Titre: Title:	Méthodes d'apprentissage fédéré pour la détection d'objets en temps réel dans l'internet des véhicules
Auteur: Author:	Cyprien Quéméneur
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Quéméneur, C. (2024). Méthodes d'apprentissage fédéré pour la détection d'objets en temps réel dans l'internet des véhicules [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59434/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/59434/
Directeurs de recherche: Advisors:	Soumaya Cherkaoui
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Méthodes d'apprentissage fédéré pour la détection d'objets en temps réel dans l'Internet des véhicules

CYPRIEN QUÉMÉNEUR
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Méthodes d'apprentissage fédéré pour la détection d'objets en temps réel dans l'Internet des véhicules

présenté par **Cyprien QUÉMÉNEUR**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Alejandro QUINTERO, président
Soumaya CHERKAOUI, membre et directrice de recherche
Omar ABDUL WAHAB, membre

DÉDICACE

À tous ceux qui ont nourri ma passion pour les sciences.

REMERCIEMENTS

En premier lieu, je voudrais témoigner de ma reconnaissance à ma directrice de recherche, Prof. Soumaya Cherkaoui, pour avoir encadré mes travaux et m'avoir apporté son expertise durant toute ma maîtrise en génie informatique au sein du laboratoire Lincs. J'adresse également mes remerciements au Conseil de recherches en sciences naturelles et en génie du Canada pour le soutien financier apporté à ma recherche, ainsi qu'à l'Alliance de recherche numérique du Canada et au BC DRI Group pour les ressources computationnelles qui ont été mises à ma disposition. Je remercie enfin tous les membres de Lincs et de la communauté de Polytechnique Montréal pour m'avoir apporté un environnement accueillant et stimulant, m'ayant permis de concevoir mon premier article de recherche.

RÉSUMÉ

L'Internet des véhicules émerge comme une technologie essentielle pour la conduite autonome et les systèmes de transport intelligents, en permettant le traitement de données volumineuses à faible latence dans un vaste réseau interconnecté incluant véhicules, infrastructures, piétons et le nuage. Les véhicules autonomes, qui dépendent largement de modèles d'apprentissage automatique, spécifiquement de réseaux de neurones, peuvent fortement bénéficier des riches données sensorielles générées dynamiquement à la périphérie du réseau pour améliorer leurs performances. L'utilisation de ces données exige toutefois de pouvoir concilier l'entraînement des modèles avec la préservation de la confidentialité des données sensibles des utilisateurs. L'apprentissage fédéré, technique où un ensemble de clients entraînent collectivement un modèle d'apprentissage automatique sans échange de données locales, apparait comme une solution prometteuse pour protéger la vie privée des usagers de la route tout en réduisant le coût des communications dans les réseaux véhiculaires. Néanmoins, en dépit de la popularité croissante de l'apprentissage fédéré, les différents cadriciels y étant consacrés se limitent généralement à la classification d'images. Subséquemment, les tâches plus avancées et critiques pour la conduite autonome, comme la détection d'objets, ont été pour bonne partie ignorées. Dans ce mémoire, nous introduisons FedPylot, un prototype d'apprentissage fédéré simulant l'entraînement de modèles de détection d'objets en temps réel par des véhicules autonomes et connectés reposant sur l'Internet des véhicules. Notre simulateur adopte une architecture client-serveur conçue pour être exécutée sur des systèmes de calcul à haute performance et intègre un chiffrement hybride pour sécuriser les communications entre le serveur et les clients véhiculaires. Nous examinons en particulier l'optimisation fédérée du modèle de l'état de l'art YOLOv7, dans un contexte marqué par l'hétérogénéité statistique, dont dérive conceptuelle, asymétrie dans la distribution des labels et déséquilibre des jeux de données locaux, en incluant toutefois dans nos simulations les hypothèses simplificatrices de persistance, synchronicité et pleine participation des clients. En outre, nous intégrons dans notre évaluation non seulement les performances prédictives de plusieurs variantes de YOLOv7, mais également le coût associé aux communications et la vitesse d'inférence des modèles, et ce, afin de présenter une approche équilibrée des défis auxquels sont confrontés les véhicules autonomes. Nous démontrons des résultats prometteurs pour l'applicabilité de l'apprentissage fédéré dans l'Internet des véhicules, y compris pour réaliser des tâches avancées comme la détection d'objets en temps réel. Nous espérons que FedPylot encouragera de plus amples recherches dans cette direction, et sera enrichi à l'avenir de nouvelles fonctionnalités se rapportant à la sécurité, à la diversité des scénarios envisagés ou encore à l'efficacité des communications.

ABSTRACT

The Internet of Vehicles is emerging as an essential technology for autonomous driving and intelligent transportation systems, by enabling low-latency processing of voluminous data in a vast interconnected network comprising vehicles, infrastructures, pedestrians, and the cloud. Autonomous vehicles, which rely heavily on machine learning models, specifically neural networks, can benefit greatly from the rich sensory data dynamically generated at the network edge to improve their performances. However, the use of such data requires the ability to reconcile model training with preserving the confidentiality of sensitive user data. Federated learning, a technique where a set of clients collectively train a shared machine learning model without exchanging their local data, appears to be a promising solution for protecting the privacy of road users while reducing the cost of communications in vehicular networks. Nevertheless, despite the growing popularity of federated learning, the various frameworks devoted to it are generally limited to image classification. As a result, more advanced tasks critical to autonomous driving, such as object detection, have largely been ignored. In this thesis, we introduce FedPylot, a federated learning prototype simulating the training of real-time object detection models by autonomous and connected vehicles relying on the Internet of Vehicles. Our simulator adopts a client-server architecture designed to run on high-performance computing systems and incorporates hybrid encryption to secure communications between the server and vehicular clients. In particular, we examine the federated optimization of the state-of-the-art YOLOv7 model, in a context marked by statistical heterogeneity, including concept drift, label distribution skews and local dataset imbalance, while including in our simulations the simplifying assumptions of persistence, synchronicity, and full client participation. In addition, we integrate into our evaluation not only the predictive performance of several variants of YOLOv7, but also the costs associated with communications and the speed of model inference, in order to present a balanced approach to the challenges facing autonomous vehicles. We demonstrate promising results for the applicability of federated learning in Internet of Vehicles, including for performing advanced tasks such as real-time object detection. We hope that FedPylot will encourage further research in this direction and will be enriched in the future with new functionalities relating to safety, the diversity of scenarios supported, and communication efficiency.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AES	Advanced Encryption Standard
AUC	Area Under Curve
BCE	Binary Cross-Entropy
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DETR	Detection Transformer
ELAN	Efficient Layer Aggregation Network
EMA	Exponential Moving Average
FL	Federated Learning
FP16	Half-precision
FPN	Feature Pyramid Network
FPS	Frames-Per-Second
GCM	Galois/Counter Mode
GELAN	Generalized Efficient Layer Aggregation Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
HPC	High-Performance Computing
I/O	Input and Output
IID	Independent and Identically Distributed
IoU	Intersection over Union
IoV	Internet of Vehicles
ITS	Intelligent Transportation Systems
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
mAP	Mean Average Precision
MiB	Mebibyte
ML	Machine Learning
MPI	Message Parsing Interface
NMS	Non-Maximum Suppression
NSERC	Natural Sciences and Engineering Research Council of Canada
OAEP	Optimal Asymmetric Encryption Padding
ONNX	Open Neural Network Exchange

OTA	Optimal Transport Assignment
pFL	Personalized Federated Learning
PGI	Programmable Gradient Information
R-CNN	Region-based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RGPD	Règlement Général sur la Protection des Données
RSA	Rivest–Shamir–Adleman
SGD	Stochastic Gradient Descent
SiLU	Sigmoid Linear Unit
SPP	Spatial Pyramid Pooling
SSD	Single Shot MultiBox Detector
SSH	Secure Shell
TLS	Transport Layer Security
V2X	Vehicle-to-Everything
ViT	Vision Transformer
YOLO	You Only Look Once
YOLOR	You Only Learn One Representation

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Motivations

Les progrès en intelligence artificielle et dans les technologies de communication véhiculaire favorisent l'essor de l'Internet des véhicules. Celui-ci rassemble les véhicules, les piétons, les infrastructures de la route et le nuage informatique, dans un vaste réseau à même de tirer profit des mégadonnées qu'il génère [1]. L'Internet des véhicules, à son tour, facilite le développement des systèmes de transport intelligents et de la conduite autonome, qui sont appelés à remodeler la mobilité en rationalisant le trafic, en améliorant la sécurité, et en réduisant les émissions des véhicules et la consommation de carburant. Toutefois, si les véhicules connectés peuvent compter sur le partage d'information à longue portée, à faible latence, fiable et sécurisé permis par l'Internet des véhicules [2] pour améliorer leur perception contextuelle, leur capacité à résoudre des tâches de navigation complexes demeure largement dépendante de modèles d'apprentissage profond [3]. De même, si les véhicules ont certes accès à un large ensemble de capteurs multimodaux, incluant caméras, LiDAR, radars et systèmes GPS, pour collecter les données nécessaires à l'entraînement de réseaux de neurones, la transmission de celles-ci vers le nuage soulève des problèmes de confidentialité et de bande passante.

L'apprentissage fédéré est un paradigme émergent en apprentissage automatique, où l'entraînement des modèles a pour spécificité de reposer sur la collaboration d'un ensemble de clients ne partageant pas leurs données locales. En effet, le partage des données brutes dont disposent les clients est prohibé en apprentissage fédéré, et ces derniers entraînent alors localement leurs modèles d'apprentissage respectifs et transmettent régulièrement des mises à jour à un agent centralisé, responsable de leur agrégation, du calcul d'un modèle global, et de la dissémination de celui-ci. L'apprentissage fédéré permet ainsi de mieux protéger la vie privée des utilisateurs et peut, par la même occasion, simplifier l'adhésion aux législations comme RGPD [4]. De plus, l'absence de transmission de données brutes résulte en une réduction de la charge sur le réseau et favorise les applications exigeant une faible latence. Enfin, en intégrant un très grand nombre de clients dans le processus d'entraînement, l'apprentissage fédéré peut améliorer la convergence des modèles par rapport à l'apprentissage centralisé, en tirant parti de la vaste puissance de calcul des clients, de la parallélisation augmentée du problème, et d'une quantité importante de données très diversifiées. Les champs d'application identifiés comme particulièrement prometteurs pour l'apprentissage fédéré incluent l'Internet des objets [5], y compris l'Internet des véhicules [6, 7], et le domaine médical [8].

Dans le contexte des réseaux véhiculaires, les clients du système fédéré sont principalement les véhicules, tandis que le serveur d'agrégation est positionné stratégiquement à la périphérie du réseau afin de minimiser les temps de latence. De surcroît, un ensemble de tels serveurs périphériques, dispersés dans une région géographique donnée, peuvent subséquemment se rabattre sur une plateforme infonuagique afin de faciliter l'orchestration du système fédéré ou fournir une seconde couche d'agrégation [9].

Les véhicules autonomes, pour naviguer de façon sécuritaire dans des environnements réels, marqués par leurs caractères complexes et dynamiques, reposent en partie sur des modules de vision en temps réel avancés, dont les capacités doivent rivaliser avec la perception humaine [10]. Améliorer les capacités de vision des véhicules en exploitant les avantages de l'apprentissage fédéré est un domaine de recherche en plein essor, ayant produit des résultats encourageants pour la reconnaissance des panneaux signalétiques [11–13], l'identification des nids-de-poule et autres endommagements de la route [14–16], la segmentation sémantique [17,18], la détection d'objets [19–34], la prédiction de trajectoire [35] et l'évitement des collisions [36]. Toutefois, les études susmentionnées ont une applicabilité limitée par le fait qu'elles ont généralement recouru à des modèles d'apprentissage inadaptés en pratique au contexte de la vision en temps réel, ou bien à des architectures pertinentes, mais aujourd'hui techniquement dépassées ou trop petites pour tirer pleinement parti des ressources de calcul des véhicules. Ces limitations peuvent s'expliquer par le manque de cadriciels d'apprentissage fédéré proposant des solutions clé en main pour la détection d'objets. En effet, les cadriciels les plus largement reconnus, tels que Tensorflow Federated [37], FedML [38], PySyft [39], Flower [40] ou Fate [41], pour en nommer quelques-uns, proposent un large ensemble de fonctionnalités, plus ou moins tournées vers la recherche ou le contexte industriel. Cependant, ils ont à l'origine été conçus pour la classification, et l'intégration de bibliothèques tierces nécessaires à la réalisation de tâches avancées de vision est, à ce jour, peu développée.

1.2 Contributions

Dans ce mémoire, nous portons notre attention sur le modèle de détection d'objets en temps réel YOLOv7 [42], soit l'une des dernières entrées dans cette série de modèles, autour duquel nous construisons notre prototype d'apprentissage fédéré open source FedPylot. Celui-ci répond aux difficultés à intégrer des modèles complexes, tels que les détecteurs d'objets de pointe, dans les cadriciels d'apprentissage fédéré les plus populaires, tout en maintenant la possibilité de mener des expérimentations sur un grand nombre de nœuds de calcul. Nous proposons ainsi une alternative flexible et simple à prendre en mains d'apprentissage fédéré pour la détection d'objets, à l'égard de la communauté académique et des enthousiastes.

Nous enrichissons également notre recherche en nous confrontant au problème de l'hétérogénéité statistique en explorant l'optimisation fédérée de YOLOv7 dans un cadre où les clients véhiculaires, et donc les sources de données, sont séparables selon leurs contextes spatio-temporels respectifs.

Nos contributions peuvent être résumées comme suit :

— Nous présentons FedPylot, un simulateur d'apprentissage fédéré pour systèmes de calcul à haute performance, dédié à l'optimisation de détecteur d'objets en temps réel. Nous protégeons, par ailleurs, les communications entre les participants par un système cryptographique hybride.

— À notre connaissance, nous proposons le premier programme permettant des expériences d'apprentissage fédéré à grande échelle avec YOLOv7. Dans notre évaluation. nous avons tenu compte des performances prédictives et de la vitesse d'inférence des modèles utilisés, ainsi que du coût des communications qu'engendre l'entraînement. En outre, nous avons mis l'accent sur l'optimisation locale de YOLOv7 et avons expérimenté sur les hyperparamètres de l'agrégateur côté serveur, en particulier le moment et le taux d'apprentissage.

— Nous validons FedPylot sur deux jeux de données de conduite autonome et simulons une hétérogénéité statistique réaliste, en fractionnant les données selon leur origine spatio-temporelle et les séquences de navigation dont elles sont issues. En particulier, nous nous sommes intéressés aux questions de dérive conceptuelle, d'asymétrie dans la distribution des labels et de déséquilibre des jeux de données. Nous avons enfin capturé l'hétérogénéité à différents niveaux de granularité en faisant varier l'étiquetage d'un jeu de données, dont à l'origine la fréquence des annotations suit une distribution marquée de loi de puissance.

1.3 Plan du mémoire

Le reste de ce mémoire est organisé de la façon suivante. Le Chapitre 2 présente les concepts fondamentaux de l'apprentissage fédéré et de la détection d'objets et comprend un état de l'art portant sur l'application conjointe de ces disciplines aux véhicules autonomes. Le Chapitre 3, quant à lui, détaille certains aspects méthodologiques complémentaires à l'article que nous avons soumis pour publication. Ce dernier est inclus dans le Chapitre 4, où il reprend plus brièvement toutes les autres sections de ce mémoire et introduit notre plan d'expériences, nos résultats et leur interprétation. Enfin, le Chapitre 5 conclut ce mémoire et présente plusieurs pistes d'améliorations futures. Des annexes additionnelles sont également disponibles en fin de ce document.

CHAPITRE 2 REVUE DE LITTÉRATURE

Nous introduisons dans les deux premières sous-sections de ce chapitre les notions en apprentissage fédéré et en détection d'objets nécessaires à la compréhension de notre travail, ainsi que des informations complémentaires permettant de mieux contextualiser comment notre contribution s'inscrit dans la littérature existante. La dernière sous-section est un état de l'art spécifiquement consacré aux travaux antérieurs portant sur le même sujet de recherche.

2.1 Apprentissage fédéré

2.1.1 Formalisation

Principes de base

Nous présentons tout d'abord la formulation classique de l'apprentissage fédéré, introduite pour la première fois par McMahan et al. [43] en 2016. L'apprentissage fédéré cherche à résoudre les problèmes de confidentialité traditionnellement rencontrés dans le contexte de l'apprentissage automatique. Supposons m clients $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m\}$, disposant chacun de leur propre jeu de données local $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_m\}$ de tailles respectives $\{n_1, n_2, ..., n_m\}$. Posons également $n = \sum_{i=1}^{m} n_i$, le nombre total d'échantillons. L'approche conventionnelle, que l'on qualifiera de centralisée, pour entraîner un modèle d'apprentissage supervisé sur ces données, exige de transmettre toutes les données locales vers une unique plateforme. À l'inverse, en apprentissage fédéré, les clients n'échangent aucune donnée brute et entraînent chacun leur modèle localement. On fait alors intervenir un serveur central $\mathcal{S}$, dont le but, au cours de multiples cycles de communication, est de former un modèle commun w, obtenu en agrégeant les poids des modèles locaux. La fonction objective locale, possiblement non convexe, du client $\mathcal{C}_i$ est le risque empirique

$$L_i(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell_i(x, y; w)]. \tag{2.1}$$

L'optimisation fédérée consiste alors à minimiser la somme de ces fonctions, pondérées par la taille des jeux de données locaux

$$\min_w \mathcal{L}(w) = \sum_{i=1}^{m} \frac{n_i}{n} L_i(w). \tag{2.2}$$

En pratique, on suppose que les différents clients sont des appareils périphériques, comme une flotte de téléphones ou de véhicules, ou bien encore des institutions du même type,

comme des hôpitaux qui chercheraient à collaborer tout en préservant la vie privée de leurs patients. Les données sont partitionnées horizontalement, c'est-à-dire que les jeux de données locaux diffèrent par leurs échantillons, mais partage le même espace de caractéristiques (ou *features*). Suivant la formalisation introduite par Yang et al. [44], où $\mathcal{X}$, $\mathcal{Y}$ et $\mathcal{I}$ désignent respectivement l'espace des caractéristiques, l'espace des labels et l'espace des identifiants des échantillons, l'apprentissage fédéré horizontal correspond au cas où les données d'entraînement sont distribuées entre les partis de la manière suivante :

$$\forall(\mathcal{D}_i, \mathcal{D}_j), i \neq j \Rightarrow \mathcal{X}_i = \mathcal{X}_j,\ \mathcal{Y}_i = \mathcal{Y}_j,\ \mathcal{I}_i \neq \mathcal{I}_j. \tag{2.3}$$

Algorithme d'optimisation FedAvg

L'algorithme d'optimisation fédérée initialement introduit par McMahan et al. se nomme FedAvg. Celui-ci est toujours largement utilisé et nous le présentons ci-après. La première étape est celle de l'initialisation du modèle global, qui est effectuée par le serveur central, ou bien de manière aléatoire ou à partir de poids pré-entraînés. Le serveur transmet alors le modèle aux clients et l'entraînement commence. Pour que l'optimisation fédérée puisse converger, il est essentiel que tous les clients partent des mêmes poids. Il s'ensuit R cycles de communication, durant lesquels les clients participent collectivement au processus d'optimisation fédérée. Au début du cycle de communication t, un sous-ensemble de clients, dont les indices forment un ensemble K, est échantillonné aléatoirement pour participer à ce cycle. Ainsi tous les clients ne participent typiquement pas à tous les cycles de communication. Les clients sélectionnés reçoivent le modèle global w^t du serveur et effectuent E époques d'entraînement sur leurs jeux de données respectifs avec l'algorithme de descente stochastique du gradient. Le taux d'apprentissage local η_l et la taille des lots B sont supposés communs pour tous les clients. À la fin du cycle de communication, le serveur $\mathcal{S}$ collecte les modèles locaux w_i^t entraînés par chaque client échantillonné, et les agrège pour former le modèle joint w^{t+1}, par simple moyenne pondérée. La totalité de la procédure est résumée dans l'Algorithme 1. Reste enfin le problème de l'évaluation du modèle global. Lorsqu'un jeu de validation de bonne qualité est maintenu par le serveur, et que ce dernier dispose d'une puissance de calcul suffisante, alors le modèle global peut-être testé côté serveur après chaque étape d'agrégation. En pratique, ces conditions n'étant pas toujours réunies, il demeure possible d'évaluer le modèle global séparément sur les jeux de validations locaux de plusieurs clients, puis de collecter et de moyenner les statistiques résultantes.

