Federated Learning for Object Detection in Autonomous Vehicles

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINNEERING

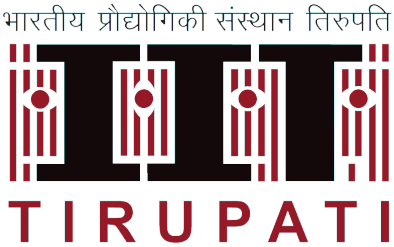
by

SANJAY VARSHITH CS22B029

SAI PREETHIKA CS22B006

Supervisor(s)

Dr. VISHNU CHALAVADI



DEPARTMENT OF COMPUTER SCIENCE AND ENGINNEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

NOVEMBER 2025

DECLARATION

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission to the best of my knowledge. we understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 29-11-2025

Sanjay
CS22B029

Preethika
CS22B006

BONA FIDE CERTIFICATE

This is to certify that the report titled **Federated Learning for Object Detection**, submitted by **Sanjay and Preethika**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the project work done by him [or her] under our or [my] supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Tirupati
Date: 29-11-2025

Dr.Vishnu Chalavadi
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517501

ACKNOWLEDGMENTS

We would like to express my sincere gratitude to my guide, Dr. VISHNU CHALAVADI, for his invaluable guidance, continuous support, and insightful feedback throughout this research. His expertise and encouragement were instrumental in the successful completion of this work.

ABSTRACT

This thesis presents a **federated learning framework** for privacy-preserving **object detection** using contemporary deep learning models. Traditional centralized vision pipelines require aggregating data in one location, increasing privacy risks and limiting real-world applicability when data is sensitive, regulated, or geographically distributed. **Federated Learning (FL)** mitigates these challenges by enabling decentralized model training without transferring raw data.

The study first evaluates the lightweight **YOLOv8n** model under FL conditions using the **KITTI dataset**. A baseline using **Federated Averaging (FedAvg)** is compared with advanced aggregation methods—**FedProx** and **FedScaffold**—to analyze their impact on **gradient drift, convergence speed,** and model robustness under non-IID settings.

To extend beyond convolutional architectures, the framework incorporates the **Detection Transformer (DETR)**, which replaces anchors and **Non-Maximum Suppression (NMS)** with **direct set prediction** using a **transformer encoder–decoder** and **multi-head self-attention**. Experiments reveal meaningful differences in communication cost, stability, and accuracy trends between YOLOv8 and DETR in federated environments. Model performance is evaluated using **mAP@50**, **precision**, and **recall**. Results indicate that federated learning is feasible for object detection and can achieve competitive accuracy with appropriate aggregation strategies.

Layout of the Thesis: Chapter 1 Introduces the problem statement, motivation, and objectives Chapter 2 reviews existing work on federated learning and object detection frameworks. Chapter 3 discusses the mathematical formulation, experimental setup, and methodology. Chapter 4 presents the experimental results, evaluation metrics, comparative analysis, and demonstrates privacy preservation within the federated learning setup. Chapter 5 concludes with key findings and outlines future research directions

Keywords: Federated Learning, Object Detection, YOLOv8n, DETR, Transformers, FedAvg, FedProx, FedScaffold.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

FL	Federated Learning
FedAvg	Federated Averaging
Non-IID	Non-Independently and Identically Distributed
YOLOv8	You Only Look Once, version 8
PANet	Path Aggregation Network
DETR	Detection Transformer
CNN	Convolutional Neural Network
SAM	Segment Anything Model
TP	True Positive
FP	False Positive
FN	False Negative
P	Precision
R	Recall
F1	F1 Score
AP	Average Precision
mAP	mean Average Precision
IoU	Intersection over Union
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute Dataset
GT	Ground Truth
MHSA	Multi-Head Self-Attention
FFN	Feed-Forward Network
CSPDarknet	Cross Stage Partial Darknet (YOLO backbone)
C2f	Cross-Stage Partial with 2-way fusion
CE	Cross-Entropy
NMS	Non-Maximum Suppression
GPU	Graphics Processing Unit
VRAM	Video Random Access Memory

CHAPTER 1

INTRODUCTION

1.1 Objectives and Scope

The driving force behind this research is the critical need for machine learning models that can operate effectively without compromising privacy, particularly in sensitive fields like autonomous driving, smart surveillance, and intelligent transportation. Traditionally, training these models meant hoarding massive amounts of image data on a single central server. However, this approach creates significant friction regarding data ownership, legal compliance, and privacy risks—especially when the data is collected from real-world user environments. Federated Learning (FL) offers a way around these bottlenecks by allowing models to learn collaboratively while keeping the raw data locked on the local devices where it originated.

Beyond the privacy aspect, this work is motivated by the need to understand how different object detection architectures behave in a decentralized environment. While convolution-based detectors like the YOLO family are the industry standard for real-time speed, their performance can degrade when data is unevenly distributed across different clients (non-IID data). To test the robustness of modern architectures, we also integrate the Detection Transformer (DETR). Unlike traditional detectors that rely on manual post-processing steps, DETR utilizes attention mechanisms and global reasoning. We hypothesize that these transformer-based traits might offer better feature alignment when training is fragmented across a network.

Practically, this project focuses on implementing a federated object detection pipeline using two simulated clients, each holding a partitioned subset of the KITTI dataset. Rather than attempting to classify every available object, the implementation specifically targets the *Cyclist* and *Truck* categories to allow for a focused analysis of training behavior. To ensure the results are robust, we benchmark the standard FedAvg aggregation algorithm against more advanced methods like FedProx and FedScaffold, which are designed to handle the statistical heterogeneity often found in real-world networks.

Specifically, this research aims to:

- Design and deploy a functioning federated learning pipeline capable of handling decentralized object detection tasks.
- Conduct a comparative analysis of how YOLOv8n and DETR architectures perform when subjected to federated constraints.
- Evaluate the stability and convergence speed of different aggregation strategies (FedAvg, FedProx, and FedScaffold) to determine which handles non-IID data most effectively.
- Assess whether federated object detection is a viable solution for privacy-critical applications in autonomous vehicle networks and roadside perception systems.

CHAPTER 2

Literature Review

Federated Learning (FL) has emerged as the dominant paradigm for privacy-preserving model training, particularly in scenarios where data centralization is prohibited by privacy laws or bandwidth constraints. This chapter synthesizes existing literature across three pillars: the mechanics of decentralized optimization, modern object detection architectures (specifically YOLOv8 and DETR), and the intersection of these fields within autonomous driving contexts.

2.1 Federated Learning Fundamentals

Introduced by Google in 2017, Federated Learning fundamentally shifts the training workload from a central server to the edge [4]. In this architecture, raw data never leaves the client device. Instead, clients compute weight updates (gradients) locally, transmitting only these updates to a central aggregator. This structure effectively circumvents the privacy risks associated with pooling sensitive user data into a single repository.

