MOBIFLOW: DATASETS AND BENCHMARKS FOR FEDERATED LEARNING IN REAL-WORLD MOBILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in Federated Learning (FL) have established it as a promising paradigm for real-world domains where privacy, regulatory constraints, or data ownership concerns prevent centralized data collection, such as mobility. Despite growing interests in tasks including traffic or passenger flow prediction, the empirical evaluation of FL strategies in the mobility domain remains limited. Due to privacy concerns, existing studies often rely on proprietary datasets, hindering reproducibility, while publicly available mobility datasets are rarely designed for federated settings.

To address these limitations, we introduce *MobiFLow*, an open-source federated dataset suite for mobility applications built upon *natural partitions*, i.e., splits derived from intrinsic dataset properties. The suite comprises seven datasets spanning diverse mobility tasks, modalities, and scales, covering both cross-device and cross-silo FL scenarios. MobiFLow is complemented by standardized preprocessing, baseline models, benchmarking utilities, and practical evaluation guidelines to support reproducible research. Its modular design allows researchers to seamlessly integrate new datasets and federated learning strategies within a unified framework. We provide baseline results using widely adopted FL strategies and show that realistic mobility heterogeneity makes collaboration far from uniformly beneficial: local training is often surprisingly strong, pooled training can be competitive when client distributions are relatively homogeneous, and no general-purpose FL method consistently dominates across datasets, underscoring the need for realistic benchmarks and mobility-aware FL methods. By releasing datasets, code, and benchmarking tools, MobiFLow aims to promote reproducible research and establish a common ground for the systematic evaluation and fair comparison of FL methods in mobility.

1 INTRODUCTION

Recent advances in Federated Learning (FL) have established it as a promising paradigm for collaboratively training machine learning models across multiple data providers without requiring the exchange of raw, potentially sensitive data (McMahan et al., 2017). By design, FL enables decentralized learning while preserving data locality, making it particularly appealing in domains where privacy, regulatory constraints, or data ownership concerns prevent centralized data collection. Depending on the nature of the participating entities, FL is commonly categorized into two settings: *cross-device* FL, where a large number of edge or Internet of Things (IoT) devices participate, and *cross-silo* FL, where a smaller number of more powerful clients—such as companies, institutions, or organizations—collaborate. These two settings present fundamentally different challenges (Kairouz & McMahan, 2021). Cross-device FL must operate under the constraints of edge environments, including limited bandwidth, memory, and computational capacity (Lim et al., 2020). It also requires addressing issues such as unreliable client availability, large-scale participant selection (Yang et al., 2021), system heterogeneity across devices, and communication efficiency (Kairouz & McMahan, 2021). In contrast, cross-silo FL typically benefits from more stable infrastructure, higher computational resources, and reliable connectivity (Li et al., 2021). However, it is particularly affected by statistical heterogeneity, as data distributions may vary significantly across organizations due to differing collection processes, or user populations (Rocco di Torrepadula et al., 2025), often introducing biases that can degrade model performance and fairness.

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.

Given its recognized benefits, such as privacy preservation and reduced data movement, FL is gaining increasing attention in real-world applications, including mobility-related tasks like traffic forecasting (Liu et al., 2020), demand estimation, and passenger flow prediction (Rocco di Torrepadula et al., 2025). Such interest is further driven by the sensitivity of the mobility data, often reflecting individual movement patterns and behavioral habits (Xia et al., 2023). Mobility-related problems can naturally be formulated in both cross-device and cross-silo settings. In a cross-device scenario, clients may correspond to sensors deployed across road networks, vehicles, or personal devices used by drivers and passengers. In a cross-silo setting, clients may represent higher-level entities such as public transportation operators, mobility service providers, or road infrastructure managers (Rocco di Torrepadula et al., 2025).

Despite this growing interest, the empirical evaluation of FL methods in mobility remains limited. The availability of suitable mobility datasets is constrained by privacy concerns and access restrictions (Russo et al., 2025), and existing public datasets are rarely designed or used in federated settings. Consequently, many studies rely on proprietary data, hindering reproducibility and preventing fair and consistent comparisons across approaches (Kapoor & Kumar, 2024; Russo et al., 2025). Recent efforts have emphasized the importance of realistic and standardized benchmarks for evaluating FL methods in practical scenarios, spanning tasks from image classification (Lai et al., 2022; Liang et al., 2021) to natural language processing (Ye et al., 2024), and domains such as healthcare (Ogier du Terrail et al., 2022) and finance (Liang et al., 2021). However, to the best of our knowledge, no existing benchmark addresses the specific challenges of mobility data, which are characterized by strong spatial and temporal dependencies, as well as complex and structured patterns of data heterogeneity.

To address this gap, we introduce *MobiFLow*, an open-source federated dataset suite for mobility applications built upon natural data partitions. By natural partitions, we refer to splits derived from intrinsic properties of the datasets—such as sensor identifiers, service providers, or geographic regions—thereby reflecting real-world data ownership and distribution. This design enables a more faithful representation of client heterogeneity compared to commonly used artificial partitioning schemes. In addition, MobiFLow is designed to support both cross-device and cross-silo scenarios, enabling a unified evaluation framework across different FL settings. The framework is further complemented by code examples and benchmarking guidelines to facilitate reproducible and extensible research. Our ambition is for MobiFLow to become a standard reference benchmark for federated learning in mobility. In summary, our contributions are threefold:

- We develop an open-source federated mobility dataset suite covering both cross-device and cross-silo scenarios. The suite includes seven publicly available datasets, including one previously unreleased, spanning multiple mobility-related tasks such as traffic and passenger flow prediction, with diverse modalities and scales. To enable consistent comparisons, we define natural client partitions reflecting real-world data distributions and provide a dedicated preprocessing pipeline for each dataset.
- We standardize tasks, metrics, reference models, three-fold protocols, and local, pooled, and federated settings; provide guidelines for fair and reproducible evaluation; and present illustrative results highlighting the impact of key design choices.
- We release open-source code to ensure reproducibility and ease of integration with existing FL frameworks. Our implementation is designed to be extensible, enabling the research community to contribute additional datasets, benchmarking protocols, and FL methods to MobiFLow.

2 Related Work

Recently, considerable effort has been devoted to develop benchmark suites for fair and comprehensive evaluation of FL methods, focusing on different aspects. A key challenge in FL is statistical heterogeneity across clients (non-iid data), a defining feature of real-world deployments that must be explicitly considered when comparing approaches (Tan et al., 2023; Kulkarni et al., 2020). To this purpose, several benchmarks adopt *natural client splits*, i.e., partitions derived from meaningful attributes inherent to the data, rather than relying on synthetic strategies such as label-based splits (McMahan et al., 2017), Dirichlet sampling (Gutierrez et al., 2024), or generative models like the Pachinko Allocation Model (PAM) (Li & McCallum, 2006), which often fail to capture the complex and structured heterogeneity observed in real-world federated datasets. In this direction, the

LEAF benchmark suite (Caldas et al., 2018) focuses on cross-device FL and provides naturally partitioned datasets spanning a variety of tasks, including language modeling, sentiment analysis, and image classification. Similarly, FedScale (Lai et al., 2022) targets large-scale mobile environments, emphasizing scalability and system-level challenges, while TensorFlow Federated (Bonawitz et al., 2019) offers a broader collection of benchmarks covering both cross-device and cross-silo scenarios. In the cross-silo setting, FLamby (Ogier du Terrail et al., 2022) represents a notable contribution, aggregating multiple medical datasets into a unified benchmark suite with natural client partitions, complemented by reproducible code and standardized evaluation protocols. More recent efforts have further specialized benchmark design to study specific FL settings and challenges. For instance, (Chen et al., 2022) introduces a benchmark tailored to personalized FL, accounting for practical aspects such as partial client participation and generalization to previously unseen clients. (Zhang et al., 2025) focuses on heterogeneous FL, where client models may differ significantly in architecture or capacity, while (Ye et al., 2024) targets the federated fine-tuning of large language models, reflecting the growing interest in adapting FL to emerging model paradigms.

Despite these advances, existing benchmarks primarily cover domains such as computer vision (Lai et al., 2022; Liang et al., 2021), natural language processing (Ye et al., 2024), healthcare (Ogier du Terrail et al., 2022), and finance (Liang et al., 2021). To the best of our knowledge, no benchmark currently exists for mobility datasets, which introduce distinct challenges, primarily due to their strong spatial and temporal characteristics, requiring careful handling to ensure meaningful evaluation. Consequently, most existing studies rely on proprietary datasets that cannot be shared due to privacy concerns, limiting reproducibility and hindering fair comparisons across methods (Rocco Di Torrepadula et al., 2024). To address this gap, we propose MobiFLow, a unified suite of datasets and benchmarks for federated mobility applications.

3 The MobiFLow Dataset Suite

3.1 Structure Overview

MobiFLow is implemented as a Python library organized around two core components: (i) a collection of datasets coupled with baseline models and evaluation metrics, and (ii) a suite of federated learning (FL) strategies with dedicated benchmarking utilities. The framework is explicitly designed to be modular and extensible. Each component follows a simple and standardized application programming interface (API), enabling straightforward reuse, reproducibility, and seamless extension to new datasets and FL methods. The full implementation is publicly available at `https://anonymous.4open.science/r/MobiFLow`.

3.2 Datasets, Metrics, and Baseline Models

MobiFLow comprises seven mobility datasets, each with a natural client partition based on intrinsic properties such as collection devices, service providers, or sensor locations. For each dataset, we define a specific predictive task (e.g., traffic forecasting or passenger flow prediction), evaluation metrics, and a reproducible cross-validation protocol. In scenarios involving time series forecasting, we adopt time-aware cross-validation strategies (Bergmeir & Benítez, 2012), meant to preserve temporal ordering. In addition, we provide a baseline model for each task. Our goal is to establish a consistent and controlled benchmark for FL in realistic mobility scenarios, rather than establishing state-of-the-art models on individual datasets. We therefore adopt simple and reproducible models across local, pooled, and federated settings, allowing performance differences to be attributed to the learning paradigm rather than architectural choices. To this purpose, whenever prior work defines a standard reference architecture for a dataset (e.g., Ibañez et al. (2025); Zhou et al. (2025)), we reuse it to maintain comparability. Otherwise, we select models that are widely used in the relevant literature (Rocco Di Torrepadula et al., 2024). Overall, these choices prioritize fairness and reproducibility over raw predictive performance. Nevertheless, the modular design of MobiFLow facilitates the future integration of more advanced or domain-specific architectures, such as graph neural networks (GNNs), which are gaining prominence in mobility applications. To demonstrate this extensibility, Appendix C.1 presents additional experiments with two GNN-based baselines for the SmartMaceió and FedBologna datasets.

Table 1: Overview of the datasets included in MobiFLow.

Property	FedBologna	FedCollecty	FedCrawdad	FedMelbourne	FedMETR-LA	FedSmartMaceió	FedSUNT
Mobility setting	Urban road traffic	Multimodal personal mobility	Taxi mobility traces	Rail passenger demand	Road-network traffic forecasting	Smart-city traffic monitoring	Public-transport demand
Input (x)	Vehicle counts with spatial and temporal information	Smartphone inertial and device traces	Sequential GPS, timestamps, trip-state	Boarding and service data with spatial and temporal information	Loop-detector speed time series on graph	Urban traffic speed measurements	Ticketing data with spatial and temporal information
Task type	Time series forecasting	Classification	Sequential prediction	Time series forecasting	Forecasting	Forecasting	Time series forecasting
Prediction (y)	Future vehicle count	Transport mode	Mobility state / next-step behavior	Future passenger load	Future traffic speed	Future speed	Future passenger boarding
Acquiring Entity	City municipality	Research Group	-	Transport company	Transport company	Traffic optimization company	Transport company
Acquiring sensors	Traffic sensors	Smartphones	Automatic Ticketing and Vehicle Location Systems	Ticketing systems and barrier counts	Traffic Sensors	Traffic Sensors	Automatic Ticketing and Vehicle Location Systems
Dataset source	(Comune di Bologna, Progetto Open Data, 2025)	(Erdelić et al., 2023)	(Piorkowski et al., 2022)	(Open Data, Transport Victoria Region, 2025)	(Guo et al., 2019)	New Dataset	(Ferreira et al., 2025)
# clients (cross-silo)	6	–	–	11	–	–	5
# clients (cross-device)	841	13	533	–	207	12	2,499
# examples	5,915,004	831,897	360,806	11,976,536	7,085,196	2,166,480	45,288,238
Temporal span	1 year	5 months	1 year	6 years	4 months	2 years and 5 months	1 year
Client extraction	Spatial groups of urban sensors	User- or device-level split	Taxi-level split	Operational line / service-group silos	Spatial sensor communities	Spatial road-segment or sensor groups	Spatial clusters of stops or sensors
Model	3-layer LSTM	LSTM (Ibañez et al., 2025)	RNN (Rossi et al., 2019)	BiLSTM (Liyanage et al., 2022)	BiLSTM (Ounoughi & Yahia, 2024)	Transformer (Zhou et al., 2025)	2-layer LSTM (Ferreira et al., 2025)
Metrics	RMSE	Macro-F1	Accuracy / Macro-F1	RMSE	RMSE	RMSE	RMSE
Input dimension	(3, 20)	(9, 512)	(2, 8)	(3, 22)	(3, 12)	(3, 12)	(3, 16)

For each dataset, the framework also offers comprehensive documentation, metadata, and utilities to: (i) download the raw data; (ii) construct natural client partitions; (iii) perform preprocessing; (iv) iterate over the preprocessed dataset; (v) tune the baseline model. Built on PyTorch[1], the interface simplifies access to federated partitions while allowing their customization for alternative experimental settings. The datasets included in MobiFLow are summarized in Table 1 and discussed as follows. Further details are available in Appendix A.

FedBologna is derived from open data released by the municipality of Bologna, Italy (Comune di Bologna, Progetto Open Data, 2025). It consists of hourly vehicle flow measurements collected throughout 2025 by road sensors distributed across different areas of the city. The associated learning task is short-term traffic forecasting, where the objective is to predict future vehicle counts. The dataset naturally supports both cross-device federation, where each sensor is a client, and cross-silo federation, where sensors are grouped by administrative district. Following a recent mobility-forecasting survey by Rocco Di Torrepadula et al. (Rocco Di Torrepadula et al., 2024), which identifies recurrent models as a standard choice for short-term sequential forecasting, we use a three-layer LSTM baseline and evaluate performance with RMSE.

FedCollecty is based on the Collecty multimodal smartphone dataset (Erdelić et al., 2023), which contains mobility traces (i.e., sequences of timestamped GPS positions) collected from smartphones and labelled with transport modes (e.g., bike, public transit, on foot). This dataset targets transport mode classification from inertial and device-level signals, and naturally supports a cross-device federated setting in which each client corresponds to a user or a device. Compared to the other forecasting datasets in the suite, FedCollecty introduces a classification-oriented benchmark, with strong user-level heterogeneity. As a baseline, we adopt an LSTM classifier following the recent work of Ibañez et al. (Ibañez et al., 2025), which applies the same architecture to the analogous task of transport-mode detection from smartphone sensor data. Performance is assessed using macro-averaged F1.

