FLOW: Federated Large Model for Wireless Traffic Prediction

Ruining Wang[†], Chuanting Zhang[*], Haixia Zhang[*], Liang Zhang[§], Basem Shihada[§], and Jingping Qiao[‡]

[†]School of Software, Shandong University, Jinan, China
[*]Institute of Intelligent Communication Technologies, Shandong University, Jinan, China
[§]CEMSE Division, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[‡]School of Information Science and Engineering, Shandong Normal University, Jinan, China

***Abstract*—The rapid growth of wireless traffic makes accurate forecasting vital for network management. Centralized learning faces high communication costs, while federated learning struggles with spatio-temporal dependencies and non-IID data. This paper proposes FLOW, a federated large language model framework for wireless traffic prediction. FLOW incorporates a LoRA-based local predictor with lightweight alignment and mapping layers, preserving strong modeling ability under edge resource constraints. Experiments on real-world Milano and Trento datasets show significant gains, reducing MSE by 11% and MAE by 9.5% compared to baselines. Interestingly, smaller models such as BERT and GPT-2 outperform larger ones in most cases, benefiting from LoRA's efficient fine-tuning on limited data. Moreover, FLOW demonstrates strong few-shot learning, surpassing baselines with only 5% of training data, and exhibits robust zero-shot cross-domain generalization. Ablation studies confirm the critical role of federated aggregation and LoRA fine-tuning, highlighting the effectiveness of the proposed framework.**

***Index Terms*—Wireless traffic prediction, Federated learning, Large language models, LoRA fine-tuning, Edge computing**

I. Introduction

Accurate wireless traffic prediction [1] has become a cornerstone of intelligent network management in the 5G era and beyond. By anticipating future traffic demand, operators can optimize spectrum allocation, balance network loads, and proactively configure resources to enhance service quality. More importantly, precise prediction provides the foundation for energy-aware control strategies, such as adaptive base station sleeping and carrier on–off switching, which are vital to achieving green and sustainable mobile networks [2], [3]. With the emergence of 6G, where ultra-dense deployments, diverse services, and stringent energy-efficiency requirements will dominate, wireless traffic forecasting is expected to play an even greater role in supporting autonomous network optimization and carbon-neutral operations.

To meet these demands, a large body of research has explored methods for traffic prediction [4], [5]. Early studies applied statistical models such as ARIMA and stable processes to capture temporal trends [6], while later works adopted deep learning frameworks including CNNs, RNNs, and graph-based models to better extract spatio-temporal correlations [7]. More recently, federated learning (FL) has been introduced as a distributed solution to address data privacy and communication bottlenecks by enabling collaborative training across base stations without centralizing raw traffic data [8]–[11]. Within this paradigm, advances such as gradient-similarity aggregation, multi-time scale modeling, and communication-efficient compression schemes have shown promising results in reducing the impact of data heterogeneity and communication overhead.

Despite this progress, several challenges remain. First, statistical heterogeneity among base stations, caused by differences in user behaviors and service distributions, often leads to unstable convergence and degraded accuracy in federated settings. Second, many existing FL-based models rely on relatively shallow architectures, which struggle to capture the long-term nonlinear dependencies that characterize real-world wireless traffic. Third, while large Transformer-based models have demonstrated strong predictive power, their high computational and memory demands make direct deployment on resource-constrained edge devices impractical [12]. These limitations hinder the development of a unified solution that is both accurate and deployable at scale.

To overcome these challenges, this paper proposes FLOW, a Federated Large mOdel framework for Wireless traffic prediction. The key innovation of FLOW lies in its integration of large language models (LLMs) into the federated learning paradigm. By leveraging the long-sequence modeling and generalization capabilities of LLMs, FLOW effectively addresses data heterogeneity and mitigates cold-start issues at new base stations. At the same time, FLOW adopts a LoRA-based fine-tuning strategy that substantially reduces the number of trainable parameters, thereby lowering the computational burden on edge nodes and improving communication efficiency during federated updates. Through this design, FLOW not only delivers high prediction accuracy but also achieves a lightweight and scalable implementation suitable for practical deployment in next-generation mobile networks. To summarize, the main contributions of this paper are as follows:

- We propose FLOW, a federated large model framework, for predicting wireless traffic. FLOW is the first integration of large language models with federated learning for wireless traffic forecasting, addressing privacy protection, traffic heterogeneity, and complex temporal dependencies.

