Computational Offloading and Delay Minimization for UAV-aided Edge Federated Learning

by

Mudassar Liaq

A dissertation

presented to Lakehed University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the program of

Electrical and Computer Engineering

Lakehead University

December 2025

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Lakehead University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Lakehead University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Publications

- **M. Liaq** and **W. Ejaz**, "Delay Optimization in Hierarchical UAV-aided Federated Learning for Straggling IoT Devices," IEEE Internet of Things Journal, Dec. 2025.
- **M. Liaq** and **W. Ejaz**, "Minimizing Delay in Queuing-based UAV-aided Federated Learning with Straggling IoT Devices," in IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Sep. 2025.
- **M. Liaq**, S. Sharif, S. Zeadally, and **W. Ejaz**, "Utilization of Machine Learning in Future Wireless Networks for Resource Optimization: A Survey," Ad Hoc Networks, p. 103983, Jul. 2025.
- **M. Liaq** and **W. Ejaz**, "Minimizing Delay in UAV-Aided Federated Learning for IoT Applications With Straggling Devices," IEEE Open Journal of the Communications Society, vol. 5, no. 1, pp. 7653–7667, Dec. 2024.
- **M. Liaq** and **W. Ejaz**, "Computational Offloading and Delay Minimization for UAV-aided Edge Federated Learning," in IEEE International Conference on Communications (ICC), Jun. 2024.
- **M. Liaq** and **W. Ejaz**, "Delay-Optimized Hierarchical UAV-MEC Federated Learning Using Federated GANs for IoT Networks," IEEE Transactions on Machine Learning in Communications and Networking, Dec. 2025. **(Submitted)**.

Examining Committee Membership

The following served on the Examining Committee for this thesis.

Supervisor:	Dr. Waleed Ejaz LU Research Chair & Associate Professor Department of Electrical and Computer Engineering Lakehead University
External Examiner:	Dr. Isaac Woungang Professor Department of Computer Science Toronto Metropolitan University
Internal Examiner:	Dr. Salama Ikki Professor Department of Electrical and Computer Engineering Lakehead University
Internal Examiner:	Dr. Thangarajah Akilan Associate Professor Department of Software Engineering Lakehead University

Abstract

Future wireless networks are expected to support increasingly diverse, data-intensive, and latency-critical applications driven by massive deployments of Internet of things (IoT) devices. As dynamic traffic grows across next-generation wireless systems, efficient resource optimization becomes essential to ensure reliability, scalability, and timely decision-making. Traditional optimization techniques, while mathematically rigorous, often suffer from high computational complexity and limited adaptability in large-scale heterogeneous environments. In contrast, machine learning has emerged as a powerful alternative for resource optimization in future wireless networks. This thesis leverages insights from recent survey findings on ML-based resource optimization, particularly within federated learning (FL) environments, to design adaptable optimization frameworks suited for dynamic wireless systems.

Building upon this foundation, the thesis presents three increasingly capable uncrewed aerial vehicle (UAV)-assisted FL frameworks. First, we develop a UAV-aided FL (UAFL) system that mitigates the straggler effect by offloading partial datasets from resource-constrained IoT devices to UAV-mounted mobile edge computing (MEC) servers. We formulate a system-delay minimization problem under computation, communication, and quality of service (QoS) constraints, solving it using epigraph transformation, deterministic simplex optimization, and deep reinforcement learning (DRL) for improved run-time performance.

Next, we propose a hierarchical queue-based UAV-aided FL framework. The system introduces multi-queue architectures at IoT devices, follower UAVs, and leader UAVs to manage irregular data arrivals and heterogeneous processing capabilities. We apply Lyapunov drift-plus-penalty optimization to stabilize queues and minimize delay by decomposing the problem into sequential quadratic programming (SQP)-solvable subproblems. Simulation results demonstrate significant improvements in delay reduction, queue stability, and overall system throughput compared to standard FL and UAFL.

Finally, we extend the hierarchical system by integrating generative adversarial networks (GANs) to counter the impact of non-independent and identical data (Non-IID) data across distributed devices. Offloaded datasets are utilized to train distributed GANs at follower UAVs, while aggregated generator models at the leader UAV produce synthetic samples that enhance data diversity. This GAN-augmented HUAFL framework improves model accuracy, accelerates convergence, and maintains queue stability within a UAV-MEC-assisted

architecture. Experimental results show that this integrated approach outperforms baseline FL, hierarchical FL, and UAFL across accuracy, latency, and resource utilization metrics.

Overall, this thesis presents a unified, scalable, and computation-efficient set of frameworks for FL in mission-critical IoT environments. By integrating ML-based resource optimization, hierarchical queuing, and GAN-based data augmentation, the proposed approaches advance the state of UAV-assisted edge intelligence for next-generation wireless networks.

Acknowledgements

Salma Yasmeen, I am proud to be your son.

Everything I am is by the grace of the Almighty; after that, you are the greatest reason behind every success I have achieved.

I would like to thank all the people who made this thesis possible. My deepest appreciation goes to Dr. Waleed Ejaz for his supervision, mentorship, and thoughtful advice. I am grateful to my colleagues and members of the Wireless Communication & Networks (WCN) Research Group for sharing knowledge, ideas, and encouragement. I would like to especially thank Mohammad Akram and Sana Sharif for their unwavering support, guidance, and warm companionship throughout this journey.

To my wife, Wafa Shafqat, whose presence has been invaluable throughout this journey; and to my sisters, Shahwana and Ridda, for their unwavering support. To Waqar Ahmed, and to my nephews, Aayad Waqar and Aarib Waqar, thank you for bringing joy and brightness into my everyday life. To Aayzel Waqar, who came into this world and left it without me by her side; to my mother, as time reshapes her days; and most importantly, to my father, Liaq Mohammad Khan - for every single absence, I am sorry. Liaq Sahab, these are Muneer Niazi's words, yet I feel them deeply when I think of you and Miss Salma.

ہمیشہ دیر کر دیتا ہوں میں ہر کام کرنے میں
ضروری بات کہنی ہو کوئی وعدہ نبھانا ہو
اسے آواز دینی ہو اسے واپس بلانا ہو
ہمیشہ دیر کر دیتا ہوں میں

مدد کرنی ہو اس کی یار کی ڈھارس بندھانا ہو
بہت دیرینہ رستوں پر کسی سے ملنے جانا ہو
ہمیشہ دیر کر دیتا ہوں میں

بدلتے موسموں کی سیر میں دل کو لگانا ہو
کسی کو یاد رکھنا ہو کسی کو بھول جانا ہو
ہمیشہ دیر کر دیتا ہوں میں

کسی کو موت سے پہلے کسی غم سے بچانا ہو
حقیقت اور تھی کچھ اس کو جا کے یہ بتانا ہو
ہمیشہ دیر کر دیتا ہوں میں ہر کام کرنے میں۔۔۔۔۔

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acronyms	Description
5G	Fifth-generation
6G	Sixth-generation
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
B&B	Branch and Bound
BS	Base Station
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
FC	Fully Connected
FL	Federated Learning
GAN	Generative Adversarial Network
Gen-AI	Generative AI
HFL	Hierarchical Federated Learning
IID	Independent and Identically Distributed Data
IoT	Internet of Things
IoV	Internet of Vehicles
KLD	Kullback–Leibler Divergence
MEC	Mobile Edge Computing
MINLP	Mixed-Integer Nonlinear Programming
ML	Machine Learning
Non-IID	Non-Independent and Identically Distributed Data
NP	Nondeterministic Polynomial Time
PCCP	Penalty Convex-Concave Procedure
QoE	Quality of Experience
QoS	Quality of Service
SQP	Sequential Quadratic Programming
SVM	Support Vector Machine

UAFL	UAV-Assisted Federated Learning
UAV	Uncrewed Aerial Vehicle
URLLC	Ultra Reliable Low Latency Communication

List of Symbols

Symbol	Description
A_k	Number of data batches generated by IoT device k
$A_{k,w}(t)$	Dataset batches expected to arrive at time $t+w$ at IoT device k
$A_{k,-1}(t)$	Arrival queue containing currently available batches at time t for device k
B_k^n	Number of mini-batches per epoch for IoT device k
b_k	Number of records in each mini-batch at IoT device k
$b_k^{(I,l)}(t)$	Batches forwarded to local processing at IoT device k at time t
$b_k^{(I,o)}(t)$	Batches offloaded from IoT device k at time t
$b_m^{(U,l)}(t)$	Dataset batches processed locally at UAV m
$b_m^{(U,o)}(t)$	Dataset batches offloaded by UAV m to the leader UAV at time t
$\beth_k$	Total computation cycles required for FL training at IoT device k
C_k	Dataset of IoT device k
$\lvert C_k \rvert$	Dataset size (in bits) of IoT device k
D_k^O	Offloading delay of IoT device k
$D_k^{(I,l)}$	Computation delay at IoT device k
$D_m^{(U,l)}$	Computation delay at UAV m
$D_m(t)$	Transmission capacity from UAV m to leader UAV
D_{ab}	Aggregation and broadcasting delay
F_k	Loss function over dataset of IoT device k
f_i	Loss for a single data record
G_k	Channel gain between IoT device k and UAV
$Q_k^{(I,\alpha)}(t)$	Integrated backlog of all queues at IoT device k
$Q_k^{(I,o)}(t)$	Offloading queue at IoT device k
$Q_k^{(I,l)}(t)$	Local processing queue at IoT device k
$Q_m^{(U,\alpha)}(t)$	Arrival queue at UAV m

$Q_m^{(U,l)}(t)$	Local processing queue at UAV m
$Q_m^{(U,o)}(t)$	Offloading queue at UAV m
h	Priority configuration variable ($h \in [0,1]$)
$\hat{B}$	Bandwidth of the communication link
L_k	Distance between IoT device k and the UAV-MEC server
λ	CPU cycles required to train 1 bit of data
N	Number of global FL iterations required for convergence
ω_k	Local model parameters of IoT device k
Ω_m^U	Model parameters computed at UAV m
ω	Global aggregated model parameters
$p_k^{(I,o)}(t)$	Transmit power of IoT device k for offloading at time t
$p_k^{(I,l)}(t)$	Local processing power of IoT device k at time t
$p_m^{(U,o)}(t)$	Transmit power of UAV m for offloading at time t
$p_m^{(U,l)}(t)$	Local processing power of UAV m at time t
P_k	Power budget of IoT device k
P_k^O	Transmission power of IoT device k
$P_{\min}^I, P_{\max}^I$	Min/max power constraints at IoT devices
$P_{\min}^U, P_{\max}^U$	Min/max power constraints at UAVs
R_k	Number of data samples (rows) in dataset of IoT device k
R	Total number of processed rows across IoT devices
s_k	CPU computation capacity of IoT device k
s_m^U	CPU computation capacity of UAV m
ν_k	Capacitance coefficient for IoT device k
ν_m^U	Capacitance coefficient for UAV m
$\mathfrak{S}_q$	Combined dataset of offloaded data at UAV-MEC server q
$\mathfrak{s}_q$	Size of aggregated offloaded dataset at UAV-MEC server q
t_m^P	Processing power available at UAV m
T_m	Power budget at UAV m

θ_k^I	Number of rows offloaded by IoT device k
Θ_k^I	Offloaded dataset of IoT device k
θ	Total number of rows offloaded across IoT devices
$\sigma_{k,t}$	Magnitude of local gradient of device k at iteration t
$\bigtriangledown f(\omega_k : x_i, y_i)$	Gradient computed for training sample (x_i, y_i)
Υ	Number of local training epochs per FL iteration
χ_k	Selection variable for IoT device k
U	Set of selected IoT devices for the current iteration
ϱ	Angle of IoT device k with respect to UAV-MEC server
φ_k	Path-loss factor for IoT device k
$\eta_{\text{LoS}}, \eta_{\text{NLoS}}, a, b$	Communication environment parameters
ϵ	Target training loss threshold

Chapter 1

Introduction

The rapid expansion of the Internet of things (IoT), autonomous systems, and intelligent sensing platforms has transformed the landscape of future wireless networks. Modern mission-critical applications, such as forest fire monitoring, flash flood detection, border surveillance, environmental sensing, and wildlife tracking, require the ability to collect, process, and transmit large quantities of real-time data under stringent latency and reliability constraints. These systems are commonly deployed in remote, harsh, or infrastructure-limited environments, where stable connectivity remains uncertain.

Recent real-world disasters, such as the severe flash flood in San Diego in 2024 that led to hundreds of deaths and displaced more than 800 individuals [1], and the 2025 flash floods in Texas [2] that caused even more damage, demonstrate the need for timely and accurate real-time decision-making. Early detection in such scenarios relies on geographically distributed IoT devices capable of continuous monitoring. However, placing fixed ground-based base stations in large forested or mountainous regions is economically infeasible and operationally inefficient because these areas usually require high connectivity only during specific seasons or extreme weather conditions while remaining idle for the rest of the year.

To overcome these limitations, the integration of uncrewed aerial vehicles (UAVs) into future wireless networks has emerged as a promising solution [3, 4]. UAVs offer flexible

coverage, rapid deployment, mobility, and the ability to cover large geographical areas, making them ideal for supporting IoT deployments in these remote regions. When equipped with mobile edge computing (MEC) capabilities, UAVs can perform localized processing, caching, and computation, thereby reducing latency and minimizing the computational load on spatially distributed on-ground IoT devices.

1.1 Distributed Learning in UAV-Assisted IoT Networks

As IoT deployments scale, traditional centralized machine learning becomes increasingly impractical. Sending raw data to a central server results in high communication costs, especially when datasets are large. It also raises privacy and security concerns because sensitive information must be transmitted over the network. In addition, many remote or resource-constrained environments lack the bandwidth and computational capacity required to support such centralized processing. Federated learning (FL) has emerged as an effective choice [5] in these scenarios for decision making. In FL, data remains on each IoT device, and only model updates or gradients are shared with an aggregation server. This preserves data privacy, reduces communication cost, and enables learning over geographically distributed datasets. However, despite these advantages, practical implementations of FL still faces several fundamental challenges:

- **Straggler Effect:** Devices in a federated network exhibit heterogeneous computing power, dataset sizes, and battery levels. Devices with insufficient computational resources or larger datasets become "stragglers," extending local training time and delaying global model aggregation [6].
- **Resource Constraints:** Both IoT devices and UAV-MECs operate under strict energy budgets. Long-term participation in FL without optimization reduces battery

life, potentially leading to unavailability during critical monitoring periods.

- **Communication Bottlenecks:** IoT devices and UAVs operate over unstable wireless channels. Issues such as fading, shadowing, interference, and path loss create communication bottlenecks, particularly when hundreds of devices simultaneously attempt to transmit updates.

- **Non-independent and identical data (Non-IID) Data:** Data collected from distributed sensors in different geographic locations is often Non-IID. This causes slow convergence and degrades global model accuracy.

Various strategies have been proposed to mitigate these challenges, including partial offloading of local datasets to MEC servers [7,8] and dynamic device selection [9,10]. Nevertheless, integrating these mechanisms into UAV-assisted networks remains challenging due to the inherent mobility limitations, energy budgets, and fluctuating connectivity associated with UAVs. When UAVs are equipped with MEC capabilities, they become natural candidates for offloading in FL-enabled environments. However, due to their limited power, computational resources, and flight time, it becomes crucial to optimize these offloading decisions. This motivates the need for a holistic framework that jointly optimizes offloading decisions, minimizes system delay, and ensures sustained performance in realistic mission-critical deployments. These limitations highlight the necessity for a structured, hierarchical and queue-aware FL approach, which we introduce in the following section.

1.2 Hierarchical and Queue-Aware FL

To address the inherent limitations of single-layer FL, hierarchical federated learning (HFL) has been introduced [11]. While conventional FL relies on a single aggregation layer, HFL introduces a multi-tier structure in which follower UAV-MECs act as intermediate servers and aggregate model updates before forwarding them to a leader UAV in FL iterations. This architecture brings several advantages:

- **Reduced Communication Load:** Local updates from connected IoT devices are aggregated at intermediate UAVs, reducing uplink communication.
- **Better Scalability:** This multi-hop aggregation enables the system to scale to hundreds of IoT devices.
- **Mitigated Straggler Effects:** Workload is distributed across multiple tiers, reducing the number of direct IoT to leader UAV transmissions and reducing computation bottlenecks.
- **Enhanced Robustness:** Failure of individual devices or UAVs does not compromise the entire training pipeline.

Despite its strengths, HFL becomes significantly more complex when real-world data generation patterns are taken into account. IoT devices typically generate dataset records asynchronously, triggered by environmental conditions or events. Consequently, the arrival of data to be processed follows a stochastic and irregular pattern. Ignoring this randomness leads to inaccurate delay models and underestimates system bottlenecks.

Queueing theory provides a natural framework to incorporate these stochastic arrivals by modeling data arrivals, service rates, buffer capacities, and scheduling decisions. Using queues, we can (i) characterize the temporal evolution of local and offloaded datasets, (ii) ensure timely processing of mission-critical information, (iii) perform realistic delay analysis, and (iv) optimize resource allocation based on queue backlogs. Integrating queueing with HFL forms a queue-aware hierarchical FL pipeline capable of handling dynamic data arrivals, heterogeneous device capabilities, and fluctuating network conditions. This queue-aware formulation establishes the foundation upon which we next address the challenges of Non-IID data and the resulting accuracy degradation using GAN-based augmentation.

1.3 Addressing Non-IID Data and Accuracy Degradation via GANs

A major challenge in distributed FL systems is the inherent Non-IID nature of data collected across IoT devices. Sensors placed in different locations, for example, flood-prone versus fire-prone forests, observe entirely different patterns, leading to imbalanced datasets. This regional bias degrades global model accuracy and leads to slow convergence, particularly in classification tasks. Generative adversarial networks (GANs) have emerged as a powerful method for synthetic data generation and augmentation. Recent studies have explored their use in distributed and federated environments [12–14]. Incorporating GANs into HFL, we can augment underrepresented classes, balance dataset distributions across regions, improve model generalization, reduce the number of global iterations required for convergence, and mitigate the impact of rare or infrequent events. The integration of GANs into HFL is particularly beneficial for mission-critical applications, where occasional events (e.g., early signs of a flood or fire) must be detected despite their scarcity. However, combining GANs with HFL introduces new challenges related to UAV energy consumption, computational load, and training delay. This motivates the need for comprehensive system design and optimization strategies to balance improvements in accuracy with overhead. These considerations naturally lead to the broader question of how machine-learning-driven resource optimization can efficiently support such enhanced learning pipelines in future wireless networks, which we explore next.

1.4 Machine Learning Based Resource Optimization in Future Wireless Networks

Future wireless networks (fifth-generation (5G), post-5G, and sixth-generation (6G)) are characterized by ultra-dense deployments, massive connectivity, and extreme performance

requirements. Efficient resource optimization involving spectrum, energy, bandwidth, and computation is consequently essential. Traditional optimization techniques often lack adaptability and scalability under dynamic environments. Machine learning (ML)-based resource optimization [15–18] provides this essential flexibility. It can be used for autonomous learning of optimal resource allocation policies, prediction of future traffic patterns and proactive adaptation, high-precision estimation of channel conditions, reduction of energy consumption through intelligent scheduling, improved load balancing and congestion control, among others.

Our work systematically analyzes these developments, categorizing resource optimization approaches according to the level of machine learning integration and identifying open research challenges. These insights form a foundational layer for the frameworks proposed in this dissertation, motivating the need for machine learning-driven optimization across all tiers of UAV-assisted FL systems.

1.5 Problem Statement

We propose a UAV-assisted federated learning (UAFL) framework where we utilize UAV-MEC computation capacity to process a portion of the datasets from straggling devices to decrease overall system delay. In many real-world IoT deployments, such straggling behavior is common due to heterogeneous device capabilities, varying energy levels, and inconsistent wireless connectivity. By offloading part of the workload to UAV-MEC servers, the overall convergence time of the FL process can be significantly reduced. To achieve this, we formulate an optimization problem aimed at minimizing system delay while considering UAV-MEC computation power, IoT devices' computation and communication power, and quality of service (QoS) constraints. This optimization ensures that task allocation between local IoT processing and UAV-assisted computation is dynamically adjusted to achieve minimal end-to-end delay.

To further enhance system performance, we propose utilizing multiple UAVs instead of a single UAV to cover a larger area populated with IoT devices participating in FL. To address this, we establish a hierarchical structure in which one UAV is designated as the leader UAV, responsible for aggregating updates from all follower UAVs. Each follower UAV aggregates model weights from its associated IoT devices and forwards them to the leader UAV. The leader UAV performs global aggregation and broadcasts the updated global model back to the follower UAVs, which subsequently distribute the model to their connected IoT devices. This hierarchical aggregation approach ensures global consistency, reduces communication overhead, and maintains a structured synchronization pattern across widely distributed UAVs.

We further aim to improve UAFL performance by integrating Generative AI (Gen-AI)-based data augmentation to enhance and balance the dataset. In our system, data generation is realistically assumed to be stochastic, leading to an imbalanced and sparsely represented dataset. Such conditions limit the effectiveness of conventional optimization models, especially those relying on frequent occurrences of certain classes or system states. Our Gen-AI approach generates additional representative samples and ensures a more balanced distribution across dataset classes. This design choice benefits the system in two important ways. First, it improves model accuracy for resource optimization by enabling training on a more balanced dataset. Second, it enables the use of more sophisticated machine learning models for resource allocation and system optimization, which were previously infeasible due to limited data availability. Additionally, the enriched dataset helps the system effectively learn complex edge-case behaviors, improving robustness under rare failures or unpredictable network conditions.

1.6 Thesis Objectives

The objective of this thesis is to investigate resource optimization in the UAV-MEC environment by leveraging the spare computation capacity available on MEC servers. The primary goal is to enhance network performance by offloading and processing a portion of the training dataset from resource-constrained IoT devices to UAV-MECs servers, thereby reducing system-wide delay during FL training. In this work, we formulate an optimization problem that considers both communication and computation constraints to efficiently utilize resources and minimize overall latency.

We consider a UAFL environment consisting of K number of IoT devices and Q number of UAV-MEC servers. The *k-th* IoT device has its dataset C_k that consists of input-output pairs (x_i, y_i), where x_i are input features, while y_i are output labels. The *k-th* IoT device has a convolutional neural network (CNN) where it runs stochastic gradient descent [19] on its dataset using loss function $f(\omega_k) = loss(x_i, y_i; \omega_k)$ to generate model parameters ω_k. Different straggling IoT devices offload their datasets based on their computation requirements to the *q-th* UAV-MEC server which combines and processes all offloaded datasets on its CNN. Then, it sends these computed model parameters ω_q^U to the global FL model at the primary aggregation UAV-MEC server to incorporate the effect of offloaded data on the global FL neural network.

