Towards Federated Learning in Edge Computing for Real-Time Traffic Estimation in Smart Cities

Matteus V. S. Silva[1], **Luiz F. Bittencourt**[1], **Adín Ramírez Rivera**[1]

[1]Instituto de Computação—Universidade Estadual de Campinas (Unicamp)
Campinas—SP—Brazil

Abstract. *The wide proliferation of sensors and devices of Internet of Things (IoT), together with Artificial Intelligence (AI), has created the so-called* Smart Environments. *From a network perspective, these solutions suffer from high latency and increased data transmission. This paper proposes a Federated Learning (FL) architecture for Real-Time Traffic Estimation, supported by Roadside Units (RSU's) for model aggregation. The solution envisages that learning will be done on clients with their local data, and fully distributed on the Edge, with high learning rates, low latency, and less bandwidth usage. To achieve that, this paper discusses tools and requirements for FL implementation towards a model for real-time traffic estimation, as well as how such solution could be evaluated using VANET and network simulators. As a first practical step, we show a preliminary evaluation of a learning model using a data set of cars that demonstrate a distributed learning strategy. In the future, we will use a similar distributed strategy within our proposed architecture.*

1. Introduction

The expansion of Internet of Things (IoT) sensors and devices, united to Artificial Intelligence (AI), has created the so-called *Smart Environments* in industries, cities, factories, agriculture among others [Valerio et al. 2017]. More specifically, Smart Cities congregate and need many different Smart Applications, composing a truly Smart ecosystem [Valerio et al. 2018]. Moreover, the proliferation of mobile computing and IoT results in billions of mobile devices communicating each other and connected to Internet, generating huge amount of data bytes at the Edge of network. Cisco estimates that almost $850\,\mathrm{ZB}$ (Zettabyte) will be generated at the Edge until 2021 [Index 2018].

Smart vehicles are often referred as important components of Smart Cities, providing smooth (and, in the future, autonomous) mobility to citizens. There are many proposals to use Cloud Computing [Zhang et al. 2010] to support Smart Vehicles, and some of them have already been implemented, like Cloud-based software update and training of powerful Deep Learning (DL) models [Kehoe et al. 2015].

It is anticipated that there will be more than 200 sensors in vehicles in the future, with the total sensor bandwidth ranging from $3\,\mathrm{Gbit\,s^{-1}}$ (approximately $1.4\,\mathrm{TB\,h^{-1}}$) for $40\,\mathrm{Gbit\,s^{-1}}$ (around $19\,\mathrm{TB\,h^{-1}}$) [Heinrich 2017]. As estimated by Intel, each autonomous vehicle will generate about $4000\,\mathrm{GB}$ of data per day, equivalent to the mobile data generated by almost 3000 people [Sensors 2017]. This will result in unprecedented pressure on communication infrastructures. From the network point of view, the use of

Workshop of Urban Computation (CoUrb) 2020

Cloud computing in this scenario brings issues like high latency and high bandwidth consumption due to massive data transmission [Barik et al. 2020].

To solve this problem, Edge Computing, an emerging paradigm which takes computing tasks and services from the core to the network edge, closer to end users and data sources, has been widely recognized as a promising solution for the aforementioned issues [Zhou et al. 2018].

What makes Edge Computing so promising is its applicability of local data analysis with strong Machine Learning (ML) usage, distributedly, as close as possible to the final devices. This conjunction of technologies is known as "Edge Intelligence" (EI) [Valerio et al. 2018]. According to Zhou et al. [2018], the main distributed ML technologies are Gossip Training, Knowledge Transfer Learning, DNN Splitting, Gradient Compression, Aggregation Frequency Control, and Federated Learning (FL). Among these, FL stands out for performing ML for data analysis directly on the final devices [Konečnỳ et al. 2016b] and can be employed in both architectures. Notwithstanding, properly modeling of FL to be run at the edge is needed, especially, in the contex of this paper, considering the amount of data generated by cars and how it can be efficiently utilized for specific application scenarios. We are interested in using data generated by cars or by other infrastructures (e.g., cameras) to precisely estimate traffic in a distribute way.

