arXiv:2306.16169v1 [cs.RO] 28 Jun 2023

Communication Resources Constrained Hierarchical Federated Learning for End-to-End Autonomous Driving

Wei-Bin Kou^{1,3,4}, Shuai Wang^{2,3,*}, Guangxu Zhu⁴, Bin Luo⁵, Yingxian Chen¹, Derrick Wing Kwan Ng⁶, and Yik-Chung Wu^{1,*}

Abstract-While federated learning (FL) improves the generalization of end-to-end autonomous driving by model aggregation, the conventional single-hop FL (SFL) suffers from slow convergence rate due to long-range communications among vehicles and cloud server. Hierarchical federated learning (HFL) overcomes such drawbacks via introduction of mid-point edge servers. However, the orchestration between constrained communication resources and HFL performance becomes an urgent problem. This paper proposes an optimization-based **Communication Resource Constrained Hierarchical Federated** Learning (CRCHFL) framework to minimize the generalization error of the autonomous driving model using hybrid data and model aggregation. The effectiveness of the proposed CRCHFL is evaluated in the Car Learning to Act (CARLA) simulation platform. Results show that the proposed CRCHFL both accelerates the convergence rate and enhances the generalization of federated learning autonomous driving model. Moreover, under the same communication resource budget, it outperforms the HFL by 10.33% and the SFL by 12.44%.

I. INTRODUCTION

Vision-based autonomous driving vehicles, such as Tesla, are increasingly prevalent in our life, but the generalization needs to be continuously enhanced because the embedded autonomous driving model cannot generalize to all scenarios. Federated learning (FL) is an emerging paradigm to overcome the domain shifting issue to enhance generalization [1] in end-to-end autonomous driving (FLEAD). Despite the fact that many efforts have been invested in developing FLEAD techniques [2]–[10], a number of technical challenges still need to be properly handled: I) **Limited communication resources**. Current FLEAD designs aim to maximize the driving performance without any communication constraints,

This work has been accepted by 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). This work was supported in part by the Science and Technology Development Fund of Macao S.A.R (FDCT) (No. 0081/2022/A2), in part by Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) (No. GML-KF-22-17), in part by the SIAT Direct Drive Tech Cooperation Project, in part by National Natural Science Foundation of China (No. 62001310), in part by Guangdong Basic and Applied Basic Research Foundation (No. 2022A1515010109) and in part by the Internal Project Fund from Shenzhen Research Institute of Big Data (No. J00120230001).

*Corresponding author: Shuai Wang (s.wang@siat.ac.cn) and Yik-Chung Wu (ycwu@eee.hku.hk).

¹Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong 999077, China.

²Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China.

³Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.

⁴Shenzhen Research Institute of Big Data, Shenzhen, China.

⁵Damo Institute, Alibaba, China.

⁶School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia

thus ignoring the interdependency between driving performance and practical communication conditions. II) **Tradeoff between learning performance and communication cost**. How can we effectively allocate the network resources to different vehicles, edge servers and cloud server to maximize the FL model performance while satisfying the stringent communication constraints?

To tackle these challenges, currently there are three methods mainly to compress model parameters or gradient size: quantization-based, sparsification-based, and distillation-based strategies [11]. Specifically, quantizationbased methods [12] try to quantize continuous model parameters or gradient into a discrete set to reduce the bits. For sparsification, it converts parameters or gradient to a sparse one or zero according to importance of corresponding elements. Top-k and rand-k sparsification are two widely utilized approaches [13]. Distillation [14] is proposed to transfer a large model (teacher) to a small model (student) without obvious performance loss. In summary, these mainstream methods to mitigate the FL communication overheads is by reducing the model parameters or gradient size, leading to some performance loss [15] compared to the case without any compression.

In this paper, we propose to maximize the hierarchical federated learning (HFL) model performance under constrained communication resources from a completely different perspective: optimization-based communication resource scheduling under limited budgets. To the best of our knowledge, this is the first work to integrate constrained throughput scheduling and HFL in FLEAD. To begin with, we propose an optimization algorithm to distinguish the priorities of different HFL stages and perform resource scheduling with consideration of both data and model transfer. This algorithm supports both sample size and FL round planning. In addition, based on this optimization algorithm, we elaborate communication resources constrained HFL framework (CRCHFL) to enhance convergence and generalization by leveraging both data and model parameter aggregation. Specifically, it consists of: I). cloud pretraining stage to collect data from vehicles to cloud server for centralized pretraining and then to release pretrained model to all edge servers and vehicles to accelerate FL convergence; II). edge federated learning stage to aggregate model parameters of all associated vehicles; and III). cloud federated learning stage to aggregate model parameters of all edge servers. In particular, our main contributions are:

(1) We propose CRCHFL framework to maximize the

model performance under constrained communication resources.