- We design a lightweight, edge-deployable predictor that combines LLM generalization with minimal local updates, thereby greatly reducing training resources.
- Comprehensive experiments on real-world Milano and Trento datasets show superior performance, with average error reductions of 11% (MSE) and 9.5% (MAE), and strong few-shot and zero-shot generalization.

The rest of this paper is organized as follows. Section II outlines our problem formulation and the design of the FLOW framework. Section III presents experimental results and analysis. Section IV concludes the paper.

II. Problem Formulation and FLOW Framework

A. Problem Definition

Assume there are K base stations (clients), each base station k has its own wireless traffic data, denoted as $\mathcal{D}_k = \{x_1^k, x_2^k, ..., x_T^k\}$, where T represents the total number of historical time steps, and x_t^k represents the wireless traffic value of base station k at time step t. The goal of the wireless traffic forecasting problem is to predict the wireless traffic for the next F time steps using the traffic from the past L_{hist} historical time steps.

For machine learning-based wireless traffic forecasting techniques, we use a sliding window scheme to construct multiple training samples from historical traffic $\mathcal{D}_k$. Specifically, for base station k, we can construct a training sample set $\{(\mathbf{X}_i^k, \mathbf{Y}_i^k)\}_{i=1}^{n_k}$, where $\mathbf{X}_i^k = \{x_i^k, x_{i+1}^k, ..., x_{i+L_{hist}-1}^k\} \in \mathbb{R}^{L_{hist}}$ represents the input sequence of length L_{hist}, $\mathbf{Y}_i^k = \{x_{i+L_{hist}}^k, x_{i+L_{hist}+1}^k, ..., x_{i+L_{hist}+F-1}^k\} \in \mathbb{R}^F$ represents the corresponding prediction target sequence of length F, and n_k represents the number of training samples for base station k.

Thus, the wireless traffic forecasting problem is transformed into a standard time series prediction problem, which can be formally expressed as:

$$\hat{\mathbf{Y}}^k = f(\mathbf{X}^k; \mathbf{W}) \tag{1}$$

where $f(\cdot)$ represents the chosen prediction model, $\mathbf{W}$ represents the model parameters, and $\hat{\mathbf{Y}}^k$ represents the predicted future traffic sequence. In our work, we choose a large language model-based prediction model as $f(\cdot)$. Our goal is to minimize the prediction error across all K base stations, so the model parameters $\mathbf{W}$ are obtained by solving the following optimization problem:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \left(\frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}\big(f(\mathbf{X}_i^k; \mathbf{W}),\, \mathbf{Y}_i^k\big) \right) \tag{2}$$

where $\mathcal{L}(\cdot, \cdot)$ is the loss function, typically chosen as Mean Squared Error (MSE, $\|\cdot\|_2^2$) or Mean Absolute Error (MAE, $\|\cdot\|_1$), and $\mathbf{Y}_i^k$ represents the true traffic values for the i-th sample of the k-th base station.

B. Model Architecture

Applying pre-trained LLMs to edge computing requires lightweight and efficient models. Existing approaches like TimeLLM assume LLMs inherently understand time series, using complex reprogramming layers to map data into natural language space while freezing the LLM. This design has two drawbacks: it relies on massive LLMs, which are unsuited for low-resource settings, and it wastes expressive power by forcing temporal data into a language-based space.