Algorithm 1 FedAvg – adapté de [43]

Serveur $\mathcal{S}$ exécute :
  Initialiser le modèle w^0
  pour chaque cycle $t = 0, ..., R-1$ **faire**
    Échantillonner un sous-ensemble de clients d'indices dans K
    pour chaque $\mathcal{C}_i$ si $i \in K$ **en parallèle faire**
      $w_i^t = \textbf{MiseAJourClient}(w^t)$
    fin pour
    $n_t = \sum_{i \in K} n_i$
    $w^{t+1} = \sum_{i \in K} \frac{n_i}{n_t} w_i^t$
  fin pour
MiseAJourClient(w^t) **:** *// exécuter sur le client $\mathcal{C}_i$*
  $w_i^t = w^t$
  $\mathcal{B}_i =$ Diviser $\mathcal{D}_i$ en lots de taille B
  pour chaque époque $e = 1, ..., E$ **faire**
    pour chaque lot $b \in \mathcal{B}_i$ **faire**
      $g_i^t = \nabla \ell(b; w_i^t)$
      $w_i^t = w_i^t - \eta_l g_i^t$
    fin pour
  fin pour
  retourner w_i^t au serveur

Généralisation de FedAvg

Nous présentons ici une généralisation de FedAvg proposée par Reddi et al. [45], intitulée FedOpt, qui reformule l'agrégation centrale comme un problème d'optimisation. En effet, en supposant qu'à la fin du cycle de communication t, que $\mathcal{C}_i$, plutôt que de communiquer directement le modèle w_i^t, transmette plutôt un vecteur de mise à jour $\Delta_i^t = w^t - w_i^t$, le serveur peut alors former une mise à jour globale en combinant ces vecteurs locaux

$$\Delta^t = \sum_{i \in K} \frac{n_i}{n_t} \Delta_i^t. \tag{2.4}$$

Le vecteur Δ^t, que l'on nommera alors le *pseudo-gradient*, peut-être passé à n'importe quel optimiseur `OPTSERVEUR` pour actualiser le modèle global. En particulier, l'agrégation de FedAvg est équivalente à opérer une descente du gradient stochastique sur w^t et Δ^t avec un taux d'apprentissage côté serveur η de 1, comme suit :

$$w^{t+1} = \texttt{SGD}(w^t, \Delta^t, \eta = 1) = w^t - \Delta^t = \sum_{i \in K} \frac{n_i}{n_t} w_i^t. \tag{2.5}$$

De plus, afin de strictement généraliser FedAvg, les optimiseurs locaux dans FedOpt ne sont plus nécessairement supposés être la descente stochastique du gradient. On la remplace donc par un optimiseur générique `OPTCLIENT`. La procédure complète est résumée dans l'Algorithme 2.

Algorithm 2 FedOpt – adapté de [45]

Serveur $\mathcal{S}$ exécute :
  Initialiser le modèle w^0
  pour chaque cycle $t = 0, ..., R-1$ **faire**
    Échantillonner un sous-ensemble de clients d'indices dans K
    pour chaque $\mathcal{C}_i$ si $i \in K$ **en parallèle faire**
      $\Delta_i^t = \textbf{MiseAJourClient}(w^t, t)$
    fin pour
    $n_t = \sum_{i \in K} n_i$
    $\Delta^t = \sum_{i \in K} \frac{n_i}{n_t} \Delta_i^t$
    $w^{t+1} = \texttt{OPTSERVEUR}(w^t, \Delta^t, \eta, t)$
  fin pour
MiseAJourClient(w^t, t) **:** *// exécuter sur le client $\mathcal{C}_i$*
  $w_i^t = w^t$
  $\mathcal{B}_i =$ Diviser $\mathcal{D}_i$ en lots de taille B
  pour chaque époque $e = 1, ..., E$ **faire**
    pour chaque lot $b \in \mathcal{B}_i$ **faire**
      $g_i^t = \nabla \ell(b; w_i^t)$
      $w_i^t = \texttt{OPTCLIENT}(w_i^t, g_i^t, \eta_l, t)$
    fin pour
  fin pour
  $\Delta_i^t = w^t - w_i^t$
  retourner Δ_i^t au serveur

2.1.2 Extensions

Apprentissage fédéré vertical

En complément de l'apprentissage fédéré horizontal, il est également possible de partitionner verticalement les données d'entraînement. Dans ce cas, les jeux de données des clients partagent le même espace échantillon, mais diffèrent dans l'espace des caractéristiques. Toujours selon la même formalisation, on a alors

$$\forall (\mathcal{D}_i, \mathcal{D}_j), i \neq j \Rightarrow \mathcal{X}_i \neq \mathcal{X}_j,\ \mathcal{Y}_i \neq \mathcal{Y}_j,\ \mathcal{I}_i = \mathcal{I}_j. \tag{2.6}$$

L'exemple donné par Yang et al. est celui-ci d'une banque et d'une entreprise de commerce en

ligne opérant dans la même ville. Les utilisateurs de ces institutions seront semblablement les mêmes. Cependant, les informations collectées seront très différentes : revenu, comportement d'achat et historique de crédit pour l'une, historique de navigation et d'achat pour l'autre. Subséquemment, il est possible pour ces deux institutions d'agréger ces caractéristiques pour entraîner un modèle commun qui prédirait le comportement d'achat des utilisateurs.

Apprentissage fédéré personnalisé

L'apprentissage fédéré personnalisé cherche à s'adapter au contexte spécifique de chaque client, en permettant aux modèles locaux de ces derniers de conserver un degré de liberté par rapport au modèle global. Plusieurs techniques peuvent contribuer à atteindre cet objectif, tels que le découplage des paramètres des modèles locaux du modèle global ou encore la distillation de connaissances [46].

Apprentissage fédéré avec chaîne de blocs

La chaîne de blocs est une technologie originellement introduite pour le système BitCoin pour garantir la sécurité, l'immutabilité et la transparence de données numériques sans recourir à une autorité centrale, et ce grâce à des méthodes cryptographiques et des mécanismes de consensus. Dans le contexte de l'apprentissage fédéré, une chaîne de blocs peut être utilisée pour stocker les mises à jour émises par les clients, supprimant alors le besoin de recourir à un serveur central pour l'agrégation, et ainsi améliorer la sécurité et la robustesse du système et mieux protéger la confidentialité des données [47]. La chaîne de blocs peut également faciliter la création de facteurs incitatifs pour encourager les clients avec le plus de données de participer à l'apprentissage fédéré.

Apprentissage fédéré hiérarchique

On désigne par apprentissage fédéré hiérarchique tout système fédéré multicouche faisant intervenir plusieurs agents d'agrégation intermédiaires et un serveur global. Nous avons déjà mis en avant une telle architecture en Introduction, puisque cette organisation est particulièrement pertinente dans le contexte de l'Internet des véhicules. Par simplicité, dans la suite ce mémoire, on qualifiera également de hiérarchique tout système fédéré où des clients regroupés en grappes produisent conjointement un unique modèle au sein de leur grappe respective, et ce même sans introduction d'un serveur intermédiaire, par exemple, via des méthodes de perception coopérative, dans le cas où les clients seraient des véhicules.

2.1.3 Hétérogénéité statistique

Contexte

On désigne par hétérogénéité statistique, ou bien hétérogénéité des données, le cas où les données ne sont pas IID entre les clients. L'hétérogénéité statistique est un des problèmes et axes de recherche majeurs de l'apprentissage fédéré, car celle-ci peut causer la divergence des modèles locaux durant l'entraînement, et donc nuire aux performances du modèle global. Outre la possibilité de violation de l'hypothèse d'indépendance, Kairouz et al. [48] dresse la liste des cas où les données ne sont pas identiquement distribuées entre deux clients distincts $\mathcal{C}_i$ et $\mathcal{C}_j$. C'est-à-dire que $P_i \neq P_j$, sachant que l'on note $(x, y) \sim P_i(x, y)$ un exemple tiré de la distribution locale de $\mathcal{C}_i$. La liste est la suivante

— **Décalage des covariables** *(covariate shift)* : les distributions marginales $P_i(x)$ varient selon les clients. Par exemple, le même objet peut apparaître différemment selon l'étalonnage des caméras ayant capturé les images de ce dernier.
— **Dérive conceptuelle** *(concept drift)* : le même label peut être décrit par différentes caractéristiques pour différents clients, par exemple, en raison de variations météorologiques ou géographiques. Dans ce cas, ce sont les distributions conditionnelles $P_i(x|y)$ qui varient entre les clients.
— **Asymétrie dans la distribution des labels** *(label distribution skew)* : la distribution des labels peut varier entre les clients, par exemple, parce que la probabilité de rencontrer certains labels est liée à des conditions environnementales particulières. Ici, ce sont les distributions marginales $P_i(y)$ qui changent entre les clients.
— **Décalage conceptuel** *(concept shif)* : cas où le même vecteur de caractéristiques correspond à des labels différents selon les clients, par exemple, en raison d'écarts culturels ou régionaux. On observe que les distributions conditionnelles $P_i(y|x)$ changent avec les clients.
— **Déséquilibre** *(Unbalancedness)* : le nombre d'échantillons stockés peut considérablement varier d'un client à l'autre.

Pour chercher à résoudre le problème des données non-IID, il faut d'abord pouvoir le simuler. En classification d'images, soit la tâche la plus largement explorée en apprentissage fédéré, la distribution de Dirichlet est communément utilisée pour partitionner artificiellement un jeu de données en plusieurs groupes hétérogènes [49]. En revanche, pour les tâches plus riches, comme la détection d'objets, on cherche davantage à séparer naturellement les données en exploitant des métadonnées pertinentes, telles que des critères météorologiques ou spatio-temporelles pour le cas des systèmes de transport. De nombreuses techniques ont

été introduites pour améliorer la qualité de l'entraînement dans le cas non-IID, nous en présentons quelques-unes ci-après.

Corrections locales

Une première gamme de méthodes cherche à stabiliser l'optimisation du modèle global en introduisant des correctifs durant les mises à jour des modèles locaux, afin de prévenir la divergence de ces derniers. Ces techniques incluent notablement FedProx [50], qui introduit un terme de proximité dans chaque fonction objective locale et améliore la robustesse du processus d'optimisation face à des mises à jour locales variables, SCAFFOLD [51], qui utilise des variables de contrôle pour effectuer une réduction de variance et corriger les biais de dérive locale, et MOON [52], qui utilise de l'apprentissage contrastif au niveau des modèles et se base sur les similarités entre leurs représentations pour corriger l'optimisation locale. Cependant, puisque ces méthodes ont d'abord été conçues dans le contexte de la classification d'images, et exploitent des informations locales propres à la tâche considérée, on peut s'attendre à ce qu'elles soient moins efficaces lorsque appliquées à des tâches de vision par ordinateur plus avancées, comme cela a été montré pour la segmentation sémantique [53].

Optimisation côté serveur

Ayant reformulé l'agrégation effectuée par le serveur en un problème d'optimisation, il a été montré empiriquement que plusieurs optimiseurs classiques de l'apprentissage automatique peuvent améliorer la convergence du modèle global lorsque les données sont non-IID. Ces méthodes, puisqu'elles sont cantonnées côté serveur, ont pour avantage d'être orthogonales à celles décrites ci-dessus. On citera en particulier FedAvgM [49] qui intègre un terme de vélocité v à la mise à jour globale, accumulant les pseudo-gradients passés qui déclinent à un rythme exponentiel contrôlé par un facteur constant $\beta \in [0, 1)$. On pourra également enrichir cette technique d'un taux d'apprentissage constant côté serveur η, de telle manière que l'on retrouve FedAvg en prenant $\beta = 0$ et $\eta = 1$. L'actualisation du modèle global s'exprime alors

$$v^{t+1} = \beta v^t + \Delta^t \tag{2.7a}$$

$$w^{t+1} = w^t - \eta v^{t+1}. \tag{2.7b}$$

De la même manière, on peut introduire les méthodes fédérées FedAdagrad, FedAdam et FedYogi [45] simplement en employant côté serveur les optimiseurs adaptatifs bien connus AdaGrad [54], Adam [55] et Yogi [56]. Posons $\delta^t = (\Delta^t)^2$, on exprime alors la mise à jour du

modèle global de la façon suivante :

$$m^t = \beta_1 m^{t-1} + (1-\beta_1)\Delta^t \tag{2.8a}$$

$$v^t = \begin{cases} v^{t-1} + \delta^t & \text{pour FedAdagrad} \\ \beta_2 v^{t-1} + (1-\beta_2)\delta^t & \text{pour FedAdam} \\ v^{t-1} - (1-\beta_2)\delta^t \texttt{signe}(v^{t-1} - \delta^t) & \text{pour FedYogi} \end{cases} \tag{2.8b}$$

$$w^{t+1} = w^t - \eta \frac{m^t}{\sqrt{v^t} + \tau}. \tag{2.8c}$$

Personnalisation

L'apprentissage fédéré personnalisé, que nous avons introduit à la sous-section précédente, prend tout son sens dans le contexte de l'hétérogénéité statistique. Là où les méthodes de corrections locales cherchent à extraire des données hétérogènes les invariants permettant d'entraîner un unique modèle global satisfaisant au mieux tous les clients simultanément, la personnalisation de modèles permet de confronter le cas non-IID en adaptant le modèle global aux distributions de données observées par chaque client. Parmi les méthodes les plus populaires de personnalisation, on pourra citer FedBN [57], qui consiste et ne jamais agréger les paramètres associés à la normalisation par lots des modèles locaux.

Autres considérations

Appliquer un moment dans l'optimisation côté client permet d'améliorer la stabilité des mises à jour locales et accélérer la convergence de l'optimisation fédérée, y compris lorsque les données d'entraînement sont non-IID [58, 59]. Il a également été montré que démarrer l'apprentissage fédéré en initialisant le modèle global à partir de poids pré-entraînés, par exemple, à la suite d'une phase d'entraînement sur des données maintenues par le serveur, stabilise l'agrégation globale et réduit l'écart de performance observé avec l'apprentissage centralisé, en particulier lorsque les données sont hétérogènes [60, 61].

2.1.4 Confidentialité et sécurité

Un autre axe majeur de recherche en apprentissage fédéré consiste à construire des méthodes permettant de se prémunir contre des adversaires attaquant un système fédéré. Ceci est d'autant plus crucial que parmi les champs d'applications les plus populaires de l'apprentissage fédéré figurent les domaines de la santé et des transports, qui sont des domaines critiques. On distingue deux types de menaces : celles cherchant à compromettre la confidentialité des données et celles cherchant à compromettre la sécurité du système en détériorant ses

performances. Nous listons ci-dessous quelques exemples de vecteurs d'attaque auxquels un adversaire est susceptible de recourir :

- **Attaque par reconstruction de données :** un adversaire ayant obtenu des gradients envoyés par des clients au serveur, par exemple, en les interceptant, peut les analyser et déduire des informations sur les données d'origine.
- **Attaque par empoisonnement de modèles :** un adversaire contrôlant un ou plusieurs clients du système peut volontairement altérer les gradients transmis au serveur pour corrompre le modèle global.
- **Attaque par inférence d'appartenance :** un adversaire peut chercher à déterminer si une entrée spécifique faisait partie du jeu d'entraînement d'un modèle, si ce dernier réagit différemment lorsque évaluer sur ses données d'entraînement.
- **Serveur honnête mais curieux :** cas où le serveur central rempli certes correctement son rôle d'orchestrateur, mais cherche à extraire des informations sur les données des utilisateurs à partir des vecteurs de mise à jour qu'il collecte.
- **Serveur compromis :** un adversaire ayant compromis l'agent central est susceptible de causer des dommages importants, il pourrait, par exemple, insérer du code malveillant dans le modèle global et forcer sa dissémination à travers les clients.

Trois méthodes sont généralement considérées pour préserver la confidentialité des données ou la sécurité du système : le calcul multipartite sécurisé [62], un protocole cryptographique permettant à un ensemble de partis de calculer conjointement une fonction sur leurs entrées sans se les révéler mutuellement, le chiffrement homomorphe [63], qui permet d'effectuer des opérations sur des données chiffrées, et la confidentialité différentielle [64], une technique ajoutant du bruit aux calculs statistiques compliquant la déduction des données d'un client spécifique. Ces techniques, bien que populaires, ne sont pas exemptes de défauts. Ainsi, elles peuvent négativement affecter la charge calculatoire, alourdir les communications, ou encore dégrader les performances du modèle global.

2.1.5 Cadriciels

De nombreux cadriciels ont été conçus afin de faciliter l'exploration des multiples facettes de l'apprentissage fédéré. Parmi les plus populaires, ainsi déterminés par le nombre d'étoiles sur leur dépôt GitHub respectif, on pourra citer TensorFlow Federated, FedML, PySyft, Flower, ou encore Fate. Nous les avions précédemment évoqués en introduction, et ils sont tous toujours activement maintenus. Les cadriciels d'apprentissage fédéré sont susceptibles de se distinguer les uns des autres par le support offert à différents scénarios fédérés, distributions de données, bibliothèques d'apprentissage machine, méthodes de protection et de chiffrement,

protocoles de communication, ou encore leur capacité à supporter des simulations à grande échelle ou un déploiement en situation pratique sur des appareils périphériques. La principale limitation de ces cadriciels est dans le peu de tâches avancées qu'ils supportent, notamment en vision par ordinateur, où l'entraînement de modèles de détection d'objets, de segmentation, de détection de poses, etc., n'est généralement pas supporté ou bien difficile et peu documenté. Puisque nous n'avons finalement utilisé aucun de ses cadriciels dans nos expériences en raison de cette limitation, une récapitulation exhaustive de leurs fonctionnalités est hors du champ couvert par ce mémoire. En conséquence, on renverra à la très détaillée revue de littérature consacrée à ce sujet par Riedel et al. [65], pour de plus amples détails.

2.2 Détection d'objets

2.2.1 Généralités

Définitions

La détection d'objets est une tâche de vision par ordinateur qui consiste à identifier, sur une image ou une vidéo, l'ensemble des instances appartenant à des classes prédéfinies, ainsi qu'à les localiser en délimitant leurs contours par des boites englobantes, généralement rectangulaires [66]. En cela, la détection d'objets étend la classification d'images, qui ne fait correspondre à une image donnée qu'un seul label, et ajoute une étape de régression au problème. On donne un exemple de détection d'objets à trois classes à la Figure 2.1. En outre, les données vidéo peuvent être enrichies, par exemple, en recourant à des capteurs LiDAR, pour obtenir des informations sur la profondeur des objets et les localiser en trois dimensions, on parle alors de détection d'objets 3D.

L'apprentissage profond a eu un impact majeur sur la discipline, où les réseaux de neurones convolutifs sont devenus la base des détecteurs d'objets modernes, bien que des architectures fondées sur des transformateurs aient récemment opéré une percée [67]. Les réseaux de neurones convolutifs sont un cas particulier des réseaux de neurones à propagation avant, très utilisés en vision par ordinateur. Ils intègrent des opérations de convolution en raison de ses propriétés particulièrement pertinentes en traitement d'images, à savoir : connectivité éparse, partage de paramètres et équivariance à la translation [68]. Quant aux transformateurs, ils sont construits sur le mécanisme d'auto-attention et ont initialement été conçus pour la traduction automatique [69]. Ils ont été adaptés avec succès au domaine de la vision [70], où ils n'ont toutefois pas atteint la même dominance qu'en traitement de la langue naturelle.

