# Proposing a framework for evaluating learning strategies in vehicular CPSs



Citation for the original published paper (version of record):
Havers, B., Papatriantafilou, M., Koppisetty, A. et al (2022). Proposing a framework for evaluating learning strategies in vehicular CPSs. Middleware 2022 Industrial Track - Proceedings of the 23rd International Middleware Conference Industrial Track, Part of Middleware 2022: 22-28.
http://dx.doi.org/10.1145/3564695.3564775



(article starts on next page)

# Proposing a Framework for Evaluating Learning Strategies in Vehicular CPSs

Bastian Havers
havers@chalmers.se
Volvo Cars, Chalmers University
Sweden

Marina Papatriantafilou
ptrianta@chalmers.se
Chalmers University
Sweden

Ashok Koppisetty
ashok.chaitanya.koppisetty@volvocars.com
Volvo Cars
Sweden

Vincenzo Gulisano
vincenzo.gulisano@chalmers.se
Chalmers University
Sweden

## ABSTRACT

Highly-connected Vehicular Cyber-Physical Systems (VCPSs) offer manifold opportunities for distributing learning across the contained vehicles, road-side units and servers. However, simulating and evaluating particular distributed learning schemes poses a difficult problem in requiring realistic modeling of the vehicular fleet, communication, and the learning itself. In this work, we postulate a set of requirements for a framework simulating a complete learning workflow in a VCPS, and propose a modular architecture for it. Using a prototype implementation, we show with an example experiment the capabilities the proposed framework delivers for evaluating novel learning schemes in custom scenarios.

## CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; **Simulation tools**; • **Computer systems organization** → **Embedded and cyber-physical systems**.

## KEYWORDS

Vehicular Cyber-Physical Systems, Distributed Learning, Machine Learning, Simulation

**ACM Reference Format:**
Bastian Havers, Marina Papatriantafilou, Ashok Koppisetty, and Vincenzo Gulisano. 2022. Proposing a Framework for Evaluating Learning Strategies in Vehicular CPSs. In *23rd International Middleware Conference (Middleware '22), November 7–11, 2022, Quebec, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3564695.3564775

## 1 INTRODUCTION

Using Machine Learning (ML) in Vehicular Cyber-Physical Systems (VCPSs) of smart and connected vehicles has shown manifold valuable applications like object recognition from visual data [1], vehicle maneuver planning [16, 23], or driver intention recognition [2, 28]. Applications like these for smart driver assistance functionality are projected to deliver fully autonomous driving in the near future. As the data fed to the underlying ML models is usually sensed by vehicles while the training of such models is performed centrally, vehicle manufacturers and fleet operators need to retrieve raw data from the vehicles. This can happen either via sporadic physical access to vehicles or frequent wireless transmission. While the latter is preferable to access fresh data, it nonetheless incurs variable costs for cellular broadband usage[1] (this effect can be alleviated through smart preprocessing/compression on the edge [14, 24], but the issue of scaling remains ). Notice that both approaches share the risk of potential exposure of sensitive user data at the central data center, which should be minimized according to privacy regulations such as GDPR. Central data gathering can thus hinder scalability of the training of models going forward due to rising transmission costs and legal requirements. Such shortcomings can be alleviated by alternatives utilizing the computational power on the system's *edge*, i.e., the vehicles' onboard devices.

The spectrum of those alternatives spans from Federated Learning (FL) [6, 8, 10, 20], where a central server aggregates only model parameters (instead of raw data) from the edge, to Gossip Learning (GL) [18, 26, 30], in which devices communicate their models directly with each other without central coordination. Comparing which of these approaches best suits the needs of a VCPS in question can be difficult due to the variable system dimensions. Leaving out the option of data collection via physical access due to its impracticality (e.g. stale data, hard to scale to large fleets), some of such dimensions include vehicular on-board capabilities, available communication channels (e.g. can vehicles employ vehicle-to-vehicle communication or only vehicle-to-cloud?), individual vehicle usage patterns dictating when vehicles are turned on and how they are moving about (influencing for example network coverage at a vehicle's location or proximity to other vehicles), the data distribution in the fleet [9] and fleet size. Taken together, these forbid a one-size-fits-all solution to decentralized learning in a VCPS.

It is costly, potentially slow, risky, and impractical to evaluate decentralized learning approaches directly in the actual fleet due to, among others, the complexity and safety requirements surrounding



*Middleware 2022, November 7 – 11, 2022, Québec City, Québec, Canada*

ACM ISBN 978-1-4503-9917-3/22/11…$15.00
https://doi.org/10.1145/3564695.3564775

[1]While cellular broadband costs have fallen sharply over the past years [37], the data amounts required to train well-performing modern ML models such as Deep Neural Networks (DNNs) over multi-dimensional data (e.g., images) are growing approximately in proportion to increasing model complexity [12], potentially offsetting decreased transmission costs per byte sent.

on-board software. Thus, a framework able to accurately simulate learning processes inside a VCPS, allowing the extraction of custom metrics that make various learning approaches comparable and parameterizable at lower costs and higher speeds, is needed before turning to real-world test runs.