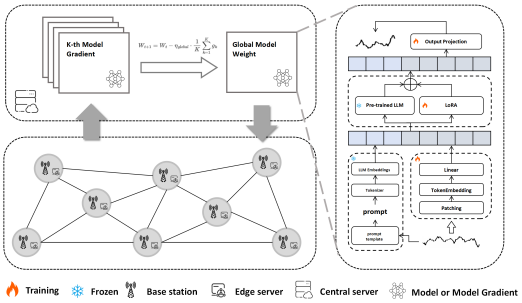

Fig. 1. Model Architecture

We instead transfer LLMs' sequence modeling ability directly to time series prediction via parameter-efficient fine-tuning. Rather than making LLMs understand language-encoded time series, we guide them to capture temporal dynamics without strict size requirements. To this end, we design a lightweight local predictor with three modules: (1) Modal Alignment Layer, (2) LoRA Fine-tuned LLM Layer, and (3) Output Mapping Layer.

Modal Alignment Layer. This layer preprocesses raw time series $\mathbf{X}$ and converts it into embedding sequences $\mathbf{E}_{ts}$ for the LLM. The process includes normalization, sliding-window patching, and convolutional embedding.

First, the input $\mathbf{X} \in \mathbb{R}^{B \times L_{hist}}$ is normalized with mean μ and standard deviation σ to remove dimensional differences and stabilize training. Next, sliding windows of length L_{patch} and step s segment $\mathbf{X}$ into overlapping patches. When the final patch exceeds sequence length, boundary replication is used. This produces $\mathbf{X}_{patch} \in \mathbb{R}^{B \times P_n \times L_{patch}}$.

To capture local temporal dependencies, we treat $\mathbf{X}_{patch}$ as a multi-channel signal and apply 1D convolution:

$$\mathbf{H} = \text{Conv1d}(\mathbf{X}_{patch}). \tag{3}$$

The convolution kernel fuses information across adjacent patches, and multiple kernels (d_{model}) extract diverse local patterns. Finally, a linear projection maps $\mathbf{H} \in \mathbb{R}^{B \times P_n \times d_{model}}$ into the LLM hidden dimension d_{llm}, yielding $\mathbf{E}_{ts} \in \mathbb{R}^{B \times P_n \times d_{llm}}$.

LoRA Fine-tuned Large Model Layer. After converting time series data into embeddings $\mathbf{E}_{ts}$, this layer leverages LLMs' sequence modeling power to capture complex temporal dependencies. To align with prediction tasks, we design structured prompts P composed of three parts: 1) **Dataset Description** (P_{data}): background such as base station location and time range; 2) **Task Description** (P_{task}): the specific prediction objective; 3) **Input Statistical Features** (P_{stats}): summaries of X (e.g., max/min, median, trends, periodicity). These components are concatenated into a structured template with special markers to form the final prompt.

After generating text prompt P, we first use the LLM's accompanying tokenizer to convert it into integer ID sequence P_{ids}. Subsequently, through the LLM's built-in word embedding matrix $\mathbf{W}_{embed}$, we map the ID sequence to embedding vectors $\mathbf{E}_{prompt}$:

$$\mathbf{E}_{prompt} = \mathbf{W}_{embed}(\text{Tokenizer(P)}) \tag{4}$$

Next, we concatenate the prompt embedding $\mathbf{E}_{prompt}$ with the temporal embedding $\mathbf{E}_{ts}$ generated in the previous stage along the sequence dimension to construct the final input to the LLM:

$$\mathbf{E}_{input} = \text{concat}(\mathbf{E}_{prompt}, \mathbf{E}_{ts}) \tag{5}$$

We feed this input into the large language model. Unlike the commonly adopted approach of freezing large model layers in recent years, we use LoRA technology to efficiently fine-tune the model. This enables the model to adapt parameters toward wireless traffic forecasting tasks while maintaining pre-trained knowledge, thereby learning temporal patterns specific to this domain. The final output of this layer is the hidden state sequence $\mathbf{H}_{LLM}$ processed by the LLM:

$$\mathbf{H}_{LLM} = \text{LLM}_{LoRA}(\mathbf{E}_{input}) \tag{6}$$

where $\mathbf{H}_{LLM} \in \mathbb{R}^{B\times(L_{prompt}+P_n)\times d_{llm}}$, and L_{prompt} is the length of the prompt sequence.

Output Mapping Layer. As the final stage, this layer decodes the LLM's hidden states into numerical predictions for the next F steps.