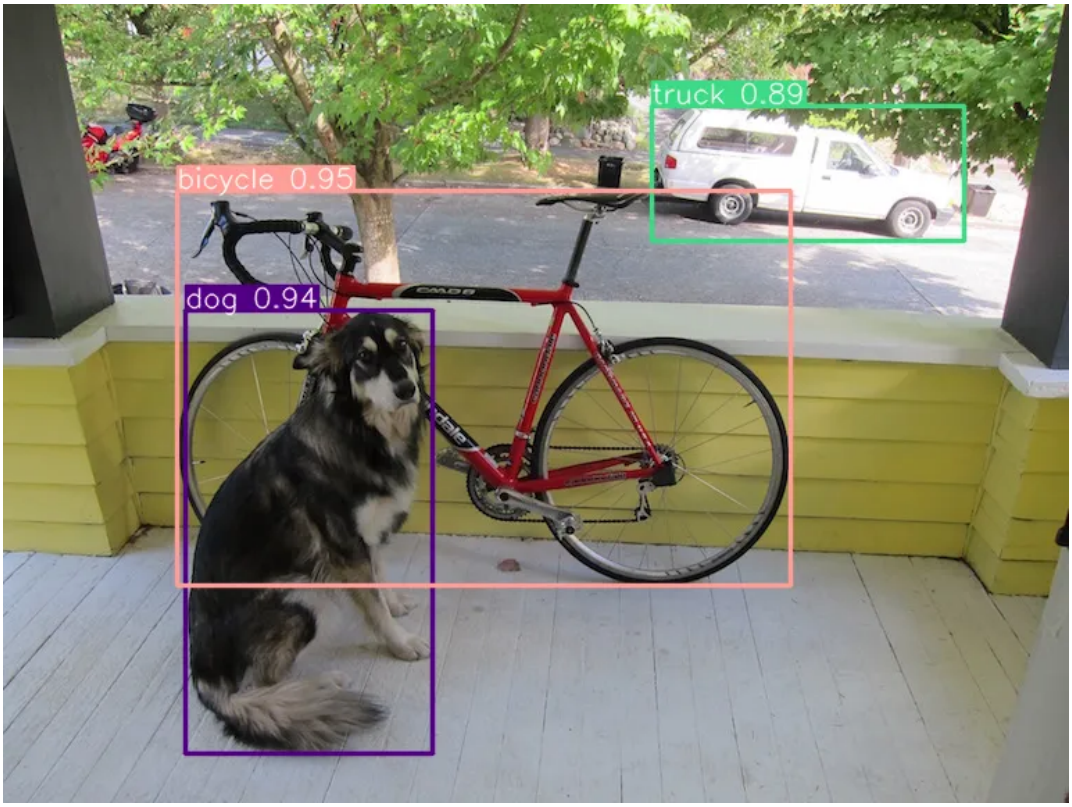

FIGURE 2.1 **Exemple de détection d'objets**. Le modèle, ici YOLOv10, délimite les objets par une boite englobante et les classifie. Un score de confiance est associé à chaque prédiction.

Métriques d'évaluation

L'évaluation des performances des modèles de détection d'objets est plus complexe qu'en classification, et fait intervenir plusieurs métriques qui sont listées par Padilla et al. [71], et dont nous proposons un récapitulatif. En premier lieu, on détermine si une prédiction est suffisamment proche spatialement de la boite englobante théorique correspondante, pour cela, on a recours à l'IoU qui est défini comme

$$\text{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{2.9}$$

où B_p est la boite prédite, B_{gt} la boite théorique, et t un seuil de confiance à partir duquel on considère la prédiction comme bien localisée. Maintenant en mesure de déterminer si une prédiction est correcte ou non, on peut introduire la précision P et le rappel R

$$\text{P} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Positifs}} \quad \text{et} \quad \text{R} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Négatifs}}. \tag{2.10}$$

La précision AP_t^k du modèle sur la classe k pour le seuil t est déduite par interpolation de l'aire sous la courbe précision $\times$ rappel. La précision moyenne du modèle sur tout un jeu de données de N classes est ensuite dérivée par simple moyennage

$$\mathrm{mAP}_t = \frac{1}{N} \sum_{k=1}^{N} \mathrm{AP}_t^k. \tag{2.11}$$

Enfin, pour améliorer la robustesse des résultats, on introduit la mAP, qui est obtenue en moyennant les mAP_t sur 10 seuils de confiance t allant de 50% à 95%, avec un pas de 5%. La mAP est la métrique privilégiée pour évaluer les performances des détecteurs d'objets. En pratique, son implémentation en interne peut subtilement varier d'un projet à l'autre, aussi, pour comparer des détecteurs d'objets n'étant pas inclus au sein d'un même cadriciel, on prendra soin de se référer à un programme tiers pour évaluer lesdits modèles. Par exemple, le très populaire jeu de données de détection d'objets généraliste MS COCO [72] est accompagné du progiciel *pycocotools*, qui permet de s'assurer que les modèles testés soient toujours évalués de manière consistante.

2.2.2 Détection d'objets en temps réel

Contexte

Dans une situation de temps réel, on cherche à détecter des objets sur un flot continu. Le modèle effectue donc des prédictions sur des images une à une, et sa vitesse, généralement mesurée en images par seconde, joue un rôle crucial. Au sein des architectures utilisées en détection d'objets, on distingue celles dont l'inférence consiste de deux étapes, comme R-CNN [73] et SPP-Net [74], qui génèrent d'abord des propositions de régions avant de les envoyer à un modèle de classification, de celles qui localisent et classent les objets en un seul passage, comme YOLO [75], SSD [76] et RetinaNet [77]. Si des progrès ont été accomplis pour accélérer la vélocité des modèles 2-étapes, comme avec Faster-R-CNN [78], ces secondes architectures sont généralement préférées dans le contexte de la détection en temps réel, car plus rapides. La série de modèles YOLO est particulièrement populaire, elle a été introduite par Joseph Redmon, qui est largement responsable de la création de l'architecture originale et de ses deux suites YOLOv2 [79] et YOLOv3 [80]. En effet, grâce à de nombreuses améliorations successives, les modèles YOLO ont acquis une reconnaissance significative dans le domaine de la détection d'objets et ont été amplement utilisés dans diverses applications, y compris les véhicules autonomes, les systèmes de surveillance et la robotique [81]. YOLO a également été utilisé dans le contexte de la détection d'objets 3D, avec le modèle Complex-YOLOv4 [82].

Présentation de YOLOv7

Puisque YOLOv7 est le modèle que nous avons utilisé dans nos travaux, nous lui accordons une place particulière dans cette revue de littérature. De nombreux éléments que nous rapportons sont issus de l'examen du code source de YOLOv7 et ne sont pas détaillés directement dans l'article ayant introduit ce modèle.

YOLOv7 est l'une des dernières itérations de la série de modèles YOLO, et était le plus performant des modèles de la plage de 5–160 FPS au moment de sa publication, en juillet 2022. YOLOv7 a été entraîné à partir de zéro sur le jeu de données MS COCO et présente de nouvelles structures optimisées et méthodes d'optimisation nommées *trainable bag-of-freebies*. Ces derniers augmentent le coût d'apprentissage et les performances prédictives d'un modèle sans pour autant ralentir sa vitesse d'inférence, un concept déjà abordé dans YOLOv4 [83]. Parmi ceux-ci, on retrouve la re-paramétrisation planifiée des couches de convolution, basée sur l'analyse du parcours du gradient, et une nouvelle stratégie pour l'attribution des labels. On retrouve également des concepts préexistants comme l'intégration des statistiques des couches de normalisation des lots directement dans les couches convolutives à l'inférence, la connaissance implicite de YOLOR [84] fusionnée aux couches convolutives pour les additions et les multiplications, et l'utilisation d'une moyenne mobile exponentielle des modèles entraînés pour construire le modèle d'inférence final. YOLOv7 introduit également des innovations architecturales, comme Extended-ELAN, une variante du réseau ELAN [85] permettant d'empiler indéfiniment des blocs de calcul sans perte des capacités d'apprentissage, et une nouvelle méthode pour dériver des variantes adaptées à différentes échelles à partir de l'architecture première, et ce tout en préservant ses propriétés et sa structure optimale.

Plus généralement, YOLOv7 est un modèle dit avec ancre, ou *anchor-based*, basé sur une FPN [86]. Les auteurs introduisent d'abord YOLOv7-tiny et YOLOv7, des modèles P5 respectivement conçus pour l'évaluation sur des GPUs de puissance faible et moyenne, et YOLOv7-W6, un modèle P6 conçu pour les GPUs de centre de données. La technique de mise à l'échelle est ensuite appliquée à YOLOv7 pour dériver YOLOv7-X, et à YOLOv7-W6 pour obtenir YOLOv7-E6, YOLOv7-D6 et YOLOv7-E6E, ce dernier remplaçant ELAN par Extended-ELAN en tant qu'unité de calcul principale. La résolution recommandée pour les données d'entraînement est de 640×640 pour les modèles P5 et de 1280×1280 pour les modèles P6, les entrées passées à YOLOv7 étant redimensionnées sans déformation, en appliquant un remplissage si nécessaire. Pendant l'apprentissage, la fonction de perte est décomposée en trois sous-fonctions équilibrées par un triplet d'hyperparamètres (α, β, γ). Ces sous-fonctions mesurent les performances du modèle selon trois différentes modalités. En particulier, l'entropie croisée binaire est utilisée par la perte de classification et la perte

d'*objectness* (ce dernier concept renvoyant la confiance qu'a le modèle qu'une région donnée contient un objet). La perte de régression, quant à elle, évalue la qualité de la localisation des boites englobantes et utilise la CIoU [87]. La perte totale s'écrit donc

$$\mathcal{L}_{totale} = \alpha \mathcal{L}^{\text{BCE}}_{\text{classification}} + \beta \mathcal{L}^{\text{BCE}}_{\text{objectness}} + \gamma \mathcal{L}^{\text{CIoU}}_{\text{régression}}. \tag{2.12}$$

Toutes les variantes de YOLOv7 utilisent la fonction d'activation SiLU (à l'exception de YOLOv7-tiny où LeakyReLU est utilisée à la place), l'accumulation du gradient, la stratégie SimOTA introduite par YOLOX pour l'attribution des labels [88], une précision mixte durant l'entraînement, une demi-précision pour l'inférence et plusieurs techniques d'augmentation de données. L'architecture complète de YOLOv7 est illustrée à la Figure 2.2.

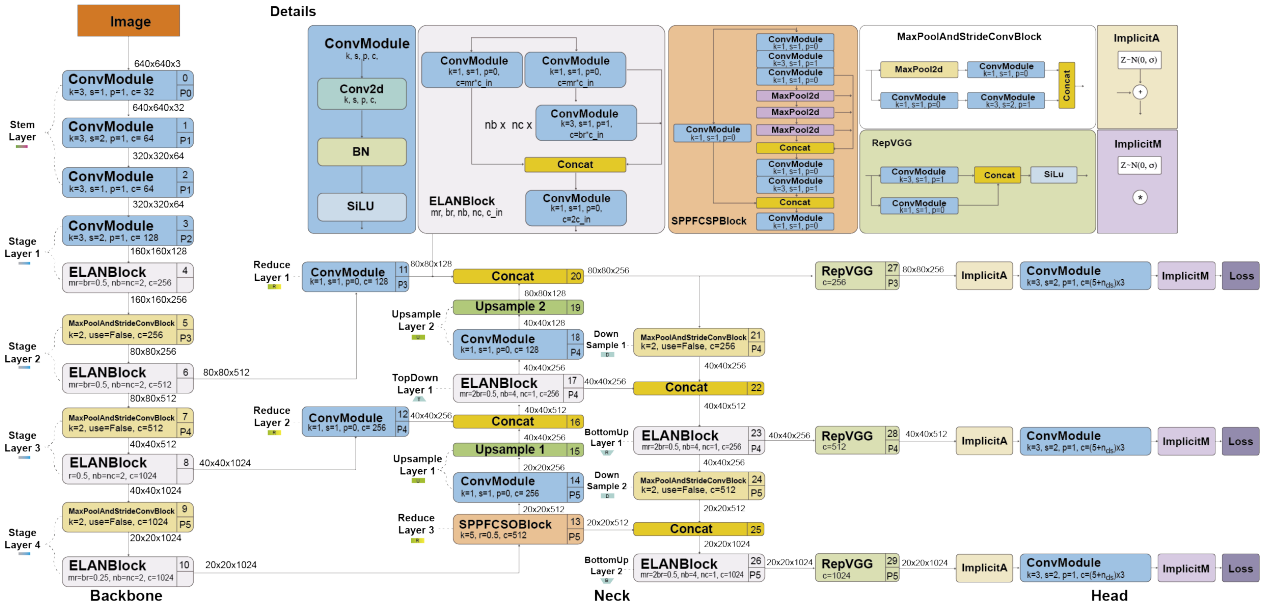

FIGURE 2.2 **Architecture de YOLOv7.** Le modèle introduit des innovations architecturales, ainsi que des structures optimisées et des méthodes d'optimisation appelées *trainable bag-of-freebies*. Ces dernières améliorent les prédictions de YOLOv7 au prix d'un entraînement plus long, mais sont sans impact sur la vitesse d'inférence du modèle. Image reproduite sans modification [81] (Licence : CC BY 4.0).

L'algorithme d'optimisation est la descente stochastique du gradient avec moment de Nesterov, mais avec la subtilité que l'on distingue trois groupes de paramètres au sein de YOLOv7, à savoir les biais, les paramètres de normalisation des lots et enfin le reste des paramètres. Une régularisation L2 est appliquée uniquement à ces derniers, et à chaque groupe, on associe un taux d'apprentissage suivant une phase d'échauffement linéaire, suivie d'une phase de recuit cosinus. Comme pour la plupart des détecteurs d'objets, l'algorithme NMS est inclus en

post-traitement pour filtrer les boites englobantes redondantes et conserver seulement celles dans lesquelles les niveaux de confiance du modèle sont les plus élevés. En sus de la détection d'objets en 2D, YOLOv7 a été étendu à l'estimation de poses et à la segmentation d'instances, et une variante dite sans ancre, ou *anchor-free*, nommée YOLOv7-AF, a été introduite.

Modèles publiés après YOLOv7

Plusieurs modèles de détection d'objets en temps réel ont été rendus publics à la suite de YOLOv7, nous présentons brièvement les plus notables d'entre eux ci-après, par ordre chronologique de publication.

— **RTMDet** [89] (décembre 2022) : est une nouvelle architecture de modèle convolutif 1-étape, présenté par les créateurs de la plateforme MMDetection, qui propose de nombreuses implémentations de modèles de l'état de l'art sous license Apache-2.0.

— **YOLOv6 v3.0** [90] (janvier 2023) : est un modèle proposé par la compagnie chinoise Meituan. Cette réédition de YOLOv6, qui intègre des techniques récentes, améliore significativement les performances du modèle, qui demeurent toutefois inférieures à celles de YOLOv7-AF.

— **YOLOv8** [91] (janvier 2023) : est le dernier modèle en date de la compagnie Ultralytics, créatrice du très populaire modèle YOLOv5 [92]. Les performances de base de YOLOv8 sont semblables à celle de YOLOv7, mais Ultralytics propose un support à long terme du modèle ainsi qu'une meilleure intégration des outils conçus pour un déploiement en situations réelles. En revanche, aucun article de recherche n'est associé à ce modèle et son implémentation officielle est placée sous une licence plus restrictive que YOLOv7 (AGPL-3.0), avec cependant une licence commerciale disponible.

— **RT-DETR** [93] (avril 2023) : a ouvert de nouvelles possibilités de recherche en montrant que des modèles de type transformateur, communément trop lents, peuvent être modifiés pour atteindre des performances de l'état de l'art en temps réel. Les modèles de la lignée DETR éliminent le besoin de recourir à de nombreux composants conçus à la main, tel que le NMS, pour effectuer une détection de bout en bout.

— **YOLOv9** [94] (février 2024) : est une amélioration directe de YOLOv7-AF proposée en partie par les mêmes auteurs. YOLOv9 introduit le mécanisme PGI pour prévenir la perte d'information dans les couches profondes du réseau de neurones et le réseau GELAN, qui généralise l'ancien bloc ELAN. En outre, YOLOv9 supporte l'apprentissage multitâche avec la segmentation.

— **YOLOv10** [95] (mai 2024) : se place dans la continuité de YOLOv8 (et hérite donc de la licence AGPL). Il s'agit du premier modèle de la série YOLO à se concentrer sur

le problème de la détection de bout-en-bout en éliminant NMS du post-traitement. Lorsque le post-traitement est pris en compte dans l'évaluation de la vitesse des modèles, YOLOv10 atteint les meilleures performances.

2.2.3 Jeux de données pour la conduite autonome

La détection d'objets étant l'une des tâches les plus fondamentales de vision par ordinateur, et d'une importance critique pour les véhicules autonomes, de nombreux jeux de données de conduite ont été assemblés pour capturer toujours plus de scénarios et faire avancer l'état de l'art. Un bon jeu de données cherchera à se démarquer des propositions préexistantes en cumulant autant que possible de critères désirables tels que : multimodalité en incluant différents types de capteurs, grande quantité de données labellisées et non labellisées dont images acquises consécutivement pour capturer les dépendances temporelles, diversité géographique et d'environnements, diversité météorologique (dont neige et brouillard), diversité d'angles d'acquisition, conduite de nuit, inclusion de classes rares, richesse des métadonnées dont contexte dynamique (comme la direction et la vitesse des véhicules), réduction des biais dans la collecte de données, ou encore annotations multitâches pour associer la détection d'objets avec la segmentation ou l'estimation de pose.

Nous proposons ici une brève description des jeux de données ayant été utilisés dans le cadre des travaux que nous présentons à la sous-section suivante.

— **KITTI** [96] : comprend un ensemble de jeux de données de vision par ordinateur centrés sur la conduite autonome. KITTI est un pionnier en raison de son approche multimodale intégrant caméras, capteurs LiDAR, et localisation GPS, et supporte la détection d'objets 2D et 3D, le suivi d'objets, ou encore la segmentation. Jouissant toujours d'une grande popularité en raison de son ancienneté, KITTI souffre pourtant d'une faible diversité géographique, météorologique, ainsi que dans l'orientation et l'illumination des objets. Le jeu de données consacré à la détection d'objets 2D compte huit classes et est de surcroît de petite taille.

— **Cityscapes** [97] : est un jeu de données de référence pour la segmentation en milieu urbain. Il comprend 5000 images avec des annotations pixel par pixel détaillées ainsi que 20000 images supplémentaires avec des annotations plus grossières, capturées dans 50 villes, majoritairement allemandes, sur plusieurs mois. Le tout couvre 30 classes d'objets couramment rencontrées dans des scènes complexes et diversifiées. Cette offre est complétée par Cityscapes 3D, une extension contenant des annotations de boites englobantes pour les véhicules.

— **BDD100K** [98] : est un jeu de données notablement vaste, puisqu'il comporte 100 000

vidéos de 40 secondes filmées en haute définition sur plus de 1000 heures de conduite à travers différentes régions des États-Unis. Il supporte une multitude de tâches de vision par ordinateur, dont la détection d'objets, le suivi, la segmentation et la détection de voie, et facilite l'apprentissage multitâche sur des données hétérogènes, mais ne contient en revanche pas d'autres modalités que la vidéo. L'ensemble de données dédié à la détection d'objets contient 100 000 images annotées (une par vidéo) et dix classes.

