Computer Communications 258 (2026) 108635

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Expert-wise federated learning with sparse routing for distributed traffic prediction☆

Yetong Wang [a], Wenze Xiong [b], Wanxin Li [a,*], Hao Guo [c], Jie Zhang [a,*], Nguyen Van Huynh [d], Mark Nejad [e]

[a] School of Advanced Technology, Xi'an Jiaotong-Liverpool University, 111 Ren'ai Road, Suzhou, 215123, Jiangsu, China
[b] Business School, University of Auckland, 12 Grafton Road, Auckland, 1010, Auckland, New Zealand
[c] School of Software, Northwestern Polytechnical University, Taicang Campus, Suzhou, 215400, Jiangsu, China
[d] School of Computer Science and Informatics, University of Liverpool, L69 3DR, Liverpool, UK
[e] Department of Civil and Environmental Engineering, University of Delaware, Newark, 19716, DE, USA

ARTICLE INFO

Keywords:
Federated learning
Edge computing
Mixture of experts
Traffic forecasting
Smart mobility
Urban computing

ABSTRACT

Accurate traffic prediction in distributed smart mobility environments remains challenging because local traffic streams exhibit heterogeneous temporal patterns, while collaborative model training must operate across distributed regional settings. Existing federated learning (FL) methods typically rely on full-model aggregation. This can mix updates associated with different temporal roles, thereby producing averaged parameters that fail to represent either recurrent or irregular traffic dynamics. To address the above challenge, we propose Federated Sparse Routing for Federated Learning (FSR-FL), a federated Mixture-of-Experts framework for distributed traffic prediction. FSR-FL introduces a dynamics-informed routing mechanism to select between recurrent and non-recurrent experts, and an expert-wise aggregation strategy that synchronizes shared experts according to their actual utilization. In this way, the framework separates heterogeneous temporal dynamics locally while maintaining structured collaboration through shared-expert synchronization rather than full-model aggregation. Experiments on the METR-LA and Luzern datasets show that FSR-FL reduces prediction RMSE by up to 5.6% and 6.7%, respectively, compared with standard federated baselines, while remaining effective across diverse traffic patterns. These properties make the framework suitable for privacy-preserving urban computing and edge-side intelligent transportation applications.

1. Introduction

Modern urban mobility sensing systems are increasingly deployed across geographically distributed regions, where sensing, storage, and prediction are often distributed across multiple local infrastructures rather than within a single centralized platform. In such settings, traffic forecasting is not only an application problem but also a distributed learning problem, because prediction models must operate across regional clients while respecting data locality. Accurate short and medium term forecasting remains important for congestion mitigation, signal coordination, public transport operation, and broader urban mobility management [1–3]. Recent deep learning models have shown strong predictive capability for traffic analysis, from congestion-oriented architectures [4] to integrated network-wide forecasting frameworks [5]. These results confirm the value of data-driven prediction for intelligent transportation systems [6], but most existing methods are still developed under centralized training assumptions.

This assumption is difficult to maintain in realistic regional deployments. Traffic data are naturally distributed across regions and organizations, which limits direct data sharing in practice [7]. At the same time, traffic streams are highly heterogeneous, with temporal patterns shaped by recurring cycles and irregular events such as accidents and extreme weather [8]. Therefore, a collaborative forecasting framework for regional traffic systems must address two requirements simultaneously: preserving local data on edge-side clients and supporting effective coordination across heterogeneous environments. More broadly, this makes traffic prediction not only a forecasting problem, but also an urban computing problem in which distributed sensing regions must support localized analytics and cross-region collaboration.

☆ This article is part of a Special issue entitled: 'Next-Generation IoT Urban Computing' published in Computer Communications.

* Corresponding authors.
E-mail addresses: yetong.wang19@student.xjtlu.edu.cn (Y. Wang), wenze.xiong@auckland.ac.nz (W. Xiong), wanxin.li@xjtlu.edu.cn (W. Li), haoguo@nwpu.edu.cn (H. Guo), jie.zhang01@xjtlu.edu.cn (J. Zhang), huynh.nguyen@liverpool.ac.uk (N. Van Huynh), nejad@udel.edu (M. Nejad).

https://doi.org/10.1016/j.comcom.2026.108635
Received 14 April 2026; Received in revised form 9 July 2026; Accepted 31 July 2026
Available online 7 August 2026

Federated learning (FL) provides a natural basis for such collaboration by allowing clients to train locally and exchange model updates without exposing raw measurements [9]. Several studies have shown that FL can achieve competitive forecasting performance in distributed traffic settings [10]. Nevertheless, standard FL typically relies on full-model aggregation, which can be ineffective when clients exhibit substantially different traffic regimes. Under heterogeneous conditions, client updates may interfere with one another, biasing the global model toward averaged behaviors that do not adequately represent local dynamics [11]. Personalized and multi-task variants partially alleviate this issue, but they still commonly operate at the whole-model or whole-task level rather than through structured coordination of model components [12]. The main limitation considered in this work is not merely the accuracy drop of standard FL, but the role inconsistency caused by full model aggregation. In heterogeneous traffic environments, the same shared parameters may be updated by recurrent peak hour patterns on one client and by event-driven or weather affected fluctuations on another client. Averaging these updates can produce a global model that fits an averaged trend but does not explicitly preserve the physical distinction between stable periodic behavior and irregular local disturbances. This also weakens interpretability, since a full model does not indicate which temporal regime contributes to a prediction or which regime is being transferred across clients.

Mixture-of-Experts (MoE) offers a complementary mechanism for handling heterogeneous temporal patterns by assigning different inputs to specialized experts [13,14]. Through gating, MoE can distinguish routine and irregular traffic behaviors, thereby reducing interference between competing patterns [15]. This expert specialization has shown promise for traffic forecasting and can improve interpretability by leveraging routing behavior [16]. However, most existing MoE-based traffic models assume centralized training, while current FL-MoE studies remain limited in traffic settings [17]. In particular, uniform aggregation across experts can blur expert-specific roles and weaken the benefit of specialization [18]. Moreover, some federated traffic methods rely on additional spatial or graph information for coordination [19,20], which may increase the exposure of sensitive network topology. These limitations motivate a more modular federated design in which local specialization is preserved while only transferable expert knowledge is synchronized.

To address these limitations, we propose Federated Sparse Routing for Federated Learning (FSR-FL), a federated Mixture-of-Experts framework for distributed traffic prediction across heterogeneous regional clients. In this work, we distinguish between two fundamental traffic regimes: (i) recurrent patterns, which refer to regular and predictable behaviors such as daily and weekly cycles, and (ii) non-recurrent patterns, which encompass irregular, event-driven fluctuations caused by incidents, special events, or extreme weather. FSR-FL models these regimes using two specialized experts, keeps routing local for client-side adaptation, and performs expert-wise aggregation on the server based on actual expert utilization. This partial-sharing design enables structured global coordination without resorting to full-model aggregation, making it better suited to heterogeneous edge-oriented deployments. Our main contributions are summarized as follows:

- We propose FSR-FL, a federated Mixture-of-Experts framework for distributed traffic prediction across heterogeneous regional clients. It separates traffic dynamics into specialized experts and employs a partial-sharing design to balance global collaboration with local adaptation in smart mobility-oriented edge environments.
- We design a dynamics-informed routing mechanism and an expert-wise aggregation strategy. The routing module guides expert selection based on traffic volatility, while the aggregation strategy performs structured, expert-level synchronization based on utilization frequency, ensuring that global updates more accurately reflect valid local patterns.
- We evaluate the proposed framework on the METR-LA dataset and, for the first time in this context, the Luzern dataset. Extensive experiments show that FSR-FL consistently outperforms comprehensive baselines in both short-term and medium-term forecasting scenarios under offline and simulated online settings relevant to intelligent transportation applications.

2. Related work

2.1. Centralized traffic forecasting models

Traffic forecasting has been studied through a broad range of centralized methods, from classical statistical models to recent neural architectures. Classical approaches remain useful as lightweight and interpretable references. Representative examples include seasonal ARIMA, Kalman-filter-based prediction, exponential smoothing variants, pattern matching methods such as enhanced K-nearest neighbor forecasting, and support vector regression [21–25]. These methods are effective when periodicity is strong or online updates are required, but their capacity is limited under regime shifts, nonlinearity, and high-dimensional cross-sensor dependence.

Learning-based forecasting models improved temporal and spatial representation power. Recurrent models such as GRUs remain widely used for stable temporal modeling with moderate context lengths [26], while temporal convolutions enable efficient parallel learning of local patterns [27]. Spatiotemporal graph learning further models cross-sensor interactions, with representative examples including diffusion-style graph recurrence, Graph WaveNet, and AGCRN [28–30]. At the same time, a critical review has shown that graph representations should not be treated as universally beneficial, because performance depends strongly on how network structure is defined and integrated [31].

Recent studies have also explored long-context and lightweight alternatives. Informer is a widely used, efficient attention architecture [32], while empirical analyses suggest that simpler models can remain competitive on long-horizon benchmarks [33]. In the same direction, compact architectures such as GLinear and SFNN show that efficient forecasting can still be achieved without highly complex structural dependencies [34,35]. In parallel, Mixture-of-Experts (MoE) has been revisited for traffic forecasting to separate heterogeneous patterns across time and space. Existing designs, including spatiotemporal MoE and cascading expert models, indicate that expert specialization can better capture diverse temporal regimes and provide a clearer basis for interpretation [13–15]. However, these models are still developed under centralized training assumptions.

2.2. Federated and modular collaborative forecasting

When data are distributed across regions, federated learning provides a practical framework for collaborative forecasting without sharing raw data. Prior work includes federated traffic prediction [36,37] and newer designs that emphasize spatiotemporal dependency, such as online correlation-based federation and inter-client spatial dependency modeling for federated graph prediction [19,20]. Cross-city personalization has also been explored [12]. Other studies further consider edge deployment, blockchain-assisted federation, and decentralized secure coordination for traffic prediction [38,39]. These studies show that federated collaboration is feasible in traffic settings, but most methods still coordinate knowledge at the full-model level.

This limitation becomes more evident under strong client heterogeneity. When local traffic regimes differ substantially across regions, full-model synchronization may mix updates that serve different functional roles, thereby weakening local adaptation and reducing the interpretability of the shared representation. Recent personalized and modular FL methods have introduced client-specific heads, layer-wise

Table 1
Comparative analysis of proposed FSR-FL with existing methods.

Category	Representative works	Time complexity	Space complexity	Comm. cost	Personalization	Privacy	Heterogeneity	Lightweight
Classical methods	Kumar and Vanajakshi [21], Emami et al. [22], Kumar et al. [23], Habtemichael and Cetin [24] and Lin et al. [25]	$O(T) \sim O(N^3)$	$O(T) \sim O(N^2)$	–	○	○	○	●
Centralized deep learning	Zhao et al. [27], Dey and Salem [26], Li et al. [28], Bai et al. [30], Hussain Rizvi et al. [34] and Sun et al. [35]	$O(LD) \sim O(LD^2K)$	$O(D) \sim O(LDK)$	–	○	○	○	◗
Transformer	Zhou et al. [32]	$O(L \log L)$	$O(L \log L)$	–	○	○	○	○
Federated learning	McMahan et al. [9], Liu et al. [36], Liu et al. [37], Liu et al. [19], Yang et al. [20], Zhang et al. [12], Meese et al. [38] and Guo et al. [39]	$O(CELD^2) \sim O(CELND^2)$	$O(\lvert\theta\rvert)$	$O(RC\lvert\theta\rvert)$	◗	●	◗	○
Centralized MoE	Ke et al. [13] and Li et al. [14]	$O(LMD^2)$	$O(MD^2)$	–	●	○	●	◗
Proposed	**FSR-FL**	$\mathbf{O(EnLD^2K)}$	$\mathbf{O(D^2)}$	$\mathbf{O(RCP_{exp})}$	●	●	●	●

Notation: L: sequence length; D: hidden dimension; N: number of nodes; K: kernel size; T: time series length; C: number of clients; E: number of local epochs; M: number of experts; R: communication rounds; n: number of samples; S: state dimension; $|\theta|$: total parameters; P_{exp}: shared expert parameters ($P_{exp} \ll |\theta|$). The symbols ●(full support), ◗(partial support), and ○(no support) indicate the level of support for each characteristic; "–" denotes not applicable.

aggregation, and modular parameter partitioning [12,40]. These methods improve client-level adaptation, but their personalization is usually tied to clients, tasks, or fixed model partitions, rather than to the changing temporal role of each local window.

Federated MoE methods are also relevant to heterogeneous FL. FedMix learns user-specific selections of expert models, while recent FedMoE-style methods construct personalized sub-MoE structures for heterogeneous clients [41,42]. However, these methods are mainly designed for user-level expert selection or large-model adaptation, and are not specifically developed for traffic forecasting. In regional traffic prediction, heterogeneity may vary over time within the same client, since regular intervals and irregular conditions can require different temporal modeling roles. Centralized MoE-based traffic models have shown that expert specialization can separate heterogeneous temporal patterns [13–15], but they do not address how such expert knowledge should be coordinated across distributed clients. This leaves a gap between traffic-regime specialization and federated collaboration under data-locality constraints [43,44].