- **Objective-1**: We propose a UAV-aided FL framework (UAFL) aimed at minimizing system latency by optimizing the communication and computation power of IoT devices, alongside optimizing computation power for the UAV-MEC server and offloading dataset sizes for various IoT devices to reduce computation and communication delays. Additionally, we introduce device importance into the UAFL framework and formulate an optimization problem to minimize delay while considering power, offloading, and QoS constraints. To solve this problem, we employ deterministic concurrent simplex with root relaxation (traditional optimization) to improve system delay, sup-

plemented by a reinforcement learning-based solution to enhance runtime complexity. Simulation results demonstrate the performance of the proposed UAFL framework and reinforcement learning-based solution compared to traditional FL [20] and edge-based learning systems [21], with UAFL demonstrating superior performance. Furthermore, we provide simulation results to investigate the impact of device importance and device dropout on system performance.

- **Objective 2**: We propose employing multiple UAVs, rather than a single one, to extend coverage over a larger area containing IoT devices for FL. However, this introduces a challenge concerning consistency, as the utilization of multiple UAVs may compromise model optimization. To address this concern, we propose implementing a hierarchical structure among the UAVs, designating one UAV as the leader UAV responsible for aggregating updates from follower UAVs. Each follower UAV collects model weights from its associated devices and forwards them to the leader UAV. Subsequently, the leader UAV aggregates these weights from follower UAVs and disseminates global weights back to each follower UAV which then distributes these weights to all connected IoT devices.

- **Objective-3**: We propose enhancing the performance of UAFL by integrating Gen-AI to refine and balance our dataset. Within our system, the collected dataset becomes imbalanced because not all IoT devices receive data samples from every class, resulting in certain classes becoming underrepresented in the aggregated dataset. We utilize a Gen-AI-based approach to improve the dataset with additional samples and achieve a more balanced distribution across its various classes. This mechanism improves system performance in two ways. Firstly, it enhances model accuracy for resource optimization by leveraging a more balanced dataset during training. Secondly, it enables the utilization of other machine learning models for resource optimization that were previously impractical due to their dependency on large volumes of data.

1.7 Thesis Contributions

This dissertation addresses these gaps with four major contributions:

- **Contribution 1:** We provide a comprehensive survey on machine learning-driven resource optimization techniques for future wireless networks, with particular emphasis on emerging UAV-assisted, FL-enabled, and edge-computing architectures. The survey systematically reviews state-of-the-art works in computation offloading, energy optimization, and intelligent resource scheduling, highlighting their applicability, limitations, and future potential.

- **Contribution 2:** We present a comprehensive computation offloading and delay minimization framework for UAV-assisted FL, specifically designed to mitigate the straggling effect in heterogeneous IoT environments. The framework leverages the excess computation capacity available at UAVs-MECs servers to process a portion of the training datasets from resource-constrained IoT devices that would otherwise slow down global model convergence. Our formulation incorporates both communication and computation constraints, ensuring efficient task distribution between local processing and UAV-assisted computation. By optimizing computation power, communication power, and dataset offloading ratios, we propose a delay-sensitive FL mechanism to reduce system-wide latency.

- **Contribution 3:** We develop a queue-based hierarchical FL architecture that integrates a multi-tier network of UAVs-MEC servers with stochastic data arrivals from underlying IoT devices. This architecture models the dynamics of data generation, processing, and offloading using multiple interconnected queues at the IoT, follower UAV, and leader UAV. We employ a hierarchical FL structure, with follower UAVs performing local aggregation and a leader UAV conducting global updates. The proposed framework ensures scalability, improved synchronization, and enhanced robustness in

ever-changing incoming data flows. This proposed architecture demonstrates that hierarchical FL combined with queue-based data scheduling can further optimize delay and performance in large-scale UAV-assisted edge networks.

- **Contribution 4:** We propose a GAN-enhanced hierarchical FL framework that improves learning accuracy in highly Non-IID and imbalanced dataset environments while simultaneously optimizing system iteration delay. By employing GAN, we add generated samples from underrepresented classes, thereby balancing the dataset and reducing the adverse effects of skewed data distributions. The GAN is trained on the follower UAVs-MEC and leader UAV-MEC using the datasets offloaded form IoT devices to follower UAV-MECs. This augmented dataset contributes to more accurate local updates sent to the leader UAV. This contribution demonstrates how augmenting hierarchical FL with intelligent data generation enhances robustness, accelerates convergence, and improves the reliability of model updates, especially in settings characterized by rare events and sparse observations.

1.8 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 – Background and State of the Art: Provides essential background and a focused review of related research on machine learning-based resource optimization in next-generation wireless networks. This chapter examines the most common objective functions, system constraints, and machine learning algorithms applied to resource allocation problems. Furthermore, it categorizes optimization strategies into comprehensive, partial, and environment-only solutions, and concludes with open research challenges particularly relevant to UAV-assisted and FL-enabled edge intelligence.

Chapter 3 – UAV-Aided Edge FL: System Model, Optimization, and Performance Evaluation: This chapter presents the proposed UAFL framework. It intro-

duces the system model where straggling IoT devices offload portions of their datasets to UAVs-MECs servers for reduced computation delay. The chapter formulates the joint communication–computation optimization problem incorporating UAV-MEC computation power, device communication budgets, and QoS constraints. Both deterministic optimization, using concurrent simplex with root relaxation, and reinforcement learning-based solutions are developed to minimize end-to-end system delay. The chapter also provides a performance evaluation, comparing the proposed UAFL scheme with standard FL and edge-learning baselines. Results further analyze the effect of device selection, device importance, training efficiency and convergence.

Chapter 4 – Hierarchical UAV-Assisted FL Architecture: This chapter introduces a queue-aware hierarchical FL architecture incorporating multi-tier UAVs-MEC servers. The proposed architecture comprises of IoT devices, follower UAVs responsible for local aggregation and a leader UAV responsible for global aggregation, enabling FL across wider geographical areas. We detail hierarchical communication and aggregation, queues based on stochastic data arrivals, and analyze system stability. Performance evaluation demonstrates improvements in scalability, synchronization efficiency, and training delay compared to single-UAV systems by utilizing Lyponov optimization framework. The simulation results show the effects of UAV topology, queue dynamics, and varying arrival rates on global model consistency.

Chapter 5 – GANs Enhanced Hierarchical FL: This chapter proposes a GAN-enhanced data augmentation approach integrated into the hierarchical UAFL system. The proposed scheme addresses dataset imbalance and the Non-IID nature of data generated by distributed IoT devices. We present the system model, the augmented data utilization, and mechanisms to ensure stability across hierarchical UAVs using Lyapunov optimization. Simulation results show improvements in model accuracy, robustness to device heterogeneity, and resilience to rare failure patterns. Additional analysis evaluates the role of augmented data in improving accuracy and ensuring global model convergence under given conditions.

Chapter 6 – Conclusion: Summarizes the research contributions of this dissertation, reflects on key findings across the proposed frameworks, and presents several advanced research directions for future works.

Finally, references are included at the end of the thesis.

1.9 Chapter Summary

This chapter introduced the motivation, challenges, and context behind enabling efficient, scalable, and accurate distributed learning in UAV-assisted wireless networks. It highlighted the limitations of centralized learning, the potential of FL, and the need for advanced architectures such as HFL and queue-aware learning frameworks. The chapter also discussed the critical role of GANs in addressing Non-IID data challenges and summarized how machine learning-based resource optimization frameworks contribute to next-generation wireless systems. The chapter identified four core research gaps and outlined the main contributions of this dissertation, which collectively address delay minimization, hierarchical learning, stochastic data arrival modeling, and accuracy enhancement in distributed training. The next chapter builds on this foundation by presenting a detailed review of background concepts and related research.

Chapter 2

Background and State of the Art

2.1 Why Resource Optimization is Important

Resource optimization lies at the core of modern and future wireless communication systems. As the number of IoT devices continues to grow exponentially, networks face unprecedented strain in terms of bandwidth demand, traffic density, and service heterogeneity. Applications in the IoT, autonomous navigation, augmented and virtual reality, real-time sensing, and large-scale artificial intelligence (AI) deployments all require networks to deliver high-speed, ultra-reliable, and low-latency connectivity. This increase in data generation and device density has pushed traditional wireless systems toward their operational limits.

Future networks, particularly those envisioned for beyond-5G and 6G infrastructures, must accommodate a broad spectrum of use cases spanning terrestrial, aerial, and underwater environments [22]. These environments impose highly diverse performance requirements, ranging from stringent latency demands to massive connectivity and energy efficiency. As traffic volumes surge and service expectations rise, the efficient use of limited resources becomes non-negotiable. Spectrum, bandwidth, power, processing capabilities, and backhaul capacity are all fundamentally constrained, and must therefore be allocated intelligently to ensure stable and reliable system performance.

Resource optimization plays a critical role in achieving this goal. Efficient spectrum utilization techniques such as dynamic spectrum access, cognitive radio, and spectrum sharing help alleviate congestion and interference by adaptively allocating frequencies according to real-time traffic conditions [23]. Likewise, bandwidth management strategies, including multiplexing, beamforming, and network slicing, enable networks to serve diverse applications with varying throughput and latency requirements. These strategies enhance spectral efficiency and ensure that high-bandwidth services, such as immersive media or cloud-based computation, receive sufficient resources [24]. Energy efficiency is also vital in wireless networks, especially as systems move toward greener communication models. Minimizing energy use is necessary to reduce operational costs and to support IoT devices powered by batteries or energy-harvesting mechanisms. Techniques such as energy-aware routing, adaptive transmission control, and predictive scheduling help extend device lifetimes and maintain consistent network performance in large-scale deployments.

Ultimately, effective resource optimization contributes directly to network reliability, scalability, and user quality of experience (QoE). By dynamically managing limited resources and mitigating congestion, networks can maintain high service quality even under peak demand [25–27]. This is particularly vital for mission-critical applications, including healthcare, transportation, emergency response, and industrial automation, where delays or communication failures can lead to severe consequences. In summary, resource optimization forms the foundation upon which next-generation wireless systems must be built. Without intelligent and adaptive resource management, the ambitious goals of 6G and future IoT-driven ecosystems cannot be realized.

2.2 Resource Optimization in Wireless Networks

Resource optimization has become a central requirement in modern wireless networks due to the rapid increase in connected IoT devices, heterogeneous traffic patterns, and stringent

performance demands. As next-generation networks expand to support diverse applications, ranging from enhanced broadband to massive IoT and ultra-reliable low-latency communication, efficient management of spectrum, power, bandwidth, and compute resources is essential. The surveys in [23, 28, 29] highlight that wireless networks inherently operate over constrained and interference-limited media, making optimal resource allocation fundamental for ensuring reliable connectivity.

Spectrum scarcity remains a major challenge, particularly as large numbers of IoT devices compete for limited bandwidth. ML–based spectrum management and adaptive allocation approaches surveyed in [23] demonstrate that dynamic spectrum access and interference-aware scheduling are crucial for maximizing spectral efficiency. Traditional rule-based methods often fall short in such environments, motivating the need for more intelligent, context-aware optimization strategies.

Wireless networks also face highly dynamic channel variations caused by mobility, multipath fading, and environmental fluctuations. These variations directly influence signal quality and achievable data rates, demanding continuous adaptation of transmission power, channel assignment, and routing decisions. The authors of [30] emphasized that AI-enabled radio management plays a significant role in addressing these fluctuations by enabling predictive interference mitigation, adaptive link control, and intelligent traffic steering.

Energy efficiency is another critical aspect of wireless resource optimization. The work in [29] emphasized the need for energy-aware communication, particularly in dense 5G and beyond systems where power consumption at base stations and IoT devices significantly affects overall network sustainability. Similarly, the surveys [31, 32] highlighted resource management frameworks that prioritize energy-centric allocation to prolong device lifetime and support green communication objectives in 6G networks.

Latency-sensitive applications such as industrial automation, remote monitoring, and vehicular networks require stringent delay guarantees. The survey on ultra reliable low latency communication (URLLC) in 6G networks [25] demonstrated that both supervised and

reinforcement learning approaches can be used to optimize delay and reliability under tight latency constraints. These methods enable networks to proactively manage resources based on predicted traffic surges or channel degradation, thereby maintaining desirable service quality.

Furthermore, the complexity of future wireless architectures, incorporating fog devices, edge servers, UAVs, and virtualized network functions, adds new dimensions to resource optimization. Studies in [33] and [34] show that dynamic task scheduling, load balancing, and joint communication–computation resource allocation are essential to prevent bottlenecks in fog and edge-enabled networks. These systems require coordinated management across heterogeneous layers, often involving distributed decision-making and multi-objective optimization.

Finally, wireless networks must ensure fairness and QoE across diverse user categories. As noted in [35], multi-dimensional optimization frameworks involving communication, computation, caching, and control (4C) are needed to balance competing requirements in large-scale systems. Likewise, [27] emphasized that UAV-assisted networks introduce additional constraints, such as flight energy and aerial link variability, that must be jointly optimized with conventional radio resources.

In summary, resource optimization in wireless networks is driven by the scarcity of spectrum, dynamic channel conditions, increasing energy demands, latency-sensitive services, and architectural complexity. Insights from recent surveys make it clear that the growing scale and diversity of wireless ecosystems require holistic, intelligent, and adaptive resource management strategies capable of sustaining performance in next-generation communication systems.

2.3 Optimization Within Machine Learning Environment

This category covers works where the researchers took a machine learning environment in wireless networks and tried to optimize some aspects of this machine learning environment using traditional optimization approaches. The most common use cases are based on FL environments where the authors work to optimize the system's performance by optimizing FL parameters (e.g., local and global iterations) [36–39]. These studies provide useful baselines and design patterns for how classical optimization tools can be embedded into learning-centric pipelines deployed over wireless networks.

2.3.1 Energy Minimization

This subsection covers machine learning-based studies that incorporate power and energy considerations in their objective functions. The selected works focus on system-wide energy consumption [40, 41] and related energy-efficiency formulations [42]. These works are particularly relevant for FL in wireless settings, where communication and computation costs at IoT devices or edge devices must be balanced against convergence requirements.

In [40], the authors introduced a novel hierarchical federated edge learning framework and formulated a computation and communication resource allocation problem. They aimed to minimize the overall energy and delay for each global iteration. They imposed link and task-specific (FL training) constraints. The formulated problem was non-convex, so they decomposed the scheduling algorithm into two sub-problems. The first sub-problem focused on resource allocation in a single-edge server, which they addressed using the Lagrange multiplier method. For the second sub-problem they proposed an algorithm that constantly adjusts the edge association strategy by changing group formation to minimize overhead. They performed an appropriate comparison of their work with an optimal traditional baseline algorithm to demonstrate the effectiveness of their approach.

In [41], the authors focused on resource allocation for machine learning tasks at the edge in wireless networks. They introduced an edge machine learning framework comprising various algorithms to facilitate resource allocation. They formulated the problem as minimizing average long-term optimization task to balance energy consumption, learning accuracy, and latency. The objective was to minimize the expected energy consumption value, subject to data rate, delay, task-specific (FL accuracy), and compute resource utilization (CPU bounds) constraints. To solve this problem, the authors proposed a set of Lyapunov optimization-based algorithms. The authors provided theoretical and numerical results with relevant comparisons, demonstrating the impact of their proposed approach.

In [42], the authors addressed the service placement and resource allocation problem in multi-user MEC scenarios, focusing on optimizing resource allocation and AI service placement. The latter involves determining whether placing the AI model at the edge improves performance. They formulated this as a mixed-integer nonlinear programming (MINLP) problem, aiming to optimize the total energy consumption and task computation time across all users with respect to data rate, link, and compute resource utilization (CPU frequency bounds) constraints. Initially, they proposed separate solutions for resource allocation and service placement: an ellipsoid method for resource allocation and a greedy search algorithm for service placement. They introduced a joint allocation scheme using the alternating direction method of multipliers (ADMM). Utilizing a bisection search algorithm, this approach achieved a linear time complexity of K^1, offering improved efficiency. The authors conducted performance comparisons of their proposed approach with ADMM, greedy, and optimal traditional optimization baselines.

In [43], the authors addressed energy consumption minimization and energy harvesting maximization in a FL setting with heterogeneous IoT devices. They formulated this as an optimization problem aiming to minimize energy consumption for training one iteration, subject to constraints related to subchannel and transmission power, data size, and task scheduling. Using the Lagrangian dual method, the authors decoupled the variables and

proposed solutions for processor frequency and dataset size allocation. Additionally, they optimized transmission power using an iterative algorithm and employed a closed-form solution for time splitting. They conducted a comparative analysis of their proposed approaches with variations of their algorithms. To enhance their work, authors could have incorporated an optimization benchmark algorithm (e.g., branch and bound (B&B)) to provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.

Lessons learned: We observed the generic issue of providing an appropriate benchmark in this section. In [40], [41], [42], the authors provided an appropriate benchmark comparison while [43] could have incorporated an optimization benchmark algorithm (e.g., B&B) to provide a standard reference point for evaluating the performance and effectiveness of the proposed methods. From a thesis perspective, these works collectively illustrate that even when the networking stack is fixed, careful optimization of learning-related variables (e.g., frequency scaling, dataset partitioning, and iteration scheduling) can yield significant energy savings without sacrificing model performance. Table 2.1 summarizes all the works discussed in the section above.

2.3.2 Latency Minimization

Time is an important metric in optimization problems. We covered those works here that formulate time-based objective functions (latency, delay, time) in a machine learning environment. In this subsection, we consider delay [37, 44], followed by latency [45] and time (other representations) [36] works. These studies highlight how temporal performance metrics, such as end-to-end delay or service latency, can be explicitly embedded into optimization formulations for learning systems running over wireless infrastructures.

In [37], the authors addressed optimization in a FL environment tailored for cognitive IoT applications in smart industries. They aimed to enhance system performance by minimizing computation delay and model error rate, treating the problem as an integer linear programming problem. Their objective function focused on minimizing the overall cost,

Table 2.1: Summary of energy-based publications for machine learning environment-only solutions,
ADMM: alternating direction method of multipliers; B&B: branch and bound; FL: federated learning.

Refer	Objectives	Constraint type	Proble type	Classica approach	Remarks
[40]	Minimize energy and delay	Link, task-specific (FL training)	Non-convex	Lagrange multiplier method	The authors performed an appropriate comparison with an optimal traditional baseline algorithm.
[41]	Minimize energy consumption	Data rate, delay, task-specific (FL accuracy), compute resource utilization (CPU bounds)	Non-convex	Lyapunov optimization	The authors provided theoretical and numerical results with relevant comparisons.
[42]	Minimize total energy and task computation time	Data rate, link, compute resource utilization (CPU frequency bounds)	-	ADMM	The authors compared their work with ADMM, greedy, and optimal baselines.
[43]	Minimize the energy consumption for one iteration	Transmission power, delay, compute resource utilization (CPU time, CPU frequency, data storage)	-	-	The authors compared results with variations of their own work. To enhance the study, an optimization benchmark algorithm (e.g. B&B) could have been incorporated. This would provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.

which comprised computation delay and global error rate with multiple constraints related to network links. Demonstrating the convexity of the problem, they utilized standard convex optimization methods, such as in CVXPY [46], for its solution. They also addressed resource block allocation, relaxing constraints to render the problem convex, and employed standard convex toolkit methods to solve it. The authors compared results of their proposed approach with variations of their own works. To enhance their work, the authors could have incorporated an optimization benchmark algorithm (e.g., B&B) to provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.

In their work, [44], the authors addressed optimizing an air-ground integrated wireless network with a focus on minimizing service latency. Their problem formulation focused on model decision, computation resource allocation, and UAV trajectory control with constraints related to compute resource utilization, power, and communication latency. Given the non-convex nature of the formulated problem MINLP, they decomposed it into three sub-problems and converted each into a convex sub-problem. Then, they proposed an iterative algorithm to optimize each sub-problem using traditional optimization methods like B&B or the Lagrange duality method. Notably, machine learning was solely involved in deep neural network (DNN) model selection, where each vehicle had different AI models, and the selection process utilized traditional optimization techniques. The study utilized a relatively small simulation size (six IoT devices and one UAV), which may impact the generalizability of the results.

In their work, [45], the authors addressed the issue of DNN partitioning and offloading to resource-constrained edge devices. They focused on a DNN model requiring significant computational power for training. To facilitate computation on edge devices, they partitioned the model into smaller chunks for distribution to various edge devices. The authors formulated their resource allocation problem to minimize the maximum latency of all IoT devices, employing a min-max approach with link and compute resource utilization (CPU bound) constraints. They addressed it as a multi-choice knapsack problem (nondeterministic

polynomial time (NP)-hard). They proposed an iterative alternating algorithm that iteratively optimized by reallocating computation resources with minimal utility. Furthermore, they enhanced their algorithm by introducing a variable time step to improve efficiency. The study utilized a relatively small simulation size (four IoT devices and one server), which may impact the generalizability of the results.

In [36], the authors solved the challenges of deploying UAVs as base stations (BSs) in a FL environment, where the UAVs had limited battery capacity, and the attached IoT devices varied in terms of battery and computation power. They developed an asynchronous FL framework, allowing networks to be trained locally without transmitting the entire dataset to the UAV BS. Local updates were computed asynchronously on high-computation IoT devices. The authors further formulated the problem of UAV placement, device selection, and resource management, aiming to minimize model execution time and learning accuracy loss over time, considering constraints related to link, transmission power, QoS, and compute resource utilization (CPU bounds). The formulated problem was non-convex and NP-hard. To address these challenges, the authors proposed an asynchronous FL framework utilizing asynchronous advantage actor-critic deep reinforcement learning (DRL). They proposed two algorithms based on reinforcement learning: the first algorithm for UAV placement, decision selection, and resource management, and the second for asynchronous FL. The authors compared their work against the gradient-based benchmarks to demonstrate the efficacy of their proposed approach.

Lessons learned: In [36], the authors compare their method against a standard gradient-based benchmark, providing a clearer baseline for evaluation. In contrast, [37] evaluates performance only against variations of its own proposed algorithm, which limits the breadth of comparison. Furthermore, the simulation settings used in [44] and [45] could be expanded—such as increasing the number of participating devices (e.g., beyond six IoT devices and one UAV, or four IoT with one server), to strengthen the robustness and generalizability of their results. Overall, the latency-focused studies reviewed here show that

objective functions centered on time (rather than purely on energy or accuracy) often lead to multi-stage formulations combining placement, offloading, and scheduling decisions. This perspective will be particularly relevant when analyzing delay in UAV-assisted FL frameworks later in this thesis.

We studied recent available literature on time-based objectives for resource optimization within machine learning environments. The discussion highlights how different authors translate application-level delay requirements into concrete optimization problems over learning and system parameters. The upcoming section discusses works with machine learning arguments-based objectives for resource optimization. Table 2.2 summarises all the works discussed in the above section.

2.3.3 Parameter Optimization In Machine Learning Based Methods

This subsection reviews recent works of authors who focused on the machine learning parameters as objective functions. These broadly include loss [38, 47], training time [39, 48], and cost [49] among others [50,51]. Unlike the previous subsections, which emphasize energy or latency directly, the works discussed here treat performance indicators intrinsic to the learning algorithm (e.g., training loss, classification error, or divergence measures) as the primary optimization objective.

