AUTOFED: PERSONALIZED FEDERATED TRAFFIC PREDICTION VIA ADAPTIVE PROMPT

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate traffic prediction underpins Intelligent Transportation Systems, but privacy concerns confine traffic data to silos and force most existing methods to train locally. Federated Learning (FL) enables collaborative training, yet standard FL struggles with the non-IID data of geographically dispersed clients, and existing Personalized FL (PFL) frameworks for traffic prediction require extensive dataset-specific adaptation, including hand-crafted graphs, per-dataset hyper-parameter tuning, and manual selection of pattern-extraction filters. We propose AutoFed, a PFL framework that removes this manual design. Inspired by prompt learning, AutoFed uses a federated representor with a client-aligned adapter to distill local data into a compact prompt matrix produced by a globally shared generator, and this prompt conditions a personalized predictor, so each client benefits from cross-client knowledge while retaining local specificity. Because the graph encoder stays private and only feature-axis modules are shared, no node-indexed tensor enters the shared block, so aggregation never has to reconcile clients whose road networks differ in size. AutoFed therefore supports clients with different numbers of nodes, at a per-round communication cost independent of graph size. On real-world travel demand and traffic flow datasets, AutoFed is competitive with or better than the strongest baseline across the federated scenarios we study, including a ten-client setting with widely differing client sizes, while communicating more than 30 times fewer parameters per round than full-model federation. Code is available at `https://anonymous.4open.science/r/AutoFed-3440`.

1 INTRODUCTION

Traffic prediction (TP) underpins modern Intelligent Transportation Systems, from ride-hailing dispatch to congestion management. Its data is abundant but also sensitive and fragmented Zhang et al. (2024a): privacy regulations and commercial interests confine it to silos held by municipal agencies in different jurisdictions or by private operators such as Uber and Lyft. Centralized training on the union of these silos is therefore generally infeasible, while local-only training generalizes poorly because each silo sees too narrow a slice of traffic behaviour.

Federated Learning (FL) has emerged as a promising paradigm for collaboratively training models without sharing raw data. Conventional FL algorithms such as FedAvg McMahan et al. (2017) assume client data are independent and identically distributed (IID), which rarely holds for traffic prediction, where patterns vary across regions with urban layout, population density, and economic activity. A single global model therefore drifts and underperforms for many participants. Personalized Federated Learning (PFL) addresses this by letting clients learn customized models while still drawing on collective knowledge Tan et al. (2022).

Existing PFL frameworks are nevertheless not directly suited to TP, which additionally involves complex spatiotemporal dependencies, non-stationary fluctuations, and numerous external features Aouedi et al. (2025). Adapting them typically requires manual effort: intricate graph construction Shang & Li (2025b;a); Yuan et al. (2022) or hyper-parameter tuning Hu et al. (2024); Zhou et al. (2025; 2024). In Hu et al. (2024); Zhou et al. (2025), for example, the filter method and the number of patterns alone shift performance by more than 5% on the datasets those papers study, and the optimal setting differs across datasets, so it can only be chosen with prior knowledge of the data and the client configuration. This is a practical barrier to scalable deployment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A deeper obstacle lies beneath these manual efforts. Most PFL methods share knowledge by aggregating model parameters, yet in traffic prediction different clients own road networks with different numbers of nodes, so any parameter tied to the node axis, such as the adaptive node embedding or adjacency of a graph model, has an incompatible shape across clients and cannot be averaged element-wise. Prior work sidesteps this either by assuming homogeneous clients or by imposing a hand-crafted unified graph, which is precisely what drives the manual design burden above. This motivates a different route: leave every node-dependent parameter private, and share only modules whose parameters live on the feature axis and therefore have the same shape at every client.

To this end, we propose AutoFed, a PFL framework for traffic prediction that keeps the personalized predictor intact and moves the collaboration into the prompt. AutoFed pairs a Personalized Predictor (PP) that adapts to local data distributions with a Federated Representor (FR) that distills each client's data into a compact prompt matrix generated by a globally shared module. This prompt conditions the PP as a prefix token, so the only quantities that cross the network boundary are the feature-axis parameters of the prompt generator, while every node-dependent parameter stays private. We adopt the Adaptive Graph Convolutional Recurrent Network (AGCRN) Bai et al. (2020) as an off-the-shelf backbone rather than as a contribution, so that our novelty sits in the federated collaboration mechanism itself. Our contributions are threefold. First, we introduce a federated representor that aligns heterogeneous local representations into a shared prompt space via a client-aligned adapter, enabling clients with differing graph sizes to collaborate in a common parameter space without any hand-crafted unified graph. Each client instantiates its own prompt locally and never transmits it, so the exchanged quantity is the parameter set of the shared prompt generator, a few thousand values that do not depend on the number of nodes. Full-model federation instead transmits the AGCRN backbone, which carries node-indexed tensors that grow with the graph. That is where standard parameter-aggregation PFL does not directly apply. Second, the prompt is produced by an auto-encoder that compresses the per-node feature vector and an adapter that aligns clients, both shared and both acting only on the per-node feature axis, which replaces the hand-crafted graph construction and pattern-filter selection that prior work must re-tune per dataset with a single reconstruction objective, while keeping only a few standard hyper-parameters that are fixed across all datasets. Third, experiments on real-world travel demand and traffic flow data show AutoFed matching or improving on the strongest baseline across federated scenarios at low communication overhead, and a representation-level analysis reveals how the prompt aligns clients and what traffic information it encodes.

2 METHODOLOGY

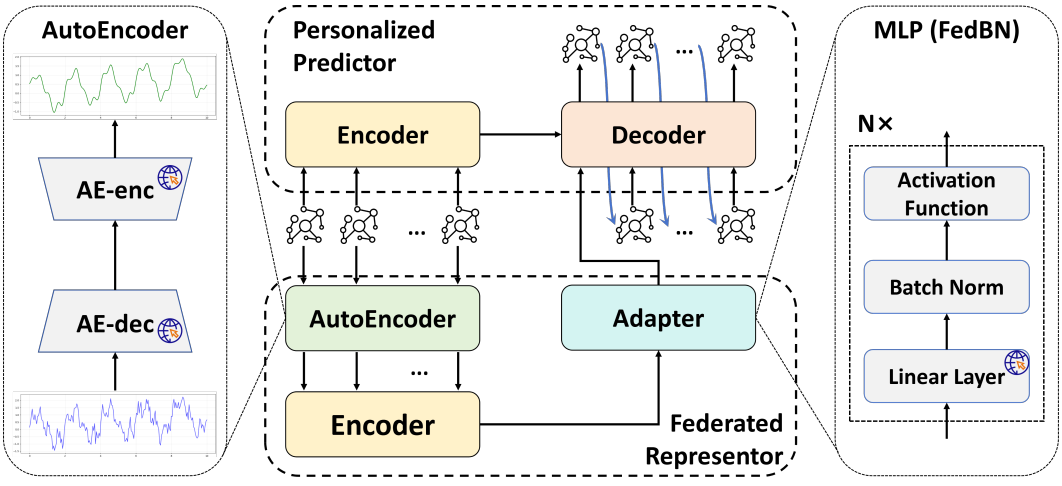


Figure 1: Overview of AutoFed. The Personalized Predictor is a graph time series encoder-decoder trained locally. The Federated Representor combines an auto-encoder, a graph time series encoder, and a client-aligned adapter to produce the prompt matrix consumed by the predictor decoder. Modules marked with the globe icon are shared.

2.1 PROBLEM SETUP

We study the federated TP task, which involves $\mathcal{N}$ clients $\mathcal{C} = \{C_1, C_2, \ldots, C_{\mathcal{N}}\}$ with non-IID data and a centralized server S. For each client $i \in \{1, \ldots, \mathcal{N}\}$, the transportation network can be represented as $G_i = (V_i, \mathcal{E}_i)$, where V_i and $\mathcal{E}_i$ denote the nodes and edges, respectively, and $|\cdot|$ denotes the size of a set. Additionally, $A_i \in \mathbb{R}^{|V_i|\times|V_i|}$ is the weighted adjacency matrix, representing the relationship, for instance spatial or dependency based, among traffic nodes. We write $n_i := |V_i|$ for the number of nodes of client i, the quantity that differs across clients. Each client possesses a private dataset $D_i = \{X_i, Y_i\}$, where $X_i \in \mathbb{R}^{S_i,\mathcal{T},n_i,K}$ represents the historical traffic features and $Y_i \in \mathbb{R}^{S_i,\mathscr{T},n_i}$ is the ground truth for future demand. Here S_i denotes the number of training samples of client i, $\mathcal{T}$ and $\mathscr{T}$ are the lengths of the historical information sequence and the future horizon to be predicted, respectively, and K is the feature dimension.

Within this scenario, the network of client i at round m is written $\mathrm{f}(\cdot; \theta_i^m, \Theta^m)$, where the private part θ_i^m learns the aspects of the data specific to that client's distribution and the shared part Θ^m learns a global representation common to all clients. Consequently, the target of PFL is defined as:

$$\arg \min_{\theta_1,\theta_2,\ldots,\theta_{\mathcal{N}},\Theta} \sum_{i=1}^{\mathcal{N}} \frac{|V_i|}{\sum_{i=1}^{\mathcal{N}} |V_i|} \mathrm{L}(\mathrm{f}(X_i; \theta_i, \Theta), Y_i) \,, \tag{1}$$

where $\mathrm{L}(\cdot, \cdot)$ is the loss function.

2.2 METHOD OVERVIEW

Fig. 1 shows the two components of AutoFed. The Personalized Predictor (PP) is an encoder-decoder graph time series network trained locally. The encoder receives historical data $x \in \mathbb{R}^{\mathcal{T},n,K}$, and the decoder predicts $\hat{y} \in \mathbb{R}^{\mathscr{T},n}$ auto-regressively. For client i, x comes from X_i with $n = |V_i|$. The Federated Representor (FR) generates a prompt matrix $p_g \in \mathbb{R}^{n,k}$ from the same input x, where k is the hidden dimension, and the PP decoder consumes it as its first token. The design rests on the intuition that clients may share traffic patterns Zhou et al. (2024; 2025), so the prompt can convey pattern information from x and guide decoding. During training the FR is partially shared among clients, which lets it learn a common pattern representation. Here "shared" refers to the prompt generator only, since each client computes its own p_g locally.

We now make the split between θ_i and Θ explicit, because it is what allows clients with different node counts to participate. The shared block Θ comprises the auto-encoder and the shared linear layers of the adapter, both of which act on the per-node feature axis: the auto-encoder maps K input features to k and back, and the adapter linear maps k to k. Their parameters therefore have the same shape at every client and can be averaged element-wise even when road networks differ in size. Everything that touches the node axis is private: the AGCRN predictor with its learnable node embedding $P_i \in \mathbb{R}^{n_i\times k}$, the personalized FR encoder, and all batch normalization layers. The node embedding is exactly the tensor whose shape differs across clients, and it is never aggregated. The prompt matrix $p_g \in \mathbb{R}^{n_i\times k}$ is likewise a local activation fed to the private decoder, not a message. Only Θ is transmitted, with size $O(k^2 + Kk)$ independent of n_i. Proposition 1 states this formally. An adversary who observes Θ sees the aggregate of the auto-encoder and adapter layers, which is the exposure any parameter-averaging scheme carries. The split confines what is specific to one client to the private block, so the tensors carrying a client's spatial structure, namely the node embedding and the derived adjacency, never leave the client. A deployment with strict confidentiality requirements can compose the shared block with secure aggregation or a noise mechanism.

2.3 PERSONALIZED PREDICTOR

Our framework needs a graph time series backbone, and we use one with an adaptive graph structure, the AGCRN Bai et al. (2020). Most traffic prediction models rely on manually designed graphs Shang & Li (2025b); Shang et al. (2025), which perform well but are slow to construct and require weighing spatial connections, distances, and node similarities, whose usefulness varies across regions and datasets. AGCRN instead learns the graph structure through a node embedding, denoted $P \in \mathbb{R}^{n,k}$

(to avoid a clash with the number of local epochs). The AGCRN process can be represented as:

$$\begin{aligned} \tilde{A} &= \text{Softmax}(\text{ReLU}(PP^\top)) \,, \\ u_t &= \sigma(\text{GCN}_u(x_t, H_{t-1}, \tilde{A})) \,, \\ r_t &= \sigma(\text{GCN}_r(x_t, H_{t-1}, \tilde{A})) \,, \\ C_t &= \tanh(\text{GCN}_C(x_t, (r_t \odot H_{t-1}), \tilde{A})) \,, \\ H_t &= u_t \odot H_{t-1} + (1 - u_t) \odot C_t \,, \end{aligned} \tag{2}$$

where the matrix $\tilde{A} \in \mathbb{R}^{n,n}$ is the adaptive adjacency matrix, $x_t \in \mathbb{R}^{n,K}$ represents the input x at frame t (i.e. $x = \{x_1, x_2, \ldots, x_\mathcal{T}\}$), and $u_t, r_t, C_t, H_{t-1} \in \mathbb{R}^{n,k}$ represent the update gate, reset gate, candidate state, and previous hidden state of the GRU at frame t, respectively. Specifically, we employ the same AGCRN structure both the encoder and decoder in the PP.

2.4 Federated Representor

As shown in Fig. 1, the FR has three components: an auto-encoder that reduces the per-node feature vector to a lower-dimensional code, a graph time series encoder that compresses it while extracting spatial information, and a client-aligned adapter that transfers the local representation into a global feature prompt guiding the PP.

