Edge-assisted Gossiping Learning: Leveraging V2V Communications between Connected Vehicles

Giuseppe Di Giacomo[†‡], Jérôme Härri[†], Carla Fabiana Chiasserini[‡]

[†]EURECOM, Communication Systems, 06904 Sophia-Antipolis, France
E-mail: {Giuseppe.Di-Giacomo,Jerome.Haerri}@eurecom.fr
[‡]CARS@Polito, Politecnico di Torino, 10129 Torino, Italy.
E-mail: carla.chiasserini@polito.it

***Abstract*— Intelligent vehicles are quickly becoming mobile, powerful computers, able to collect, exchange, and process sensed data. They are therefore expected not just to consume ITS services, but also to actively contribute to the implementation of relevant ITS applications. With an increasing role of machine learning (ML) approaches, vehicles are called to put into use their computing capabilities and sensed data for the training of ML models. This can be enacted through distributed learning approaches, which however may lead to significant communication overhead or to learners converging to different models. In this work, we envision a new distributed learning scheme, named EAGLE, that, with the assistance of the network edge, aims at exploiting the vehicles' data and computing capabilities, while enabling an efficient learning process. To this end, EAGLE combines the advantages of two existing schemes, namely, federated learning and gossiping learning, yielding a distributed paradigm that ensures both scalability and model consistency. Our results, obtained using two different real-world data sets, show that EAGLE can improve learning accuracy by 20%, while reducing the communication overhead by 45%.**

***Index Terms*— Connected vehicles, Distributed machine learning, Deep learning**

I. Introduction

Over the last years, vehicles have been equipped with an increasing number of advanced driver assistance systems (ADAS), like radars, lidars and cameras, which can sense the vehicle surroundings. Indeed, one of the major challenges in the automotive domain is to make vehicles able to detect and autonomously react to external inputs such as obstacles and other vehicles' actions. Relevant examples include vehicles that automatically break in front of obstacles, or control their speed in order to keep a safe distance from the vehicle ahead.

Data collection through ADAS can be put to best use when combined with Machine Learning (ML), and in particular Deep Learning (DL), approaches [1]. Many tasks related to Intelligent Transportation Systems (ITS) and autonomous driving, requiring forecasting and decision making processes, like travel time estimation, eco-driving, and traffic optimization, can indeed benefit from it [2].

In spite of their good performance, however, DL-based approaches pose several hurdles, among others the significant amount of data to collect and process in order to correctly train DL models, or privacy aspects of transmitting such data to a central learning node.

Distributed DL implementation techniques have recently emerged to mitigate these shortcomings and enable distributing the computational burden by leveraging the data collected by a multitude of intelligent vehicles and letting such data to be processed directly by them. This avoids transferring data towards a centralized learning node, thus exploiting the devices' computational capability and protecting sensitive information.

In this context, one of the most popular approaches is Federated Learning (FL) [3], where a number of learning nodes train the same model and send their parameters to a central node, which combines such parameters and returns the resulting model to the learning nodes. The process is repeated till the required model accuracy is achieved. However, it has been observed [4] that the communication overhead entailed by FL may be significant. Alternatively, Gossip Learning (GL) permits to save bandwidth by letting vehicles exchange and merge models in a fully distributed way. However, current GL strategies [5] have been limited to random model passing strategy, ignoring local context typically observed in vehicular scenarios.

In this work, motivated by the fact that vehicles can generate suitable data sets by labelling the data themselves [1] and could leverage local vehicular context for GL if properly supervised, we introduce a novel technique, named Edge Assisted Gossiping LEarning (EAGLE), which regroups reduced communication bandwidth of GL with efficient context identification of FL. So doing, EAGLE achieves the same learning performance as FL, but with substantially fewer transmissions. Specifically, EAGLE efficiently clusters vehicles into groups of similar contexts, within which each member sequentially contributes to the training task. The models collaboratively trained within each group are then sent to a central coordinator, which aggregates them. We validate EAGLE features through the easily tunable CIFAR-10 dataset and evaluate EAGLE performance on a vehicular trajectory prediction ADAS using the NGSIM US-101 highway dataset.

The rest of the paper is organized as follows. Section II discusses related work and highlights the novelty of the proposed solution. The EAGLE scheme is introduced in Section IV, and evaluated against FL under a real-world scenario in Section V. Finally, Section VI concludes the paper and presents directions for future research.

II. Related work

Two main approaches to distributed learning are relevant to our work: FL and gossip learning (GL). As depicted in Fig. 1(left), the vanilla algorithm of FL [3] relies on a coordinator that at each round selects C learning nodes and sends them the latest version of the model. Then, upon receiving the parameters of the models newly trained by the C nodes, the coordinator combines them by computing a weighted average.

