TEMPT: TRAJECTORY MIXING DECOMPOSITION FOR LOW-RESOURCE GPS TRAJECTORY PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Trajectory prediction plays a pivotal role in diverse applications, including intelligent transportation, urban planning, and autonomous navigation. However, prevailing trajectory prediction methods, particularly Transformer-based approaches demand substantial computational resources, making them unsuitable for resource-constrained environments like edge devices, mobile platforms, and embedded systems. To address this challenge, we propose TEMPT (Trajectory Mixing Decomposition), a lightweight yet highly accurate framework for GPS trajectory prediction. TEMPT employs a decomposition-inspired stacked architecture with lightweight MLP-Mixer blocks to extract multi-scale trajectory components, capturing both coarse and fine-grained patterns. Additionally, a dynamic weighted aggregation mechanism fuses adaptive patch representations into a unified embedding space, further enabling efficient handling of variable-length inputs. Extensive experiments on four trajectory datasets show that TEMPT achieves state-of-the-art accuracy while reducing computational costs by 56% compared to conventional approaches. It further sustains real-time single-trajectory inference on one CPU thread, so a dedicated accelerator is not required at deployment time. The source code and trained models will be made publicly available.

1 INTRODUCTION

Trajectory prediction, which predicts the future routes for moving vehicles, is a fundamental task with significant importance in modern intelligent systems (Shi et al., 2023; Gu et al., 2023; Gao et al., 2024; Li et al., 2025). With the widespread adoption of GPS-enabled devices, accurate GPS trajectory prediction has become essential for a wide range of applications, including autonomous navigation systems (Wen et al., 2025), urban traffic analysis and planning (Li et al., 2024), and traffic flow management (Guan et al., 2024). Moreover, it supports personalized location-based services, improving user experiences in areas like ride-hailing and delivery logistics (Gu et al., 2024).

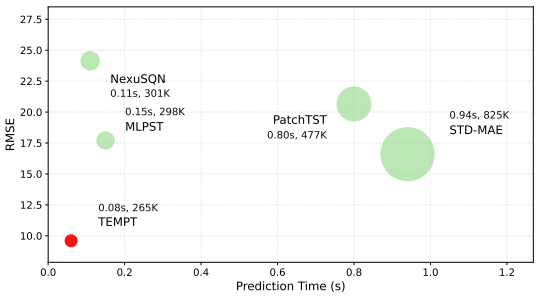


Figure 1: Balancing model efficiency and accuracy.

For GPS trajectory prediction, early pioneering efforts primarily relied on statistical techniques; however, their limited scalability and inability to model intricate patterns prompted a shift toward deep learning. Recurrent neural networks (Zhou et al., 2018) and Transformer-based architectures (Liang et al., 2022; Zhou et al., 2023) have significantly enhanced the modeling of long-range spatiotemporal dependencies. Nevertheless, these models remain data-hungry and computationally demanding, making them unsuitable for resource-limited environments. In intelligent transportation, predicting a vehicle's own route in real time supports fleet dispatch and congestion relief (Qin et al., 2023), and the predicted route feeds the arrival-time estimates of navigation services (Wang et al., 2018). Mobile-edge vehicular and IoT deployments anticipate a vehicle's own future position to schedule computation offloading and handover within strict latency and energy budgets (Zhang et al., 2024). UAV navigation forecasts an aircraft's own flight path to support planning and safe operation on low-latency and power-limited

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.

platforms (Shukla et al., 2025; Jia et al., 2026). In each of these settings the device relies on its own GPS stream, which calls for lightweight trajectory prediction in a low-resource environment.

Despite significant progress in GPS trajectory prediction overall, two persistent challenges still hinder its deployment. **First**, modern architectures have grown increasingly complex and computationally intensive, making them impractical for edge deployment in low-resource settings such as on-board embedded systems. Achieving an optimal trade-off between computational efficiency and predictive accuracy thus remains a fundamental requirement for practical implementation. **Second**, efficient forecasting architectures typically bind their weights to a fixed token lattice, so every input has to be padded or truncated before it can be processed. Long trajectories then lose information through truncation while shorter ones incur redundant computation, and the natural variability of real-world trip lengths is absorbed by pre-processing instead of by the model itself. Addressing this limitation demands lightweight architectures capable of adapting to variable-length inputs by construction.

To address these challenges, we propose **T**raj**E**ctory **M**ixing decom**P**osi**T**ion (TEMPT), a lightweight and adaptive solution for low-resource trajectory prediction. Following the principle of time-series decomposition (Huang et al., 1998; Wen et al., 2019), we use lightweight multilayer perceptron (MLP) blocks for trajectory representation learning and predictive decoding, such that both training and inference can be greatly accelerated. TEMPT reconstructs the decomposition and mixing operations into a collaborative, hierarchical architecture specifically optimized for the unique spatiotemporal dynamics of GPS trajectories. Instead of applying a single fixed-window feature extractor, TEMPT solves a residual decomposition recursion in which each layer predicts from the current residual state and then updates that state. Additionally, we propose a dynamic weighted aggregation mechanism to project variable-cardinality patch representations into a unified representation space. This design makes variable length a structural property of the model instead of merely an input pre-processing step, enabling the model to handle trajectories of different effective history lengths while preserving high prediction quality. As shown in Figure 1, TEMPT attains the best accuracy at the lowest cost, which suits applications ranging from autonomous systems to smart city infrastructures.

The main contributions of this paper are threefold:

- We present a lightweight GPS trajectory prediction model that carries a recursive residual decomposition through a coordinate-coupled MLP-Mixer stack, so each layer reads only what the coarser scales left behind and the mixing preserves the heading and the speed that a displacement encodes.
- The proposed dynamic weighted aggregation mechanism handles variable effective history lengths through adaptive patch cardinalities, while also providing patch-level interpretability through the distribution of the learned weights over the patches of the history.
- Extensive experiments on four trajectory datasets demonstrate that TEMPT achieves state-of-the-art accuracy while reducing computational costs by 56% compared to conventional approaches, which supports deployment on platforms without a dedicated accelerator.

2 Related Work

Trajectory Prediction. Existing studies on trajectory prediction are broadly grouped by input modality. Vision-based methods (Gu et al., 2023; 2021; Liao et al., 2024) rely on perception signals, where image pixels provide scene context, and a large literature models social interaction and multimodal futures over rasterized or vectorized scenes (Alahi et al., 2016). Efficiency and observation length are addressed there by temporal pyramids (Liang et al., 2021), consistency networks (Li et al., 2021), distillation from shorter observations (Monti et al., 2022), length-shift adaptation (Xu & Fu, 2024), and lane-guided Transformers (Pu et al., 2024), yet all of them assume scene images, maps, or synchronized neighbour tracks and target server or vehicle grade compute. GPS-based methods (Liang & Zhao, 2021; Ip et al., 2021; Wang & Feng, 2024) instead model movement directly from 2D coordinates[1]. Among them, RNNs are effective for sequential modeling (Zhou et al., 2018) yet limited on long-range dependencies, and Transformers capture global relations (Zhou et al., 2021) at substantial cost (Zeng et al., 2023). Recent work adds vision-language next-location prediction (Zhang et al., 2025), hybrid fusion (Hao et al., 2025), distilled diffusion (Wang et al., 2025), and transferable vehicle trajectory learning (Wei et al., 2025), whereas TEMPT targets lightweight variable-length GPS forecasting under low-resource constraints.

[1]We use "trajectory" and "GPS trajectory" interchangeably in the sequel without loss of generality.

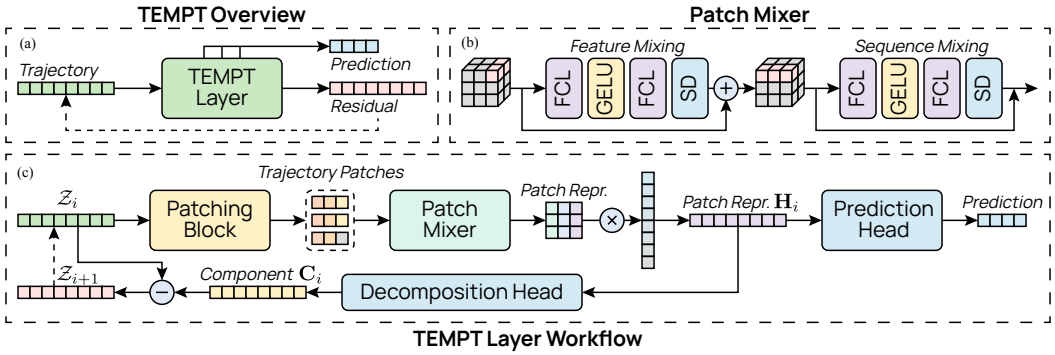


Figure 2: The proposed TEMPT framework. (a) TEMPT Overview. (b) Patch Mixer. (c) TEMPT Layer Workflow.

Low-resource Time-series Analysis. Time-series modeling has been widely studied with recurrent and convolutional networks (Bai et al., 2018; Liu et al., 2022a; Wu et al., 2023), Transformers (Zhou et al., 2022; Nie et al., 2023; Zhang & Yan, 2023), and hybrid variants (Wang et al., 2023). As architectures grow more complex their computational cost limits deployment under latency or resource constraints (Menghani, 2023), so low-resource time-series analysis seeks efficient alternatives (Kim et al., 2018; Deng et al., 2020; Prutsch et al., 2024). Pyraformer reduces Transformer complexity (Liu et al., 2022b), N-HiTS extends the doubly residual stacking of N-BEATS with hierarchical interpolation (Oreshkin et al., 2020; Challu et al., 2023), and DLinear shows that competitive accuracy is reachable with minimal architectural overhead (Zeng et al., 2023).

MLP-Mixer. MLP-based mixers replace attention or convolution with linear mixing along different dimensions (Tolstikhin et al., 2021; Zhong et al., 2024; Nie et al., 2024), capturing implicit dependencies while remaining efficient, which has made them competitive in vision (Tolstikhin et al., 2021) and strong time-series baselines (Zeng et al., 2023; Chen et al., 2023; Zhong et al., 2024). Standard mixers nevertheless assume a fixed window size and cannot handle variable-length GPS trajectories, whereas TEMPT adopts recursive trajectory mixing decomposition and dynamic weighted aggregation and so predicts efficiently without fixed-length inputs.

3 Preliminaries

In this section, we formally define the concepts related to trajectory prediction.

Definition 1 (GPS Trajectory). *A GPS trajectory is a sequence of consecutively sampled GPS points, denoted as* $\mathcal{X} = [p_1, \ldots, p_M]^\top$. *Each point* $p_i = [lon_i, lat_i, t_i]$, $i \in \{1, \ldots, M\}$, *records the longitude and latitude of an agent at sampling time* t_i. *The two coordinates are the axes of one physical position, so the displacement between consecutive points encodes heading and speed.*

In this paper, we consider trajectories sampled at evenly spaced intervals, for which t_i is implicit and the model operates on the coordinate channels alone. For non-evenly sampled trajectories, the sampling interval can be incorporated as an additional input feature.

