Coreset-sharing based Collaborative Model Training among Peer Vehicles

Han Zheng, Mengjing Liu, Fan Ye, Yuanyuan Yang
Department of Electrical and Computer Engineering, Stony Brook University
{han.zheng, mengjing.liu, fan.ye, yuanyuan.yang}@stonybrook.edu

***Abstract*—Decentralized model training for on-road vehicles offers the potential to harness huge amounts of data at low costs. However, existing approaches usually depend on the existence of a coordinator, tight synchronization, or a connected cluster, all of which can be challenging or infeasible for fast-moving vehicles. In this work, we propose Learning by Chatting (LbChat), a fully decentralized and asynchronous model training approach leveraging coreset-sharing to eliminate the need for a coordinator, tight synchronization, or even a connected cluster. Different from conventional decentralized learning methods, a vehicle not only exchanges its local model but also a coreset, a condensed abstract of its local training data, with opportunistically encountered peers. A vehicle measures its model's performance on a peer's coreset, and a lower performance indicates more different data, thus a more "valuable" model from the peer. Such models are compressed less during exchange to maximize the aggregate gain from each encounter. Extensive evaluations on the driving decision-making task demonstrate that LbChat is strongly competitive with the central server or roadside infrastructure-based approaches (e.g., federated learning). Compared to recent fully decentralized vehicular learning benchmarks, LbChat outperforms them significantly by up to 20% higher driving success rate in the most challenging driving condition, demonstrating the power of insights gained from coresets on peer models' value.**

***Index Terms*—Vehicular network; coreset; vehicular model training; opportunistic communication; decentralized learning**

I. Introduction

The demand for models in autonomous vehicles spans various applications. Collecting data required for training these models with dedicated fleets entails tremendous resource investments [1] and is difficult to cover various critical but corner cases [2]. In response to this data-hungry scenario, as an alternative, federated learning facilitates leveraging data from numerous on-road vehicles. While recent research has demonstrated the effectiveness of such approaches [3], they still encounter challenges stemming from limited/unstable data rates to the backend infrastructure and sporadic cellular network coverage [4]. As a result, fully decentralized methods emerge as a promising solution [5] [6], as they empower vehicles to train local iterations and communicate models directly with nearby peers, thereby generating aggregated models and circumventing the aforementioned constraints.

In the landscape of decentralized/gossip learning, existing works can be broadly categorized into two groups: 1) Synchronous learning (e.g., decentralized federated learning) that requires tight synchronization among work nodes during the whole training process: all nodes are expected to start and finish each global "round" simultaneously. A round refers to a period during which all nodes train local iterations and exchange models with their neighbors. However, this becomes extremely challenging or simply infeasible among on-road vehicles. The *contact duration* between two vehicles in which they are within the radio range and can communicate, may last only for tens of seconds or shorter [7]. Wherever central servers or road-side units (RSUs) are not available (e.g., limited cellular coverage or RSU deployment), multi-hop coordination thus tight synchronization is difficult to achieve due to the limited communication range, loss over wireless channels, and high mobility of vehicles. While some recent works show tolerance to stragglers and communication delays [8], they still adhere to a round-based approach and assume delays are bounded. 2) Asynchronous learning does not require a common notion of "round". Instead, individual nodes train, exchange, and merge models whenever they encounter neighbors. However, existing works (e.g. [9]) usually assume a connected cluster/topology of vehicles, which is often infeasible among fast-moving ones. Notably, in both categories, existing works exchange among peers only models but not local training data.

In this paper, we propose Learning by Chatting (LbChat), a fully decentralized and asynchronous model training approach, where vehicles exchange not only models but also *coresets* [10], succinct summaries of the local training datasets, with opportunistically encountered peers. By measuring the model's performance on a peer's coreset and comparing how much it is lower than that of the peer's model, a vehicle can assess the "value" of that peer's model: a larger performance gap indicates the peer's model is potentially better-trained on much different data, thus regarded more "valuable" to this vehicle. This enables vehicles to allocate communication resources to valuable models within short contact durations to maximize the aggregate gain. Thus we eliminate the need for coordinators, consistent global rounds, or even connected topology among moving vehicles, which are difficult to maintain.

Specifically, a vehicle keeps training local iterations using the local dataset, while opportunistically exchanging coresets and aggregating models from other vehicles upon encounters. During pairwise "chat", vehicles first exchange their coresets

This work is supported in part by NSF grant 2007715.

and evaluate the local model's performance on the peer coreset, subsequently exchanging the evaluation results. Based on the results, vehicles dynamically optimize the model compression ratios for the exchange. Intuitively, this optimization tends to invest more time in receiving an expected "valuable" model with a lower compression ratio, and less time to acquire a less valuable model with a higher compression ratio. Since the coreset size can be substantially smaller than that of a vehicular learning model (e.g. two orders of magnitude in our experiments, a coreset at 0.6MB while an imitation learning model at 52MB), exchanging coresets takes very little time. Yet this small "investment" allows vehicles to make intelligent decisions on using the limited contact durations for the most "valuable" models for maximum gain. We assume adequate incentive mechanisms (e.g. [11]) and privacy protection techniques (e.g. [12]) exist for sharing such coreset data.

Our contribution is three-fold:

• We introduce local training data sharing into peer vehicular model training, whereas previous works only exchange models. We explore a fully decentralized and asynchronous paradigm, which eliminates the necessity for consistent global "rounds" or connected topology, hard to maintain under short contact durations and wireless losses among moving vehicles.

• We propose LbChat, a coreset-based approach, for optimizing collaborative model training. LbChat utilizes coreset as an estimator to quantitatively assess the value of models from peer vehicles. By constructing and exchanging coresets, and conducting evaluations on them, vehicles adapt the model compression ratio during exchanges and weights in aggregation, aiming to maximize the aggregate benefits from each encounter. Vehicles also absorb peer coresets and expand local datasets, enriching local training data. Route sharing is further integrated to address wireless losses by prioritizing neighbors with higher probabilities of exchange completion.

• We evaluate our method on the popular driving decision-making task using the CARLA simulator. In online evaluations, LbChat achieves driving success rates strongly competitive to central server/roadside infrastructure-based approaches (e.g., federated learning). When compared to fully decentralized vehicular learning benchmarks subject to the same constraints, LbChat outperforms them by up to 20% higher driving success rate in the most challenging driving conditions, demonstrating the effectiveness of LbChat and unveiling the potential of data sharing in peer vehicular model training.

To our best knowledge, this is the first work to explore a coreset-sharing based solution for decentralized and asynchronous vehicular model training. The insights of peer models' value through coresets is the key for optimized model exchange and aggregation.

II. Preliminaries

A. Communication model & Opportunistic model training

We consider a set of vehicles $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, each equipped with a local dataset and maintaining a local model. Continuously training local iterations, the vehicles share coresets, pertinent information (e.g. route), and exchange models with encountered peers. We assume that vehicles have access to assistant information (e.g. future routes in next few minutes, which can be obtained from navigation services) and possess similar hardware as well as communication capabilities. [1] We denote the available bandwidth of v_i in a pair-wise communication as B_i. We also assume the loss in wireless communication can be roughly estimated based on certain models (e.g., distance based ones [13] [7]).

