PLAN GLOBALLY, ACT LOCALLY: ADAPTIVE SEGMENT FORECASTING FOR LONG-HORIZON HUMAN MOBILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon human mobility forecasting must predict a user's future movements over multiple days without accumulating errors from repeated short-horizon rollouts. Human mobility has a natural hierarchy: people make high-level plans across days, while their movements appear as fine-grained location choices over time. Following the principle of planning globally and acting locally, we propose **PlanCAD** (Plan–Critic Adaptive Diffusion) for long-horizon human mobility forecasting. PlanCAD uses daily plan tokens to capture long-horizon structure and segment-level generation to produce slot-level location predictions. Specifically, PlanCAD uses a large language model (LLM)-based mobility backbone to encode historical trajectories into day-level representations, and a learnable planner generates future daily plan tokens from these representations. A critic scores candidate execution horizons using plan-level and trajectory-level statistics, deciding whether to predict a longer segment or replan earlier. Given the selected plan, a diffusion actor refines future mobility representations in the LLM hidden space and maps them to location predictions. Experiments on real-world mobility datasets show that PlanCAD improves 7-to-7 Acc@3 on Kumamoto from 50.09% to 51.10% over the strongest baseline, with consistent gains on Sapporo and Hiroshima. Ablation studies and analysis further show that complementary critic features support adaptive horizon selection, while plan-conditioned generation improves multi-day forecasting stability. Code is available at `https://anonymous.4open.science/r/PlanCAD-BAF8`.

1 INTRODUCTION

Long-horizon mobility forecasting fails in two characteristic ways: predictors that roll out one day at a time accumulate errors as each prediction becomes the input to the next, while predictors that commit to a fixed multi-day window must choose that window before knowing whether the current context supports it. Both failure modes leave the same question unanswered: how far ahead should the model forecast before updating its context and replanning? Human mobility is up to 93% predictable in aggregate yet varies substantially in fine-grained location choices Song et al. (2010), reflecting a hierarchy in which people make high-level plans across days while their movements unfold as slot-level choices over time. A person may commute, shop, or visit leisure places repeatedly across days, yet visit them in different orders, at different times, or interleaved with different stops. Long-horizon forecasting therefore requires both capturing this multi-day regularity and deciding, at each step, how far the current context can be trusted.

Most existing methods were designed for short horizons, and their long-horizon extensions usually rely on either recursive rollout or fixed-length prediction. Mobility forecasting methods, including Markov and probabilistic models Qiao et al. (2018), recurrent networks Feng et al. (2018), attention-based models Yang et al. (2022), graph-based models Zhang et al. (2025), and recent LLM-based models, have achieved strong performance on next-location and next-day prediction by modeling temporal regularity and spatial dependency He et al. (2025); Gong et al. (2024). However, extending these short-horizon predictors to multiple future days is non-trivial. A common strategy is recursive rollout, which repeatedly feeds predicted locations back into the historical context, but this can amplify early errors over time. Another strategy is direct fixed-window prediction, which generates a

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.

pre-defined future horizon in one pass, but the horizon length is fixed before inference and may not match the reliability of the current context Nie et al. (2022); Wang et al. (2024a). Recent segment-level, block-based, or diffusion-style generators reduce the number of recursive updates by predicting larger future units, yet they still typically rely on a fixed segment length Rasul et al. (2021); Yuan & Qiao (2024); Long et al. (2025b). Since the reliable forecasting length may vary across users, days, and contexts, these strategies can either update too frequently through one-day rollout or commit to an unreliable future segment when the context becomes uncertain. These strategies therefore leave a key question unresolved: how far should the model forecast from the current context?

This question highlights two complementary considerations for long-horizon mobility forecasting. At a coarse scale, daily routines provide structure for future mobility across days. At a fine scale, location choices remain context-dependent, making the reliable forecasting length vary across situations. Together, these considerations motivate a "plan globally, act locally" view: planning future day-level structure, evaluating candidate segment horizons, and generating slot-level predictions locally.

Following this view, we propose **PlanCAD** (Plan–Critic Adaptive Diffusion), a Plan–Critic–Act framework for long-horizon human mobility forecasting. Given historical trajectories, PlanCAD uses an LLM-based mobility backbone to obtain day-level representations, from which a learnable planner produces future latent daily plan tokens. These plan tokens provide coarse multi-day structure without directly committing to all future slot-level locations. To determine how far the current plan should be used, a critic evaluates candidate execution horizons using plan-level and trajectory-level statistics. Conditioned on the selected horizon and its corresponding plan, a diffusion Ho et al. (2020) actor refines future mobility representations in the LLM hidden space and maps them to slot-level location predictions. In this way, PlanCAD connects global planning, adaptive horizon evaluation, and local generation within a unified long-horizon forecasting framework.

Our main contributions are summarized as follows:

- We propose **PlanCAD**, a long-horizon mobility forecasting framework that explicitly couples plan-level structure with adaptive segment execution, moving beyond both repeated one-day rollout and fixed-horizon prediction.
- We introduce horizon-aware plan conditioning and a supervised critic that selects among candidate plan segments using plan-level and trajectory-level statistics, thereby determining how far the model forecasts before updating the prediction.
- We design a plan-conditioned diffusion actor that refines future mobility representations in the LLM hidden space and maps them to location logits, preserving slot-level prediction detail while using latent daily plan tokens as high-level guidance.
- We evaluate PlanCAD on real-world multi-day mobility forecasting tasks and show gains in forecasting accuracy and geographical consistency. The analysis further suggests that reliable long-horizon forecasting requires combining trajectory regularity with plan.

2 Related Work

Mobility and Time-series Forecasting. Recent mobility and time-series forecasting has progressed from context-aware next-location prediction to transferable and generative modeling. CMHSA captures temporal, spatial, and user-level dependencies with contextual self-attention Hong et al. (2023), while DSTPP models spatio-temporal event uncertainty via diffusion Yuan et al. (2023). PMT pretrains trajectory representations for mobility tasks Wu et al. (2026), COLA separates shared and city-specific factors for cross-city simulation Wang et al. (2024b), and ST-MoE-BERT uses mixture-of-experts routing for cross-city transfer He et al. (2024). Other forecasters improve sequence modeling through patching Nie et al. (2022), multi-periodicity Wu et al. (2022), inverted tokenization Liu et al. (2023), multiscale mixing Wang et al. (2024a), test-time adaptation Han et al. (2025), trajectory-flow diffusion Long et al. (2025b), location tokenization Chen et al. (2025), conditional diffusion Long et al. (2025a), uncertainty modeling Shen et al. (2026); Yuan & Qiao (2024), non-stationarity handling Liu et al. (2024a), and sparse scaling Shi et al. (2024). These methods improve forecasting and generation, but most focus on next-location prediction, short-horizon continuation, or fixed-granularity forecasting. They therefore do not directly address multi-day rollout degradation, where errors accumulate over time.

LLM-based Mobility and Foundation Forecasting. A more recent line uses LLMs or foundation models for mobility and long-horizon forecasting. LLM-Mob adapts prompted LLMs to time-aware mobility prediction Wang et al. (2023), Mobility-LLM reprograms check-in sequences with intention memory Gong et al. (2024), AgentMove uses memory and world knowledge for zero-shot mobility reasoning Feng et al. (2025), and MoveFM-R adds language-based reflection Meng et al. (2025). RHYTHM introduces hierarchical temporal tokenization with a frozen LLM backbone He et al. (2025), while Time-LLM maps time series into a frozen LLM interface Jin et al. (2023). Foundation forecasters such as Chronos, TimesFM, Moirai, and Timer-XL show the benefit of pretraining and long-context modeling Ansari et al. (2024); Das et al. (2023); Woo et al. (2024); Liu et al. (2024b). However, these methods usually view the future as a fixed-unit output stream, without separating high-level planning, plan-conditioned execution, and adaptive replanning. PlanCAD addresses this gap with a Plan–Critic–Act framework.

3 Problem Formulation

Long-horizon mobility forecasting. Let $\mathcal{X}_{1:T} = \{x_1, x_2, \ldots, x_T\}$ denote a user's historical mobility trajectory, where each record $x_i = (t_i, l_i)$ contains the timestamp and location. The location l_i belongs to a finite location set $\mathcal{L}$. Let H denote the number of time slots in one day and K denote the forecasting horizon in days. Given a future timestamp sequence $\mathcal{T}^{(K)}_{T+1:T+KH} = \{t_{T+1}, t_{T+2}, \ldots, t_{T+KH}\}$, our goal is to predict the corresponding future location sequence $\mathcal{Y}^{(K)}_{T+1:T+KH} = \{l_{T+1}, l_{T+2}, \ldots, l_{T+KH}\}$.

Long-horizon forecasting is commonly handled by one-day autoregressive rollout or fixed-horizon prediction. The former accumulates feedback errors, while the latter cannot adapt how far the model should execute before replanning. We analyze both limitations in Appendix B.

Adaptive segment-level plan-conditioned forecasting. Instead of forecasting the entire horizon with a fixed length, we decompose long-horizon mobility forecasting into adaptive segment-level prediction. Let R be the remaining forecasting horizon and $k \in \{1, \ldots, \min(K_{\text{seg}}, R)\}$ be a candidate segment length, where K_{seg} is the maximum segment length. For each k, we associate a plan condition $\mathcal{P}^{(k)}$ that represents the high-level mobility intention of the next k-day segment. We then model the segment as a plan-conditioned predictive distribution.

At each segment step, the adaptive forecasting objective is to determine a suitable segment length and predict the corresponding future segment:

$$k^\star = \arg \max_{k \in \{1,\ldots,\min(K_{\text{seg}},R)\}} S(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k), \qquad \hat{\mathcal{Y}}^{(k^\star)} \sim p_\theta \left(\mathcal{Y}^{(k^\star)} \mid \mathcal{X}_{1:T}, \mathcal{P}^{(k^\star)}, \mathcal{T}^{(k^\star)} \right),$$

where $S(\cdot)$ is a horizon suitability function, $p_\theta(\cdot)$ denotes a parameterized plan-conditioned predictive distribution over future locations, and $\mathcal{T}^{(k^\star)}$ denotes the future timestamps of the selected segment. The remaining horizon is updated as $R \leftarrow R - k^\star$, and the process continues until $R = 0$.

4 Method

In this section, we introduce PlanCAD by decomposing long-horizon forecasting into three stages: planning future day-level intentions, selecting an execution horizon, and generating locations.

4.1 Overview: Plan–Critic–Act

Figure 1 illustrates the proposed Plan–Critic–Act framework for long-horizon mobility forecasting. Given the current historical window, the mobility representation backbone converts historical records into day-level tokens and encodes known future timestamps into temporal queries, providing shared context for planning, horizon selection, and generation. The planner produces future daily plan tokens on a fixed segment canvas. For each candidate execution horizon, a masked plan keeps the selected plan tokens and suppresses unselected future tokens. The critic scores these candidate horizons using trajectory, plan, and horizon features, and selects the execution length for the next segment. Conditioned on the selected plan and future timestamp queries, the diffusion actor refines the future

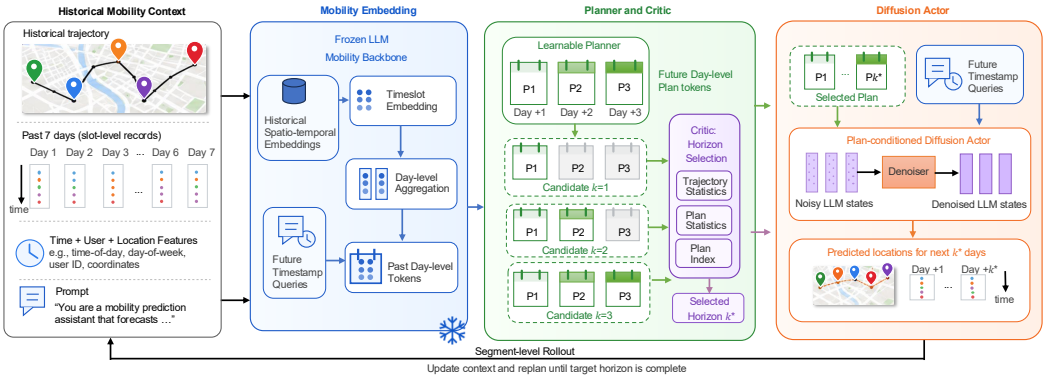

Figure 1: Overview of PlanCAD. Given a historical mobility window, PlanCAD builds day-level mobility tokens, generates future plan tokens, selects an adaptive execution horizon with a critic, and uses a plan-conditioned diffusion actor to predict the next segment. The segment is appended to the context and the process repeats until the target horizon is completed.

segment in the frozen LLM hidden space and projects the denoised representations to location logits. This procedure is repeated until the target forecasting horizon is completed.

