EcoPredict: Assessing Distributed Machine Learning Methods for Predicting Urban Emissions

Carnot Braun, Joahannes B. D. da Costa, Leandro A. Villas, Allan M. de Souza
Universidade Estadual de Campinas (UNICAMP), Institute of Computing (IC), Brazil
Contact: c255785@dac.unicamp.br, https://ic.unicamp.br/~joahannes.costa, {lvillas, allanms}@unicamp.br

***Abstract*—The growing number of vehicles has led to increased emissions of polluting gases, necessitating accurate forecasting for effective mitigation strategies and sustainable urban development. Leveraging computational resources in vehicles, this study presents a framework, called EcoPredict, for predicting CO_2 emissions in collaborative vehicular network environments. The framework implements three forms of learning methods — centralized, federated, and split — using urban sensor networks for data collection. Experiments carried out in realistic vehicular mobility scenarios demonstrate the framework's robustness and efficiency in providing real-time emission predictions. Each learning architecture has its own advantages and limitations regarding performance, training time, latency, communication overhead, and data privacy. Therefore, this work aims to assess their performance to analyze their effectiveness in urban environments.**

I. Introduction

In recent years, the number of connected vehicles has been increasing, leading to various problems such as traffic congestion and elevated emission of polluting gases [1]. In this scenario, modern vehicles now have the capability to act as active agents, collecting and sharing contextual information in their surroundings [2], [3]. The implementation of Connected and Autonomous Vehicles (CAVs) is expected to revolutionize urban mobility, potentially reducing traffic jams, energy consumption, and greenhouse gas emissions, thus helping to reduce environmental impacts [4], [5]. However, concerns regarding personal safety and privacy issues related to autonomous vehicles are also prominent [6].

Numerous works apply different strategies for traffic forecasting in CAVs, mainly highlighting the need for instantaneous predictions [7]. Considering specifically the scenario of pollutant gas emissions, accurate forecasting of carbon dioxide (CO_2) emissions is crucial for monitoring and reducing the environmental impacts of growing urban traffic [8]. However, research remains scarce for these environmental applications, especially given the current challenges and impacts in terms of data acquisition, data processing, and spatio-temporal validity of information [7]. In this sense, adapting traffic forecasting strategies for ecological applications can be crucial for city planning and environmental development in big cities [4].

One of the main approaches used for prediction is Recurrent Neural Networks (RNNs) due to their ability to handle sequential data and capture dependencies over time [9]. Additionally, the prediction process can be carried out in three ways as follows: *(1) Centralized learning* aggregates all data in a central server, offering high model accuracy due to the comprehensive dataset. However, it raises significant privacy concerns and potential data transmission bottlenecks; *(2) Distributed learning* processes data locally on edge devices, enhancing privacy and reducing latency but potentially sacrificing some model accuracy due to decentralized data handling; and *(3) Split learning* offers a middle ground by dividing the model between the edge and the server, aiming to balance the benefits of both centralized and distributed approaches.

In this sense, comparing these approaches in the context of urban mobility for CO_2 emission prediction helps address several key challenges. For example, centralized learning can struggle with data privacy and communication overheads, while distributed learning may face issues with model performance due to data fragmentation and the dynamic nature of vehicular scenarios. [10]. On the other hand, Split Learning (SL) provides an opportunity to optimize the trade-offs between data privacy, model accuracy, and operational efficiency [11]. In this way, these approaches can assist in decision-making regarding urban planning and applied depending on the objectives defined in the analyses, in terms of performance and efficiency.

Considering all the points mentioned, this paper proposes EcoPredict, a framework for comparing different learning approaches to predict CO_2 emissions in vehicular scenarios. EcoPredict employs centralized, distributed, and split learning in realistic mobility traces (Luxembourg and Cologne cities) using Simulation of Urban MObility (SUMO). Also, it considers vehicular communications infrastructures as urban sensor networks to collect vehicle CO_2 emission data, generating time series for different regions within the cities. In summary, the contributions of this work are: *(i)* proposal of a framework that employs three approaches for analyzing vehicular data in urban scenarios, contributing to better environmental management and more efficient urban transportation systems; and *(ii)* in a detailed performance evaluation with realistic mobility traces, we show that from the same source of vehicular data, valuable insights can be extracted for urban planning.

