Containerized Federated Learning Approach for Autonomous Driving Model Training

Yatish Sikka, Anisha Katiyar, Aariz Faridi, Nikita Miller
M.S. Applied Machine Learning
University of Maryland,
College Park, Maryland

***Abstract*—This project proposes a containerized federated learning (FL) pipeline to train real-time object detection models in autonomous driving systems, focusing on system monitoring, data privacy, and deployment efficiency. By leveraging YOLOv9t as the core model and orchestrating training with Flower, our system enables decentralized learning across distributed clients without sharing raw sensor data. Clients are containerized using Docker and dynamically instantiated via Amazon Machine Images (AMI), allowing for efficient onboarding during FL rounds. Class imbalance is addressed through targeted data augmentation using Albumentations, while Optuna is used for hyperparameter tuning at the client level. Performance and model drift are continuously monitored using ClearML and AWS CloudWatch. Experimental results demonstrate consistent improvements in mAP@0.5 and mAP@0.5–0.95 scores, efficient model drift detection, and reduced variance across training rounds compared to non-containerized baselines. This work highlights a production-ready FL architecture combining MLOps, edge intelligence, and scalable training for privacy-preserving autonomous vehicle model deployment. Source code for this project can be found at:**
https://github.com/aariz11858/Containerized-Federated-Learning-for-Autonomous-Driving-Utilizing-MLOps.

Keywords: Federated Learning, YOLOv9, Containerization, MLOps, Autonomous Driving, Optuna, Model Drift, Data Privacy, ClearML, AWS

I. Introduction

The advancement of autonomous driving systems has introduced new challenges in the training of large-scale machine learning models on sensor-rich data such as images, LiDAR, and GPS. These models power critical functions such as object detection, pedestrian tracking, and traffic light recognition, all of which require high accuracy and adaptability to diverse and dynamic environments. Traditionally, such models are trained in centralized environments where raw data is aggregated into a single server. However, this approach raises significant concerns around privacy, data transmission overhead, and system scalability—particularly when deployed across a fleet of autonomous vehicles operating in varied geographic conditions.

To address these challenges, Federated Learning (FL) has emerged as a privacy-preserving solution that enables decentralized model training across distributed clients without transferring raw data. FL allows autonomous vehicles to learn collaboratively from diverse conditions while preserving user privacy and reducing communication overhead. By leveraging local computation and sharing only model updates, FL supports continuous improvement of models across fleets, enabling personalization, adaptability, and on-device learning. This makes it particularly well-suited for autonomous driving, where responsiveness, security, and large-scale deployment are all mission-critical.

In this project, we propose a containerized federated learning framework for autonomous driving that integrates modern MLOps practices, scalable cloud infrastructure, and real-time optimization. Our system leverages YOLOv9t for fast and accurate object detection, uses Flower to orchestrate the FL process, and is containerized using Docker for reproducibility. Each client is dynamically instantiated using Amazon Machine Images (AMI), enabling horizontal scalability during federated rounds. Class imbalance is addressed through targeted data augmentation strategies, while Optuna performs client-specific hyperparameter tuning. Finally, performance and model drift are continuously monitored using ClearML and AWS CloudWatch, making the pipeline robust, adaptable, and production-ready. This work provides a practical blueprint for deploying scalable, privacy-preserving, and cloud-integrated FL systems in real-world autonomous vehicle networks.

II. Literature Review

The advent of autonomous driving has created an urgent need for robust, privacy-preserving machine learning models that can learn from diverse, real-world data. The traditional centralized approach to training such models, where raw data is shared with a central server, raises significant concerns related to privacy, security, and data ownership. In response to these concerns, federated learning (FL) has gained traction as an approach that allows models to be trained on decentralized data sources, enabling devices (or vehicles) to process and learn from data locally and share only model updates. This approach preserves the privacy of sensitive data while still benefiting from collaborative learning across multiple devices.