4.2 Mobility Representation Backbone

We use an LLM-based mobility backbone to convert structured mobility records into contextual representations for planning and generation. Each historical record is encoded from temporal attributes and observed spatial attributes, while future slots are encoded only from known temporal attributes. Specifically, the historical sequence is first mapped into time-slot representations by an embedding module $\mathrm{Embed}(\cdot)$, and then grouped into $M = T/H$ historical days, where H is the number of slots per day. A daily aggregation module $\mathrm{DayAgg}(\cdot)$ produces day-level historical tokens:

$$E_{1:M} = \mathrm{DayAgg}(\mathrm{Embed}(\mathcal{X}_{1:T})).$$

These tokens summarize historical mobility patterns and provide the shared context for the planner and critic. For future prediction, since future locations are unknown, we encode only known future temporal attributes into query embeddings:

$$Q^{(k)} = \mathrm{TimeEnc}(\mathcal{T}^{(k)}).$$

Here, $\mathrm{Embed}(\cdot)$, $\mathrm{DayAgg}(\cdot)$, and $\mathrm{TimeEnc}(\cdot)$ denote the historical spatio-temporal encoder, the slot-to-day aggregation module, and the future temporal query encoder, respectively. We further enrich the historical tokens and future queries with offline prompt-derived semantic embeddings computed by a frozen LLM. Details of the modules and prompt templates are provided in Appendix C.1.

4.3 Horizon-aware Planning and Critic Selection

Given the historical day-level tokens $E_{1:M}$, we use a learnable-query cross-attention Vaswani et al. (2017) planner to generate a sequence of future daily plan tokens:

$$P_{1:K_{\mathrm{seg}}} = \mathrm{Planner}(E_{1:M}),$$

where P_j represents the high-level mobility plan for the j-th future day. The planner uses learnable future-day queries to attend to historical day-level tokens, enabling horizon-aware plan construction.

For a candidate execution horizon k, we construct a horizon-specific plan by keeping the first k plan tokens and setting the remaining $K_{\mathrm{seg}} - k$ tokens to zero:

$$\mathcal{P}^{(k)} = \mathrm{Mask}(P_{1:K_{\mathrm{seg}}}, k).$$

Here, $\mathrm{Mask}(\cdot, k)$ denotes horizon masking over daily plan tokens. This keeps the planning canvas fixed while preventing the actor from conditioning on unselected future plan tokens.

The critic evaluates each candidate horizon through a two-step scoring process. First, a feature extractor $\Phi(\cdot)$ constructs a candidate-level feature vector from the current mobility context, the masked plan, and the horizon index:

$$\phi_k = \Phi(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k), \quad k \in \{1, \ldots, \min(K_{\mathrm{seg}}, R)\}.$$

Here, ϕ_k summarizes both trajectory-level context features and plan-level statistics for the candidate horizon k. Then, a learned critic scorer $C_\psi(\cdot)$, as an MLP, maps this feature vector to a scalar score:

$$s_k = C_\psi(\phi_k) \approx S(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k).$$

The scores s_k are later supervised by segment-level prediction quality and used to select the execution horizon during rollout. Implementation details of the planner, horizon masking, feature extractor Φ, and critic scorer C_ψ are provided in Appendix C.2.

4.4 Plan-conditioned Diffusion Actor

The actor models the plan-conditioned predictive distribution by generating fine-grained future mobility representations. Given a candidate segment length k, the actor first aligns the plan condition $\mathcal{P}^{(k)}$ with future time slots. The j-th daily plan token is assigned to the H slots of the j-th future day:

$$A^{(k)} = \mathrm{Align}(\mathcal{P}^{(k)}), \qquad C^{(k)} = \mathrm{Fuse}\big(Q^{(K_{\mathrm{seg}})}, A^{(k)}\big),$$

where $Q^{(K_{\mathrm{seg}})} = \mathrm{TimeEnc}(\mathcal{T}^{(K_{\mathrm{seg}})})$ contains known future temporal attributes, and $C^{(k)}$ is the plan-temporal condition on the fixed K_{seg}-day canvas. Positions beyond the candidate horizon kH are ignored by the valid-horizon mask. Instead of diffusing over discrete location identifiers, the actor performs hidden-space diffusion in the frozen LLM representation space. Let $U^{\mathrm{LLM}}_{1:K_{\mathrm{seg}}H}$ denote the clean future hidden states produced by the frozen LLM backbone under the plan-conditioned queries. At diffusion timestep τ, the noisy hidden states are obtained as

$$U^{(k)}_\tau = \sqrt{\bar{\alpha}_\tau} U^{\mathrm{LLM}}_{1:K_{\mathrm{seg}}H} + \sqrt{1-\bar{\alpha}_\tau}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

The denoiser predicts noise given the plan-temporal sequence and segment plan summary:

$$\hat{\epsilon} = D_\theta\left(U^{(k)}_\tau, \tau \mid C^{(k)}, \bar{\mathcal{P}}^{(k)}\right),$$

where D_θ is the learnable denoising network, $C^{(k)}$ provides slot-level plan-temporal conditioning, and $\bar{\mathcal{P}}^{(k)}$ denotes a segment-level summary of the plan condition. During training, we use a progressive noise schedule that gradually increases the maximum diffusion timestep, stabilizing early hidden-space denoising. At inference, we apply a single-step refinement to the hidden representations by sampling the noise once and applying the denoiser once: $O^{(k)} = W_o \hat{U}^{(k)}_0 + b_o$, where W_o and b_o are trainable output projection parameters. The logits define the plan-conditioned predictive distribution:

$$p_\theta\left(\mathcal{Y}^{(k)} \mid \mathcal{X}_{1:T}, \mathcal{P}^{(k)}, \mathcal{T}^{(k)}\right) = \prod_{i=1}^{kH} \mathrm{Cat}\left(l_{T+i}; \mathrm{softmax}(O^{(k)}_i)\right).$$

Here, $\mathrm{Cat}(\cdot;\pi)$ denotes a categorical distribution with probabilities π over the location set $\mathcal{L}$, and $O^{(k)}_i$ is the location logit vector for future slot $T+i$. Details of plan-to-slot alignment, plan-temporal fusion, hidden-space diffusion, and the progressive noise schedule appear in Appendix C.3.

4.5 Learning and Segment-level Rollout

The actor outputs location logits $O^{(k)}_{1:K_{\mathrm{seg}}H}$ over the location set $\mathcal{L}$, where $O^{(k)}_{i,\ell}$ denotes the logit of location ℓ at slot i. For an executed horizon k, only the first kH positions are valid, and the corresponding predicted segment is obtained from these logits as

$$\hat{\mathcal{Y}}^{(k)} = \{\hat{y}_1, \ldots, \hat{y}_{kH}\}, \qquad \hat{y}_i = \arg\max_{\ell \in \mathcal{L}} O^{(k)}_{i,\ell}.$$

During training, ground-truth future locations are available and are used to supervise both segment generation and horizon selection. For each candidate horizon $k \in \{1, \ldots, \min(K_{\mathrm{seg}}, R)\}$, we apply the masked plan condition $\mathcal{P}^{(k)}$, run the actor on the fixed K_{seg}-day canvas, and evaluate the classification loss over the first kH positions. The candidate with the lowest segment-level loss defines an oracle horizon k^{sup}. The critic is trained so that its scores s_k approximate this oracle choice, while the actor is trained on the valid positions of the selected segment.

At test time, future labels are unavailable, so the oracle horizon k^{sup} cannot be computed. The model instead selects the execution horizon using the learned critic:

$$k^\star = \arg \max_{k \in \{1,\ldots,\min(K_{\mathrm{seg}},R)\}} s_k, \qquad s_k = C_\psi(\Phi(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k)).$$

The actor then generates the corresponding $k^\star$-day segment in one forward pass. The historical window is advanced using the predicted locations, with spatial coordinates recomputed from the predicted location identifiers. The remaining horizon is updated as $R \leftarrow R - k^\star$, and the procedure repeats until $R = 0$. Detailed learning objectives are provided in Appendix C.4, and the overall training and rollout procedures are summarized in Appendix C.5.

4.6 Analysis: Computational Efficiency and Error Propagation

Computational efficiency. Our efficiency comes from historical tokenization and segment-level execution. Day-level aggregation compresses the MH historical slots into $M = T/H$ historical tokens, so a K_{seg}-day segment is processed with attention length $M + K_{\mathrm{seg}}H$ rather than $MH + K_{\mathrm{seg}}H$, reducing the per-segment attention cost from $\mathcal{O}((MH+K_{\mathrm{seg}}H)^2)$ to $\mathcal{O}((M+K_{\mathrm{seg}}H)^2)$. In addition, for a K-day forecast, adaptive segment-level rollout requires S executions with $\lceil K/K_{\mathrm{seg}} \rceil \leq S \leq K$, instead of K one-day executions. The resulting rollout cost is $\mathcal{O}(S(M+K_{\mathrm{seg}}H)^2)$, reducing sequential model invocations and context updates while preserving slot-level future generation.

Error propagation. Segment-level generation reduces feedback-induced error accumulation by delaying context updates. As derived in Appendix B, in one-day rollout, errors can propagate through K daily feedback updates, with the simplified expansion $e_K \leq \gamma^K e_0 + \sum_{r=1}^{K} \gamma^{K-r}\epsilon_r$, where γ denotes the error amplification factor. In our method, the selected plan provides a shared condition within each segment, and the critic can shorten the execution horizon when long segment execution is unreliable. A selected k-day segment is generated in one pass and predictions are fed back only after the segment is completed, resulting in S segment-level feedback updates: $e_S \leq \gamma^S e_0 + \sum_{s=1}^{S} \gamma^{S-s}\epsilon_s$. Since $S \leq K$, our method shortens the feedback chain through which early errors can propagate.

5 Experiments

In this section, we evaluate PlanCAD on multi-day human mobility forecasting on real-world datasets.

5.1 Experimental Setup

Datasets. We evaluate PlanCAD on three real-world human mobility datasets from YJMob100K Yabe et al. (2024): Kumamoto, Sapporo, and Hiroshima. Each day is divided into 48 half-hour slots, with chronological train/validation/test splits by day. Dataset statistics are provided in Appendix D.1.

Evaluation metrics. We primarily evaluate location prediction with Accuracy@q (Acc@q), which measures whether the ground-truth location appears among the top-q predicted candidates. We also report MRR for ranking quality, DTW for spatial trajectory alignment, and BLEU for sequence-level overlap between predicted and ground-truth trajectories. Acc@q and BLEU are reported in percentage (%). Detailed metric definitions and reported q values are provided in Appendix D.2.

Settings. We evaluate two long-horizon tasks: 7-to-3 and 7-to-7, where the model predicts the next 3 or 7 days from the past 7 days, respectively. Each future day contains 48 time slots. For PlanCAD, the maximum segment length is set to 3 days. All reported results are averaged over three independent runs with different random seeds. Hyperparameters are selected based on validation performance, with computational resources and detailed settings reported in Appendix D.3.

Baselines. We compare PlanCAD with recurrent baselines, LSTM Hochreiter & Schmidhuber (1997) and DeepMove Feng et al. (2018); transformer-based baselines, PatchTST Nie et al. (2022), iTransformer Liu et al. (2023), ST-MoE-BERT He et al. (2024), PMT Wu et al. (2026), CMHSA Hong et al. (2023), and COLA Wang et al. (2024b); and the LLM-based baseline Mobility-LLM Gong et al. (2024). Time-LLM Jin et al. (2023) is included in the transferability and efficiency analysis and

Table 1: Main forecasting results on three cities. Acc@q is reported in percentage (%) under 7-to-3 and 7-to-7 settings, where higher is better. Results are averaged over three runs; standard deviations are omitted since all are $\leq 2\%$. The **best** and second-best results are highlighted.