In this paper, we bring intelligence to the Edge of the network with an architecture for FL using decentralized data storage and processing. We focus on Real-Time Traffic Estimation, aiming at the understanding of the traffic flow and routes suggestion. We propose to reduce the use of resources from the Cloud by leveraging intelligence at the edge through FL, as proposed by Konečnỳ et al. [2016b]. It consists of providing a DL model trained in the cloud with historical data, so that vehicles can take advantage of this learning model with their local data set. As the model is updated at the edge, it must be uploaded to Roadside Units (RSU's) [Zhang and Letaief 2019] for aggregation (model update). Ultimately, this process let vehicles to learn directly from their data without the need to send raw data over the network during this process.

As a first step towards the implementation of this model, we implemented a supervised classification network on a federated learning setup. We obtained preliminary results using 8144 training car images, and other 8041 testing images, to understand how we could learn using such type of data, and afterwards apply it to estimate traffic. The first results of the implementation suggest that training the model on the clients is feasible to guarantee certain level of precision.

2. Background

Deep Learning (DL) uses Artificial Neural Networks (ANN) [Bithas et al. 2019] as primary source of modeling. ANNs consist of a series of layers that are capable of generating non-linear outputs based on the layer activations and input data [Zhou et al. 2019]. Neurons in the input layer receive data and propagate it to the middle layers [Zhou et al. 2018]. The neurons in the middle layers generate the weighted sums of the input data and emit the weighted sums using the specific activation functions and the outputs are propagated to the output layer. The use of more complex and abstract layers than a typical models allows the ANN to achieve high precision inference in the tasks [Zhou et al. 2018]. In our

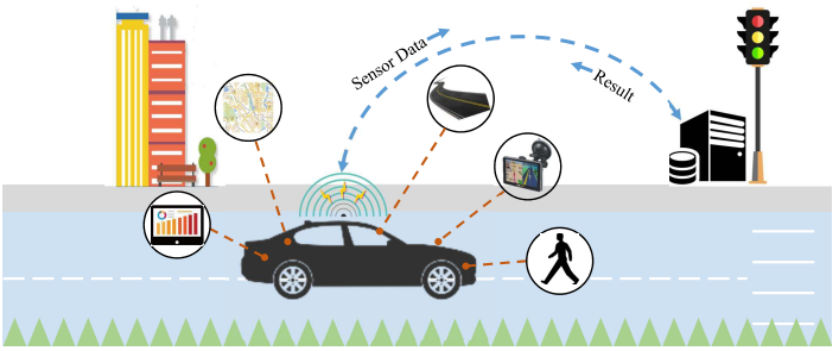


Figure 1. A smart vehicle perceives the environment through sensors and exchanges messages with the server. Usually, the vehicle sends the data to the server; then, the server updates the model; and sends the response to the client. Adapted from Zhang and Letaief's [2019].

proposal, we implement the traffic prediction (in our initial prototype an image classifier) through a NN that is updated either in the clients or in the cloud.

Federated Learning (FL) is a recently proposed ML paradigm that allows to collaboratively train a shared model for many users without direct access to raw data [Konečnỳ et al. 2015; McMahan et al. 2016]. Users train a ML model on the local data set and upload it to a server for a global model aggregation [McMahan and Ramage 2017]. However, moving data to a central server for training introduces a prohibitive communication overhead [Zhang and Letaief 2019]. According to McMahan and Ramage [2017], the device downloads the model trained in the Cloud with a large set of historical data, enhances it by learning from the data on the devices and summarizes the changes as a small, focused update. Only this update for the model is sent to the Cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All training data remains on the device and no individual updates are stored in the Cloud. FL has an increasingly important role to play in many applications, for example, healthcare, finance and transportation [Lim et al. 2019]. FL is used in our architecture to operate an intelligent, real-time traffic estimation at the edge of the network.