A number of studies have highlighted the advantages of federated learning in autonomous driving systems. For instance, McMahan et al.[1] introduced the concept of federated averaging, a method that enables decentralized training in a privacy-preserving manner, while exploring its effectiveness for a range of applications, including autonomous vehicles. Additionally, research by Konecny et al.[2] demonstrated how federated learning could be applied in the context of smartphones, showing its potential for collaborative learning

across distributed data sources without compromising privacy. More recently, Yang et al.[3] expanded on the scalability of federated learning in large-scale systems, emphasizing its role in training models across highly distributed data sources, such as those found in autonomous driving.

However, while federated learning addresses privacy concerns and allows for decentralized model training, there are challenges related to system orchestration, efficiency, and model management. This is where containerization and MLOps become crucial. Containerization allows for the easy deployment and efficient scaling of the federated learning system, ensuring that components can be efficiently managed across different nodes and environments. Furthermore, MLOps practices, which include automation of model development, deployment, and monitoring, have been shown to streamline the operational aspects of machine learning systems. Work by Sculley et al.[4] has highlighted the importance of MLOps in managing the end-to-end lifecycle of machine learning models, reducing the complexity of deploying systems at scale.

III. Problem Statement

Autonomous vehicles rely heavily on continuous learning from real-world driving data to maintain and improve their perception capabilities. Traditionally, this data is collected centrally, aggregated from multiple sources into a centralized server where models are trained and updated. However, this approach introduces significant privacy risks as sensitive location and driving data is transmitted and stored in central repositories. Additionally, it poses challenges in terms of deployment efficiency and latency, especially as the volume of data from vehicles increases exponentially.

Existing Federated Learning (FL) solutions provide a promising alternative by enabling decentralized training, where models are trained directly on-device (or locally) and only model updates are shared with a central server. However, most FL implementations are either research-focused or lack the orchestration, real-time monitoring, and deployment readiness required for production-level autonomous driving systems.

Key challenges include:

- **Orchestration inefficiency**: Synchronizing clients and managing training rounds across dynamic, large-scale fleets is complex.
- **Model monitoring limitations**: Detecting model drift and performance degradation in real-time is often lacking in existing FL frameworks.
- **Deployment efficiency and automation gaps**: Onboarding new clients and handling infrastructure failures is not streamlined.

Thus, there is a clear need for an efficient, privacy-preserving FL pipeline tailored for autonomous vehicles, which integrates efficient onboarding, real-time model monitoring, and production-ready deployment mechanisms. This project aims to address these gaps by building an end-to-end solution combining FL with MLOps practices, leveraging tools like Flower, YOLOv9, Docker, ClearML, and AWS CloudWatch.

IV. Solution and Its Significance

A. Proposed Solution

To address the challenges of centralized training in autonomous driving, we propose a fully modular and reproducible Federated Learning (FL) pipeline, augmented with MLOps best practices. The core of our solution integrates hyperparameter optimization, orchestrated training, model aggregation, monitoring, and deployment automation to enable privacy-preserving, efficient, and real-time model updates for self-driving vehicles.

The pipeline consists of the following key components:

- **Data Preprocessing and Augmentation:** The initial stage involves cleaning, formatting, and augmenting the driving dataset to prevent class imbalance. Techniques like Gaussian blur, horizontal flip, contrast adjustments, and random rotation and scaling were applied to enhance generalization and robustness, critical for diverse real-world scenarios.
- **Hyperparameter Optimization with Optuna:** Rather than relying on manual or brute-force methods like Grid or Random Search, we leveraged Optuna's Tree-structured Parzen Estimator (TPE) for efficient hyperparameter optimization. Optuna adaptively explores promising regions of the search space while pruning sub-optimal configurations early, drastically reducing computation time. We tuned a comprehensive set of hyperparameters critical to the training stability and performance of YOLOv9:
  - **Learning Rate:** Controls the step size during optimization updates. A fine-tuned learning rate ensures stable convergence without overshooting or stagnation.
  - **Momentum:** Helps accelerate gradients in the correct direction, smoothing oscillations and improving convergence speed and stability.
  - **Weight Decay:** Regularizes model complexity by penalizing large weights, thus reducing overfitting and enhancing generalization.
  - **Warmup Epochs:** Gradually increases the learning rate during initial epochs to stabilize early training phases and prevent divergence.
  - **HSV Hue (`hsv_h`):** Adjusts color augmentation by shifting image hue. Excessive values can distort color fidelity, so a conservative range is used.
  - **Mosaic Augmentation:** Combines multiple images into one during training to improve spatial context learning. Values below 0.5 harm generalization capability.
  - **Mixup Augmentation:** Blends two images and their labels, enhancing model robustness but potentially causing label ambiguity if set too high.
  - **Rotation Degrees:** Applies random image rotations to improve robustness to object orientation variations. Excessive rotation can introduce unrealistic augmentations.

– **Scale Jittering:** Randomly scales images to simulate varying object sizes and camera distances, enhancing model adaptability to real-world scenarios.

All hyperparameters were optimized with respect to the `mAP@0.5-0.95` metric, ensuring a balanced evaluation across multiple Intersection-over-Union (IoU) thresholds.

- **Federated Learning with Flower:** Flower orchestrates decentralized training across multiple clients (vehicles) and a central server. Each client trains locally on its on-device data, and only model weight updates are shared with the server. The server performs secure FedAvg aggregation to update the global model without exposing raw data.
- **Monitoring with ClearML and AWS CloudWatch:** ClearML tracks all experiment metadata, including hyperparameters, model versions, and performance metrics. Continuous monitoring of mAP@0.5 and mAP@0.5-0.95 allows early detection of data drift and performance degradation. AWS CloudWatch extends this by tracking system-level metrics such as client uptime, CPU and Memory usage and network activity.
- **Client Instantiation:** To enable seamless and efficient deployment of federated learning clients, we implemented a custom deployment pipeline leveraging Amazon Machine Images (AMIs). Initially, a single EC2 instance was configured with the required YOLOv9 training environment, including Dockerized containers for reproducible execution. This base instance was then converted into an AMI, capturing the full client environment. Subsequent client nodes were instantiated directly from this AMI, ensuring fast and consistent deployment of new clients. This approach significantly reduced setup times, as new instances inherited all pre-configured dependencies and container images without requiring repetitive provisioning steps. This AMI-based deployment strategy offered a practical and cost-effective solution for instantiating federated learning clients in real-world scenarios, combining the benefits of Docker reproducibility with the reliability of AWS infrastructure.

B. Pipeline Workflow

Figure 1 illustrates the hyperparameter optimization work flow. The pipeline begins with data preprocessing, followed by YOLOv9t model training on an 80% split of the dataset. Hyperparameter optimization via Optuna iteratively refines the model, with all results logged and visualized in ClearML. Figure 2 illustrates the Federated Learning and MLOps architecture. We first instantiate each client's YOLOv9 modelwith the best model weights to run 1 epoch on. The server aggregates client updates using the Flower framework, evaluates performance, and checks for significant degradation ($> 10\%$ drop in mAP@0.5-0.95). If data drift is detected, the server reverts to the last stable model. This loop continues with real-time monitoring through ClearML and AWS CloudWatch.