— **nuScenes** [99] : est un large jeu de données multimodales de conduites réalisées à Boston et Singapour. Ainsi, sont inclues données vidéo, LiDAR et radar, acquises à 360° et réparties dans mille différentes scènes, dont de nuit, et par temps de pluie et de neige, sur une échelle de temps de plusieurs mois. Avec 23 classes, il s'agit également de l'un des quelques jeux de données de détection d'objets à inclure des classes rares. En complément de nuScenes, le jeu de données nuImages suit les mêmes spécifications en offrant toutefois une quantité beaucoup plus importante d'images, avec des annotations 2D uniquement.

— **Oxford Radar RobotCar** [100] : met l'accent, comme son nom l'indique, sur l'importance des données radar pour améliorer les capacités de perception des véhicules autonomes. Les concepteurs argumentent que cette modalité est sous-explorée alors qu'elle manifeste une grande robustesse face aux conditions environnementales adverses comme la pluie, la neige ou le brouillard. Les données collectées grâce à des radars à ondes millimétriques sont introduites en complément du jeu de données préexistant Oxford RobotCar [101], qui contient des données vidéo, LiDAR et GPS.

— **CADC** [102] : pour *Canadian Adverse Driving Conditions* est un jeu de données multimodal, constitué d'images et d'annotations LiDAR intégralement collectées durant l'hiver dans la région de Waterloo au Canada. Il s'agit du premier jeu de données centré spécifiquement sur la conduite dans des conditions météorologiques difficiles.

— **SODA10M** [103] : se distingue des propositions précédentes en soulignant l'importance de l'apprentissage semi-supervisé dans les approches modernes de vision. Ainsi, SODA10M affiche un nombre impressionnant de 10 millions d'images capturées dans des environnements diversifiées, tirés de 32 différentes villes, mais seulement 20 000 d'entre elles ont été labellisées, avec des annotations 2D pour six classes d'objets.

— **V2XSet** [104] : a été publié conjointement avec le modèle de vision V2X-ViT, afin de lui servir de support. V2XSet se focalise sur le problème de la perception coopérative en faisant intervenir autant véhicules autonomes qu'infrastructures environnantes. Le jeu de données comporte 11447 entrées uniques tirées de 55 scènes d'au plus 25 secondes, faisant intervenir entre deux et sept agents intelligents. Ces scènes ont été simulées dans CARLA [105], que nous introduisons ci-dessous.

En sus des jeux de données statiques, plusieurs travaux explorent l'utilisation de simulateurs véhiculaires pour assister le développement, l'entraînement et la validation des systèmes de conduite autonome. CARLA est un simulateur open source populaire construit sur l'Unreal Engine, qui permet de spécifier les conditions environnementales à observer durant la simulation et configurer différents types de capteurs. L'environnement virtuel de CARLA a, en outre, pour avantage de pouvoir être utilisé pour faciliter l'implémentation de méthodes d'apprentissage par renforcement.

2.3 Application aux véhicules autonomes

Un véhicule autonome est un système auto-opérant intégrant des capteurs, des actionneurs et des algorithmes avancés d'intelligence artificielle, le rendant à même de percevoir et de naviguer dans divers environnements, en limitant ou supprimant l'intervention humaine. Pour cela, il doit atteindre un haut degré de conscience situationnelle, s'adapter à des conditions dynamiques et garantir une conduite sûre en temps réel. SAE International catalogue six niveaux d'automatisation faisant autorité dans la communauté des ingénieurs [106], résumés dans la Figure 2.3. En 2024, les véhicules autonomes les plus avancés peuvent opérer sans intervention humaine, mais avec des restrictions importantes, on parle alors de niveau quatre d'autonomie. Ces restrictions incluent des contraintes de conditions météorologiques, mais surtout le géorepérage qui limite ces véhicules à naviguer dans des aires géographiques précises. En raison de ces restrictions géographiques, les véhicules autonomes de niveau quatre ne s'inscrivent donc pas dans un usage personnel et prennent d'autres formes, tels que les robotaxis de la compagnie Waymo qui participent à la mobilité comme service.

Dans ce qui suit, nous passons en revue les travaux antérieurs qui ont appliqué l'apprentissage fédéré à la détection d'objets en 2D dans les véhicules autonomes. Rjoub et al. [19] ont étudié la détection d'objets fédérée en temps réel dans des scénarios de conduite marqués par des conditions météorologiques difficiles, en particulier en présence de fortes chutes de neige. Chen et al. [21] se sont intéressés aux contraintes de communication et de calcul auxquelles sous soumis les véhicules pour réaliser l'entraînement fédéré de détecteurs d'objets. Bommel [22], et plus tard Rjoub et al. [23], ont appliqué l'apprentissage actif pour confronter la problématique de l'absence d'étiquetage disponible pour les données collectées durant la conduite et permettre l'optimisation fédérée de modèles de détection d'objets en temps réel. Pour le premier article, le corps, ou *backbone* du modèle est gelé, et n'est donc pas échangé durant l'entraînement, réduisant subséquemment le coût des communications. Dai et al. [24] ont proposé FLAME, un cadriciel destiné à faciliter l'exploration de la détection d'objets fédérée sur des données acquises en continu par des véhicules autonomes. Wang et al. [25]

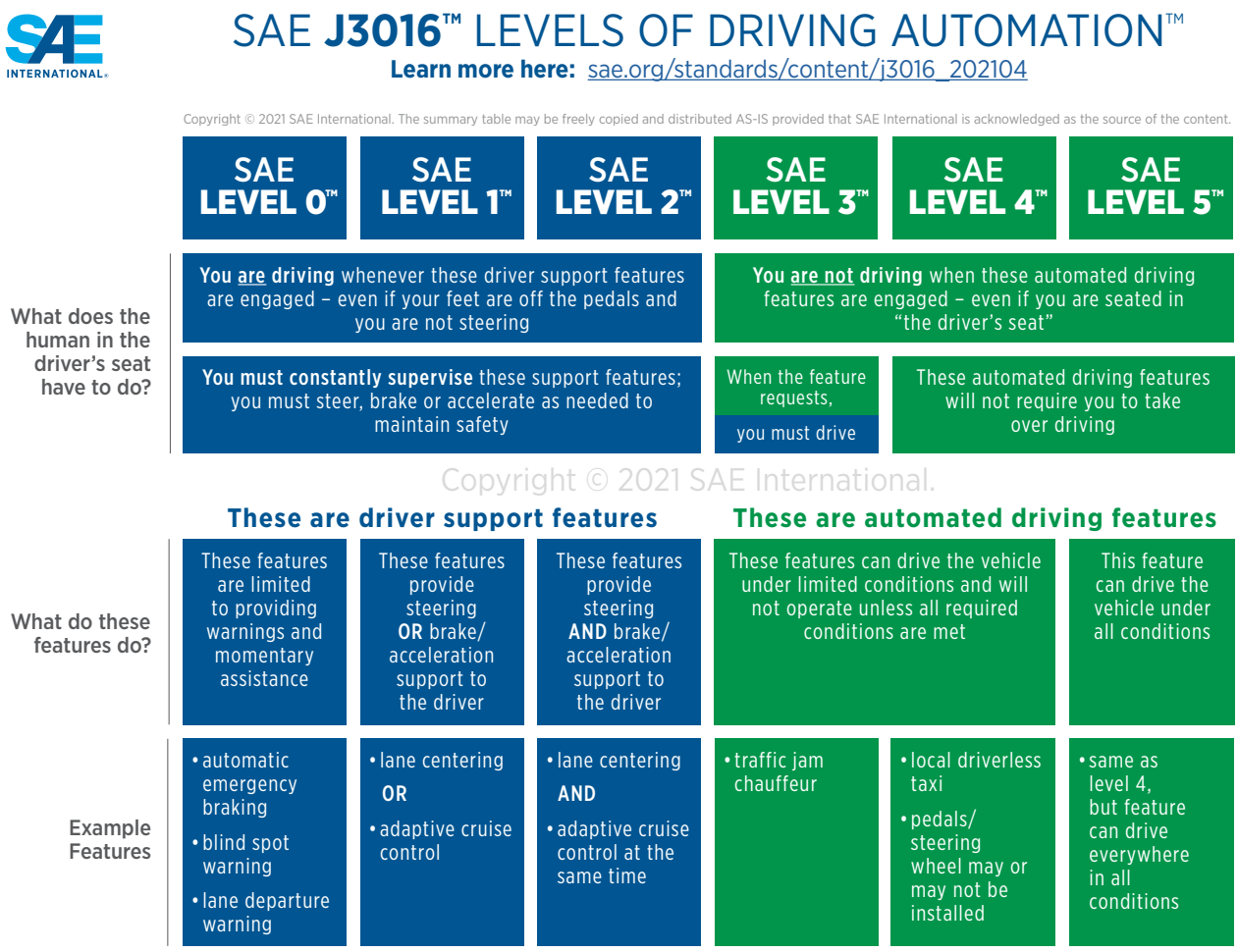

FIGURE 2.3 **Six niveaux d'automatisation de la conduite par SAE International.** Les véhicules autonomes de pointe parviennent au niveau quatre d'autonomie et peuvent donc se déplacer sans intervention humaine, avec toutefois des limitations importantes, en particulier le besoin de recourir au géorepérage. Aucune technologie à ce jour ne permet d'approcher le niveau cinq, où un véhicule se conduirait sans restrictions, y compris géographiques. Charte visuelle officielle associée à la dernière révision du standard SAE J3016 [106], reproduite sans modification (Copyright © 2021 SAE International).

ont conçu CarlaFLCAV, un simulateur ouvert d'apprentissage fédéré, prenant en charge un large éventail de tâches de perception automobile, y compris la détection d'objets, et se sont attaqués à l'optimisation des ressources du réseau et du placement des capteurs routiers. Chi et al. [26] ont conçu une méthode semi-supervisée permettant l'entraînement de modèles de détection d'objets par apprentissage fédéré sur des données non étiquetées nouvellement collectées pendant la conduite, étant donné une petite quantité de données préexistantes et bien annotées. Rao et al. [27] se sont intéressés aux contraintes de puissance de calcul et de charge sur le réseau de communication posées par l'utilisation de l'apprentissage fédéré et ont

proposé FedWeg, un processus d'entraînement de modèles intégrant de la *sparsification* et testé sur un système distribué comportant un serveur et plusieurs ordinateurs portables. Bien que contextualisé pour l'Internet des véhicules, on notera cependant que l'article utilise un jeu de données d'images de mains [107], et non un jeu de données de conduite. Su et al. [28] ont proposé FedOD, un cadriciel d'apprentissage fédéré personnalisé et inter-domaines pour la détection d'objets basé sur des méthodes de distillations, et ont validé leur proposition sur des jeux de données de conduite autonome. Enfin, Kim et al. [29] ont proposé une stratégie d'entraînement en deux étapes, appelée FedSTO, et ont abordé la détection d'objets fédérée semi-supervisée dans des situations dans lesquelles les ensembles de données locaux non-IID des clients véhiculaires sont intégralement non labellisés, des données étiquetées n'étant disponibles que pour le serveur central. Le coût des communications est également réduit par l'ablation du cou du modèle durant la première phase d'entraînement.

En revanche, d'autres chercheurs ont préféré porter leur attention sur les situations où les véhicules acquièrent des données multimodales grâce à leurs capteurs. En particulier, Zheng et al. [30] ont proposé AutoFed, un cadriciel fédéré dédié à la détection de véhicules en vue à vol d'oiseau, à partir de données LiDAR et radar non-IID. AutoFed a de surcroît été déployé sur un serveur accompagné de plusieurs appareils périphériques, spécifiquement des NVIDIA Jetson TX2. Mishra et al. [31] ont implémenté un système d'apprentissage fédéré à chaîne de blocs avec des contrats intelligents. Ainsi, les véhicules n'ont plus besoin de recourir à un serveur central pour entraîner conjointement un modèle de détection d'objets 3D. De plus, Chi et al. [32] ont montré que l'entraînement fédéré de détecteurs d'objets 3D peut être amélioré en incorporant au système les infrastructures environnantes du réseau véhiculaire plutôt que de se limiter aux seuls véhicules.

Notre proposition se distingue des travaux précédents en se concentrant sur l'optimisation fédérée d'un modèle récent, YOLOv7, et en conduisant une évaluation plus complète de ses performances. FedPylot permet en outre de réaliser des simulations à grande échelle sur des grappes de calcul. Notre stratégie de séparation des données, détaillée dans la section 4.6.3, présente également des particularités que l'on ne retrouve pas dans les articles ci-dessus. Nous proposons enfin l'un des quelques travaux à fournir une implémentation ouverte.

Notons néanmoins que deux articles publiés en 2024 ont expérimenté avec YOLOv8, un modèle aux performances comparables à celui que nous avons utilisé. Ces articles ont respectivement introduit FedProx+LA, une méthode pour traiter les biais de distribution des labels dans les réseaux véhiculaires, qui a montré une amélioration en termes de performances et de vitesse de convergence [33], et une technique hiérarchique adaptative pour traiter les limitations de stockage et de bande passante, validée pour détecter des voitures dans des conditions

météorologiques et d'éclairage variables [34]. Toutefois, ces travaux n'ont considéré que la variante nano, soit la plus petite, de YOLOv8, qui est conçue pour les appareils périphériques et dont les capacités d'apprentissage sont limitées. Cela souligne encore davantage la nécessité de fournir un support pour l'expérimentation fédérée à plus grande échelle en détection d'objets. Le prototype présenté en 2021 par Jallepalli et al. [20] est finalement le plus semblable à notre proposition. À notre connaissance, il s'agit de la seule contribution présentant l'optimisation fédérée d'un détecteur d'objets dans un environnement de calcul intensif à des fins de conduite autonome. Nous améliorons considérablement ce travail antérieur en remplaçant YOLOv3 par YOLOv7, en introduisant une plus grande diversité de données, plus de clients, des situations non-IID réalistes, plus de configurations pour l'optimisation locale et globale du modèle, des mesures des coûts de communication et de la vitesse d'inférence pour plusieurs variantes de YOLOv7, et en remplaçant les communications de bas niveau basées sur la programmation de socket par MPI, et le schéma de chiffrement symétrique Fernet par un schéma hybride plus sûr et tout aussi efficace.

On présente des informations complémentaires pour les articles présentés ci-dessus dans le Tableau 2.1, et que nous comparons à notre proposition. Spécifiquement, on récapitule pour chaque article : sa date de publication, la distribution des données utilisées et toutes autres informations pertinentes relatives à celles-ci, le détecteur utilisé, le jeu de données ou simulateur employé, s'il s'agissait de détection en deux ou trois dimensions, et si des méthodes de protection ou de compression ont été implémentées. La dernière colonne précise également si la simulation a été distribuée sur plusieurs machines, que ce soit dans le nuage ou sur des appareils périphériques, plutôt qu'un entraînement séquentiel exécuté sur un seul ordinateur.

Tableau 2.1 Articles portant sur la détection d'objets fédérée pour véhicules autonomes

Référence	Année	Données	Modèle	Jeu de données ou simulateur	Modalité	Protection	Compression	Déploiement distribuée
[19]	2021	IID	YOLO	CADC	2D	Non	Non	Non
[20]	2021	IID	YOLOv3	KITTI	2D	Chiffrement symétrique	Non	Grappe de calcul (Socket)
[21]	2021	IID	SSD	Non spécifié	2D	Non	Quantification Sparsification	Non
[22]	2021	IID Labélisation partielle	YOLOv5s	KITTI	2D	Non	Ablation	Non
[23]	2022	IID Labélisation partielle	YOLOR	SODA10M	2D	Non	Non	Non
[24]	2023	Non-IID En ligne	YOLOv2	CARLA	2D	Non	Non	Non
[25]	2023	IID Non-IID	YOLOv5	KITTI CARLA	2D 3D	Non	Non	Non
[26]	2023	IID Labélisation partielle	Faster-RCNN	nuImages	2D	Non	Non	Non
[27]	2023	IID	YOLOv3	Hand Dataset	2D	Non	Sparsification	Serveur et ordinateurs portables
[28]	2023	non-IID	RetinaNet	BDD100K SODA10M nuScenes	2D	Non	Non	Non
[29]	2023	IID Non-IID Labélisation partielle	YOLOv5	BDD100K SODA10M CityScapes	2D	Non	Ablation	Non
[30]	2023	IID Non-IID	Modèle 2-étapes personnalisé	Oxford Radar RobotCar nuScenes	LiDAR et radar seulement	Non	Non	Serveur et NVIDIA Jetson TX2
[31]	2023	IID	Complex-YOLOv4	KITTI	3D	Chaîne de blocs	Non	Non
[32]	2023	Non-IID Hiérarchique	V2X-ViT	V2XSet	3D	Non	Non	Non
[33]	2024	IID Non-IID	YOLOv8n	nuScenes	2D	Non	Non	Non
[34]	2024	Non-IID Hiérarchique	YOLOv8n	CARLA	2D	Non	Non	Non
FedPylot	**2024**	**IID Non-IID**	**YOLOv7-tiny YOLOv7 YOLOv7-X YOLOv7-W6**	**KITTI nuImages**	**2D**	**Chiffrement hybride**	**Demi-précision**	**Grappe de calcul (MPI)**

CHAPITRE 3 MÉTHODOLOGIE

Nous fournissons ici des détails en complément de l'article de recherche disponible au chapitre suivant. Ces éléments ne sont pas nécessaires à la compréhension de notre article, mais visent à éclairer les raisons ayant guidé certains de nos choix méthodologiques.

3.1 Choix du modèle et des jeux de données

Lorsque s'est initialement posé la question du choix du modèle de détection d'objets en temps réel à employer, YOLOv7 était la version la plus récente de la série de modèles YOLO et le plus performant des modèles de détection en temps réel. En outre, la série YOLO était un choix de prédilection en raison de la réputation de simplicité de ses modèles. De plus, YOLOv7 semblait un choix logique, car reprenant en partie l'implémentation du très populaire YOLOv5, auquel ont été ajoutées les innovations majeures introduites dans YOLOR et YOLOX, en sus de ses propres nouveautés, en rupture avec l'état de l'art. Par la suite, au cours du développement de FedPylot, de nouveaux modèles ont été rendus disponibles et notre décision de conserver YOLOv7 s'est faite pour essentiellement deux raisons. D'une part, les améliorations introduites par les modèles ultérieurs ont été plus incrémentales, d'autre part, les auteurs de YOLOv7 avaient communiqué sur leur volonté, malheureusement jamais concrétisée, d'étendre leur modèle à la détection 3D, ce qui aurait été un atout majeur dans le contexte de la conduite autonome.

Quant au choix des jeux de données à utiliser, celui-ci s'est fait en deux temps. KITTI s'est imposé très rapidement, en effet, puisque ce jeu de données est extrêmement populaire, il fournit un point de référence immédiat au lecteur. De plus, son homogénéité et sa simplicité le rendent adapté au cas IID, enfin sa petite taille (seulement 7481 images annotées, entraînement et validation confondus) en a fait un excellent candidat pour prototyper notre programme avant d'entamer des expériences plus lourdes calculatoirement. Il convenait ensuite de déterminer un second jeu de données, avec plus d'échantillons et la possibilité de les séparer sémantiquement pour étudier le cas non-IID. Parmi les options possibles, nous avons choisi nuImages, ce jeu de données contenant beaucoup d'images annotées, des échantillons issus de deux villes très différentes et éloignées (Boston et Singapour), des variations de luminosité, d'angles et de conditions météorologiques, des images de nuit, une plage d'échantillonnage importante (de janvier à septembre), et des identifiants regroupant les images collectées lors d'une même séquence de navigation.

