


Which One is Better? Distributed Artificial Intelligence Strategies for Accurate Vehicular Emissions Forecasting

Carnot Braun   [Universidade Estadual de Campinas | c255785@dac.unicamp.br]

Rafael O. Jarczewski  [Universidade Estadual de Campinas | rojarczewski@lrc.ic.unicamp.br]

Allan M. de Souza  [Universidade Estadual de Campinas | allanms@unicamp.br]

 Institute of Computing, Universidade Estadual de Campinas, Cidade Universitária Zeferino Vaz - Barão Geraldo, Campinas - SP, 13083-970, Brazil.

Received: 05 November 2024 • **Accepted:** 26 May 2025 • **Published:** 02 October 2025

Abstract The rapid expansion of urban vehicular networks has led to increasing carbon emissions, posing significant environmental challenges in densely populated areas. Accurate emission predictions are crucial for sustainable urban planning, but current methods face limitations in handling large-scale dynamic data while balancing latency, privacy, and communication efficiency. This paper proposes a comprehensive framework study that compares centralized, federated, and shared learning approaches for CO₂ emissions prediction in vehicular networks, using data from vehicles and roadside units (RSUs) to predict emissions in diverse urban scenarios. By evaluating the performance of each approach on latency, communication overhead, and prediction accuracy, this work provides insights into optimizing learning strategies for real-time, scalable, and privacy-preserving emissions management in intelligent transportation systems. The findings offer valuable guidance to urban planners and policymakers, fostering the development of sustainable urban mobility solutions.

Keywords: Distributed AI Strategies, Vehicular Emissions, Greenhouse Emission, Federated Learning, Split Learning

1 Introduction

In recent years, the exponential rise of vehicular mobility has intensified urban challenges, particularly related to traffic congestion de Souza *et al.* [2020, 2019] and the increase in harmful emissions from conventional fuel-powered engines Sánchez *et al.* [2024]. The constant movement of vehicles in densely populated areas contributes significantly to CO₂ and other pollutants, creating both environmental and public health concerns. Emissions such as carbon monoxide (CO), nitrogen oxides (NO_x), and particulate matter (PM) degrade air quality, exacerbating respiratory and cardiovascular conditions while contributing to climate change Rahman and Thill [2023]; World Health Organization [2019]. These issues underscore the urgent need for solutions that can manage urban traffic more effectively and reduce the environmental footprint of connected vehicles, driving efforts toward more sustainable urban mobility practices.

Several strategies have emerged to address these challenges, with many focusing on traffic forecasting to support decision-making in intelligent transportation systems (ITS) Fei and Ling [2023]. For environmental applications, predicting carbon dioxide (CO₂) emissions accurately is critical, as it enables monitoring and management of the environmental impact of increasing urban traffic Zhang *et al.* [2023]. However, CO₂ forecasting presents unique challenges, as data acquisition can be difficult due to the need for high-quality, real-time data, which is very difficult to acquire in dynamic scenarios, and issues of data scarcity and heterogeneity in urban areas complicate processing. Additionally, ensuring the spatial-temporal consistency of collected data is essential for valid emission forecasting, but challenging to achieve due to

the dynamic nature of urban traffic and varying vehicle types Fei and Ling [2023]. While certain solutions focus directly on CO₂ prediction, adapting existing traffic forecasting models for emissions remains an effective approach to improve urban planning and advance sustainable development Rahman and Thill [2023].

Traditional approaches for predicting dynamic aspects in urban mobility often rely on a Centralized Learning (CL) architecture, where vehicles and roadside units (RSUs) continuously transmit raw data to a central server for processing Belal *et al.* [2024]. This server aggregates the data, enabling machine learning models to analyze comprehensive datasets. CL offers the advantage of high model accuracy, as the complete dataset provides a holistic view of traffic and emission patterns. However, it also presents notable limitations: centralized architectures can lead to bottlenecks due to the substantial data volume transmitted, resulting in high latency, increased communication costs, and privacy concerns associated with sharing raw data from numerous vehicles Sze *et al.* [2017].

These challenges underscore the need for alternative solutions, paving the way for distributed learning approaches, which reduce communication overhead and enhance data privacy by allowing vehicles to process data locally Zhang *et al.* [2024]; de Souza *et al.* [2022]. Distributed learning approaches, such as Federated Learning (FL) and Split Learning (SL), decentralize the model training process, mitigating the limitations of centralized architectures while enabling efficient, scalable, and privacy-preserving data processing in dynamic vehicular environments. This shift towards distributed solutions aligns with the growing complexity and scale of urban traffic systems, where maintaining data privacy

and minimizing latency are increasingly important Zhou *et al.* [2024]; Capanema *et al.* [2024].

The use of diverse learning approaches provides valuable insights into optimizing CO₂ emission forecasting by addressing the specific challenges of urban vehicular environments. These architectures—centralized, federated, and SL offer distinct advantages and trade-offs that can address issues such as data privacy, latency, and model accuracy de Souza *et al.* [2024].

In previous work, we have studied the impact of CO₂ on vehicular networks, considering centralized, federated, and SL architectures called EcoPredict. However, the communication overhead produced by these approaches still lacks a detailed analysis, especially for data transmitted between clients and servers Braun *et al.* [2024]. While EcoPredict demonstrates the feasibility of real-time prediction using urban sensor networks, our focus is on evaluating the impact of these architectures on data traffic, identifying the challenges and limitations imposed by the communication overhead of different types of models. In addition, we seek to understand how adaptive configurations can optimize these transmissions in different urban scenarios, ensuring a balance between prediction accuracy, computational efficiency, and data privacy.

CL aggregates all data on a central server, which supports high model accuracy due to comprehensive data access. However, it is limited by significant data privacy concerns and potential communication bottlenecks. FL, on the other hand, allows data to be processed locally on edge devices, enhancing data privacy and reducing communication latency. Despite these benefits, FL can experience reduced accuracy because the model training relies on decentralized data, which may be incomplete or unbalanced Belal *et al.* [2024]; de Souza *et al.* [2024]. SL serves as a compromise, partitioning the model between edge devices and a central server to optimize both data privacy and model performance. However, SL still faces limitations, particularly in handling real-time data transfer between devices, which can result in latency challenges in dynamic vehicular networks Zhang *et al.* [2024].

Comparing these approaches is essential to identify the most effective methods for managing the trade-offs in vehicular scenarios, where efficient communication, privacy, and accurate emission predictions are critical for sustainable urban development.

In this way, this paper presents a comprehensive framework analysis of different learning approaches, including centralized, federated, and SL for CO₂ emission prediction in urban scenarios. Focusing on performance, latency, and communication overhead, this work evaluates the suitability and trade-offs of each approach within the context of real urban mobility data from cities such as Luxembourg Codecá *et al.* [2017], Cologne Uppoor *et al.* [2013], and Ingolstadt Lobo *et al.* [2020]. Using SUMO for detailed traffic simulations, the framework define the RSUs (clients) collect emission data from connected vehicles, transforming this information into time series datasets for further analysis.

The findings highlight each approach's strengths and limitations: (i) CL provides high accuracy but suffers from significant privacy constraints; (ii) FL enhances privacy but may sacrifice model accuracy due to decentralized data processing; and (iii) SL strikes a balance between performance and

privacy, however it can suffer from high latency due its sequential nature. These insights guide the selection of the most effective learning paradigm for sustainable urban mobility management.

The main contributions of this paper can be summarized as follows: (i) We focus on evaluating the overhead of three communication approaches for urban emissions prediction, specifically the data transmitted from client to server (uplink) and from server to client (downlink), which are not present in other papers in the literature; (ii) Adaptive configurations that can be adjusted to various urban scenarios and infrastructures; (iii) Performance evaluation using realistic mobility traces.