In opportunistic vehicular model training, each vehicle trains a model locally with an identical model structure. Each vehicle v_i possesses its local dataset $D_i \subset \mathcal{D}$ and maintains its local model $\boldsymbol{x}_i \in \mathcal{P}$, with $\mathcal{D}$ representing the metric space and $\mathcal{P}$ denoting the parameter space. We assume that the datasets on vehicles are i.i.d. and the models on vehicles have the same initialization. A vehicle exchanges its local model with encountered peers and aggregates received parameters to update its local model. In LbChat, a vehicle also exchanges its coreset upon encounters to direct model exchange and merging and incorporates the received coresets into its local dataset. As the local model evolves and the local dataset expands over time, we denote the local model and local dataset of vehicle v_i at time t as $\boldsymbol{x}_i^t$ and D_i^t, respectively.

Our objective is to train local models on the *n* vehicles with their datasets, which can be formally defined by:

$$\min_{\boldsymbol{x}_i^T \in \mathbb{R}, i=1,...,n} \quad \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i^T; D_i^T) \tag{1}$$

where T is the maximum time for training. We expect that models on vehicles to be well trained finally and can work on the overall joint dataset. We note that, in practice, the models may not be exactly the same. We consider the objective as the weighted sum of loss over the local dataset:

$$f(\boldsymbol{x}; D) = \sum_{d \in D} w(d) f(\boldsymbol{x}; d), \tag{2}$$

where d is a single data sample and $w(d)$ is its original weight.

B. Coreset

Coreset is a compact subset approximating a substantially larger dataset, thereby reducing complexity measures such as time, space, and communication for running algorithms (e.g. clustering) on large datasets. Notably, coreset techniques have demonstrated success in addressing optimization problems (e.g. logistic regression and Gaussian mixture models [14]). In this work, our focus on coreset for continuous-and-bound (CnB) learning problems. We present the formal definition of continuous-and-bound learning problems below as [15].

Definition II.1 (Continuous-and-Bounded (CnB) Learning)**.** Let $\alpha, l > 0$, and $\tilde{\boldsymbol{x}} \in \mathcal{P}$. Denote by $\mathbb{B}(\tilde{\boldsymbol{x}}, l)$ the ball centered

[1] We leave heterogeneous hardware and communication capabilities to future investigation.

at $\tilde{\boldsymbol{x}}$ with radius l in the parameter space $\mathcal{P}$. An objective is called a Continuous-and-Bounded learning problem with the parameters $(\alpha, l, \tilde{\boldsymbol{x}})$ if the loss function $f(\cdot; d)$ is α-Lipschitz continuous for $\forall d \in D$, and $\boldsymbol{x}$ is restricted within $\mathbb{B}(\tilde{\boldsymbol{x}}, l)$.

A function $g : \mathcal{P} \to \mathbb{R}$ is α-Lipschitz continuous if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{P}$, $|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_2)| \leq \alpha \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$, where $\|\cdot\|$ is a specified norm in $\mathcal{P}$. We note that it is worth mentioning that within the context of CnB learning, various alternative variants can be considered by substituting the α-Lipschitz continuous assumption [15]. We formally define the coreset for CnB Learning problems as below:

Definition II.2 (ε-coreset)**.** Let $\varepsilon > 0$. Given a dataset $D \subset \mathcal{D}$ and the objective function $f(\boldsymbol{x}; D)$, we say that a weighted set $C \subset \mathcal{D}$ is an ε-coreset of D if $\forall \boldsymbol{x} \in \mathbb{B}(\tilde{\boldsymbol{x}}, l)$, we have:

$$|f(\boldsymbol{x}; C) - f(\boldsymbol{x}; D)| \leq \varepsilon f(\boldsymbol{x}; D). \tag{3}$$

ϵ is usually small so that $f(\boldsymbol{x}; C)$ is close to $f(\boldsymbol{x}; D)$. We denote the weight of a single data sample d within a coreset C as $w_C(d)$ if it is selected to be part of the coreset. We have the weighted sum of loss over the coreset as follows:

$$f(\boldsymbol{x}; C) = \sum_{d \in C} w_C(d) f(\boldsymbol{x}; d). \tag{4}$$

It is essential to clarify that $w_C(d)$ has a different meaning from $w(d)$, as the former represents the weight solely within the corresponding coreset, while the latter is the original weight of the data sample. If C is an ε-coreset of D, we can efficiently employ an existing optimization algorithm on C to obtain an approximate solution. In this work, an important goal in constructing the coreset is to ensure that the size of C remains sufficiently small, thus facilitating efficient computation and communication within constraints.

A significant portion of existing coreset construction methods are based on the theory of sensitivity proposed by [16]. Informally, sensitivity quantifies the importance of each data sample to the dataset D across all potential solutions. However, this type of method is data-dependent [15], in which the size of the coreset depends on the value of the total sensitivity of the dataset, and can vary for different inputs. Alternatively, partition-based coreset construction methods [14] [15] adopt a different strategy by partitioning the given dataset into several parts and then performing random sampling within each part, which yields a coreset of a data-independent size. Therefore, in this work, we focus on the partition-based method.

III. Method

In this section, we present LbChat in detail (Fig. 1). We begin by introducing how a vehicle determines the sequence of coreset and model exchange when encountering other vehicles in III-A. Then we present the coreset construction method in III-B. Following this, we describe how a vehicle adaptively compresses its model and aggregates models in III-C, all based on the exchanged coresets and the evaluation results on them. To further enhance the training, we show the local dataset expansion and the coreset updating in III-D, presenting the entire workflow of LbChat in the same subsection.

Table I
Frequent used Notations

Notation	Explanation
v_i	Vehicle i
B_i	The available bandwidth of v_i
$\boldsymbol{x}_i$	The model parameter of v_i
$f(\cdot;\cdot)$	The local loss function
D_i	The local dataset of v_i
C_i	The coreset of v_i
$w(d)$	The original weight a data sample d
$w_C(d)$	The weight of a data sample d in the corresponding coreset
φ_i	The compression ratio for $\boldsymbol{x}_i$
ψ_i	The reciprocal of φ_i

A. Sequence determination of exchange

In LbChat, when a vehicle encounters other vehicles, it determines the sequence of pairwise coreset and model exchange with them, prioritizing neighbors with a higher probability of exchange completion. By exchanging assistive information such as its location, speed, route (in next few minutes), and available bandwidth with the encountered peers, the vehicle can estimate the potential wireless communication losses, contact durations, and probabilities of model sharing completion. Following [7], we calculate the contact duration based communication priority $z_{i,j}$ by a truncated ratio and the probability of a successful model sending $p_{i,j}$ between vehicles v_i and v_j by using a distance-based wireless loss. A larger $z_{i,j}$ indicates a shorter yet sufficient contact duration between the two vehicles, and a smaller wireless loss results in a larger probability $p_{i,j}$. Considering the available bandwidth as well, we define a priority score for determining the exchange sequence as follows:

$$c_{i,j} = z_{i,j} p_{i,j} \min\{B_i, B_j\}, \tag{5}$$

where B_i and B_j are available bandwidth at v_i and v_j correspondingly. A vehicle prioritizes exchanging the coreset and model with peers of higher scores when there are multiple available neighbors. We note that the route and bandwidth information is of small size (e.g., 184 bytes in our experiments) and lightweight for transmission. Even with potential retransmissions, the time taken to share such assistive information can be neglected. As vehicles asynchronously determine the exchange sequence, there can be a small possibility of deadlocks, which can be addressed by setting a maximum waiting time or utilizing other existing approaches (e.g. [17]).