Avant de débuter l'apprentissage fédéré, on se doit de prétraiter les jeux de données afin de les adapter au format attendu par YOLOv7, et, dans le cas de nuImages, extraire les métadonnées permettant de répartir les échantillons selon la méthode de séparation non-IID qui aura été choisie. Pour KITTI comme pour nuImages, les boites englobantes sont définies selon leurs coordonnées absolues avec le point en haut à gauche (x_g, y_h) et le point en bas à droite (x_d, y_b). Or, YOLOv7 s'attend à recevoir les coordonnées du centre de la boite (x, y), sa largeur l et sa hauteur h, le tout normalisé par largeur L et la hauteur H de l'image. Par suite, on doit effectuer la conversion suivante :

$$x = \frac{x_g + x_d}{2L}, \quad y = \frac{y_h + y_b}{2H}, \quad l = \frac{x_d - x_g}{L}, \quad h = \frac{y_b - y_h}{H}. \tag{3.1}$$

Pour KITTI, à chaque image correspond un fichier texte contenant la liste des coordonnées des boites et les classes qui y sont associées. Puisque cette structure convient à YOLOv7, seule la conversion des coordonnées est nécessaire. Pour nuImages, le problème est plus complexe, car les métadonnées sont placées dans des fichiers JSON et organisées selon une base de données relationnelle, ainsi que retranscrit à la Figure 3.1. Ainsi, en sus de leur conversion, il convient d'extraire les coordonnées (*bbox* de la table *object_ann*) ainsi que la classe correspondante (*name* de la table *category*) à partir des échantillons listés dans la table *sample*, et de sauvegarder le tout dans un fichier texte. Pour séparer les données, on récupère également les informations spatio-temporelles (*vehicle*, *date_captured*, *location*) de la table *log*.

3.2 Calcul haute performance

Toutes nos expériences, à l'exception des mesures de vitesse d'inférence réalisées sur Google Colab, ont été effectuées sur la grappe de calcul Cedar de l'Alliance de recherche numérique du Canada. L'Alliance, en partenariat avec les organisations régionales ACENET, Calcul Québec, Compute Ontario, BC DRI Group et Prairies DRI, propose cinq différentes grappes : Béluga, Cedar, Graham, Narval et Niagara. Le choix de Cedar a été motivé par le fait qu'accéder à Internet est possible sur ses nœuds de calcul, permettant donc de surveiller les expériences avec des outils de MLOps comme Weights & Biases, et qu'il s'agisse de la seule grappe à posséder des V100, soit le même GPU que celui utilisé dans l'article ayant introduit YOLOv7. En tout, 192 nœuds contenant chacun quatre V100-SXM2-32GB accompagnés de 32 CPUs sont disponibles, en sus de 146 autres nœuds proposant d'autres modèles de GPUs. Les grappes de l'Alliance utilisent InfiniBand [108] pour connecter les nœuds de calcul, sauf pour Cedar où Intel OmniPath [109] est utilisé à la place. L'ordonnancement du niveau de priorité des requêtes soumises par les utilisateurs d'une grappe est réalisé avec Slurm [110],

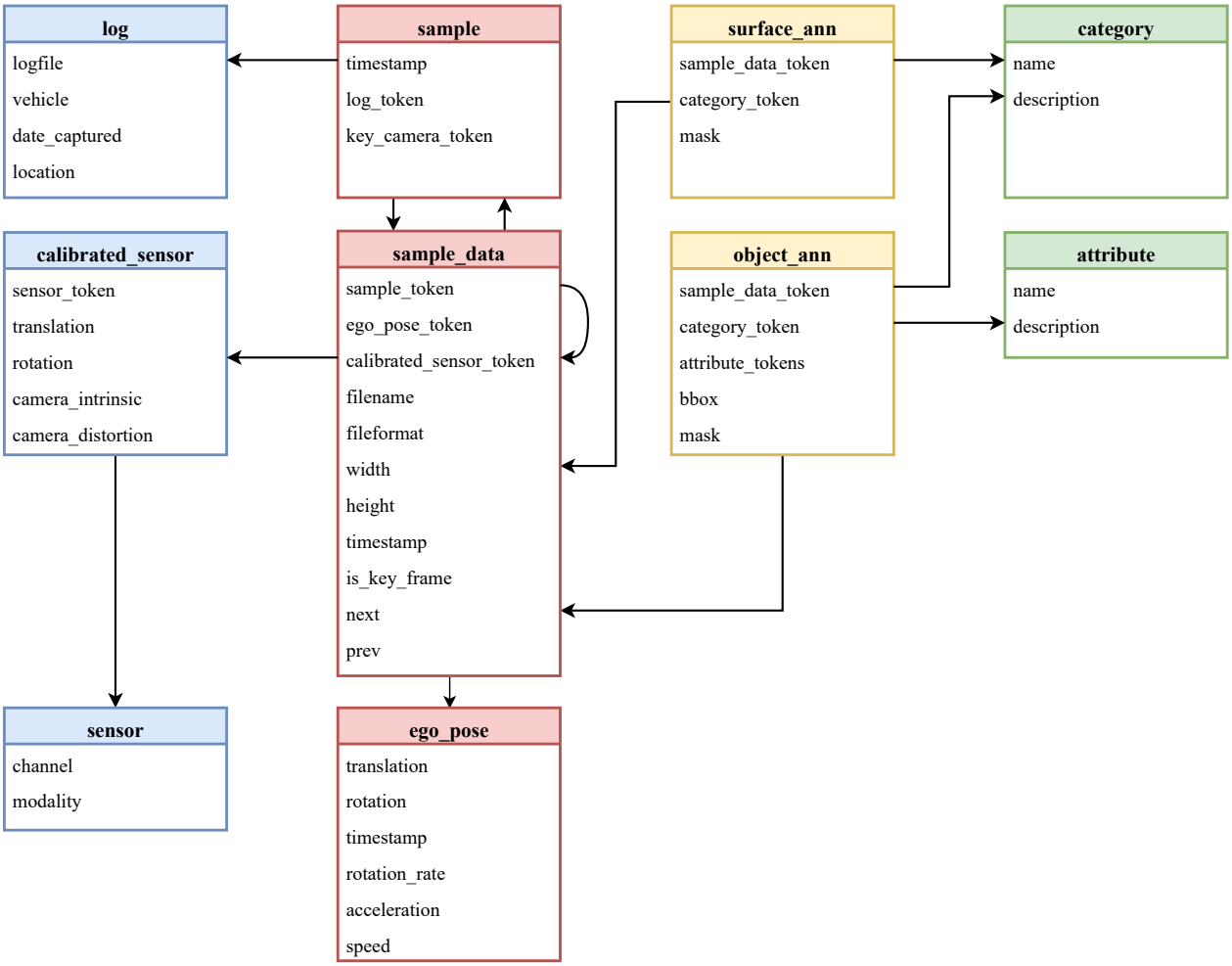

FIGURE 3.1 **Base de données relationnelle organisant les métadonnées de nuImages**. Outre les coordonnées des boites englobantes et l'identifiant des classes, on extrait les informations spatio-temporelles nécessaires à la séparation des échantillons.

et sur Cedar une tâche peut durer jusqu'à 28 jours sans interruption. Afin d'assurer la stabilité du système, les jeux de données utilisés en vision par ordinateur, qui sont typiquement constitués de quelques milliers à plusieurs millions de fichiers de faible volume, doivent être maintenus sous format archivé dans le stockage réseau. Il est alors conseillé de copier et d'extraire les données directement sur les disques locaux des nœuds de calcul, plus performants, afin d'accélérer le traitement. Les opérations ont alors lieu dans un dossier temporaire, dont il convient de transmettre le contenu pertinent, en sens inverse, vers le stockage réseau, en fin de tâche. Une représentation schématique d'une grappe de calcul est donnée à la Figure 3.2. Enfin, pour simuler le système fédéré à travers plusieurs nœuds de calcul, ces nœuds doivent pouvoir échanger des messages entre eux pour permettre les communications client-serveur, et l'entraînement des modèles locaux doit se faire en parallèle. L'outil le plus adapté pour

réaliser cette tâche est MPI, un standard ouvert décrivant un système de passage de messages portable pour architecture de calcul parallèle. MPI permet de traiter efficacement la gestion des processus et des erreurs, ainsi que les communications point-à-point et collectives entre les nœuds de calcul. Notre utilisation de MPI pour simuler les communications dans un système fédéré est explicitée au Chapitre suivant.

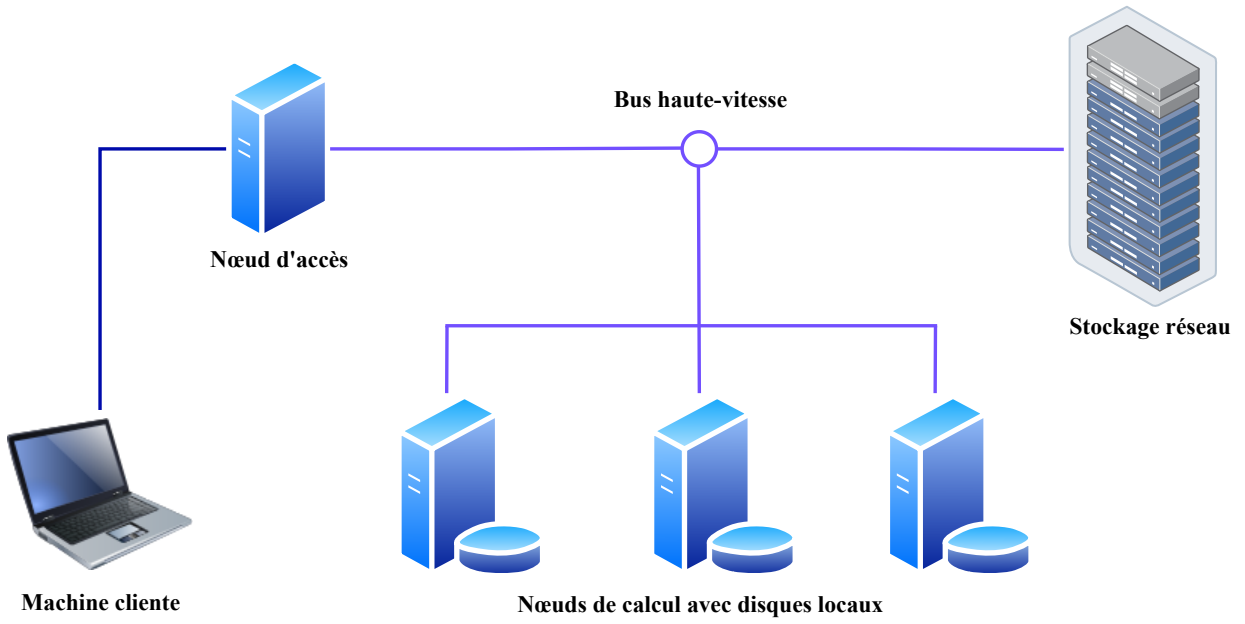

FIGURE 3.2 **Schéma simplifié d'une grappe de calcul**. Le client se connecte au système grâce à un nœud d'accès, tandis que les calculs intensifs sont effectués sur des nœuds dédiés, pouvant contenir des GPUs. Pour entraîner des réseaux de neurones, il est conseillé de transférer les données d'entraînement depuis le stockage réseau vers les disques locaux des nœuds de calcul. Les nœuds et le stockage sont reliés à haut débit, par exemple, via InfiniBand.

Durant le développement, nous avons maintenu deux environnements sous Python 3.9.6. Un environnement local dédié au prototypage, et un autre distant, sur Cedar, pour l'entraînement centralisé ou fédéré des modèles. Le même dépôt GitHub a été utilisé pour synchroniser le code source entre ces deux environnements, tandis que les transferts des jeux de données et des résultats expérimentaux ont été conduits avec Globus [111].

3.3 Implémentation

FedPylot a une organisation simple et construite autour de YOLOv7, l'implémentation officielle de ce dernier étant redistribuée avec des modifications mineures au sein de notre propre code source. Dans notre programme, un ensemble de scripts Bash permettent de

télécharger les poids pré-entraînés de YOLOv7 et soumettre une tâche avec Slurm, tandis que des scripts Python permettent de prétraiter KITTI et nuImages, répartir les jeux de données locaux sur les différents nœuds de calcul de la grappe, et démarrer l'entraînement par apprentissage fédéré. En outre, la logique client-serveur est incluse dans son propre module selon une implémentation orientée objet, pouvant être étendue à l'avenir. Enfin, le compte rendu des mesures de vitesse sur Colab est inclus sur un calepin Jupyter [112] séparé, des fichiers de configuration pour les jeux de données et les hyperparamètres sont sauvegardés dans leur propre dossier, et les fichiers complémentaires (licence, notice de copyright, fichier Lisez-moi et liste des progiciels à installer) sont disponibles à la racine du projet. Le code source de FedPylot est disponible sous licence GPL-3.0 à l'adresse suivante : `https://github.com/cyprienquemeneur/fedpylot`.

3.4 Pour aller plus loin

De par l'aspect multidisciplinaire de l'apprentissage fédéré, notre prototype est nécessairement incomplet. FedPylot continuera d'être utilisé au sein de notre groupe de recherche et a pour vocation d'être enrichi de nouvelles fonctionnalités, selon les besoins qui se manifesteront au cours des projets futurs. Les éléments que nous détaillons ci-après ne font pas partie des aspects essentiels de notre recherche, mais sont des points de départ implémentés en vue de ces prochaines améliorations.

Chiffrement des communications : nous avons mis en place un chiffrement hybride pour protéger les poids du modèle partagé d'adversaires extérieurs au système fédéré durant les échanges entre les clients et le serveur. Cette technique est standard dans de nombreux protocoles tels que TLS et SSH, on peut donc considérer son inclusion comme la plus élémentaire des protections. En revanche, ajouter à FedPylot des méthodes pour résoudre des défis plus spécifiques au contexte fédéré, comme identifier et éliminer un adversaire contrôlant un ou plusieurs clients du système, ou se prémunir contre un agent d'agrégation curieux, pourra être l'objet de recherches futures. Toutefois, implémenter des méthodes de chiffrement, même simples, dans des environnements de calcul à haute performance est complexifié par l'absence de toute prise en charge native par les différentes bibliothèques MPI [113].

Compression des modèles : un champ actif de recherche en apprentissage fédéré est d'utiliser des méthodes de compression de modèles, telle que la quantification ou le traitement de matrices creuses, ou *sparsification*, afin de réduire le coût des communications entre les clients et le serveur tout en limitant les pertes de performance [114]. Dans notre simulation, nous nous sommes limités à l'utilisation de représentations en demi-précision pour les communications, puisqu'il s'agit du niveau de précision utilisé par défaut par YOLOv7 lorsqu'il

est déployé pour inférence.

Optimisation côté serveur : outre FedAvg et FedAvgM, pour lesquels nous présentons des résultats dans notre article, nous avons également implémenté d'autres optimiseurs pour la mise à jour du modèle global, soit FedAdagrad, FedAdam et FedYogi. Nous avons été limités dans notre recherche d'hyperparamètres par nos ressources disponibles, les expérimentations avec FedAvgM ayant partiellement épuisé notre budget de calcul. C'est pourquoi nous avons ultimement décidé d'exclure ces optimiseurs de notre plan d'expériences, bien qu'ils soient supportés par FedPylot.

CHAPITRE 4 ARTICLE 1 : FEDPYLOT : NAVIGATING FEDERATED LEARNING FOR REAL-TIME OBJECT DETECTION IN INTERNET OF VEHICLES

Ce chapitre présente l'article scientifique qui est au cœur de ce mémoire. Celui-ci est intégralement rédigé en anglais.

Auteurs

Cyprien Quéméneur <cyprien.quemeneur@polymtl.ca>
Soumaya Cherkaoui <soumaya.cherkaoui@polymtl.ca>
Département de génie informatique et génie logiciel, École Polytechnique de Montréal.

Soumis pour publication à

IEEE Internet of Things Journal, le 13 mai 2024.

Note sur la contribution

Je suis responsable de la conception de l'article, incluant recherche bibliographique, création de la solution, expérimentation et analyse des résultats. Ma directrice de recherche, seule coautrice, m'a aidé dans la définition et planification de ma recherche. Nous avons cocontribué à la rédaction de l'article.

4.1 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.

4.2 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 [6,7]. 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 Figure 4.1, 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 [9].

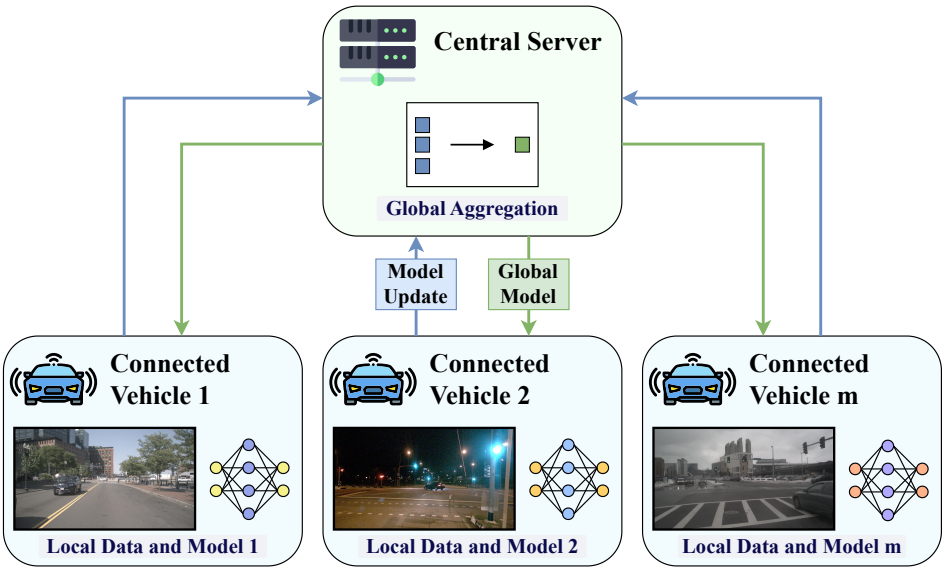

FIGURE 4.1 **Vehicular clients collaboratively learn a joint model with FL**. The vehicles collect driving data using various sensors to train their own local model, while a central server is responsible for regularly gathering and aggregating local weight-update vectors to compute a global model, which 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 [10]. Numerous studies have explored the potential to improve the visual perception capabilities of vehicles with FL, and covered

traffic sign recognition [11–13], pothole and other road damage detection [14–16], semantic segmentation [17, 18], and more commonly object detection [19–34]. Other tasks relying in part but not solely on the visual perception of vehicles have also been explored with FL, such as trajectory prediction [35] and collision avoidance [36]. 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 [42], 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 [81]. 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 granularities 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 [38], PySyft [39], Flower [40]), 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 4.3 outlines the fundamental concepts and tools used in this work. Subsequently, Section 4.4 provides a review of the

current literature surrounding the intersection of object detection, federated learning, and autonomous driving. In Section 4.5, we detail the theoretical design of our FL prototype. Section 4.6 is dedicated to the implementation of FedPylot and our experimental settings. Section 4.7 outlines the results of our experiments. Finally, Section 4.8 concludes the paper.

4.3 Background

4.3.1 Federated learning

Formulation

FL was proposed in 2016 by McMahan et al. [43]. 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

$$\min_{w} \mathcal{L}(w) = \sum_{i=1}^{m} \frac{n_i}{n} L_i(w) \tag{4.1}$$

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

$$w^{t+1} = \sum_{i \in K} \frac{n_i}{n_t} w_i^t. \tag{4.2}$$

Reddi et al. [45] 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, (4.2) is equivalent to

$$w^{t+1} = w^t - \Delta^t = w^t - \sum_{i \in K} \frac{n_i}{n_t} \Delta_i^t \tag{4.3}$$

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 [46], while some implementations leverage blockchain to achieve complete decentralization by eliminating the need for a central server entirely [47] ; one may also assume a vertical partitioning, where the local datasets differ in the feature space instead of the sample space [44].

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. [48] 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 [49]. 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 [50] which introduces a proximal term to each local objective function to improve robustness against variable local updates, SCAF-

FOLD [51] which uses control variates to perform variance reduction and counterbalance the client-drift, and MOON [52] 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 [53]. 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 [55], are orthogonal to the ones mentioned above and can improve convergence in non-IID settings [45]. Similarly, Hsu et al. [49] 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

$$v^{t+1} = \beta v^t + \Delta^t \tag{4.4a}$$

$$w^{t+1} = w^t - \eta v^{t+1} \tag{4.4b}$$

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 [58,59]. 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 [60, 61].