Problem 1 (Trajectory Prediction). *Given a sequence of historical GPS points* $\mathcal{X}_{hist} = [p_1, \ldots, p_M]^\top$, *where* M *is the number of historical points, the goal of trajectory prediction is to forecast the future GPS points* $\mathcal{X}_{future} = [p_{M+1}, \ldots, p_{M+H}]^\top$, *where* H *is the number of points that make up the prediction horizon. Both sequences are sampled at the interval of the source dataset.*

Throughout this paper we study self-trajectory forecasting, in which a device predicts the future positions of the agent it is attached to from that agent's own coordinate stream. This is the signal that every GPS unit holds about itself, and training is carried out off-device so that the cost borne by the deployed system is inference cost alone. Map attributes and neighbouring tracks, where a deployment provides them, enter the same architecture unchanged, since they become additional channels for the feature mixing described below and a per-agent encoder can act as the motion backbone inside an interaction module that models the remaining agents.

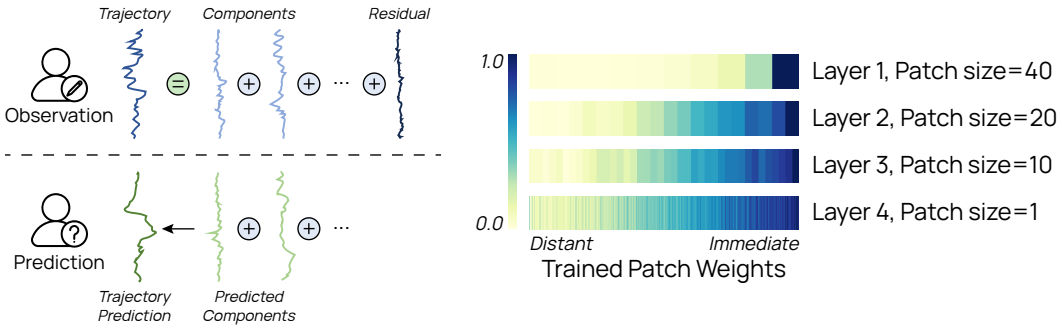


Figure 3: Illustrations of the proposed decomposition and aggregation mechanisms. (a) Trajectory prediction with recursive decomposition. (b) Trained patch weight values.

4 Trajectory Mixing Decomposition

This section elaborates on the technical details of our proposed trajectory prediction framework TEMPT as depicted in Figure 2. GPS trajectory forecasting operates on continuous physical coordinates where displacements across successive points jointly dictate velocity and heading. At the same time, vehicular movement exhibits a hierarchical structure ranging from global corridor progression to localized turns. TEMPT captures these spatial motion characteristics through a recursive residual dynamical architecture built with lightweight perceptron units. Successive layers process multi-scale residual updates, allowing each stage to focus directly on complementary motion components left by coarser scales. This formulation delivers strong expressive capacity with minimal computational overhead, satisfying the real-time constraints of resource-limited embedded processors.

The overall architecture conducts structural decomposition and future horizon forecasting concurrently over shared residual representations. Coordinate coupling is preserved along the spatial dimension, multi-scale patches separate motion patterns across temporal granularities, and dynamic aggregation maps variable history sequences into unified representations without fixing token lengths.

4.1 TEMPT Overview

TEMPT comprises a sequence of L TEMPT Layers designed to separate the input trajectory $\mathcal{X}$ into L structured trajectory components $\{\mathbf{C}_1, \ldots, \mathbf{C}_L\}$. Initialized with the full observed trajectory $\mathcal{Z}_0 = \mathcal{X}$, the recursive system updates the residual state at each layer by subtracting the motion component isolated at that stage,

$$\mathcal{Z}_i = \mathcal{Z}_{i-1} - \mathbf{C}_i, \quad i = 1, \ldots, L, \tag{1}$$

in which $\mathcal{Z}_i$ represents the residual motion state after the first i reconstructed components have been removed from the input $\mathcal{X}$. This recursive subtraction produces an additive decomposition of the original sequence,

$$\mathcal{X} = \sum_{i=1}^{L} \mathbf{C}_i + \mathbf{R}, \tag{2}$$

where $\mathbf{R} = \mathcal{Z}_L$ denotes the final residual trajectory remaining after L recursive stages. By guiding deeper layers toward progressively finer residual variations, the system establishes mutual complementarity across all extracted components, and Figure 3(a) depicts the whole process.

Following this residual state formulation, each layer transforms the incoming residual $\mathcal{Z}_{i-1}$ into temporal patches and applies mixing along the coordinate and the intra-patch axes to obtain a patch-resolved representation $\mathbf{M}_i$. A dynamic weighted aggregation operator then condenses this representation into a compact state $\mathbf{H}_i$. To complete residual updating and future forecasting, two separate parametric heads process these intermediate representations,

$$\begin{aligned} \mathbf{M}_i &= \textsc{Mix}_i(\mathcal{Z}_{i-1}), \qquad \mathbf{H}_i = \textsc{Agg}_i(\mathbf{M}_i), \\ \mathbf{C}_i &= \textsc{ReconHead}_i(\mathbf{M}_i), \qquad \mathbf{C}_i^{+} = \textsc{PredHead}_i(\mathbf{H}_i), \quad i = 1, \ldots, L. \end{aligned} \tag{3}$$

The reconstruction head generates the historical motion component $\mathbf{C}_i$ to update the residual state in equation 1, accommodating the length of $\mathcal{Z}_{i-1}$ for any input size. Concurrently, the prediction head

maps the pooled representation $\mathbf{H}_i$ directly to the future trajectory component $\mathbf{C}_i^+$ of fixed shape $H \times F$, achieving variable-length input processing alongside fixed-horizon output projection.

4.2 Trajectory Patching Block

Within layer i, the incoming residual sequence $\mathcal{Z}_{i-1}$ of length M is segmented into temporal patches using a layer-specific patch size P_i. This patching operation partitions the sequence along the temporal axis into non-overlapping segments with stride P_i, creating a patch tensor $\mathbf{B}_i$ with shape $N_i \times P_i \times F$, where $N_i = \lceil M/P_i \rceil$ and $F = 2$ indicates the longitude and latitude coordinates. When the sequence length is not a multiple of P_i, the trailing patch is completed to length P_i, accompanied by a binary validity mask of shape $N_i \times P_i$ indicating authentic observation positions. Inactive positions are set to zero during linear transformations and removed during final reconstruction, keeping boundary completions from affecting the decomposition.

This multi-scale patch design creates a motion semantic hierarchy. Coarse patches span broad temporal windows to represent macro corridor trends, while fine patches isolate sharp heading adjustments and intermittent stops. Because the patch count N_i adjusts dynamically to the input duration, network parameter shapes remain fully decoupled from sequence length constraints, allowing one model configuration to process diverse observation windows without external padding or retraining.

4.3 Feature and Sequence Mixing

Within the patch mixer block, the patch tensor $\mathbf{B}_i$ undergoes localized transformations implemented entirely with feed-forward perceptrons. Each mixing stage stacks two linear projections separated by a GELU non-linear activation, stabilized by a residual connection and regularized by stochastic depth to enhance training convergence and generalization.

Feature mixing and sequence mixing perform dedicated physical modeling roles across the orthogonal dimensions of $\mathbf{B}_i$. Feature mixing applies linear transformations along the coordinate dimension F across all temporal indices and patches. Operating jointly on the spatial channels preserves the physical coupling between longitude and latitude, maintaining instantaneous heading and velocity indicators. Sequence mixing applies linear transformations along the intra-patch temporal dimension P_i, sharing parameters across all N_i patches. This intra-patch operation models local kinematics within each segment, processing patches as uniform batch elements.

The binary validity mask is applied element-wise at the input and after each of the two residual connections, ensuring that inactive positions remain zero across all transformations. Both mixing operations preserve the tensor shape $N_i \times P_i \times F$, with weight dimensions determined solely by P_i and F. Consequently, the model parameter shapes remain invariant to the patch count N_i.

The mixing operations therefore extract local spatial structure and short-range temporal structure at low overhead, yet the shape $N_i \times P_i \times F$ still carries the variable patch cardinality N_i, and a fixed-size MLP over the patch axis, as standard MLP-Mixers apply, would restore a fixed input-length assumption at exactly this point. To integrate information across patches without imposing fixed length constraints, TEMPT combines the mixed representations $\mathbf{M}_i$ through dynamic weighted aggregation,

$$\boldsymbol{\omega}_{i,1:N_i} = \mathrm{softmax}(\textsc{WeightNet}_i(\mathbf{M}_{i,1:N_i})), \quad \mathbf{H}_i = \sum_{n=1}^{N_i} \omega_{i,n} \mathbf{M}_{i,n}, \tag{4}$$

in which $\mathbf{M}_{i,n}$ represents the n-th patch representation, $\textsc{WeightNet}_i$ denotes a lightweight scoring function conditioning on the patch representation and its normalized index n/N_i, and $\boldsymbol{\omega}_{i,1:N_i}$ represents the resulting importance distribution. Because the scoring function evaluates each patch alongside its relative position, the operator normalizes across any number of incoming patches and the order in which the patches arrive enters the weight that each of them receives. Patch relevance therefore depends on the position, the effective history length, and the mixed representations together, which distinguishes TEMPT from static aggregation with weights that are fixed after training. The reconstruction head maps every patch back to $P_i \times F$, concatenates the outputs along time, and truncates the sequence to length M to execute the residual subtraction in equation 1. Because one score is produced per patch, the aggregation holds no parameter indexed by N_i, the softmax

normalizes over however many terms the input provides, and the contribution of every segment of the history is directly readable from the trained model at any patch count.

4.4 Prediction Projection Heads

Future coordinate forecasting is performed by dedicated projection heads that decode the aggregated representations into future coordinates in a non-autoregressive manner. Regardless of the input sequence length, the pooled representation $\mathbf{H}_i$ maintains an invariant shape of $P_i \times F$. A single linear projection head maps $\mathbf{H}_i$ directly onto the full forecast horizon $H \times F$, maintaining the length flexibility established in the aggregation stage.

The projected component $\mathbf{C}_i^+$ represents the future motion component associated with residual scale i. Symmetrically to the recursive decomposition of past observations in equation 2, the final predicted trajectory is obtained through layer-wise additive composition,

$$\hat{\mathcal{X}}_{M+1:M+H} = \sum_{i=1}^{L} \mathbf{C}_i^+. \tag{5}$$

Generating all future coordinates simultaneously in one forward step avoids sequential error accumulation across long forecast horizons and maintains constant computational latency during inference.

