A Gossip Learning Approach to Urban Trajectory Nowcasting for Anticipatory RAN Management

Mina Aghaei Dinani,
Adrian Holzer
University of Neuchatel,
Switzerland
name.surname@unine.ch

Hung Nguyen
The University of Adelaide,
Australia
hung.nguyen@adelaide.edu.au

Marco Ajmone Marsan
Institute IMDEA Networks,
Spain
ajmone@polito.it

Gianluca Rizzo
HES-SO Valais, Switzerland, and
University of Foggia, Italy
gianluca.rizzo@hevs.ch

Abstract—**In future radio access networks, machine learning (ML) based strategies for short-term forecasting of vehicular trajectories will be key for anticipatory resource allocation and management at the mobile edge. However, training ML models in a centralized fashion, over data collected from a massive heterogeneous and dynamic set of devices, poses significant scalability, reliability, and efficiency challenges, which are still open to date.**

In this paper, we look at the specific issue of scalable and resource-efficient training of ML models in a vehicular environment. To address such a challenge, we propose a new Gossip Learning scheme, i.e., a fully distributed, collaborative training approach based on direct, opportunistic model exchanges via wireless device-to-device (D2D) communications with no centralized support. Our approach is based on constantly improving each node's own model instance through knowledge transfer among nodes, and on different strategies for estimating the potential contribution of neighboring nodes to the training process at a node. Extensive numerical assessments on a variety of measurement-based dynamic urban scenarios suggest that our schemes are able to converge rapidly and provide sufficiently accurate forecasts of vehicle position for time horizons which are typical of future 5G/6G dynamic resource allocation algorithms.

I. INTRODUCTION

Machine learning based techniques for vehicle trajectory nowcasting, i.e., short-term trajectory forecasting [1], play a central role in future 5G/6G radio access networks (RAN) as a key enabler of road safety algorithms for autonomous vehicles [2] or of anticipatory RAN resource management strategies [3], to mention a few.

Machine learning (ML) based techniques for vehicle trajectory nowcasting, i.e., short-term trajectory forecasting [1], play a central role in future 5G/6G vehicular networks as a key enabler of anticipatory radio access network (RAN) resource management strategies [3], as well as road safety algorithms for autonomous vehicles [2]. Several challenges, however, stand in the way of designing a scalable and efficient strategy for training an ML prediction model for vehicular trajectory nowcasting. The most prominent one is how to take advantage of the availability of large amounts of privacy-sensitive data distributed among users without exposing it to breaches. Indeed, training a prediction model in a centralized fashion would not only put a high privacy risk on the central server,

but it would also be difficult to scale to large numbers of vehicles, and it would be inefficient from the viewpoint of RAN bandwidth utilization.

To overcome these shortcomings of centralized approaches, several distributed ML schemes have recently emerged [4]–[6]. Among these, Federated Learning (FL) [6] offers potentially better privacy protection than traditional approaches by substituting data sharing with model sharing. However, it relies on a central coordinating node that dispatches and aggregates models trained locally on each node. As such, it still bears all the main shortcomings of centralized architectures in terms of bandwidth utilization, scalability, and single point of failure. To address these drawbacks, several fully decentralized approaches have been proposed [7]–[10]. Among these, a promising technique is Gossip Learning (GL) [7]. It is a fully decentralized version of FL, where individual nodes train, exchange, and merge models with neighbors without the need for a central coordinator. However, most of these existing solutions for decentralized model training mainly assume no node mobility and no churn, making them unsuitable for dynamical settings such as vehicular scenarios.

In this paper, we tackle the above-mentioned open issues for the specific problem of training an ML model for trajectory nowcasting in an urban setting. We propose a novel GL approach that allows new nodes in the area to build over, and benefit from, models elaborated by other nodes. Our approach does not require support from the infrastructure, nor a connected interconnection graph, and it allows continuously improving the model and keeping it up to date with changes in the context. Moreover, it does not require maintaining a global state (such as in [11]), thus enabling better scalability. Our collaborative approach enables high-performing nodes to support those nodes whose model performs less well (e.g. because they do not encounter enough neighbors, or because they do not possess the data required to train a high-performing model by themselves).

Specifically, the main contributions of this paper are as follows:

- We propose a novel and fully distributed GL approach for highly dynamic settings, based on node cooperation and asynchronous communications, and on the combination of local model training with merging of models received via

opportunistic exchanges.

- We present three practical model aggregation strategies, based on iterative weighted model averaging and on different estimators of the impact of each contribution on the accuracy of the model resulting from the aggregation.
- We characterize analytically its convergence properties, showing that our GL scheme converges to an optimal set of parameters under mild assumptions on the connectivity and regularization of the data.
- We assess the effectiveness of our GL strategies over measurement-based mobility traces against a set of relevant baselines, as functions of different urban settings and traffic configurations. Results suggest that our approach is effective and able to converge and adapt even in the presence of large variations in vehicle density and traffic patterns. They also suggest that the performance of our schemes is comparable (if not better, in some cases) to Federated Learning schemes.

The rest of this paper is organized as follows. Section II presents the system model, and in Section III our GL algorithms are described in detail. Section IV presents the main results related to the convergence of our algorithms, and Section V is dedicated to their numerical assessment. Section VI discusses the state of the art. Finally, conclusions and future directions of investigation are summarized in Section VII.

II. SYSTEM MODEL

We consider nodes moving on a road grid according to an arbitrary mobility model. We assume that each node knows its position at any time, and that nodes communicate in a peer-to-peer fashion using wireless technology (e.g., WiFi, BLE, or cellular D2D). When two nodes are in range of each other (and thus able to exchange information directly), we say that they are *in contact*.

Without loss of generality, we assume time to be divided into equal-sized slots, and let $t \in \mathbb{N}$ be the slots' label. We focus on a specific region of the road grid within which, at any slot t, each node tries to predict its own location h slots in the future. h is thus the *forecast horizon*, assumed to be equal for all nodes. Let $i \in \mathbb{N}$ denote the unique identifier of a node. We assume that each node entering the region possesses a *local dataset* $\mathcal{S}_i$, generally different in size and composition for each node, and composed by time series pertaining to its past trajectories in the region. Specifically, if with (x_t, y_t) we denote the node position at the beginning of the t-th slot, the s-th element of the local dataset is a time series $\{(x_t, y_t)\}_{t=t_s,\ldots,t_s+l_s}$ of size l_s by 2, representing a trajectory, i.e. a sequence of positions over the plane over l_s consecutive time slots, starting at slot t_s.

III. A DISTRIBUTED GOSSIP LEARNING ARCHITECTURE

A. Model architecture

We assume that each node, in order to predict its own trajectory, employs a Long Short Term Memory (LSTM) neural network, a special kind of Recurrent Neural Network (RNN), widely used for vehicle trajectory prediction [12]. More generally, in the case of time series forecasting, when a sequence of input and output multi-variant data is available,

Symbol	Description
i	Node label
t	Time slot label
j	Round label
w_j^i	Array of coefficients of the local model instance of node i at the beginning of round j
ξ_j^i	Performance estimator of the model from node i at the beginning of round j
K_j^i	Set of nodes whose models have been received by node i during phase 1 of round j
β	Cutoff value
l_i	loss of the model received from node i
A	Total number of cells in the region
a	Cell label
$\mathcal{S}_i$	Local data set of node i
α	Number of input steps of the LSTM model
h	Forecast horizon (in n. of slots)

TABLE I: Main notation used in the paper.

encoder-decoder LSTM models have shown to outperform other approaches for trajectory prediction, such as Kalman filtering [13] or support vector machines [14]–[16].

Specifically, the architecture of the LSTM model at each node is given by the concatenation of two LSTM layers (*encoder* and *decoder* respectively). At each time t, the encoder layer takes as input a trajectory over the last α time slots. The output of the encoder layer is a fixed-length vector that captures the trajectories' temporal structure. The decoder layer maps such a vector representation back to a variable-length target sequence of h coordinate labels that describe the forecasted node position from time slot $t+1$ up to $t+h$.

The training process of such LSTM model takes place over a sequence of iterations (denoted as *epochs*). At each epoch, the parameters of the model are updated. We adopt a mini-batch gradient descent approach, in which the training data during one epoch is partitioned into two or more batches. This allows avoiding storing all training data in memory [17]. We employ the Adam optimizer [18] because of its computational efficiency and simplicity, as it requires little memory. It is well suited for problems with many parameters, and appropriate to cope with non-stationary objectives [18]. Nonetheless, our approach extends immediately to other optimizers and, more generally, to other ML architectures.

B. Gossip Learning algorithm

In what follows, we describe the training algorithm run by each node within the considered region of the road grid. Its goal is to generate more accurate models than those resulting from training exclusively on the local dataset of each node, without relying on any parameter server or centralized coordination function, and exploiting only opportunistic communications during ephemeral contacts between nodes.

We assume that, in addition to a local dataset, each node entering the region is endowed with an instance of the encoder-decoder LSTM model. Such an instance is initialized randomly, and trained over the node's local dataset. In addition, it may incorporate a priori information on patterns of vehicular trajectories in the region. The composition of the local dataset at ingress time, and the coefficients of the LSTM model can be tuned to model settings with various degrees of availability of a priori knowledge about patterns of mobility in the region.

At every time slot, each node in the region feeds its local instance of the LSTM model with the last α samples of the time series representing the node's own trajectory over the last α slots, and it uses its local instance to issue a prediction about its position in the next h slots. In addition, it keeps expanding its local dataset by adding new data about its trajectory in the region as it unfolds over time.

Starting from ingress time, the training algorithm run by each node develops into a sequence of *rounds*. We assume that all rounds have the same duration, and are equal for all nodes, but they are not synchronized across nodes.

Each round is composed of three *phases*. During the first one, each node sends the coefficients of its local model instance to all nodes with which it comes in contact, and it receives the coefficients of their local model instance. As local instances change over time due to the training process, the version of the local instance received from each node is relative to the time at which the exchange occurs.

In the second phase, similarly to traditional FL, each node combines the model instances received during the first phase to produce an updated version of the local model instance (denoted as *meta-model*). Finally, in the third phase, each node trains the meta-model on its own local dataset to include data about the most recent part of the node trajectory, thus producing a new version of its local model instance. This third phase may occur every $m \geq 1$ rounds, where m depends on the rate at which the local dataset is enriched with new data points. Thus, the duration of each phase within a round and the frequency of the training of the meta-model over the local dataset can be tuned and adapted to the specific setup. The total number of epochs for each local training step is chosen in such a way as to balance between minimizing the risk of overfitting, and maximizing model accuracy.

As a consequence of mobility, nodes enter and exit the given region. Thus, the population of nodes participating in a GL scheme is subject to *churn*. Indeed, as we assume that the trained model is of use only for nodes in the region, all nodes exiting the region discard the trained model and thus stop contributing to the GL scheme.

As evident in such a training strategy, each node never shares its data with others. Instead, nodes share their model instances, thus providing support for the protection of data confidentiality. No support from servers or from infrastructure-based communication technologies is required, and stable connectivity among nodes is not needed.

A key aspect of GL is the *model merging* strategy, which determines the coefficients of the meta-model as a combination of the corresponding coefficients of the models to be merged. The strategy for choosing the weights associated with every model to be merged is a key factor influencing the performance of the meta-models and thus of the whole training scheme. When GL is applied to dynamic settings, each node's context, and thus its neighbours (and the models received from them) constantly vary over time. In these conditions, many factors may impact the utility of the received models, in terms of their potential for generating a meta-model which improves over the existing local instance. Therefore, there are many possible ways in which these factors can be accounted for in choosing which model instances to combine, and thus in defining a specific merging algorithm.

