LOOKING AHEAD AND BACK: ANTICIPATORY GENERATIVE MODELING FOR ROAD-NETWORK TRAJECTORIES

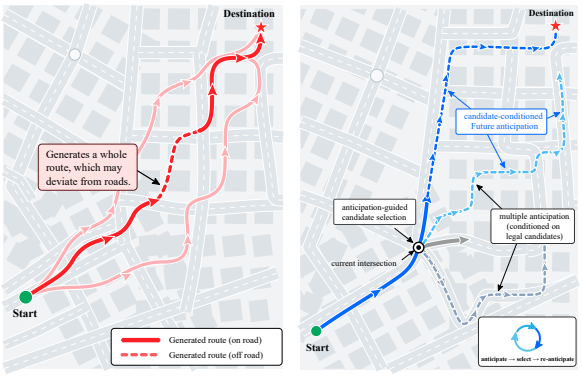
Anonymous authors
Paper under double-blind review

ABSTRACT

Trajectory generation underpins urban applications such as traffic management and urban planning. Within road networks, vehicle movement along road segments is largely deterministic, whereas transitions at intersections are critical—multiple outgoing paths may appear equally plausible, yet they can lead to substantially different downstream routes. Existing generative methods such as diffusion models simulate location sequences and fail short of next transition selection, even producing trajectories that drift off the road network. To fill this gap, we advocate a fundamental shift: moving beyond location simulation toward behavior modeling on the road network. In this paper, we revisit trajectory generation through the lens of generative route planning, and introduce a new graph-constrained generative paradigm, Anticipatory Generative Modeling, which resolves next transition selection by incorporating the *Anticipation* of how the future route may unfold toward the destination. Concretely, we propose a novel generative model named FORESIGHT that formulates a probabilistic model for "*anticipate–select–update–re-anticipate*" loop that enables longer-term foresight while continuing sequential generation along the historical trajectory. Extensive experiments demonstrate that FORESIGHT wins state-of-the-art baselines. Codes: `https://anonymous.4open.science/r/Foresight-67ED`.

1 INTRODUCTION

Generating human mobility data, particularly vehicle movement trajectories, plays a fundamental role in a wide spectrum of urban applications, including traffic management, navigation, and urban planning (Shi et al., 2024; Li et al., 2026). Within road networks, vehicle movement is constrained by the physical topology: movement along road segments is largely deterministic, whereas *transitions at intersections are critical*—multiple outgoing paths may appear equally plausible given the traversed history, yet they can lead to substantially different downstream routes.



(a) Diffusion model: global route generation. (b) **FORESIGHT**: Anticipatory route generation.

Figure 1: Motivated example for anticipatory generative modeling.

Conditioned on a given destination, a central challenge in road-network trajectory generation lies in modeling the next transition selection, especially at road intersections. In the literature, deep generative models such as diffusion-based methods are frequently employed (Zhu et al., 2023; Wei et al., 2024; Zhu et al., 2024; Shi et al., 2024; Li et al., 2026). These models typically treat trajectories as holistic sequences of 2D locations and synthesize complete trajectories by denoising from random noise, often overlooking the discrete decision of next-transition selection at intersections. Consequently, they are prone to producing drift-off trajectories that devi-

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.

ate from the underlying road network, as illustrated in Fig. 1a, which undermines their effectiveness for capturing real-world human mobility patterns. Regarding transition selection, classical search or dynamic programming, e.g., Dijkstra and A-star, attempt to find optimal solutions by exhaustively enumerating the action space, but they struggle to scale to large-scale road networks (Dijkstra, 1959; Hart et al., 1968). Although reinforcement learning has been explored for action selection, the sparse reward signal associated with destination reachability makes it ill-conditioned for generating step-wise action sequences on road networks (Bello et al., 2017; Kool et al., 2019; Sun et al., 2023). That is, route planning for road-network trajectories remains a largely open problem.

In this paper, we revisit trajectory generation on road networks, shifting from location simulation towards behavior modeling, and introduce a new graph-constrained generative paradigm, *Anticipatory Generative Modeling*, for route planning. The core of this paradigm is **anticipation**, a provisional short-horizon continuation used to guide next-transition selection. A transition generator remains ambiguous when conditioned solely on the traversed history, since every legal transition on the road network may appear equally plausible. As illustrated in Fig. 1b, the situation changes when the model incorporates anticipation of how the future route may unfold toward the destination. Accordingly, we construct a 4-phase loop of "*anticipate–select–update–re-anticipate*" that enables longer-term foresight while continuing sequential generation along the historical trajectory.

Specifically, we present an anticipatory generation model named FORESIGHT to produce plausible trajectories on road networks. Given the destination g, we formalize *Anticipatory Generative Modeling* (AGM) at the heart of our model, which first predicts a provisional, destination-conditioned continuation over a finite horizon, and then uses its representation to select the current transition:

$$c_t, g \longrightarrow R_t = (S_{t+1}, \ldots, S_{t+H}) \longrightarrow q_t(\cdot), \tag{1}$$

where R_t is inferred from the available traversed history c_t. It thus serves as predictive context for the current selection, rather than being executed. After each transition, the history is updated and a new continuation is predicted. This loop preserves closed-loop generation while exposing downstream context. H limits the explicitly represented future window, not the total trajectory length. FORESIGHT employs a Future Decoder to predict R_t, and a topology-constrained Transition Network to score and normalize legal outgoing candidates. A direction-aware fiber state encodes the geometry of each realized movement, while candidate-conditioned re-anticipation predicts the continuation induced by each hypothetical candidate after the current distribution is formed. The training objective couples the likelihood of the current legal transition with future-window prediction and candidate-conditioned future compatibility. During inference, an optional future-consistency score can further refine candidate selection. After each executed movement, FORESIGHT recomputes the continuation from the updated history. The independent `STOP` event is admissible only at the destination, ensuring that validity and termination are governed by the same graph-constrained process. Our key contributions are summarized as follows:

- *Generative Route Planning.* We revisit trajectory generation through the lens of behavior modeling and fundamentally shift the focus from location simulation to route planning on road networks.
- *Anticipatory Generative Modeling.* Essentially different from diffusion models, we introduce a new graph-constrained generative paradigm, which resolves next transition selection by incorporating the anticipation of how the future route may unfold toward the destination.
- *Methodology Innovation.* We purpose a novel FORESIGHT model that formulates a probabilistic model with a direction-aware fiber for "*anticipate–select–re-anticipate*" loop that enables longer-term foresight while continuing sequential generation along the historical trajectory.
- *Empirical Validation.* FORESIGHT achieves state-of-the-art performance across various metrics, demonstrating its effectiveness in generating plausible trajectories.

2 RELATED WORK

Trajectory generation. Trajectory generation methods span autoregressive models that recursively predict subsequent locations or road segments and latent-variable or diffusion models that synthesize trajectories more globally. In mobility modeling, diffusion-based approaches such as DiffTraj generate complete GPS sequences (Zhu et al., 2023), while road-network methods increasingly incorporate topology into either step-wise or sequence-level generation. Autoregressive methods such as ProgRPGAN (Fu & Lee, 2021), Semantic Routing (Zhao et al., 2024), and HOSER (Cao

et al., 2025) construct routes incrementally, conditioning each local choice on the destination, map semantics, or traversed history. Graph-Constrained Diffusion, Diff-RNTraj, and ControlTraj instead generate complete paths or trajectories while explicitly encoding road-network structure (Shi et al., 2024; Wei et al., 2024; Zhu et al., 2024). These approaches therefore primarily model either the immediate next transition or the trajectory as a complete sequence.

Iterative replanning. Receding-horizon control offers a complementary closed-loop paradigm: the planner repeatedly predicts over a finite horizon, executes the first action, and replans from the resulting state. Related principles have been adopted in learned driving planners that reason over continuous trajectories, occupancy representations, or traffic-scene forecasts. (Liu et al., 2026; Zheng et al., 2025; Huang et al., 2026; Jiang et al., 2024; Wu et al., 2024). Foresight brings this interface to a discrete road graph: each graph-legal transition is informed by an explicitly predicted, goal-conditioned future window, while a position- and direction-aware fiber state preserves local context. After execution, the anticipated continuation is retained when the transition is aligned and revised when the rollout deviates, coupling trajectory generation with closed-loop replanning.

Planning with future information. Classical search methods reason over candidate futures: Dijkstra (Dijkstra, 1959) accumulates path costs, A star (Hart et al., 1968) adds heuristic estimates, and Yen's algorithm enumerates alternative loopless paths (Yen, 1971). Learned search and routing methods incorporate longer-term consequences through neural heuristics, returns, value functions, or target-conditioned policies (Yonetani et al., 2021; Jain et al., 2021; Bello et al., 2017; Kool et al., 2019; Zeng et al., 2023). Sequence-modeling and diffusion-based approaches instead represent or generate imagined trajectories (Janner et al., 2021; Chen et al., 2021; Wu et al., 2023; Janner et al., 2022). These works encode future information through search, value estimates, or trajectory prediction. Foresight explicitly represents a goal-conditioned short-horizon continuation and uses it to select the next legal graph transition.

3 Preliminaries and Problem Formulation

Road-network trajectories. We represent the road network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of road locations and $\mathcal{E}$ contains directed connections. Let $X_t \in \mathcal{V}$ denote the location at step t, and let $A_t = \psi(X_t, X_{t-1})$ denote the direction-aware attribute of the incoming movement (with an initialization token at $t = 0$). The corresponding road state is $S_t = (A_t, X_t)$, and the realized history before the next decision is $c_t = S_{0:t}$. A trajectory is conditioned on a destination $g \in \mathcal{V}$. Each movement follows the graph topology, with $X_{t+1} \in \mathcal{N}^+_{\mathcal{G}}(X_t)$, where $\mathcal{N}^+_{\mathcal{G}}(X_t)$ denotes the outgoing neighbors of X_t. For a neighbor $u \in \mathcal{N}^+_{\mathcal{G}}(X_t)$, define the complete movement candidate $s^{(u)}_{t+1} = \big(\psi(u, X_t), u\big)$. We denote by $\mathcal{U}^{\text{move}}_t = \{s^{(u)}_{t+1} : u \in \mathcal{N}^+_{\mathcal{G}}(X_t)\}$ the set of legal movement candidates available at step t. The complete legal event set additionally includes the independent termination event only at the destination:

$$\mathcal{U}_t = \begin{cases} \mathcal{U}^{\text{move}}_t, & X_t \neq g, \\ \mathcal{U}^{\text{move}}_t \cup \{\text{STOP}\}, & X_t = g. \end{cases}$$

Actions and stopping time. At step t, the model selects a legal event A_{t+1} after observing c_t. A movement event is one of the complete-state candidates in $\mathcal{U}^{\text{move}}_t$ and produces its associated location X_{t+1}; STOP creates no new road state. Trajectory termination is defined by the stopping time

$$\tau = \inf\{t \geq 0 : X_t = g, A_{t+1} = \text{STOP}\}. \tag{2}$$

Because STOP is legal only at the destination, $A_{\tau+1} = \text{STOP}$ implies $X_\tau = g$. Reaching g alone does not terminate generation; the model must explicitly select STOP. An external transition budget may truncate generation before this selection, which is distinct from model-selected termination.

Conditional trajectory generation. Given the traversed history c_t and destination g, the task is to generate a topologically valid continuation and terminate it at the destination. A conventional sequential formulation selects a legal action according to $p(A_{t+1} \mid c_t, g)$, appends the resulting road location after a movement, and repeats until STOP is selected. Such a formulation makes the next decision directly from the observed history and destination. We instead study how to inform this local decision with the trajectory continuation that each candidate movement may induce. The conditional future of interest is $P(S_{t+1:\tau} \mid c_t, g)$, whose first event determines the next legal transition and whose remaining states describe a possible downstream continuation.

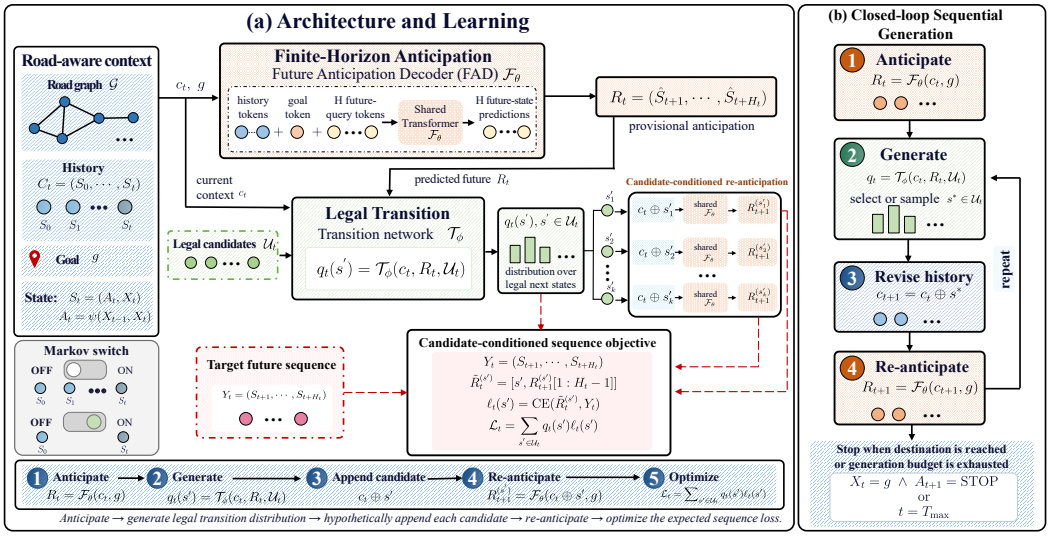


Figure 2: **Overview of FORESIGHT.** Further architectural details are provided in Appendix C.

4 METHODOLOGY

Distinguished from previous methods, we advocate a fundamental shift toward generative route planning on road networks. To resolve the next-transition ambiguity that arises from relying solely on the traversed history, we introduce a new graph-constrained generative paradigm, Anticipatory Generative Modeling (AGM), which addresses this challenge by anticipating how the future route unfolds toward the destination. Accordingly, we propose an anticipatory model, FORESIGHT, whose overall architecture is sketched in Fig. 2. *It recursively executes an "anticipate–select–update–re-anticipate" loop for stepwise routing on the road network.* Specifically, the Future Decoder first constructs an anticipation of a provisional short-horizon continuation. Then, the Transition Network selects the topologically valid outgoing direction based on its consistency with the anticipated future; the selected transition is appended to update the traversed sequence, and the model re-anticipates the trajectory suffix upon committing to the transition. This process repeats until `STOP` is chosen at the destination. Altogether, this endows the routing generator with longer-term foresight capacity while maintaining sequential generation along the historical trajectory.

4.1 ANTICIPATORY GENERATIVE MODELING

We model goal-conditioned road trajectories as realizations of an underlying stochastic process. Given an origin x_0 and destination g, each training trajectory $s_{0:\tau}^{(n)}$ is sampled from $P_{\text{data}}(S_{0:\tau} \mid X_0 = x_0, g)$ and represents one feasible route between the two endpoints. Trajectories with the same endpoints may follow different paths, so an observed trajectory is one possible realization of the process rather than a unique target that all generated trajectories must match.

At an anchor indexed by t, the traversed history c_t and destination g induce a conditional future process $P(S_{t+1:\tau} \mid c_t, g)$. The suffix observed in a training example is one realization of this process: its first element specifies the next state, while the remainder records one possible downstream continuation. At inference, **FORESIGHT** has access only to the traversed history and destination, from which it forms an anticipation of how the trajectory may continue.

A conventional stepwise generator directly models the next-state conditional $P(S_{t+1} \mid c_t, g)$, corresponding to the one-step marginal of the full future process:

$$P(s_{t+1} \mid c_t, g) = \sum_{s_{t+2:\tau}} P(s_{t+1}, s_{t+2:\tau} \mid c_t, g).$$

Although longer-term regularities may still be learned from data, they are not explicitly represented as anticipations of downstream evolution that directly inform the current transition.

Anticipatory Generative Modeling (AGM) instead makes plausible downstream evolution explicit before selecting the current transition. Given (c_t, g), **FORESIGHT** anticipates how the trajectory

may continue and uses this information to choose among topologically feasible transitions, without prescribing a unique future route.

Because the stopping time τ is random, the future continuation $S_{t+1:\tau}$ has variable length and becomes increasingly uncertain at longer horizons. Rather than representing the entire continuation before every transition, **FORESIGHT** realizes AGM through a finite anticipation window, leaving more distant evolution to subsequent predictions.

4.2 FINITE-HORIZON ANTICIPATION

For a realized trajectory, each traversed prefix c_t defines an *anchor* at step t, from which the conditional future process extends to the random stopping time τ. Since τ is unknown when the current decision is made, explicitly representing the full continuation would require modeling a variable-length predictive object. We instead restrict the explicitly represented future to a fixed horizon of H positions. Let Y_t denote the corresponding realized future window:

$$Y_t = S_{t+1:t+H}, \qquad Y_t \sim P_{\text{data}}(Y_t \mid c_t, g). \tag{3}$$

Here, Y_t provides the realized training target, whereas the model-predicted finite-horizon anticipation is denoted by $\widehat{R}_t$. The horizon H bounds the amount of future context exposed to the current decision rather than the length of the generated trajectory. If STOP occurs within the window, all subsequent positions are masked.

During training, the observed window Y_t supervises the predictive distributions represented by $\widehat{R}_t$. During inference, **FORESIGHT** predicts $\widehat{R}_t$ from (c_t, g) and uses its representation as downstream context when selecting the next legal transition.