- **Auto-Encoder:** Each client observes K measurements per node at every time step, and that feature vector is wide relative to the compact prompt the decoder consumes. The auto-encoder reduces it and does nothing else: its encoder maps the K input features at each node to k latent features and its decoder maps them back, so both directions are $K \times k$ or $k \times K$ maps applied to every node independently. No parameter it owns combines two nodes, and the module therefore performs no operation across clients or across space. Its contribution is a dimensionality reduction on the per-node feature axis, which the graph encoder below builds on, and a parameter shape that no client's graph size can change. Sharing it across clients costs nothing in expressiveness, because the map it learns is one that every client applies to its own nodes.
  The module is trained by the reconstruction objective

$$\begin{aligned} p &= \text{AE-enc}(x) \,, \\ \hat{x} &= \text{AE-dec}(p) \,, \\ \delta^* &= \arg\min_\delta \|\hat{x} - x\|_1 \,, \end{aligned} \tag{3}$$

  so the encoder compresses $x \in \mathbb{R}^{\mathcal{T},n,K}$ into $p \in \mathbb{R}^{\mathcal{T},n,k}$ and the decoder reconstructs $\hat{x} \in \mathbb{R}^{\mathcal{T},n,K}$, with δ the AE parameters and $\|\cdot\|_1$ the reconstruction target.
  Compression has a second consequence that explains why this module serves the prompt well. Following the usual reading of traffic series Boto-Giralda et al. (2010); Zhou et al. (2025); Qin et al. (2024), write the input at a node as $x = \tilde{x} + \nu$, where $\tilde{x}$ is the regular pattern and ν is a stochastic remainder with zero mean. The slowly varying part of $\tilde{x}$ carries most of the amplitude of a traffic series and is what a k-dimensional code can be expected to preserve, whereas ν is unpredictable from the surrounding window and therefore contributes little to reducing the reconstruction error. A bottleneck of width k thus spends its capacity on the persistent pattern and drops the remainder, and the studies of auto-encoders under an information bottleneck Nolan et al. (2022); Kieu et al. (2022) report the same ordering, that low-frequency high-amplitude components are reconstructed first. The module therefore reaches the effect that a low-pass filter is chosen for, without anyone specifying a filter family or a cutoff frequency. Sharing it across clients follows from the same reading, since the property it exploits is a property of traffic series rather than of one road network, and Section 3 tests that choice against the ablation.
- **Graph Time Series Encoder:** The representation p still carries a sequence length, which cannot serve directly as the decoder's single conditioning token. We therefore compress it with a graph time series encoder that extracts additional spatial information in the same pass. Specifically, we adopt the same encoder structure as the PP, compressing the feature p into a single matrix $p_l \in \mathbb{R}^{n,k}$, referred to as the local feature. Given the differing graph structures and data distributions among clients, we opt for a personalized encoder for each client.
- **Client-Aligned Adapter:** To share knowledge among clients, we introduce a client-aligned adapter that transfers the local pattern representation p_l to a global representation $p_g \in \mathbb{R}^{n,k}$ through

a shared MLP. Because non-IID variation persists in the local feature space p_l, we apply the FedBN Li et al. (2021) strategy: the linear layer parameters are shared among clients, while the batch normalization layers stay distinct and keep their own statistics, namely the means, standard deviations, and rescaling parameters.

To summarize, the full process of the PP can be expressed as follows:

$$\begin{aligned} H_{\mathcal{T}} &= \text{Encoder}(x; \mathbf{0}) \ , \\ z &= \text{Decoder}(p_g; H_{\mathcal{T}}) \ , \\ \hat{y} &= \text{MLP}(z) \ , \end{aligned} \tag{4}$$

where $\mathbf{0}$ is the zero initial hidden state of the encoder, p_g is the decoder's first input token, and $H_{\mathcal{T}}$, the encoder's final hidden state, initializes the decoder. The decoder outputs $z \in \mathbb{R}^{\mathcal{T},n,k}$, which an additional MLP ($\mathbb{R}^k \to \mathbb{R}$) maps to $\hat{y} \in \mathbb{R}^{\mathcal{T},n}$. The intermediate z is needed because, being auto-regressive, the decoder requires every token to have the same output dimension as the first prompt token p_g. Encoder and decoder share a structure but take inputs of different dimensions ($x \in \mathcal{R}^{\mathcal{T},n,K}$ versus $p_g \in \mathcal{R}^{n,k}$), because the decoder is auto-regressive while the encoder processes the full sequence.

2.5 Training Process

Training couples two objectives, the reconstruction loss of the auto-encoder and the prediction loss of the PP. We take the mean absolute error for both, so that for client i,

$$\begin{aligned} L_i^{ae} &= \mathbf{E}_{x \in X_i} ||x - \hat{x}||_1 \ , \\ L_i^{pre} &= \mathbf{E}_{y \in Y_i} ||y - \hat{y}||_1 \ , \end{aligned} \tag{5}$$

where $|| \cdot ||_1$ denotes the L_1 norm. The overall loss function for client i is then defined as:

$$\begin{aligned} L_i &= L_i^{pre} + \alpha \, L_i^{ae} \ , \\ \alpha &= \text{sg}\left[\frac{L_i^{ae}}{L_i^{pre}} \right] , \end{aligned} \tag{6}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator, so the backward pass treats α as the constant of Eq. equation 6. Without it, substituting the second line into the first gives $L_i = L_i^{pre} + (L_i^{ae})^2 / L_i^{pre}$, whose gradient with respect to L_i^{pre} carries the factor $1 - (L_i^{ae}/L_i^{pre})^2$. That factor turns negative precisely when $L_i^{ae} > L_i^{pre}$, which is the regime the weighting rule is designed for, so the update would ascend on the prediction loss instead of descending. Treating α as a non-differentiable schedule removes this term and leaves a plain weighted gradient

$$\nabla L_i = \nabla L_i^{pre} + \alpha \, \nabla L_i^{ae}, \tag{7}$$

where the ratio adapts the balance over training without any schedule to set. Inspired by Zhao (2025); Chen et al. (2018); Kendall et al. (2018), this balances the two objectives: when the auto-encoder has a high error the prompt it produces is unreliable, so more weight on reconstruction is warranted before relying on that prompt for prediction, and conversely when prediction is the harder task.

3 Experiment

3.1 Experiment Setup

We compare AutoFed against general FL methods (FedAvg McMahan et al. (2017), FedProx Li et al. (2020)), PFL methods (FedPer Arivazhagan et al. (2019), pFedMe T Dinh et al. (2020)), and FL methods for traffic prediction (FedTPS Zhou et al. (2024; 2025), FedGCN Hu et al. (2024)), all on the same AGCRN backbone:

- **FedAvg McMahan et al. (2017):** This classical FL method enables each client to train on their individual data at each round, after which the server parameters are updated to the average of all clients' parameters, which are then distributed before the next round.
- **FedProx Li et al. (2020):** FedProx introduces a proximal term to the loss function of FedAvg, preventing the local model from deviating too far from the global model. However, it still aims to optimize a global model rather than personalized ones.

- **FedPer Arivazhagan et al. (2019):** As a classical PFL method, FedPer allows different clients to share a common base layer for robust feature extraction while maintaining personalized output layers for tailored results.
- **pFedMe T Dinh et al. (2020):** pFedMe is a PFL framework that uses Moreau envelopes as a regularized loss function, enabling each client to efficiently find its optimal model while leveraging global information.
- **FedTPS Zhou et al. (2024; 2025):** FedTPS shares knowledge through a traffic pattern repository, using a low-pass filter for stable pattern extraction and a fixed set of patterns as the conditioning signal. Both the filter and the number of patterns are manual settings that must be re-tuned per dataset, and aggregating the repository by client similarity adds computation.
- **FedGCN Hu et al. (2024):** FedGCN assumes that potential neighbors of all real nodes lie within the graph, and introduces the MendGCN module to account for these external factors.

Because clients have different numbers of nodes, the parameter-aggregation baselines require an explicit convention, which we state because it affects their results. For FedAvg, FedProx, FedPer, and pFedMe we aggregate exactly those backbone tensors whose shape is identical across clients: the recurrent weights, the graph-convolution weights, and the feature-axis projections. Tensors tied to the node axis cannot be element-wise averaged when $|V_i|$ differs, so they stay local and are never aggregated. This covers the AGCRN node embedding P_i and the adaptive adjacency derived from it. Where a baseline shares the node embedding by construction we follow that baseline's own convention, so FedGCN shares its graph-convolution weights and keeps the per-client graph. We flag this because it is a genuine advantage of the setting rather than a courtesy to the baselines: with the node-dependent block retained locally, parameter-averaging methods lose the tensor that carries spatial structure, whereas AutoFed has no such tensor in its shared block. All methods, baselines included, run under one configuration fixed across datasets and scenarios, so the gap is not attributable to per-method tuning. Local training, in which each client trains only on its own data, is a further baseline, and we ablate the auto-encoder and the FedBN adapter individually. Experiments run on a workstation with an Intel Core i7-14700KF CPU, an NVIDIA RTX 4080 GPU, and Windows 11.

We conduct two case studies, on Travel Demand Prediction (TDP) and Traffic Flow Prediction (TFP), both with real-world data. For TDP we use ride-hailing demand for Uber and Lyft across the five districts of New York City from the NYC Taxi and Limousine Commission dataset Deri & Moura (2015). Treating all Uber and Lyft data as two clients gives scenario S0, in which the graph spans the whole city. This is not optimal: traffic demand in Manhattan has limited impact on Queens, since most ride-hailing activity is confined within a district, and merging districts into one graph introduces interference that complicates training, particularly for models with adaptive adjacency matrices such as AGCRN. Splitting the city-wide graph into regional subgraphs predicts better (Table 1), so we treat each company within each district as a separate client, a choice motivated by modelling effectiveness rather than by privacy. This yields S1 (five Uber clients), S2 (five Lyft clients), and S3 (all ten clients). For TFP we use PEMS03, PEMS04, and PEMS08 from the CalTrans Performance Measurement System Chen (2002), which cover different regions and periods of California highways, partitioning each into 2, 4, 6, 8, and 10 clients and reporting the average over these setups.

In the experiments, we split the data into training, validation, and test sets chronologically with a ratio of 6:2:2, taking contiguous blocks in time rather than random samples, so that no test interval precedes a training interval. Input windows are formed with a stride of one and adjacent windows overlap, as is standard for these benchmarks. The batch size is set to 128, and the learning rate is 10^{-3} with the Adam optimizer. For federated learning configurations, we set the number of communication rounds to 50 and use one local epoch. All experiments were repeated three times, and the standard deviations are reported in the tables. During experiment, we consider the following evaluation metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Square Error (MSE), and Mean Absolute Percentage Error (MAPE). For MAPE we follow the convention of the PEMS benchmarks and compute it only over entries whose ground-truth demand is above a small threshold, since the quantity is otherwise undefined where demand is zero. The TDP series have no zero-demand entries, so no threshold is applied there. Inputs are scaled per client by the mean and standard deviation of that client's training split, and targets are scaled by the same statistics, which is what makes the two loss terms in Eq. equation 6 comparable in magnitude. All three runs of a setting share one data partition and differ only in initialization and shuffling order, and we report their mean and standard deviation.

Table 1: Model performance on the TDP task. Best results in **bold**, with ± giving the standard deviation over three runs. The same convention applies to all later tables.

Type	Method	Uber (S0)				Lyft (S0)			
		MAE	RMSE	MSE	MAPE/%	MAE	RMSE	MSE	MAPE/%
	Local Training	3.70±0.13	6.25±0.27	39.17±3.41	40.15±0.69	2.08±0.10	3.31±0.17	10.95±1.09	51.47±1.00
FL	FedAvg McMahan et al. (2017)	3.58±0.09	6.08±0.21	37.17±2.69	39.15±0.35	1.96±0.08	3.13±0.14	9.79±0.89	49.41±0.50
	FedProx Li et al. (2020)	3.55±0.02	5.96±0.06	35.68±0.78	40.34±0.41	1.88±0.08	2.97±0.15	8.83±0.91	49.67±0.46
PFL	FedPer Arivazhagan et al. (2019)	3.67±0.14	6.23±0.31	39.04±3.83	40.24±1.29	1.90±0.05	2.99±0.10	8.93±0.61	50.84±0.15
	pFedMe T Dinh et al. (2020)	3.63±0.04	6.19±0.13	38.42±1.63	40.13±0.21	2.08±0.11	3.32±0.20	11.09±1.32	51.85±1.11
TrafficFL	FedTPS Zhou et al. (2024; 2025)	3.65±0.15	6.23±0.33	39.00±4.13	39.43±0.81	1.95±0.07	3.10±0.13	9.63±0.79	50.58±0.29
	FedGCN Hu et al. (2024)	**3.44±0.01**	**5.72±0.06**	**32.83±0.74**	38.61±0.03	1.85±0.03	2.94±0.06	8.68±0.36	**48.57±0.16**
Ours	AutoFed	3.48±0.11	5.92±0.24	35.18±2.82	**38.44±0.45**	**1.84±0.03**	**2.90±0.05**	**8.43±0.28**	48.85±0.27
	w/o AE	3.67±0.23	6.20±0.38	38.62±4.73	39.88±0.75	2.07±0.17	3.30±0.29	10.95±1.94	50.82±1.80
	w/o FedBN	3.66±0.12	6.20±0.27	38.56±3.29	40.20±0.55	1.98±0.02	3.17±0.04	10.04±0.24	50.21±0.41