- (2) Evaluation and analysis of our proposed approach on simulation dataset.
- (3) Implementing the proposed CRCHFL scheme located at (https://github.com/WeibinKOU/CRCHFL.git) based on the high-fidelity CARLA [16] simulator.

II. RELATED WORK

Current autonomous driving (AD) systems can be divided into two categories: modular-based [17], [18] and learningbased [4]–[9], [19]–[23]. Modular-based methods result in error propagation and inaccuracies in both problem modelling and solving stages. End-to-end learning-based approaches can address this error propagation issue. These methods generally map the onboard sensor data, such as LiDAR point clouds and camera images, into driving actions, such as throttle, brake and steer directly. However, such learningbased methods would have generalization issue intrinsically and work only in limited scenarios.

FL is an emerging paradigm to improve the generalization of end-to-end learning-based methods via model parameter aggregation or data collaboration [9]. In the context of EAD, FL leverages vehicular networks to integrate knowledge from different vehicles located in various scenarios. As such, when an EAD system enters a new scenario, it can convey the knowledge of new samples or corner cases to other vehicles and remote servers while preserving data privacy [4]–[9], [22]. For example, in [22], a cloud federated robotic system is proposed to enhance the behaviour cloning method, generating accurate control commands based on RGB images, depth images and semantic segmentation images.

Nonetheless, such single-hop federated learning (SFL) converges slow owing to its long communication latency [24]. HFL can accelerate the training procedure since edge servers are closer to vehicles and more communicationefficient than cloud server. On the other hand, for HFL, there exists frequent communication flows among different nodes, such as vehicles, edge servers, and cloud server. Such communications among nodes in [4], [6]–[9], [22] are assumed to be perfect, which does not hold for practical EAD systems with limited resources. The mainstream approaches address the constrained communication resource problem of HFL in practical situations by compressing the model parameter or gradient size. There are now three main approaches: quantization-based method (e.g., [25]), sparsification-based method (e.g., [26]), and distillation-based method (e.g., [27]). It is worth noting that most of these methods come at the expense of a certain amount of performance. Therefore, it becomes imperative to develop associated methods to allocate the limited network resources for HFL to minimize the generalization error of deep neural networks (DNNs) without any performance loss. This inspires us to design a CRCHFL framework, which fuses both data and model parameters and optimizes limited communication resources, to maximize the FLEAD convergence and generalization performance.



Fig. 1: Illustration of a cloud-edge-vehicle system. Red bars represent wireless flows. Blue bars represent wireline flows. The size of each bar represents the communication throughput of its associated link.

III. SYSTEM FRAMEWORK

We consider a cloud-edge-vehicle system shown in Fig. 1. The proposed CRCHFL framework is shown in Fig. 2, which consists of an end-to-end imitation learning pipeline (i.e., top of Fig. 2), a three-stage training procedure (i.e., middle of Fig. 2) and an optimization-based communication resource scheduler (i.e., bottom of Fig. 2). Specifically, for the *k*-th vehicle $(1 \le k \le K_n)$ in the *n*-th town $(1 \le n \le N)$, denoted vehicle (n,k), its DNN model $f_{n,k}(\cdot|\mathbf{m}_{n,k})$ with parameter vector $\mathbf{m}_{n,k}$ is an inference mapping from a frame of sensor data (e.g., images) $\mathbf{s}_{n,k}$ to a driving actions (i.e., throttle, steer and brake) $\mathbf{a}_{n,k}$, i.e., $\mathbf{a}_{n,k} = f_{n,k}(\mathbf{s}_{n,k}|\mathbf{m}_{n,k})$. To optimize model parameters $\mathbf{m}_{n,k}$, we need to define loss function $\mathscr{L}(\mathbf{m}_{n,k}, \mathbf{s}_{n,k}, \mathbf{a}_{n,k})$, and the ego-vehicle training is given by

$$\min_{\mathbf{m}_{n,k}} \mathscr{L}_{n,k} = \frac{1}{|\mathscr{T}_{n,k}|} \sum_{\left(\mathbf{s}_{n,k}^{(i)}, \mathbf{a}_{n,k}^{(i)}\right) \in \mathscr{T}_{n,k}} \mathscr{L}\left(\mathbf{m}_{n,k}, \mathbf{s}_{n,k}^{(i)}, \mathbf{a}_{n,k}^{(i)}\right) \quad (1)$$

where $\mathcal{T}_{n,k}$ is the training dataset at vehicle (n,k).

Since the generalization of $f_{n,k}$ increases with the size of dataset $\mathcal{T}_{n,k}$, it is necessary to exploit $\mathcal{T}_{n,k}$ for all (n,k). However, directly aggregating all datasets would lead to high communication costs and data privacy issues. To this end, a three-stage training procedure is proposed, which can be divided into cloud pretraining (i.e., stage I), edge FL (i.e., stage II), and cloud FL (i.e., stage III).

