FedPylot: Navigating Federated Learning for Real-Time Object Detection in Internet of Vehicles

Cyprien Quéméneur, *Graduate Student Member, IEEE,* and Soumaya Cherkaoui, *Senior Member, IEEE*

***Abstract*—The Internet of Vehicles (IoV) emerges as a pivotal component for autonomous driving and intelligent transportation systems (ITS), by enabling low-latency big data processing in a dense interconnected network that comprises vehicles, infrastructures, pedestrians and the cloud. Autonomous vehicles are heavily reliant on machine learning (ML) and can strongly benefit from the wealth of sensory data generated at the edge, which calls for measures to reconcile model training with preserving the privacy of sensitive user data. Federated learning (FL) stands out as a promising solution to train sophisticated ML models in vehicular networks while protecting the privacy of road users and mitigating communication overhead. This paper examines the federated optimization of the cutting-edge YOLOv7 model to tackle real-time object detection amid data heterogeneity, encompassing unbalancedness, concept drift, and label distribution skews. To this end, we introduce FedPylot, a lightweight MPI-based prototype to simulate federated object detection experiments on high-performance computing (HPC) systems, where we safeguard server-client communications using hybrid encryption. Our study factors in accuracy, communication cost, and inference speed, thereby presenting a balanced approach to the challenges faced by autonomous vehicles. We demonstrate promising results for the applicability of FL in IoV and hope that FedPylot will provide a basis for future research into federated real-time object detection. The source code is available at https://github.com/cyprienquemeneur/fedpylot.**

***Index Terms*—Federated learning, Internet of Vehicles, object detection, autonomous driving, intelligent transportation systems.**

I. Introduction

INTELLIGENT transportation systems (ITS) are expected to reshape mobility by enhancing safety, streamlining traffic flow, reducing vehicle emissions and fuel consumption, and providing infotainment services. This transformation is powered by advances in machine learning (ML) and vehicle-to-everything (V2X) communication technologies, fostering seamless cooperation between a network of vehicles, pedestrians, and infrastructures, generating a vast amount of data and integrated into a cohesive Internet of Vehicles (IoV) [1]. To enable data sharing, IoV relies on state-of-the-art wireless network technologies that can offer long-range, low latency, reliable, and secured transfers [2]. In turn, connected automated vehicles can leverage the information sharing facilitated by IoV to enhance their situational awareness, yet their abilities to solve advanced navigation tasks nonetheless hinge on machine learning (ML) models [3]. Furthermore, although vehicles can utilize an array of sensors such as cameras, LiDAR, radar, and GPS systems to collect diverse multimodal data, crucial for supporting ML and making timely decisions, the offloading of model training to the cloud raises privacy, availability, and, in the long run, scalability concerns. To palliate these issues, federated learning (FL) has been proposed as a solution to facilitate collaborative edge model training and protect user privacy while alleviating the communication bottlenecks arising in IoV [4], [5]. In FL, raw data sharing is prohibited and training takes place locally on edge clients, which then rely on a central aggregation agent to regularly gather and combine model updates. This process is depicted for IoV in Fig. 1, where the main clients are vehicles, and the central aggregation server is strategically positioned at a network edge location to reduce latencies. Additionally, edge servers dispersed across a given geo-region may themselves fallback on a cloud platform to facilitate orchestration at scale, or provide a second layer of aggregation [6].

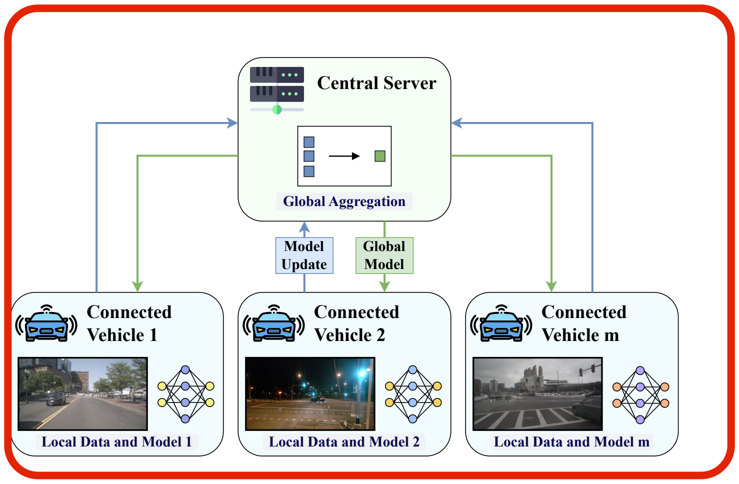

Fig. 1. **Vehicular clients collaboratively learn a joint model with FL.** The vehicles collect driving data using various sensors to train their own local model, while a central server is responsible for regularly gathering and aggregating local weight-update vectors to compute a global model, which is subsequently disseminated to the clients for further training. The raw data are kept private, but are typically non-identically distributed due to the decentralized nature of the clients (samples taken from nuImages).

Real-time vision is an indispensable requirement for automated vehicles, as attaining full driving autonomy involves sophisticated vision-based systems capable of achieving human-level perception in complex and dynamic environments [7]. Numerous studies have explored the potential to improve the visual perception capabilities of vehicles with FL, and covered traffic sign recognition [8]–[10], pothole and other road damage detection [11]–[13], semantic segmentation [14], [15], and more commonly object detection [16]–[31]. Other

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). (*Corresponding author: Cyprien Quéméneur.*)

The authors are with the Department of Computer and Software Engineering, Polytechnique Montréal, Montreal, QC H3T 1J4 Canada (e-mail: cyprien.quemeneur@polymtl.ca; soumaya.cherkaoui@polymtl.ca).

tasks relying in part but not solely on the visual perception of vehicles have also been explored with FL, such as trajectory prediction [32] and collision avoidance [33]. However, these investigations typically employ either models not suitable for real-time vision or relevant but outdated ML models. This, in turn, diminishes the applicability of the findings for evaluating the feasibility of FL with modern real-time vision ML models. In contrast, this work shifts its focus to YOLOv7 [34], one of the latest entries in the YOLO (You Only Look Once) series of real-time object detection models. Through numerous successive improvements, YOLO models have gained significant recognition in the field of computer vision and have been widely used in various applications, including autonomous vehicles, surveillance systems, and robotics [35]. Furthermore, this research explores the application of YOLOv7 within a FL framework for autonomous driving situations marked by data sources stemming from vehicles associated with different geographical locations and timeframes, resulting in vehicles encountering heterogeneous data.

The contributions of this paper can be summarized as follows:

- We develop FedPylot, a Message Parsing Interface (MPI)-based FL prototype dedicated to federated object detection experiments on high-performance computing (HPC) systems, and implement a hybrid cryptosystem to secure the communications between FL participants.
- We propose, to the best of our knowledge, the first federated framework for YOLOv7 that allows for high-scale experimentation. We considered predictive performances, inference speed and communication overhead in our evaluation, emphasized local optimization and included server-side momentum and custom learning rates.
- We demonstrate FedPylot on two relevant autonomous driving datasets and simulate data heterogeneity arising from spatiotemporal shifts, and account for unbalancedness, concept drift, and label distribution skews. We capture heterogeneity at different granularity by including two class maps of a long-tail distribution, while considering navigation sequences in our splitting strategy.

We make FedPylot open-source in the hope that it will be helpful to the object detection community for their FL-related projects. FedPylot retains the ability to scale to a large number of computation nodes, while being easier to approach than advanced FL frameworks (e.g., FedML [36], PySyft [37], Flower [38]), in which integrating complex self-defined models, like state-of-the-art object detectors, can prove troublesome.

The remainder of this paper is organized as follows. Section II outlines the fundamental concepts and tools used in this work. Subsequently, Section III provides a review of the current literature surrounding the intersection of object detection, federated learning, and autonomous driving. In Section IV, we detail the theoretical design of our FL prototype. Section V is dedicated to the implementation of FedPylot and our experimental settings. Section VI outlines the results of our experiments. Finally, Section VII concludes the paper.