In this paper, we collate the requirements such a framework needs to fulfill in order to help OEMs and fleet operators in testing and evaluating practical alternatives to centralized ML, propose a general modular architecture for such a framework to guide its design, and show example results obtained with a prototype implementation that we have developed.

## 2 RELATED WORK

To the best of our knowledge, no single framework exists that answers all challenges mentioned in § 1. Nonetheless, many complementary tools have been developed. Since we aim at discussing the challenges and needs of a comprehensive framework rather than specific implementations of it, we list and discuss some relevant related examples here.

There exist several established tools to simulate vehicular traffic, for example SUMO [19] and VISSIM [11]. Coupled with network simulators such as OMNet++ [34] or NS3 [4], these can yield combined tools to simulate and evaluate Vehicular Ad-Hoc Networks (VANETs). An overview of such combined tools can be found in [36], for example VEINS [31] or ezCar2x [29]. These tools both enable the testing of applications relying on vehicle-to-vehicle and vehicle-to-cloud communication and could serve as a basis for the framework proposed in this work. However, their starting point is the simulation of vehicles' trajectories, while fleet operators and vehicle manufacturers typically have access to unbiased real-world vehicle trajectories and may thus not require full-blown traffic simulations, which may add extra overhead. It should be noted that SUMO allows to generate routes from existing GPS data and can thus allow experimentation exclusively with real data.

Several frameworks focus solely on the aspect of learning in a distributed system, omitting any connection to VCPSs. Besides the lacking ability to implement vehicular dynamics directly, they offer no direct or only limited support for more advanced or hybrid learning strategies. Flower [3] is an open-source framework that allows to implement and experiment with various flavors of FL on actual edge devices and single-machine setups and is flexible in its support for various ML frameworks. However, its focus lies on supporting FL only; thus, support for other types of distributed learning such as GL is lacking. More straightforward tools such as FLSim [35] offer, after some customization, a stripped-down but also less feature-rich experimentation with FL strategies. The framework TensorFlow Federated [33] offers "Federated Analytics" functionality for more varied federated computations. However, for implementing strategies such as GL [15], no singular framework (comparable for example to Flower) could be found by the authors of this paper.

In [7], the authors explore and evaluate one approach to GL in a VCPS by combining the SUMO traffic and OMNet++ communication simulators with the Keras ML framework [5]. However, as that work aims at evaluating only GL, it remains open if their experimentation framework would suit the needs of a more flexible tool that allows even for hybrid solutions and simulated on-board deployment.

In summary, while all elementary parts of the sought-after framework exist, no single tool covers all required dimensions in the appropriate depth, and the complete design of such a unifying framework remains open.

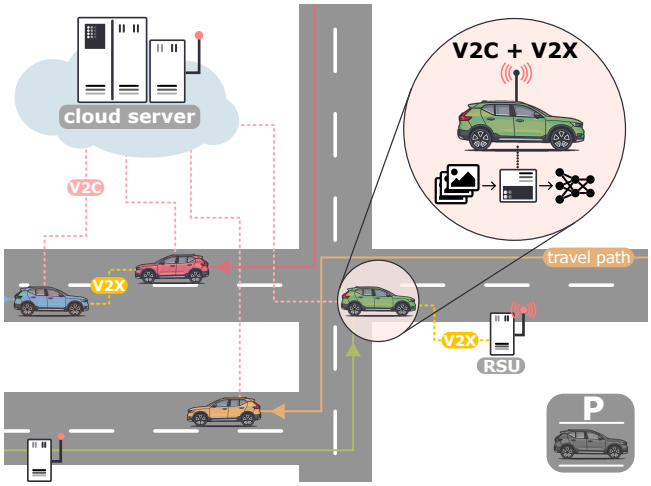


**Figure 1: Sketch of a VCPS. Via V2C (vehicle-to-cloud), any car can communicate with the cloud server, while V2X (vehicle-to-anything) is local-only: between cars, and cars and road-side units (RSUs). Each car (see inset) collects data to train an ML model (generally, also cloud server and RSU are capable of training). Vehicular dynamics are dictated by their travel paths. Cars that are turned off (see: parked grey car) temporarily do not partake in the VCPS.**

## 3 PROBLEMS AND REQUIREMENTS

Offering a framework to aid in developing, evaluating, and optimizing strategies for learning from data gathered on the edge of a VCPS poses several challenges. Such a framework must be able to take the specifics of a highly evolving, connected vehicular fleet into account both on the level of the individual vehicle and the whole fleet. Furthermore, sufficiently realistic modeling of various forms of communication between various actors in the system is required, as well as modeling of the learning aspect itself. Eventually, all these aspects have to be simulated by taking into account the real-time behavior of the various components.