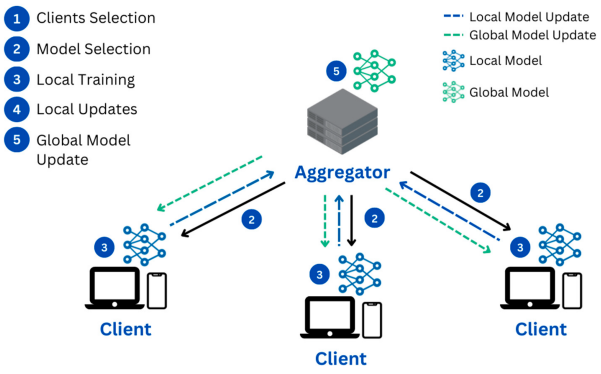


Figure 2.1: Baseline Architecture of Federated Learning

However, moving training to the edge introduces the challenge of statistical heterogeneity. Unlike centralized datasets which can be shuffled to be Identically and Independently Distributed (IID), real-world federated data is often highly non-IID. Zhao et al. [8] demonstrated that model accuracy degrades significantly when the data distribution among clients is skewed—a common occurrence in automotive networks where a vehicle in a city center encounters different object classes than one in a rural area.

2.1.1 Federated Optimization Algorithms

To mitigate the divergence caused by heterogeneous data and varying client resources, several aggregation strategies have been developed.

Federated Averaging (FedAvg)

FedAvg is the canonical algorithm for FL. It reduces communication overhead by allowing clients to perform multiple epochs of local Stochastic Gradient Descent (SGD) before synchronizing with the server. The server aggregates these local models via a weighted average:

$$w_{global} = \sum_{k=1}^{K} \frac{n_k}{n_{total}} w_k \tag{1}$$

where n_k is the data volume on client k and n_{total} is the aggregate volume across all participating clients. While computationally efficient, FedAvg suffers from "client drift" in non-IID settings, where local models move towards their local optima rather than the global optimum.

FedProx

Li et al. [7] proposed FedProx to handle statistical and systems heterogeneity. It introduces a proximal term to the local objective function:

$$\min_{w} F_k(w) + \frac{\mu}{2} \|w - w_{global}\|^2 \tag{2}$$

This term limits the impact of variable local updates by penalizing models that deviate excessively from the global weights, ensuring convergence even when clients perform variable amounts of local work due to hardware constraints.

SCAFFOLD

SCAFFOLD tackles client drift by estimating the update direction of the server and the client using control variates [6]. The update rule effectively corrects the local gradient, denoted as:

$$w_k^{t+1} = w_k^t - \eta (\nabla f_k(w_k^t) - c_k + c) \tag{3}$$

By maintaining state parameters (c_k and c), SCAFFOLD aligns local updates with the global optimization trajectory, proving more robust than FedAvg in highly skewed data environments.

2.2 Modern Object Detection Architectures

The evolution of object detection has moved from computationally expensive region-based methods to efficient single-stage detectors and, more recently, transformer-based approaches.

2.2.1 YOLOv8

YOLOv8 [5] is the latest iteration in the "You Only Look Once" family, representing the state-of-the-art in CNN-based detection. It introduces an anchor-free detection mechanism and a decoupled head, which processes objectness, classification, and regression tasks independently. The architecture utilizes a CSP (Cross Stage Partial) Darknet backbone, optimized for gradient flow and reduced computational complexity. For federated environments, YOLOv8 is particularly attractive due to its favorable trade-off between inference speed and parameter count, making it deployable on resource-constrained edge devices found in vehicles.

2.2.2 Detection Transformer (DETR)

DETR [1] offers a radical departure from CNN traditions by treating object detection as a set prediction problem. It eliminates heuristic components like anchor generation and Non-Maximum Suppression (NMS). The architecture employs a Transformer encoder-decoder that reasons about the relations between objects and the global image context.

A key innovation is the use of bipartite matching loss, which enforces a one-to-one prediction structure. While DETR traditionally requires longer training times, its global attention mechanism makes it theoretically more robust to occlusions and complex scenes often found in driving datasets like KITTI [3].

2.3 Federated Object Detection in Autonomous Driving

The application of FL to object detection is driven by the need to utilize massive distributed datasets without compromising privacy. Devarajan et al. (2025) [2] highlight the utility of YOLO-based FL frameworks for creating transparent and secure driving models.

However, training these models federatedly presents unique difficulties:

- **Label Distribution Skew:** In datasets like KITTI [3], classes such as "Pedestrian" or "Cyclist" may appear frequently in some clients (urban drivers) and rarely in others (highway drivers), destabilizing the aggregation process.
- **Bandwidth Overhead:** Transmitting the weights of large transformer models like DETR puts strain on V2X (Vehicle-to-Everything) communication channels.

Current research suggests a divergence in suitability: while YOLOv8 offers superior communication efficiency, the global context awareness of DETR may offer resilience against the localized biases inherent in non-IID client data.

2.4 Summary

The literature indicates a maturity in both Federated Learning algorithms and object detection architectures independently. However, the integration of advanced detectors like YOLOv8 and DETR into non-IID federated environments remains under-explored. The specific challenges of adapting these models to the heterogeneous nature of the KITTI dataset form the basis for the methodology proposed in the subsequent chapter.

CHAPTER 3

Methodology and Experimental Setup

3.1 Overview

This chapter delineates the procedural framework employed to implement federated learning for object detection, specifically contrasting the YOLOv8 and DETR architectures. The discussion encompasses the mathematical formulation of the aggregation strategy, the specifications of the experimental environment, dataset curation, and the workflow established to train and benchmark these models within a decentralized, privacy-centric paradigm.