Type	Method	Uber (S1)				Lyft (S2)			
		MAE	RMSE	MSE	MAPE/%	MAE	RMSE	MSE	MAPE/%
	Local Training	3.45±0.05	5.43±0.10	33.69±1.56	39.45±0.13	2.00±0.04	3.06±0.06	9.61±0.39	49.16±0.19
FL	FedAvg McMahan et al. (2017)	3.72±0.16	5.96±0.27	41.30±4.20	40.17±0.10	2.02±0.03	3.09±0.06	9.78±0.38	49.85±0.02
	FedProx Li et al. (2020)	3.63±0.12	5.83±0.22	39.02±2.40	40.09±0.56	1.96±0.07	3.00±0.12	9.54±0.84	51.92±0.46
PFL	FedPer Arivazhagan et al. (2019)	3.75±0.15	5.95±0.26	40.40±3.50	40.87±0.37	1.95±0.06	3.01±0.10	9.57±0.67	51.03±0.23
	pFedMe T Dinh et al. (2020)	3.45±0.07	5.47±0.14	33.94±1.89	39.70±0.38	1.92±0.05	2.95±0.10	9.21±0.67	50.64±0.44
TrafficFL	FedTPS Zhou et al. (2024; 2025)	3.52±0.06	5.54±0.09	34.75±0.91	39.76±0.51	1.87±0.02	2.86±0.03	8.62±0.15	50.56±0.30
	FedGCN Hu et al. (2024)	3.58±0.11	5.71±0.21	36.34±2.43	39.52±0.39	2.02±0.05	3.10±0.09	9.91±0.56	**48.90±0.21**
Ours	AutoFed	**3.40±0.04**	**5.34±0.07**	**32.43±0.92**	38.89±0.36	**1.84±0.03**	**2.80±0.05**	**8.37±0.31**	49.70±0.36
	w/o AE	3.46±0.06	5.44±0.12	33.64±1.44	38.81±0.52	1.89±0.03	2.91±0.06	8.98±0.33	49.88±0.34
	w/o FedBN	3.43±0.07	5.41±0.13	32.98±1.25	**38.75±0.26**	1.89±0.03	2.89±0.06	8.84±0.39	50.35±0.12

Type	Method	Uber (S3)				Lyft (S3)			
		MAE	RMSE	MSE	MAPE/%	MAE	RMSE	MSE	MAPE/%
	Local Training	3.45±0.05	5.43±0.10	33.69±1.56	39.45±0.13	2.00±0.04	3.06±0.06	9.61±0.39	49.16±0.19
FL	FedAvg McMahan et al. (2017)	3.73±0.12	6.02±0.12	42.20±0.77	41.43±1.45	1.93±0.06	2.99±0.11	9.42±0.69	50.09±0.57
	FedProx Li et al. (2020)	3.73±0.16	6.01±0.31	43.09±5.37	40.97±1.04	2.03±0.07	3.12±0.13	10.05±0.78	**48.81±0.51**
PFL	FedPer Arivazhagan et al. (2019)	3.90±0.20	6.17±0.33	43.86±4.81	41.34±0.63	1.97±0.05	3.05±0.10	9.90±0.67	51.55±0.26
	pFedMe T Dinh et al. (2020)	3.44±0.03	5.42±0.04	33.30±0.25	39.99±0.13	1.90±0.04	2.91±0.07	8.92±0.41	50.69±0.18
TrafficFL	FedTPS Zhou et al. (2024; 2025)	3.45±0.03	5.45±0.03	33.85±0.20	39.24±0.20	1.91±0.05	2.94±0.09	9.13±0.54	50.36±0.25
	FedGCN Hu et al. (2024)	3.57±0.07	5.66±0.10	35.79±0.93	40.01±0.62	1.95±0.04	3.01±0.06	9.51±0.39	51.32±1.0
Ours	AutoFed	**3.36±0.04**	**5.27±0.07**	**31.56±0.89**	**38.82±0.49**	**1.85±0.02**	**2.82±0.04**	**8.40±0.19**	50.33±0.29
	w/o AE	3.47±0.06	5.48±0.11	34.16±1.22	39.00±0.19	1.87±0.03	2.87±0.06	8.82±0.37	49.59±0.17
	w/o FedBN	3.43±0.06	5.39±0.08	32.93±0.83	39.69±0.32	1.88±0.03	2.88±0.06	8.79±0.40	50.53±0.18

3.2 Experiment Results

Tables 1 and 2 report the TDP and TFP results, each as the mean ± standard deviation over three runs.

Three observations follow from Table 1. First, on S0, where each operator acts as one city-wide client, FedGCN is better on Uber MAE, RMSE, and MSE while AutoFed is better on the remaining five columns, its Lyft MAPE of 48.85 sitting inside one standard deviation of FedGCN's 48.57. S0 is the hardest scenario for every method, which supports the motivation above, since a single city-wide graph forces an adaptive-adjacency model such as AGCRN to reconcile heterogeneous local dynamics within one set of nodes. Second, in the regional scenarios S1 to S3 AutoFed ranks first in almost every metric, and the gains hold as the client count grows from five to ten, the only place where the count varies without being averaged over. Third, S3, with ten clients of widely differing sizes, leaves AutoFed essentially unchanged relative to S1 and S2, while the baselines do not degrade uniformly, as FedAvg's Uber MAE rises from 3.72 to 3.73 even as its Lyft MAE falls from 2.02 to 1.93. No baseline closes the gap there, consistent with the stability analysis of Appendix E (Remark 2), where an outlier client perturbs the shared prompt generator only in proportion to its aggregation weight.

The ablations separate the two modules. Removing the FedBN adapter degrades every metric in every scenario, most where clients are most heterogeneous, consistent with its role in absorbing client statistics. The auto-encoder behaves differently across the two tasks. On TDP its removal is clearly harmful, costing up to 0.24 MAE on S0. On TFP the two variants lie within one standard deviation on PEMS03 and PEMS08, and the ablated model is marginally better on PEMS04 MAE and MAPE (20.76 vs 20.79 and 14.32 vs 14.72). This is the pattern the compression reading predicts rather than an anomaly: the flow benchmarks are smoother than the demand series, so a bottleneck at this width has less redundant variation to remove and the adapter carries more of the load. The adapter is what

Table 2: Model performance on the TFP task. Each cell averages a metric over the five client counts $(2, 4, 6, 8, 10)$ and three runs.

Dataset	Type	Method	MAE	RMSE	MSE	MAPE/%
PEMS03		Local Training	18.01±0.26	27.66±0.39	907.39±16.28	18.11±1.16
	FL	FedAvg McMahan et al. (2017)	17.89±0.24	27.70±0.42	901.38±20.13	18.15±0.88
		FedProx Li et al. (2020)	17.81±0.35	27.50±0.39	885.76±29.52	18.36±0.93
	PFL	FedPer Arivazhagan et al. (2019)	18.00±0.18	27.80±0.56	909.52±12.73	18.90±0.93
		pFedMe T Dinh et al. (2020)	16.99±0.28	26.59±0.34	853.63±25.99	16.98±0.91
	TrafficFL	FedTPS Zhou et al. (2024; 2025)	17.19±0.16	26.89±0.61	869.70±12.07	17.19±0.40
		FedGCN Hu et al. (2024)	18.82±0.22	28.87±0.48	985.89±22.28	20.35±1.42
	Ours	AutoFed	**16.15±0.17**	**25.57±0.39**	**825.39±46.40**	**16.06±0.48**
		w/o AE	16.27±0.12	25.58±0.30	795.82±11.35	16.60±0.44
		w/o FedBN	16.93±0.31	26.50±0.52	849.11±28.88	17.27±0.63
PEMS04		Local Training	22.43±0.23	33.84±0.31	1204.04±8.75	16.11±0.39
	FL	FedAvg McMahan et al. (2017)	23.06±0.50	34.64±0.39	1266.97±41.00	16.90±0.87
		FedProx Li et al. (2020)	22.99±0.57	34.59±0.45	1263.92±47.18	16.45±0.77
	PFL	FedPer Arivazhagan et al. (2019)	22.99±0.06	34.68±0.29	1272.55±6.34	16.59±0.20
		pFedMe T Dinh et al. (2020)	21.40±0.30	32.77±0.52	1150.65±20.94	14.91±0.38
	TrafficFL	FedTPS Zhou et al. (2024; 2025)	21.83±0.42	33.49±0.79	1204.27±55.71	15.37±0.41
		FedGCN Hu et al. (2024)	22.92±0.39	34.86±0.26	1298.40±25.27	16.38±0.64
	Ours	AutoFed	**20.79±0.13**	**31.81±0.32**	**1084.61±5.91**	**14.72±0.49**
		w/o AE	20.76±0.07	31.90±0.31	1091.59±5.54	14.32±0.22
		w/o FedBN	21.94±0.26	33.47±0.47	1193.12±10.30	15.12±0.13
PEMS08		Local Training	18.65±0.36	27.64±0.53	790.42±23.90	12.44±0.45
	FL	FedAvg McMahan et al. (2017)	19.29±0.42	28.78±0.38	860.93±26.91	13.04±0.51
		FedProx Li et al. (2020)	19.26±0.16	28.72±0.14	857.98±12.54	13.02±0.28
	PFL	FedPer Arivazhagan et al. (2019)	19.18±0.31	28.62±0.33	851.10±20.90	12.71±0.36
		pFedMe T Dinh et al. (2020)	17.69±0.46	26.64±0.67	740.20±30.35	11.40±0.47
	TrafficFL	FedTPS Zhou et al. (2024; 2025)	17.97±0.41	27.05±0.64	764.11±29.99	11.81±0.21
		FedGCN Hu et al. (2024)	19.66±0.52	29.10±0.55	883.01±39.81	13.77±1.09
	Ours	AutoFed	**16.83±0.17**	**25.62±0.21**	**684.63±14.16**	**11.10±0.26**
		w/o AE	16.89±0.23	25.71±0.24	689.78±16.04	11.23±0.21
		w/o FedBN	17.96±0.33	27.16±0.44	769.56±26.49	11.80±0.39

non-IID adaptation depends on, and the auto-encoder contributes a learned reduction that needs no per-dataset filter selection.

AutoFed shares roughly 175K parameters and leaves several million private, so the natural alternative reading of the gap is capacity allocation rather than personalization, and the baselines separate the two. Local Training, the extreme of that spectrum, shares nothing and is worse than AutoFed everywhere, so private parameters alone are not sufficient. FedPer, with a fixed cut of the same backbone, is also worse everywhere and worse than Local Training on several TDP metrics, so a generic split does not reproduce our gains. What distinguishes AutoFed is the object being shared rather than the amount: its shared block produces a prompt recomputed per input window and reconditioned on each client's graph through the private encoder. The w/o AE row shares strictly fewer parameters and is still competitive on TFP, which is inconsistent with shared capacity alone driving the ranking.

On TFP, AutoFed obtains the best results on all three PEMS benchmarks. Each cell of Table 2 averages the five client counts and three runs, so it reports average performance rather than any single count. A method that degrades at large client counts and one that is flat produce the same average, and the per-count TFP breakdown that separates them would be dominated by variation across the partition draw. The scaling evidence is instead scenario S3 of Table 1, where each client count is a separate column and AutoFed's lead survives. The numbers differ slightly from FedTPS Zhou et al. (2024; 2025) because we removed tricks such as curriculum learning from their release, which could be applied to any method or to none. Appendix D shows that the decoder depends on the prompt rather than treating it as a passive side input: corrupting it at inference raises the test error by 43% to more than 200%.

4 CONCLUSION

We propose AutoFed, a personalized federated framework for traffic prediction that pairs a locally trained predictor with a federated representor turning non-IID client data into a shared prompt. Sharing only the feature-axis prompt generator while every node-dependent parameter stays private lets clients with road networks of differing sizes collaborate, holds per-round communication independent of graph size, and removes the per-dataset manual design that prior work repeats. Experiments on real demand and flow data show competitive accuracy at reduced cost, and interventions confirm that the decoder actively relies on the prompt.

AI USE STATEMENT

In this work, we used generative AI tools for paper polishing, including grammar and wording checks, verification and support of mathematical derivations, and coding. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the principles outlined in the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The code for this paper is available at `https://anonymous.4open.science/r/AutoFed-3440`.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328, 2021.

Ons Aouedi, Van An Le, Kandaraj Piamrat, and Yusheng Ji. Deep learning on network traffic prediction: Recent advances, analysis, and future directions. *ACM Computing Surveys*, 57(6):1–37, 2025.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Muhammad Tayyab Asif, Justin Dauwels, Chong Yang Goh, Ali Oran, Esmail Fathi, Muye Xu, Menoth Mohan Dhanya, Nikola Mitrovic, and Patrick Jaillet. Spatiotemporal patterns in large-scale traffic speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):794–804, 2013.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Daniel Boto-Giralda, Francisco J Díaz-Pernas, David González-Ortega, José F Díez-Higuera, Míriam Antón-Rodríguez, Mario Martínez-Zarzuela, and Isabel Torre-Díez. Wavelet-based denoising for traffic volume time series forecasting with self-organizing neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 25(7):530–545, 2010.

Zhijie Cai, Haolong Chen, and Guangxu Zhu. Feedsign: robust full-parameter federated fine-tuning of large models with extremely low communication overhead of one bit. *arXiv preprint arXiv:2501.17610*, 2025.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Ying Chang, Xiaohu Shi, Xiaohui Zhao, Zhaohuang Chen, and Deyin Ma. Dual prompt personalized federated learning in foundation models. *Scientific Reports*, 15(1):28026, 2025.