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 [62], homomorphic encryption [63], and differential privacy [64] can help alleviate this problem, but may incur increased computation and communication overheads or performance degradation.

4.3.2 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 [66]. 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 [73] and SPP-Net [74], first generate region proposals before sending them to a classification model, while one-stage object detectors, such as YOLO [75], SSD [76] and RetinaNet [77], 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) [67]. 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 [93].

The most popular metrics used to evaluate the predictive performance of object detectors are reviewed by Padilla et al. [71], 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

$$\text{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{4.5}$$

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

$$\text{mAP}_t = \frac{1}{N} \sum_{k=1}^{N} \text{AP}_t^k. \tag{4.6}$$

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.

4.3.3 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 [92]. YOLOv7 was trained from scratch on MS COCO [72] 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 [84] 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 [83]. YOLOv7 also introduced architectural improvements, including Extended-ELAN, a modified variant of the Efficient Layer Aggregation Network (ELAN) [85] 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) [86]. 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 [87]. 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 [88], 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 [90] and YOLOv8 [91]. YOLOv7 was used as the basis for YOLOv9 [94], which introduced Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN).

4.4 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. [19] investigated federated real-time object detection in adverse weather driving scenarios with the original YOLO model. Chen et al. [21] studied FL in vehicular object detection with SSD under communication and computation constraints. Bommel [22] and later Rjoub et al. [23] 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. [24] 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 [79]. Wang et al. [25] 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. [26] leveraged a soft teacher semi-supervised object detection framework to perform FL training with Faster-R-CNN [78] on unlabeled data collected while driving, given a small amount of well-curated preexisting data. Rao et al. [27] proposed FedWeg, a sparse FL training process that they evaluated on YOLOv3 [80], to accommodate computational and communication constraints in IoV. Su et al. [28] 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. [29] 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. [30] 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. [31] 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 [82]. Moreover, Chi et al. [32] 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 4.6.3, 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 [33], and an adaptive clustered FL technique to address storage and bandwidth limitations, validated for car detection under varying weather and lighting conditions [34]. 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. [20] 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.

4.5 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 (4.4), although FedPylot also supports FedAdagrad, FedAdam and FedYogi [45]. 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.

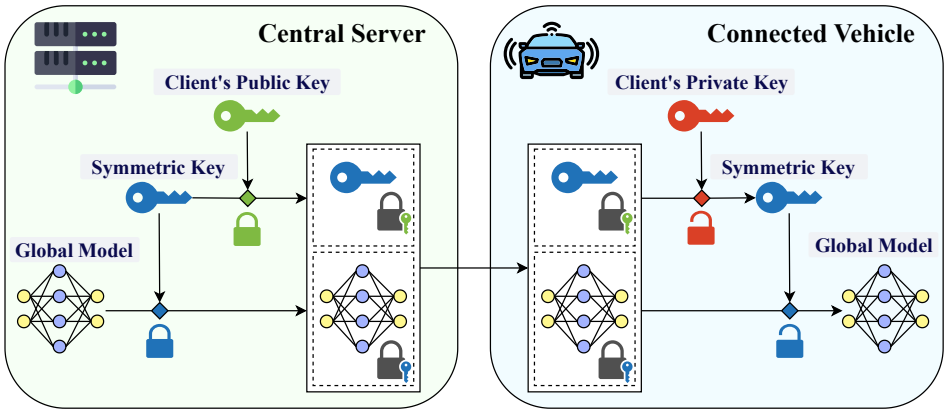

FIGURE 4.2 **Hybrid cryptosystem for server-client communications**. Transmissions between the server and the vehicular clients are encrypted using a highly efficient symmetric algorithm. The symmetric key is generated by the server and protected using a public-key cryptosystem when passed to the clients.

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-

Algorithm 3 Federated YOLOv7

Input : Communication rounds R, local epochs E, local mini-batch size B, server learning rate η, server momentum factor $\beta \in [0, 1)$, YOLOv7 set of hyperparameters $\mathcal{H}$.

Output : Best re-parameterized joint model w_{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
    $\Delta^t = \sum_{i=1}^{m} \frac{n_i}{n} \Delta_i^t$
    $v^{t+1} = \beta v^t + \Delta^t$
    $w^{t+1} = w^t - \eta v^{t+1}$
    $w_{\mathrm{r}}^{t+1} = \texttt{reparameterize}(w^{t+1})$
    Evaluation metrics = $\texttt{test}(w_{\mathrm{r}}^{t+1}, \mathcal{D}_{\mathcal{S}}, \mathcal{H})$
  end for
  return w_{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
  $w_i^t = \texttt{train}(w^t, \mathcal{D}_i, E, B, \mathcal{H}, t)$
  $\Delta_i^t = w^t - w_i^t$
  $\{\Delta_i^t\} \leftarrow$ Encrypt Δ_i^t with sk^t
  return $\{\Delta_i^t\}$, $|\mathcal{D}_i|$ to server

---

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 Figure 4.2, 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 4.3.1, becomes mandatory. The federated procedure is summarized in Algorithm 3, while the remaining details are covered in the next Section.

4.6 Experiments

4.6.1 Prototype implementation details

Federated simulation

Our experiments were carried out with PyTorch [115] 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 [109], and we handled the communications between the server and the clients with MPI, using the Open MPI implementation of the standard [116], and the mpi4py Python package [117]. 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 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.

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 [118] with the OAEP [119] padding scheme, while data encapsulation was handled with a 256-bit symmetric key using the Advanced Encryption Standard (AES) algorithm [120] with Galois/Counter Mode (GCM) [121], and with scrypt as the password-based key derivation

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

function [122]. 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.

4.6.2 Datasets

We conducted experiments on two relevant autonomous driving datasets : the 2D object detection subset of the KITTI Vision Benchmark Suite [96] and nuImages, an extension of nuScenes dedicated to 2D object detection [99]. 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 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 Figure 4.3, 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.

4.6.3 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 homoge-

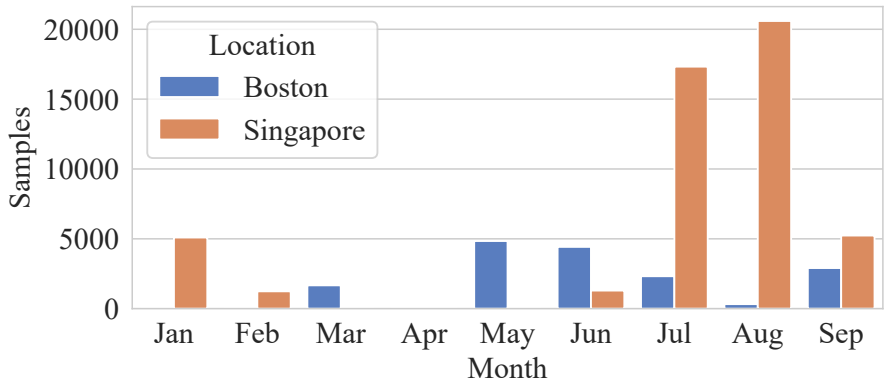

FIGURE 4.3 **Distribution of samples in nuImages training data.** The dataset is composed of nearly 500 driving logs acquired through six different cameras, and features a wide range of driving scenarios and weather conditions

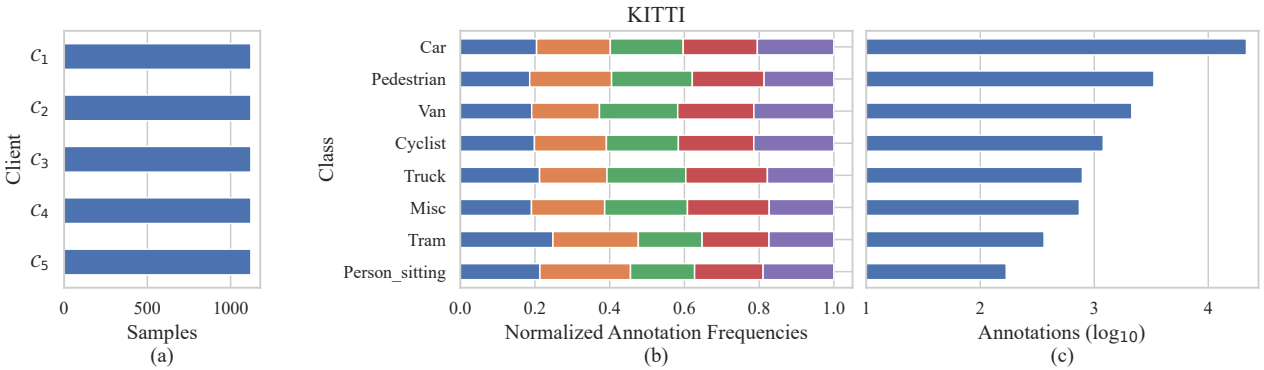

FIGURE 4.4 **KITTI split.** 25% of KITTI training data are stored on the server, while the 75% left are distributed IID among five clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

neously among five clients, as shown in Figure 4.4. 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 Singa-

pore. 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 Figure 4.5 and Figure 4.6, 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.

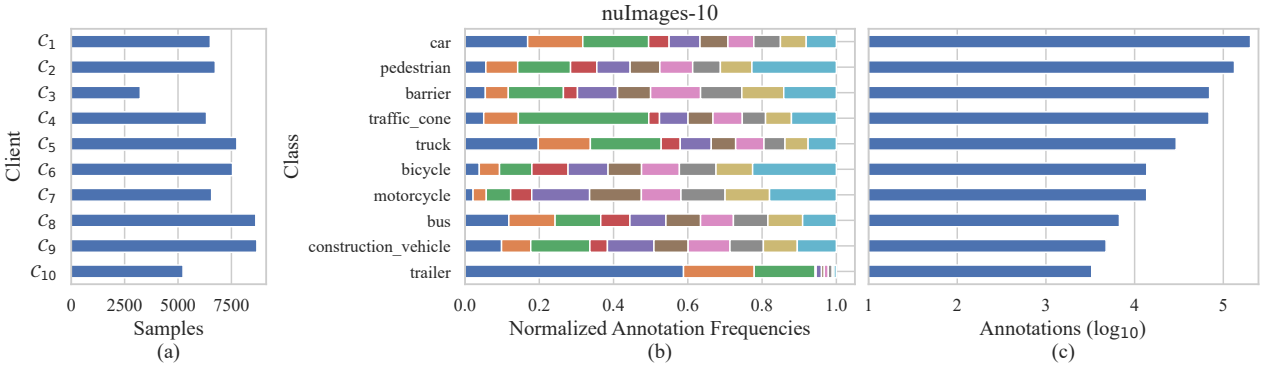

FIGURE 4.5 **nuImages-10 split.** The original classes are mapped to ten labels, and the training data are split non-IID among ten clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

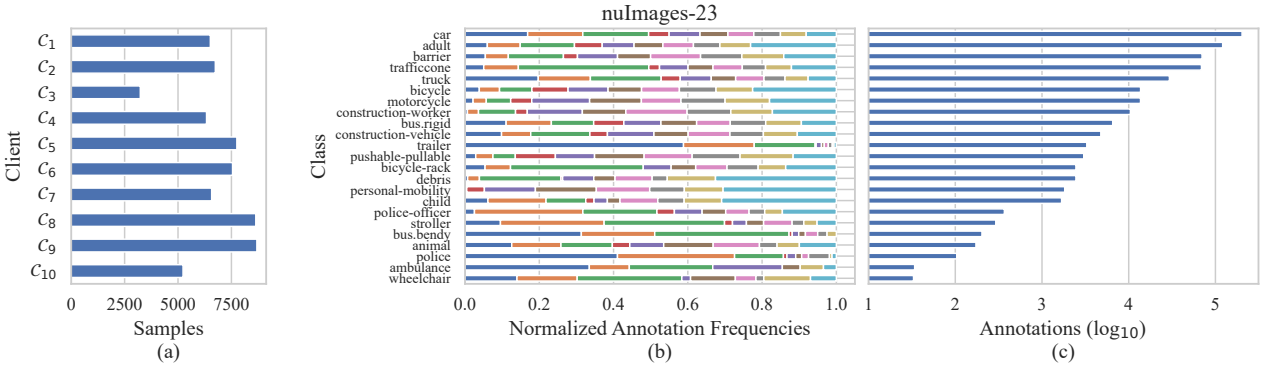

FIGURE 4.6 **nuImages-23 split.** The full long-tail distribution is retained, and the training data are split non-IID among ten clients. We report (a) the number of samples held by each client, (b) the normalized number of annotations held per sample per client, and (c) the label distribution in the original training set.

4.6.4 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.

Lastly, we performed ablation with the FedAvgM (4.4) 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.

4.7 Results

4.7.1 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 Figure 4.7, and we report the highest testing accuracy achieved for each setting in Table 4.1. In the federated setting, the aggregated training loss is derived from the weighted average of the local losses computed by each client following (4.1), and the mAP is 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 3. 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.

Tableau 4.1 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%

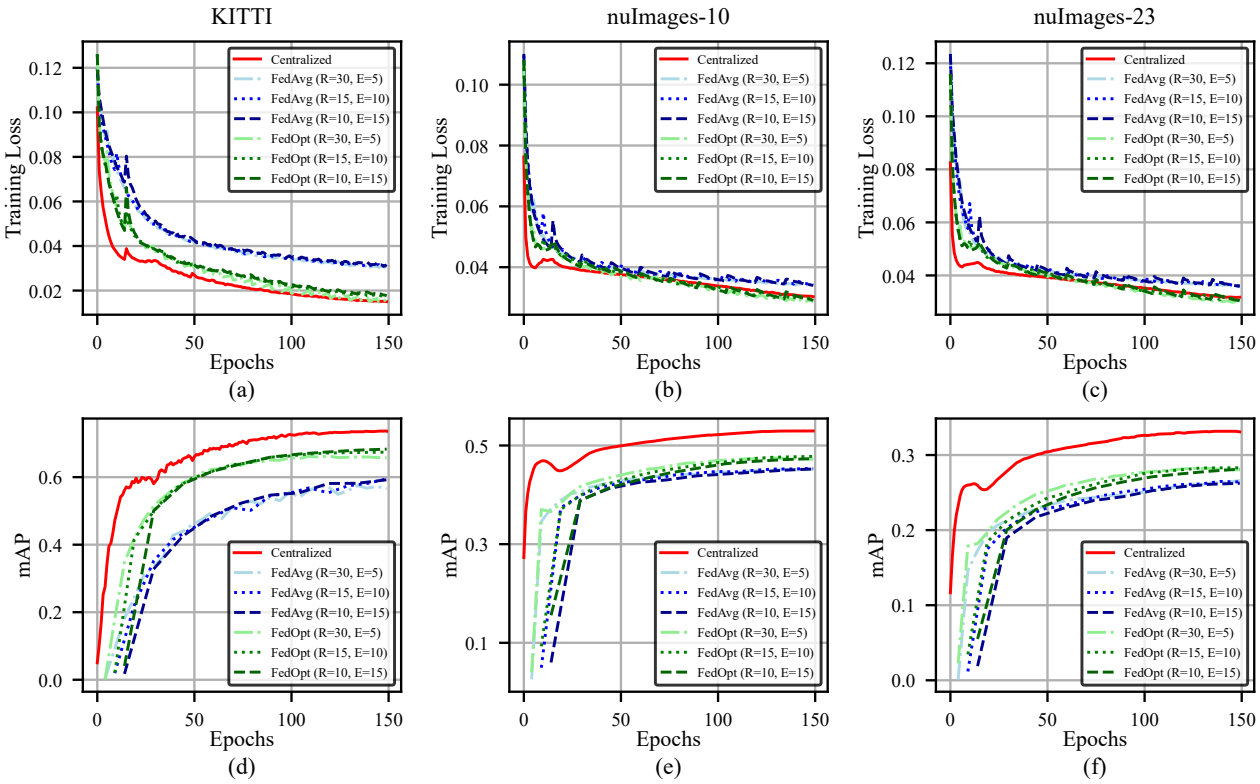

FIGURE 4.7 **Client-side optimization results**. Three round lengths were considered for the federated baseline FedAvg and the locally optimized algorithm FedOpt. We report the evolution for the centralized and federated settings of YOLOv7's (a) training loss on KITTI, (b) training loss on nuImages-10, (c) training loss on nuImages-23, (d) testing accuracy on KITTI, (e) testing accuracy on nuImages-10, and (f) testing accuracy on nuImages-23.

4.7.2 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 Figure 4.8, 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 training process and led to inaccurate predictions, especially on non-IID data. A qualitative comparison between FedAvg and FedOptM is shown in Figure 4.9.

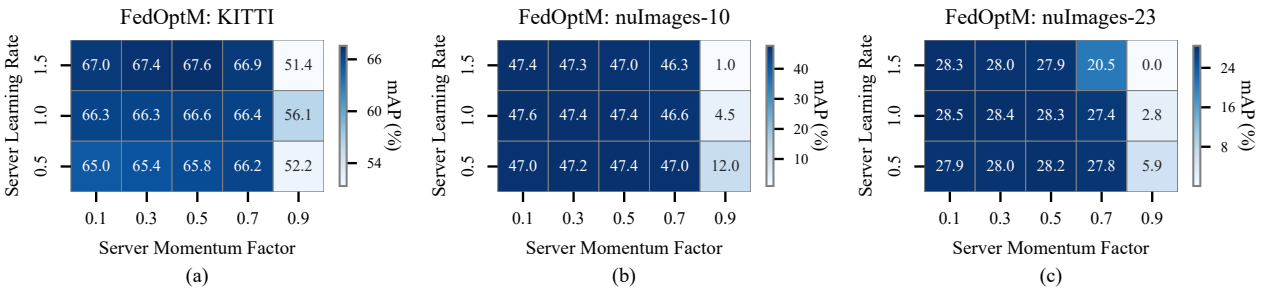

FIGURE 4.8 **Grid search on FedOptM server-side hyperparameters**. Training lasted 30 rounds of 5 epochs on (a) KITTI, (b) nuImages-10, and (c) nuImages-23.

FIGURE 4.9 **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).

4.7.3 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 4.2. 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. 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.

Tableau 4.2 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

4.8 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.

4.9 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.

CHAPITRE 5 CONCLUSION

Le présent chapitre conclut ce mémoire. Nous y proposons une synthèse de nos travaux de recherche, puis nous faisons état des limites de notre contribution. Enfin, nous discutons de plusieurs pistes d'amélioration qui pourront être poursuivies à l'avenir. Des compléments à nos résultats expérimentaux sont également fournis en annexe, en fin de document.

5.1 Synthèse des travaux

Nous avons montré comment, en se reposant sur l'Internet des véhicules, l'apprentissage fédéré propose une solution viable pour entraîner à grande échelle des modèles d'apprentissage automatique pour la conduite autonome, tout en préservant la vie privée des usagers de la route. Notre plus importante contribution est la conception de FedPylot, un prototype pour l'optimisation fédérée de détecteurs d'objets en temps réel, déployable sur des grappes de calcul et reposant sur MPI. Pour le moment, FedPylot intègre uniquement YOLOv7, un modèle de l'état de l'art, dont nous avons fait valoir la robustesse de l'entraînement, y compris sous des conditions statistiquement hétérogènes marquées par la dérive conceptuelle, l'asymétrie dans la distribution des labels et le déséquilibre dans la taille des jeux de données locaux. À cette fin, nous avons conduit une analyse expérimentale exhaustive sur des jeux de données pertinents pour la conduite autonome, soit KITTI pour le cas IID, et nuImages pour le cas non-IID. Dans ce second cas, nous avons réparti les données selon leur origine spatio-temporelle et les séquences de navigation où elles ont été collectées. FedPylot intègre de surcroit plusieurs optimiseurs globaux, dont FedAvgM, pour lequel nous avons étudié l'impact des hyperparamètres, ainsi que deux méthodes d'optimisation locales, dont nous avons mesuré comment elles affectent la convergence de l'entraînement. En outre, non avons non seulement considéré les performances prédictives de plusieurs variantes de YOLOv7, mais nous avons également intégré les questions de vitesse d'inférence et de coût associé aux communications dans notre évaluation. Nous avons enfin esquissé la possibilité d'améliorer les communications de FedPylot en termes d'efficacité et de sécurité, en commençant par inclure des conversions en demi-précision pré-transfert et en implémentant un chiffrement hybride reposant sur AES-GCM et RSA. En définitive, nous espérons que FedPylot incitera et facilitera de plus amples recherches au croisement de l'apprentissage fédéré et de la détection d'objets.