Beside such vanilla algorithm, several others have been proposed, in order to overcome the challenges posed by FL. Indeed, one of the main issues in FL is the communication overhead entailed by the exchange of model parameters between the coordinator and the learners. Especially when the number of involved client is very large, FL may be affected by scalability issues [6]. To reduce the amount of transferred data, [7] envisions two weights updates, with the former using a smaller number of parameters to learn model updates, and the latter applying compression on complete updates. With a similar goal, [8] introduces a hierarchical federated learning (HFL), which relies on a hierarchical network structure where model aggregation is also performed locally by intermediate nodes that are closer to the learning ones. Such aggregations are very frequent, unlike the global ones performed by the coordinator. In this context, [8] leverages edge nodes, placed between the coordinator and the learning nodes, for local aggregations: as they are closer to the users, bandwidth can be saved.

Another relevant challenge in FL is represented by the statistical data heterogeneity: non i.i.d. data and its unbalanced availability across the different learning nodes may lead to severe degradation of learning performance. A further body of work focus on the system heterogeneity, i.e., devices featuring different processing, communication and energy capabilities, which has again negative effects on the learning performance. Both issues have fostered studies on the optimal selection of the learning nodes, based on both the type and the amount of data they own, and their capabilities.

Finally, even if in FL only weights or gradient values are shared, there are still concerns about data privacy, which leads to integrating in the FL framework techniques such as differential privacy or encryption.

Other works have focused on totally distributed architectures, removing the central controller and letting the learning nodes update the model parameters at each round via peer-to-peer communications. GL, the second relevant method to our work, applies this approach [5]. Again, data is locally stored at the learning nodes and all nodes train the same model. However, as depicted in Fig. 1(center), in GL each learning node first initializes a local model; then, upon receiving the model parameters from another node, it updates its local ones, either simply overwriting them, or combining them. Notice that, thanks to the lack of a coordinator, GL exhibits high robustness and scalability; on the other hand, its performance depends strongly on the network topology [9]. In particular, in non-fully connected networks, as it is often the case for vehicular networks, groups of learning nodes that are not in radio visibility may converge to different models. However, it has been shown [10] that, when GL is used for the training of neural networks for vehicles' trajectory prediction, good performance can be obtained whenever vehicles collect an adequate amount of data, even if sample distributions are unbalanced and non i.i.d.

A real-world scenario in which FL is applied in the vehicular domain is presented in [11]. Therein, the authors propose a method exploiting FL for power and resource allocation, aiming to achieve ultra-reliable and low-latency communication. Still in the vehicular context, [12] presents a selective model aggregation technique for image classification. According to this method, local models to be sent to the coordinator are picked considering the quality of the training images and the vehicles' computation capability.

Novelty. As highlighted above, FL may lead to communication bottleneck and scalability issues. On the other hand, GL is not the ideal choice for a vehicular scenario, as it may lead to different local models. Thus, in this work we envision a new scheme, named Edge Assisted Gossiping LEarning (EAGLE), which, as highlighted in Fig. 1 and better detailed in the following, makes the most out of FL and GL. As demonstrated by the results shown in Section V, EAGLE copes with scalability issues better than FL by leveraging Vehicle-to-Vehicle (V2V) communications and allowing part of the training stage within the vehicular network, without involving the central coordinator. Also, it entails a low communication overhead, while making sure that all involved learning nodes ultimately get the same trained model. Moreover, even though EAGLE may seem similar to HFL, the latter simply exploits for the aggregation nodes that are closer to the users, but does not make use of local V2V communications and direct interaction between vehicles as EAGLE does.

III. Connected Vehicles Network Architecture and Use Case

Without loss of generality, in this paper we focus on a trajectory risk assessment application, which, considering the vehicles' current and recent positions, estimates their future trajectories and assesses potential risks. Specifically, an autonomous ego vehicle, upon reception of the position from its neighboring vehicles, forecasts their trajectories and detects whether it is on collision course with any of its neighbors. A fundamental component of such application is therefore the ML-driven trajectory forecast module, which is typically implemented through a Long-Short Term Memory (LSTM) model [13], [14].

We consider that the ML model is trained in a distributed fashion, and that, in the case of FL and EAGLE, a *coordinator* module, runs as an *edge* service at the network infrastructure. The coordinator manages the model training, as well as the exchange of the model parameters and their averaging.