The remainder of this paper is organized as follows: Section 2 presents a comprehensive background, highlighting centralized, federated, and SL approaches in vehicular and urban scenarios. Section 3 details the proposed framework and its components, describing the system architecture and the training approaches for emission prediction. Section 4 outlines the experimental setup, including the simulation environment, data sources, and model configurations used in the analysis. In addition, the results are presented, comparing each approach in terms of performance, latency, and communication overhead. Finally, Section 5 discusses the implications of the findings and possible avenues for future work.

2 Background

This section presents foundational concepts and reviews relevant research on learning approaches in vehicular networks. The background is divided into three primary approaches: (i) CL, (ii) FL, and (iii) SL. Figure 1 illustrates the functioning of these three approaches, providing a visual representation of their processes in urban mobility scenarios.

2.1 Centralized Learning

CL involves aggregating data (i.e., raw data) from multiple vehicles at a central server for processing and decision-making 1(a). This approach typically results in higher accuracy due to the availability of large datasets, but it also poses significant scalability and communication challenges.

Muhammad Saleem *et al.* Saleem *et al.* [2022] developed a traffic monitoring system where all vehicle data is sent to a central server for traffic prediction. This centralized approach allows the server to aggregate a large amount of data, which can improve the accuracy of predictions by providing a holistic view of the traffic situation. However, this dependence on centralization introduces significant bottlenecks, particularly in high-density environments where data volumes are substantial. As data is constantly transmitted to the server, the system's scalability and robustness are compromised, making it vulnerable to network congestion and server overload, which are critical in dynamic vehicular environments.

Yang *et al.* Yang [2022] introduced a centralized traffic control solution, leveraging a central server to optimize traffic flow based on data collected from vehicles and roadside sensors. The server processes large datasets to make real-time decisions aimed at improving traffic management. While this method provides high model accuracy, the main limitation

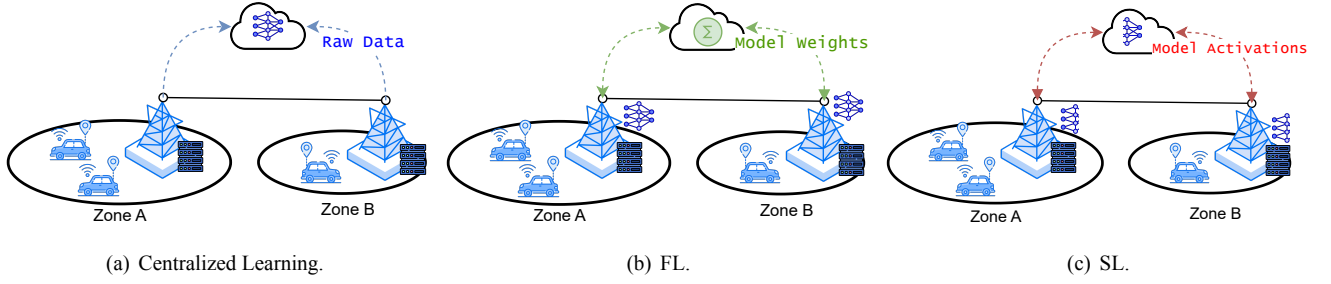


Figure 1. Examples of how each training approach works. Even though they start from the same scenario, each approach has its own specific operational characteristics.

is the over-reliance on the central server. This creates vulnerabilities related to communication failures and scalability issues, as the server becomes a single point of failure. In environments with unstable network connectivity or a large number of vehicles, this approach can suffer from significant latency, leading to delayed decision-making and reduced system efficiency.

Wang *et al.* Wang *et al.* [2020] proposed a cloud computing architecture for real-time data processing from connected vehicles. In this system, vehicles transmit vast amounts of data to the cloud, where it is processed to generate insights into traffic patterns and vehicle behavior. Cloud-based processing offers powerful computational resources, enabling complex analyses and more accurate predictions. However, the reliance on high-quality, continuous connectivity between vehicles and the cloud is a major limitation. In mobile vehicular networks, maintaining a stable connection can be challenging, especially in areas with fluctuating network conditions, which can result in delays or loss of data, ultimately affecting the system's ability to provide timely insights.

Wu *et al.* Wu *et al.* [2018] proposed an Internet of Things-based solution with centralized learning, where sensors installed on vehicles transmit data to a central server. The server processes this data to train models, which are then used to make traffic or environmental predictions. While the use of IoT enhances data collection and provides more detailed insights, the centralized nature of this system presents significant scalability issues. As the number of connected vehicles increases, the server must handle an ever-growing volume of data, which can lead to bottlenecks in communication and processing. This centralization also makes the system more vulnerable to network disruptions, especially in urban environments with high vehicle density.

Zhao *et al.* Zhao *et al.* [2019] proposed an accident prediction system for intelligent transportation systems (ITS) using centralized learning, where vehicles send data to cloud servers for processing. The system provides accurate accident predictions by analyzing the data collected from multiple vehicles and traffic conditions. However, the latency associated with transmitting data to the cloud and processing it centrally makes this solution less ideal for critical environments with stringent real-time requirements. Delays in communication and processing can undermine the system's effectiveness in preventing accidents in fast-changing traffic scenarios, where immediate actions are necessary.

Farnoush Falahatraftar *et al.* Falahatraftar *et al.* [2021]

developed a deep learning-based centralized system to predict mobility patterns in vehicular networks. The model is trained using data aggregated from sensors and mobile devices on a central server. While the system benefits from the centralized aggregation of large datasets, allowing for more accurate mobility predictions, it faces challenges related to latency and server vulnerabilities. The constant data transmission required for real-time predictions increases communication overhead, which can become a significant burden in networks with a large number of vehicles. Additionally, the system's dependence on a central server introduces risks of failure if the server becomes overloaded or experiences downtime.

2.2 Federated Learning

FL allows vehicles to collect data locally and train models, sending only the update weights to a central server for aggregation, depicted in Figure 1(b). This decentralized approach effectively preserves privacy but introduces challenges related to latency and communication overhead.

FL allows vehicles to collect data locally and train models, sending only the updated weights, the parameters derived from local training, which encapsulate the learned features without revealing raw data to a central server for aggregation 1(b). This decentralized approach effectively preserves privacy by keeping the raw data on each vehicle; instead, only the updated weights are shared, enabling the central server to combine these updates into a global model. However, this method introduces challenges related to latency and communication overhead.

Ahmet M. Elbir *et al.* Elbir *et al.* [2022] proposed a FL solution for vehicular networks aimed at preserving the privacy of vehicle data during collaborative model training. In this context, vehicles collect data locally, train their models, and send the updated weights to a central server for aggregation. While this decentralized approach effectively preserves privacy, the reliance on a central server for model aggregation introduces challenges related to latency and communication overhead, especially in high-density scenarios.

Manoj Kumar *et al.* Kumar [2024] developed a FL framework for traffic flow prediction in vehicular networks using recurrent neural networks (RNN). They introduced an effective traffic flow prediction method (ETraffic-FP) based on actual historical traffic data. The local Traffic-FP model consists of three main components: a recurrent long short-term capture network (RLSCN), a federated gated graph attentive network (FGAN), and a semantic connection relationship

capture network (SCRCN). This approach reduces data transmission and preserves privacy. However, it faces challenges due to heterogeneous data across vehicles and potential synchronization delays, particularly when vehicles move through regions with varying traffic conditions.

Kaur *et al.* Kaur [2024] proposed a FL based spatio-temporal approach termed Fed-STGRU for traffic prediction, avoiding transmitting raw user data over the network. Their scheme maintains privacy while achieving comparable accuracy and loss to baseline algorithms such as TGCN, FedAvg, and FedGRU. Although this approach helps address privacy concerns, the communication overhead from frequent updates, especially in scenarios with many vehicles, leads to challenges in scalability and latency, particularly during peak traffic periods.