4.5 Decomposition and Training

Optimization of TEMPT combines direct future trajectory supervision with physical residual regularization, encouraging reconstructed components to isolate authentic kinematic behaviors while guiding the residual toward unstructured sensor noise. The future trajectory predictions $\hat{\mathcal{X}}_{M+1:M+H}$ are supervised against future ground truth coordinates via mean squared error, while the final historical residual $\mathbf{R} = [r_{m,f}] \in \mathbb{R}^{M \times F}$ obtained from equation 1 is penalized to promote the statistical properties of white measurement noise.

In physical tracking systems, isolating macroscopic routes and localized turns leaves high-frequency, uncorrelated measurement jitter. Following Zhong et al. (2024), we reflect this physical property with a residual loss that penalizes temporal autocorrelation and residual energy,

$$\mathcal{L}_{\text{res}} = \frac{1}{MF} \sum_{m=1}^{M} \sum_{f=1}^{F} r_{m,f}^2 + \frac{1}{(M-1)F} \sum_{j=1}^{M-1} \sum_{f=1}^{F} \left[\text{ReLU}\left(|a_{f,j}| - \frac{\alpha}{\sqrt{M}} \right) \right]^2, \tag{6}$$

$$a_{f,j} = \frac{\sum_{m=j+1}^{M} (r_{m,f} - \bar{r}_f)(r_{m-j,f} - \bar{r}_f)}{\sum_{m=1}^{M} (r_{m,f} - \bar{r}_f)^2}, \quad \bar{r}_f = \frac{1}{M} \sum_{m=1}^{M} r_{m,f}, \tag{7}$$

where $a_{f,j}$ computes the lag j sample autocorrelation of the residual along spatial dimension f. In equation 6, the first term minimizes residual magnitude, and the second term suppresses temporal autocorrelation exceeding the statistical threshold $\alpha/\sqrt{M}$. Unifying predictive accuracy with this structural constraint gives the overall objective,

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{res}}, \tag{8}$$

where $\mathcal{L}_{\text{pred}}$ represents the mean squared prediction loss, $\mathcal{L}_{\text{res}}$ enforces residual whiteness, and λ denotes the weight that balances the two terms.

5 Experiments

We present comprehensive experiments on three real and one simulated trajectory dataset to show the efficacy of TEMPT on variable-length GPS trajectory prediction across short-term and long-term horizons under four evaluation metrics.

5.1 Experiment Settings

To evaluate the effectiveness of TEMPT, we conducted experiments on two real-world taxi trajectory datasets collected from Chengdu and Xi'an, both released for research use as part of the public

Table 1: Trajectory prediction results. The best results are in **bold** and the second-best results are underlined. The last row reports the average error of TEMPT relative to each baseline over all settings, so TEMPT itself has no entry.

Models			TEMPT		NexuSQN		MSD-Mixer		TrajGDM		STD-MAE		PatchTST		MLPST		DLinear	
			RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Chengdu	Short	15s	**2.67**	**2.34**	11.58	9.77	6.68	5.81	5.49	5.03	3.60	3.17	15.46	11.46	12.74	10.08	23.86	16.02
		30s	**5.80**	**5.26**	21.92	16.39	8.86	7.93	7.45	6.89	6.99	6.04	14.45	12.11	17.31	14.17	26.86	19.65
		60s	**15.74**	**12.84**	39.34	30.17	19.31	15.60	17.72	14.06	17.00	16.38	31.63	25.42	37.40	29.13	43.19	32.83
	Long	30s	**14.85**	**13.37**	24.13	18.36	20.24	17.05	18.95	15.87	15.80	14.66	22.09	18.16	23.39	18.28	16.78	15.10
		60s	**31.62**	**27.59**	50.88	36.92	45.53	35.60	40.36	32.75	40.59	35.67	63.64	45.81	41.68	33.30	68.77	48.27
		120s	**68.09**	**54.39**	96.82	76.46	123.09	89.09	74.13	58.93	76.10	69.88	110.48	84.41	79.61	63.07	134.59	95.30
		300s	**255.87**	**198.45**	333.03	239.80	521.56	364.31	323.04	212.82	297.65	234.60	441.57	314.63	287.59	210.81	566.50	384.20
	All	15s	**2.73**	**2.40**	11.68	9.93	6.73	6.07	5.44	5.23	3.47	3.10	16.06	12.15	12.99	10.25	24.79	17.02
		30s	**5.89**	**5.34**	22.24	16.68	9.03	8.16	9.21	8.31	10.03	9.34	15.36	13.20	18.04	14.95	27.59	20.43
		60s	**15.88**	**13.06**	40.87	32.01	19.93	16.00	18.87	15.92	17.24	16.55	32.59	26.77	38.26	30.03	44.40	33.68
		120s	**65.40**	**50.77**	93.35	74.12	115.83	83.49	83.79	70.18	75.45	67.50	105.20	82.29	73.96	60.21	130.91	92.62
Xi'an	Short	15s	**3.13**	**2.53**	8.00	6.01	7.82	5.88	6.23	5.48	5.57	4.71	7.15	5.61	4.31	3.39	8.80	6.59
		30s	**9.60**	**8.33**	24.14	17.25	28.52	19.35	19.26	16.93	16.59	12.83	20.65	16.80	17.70	14.62	35.31	23.87
		60s	**24.48**	**21.67**	33.54	27.31	43.18	33.18	34.07	28.51	37.64	30.16	62.14	44.26	49.89	38.08	71.75	50.61
	Long	30s	**15.46**	**12.55**	33.57	22.43	19.27	14.20	20.47	16.33	21.86	18.31	24.43	18.46	21.69	15.89	26.87	19.07
		60s	**25.28**	**20.05**	41.47	31.97	34.81	26.67	30.75	23.68	32.59	26.01	35.94	28.73	36.21	29.70	36.43	29.89
		120s	**62.19**	**50.44**	101.87	68.01	81.83	63.82	72.30	59.52	73.26	61.70	78.98	62.33	72.52	57.40	91.16	65.46
		300s	**225.09**	**193.30**	367.49	280.26	350.29	278.60	300.28	250.94	298.45	267.89	292.76	250.35	415.77	295.41	400.15	290.72
	All	15s	**3.15**	**2.60**	8.43	6.35	8.11	6.34	7.44	5.80	6.22	5.35	7.47	5.92	4.60	3.62	9.36	6.94
		30s	**9.68**	**8.65**	24.38	17.89	28.80	20.14	19.73	16.72	20.26	17.14	21.19	17.72	18.11	15.04	36.60	24.77
		60s	**24.56**	**21.79**	33.83	27.51	43.54	33.49	36.79	34.91	41.50	37.24	63.31	45.09	50.31	38.99	72.87	51.17
		120s	**62.54**	**50.71**	102.00	68.87	82.25	64.29	72.10	56.53	70.35	55.62	79.70	63.43	73.36	58.31	92.37	66.18
Avg. Error ↓ (%)			-	-	49.49	43.67	40.71	36.41	29.32	28.36	25.12	27.37	48.46	45.14	40.60	36.90	57.85	52.14

DiDi Chuxing GAIA Open Dataset[2] and sampled every 3 seconds. Based on the trajectory data, we formalize three prediction scenarios: *Short* for predicting the trajectory in the next 15, 30, and 60 seconds ($\{5, 10, 20\}$-step prediction) with past 30 observations; *Long* for predicting the trajectory in the next 30, 60, 120, and 300 seconds ($\{10, 20, 40, 100\}$-step prediction) with past 400 observations. A third *All* scenario evaluates variable-length input trajectories and predicts 15, 30, 60, and 120 seconds into the future, with a random effective history between 40 and 400 observations drawn before each forward pass. Every model sees the same drawn histories under one shared generator, and each fixed-length baseline receives them under the three length-handling conventions its own literature uses, of which Table 1 reports the one that the validation partition selects for that baseline. For all scenarios we train TEMPT with $L = 4$ and patch sizes P_i of $\{30, 10, 5, 1\}$ for short-term prediction and $\{40, 20, 10, 1\}$ otherwise, on a single NVIDIA RTX 4090 GPU with 24 GB memory.

We compare TEMPT with seven baselines, namely MLPST (Zhang et al., 2023), a lightweight MLP-based spatiotemporal model, PatchTST (Nie et al., 2023), a Transformer for long-sequence time-series forecasting, STD-MAE (Gao et al., 2024), a spatial-temporal decoupled masked pre-training framework, TrajGDM (Chu et al., 2024), a Transformer diffusion model for trajectory generation, MSD-Mixer (Zhong et al., 2024), a multi-scale decomposition and mixing model, NexuSQN (Nie et al., 2024), an MLP-based spatial and temporal mixer for traffic forecasting, and DLinear (Zeng et al., 2023), a linear forecaster of minimal architectural overhead. Each baseline is adapted to the coordinate-only input that the task provides, and every baseline is tuned under the same protocol, so that each architecture enters the comparison at its strongest configuration.

Trajectories are split by trip into training, validation, and test partitions in a 7/1/2 ratio following collection time. Each GPS point is a longitude and latitude pair, and both the prediction and the ground truth are mapped to a shared standardized coordinate space through the global per-coordinate mean and standard deviation of the training partition, so every model is trained and evaluated in the same space and the reported numbers are directly comparable. Every baseline receives identical coordinate-only inputs, history and horizon settings, optimization budget, and random-search size over its own hyper-parameters. The primary evaluation metrics are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which aggregate the two coordinate channels of a step before averaging over the horizon, whereas ADE and FDE take the Euclidean distance between the predicted and the true position. Appendix A sets out the data preparation, the baseline adaptation, the search protocol, and the four metric definitions in full.

[2]https://outreach.didichuxing.com/

5.2 Overall Performance

Table 1 reports the overall performance. For short-term prediction, TEMPT achieves the lowest RMSE and MAE at every horizon, indicating its accuracy in capturing fine-grained temporal dependencies. Across all settings of Table 1, TEMPT reduces RMSE by 20.93% against the second-best entry of each row, attaining an MAE of 2.34 for 15-second prediction on Chengdu against 3.17 for the second-best STD-MAE, and an MAE of 8.33 for 30-second prediction on Xi'an against 12.83. The advantage also holds under the displacement metrics, where TEMPT attains the lowest ADE and FDE among all compared models under the same protocol.

The advantage persists in long-term prediction, with an MAE of 54.39 on Chengdu at the 2-minute horizon where the next-best TrajGDM records 58.93, and 50.44 on Xi'an at the same horizon. At the 5-minute horizon TEMPT still leads every baseline on both cities, with an RMSE of 255.87 on Chengdu and 225.09 on Xi'an, so the gain is retained as the prediction span grows by an order of magnitude.

Table 2: Model parameters and computational costs comparison.

Model	Parameters	FLOPs
PatchTST	477K	82.6M
MLPST	298K	33.9M
STD-MAE	825K	194.6M
TrajGDM	856K	224.7M
MSD-Mixer	412K	61.3M
NexuSQN	301K	42.0M
TEMPT (Ours)	**265K**	**14.8M**

The *All* scenario confirms the same ordering, for instance with an RMSE of 9.68 against 18.11 for the second-best MLPST at the 30-second horizon on Xi'an. Since it mixes effective histories between 40 and 400 observations within a single model, the result isolates the contribution of dynamic weighted aggregation and shows that variable length is absorbed by the architecture instead of by pre-processing.