5.2 Limitations de la solution proposée

Durant cette recherche, nous avons testé deux méthodes pour l'optimisation locale de nos modèles. La première était une simple descente stochastique du gradient, la seconde, qui est utilisée par défaut dans l'implémentation de YOLOv7, inclut notamment un moment de Nesterov ainsi qu'un ordonnancement complexe différenciant trois taux d'apprentissage. Nous avons tenu à tester cette méthode pour étudier son impact sur la convergence des modèles en fonction du niveau d'hétérogénéité statistique. Néanmoins, elle nous a amenés à formuler trois hypothèses simplificatrices, que nous avons conservées pour toutes nos simulations :

— **États persistants :** nous avons supposé que les véhicules sont toujours capables de maintenir leur état interne d'un cycle de communication à l'autre, cela implique notamment que durant toute la participation d'un client, celui-ci ne doit jamais subir un échec. Sans cette hypothèse, il n'est pas à priori possible de recourir au moment dans l'optimisation locale sans requérir des informations supplémentaires de la part du serveur, et donc augmenter le coût des communications.

— **Participation totale :** nous avons supposé que tous les clients participaient à chaque cycle de communication, ainsi les gradients locaux des clients sont toujours récents et on n'observe pas de dégradation de performance lorsque le moment est appliqué durant l'optimisation locale. En pratique, lorsque le nombre de clients est important, il est préférable que seule une fraction de ceux-ci participe à un cycle donné.

— **Synchronicité :** durant un cycle de communication, nous attendons que tous les clients aient complètement terminé leur entraînement local avant de procéder à l'agrégation des mises à jour, ce qui permet de synchroniser les ordonnanceurs locaux des clients. En situation réelle, un déséquilibre important dans la taille des jeux de données locaux ou dans la puissance de calcul des clients conduirait, sous cette hypothèse, à une perte d'efficacité de tout le système.

Nous avons également supposé que les jeux de données locaux utilisés durant l'entraînement étaient entièrement labellisés. Cette hypothèse est peu réaliste puisqu'en pratique les véhicules collectent régulièrement de nouvelles données d'entraînement pendant la conduite. Cependant, entraîner un modèle sur des données partiellement étiquetées est, comme nous l'avons montré dans les Sections 2.3 et 4.4, déjà l'un des sujets les plus couverts dans la littérature, aussi, nous n'avons pas jugé bon d'en faire un point essentiel de notre recherche. Par ailleurs, notre solution ne prend pas en charge présentement certaines fonctionnalités désirables pour simuler l'apprentissage fédéré dans l'Internet des véhicules. En particulier, notre simulateur ne supporte pas les organisations fédérées hiérarchiques, et ne tire pas non plus parti des données multimodales collectées par les véhicules, puisque nous n'avons considéré

que des données vidéo pour entraîner notre modèle. Enfin, l'apprentissage fédéré étant un large champ de recherche multidisciplinaire, il existe naturellement de nombreuses possibilités d'extension qui n'étaient pas le but essentiel de ce mémoire et qui complèteraient notre solution. On pourra citer l'inclusion de méthodes avancées de sécurisation et de conservation de la vie privée, de techniques avancées de compression pour réduire la charge sur le réseau, de moyens incitatifs pour encourager les clients disposant du plus de données locales à intégrer le processus d'optimisation fédérée, ou encore de méthodes favorisant la convergence vers un modèle central garantissant l'équité entre les clients véhiculaires.

5.3 Améliorations futures

Nous prévoyons de continuer à maintenir et à améliorer FedPylot dans le futur. Nous pensons que tant que les cadriciels majeurs d'apprentissage fédéré ne faciliteront pas l'intégration de bibliothèques comprenant des détecteurs d'objets de l'état de l'art, notre prototype continuera, en dépit de sa simplicité, de présenter une bonne entrée en matière dans ce domaine, et donc une plus-value pour la communauté des chercheurs.

À moyen terme, nous envisageons d'ajouter YOLOv9 à FedPylot et de travailler à relâcher les hypothèses limitantes citées ci-dessus, afin de rendre notre prototype plus flexible et à même de simuler des situations plus réalistes. À plus long terme, FedPylot continuera d'être utilisé au sein de notre laboratoire et sera amélioré par les prochaines générations d'étudiants de Lincs. Parmi les projets prévus, on citera l'ajout d'un second protocole de communication, plus flexible que MPI, permettant d'expérimenter sur une mini-flotte de véhicules de laboratoire (en plus de conserver le module principal pour l'entraînement sur grappes de calcul). Également, un second volet de recherche prévu se concentrera sur la confidentialité différentielle et le chiffrement homomorphe, dans le cadre de la détection d'objets par les véhicules.

RÉFÉRENCES

[1] W. Xu *et al.*, "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, n°. 1, p. 19–35, janv. 2018.

[2] C. R. Storck et 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, p. 117 593–117 614, 2020.

[3] S. Grigorescu *et al.*, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, n°. 3, p. 362–386, 2020.

[4] P. Voigt et A. v. d. Bussche, *The EU General Data Protection Regulation (GDPR) : A Practical Guide*, 1er éd. Springer Publishing Company, Incorporated, juill. 2017.

[5] D. C. Nguyen *et al.*, "Federated learning for Internet of Things : A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, n°. 3, p. 1622–1658, 2021.

[6] Y. Lu *et al.*, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, n°. 4, p. 4298–4311, avr. 2020.

[7] D. M. Manias et A. Shami, "Making a case for federated learning in the Internet of Vehicles and intelligent transportation systems," *IEEE Network*, vol. 35, n°. 3, p. 88–94, mai 2021.

[8] D. C. Nguyen *et al.*, "Federated learning for smart healthcare : A survey," *ACM Computing Surveys*, vol. 55, n°. 3, p. 60 :1–60 :37, févr. 2022. [En ligne]. Disponible : https://doi.org/10.1145/3501296

[9] X. Zhou *et al.*, "Two-layer federated learning with heterogeneous model aggregation for 6G supported Internet of Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, n°. 6, p. 5308–5317, juin 2021.

[10] J. Janai *et al.*, "Computer vision for autonomous vehicles : Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, n°. 1–3, p. 1–308, juill. 2020.

[11] K. Xie *et al.*, "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Transactions on Vehicular Technology*, vol. 71, n°. 9, p. 9980–9992, sept. 2022.

[12] A. A. Padaria *et al.*, "Traffic sign classification for autonomous vehicles using split and federated learning underlying 5G," *IEEE Open Journal of Vehicular Technology*, vol. 4, p. 877–892, 2023.

[13] Z. Lian *et al.*, “Traffic sign recognition using optimized federated learning in Internet of Vehicles,” *IEEE Internet of Things Journal*, vol. 11, n°. 4, p. 6722–6729, févr. 2024.

[14] Y. Yuan *et al.*, “FedRD : Privacy-preserving adaptive federated learning framework for intelligent hazardous road damage detection and warning,” *Future Generation Computer Systems*, vol. 125, p. 385–398, déc. 2021.

[15] S. Alshammari et S. Song, “3Pod : Federated learning-based 3 dimensional pothole detection for smart transportation,” dans *2022 IEEE International Smart Cities Conference (ISC2)*, sept. 2022, p. 1–7.

[16] P. K. Saha, D. Arya et Y. Sekimoto, “Federated learning–based global road damage detection,” *Computer-Aided Civil and Infrastructure Engineering*, 2024.

[17] L. Fantauzzo *et al.*, “FedDrive : Generalizing federated learning to semantic segmentation in autonomous driving,” dans *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, oct. 2022, p. 11 504–11 511.

[18] E. Fanì, M. Ciccone et B. Caputo, “FedDrive v2 : An analysis of the impact of label skewness in federated semantic segmentation for autonomous driving,” oct. 2023. [En ligne]. Disponible : http://arxiv.org/abs/2309.13336

[19] G. Rjoub *et al.*, “Improving autonomous vehicles safety in snow weather using federated YOLO CNN learning,” dans *Mobile Web and Intelligent Information Systems*. Springer International Publishing, 2021, p. 121–134.

[20] D. Jallepalli *et al.*, “Federated learning for object detection in autonomous vehicles,” dans *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, août 2021, p. 107–114.

[21] Y. Chen, C. Wang et B. Kim, “Federated learning with infrastructure resource limitations in vehicular object detection,” dans *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, déc. 2021, p. 366–370.

[22] J. Bommel, “Active learning during federated learning for object detection,” juill. 2021. [En ligne]. Disponible : https://essay.utwente.nl/86855/

[23] G. Rjoub, J. Bentahar et Y. A. Joarder, “Active federated YOLOR model for enhancing autonomous vehicles safety,” dans *Mobile Web and Intelligent Information Systems*. Springer International Publishing, 2022, p. 49–64.

[24] S. Dai *et al.*, “Online federated learning based object detection across autonomous vehicles in a virtual world,” dans *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, janv. 2023, p. 919–920.

[25] S. Wang *et al.*, “Federated deep learning meets autonomous vehicle perception : Design and verification,” *IEEE Network*, vol. 37, n°. 3, p. 16–25, mai 2023.

[26] F. Chi *et al.*, “Federated semi-supervised learning for object detection in autonomous driving,” dans *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, juin 2023, p. 1–5.

[27] L. Rao *et al.*, “Sparse federated training of object detection in the Internet of Vehicles,” dans *ICC 2023 - IEEE International Conference on Communications*, mai 2023, p. 1768–1773.

[28] S. Su *et al.*, “Cross-domain federated object detection,” dans *2023 IEEE International Conference on Multimedia and Expo (ICME)*, juill. 2023, p. 1469–1474.

[29] T. Kim *et al.*, “Navigating data heterogeneity in federated learning : A semi-supervised approach for object detection,” dans *Thirty-seventh Conference on Neural Information Processing Systems*, nov. 2023.

[30] T. Zheng *et al.*, “AutoFed : Heterogeneity-aware federated multimodal learning for robust autonomous driving,” dans *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, juill. 2023, n°. 15, p. 1–15.

[31] A. Mishra *et al.*, “Swarm learning in autonomous driving : A privacy preserving approach,” dans *Proceedings of the 2023 15th International Conference on Computer Modeling and Simulation*. Association for Computing Machinery, août 2023, p. 271–277.

[32] F. Chi *et al.*, “Federated cooperative 3D object detection for autonomous driving,” dans *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, sept. 2023, p. 1–6.

[33] A. Khalil *et al.*, “Federated learning with heterogeneous data handling for robust vehicular object detection,” mai 2024. [En ligne]. Disponible : http://arxiv.org/abs/2405.01108

[34] ——, “Driving towards efficiency : Adaptive resource-aware clustered federated learning in vehicular networks,” *The 22nd Mediterranean Communication and Computer Networking Conference (MedComNet'24)*, avr. 2024.

[35] M. Han *et al.*, “Federated learning-based trajectory prediction model with privacy preserving for intelligent vehicle,” *International Journal of Intelligent Systems*, vol. 37, n°. 12, p. 10 861–10 879, 2022.

[36] R. Yu *et al.*, “Personalized driving assistance algorithms : Case study of federated learning based forward collision warning,” *Accident Analysis & Prevention*, vol. 168, p. 106609, avr. 2022.

[37] The TensorFlow Federated Authors, “TensorFlow Federated,” déc. 2018. [En ligne]. Disponible : https://github.com/google-parfait/tensorflow-federated

[38] C. He *et al.*, “FedML : A research library and benchmark for federated machine learning,” nov. 2020. [En ligne]. Disponible : http://arxiv.org/abs/2007.13518

[39] A. Ziller *et al.*, “PySyft : A library for easy federated learning,” dans *Federated Learning Systems : Towards Next-Generation AI*. Springer International Publishing, 2021, p. 111–139.

[40] D. J. Beutel *et al.*, “Flower : A friendly federated learning research framework,” mars 2022. [En ligne]. Disponible : http://arxiv.org/abs/2007.14390

[41] Y. Liu *et al.*, “FATE : an industrial grade platform for collaborative learning with data protection,” *Journal of Machine Learning Research*, vol. 22, n°. 1, p. 226 :10 320–226 :10 325, janv. 2021.

[42] C.-Y. Wang, A. Bochkovskiy et H.-Y. M. Liao, “YOLOv7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, juin 2023, p. 7464–7475.

[43] B. McMahan *et al.*, “Communication-efficient learning of deep networks from decentralized data,” dans *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, avr. 2017, p. 1273–1282.

[44] Q. Yang *et al.*, “Federated machine learning : Concept and applications,” *ACM Transactions on Intelligent Systems and Technology*, vol. 10, n°. 2, p. 12 :1–12 :19, janv. 2019.

[45] S. J. Reddi *et al.*, “Adaptive federated optimization,” dans *International Conference on Learning Representations*, 2021.

[46] A. Z. Tan *et al.*, “Towards personalized federated learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, n°. 12, p. 9587–9603, déc. 2023.

[47] D. C. Nguyen *et al.*, “Federated learning meets blockchain in edge computing : Opportunities and challenges,” *IEEE Internet of Things Journal*, vol. 8, n°. 16, p. 12 806–12 825, août 2021.

[48] P. Kairouz *et al.*, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, n°. 1–2, p. 1–210, juin 2021.

[49] T.-M. H. Hsu, H. Qi et M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” sept. 2019. [En ligne]. Disponible : http://arxiv.org/abs/1909.06335

[50] T. Li *et al.*, “Federated optimization in heterogeneous networks,” *Proceedings of Machine Learning and Systems*, p. 429–450, mars 2020.

[51] S. P. Karimireddy *et al.*, “SCAFFOLD : Stochastic controlled averaging for federated learning,” dans *Proceedings of the 37th International Conference on Machine Learning*. PMLR, nov. 2020, p. 5132–5143.

[52] Q. Li, B. He et D. Song, “Model-contrastive federated learning,” dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, juin 2021, p. 10 713–10 722.

[53] J. Miao *et al.*, “FedSeg : Class-heterogeneous federated learning for semantic segmentation,” dans *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, juin 2023, p. 8042–8052.

[54] J. Duchi, E. Hazan et Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *The Journal of Machine Learning Research*, vol. 12, p. 2121–2159, juill. 2011.

[55] D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization,” janv. 2017. [En ligne]. Disponible : http://arxiv.org/abs/1412.6980

[56] M. Zaheer *et al.*, “Adaptive methods for nonconvex optimization,” dans *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[57] X. Li *et al.*, “FedBN : Federated learning on non-IID features via local batch normalization,” dans *International Conference on Learning Representations*, oct. 2020.

[58] W. Liu *et al.*, “Accelerating federated learning via momentum gradient descent,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, nº. 8, p. 1754–1766, août 2020.

[59] J. Xu *et al.*, “FedCM : Federated learning with client-level momentum,” juin 2021. [En ligne]. Disponible : http://arxiv.org/abs/2106.10874

[60] H.-Y. Chen *et al.*, “On the importance and applicability of pre-training for federated learning,” dans *The Eleventh International Conference on Learning Representations*, sept. 2022.

[61] J. Nguyen *et al.*, “Where to begin ? On the impact of pre-training and initialization in federated learning,” dans *The Eleventh International Conference on Learning Representations*, sept. 2022.

[62] A. C.-C. Yao, “How to generate and exchange secrets,” dans *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, oct. 1986, p. 162–167.

[63] C. Gentry, “Fully homomorphic encryption using ideal lattices,” dans *Proceedings of the forty-first annual ACM symposium on Theory of computing*. Association for Computing Machinery, mai 2009, p. 169–178.

[64] C. Dwork *et al.*, “Calibrating noise to sensitivity in private data analysis,” dans *Theory of Cryptography*. Springer Berlin Heidelberg, 2006, p. 265–284.

[65] P. Riedel *et al.*, “Comparative analysis of open-source federated learning frameworks - a literature-based survey and review,” *International Journal of Machine Learning and Cybernetics*, juin 2024.

[66] S. S. A. Zaidi *et al.*, “A survey of modern deep learning based object detection models,” *Digital Signal Processing*, vol. 126, p. 103514, juin 2022.

[67] N. Carion *et al.*, “End-to-end object detection with transformers,” dans *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, p. 213–229.

[68] I. J. Goodfellow, Y. Bengio et A. Courville, *Deep Learning*. Cambridge, MA, USA : MIT Press, 2016.

[69] A. Vaswani *et al.*, “Attention is all you need,” dans *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[70] A. Dosovitskiy *et al.*, “An image is worth 16x16 words : Transformers for image recognition at scale,” dans *International Conference on Learning Representations*, oct. 2020.

[71] R. Padilla, S. L. Netto et E. A. B. da Silva, “A survey on performance metrics for object-detection algorithms,” dans *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, juill. 2020, p. 237–242.

[72] T.-Y. Lin *et al.*, “Microsoft COCO : Common objects in context,” dans *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, p. 740–755.

[73] R. Girshick *et al.*, “Rich feature hierarchies for accurate object detection and semantic segmentation,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, juin 2014, p. 580–587.

[74] K. He *et al.*, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, n°. 9, p. 1904–1916, sept. 2015.

[75] J. Redmon *et al.*, “You only look once : Unified, real-time object detection,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, juin 2016, p. 779–788.

[76] W. Liu *et al.*, “SSD : Single shot MultiBox detector,” dans *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, p. 21–37.

[77] T.-Y. Lin *et al.*, “Focal loss for dense object detection,” dans *Proceedings of the IEEE International Conference on Computer Vision*, 2017, p. 2980–2988.

[78] S. Ren *et al.*, “Faster R-CNN : Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, n°. 6, p. 1137–1149, juin 2017.

[79] J. Redmon et A. Farhadi, “YOLO9000 : Better, faster, stronger,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 7263–7271.

[80] ——, “YOLOv3 : An incremental improvement,” avr. 2018. [En ligne]. Disponible : http://arxiv.org/abs/1804.02767

[81] J. Terven, D.-M. Córdova-Esparza et J.-A. Romero-González, “A comprehensive review of YOLO architectures in computer vision : From YOLOv1 to YOLOv8 and YOLO-NAS,” *Machine Learning and Knowledge Extraction*, vol. 5, n°. 4, p. 1680–1716, déc. 2023.

[82] M. Simon *et al.*, “Complex-YOLO : An Euler-region-proposal for real-time 3D object detection on point clouds,” dans *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019, p. 197–209.

[83] A. Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao, “YOLOv4 : Optimal speed and accuracy of object detection,” avr. 2020. [En ligne]. Disponible : http://arxiv.org/abs/2004.10934

[84] C.-Y. Wang, I.-H. Yeh et H.-Y. M. Liao, “You only learn one representation : Unified network for multiple tasks,” *Journal of Information Science and Engineering*, vol. 39, n°. 3, p. 691–709, mai 2023.

[85] C.-Y. Wang, H.-Y. M. Liao et I.-H. Yeh, “Designing network design strategies through gradient path analysis,” nov. 2022. [En ligne]. Disponible : http://arxiv.org/abs/2211.04800

[86] T.-Y. Lin *et al.*, “Feature pyramid networks for object detection,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 2117–2125.

[87] Z. Zheng *et al.*, “Distance-IoU loss : Faster and better learning for bounding box regression,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, p. 12 993–13 000, avr. 2020.