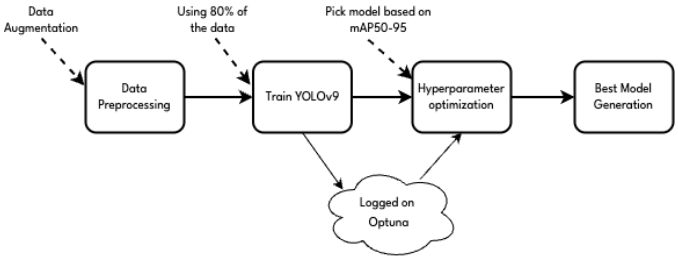

Fig. 1: Hyperparameter Optimization Flow

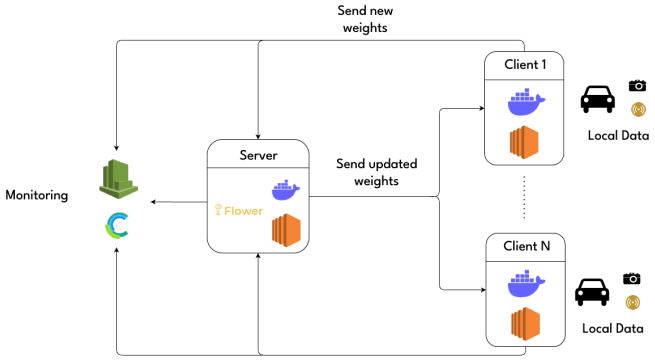

Fig. 2: Federated Learning Pipeline with MLOps Integration

C. Significance and Novelty

- **Privacy-Preserving FL:** By decentralizing training, sensitive raw data never leaves the client devices, aligning with privacy regulations and enhancing data security.
- **Efficient Model Optimization:** Optuna's adaptive sampling and pruning significantly reduced optimization time compared to traditional grid and random search, ensuring faster convergence to high-performing hyperparameter configurations.
- **End-to-End Monitoring:** The integration of ClearML and AWS provides continuous experiment tracking and resource monitoring, addressing key operational challenges in real-world FL deployments.
- **Robustness via Performance Thresholds:** Implementing a performance safeguard where model updates are only accepted if they meet or exceed previous benchmarks ($> 10\%$ threshold) ensures system reliability and prevents model degradation.
- **Efficient Deployment Architecture:** Dockerized environments and AMI integration enable effortless horizontal scaling of clients, with rapid instantiation of new nodes.

This containerized and modular approach ensures that our federated learning system is technically robust, reproducible, and efficient. It is well-suited for real-world deployment in autonomous vehicle applications, where reliability, environment consistency, and efficient resource utilization are critical.

V. Evaluation Results

A. Hyperparameter Optimization

To assess model accuracy, we used the mean Average Precision (mAP@0.5-0.95) metric, which evaluates detection performance across multiple Intersection-over-Union (IoU) thresholds. Figure 3 illustrates the comparative performance of baseline and Optuna-optimized models.

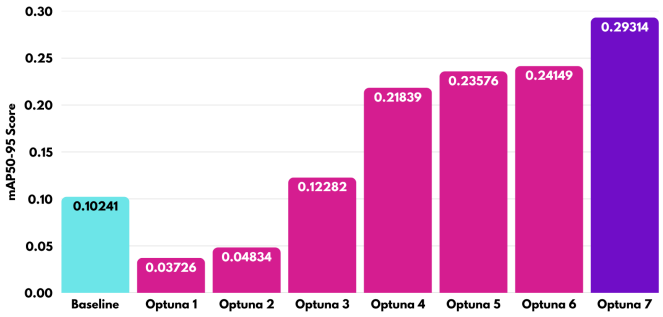

Fig. 3: mAP@0.5-0.95 Results for Baseline, and Optuna-Tuned Models

The Optuna-tuned model achieved the highest mAP@0.5-0.95 score of 0.293, significantly outperforming the baseline configuration. The baseline model was generated by randomly sampling hyperparameter values from the same search space provided to Optuna, ensuring a fair comparison of the efficiency of the optimization.

B. MLOps Metrics and Monitoring

A major contribution of this project is the seamless integration of MLOps into the FL workflow. Key monitoring and automation features include:

- **ClearML Tracking:** All experiments, including hyperparameter configurations, performance metrics (mAP@0.5–0.95), and model versioning, were logged and visualized through ClearML dashboards. This enabled real-time comparison of FL model versions, ensuring reproducibility.
- **Containerized vs Non-Containerized Comparison:** A key evaluation was the impact of containerization on model performance. As shown in Figure 4, the containerized FL setup consistently achieved higher mAP@0.5-0.95 scores across rounds compared to the non-containerized setup. This improvement can be attributed to the consistent training environments provided by Docker, which reduced variability and enhanced convergence stability across clients.