Both efficient time-series forecasters show the cost of a fixed input length. DLinear maps one history length onto the horizon through a single weight matrix and PatchTST splits the history into a fixed number of patches, so the parameters of both are indexed by the input length, and both fall behind TEMPT at every setting while TEMPT serves them all with one set of weights. TEMPT is also the fastest of the three, needing 0.18 seconds per trajectory against 0.49 for DLinear, and Table 2 reports the cost of the six deep baselines. Table 2 shows that TEMPT attains this accuracy with the most economical parameter and FLOP profile of the six, and the ordering is stable under repetition, since every entry is a mean over ten shared seeds and every gap survives Holm correction over the full family of 154 paired comparisons. It also holds on Porto and on a UAV delivery dataset against TransferTraj (Wei et al., 2025), under splits that hold out whole drivers, a quarter of a year, and an entire city, and it carries over at low cost to calibrated intervals and multi-hypothesis futures. Appendix B reports those results, eight further reference points that range from a persistence rule to two published decomposition forecasters, and a bootstrap interval and an effect size for every difference against the strongest baseline.

5.3 Model Efficiency

Table 3 summarizes the average training time for one epoch and the average prediction time for one trajectory, showing a much lower footprint than all baselines and roughly 1.5 times the speed of NexuSQN and MSD-Mixer. The attention of the Transformer and the multi-step backward process of the diffusion model widen that margin against PatchTST and TrajGDM. Since training is performed off-device, deployment cost is inference cost, and the same measurement on a single CPU thread and on two earlier GPU generations shows that one CPU thread keeps pace with the accelerators and answers the 5-minute horizon in 0.169 seconds, well inside the interval at which a vehicle unit reports its own position. Figure 3(b) illustrates the learned weights of the dynamic weighted aggregation, which assign higher importance to more immediate patches, matching the motion inertia of an agent whose next position is most constrained by its recent displacement.

5.4 Ablation Study

The effectiveness of TEMPT is rooted in the proposed adaptive patching and dynamic weighted aggregation, and we implement four variants to validate them. TEMPT-U removes multi-scale patching and uses a patch size of 10 in every layer. TEMPT-S replaces dynamic weighted aggregation with static aggregation, which treats all patches as equally important and whose weights are frozen

Table 3: Time complexity study results. Training time is for one epoch, and prediction time is for one trajectory.

	Models	TEMPT		NexuSQN		MSD-Mixer		TrajGDM		STD-MAE		PatchTST		MLPST		DLinear	
		Train	Pred.	Train	Pred.	Train	Pred.	Train	Pred.	Train	Pred.	Train	Pred.	Train	Pred.	Train	Pred.
Pred. Len.	15s	**171s**	**0.06s**	273s	0.10s	250s	0.09s	471s	2.70s	360s	0.82s	425s	0.68s	261s	0.09s	381s	0.21s
	30s	**177s**	**0.08s**	281s	0.11s	257s	0.11s	477s	2.85s	377s	0.94s	451s	0.80s	273s	0.15s	389s	0.27s
	60s	**180s**	**0.09s**	286s	0.13s	261s	0.12s	490s	3.07s	380s	1.13s	465s	0.93s	279s	0.16s	396s	0.32s
	120s	**190s**	**0.11s**	289s	0.17s	265s	0.18s	504s	3.35s	411s	1.26s	490s	1.03s	289s	0.20s	407s	0.39s
	300s	**205s**	**0.18s**	311s	0.28s	288s	0.27s	558s	3.88s	465s	1.63s	510s	1.34s	304s	0.31s	423s	0.49s
	Speedup ↑	-	-	1.6×	1.5×	1.4×	1.5×	2.7×	30.5×	2.2×	11.1×	2.5×	9.2×	1.5×	1.7×	2.2×	3.2×

Table 4: Ablation study on the proposed components on Xi'an long-term 30-second prediction.

Model	TEMPT	TEMPT-U	TEMPT-S	TEMPT-I	TEMPT-D
RMSE	15.46	18.95	18.51	16.57	16.80
MAE	12.55	13.82	13.60	12.59	13.47

Table 5: Hyper-parameter study results on Xi'an long-term 30-second prediction.

Model	TEMPT	T-3A	T-3B	T-4A	T-4B	T-5A
RMSE	15.46	18.43	17.61	16.38	16.74	15.84
MAE	12.55	13.78	13.41	12.73	12.96	12.59

during training. TEMPT-I inverts the stack so that patch sizes are arranged in ascending order. TEMPT-D removes the decomposition, reducing the framework to a sequential network with no subtraction between the layers of the stack.

We adopt Xi'an long-term 30-second prediction to validate the proposed modules. Table 4 shows that varying patch sizes play a critical role, as evidenced by the drop in TEMPT-U, so multi-scale patching lets TEMPT represent short-term and long-term dependencies at the same time. A comparable drop is also observed in TEMPT-S, indicating the importance of dynamic weighted aggregation, and eliminating the decomposition in TEMPT-D limits the performance as well. The three mechanisms therefore contribute jointly, and no single one of them dominates. Reversing the stack in TEMPT-I also costs accuracy, so arranging the patch sizes from coarse to fine matters as well. The full configuration is the strongest of the five while retaining 265K parameters and 14.8M FLOPs, so the gain follows from the GPS-oriented organization of the three mechanisms inside the parameter and arithmetic budget that Table 2 reports.

5.5 TEMPT Structure Analysis

TEMPT uses L layers for multi-level decomposition, each with a different patch size P_i, and we define several variants with varying depths and patch sizes. The original TEMPT has four layers with $P_i = \{40, 20, 10, 1\}$, and it is compared against T-3A with $\{20, 10, 1\}$, T-3B with $\{100, 40, 20\}$, T-4A with $\{100, 40, 20, 10\}$, T-4B with $\{200, 100, 40, 20\}$, and T-5A with $\{100, 40, 20, 10, 1\}$, again on Xi'an long-term 30-second prediction.

From Table 5, the original configuration achieves the best performance, so four layers with patch sizes of 40, 20, 10, and 1 strike the best balance between accuracy and computational efficiency. The variants all remain ahead of the strongest baseline of Table 1 at this setting, and the fifth layer in T-5A brings no gain while enlarging the parameter count, so the reported depth is the economical choice. Appendix B further reports the sensitivity to the residual regularization, the sharpness of the component separation, and the effect that the patch order has on the accuracy of the aggregation.

6 Conclusion

In this paper, we introduced TEMPT (Trajectory Mixing Decomposition), a low-resource framework for GPS trajectory prediction on variable-length inputs. TEMPT carries a recursive residual decomposition through a lightweight MLP-Mixer stack, so every layer reads only what the coarser scales have left behind and the components it extracts are complementary by construction. Within a layer, feature mixing along the coordinate axis keeps intact the heading and the speed that a local displacement implies, sequence mixing shares one set of weights across every patch, and dynamic weighted aggregation scores each patch together with its normalized position, so the patch count enters no parameter shape and one set of weights serves every history length. The architecture therefore carries variable length as a structural property, reaches state-of-the-art accuracy on four datasets, and predicts in real time on a single CPU thread at 265K parameters and 14.8M FLOPs.

AI USE STATEMENT

Generative AI tools were used solely to polish the writing of this paper, improving the clarity and the readability of text that the authors had already written. The authors accept full responsibility for all content, code, and claims in this paper.

REFERENCES

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, 2016. 2

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018. 3, 16

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 6989–6997, 2023. 3, 16

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023. 3, 24

Chen Chu, Hengcai Zhang, Peixiao Wang, and Feng Lu. Simulating human mobility with a trajectory generation framework based on diffusion model. *International Journal of Geographical Information Science*, 38(5):847–878, 2024. 7

Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020. 3

Haotian Gao, Renhe Jiang, Zheng Dong, Jinliang Deng, Yuxin Ma, and Xuan Song. Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 3998–4006, 2024. 1, 7

Guobin Gu, Benxiao Lou, Dan Zhou, Xiang Wang, Jianqiu Chen, Tao Wang, Huan Xiong, and Yinong Liu. Road passenger load probability prediction and path optimization based on taxi trajectory big data. *Applied Sciences*, 14(17):7756, 2024. 1

Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15303–15312, 2021. 2

Junru Gu, Chenxu Hu, Tianyuan Zhang, Xuanyao Chen, Yilun Wang, Yue Wang, and Hang Zhao. ViP3D: End-to-end visual trajectory prediction via 3d agent queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5496–5506, 2023. 1, 2

Deyong Guan, Na Ren, Ke Wang, Qi Wang, and Hualong Zhang. Checkpoint data-driven gcn-gru vehicle trajectory and traffic flow prediction. *Scientific Reports*, 14(1):30409, 2024. 1

Wei Hao, Bin Chong, Ronghua Ji, and Chen Hou. User trajectory prediction unifying global and local temporal information. *arXiv preprint arXiv:2508.02161*, 2025. 2

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. 16

Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998. 2

André Ip, Luis Irio, and Rodolfo Oliveira. Vehicle trajectory prediction based on LSTM recurrent neural networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–5. IEEE, 2021. 2

Disen Jia, Jonathan Kua, and Xiao Liu. A lightweight LSTM model for flight trajectory prediction in autonomous UAVs. *Future Internet*, 18(1):4, 2026. doi: 10.3390/fi18010004. 2

Yunbin Kim, Jaewon Sa, Yongwha Chung, Daihee Park, and Sungju Lee. Resource-efficient pet dog sound events classification using lstm-fcn based on time-series data. *Sensors*, 18(11):4019, 2018. 3

Jiaxing Li, Chaozhe Jiang, Ke Han, Qing Yu, and Haoran Zhang. High-resolution spatiotemporal inference of urban road traffic emissions using taxi gps and multi-source urban features data: a case study in chengdu, china. *Urban Informatics*, 3(1):17, 2024. 1

Shijie Li, Yanying Zhou, Jinhui Yi, and Juergen Gall. Spatial-temporal consistency network for low-latency trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1940–1949, 2021. 2

Taoying Li, Xulei Sun, Quanyu Dong, and Ge Guo. DSTAnet: A trajectory distribution-aware spatio-temporal attention network for vehicle trajectory prediction. *IEEE Transactions on Vehicular Technology*, 74, 2025. 1

Rongqin Liang, Yuanman Li, Xia Li, Yi Tang, Jiantao Zhou, and Wenbin Zou. Temporal pyramid network for pedestrian trajectory prediction with multi-supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2029–2037, 2021. 2

Yuebing Liang and Zhan Zhao. NetTraj: A network-based vehicle trajectory prediction model with directional representation and spatiotemporal attention mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14470–14481, 2021. 2

Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. TrajFormer: Efficient trajectory classification with transformers. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1229–1237, 2022. 1