2.3 Split Learning

SL integrates elements of both centralized and FL by partitioning the model between the client and server, illustrated in Figure 1(c), but unlike other approaches, clients share only the model activations. This allows for more efficient communication while still addressing privacy concerns.

Abedi *et al.* Abedi [2024] integrated FL with SL in their work, FedSL, aiming to optimize communication and enhance privacy in sequential data. The combination of these approaches is crucial, as traditional FL may require devices to transmit large model updates to a central server, leading to high communication costs. However, the limitation imposed by static model partitioning restricts the effectiveness of this method in vehicular networks, where device capabilities often vary significantly. The flexibility of SL allows for dynamic distribution of model components, enabling better adaptation to the individual capabilities of vehicles. This approach not only minimizes data traffic but also maximizes learning efficiency, which is critical in the ever-changing environments of vehicular networks.

Eric Samikwa *et al.* Samikwa *et al.* [2024] proposed a version of SL adapted for heterogeneous devices, in which vehicles train parts of a model based on their computational capabilities. This approach is essential for alleviating the processing load on less powerful vehicles, enabling them to actively participate in learning without being overwhelmed. However, reliance on a partially centralized structure can still lead to bottlenecks and increased communication overhead. Implementing a more decentralized SL could further enhance system efficiency, allowing vehicles to collaborate without needing a central control point, which would be beneficial for improving scalability and resilience in scenarios with heavy traffic.

Guhan Zheng *et al.* Zheng *et al.* [2024] presents a novel framework designed to address the complex challenges of vehicular semantic communications in highly dynamic environments. The proposed framework, MSFTL (Mobility-Aware Split-Federated with Transfer Learning), integrates a split-FL model and transfer learning to adapt to the high mobility of vehicles and manage the real-time offloading of tasks. This method splits the semantic coder model into four components, allowing vehicles to train parts of the coder, while an edge component handles other parts, optimizing computational

workload and minimizing communication costs. The MSFTL framework is further supported by a Stackelberg game-based resource optimization mechanism, enhancing its adaptability and efficiency. One key advantage of this approach is its ability to maintain high model accuracy while reducing latency and computational demand, which is critical in mobile vehicular networks. However, the approach has limitations, particularly regarding communication overhead in scenarios with very large datasets. This framework's integration of both SL and transfer learning allows it to operate in complex mobility contexts, making it highly relevant to vehicular networks where frequent and fast model updates are required.

2.4 Comparative Analysis

In reviewing existing approaches (Table 1), centralized learning methods demonstrate high accuracy due to their access to aggregated datasets. However, their reliance on centralized data processing introduces scalability challenges, computational bottlenecks, and privacy risks, particularly in large-scale vehicular networks. FL mitigates some of these issues by enabling local training and preserving data privacy. Yet, data fragmentation across distributed clients can lead to model convergence difficulties, and synchronization overhead may hinder performance, especially in high-mobility scenarios. SL, by distributing model layers between clients and a central server, reduces communication costs and accommodates heterogeneous devices. Nevertheless, its dependence on inter-device coordination presents challenges in dynamic vehicular networks with intermittent connectivity.

To overcome these limitations, this paper introduces, a framework that integrates centralized, federated, and SL approaches for CO₂ emissions prediction, with the flexibility to adapt to various vehicular conditions. Unlike previous works, the framework employs dynamic model partitioning to allocate computational load based on device capabilities and network conditions, thus enhancing latency and communication efficiency. Additionally, it incorporates multiple machine learning models, enabling real-time adaptability in urban vehicular scenarios while maintaining both privacy and prediction accuracy.

3 Predicting Vehicular Emissions

This section describes the framework used to predict CO₂ emissions in urban mobility environments using CL, FL, and SL approaches. The key idea is to provide a platform to collect and share mobility-related data to train a model for forecasting on the data collected. In this way, Subsection 3.1 introduces the system model considered in this work, while Subsection 3.2 describe the data considered to train the machine learning models and how it is collected. Finally, Subsection 3.3 provides details on how the different approaches analyzed work, with the help of pseudo-codes, that algorithms presented were adapted from methods established in the literature.

Table 1. Comparative Analysis of Related Approaches in Vehicular Networks

Study	Approach	Advantages	Limitations	Latency	Communication
Saleem <i>et al.</i> [2022]	Centralized Learning	High accuracy in traffic predictions	Scalability issues in high-density environments	High	High
Yang [2022]		Real-time optimization	Vulnerability to server failure	High	High
Wang <i>et al.</i> [2020]		Powerful processing capabilities	Dependency on stable connectivity	High	High
Wu <i>et al.</i> [2018]		Detailed insights via IoT data	Limited scalability, prone to bottlenecks	High	High
Zhao <i>et al.</i> [2019]		Accurate accident predictions	High latency for critical responses	High	High
Falahatraftar <i>et al.</i> [2021]		Predictive accuracy with large datasets	Communication burden and server overload risk	High	High
Elbir <i>et al.</i> [2022]	Federated Learning	Preserves privacy	Synchronization issues in high-density environments	Medium	Medium
Kumar [2024]		Spatio-temporal accuracy	Synchronization delays in varying regions	Medium	Medium
Kaur [2024]		Enhanced data privacy	Communication overhead, scalability challenges	Medium	Medium
Abedi [2024]	Split Learning	Reduces communication load, increases privacy	Limited by static partitioning	Medium	Medium
Samikwa <i>et al.</i> [2024]		Adapts to heterogeneous devices	Partially centralized, potential bottlenecks	Medium	Medium
Zheng <i>et al.</i> [2024]		High adaptability and efficiency	Communication overhead in large datasets	Medium	Medium

3.1 System Model

The system model employed in this work comprises vehicles, the set R of Roadside Unit (RSU), the set S of roads in the monitored city, and a remote server in the cloud. The city is divided into $|R|$ regions, where $|R| = m$ represents the number of RSUs in the scenario. Each road in the city is denoted as $s_j \in S = \{s_1, s_2, \dots, s_n\}$, where n is the total number of roads on which vehicles circulate and communicate with the RSUs. Each RSU $r_i \in R = \{r_1, r_2, \dots, r_m\}$ covers a specific area, sensing and collecting data from all vehicles in the respective region. In this context, the RSUs act as urban sensors, collecting parameters such as vehicle speed, average number of vehicles, fuel consumption, and CO₂ emissions. Finally, the RSUs communicate with the remote server via wired connections.

3.2 Data Collection

RSUs collect CO₂ emissions data at regular intervals of 10 seconds that are naturally sent in vehicular networks or by sensing and measuring the CO₂ emissions within the region. This data collection occurs through two main methods: (i) beacon messages, which are periodically broadcast by vehicles as part of standard vehicular communications, and (ii) direct sensing of CO₂ emissions within the coverage area of each RSU. Each RSU r_i , operating in a specific region, monitors the roads in its vicinity and collects emissions data from passing vehicles. Creß *et al.* [2024]

This arrangement generates a time series of CO₂ emissions recorded over time. For each r_i , a time series is constructed based on the movements of vehicles in its region. The collected data can be represented as a vector $X^i = \{x_1^i, x_2^i, \dots, x_k^i\}$, where k denotes the total number of time steps. Each element of this vector consists of a tuple: $x_j^i = (timestamp, vehicleId, roadId, CO_2)$ where for each record $x_j^i \forall j \in \{1, 2, \dots, k\}$, the *timestamp* represents the exact time at which the data was captured, *vehicleId* is a unique identifier assigned to each vehicle, *roadId* is the road segment where the vehicle is located during data collection, and *CO₂* is the CO₂ rating level calculated for that vehicle at that time.