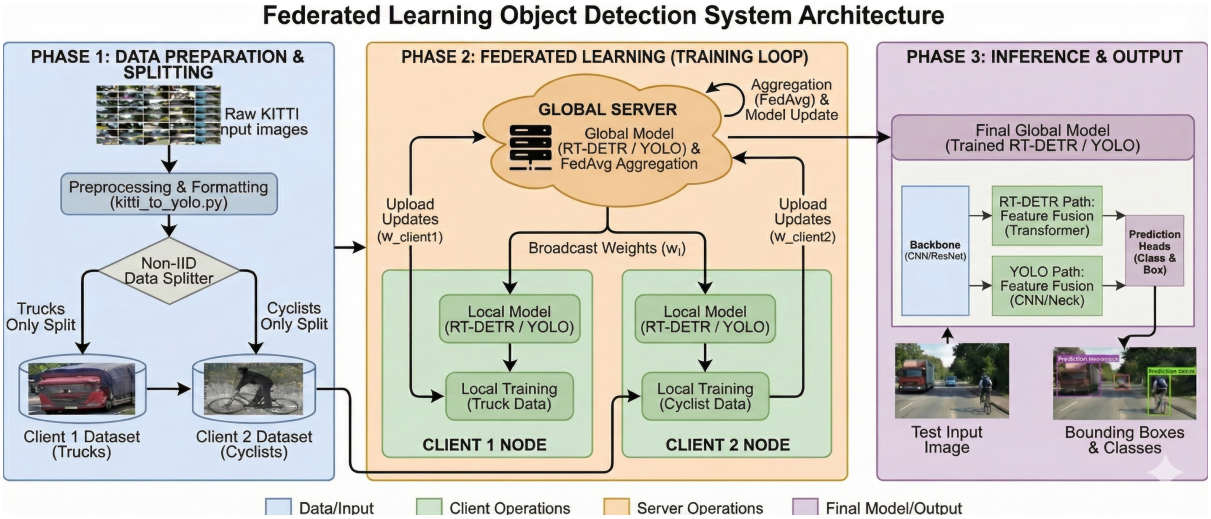


Figure 3.1: Overall Model Architecture

3.2 Model Architectures

3.2.1 YOLOv8 Architecture

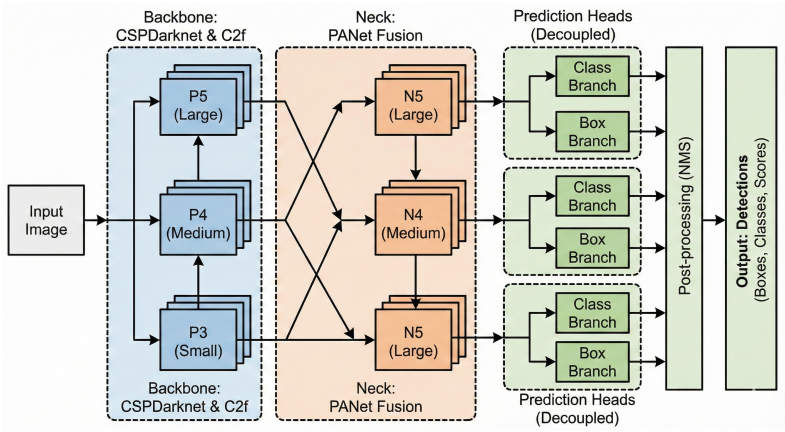


Figure 3.2: YOLOv8 Model Architecture

The YOLOv8 framework operates as a cohesive, real-time object detection pipeline partitioned into three primary components: the backbone, neck, and prediction heads. This architecture was chosen due to its optimal trade-off between inference latency, spatial feature extraction capabilities, and the separation of prediction logic.

1. Backbone: CSPDarknet and C2f Blocks

Feature extraction is managed by the backbone, which captures hierarchical patterns from the input image. It relies on a CSPDarknet53 structure augmented with C2f modules. These blocks are engineered to enhance gradient propagation and reduce computational load by dividing feature maps into distinct paths that are subsequently concatenated:

$$F_{\text{out}} = \text{Concat}\big(F_1,\ g(F_2)\big) \tag{4}$$

where $g(\cdot)$ denotes the convolutional operations.

2. Neck: PANet Feature Aggregation

The neck utilizes a Path Aggregation Network (PANet) to merge features across various scales. This configuration bolsters bottom-up information flow, ensuring that the network maintains

detection robustness for objects of differing dimensions. By consolidating features from levels P3, P4, and P5, the mechanism computes:

$$F_{\text{PANet}} = \phi(F_{P3}, F_{P4}, F_{P5}) \tag{5}$$

where ϕ represents the sequence of upsampling, downsampling, and lateral integration steps.

3. Decoupled Detection Heads

A distinct feature of YOLOv8 is the shift away from coupled heads. Instead, classification tasks and bounding box regression are optimized via separate branches:

$$\text{Head}_{cls}(F) \neq \text{Head}_{box}(F) \tag{6}$$

Bounding box coordinates are generated through an anchor-free regression method:

$$(t_x, t_y, t_w, t_h)$$

where (t_x, t_y) represent center offsets and (t_w, t_h) indicate dimensions. Empirical observations suggested that this decoupled approach accelerated convergence and improved mAP scores during our trials, especially for smaller entities such as cyclists.

3.2.2 DETR Architecture

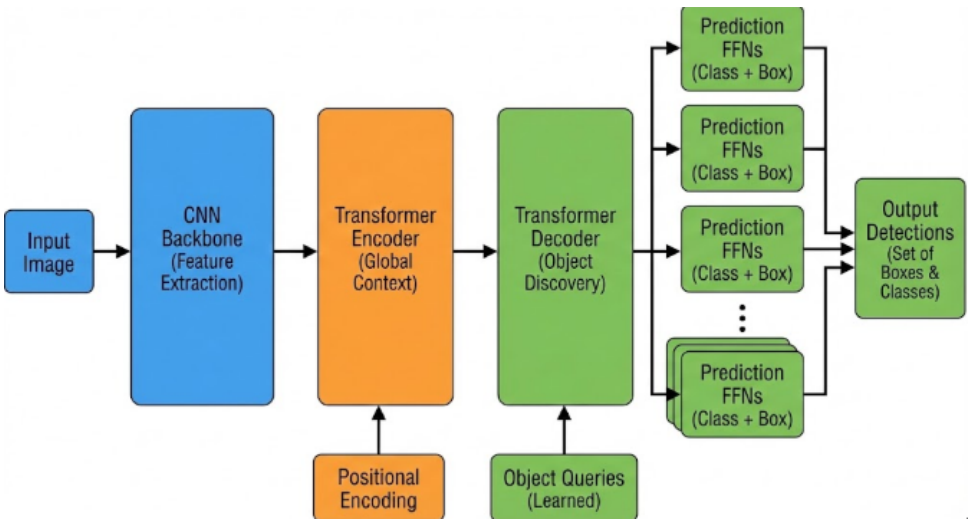


Figure 3.3: DETR Transformer-Based Object Detection Architecture

The Detection Transformer (DETR) represents a paradigm shift, reformulating object detection as a direct set prediction task. This methodology removes reliance on heuristic post-processing steps such as anchor generation or Non-Maximum Suppression (NMS). The system comprises a CNN backbone, a transformer encoder, a transformer decoder, and set prediction heads.

1. CNN Backbone (Feature Extraction)

A ResNet backbone processes the raw input to generate a lower-resolution feature map:

$$F = \text{CNN}(I) \tag{7}$$

These features serve as the initial input sequence for the transformer layers.

2. Positional Encoding

Given that transformers inherently lack an understanding of sequence order, fixed 2D sine-cosine positional encodings are added to the feature maps to retain spatial context:

$$Z_0 = F + PE \tag{8}$$

3. Transformer Encoder (Global Context Modeling)

The encoder subjects the input tokens to multi-head self-attention mechanisms:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{9}$$

This operation enables the network to analyze long-range dependencies and establish spatial reasoning across the entire image field.

4. Transformer Decoder (Object Query Refinement)

The decoder utilizes a predefined set of learnable embeddings, termed object queries. Each query is tasked with identifying a unique object instance:

$$Q_{t+1} = \text{Decoder}(Q_t, \text{ EncoderOutput}) \tag{10}$$

These queries interface with the global image features to isolate specific object characteristics.