2.3. Comparison with existing works

To differentiate FSR-FL from existing approaches, Table 1 presents a comparative analysis across multiple key design dimensions. Specifically, Personalization evaluates the model's adaptability to local data patterns, Privacy preserves sensitive information, Heterogeneity assesses the capability to handle diverse distributions, and Lightweight reflects the suitability for resource-constrained deployment. In terms of computational efficiency, FSR-FL optimizes space complexity to $O(D^2)$, a significant reduction compared to the $O(|\theta|)$ typically required by standard Federated Learning. Furthermore, its communication cost is bounded by $O(RCP_{exp})$, where $P_{exp} \ll |\theta|$, ensuring scalability in bandwidth-constrained environments. While classical methods offer lightweight profiles with $O(N^2)$ space complexity, they lack task-specific adaptability. Conversely, Centralized MoE (M experts) and Transformers handle data heterogeneity and personalization through architectural depth but require data consolidation, thereby failing to meet privacy requirements. Standard Federated Learning enables distributed training (E epochs) but often struggles with localized adaptation. FSR-FL addresses these limitations by integrating a personalized routing mechanism with expert-wise federation, achieving a superior balance of $O(EnLD^2\ K)$ time efficiency, robust privacy protection, and personalization.

3. System overview

The proposed FSR-FL framework is structured into three hierarchical layers to facilitate collaborative traffic forecasting across heterogeneous regions, as illustrated in Fig. 1. At the foundational Physical Traffic Layer, traffic sensors continuously collect raw data streams from distributed regions. To maintain data locality and security, these distinct regional datasets are stored and processed exclusively on local edge devices (e.g., roadside units), ensuring that raw information never leaves its source.

The middle layer consists of the Client MoE Architecture, where the local learning process takes place. Clients are flexibly deployed across geographic areas, each serving as a regional edge intelligence node for urban mobility management. Each client employs a specialized structure designed to handle diverse data patterns. As shown in the figure, this architecture contains two distinct modules: Expert A, which captures recurrent periodic patterns, and Expert B, which models non-recurrent changes. A local gating network analyzes the input features to dynamically assign importance weights and route the data to the appropriate expert. This design allows the client to adapt to specific local traffic conditions before generating the final prediction.

The top layer features the FL Server, which coordinates the global collaboration. Clients upload their model updates to the server, which performs an expert-wise aggregation. This mechanism synchronizes Expert A and Expert B separately, based on their respective utilization frequencies across the network. The server then distributes the refined global parameters back to the clients, completing the workflow from data input to the final collaborative output. From a distributed systems perspective, the framework separates local specialization from shared synchronization by confining routing and output adaptation to clients while limiting inter-client coordination to shared-expert synchronization.

4. Methodology

4.1. Problem formulation

Consider a federated system with clients from different regions indexed by $k \in \{1, \ldots, K\}$. Each client k stores a local multivariate traffic stream and optional exogenous variables. Let t denote the global time index, L the look-back window length and forecasting horizon H. Before constructing local forecasting windows, each client maps its observations to a unified temporal grid defined by the data source. Optional exogenous variables are aligned to the same timestamps, and missing values are completed locally using a fixed preprocessing rule.

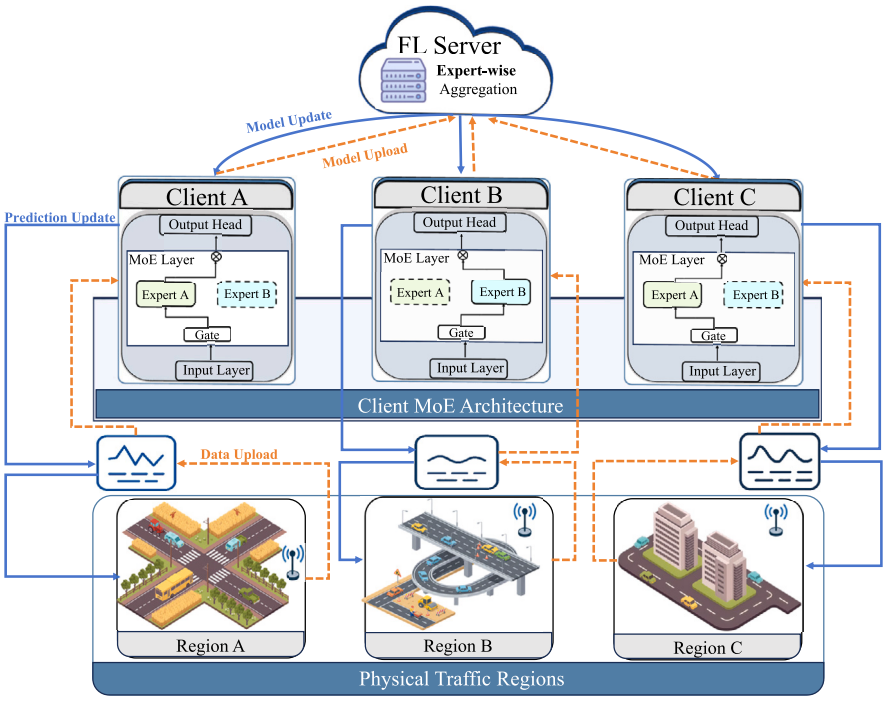


Fig. 1. The overall system architecture of the proposed FSR-FL framework for multi-region traffic prediction.

All participating clients use the same temporal resolution, look-back length L, and forecasting horizon H for window construction. The effective sequence segment used to generate local samples is also aligned across participating clients, ensuring that the constructed windows are comparable in length and chronological coverage.

4.1.1. Input construction

For client k at time t, the traffic observation is represented by a vector $\boldsymbol{s}_{k,t} \in \mathbb{R}^{d_s}$ and the exogenous information is represented by a vector $\boldsymbol{e}_{k,t} \in \mathbb{R}^{d_e}$. The traffic history and exogenous history over a window of length L are stacked as Eqs. (1) and (2)

$$\boldsymbol{S}_{k,t} = \left[\boldsymbol{s}_{k,t-L+1}, \dots, \boldsymbol{s}_{k,t}\right] \in \mathbb{R}^{L\times d_s}, \tag{1}$$

$$\boldsymbol{E}_{k,t} = \left[\boldsymbol{e}_{k,t-L+1}, \dots, \boldsymbol{e}_{k,t}\right] \in \mathbb{R}^{L\times d_e}. \tag{2}$$

The model input $\boldsymbol{X}_{k,t}$ is obtained by concatenating $\boldsymbol{S}_{k,t}$ and $\boldsymbol{E}_{k,t}$ along the feature dimension, as defined in Eq. (3).

$$\boldsymbol{X}_{k,t} = \left[\boldsymbol{S}_{k,t}, \boldsymbol{E}_{k,t}\right] \in \mathbb{R}^{L\times(d_s+d_e)}. \tag{3}$$

4.1.2. Prediction target

Given the input window $\boldsymbol{X}_{k,t}$ in Eq. (3), the forecasting task is to predict the next H traffic vectors, forming the target sequence $\boldsymbol{Y}_{k,t}$ in Eq. (4).

$$\boldsymbol{Y}_{k,t} = \left[\boldsymbol{s}_{k,t+1}, \dots, \boldsymbol{s}_{k,t+H}\right] \in \mathbb{R}^{H\times d_s}, \tag{4}$$

Each client k then constructs its local dataset as $D_k = \{(\boldsymbol{X}_{k,t}, \boldsymbol{Y}_{k,t})\}$. Based on these local datasets, the federated learning objective is defined in Eq. (5), where the server optimizes the global parameter set Θ by minimizing the weighted average loss across clients while keeping raw data local.

$$\min_{\Theta} \sum_{k=1}^{K} \frac{|D_k|}{\sum_{j=1}^{K}|D_j|}\, \mathbb{E}_{(\boldsymbol{X},\boldsymbol{Y})\sim D_k}\left[\ell\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right)\right]. \tag{5}$$

Here, Θ denotes the trainable model parameters, which will be divided into shared and private components. The loss function ℓ quantifies the prediction error between the ground-truth sequence $\boldsymbol{Y}$ and the predicted output $\hat{\boldsymbol{Y}}$.

4.2. Framework overview of FSR-FL

Building on the system-level architecture in Section 3, Fig. 2 summarizes the model-side workflow of FSR-FL from local input construction to expert-wise federated synchronization. Specifically, each client performs **(1) temporal windowing and feature construction** to align raw traffic streams with exogenous variables, and forms **(2) the input time window and features** $\boldsymbol{X}_{k,t}$. To expose short-term regime variation, the client further applies **(3) input decomposition**, where temporal differences are computed as

$$\Delta\boldsymbol{X}_{k,t} = \left[\boldsymbol{x}_{k,t-L+2} - \boldsymbol{x}_{k,t-L+1}, \dots, \boldsymbol{x}_{k,t} - \boldsymbol{x}_{k,t-1}\right]. \tag{6}$$

Next, **(4) the routing network** uses a client private encoder to extract local dynamic representations. A routing head maps these representations to expert selection probabilities, while the volatility regression head provides an auxiliary regularization signal that encourages the encoder to retain local fluctuation information. Based on these routing probabilities, **(5) the shared experts and federated server** coordinate a Recurrent Expert and a Non-Recurrent Expert, whose outputs are combined while only expert parameters are synchronized across clients. Finally, **(6) the local output** maps the mixed expert representation to the prediction $\hat{\boldsymbol{Y}}_{k,t}$. In this way, FSR-FL follows a partial-sharing design: decomposition, routing, and output mapping remain local, whereas transferable temporal knowledge is exchanged only through shared-expert aggregation.

4.3. Dynamics-informed gating via auxiliary volatility regression

To characterize window-level traffic fluctuation, we compute a scalar volatility target from the traffic sequence rather than from the full feature vector. Let $\bar{s}_{k,t,i}$ denote the mean of the ith traffic variable over the look-back window L. The volatility target is defined as the average temporal standard deviation over the traffic variables:

$$v_{k,t} = \frac{1}{d_s}\sum_{i=1}^{d_s}\sqrt{\frac{1}{L}\sum_{\tau=t-L+1}^{t}\left(s_{k,\tau,i} - \bar{s}_{k,t,i}\right)^2} \tag{7}$$

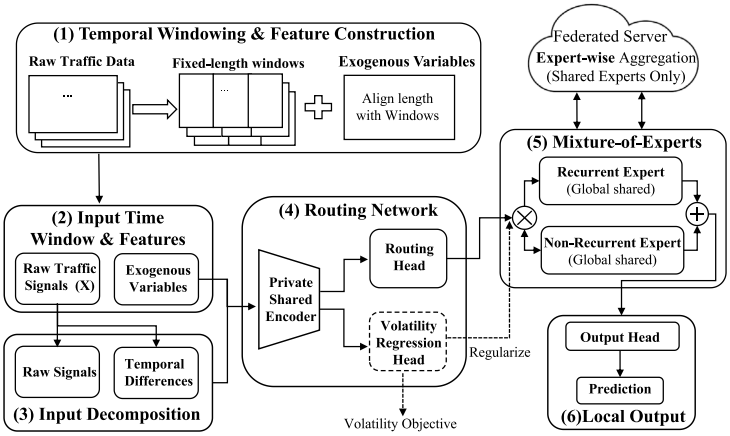


Fig. 2. An overview of the FSR-FL framework with dynamics-informed routing and expert-wise aggregation.

where d_s is the number of traffic variables. The target in equation. (7) is used as a self-supervised auxiliary regularization signal for the client-private routing encoder. It should not be interpreted as a hard label for non-recurrent events, since recurrent peak periods may also exhibit high fluctuation and sustained disruptions may have low short-term variation. When exogenous covariates are available, they are used as conditioning inputs to the routing network, but they are not included in the volatility target.

This definition is chosen for three reasons. First, it is invariant to local level shifts within a window, since it measures deviations from the window mean. Second, it is permutation-invariant with respect to the ordering of traffic variables, thereby avoiding the assignment of a fixed priority to a specific client or flow dimension. Third, it provides a stable scalar target for auxiliary regression. For two traffic windows S and S', the target satisfies

$$\left|v(S) - v(S')\right| \le \frac{1}{\sqrt{Ld_s}} \left\|S - S'\right\|_F . \tag{8}$$

Thus, small perturbations in the input window lead to bounded changes in the auxiliary target. The scalar target may smooth feature-specific spikes, but this is intentional because the volatility head is not designed as a feature-wise anomaly detector. Feature-specific information is still available to the gate through the latent representation $h_{k,t}$ and the explicit difference term $\Delta X_{k,t}$.