The anticipation is revised after every realized movement:

$$c_t \longrightarrow \widehat{R}_t \longrightarrow S_{t+1} \longrightarrow c_{t+1} \longrightarrow \widehat{R}_{t+1}, \qquad \widehat{R}_{t+1} = \mathcal{F}_\theta(c_{t+1}, g). \tag{4}$$

Thus, the H-step anticipation is not executed open-loop; it is recomputed from the updated history after each realized movement.

4.3 FORESIGHT: NEURAL ARCHITECTURE

We instantiate finite-horizon anticipation with **FORESIGHT**, which couples a shared Future Anticipation Decoder with a topology-constrained Transition Network. Figure 2 summarizes the architecture and its step-wise generation process. Given the traversed history c_t and destination g, the decoder predicts a finite-horizon anticipation, whose contextual representations are then used by the Transition Network to score the legal transitions from the current state:

$$\widehat{R}_t = \mathcal{F}_\theta(c_t, g), \qquad q_t = \mathcal{T}_\phi(c_t, g, \widehat{R}_t; \mathcal{U}_t). \tag{5}$$

Here, $\widehat{R}_t$ contains contextual representations for the next H positions together with the parameters of their associated categorical distributions, and q_t is the distribution over the legal candidate set $\mathcal{U}_t$. We denote the categorical distribution at future position h by $\widehat{P}_{t,h}$.

The Future Anticipation Decoder $\mathcal{F}_\theta$ jointly encodes the traversed history, destination, and an ordered sequence of future queries with a shared Transformer. Its hidden states provide contextual representations for the next H positions, while prediction heads parameterize categorical distributions over the global predictive space. A structured visibility rule allows each future query to attend to the observed history, destination, and preceding query context, while preventing access to realized future states. The resulting anticipation therefore depends only on information available at the current anchor.

From the current road state x_t, Foresight forms the movement candidate set $\mathcal{U}_t^{\text{move}}$ from the topologically valid outgoing transitions. The legal candidate set is

$$\mathcal{U}_t = \begin{cases} \mathcal{U}_t^{\text{move}} \cup \{\text{STOP}\}, & x_t = g, \\ \mathcal{U}_t^{\text{move}}, & x_t \neq g, \end{cases}$$

so STOP is available only at the destination. The Transition Network scores every legal candidate with the same compatibility function:

$$\mathrm{score}_t(s') = \mathrm{score}_\phi(c_t, g, \widehat{R}_t, s'), \qquad q_t(s') = \frac{\exp\ \mathrm{score}_t(s')}{\sum_{\bar{s}\in\mathcal{U}_t} \exp\ \mathrm{score}_t(\bar{s})}, \qquad s' \in \mathcal{U}_t. \tag{6}$$

Thus, movement candidates and STOP are treated uniformly within the same topology-constrained distribution, while the anticipated future directly informs their scores.

After q_t has been formed, each movement candidate $s' \in \mathcal{U}_t^{\mathrm{move}}$ can be appended hypothetically to the traversed history and re-evaluated by the same decoder:

$$c_t^{(s')} = c_t \oplus s', \qquad \widehat{R}_{t+1}^{(s')} = \mathcal{F}_\theta(c_t^{(s')}, g), \qquad s' \in \mathcal{U}_t^{\mathrm{move}}. \tag{7}$$

These candidate-conditioned predictions represent the continuations induced by hypothetical legal movements. They are not separate observed targets; Section 4.4 combines them with the single realized continuation to define the sequence learning objective.

4.4 Learning Objective

For an anchor t on a sampled training trajectory, we extract an H-position future reference

$$y_t = (y_{t,1}, \ldots, y_{t,H}).$$

This reference is one realization of the conditional future process given the traversed history and goal, rather than a unique target route. Its first entry $y_{t,1}$ is the realized next event, while $y_{t,2:H}$ records the subsequent continuation. If STOP occurs within the horizon, its position is retained and all subsequent positions are masked; if $y_{t,1} = \mathrm{STOP}$, only the first position contributes. The reference is used only for training supervision and is unavailable at inference.

To combine the current transition with subsequent future predictions in the same H-position objective, all positions are represented in a common prediction space. The Transition Network defines q_t over the local action set $\mathcal{U}_t$, whereas future positions are predicted over the global space $\mathcal{Y} = \mathcal{V} \cup \{\mathrm{STOP}\}$. We therefore map q_t to a global distribution $\bar{q}_t$, assigning each movement probability to its resulting road state, assigning the STOP probability to STOP when applicable, and setting all other entries to zero. Within the transition-coupled objective, $\bar{q}_t$ provides the prediction for $y_{t,1}$.

The Future Decoder is directly supervised by the masked future-window cross-entropy

$$\mathcal{L}_{F,t} = -\sum_{h=1}^{H} m_{t,h} \log \widehat{P}_{t,h}(y_{t,h}).$$

For each admissible movement candidate $s' \in \mathcal{U}_t^{\mathrm{move}}$, we further use the hypothetical re-anticipation

$$\widehat{R}_{t+1}^{(s')} = \mathcal{F}_\theta(c_t \oplus s', g)$$

to obtain candidate-conditioned predictions for positions $h = 2, \ldots, H$. Since the training data provide no counterfactual continuation for unexecuted candidates, every candidate-conditioned prediction is evaluated against the same observed suffix $y_{t,2:H}$. This suffix serves as a compatibility reference rather than a factual continuation conditioned on the hypothetical candidate. Let $\widetilde{P}_{t,h}^{(s')}$ denote the corresponding global prediction at downstream position h. We define

$$\ell_t^{\mathrm{comp}}(s') = -\sum_{h=2}^{H} m_{t,h} \log \widetilde{P}_{t,h}^{(s')}(y_{t,h}).$$

The anchor-level transition-coupled objective is

$$\mathcal{L}_t = -m_{t,1} \log \bar{q}_t(y_{t,1}) + \sum_{s'\in\mathcal{U}_t^{\mathrm{move}}} q_t(s')\, \ell_t^{\mathrm{comp}}(s'). \tag{8}$$

The first term supervises the realized next event, while the second evaluates the expected downstream compatibility of candidate movements under q_t. The complete anchor loss is

$$\mathcal{J}_t = \mathcal{L}_{F,t} + \mathcal{L}_t.$$

Candidate-conditioned re-anticipation is computed only after q_t is formed and therefore affects the transition distribution only through the training objective. Training averages $\mathcal{J}_t$ over sampled trajectories and temporal anchors.

4.5 SEQUENTIAL GENERATION

At generation step t, given the realized history c_t and destination g, FORESIGHT first predicts the current finite-horizon anticipation and uses it to define a base distribution over the legal action set:

$$\widehat{R}_t = \mathcal{F}_\theta(c_t, g), \qquad q_t = \mathcal{T}_\phi(c_t, g, \widehat{R}_t; \mathcal{U}_t). \tag{9}$$

Generation uses only the realized history, the destination, and model-predicted anticipations; the future reference used for training is never available. By default, the next action is selected directly from q_t. Optionally, FORESIGHT can further rerank legal movements according to the consistency between the current anticipation and the continuation predicted after each hypothetical candidate.

For a movement candidate $s' \in \mathcal{U}_t^{\mathrm{move}}$, let

$$\widehat{R}_{t+1}^{(s')} = \mathcal{F}_\theta(c_t \oplus s', g)$$

denote its candidate-conditioned re-anticipation, and let $\widehat{P}_{t,h}$ and $\widehat{P}_{t+1,h}^{(s')}$ denote the categorical distributions associated with position h of $\widehat{R}_t$ and $\widehat{R}_{t+1}^{(s')}$, respectively. Since advancing by one movement shifts the prediction horizon by one position, future consistency is measured over their overlapping predictions:

$$E_{\mathrm{fut}}(s') = \frac{1}{H-1} \sum_{h=1}^{H-1} D_{\mathrm{JS}} \left(\widehat{P}_{t,h+1} \,\|\, \widehat{P}_{t+1,h}^{(s')} \right), \qquad s' \in \mathcal{U}_t^{\mathrm{move}}, \tag{10}$$

where D_{JS} is the Jensen–Shannon divergence. We set $E_{\mathrm{fut}}(\mathrm{STOP}) = 0$. The resulting selection energy and distribution are

$$E_t(s') = -\log q_t(s') + w E_{\mathrm{fut}}(s'), \qquad \pi_t(s') = \frac{\exp[-E_t(s')]}{\sum_{u \in \mathcal{U}_t} \exp[-E_t(u)]}, \qquad \hat{a}_{t+1} = \mathrm{Select}(\pi_t; \mathcal{U}_t). \tag{11}$$

When $w = 0$, $\pi_t = q_t$, recovering selection from the base transition distribution. Candidate-conditioned branches used in reranking remain hypothetical: they are evaluated only to score the current decision, and only the selected action $\hat{a}_{t+1}$ is executed and appended to the realized history.

If $\hat{a}_{t+1} \in \mathcal{U}_t^{\mathrm{move}}$, executing the selected movement determines the next road position. Let $\mathrm{dest}(\hat{a}_{t+1})$ denote the destination road position of the selected legal movement. Consistent with the state definition $S_t = (A_t, X_t)$, the realized state and updated history are

$$x_{t+1} = \mathrm{dest}(\hat{a}_{t+1}), \qquad s_{t+1} = (\hat{a}_{t+1}, x_{t+1}), \qquad c_{t+1} = c_t \oplus s_{t+1}. \tag{12}$$

FORESIGHT then recomputes the anticipation from the updated realized history:

$$\widehat{R}_{t+1} = \mathcal{F}_\theta(c_{t+1}, g). \tag{13}$$

The anticipation therefore informs the current transition without prescribing an open-loop route. Each executed movement changes the realized history and induces a new prediction, yielding a closed-loop anticipate–select–update–re-anticipate process.

If $\hat{a}_{t+1} = \mathrm{STOP}$, generation terminates immediately and no subsequent movement state is appended. Reaching $x_t = g$ only makes STOP admissible; termination still requires selecting it. A maximum generation budget provides an external evaluation-time truncation: reaching the budget before STOP is recorded as a timeout rather than as stochastic termination.

5 EXPERIMENTS

We study whether Foresight can reproduce both the spatial distribution of held-out mobility and the route structure of individual trips. The experiments are organized around four questions: how the model compares with existing generators, whether its samples occupy space like the test set, what its generated routes look like under matched conditions, and which parts of the anticipatory state are responsible for the gains.

Table 1: Overall performance across cities. **D-JSD/R-JSD denote Distance/Radius JSD and are reported in units of 10^{-2}.** HD is Hausdorff distance and EDR is edit distance on real sequences. Cells report OD-bootstrap means under the common evaluation protocol (see Appendix E). Lower is better.

	Porto				San Francisco				Beijing			
Method	D-JSD	R-JSD	HD	EDR	D-JSD	R-JSD	HD	EDR	D-JSD	R-JSD	HD	EDR
Dijkstra	2.47	1.31	0.481	0.358	2.18	1.29	0.399	0.305	2.64	1.56	0.449	0.341
Yen-k	2.36	1.29	0.503	0.386	2.34	1.39	0.497	0.377	2.78	1.67	0.552	0.414
Markov	2.61	1.46	0.613	0.462	2.39	1.42	0.602	0.458	2.88	1.72	0.681	0.503
SemRoute	1.68	1.07	0.514	0.384	1.69	0.91	0.481	0.363	2.16	1.19	0.548	0.414
HOSER	1.57	0.96	0.431	0.317	1.82	0.98	0.412	0.306	2.04	1.09	0.463	0.351
TS-TrajGen	3.24	1.83	0.641	0.481	3.06	1.86	0.608	0.462	3.42	2.16	0.684	0.503
GDP	2.43	1.57	0.531	0.397	2.31	1.31	0.501	0.376	2.76	1.68	0.553	0.417
Diff-RNTraj	2.34	1.39	0.476	0.357	2.19	1.28	0.417	0.319	2.63	1.54	0.468	0.352
DiffTraj	2.83	1.76	0.637	0.479	2.57	1.61	0.617	0.468	3.01	1.98	0.692	0.504
TrajFlow	1.72	1.08	0.536	0.401	1.71	0.88	0.472	0.354	2.23	1.21	0.546	0.411
Seed	1.79	1.14	0.528	0.398	1.86	1.04	0.506	0.382	2.28	1.29	0.559	0.419
FORESIGHT-F	1.53	0.79	0.401	0.312	**1.39**	0.71	**0.386**	**0.298**	**1.68**	**0.88**	**0.418**	**0.326**
FORESIGHT-M	**1.01**	**0.71**	**0.375**	**0.285**	2.68	**0.65**	0.542	0.415	1.87	1.01	0.438	0.344

Table 2: Ablation results for Porto and San Francisco. **JSD values are scaled by 10^{-2}.**

	Porto				San Francisco			
Variant	D-JSD	R-JSD	HD	EDR	D-JSD	R-JSD	HD	EDR
FORESIGHT-F	1.53	0.79	0.401	0.312	**1.39**	0.71	**0.386**	**0.298**
FORESIGHT-M	**1.01**	**0.71**	**0.375**	**0.285**	2.68	**0.65**	0.542	0.415
w/o $\mathcal{L}_{F,t}$	2.25	1.35	0.500	0.390	2.10	1.25	0.480	0.380
w/o ℓ_t^{comp}	2.55	1.55	0.550	0.420	2.40	1.45	0.530	0.410
w/o E_{fut}	2.05	1.20	0.460	0.360	1.95	1.15	0.450	0.350

Experimental setup. We evaluate goal-conditioned generation on the Porto, Beijing, and San Francisco datasets with the official HOSER graphs and splits.[1] All methods use matched test OD conditions, replicate counts, and generation budgets; we report Distance/Radius JSD, Hausdorff distance, and EDR (lower is better). Full protocol, training, and metric details appear in Appendices D.

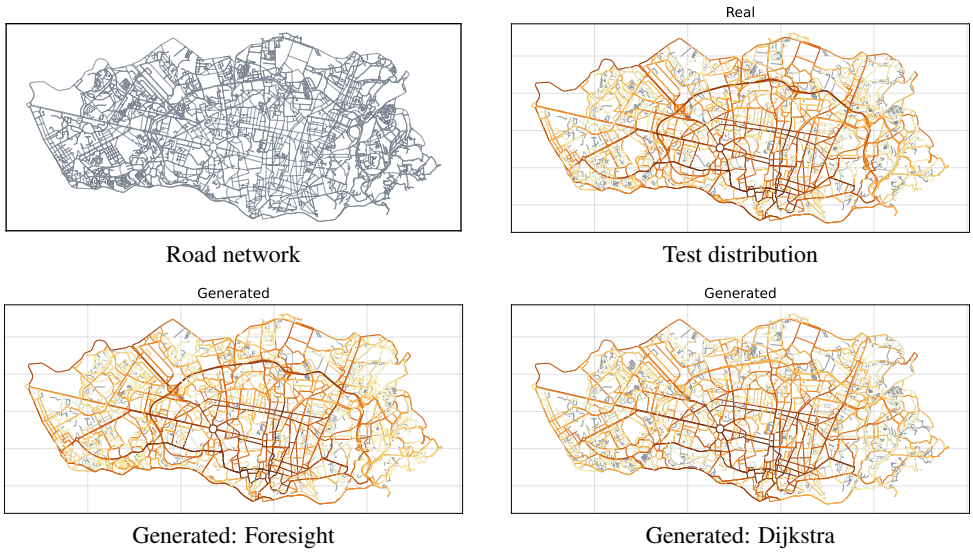
Baselines. Baselines span the Markov reference, Dijkstra and Yen's k-shortest-path search (Dijkstra, 1959; Yen, 1971), Semantic Routing and HOSER (Zhao et al., 2024; Cao et al., 2025), and learned generators: TS-TrajGen (Con, 2023), GDP (Shi et al., 2024), Diff-RNTraj (Wei et al., 2024), DiffTraj (Zhu et al., 2023), TrajFlow (Li et al., 2026), and Seed (Rao et al., 2025). Adaptation and implementation details are in Appendix D; all methods use the matched evaluation protocol above.

Overall performance. Table 1 reports distribution-level and route-level fidelity under the common evaluation protocol. Distance JSD and Radius JSD are reported in units of 10^{-2}; complete results with bootstrap standard deviations, DTW, and completion diagnostics are provided in Appendix E. The two Foresight variants show complementary behavior: the full-history variant is strongest on San Francisco and Beijing, whereas the Markov-history variant performs best on Porto and remains competitive elsewhere. Completion diagnostics use the same test conditions and retain failed rollouts in their denominators.

Spatial distributions and route examples. The quantitative comparison is complemented by spatial and route views under matched held-out conditions; the complete spatial and route panels are in Appendix F.

Ablation of anticipatory modeling. We ablate complete trajectory history, future-window supervision $\mathcal{L}_F$, candidate-conditioned continuation loss ℓ_t^{con}, and inference-time future energy E_{fut}. As shown in Table 2, candidate-conditioned continuation contributes the most, followed by future-window supervision, while future-energy reranking provides a smaller but consistent improvement.

[1]The HOSER dataset is available at `https://huggingface.co/datasets/caoji2001/HOSER-dataset`.