5. Set Prediction Using Hungarian Matching

DETR generates a fixed number of predictions:

$$\hat{y} = \{(\hat{b}_i, \hat{c}_i)\}_{i=1}^{N} \tag{11}$$

To map these predictions to ground truth annotations, the Hungarian algorithm (bipartite matching) is employed:

$$\pi^* = \arg\min_{\pi} \sum_{i} \mathscr{L}_{match}(y_i, \hat{y}_{\pi(i)}) \tag{12}$$

This strategy effectively manages duplicate detections without requiring NMS, a trait that proves advantageous for maintaining consistency across federated model updates.

3.3 Mathematical Formulation

The primary objective involves assessing the impact of federated learning on model convergence and generalization capabilities when direct access to raw data is restricted.

Consider a network of K clients, where every client k possesses a local dataset $\mathscr{D}_k$ containing M distinct object classes. If $n_{k,c}$ represents the sample count for class c on client k, the local dataset volume is defined as:

$$n_k = \sum_{c=1}^{M} n_{k,c} \tag{13}$$

Consequently, the aggregate sample count across the entire network is:

$$n = \sum_{k=1}^{K} n_k = \sum_{k=1}^{K} \sum_{c=1}^{M} n_{k,c} \tag{14}$$

The global optimization goal is to minimize the weighted average of the local loss functions:

$$\min_{w} F(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{15}$$

Here, $F_k(w)$ represents the loss computed locally:

$$F_k(w) = \frac{1}{n_k} \sum_{c=1}^{M} \sum_{j=1}^{n_{k,c}} \ell(w; x_{k,c,j}, y_{k,c,j}) \tag{16}$$

Upon the completion of a communication round, the global model parameters are updated using a weighted average of the client contributions. This aligns with the standard Federated Averaging (FedAvg) protocol:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1} \tag{17}$$

3.4 Experimental Setup

The simulation environment was constructed using two distinct client nodes to train the object detection models. Each node was allocated a specific, non-overlapping partition of the KITTI object detection dataset. This configuration was intended to replicate a Non-IID distribution, where object categories differed substantially between clients (specifically isolating Cyclists versus Trucks).

The software stack for orchestration and training included:

- Ultralytics YOLOv8 for CNN-based detection.
- Facebook DETR implementation for transformer-based detection.

Hardware Specifications

- **GPU:** NVIDIA RTX 3050 (8GB VRAM)
- **RAM:** 32GB
- **OS:** Linux environment

3.5 Materials

Figure 3.4: KITTI Dataset

- **Dataset:** KITTI Vision Benchmark Suite (9 classes, reduced to 2 for the scope of this study).
- **Models:** YOLOv8n and DETR (utilizing a ResNet-50 backbone).
- **Frameworks:** PyTorch, Ultralytics, HuggingFace, Flower.

3.6 Procedure and Implementation

The experimental workflow was divided into four distinct phases: data preprocessing, Non-IID partitioning, federated simulation, and performance evaluation. The Flower framework facilitated the orchestration, while PyTorch handled underlying model operations.

3.6.1 Data Preprocessing

Initial steps involved converting the raw KITTI 2D bounding box annotations into the normalized YOLO format $(x_{center}, y_{center}, width, height)$. To guarantee a valid comparison between the YOLOv8 and DETR architectures, input images were resized to uniform dimensions of 640×640

pixels. Additionally, a centralized baseline model was trained on the complete dataset to serve as a performance benchmark.

3.6.2 Non-IID Data Partitioning

To enforce a strict Non-IID setting, the dataset was split based on class exclusivity using a custom filtering logic:

1. **Client 1 (Truck Specialist):** Allocated exclusively images containing the 'Truck' class.
2. **Client 2 (Cyclist Specialist):** Allocated exclusively images containing the 'Cyclist' class.
3. **Exclusion Criteria:** To prevent data leakage and ensure that local models had zero exposure to the opposing class during training, any images containing both classes were removed from the training set.

3.6.3 Federated Simulation Loop

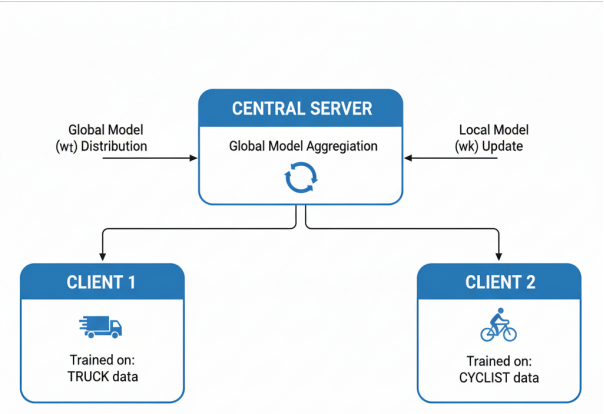


Figure 3.5: Federated Learning Process for selected classes across respective clients

The FedAvg strategy was employed with a fit fraction of 1.0, mandating full client participation in every round. The procedure followed these steps:

1. **Initialization:** A seed model was generated with a specific 2-class head (Truck/Cyclist). For the DETR architecture, a pre-initialization step was required to align the Transformer class embeddings, ensuring both clients commenced training with identical weights.
2. **Local Training (Client-Side):** In round t, the central server distributed the global weights w_t to Client k. Subsequently, the client updated its model w_k by training on its local data

partition $\mathscr{D}_k$. Given the substantial VRAM requirements of DETR's attention maps, the batch size was capped at $B = 1$ (in contrast to $B = 4$ for YOLO).

$$w_k^{t+1} \leftarrow \text{Train}(w_t, \mathscr{D}_k, \text{epochs} = 1) \tag{18}$$

3. **Aggregation (Server-Side):** The server aggregated the updated parameters from both clients to compute the new global model state:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1} \tag{19}$$

3.6.4 Evaluation Strategy

The efficacy of the models was assessed using Mean Average Precision at an IoU of 0.50 (mAP@50) via two distinct testing protocols:

- **Global Generalization:** The aggregated global model was evaluated on a held-out validation set containing both classes to confirm its ability to synthesize knowledge from separate clients.
- **Specialization Analysis:** Local client models were tested against the dataset of the opposing class (e.g., Client 1 tested on Cyclist data) to quantify the extent of catastrophic forgetting and the specific influence of the Non-IID distribution.

CHAPTER 4

Results and Discussions

This chapter details the experimental outcomes of implementing Federated Learning (FL) for object detection. We compared two architectures, YOLOv8 and RT-DETR, using the KITTI dataset in a non-IID setting. The main objective was to see how well the Global model aggregates knowledge compared to local clients that only see specific classes (Trucks or Cyclists).

4.1 Evaluation Metrics

We used standard metrics to quantify performance. Defining TP as True Positives, FP as False Positives, and FN as False Negatives:

- **Precision (P)**: The accuracy of positive predictions.

$$P = \frac{TP}{TP + FP} \tag{20}$$

- **Recall (R)**: The ability to find all positive instances.

$$R = \frac{TP}{TP + FN} \tag{21}$$

- **F1 Score**: The harmonic mean of Precision and Recall.