The learning VCPS and the contained simulated agents are sketched in Figure 1: Vehicles, equipped with an on-board unit to transform data into an ML model (see inset), a cloud server, connected to the vehicles via a V2C (vehicle-to-cloud) connection, and RSUs (Road-Side Units) that can communicate with the vehicles via short-range V2X (vehicle-to-anything) and the cloud server via a wired connection.

*Preliminaries.* To define the requirements of the framework in detail, we introduce a few key concepts: A *learning problem* defines a real-world problem an analyst wants to solve, e.g. the predictive maintenance of a certain component of the vehicle. Solving this

learning problem requires the collection of relevant data by the vehicle to learn from, with the distribution of data potentially varying strongly between vehicles. This data is then processed using *Machine Learning*, which creates and iteratively updates an ML model of the data, using techniques that span from *supervised* ones (where ground truths for each data instance are accessible, e.g. generated by humans or sensors) to *semi-supervised* or *unsupervised* ones (where no ground truth exists, for example when trying to identify anomalies or when clustering data). The learning problem is solved once one can make predictions with the ML model satisfying some requirements for the prediction's accuracy (for supervised learning, this can be measured through *testing* by for example the ratio of correct vs. wrong predictions or a prediction's closeness to a ground truth; for unsupervised learning, this could be a measure for the performance of the clustering).

How the learning problem is solved is defined through the *learning strategy*, describing how data and models are exchanged between vehicles and other actors in the system, and how and where models are trained and potentially tested. Two example learning strategies for supervised learning are the following[2]:

*Federated Learning*

The strategy proceeds in rounds. In each round, the cloud server selects a subset of vehicles and transmits to them a so-called global model. Each receiving vehicle $v_i$ uses its local data to fine-tune (retrain) the global model locally, then sends the retrained model $w_i$ back to the cloud server. The latter aggregates the received models into a new global model $w$, using for example Federated Averaging: $w = \sum_i w_i \cdot d_i / (\sum_j d_j)$, where $d_k$ is the data amount on vehicle $k$ (as presented in [22]).

*Opportunistic ~*

Each vehicle $v_i$ begins by training its own local model. Upon getting close in space to another vehicle $v_j$, both vehicles exchange their models $w_i, w_j$, retrain the received model, and send it back to the sender, who aggregates the received model with its original model. Thus, each vehicle plays the role of a cloud server in FL for all vehicles in its vicinity (as presented in [7]).

A major aspect of a VCPS affecting the feasibility of various learning strategies is the different modes of communication between actors. We differentiate two main types:

a) *Long-range cellular* ***Vehicle-to-Cloud (V2C)***: Vehicles can employ metered cellular connections to connect with cellphone towers, using e.g. 4G/LTE or 5G [21] communication technology. As cellphone towers are connected to the Internet, V2C allows vehicles to communicate with servers located at the vehicle manufacturers. Communication speeds achievable via V2C are the same as those achievable by mobile phones using the same cellular technology and can range from 1000 to more than 10000 KB/s in ideal conditions. As sketched in Figure 1, the cloud server can, barring coverage issues stemming from e.g. tunnels, connect to any vehicle that is turned on.
b) *Short-range* ***Vehicle-to-X (V2X)***: This second family of connections is more short-ranged. It encompasses various standards such as *IEEE 802.11p* (relying on WiFi) or the more recent *Cellular-V2X* (relying on 4G/5G) to enable vehicle-to-vehicle communication (V2V) or the communication via and with roadside units (RSUs). Line-of-sight range of these connections can exceed 1000 m, although this range is reduced in the presence of obstacles [27].

[2]Note that, implicitly, it is assumed that in these approaches the ground truth of the learning problem can be generated on the vehicles themselves

The viability of V2X communication is strongly dependent on the vehicles' *spatial dynamics*, i.e., how and when they move along a given network of roads to come into proximity of other vehicles and RSUs. As an example, Figure 1 shows each vehicle's travel path, dictating encounters in the VCPS.

*Requirements list.* Using the above concepts and definitions, the framework should eventually allow the user to evaluate, for a given learning problem, spatial dynamics and modes of communication, a specific learning strategy using metrics relevant to the user.