	Kumamoto				Sapporo				Hiroshima			
	7-to-3		7-to-7		7-to-3		7-to-7		7-to-3		7-to-7	
Model	Acc@1	Acc@3	Acc@1	Acc@3	Acc@1	Acc@3	Acc@1	Acc@3	Acc@1	Acc@3	Acc@1	Acc@3
LSTM	24.07	44.03	22.01	39.75	23.05	39.37	20.16	34.51	23.18	40.78	20.34	36.10
DeepMove	26.83	48.78	24.97	45.29	27.49	46.16	25.52	42.72	25.31	43.55	23.32	40.19
PatchTST	26.64	48.11	26.15	48.04	**30.00**	**50.12**	29.16	48.84	29.16	**50.60**	28.02	48.79
iTransformer	25.51	47.43	25.48	48.21	27.30	49.34	26.83	48.87	25.45	47.84	24.97	46.97
ST-MoE-BERT	24.04	44.41	22.97	42.67	28.28	47.73	27.21	45.96	27.00	47.38	26.37	45.84
PMT	27.53	50.30	26.68	48.79	28.50	48.35	27.40	46.39	27.72	48.81	26.81	47.18
COLA	27.54	49.28	27.42	49.52	29.47	49.68	28.97	48.90	29.06	50.55	28.73	50.18
CMHSA	**28.56**	50.41	27.91	50.09	28.87	48.92	27.94	47.55	28.27	49.55	27.68	48.55
Mobility-LLM	20.41	37.93	16.03	30.14	22.23	38.19	21.54	35.98	23.01	45.60	21.21	39.04
PlanCAD-Full	28.02	**50.43**	**28.50**	**51.10**	29.84	49.26	**29.40**	**50.42**	**30.86**	50.08	**30.30**	**50.94**

uses Llama-3.2-1B as the pretrained backbone. For fair comparison, we equip time-series models with the same spatio-temporal embeddings. For methods designed for next-day prediction, we produce long-horizon forecasts through autoregressive rollout by appending each predicted day to the historical window. Unless otherwise specified, each baseline follows its original publication and is assessed on identical input windows, forecasting horizons, and evaluation metrics.

Model variants. We evaluate several PlanCAD variants to isolate the effect of planning, horizon selection, critic design, diffusion refinement, and backbone choice. To assess adaptive execution, we compare against fixed-horizon variants, FIXED-1 and FIXED-3, which always execute one-day or three-day segments. We also include ORACLE, which follows the full PlanCAD configuration but directly uses the oracle-best candidate horizon. For the actor, we evaluate NO-PLAN, NO-DIFF, MATCHED-NO-DIFF (a parameter-matched deterministic decoder), and RANDOM-NOISE-DIFF to examine the roles of plan conditioning and hidden-space diffusion. For critic design, PLANCAD-FULL uses the hybrid critic, while CRITIC-PLAN, CRITIC-TRAJ, and CRITIC-HORIZON isolate different candidate features for horizon scoring. Finally, we test different frozen LLM backbones, denoted as PLANCAD-LLAMA-3B, PLANCAD-QWEN-1.5B, PLANCAD-GEMMA-2B, and PLANCAD-DEEPSEEK-1.5B. Detailed definitions of all variants are provided in Appendix D.4.

5.2 OVERALL SUPERIORITY AND COMPONENT ANALYSIS

Accuracy Table 1 shows that PlanCAD-Full achieves competitive slot-level accuracy across all three cities, with clearer advantages in the challenging 7-to-7 setting. On Kumamoto, it achieves the best Acc@3 under 7-to-3 and the best Acc@1/Acc@3 under 7-to-7, improving over the strongest baseline from 27.91% to 28.50% in Acc@1 and from 50.09% to 51.10% in Acc@3. On Sapporo, PlanCAD-Full achieves the second-best Acc@1 under 7-to-3 and the best Acc@1/Acc@3 under 7-to-7, while on Hiroshima it achieves the best Acc@1 under both horizons and the best Acc@3 under 7-to-7. These results suggest that PlanCAD-Full becomes more advantageous as the horizon increases. Unlike CMHSA or Mobility-LLM, PlanCAD-Full combines daily plan guidance with critic-guided horizon selection for adaptive long-horizon forecasting. Additional execution-mode and lead-day analyses in Appendix D.5 show that PlanCAD remains competitive under direct multi-day baseline prediction and maintains its advantage across forecast lead days.

Geographical Evaluation. Table 10 reports MRR, BLEU, and DTW under 7-to-3 and 7-to-7 settings in Appendix D.6. PlanCAD-Full achieves competitive geographical and ranking performance without explicitly optimizing for these metrics. Under 7-to-3, it achieves the best MRR on Hiroshima (0.4153) and the second-best on Kumamoto (0.4042). Under 7-to-7, it achieves the best MRR on Sapporo (0.4114) and Hiroshima (0.4181), the second-best on Kumamoto (0.4115), and the best BLEU on Kumamoto (16.73). These results show that the benefits of adaptive planning extend beyond slot-level accuracy to long-horizon ranking and geographical consistency.

Component Ablation. Table 2 shows the contribution of the Plan–Critic–Act design. The fixed-horizon variants highlight the need for adaptive execution: FIXED-1 maintains strong accuracy but has the longest inference time, while FIXED-3 is faster but drops to 25.27% Acc@1 and 45.46% Acc@3. PlanCAD-Full achieves comparable performance to the oracle selector, matching its 50.43%

Table 2: Component ablation on Kumamoto under the 7-to-3 forecasting setting. We report accuracy, geographical consistency, and efficiency. Acc@q, MRR, and BLEU are reported in percentage (%). Training and inference time are measured in seconds per 200 users. Higher is better for Acc@q, MRR, and BLEU; lower is better for DTW and runtime. The **best** and second-best results are highlighted.

Model	Acc@1	Acc@3	Acc@5	MRR	BLEU	DTW	Training	Inference
ORACLE	27.98	50.43	56.85	40.39	13.95	5668	1675	77.98
FIXED-1	28.10	50.15	56.78	39.96	12.51	5875	1650	94.31
FIXED-3	25.27	45.46	51.73	36.46	10.97	8983	1210	62.42
RANDOM-NOISE-DIFF	0.50	1.26	1.82	1.14	0.05	32583	**574**	**40.58**
MATCHED-NO-DIFF	**28.27**	**50.87**	**56.93**	**40.64**	13.50	**5508**	2241	79.59
NO-DIFF	27.95	50.15	56.67	40.33	13.36	5659	1082	66.78
NO-PLAN	25.70	48.60	56.10	38.42	11.24	5585	1342	76.94
PLANCAD-FULL	28.02	50.43	56.86	40.42	**13.99**	5667	1669	74.43

Table 3: Critic feature ablation on Kumamoto with Llama-1B. We report accuracy and geographical consistency. Acc@q, MRR, and BLEU are reported in percentage (%). Higher is better for Acc@q, MRR, and BLEU; lower is better for DTW. The **best** and second-best results are highlighted.

Model	7-to-3						7-to-7					
CRITIC-	Acc@1	Acc@3	Acc@5	MRR	BLEU	DTW	Acc@1	Acc@3	Acc@5	MRR	BLEU	DTW
PLAN	28.04	**50.91**	**57.32**	**40.57**	13.78	**5486**	28.31	51.08	**58.34**	41.01	**17.09**	5472
TRAJ	**28.08**	50.67	57.03	40.45	**14.31**	5580	28.40	**51.39**	**58.34**	41.12	17.06	5463
HORIZON	28.05	50.55	56.90	40.46	13.91	5508	28.29	51.09	57.94	40.95	16.39	**5387**
PLANCAD-FULL	28.02	50.43	56.86	40.42	13.99	5667	**28.50**	51.10	58.32	**41.15**	16.73	5480

Acc@3 while improving Acc@5, MRR, and BLEU. Removing plan conditioning reduces Acc@1 from 28.02% to 25.70% and BLEU from 13.99% to 11.24%, confirming the value of plan guidance. MATCHED-NO-DIFF performs better on several metrics, while PlanCAD-Full achieves the best BLEU, highlighting complementary behavior between deterministic decoding and diffusion refinement. Under severe perturbations, the diffusion actor retains higher Acc@1 and Acc@3 than its parameter-matched counterpart in the 7-to-3 setting, suggesting robustness to latent corruption; additional analyses appear in Appendix D.7.

5.3 UNDERSTANDING ADAPTIVE HORIZON SELECTION

Feature Correlations. To understand how the critic chooses execution horizons, we first analyze the correlation between candidate statistics and forecasting quality in Figure 2(a). Trajectory-level statistics are more informative: transition rate is negatively correlated with Acc and Within-500m accuracy (-0.71 and -0.63), while repeat stay shows positive correlations (0.65 and 0.53). This suggests that frequent transitions make long-horizon prediction harder, whereas repeated stays provide stronger regularity. Plan-level dispersion features show weaker correlations, indicating that they are insufficient alone but can still offer complementary uncertainty cues. This motivates a hybrid critic that combines plan statistics with recent trajectory regularity.

Critic Feature Ablation. Table 3 examines the effect of critic features on forecasting performance. For 7-to-3, the plan-only critic performs strongly, achieving the best Acc@3 (50.91%), Acc@5 (57.32%), MRR (40.57%), and DTW (5486), while the trajectory-based critic achieves the best Acc@1 (28.08%) and BLEU (14.31%). For 7-to-7, PlanCAD-Full provides balanced performance, achieving the best Acc@1 (28.50%) and MRR (41.15%), while remaining close to the best Acc@3 and Acc@5. The trajectory-based critic leads in Acc@3, the plan-only critic achieves the best BLEU, and the horizon-based critic obtains the lowest DTW. These results indicate that different critic features favor different aspects of long-horizon forecasting rather than yielding a dominant configuration. We therefore adopt the hybrid critic in PlanCAD-Full as the default, incorporating plan, trajectory, and horizon information for balanced performance across settings.

Critic-selected Plan Behavior. We study the selected plans in Figure 2(b) to understand how plan-level signals shape adaptive horizon selection; statistical details are provided in Appendix D.8. The dispersion differs across settings: 7-to-7 covers a wider range, while 7-to-3 concentrates around 0.23–0.32 and rarely falls below 0.2. This suggests that dispersion has horizon-dependent meanings. For 7-to-3, similar plan tokens provide limited evidence for distinguishing short segments, so the critic favors locally actionable plans with sufficient day-to-day variation. For 7-to-7, low dispersion can

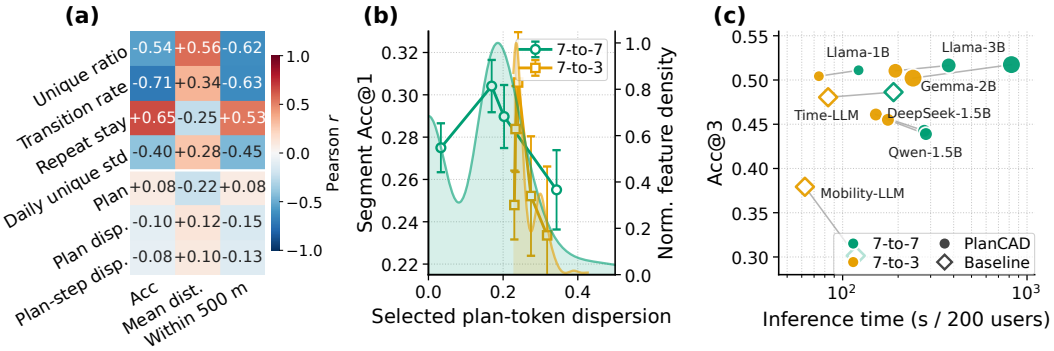

Figure 2: Analysis of PlanCAD behavior. (a) Pearson correlation between critic-related features and geographical metrics, where darker red indicates stronger positive correlation and darker blue indicates stronger negative correlation. (b) Binned segment-level Acc@1 over selected plan-token dispersion; markers show bin means, error bars denote ± 1 standard error. (c) Accuracy–efficiency trade-off across PlanCAD backbones and baselines; higher and further left indicates better accuracy and faster inference. Gray lines are visual guides for related models/settings.

instead indicate stable mobility intentions over a longer rollout, making a broader range acceptable when recent trajectory context is regular.

The accuracy curve further shows that dispersion is non-monotonic. For 7-to-7, Acc@1 rises from about 27.5% at low dispersion to about 30.5% around 0.17, then falls to about 25.5% above 0.34. Moderate dispersion reflects day-to-day variation, while excessive dispersion indicates scattered future intentions and lower reliability. For 7-to-3, the narrower range and larger error bars indicate that dispersion is less separable for short segments. Overall, plan-level statistics provide horizon-dependent uncertainty cues, but reliable adaptive selection still requires trajectory-level context.

