ranking metrics at the first future position on MovieLens-1M. We use the existing models trained with prediction horizon $H = 5$ and evaluate only their first-position predictions, without retraining. All models are evaluated on the same 6,040 test users and corresponding first-position targets. This experiment assesses immediate next-item ranking within the multi-step prediction setting, rather than the performance of models separately trained with $H = 1$.

As shown in Table 10, LapTop achieves the highest observed scores across all six metrics. Compared with LPDO, LapTop improves HR@10 from 0.257285 to 0.266391, corresponding to a relative improvement of approximately 3.54%, while NDCG@10 increases from 0.134090 to 0.135326. The differences at some cutoffs are small, particularly for HR@5 and NDCG@20. These results indicate that LapTop maintains competitive immediate next-item ranking while learning to predict complete future trajectories. This evaluation complements the trajectory-level results; by itself, it does not isolate the contribution of cross-horizon coordination.

Table 10: First-horizon recommendation performance on MovieLens-1M. All models are trained with prediction horizon $H = 5$ and evaluated only at the first future position without retraining. Higher values indicate better performance. Bold values denote the best result for each metric.

Model	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
SASRec	0.092881	0.057403	0.158775	0.078612	0.258609	0.103788
DiffuRec	0.105629	0.065317	0.175662	0.087759	0.281457	0.114268
DCRec	0.151987	0.096585	0.240232	0.124881	0.362914	0.155872
LPDO	0.160596	0.101157	0.257285	0.134090	0.387914	0.166712
LapTop	**0.161093**	**0.103118**	**0.266391**	**0.135326**	**0.390232**	**0.167045**

S Cold-Start Evaluation

We evaluate LapTop under interaction-limited cold-start conditions from both the user and item perspectives on MovieLens-1M, with prediction horizon $H = 5$. The user-side experiment simulates limited observed history through recent-history truncation, while the item-side experiment evaluates targets with limited training interactions. All methods are evaluated using existing checkpoints without retraining or fine-tuning. These experiments assess cold-start conditions with limited interaction evidence: they do not use a held-out-user split or evaluate previously unseen item IDs.

S.1 User-Side Cold Start with Limited History

To simulate limited-history cold-start conditions, we retain only the $L = 3$ or $L = 5$ most recent historical interactions for each test user. The same five-item ground-truth future trajectory is preserved across both settings and all compared methods. This paired evaluation controls the prediction targets while varying the amount of historical evidence available at inference.

Table 11: User-side cold-start evaluation on MovieLens-1M with prediction horizon $H = 5$. L denotes the number of observed historical interactions. SH, SN, and SM denote SeqHR, SeqNDCG, and SeqMatch, respectively. Higher values are better except for Seq-NLL. Bold values indicate the best result at each history length.

L	Model	SH@5 ↑	SN@5 ↑	SH@10 ↑	SN@10 ↑	SM@50 ↑	Seq-NLL ↓
3	SASRec	0.012211	0.007276	0.025056	0.011323	0.000326	39.301274
3	LPDO	0.014431	0.008230	0.029710	0.013201	0.000651	38.791202
3	LapTop	**0.019939**	**0.011748**	**0.037666**	**0.016680**	**0.005971**	**37.616442**
5	SASRec	0.024829	0.014854	0.046206	0.020561	0.010365	38.306680
5	LPDO	0.033414	0.019264	0.060103	0.026239	0.016824	37.205852
5	LapTop	**0.039955**	**0.022735**	**0.067214**	**0.028545**	**0.034411**	**35.722243**

Table 11 shows that LapTop achieves the best observed performance across all reported metrics at both history lengths. Compared with LPDO, LapTop improves SeqHR@10 from 0.029710 to 0.037666 at $L = 3$ and from 0.060103 to 0.067214 at $L = 5$, corresponding to relative improvements of approximately 26.8% and 11.8%, respectively. The advantage also extends to strict trajectory-level matching: SeqMatch@50 increases from 0.000651 to 0.005971 at $L = 3$, and from 0.016824 to 0.034411 at $L = 5$. LapTop additionally obtains the lowest Seq-NLL in both settings.

All three methods benefit from observing five rather than three interactions, highlighting the importance of historical evidence. The results suggest that LapTop's latent trajectory conditioning remains useful under limited-history cold-start conditions, although absolute trajectory-matching rates remain low when only three interactions are available.

S.2 Item-Side Cold Start with Limited Interactions

We further evaluate item-side cold-start performance using items with limited training interactions. Item frequencies are computed exclusively from the training split. Among items with non-zero training frequency, the bottom 20% are designated as cold items. The resulting subset contains 740 items and 300 valid ground-truth positions across 264 test trajectories. The numbers of valid targets

at horizons 1–5 are 51, 45, 52, 68, and 84, respectively. Evaluation is restricted to positions whose ground-truth item belongs to this cold-item subset.

Table 12: Item-side cold-start evaluation on MovieLens-1M with prediction horizon $H = 5$. Cold items are defined using the bottom 20% of non-zero training item frequencies. Metrics are evaluated on the corresponding cold-item target positions. Higher values are better except for NLL. Bold values indicate the best results, including ties.

Model	SHR@10 ↑	SNDCG@10 ↑	SHR@20 ↑	SNDCG@20 ↑	NLL ↓
SASRec	0.000000	0.000000	0.000000	0.000000	11.701878
LPDO	0.000000	0.000000	0.022579	0.007696	10.990507
LapTop	**0.019656**	**0.006861**	**0.025936**	**0.009342**	**9.873274**

As shown in Table 12, LapTop achieves the highest observed SHR and SNDCG at both reported cutoffs. LapTop is the only compared method with non-zero SHR@10 and SNDCG@10 on this subset. It also obtains the lowest NLL of 9.873274, corresponding to reductions of approximately 10.2% relative to LPDO and 15.6% relative to SASRec. These results provide complementary evidence that LapTop remains effective when target items have limited interaction support in the training data. Since this evaluation contains only 300 valid target positions, we interpret it as suSeq-NLLementary evidence of performance under the specified cold-item setting.