This paper is organized as follows. Section II introduces the system model, scenario description, problem definition, and EcoPredict operation. Section III discusses the performance evaluation and results obtained. Finally, Section IV presents the conclusions and future works.

II. System Overview

This section describes EcoPredict, a framework designed to predict CO_2 emissions in urban mobility environments using

100th IEEE Vehicular Technology Conference (VTC2024-Fall)

centralized, distributed, and split learning approaches.

A. System model

The system model employed in this work is composed of vehicles, communication infrastructures (*e.g.*, Roadside Units (RSUs)), and a remote server in the Internet cloud. In this scenario, vehicles move around the city and can communicate with the RSUs. Each RSU r_i, denoted as $r_i \in R = \{r_1, r_2, \ldots, r_m\}$, has its coverage area in meters, can collect data from all roads within its coverage, and has wired communication with the remote server. Each road is denoted as $s_j \in S = \{s_1, s_2, \ldots, s_n\}$. The city is divided into $|R|$ regions, where $|R| = m$ and represents the number of RSUs present in the scenario. In this scenario, RSUs act as urban sensors, which measure parameters like average speed, fuel consumption, and CO_2 emissions.

Figure 1 presents the components of the framework in detail. In essence, EcoPredict is composed of five modules, which are called phases, ranging from input data processing (Phases 1 and 2) to the service deliver (Phase 5) generated through the Machine Learning (ML) models (Phase 4). Additionally, the Context Selector module (Phase 3) provides network operators with the ability to identify the context for decision-making regarding which ML model to use, depending on the previously defined operational objectives. Considering these aspects, this work presents a case study on the application of the framework for predicting greenhouse gases in vehicular environments.

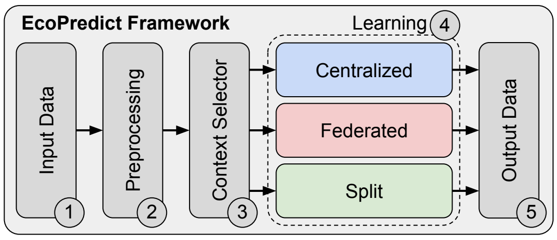


Fig. 1. EcoPredict structure with its main phases.

With the historical data gathered by each RSUs, a ML model can be trained to learn the CO_2 emissions patterns and predict how will the the emissions in the near future. One advantage of this framework is its customization capability, where the components of each phase can be modified according to the type of data being processed and the model trained. Each phase will be explained in detail below.

B. EcoPredict phases

1) Input Data: CO_2 emission data is collected by RSUs every 10 units of time (seconds, in this case), which can reported by the vehicles through beacon messages that are naturally sent in vehicular networks or by sensing and measuring the CO_2 emissions within the region. Since the data is collected at regular intervals, it produces a time series containing CO_2 emission data over time. In summary, let $Z(i) = \{z_1, z_2, \ldots, z_k\}$ be a vector that represents the collected data over time k for each RSU r_i, in which each element consist of a tuple $\{timestamp, vehicleId, vehicleRoad, co2Emission\}$.

2) Preprocessing: The preprocessing stage is fundamental in any ML model application. In our case, the steps included data cleaning, normalization, and feature extraction. First, any missing or inconsistent data points were resolved to ensure the integrity of the dataset by calculating the 60-sample moving average. Then the data is normalized to a common scale to improve model performance. One specific transformation was the conversion of raw CO_2 emission values into a time series format, enabling temporal pattern recognition essential for accurate forecasting in ML models.