In [47], the authors addressed the challenge of processing machine-type communication data volumes on edge devices constrained by limited power allocation typically optimized for communication throughput rather than learning throughput. Traditional algorithms like water-filling and min-max fairness proved inefficient in this context. The authors aimed to minimize classification error while considering total power, data rate, and sample collection. They developed a learning-centric power allocation algorithm variant, formulating it as a non-convex, non-smooth optimization problem. Their solution leveraged a majorization-minimization framework, constructing upper bounds on the empirical classification error

Table 2.2: Summary of latency-based works for machine learning environment-only solutions,
B&B: branch and bound; FL: federated learning; IoT: Internet of things; MINLP: mixed-integer nonlinear programming; QoS: quality of service; UAV: uncrewed aerial vehicle; UE: user equipment.

Refe	Objectives	Constraint type	Proble type	Classic approach	Remarks
[37]	Minimize delay and global error rate	Link	Integer programming	Convex toolkit	The authors compared results of their proposed approach with results of variations of their own work. To enhance the study, an optimization benchmark algorithm (e.g., B&B) could have been incorporated. This would provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.
[44]	Minimize service latency	Link, task-specific (FL accuracy, UAV velocity, UAV position), compute specific constraints (UAV CPU bounds, UE cpu bounds.)	MINLP	B&B	The study utilized a relatively small simulation size (six IoT IoT devices and one UAV), which may impact the generalizability of the results.
[45]	Minimize maximum latency for all IoT deviecs	Link, compute resource utilization (CPU bound)	-	Iterative alternating algorithm	The study utilized a relatively small simulation size (four IoT devices and one server), which may affect the generalizability of the results.
[36]	Minimize machine learning model execution time and model training loss	Link, transmission power, QoS, compute resource utilization (CPU bounds)	-	-	The authors compared their work against gradient-based benchmarks.

model and replacing the problem with surrogate problems. They framed their objective as minimizing the maximum classification error with power budget and data rate constraints. The authors derived the optimal solution through gradient descent or brute force search, further refining parameters for generalization error using convolutional neural networks and support vector machine (SVM). The authors compared their results with an optimal traditional baseline demonstrating the efficacy of approach proposed.

In [38], the authors worked on FL, employing classical optimization techniques to optimize FL accuracy and training trade-offs in MEC networks. They formulated the problem as a minimization task, aiming to minimize the combination of training loss and model training cost in the FL framework. Their formulation encompassed three constraints related to energy consumption, number of users, and dataset size. Given the nonlinear and non-convex nature of the problem, they initially decomposed it into three independent sub-problems: accuracy loss, communication cost, and joint optimization. Subsequently, they devised an iterative algorithm (non-machine learning) to solve each derived sub-problem iteratively. The authors compared the results with variations of their own work. To enhance their work, the authors could have incorporated an optimization benchmark algorithm (e.g., B&B) to provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.

In [39], the authors proposed an innovative FL algorithm designed to manage heterogeneous datasets across diverse IoT devices in wireless networks. They assumed that the loss function at each IoT was smooth and non-convex. By employing a time-sharing multiaccess protocol for communication standards, their algorithm achieves the tradeoff between convergence time and energy consumption across IoT devices with varying power capacities. Their optimization problem aimed to minimize training time and energy consumption under constraints such as a time-sharing uplink transmission limit, computing time, and local accuracy. To solve this wireless domain problem, they divided it into three sub-problems, with the first two converted into convex problems, leaving the third as non-convex. They

solved these sub-problems using traditional optimization algorithms to achieve the overall objective. A major limitation of their approach was its assumption that local processing must be completed before initiating communication between IoT devices for global training. The authors considered many IoT devices in their simulations and compared their work with optimal solutions.

In [48], the authors minimized the energy consumption of a UAV swarm by optimizing various parameters in a FL environment. Their optimization targets included convergence threshold, local and global iterations, bandwidth, and computation resources. The objective function aimed to optimize the overall energy consumption, integrating constraints related to local convergence thresholds, local iteration counts, computation resource allocations, bandwidth, and convergence thresholds. They approached this complex problem by segmenting it into three sub-problems, addressing them directly and via a novel iterative algorithm through successive convex optimization. Furthermore, they investigated fairness in joint optimization, introducing a new variable in the objective function and addressing this revised problem with a new iterative algorithm proposal. The authors compared their work with a standard benchmark to demonstrate the superiority of their proposed approach.

The authors of [49] explored the Internet of vehicles characterized by multiple highly mobile vehicles. They initially developed an algorithm to determine the participation of specific vehicles in FL. Subsequently, they focused on the optimization problem of resource allocation to reduce the cost of FL for participating vehicles. They considered this problem as the minimization of FL costs (comprising of participation in the current round and waiting time for the current round to conclude). They formulated the problem subject to link, QoS, and transmission power constraints. Their proposed solution involved an algorithm for decision-making regarding participation in FL, which was input into a multi-agent DRL algorithm responsible for training and resource allocation. They developed an algorithm outlining the allocation procedure for multi-agent deep deterministic policy gradient (DDPG) based joint resource allocation, encapsulating their comprehensive approach to FL optimization in In-

ternet of vehicless (IoVs). The authors compared their work with a standard benchmark to show the impact of their proposed approach.

In [50], the authors addressed the formulation of a resource allocation and scheduling problem in wireless networks. Their problem formulation aimed at minimizing a combination of the probability of selecting a helper as the FL gradient uploader, the channel assignment indicator, and the number of bits allocated by each helper for subchannels. They comprehensively addressed the communication dynamics with data rate, and task-specific (helper selection probabilities) constraints. Transforming the problem into a MINLP via the transformation method, they reduced the complexity in their formulation. The equivalent problem was then decomposed into two sub-problems. First, they addressed the resource allocation sub-problem using the B&B method, and then with their proposed algorithm providing a sub-optimal solution with reduced complexity. Second, they employed the penalty convex-concave procedure (PCCP) technique [52] for helper scheduling, leveraging the interior point method to find efficient solutions. This multi-faceted approach enabled them to address the intricacies of resource allocation and scheduling in their FL framework. The authors compared their work with a standard benchmark (i.e., B&B) to demonstrate the efficacy of their proposed approach.

In [51], the authors operated in a wireless FL environment and their goal was to address the training time acceleration problem. They introduced a novel metric, "learning efficiency," defined as the ratio of global loss delay to end-to-end efficiency. The objective function was formulated to maximize this learning efficiency. Their problem included constraints related to link and compute resource utilization (FL round data bounds, batch size bounds). They subsequently divided the problem into two separate sub-problems. The first subproblem aimed to solve the local gradient calculation, uploading, and formulation, framed as a min-max optimization problem. They then transformed this problem into a convex one and resolved it using fractional optimization. They proposed a two-dimensional search algorithm as a solution to this sub-problem. For the second subproblem, they solved the

issue of downloading the global gradient and updated the local model by applying Karush-Kuhn-Tucker conditions. The study utilized a relatively small simulation size (with twelve IoT devices), which may impact the generalizability of the results.

In [53], the authors operated in a hierarchical FL environment with heterogeneous IoT devices. They proposed resource allocation and user assignment while considering the user data distributions and wireless communication characteristics. They utilized the Kullback–Leibler divergence (KLD) to address the data imbalance problem. The objective here was to optimally allocate end users to edge devices while minimizing the KLD. They formulated the problem with different constraints related to link and system power. The formulated problem is an integer programming NP-hard problem. The authors initially transformed this problem into a linear programming problem. They then determined the linear programming solution and optimized the bandwidth allocation for available end users, ensuring KLD minimization in the process. They compared their work with state-of-the-art benchmarks and demonstrated its superiority.

Lessons learned: The authors in [47], [48], [49], [50], [53] compared their work with the standard benchmark. In [38], the authors compared their results with variations of their own work. To enhance their work, the authors could have incorporated an optimization benchmark algorithm (e.g., B&B) to provide a standard reference point for evaluating the performance and effectiveness of the proposed methods. In [39], the authors compared their work with an optimal baseline and used a reasonably large simulation size (one hundred IoT devices). In contrast,the authors in [51] used a small simulation size with twelve IoT devices only. Taken together, these works demonstrate a rich design space where learning performance metrics (e.g., loss, classification error, or distributional divergence) are explicitly optimized under communication and computation constraints. This line of research is closely aligned with the goals of this dissertation, which also seeks to jointly reason about learning quality and system-level costs in UAV-assisted FL architectures. Table 2.3 summarizes all the works discussed in the section above.

Table 2.3: Parameter optimization in machine learning based methods for machine learning environment-only solutions
B&B: Branch and bound; FL: Federated learning; KLD: Kullback–Leibler divergence; QoS: Quality of service; UAV: Uncrewed aerial vehicle; UE: User equipment.

Refer	Objectives	Constraint type	Proble type	Classica Ap- proach	Remarks
[47]	Minimize classification error	Data rate, system power	Non-smooth and non-convex	Majoritization-minimization framework	The authors performed an appropriate comparison with an optimal traditional baseline.
[38]	Minimize training loss and training cost	Link, compute resource utilization (data size constraint), system power	Nonlinear and non-convex	Iterative algorithm	The authors compared the results of their proposed work with variations of their own work. An optimization benchmark algorithm (e.g., B&B) could have been incorporated to enhance the study. This would provide a standard reference point for evaluating the performance and effectiveness of the proposed methods.
[39]	Minimize training time and energy consumption	Link, delay, system power, computation task scheduling (CPU frequency)	Non-convex	Traditional optimization algorithms	An appropriate comparison with an optimal baseline solution (standard FL) is done using a simulation setup comprising a hundred IoT devices. While this setup may be considered limited in scale, it demonstrates the feasibility and relative performance of the proposed method in a controlled environment.

Continued on next page

Table-continued from previous page					
Refer	**Objectives**	**Constraint type**	**Proble type**	**Classic approach**	**Remarks**
[48]	Minimize UAV's overall training energy consumption	Link, task-specific (FL iterations for convergence, UAV CPU bounds, QoS	Convex	Successive convex optimization	The authors compared their work with a traditional optimization (i.e., B&B) benchmark.
[49]	Maximize FL delay imbalance between IoT devices	Link, transmission power, QoS, compute resource utilization	-	Proposed algorithm	The authors compared their work with a traditional optimization (i.e., B&B) benchmark.
[50]	Minimize probability of selection of FL helper	Data rate, task-specific (helper selection probability)	Non-convex	Learning-centric power allocation	The authors compared their work with a traditional optimization (i.e., B&B) benchmark.
[51]	Maximize learning efficiency	Link, compute resource utilization (FL round data bounds, batch size bounds)	Non-convex	Fractional optimization method	The study utilized a relatively small simulation size (with twelve IoT devices), which may impact the generalizability of the results.
[53]	Minimize KLD edge devices)	Link, system power	Non-convex	Kullback-Liebler divergence KLD	The authors compared their work with state-of-the-art benchmarks and demonstrated its superiority.

2.4 Resource Optimization in UAV-Assisted FL-Based Wireless Networks

Multiple works have investigated different aspects of cooperative learning schemes, including FL, to enhance system performance in terms of delay, energy consumption, and learning accuracy. A significant portion of this literature focuses on mitigating the impact of straggling IoT devices, either through algorithmic modifications to the learning protocol or through

better utilization of communication and computation resources. In the context of this dissertation, these works provide important building blocks and baselines for understanding how delay and energy can be optimized in UAV-assisted FL systems.

2.4.1 Straggler Mitigation in Cooperative and Federated Learning

Early works on cooperative and distributed learning explored straggler mitigation without explicitly targeting UAV-assisted architectures. For example, the authors of [7] adopted a collaborative learning environment and proposed a distributed computing scheme based on a coupon collection problem. Their design achieved near-minimal average recovery thresholds and communication loads, effectively mitigating the straggler effect by exploiting coded redundancy in the distributed computations.

While these early distributed learning approaches effectively address straggler mitigation through coded computation and redundancy, they typically assume centralized access to data or require extensive data transfer across the network. Such assumptions are increasingly impractical in large-scale IoT systems, where raw data is often privacy-sensitive, bandwidth-constrained, and geographically distributed. FL emerges as a natural paradigm in this context, as it enables collaborative model training while keeping data localized at IoT devices, thereby preserving privacy and reducing communication overhead. However, the decentralized and heterogeneous nature of FL introduces new challenges, particularly in the presence of straggling and intermittently connected IoT devices, motivating dedicated straggler-aware designs within the FL framework.

Within FL specifically, several works have examined asynchronous aggregation as a way to reduce the impact of slow IoT devices. In [54], an asynchronous online FL framework was introduced where the central server continually aggregated updates from IoT devices that perform online learning on continuous local data streams. Similarly, [55] proposed an asynchronous FL algorithm that introduced a scheduler between the server and IoT devices, responsible for managing global model updates and disseminating the most recent

global weights to participating IoT devices. In both cases, the straggler effect is mitigated by decoupling global aggregation from the slowest IoT devices, at the cost of additional protocol complexity and potential staleness in updates.

Coded computing has also been used to address stragglers effect in distributed learning. In [56], the authors proposed a coded computing framework that reformulated the training task as distributed linear regression and generates local parity datasets at each client. These parity datasets are then aggregated at the server to reconstruct gradients corresponding to straggling IoT devices. The idea was extended in [57], where a hierarchical coded computing framework introduced helper IoT devices between the aggregation server and IoT devices. This framework leverages aligned repetition coding and minimum distance separable (MDS) coding to improve gradient aggregation efficiency and robustness against stragglers.

Overall, these works demonstrate that straggler mitigation can be approached via asynchronous aggregation, coded redundancy, or hierarchical structures. However, most of them either assume static infrastructures or do not explicitly model the mobility, energy, and computational constraints of UAV-assisted edge systems.

2.4.2 MEC-Assisted FL and Device Selection in Wireless Networks

To further mitigate the straggler effect and improve performance, researchers have explored exploiting the computation power available at MEC servers. The concept of MEC was formally introduced in [58], and subsequent works have applied task offloading and resource allocation strategies in MEC environments. For instance, [59] designed task offloading schemes under time-sharing and computation deadline constraints to minimize the total energy consumption of IoT devices.

When FL is deployed at the network edge, system delay and energy become tightly coupled. In [60], the authors optimized caching and communication within FL frameworks operating over MEC servers using deep reinforcement learning to adapt policies over time.

Similarly, [61] proposed training and deploying deep reinforcement learning agents on IoT devices, with the goal of reducing transmission costs and improving delay performance through learned decision-making policies.

The importance of device participation and selection has also been emphasized. In [62], the authors adopted FL in an IoVs environment and proposed a fully distributed algorithm inspired by multi-agent deep reinforcement learning to optimize FL cost, accounting for the high mobility and heterogeneous resources of vehicles. Other approaches, such as [63–65], focused explicitly on joint latency and energy optimization in FL environments by tailoring communication, computation, or scheduling decisions to system constraints.

Reduction in system delay has also been studied through prioritization of IoT devices. In [9], a probabilistic device selection framework was proposed, where device selection probabilities are dynamically adjusted based on their contribution, enabling effective aggregation. Related works [10, 66, 67] investigated device selection and resource allocation strategies that aim to improve FL performance by balancing learning accuracy, delay, and resource usage. These methods demonstrate that careful selection of participating IoT devices can substantially improve convergence speed and efficiency, but they may also introduce bias if high-priority IoT devices are systematically preferred.

2.4.3 UAV-Assisted FL and Joint Resource Optimization

A growing line of research explicitly incorporates UAVs into FL architectures to leverage their mobility and 3D deployment flexibility. In [68], the authors proposed a UAV-based framework that integrates aerial and terrestrial networks for FL, considering four collaboration configurations: air-to-ground, ground-to-air, air-to-air, and mixed environments. They provided a case study on ground-to-air FL, focusing on how the hovering location of the UAV affects accuracy and energy consumption. While the system model and formulation were limited, the work highlighted the potential benefits of aerial platforms for expanding coverage and enhancing training performance.

The work in [69] further examined a UAV-based FL framework, formulating a joint optimization of UAV positioning and iteration time to minimize energy consumption while maintaining model performance. Using alternating convex optimization, they showed how the UAV trajectory and communication parameters can be adapted to balance energy and learning objectives. In contrast, [70] proposed an asynchronous FL framework in a UAV-assisted setting, targeting the straggler effect directly by selecting low-latency IoT devices through reinforcement learning. Their weighted-sum formulation jointly optimized device selection, UAV placement, and resource management, with the primary objective of minimizing aggregation time and improving accuracy in asynchronous environments.

Device scheduling with explicit learning-oriented objectives has also been studied in UAV-assisted FL. In [71], the authors focused on selecting IoT devices with high "data importance" to minimize the expected final training loss. Their scheduling strategy aimed to prioritize IoT devices whose data had the greatest potential impact on model performance. Furthermore, [72] employed federated deep reinforcement learning to jointly optimize resource allocation and UAV deployment in order to maximize long-term throughput and QoS. They formulated the problem as maximizing system throughput by adjusting access control, beamwidth, and UAV location, illustrating how UAV mobility and wireless resources can be jointly controlled in a learning-centric environment.

2.4.4 Summary and Positioning of This Dissertation

Table 2.4 summarizes the above discussion. Broadly, the existing literature on resource optimization in FL-based wireless networks provides three main categories of solutions to reduce the straggler effect in IoT applications:

- **Asynchronous global updates:** Works such as [54, 55, 59, 70, 71] decouple global aggregation from the slowest IoT devices by allowing asynchronous updates or online learning. While this reduces waiting time per iteration, it introduces additional protocol complexity and requires careful management of model staleness and synchronization

Table 2.4: Summary of related works to mitigate Straggler's effect.
UAV: U; MEC:M; Synchronous framework: S;FL: Federated learning; IDI: IoT device importance; T-Sol: Traditional Solution Approach; ML-Sol: Machine learning-based solution.

Ref.	U	M	S	FL	IDI	Objective	T-Sol	ML-Sol	Remarks
[7]	✗	✗	✓	✗	✗	Minimize recovery threshold	Batched coupon collection	✗	Proper theoretical bounds provided. Simulation size is good.
[54]	✗	✓	✗	✓	✗	Modified aggregation function of FL	✗	Online async FL algorithm	Asynchronous therefore communication bottleneck.
[55]	✗	✗	✗	✓	✗	Modified aggregation function of FL	✗	Async FL algorithm	Asynchronous therefore communication bottleneck.
[56]	✗	✓	✓	✓	✗	Improve system delay	✗	offloading parity data based aggregation FL algorithm	Changing datasets will cause parity issues.
[59]	✗	✓	✗	✗	✗	Minimize energy consumption	Async arrival order deadline	✗	High runtime complexity.
[60]	✗	✓	✓	✓	✗	Reduce DRL training time	✗	In-edge and IoT device FL training algorithm	Possible issues with Non-IID data.
[61]	✗	✓	✓	✓	✗	Minimize system delay & queue	✗	DRL based solution	Lack of comparison with traditional benchmark.
[62]	✗	✓	✓	✓	✓	Minimize cost per participant	✗	DRL based participants selection	Relatively small simulation size.
[63]	✗	✓	✓	✓	✗	Minimize system delay	✗	DRL based	Proper simulation results with appropriate size.
[64]	✗	✓	✓	✓	✗	Minimize system delay and energy	✗	Hierarchical FL based solution	Increase in delay due to increase in number of layers in FL stack.

Continued on next page

Table-continued from previous page									
Ref.	**U**	**M**	**S**	**FL**	**ID**	**Objective**	**T-Sol**	**ML-Sol**	**Remarks**
[9]	✗	✗	✓	✓	✓	Optimal IoT device selection	✗	Contribution-based and probabilistic FL IoT device selection algorithms	Possible compromise in accuracy of FL due to high percentage of devices drop.
[10]	✗	✓	✓	✓	✓	Minimize offloading ratio and selected IoT devices	✗	FL IoT device sampling algorithm	Authors have utilized testbed data to validate their results.
[68]	✓	✓	✓	✓	✗	Minimize Energy consumption	✗	✗	Limited case study with focus on results with two UAV-deployment schemes provided.
[69]	✓	✓	✓	✓	✗	Minimize Energy consumption	Alternating convex optimization	✗	Focus on optimizing energy consumption using iteration time and UAV location.
[70]	✓	✓	✗	✓	✓	Optimize aggregation time and accuracy	✗	DRL based solution	Focus on model accuracy and model execution time.
[71]	✗	✓	✗	✓	✓	Minimize final training loss expectation	Gradient aware device selection	✗	Focus on model loss by optimizing device selection strategy.
[72]	✗	✗	✓	✗	✗	Maximize long-term throughput	✗	DRL based approach	Optimize throughput by access control, beamwidth and UAV locations.
Our work	✓	✓	✓	✓	✓	Improve system delay and energy consumption	Concurrent simplex with root relaxation approach and client importance based selection algorithm	DRL based approach	

overhead.

- **Device selection and scheduling:** A second class of approaches [9, 10, 62, 66, 70, 71] selects only a subset of IoT devices in each global iteration, often based on importance, channel quality, or cost metrics. These schemes can significantly reduce delay and resource usage; however, if priority is consistently given to certain devices (e.g., those with lower latency), there is a risk of degrading global model accuracy by under-utilizing data from slower or more constrained IoT devices.

- **MEC-assisted computation offloading:** The third line of work exploits spare computation capacity at MEC servers to mitigate the straggler effect [10, 54, 57, 60–64, 68, 69]. Most of these studies assume fixed MEC servers with abundant computing resources and stable connectivity [10, 54, 57, 60–64]. Only a subset explicitly considers UAV-MEC configurations [68–70], and even there, the primary optimization goals often focus on energy consumption, UAV positioning, or throughput rather than detailed system delay and queue dynamics.

These prior works clearly show that UAV-assisted and MEC-enabled FL frameworks can be made more efficient through asynchronous aggregation, intelligent device selection, and computation offloading. However, several important gaps remain in the context of mission-critical IoT applications. First, many existing approaches either neglect the temporal structure of data arrivals (e.g., queue dynamics) or assume idealized MEC resources. Second, UAV-MEC servers are typically treated as highly capable static edge devices, whereas in practice they operate under stringent power, computation, and mobility constraints. Third, only limited attention has been paid to jointly optimizing delay, offloading decisions, and learning performance in a unified framework for UAV-aided FL.

In this dissertation, we focus on reducing system latency in UAV-aided FL environments by explicitly modeling and optimizing the utilization of spare computational capacity on UAV-MEC servers and by designing enhanced data offloading strategies. To further im-

prove performance, we propose prioritization schemes that select IoT devices for each FL iteration in a way that balances delay reduction with learning performance. Our approach thus provides a different optimization perspective, emphasizing timely and efficient model training under realistic UAV-MEC constraints, and complements the primarily energy- and throughput-oriented approaches in the existing literature. For benchmarking, we compare our proposed frameworks against standard FL [20] and edge-based learning systems [21], focusing on the efficiency and scalability of FL within IoT ecosystems.