II. Background

A. Federated learning

1) Formulation: FL was proposed in 2016 by McMahan et al. [39]. In traditional FL, m clients $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m\}$, with the help of an orchestrating server $\mathcal{S}$, participate during several communication rounds in the collaborative training of a shared machine learning model w without revealing their respective local dataset $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_m\}$. We assume a horizontal partitioning where the local datasets of the clients differ in their data samples but share the same feature space. The size of each local dataset is designated by $n_i = |\mathcal{D}_i|$, with $n = \sum_{i=1}^{m} n_i$. The optimization goal is to minimize

$$\min_{w} \mathcal{L}(w) = \sum_{i=1}^{m} \frac{n_i}{n} L_i(w) \tag{1}$$

where $L_i(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell_i(x, y; w)]$ is the possibly non-convex local objective function of client $\mathcal{C}_i$. At the beginning of the training process, the central server initializes the global model from random or pre-trained weights and shares it with the clients. In the original and baseline algorithm FedAvg, at the beginning of the communication round t, a subset of clients with indices in K is randomly selected to participate in the round. These clients then receive the shared model w^t from the server and perform several epochs E of local training on their respective local dataset using stochastic gradient descent (SGD) with local mini-batch size B. The central server then collects the updated local models w_i^t from each client $\mathcal{C}_i$ and aggregates them, thus yielding the next instance of the global model

$$w^{t+1} = \sum_{i\in K} \frac{n_i}{n} w_i^t. \tag{2}$$

Reddi et al. [40] formalized FedOpt as a direct generalization of FedAvg. In FedOpt, it is no longer assumed that the local optimizers of the clients are necessarily SGD, and the update rule of the global model is rephrased as an optimization problem. Assuming that $\mathcal{C}_i$ transmits the weight-update vector $\Delta_i^t = w^t - w_i^t$ to the server, the latter can aggregate the local updates to form a *pseudo-gradient* Δ^t, which is then inputted to a server optimizer. In particular, (2) is equivalent to

$$w^{t+1} = w^t - \Delta^t = w^t - \sum_{i\in K} \frac{n_i}{n} \Delta_i^t \tag{3}$$

which corresponds to using SGD as the server optimizer with a learning rate of 1. To assess the convergence of the training procedure, it is necessary to evaluate the global model. Evaluation can be performed server-wise on a separate dataset $\mathcal{D}_\mathcal{S}$ left on the server, or separately on some clients, before aggregating the resulting local statistics. The original FL process can be declined in a variety of ways, for example, personalized federated learning (pFL) allows for some degree of customization of the shared model that is no longer unique to all clients [41], while some implementations leverage blockchain to achieve complete decentralization by eliminating the need for a central server entirely [42]; one may also assume a vertical partitioning, where the local datasets differ in the feature space instead of the sample space [43].

2) Data heterogeneity: In FL, the data are usually non-identically distributed (non-IID) among the clients, which may lead to local model divergence during training and degraded accuracy for the global model. Kairouz et al. [44] presented various forms of data heterogeneity, three of which are relevant to our study:

- Concept drift: The same label may have vastly different features for different clients, for example, due to varying weather conditions, locations, or time scales.
- Label distribution skew: Differences in the distribution of data labels can arise across clients, for example, because the likelihood of encountering certain labels is tied to some specific clients' environment.
- Unbalancedness: The number of data samples stored can vary greatly from client to client.

FL practitioners have devised several strategies to simulate data heterogeneity in their experiments. In image classification, label skewness is widely studied and the Dirichlet distribution is commonly used to create artificial non-IID splits of a dataset [45]. Conversely, in tasks such as object detection, data heterogeneity is multifaceted and better simulated by identifying natural semantic separations within the dataset, such as weather or location in the case of ITS. Many popular techniques have been introduced to address the issue of data heterogeneity, including, but not limited to, FedProx [46] which introduces a proximal term to each local objective function to improve robustness against variable local updates, SCAFFOLD [47] which uses control variates to perform variance reduction and counterbalance the client-drift, and MOON [48] which applies contrastive learning at the model level to correct local training based on similarities between model representations. However, many of these algorithms were originally designed for image classification and may not be as effective when applied as is to more challenging vision tasks [49]. Other methods are more readily compatible with object detection, and we discuss them in the following. Strategies based on replacing server-side SGD by a more advanced ML optimizer, such as Adam [50], are orthogonal to the ones mentioned above and can improve convergence in non-IID settings [40]. Similarly, Hsu et al. [45] introduced momentum in server-side optimization to create FedAvgM and empirically showed improved performances under heterogeneous distributions. In FedAvgM, the global update integrates a velocity term v, which accumulates exponentially decaying past pseudo-gradients at a rate controlled by a constant momentum factor $\beta \in [0, 1)$. FedAvgM can be generalized by adding a supplementary constant learning rate η, the update rule thus becoming as follows

$$v^{t+1} = \beta v^t + \Delta^t \tag{4a}$$
$$w^{t+1} = w^t - \eta v^{t+1} \tag{4b}$$

and where taking $\beta = 0$ and $\eta = 1$ yields back the original FedAvg algorithm. Momentum can be applied at the client-level as well to improve the stability of local updates, while retaining convergence in non-IID settings [51], [52]. Starting federated learning after a pre-training phase instead of random initialization, using proxy data available on the server, has also been shown to improve the stability of global aggregation and close the gap with centralized learning, even when data are heterogeneous [53], [54].

3) Other challenges in FL: FL elicits a multitude of other challenges, such as designing incentives to encourage the clients with the largest data pools to participate in the federated process, ensuring fairness between clients, accounting for heterogeneity in the computing capabilities of the clients, designing FL frameworks, and limiting overheads. Furthermore, despite its added privacy benefits, FL is not exempt from risks. Malicious actors participating can gain insight into the original training data or degrade training integrity by conducting attacks such as model inversion and model update poisoning. Counter techniques like secure multi-party computation [55], homomorphic encryption [56], and differential privacy [57] can help alleviate this problem, but may incur increased computation and communication overheads or performance degradation.

B. Object detection

Object detection is a computer vision task in which a model seeks to detect all instances of object classes of interest in an image and report their location and spatial extent by delimiting them with bounding boxes [58]. In real-time object detection, the inference speed of the model is also deemed critical to the realization of the task and is commonly measured either in milliseconds or in frames-per-second (FPS). Deep learning made a substantial impact to object detection in the past decade, and modern object detectors are usually based on convolutional neural networks. Two-stage object detectors, such as R-CNN [59] and SPP-Net [60], first generate region proposals before sending them to a classification model, while one-stage object detectors, such as YOLO [61], SSD [62] and RetinaNet [63], locate and classify objects in a single swipe, usually making them faster but less accurate. One-stage and two-stage object detectors alike commonly rely on the non-maximum suppression (NMS) post-processing technique to filter overlapping predictions and retain only the most relevant bounding box for a given object. More recently, transformer-based object detectors have gained interest, such as detection transformers (DETRs) [64]. DETRs eliminate the need for many hand-designed components such as NMS to perform end-to-end detection, and efforts are being directed to enable them in the real-time setting [65].

The most popular metrics used to evaluate the predictive performance of object detectors are reviewed by Padilla et al. [66], for which we propose a brief recapitulation in the following. The location accuracy of the predicted bounding box B_p at threshold t is given by the ratio between the area of overlap and the area of union of B_p and the ground truth B_{gt}, and is called the intersection over union

$$\mathrm{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{5}$$

with the prediction being deemed correct if $\mathrm{IoU} \geq t$. Having established the criterion to determine the correctness of the detection, it is possible to derive the precision $\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$

and recall $\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ of the model, where TP, FP and FN refer to True Positive, False Positive and False Negative, respectively. The average precision AP_t^k of the model for class k at threshold t is then interpolated from the area under curve (AUC) of the precision × recall curve. The overall accuracy of the model evaluated on a dataset of N classes is called the mean average precision, and is defined as

$$\mathrm{mAP}_t = \frac{1}{N}\sum_{k=1}^{N} \mathrm{AP}_t^k. \tag{6}$$

To benchmark a model using a single threshold t may not be satisfactory. Therefore, it is common to average the mean average precision across ten IoU thresholds, from 50% to 95% with a step of 5%, resulting in a metric which is simply referred to as mAP.

C. YOLOv7

YOLOv7 is one of the latest entries in the YOLO family of one-stage real-time object detectors. At release, in July 2022, it was the fastest and most accurate object detector in the 5 to 160 FPS range, had fewer parameters than comparatively performing models, and was a leap from the still broadly popular YOLOv5 [67]. YOLOv7 was trained from scratch on MS COCO [68] and features optimized structures and optimization methods dubbed *trainable bag-of-freebies*. These include planned re-parameterized convolution based on gradient flow path analysis, and a novel label assignment strategy, as well as some pre-existing concepts, like embedding batch normalization statistics directly in convolutional layers for inference, YOLOR implicit knowledge [69] merged into convolutional layers for additions and multiplications, and using an exponential moving average (EMA) model as the final inference model. Bag-of-freebies strengthens the training cost and predictive performance of a model but without increasing latencies during inference, a concept previously discussed in YOLOv4 [70]. YOLOv7 also introduced architectural improvements, including Extended-ELAN, a modified variant of the Efficient Layer Aggregation Network (ELAN) [71] that allows stacking computational blocks indefinitely while retaining learning ability, and a new method for compound scaling of concatenation-based models that preserves the model's properties and optimal structure.