In this work, we propose three different approaches to meta-model computation: *Decentralized Averaging* (DA), *Decentralized Powerloss* (DP), and *Location-Based Averaging* (LBA). Each corresponds to a different way to account, in the merging process, for the specificities of the environment in which the distributed model training occurs. They are all based on associating a *performance estimator* (which we denote with ξ) with each model instance which has to be merged, i.e., an estimate of the potential of the model to improve the performance of the merged model with respect to the existing local model instance. The performance estimator is used to compute the weight attributed to each model in the derivation of the merged model. These merging strategies differ in how the estimator is derived and updated.

Moreover, as such an estimate is absolute and not relative to the specific set of models to be merged, a cutoff value $0 \leq \beta \leq 1$ is introduced, so that only a fraction β of the largest contributions (in terms of weight used in the computation of the merged model) is used in the derivation of the merged model. By tuning the cutoff value, it is thus possible to prevent poorly performing model instances from negatively affecting the accuracy of the merged model. An example is given by those settings in which nodes with high-performing models are relatively rare and sparsely distributed over the region, so that the vast majority of models to be merged are likely to perform poorly on the prediction task of the node. In settings where models not used in the merging process are not exchanged, tuning the cutoff value can also be beneficial to implement different tradeoff points between communication efficiency, rate of convergence of the training process, and model accuracy.

Decentralized Averaging (DA). The key idea behind this merging strategy is to attribute a weight to model instances based on the number of samples they incorporate, either directly (i.e. by training on the local dataset), or indirectly (i.e. by merging). Thus, in this strategy, weights are computed based only on the properties of the models to be merged, not on the data or the context of the node which performs the merge operation. That is, nodes in different parts of the scenario, with different local data sets, but with the same set of model instances to be merged produce the same merged model. Such a property, in an abstract scenario with full connectivity among all nodes, would bring the GL training process towards solving a single consensus problem, thus producing a single trained model with the same coefficients for all nodes. However, in realistic scenarios, spatial heterogeneity in the data being collected by nodes and fed to the training process, together with lack of full connectivity among nodes, typically result into a spatial heterogeneity in the models resulting from the GL training process, and thus into a form of local specialization of models. In the numerical section, we will assess the impact of these aspects on the performance of models trained via the

DA strategy.

The details of the DA strategy are outlined in Algorithm 1. For every node i present in the region during round j, with w_j^i we denote the array of coefficients of the local model instance of node i at the beginning of round j, with estimator ξ_j^i. $w_j^{i,out}$ is instead the array of coefficients of the local model instance of node i at the end of the second phase of round j, with estimator $\xi_j^{i,out}$. Finally, K_j^i is the set of those nodes whose models have been received by node i during the first phase of round j, and $K_j^i(\beta) \subseteq K_j^i$ is the subset composed by those nodes in K_j^i whose models' estimator at round j, normalized, is above a cutoff value β. That is,

$$K_j^i(\beta) = \left\{ k | k \in K_j^i, \frac{\xi_j^k}{\sum\limits_{h \in K_j^i} \xi_j^h} \geq \beta \right\} \quad (1)$$

At the beginning of the first round, when the local model instance is trained on the local dataset, the performance estimator of the local model is set as the number of data points in the local dataset. After that, whenever a set of model instances is merged, the meta-model is computed as a weighted sum over all merged model instances. In this sum, the contribution of the $k-th$ merged instance is weighted by its associated performance estimator, normalized over the sum of all such parameters from all merged instances. As shown in Algorithm 1, if a cutoff β is applied, only those instances whose normalized weight is larger than β are included in the merging process. The performance estimator of the meta-model resulting from the merging operation is computed as a (normalized) weighted sum of the estimators of all the merged model instances, with the same weights as those used for the derivation of the meta-model itself. Whenever a model instance is trained again over the local dataset (i.e., during the third phase of a round), the performance estimator of the local model instance *after* the local training is the sum of its value *before* the local training, and the number of data points which have been added to the local dataset during round j.

Hence, in DA the performance estimator of a model instance is a loose estimate of the number of data points included in the model by training or merging, over the course of the training process. Such a choice is based on the intuition that models including larger amounts of data points are potentially more accurate. In reality model accuracy depends also in a complex manner on several other factors, such as the distribution of the data included in a model (which in our GL training algorithm is a function of the patterns of contacts and model exchanges among nodes, and thus of the features of mobility patterns themselves). In Section V, we explore such interdependency by assessing the algorithm on various scenarios and mobility patterns.

Location-Based Averaging (LBA). The intuition behind the LBA strategy is that, in models used for nowcasting, affinity among models is related to affinity in prediction tasks, possibly because they are relative to the same area. Thus, the goal of GL with LBA is to train a set of models, each capturing the features of mobility of a specific, predefined area within the region. Unlike DA (in which it is rather an emerging property),

---

Algorithm 1 Decentralized Averaging at node i in round j

function MERGEMODELSDA($\{(w_j^k, \xi_j^k)\}_{k \in K_j^i}$, β)

    compute $K_j^i(\beta)$ (eq. 1)

$$w_j^{i,out} \leftarrow \sum_{k \in K_j^i(\beta)} \frac{\xi_j^k}{\sum\limits_{h \in K_j^i(\beta)} \xi_j^h} w_j^k$$

$$\xi_j^{i,out} \leftarrow \sum_{k \in K_j^i(\beta)} \frac{(\xi_j^k)^2}{\sum\limits_{h \in K_j^i(\beta)} \xi_j^h}$$

end function

---

in LBA model specialization is thus explicitly accounted for in the training process. Specifically, in LBA the given region is partitioned into A cells. For each cell, each node (independently of whether it is located in that cell or not) trains a model which is meant to be used for trajectory predictions only when a node is in that cell. Thus, every node trains A models at the same time, maintaining for each a specific performance estimator. Similarly to DA, this estimator is related to how many data points of a specific cell have been included (by training and merging) in the model associated with that cell. In scenarios with enough heterogeneity in mobility patterns, this approach to distributed training of personalized (or, more specifically, location-specific) models should potentially enable higher accuracy levels, as it avoids combining models with little affinity among them.

The LBA merging strategy is outlined in Algorithm 2. With $a = 1, ..., A$, we denote the label of the a-th cell in the region. $\mathbf{w}_j^i = (w_{j,1}^i, ..., w_{j,a}^i, ..., w_{j,A}^i)$ denotes the array of the coefficients of the A local model instances of node i at the beginning of round j, one for each cell. Similarly, with $\boldsymbol{\xi}_j^i$ we denote the array of estimators associated with them. $K_j^i(\beta, a) \subseteq K_j^i$ is the subset composed by those nodes whose contribution to the merged model for cell a is above the cutoff β:

$$K_j^i(\beta, a) = \left\{ k | k \in K_j^i, \frac{P_{j,a}^k S_j^k}{\sum\limits_{h \in K_j^i} P_{j,a}^h S_j^h} \geq \beta \right\} \quad (2)$$

where

$$P_{j,a}^k = \frac{\xi_{j,a}^k}{\sum\limits_{a' \in 1, ..., A} \xi_{j,a'}^k}$$

and

$$S_j^k = \frac{\sum\limits_{a' \in 1, ..., A} \xi_{j,a'}^k}{\sum\limits_{h \in K_j^i} \sum\limits_{a' \in 1, ..., A} \xi_{j,a'}^h}$$

For each cell, the performance estimator $\xi_{j,a}^{i,out}$ of node i's meta-model for cell a in round j is computed as the weighted sum of the estimators of the models to be merged. For any model to be merged from node k and cell a, the weight is the normalized product of two components. Similarly to DA, the first component $P_{j,a}^k$ is directly proportional to the estimator of node k's model, and thus to the relative number of data points

Algorithm 2 Location-Based Averaging at node i in round j

---

function MERGEMODELSLBA($\{(\mathbf{w}_j^k, \boldsymbol{\xi}_j^k)\}_{k \in K_j^i}, \beta$)

    for every node $k \in K_j^i$ **do**

$$S_j^k \leftarrow \frac{\sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^k}{\sum\limits_{h \in K_j^i} \sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^h}$$

    end for

    for every cell a **do**

        for every node $k \in K_j^i$ **do**

$$P_{j,a}^k \leftarrow \frac{\xi_{j,a}^k}{\sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^k}$$

        end for

    compute $K_j^i(\beta, a)$ (eq. 2)

$$w_{j,a}^{i,out} \leftarrow \frac{\sum\limits_{k \in K_j^i(\beta,a)} P_{j,a}^k S_j^k w_{j,a}^k}{\sum\limits_{h \in K_j^i(\beta,a)} P_{j,a}^h S_j^h}$$

$$\xi_{j,a}^{i,out} \leftarrow \frac{\sum\limits_{k \in K_j^i(\beta,a)} P_{j,a}^k S_j^k \xi_{j,a}^k}{\sum\limits_{h \in K_j^i(\beta,a)} P_{j,a}^h S_j^h}$$

    end for
end function

Algorithm 3 Decentralized Powerloss at node i in round j

---

function MERGEMODELSDP($\{w_j^k\}_{k \in K_j^i}, \beta$)

    compute $K_j^i(\beta)$ (eq. 3)

    for every node $k \in K_j^i(\beta)$ **do**

        Compute $l_{j,k}^i$

    end for

$$w_j^{i,out} \leftarrow \frac{\sum\limits_{k \in K_j^i(\beta)} |\log_{10} l_{j,k}^i| w_j^k}{\sum\limits_{h \in K_j^i(\beta)} |\log_{10} l_{j,h}^i|}$$

end function

---

incorporated in that model for cell a. The second component S_j^k is proportional to the total amount of data points included in all models of node k. The product $P_{j,a}^k S_j^k$ is thus an indicator of how well a model is estimated to perform in cell a with respect to the other cells. Indeed, the fact that, for a given node, a model for a given cell is not as good as that for the others might signal that the node has not been able to include enough information about trajectories in that cell. Note that at the beginning of the first round, for every cell a, the performance estimator equals the number of data points in the local dataset relative to that cell.

Decentralized Powerloss (DP). In this strategy, the weights associated with each model to be merged are derived from a measure of the model's performance over the most recent trajectory of the merging node. The intuition behind this is that such a measure is likely to be correlated to the model's performance in the near future along the merging node's trajectory. As a consequence, in DP nodes exchange models but not their performance estimators, as they are computed by the node which receives those models, based on its own recent trajectory. Therefore, performance evaluators have a very narrow validity (i.e., limited to the merging node and to a specific time) and they are recomputed whenever they are needed, i.e. just before the merging operation.

The DP merging strategy is detailed in Algorithm 3. In round j at node i, the performance measure of a model to be merged received from node $k \in K_j^i$ consists of its loss, denoted as $l_{j,k}^i$. Such loss is computed over the merging node's validation set, composed by the last V samples of its trajectory. The weight associated with each model to be merged is then computed as a (normalized) logarithmic function of the loss, as this choice increases the weight of those models with small losses. Thus,

the expression of $K_j^i(\beta) \subseteq K_j^i$ for the DP strategy, when a cutoff value is applied to merging weights, is:

$$K_j^i(\beta) = \left\{ k \mid k \in K_j^i, \frac{|\log_{10} l_{j,k}^i|}{\sum\limits_{h \in K_j^i} |\log_{10} l_{j,h}^i|} \geq \beta \right\} \quad (3)$$

In general, the choice of the size of the validation set is a compromise between getting an estimation which, on the one side, has low noise, and on the other side, is representative of the actual performance of the model on the merging node trajectory in the near future. As a loss function, in the present work, we have adopted the mean squared error (where the error is the distance between the actual position of the merging node and its predicted position). However, the DP strategy is very general, and it applies to any other type of loss function.