The LLM output $\mathbf{H}_{LLM} \in \mathbb{R}^{B\times(L_{prompt}+P_n)\times d_{llm}}$ contains both prompt and temporal representations. Since only temporal features are relevant, we extract the last P_n hidden states, yielding $\mathbf{H}ts \in \mathbb{R}^{B\times P_n\times d_{llm}}$.

Each sample's feature matrix ($P_n \times d_{llm}$) is then flattened into a vector:

$$h_{flat} = \text{Flatten}(H_{ts}) \tag{7}$$

which aggregates local dynamics and contextual information into a unified representation.

Finally, h_{flat} is passed through a fully connected layer to map features to prediction length F, followed by denormalization with mean μ and standard deviation σ to restore values to the original scale:

$$\hat{\mathbf{Y}} = \sigma \cdot \text{Linear}(h_{flat}) + \mu. \tag{8}$$

C. Training Process

Under the federated learning framework, FLOW adopts a lightweight training strategy with three main considerations: (1) pre-trained LLMs already provide strong transfer capabilities, allowing meaningful optimization with minimal local updates; (2) unlike traditional full-epoch training, lightweight updates greatly reduce edge node computation and improve feasibility in resource-limited settings; (3) gradients, compared to full parameters, have smaller ranges and thus better numerical stability during transmission and floating-point operations.

In each communication round t, the server distributes global parameters $\mathbf{W}_t$ to selected clients. Client k initializes its local model $\mathbf{W}_{t,k} \leftarrow \mathbf{W}_t$ and trains on local data $\mathcal{D}_k$ for a small number of steps S (not a full epoch), yielding $\mathbf{W}'_{t,k}$. The equivalent average gradient is then computed as:

$$\mathbf{g}_k = \frac{\mathbf{W}_t - \mathbf{W}'_{t,k}}{\eta_{global}} \tag{9}$$

Clients upload $\mathbf{g}_k$, and the server updates global parameters using:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_{global} \cdot \frac{1}{K}\sum_{k=1}^{K} \mathbf{g}_k \tag{10}$$

This procedure repeats until convergence or a preset number of rounds.

III. Experiments

A. Experiment Setup

Datasets. This paper uses real wireless network traffic datasets provided by the Italian Telecom Big Data Challenge [13] for evaluation, covering two typical regions in Italy: Milan (Milano) and Trento Province (Trento). The Milan region is divided into 10,000 grid cells, and the Trento region is divided into 6,575 grid cells. Each grid cell corresponds to a base station's service area, recording three types of wireless communication activities: Short Message Service (SMS), voice calls (Call), and internet services (Net). Original data is collected at 10-minute intervals, spanning from November 1, 2013, to January 1, 2014, for a total of 61 days. To mitigate data sparsity issues, we resample the data to hourly intervals.

Experimental Configuration. To ensure statistical significance and generalizability of experimental results, we randomly select 50 base stations from each dataset as participating nodes in federated learning. Data is divided chronologically, with the last 7 days as the test set and the remaining data for training. We set historical sequence length $L_{hist} = 96$ (4 days) and prediction sequence length $F = 24$ (1 day), which aligns with actual needs in wireless network operations.

For different dataset characteristics, we adopt differentiated training strategies: Milano dataset executes 10 rounds of federated communication with 5 batches of local training per round; Trento dataset executes 20 rounds of federated communication with 10 batches of local training per round. Local training batch size is set to 32, LoRA rank $r = 16$, and scaling factor $\alpha = 32$. All baseline methods are independently trained for 20 epochs under the same distributed settings to ensure fair comparison. We adopt MSE and MAE as evaluation metrics.

B. Baseline Methods

To comprehensively evaluate the performance of FLOW, we select eight representative baselines covering traditional statistical methods, machine learning methods, and deep learning methods:

- **ARIMA** [14]: Classic autoregressive integrated moving average model that captures linear trends and periodic patterns in time series through differencing, autoregressive, and moving average components.