B. Coreset construction

Once the exchange sequence is determined, a vehicle commences the pairwise exchange and aggregation. However, the wireless bandwidth at vehicles is commonly limited (e.g. 31Mbps) [13] [18]. Even considering lightweight models (e.g.,

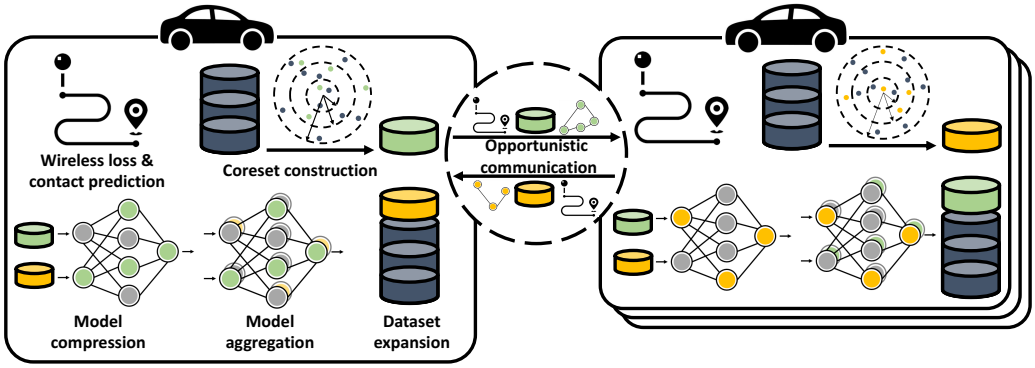


Figure 1. Overview of LbChat. Vehicles opportunistically "chat" with encountered peers, all while considering the constrained wireless communication in dynamic vehicular environments. Each vehicle maintains a compact and continuously updating coreset. Upon encountering other vehicles, a vehicle determines the sequence of pairwise exchange by estimating the probabilities of exchange completion. Then in each pairwise communication, vehicles exchange their respective coresets and perform evaluations on them to quantify the values of the other's local model to themselves. Subsequently, based on the evaluation results, vehicles tune the model compression ratios to maximize the joint aggregation benefits from the encounter under bandwidth and time constraints, and aggregate models locally. Finally, vehicles expand local datasets by absorbing received coresets, further enriching their training data.

an imitation learning model [19] of tens MB), exchanging models can take tens of seconds, easily surpassing the contact duration between two vehicles and impeding model exchanges with other encountered peers. Although existing compression techniques can reduce the model size, blind utilization of lossy compression can harm the model performance greatly [20].

To tackle this dilemma, our key idea is assessing the "value" of the encountered vehicle's model before exchange. More valuable models are less compressed for higher gain, and vice-versa. To this end, we use coreset to facilitate model value assessment. Upon encounter, two vehicles exchange coresets with each other, and measure the local model's performance on the peer's coreset. A lower performance than that of the peer's model indicates more different peer data, thus more valuable the peer model; and the larger the gap, the higher the value. We will present the model value assessment, compression, exchange, and following aggregation in III-C. The size and quality of coreset are critical to fast and reliable value assessment. We aim to construct a coreset that can approximate the local dataset closely while maintaining a minimal size for fast exchanging, striking a balance between efficacy and efficiency.

We present a concise and efficient coreset construction method (Algorithm 1) based on layered sampling [15], which constructs a compact mini-set from the original dataset satisfying the definition in II-B with theoretical guarantees. The method first partitions the local dataset into concentric layers (rings) based on the evaluation losses of data samples. The process begins by calculating the center of these layers, which corresponds to the smallest loss over the local dataset based on the current model. Subsequently, the local dataset is partitioned into a maximum of $log(|D|+1)$ layers, with each data sample being assigned to an appropriate layer based on its loss. Data samples with larger losses are placed in outer layers. After the partitioning, we take a small set from each layer and take the union of these sets to form the desired coreset. Different from [15] which assumes that data samples have equal weights and takes samples uniformly at random, we consider a more general scenario where data samples have different and continuously updated weights (presented in III-D). We perform weighted random sampling at each layer. We note that our coreset construction method can readily integrate existing efficient sampling techniques.

Algorithm 1 Layered sampling based coreset construction

Input: Weighted local dataset D; loss function $f(\cdot;\cdot)$;
Output: Coreset C;

Calculate the center: $f(\boldsymbol{x};\tilde{d}) = \min_{d\in D} f(\boldsymbol{x};d)$;
Calculate the radius of 0-th layer: $R = \frac{f(\boldsymbol{x};D)}{|D|}$;
for d in D **do**
  Calculate the distance from the center: $dist_d = \frac{f(\boldsymbol{x};d)-f(\boldsymbol{x};\tilde{d})}{R}$;
  Assign d to $\lfloor log(\frac{dist_d}{R}) \rfloor$-th layer;
end for
Let $L =$ total # of layers;
for $j = 0, 1, .., L$ **do**
  Let $\hat{D}_j$ be the set of data samples in the j-th layer;
  Take $\hat{C}_j \subset \hat{D}_j$ by $w(d)$-weighted random sampling;
  for $d \in \hat{C}_j$ **do**
    Assign the weight inside the coreset: $w_C(d) = \frac{\sum_{d'\in\hat{D}_j} w(d')}{\sum_{d'\in\hat{C}_j} w(d')}$;
  end for
end for
Construct coreset: $C = \bigcup_{j=0}^{L} \hat{C}_j$.

The method described in Algorithm 1 can achieve a ε-coreset of size $|C| = \Theta(\frac{log|D|}{\varepsilon^2}(ddim \cdot log(1/\varepsilon) + log(1/\eta)))$

in linear time with the probability of $1-\eta$, where $ddim$ is the doubling dimension of parameter space $\mathcal{P}$. And $|C|$ depends on the Lipschitz constant α and $\inf_{\boldsymbol{x}\in\mathbb{B}(\tilde{\boldsymbol{x}},l)}(1/|D|)f(\boldsymbol{x};D)$. However, $\inf_{\boldsymbol{x}\in\mathbb{B}(\tilde{\boldsymbol{x}},l)}(1/|D|)f(\boldsymbol{x};D)$ can be too small in practice, thereby necessitating a larger coreset to effectively represent the local dataset, which can result in additional bandwidth and time overhead in the vehicular communication and crowd out limited resources. To address this, we add penalty terms in the local loss function for constructing a compact coreset. We modify the loss function in Equations (2) and (4) as:

$$f(\boldsymbol{x};\xi)=\sum_{d\in\xi}w_{\xi}(d)f(\boldsymbol{x};d)+\lambda_1\|\boldsymbol{x}\|+\lambda_2\sigma(\boldsymbol{x}), \quad (6)$$

where $w_{\xi}(d)=w_C(d)$ if ξ is the corresponding coreset, otherwise $w_{\xi}(d)=w(d)$. λ_1 and λ_2 are coefficients of two penalty terms, and $\|\cdot\|$ is the L2-regularizer. The first two terms minimize the empirical risk and structural risk, correspondingly. Additionally, we introduce a problem-dependent penalty through the function $\sigma(\cdot)$. For the specific problem of a Bird-Eye-View (BEV)-based driving decision model [19], we define $\sigma(\boldsymbol{x})$ as the entropy of the losses observed with data samples of different driving commands (e.g., 'turning left'). This enables the model to effectively address all driving commands without introducing any bias. For different models or problems, $\sigma(\cdot)$ can be tailored accordingly, providing flexibility to various scenarios within the LbChat framework.