Based on our own experiments with evaluating and refining learning strategies in a VCPS (for example as part of the Edge Lab initiative [32]), as well as reports and research from various OEMs [8, 13], we have postulated the following key requirements for the sought-after framework:

(1) *Realistic fleet model.* The framework needs to be able to model the dynamics of each vehicle in a fleet, consisting of the vehicle's spatial trajectory as defined by real mobility data and its current state, which is depending on the state of the other system actors as well as external factors (e.g., a vehicle could be turned off during the system's evolution by the driver, making it unavailable).
(2) *Realistic ML support.* To solve the learning problem, the actors in the system need to be able to perform the training, exchange, and testing of ML models, and do this in accordance with real-world hardware capabilities of modern vehicles. Support for various types of ML models is required to be able to tackle the manifold learning problems arising in VCPSs, and to handle various data distributions and preprocessing steps that the learning problem dictates.
(3) *Realistic communication model.* Taking into account the two forms of communication laid out previously, the framework has to realistically model how vehicles communicate with each other, with RSUs, and with the central server (the specifics of this are defined by the VCPS at hand and by the intended learning strategy). Depending on the state and the location of actors, communication may or may not be possible at a given point in time, and may fail at any time.
(4) *Fine-grained metrics.* The set of supported metrics should encompass the accuracy of the ML models in the system at various points in time and the volume of communication transmitted via the various communication channels, to enable a thorough evaluation of various learning strategies. The implementation of more custom metrics should be possible, such as computational workloads of individual vehicles or the provenance of data.
(5) *Flexible learning strategy support.* The framework should allow the flexible implementation and parametrization of learning strategies to allow for easy experimentation and iteration. This means supporting centralized ML, FL, GL, as well as hybrid approaches.

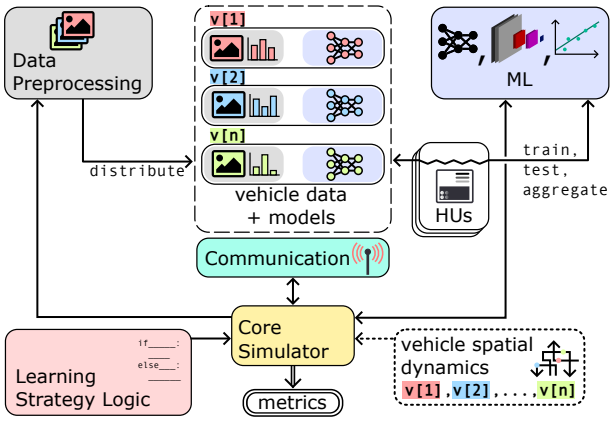


**Figure 2: Proposal of the framework's modular architecture, consisting of various modules (colored boxes, see § 4) centered around a Core Simulator (HU: Hardware Unit; v[i]: *i*-th agent).**

(6) *Quick execution.* The framework should realize a significant speed-up over an experiment in a real VCPS and reduce unnecessary overheads to allow quick experiment repetition when varying learning strategies.

With these requirements defined, we will in the following section present a proposal for a framework architecture to tackle the ensuing challenges.

## 4 ARCHITECTURE PROPOSAL

The architecture proposed here is built around a Core Simulator, providing the elementary functionality of creating virtual agents and then proceeding in discrete steps through the simulation time, separated from the modules relevant to the learning problem to increase flexibility and usability. Furthermore, modules relevant to the learning problem may be designed as targeting the hardware platform of real connected vehicles, while the Core Simulator can be executed on off-the-shelf hardware. Figure 2 details the architecture proposal: At the center sits the aforementioned *Core Simulator* that orchestrates the remaining modules, starting with the *Data Preprocessing* module. It provides the data residing on each of the $n$ simulated agents, here designated $v[1], v[2], \ldots, v[n]$ (these agents are vehicles, road-side units, or the cloud server, see Figure 1). As an example, for the problem of recognizing road signs from traffic scenes, this module could crop and resize a given input data set of images of traffic scenes (e.g. recorded by actual vehicles), split the dataset into $n$ subsets according to a predefined distribution, and assign each subset to a simulated vehicle as well as a test set to the simulated cloud server, all according to the specification of the learning problem at hand.

The *ML module*, likewise, keeps tabs on the current model(s) of each agent in the system (not all actors may have their own model), and provides functionality to train and test any model with any data and to aggregate models into new ones and assign these to certain agents. For example, in the aggregation step of Federated Learning, the ML module may read the models of a subset of vehicles, perform a weighted average of these (e.g. by the data amount on each vehicle in the subset [22]), and assign the result as the new model of the cloud server. The ML module deploys these operations to one or more *HU* (Hardware Unit), instances of the actual hardware existing within vehicles that allows achieving realistic performance and training times (while an agent is busy training, it may not be available for other operations). When not impacting performance, the HUs can run multiple operations in parallel to speed up the simulation, e.g. simultaneously training multiple agents' models. Note that the HU corresponds to the training-capable simulated agent, e.g. a vehicular on-board unit (OBU) or server hardware. Additionally, the ML module exposes metrics about the accuracy of various models in the simulated system.