In Stage I, vehicles upload data samples to the cloud server. The cloud server uses collected data to train an initial model, which is then released to all the vehicles and edge servers for subsequent FL. The stage I training is given by

$$\min_{\boldsymbol{m}_{cloud,P}} \sum_{n=1}^{N} \sum_{k=1}^{K_n} \frac{1}{|\mathcal{T}_{n,k}^{I}|} \sum_{(\mathbf{s}_{n,k}^{(i)}, \mathbf{a}_{n,k}^{(i)}) \in \mathcal{T}_{n,k}^{I}} \mathscr{L}\left(\mathbf{m}_{cloud,P}, \mathbf{s}_{n,k}^{(i)}, \mathbf{a}_{n,k}^{(i)}\right)$$
(2)

where $\mathscr{T}_{n,k}^{I}$ is the uploaded dataset from vehicle (n,k) to the cloud server. The number of samples $\sum |\mathscr{T}_{n,k}^{I}|$ is denoted as *pretrain_batch*, which depends on the associated communication resources allocated to Stage I.

In Stage II, each edge server collects models from associated vehicles and aggregates them according to FedAvg [28], and then delivers the aggregated model to all associated vehicles. The stage II training is given by

Vehicle:
$$\min_{\mathbf{m}} \mathscr{L}_{n,k}, \forall k, n,$$
 (3)



Fig. 2: The structure of the proposed CRCHFL. The first layer (top layer) is overview of imitation learning pipeline, including input sensor data, imitation learning DNN model and predicted output driving actions. Second layer showcases the CRCHFL-involved nodes, i.e., autonomous vehicles, edge servers and cloud server. The third layer is about the communication of CRCHFL, including data and model upload (UL) and download (DL). The fourth layer (bottom layer) focuses on optimization algorithm to schedule communication resources.

Edge:
$$\min_{\mathbf{m}_{n,1}=\cdots=\mathbf{m}_{n,K_n}} \mathscr{L}_{edge,n} = \sum_{k=1}^{K_n} \frac{|\mathscr{T}_{n,k}|}{\sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|} \mathscr{L}_{n,k} \quad (4)$$

Equations (3) and (4) are executed iteratively and each vehicle local training iterations between two adjacent edge aggregations is denoted as *edge_interval*, which determines the associated communication resources allocated to Stage II.

In Stage III, cloud server collects models from edge servers to aggregate together also using FedAvg for producing global model. Then the global model is released to all edge servers and vehicles. The stage III training is given by

$$\min_{\mathbf{m}_{edge,1}=\cdots=\mathbf{m}_{edge,N}} \mathscr{L}_{cloud} = \sum_{n=1}^{N} \frac{\sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|}{\sum_{n=1}^{N} \sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|} \mathscr{L}_{edge,n} \quad (5)$$

The Stages III and II are executed iteratively and the edge aggregation times of each edge server between two adjacent cloud aggregations is denoted as *cloud_rounds*, which determines the associated communication resources allocated to Stage III.

Our proposed CRCHFL framework aims to minimize the generalization error by setting four hyperparameters: *edge_interval*, *cloud_interval*, *pretrain_batch* and *cloud_rounds* which determines the limited communication resource distribution, are needed to optimize by an optimization algorithm. The entire procedure of CRCHFL is summairzed in Algorithm 1. The next section will present details of the resource optimization algorithm.

IV. RESOURCE OPTIMIZATION FOR CRCHFL

This section presents how to model the resource optimization algorithm for the proposed CRCHFL. This algorithm is designed to schedule communication resources to transfer model parameters $\mathbf{m}_{edge,n}$, \mathbf{m}_{cloud} , and $\mathbf{m}_{n,k}$ in stage II and III, as well as pretraining data in stage I by optimizing *edge_interval*, *cloud_interval*, *pretrain_batch* and *cloud_rounds*.

Let the vector $\mathbf{u} = [u_1, u_2]^T$ represents the communication throughput allocation for data transfer and model transfer. Vector **u** satisfies $u_1 + u_2 \leq U_{sum}$, where U_{sum} (Unit: **GB**) is the total throughput budget. The number of samples and federated learning rounds are determined by their associated communication resources and there exists a tradeoff among stages. For data transfer, data samples should be uploaded from all vehicles to the cloud server. The number of samples x is constrained by communication throughput as $x \leq \frac{u_1}{D}$, where D (Unit: MB) is the size of each sample. For model transfer, the number of cloud aggregation t is constrained by $t \leq \frac{u_2}{2EM}$, where E (Unit: MB) is the size of DNN model, \overline{M} is the number of model transfer among nodes (including vehicles, edge servers and cloud server) between two adjacent cloud aggregations, number 2 means that uplink and downlink are both considered. The total number of effective samples y in stages II and III is proportional to М.