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{22}$$

- **mean Average Precision (mAP)**: The average precision across classes at an IoU threshold of 0.50.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{23}$$

4.2 YOLOv8 Federated Evaluation

We trained YOLOv8 using the FedAvg strategy. The network consisted of two clients with non-IID data distributions: Client 1 held Truck data, while Client 2 held Cyclist data.

4.2.1 YOLOv8 Quantitative Analysis

Table 4.1 shows the breakdown of performance metrics. We compared the Global Model against the local clients on both datasets.

Table 4.1: YOLOv8: Unified Performance (Global vs. Clients)

Model	Test Dataset	Precision	Recall	mAP@50	F1 Score
Global Model	Truck Data	0.9767	0.8070	0.9224	0.8838
Client 1 (Trucks)	Truck Data	0.9318	0.8221	0.9124	0.8735
Client 2 (Cyclists)	Truck Data	0.8780	0.3462	0.6040	0.4966
Global Model	Cyclist Data	0.8720	0.5743	0.7225	0.6925
Client 1 (Trucks)	Cyclist Data	0.8974	0.1155	0.5031	0.2047
Client 2 (Cyclists)	Cyclist Data	0.7727	0.6284	0.6968	0.6931

Analysis of Results:

- Global Model Robustness: Interestingly, the Global YOLO model performed exceptionally well. On Truck data, it actually achieved a slightly higher mAP (0.9224) than the local specialist Client 1 (0.9124). This suggests that the aggregation process helped smooth out some noise from the local training.
- Local Specialization: Client 1 struggled significantly with Cyclists (Recall: 0.1155) and Client 2 struggled with Trucks (Recall: 0.3462). This confirms that the clients were indeed specializing in their local data partitions, though the pre-trained weights likely prevented the score from dropping to absolute zero.

4.2.2 YOLOv8 Qualitative Analysis (Visual Detection)

The figures below demonstrate the impact of aggregation visually. We present the detections from the local clients (before aggregation) followed by the Global Model (after aggregation).

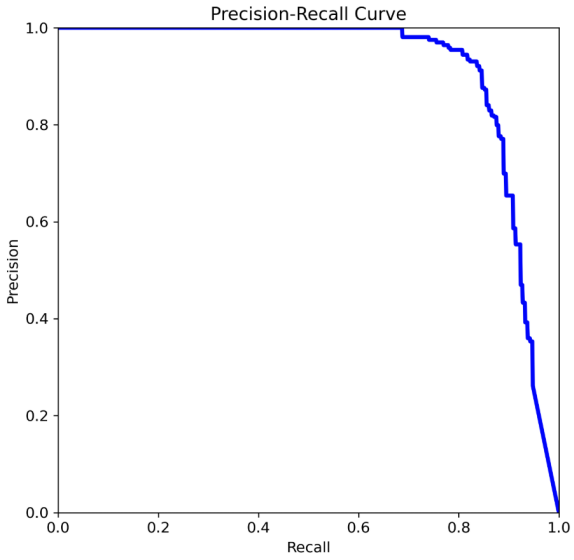
Figure 4.1: YOLO Client 1 (Truck Specialist): The model correctly identifies the Truck with high confidence but completely misses the Cyclist.

Figure 4.2: YOLO Client 2 (Cyclist Specialist): The model detects the Cyclist but ignores the Truck, treating it as background.

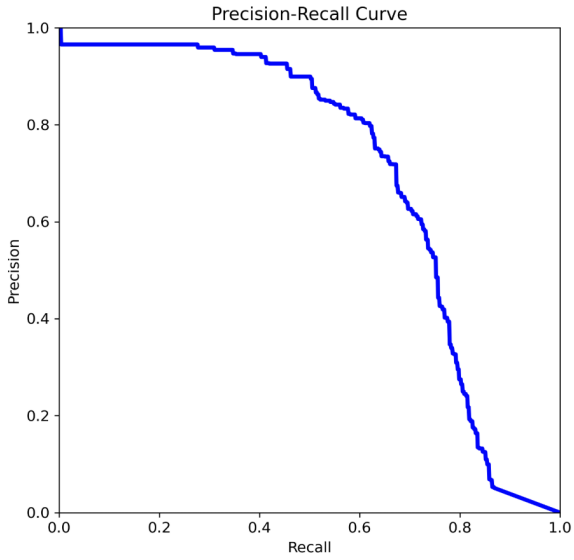
Figure 4.3: YOLO Global Model: After Federated Averaging, the model successfully detects BOTH the Truck and the Cyclist in the same scene.

4.2.3 YOLOv8 Training Curves

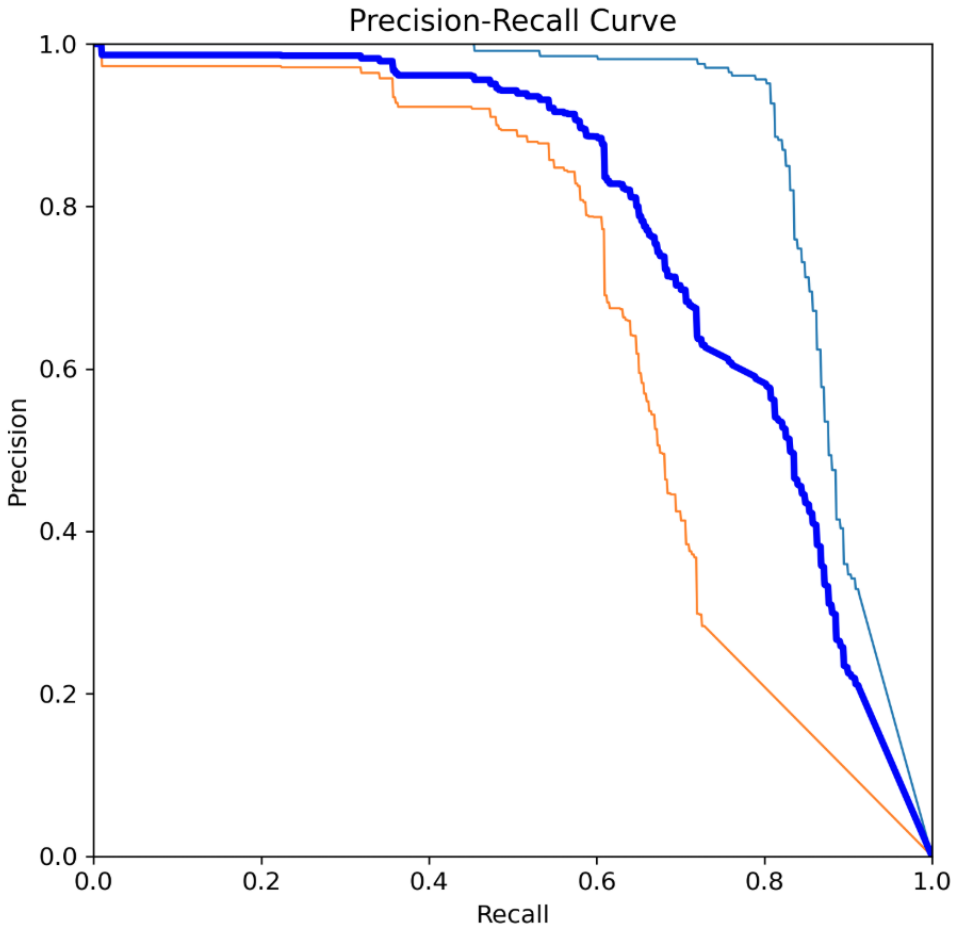
The graphs below visualize the training process.