TABLE I
OVERALL PERFORMANCE COMPARISON ON MILANO AND TRENTO DATASETS. **BOLD**: BEST PERFORMANCE, UNDERLINE: SECOND BEST

Model	Milano						Trento					
	MSE			MAE			MSE			MAE		
	Call	Net	SMS	Call	Net	SMS	Call	Net	SMS	Call	Net	SMS
ARIMA	1.119	1.296	2.700	0.770	0.830	1.107	8.353	9.002	16.133	1.700	1.909	2.331
Lasso	0.482	0.602	3.315	0.472	0.526	1.035	5.317	9.052	19.052	0.977	1.377	2.146
SVR	0.563	0.606	1.448	0.517	0.547	0.802	3.376	4.980	7.166	0.971	1.280	1.561
LSTM	0.418	0.610	0.978	0.468	0.608	0.646	3.917	10.929	8.521	0.883	1.842	1.561
TFT	0.279	0.491	**0.834**	0.332	0.515	**0.535**	3.494	10.661	8.001	0.773	1.820	1.427
DLinear	0.360	0.480	0.851	0.474	0.545	0.609	1.712	5.953	4.935	0.665	1.355	1.199
TimeLLM	0.203	0.300	0.893	0.318	0.392	0.608	1.579	3.337	4.495	0.595	1.019	1.152
FLOW (DS-1.5B)	0.177	0.268	0.935	0.311	0.370	0.634	1.451	3.307	4.149	0.538	0.998	1.049
FLOW (GPT-2)	0.192	0.254	0.840	0.318	0.355	0.577	**1.392**	2.962	4.108	0.536	0.929	1.033
FLOW (BERT)	**0.157**	**0.238**	0.845	**0.281**	**0.343**	0.575	1.399	**2.902**	**4.083**	**0.532**	**0.925**	**1.031**

- **Lasso** [15]: L1-regularized linear regression method with automatic feature selection capability, suitable for high-dimensional sparse time series modeling.
- **SVR** [16]: Support Vector Regression that uses kernel tricks to model nonlinear temporal dependencies in high-dimensional spaces.
- **LSTM** [17]: Long Short-Term Memory network specifically designed to handle long-term dependencies in sequential data.
- **TFT** [18]: Temporal Fusion Transformer that combines gating mechanisms and multi-head attention to dynamically select relevant features and model complex temporal dependencies.
- **DLinear** [19]: Lightweight linear model that separately models trend and seasonal components, achieving excellent performance while maintaining simplicity.
- **TimeLLM** [20]: Large language model-based time series prediction method that leverages rich knowledge of pre-trained language models for prediction by reprogramming time series features into natural language prompts.

All baseline methods are trained in the same distributed environment, with each client independently optimizing model parameters to simulate data isolation in real scenarios.

C. Overall Performance Comparison

Table I shows the performance comparison results of FLOW with all baseline methods on two datasets. We use three different pre-trained language models as the backbone models for FLOW's local predictor: BERT [21], GPT-2, and DeepSeek-1.5B [22].

Experimental results show that FLOW significantly outperforms all baseline methods in the vast majority of evaluation scenarios. On the Milano dataset, the best configuration (BERT) achieves an average MSE reduction of 11.1% and MAE reduction of 9.0% compared to the strongest baseline TimeLLM. On the Trento dataset, the corresponding improvements are 10.9% and 10.1%, respectively. This performance improvement validates the effectiveness of our design in the wireless traffic forecasting domain.

For our method, smaller parameter models (BERT, GPT-2) outperform large-scale models (DeepSeek-1.5B) in most tasks. This phenomenon can be attributed to the characteristics of LoRA fine-tuning: the parameter space of smaller models is easier to optimize effectively on limited data, thereby avoiding overfitting problems. Meanwhile, FLOW demonstrates consistent superiority across three different types of wireless traffic (Call, Net, SMS), proving the universality and robustness of the method. Considering the computational resource constraints of edge nodes, the excellent performance of small-parameter models provides significant practical value for actual deployment.