Being based on performance metrics which are specific to each merging node, our GL scheme with the DP merging strategy implements model personalization. Indeed, it produces a model which is generally different at each node, being tightly related to the context (the road segment in which the car is located at a given point in time) and the specific prediction task of each node.

IV. CONVERGENCE PROPERTIES

In this section, we show that under some mild assumptions on the system, loss function and network topology, our GL training scheme converges. We first formally define the optimization objectives and key assumptions, which the convergence analysis will follow. For the derivations, we refer to the DA merging strategy, though the results can be extended to the other merging strategies with some marginal modifications. The key idea underlying the derivation of the proof is based on defining an objective function for the training process, which is a function of the local loss function as well as of the difference, at each node, between the local model and the models received from neighbors. Indeed, a decrease over time of these two components indicates, on one side, that the training process is progressing, and on the other, that nodes are successful in sharing the learned knowledge with their neighbors. then we show that, when the aforementioned mild assumptions are satisfied, the objective function decreases as the iterations progress.

A. Notation and assumptions

As already stated, the goal of collaborative model training schemes such as GL is to let nodes jointly improve their

models by leveraging both their local datasets and similar data available in the neighbourhood, where a time-varying connectivity graph defines neighbourhoods. Let us consider a time interval $[1, J]$ of duration equal to J rounds, and let us assume for simplicity the starting time of each round to be the same for all nodes. Let $\mathcal{N}$ denote the set of nodes (of cardinality N) which are present in the region in at least one of the slots of the given time interval. Each node $i \in \mathcal{N}$ has a local data distribution μ_i over the space of possible trajectories within the region. It has a local data set $\mathcal{S}_i$ composed of trajectories drawn randomly from μ_i. Each trajectory has a probability p_d (same for all trajectories) of being included in the local dataset. $\forall i$, we assume μ_i not to change over time, and $\beta = 1$.

In general, at every round, as the training process evolves, the coefficients of the models associated with every node in the region change. Let $(w_j^i)_{j=1}^J$ be the sequence of iterations generated by the GL algorithms running for J rounds from an initial point $w_0^i \in \mathbb{R}^p$. The goal of node i is to learn a model instance whose coefficients w_J^i minimize the expected loss $\mathbb{E}_{z \sim \mu_i}[l(w_J^i; z)]$ after J iterations, where $l(w_J^i; z)$ denotes the mean loss of w_J^i evaluated over every point of the trajectory z, for any choice of the loss function.

Given that the local model instance of a node is trained over its local dataset by using LSTM and Adam optimizer, the goal of the local training phase is to select those model parameters which minimize loss through Stochastic Gradient Descent [19]. Specifically, if we denote with $w_j^{i,loc}$ such a model instance, we have

$$w_j^{i,loc} = \arg\min_{w \in \mathbb{R}^p} \mathcal{L}_i(w; \mathcal{S}_i)$$

$\mathcal{L}_i(w; \mathcal{S}_i)$ is the *local loss function* for the i-th node:

$$\mathcal{L}_i(w; \mathcal{S}_i) = \frac{1}{s_i} \sum_{z \in \mathcal{S}_i} l(w; z) + \lambda_i \|w\|^2 \qquad (4)$$

where $\lambda_i \geq 0$ is a regularization parameter, and s_i is the cardinality of $\mathcal{S}_i$. In our gossip learning scheme, nodes use information from neighbours to supplement their data through several iterative rounds. We formalize the time-varying model by denoting $G^j = (\mathcal{N}, E, \mathbf{X}^j)$ as a weighted connected graph over the set of nodes, where $E = \mathcal{N} \times \mathcal{N}$ the set of all potential edges between the nodes and $\mathbf{X}^j \in \mathbb{R}^{n \times n}$ is the non-negative weight matrix at round j, where the specific merging strategy is determined the weights. In this model, only the weights $\mathbf{X}^j$ change with time, where $X_{ik}^j > 0$ if and only if $k \in K_j^i$. At round j the two nodes i, k are connected, and their local models and data are used in each other updates. There are two factors influencing how the values of X_{ik}^j change from one round to the other:

- Dynamic changes in the topology. They are due to node mobility, and they are out of the control of the learning algorithms.
- Changes that are caused by the way each node merges the models received from its neighbours. Weights X_{ik}^j are specific to the merging strategy, and they may depend on w_j^i.

Note that the three averaging algorithms differ in computing X_{ik}^j.

Let $\mathbf{w}_j = (w_j^1; \ldots w_j^i; \ldots w_j^N) \in \mathbb{R}^{N \times p}$. The overall goal of our gossip learning schemes is to find an array $\mathbf{w}_J$ of models, one for each of those nodes who spend in the region at least one round in the interval $[1, J]$, which minimizes a given system objective function $Q^J(\mathbf{w}_J)$.

The expression of the overall objective function $Q^j(\mathbf{w}_j)$ at round $j \in [1, J]$ can be written as

$$
\begin{aligned}
Q^j(\mathbf{w}_j) &= \sum_{i \in \mathcal{N}} Q^{i,j}(\mathbf{w}_j) \\
&= \sum_{i \in \mathcal{N}} \left[Q_{p2}^{i,j}(\mathbf{w}_j) + \nu Q_{p3}^{i,j} \left(\sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) \right],
\end{aligned}
\tag{5}
$$

$\nu > 0$ is a trade-off parameter that is used to balance between the minimization of the local loss function and that of differences in model coefficients among nodes. The functions $Q_{p2}^{i,j}$ and $Q_{p3}^{i,j}$ are defined below.

Model merging (phase 2): The objective function for node i at the second phase of round j is

$$Q_{p2}^{i,j}(\mathbf{w}_j) = H \left(w_j^i - \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) \qquad (6)$$

In this phase, each node aims at minimizing the difference between the coefficients of its local model and those of a weighted average of the model from neighbouring nodes (i.e., those of the merged model). For any vector x, $H(x)$ denotes the sum of the absolute value of the elements of x.

Local training (phase 3): Let $w_{j,p2}^i = \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k$. In the third phase of a round, at each node i the model is trained over the local dataset in a way which aims at minimizing the objective function

$$Q_{p3}^{i,j}(w_{j,p2}^i) = \mathcal{L}(w_{j,p2}^i; \mathcal{S}_i). \qquad (7)$$

Thus, from (6) and (7), the expression of the objective function for every node i is

$$Q^{i,j}(\mathbf{w}_j) = H \left(w_j^i - \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) + \nu \mathcal{L}_i \left(\sum_{k \in \mathcal{N}} X_{ik}^j w_j^k; \mathcal{S}_i \right) \tag{8}$$

This sum has two distinct components, given by the local losses and the differences between local models. The objective in each round is thus to minimize the weighted sum of these two components.

We prove in the rest of this section that our DA algorithm converges by showing that the objective function gets closer to the optimal value after every round.

We make the following assumptions on the loss function, which are standard in the analysis of coordinated learning methods in [20], [21].

Assumption 1. For any $w \in \mathbb{R}^p$, the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ is convex in all rounds and $\forall i$.

Proposition 1: When Assumption 1 holds, in any round j, $Q^j(\mathbf{w}_j)$ is convex.

The proof proceeds in the same way as in [21].

Assumption 2. $\forall i$, for any $w \in \mathbb{R}^p$, the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ has L_i^{loc}-Lipschitz continuous gradient in all rounds.

Assumption 3. $\forall i$, for any $w \in \mathbb{R}^p$, there exists a $\sigma_i > 0$ such that the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ is σ_i-strongly convex.

Assumption 3 implies that $Q^j(\mathbf{w}_j)$ is σ-strongly convex with $\sigma \geq \nu\sigma_i > 0$. That is, for any $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^{N \times p}$,

$$Q^j(\mathbf{w}') \geq Q^j(\mathbf{w}) + \bigtriangledown Q^j(\mathbf{w})(\mathbf{w}' - \mathbf{w}) + \frac{\sigma}{2}\|\mathbf{w}' - \mathbf{w}\|^2, \quad (9)$$

$\forall i$, let $L_i^j = (1 + \nu L_i^{loc})$, $\alpha_i = 1/L_i^j$, and

$$L_{max} = \max_{i \in \mathcal{N}, j \in [1,J]} L_i^t.$$

A desirable property of a GL training strategy is that the objective function decreases as the iterations progress. The following result, which is the main result of GL convergence, states that when assumptions 1 to 3 hold, our GL scheme with DA merging strategy converges.

Theorem 1: Let Assumptions 1 to 3 hold. For $J > 0$, let $(\mathbf{w}_j)_{j=1}^J$ be the sequence of iterates generated by the GL training algorithm with DA merging, running for J iterations from an initial point $\mathbf{w}_0 \in \mathbb{R}^{N \times p}$. Then $\forall i$ there exist a $j' \in [1,J]$ such that, if we set $\mathbf{w}_*^{i,j'} = \arg\min_{\mathbf{w} \in \mathbb{R}^{N \times p}} Q^{i,j'}(\mathbf{w})$, we have:

$$\mathbb{E}[Q^{i,J}(\mathbf{w}_J) - Q^{i,j'}(\mathbf{w}_*^{i,j'})] \leq$$
$$\left(1 - \frac{\sigma}{nL_{max}}\right)^{J-j'} \mathbb{E}[Q^{i,j'}(\mathbf{w}_0) - Q^{i,j'}(\mathbf{w}_*^{i,j'})] \quad (10)$$

For the proof, please refer to Appendix -A.

V. NUMERICAL ASSESSMENT

To assess the performance of our GL approach numerically, we considered different measurement-based mobility datasets relative to different cities, different areas within a city, and various time intervals during the day. Specifically, a first setup is based on the LuST dataset [22], consisting of measurement-based vehicular traces from Luxembourg City, covering a time interval of 24 hours. The second scenario is based on the TAPAS Cologne dataset [23], again a measurement-based set of vehicular traces covering the greater urban area of Cologne for a whole day. Fig. 1 shows the position and size of the regions in Cologne and Luxembourg, respectively, within which our framework was assessed. In both scenarios, the region is a square of side 1 km, covering a large fraction of the city center. We used Keras [24] to implement our GL algorithms, SUMO [25] for vehicular mobility simulation, and the Omnet++ framework [26] for opportunistic communications among vehicles. Unless otherwise specified, we considered a slot duration of one second (typical of the sampling frequency of many present-day car fleet management applications) and a round duration of 15 s, equal for all nodes. Since in both scenarios, the average vehicle speed is 11 m/s (39.6 km/h), the chosen slot duration ensures that, on average,

the sets of a given node's neighbors in consecutive rounds differ significantly. At the same time, such a round duration is short enough to allow the node to capture in its local model changes in the context due to mobility. Within each round, we assumed the second and third phases of our GL schemes to take an amount of time that is negligible with respect to that required by the first phase. These assumptions are based on the fact that the time required by local training is negligible, given the small size of the model (almost 70 KB in our case) and of the local dataset. Moreover, model merging is not a computationally intensive task, consisting of a weighted sum of the model parameters. We considered a forecast horizon of 5 s, compatible with such applications as predictive collision avoidance systems, MEC processor reservation strategies, and 5G beam steering and resource allocation strategies [27].

The input of the LSTM model is composed of 12 steps, with a gap of 5 s between two consecutive steps. Thus, our LSTM model requires at least the last minute of the trajectory of a car to issue a trajectory prediction.