3) Context Selector: The context selection phase allows for the identification of data behavior, such as trends, seasonality, stationarity, dependencies between variables, etc. Also, understanding the characteristics of the devices involved in the learning process is also an important factor at this stage. With this, network operators can determine which ML models are most suitable for the identified context. This enables more accurate decision-making to achieve the defined objectives.

4) Learning: In this phase, three different architectures can be employed to train the ML model. Figure 2 shows the learning methods that can be used, which are detailed described bellow:

- **Centralized Learning**: in this approach (Figure 2(a)), the end devices (*e.g.*, smartphones, vehicles, RSU, etc.) produce data and share it with a central model (*i.e.*, remote server). This way, the central model can be trained with all the generated data, consequently ensuring better learning. However, this architecture has substantial limitations related to latency, security, and data privacy.
- **Federated Learning (FL)**: enables an efficient and collaborative model while ensuring data security and privacy for the devices and maintaining low latency [12]. FL is an implementation of distributed learning. In this setup (Figure 2(b)), edge devices (also called clients) produce data and train a model locally. However, this model is then shared with a central server responsible for aggregating the knowledge from all received models. For model sharing, devices only share the weights or gradients learned during the training phase. This way, a new collaborative model is generated, which is then shared with the end devices again for further iterations. This process occurs over multiple stages, known as communication rounds. Therefore, this approach can produce a more robust model, as it is aggregated with the knowledge generated by all the devices in the system. Additionally, it ensures user data privacy since only models, not raw data, are shared between the devices and the server.
- **Split Learning (SL)**: allows the training of a model in a distributed manner by splitting a global model into smaller parts that are distributed to different devices. In this way, each device trains only a portion of the model and sequentially forwards the result to another device to continue the training. In this approach (Figure 2(c)), devices can either forward the results to other devices or to the server. For example, end devices RSUs can

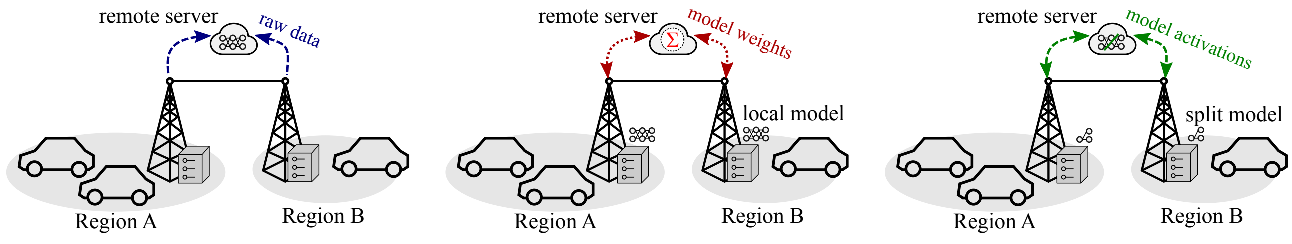


Fig. 2. Examples of the training approaches investigated. Even starting from the same scenario, each approach has its own specific operating characteristics.

execute the initial layers of the model and then send the intermediate results to other devices and/or the cloud to feed the remaining layers and produce the final results.

5) Output Data: At the end of the learning process, the output data (*i.e.* the model predictions) are provided. This information can be not only used to support better urban planning decisions, but also serve as input for intelligent services to reduce CO_2 emissions [13].

Each of the learning architecture has advantages and limitations regarding performance, training time, latency, communication overhead, and data privacy. Hence, it is important to assess their performance to analyze which one wold be better depending on the scenario that they will be employed and the restrictions regarding resource and privacy constraints.

III. System Evaluation

This section introduces the methodology and metrics used to evaluate the effectiveness of the proposed framework for predicting CO_2 emissions considering different learning architectures. The evaluation focuses on comparing the performance of the learning approaches of the proposed framework, using the Mean Squared Error (MSE) as an indicator of performance, as well as training time to assess the computational cost.