More generally, YOLOv7 is an anchor-based model that uses a feature pyramid network (FPN) [72]. The authors introduce YOLOv7-tiny and YOLOv7, which are P5 models respectively designed for edge and normal GPUs, and YOLOv7-W6, which is a P6 model designed for cloud GPUs. The compound scaling method is applied to YOLOv7 to derive YOLOv7-X, and to YOLOv7-W6 to derive YOLOv7-E6, YOLOv7-D6, YOLOv7-E6E, with the latter replacing ELAN by Extended-ELAN as its main compute unit. The recommended training input resolution is 640×640 for P5 models and 1280×1280 for P6 models. Images passed to YOLOv7 are resized to the given input while maintaining aspect ratio, while padding is applied if necessary. During training, the loss is computed by summing three sub-functions balanced by predefined gain hyperparameters. These sub-functions measure the performance of the model across different modalities of the object detection problem. In particular, the classification loss and objectness loss use the common Binary Cross-Entropy (BCE) loss, whereas the bounding box regression loss sub-function is instead based on the Complete IoU (CIoU) loss [73]. All YOLOv7 variants use the SiLU activation function (except for YOLOv7-tiny where LeakyReLU is used instead), the SimOTA strategy introduced in YOLOX for label assignment [74], mosaic, mixup and left-right flip augmentations, gradient accumulation, automatic mixed precision training, half precision at inference, and NMS for post-processing.

In addition to 2D object detection, YOLOv7 was extended to support pose estimation and instance segmentation, and an anchor-free variant, YOLOv7-AF, was made available with base performances on par or surpassing later releases, including YOLOv6 3.0 [75] and YOLOv8 [76]. YOLOv7 was used as the basis for YOLOv9 [77], which introduced Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN).

III. Related work

In the following, we propose a brief review of previous work that applied FL to tackle 2D object detection in autonomous vehicles. Rjoub et al. [16] investigated federated real-time object detection in adverse weather driving scenarios with the original YOLO model. Chen et al. [17] studied FL in vehicular object detection with SSD under communication and computation constraints. Bommel [18] and later Rjoub et al. [19] used active learning, with respectively YOLOv5s and YOLOR, to address the predicament of sparse data labeling in FL for real-time object detection in autonomous vehicles. Dai et al. [20] proposed FLAME, a framework intended to facilitate the exploration of online federated object detection on data continuously streamed by autonomous vehicles, which they demoed with YOLOv2 [78]. Wang et al. [21] designed CarlaFLCAV, an open FL simulation platform that supports a wide range of automotive perception tasks, including object detection with YOLOv5, and tackled network resource and road sensor placement optimization. Chi et al. [22] leveraged a soft teacher semi-supervised object detection framework to perform FL training with Faster-R-CNN [79] on unlabeled data collected while driving, given a small amount of well-curated preexisting data. Rao et al. [23] proposed FedWeg, a sparse FL training process that they evaluated on YOLOv3 [80], to accommodate computational and communication constraints in IoV. Su et al. [24] proposed FedOD, a cross-domain pFL framework for object detection based on multi-teacher distillation, and validated their proposal with RetinaNet on autonomous driving datasets. Finally, Kim et al. [25] proposed a two-stage training strategy named FedSTO and tackled semi-supervised federated object detection with YOLOv5 in heterogeneous situations where the local datasets of the vehicular clients are fully unlabeled and labeled data are only available to the central server.

In contrast, other researchers instead shifted their attention to multimodal data. In particular, Zheng et al. [26] proposed

AutoFed, a FL framework dedicated to bird's-eye view vehicle detection where data heterogeneity results from the inclusion of multiple sensing modalities, and trained a custom two-stage object detector that accommodates LiDAR and radar data on several NVIDIA Jetson TX2 devices. Mishra et al. [27] argued in favor of a fully decentralized blockchain-based autonomous driving FL system with smart contracts, called *swarm learning* in the paper, and validated their proposal on 3D point cloud object detection with Complex-YOLOv4 [81]. Moreover, Chi et al. [28] showed that federated 3D object detection can be improved by incorporating infrastructure into a clustered federated procedure, and included a complex multistep perception system in their experiments, where feature maps extracted from point clouds were processed by a vision transformer.

Our proposal diverges from previous work by focusing on the federated optimization of a recent real-time object detector and proposing a comprehensive evaluation of its performances, while allowing high-scale simulations in data centers. Our splitting strategy, which we detail in Section V-C, features particularities not found in these papers. We also propose one of few works to provide an open implementation.

Two papers experimented with YOLOv8 and respectively introduced FedProx+LA, a FL method to address label distribution skews in vehicular networks which showed improved performance and convergence speed for object detection [29], and an adaptive clustered FL technique to address storage and bandwidth limitations, validated for car detection under varying weather and lighting conditions [30]. However, these works only considered the nano, i.e., the smallest, variant of YOLOv8, which is designed for edge devices and has limited learning capabilities. This further emphasizes the need to provide support for larger-scale FL experimentation. Perhaps most similar to our proposal is the prototype introduced by Jallepalli et al. [31] which, to the best of our knowledge, is the only contribution that has featured the federated optimization of an object detector in an HPC environment for the purpose of autonomous driving. We significantly improve upon this prior work by replacing YOLOv3 by YOLOv7, introducing higher data diversity, more clients, realistic non-IID settings, more configurations for local and server-wise optimization, measurements of communication costs and inference speed for several model variants, and by replacing low-level socket-based communications by MPI, and the symmetric Fernet encryption scheme by a more secure hybrid one.

IV. System design

Several connected and autonomous vehicles collect video data in real time using onboard cameras, while participating in the federated training of a shared object detection model using the YOLOv7 architecture. A server acts as the FL system orchestrator and is responsible for the collection and aggregation of local model updates, server-side optimization, and dissemination of model parameters. The server possesses its own set of well-curated representative data, which is used to assess the quality of the global model at the end of each communication round. The server optimizer used in our experiments is FedAvgM (4), although FedPylot also supports

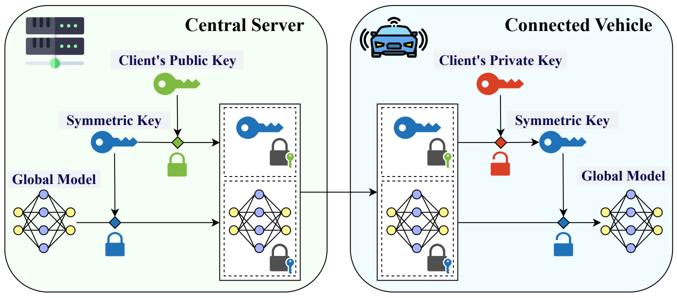

Fig. 2. **Hybrid cryptosystem for server-client communications**. Transmissions between the server and the vehicular clients are encrypted using a highly efficient symmetric algorithm. The symmetric key is generated by the server and protected using a public-key cryptosystem when passed to the clients.

Algorithm 1 Federated YOLOv7

```
Input: Communication rounds $R$, local epochs $E$, local mini-batch size $B$, server learning rate $\eta$, server momentum factor $\beta \in [0,1)$, YOLOv7 set of hyperparameters $\mathcal{H}$.
Output: Best re-parameterized joint model $w_{\mathrm{r}}^{*}$.

Server $\mathcal{S}$ executes:
  Gather public key $pk_i$ from each client $\mathcal{C}_i$
  Initialize model $w^0$ from pretrained weights
  Initialize server-side momentum $v^0 = 0$
  for $t = 0, ..., R-1$ do
    Initialize symmetric key $sk^t$
    $\{w^t\} \leftarrow$ Encrypt $w^t$ with $sk^t$
    $\{sk^t\}_{i:1:m} \leftarrow$ Encrypt $sk^t$ for each $pk_i$
    for each client $\mathcal{C}_i$ in parallel do
      $\{\Delta_i^t\}, n_i =$ ClientUpdate($\{w^t\}, \{sk^t\}_i, t$)
    end for
    $n = \sum_{i=1}^{m} n_i$
    $\Delta_{i:1:m}^t \leftarrow$ Decrypt each $\{\Delta_i^t\}$ with $sk^t$
    $\Delta^t = \sum_{i=1}^{m} \frac{n_i}{n} \Delta_i^t$
    $v^{t+1} = \beta v^t + \Delta^t$
    $w^{t+1} = w^t - \eta v^{t+1}$
    $w_{\mathrm{r}}^{t+1} = \mathtt{reparameterize}(w^{t+1})$
    Evaluation metrics $= \mathtt{test}(w_{\mathrm{r}}^{t+1}, \mathcal{D}_{\mathcal{S}}, \mathcal{H})$
  end for
  return $w_{\mathrm{r}}^{*}$ based on mAP
ClientUpdate($\{w^t\}, \{sk^t\}_i, t$):    // run on client $\mathcal{C}_i$
  $sk^t \leftarrow$ Decrypt $\{sk^t\}_i$ with private key
  $w^t \leftarrow$ Decrypt $\{w^t\}$ with $sk^t$
  $w_i^t = \mathtt{train}(w^t, \mathcal{D}_i, E, B, \mathcal{H}, t)$
  $\Delta_i^t = w^t - w_i^t$
  $\{\Delta_i^t\} \leftarrow$ Encrypt $\Delta_i^t$ with $sk^t$
  return $\{\Delta_i^t\}$, $|\mathcal{D}_i|$ to server
```

FedAdagrad, FedAdam and FedYogi [40]. To more accurately reflect the state of the model in deployment, evaluation is performed on the re-parameterized model, which has fewer parameters yet retains the same predictive performances as the base model. A round encompasses a fixed number of local training epochs common to all clients with a shared batch size irrespective of the size of the training sets, hence the number of local optimization steps may vary between clients. For simplicity, we assumed full client participation for each round, availability of the training data labels at the edge, and synchronicity of aggregation.