TABLE II
FEW-SHOT LEARNING PERFORMANCE

Method	Full Data		5% Data		10% Data	
	MSE	MAE	MSE	MAE	MSE	MAE
TFT	7.385	1.340	10.460	1.871	10.030	1.683
DLinear	4.200	1.073	6.320	1.616	5.323	1.370
FLOW (Qwen-0.6B)	3.058	0.895	3.928	1.142	3.855	1.073
FLOW (GPT-2)	2.821	0.833	3.775	1.025	3.634	0.968
FLOW (BERT)	**2.795**	**0.829**	**3.363**	**0.979**	**3.556**	**0.962**

D. Few-shot Learning Capability Evaluation

To verify FLOW's adaptation capability in data-scarce scenarios, we designed few-shot learning experiments. Table II shows performance comparison results using only 5% and 10% of training data, respectively. Experimental results show that even under extremely data-scarce conditions, FLOW can still achieve or even exceed the performance of baseline methods on complete data. For example, FLOW (BERT) using only 5% data achieves an average MSE of 3.36 on the Trento dataset, significantly better than DLinear's average MSE (4.20) on complete data. This result fully demonstrates the effective transfer of large language models' strong few-shot learning capabilities in wireless traffic forecasting tasks.

Notably, from 5% to 10% data, FLOW's average performance shows stable improvement trends, proving the model's

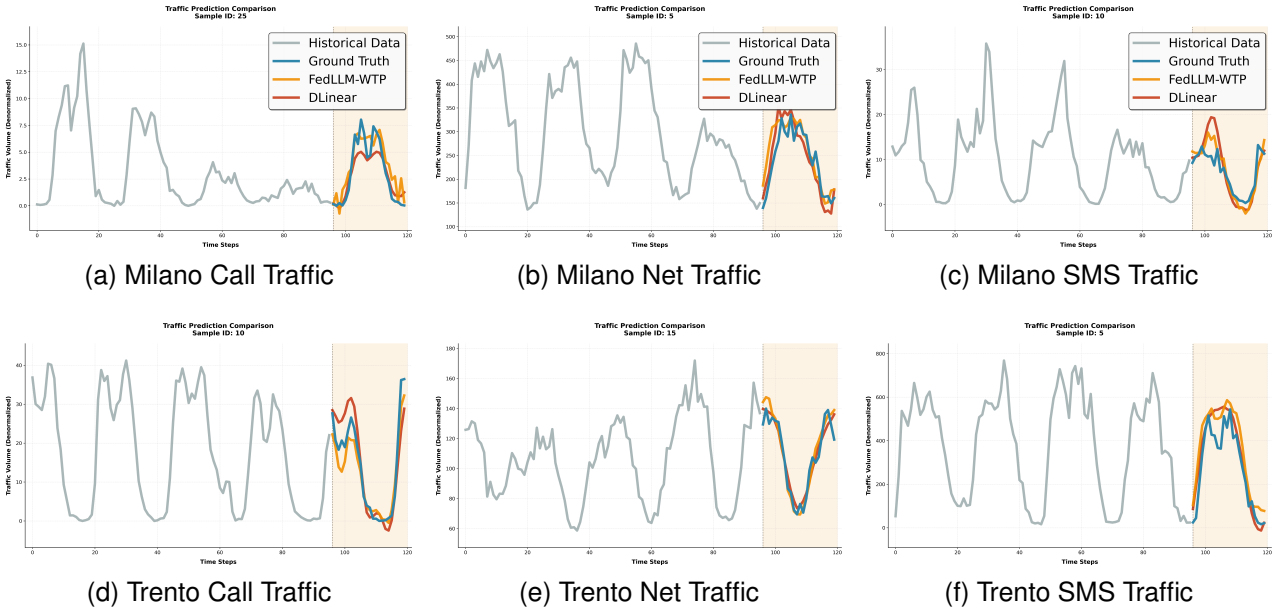

Fig. 2. Traffic Prediction Results Comparison on Milano and Trento Datasets

good data scalability. This excellent few-shot learning capability is of great significance for practical deployment, especially in newly built base stations or at the early stages of data collection, enabling the rapid establishment of reliable prediction models.