FedCrawdad is derived from taxi mobility traces collected in the San Francisco Bay area, USA, within the CabSpotting project (Piorkowski et al., 2022). The dataset consists of taxi mobility traces (e.g., timestamped sequences of GPS positions), annotated with taxi occupation status, collected from approximately 500 taxis over a period of 30 days. In our benchmark, we use this dataset for next-destination prediction from recent pick-up/drop-off histories, following Rossi et al. (Rossi et al., 2019), who formulate the same task on this dataset and adopt an RNN model. Accordingly, we use an RNN as the reference baseline. Clients are defined at the taxi level, yielding a natural cross-device federation.

FedMelbourne is derived from the Transport Victoria Open Data Portal, which provides access to open mobility data from the Victoria region in Australia. We consider the Melbourne train passenger-count dataset, which reports boardings, alightings, and load-related statistics for individual train services from 2018 to 2024 (Open Data, Transport Victoria Region, 2025). Within our

[1] https://pytorch.org/

benchmark, this dataset targets short-term passenger-load forecasting. Because the dataset does not expose a natural device-level split, we consider only a cross-silo setting, where clients correspond to service-area groups. Following prior work on passenger-demand prediction in Melbourne (Liyanage et al., 2022), we use a BiLSTM baseline and evaluate performance with RMSE.

FedMETR-LA builds on METR-LA, a widely used dataset for multi-step traffic-speed forecasting from loop detectors across Los Angeles (Guo et al., 2019; Jagadish et al., 2014). Although not originally federated, it naturally supports sensor-based client partitions and has been adopted in prior FL studies (Al-Huthaifi et al., 2024; Liu et al., 2024).However, the lack of a standardized benchmark has led to inconsistent data partitioning and evaluation protocols, hindering fair comparisons. To address this, MobiFLow provides a standardized formulation in which each detector represents a client. As for the baseline model, while recent work on METR-LA (e.g. FedAGAT (Al-Huthaifi et al., 2024) and REFOL (Liu et al., 2024)) employs sophisticated graph and concept-drift components, MobiFLow prioritizes simplicity and auditability, adopting GRIZZLY (Ounoughi & Yahia, 2024), a sequence-to-sequence stacked bidirectional LSTM.

FedSmartMaceió is a *novel* dataset firstly introduced in this benchmark. It is built from traffic-speed measurements collected in Maceió, Brazil, where each sensor records average vehicle speed at regular intervals. Hence, the associated task is short-term traffic-speed forecasting. The dataset naturally supports a cross-device federated setting, with each sensor treated as an individual client. As no established baselines exist for this dataset, we adopt a transformer-based model given their recent success in mobility forecasting. Specifically, we use LGSTformer-T, a temporal-only variant of LGSTformer (Zhou et al., 2025), where the spatial module is removed (each client corresponds to a single sensor) and only the input embedding and temporal module (LGTM) are retained.

FedSUNT is derived from the Salvador Urban Network Transportation (SUNT) dataset (Ferreira et al., 2025), a large-scale public transport dataset integrating ticketing records, vehicle information, and stop-level data for bus and subway systems. The dataset covers the period from March 2024 to March 2025 and includes data from approximately 2,000 vehicles operating across nearly 400 lines, connecting around 3,000 stops and stations in the city of Salvador, Brazil. In MobiFLow, this dataset targets short-term passenger-flow forecasting (specifically, number of passengers boarding vehicles at upcoming stops) and supports both cross-device federation, where clients are vehicles, and cross-silo federation, where clients are route clusters. Following the code associated with the original dataset paper (Ferreira et al., 2025), we use a single-layer LSTM baseline and evaluate performance with RMSE.

3.3 Federated Learning Strategies in MobiFLow

To facilitate FL benchmarking, MobiFLow provides a set of seven federated training algorithms, referred to as *strategies*. These methods are implemented in plain PyTorch to ensure independence from any specific FL framework. At the same time, MobiFLow remains compatible with alternative implementations from existing federated learning libraries. All strategies follow a unified API that is fully compatible with the dataset interface, ensuring that each method can be consistently evaluated across all datasets. The framework further includes scripts for FL strategies hyperparameter tuning as well as end-to-end benchmarking, enabling users to reproduce experiments and conduct comparative analyses with minimal effort.

We note that, while mobility tasks require specialized models to properly capture their spatiotemporal structure, a recent survey (Rocco Di Torrepadula et al., 2024) highlights that many existing approaches are difficult to reproduce, either due to unavailable implementations (Li & Liu, 2023) or code tightly coupled with proprietary datasets (Rocco di Torrepadula et al., 2025). For this reason, MobiFLow focuses on standard, widely used federated optimization baselines, detailed in the following, that can be consistently implemented and fairly evaluated across datasets. Although not tailored to mobility, this choice ensures a controlled and reproducible benchmarking setup, avoiding biases from dataset-specific designs. Notably, the modular structure of MobiFLow allows easy integration of new federated learning methods, enabling consistent evaluation of future or previously unavailable approaches.

FedAvg. Federated Averaging (FedAvg) (McMahan et al., 2017), is the simpler FL baseline. At each communication round, every client initializes from the current global model, performs a fixed number of local mini-batch updates on its own data, and sends the resulting model parameters to the

Table 2: Summary of client-level data heterogeneity across datasets. We report the number of clients, data size statistics (min/median/max), size imbalance via coefficient of variation (CV), and distribution heterogeneity using Jensen–Shannon (JS) divergence and Wasserstein distance relative to the global distribution.

Dataset	Setting	# Clients	Size (min/med/max)	Size CV	JS (mean/max)	Wasserstein (mean/max)
FedBologna	cross-device	841	165 / 5.4k / 10.8k	0.07	0.13 / 0.30	0.03 / 0.17
	cross-silo	6	426k / 836k / 890k	0.23	0.003 / 0.005	0.004 / 0.007
FedCollecty	cross-device	13	530 / 26k / 44k	1.13	0.21 / 0.52	1.43 / 2.83
FedCrawdad	cross-device	533	2 / 491 / 1.2k	0.36	0.01 / 0.14	0.001 / 0.003
FedMelbourne	cross-silo	10	0 / 438k / 1.78M	0.86	0.009 / 0.029	0.005 / 0.011
FedMETR-LA	cross-device	207	24k / 24k / 24k	0.00	0.11 / 0.49	0.08 / 0.37
FedSmartMaceio	cross-device	12	129k / 129k / 129k	0.00	0.016 / 0.027	0.047 / 0.063
FedSUNT	cross-device	2438	0 / 14k / 41.7k	0.53	0.02 / 0.41	0.005 / 0.075
	cross-silo	5	1.36M / 5.0M / 10.0M	0.67	0.002 / 0.004	0.001 / 0.002

server. The server then computes a sample-size-weighted average and broadcasts the updated global model back to all clients. FedAvg is simple and widely used, but its performance can deteriorate when client distributions are strongly heterogeneous and the number of local updates becomes large.

FedProx. FedProx (Li et al., 2020) extends FedAvg by adding a proximal regularization term to the local objective. This term penalizes large deviations between a client model and the current global model, thereby stabilizing local training under heterogeneous data distributions. In our setting, FedProx serves as a natural heterogeneity-aware baseline for mobility scenarios where regional, temporal, or operator-specific shifts may induce substantial client drift.

SCAFFOLD. SCAFFOLD (Karimireddy et al., 2020) addresses client drift through control variates that correct the bias introduced by local updates on non-identically distributed data. Intuitively, each client maintains a correction term that helps align its local optimization trajectory with the global objective, while the server uses the aggregated information to update the shared model. We include SCAFFOLD as a principled baseline designed specifically for heterogeneous federated optimization.

Cyclic Learning. Cyclic Learning (Chang et al., 2018) provides a sequential alternative to parallel aggregation-based FL. Instead of averaging client updates at every round, the model is passed from one client to the next, with each client performing local optimization before handing the updated model to the following participant. In our benchmark, one round corresponds to a full pass through all clients. This strategy serves as a simple yet informative sequential baseline.

FedOpt Variants: FedAdam, FedYogi, and FedAdagrad. We also evaluate the adaptive federated optimization methods FedAdam, FedYogi, and FedAdagrad (Reddi et al., 2021) , which extend Adam (Kingma & Ba, 2014), Yogi (Zaheer et al., 2018), and Adagrad (Lydia & Francis, 2019) to FL. By maintaining server-side statistics of aggregated client updates, these methods adapt the global update and can improve convergence when client updates vary substantially in magnitude or direction (Reddi et al., 2021), providing stronger alternatives to plain averaging.

3.4 Dataset Heterogeneity

As described in Section 3.2, the considered datasets span a wide range of mobility scenarios, tasks, and federated learning settings, with temporal spans ranging from weeks to years (e.g., FedMelbourne). To further characterize these datasets, Table 2 summarizes client-level data heterogeneity, which can significantly influence the effectiveness of FL strategies. In particular, we report the number of clients, client data size statistics (min/median/max), size imbalance via coefficient of variation (CV), and distribution heterogeneity measured by Jensen–Shannon (JS) divergence and Wasserstein distance relative to the global distribution. Complementary insights are provided in Figure 1, which shows the empirical cumulative distribution functions (ECDFs) of the target variable for each dataset. The global distribution is shown in black, while individual client distributions are depicted in color. For readability, in cross-device settings we display a subset of 10 clients, including the 8 most dissimilar and 2 most similar (dashed) to the global distribution.

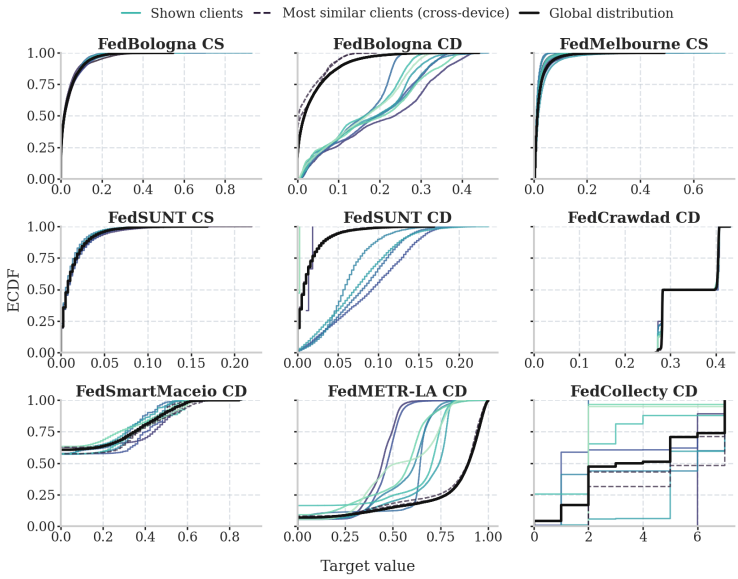


Figure 1: ECDFs of the target variable across datasets. In cross-device settings, 10 clients are shown (8 most dissimilar and 2 most similar to the global distribution).

Taken together, Table 2 and Figure 1 reveal a consistent pattern: cross-device datasets exhibit substantially higher distribution heterogeneity compared to their cross-silo counterparts, as observed for both FedBologna and FedSUNT. This reflects the more diverse and fragmented nature of data collected across many individual devices. An exception is FedSmartMaceió, which, despite being a cross-device dataset, shows relatively low heterogeneity; this can be attributed to the limited number of clients (only a dozen), resulting in more homogeneous data partitions. In contrast, cross-silo datasets tend to be close to i.i.d. in terms of data distribution, as indicated by the very low JS and Wasserstein values. However, they may still exhibit a significant imbalance in client data sizes, as in the case of FedSUNT and FedMelbourne, which include clients with empty data.

4 An Example of FL Benchmark with MobiFLow

This section describes a reference protocol to perform a FL benchmark on mobility datasets using MobiFLow, with the goal of enabling fair and reproducible comparisons across FL strategies. Thanks to its modular design, the framework can be readily extended to incorporate additional datasets, new algorithms, or alternative experimental protocols. All experiments were run on a Debian 6.12.74 server with an NVIDIA A100 GPU, using Python 3.11 and PyTorch 2.11.0 with GPU acceleration. A Docker image is provided in the repository for reproducibility.

4.1 Cross-Validation Details

To obtain statistically meaningful results, we adopt a three-fold cross-validation scheme for each client dataset. For tasks involving time series forecasting, we employ time-aware cross-validation: the data is split into progressively expanding training sets, along with corresponding validation and test sets, across multiple folds defined by different temporal cut-off points. This ensures that temporal ordering is preserved within each fold. Model performance is evaluated on the local test sets and subsequently aggregated by averaging across clients and folds.

4.2 Hyperparameters Tuning and Baselines

We compare three learning settings: *pooled*, where a single centralized model is trained on all data; *local*, where each client trains independently, without cross-client collaboration; and *federated*, where clients collaboratively train a shared model by exchanging updates while keeping data local.

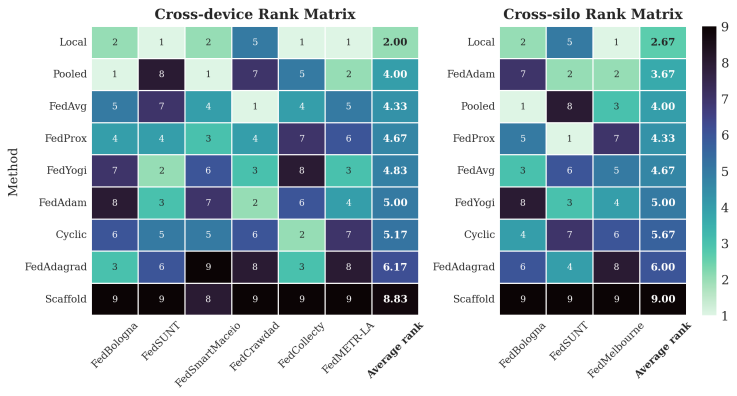


Figure 2: Ranking matrix of the evaluated strategies across all datasets, based on RMSE for regression and Macro-F1 for classification. The average rank of each strategy is shown in bold.

For each dataset, model hyperparameters are first optimized in the pooled setting and then reused for local training. In addition to model-specific hyperparameters, each FL strategy introduces its own set of configuration hyperparameters, such as local and server learning rates. To fine-tune them, for each dataset, we then fix the model architecture and hyperparameters to those used in the pooled and local baselines, and restrict the optimization process to FL-specific hyperparameters only, consistently with prior work (Ogier du Terrail et al., 2022). This design isolates the effect of federation from architectural and model-selection confounders. Further details are provided in Appendix B.