A. Scenario description and Methodology

The experiments were carried out with the Simulation of Urban MObility (SUMO) 1.16.0. The algorithms were implemented in Python 3.12.0 and connected to SUMO through the TraCI interface. To evaluate the framework in different scenarios, we considered three well-known realistic vehicular mobility traces[1]: TAPASCologne and Luxembourg SUMO Trace (LuST). Both TAPASCologne and LuST have 24 hours of vehicular mobility data. We consider 30% of the traffic in each scenario. The RSUs's positions were considered based on the OpenCelliD[2] information. Each RSU collects the data from all the roads that are under coverage. Moreover, we use a Fast Ethernet communication link of 100 Mbps between RSUs and the remote server. Therefore, as the RSU data is aggregated to create a single time series per RSU, they represent the clients in each training approach.

The data that can be collected and processed by EcoPredict is diverse, including fuel consumption, noise emission, CO_2 emission, Nitric Oxide (NOx) emission, average speed in the region covered by the RSU, among others[3]. However, in our experiments, we only considered CO_2 emissions for the case study. In this way, for training the ML models, a 65/35 approach was used for the creation of training and testing data, given the arrangement of the data during the time range. Figure 3 illustrates the division of the datasets (for each scenario.

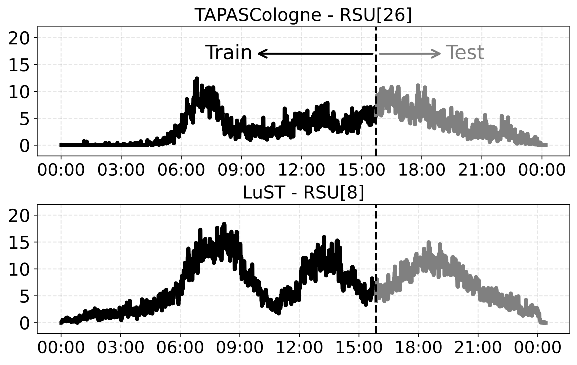


Fig. 3. Example of data (CO_2 emission in grams $\times$ 1000) of Roadside Unit (RSU) used for the training and prediction processes in each vehicular mobility scenario.

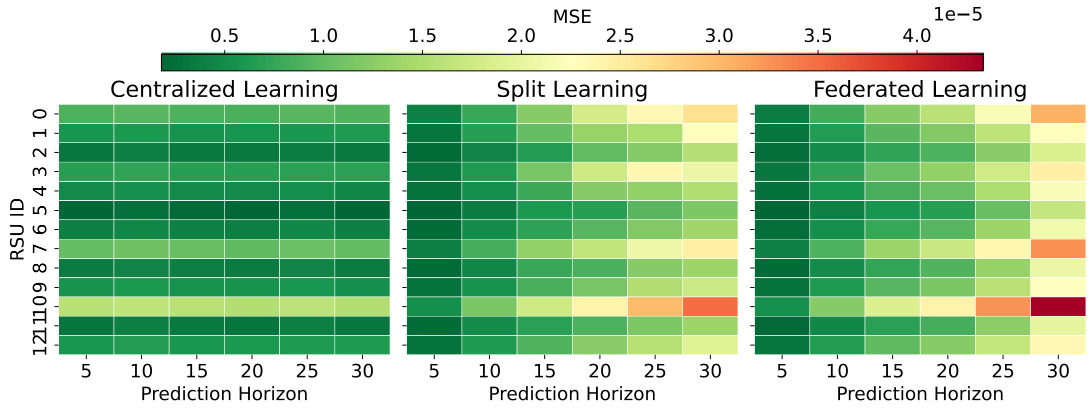
Furthermore, for the model trained, we use the following hyperparameters: The model used for predicting CO_2 emissions is a Convolutional Neural Network (CNN)-Long Short-term Memory (LSTM) implemented using TensorFlow and Keras libraries. The architecture includes one Conv1D layer with 64 filters, a kernel size of 2, and ReLU activation, followed by an LSTM layer with 50 neurons and ReLU activation, and finally a Dense layer with 1 neuron for the output. The model uses the Adam optimizer and MSE as the loss function. The output of the model is a prediction of future CO_2 emission values based on the input time series data.