Each dataset X^i consists of a continuous time series that reflects the dynamic changes in CO₂ emissions as vehicles travel along different road segments. This comprehensive dataset provides crucial information about emissions across multiple regions and time periods, making it valuable for

building predictive models. Before training, the dataset undergoes a pre-processing step, forming a new dataset $\mathcal{D} = (X, Y)$, where X is data collected up to a certain time and Y is data collected from that time up to a prediction horizon τ . The idea is that this new set \mathcal{D} is used to train a machine learning model $h_\theta(\cdot)$, with parameters θ , to predict new CO₂ emission data, approximating a real and unknown function f , given by $f(x_i, \tau) = y$ where x_i is the input vector, and τ is the horizon that defines how far into the future the value should be predicted.

The goal, therefore, is to train a machine learning model $h_\theta(\cdot)$ to learn a mapping $X \rightarrow Y$ by adjusting its set of parameters θ to minimize a loss function L :

$$L = \frac{1}{|Y|} \sum_{i \in Y} (y_i - h_\theta(X))^2 \quad (1)$$

The loss function L measures the prediction quality provided by the model $h(\cdot)$ with the set of parameters θ . The better the model, the lower its cost. Thus, we consider three different learning approaches (CL, FL, and SL) to train the model $h(\cdot)$ with respect to the loss function L , which will be described in the following subsections.

3.3 Learning Approaches

In the CL approach, the raw data collected from vehicles and RSUs is transmitted to a central server, where the model is trained on the complete dataset. This allows for a comprehensive understanding of the data and often results in high model accuracy. However, it introduces latency, privacy risks, and high communication overhead due to the transmitted data volume.

The goal of the CL approach is to minimize the loss function L over the aggregated dataset \mathcal{D} from all RSUs, optimizing the model parameters θ of $h(\cdot)$:

$$\theta^* = \arg \min_{\theta} \sum_{(x,y) \in \mathcal{D}} L(h_\theta(x), y). \quad (2)$$

To minimize this equation the gradient descent is used iteratively to update θ :

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_{\theta} L(h_\theta(x), y), \quad (3)$$

where η is the learning rate and ∇_{θ} is the gradient calculated over the parameters θ .

The algorithm 1 describes a CL process where the central server initializes a model $h(\cdot)$ with parameters θ_0 . At each time step t , each RSU r_i in the set R collects local data \mathcal{D}_t^i and sends it to the server. The central server then aggregates the data from all RSUs, forming a complete dataset $\mathcal{D}_t = \sum_{i=1}^R \mathcal{D}_t^i$. To train the model, the server iteratively processes each minibatch (x_B, y_B) of the dataset, computing the gradient $\nabla_{\theta} L(h_{\theta}(x_B), y_B)$ based on the loss function L and updating the model parameters using (3). This process continues at each time step, ensuring that the model is trained centrally using the aggregated data.

Algorithm 1 CL

Input: model $h(\cdot)$, initial parameters θ_0 , number of time steps T and collection of RSUs R .

```

1: Initialize model  $h(\cdot)$  with parameters  $\theta_0$  on the central server
2: for each time step  $t \in \{1, 2, 3, \dots, T\}$  do
3:   for each RSU  $r_i \in R$  do
4:     Collect data  $\mathcal{D}_t^i$ 
5:     Send  $\mathcal{D}_t^i$  to the server
6:   end for
7:   Aggregate all data  $\mathcal{D}_t = \sum_i^R \mathcal{D}_t^i$ 
8:   for each minibatch  $(x_B, y_B) \subset \mathcal{D}_t$  do
9:     Compute gradient  $\nabla_{\theta} L(h_{\theta}(x_B), y_B)$ 
10:    Update  $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \nabla_{\theta} L(h_{\theta}(x_B), y_B)$ 
11:   end for
12: end for
13: Return:  $h_{\theta_T}(\cdot)$ 

```

FL distributes model training across each RSU. Each device trains a model locally on its own data, transmitting only the model updates (e.g., weights or gradients) to a central server, which aggregates them into a global model. This approach enhances privacy and reduces data volume, but it may introduce challenges related to latency and synchronization.

In FL, each RSU $r_i \in R$, in each round t , computes its local model update $\Delta\theta_t^i$ by optimizing the parameters on its dataset \mathcal{D}_i . The server aggregates these updates using some strategy, such as weighting the updates by the size of the respective dataset and adding them to the parameters from the previous round:

$$\theta_t \leftarrow \theta_{t-1} + \frac{1}{|\mathcal{D}_t|} \sum_{i=1}^{|R|} \Delta\theta_t^i \cdot |\mathcal{D}^i| \quad (4)$$

where $|R|$ is the set of RSUs. The local update $\Delta\theta_t^i$ is calculated as: $\theta_t^i - \theta_{t-1}^i$, and the parameter θ_t^i at client i and round t is calculated as:

$$\theta_t^i = \theta_{t-1}^i - \eta \cdot \nabla_{\theta} L(h_{\theta}(x), y) \text{ for } (x, y) \in \mathcal{D}_i. \quad (5)$$

where η is the learning rate, L is the loss function, and $h_{\theta}(\cdot)$ is the model learned based on parameters θ .

Algorithm 2 describes a FL process where a global model $h(\cdot)$ with parameters θ is initialized at the server. For each communication round t , the server distributes the current global model θ_{t-1} to all RSUs in parallel. Each RSU performs a local update by setting its parameters $\theta_t^i \leftarrow \theta_{t-1}$

and iteratively processing each minibatch $(x_B, y_B) \subset \mathcal{D}_t^i$. During this local training, the RSU computes the gradient $\nabla_{\theta} L(h_{\theta}(x_B), y_B)$ based on the loss function L and updates its local parameters using (5). After completing local training, each RSU sends its local update $\Delta\theta_t^i$ back to the server. The server aggregates these local updates using (4) and distributes the updated global parameters θ_t to all RSUs in the next round. This iterative process allows the model to be trained collaboratively without centralizing the data.

Algorithm 2 FL

Input: model $h(\cdot)$, initial parameters θ_0 , number of rounds T and collection of RSUs R .

```

1: Initialize the global model  $h(\cdot)$  with parameters  $\theta_0$  on the server.
2: for each round of communication  $t \in \{1, 2, 3, \dots, T\}$  do
3:   for each RSU  $r_i \in R$  in parallel do
4:     Local Update:  $\theta_t^i \leftarrow \theta_{t-1}$ 
5:     Collect data  $\mathcal{D}_t^i$ 
6:     for each minibatch  $(x_B, y_B) \subset \mathcal{D}_t^i$  do
7:       Compute gradient  $\nabla_{\theta} L(h_{\theta}(x_B), y_B)$ 
8:       Updates  $\theta_t^i \leftarrow \theta_t^i - \eta \cdot \nabla_{\theta} L(h_{\theta}(x_B), y_B)$ 
9:     end for
10:    Sends the local update  $\Delta\theta_t^i = \theta_t^i - \theta_{t-1}$  to the server
11:   end for
12:   Aggregation:  $\theta_t \leftarrow \theta_{t-1} + \frac{1}{|\mathcal{D}_t|} \sum_{i=1}^{|R|} \Delta\theta_t^i \cdot |\mathcal{D}^i|$ 
13:   Distributes updated  $\theta$  to each RSU
14: end for
15: Return:  $h_{\theta_T}(\cdot) = 0$ 

```

In SL the global model is divided into two segments: an initial segment $h_{\theta}^l(\cdot)$ processed locally on RSUs, and a final segment $h_{\theta}^s(\cdot)$ processed on the central server. The RSUs process their data through $h_{\theta}^l(\cdot)$, generating intermediate outputs sent to the server for further processing. This method reduces data transfer volume but may be affected by network stability and latency due to intermediate data transfer.