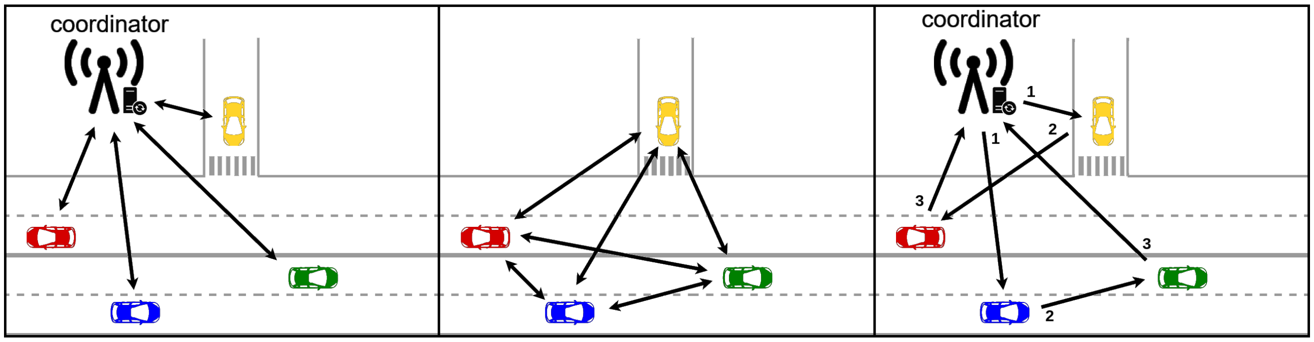


Fig. 1: Schematic representation of how FL (left), standard GL (center), and EAGLE (right) work. In EAGLE, the information flow during a training round involves three steps: the coordinator transmits the latest version of the ML model to G vehicles (step 1); each of them locally trains the model and transmits the updated parameters to a neighboring vehicle (step 2); the last vehicles of each training-subtask send the model parameters back to the coordinator (step 3).

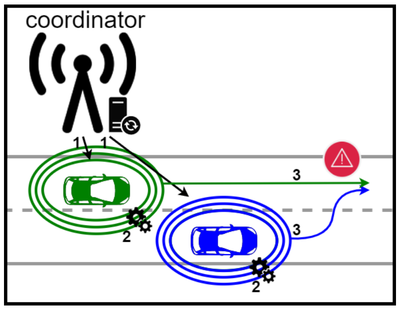


Fig. 2: Risk reasoning application: network architecture and information flow.

The information necessary for the model training and the trajectories forecast are obtained through V2V and Infrastructure-to-Vehicle (I2V) communications. Specifically, vehicles and road infrastructures are equipped with on-board units (OBU) and road side units (RSUs), respectively. Vehicles' current and recent positions are assumed to be exchanged through periodic transmission of ETSI Cooperative Awareness Messages (CAM) over Vehicle-to-Vehicle (V2V) communications[1]. The coordinator, instead, collects the ML model parameters and shares the updated ones with vehicles over Vehicle-to-Infrastructure (V2I) communications[2].

Fig. 2 illustrates such network architecture and the risk reasoning application operational flow. Vehicles first download a trained ML from the coordinator via V2I communication (step 1). Then, upon reception of CAMs from other vehicles through V2V communications, they integrate their neighbors' current and recent positions into the trained ML model to obtain future positions (step 2). Finally, the on-board risk reasoning logic extract potential upcoming hazard (step 3).

A relevant challenge is therefore to generate *well-trained* ML models, and, accordingly, in this paper we focus on the training of such models through FL and our proposed EAGLE scheme. We remark that, while the vehicles' positions are exchanged over V2V in all schemes, FL exchanges the model parameters over V2I only and GL over V2V only. On the contrary, as depicted in Fig. 1, our proposed EAGLE scheme exploits for the transmission of the model parameters both V2I and V2V, which mitigates the communication overhead entailed by FL, as well as the limited merging scope and efficiency entailed by GL.

[1]Without loss of generality, it is assumed that CAM are transmitted either on ITS-G5 or LTE/5G V2X technologies.

[2]Without loss of generality, it is assumed that ML models may be exchanged either through 5G or ITS technologies.

IV. Edge-Assisted Gossiping Learning

In this section, we present the *Edge Assisted Gossip LEarning* (EAGLE), designed to combine the strengths of both FL and GL, while mitigating their shortcomings in highly dynamic vehicular scenarios. Without loss of generality, we assume that the *coordinator* edge service acts as an ML coordinator, which has a DL model to train. Algorithm 1 describes the different steps of EAGLE.

At each training round, the central controller picks G vehicles in the target area (Line 3), based on a given criterion. Several options are available, ranging from the quality of the data that different types of vehicles may own, or the number of vehicle's neighbors. Then, the controller transmits the latest available model to the G vehicles, which locally train the model, each for a given number of epochs (Line 5). After completing this task, each vehicle forwards the trained model (i.e., the locally computed model parameters) to one of its neighbors, instead of sending it back to the central controller. In this way, G parallel GL-subtasks are started (Line 6). In each GL-subtask, a target total number, N, of vehicles sequentially train the model (Lines 12, 13 and 14), again, each for a given number of epochs, and within such subtask the next-learning vehicle is selected according to one of the aforementioned criteria.