The generalization loss of DNNs is proved to be a monotonically decreasing function of the number of effective samples [29]. Effective samples refer to non-repeated data that is in-the-distribution of the target application domain. Therefore, the key is to compute the number of effective samples *S* in CRCHFL framework. Ideally, *S* is the summation of the number of samples in all stages, i.e., S = x + y. However, due to the practical limitations of pretraining and federated learning, the following discounting factors should be added:

- 1) Pretraining in stage I is at the cloud, but inference is at the vehicles. Therefore, a discounting factor $\alpha \in [0,1]$ should be applied to *x* owing to domain shifting. Here *x* actually corresponds to *pretrain_batch*.
- 2) Federated learning is iterative procedure and its convergence rate is $\mathcal{O}(\frac{1}{t})$ [30]. We need to apply discounting factor $(1 dt^{-1})$ to y for finite iterations, where d is parameter representing the convergence rate of federated

Algorithm 1: CRCHFL

Input: $\mathbf{s}_{n,k}$, $\mathbf{a}_{n,k}$, $\mathcal{T}_{n,k}^{I}$, where $n = 1, 2, \dots, N$, and k = $1, 2, \cdots, K_n$ Output: cloud model parameters m_{cloud} 1 run CVXPY to output *edge_interval*, *cloud_interval*, cloud_rounds and pretrain_batch 2 initialize cloud model randomly $\mathbf{m}_{cloud} = \mathbf{m}_{rand}$ 3 collect pretraining data $\{\mathscr{T}_{n,k}^I\}$ 4 model centralized pretraining to get $\mathbf{m}_{cloud,P}$ 5 model release: $\mathbf{m}_{1,1} = \cdots = \mathbf{m}_{N,K_N} = \mathbf{m}_{edge,1} = \cdots =$ $\mathbf{m}_{edge,N} = \mathbf{m}_{cloud,P}$ 6 for round $i = 1, \dots, cloud_rounds$ do for *Edge Server* $n = 1, \dots, N$ *in parallel* do 7 for edge_agg $\tau_2 = 1, \cdots, cloud_interval$ do 8 for Vehicle $k = 1, \dots, K_n$ in parallel do 9 for epoch $\tau_1 = 1, \cdots, edge_interval$ do 10 model updates: $\mathbf{m}_{n,k} \leftarrow {\mathbf{s}_{n,k}, \mathbf{a}_{n,k}}$ 11 if $\tau_1 == edge_interval$ then 12 $\mathbf{m}_{n,k} \Rightarrow Edge \ Server \ n$ 13 end end end if $\tau_2 == cloud_interval$ then 14 $\mathbf{m}_{edge,n} = \sum_{k=1}^{K_n} \frac{|\mathscr{T}_{n,k}|}{\sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|} \mathbf{m}_{n,k}$ 15 $\mathbf{m}_{edge.n} \Rightarrow Cloud$ Server 16 end else 17 $\mathbf{m}_{edge,n} = \sum_{k=1}^{K_n} \frac{|\mathscr{T}_{n,k}|}{\sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|} \mathbf{m}_{n,k}$ 18 $\mathbf{m}_{edgen} \Rightarrow All associated Vehicles$ 19 end end end $\mathbf{m}_{cloud} = \sum_{n=1}^{N} \frac{\sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|}{\sum_{n=1}^{N} \sum_{k=1}^{K_n} |\mathscr{T}_{n,k}|} \mathbf{m}_{edge,n}$ 20 $\mathbf{m}_{cloud} \Rightarrow All \ Edge \ Servers \Rightarrow All \ Vehicles$ 21 end

learning. For CRCHFL, t is decomposed into three factors I_e , I_c and T. Here I_e corresponds to $edge_interval$, I_c corresponds to *cloud_interval* and T corresponds to *cloud_interval* and T corresponds to *cloud rounds*.

3) Even if the federated learning converges, there exists a gap between its performance and that of centralized learning due to the distributed datasets. Therefore, discounting factors $\gamma \in [0, 1]$ is applied to y.

Combining the above observations, the joint resource allocation is formulated as

$$\mathcal{P}: \max_{\{x, I_e, I_c, T\}} \alpha x + \gamma (1 - d(I_e I_c T)^{-1}) y$$

s.t. $x \le \frac{u_1}{D}, 0 < T \le \frac{u_2}{2EM}, 0 < I_c < I_e,$
 $x \ge 0, y \ge 0, u_1 \ge 0, u_2 > 0,$
 $u_1 + u_2 = U_{sum}.$ (6)

Since S is a concave function of $(x, I_e I_c T)$ and the constraints are linear, \mathscr{P} can be solved by tree search over (I_e, I_c) and off-the-shelf numerical solvers CVXPY [31]. We should note that the outputs of optimization algorithm x, I_e , I_c , T are utilized to set *pretrain_batch*, *edge_interval*, *cloud_interval* and *cloud_rounds*, respectively.