4.3 Federated Setup

To ensure fair comparisons between FL strategies and the pooled baseline, we standardize the training budget so that the total number of optimization steps matches as closely as possible across settings, following (Ogier du Terrail et al., 2022). The pooled budget is computed as $\left\lfloor \frac{N}{B} \right\rfloor \cdot E$, where N is the number of training samples, B the batch size, and E the number of epochs. Hence, for federated settings, we fix the number of communication rounds to $R = 200$ and distribute the budget uniformly across rounds and participating clients. This yields the number of local updates per client:

$$\text{num_updates} = \frac{\left\lfloor \frac{N}{B} \right\rfloor \cdot E}{R \cdot C} \tag{1}$$

where C is the number of participating clients per round, depending on the FL setting. In cross-device settings, we use partial participation with $C = 30$ active clients per round, or all eligible clients when fewer than 30 are available. In cross-silo settings, all clients participate unless otherwise specified. Experiments that depart from this reference configuration explicitly report the corresponding participation setting. We fix the number of communication rounds and derive the number of local updates accordingly, rather than the converse, due to the varying sizes of the datasets. This ensures a consistent communication budget while preserving comparability with the pooled baseline in terms of total computation.

4.4 Benchmark Results

An overview of the benchmarking results obtained with MobiFLow is presented in Figure 2 in the form of a rank matrix. Additional details, including statistical significance and pairwise comparisons, are reported in Appendix C. The main empirical finding is that, under realistic mobility heterogeneity, collaboration is not uniformly beneficial: local training is often the strongest baseline, pooled training is competitive when client distributions are relatively homogeneous, and no single FL strategy dominates across datasets. More in detail, according to the rank matrix *local baselines* are, on average, the best-performing models. In scenarios characterized by strong heterogeneity across clients (e.g., FedSUNT, FedCollecty, and FedMETR-LA in the cross-device setting), collaboration across clients appears to be less effective and may even degrade performance compared to

training models independently on each client. In some cases, the strong performance of local models can also be attributed to the substantial size of the local datasets. For instance, in FedMelbourne, each client has access to several years of data, which is sufficient to train highly accurate models without requiring any form of collaboration.

On average, the second-best performing approach is the *pooled baseline*, which outperforms local training in specific scenarios, such as cross-device FedBologna and FedSmartMaceio, and cross-silo FedSUNT. These results can be explained considering the underlying data characteristics summarized in Table 2. In FedSUNT and FedSmartMaceio, the data distribution across clients is close to i.i.d., as indicated by the low JS divergence values, making centralized training particularly effective. In contrast, in FedBologna, clients often have limited local data, which highlights the benefit of aggregating information across clients.

Considering FL strategies, no single method consistently emerges as the best across all scenarios. Adaptive methods such as *FedAdagrad, FedAdam*, and *FedYogi* generally achieve the strongest performance among FL baselines, often approaching the best-performing method. This is particularly evident in settings with moderate heterogeneity (e.g., cross-silo FedSUNT and FedMelbourne, and cross-device FedSmartMaceio and FedCrawdad). *FedProx* emerges as the best-performing method on FedSUNT, likely because its proximal term stabilizes local updates, which can be particularly advantageous in settings with moderate heterogeneity and large local datasets. Conversely, in scenarios with strong heterogeneity and a limited number of clients, such as FedCollecty, alternative approaches like *Cyclic* can represent a competitive solution. In contrast, *Scaffold* underperforms across most scenarios, likely due to its higher sensitivity to hyperparameter choices, which makes it harder to tune effectively in heterogeneous settings.

Overall, these results show that achieving strong FL performance in mobility settings is non-trivial, especially with general-purpose methods. Cross-client collaboration is often ineffective and can even degrade performance, as local baselines frequently outperform both FL methods and, in some cases, the pooled model. This aligns with prior findings (Rocco di Torrepadula et al., 2025) and reflects the inherent heterogeneity of mobility data, where entities operate in distinct regions or contexts. Enforcing uniform collaboration without accounting for such differences can be detrimental, while incorporating domain-specific structure (e.g., spatial information) could substantially improve performance.

5 Conclusion

In this paper, we introduced MobiFLow, an open-source federated mobility dataset suite covering both cross-device and cross-silo scenarios. Through standardized datasets, code examples, and evaluation guidelines, MobiFLow supports reproducible and systematic experimentation across heterogeneous FL mobility settings. We demonstrated how MobiFLow can be used to benchmark a range of widely adopted FL strategies across seven mobility datasets spanning different tasks, modalities, and data distributions. Our results show that general-purpose FL methods often struggle in these settings, underlining the need for FL approaches that explicitly account for domain-specific characteristics such as spatial structure and client-level discrepancies. At the same time, MobiFLow is designed with extensibility in mind: new datasets, tasks, and FL strategies can be seamlessly integrated, enabling future work to evaluate domain-specific methods under a common and controlled experimental setup.

Despite its flexibility, MobiFLow currently relies on GPU acceleration to ensure practical training times, given the scale of several included datasets. While, in principle, the framework could be executed on CPU-only systems, this would be computationally prohibitive in most cases. Furthermore, the current implementation relies on PyTorch; support for other backends, such as TensorFlow or JAX, remains future work. Moreover, MobiFLow standardizes the information provided by the original sources, which do not capture all contextual factors affecting mobility (e.g. weather or public events). Their absence may introduce unobserved distribution shifts. Finally, while the current benchmark primarily focuses on data heterogeneity across clients, other important aspects of FL are not yet explicitly considered. Future extensions could address other dimensions, such as privacy-preserving techniques, personalization strategies, and system-level constraints beyond model-payload volume, including network latency, bandwidth variability, compression, and client availability.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, assist in the writing of proofs, clean and reformat datasets, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to implement methods, generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design or provide feedback on research methodology or experiments. Additionally, we used generative AI tools to create or edit software code.

All AI-assisted outputs were reviewed by the authors. Code was verified and tested for correctness by the first two authors; proof-related writing was reviewed by all authors; and AI-assisted analyses and interpretations were assessed and corrected where necessary. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

MobiFLow aims to support reproducible and systematic research on federated learning for mobility applications. By facilitating controlled comparisons across realistic data partitions, it aims to contribute to more reliable mobility forecasting and transportation planning. Nevertheless, the use of mobility data could raise ethical considerations.

Mobility data may reflect existing geographic, social, and infrastructural inequalities. Consequently, models evaluated with MobiFLow may perform unevenly across regions, transport modes, or user groups and could reinforce disparities if used in transportation planning or service provision. Moreover, although FL reduces the need to centralize data, it does not completely eliminate privacy risks: mobility information remains sensitive and may be misused for surveillance or to infer behavioral patterns. Benchmark results may also be overgeneralized to settings with substantially different data distributions, fairness constraints, or privacy requirements.

We therefore position MobiFLow as a research benchmark rather than a deployment-ready decision system. It does not release high-risk models or re-host raw datasets; instead, it provides benchmarking code, standardized wrappers, baseline implementations, and scripts that retrieve data from official sources under their respective terms. These safeguards reduce provenance and integrity risks but do not remove biases or privacy concerns already present in the source data. Any deployment based on this benchmark should therefore undergo context-specific evaluation of fairness, privacy, robustness, and potential misuse.

We respect the ownership and licensing of all external assets. Each dataset is credited through its original paper or public portal, with available licenses documented in Appendix A, including CC BY 4.0 for FedBologna, FedMelbourne, and FedSUNT. The newly introduced FedSmartMaceio dataset is released under CC BY-NC 4.0. Finally, the benchmark code and FL strategies are implemented by us in PyTorch, while architectures derived from prior work are appropriately cited.

REPRODUCIBILITY STATEMENT

The paper provides all the necessary details to reproduce the experimental results, including dataset construction, preprocessing steps, model architectures and configurations, training protocols, and evaluation procedures. An overview of the datasets and the experimental setup is provided in the main paper, while complete dataset, experimental, sensitivity and resource details are provided in Appendices A–E. To further support reproducibility, we release a well-documented, MIT-licensed GitHub repository (`https://anonymous.4open.science/r/MobiFLow`) containing the full codebase, including scripts to download and preprocess the datasets and to train all considered strategies, along with a Docker image to ensure a consistent and easily deployable environment. Moreover, the FedSmartMaceio dataset, firstly introduced in the paper, is made available under the CC BY-NC 4.0 license through an anonymous Google Drive link (`https://drive.google.com/file/d/1sOwfeTQp4RIFYZoF9dILjlvxYc2Tw0fq/view?usp=sharing`), with clear instructions for access and preparation provided in the paper and in Appendix A. The use of Google Drive is intended to preserve anonymity during the review process; upon acceptance, we plan to migrate the dataset to a more permanent and citable hosting platform (e.g., Zenodo).

REFERENCES

Rasha Al-Huthaifi, Tianrui Li, Zaid Al-Huda, and Chongshou Li. FedAGAT: Real-time traffic flow prediction based on federated community and adaptive graph attention network. *Information Sciences*, pp. 120482, March 2024. ISSN 0020-0255. doi: 10.1016/j.ins.2024.120482. URL https://www.sciencedirect.com/science/article/pii/S0020025524003955.

Christoph Bergmeir and José M Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388, 2019.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Ken Chang, Niranjan Balachandar, Carson Lam, Darvin Yi, James Brown, Andrew Beers, Bruce Rosen, Daniel L. Rubin, and Jayashree Kalpathy-Cramer. Distributed deep learning networks among institutions for medical imaging. *Journal of the American Medical Informatics Association*, 25(8):945–954, 2018. doi: 10.1093/jamia/ocy017. URL https://academic.oup.com/jamia/article/25/8/945/4956468.

Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. pfl-bench: A comprehensive benchmark for personalized federated learning. *Advances in Neural Information Processing Systems*, 35:9344–9360, 2022.

Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2 edition, 1988.

Comune di Bologna, Progetto Open Data. Rilevazione flusso veicoli tramite spire-anno 2025, 2025. URL https://opendata.comune.bologna.it/explore/dataset/rilevazione-flusso-veicoli-tramite-spire-anno-2025/information/. Available at https://opendata.comune.bologna.it/explore/dataset/rilevazione-flusso-veicoli-tramite-spire-anno-2025/information/.

Martina Erdelić, Tomislav Erdelić, and Tonči Carić. Dataset for multimodal transport analytics of smartphone users-collecty. *Data in brief*, 50:109481, 2023.

Marcos V Ferreira, Matheus Souza, Tatiane N Rios, Islame FC Fernandes, Jorge Nery, João Gama, Albert Bifet, and Ricardo A Rios. Salvador urban network transportation (sunt): A landmark spatiotemporal dataset for public transportation. *Scientific Data*, 12(1):1320, 2025.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 922–929, 2019.

Daniel Mauricio Jimenez Gutierrez, Aris Anagnostopoulos, Ioannis Chatzigiannakis, and Andrea Vitaletti. Fedartml: A tool to facilitate the generation of non-iid datasets in a controlled way to support federated learning research. *IEEE Access*, 12:81004–81016, 2024.

Gerard Caravaca Ibañez, Luis J de la Cruz Llopis, Adrian Catalín Diaconeasa, Alberto Bazán Guillén, and Mónica Aguilar Igartua. Mobilitapp: a deep learning-based tool for transport mode detection to support sustainable urban mobility. *IEEE access*, 13:68439–68461, 2025.

Hosagrahar V Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

Peter Kairouz and H Brendan McMahan. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021.

Ayshika Kapoor and Dheeraj Kumar. Federated Learning for Urban Sensing Systems: A Comprehensive Survey on Attacks, Defences, Incentive Mechanisms, and Applications. *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024. ISSN 1553-877X. doi: 10.1109/COMST.2024.3434510. URL https://ieeexplore.ieee.org/abstract/document/10612842. Conference Name: IEEE Communications Surveys & Tutorials.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/karimireddy20a.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of Personalization Techniques for Federated Learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 794–797, July 2020. doi: 10.1109/WorldS450073.2020.9210355. URL https://ieeexplore.ieee.org/abstract/document/9210355.

Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*, pp. 11814–11827. PMLR, 2022.

Can Li and Wei Liu. Multimodal Transport Demand Forecasting via Federated Learning. *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2023. ISSN 1558-0016. doi: 10.1109/TITS.2023.3325936. URL https://ieeexplore.ieee.org/abstract/document/10299587. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3347–3366, 2021.

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. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pp. 577–584, 2006.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018. URL https://arxiv.org/abs/1707.01926.

Yuan Liang, Yange Guo, Yanxia Gong, Chunjie Luo, Jianfeng Zhan, and Yunyou Huang. FLBench: A Benchmark Suite for Federated Learning. In Wanling Gao, Kai Hwang, Changyun Wang, Weiping Li, Zhigang Qiu, Lei Wang, Aoying Zhou, Weining Qian, Cheqing Jin, and Zhifei Zhang (eds.), *Intelligent Computing and Block Chain*, pp. 166–176, Singapore, 2021. Springer. ISBN 978-981-16-1160-5. doi: 10.1007/978-981-16-1160-5_14.

Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE communications surveys & tutorials*, 22(3):2031–2063, 2020.

Qingxiang Liu, Sheng Sun, Yuxuan Liang, Xiaolong Xu, Min Liu, Muhammad Bilal, Yuwei Wang, Xujing Li, and Yu Zheng. Refol: resource-efficient federated online learning for traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 26(2):2777–2792, 2024.

Yi Liu, Shuyu Zhang, Chenhan Zhang, and James J.Q. Yu. FedGRU: Privacy-preserving Traffic Flow Prediction via Federated Learning. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, September 2020. doi: 10.1109/ITSC45102.2020.9294453. URL https://ieeexplore.ieee.org/abstract/document/9294453.

Sohani Liyanage, Rusul Abduljabbar, Hussein Dia, and Pei-Wei Tsai. Ai-based neural network models for bus passenger demand forecasting using smart card data. *Journal of Urban Management*, 11(3):365–380, 2022.

Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Manfredi Napolitano, Alessandra Somma, Alessandra De Benedictis, and Nicola Mazzocca. Flowtwin: A digital twin for traffic flow monitoring. *IEEE Open Journal of Intelligent Transportation Systems*, 6:1551–1568, 2025.

Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, et al. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. *Advances in Neural Information Processing Systems*, 35:5315–5334, 2022.

Open Data, Transport Victoria Region. Train Service Passenger Counts, 2025. URL https://opendata.transport.vic.gov.au/dataset/train-service-passenger-counts. Available at https://opendata.transport.vic.gov.au/dataset/train-service-passenger-counts.

Chahinez Ounoughi and Sadok Ben Yahia. Sequence to sequence hybrid bi-lstm model for traffic speed prediction. *Expert Systems with Applications*, 236:121325, 2024.

Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. Crawdad epfl/mobility, 2022. URL https://dx.doi.org/10.15783/C7J010.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2021. URL https://arxiv.org/abs/2003.00295.