Communications between participants are secured through a hybrid cryptosystem. It consists in combining a highly efficient symmetric encryption algorithm, which is used to encrypt the plaintext but requires the same symmetric key to be available to both the sender and the receiver, with a more practical but several orders of magnitude more expensive public-key encryption algorithm, used only to protect the symmetric key from adversaries when it is being transferred. We design our system so that each vehicular client generates a public-private key pair at the beginning of the federated process and immediately transmits its public key to the server. Meanwhile, the server generates a new symmetric key at the beginning of each round and uses it to encrypt the global model, before passing the two to the clients, either alongside each other as in Fig. 2, or separately if decoupling key and model exchanges is deemed more practical. The clients can then use this same key to decrypt the global model and encrypt their local updates. Hybrid encryption is straightforward and well adapted to environments with low-latency constraints. Indeed, it incurs only negligible communication overhead and, in our setting, the encryption overhead is largely dominated by the cost of model training. However, it is only suitable when revealing the model updates to the server is acceptable. If this condition is not met, the addition of advanced privacy techniques, such as those mentioned in Section II-A3, becomes mandatory. The federated procedure is summarized in Algorithm 1, while the remaining details are covered in the next Section.

V. Experiments

A. Prototype implementation details

1) Federated simulation: Our experiments were carried out with PyTorch [82] on Cedar.[1] Each FL participant was identified with exactly one compute node equipped with one Tesla V100-SXM2-32GB GPU, eight CPU cores, and up to 6000 MiB of memory per core. Before the beginning of a FL experiment, the local dataset of each client was transferred from the network storage to the local disk of the corresponding compute node, isolating it from the other participants, and accelerating input and output (I/O) transactions. The same was done with the validation set and the server node. The interconnect was Intel Omni-Path [83], and we handled the communications between the server and the clients with MPI, using the Open MPI implementation of the standard [84], and the mpi4py Python package [85]. MPI was specifically designed for HPC environments and is available as a communication backend in several FL frameworks, including FedML. During a simulation, one MPI process is created per compute node, i.e., per FL participant, and the central server uses collective communications, including *gathering*, *broadcasting*, and *scattering*, to orchestrate transfers. In practice, we distinguish the first round, where the entire model checkpoint initialized

[1]Cedar is a general-purpose computer cluster of the Digital Research Alliance of Canada. Documentation is at https://docs.alliancecan.ca/wiki/Cedar.

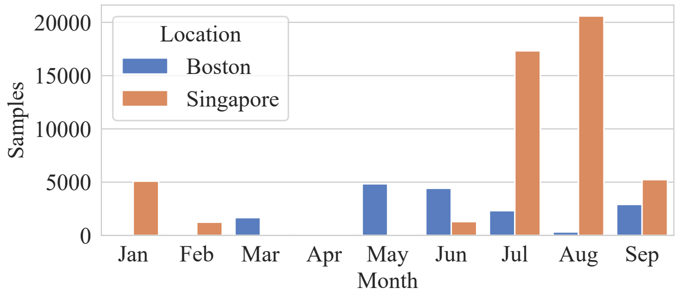

Fig. 3. **Distribution of samples in nuImages training data.** The dataset is composed of nearly 500 driving logs acquired through six different cameras, and features a wide range of driving scenarios and weather conditions

from pre-trained weights must be transmitted to the clients, from the following rounds, where only the transmission of the learnable parameters is required. Furthermore, weights were represented using half-precision (FP16) during transfer to reduce communication overhead.

2) Encryption: The hybrid cryptosystem used to secure the communications between the FL participants was implemented with Python's cryptography package. The key encapsulation scheme is based on RSA [86] with the OAEP [87] padding scheme, while data encapsulation was handled with a 256-bit symmetric key using the Advanced Encryption Standard (AES) algorithm [88] with Galois/Counter Mode (GCM) [89], and with scrypt as the password-based key derivation function [90]. AES-GCM requires the use of a cryptographic nonce for each new encryption to prevent replay attacks. In our scheme, the 12-byte nonce used during each data encryption and the salt used when creating a new symmetric key were all generated with Python's secrets package, to ensure the use of cryptographically strong random numbers. Encrypting with AES-GCM also yields a 16-byte authentication tag, which can be safely transmitted alongside the nonce and ciphertext to ensure the integrity and authenticity of the message.

B. Datasets

We conducted experiments on two relevant autonomous driving datasets: the 2D object detection subset of the KITTI Vision Benchmark Suite [91] and nuImages, an extension of nuScenes dedicated to 2D object detection [92]. KITTI is a pioneering and still widely popular benchmark, which features synchronized stereo RGB images, GPS coordinates, and LiDAR point clouds, and supports 2D and 3D object detection. However, it suffers from low data diversity in terms of weather and lightning conditions, location, and object orientation. The 2D object detection subset of KITTI does not feature a pre-existing train/validation split and contains 7481 labeled images, with the most common dimension being 1242×375 (small variations are present). KITTI contains a *DontCare* class corresponding to regions where objects were not labeled which was excluded from our experiments, thus leaving eight classes in the dataset. nuImages contains almost 500 driving logs of varying length and is organized as a relational database, featuring 67279 labeled training images and a separate predefined validation set of 16445 labeled

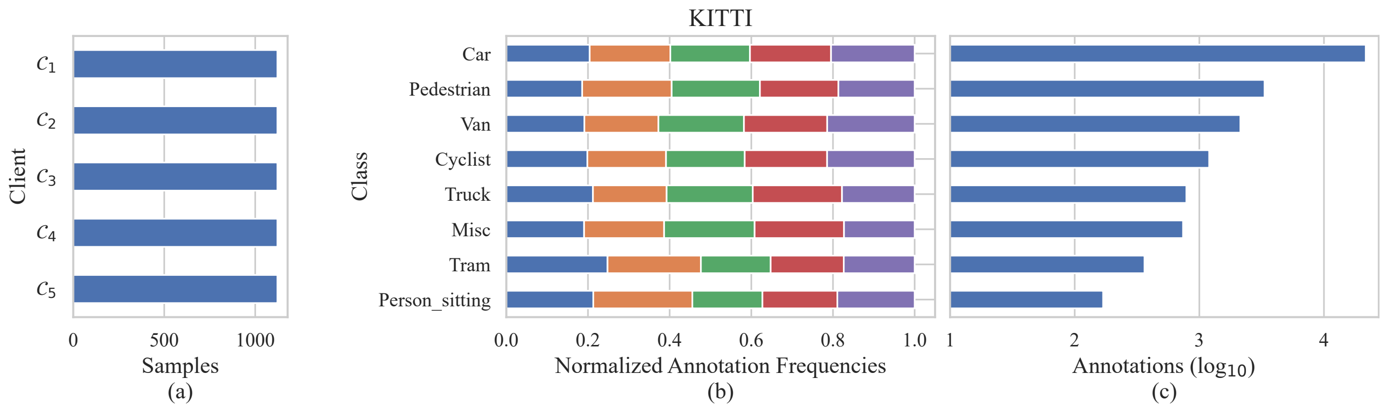

Fig. 4. **KITTI split.** 25% of KITTI training data are stored on the server, while the 75% left are distributed IID among five clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

images, all samples being of size 1600×900. The data were acquired through six cameras oriented to provide a 360-degree view around the vehicle with some small overlaps. The scenes were taken in Boston and Singapore from January to September, as shown in Fig. 3, and feature rain, snow, and nighttime, making nuImages significantly more diverse than KITTI. nuImages also includes 23 different classes spread in a long-tail distribution. We did not consider the predefined non-annotated test sets of KITTI and nuImages in our experiments. In both cases, the bounding boxes are represented with their top left and bottom right absolute coordinates, thus the annotations of the bounding boxes first had to be converted to the YOLO format, where a bounding box is represented with normalized center coordinates, box width, and height.