Communication between agents is handled in the core simulator by a *Communication* module, which for example could be a communication simulator in itself (as in [31]). This module models the transmission of various types of data in the system per the communication type's properties, impacting the bandwidth and range of communication (for V2X, the range can be quite limited and highly dependent on the involved vehicle's position, requiring the core simulator to pass trajectory data to the Communication module). The Communication module also keeps track of the data volumes transmitted and exposes this metric to the Core Simulator. In the aforementioned FL aggregation step, the Communication module would simulate cellular transmission of the models between involved vehicles and the cloud server, covering both the transmission duration and its potential (partial) failure for vehicles that are unreachable or turning off.

*Vehicle spatial dynamics* enter the Core Simulator statically, e.g. as a file of GPS traces of all moving agents in the system. This supports the use of historic GPS data, but also of simulated data (pre-calculated with e.g. SUMO). As the act of learning in the VCPS is assumed to not influence individual vehicles' trajectories, it is sufficient that the spatial dynamics data of the VCPS is replayed by the core simulator.

The interaction of all modules of the system is parameterized by a set of rules given in the *Learning Strategy Logic* module, defining how the agents react in which situation and thus encoding the learning strategy that is to be tested in a certain experiment run. A comprehensive example of a learning strategy, together with an evaluating experiment, can be found in section § 5.2.

Eventually, the Core Simulator outputs an experiment run's metrics timestamped in simulated time to enable analysis of the system's evolution under a learning strategy.

## 5 A PROTOTYPE IMPLEMENTATION: ROADRUNNER

### 5.1 Implementation details

Having presented a proposal for the architecture of the desired framework in § 4, we will in this section present a prototype implementation, Roadrunner, that we have used for initial experiments with various learning strategies.

Roadrunner's Core Simulator is written in Java and based on a messaging scheme between simulated agents. The Core Simulator uses user-defined network speeds for the two communication types, V2C and V2X. Message transmission fails if agents are not in the appropriate state (e.g. V2X messages can only be exchanged if the

participants are within range of each other, and a vehicle shutting off will result in any incoming or outgoing message failing). Vehicles' spatial trajectories are read from an input file, and at each point in simulated time, the Core Simulator will change the state of participating agents according to their current position and state (i.e., once they have been turned on or off).

The Data Preprocessing and ML modules are written in Python, based on the ML framework PyTorch [25]. The latter modules were inspired by the work of [35], and further modularized, extended, and converted into individual scripts that operate on vehicle data and models stored as files on disk. To interface with the Data Preprocessing and ML modules, the Java Core Simulator calls appropriate scripts. These perform, for example, the initial distribution of data onto the agents by splitting an original dataset into subsets and storing each subset, assigned to a particular agent, on disk, or the training of some agent's model by reading its data and model file and performing the training operation. The scripts time the duration of their execution and pass this value to the Core Simulator to appropriately model the time spent by agents in various states. The ML computations themselves are executed on a GPU as an OBU stand-in (as GPUs are expected standard hardware on smart vehicles) using build-in functionality of PyTorch[3].

Using the logging tool Log4j, metrics are continuously extracted from the simulation to represent the state of every actor, the accuracy of the ML models in the simulation, and data transmission volumes at every point in simulated time.

## 5.2 Sample Experiment: Testing an opportunistic learning strategy

To exemplify the understanding our framework can enable, we show an experiment from a real-world example. As shown in early works on FL [22] (see § 3 for a primer on the FL learning strategy), increasing the number of participants in an FL round can be one way to increase the accuracy of the final model. However, when deploying FL in a VCPS, and connecting cloud server and vehicles with a V2C connection, contacting additional vehicles per round results in increased cellular costs. Inspired by Opportunistic Learning (see § 3), we explore the addition of V2X to increase the number of vehicles reached in each round. FL uses Federated Averaging (FA, see § 3), which is mathematically associative, to aggregate a new model through *intermediate aggregation* (see Figure 3). Around this idea, we designed the learning strategy `OPP` (opportunistic):

**Server**: Send latest global model $w$ to $R$ random vehicles ("reporters") via V2C, start round timer. At end of round, request new models from reporters. Aggregate received models into new global model via FA, then start next round.
**Reporters**: Upon receiving $w$ from server, retrain $w$. Upon opportunistically meeting a non-reporter vehicle, send to it $w$ via V2X. Wait to receive back retrained model and aggregate it with own model via FA, to replace the own model. At end of round, send own model back to the server.
**Non-reporters**: Upon receiving $w$ via V2X from nearby reporter, retrain $w$. Send it back to reporter after retraining, if reporter is still in range. Else, discard $w$.