2.5 Queue-Aware and GAN-Enhanced Federated Learning

In addition to resource allocation and hierarchical model design, two recent lines of work are particularly relevant for this dissertation: (i) the integration of queueing mechanisms into FL to better capture stochastic task arrivals and service dynamics, and (ii) the use of GANs to improve learning performance in Non-IID and privacy-constrained federated settings. Together, these directions motivate the development of queue-aware, GAN-enhanced FL architectures for UAV-assisted mission-critical networks.

2.5.1 Queueing Mechanisms in Federated Learning

Queueing-based formulations have recently been proposed to model the temporal evolution of computational tasks and model updates in FL systems. In [73], the authors considered collaborative heterogeneous multimedia edge networks, where task-priority queues are maintained at edge servers. IoT devices offload tasks to nearby edge devices, which can further relay tasks among themselves. By jointly optimizing task routing and offloading decisions, their framework reduces both delay and energy consumption, highlighting the benefits of explicit queue modeling in multi-edge environments.

In [74], FL is modeled as a closed Jackson network, where clients generate model update

tasks that are processed by a central server through a queueing structure. The authors proposed a non-uniform client sampling scheme that adapts to heterogeneous computational capacities, thereby reducing the average waiting time in asynchronous FL. This work explicitly links client selection policies to queue dynamics, demonstrating that sampling strategies can be optimized with respect to both learning and delay.

The work in [75] examined task offloading from IoT devices to edge devices, where task queues are managed before transmitting local models to the FL server. While the framework captures queue evolution on the device and edge side, the computational resources at the FL server remain largely underutilized from a scheduling perspective, leaving room for more comprehensive queue-aware designs that jointly consider edge and aggregation stages.

Overall, these queueing-based approaches show that modeling stochastic arrivals, service rates, and buffer states can significantly improve the timeliness and efficiency of FL systems. However, most existing works focus on flat or single-tier architectures and do not yet consider hierarchical or UAV-assisted FL with coupled queues across multiple layers.

2.5.2 GAN-Based Data Augmentation in Federated Learning

To alleviate the adverse effects of Non-IID data, class imbalance, and privacy constraints in federated settings, several recent works have incorporated GANs into FL architectures. In [12], the authors proposed an asynchronous FL framework for training GANs, where generators and discriminators are updated in a decentralized manner. Graph neural networks are used to model discriminator interactions, enabling scalable and asynchronous updates of the adversarial components across clients.

A personalized FL approach for smart healthcare was presented in [13]. In their proposed approach, IoT devices trained local discriminators while generators produced synthetic samples to re-balance underrepresented classes. The aggregated synthetic dataset was then shared among IoT devices, improving global model accuracy while respecting data locality and privacy concerns.

The work in [14] addressed privacy and data imbalance by introducing a divide-and-conquer GAN framework. Instead of sharing raw data, clients upload generator models to the server, which aggregates a subset of these generators into a global model. The global generator then produces IID-like synthetic samples to support local training, with fully homomorphic encryption ensuring privacy of exchanged parameters.

In [76], both generator and discriminator networks are trained locally, but only discriminators are transmitted to the server. The server aggregates discriminators and redistributes updated generator models, enabling privacy-preserving adversarial training without exposing raw data. A related extension to wireless edge environments was explored in [77], where edge servers aggregate IoT updates based on a synthesis score that accounts for model quality and dataset size, followed by hierarchical aggregation between edge and cloud layers.

The authors of [78] considered resource-constrained IoVs systems and proposed a GAN-based FL framework combined with a DDPG-based selection mechanism. Their goal was to balance training cost and synthetic data quality across heterogeneous devices. In [79], a weighted aggregation scheme was introduced in which each client's contribution to the global GAN model was scaled by its mean maximum discrepancy score, improving stability and representativeness in generator updates.

To strengthen privacy guarantees, [80] proposed a framework where each IoT device trains a local GAN and shares only synthetic data with the server. The server aggregates this synthetic data to build improved models while avoiding direct access to raw local datasets. Finally, [81] introduced a GAN-augmented FL framework that relies on synthetic data generation (using models trained with large-scale tools such as ChatGPT and Stable Diffusion in an AC-GAN setup) to mitigate Non-IID effects, thereby enhancing convergence and global accuracy.

Collectively, these GAN-assisted FL frameworks demonstrate that generative models can effectively address data heterogeneity, scarcity, and privacy constraints by enriching local datasets with synthetic samples or replacing sensitive data with privacy-preserving

surrogates.

2.5.3 Remaining Gaps and Motivation

The queue-aware and GAN-enhanced FL works discussed above are highly relevant to the objectives of this dissertation, but they exhibit several limitations when viewed from the perspective of mission-critical UAV-assisted systems.

First, queueing-based FL frameworks [73–75] primarily target flat edge or cloud architectures and do not explicitly model coupled queues across hierarchical tiers (e.g., IoT $\rightarrow$ follower UAV-MEC $\rightarrow$ leader UAV-MEC). As a result, they do not capture how backlogs and service rates at different layers interact to determine overall system delay in UAV-assisted networks.

Second, GAN-augmented FL frameworks [12–14, 76–81] are typically designed for static edge–cloud or server–client configurations. They rarely consider UAV-MEC constraints such as limited energy, dynamic coverage, and time-varying connectivity, nor do they account for queueing effects when synthetic data generation and offloading occur concurrently with federated training.

To the best of our knowledge, there is no existing work that jointly integrates: (i) hierarchical, queue-based modeling of data and task flows in UAV-assisted FL; and (ii) GAN-driven data augmentation at UAV-MEC servers to improve accuracy under Non-IID conditions, while explicitly considering delay and resource constraints. This dissertation addresses this gap by developing a queue-aware hierarchical FL framework that leverages GAN-based data augmentation at intermediate UAV-MEC layers, with the dual goals of mitigating the straggler effect and enhancing global model accuracy in dynamic, resource-constrained wireless environments.

2.6 Research Gap

The literature on optimizing FL for heterogeneous and resource-constrained wireless systems has grown significantly, with particular emphasis on mitigating the straggler effect, reducing delay, and improving energy efficiency. Existing efforts fall into several categories, yet each category exhibits limitations that restrict their applicability to realistic, multi-tier UAV-assisted environments.

2.6.1 Limitations of Existing Straggler-Mitigation Techniques

A large body of work focuses on asynchronous FL to alleviate the impact of slow IoT devices [54, 55, 59, 74]. Although asynchronous aggregation reduces waiting time, it introduces stale model updates, leading to degraded convergence and loss of global accuracy, particularly detrimental in mission-critical applications. Coded FL approaches [56,57] improve robustness by generating redundant parity data, but suffer from substantial communication overhead and increased computational load, making them less suitable for energy-limited UAV–MEC systems. Participant-selection methods [9, 62, 63] help reduce latency by excluding slow IoT devices, but may compromise statistical diversity and thus global accuracy, especially when data are highly Non-IID Computation offloading strategies [61, 65] reduce device-side delay but often treat MEC servers as static, high-capacity devices and do not account for the dynamic mobility, energy constraints, or coverage variability of UAV–MEC systems.

2.6.2 Limitations of Hierarchical FL in Dynamic Wireless Networks

Hierarchical FL (HFL) has emerged as a promising way to scale FL in large and heterogeneous networks, with several works optimizing resource allocation, association, and energy use [11, 57, 64, 82–85]. However, these approaches generally assume Static or grid-powered edge servers, not mobile UAV–MEC platforms. They also consider only deterministic task

arrivals, ignoring stochastic data generation typical of IoT sensing. They usually have single-tier or simplified multi-tier structures, with limited coupling between follower and leader servers. As a result, the temporal evolution of data, backlog accumulation, and real-time congestion effects in multi-tier UAV-assisted systems are not captured.

2.6.3 Gaps in Queueing-Aware FL Approaches

Queueing mechanisms have recently been introduced to model task arrivals, service rates, and scheduling in FL [73–75]. While these works demonstrate the value of queue-aware modeling, they are limited in the few ways. They focus on flat, single-server or asynchronous FL settings. They do not model hierarchically connected queues. They do not consider the mobility and energy constraints inherent to UAV-assisted systems. They do not integrate queue-aware processing with learning-quality improvements, such as synthetic data generation. Thus, the role of queueing in multi-tier UAV-assisted FL remains largely unexplored.

2.6.4 Gaps in GAN-Augmented FL for Non-IID and Resource-Constrained Settings

GAN-augmented FL frameworks [12–14, 76–81] demonstrate strong potential for mitigating Non-IID data and improving accuracy. However, existing works typically assume i). Static edge–cloud or client–server configurations, ii). No UAV mobility, coverage fluctuations, or energy constraints, iii). No modeling of queueing dynamics when synthetic-data generation and offloading occur concurrently and iv) No hierarchical interaction between multiple layers of GAN-assisted FL. Consequently, the integration of GAN-based augmentation into realistic queue-driven UAV–MEC environments remains unaddressed.

2.6.5 Summary of Research Gaps

The limitations discussed above reveal four critical gaps:

- **Lack of queue-aware hierarchical FL frameworks** that capture stochastic data arrivals, multi-tier queues, and coupled delays in UAV-assisted systems.
- **Insufficient modeling of UAV–MEC constraints** (mobility, power limits, trajectory-dependent connectivity) within FL optimization.
- **Absence of joint queueing + GAN augmentation approaches** that improve both delay performance and learning accuracy under Non-IID data.
- **No unified framework** that simultaneously addresses straggler mitigation, queue evolution, hierarchical aggregation, and accuracy enhancement in dynamic UAV-assisted networks.

2.6.6 Motivation for This Dissertation

This dissertation addresses these gaps by developing a novel **queue-aware, hierarchical, GAN-enhanced FL framework** tailored for UAV-assisted IoT environments. The proposed system

- Models multi-tier queues across IoT IoT devices, follower UAV–MEC servers, and leader UAV–MEC servers;
- Optimizes computation offloading, delay, and resource allocation under UAV power and mobility constraints;
- Performs GAN-based data augmentation at UAV–MECs to improve accuracy under Non-IID conditions;
- Provides a holistic evaluation of system delay, backlog evolution, and learning performance.

To the best of our knowledge, no existing work jointly optimizes queue dynamics, hierarchical FL, UAV–MEC resource constraints, and GAN-based accuracy enhancement within a single unified framework.

2.7 Summary

This chapter discussed the progression of resource optimization methods from traditional wireless networks to modern machine learning-driven and UAV-assisted FL systems. We first highlighted why resource optimization is crucial in future wireless networks that face heterogeneous device capabilities, dynamic traffic patterns, and strict latency and reliability requirements. Classical wireless resource optimization techniques were reviewed, followed by studies that perform optimization within machine learning environments, particularly within FL. These works addressed key objectives such as energy minimization, latency reduction, and model-parameter optimization.

Next, we surveyed resource optimization techniques in UAV-assisted FL-based wireless networks, including approaches for mitigating the straggler effect, MEC-assisted FL, device selection, and joint communication–computation optimization. While these studies introduced important progress, many relied on fixed MEC servers, single-tier network architectures, or deterministic assumptions regarding data arrivals. We then reviewed queue-aware FL approaches that incorporate stochastic task arrivals and scheduling constraints, along with GAN-enhanced FL frameworks designed to mitigate Non-IID data effects through synthetic data generation. However, existing works consider these components in isolation: queueing mechanisms are rarely integrated within hierarchical FL, UAV–MEC systems are commonly modeled without dynamic queues, and GAN-based augmentation is mostly restricted to static edge–cloud settings. These combined observations reveal a clear gap in the literature and motivate the development of the queue-aware, UAV-aided, and GAN-enhanced FL frameworks proposed in this thesis.

Chapter 3

Minimizing Delay in UAV-aided Federated Learning for IoT Applications

3.1 Introduction

Mission-critical IoT applications, such as forest-fire monitoring, flash-flood detection, border-patrol surveillance, and wildlife activity tracking, rely on real-time data collection and rapid decision-making under strict latency and reliability constraints. Many of these applications operate in remote or infrastructure-sparse environments, where permanent terrestrial base stations are either economically infeasible or operationally inefficient, since high connectivity is needed only intermittently during specific seasons or emergency periods. In such scenarios, deploying low-cost IoT devices deep inside forests or along remote borders is practical, but ensuring reliable communication and timely data processing remains a major challenge.

UAVs equipped with MEC capabilities have emerged as a promising solution to these challenges. Their mobility, flexible deployment, and ability to provide on-demand coverage enable efficient data collection and localized processing in dynamic environments. UAV-

MEC platforms can serve as aerial aggregation points and edge processors, reducing the reliance on distant cloud resources and significantly lowering communication latency.

FL, which trains models collaboratively across IoT devices without sharing raw data, offers a privacy-preserving and communication-efficient paradigm for distributed intelligence in such environments. However, the practical deployment of FL in UAV-assisted IoT networks is hindered by several limitations. IoT devices often differ in computational capabilities, battery budgets, and dataset volumes, leading to the well-known straggler effect: devices with insufficient processing power or disproportionately large datasets take significantly longer to complete local training, thereby delaying each global aggregation round. In mission-critical settings, such delays can severely degrade responsiveness and system performance.

To mitigate straggler behavior, partial dataset offloading to MEC servers has emerged as a promising direction. When UAVs serve as MEC platforms, they can process computationally intensive portions of local datasets on behalf of constrained IoT devices. However, UAV-MEC servers themselves operate under strict energy and processing limits, and their computational budgets must be carefully managed. The coupled nature of device-side processing, UAV-side computation, wireless channel variations, and dataset sizes makes optimal offloading nontrivial.

Existing works have explored asynchronous FL, coded FL, device selection strategies, and MEC-assisted processing to alleviate straggler effects. While these approaches improve performance under specific conditions, they often suffer from limitations such as stale updates (in asynchronous FL), high communication overhead (in coded FL), or degraded model generalization due to excluding slower devices (in device selection methods). Moreover, most MEC-based solutions assume static ground servers with abundant resources, overlooking the unique operational constraints of UAV-MEC systems.

These challenges motivate the need for a unified framework that optimizes computation offloading, energy consumption, and system delay in UAV-assisted FL environments. In particular, mission-critical IoT applications demand (i) delay-aware offloading decisions, (ii)

efficient utilization of UAV-MEC computational resources, and (iii) MEChanisms to incorporate device importance without compromising global model performance.

To address these challenges, this chapter proposes a UAV-aided FL (UAFL) framework in which straggling IoT devices offload a controlled portion of their datasets to a UAV-MEC server for remote processing. The framework optimizes transmission power, processing power, and offloaded data size under system power and quality-of-service constraints. Additionally, we incorporate a device-importance-aware participation strategy to improve the timeliness of FL iterations. The proposed optimization problem aims to minimize system delay while managing the distribution of computational load between IoT devices and the UAV-MEC server.

The following sections present the system architecture, mathematical formulation, proposed optimization methods, and an extensive performance evaluation comparing the UAFL framework with traditional FL and classical edge-learning baselines.

3.2 System Model and Problem Formulation

We consider a UAFL environment consisting of K number of IoT devices and Q number of UAV-MEC servers. The *k-th* IoT device has its dataset C_k that consists of input-output pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, where $\boldsymbol{x}_i$ are input features, while $\boldsymbol{y}_i$ are output labels. The *k-th* IoT device has a convolutional neural network (CNN) where it runs stochastic gradient descent [19] on its dataset using loss function $f(\omega_k) = loss(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to generate model parameters ω_k. Different straggling IoT devices offload their datasets based on their computation requirements to the *q-th* UAV-MEC server. The *q-th* UAV-MEC server combines and processes all offloaded datasets on its CNN. Then, it sends these computed model parameters ω_q^U to the global FL model at the primary aggregation UAV-MEC server to incorporate the effect of offloaded data on the global FL neural network. System model for proposed system is shown in Fig. 3.1.

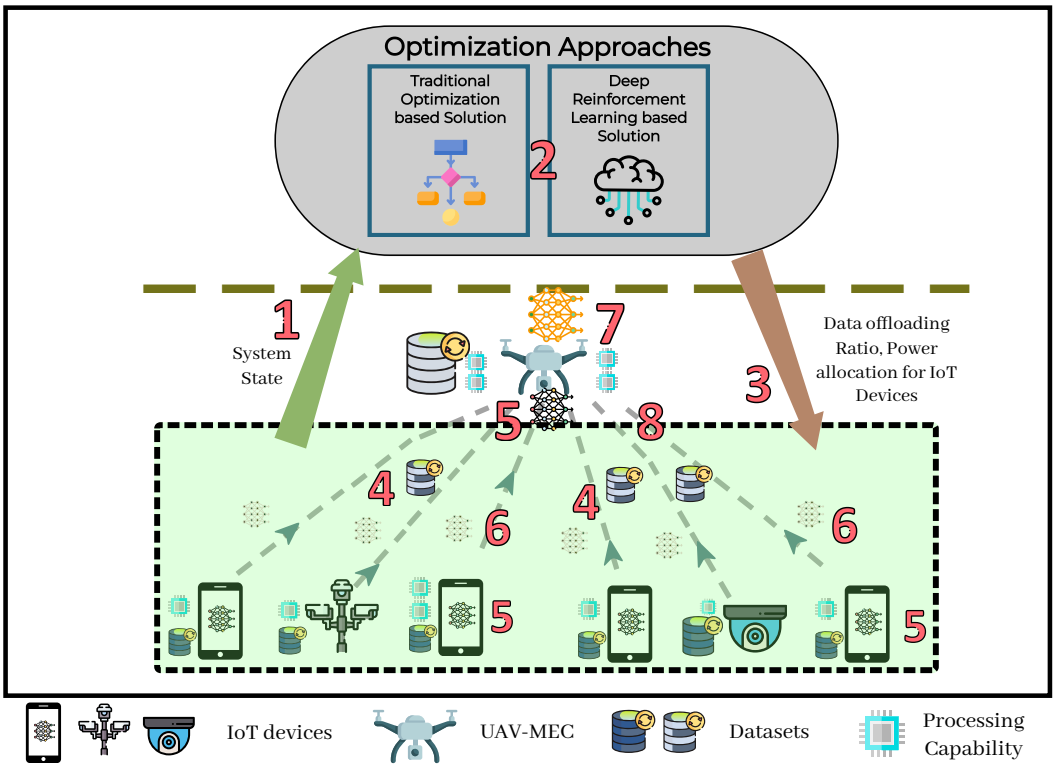

Figure 3.1: System model for UAV-aided FL platform.

In step 1, the system state is gathered, which contains information about device power, available computation capacity, and communication capacity of IoT devices and UAV-MEC in the environment and is passed on to optimization algorithms. The optimization algorithm makes offloading and power decisions based on the state information (step 2) and then communicates it back to the network entities (step 3) (i.e., UAV-MEC and IoT devices). The FL process comprises two stages; the first stage is where communications related to upcoming FL iterations take place. In this stage, the system also offloads the dataset from the straggler's node to the UAV-MEC (step 4). This offloading dataset can range from a small fraction to the entire dataset, depending on the decision made by our proposed algorithm, which considers factors such as IoT capacity, UAV-MEC capacity, and network conditions (e.g., bandwidth, data rate). In terms of Θ , if we are offloading few bytes of dataset from *k-th* IoT device then Θ_k^I will be very near to 0 and on the other extreme, if we are offloading whole dataset then Θ_k^I will be equal to Θ_{max}. Out system implementation is

such that it supports offloading in whole range of Θ, i.e., from 0 to Θ_{max} for all IoT devices. Once the communication stage is over, the FL process starts the FL training at IoT devices on their local dataset (i.e., the dataset which is left after offloading) and UAV-MEC on offloaded datasets (step 5). Once the training is complete on IoT devices and UAV-MECs, the resultant model parameters are forwarded to the aggregation model, aggregating these model parameters into a single global model (step 7). The global model then sends these parameters back to the participating IoT devices to start the process for the next global iteration.

The system considers one UAV-MEC as the primary aggregation server (along with its computation), and all other UAV-MECs are computation UAVs only. All UAV-MECs collect model parameters from connected IoT devices and then send these model parameters along with their own computed model parameters with this primary UAV. This primary UAV-MEC server then aggregates these received model parameters from other UAV-MEC servers and its own connected IoT devices, with its existing global learning model parameters and returns the updated model parameters to other UAV-MEC servers. UAV-MECs, including primary aggregation server, then broadcast model parameters to the connected IoT devices. This work does not consider selection and communication between UAV-MEC servers (we intend to work on it in the future). We consider this communication and model transmission time between UAV-MEC servers as a fixed part of aggregation and broadcasting delay.

The proposed system provides a QoS guarantee to IoT devices by ensuring a minimum data rate X_{min}. Since the UAV-MEC and IoT devices both have a limited power budget (which in turn impacts communication and computation limits), we work to find the best dataset offloading strategy to reduce system delay in this work.

3.2.1 System Delay Formulation

In each global iteration for FL, we have one or multiple local iterations at *k-th* IoT device. In each local iteration, the training continues until the loss on the local dataset is less than

a pre-defined threshold. Loss function on the *k-th* IoT device's CNN can be defined as:

$$F_k(\omega_k) = \frac{1}{R_k - \theta_k^I} \sum_{i \in (C_k \setminus \Theta_k^I)} f_i(\omega_k), \tag{3.1}$$

where ω_k is model parameters, C_k is the dataset, and $R_k = |C_k|$ represent samples in dataset at *k-th* IoT device. Θ_k^I is the offloaded dataset, $\theta_k^I = |\Theta_k^I|$ is the number of records (samples) in the offloaded dataset, and f_i is the loss function for *i-th* dataset record. In this work, $f_i(\cdot)$ denotes a standard supervised learning loss function; for classification tasks, it is instantiated as the categorical cross-entropy loss, while the formulation is kept generic to preserve applicability to other loss functions and learning tasks. The per-sample loss function $f_i(\cdot)$ is evaluated on the model output corresponding to the i-th sample, regardless of the underlying data modality, enabling support for image-, signal-, and tensor-based inputs commonly used in CNNs.

Offloading delay is the ratio of data to be offloaded and achievable bandwidth. Offloading delay for *k-th* IoT can be defined as:

$$D_k^O = \chi_k \frac{\Theta_k^I}{\hat{B} log_2(1 + p_k^O G_k)}, \tag{3.2}$$

where $\hat{B}$ denotes bandwidth, p_k^O denotes transmission power and G_k denotes gain at *k-th* IoT device. χ_k represents the IoT device selection variable. It will be one for the IoT device selected for the iteration and zero when the IoT device is not selected. The channel gain for *k-th* IoT device can be calculated as [86]: $G_k = 10^{-\varphi_k/10}$, where φ_k is the path loss for *k-th* IoT device. We consider the air-to-ground channel model utilized in [87]. Path loss between UAV-MEC and *k-th* IoT device can be calculated as:

$$\varphi_k = \frac{\eta_{LoS} - \eta_{NLoS}}{1 + a \exp[-b(\frac{180}{\pi} \varrho_k - a)] + 20 \log(\frac{4\pi \hat{f} d_k}{\hat{c}}) + \eta_{NLos}}, \tag{3.3}$$

where $a, b, \eta_{Los}, \eta_{NLos}$ are decided based on communication environment. $\varrho = \sin^{-1}(z_k/d_k)$ is

the angle of IoT device k to UAV-MEC. d_k is the distance from *k-th* IoT device to UAV-MEC and z_k denotes the height of UAV-MEC from ground.