Franca Rocco Di Torrepadula, Enea Vincenzo Napolitano, Sergio Di Martino, and Nicola Mazzocca. Machine Learning for public transportation demand prediction: A Systematic Literature Review. *Engineering Applications of Artificial Intelligence*, 137, 2024. doi: 10.1016/j.engappai.2024.109166. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85202019030&doi=10.1016%2fj.engappai.2024.109166&partnerID=40&md5=5357136c9fd2305bd2483f1d9f66f317. Type: Short survey.

Franca Rocco di Torrepadula, Marco Fisichella, Sergio Di Martino, and Nicola Mazzocca. Fedflow: a personalized federated learning framework for passenger flow prediction. *Machine Learning*, 114(7):1–26, 2025.

Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2980–2989, 2019.

Debora Russo, Franca Rocco Di Torrepadula, Luigi Libero Lucio Starace, Sergio Di Martino, and Nicola Mazzocca. A framework for generating synthetic urban mobility datasets with customizable anomalous scenarios. *IEEE Open Journal of Intelligent Transportation Systems*, 6:1439–1458, 2025.

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, December 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2022.3160699. URL https://ieeexplore.ieee.org/abstract/document/9743558. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Mengran Xia, Dawei Jin, and Jingyu Chen. Short-Term Traffic Flow Prediction Based on Graph Convolutional Networks and Federated Learning. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1191–1203, January 2023. ISSN 1558-0016. doi: 10.1109/TITS.2022.3179391. URL https://ieeexplore.ieee.org/abstract/document/9794333. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *arXiv preprint arXiv:2101.11203*, 2021.

Rui Ye, Rui Ge, Xinyu Zhu, Jingyi Chai, Yaxin Du, Yang Liu, Yanfeng Wang, and Siheng Chen. Fedllm-bench: Realistic benchmarks for federated learning of large language models. *Advances in Neural Information Processing Systems*, 37:111106–111130, 2024.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

Jianqing Zhang, Xinghao Wu, Yanbing Zhou, Xiaoting Sun, Qiqi Cai, Yang Liu, Yang Hua, Zhenzhe Zheng, Jian Cao, and Qiang Yang. Htfllib: A comprehensive heterogeneous federated learning library and benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5900–5911, 2025.

Wenfeng Zhou, Guojiang Shen, Zhenzhen Zhao, Zhaolin Deng, Tao Tang, Xiangjie Kong, Amr Tolba, and Osama Alfarraj. A transformer-based approach for traffic prediction with fusion spatiotemporal attention. *Knowledge-Based Systems*, pp. 114466, 2025.

A Datasets Details

This appendix provides further details on the seven datasets included in MobiFLow. For each dataset, we describe its characteristics, the download and preprocessing pipeline, the associated predictive task, and the corresponding baseline model and loss function.

A.1 FedBologna

A.1.1 Description

FedBologna is derived from open traffic data released by the municipality of Bologna, Italy (Comune di Bologna, Progetto Open Data, 2025), and is distributed under the **Creative Commons Attribution 4.0 (CC BY 4.0) license**. It consists of vehicle-flow measurements collected by road sensors distributed across different areas of the city during 2025. In the raw release, the data are stored in Parquet format and organized at daily resolution, with 24 hourly traffic measurements recorded for each sensor and day.

The dataset is automatically downloaded from the original source through a dedicated script, which creates the required local directory structure, retrieves the Parquet file via streaming, and writes it atomically to avoid corruption in case of interrupted downloads. A download status file is also maintained to ensure a robust and reproducible setup process by preventing unnecessary repeated downloads. An optional debug mode allows inspecting a preview of the downloaded data.

FedBologna naturally supports both cross-device and cross-silo federated learning settings. In the cross-device scenario, each individual sensor (`id_uni`) is treated as a distinct client. In the cross-silo scenario, sensors are assigned to Bologna's administrative districts through a spatial join between sensor coordinates and a GeoJSON file containing the city's district boundaries, yielding geographically coherent silos that reflect spatial heterogeneity in traffic dynamics.

The preprocessing pipeline first converts the raw daily wide-format data into hourly time series and enriched with temporal features, including month, ISO week, day of month, day of week, and hour of day. Categorical attributes are encoded globally, while numerical features are scaled using a global min–max normalization. To avoid temporal leakage, the data are split chronologically using rolling time-aware cross-validation based on calendar weeks, with approximately 80%, 10%, and 10% of the weeks assigned to training, validation, and test sets, respectively. The hourly series is then transformed into a supervised forecasting problem using sliding windows of length `history` to predict the subsequent `horizon` future vehicle counts. In our experiments, we set `history`$= 3$ and `horizon`$= 1$, in line with common practice in related traffic forecasting work (Rocco Di Torrepadula et al., 2024); however, both values are configurable hyperparameters in the preprocessing pipeline, allowing different forecasting settings to be easily explored. In the cross-device setting, windows are generated independently for each sensor. In the cross-silo setting, windows are first constructed separately for each sensor and then aggregated within the corresponding district-level client, preserving temporal continuity while enabling geographically grounded federated silos.

A.1.2 TASK

The learning task is short-term traffic forecasting, where the goal is to predict future vehicle counts from recent observations. The input consists of multivariate time-series data combining traffic measurements with temporal and contextual features, and the problem is formulated as a regression task. More formally, given a sequence of past traffic measurements of length `history` for a specific sensor (or a group of sensors in the cross-silo setting), the model is required to predict the vehicle count in the subsequent `horizon` time steps.

A.1.3 BASELINE MODEL AND LOSS FUNCTION

While the Bologna dataset has been used in prior work for mobility analytics and simulation (e.g., (Napolitano et al., 2025)), to the best of our knowledge, it has not been employed for predictive tasks such as traffic forecasting, which naturally arise when dealing with time series of traffic data. For this reason, as a baseline model for FedBologna, we follow the guidelines provided in a recent survey on mobility forecasting (Rocco Di Torrepadula et al., 2024), which identifies recurrent neural networks (RNNs) as a standard choice for short-term forecasting due to their ability to capture temporal dependencies in sequential data. Specifically, we employ a three-layer LSTM that takes as input a multivariate time series of shape $(\texttt{batch_size}, \texttt{history}, n_{\text{features}})$, where each sequence contains the past `history` observations and their associated features. The three LSTM layers have a hidden size of 128, followed by a fully connected hidden layer of size 64. The model is trained with a learning rate of 10^{-3} and batch size 32, corresponding to the best hyperparameters selected on the pooled training data through Optuna-based hyperparameter optimization. The search space includes hidden sizes $\{64, 128, 256\}$, fully connected sizes $\{32, 64, 128\}$, learning rates $\{10^{-4}, 3{\cdot}10^{-4}, 10^{-3}\}$, and batch sizes $\{32, 64, 128\}$, with model selection based on validation RMSE. The model is trained with a regression loss and evaluated using RMSE.

A.2 FEDCOLLECTY

A.2.1 DESCRIPTION

FedCollecty is built on top of the Collecty dataset, a multimodal transport analytics collection released by Erdelić et al. (Erdelić et al., 2023) and distributed on Kaggle (`merdelic/dataset-for-multimodal-transport-analytics`). The Collecty data paper (Erdelić et al., 2023) is published as open access under the **Creative Commons Attribution 4.0 License (CC BY 4.0)**, and the data collection received informed consent from all participants, with anonymization of personal data as described by the original authors. It consists of inertial sensor recordings continuously sampled from smartphones carried by 27 users while travelling around the city of Zagreb using a variety of transport modes. In the raw release, the data are stored in Parquet format and organized at trip resolution, with each row containing synchronized accelerometer, gyroscope, and magnetometer measurements together with a manually annotated transport-mode label and metadata identifying the user and the trip. The dataset is automatically downloaded from Kaggle through a dedicated script that creates the required local directory structure, retrieves the archive via the Kaggle CLI, extracts it, and tracks progress through a download status file in order to

ensure a robust and reproducible setup process by preventing unnecessary repeated downloads. The pipeline is idempotent and self-recovers from partial downloads or missing files by inspecting the state of the filesystem against the persisted status. An optional debug/sample mode allows restricting preprocessing to the first two users for fast inspection runs. FedCollecty is exclusively used in the cross-device federated learning setting, where each individual user is treated as a distinct client; the cross-silo setting is not supported, as the dataset does not provide a meaningful higher-level grouping of users beyond their identity.

The preprocessing pipeline parses the raw Parquet file row-group by row-group to keep memory usage bounded, discards rows whose label is `Unknown`, and subsamples the original $\sim$10 ms-resolution stream by a configurable factor (set to 100 in our experiments, yielding an effective sampling step of approximately 1 s) before retaining nine sensor channels corresponding to the three axes of the linear accelerometer, gyroscope, and magnetometer. Sensor values are scaled using a global min–max normalization fitted exclusively on the training portion of all users to avoid leakage, and the fitted scaler is persisted alongside a label map and a deterministic user-to-client identifier mapping for reproducibility. To avoid leakage between training, validation, and test data, the split is performed at the trip level rather than at the row level: for each user, trips are randomly partitioned with approximately 70%, 15%, and 15% of the trips assigned to training, validation, and test sets, respectively, with a random seed derived from `split_id` so that multiple non-overlapping splits can be produced for cross-validation. The continuous per-trip stream is then transformed into a supervised classification problem using sliding windows of length `window_size` with 50% overlap, with each window assigned the dominant transport-mode label of its constituent samples. In our experiments, we set `window_size`$= 512$ samples (corresponding to approximately 8.5 minutes at the chosen subsampling rate), in line with common practice in related transport-mode classification work (Ibañez et al., 2025); however, this value is a configurable hyperparameter in the preprocessing pipeline, allowing different window settings to be easily explored. Users whose training split does not contain at least one full window are automatically excluded from the federation.

A.2.2 Task

The learning task is transport-mode classification, where the goal is to predict the means of transport used by an individual along a short observation window from the readings of the smartphone inertial sensors carried by the user. The input consists of multivariate time-series data combining linear accelerometer, gyroscope, and magnetometer measurements, and the problem is formulated as a multi-class classification task over eight categories: Car, Bus, Walking, Bicycle, Train, Tram, Running, and Electric Scooter. More formally, given a sliding window of length `window_size` containing nine synchronized inertial sensor channels for a specific user, the model is required to predict the transport-mode label associated with the window.

A.2.3 Baseline Model and Loss Function

As a baseline model for FedCollecty, we adopt the hierarchical CNN+LSTM architecture proposed by Ibañez et al. (Ibañez et al., 2025) in the MobilitApp framework, specifically designed for smartphone-based transport-mode classification from inertial sensors. The model takes as input a multivariate time series of shape `(batch_size, window_size, 9)` and processes it through three parallel sensor-specific trunks, one for each of the accelerometer, gyroscope, and magnetometer channel groups. Each trunk consists of four stacked one-dimensional convolutional blocks with 32, 64, 128, and 512 filters and kernel sizes 7, 5, 3, and 3, respectively, where each block applies batch normalization, ReLU activation, spatial dropout with rate 0.05, and max pooling with stride 2. After every convolutional block the intermediate feature map is fed through a dedicated skip-connected LSTM with `hidden_size` units, and the last hidden states of the four LSTMs of each trunk are concatenated. The resulting representations from the three trunks are then concatenated and passed through a fully connected head consisting of a linear layer with `fc_size` units followed by a ReLU activation, and a final linear projection to the eight class logits. The model is trained with `hidden_size`$= 128$, `fc_size`$= 128$, learning rate 10^{-3}, weight decay 10^{-4}, and batch size 32, corresponding to the best hyperparameters selected on the pooled training data through Optuna-based hyperparameter optimization with a Hyperband pruner. The search space includes hidden sizes $\{64, 128, 256\}$, fully connected sizes $\{128, 256, 512\}$, learning rates $\{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$, weight decays $\{10^{-4}, 10^{-3}, 10^{-2}\}$, and batch sizes $\{32, 64, 128\}$, with model selection based on validation macro F1 score. The model is trained with a cross-entropy

loss with label smoothing 0.1, as prescribed by the original paper, and evaluated using accuracy, macro F1, macro precision, and macro recall.

A.3 FEDCRAWDAD

A.3.1 DESCRIPTION

The CRAWDAD Cabspotting dataset (Piorkowski et al., 2022) is distributed as **Open Access** through IEEE DataPort under the CRAWDAD terms of use, which require user registration with a free IEEE account, prohibit any attempt to reidentify cab drivers or passengers, and mandate acknowledgement of the data source in publications.It is a collection of GPS traces recorded by approximately 500 taxi cabs operating in the San Francisco Bay Area over about one month. In the raw release, each cab is stored as a plain-text trace file in which every row contains a timestamped GPS reading (latitude, longitude) together with a binary flag indicating whether the vehicle is occupied.

The dataset is automatically downloaded from IEEE DataPort through a dedicated script that ensures a reproducible and idempotent setup by tracking progress in a status file and self-recovering from partial downloads. FedCrawdad is exclusively used in the cross-device setting, where each cab is treated as a distinct client, leading to 533 federated clients in our subset; the cross-silo setting is not supported, as the dataset provides no meaningful higher-level grouping of cabs. In each communication round, 30 active clients are sampled to participate in training.

The preprocessing pipeline closely follows Rossi et al. (Rossi et al., 2019): for each cab, GPS readings are sorted chronologically and contiguous *occupied* segments are reduced to (pick-up, drop-off) coordinate pairs, discarding segments shorter than a configurable minimum length. Consecutive trips whose gap between drop-off and the next pick-up exceeds three hours are split into separate work-shift subsequences to avoid bridging unrelated driving sessions. The resulting coordinates are normalized through a persisted global min–max scaler, which is later used to invert predictions back to the geographic domain at evaluation time. The data are split chronologically at the trip level following the original $65\%/15\%/20\%$ ratio for training, validation, and test; we additionally introduce a rolling window offset controlled by `split_id` that shifts the starting position of the train/val window inside the cab's trip sequence, enabling multiple non-overlapping splits for cross-validation while preserving chronological ordering. Each split is then transformed into a supervised forecasting problem using sliding windows of length $k = 8$, where each window contains the four most recent (pick-up, drop-off) pairs together with the current pick-up point, and the target is the corresponding drop-off coordinate, matching the input length used in the original work for the San Francisco subset. Cabs whose training split contains no valid window are automatically excluded.

A.3.2 TASK

The learning task is next-destination prediction, where the goal is to predict the geographic coordinates of the drop-off point of the current trip given the recent trip history of the cab and the location where the passenger has just been picked up. The input consists of a short sequence of geographic coordinates and the problem is formulated as a multivariate regression task with a two-dimensional output. More formally, given a sequence of $k = 8$ past spatial points encoding the four most recent (pick-up, drop-off) pairs of a specific cab and the current pick-up coordinate, the model is required to predict the (latitude, longitude) of the next drop-off.