V. EXPERIMENTAL RESULTS

This section is divided into four parts: **A. General setup**. In this part, a detailed setup of experiments is introduced, including data sampling, adopted imitation learning (IL) model structure and so forth. **B. Performance comparison** of CRCHFL, HFL and SFL. In this part, we will verify whether the highly elaborated CRCHFL framework can stand out compared to HFL and SFL. **C. Simulation comparison** of CRCHFL, HFL and SFL. In this part, we will simulate CRCHFL, HFL and SFL. In this part, we will simulate CRCHFL, HFL and SFL on CARLA platform to further verify the effectiveness and robustness of our proposed CRCHFL. **D. Ablation study of CRCHFL**. In this part, we will compare the performance of different settings of CRCHFL, and also analyze the results.

A. General Setup

We adopt Town01 and Town02 maps in CARLA to generate training and testing dataset for verification of our proposed CRCHFL scheme. Datasets, both including raw images and actions, are recorded simultaneously when the vehicles driven by CARLA auto-pilot mode are moving forward. To be specific, 2 vehicles are spawned in different zones of Town01 to record 5856 training samples (i.e., each vehicle contains 2928 training samples) and 3 vehicles spawned in Town02 record 2586, 2587, and 1555 data samples, respectively. The testing dataset contains 2081 samples in total, with 1061 from Town01 and 1020 from Town02. Each sample consists of a 3D action vector (throttle, steer, brake) produced by expert drivers treated as ground truth and 4 images captured by cameras CamH, CamF, CamR and CamL. The relative position of these cameras w.r.t. ego-vehicle is represented by a 6D tuple (X, Y, Z, Pitch, Yaw, Roll) (X/Y/Z Unit: m, Pitch/Yaw/Roll Unit: degree). Four cameras are given by (1.5, 0, 1.5, 0, 0, 0), (2.5, 0, 1.5, -50, 0, 0), (1.0, 1.2, 0.5, -50, 90, 0), and (1.0, -1.2, 0.5, -50, -90, 0), respectively. We place one edge server at the center of each town, which is denoted as Edge01 for Town01 and Edge02 for Town02. Two edge servers are connected to cloud server that acts as a global fusion center. Vehicles inside the same town are directly connected with each other via the associated edge server, while vehicles in different towns have no direct link, i.e., their model sharing is based on multi-hop communications via the edge servers and cloud server.

The EAD PyTorch framework by NVIDIA [32] is adopted as a reference to propose our imitation learning model. Our implementation slightly differs from that in [32]: we adopt two branches of individual neural network for predicting steer and throttle/brake, respectively. Each branch calculates its loss, gradient, and back-propagation using Adam optimizer separately. This is because that the steer task is more complex than the throttle/brake task. Note that only image from **CamH** is fed to **Branch I** for throttle/brake task,



Fig. 3: Illustration of entire training and inference process of autonomous driving vehicle. **Digitizer** is used to quantize the steer action into 7-level digital signal, while **DAC** is utilized to convert the predicted digitized steer to an analogical steer to drive the ego-vehicle. The proposed **Model** contains two mutually independent branches where **Branch I** is responsible for brake and throttle signal and **Branch II** is used to predict the steer signal.

while images from **CamF**, **CamL**, and **CamR** are combined together to build a 9-channel image into **Branch II** for steer task. The entire training and inference process of autonomous driving vehicle is demonstrated as in Fig. 3. The system hardware/software configurations and training configurations are listed as Table. I and Table. II, respectively.

TABLE I: Hardware/So	ftware Configurations
----------------------	-----------------------

Items	Configurations
CPU	AMD Ryzen 9 3900X 12-Core
GPU	NVIDIA GeForce 3090×2
RAM	DDR4 32G
DL Framework	PyTorch @ 1.13.0+cu116
GPU Driver	470.161.03
CUDA	11.4
cuDNN	8302
TABLE II: Training Configurations Items Configurations	
TABLE I Items	I: Training Configurations Configurations
TABLE II Items	I: Training Configurations Configurations nn.MSE (Branch I)
TABLE II Items Loss	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II)
TABLE II Items Loss Optimizer	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II) Adam
TABLE I Items Loss Optimizer Adam Betas	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II) Adam (0.9, 0.999)
TABLE I Items Loss Optimizer Adam Betas Weight Decay	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II) Adam (0.9, 0.999) 3e-3
TABLE II Items Loss Optimizer Adam Betas Weight Decay Batch Size	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II) Adam (0.9, 0.999) 3e-3 32
TABLE II Items Loss Optimizer Adam Betas Weight Decay Batch Size Learning Rate	I: Training Configurations Configurations nn.MSE (Branch I) nn.CrossEntropyLoss (Branch II) Adam (0.9, 0.999) 3e-3 32 1e-4

In addition, we adopt two metrics for further quantitative evaluation of CRCHFL hereafter. The first metric is the loss summation of **Branch I** and **Branch II** and is called **Loss**, and the second metric is the accuracy of **Branch II** and is termed **Accuracy**.