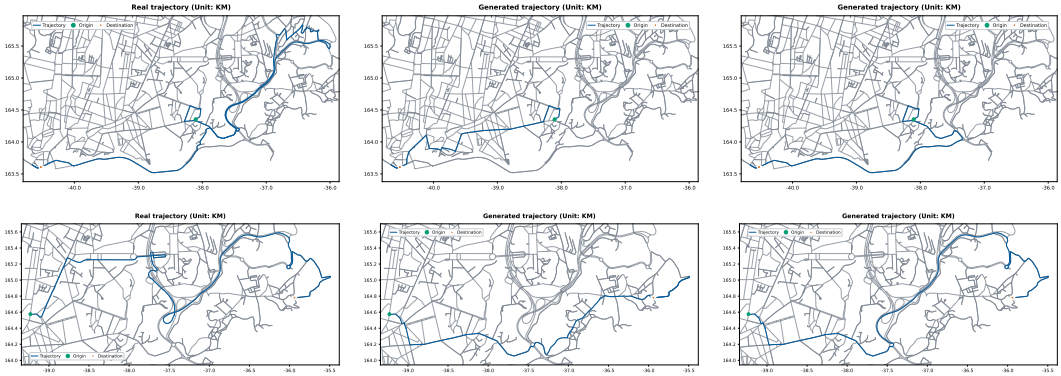


Road network | Test distribution

Generated: Foresight | Generated: Dijkstra

Figure 3: Porto spatial comparison. Top: road network and test-set trajectory distribution. Bottom: trajectory distributions generated by Foresight and Dijkstra.

Full results, including dataset-specific history effects, standard deviations, DTW, and Beijing experiments, are provided in Appendix E.



Real Trajectory | Dijkstra | Foresight

Figure 4: Matched Porto route comparison for two fixed test OD conditions.

6 Conclusion

This work revisits trajectory generation through the lens of generative route planning on road networks. To this end, we introduce a new graph-constrained generative paradigm, Anticipatory Generative Modeling, which resolves the ambiguity in next-transition selection by anticipating how the future route unfolds toward the destination. Specifically, we present an anticipatory model, Foresight. At each step, it (i) constructs an *anticipation* of a provisional short-horizon continuation, (ii) *selects* among topologically valid transitions via a direction-aware fiber, (iii) *updates* the traversed sequence through the decoder, and (iv) *re-anticipates* the trajectory suffix upon committing to the transition. Experiments on real-world datasets show that Foresight achieves superior performance in both trajectory distribution matching and route fidelity under origin–destination constraints. These results suggest that the proposed anticipation mechanism helps disambiguate locally plausible transitions while preserving topological validity throughout sequential generation.

AI USE STATEMENT

AI tools were used solely for language polishing and formatting adjustments during manuscript preparation. They were not used to develop the research ideas or methodology, conduct experiments, analyze experimental results, or determine the scientific claims and conclusions of this work. The authors reviewed and take responsibility for all content in the manuscript.

REPRODUCIBILITY STATEMENT

We document the inputs and procedures needed to reproduce the stated experiments. We use the public HOSER road graphs and official train/validation/test splits for Porto, Beijing, and San Francisco (`https://huggingface.co/datasets/caoji2001/HOSER-dataset`). Preprocessing reads `roadmap.geo`, `roadmap.rel`, and the supplied trajectory files; it validates road identifiers, graph-valid transitions, timestamp consistency, and finite attributes. Continuous-feature statistics are fitted on training data and reused for validation, test, and generation. Trajectories are not re-split, and test trajectories do not fit preprocessing statistics or model parameters. Each run is bound to a versioned data/configuration manifest, a validation-selected checkpoint, and a fixed test-condition and replicate list.

The method appendices fix the model interfaces and training boundaries. Shared Base/Fiber encoders construct history-consistent node–fiber states and enumerate only graph-legal successors. A structured Future Transformer uses a fixed $H = 64$ query window, global node-ID targets, and an independent `STOP` event; its visibility mask excludes future labels from the context. The Transition Network builds a candidate-independent `q_trans` readout, then scores complete legal candidate states with a projected dot product and positive temperature. Full-history and Markov variants share this interface; the latter retains only $[S_0, S_t]$ and the goal in both modules. One joint optimizer transaction combines masked future event cross-entropy, current-event NLL, and expected future cross-entropy over candidate branches. Future targets are stop-gradient, Transition inputs are detached, and discrete actions are detached before history append. The reported training configuration uses AdamW for 60,000 updates with fixed validation checkpoint selection; each run records scale dimensions, parameter counts, normalization settings, precision, clipping, and random states. The appendix lists the five scale presets, their widths, depths, heads, horizons, and city-dependent vocabulary parameter counts for direct checkpoint comparison.

Generation uses the frozen model in evaluation/no-gradient mode. At every decision it rebuilds the fixed future window from the realized history, passes the downstream future representation to Transition, scores legal outgoing graph edges, and admits `STOP` only at the goal. Reaching the goal alone does not terminate a rollout. The reported protocol uses 5,000 fixed test conditions, and a maximum of 256 transitions per attempt; greedy or sampling selection and candidate-conditioned future-consistency reranking are run-level fields. Every attempt is retained, including timeouts, invalid transitions, and numerical failures, with condition/OD identifiers, seed, node sequence, stop reason, and checkpoint/configuration references. Baselines share the graph, test conditions, replicate policy, and failure-accounting rule; their adapters, decoding/search settings, and budgets are recorded separately.

Evaluation is reproducible from saved generation records and the evaluation configuration. The primary metrics are Distance JSD, Radius JSD, Hausdorff distance, DTW, and EDR; undefined metrics stay unavailable rather than becoming zeros. Completion statistics use the complete attempted corpus as their denominator. Exact-OD bootstrap is primary and uses 1,000 resamples with bootstrap seed 0; condition-bootstrap and HOSER-style file-level aggregation are secondary checks. The experimental appendix specifies supports, binning, matching rules, and Python/PyTorch-style pseudocode for training, inference, and evaluation. The anonymous code repository listed in the abstract is the release location for implementation and configuration files (`https://anonymous.4open.science/r/Foresight-67ED`). Cells without measured values identify missing runs or inputs and are not interpreted as results.

REFERENCES

Continuous trajectory generation based on two-stage gan. In *AAAI 2023*, 2023. doi: 10.1609/aaai.v37i4.25557. URL https://doi.org/10.1609/aaai.v37i4.25557.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017. URL https://openreview.net/forum?id=rJY3vK9eg.

Ji Cao, Tongya Zheng, Qinghong Guo, Yu Wang, Junshu Dai, Shunyu Liu, Jie Yang, Jie Song, and Mingli Song. Holistic semantic representation for navigational trajectory generation. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i1.31978.

Lei Chen, M. Tamer Özsu, and Vassilis Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pp. 491–502. ACM, 2005.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097, 2021.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271, 1959. doi: 10.1007/BF01386390.

Tao-yang Fu and Wang-Chien Lee. Progrpgan: Progressive gan for route planning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 393–403. Association for Computing Machinery, 2021. doi: 10.1145/3447548.3467406.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4:100–107, 1968.

Zhao Huang, Bowen Zhang, Zhongzhu Li, and Di Lin. Occdriver: Future occupancy guided dual-branch trajectory planner in autonomous driving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=abJCjkIwi5.

Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

Jayant Jain, Vrittika Bagadia, Sahil Manchanda, and Sayan Ranu. NeuroMLR: Robust & reliable route recommendation on road networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Sl0WX9H6ZJg.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem, 2021. URL https://arxiv.org/abs/2106.02039.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Chiyu Max Jiang, Yijing Bai, Andre Cornman, Christopher Davis, Xiukun Huang, Hong Jeon, Sakshum Kulshrestha, John Lambert, Shuangyu Li, Xuanyu Zhou, Carlos Fuertes, Chang Yuan, Mingxing Tan, Yin Zhou, and Dragomir Anguelov. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. In *Advances in Neural Information Processing Systems*, volume 37, pp. 55729–55760. Curran Associates, Inc., 2024. doi: 10.52202/079017-1771.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Peiran Li, Jiawei Wang, Haoran Zhang, Xiaodan Shi, Noboru Koshizuka, Chihiro Shimizu, and Renhe Jiang. Trajflow: Nation-wide pseudo GPS trajectory generation with flow matching models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BDOldEjwCE.

Shu Liu, Wenlin Chen, Weihao Li, Zheng Wang, Lijin Yang, Jianing Huang, YipinZhang, Zhongzhan Huang, Ze Cheng, and Hao Yang. Bridgedrive: Diffusion bridge policy for closed-loop trajectory planning in autonomous driving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dJKhjK4zpp.

Xuan Rao, Shuo Shang, Renhe Jiang, Peng Han, and Lisi Chen. Seed: Bridging sequence and diffusion models for road trajectory generation. In *Proceedings of the ACM Web Conference 2025*, pp. 2007–2017, 2025. doi: 10.1145/3696410.3714951.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

Dingyuan Shi, Yongxin Tong, Zimu Zhou, Ke Xu, Zheng Wang, and Jieping Ye. GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vuK8MhVtuu.

Qian Sun, Le Zhang, Huan Yu, Weijia Zhang, Yu Mei, and Hui Xiong. Hierarchical reinforcement learning for dynamic autonomous vehicle navigation at intelligent intersections. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 4852–4861. Association for Computing Machinery, 2023. doi: 10.1145/3580305.3599839.

Tonglong Wei, Youfang Lin, Shengnan Guo, Yan Lin, Yiheng Huang, Chenyang Xiang, Yuqing Bai, and Huaiyu Wan. Diff-rntraj: A structure-aware diffusion model for road network-constrained trajectory generation. *IEEE Trans. on Knowl. and Data Eng.*, 36(12):7940–7953, 2024. ISSN 1041-4347. doi: 10.1109/TKDE.2024.3460051.

Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning*, 2023.

Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. In *Advances in Neural Information Processing Systems*, volume 37, pp. 114048–114071. Curran Associates, Inc., 2024. doi: 10.52202/079017-3622.

Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971. doi: 10.1287/mnsc.17.11.712. URL https://doi.org/10.1287/mnsc.17.11.712.

Ryo Yonetani, Tatsunori Taniai, Mohammadamin Barekatain, Mai Nishimura, and Asako Kanezaki. Path planning using neural a* search. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12029–12039. PMLR, 2021.

Ximu Zeng, Quanlin Yu, Shuncheng Liu, Yuyang Xia, Han Su, and Kai Zheng. Target-oriented maneuver decision for autonomous vehicle: A rule-aided reinforcement learning framework. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 3124–3133, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3583780.3615072.

Eric Zhao, Pranjal Awasthi, Zhengdao Chen, Sreenivas Gollapudi, and Daniel Delling. Semantic routing via autoregressive modeling. In *Advances in Neural Information Processing Systems*, volume 37, pp. 10060–10087, 2024. doi: 10.52202/079017-0323.

Yinan Zheng, Ruiming Liang, Kexin ZHENG, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wM2sfVgMDH.

Yuanshao Zhu, Yongchao Ye, Shiyao Zhang, Xiangyu Zhao, and James Yu. Difftraj: Generating gps trajectory with diffusion probabilistic model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 65168–65188. Curran Associates, Inc., 2023. doi: 10.52202/075280-2843.

Yuanshao Zhu, James Jianqiao Yu, Xiangyu Zhao, Qidong Liu, Yongchao Ye, Wei Chen, Zijian Zhang, Xuetao Wei, and Yuxuan Liang. Controltraj: Controllable trajectory generation with topology-constrained diffusion model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 4676–4687. Association for Computing Machinery, 2024. doi: 10.1145/3637528.3671866.

A NOTATION

Table 3 summarizes the symbols used in the problem formulation, model, and training objective. We distinguish road locations, direction-aware states, legal events, and the global output alphabet. The training target y_t is a realized reference window, whereas $\widehat{R}_t$ is the model-predicted anticipation.

Table 3: Core notation used in the paper.

Symbol	Meaning
Trajectory, state, and legal events	
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Directed road graph.
$X_t \in \mathcal{V}$	Location at step t.
$X_{0:T}$	Location trajectory; T is an index (and may be the endpoint of a realized trajectory).
$g \in \mathcal{V}$	Destination location.
$A_t = \psi(X_t, X_{t-1})$	Direction/fiber attribute induced by the incoming movement; A_0 is the initialization token.
$S_t = (A_t, X_t)$	Direction-aware road state.
$c_t = S_{0:t}$	Realized direction-aware state history before the next decision.
$\mathcal{N}_{\mathcal{G}}^{+}(X_t)$	Outgoing neighbors of X_t.
$s'(u) = (\psi(u, X_t), u)$	Complete direction-aware movement candidate for neighbor u.
$\mathcal{U}_t^{\text{move}}$	Legal movement candidates, one for each outgoing neighbor.
$\mathcal{U}_t$	Legal event support; it adds STOP only when $X_t = g$.
τ	Model stopping time: the first step at which the selected event is STOP; this is legal only when $X_t = g$.
$\oplus$	History append operation, e.g., $c_{t+1} = c_t \oplus S_{t+1}$.
Anticipation, targets, and candidate branches	
$\widehat{R}_t = \mathcal{F}_\theta(c_t, g)$	Model-predicted finite-horizon anticipation with H positions (written as R_t in the probabilistic appendix).
H	Anticipation and supervision-window length.
$y_t = (y_{t,1}, \ldots, y_{t,H})$	Realized training target window over locations and STOP; it retains STOP and uses PAD after STOP.
$\mathcal{Y} = \mathcal{V} \cup \{\text{STOP}\}$	Global output alphabet.
PAD	Storage-only padding symbol; it is not in $\mathcal{Y}$.
$m_{t,h} \in \{0, 1\}$	Mask: valid through STOP, then zero.
$\widehat{P}_{t,h}$	Decoder categorical distribution over $\mathcal{Y}$ at anticipation position h.
$\mathcal{L}_{F,t}$	Masked future-window cross-entropy for the decoder.
$\mathcal{F}_\theta$	Future Anticipation Decoder mapping (c_t, g) to $\widehat{R}_t$.
$\mathcal{T}_\phi$	Transition Network mapping $(c_t, g, \widehat{R}_t, \mathcal{U}_t)$ to legal-event scores/probabilities.
$q_t(e)$	Probability of legal event $e \in \mathcal{U}_t$.
$\bar{q}_t$	Sparse q_t aligned to $\mathcal{Y}$ via ι.
ι	Event-to-output alignment: $\iota(s'(u)) = u$ and $\iota(\text{STOP}) = \text{STOP}$.
$c_t^{(s')} = c_t \oplus s'$	History after appending candidate s'.
$\widehat{R}_{t+1}^{(s')}$	Candidate-conditioned re-anticipation after appending movement s'.
Learning and model parameters	
$\widetilde{P}_{t,h}^{(s')}$	Global distribution aligned at position h: $\bar{q}_t$ at $h = 1$ and the candidate-conditioned decoder distribution afterward.
$\ell_t^{\text{comp}}(s')$	Masked suffix compatibility cross-entropy for movement candidate s'.
$\mathcal{L}_t$	Transition-coupled anchor objective: current-event NLL plus the q_t-weighted candidate compatibility loss.
$\mathcal{J}_t$	Complete anchor loss, $\mathcal{L}_{F,t} + \mathcal{L}_t$.
$E_{\text{fut}}(s')$	Optional future-consistency energy for movement candidate s'.
π_t	Optional reranked legal-event distribution used for selection.
θ, ϕ	Future Decoder and Transition Network parameters, respectively.
$\mathbf{S}$	Random-length stopped state trajectory, $S_{0:\tau}$.

The directional attribute is determined by the incoming movement, whereas the destination condition is determined by node identity. The realized target y_t and mask $m_{t,h}$ are used only for training; at

inference, $\widehat{R}_t$ is predicted from (c_t, g) and only the selected event is executed. STOP is an event in $\mathcal{U}_t$ only at the destination and does not correspond to a successor state.

B Probabilistic Foundation of Foresight

This appendix gives a probabilistic interpretation of Foresight that is consistent with Section 4. The underlying object is a random-length trajectory on a directed road network. A training trajectory is one realization of this process, and a future window extracted from it is one sampled realization at a temporal anchor. In contrast, R_t denotes a model-predicted anticipation computed from the history and goal. Keeping these objects distinct is essential when hypothetical candidate-conditioned continuations are evaluated.

B.1 Random-Length Road-Network Trajectory Process

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed road network and let $X_t \in \mathcal{V}$ denote the road state at step t. For an ordinary movement, the direction-aware action is induced by two consecutive road states,

$$A_t = \psi(X_t, X_{t-1}) \in \mathcal{A}_{\text{move}}, \qquad S_t = (A_t, X_t), \quad t \geq 1, \tag{14}$$

where $A_0 = \text{BOS}$ and $S_0 = (\text{BOS}, X_0)$. Thus S_t is induced by the road-state process; A_t is not an additional independent latent process. The realized history before the next decision is

$$c_t = S_{0:t}, \qquad g \in \mathcal{V}, \tag{15}$$

where g is the destination and X_0 is the origin encoded by S_0.

The action alphabet includes an explicit terminal event,

$$\mathcal{A}^+ = \mathcal{A}_{\text{move}} \cup \{\text{STOP}\}. \tag{16}$$

The stopping time is defined by that event alone:

$$\tau = \inf\{t \geq 0 : A_{t+1} = \text{STOP}\}. \tag{17}$$

At $X_t = g$, STOP becomes admissible, but reaching the destination does not terminate the process automatically. If $A_{t+1} \in \mathcal{A}_{\text{move}}$, a new road state is realized and $A_{t+1} = \psi(X_{t+1}, X_t)$; if $A_{t+1} = \text{STOP}$, no ordinary X_{t+1} or S_{t+1} is generated. With the decision filtration

$$\mathcal{F}_t = \sigma(S_0, A_1, S_1, \ldots, S_t, A_{t+1}), \tag{18}$$

τ is a stopping time. The complete stopped trajectory is

$$\mathbf{S} = S_{0:\tau} = (S_0, \ldots, S_\tau), \tag{19}$$

whose length is random. At the data level,

$$\mathbf{S}^{(n)} \sim P_{\text{data}}(\mathbf{S} \mid X_0, g). \tag{20}$$

A training trajectory is one realization of this goal-conditioned process, not a deterministic specification of a unique route between its endpoints. The conditional distribution is learned from regularities across trajectory samples and temporal anchors; a single realization does not identify the full multimodal distribution.