Real-Time Traffic Estimation. Connecting vehicles via Ad Hoc Vehicle Networks (VANETs) represents an initial attempt to support safety-related applications, such as traffic and new routes suggestion [Siegel et al. 2017]. Vehicles can communicate with the infrastructure on the road through Vehicle-to-Infrastructure (V2I) communications to also obtain information related to the road and traffic [Zhang and Letaief 2019] (see Fig. 1). This infrastructure, composed of communications nodes installed within called Road Side Unit (RSU), can provide Internet access to vehicles and can rebroadcast messages delivered by vehicles in low vehicle density scenarios [Barrachina et al. 2013].

Recently, with integrated processing for highly latency-sensitive tasks, it is possible to make decisions in real-time to control the vehicle with data pre-processing in order to reduce communication bandwidth [Siegel et al. 2017], in the shortest time possible, according to the application [Zhang and Letaief 2019].

Real-time traffic estimation can be performed in several ways, often translated into

an estimate of *traffic flow*. According to Kar et al. [2017], turning Vehicles into nodes in Edge Computing can enable estimation of current traffic. Traffic flow can be estimated using several information generated at the edge. For instance, images from surveillance cameras or road cameras can serve input data sets to learning algorithms, which allow us to capture information from sets of vehicles in a traffic jam as well as from approaching vehicles, resulting in a wealth of knowledge about real-time traffic conditions. Another examples of important information is geolocation of cameras, vehicles speed and acceleration, weather conditions, and so on. All such information can be combined into a learning framework to better estimate and understand traffic behavior, allowing better route suggestion and traffic delay estimations.

3. Related Work

Among the various ways of dealing with network setbacks in the context of applications with Smart Vehicles, ML presents many techniques and forms of deployment (centralized, distributed, etc.), which makes it highly applicable and versatile. We chose to work with FL as it operates distributedly. The following works bring ways to apply it in Smart Vehicles scenarios. In the following, we discuss how each of them contributes to address the aspects presented in Section 1.

In Rizwan et al.'s [2016] work, a vehicle collects data through ten IoT sensors in real time and sends them to the cloud via the Internet. Then, it performs Big Data analysis to understand local traffic and make predictions. These forecasts are presented to the user, from time to time, in the form of local traffic conditions, but do not show new routes. The scheme of this work does not guarantee low latency, since its focus is to work on the infrastructure in the Cloud and presentation to the user.

Bonawitz et al. [2019] use FL and follow the cloud aggregation approach in a similar application. Despite not going that far in applications involving vehicles, the authors mention FL with mobility. In this sense, they emphasize the success of placing fewer packets on the network, because the data remains on the device, transmitting the models only, thus a reduction in bandwidth use is considerable.

Vehicles and FL, in turn, are already part of studies by Ye et al. [2020]. Once again, the importance of the aggregation model is highlighted. The application, images from vehicle cameras, runs on an infrastructure with a central server and vehicle clients that share their local models. For that, the Central Server chooses M local models to add. In order to select the best local DL with satisfactory image quality and computing capacity, the election process is formulated with a two-dimensional contractual-theoretical theory of reward image compensation, resolved by a greedy solution. The Central Server in this work does not necessarily represent the Cloud, but any other type of Central server. This opens room to bring this entity closer to the user, placing it in a Fog or Edge. In addition, it is important to select not only the best, but also a good enough number of models to perform the aggregation, otherwise the application may be deficient.

A point previously addressed, but well explored by Lu et al. [2019], is synchronization. Their solution focuses more on privacy than on traffic. They created an architecture with Macrobase stations (MBS), several RSU's and mobile vehicles. MBS have powerful computing and storage resources. In addition, each RSU is connected to MBS by uplink and to vehicles (V2R) by downlink. RSUs serve as more connection points on

the route. The vehicle then sends data to the one closest to it at the moment. The MBS figure can be the Cloud or another Central server. However, it is a centralized point that uses RSUs as a form of Access Point (AC) only.