IV. Conclusion

This paper proposes FLOW, a federated wireless traffic forecasting framework that integrates federated learning with large language models. The framework achieves accurate modeling of complex temporal dependencies while meeting edge computing resource constraints through a simplified modal alignment layer, LoRA fine-tuned large model layer, and output mapping layer. FLOW outperforms traditional baseline methods, achieving an average MSE reduction of 11.1% on the Milano dataset and 10.9% on the Trento dataset. Notably, small parameter models perform better than large-scale models in federated environments, providing essential guidance for edge deployment. Additionally, FLOW demonstrates excellent few-shot learning and zero-shot cross-domain generalization capabilities, proving effective transfer of large language model pre-trained knowledge in time series prediction. Ablation experiments validate the importance of federated aggregation and LoRA fine-tuning. This work presents a novel solution for intelligent traffic forecasting in distributed environments, offering significant practical value for wireless network operations.

Acknowlegement

This work was supported in part by NSFC under No. 62401338, in part by the Joint Funds of the NSFC under Grant No. U22A2003, in part by the Shandong Province Excellent Youth Science Fund Project (Overseas) under Grant No. 2024HWYQ-028, and by the Fundamental Research Funds of Shandong University.

References

[1] H. Chai, S. Zhang, X. Qi, B. Qiu, and Y. Li, "Uomo: A universal model of mobile traffic forecasting for wireless network optimization," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, (New York, NY, USA), p. 4308–4319, Association for Computing Machinery, 2025.

[2] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1389–1401, 2019.

[3] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1656–1659, 2018.

[4] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Expert Systems with Applications*, vol. 201, p. 117163, 2022.

[5] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless traffic prediction with scalable gaussian process: Framework, algorithms, and verification," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[6] Y. Shu, M. Yu, J. Liu, and O. Yang, "Wireless traffic modeling and prediction using seasonal arima models," in *IEEE International Conference on Communications, 2003. ICC '03.*, vol. 3, pp. 1675–1679 vol.3, 2003.

[7] W. Shen, H. Zhang, S. Guo, and C. Zhang, "Time-wise attention aided convolutional neural network for data-driven cellular traffic prediction," *IEEE Wireless Communications Letters*, vol. 10, no. 8, pp. 1747–1751, 2021.

[8] C. Zhang, H. Zhang, S. Dang, B. Shihada, and M.-S. Alouini, "Gradient compression and correlation driven federated learning for wireless traffic prediction," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 4, pp. 2246–2258, 2025.

[9] C. Zhang, S. Dang, B. Shihada, and M.-S. Alouini, "Dual attention-based federated learning for wireless traffic prediction," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10, 2021.

[10] L. Zhang, C. Zhang, and B. Shihada, "Efficient wireless traffic prediction at the edge: A federated meta-learning approach," *IEEE Communications Letters*, vol. 26, no. 7, pp. 1573–1577, 2022.

[11] F. Gao, C. Zhang, J. Qiao, and A. Dong, "Wireless traffic prediction with l2 paradigm optimized federated learning," in *2024 IEEE 24th International Conference on Communication Technology (ICCT)*, pp. 953–957, 2024.
[12] D. Sun, C. Zhang, J. Qiao, T. Li, and H. Zhang, "Generative pretrained transformer for wireless traffic prediction," in *Wireless Artificial Intelligent Computing Systems and Applications* (Z. Cai, Y. Zhu, Y. Wang, and M. Qiu, eds.), (Singapore), pp. 370–381, Springer Nature Singapore, 2025.
[13] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Scientific Data*, vol. 2, p. 150055, 10 2015.
[14] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. USA: Prentice Hall PTR, 3rd ed., 1994.
[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, pp. 267–288, 12 2018.
[16] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the 10th International Conference on Neural Information Processing Systems*, NIPS'96, (Cambridge, MA, USA), p. 155–161, MIT Press, 1996.
[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[18] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," 2020.
[19] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," 2022.
[20] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen, "Time-llm: Time series forecasting by reprogramming large language models," 2024.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.
[22] DeepSeek-AI, "Deepseek-v3 technical report," 2025.