A client-private encoder $\phi_k(\cdot;\psi_k)$, parameterized by ψ_k, maps the raw input window to a latent representation $\boldsymbol{h}_{k,t} \in \mathbb{R}^{d_h}$, as defined in Eq. (9):

$$\boldsymbol{h}_{k,t} = \phi_k(\boldsymbol{X}_{k,t};\psi_k), \tag{9}$$

Based on $\boldsymbol{h}_{k,t}$ and the decomposed residual view $\Delta\boldsymbol{X}_{k,t}$, the private gating network $g_k(\cdot;\omega_k)$ produces the routing distribution over the experts. As shown in Eq. (10), the output vector $\boldsymbol{\pi}_{k,t}$ contains the mixture weights for the Recurrent (R) and Non-recurrent (N) experts:

$$\boldsymbol{\pi}_{k,t} = \left[\pi_{k,t}(R), \pi_{k,t}(N)\right] = g_k(\boldsymbol{h}_{k,t}, \Delta\boldsymbol{X}_{k,t};\omega_k) \in \mathbb{R}^2, \tag{10}$$

The explicit first-order difference term is included because it provides a direct channel for local change in the gate. Although the hidden representation extracted from $X_{k,t}$ contains temporal information, it is optimized primarily for prediction and may smooth short-term increments. The difference term captures local temporal variation prior to representation learning and is a common preprocessing signal to make time-series dynamics more stationary. Therefore, $X_{k,t}$ and $\Delta X_{k,t}$ play complementary roles: the former preserves the level and context of traffic states, while the latter highlights short-term changes that are important for routing.

The routing coefficients are constrained to the probability simplex in (11), so that all elements are non-negative and sum to one:

$$\forall m \in R, N, \quad \pi_{k,t}(m) \ge 0, \quad \text{and} \quad \sum_{m\in R,N} \pi_{k,t}(m) = 1. \tag{11}$$

To inject dynamics awareness into the routing pathway, we introduce a lightweight auxiliary regressor $a_k(\cdot;\eta_k)$ with parameters η_k. It predicts the window volatility $\hat{v}_{k,t}$ from the latent representation $\boldsymbol{h}_{k,t}$, as shown in Eq. (12):

$$\hat{v}_{k,t} = a_k(\boldsymbol{h}_{k,t};\eta_k) \in \mathbb{R}, \tag{12}$$

Its training objective is to reduce the discrepancy between the predicted volatility and the locally computed target $v_{k,t}$ from Eq. (7). For a local mini-batch $\mathcal{B}_k$, the auxiliary loss $\mathcal{L}_{\text{vol}}^{(k)}$ is defined in Eq. (13):

$$\mathcal{L}_{\text{vol}}^{(k)} = \frac{1}{|\mathcal{B}_k|} \sum_{t\in\mathcal{B}_k} \left\|\hat{v}_{k,t} - v_{k,t}\right\|_2^2 . \tag{13}$$

This encourages the encoder to capture traffic fluctuations and provides the gating network with dynamics-aware features for expert selection.

4.3.1. Routing regularization

We apply two regularization terms to the routing distribution to avoid trivial solutions and improve training stability.

Entropy regularization. We penalize high-entropy routing distributions to discourage uniform expert weights and promote more decisive routing. The entropy loss $\mathcal{L}_{\text{ent}}^{(k)}$ is computed over the batch as shown in Eq. (14):

$$\mathcal{L}_{\text{ent}}^{(k)} = \frac{1}{|\mathcal{B}_k|} \sum_{t\in\mathcal{B}_k} \left(- \sum_{m\in\{R,N\}} \pi_{k,t}(m) \log\left(\pi_{k,t}(m) + \epsilon\right) \right), \tag{14}$$

where $\epsilon > 0$ is a small constant for numerical stability.

Target utilization regularization. To reduce mode collapse, where the gate overuses a single expert, we introduce a load-balancing term that encourages the average routing behavior to match a predefined target distribution $\rho \in \mathbb{R}^2$, satisfying $\sum_{m\in\{R,N\}} \rho(m) = 1$. We first compute the average routing distribution $\bar{\boldsymbol{\pi}}_k$ over the mini-batch as shown in Eq. (15):

$$\bar{\boldsymbol{\pi}}_k = \frac{1}{|\mathcal{B}_k|} \sum_{t\in\mathcal{B}_k} \boldsymbol{\pi}_{k,t}. \tag{15}$$

The deviation from the target ρ is then penalized by the balance loss in Eq. (16):

$$\mathcal{L}_{\text{bal}}^{(k)} = \left\|\bar{\boldsymbol{\pi}}_k - \rho\right\|_2^2 , \tag{16}$$

The target ρ reflects the expected proportion of recurrent and non-recurrent patterns in the deployment environment.

4.4. Hybrid mixture-of-experts formulation

4.4.1. Shared experts

The framework uses two shared experts to capture complementary temporal properties. Experts are designed with different structural biases rather than as identical networks with different labels. The recurrent expert represents sequence encoders with temporal state modeling ability, while the non-recurrent expert represents short-term temporal encoders that emphasize local variations and short-term deviations. Other architectures with the same roles for recurrent and local dynamics can also be used within the proposed framework. The Recurrent Expert, parameterized by θ_R, models sequential dependencies and periodic patterns. Its output representation $\boldsymbol{z}_{k,t}(R)$ is defined in Eq. (17):

$$\boldsymbol{z}_{k,t}(R) = f_R(\boldsymbol{h}_{k,t}; \theta_R) \in \mathbb{R}^{d_z}, \tag{17}$$

The Non-Recurrent Expert, parameterized by θ_N, captures instantaneous fluctuations and irregular events. Its output $\boldsymbol{z}_{k,t}(N)$ is defined in Eq. (18):

$$\boldsymbol{z}_{k,t}(N) = f_N(\boldsymbol{h}_{k,t}; \theta_N) \in \mathbb{R}^{d_z}, \tag{18}$$

where $\boldsymbol{h}_{k,t}$ is the client-side latent representation and d_z denotes the expert feature dimension.

4.4.2. Sparse routing and prediction

The expert outputs are combined into a unified representation through sparse routing. Let $S_{k,t}$ denote the set of active experts selected according to the top-k gating probabilities (e.g., $k = 1$ or $k = 2$). The mixed representation $\boldsymbol{z}_{k,t}$ is computed as the convex combination in Eq. (19):

$$\boldsymbol{z}_{k,t} = \sum_{m \in S_{k,t}} \pi_{k,t}(m) \boldsymbol{z}_{k,t}(m), \tag{19}$$

This formulation preserves generality for different sparsity levels. In this work, we primarily adopt a **Top 1 strategy** ($|S_{k,t}| = 1$) to reduce inference overhead, so that only the dominant expert branch is executed while its output remains scaled by the gating probability $\pi_{k,t}(m)$. The routing layer is implemented as a deterministic sparse operator. The gate first computes continuous expert probabilities through a softmax layer, and the Top 1 operator then selects the active expert index. Conditional on the selected route, the forward output remains differentiable with respect to the selected expert parameters and the selected gate probability. Therefore, during backpropagation, gradients are propagated through the active expert branch and the corresponding softmax probabilities, while the selected index is treated as a piecewise-constant routing decision. The same routing rule is used in both training and inference, thereby avoiding sampling variance and keeping the sparse execution behavior consistent.

The mixed representation is then mapped to the final forecast by a client-specific output head $o_k(\cdot; \xi_k)$, parameterized by ξ_k, as described in Eq. (20):

$$\hat{\boldsymbol{Y}}_{k,t} = o_k(\boldsymbol{z}_{k,t}; \xi_k) \in \mathbb{R}^{H \times d_s}, \tag{20}$$

For a local mini-batch $\mathcal{B}_k$, the prediction loss $\mathcal{L}^{(k)}_{\text{pred}}$ is defined in Eq. (21):

$$\mathcal{L}^{(k)}_{\text{pred}} = \frac{1}{|\mathcal{B}_k|} \sum_{t \in \mathcal{B}_k} \left[\alpha \, \ell_{\text{Huber}}(\hat{\boldsymbol{Y}}_{k,t}, \boldsymbol{Y}_{k,t}) + (1-\alpha) \, \|\hat{\boldsymbol{Y}}_{k,t} - \boldsymbol{Y}_{k,t}\|_1 \right], \tag{21}$$

where $\boldsymbol{Y}_{k,t}$ is the ground truth, and $\alpha \in [0,1]$ balances the Huber loss and the ℓ_1 norm.

4.4.3. Full local objective

The local training objective combines the forecasting loss with the auxiliary volatility supervision and routing regularization terms. The total loss $\mathcal{L}^{(k)}$ is defined in Eq. (22):

$$\mathcal{L}^{(k)} = \mathcal{L}^{(k)}_{\text{pred}} + \lambda_{\text{vol}} \mathcal{L}^{(k)}_{\text{vol}} + \lambda_{\text{ent}} \mathcal{L}^{(k)}_{\text{ent}} + \lambda_{\text{bal}} \mathcal{L}^{(k)}_{\text{bal}}. \tag{22}$$

where λ_{vol}, λ_{ent}, and λ_{bal} are non-negative coefficients controlling the contributions of volatility regression, entropy regularization, and load balancing, respectively.

4.5. Expert-wise federated aggregation

4.5.1. Partial parameter sharing

The full parameter set of client k is denoted by Θ_k, as defined in Eq. (23):

$$\Theta_k = \{\psi_k, \omega_k, \eta_k, \xi_k, \theta_R, \theta_N\}, \tag{23}$$

where $\{\psi_k, \omega_k, \eta_k, \xi_k\}$ are the client-private components, namely the encoder, gating network, volatility head, and output head, as introduced in Eqs. (9)–(20). The set $\{\theta_R, \theta_N\}$ contains the shared Recurrent and Non-Recurrent expert parameters defined in Eqs. (17)–(18). Accordingly, FSR-FL separates Θ_k into a private set Θ^{priv}_k and a shared set Θ^{sh}. The private parameters are defined in Eq. (24):

$$\Theta^{\text{priv}}_k = \{\psi_k, \omega_k, \eta_k, \xi_k\}, \tag{24}$$

and the shared parameters participating in federation are defined in Eq. (25):

$$\Theta^{\text{sh}} = \{\theta_R, \theta_N\}, \tag{25}$$

During each communication round, only Θ^{sh} is exchanged between clients and the server, while the routing-related and personalization modules remain local.

4.5.2. Expert-specific aggregation mechanism

Let r denote the current communication round, and let $\mathcal{K}^r \subseteq \{1, \ldots, K\}$ be the set of participating clients. After local training, each client $k \in \mathcal{K}^r$ returns updated expert parameters $\theta^{r+1}_{R,k}$ and $\theta^{r+1}_{N,k}$. To guide aggregation, client k computes expert utilization statistics from the routing probabilities $\boldsymbol{\pi}_{k,t}$ in Eq. (10).

Expert utilization frequency. Let $\mathcal{B}^r_k$ denote the set of training sample indices processed by client k during round r. We first assign each sample to the expert with the highest routing probability, as defined in Eq. (26):

$$\hat{m}_{k,t} = \arg\max_{m \in \{R,N\}} \pi_{k,t}(m), \qquad t \in \mathcal{B}^r_k, \tag{26}$$

The utilization frequency $u^r_{m,k}$ for expert $m \in \{R, N\}$ is then computed by Eq. (27):

$$u^r_{m,k} = \frac{1}{|\mathcal{B}^r_k|} \sum_{t \in \mathcal{B}^r_k} \mathbb{I}\left(\hat{m}_{k,t} = m\right), \tag{27}$$

where $\mathbb{I}(\cdot)$ is the indicator function. By definition, $u^r_{R,k} + u^r_{N,k} = 1$.

Utilization-weighted aggregation. The server aggregates each expert using weights that reflect both local data volume and expert utilization. Let $n^r_k = |\mathcal{B}^r_k|$ be the number of training samples for client k. The aggregation weight $w^r_{m,k}$ is defined in Eq. (28):

$$w^r_{m,k} = n^r_k \, u^r_{m,k}, \qquad m \in \{R, N\}, \; k \in \mathcal{K}^r, \tag{28}$$

The server then updates each global expert independently according to Eq. (29):

$$\theta^{r+1}_m = \frac{\sum_{k \in \mathcal{K}^r} w^r_{m,k} \, \theta^{r+1}_{m,k}}{\sum_{k \in \mathcal{K}^r} w^r_{m,k}}. \tag{29}$$

This makes the contribution of client k to expert m proportional to how often that expert is actually used on the client's local data. It is worth

noticing that $w^r_{m,k} = n^r_k u^r_{m,k}$ represents the effective number of local windows assigned to expert m, rather than a sample-size-only weight. Under the aligned window construction used in this formulation, all participating clients are mapped to the same temporal grid and use the same effective sequence length, temporal resolution, look-back length, and forecasting horizon. Therefore, n^r_k is fixed across clients within a communication round and does not introduce an additional large-sample dominance effect. The expert-wise aggregation weight is consequently modulated by $u^r_{m,k}$, which reflects the actual utilization of expert m on client k.