A.3.3 BASELINE MODEL AND LOSS FUNCTION

As a baseline model for FedCrawdad, we adopt the LSTM with self-attention proposed by Rossi et al. (Rossi et al., 2019), originally introduced for next-destination prediction on CRAWDAD Cabspotting and on the Porto and Manhattan taxi datasets. The model takes as input the sequence of $k = 8$ normalized two-dimensional coordinates and first applies a temporal self-attention module that produces softmax weights over the time dimension and re-weights the input sequence accordingly. The re-weighted sequence is then passed through a single-layer LSTM with `hidden_size` units, followed by ReLU, dropout with rate `dropout`, and a fully connected head that maps the last hidden state to the two-dimensional drop-off prediction. With respect to the original paper, we omit the optional categorical embeddings (location cluster, bag-of-clusters, driver, hour, weekday,

day-type), as the corresponding annotations are not available in the CRAWDAD release; only the spatial sequence of (latitude, longitude) coordinates is used as input.

The best hyperparameters, selected on the pooled training data through Optuna-based optimization with a Hyperband pruner over the search space `hidden_size`$\in \{64, 128, 256\}$, dropout $\in \{0.3, 0.5\}$, learning rate $\in \{10^{-4}, 10^{-3}, 10^{-2}\}$, and batch size $\in \{32, 64, 128\}$, are `hidden_size`$= 64$, dropout 0.3, learning rate 10^{-2}, and batch size 64. The model is trained with mean squared error on the normalized coordinates and evaluated using the Error Distance Score (EDS), defined as the mean Haversine distance (in kilometres) between the predicted and ground-truth drop-off points after inverting the min–max normalization.

A.4 FEDMELBOURNE

A.4.1 DESCRIPTION

This dataset is derived from the Transport Victoria Open Data Portal, which provides access to open mobility data from the Victoria region in Australia, and is distributed under the **Creative Commons Attribution 4.0 (CC BY 4.0) license**. I. In particular, we consider the Melbourne train passenger-count dataset, which reports boardings, alightings, and load-related statistics for individual train services from 2018 to 2024 (Open Data, Transport Victoria Region, 2025). Within our benchmark, this dataset is used for a short-term passenger flow forecasting task, where the objective is to predict the number of passengers on board trains at future time steps.

The raw data are distributed as yearly CSV files, one for each year from 2018 to 2024, and are automatically downloaded from the official source through a dedicated script. The downloader creates the required local directory structure, retrieves each file via streamed HTTP requests, and supports resumable downloads through partial files and byte-range requests, making the acquisition process robust to unstable network connections. A download status file is maintained to track progress across years and to avoid unnecessary repeated downloads.

Since the dataset does not include a natural notion of individual data-collecting devices, we do not define a cross-device setting. Instead, we consider a cross-silo scenario in which clients are defined by the provided `Group` attribute, corresponding to service areas such as *City Loop*. Each distinct group is mapped to a numerical client identifier, yielding logical federated partitions that reflect geographically and operationally meaningful subdivisions of the rail network.

The preprocessing pipeline first enriches the raw records with temporal features derived from the scheduled arrival and departure times. From these timestamps, additional temporal features are extracted, including year, month, day, weekday, and a fine-grained time-slot index based on 3-minute intervals. The time gap between scheduled arrival and departure is also computed in seconds and used as an additional numerical feature. Categorical features are encoded through a global categorical encoder, while numerical features are normalized using a global min–max scaler fitted across the dataset.

Similarly to FedBologna, the data are split chronologically using rolling time-aware cross-validation based on calendar months, with approximately 80%, 10%, and 10% of the months assigned to training, validation, and test sets, respectively. Before constructing supervised samples, raw observations are aggregated within each client by train, day, and time slot. When multiple records fall into the same time slot, passenger-related quantities are summed, while the remaining attributes are taken from the last observation in temporal order. This produces a consistent multivariate time series for each train trajectory within each silo.

The forecasting task is then formulated as a supervised learning problem using sliding windows. In our experiments, as for the other regression tasks, we set `history`$= 3$ and `horizon`$= 1$, in line with common practice in related forecasting work. Sliding windows are generated independently for each train and day, and the resulting samples are finally aggregated at the client level according to the corresponding service-area group.

A.4.2 TASK

The associated learning task is short-term passenger-flow forecasting. More specifically, given a sequence of recent observations for a train service within a specific service-area group, the objective

is to predict the passenger load at the next future time step. In our formulation, the prediction target is `Passenger_Arrival_Load`, i.e., the number of passengers on board upon arrival at a scheduled stop, making the task a multivariate time-series regression problem.

A.4.3 Baseline and Loss Function

As a baseline model for this dataset, we adopt a bidirectional long short-term memory (BiLSTM) network, as done in a recent work on passenger demand prediction in Melbourne (Liyanage et al., 2022). The model takes as input a multivariate time series of shape $(\text{batch_size}, \texttt{history}, n_{\text{features}})$, where each sequence contains the past `history` observations together with the associated contextual covariates. The sequential encoder consists of a single bidirectional LSTM layer, whose output at the last time step is used as a compact representation of the recent temporal dynamics. This representation is then passed through a fully connected prediction head composed of a linear layer, a ReLU activation, and a final linear layer producing the forecast for the next `horizon` time steps.

In our experiments, the best-performing configuration uses a hidden size of 512 for the BiLSTM layer and a fully connected hidden dimension of 256, with learning rate 10^{-3} and batch size 32. These hyperparameters are selected on the pooled training data through Optuna-based hyperparameter optimization. The search space includes hidden sizes $\{64, 128, 256, 300, 512\}$, fully connected sizes $\{32, 64, 128, 256\}$, learning rates $\{10^{-4}, 3\cdot10^{-4}, 10^{-3}, 3\cdot10^{-3}, 10^{-2}, 10^{-1}\}$, and batch sizes $\{32, 64, 128\}$, while model selection is based on validation RMSE. Training is performed with a regression loss and evaluated using RMSE.

A.5 FedMETR-LA

A.5.1 Description

FedMetrLA is built on top of the publicly available METR-LA traffic speed dataset, which contains vehicle speed measurements collected from 207 loop-detector sensors deployed on highways of Los Angeles County and is distributed under the **MIT license**. The dataset is distributed on Kaggle (`annnnguyen/metr-la-dataset`) and consists of a single HDF5 file storing a wide-format matrix of speed readings (in mph) at a uniform sampling interval of 5 minutes, together with a precomputed sensor-level adjacency matrix.

The dataset is automatically downloaded from Kaggle through a dedicated script that creates the required local directory structure, retrieves the archive via the Kaggle CLI, extracts it, and tracks progress through a download status file in order to ensure a robust and reproducible setup process by preventing unnecessary repeated downloads. An optional debug/sample mode allows restricting preprocessing to the first 10 sensors for fast inspection runs. FedMetrLA is exclusively used in the cross-device federated learning setting, where each individual loop-detector sensor is treated as a distinct client; the cross-silo setting is not supported, as the dataset does not provide a meaningful higher-level grouping of sensors beyond their geographical coordinates.

The preprocessing pipeline parses the raw HDF5 matrix into a per-sensor time series and enriches each window with two temporal features, namely hour of day and day of week, recovered from the original timestamp index. Speed values are scaled using a global min–max normalization fitted exclusively on the training portion of all sensors to avoid temporal leakage, and the fitted scaler is persisted so that evaluation metrics can be reported on the original mph scale. To avoid temporal leakage, the data are split chronologically using a rolling time-aware cross-validation scheme based on absolute timestep indices, with approximately 70%, 15%, and 15% of the timesteps assigned to training, validation, and test sets, respectively; multiple non-overlapping validation/test windows can be produced in a single pass to support (k)-fold-style evaluation. The continuous series is then transformed into a supervised forecasting problem using sliding windows of length `history` to predict the subsequent `horizon` future speed values, with NaN-containing windows discarded. In our experiments, we set `history`$= 12$ (corresponding to one hour of past observations) and `horizon`$= 3$ (15 minutes ahead) however, both values are configurable hyperparameters in the preprocessing pipeline, allowing different forecasting settings to be easily explored. Windows are generated independently for each sensor.

A.5.2 TASK

The learning task is short-term traffic speed forecasting, where the goal is to predict the future speed evolution of an individual road sensor from its recent observations. The input consists of multivariate time-series data combining normalized speed measurements with temporal contextual features, and the problem is formulated as a regression task. More formally, given a sequence of past speed measurements of length `history` for a specific sensor, the model is required to predict the speed values in the subsequent `horizon` time steps.

A.5.3 BASELINE MODEL AND LOSS FUNCTION

As a baseline model for FedMetrLA, we adopt the GRIZZLY architecture proposed by Ounoughi (Ounoughi & Yahia, 2024), a sequence-to-sequence stacked bidirectional LSTM specifically designed for traffic speed forecasting on the METR-LA benchmark. The model takes as input a multivariate time series of shape $(\texttt{batch_size}, \texttt{history}, 3)$, where each timestep contains the normalized speed value together with the corresponding hour of day and day of week. Categorical temporal features are projected through dedicated learned embeddings of dimension 4 each, which are concatenated with the normalized speed channel before being fed into the recurrent backbone. The backbone consists of three stacked bidirectional LSTM layers with hidden sizes $(\texttt{hidden_size}, \texttt{fc_size}, \texttt{fc_size})$, with a sigmoid activation applied between successive layers as prescribed in the original paper for the METR-LA setting, followed by a fully connected output head that maps the last timestep representation to the `horizon` future predictions. The model is trained with a hidden size of 64, a fully connected size of 16, a learning rate of $(3 \cdot 10^{-4})$, and batch size 512, corresponding to the best hyperparameters selected on the pooled training data through Optuna-based hyperparameter optimization. The search space includes hidden sizes (32,64,128), fully connected sizes (16,32,64), learning rates $(10^{-4}, 3 \cdot 10^{-4}, 10^{-3})$, and batch sizes (32,64,128,256,512), with model selection based on validation RMSE. The model is trained with a regression loss (Mean Squared Error) and evaluated using RMSE, MAE, and MAPE computed on speed values inverted back to the original mph scale.

A.6 FEDSMARTMACEIÓ

A.6.1 DESCRIPTION

FedSmartMaceió is a novel dataset introduced in this work, built on urban traffic data collected by SmartVia in Maceió, Brazil. To our knowledge, this is the first release of these data as a federated learning benchmark. We make the dataset publicly available[2] alongside the benchmark to support reproducibility and research on federated traffic forecasting in real-world, underrepresented Latin American contexts. It is released under the **CC BY-NC 4.0** license.

The collection consists of vehicle-speed measurements continuously recorded by a network of 12 measurement locales across the city. In the raw release, the data are stored in tabular format and report, for each sensor and each timestamp, the average vehicle speed observed within a 5-minute aggregation slot, together with the geographic coordinates of the sensor; missing observations are explicitly marked with a sentinel value(-1) rather than dropped, so that temporal continuity is preserved.

The dataset is automatically downloaded from a remote storage location through a dedicated script that creates the required local directory structure, retrieves the archive, and tracks progress through a download status file in order to ensure a robust and reproducible setup process by preventing unnecessary repeated downloads. An optional debug mode allows inspecting a preview of the downloaded data. FedSmartMaceió naturally supports cross-device federated learning settings. In the cross-device scenario, each individual local (identified by its anonymized `HASH`) is treated as a distinct client, leading to 12 federated clients.

The preprocessing pipeline first rounds all timestamps to a uniform 5-minute grid and reshapes the raw observations into a per-sensor continuous time series indexed by slot of day; missing values are preserved in place using a sentinel so that downstream evaluation can ignore them. Speed values are

[2] `https://drive.google.com/file/d/1sOwfeTQp4RIFYZoF9dILjlvxYc2Tw0fq/view?usp=sharing`

scaled using a global min–max normalization fitted exclusively on the valid (non-missing) values of the training portion of all sensors to avoid temporal leakage, and the fitted scaler is persisted so that evaluation metrics can be reported on the original km/h scale. To avoid temporal leakage, the data are split chronologically using a rolling time-aware cross-validation scheme based on calendar days rather than absolute timesteps, with approximately 80%, 10%, and 10% of the days assigned to training, validation, and test sets, respectively; the train window has a fixed size and slides forward together with the validation and test windows so that earlier splits use older data and the final split uses the most recent days, while sliding windows never cross split boundaries.

A.6.2 TASK

The learning task is short-term traffic speed forecasting, where the goal is to predict future vehicle speeds from recent observations. The input consists of multivariate time-series data combining speed measurements with temporal contextual features, and the problem is formulated as a regression task. More formally, given a sequence of past speed measurements of length `history` for a specific sensor (or a group of sensors in the cross-silo setting), the model is required to predict the speed values in the subsequent `horizon` time steps.

A.6.3 BASELINE MODEL AND LOSS FUNCTION

As a baseline model for FedSmartMaceió, we adopt LGSTformer-T, a temporal-only adaptation of the LGSTformer architecture proposed by Zhou et al. (Zhou et al., 2025), a recent Transformer-based traffic forecasting model that fuses spatial and temporal attention through a local–global gating mechanism.

The model takes as input a tensor of shape `(batch_size, history, 3)`, where each timestep contains the normalized speed, the slot of day (vocabulary 288), and the day of week (vocabulary 7). The two categorical features are mapped through learnable embeddings while the speed channel is linearly projected; the three resulting representations of dimension `embed_dim` are concatenated, yielding an input of dimension $d_{\text{model}} = 3 \cdot \texttt{embed_dim}$. The backbone consists of `num_layers` stacked LGTM blocks, each fusing (i) a local temporal representation produced by a multi-scale gated tanh unit (M-GTU) with parallel causal convolutions of kernel sizes 3, 5, and 7, and (ii) a global temporal representation produced by multi-head self-attention along the time axis with `num_heads` heads. A dual-path adaptive gate learns instance-dependent mixing coefficients between the two pathways, followed by residual connections, layer normalization, and a feed-forward network of hidden dimension $4 \cdot d_{\text{model}}$. A linear head maps the last-timestep representation to the `horizon` future speed values.

The best hyperparameters, selected on the pooled training data through Optuna-based optimization with a Hyperband pruner over the search space `embed_dim`$\in \{16, 24, 32\}$, `num_layers`$\in \{2, 3\}$, `num_heads`$\in \{4, 8\}$, dropout $\in \{0.0, 0.1, 0.2\}$, learning rate $\in \{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}\}$, and batch size $\in \{32, 64, 128\}$, are `embed_dim`$= 24$ ($d_{\text{model}} = 72$), `num_layers`$= 3$, `num_heads`$= 4$, dropout 0, learning rate 10^{-3}, and batch size 64. The model is trained with mean squared error and evaluated using RMSE, MAE, and MAPE on the original km/h scale, with metrics additionally reported at the 15, 30, and 60 minute horizon checkpoints commonly used in the traffic forecasting literature.

A.7 FEDSUNT

A.7.1 DESCRIPTION

FedSUNT is derived from the Salvador Urban Network Transportation (SUNT) dataset (Ferreira et al., 2025), a large-scale public transport dataset integrating automated fare collection (AFC), vehicle location (AVL), and GTFS data for bus and subway systems in Salvador, Brazil. The dataset covers the period from March 2024 to March 2025 and includes data from approximately 2,000 vehicles operating across nearly 400 lines and 3,000 stops.