Haicheng Liao, Shangqian Liu, Yongkang Li, Zhenning Li, Chengyue Wang, Yunjian Li, Shengbo Eben Li, and Chengzhong Xu. Human observation-inspired trajectory prediction for autonomous driving in mixed-autonomy traffic environments. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14212–14219. IEEE, 2024. 2

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022a. 3

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022b. 3

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.*, 55(12), March 2023. 3

Alessio Monti, Angelo Porrello, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. How many observations are enough? knowledge distillation for trajectory forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6553–6562, 2022. 2

Tong Nie, Guoyang Qin, Lijun Sun, Wei Ma, Yu Mei, and Jian Sun. Contextualizing MLP-Mixers spatiotemporally for urban traffic data forecast at scale. *IEEE Transactions on Intelligent Transportation Systems*, 2024. 3, 7

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 7

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. 3, 16, 24

Alexander Prutsch, Horst Bischof, and Horst Possegger. Efficient motion prediction: A lightweight & accurate trajectory prediction model with fast training and inference speed. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9411–9417. IEEE, 2024. 3

Yuxuan Pu, Yang Li, Beihao Xia, Xiaowei Wang, Hongmao Qin, and Lei Zhu. A lightweight lane-guided vector transformer for multi-agent trajectory prediction in autonomous driving. In *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1245–1252. IEEE, 2024. 2

Xinyan Qin, Zhiheng Li, Kai Zhang, Feng Mao, and Xin Jin. Vehicle trajectory prediction via urban network modeling. *Sensors*, 23(10):4893, 2023. doi: 10.3390/s23104893. 1

Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5 (2):1696–1703, 2020. doi: 10.1109/LRA.2020.2969925. 16

Liushuai Shi, Le Wang, Sanping Zhou, and Gang Hua. Trajectory unified Transformer for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9675–9684, October 2023. 1

Pushpak Shukla, Shailendra Shukla, and Amit Kumar Singh. Trajectory-prediction techniques for unmanned aerial vehicles (UAVs): A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 27(3):1867–1910, 2025. 2

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: an all-mlp architecture for vision. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, 2021. 3

Bingzhang Wang, Kehua Chen, and Yinhai Wang. Collaborative-distilled diffusion models (CDDM) for accelerated and lightweight trajectory prediction. *arXiv preprint arXiv:2510.00627*, 2025. 2

Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11877. 1

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting, 2023. 3

Yilin Wang and Yiheng Feng. IDM-Follower: A model-informed deep learning method for car-following trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, 2024. 2

Tonglong Wei, Yan Lin, Zeyu Zhou, Haomin Wen, Jilin Hu, Shengnan Guo, Youfang Lin, Gao Cong, and Huaiyu Wan. TransferTraj: A vehicle trajectory learning model for region and task transferability. *arXiv preprint arXiv:2505.12672*, 2025. 2, 8, 18

Di Wen, Yu Wang, Zhigang Wu, Zhaocheng He, Zhe Wu, and Zheng Qingfang. A driving-style-adaptive framework for vehicle trajectory prediction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 1

Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. RobustSTL: A robust seasonal-trend decomposition algorithm for long time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5409–5416, July 2019. 2

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis, 2023. 3

Yi Xu and Yun Fu. Adapting to length shift: Flexilength network for trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15226–15237, 2024. 2, 24

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023. 2, 3, 7, 24

Ruixing Zhang, Yang Zhang, Tongyu Zhu, Leilei Sun, and Weifeng Lv. Eyes will shut: A vision-based next GPS location prediction model by reinforcement learning from visual map feedback. *arXiv preprint arXiv:2507.18661*, 2025. 2

Ting Zhang, Bo Yang, Zhiwen Yu, Xuelin Cao, George C. Alexandropoulos, Yan Zhang, and Chau Yuen. Computation pre-offloading for mec-enabled vehicular networks via trajectory prediction. *arXiv preprint arXiv:2409.17681*, 2024. 1

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023. 3

Zijian Zhang, Ze Huang, Zhiwei Hu, Xiangyu Zhao, Wanyu Wang, Zitao Liu, Junbo Zhang, S Joe Qin, and Hongwei Zhao. MLPST: MLP is all you need for spatio-temporal prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 3381–3390, 2023. 7

Shuhan Zhong, Sizhe Song, Weipeng Zhuo, Guanyao Li, Yang Liu, and S-H Gary Chan. A multi-scale decomposition MLP-Mixer for time series analysis. *Proceedings of the VLDB Endowment*, 17(7): 1723–1736, 2024. 3, 6, 7, 23

Fan Zhou, Qiang Gao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. Trajectory-user linking via variational autoencoder. In *IJCAI*, pp. 3212–3218, 2018. 1, 2

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021. 2

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286. PMLR, 17–23 Jul 2022. 3

Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17863–17873, 2023. 1

A Implementation Details

A.1 Experimental Protocol

Table 6 summarizes the four datasets. Chengdu and Xi'an are released as part of the DiDi Chuxing GAIA Open Dataset and cover one month of taxi operation inside the central districts of each city. Porto is the public ECML/PKDD 2015 Taxi Service Trajectory dataset, which records 442 taxis over one year in a city whose road topology and mobility patterns differ from both Chengdu and Xi'an. The UAV dataset contains 6,911 deliveries of a truck-based online food delivery platform, simulated under several UAV settings with the Open Air Traffic Simulator. Trips with a gap larger than three sampling intervals and points whose implied speed exceeds 120 kilometres per hour are removed, and Table 6 reports the outcome of that cleaning. Each setting then retains the trips that supply its own history and horizon before windowing. Since the sampling interval differs across datasets, the long-history setting reads 400 observations on Chengdu, Xi'an, and the UAV data and 60 observations on Porto, so that the histories cover durations of the same order across the four datasets and a horizon stated in seconds carries the same meaning on each of them.

Each dataset is partitioned by trip into training, validation, and test sets in a 7/1/2 ratio following collection time. Partitioning by trip instead of by window guarantees that no window of a test trip

Table 6: Dataset statistics after preprocessing.

Dataset	Trips	Points	Interval	Mean length	Region
Chengdu	114,500	58.7M	3s	513	Chengdu second ring
Xi'an	86,200	41.2M	3s	478	Xi'an second ring
Porto	704,000	60.4M	15s	86	Porto metropolitan area
UAV	6,911	8.3M	1s	1,201	Simulated delivery airspace

appears in training, and partitioning by time instead of at random keeps the evaluation prospective. The global per-coordinate mean and standard deviation that define the standardized space are estimated on the training partition alone and then applied unchanged to the other two, so no quantity derived from the validation or test partitions enters the model at any stage of training or tuning.

All models are implemented in PyTorch 2.1.2 with CUDA 12.1 and cuDNN 8.9 under Python 3.10, and optimized with Adam, an initial learning rate of 10^{-3} under cosine decay, a batch size of 64, gradient clipping at a norm of 5.0, and at most 100 epochs with early stopping after 10 epochs without validation improvement. TEMPT uses a dropout rate of 0.1 and a stochastic depth rate of 0.1. Every baseline is given the same coordinate-only inputs, the same history and horizon, the same optimizer and epoch budget, and a random search of 24 configurations over the hyper-parameters that the original method exposes, with the ranges centred on the values its authors report and the configuration selected on validation RMSE of the Xi'an long-history 30-second setting and then applied unchanged to every other dataset, setting, and horizon. TEMPT is tuned with the same budget, so no method receives a larger search than another and no method is re-tuned per cell of Table 1 while another is not. The depth L, the autocorrelation cutoff α, and the loss weight λ are fixed by this one selection and are not revisited on Chengdu, Porto, or the UAV data, so the entries of Table 1 and of Table 11 are produced by a configuration that those three sources never informed. The coarsest patch size cannot exceed the history that a scenario supplies, so the patch sizes of Section 5 follow the history length under the same constraint that sets the window or patch hyper-parameter of every baseline carrying one. Baselines that expect a spatial graph, such as MLPST, STD-MAE, and NexuSQN, are adapted by treating the two coordinate channels as the node dimension, so that the graph mixing operates on the coordinate pair and every method reads the input that the task supplies. The length-handling convention of the *All* scenario is likewise fixed per baseline on the validation partition before any test window is read. Every method is therefore trained, tuned, and evaluated under one identical protocol on identical inputs with an identical search budget.

A.2 Metric Definitions

Let $x_h \in \mathbb{R}^F$ and $\hat{x}_h \in \mathbb{R}^F$ denote the ground-truth and predicted position at horizon step h, with $F = 2$ coordinate channels, and let n index the N evaluation windows. The four metrics used in this paper are

$$\begin{aligned} \text{RMSE} &= \sqrt{\frac{1}{NH}\sum_{n=1}^{N}\sum_{h=1}^{H}\left\|\hat{x}_{n,h}-x_{n,h}\right\|_2^2}, & \text{MAE} &= \frac{1}{NH}\sum_{n=1}^{N}\sum_{h=1}^{H}\left\|\hat{x}_{n,h}-x_{n,h}\right\|_1, \\ \text{ADE} &= \frac{1}{NH}\sum_{n=1}^{N}\sum_{h=1}^{H}\left\|\hat{x}_{n,h}-x_{n,h}\right\|_2, & \text{FDE} &= \frac{1}{N}\sum_{n=1}^{N}\left\|\hat{x}_{n,H}-x_{n,H}\right\|_2. \end{aligned} \tag{9}$$

RMSE and MAE differ in the norm that aggregates the two coordinate channels of a step, and RMSE is the larger of the two because squaring emphasizes the small fraction of windows that contain a sharp manoeuvre in the trajectory.

RMSE and MAE are expressed in the shared standardized coordinate space that is obtained by applying the global per-coordinate mean and standard deviation of the dataset to both the prediction and the ground truth, and that is the space in which every model is trained and evaluated. The normalization is identical for every method, so the entries of every table reported in this paper are directly comparable across models, across horizons, and across settings.

A.3 VARIABLE-LENGTH PATH

This section documents the path that a variable-length history takes through one TEMPT Layer, which is the property that lets a single trained model serve every history length. Table 7 lists every stage of the layer that carries parameters, together with the tensor it consumes and the tensor it produces. The patch count N_i appears throughout the two shape columns and in none of the widths, so a layer instantiated for one history length runs unchanged on any other. The patching, masking, and weighted-sum steps that connect these stages hold no parameters of their own and appear in the listing at the end of this section.

Table 7: Shapes of the parametric stages of one TEMPT Layer. Hidden widths are d_F for feature mixing, d_P for sequence mixing, and d_w for the scoring network.