B.2 Goal-Conditioned Future Process and Anticipatory Modeling

At an anchor $t \leq \tau$, the complete future is the variable-length suffix $S_{t+1:\tau}$ with conditional distribution

$$P(S_{t+1:\tau} \mid c_t, g). \tag{21}$$

The usual next-step conditional is its one-step marginal,

$$P(S_{t+1} \mid c_t, g) = \sum_{S_{t+2:\tau}} P(S_{t+1:\tau} \mid c_t, g). \tag{22}$$

The summation represents marginalization over downstream continuations. Long-range regularities may therefore be encoded implicitly, but the future is not an explicit context for the current decision.

Anticipatory Generative Modeling (AGM) makes such context explicit by predicting a finite future representation before selecting the current legal transition.

For a sampled trajectory $\mathbf{S}^{(n)}$, the observed suffix $S^{(n)}_{t+1:\tau^{(n)}}$ is one realization of the conditional future for $(c_t^{(n)}, g)$. It is a supervision reference, not a unique route and not an inference input. The learned computation is

$$R_t = \mathcal{F}_\theta(c_t, g), \qquad q_t = \mathcal{T}_\phi(c_t, g, R_t; \mathcal{U}_t). \tag{23}$$

These arrows describe computation and do not assert a conditional-independence statement such as $S_{t+1} \perp (c_t, g) \mid R_t$.

B.3 Finite-Horizon Anticipation

The full future has random length, whereas the model exposes a fixed number H of future positions at each decision. Let

$$\mathcal{Y} = \mathcal{V} \cup \{\mathrm{STOP}\}, \qquad y_t = (y_{t,1}, \ldots, y_{t,H}), \quad y_{t,h} \in \mathcal{Y} \cup \{\mathrm{PAD}\}. \tag{24}$$

For a sampled trajectory and anchor t, define the STOP-aware reference by

$$y_{t,h} = \begin{cases} X_{t+h}, & 1 \le h \le \tau - t, \\ \mathrm{STOP}, & h = \tau - t + 1, \\ \mathrm{PAD}, & h > \tau - t + 1, \end{cases} \qquad m_{t,h} = \mathbf{1}\{h \le \tau - t + 1\}. \tag{25}$$

The STOP position is supervised, while positions after STOP are masked. PAD is only a storage symbol and is not a decoder output.

The model-predicted anticipation is

$$R_t = \mathcal{F}_\theta(c_t, g), \tag{26}$$

whose H prediction positions provide downstream context for the Transition Network. Thus H is the explicit anticipation horizon, not the length of a generated trajectory, and fixing H does not impose conditional independence on more distant states. The sampled reference y_t is used only during training.

The anticipation is provisional and is recomputed after each realized movement:

$$R_t \longrightarrow s_{t+1} \longrightarrow c_{t+1} \longrightarrow R_{t+1}, \qquad c_{t+1} = c_t \oplus s_{t+1}, \quad R_{t+1} = \mathcal{F}_\theta(c_{t+1}, g). \tag{27}$$

B.4 Legal Transition, Candidate-Conditioned Prediction, and Learning

Let $\mathcal{N}_G^+(X_t)$ denote the outgoing neighbors of X_t. The legal movement candidates are

$$\mathcal{U}_t^{\mathrm{move}} = \{s'(u) : u \in \mathcal{N}_G^+(X_t)\}, \qquad s'(u) = \big(\psi(u, X_t), u\big), \tag{28}$$

and the current event support is

$$\mathcal{U}_t = \begin{cases} \mathcal{U}_t^{\mathrm{move}}, & X_t \ne g, \\ \mathcal{U}_t^{\mathrm{move}} \cup \{\mathrm{STOP}\}, & X_t = g. \end{cases} \tag{29}$$

The transition distribution is normalized only on this topology-constrained support:

$$q_t = \mathcal{T}_\phi(c_t, g, R_t; \mathcal{U}_t), \qquad \sum_{e \in \mathcal{U}_t} q_t(e) = 1. \tag{30}$$

Future positions use the global alphabet $\mathcal{Y}$, whereas current events use $\mathcal{U}_t$. With $\iota(s'(u)) = u$ and $\iota(\mathrm{STOP}) = \mathrm{STOP}$, align the current distribution by

$$\bar{q}_t(y) = \sum_{\substack{e \in \mathcal{U}_t \\ \iota(e) = y}} q_t(e), \qquad y \in \mathcal{Y}. \tag{31}$$

This is a probability-space alignment, not a second classifier.

After q_t has been formed, each hypothetical movement candidate defines a candidate-conditioned history and re-anticipation:

$$c_t^{(s')} = c_t \oplus s', \qquad R_{t+1}^{(s')} = \mathcal{F}_\theta(c_t \oplus s', g), \quad s' \in \mathcal{U}_t^{\text{move}}. \tag{32}$$

The order is

$$R_t \longrightarrow q_t \longrightarrow c_t \oplus s' \longrightarrow R_{t+1}^{(s')}. \tag{33}$$

$R_{t+1}^{(s')}$ is a model prediction under a hypothetical history, not an observed counterfactual continuation. STOP has no ordinary continuation branch.

Let $p_{t+1,j}^{(s')}$ be the decoder distribution at downstream position j. For alignment with positions $2, \ldots, H$, define

$$\widetilde{P}_{t,1}^{(s')} = \bar{q}_t, \qquad \widetilde{P}_{t,h}^{(s')} = p_{t+1,h-1}^{(s')}, \quad 2 \le h \le H. \tag{34}$$

The sampled-future compatibility cost is

$$\ell_t^{\text{comp}}(s') = -\sum_{h=2}^{H} m_{t,h} \log \widetilde{P}_{t,h}^{(s')}(y_{t,h}). \tag{35}$$

The same sampled suffix is a compatibility reference for all hypothetical branches. For an unexecuted candidate, $y_{t,2:H}$ is generally not a sample from $P_{\text{data}}(\cdot \mid c_t \oplus s', g)$; the data contain no factual future for every counterfactual movement. The unified anchor criterion is

$$\mathcal{L}_t = -m_{t,1} \log \bar{q}_t(y_{t,1}) + \sum_{s' \in \mathcal{U}_t^{\text{move}}} q_t(s') \ell_t^{\text{comp}}(s'). \tag{36}$$

The first term supervises the observed current event, including STOP. The second couples transition probabilities to sampled-future compatibility over hypothetical movement branches. It is one unified criterion rather than separately weighted transition, future, and rollout losses. Candidate re-anticipation is not a forward input to q_t, but its compatibility cost affects learning through the objective.

For valid trajectory-anchor pairs $\mathcal{I}$, the dataset objective is

$$\widehat{\mathcal{R}}(\Theta) = \frac{1}{|\mathcal{I}|} \sum_{(n,t) \in \mathcal{I}} \mathcal{L}_t^{(n)}, \qquad \mathcal{R}(\Theta) = \mathbb{E}_{\mathbf{S} \sim P_{\text{data}}} \left[\sum_{t \le \tau} \mathcal{L}_t(\mathbf{S}) \right]. \tag{37}$$

The empirical average is a Monte Carlo estimate of the expected value of the defined training criterion. This interpretation does not turn an unobserved branch into an observed counterfactual sample, nor does it claim that $\ell_t^{\text{comp}}(s')$ is an unbiased estimator of a true counterfactual conditional likelihood.

B.5 Sequential Generation as a Stochastic Process

At inference, only the realized history and destination are available:

$$R_t = \mathcal{F}_\theta(c_t, g), \qquad q_t = \mathcal{T}_\phi(c_t, g, R_t; \mathcal{U}_t), \qquad \hat{a}_{t+1} = \text{Select}(q_t; \mathcal{U}_t). \tag{38}$$

The sampled training window y_t is never provided. If $\hat{a}_{t+1} = s' \in \mathcal{U}_t^{\text{move}}$, then

$$s_{t+1} = s', \qquad c_{t+1} = c_t \oplus s_{t+1}, \qquad R_{t+1} = \mathcal{F}_\theta(c_{t+1}, g). \tag{39}$$

If $\hat{a}_{t+1} = \text{STOP}$, generation terminates with $\tau = t$ and no ordinary state is appended. A maximum generation budget may truncate a run before STOP, but it does not redefine τ.

For stochastic action selection, the probability of a stopped realization is schematically

$$P_\Theta(S_{1:\tau}, A_{\tau+1} = \text{STOP} \mid S_0, g) = \left[\prod_{t=0}^{\tau-1} q_t(S_{t+1}) \right] q_\tau(\text{STOP}), \tag{40}$$

where each q_t is recomputed from the current realized history. Training may evaluate all hypothetical movement branches for compatibility; inference realizes exactly one legal event and updates only that branch. The resulting closed-loop process is

$$\begin{aligned}
&(X_{t-1}, X_t) \to A_t = \psi(X_t, X_{t-1}), \quad (A_t, X_t) \to S_t;\\
&(c_t, g) \to R_t \to q_t \to \hat{a}_{t+1};\\
&\text{movement: } \hat{a}_{t+1} \to s_{t+1} \to c_{t+1} \to R_{t+1};\\
&\text{STOP: } \hat{a}_{t+1} = \mathrm{STOP} \to \tau;\\
&\text{training branch: } q_t \to s' \to c_t \oplus s' \to R_{t+1}^{(s')} \dashrightarrow y_t.
\end{aligned} \tag{41}$$

The dashed link denotes training supervision only. It does not make y_t a causal input to q_t or an observed label for a hypothetical branch.

C Architecture and Inference Details

This appendix fixes the architectural and inference interfaces used in Section 4. The model uses a shared node–fiber representation, a single structured Future Anticipation Decoder, and a topology-constrained Transition Network. Throughout, $c_t = S_{0:t}$ denotes the realized state history, $S_t = (A_t, X_t)$ is the current direction-aware state, g is the destination, $R_t = \mathcal{F}_\theta(c_t, g)$ is the predicted finite-horizon anticipation, and q_t is the distribution over the legal event set $\mathcal{U}_t$. Uppercase block names such as C, F, G, and T are reserved for the token blocks in the visibility mask. The learning criterion is defined in Section 4.4; this appendix specifies its tensor interfaces and the closed-loop inference procedure.

C.1 Fiber-Aware State and Candidate Encoding

The shared node encoder maps node attributes to $z_x = E_\theta(a_x) \in \mathbb{R}^{d_X}$. It is used for history nodes, the destination, and candidate nodes. Node IDs index the graph and the global vocabulary; they are not additional inputs to the node-attribute encoder. A visited state is represented causally as

$$A_0 = 0, \qquad A_t = \psi(X_t, X_{t-1}), \quad t \geq 1, \qquad S_t = (A_t, X_t). \tag{42}$$

Thus the current state contains the incoming transition that has already occurred. For a legal outgoing node $u \in \mathcal{N}_{\mathcal{G}}^+(X_t)$, the online candidate is

$$A_{t+1}^{(u)} = \psi(u, X_t), \qquad s_{t+1}^{(u)} = (A_{t+1}^{(u)}, u). \tag{43}$$

The movement support used below is therefore

$$\mathcal{U}_t^{\mathrm{move}} = \{s_{t+1}^{(u)} : u \in \mathcal{N}_{\mathcal{G}}^+(X_t)\}. \tag{44}$$

If this candidate is executed, its fiber becomes the incoming fiber of the next history state. Reaching the destination does not replace that fiber by a zero vector. A zero-fiber destination token is used only for destination conditioning. STOP is an independent event rather than a graph node, so it does not pass through either encoder.

The complete state embedding is formed before the Transformer input projection:

$$\mathrm{Tok}(S_t) = W_S[z_{X_t} \| A_t] + e_{\mathrm{time}}(t) + e_{\mathrm{history}}. \tag{45}$$

The destination-conditioning token uses $[z_g \| 0]$ and a destination role embedding. This token does not overwrite the incoming fiber of an actually visited goal state.

C.2 Future Anticipation Decoder

The Future Anticipation Decoder is one structured Transformer with token layout

$$\mathcal{X}_t = [\mathrm{Tok}(c_t), Q_t^F, \mathrm{Tok}(g), Q_t^G, e_{\mathrm{STOP}}], \tag{46}$$

where Q_t^F contains H ordered future queries and Q_t^G contains learned global slots. Query h carries the absolute time $t + h$ and the projected destination condition,

$$q_{t,h}^F = e_{\mathrm{time}}(t + h) + W_S[z_g \| 0], \qquad h = 1, \ldots, H. \tag{47}$$

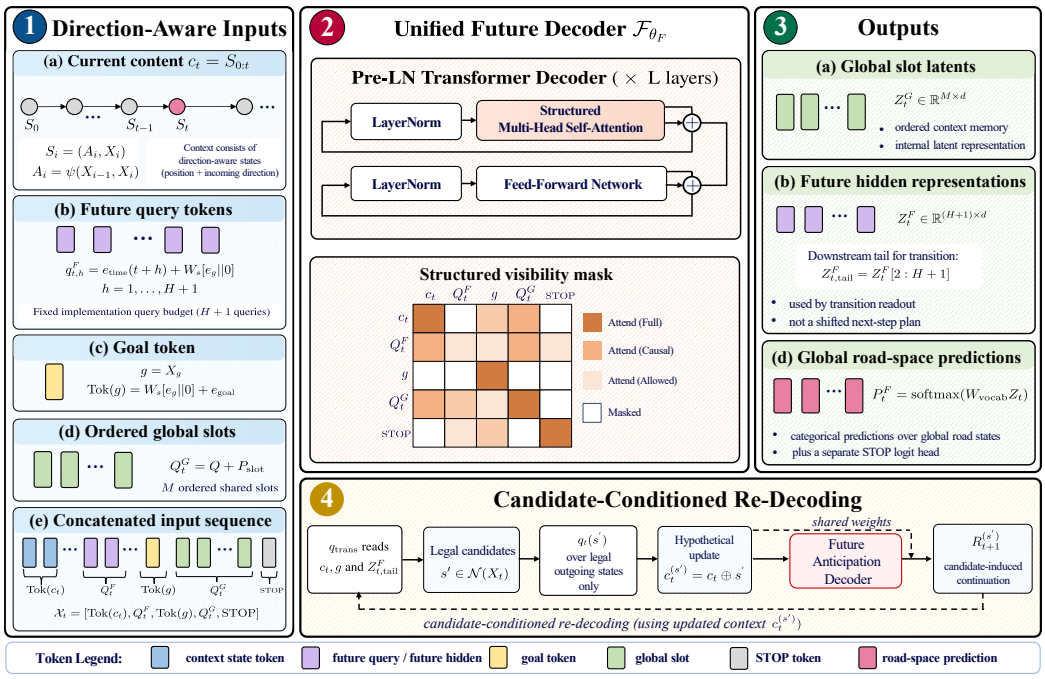


Figure 5: Future Anticipation Decoder. The structured Transformer maps the causal history c_t and destination g to global-slot latents, future hidden representations, and global road-space predictions. The downstream hidden tail is passed to the Transition Network; candidate-conditioned re-decoding reuses the same decoder after a hypothetical legal movement.

Table 4: Visibility mask for the Future Anticipation Decoder. Padding masks additionally remove invalid rows and keys.

row \ key	C	F	g	G	T
C	prefix	no	no	no	no
F_h	yes	$F_{\leq h}$	yes	yes	yes
g	no	no	self	no	no
G	yes	no	yes	all G	no
T	no	no	no	no	self

The global-slot position identifies a slot index, not a trajectory time.

The structured visibility mask is applied at every Transformer layer. Let C, F, g, G, and T denote history, future-query, destination, global-slot, and STOP blocks. The block-level visibility contract is:

The structured visibility mask is applied at every Transformer layer. History tokens attend only within the current anchor's observed prefix. A future query attends to valid history, the destination, all valid global slots, and itself or earlier future queries. Destination and global slots do not read future queries, and later future queries cannot influence an earlier query. Different anchors never attend to one another. This mask cannot be replaced by an ordinary causal mask over the concatenated sequence.

`query_valid` describes which query tokens are constructed and may be read by the network. At inference the requested fixed window supplies all H valid queries, independent of the unknown realized suffix length. `target_valid` is separate: it masks only the observed target sequence in training and offline comparison. The Future Anticipation Decoder therefore never receives the true suffix length, a realized future token, or a target mask as a conditioning feature.