Chao Chen. *Freeway performance measurement system (PeMS)*. University of California, Berkeley, 2002.

Hang Chen, Collin Meese, Mark Nejad, and Chien-Chung Shen. Individualized federated learning for traffic prediction with error-driven aggregation. *Journal on Autonomous Transportation Systems*, 3 (4):1–31, 2026a.

Haolong Chen, Hanzhi Chen, Zijian Zhao, Kaifeng Han, Guangxu Zhu, Yichen Zhao, Ying Du, Wei Xu, and Qingjiang Shi. An overview of domain-specific foundation model: key technologies, applications and challenges. *Science China Information Sciences*, 69(1):111301, 2026b.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Guowen Dai, Jinjun Tang, Jie Zeng, Chen Hu, and Chuyun Zhao. Road network traffic flow prediction: A personalized federated learning method based on client reputation. *Computers and Electrical Engineering*, 120:109678, 2024.

Joya A Deri and José MF Moura. Taxi data in new york city: A network perspective. In *2015 49th asilomar conference on signals, systems and computers*, pp. 1829–1833. IEEE, 2015.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Yuchen Fang, Yanjun Qin, Haiyong Luo, Fang Zhao, Bingbing Xu, Liang Zeng, and Chenxing Wang. When spatio-temporal meet wavelets: Disentangled traffic forecasting via efficient spectral graph attention networks. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 517–529. IEEE, 2023.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025.

Na Hu, Wei Liang, Dafang Zhang, Kun Xie, Kuanching Li, and Albert Y Zomaya. Fedgcn: A federated graph convolutional network for privacy-preserving traffic prediction. *IEEE Transactions on Sustainable Computing*, 9(6):925–935, 2024.

Yu-Hao Huang, Chang Xu, Yueying Wu, Wu-Jun Li, and Jiang Bian. Timedp: Learning to generate multi-domain time series with domain prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17520–17527, 2025.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024.

Seerat Kaur, Sukhjit Singh Sehra, Darisuh Ebrahimi, and Emad A Mohammed. A traffic-aware federated learning prediction framework with custom aggregation. *Machine Learning with Applications*, pp. 100829, 2025.

Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Michael Chen. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation research part C: Emerging technologies*, 85:591–608, 2017.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Tung Kieu, Bin Yang, Chenjuan Guo, Christian S Jensen, Yan Zhao, Feiteng Huang, and Kai Zheng. Robust and explainable autoencoders for unsupervised time series outlier detection. In *2022 IEEE 38th International conference on data engineering (ICDE)*, pp. 3038–3050. IEEE, 2022.

Wei-Bin Kou, Qingfeng Lin, Ming Tang, Rongguang Ye, Shuai Wang, Guangxu Zhu, and Yik-Chung Wu. Fast-convergent and communication-alleviated heterogeneous hierarchical federated learning in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Jia Li, Xiangguo Sun, Yuhan Li, Zhixun Li, Hong Cheng, and Jeffrey Xu Yu. Graph intelligence with large language models and prompt learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6545–6554, 2024.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

Zongqian Li, Yixuan Su, and Nigel Collier. A survey on prompt tuning. *arXiv preprint arXiv:2507.06085*, 2025.

Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1010–1018, 2011.

Jun Luo, Chen Chen, and Shandong Wu. Mixture of experts made personalized: Federated prompt learning for vision-language models. In *International Conference on Learning Representations*, volume 2025, pp. 58061–58080, 2025.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Michael Nolan, Bijan Pesaran, Eli Shlizerman, and Amy Orsborn. Multi-block rnn autoencoders enable broadband ecog signal reconstruction. *bioRxiv*, pp. 2022–09, 2022.

Jiaming Pei, Wenxuan Liu, Jinhai Li, Lukun Wang, and Chao Liu. A review of federated learning methods in heterogeneous scenarios. *IEEE Transactions on Consumer Electronics*, 70(3):5983–5999, 2024.

Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: a systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, 21(2):1–31, 2021.

Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohri, Yu Sun, and Manzil Zaheer. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pp. 17716–17758. PMLR, 2022.

Yutao Qin, Chuliang Wu, and Yao Hu. Traffic flow prediction by an ensemble framework with data denoising and functional principal components analysis. *Transportation Research Record*, 2678 (11):929–947, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Jonathan Scott, Hossein Zakerinia, and Christoph H Lampert. Pefll: Personalized federated learning by learning to learn. In *The Twelfth International Conference on Learning Representations*, 2024.

Yitong Shang and Sen Li. Security-enhanced spatiotemporal ride-hailing demand prediction—part ii: Vertical federated learning. *IEEE Transactions on Intelligent Transportation Systems*, 26(9): 13363–13377, 2025a. doi: 10.1109/TITS.2025.3564629.

Yitong Shang and Sen Li. Security-enhanced spatiotemporal ride-hailing demand prediction—part i: Horizontal federated learning. *IEEE Transactions on Intelligent Transportation Systems*, 2025b.

Yitong Shang, Duo Li, Yang Li, and Sen Li. Explainable spatiotemporal multi-task learning for electric vehicle charging demand prediction. *Applied Energy*, 384:125460, 2025.

Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. Federated adaptive prompt tuning for multi-domain collaborative learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 15117–15125, 2024.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in neural information processing systems*, 33:21394–21405, 2020.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12):9587–9603, 2022.

Xing Wang, Jixiang Chen, Feifei Chen, Biao Jin, Fumin Zou, Lyuchao Liao, and Ruihao Zeng. Fedctpm: A federated cross-regional collaboration learning approach for traffic flow prediction. *Expert Systems with Applications*, pp. 130779, 2025.

Ying Wang, Renjie Fan, Bo Gong, Hong Wen, and Yuanxi Yu. At-spnet: A personalized federated spatio-temporal modeling method for cross-city traffic prediction. *Concurrency and Computation: Practice and Experience*, 38(3):e70577, 2026.

Hao Xue and Flora D Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6851–6864, 2023.

Bailin Yang, Shulin Sun, Jianyuan Li, Xianxuan Lin, and Yan Tian. Traffic flow prediction using lstm with feature enhancement. *Neurocomputing*, 332:320–327, 2019.

Fu-En Yang, Chien-Yi Wang, and Yu-Chiang Frank Wang. Efficient model personalization in federated learning via client-specific prompt generation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19102–19111. IEEE, 2023.

Linghua Yang, Wantong Chen, Xiaoxi He, Shuyue Wei, Yi Xu, Zimu Zhou, and Yongxin 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*, pp. 6105–6116, 2024.

Xiaoming Yuan, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi. Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 24(8):8738–8748, 2022.

Lun Zhang, Qiuchen Liu, Wenchen Yang, Nai Wei, and Decun Dong. An improved k-nearest neighbor model for short-term traffic flow prediction. *Procedia-Social and Behavioral Sciences*, 96:653–662, 2013.

Rongqing Zhang, Jingxin Mao, Hanqiu Wang, Bing Li, Xiang Cheng, and Liuqing Yang. A survey on federated learning in intelligent transportation systems. *IEEE Transactions on Intelligent Vehicles*, 2024a.

Yu Zhang, Hua Lu, Ning Liu, Yonghui Xu, Qingzhong Li, and Lizhen Cui. Personalized federated learning for cross-city traffic prediction. In *IJCAI*, pp. 5526–5534, 2024b.

Zijian Zhao. Let network decide what to learn: Symbolic music understanding model based on large-scale adversarial pre-training. In *Proceedings of the 2025 International Conference on Multimedia Retrieval*, pp. 2128–2132, 2025.

Zijian Zhao, Fanyi Meng, Zhonghao Lyu, Hang Li, Xiaoyang Li, and Guangxu Zhu. Csi-bert2: A bert-inspired framework for efficient csi prediction and classification in wireless communication and sensing. *IEEE Transactions on Mobile Computing*, pp. 1–17, 2025. doi: 10.1109/TMC.2025.3640420.

Hang Zhou, Wentao Yu, Sheng Wan, Yongxin Tong, Tianlong Gu, and Chen Gong. Traffic pattern sharing for federated traffic flow prediction with personalization. In *2024 IEEE International Conference on Data Mining (ICDM)*, pp. 639–648. IEEE, 2024.

Hang Zhou, Wentao Yu, Sheng Wan, Yongxin Tong, Tianlong Gu, and Chen Gong. Fedtps: traffic pattern sharing for personalized federated traffic flow prediction. *Knowledge and Information Systems*, pp. 1–27, 2025.

Hang Zhou, Wentao Yu, Yang Wei, Guangyu Li, Sha Xu, and Chen Gong. Inter-client dependency recovery with hidden global components for federated traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 28946–28954, 2026.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A Related Work

A.1 Traffic Prediction

Traffic prediction (TP) encompasses travel demand, traffic flow, and travel time prediction. Early work treated each traffic node as an independent entity, applying statistical methods Liu et al. (2011), cluster-based pattern matching Zhang et al. (2013), and neural networks Yang et al. (2019), which overlooks the spatial relationships among nodes even though a traffic jam in one area usually raises flow in neighbouring regions. Most subsequent methods therefore extract spatial features within each time frame and then apply a temporal network, for example the CNN-LSTM of Liu et al. (2011); Asif et al. (2013); Ke et al. (2017). Convolutional spatial extraction, however, represents the non-Euclidean structure of a road network poorly Zhou et al. (2024; 2025). Graph Neural Networks (GNNs) address this. AGCRN Bai et al. (2020) in particular couples a GCN-LSTM with adaptive adjacency matrices learned end-to-end, while STWave Fang et al. (2023) decouples traffic data into stationary and non-stationary components and models each with temporal and spatial attention.

Most of these models assume centralized training, which conflicts with the privacy constraints that regulations such as the GDPR and CCPA now impose on traffic data. Federated variants therefore have begun to appear: Shang et al. Shang & Li (2025b;a) build horizontal and vertical FedAvg McMahan et al. (2017) schemes around a carefully designed traffic graph, Hu et al. Hu et al. (2024) add a MendGCN module in FedGCN to relate graph nodes to external factors, and Zhou et al. Zhou et al. (2024; 2025) propose FedTPS, which shares knowledge among clients through a traffic pattern repository. A common feature of all these methods is their reliance on manual feature design and hyper-parameter optimization tailored to particular clients or datasets, which hinders their efficient adaptation to new contexts.

A.2 Federated Learning

FL is now widely applied in privacy-sensitive and resource-limited scenarios Cai et al. (2025), including medical recognition Pfitzner et al. (2021), traffic prediction Shang et al. (2025), and autonomous driving Kou et al. (2025). Standard methods such as FedAvg nevertheless assume IID client data, which rarely holds in practice. Among the model-based remedies for non-IID data surveyed in Pei et al. (2024), PFL is the most widely adopted. Clients share only part of the model, leveraging common knowledge while retaining personalized parameters or structures for their local distribution.

Starting with FedPer Arivazhagan et al. (2019), clients share upstream layers for robust feature extraction and keep personalized downstream layers, a paradigm also well studied in transfer learning Zhuang et al. (2020). Later work adds clustering Chang et al. (2025), meta-learning Fallah et al. (2020), and hyper-networks Scott et al. (2024). PFL can also accommodate clients with different model structures by aggregating common sub-networks, which suits clients with different computational budgets. Adapting these methods to traffic prediction, however, still runs into the obstacles described above.

The recent federated traffic predictors differ from ours in instructive ways. FedGTP Yang et al. (2024) has the server hold a global graph while each client stores only the embeddings of its observed nodes, recovering inter-client spatial dependency but requiring a shared node index, which fails when agencies cover disjoint road networks. pFedCTP Zhang et al. (2024b) shares a space-independent traffic-pattern module and interpolates between local and global parameters per layer. FedHINT Zhou et al. (2026) shares a global component and averages clients by node count rather than sample count. FedAPT Su et al. (2024) is closest in mechanism, freezing the backbone while learning a meta prompt with an adaptive network and per-client keys, but its prompt is personalized to a client's domain, whereas our useful variation is between input windows within one client. That is why our prompt is regenerated per input, and why shuffling it across windows is the most damaging corruption in Appendix D. AT-SPNet Wang et al. (2026) also splits a spatio-temporal model into a shared temporal branch and a private spatial branch, the decomposition closest to ours, but the split is fixed by the architecture and the shared branch extracts temporal features rather than generating per-input conditioning signals. FedCTPM Wang et al. (2025) shares a common pattern module and interpolates between local and global parameters, and NeighborFL Chen et al. (2026a) individuals each client by restricting aggregation to neighbors with an error-driven weighting rule. Both change

who aggregates with whom rather than what is communicated. A related family replaces uniform averaging with weights derived from a quality signal, such as client reputation Dai et al. (2024) or traffic-domain statistics Kaur et al. (2025). These improve aggregation on top of parameter sharing, so they inherit the node-axis incompatibility it faces on traffic graphs and leave the object being shared unchanged. None of them generates a prompt from the client's own data through a shared, node-axis-free generator.

A.3 Prompt Learning

The success of Large Language Models (LLMs) Chen et al. (2026b) has brought related techniques such as Masked Language Model pre-training Zhao et al. (2025), Retrieval-Augmented Generation Han et al. (2025), and LLMs themselves Jin et al. (2024) into time series analysis. Prompt learning stands out for its efficiency: prompts directly influence decoding, a principle established since T5 Raffel et al. (2020), and prompt tuning adapts models cheaply through learnable tokens Li et al. (2025).