In each round t , each RSU computes an intermediate output $O_i = h_{\theta}^l(\mathcal{D}_t^i)$. The server processes O_i with $h_{\theta}^s(\cdot)$ to minimize the loss $L(h_{\theta}^s(O_i), y_i)$ and updates the model parameters θ^s via gradient descent. Specifically, the server updates θ^s according to the gradient $\nabla_{\theta^s} L(h_{\theta}^s(O_i), y_i)$, scaled by the learning rate η , so that $h_{\theta}^s(O_i)$ gets closer to the true labels y_i . This is done every second round:

$$\theta_t^s \leftarrow \theta_{t-1}^s - \eta \cdot \nabla_{\theta^s} L(h_{\theta}^s(O_i), y_i) \quad (6)$$

This approach allows the server to learn from decentralized data while reducing communication costs, since only intermediate outputs are shared instead of the raw data.

The Algorithm 3 describes the SL process, where a model h_{θ} is split into a local part h_{θ}^l , with parameters θ^l on the devices (RSUs), and a part on the server h_{θ}^s , with parameters θ^s . For each communication round $t \in \{1, 2, 3, \dots, T\}$, each RSU $r_i \in R$ performs a local forward pass, computing $O_i = h_{\theta}^l(\mathcal{D}_t^i)$ using its local data \mathcal{D}_t^i and sending the output O_i to the server. At the server, the received outputs O_i are used to compute the gradient $\nabla_{\theta^s} L(h_{\theta}^s(O_i))$ based on the loss function L . The server updates its parameters using

Algorithm 3 SL

Input: partial models $h^l(\cdot)$ and $h^s(\cdot)$, initial parameters θ_0^l and θ_0^s , number of rounds T and collection of RSUs R .

```

1: Initialize partial models  $h_\theta^l, h_\theta^s$  with parameters  $\theta_0^l$  on
   devices and  $\theta_0^s$  on server.
2: for each round of communication  $t \in \{1, 2, 3, \dots, T\}$ 
   do
3:   for each RSU  $r_i \in R$  in parallel do
4:     Collect data  $(x_i, y_i) \in \mathcal{D}_t^i$ 
5:     Forward Local: Compute  $O_i = h_\theta^l(x_i)$ 
6:     Send  $O_i$  to server
7:   end for
8:   Server Update:
9:   for each  $O_i$  do
10:    Compute gradient  $\nabla_{\theta^s} L(h_\theta^s(O_i), y_i)$ 
11:    Update  $\theta_t^s \leftarrow \theta_{t-1}^s - \eta \cdot \nabla_{\theta^s} L(h_\theta^s(O_i), y_i)$ 
12:   end for
13:   Send the updated gradient  $\nabla_{\theta^s}$  back to each RSU to
       end the backpropagation process.
14: end for

```

$\theta_t^s \leftarrow \theta_{t-1}^s - \eta \cdot \nabla_{\theta^s} L(h_\theta^s(O_i))$, where η is the learning rate. After the server-side update is complete, the server sends the updated gradient ∇_{θ^s} back to each RSU, completing the backpropagation process and updating its local parameters. This coordinated training enables distributed learning while maintaining device-level data privacy.

4 Performance Analysis

In this section, we discuss the simulation environment, data processing techniques, and model configurations used to evaluate CO₂ emissions prediction in urban vehicular scenarios. The evaluation considers various traffic patterns simulated with realistic mobility traces (TAPASCologne, LuST, and InTAS) within the SUMO platform. We examine three machine learning models—LSTM, CNN, and CNN+LSTM trained using CL, FL, and SL strategies, each one tested on diverse traffic data. The analysis includes performance metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), communication overhead, and the accuracy impact across different prediction horizons and learning methods, providing a comprehensive understanding of the models' predictive abilities and resource requirements.

4.1 Simulation Platform

The experiments were conducted using Simulation of Urban MObility (SUMO) Krajzewicz and et al. [2012] version 1.16.0, a highly adaptable vehicular traffic simulation platform. SUMO was connected to our Python 3.12.0 implementation using the TraCI interface, which allows for real-time interaction and data extraction from the simulation, using the codes developed in the repository¹, responsible for extracting the desired data from each scenario. We utilized three well-known realistic vehicular mobility traces: TAPAS-Cologne Uppoor *et al.* [2013], Luxembourg SUMO Trace (LuST) Codecá *et al.* [2017], and a InTAS Lobo *et al.* [2020].

¹<https://github.com/carnotbraun/UMI>

These traces provide 24 hours of vehicular mobility data, allowing for a comprehensive evaluation across different traffic patterns. In each scenario, 30% of the total traffic was simulated to replicate typical urban vehicular environments.

The RSUs's positions were considered based on the Open-CellID² information. Each RSU is responsible for collecting data from all the roads within its coverage area. A Fast Ethernet link is considered to connect each RSU to the remote server, ensuring sufficient bandwidth for high-frequency data transmissions. In this setup, RSUs act as clients in each training approach, aggregating and sending data based on their coverage zones.

The variety of data that can be processed includes fuel consumption, noise emissions, CO₂ emissions, Nitric Oxide (NOx) emissions, and average vehicle speed within each RSU's coverage area³. In our study, we focused exclusively on CO₂ emissions for training and evaluating the Machine Learning (ML) models, with data split into training and testing sets using a 65/35 ratio. This split was maintained consistently across different time intervals, as illustrated in Figure 2, which shows the time series data collected for each scenario.

The Scenarios traces provide realistic traffic patterns and vehicle behaviors, essential for robust model evaluation. TAPASCologne simulates traffic flows over a large urban area, with high-density traffic around specific urban points, while LuST provides a detailed representation of a smaller region as well InTAS. Each trace's duration was divided into intervals that allow for time series analysis of CO₂ emissions, focusing on temporal changes and predictive modeling.

For the simulations, 12 RSUs were considered for each scenario, which had a coverage radius of approximately 2 kilometers. In addition, 50 training epochs were considered for the centralized approach and 10 rounds with 5 epochs for the SL and FL for the experiments.

4.2 Model's Description

The LSTM architecture includes a 50-unit LSTM layer to capture temporal dynamics, followed by a dense layer with 32 neurons using ReLU activation, and a single output neuron for continuous prediction. The CNN architecture consists of a Conv1D layer with 64 filters and a kernel size of 2 using ReLU activation, a MaxPooling1D layer to reduce spatial dimensionality, a flatten layer, dense layers with 50 and 32 neurons using ReLU, and a single output neuron for continuous prediction.

The CNN+LSTM architecture combines a Conv1D layer with 64 filters (kernel size 2, ReLU activation), a MaxPooling1D layer, a 50-unit LSTM layer, a dense layer with 32 neurons using ReLU, and a single output neuron for continuous prediction.

We use the Adam optimizer and MSE loss for all networks with a learning rate of 0.001. In SL, each architecture is divided at the dense layer with 32 neurons: the initial layers remain on the client side, while the final layers are on the server side. In FL, all clients maintain the complete architecture, and the server's role is limited to receiving the model

²<https://www.opencellid.org/>

³<https://sumo.dlr.de/docs/Simulation/Output/>

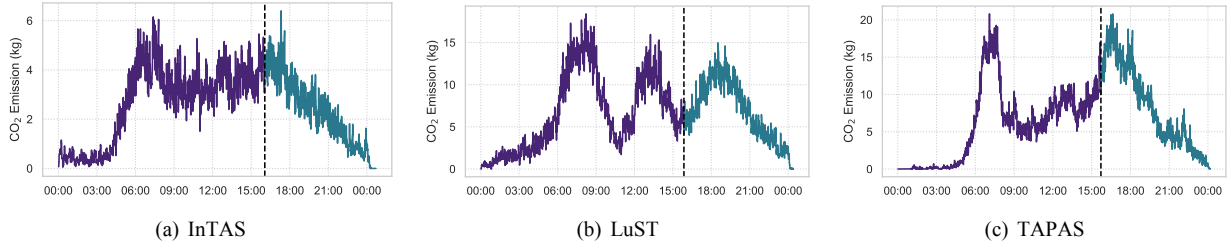


Figure 2. Example data (CO₂ emission in grams \times 1000) of Roadside Unit (RSU) used for the training and prediction processes in each vehicular mobility scenario and learning approach

weights, aggregating them, and returning the updated weights to each client.