On each IoT device, we have stochastic gradient descent running, and the total computation $\beth_k$ at *k-th* IoT device can be calculated as [88]:

$$\beth_k = \Upsilon B_k^n b_k \lambda = \Upsilon C_k \lambda, \tag{3.4}$$

where Υ denotes epochs per round, $B_k^n = \frac{\theta_k}{b_k}$ denotes (mini) batches per epoch for *k-th* IoT device, b_k denotes the number of records in each batch and λ denotes cycles required for training 1 bit of data. We can calculate the computation capacity of *k-th* IoT as:

$$s_k = \sqrt{\frac{p_k^P}{\nu_k}}, \tag{3.5}$$

where ν_k denotes the capacitance coefficient and p_k^P represents the power available for processing. Computation delay at *k-th* IoT device can be written as:

$$D_k^P = \frac{(C_k - \Theta_k^I)}{s_k}. \tag{3.6}$$

The loss function at the *q-th* UAV-MEC's CNN with the offloaded dataset is defined as:

$$F(\omega_q^U) = \frac{1}{\mathfrak{s}_q} \sum_{i \in \mathfrak{S}_q} f_i(\omega_q^U), \tag{3.7}$$

where ω_q^U denotes the model parameters of *q-th* UAV-MEC after processing offloaded datasets. $\mathfrak{S}_q = \sum_{k \in K} \chi_k \Theta_k^I$ is the combined dataset of offloaded datasets (to *q-th* UAV-MEC server), $\mathfrak{s}_q = |\mathfrak{S}_q|$ is combine length of all offloaded datasets. We can redefine the data aggregation process as follows:

$$\omega = \frac{1}{R} \left(\sum_{k \in K} \chi_k (R_k - \theta_k^I) \omega_k + \theta \sum_{q \in Q} \omega_q^U \right), \tag{3.8}$$

where ω are the global aggregated model parameters transmitted by the UAV-MEC aggrega-

tion server back to IoT devices, $R = \sum_{k \in K} R_k$, are the samples of total dataset offloaded by IoT devices and $\theta = \sum_{q \in Q} \mathfrak{s}_q$ is the sum of all offloaded dataset samples . The computation capacity of *q-th* UAV-MEC can be written as:

$$s_q^U = \sqrt{\frac{t_q^P}{\xi_q}}, \tag{3.9}$$

where ξ_q is the capacitance coefficient and t_q^P is the power available for processing at *q-th* UAV. The computation delay of *q-th* UAV-MEC can be written as:

$$E_q^P = \frac{\Upsilon \lambda \sum_{k \in K} \Theta_k^I}{\sqrt{\frac{t_q^P}{\xi_q}}}. \tag{3.10}$$

System processing delay can be defined as:

$$D_P = N \max\{\max_{k \in K} D_k^P, \max_{q \in Q} E_q^P\} + N D_{ab}, \tag{3.11}$$

where D_{ab} is the aggregation and broadcasting delay, N is the number of iterations required to converge the model to a specific accuracy and can be written as [89]:

$$N = \left[\left(1 + \frac{1}{K} \right) \Upsilon \alpha + \frac{\beta}{\Upsilon} \right] \frac{1}{\epsilon}. \tag{3.12}$$

where α and β are FL parameters (configuration and data distribution), and ϵ denotes the certain training loss that we want to achieve. We utilize the approach in [47] to calculate these parameters. More specifically, we do curve fitting on historical data of FL and derive the relationship between training rounds and data characteristics to derive values of α and β.

3.2.2 Node Importance

Our system model considers K IoT devices, each with different computational and communication capacities. If any of these devices encounter challenges in processing its local dataset, it introduces delays in the system. We employ dataset offloading to the UAV-MEC to mitigate this issue, leveraging its computational capabilities. The optimization of system delay is further examined by evaluating the significance of each connected IoT device [90]. This involves reducing the number of IoT devices in each iteration, focusing solely on those with higher importance. The importance is determined by considering reliability, connectivity, and participation in iterations. Afterwards, we calculate importance using an update gradient, i.e., the update gradient is computed for each IoT device at the UAV-MEC server. Utilizing these gradients, the UAV-MEC calculates the importance of each IoT device. In this context, the gradient serves as a metric for measuring the contribution of a specific node to the convergence of the FL algorithm. The gradient of *k-th* IoT device at time t can be calculated as [91]:

$$\sigma_{k,t} = || \sum_{t=1}^{T} \sum_{i \in C_k} \bigtriangledown f(\omega_k : \boldsymbol{x}_i, \boldsymbol{y}_i)||, \tag{3.13}$$

where $\sigma_{k,t}$ denote magnitude of local gradient of each device at time t, $\bigtriangledown f(\omega_k : \boldsymbol{x}_i, \boldsymbol{y}_i)$ denotes the gradient at sample$(\boldsymbol{x}_i, \boldsymbol{y}_i)$ of dataset.

The UAV-MEC calculates the importance of *k-th* IoT device using the following:

$$\Upsilon_{k,t} = h \frac{\sigma_{k,t}}{\sum_{i \in U} \sigma_{i,t}} + (1-h) \frac{\max_{i \in U} L_k - L_i}{\sum_{j \in U} (\max_{i \in U} L_j - L_i)}, \tag{3.14}$$

where $\Upsilon_{k,t}$ is the importance of node k at time t, h is user-defined importance configuration variable and $h \in [0, 1]$, L_k is the distance of node k from UAV-MEC. U denotes the set of devices selected for this iteration. The UAV-MEC selects the top X percent devices from the devices (i.e., added to set U) with maximum importance calculated at the above step and sets $\chi = 1$ for selected IoT devices and $\chi = 0$ for not selected IoT devices.

3.2.3 Optimization Problem

In the optimization problem, we minimize system delay while considering energy variables and dataset offloading size. In simple terms we are working to reduce system delay by optimizing system's processing power (P_P^k), transmission power (P_O^k) along with data dataset offloading size (Θ_k^I). The complete optimization problem can be written as:

$$\min_{\Theta_k^I, p_k^O, P_P^k} : \max_{k \in K} D_k^O + D_P$$

Subject to:

$$\begin{aligned}
&C1 : \sum_{k \in K} p_k^O + \sum_{k \in K} p_k^P \leq P_{total}^{IoT} \\
&C2 : \sum_{q \in Q} t_q^O + \sum_{q \in Q} t_q^P \leq P_{total}^U \\
&C3 : 0 \leq p_k^O, p_k^U \leq P_{max}^{IoT}, \forall k \in K \\
&C4 : P_{min}^{IoT} \leq p_k^O + p_k^U \leq P_{max}^{IoT}, \forall k \in K \\
&C5 : P_{min}^U \leq t_q^P \leq P_{max}^U \forall q \in Q \\
&C6 : 0 \leq \Theta_k^I \leq \Theta_{max}, \forall k \in K \\
&C7 : \hat{B} log_2(1 + p_k^O G_k) \geq X_{min}, \forall k \in K \\
&C8 : \chi_k \in \{0, 1\}, \forall k \in K,
\end{aligned} \tag{3.15}$$

where P_{total}^{IoT} denotes the total power budget for IoT devices, P_{total}^U represents the total power budget for UAVs, P_{max}^{IoT} represents maximum power limit for an IoT device, and P_{min}^{IoT} denotes minimum power limit for an IoT device. P_{min}^U and P_{max}^U represent the minimum and maximum computation power budget for UAVs, respectively. C1 and C2 limit the total power consumption of IoT devices and UAVs, respectively. C3 and C4 ensure that communication and computation power is greater than zero. C5 limits the computation power for each UAV. C6 limits the maximum offloaded data to be less than available data on IoT devices. C7 is a QoS constraint and ensures the data rate achieved by each IoT exceeds the minimum data

rate. C8 ensures that each IoT device is either selected or not-selected i.e. χ is either zero or one for each iteration.

3.3 Proposed Solution

In this section, we propose two solutions for solving the problem in (5.16). First, we solve the problem using a traditional optimization approach. Afterward, we propose a secondary reinforcement learning solution with less runtime complexity.

3.3.1 Traditional Optimization-based Solution

The objective function is not jointly convex on optimization variables, as system variables are not mutually convex. This makes the proposed problem a non-convex optimization problem. We decouple this problem into a master and two sub-problems. We consider θ_k^I as a coupling variable and drive the solution fixing θ_k^I for sub-problems. The first sub-problem can be stated as:

$$\begin{aligned} \min_{p_k^O} : & \left\{ \max_{k \in K} \left(\chi_k \frac{\Theta_k^I}{\hat{B} log_2(1 + p_k^O G_k)} \right) \right\} \\ & \text{Subject to: } C1 \text{ and } C3. \end{aligned} \tag{3.16}$$

We optimize (3.16) for p_k^O, and all the constraints are linear, making it a linear sub-problem. We convert this problem into epigraph form and introduce an auxiliary variable T_1. The problem can now be represented as:

$$\begin{aligned} & \min_{p_k^O, T_1} : T_1 \\ & \text{Subject to: } C1 \text{ , } C3 \text{ and} \\ & C9 : \chi_k \frac{\Theta_k^I}{\hat{B} log_2(1 + p_k^O G_k)} < T_1 \forall \ k \in K. \end{aligned} \tag{3.17}$$

We optimize the computation delay at IoT devices and UAV-MEC for the second sub-problem. We improve the delay by optimizing processing power on IoT devices and UAV-MEC. The problem can be written as:

$$\min_{P_P^k,\chi_k,t_q^P} : \max\left\{\max_{k\in K}\chi_k\left(\frac{(C_k-\Theta_k^I)}{\sqrt{\frac{p_k^P}{\nu_k}}}\right), \max_{q\in Q}\left(\frac{\sum_{k\in K}\Theta_k^I}{\sqrt{\frac{t_q^P}{\xi_q}}}\right)\right\} \tag{3.18}$$
$$\text{Subject to: } C1 \text{ , } C3 \text{ and } C5.$$

By converting (5.16) into epigraph form and introducing an auxiliary variable, we can convert the problem into a minimization problem.

$$\begin{aligned}
&\min_{p_P^k,p_U,T_2} : T_2 \\
&\text{Subject to: } C1 \text{ , } C3 \\
&C10 : T_2 >= \chi_k\frac{(C_k-\Theta_k^I)}{\sqrt{\frac{p_k^P}{\nu_k}}}\forall k\in K \\
&C11 : T_2 >= \frac{\sum_{k\in K}\Theta_k^I}{\sqrt{\frac{t_q^P}{\xi_q}}}\forall q\in Q.
\end{aligned} \tag{3.19}$$

where C10 and C11 are constraints introduced by converting the problem into epigraph form. Replacing the P1 and P2 in the original problem and reshuffling variables in C9, C10, and C11 constraints.

$$\begin{aligned}
&\min_{\Theta_k^I,p_k^O,P_P^k,T_1,T_2} : T1 + ND_{ab} + N\Upsilon\lambda T2 \\
&\text{Subject to: } C1-C7, \\
&C12 : T_1(\hat{B}log_2(1+p_k^O G_k)) - (\chi_k\Theta_k^I) \geq 0\forall k\in K \\
&C13 : T_2\sqrt{\frac{p_k^P}{\nu_k}} - (C_k-\Theta_k^I) \geq 0\forall k\in K \\
&C14 : T_2\sqrt{\frac{t_q^P}{\xi_q}} - (\sum_{k\in K}\Theta_k^I\chi_k) \geq 0\forall q\in Q.
\end{aligned} \tag{3.20}$$

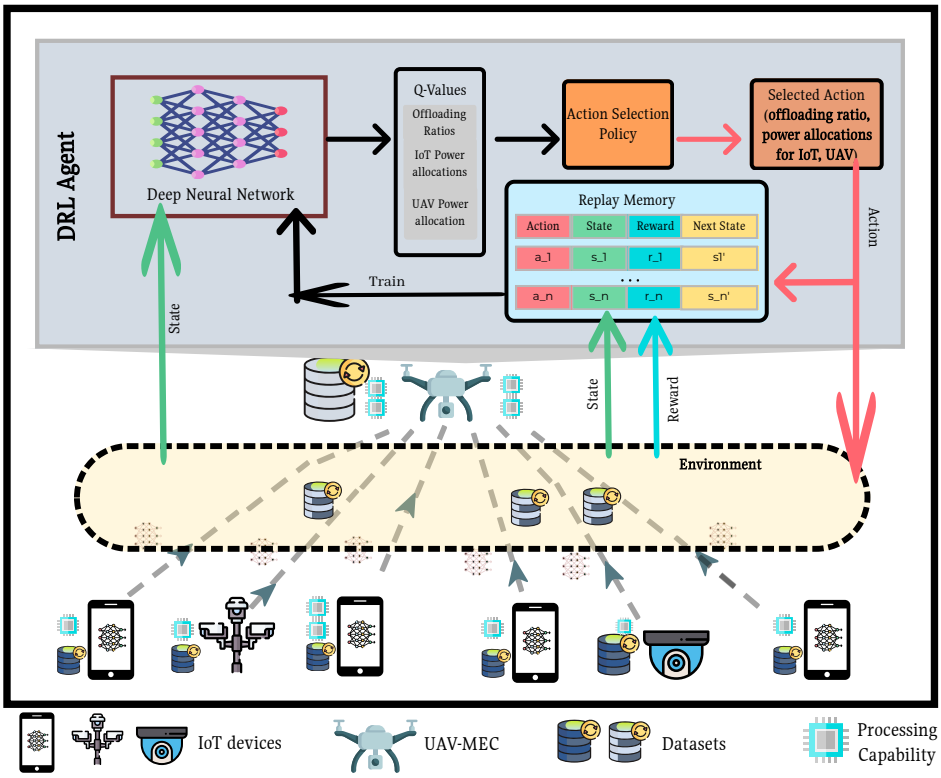

Figure 3.2: DRL-based solution for UAFL framework.

To solve this modified problem, we use the deterministic concurrent simplex algorithm. This algorithm applies dual and primal simplex with root relaxations to find the optimal solution concurrently [92]. We utilize Python with the Gurobi optimization toolkit for the simulations.

3.3.2 Reinforcement Learning-based Solution

In the proposed UAFL framework, each UAV-MEC functions as an agent, with the environment enclosing all IoT devices. We utilize a single-agent variant of deep Q networks in this work. In our scenario, the agent is deployed at the UAV-MEC server. Within the framework of reinforcement learning (i.e., deep Q-learning), the agent maintains a Q value table $Q(\mathfrak{s}^j, a^j)$, where $\mathfrak{s}$ denotes the state vector, a signifies the actions taken by the agent, and Q value denotes the expected reward obtained by executing action a in state $\mathfrak{s}$ at time j. The agent determines the action a based on the Q value table and communicates it to the

environment. The environment, in turn, provides the reward value r^j to the agent, based on the current system state and action. Additionally, the environment provides the subsequent state $\mathfrak{s}^{j+1}$ based on the action a^* chosen by the agent, which the agent utilizes to update its Q value. The agent takes the action with the highest Q value under the specified action selection policy (E-greedy policy in our case). The overall view of the DRL-based solution for UAFL is illustrated in Fig. 3.2.

The updated value of the agent's Q table can be written as:

$$Q_{\text{updated}}(\mathfrak{s}_k, a^j) = Q(\mathfrak{s}_k, a^j) + \mathfrak{a}\big[r + \mathfrak{B}\max_{a\in A} Q(\mathfrak{s}^{j+1}, a) - Q(\mathfrak{s}^j, a)\big], \tag{3.21}$$

where $\mathfrak{a}$ is the learning rate, and $\mathfrak{B}$ is the attenuation factor.

At any time j, the state vector of the environment encompasses six parts: (i) power budget for UAV-MECs $t = [T_1, T_2, ..., T_q]$, where T_q is the power budget for qth UAV, (ii) the power budget for IoT devices $P = [P_1, P_2, ..., P_k]$, where P_k represents power budget for $k-th$ IoT device, (iii) the minimum and maximum power limits for UAVs $t^{lim} = [(t_1^{max}, t_1^{min}), (t_2^{max}, t_2^{min}), ...(t_q^{max}, t_q^{min})]$, where t_q^{max} and t_q^{min} represent maximum and minimum power limits for $q-th$ UAV-MEC respectively, (iv) the power limits for IoT devices $p_k^{lim} = [(p_1^{max}, p_1^{min}), (p_2^{max}, p_2^{min}), ...(p_k^{max}, p_k^{min})]$, (v) the device selection variable for IoT devices $\chi = [\chi_1, \chi_2, ..., \chi_k]$, where χ_k represents the selection of $k-th$ IoT device for the iteration, and (vi) the minimum data rate IoT devices require X_{min}.

The system state can be represented as:

$$\mathfrak{s}^j = \big\{T, P, t^{lim}, p^{lim}, \chi, X_{min}\big\} \tag{3.22}$$

Whenever a new state is generated (i.e., a new global iteration occurs), the agent will choose a new appropriate action to act on the environment according to the current system state. In our scenario, the action is composed of the following variables: (i) the power allocation for all IoT devices $p = [p_1, p_2, ..., p_k]$, where p_k represent the power allocated to

$k-th$ IoT device, (ii) power allocated to UAVs $t = [t_1, t_2, ...t_q]$, where t_q is the power allocated to $q-th$ UAV-MEC, (iii) offloading decisions/ratio/percentage for IoT devices $\Theta = [\Theta_1, \Theta_2, ...\Theta_k$, where Θ_k is the offloading variable for $k-th$ IoT device. The whole action space can be represented as:

$$a^j = \{p, t, \Theta\} \tag{3.23}$$

Once the action is sent to the environment, the reward will be generated by the reward r^j. In our scenario, this reward is calculated by combining the offloading delay and processing delay from (4.22) and (5.15) respectively then multiplying the result by a negative factor. This reward is fed back to the agent. Stated the reward is a reciprocation of system delay. The more delays result from actions sent by the agent, the less will be the reward and vice versa.

We can deploy this DRL-based solution alongside a traditional optimization-based solution for training. The DRL-based solution receives the same data as the traditional optimization solution and trains on that dataset for a specific amount of iterations (our experiments showed that after 2000 iterations, predictions of the DRL-based solution were on par with the traditional optimization solution). Once we deploy a trained DRL-based solution to the UAV-MEC environment, the decision-making cost is reduced drastically from exponential time to constant. Specifically, in this case, the worst-case complexity of the traditional optimization approach will be $O(2^n)$. For the DRL-based solution, the runtime complexity is $O(1)$. This reduced complexity makes DRL-based solution more desirable to be utilized in most cases.

3.4 Performance Evaluation

We present the performance analysis of the proposed UAFL framework compared to traditional FL [20] and centralized edge-AI learning (CEL) which refers to the application of

the MEC system for computation offloading [21]. In the traditional FL, each IoT device performs local FL iterations to train its model parameters using its local dataset. Once the local iterations are completed, the device sends its computed parameters to the aggregation server (in this case, the UAV-MEC server). The aggregation server aggregates model parameters from all IoT devices and returns a single global aggregated model parameter. Each IoT device then updates its local parameters based on the received parameters from the aggregation server before the next global iteration. In contrast, in the edge-AI approach (CEL), datasets are offloaded to edge servers, and FL iterations are performed over the edge using various computation UAV-MEC servers. In this work, we employ different approaches, each represented by the following schemes:

- "FL": traditional FL [20]
- "CEL": the Centralized Edge-AI Learning approach presented in [21]
- "UAFL": our traditional solution approach
- "UAFL_drop": our traditional solution approach with integrated node selection
- "DRL": our DRL-based solution approach

3.4.1 Configuration and experimental procedures

The experimental configuration is selected to reflect a practical UAV-assisted IoT deployment while maintaining computational tractability. Specifically, we consider a network comprising $K = 20$ IoT devices served by a single UAV-MEC ($Q = 1$), representing a moderate-density sensing cluster commonly adopted in UAV-assisted edge learning studies [93]. This setting allows the straggler effect, queue dynamics, and delay–accuracy trade-offs to be clearly observed without introducing additional coordination complexity associated with multi-UAV systems. While K is fixed in the baseline experiments, the proposed optimization framework is independent of the network size and can be directly extended to denser deployments.

The bandwidth for the channel is set to $\hat{B} = 20MHz$. The power budget for UAV-MECs P^{U}_{total} and IoT devices P^{IoT}_{total} is set to be 10 watts (unless explicitly stated otherwise). In each power section, i.e., IoT device power or UAV-MEC power, we subdivided the power into computation/processing power p^{P}_{k}/t^{P}_{q} and offloading power $p^{O}_{k}/\ t^{O}_{q}$. We consider that the power budget for communication for UAVs t^{O}_{q} is fixed. It can be considered constant as we communicate with a fixed number of IoT devices and UAV-MECs and almost a fixed communication load for each IoT device and UAV-MEC. We consider a minimum and maximum power limit for both computations and communications for each IoT device. The minimum power for an IoT device is P^{IoT}_{min}=0.01 W, and the maximum power is P^{IoT}_{max}= 0.5 W for both communication and computations. For UAV, the lower limit for computation power is P^{U}_{min}= 5W, and the upper limit is P^{U}_{max}= 10 W. The selected system parameters align with standard 5G new radio channel configurations supported for aerial communications, as specified in 3GPP TR 36.777 [93] and widely used in related UAV-MEC literature. We consider the MNIST dataset of size 400Mb in our simulations [94]. We utilized 80,20,20 training, testing, and validation splits for our dataset. We trained our model against the training dataset and improved it against the validation dataset. For reporting results we utilized testing portion of dataset not shown to models before. The MNIST dataset is adopted as a benchmark to provide a controlled and reproducible learning environment, allowing the impact of the proposed queue-aware offloading and FL mechanisms to be isolated from dataset-specific complexity. The objective of the experimental evaluation is not to demonstrate state-of-the-art image classification performance, but rather to analyze system-level metrics such as training delay, queue stability, offloading efficiency, and resource utilization under Non-IID data distributions. Robustness of the proposed framework is evaluated with respect to dynamic network conditions, heterogeneous device capabilities, and non-IID data distributions, rather than across multiple datasets. Since the proposed optimization and offloading strategies are independent of the specific dataset and loss function, the conclusions drawn from MNIST generalize to other supervised learning tasks. Using a lightweight

dataset enables extensive experimentation across varying traffic loads, offloading ratios, and resource constraints, which would be computationally prohibitive with large-scale datasets in UAV-assisted environments. We consider the capacitance coefficient for UAV (ξ) and IoT (ν) to be $1e^{-19}$ [95]. We fixed a time slot of 200ms for parameter aggregation and model broadcast. We consider α to be 1 and β as 40. We set the target training accuracy to be 98%. The results are aggregated over 10^3 Monte-Carlo simulations and include a 95% confidence interval. We set the importance criteria of top 80% for IoT devices selected for iterations. For the neural network structure utilized by FL, we consider a CNN with two convolutions and a linear output layer with flattening in between. We prioritize IoT devices based on Eq. (3.14) for simulations. For reinforcement learning, we consider the epsilon greedy policy. We do 2000 training iterations for training. We utilize replay memory of size 10000. The batch size is set to 128, the epsilon start is set to 0.99, and the epsilon end is set to 0.1. We consider the decay rate for epsilon to be 1000 (higher means slower decay). We use Huber loss as a loss function and Adam optimizer with a learning rate of 0.0001 for network optimization. We utilize a deep neural network (DNN) with three layers for Q-table estimation. Detailed simulation parameters are shown in Table 5.1.