In time series analysis, most work designs prompts to adapt pre-trained LLMs or their architectures to a task Cao et al. (2024); Xue & Salim (2023). For instance, Huang et al. (2025); Zhou et al. (2024; 2025) use learnable prototypes (or patterns) to guide decoding, but the number of prototypes strongly affects performance and the optimal quantity varies across datasets Zhou et al. (2024; 2025). We instead generate the prompt with a neural network, which removes the prototype count as a design choice altogether. Graph-based prompt learning Li et al. (2024) focuses on spatial prompt representations, whereas we emphasize the temporal dimension.

Federated prompt learning is the closest setting to ours. pFedPG Yang et al. (2023) generates a client-specific prompt through a hyper-network conditioned on client identity, so the prompt is a function of which client is speaking. pFedMoAP Luo et al. (2025) treats each local prompt as an expert, letting a client retrieve several non-local experts from a server pool and weight them through a locally trained gate, so personalization again enters through the gate. Both are effective and cheap, and the second suits vision-language models where the prompt substitutes for a fine-tuned backbone. The difference is which axis is personalized. Our clients are already distinguished by their data and private predictors, so the useful variation lies between input windows within a client rather than between clients, and the prompt is regenerated per window rather than looked up per client. That places it on a per-node, per-window axis, which is why the shared generator maps a feature axis and no node-indexed tensor can enter the shared block.

B Algorithm

The detailed process of our AutoFed is illustrated in Algorithm 1.

C Training Cost and Complexity Analysis

We analyze the training efficiency of AutoFed from two complementary angles: an empirical comparison of the per-round cost against representative baselines, and an asymptotic analysis that explains the observed trends.

C.1 Empirical Training Costs

Using scenario S1 of the TDP task as a running example, we compare the convergence speed, computational cost, and communication cost during training. Fig. 2 shows how the validation metrics evolve as training proceeds, where AutoFed converges faster than the baselines. Table 3 reports the detailed per-round costs. AutoFed and FedTPS have the lowest communication costs but the highest computation times, since both generate a prompt at each round. Notably, AutoFed reaches a communication cost close to that of FedTPS while achieving markedly better accuracy across tasks (Tables 1 and 2).

Algorithm 1 Training Process of AutoFed

Require: Number of clients $\mathcal{N}$, local datasets $\{D_i\}_{i=1}^{\mathcal{N}}$, shared model parameters Θ^0, personal model parameters $\{\theta_i^0\}_{i=1}^{\mathcal{N}}$, communication rounds M, local epochs E
Ensure: Personalized models $\{\theta_i^M, \Theta^M\}_{i=1}^{\mathcal{N}}$
Server executes:
Initialize shared parameters Θ^0
Broadcast Θ^0 to all clients
for each communication round $m = 0, 1, \ldots, M-1$ **do**
  Randomly select a subset of clients $S_m \subseteq \{1, \ldots, \mathcal{N}\}$
  for each client $i \in S_m$ **in parallel do**
    $\theta_i^{m+1}, \Delta\Theta_i^m \leftarrow \text{ClientUpdate}(i, \theta_i^m, \Theta^m)$
  end for
  Aggregate shared parameters: $\Theta^{m+1} \leftarrow \Theta^m + \frac{1}{|S_m|} \sum_{i \in S_m} \Delta\Theta_i^m$
  Broadcast Θ^{m+1} to all clients
end for

ClientUpdate(client i, personal parameters θ_i, shared parameters Θ):
Initialize local shared parameters: $\Theta_i \leftarrow \Theta$
for each local epoch $e = 1$ to E **do**
  for each batch $(X_b, Y_b) \in D_i$ **do**
    Forward Pass:
    Compute compressed features: $p = \text{AE-enc}(X_b)$, $\hat{X}_b = \text{AE-dec}(p)$
    Extract local features: $p_l = \text{Encoder}(p)$
    Generate global prompt: $p_g = \text{Adapter}(p_l)$
    Generate predictions: $\hat{Y}_b = \text{PP}(X_b, p_g)$
    Loss Computation:
    Compute AE loss: $L^{ae} = \text{MAE}(X_b, \hat{X}_b)$
    Compute prediction loss: $L^{pre} = \text{MAE}(Y_b, \hat{Y}_b)$
    Compute adaptive weight (stop-gradient): $\alpha = L^{ae}/L^{pre}$
    Compute total loss: $L = L^{pre} + \alpha \cdot L^{ae}$
    Backward Pass: the gradient is $\nabla L^{pre} + \alpha \nabla L^{ae}$; α carries no gradient
  end for
end for
Compute parameter update: $\Delta\Theta_i = \Theta_i - \Theta$
Return θ_i, $\Delta\Theta_i$

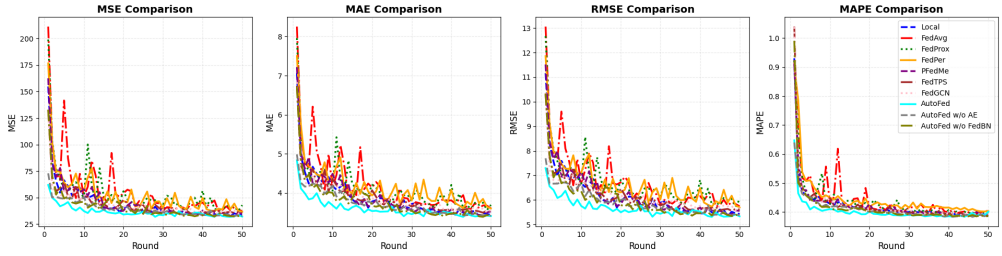


Figure 2: Training Process: This figure shows the change of different metrics in valid set during training.

Table 3: Training Costs Comparison

Method	Computation (time/round) (s)	Communication (parameter/round)
FedAvg McMahan et al. (2017)	**76.2**	5.5M
FedProx Li et al. (2020)	77.2	5.5M
FedPer Arivazhagan et al. (2019)	81.6	2.9M
pFedMe T Dinh et al. (2020)	90.0	5.5M
FedTPS Zhou et al. (2024; 2025)	106.4	**0.1K**
FedGCN Hu et al. (2024)	<u>76.8</u>	12.9M
AutoFed	104.6	<u>175.0K</u>

C.2 Computation and Communication Complexity

We now complement Table 3 with an asymptotic analysis that explains its two trends: AutoFed has a computation cost of the same order as a single AGCRN backbone, while its communication cost is independent of the graph size. We use the notation of Section 2: n is the number of nodes of a client, K the input feature dimension, k the hidden and prompt dimension, $\mathcal{T}$ and $\mathscr{T}$ the history and horizon lengths, L the number of AGCRN layers, and E the number of local epochs per round. Throughout, $n_i := |V_i|$ is the node count of client i and S_i its number of training samples; we write n for a generic client when the index is not needed.

Computation. The dominant cost of the backbone is the adaptive graph convolution inside AGCRN. Forming the adaptive adjacency $\tilde{A}$ from the learnable node embedding costs $O(n^2k)$, and each graph convolution over the dense $n \times n$ adjacency costs $O(n^2k + nk^2)$, where the $O(n^2k)$ term dominates because $n \gg k$ in traffic graphs. A GRU cell applies three such convolutions, so one AGCRN pass over a length-T sequence with L layers costs $O(TLn^2k)$. AutoFed performs three such passes per sample, namely the PP encoder at $O(\mathcal{T}Ln^2k)$, the PP decoder at $O(\mathscr{T}Ln^2k)$, and the FR graph encoder at $O(\mathcal{T}Ln^2k)$. The remaining FR modules act on already compressed features and are cheaper: the auto-encoder applies a per-node feature map at a cost of $O(\mathcal{T}nKk)$, and the client-aligned adapter applies a shared per-node linear map to the single matrix $p_l \in \mathbb{R}^{n\times k}$ produced by the FR encoder, at a cost of $O(nk^2)$. The per-sample forward cost is therefore

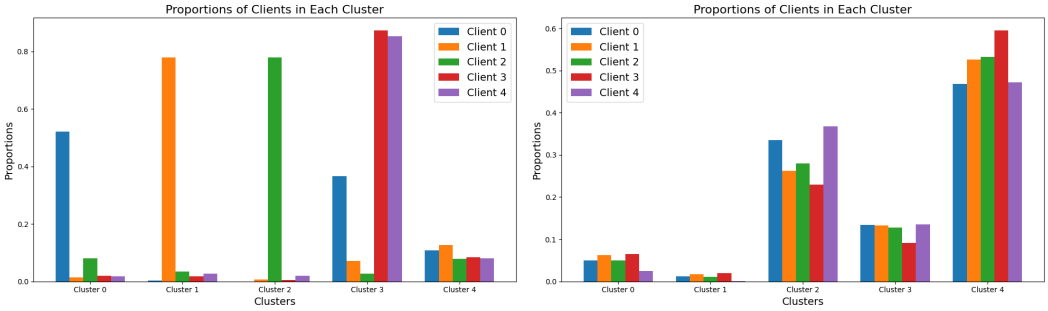
$$O\big((2\mathcal{T} + \mathscr{T})\, L\, n^2k \;+\; \mathcal{T}nKk \;+\; nk^2\big), \tag{8}$$

and one training round over client i with E local epochs multiplies this by ES_i. The adapter term $O(nk^2)$ carries no sequence length and is linear rather than quadratic in n, so it is dominated by the graph-convolution term whenever $k \ll \mathcal{T}Ln$, which holds throughout our setting. A plain AGCRN predictor, as used by FedAvg-style baselines, already costs $O((\mathcal{T} + \mathscr{T})Ln^2k)$, so the modules AutoFed adds, namely the FR encoder, the auto-encoder, and the adapter, leave the asymptotic order unchanged and raise the cost only by a constant factor under this accounting. The n^2 dependence stems from the dense adaptive adjacency and is inherent to AGCRN rather than introduced by AutoFed. Table 3 reflects this, with AutoFed's per-round time of 104.6s a small constant factor above FedAvg at 76.2s.

Communication. Only the shared block Θ is exchanged each round. The following observation makes precise why this cost does not grow with the graph.

Proposition 1 (Node-count-independent communication)**.** *Per round, each client transmits $O(|\Theta|)$ parameters, where the shared block satisfies $|\Theta| = O(k^2 + Kk)$. In particular it is independent of the node count n, the sequence lengths $\mathcal{T}$ and $\mathscr{T}$, and the size of the personalized predictor.*

Proof. The shared block Θ consists of the auto-encoder and the shared linear layers of the adapter (Section 2); the AGCRN predictor, the personalized FR encoder, and all batch normalization layers are private and are never communicated. Because Θ is shared across clients whose graphs have different node counts, its layers cannot act on the node axis and must act on the per-node feature dimension instead: the auto-encoder maps the feature dimension K to k and back using $O(Kk)$



(a) Client Distribution under Local Representation (b) Client Distribution under Global Representation

Figure 3: Comparison Between Local Representation and Global Representation

parameters, and the shared adapter linear maps k to k using $O(k^2)$ parameters. Neither depends on n or on the sequence lengths, which enter only as the sizes of the axes over which the layers are broadcast. Summing gives $|\Theta| = O(k^2 + Kk)$. Each round uploads the update $\Delta\Theta_i$ and downloads Θ, both of this size. □

By contrast, aggregating a full model as in FedAvg or FedGCN transmits the prediction backbone as well, and that backbone is what accounts for the gap in Table 3: AutoFed communicates 175.0K parameters per round, against 5.5M for FedAvg and 12.9M for FedGCN, which is roughly 31 and 74 times fewer parameters respectively. The comparison is between the shared block, which contains no node-dependent tensor at all, and a full backbone whose tensors are dominated by the recurrent and graph-convolution weights.

We are explicit about what drives the ratio. Under the convention of Section 3 the node-axis tensors of the baselines stay local, so the gap does not come from an aggregated node embedding, which is in any case small: 19.6K parameters at PEMS04 against 5.5M in the backbone, under 0.4%. What we claim is the weaker statement that the shared block has a size independent of the node count whereas a fully shared backbone does not. AutoFed therefore trades a modest constant-factor rise in local computation for a communication cost that no longer grows with the graph, which is the regime that matters in cross-silo traffic prediction, where bandwidth rather than local compute is the binding constraint.

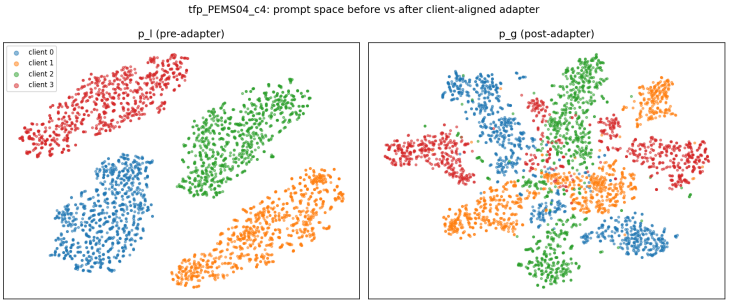
D Representation Visualization

We use "causal" throughout in the interventional sense of a controlled input perturbation rather than in the sense of a causal identification from observational data.