For local training, we adopted a mini-batch Gradient Descent approach, with batches of size 32 (as indicated in e.g., [17]), and a 10^{-3} learning rate, as suggested in [28]. Unless otherwise specified, we assume that nodes merge all received model instances during a round. The values of such model hyperparameters as the number of neurons (50), the batch size, and the number of input time steps have been tuned based on an extensive set of simulations.

To determine the local dataset of each vehicle when entering the region, for each scenario (and for each time interval), we have built a *scenario database*, consisting of past trajectories within the given region, except those taking place in the same time interval considered in our experiments. Indeed, as both datasets are relative to a single 24 h period, this choice minimizes the probability for users to have in their local dataset the same trajectory they are taking in the considered time interval. To each vehicle entering the region of interest during the given time interval, we have assigned a local dataset, different for each node, and obtained by sampling uniformly at random the scenario database. As a result, on average, in both scenarios, each vehicle's local dataset contains data corresponding to about 5 minutes of trajectories. These choices have been made, on one side, to avoid the case of nodes having no local dataset at their ingress in the region of interest. Indeed, such a "clean slate" scenario would not represent realistic settings where all nodes possess at least some relevant data. On the other side, collaborative training schemes like GL make sense when nodes are not able to achieve by themselves (i.e. with only local training) a satisfactory level of accuracy because their local dataset is not large enough. For the LBA strategy, deciding on how to partition the region has a strong impact on its performance. After an extensive empirical evaluation, in both of the considered scenarios we defined for the LBA strategy a partition consisting of 9 square cells of side 333.3 m. This choice has been a compromise between, on the one side, having trajectories that are as homogeneous as possible within the same cell, and the fact that (for the same size of the

(a) Cologne (Innenstadt)　　　　　　　　　　　　　(b) Luxembourg City

Fig. 1: Map and road grid of the considered scenarios, with the region of interest in red.

Name	City	Time	Mean sojourn time	Tx radius
Lux rush hour	Luxembourg	7:00-7:30	14 min 8 s	150 m
Lux off-peak	Luxembourg	16:00-19:00	11 min 12 s	150 m
Cologne off-peak	Cologne	14:00-17:00	4 min 4 s	450 m

TABLE II: Scenarios considered in our experiments.

local dataset) a finer partition would reduce the portions of the local data set associated with each cell. As we have verified experimentally, the latter has a negative impact on the quality of the model trained by a node as it enters the region, and thus on the speed of convergence of the training process.

In addition to our schemes, we have considered the following baseline approaches:

- *Centralized Federated Learning (FL)* (in the version described in [6]), applied to the same two stages LSTM model trained by our schemes. In FL, a parameter server orchestrates the various algorithmic stages and coordinates all the participating nodes. For a fair comparison with our schemes, we have assumed that at any time slot t, the dataset available to the FL server coincides with the union of the local datasets of all the nodes which have spent at least one time slot in the given region, from the beginning of the GL scheme up to time slot t. Namely, for nodes that have exited the region by time t, we consider the local dataset at exit time. For all the others, we consider the local dataset at time slot t. All the model meta-parameters have been set to coincide with those of our GL scheme. We assumed the rounds of the FL scheme to be the same as those of our GL scheme. Again, for fairness of comparison, at every round, we assumed random client subsampling, with an average number of selected clients coinciding with the average number of nodes that each node comes in contact with during a round.
- *Local training*, in which each node trains the two-stage LSTM model only on its local dataset, with no exchange of data or models with neighboring nodes or a server. Such training is performed at the node ingress in the given region. In addition, at regular intervals (whose duration is equal to that of a round), the local model is re-trained over the local dataset, which keeps increasing as the node moves in the

given region.
- *DFed Pow* [29]. It is a fully distributed learning scheme that inherits several features from FL schemes. Specifically, in each time slot, each node with its neighbors constitutes a federation (in the same sense as in classical FL schemes) aimed at training the given node's model. Thus, the given node plays the role of the parameter server, whereas its neighbors (who generally change from one slot to the next) play the role of clients. At every time slot, each node sends its model to all neighbors, who train it on their local dataset and send it back, to be merged with those sent by all other neighbors. Thus, in this scheme, there are as many federations as nodes.
- *Dead Reckoning (DR)*, in which each node forecasts its trajectory by extrapolating over its current position, velocity, and direction.

For our experiments, we have considered three different settings. As shown in Table II, they differ in terms of the city within which the given region is located (Luxembourg, Cologne), the start time and the duration of the considered time interval. The first scenario (denoted as *Lux rush hour*) is relative to the time interval 7:00 AM - 7:30 AM in the Luxembourg City region. This corresponds to a rush hour, with a high density of vehicles (for an average of about 300 vehicles in the given region). The second scenario (*Lux off-peak*) is relative to the time interval 4:00 PM - 7:00 PM in Luxembourg City, and it is characterized by much lower traffic intensity. In both these scenarios, the transmission radius has been set to 150 m (e.g. typical of DSRC in urban environments [30]). The third scenario (*Col off-peak*) is relative to the 2:00 PM - 5:00 PM time interval in Cologne city. It is characterized by a much lower density of vehicles and a shorter mean sojourn time with respect to Luxembourg. To enable effective opportunistic model exchanges in such a low-density scenario, we have assumed a transmission range of 450 m, e.g., compatible with BLE version 5 [31].
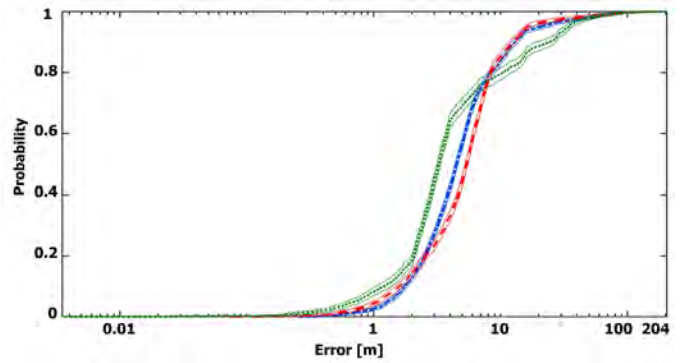
A. Performance from entrance time

One of the main performance metrics of our algorithms is the *mean error* at the t-th time slot, defined as the distance between the forecasted and the actual position, averaged across all vehicles present in the scenario during that slot. In the first
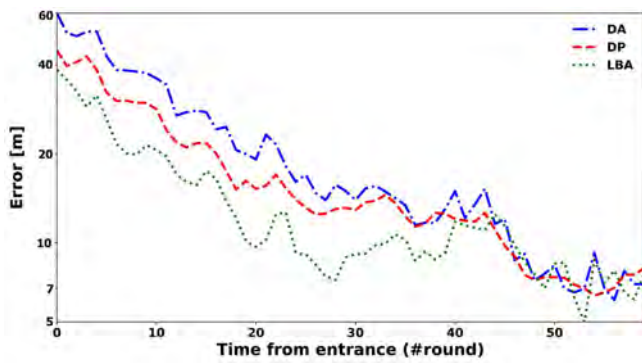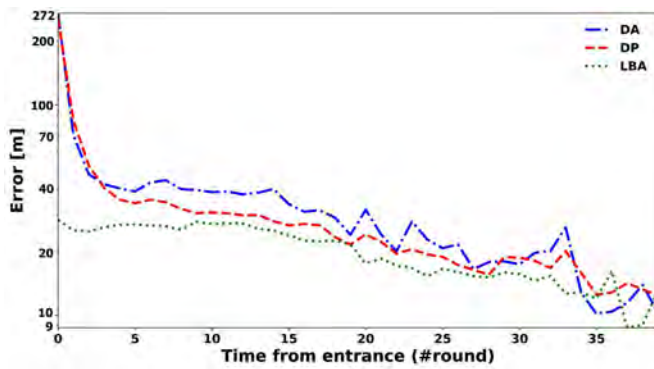
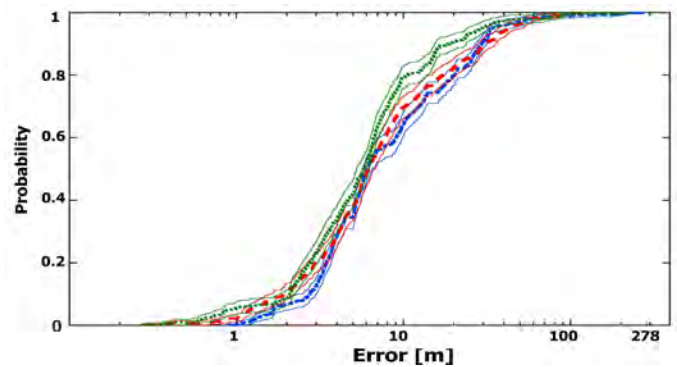(a) Mean error versus time from entrance

(b) Empirical CDF of the error for the DA, DP and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 50 and 60 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.
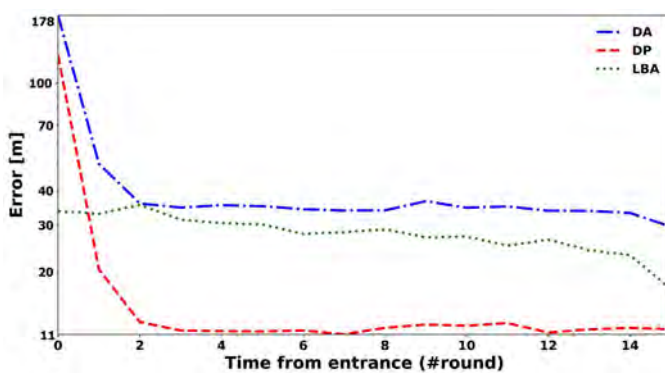
Fig. 2: *Lux rush hour* scenario.

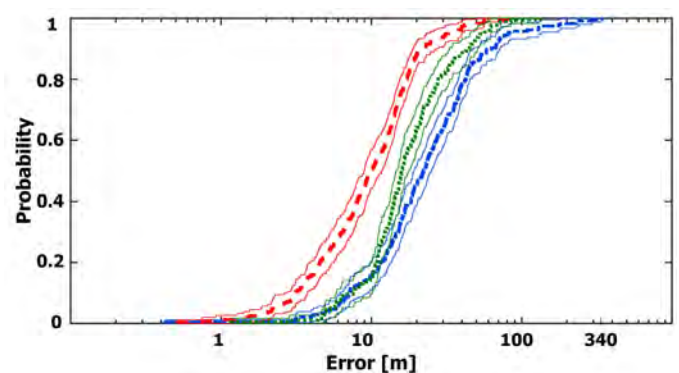(a) Mean error versus time from entrance

(b) Empirical CDF of error for the DA, DP, and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 30 and 40 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.

Fig. 3: *Lux off-peak* scenario.

(a) Mean error versus time from entrance

(b) Empirical CDF of error for the DA, DP, and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 12 and 16 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.

Fig. 4: *Cologne off-peak* scenario.

set of experiments, we have characterized the evolution of the mean accuracy achieved by the model instance of each node as a function of the time spent by the node in the region. Indeed, this measures how quickly our GL schemes improve

9

the performance of a model instance over that achievable by training it exclusively over a local dataset. Fig. 2 to 4 show the mean error as a function of the vehicle sojourn time for the three scenarios considered. In each setting, results have been averaged across the whole time interval.

As the figures show, despite the short time from bootstrap, in all scenarios, nodes are able to achieve high levels of accuracy after spending only a few minutes in the region. In the Cologne scenario, the mean error decays more rapidly than in the Luxembourg scenarios, because the larger transmission radius in the Cologne scenario brings faster exchanges of models among nodes in the region. At the same time, the shorter mean sojourn time brings the accuracy to stabilize to a higher value than in the two Luxembourg scenarios, as the GL algorithms have less time to progress further.