Thus, when during some round each of the $R$ reporters aggregates

[3]See https://pytorch.org/docs/stable/cuda.html

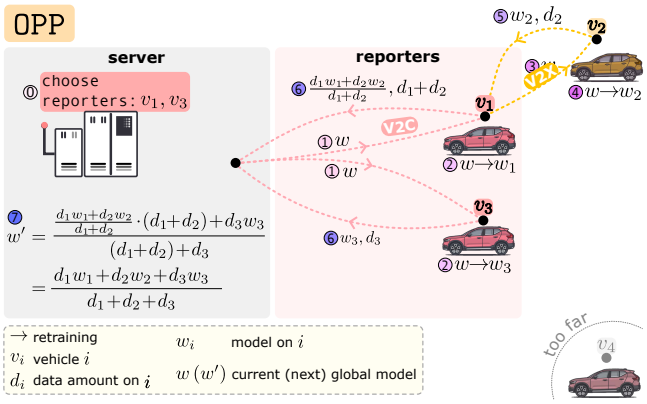


**Figure 3: A single round of `OPP` with two reporters ($v_1, v_3$) and one non-reporter ($v_3$). Encircled numbers indicate the order of events. $v_4$ is too far from the reporters to participate.**

the models from on average $NR$ non-reporters, this learning strategy results in $N = R \cdot (NR + 1)$ model contributions in that round, but requires only $R \leq N$ connections via V2C[4].

Figure 3 presents a sketch of a single round of `OPP` for two reporters: The circled numbers in the figure indicate the order of events (0-7) and black right-arrows indicate retraining. $d_i$ are the data amounts used for retraining on each vehicle. In detail: (1) the model $w$ is sent out to the reporters $v_1, v_3$, which retrain the model using their local data (2). (3) meeting non-reporter $v_2$, $v_1$ forwards the model $w$ there (via V2X), where it is retrained to $w_2$ (4) and sent back together (via V2X) with the training data amount $d_2$ (5). Then (6), $v_1$ transmits the intermediate aggregate of its own and the model from $v_2$ to the server, and $v_3$ does the same. Finally (7), this intermediate aggregation yields a new model $w'$ that is identical to the case in which all three involved vehicles are reporters. Intuitively, this approach should improve performance, given that the reporters get close to other vehicles during their trajectory for times long enough to facilitate the exchange (e.g. in the sketch, $v_4$ is too far away from any reporter), making this approach highly dependent on the density of vehicles. Furthermore, it is intuitive that a longer round duration will give more opportunities for local aggregation of weights. Simultaneously, it will also increase the duration of the whole learning process, and increase the probability that a reporter vehicle is turned off by the driver before a round ends, effectively discarding the models collected by this reporter.

*Experiment Setup.* To test whether these intuitions are correct, we use Roadrunner. In the following experiment, we assume a fixed V2C communication budget dictating the number of learning rounds we can perform. As a baseline case `BASE`, we perform FL in the VCPS, contacting 5 vehicles each round over 75 rounds of 30 seconds duration. In `OPP`, we also designate 5 reporters per round for 75 rounds (thus using the same V2C communication budget); however, we let reporters try to exchange their weights with

[4]In the worst case, no reporter will meet another vehicle for long enough to facilitate a successful exchange of weights, thus $NR = 0$ and $R = N$. Note that we disregard the case of reporters turning off during a round in this inequality, as that would also impact standard FL

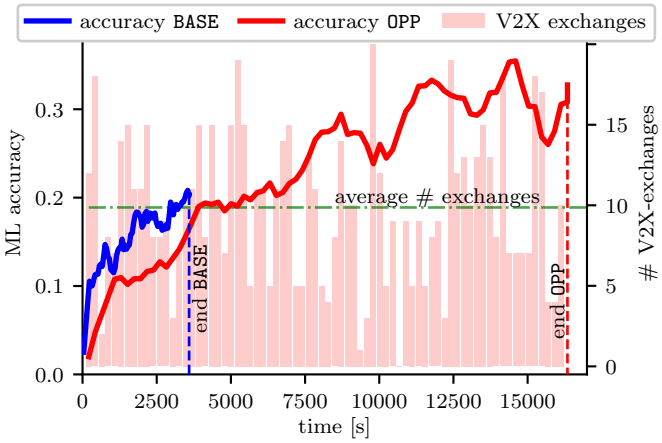


**Figure 4: Evaluation experiment of learning strategy OPP using Roadrunner: The blue and red solid curves show the accuracy of the global model of BASE and OPP, respectively, and the bar plot shows the number of V2X exchanges that occur during a given round of OPP. Additionally indicated in the figure are the average number of V2X exchanges and the points in time at which 75 rounds of BASE and OPP have been completed (and the respective run ends).**

encountered vehicles, and set the round duration to 200 seconds. Vehicle spatial dynamics are dictated by a proprietary real-world GPS dataset of the city of Gothenburg, Sweden. As a supervised learning problem, we choose the widely used training of a Convolutional Neural Network (CNN) for image recognition over the CIFAR-10 dataset [17] as a representative of an automotive image recognition problem. This dataset contains 60000 32x32 pixel color images in 10 classes, of which 50000 are used for training and 10000 for testing the learning algorithm. The CNN has two convolutional layers with max pooling followed by three fully connected layers; during training, vehicles perform two epochs of stochastic gradient descent with momentum. For the distribution of data over the vehicles, we choose a highly skewed distribution of classes in which every vehicle holds 80 samples to emulate the real-world scenario of highly personalized data. V2X range is set to 200 m as an average for urban driving.