C. Adaptive model exchange and aggregation

To optimize model exchange and aggregation under constrained communication, vehicles exchange coresets with encountered peers and compress models based on the value assessment using the coresets. We emphasize that the size of our constructed coreset is substantially smaller than that of the vehicular learning model (e.g., BEV-based driving decision model). As a result, a small "investment" in exchanging coresets gives vehicles insights into which models are the more valuable, to effectively use the limited communication.

Given two vehicles v_i and v_j encounter each other, they exchange their current coresets C_i and C_j. Subsequently, they evaluate their local models $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ on both coresets and exchange the corresponding losses. Informally, the smaller v_j's model loss is on C_j than v_i's model, the more "valuable" v_j's model is to v_i. We define the model compression ratio as $\varphi=S/S_c$, where S is the size of the model before compression and S_c is the size after compression. We denote $\psi=1/\varphi$. $\psi=0$ indicates not sending model and $\psi=1$ indicates sending model without compression. We define the optimization problem for compression ratios as Equation (7) , where $\epsilon(\cdot)$ is the rectified linear unit function, ϕ_i and ϕ_j are mapping functions of v_i and v_j that predict the corresponding losses of compressed models in terms of ψ, λ_c is a penalty coefficient, T_B is the time budget for the pair-wise model exchange, $T_{contact}$ is the estimated contact duration between two vehicles, and $T_c=S(\psi_i+\psi_j)/\min\{B_i,B_j\}$ is the time for exchanging models between two vehicles.

$$\begin{aligned}\max_{\psi_i,\psi_j}\quad & \epsilon(\phi_i(f(\boldsymbol{x}_i;C_i),\psi_i)-f(\boldsymbol{x}_j;C_i))\\ & +\epsilon(\phi_j(f(\boldsymbol{x}_j;C_j),\psi_j)-f(\boldsymbol{x}_i;C_j))\\ & +\lambda_c(\min\{T_B,T_{contact}\}-T_c)\\ \text{subject to}\quad & T_c\le\min\{T_B,T_{contact}\},\\ & \psi_i,\psi_j\in[0,1],\end{aligned} \quad (7)$$

There are three terms in the objective of Equation (7). The first two terms measure a joint potential gain of exchanging models with fairness consideration. The first term calculates the truncated difference of weighted losses on coreset C_i between two models. The larger difference indicates a higher value that $\boldsymbol{x}_i$ holding for v_j, resulting in a greater potential gain that v_j can obtain by receiving v_i's model $\boldsymbol{x}_i$. We use the rectified linear unit function to ensure the difference is non-negative. Similarly, the second term calculates the truncated weighted loss difference on coreset C_j, thus the potential gain that v_i can obtain by receiving v_j's model $\boldsymbol{x}_j$. The third term is an award term, which measures the amount of time saved in exchanging models. By incorporating this term, two vehicles can promptly conclude the current exchange and decouple from each other when they are not interested in each other's model, and quickly move on to other encountered vehicles that might offer more valuable model exchanges.

We set T_B to a fixed time for simplicity, which can be set by other strategies (e.g. a dynamic number based on the number of neighbors). We use the mapping function ϕ to capture the relationship between model performance (represented by the weighted loss on the coreset) and the reciprocal of model compression ratio ψ. To obtain such mapping functions, a vehicle first samples a series of ψ's $\{0,\psi_0,..,\psi_k,1\}$. Then the vehicle generates a corresponding series of compressed models based on these compression ratios and evaluates their performance on the coreset. We represent the results as a set of pairs $\{(0,0),..,(\psi_k,f(\hat{\boldsymbol{x}}^{\psi_k};C)),(1,f(\boldsymbol{x};C))\}$, where $\hat{\boldsymbol{x}}^{\psi}$ denotes the compressed model under ψ. As $|C|$ is small, evaluating losses on them is computation-efficient. A vehicle exchanges the results with the encountered peer. With such results, we utilize an effective interpolation and curve fitting method [21] to generate the mapping function ϕ. With explicit mapping functions, we can solve the optimization problem defined in Equation (7) with existing solvers efficiently.

We notice a possible conflict of interest between vehicles in model exchange, as both vehicles prefer to spend limited bandwidth and time on receiving a model of high value to optimize their performance, rather than sending out their models. To address the conflict, we demand fairness between the two vehicles by simply adding the first two terms. This strikes a balance that encourages effective and mutually beneficial model sharing between vehicles. We also note that if there is a significant difference between the models of two vehicles, the representation ability of one vehicle's coreset may degrade

from the perspective of the other vehicle's model. In this case, the object in Equation (7) can also be utilized to guide model exchange and aggregation, as the performance diversity, measured by loss differences on the same sets of data samples, remains a valuable metric for model comparison.

After obtaining the optimized ψ values, a vehicle compresses its model accordingly and sends the compressed model to the target vehicle. Regarding the compression method, we consider top-k sparsification [22] in this work, in which the component's k-largest magnitudes in $\boldsymbol{x}$ are transmitted. When k is small, we can represent a compressed model by index-value pairs [23], further reducing the size of the model. We note that other biased/unbiased model compression methods can also be applied to our design, such as quantization. Upon receiving the compressed model $\hat{\boldsymbol{x}}_j$ from vehicle v_j, vehicle v_i aggregates it with its local model $\boldsymbol{x}_i$ based on the normalized losses on the joint coreset as below:

$$\bar{\boldsymbol{x}}_i = \frac{f(\boldsymbol{x}_i; D_i \bigcup C_j)\boldsymbol{x}_i + f(\hat{\boldsymbol{x}}_j; D_i \bigcup C_j)\hat{\boldsymbol{x}}_j}{f(\boldsymbol{x}_i; D_i \bigcup C_j) + f(\hat{\boldsymbol{x}}_j; D_i \bigcup C_j)}. \tag{8}$$

As $f(\boldsymbol{x_i}; D_i)$ has been computed during local training, caching these losses can further reduce repeated future computations. The equation assigns larger weights to better-performing models to adaptively aggregate them.

D. Local dataset expansion & Coreset updating

In the LbChat, each vehicle continuously receives coresets from encountered peers. A vehicle v_i expands its local dataset D_i by absorbing the received coreset C_j from vehicle v_j. We keep the original weights $w(d)$ of all data samples in the expanded local dataset $D_i \bigcup C_j$ to be the same.