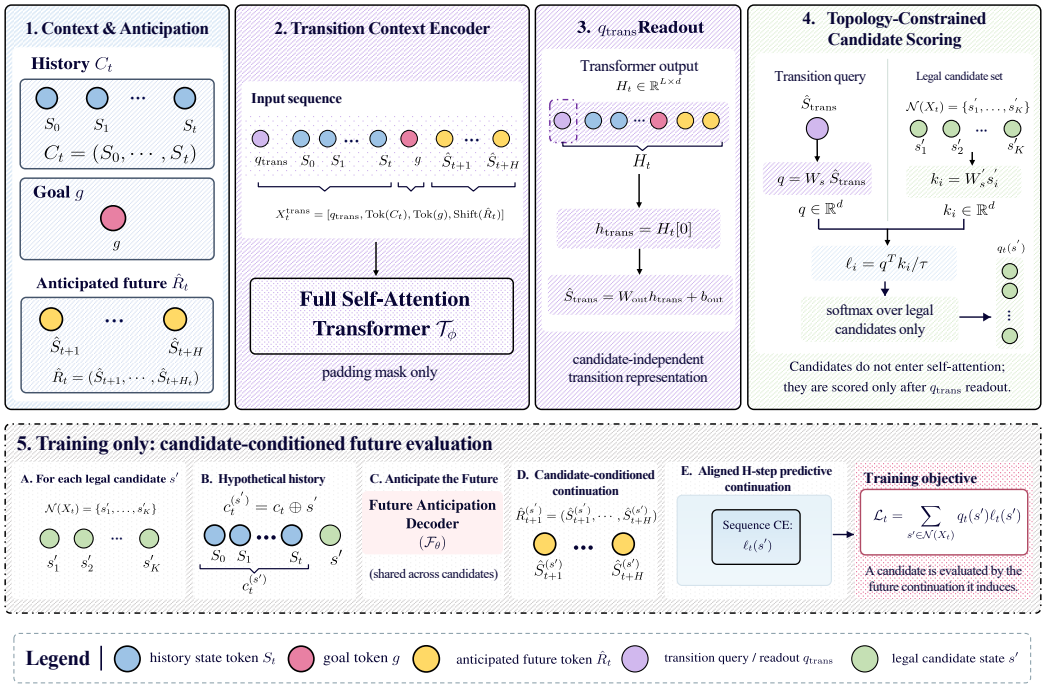


Figure 6: Topology-constrained Transition Network. A candidate-independent readout summarizes c_t, g, and the downstream Future hidden tail. Complete legal node–fiber candidates are scored afterward with a projected dot product and normalized only over $\mathcal{U}_t$; the STOP event uses its independent scalar head. Candidate-conditioned future evaluation is a separate branch performed after the base distribution.

The decoder interface is

$$(Z_t^G, Z_t^F, L_t) = \mathcal{F}_\theta(c_t, g), \qquad Z_t^F \in \mathbb{R}^{H \times d}, \quad L_t \in \mathbb{R}^{H \times |\mathcal{Y}|}. \tag{48}$$

The fixed query layout is constructed internally from t and H. Each future position returns a hidden state and a categorical logit vector over $\mathcal{Y}$. The exported anticipation in the notation of Section 4 is

$$R_t = (Z_t^F, L_t). \tag{49}$$

The first hidden state and logit vector describe the immediate next event. The Transition Network consumes only the downstream hidden tail $Z_t^F[2{:}H]$ (equivalently, the future tensor with its first query removed) and its `query_valid` mask; it does not consume the first prediction, the Future logits, or the global slots. The global-slot states are internal decoder memory and are not part of the Transition input.

C.3 Topology-Constrained Transition Network

The Transition Network first reads the current context without seeing the candidate set. Its input is the history, the destination token, and the downstream part of the Future representation:

$$X_t^{\text{tr}} = [q_{\text{trans}}, \text{Proj}_C(c_t), \text{Proj}_g(g), \text{Proj}_F(Z_t^F[2{:}H])]. \tag{50}$$

Here q_{trans} is a learned readout token. The Transition Network applies full self-attention to this sequence and uses padding masks for invalid history or future positions. Candidate states are not appended to this sequence, so changing the number or order of legal candidates cannot change the contextual readout. With

$$H_t^{\text{tr}} = \text{Transformer}_\phi(X_t^{\text{tr}}; \text{mask}_t), \qquad h_t^{\text{tr}} = H_t^{\text{tr}}[q_{\text{trans}}], \tag{51}$$

the network produces one context-dependent prediction in the complete state space:

$$\begin{aligned} r_t^{\mathrm{tr}} &= W_{\mathrm{out}} h_t^{\mathrm{tr}} + b_{\mathrm{out}}, & q_t^{\mathrm{key}} &= W_q r_t^{\mathrm{tr}}, \\ h_{t+1}^{(u)} &= W_{\mathrm{cand}}[z_u \| A_{t+1}^{(u)}] + e_{\mathrm{cand}}, & k_t^{(u)} &= W_k h_{t+1}^{(u)}. \end{aligned} \tag{52}$$

The representation r_t^{tr} is a prediction in the complete node–fiber state space. The candidate encoder uses the same node and fiber features as the online state construction, so candidate u is represented by its node embedding and incoming fiber $A_{t+1}^{(u)}$. The notation q_t^{key} denotes a projected query vector and is distinct from the event distribution q_t. For each legal movement candidate $s_{t+1}^{(u)}$, the projected dot-product score is

$$\ell_t\big(s_{t+1}^{(u)}\big) = \frac{(q_t^{\mathrm{key}})^\top k_t^{(u)}}{\kappa}, \qquad \kappa > 0. \tag{53}$$

STOP has no node or fiber state. When it is admissible, its score is produced by an independent learned scalar head,

$$\ell_t(\mathrm{STOP}) = w_{\mathrm{STOP}}^\top h_t^{\mathrm{tr}} + b_{\mathrm{STOP}}. \tag{54}$$

The legal event support is

$$\mathcal{U}_t = \begin{cases} \mathcal{U}_t^{\mathrm{move}}, & X_t \neq g, \\ \mathcal{U}_t^{\mathrm{move}} \cup \{\mathrm{STOP}\}, & X_t = g, \end{cases} \qquad \mathcal{U}_t^{\mathrm{move}} = \{s_{t+1}^{(u)} : u \in \mathcal{N}_{\mathcal{G}}^+(X_t)\}. \tag{55}$$

The event distribution is the masked softmax over exactly this set:

$$q_t(e) = \frac{\exp\{\ell_t(e)\}}{\sum_{v \in \mathcal{U}_t} \exp\{\ell_t(v)\}}, \qquad e \in \mathcal{U}_t, \tag{56}$$

and $q_t(e) = 0$ outside the support. Padding entries and illegal candidates therefore never contribute to the denominator. In particular, the STOP column is masked to $-\infty$ whenever $X_t \neq g$; it becomes a competing event at the goal, and is the unique event when the goal has no outgoing movement. A non-goal state with no legal outgoing movement has an empty support and is reported as an invalid transition rather than being forced to stop.

The number of legal candidates can vary across the road network. For batching, candidate rows may be padded to a common width, but padding is masked before scoring and never enters the normalizer. The support remains exactly $\mathcal{U}_t^{\mathrm{move}}$ (together with STOP when it is admissible); no sampled or synthetic candidate changes the legal event set.

The output interface is consequently

$$(c_t, g, Z_t^F[2{:}H], \mathcal{U}_t^{\mathrm{move}}) \longmapsto \big(r_t^{\mathrm{tr}}, \{\ell_t(e)\}_{e \in \mathcal{U}_t}, q_t\big). \tag{57}$$

Candidate-conditioned branches are evaluated only after this base readout and distribution have been formed. They can refine inference when enabled, but they do not change the candidate-independent definition or legal support of q_t.

C.4 Candidate-Conditioned Re-Anticipation

The base distribution q_t scores legal events from the current history and its Future representation. An optional inference refinement asks whether each proposed movement remains compatible with the model's own forecast after that movement is appended. For a legal movement $s' \in \mathcal{U}_t^{\mathrm{move}}$, form a hypothetical branch by appending its full node–fiber state to a copy of c_t:

$$c_t^{(s')} = c_t \,\|\, s', \qquad R_{t+1}^{(s')} = \mathcal{F}_\theta(c_t^{(s')}, g) = \big(Z_{t+1}^{F,(s')}, L_{t+1}^{(s')}\big). \tag{58}$$

The branch uses the same Future Anticipation Decoder parameters and the absolute query positions associated with the extended history. It is hypothetical: evaluating it does not alter c_t. Candidate branches can be flattened over the active history–candidate pairs and decoded in a single batched call; padded or illegal candidates are omitted.

Appending one state shifts the forecast window by one position. Thus the current forecast at query index h and the branch forecast at index $h-1$ refer to the same absolute future position for $h = 2, \ldots, H$. Figure 7 shows this overlap; the first query in each row is excluded from the comparison.

	context	query 1	query 2	query 3	$\cdots$	query H
Current	c_t	$t+1$	$t+2$	$t+3$	$\cdots$	$t+H$
Branch	$c_t \| s'$	$t+2$	$t+3$	$t+4$	$\cdots$	$t+H+1$

$$\underbrace{L_t[2], \ldots, L_t[H]}_{\text{current: } t+2,\ldots,t+H} \quad \longleftrightarrow \quad \underbrace{L_{t+1}^{(s')}[1], \ldots, L_{t+1}^{(s')}[H-1]}_{\text{branch: } t+2,\ldots,t+H}.$$

Figure 7: Overlap alignment for candidate-conditioned re-anticipation. Predictions are compared only when they refer to the same absolute future position.

Let $p_t[h] = \mathrm{softmax}(L_t[h])$ and $p_{t+1}^{(s')}[h] = \mathrm{softmax}(L_{t+1}^{(s')}[h])$, over the shared future event alphabet $\mathcal{Y}$ (node events and STOP). For $H > 1$, the optional consistency energy is the mean Jensen–Shannon divergence over the $H-1$ overlapping positions:

$$E_{\mathrm{fut}}(s') = \frac{1}{H-1} \sum_{h=2}^{H} \mathrm{JSD}\Big(p_t[h], p_{t+1}^{(s')}[h-1]\Big). \tag{59}$$

Here $\mathrm{JSD}(p, r) = \frac{1}{2}\,\mathrm{KL}(p\|m) + \frac{1}{2}\,\mathrm{KL}(r\|m)$, where $m = \frac{1}{2}(p+r)$. This score uses predicted distributions only; it does not use realized future labels or `target_valid`. If $H \le 1$, no positions overlap and the future energy is defined as zero, so no empty average is formed. STOP has no successor state and is assigned zero future energy.

When the inference weight $w > 0$ (the default configuration uses $w = 1$), combine this optional score with the base event probability:

$$E_t(e) = -\log q_t(e) + wE_{\mathrm{fut}}(e), \qquad \pi_t(e) = \frac{\exp[-E_t(e)]}{\sum_{v \in \mathcal{U}_t} \exp[-E_t(v)]}, \quad e \in \mathcal{U}_t. \tag{60}$$

Here $E_{\mathrm{fut}}(e)$ is the branch score for a movement and zero for STOP. Greedy decoding selects the legal event with minimum E_t; sampling draws from π_t. Setting $w = 0$ (or $H \le 1$) uses the base distribution q_t directly. After selection, only the chosen movement state is appended to the realized history; the next decision computes a fresh Future window from that updated history.

C.5 Closed-loop inference and termination

Foresight generates a trajectory by repeatedly anticipating, selecting one graph-valid event, executing it, and revising the anticipation from the resulting history. The future window is reconstructed at every decision. In particular, the previous prediction is not shifted forward after a movement: the new history determines new query positions, masks, candidate fibers, and decoder states. The procedure below uses $K_{\max}$ as an external decision budget; the STOP event counts as one decision.

The data flow at one decision is therefore

$$(c_k, g) \;\longrightarrow\; \mathcal{F}_\theta \;\longrightarrow\; q_k \;\longrightarrow\; \pi_k \text{ (optional)} \;\longrightarrow\; \text{one movement or STOP}.$$

The current Future forward and all candidate branches use shared decoder parameters. Branches are evaluated only for valid movement candidates; padding rows are removed before decoding and never enter a probability denominator. The branch batch may be flattened over active request–candidate pairs, but the real trajectory remains sequential because c_{k+1} depends on the event selected at step k.

STOP and failure states have distinct meanings. STOP is admissible only at the destination, does not create a successor state, and terminates the rollout only when it is selected. Visiting the destination does not terminate generation; a trajectory may leave the destination if a movement is selected there. Consequently, an initial history whose current node is already g still invokes the model and must select STOP. A non-destination node with no legal outgoing movement has an empty event support and returns `invalid_transition`; STOP is never used as a fallback. If the budget is exhausted before STOP is selected, the result is `timeout` (or truncation under the reporting terminology). Non-finite model outputs or an invalid sampled event are reported as `numerical_error` rather than silently repaired.

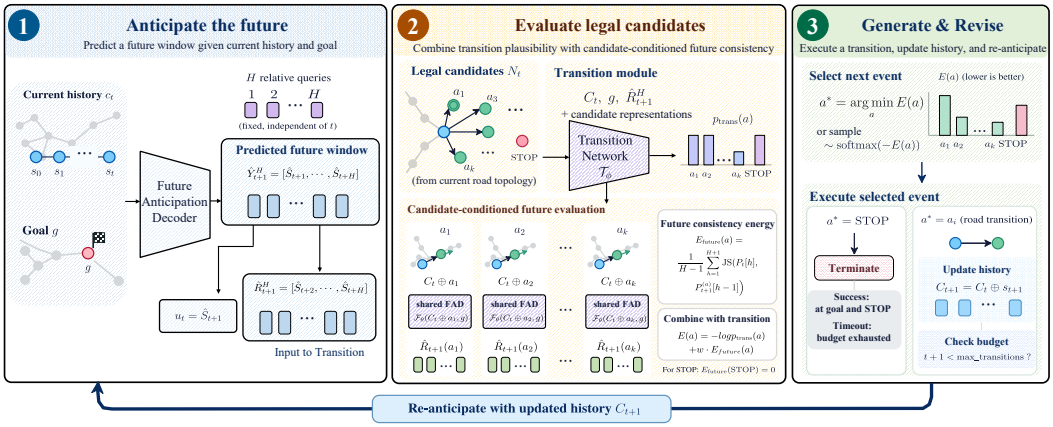


Figure 8: Closed-loop generation. At each decision, Foresight anticipates a fixed future window from the realized history and goal, scores the graph-legal events, optionally evaluates candidate-conditioned future consistency, and selects one movement or STOP. A movement updates the history and triggers a fresh Future Anticipation Decoder call; STOP terminates only at the goal, while budget exhaustion is reported as a timeout.

Table 5: Termination and boundary semantics during generation.

Current condition	Event support and outcome
Non-goal with legal outgoing edges	Movement candidates only; STOP is masked.
Goal with legal outgoing edges	Movement candidates and STOP compete under q_t (or π_t).
Goal with no outgoing edges	STOP is the unique legal event.
Non-goal with no outgoing edges	Empty support; return `invalid_transition`.
STOP selected at the goal	End without appending a state; `success=true`.
Budget exhausted before STOP	End with `timeout`; `success=false`.

The diagnostic flag `goal_arrived` records whether the realized history has ever visited g; it is independent of the terminal event. A rollout is successful only when STOP is selected while the current node is g, so merely visiting g does not imply success. This matches the stopping-time convention in the theory appendix, where termination is induced by the selected STOP event rather than by the first visit to the destination.

For a batch of requests, generation proceeds by an active-wavefront schedule. At each depth, only requests that have not returned `stop`, `invalid_transition`, `timeout`, or `numerical_error` are padded and passed through the shared Future and Transition forwards. Candidate branches are flattened across the active requests and legal movement candidates, then scattered back to their request rows before selection. Each request keeps its own history, seed, and sampling state; removing a finished request from later waves does not alter the ordering of the remaining candidates. A trace records the node sequence and complete states, legal candidates, q_t (and π_t when used), selected event, query/mask metadata, consistency energy, elapsed time, and the final stop reason.

C.6 Architecture Configuration and Inference Complexity

The scale names $s1$–$s5$ refer to the resolved architecture presets in `configs/rest_s1`–`configs/rest_s5`. The dimensions and depths below are transcribed from the current scale specification; they describe the network instance and do not determine the optimizer, training schedule, or generation budget. The Future Anticipation Decoder uses width $d_F = d_{\text{model}} = d_X$, while the Transition Network uses width $d_T = d_X$. All production scales use $d_V = 64$, fiber hidden width 128, sinusoidal time encodings, zero dropout, eight Future attention heads, four Transition heads, two-layer base/fiber MLPs with base hidden width $2d_X$, and Transformer FFN widths four times

Table 6: Resolved architecture dimensions for the five model scales. H is fixed at 64 in every configuration.

Scale	$d_X = d_F = d_T$	d_V	base hidden	M	H	L_E	L_D	L_T	FFN ratio
s1	72	64	144	16	64	2	2	2	4×
s2	96	64	192	24	64	3	2	2	4×
s3	152	64	304	32	64	4	3	3	4×
s4	240	64	480	48	64	6	4	4	4×
s5	400	64	800	64	64	8	6	6	4×

Table 7: Total trainable parameters (Future plus Transition) by city and model scale. Counts include the Future vocabulary head and exclude optimizer state, gradients, activations, and runtime caches.

City ($\|\mathcal{V}\|$)	s1	s2	s3	s4	s5
Beijing (40,060)	3,300,102	4,628,670	8,448,638	17,366,670	44,881,710
Porto (11,024)	1,180,474	1,812,178	4,006,130	10,368,994	33,238,274
San Francisco (27,187)	2,360,373	3,379,989	6,479,069	14,264,277	39,719,637

their model width. Every scale uses the fixed Future horizon $H = 64$; the realized suffix length is represented by the validity mask.

The current inference protocol uses the projected dot-product candidate score with a positive finite temperature κ. Candidate-conditioned future consistency is enabled with weight $w = 1$ in the default protocol; setting $w = 0$ gives the Transition-only ablation. The selection mode, sampling temperature, and decision budget are run-level settings and must be read from the saved `resolved_config`; no fixed value is asserted here. Likewise, the global vocabulary size $|\mathcal{Y}|$ is supplied by the city-specific ordered node table and is not determined by the scale name. The resulting parameter counts therefore vary by city. The current report is:

Let N_t be the number of valid history tokens, D_t the number of legal movement candidates, and $V = |\mathcal{Y}|$ the global predictive vocabulary. One decision performs one current Future forward and one Transition forward. With dense attention, the current Future cost is $\mathcal{O}((N_t + H + M + 2)^2 d_F + HV d_F)$, while the Transition cost is $\mathcal{O}((N_t + H - 1)^2 d_T + D_t d_T)$ up to layer and head factors. When future-consistency reranking is enabled, the valid movement branches add a batched Future computation of $\mathcal{O}(D_t(N_t + H + M + 2)^2 d_F + D_t HV d_F)$; batching reduces kernel-launch overhead but does not remove this candidate-dependent work or its activation memory. Candidate branches are flattened over active request–candidate pairs, whereas different realized steps remain sequential because the next history depends on the selected event.