The evaluation took place on server hardware with an Intel Xeon E5-2620 v3 2.40GHz processor running the Core Simulator and an Nvidia GeForce GTX 1080 Ti graphics card as a vehicular on-board unit stand-in[5].

*Evaluation.* In Figure 4, we visualize the results from one experiment run: BASE (solid blue curve) finishes 75 rounds of training after 3592 seconds, while OPP requires 16342 seconds. The large speedup of the baseline is here explained by the much shorter round duration: as vehicles are not instructed to communicate with other vehicles in BASE, the round time can be set to a value that only covers the time period for transmitting and locally re-training the model. The bar plot indicates how many model exchanges via V2X occur in OPP, ranging from zero to 20. Thus, unlike in vanilla FL, the number of contributions to the model is not static, but varies over rounds, depending on the spatial dynamics of the vehicle (the dynamics allow or disallow encounters close enough and long enough to exchange models via V2X). On average, just below 10 additional model contributions are thus collected per round (indicated by the dash-dotted horizontal line in Figure 4). Finally, this results in a 50% increase in final accuracy of OPP over BASE, while incurring a 4.5 times longer total real-world run time, while employing the same V2C communication budget and thus equal costs (assuming on-board training and V2X usage costs are negligible).

The ability exemplified here, of quantifying trade-offs between metrics such as data volumes, accuracy and duration, is crucial for an analyst to make informed decisions about a learning strategy and is the core contribution of any framework abiding by the requirements from § 3.

[5]While more modern GPUs, especially for the vehicular domain, outperform the GPU used in this experiment, it can be assumed that the available headroom for ML training in a vehicular setting is limited as on older GPUs.

## 6 CONCLUSIONS AND OUTLOOK

In this paper, we have motivated the need for a tool to evaluate various learning strategies in a Vehicular Cyber-Physical System, to help fleet operators and OEMs learn from data generated on the vehicles themselves in a fashion that is optimal for their particular fleet and use case. Furthermore, we have collated a list of specific requirements for such a tool from an (industrial) user's perspective. As we show in the Related Work, existing works partially deliver the required functionality, but no single tool fulfills the complete requirements list. Therefore, we propose an architecture for a complete tool that can evaluate various learning strategies, and we present our prototype implementation of the tool along with preliminary results to demonstrate the tool's validity. As exemplified in § 5.2, the framework enables further understanding of the tradeoffs between dimensions such as time, cost, and accuracy when developing efficient learning strategies in a specific VCPS. In future work, we want to open-source the prototype implementation presented in § 5 to open it for contributions from the community, with the goal of increasing its resilience and enabling thorough testing of the framework. Possible next steps are implementing additional functionality in the prototype framework, where for example the simulation of various forms of communication could be enriched by integrating existing third-party network simulators, and increasing the parallelism of the simulation to speed up learning strategy development iterations.

## ACKNOWLEDGMENTS

This work is supported by Volvo Car Corporation (Volvo Cars) and the Swedish Government Agency for Innovation Systems VINNOVA, project "Automotive Stream Processing and Distributed Analytics (AutoSPADA)" (DNR 2019-05884) in the funding program FFI: Strategic Vehicle Research and Innovation.

## REFERENCES

[1] Furqan Alam, Rashid Mehmood, and Iyad Katib. 2017. D2TFRS: An object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In *International Conference on Smart Cities, Infrastructure, Technologies and Applications*. Springer, 155–168.

[2] Holger Berndt, Jorg Emmert, and Klaus Dietmayer. 2008. Continuous driver intention recognition with hidden markov models. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 1189–1194.

[3] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).

[4] Gustavo Carneiro. 2010. NS-3: Network simulator 3. In *UTM Lab Meeting April*, Vol. 20. 4–5.

[5] Francois Chollet. 2018. Deep learning with Python and Keras: The practical guide from the developer of the Keras library. *MITP-Verlags GmbH & Co. KG, Bonn* (2018).

[6] Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Chang-Heng Wang, Jérôme Härri, and Onur Altintas. 2020. On the orchestration of federated learning through vehicular knowledge networking. In *2020 IEEE Vehicular Networking Conference (VNC)*. IEEE, 1–8.

[7] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. 2021. Gossip Learning of Personalized Models for Vehicle Trajectory Prediction. In *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 1–7.

[8] Sonal Doomra, Naman Kohli, and Shounak Athavale. 2020. Turn Signal Prediction: A Federated Learning Case Study. *arXiv preprint arXiv:2012.12401* (2020).