Note that the values of accuracy achieved by each algorithm have been evaluated on a very conservative worst-case scenario, in which each node has only a small initial dataset, and in which our scheme has been running for at most three hours. The good performance of our schemes in these conditions in spite of these assumptions suggests that in settings where our schemes have been running for longer periods, their performance is likely to be better than predicted by the reported experiments.

Our results show that, in general, the DP merging strategy performs better than DA, both in terms of convergence speed and mean error, across all experiments. Indeed, while DA's model quality estimate is not directly related to a model's performance at a given point of the region, that of DP is instead a function of each merging model's performance over the most recent part of the given node's trajectory. It is thus close to the performance that the merging model would deliver on the trajectory of the given node in the near future. As for the LBA strategy, results suggest that it is superior to the other two strategies when nodes spend a "long enough" period of time in each cell (as in the Luxembourg scenarios). In the Cologne scenario, instead, as the transmission radius is larger than the side of a cell, every user on average receives and merges model instances from neighboring cells. This averages out models across cells, destroying the ability of LBA to accurately model those mobility patterns which are specific to each cell, and thus the performance advantage of LBA over the other two GL algorithms.

From the plots of the CDFs of prediction error in Fig. 2 to Fig. 4, it can be seen that in each scenario, and for all merging strategies, the error distribution is heavily skewed towards values higher than the mean. This is more evident in Fig. 5, which shows, for each of the three scenarios and time intervals, a scatter-box plot of error. This figure shows how the vast majority of the outliers lie in the upper part of the logarithmic scale. Thus, in these scenarios, the mean error is heavily influenced by a few poor performers. Fig. 6 shows the spatial distribution of the error for the DA, DP, and LBA algorithms in the Lux off-peak scenario, averaged over a given time interval. The figure shows that the error is indeed larger at the borders of the considered region. Indeed, border areas

contain many nodes that just entered the RZ, and that did not have the time to perform many iterations of our GL scheme. Thus, not only their model is likely to perform worse than average, but also their contribution to the improvement of the model of other nodes is likely to be marginal. This slows down the improvement of model accuracy for those nodes whose trajectory is mostly contained within border areas, who end up collecting few or no high-quality models from neighbors.

In another set of experiments, we investigated how our GL algorithms evolve over time from node entrance in the region. To this end, we have tracked the evolution over time of the *weight ratio*, i.e. of the ratio between the weight used in the merging task for the local model, and the average of those attributed to models received from other nodes. Indeed, such a ratio at a given slot t indicates how much the distributed learning process weights the knowledge acquired until t versus the knowledge that it could acquire from neighbors. From Fig. 7(a) we can see that, as expected, as nodes spend more time in the region, the weight ratio (i.e. the weight given to the local model) for the DA algorithm generally increases, in all of the three scenarios. However, this increase is slow in the Luxembourg scenarios, and a bit faster in the Cologne scenario. This is because in the latter, a larger transmission radius and mean node speed account for a higher rate of contacts among nodes and thus a faster progression of the training scheme. As weights in DA are related to the number of data points incorporated in each model, faster dynamics bring to merge more models, and thus to increase more quickly the amount of data points incorporated in the local model.

The weight ratio in the DP strategy presents slightly different features, due to its being based on the losses of each model to be merged, measured on the local dataset of the merging node. In all of the three scenarios, we observe an initial decrease in the weight ratio. Indeed, in the first rounds, the local model incorporates mainly the data from its local dataset, which might not be very pertinent to the prediction task that the node needs to perform (possibly because the local dataset contains data about other parts of the city than that in which the node is located). Thus, the nodes need to learn more from the environment (i.e. from nodes who spent enough time in the region), and forget, at least in part, what they have learned on their local dataset. After this initial phase, however, the weight ratio again increases over time, as the local model starts to be fit for the node context.

In order to get a better idea of the prediction performance of the models trained with our three GL algorithms, in Fig. 8 we have plotted a portion of a node trajectory, sampled every 15 s (points denoted as GT in the figure), as well as the forecasts of our three algorithms. The differences in accuracy among these algorithms emerging from Fig. 8 are in accordance with those from Fig. 4. The figure shows that all three algorithms are able to forecast changes in the direction and speed of the vehicle with a good degree of accuracy. Among these however, DP exhibits also good accuracy in predicting a sharp slowdown of the vehicle ("halt point" in the figure) and its duration, a trajectory feature among the most complex to

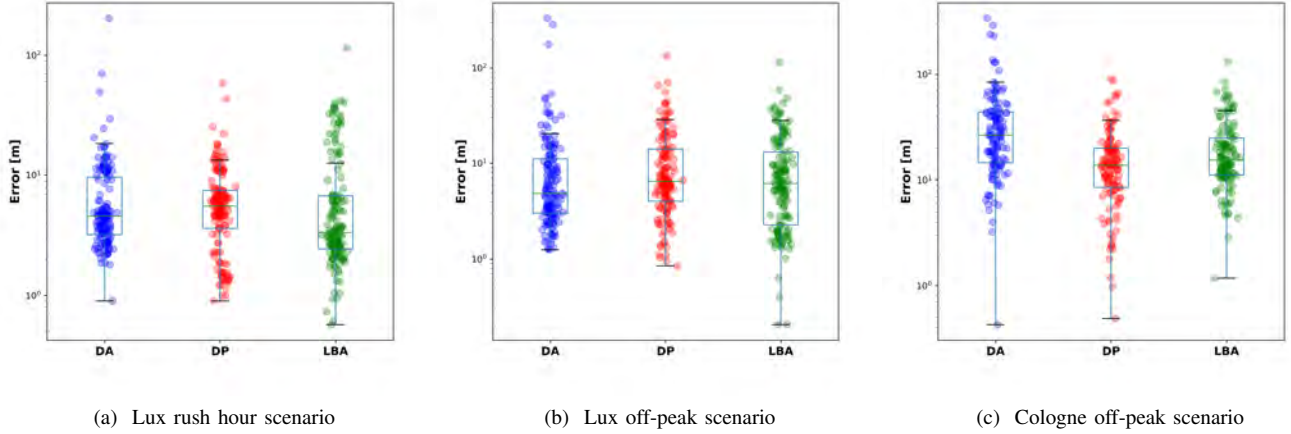(a) Lux rush hour scenario   (b) Lux off-peak scenario   (c) Cologne off-peak scenario

Fig. 5: Scatter-box plot of error for the DA, DP, and LBA algorithms. Each point is the error of a single vehicle, averaged in the time interval between rounds 50 and 60 (Lux rush hour scenario), between rounds 30 and 40 (Lux off-peak scenario), and between rounds 12 and 16 (Cologne off-peak scenario) from node entrance.

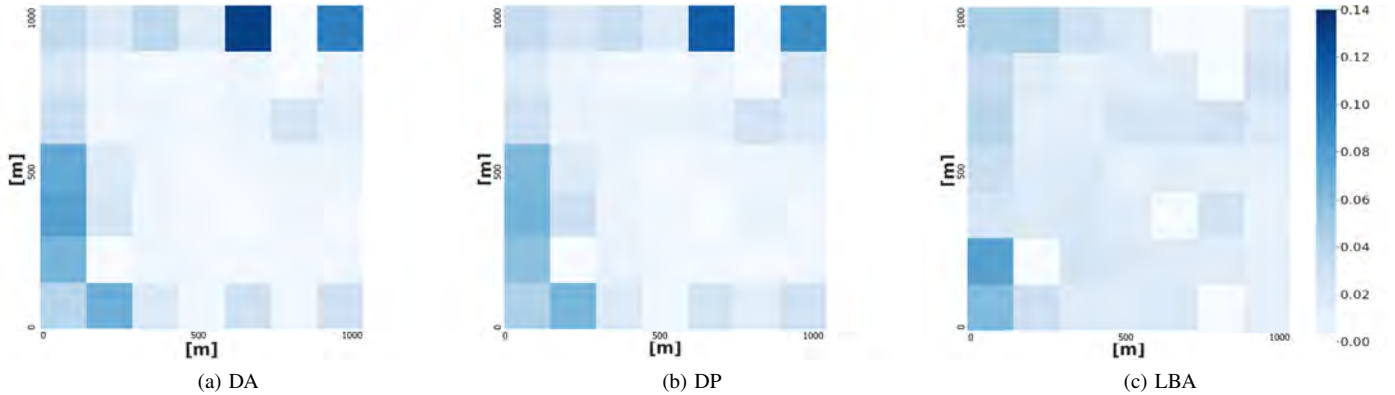(a) DA   (b) DP   (c) LBA

Fig. 6: Spatial distribution of error for the DA, DP, and LBA algorithms, averaged across vehicles and in the time interval between rounds 30 and 40 from node entrance, in the Lux off-peak scenario.

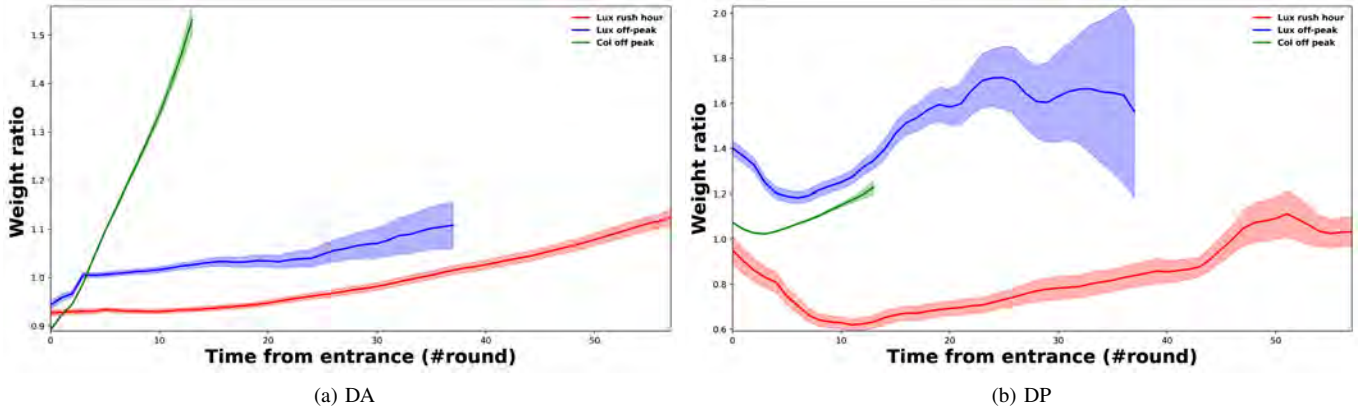(a) DA   (b) DP

Fig. 7: Local models' weight (coefficient) ratio versus time from the entrance for the DA and DP strategies, in the three considered scenarios.

forecast correctly, as it depends also on traffic conditions. Note that prediction errors can be significantly reduced by simply projecting the estimated vehicle position onto the road grid. We are not implementing this step since we prefer to report the raw performance of the vehicle trajectory estimation rather than an improved estimate which includes some postprocessing

algorithm (such as projection or fusion with other estimates, possibly dead reckoning)

B. Performance from start time

In another set of experiments, we have characterized the convergence properties of our distributed training approach. To this end, we have focused on settings and time intervals in
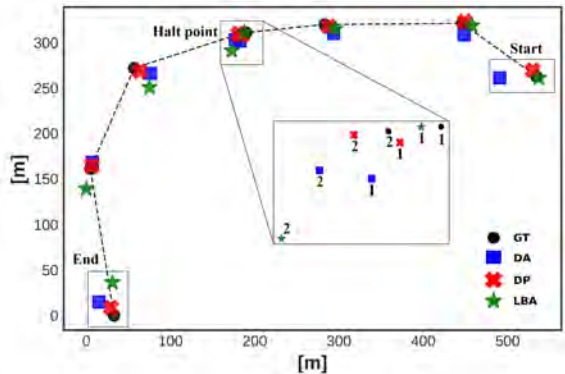
Fig. 8: Sample trajectory predicted by the DA, DP, and LBA algorithms versus the ground truth (GT) for a vehicle in the *Cologne off-peak* scenario. The time interval between two consecutive points is 15 s.