Zero-utilization safeguard. If no participating client utilizes expert m, the total weight becomes zero, as described in Eq. (30):

$$\sum_{k\in\mathcal{K}^r} w^r_{m,k} = 0, \tag{30}$$

In this case, the server falls back to uniform averaging to avoid division by zero, as shown in Eq. (31):

$$\theta^{r+1}_m = \frac{1}{|\mathcal{K}^r|}\sum_{k\in\mathcal{K}^r} \theta^{r+1}_{m,k}. \tag{31}$$

Algorithm 1 Client-Side Local Update (FSR-FL)

Require: Local dataset $\mathcal{D}_k$, shared experts (θ^r_R, θ^r_N), private parameters Θ^{priv}_k, hyperparameters $\lambda_{\text{vol}}, \lambda_{\text{ent}}, \lambda_{\text{bal}}$
Ensure: Updated experts $(\theta^{r+1}_{R,k}, \theta^{r+1}_{N,k})$, utilization $(u^r_{R,k}, u^r_{N,k})$, sample count n^r_k
Receive (θ^r_R, θ^r_N) and set $\theta_R \leftarrow \theta^r_R$, $\theta_N \leftarrow \theta^r_N$
Initialize an index set $\mathcal{B}^r_k \leftarrow \emptyset$ to track processed samples
for each local epoch **do**
  for each minibatch $\{(\boldsymbol{X}_{k,t}, \boldsymbol{Y}_{k,t})\}_{t\in\mathcal{B}_k}$ **do**
    Update processed indices $\mathcal{B}^r_k \leftarrow \mathcal{B}^r_k \cup \mathcal{B}_k$
    Compute decomposition $\Delta\boldsymbol{X}_{k,t}$ via equation(6) and encoding $\boldsymbol{h}_{k,t}$ via equation(9)
    Compute routing probabilities $\boldsymbol{\pi}_{k,t}$ via equation(10)
    Compute expert outputs via equation(17)–(18) and mixture $\boldsymbol{z}_{k,t}$ via equation(19)
    Compute prediction $\hat{\boldsymbol{Y}}_{k,t}$ via equation(20)
    Compute volatility target $v_{k,t}$ via equation(7) and prediction $\hat{v}_{k,t}$ via equation(12)
    Compute losses: $\mathcal{L}^{(k)}_{\text{pred}}$ (21), $\mathcal{L}^{(k)}_{\text{vol}}$ (13), $\mathcal{L}^{(k)}_{\text{ent}}$ (14), $\mathcal{L}^{(k)}_{\text{bal}}$ (16)
    Minimize $\mathcal{L}^{(k)}$ in equation(22) to update Θ^{priv}_k and local expert copies (θ_R, θ_N)
  end for
end for
Compute hard assignments $\hat{m}_{k,t}$ for all $t \in \mathcal{B}^r_k$ via equation(26)
Compute utilization frequencies $(u^r_{R,k}, u^r_{N,k})$ via equation(27) and set $n^r_k \leftarrow |\mathcal{B}^r_k|$
Upload message $\boldsymbol{m}^r_k = (\theta^{r+1}_{R,k}, \theta^{r+1}_{N,k}, u^r_{R,k}, u^r_{N,k}, n^r_k)$ and keep Θ^{priv}_k local

4.6. Implementation and complexity analysis

The complete training procedure of FSR-FL is summarized in Algorithms 1 and 2. Algorithm 1 describes the client-side update, including local optimization, expert utilization estimation, and upload of expert-specific statistics. Algorithm 2 presents the server-side expert-wise aggregation, where the shared experts are synchronized independently based on utilization-weighted contributions. Together, they implement the partial-sharing mechanism described in previous sections.

Complexity analysis. Let P_{exp} denote the parameter count of the shared experts Θ^{sh}, and let P_{full} denote the total parameter count of the full model. Since only the shared experts are exchanged, the communication volume in round r over the participating clients $\mathcal{K}^r$ scales as in Eq. (32):

$$C^{(r)}_{\text{FSR}} = \mathcal{O}\big(|\mathcal{K}^r|\, P_{\text{exp}}\big), \quad \text{where } P_{\text{exp}} \ll P_{\text{full}}. \tag{32}$$

Algorithm 2 Server-Side Expert-Wise Aggregation (FSR-FL)

Require: Client uploads $\{\boldsymbol{m}^r_k\}_{k\in\mathcal{K}^r}$ from participating clients
Ensure: Aggregated experts $(\theta^{r+1}_R, \theta^{r+1}_N)$
for each expert $m \in \{R, N\}$ **do**
  Compute expert-specific weights $\{w^r_{m,k}\}$ via equation(28)
  if $\sum_{k\in\mathcal{K}^r} w^r_{m,k} > 0$ **then**
    Aggregate expert m via equation(29)
  else
    Safeguard: Aggregate expert m uniformly via equation(31)
  end if
end for
Broadcast updated experts $(\theta^{r+1}_R, \theta^{r+1}_N)$ to all clients

For computation, let E denote the number of local epochs, B the mini-batch size, and n^r_k the number of processed samples. If c_k is the average per-sample cost of a full forward and backward pass through the modules in Eqs. (6)–(20), then the local processing time of client k is given by Eq. (33):

$$T^{(r)}_{\text{cli},k} = \mathcal{O}\big(E\, n^r_k\, c_k\big). \tag{33}$$

The additional cost of computing utilization statistics via Eqs. (26)–(27) is $\mathcal{O}(n^r_k)$, which is negligible compared with gradient-based training. On the server side, Algorithm 2 performs weighted averaging with complexity $\mathcal{O}(|\mathcal{K}^r|\, P_{\text{exp}})$ per round, keeping aggregation lightweight. The server also receives lightweight metadata from each participating client, including the local sample count and expert utilization statistics. This metadata contains only M scalar values per client; in our implementation, $M = 2$, which corresponds to about 8 bytes with 32-bit floats or 16 bytes with 64-bit floats, excluding framework-level transport headers. In contrast, the shared expert payload contains P_{exp} parameters and requires about $4P_{\text{exp}}$ bytes under 32-bit storage. Thus, the metadata scales as $O(|\mathcal{K}^r|M)$, whereas the dominant model payload scales as $O(|\mathcal{K}^r|P_{\text{exp}})$. Since $P_{\text{exp}} \gg M$, the metadata remains a lower-order term and does not change the complexity.

5. Experiments and evaluation

This section evaluates the proposed framework in simulated edge computing environments for distributed intelligent transportation analytics. Following the expanded experimental design, we organize the evaluation into dataset protocol, baselines and implementation settings, evaluation metrics, main forecasting performance, distribution and temporal variation analysis, expert activation behavior, ablation studies, routing efficiency, and sensitivity analysis.

5.1. Datasets and experimental protocol

We evaluate the proposed framework on two benchmark traffic datasets collected from highway loop detectors. The first is METR-LA [45], which contains traffic speed readings in miles per hour (mph) from 207 sensors in Los Angeles County, sampled every 5 min from March 1 to June 30, 2012. The second is the Integrated Urban Traffic-Flood (IUTF) dataset [46], which provides traffic flow measurements in vehicles per hour (vph) at 5-minute resolution together with hourly precipitation data. In this study, we use the Luzern subset (158 sensors), combining local traffic series with city-level average precipitation as an exogenous input. To the best of our knowledge, this is the first use of IUTF in a federated traffic forecasting setting.

To avoid information leakage, all time series are split chronologically, with the first 70% used for training and the remaining 30% for testing. For the IUTF subset, we extract a continuous period with the same duration as METR-LA and apply the same input construction and sampling protocol. Missing values are handled by forward fill, linear

interpolation, and backward fill. After imputation, Z-score normalization is applied using statistics computed only from the training split. For Luzern, precipitation is treated as an exogenous feature and normalized using the same training-only statistics.

To study the effect of data presentation, we consider two training protocols. The offline protocol uses standard batch learning with random shuffling, while the online protocol preserves temporal order by feeding data sequentially without shuffling. The latter simulates an edge device learning from a live traffic stream and allows us to examine prediction and synchronization behavior under more realistic edge-oriented conditions relevant to smart mobility applications.

5.2. Baselines and implementation settings

We compare the proposed method with the following baselines:

TCN. A Temporal Convolutional Network trained locally to represent short-term forecasting without collaboration [27].

GRU. A Gated Recurrent Unit model trained locally as a recurrent baseline for sequential dependency modeling [26].

FedAvg. The standard federated averaging method that aggregates full-model updates as a generic federated baseline [9]. In this study, FedAvg is implemented on the same federated MoE backbone as the proposed method. This setting controls for the effect of the MoE backbone and isolates the role of the proposed aggregation rule.

FedGRU. A federated GRU model in which the server aggregates client-side recurrent parameters into a global predictor [36].

FedTCN. A federated TCN model that aggregates local convolutional updates for distributed traffic prediction [37].

SFNN. A simple feedforward neural network trained locally as a lightweight forecasting baseline [35].

GLinear. A compact linear forecasting model with minor non-linear enhancement [34].

MoE-FedProx. MoE-FedProx applies FedProx to the same federated MoE backbone used by our method [47]. The proximal regularization is applied to the shared MoE layer to reduce local drift under client heterogeneity, while the gate network, feature extractor, and output head remain client specific.

FedMix-style MoE. FedMix trains a set of specialized models and lets each client select a client specific mixture to handle heterogeneous data distributions [41]. For a controlled comparison in our forecasting setting, we implement a lightweight FedMix-style MoE where each client mixes a server aggregated global expert with a private local expert through a local gate.

MoE-pFedMe. MoE-pFedMe adapts pFedMe to the same shared MoE and local module decomposition [48]. Each client maintains a personalized MoE model regularized toward the global shared MoE parameters, providing a personalized federated learning baseline for comparison.

Centralized-MoE. Centralized-MoE is used as a non federated settings to quantify the cost of federation. It trains with centralized access to all Luzern training data, shares the MoE layer across clients, and keeps the gate network, feature extractor, and output head specific.

All experiments were conducted on a local machine with an i9-13900HX CPU, 32 GB RAM, and an 8 GB RTX 4060 GPU using the Flower framework. The proposed backbone is a sparse Mixture-of-Experts designed for resource-constrained clients. The Recurrent Expert is implemented with a single-layer GRU, and the Non-Recurrent Expert is implemented with a TCN block with dilated convolutions. Both experts use a hidden size of 64, resulting in 29,120 shared parameters for the GRU expert and 41,152 for the TCN expert, for a total of 70,272 shared parameters.

For routing, we adopt a Top-1 strategy, so that each sample is processed only by the expert with the highest gating probability. The final representation is formed by the selected expert output weighted by its routing probability. Training is conducted for 20 communication rounds with 4 local epochs per round and a batch size of 20, using a fixed random seed of 42. The look-back window length is set to 12. Local optimization uses AdamW with weight decay 10^{-4}, and the learning rate decays from 0.01 to 0.005 between rounds 10 and 20.

5.3. Evaluation metrics and statistical testing

Prediction performance is evaluated using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) at two forecasting horizons: 15 min for short-term prediction and 30 min for medium-term prediction.

In addition to the main prediction metrics, we use task-specific diagnostic and statistical analyses where needed. The temporal variation ratio is introduced in the distribution analysis to examine whether the predicted series amplifies short-term fluctuations. The Diebold–Mariano test [49] is reported together with the regression-head ablation to assess whether the paired forecasting-error differences are statistically meaningful. Runtime per batch is used in the routing efficiency analysis to compare sparse Top-1 routing with dense Top-2 routing under the same execution setting.

5.4. Main forecasting performance

Table 2 presents the prediction performance of the proposed framework compared to various baselines. The darker background colors in the table highlight the top-performing results, indicating that the Offline FSR-FL approach consistently achieves the lowest RMSE and MAE across both datasets. While the Online version experiences a slight performance decrease due to strict sequential processing, it remains competitive against standard federated baselines.

When analyzing methods trained only on local data, including TCN, GRU, GLinear, and SFNN, we observe that they generally exhibit higher error rates than federated approaches. Although specific lightweight models such as GLinear or SFNN may show decent performance on certain metrics, they rely on isolated data silos. This isolation prevents them from benefiting from the diverse traffic patterns found in other regions, which restricts their generalization capability. For instance, the local TCN model results in a high RMSE of 32.1069 on the Luzern dataset for the 30-minute horizon, confirming that local data alone is often insufficient for robust modeling and highlighting the necessity of collaborative learning strategies.

In the federated setting, FSR-FL outperforms standard architectures (FedGRU, FedTCN) and the FedAvg baseline, which uses the same MoE backbone but with uniform aggregation. The results show that FSR-FL outperforms all these federated variants. Specifically, compared to FedAvg, which has an RMSE of 3.3018 on the METR-LA dataset, our method reduces the error to 3.1151, representing an improvement of approximately 5.6%. Similarly, on the Luzern dataset, we lower the RMSE from 30.7685 to 28.7168, achieving a gain of about 6.7%. These improvements confirm that our aggregation mechanism, which weights updates based on expert utilization, is more effective than simple parameter averaging.

The comparison with MoE-FedProx, MoE-pFedMe, and FedMix-style MoE further indicates that the performance gain is not only due to the MoE backbone. These methods introduce proximal regularization, personalization, or client-specific expert mixing, but they do not consistently match the proposed method. This suggests that explicitly aligning aggregation with expert utilization is better suited to heterogeneous traffic forecasting than directly applying existing federated or personalized learning mechanisms within the proposed structure.