4.3 Evaluation Metrics

The performance of each learning model was assessed using three primary error metrics: (i) MSE and (ii) RMSE. MSE quantifies the average squared difference between predicted and actual values, making it particularly sensitive to larger prediction errors. Conversely, RMSE offers an interpretable error metric by taking the square root of MSE, ensuring error measures remain on the same scale as the original data values. These metrics together provide a comprehensive understanding of the models' predictive accuracy by capturing both small and large error contributions.

In addition, the overhead was measured to assess the time spent by each learning approach, which refers to the time required for training the model considering all iterations between client and server, in other words it considers the training time (*feed forward* and *backpropagation*) and transmission time (model and gradients shared between client and server considering uplink and downlink). This metric captures both directions of communication in each round and reflects the total time overhead introduced by the learning framework in the vehicular environment.

4.4 Results and Analysis

Table 2 presents the RMSE values of the models discussed in previous sections for CO₂ prediction. We can see that LSTM generally shows the lowest RMSE values, indicating better performance in CO₂ prediction. However, CNN shows competitive performance in some scenarios and horizons but is often outperformed by LSTM. CNN+LSTM did not achieve a sufficient and competitive performance, although it performed well at times with the centralized method.

The FL approach demonstrates better values, indicating the effectiveness of FL in CO₂ prediction. When looking at the SL, we can see that it shows higher RMSE values in longer prediction horizons. As expected, RMSE values increase as prediction horizons extend.

When analyzing the performance of CNN model, it shows better results for shorter and intermediate horizons. CNN outperforms LSTM in CL to LuST and InTAS scenarios, and in SL for TAPAS.

However, LSTM generally has the lowest relative and absolute growth, ranging from 2% to 7%. This trend is observed

at TAPAS $\tau = 5$ and InTAS $\tau = 30$. Interestingly, LSTM demonstrates the best performance in FL across three scenario, suggesting it is particularly well-suited for federated setups in these cases.

Nonetheless, CNN+LSTM performed the worst, showing lower results than both CNN and LSMT models. This behavior indicates that CNN+LSTM model struggles to effectively understand the data.

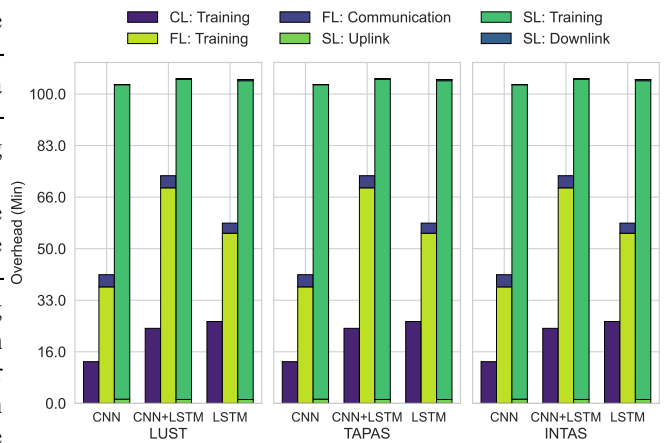


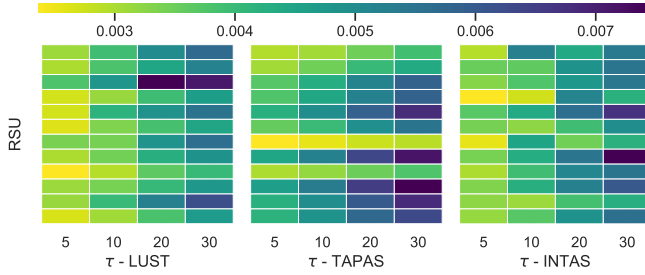
Figure 3. Overhead produced by each strategy in each scenario.

Figure 3 shows the latency, the communication overhead associated with different learning models (CNN, CNN+LSTM and LSTM) in the CL, FL, and SL training scenarios. The objective of the analysis is to verify the impact of communication overhead when varying the model and training method, focusing on vehicular network environments.

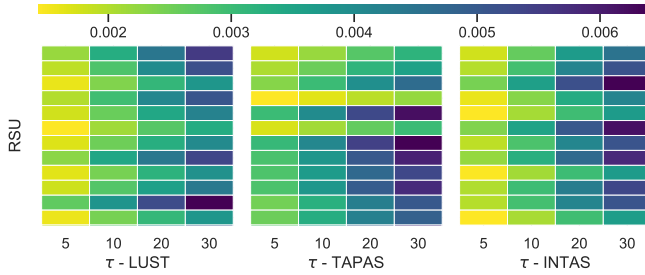
In the three models, the overhead was observed in specific activities of each training scenario. For the centralized scenario, only training on the server is considered, with data aggregated from all collection points. In the federated scenario, the overhead is segmented into three stages: training on local devices, communication of local models to the server, and aggregation of models on the server. In SL, the overhead is assessed at the stages of sending activations (*uplink*) from devices, partial training on the server, and returning gradients to devices (*downlink*). This process makes the training time in SL longer than the others. Otherwise, FL has to wait for all clients to process and send back the parameters, which causes a higher latency than centralized and SL. However, it is important to notice that this limitation could be reduced by parallelizing the server tasks (i.e., *feed forward* and *backpropagation* in its model piece) at a cost of generalization.

Table 2. RMSE results for each AI strategy for different horizons (τ) and models

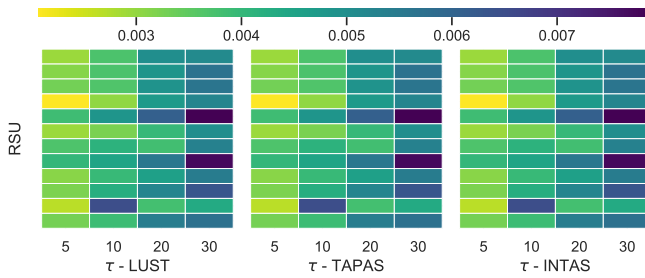
Scenario	Models	Centralized				Federated				Split			
		$\tau = 5$	$\tau = 10$	$\tau = 20$	$\tau = 30$	$\tau = 5$	$\tau = 10$	$\tau = 20$	$\tau = 30$	$\tau = 5$	$\tau = 10$	$\tau = 20$	$\tau = 30$
LUST	CNN	0.0025	0.0031	0.0039	0.0047	0.0019	0.0028	0.0036	0.0045	0.0031	0.0037	0.0048	0.0053
	CNN+LSTM	0.0028	0.0034	0.0047	0.0049	0.0019	0.0028	0.0036	0.0044	0.0033	0.0039	0.0048	0.0059
	LSTM	0.0025	0.0032	0.0040	0.0046	0.0016	0.0026	0.0035	0.0043	0.0029	0.0035	0.0045	0.0053
TAPAS	CNN	0.0042	0.0053	0.0069	0.0080	0.0031	0.0045	0.0064	0.0077	0.0030	0.0036	0.0045	0.0055
	CNN+LSTM	0.0044	0.0055	0.0069	0.0081	0.0032	0.0045	0.0064	0.0077	0.0032	0.0049	0.0052	0.0061
	LSTM	0.0041	0.0051	0.0068	0.0080	0.0027	0.0043	0.0062	0.0075	0.0032	0.0037	0.0050	0.0052
INTAS	CNN	0.0031	0.0036	0.0045	0.0057	0.0020	0.0029	0.0041	0.0051	0.0030	0.0035	0.0045	0.0053
	CNN+LSTM	0.0033	0.0043	0.0054	0.0056	0.0020	0.0029	0.0041	0.0050	0.0032	0.0040	0.0048	0.0058
	LSTM	0.0031	0.0043	0.0046	0.0053	0.0017	0.0027	0.0039	0.0049	0.0029	0.0038	0.0045	0.0053



(a) CL



(b) FL



(c) SL

Figure 4. Comparison of the RMSE of the CNN+LSTM model in the CL, FL and SL training scenarios, showing the impact of each approach on the prediction accuracy in different time horizons.

