

A More Details of Model Implementation

A.1 Code Availability

We have made our code available as open source in an anonymous repository: <https://anonymous.4open.science/r/FlowNet-NeurIPS2025-52EC/>.

A.2 Initialization

For the linear layers in M-MLP, we employed Kaiming initialization to set their initial parameters. All other linear layers in the model were initialized using Xavier initialization. GeLU is employed as the activation function in M-MLP. Specifically, for linear layers in the ASM module, we initialized the weight parameters w to zero and set the bias terms b to the average distance matrix for each corresponding dataset. Through this initialization method, each node can have a sufficiently large and identical receptive field at the beginning of training, and can optimize its own perception radius through gradient descent and backpropagation.

B More Details of Experiments

B.1 Datasets

- **PEMS04F.** The PEMS04 [35] dataset, constructed from the Caltrans Performance Measurement System (PeMS), captures traffic flow dynamics across 307 sensor nodes in California over a two-month period (January 1 to February 28, 2018) with 5-minute granularity (16,992 timesteps). The original dataset includes three key traffic metrics (traffic flow, lane occupancy, and average speed), and we focus on the traffic flow (denoted as PEMS04F) attribute as the prediction target.
- **DeepBase.** The DeepBase [38] dataset is a hydrological dataset providing daily baseflow estimates for 1,661 basins across the contiguous United States (CONUS) from 1981 to 2022. This dataset captures the slow-varying groundwater contributions to streamflow at a daily temporal resolution.
- **SINPA.** The SINPA dataset [39] captures large-scale parking availability dynamics across Singapore, covering 1,687 parking lots over a one-year period (July 2020 – June 2021). It provides high-frequency spatio-temporal observations recorded at 15-minute intervals, where each node represents the available parking spaces at a specific lot. While the original dataset integrates diverse urban features, we focuses on modeling parking vacancy counts as the primary prediction target.

Table 3: Description of the datasets.

Dataset	Category	Data Type	#Nodes	#Time points	Resolution	Date Range
PEMS04F	Traffic	Traffic flow	307	16992	5 min	Jan.1, 2018 - Feb.28, 2018
DeepBase	Hydrology	Base flow	1661	14975	1 day	Jan.1, 1981 - Dec.31, 2022
SINPA	Urban Mobility	Parking slot	1687	35040	15 min	Jul.1, 2020 - Jun.30, 2021

B.2 Preprocess

For the graph-based baseline, we construct a weighted adjacency matrix by applying a thresholded Gaussian kernel to pairwise Euclidean distances between nodes:

$$W_{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\right), & \text{if } \text{dist}(v_i, v_j) \leq \kappa \\ 0, & \text{otherwise} \end{cases}$$

where edge weight $W_{ij} \in [0, 1]$ encodes proximity between stations v_i and v_j , σ controls the kernel width, and κ sparsifies connections beyond local neighborhoods.

We standardize the spatiotemporal dataset $\mathbf{X} \in \mathbb{R}^{N \times T}$ (N nodes, T timesteps) via z-score normalization:

$$\tilde{x}_{i,t} = \frac{x_{i,t} - \mu_i}{\sigma_i} \quad \forall i \in \{1, \dots, N\},$$

where $\mu \in \mathbb{R}^N$ and $\sigma \in \mathbb{R}^N$ denote per-node training means and standard deviations. During evaluation, predictions on validation/test sets are inversely transformed $\hat{x}_{i,t} = \tilde{x}_{i,t} \cdot \sigma_i + \mu_i$ before computing evaluation metrics.

B.3 Training & Validation

We configure prediction horizons based on temporal granularity and dominant frequencies across datasets. For PEMS04F and SINPA, short-term forecasting uses 12-step input/output sequences (1h/3h), while long-term forecasting employs 288-step windows (1d/3d). DeepBase adopts 32-step (monthly, ensuring divisibility for patching) and 360-step (yearly) horizons, respectively. The datasets are partitioned as follows: PEMS04F (70% train, 10% validation, 20% test), DeepBase (pre-2021 train, 2021-2022 validation/test), and SINPA (pre-May-2021 train, May-June 2021 validation/test). We implement sliding window sampling with stride 1 during training, aligning window size with output horizon during inference.

Following standard evaluation protocols in machine learning, we quantify the predictive performance of regression models through two widely adopted metrics: the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Formally, let $Y_i \in \mathbb{R}$ represent the ground-truth value of the i -th data instance and $\hat{Y}_i \in \mathbb{R}$ denote its corresponding predicted value. These error metrics are computed as

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i| \quad (11)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2} \quad (12)$$

where n denotes the total number of test samples. During training, we employ MAE as the loss function for FlowNet and all baselines. Our Early Stopping mode protocol monitors the MAE metric at the validation set and terminates training when no improvement is observed for 10 consecutive epochs relative to the best recorded value, indicating potential overfitting. We retain the model parameters, achieving the lowest validation MAE for final evaluation on the test set.