Centralized-MoE provides a non federated reference for examining the accuracy cost of federation. On METR-LA, Offline FSR-FL achieves

Table 2
Average prediction performance comparison on METR-LA and Luzern datasets. **Darker background colors indicate better performance (lower error)**, highlighting the top three results in each column.

Method	Dataset: METR-LA				Dataset: Luzern			
	15 min (H = 3)		30 min (H = 6)		15 min (H = 3)		30 min (H = 6)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
TCN (Local-Only)	4.4504	3.5837	5.1952	4.3002	32.6995	28.2142	32.1069	26.8492
GRU (Local-Only)	5.2183	4.3389	5.1806	4.2741	39.6392	34.3193	32.0130	26.9358
GLinear (Local-Only)	4.0237	2.0401	4.8514	2.3164	38.9592	16.5555	42.2225	17.8594
SFNN (Local-Only)	3.8422	2.0191	4.6189	2.3565	38.4212	18.5960	40.7747	19.3670
FedGRU	3.9166	2.0815	4.5978	2.3821	39.8026	19.2660	34.2160	16.5453
FedTCN	3.9538	2.1141	4.6538	2.4824	40.2921	18.7822	43.8511	21.8498
FedAvg	2.7856	1.9485	3.3018	2.2016	30.9441	26.7818	30.7685	26.3711
MoE-FedProx	3.4699	2.4148	4.0193	2.8111	34.8348	30.0430	47.1480	42.1716
MoE-pFedMe	4.4733	3.3021	3.7885	2.6514	35.7283	30.7771	35.3382	29.6401
FedMix-style MoE	3.8551	1.9476	4.6018	2.2228	40.0729	35.6236	38.9020	34.1790
Centralized-MoE	2.9908	1.9781	3.3931	2.2240	**27.6411**	23.3648	28.8885	24.1686
Online (**FSR-FL**)	2.8215	2.0049	3.3355	2.2661	39.3606	35.2919	37.2503	32.6777
Offline (**FSR-FL**)	**2.7453**	**1.9051**	**3.1151**	**2.1983**	29.1607	25.0436	**28.7168**	24.3478

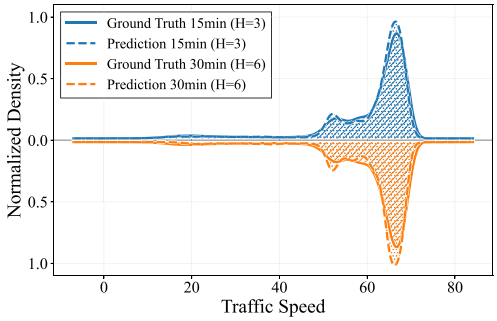


Fig. 3. Ground truth and predicted traffic speed distributions are compared on the METR-LA dataset under offline settings.

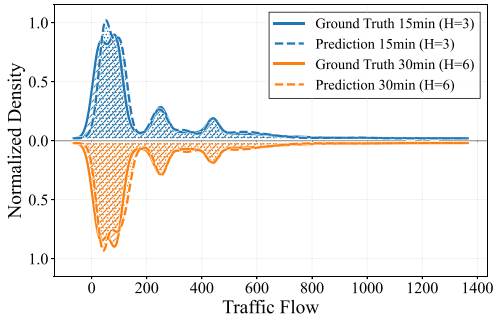


Fig. 4. Ground truth and predicted traffic flow distributions are compared on the Luzern dataset under offline settings.

lower RMSE than Centralized-MoE at both horizons, decreasing the error from 2.9908 to 2.7453 at $H = 3$ and from 3.3931 to 3.1151 at $H = 6$. On Luzern, Centralized-MoE is better at $H = 3$, with RMSE 27.6411 compared with 29.1607 for Offline FSR-FL. However, the two methods are close at $H = 6$, with Offline FSR-FL achieving an RMSE of 28.7168 compared with 28.8885 for Centralized-MoE. These results show that the accuracy cost of federation is limited and dataset dependent, while FSR-FL remains competitive without centralized access to all client data.

5.5. Prediction distribution and temporal variation analysis

The distribution plots in Figs. 3 and 4 provide deeper insight into these results. For the METR-LA dataset (Traffic Speed), the prediction task becomes harder as the time horizon increases. Consequently, the error at 15 min (2.74 RMSE) is lower than at 30 min (3.11 RMSE). In contrast, the Luzern dataset (Traffic Flow) exhibits an anomaly: the 30 min prediction (28.71 RMSE) is more accurate than the 15 min prediction (29.16 RMSE). We attribute this to the nature of traffic flow data. Short intervals, such as 15 min, often exhibit high stochastic noise and fluctuations. A longer 30-min window smooths out this noise, allowing the model to capture underlying traffic volume trends more effectively. As shown in Fig. 4, our model successfully fits the bimodal distribution of the ground truth even with these complex flow dynamics. Beyond prediction accuracy, the results indicate that structured expert-level synchronization is more effective than uniform full-model aggregation when clients exhibit heterogeneous traffic regimes.

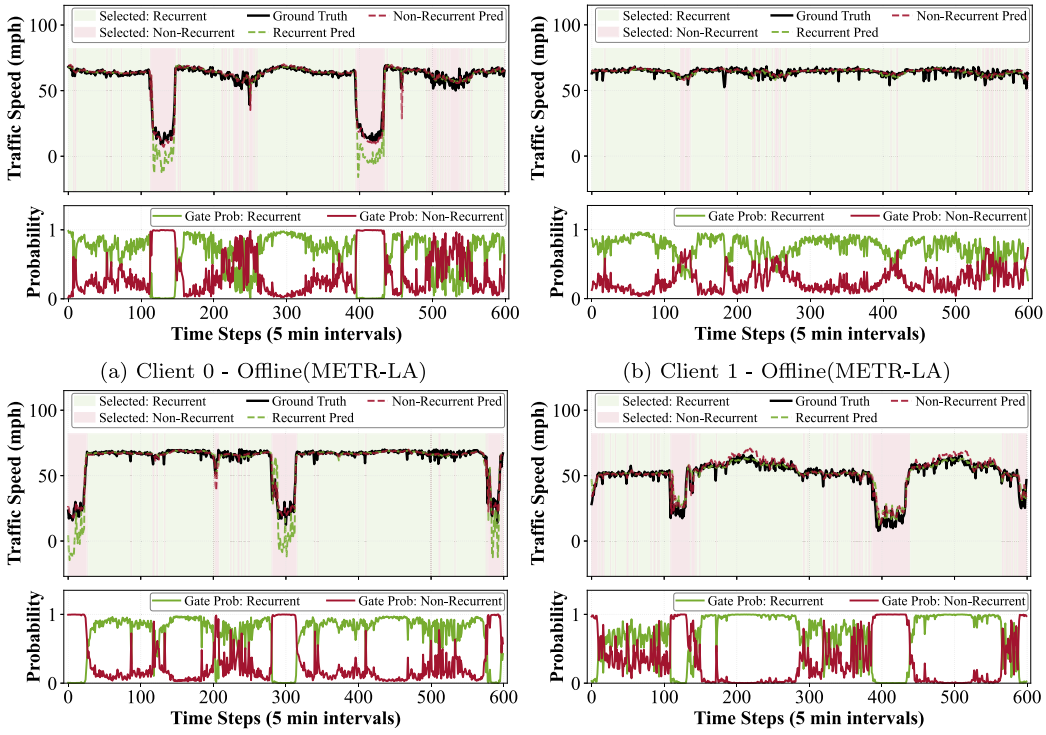
To further analyze whether the Luzern 15 min prediction learns spurious high frequency patterns, we add a temporal variation consistency check after the distribution comparison. In this context, spurious high frequency learning means that the model introduces artificial short term fluctuations that are stronger than those observed in the ground truth. We therefore compute a prediction-to-ground-truth temporal variation ratio as follows:

$$R_{\Delta}^{(H)} = \frac{\mathrm{Std}\left(\hat{y}_t^{(H)} - \hat{y}_{t-1}^{(H)}\right)}{\mathrm{Std}\left(y_t^{(H)} - y_{t-1}^{(H)}\right)}. \tag{34}$$

Here, $y_t^{(H)}$ and $\hat{y}_t^{(H)}$ denote the ground truth and prediction at horizon H, respectively. A value of $R_{\Delta}^{(H)}$ larger than one would indicate that the prediction amplifies short term variations relative to the target sequence. On the Luzern offline setting, we obtain $R_{\Delta}^{(3)} = 0.2527$ for the 15 min horizon and $R_{\Delta}^{(6)} = 0.2727$ for the 30 min horizon. Since both values are well below one, the predictions do not show excessive short term oscillations. This indicates that the slightly lower 30 min RMSE reflects the horizon dependent predictability of the Luzern flow series rather than spurious high frequency learning at the 15 min horizon.

5.6. Expert activation and routing behavior

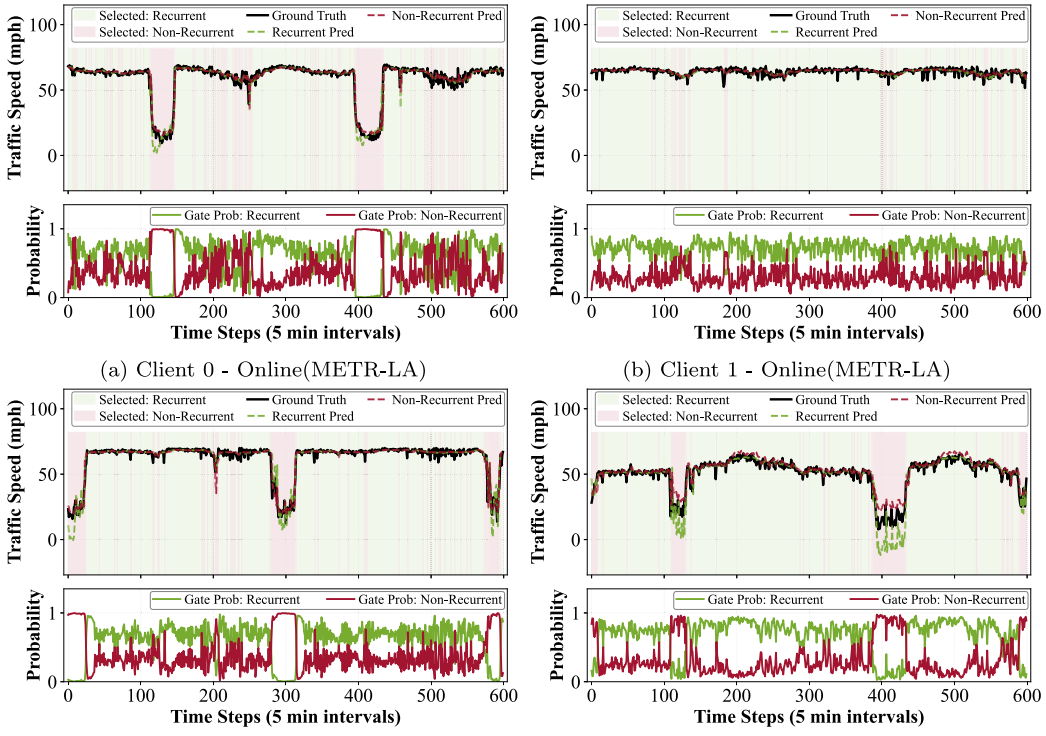
To analyze the behavior of the proposed framework, we visualize prediction patterns and expert activations for representative clients. Each visualization follows a consistent format: the upper panel displays the ground-truth traffic alongside predictions from both the recurrent and non-recurrent experts, while the lower panel shows the corresponding gating probabilities over time. This layout allows us to observe how the model decomposes temporal structures into stable or transient components and whether the routing logic aligns with actual traffic regimes.