Samarakoon et al. [2018] propose an approach to minimize energy consumption in all vehicle network users (VUEs), with power transmission and resource allocation together to enable ultra-reliable low latency communication (URLLC) using Extreme Theory of Value (EVT). With FL, the distribution of these extreme events is estimated by the VUE in a decentralized manner. Here, we have a solution that seeks total decentralization. However, during aggregation, at least one vehicle must be elected as central and V2V sharing, which can be costly for the network as it is always necessary to make an election, something that the authors themselves concluded.

A neglected aspect in the literature is scalability. Most of the literature work with reduced areas. The work of Konečnỳ et al. [2016a] considers scalability, but focused on model training directly on devices. They employed algorithm optimization to decrease the network usage, something different from the proposal in this paper, but with a similar purpose.

As observed in the analysis of the selected studies, the distribution of ML, especially through FL, is still a very preliminary proposal. FL is a way of distributing learning across devices. A few works focus on privacy at the edge, something different from this paper, which aims much more at network aspects than privacy. When it comes specifically to vehicles, there are some approaches, but very little focused on traffic applications. It is also observed the aggregation of the model is still done through a central point. In our proposal, this aggregation point can be at the edge of the network.

With regard to synchronization, at first our proposal foresees synchronization, as discussed in Section 4. However, the aggregation distribution will enable this to done differently, using aggregation rounds accordingly to the current status of the network (e.g. number of vehicles, amount of information being obtained, and so on). In summary, the innovative points of this work in relation to the related works is the use of FL to improve response time and reduce network use, validated specifically in traffic applications, and to distribute the aggregation by the RSU's.

4. Proposal

The goal of our work is to bring intelligence to the Edge of the network through an architecture based on RSU's and FL with the intention of ensuring low latency and the best possible use of bandwidth by decreasing the amount of data traveling through the network. This paper brings an initial discussion on an infrastructure architecture to achieve such objective.

The proposed architecture includes vehicles, road side units (RSUs), and the cloud. Initially, we are considering a real-time traffic estimation application, but the architecture can be used to other applications in the VANETs context. Therefore, we used existing ML solutions which can be applied for Real-Traffic Estimation [Adetiloye and Awasthi 2017, 2019], then the main focus is bringing it as FL architecture at the Edge of network. The proposed model considers that the communication with Edge devices occurs in unreliable networks with limited upload speeds, which makes it crucial to minimize this communication. This can be achieved by communicating solely the learning