C. *Data-splitting strategy*

As KITTI is a low-variance dataset, which does not allow for any easy natural semantic separation of the data, it served as our IID setting. We randomly sampled 25% of the training data to store on the central server, while the remaining samples were distributed homogeneously among five clients, as shown in Fig. 4. Regarding nuImages, we created non-IID splits by relying on the spatiotemporal metadata available for the training set to generate unbalancedness, concept drifting, and label distribution skewness. The set held by the server is simply constituted of nuImages's predefined validation data, while the training splits are organized as follows. Clients $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ received the data collected in Boston, respectively, for the months of March and May, June and July, and September, while $\mathcal{C}_4$ received the data collected in Singapore in January and February. Singapore data from June to August, which are overrepresented in the training set, were distributed among clients $\mathcal{C}_5$ to $\mathcal{C}_9$; however, the distribution was not made based on individual samples, but by randomly sampling entire data logs, thus ensuring a minimal threshold of data heterogeneity based on the specificities of each navigation sequence. Finally, client $\mathcal{C}_{10}$ received the data of September from Singapore. Furthermore, we implemented two class maps for nuImages to observe the impact of the dataset's long-tailed distribution on federated optimization. We refer to the dataset resulting from the map replicated from the nuScenes competition, where only ten classes are retained, as nuImages-10, and to the base dataset with all 23 classes as nuImages-23. We note that the aforementioned splitting strategy naturally leads to unbalancedness and label distribution skews, as shown in Fig. 5 and Fig. 6, with the latter being further exacerbated when the full long-tail distribution is included. In particular, among the clients differing only by their navigation sequences, the differences in label distributions only become stark for the rarer classes, thus illustrating how our splits capture heterogeneity at varying granular levels.

D. *Design of experiments*

For the first set of experiments, we implemented FedAvg as a baseline. The local and server-side optimizers were both SGD with a fixed learning rate of respectively 0.01 and 1.

The second FL method, FedOpt, focuses on client-side optimization. The server-side optimizer is still SGD, but we adapted the original YOLOv7 training procedure to the federated setting, with default fine-tuning hyperparameters. Specifically, the model parameters are divided into three groups consisting of the biases, the batch normalization parameters, and the remaining parameters to which weight decay is applied with 0.0005 chosen as the unscaled regularization constant. Each group adopts its own one-cycle learning rate scheduling policy, which includes a linear warm-up period followed by cosine decay annealing. The learning rate is initialized at 0.1 for the bias group and at 0 for the two remaining groups, and all the learning rates evolve towards 0.01 during warm-up, and 0.001 during decay. To maintain synchronicity, we fixed the warm-up duration to be a fixed number of epochs common to all clients (respectively 30 and 15 for KITTI and nuImages). Furthermore, Nesterov momentum is also applied locally in FedOpt, with a starting momentum factor of 0.8 which is linearly increased to 0.937 during warm-up and maintained constant thereafter. We chose not to aggregate the local momentums, thus reducing communication overhead but forcing the clients to maintain a persistent state between rounds of training. A possible improvement to FedPylot would be to implement an additional stateless variant of client-level momentum, to support it irregardless of the participation rate.

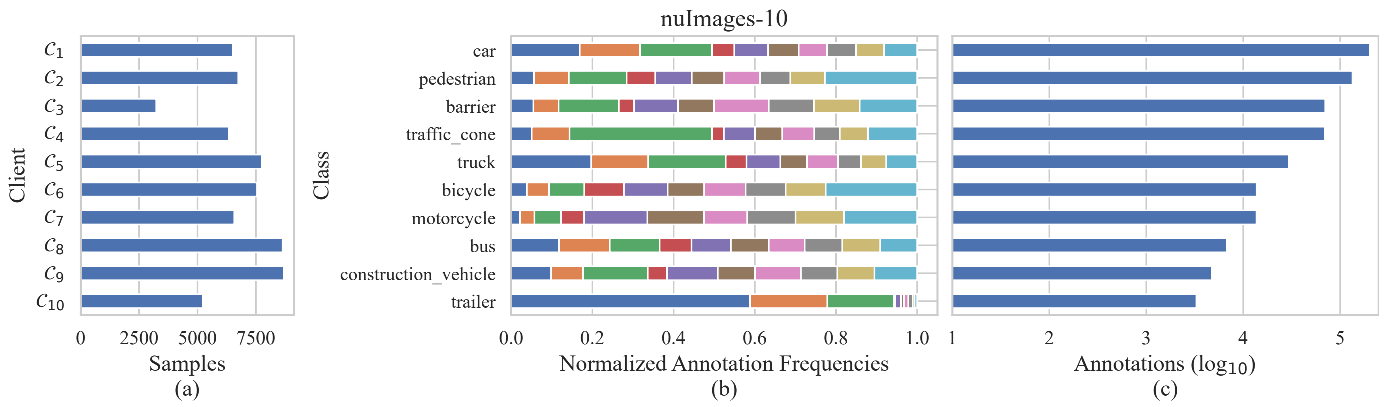

Fig. 5. **nuImages-10 split.** The original classes are mapped to ten labels, and the training data are split non-IID among ten clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

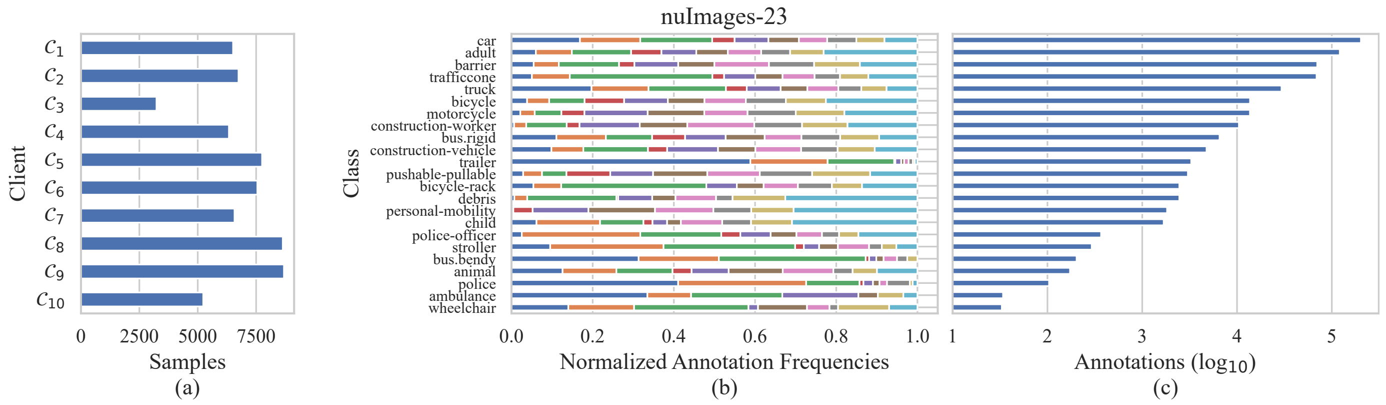

Fig. 6. **nuImages-23 split.** The full long-tail distribution is retained, and the training data are split non-IID among ten clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

Lastly, we performed ablation with the FedAvgM (4) server-side optimizer alongside the local optimization described in FedOpt, a combination which we refer to as FedOptM. We performed a grid search for the server learning rate $\eta \in \{0.5, 1.0, 1.5\}$ as well as the momentum factor $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We restricted η to a small region around its default value, as the local learning rate schedules were left unchanged.

In all instances above, we used the original anchor-based YOLOv7 architecture with weights pre-trained on MS COCO, letterbox resize of 640, a batch size of 32, mosaic and horizontal flipping augmentations with respective probabilities 1.0 and 0.5, a confidence threshold of 0.001, an IoU threshold for NMS of 0.65 on testing data, and gains of respectively 0.05, 0.3 and 0.7 for the box regression, classification, and objectness losses. When applicable, we allowed the learning rates and the momentum factor to evolve within the communication rounds to accommodate the communication constraints in IoV. Local EMA models were not aggregated and the scheduling policies were maintained locally. FedAvg and FedOpt were compared against each other during 150 epochs of training and with rounds of different length, $(R, E) \in \{(30, 5), (15, 10), (10, 15)\}$, whereas for FedOptM we specifically focused on the 30 rounds setting to allow for more server optimization steps. FL was compared against the default centralized procedure, where the model was trained out of the box on all data for 150 epochs. For simplicity, the warm-up length was kept the same as in the federated experiments.