The dataset is distributed under the **Creative Commons Attribution 4.0 (CC BY 4.0) license** as a compressed archive and is automatically downloaded and extracted through a dedicated script with filesystem-based status tracking, ensuring a robust and reproducible setup procedure. Within MobiFLow, we consider a short-term passenger flow forecasting task, where the goal is to predict

the number of passenger boardings at upcoming time steps. The dataset supports both cross-device and cross-silo federated settings. In the cross-device scenario, each vehicle is treated as an individual client. In the cross-silo setting, routes are clustered geographically according to their GTFS-based spatial trajectories, and each cluster is treated as a client, simulating operators managing services in homogeneous areas.

Preprocessing first requires integrating heterogeneous data sources. Specifically, daily AFC records are matched with AVL vehicle-position data using a time-based nearest-neighbor join within a fixed tolerance, so that each ticket-validation event is associated with the most recent vehicle observation on the same route and vehicle. The resulting data are further enriched with GTFS metadata, such as operator and vehicle type, and stored as monthly Parquet files to ensure scalability. Temporal features are then extracted from the event timestamp, including month, day, weekday, a 5-minute time-slot index, and an ISO week identifier used for time-aware splitting. Categorical features are encoded using a global categorical encoder, while numerical variables are normalized via min–max scaling fitted across the full dataset.

The data are split chronologically using rolling time-aware cross-validation based on calendar weeks, with approximately 80%, 10%, and 10% of the weeks assigned to training, validation, and test sets, respectively. Before constructing supervised samples, events are aggregated by vehicle, day, and time slot, summing boardings and retaining the last valid observation for the remaining attributes within each slot. Forecasting samples are finally constructed via sliding windows with `history`$= 3$ and `horizon`$= 1$. In the cross-device setting, windows are generated independently for each vehicle. In the cross-silo setting, windows are built per vehicle and then assigned to clients according to the corresponding route cluster.

A.7.2 Task

The associated learning task is short-term passenger-flow forecasting. More specifically, given a sequence of recent observations for a vehicle, the objective is to predict the number of passenger boardings at the next future time step. In our formulation, the prediction target is the aggregated number of `boardings` within a time slot, making the task a multivariate time-series regression problem.

A.7.3 Baseline and Loss Function

As a baseline model for FedSUNT, we adopt a single-layer LSTM network, as employed in the original dataset paper (Ferreira et al., 2025). The model takes as input a multivariate time series of shape $(\text{batch_size}, \texttt{history}, n_{\text{features}})$, where each sequence contains the past `history` observations together with the associated contextual covariates. The hidden representation at the last time step is then passed to a fully connected prediction head composed of a linear layer, a ReLU activation, and a final linear layer that outputs the forecast for the next `horizon` time steps.

In our experiments, the best-performing configuration uses an LSTM hidden size of 256, a fully connected hidden dimension of 128, learning rate 10^{-4}, and batch size 32. These hyperparameters are selected on the pooled training data through Optuna-based hyperparameter optimization. The search space includes hidden sizes $\{64, 128, 256\}$, fully connected sizes $\{32, 64, 128\}$, learning rates $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, and batch sizes $\{32, 64, 128\}$, with model selection based on validation RMSE. During training, gradient clipping with maximum norm 1.0 is applied to improve optimization stability. The model is trained with a regression loss and evaluated using RMSE.

B Experimental Details

As described in Section 4.2, we first optimize the hyperparameters of the baseline model on the pooled training data for each dataset. Subsequently, for each FL strategy, we fix these model hyperparameters and restrict the optimization to FL-specific hyperparameters, following prior work (Ogier du Terrail et al., 2022). Table 3 summarizes the hyperparameter search space considered for the FL strategies, while Table 4 reports the best configurations obtained for each strategy and dataset. Specifically, we tune the proximal coefficient μ for FedProx, which controls the strength of regularization toward the global model and mitigates client drift under data heterogeneity. For Cyclic training, we tune the `learning_rate_scaler`, which adjusts the magnitude of client up-

dates across rounds. For adaptive server-based methods (FedAdam, FedYogi, and FedAdagrad) and Scaffold, we tune the `server_learning_rate`, which governs the contribution of aggregated updates to the global model, trading off convergence speed and stability.

Table 3: Hyperparameter search space for the considered FL methods.

Method	**Hyperparameter**	**Tested values**
FedProx	μ	$\{10^{-5}, 3{\cdot}10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}, 3{\cdot}10^{-2}, 10^{-1}\}$
Cyclic	learning_rate_scaler	$\{0.05, 0.1, 0.2, 0.3, 0.5, 1.0, 2.0, 5.0\}$
FedAdam FedYogi FedAdagrad Scaffold	server_learning_rate	$\{10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}, 10^{-2}, 3{\cdot}10^{-2}, 10^{-1}, 3{\cdot}10^{-1}, 1.0\}$

Table 4: Best FL-specific hyperparameters across datasets and federation settings. For the sake of conciseness, in the table "learning rate" is abbreviated "lr"

Dataset	**Federation**	**FedProx** (μ)	**Cyclic** (lr scaler)	**Scaffold** (server lr)	**FedAdam** (server lr)	**FedYogi** (server lr)	**FedAdagrad** (server lr)
FedBologna	Cross-device	10^{-5}	0.1	10^{-4}	10^{-2}	$3 \cdot 10^{-3}$	$3 \cdot 10^{-2}$
FedBologna	Cross-silo	10^{-5}	1.0	10^{-4}	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	10^{-2}
FedCollecty	Cross-device	10^{-4}	0.05	10^{-3}	10^{-3}	10^{-3}	10^{-3}
FedCrawdad	Cross-device	10^{-4}	10.0	10^{-1}	10^{-2}	10^{-2}	10^{-1}
FedMelbourne	Cross-silo	10^{-5}	1.0	$3 \cdot 10^{-1}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
FedMETR-LA	Cross-device	10^{-5}	0.05	$3 \cdot 10^{-2}$	10^{-3}	10^{-3}	10^{-4}
FedSmartMaceió	Cross-device	10^{-1}	0.05	1.0	10^{-3}	10^{-3}	10^{-2}
FedSUNT	Cross-device	10^{-5}	0.1	10^{-4}	10^{-4}	10^{-4}	$3 \cdot 10^{-3}$
FedSUNT	Cross-silo	10^{-5}	0.3	10^{-4}	10^{-4}	10^{-4}	10^{-3}

C Extended Results

An overview of the benchmarking results obtained with MobiFLow is presented in Figure 2 in the form of a rank matrix. The main empirical finding is that, under realistic mobility heterogeneity, collaboration is not uniformly beneficial: local training often emerges as the strongest baseline, pooled training remains competitive when client distributions are relatively homogeneous, and no single FL strategy consistently dominates across datasets. This appendix provides additional details on the results for each dataset. In particular, Figures 3–8 report bar plots of the evaluation metrics across strategies in green and purple for cross-device and cross-silo settings, respectively [3]. Moreover, Figures 9–15 present the outcomes of the statistical tests used to assess whether performance differences among methods are significant. Specifically, we perform paired statistical tests across runs, using paired t-tests at a significance level of $\alpha = 0.05$, and apply the Holm–Bonferroni correction within each dataset, FL setting, metric, and comparison scope to account for multiple pairwise comparisons. In the figures, statistically significant differences after Holm–Bonferroni correction are denoted by ✓, while non-significant differences are indicated by $\times$. In addition, the figures visualize pairwise effect sizes using Cohen's d_z (Cohen, 1988), i.e., the mean paired difference divided by the standard deviation of the paired differences, through a color scale ranging from red to blue. Blue shades indicate that the row method outperforms the column method, while red shades indicate the opposite; the color intensity reflects the magnitude of the effect size.

[3] Results for Scaffold on the FedCrawdad dataset are omitted, as they exhibit significantly higher error than the other methods and would reduce the readability of the figure; this behavior may be indicative of instability or divergence under the considered hyperparameter configuration for this dataset.

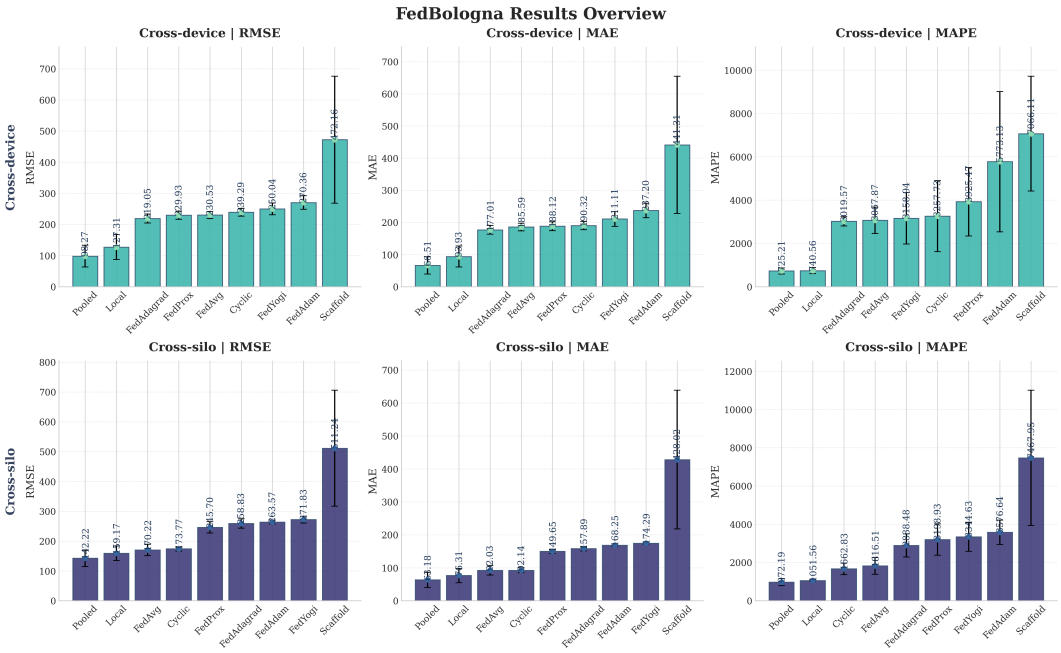


Figure 3: Bar plots of RMSE, MAE, and MAPE for the FedBologna dataset under cross-device (green) and cross-silo (purple) settings.

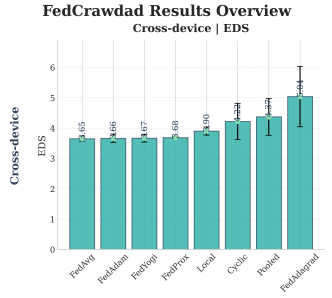


Figure 4: Bar plots of EDS for the FedCrawdad dataset under cross-device setting.

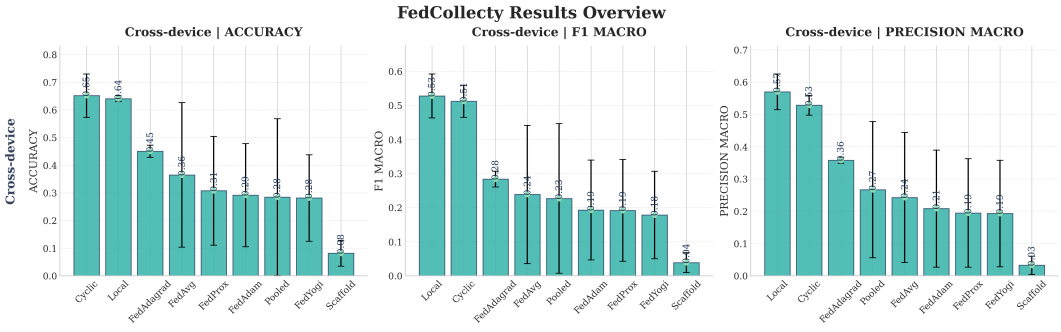


Figure 5: Bar plots of accuracy, F1 macro, recall macro, and precision macro for the FedCollecty dataset under cross-device setting.

C.1 GRAPH-BASED BASELINES

Several studies have demonstrated that graph neural networks can improve time-series forecasting by exploiting spatial relationships among locations to better model temporal dynamics (Rocco

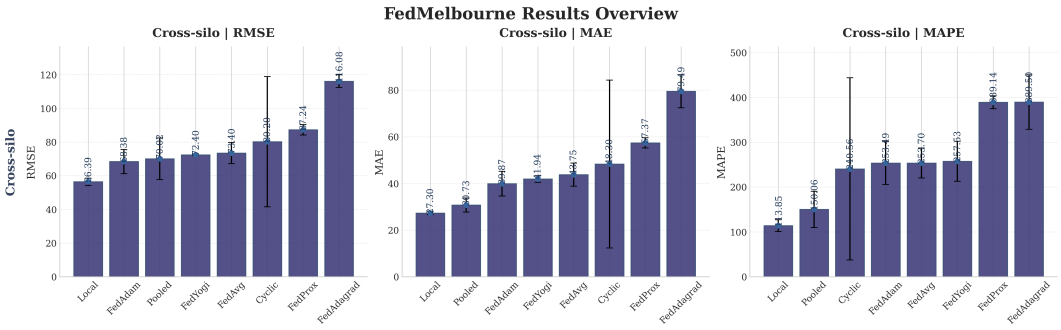


Figure 6: Bar plots of RMSE, MAE, and MAPE for the FedMelbourne dataset under cross-silo setting.

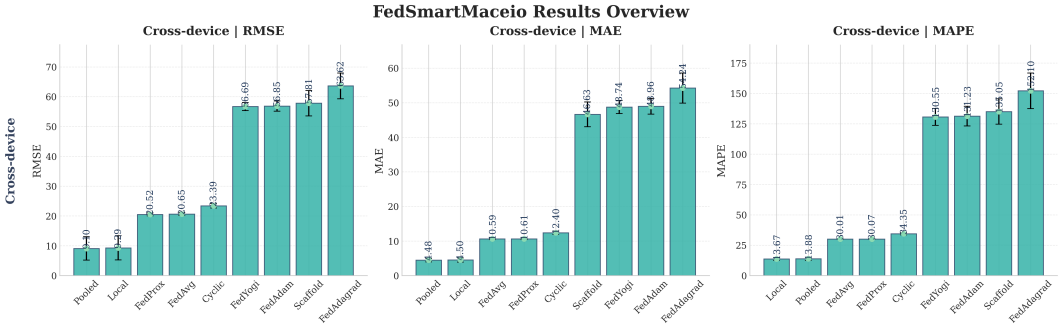


Figure 7: Bar plots of RMSE, MAE, and MAPE for the FedSmartMaceio dataset under cross-device setting.