(a) Client 1 (Truck Specialist)



(b) Client 2 (Cyclist Specialist)



(c) Global FL Model (All Classes)

Figure 4.4: Precision-Recall Curves for YOLO in Federated Learning.

Figure 4.5: YOLO Training Performance: Client 1 (Left) learns Trucks rapidly, while Client 2 (Right) learns Cyclists. Note the flatline at 0.0 for the unseen class.

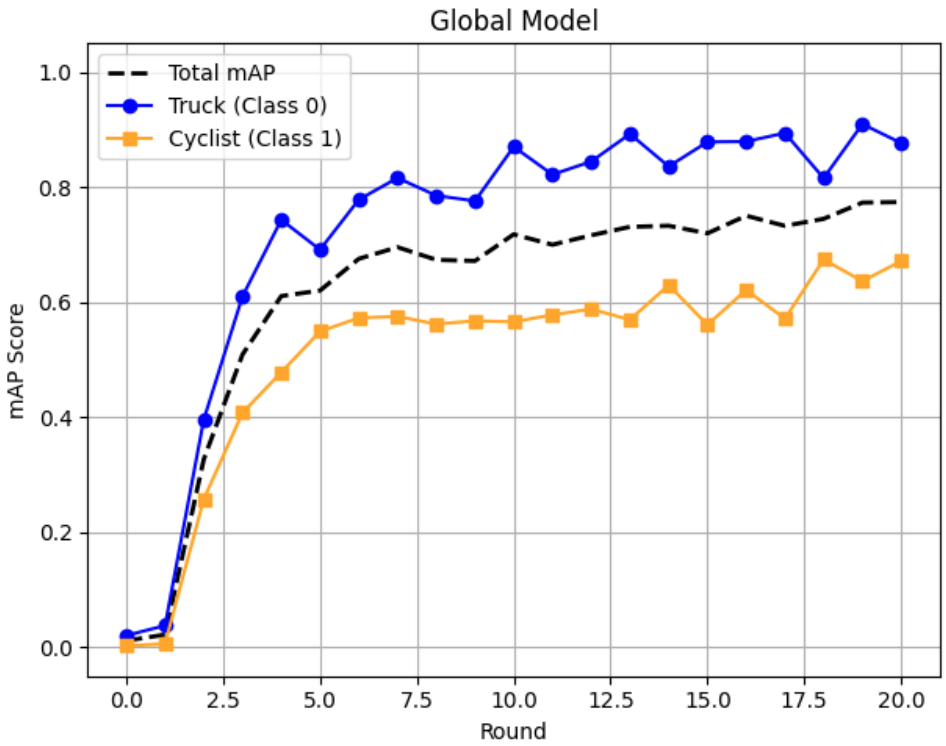


Figure 4.6: YOLO Global Model Convergence: The server successfully aggregates weights to achieve high mAP on both classes simultaneously.

4.3 RT-DETR Federated Evaluation

We repeated the experiment using the RT-DETR architecture to see if a Transformer-based model handles weight aggregation differently than the CNN-based YOLO.

4.3.1 RT-DETR Quantitative Analysis

Table 4.2 summarizes the performance.

Table 4.2: RT-DETR: Unified Performance (Global vs. Clients)

Model	Test Dataset	Precision	Recall	mAP@50	F1 Score
Global Model	Truck Data	0.9543	0.9183	0.9332	0.9360
Client 1 (Trucks)	Truck Data	0.8233	0.8894	0.8285	0.8551
Client 2 (Cyclists)	Truck Data	0.9000	0.0865	0.4911	0.1579
Global Model	Cyclist Data	0.8708	0.7591	0.8189	0.8111
Client 1 (Trucks)	Cyclist Data	0.8206	0.4983	0.6180	0.6201
Client 2 (Cyclists)	Cyclist Data	0.7997	0.7905	0.8014	0.7951

Analysis of Results:

- Global Model Dominance: The RT-DETR Global model outperformed the local clients in almost every metric. For Trucks, the Global model reached 0.9332 mAP compared to Client 1's 0.8285. For Cyclists, it reached 0.8189 mAP compared to Client 2's 0.8014. This indicates that RT-DETR benefits heavily from the aggregated weights, creating a model that is better than the sum of its parts.
- Recall Issues in Non-Experts: Look at Client 2 (Cyclist expert) testing on Truck data: the Recall is extremely low (0.0865). It effectively misses almost every truck. However, Client 1 (Truck expert) maintained a moderate Recall (0.4983) on Cyclists. This implies that while the model specializes, the Transformer attention mechanism might retain some general features of "smaller objects" like cyclists better than "large objects" like trucks when not explicitly trained on them.

4.3.2 RT-DETR Qualitative Analysis (Visual Detection)

We now examine the visual output of the RT-DETR models. The separation between local expertise and global generalization is clearly visible.

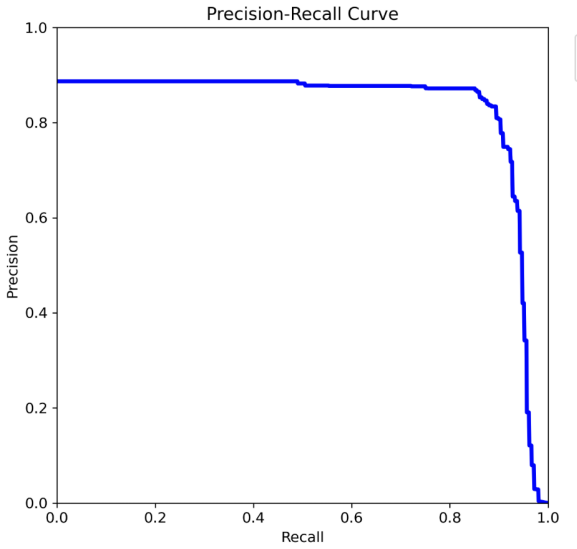
Figure 4.7: RT-DETR Client 1: The Truck specialist successfully bounds the large vehicle but fails to detect the cyclist on the right.

Figure 4.8: RT-DETR Client 2: The Cyclist specialist ignores the truck entirely but places a high-confidence bounding box on the cyclist.