C Details of Baselines

- **Autoformer** [40]: Autoformer integrates decomposition architecture into Transformers, replacing self-attention with an auto-correlation mechanism to capture periodic dependencies. It progressively decomposes trend and seasonal components, achieving efficient long-term forecasting with linear complexity.
- **PatchTST** [41]: By segmenting time series into independent patches and adopting channel-independent modeling, PatchTST enhances local semantic extraction and reduces computational costs. It outperforms traditional Transformers in long-term forecasting tasks by leveraging vision transformer-inspired patch processing and self-supervised learning.
- **Crossformer** [42]: Crossformer employs a two-stage attention mechanism to model cross-dimension dependencies in multivariate time series. Its hierarchical encoder-decoder structure and dimension-segment-wise embedding efficiently capture interactions between time steps and variables.
- **iTransformer** [43]: iTransformer inverts the standard architecture by treating time points as tokens and applying attention across variables.
- **FEDformer** [44]: Combining frequency-domain transformations with seasonal-trend decomposition, FEDformer reduces computational overhead through random frequency component selection. Its linear complexity and frequency-enhanced blocks make it effective for long-term forecasting in energy and weather datasets.
- **SCINet** [45]: SCINet uses a binary tree structure with interactive convolution blocks to hierarchically decompose time series. This architecture captures multi-resolution temporal dependencies and mitigates information loss, outperforming RNN and Transformer models in efficiency and accuracy.
- **STGCN** [24]: STGCN integrates graph convolutional networks (GCNs) and gated temporal convolutions to model traffic networks. Its fully convolutional design processes large-scale spatiotemporal data efficiently.

- **GWNET** [34]: This model introduces an adaptive adjacency matrix to learn hidden spatial dependencies and dilated convolutions for long-range temporal patterns. It addresses incomplete graph structures in traffic forecasting and achieves linear complexity with stable multi-step predictions.
- **STTN** [16]: Spatial-Temporal Transformer Network replaces GCNs with dynamic spatial attention to capture time-varying node relationships. Its non-autoregressive multi-step prediction framework avoids error accumulation, significantly improving long-horizon forecasting.
- **STAEformer** [19]: By incorporating spatiotemporal adaptive embeddings into vanilla Transformers, STAEformer dynamically adjusts to traffic patterns without complex architectural modifications by preserving intrinsic chronological information and spatial heterogeneity through lightweight adaptive components.

All baseline code implementations are based on the time-series-library [58] and LargeST [59]. Specifically, to ensure the fairness of the comparison and guarantee that most models can run on our existing computing resources, we set the hidden dimension of all models to 64. For encoder-only models, we stacked 2 encoder layers. For encoder-decoder architecture models, we stacked 1 encoder layer and 1 decoder layer.

D More Experiments

D.1 Hyperparameter Study

We conduct a hyperparameter analysis on the PEMS04F dataset for short-term spatio-temporal forecasting, focusing on two architectural parameters: the number of experts in M-MLP and the hidden dimension size. Our experiments evaluate how these choices impact prediction accuracy (MAE, RMSE) under fixed training protocols. This study aims to guide practical configurations that balance model capacity and computational efficiency in real-world deployments.

Effects of Experts. We investigate how the number of experts in M-MLP impacts performance. Increasing experts from 2 to 4 reduces MAE by 0.17 (18.65 \rightarrow 18.48) and RMSE by 0.05 (29.08 \rightarrow 29.03), suggesting that additional experts improve the model’s ability to capture heterogeneous flow interactions. However, further increasing experts to 8 or 16 yields diminishing returns, indicating redundancy in task-specific feature partitioning. We hypothesize that mid-sized configurations (e.g., 4 experts) strike an optimal balance between specialization and computational efficiency for this task.

Effects of Hidden Dimension. We observe that expanding the hidden dimension from 32 to 96 progressively improves accuracy (MAE: 18.74 \rightarrow 18.41; RMSE: 29.35 \rightarrow 29.03), as larger dimensions better encode complex flow dynamics. However, scaling to 128 dimensions degrades performance, likely due to overfitting or optimization challenges in high-dimensional parameter spaces. This suggests that moderate hidden dimensions maximize representational power while maintaining generalization, aligning model capacity with data availability.