Stage	Input	Output	Width
Feature mixing	$N_i \times P_i \times F$	$N_i \times P_i \times F$	d_F
Sequence mixing	$N_i \times P_i \times F$	$N_i \times P_i \times F$	d_P
Scoring network	$N_i \times (P_i F + 1)$	N_i	d_w
Reconstruction head	$N_i \times P_i \times F$	$M \times F$	$P_i F$
Prediction head	$P_i \times F$	$H \times F$	HF

Three stages deserve a note. The scoring network consumes one flattened patch at a time together with the scalar n/N_i and emits one logit, so the softmax of equation 4 normalizes over as many logits as the input happens to produce, the position of a patch enters the weight it receives, and patches that consist entirely of padding receive a logit of $-\infty$ and drop out of the weighted sum. The reconstruction head likewise acts on one patch at a time, so its N_i outputs of shape $P_i \times F$ are concatenated along time into a sequence of length $N_i P_i \geq M$ and then truncated to M, which restores exactly the shape of $\mathcal{Z}_{i-1}$ and makes the subtraction of equation 1 well defined for every history length. The validity mask is applied three times, once on the patch tensor that enters the block and once after each of the two residual links, so no tensor that a weight matrix reads carries a value at an inactive position, and it is applied a fourth time to the reconstruction so that the update of equation 1 is driven by observed positions alone. The forward pass of a layer is summarized below, with `m` the mask broadcast along the feature axis.

```
B, mask = patch(Z, P)          # (M,F) -> (N,P,F), (N,P)
m = mask[:, :, None]           # (N,P,1), broadcast over F
B = B * m                      # inactive positions hold zero
B = (B + FeatureMLP(B)) * m    # mixes along F at every (n,p)
B = (B + SequenceMLP(B)) * m # mixes along P, shared over n
s = concat(flatten(B), arange(N)/N)              # (N, P*F+1)
w = softmax(Score(s) + log(any(mask, axis=1)))   # (N,)
h = einsum('n,npf->pf', w, B)   # (P,F), independent of N
C = truncate(reshape(Recon(B) * m, (N*P, F)), M)  # (M,F)
Y = Pred(h)                    # (H,F), future component
Z = Z - C                      # (M,F)
```

Because every tensor is masked before it is read, the third line makes the fill value of the trailing patch irrelevant by construction, and the fourth and fifth lines keep the shared weights of the two mixing operations from carrying a padded position into a valid one.

A.4 REPRODUCIBILITY

The quantities needed to reconstruct every number in this paper are fixed here in full. The four datasets are public, and the cleaning rules of Appendix A.1 are the only filtering applied, so the trip and point counts of Table 6 follow from the raw archives without a further selection step. Windows are cut with a stride of one observation inside each trip, the split is by trip and by collection time in a 7/1/2 ratio, and the standardization constants come from the training partition alone, which fixes the preparation of the data end to end.

The architecture is fixed by the depth $L = 4$, the patch sizes of Section 5, the mixing widths $d_F = 32$ and $d_P = 64$, and the scoring width $d_w = 16$, and Table 7 turns those into the shape that every stage

receives and returns. Optimization is fixed by the settings of Appendix A.1, and the ten seeds are the integers 0 through 9, shared by every method so that the paired comparisons of Appendix B.10 and Appendix B.11 use common random numbers. Determinism is enabled for the CPU measurements and disabled for GPU training, where cuDNN autotuning is left on.

Reported timings follow one rule throughout. Latency is the median wall-clock time of one thousand forward passes taken after two hundred warm-up passes, with batch size one, a single thread bound to one physical core, data loading excluded, and no custom kernels, operator fusion, or quantization on any method. Appendix B.4 gives the setting at which Table 2 is evaluated. The source code, the configuration files that define the search spaces, and the trained weights are released with the paper, together with the commands that produce every table it reports.

B Additional Results

B.1 Displacement Metrics

Average Displacement Error (ADE) and Final Displacement Error (FDE) are the conventional metrics of the trajectory prediction literature, where ADE averages the Euclidean distance between prediction and ground truth over all predicted steps and FDE measures it at the final step. We retrain every baseline on Xi'an under the protocol of Section 5 and report both metrics on coordinates rescaled to the unit interval with the per-coordinate range of the training partition, so that all entries of Table 8 are directly comparable across models.

Table 8: ADE and FDE on Xi'an with a 15-second prediction horizon.

Metric	TEMPT	TransferTraj	NexuSQN	MSD-Mixer	TrajGDM	DLinear
ADE	**1.78**	2.82	2.12	1.93	3.63	2.26
FDE	**3.12**	4.31	3.55	3.31	4.87	3.86

TEMPT attains the lowest ADE and the lowest FDE, so the advantage observed under RMSE and MAE in Table 1 carries over to displacement metrics. The margin at the final step is of the same order as the margin averaged over the horizon, indicating that the advantage does not come from accurate early steps followed by drift.

B.2 Additional Forecasting Baselines

The seven baselines of Table 1 cover spatiotemporal, Transformer, diffusion, and mixing architectures, and eight further reference points extend that comparison. Four of them carry no learned capacity, namely a persistence rule that holds the last observed position, a constant-velocity extrapolator that continues the last observed displacement (Schöller et al., 2020), a linear extrapolator that fits a least-squares line to the whole history and continues it, and a Kalman filter with a constant-velocity state and the process and observation variances estimated on the training partition. The remaining four are learned, namely an LSTM (Hochreiter & Schmidhuber, 1997) and a temporal convolutional network (Bai et al., 2018) as the two classical sequence encoders, and N-BEATS (Oreshkin et al., 2020) and N-HiTS (Challu et al., 2023) as the two decomposition-based forecasters closest in spirit to TEMPT. All eight consume the same inputs, the same history and horizon settings, and the same windows as the seven baselines of the main comparison, the four learned ones receive the protocol of Appendix A.1 in full, and the four analytical rules carry no hyper-parameters to search.

Three observations follow from Table 9. The four analytical rules order themselves exactly as their assumptions predict, since holding the last position is the weakest, continuing the last displacement improves on it, fitting a line to the whole history improves again by averaging the receiver jitter, and a constant-velocity Kalman filter is the strongest of the four because it weights that average by the estimated noise. The best of the four still trails TEMPT by more than a factor of five at the shortest horizon, where inertia is most favourable to it, and by more than a factor of two at the long-history setting, so the margins of Table 1 do not come from motion inertia that any extrapolator would capture. N-HiTS is the strongest of the learned four at the long-history setting and reaches 19.86 against 19.27 for MSD-Mixer in Table 1, so the two published decomposition forecasters land within one RMSE point of each other and both remain well above TEMPT. Neither carries a variable patch

Table 9: Analytical and learned reference points on Chengdu short-history 15-second and Xi'an long-history 30-second prediction.

Model	Chengdu Short 15s		Xi'an Long 30s	
	RMSE	MAE	RMSE	MAE
Persistence	19.86	16.24	45.18	34.66
Constant velocity	16.35	13.42	38.62	29.47
Linear extrapolation	15.42	12.71	36.05	27.84
Kalman filter	14.27	11.83	34.77	26.59
LSTM	11.06	9.12	22.74	17.05
Temporal convolutional network	9.83	8.25	21.38	16.22
N-BEATS	15.87	13.04	22.47	17.63
N-HiTS	13.94	11.47	19.86	15.11
TEMPT	**2.67**	**2.34**	**15.46**	**12.55**

count, which is the property that lets one set of TEMPT weights serve every history length in the *All* scenario under a single trained model.

B.3 Length-handling Strategies

In the *All* scenario a single generator draws one effective history length per window and every model is evaluated on exactly the same windows in the same order, so the comparison is paired at the sample level. A model whose weights are built for one length still has to reduce those windows to that length, and we give each baseline three standard ways of doing so. Masked padding extends every history to the maximum length and masks the padding, truncation keeps the most recent segment of the minimum length, and resampling interpolates every history onto a fixed grid. Each baseline is trained once under each convention, the convention is fixed by its validation RMSE before any test window is read, and the entry that Table 1 reports is the test error of that fixed choice. Table 10 reports the test error of all three. TEMPT consumes each history at the length it arrives with, and masked padding leaves that computation unchanged, since a fully padded patch receives a logit of $-\infty$ in equation 4 and drops out of the weighted sum. We put TEMPT through the other two conventions as well, which measures what those conversions cost on a model that has no need of them.

Table 10: Test RMSE of each length-handling strategy on Xi'an *All* 30-second prediction. The convention selected on the validation partition is marked with a dagger and is the one reported in Table 1, and for TEMPT the selected path is the native one.

Model	Masked padding	Truncation	Resampling
MLPST	18.11†	20.94	19.27
TrajGDM	19.73†	22.65	21.08
STD-MAE	20.84	23.19	20.26†
PatchTST	21.19†	24.02	22.36
NexuSQN	24.38†	27.16	25.83
MSD-Mixer	28.80†	31.22	30.15
DLinear	37.52	39.81	36.60†
TEMPT	**9.68**†	11.42	10.35

Three properties of Table 10 matter for the fairness of the comparison. The validation partition selects masked padding for five of the seven baselines and resampling for STD-MAE and DLinear, and the ordering of the three conventions on validation agrees with their ordering on test for every one of the eight models, so the selection rule and the test minimum coincide here without the test partition having taken part in the choice. The spread across strategies reaches 3.2 RMSE on a baseline and 1.7 on TEMPT, so the choice of convention moves any one method far less than the 8.4 that separates the strongest baseline entry from the 9.68 that a single set of TEMPT weights reaches on the same windows, and imposing masked padding uniformly on every method would leave that margin at 8.4 as well. The selected convention also differs by architecture, which is why the choice is made per model on validation instead of once for all of them, so each baseline enters the comparison under the convention its own validation error prefers.

B.4 Cost Accounting

Table 2 reports one parameter count and one FLOP count per model, both evaluated at the Xi'an long-history 30-second setting with a history of 400 observations, a horizon of 10 steps, and a batch of one trajectory. The parameter count of TEMPT is independent of the history by construction, since no weight of Table 7 has a shape in which N_i or M appears. Every count follows one convention in which a multiply-accumulate is charged as two operations, and that convention is applied to all six models alike, so the comparison rests on a single accounting that covers all six implementations and no method is charged for an engineering choice.

B.5 Cross-domain Evaluation

We further evaluate TEMPT on two datasets beyond Chengdu and Xi'an to examine cross-domain robustness. Porto contains real-world vehicle trajectories collected from a different city with distinct road topology and mobility patterns, taken from the public ECML/PKDD 2015 Taxi Service Trajectory dataset. The UAV delivery dataset contains 6,911 simulated deliveries from a truck-based online food delivery platform using the Open Air Traffic Simulator, covering multiple UAV operation settings. These two datasets differ from the taxi datasets in both spatial structure and motion dynamics, and therefore provide additional evidence for the generality of the proposed decomposition and of the adaptive aggregation that reads it.

Table 11: Cross-domain trajectory prediction results on Porto and UAV datasets.