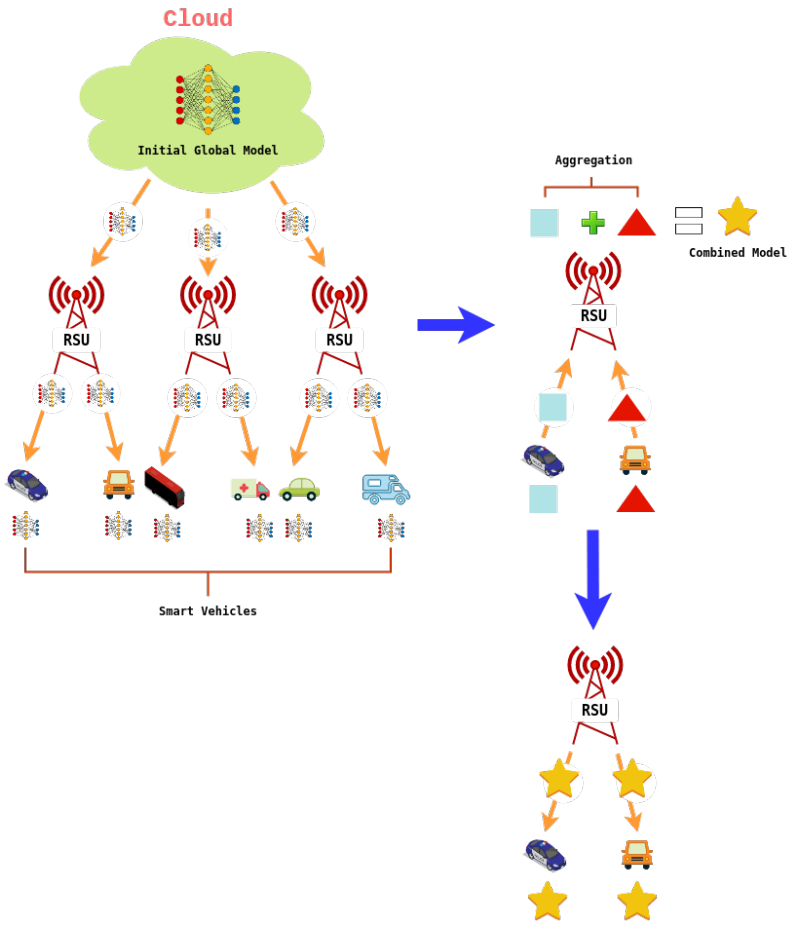


Figure 2. Architecture scheme. The initial global model is trained in the Cloud and distributed to RSUs, at the Edge. Each vehicle that enters and connects one of the RSUs will receive a copy of that model to execute FL. Then, the RSUs aggregate the model and redistribute it as needed.

model, while the raw data is kept at the edge, e.g. on vehicles or cameras generating traffic images. Note that camera images can come from traffic cameras, surveillance cameras, and dashboard cameras on vehicles, for instance.

4.1. Architecture

Our architecture (see Fig. 2) comprises a Cloud that performs the heavy computing for training and aggregating models, a set of RSU's as intermediate aggregators, and endpoint vehicles that collect and process individual data. Vehicles communicate with RSU's, which in turn can communicate with the Cloud. We consider the Cloud and the RSUs as *computing servers* that can perform heavy model aggregation tasks.

A global model is updated in the computing servers using data sets composed of historical data collected over time. This data set should contain data collected at the edge according to the needs of the application(s) being considered. In our specific application, we consider that cars, cameras, weather stations, and other relevant sensors generate data for traffic estimation. In the current stage, we only started to investigate image data sets.

Computing servers distribute the model after updating it towards the corresponding subset of nodes that need it in the network, following the hierarchy presented in Figure 2.

With the model in place, vehicles can run the ML method using local data. From time to time, an *aggregation function* is executed in the computing servers, in which vehicles share their local models to update the intermediate and the global models. Initially, we propose to distribute the aggregation by placing it in the RSU's or in the Cloud. However, determining where the aggregation is executed can be flexible and dynamic depending on the amount of data generated and the variability of the model through time.

4.2. Proposed Methodology

Different algorithms can be used in FL. Nilsson et al. [Nilsson et al. 2018] indicate three algorithms: Federated Averaging (FedAvg), Federated Stochastic Variance Reduced Gradient (FSVRG), and Cooperative Machine Learning (CO-OP).

We chose to use FedAvg in this work as it has the most simplified strategy and often the best performance [Nilsson et al. 2018]. According to [Nilsson et al. 2018], FedAvg trains a particular model w_t, where t denotes the iteration round, by sharing it with a fraction of clients and then pooling their individual models. For each round of learning, FedAvg randomly initializes the global model w_0. The server selects a subset of clients S_t, such that $|S_t| = C \cdot K \geq 1$ where C is the fraction of the K clients, and distributes the current global model to the clients in S_t. At the k-th client, it updates (through stochastic gradient descent) its version of the model w_t^k using its local data and producing the next iteration of models w_{t+1}^k. The local models are transmitted to the server which fuses them together to produce a new global model w_{t+1} that will be transmitted, and the process iterates. We formalize this process in Algorithm 1.

For instance, we can use FedAvg as follows. Once a model has been initialized in the Cloud, it is transmitted to RSU's and, at last, sent to vehicles. Then, within each vehicle, using local sensory data it will update the model using a Real-Time Traffic Estimation application. Over time, and with the unique experience from each vehicle, the model distributed in the vehicles will start to differ among each other. Then, the vehicles will share their local models to the computing servers for aggregation. Afterwards, the model will be redistributed, and the process will start again. Note that in this scenario, the data collected by the vehicles stay within them, and only the trained model is distributed.