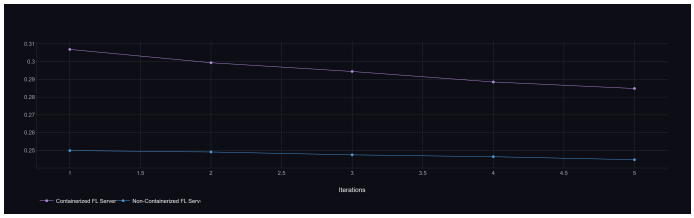

Fig. 4: Comparison of mAP@0.5-0.95 scores across rounds: Containerized vs Non-Containerized FL

- **Data Drift Detection:** A threshold-based mechanism was implemented where the FL server compares the current round's mAP@0.5-0.95 to the previous round. If a degradation of more than **10%** is detected, the server reverts to the last stable model to prevent performance regressions. As seen in Figure 5, the validation curve never falls sharply across rounds, indicating that the drift protection mechanism effectively prevents large performance drops.

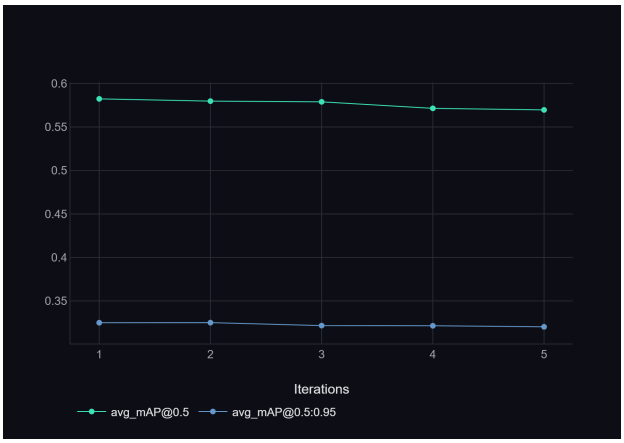

Fig. 5: Validation scores per round with drift protection enabled.

- **Client Participation Metrics:** ClearML was used to track the number of active clients participating in each training round. As shown in Figure 6, client participation increased progressively as new clients were started in later rounds, demonstrating the dynamic scalability of our FL system. This ensures transparency in FL contributions and client failure detection.

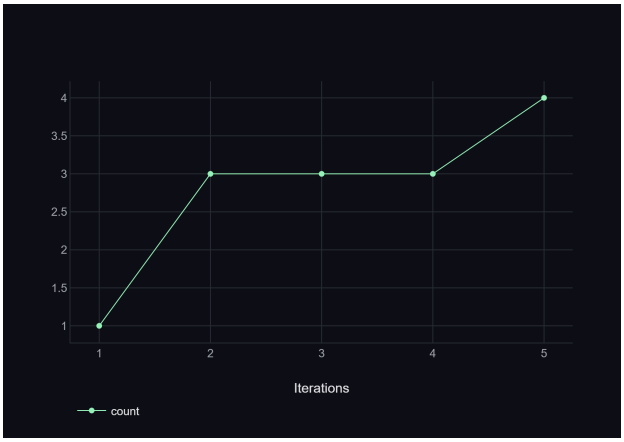

Fig. 6: Number of clients participating per round.