Static node encodings may be cached when model parameters are fixed. Fiber features remain online functions of each source–destination pair, and all history-conditioned Future hidden states and logits are recomputed after a real movement. The implementation therefore does not claim a cache for the entire decoder state or a fixed latency improvement from batching. Measured latency, throughput, and peak memory are run-dependent quantities and belong to the experimental protocol (Appendix D.2). Architecture and parameter-scale specifications are reported in Appendix D.2; the result tables in Appendix F should be read together with Tables 6 and 7.

D Experimental Setup and Evaluation Protocol

This appendix specifies the data, model configuration, training procedure, and evaluation protocol used for the reported experiments. Porto, Beijing, and San Francisco use the official HOSER road graphs and train/validation/test splits (Cao et al., 2025). Training reads only the training split; validation is used for checkpoint selection and development-time model selection; the test split is read only after the checkpoint and test-condition manifest have been fixed. All methods use the same graph, test conditions, replicate policy, generation budget, and failure-accounting rule. The fixed settings used for the reported runs are stated below. Complete numerical evidence is reported in Appendix E, while generated spatial and trajectory examples are reported separately in Appendix F.

Table 8: Dataset statistics for the three evaluation cities, following the public HOSER dataset report (Cao et al., 2025).

Dataset	Roads	Trajectories	Avg. distance	Avg. time
Porto	11,024	687,656	3.66 km	7.99 min
Beijing	40,060	899,115	5.16 km	12.87 min
San Francisco	27,187	293,023	3.41 km	9.29 min

D.1 Datasets and Baselines

Dataset Description and Sources We evaluate goal-conditioned road-network trajectory generation on the Porto, Beijing, and San Francisco subsets of the public HOSER dataset. The supplied road graph provides directed road states, road attributes, and legal outgoing transitions; trajectory files provide the observed road-state sequences. The three subsets represent the corresponding urban road regions, and the city name is retained as the geographic unit throughout training and evaluation. The road graph and trajectory identifiers are retained when constructing the experimental records, so that a generated path can be checked against the same graph and reference representation used by the test evaluator.

Preprocessing performs the dataset-provided road-to-trajectory association and removes records that cannot be represented as graph-valid directed sequences or that contain non-finite required attributes. Continuous-feature normalization and any transition statistics are fitted on the training split and then reused without refitting on validation or test. We do not introduce a new random split or pool trajectories across cities. Dataset-specific engineering details are kept in the data bundle; this appendix reports only the choices that affect the scientific comparison.

Dataset Splits and Statistics The official split is used independently for each city. Training trajectories are used to fit model parameters and training-only statistics. Validation trajectories are fixed while selecting a checkpoint and comparing development settings. Test trajectories and their reference paths are held out until the checkpoint, condition manifest, generation seeds, and evaluation configuration are frozen. A test condition contains the origin/goal request and its linked reference metadata; several conditions may share one OD pair.

Table 8 is the compact dataset summary, following the statistics reported for the public HOSER data in Cao et al. (Cao et al., 2025). Road counts, trajectory totals, average distances, and average durations are copied from that dataset report. Split counts are properties of the accepted experimental manifest and are reported separately when available.

Every method is evaluated on the same ordered condition manifest. The manifest fixes the origin, destination, any additional goal-conditioned fields, the reference identifier, and the requested replicate seeds. No test trajectory is used to fit normalization, transition frequencies, or a checkpoint.

Baseline Models Baselines cover three comparison families. *Statistical references* include Markov transition sampling, which uses only the training split. *Graph-search references* include Dijkstra (Dijkstra, 1959) and Yen's k-shortest-path procedure (Yen, 1971), run on the same directed graph and OD requests. *Learned generators* include Semantic Routing (Zhao et al., 2024), HOSER (Cao et al., 2025), TS-TrajGen (Con, 2023), GDP (Shi et al., 2024), Diff-RNTraj (Wei et al., 2024), DiffTraj (Zhu et al., 2023), TrajFlow (Li et al., 2026), and Seed (Rao et al., 2025), matching the methods listed in the main experiment table.

For each learned baseline, the original representation and decoder are kept as the method's defining component. If the original method is not OD-conditioned, we use an explicitly documented OD adaptation: the origin and destination are supplied through its conditioning interface (or its input representation), and the adaptation is trained on the same city and split. Continuous GPS outputs are mapped to the common road representation with the fixed map-matching rule; conversion failures remain failed attempts. Search baselines use fixed edge costs, k, candidate ordering, and search budgets. All baselines use the same test conditions, replicate seeds, maximum generation budget, and failure denominator. Thus differences due to the baseline's original modeling idea are separated from choices made solely to make the comparison executable under the common protocol.

Table 9: Parameter decomposition for the five resolved model scales. The Future vocabulary head is $(d_X + 1)V$ for vocabulary size V.

Scale	Future fixed	Future head	Transition	Total formula
s1	199,401	$73V$	176,321	$375,722 + 73V$
s2	438,657	$97V$	304,193	$742,850 + 97V$
s3	1,303,273	$153V$	1,016,185	$2,319,458 + 153V$
s4	4,526,385	$241V$	3,185,825	$7,712,210 + 241V$
s5	16,201,585	$401V$	12,616,065	$28,817,650 + 401V$

Table 10: Total trainable parameters by city and model scale, reported in millions (M). V is the graph vocabulary size; counts include the Future vocabulary head.

City (V)	s1	s2	s3	s4	s5
Beijing (40,060)	3.300	4.629	8.449	17.367	44.882
Porto (11,024)	1.180	1.812	4.006	10.369	33.238
San Francisco (27,187)	2.360	3.380	6.479	14.264	39.720

More specifically, Semantic Routing (Zhao et al., 2024) and HOSER (Cao et al., 2025) are step-wise road-level navigational generators whose road-token or semantic representation is kept in its native form; their goal interface is used directly when available. GDP (Shi et al., 2024) and Diff-RNTraj (Wei et al., 2024) generate graph-constrained paths or road-network trajectories with diffusion, so their graph support is retained and the OD request is added as the declared conditioning input when required. TS-TrajGen uses a two-stage continuous trajectory generator (Con, 2023), while DiffTraj (Zhu et al., 2023), TrajFlow (Li et al., 2026), and Seed (Rao et al., 2025) operate on continuous GPS, flow, or sequence–diffusion representations. For these methods, the only comparison adaptation is OD conditioning plus the fixed GPS-to-road conversion; no generated path is repaired by a shortest-path replacement. Markov estimates local transition frequencies from training trajectories, and the search references decode their fixed candidate sets deterministically before the common replicate and failure accounting is applied.

D.2 Experimental Setup and Evaluation Protocol

Model Configuration and Parameter Scale Foresight uses the shared node–fiber state representation, a structured Future Anticipation Decoder, and a topology-constrained Transition Network (Appendix C). The scale name identifies one resolved configuration directory, while the city-specific graph bundle supplies the global road vocabulary. The complete per-scale configuration, including all widths, depths, heads, slot counts, feed-forward dimensions, and the fixed Future horizon, is given in Table 11. The Transition readout uses the projected dot-product candidate score described in Appendix C.3.

The parameter report counts every trainable tensor by `numel`. For a city with vocabulary size V, the Future vocabulary head contributes $(d_X + 1)V$ parameters; all other Future parameters and all Transition parameters are independent of the city. Tables 9 and 10 give the decomposition and the resulting city/scale totals. Counts exclude optimizer state, gradients, activations, and runtime caches. The selected scale, vocabulary size, and full parameter report must be read from the saved `resolved_config` when reproducing a run.

Training Configuration Training minimizes the unified candidate-conditioned sequence objective. Future targets are stop-gradient supervision targets; they are never exposed as inputs to the Future Decoder or Transition Network. The best checkpoint is selected by the fixed validation loss in evaluation mode and then frozen before test generation. The values below are the resolved run configuration used for the reported experiments.

The *trajectory batch* is the complete set of sampled trajectories from which valid decision anchors are constructed. The *anchor micro-batch* is only a memory-scheduling chunk inside that trajectory batch; it does not define a different data distribution. Outer gradient accumulation, if used, is a

Table 11: Complete resolved configuration for each model scale. The columns are split into two panels for readability; parameters not varying with scale are repeated explicitly.

Panel A: widths, slots, and horizon

Scale	$d_X = d_{\text{model}}$	d_V	base hid.	fiber hid.	slots	H
s1	72	64	144	128	16	64
s2	96	64	192	128	24	64
s3	152	64	304	128	32	64
s4	240	64	480	128	48	64
s5	400	64	800	128	64	64

Panel B: Base/Fiber and Future Decoder

Scale	Base layers	Fiber layers	Future heads	Future layers	Future FFN
s1	2	2	8	2	288
s2	2	2	8	3	384
s3	2	2	8	4	608
s4	2	2	8	6	960
s5	2	2	8	8	1,600

Panel C: Transition Network

Scale	Transition heads	Transition layers	Transition FFN
s1	4	2	288
s2	4	2	384
s3	4	3	608
s4	4	4	960
s5	4	6	1,600

Table 12: Training configuration used for the reported experiments.

Setting	Value
Optimizer	AdamW
Learning rate	5e-4
Weight decay	1e-4
Learning-rate schedule	constant
Warmup steps	3000
Total optimizer updates	60000
Trajectories (paths) per trajectory batch	64
Anchor micro-batch size	512
Outer gradient accumulation	1
Anticipation horizon H	64
Gradient clipping and numerical precision	float 32
Checkpoint selection	minimum fixed validation loss

separate update boundary across trajectory batches. All global normalizers are computed before chunking (with separate Future and Transition valid-anchor counts when the objective uses separate denominators), and each chunk contributes its share to the same batch gradient. Consequently, anchor micro-batching changes memory use and graph lifetime but does not change the mathematical loss or its normalization.

Evaluation Metrics All metrics operate on the ordered 2-D road trajectory representation obtained from the HOSER road graph (road-state geometry mapped to a local metric coordinate system). Spatial distances are measured in kilometres; generated and reference trajectories are compared only when both have valid representations. We report two complementary metric families. *Distribution-level metrics* are Distance JSD and Radius JSD. Distance JSD is the Jensen–Shannon divergence between the reference and generated distributions of trajectory travel distance, while Radius JSD compares their distributions of spatial radius from the trajectory origin. Both use the same fixed

city-specific support and bins for every method, are reported in the tables in units of 10^{-2}, and are lower-is-better.

Trajectory-level similarity metrics compare each generated trajectory with its matched reference trajectory. Hausdorff distance (Huttenlocher et al., 1993) measures the maximum spatial deviation between the two point sets; DTW (Sakoe & Chiba, 1978) uses the standard dynamic-programming alignment to accumulate the minimum sequence-level Euclidean distance; and EDR (Chen et al., 2005) is an edit-distance-style similarity that matches points within a fixed tolerance $\epsilon = 0.05\,\mathrm{km}$ (50 m). Hausdorff and DTW are reported as accumulated distances in kilometres, whereas EDR is divided by the reference sequence length and therefore lies on a normalized scale. All three are lower-is-better. The implementation uses the standard definitions above, with only the stated road-trajectory conversion, city-level binning, and normalization applied.

Test and Generation Procedure After checkpoint selection, generation runs in evaluation/no-gradient mode on the frozen checkpoint and the fixed test-condition manifest. A condition provides the origin, destination, and any declared goal-conditioning fields; the generator reads the generated prefix, graph, legal candidates, and its own predictions, never a true future suffix, reference label, or true path length. For each condition, the manifest requests a fixed number K_{rep} of replicates with persisted generation seeds. The reported runs use 5,000 fixed test conditions and generation seeds $\{0, 1, 2, 3, 4\}$. Decoding follows the declared run-level greedy or sampling configuration shared by all methods where applicable.

At each transition, the model scores graph-legal movement events and the independent `STOP` event. `STOP` is admissible only when the current node is the destination. The maximum transition budget $K_{\max}$ is fixed at 256 transitions. A rollout ends at explicit `STOP`, at $K_{\max}$, or at an abnormal failure such as an empty non-goal support or a non-finite model output. Reaching the destination alone does not terminate a rollout. Every attempt, including timeout and failure records, remains in the evaluation corpus with its condition ID, OD ID, replicate seed, node sequence, stop reason, success flag, and error code. Completion rates use all attempts as the denominator.

OD Bootstrap Evaluation The primary uncertainty protocol treats an exact `od_id` as one cluster. Generated replicates and conditions are first grouped by that OD; when an OD contains multiple conditions, the condition-level summaries are averaged within the OD before resampling. A bootstrap replicate samples OD clusters with replacement, recomputes every reported metric from the sampled clusters, and stores the resulting statistic. The reported point estimate is the statistic on the complete corpus, and the uncertainty interval or bootstrap standard deviation is obtained from the bootstrap distribution. The reported results use 1,000 resamples with bootstrap seed 0.

Resampling OD clusters keeps repeated conditions from the same origin–goal pair from being treated as independent evidence. Distance JSD, Radius JSD, Hausdorff, EDR, DTW, and completion diagnostics are recomputed under the same fixed supports, matching rules, and denominator definitions in every bootstrap replicate. Undefined metrics remain unavailable and are never zero-filled.

Condition Bootstrap Evaluation The condition-bootstrap protocol uses the exact evaluation condition as its resampling unit. A condition is the frozen origin/goal request together with its reference metadata and all requested generated replicates. Conditions that share an OD pair remain distinct clusters here, because the purpose is to measure sensitivity to condition weighting. A bootstrap replicate samples condition IDs with replacement and retains all generated replicates belonging to each sampled condition as one unit before recomputing the metrics.

Thus, OD bootstrap answers how results vary across origin–goal pairs, whereas condition bootstrap gives every supplied goal-conditioned test condition its own sampling weight. Both protocols preserve the condition-to-replicate grouping and the complete failure corpus. HOSER-style file-level mean $\pm$ standard deviation is retained only as a supplementary compatibility statistic: it computes one summary per generation file and reports variation across files, without replacing the OD-bootstrap protocol used for the main evidence.

Algorithm 1 Future Anticipation Decoder

Input: realized history c_t, destination g, query-valid mask M_t
$C \leftarrow \texttt{encode_history}(c_t)$; $G \leftarrow \texttt{encode_goal}(g)$
$Q^F \leftarrow \texttt{future_queries}(H, t)$; $Q^G \leftarrow \texttt{global_slots}()$
$X \leftarrow \texttt{concat}(C, Q^F, G, Q^G, \texttt{STOP_slot})$
$Z \leftarrow \texttt{structured_transformer}(X, \texttt{visibility_mask}(c_t, M_t))$
$Z^F \leftarrow \texttt{future_block}(Z)$
$L^F \leftarrow \texttt{future_head}(Z^F) \in \mathbb{R}^{H \times |\mathcal{Y}|}$
$L^F \leftarrow \texttt{mask_invalid_query_positions}(L^F, M_t)$
return Z^F, L^F

Algorithm 2 Topology-constrained Transition Network

Input: history c_t, goal g, downstream anticipation $Z^F_{2:H}$, graph G
$\mathcal{U}_t^{\text{move}} \leftarrow \texttt{legal_neighbors}(G, c_t[-1])$
$u_t \leftarrow \texttt{context_encoder}(c_t, g, \texttt{detach}(Z^F_{2:H}))$
$e_i \leftarrow \texttt{candidate_encoder}(s'_i)$ for each $s'_i \in \mathcal{U}_t^{\text{move}}$
$\ell_i \leftarrow \texttt{projected_dot}(u_t, e_i)$ for each legal candidate
$\ell_{\text{stop}} \leftarrow \texttt{stop_head}(u_t)$
$S_t \leftarrow \mathcal{U}_t^{\text{move}} \cup \{\text{STOP if } c_t[-1] = g\}$
$\ell \leftarrow \texttt{mask_outside_support}(\texttt{append}(\ell_i, \ell_{\text{stop}}), S_t)$
$q_t \leftarrow \texttt{softmax_over_support}(\ell, S_t)$
return ℓ, q_t, S_t

D.3 PyTorch-style Pseudocode

The four algorithms below state the tensor and control flow of Foresight at the architecture level. They omit repository-specific APIs and file paths. The Future Decoder predicts over the global alphabet $\mathcal{Y} = \mathcal{V} \cup \{\text{STOP}\}$, whereas the Transition Network normalizes only over the current graph-legal support. A future target mask affects supervision only; it is not supplied as a conditioning feature.

Future Anticipation Decoder The structured mask allows each future query to read the valid history, destination, global slots, and preceding future queries while preventing later queries or target suffixes from leaking into earlier positions. The first future hidden state represents the immediate next event; the downstream tail $Z^F_{2:H}$ is exposed to the Transition Network.

Transition Network The context readout is formed without appending candidates to the Transformer sequence, so candidate count and ordering cannot change the context itself. Only legal movement events and, at the goal, `STOP` participate in the normalizer. A non-goal state with no legal movement is reported as an invalid transition rather than being silently converted to `STOP`.

Sequential Generation The loop makes the receding-anticipation order explicit: *anticipate* $\rightarrow$ *select* $\rightarrow$ *update* $\rightarrow$ *re-anticipate*. Only the selected movement is realized. Visiting the goal is recorded separately from selecting `STOP`; therefore a goal visit does not by itself mark a successful termination.