When either N vehicles have participated and completed the GL-subtask, or a vehicle does not happen to have any neighbor to further forward the model to, the updated model parameters are sent back to the central controller (Line 17). The latter aggregates all the received models by taking the weighted average on the values obtained from all GL-subtask (Line 7).

Algorithm 1 The EAGLE scheme

```
Require: $\eta_i$: initial learning rate, $n_g$: number of samples used in subtask $g$, $n_r$: total number of samples owned by the learning nodes selected in round $r$, $d$: decay coefficient, $t_g$: target number of vehicles for each GL-subtask
 1: Initialization of weights $w_0$
 2: for each round $r = 1, 2 \dots$ do
 3:     selection of $G$ vehicles
 4:     for each vehicle $v$ in parallel do
 5:         $w_v \leftarrow \texttt{client\_update}(w_r, r, 0)$
 6:         $w_{r+1}^{(g)} \leftarrow \texttt{GL\_subtask}(w_v, r)$
 7:     $w_{r+1} \leftarrow \sum_{g=1}^{G} \frac{n_g}{n_r} w_{r+1}^{(g)}$
 8: procedure GL_subtask($w, r$)
 9:     $n \leftarrow 0$
10:     while $n < N - 1$ do
11:         if $n$ has neighbors then
12:             $k \leftarrow \texttt{select\_neighbor}(n)$
13:             transmit $w$ to vehicle $k$
14:             $w \leftarrow \texttt{client\_update}(w, r, n+1)$
15:             $n \leftarrow n + 1$
16:         else break
17:     return $w$ to coordinator
18: procedure client_update($w, r, n$)
19:     $i \leftarrow (r-1) \cdot t_g + n$
20:     $\eta \leftarrow \eta_i \cdot d^i$
21:     for each local epoch $\ell = 1, 2 \dots$ do
22:         for each batch $b = 1, 2 \dots$ do
23:             $w \leftarrow w - \eta \nabla l(w, b)$
24:     return $w$
25: procedure select_neighbor($n$)
26:     $k \leftarrow$ selected neighbors of $n$
27:     return $k$
```

Note that the algorithm also implements a dual strategy for the learning rate decay (Lines 19 and 20). Whereas the decay is applied at each round in FL, EAGLE applies a decay to the learning rate both whenever a new round is started and every time a vehicle receives the model from a neighbor within the GL-subtask.

The information flow in EAGLE is illustrated in Fig. 1(right). At the beginning of every round, the coordinator sends the latest available model to the chosen G vehicles (step 1). As vehicles periodically broadcast CAMs carrying the sender's current and recent positions, each of the G vehicles can leverage the collected information to locally train the ML model. Such a node will then transfer the model parameters to one of its neighboring vehicles (step 2). Step 2 is repeated until the sequential training within the vehicles' group is completed and the last vehicle of the GL-subtask sends the model parameters back to the coordinator (step 3).

At last, we remark that: (i) thanks to the aggregation phase performed by the coordinator, EAGLE allows for the convergence of all local instances of the same model, and (ii) by exploiting direct communication between vehicles, it substantially reduces the number of transmissions between learning nodes and central controller, hence increasing the scalability of the training task.

V. Experimental results

We now compare the performance of EAGLE against standard FL via simulation, using two different real-world datasets, namely, CIFAR-10 [15] and NGSIM US-101 [16]. CIFAR-10 has indeed been widely exploited for image classification in prior art, and, hence, it allows testing the proposed solution using a well-known dataset. NGSIM US-101, instead, is specific to the vehicular domain, as it provides a collection of vehicles trajectories, along with other relevant information. This dataset can be exploited to predict future vehicles' trajectories, thus suiting well the risk reasoning application we consider. Further details on the datasets and on how we used them, as well as on the simulation settings, are provided below.

A. Experimental settings

The **CIFAR-10** dataset contains 60,000 RGB figures of 32×32 size, which belong to 10 different classes: 50,000 images are used for training, while the remaining for testing.

Two kinds of experiments are performed with this dataset: first, data are divided in an i.i.d. manner among the clients; then, samples are assigned to them in a non-i.i.d. way. In both cases, we use as model a convolutional neural network, whose architecture is publicly available in [17] and has been developed using the PyTorch framework. In total, the trainable model parameters are 122,570, corresponding to 0.5 MB data to be transmitted when using the 32-bit floating point format. Furthermore, we use the cross entropy loss and the Mini-batch gradient descent [18] algorithm for the optimization.