We utilized Python to create these simulation environments. For traditional optimization, we utilized Gurobi, while for implementing FL and deep reinforcement learning, we utilized PyTorch. We utilized the Google Cloud Platform compute engine to run these simulations. More details of this environment are enlisted in Table 4.2.

3.4.2 Simulation Results

We present the simulation results to evaluate the performance of our proposed framework based on metrics such as testing accuracy, system-wide delay, and power saved under varying network conditions and resource constraints. The results are compared against baseline approaches to demonstrate the effectiveness of our method in optimizing model training in a UAV-assisted FL environment. It's worth noting that the methods proposed throughout the

Table 3.1: Simulation parameters.

Description	**Parameter**	**Value**
IoT devices	k	20
UAV-MEC servers	Q	1
Channel bandwidth	$\hat{B}$	20 MBs
Power budget for UAV-MECs	P^{U}_{total}	10 W
Power budget for IoT devices	P^{IoT}_{total}	10 W
Minimum power for IoT device	P^{IoT}_{min}	0.01 W
Maximum power for IoT device	P^{IoT}_{max}	0.5 W
Minimum computation power for UAV-MEC server	P^{U}_{min}	5 W
Maximum computation power for UAV-MEC server	P^{U}_{max}	10 W
Capacitance coefficient for UAV-MEC	ξ	$1e^{-19}$
Capacitance coefficient for IoT	ν	$1e^{-19}$
Dataset utilized for FL	MNIST	400Mb
FL parameter	α	1
FL parameter	β	40
FL training accuracy		98%
FL model	CNN	2 convolution layers and 1 linear layer
Transmission and broadcast delay	D_{ab}	200ms
Minimum data rate	X_{min}	1Mb/s
Device selection base on importance	χ	top 80%
Policy iteration	-	Epsilon Greedy policy
Training epochs	-	10000
Replay Memory	-	2000
DRL batch size	-	128
Epsilon start	-	0.99
Epsilon end	-	0.1
Epsilon decay rate	-	1000
Loss function	-	Huber loss
Optimizer	-	Adam
Optimizer learning rate	-	0.0001
DQN neural network	-	DNN with three layers

Table 3.2: Implementation details.

Parameter	**Value**	**Specification**
Language	Python	3.10.0
Traditional Optimization	Gurobi	10.0.2
Federated Learning	PyTorch	2.1.0
Reinforcement Learning	PyTorch	2.1.0
Compute Engine	Google Cloud Compute	current
Number of vCPU Cores	8	N1
Memory	32 Gb	RAM
Permanent Storage	500Gb	SSD
Graphics Processing Unit	16Gb	Nvidia Tesla T4

thesis are not intended to compete as alternative classifiers, but rather to address complementary aspects of UAV-assisted FL systems. Accordingly, they are evaluated using different performance metrics aligned with their respective objectives. Specifically, the methods are compared and evaluated along the following dimensions: i) **Predictive performance:** measured in terms of classification accuracy to ensure learning quality is preserved, ii) **Computational complexity:** assessed via per-round computation cost and solver convergence behavior, iii) **Offloading capability:** evaluated through the amount of data and computation successfully offloaded under delay constraints, iv) **Resource efficiency:** quantified in terms of power consumption, bandwidth utilization, and queue stability.

System Accuracy

Fig. 3.3 shows the testing accuracy of each IoT device's local model before the weights are offloaded to the UAV-MEC server for aggregation for each iteration. The initial rounds show some variability in accuracy across the IoT devices, reflecting differences in local data distributions and learning progress. However, as the global iterations proceed, the accuracy of each device steadily improves, showing the effectiveness of the proposed UAFL approach. The results show the advantage of the UAFL, where the UAV-MEC server acts as a central

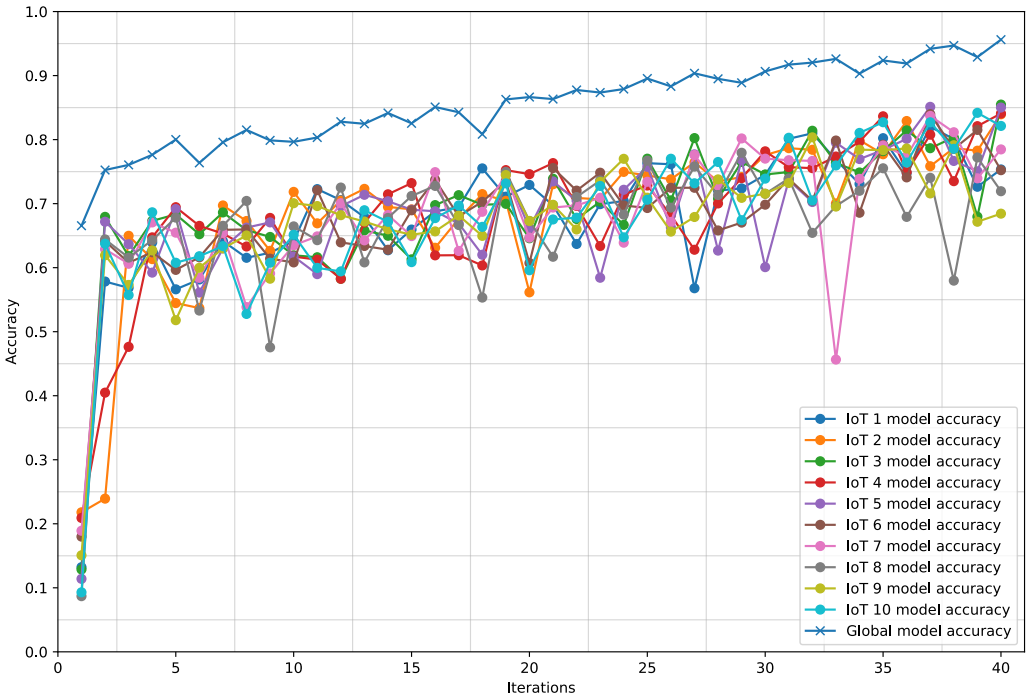

Figure 3.3: Test accuracies for global model and individual IoT devices for UAFL.

entity that enhances learning by leveraging offloaded data and providing global weight updates without compromising system-wide accuracy. As the iterations progress, the system ensures convergence towards higher accuracy for all IoT devices, highlighting the effectiveness of the proposed collaborative learning approach. The figure shows that, despite the initial variability, UAFL leads to consistent improvements in accuracy across all IoT devices as the number of global iterations increases.

Fig. 3.4 (a) illustrates the average testing accuracy of IoT devices for each iteration for different solution approaches. Among the approaches, FL demonstrates the highest average accuracy, which is attributed to the fact that FL processes the largest amount of data on each IoT device, which results in more generalizable local models and, consequently, higher average accuracy per device. In contrast, UAFL and DRL exhibit slightly lower accuracies, falling within a similar range. This can be explained by the smaller datasets processed

Figure 3.4: Average accuracies vs. global FL iterations for the proposed frameworks in terms of (a) average IoT device accuracies and (b) global model accuracies.

at each iteration compared to FL, leading to a less generalized model on individual IoT devices. The UAFL_drop approach, which incorporates IoT device dropouts during each

global iteration, shows marginally lower performance than UAFL, DRL, and FL. This is due to the reduced collective learning capacity when some IoT devices are excluded from the training process. This results in less data diversity and a smaller pool of model updates. CEL shows zero accuracy for IoT devices, as all data is offloaded to the UAV-MEC server, and no model training occurs locally on the IoT devices.

Fig. 3.4 (b) presents the global model accuracy on the UAV-MEC server for each global iteration. The accuracy of the global model corresponds to the average IoT device accuracies depicted in Fig. 3.4 (a). Notably, CEL slightly outperforms the other approaches in terms of global accuracy. This can be attributed to the fact that the entire dataset from all IoT devices is offloaded to the UAV-MEC server, allowing for training a single comprehensive model over the combined data. UAFL, DRL, and FL show comparable performance in global accuracy, with slight variations. However, UAFL_drop lags behind, especially in earlier iterations. The reason for this delay in achieving the target accuracy is the reduced number of devices selected for training in each iteration, which results in a lower variety of datasets available for learning. Consequently, the UAFL_drop model struggles to generalize effectively, reaching the required accuracy threshold only in the final iteration, whereas other approaches achieve this threshold few of iterations earlier.

System Delay

Fig. 3.5 illustrates the relationship between system delay and the power allocation for IoT devices for $K = 10$ IoT devices. Notably, the system delay decreases as the power budget allocated to IoT devices increases. This phenomenon occurs because a higher power budget enables IoT devices to engage in more local processing, thereby reducing computation delay in the system, particularly evident in scenarios involving FL, UAFL, UAFL_drop, and DRL. Contrarily, in the context of CEL, there is a marginal alteration in delay with increasing power per device. This is primarily due to the data being entirely offloaded to the UAV-MEC server, where greater power at IoT devices translates to increased capacity

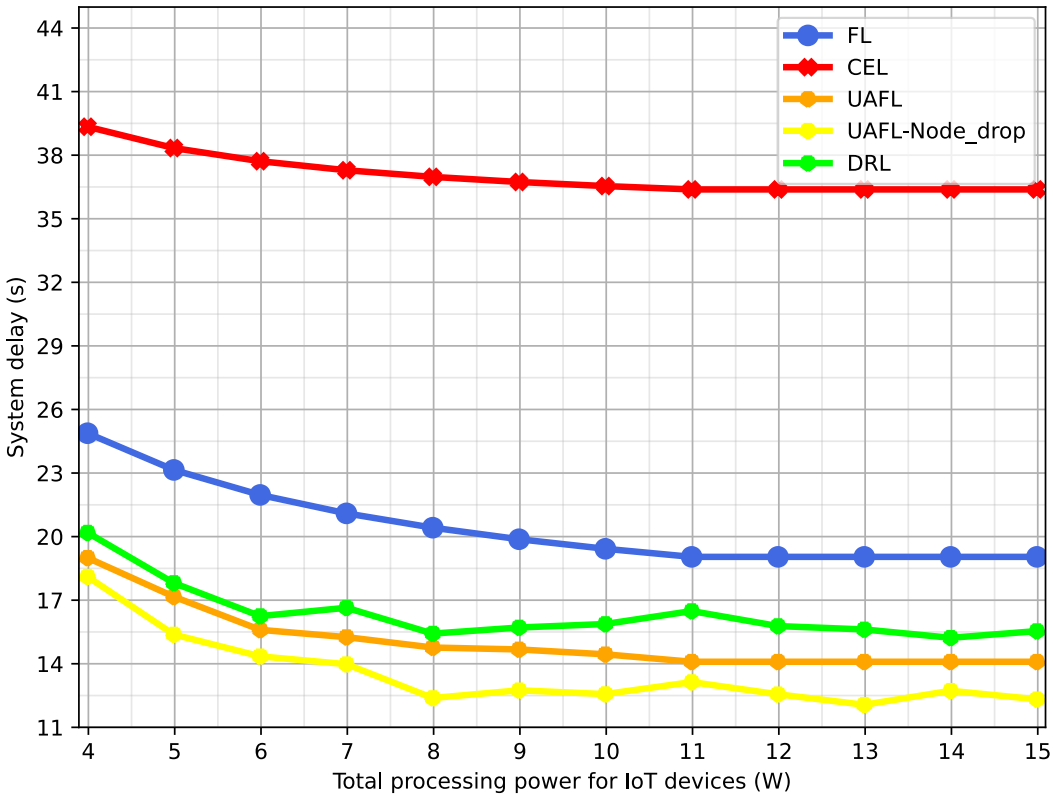

Figure 3.5: System delay versus power allocation for IoT devices for different solution approaches.

for offloading. In contrast, the effect is minimal in FL since no data offloading occurs. However, in DRL, UAFL and UAFL_drop, where data offloading is part of the process but constitutes a smaller proportion than local processing, optimal offloading results in lower system delay than FL. Despite UAFL_drop exhibiting similar performance to UAFL, it experiences slightly lower delay attributed to less number of devices for the same power budget in the iteration. Additionally, the DRL-based implementation of UAFL performs slightly worse than UAFL but still significantly outperforms FL and CEL by utilizing an optimal dataset offloading size. We can see from these results that UAFL_drop out performs other schemes. The fewer IoT devices allow for a slightly higher power budget per device, resulting in a slightly shorter delay.

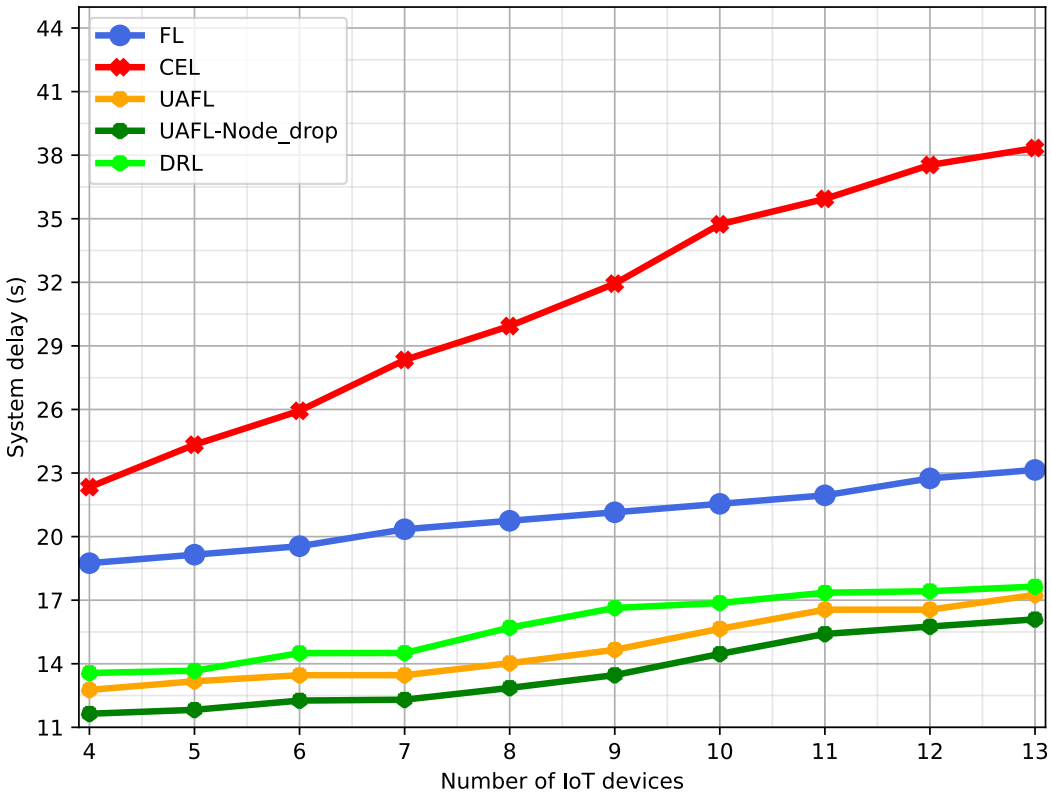

Figure 3.6: System delay versus number of IoT devices with fixed power budget.

Fig. 3.6 shows the system delay as the number of IoT devices increases for the fixed power budget for IoT devices at 10W. The result indicates that the system delay also increases as the number of IoT devices increases. Among the options considered, CEL exhibits the highest delay. In CEL, where the dataset is offloaded to the UAV-MEC server, an increase in the number of IoT devices leads to a linear increase in the size of the processing dataset, hence the escalating delay. FL outperforms CEL as it processes data locally without offloading, resulting in better delay performance. However, as the number of IoT devices increases, the reduced power per device leads to increased delay as processing local datasets becomes more resource-intensive. UAFL and UAFL_drop demonstrate the best performance among the four approaches (CEL, FL, UAFL, and DRL). Regarding UAFL and UAFL_drop, their performance surpasses FL due to optimal offloading, which minimizes local processing in favor

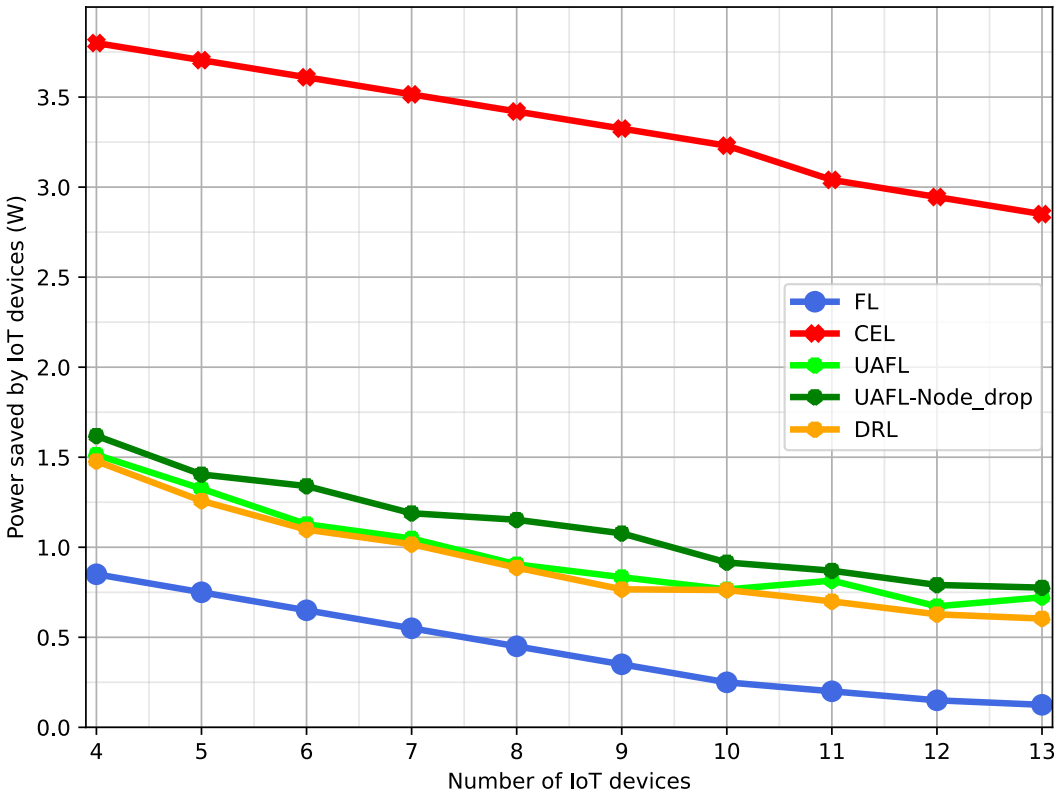

Figure 3.7: Computation power saved versus number of IoT devices with fixed power budget.

of UAV-MEC server processing. UAFL_drop demonstrates a slight marginal improvement over UAFL, as the dropped power budget from devices during iterations can be redistributed among other devices, enhancing overall performance. By optimizing the dataset offloading size, proposed schemes can offload some data to the UAV-MEC for processing, reducing the burden on individual devices. Additionally, the offloading cost is lower than that of processing locally, contributing to superior performance. While the DRL-based approach performs significantly better than CEL and FL, it falls slightly short of UAFL_drop and UAFL's performance. Though not exact, the DRL's capability to converge close to optimal solution results in slightly lower performance than UAFL.

Power Saved

Fig. 3.7 illustrates the IoT power conservation achieved with increasing IoT devices. The findings reveal consistent power savings in the case of CEL, contrasting with diminished savings observed in FL, UAFL, and UAFL_drop. Specifically, CEL exhibits the highest and relatively constant power conservation. The CEL approach offloads all the data from IoT devices to the UAV-MEC server. The only power consumed by IoT devices is on data offloading to the UAV-MEC server. The rest of the power is saved as the UAV-MEC server now does processing. This saves most of the power of IoT devices. As the number of IoT devices increases, the power saved decreases consistently. We can conclude that CEL saves the most power on IoT devices compared to other approaches. In the case of FL, as the number of IoT devices increases, the power budget for each device decreases Additionally FL consumes the available power solely for local processing since there is no offloading to the UAV-MEC server, saving all remaining power after processing the local dataset therefore, we can observe a drastic decrease in power saved in the case of FL. Meanwhile, DRL, UAFL and UAFL_drop involve both offloading and processing the local datasets, which would typically increase power consumption. However, by optimizing the dataset offloading size, these approaches achieve greater power savings than FL. UAFL_drop performs slightly better than UAFL and DRL, with additional power saved as some IoT devices are dropped before iteration, thereby making the same power available to the rest of the IoT devices. DRL-based implementation of UAFL produces results that are better than FL but slightly worse than UAFL and UAFL_drop. It produces better results than FL, using the optimal dataset offloading size. Still, it produces inferior results compared to UAFL, as its optimal dataset offloading size is slightly less optimal than that of UAFL. This difference in the optimal ratio in the DRL-based solution from UAFL is responsible for power overuse in DRL-based implementation. Overall, the results suggest that CEL outperforms FL, UAFL, and UAFL_drop and DRL in terms of power conservation at IoT devices.