C. Client Deployment Efficiency

One of the critical factors in evaluating FL pipelines is deployment efficiency, i.e. the system's ability to instantiate new client nodes seamlessly. We measured:

- **Client Startup Time:** Average time taken to instantiate and connect a new client instance to the FL server was observed at **2 seconds** using AMI-based cloning on AWS EC2. By pre-configuring the client environment into an AMI, instance launch times were reduced significantly compared to traditional provisioning methods. Docker containers ensured environment consistency, while AMIs expedited deployment by eliminating redundant setup steps.
- **Round Duration:** The average federated learning (FL) round, including training and aggregation, took 5 minutes with 3 clients. Since the server synchronizes at round boundaries, new clients joining mid-round must wait until the current round completes to participate. This design ensures consistency but ties client onboarding latency to round duration. However, thanks to pre-configured Dockerized AMIs, new clients are deployed quickly and are ready to join the next round with minimal overhead, maintaining deployment efficiency.

These results indicate that the pipeline scales horizontally with minimal manual intervention, ensuring adaptability to varying workload demands.

D. Infrastructure Performance

- **AWS CloudWatch Metrics:** Network activity, instance uptimes, and resource utilization (CPU, memory) were monitored via AWS CloudWatch. Automated alarms were configured to detect anomalies like stalled clients or resource saturation.
- **Cost Efficiency:** Spot instances and lightweight YOLOv9t (tiny) models were utilized to balance cost and performance, achieving scalable training at optimized compute costs.

E. MLOps Significance in Federated Learning

The robust MLOps foundation provided several critical benefits:

- Automated deployment, monitoring, and scaling reduced manual overhead, making FL operations production-ready.
- Experiment reproducibility and continuous tracking streamlined model lifecycle management.
- Real-time performance validation ensured reliable model updates without compromising accuracy.

The use of AMI-based instance cloning provided a robust mechanism for deploying federated learning clients while maintaining environment consistency. This approach, combined with Docker containerization and ClearML monitoring, exemplified MLOps principles by automating deployment, reducing operational overhead, and ensuring reproducibility across dynamically instantiated clients.

Overall, the integrated MLOps-driven FL pipeline demonstrated significant improvements in model performance, operational efficiency, and deployment scalability, aligning with industry-standard practices for autonomous vehicle applications.

VI. Limitations and Future Scope

A. Limitations

Despite the successful implementation of a containerized, cloud-native federated learning pipeline, several limitations were observed during the project:

- **Simulated Clients:** FL clients were run in isolated containers on cloud infrastructure (AWS EC2), not on real vehicles or edge devices. This limited our ability to evaluate hardware constraints, latency, and true on-device behavior in real-world driving conditions.
- **Fixed Non-IID Data Split:** Each client was assigned a static subset of data simulating non-IID distribution. In practical applications, data distributions may evolve over time or vary dynamically across regions, requiring more advanced mechanisms for handling data drift and distribution shifts.
- **Federated Aggregation Simplicity:** We used the standard FedAvg algorithm to aggregate model weights. While effective for proof of concept, it does not consider factors like varying data quality, client importance, stragglers, or potential adversarial updates—factors critical for robust production systems.
- **Static Label Set Limitation:** The model was trained on a predefined and static label set derived from the dataset. If an object class (e.g., a pedestrian or cyclist) is not present in the original training labels, the model cannot detect it at runtime. This limits the adaptability of the system to unseen object types or evolving environments unless the dataset and labels are updated proactively.

B. Future Scope

Building upon the current system, several enhancements can be explored to elevate the pipeline's scalability, reliability, and real-world applicability:

- **Non-IID Federated Learning Experiments:** Introducing heterogeneous datasets across clients to simulate realistic driving conditions will help assess the model's generalization and FL system's aggregation robustness.
- **Advanced Aggregation Techniques:** Incorporating algorithms like FedProx, FedNova, or secure aggregation methods will improve resilience against straggler clients, unreliable updates, and privacy threats.
- **Dynamic Dataset Handling:** Automating data ingestion pipelines where new clients receive dynamically assigned datasets upon instantiation would improve scalability for large-scale deployments.
- **Server-Side Scalability via Hierarchical FL:** To handle scenarios with high client concurrency or uptime peaks, multiple federated learning servers could be instantiated simultaneously—each managing a distinct client subset. This enables horizontal scaling beyond a single coordinator. After training rounds, model updates from each server can be evaluated and either the best-performing model adopted or their parameters meta-aggregated using hierarchical FL techniques. This approach would support true distributed orchestration under load.