Experiments with i.i.d data are performed dividing training samples in an i.i.d. manner across 100 learners, out of which only 10 are picked at each round. The learning rate is initially set to 0.15; it is then decreased at each round for FL and, according to the mechanism reported in Algorithm 1, for EAGLE, using in both cases a decay factor of 0.99. Finally, the data batch size is set to 50, and the number of local epochs is equal 5.

For experiments with non-i.i.d. data, instead, samples are assigned in such a way that each client has data belonging only to a single class. The total number of clients is still 100, but, contrary to the previous case, at each round 50 of them are randomly selected. This allows us to create 5 GL-subtasks of 10 clients each and to pick clients within each GL-subtask by using two different strategies, random e class-based, as explained in Section V-B. The initial learning rate is 0.005, while the decay factory is set to 0.999. The batch size is now equal to 64 and the number of local epoch is 1.

The **NGSIM US-101** dataset contains the vehicles' trajectories collected on the US-101 highway. Thus, in this case we use an LSTM model, which, when correctly trained, can extract temporal patterns of past information to forecast the future vehicles' locations. Since the data has been collected by cameras placed along the highway, it is necessary to simulate samples that are recorded by vehicles instead of an external system. To this end, we consider that each vehicle can receive CAMs only from neighbors within a range of 15 m.

The learning model architecture is similar to the one used in [13], and consists of one LSTM layer, followed by 3 fully connected layers, with the last one providing the final output. The hidden and output size of the LSTM is set to 256, while the size of the dense layers are respectively 256, 128, and 2, which is the number of the output features. The number of trainable parameters is 376,450, entailing an amount of data to be transmitted of 1.5 MB when using the 32-bit floating point format. The features used for model training are: velocity, acceleration, lane ID, space headway and time headway for the target neighboring vehicle. Input to the LSTM is fed in sequences of 100 samples, corresponding to a 10-second interval. As they overlap over a 9-second interval, two successive windows have 90 values in common. The network then uses the first 90 samples to predict the future 10 vehicle's positions.

Further, for the training we used the mean squared error as loss function, and Mini-batch gradient descent optimizer with starting learning rate of 10^{-4}. The batch size is 32, and we use the same settings as before for the decay factor. After the pre-processing step, 90% of the vehicles are used for training, and the remaining for testing. At every training round, 50 clients are selected, for both FL and EAGLE; in the latter, within the same GL-subtask, whose maximum size is set to 10, a vehicle transmits the model to its closest neighbor.

B. Numerical results

CIFAR-10 We start by comparing the performance of our EAGLE scheme against that of FL, when using the CIFAR-10 dataset. In this case, we consider a classification task.

i.i.d. Data. We adopt the same simulation settings [3] as in [3]. Further, at every round, 10 clients participate in the learning task, and each group formed in EAGLE to perform a GL-subtask includes either 2 or 5 vehicles.

Fig. 3(a) presents the value of loss function as the number of rounds varies, for both EAGLE and FL. We observe that, due to overfitting, both the curves referring to FL and to EAGLE with 2-client groups show an increasing loss after a descending phase. For EAGLE with 5-client groups, instead, the loss function always decreases with the number of rounds, since the learning rate is decreased whenever a client receives the new model weights, either from the server or from another vehicle. It follows that using 5-client groups leads to a more substantial decay, thus preventing the loss function from increasing.

Fig. 3(b) depicts the obtained accuracy level, which, as expected, grows with the number of rounds. In particular, the accuracy achieved by EAGLE with 5-client groups increases faster, while FL exhibits the slowest convergence. Also, while FL and EAGLE with 2-client groups have final comparable performance, EAGLE with 5-client groups yields higher accuracy. The reason for such an improvement is that, at each round, the coordinator receives fewer models, but each of them is the result of the training performed sequentially by more clients. This entails that more data has been used for the training of each received model, and, since the number of epochs computed at each node is always the same, the overall number of computation epochs, performed before sending the model weights back to the coordinator, is larger in the case of EAGLE with 5-client groups.