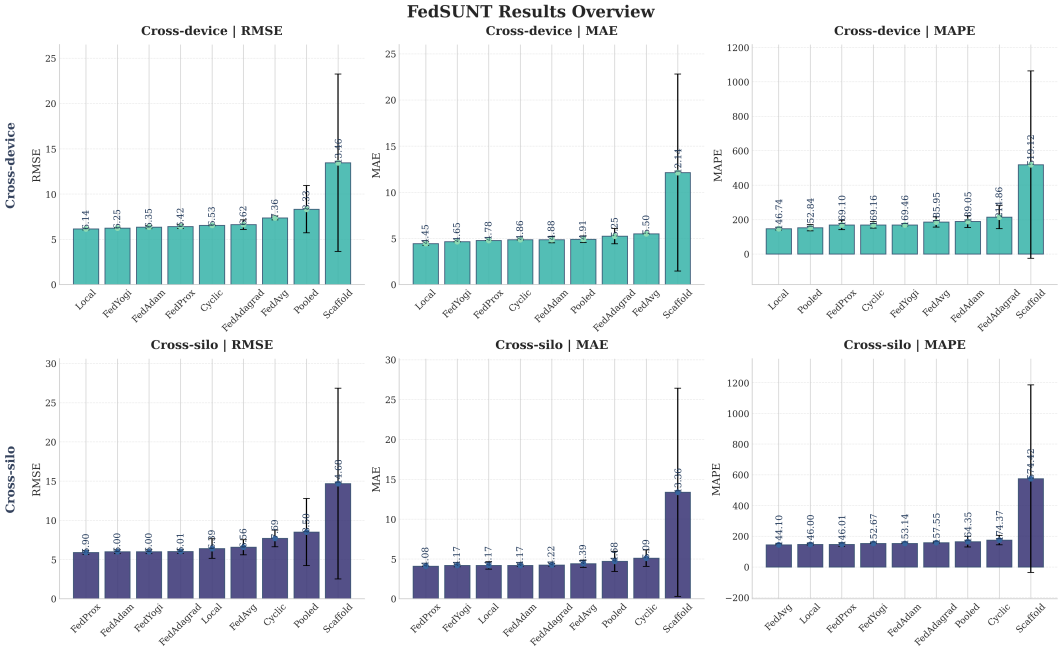


Figure 8: Bar plots of RMSE, MAE, and MAPE for the FedSUNT dataset under cross-device (green) and cross-silo (purple) settings.

Di Torrepadula et al., 2024). Building on these findings and the modular design of MobiFLow,

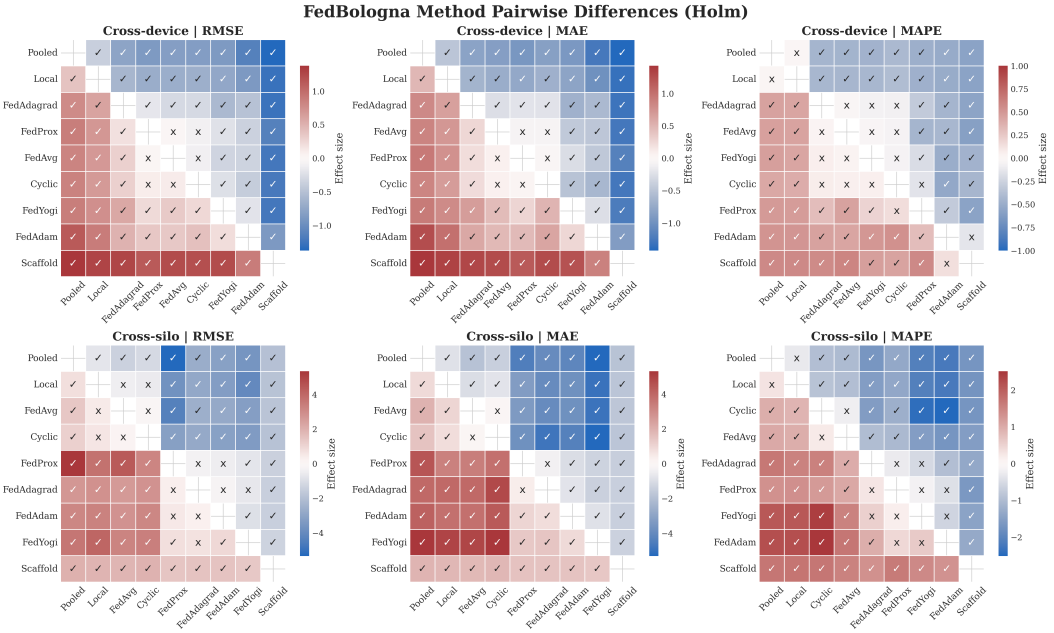


Figure 9: Pairwise comparison of methods on the FedBologna dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

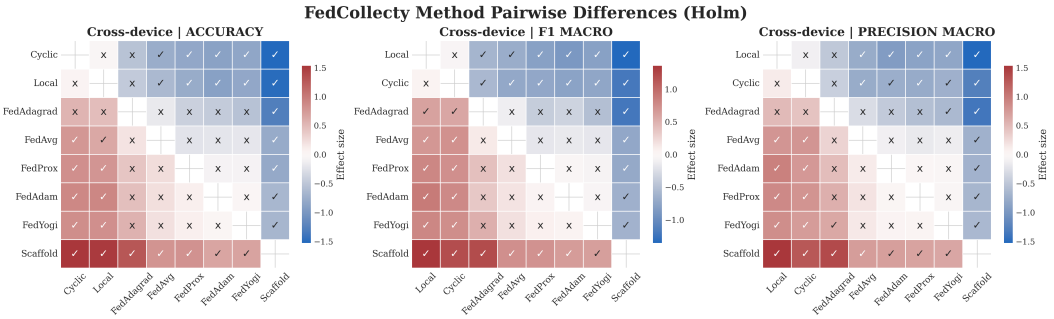


Figure 10: Pairwise comparison of methods on the FedCollecty dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

which facilitates the integration of new federated learning methods, we introduce two GNN-based baselines tailored to the SmartMaceió and Bologna datasets.

C.1.1 SmartMaceió results

Table 5 summarizes the comparative results on the SmartMaceió dataset. The Bi-LSTM achieves the lowest RMSE under local and pooled training, whereas the ST-GNN and LGSTformer outperform it under most federated strategies. Under FedAvg, for example, the ST-GNN and LGSTformer achieve RMSE values of 7.92 and 7.95, respectively, compared with 11.21 for the Bi-LSTM. SCAFFOLD is an exception: the Bi-LSTM obtains the lowest RMSE, although all three models exhibit substantially higher errors under this strategy. Local training yields the lowest RMSE for each architecture, indicating that the improvements over the Bi-LSTM under most federated strategies do not translate into an advantage over local training.

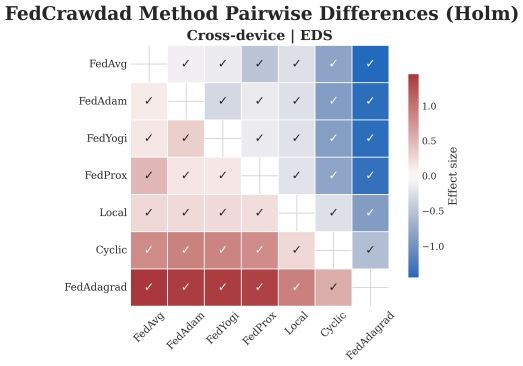


Figure 11: Pairwise comparison of methods on the FedCrawdad dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

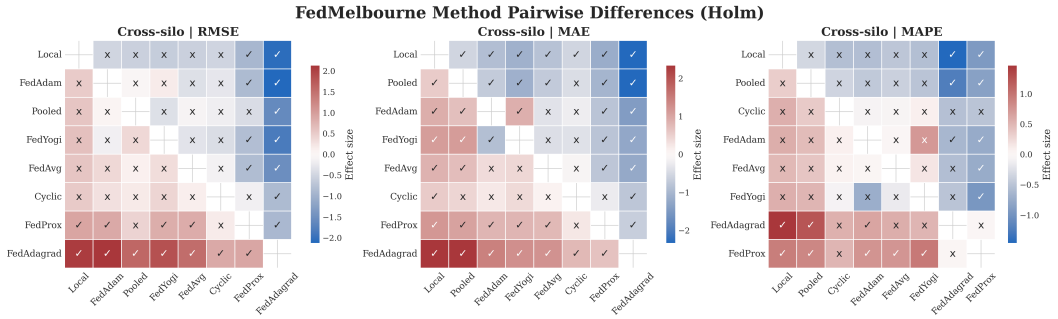


Figure 12: Pairwise comparison of methods on the FedMelbourne dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

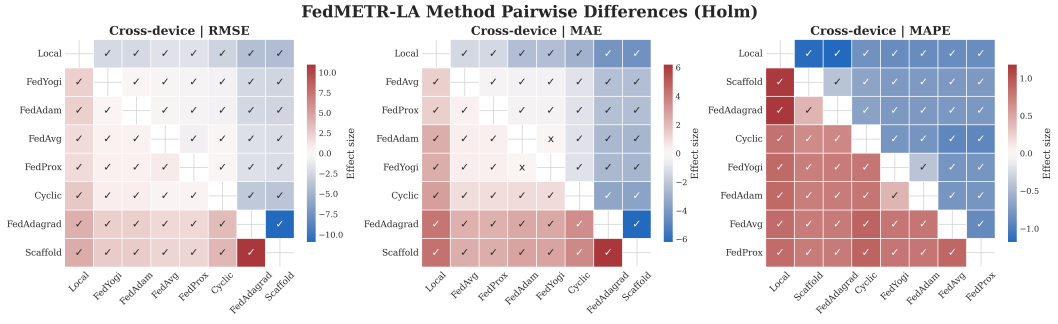


Figure 13: Pairwise comparison of methods on the FedMETR-LA dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

C.1.2 Bologna results across forecasting horizons

Previous studies have suggested that the benefits of graph-based models may become more evident at longer forecasting horizons, when spatial dependencies among sensors can play a greater role (Li et al., 2018). We therefore conducted additional experiments on the Bologna dataset with forecasting horizons of $h \in \{1, 6, 8, 10, 12\}$.

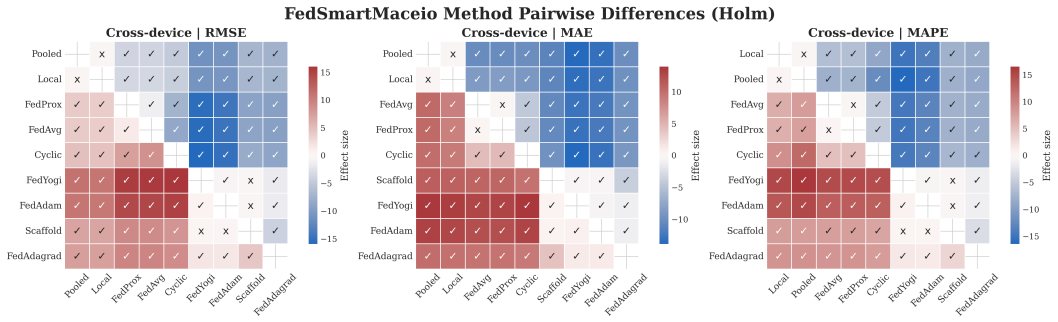


Figure 14: Pairwise comparison of methods on the FedSmartMaceio dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

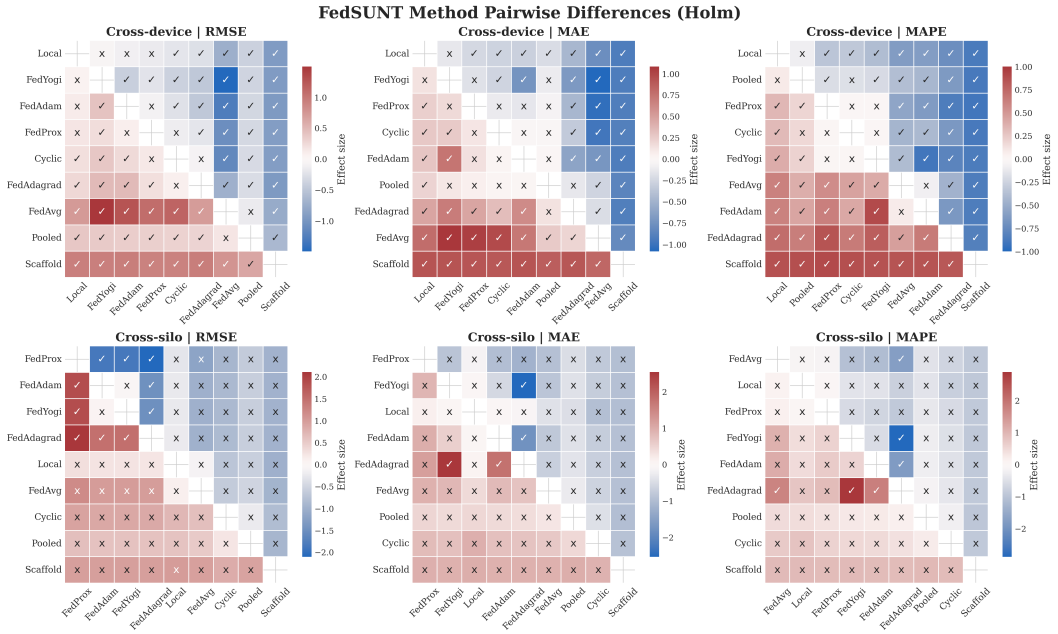


Figure 15: Pairwise comparison of methods on the FedSUNT dataset. Cells report paired t-test results ($\alpha = 0.05$), with ✓ indicating significant differences and × otherwise. Colors encode effect sizes: blue (row outperforms column) and red (vice versa).

Table 6 reports the absolute test RMSE, and Table 7 presents the corresponding relative differences, calculated as

$$\Delta(\%) = 100 \times \frac{\mathrm{RMSE}_{\mathrm{GNN}} - \mathrm{RMSE}_{\mathrm{LSTM}}}{\mathrm{RMSE}_{\mathrm{LSTM}}}. \tag{2}$$

Negative values indicate lower RMSE for the GNN.

The comparison between $h = 1$ and $h \geq 6$ should be interpreted cautiously because increasing the horizon also changes the forecasting task. At $h = 1$, the per-sensor LSTM substantially outperforms the GNN under Local and FedAvg training, with smaller differences under FedProx and a slight advantage for the GNN under FedYogi. One possible explanation is that next-step traffic depends strongly on each sensor's recent temporal dynamics, limiting the additional benefit of spatial information.

For horizons of six steps or more, the GNN becomes competitive under Local and federated training. Its relative performance remains broadly stable from $h = 6$ to $h = 12$, with RMSE reductions of approximately 6–8% under Local training and 3–4% under FedProx. Under FedAvg, the LSTM

Table 5: RMSE on the SmartMaceió dataset. Lower values indicate better performance. The best result in each row is highlighted in bold.