In conclusion, while the current implementation validates the viability of a containerized, MLOps-driven federated learning pipeline for autonomous vehicle perception tasks, the outlined future enhancements will pave the way for its deployment in large-scale, real-world applications.

VII. Team Contributions and Tools

A. Individual Contributions

- **Anisha:** Implemented data augmentation techniques and focused on hyperparameter optimization using Optuna, configuring search spaces, and analyzing optimization results to make the custom YOLOv9 model.
- **Aariz:** Designed and implementated the Federated Learning (FL) pipeline using Flower. Developed the dynamic client instantiation logic and integrated the custom YOLOv9 model.
- **Nikita:** Implemented MLOps monitoring stack, including CloudWatch metrics for round duration and client participation. Contributed to data drift detection mechanisms and managed ClearML experiment tracking and visualization.
- **Yatish:** Developed a containerized federated learning framework using Docker. Designed a scalable client-server architecture with automated deployment and recovery, and created a custom AMI for rapid client initialization.

B. Implementation Tools

The following tools and frameworks were employed to realize the project:

Tool	Category	Purpose
Optuna	MLOps	Performed hyperparameter tuning for YOLOv9 baseline
Flower	Federated Learning	Orchestrated communication between clients and server; performed secure weight aggregation via FedAvg
Docker	Containerization	Containerized each client node and server for reproducible and isolated execution
ClearML	MLOps	Tracked model metrics (mAP50–95), model drift, system metrics, client participation, and FL round duration
AWS CloudWatch	MLOps	Monitored system usage and infrastructure-level logs

TABLE I: Tools used in the project and their corresponding roles.

References

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & A. y Arcas, B. (2017). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS). arXiv preprint arXiv:1602.05629.

[2] Konecny, J., McMahan, H. B., Yu, F. X., Richtarik, P., Suresh, A. T., & Bacon, D. (2016). *Federated Learning: Strategies for Improving Communication Efficiency*. arXiv preprint arXiv:1610.05492.

[3] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2020). *Federated Machine Learning: Concept and Applications*. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1-19.

[4] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., & Young, M. (2015). *Hidden Technical Debt in Machine Learning Systems*. Advances in Neural Information Processing Systems (NeurIPS), 28. arXiv preprint arXiv:1506.00666.

[5] C. Wang, A. Bochkovskiy, and H. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[6] Jocher, Glenn, et al. YOLOv8: Next-generation object detection. *Ultralytics*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[7] S.H. Tsang, "Review: YOLOv8 Object Detection," Medium, 2023. [Online]. Available: https://sh-tsang.medium.com/review-yolov8-object-detection-5214fa105731.

[8] C. Wang, A. Bochkovskiy, and H. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," *arXiv preprint arXiv:2402.13616*, 2024.

[9] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10781-10790.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," *arXiv preprint arXiv:2005.12872*, 2020.

[11] B. Li *et al.*, "Equalized Focal Loss for Dense Long-Tailed Object Detection," in *CVPR*, 2022, pp. 6990–6999.

[12] Y. Ma *et al.*, "Information Amount-Guided Angular Margin Loss for Long-Tailed Object Detection," in *ICLR*, 2025.

[13] B. Yaman *et al.*, "Instance-Aware Repeat Factor Sampling for Long-Tailed Object Detection," arXiv:2305.08069, 2023.

[14] J. Yang *et al.*, "Long-Tailed Object Detection for Multimodal Remote Sensing Images," *Remote Sensing*, vol. 15, no. 18, 4539, 2023.

[15] S. Lin *et al.*, "Explore the Power of Synthetic Data on Few-Shot Object Detection," in *CVPR Workshop (GCV)*, 2023.