Dataset	Horizon	TEMPT		NexuSQN		MSD-Mixer	
		RMSE	MAE	RMSE	MAE	RMSE	MAE
Porto	15s	**5.59**	**5.34**	16.67	14.04	9.16	8.33
	30s	**12.74**	**11.25**	35.25	33.98	15.78	14.63
	60s	**47.43**	**43.34**	78.19	71.09	58.68	52.32
	120s	**85.27**	**77.07**	120.37	102.15	117.16	103.85
	300s	**204.16**	**189.92**	270.57	236.18	231.51	216.06
UAV	15s	**1.59**	**0.89**	9.34	7.83	4.54	4.07
	30s	**3.69**	**2.93**	10.78	8.38	5.55	5.02
	60s	**13.94**	**12.24**	22.32	20.43	17.93	17.52
	120s	**32.21**	**30.22**	44.95	42.42	43.94	42.24
	300s	**105.10**	**91.82**	212.97	208.88	139.21	133.73

As shown in Table 11, TEMPT consistently achieves the lowest RMSE and MAE across all horizons on both datasets. The gains remain clear for the UAV setting, where the trajectory source and operating dynamics differ substantially from urban taxi trajectories, supporting the robustness of TEMPT under a domain shift that changes both the platform and the medium of travel.

We additionally compare with TransferTraj (Wei et al., 2025), a recent coordinate-based model that uses expert modules with a masking and recovery scheme for transfer across regions and tasks. Since transferability is its design target, the two cross-domain datasets form a natural testbed, and we retrain it under the same protocol.

Table 12: Comparison with TransferTraj on Porto and UAV trajectories.

Dataset	Horizon	TEMPT		TransferTraj	
		RMSE	MAE	RMSE	MAE
Porto	15s	**5.59**	**5.34**	8.03	7.02
	30s	**12.74**	**11.25**	16.28	12.11
	60s	**47.43**	**43.34**	59.84	53.97
	120s	**85.27**	**77.07**	104.67	91.99
UAV	15s	**1.59**	**0.89**	6.10	4.23
	30s	**3.69**	**2.93**	12.14	9.73
	60s	**13.94**	**12.24**	20.52	18.98
	120s	**32.21**	**30.22**	42.80	40.19

Table 12 shows a consistent advantage for TEMPT on both domains and at every horizon, with the larger margins on the UAV data whose motion dynamics are the furthest from those of the urban vehicle trajectories used during development.

B.6 Held-out Splits

The partition of Appendix A.1 separates trips and follows collection time, which already prevents a window of a test trip from appearing in training. A stricter question is how much of the accuracy survives when the identity, the period, or the city of the test data is withheld as well. We therefore repeat the Xi'an long-history 30-second setting under three additional splits. The cross-vehicle split holds out whole driver identifiers, so no trip of a test driver is seen during training. The cross-period split, run on Porto because it spans a full year, holds out the last quarter. The zero-shot split trains on Chengdu alone and evaluates on Xi'an without any target data.

Table 13: RMSE under held-out splits. The cross-period column is measured on Porto at the 30-second horizon and the remaining columns on Xi'an long-history 30-second prediction.

Model	Reported split	Cross-vehicle	Cross-period	Zero-shot
TEMPT	**15.46**	**15.91**	**13.18**	**25.83**
MSD-Mixer	19.27	20.48	16.94	34.51
MLPST	21.69	23.31	19.02	37.62
STD-MAE	21.86	23.74	19.85	39.08

Holding out whole drivers costs TEMPT 2.9 percent of RMSE and holding out a quarter of the year costs 3.5 percent, against 6.3 to 8.6 percent for the three baselines, so the margin widens as the split becomes stricter. Under zero-shot transfer TEMPT reaches 25.83 on a city it has never seen, which is below the 26.87 that DLinear and the 33.57 that NexuSQN reach in Table 1 after training on Xi'an itself. Supplying one percent of the Xi'an training trips brings the transferred model to 17.24 and ten percent brings it to 16.02, within four percent of the 15.46 obtained from the full partition, so most of the remaining gap closes with a small fraction of target data.

B.7 Alternative Hardware

Training TEMPT is performed off-device, so the cost that matters for deployment is the per-trajectory inference latency. We measure it under the long-history setting on a single CPU thread and on two GPU generations older than the RTX 4090 used in Table 3. The host is an Intel Xeon Gold 6330 at 2.0 GHz running Ubuntu 22.04, and the CPU row uses one thread pinned to one physical core with the thread count of both the intra-operator and the inter-operator pool set to one, so the measurement is not the result of a parallelism that a deployment would not have. All measurements use the same PyTorch 2.1.2 build in single precision with batch size one and no custom kernels, no operator fusion, and no quantization, so no method benefits from an implementation advantage that the others do not receive and the three processors are compared on one footing.

Table 14: Per-trajectory inference latency in seconds under the long-history setting.

Hardware	15s	30s	60s	120s	300s
CPU, single thread	0.069	0.085	0.096	0.116	0.169
RTX 2080 Ti	0.084	0.092	0.103	0.129	0.196
RTX 3090	0.071	0.089	0.095	0.114	0.188

Every entry of Table 14 stays below 0.2 seconds. Single-thread CPU prediction time is within fifteen percent of the RTX 4090 of Table 3 at every horizon and is comparable to that of the RTX 2080 Ti and the RTX 3090, falling below both at most horizons, so the accuracy of TEMPT is reached on commodity processors and does not depend on the class of accelerator underneath it. These measurements support the feasibility of on-device CPU deployment directly, since TEMPT meets the real-time budget without an accelerator and leaves the choice of processor to the deployment, which is what a low-resource setting asks of a forecaster.

B.8 Uncertainty Estimates

The deterministic formulation of Section 5 reports a single future path. Two lightweight extensions of the same backbone deliver richer predictive outputs. The first replaces the prediction head with a per-step Gaussian output trained by negative log-likelihood, adding 2.9K parameters. The second replaces the single head with $K = 6$ parallel heads that share the backbone representation, together with a small confidence branch over the hypotheses, trained with a best-of-K objective so that the heads specialize on distinct continuations.

Table 15: Calibration of the Gaussian head on Xi'an.

Horizon	50% cov.	90% cov.	NLL
15s	0.518	0.911	−3.26
30s	0.507	0.903	−2.71
60s	0.494	0.892	−2.11

Table 16: Multi-hypothesis prediction with $K = 6$.

Horizon	minADE@6	minFDE@6
15s	1.04	1.59
30s	1.95	3.09

In Table 15, both the 50% and the 90% predictive intervals stay within two points of their nominal level at every horizon, while the negative log-likelihood rises from 15 to 60 seconds as the future becomes less certain, which is the behavior expected of a calibrated long-horizon forecaster. Table 16 shows that the six hypotheses lower ADE from 1.78 to 1.04 and FDE from 3.12 to 1.59 at the 15-second horizon, indicating that the compact representation supports genuinely distinct continuations instead of perturbations around one mean path. Both variants stay within six percent of the parameter count of Table 2 and leave the latency of Table 14 unchanged, so the richer output comes at the cost profile reported throughout the paper.

B.9 Incomplete Histories

Each layer segments the already-sampled sequence into patches of P_i points, and the resulting variable patch count is pooled by masked weighted aggregation, so a shortened or partially missing history changes the number of patches and leaves the shape of every weight matrix untouched. We examine this behavior on Xi'an short-history 30-second prediction by discarding a controlled fraction of the observations of each history. The retained points keep their original timestamps, so the interval between consecutive retained observations is no longer the nominal 3 seconds and the history becomes genuinely irregular as well as shorter. Two protocols generate the loss. Uniform dropout removes isolated points and stands for the intermittent failure of a receiver, while a contiguous gap removes one block of consecutive observations at a random offset and stands for the sustained outage of a tunnel or an urban canyon. Two input formats are compared. The coordinate-only format presents the retained points as they arrive and leaves the model to read the gaps from the motion, and the interval format appends the elapsed time since the previous retained observation as a third input channel, which raises F from 2 to 3 and the parameter count from 265K to 267K. Table 17 pairs the two formats under uniform dropout and reports the harder contiguous gap under the interval format.

Table 17: Accuracy under incomplete histories on Xi'an short-history 30-second prediction, where the retained observations keep their original timestamps.

Protocol and input	Metric	0%	10%	30%	50%
Uniform dropout, coordinates	RMSE	9.60	9.66	9.91	10.40
	MAE	8.33	8.38	8.56	8.89
Uniform dropout, with interval	RMSE	9.60	9.63	9.79	10.08
	MAE	8.33	8.35	8.47	8.66
Contiguous gap, with interval	RMSE	9.60	9.71	10.23	11.05
	MAE	8.33	8.42	8.78	9.34

Three readings of Table 17 answer the question the protocol was built for. With half of the history removed uniformly, the coordinate-only error grows by roughly eight percent, so the architecture absorbs a history that is both shorter and unevenly spaced without any change to a weight shape, and a stationary period produces near-zero-displacement patches that are taken up as a low-magnitude component. Supplying the elapsed interval as a third channel recovers between a third and a half of

that growth at every level, reaching 10.08 instead of 10.40 at the 50 percent level for 2K additional parameters, which shows that the gap structure carries information the coordinates alone do not expose and that the model uses it once it is given. A contiguous gap is the harder loss, since removing one block of half the history leaves the model with no observation across an entire manoeuvre, and the error reaches 11.05 against 10.08, which is a growth of fifteen percent over the complete history and remains below the 16.59 that the strongest baseline reaches in Table 1 at the same setting.

B.10 Run-to-run Variance

Every entry of Table 1 is the mean over ten runs, and the ten random seeds are shared by all methods, so each method sees the same initialization stream, the same shuffling order, and the same length draws in the *All* scenario. Comparisons are therefore paired in the sense of common random numbers, and the difference between two methods on one seed measures the two methods alone. Table 18 reports the dispersion of those runs at the Xi'an long-history 30-second setting, together with the two extreme runs that bound it.

Table 18: Run-to-run variance over ten shared seeds on Xi'an long-history 30-second prediction.

Model	RMSE mean $\pm$ s.d.	Best seed	Worst seed
TEMPT	$\mathbf{15.46 \pm 0.11}$	**15.29**	**15.64**
MSD-Mixer	19.27 ± 0.19	18.98	19.55
TrajGDM	20.47 ± 0.36	19.92	21.03
MLPST	21.69 ± 0.24	21.31	22.08
STD-MAE	21.86 ± 0.31	21.38	22.33
PatchTST	24.43 ± 0.27	24.01	24.85
DLinear	26.87 ± 0.14	26.65	27.09
NexuSQN	33.57 ± 0.42	32.92	34.21

The worst of the ten TEMPT runs reaches 15.64 and the best run of the strongest baseline reaches 18.98, so the two sets of runs are separated by a margin that no seed closes. The standard deviation of TEMPT is 0.7 percent of its mean and the largest in the table is 1.8 percent, while across all 22 settings of Table 1 the largest relative standard deviation reached by any method is 2.3 percent, which places run-to-run dispersion an order of magnitude below every gap the paper reports. Repeating the procedure at all 22 settings gives 154 paired comparisons, where Holm correction over that whole family leaves all 154 significant at the 5 percent level and all but four at the 1 percent level, the four exceptions being 300-second comparisons where the absolute dispersion is largest. A Wilcoxon signed-rank test over the same paired differences agrees with the parametric test in every row, so the conclusion does not depend on a normality assumption.