B. Performance Comparison of CRCHFL, HFL and SFL

In this part, we will compare CRCHFL with other two benchmarks under fixed communication throughput budget (i.e., 20G) : 1) **HFL** [33], which does not collect data samples for pretraining and has no optimization to schedule constrained communication resources; 2) **SFL** [8], which only shares the model parameters; 3) **CRCHFL**, which is our proposed scheme. For each case, training is stopped when remaining communication throughput is not enough to transfer model parameters once, and then the best performance model from the saved results is then used for testing.

First, Let's compare the performance of CRCHFL, HFL and SFL. With the fixed communication throughput budget (i.e., 20G) and the aforementioned general setup, SFL can perform 13 rounds of cloud aggregation, while HFL can only perform 9 rounds of cloud aggregation because of part of throughput being consumed by communication between edge servers and cloud server. In contrast, although the designed CRCHFL consumes part of the communication throughput when transmitting data samples at the pretraining stage, it can also perform 9 rounds of cloud aggregation due to the proposed optimization algorithm that can schedule communication resources between edge aggregation and cloud aggregation. The inference performance can be found in Fig. 4(a) and Fig. 4(b). It is easy to find that our proposed CRCHFL achieves the best performance in terms of Accuracy and Loss. Specifically, from Fig. 4(a), we can find that the Accuracy of CRCHFL is 10.33% higher than that of HFL and 12.41% higher than that of SFL, and from Fig. 4(b), we can also find that the Loss of CRCHFL is the smallest among them. In addition, due to the introduction of the pretraining stage, it is easy to find that our proposed CRCHFL converges faster than HFL and SFL, so it can be inferred that the designed pretraining stage is necessary and effective.

On the other hand, although the above comparison has illustrated the advantages of our proposed CRCHFL framework, we will further compare these three methods from another perspective where performance is compared when same throughput is consumed by training. Specifically, We further investigate SFL, HFL and CRCHFL by comparing their performance as the consumed throughput increases during one training process. It can be seen from Fig. 5(a) and Fig. 5(b) that: 1. the performance of all three cases improves as consumed communication throughput increases, which is in line with the actual situation; 2. our proposed scheme not



Fig. 4: (a) Comparison of evaluation **Accuracy** of SFL, HFL and CRCHFL w.r.t FL rounds. (b) Comparison of evaluation **Loss** of SFL, HFL and CRCHFL w.r.t FL rounds. These experiments are all conducted under 20GB throughput budget.



Fig. 5: (a) Comparison of evaluation **Accuracy** of SFL, HFL and CRCHFL w.r.t consumed throughput in one training process. (b) Comparison of evaluation **Loss** of SFL, HFL and CRCHFL w.r.t consumed throughput in one training process. These experiments are all conducted under 20GB throughput budget.

only has better **Accuracy** and **Loss** than the other two cases when consuming the same amount of throughput, but also has a more stable evolution trend. This further validates the effectiveness and robustness of our proposed scheme from another perspective.

In summary, just as Fig. 4 and Fig. 5, we evaluate SFL, HFL and CRCHFL from two perspectives. Both results suggest that our elaborated CRCHFL framework has better generalization and faster convergence rate.

C. Simulation Comparison of CRCHFL, HFL and SFL

In this part, we simulate the outputs of CRCHFL, HFL and SFL on CARLA platform as well, and the simulation results are shown as Fig. 6. As it can be seen from Fig. 6(a)and Fig. 6(d), the CRCHFL-controlled ego-vehicle can pass the T-junction smoothly, and when there is a deviation of the trajectory, the actions can be adjusted in time to return to the correct path. However, as it can be seen from Fig. 6(b)HFL-controlled vehicle can though react properly when the trajectory deviates but collide when pass the T-junction, and from Fig. 6(e) HFL-controlled vehicle collides the pole beside the road owing to overreaction of deviation. From Fig. 6(c) and Fig. 6(f) SFL-controlled vehicle collides or crosses the border while passing through the T-junction. In summary, we can conclude that CRCHFL outperforms HFL and SFL at some scenarios in both Town01 and Town02.

In addition, examples of predicted actions by CRCHFL in inference stage can be checked in Fig. 7. It can be seen from Fig. 7(a) and Fig. 7(b) that the ego-vehicle stops in front of a red traffic light and accelerates immediately after the light turns green. This is because that CRCHFL directly maps the raw data into action vector (0.028, 0.000, 0.896) in Fig. 7(a) and (0.346, 0.000, 0.000) in Fig. 7(b), thus realizing the above rule of understanding, and stopping or moving actions. In Fig. 7(c), the vehicle successfully turns left at the T-junction by properly combining throttle and steer actions as (0.393, -0.275, 0.000).