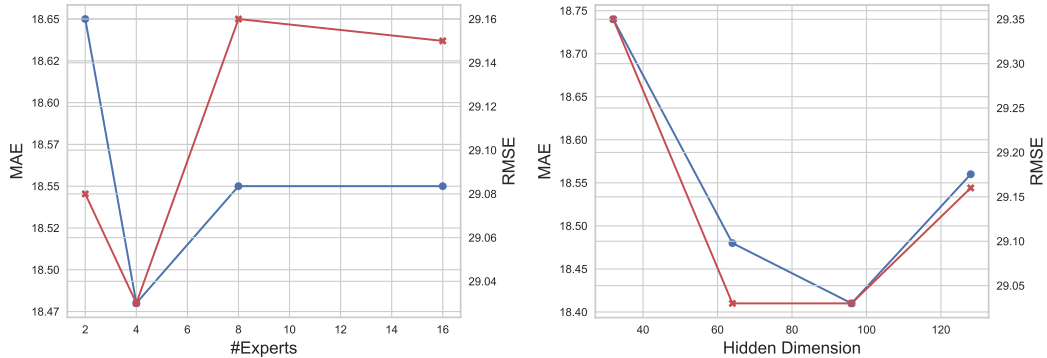


Figure 6: Hyper-parameter study.

958 D.2 Further Analysis

959 We analyze task-specific differences in node degree and perceptual radius distributions for short-
 960 and long-term predictions on the SINPA dataset. Short-term predictions exhibit a sharp node degree
 961 distribution peaking, indicating reliance on highly connected nodes to capture dense local interactions.
 962 In contrast, long-term predictions show a flatter distribution peaking below 500, suggesting sparser
 963 node sampling to prioritize broader contextual patterns over fine-grained dynamics. Short-term
 964 predictions concentrate around moderate radii, balancing localized state changes and mid-range
 965 dependencies. Long-term predictions, however, favor smaller radii with greater dispersion, reflecting
 966 adaptive trade-offs: smaller radii suppress noise from distant nodes, while occasional long-range
 967 connections capture systemic shifts, such as holiday-driven destination changes. These divergences
 968 reveal that the model dynamically tailors its spatial aggregation strategy to temporal scope: short-term
 969 tasks emphasize resolution-rich local dynamics, whereas long-term tasks hierarchically integrate
 970 multi-scale dependencies at lower resolution. The distinct distributions further validate the necessity
 971 of task-aware spatial perception mechanisms in spatio-temporal forecasting.

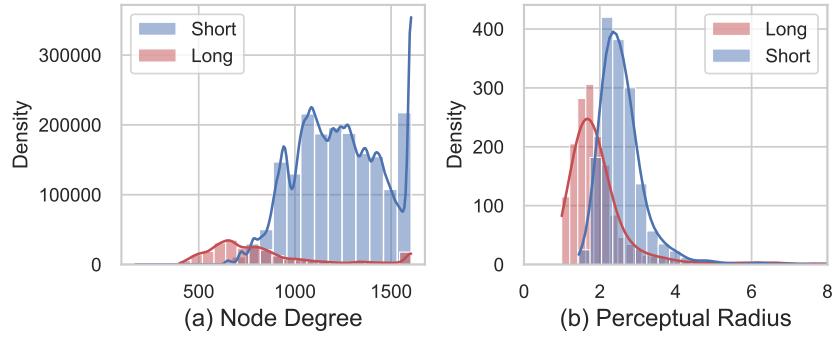


Figure 7: Node property distributions for short-term (blue) and long-term (red) forecasting.

972 E More Discussion

973 **Limitations.** While FlowNet achieves state-of-the-art prediction accuracy, its computational effi-
 974 ciency is constrained by pairwise operations. Specifically, the Flow Allocation operation incurs an
 975 $O(N^2)$ complexity due to node-wise flow redistribution, and the Flow Evolution Module (FEM)
 976 scales quadratically with the prediction horizon ($O(T^2)$). For systems with large node counts or
 977 long-term forecasting tasks, FlowNet remains less efficient than STGNN-based methods, though it
 978 outperforms Transformer-based models in both speed and memory usage. We argue that the accuracy
 979 gains justify this trade-off in many real-world applications. Future work will explore sparse flow
 980 tokenization and hierarchical grouping to mitigate these bottlenecks.

981 **Societal Impacts.** FlowNet’s ability to model flow-driven dynamics can benefit urban planning, envi-
 982 ronmental protection, and disaster response. The framework enhances interpretability by explicitly
 983 quantifying flow exchanges, enabling policymakers to design targeted regulations. Furthermore, its
 984 conservation-law alignment promotes physically plausible predictions, reducing risks of harmful
 985 decisions based on spurious correlations.