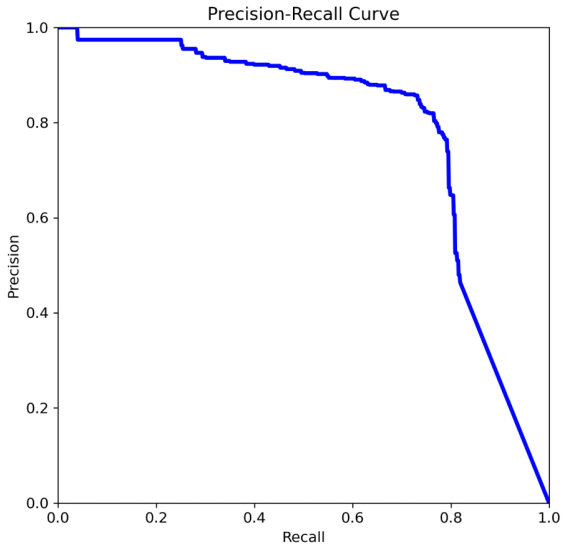
Figure 4.9: RT-DETR Global Model: The aggregated model demonstrates superior recall, detecting both the distant Truck and the Cyclist that the individual clients missed or detected with lower confidence.

4.3.3 RT-DETR Training Curves

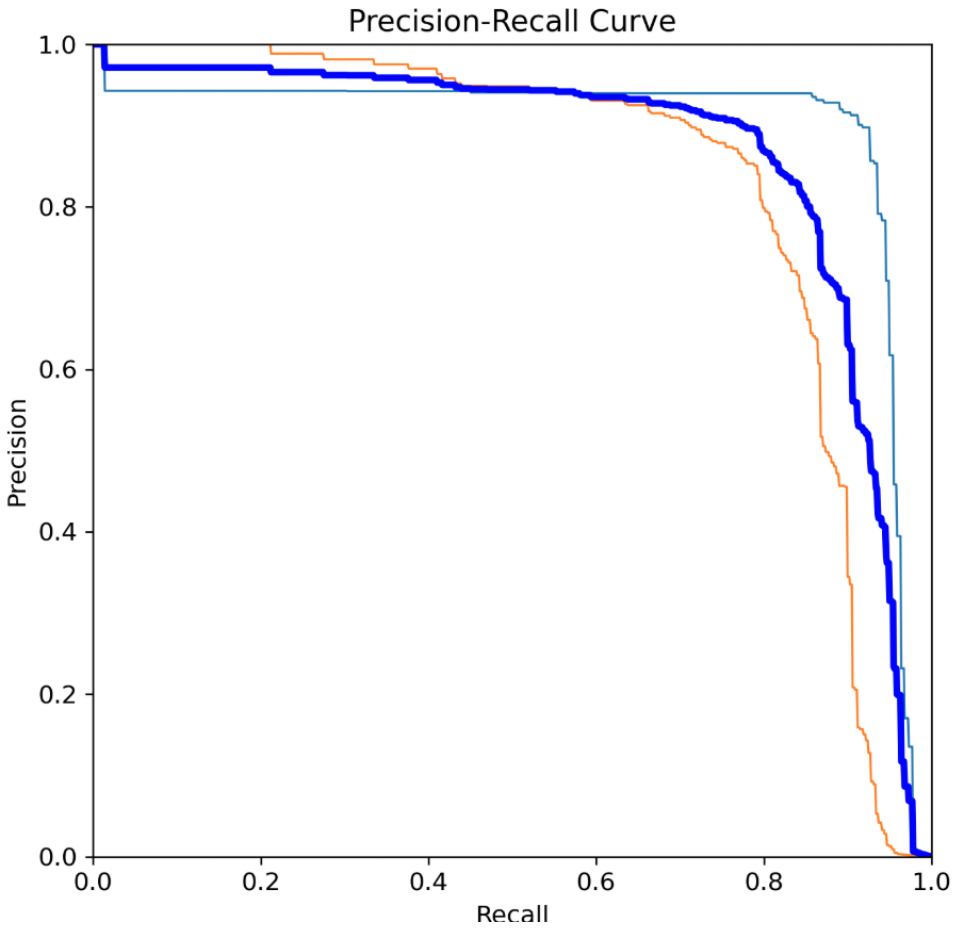
The visual performance of the DETR models is presented below in Figure 4.10 and Figure 4.11.



(a) Client 1 (Truck Specialist)



(b) Client 2 (Cyclist Specialist)



(c) Global FL Model (All Classes)

Figure 4.10: Precision-Recall Curves for RT-DETR in Federated Learning.

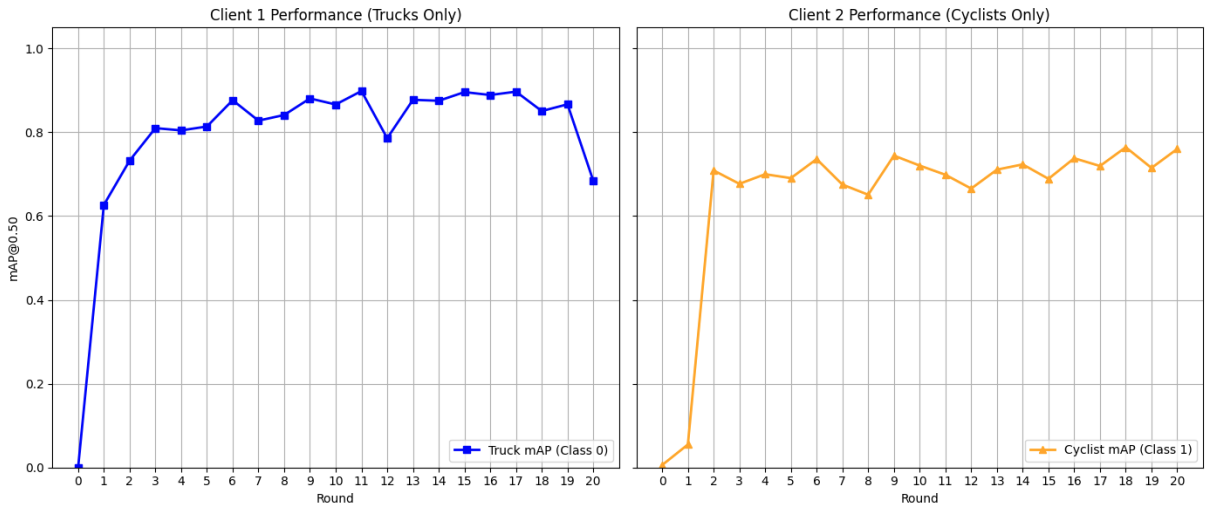


Figure 4.11: RT-DETR Training Performance: Client 1 (Left) learns Trucks rapidly, while Client 2 (Right) learns Cyclists. Note the flatline at 0.0 for the unseen class.