Hereafter, EcoPredict fits the model through the learning architecture to predict the CO_2 emissions considering a prediction horizon. We use several prediction horizons to evaluate the performance of the model raging from 5 steps in the future to 30 steps. For the centralized architecture we considered 50 epochs to fit the model with a batch size of 64. On the other hand, to implement the distributed architectures (*i.e.* FL and SL) we use Flower framework [14] that allows

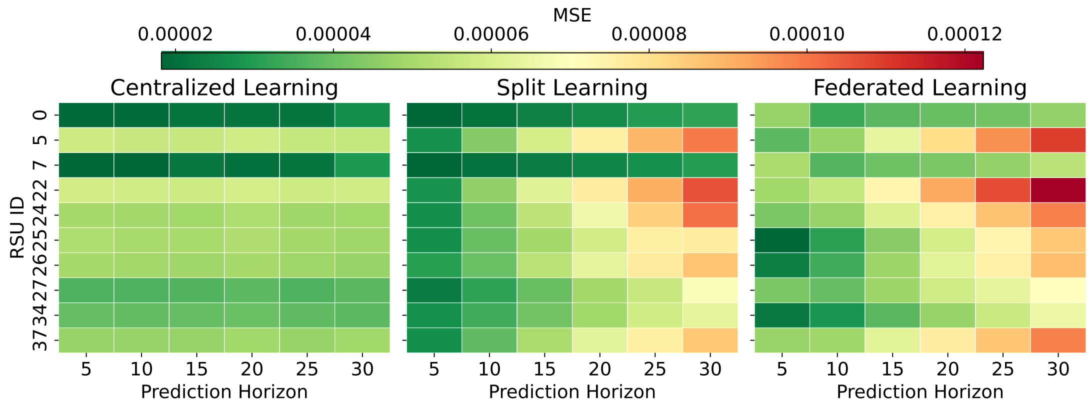
[1]https://sumo.dlr.de/docs/Data/Scenarios.html

[2]https://www.opencellid.org/

[3]https://sumo.dlr.de/docs/Simulation/Output/



(a) Average Mean Squared Error (MSE) for each Roadside Unit (RSU) in LuST scenario.



(b) Average Mean Squared Error (MSE) for each Roadside Unit (RSU) in TAPASCologne scenario.

Fig. 4. Comparison of the average MSE for each RSU in LuST (a) and TAPASCologne (b) scenarios considering each one of the learning architectures.

the communication between the clients and the server through gRPC. In particular, to implement the split-learning architecture, we adapt the framework to work as desired, thus we kept the CNN at each client and the LSTM and Dense layer in the server. In this way, clients have to pass their local data through the CNN layer to produce the activation's and send them to the server, then the server passes the activations received from each client to the LSTM and dense layer to finalize the feed-forward process. After that, the server is responsible to measures the error of the model and start the back-propagation to adjust the gradients of the entire model, thus the gradients must be sent to the client to finish the training process. Finally, FL we considered 5 training rounds and 10 local epochs and for SL 50 communications between client and server were performed to train the model.

B. *Learning Architecture Analysis*

To evaluate each learning architecture for predicting CO_2 emissions, we considered the error in predictions based on MSE and training time. Figure 4 illustrates the prediction performance (*i.e.*, error) for some RSUs in each scenario, considering centralized, split, and federated learning architectures. Each line cell in the figure represents the average error for one RSU considering a specific prediction horizon, with the cell color representing the MSE for that configuration.