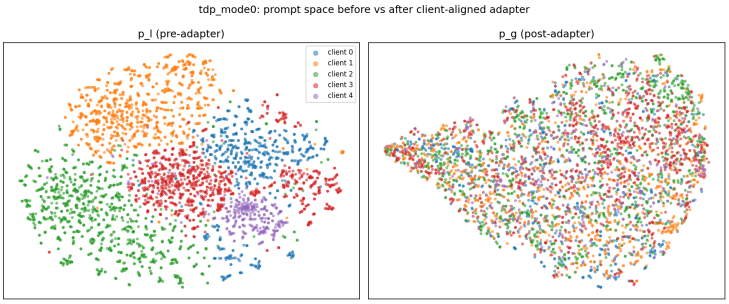
In this section we analyze the prompt produced by the federated representor in order to understand what the client-aligned adapter learns and why the prompt helps prediction. We organize the analysis around three questions: how the adapter reshapes the prompt space across clients, whether the prompt is causally relied upon by the decoder, and what traffic information it encodes. Throughout, we distinguish the local prompt p_l, the output of the low-frequency encoder before the client-aligned adapter, from the global prompt p_g, the output of the adapter that is fed to the decoder as the prefix token. Unless stated otherwise, we capture p_l and p_g with forward hooks on held-out test windows so the trained models are left untouched, and every row of p_l or p_g corresponds to one node in one input window. We report results on both the TDP task (scenarios S0 and S1) and the TFP task (PEMS04 and PEMS08 with four clients), and use scenario S1 of the TDP task as a running example when a single setting is shown.

D.1 How the Adapter Reshapes the Prompt Space

We first ask how the client-aligned adapter reorganizes the prompt across clients, using two complementary views of the same phenomenon. The first is a clustering view (Fig. 3): we combine all p_l



(a) TFP, PEMS04 with four clients



(b) TDP, scenario S0

Figure 4: Two-dimensional t-SNE embeddings of the prompt before (p_l, left) and after (p_g, right) the client-aligned adapter, colored by client.

and p_g across clients, apply KMeans clustering initialized at the per-client centers, and measure the proportion of each client within every cluster. Under the local prompt, clients differ substantially and their features concentrate in different clusters, whereas under the global prompt each cluster becomes a relatively uniform mixture of clients, indicating that the adapter suppresses client-specific characteristics and surfaces globally shared traffic patterns. The second is a projection view (Fig. 4): we embed p_l and p_g into two dimensions with t-SNE (PCA to 50 dimensions beforehand) and color each point by its client of origin. The two views agree. Before the adapter, the local prompts of different clients occupy clearly separated regions, reflecting the heterogeneous statistics of each region; after the adapter, the clusters are pulled together and interleave heavily, so a point's client can no longer be read off from its position.

These measures have a defined reading. KMeans initialized at the per-client centers starts from one seed per client, so the cluster purity in Fig. 3 captures how the initialization interacts with client mixing alongside the mixing itself, and purity also varies with the cluster count and the unequal client sizes. t-SNE separation depends on the perplexity and on the initialization, which makes the two-dimensional view a consistency check whose role is to corroborate the quantitative measures below. The alignment numbers are as follows. The silhouette score, which uses the client label as the cluster assignment and for which a higher value means cleaner per-client separation, drops consistently from p_l to p_g on every setting, from 0.162 to 0.003 on PEMS04 and from 0.098 to 0.013 on PEMS08. On the two TDP scenarios the score is already negative for p_l (about -0.02 to -0.05) and becomes more negative for p_g; here the clients overlap in prompt space even before the adapter because of how the demand data is partitioned, so for TDP the silhouette is a weak indicator and we rely instead on the geometric measure below. To check that the relocation preserves the internal geometry of the representation rather than destroying it, we compute the linear centered kernel alignment (CKA) between p_l and p_g per client. On the TFP task the CKA is high (0.92 to 0.96), showing that the adapter mainly repositions each client as a whole while keeping the relative structure of its samples intact; on the TDP task it is lower and more variable (0.31 to 0.76), indicating a stronger rewrite for the more heterogeneous demand data. In both cases the adapter projects heterogeneous local representations into a common coordinate system, which is the mechanism the alignment analysis predicts.

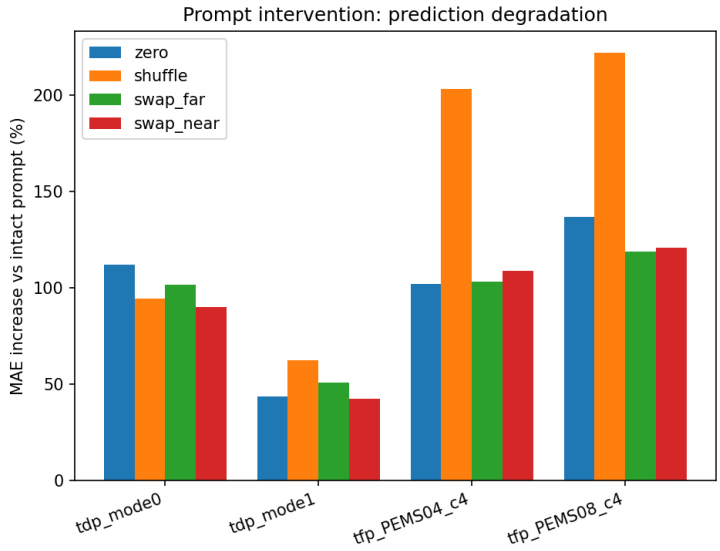


Figure 5: Relative increase in test MAE when the prompt p_g is corrupted at inference, averaged over clients.

Setting	`zero`	`shuffle`	`swap-far`	`swap-near`
TDP S0	+112	+95	+102	+90
TDP S1	+43	+62	+51	+42
TFP PEMS04	+102	+203	+103	+109
TFP PEMS08	+137	+222	+119	+121

Table 4: Relative test MAE increase (%) under prompt intervention, averaged over clients. Errors are measured on each task's native scale through the per-client scaler, so magnitudes are comparable within a task but not across tasks.

D.2 Is the Prompt Causally Used?

A visualization on its own cannot tell us whether the prompt is actually relied upon by the model. We therefore run a causal intervention: at inference we overwrite the adapter output p_g and measure how much the test MAE rises relative to the intact model (Fig. 5, Table 4). We consider four corruptions. `zero` replaces the prompt with zeros, ablating it entirely; `shuffle` permutes the prompt rows within a batch, keeping the value distribution but breaking the correspondence between a prompt and its own window and node; `swap-far` and `swap-near` replace every row with the mean prompt of the most and least distant donor client, where distance is the Euclidean distance between per-client mean prompts. Across all settings every corruption inflates the error substantially, by 43% to more than 200%, so the decoder genuinely consumes the prompt rather than treating it as a passive side input. The pattern across corruptions is informative. `shuffle` is consistently the most harmful, roughly tripling the error on TFP, but it breaks the match between a prompt and its own window and node at the same time, so on its own it shows only that the prompt is input-specific. The `swap` variants separate the two: they substitute a donor-client mean of the correct shape, which preserves the node correspondence and destroys only the per-input variation, and both are as damaging as zeroing the prompt. Replacing the prompt with a static donor vector therefore hurts no more than ablating it, which indicates that a frozen city embedding is nearly worthless and that the value of the prompt lies in its input-conditioned variation. This supports generating the prompt dynamically from each input rather than storing one vector per client.

Distance between clients in prompt space does not predict transferability: `swap-far` and `swap-near` cause almost the same degradation, and the nearer donor is sometimes worse, which

	Level (R^2)		Volatility (R^2)	
Setting	probe	control	probe	control
TDP S0	0.10	0.00	0.13	0.00
TDP S1	0.14	0.00	0.20	0.00
TFP PEMS04	0.87	0.01	0.51	0.00
TFP PEMS08	0.79	0.00	0.35	0.00

Table 5: Linear probing of p_g. We report the test R^2 of a ridge regressor predicting the within-window mean (level) and standard deviation (volatility) of the primary channel, against a shuffled-label control.

reinforces the point above. What matters is not which city a prompt is borrowed from but whether it is dynamically matched to the current input, so a static prompt is damaging regardless of its source.

D.3 What Does the Prompt Encode?

Finally we ask what traffic content the prompt carries. We fit a ridge regressor from p_g to two simple window statistics of the primary channel, its mean, which reflects the current traffic level, and its standard deviation, which reflects short-term volatility, and report the test R^2 against a control trained on shuffled labels (Table 5). If the prompt encodes these quantities linearly, the probe should score well above the control, whose R^2 is near zero by construction. On the TFP task the current traffic level is decoded strongly (R^2 of 0.87 and 0.79) and volatility moderately (0.51 and 0.35), so the flow prompt clearly represents how busy and how bursty the current window is. On the TDP task the same probe is weak (R^2 of 0.10 to 0.20) but still far above the control, which is expected because the demand input has nine channels and the mean of a single channel is not its dominant signal. Overall the prompt is not an opaque code: it linearly exposes interpretable traffic quantities, most strongly on the flow task where such statistics are the primary signal.

Taken together, the three analyses give a concrete answer to what the prompt is and why it helps. The adapter aligns heterogeneous local prompts into a shared space (Figs. 3 and 4), the resulting prompt is relied upon by the decoder and must be matched to each specific input rather than fixed per client (Table 4), and it encodes interpretable traffic level and volatility (Table 5). The evidence is strongest and most consistent on the TFP task, where flow is a low-dimensional, strongly periodic signal: alignment is clean, the prompt is heavily used, and its content is decoded with high fidelity. On the TDP task the same trends hold in direction but are weaker, because travel demand is a higher-dimensional and more diffuse signal whose informative structure is not captured by a single-channel statistic; there the story rests mainly on the alignment and intervention evidence rather than on the linear probe. In both regimes the prompt turns out to be input-conditioned, causally used, and partially interpretable rather than an opaque code. The interventions perturb the prompt at inference and measure the change in test error, which shows that the decoder reads the prompt but does not identify what it carries.

E Theoretical and Interpretive Analysis

This appendix collects the theoretical analysis of AutoFed in two parts. The first, "Convergence and Stability of Federated Training", is rigorous for a fixed-weight surrogate: it holds the auxiliary weight fixed at whichever value the adaptive rule produces, assumes full participation with one local step per round, and analyzes the resulting objective. Fixing the weight isolates the optimization structure of the shared and private parameterization, which is what the analysis is meant to expose. The practical algorithm uses the adaptive rule of Eq. equation 6, and because that weight is recomputed from the current losses it varies from step to step, so the training follows a sequence of objectives rather than one fixed objective and the proposition is not a formal guarantee for the adaptive scheme. Within the surrogate the iterates cannot diverge for a suitable step size and converge at the usual nonconvex rate, while the general regime of many local steps and partial participation carries the usual drift term rather than a new guarantee. The analysis treats Algorithm 1 as partial model personalization. It also shows that the optimal value of the personalized objective is no larger than that of a fully shared model, which answers whether prompt based personalization can degrade to plain federated averaging, and explains why a single outlier client cannot destabilize training.

The second, "The Prompt as a Global Conditioning Signal" and the subsections that follow it, is deliberately interpretive rather than fully rigorous. It gives a probabilistic reading of the prompt as a global conditioning signal, a domain adaptation heuristic for why aligning clients in the prompt space aids generalization, and a low rank view of prompt conditioning. These arguments build intuition and are consistent with our experiments, but they rely on informal steps that we flag explicitly. Their rigorous versions, in particular a quantitative contraction factor for the cross client divergence, are left for future work.

Throughout, $z = (\Theta, \theta_1, \ldots, \theta_{\mathcal{N}})$ denotes the full parameter vector, where Θ collects the shared parameters (the auto-encoder and the shared linear layers of the adapter) and θ_i collects the parameters kept private to client i (its AGCRN predictor, personalized encoder, and batch normalization layers). We write the objective of Eq. equation 1 as

$$F(z) = \sum_{i=1}^{\mathcal{N}} w_i F_i(\Theta, \theta_i), \qquad w_i > 0, \quad \sum_{i=1}^{\mathcal{N}} w_i = 1, \tag{9}$$

where $F_i(\Theta, \theta_i) = \mathbb{E}_{\xi \sim \mathcal{D}_i}[\ell_i(\Theta, \theta_i; \xi)]$ is the expected local loss of client i and $\ell_i \geq 0$ is the per sample MAE loss of Eq. equation 5. The analysis is stated for arbitrary fixed weights, so it covers both instantiations used in the paper: Eq. equation 1, where w_i is the node count share $|V_i| / \sum_j |V_j|$, and Algorithm 1, whose aggregation is uniform with $w_i = 1/\mathcal{N}$. Only $w_{\min}$ differs between the two.

E.1 Rigorous Part: Convergence and Stability of Federated Training

Because each private block θ_i appears in only the i-th summand of Eq. equation 9, the block gradients are

$$\nabla_\Theta F = \sum_{i=1}^{\mathcal{N}} w_i \nabla_\Theta F_i, \qquad \nabla_{\theta_i} F = w_i \nabla_{\theta_i} F_i. \tag{10}$$

This is exactly the structure of partial model personalization Pillutla et al. (2022): a shared block Θ that is aggregated across clients and private blocks θ_i that are updated only locally. AutoFed's training in Algorithm 1 is the simultaneous variant, in which both blocks are updated in the same local pass and only Θ is communicated. We make the following standard assumptions.

Assumption 1 (Smoothness)**.** Each F_i is continuously differentiable and L-smooth in z, that is $\|\nabla F_i(z) - \nabla F_i(z')\| \leq L\|z - z'\|$ for all z, z'.

Assumption 2 (Lower bound)**.** F is bounded below by some $F^\star > -\infty$. This holds here because $\ell_i \geq 0$ implies $F \geq 0$.