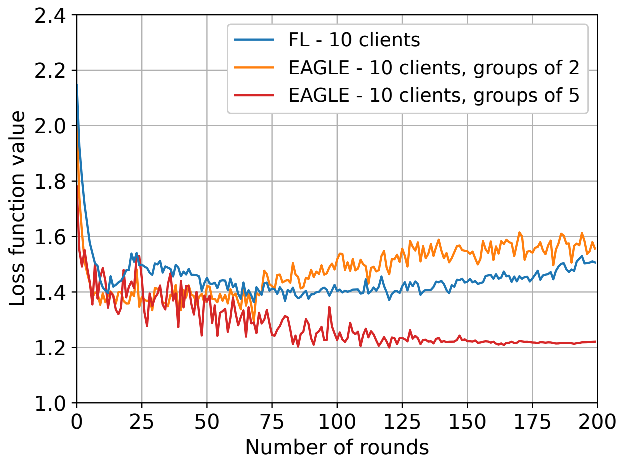
This is confirmed by the results in Fig. 3(c), which shows the achieved accuracy as a function of the total number of performed epochs. All investigated cases have a similar behavior, with EAGLE with 5-client groups exhibiting a lower accuracy level in the early stage, but a larger value in the end.

Interestingly, the behavior in terms of communication overhead instead differs quite significantly across the considered schemes and configurations. As shown in Fig. 3(d), EAGLE achieves the same, or even higher, accuracy level than FL, in a substantially lower number of transmissions. Again, a higher number clients within each EAGLE GL-subtask leads to better performance. We underline that the obtained overhead reduction is due to the fact that at every round in FL every node contributing to the learning task sends its model weight to the coordinator, and the coordinator sends an update to each node. Denoting the number of nodes by C, this results in $2C$ transmissions per round. On the contrary, under EAGLE with $N-$client groups, each of the C nodes transmits the model either to another vehicle or to the server, which sends the updated version of the model only to one vehicle per group, thus resulting in $C + C/N$ transmissions.

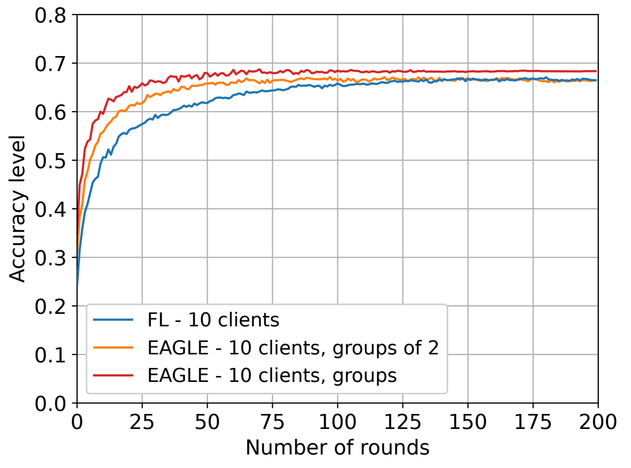
Non-i.i.d. data. Results of CIFAR-10 with non-i.i.d. data are shown in Fig. 4. In particular, Fig. 4(a) depicts the loss function value versus the number of rounds for both EAGLE and FL, while Fig. 4(b), (c) and (d) present the accuracy level versus the number of rounds, epochs and required transmissions, respectively. As far as EAGLE is concerned, two cases are considered. First, GL-subtasks learners are randomly picked; even with this straightforward strategy, EAGLE does not only show a faster performance improvement in terms of rounds and transmissions as in the previous case, but it also yields a much higher accuracy than FL. Second, a class-based criterion is employed for the GL-subtasks so that every client has data of a unique class and, since the total number of classes is 10, each of the 10 clients in every GL-subtask is representative of a class. This implies that, at every round, the server receives for each GL-subtask a model that has been trained using data of all classes. For this reason, when groups are formed according to the class-based criterion, EAGLE exhibits better performance than when the random selection strategy is applied.

NGSIM US-101. We now turn to the NGSIM US-101

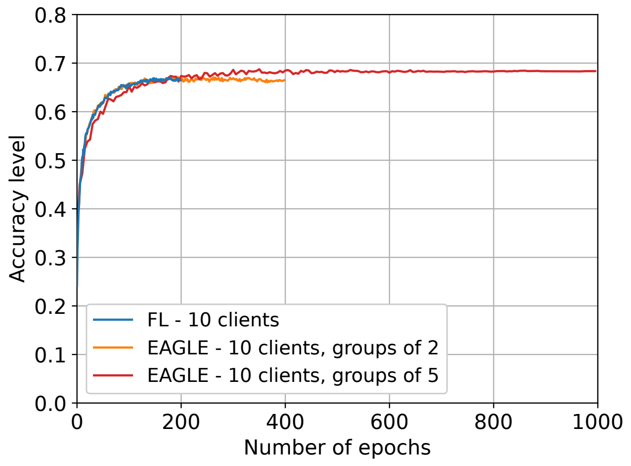
[3]The FL accuracy achieved here is lower, as we do not pre-process the data.