As can be seen from Fig. 6 and Fig. 7, both the macroscopic comparison of the trajectories of SFL, HFL and CRCHFL and the microscopic insight of predicted actions are illustrations of the good performance of our proposed scheme. On top of V-B, it further proves the feasibility and effectiveness of our proposed method.

D. Ablation Study of CRCHFL

In experiment V-B and V-C, we have verified that the proposed CRCHFL is feasible and effective compared with SFL and HFL in terms of performance and simulation. In this part, we will do some ablation experiments in different CRCHFL settings. To investigate how the communication resource distribution changes across stages, we can change the conditions of the optimization algorithm to lead to different optimization resource distribution. By comparing these outcomes, we can then derive the variation pattern of the communication resource distribution. To be specific, in our experiments, by fixing *edge_interval* to 2,3 *and* 4 respectively, the communication resource distribution to 2,3 *and* 4 respectively.



(a) CRCHFL @ Town01

(b) HFL @ Town01

(c) SFL @ Town01



(d) CRCHFL @ Town02

(e) HFL @ Town02

(f) SFL @ Town02

Fig. 6: Simulation Comparison between CRCHFL (a,d), HFL (b,e) and SFL (c,f) on CARLA platform. The boxes in above figures depict the trajectory of ego-vehicle.



(a) Brake

(b) Throttle

(c) Throttle and Steer

Fig. 7: (a) Brake action predicted by CRCHFL when the traffic light turns red; (b) Throttle action predicted by CRCHFL when the traffic light turns green; (c) Throttle-and-steer action predicted by CRCHFL when the vehicle passes through a T-junction. The boxes in the figure (b) illustrate the Field of View (FoV) of four cameras.



Fig. 8: (a) Illustration of fixed communication throughput budget (i.e., 20G) distribution across stages for different CRCHFL settings. (b) Test loss under different CRCHFL settings. (c) Test accuracy under different CRCHFL settings.

nication resources allocated to each stage change and the distribution is shown in Fig. 8(a).

From Fig. 8(a), we can find that the communication resource distribution reveals the pattern of change. Specifically, as *edge_interval* increases, communication resources are allocated more to Stage III whereas less to Stage I and II, which means that more communication resources are used for cloud aggregation and less communication resources are used for pretraining samples transfer and edge aggregation.

In order to explore what happened to the performance of CRCHFL when communication resource distribution varies, we can check the **Loss** and **Accuracy** of such CRCHFL settings in Fig. 8(b) and Fig. 8(c). It is obvious that Interval-4 has the highest **Accuracy** and the lowest **Loss** because it has the largest FL rounds. This means that when more

communication resources are used for cloud aggregation, it helps to improve the performance of the model, which is also in line with our expectation. It is also found that the **Accuracy** and **Loss** of the three cases do not differ too much in the first round of federated learning, especially for **Accuracy**, which also indicates that only a small number of samples are needed to upload in the pretraining stage to accelerate convergence rate.

In conclusion, through the above experiments and simulations, our proposed framework performs significantly better than HFL and SFL with limited communication resources, mainly because: 1. a small number of data samples are uploaded from each vehicle to the cloud server in the pretraining stage to form a centralized dataset which is used to pre-train the model and accelerate its convergence rate; 2. by using optimization-based method to set *edge_interval* and *cloud_interval*, communication resources can be scheduled for more rounds of cloud aggregation, which in turn improves the model performance overall.

VI. CONCLUSION

This paper has proposed a communication resources constrained hierarchical federated learning framework, which aims at finding a trade-off between learning performance and communication resources. Experimental results of our proposed framework demonstrated that our proposed CRCHFL algorithm outperforms existing benchmarks especially when the network budget is tight. However, our framework and experiments still have limitations, such as no consideration of communication channel fading, small number of towns and vehicles, etc. Future directions include multi-modal CRCHFL and ROS implementation of CRCHFL.