[9] Romaric Duvignau, Bastian Havers, Vincenzo Gulisano, and Marina Papatriantafilou. 2021. Time-and Computation-Efficient Data Localization at Vehicular Networks' Edge. *IEEE Access* 9 (2021), 137714–137732.

[10] Ahmet M Elbir, Burak Soner, and Sinem Coleri. 2020. Federated Learning in Vehicular Networks. *arXiv preprint arXiv:2006.01412* (2020).

[11] Martin Fellendorf and Peter Vortisch. 2010. Microscopic traffic flow simulator VISSIM. In *Fundamentals of traffic simulation*. Springer, 63–93.

[12] Google. 2022. The Size and Quality of a Data Set. https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality.

[13] Tobias Grosse-Puppendahl. 2018. Leveraging AI Technologies for Porsche's Future. https://www.linkedin.com/pulse/leveraging-ai-technologies-porsches-future-tobias-grosse-puppendahl.

[14] Bastian Havers, Romaric Duvignau, Hannaneh Najdataei, Vincenzo Gulisano, Marina Papatriantafilou, and Ashok Chaitanya Koppisetty. 2020. DRIVEN: A framework for efficient Data Retrieval and clustering in Vehicular Networks. *Future Generation Computer Systems* 107 (2020), 1–17.

[15] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip learning as a decentralized alternative to federated learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 74–90.

[16] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller. 2018. Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction. *IEEE Transactions on Intelligent Vehicles* 3, 1 (2018), 5–17. https://doi.org/10.1109/TIV.2017.2788208

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. http://www.cs.toronto.edu/kriz/cifar.html.

[18] Sangsu Lee, Xi Zheng, Jie Hua, Haris Vikalo, and Christine Julien. 2021. Opportunistic Federated Learning: An Exploration of Egocentric Collaboration for Pervasive Computing Applications. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–8.

[19] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO, In The 21st IEEE International Conference on Intelligent Transportation Systems. *IEEE Intelligent Transportation Systems Conference (ITSC)*. https://elib.dlr.de/124092/

[20] Sidi Lu, Yongtao Yao, and Weisong Shi. 2019. Collaborative learning on the edges: A case study on connected vehicles. In *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*.

[21] IHS Markit. 2021. These OEMs are launching 5G-enabled cars years before the tech goes mainstream. https://ihsmarkit.com/research-analysis/these-oems-are-launching-5genabled-cars-years-before-the-tech-.html

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

[23] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. 2016. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles* 1, 1 (2016), 33–55. https://doi.org/10.1109/TIV.2016.2578706

[24] Dimitris Palyvos-Giannas, Bastian Havers, Marina Papatriantafilou, and Vincenzo Gulisano. 2020. Ananke: a streaming framework for live forward provenance. *Proceedings of the VLDB Endowment* 14, 3 (2020), 391–403.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[26] Jason Posner, Lewis Tseng, Moayad Aloqaily, and Yaser Jararweh. 2021. Federated learning in vehicular networks: opportunities and solutions. *IEEE Network* 35, 2 (2021), 152–159.

[27] Qualcomm. 2019. Cellular-V2X Technology Overview. https://www.qualcomm.com/media/documents/files/c-v2x-technology-overview.pdf

[28] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. 2018. Understanding Pedestrian Behavior in Complex Traffic Scenes. *IEEE Transactions on Intelligent Vehicles* 3, 1 (2018), 61–70. https://doi.org/10.1109/TIV.2017.2788193

[29] Karsten Roscher, Sebastian Bittl, AA Gonzalez, M Myrtus, and Josef Jiru. 2014. ezCar2X: rapid-prototyping of communication technologies and cooperative ITS applications on real targets and inside simulation environments. In *11th Conference Wireless Communication and Information*. 51–62.

[30] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. 2020. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal* 7, 5 (2020), 4641–4654.

[31] Christoph Sommer, David Eckhoff, Alexander Brummer, Dominik S Buse, Florian Hagenauer, Stefan Joerer, and Michele Segata. 2019. Veins: The open source vehicular network simulation framework. In *Recent advances in network simulation*. Springer, 215–252.

[32] AI Sweden. 2022. Edge Learning Lab. https://www.ai.se/en/data-factory/edge-lab.

[33] TensorFlow. 2022. TensorFlow Federated. https://www.tensorflow.org/federated.

[34] Andras Varga. 2010. OMNeT++. In *Modeling and tools for network simulation*. Springer, 35–59.

[35] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1698–1707. https://github.com/iQua/flsim

[36] Julia Silva Weber, Miguel Neves, and Tiago Ferreto. 2021. VANET simulators: an updated review. *Journal of the Brazilian Computer Society* 27, 1 (2021), 1–31.

[37] Business Wire. 2021. Strategy Analytics: Mobile Data Revenue Falls Below US$1 per Gigabyte as 5G Uplift Proves Elusive. https://www.businesswire.com/news/home/20210413005861/en/Strategy-Anal.