5.4 Transferability and Efficiency.

We evaluate PlanCAD's transferability and efficiency across frozen LLM backbones. Figure 2(c) shows that Llama- and Gemma-based variants achieve the strongest accuracy-efficiency trade-off. In the 7-to-3 setting, Gemma-2B achieves the highest Acc@3 of 51.04%, while Llama-1B reaches a comparable 50.43% with much lower inference time. This suggests that stronger backbones can improve prediction quality, but the gain is not always proportional to the added computational cost. In the 7-to-7 setting, Llama-1B achieves a competitive Acc@3 of 51.10% and BLEU of 16.73% with only 122.79 seconds per 200 users, outperforming Time-LLM (48.63%) and Mobility-LLM (30.14%) in Acc@3. Although Llama-3B achieves strong accuracy, its larger parameter scale leads to a less favorable efficiency trade-off. Overall, these results show that PlanCAD transfers across LLM backbones and can operate effectively with lightweight frozen models.

6 Conclusion and Limitations

We propose PlanCAD, a long-horizon mobility forecasting framework. PlanCAD uses daily plan tokens to represent coarse multi-day intentions, a critic to choose the execution horizon, and a diffusion actor to generate local slot-level predictions. Experiments show that PlanCAD outperforms state-of-the-art methods and is especially effective as the forecasting horizon increases. The analysis further suggests that reliable long-horizon forecasting requires combining trajectory regularity with plan: moderate plan variation supports multi-day execution.

Limitations. PlanCAD still has several limitations. First, its performance can vary with the choice and quality of the frozen LLM backbone, suggesting that better mobility-oriented backbone adaptation may further improve robustness. Second, although adaptive segment execution improves inference efficiency, training PlanCAD is more expensive than training simple recurrent or transformer baselines due to the additional planner, critic, and diffusion modules. Third, the critic is currently trained with supervised segment-level oracle signals; future work could explore reinforcement learning or online feedback to learn adaptive horizon policies more directly.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and assisting with translation. For method implementation, generative AI assisted with modifying, debugging, and refining code based on the authors' specified methods. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, drafting parts of the research paper, summarizing and analyzing existing literature, sourcing and searching for information, editing the research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. AI-assisted code was reviewed and tested for correctness by the authors, and AI-assisted writing and literature-related content was reviewed and verified against the relevant sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses anonymized human mobility data, where individuals are represented only by anonymous identifiers and no personally identifying information is available. Nevertheless, mobility trajectories are inherently privacy-sensitive, and fine-grained movement patterns may reveal information about individuals or communities. Mobility models may also inherit biases in the underlying data and could be misused for surveillance, profiling, or individual tracking. Accordingly, practical use should employ appropriate access control, privacy protection, and fairness safeguards. Our work is intended for mobility forecasting research and should not be used to identify or track individuals or support high-stakes decisions about them.

REPRODUCIBILITY STATEMENT

We use publicly available mobility datasets and pretrained model backbones. We provide detailed descriptions of the PlanCAD architecture, training procedure, and inference pipeline in the main paper and appendix. The appendix further reports implementation details, hyperparameter settings, dataset preprocessing and evaluation protocols, as well as definitions of the model variants used in our ablation studies. We report results over multiple random seeds for the main experiments and specify the experimental settings for additional analyses. The code, including detailed instructions to reproduce the main experimental results, is publicly available at `https://anonymous.4open.science/r/PlanCAD-BAF8`.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Yile Chen, Yicheng Tao, Yue Jiang, Shuai Liu, Han Yu, and Gao Cong. Enhancing large language models for mobility analytics with semantic location tokenization. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 262–273, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*, pp. 1459–1468, 2018.

Jie Feng, Yuwei Du, Jie Zhao, and Yong Li. Agentmove: A large language model based agentic framework for zero-shot next location prediction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1322–1338, 2025.

Letian Gong, Yan Lin, Xinyue Zhang, Yiwen Lu, Xuedi Han, Yichen Liu, Shengnan Guo, Youfang Lin, and Huaiyu Wan. Mobility-llm: Learning visiting intentions and travel preference from human

mobility data with large language models. *Advances in Neural Information Processing Systems*, 37:36185–36217, 2024.

Huaxu Han, Shuliang Wang, Sijie Ruan, Qianyu Yang, Yuxuan Liang, Ziqiang Yuan, Cheng Long, Hanning Yuan, and Yu Zheng. Adamove: Efficient test-time adaptation for human mobility prediction. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 3655–3667. IEEE, 2025.

Haoyu He, Haozheng Luo, and Qi R Wang. St-moe-bert: A spatial-temporal mixture-of-experts framework for long-term cross-city mobility prediction. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Human Mobility Prediction Challenge*, pp. 10–15, 2024.

Haoyu He, Haozheng Luo, Yan Chen, and Qi R Wang. Rhythm: Reasoning with hierarchical temporal tokenization for human mobility. *arXiv preprint arXiv:2509.23115*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Ye Hong, Yatao Zhang, Konrad Schindler, and Martin Raubal. Context-aware multi-head self-attentional neural network model for next location prediction. *Transportation Research Part C: Emerging Technologies*, 156:104315, 2023.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Peiyuan Liu, Beiliang Wu, Yifan Hu, Naiqi Li, Tao Dai, Jigang Bao, and Shu-tao Xia. Timebridge: Non-stationarity matters for long-term time series forecasting. *arXiv preprint arXiv:2410.04442*, 2024a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. *arXiv preprint arXiv:2410.04803*, 2024b.

Qingyue Long, Can Rong, Huandong Wang, and Yong Li. One fits all: General mobility trajectory modeling via masked conditional diffusion. *arXiv preprint arXiv:2501.13347*, 2025a.

Qingyue Long, Yuan Yuan, and Yong Li. A universal model for human mobility prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 894–905, 2025b.

Fanjin Meng, Yuan Yuan, Jingtao Ding, Jie Feng, Chonghua Han, and Yong Li. Movefm-r: Advancing mobility foundation models via language-driven semantic reasoning. *arXiv preprint arXiv:2509.22403*, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Yuanyuan Qiao, Zhongwei Si, Yanting Zhang, Fehmi Ben Abdesslem, Xinyu Zhang, and Jie Yang. A hybrid markov-based model for human mobility prediction. *Neurocomputing*, 278:99–109, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.05.101. Recent Advances in Machine Learning for Non-Gaussian Data Processing.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Lifeng Shen, Xuyang Li, and Lele Long. Tsgdiff: Rethinking synthetic time series generation from a pure graph perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25304–25311, 2026.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. *arXiv preprint arXiv:2409.16040*, 2024.

Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010. doi: 10.1126/science.1177170.

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

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024a.

Xinglei Wang, Meng Fang, Zichao Zeng, and Tao Cheng. Where would i go next? large language models as human mobility predictors. *arXiv preprint arXiv:2308.15197*, 2023.

Yu Wang, Tongya Zheng, Yuxuan Liang, Shunyu Liu, and Mingli Song. Cola: Cross-city mobility transformer for human trajectory simulation. In *Proceedings of the ACM Web Conference 2024*, pp. 3509–3520, 2024b.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Xinhua Wu, Haoyu He, Yanchao Wang, and Qi R Wang. Pretrained mobility transformer: A foundation model for human mobility. *IEEE Transactions on Intelligent Transportation Systems*, 2026.

Takahiro Yabe, Kota Tsubouchi, Toru Shimizu, Yoshihide Sekimoto, Kaoru Sezaki, Esteban Moro, and Alex Pentland. Yjmob100k: City-scale and longitudinal dataset of anonymized human mobility trajectories. *Scientific Data*, 11(1):397, 2024.

Song Yang, Jiamou Liu, and Kaiqi Zhao. Getnext: trajectory flow map enhanced transformer for next poi recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval*, pp. 1144–1153, 2022.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742*, 2024.

Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. Spatio-temporal diffusion point processes. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 3173–3184, 2023.

Qi Zhang, Zhenliang Ma, Pengfei Zhang, and Erik Jenelius. Mobility knowledge graph: review and its application in public transport. *Transportation*, 52(3):1119–1145, 2025.

A Extended Related Work

Human Mobility Prediction. Human mobility prediction aims to infer future locations from historical trajectories by capturing recurring routines, temporal periodicity, spatial dependency, and user-specific preferences. Early Markov and probabilistic models characterize mobility as transitions between locations, but their limited memory makes it difficult to represent long-range and multi-scale

patterns. Recurrent neural networks and attention-based models improve sequential modeling by learning dependencies across historical visits, while transformer-based methods further enhance long-range temporal reasoning. Graph-based models complement these approaches by modeling spatial relations among locations and interactions between users and places. Together, these methods have achieved strong performance on next-location and next-day prediction tasks. However, they are mostly designed for short-horizon forecasting, and extending them to multiple future days usually requires an external strategy.

Long-Horizon Mobility Forecasting. To obtain long-horizon mobility forecasts, existing studies commonly rely on three types of extension strategies: recursive rollout, direct fixed-window prediction, and segment-level or block-based generation. Recursive rollout repeatedly applies a next-location or next-day predictor and feeds the predicted locations back into the input context. Although simple and model-agnostic, this strategy can amplify early errors over time because later predictions are conditioned on previously generated results. Direct fixed-window prediction avoids such repeated feedback by generating a pre-defined future horizon in one pass, but the horizon length is fixed before inference and may not match the reliability of the current context. Segment-level or block-based generation provides a third strategy by predicting future mobility in larger temporal units rather than fully step by step; diffusion-style generators and block-wise sequence models are representative examples of this direction. This can reduce the number of recursive updates, but the segment or block length is typically fixed before inference. Overall, these strategies lack an explicit mechanism for deciding how far the current forecast should be trusted before replanning.

LLM-Based Mobility Modeling. LLMs have emerged as general-purpose foundation models and have been successfully adapted beyond natural language to various sequential domains, including time series forecasting and human mobility modeling. Freezing pretrained LLM backbones entirely has emerged as an alternative to parameter-efficient fine-tuning, with recent work repurposing these models as sequential representation extractors that lower computational overhead. In mobility forecasting, LLM-based models typically combine temporal tokenization, prompt-based conditioning, and trajectory-specific embeddings to encode historical trajectories and future temporal queries. Such designs are effective for capturing recurring routines, periodic patterns, and long-range dependencies in human movement. Nevertheless, they mainly focus on representation learning and short-horizon or fixed-horizon prediction, without explicitly deciding how future daily plans should be selected and executed over long horizons.

B Analysis of Rollout Strategy Limitations

Autoregressive rollout and error propagation. Most mobility forecasting models are trained for a short horizon and extended to longer horizons by autoregressive rollout. Let $\mathcal{X}^{(0)} = \mathcal{X}_{1:T}$ be the initial context. A one-day predictor first generates $\hat{\mathcal{Y}}^{(1)} = \{\hat{x}_{T+1}, \ldots, \hat{x}_{T+H}\} = f(\mathcal{X}^{(0)}, \mathcal{T}^{(1)}, 1)$. To predict the following days, the oldest H records are removed after each step and the generated day is appended back to the input context. Therefore, before predicting the k-th future day, the context is

$$\mathcal{X}^{(k-1)} = \{x_{(k-1)H+1}, \ldots, x_T, \hat{x}_{T+1}, \ldots, \hat{x}_{T+(k-1)H}\}, \quad (k-1)H < T.$$

The k-th day is then predicted as

$$\hat{\mathcal{Y}}^{(k)} = \{\hat{x}_{T+(k-1)H+1}, \ldots, \hat{x}_{T+kH}\} = f(\mathcal{X}^{(k-1)}, \mathcal{T}_k^{(1)}, 1).$$

Here the last argument remains 1, since autoregressive rollout repeatedly applies a one-day predictor. This shows that, as k increases, the original prefix $\{x_1, \ldots, x_{(k-1)H}\}$ is replaced by generated records $\{\hat{x}_{T+1}, \ldots, \hat{x}_{T+(k-1)H}\}$. When $(k-1)H = T$, the original history is completely shifted out and the model conditions entirely on its own generated trajectory. Thus, an early prediction error is not only an output error but also becomes part of the input for later predictions, leading to error propagation across rollout steps.

To make this feedback effect explicit, let e_r denote the context error after the r-th daily update, ϵ_r denote the prediction error introduced at that update, and γ denote the amplification factor from existing context error to later prediction error. A simplified recurrence for one-day rollout is

$$e_r \leq \gamma e_{r-1} + \epsilon_r, \qquad r = 1, \ldots, K.$$

Unrolling this recurrence over K daily updates gives

$$e_K \leq \gamma^K e_0 + \sum_{r=1}^{K} \gamma^{K-r} \epsilon_r.$$

This expression shows that errors introduced at early rollout steps can be repeatedly carried forward and amplified through subsequent daily context updates.