TABLE I
IMPACT OF CLIENT-SIDE OPTIMIZATION ON YOLOV7 TESTING ACCURACY COMPARED AGAINST CENTRALIZED LEARNING

Dataset	Centralized	Federated			
		Method	R30E5	R15E10	R10E15
KITTI	73.7%	FedAvg	57.5%	59.5%	59.2%
		FedOpt	66.3%	67.4%	**68.3%**
nuImages-10	52.9%	FedAvg	45.2%	45.4%	45.2%
		FedOpt	47.5%	**47.8%**	47.3%
nuImages-23	33.2%	FedAvg	26.6%	26.5%	26.2%
		FedOpt	28.2%	**28.3%**	28.1%

VI. Results

A. *Effects of client-side optimization*

The evolution of the training loss and mAP for centralized learning, as well as FedAvg and FedOpt, is shown in Fig. 7, and we report the highest testing accuracy achieved for each setting in Table I. In the federated setting, the aggregated training loss is derived from the weighted average of the local losses computed by each client following (1), and the mAP is

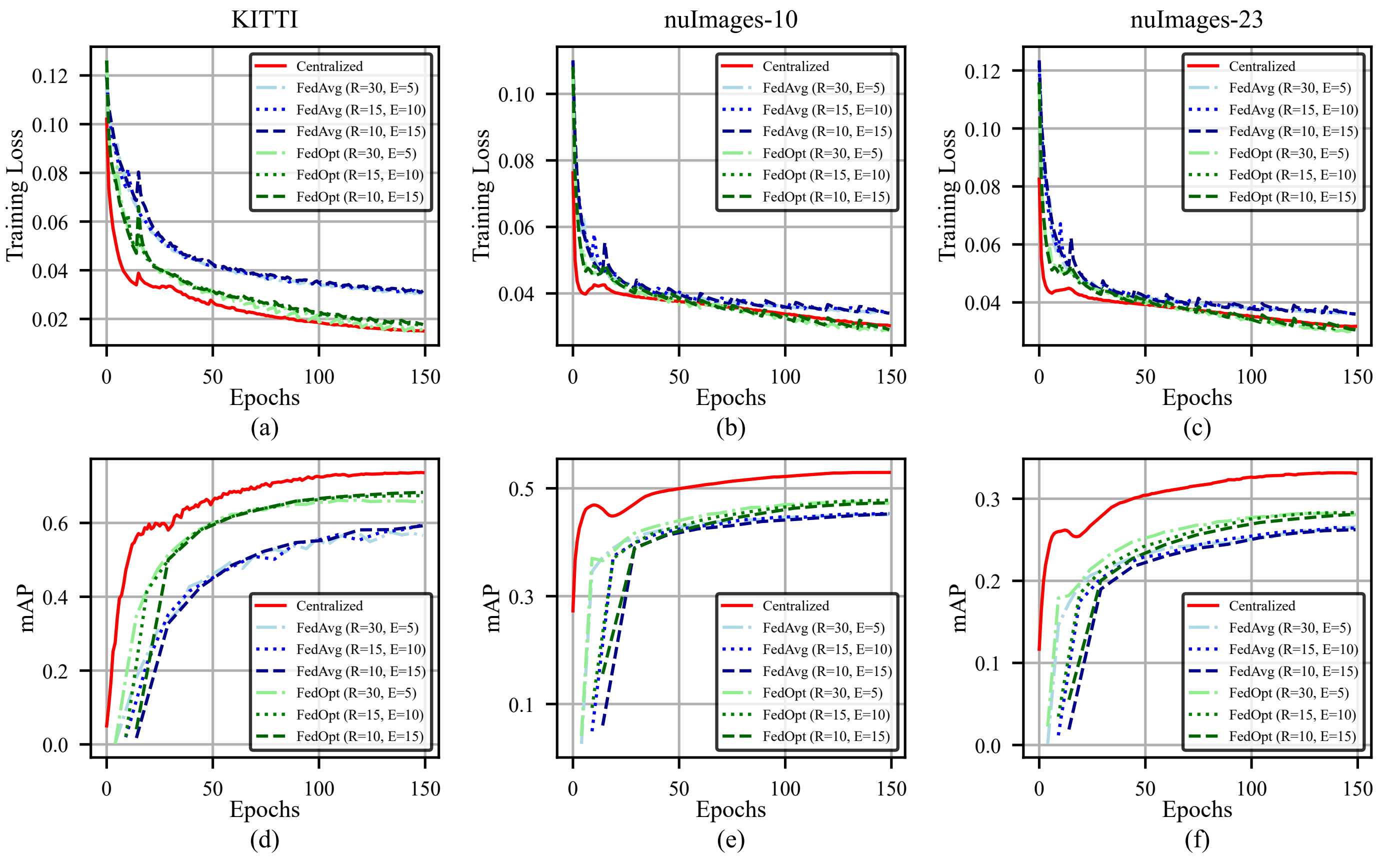

Fig. 7. **Client-side optimization results**. Three round lengths were considered for the federated baseline FedAvg and the locally optimized algorithm FedOpt. We report the evolution for the centralized and federated settings of YOLOv7's (a) training loss on KITTI, (b) training loss on nuImages-10, (c) training loss on nuImages-23, (d) testing accuracy on KITTI, (e) testing accuracy on nuImages-10, and (f) testing accuracy on nuImages-23.

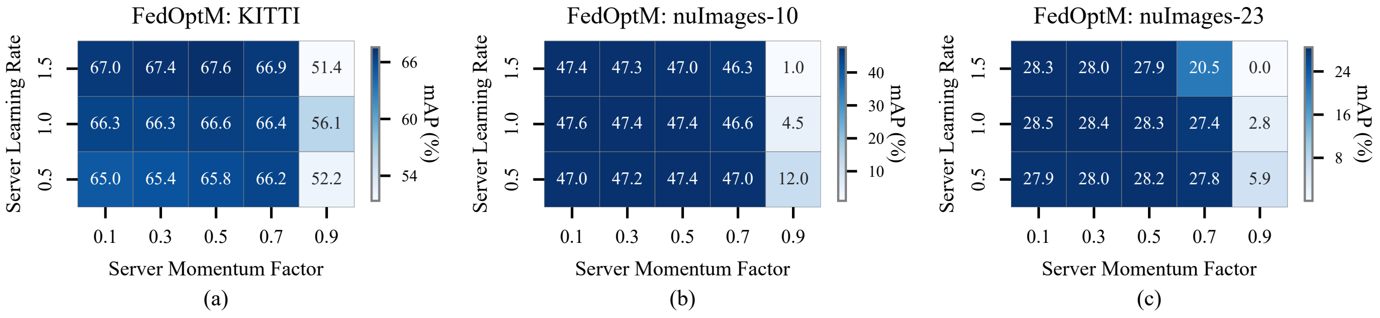

Fig. 8. **Grid search on FedOptM server-side hyperparameters**. Training lasted 30 rounds of 5 epochs on (a) KITTI, (b) nuImages-10, and (c) nuImages-23.

measured by evaluating the global model at the end of each communication round on a separate set of unseen examples stored on the central server, as defined in Algorithm 1. The advanced local optimization scheme resulted in a consistent performance increase across all levels of heterogeneity, but more so on the IID setting, where improved local updates are not countered by the client-drift. Longer training rounds were beneficial on IID data, but also did not lead to degradations in the heterogeneous settings, which we attribute to our choice of splitting strategy and use of pre-trained weights. Performance drops, relatively to the centralized setting, were more notable for nuImages-23 than nuImages-10, confirming that the inclusion of the long-tail distribution adversely impacted federated optimization.

B. *Ablation on server learning rate and momentum*

The full grid search details for FedOptM on the effect of the server learning rate and momentum on testing accuracy are reported in Fig. 8, and we comment on the improvements relative to FedOpt for comparable communication rounds number and length. Training on KITTI being very stable, it benefited from higher server-side learning rates, and the mAP reached 67.6% (+1.3%). Little improvements were observed on nuImages-10 47.6% (+0.1%), but small momentum values led to small but consistent improvements for the more heterogeneous nuImages-23 28.5% (+0.3%). However, large increases in update volume resulting from choosing a high momentum factor, such as 0.9, significantly disrupted the