5. Evaluating the Proposal

Once the architecture has been deployed, methods and mechanisms utilized to run models and distribute the knowledge will be evaluated. In this section, first, we show preliminary results regarding one of the tools needed in the proposed architecture. Then, we discuss how we plan to evaluate the whole proposal, using simulations, as the solution deployment evolves.

5.1. Federated Learning Initial Assessment

The implementation of this solution will take place in two stages: development and simulation. The development is about building the FL, which includes the model of learning, training, manipulation of the dataset, its distribution to clients and aggregation with FedAvg. This section shows the feasibility of distributedly learning a deep learning model

Algorithm 1 FedAvg, adapted from Nilsson et al.'s [2018].

Require: Amount of clients K in the system, a fraction K of clients to select the models from, a set $\{\eta_k\}$ of learning rates, and a ClientUpdate function defined according to the application.

initialize w_0
for each round $t = 0, 1, \ldots$ **do**
  $m \leftarrow \max(\lfloor C \cdot K \rfloor, 1)$
  S_t is a random set of m clients
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k = \text{ClientUpdate}(k, w_t)$ ▷ Real-Time Traffic Estimation
  end for
  $\eta_\sigma = \sum_{k \epsilon S_t} \eta_k$
  $w_{t+1} = \sum_{k \epsilon S_t} \frac{\eta_k}{\eta_\sigma} w_{t+1}^k$
end for

(in our case a deep learning classifier) in a federated learning setup, that is, the model is learned in the clients and distributed to them. The results show how clients (car) perform on executing learning after they receive the model a centralized server (e.g., a cloud or RSU).

5.1.1. Implementation

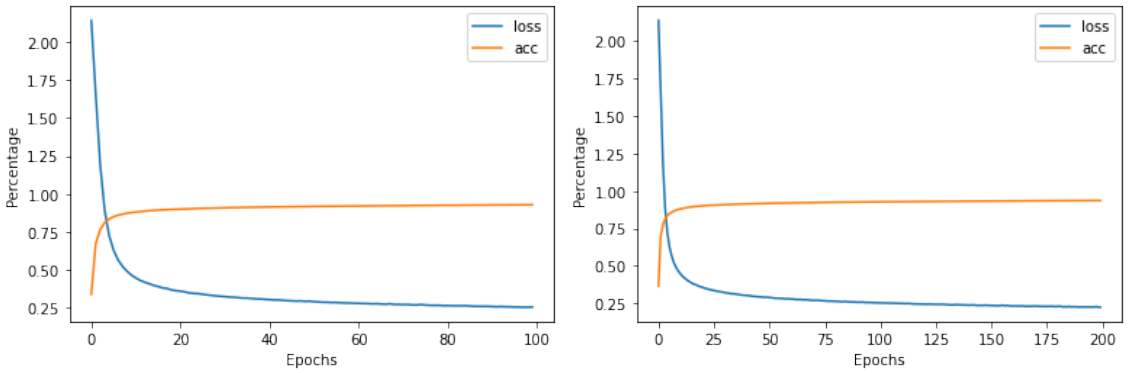
We used Python and TensorFlow in conjunction with the Keras library at this point. The crucial points to be evaluated are the construction of the model and the parameters that define the number of clients and rounds.

The application made here is a simple supervised learning approach, that is, it associates the image to its correct label. The training shows which label should go to which image. This will serve as the basis for counting vehicles in the traffic flow.

To evaluate the model, we compute two metrics from the experiments: accuracy and loss. The classification accuracy can be seen as the percentage of learning, i.e., the association the model makes between the image and the correct label, that clients obtain after training. And the loss is how far the predicted values deviate from current values in training data. As we will demonstrate, while the loss decreases, the accuracy increases. Similarly, the accuracy depends on how the model is *fit*.

The development took place on Google Colab platform. A single computer was used, but with GPU support enabled to help processing. At this stage, no other form of distribution has been implemented so far, such as using multiple CPU's or partitioning among cores, and the clients used here are simulated in the code and its quantity is fixed.