[88] Z. Ge *et al.*, “YOLOX : Exceeding YOLO series in 2021,” août 2021. [En ligne]. Disponible : http://arxiv.org/abs/2107.08430

[89] C. Lyu *et al.*, “RTMDet : An empirical study of designing real-time object detectors,” déc. 2022. [En ligne]. Disponible : http://arxiv.org/abs/2212.07784

[90] C. Li *et al.*, “YOLOv6 v3.0 : A full-scale reloading,” janv. 2023. [En ligne]. Disponible : http://arxiv.org/abs/2301.05586

[91] G. Jocher, A. Chaurasia et J. Qiu, “Ultralytics YOLO,” janv. 2023. [En ligne]. Disponible : https://github.com/ultralytics/ultralytics

[92] G. Jocher, “YOLOv5 by Ultralytics,” mai 2020. [En ligne]. Disponible : https://github.com/ultralytics/yolov5

[93] W. Lv *et al.*, “DETRs beat YOLOs on real-time object detection,” juill. 2023. [En ligne]. Disponible : http://arxiv.org/abs/2304.08069

[94] C.-Y. Wang, I.-H. Yeh et H.-Y. M. Liao, “YOLOv9 : Learning what you want to learn using programmable gradient information,” févr. 2024. [En ligne]. Disponible : http://arxiv.org/abs/2402.13616

[95] A. Wang *et al.*, “YOLOv10 : Real-time end-to-end object detection,” mai 2024. [En ligne]. Disponible : http://arxiv.org/abs/2405.14458

[96] A. Geiger, P. Lenz et R. Urtasun, “Are we ready for autonomous driving ? The KITTI vision benchmark suite,” dans *2012 IEEE Conference on Computer Vision and Pattern Recognition*, juin 2012, p. 3354–3361.

[97] M. Cordts *et al.*, “The Cityscapes dataset for semantic urban scene understanding,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, p. 3213–3223.

[98] F. Yu *et al.*, “BDD100K : A diverse driving dataset for heterogeneous multitask learning,” dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, p. 2636–2645.

[99] H. Caesar *et al.*, “nuScenes : A multimodal dataset for autonomous driving,” dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, p. 11 621–11 631.

[100] D. Barnes *et al.*, “The Oxford Radar RobotCar dataset : A radar extension to the Oxford RobotCar dataset,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, mai 2020, p. 6433–6438.

[101] W. Maddern *et al.*, “1 year, 1000 km : The Oxford RobotCar dataset,” *The International Journal of Robotics Research*, vol. 36, n°. 1, p. 3–15, janv. 2017.

[102] M. Pitropov *et al.*, “Canadian Adverse Driving Conditions dataset,” *The International Journal of Robotics Research*, vol. 40, n°. 4-5, p. 681–690, avr. 2021.

[103] J. Han *et al.*, “SODA10M : A large-scale 2D self/semi-supervised object detection dataset for autonomous driving,” nov. 2021. [En ligne]. Disponible : http://arxiv.org/abs/2106.11118

[104] R. Xu *et al.*, “V2X-ViT : Vehicle-to-everything cooperative perception with vision transformer,” dans *Computer Vision – ECCV 2022*. Cham : Springer Nature Switzerland, 2022, p. 107–124.

[105] A. Dosovitskiy *et al.*, “CARLA : An open urban driving simulator,” dans *Proceedings of the 1st Annual Conference on Robot Learning*. PMLR, oct. 2017, p. 1–16.

[106] SAE International, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles J3016_202104,” avr. 2021. [En ligne]. Disponible : https://www.sae.org/standards/content/j3016_202104/

[107] A. Mittal, A. Zisserman et P. H. Torr, “Hand detection using multiple proposals,” dans *British Machine Vision Conference*, vol. 2, n°. 3. Citeseer, 2011, p. 5.

[108] R. Buyya, T. Cortes et H. Jin, “An introduction to the InfiniBand architecture,” dans *High Performance Mass Storage and Parallel I/O : Technologies and Applications*. IEEE, 2002, p. 616–632.

[109] M. S. Birrittella *et al.*, “Intel Omni-Path Architecture : Enabling scalable, high performance fabrics,” dans *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, août 2015, p. 1–9.

[110] M. A. Jette et T. Wickberg, “Architecture of the Slurm workload manager,” dans *Job Scheduling Strategies for Parallel Processing*. Springer Nature Switzerland, 2023, p. 3–23.

[111] I. Foster, “Globus online : Accelerating and democratizing science through cloud-based services,” *IEEE Internet Computing*, vol. 15, n°. 3, mai 2011.

[112] T. Kluyver *et al.*, “Jupyter Notebooks – a publishing format for reproducible computational workflows,” dans *Positioning and Power in Academic Publishing : Players, Agents and Agendas*. IOS Press, 2016, p. 87–90.

[113] A. Naser *et al.*, “An empirical study of cryptographic libraries for MPI communications,” dans *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, sept. 2019, p. 1–11.

[114] S. M. Shah et V. K. N. Lau, “Model compression for communication efficient federated learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, n°. 9, p. 5937–5951, sept. 2023.

[115] A. Paszke *et al.*, “PyTorch : An imperative style, high-performance deep learning library,” dans *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, p. 8026–8037.

[116] E. Gabriel *et al.*, “Open MPI : Goals, concept, and design of a next generation MPI implementation,” dans *Proceedings, 11th European PVM/MPI Users’ Group Meeting*, Budapest, Hungary, sept. 2004, p. 97–104.

[117] L. Dalcin et Y.-L. L. Fang, “mpi4py : Status update after 12 years of development,” *Computing in Science & Engineering*, vol. 23, n°. 4, p. 47–54, juill. 2021.

[118] R. L. Rivest, A. Shamir et L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, n°. 2, p. 120–126, févr. 1978.

[119] M. Bellare et P. Rogaway, “Optimal asymmetric encryption,” dans *Advances in Cryptology — EUROCRYPT'94*. Springer, 1995, p. 92–111.

[120] J. Daemen et V. Rijmen, “AES proposal : Rijndael,” 1999. [En ligne]. Disponible : https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf

[121] D. McGrew et J. Viega, “The Galois/counter mode of operation (GCM),” *NIST Modes of Operation Process*, 2004. [En ligne]. Disponible : https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf

[122] C. Percival, “Stronger key derivation via sequential memory-hard functions,” janv. 2009. [En ligne]. Disponible : https://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf

ANNEXE A COURBES D'APPRENTISSAGE POUR LE CAS DE BASE

Nous fournissons les courbes d'apprentissage détaillées de YOLOv7 correspondant au cas de base, c'est-à-dire lorsque l'entraînement est centralisé, pour KITTI, nuImages-10 et nuImages-23, respectivement dans les Figures A.1, A.2 et A.3. Ces résultats permettent ensuite de mieux appréhender le cas fédéré, bien plus complexe. Les courbes incluent l'évolution du moment et des taux d'apprentissage (où *lr2* désigne le taux pour le groupe de biais, et *lr0* et *lr1* correspondent aux deux autres groupes de paramètres), les sous-fonctions de perte d'entraînement et de validation pour les trois sous-objectifs du problème de détection d'objets, ainsi que la précision, le rappel, la mAP_{50} et la mAP. Puisque le réglage fin de YOLOv7 aux jeux de données que nous avons choisis n'était pas un aspect important de cette recherche, nous avons, à l'exception de la durée d'échauffement, conservé les hyperparamètres par défaut. En définitive, il est donc très vraisemblablement possible d'améliorer les résultats ci-dessous.

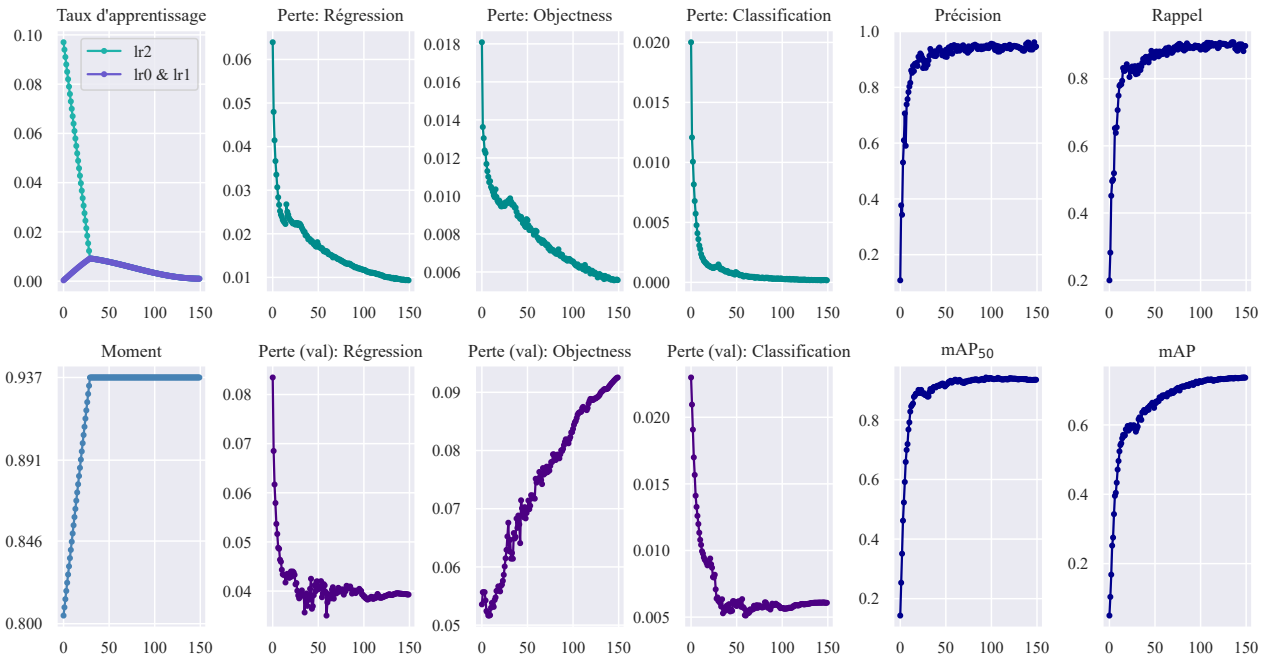

FIGURE A.1 Courbes d'apprentissage pour l'entraînement centralisé de YOLOv7 sur KITTI pour 150 époques.

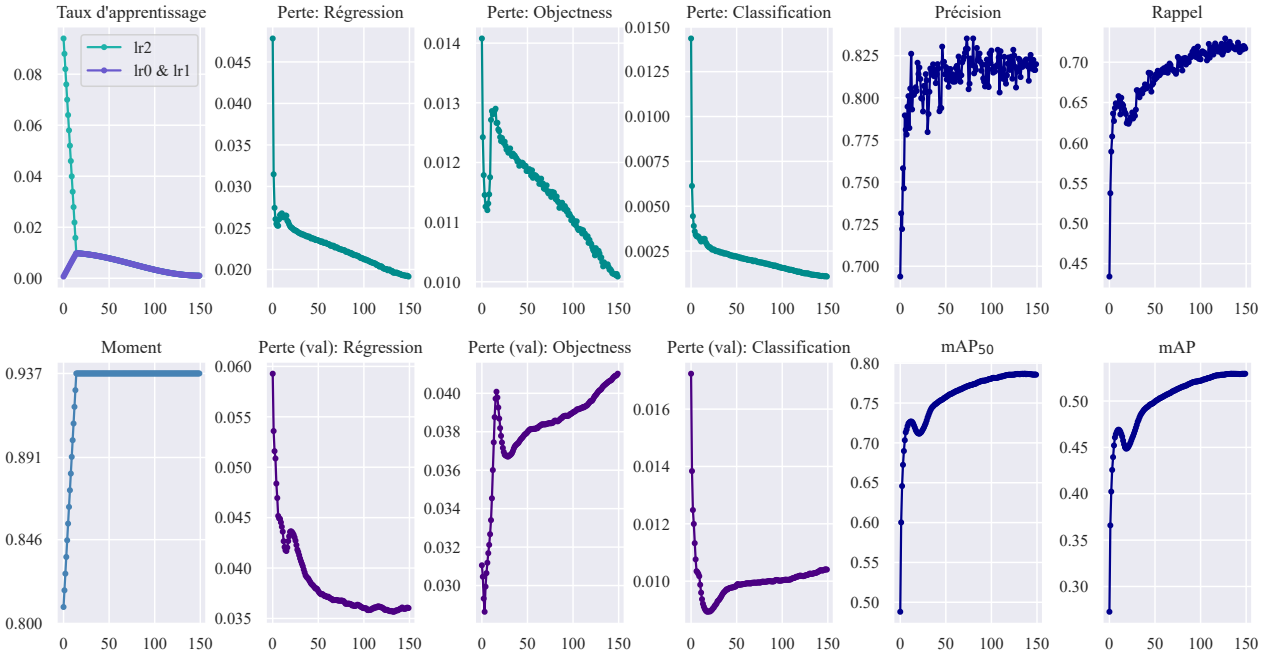

FIGURE A.2 Courbes d'apprentissage pour l'entraînement centralisé de YOLOv7 sur nuImages-10 pour 150 époques.

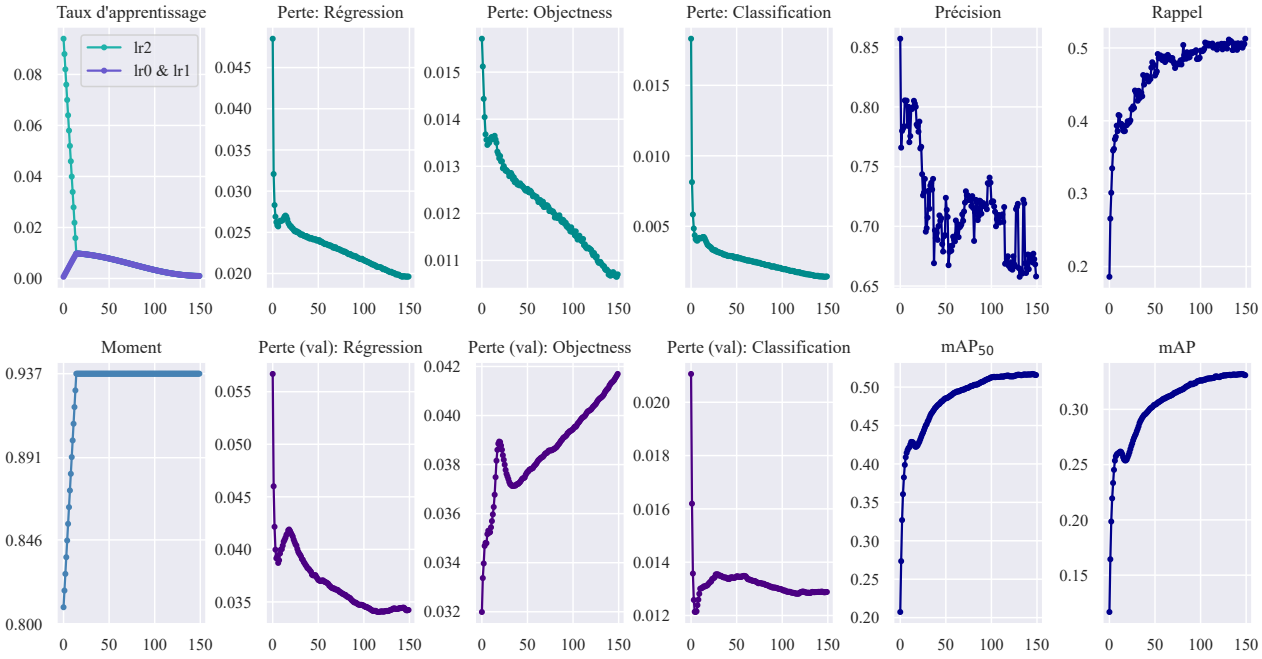

FIGURE A.3 Courbes d'apprentissage pour l'entraînement centralisé de YOLOv7 sur nuImages-23 pour 150 époques.

ANNEXE B EXEMPLES SUPPLÉMENTAIRES DE DÉTECTION

On illustre dans les Figures B.1 et B.2 les capacités prédictives de YOLOv7 entraîné avec FedAvgM, ainsi que détaillé dans la Section 4.7.3, pour quelques images sélectionnées aléatoirement dans les jeux de validation de respectivement KITTI et nuImages (10 classes).

FIGURE B.1 Prédictions pour un lot pris aléatoirement dans le jeu de validation de KITTI.

FIGURE B.2 Prédictions pour un lot pris aléatoirement dans le jeu de validation de nuImages.

ANNEXE C RÉSULTATS PRÉDICTIFS DÉTAILLÉS PAR CLASSE

Les métriques d'évaluation sont données détaillées par classe pour respectivement KITTI dans la Table C.1, nuImages-10 dans la Table C.2 et nuImages-23 dans la Table C.3. Dans tous les cas, nous avons utilisé les modèles employant la variante de base de YOLOv7, entraînée avec FedAvgM, ainsi que précédemment évoqué dans la Section 4.7.3.

Tableau C.1 Résultats détaillés par classe pour KITTI

Classe	#Annotations	Précision	Rappel	mAP_{50}	mAP
Car	7110	0.94	0.94	0.962	0.78
Pedestrian	1151	0.915	0.785	0.838	0.483
Van	780	0.94	0.924	0.951	0.766
Cyclist	428	0.939	0.848	0.899	0.623
Truck	307	0.985	0.971	0.986	0.821
Misc	233	0.93	0.901	0.935	0.704
Tram	148	0.95	0.919	0.925	0.694
Person_sitting	53	0.92	0.66	0.787	0.535
all	**10210**	**0.94**	**0.869**	**0.91**	**0.676**

Tableau C.2 Résultats détaillés par classe pour nuImages-10

Classe	#Annotations	Précision	Rappel	mAP_{50}	mAP
car	47279	0.852	0.826	0.881	0.637
pedestrian	32185	0.847	0.69	0.771	0.446
traffic_cone	18587	0.858	0.774	0.835	0.478
barrier	18433	0.82	0.694	0.764	0.488
truck	6858	0.766	0.639	0.705	0.513
bicycle	3352	0.777	0.722	0.763	0.53
motorcycle	3097	0.81	0.823	0.859	0.586
bus	1885	0.822	0.625	0.715	0.543
construction_vehicle	1303	0.76	0.609	0.647	0.344
trailer	486	0.7	0.216	0.309	0.198
all	**133465**	**0.801**	**0.662**	**0.725**	**0.476**

Tableau C.3 Résultats détaillés par classe pour nuImages-23

Classe	#Annotations	Précision	Rappel	mAP_{50}	mAP
car	47279	0.81	0.846	0.881	0.635
adult	28721	0.77	0.719	0.761	0.44
trafficcone	18587	0.807	0.804	0.832	0.474
barrier	18433	0.765	0.726	0.764	0.488
truck	6858	0.724	0.665	0.707	0.512
bicycle	3352	0.704	0.748	0.76	0.522
construction_worker	3117	0.743	0.568	0.61	0.321
motorcycle	3097	0.766	0.837	0.857	0.584
bus.rigid	1823	0.738	0.659	0.708	0.538
construction_vehicle	1303	0.729	0.614	0.635	0.34
debris	710	0.515	0.141	0.171	0.097
pushable_pullable	645	0.557	0.482	0.428	0.266
bicycle_rack	603	0.572	0.657	0.574	0.338
trailer	486	0.66	0.257	0.332	0.196
personal_mobility	453	0.477	0.629	0.501	0.225
child	251	0.486	0.104	0.105	0.055
police_officer	96	0.509	0.281	0.247	0.167
animal	82	0.497	0.183	0.152	0.067
stroller	70	0.729	0.269	0.358	0.243
bus.bendy	62	0.392	0.048	0.062	0.042
police	35	1	0	0	0
ambulance	8	1	0	0	0
wheelchair	2	1	0	0	0
all	**136073**	**0.693**	**0.445**	**0.454**	**0.285**