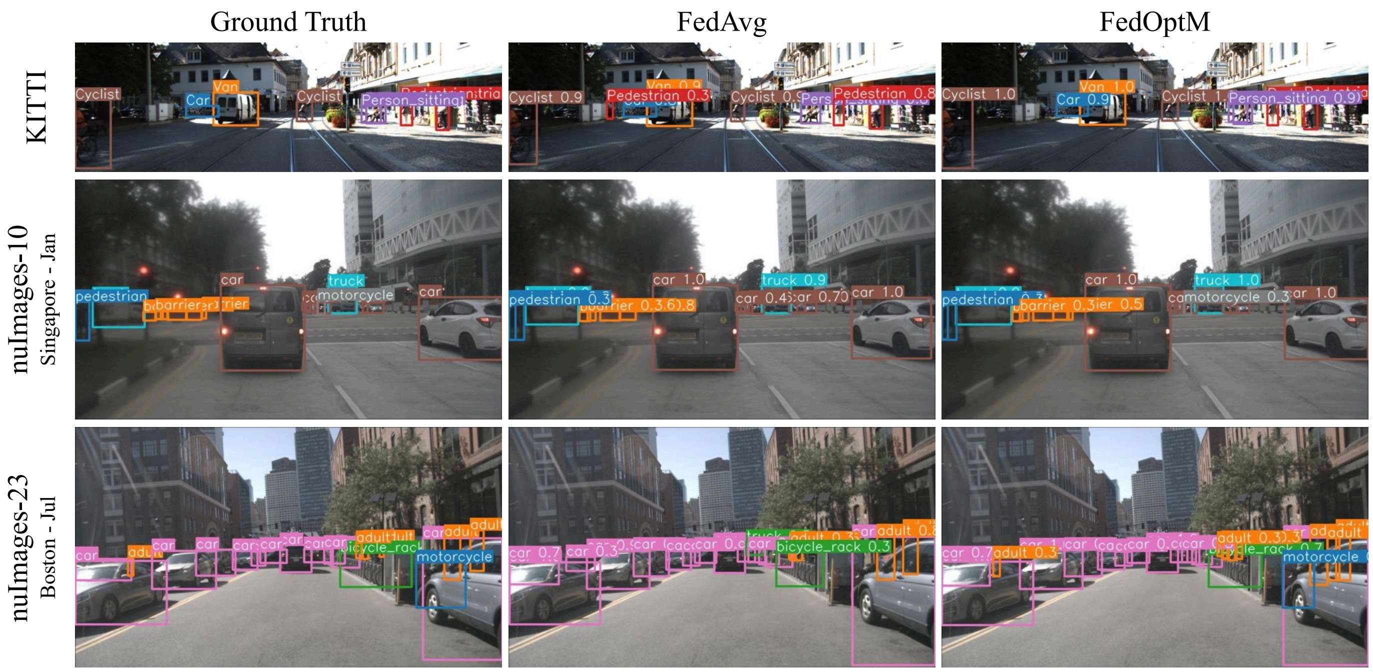

Fig. 9. **Qualitative results of YOLOv7 on testing data.** Images are resized to the input resolution of 640px before inference, while retaining aspect ratio. The model was trained with the FedAvg baseline and the FedOptM optimized procedure on KITTI (IID) and two separate class maps of nuImages (non-IID).

training process and led to inaccurate predictions, especially on non-IID data. A qualitative comparison between FedAvg and FedOptM is shown in Fig. 9.

C. Communication costs and inference speed

We measured the testing accuracy and inference speed of different YOLOv7 variants, as well as the communication overhead resulting from server-client communications during training, on KITTI, nuImages-10 and nuImages-23. We considered the three P5 models (YOLOv7-tiny, YOLOv7, YOLOv7-X) and the base P6 model (YOLOv7-W6), and we report our findings in Table II. The federated algorithm was FedOptM, and we reused the server hyperparameters that produced the best performances during ablation. For YOLOv7-W6, letterbox resizing was increased from 640px to 1280px, while the training batch size was reduced from 32 to 8. Training lasted 30 communication rounds of 5 epochs each, and all other hyperparameters remained as previously established for all model variants. To facilitate comparisons, latency measurements were conducted in a single Colab environment separately from model training, and results were averaged over 20 and 5 runs for KITTI and nuImages, respectively. Similarly to the original benchmarking of YOLOv7, we measured latencies on a V100 in FPS at batch size 1, following re-parameterization and model tracing, but without porting the models to ONNX or TensorRT. Only model inference is considered, thus the pre-processing and post-processing costs (including NMS) are not accounted for. The communication cost is reported in megabytes for exactly one-to-one transmission of the symmetrically encrypted learnable parameters between the server and a client, and during a round, there are twice as many transmissions as there are participating clients.

TABLE II
GENERAL PERFORMANCES FOR SEVERAL YOLOv7 VARIANTS

Dataset	Model	Size	mAP	FPS	Overhead
KITTI	YOLOv7-tiny	640	53.4%	208	12.2MB
	YOLOv7	640	67.6%	142	74.8MB
	YOLOv7-X	640	68.6%	123	142.1MB
	YOLOv7-W6	1280	71.8%	115	162.5MB
nuImages-10	YOLOv7-tiny	640	33.3%	201	12.2MB
	YOLOv7	640	47.6%	138	74.8MB
	YOLOv7-X	640	49.4%	113	142.1MB
	YOLOv7-W6	1280	53.1%	92	162.6MB
nuImages-23	YOLOv7-tiny	640	18.1%	198	12.3MB
	YOLOv7	640	28.5%	136	74.9MB
	YOLOv7-X	640	29.6%	112	142.3MB
	YOLOv7-W6	1280	32.4%	92	163.0MB

The initial broadcast featuring the entire checkpoint of the model has a marginally higher cost than the one reported in the table. Using half-precision during model transfer reduced the communication overhead and is straightforward to implement, and we leave the implementation of more advanced model compression techniques to future work.

VII. CONCLUSION

In this study, we advocate for FL to address the privacy and scalability challenges inherent to IoV. We propose FedPylot, a practical MPI-based client-server prototype that features hybrid encryption to simulate the federated training of modern object detectors on HPC systems. Through a comprehensive experimental analysis conducted on datasets relevant to autonomous driving, where we considered prediction quality, model latency, and communication overhead, we show the potential of FL to realize object detection and address the real-time processing requirements of autonomous vehicles. Our findings highlight the robustness of the federated optimization of YOLOv7 under realistic heterogeneity constraints, and we hope that FedPylot will invite further research to bring state-of-the-art object detection to FL. Possible future improvements that we are considering for FedPylot include adding support for other object detectors, multimodal sensing capabilities, advanced privacy techniques, asynchronous and low-participation-rate configurations, and model personalization.

Acknowledgments

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, for the financial support of this research. Furthermore, the authors express their gratitude to the BC DRI Group and the Digital Research Alliance of Canada, for the computing resources they provided.

References

[1] W. Xu *et al.*, "Internet of vehicles in big data era," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018.

[2] C. R. Storck and F. Duarte-Figueiredo, "A survey of 5G technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by Internet of Vehicles," *IEEE Access*, vol. 8, pp. 117 593–117 614, 2020.

[3] S. Grigorescu *et al.*, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, 2020.

[4] Y. Lu *et al.*, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[5] D. M. Manias and A. Shami, "Making a case for federated learning in the Internet of Vehicles and intelligent transportation systems," *IEEE Netw.*, vol. 35, no. 3, pp. 88–94, May 2021.

[6] X. Zhou *et al.*, "Two-layer federated learning with heterogeneous model aggregation for 6G supported Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5308–5317, Jun. 2021.

[7] J. Janai *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Found. Trends Comput. Graph. Vis.*, vol. 12, no. 1–3, pp. 1–308, Jul. 2020.

[8] K. Xie *et al.*, "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9980–9992, Sep. 2022.

[9] A. A. Padaria *et al.*, "Traffic sign classification for autonomous vehicles using split and federated learning underlying 5G," *IEEE Open J. Veh. Technol.*, vol. 4, pp. 877–892, 2023.

[10] Z. Lian *et al.*, "Traffic sign recognition using optimized federated learning in Internet of Vehicles," *IEEE Internet Things J.*, vol. 11, no. 4, pp. 6722–6729, Feb. 2024.

[11] Y. Yuan *et al.*, "FedRD: Privacy-preserving adaptive federated learning framework for intelligent hazardous road damage detection and warning," *Future Gener. Comput. Syst.*, vol. 125, Dec. 2021.

[12] S. Alshammari and S. Song, "3Pod: Federated learning-based 3 dimensional pothole detection for smart transportation," in *Proc. IEEE Int. Smart Cities Conf. (ISC2)*, Sep. 2022, pp. 1–7.

[13] P. K. Saha *et al.*, "Federated learning–based global road damage detection," *Comput.-Aided Civil Infrastruct. Eng.*, 2024.

[14] L. Fantauzzo *et al.*, "FedDrive: Generalizing federated learning to semantic segmentation in autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2022, pp. 11 504–11 511.

[15] E. Fanì *et al.*, "FedDrive v2: An analysis of the impact of label skewness in federated semantic segmentation for autonomous driving," Oct. 2023. [Online]. Available: http://arxiv.org/abs/2309.13336

[16] G. Rjoub *et al.*, "Improving autonomous vehicles safety in snow weather using federated YOLO CNN learning," in *Int. Conf. Mobile Web Intell. Inform. Syst.*, 2021, pp. 121–134.

[17] Y. Chen *et al.*, "Federated learning with infrastructure resource limitations in vehicular object detection," in *Proc. IEEE/ACM Symp. Edge Comput.*, Dec. 2021, pp. 366–370.