B.1 Fixed-horizon prediction

Another common strategy is to train a model with a fixed multi-day horizon K_f:

$$\hat{\mathcal{Y}}^{(K_f)} = f(\mathcal{X}^{(0)}, \mathcal{T}^{(K_f)}, K_f).$$

This avoids updating the context after every single day, but it also fixes the execution length in advance. When K_f is small, the model still requires frequent feedback updates and may accumulate rollout errors. When K_f is large, the model must generate a long future segment even under high uncertainty, which can reduce prediction reliability. Moreover, if the remaining horizon is shorter than K_f, the model must either generate unnecessary steps or rely on truncation.

C Model Detail

C.1 Mobility Representation Backbone Details

This appendix details the representation modules used in the main text: $\mathrm{Embed}(\cdot)$, $\mathrm{DayAgg}(\cdot)$, and $\mathrm{TimeEnc}(\cdot)$, as well as the prompt-derived semantic embeddings. The historical encoder $\mathrm{Embed}(\cdot)$ maps each observed mobility record into a slot-level representation. The daily aggregation module $\mathrm{DayAgg}(\cdot)$ converts slot-level representations into structural day-level tokens. The future temporal encoder $\mathrm{TimeEnc}(\cdot)$ constructs structural query embeddings from known future temporal attributes without using future locations. Prompt-derived semantic embeddings are then added to the structural historical tokens and future queries.

Historical mobility embedding. Each historical record is represented as

$$x_i = (u_i, \mathrm{tod}_i, \mathrm{dow}_i, c_i, l_i),$$

where u_i is the user identifier, tod_i and dow_i denote time-of-day and day-of-week, c_i denotes spatial coordinates, and l_i is the discrete location identifier. We encode temporal and spatial-location attributes separately:

$$z_i^{\mathrm{temp}} = \mathrm{Proj}_{\mathrm{temp}}\left([e_u(u_i); e_{\mathrm{tod}}(\mathrm{tod}_i); e_{\mathrm{dow}}(\mathrm{dow}_i)]\right),$$

$$z_i^{\mathrm{spat}} = \mathrm{Proj}_{\mathrm{spat}}\left([\phi_c(c_i); e_l(l_i)]\right).$$

Here, $\mathrm{Proj}_{\mathrm{temp}}(\cdot)$ and $\mathrm{Proj}_{\mathrm{spat}}(\cdot)$ are learnable MLP projections that map temporal and spatial-location features to the LLM hidden dimension. The historical time-slot embedding is obtained by combining these two components:

$$z_i = z_i^{\mathrm{temp}} + z_i^{\mathrm{spat}}.$$

In implementation, $\phi_c(\cdot)$ projects both raw coordinates and their periodic transformations, e.g., sine and cosine features, to better represent spatial structure. Applying this encoder to all historical records gives

$$Z_{1:T} = \mathrm{Embed}(\mathcal{X}_{1:T}) = \{z_1, \ldots, z_T\}.$$

Daily aggregation. Given $T = MH$ historical time slots, we divide the historical sequence into M daily segments, each containing H slots. For the m-th historical day, the corresponding slot-level embeddings are

$$Z_m = \{z_{(m-1)H+1}, \ldots, z_{mH}\}.$$

We use a hierarchical aggregation module to convert slot-level embeddings into day-level tokens. First, an intra-day attention block processes the H slot embeddings within each day:

$$\widetilde{Z}_m = \mathrm{Attn}_{\mathrm{intra}}(Z_m).$$

Then, a learnable pooling layer summarizes the H refined slot embeddings into a single day-level token:

$$e_m = \mathrm{Pool}(\widetilde{Z}_m).$$

After obtaining the preliminary daily tokens $\bar{E}_{1:M} = \{e_1, \ldots, e_M\}$, we apply inter-day attention to capture cross-day dependencies:

$$E^{\mathrm{str}}_{1:M} = \mathrm{Attn}_{\mathrm{inter}}(\bar{E}_{1:M}).$$

Thus,

$$E^{\mathrm{str}}_{1:M} = \mathrm{DayAgg}(Z_{1:T}).$$

This aggregation captures within-day mobility patterns while preserving cross-day temporal structure. It also reduces the effective sequence length from T time slots to M day-level tokens before the representations are used by the LLM-based backbone and downstream modules.

Future temporal queries. For prediction, future timestamps are available but future locations are unknown. Therefore, for a candidate horizon k, we encode only known future temporal attributes:

$$q_j = \mathrm{Proj}_{\mathrm{fut}}\left([e_u(u); e_{\mathrm{tod}}(\mathrm{tod}_{T+j}); e_{\mathrm{dow}}(\mathrm{dow}_{T+j})]\right), \qquad j = 1, \ldots, kH,$$

where $\mathrm{Proj}_{\mathrm{fut}}(\cdot)$ is a learnable projection network to the LLM hidden dimension. The structural future query sequence is denoted by

$$Q^{\mathrm{str},(k)} = \mathrm{TimeEnc}(\mathcal{T}^{(k)}) = \{q_1, \ldots, q_{kH}\}.$$

No future location identifiers or coordinates are used in constructing these structural future queries.

Prompt-derived semantic embeddings. In addition to structural spatio-temporal representations, we use prompt-derived semantic embeddings to provide high-level mobility context. These embeddings are computed offline using a frozen LLM and are added to the corresponding historical day tokens and future temporal queries. The LLM is not prompted online during training or rollout.

For each historical day m, let

$$X_m = \{x_{(m-1)H+1}, \ldots, x_{mH}\}$$

denote the mobility records within that day. We construct a trajectory-description prompt $\mathrm{Prompt}_{\mathrm{hist}}(X_m)$ that summarizes the user's daily movement pattern, including time-stamped locations, key transitions, and major stay locations. The frozen LLM encodes this prompt, and we use the hidden state of the final token as the semantic embedding:

$$S^{\mathrm{hist}}_m = \mathrm{SelectLast}\left(\mathrm{LLM}_{\mathrm{frozen}}\left(\mathrm{Prompt}_{\mathrm{hist}}(X_m)\right)\right).$$

The enriched historical day token is then

$$E_m = E^{\mathrm{str}}_m + S^{\mathrm{hist}}_m, \qquad m = 1, \ldots, M.$$

For future prediction, future locations are unknown, but the prediction task and future temporal attributes are known. We therefore construct a task-description prompt $\mathrm{Prompt}_{\mathrm{task}}(\mathcal{T}^{(k)})$, which describes the forecasting task, temporal granularity, and the expected mobility prediction format. Its semantic embedding is

$$S^{\mathrm{task}} = \mathrm{SelectLast}\left(\mathrm{LLM}_{\mathrm{frozen}}\left(\mathrm{Prompt}_{\mathrm{task}}(\mathcal{T}^{(k)})\right)\right).$$

The enriched future temporal query is

$$Q^{(k)}_j = Q^{\mathrm{str},(k)}_j + S^{\mathrm{task}}, \qquad j = 1, \ldots, kH.$$

No future location identifiers or coordinates are used in either the future temporal query or the task prompt.

Prompt templates. We use two types of prompts to obtain semantic embeddings.

Historical trajectory prompt

This is the trajectory of user <User_ID> on day <Day_ID> (<Day_of_Week>). The trajectory contains <N> records at fixed time intervals. The coordinate sequence is: <time: (x, y)>. Key transitions include: <transition summary>. Main stay locations include: <stay summary>.

Future task prompt

You are a mobility prediction assistant. Given the user's historical movement patterns and future timestamps, predict the user's future locations in a grid-based urban environment. The prediction should capture frequent locations, daily routines, temporal regularities, and weekday/weekend differences.

C.2 Planner and Critic Details

Learnable-query planner. Given historical day-level tokens $E_{1:M} \in \mathbb{R}^{M\times d}$, the planner generates K_{seg} future daily plan tokens on a fixed segment canvas. We introduce K_{seg} learnable future-day query vectors

$$R_{1:K_{\text{seg}}} = \{r_1, \ldots, r_{K_{\text{seg}}}\}, \qquad r_j \in \mathbb{R}^d,$$

where each r_j is a trainable query vector used to extract the plan token for the j-th future day and d is the LLM hidden dimension. These queries attend to the historical day-level tokens through multi-head attention:

$$H^{\text{plan}}_{1:K_{\text{seg}}} = \text{MHA}\left(Q = R_{1:K_{\text{seg}}}, K = E_{1:M}, V = E_{1:M}\right),$$

where $\text{MHA}(\cdot)$ denotes multi-head attention. We then apply a Transformer-style residual update with layer normalization and a feed-forward network:

$$\tilde{P}_{1:K_{\text{seg}}} = \text{LN}\left(R_{1:K_{\text{seg}}} + H^{\text{plan}}_{1:K_{\text{seg}}}\right),$$

$$P_{1:K_{\text{seg}}} = \tilde{P}_{1:K_{\text{seg}}} + \text{FFN}_{\text{plan}}(\tilde{P}_{1:K_{\text{seg}}}).$$

Here, $\text{LN}(\cdot)$ denotes layer normalization, and FFN_{plan} is a two-layer MLP with hidden dimension $4d$, GELU activation, and dropout. The resulting $P_j \in \mathbb{R}^d$ is the unmasked plan token for the j-th future day.

Horizon masking. For a candidate execution horizon k, we use a fixed K_{seg}-day planning canvas and mask future plan tokens beyond the candidate horizon:

$$\mathcal{P}^{(k)} = \text{Mask}(P_{1:K_{\text{seg}}}, k), \qquad \mathcal{P}^{(k)}_j = \begin{cases} P_j, & j \le k, \\ \mathbf{0}, & j > k. \end{cases}$$

Thus, $\mathcal{P}^{(k)} \in \mathbb{R}^{K_{\text{seg}}\times d}$ always has the same length, while only the first k daily plan tokens remain nonzero. Here, $P_{1:K_{\text{seg}}}$ denotes the unmasked planner output, whereas $\mathcal{P}^{(k)}$ denotes the candidate-specific plan condition used by subsequent modules.

Candidate plan statistics. The hybrid feature extractor $\Phi(\cdot)$ combines plan-level statistics and historical trajectory statistics, as detailed below. For each candidate horizon k, the feature extractor summarizes the candidate-specific plan condition $\mathcal{P}^{(k)}$ into scalar statistics. Let $\bar{\mathcal{P}}^{(k)}$ denote the mean of the nonzero plan tokens:

$$\bar{\mathcal{P}}^{(k)} = \frac{1}{k}\sum_{j=1}^{k} \mathcal{P}^{(k)}_j = \frac{1}{k}\sum_{j=1}^{k} P_j.$$

We use the normalized horizon length

$$r_k = \frac{k}{K_{\text{seg}}},$$

the norm of the plan summary

$$n_k = \|\bar{\mathcal{P}}^{(k)}\|_2,$$

the within-plan token dispersion

$$d_k^{\mathrm{tok}} = \frac{1}{k}\sum_{j=1}^{k}\left\|\mathcal{P}_j^{(k)} - \bar{\mathcal{P}}^{(k)}\right\|_2,$$

and the stepwise variation between consecutive daily plan tokens

$$d_k^{\mathrm{step}} = \begin{cases} \frac{1}{k-1}\sum_{j=1}^{k-1}\left\|\mathcal{P}_{j+1}^{(k)} - \mathcal{P}_j^{(k)}\right\|_2, & k > 1, \\ 0, & k = 1. \end{cases}$$

Together, these features describe the candidate horizon length, the magnitude of its plan representation, the consistency of plan tokens within the segment, and the day-to-day variation of the plan.