The preceding results show that UAFL and UAFL_drop exhibit superior performance in

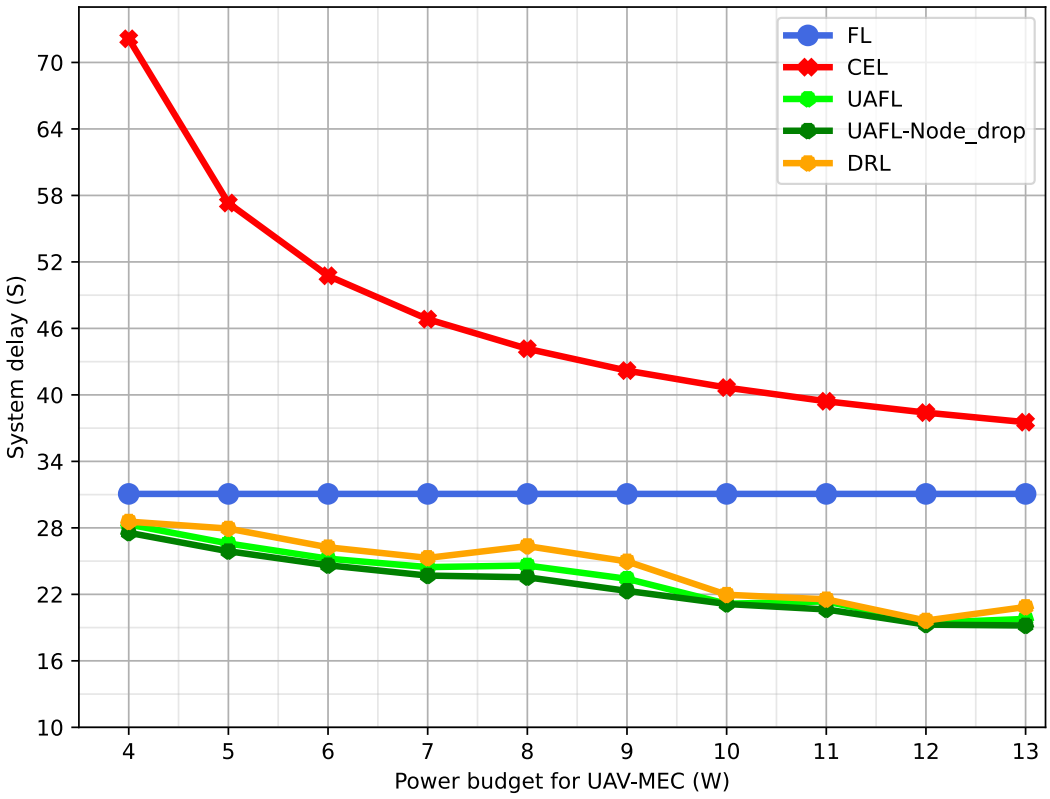

Figure 3.8: Computation power saved versus number of IoT devices with fixed power budget.

terms of delay reduction, whereas CEL surpasses these methods in conserving power across IoT devices. Despite CEL demonstrating the most effective power-saving capabilities, the consequential impact on delay stemming from offloading and processing all datasets from IoT devices on a single UAV-MEC server is substantial. This system aims to enhance both power conservation and delay, wherein DRL, UAFL and UAFL_drop emerge as the top-performing strategies.

Fig. 3.8 shows the system delay with respect to UAV power budget changes. As the power budget increases, system delay decreases across different schemes. For CEL, the delay is reduced most significantly, as the increased UAV power enhances its computation capacity, leading to better performance. In contrast, FL shows no improvement in system delay, since the processing capacity of the IoT devices remains unchanged. The limited computational

task at the UAV for aggregation also results in a negligible effect, as no data is offloaded to take advantage of the increased power budget. UAFL demonstrates a gradual decrease in system delay as the UAV power budget rises. This is because a higher power budget allows for larger dataset offloading, ensuring an optimal balance between the delay incurred from offloading and faster processing at the UAV. Due to this gradual increase in offloading, we see a slight decrease in system delay as the UAV power budget increases. UAFL_drop performs slightly better than UAFL as the number of devices is reduced for each iteration, and that results in more capacity per IoT device at UAV. DRL shows comparable results to UAFL; however, has slightly higher system delay as it finds an almost-optimal dataset offloading size. This is due to the nearly optimal dataset offloading size chosen by DRL, which is less efficient than UAFL, leading to marginally greater system delay compared to UAFL and UAFL_drop.

Based on the results above, we see that UAFL is better for reducing system delay, while CEL saves the most power on IoT devices. Although CEL saves the most power, it also adds the most delay to the system by processing all datasets on the UAV-MEC server. UAFL performs better in reducing delay compared to CEL, FL, and DRL-based methods; however, adds more run-time complexity. However, the DRL-based method performs almost as well as UAFL, with the added benefit of much lower runtime cost.

In summary, the traditional optimization approaches including UAFL_drop and UAFL performed best with minimum system delay. Out of these two, UAFL_drop slightly outperforms UAFL because it can additionally select IoT devices for each FL iteration. In mission-critical scenarios, the application of traditional optimization solutions becomes limited due to their high computation complexity. Although the DRL-based solution's performance is slightly inferior compared to the traditional optimization solution. However, its applicability in mission-critical scenarios is much more favorable due to lower runtime complexity.

3.5 Conclusion

This chapter proposed a UAFL framework to reduce the straggler's effect by offloading computation from the straggler's device on the UAV-MEC. UAFL optimized computation power, communication, and data offloading strategies by leveraging UAVs to reduce system delay significantly. We consider computation offloading while considering UAV-MEC computation power, IoT's communication and computation power, and QoS constraints. We formulate a system-wide delay optimization problem to optimize system delay by optimally adjusting offloading to UAV server on stragglers IoTs, UAV-MEC computation power, IoT's communication, and computation powers. We transform the problem to epigraph form and introduce auxiliary variables. We then solve this problem using deterministic concurrent simplex with root relaxations. We propose a second DRL-based solution to reduce the run-time complexity of the problem. Extensive simulations are done, and the results ensure the proposed scheme's effectiveness compared to traditional FL and CEL. Results depict that the proposed DRL-based scheme is also as effective as the proposed traditional solution-based scheme. From the above work, we conclude that by careful utilization of UAV-MEC and IoT resources, the performance of FL can be improved in terms of delay and power utilization while mitigating the straggler effect. This work can be further enhanced by increasing the coverage area by increasing the number of UAV-MECs in the system. Generative AI-based solution approaches can be explored to further reduce delay in the proposed system.

Chapter 4

Delay Optimization in Hierarchical UAV-aided Federated Learning for Straggling IoT Devices

4.1 Introduction

Future IoT applications, particularly in mission-critical scenarios such as forest fire surveillance, flash flood alert systems, require both high reliability and low latency. These systems often generate large volumes of real-time data in geographically dispersed wireless networks. For example, a flash flood in San Diego led to multiple fatalities and displaced over 800 individuals in 2024 [1], and very recently in July 2025 in Texas [2], highlighting the critical importance of timely information processing. Deploying cost-efficient IoT devices in remote, disaster-prone areas can facilitate early detection and response. However, maintaining a permanent communication infrastructure in such locations is economically infeasible due to their sporadic usage patterns. UAVs have emerged as a viable solution to this problem by offering flexible, on-demand wireless coverage and computational support [3,4]. Their ability to dynamically traverse large areas makes them well suited for supporting IoT deployments

in challenging environments.

Recent advances in FL have further enabled collaborative model training across decentralized heterogeneous IoT devices while preserving data privacy [5]. However, the data generated by these heterogeneous IoT devices is considerable and sensitive, which makes it challenging to transmit to edge servers [63]. In traditional collaborative models (including FL), the data is processed locally, and results are sent to the edge server. Since IoT devices have varying computational capabilities, some devices may process their data more slowly than others, creating a delay for all devices in that iteration, thus giving rise to an effect called Straggler's effect [96]. Straggler's effect occurs when a subset of IoT devices processes their local data much slower than the rest of IoT devices in an iteration, therefore introducing a delay for all the IoT devices within that iteration.

Authors of [11] proposed HFL to address the straggling issue of IoT devices by introducing a multi-layered structure, where learning tasks were distributed across different tiers of devices and servers. Hierarchy reduces the communication and computation burden on each IoT device, enhancing scalability and resilience across the entire network, especially when MEC is utilized. In case of HFL, higher layers are usually composed of UAVs with MEC computing capabilities. These UAVs with MEC capabilities can be used for offloading datasets (entirely or partially) in the FL environment and computing model parameters on those datasets using spare capacity on their end to further reduce system delay. In the case of UAV-MEC servers, data offloading becomes more critical as the UAV-MECs have limited processing capacity and power.

If the dataset records are generated at irregular intervals, they must be incorporated into FL training in a structured manner. Therefore, queuing becomes a natural choice. The integration of queuing within FL frameworks addresses the challenge of efficiently managing these datasets in the form of batches where devices show heterogeneous computational capacities and network conditions. Queuing models allow for the optimization of dataset batches, scheduling and resource allocation by considering the varying capabilities of devices

and the stochastic nature of task arrivals [73, 74].

The combination of these approaches, i.e., HFL and queuing, offers a robust solution for overcoming the challenges posed by the decentralized and dynamic nature of future wireless networks for FL. By ensuring efficient resource management, reducing communication latency, and enhancing the scalability of learning processes, these techniques are essential for realizing the full potential of FL in UAV-aided wireless networks. In this chapter, we explore the integration of these methodologies to optimize FL performance in UAV-aided wireless networks, with a focus on addressing the challenges of resource allocation, delay mitigation, and system scalability.

Building on the UAFL framework developed in Chapter 3, this chapter advances the design towards a queue-based hierarchical FL architecture tailored for UAV-assisted IoT networks. We explicitly model data arrivals and service processes via multiple queues at IoT devices, follower UAVs, and leader UAVs, and we leverage this structure to design resource allocation and offloading policies that minimize long-term system delay while respecting power and QoS constraints. By integrating hierarchical aggregation with queue-based control, the proposed framework aims to mitigate the straggler effect more systematically, ensure queue stability under stochastic arrivals, and enhance the scalability of distributed learning in mission-critical UAV-aided environments.

4.2 System Model and Problem Formulation

We consider a hierarchical wireless FL network comprising three tiers: IoT devices, follower UAVs, and a leader UAV, as Fig. 5.1 illustrates. The tier-I consists of K IoT devices with limited computational resources, each generating data in real-time. The tier-II comprises $\mathcal{M}$-1 UAV equipped with MEC capabilities. The tier-III contains a single leader UAV-MEC server responsible for aggregating model parameters and performing centralized coordination along with its MEC capabilities. Each IoT device in tier-I is connected to a follower UAV-

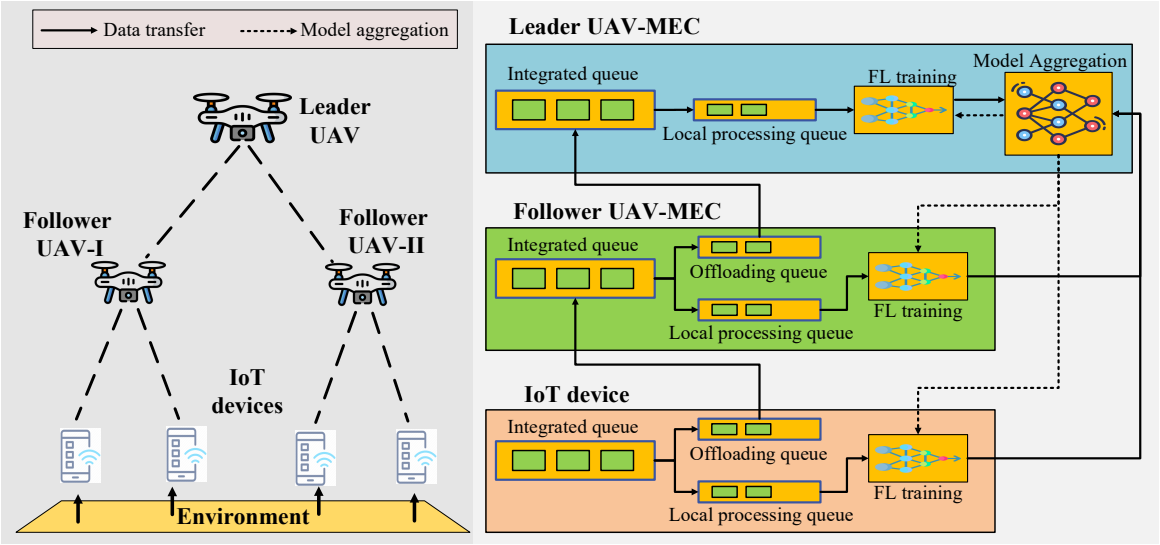

Figure 4.1: System model for UAV-aided FL platform.

MEC in tier-II, and all tier-II UAV-MEC servers are, in turn, connected to the primary UAV-MEC server in tier-III. During the *t-th* iteration, the generated dataset batches are to be processed at the *k-th* IoT device such that $E\{A_k(t)\} = \lambda_k$, where λ_k denotes the average data batch arrival rate. We assume that the arrivals are independent across both iterations and IoT devices (i.e, independent and identically distributed). The *k-th* IoT device stores dataset batches that are fully received within a window of size W_k.

Queuing model for IoT device

Each IoT device $k \in 1, 2, 3, ..., K$ possesses a local dataset C_k, consisting of input-output pairs (x_ι, y_ι), where x_ι represents the input features and y_ι the corresponding output labels. This dataset is divided into batches of size B. Each IoT device maintains four distinct types of queues to manage incoming stochastic data batches: i) **Incoming queue ($A_{k,w}(t)$):** Temporarily hold batches expected to arrive in future iterations, indexed by lookahead window $w \in 0, 1, ..., W_k - 1$; ii) **Arrival queue ($A_{k,-1}(t)$):** Holds batches arriving at the current iteration; iii) **Local Processing Queue ($Q_k^{(I,l)}(t)$):** Stores batches selected for on-device processing; and iv) **Offloading Queue ($Q_k^{(I,o)}(t)$):** Contains batches designated for offloading to a follower UAV. Figure 5.1 provides a comprehensive overview of the queueing

architecture utilized in our framework.

Incoming Queues and Arrival Queues in IoT devices

During each iteration t, the *k-th* IoT device may forward not only the data batches arriving at the current iteration (held in the arrival queue) but also batches from the incoming queues received completely. Let $\mu_{k,w}(t)$ denote the number of data batches forwarded from the incoming queue $A_{k,w}(t)$ to the arrival queue, where $w \in \{-1, 0, \ldots, W_k - 1\}$. These forwarded batches are subsequently distributed between the local processing queue and the offloading queue. We denote the number of data batches forwarded to the local processing queue and the offloading queue as $b_k^{(I,l)}(t)$ and $b_k^{(I,o)}(t)$, respectively.

$$0 \leq b_k^{(I,\beta)}(t) \leq b_{k,\max}^{(I,\beta)}, \quad \forall \beta \in \{l, o\}, \tag{4.1}$$

where $b_{k,\max}^{(I,\beta)}$ is a positive constant. As a result, we have:

$$\sum_{w=-1}^{W_i-1} \mu_{k,w}(t) = b_k^{(I,l)}(t) + b_k^{(I,o)}(t). \tag{4.2}$$

Next, we define the queuing dynamics for the various queues maintained at the IoT device. The queue $A_{k,w}(t)$ is updated at the end of each iteration, specifically when the first phase of the FL process concludes and the lookahead window advances by one iteration. The update rule is given as follows:

$$A_{k,(W_k-1)}(t+1) = A_k(t+W_k). \tag{4.3}$$

If $0 \leq w \leq W_k - 2$, then

$$A_{k,w}(t+1) = [A_{k,w+1}(t) - \mu_{k,w+1}(t)]^+, \tag{4.4}$$

where $[x]^+ = \max\{x, 0\}$ for $x \in \mathbb{R}$. In iteration $(t+1)$, the amount of data batches that will

arrive after $(W_k - 1)$ iterations is $A_k(t + W_k)$ and it remains unknown until iteration $(t + 1)$.

Regarding the arrival queue $A_{k,-1}(t)$, it records the actual backlog of IoT device k with the update equation as follows:

$$A_{k,-1}(t+1) = [A_{k,-1}(t) - \mu_{k,-1}(t)]^+ + [A_{k,0}(t) - \mu_{k,0}(t)]^+. \tag{4.5}$$

Note that $\mu_{k,-1}(t)$ represents the number of data batches distributed from the arrival queue $A_{k,-1}(t)$ during iteration t. We now define an integrated queue at the IoT device k, which captures the total backlog from both the arrival queue and all incoming queues. This integrated queue is denoted by $Q_k^{(I,\alpha)}(t) = \sum_{w=-1}^{W_k-1} A_{k,w}(t)$. Under a fully efficient service policy, the update rule for $Q_k^{(I,\alpha)}(t)$ is given by:

$$Q_k^{(I,\alpha)}(t+1) = \left[Q_k^{(I,\alpha)}(t) - \left(b_k^{(I,l)}(t) + b_k^{(I,o)}(t)\right)\right]^+ + A_k(t + W_k). \tag{4.6}$$

The input to the integrated queue $Q_k^{(I,\alpha)}(t)$ consists of data batches that were expected to arrive at IoT device k in iteration $(t + W_k)$ but have fully arrived during the current iteration t. The output of this queue comprises the data batches forwarded to the local processing queue and the offloading queue. The total number of data batches forwarded from the integrated queue in iteration t is given by $b_k^{(I,l)}(t) + b_k^{(I,o)}(t)$, representing its service capacity. However, if this capacity exceeds the current backlog size of $Q_k^{(I,\alpha)}(t)$, the actual number of forwarded batches will be limited by the available backlog, and thus the effective output will be smaller than $b_k^{(I,l)}(t) + b_k^{(I,o)}(t)$.

Offloading queues in IoT devices

At iteration t, the data batches in the offloading queue $Q_k^{(I,o)}(t)$ are transmitted to a follower UAV selected from the set $\mathcal{M}$. The transmission capacity depends on the transmit power decision $p_k^{(I,o)}(t)$, where $p_k^{(I,o)}(t)$ denotes the transmit power from the IoT device k to the follower UAV $m \in \mathcal{M}$. It is assumed that each IoT device is connected to exactly one

follower UAV. The transmit power is non-negative and is constrained by an upper bound. Specifically,

$$0 \leq p_k^{(I,o)}(t) \leq p_{k,\max}^{(I,o)}, \quad \forall\ k \in K,\ t \in [0, \infty]. \tag{4.7}$$

By modifying Shannon's capacity formula, the transmission capacity from IoT device k to follower UAV m, expressed in terms of the number of batches, is given as:

$$R_{k,m}(t) = \frac{\tau_0 \mathbb{W} \log_2 \left(1 + p_k^{(I,o)}(t) G_k / N_0 \mathbb{W}\right)}{B}, \tag{4.8}$$

where τ_0 denotes the average duration of each iteration $\mathbb{W}$ is the channel bandwidth, B represents the batch size (in bits) and G_k represents the channel gain for the *k-th* IoT device. The channel gain G_k is computed as $G_k = 10^{-\varphi_k/10}$, where φ_k denotes the path loss experienced by the *k-th* IoT device, as described in [86].

We adopt the air-to-ground channel model utilized in [87] following the 3GPP TR 36.777 specification [93]. Under this model, the path loss between the UAV-MEC and the *k-th* IoT device is given by:

$$\varphi_k = \frac{\eta_{LoS} - \eta_{NLoS}}{1 + a \exp[-b(\frac{180}{\pi}\varrho_k - a)] + 20 \log_{10}(\frac{4\pi \hat{f} d_k}{\hat{c}}) + \eta_{NLoS}}, \tag{4.9}$$

where a and b represent the ratio of unit land to buildings and the average number of buildings per unit area, respectively, while η_{LoS} and η_{NLoS} denote the excess path loss associated with line-of-sight and non-line-of-sight communication. These parameters are selected based on the characteristics of the communication environment [93]. $\varrho = \sin^{-1}(z_k/d_k)$ is the angle of IoT device k to UAV-MEC. d_k is the distance from *k-th* IoT device to UAV-MEC and z_k denotes the height of UAV-MEC from ground.

By adjusting the transmit power $p_k^{(I,o)}(t)$, we can offload different amounts of data batches from IoT device k to follower UAV m in iteration t. Accordingly, the update equation of

offloading queue $Q_k^{(I,o)}(t)$ is

$$Q_k^{(I,o)}(t+1) \leq \left[Q_k^{(I,o)}(t) - R_{k,m}(t)\right]^+ + b_k^{(I,o)}(t). \tag{4.10}$$

The inequality in the queue update reflects that the actual number of data batches arriving at the offloading queue $Q_k^{(I,o)}(t)$ may be less than $b_k^{(I,o)}(t)$. This is because $b_k^{(I,o)}(t)$ represents the number of data batches forwarded from the integrated queue $Q_k^{(I,\alpha)}(t)$ to the offloading queue, not necessarily the number of batches successfully transmitted over the wireless channel.

Queuing Model for UAVs

It is essential to note that each dataset batch is treated as the fundamental unit of processing; that is, it must either be processed entirely at the IoT device or fully offloaded to the UAV-MEC server. Follower UAV-MEC servers receive and manage offloaded dataset batches from their associated IoT devices. Each follower UAV-MEC processes a portion of these dataset batches through its local processing queue (based on their computation capability) and offloads the remaining batches to a tier-III UAV-MEC for further processing.

Each follower UAV ($m \in \mathcal{M} - 1$) maintains three types of queues: i) **Arrival queue** $Q_m^{(U,\alpha)}(t)$; ii) **Local processing queue** $Q_m^{(U,l)}(t)$; and iii) **Offloading queue** $Q_m^{(U,o)}(t)$. Similar to the IoT devices, at follower UAVs, data batches offloaded from the IoT device are first stored in the arrival queue $Q_m^{(U,\alpha)}(t)$, and then distributed to $Q_m^{(U,l)}(t)$ for local processing or to $Q_m^{(U,o)}(t)$ for further offloading. The leader UAV in tier-III follows the same queuing structure as the follower UAVs from tier-II, except that it does not maintain an offloading queue. Technically, this is modeled by setting $b_M^{U,o} = 0$.

Arrival Queues in UAVs

The arrivals at the *m-th* follower UAV consist of data batches offloaded from the subset of IoT devices ($K_m \in K$) that are connected to *m-th* UAV. These batches are first stored in the arrival queue and subsequently distributed to the local processing and offloading queues. We denote the number of data batches forwarded to the local processing queue and the offloading queue in iteration t as $b_m^{(U,l)}(t)$ and $b_m^{(U,o)}(t)$, respectively, such that

$$0 \leq b_m^{(U,\beta)}(t) \leq b_{m,\max}^{(U,\beta)}, \quad \forall \beta \in \{l, o\}, m \in \mathcal{M}, \tag{4.11}$$

where each $b_{m,\max}^{(U,\beta)}$ is a positive constant. Accordingly, arrival queue $Q_m^{(U,\alpha)}(t)$ is updated as follows:

$$Q_m^{(U,\alpha)}(t+1) \leq \left[Q_m^{(U,\alpha)}(t) - \left(b_m^{(U,l)}(t) + b_m^{(U,o)}(t)\right)\right]^+ + \sum_{k \in K_m} R_{k,m}(t). \tag{4.12}$$

where K_m represents all the IoT devices that are connected to *m-th* UAV-MEC and $R_{k,m}(t)$ is the transmission rate between the *k-th* IoT device and the *m-th* follower UAV.