Besides the straightforward coreset updating in Algorithm 1 running on the local dataset, we present a further improvement in the case of frequent encounters, which is appropriate to a rapidly expanding local dataset in the early stage. An important property of the ε-coreset is: If both C_1 and C_2 are respectively the ε-coresets of datasets D_1 and D_2, and $D_1 \bigcup D_2 = \emptyset$, their union $C_1 \bigcup C_2$ is an ε-coreset of $D_1 \bigcup D_2$ [15]. This property allows vehicle v_i to use $f(\boldsymbol{x}_i; C_i \bigcup C_j)$ to approximate $f(\boldsymbol{x}_i; D_i \bigcup C_j)$ if $D_i \bigcup C_j = \emptyset$, where the $w_C(d)$ of data samples in C_j need to be updated in this case. By caching and reusing the losses in Equation (8), such updating takes negligible computation. By doing so, a vehicle can update the coreset for the expanded local dataset by merging its coreset with the received coreset. However, a simple union of the coresets could lead to a significant increase in the coreset size. To address this issue, we utilize a 'reduce' operation [10] after each merging operation to keep the coreset size constant. We summarize LbChat in Algorithm 2.

Algorithm 2 LbChat

Input: $t = 0$; A set of vehicles $\mathcal{V} = \{v_i | i = 1, .., N\}$, each vehicle with a weighted local dataset D_i^0 and a model $\boldsymbol{x}_i^0$; the maximum time for training among vehicles T;
Output: Vehicular models $\{\boldsymbol{x}_i^T | i = 1, .., N\}$;

For each vehicle v_i (vehicles act in parallel):
while $t < T$ **do**
  Training local iterations with D_i^t;
  if Encountering other vehicle(s) **then**
    Exchange route, bandwidth, and other information;
    Determine the exchange sequence with (5);
    for v_j in the sequence **do**
      Construct coreset C_i^t with Algorithm 1;
      Send C_i^t and receive C_j^t;
      Evaluate $\boldsymbol{x}_i^t$ on both C_i^t and C_j^t;
      Compute the required results for (7);
      Exchange the results with v_j;
      Optimize the model compression ratio φ_i^t with (7);
      Send the compressed model $\hat{\boldsymbol{x}}_i^t$ and receive $\hat{\boldsymbol{x}}_j^t$;
      Aggregate models with (8);
      Expand D_i^t;
    end for
  end if
end while

IV. Experiments

A. Experimental setup

We focus on a BEV-based driving decision-making, which is a fundamental but crucial task in autonomous driving [1]. BEV is a high-level perception of the surrounding environment, which is a sparse binary tensor depicting the front view of a vehicle in a top-down view. For this task, we utilize an imitation learning model with the same structure as the privileged agent in [19], whose size is 52 MB before compression. The model takes a BEV and a high-level command (e.g. "turn left") from navigation services as inputs and outputs the next few waypoints the vehicle should follow.

We collect data with the popular CARLA simulator [24]. We utilize the largest built-in map supporting multiple expert autopilots in CARLA to simulate realistic driving environments, which covers an area of about 1km×1km, including both town and rural areas. We run 32 built-in *expert autopilot* vehicles as [25], which perform safe and professional driving using the built-in model and privileged information in CARLA following routes based on road topology. An additional 50 cars and 250 pedestrians are added into the simulated world as background traffic, to further enhance the realism of the environment. These cars and pedestrians are initialized at random locations and keep roaming on the map. For simplicity, we use "vehicle" to refer to *expert autopilot* only in the following unless otherwise stated. Vehicles collect data at two frames per second (fps) in the simulated world. Each frame of a vehicle contains the current BEV, the next high-level command, and the next few waypoints planned of the vehicle. We run the vehicles for one hour to collect the local datasets

for training. To simulate vehicle movements and encounters for a longer time, we run the vehicles for an additional 120 hours and collect their locations at the same fps.

We initialize vehicles in the simulated world at the same time, simulating inter-vehicle communications and performing collaborative model training. We utilize the local loss function based on [19] with the modification in Equation (6)[2]. We set each coreset as containing 150 data samples (frames) by default, at an approximate coreset size of 0.6MB with simple lossless compression. The hyper-parameters of all models are initiated to be identical. The learning rate is $1e^{-4}$, and the batch size is 64. We set the time budget T_B in Equation (7) to be 15 seconds. In communication simulation, we adopt the same parameters as [18] [26]. Specifically, we consider a packet size of 1500 bytes, a maximum bandwidth of 31Mbps, a maximum communication range is 500m, and up to three retransmissions per packet upon losses. In this setting, the time to transmit a coreset is less than 0.5 seconds, which accounts for only a small fraction of the contact duration between the two vehicles and is much shorter than the time of transmitting a model. To estimate the wireless loss, we use a distance-based wireless loss model [7], which utilizes a distance-loss lookup table based on [13]. We note that there are other ways to estimate the wireless loss in vehicular communications, which can be incorporated into LbChat seamlessly. The total size of route and bandwidth information required for calculating the score in Equation (5) is 184 bytes. We conduct experiments using an RTX2060 GPU. Given the recent advances in TFLOPS-level powerful onboard computation devices for vehicles [27], except for the local training time, we ignore time for computation and evaluation of Algorithm 2 in simulation.

B. Benchmarks

We empirically compare LbChat with a broad range of benchmarks of only model sharing as representatives, which are (adapted from) recent works in central server/roadside infrastructure-based decentralized learning and fully decentralized (a)synchronous learning for vehicles.

• ProxSkip [28] is a provably efficient federated learning approach based on central-server coordination, which offers an effective acceleration of communication compared to conventional methods (e.g. FedAvg). We assume no communication bandwidth constraint to the backend in ProxSkip, which is idealistic and non-practical in real environments.

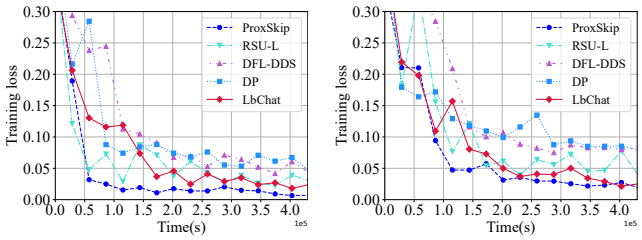
• RSU-L [29] proposes an RSU-based opportunistic learning approach for vehicles. Each RSU serves as a coordinator, maintaining an RSU model, receiving models from vehicles, and sending the aggregated model back. We simulate the behavior of RSUs at road crosses as [29]. We assume no backend bandwidth constraint at RSUs.

• DFL-DDS [30] is a synchronous and fully decentralized learning approach for vehicles, in which each vehicle tunes the aggregation weights to diversify the data sources contributing to its model. We set the time of one round to be the same as T_B of LbChat. The original work assumes a vehicle can finish model exchanging with all encountered peers. For a fair comparison, we consider the same communication ability and constraints as LbChat and compute a model compression ratio for each encounter to ensure the vehicle pair can finish the model exchange within the contact duration.

• Decentralized Powerloss (DP) [5] is a gossip learning approach for vehicles based on evaluations of loss-based model merging. A vehicle evaluates received models over the local validation dataset and derives weights in model aggregation from a normalized logarithmic function of the loss. Similarly, for a fair comparison, we also consider the same communication ability and constraints as LbChat and compute a model compression ratio for each encounter.