Historical trajectory statistics. We also extract lightweight trajectory statistics from the current historical window to describe recent mobility patterns. Let $l_{1:T_{\mathrm{valid}}}$ denote the historical location sequence after removing padding tokens, where T_{valid} is the number of valid historical slots. We use $\mathrm{Unique}(\cdot)$ to denote the set of distinct location identifiers in a sequence. The historical location diversity is computed as

$$u_{\mathrm{hist}} = \frac{|\mathrm{Unique}(l_{1:T_{\mathrm{valid}}})|}{T_{\mathrm{valid}}}.$$

The transition rate is computed as

$$\rho_{\mathrm{trans}} = \frac{1}{T_{\mathrm{valid}} - 1}\sum_{i=2}^{T_{\mathrm{valid}}} \mathbb{1}(l_i \neq l_{i-1}),$$

where $\mathbb{1}(\cdot)$ is the indicator function. To capture stay behavior, let $\{\ell_1, \ldots, \ell_G\}$ be the run lengths of consecutive identical locations in the historical sequence. The normalized mean stay duration is

$$\mu_{\mathrm{stay}} = \frac{1}{H} \cdot \frac{1}{G}\sum_{g=1}^{G} \ell_g.$$

For each historical day m, we compute the daily location diversity:

$$u_m = \frac{|\mathrm{Unique}(l_{(m-1)H+1:mH})|}{H},$$

and define the cross-day variability as

$$\sigma_{\mathrm{day}} = \mathrm{Std}(u_1, \ldots, u_M),$$

where $\mathrm{Std}(\cdot)$ denotes the standard deviation. These statistics capture whether the recent trajectory is repetitive, transition-heavy, or variable across days, providing trajectory-level context for horizon scoring.

Critic feature extractor and scoring network. The critic uses a candidate feature extractor $\Phi(\cdot)$ followed by an MLP horizon scorer. For each candidate horizon k, the feature extractor constructs

$$\phi_k = \Phi(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k).$$

In the hybrid setting, this feature vector is

$$\phi_k = [r_k,\, n_k,\, d_k^{\mathrm{tok}},\, d_k^{\mathrm{step}},\, u_{\mathrm{hist}},\, \rho_{\mathrm{trans}},\, \mu_{\mathrm{stay}},\, \sigma_{\mathrm{day}}].$$

Here, r_k encodes the normalized horizon length; n_k, d_k^{tok}, and d_k^{step} are plan-level statistics derived from $\mathcal{P}^{(k)}$; and u_{hist}, ρ_{trans}, μ_{stay}, and σ_{day} are trajectory-level statistics derived from the current historical window $\mathcal{X}_{1:T}$.

The critic score is

$$s_k = C_\psi(\phi_k),$$

where C_ψ is a three-layer MLP:

$$\begin{aligned} h_1 &= \mathrm{ReLU}(W_1\phi_k + b_1), \\ h_2 &= \mathrm{ReLU}(W_2\,\mathrm{Dropout}(h_1) + b_2), \\ s_k &= W_3\,\mathrm{Dropout}(h_2) + b_3. \end{aligned}$$

The critic is trained as a supervised horizon selector using the segment-level oracle defined in Appendix C.4.

C.3 Plan-conditioned Diffusion Actor Details

Plan-to-slot alignment. For a candidate horizon k, the masked plan condition $\mathcal{P}^{(k)} \in \mathbb{R}^{K_{\text{seg}} \times d}$ contains one daily plan token for each future day in the planning canvas, where d is the LLM hidden dimension. To condition slot-level prediction, we expand daily plan tokens to future time slots by assigning each daily plan token to the H slots of the corresponding future day. For slot $i \in \{1, \dots, K_{\text{seg}}H\}$, let

$$j(i) = \left\lceil \frac{i}{H} \right\rceil$$

denote its future-day index. The aligned plan sequence is defined as

$$\tilde{\mathcal{P}}_i^{(k)} = \mathcal{P}_{j(i)}^{(k)}, \qquad i = 1, \dots, K_{\text{seg}}H.$$

Equivalently, $\tilde{\mathcal{P}}_{1:K_{\text{seg}}H}^{(k)}$ is obtained by repeating each daily plan token for H consecutive slots. Thus, all slots within the same future day share the same high-level plan token. Since $\mathcal{P}^{(k)}$ has already zeroed out plan tokens beyond horizon k, slots beyond the candidate horizon receive zero plan conditioning.

Plan-temporal fusion. Let $Q^{(K_{\text{seg}})} = \{q_1, \dots, q_{K_{\text{seg}}H}\}$ denote the future temporal query sequence constructed from known future temporal attributes. We first inject aligned plan information into the future temporal queries by normalized addition:

$$\bar{q}_i^{(k)} = \text{LN}_q(q_i) + \text{LN}_p(\tilde{\mathcal{P}}_i^{(k)}), \qquad i = 1, \dots, K_{\text{seg}}H,$$

where $\text{LN}_q(\cdot)$ and $\text{LN}_p(\cdot)$ are layer normalization modules applied to temporal queries and aligned plan tokens, respectively. Let

$$\bar{Q}_{1:K_{\text{seg}}H}^{(k)} = \{\bar{q}_1^{(k)}, \dots, \bar{q}_{K_{\text{seg}}H}^{(k)}\}.$$

The slot-level queries further attend to the daily plan tokens through multi-head cross-attention:

$$H_{1:K_{\text{seg}}H}^{\text{actor},(k)} = \text{MHA}_{\text{actor}}\left(Q = \text{LN}_{\text{qry}}(\bar{Q}_{1:K_{\text{seg}}H}^{(k)}), K = \text{LN}_{\text{key}}(\mathcal{P}^{(k)}), V = \text{LN}_{\text{val}}(\mathcal{P}^{(k)})\right),$$

where $\text{MHA}_{\text{actor}}(\cdot)$ denotes the actor-side multi-head attention module. We then apply a Transformer-style residual update:

$$C'^{(k)}_{1:K_{\text{seg}}H} = \text{LN}_{\text{attn}}\left(\bar{Q}_{1:K_{\text{seg}}H}^{(k)} + H_{1:K_{\text{seg}}H}^{\text{actor},(k)}\right),$$

$$C_{1:K_{\text{seg}}H}^{(k)} = C'^{(k)}_{1:K_{\text{seg}}H} + \text{FFN}_{\text{actor}}(C'^{(k)}_{1:K_{\text{seg}}H}),$$

where $\text{FFN}_{\text{actor}}$ is a feed-forward network applied to each slot representation. The resulting $C_{1:K_{\text{seg}}H}^{(k)}$ is the plan-temporal conditioning sequence used by the diffusion denoiser.

LLM hidden targets. The historical day-level tokens and plan-temporal future queries are passed through the frozen LLM backbone. Let

$$U_{1:K_{\text{seg}}H}^{\text{LLM}} \in \mathbb{R}^{K_{\text{seg}}H \times d}$$

denote the resulting future hidden states corresponding to the $K_{\text{seg}}H$ future slots. These hidden states are detached from the LLM computation graph and used as teacher hidden targets for the diffusion actor. Therefore, the diffusion actor learns to generate representations in the LLM hidden space while the pretrained LLM backbone remains frozen.

Diffusion conditioning. The diffusion denoiser uses both a segment-level plan summary and the slot-level plan-temporal conditioning sequence. The segment-level plan summary is computed by averaging the nonzero plan tokens:

$$\bar{\mathcal{P}}^{(k)} = \frac{1}{k}\sum_{j=1}^{k} \mathcal{P}_j^{(k)} = \frac{1}{k}\sum_{j=1}^{k} P_j.$$

For each future slot, this plan summary is expanded to slot level and combined with the corresponding plan-temporal condition:

$$g_i^{(k)} = \mathrm{Proj}_{\mathrm{cond}}\left([\bar{\mathcal{P}}^{(k)}; C_i^{(k)}]\right), \qquad i = 1, \ldots, K_{\mathrm{seg}}H,$$

where $[\cdot;\cdot]$ denotes concatenation and $\mathrm{Proj}_{\mathrm{cond}}(\cdot)$ is a learnable projection network to the LLM hidden dimension. The resulting sequence

$$G_{1:K_{\mathrm{seg}}H}^{(k)} = \{g_1^{(k)}, \ldots, g_{K_{\mathrm{seg}}H}^{(k)}\}$$

is used as the final conditioning sequence for the diffusion denoiser.

Gaussian hidden-space diffusion. We apply Gaussian diffusion to the LLM hidden targets. At diffusion step τ, the clean hidden target $U_{1:K_{\mathrm{seg}}H}^{\mathrm{LLM}}$ is perturbed as

$$U_\tau^{(k)} = \sqrt{\bar{\alpha}_\tau} U_{1:K_{\mathrm{seg}}H}^{\mathrm{LLM}} + \sqrt{1-\bar{\alpha}_\tau}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where ϵ is standard Gaussian noise and

$$\bar{\alpha}_\tau = \prod_{r=1}^{\tau}(1-\beta_r).$$

We use a linear variance schedule β_τ from $\beta_{\min} = 10^{-4}$ to $\beta_{\max} = 0.02$. The denoiser predicts the injected noise:

$$\hat{\epsilon}^{(k)} = \epsilon_\theta\left(U_\tau^{(k)}, \tau, G_{1:K_{\mathrm{seg}}H}^{(k)}\right),$$

where $\epsilon_\theta(\cdot)$ is a per-token MLP denoiser conditioned on the noisy hidden state, the diffusion step, and the conditioning sequence.

In implementation, the diffusion step τ is encoded by a sinusoidal timestep embedding $\mathrm{TimeEmb}(\tau)$, which is projected to the hidden dimension by MLP_τ. The conditioning vector $g_i^{(k)}$ is also projected by MLP_c. For each future slot i, the denoiser input is

$$h_i^{(k)} = U_{\tau,i}^{(k)} + \mathrm{MLP}_\tau(\mathrm{TimeEmb}(\tau)) + \mathrm{MLP}_c(g_i^{(k)}),$$

where MLP_τ and MLP_c are learnable projection networks. The predicted noise is

$$\hat{\epsilon}_i^{(k)} = \mathrm{MLP}_{\mathrm{denoise}}\left(\mathrm{LN}_{\mathrm{denoise}}(h_i^{(k)})\right),$$

where $\mathrm{LN}_{\mathrm{denoise}}$ is layer normalization and $\mathrm{MLP}_{\mathrm{denoise}}$ is the final per-token denoising network. The denoised hidden representation is recovered by

$$\hat{U}_0^{(k)} = \frac{U_\tau^{(k)} - \sqrt{1-\bar{\alpha}_\tau}\hat{\epsilon}^{(k)}}{\sqrt{\bar{\alpha}_\tau} + \varepsilon},$$

where ε is a small numerical constant for stability. The denoised hidden states are projected to location logits:

$$O_i^{(k)} = W_o \hat{U}_{0,i}^{(k)} + b_o,$$

where W_o and b_o are trainable output projection parameters mapping hidden representations to the location set.

Progressive noise schedule. To stabilize hidden-space diffusion training, we progressively increase the maximum diffusion timestep used during training. Early in training, the actor denoises nearly clean hidden states; later, it is exposed to stronger noise. Let n denote the global training step and T denote the total number of diffusion steps. We sample

$$\tau \sim \mathrm{Uniform}\{0, \ldots, T_n - 1\},$$

where T_n is the maximum timestep available at training step n:

$$T_n = \begin{cases} 1, & n < 1000, \\ 10, & 1000 \le n < 2000, \\ 50, & 2000 \le n < 4000, \\ 100, & 4000 \le n < 8000, \\ 200, & 8000 \le n < 16000, \\ \min(300, T), & n \ge 16000. \end{cases}$$

This curriculum gradually increases denoising difficulty and improves stability during early training.

Valid-horizon masking. For a candidate or executed horizon k, only the first kH positions are valid. We define a valid-horizon mask $m^{(k)} \in \{0,1\}^{K_{\text{seg}}H}$ as

$$m_i^{(k)} = \begin{cases} 1, & i \le kH, \\ 0, & i > kH. \end{cases}$$

Both the classification and diffusion losses are weighted by this mask, so positions beyond kH do not contribute to the candidate loss or the emitted segment.

Inference. At inference, we fix $t = 0$, sample the noise once, and apply the denoiser once to refine the future hidden representations produced by the frozen LLM. No iterative reverse diffusion chain is used.

C.4 Learning Objectives

Frozen and trainable components. The day-level historical tokens and future temporal queries are processed by a pretrained LLM backbone for contextual sequence modeling. The LLM parameters are kept frozen throughout learning. All mobility-specific components, including the embedding projections, day-level aggregation, planner, critic, diffusion actor, and output classifier, remain trainable and are updated through the learning objectives defined below.