Algorithm	Mean error [m]		
	Lux rush hour	Lux off-peak	Col off-peak
DP	7.52	13.3	12.37
DA	7.18	11.90	31.30
LBA	7.22	10.28	20.25
DR	10.18	11.64	25.19

TABLE III: Mean error for the last two rounds from the start time of our GL schemes (DA, DP, and LBA) as well as for Dead Reckoning (DR), in the three considered scenarios.

which mobility patterns do not vary significantly, to reliably assess convergence and mean accuracy over time from the start of the GL schemes. Specifically, we considered the *Lux rush hour* scenario. On the one side, its duration (30 min) is long enough to allow our training framework to progress. On the other side, as we verified, it is short enough for vehicular mobility patterns not to vary significantly.

As Fig. 9 shows, our three GL algorithms steadily improve the average model performance over time. Note that the mean error in these plots is computed over all nodes present in the scenario at a given time slot, including those whose model has only been trained locally, e.g. because they have just arrived in the given region. As the plot shows, our GL schemes perform significantly better than local training, thus supporting the effectiveness of our collaborative model training strategies. Indeed, the local training strategy only marginally improves its performance over time (except for the first $10-20$ rounds), reaching values of mean error more than one order of magnitude larger than those achieved through our collaborative training schemes.

Another essential aspect emerging from these results is that the performance of our DA and DP strategies is very close to that of the equivalent centralized FL scheme, while that of the LBA strategy is significantly better, at least in the Lux rush hour scenario. This further supports the notion that training a model over a fully distributed, serverless architecture does not necessarily come at the cost of performance.

Fig. 9 allows also to compare the performance of our schemes with that of the distributed FL strategy denoted as "Flow-FL" [11]. Indeed, as shown in that paper, in the best conditions its performance can be assimilated to that of Federated Learning. This happens when node density and mobility are such as to allow maintaining a global state in a gossip-based shared memory in a reliable fashion for the whole duration of the training algorithm. Thus, in the most favorable conditions for Flow-FL, the considerations made for Federated Learning performance with respect to our GL schemes hold also for Flow-FL.

In order to assess the overall performance of our GL schemes

in terms of prediction accuracy, in Table III we have compared the mean error of our GL strategies over the last two communication rounds of the time interval of each scenario, with that of dead reckoning (DR) strategy. Indeed DR is a natural benchmark for trajectory nowcasting. As these results show, in every scenario there is at least one GL-based strategy that outperforms DR. This shows that our collaborative learning strategies, despite being implemented over a simple LSTM architecture, enable a satisfactory performance in terms of prediction accuracy over a wide range of operating conditions. It also suggests the need for tuning the choice of the specific GL approach as a function of these conditions.

As already stated, the main competing approach to our GL learning schemes is the DFed Pow algorithm of [29]. This approach is applied to a discretized version of trajectory nowcasting, consisting of forecasting in which cell of the given region a vehicle will be at some point in the future. Thus, in order to assess the relative performance of our schemes with respect to it, we have considered the LBA strategy in the Lux rush hour scenario (the best-performing strategy in that scenario). To make performance comparison possible, we cast the regression result of the LBA strategy (i.e. the prediction of the specific location in which the vehicle will be 5 s in the future) into a classification result (i.e. a prediction of the cell in which the node will be). This is implemented by partitioning the given region in 49 square cells of side 150 m (similarly to what is done in [29]).

Fig. 10 shows that LBA performs significantly better than DFed Pow in terms of mean accuracy. The low performance of DFed Pow is due to the fact that it does not allow high-performing models to be used by nodes other than the ones to which they belong. For this reason, models from nodes moving in regions with low node density (and thus unfavorable for collaborative training), or whose local dataset is not able to support the training of a high-performing model, consistently experience unacceptably poor performance. Conversely, our collaborative GL schemes enable high-performing nodes to support poor-performing ones, by transferring their models and thus their learned knowledge, and to keep on improving them as new data is constantly being generated and used for collaborative training in the given region.

C. Impact of transmission radius, and of cutoff value

One of the primary parameters affecting the performance of our schemes is the number of models merged at each round. Thus, in another set of experiments, we have characterized its impact on accuracy and convergence speed.

Among the key parameters affecting the performance of our GL schemes, transmission radius has a key role. It determines
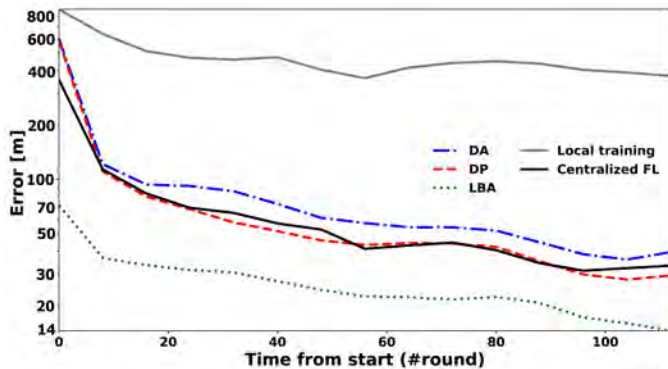
Fig. 9: Mean error versus time from the start for our GL schemes, as a function of the merging strategy, as well as for centralized FL and the fully local training, in the *Lux rush hour* scenario.
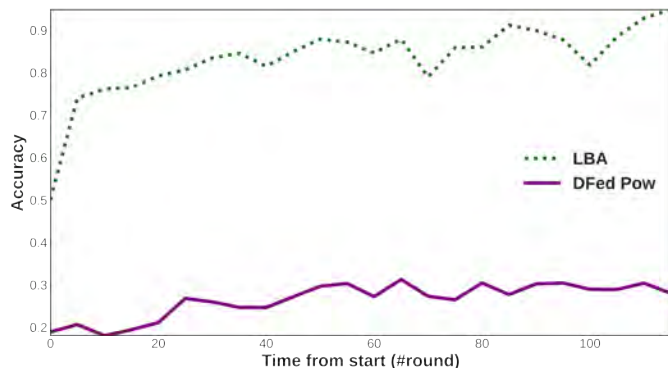

Fig. 10: Mean accuracy versus time from the start for the LBA strategy, as well as for the DFed Pow algorithm [29], in the *Lux rush hour* scenario.
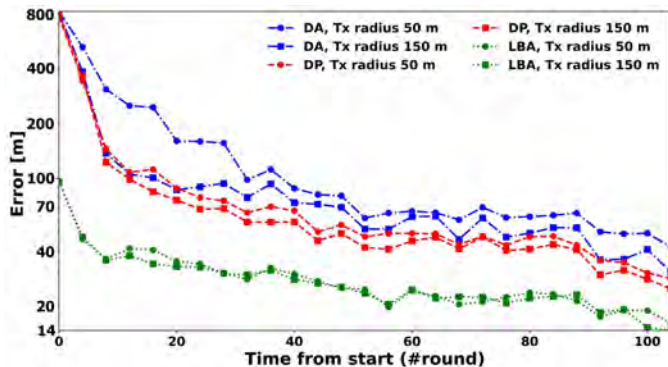

Fig. 11: Mean error versus time from the start (first 30 minutes), for different values of transmission radius, for the DA, DP, and LBA algorithms, in the *Lux rush hour* scenario.

not only the number of model instances merged at each round, but also the level of heterogeneity (e.g., in terms of accuracy) within the set of merged instances. Indeed, the trajectory prediction task is highly context-specific, particularly in realistic scenarios with nonuniformities in the spatial configuration of the road grid. As Fig. 11 shows, in our experiments in DA and DP algorithms, a larger transmission radius (and thus a larger number of models merged at each round) is associated with a decrease of the mean error at any point in time. However, our results suggest that this is less the case for the LBA scheme. As
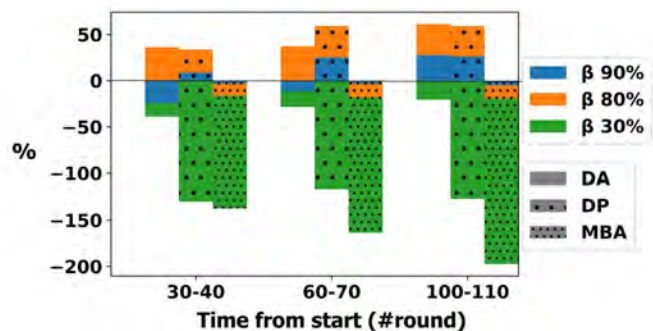

Fig. 12: % decrease in mean error for the DA, DP and LBA algorithms with respect to the setup with $\beta = 100\%$ versus time from the start, as a function of the cutoff value, Luxembourg scenario.

already seen, in LBA (which trains a number of cell-specific models in parallel), a transmission radius comparable to or larger than cell size brings to merging together models associated with neighboring cells, which are, in general, not a good fit for the given cell. In realistic and spatially inhomogeneous settings, this may potentially degrade the performance of LBA, as it destroys the cell-specific features of each model.

One of the key features of model merging is that it does not necessarily produce a more accurate model than any of the models to be merged. This is the case, for instance, when those models are not sufficiently affine (e.g., in our setup, in terms of the specific city area within which a trajectory prediction has to be produced), or when some of them do not perform well for the specific prediction task at hand. As explained in our algorithms, the likelihood of this effect manifesting itself is minimized through an appropriate choice of the weights used in the merging process (which are, as we have seen, somehow related to a notion of the relevance of the model for the specific task). In addition, this is also achieved by tuning the cutoff value β, which prevents models with small values of weight from taking part in the merging process.

Thus, in another set of experiments, we assessed the impact of the cutoff value (expressed as the ratio of the sum of the weights retained in the merging process over the sum of the weights of all models to be merged) on prediction accuracy. As Fig. 12 suggests, there exists an optimal value of the cutoff, which in the considered settings is between 80% and 90%. Indeed, in all three algorithms, when the value of the cutoff is to retain only 30% of the contributions, the mean accuracy worsens with respect to no cutoff. In contrast, higher percentages of retained models yield an improvement in accuracy. As expected, in our experiments, the performance improvements and the optimal cutoff value highly depend on the specific merging algorithm and several features of the considered scenario, such as mean node density, mobility pattern, and mean vehicle sojourn time in the region. It suggests the need for strategies that adapt β to the specific setup.