In the figure 4 presented for the CNN+LSTM model, the RMSE is analyzed in three different scenarios (LUST, TAPAS, INTAS) for the centralized, FL and SL approaches. RMSE is a performance metric used to evaluate the accuracy of the model's predictions in different forecast horizons ($\tau = 5$, $\tau = 10$, $\tau = 20$ and $\tau = 30$).

In the centralized scenario Figure 4(a), it is observed that the RMSE is consistent across the forecast horizons for each approach, maintaining relatively low values. The LUST approach presents RMSE values close to 0.005, while TAPAS

and INTAS present smaller variations, with INTAS maintaining the lowest RMSE values among the three approaches. This suggests that, in the CL scenario, the CNN+LSTM model has a stable performance in terms of prediction error, with less variability among the evaluated approaches.

In FL Figure 4(b), the RMSE shows a slight increase compared to the centralized scenario, especially in longer forecast horizons. The LUST approach, for example, achieves RMSE values around 0.007, while TAPAS and INTAS maintain a slightly lower variation. This increase in RMSE can be attributed to the division of data between devices in the federated scenario, which impacts the accuracy of the forecasts due to the heterogeneity of the local data. However, as in the centralized scenario, the INTAS approach continues to present the lowest RMSE values, indicating greater robustness of the CNN+LSTM model when applied in this method.

For SL Figure 4(c), we can see that the error remains at more intermediate levels than the previous approaches, that is, SL achieves lower error than FL, but higher error than CL. Since only part of the model is trained locally before being sent to the server for further processing, there is a possible loss of accuracy due to the fragmentation of the training. In short forecast horizons, the RMSE tends to be comparable to that of the centralized scenario, benefiting from the lower communication load. However, for longer horizons, there was a gradual increase in RMSE, although to a lesser extent than in the federated scenario, due to the coordination between the local and server stages, which may introduce a slight degradation in the accuracy of the predictions.

In summary, the federated scenario presented the lowest RMSE for the CNN+LSTM model, offering high accuracy due to the complete training on the server with aggregated data. In the federated scenario, the RMSE increases slightly, especially in longer horizons, due to the division of data between devices, which reduces accuracy. In SL, the RMSE is at an intermediate level, with performance close to the centralized one for short horizons, but a slight degradation in long horizons, since the model is trained in separate parts between the devices and the server. Each scenario, therefore, brings a different balance between accuracy and communication efficiency. Finally, we also noticed that the LUST scenario was consistently more accurate in both approaches, suggesting a superior performance in mitigating the impacts of data distribution.

Figure 5 represents the regression line for test data for each solution; We evaluate the global model on a centralized

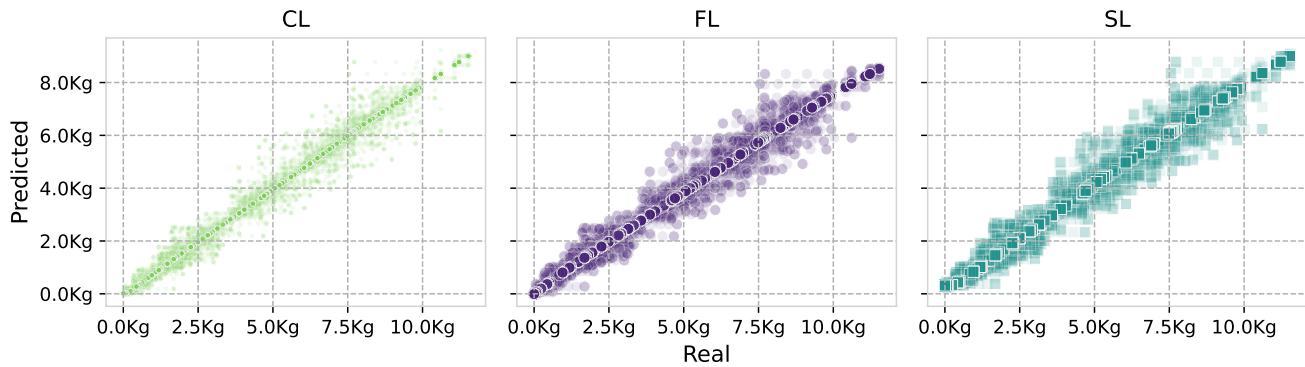


Figure 5. Predictions Comparison between Learning Approaches

set of all test data. The points closest to the diagonal indicate lower error. Note that all solutions showed promising results, indicating the model's convergence and learning. However, for higher CO₂ emission values, all three approaches start to exhibit instabilities, as seen in the 6000 to 8000 intervals. This instability suggests that further tuning may be required for improved accuracy in higher emission ranges.

5 Conclusion

In conclusion, the framework study demonstrated machine learning models' effectiveness in predicting CO₂ emissions in different vehicular mobility scenarios, using centralized, FL, and SL approaches. Simulations with different mobility scenarios (TAPASCologne, LuST, InTAS) and learning methods allowed an in-depth analysis of the performance of CNN, CNN+LSTM and LSTM models.

This work investigated the use of machine learning in vehicular networks for predicting CO₂ emissions, highlighting LSTM as the model that presented an effective combination of accuracy and efficiency, especially in federated learning. This makes it promising for applications in urban environments with communication limitations. The analyzed approaches contribute significantly to the understanding of the trade-offs between accuracy and communication overhead in vehicular networks, encouraging the development of balanced solutions between high accuracy and resource efficiency.

In future work, we intend to consider additional metrics to evaluate the framework, implement more robust algorithms for distributed methods, and explore new models and datasets that can be applied.

Declarations

Acknowledgements

This project was supported by the Green Mobility and Innovation Program (Mover), line VI, vehicular connectivity. UFMG Support Foundation (Fundep). CO2nnect Project, No. 29271*13. This work was partially sponsored by project #23/00673-7, São Paulo Research Foundation (FAPESP), CNPq project #405940/2022-0 and CAPES project #88887.954253/2024-00 and FAEPEX/PIND UNICAMP project #519.287.

Authors' Contributions

Carnot Braun: Contributed to the design of the methodology and writing of the manuscript, conducted the analysis, and implemented the data collection for each scenario. Rafael O. Jarczewski: Helped conceptualize the study and contributed to the editing of the manuscript and the training of techniques for metric approaches. Allan M. de Souza: Guided the work's structuring and the manuscript's review. All authors read and approved the final manuscript.

Competing interests

The authors declare no conflicts of interest.

Availability of data and materials

The LuST Scenario can be found at: <https://github.com/lcodeca/LuSTScenario>. Additionally, the InTAS Scenario can be downloaded via the following link: <https://github.com/silaslobo/InTAS>. Finally, the TAPAS Scenario could be downloaded at: https://sourceforge.net/projects/sumo/files/traffic_data/scenarios/TAPASCologne/. The scripts for the experiments and the data obtained are available upon request to the authors.