Segment-level critic supervision. The critic is trained to approximate the best execution horizon using ground-truth future locations available during training. For each candidate horizon k, we apply the corresponding masked plan condition $\mathcal{P}^{(k)}$, run the actor on the fixed K_{seg}-day canvas, and obtain location logits $O^{(k)}_{1:K_{\text{seg}}H}$. We then evaluate the candidate by computing the average classification loss over the first kH valid positions:

$$\mathcal{L}_{\text{CE}}^{(k)} = \frac{1}{kH} \sum_{i=1}^{kH} \text{CE}(O_i^{(k)}, y_i),$$

where y_i is the ground-truth location label at future slot i, and $\text{CE}(\cdot,\cdot)$ denotes the cross-entropy loss between location logits and the target location label. The candidate with the lowest segment-level classification loss is treated as the oracle execution horizon:

$$k^{\text{sup}} = \arg\min_k \mathcal{L}_{\text{CE}}^{(k)}.$$

Thus, k^{sup} provides a supervised target for horizon selection. Given critic scores $\{s_k\}$ over candidate horizons, the critic is trained to assign the highest probability to this oracle horizon:

$$\mathcal{L}_{\text{critic}} = -\log \frac{\exp(s_{k^{\text{sup}}})}{\sum_k \exp(s_k)}.$$

This loss teaches the critic to select horizons that are expected to yield lower segment-level prediction error.

Masked actor objectives. Given an executed horizon k, the actor produces logits and hidden-space diffusion outputs on the fixed K_{seg}-day canvas. However, only the first kH positions correspond to the executed segment. We therefore use the valid-horizon mask $m^{(k)}$ defined above. The location prediction loss is computed only over valid positions:

$$\mathcal{L}_{\text{CE}} = \frac{1}{\sum_i m_i^{(k)}} \sum_{i=1}^{K_{\text{seg}}H} m_i^{(k)} \text{CE}(O_i^{(k)}, y_i).$$

This objective trains the output classifier to predict future location identifiers for the executed segment, while ignoring canvas positions beyond the selected horizon.

The same valid-horizon mask is applied to the hidden-space diffusion objective. Let $U_{\tau,i}^{(k)}$ denote the noisy LLM hidden state at diffusion step τ, ϵ_i the Gaussian noise injected into slot i, and $g_i^{(k)}$ the final diffusion conditioning vector defined in Appendix C.3. The diffusion loss is

$$\mathcal{L}_{\text{diff}} = \frac{1}{\sum_i m_i^{(k)}} \sum_{i=1}^{K_{\text{seg}}H} m_i^{(k)} \left\| \epsilon_i - \epsilon_\theta(U_{\tau,i}^{(k)}, \tau, g_i^{(k)}) \right\|_2^2.$$

This trains the diffusion actor to denoise LLM hidden representations only within the executed segment.

The final objective combines location prediction, hidden-space diffusion, and critic supervision:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{diff}}\mathcal{L}_{\mathrm{diff}} + \lambda_{\mathrm{critic}}\mathcal{L}_{\mathrm{critic}}.$$

Here, λ_{diff} and $\lambda_{\mathrm{critic}}$ control the contribution of the diffusion and critic losses.

C.5 Overall Pipeline Pseudocode

Algorithm 1 Training with Segment-level Horizon Supervision

Require: Historical window $\mathcal{X}_{1:T}$; future timestamps $\mathcal{T}^{(K_{\mathrm{seg}})}$; future labels $\mathcal{Y}^{(K_{\mathrm{seg}})}$

Encode historical records: $E_{1:M} = \mathrm{DayAgg}(\mathrm{Embed}(\mathcal{X}_{1:T}))$
Encode future temporal queries on the fixed canvas: $Q^{(K_{\mathrm{seg}})} = \mathrm{TimeEnc}(\mathcal{T}^{(K_{\mathrm{seg}})})$
Generate unmasked future daily plan tokens: $P_{1:K_{\mathrm{seg}}} = \mathrm{Planner}(E_{1:M})$
for $k = 1, \ldots, K_{\mathrm{seg}}$ **do**
    Construct candidate-specific plan condition: $\mathcal{P}^{(k)} = \mathrm{Mask}(P_{1:K_{\mathrm{seg}}}, k)$
    Extract candidate feature: $\phi_k = \Phi(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k)$
    Compute critic score: $s_k = C_\psi(\phi_k)$
    Build plan-temporal conditioning sequence: $C^{(k)} = \mathrm{Fuse}(Q^{(K_{\mathrm{seg}})}, \mathcal{P}^{(k)})$
    Build final diffusion conditioning sequence: $G^{(k)} = \mathrm{Proj}_{\mathrm{cond}}([\bar{\mathcal{P}}^{(k)}; C^{(k)}])$
    Run the diffusion actor on the fixed K_{seg}-day canvas
    Compute segment loss $\mathcal{L}_{\mathrm{CE}}^{(k)}$ over the first kH positions
end for
Set oracle horizon: $k^{\mathrm{sup}} = \arg\min_k \mathcal{L}_{\mathrm{CE}}^{(k)}$
Train critic with $\mathcal{L}_{\mathrm{critic}}$ to score k^{sup} highest
Train actor with valid-horizon mask $m^{(k^{\mathrm{sup}})}$
Update trainable mobility-specific modules; keep LLM backbone frozen

Algorithm 2 Segment-level Rollout Inference

Require: Historical window $\mathcal{X}_{1:T}$; target horizon K; maximum segment length K_{seg}

Set remaining horizon $R \leftarrow K$
while $R > 0$ **do**
    Encode current window: $E_{1:M} = \mathrm{DayAgg}(\mathrm{Embed}(\mathcal{X}_{1:T}))$
    Encode future temporal queries on the fixed canvas: $Q^{(K_{\mathrm{seg}})} = \mathrm{TimeEnc}(\mathcal{T}^{(K_{\mathrm{seg}})})$
    Generate unmasked future daily plan tokens: $P_{1:K_{\mathrm{seg}}} = \mathrm{Planner}(E_{1:M})$
    for $k = 1, \ldots, \min(K_{\mathrm{seg}}, R)$ **do**
        Construct candidate-specific plan condition: $\mathcal{P}^{(k)} = \mathrm{Mask}(P_{1:K_{\mathrm{seg}}}, k)$
        Extract candidate feature: $\phi_k = \Phi(\mathcal{X}_{1:T}, \mathcal{P}^{(k)}, k)$
        Compute critic score: $s_k = C_\psi(\phi_k)$
    end for
    Select execution horizon: $k^\star = \arg\max_k s_k$
    Build plan-temporal conditioning sequence: $C^{(k^\star)} = \mathrm{Fuse}(Q^{(K_{\mathrm{seg}})}, \mathcal{P}^{(k^\star)})$
    Build final diffusion conditioning sequence: $G^{(k^\star)} = \mathrm{Proj}_{\mathrm{cond}}([\bar{\mathcal{P}}^{(k^\star)}; C^{(k^\star)}])$
    Predict the $k^\star$-day segment with the diffusion actor
    Convert logits $O_{1:k^\star H}^{(k^\star)}$ to predicted locations
    Recompute spatial coordinates from predicted location identifiers
    Advance the historical window by $k^\star H$ slots using predicted records
    Update remaining horizon $R \leftarrow R - k^\star$
end while
return Predicted K-day mobility sequence

D EXPERIMENT SETTINGS

D.1 DATASET DETAILS

We use three city-level subsets from YJMob100K: Kumamoto, Sapporo, and Hiroshima. Each dataset contains anonymized user mobility trajectories over 75 days. The spatial resolution is $500\mathrm{m} \times 500\mathrm{m}$, and the location set contains 40k discrete places. Each day is discretized into 48 time slots, corresponding to 30-minute intervals. The three datasets share the same duration, spatial resolution, and location set size, but differ in the number of users. We report the time and storage costs of preprocessing and semantic embedding generation in Table 4.

Table 4: Preprocessing cost across datasets.

Dataset	Users	Preprocessing Time (h)	Storage Size (GB)
Kumamoto	3k	3.1	1.6
Sapporo	1k	1.3	0.8
Hiroshima	1k	1.3	0.8

Chronological split. We split each city dataset chronologically by prediction target days into training, validation, and test periods using a 70%/20%/10% ratio. The historical window for each sample is taken from the days preceding its prediction target, while the split assignment is determined by the target days. This avoids label leakage: validation and test target locations are never used for model training. For the 7-to-7 long-horizon setting, where the input consists of 7 historical days and the target consists of the following 7 days, a test sample may use days 61–67 as history and days 68–74 as prediction targets. Boundary samples without a complete history or target horizon are excluded.

D.2 EVALUATION METRICS

We evaluate both slot-level location prediction and trajectory-level similarity. Let N denote the total number of evaluated future slots across all test samples. For slot i, let y_i be the ground-truth location and $\hat{\mathbf{p}}_i$ be the predicted probability distribution over the location set $\mathcal{L}$. In a K-day forecasting task, each sample contributes KH future slots.

Accuracy@q. Accuracy@q measures whether the ground-truth location appears among the top-q predicted candidates:

$$\mathrm{Acc@}q = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(y_i \in \text{Top-}q(\hat{\mathbf{p}}_i)\right),$$

where Top-$q(\hat{\mathbf{p}}_i)$ denotes the set of q locations with the highest predicted probabilities at slot i, and $\mathbf{1}(\cdot)$ is the indicator function. Higher Accuracy@q indicates better prediction performance.

Mean Reciprocal Rank. Mean Reciprocal Rank (MRR) evaluates the rank of the ground-truth location in the predicted distribution:

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\mathrm{rank}_i(y_i)}.$$

Here, $\mathrm{rank}_i(y_i)$ is the position of y_i after sorting all candidate locations by predicted probability in descending order. A rank of 1 indicates that the ground-truth location is assigned the highest probability. Higher MRR indicates better ranking quality.

Dynamic Time Warping. Dynamic Time Warping (DTW) measures spatial alignment between the predicted and ground-truth trajectories:

$$\mathrm{DTW}(\mathcal{Y}, \hat{\mathcal{Y}}) = \min_{\pi} \sum_{(i,j)\in\pi} d(y_i, \hat{y}_j),$$

where π is an alignment path between the two sequences that maintains chronological ordering, and $d(y_i, \hat{y}_j)$ is the Euclidean distance between the coordinates of locations y_i and $\hat{y}_j$. Lower DTW indicates better spatial alignment.

BLEU. BLEU measures sequence-level overlap between predicted and ground-truth location sequences:

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N_g} w_n \log p_n\right),$$

where p_n is the modified precision of n-grams, w_n is the weight for the n-gram order, N_g is the maximum n-gram order, and BP is the brevity penalty. Higher BLEU indicates stronger sequence-level agreement.

D.3 Experiment Details

Computational resources. All experiments are implemented in PyTorch with the Hugging Face Transformers library[1]. Offline prompt-derived semantic embeddings are computed once during preprocessing and reused during training and rollout. Experiments are run on a single NVIDIA Tesla V100-DGXS-32GB GPU from a DGX Station with four such GPUs, an Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, 40 logical CPUs, and 251GB system memory. The NVIDIA driver version is 570.133.07, and the CUDA version reported by `nvidia-smi` is 12.8. We use Python 3.8.10 and Transformers 4.46.3.

Hyperparameters. Hyperparameters are selected based on validation performance. Unless otherwise specified, we use a 7-day historical window, 48 slots per day, and a maximum segment length of 3 days. The model is trained with AdamW. We search the learning rate over $\{1 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$ and weight decay over $\{0, 10^{-3}, 10^{-2}\}$. Batch size, dropout rate, diffusion loss weight, critic loss weight, and the number of training epochs are selected on the validation set. All reported three-run results use seeds $\{42, 3407, 0\}$, and the checkpoint with the highest validation Acc@1 is used for evaluation. Detailed command-line configurations and hyperparameter settings are included in the released code.

D.4 Model Variant Details

We evaluate several PlanCAD variants to analyze the effects of critic features, horizon selection strategies, diffusion design, and frozen LLM backbone choices. In particular, Critic-Traj is a deliberately restricted trajectory-only control that removes plan-derived features while retaining the same observed trajectory statistics across candidate horizons, providing a reference for assessing the contribution of plan-dependent critic features. Tables 6–7 summarize the variant definitions.

[1] `https://huggingface.co/docs/transformers`

Table 5: Horizon selection and diffusion variants.

Variant	Description
FIXED-1	Always executes a one-day segment before updating the historical window.
FIXED-3	Always executes a three-day segment when the remaining horizon allows.
ORACLE	Uses the same trained PlanCAD-Full checkpoint but replaces the critic at test time with an oracle that selects the feasible horizon k minimizing the ground-truth local segment cross-entropy.
NO-PLAN	Removes plan conditioning while retaining the LLM-based actor and diffusion refinement; the actor relies only on historical context and future temporal queries.
NO-DIFF	Removes the diffusion actor; the model instead feeds historical context and future temporal queries into the LLM and classifies the resulting future hidden states.
MATCHED-NO-DIFF	Replaces diffusion refinement with a deterministic residual decoder using the same denoiser architecture, conditioning, and parameter count, without noise injection, x_0 reconstruction, or diffusion loss.
RANDOM-NOISE-DIFF	Replaces the learned hidden-space diffusion refinement with a naive random-noise variant.