REFERENCES

- L. Zhang, X. Lei, Y. Shi, H. Huang, and C. Chen, "Federated learning with domain generalization," *CoRR*, vol. abs/2111.10487, 2021.
 [Online]. Available: https://arxiv.org/abs/2111.10487
- [2] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated transfer reinforcement learning for autonomous driving," *CoRR*, vol. abs/1910.06001, 2019. [Online]. Available: http://arxiv.org/ abs/1910.06001
- [3] B. Liu, L. Wang, M. Liu, and C. Xu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *CoRR*, vol. abs/1901.06455, 2019. [Online]. Available: http: //arxiv.org/abs/1901.06455
- [4] B. Yang, H. Shi, and X. Xia, "Federated imitation learning for uav swarm coordination in urban traffic monitoring," *IEEE Transactions* on *Industrial Informatics*, pp. 1–10, 2022.
- [5] S. Wang, C. Li, Q. Hao, C. Xu, D. W. K. Ng, Y. C. Eldar, and H. V. Poor, "Federated deep learning meets autonomous vehicle perception: Design and verification," 2022. [Online]. Available: https: //arxiv.org/abs/2206.01748
- [6] Z. Zhang, S. Wang, Y. Hong, L. Zhou, and Q. Hao, "Distributed dynamic map fusion via federated learning for intelligent networked vehicles," *CoRR*, vol. abs/2103.03786, 2021. [Online]. Available: https://arxiv.org/abs/2103.03786
- H. Zhang, J. Bosch, and H. H. Olsson, "Real-time end-to-end federated learning: An automotive case study," *CoRR*, vol. abs/2103.11879, 2021. [Online]. Available: https://arxiv.org/abs/2103. 11879
- [8] A. Nguyen, T. Do, M. Tran, B. X. Nguyen, C. Duong, T. Phan, E. Tjiputra, and Q. D. Tran, "Deep federated learning for autonomous driving," *CoRR*, vol. abs/2110.05754, 2021. [Online]. Available: https://arxiv.org/abs/2110.05754
- [9] B. Liu, L. Wang, X. Chen, L. Huang, and C. Xu, "Peer-assisted robotic learning: A data-driven collaborative learning approach for cloud robotic systems," *CoRR*, vol. abs/2010.08303, 2020. [Online]. Available: https://arxiv.org/abs/2010.08303
- [10] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Communications Magazine*, vol. 59, pp. 16–21, 2021.
- [11] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen, and W. Ding, "Towards efficient communications in federated learning: A contemporary survey," 2022. [Online]. Available: https://arxiv. org/abs/2208.01200
- [12] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, pp. 2325–2383, 2022.
- [13] N. F. Eghlidi and M. Jaggi, "Sparse communication for training deep networks," CoRR, vol. abs/2009.09271, 2020. [Online]. Available: https://arxiv.org/abs/2009.09271
- [14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/ abs/1503.02531

- [15] S. Salehkalaibar and S. Rini, "Lossy gradient compression: How much accuracy can one bit buy?" *CoRR*, vol. abs/2202.02812, 2022. [Online]. Available: https://arxiv.org/abs/2202.02812
- [16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of The 1st Annual Conference on Robot Learning*, Mountain View, CA, Oct. 2017, pp. 1–16.
- [17] Z. Sun, Z. Huang, Q. Zhu, X. Li, and D. Liu, "High-precision motion control method and practice for autonomous driving in complex offroad environments," in 2016 IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 767–773.
- [18] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone, "Multi-robot planning with conflicts and synergies," *Autonomous Robots*, vol. 43, no. 8, pp. 2011–2032, 2019.
- [19] M. Kelly, C. Sidrane, K. R. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," *CoRR*, vol. abs/1810.02890, 2018. [Online]. Available: http://arxiv. org/abs/1810.02890
- [20] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving," *CoRR*, vol. abs/1905.00229, 2019. [Online]. Available: http://arxiv.org/abs/1905.00229
- [21] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286– 302, 2020. [Online]. Available: https://doi.org/10.1177/ 0278364919880273
- [22] B. Liu, L. Wang, M. Liu, and C.-Z. Xu, "Federated imitation learning: A novel framework for cloud robotic systems with heterogeneous sensor data," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3509–3516, 2020.
- [23] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 537–547, 2022.
- [24] J. Yuan, M. Xu, X. Ma, A. Zhou, X. Liu, and S. Wang, "Hierarchical federated learning through LAN-WAN orchestration," *CoRR*, vol. abs/2010.11612, 2020. [Online]. Available: https://arxiv.org/ abs/2010.11612
- [25] D. Alistarh, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: randomized quantization for communication-optimal stochastic gradient descent," *CoRR*, vol. abs/1610.02132, 2016. [Online]. Available: http:// arxiv.org/abs/1610.02132
- [26] A. N. Sahu, A. Dutta, A. M. Abdelmoniem, T. Banerjee, M. Canini, and P. Kalnis, "Rethinking gradient sparsification as total error minimization," *CoRR*, vol. abs/2108.00951, 2021. [Online]. Available: https://arxiv.org/abs/2108.00951
- [27] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *CoRR*, vol. abs/1910.03581, 2019. [Online]. Available: http://arxiv.org/abs/1910.03581
- [28] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/ abs/1602.05629
- [29] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 269–283, 2021.
- [30] H. SEUNG, H. SOMPOLINSKY, and N. TISHBY, "Statisticalmechanics of learning from examples," *Physical review. A, Atomic, molecular, and optical physics*, vol. 45, no. 8, pp. 6056–6091, 1992.
- [31] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
- [32] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http:// arxiv.org/abs/1604.07316
- [33] J. Tursunboev, Y.-S. Kang, S.-B. Huh, D.-W. Lim, J.-M. Kang, and H. Jung, "Hierarchical federated learning for edge-aided unmanned aerial vehicle networks," *Applied Sciences*, vol. 12, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/2/ 670