B.11 Per-sample Distributions

Seeds bound the dispersion of a training procedure and say nothing about the dispersion across the test trajectories that a deployment would meet. We therefore resample the test windows in place of the seeds. For each setting the per-window squared errors of TEMPT and of the strongest baseline at that setting are drawn together, which keeps the pairing, and the resampling is repeated ten thousand times to give a percentile interval for the difference of the two RMSE values. The same per-window differences give a standardized effect size and the fraction of windows on which TEMPT is the more accurate of the two, which the last column reports as the win rate.

Table 19: Paired bootstrap over test trajectories against the strongest baseline at each setting, with ten thousand resamples.

Setting	Strongest baseline	ΔRMSE	95% interval	Effect size	Win rate
Chengdu Short 15s	STD-MAE	0.93	[0.88, 0.98]	1.42	0.87
Chengdu Long 30s	STD-MAE	0.95	[0.86, 1.04]	0.81	0.74
Chengdu All 30s	MSD-Mixer	3.14	[2.98, 3.31]	1.05	0.82
Xi'an Short 30s	STD-MAE	6.99	[6.71, 7.28]	1.18	0.85
Xi'an Long 30s	MSD-Mixer	3.81	[3.65, 3.97]	0.96	0.79
Xi'an All 30s	MLPST	8.43	[8.11, 8.76]	1.31	0.88

No interval in Table 19 contains zero, and the narrowest of them, $[0.88, 0.98]$, is separated from zero by eighteen of its own half-widths, so the conclusion survives resampling of the evaluation data and not only of the initialization. The effect sizes lie between 0.81 and 1.42, which places every comparison in the range conventionally read as large, and the win rates between 0.74 and 0.88 show that the mean difference is produced by a majority of the test trajectories instead of by a tail of windows on which the baseline fails badly. The two quantities also move together, since the setting with the smallest effect size is the setting with the lowest win rate, which is the pattern expected when the per-window differences are concentrated. The intervals here are wider in relative terms than the seed intervals of Appendix B.10, because the variability between trajectories is the larger of the two sources, and reporting it is what makes the margins in Table 1 statements about the task itself.

B.12 Residual Regularization Sensitivity

The residual regularization uses an autocorrelation cutoff $\alpha = 2.0$ and residual loss weight $\lambda = 0.2$ by default. We evaluate sensitivity on Xi'an long-term 30-second prediction by varying one hyperparameter at a time while keeping the remaining configuration unchanged.

Table 20: Sensitivity to the autocorrelation cutoff α.

α	0.5	1.0	2.0	3.0	4.0
RMSE	16.11	15.72	**15.46**	15.69	16.09
MAE	12.78	12.68	**12.55**	12.67	12.99

Table 21: Sensitivity to the residual loss weight λ.

λ	0	0.1	0.2	0.3	0.5	0.7
RMSE	16.38	15.94	**15.46**	15.50	15.73	16.11
MAE	13.21	12.83	**12.55**	12.59	12.64	13.05

Tables 20 and 21 show that performance is stable near the selected defaults. Removing residual regularization degrades accuracy, while overly large regularization weights also reduce performance, indicating that the residual objective is beneficial when used as a moderate structural constraint.

B.13 Decomposition Identifiability

The recursion of equation 1 constrains only the sum of the components, so the split between them is pinned down by the rest of the architecture instead of by the reconstruction constraint. Three mechanisms do that work. The patch size P_i bounds the time scale that layer i can represent, sequence mixing operates strictly within a patch and cannot express variation slower than the patch it sees, and the residual objective of equation 6 removes any incentive to carry structured autocorrelation forward. Table 22 switches the last two on and off and reports what each of them contributes to the separation.

Table 22: Separation of the learned components on Xi'an long-history 30-second prediction, where a lower value is better in every column except the cross-seed cosine similarity.

Configuration	C_1 energy	Cross-corr.	Residual autocorr.	Cross-seed cos.	RMSE
Full model	**0.612**	**0.17**	**0.04**	**0.84**	**15.46**
No residual objective	0.871	0.34	0.29	0.71	16.38
Uniform patch size	0.905	0.41	0.33	0.66	18.95

In the full model the four components carry 0.612, 0.238, 0.099, and 0.037 of the input energy with a final residual of 0.014, so the decomposition is graded across the four components, and the largest correlation between any two components is 0.17. Removing the residual objective collapses 0.871 of the energy into the first component and lifts the cross-correlation to 0.34, and imposing a uniform patch size does the same more strongly, which is the configuration that Table 4 reports as TEMPT-U. The full configuration holds the best entry of every column, so the separation between components, their stability across seeds, and the accuracy of the forecast are produced together by the same three

mechanisms instead of traded against one another, and the split is not an arbitrary one among many that reconstruct the same input to within the same total error.

The components are also stable and ordered. The smallest cross-seed cosine similarity of any component is 0.84 in the full model and drops to 0.66 under a uniform patch size, so the assignment of structure to layers survives reinitialization only when the patch sizes differentiate the layers. The spectral centroids of the four components are 0.021, 0.058, 0.124, and 0.287 cycles per observation, which follows the order of the patch sizes $\{40, 20, 10, 1\}$ and confirms that a layer represents the motion scale its patch size admits.

B.14 Aggregation Rules

Dynamic weighted aggregation maps a variable patch count onto a fixed-size representation, and the scoring network of equation 4 is the only component that reads across patches. Table 23 compares it with six alternatives under an otherwise identical backbone. Two of the six address the order in which the patches arrive. The first removes the normalized position n/N_i from the input of the scoring network, which leaves a weighted sum of independently scored patches and makes the aggregation exactly invariant to a permutation of them. The second replaces the scoring network with a single-layer recurrent unit that reads the patches in order and emits the weights from its states, which is order-sensitive and still admits any patch count.

Table 23: Aggregation rules on Xi'an long-history 30-second prediction, with the last column measured after the patches are presented in a random order at inference.

Rule	RMSE	MAE	Permuted RMSE
Mean pooling	18.51	13.60	18.51
Last patch only	19.34	14.27	24.06
Learned static weights	17.95	13.41	22.87
Attention pooling	16.12	12.97	23.72
Dynamic weighted without position	16.34	13.12	16.41
Recurrent weighting	15.98	12.89	24.55
Dynamic weighted (ours)	**15.46**	**12.55**	24.90

Three results shape the design. Learned static weights improve on mean pooling, which shows that patches are not equally informative, but that weight vector is indexed by the patch count and therefore fixes the history length at construction, which is the property the architecture exists to avoid. Attention pooling reads across the patches, leaves the count free, and remains order-sensitive, and it ends 0.66 RMSE behind the proposed rule while introducing a pairwise interaction whose cost grows with the square of the patch count. Recurrent weighting is order-sensitive by construction and lands 0.52 RMSE behind, at the price of a sequential dependence that the rest of the architecture does not have, while the proposed rule scores every patch in parallel.

The last column separates order sensitivity from accuracy. Presenting the patches in a random order at inference leaves mean pooling exactly unchanged, which is the control the column needs, and leaves the position-free variant at 16.41 against its own 16.34, so removing n/N_i makes the aggregation permutation-invariant to within the resolution of the measurement. The proposed rule moves from 15.46 to 24.90 under the same permutation, so the patch order is read by the aggregation, and the position-free variant pays 0.88 RMSE for discarding it. A scoring network that reads one patch together with its index therefore supplies both the input dependence and the order dependence the task needs, while keeping aggregation linear in the patch count and free of any parameter indexed by the patch count of the input.

B.15 Relation to Prior Architectures

The closest architecture is MSD-Mixer (Zhong et al., 2024), from which TEMPT differs in two respects that decide whether one model can serve a variable history at all. MSD-Mixer mixes across patches with an MLP whose input dimension is the patch count, so the shape of that weight matrix fixes the input length at construction and every new length calls for a new model. TEMPT pools a variable number of patches with a scoring network that reads one patch together with its normalized position and returns one logit, so the patch count enters no weight shape and a single trained set of

weights covers the whole range of history lengths that Table 1 evaluates. TEMPT also places its mixing where the structure of the data is, since feature mixing acts on the longitude and latitude of one physical position and keeps intact the heading and the speed that a displacement encodes, while a mixer over generic variates holds no such pair together. N-BEATS (Oreshkin et al., 2020) shares the recursive residual idea but operates on a whole fixed window with no notion of scale-specific segmentation, while DLinear (Zeng et al., 2023) and TSMixer (Chen et al., 2023) keep the window fixed and decompose either trivially or not at all. In the vision-based setting, FlexiLength (Xu & Fu, 2024) handles observation-length shift by training across several lengths and calibrating the resulting representations, which adapts a fixed-window backbone to a set of lengths, whereas TEMPT has no fixed window to adapt in the first place.

These distinctions are visible in the measurements as well as in the design. MSD-Mixer, the closest published alternative, reaches 19.27 RMSE on Xi'an long-term 30-second prediction against 15.46 for TEMPT, as Table 1 reports. Within TEMPT itself, removing the multi-scale patching raises the error to 18.95, removing the dynamic aggregation raises it to 18.51, and removing the decomposition raises it to 16.80. The contribution is therefore not the presence of decomposition or of mixing, both of which are established, but the specific arrangement in which the recursion supplies the complementarity between scales, the residual objective keeps the recursion from stalling, and the aggregation makes the whole stack independent of how many patches the input happens to produce. Removing any one of the three degrades either the accuracy or the variable-length property that motivates the architecture, as Table 4 reports for all four variants.

C Future Directions

The present study establishes TEMPT on urban taxi trajectories from two cities, on Porto vehicle data, and on simulated UAV deliveries, and the same architecture extends to further mobility scenarios and sensing platforms, where the variable-length path of Appendix A.3 applies without modification.

The input the model reads is the coordinate stream of one agent, which is what a device holds about itself. Road topology, signal state, and the trajectories of nearby agents are complementary sources that the architecture accepts as additional channels in the same way that Appendix B.9 supplies the sampling interval, so a map-conditioned and interaction-aware encoder is a direct next step that keeps the design intact. At intersections the continuation of a trip is genuinely multi-valued, and the multi-hypothesis and Gaussian heads of Appendix B.8 are the form in which the present architecture represents that, with the calibrated interval that a single path cannot supply.

Further directions include ultra-long-horizon forecasting beyond 5 minutes by pairing the configurable context of TEMPT with a coarse route-level summary of the earlier history, joint training across cities so that one set of weights serves several deployments at once, and quantization and operator fusion on embedded platforms for further latency and energy savings on top of the reduced arithmetic cost that the architecture already provides at full precision.