(a) Client 0 - Offline(METR-LA) (b) Client 1 - Offline(METR-LA)

(c) Client 2 - Offline(METR-LA) (d) Client 3 - Offline(METR-LA)

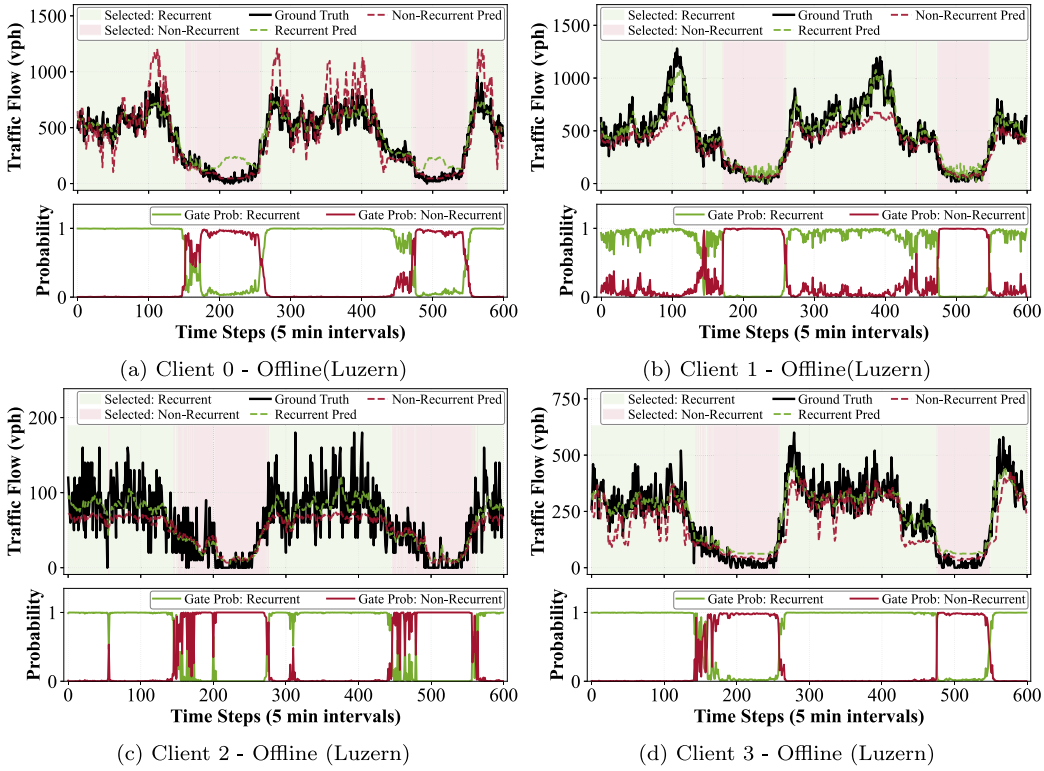
Fig. 5. Comparison of offline forecasting performance and expert gating probabilities for four representative clients in the METR-LA dataset.



(a) Client 0 - Online(METR-LA) (b) Client 1 - Online(METR-LA)

(c) Client 2 - Online(METR-LA) (d) Client 3 - Online(METR-LA)

Fig. 6. Comparison of online forecasting performance and expert gating probabilities for four representative clients in the METR-LA dataset.



(a) Client 0 - Offline(Luzern) (b) Client 1 - Offline(Luzern)

(c) Client 2 - Offline (Luzern) (d) Client 3 - Offline (Luzern)

Fig. 7. Comparison of offline forecasting performance and expert gating probabilities for four representative clients in the Luzern dataset.

Figs. 5 and 6 report the offline and online analyses on four representative clients on METR-LA dataset. As shown in Figs. 5(a)–5(d) and Figs. 6(a)–6(d), the gating mechanism provides a clear interpretation of expert usage by adapting the routing weights to local traffic regimes. During smooth and stable segments, the gate tends to assign more weight to the Recurrent Expert, which captures regular temporal evolution, for example Client 2 stays around 0.6 in the recurrent probability between steps 350 and 550. When irregular patterns appear, such as abrupt speed drops or congestion, the gate shifts to the Non-Recurrent Expert, with the non-recurrent probability rising to near 1.0 for Client 0 around steps 120–150 and 400–440, and for Client 2 near step 300, consistent with the ground-truth declines. For Client 1, the near-balanced routing probabilities reflect a mixed temporal state. The corresponding sequence exhibits frequent local variations, in which recurrent evolution and short-term deviations jointly affect the prediction. The gate therefore assigns comparable probabilities to the two experts because both temporal roles are relevant for this client. Overall, the offline and online views show consistent switching behavior, supporting that the routing is driven by temporal pattern changes rather than arbitrary selection.

Figs. 7 and 8 show the forecasting performance for the Luzern dataset. Unlike the traffic speed data in METR-LA, the traffic flow in Luzern typically follows a standard daily cycle. Consequently, the Recurrent Expert dominates during stable high-flow periods to capture predictable patterns, as evidenced by the sustained high recurrent probability for Client 1 during peaks. In contrast, the model switches to the Non-Recurrent Expert when the flow deviates from normal periodicity, such as during sudden volume drops. For example, when Client 0 experiences a sharp decline in flow between steps 150 and 250, the gate quickly shifts preference to the non-recurrent expert to handle this irregularity. This behavior confirms that the routing strategy effectively distinguishes between routine cycles and unexpected fluctuations in both offline and online settings.

A further comparison between the offline and online visualizations reveals the impact of data presentation on the gating mechanism. The offline model produces sharper probability transitions that align precisely with traffic events, as random shuffling provides a global view of the data distribution. In contrast, the online model processes data sequentially, making it more sensitive to recent observations. This results in a slight adaptation lag, where the gate requires a few extra time steps to switch experts during sudden changes. Despite this delay, the online model successfully identifies major patterns and remains consistent with the offline baseline, confirming its robustness for real-time deployment.

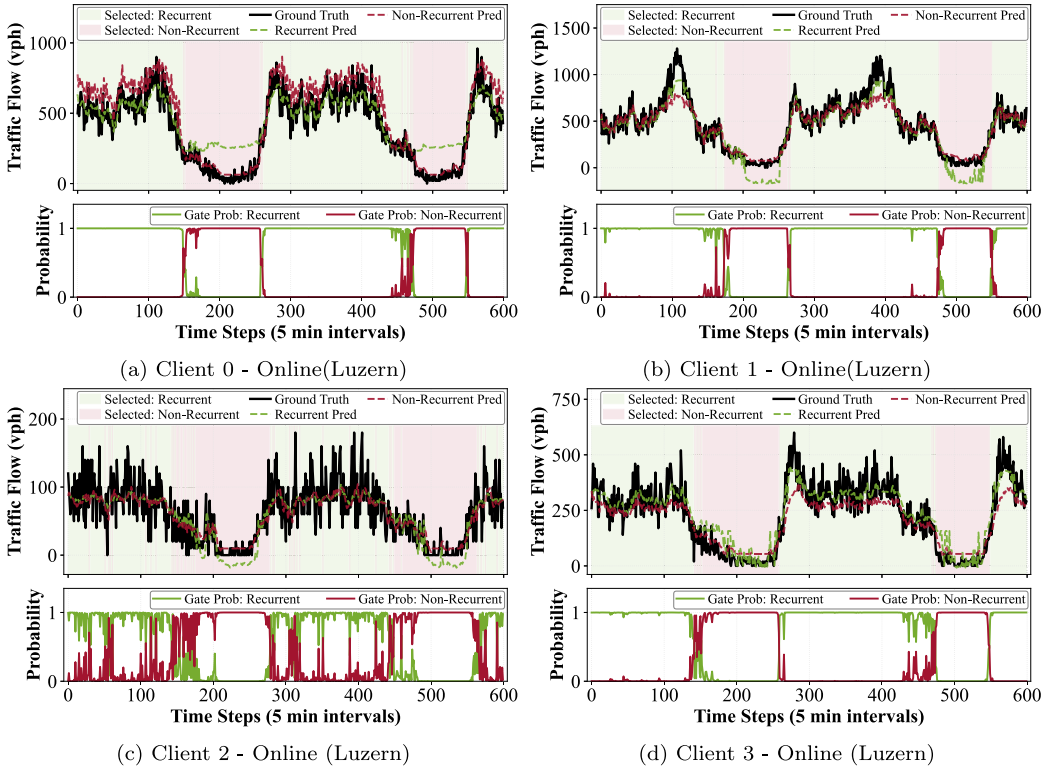
5.7. Ablation study

5.7.1. Effect of the regression head

Table 3 analyzes the impact of the Regression Head (RH). The effectiveness of this module depends heavily on the data distribution shown in previous Figs. 3 and 4. For the METR-LA dataset, which has a bounded speed distribution, the RH module offers only marginal gains in the offline setting, reducing RMSE from 2.7606 to 2.7453. However, it causes a slight performance drop during online adaptation, suggesting that forcing volatility prediction on stationary data with limited samples may introduce noise. In contrast, the RH module is crucial for the Luzern dataset, which features a long-tailed and highly skewed flow distribution. In this case, the module serves as a volatility detector, helping the gate anticipate shifts. Removing the RH in the online setting leads to a significant increase in error, where the 30-minute RMSE rises from 37.2503 to 42.2811. This confirms that explicit physical constraints are essential for adapting to non-stationary and high-variance data.

5.7.2. Significance test for the regression head ablation

Table 3 shows that the effect of the regression head (RH) varies across datasets, training protocols, and forecasting horizons. To assess whether the observed differences are statistically meaningful, we apply the Diebold–Mariano (DM) test to the paired test-window errors of the proposed model and the w/o RH variant. Both models are evaluated on the same chronological test windows.



(a) Client 0 - Online(Luzern) (b) Client 1 - Online(Luzern)

(c) Client 2 - Online (Luzern) (d) Client 3 - Online (Luzern)

Fig. 8. Comparison of online forecasting performance and expert gating probabilities for four representative clients in the Luzern dataset.

Table 3
Ablation experiment of the Regression Head (RH) module in the proposed FSR-FL framework.

Dataset	Settings	Structure	15 min (H = 3)		30 min (H = 6)	
			RMSE	MAE	RMSE	MAE
METR-LA	Offline	w/o RH	2.7606	1.9178	3.1771	2.2220
		Proposed	2.7453	1.9051	3.1151	2.1983
	Online	w/o RH	2.7852	1.9693	3.3175	2.2358
		Proposed	2.8215	2.0049	3.3355	2.2661
Luzern	Offline	w/o RH	26.9475	22.6779	31.3995	26.9015
		Proposed	29.1607	25.0436	28.7168	24.3478
	Online	w/o RH	42.0324	37.8156	42.2811	37.8167
		Proposed	39.3606	35.2919	37.2503	32.6777

For each test window t, the error difference is defined as

$$d_t = e_t^{\mathrm{w/oRH}} - e_t^{\mathrm{Prop}}, \qquad H_0 : \mathbb{E}[d_t] = 0, \quad H_1 : \mathbb{E}[d_t] \neq 0. \tag{35}$$

A positive DM statistic indicates lower error for the proposed model, while a negative statistic indicates lower error for w/o RH. At the 5% significance level, H_0 is rejected when the p-value is below 0.05. Otherwise, the difference is not treated as statistically significant.

We report the DM statistics for MAE and MSE. MSE is used for the RMSE-related comparison because RMSE is obtained after averaging squared errors and applying a square-root transformation, so it does not directly provide a paired test-window error sequence. The DM test uses heteroskedasticity- and autocorrelation-consistent (HAC) variance estimation to account for serial dependence in forecasting errors. For multi-step prediction, the HAC lag is set to $H-1$, which is 2 for $H=3$ and 5 for $H=6$ (see Table 4).

The METR-LA results show that RH significantly reduces MAE in the offline setting at both horizons. For MSE, the reduction is significant at $H=6$, while the $H=3$ result is not significant at the 0.05 level. In the online setting, w/o RH obtains significantly lower MAE at both horizons and significantly lower MSE at $H=3$, while the MSE difference at $H=6$ is not significant.

For Luzern, RH shows a stronger positive effect in most settings. In the offline setting, w/o RH is significantly better at $H=3$, but RH becomes significantly better at $H=6$ for both MAE and MSE. In the online setting, RH significantly reduces both MAE and MSE at $H=3$ and $H=6$. These results confirm that the RH ablation differences are statistically meaningful, but the direction of the effect depends on the dataset, training protocol, and forecasting horizon.

5.7.3. Effect of the expert structure

Table 5 confirms the need for the MoE architecture by comparing it with single-expert baselines. Results show that single-expert models perform poorly across diverse traffic patterns. Specifically, the Recurrent model cannot adapt to rapid changes in the Luzern online scenario, leading to a high 15-minute RMSE of 78.6672. Conversely, the Non-Recurrent model lacks the memory needed for stable predictions. In contrast, the FSR-FL framework consistently outperforms these baselines in all settings. For the METR-LA offline task (15 min), FSR-FL achieves an RMSE of 2.7453, which is significantly better than the Recurrent (4.0686) and Non-Recurrent (3.0251) models. This proves that the gating mechanism effectively balances the temporal memory of the recurrent expert with the pattern recognition of the non-recurrent expert.

5.7.4. Gate input ablation on the explicit difference term

To clarify the role of the explicit difference term in Eq. (9), we further conduct a gate input ablation under both $H=3$ and $H=6$. The original gate input includes the explicit first-order difference channel ΔX, while the ablated variant removes this channel. For Luzern, the exogenous rainfall variable is retained in both variants. The expert inputs, expert architectures, aggregation strategy, and training protocol are kept unchanged.

Table 6 reports the ablation results of the explicit difference term in the gate input. On METR-LA, removing ΔX increases the error in

Table 4
Diebold–Mariano test for the regression head ablation.

Dataset	Setting	Horizon	MAE		MSE	
			DM statistics	p-value	DM statistics	p-value
METR-LA	Offline	H = 3	2.476	0.0133	1.868	0.0618
	Offline	H = 6	2.203	0.0276	3.655	2.57×10^{-4}
	Online	H = 3	−6.999	2.61×10^{-12}	−5.072	3.95×10^{-7}
	Online	H = 6	−3.611	3.06×10^{-4}	−0.101	0.9193
Luzern	Offline	H = 3	−29.394	5.18×10^{-188}	−2.766	0.0057
	Offline	H = 6	29.949	5.02×10^{-195}	13.886	9.58×10^{-44}
	Online	H = 3	22.683	3.17×10^{-113}	23.761	5.51×10^{-124}
	Online	H = 6	18.853	5.87×10^{-79}	8.783	1.65×10^{-18}

Table 5
Ablation experiment of different expert types in the proposed FSR-FL framework.