Assumption 3 (Unbiased gradients with bounded variance)**.** Each client's stochastic block gradient $g_{i,\bullet}$, computed on a sampled mini batch, is unbiased with $\mathbb{E}[g_{i,\bullet} \mid z] = \nabla_\bullet F_i(z)$ and $\mathbb{E}\|g_{i,\bullet} - \nabla_\bullet F_i\|^2 \leq \sigma^2$, and the noises of different clients are mutually independent.

Assumption 4 (Smoothness and bounded variance of the auxiliary objective)**.** F^{ae} is continuously differentiable and L-smooth in z with the same constant as F_i in Assumption 1, so that F^α is L_{eff}-smooth, and its stochastic gradient has variance at most σ^2 on the same mini batch, with noise independent of that of ∇F_i.

The loss is MAE, which is non-differentiable at zero residual, and the adaptive adjacency is built through a ReLU-like nonlinearity, which is likewise not continuously differentiable. The analysis therefore proceeds on a smoothed surrogate, for instance a Huber loss and a smooth analogue of the nonlinearity, which is the standard treatment for non-smooth objectives and preserves the qualitative reading of the bound.

We first analyze the transparent regime of full participation with one local step per round, which is the regime Proposition 2 covers; the general case is discussed afterward and is where the guarantee is approximate. Throughout, the auxiliary weight is held fixed at a constant α, and the objective the step descends is the surrogate

$$F^\alpha(z) := F(z) + \alpha F^{ae}(z), \qquad F^{ae}(z) := \sum_{i=1}^{\mathcal{N}} w_i \,\mathrm{MAE}(X_i, \hat{X}_i(z)), \tag{11}$$

which is the fixed-weight version of the quantity Eq. equation 7 is a gradient of. The adaptive rule of Eq. equation 6 is what the implementation uses, and it is not covered by the results below. In this regime one round performs, with step size $\eta > 0$,

$$\Theta^{m+1} = \Theta^m - \eta \sum_i w_i \, g^m_{i,\Theta}, \qquad \theta^{m+1}_i = \theta^m_i - \eta \, g^m_{i,\theta}, \tag{12}$$

where the shared update aggregates client contributions with the objective weights w_i. Uniform averaging as written in Algorithm 1 is the special case $w_i = 1/\mathcal{N}$, for which every result below holds verbatim with the uniformly weighted objective. Let h^m denote the stacked update direction in Eq. equation 12. Using Eq. equation 10, a short computation gives $\mathbb{E}[h^m \mid z^m] = P\,\nabla F^\alpha(z^m)$, where $P = \mathrm{blockdiag}(I, \, w_1^{-1} I, \ldots, w_{\mathcal{N}}^{-1} I)$ is a fixed positive definite preconditioner with $\lambda_{\min}(P) = 1$ and $\lambda_{\max}(P) = 1/w_{\min}$, writing $w_{\min} := \min_i w_i$.

Proposition 2 (Descent and convergence)**.** *Under Assumptions 1 to 3, if the step size satisfies $\eta \le w_{\min}/(L(1+\alpha_{\max}))$ with $\alpha_{\max}$ the value of the adaptive weight held fixed by the analysis, then the iterates of Eq. equation 12 obey, for every round m,*

$$\mathbb{E}[F^\alpha(z^{m+1}) \mid z^m] \le F^\alpha(z^m) - \frac{\eta}{2}\|\nabla F^\alpha(z^m)\|^2 + \frac{L_{\mathrm{eff}}\eta^2}{2} V, \tag{13}$$

with $L_{\mathrm{eff}} := (1+\alpha_{\max})L$ and $V := (\mathcal{N}+1)(1+\alpha_{\max})^2\sigma^2$. Averaging over M rounds,

$$\frac{1}{M}\sum_{m=0}^{M-1} \mathbb{E}\|\nabla F^\alpha(z^m)\|^2 \le \frac{2\big(F^\alpha(z^0) - F^{\alpha\star}\big)}{\eta M} + L_{\mathrm{eff}}\eta V, \tag{14}$$

where $F^{\alpha\star} := \min_z F^\alpha(z)$. In particular, the choice $\eta = \min\{\, w_{\min}/L_{\mathrm{eff}}, \, \sqrt{2(F^\alpha(z^0) - F^{\alpha\star})/(L_{\mathrm{eff}} V M)}\,\}$ gives $\min_{m<M} \mathbb{E}\|\nabla F^\alpha(z^m)\|^2 = O(1/\sqrt{M})$, and in the deterministic case $\sigma = 0$ the sum over rounds of the squared gradient norm is bounded, so the average squared gradient norm over M rounds is $O(1/M)$ as well. Both statements bound the average squared gradient norm rather than the distance of the iterates to a stationary point, which is the standard guarantee at this level of generality and is what the small-σ experiments are consistent with.

Proof. Fix a round m and condition on z^m. Each F_i is L-smooth in z by Assumption 1, so their convex combination F is also L-smooth, and the descent lemma gives

$$F^\alpha(z^{m+1}) \le F^\alpha(z^m) + \langle \nabla F^\alpha(z^m), z^{m+1} - z^m \rangle + \tfrac{L_{\mathrm{eff}}}{2}\|z^{m+1} - z^m\|^2.$$

The step takes $z^{m+1} - z^m = -\eta h^m$ with $\mathbb{E}[h^m \mid z^m] = P\nabla F^\alpha(z^m)$ by Eq. equation 12. Smoothness of F^{ae} is assumed rather than derived, and it is Assumption 4, not Assumption 1 alone, that supplies it, since Assumption 1 constrains F_i only and imposes nothing on the reconstruction term. With both terms L-smooth, F^α is L_{eff}-smooth as a function of z and the descent lemma applies with that constant. The stochastic direction estimates ∇F^α with the variance of Eq. equation 11: the two blocks of the estimate share a mini batch and are therefore correlated, so their variances add with a cross term bounded by $2\alpha_{\max}\sigma^2$ under Cauchy-Schwarz, giving $\mathbb{E}\|g^{\mathrm{eff}} - \nabla F^\alpha\|^2 \le (1+\alpha_{\max})^2\sigma^2$ for every block, which is the variance bound assumed in Assumption 4. Hence

$$\mathbb{E}[F^\alpha(z^{m+1}) \mid z^m] \le F^\alpha(z^m) - \eta\langle \nabla F^\alpha, P\nabla F^\alpha\rangle + \tfrac{L_{\mathrm{eff}}\eta^2}{2}\,\mathbb{E}\|h^m\|^2.$$

Because $\mathbb{E}[h^m] = P\nabla F^\alpha$, we have $\mathbb{E}\|h^m\|^2 = \|P\nabla F^\alpha\|^2 + \mathbb{E}\|h^m - P\nabla F^\alpha\|^2$. Under Assumption 3 and the independence of client noise, writing $\bar\sigma^2 := (1+\alpha_{\max})^2\sigma^2$ for the inflated per block variance of the preceding paragraph, the shared block variance is $\mathbb{E}\|\sum_i w_i(g^{\mathrm{eff}}_{i,\Theta} - \nabla_\Theta F_i)\|^2 = \sum_i w_i^2\,\mathbb{E}\|g^{\mathrm{eff}}_{i,\Theta} - \nabla_\Theta F_i\|^2 \le \bar\sigma^2 \sum_i w_i^2 \le \bar\sigma^2$, using $\sum_i w_i^2 \le \max_i w_i \le 1$, while the $\mathcal{N}$ private blocks each contribute at most $\bar\sigma^2$, so $\mathbb{E}\|h^m - P\nabla F^\alpha\|^2 \le (\mathcal{N}+1)\bar\sigma^2 = V$. For the two deterministic terms, expand $\langle \nabla F^\alpha, P\nabla F^\alpha\rangle = \|\nabla_\Theta F^\alpha\|^2 + \sum_i w_i^{-1}\|\nabla_{\theta_i} F^\alpha\|^2$ and $\|P\nabla F^\alpha\|^2 = \|\nabla_\Theta F^\alpha\|^2 + \sum_i w_i^{-2}\|\nabla_{\theta_i} F^\alpha\|^2$. Since $w_i^{-1} \le w_{\min}^{-1}$ for every i, each coefficient obeys $w_i^{-2} \le w_{\min}^{-1} w_i^{-1}$, hence $\|P\nabla F^\alpha\|^2 \le w_{\min}^{-1}\langle \nabla F^\alpha, P\nabla F^\alpha\rangle$. Substituting,

$$\mathbb{E}[F^\alpha(z^{m+1}) \mid z^m] \le F^\alpha(z^m) - \eta\Big(1 - \tfrac{L_{\mathrm{eff}}\eta}{2w_{\min}}\Big)\langle \nabla F^\alpha, P\nabla F^\alpha\rangle + \tfrac{L_{\mathrm{eff}}\eta^2}{2} V.$$

The condition $\eta \le w_{\min}/(L(1+\alpha_{\max})) = w_{\min}/L_{\text{eff}}$ makes $1-\frac{L_{\text{eff}}\eta}{2w_{\min}} \ge \frac{1}{2}$, and $\langle \nabla F^\alpha, P\nabla F^\alpha\rangle \ge \lambda_{\min}(P)\|\nabla F^\alpha\|^2 = \|\nabla F^\alpha\|^2$, which gives Eq. equation 13. Two consequences are worth stating. The rate in Eq. equation 14 is a rate for F^α, and F^α and F share a stationary point only when $\alpha = 0$, so the bound should be read as convergence of the surrogate objective. The auxiliary term is bounded below by zero and its gradient vanishes at any $\hat{x} = x$, so when the auto-encoder is at or near its optimum the second term can be dropped and the guarantee transfers to F with $\alpha_{\max}$ set to the residual weight rather than to 1. Rearranging Eq. equation 13, summing over $m = 0, \dots, M-1$, taking total expectation, and telescoping with Assumption 2 gives $\frac{\eta}{2}\sum_m \mathbb{E}\|\nabla F^\alpha(z^m)\|^2 \le F^\alpha(z^0) - F^{\alpha\star} + \frac{L_{\text{eff}}\eta^2}{2}VM$; dividing by $\eta M/2$ yields Eq. equation 14. The stated step size balances the two terms of Eq. equation 14 and gives the $O(1/\sqrt{M})$ bound; when $\sigma = 0$ we have $V = 0$, so Eq. equation 13 is a monotone decrease and $\sum_m \|\nabla F^\alpha(z^m)\|^2 \le 2(F^\alpha(z^0) - F^{\alpha\star})/\eta$, that is $O(1/M)$. □

Remark 1 (Effect of the adaptive weight)**.** *For the fixed weight surrogate, Proposition 2 shows that carrying an auxiliary objective costs nothing beyond a factor $1 + \alpha_{\max} = 2$ in the admissible step size, and this holds because α is stop-gradient and therefore does not differentiate into the prediction term. Had α been differentiated through, the gradient of L_i^{pre} would carry the factor $1 - (L_i^{ae}/L_i^{pre})^2$ noted in Section 2, which is negative whenever the reconstruction error exceeds the prediction error; that differentiated direction is not the gradient of any fixed objective, so it is not a descent direction for a fixed objective and the guarantees above would not apply to it. Because the rule sets $\alpha = L_i^{ae}/L_i^{pre}$, a small L_i^{pre} late in training can push α upward, and $L_{\text{eff}} = (1+\alpha_{\max})L$ grows with it. The step size is therefore chosen against the weight actually in force in a round, which keeps the descent condition satisfied round by round.*

Proposition 2 illustrates for any step size below $w_{\min}/L_{\text{eff}}$, the expected surrogate objective decreases every round up to a noise floor of order η^2, so within this regime the training cannot diverge, and the averaged squared gradient vanishes at the standard nonconvex rate. The step size ceiling carries two intuitive readings. The factor $w_{\min}$ makes precise the fact that severely imbalanced clients require a smaller learning rate for the same guarantee, and the factor $1 + \alpha_{\max}$ says that placing weight α on the auxiliary reconstruction objective costs that factor in the admissible step size. That the weight enters only through this multiplicative constant, and not through the rate, is a consequence of treating α as a stop-gradient schedule rather than a differentiated term of the objective.

Corollary 1 (The personalized optimum is no worse than the shared optimum)**.** *Let $F^\star_{\text{pers}} = \min_{\Theta,\{\theta_i\}} F(\Theta, \{\theta_i\})$ be the optimal value of the training objective of Eq. equation 9 when each client keeps its own predictor, and let $F^\star_{\text{shared}} = \min_{\Theta,\theta}\sum_i w_i F_i(\Theta, \theta)$ be the optimal value of the same objective when all clients are forced to share a single predictor, as in plain federated averaging. Then $F^\star_{\text{pers}} \le F^\star_{\text{shared}}$. The statement compares objective values, not algorithms and not test errors.*

Proof. The shared model is the restriction of Eq. equation 9 to the subspace $\{\theta_1 = \cdots = \theta_{\mathcal{N}}\}$. Minimizing over a superset can only lower the minimum, hence $F^\star_{\text{pers}} \le F^\star_{\text{shared}}$. □

Remark 2 (Degradation to FedAvg and outlier clients)**.** *Corollary 1 compares two optima rather than two algorithms or two test errors: the smallest attainable value of the personalized training objective cannot exceed the smallest attainable value of the shared objective, because the shared model is the special case $\theta_1 = \cdots = \theta_{\mathcal{N}}$ of the personalized formulation. The personalized hypothesis class therefore contains the fully shared model, and adding the prompt based components cannot raise the attainable training objective above the fully shared one. The corollary speaks to the attainable objective of the two formulations, and the comparison of the two updates on held out data is settled empirically in Section 3. In addition, a single client j whose distribution differs sharply from the rest enters the shared update in Eq. equation 12 only through the term $w_j\, g_{j,\Theta}$, whose norm is $O(w_j)$ under Assumption 1 on a bounded parameter domain. The client's idiosyncrasies are otherwise absorbed by its fully private block θ_j, namely its AGCRN predictor and batch normalization statistics, and never enter the private updates of other clients. An outlier therefore perturbs the shared prompt generator only in proportion to its aggregation weight and cannot by itself destabilize training.*

General case. The analysis above isolates the essential mechanism in the one local step, full participation setting. With $K > 1$ local steps and partial client participation, the same argument carries through up to an additional client drift term of order $O(\eta^2 K^2(\sigma^2 + \zeta^2))$, where ζ^2 bounds the cross client diversity of the shared block gradient, $\sum_i w_i\|\nabla_\Theta F_i - \nabla_\Theta F\|^2 \le \zeta^2$ evaluated at

the personalized parameters. This places the general regime within reach of the standard partial participation analysis Pillutla et al. (2022), which extends to this setting with an additional client drift term of this order, so the rate for the general regime takes the expected form below. A favorable and distinctive feature is that here ζ^2 is measured at the personalized points θ_i: because each client keeps a private predictor and private normalization, much of the heterogeneity that inflates ζ^2 in fully shared FL is absorbed locally, and the effective diversity seen by the shared prompt generator is smaller than in FedAvg.