D. Impact of nonstationarity in vehicular mobility

A clear feature arising from the numerical assessment of our algorithms is the high impact that node density and mobility patterns have on accuracy and convergence rate. Since these

(a) 2:00 PM - 5:00 PM in the Cologne   (b) 4:00 PM - 7:00 PM in the Luxembourg
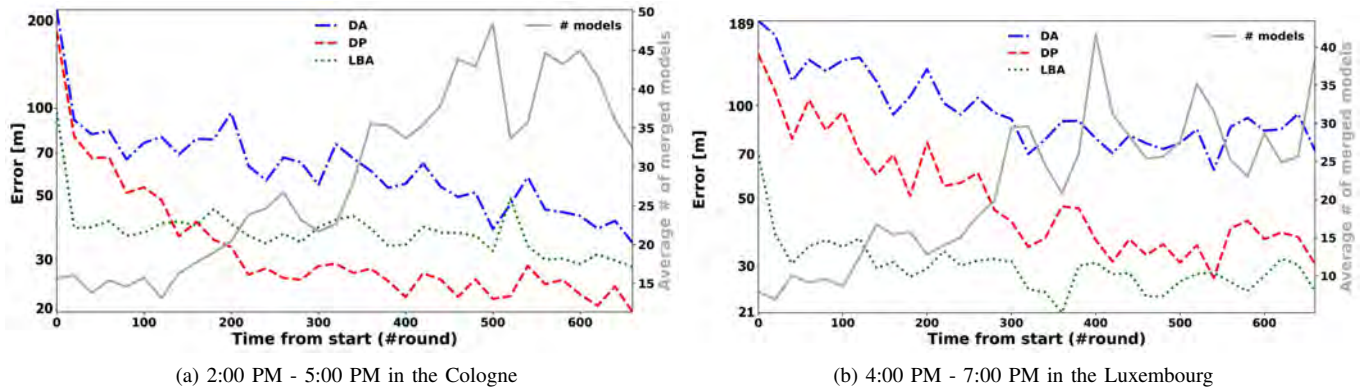
Fig. 13: Mean error versus time for the DA, DP and LBA algorithms.

features change substantially over time in realistic scenarios, we performed a set of experiments over time windows during which the configuration of vehicular traffic exhibits significant variations. Specifically, we have considered a time interval of three hours, from 4:00 PM to 7:00 PM in the Luxembourg scenario and from 2:00 PM to 5:00 PM in the Cologne scenario. As Fig. 13 shows, the substantial differences in a road grid configuration, time of the day, and mobility patterns, in both scenarios, the mean error decreases steadily for increasing time from the start of our schemes, despite variations of up to one order of magnitude in the number of vehicular nodes in the region. These features suggest that, except for minor fluctuations in the mean error, in realistic scenarios, our GL schemes are able not only to adapt in a timely manner to changes in mobility patterns but also to keep on improving the model performance over time.

VI. RELATED WORK

Recently, distributed learning architectures have received a lot of attention from the research community [21], [32]–[38], aiming at overcoming some of the main limitations of centralized approaches (such as limited scalability, communication bottlenecks, performance under data imbalance and heterogeneity, under device heterogeneity and churn, to name a few) [32]. Decentralized Federated Learning, based on peer-to-peer communication between agents, has been proposed in [34], [35] as a way to address some of these shortcomings. Under this mechanism, each worker only demands communication with its neighbours for model exchanging in synchronous or asynchronous manners. To achieve a high training efficiency, however, proposed synchronous schemes are based on static topologies, such as a ring [36] or a static mesh [21], [37]. Thus they do not apply to dynamic settings. [38] presents a scheme with full connectivity among nodes, which adopts a hypothesis transfer learning approach to derive merged models. This work shows that even a straightforward GL scheme may enable substantial gains in terms of communication efficiency while at the same time achieving accuracy levels which are comparable to or even better than those of centralized approaches. Overall, the synchronisation requirement in all of these schemes implies that they are severely hampered by heterogeneity in computing power among nodes (particularly in training time). This wors-

ens as the number of workers increases [34], [39]. For this reason, asynchronous solutions [8], [10] have been designed. [8] proposes a solution in which local models are distributed over a logically fully connected peer-to-peer network. However, the full connectivity requirement still brings scalability and connectivity issues. [10] presents a fully serverless FL approach in which nodes receive a combined model from their neighbours in a static topology, and each one independently performs training on its local dataset in an asynchronous manner. [40], [41] explore the relationship between the (static) connectivity graph structure and convergence rate. Again, these results consider scenarios where each node communicates with all other nodes and/or the connectivity graph is static. [42] addresses the churn issue in a scenario where a set of static nodes learn a single global model through a gossip-based communication scheme. This work shows that, at least in static node scenarios, gossip schemes may achieve a level of accuracy comparable to (if not better than) FL. By considering a static scenario, however, [42] does not account for one of the critical elements of gossiping in a dynamic setting, i.e. the tight link between proximity in space among nodes, context information, correlation in the composition of local datasets, and correlation in tasks among nodes.

[21] proposes the first version of a GL scheme for multi-task learning on a static mesh network, proving its convergence. Unlike our work (and the vast majority of practical cases), such a GL scheme assumes that the relationship among tasks of different nodes is known in advance, and it does not specify how those weights should be derived. This is addressed in [43], which proposes a scheme for learning, in a distributed fashion, both the relationship among tasks and the weights to be used in model merging. However, such a scheme assumes a static node mesh without churn, making it unfit for applications in dynamic scenarios, such as in vehicular/pedestrian settings. Thus, none of these works considers scenarios with a dynamic topology among nodes in which patterns of model exchanges result from opportunistic contacts. This leaves open a set of issues relative to the performance of GL schemes, in terms of convergence and convergence speed, as well as the accuracy of the trained models [44], which we start addressing in the present work.

14

The issue of short-term vehicular trajectory prediction has recently received much attention, given the growing amount of applications and use cases, both in ITS and Autonomous driving domain, and in 5G and beyond network optimization and dynamic management [45], [46]. The growing availability of a large amount of car/user floating data has generated a large body of works which apply various deep learning techniques to the problem of predicting network-wide vehicle movement patterns in urban scenarios [2], [45]–[47]. [48] applies a LSTM architecture to the problem of vehicle nowcasting in urban scenarios based on a centralized scheme.

The application of a gossip-based learning scheme for the issue of trajectory prediction has been proposed in [11]. The scheme aims at training a *single* ML model. It is based on a distributed implementation of a central coordinated function, with a global state maintained in a gossip-based shared memory, periodically updated via flooding. Consequently, such a single consensus scheme works well only when nodes form a single connected component, which drives the evolution of the learning process. Indeed, nodes who disconnect from such component cannot participate in the process until they reconnect. Differently from our scheme, this feature makes the solution in [11] ineffective in scenarios with more than one cluster, or in sparse scenarios where store-carry-and-forward is the main mode of content diffusion.

In a previous work [29], we presented a distributed scheme applied to a classification problem to derive a forecast about the section of the considered urban region where a vehicle will be at a given time in the future. Specifically, the given region of the plane was partitioned into *cells*, and each vehicle tried to predict in each cell it would be in h slots ahead in the future. Each node implemented a distributed version of FL, periodically sending its model to all neighbours, who train it on their local database and send it back to be merged with those sent by all other neighbours.

However, as shown in [29], this scheme implies twice the amount of model exchanges than the GL schemes in the present paper. In addition, it requires a very high number of local training steps per node, making it unsuitable for resource-constrained (computing and/or energy) devices. Moreover, the schemes in [29] suffer from a very high disparity in performance among nodes, with nodes in regions with low node density consistently experiencing unacceptably poor performance.

Conversely, the approach in the present paper allows high-performing nodes to quickly disseminate their models to those that cannot build a high-performing one. Moreover, in this way, (good) models persist probabilistically in the region over time, and they are constantly improved (and updated, as mobility patterns change over time) even by those nodes which do not possess enough neighbours or data to be able to train a high-performing model themselves.

VII. Conclusions and future work

Vehicular position nowcasting has recently received much attention due to the growing amount of applications and use cases in 5G/B5G networks. In this paper, we propose a set of gossip learning algorithms for collaborative training of ML models for nowcasting in dense urban environments where node mobility and network churn are high.

We show that very good performance can be achieved in realistic dynamic environments within only a few hours from the initial installation of the position nowcasting application and with the very small initial dataset for each node. Our algorithms are proven to converge under very mild assumptions on the connectivity patterns and the data. We showed that our GL algorithms trained over a fully distributed, serverless architecture perform at least as well as server-based approaches such as federated learning.

Future developments of this work will include, first of all, attempts to reduce the average and variance of the error in the vehicle position estimation, possibly integrating the prediction of the GL model with data available onboard. Another future direction of the investigation will involve extending the analysis of the feasibility of GL training approach to other learning tasks, and a more fine-grained analysis of the impact of the main system parameters on GL performance, by means of synthetic dynamic graphs. Finally, other contributions rely on evaluating the effects of heterogeneity in node mobility, connectivity among nodes, and computing power available at nodes and investigating their effect on the convergence of our schemes and model accuracy.

References

[1] Z. Xiao, P. Li, V. Havyarimana, G. M. Hassana, D. Wang, and K. Li, "GOI: A Novel Design for Vehicle Positioning and Trajectory Prediction Under Urban Environments," *IEEE Sensors Journal*, vol. 18, no. 13, pp. 5586–5594, 2018.

[2] L. Lin, S. Gong, T. Li, and S. Peeta, "Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles," in *The Autonomous Vehicles Symposium*, vol. 2018, 2018.

[3] Q. Liu, G. Chuai, J. Wang, and J. Pan, "Proactive mobility management with trajectory prediction based on virtual cells in ultra-dense networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8832–8842, 2020.

[4] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[5] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A Distributed Machine-learning System." in *Cidr*, vol. 1, 2013, pp. 2–1.

[6] 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.

[7] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[8] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," *arXiv preprint arXiv:1611.09726*, 2016.

[9] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.

[10] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[11] N. Majcherczyk, N. Srishankar, and C. Pinciroli, "Flow-FL: Data-driven federated learning for spatio-temporal predictions in multi-robot systems," in *IEEE ICRA*, 2021, pp. 8836–8842.

[12] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *2017 ITSC*. IEEE, 2017, pp. 353–359.

[13] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, "Stochastic predictive control of autonomous vehicles in uncertain environments," in *12th International Symposium on Advanced Vehicle Control*, 2014, pp. 712–719.

[14] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE ITSC*. IEEE, 2017, pp. 399–404.

[15] P. Ondrúška and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proceedings of AAAI*, 2016, pp. 3361–3367.

[16] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 797–802.

[17] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *IEEE ITSC*, 2017, pp. 353–359.

[20] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[21] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 473–481.

[22] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario," in *IEEE VNC*, Dec 2015, pp. 1–8.

[23] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2013.

[24] F. Chollet, *Deep Learning with Python and Keras: The practical manual from the developer of the Keras library*. MITP-Verlags GmbH & Co., 2018.

[25] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using sumo," in *IEEE ITSC*, 2018, pp. 2575–2582.

[26] A. Varga, *OMNeT++*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59.

[27] A. Mahmood, L. Beltramelli, S. Fakhrul Abedin, S. Zeb, N. I. Mowla, S. A. Hassan, E. Sisinni, and M. Gidlund, "Industrial iot in 5g-and-beyond networks: Vision, architecture, and design trends," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4122–4137, 2022.

[28] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[29] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "Gossip learning of personalized models for vehicle trajectory prediction," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2021, pp. 1–7.

[30] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of dsrc and cellular network technologies for v2x communications: A survey," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9457–9470, 2016.

[31] "Things you should know about bluetooth range," https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range, (Accessed on 06/09/2022).

[32] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[33] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[34] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[35] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Communication efficient framework for decentralized machine learning," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2020, pp. 1–5.

[36] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[37] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[38] L. Valerio, A. Passarella, and M. Conti, "Hypothesis transfer learning for efficient data computing in smart cities environments," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–8.

[39] L. Zhao, W.-Z. Song, X. Ye, and Y. Gu, "Asynchronous broadcast-based decentralized learning in sensor networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 33, no. 6, pp. 589–607, 2018.

[40] L. Giaretta and S. Girdzijauskas, "Gossip Learning: Off the Beaten Path," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1117–1124.