Table 6: Critic feature variants. PlanCAD-Full uses the hybrid critic.

Variant	Description
CRITIC-PLAN	Uses only plan-level statistics extracted from candidate daily plan tokens.
CRITIC-TRAJ	Uses only trajectory-level statistics extracted from the historical mobility window.
CRITIC-HORIZON	Uses only the normalized candidate horizon length as the critic input.
CRITIC-HYBRID	**Used in PlanCAD-Full.** Combines plan-level statistics, trajectory-level context features, and the normalized candidate horizon length for horizon scoring.

Table 7: Frozen LLM backbone variants used in PlanCAD.

Variant	Backbone
PLANCAD-LLAMA-1B (PlanCAD-Full)	Llama-3.2-1B
PLANCAD-LLAMA-3B	Llama-3.2-3B
PLANCAD-QWEN-1.5B	Qwen2.5-1.5B
PLANCAD-GEMMA-2B	Gemma-2-2B
PLANCAD-DEEPSEEK-1.5B	DeepSeek-R1-Distill-Qwen-1.5B

D.5 EXECUTION-MODE AND LEAD-DAY ANALYSIS

We further examine whether PlanCAD's long-horizon gains mainly arise from avoiding error accumulation in autoregressive rollout. Following the main ablation setting, we conduct this diagnostic analysis on Kumamoto and examine two representative baselines: CMHSA as a strong accuracy baseline and Mobility-LLM as an LLM-based mobility model. For direct multi-day prediction, each baseline predicts the full future segment from the same observed history without autoregressive feedback. The autoregressive baseline and PlanCAD-Full results are taken from the main experiments.

Table 8: Acc@1 and Acc@3 comparison of autoregressive and direct multi-day prediction. The **best** and second-best results are highlighted.

	7-to-3		7-to-7	
Model / Execution Mode	Acc@1	Acc@3	Acc@1	Acc@3
CMHSA (Autoregressive)	**28.56**	50.41	27.91	50.09
CMHSA (Direct Multi-day)	27.16	49.40	27.07	50.41
Mobility-LLM (Autoregressive)	20.41	37.93	16.03	30.14
Mobility-LLM (Direct Multi-day)	20.07	41.00	23.80	45.80
PlanCAD-Full	28.02	**50.43**	**28.50**	**51.10**

Direct multi-day prediction does not consistently improve the baselines. For CMHSA, direct prediction reduces Acc@1 under both forecasting settings, while Acc@3 decreases under 7-to-3 and improves only slightly under 7-to-7. Mobility-LLM benefits more from direct prediction, particularly under 7-to-7, where both Acc@1 and Acc@3 improve substantially. Nevertheless, PlanCAD-Full achieves stronger Acc@1 and Acc@3 than the direct multi-day baselines in both settings, suggesting that its long-horizon gains cannot be explained solely by avoiding autoregressive error accumulation.

Lead-Day Analysis. We further examine whether the aggregate long-horizon improvement is concentrated on particular forecast days. We compare PlanCAD-Full with CMHSA under both autoregressive and direct multi-day execution and report Acc@1 and Acc@3 separately for each forecast lead day in Table 9.

Table 9: Per-lead-day Acc@1 and Acc@3 comparison of PlanCAD-Full with CMHSA under autoregressive and direct multi-day execution. The **best** and second-best results are highlighted.

	CMHSA (Autoregressive)		CMHSA (Direct Multi-day)		PlanCAD-Full	
Lead Day	Acc@1	Acc@3	Acc@1	Acc@3	Acc@1	Acc@3
			7-to-3			
1	**28.61**	**52.43**	27.56	50.76	27.68	50.49
2	**26.10**	**46.39**	24.68	44.03	26.02	45.83
3	30.79	**56.38**	29.33	53.47	**31.30**	56.28
			7-to-7			
1	**28.87**	**50.34**	26.79	49.74	27.20	49.99
2	25.24	44.27	24.06	44.35	**26.10**	**46.45**
3	30.05	54.52	28.55	52.65	**31.74**	**55.53**
4	**29.11**	50.33	27.50	**51.59**	28.58	51.47
5	28.20	50.75	27.56	50.58	**28.25**	**51.23**
6	27.02	49.18	26.87	**51.91**	**27.24**	50.39
7	28.41	51.61	28.13	51.75	**28.97**	**52.59**

PlanCAD-Full remains competitive with autoregressive CMHSA across lead days and consistently outperforms direct multi-day CMHSA in Acc@1. Under 7-to-7, PlanCAD-Full achieves the best Acc@1 on five of seven lead days and the best Acc@3 on four days, including both metrics on Days 2, 3, 5, and 7. Under 7-to-3, autoregressive CMHSA remains stronger on most lead-day metrics, while PlanCAD-Full achieves the best Acc@1 on Day 3. These results suggest that PlanCAD's gains become more evident over the longer forecasting horizon and are sustained across multiple forecast days rather than being driven by a small subset of the prediction horizon.

D.6 Detailed Geographical Evaluation

We provide the complete geographical evaluation results across Kumamoto, Sapporo, and Hiroshima in Table 10. In addition to slot-level accuracy reported in the main text, we evaluate ranking quality

Table 10: Geographical evaluation on three datasets. We report MRR, BLEU, and DTW under 7-to-3 and 7-to-7 forecasting settings. Higher is better for MRR and BLEU; lower is better for DTW. The **best** and second-best results are highlighted.

7-to-3

	Kumamoto			Sapporo			Hiroshima		
Model	MRR	BLEU	DTW	MRR	BLEU	DTW	MRR	BLEU	DTW
LSTM	0.3540	13.69	5399.4	0.3253	16.06	4769.5	0.3344	15.93	6060.4
DeepMove	0.3925	**16.32**	6165.4	0.3810	17.63	4100.7	0.3587	15.96	5553.5
PatchTST	0.3938	14.53	5254.0	**0.4139**	18.00	**3917.9**	0.4137	17.76	5102.9
iTransformer	0.3812	8.49	5731.0	0.3965	8.25	5524.0	0.3827	8.04	5727.0
ST-MoE-BERT	0.3518	6.97	6600.0	0.3930	16.60	4040.3	0.3870	15.29	5412.0
PMT	0.4032	11.00	4922.0	0.3971	16.29	4016.9	0.3973	15.53	5223.2
COLA	0.3972	12.19	5332.0	0.4090	**21.17**	3968.5	0.4127	**20.78**	**5030.4**
CMHSA	**0.4144**	14.06	**4791.0**	0.4021	19.47	3967.5	0.4039	19.07	5096.2
Mobility-LLM	0.3097	12.06	14727.0	0.3047	15.02	7810.0	0.2983	13.98	7101.0
PlanCAD-Full	0.4042	13.99	5667.4	0.4074	12.85	5543.4	**0.4153**	13.33	5405.2

7-to-7

	Kumamoto			Sapporo			Hiroshima		
Model	MRR	BLEU	DTW	MRR	BLEU	DTW	MRR	BLEU	DTW
LSTM	0.3219	12.74	6371.0	0.2864	15.21	5419.1	0.2964	14.59	7286.9
DeepMove	0.3918	16.46	**4705.3**	0.3542	15.85	4571.9	0.3322	14.15	6369.5
PatchTST	0.4066	15.44	4838.0	0.4032	16.11	4146.6	0.3987	16.11	6400.7
iTransformer	0.3856	8.28	5466.0	0.3921	7.65	5231.0	0.3782	7.49	5768.0
ST-MoE-BERT	0.3386	6.99	6555.0	0.3226	8.27	7047.0	0.3791	13.83	**4364.7**
PMT	0.4118	11.94	4815.0	0.3821	13.92	4351.7	0.3847	13.40	5643.6
COLA	0.4023	11.68	5051.0	0.4028	**19.02**	**4112.6**	0.4092	**19.03**	5200.7
CMHSA	**0.4120**	13.21	4767.0	0.3909	16.18	4223.7	0.3962	16.03	5410.5
Mobility-LLM	0.2497	13.09	19448.0	0.3493	18.09	8672.2	0.3128	15.03	14844.0
PlanCAD-Full	0.4115	**16.73**	5479.9	**0.4114**	14.81	5248.5	**0.4181**	15.16	4727.6

and trajectory-level geographical consistency using MRR, BLEU, and DTW under both 7-to-3 and 7-to-7 forecasting settings.

Overall, the detailed results are consistent with the observations in the main text. PlanCAD-Full achieves particularly strong MRR across cities, including the best MRR on Hiroshima under both settings and on Sapporo under 7-to-7, while also achieving the best BLEU on Kumamoto under 7-to-7. Although the strongest method varies across geographical metrics and datasets, PlanCAD-Full maintains competitive ranking and trajectory-level performance across both forecasting horizons.

D.7 Additional Analysis of Diffusion Refinement

We provide additional analyses of the diffusion actor to examine its behavior under different diffusion timesteps and perturbations of the future hidden representation. These experiments focus on the role of diffusion as a hidden-space refinement mechanism.

Diffusion Timestep Sensitivity. We first vary the diffusion timestep t, which controls the corruption level of the future hidden representation, while keeping the remaining model configuration fixed. Table 11 shows that the forecasting performance remains relatively stable under small to moderate diffusion timesteps in both forecasting settings. As t becomes large, Acc@1, Acc@3, and MRR gradually decrease, consistent with the increasing difficulty of recovering more strongly perturbed hidden representations. This behavior indicates that the learned diffusion actor can effectively refine representations over a range of moderate corruption levels.

Sensitivity to Latent Perturbations. We further examine the response of the actor to perturbations applied directly to the future hidden representation. Before branching into the diffusion and parameter-matched deterministic decoders, we perturb the same future hidden representation h as $h_{\text{noisy}} =$

Table 11: Sensitivity to the diffusion timestep t under 7-to-3 and 7-to-7 forecasting. Each row evaluates the same trained checkpoint at a fixed diffusion timestep; no iterative reverse chain is performed.

	7-to-3					7-to-7				
t	Acc@1	Acc@3	MRR	BLEU	DTW	Acc@1	Acc@3	MRR	BLEU	DTW
1	28.23	50.71	0.4057	0.1361	5522.0	28.23	51.04	0.4093	0.1643	5486.2
10	28.18	50.71	0.4056	0.1355	5522.7	28.19	51.02	0.4091	0.1647	5482.8
50	28.25	50.71	0.4060	0.1372	5533.0	28.16	50.95	0.4089	0.1647	5495.1
100	28.23	50.62	0.4057	0.1394	5518.0	28.09	50.97	0.4084	0.1668	5479.5
200	28.03	50.47	0.4040	0.1460	5559.4	27.92	50.89	0.4069	0.1729	5536.8
300	27.67	50.07	0.4011	0.1563	5694.7	27.55	50.49	0.4035	0.1819	5628.0

Table 12: Acc@1 and Acc@3 sensitivity to perturbations of the future hidden representation under the 7-to-3 setting. Both actor variants receive the same perturbed hidden representation.

	Diffusion		Matched No-Diff	
σ	Acc@1	Acc@3	Acc@1	Acc@3
0.0	28.02	50.43	28.27	50.87
0.5	27.90	49.94	27.72	50.19
1.0	26.38	48.28	26.25	48.85
1.5	23.76	44.28	23.44	44.21
2.0	18.98	36.44	18.60	35.46

$h + \sigma\,\mathrm{Std}(h)\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. Thus, both decoders receive the same perturbed representation, enabling a controlled comparison. Table 12 reports 7-to-3 Acc@1 and Acc@3 across representative perturbation levels. Both variants remain comparable under mild to moderate perturbations. As the perturbation becomes stronger, the diffusion actor shows slightly smaller performance degradation, achieving higher Acc@1 and Acc@3 at $\sigma = 1.5$ and 2.0. This suggests a modest robustness benefit from diffusion refinement under severe latent corruption in the 7-to-3 setting.

D.8 Dispersion Analysis

For Figure 2(b), selected plan-token dispersion is grouped into five equal-frequency quantile bins, with tied boundaries merged when necessary. Markers report mean segment-level Acc@1 within each bin, with error bars showing ± 1 standard error. Normalized density curves are Gaussian kernel density estimates fitted separately for each setting with a bandwidth factor of 0.35.