References

- Abedi, A., K. S. (2024). Fedsl: Federated split learning on distributed sequential data in recurrent neural networks. *Multimed Tools Appl* 83. DOI: 10.1007/s11042-023-15184-5.
- Belal, Y., Ben Mokhtar, S., Haddadi, H., Wang, J., and Mashhadi, A. (2024). Survey of federated learning models for spatial-temporal mobility applications. *ACM Transactions on Spatial Algorithms and Systems*, 10(3). DOI: 10.1145/3666089.
- Braun, C., Da Costa, J. B. D., Villas, L. A., and de Souza, A. M. (2024). Ecopredict: Assessing distributed machine learning methods for predicting urban emissions. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–5. DOI: 10.1109/VTC2024-Fall63153.2024.10757672.
- Capanema, C. G. S., de Souza, A. M., Costa, J. B. D. d., Silva, F. A., Villas, L. A., and Loureiro, A. A. F. (2024). A novel prediction technique for federated learning. *IEEE*

- Transactions on Emerging Topics in Computing*, pages 1–16. DOI: 10.1109/TETC.2024.3471458.
- Codecá, L., Frank, R., Faye, S., and Engel, T. (2017). Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63. DOI: 10.1109/mits.2017.2666585.
- Creß, C., Bing, Z., and Knoll, A. C. (2024). Intelligent transportation systems using roadside infrastructure: A literature survey. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):6309–6327. DOI: 10.1109/TITS.2023.3343434.
- de Souza, A. M., Braun, T., Botega, L. C., Cabral, R., Garcia, I. C., and Villas, L. A. (2019). Better safe than sorry: a vehicular traffic re-routing based on traffic conditions and public safety issues. *Journal of Internet Services and Applications*, 10(1):17. DOI: 10.1186/s13174-019-0116-9.
- de Souza, A. M., Braun, T., Botega, L. C., Villas, L. A., and Loureiro, A. A. F. (2020). Safe and sound: Driver safety-aware vehicle re-routing based on spatiotemporal information. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3973–3989. DOI: 10.1109/TITS.2019.2958624.
- de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462. DOI: 10.2139/ssrn.4654118.
- de Souza, A. M., Oliveira, H. F., Zhao, Z., Braun, T., Loureiro, A. A., and Villas, L. A. (2022). Enhancing sensing and decision-making of automated driving systems with multi-access edge computing and machine learning. *IEEE Intelligent Transportation Systems Magazine*, 14(1):44–56. DOI: 10.1109/MITS.2019.2953513.
- Elbir, A. M., Soner, B., Çöleri, S., Gündüz, D., and Bennis, M. (2022). Federated learning in vehicular networks. In *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 72–77. DOI: 10.1109/MeditCom55741.2022.9928621.
- Falahatraftar, F., Pierre, S., and Chamberland, S. (2021). A centralized and dynamic network congestion classification approach for heterogeneous vehicular networks. *IEEE Access*, 9:122284–122298. DOI: 10.1109/ACCESS.2021.3108425.
- Fei, X. and Ling, Q. (2023). Attention-based global and local spatial-temporal graph convolutional network for vehicle emission prediction. *Neurocomputing*, 521:41–55. DOI: 10.1016/j.neucom.2022.11.085.
- Kaur, G. (2024). Federated learning based spatio-temporal framework for real-time traffic prediction. *Wireless Personal Communications*. DOI: 10.21203/rs.3.rs-2470634/v1.
- Krajewicz, D. and et al. (2012). Recent development and applications of sumo-simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3 & 4):128–138. Available at: https://personales.upv.es/thinkmind/dl/journals/sysmea/sysmea_v5_n34_2012/sysmea_v5_n34_2012_4.pdf.
- Kumar, M. (2024). A traffic flow prediction framework based on integrated federated learning and recurrent long short-term networks. *Peer-to-Peer Networking and Applications*. DOI: 10.1007/s12083-024-01792-x.
- Lobo, S., Neumeier, S., Fernandez, E., and Facchi, C. (2020). Intas - the ingolstadt traffic scenario for sumo. In *SUMO User Conference Proceedings*. DOI: 10.48550/arXiv.2011.11995.
- Rahman, M. M. and Thill, J.-C. (2023). Impacts of connected and autonomous vehicles on urban transportation and environment: A comprehensive review. *Sustainable Cities and Society*, page 104649. DOI: 10.1016/j.scs.2023.104649.
- Saleem, M., Abbas, S., Ghazal, T. M., Adnan Khan, M., Sahawneh, N., and Ahmad, M. (2022). Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques. *Egyptian Informatics Journal*, 23(3):417–426. DOI: 10.1016/j.eij.2022.03.003.
- Samikwa, E., Di Maio, A., and Braun, T. (2024). Dfl: Dynamic federated split learning in heterogeneous iot. *IEEE Transactions on Machine Learning in Communications and Networking*, 2:733–752. DOI: 10.1109/TMLCN.2024.3409205.
- Sánchez, J. A., Melendi, D., García, R., Pañeda, X. G., Corcobá, V., and García, D. (2024). Distributed and collaborative system to improve traffic conditions using fuzzy logic and v2x communications. *Vehicular Communications*, page 100746. DOI: 10.1016/j.vehcom.2024.100746.
- Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329. DOI: 10.1109/jproc.2017.2761740.
- Uppoor, S., Trullols-Cruces, O., Fiore, M., and Barcelo-Ordinas, J. M. (2013). Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075. DOI: 10.1109/TMC.2013.27.
- Wang, H., Liu, T., Kim, B., Lin, C.-W., Shiraishi, S., Xie, J., and Han, Z. (2020). Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. *IEEE Communications Surveys Tutorials*, 22(4):2349–2377. DOI: 10.1109/COMST.2020.3020854.
- World Health Organization (2019). Air pollution. Available at: <https://www.who.int/health-topics/air-pollution>. Accessed: 2025-04-05.
- Wu, C., Liu, Z., Zhang, D., Yoshinaga, T., and Ji, Y. (2018). Spatial intelligence toward trustworthy vehicular iot. *IEEE Communications Magazine*, 56(10):22–27. DOI: 10.1109/MCOM.2018.1800089.
- Yang, W. (2022). A reinforcement learning based data storage and traffic management in information-centric data center networks. *Mobile Networks and Applications*. DOI: 10.1007/s11036-020-01629-w.
- Zhang, R., Chen, H., Xie, P., Zu, L., Wei, Y., Wang, M., Wang, Y., and Zhu, R. (2023). Exhaust emissions from gasoline vehicles with different fuel detergency and the prediction model using deep learning. *Sensors*, 23(17). DOI: 10.3390/s23177655.
- Zhang, S., Li, J., Shi, L., Ding, M., Nguyen, D. C., Tan, W., Weng, J., and Han, Z. (2024). Federated learning in intelligent transportation systems: Recent appli-

- cations and open problems. *IEEE Transactions on Intelligent Transportation Systems*, 25(5):3259–3285. DOI: 10.1109/TITS.2023.3324962.
- Zhao, H., Cheng, H., Mao, T., and He, C. (2019). Research on traffic accident prediction model based on convolutional neural networks in vanet. In *2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 79–84. DOI: 10.1109/ICAIBD.2019.8837020.
- Zheng, G., Ni, Q., Navaie, K., Pervaiz, H., Min, G., Kaushik, A., and Zarakovitis, C. (2024). Mobility-aware split-federated with transfer learning for vehicular semantic communication networks. *IEEE Internet of Things Journal*, 11(10):17237–17248. DOI: 10.1109/JIOT.2024.3360230.
- Zhou, Y., Zhang, Z., Ding, F., Ahn, S., Wu, K., and Ran, B. (2024). A deep long short-term memory network embedded model predictive control strategies for car-following control of connected automated vehicles in mixed traffic. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):8209–8220. DOI: 10.1109/tits.2024.3412329.