[41] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2350–2358.

[42] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021.

[43] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully decentralized joint learning of personalized models and collaboration graphs," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 864–874.

[44] A. A. Alkathiri, L. Giaretta, S. Girdzijauskas, and M. Sahlgren, "Decentralized word2vec using gossip learning," in *NoDaLiDa*, 2021.

[45] S. Choi, H. Yeo, and J. Kim, "Network-wide vehicle trajectory prediction in urban traffic networks using deep learning," *Transportation Research Record*, vol. 2672, no. 45, pp. 173–184, 2018.

[46] H. Jiang, L. Chang, Q. Li, and D. Chen, "Trajectory prediction of vehicles based on deep learning," in *IEEE ICITE*, 2019, pp. 190–195.

[47] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 126–12 134.

[48] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "Automec: Lstm-based user mobility prediction for service management in distributed mec resources," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2020, pp. 155–159.

Mina Aghaei Dinani (mina.aghaeidi@hes-so.ch) is currently a second-year PhD candidate at HES-SO Valais and the University of Neuchatel, Switzerland. Her doctoral work explores a multidisciplinary approach encompassing Machine Learning, Gossip Learning, and opportunistic communication. She received her M.Sc. in Communications and Computer Networks Engineering at Politecnico di Torino, Italy, in 2020.

Adrian Holzer (adrian.holzer@unine.ch) is a Professor of Management Information Systems at the University of Neuchâtel. He holds a PhD in Information Systems from the University of Lausanne. He was a research associate at EPFL and an SNF research fellow at Polytechnique Montréal. His area of expertise covers context-aware distributed computing from messaging protocols to human computer interaction. His research was published in various renown outlets such as IEEE Transaction on Mobile Computing, Computer Networks, Communication of the AIS, European Journal of Information Systems, or ACM CHI.

Hung Nguyen (hung.nguyen@adelaide.edu.au) is an associate professor in Computer Science at the University of Adelaide, Australia, where he has been since 2009. He obtained his PhD in computer networking from EPFL. His current research interest is on models and algorithms for improving network performance, security, and resiliency, especially for IoT and wireless networks. He has published over 50 papers on these topics in the last ten years.

Marco Ajmone Marsan (marco.ajmone@polito.it) Marco Ajmone Marsan is a part-time research professor at the IMDEA Networks Institute in Spain. From 1974 to 2021 he was at the Politecnico di Torino, in the different roles of an academic career, with an interruption from 1987 to 1990, when he was a full professor at the Computer Science Department of the University of Milan. He obtained degrees in EE from the Politecnico di Torino and the University of California, Los Angeles (UCLA). He served in the editorial board of several international journals, and chaired the steering committee of the ACM/IEEE Transactions on Networking. He was the General Co-chair of Infocom 2013, and will be the General Co-chair of ICC 2023. He is a Fellow of the IEEE, and a member of the Academia Europaea and of the Academy of Sciences of Torino. He is qualified as "ISI Highly Cited researcher" in computer science. He received a honorary degree in Telecommunication Networks from the Budapest University of Technology and Economics. He was named Commander of the Order of Merit of the Republic of Italy by the President of Italy. He was the Vice-Rector for Research, Innovation and Technology Transfer at the Politecnico di Torino, and the Director of IEIIT-CNR. He was the Italian delegate in the ICT and IDEAS Committees of FP7.

Gianluca Rizzo (gianluca.rizzo@hevs.ch) is Associate Professor of Computer Science at Università di Foggia (UNIFG), Italy, and Senior Research Associate at HES-SO Valais, Switzerland. Previously, he has been with Institute IMDEA Network, and Adjunct Professor at UC3M, Madrid. He received his M.Sc. in EE from Politecnico di Torino in 2001, and his PhD in Computer Science in 2008 from EPFL, Switzerland. His main research interests are in performance evaluation of distributed systems.

A. Proof of Theorem 1

Proposition 2: It is given $Q^j(\mathbf{w})$ with DA merging strategy. Then $\forall \mathbf{w} \in \mathbb{R}^{N \times p}$, $\forall 0 \leq j - \delta \leq j$ with $\delta \in \mathbb{N}$, $\forall i$,

$$E[Q^{i,j-\delta}(\mathbf{w})] = E[Q^{i,j}(\mathbf{w})]$$

Proof: Let us consider the equation for the performance estimator in DA at round t:

$$\xi_j^i = \frac{\sum\limits_{k \in K_{j-1}^i} (\xi_{j-1}^k)^2}{\sum\limits_{h \in K_{j-1}^i} \xi_{j-1}^h} = \frac{\sum\limits_{k \in K_{j-1}^i} \xi_{j-1}^k}{\sum\limits_{h \in K_{j-1}^i} \xi_{j-1}^h} \frac{\sum\limits_{k_1 \in K_{j-2}^k} (\xi_{j-2}^{k_1})^2}{\sum\limits_{h \in K_{j-2}^k} \xi_{j-2}^h}$$

Proceeding backwards, and substituting recursively the same expression, we get

$$= \frac{\sum\limits_{k \in K_{j-1}^i} \xi_{j-1}^k}{\sum\limits_{h \in K_{j-1}^i} \xi_{j-1}^h} \cdot$$

$$\cdot \left(\frac{\sum\limits_{k_1 \in K_{j-2}^k} \xi_{j-2}^{k_1}}{\sum\limits_{h \in K_{j-2}^k} \xi_{j-2}^h} \cdot \left(\cdots \cdot \left(\frac{\sum\limits_{k_{\delta-1} \in K_{j-\delta}^{k_{\delta-2}}} (\xi_{j-\delta}^{k_{\delta-1}})^2}{\sum\limits_{h \in K_{j-\delta}^{k_{\delta-2}}} \xi_{j-\delta}^h} \right) \cdots \right) \right)$$

This expression is thus the product of δ factors. As we can see, each factor is dependent on the preceding one for the composition of the set K. For instance, in the second factor the set K_{j-2}^k denotes the set of nodes which in round $j-2$ sent their local instance to node k, which is the index of a node considered in round $j-1$ in the first factor. In our GL scheme, the process which assigns a set K of contributors to a given node is determined by the node's position, speed, the conditions of the wireless channel between two nodes in range, among others. Given the stochastic nature of these factors, we model their compound effect on the composition of the set K by assuming it to be a random collection of node indices among $\mathcal{N}$. We further assume now the composition of any two such sets to be independent, when associated to different nodes and/or different rounds. This is clearly an approximation, as nodes close among them will tend to have at least part of the elements of the set K in common, and the same is true for a same node in consecutive slots. With this assumption, every fraction in the above expression, and relative to a given slot, is a random variable independent from those relative to other slots. Thus, in the computation of $E[\xi_j^i]$, the expectation of the product becomes the product of the expectations. We have thus that

$$E\left[\xi_j^i\right] = \left(\prod_{\tau=j-\delta+1}^{j} E\left[\frac{\sum\limits_{k \in \mathcal{K}_\tau} \xi_\tau^k}{\sum\limits_{h \in \mathcal{K}_\tau} \xi_\tau^h} \right] \right) E\left[\frac{\sum\limits_{k \in \mathcal{K}_{j-\delta}} (\xi_{j-\delta}^k)^2}{\sum\limits_{h \in \mathcal{K}_{j-\delta}} \xi_{j-\delta}^h} \right]$$

$$= E\left[\frac{\sum\limits_{k \in \mathcal{K}_{j-\delta}} (\xi_{j-\delta}^k)^2}{\sum\limits_{h \in \mathcal{K}_{j-\delta}} \xi_{j-\delta}^h} \right]$$

Thus $E[\xi_j^i] = E[\xi_{j-1}^i] = \ldots = E[\xi_{j-\delta+1}^i]$. If we assume that the process of arrivals and departures of node is stationary, and that the sampling function which samples randomly the set of nodes present in the region into a set K to be the same for all nodes, and that the $\xi_{j-\delta+1}^i$ are drawn from a same distribution $\forall i$, then $E[\xi_j^i] = E[\xi_j^j]$ for any couple of nodes i, j present at round t. In the same way, we can show that, for any node i at rounds j and j',

$$E\left[\sum_k X_{ik}^j w_j^k \right] = E\left[\sum_k X_{ik}^{j'} w_{j'}^k \right].$$

$\blacksquare$

As a corollary, it is easy to see that $E[Q^{i,j+1}(\mathbf{w}_{j+1})] = E[Q^{i,j}(\mathbf{w}_{j+1})]$. We now prove Theorem 1. Using Taylor's expansion, and Assumption 1 and 2, and considering a step of the stochastic gradient descent for which the variation is only in the dimension i', we get:

$$Q^{i,j}(\mathbf{w}_{j+1}) = Q^{i,j}\left(\mathbf{w}_j - \frac{1}{L_i^j}[\triangledown Q^{i,j}(\mathbf{w}_j)]_{i'} \times e_{i'} \right)$$

using Taylor's expansion

$$\leq Q^{i,j}(\mathbf{w}_j) + [\triangledown Q^{i,j}(\mathbf{w}_j)]_{i'} \left(-\frac{1}{L_i^j}[\triangledown Q^{i,j}(\mathbf{w}_j)]_{i'} \right)$$

$$\leq Q^{i,j}(\mathbf{w}_j) - \frac{1}{2L_{max}} \left[\triangledown Q^{i,j}(\mathbf{w}_j) \right]_{i'}^2$$

Taking the expectation of both sides,

$$\mathbb{E}[Q^{i,j+1}(\mathbf{w}_{j+1})] \leq \mathbb{E}[Q^{i,j}(\mathbf{w}_j)] - \frac{1}{2nL_{max}} \mathbb{E}\left[\|(\triangledown Q^{i,j}(\mathbf{w}_j))\|^2 \right].$$
(11)

Here, we used the facts that $\mathbf{w}_j$ does not depend on i' and that i' is chosen randomly among $[1, \ldots, n]$. Let

$$\theta_j^i := \mathbb{E}[Q^{i,j}(\mathbf{w}_j)] - Q^{i,1}(\mathbf{w}_*^{i,1})$$

then

$$\theta_{j+1}^i \leq \theta_j^i - \frac{1}{2nL_{max}} \mathbb{E}\left[\|(\triangledown Q^{i,j}(\mathbf{w}_j))\|^2 \right]. \quad (12)$$

When $Q^{i,j}$ is σ-strongly convex with modulus $\sigma > 0$, we get

$$Q^{i,j}(\mathbf{w}_*^{i,1}) \geq Q^{i,j}(\mathbf{w}_j) - \frac{1}{2\sigma} \| \triangledown Q^{i,j}(\mathbf{w}_j)\|^2. \quad (13)$$

This implies that

$$\| \triangledown Q^{i,j}(\mathbf{w}_j)\|^2 \geq 2\sigma \left(Q^{i,j}(\mathbf{w}_j) - Q^{i,j}(\mathbf{w}_*^{i,1}) \right) \quad (14)$$

Using again Proposition 2 we have

$$\| \triangledown Q^{i,j}(\mathbf{w}_j)\|^2 \geq 2\sigma \left(Q^{i,j}(\mathbf{w}_j) - Q^{i,1}(\mathbf{w}_*^{i,1}) \right) \quad (15)$$

Combining (12) with (15) and taking the expectation, yields

$$\theta_{j+1} \leq \theta_j - \frac{\sigma}{nL_{max}} \theta_j = \left(1 - \frac{\sigma}{nL_{max}} \right) \theta_j. \quad (16)$$

Recursively applying this inequality $J - j'$ times yields Equation 10.