The TensorFlow framework works in conjunction with Keras, an open-source neural network library that offers an extensive range of algorithms. The data chosen for initial construction is the Cars196 [Krause et al. 2013] data set that comprises a set of car images and category labels. The sequence of implementation and testing took place in the following order:



(a) 100 epochs **(b) 200 epochs**

Figure 3. Loss and accuracy percentages when training the proposed model for (a) 100 and (b) 200 epochs.

1. Images, labels and clients are loaded. Currently, the number of clients is fixed, but it will be dynamic from the moment it is integrated with the simulation.
2. Model definition: a Multilayer Perceptron was implemented. This is a point that can lead to different scenarios, since the model can be changed to other neural networks, such as the Convolutional Neural Network (CNN), for example. To compose the training, part of the images were used for this purpose (8144) and the remaining images were used for testing.

5.2. Experimental Results

Model training was influenced by the number of training epochs, i.e., the number of times the model saw the full data set and adjusted on it. We run the model with The number of clients was fixed at 10 and epochs were run at 100 and 200. Taking into account that this will be done for each client, this value must be computed more accurately in a more complex implementation within a dynamic mobile environment. However, it is noticed, from a certain point, the accuracy stabilizes. The same is true with loss.

Results for accuracy and loss 100 and 200 epochs are presented in Fig. 3. The first point, then, is that the values of epochs should be computed more precisely considering the number of clients as parameters. In the preliminary tests, it was already possible to start at around 80% accuracy. The idea is that the value of epochs can be calculated to support training to deliver high precision to clients in different scenarios. This may also indicate that it could be a good idea to spend more time training and adjusting, as this time can be compensated through more precise route estimates for each user. In this first scenario, the execution of the model at each client took an average of 2.51 seconds over 10 executions (standard deviation of 0.06), which suggests the feasibility of running the federated learning at the edge with the utilized data set.

One of the next steps is to also test other types of neural networks and start working with larger numbers of clients, before integrating with simulation and moving on to dynamic clients.

5.3. Simulation Planning

To evaluate our proposal, we will use a simulated environment that provides us with traffic data. For the traffic flow, the works of Adetiloye and Awasthi [2017, 2019], among others,

suggest the use of data sets of *images* to estimate, by means of vehicle counting, how congested the road is. According to Adetiloye and Awasthi [2019], traffic congestion is defined as a particular amount of vehicles within a pre-defined distance within each other (commonly few meters in between). Traffic images, with roads and cars, can be used to estimate traffic. However, due to limitations on the simulations to produce imagery, we will focus on locally producing traffic estimation from the simulation software. We will also consider other types of information that can impact traffic, such as weather conditions, to be incorporate into the model.

Assessing the performance of the architecture will demand traffic simulations and network monitoring. To simulate RSUs and vehicles mobility, we plan to utilize OMNET++, SUMO, and VEINS. The OMNET++ Network simulator (Objective Modular Network Testbed in C++) is used to model and evaluate Network performance. It can be manipulated to create different scenarios for testing. SUMO (Simulation of Urban MObility) will be responsible for the generation of vehicle traffic, managing the flow of vehicles, lane characteristics, coordinates, speed, and acceleration of vehicles. Finally, VEINS (Vehicles in Network Simulation) will offer a graphical interface between OMNET++ and SUMO, allowing the construction of predefined graphics for vehicle networks, as well as collecting statistics from the simulated application.

This set of tools will operate with the 802.11p protocol stack, but Long Term Evolution (LTE) can also be considered, which are both implemented in the considered tools. Data transferred in the network, latency of traffic estimate, and accuracy are metrics initially planned to be collected.

6. Conclusion

We have shown the feasibility of implementing FL for real-time traffic estimation on a reduced and controlled scale. The accuracy of learning reached high values within few training rounds, but still interesting points were identified that should be evaluated in the next steps.