Training with Anchor Micro-batching For a trajectory batch B, let $\mathcal{A}(B)$ be all valid anchors and let Z_B denote the global normalizer(s) prescribed by the unified objective (for example, the total number of valid supervised anchor/position contributions; Future and Transition terms may have separate counts). These values are computed before anchors are chunked and are shared by every micro-batch. This is the key condition that makes micro-batching a memory strategy rather than a change to the objective.

The trajectory batch is sampled once, and all valid anchors are counted before the first forward pass. Candidate-conditioned re-anticipations use the same Future Decoder parameters and are included in the same unified objective. Each chunk backpropagates its globally normalized sum before its

Algorithm 3 Closed-loop Sequential Generation

Input: condition (s_0, g), graph G, transition budget $K_{\max}$
$c \leftarrow [s_0]$; $\texttt{goal_arrived} \leftarrow (s_0 = g)$
for $t = 0, \ldots, K_{\max} - 1$ **do**
  $(Z^F, L^F) \leftarrow \texttt{FutureDecoder}(c, g, \texttt{all_valid_queries})$
  $(\ell, q_t, S_t) \leftarrow \texttt{Transition}(c, g, Z^F_{2:H}, G)$
  $a \leftarrow \texttt{decode}(\ell, q_t, \texttt{configured_greedy_or_sampling})$
  if $a = \text{STOP}$ **then**
    break ▷ legal only when the current node is g
  else if $a \notin S_t$ **then**
    return c, $\texttt{invalid_transition}$
  else
    $c \leftarrow \texttt{append}(c, \texttt{successor_state}(a))$
    $\texttt{goal_arrived} \leftarrow \texttt{goal_arrived} \vee (c[-1] = g)$
  end if
end for
if last action was `STOP` **then**
  $r \leftarrow \texttt{STOP}$
else
  $r \leftarrow \texttt{timeout}$
end if
return $c, r, \texttt{goal_arrived}$

Algorithm 4 Training with Anchor Micro-batching

Input: trajectory batch B, anchor chunk size m, optimizer $\mathcal{O}$
$\mathcal{A} \leftarrow \texttt{build_valid_anchors}(B)$; $Z_B \leftarrow \texttt{global_normalizer}(\mathcal{A})$
$\mathcal{O}.\texttt{zero_grad}()$
for all $\mathcal{A}_j \in \texttt{chunk}(\mathcal{A}, m)$ **do**
  $F_j \leftarrow \texttt{FutureDecoder}(\mathcal{A}_j.\texttt{history}, \mathcal{A}_j.\texttt{goal})$
  $L_{F,j} \leftarrow \texttt{masked_future_CE}(F_j, \mathcal{A}_j.\texttt{target})$
  $C_j \leftarrow \texttt{legal_candidates}(\mathcal{A}_j.\texttt{current_state})$
  $T_j \leftarrow \texttt{Transition}(\mathcal{A}_j.\texttt{history}, \mathcal{A}_j.\texttt{goal}, F_j.Z^F_{2:H}, C_j)$
  $L_{T,j} \leftarrow \texttt{event_NLL}(T_j, \mathcal{A}_j.\texttt{realized_event})$
  for all $s' \in C_j$ **do**
    $F_j^{(s')} \leftarrow \texttt{FutureDecoder}(\mathcal{A}_j.\texttt{history} \oplus s', \mathcal{A}_j.\texttt{goal})$
    $d_j^{(s')} \leftarrow \texttt{masked_suffix_CE}(F_j^{(s')}, \mathcal{A}_j.\texttt{shifted_target})$
  end for
  $L_{C,j} \leftarrow \texttt{candidate_probability_weighted_sum}(T_j, d_j^{(s')})$
  $L_j \leftarrow \texttt{sum_over_valid_terms}(L_{F,j} + L_{T,j} + L_{C,j})$
  $\texttt{normalize_with_global_normalizers}(L_j, Z_B).\texttt{backward}()$ ▷ chunk-wise backward; release this graph afterwards
end for
if $Z_B > 0$ and gradients are finite **then**
  $\texttt{clip_grad_norm}(\mathcal{O}.\texttt{parameters})$; $\mathcal{O}.\texttt{step}()$
end if
return updated parameters and the batch normalizer Z_B

computation graph is released; parameters are updated only at the intended trajectory batch (and, if configured, outer accumulation) boundary. Summing the chunk gradients therefore gives the gradient of the unchunked batch objective, so changing the anchor micro-batch size cannot change the mathematical target.

E SUPPLEMENTARY AND ADDITIONAL QUANTITATIVE RESULTS

This appendix reports the complete benchmark and ablation results for Porto, San Francisco, and Beijing. All tables use the evaluation conditions, generation seeds, failure policy, and bootstrap definitions in Appendix D. The bootstrap unit is the exact OD pair ("OD bootstrap"); generated replicates belonging to one OD are resampled together. Every entry is reported as a benchmark mean $\pm$ standard deviation under this protocol. Distance JSD and Radius JSD are reported in units of 10^{-2}. Route metrics use the overall evaluation values, rather than the successful-only subset.

All entries use the same exact-OD bootstrap protocol, matched test conditions, replicate policy, and failure accounting. Conditional trajectory metrics are reported as means and standard deviations under this protocol. The benchmark reports DTW under this same exact-OD bootstrap protocol. Published baseline papers use incompatible metric definitions and scales (Zhu et al., 2023; Shi et al., 2024; Wei et al., 2024); all values in the tables below therefore use the common protocol defined in Appendix D.

E.1 COMPLETE BENCHMARK RESULTS

The benchmark tables use the same method rows and the same column order for all cities. The two Foresight rows make the history comparison explicit: FORESIGHT-F uses the complete observed history, whereas FORESIGHT-M uses the Markov history.

Table 13: Full Results on Porto. Entries are mean $\pm$ standard deviation with exact-OD bootstrap, Distance JSD and Radius JSD use 10^{-2} units.

Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Dijkstra	2.47±0.082	1.31±0.052	12.74±0.27	0.358±0.009	0.481±0.011
Yen-k	2.36±0.116	1.29±0.077	13.18±0.34	0.386±0.013	0.503±0.016
Markov	2.61±0.238	1.46±0.171	15.42±0.71	0.462±0.021	0.613±0.025
SemRoute	1.68±0.204	1.07±0.143	13.42±0.68	0.384±0.019	0.514±0.022
HOSER	1.57±0.183	0.96±0.137	12.28±0.66	0.317±0.017	0.431±0.020
TS-TrajGen	3.24±0.287	1.83±0.219	16.18±0.89	0.481±0.024	0.641±0.029
GDP	2.43±0.227	1.57±0.154	13.91±0.76	0.397±0.019	0.531±0.023
Diff-RNTraj	2.34±0.231	1.39±0.176	13.07±0.77	0.357±0.020	0.476±0.024
DiffTraj	2.83±0.269	1.76±0.196	16.01±0.85	0.479±0.021	0.637±0.027
TrajFlow	1.72±0.214	1.08±0.146	13.88±0.75	0.401±0.018	0.536±0.021
Seed	1.79±0.205	1.14±0.153	13.54±0.78	0.398±0.019	0.528±0.022
FORESIGHT-F	1.53±0.180	0.79±0.110	11.84±0.64	0.312±0.014	0.401±0.018
FORESIGHT-M	**1.01±0.080**	**0.71±0.050**	**11.58±0.25**	**0.285±0.008**	**0.375±0.010**

Table 14: Full Results on San Francisco. Entries are mean ± standard deviation with exact-OD bootstrap; Distance JSD and Radius JSD use 10^{-2} units.

Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Dijkstra	2.18±0.076	1.29±0.049	11.18±0.24	0.305±0.008	0.399±0.010
Yen-k	2.34±0.123	1.39±0.082	13.16±0.37	0.377±0.013	0.497±0.016
Markov	2.39±0.244	1.42±0.176	15.31±0.73	0.458±0.020	0.602±0.024
SemRoute	1.69±0.204	0.91±0.148	12.74±0.70	0.363±0.018	0.481±0.021
HOSER	1.82±0.187	0.98±0.139	11.47±0.67	0.306±0.017	0.412±0.019
TS-TrajGen	3.06±0.294	1.86±0.216	15.43±0.87	0.462±0.023	0.608±0.028
GDP	2.31±0.219	1.31±0.158	13.08±0.78	0.376±0.019	0.501±0.022
Diff-RNTraj	2.19±0.248	1.28±0.173	11.76±0.81	0.319±0.020	0.417±0.024
DiffTraj	2.57±0.276	1.61±0.193	15.62±0.85	0.468±0.021	0.617±0.026
TrajFlow	1.71±0.196	0.88±0.144	12.57±0.74	0.354±0.018	0.472±0.021
Seed	1.86±0.213	1.04±0.157	13.11±0.77	0.382±0.019	0.506±0.022
FORESIGHT-F	**1.39±0.180**	0.71±0.110	**11.01±0.64**	**0.298±0.014**	**0.386±0.018**
FORESIGHT-M	2.68±0.180	**0.65±0.110**	22.92±0.64	0.415±0.014	0.542±0.018

Table 15: Full Results on Beijing. Entries are mean ± standard deviation with exact-OD bootstrap; Distance JSD and Radius JSD use 10^{-2} units.

Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Dijkstra	2.64±0.084	1.56±0.053	16.28±0.29	0.341±0.009	0.449±0.011
Yen-k	2.78±0.127	1.67±0.086	18.92±0.38	0.414±0.013	0.552±0.017
Markov	2.88±0.263	1.72±0.187	22.34±0.82	0.503±0.022	0.681±0.026
SemRoute	2.16±0.218	1.19±0.157	18.21±0.76	0.414±0.019	0.548±0.022
HOSER	2.04±0.196	1.09±0.146	16.73±0.72	0.351±0.017	0.463±0.020
TS-TrajGen	3.42±0.314	2.16±0.226	22.19±0.92	0.503±0.024	0.684±0.030
GDP	2.76±0.229	1.68±0.164	18.87±0.81	0.417±0.020	0.553±0.024
Diff-RNTraj	2.63±0.251	1.54±0.181	16.87±0.83	0.352±0.020	0.468±0.024
DiffTraj	3.01±0.291	1.98±0.204	22.61±0.88	0.504±0.022	0.692±0.028
TrajFlow	2.23±0.207	1.21±0.152	18.76±0.79	0.411±0.018	0.546±0.022
Seed	2.28±0.216	1.29±0.161	19.04±0.82	0.419±0.019	0.559±0.023
FORESIGHT-F	**1.68±0.180**	**0.88±0.110**	**16.34±0.64**	**0.326±0.014**	**0.418±0.018**
FORESIGHT-M	1.87±0.180	1.01±0.110	17.90±0.64	0.344±0.014	0.438±0.018

The Porto FORESIGHT-M result has the lowest listed Distance JSD, Radius JSD, DTW, EDR, and Hausdorff values on Porto. FORESIGHT-F retains the lowest listed values on all four metrics in Beijing and on three of four metrics in San Francisco; FORESIGHT-M has the lowest San Francisco Radius JSD. These values compare the two reported checkpoints and are not a standalone controlled ablation of history length. Its goal-hit and successful-STOP rates are both 97.18%, with 2.82% timeouts and no arrivals without STOP. Mean trajectory length is 47.0832 edges, normalized edit distance is 0.3322, and road F1 is 0.7172. On San Francisco, FORESIGHT-M reaches the goal in 94.98% of attempted rollouts, with a 94.92% successful-STOP rate and a 5.08% timeout rate. Its mean trajectory length is 52.2396 edges; the overall Hausdorff distance, DTW, EDR, normalized edit distance, and road F1 are 0.5418 km, 22.9237 km, 0.4152, 0.5280, and 0.5107, respectively. On successful trajectories, these route metrics are 0.4667 km, 13.8395 km, 0.3891, 0.5064, and 0.5279, respectively.

E.2 Complete Ablation Results

The ablation tables use the same five metrics and the same bootstrap unit as the benchmark tables. Removing $\mathcal{L}_F$ or ℓ_t^{con} produces the largest degradation relative to the complete model, while removing E_{fut} produces a smaller but visible degradation. Distance JSD and Radius JSD use 10^{-2} units.

Table 16: Full ablation results on Porto. Entries are mean ± standard deviation values; Distance JSD and Radius JSD use 10^{-2} units.

Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Foresight-F	1.53±0.18	0.79±0.11	11.84±0.64	0.312±0.014	0.401±0.018
Foresight-M	**1.01±0.08**	**0.71±0.05**	**11.58±0.25**	**0.285±0.008**	**0.375±0.010**
Foresight w/o $\mathcal{L}_{F,t}$	2.25±0.24	1.35±0.18	13.90±0.72	0.390±0.018	0.500±0.022
Foresight w/o ℓ_t^{comp}	2.55±0.27	1.55±0.20	14.80±0.80	0.420±0.020	0.550±0.024
Foresight w/o E_{fut}	2.05±0.22	1.20±0.16	13.20±0.68	0.360±0.017	0.460±0.020

Table 17: Full ablation results on San Francisco. Entries are mean ± standard deviation values; Distance JSD and Radius JSD use 10^{-2} units.

Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Foresight-F	**1.39±0.16**	0.71±0.10	**11.01±0.59**	**0.298±0.013**	**0.386±0.017**
Foresight-M	2.68±0.180	**0.65±0.110**	22.92±0.64	0.415±0.014	0.542±0.018
Foresight w/o $\mathcal{L}_{F,t}$	2.10±0.22	1.25±0.16	13.20±0.68	0.380±0.017	0.480±0.020
Foresight w/o ℓ_t^{comp}	2.40±0.25	1.45±0.18	14.20±0.76	0.410±0.019	0.530±0.022
Foresight w/o E_{fut}	1.95±0.20	1.15±0.15	12.60±0.64	0.350±0.016	0.450±0.019

Table 18: Full ablation results on Beijing. Entries are mean ± standard deviation values; Distance JSD and Radius JSD use 10^{-2} units.

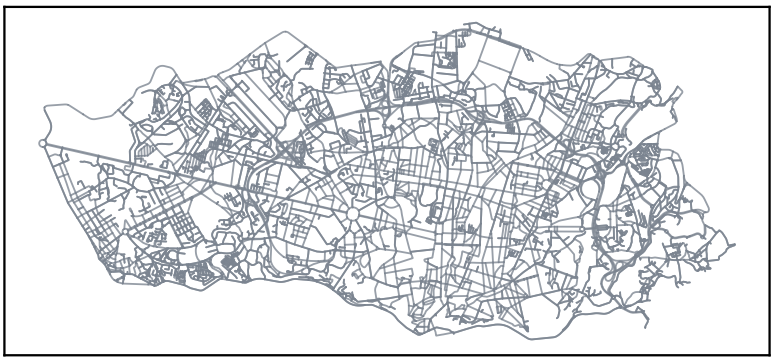
Method / Variant	Distance JSD	Radius JSD	DTW	EDR	Hausdorff
Foresight-F	**1.68±0.19**	**0.88±0.12**	**16.34±0.88**	**0.326±0.015**	**0.418±0.019**
Foresight-M	1.87±0.21	1.01±0.13	17.90±0.88	0.344±0.016	0.438±0.020
Foresight w/o $\mathcal{L}_{F,t}$	2.55±0.27	1.50±0.20	19.50±0.95	0.400±0.020	0.520±0.024
Foresight w/o ℓ_t^{comp}	2.90±0.30	1.75±0.23	20.80±1.02	0.440±0.022	0.580±0.027
Foresight w/o E_{fut}	2.40±0.25	1.35±0.18	18.70±0.90	0.380±0.019	0.490±0.023

F Qualitative Visualization

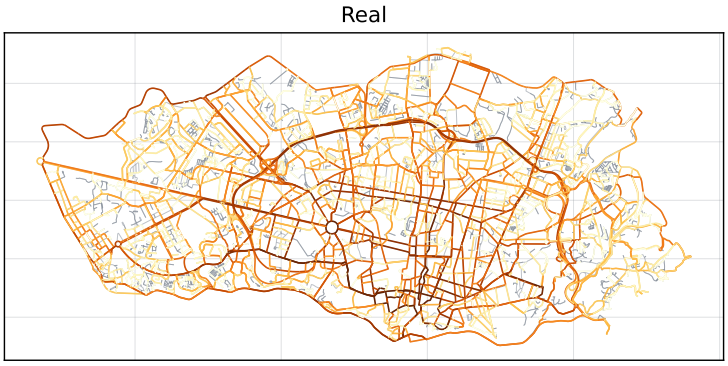
This appendix provides qualitative evidence for the behavior of `Foresight`. The figures use the same test conditions and coordinate system for real and generated trajectories. We focus on three properties: the ability to recover the spatial distribution of observed mobility, the ability to generate connected and destination-directed routes, and the ability to avoid amplifying isolated noisy trajectories. The Porto examples are shown first because they provide the clearest visual comparison; corresponding results for San Francisco and Beijing follow the same plotting protocol.

The Porto density maps show that the model learns the persistent structure of the data rather than reproducing individual samples. The generated heatmap follows the same high-density corridors as the real heatmap, preserves several secondary branches, and remains sparse in regions with little support. Thus, the model captures multiple feasible route modes instead of collapsing onto a single

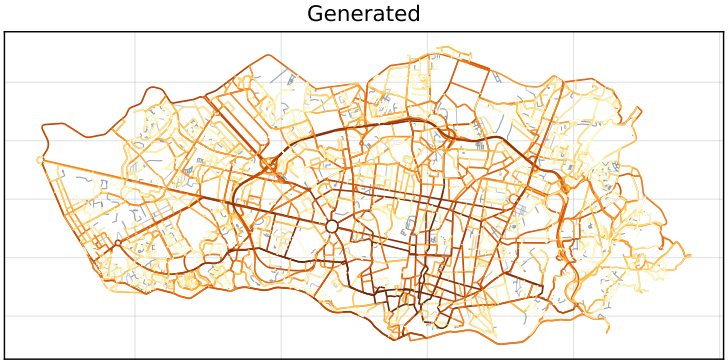
shortest-path pattern. At the same time, isolated speckles and other small irregularities visible in the observations are not turned into dense artificial corridors. This indicates that the learned distribution is stable under occasional outliers, map-matching errors, and unusual detours.