C. Illustration of training loss

We start by comparing LbChat with benchmarks, observing the training loss decreasing over time in the simulated world. The results in the case free from wireless loss are shown in Fig. 2(a). Within the given training time, we observe that LbChat converges to a similar loss as the central server-based federated learning approach (ProxSkip), and achieves almost the same loss as the RSU-based approach (RSU-L). When compared to fully decentralized benchmarks (DFL-DDS and DP), LbChat achieves a visible lower loss than them.



(a) Without wireless loss (b) With wireless loss

Figure 2. Results of training loss vs. time (LbChat & benchmarks)

We further study the case under wireless loss as described in IV-A. In ProxSkip and RSU-L, communications suffer from a wireless loss uniformly sampled from the distance-loss lookup table. For other benchmarks and LbChat, the wireless loss in communication is estimated based on the same lookup table according to the distance between vehicles. The results of the training loss over time in the simulated world are shown in Fig. 2(b). With the wireless loss, all approaches exhibit higher training losses while the loss increase in LbChat is relatively marginal compared to all benchmarks due to the neighbor prioritizing based on route-sharing. Notably, in this case, LbChat achieves almost the same loss as ProxSkip. The above results show that the proposed coreset-sharing based approach can converge like the central server/roadside infrastructure-based approaches when given adequate training time.

[2]We consider training converged when the differences among models are small enough.

Additionally, for deep insights in the case with wireless losses, we calculate and compare the successful model receiving rate on average. Unsurprisingly, we observe that due to prioritizing neighbors of higher chances of completing model exchange, LbChat has a significantly higher rate (87%) than other benchmarks (60% in ProxSkip, 60% in RSU-L, 52% in DFL-DDS and 51% in DP). The results partly explain the observations in Fig. 2(b) and demonstrate that LbChat achieves better robustness to wireless loss based on estimation of the communication completion probabilities.

D. Online evaluation

We further explore the potential of the training data-sharing assisted paradigm and compare LbChat with benchmarks by conducting online evaluations, in which we deploy the trained model on a testing autopilot to navigate through predefined navigation routes. Based on a safety-centric consideration, we utilize driving success rate, a widely used metric in autonomous driving [1] [19], to measure the model performance, which indicates the ability of a model to drive a car to destinations safely. We note that other metrics for evaluating a driving model (e.g., comfort measurements) can be incorporated into our future work. We consider driving conditions of different difficulties similar to the CARLA benchmark [24], which includes driving straight (Straight), driving with one turn (One Turn), full navigation with multiple turns (Navi. (Empty)), and the same full navigation routes but with traffic (Navi. (Normal)). To further evaluate approaches under a more challenging driving environment, we also test models with a harder condition called Navi. (Dense), where the number of roaming cars and pedestrians is 1.2× that of Navi. (Normal). We consider a trial on a given route successful if the testing autopilot can safely reach the destination within a budget time without colliding with other cars or pedestrians.

Table II
DRIVING SUCCESS RATE ON AVERAGE (W/O WIRELESS LOSS) (%)

Task	ProxSkip	RSU-L	DFL-DDS	DP	LbChat
Straight	100	100	100	100	100
One Turn	100	100	100	100	100
Navi. (Empty)	99	97	90	89	97
Navi. (Normal)	94	89	81	80	90
Navi. (Dense)	83	77	67	65	78

Table II summarizes the results in the idealistic but non-practical case of no wireless loss. In this case, LbChat achieves competitive results as the ProxSkip with at most 5% lower driving success rate among all driving conditions. When compared to RSU-L, LbChat shows almost the same performance in general and outperforms RSU-L in the hardest driving condition. Additionally, LbChat notably outperforms DFL-DDS and DP under the same communication constraints in all driving conditions and with up to 11% higher driving success rate than DFL-DDS and up to 13% higher rate than DP, both in the hardest condition.

Table III
DRIVING SUCCESS RATE ON AVERAGE (W WIRELESS LOSS) (%)

Task	ProxSkip	RSU-L	DFL-DDS	DP	LbChat
Straight	100	100	100	100	100
One Turn	100	99	96	95	100
Navi. (Empty)	93	89	79	78	93
Navi. (Normal)	87	83	72	70	86
Navi. (Dense)	75	69	57	54	74

Table III summarizes the results in the case with practical wireless loss. By comparing the results to Table II, LbChat shows robustness towards the loss in communication with a limited performance decrease (at most 4% decrease), whereas benchmarks in comparison commonly show much larger performance decreases (e.g. 11% decrease of DP in Navi. (Dense)). We also observe that LbChat achieves almost the same performance as ProxSkip with at most a 1% driving success rate decrease in all driving conditions and outperforms RSU-L by up to 5% higher driving success rate. Moreover, when compared to DFL-DDS and DP, LbChat significantly outperforms them in all driving conditions by up to a notable 20% higher driving success rate in the hardest one.

We summarize the observations in online evaluations: 1) LbChat is quite competitive to central server-based federated learning, especially under constrained communication with wireless loss. 2) LbChat can achieve almost the same performance as the RSU-based approach and even better performance when considering wireless loss. 3) Compared to decentralized/gossip learning approaches, LbChat outperforms the benchmarks significantly and shows superiority in addressing multiple real constraints (limited contact duration and bandwidth, and wireless loss) for on-road vehicles.

These results further demonstrate the effectiveness of LbChat, which does not require special coordinators, tight synchronizations, or even connected topology, making it a promising approach for the real world. The results also highlight the effectiveness of the coreset-sharing based approach and the potential of training data sharing assisted paradigm for collaborative model training among vehicles.

E. Size of the coreset

In communication-limited vehicular scenarios, the size of a coreset can affect the performance of our approach. A large coreset can represent the local dataset well but a small coreset is communication-efficient. Therefore, we study the influence of the size of coreset on model performance by considering two additional coreset sizes (1500 and 15), which correspond to 10x and 1/10 the size of our default size. Table IV summarizes the results. In two cases with(out) wireless loss, we both observe that the two sizes of coreset hurt the performance of LbChat by at most 6% and 8% driving success rate decrease, respectively. We observe that an improper size of coreset can visibly degrade the approach performance. The

results are consistent with the intuition: a larger coreset can be representative but may consume limited contact duration and impede model exchange. In contrast, a smaller coreset can save communication resources but may fail to adequately represent the diverse characteristics of the dataset. Adaptive tuning the size of coreset will be our future work.

Table IV
DRIVING SUCCESS RATE ON AVERAGE WITH DIFFERENT CORESET SIZE(%)

Task	1500 (W/O)	15 (W/O)	1500 (W)	15 (W)
Straight	100	100	100	100
One Turn	100	100	100	99
Navi. (Empty)	91	89	87	85
Navi. (Normal)	84	82	80	78
Navi. (Dense)	72	70	68	66

F. Ablations

To quantitatively measure the effects of coreset-sharing in a more fine-grained manner, we conduct two ablations to LbChat by replacing the coreset-based designs with straightforward methods. Firstly, we mask the coreset-based optimization to compression ratio in Equation (7), and vehicles use equal compression ratios in model exchange instead. Table V summarizes the results. We observe such fixed compression results in at most 7% and 8% decrease of driving success rate in two cases with(out) wireless loss, correspondingly. This is because, without the coreset-based adaptive tuning to model compression ratio, a vehicle can spend more time receiving less-compressed and less valuable models, and vice versa. Neither is efficient to utilize the limited communication opportunities to maximize the gain from each encounter.