The developed code applies, in the end, Supervised Learning, where the data set and labels must be provided. In the next step of this research, we will integrate the learning phase with the simulation, where the number of clients (cars) will be dynamically set on the simulation environment. Other future directions are:

1. Investigate different models as well as how to tune the distributed model and its parameters.
2. Define equations and methods to dynamically compute parameters such as communication rounds for model aggregation and the ideal number of *epochs* in different scenarios. It is clear that the amount of epochs affects the accuracy of learning, but it also affects performance.
3. Identify where aggregation can be best run (e.g. cloud or RSUs) based on computing requirements, performance, and response time.
4. Evaluate the proposal considering different network communication technologies (e.g. WiFi and 5G).

7. Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível

Superior—Brasil (CAPES)—Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. And this work was supported in part by the Brazilian National Council for Scientific and Technological Development (CNPq) under grants No. 307425/2017-7, 309562/2019-8, and 432943/2018-8.

References

Adetiloye, T. and Awasthi, A. (2017). Predicting short-term congested traffic flow on urban motorway networks. In *Handbook of Neural Computation*, pages 145–165. Elsevier.

Adetiloye, T. and Awasthi, A. (2019). Multimodal big data fusion for traffic congestion prediction. In *Multimodal Analytics for Next-Generation Big Data Technologies and Applications*, pages 319–335. Springer.

Barik, R. K., Priyadarshini, R., Lenka, R. K., Dubey, H., and Mankodiya, K. (2020). Fog computing architecture for scalable processing of geospatial big data. *International Journal of Applied Geospatial Research (IJAGR)*, 11(1):1–20.

Barrachina, J., Garrido, P., Fogue, M., Martinez, F., Cano, J.-C., Calafate, C., and Manzoni, P. (2013). Road side unit deployment: A density-based approach. *IEEE Intelligent Transportation Systems Magazine*, 5:30–39.

Bithas, P. S., Michailidis, E. T., Nomikos, N., Vouyioukas, D., and Kanatas, A. G. (2019). A survey on machine-learning techniques for uav-based communications. *Sensors*, 19(23):5170.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. (2019). Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Heinrich, S. (2017). Flash memory in the emerging age of autonomy. *Proceedings of the Flash Memory Summit, Santa Clara, CA, USA*, pages 7–10.

Index, C. G. C. (2018). Forecast and methodology, 2016–2021 white paper. *Updated: February*, 1.

Kar, G., Jain, S., Gruteser, M., Bai, F., and Govindan, R. (2017). Real-time traffic estimation at vehicular edge nodes. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–13.

Kehoe, B., Patil, S., Abbeel, P., and Goldberg, K. (2015). A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409.

Konečnỳ, J., McMahan, B., and Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016a). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016b). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. (2019). Federated learning in mobile edge networks: A comprehensive survey. *arXiv preprint arXiv:1909.11875*.

Lu, Y., Huang, X., Dai, Y., Maharjan, S., and Zhang, Y. (2019). Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Transactions on Industrial Informatics*.

McMahan, B. and Ramage, D. (2017). Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., and Jirstrand, M. (2018). A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, pages 1–8.

Rizwan, P., Suresh, K., and Babu, M. R. (2016). Real-time smart traffic management system for smart cities by using internet of things and big data. In *2016 international conference on emerging technological trends (ICETT)*, pages 1–7. IEEE.

Samarakoon, S., Bennis, M., Saad, W., and Debbah, M. (2018). Federated learning for ultra-reliable low-latency v2v communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

Sensors, A. (2017). Electronics expo 2017. *Detroit, USA (14–15 June 2017)*.

Siegel, J. E., Erb, D. C., and Sarma, S. E. (2017). A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2391–2406.

Valerio, L., Conti, M., and Passarella, A. (2018). Energy efficient distributed analytics at the edge of the network for iot environments. *Pervasive and Mobile Computing*, 51:27–42.

Valerio, L., Passarella, A., and Conti, M. (2017). A communication efficient distributed learning framework for smart environments. *Pervasive and Mobile Computing*, 41:46–68.

Ye, D., Yu, R., Pan, M., and Han, Z. (2020). Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access*.

Zhang, J. and Letaief, K. B. (2019). Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

Zhou, W., Li, Y., Chen, S., and Ding, B. (2018). Real-time data processing architecture for multi-robots based on differential federated learning. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 462–471. IEEE.

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., and Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762.