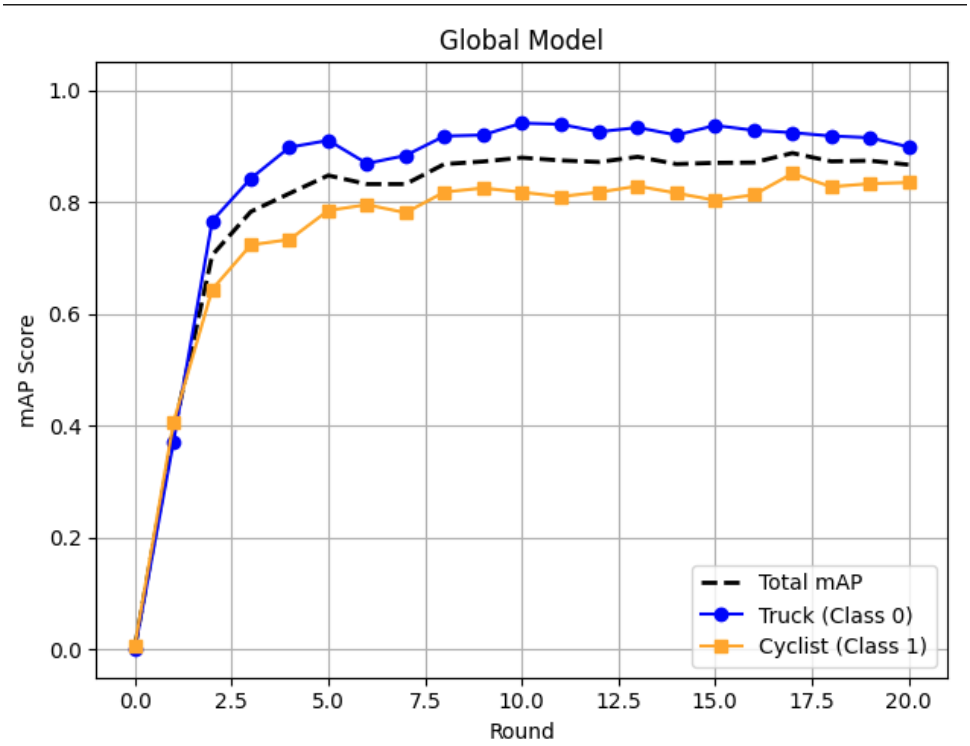


Figure 4.12: RT-DETR Global Model Convergence: The server successfully aggregates weights to achieve high mAP on both classes simultaneously.

4.4 Discussion and Comparisons

4.4.1 Architecture Comparison: YOLO vs. RT-DETR

Comparing the two architectures reveals distinct advantages for the Transformer-based approach in a Federated setting.

1. Global Model Performance: RT-DETR's global model showed superior generalization. On the Cyclist dataset, RT-DETR achieved a Global mAP of 0.8189, significantly higher than YOLO's 0.7225. This suggests that RT-DETR is more effective at learning from the aggregated weights of non-IID clients.

2. Handling the "Unseen" Class: Both models showed that local clients perform poorly on data they haven't seen locally, which is expected. However, the drop was more severe in YOLO. For example, Client 2 (Cyclist expert) testing on Trucks had a Recall of 0.34 in YOLO and only 0.08 in DETR. This indicates that DETR clients might overfit more strictly to their local distribution, whereas YOLO retains a bit more "fuzzy" knowledge from pre-training.

4.4.2 Effectiveness of Aggregation

A key takeaway from these results is that the Global Model is not just an average—it is a distinct improvement. In the RT-DETR experiments, the Global Model beat the local specialist (Client 1) on its own Truck data (0.9332 vs 0.8285 mAP). This is a strong argument for using Federated Learning: even if a client has good local data, participating in the federation and receiving weights from others (even those with different data) can improve the feature extraction capabilities of the model.

CHAPTER 5

Summary and Conclusion

This thesis established a federated learning framework for privacy-preserving object detection using YOLOv8 and RT-DETR. By training on highly heterogeneous (non-IID) subsets of the KITTI dataset, we demonstrated that decentralized clients can collaboratively train robust detection models without sharing raw data, addressing critical privacy and bandwidth challenges in autonomous systems.

5.1 Key Conclusions

The experimental analysis yields the following core insights:

1. **Feasibility:** Both YOLOv8 and RT-DETR successfully converged using the FedAvg algorithm, proving that complex object detection tasks are viable in a federated setting.
2. **Architecture Robustness:** RT-DETR (Transformer-based) demonstrated superior resilience to non-IID data compared to YOLOv8 (CNN-based). The global RT-DETR model maintained high precision and recall, whereas YOLOv8 struggled with recall due to the destructive effects of averaging divergent convolutional filters.
3. **Privacy Verification:** Local models failed significantly when tested on unseen classes (e.g., Truck specialists detecting Cyclists). This confirms that raw data remained private and no information leaked during the training process.
4. **Aggregation Value:** While individual clients suffered from catastrophic forgetting regarding unseen classes, the central server successfully aggregated these specialized local models into a high-performing generalist model.

5.2 Scope for Future Work

Future research can extend this framework through:

- **Advanced Algorithms:** Implementing strategies like FedProx or SCAFFOLD to explicitly correct the client drift observed in our non-IID experiments.

- **Efficiency:** Reducing bandwidth usage via gradient quantization or partial model transmission (e.g., updating only the detection heads).
- **Security:** Integrating differential privacy and robust aggregation protocols to defend against model poisoning attacks.
- **Real-World Deployment:** Testing the framework on dynamic video streams to evaluate temporal consistency and real-time inference on edge hardware.

In conclusion, this research confirms that state-of-the-art object detection can be effectively decentralized. The results highlight the distinct advantage of Transformer-based architectures in handling the weight divergence inherent to federated learning.

REFERENCES

[1] **N. Carion**, **F. Massa**, **G. Synnaeve**, **N. Usunier**, **A. Kirillov**, and **S. Zagoruyko** (2020). End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*.

[2] **G. G. Devarajan**, **M. Thirunavukkarasan**, **M. Mahalinghum**, **D. Arockiam**, **M.-E. Wu**, and **R. P. Mahapatra** (2025). Explainable federated learning-based secure and transparent object detection model for autonomous vehicles. *IEEE Transactions on Consumer Electronics*, **71**(2), 6263–6270.

[3] **A. Geiger**, **P. Lenz**, **C. Stiller**, and **R. Urtasun** (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, **32**(11), 1231–1237.

[4] **Google** (2017). Federated learning: Collaborative machine learning without centralized training data.

[5] **G. Jocher**, **A. Chaurasia**, and **J. Qiu** (2023). Ultralytics yolov8. URL https://github.com/ultralytics/ultralytics. Accessed: 2024-05-20.

[6] **H. Kopka** and **P. W. Daly**, *Guide to LaTeX*. Addison-Wesley Professional, 2003.

[7] **T. Li**, **A. K. Sahu**, **M. Zaheer**, **M. Sanjabi**, **A. Talwalkar**, and **V. Smith**, Federated optimization in heterogeneous networks. *In Proceedings of Machine Learning and Systems*, volume 2. 2020.

[8] **Y. Zhao**, **M. Li**, **L. Lai**, **N. Suda**, **D. Civin**, and **V. Chandra** (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.