Strategy	ST-GNN	LGSTformer	Bi-LSTM
Local	5.06	4.82	**4.69**
Pooled	5.97	5.27	**5.13**
FedAvg	**7.92**	7.95	11.21
FedProx	**7.90**	8.01	11.20
Cyclic	8.15	**7.67**	11.18
FedAdagrad	8.39	**8.04**	11.25
FedAdam	8.75	**8.73**	11.22
FedYogi	8.83	**8.61**	11.18
SCAFFOLD	36.33	45.96	**32.11**

Table 6: Test RMSE on the Bologna dataset under the reduced training budget. Each cell reports LSTM / GNN. The lower RMSE within each pair is highlighted in bold.

Strategy	$h = 1$	$h = 6$	$h = 8$	$h = 10$	$h = 12$
Local	**187.1** / 268.1	220.7 / **204.0**	223.7 / **208.0**	222.2 / **208.1**	226.7 / **210.4**
FedAvg	**191.8** / 276.0	**264.3** / 267.1	**263.9** / 269.1	**261.2** / 269.1	**264.8** / 270.4
FedProx	**267.2** / 279.2	297.0 / **288.0**	300.1 / **287.1**	297.0 / **287.6**	302.0 / **291.4**
FedYogi	282.0 / **276.1**	326.6 / **295.1**	303.9 / **291.4**	**307.8** / 322.0	333.6 / **306.8**

Table 7: Relative RMSE differences (Δ, in percent) between the GNN and LSTM on Bologna, calculated from Table 6. Negative values favor the GNN.

Strategy	$h = 1$	$h = 6$	$h = 8$	$h = 10$	$h = 12$
Local	$+43.3$	-7.6	-7.0	-6.3	-7.2
FedAvg	$+43.9$	$+1.1$	$+2.0$	$+3.0$	$+2.1$
FedProx	$+4.5$	-3.0	-4.3	-3.2	-3.5
FedYogi	-2.1	-9.6	-4.1	$+4.6$	-8.0

retains a small advantage of approximately 1–3%. Thus, extending the horizon beyond six steps does not consistently increase the GNN's relative advantage.

The results also depend on the training strategy and optimization budget. Under the reduced budget used in these additional experiments, the GNN outperforms the LSTM under Local training at $h \geq 6$. In contrast, the LSTM performs better with the larger budget used in the main evaluation. This suggests that the observed advantage of the GNN may partly reflect differences in convergence speed rather than better performance after full training. Similarly, although the models remain close under FedAvg at $h \geq 6$ in the reduced-budget experiments, the fully trained $h = 6$ result favors the LSTM, with an RMSE of 247.3 compared with 265.0 for the GNN.

Under FedProx, the GNN consistently achieves lower RMSE at $h \geq 6$, with values between 287.1 and 291.4, compared with 297.0–302.0 for the LSTM. Under FedYogi, the GNN outperforms the LSTM at four of the five horizons, but the advantage is not consistent: at $h = 10$, its RMSE increases to 322.0, compared with 307.8 for the LSTM. Since the FedYogi experiments used only ten communication rounds, these results should be treated as preliminary and individual values should not be overinterpreted.

D Sensitivity Analysis

D.1 Experimental setup

We conducted a sensitivity analysis on FedBologna using FedAvg in both cross-device and cross-silo settings to examine the impact of three factors: the number of communication rounds, the number of local update steps per participating client, and client participation.

For communication rounds, we evaluated 100, 200, and 400 rounds, corresponding to $0.5\times$, $1\times$, and $2\times$ the reference configuration. The number of local updates was adjusted inversely with the number of rounds to maintain approximately the same total optimization budget. This analysis therefore examines the allocation of a fixed budget between communication rounds and local computation. For local updates, we evaluated $0.5\times$, $1\times$, and $1.5\times$ the reference value: 353, 706, and 1,059 updates in the cross-device setting, and 2,944, 5,888, and 8,832 updates in the cross-silo setting. Unlike the communication-round analysis, this experiment varies the total optimization budget while keeping the remaining settings fixed. For client participation, we considered 25, 50, and 75 participating clients per round in the cross-device setting, and 2, 4, and 6 in the cross-silo setting. The initial participation experiment kept the number of local updates per client fixed. Its results therefore reflect changes in both participation and total computation, motivating the controlled experiment described below.

D.2 Communication rounds and local updates

Table 8 summarizes the sensitivity to communication rounds and local updates. In the cross-device setting, RMSE changed only modestly across the tested configurations. Increasing the number of rounds from 100 to 400 reduced RMSE from 239.10 to 234.61. Increasing local updates from 353 to 1,059 similarly reduced RMSE from 238.00 to 234.42.

The cross-silo setting was more sensitive to the allocation of the optimization budget. Increasing communication rounds from 100 to 200 reduced RMSE from 209.44 to 195.52, whereas increasing to 400 rounds provided only a small additional improvement, reaching 194.94. Under the approximately fixed total budget, these results suggest diminishing returns from more frequent communication beyond 200 rounds. The effect of local updates was non-monotonic in the cross-silo setting. The reference configuration of 5,888 updates achieved the lowest RMSE among the tested values, at 195.13, compared with 198.37 for 2,944 updates and 202.80 for 8,832 updates. Thus, increasing local computation did not necessarily improve predictive performance.

Table 8: Sensitivity to communication rounds and local updates on FedBologna using FedAvg.

Setting	Factor	Configuration / RMSE		
Cross-device	Communication rounds	100 / 239.10	200 / 235.52	400 / **234.61**
Cross-device	Local updates	353 / 238.00	706 / 235.61	1,059 / **234.42**
Cross-silo	Communication rounds	100 / 209.44	200 / 195.52	400 / **194.94**
Cross-silo	Local updates	2,944 / 198.37	5,888 / **195.13**	8,832 / 202.80

D.3 Client participation and optimization budget

D.3.1 Participation with fixed local updates

Table 9 reports the initial participation results. Cross-device performance remained nearly unchanged, with RMSE values of 234.39, 234.29, and 234.45 for 25, 50, and 75 participating clients, respectively. In contrast, cross-silo RMSE decreased from 218.41 with two clients to 209.05 with four clients and 189.55 with all six clients.

However, these cross-silo results do not isolate the effect of participation. Because each participating client performed 5,888 local updates, increasing participation from two to six clients also increased the total number of optimization steps per round from 11,776 to 35,328. The observed improvement may therefore reflect both broader client coverage and a larger optimization budget. Although partial participation excludes a larger fraction of the six-client federation at each round, its contribution cannot be separated from that of additional computation in this experiment.

Table 9: Client-participation sensitivity with fixed local updates per client. The total optimization budget increases with participation.

Setting	Participating clients	RMSE
Cross-device	25	234.39
	50	234.29
	75	234.45
Cross-silo	2	218.41
	4	209.05
	6	189.55

D.3.2 Participation under a fixed optimization budget

To assess participation under an equivalent optimization budget, we designed an additional cross-silo experiment in which local updates per client are scaled inversely with the number of participating clients. Let K denote the number of participating clients and U the number of local updates per client. The total number of optimization steps per communication round is

$$B_{\text{round}} = KU. \tag{3}$$

We fix $B_{\text{round}} = 35{,}328$ and keep the number of communication rounds unchanged across configurations. Consequently, both the per-round and total optimization-step budgets remain constant. Table 10 contrasts this protocol with the initial experiment. Table 11 reports the results.

Under a fixed optimization budget, RMSE decreases monotonically as participation increases from two to six clients, falling from 212.26 to 194.44 in Run 1 and from 214.53 to 195.88 in Run 2, with differences below 2.3 RMSE between runs. Equalizing the budget narrows the gap between partial and full participation compared with the original analysis, suggesting that additional computation contributed to the previously observed improvement. Nevertheless, the remaining trend supports a benefit from broader client coverage that additional local updates alone do not fully recover. Since participation and local updates per client vary jointly, and only two runs were performed, these results indicate a consistent trend rather than a precise separation of participation and computation effects.

Table 10: Cross-silo participation protocols. The controlled experiment preserves 35,328 optimization steps per round by adjusting local updates per client.

	Initial experiment		Fixed-budget experiment	
Clients	Updates/client	Steps/round	Updates/client	Steps/round
2	5,888	11,776	17,664	35,328
4	5,888	23,552	8,832	35,328
6	5,888	35,328	5,888	35,328

Table 11: Cross-silo client-participation analysis on FedBologna using FedAvg under a fixed optimization budget.

Active clients (k)	Local updates per client	Steps per round	RMSE (Run 1)	RMSE (Run 2)
2	17,664	35,328	212.26	214.53
4	8,832	35,328	204.11	206.30
6	5,888	35,328	**194.44**	**195.88**

E Computational and Communication Resources

E.1 Computational resources

Table 12: Average and maximum GPU hours by method

Method	**Average #hours**	**Max #hours**
Pooled	174.35	796.74
Local	13.72	123.11
FedProx	8.76	38.70
Cyclic	5.07	17.35
FedAvg	4.73	21.76
FedAdam	4.12	18.82
Scaffold	3.94	18.08
FedYogi	3.88	15.73
FedAdagrad	3.84	18.75

All experiments in Section 4 were run on a Debian 6.12.74 server with an NVIDIA A100 GPU, using Python 3.11 and PyTorch 2.11.0. Table 12 reports the average and maximum GPU hours for pooled, local, and federated baselines, sorted by method. Pooled training is the most computationally expensive, as it processes the full dataset centrally, resulting in the largest number of sequential gradient updates. Local training is faster than pooled, since each client operates on a smaller data partition. Federated methods are more efficient overall, as the fixed training budget is distributed across clients and rounds, reducing the number of updates per client while avoiding redundant full training (see Section 4.3). All experiments are executed sequentially on a single node. As a consequence, the reported GPU hours characterize computational requirements but do not measure network transfer or latency. We therefore complement these analyses with an analytical estimate of communication volumne in Section E.2. Among federated strategies, FedProx incurs slightly higher cost due to the additional proximal term, which increases the complexity of local optimization, whereas simpler approaches such as FedAvg and its adaptive variants introduce lower overhead.

To complement this analysis, Table 13 reports the average GPU hours per dataset under different federated settings. As expected, larger datasets (Section 3.4) incur higher computational cost. Notably, despite a fixed number of communication rounds and comparable update budgets, cross-device configurations are slower than cross-silo ones (e.g., FedSUNT and FedBologna). This behavior is explained by the larger number of participating clients per round in the cross-device setting. Since all clients are trained sequentially on a single GPU in our experiments, this results in higher per-client overhead, including model copying, dataloader initialization, optimizer setup, and aggregation. Fur-

Table 13: Average GPU hours per run by dataset and setting

Dataset	Setting	Average #hours per run
FedSUNT	cross-device	889.93
FedSUNT	cross-silo	667.08
FedBologna	cross-device	494.20
FeMelbourne	cross-silo	114.86
FedBologna	cross-silo	36.44
FedSmartmaceiò	cross-device	14.18
FedMETR-LA	cross-device	3.73
FedCrawdad	cross-device	0.44
FedCollecty	cross-device	0.27

Dataset	Federation	Active clients	Model parameters	GB/run	GB/strategy, 3 splits
FedCollecty	Cross-device	9	2.83M	40.75	122.25
FedMelbourne	Cross-silo	10	2.46M	39.33	117.99
FedBologna	Cross-device	50	349K	27.95	83.84
FedSUNT	Cross-device	30	314K	15.05	45.16
FedSmartMaceió	Cross-device	10	519K	8.30	24.89
FedBologna	Cross-silo	6	349K	3.35	10.06
FedSUNT	Cross-silo	5	314K	2.51	7.53
FedMETR-LA	Cross-device	10	64K	1.02	3.06
FedCrawdad	Cross-device	30	18K	0.85	2.54
Total: all datasets, all seven strategies, three splits					**2,921.1**

Table 14: Estimated communication volume for the federated benchmark. The last column reports the cumulative volume over the three evaluation splits for one federated strategy. The final row aggregates all seven federated strategies across all dataset/federation settings and three splits.

thermore, the smaller and more fragmented workloads in cross-device settings lead to less efficient GPU utilization compared to cross-silo scenarios, where fewer clients perform larger local updates.

E.2 Communication volume

Since the experiments emulate federated learning on a single node, we estimate communication volume analytically rather than measuring network traffic directly. Let P be the number of model parameters, K the number of active clients per round, and R the number of communication rounds. Assuming `float32` parameters, which can be represented with 4 bytes, and one model-sized download and upload per active client/round, the estimated bidirectional payload is $V = 2\times4PKR$ bytes.

This estimate assumes uncompressed model transfer and abstracts network latency and transport-level overhead. It should therefore be interpreted not as a characterization of the end-to-end communication volume, but as an estimate of the model-payload traffic. Moreover, methods exchanging additional state, such as SCAFFOLD control variates, may incur in additional traffic beyond this standardized estimate.

The results show that communication volume varies substantially across settings. Model size is a major driver of communication volume: FedCollecty is associated with the largest per-run volume (40.75 GB) despite involving only 9 active clients. This is due to its comparatively larger 2.83M-parameter model. Conversely, FedCrawdad requires less than 1 GB per run because of its substantially smaller (18K-parameter) model. Client participation also has, as expected, a strong effect. For example, FedBologna requires more communication volume (83.84 GB) in the cross-device scenario than in the cross-silo one (10.06 GB) despite using the same model architecture. This is due to the fact that more clients participate in each round in the cross-device scenario. Across all dataset/federation settings, seven federated strategies, and three splits, the resulting overall standardized model-transfer volume is approximately 2.92 TB.

Sensitivity of communication volume. To further characterize the communication requirements of the benchmark, we analyze how the estimated model-transfer volume changes with the main federated-training parameters. We perform this sensitivity analysis on FedBologna because it is the dataset used in the participation-budget analysis of Section D, allowing us to characterize both predictive and communication effects under the same controlled configurations. Specifically, this analysis focuses on the cross-device setting as a representative case and independently varies the number of active clients, the number of communication rounds, and the number of local updates performed by each client. As in the previous analysis, we assume one `float32` model-sized upload and download per active client at each communication round.

Varied parameter	Value	Local updates per client	Communication volume (GB/run)
Active clients	25	706	13.97
	50 (reference)	706	27.95
	75	706	41.92
Communication rounds	100	1,412	13.97
	200 (reference)	706	27.95
	400	353	55.89
Local updates	353	353	27.95
	706 (reference)	706	27.95
	1,059	1,059	27.95

Table 15: Sensitivity of the estimated communication volume for FedBologna in the cross-device setting.

Results are reported in Table 15 and show that communication volume scales linearly with both the number of participating clients and the number of communication rounds. For example, increasing the number of active clients from 25 to 75 increases the estimated traffic from 13.97 GB to 41.92 GB per run. Similarly, doubling the number of communication rounds from 200 to 400 doubles the communication volume from 27.95 GB to 55.89 GB. In contrast, changing the number of local optimization steps does not affect communication volume under the adopted protocol, and all three configurations require 27.95 GB per run. This follows directly from the separation between local computation and communication in our experimental design. Local updates modify the amount of computation performed between two communication events, whereas model parameters are exchanged only once per participating client and communication round.