As expected, the centralized approach achieved the best results in both scenarios, followed by the SL and FL architectures. Despite the better performance in terms of loss values, the centralized architecture suffers from the high volume of data that needs to be fed into the model. All data from the RSUs must be sent to the cloud, consequently increasing the communication overhead and training time, as shown in Table I. Moreover, for other types of services that may contain sensitive data, this approach also presents privacy issues.

TABLE I
MEAN TRAINING TIME IN EACH SCENARIO.

Scenario	Learning approach		
	Centralized	Federated	Split
LuST	**6270.33** $\pm$ **25.44**	886.9 $\pm$ 19.25	4548.59 $\pm$ 40.79
Cologne	**5077.50** $\pm$ **66.11**	977.3 $\pm$ 50.1	3955.89 $\pm$ 30.67

On the other hand, the SL approach maintains similar performance in terms of MSE. However, it potentially increases the communication overhead and training latency due to the communication between clients and the server required during the feed-forward and backpropagation process. In our experiments, we split the model into two parts (*i.e.*, the CNN layer was placed on the clients while the LSTM and Dense layers were placed on the server), thus not introducing significant overhead in the process. However, this approach compromises privacy, as data must be sent to the server to compute the error and initiate backpropagation. This issue could be mitigated by placing the error estimation on the client side. However, this would also increase the communication overhead, as additional communication would be required for each iteration. It is important to note that the SL approach is sequential, meaning that the RSUs are trained one at a time. Consequently, the latency introduced in the system may affect the last clients in the queue, as they need to wait for the training process of all preceding clients to complete.

TABLE II
MEAN LOSS VALUE BY HORIZON IN EACH APPROACH/SCENARIO.

Scenario	Learning approach			
	Horizon	Centralized	Federated	Split
LuST	5	2.23E-06 ± 1.57E-06	2.96E-06 ± 1.58E-06	2.33E-06 ± 9.76E-07
	10	5.95E-06 ± 3.63E-06	6.56E-06 ± 2.94E-06	5.56E-06 ± 2.14E-06
	15	8.64E-06 ± 5.42E-06	9.78E-06 ± 4.31E-06	9.01E-06 ± 3.29E-06
	20	1.15E-05 ± 3.23E-04	1.22E-05 ± 4.67E-06	1.26E-05 ± 4.94E-06
	25	1.50E-05 ± 1.13E-05	1.71E-05 ± 6.78E-06	1.61E-05 ± 6.14E-06
	30	1.71E-05 ± 2.03E-04	2.45E-05 ± 1.37E-05	1.88E-05 ± 6.69E-06
Cologne	5	3.39E-05 ± 2.73E-06	2.69E-05 ± 5.37E-05	2.84E-06 ± 3.89E-06
	10	2.10E-05 ± 2.24E-05	2.29E-05 ± 2.73E-05	6.41E-06 ± 8.57E-06
	15	3.35E-05 ± 3.91E-05	2.69E-05 ± 3.59E-05	1.00E-05 ± 1.32E-05
	20	4.74E-05 ± 3.93E-05	3.06E-05 ± 3.81E-05	1.35E-05 ± 1.78E-05
	25	5.74E-05 ± 1.14E-05	3.38E-05 ± 3.33E-05	1.71E-05 ± 2.25E-05
	30	6.62E-05 ± 4.02E-05	3.89E-05 ± 3.59E-05	2.04E-05 ± 2.63E-05

Finally, FL not only reduces the communication overhead due to fewer interactions between clients and the server, but also reduces the training overhead as multiple clients (*i.e.*, RSUs) are trained in parallel (see training time in Table I). In addition, since no data is sent to the cloud, it ensures the privacy of the clients, which can be crucial in some applications [13]. Although FL enabled a reduction in communication and training overhead, it also slightly reduced the performance of the model compared to other approaches. However, this decrease is very small, as presented in Table II, and thus is unlikely to significantly degrade the performance of the system using this approach.