Dataset	Settings	Expert	15 min (H = 3)		30 min (H = 6)	
			RMSE	MAE	RMSE	MAE
METR-LA	Offline	Recurrent	4.0686	3.0244	3.7159	2.7030
		Non-Recurrent	3.0251	2.1605	3.4299	2.4891
		Proposed (MoE)	2.7453	1.9051	3.1151	2.1983
	Online	Recurrent	2.9664	2.1054	3.3835	2.4812
		Non-Recurrent	3.0378	2.1883	3.4621	2.4864
		Proposed (MoE)	2.8215	2.0049	3.3355	2.2661
Luzern	Offline	Recurrent	41.3290	36.3930	48.5641	44.1535
		Non-Recurrent	44.4565	39.1822	51.1606	46.0540
		Proposed (MoE)	29.1607	25.0436	28.7168	24.3478
	Online	Recurrent	78.6672	74.4238	45.9837	41.3816
		Non-Recurrent	44.5994	40.1537	50.9773	46.0510
		Proposed (MoE)	39.3606	35.2919	37.2503	32.6777

Table 6
Ablation experiment of the explicit difference term in the gate input of the proposed FSR-FL framework.

Dataset	Setting	Gate input	15 min (H = 3)		30 min (H = 6)	
			RMSE	MAE	RMSE	MAE
METR-LA	Offline	w/ ΔX	2.7453	1.9051	3.1334	2.2164
		w/o ΔX	2.7729	1.9501	3.1882	2.2786
	Online	w/ ΔX	2.8215	2.0049	3.2994	2.3793
		w/o ΔX	2.9949	2.1825	3.2728	2.3637
Luzern	Offline	w/ ΔX	29.1607	25.0436	28.0493	23.6231
		w/o ΔX	30.7290	26.5809	28.4610	23.9283
	Online	w/ ΔX	39.3606	35.2919	37.2503	32.6777
		w/o ΔX	32.3296	28.0993	49.9390	45.5596

the offline setting for both horizons. In the online setting, the variant without ΔX performs worse at $H = 3$, but gives slightly lower error at $H = 6$. We therefore further examine the online $H = 6$ case using the Diebold–Mariano test on paired test window errors. The MAE-based loss gives a DM statistic of 2.9499 with $p = 0.0032$, while the squared error loss used as the RMSE basis gives a DM statistic of 1.8675 with $p = 0.0618$ over 41,108 paired windows. This indicates a significant difference for MAE and a marginal difference for the RMSE-based criterion.

On Luzern, the explicit difference term is generally useful, especially at $H = 6$. Removing ΔX increases the online RMSE from 37.2503 to 49.9390 and the online MAE from 32.6777 to 45.5596. These results suggest that ΔX should be interpreted as a local dynamics cue for the gate, rather than as a separate information source beyond X or a fixed expert selection rule. Its contribution becomes clearer when the sequence contains stronger local variation or changes associated with exogenous information.

Table 7
Top-2 routing analysis under sequential CPU execution.

Dataset	H	Routing	Runtime (ms/batch)	RMSE	MAE
METR-LA	3	Top-1	0.8724	3.8874	1.9051
		Top-2	1.3123	3.8618	1.9274
	6	Top-1	0.9155	4.6310	2.2165
		Top-2	1.3209	4.6364	2.2433
Luzern	3	Top-1	0.8870	27.4578	25.0432
		Top-2	1.2575	26.9661	24.6034
	6	Top-1	0.7739	26.8753	23.6279
		Top-2	1.2828	26.6358	23.3840

5.8. Routing efficiency analysis

To further examine the routing choice, we compare the default Top-1 routing with a Top-2 routing variant under a sequential CPU execution setting. The benchmark is conducted on a CPU with a batch size equal to 1, which reduces the influence of CUDA parallelism and batch-level throughput effects. The reported runtime therefore reflects the per-batch execution cost under the same saved local models, while RMSE and MAE are reported on the original data scale. Since the model contains two experts, Top-2 routing acts as a dense two-expert reference rather than another sparse routing configuration (see Table 7).

The results show that Top-2 routing has dataset-dependent effects. On METR-LA, Top-2 does not consistently improve MAE, indicating that the dominant expert selected by Top-1 is already sufficient for this dataset. On Luzern, Top-2 improves the RMSE and MAE, especially at $H = 3$, suggesting that the recurrent and non-recurrent experts can provide complementary information when the traffic sequence is affected by stronger mixed dynamics and exogenous information. Therefore, Top-2 is useful as a dense routing reference, while Top-1 remains the default routing choice because it preserves sparse regime selection and a clearer expert assignment for interpretation.

5.9. Sensitivity analysis

All sensitivity experiments are conducted under the offline setting and reported as mean ± standard deviation over three random seeds, i.e., 42, 2025, and 1723. We adopt a one factor at a time design: unless otherwise specified, the default configuration uses a look back window length of 12, standard normalization, and $\lambda_{\text{vol}} = 0.42$. The results are summarized in Table 8.

For the normalization sensitivity analysis, all scaling statistics are estimated from the training split only and then applied to the validation and test splits to avoid information leakage. We compare four normalization strategies.

Standard normalization centers the input using the training-set mean and rescales it by the training-set standard deviation:

$$x' = \frac{x - \mu_{\text{tr}}}{\sigma_{\text{tr}} + \epsilon}. \tag{36}$$

Table 8
Sensitivity analysis on the METR-LA and Luzern offline settings across different seeds.

Sensitivity factor	Value	Controlled setting	METR-LA		Luzern	
			RMSE	MAE	RMSE	MAE
L length	6	Standard, $\lambda_{\text{vol}} = 0.42$	3.171 ± 0.012	2.242 ± 0.026	30.535 ± 1.535	25.986 ± 1.575
	12	Standard, $\lambda_{\text{vol}} = 0.42$	3.140 ± 0.015	2.212 ± 0.008	27.827 ± 3.691	23.390 ± 3.734
	18	Standard, $\lambda_{\text{vol}} = 0.42$	3.138 ± 0.008	2.242 ± 0.016	27.884 ± 3.666	23.443 ± 3.738
	24	Standard, $\lambda_{\text{vol}} = 0.42$	3.124 ± 0.019	2.212 ± 0.017	28.017 ± 1.195	23.757 ± 1.202
Normalization	Standard	$L = 12$, $\lambda_{\text{vol}} = 0.42$	3.140 ± 0.015	2.212 ± 0.008	27.827 ± 3.691	23.390 ± 3.734
	Robust	$L = 12$, $\lambda_{\text{vol}} = 0.42$	3.127 ± 0.035	2.188 ± 0.033	29.306 ± 0.860	24.842 ± 0.901
	Min–max	$L = 12$, $\lambda_{\text{vol}} = 0.42$	3.236 ± 0.079	2.362 ± 0.099	29.580 ± 2.845	25.160 ± 2.892
	Log	$L = 12$, $\lambda_{\text{vol}} = 0.42$	3.159 ± 0.016	2.218 ± 0.012	29.532 ± 2.138	25.064 ± 2.198
Volatility weight	0.21	$L = 12$, Standard	3.159 ± 0.014	2.223 ± 0.018	27.171 ± 1.334	22.692 ± 1.328
	0.42	$L = 12$, Standard	3.140 ± 0.015	2.212 ± 0.008	27.827 ± 3.691	23.390 ± 3.734
	0.64	$L = 12$, Standard	3.141 ± 0.013	2.229 ± 0.004	28.224 ± 4.878	23.757 ± 4.939
	0.84	$L = 12$, Standard	3.150 ± 0.027	2.242 ± 0.032	29.893 ± 3.653	25.423 ± 3.732

Robust normalization uses the training-set median and interquartile range, which reduces the influence of extreme observations:

$$x' = \frac{x - m_{\text{tr}}}{q_{0.75,\text{tr}} - q_{0.25,\text{tr}} + \epsilon}. \tag{37}$$

Min–max normalization maps the input according to the minimum and maximum values estimated from the training split:

$$x' = \frac{x - x_{\min,\text{tr}}}{x_{\max,\text{tr}} - x_{\min,\text{tr}} + \epsilon}. \tag{38}$$

Log normalization first applies a logarithmic transform to the original non-negative measurements to compress large values, followed by standardization:

$$x' = \frac{\log(1 + x) - \mu_{\log,\text{tr}}}{\sigma_{\log,\text{tr}} + \epsilon}. \tag{39}$$

Here, μ_{tr} and σ_{tr} denote the training-set mean and standard deviation, m_{tr} denotes the training-set median, $q_{0.75,\text{tr}}$ and $q_{0.25,\text{tr}}$ denote the third and first quartiles, and ϵ is a small constant for numerical stability.

For the look back window length, the proposed method remains stable on METR-LA, where RMSE changes only from 3.124 to 3.171 and MAE remains within 2.212 to 2.242. This indicates that the model is not tied to a specific temporal window on METR-LA. On Luzern, shorter history is less effective, as using a look back window of 6 gives the largest RMSE and MAE, 30.535 and 25.986, respectively. In contrast, the settings with window lengths of 12, 18, and 24 give closer results, with RMSE around 27.827 to 28.017 and MAE around 23.390 to 23.757. This suggests that Luzern benefits from a moderate temporal context, while further increasing the window length brings limited additional gain. Overall, $L = 12$ is considered as common and balanced setting across different datasets for short-term prediction

For input normalization, robust normalization gives the lowest average error on METR-LA, with RMSE 3.127 and MAE 2.188, but the improvement over standard normalization is small. In contrast, standard normalization performs best on Luzern, reducing RMSE and MAE to 27.827 and 23.390, whereas robust, minmax, and log normalization yield higher errors. Therefore, standard normalization provides the most consistent choice across the two datasets. This also supports its use as the default setting, since it avoids dataset specific tuning and gives competitive results under both traffic and mobility demand patterns.

For the volatility weight, METR-LA is only mildly affected by changes in λ_{vol}, with RMSE staying between 3.140 and 3.159 and MAE between 2.212 and 2.242. Luzern shows stronger sensitivity: increasing λ_{vol} from 0.21 to 0.84 raises RMSE from 27.171 to 29.893 and MAE from 22.692 to 25.423. This implies that a large volatility loss weight may overemphasize short term fluctuation patterns on Luzern. Overall, $\lambda_{\text{vol}} = 0.42$ is retained as the default because it provides a balanced setting across datasets rather than optimizing for a single dataset only.

6. Discussion

The above results indicate that the proposed FSR-FL framework effectively addresses the challenges of learning from distributed and heterogeneous traffic data. A key finding is the interpretability of its routing mechanism, which reliably assigns experts based on the prevailing traffic regime, as evidenced by both datasets. The framework also demonstrates scalability and flexibility. Its modular structure allows not only for the replacement of individual experts with alternative architectures, such as attention-based or state-space models, but also supports adaptation to other spatiotemporal forecasting domains beyond traffic. Furthermore, the partial parameter sharing protocol is inherently compatible with various federated optimization algorithms and additional privacy-preserving techniques, enabling integration into diverse edge computing systems and edge-side intelligent transportation infrastructures without fundamental redesign.

From a privacy perspective, FSR-FL follows the standard FL setting in which raw traffic measurements and exogenous observations remain on local clients. Its partial-sharing design further restricts synchronization to shared expert parameters rather than the full local model, reducing the exposure of client-specific model information. Nevertheless, expert-wise aggregation still requires summary-level utilization metadata, which may reveal coarse changes in local traffic regimes if tracked over many communication rounds. Therefore, FSR-FL improves data locality and reduces full-model exposure through its system design, but it should not be interpreted as providing a formal differential privacy guarantee. Additional mechanisms such as quantization, clipping, secure aggregation, or differential privacy could further reduce metadata-related risks. However, these mechanisms would introduce extra design choices and may affect communication cost, computation cost, or prediction accuracy. A detailed evaluation of these privacy mechanisms is beyond the scope of this work, and may be considered in future extensions of FSR-FL to further improve privacy protection.

7. Conclusion

This paper presented FSR-FL, a federated sparse-routing framework for distributed traffic prediction under heterogeneous regional dynamics. By combining role-biased experts with expert-wise aggregation, FSR-FL avoids the limitations of full-model aggregation and enables more structured collaboration across clients. Extensive experiments on selected two datasets, including baseline comparison, ablation, routing efficiency, and sensitivity analyses, show that the proposed design improves prediction accuracy while maintaining data locality. In particular, FSR-FL reduces RMSE by up to 5.6% for METR-LA speed prediction and 6.7% Luzern flow prediction over standard federated baselines. Overall, FSR-FL provides a modular collaborative learning scheme for edge-side intelligent transportation systems. Future work will extend FSR-FL to asynchronous settings for delayed client participation, and integrating graph neural networks for enhanced spatial–temporal modeling. Additionally, we plan to incorporate horizon-specific losses and adaptive filtering strategies to further improve model performance.