Table V
DRIVING SUCCESS RATE ON AVERAGE WITH EQUAL COMP. RATIO (%)

Task	W/O wireless loss	W wireless loss
Straight	100	100
One Turn	100	100
Navi. (Empty)	91	85
Navi. (Normal)	83	78
Navi. (Dense)	71	66

We then study using averaging to replace the coreset-based model aggregation described in Equation (8), summarizing the results in Table VI. We observe at most a 4% decrease in driving success rates. The reason is that, by masking the coreset-based weight derivation, vehicles lose the ability to distinguish models of different values in merging. As a result, vehicles fail to prevent less-valued (i.e., poorly performing) models from negatively impacting the merged model.

G. Study of sharing coreset only

In the previous comparisons with benchmarks of pure model sharing, LbChat shows strong competitiveness and effectiveness benefiting from coreset-sharing. We compare LbChat with an additional approach named SCO, in which vehicles only share coresets with encountered peers but not exchanging and merging models, while subjecting to the same coreset size, contact duration, and communication constraints. Table VII summarizes the online evaluation results of SCO. We observe that it can achieve almost the same performance as LbChat with at most 1% driving success rate decrease without wireless loss and at most 2% decrease in the wireless loss-free case.

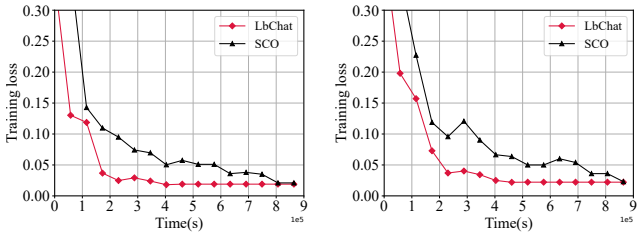
Table VI
DRIVING SUCCESS RATE ON AVERAGE WITH AVG. AGGREGATION (%)

Task	W/O wireless loss	W wireless loss
Straight	100	100
One Turn	100	98
Navi. (Empty)	95	90
Navi. (Normal)	88	83
Navi. (Dense)	75	70

We further study the training speed of the two approaches in terms of time (shown in Fig. 3). We observe that, though SCO can achieve similar training loss as LbChat, it takes about 1.5×-1.8× longer time to converge. The results suggest that: though vehicles can acquire models of almost the same performance finally by sharing coresets and expanding the training dataset only, sharing both the coreset and model can greatly accelerate the convergence. This is because by merging valuable models, a vehicle can immediately acquire knowledge from the local datasets of others with fewer local iterations, thus accelerating the model training.

Table VII
DRIVING SUCCESS RATE ON AVERAGE WITH SHARING CORESET ONLY (%)

Task	W/O wireless loss	W wireless loss
Straight	100	100
One Turn	100	100
Navi. (Empty)	96	92
Navi. (Normal)	89	84
Navi. (Dense)	77	72



(a) Without wireless loss (b) With wireless loss

Figure 3. Results of training loss vs. time (LbChat & SCO)

V. DISCUSSION

Alternative coreset construction approaches. There are other kinds of coreset construction strategies (e.g., random

sampling based [16] and clustering based algorithms [31]). Since our main idea involves quantifying model values using the loss differences on the same sets of data samples, other coreset construction approaches can be adapted in LbChat. This flexibility enables LbChat to handle diverse datasets and further enhance its capability for fully decentralized and asynchronous vehicular model training.

Model value assessment before vs. after exchange. Many decentralized learning works evaluate models on the local dataset for various purposes after the exchange, such as generating weights in model aggregation [5]. While LbChat also has adaptations, it evaluates before exchanging the model. Because the communication bandwidth and contact duration among moving vehicles are very limited, indiscriminately receiving models from encountered peers may have low chances of completion and become wasteful; yet blindly utilizing lossy compression can significantly harm model aggregation. Exchanging compact coresets allow vehicles to first evaluate model "value", and optimize the following model exchange with adaptive compression under communication constraints.

Accuracy of simulation results. Conducting real-world experiments with real vehicles, particularly on a medium or large scale requires significant resources. In the experiments, we use CARLA, a widely-used simulator in the field of autonomous driving that has been extensively used in both academic and industrial research. CARLA simulates driving environments with a high degree of realism, making it a reliable means for empirical studies. Previous research (e.g., [32]) demonstrates that when appropriately configured, results from the CARLA simulator are nearly identical to those obtained from real-world experiments. Therefore, the results from simulation are reasonably close to those from real-world experiments. We note that the coreset-sharing based model training paradigm proposed in this work can also be applied to a spectrum of tasks and models. We are interested in showing its effectiveness on different tasks and models in the future.

Other radios suitable for vehicles. New Radio V2X (NR-V2X) [33] is a promising vehicular communication technology, improving over predecessors under compatibility considerations. Recent data-centric radios have also shown high-rate, low-loss, and low-latency multicast capability [34], ideal for vehicles to share models. Some short-range radios provide accurate motion and health-related event detection [35] [36], offering potential for in-vehicle applications.

Incentive and privacy. Recent works [11] [37] discuss the incentives or markets to stimulate cooperation among vehicles while striking a balance between the conflicting interests of the platform and drivers. Exactly what incentive design is most suitable for the scenario needs orthogonal investigation. Moreover, besides training from scratch, we point out that LbChat holds the potential of continuously updating/fine-tuning pre-trained onboard models with locally collected data. We leave this study to our future work. LbChat is also privacy-friendly: it shares the BEV abstracted from but not photos or point clouds directly. As mentioned in IV-A, such BEV only contains what can be observed from outside a vehicle anyway, and without specific personal details, such as the face. Existing privacy-protection techniques for sharing data among vehicles (e.g. [12] [38] [39]) can also be adapted to our approach.

VI. Related Work

Many works provide decent proof/theoretical analysis of asynchronous decentralized learning. For example, Lian et al. proposed an asynchronous decentralized SGD algorithm for a heterogeneous environment [40], assuming bounded straggler delay and connected topology. Wang et al. studied decentralized parallel SGD and proposed a unified framework as well as convergence analysis [41], also based on assumptions about connected topology and explicit delay bound among worker nodes. Although there are tons of existing works in this area, they assume connected communication topology and other factors (e.g., a maximum delay of stragglers), which are hard to control or simply infeasible in a realistic vehicular scenario. Different from the above works, LbChat uses coresets to evaluate model values to guide model exchange and merge for fast-moving vehicles, thus eliminating assumptions on bounded delay or connected cluster/topology.

Focusing on the communication constraints in practice, recent research proposes various approaches to improve communication efficiency. For instance, Wang et al. proposed algorithms to jointly tune control knobs in federated learning to minimize cost and optimize performance [23]. In [42], Han et al. proposed an online algorithm to adaptively sparsify the gradients in federated learning. Perazzone et al. proposed a client selection algorithm for federated learning to minimize the average communication time [43]. Jiang et al. proposed a client selection and gradient compression method to alleviate the communication load and mitigate the straggler effect [44]. Though the above works study efficiency, they usually require a global coordinator. In contrast, we emphasize that our work does not require any central/edge coordinator, multi-hop coordination, or tight synchronization.