E.2 Interpretive Part: The Prompt as a Global Conditioning Signal

We interpret the prompt matrix $p_g \in \mathbb{R}^{n\times k}$ generated by the Federated Representor (FR) as a compact, globally aligned representation that encodes shared spatiotemporal patterns across clients. For a given client i, the predictive distribution satisfies the following factorization by construction of the model:

$$P(\hat{y} \mid x) = P\big(\hat{y} \mid H_{\mathcal{T}}(x), p_g(x)\big), \tag{15}$$

where $H_{\mathcal{T}}(x) = \text{Encoder}(x; \mathbf{0})$ is the final encoder hidden state extracted from the raw client-specific input x, and $p_g(x) = \text{FR}(x)$ denotes the deterministic prompt generated by the Federated Representor. The decoder operates in an auto-regressive manner, conditioned on both $H_{\mathcal{T}}(x)$ and $p_g(x)$ (as the initial token).

This factorization explicitly separates two complementary roles: $H_{\mathcal{T}}(x)$ captures fine-grained, client-local spatiotemporal dynamics via the adaptive graph structure of AGCRN (its parameters are personalized per client), while $p_g(x)$ injects a globally shared task signal that summarizes recurring traffic patterns (e.g., daily/weekly rhythms) across heterogeneous regions. Thus, even though the raw data distributions $P_i(x)$ vary across clients, the shared prompt p_g provides a common prior that the personalized decoder can condition on, reducing the burden of learning cross-client patterns from scratch.

E.3 Generalization Advantage Under Client Heterogeneity

We analyze generalization using the domain-adaptation framework of Ben-David et al. Ben-David et al. (2006; 2010), extended to the multi-client federated setting. Let $\mathcal{D}_i$ denote the data distribution of client i over (x, y). Define the mixture source distribution as $\mathcal{D}_s = \frac{1}{\sum_i |V_i|}\sum_i |V_i|\mathcal{D}_i$, i.e., the weighted average of all client distributions. For a target client t with distribution $\mathcal{D}_t$, and a hypothesis h in a hypothesis class $\mathcal{H}$, the classic bound states:

$$\epsilon_t(h) \le \epsilon_s(h) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) + \lambda^*, \tag{16}$$

where $\epsilon_s(h)$ is the expected risk on the source mixture, $d_{\mathcal{H}\Delta\mathcal{H}}$ is the $\mathcal{H}\Delta\mathcal{H}$-divergence (a symmetric measure of distribution discrepancy), and λ^* is the error of the ideal joint hypothesis for the class $\mathcal{H}$. We treat this and the remainder of the subsection as an interpretive guide rather than a formal guarantee for our setting. The $\mathcal{H}\Delta\mathcal{H}$-divergence and the bound in Eq. equation 16 are stated for bounded, typically $0/1$, losses; for the regression losses used here the technically correct quantity is the discrepancy distance of Mansour et al. Mansour et al. (2009), which specializes to the $\mathcal{H}\Delta\mathcal{H}$-divergence only in the binary case. We keep the $\mathcal{H}\Delta\mathcal{H}$ notation for readability and use the bound qualitatively, to reason about the effect of representation alignment rather than to derive a numerical rate.

In standard federated averaging (e.g., FedAvg), a single global model is trained on $\mathcal{D}_s$ without explicit representation alignment, operating directly in the input space $\mathcal{X}$. Under severe non-IID conditions, the divergence term $d_{\mathcal{H}_{\mathcal{X}}\Delta\mathcal{H}_{\mathcal{X}}}(\mathcal{D}_s^{(x)}, \mathcal{D}_t^{(x)})$ remains large, leading to client drift and poor target performance. While a direct comparison of bounds across different representation spaces requires care (as the hypothesis classes differ), AutoFed's strategy of learning a representation space where client distributions are more aligned provides an additional degree of freedom that FedAvg lacks.

AutoFed mitigates this via the Federated Representor. Let $\phi_\Theta : \mathcal{X} \to \mathcal{P}$ denote the deterministic mapping from input space $\mathcal{X} = \mathbb{R}^{\mathcal{T}\times n\times K}$ to prompt space $\mathcal{P} = \mathbb{R}^{n\times k}$ induced by the FR with shared

parameters Θ. For a hypothesis class $\mathcal{H}_{\mathcal{P}}$ defined directly on the prompt space, the standard bound applied in the prompt space yields:

$$\epsilon_t^{\text{AutoFed}}(h) \leq \epsilon_s^{\text{AutoFed}}(h) + d_{\mathcal{H}_{\mathcal{P}}\Delta\mathcal{H}_{\mathcal{P}}}(\mathcal{D}_s^{(p)}, \mathcal{D}_t^{(p)}) + \lambda^*_{\text{AutoFed}}, \tag{17}$$

where $\mathcal{D}_i^{(p)} = (\phi_\Theta)_\# \mathcal{D}_i^{(x)}$ is the pushforward of the input marginal distribution. The key design insight of AutoFed is that the shared mapping ϕ_Θ, trained jointly on all clients, is designed to yield a prompt space in which the divergence term $d_{\mathcal{H}_{\mathcal{P}}\Delta\mathcal{H}_{\mathcal{P}}}(\mathcal{D}_s^{(p)}, \mathcal{D}_t^{(p)})$ is small, by learning a shared projection that aligns client distributions. Empirically, clustering visualizations of p_g (Figure 3) exhibit substantially greater overlap across client distributions compared to those of the local feature p_l, indicating that ϕ_Θ preferentially preserves task-relevant shared patterns while suppressing client-specific idiosyncrasies.

Representation alignment carries a trade-off that AutoFed manages rather than incurs. Constraining p_g to a shared subspace makes the effective hypothesis class $\mathcal{H}_{\mathcal{P}}$ more restrictive, which enters the bound through the ideal joint error $\lambda^*_{\text{AutoFed}}$. AutoFed addresses this tension through two mechanisms: (1) client-specific batch normalization layers in the adapter preserve local distributional statistics (mean and variance), allowing the shared linear projection to focus on aligning structural patterns rather than absorbing distributional shifts; and (2) the fully personalized PP can compensate for residual client-specific discrepancies that the global prompt cannot capture, keeping $\lambda^*_{\text{AutoFed}}$ controlled. The auto-encoder within ϕ_Θ bears on the same split from the other side, by shrinking the per-node feature vector before the shared projection sees it. Because the bottleneck retains the persistent component of the input over the stochastic remainder, the representation the shared parameters are fit to carries less of the client's idiosyncratic variation, which lowers the source risk $\epsilon_s^{\text{AutoFed}}(h)$ that the shared block must cover. Our ablation shows this effect on the travel-demand task and finds it negligible on the periodic flow benchmarks, where the series are smoother and a compression at this ratio removes correspondingly less. Together the two mechanisms are what keep λ^* controlled: normalization absorbs distributional shift, and the compression removes scale and redundant variation before the shared projection, so that design needs no per-dataset filter selection.

A contraction factor for how much ϕ_Θ reduces the divergence term is the natural next step, and the empirical alignment of p_g across clients, together with the trade-off management via personalization, helps explain the consistent outperformance across scenarios S0-S3 (Table 1) and the robustness to varying numbers of clients in the TFP task (Table 2), while using a single configuration across datasets rather than dataset-specific manual graph construction or pattern-filter selection.

E.4 Prompt as Low-Dimensional Task Conditioning

We provide an additional theoretical perspective on why conditioning the PP decoder with a prompt p_g is both effective and communication-efficient. Let $f_\phi(\cdot, H_{\mathcal{T}})$ denote the decoder function with respect to its initial token. By first-order Taylor expansion around $p_g = \mathbf{0}$ (valid when $\|p_g\|$ is moderate relative to the decoder's curvature):

$$f_\phi(p_g, H_{\mathcal{T}}) \approx f_\phi(\mathbf{0}, H_{\mathcal{T}}) + J_\phi \cdot \text{vec}(p_g), \tag{18}$$

where $J_\phi = \partial f_\phi / \partial p_g|_{p_g=\mathbf{0}} \in \mathbb{R}^{(\mathscr{T}n)\times(nk)}$ is the Jacobian. The prompt $p_g \in \mathbb{R}^{n\times k}$ thus steers the decoder output within a subspace of dimension at most $\min(\mathscr{T}n, nk)$, determined by the column space of J_ϕ. We describe this as low-dimensional rather than low-rank: the matrix p_g has rank at most $\min(n, k)$, but what matters for the argument is the dimension of the output subspace it can reach, so the comparison is between the prompt dimension and the decoder output dimension, not between k and $\mathcal{T}K$.

This aligns with the intrinsic dimensionality hypothesis Aghajanyan et al. (2021), which states that task-specific adaptations in overparameterized models lie in a low-dimensional subspace. In traffic prediction, cross-client variations (e.g., amplitude scaling, phase shifts) can be captured by a compact prompt. Moreover, because the prompt serves as the initial token in the auto-regressive decoder, its influence propagates through the entire recurrent chain $h_1 \to h_2 \to \cdots \to h_{\mathscr{T}}$, creating an amplification effect that allows a low-dimensional signal to steer the full prediction trajectory.

For federated learning the relevant point is that AutoFed communicates only the shared FR components (Θ) that generate the prompt, not the decoder parameters ϕ nor the prompt itself. This decouples expressiveness (nk) from communication cost ($|\Theta_{\text{shared}}|$), providing a favorable trade-off validated in Table 3. The communication saving itself comes from the size of the shared block and not from any rank property of the prompt: as Proposition 1 shows, $|\Theta| = O(k^2 + Kk)$ is independent of the node count, whereas a fully shared model transmits a backbone that carries node-indexed tensors and therefore grows with the graph.

E.5 Practical Implications

The theoretical analysis is consistent with the design of AutoFed. The learnable prompt p_g, produced by the auto-encoder, the graph encoder, and the shared linear projection, replaces the hand-crafted low-pass filters and pattern repositories used in prior work Zhou et al. (2024; 2025), which removes a class of expert-specified hyper-parameters whose published sensitivity exceeds 5% across datasets. The ablation isolates the two shared mechanisms: removing the client-aligned adapter degrades every metric, while removing the auto-encoder degrades the travel-demand task and leaves the periodic flow benchmarks within one standard deviation, which matches a reading of the module as a feature reduction whose yield depends on how much redundant variation a series carries.

In summary, AutoFed's prompt-conditioning mechanism provides a theoretically motivated framework for cross-client knowledge transfer, reduced manual engineering, and robust generalization under realistic non-IID traffic data distributions. For a fixed auxiliary weight and full participation, its federated training is shown to drive the average squared gradient of the surrogate objective to zero and to admit an optimal objective no larger than that of a fully shared model (Proposition 2 and Corollary 1). Two quantitative refinements follow directly from the analysis: a sharp contraction factor for the cross-client divergence in the prompt space, and tight drift constants for the many local step regime.

F Limitation and Future Work

AutoFed automates the dataset-specific design that prior traffic PFL methods depend on, namely the hand-crafted graph construction and the manual selection of pattern-extraction filters that must be re-tuned whenever the dataset or client configuration changes. Like any deep model, it still exposes a few standard architectural and optimization hyper-parameters, such as the hidden dimension, the number of layers, the prompt dimension, and the learning rate. These are decoupled from per-dataset feature engineering, and we fix them to a single configuration across all datasets rather than tuning them individually. Automatically selecting the prompt dimension, which balances representation capacity against communication cost, is a promising extension.

AutoFed also trades a modest, constant-factor increase in local computation for substantially lower communication (Table 3), which is a favorable trade-off in cross-silo settings where bandwidth rather than local compute is the bottleneck. On the theoretical side, we show in Appendix E that for a fixed auxiliary weight the average squared gradient of the surrogate objective vanishes under standard assumptions, and that the optimal personalized objective is no larger than the corresponding fully shared objective; covering the adaptive weighting itself and sharpening these results with a quantitative contraction factor for the cross-client divergence are promising directions. Finally, our study focuses on single-modality traffic flow and travel-demand data, and extending AutoFed to multimodal inputs and to broader cross-domain problems is a promising avenue.