CRediT authorship contribution statement

Yetong Wang: Writing – original draft, Visualization, Methodology, Data curation. **Wenze Xiong:** Writing – original draft, Validation. **Wanxin Li:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization. **Hao Guo:** Writing – review & editing, Investigation. **Jie Zhang:** Writing – review & editing, Supervision. **Nguyen Van Huynh:** Writing – review & editing, Supervision. **Mark Nejad:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by the Jiangsu Province Science and Technology Youth Talent Promotion Program under the Grant No. JSTJ-2025-144, and the XJTLU Research Development Fund under the Grant No. RDF-22-02-106, the Natural Science Basic Research Program of Shaanxi under the Grant No. 2025JC-YBMS-688, and the Key R&D Programs of Taicang 2024 under the Grant No. TC2024SF10.

Data availability

Data will be made available on request.

References

[1] A.K. Azad, T. Atkison, A. Shah, A review on machine learning in intelligent transportation systems applications, Open Transp. J. 18 (1) (2024) http://dx.doi.org/10.2174/0126671212330496240821114216.

[2] A. Ali, I. Ullah, S.K. Singh, A. Sharafian, W. Jiang, H. I. Sherazi, X. Bai, Energy-efficient resource allocation for urban traffic flow prediction in edge-cloud computing, Int. J. Intell. Syst. 2025 (1) (2025) 1863025, http://dx.doi.org/10.1155/int/1863025.

[3] A. Mystakidis, P. Koukaras, C. Tjortjis, Advances in traffic congestion prediction: An overview of emerging techniques and methods, Smart Cities 8 (1) (2025) 25, http://dx.doi.org/10.3390/smartcities8010025.

[4] A. Alzughaibi, F.K. Karim, J.A. Darwish, Driven traffic flow prediction in smart cities using hunter-prey optimization with hybrid deep learning models, Alex. Eng. J. 107 (2024) 625–633, http://dx.doi.org/10.1016/j.aej.2024.08.083.

[5] X. Lu, C. Chen, R. Gao, Z. Xing, Prediction of high-speed traffic flow around city based on BO-XGBoost model, Symmetry 15 (7) (2023) 1453, http://dx.doi.org/10.3390/sym15071453.

[6] P. Fafoutellis, E.I. Vlahogianni, Unlocking the full potential of deep learning in traffic forecasting through road network representations: a critical review, Data Sci. Transp. 5 (3) (2023) 23, http://dx.doi.org/10.1007/s42421-023-00083-w.

[7] S. Zhang, J. Li, L. Shi, M. Ding, D.C. Nguyen, W. Tan, J. Weng, Z. Han, Federated learning in intelligent transportation systems: Recent applications and open problems, IEEE Trans. Intell. Transp. Syst. 25 (5) (2024) 3259–3285, http://dx.doi.org/10.1109/TITS.2023.3324962.

[8] C. Chai, C. Ren, C. Yin, H. Xu, Q. Meng, J. Teng, G. Gao, A multifeature fusion short-term traffic flow prediction model based on deep learnings, J. Adv. Transp. 2022 (1) (2022) 1702766, http://dx.doi.org/10.1155/2022/1702766.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A.y. Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282.

[10] N. Pavlidis, V. Perifanis, S.F. Yilmaz, F. Wilhelmi, M. Miozzo, P.S. Efraimidis, R.-A. Koutsiamanis, P. Mulinka, P. Dini, Federated learning in mobile networks: A comprehensive case study on traffic forecasting, IEEE Trans. Sustain. Comput. 10 (3) (2025) 576–587, http://dx.doi.org/10.1109/TSUSC.2024.3504242.

[11] R. Al-Huthaifi, T. Li, Z. Al-Huda, W. Huang, Z. Luo, P. Xie, FedGODE: Secure traffic flow prediction based on federated learning and graph ordinary differential equation networks, Knowl.-Based Syst. 299 (2024) 112029, http://dx.doi.org/10.1016/j.knosys.2024.112029.

[12] Y. Zhang, H. Lu, N. Liu, Y. Xu, Q. Li, L. Cui, Personalized federated learning for cross-city traffic prediction, in: K. Larson (Ed.), Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, International Joint Conferences on Artificial Intelligence Organization, 2024, pp. 5526–5534, http://dx.doi.org/10.24963/ijcai.2024/611, Main Track.

[13] Z. Ke, H. Duan, S. Qian, Interpretable mixture of experts for time series prediction under recurrent and non-recurrent conditions, 2024, arXiv:2409.03282.

[14] S. Li, Y. Cui, Y. Zhao, W. Yang, R. Zhang, X. Zhou, ST-MoE: Spatio-temporal mixture-of-experts for debiasing in traffic prediction, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM'23, Association for Computing Machinery, 2023, pp. 1208–1217, http://dx.doi.org/10.1145/3583780.3615068.

[15] W. Jiang, J. Han, H. Liu, T. Tao, N. Tan, H. Xiong, Interpretable cascading mixture-of-experts for urban traffic congestion prediction, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'24, Association for Computing Machinery, 2024, pp. 5206–5217, http://dx.doi.org/10.1145/3637528.3671507.

[16] Y. Hu, S. Li, D. Xia, Z. Zhou, W. Zhang, H. Li, X. Zhang, S. Wang, Mixture of semantic and spatial experts for explainable traffic prediction, in: Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM'25, Association for Computing Machinery, 2025, pp. 908–917, http://dx.doi.org/10.1145/3746252.3761412.

[17] M.K. Jabbar, H. Jianjun, A. Jabbar, Z. Ur Rehman, Privacy-sensitive federated learning for cross-domain adaptation: The mamba-MoE approach, Results Eng. 27 (2025) 106432, http://dx.doi.org/10.1016/j.rineng.2025.106432.

[18] J. Wang, D. Xu, L. Xing, T. Chen, Y. Liu, M. Shahidehpour, T. Yang, Personalized federated learning with mixture of experts and conformal prediction for household energy forecasting, Expert Syst. Appl. 300 (2026) 130417, http://dx.doi.org/10.1016/j.eswa.2025.130417.

[19] Q. Liu, S. Sun, M. Liu, Y. Wang, B. Gao, Online spatio-temporal correlation-based federated learning for traffic flow forecasting, IEEE Trans. Intell. Transp. Syst. 25 (10) (2024) 13027–13039, http://dx.doi.org/10.1109/TITS.2024.3429533.

[20] L. Yang, W. Chen, X. He, S. Wei, Y. Xu, Z. Zhou, Y. Tong, FedGTP: Exploiting inter-client spatial dependency in federated graph-based traffic prediction, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'24, Association for Computing Machinery, 2024, pp. 6105–6116, http://dx.doi.org/10.1145/3637528.3671613.

[21] S.V. Kumar, L. Vanajakshi, Short-term traffic flow prediction using seasonal ARIMA model with limited input data, Eur. Transp. Res. Rev. 7 (3) (2015) 21, http://dx.doi.org/10.1007/s12544-015-0170-8.

[22] A. Emami, M. Sarvi, S. Asadi Bagloee, Using Kalman filter algorithm for short-term traffic flow prediction in a connected vehicle environment, J. Mod. Transp. 27 (3) (2019) 222–232, http://dx.doi.org/10.1007/s40534-019-0193-2.

[23] S.V. Kumar, K.C. Dogiparthi, L. Vanajakshi, S.C. Subramanian, Integration of exponential smoothing with state space formulation for bus travel time and arrival time prediction, Transport 32 (4) (2017) 358–367, http://dx.doi.org/10.3846/16484142.2015.1100676.

[24] F.G. Habtemichael, M. Cetin, Short-term traffic flow rate forecasting based on identifying similar traffic patterns, Transp. Res. C 66 (2016) 61–78, http://dx.doi.org/10.1016/j.trc.2015.08.017, Advanced Network Traffic Management: From dynamic state estimation to traffic control.

[25] G. Lin, A. Lin, D. Gu, Using support vector regression and K-nearest neighbors for short-term traffic flow prediction based on maximal information coefficient, Inform. Sci. 608 (2022) 517–531, http://dx.doi.org/10.1016/j.ins.2022.06.090.

[26] R. Dey, F.M. Salem, Gate-variants of gated recurrent unit (GRU) neural networks, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS, 2017, pp. 1597–1600, http://dx.doi.org/10.1109/MWSCAS.2017.8053243.

[27] W. Zhao, Y. Gao, T. Ji, X. Wan, F. Ye, G. Bai, Deep temporal convolutional networks for short-term traffic flow forecasting, IEEE Access 7 (2019) 114496–114507, http://dx.doi.org/10.1109/ACCESS.2019.2935504.

[28] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, ICLR'18, 2018.

[29] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph WaveNet for deep spatial-temporal graph modeling, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 1907–1913.

[30] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[31] P. Fafoutellis, E.I. Vlahogianni, Unlocking the full potential of deep learning in traffic forecasting through road network representations: A critical review, Data Sci. Transp. 5 (3) (2023) 23, http://dx.doi.org/10.1007/s42421-023-00083-w.

[32] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, Proc. AAAI Conf. Artif. Intell. 35 (12) (2021) 11106–11115.

[33] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? Proc. AAAI Conf. Artif. Intell. 37 (9) (2023) 11121–11128, http://dx.doi.org/10.1609/aaai.v37i9.26317.

[34] S.T. Hussain Rizvi, N. Kanwal, M. Naeem, Bridging simplicity and sophistication using GLinear: A novel architecture for enhanced time series prediction, Digit. Signal Process. 170 (2026) 105702, http://dx.doi.org/10.1016/j.dsp.2025.105702.

[35] F.-K. Sun, Y.-C. Wu, D.S. Boning, Simple feedfoward neural networks are almost all you need for time series forecasting, 2025, arXiv:2503.23621.

[36] Y. Liu, S. Zhang, C. Zhang, J.J. Yu, FedGRU: Privacy-preserving traffic flow prediction via federated learning, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC, 2020, pp. 1–6, http://dx.doi.org/10.1109/ITSC45102.2020.9294453.

[37] T. Liu, Y. Wang, H. Zhou, J. Luo, F. Deng, Distributed short-term traffic flow prediction based on integrating federated learning and TCN, IEEE Access 12 (2024) 148026–148036, http://dx.doi.org/10.1109/ACCESS.2024.3474300.

[38] C. Meese, H. Chen, W. Li, D. Lee, H. Guo, C.-C. Shen, M. Nejad, Adaptive traffic prediction at the ITS edge with online models and blockchain-based federated learning, IEEE Trans. Intell. Transp. Syst. 25 (9) (2024) 10725–10740, http://dx.doi.org/10.1109/TITS.2024.3391053.

[39] H. Guo, C. Meese, W. Li, C.-C. Shen, M. Nejad, B2SFL: A bi-level blockchained architecture for secure federated learning-based traffic prediction, IEEE Trans. Serv. Comput. 16 (6) (2023) 4360–4374, http://dx.doi.org/10.1109/TSC.2023.3318990.

[40] X. Ma, J. Zhang, S. Guo, W. Xu, Layer-wised model aggregation for personalized federated learning, 2022, arXiv:2205.03993.

[41] M. Reisser, C. Louizos, E. Gavves, M. Welling, Federated mixture of experts, 2021, arXiv:2107.06724.

[42] H. Mei, D. Cai, A. Zhou, S. Wang, M. Xu, FedMoE: Personalized federated learning via heterogeneous mixture of experts, 2024, arXiv:2408.11304.

[43] Y. Li, X. Xu, G. Huang, M. Yao, L. Sun, J. Xu, VSFL: Trajectory prediction framework based on validity-aware semi-asynchronous federated learning in internet of vehicles, Comput. Commun. 224 (2024) 106–117, http://dx.doi.org/10.1016/j.comcom.2024.06.003.

[44] G. Fittipaldi, R.S. Couto, L.H. Costa, Exploring traffic pattern variability in vehicular federated learning, Comput. Commun. 242 (2025) 108279, http://dx.doi.org/10.1016/j.comcom.2025.108279.

[45] Z. Shao, Z. Zhang, F. Wang, Y. Xu, Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'22, Association for Computing Machinery, 2022, pp. 1567–1577, http://dx.doi.org/10.1145/3534678.3539396.

[46] X. Lin, Q. Lu, L. Chen, J.C.P. Cheng, J. Yan, J. Hong, P. Zhao, IUTF dataset: Enabling cross-border resource for analysing the impact of rainfall on urban transportation, Sci. Data (2025) http://dx.doi.org/10.1038/s41597-025-06336-3.

[47] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in: I. Dhillon, D. Papailiopoulos, V. Sze (Eds.), Proceedings of Machine Learning and Systems, vol. 2, 2020, pp. 429–450.

[48] C.T. Dinh, N.H. Tran, T.D. Nguyen, Personalized federated learning with moreau envelopes, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[49] F.X. Diebold, R.S. Mariano, Comparing predictive accuracy, J. Bus. Econom. Statist. 13 (3) (1995) 253–263, http://dx.doi.org/10.2307/1392185.