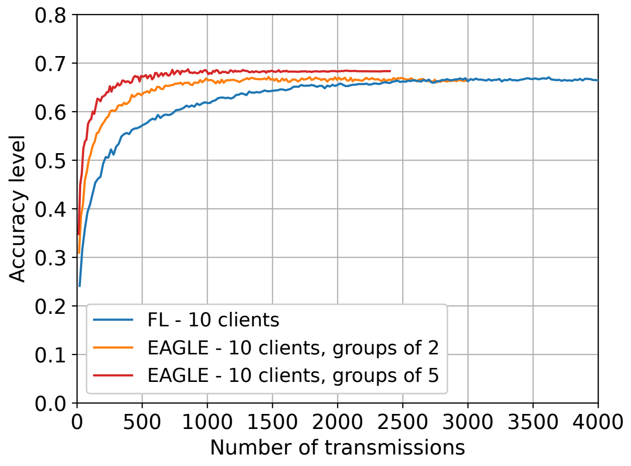
(a)



(b)



(c)



(d)

Fig. 3: CIFAR-10: comparison between FL and EAGLE with i.i.d. data.

dataset and the LSTM task for trajectory prediction, and present the obtained performance in Table I. In particular, the table includes the test lateral (x) and longitudinal (y) Mean Absolute Error (MAE) computed with respect to ground-truth after 100 rounds, along with the required total number of transmissions and number of transmissions performed by the coordinator. Looking at these results, we notice that the position errors are high for both EAGLE and FL, as achieving good performance in trajectory position is out of the scope of this work and, hence, the model we used has not been optimized. What we are interested in, instead, is the EAGLE's performance relatively to FL. On this regard, EAGLE provides noticeable better results than FL, both in terms of prediction error and number of required transmissions. In particular, as highlighted in the last row of the table, EAGLE improves the prediction accuracy by about 11-28%, and, even more remarkably, yields an overhead reduction of 45% in terms of total number of transmissions.

TABLE I: NGSIM US-101 results after 100 rounds

Scheme	x MAE [m]	y MAE [m]	of 5 No. of coord. transm.	Total no. of transm.
FL	0.87	4.25	$5 \cdot 10^3$	10^4
EAGLE	0.62	3.78	$5 \cdot 10^2$	$5.5 \cdot 10^3$
Gain	**28.7%**	**11%**	**90%**	**45%**

VI. Conclusions and Future Work

The main goal of this paper is to present a new approach, namely EAGLE, which combines FL and GL methods. We illustrated the main points by means of simple experiments, in order to highlight the potentiality of this novel method.

We addressed the problem of performing learning tasks at the network edge, towards the efficient provision of mobile services to vehicular users. With the aim to fully exploit the vehicles' data and computing/storage capabilities, we investigated distributed learning approaches such as federated learning and gossiping learning. We then proposed a