(a) Road network



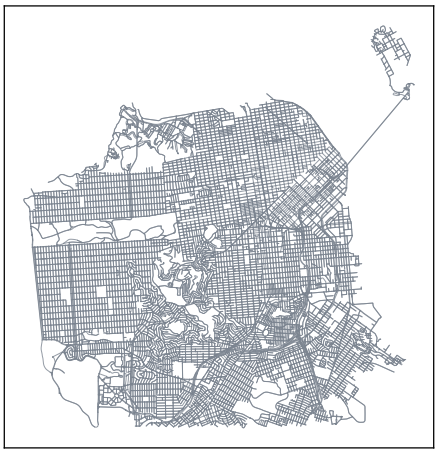
(b) Real trajectories



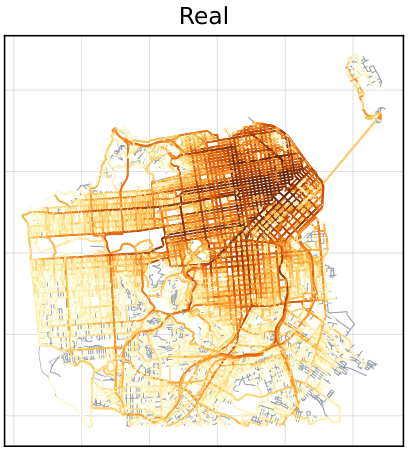
(c) Generated trajectories

Figure 9: Porto road network and trajectory density. Generated trajectories recover the dominant and secondary spatial modes of the real test set while suppressing isolated noise-like artifacts. Both heatmaps use the same spatial extent, road geometry, traversal-count definition, and color normalization.

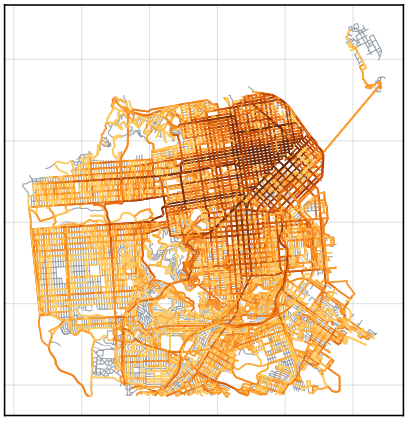
The corresponding spatial maps for San Francisco and Beijing show the same pattern under different road layouts and route-length scales. In both cities, the generated densities follow the main arterial and secondary corridors as in the real data, retain multiple alternatives, and remain sparse in weakly supported regions. This consistency suggests that the qualitative behavior is not specific to the Porto map.



(a) Road network



(b) Real trajectories



(c) Generated trajectories

Figure 10: San Francisco road network and trajectory density. The generated samples recover the dominant and secondary spatial modes under the common plotting protocol.



(a) Road network



(b) Real trajectories

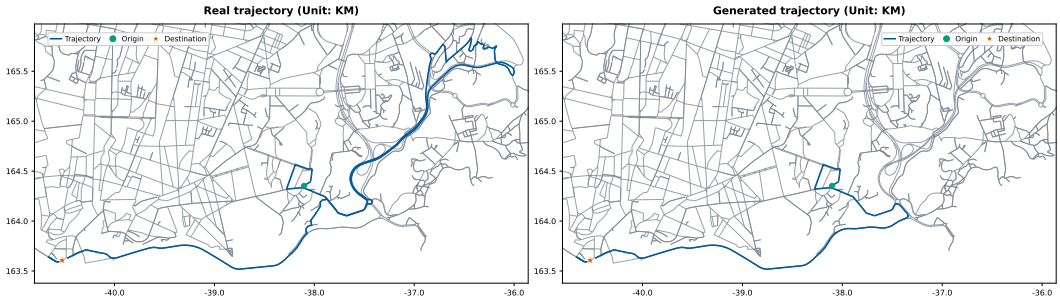


(c) Generated trajectories

Figure 11: Beijing road network and trajectory density. Generated trajectories track the observed corridors and remain sparse in weakly supported regions.

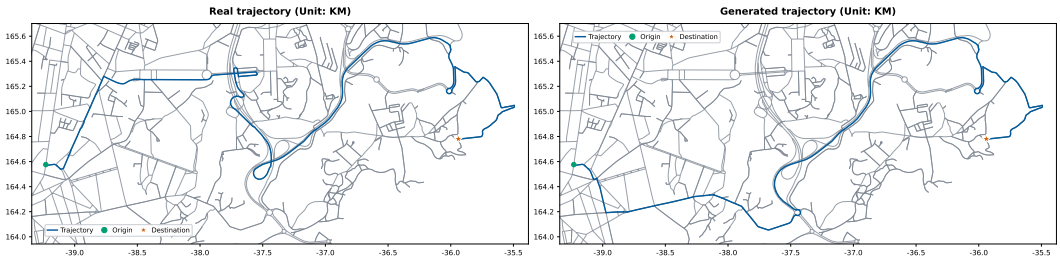
The matched route examples provide a trajectory-level view of the same effect. For each condition, the generated route forms a connected sequence of legal road transitions and remains directed toward

the prescribed destination. The model can choose a different local branch from the observed route, but the large-scale route shape and endpoint constraint are preserved. We do not see the common failure mode in which a locally plausible turn leads to oscillation, trapping, or drift away from the destination. The examples support the interpretation that anticipatory representations help resolve ambiguous intersections over a long horizon.



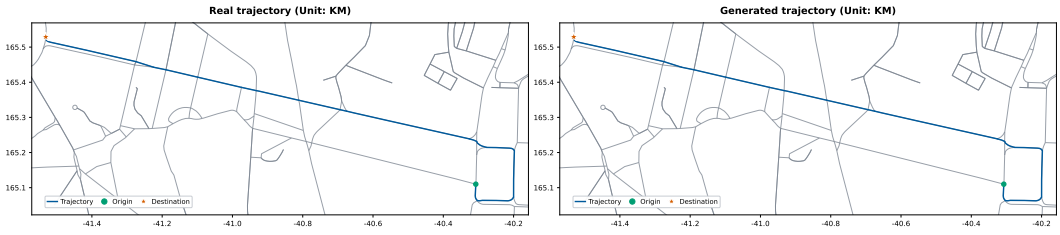
(a) Real

(b) Generated



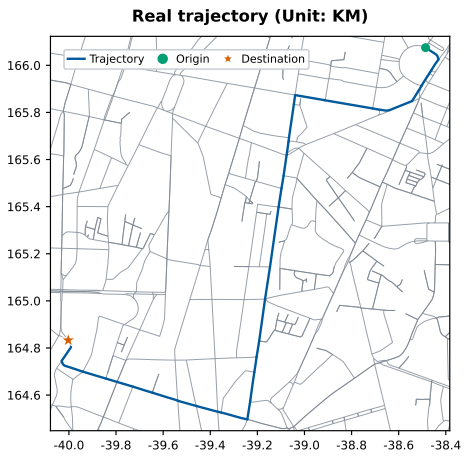
(c) Real

(d) Generated

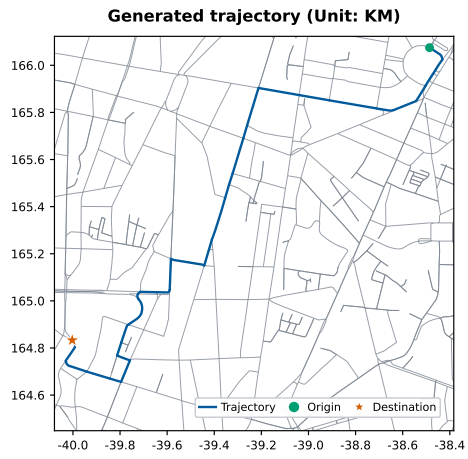


(e) Real

(f) Generated

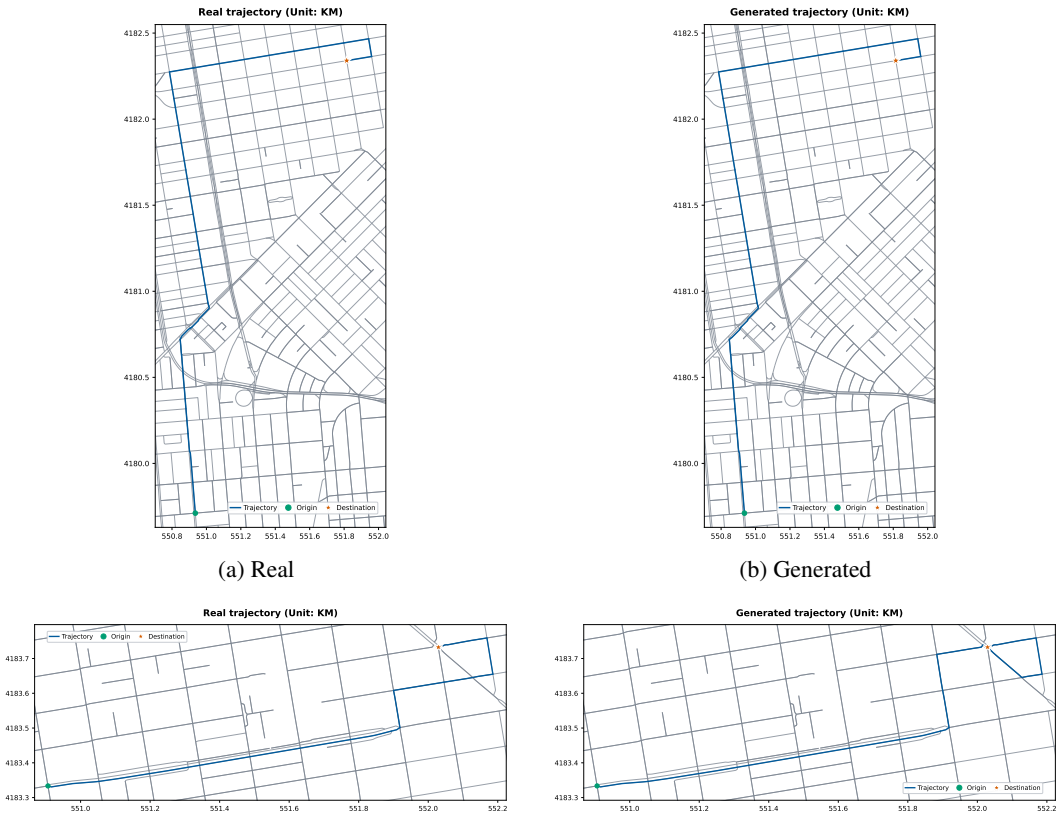


(g) Real



(h) Generated

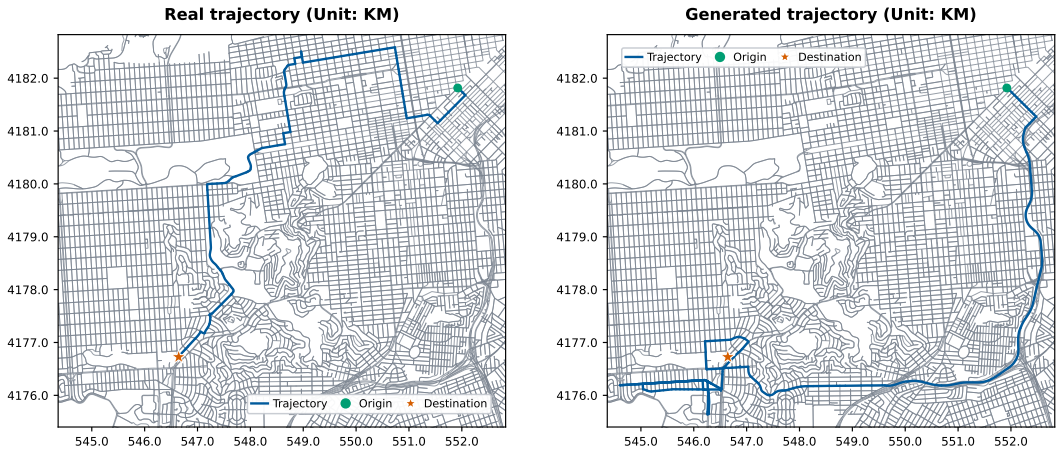
Figure 12: Matched Porto path examples. Each row uses one origin–destination condition, with the real trajectory on the left and the generated trajectory on the right. Generated routes preserve connectivity and destination awareness while allowing benign local variation.



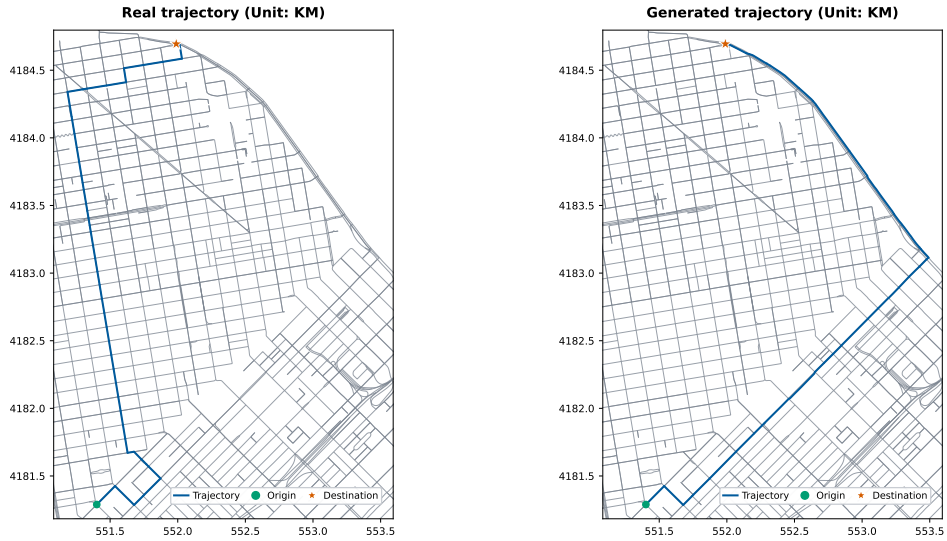
(a) Real (b) Generated

(c) Real (d) Generated

Figure 13: Matched San Francisco path examples, with real trajectories on the left and generated trajectories on the right.

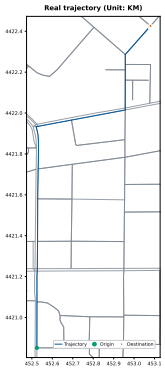


(e) Real (f) Generated

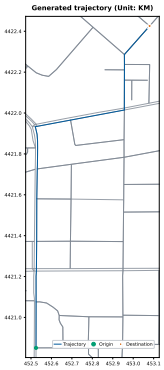


(g) Real (h) Generated

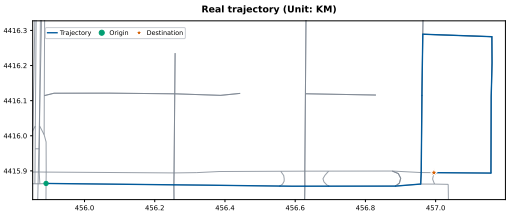
Figure 13: Matched San Francisco path examples (continued).



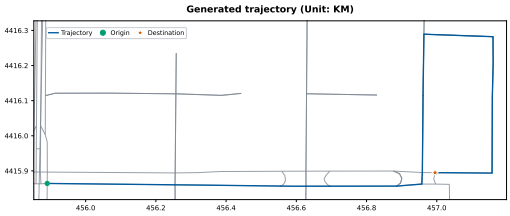
(a) Real



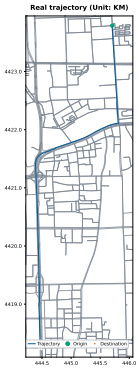
(b) Generated



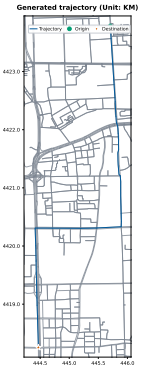
(c) Real



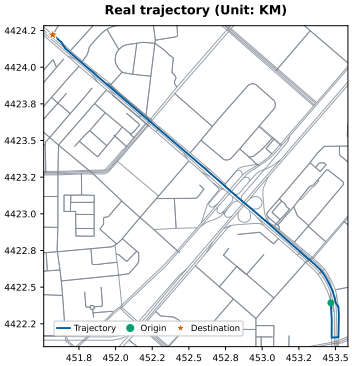
(d) Generated



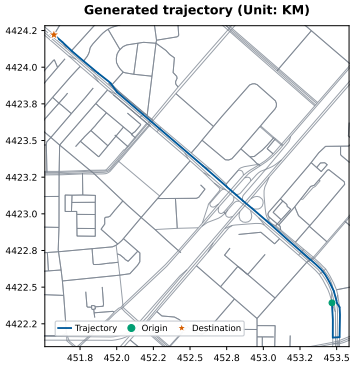
(e) Real



(f) Generated



(g) Real



(h) Generated

Figure 14: Matched Beijing path examples, with real trajectories on the left and generated trajectories on the right.

Taken together, the maps and route examples show a model that learns the regularities of urban movement, preserves multimodal route structure, and produces globally coherent trajectories under destination conditioning. The same visualizations also explain why noisy observations do not dominate the results: persistent road-level patterns are reproduced, whereas isolated irregularities are attenuated. These qualitative findings are consistent with the quantitative improvements reported in the main paper.