VII. Conclusion

We explore a training data sharing based paradigm for collaborative model training among peer vehicles. We propose LbChat, a coreset-sharing based decentralized and asynchronous model training approach, without reliance on coordinators, tight synchronizations, or connected topology among vehicles. The key idea is that the small "investment" of exchanging coresets and evaluating models on them allow vehicles to identify the most valuable peer models, thus optimizing model exchange and aggregation to maximize the aggregate gain. Empirical results demonstrate that LbChat is strongly competitive with the central server-based federated learning and roadside infrastructure-based benchmarks, and significantly outperforms recent fully decentralized vehicular learning benchmarks subject to practical constraints.

REFERENCES

[1] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *ICCV 2019*, 2019, pp. 9329–9338.

[2] K. Kim, S. Cho, and W. Chung, "Hd map update for autonomous driving with crowdsourced data," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1895–1901, 2021.

[3] Z. Zhang, S. Wang, Y. Hong, L. Zhou, and Q. Hao, "Distributed dynamic map fusion via federated learning for intelligent networked vehicles," in *ICRA 2021*. IEEE, 2021, pp. 953–959.

[4] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2020.

[5] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "A gossip learning approach to urban trajectory nowcasting for anticipatory ran managemen," *IEEE Transactions on Mobile Computing*, 2023.

[6] G. Di Giacomo, J. Härri, and C. F. Chiasserini, "Edge-assisted gossiping learning: Leveraging v2v communications between connected vehicles," in *ITSC 2022*. IEEE, 2022, pp. 3920–3927.

[7] H. Zheng, M. Liu, F. Ye, and Y. Yang, "Roadtrain: Route-assisted decentralized peer model training among connected vehicles," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023.

[8] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 344–353.

[9] G. Nadiradze, A. Sabour, P. Davies, S. Li, and D. Alistarh, "Asynchronous decentralized sgd with quantized and local updates," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6829–6842, 2021.

[10] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 291–300.

[11] C. Xiang, Y. Li, Y. Zhou, S. He, Y. Qu, Z. Li, L. Gong, and C. Chen, "A comparative approach to resurrecting the market of mod vehicular crowdsensing," in *Proc. IEEE Conf. Comput. Commun*, 2022, pp. 1–10.

[12] X. Hu, R. Li, Y. Ning, K. Ota, and L. Wang, "A data sharing scheme based on federated learning in iov," *IEEE Transactions on Vehicular Technology*, 2023.

[13] W. Anwar, N. Franchi, and G. Fettweis, "Physical layer evaluation of v2x communications technologies: 5g nr-v2x, lte-v2x, ieee 802.11 bd, and ieee 802.11 p," in *VTC2019-Fall*. IEEE, 2019, pp. 1–7.

[14] H. Ding and Z. Wang, "Layered sampling for robust optimization problems," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2556–2566.

[15] Z. Wang, Y. Guo, and H. Ding, "Robust and fully-dynamic coreset for continuous-and-bounded learning (with outliers) problems," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[16] M. Langberg and L. J. Schulman, "Universal ε-approximators for integrals," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2010, pp. 598–607.

[17] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. pearson education, 2005.

[18] "Ieee p802.11 - next generation v2x study group." [Online]. Available: https://www.ieee802.org/11/Reports/tgbd_update.htm

[19] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *ICRA 2020*, 2020.

[20] A. M Abdelmoniem, A. Elzanaty, M.-S. Alouini, and M. Canini, "An efficient statistical-based gradient compression technique for distributed training systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 297–322, 2021.

[21] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures," *Journal of the ACM (JACM)*, vol. 17, no. 4, pp. 589–602, 1970.

[22] A. Albasyoni, M. Safaryan, L. Condat, and P. Richtárik, "Optimal gradient compression for distributed and federated learning," *arXiv preprint arXiv:2010.03246*, 2020.

[23] S. Wang, J. Perazzone, M. Ji, and K. Chan, "Federated learning with flexible control," in *IEEE Conference on Computer Communications*, 2023.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *CoRL 2017*. PMLR, 2017.

[25] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3936–3947, 2016.

[26] S. Zeadally, M. A. Javed, and E. B. Hamida, "Vehicular communications for its: Standardization and challenges," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 11–17, 2020.

[27] "Tesla's new hw3 self-driving computer — it's a beast (cleantechnica deep dive)."

[28] K. Mishchenko, G. Malinovsky, S. Stich, and P. Richtárik, "Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally!" in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 750–15 769.

[29] W. Xu, H. Wang, Z. Lu, C. Hua, N. Cheng, and S. Guo, "Mobile collaborative learning over opportunistic internet of vehicles," *IEEE Transactions on Mobile Computing*, 2023.

[30] D. Su, Y. Zhou, and L. Cui, "Boost decentralized federated learning in vehicular networks by diversifying data sources," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*. IEEE, 2022, pp. 1–11.

[31] H. Lu, M.-J. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan, "Robust coreset construction for distributed machine learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, 2020.

[32] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang, "Vips: real-time perception fusion for infrastructure-assisted autonomous driving," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022, pp. 133–146.

[33] "3gpp: Etsi work programme report." [Online]. Available: http://www.3gpp.org/release-16

[34] M. Elbadry, F. Ye, P. Milder, and Y. Yang, "Pub/sub in the air: A novel data-centric radio supporting robust multicast in edge environments," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2020, pp. 257–270.

[35] Z. Xie, H. Wang, S. Han, E. Schoenfeld, and F. Ye, "Deepvs: A deep learning approach for rf-based vital signs sensing," in *Proceedings of the 13th ACM international conference on bioinformatics, computational biology and health informatics*, 2022, pp. 1–5.

[36] Z. Xie, B. Zhou, X. Cheng, E. Schoenfeld, and F. Ye, "Passive and context-aware in-home vital signs monitoring using co-located uwb-depth sensor fusion," *ACM transactions on computing for healthcare*, vol. 3, no. 4, pp. 1–31, 2022.

[37] R. Ding, Z. Yang, Y. Wei, H. Jin, and X. Wang, "Multi-agent reinforcement learning for urban crowd sensing with for-hire vehicles," in *INFOCOM 2021*. IEEE, 2021, pp. 1–10.

[38] U. Javaid, M. N. Aman, and B. Sikdar, "Drivman: Driving trust management and data sharing in vanets with blockchain and smart contracts," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. IEEE, 2019, pp. 1–5.

[39] Z. Wu, H. Zheng, L. Zhang, and X.-Y. Li, "Privacy-friendly blockchain based data trading and tracking," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*. IEEE, 2019, pp. 240–244.

[40] 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.

[41] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 9709–9758, 2021.

[42] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *2020 IEEE 40th international conference on distributed computing systems (ICDCS)*. IEEE, 2020, pp. 300–310.

[43] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1449–1458.

[44] Z. Jiang, Y. Xu, H. Xu, Z. Wang, and C. Qian, "Adaptive control of client selection and gradient compression for efficient federated learning," *arXiv preprint arXiv:2212.09483*, 2022.