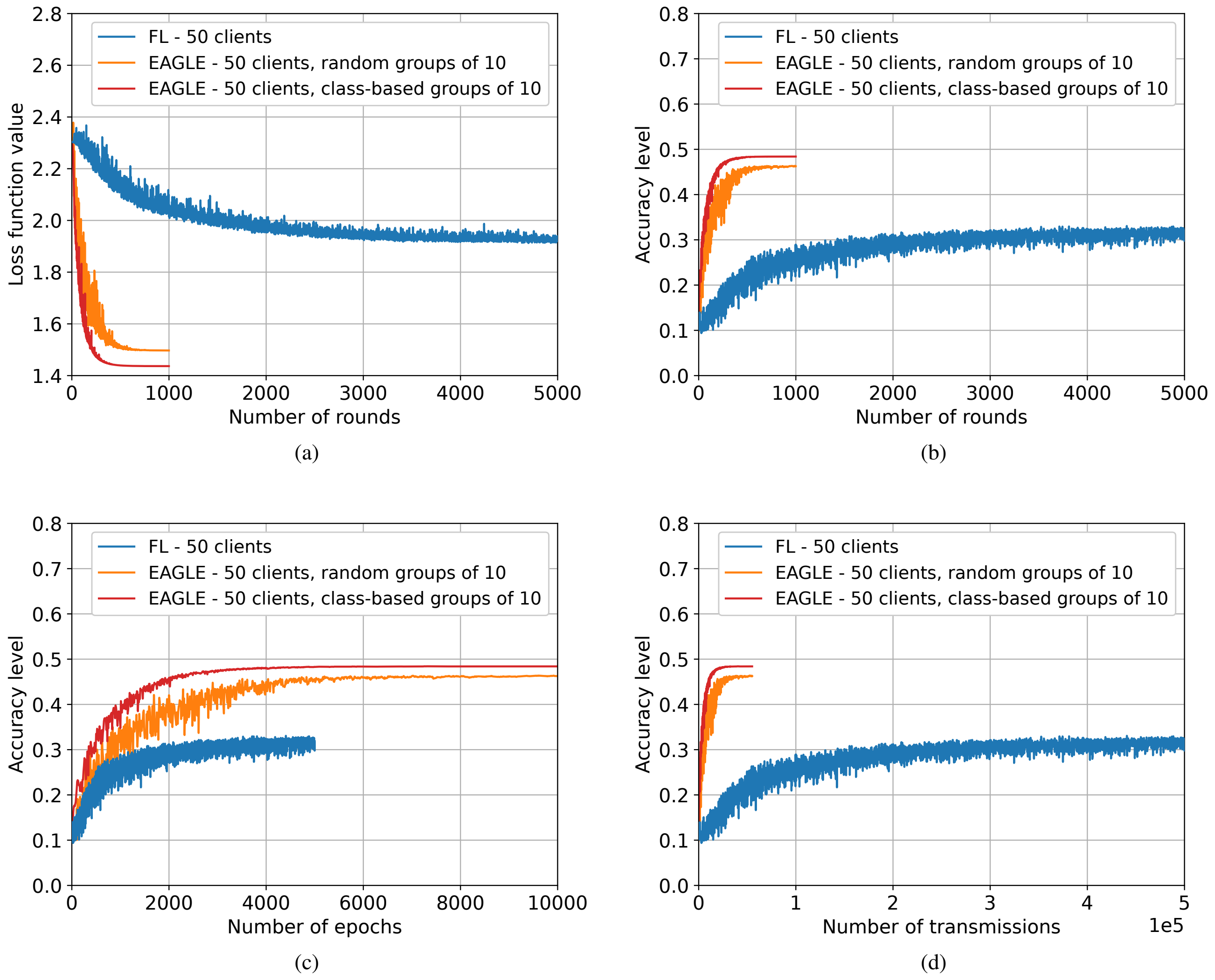


Fig. 4: CIFAR-10: comparison between FL and EAGLE with non-i.i.d. data.

new scheme, called EAGLE, which makes the most out of the two aforementioned approaches. Our results, obtained using two different data sets, show indeed that EAGLE can significantly outperform standard FL, providing both higher learning accuracy and a substantially lower data transmission overhead.

Future work will extend the performance evaluation of EAGLE by varying the learning parameters and using further data sets and models. Further, an interesting research direction is represented by the design of group formation strategies towards a robust execution of the gossiping learning sub-tasks, in spite of the highly dynamic environment.

Concerning the vehicular context, future work will consider many other factors such as the channel impact on communications, the mobility and the time required for the training and for data transmission. Also, due to a complex scenario such as the vehicular one, it will be important to make EAGLE robust to missing reception at the edge side of models, or to the reception of stale ones. Finally, it would also be relevant to analyze computational complexity and to study feasibility considering also nodes with low hardware capability.

Acknowledgment

This work was supported by the EU Commission under the RAINBOW project (Grant Agreement no. 871403), and by NPRP-S 13th Cycle Grant No. NPRP13S-0205-200265 from the Qatar National Research Fund (a member of Qatar Foundation). The views expressed are those of the authors and do not necessarily represent the project. The Commission is not liable for any use that may be made of any of the information contained therein.

References

[1] A. M. Elbir, B. Soner, and S. Coleri, “Federated learning in vehicular networks,” 2020.

[2] D. C. Selvaraj, S. S. Hegde, N. Amati, C. F. Chiasserini, and F. P. Deflorio, “A reinforcement learning approach for efficient, safe and comfortable driving,” in *24th EURO Working Group on Transportation Meeting (EWGT)*, 2021.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[4] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[5] I. Hegedüs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, vol. 11534, pp. 74–90.

[6] M. Zhang, E. Wei, and R. Berry, "Faithful edge federated learning: Scalability and privacy," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3790–3804, 2021.

[7] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017.

[8] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," 2019.

[9] L. Giaretta and S. Girdzijauskas, "Gossip learning: Off the beaten path," in *IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1117–1124.

[10] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "Poster: Mobile gossip learning for trajectory prediction," in *IEEE Vehicular Networking Conference (VNC)*, 2020, pp. 1–2.

[11] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2020.

[12] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[13] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359.

[14] L. Xin, P. Wang, C.-Y. Chan, J. Chen, S. E. Li, and B. Cheng, "Intention-aware long horizon trajectory prediction of surrounding vehicles using dual LSTM networks," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 1441–1446.

[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, May 2012.

[16] U.S. Federal Highway Administration. (2007) U.S. highway 101 dataset. [Online]. Available: https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm

[17] "Convolutional neural network (CNN)," https://www.tensorflow.org/tutorials/images/cnn.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.