[18] J. R. v. Bommel, "Active learning during federated learning for object detection," Jul. 2021. [Online]. Available: https://essay.utwente.nl/86855/

[19] G. Rjoub *et al.*, "Active federated YOLOR model for enhancing autonomous vehicles safety," in *Mobile Web Intell. Inform. Syst.*, 2022, pp. 49–64.

[20] S. Dai *et al.*, "Online federated learning based object detection across autonomous vehicles in a virtual world," in *Proc. IEEE 20th Consum. Commun. Netw. Conf.*, Jan. 2023, pp. 919–920.

[21] S. Wang *et al.*, "Federated deep learning meets autonomous vehicle perception: Design and verification," *IEEE Netw.*, vol. 37, no. 3, pp. 16–25, May 2023.

[22] F. Chi *et al.*, "Federated semi-supervised learning for object detection in autonomous driving," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Jun. 2023, pp. 1–5.

[23] L. Rao *et al.*, "Sparse federated training of object detection in the Internet of Vehicles," in *IEEE Int. Conf. Commun.*, May 2023, pp. 1768–1773.

[24] S. Su *et al.*, "Cross-domain federated object detection," in *IEEE Int. Conf. Multimedia Expo*, Jul. 2023, pp. 1469–1474.

[25] T. Kim *et al.*, "Navigating data heterogeneity in federated learning: A semi-supervised approach for object detection," in *37th Conf. Neural Inf. Process. Syst.*, Nov. 2023.

[26] T. Zheng *et al.*, "AutoFed: Heterogeneity-aware federated multimodal learning for robust autonomous driving," in *Proc. 29th Annu. Int. Conf. Mobile Comput. Netw.* ACM, Jul. 2023, no. 15, pp. 1–15.

[27] A. Mishra *et al.*, "Swarm learning in autonomous driving: A privacy preserving approach," in *Proc. 15th Int. Conf. Comput. Model. Simul.*, Aug. 2023, pp. 271–277.

[28] F. Chi *et al.*, "Federated cooperative 3D object detection for autonomous driving," in *Proc. IEEE 33rd Int. Workshop Mach. Learn. Signal Process.*, Sep. 2023, pp. 1–6.

[29] A. Khalil *et al.*, "Federated learning with heterogeneous data handling for robust vehicular object detection," May 2024. [Online]. Available: http://arxiv.org/abs/2405.01108

[30] ——, "Driving towards efficiency: Adaptive resource-aware clustered federated learning in vehicular networks," *22nd Mediterranean Commun. Comput. Netw. Conf. IEEE*, Apr. 2024.

[31] D. Jallepalli *et al.*, "Federated learning for object detection in autonomous vehicles," in *Proc. IEEE 7th Int. Conf. Big Data Comput. Service Appl.*, Aug. 2021, pp. 107–114.

[32] M. Han *et al.*, "Federated learning-based trajectory prediction model with privacy preserving for intelligent vehicle," *Int. J. Intell. Syst.*, vol. 37, no. 12, pp. 10 861–10 879, 2022.

[33] R. Yu *et al.*, "Personalized driving assistance algorithms: Case study of federated learning based forward collision warning," *Accident Anal. Prevent.*, vol. 168, p. 106609, Apr. 2022.

[34] C.-Y. Wang *et al.*, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 7464–7475.

[35] J. Terven *et al.*, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, Dec. 2023.

[36] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," Nov. 2020. [Online]. Available: http://arxiv.org/abs/2007.13518

[37] A. Ziller *et al.*, "PySyft: A library for easy federated learning," in *Federated Learn. Syst.* Springer, 2021, pp. 111–139.

[38] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," Mar. 2022. [Online]. Available: http://arxiv.org/abs/2007.14390

[39] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[40] S. J. Reddi *et al.*, "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Representations*, 2021.

[41] A. Z. Tan *et al.*, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.

[42] D. C. Nguyen *et al.*, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12 806–12 825, Aug. 2021.

[43] Q. Yang *et al.*, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, Jan. 2019.

[44] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, Jun. 2021.

[45] T.-M. H. Hsu *et al.*, "Measuring the effects of non-identical data distribution for federated visual classification," Sep. 2019. [Online]. Available: http://arxiv.org/abs/1909.06335

[46] T. Li *et al.*, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.*, pp. 429–450, Mar. 2020.

[47] S. P. Karimireddy *et al.*, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, Nov. 2020, pp. 5132–5143.

[48] Q. Li *et al.*, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 10 713–10 722.

[49] J. Miao *et al.*, "FedSeg: Class-heterogeneous federated learning for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 8042–8052.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017. [Online]. Available: http://arxiv.org/abs/1412.6980

[51] W. Liu *et al.*, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 8, pp. 1754–1766, Aug. 2020.

[52] J. Xu *et al.*, "FedCM: Federated learning with client-level momentum," Jun. 2021. [Online]. Available: http://arxiv.org/abs/2106.10874

[53] H.-Y. Chen *et al.*, "On the importance and applicability of pre-training for federated learning," in *Proc. Int. Conf. Learn. Representations*, Sep. 2022.

[54] J. Nguyen *et al.*, "Where to begin? On the impact of pre-training and initialization in federated learning," in *Proc. Int. Conf. Learn. Representations*, Sep. 2022.

[55] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. 27th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 1986, pp. 162–167.

[56] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, May 2009, pp. 169–178.

[57] C. Dwork *et al.*, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[58] S. S. A. Zaidi *et al.*, "A survey of modern deep learning based object detection models," *Digit. Signal Process.*, vol. 126, p. 103514, Jun. 2022.

[59] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[60] K. He *et al.*, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[61] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[62] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[63] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. on Comput. Vis.*, 2017, pp. 2980–2988.

[64] N. Carion *et al.*, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[65] W. Lv *et al.*, "DETRs beat YOLOs on real-time object detection," Jul. 2023. [Online]. Available: http://arxiv.org/abs/2304.08069

[66] R. Padilla *et al.*, "A survey on performance metrics for object-detection algorithms," in *Proc. Int. Conf. Syst. Signals Image Process*, Jul. 2020, pp. 237–242.

[67] G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[68] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[69] C.-Y. Wang *et al.*, "You only learn one representation: Unified network for multiple tasks," *J. Inf. Sci. Eng.*, vol. 39, no. 3, pp. 691–709, May 2023.

[70] A. Bochkovskiy *et al.*, "YOLOv4: Optimal speed and accuracy of object detection," Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.10934

[71] C.-Y. Wang *et al.*, "Designing network design strategies through gradient path analysis," Nov. 2022. [Online]. Available: http://arxiv.org/abs/2211.04800

[72] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.

[73] Z. Zheng *et al.*, "Distance-IoU loss: Faster and better learning for bounding box regression," *Proc. AAAI Conf. Artif. Intell.*, pp. 12 993–13 000, Apr. 2020.

[74] Z. Ge *et al.*, "YOLOX: Exceeding YOLO series in 2021," Aug. 2021. [Online]. Available: http://arxiv.org/abs/2107.08430

[75] C. Li *et al.*, "YOLOv6 v3.0: A full-scale reloading," Jan. 2023. [Online]. Available: http://arxiv.org/abs/2301.05586

[76] G. Jocher *et al.*, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[77] C.-Y. Wang *et al.*, "YOLOv9: Learning what you want to learn using programmable gradient information," Feb. 2024. [Online]. Available: http://arxiv.org/abs/2402.13616

[78] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.

[79] S. Ren *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[80] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[81] M. Simon *et al.*, "Complex-YOLO: An Euler-region-proposal for real-time 3D object detection on point clouds," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2019, pp. 197–209.

[82] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[83] M. S. Birrittella *et al.*, "Intel Omni-Path Architecture: Enabling scalable, high performance fabrics," in *Proc. IEEE 23rd Annu. Symp. High-Perform. Interconnects*, Aug. 2015, pp. 1–9.

[84] E. Gabriel *et al.*, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proc. 11th Eur. PVM/MPI Users' Group Meeting*, Budapest, Hungary, Sep. 2004, pp. 97–104.

[85] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *IEEE Comput. Sci. Eng.*, vol. 23, no. 4, pp. 47–54, Jul. 2021.

[86] R. L. Rivest *et al.*, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[87] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Proc. EUROCRYPT'94*, 1995, pp. 92–111.

[88] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999. [Online]. Available: https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf

[89] D. McGrew and J. Viega, "The Galois/counter mode of operation (GCM)," *NIST Modes of Operation Process*, 2004. [Online]. Available: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf

[90] C. Percival, "Stronger key derivation via sequential memory-hard functions," Jan. 2009. [Online]. Available: https://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf

[91] A. Geiger *et al.*, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[92] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 621–11 631.