IV. CONCLUSION

This work assesses the performance of distributed and centralized Machine Learning (ML) methods for predicting urban emissions. A framework is described that allows for the implementation and evaluation of centralized, Federated Learning (FL), and Split Learning (SL) architectures in urban scenarios, including TAPASCologne and LuST. Each learning architecture was analyzed by considering the quality of the predictions in terms of errors (*i.e.*, Mean Squared Error (MSE) loss) and the computational overhead required to train the model. Each learning architecture has its own advantages and limitations regarding performance, training time, latency, communication overhead, and data privacy. Hence, it is important to assess their performance to determine which architecture is best suited for specific scenarios, considering the resource and privacy constraints.

As future work, we will consider additional metrics to evaluate the framework, implement more robust algorithms for distributed methods, and explore other models and data that can be used.

ACKNOWLEDGMENTS

This project was supported by the brazilian Ministry of Science, Technology and Innovations, with resources from Law nº 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024-10.

REFERENCES

[1] J. A. Sánchez, D. Melendi, R. García, X. G. Pañeda, V. Corcoba, and D. García, “Distributed and collaborative system to improve traffic conditions using fuzzy logic and v2x communications,” *Vehicular Communications*, p. 100 746, 2024.

[2] J. B. D. da Costa, A. M. de Souza, R. I. Meneguette, E. Cerqueira, D. Rosário, C. Sommer, and L. Villas, “Mobility and deadline-aware task scheduling mechanism for vehicular edge computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, pp. 11 345–11 359, 2023.

[3] Y. Xue, L. Wang, B. Yu, and S. Cui, “A two-lane car-following model for connected vehicles under connected traffic environment,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 7445–7453, 2024.

[4] M. M. Rahman and J.-C. Thill, “Impacts of connected and autonomous vehicles on urban transportation and environment: A comprehensive review,” *Sustainable Cities and Society*, p. 104 649, 2023.

[5] Z. Lv and W. Shang, “Impacts of intelligent transportation systems on energy conservation and emission reduction of transport systems: A comprehensive review,” *Green Technologies and Sustainability*, vol. 1, no. 1, p. 100 002, 2023.

[6] V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, “Federated learning for connected and automated vehicles: A survey of existing approaches and challenges,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 119–137, 2024.

[7] X. Fei and Q. Ling, “Attention-based global and local spatial-temporal graph convolutional network for vehicle emission prediction,” *Neurocomputing*, vol. 521, pp. 41–55, 2023.

[8] R. Zhang *et al.*, “Exhaust emissions from gasoline vehicles with different fuel detergency and the prediction model using deep learning,” *Sensors*, vol. 23, no. 17, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/17/7655.

[9] Y. Zhou, Z. Zhang, F. Ding, S. Ahn, K. Wu, and B. Ran, “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*, vol. 25, no. 7, pp. 8209–8220, 2024.

[10] Y. Belal, S. Ben Mokhtar, H. Haddadi, J. Wang, and A. Mashhadi, “Survey of federated learning models for spatial-temporal mobility applications,” *ACM Transactions on Spatial Algorithms and Systems*, vol. 10, no. 3, 2024.

[11] S. Zhang *et al.*, “Federated learning in intelligent transportation systems: Recent applications and open problems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 5, pp. 3259–3285, 2024.

[12] A. M. de Souza, F. Maciel, J. B. da Costa, L. F. Bittencourt, E. Cerqueira, A. A. Loureiro, and L. A. Villas, “Adaptive client selection with personalization for communication efficient federated learning,” *Ad Hoc Networks*, vol. 157, p. 103 462, 2024.

[13] A. M. de Souza, T. Braun, L. C. Botega, L. A. Villas, and A. A. F. Loureiro, “Safe and sound: Driver safety-aware vehicle re-routing based on spatiotemporal information,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3973–3989, 2020.

[14] D. J. Beutel *et al.*, “Flower: A friendly federated learning research framework,” *arXiv preprint arXiv:2007.14390*, 2020.