Offloading queues in UAVs

For each follower UAV $m \in \mathcal{M}$, the offloading queue $Q_m^{(U,o)}(t)$ stores the data batches intended for offloading to the leader UAV in tier-III. Transmission capacity between the follower UAV and the leader UAV can be defined as:

$$D_m(t) = \frac{\tau_0 \mathbb{W} \log_2 \left(1 + p_m^{(U,o)}(t) G_m / N_0 \mathbb{W}\right)}{B}, \tag{4.13}$$

where G_m represents air-to-air channel gain calculated using works from [97], and $p_m^{(U,o)}$ represent the offloading power of UAV-mec. This $D_m(t)$ capacity depends on the current network state and is assumed to be upper bounded by a constant $D_{\max}$, i.e., $D_m(t) \leq D_{\max}$ for all m and t. Accordingly, the update equation for the offloading queue $Q_m^{(U,o)}(t)$ is given

by:

$$Q_m^{(U,o)}(t+1) \leq \left[Q_m^{(U,o)}(t) - D_m(t)\right]^+ + b_m^{(U,o)}(t). \tag{4.14}$$

Note that the actual number of data batches offloaded from follower UAV m to the leader UAV is given by $\min\{Q_m^{(U,o)}(t), D_m(t)\}$. In the case of the tier-III leader UAV, no offloading occurs, and it maintains only a local processing queue.

Local processing queues on IoT devices and UAVs

We assume that all IoT devices and UAVs can dynamically adjust their CPU frequencies in each iteration by employing dynamic voltage and frequency scaling (DVFS) techniques [98]. The computational capacity (i.e., CPU frequency) of the *k-th* IoT device can be calculated as:

$$s_k^I = \sqrt{\frac{p_k^{(I,l)}}{\nu_k^I}}, \tag{4.15}$$

where ν_k^I denotes the capacitance coefficient, and $p_k^{(I,l)}$ represents the processing power allocated to the *k-th* IoT device. Note that s_k is a non-negative and finite constant, i.e.,

$$0 \leq s_k^I \leq s_{k,\max} \forall k \in K, \tag{4.16}$$

The local processing queue on IoT device k evolves as follows:

$$Q_k^{(I,l)}(t+1) \leq \left[Q_k^{(I,l)}(t) - \frac{\tau_0 s_k^I}{L_k^I B}\right]^+ + b_k^{(I,l)}(t). \tag{4.17}$$

where L_k^I denotes the number of CPU cycles required to process 1 bit of data [99].

The computation capacity(CPU frequency) of *m-th* UAV-MEC can be written as:

$$s_m^U = \sqrt{\frac{p_m^{(U,l)}}{\nu_m^U}}, \tag{4.18}$$

where ν_m^U denotes the capacitance coefficient, and $p_m^{U,l}$ represents the power allocated for processing at the *m-th* UAV. The local processing queue for the *m-th* UAV (both follower and leader) can be expressed as:

$$Q_m^{(I,l)}(t+1) \leq \left[Q_m^{(I,l)}(t) - \frac{\tau_0 s_m^U}{L_m^I B_m}\right]^+ + b_m^{(I,l)}(t), \tag{4.19}$$

where $p_m^{U,l}$ is non-negative, finite and upper bounded by a constant $P_{m,\max}^{(U,l)}$ i.e.,

$$0 \leq p_m^{U,l} \leq P_{m,\max}^{(U,l)} \forall k \in \mathcal{M}. \tag{4.20}$$

Federated Learning

The FL process consists of three key stages. The first stage involves communication operations related to the current FL iteration. During this phase, data batches from the offloading queues of the tier-I IoTs devices are transmitted to their corresponding tier-II UAV-MEC servers. Simultaneously, tier-II UAV-MEC servers offload data from their offloading queues to the tier-III UAV-MEC server. Once the communication phase concludes, the process proceeds to the second stage, where local FL training occurs. In this stage, tier-I IoT devices train on their respective datasets gathered from their local processing queues, while tier-II and tier-III UAV-MEC servers train on data from their local queues. In the third stage, once local training is complete, the resulting model weights from both the tier-I IoTs devices and the tier-II and tier-III UAV-MEC servers are sent to the aggregation server. The UAV-MEC server in tier-III acts both as a computation server (to train on offloaded data) and an aggregation server. It combines the received model weights into a single global model. The updated global model is then broadcast back to the participating tier-I IoTs devices, thereby initiating the next global iteration.

For simplicity, the communication and model transmission delay between UAV-MEC servers is assumed to be a fixed component of the total aggregation and broadcasting delay.

In each global iteration (t) of the FL process, the *k-th* IoT device performs multiple local epochs. During each local epoch, the device iteratively processes dataset batches from its local processing queue to minimize the training loss. The loss function for the CNN model at the *k-th* IoT device can be defined as:

$$F_k(\omega_k^I) = \frac{1}{|b_k^{(I,l)}|} \sum_{\wr \in b_k^{(I,l)}} f(\wr, \omega_k^I), \tag{4.21}$$

where ω_k^I denotes the model parameters at the *k-th* IoT device, $f(\wr, \omega_k^I)$ represents the loss function evaluated on the $\wr$*-th* data record using ω_k^I weights, and $|b_k^{(I,l)}|$ denotes the total number of data samples across all batches in the local processing queue at the *k-th* IoT device.

Offloading delay is defined as the ratio between the amount of data to be offloaded and the achievable transmission bandwidth. The offloading delay for the *k-th* IoT device to *m-th* can be expressed as:

$$D_k^{(I,o)} = \frac{b_k^{(I,o)}}{R_{k,m}}, \tag{4.22}$$

We can define the average offloading delay as:

$$D^{(I,o)} = \frac{1}{K} \sum_{k \in K} D_k^{(I,o)}, \tag{4.23}$$

Each IoT device runs stochastic gradient descent [19], and the total computation $\beth_k$ at the *k-th* IoT device can be calculated as [88]:

$$\beth_k = \Upsilon b_k^{I,l} B L_k^I, \tag{4.24}$$

where Υ denotes epochs per round, $b_k^{I,l}$ denotes batches for local processing per epoch for

k-th IoT device. Computation delay at *k-th* IoT device can be written as:

$$D_k^{(I,l)} = \frac{\Upsilon b_k^{(I,l)} B L_k^I}{s_k^I}. \tag{4.25}$$

The loss function at the *m-th* UAV-MEC is defined as:

$$F(\omega_m^U) = \frac{1}{|b_m^{(U,l)}|} \sum_{\wr \in b_m^{(U,l)}} f(\wr, \omega_m^U), \tag{4.26}$$

where ω_m^U represents the model parameters of the *m-th* UAV-MEC after processing the dataset batches from its local processing queue. The term $|b_m^{(I,l)}|$ signifies the size of the dataset transferred from the integration queue to the local processing queue of the *m-th* UAV. This $b_m^{(I,l)}$ dataset consists of a subset of the offloaded batches received from the connected IoT devices, and were stored in the integration queue at the *m-th* UAV-MEC.

We can redefine the data aggregation process as follows:

$$\omega = \frac{1}{\mathfrak{R}} \left(\sum_{k \in K} |b_k^{(I,l)}| . \omega_k^I + \sum_{m \in \mathcal{M}} |b_m^{(U,l)}| . \omega_m^U \right), \tag{4.27}$$

where ω denotes the global aggregated model parameters sent back from the primary UAV-MEC aggregation server to the IoTs device. The term $R = \left(\sum_{k \in K} |b_k^{(I,l)}| + \sum_{m \in \mathcal{M}} |b_m^{(U,l)}| \right)$ represents the total number of dataset samples from the offloaded datasets that have been locally processed by the *m-th* UAV-MEC. The computation delay of *m-th* UAV-MEC can be written as:

$$D_m^{(U,l)} = \frac{\Upsilon b_m^{(U,l)} B_m L_m^U}{s_m^U}. \tag{4.28}$$

System processing delay can be defined as:

$$D^{(I,l)} = \frac{K}{K + \mathcal{M}} \{ \sum_{k \in K} D_k^{(I,l)} + \sum_{m \in \mathcal{M}} D_m^{(U,l)} \} + N D_{ab}, \tag{4.29}$$

where D_{ab} is the aggregation and broadcasting delay, N is the number of iterations required

to converge the model to a specific accuracy and can be written as [89]:

$$N = \left[\left(1 + \frac{1}{K}\right)\varrho + \frac{\kappa}{\Upsilon}\right]\frac{1}{\epsilon}, \tag{4.30}$$

where ϱ and κ are FL parameters that capture the effects of system configuration and data distribution, respectively, and ϵ represents the target training loss to be achieved. To determine these parameters, we adopt the approach proposed in [47]. Specifically, we perform curve fitting on historical FL training data to model the relationship between the number of training rounds and data characteristics, thereby estimating the values of ϱ and κ.

It is important to emphasize that, unlike our previous work [99], this chapter focuses on minimizing the average delay rather than the maximum delay. This distinction arises from the nature of our proposed framework, which jointly optimizes queue backlogs and overall system delay. Specifically, we employ the Lyapunov drift as the optimization objective, where the drift represents the squared increase in queue backlogs. A larger backlog at any device leads to a higher drift, prompting the algorithm to reduce that backlog and maintain balance across all IoT devices. By ensuring relatively uniform queue lengths among the participating IoT devices and UAV-MEC servers, our framework prevents individual nodes from experiencing excessive queuing. Consequently, this approach mitigates straggler effects and justifies the use of average delay as the optimization objective rather than the maximum per-device delay. The primary objective is to minimize system delay while ensuring queue stability and fulfilling QoS requirements.

The optimization problem can be written as:

$$\min_{\substack{b_i^{(I,\beta)},\ p_i^{(I,\beta)},\\ b_m^{(U,\beta)},\ p_m^{(U,\beta)}\\ \forall\ \beta\in\{i,o\}}} : \mathfrak{D} \triangleq (D^{(I,o)} + D^{(I,l)})$$

Subject to:

$$
\begin{aligned}
&C1: \sum_{k\in K} p_k^{(I,o)} + \sum_{k\in K} p_k^{(I,l)} \leq P_{total}^{I} \\
&C2: \sum_{m\in \mathcal{M}} p_m^{(U,o)} + \sum_{m\in \mathcal{M}} p_m^{(U,l)} \leq P_{total}^{U} \\
&C3: 0 \leq p_k^{(I,o)}, p_k^{(I,l)} \leq P_{\max}^{I}, \forall k \in K \\
&C4: P_{\min}^{I} \leq p_k^{(I,o)} + p_k^{(I,l)} \leq P_{\max}^{I}, \forall k \in K \\
&C5: P_{\min}^{U} \leq p_m^{(U,l)}, p_m^{(U,o)} \leq P_{\max}^{U} \forall m \in \mathcal{M} \\
&C6: P_{\min}^{U} \leq p_m^{(U,l)} + p_m^{(U,o)} \leq P_{\max}^{U} \forall m \in \mathcal{M} \\
&C7: \mathbb{W} log_2(1 + p_k^{(I,o)} G_k) \geq X_{\min}, \forall k \in K \\
&C8: 0 \leq b_m^{(U,\beta)}(t) \leq b_{m,\max}^{(U,\beta)}, \quad \forall \beta \in \{l, o\} \\
&C9: 0 \leq b_i^{(I,\beta)}(t) \leq b_{i,\max}^{(I,\beta)}, \quad \forall \beta \in \{l, o\} \\
&C10: \mathfrak{D} < \mathfrak{D}_{max},
\end{aligned}
\tag{4.31}
$$

where P_{total}^{I} denotes the total power budget for IoT devices, P_{total}^{U} represents the total power budget for UAVs, $P_{\max}^{I}$ represents maximum power limit for an IoT device, and $P_{\min}^{I}$ denotes minimum power limit for an IoT device. $P_{\min}^{U}$ and P_{max}^{U} represent the minimum and maximum computation power budget for UAVs, respectively. C1 and C2 limit the total power consumption of IoT devices and UAVs, respectively. C3 and C4 ensure that communication and computation power are greater than zero. C5 and C6 limit the computation power for each UAV. C7 is a QoS constraint and ensures the data rate achieved by each IoT exceeds the minimum data rate. C8 and C9 ensure that work in the offloading and processing queue is less than their maximum limits. C10 ensures that the total system delay $\mathfrak{D}$ remains below a predefined maximum ($\mathfrak{D}_{max}$), enabling efficient task execution.

4.3 Proposed Solution

We employ a Lyapunov optimization framework to solve the problem in (5.16). Lyapunov optimization is a widely used technique for dynamic control of stochastic systems, where decisions are made to stabilize queues while optimizing performance metrics such as delay or power consumption [100]. It transforms long-term time-averaged optimization problems into a sequence of per-time-slot decisions based on the current system state. This approach allows us to make real-time decisions that strike a balance between minimizing delay and maintaining system stability, without requiring future knowledge of data arrivals or channel conditions.

We transform the original problem (5.16) into a Lyapunov optimization framework and then propose a sequential quadratic programming (SQP)-based algorithm to address the resulting sub-problems. We define the long-term time-averaged expectations of system delay and total queue backlog as follows:

$$\overline{\mathfrak{D}} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\mathfrak{D}\}, \tag{4.32}$$

$$\overline{Q} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{\gamma\in\{\alpha,l,o\}} \big(\sum_{i\in K} \mathbb{E}\big\{Q_i^{I,\gamma}(t)\big\} + \sum_{m\in\mathcal{M}} \mathbb{E}\big\{Q_m^{U,\gamma}(t)\big\}\big). \tag{4.33}$$

Our goal is to reduce the long-term time-averaged expected maximum system delay, denoted by $\overline{\mathfrak{D}}$, subject to the condition that all system queues remain stable, i.e., $\overline{Q} < \infty$. Thus, the optimization problem in (5.16) can be reformulated as:

$$\begin{aligned} &\min_{b_k^I, b_m^U, p_k^{(I,o)}, p_k^{(I,l)}} \overline{\mathfrak{D}} \\ &\text{Subject to C1 - C12.} \\ &\overline{Q} < \infty. \end{aligned} \tag{4.34}$$

The Lyapunov function, which represents the system state through all queue backlogs,

is defined as:

$$L(Q(t)) \triangleq \frac{1}{2}\sum_{i\in K}\sum_{\gamma\in a,l,o}\left(Q_i^{I,\gamma}(t)\right)^2 + \frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{\gamma\in a,l,o}\left(Q_m^{U,\gamma}(t)\right)^2. \tag{4.35}$$

We define the drift-plus-penalty expression $\Delta_v L(Q(t))$ as follows:

$$\Delta v L(Q(t)) \triangleq \mathbb{E}\big[L(Q(t+1)) - L(Q(t))|Q(t)\big] + V\mathbf{E}\{P(t)|Q(t)\}. \tag{4.36}$$

The penalty function in the objective is defined as the total delay, given by $P(t) = D^{(I,o)}(t) + D^{(I,l)}(t)$. Here, V is a positive trade-off parameter which controls how much focus we want to give to delay and queue backlogs. The system delay is treated as a penalty function. Following the standard Lyapunov drift expansion in [100, Sec. 3.2, Sec. 4.2], the drift expression in (4.37) introduces a finite bound $\mathcal{C} > 0$ because all arrival and service terms in our queue updates are uniformly bounded. In particular, the arrivals $\{A_i(t+W_i), b_i^{(I,l)}(t), b_i^{(I,o)}(t), b_m^{(U,l)}(t), b_m^{(U,o)}(t)\}$ and the service processes $\{\frac{\tau_0 s_k^I}{L_k^I B}, R_i(t), \frac{\tau_0 s_m^U}{L_m^I B_m}, R_{i,m}(t), D_m(t)\}$ all have finite per-slot maxima under the system constraints, which results in constant drift term $\mathcal{C}$ analogous to the bound in [100]. According to equations (4.35), (4.6), (4.10), (4.12), (4.14), and (4.17), there exists a positive constant $\mathcal{C} > 0$ such that:

$$\begin{aligned}
&L(Q(t+1) - L(Q(t)) \leq \mathcal{C} \\
&+ \sum_{i\in K} Q_i^{(I,\alpha)}(t)\left(A_i(t+W_i) - b_i^{(I,l)}(t) - b_i^{(I,o)}(t)\right) \\
&+ \sum_{i\in K} Q_i^{(I,l)}(t)\left(b_i^{(I,l)}(t) - \frac{\tau_0 s_k^I}{L_k^I B}\right) \\
&+ \sum_{i\in K} Q_i^{(I,o)}(t)\left(b_i^{(I,o)}(t) - R_i(t)\right) \\
&+ \sum_{m\in\mathcal{M}} Q_m^{(U,\alpha)}(t)\left(\sum_{m\in\mathcal{M}} R_{i,m}(t) - b_m^{(U,o)}(t) - b_m^{(U,l)}(t)\right) \\
&+ \sum_{m\in\mathcal{M}} Q_m^{(U,l)}(t)\left(b_m^{(U,l)}(t) - \frac{\tau_0 s_m^U}{L_m^I B_m}\right)
\end{aligned} \tag{4.37}$$

$$+ \sum_{m \in \mathcal{M}} Q_m^{(U,o)}(t) \left(b_m^{(U,o)}(t) - D_m(t) \right),$$

To simplify the formulation, $D_m(t)$ is defined as an upper bound on the data rate between the follower UAV and the leader UAV. Given the high capacity of UAV-to-UAV communication links, this rate has a negligible influence on the remaining decision variables. In fact, the achievable rate on these air-to-air links is a monotonically increasing function of the UAV transmit power, while the corresponding offloading delay decreases monotonically. Since the drift-plus-penalty objective penalizes delay more strongly than the relatively small power term in the overall scheme, the optimal solution to this subproblem lies at the boundary of the transmit-power constraint. Therefore, fixing the UAV offloading power at its maximum allowable value does not alter the structure of the optimal solution. Instead, it removes a degree of freedom that would otherwise converge to the same value. Moreover, under all tested conditions, the follower–leader UAV link never becomes the bottleneck; the dominant limiting factors remain the local processing capability of the IoT devices and the IoT-to-UAV offloading rate. Hence, treating the UAV offloading power as a constant provides a near-optimal solution while significantly reducing computational complexity, without affecting the overall behavior of the proposed system. By substituting equation (4.37) into the definition of the drift-plus-penalty in (4.36), and using the fact that $\mathbb{E}[A_i(t + w_i)] = \lambda_i$, we obtain:

$$\begin{aligned}
&\Delta vL(Q(t)) \leq \mathcal{C} + V\mathbb{E}\{P(t)|Q(t)\} \\
&+ \sum_{i \in K} Q_i^{(I,\alpha)}(t)\mathbb{E}\left\{ \lambda_i - (b_i^{(I,l)}(t) + b_i^{(I,o)})|\mathbf{Q}(t) \right\} \\
&+ \sum_{i \in K} Q_i^{(I,l)}(t)\mathbb{E}\left\{ b_i^{(I,l)}(t) - \frac{\tau_0 s_k^I}{L_k^I B}|\mathbf{Q}(t) \right\} \\
&+ \sum_{i \in K} Q_i^{(I,o)}(t)\mathbb{E}\left\{ b_i^{(I,o)}(t) - R_{i,m}(t)|\mathbf{Q}(t) \right\} +
\end{aligned} \tag{4.38}$$

$$\sum_{m\in\mathcal{M}} Q_m^{(U,\alpha)}(t)\mathbb{E}\Big\{\sum_{m\in\mathcal{M}} R_{i,m}(t) - (b_m^{(U,o)}(t) + b_m^{(U,l)}(t))|\mathbf{Q}(t)\Big\}$$
$$+ \sum_{m\in\mathcal{M}} Q_m^{(U,l)}(t)\mathbb{E}\Big\{b_m^{(U,l)}(t) - \frac{\tau_0 s_m^U}{L_m^I B_m}|\mathbf{Q}(t)\Big\}$$
$$+ \sum_{m\in\mathcal{M}} Q_m^{(U,o)}(t)\mathbb{E}\Big\{b_m^{(U,o)}(t) - D_m(t)|\mathbf{Q}(t)\Big\}.$$

In the above equation, we substitute the transmission capacity ($R_{i,m}(t)$) from (5.7) and replace the penalty function $P(t)$ with delay. Practically, the power-budget of UAV-MEC is relatively large, and data to be offloaded $b_m^{U,o}$ is already optimized, The gain of performance by optimizing $p_m^{U,o}$ is very less. To reduce the complexity of the solution, we fix $p_m^{U,o}$ to its maximum value. We then rearrange the terms to obtain the following expression.

$$\begin{aligned}
&\Delta vL(Q(t)) \leq \mathcal{C} \\
&+ \sum_{i\in K} Q_i^{(I,\alpha)}(t)\mathbb{E}\big\{A_i(t+W_i)|\mathbf{Q}(t)\big\} \\
&+ \sum_{i\in K} \mathbb{E}\Big\{(Q_i^{(I,l)}(t) - Q_i^{(I,\alpha)}(t))b_i^{(I,l)}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{i\in K} \mathbb{E}\Big\{(Q_i^{(I,o)}(t) - Q_i^{(I,\alpha)}(t))b_i^{(I,o)}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{i\in K} \mathbb{E}\Big\{V\frac{\tau_0 s_k^I}{L_k^I B} - \frac{\Upsilon b_k^{(I,l)} B L_k^I}{s_k^I}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{m\in\mathcal{M}} \mathbb{E}\Big\{(Q_m^{(U,l)}(t) - Q_m^{(U,\alpha)}(t))b_i^{(U,l)}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{m\in\mathcal{M}} \mathbb{E}\Big\{(Q_m^{(U,o)}(t) - Q_m^{(U,\alpha)}(t))b_i^{(U,o)}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{m\in\mathcal{M}} \mathbb{E}\Big\{V\frac{\tau_0 s_m^U}{L_m^I B_m} - \frac{\Upsilon\sum_{k\in K} b_i^{(I,o)} B L_k^I}{s_m^U}|\mathbf{Q}(t)\Big\} \\
&+ \sum_{i\in K}\sum_{m\in\mathcal{M}} \mathbb{E}\Big\{V\tau_0\mathbb{W}log_2(1+p_i^{(I,o)}G_i/N_0\mathbb{W})/B \\
&- b_i^{(I,o)}B/\tau_0\mathbb{W}log_2(1+p_i^{(I,o)}G_i/N_0\mathbb{W})|\mathbf{Q}(t)\Big\}
\end{aligned} \tag{4.39}$$

$$- \sum_{m \in \mathcal{M}} Q_m^{(U,o)}(t) \mathbb{E}\Big\{ D_m(t) | \mathbf{Q}(t) \Big\}.$$

In (4.39), directly minimizing the expected drift-plus-penalty expression is computationally intractable. To address this, we adopt the Lyapunov optimization framework [100] and reformulate the problem into a deterministic per-slot subproblem. At each iteration t, the stochastic expectations are substituted with the instantaneous queue backlogs and system observations available at that time, resulting in a tractable deterministic problem that can be solved in real time. This transformation enables efficient online decision-making while ensuring that the time-averaged system performance asymptotically approaches the optimal solution of the original stochastic formulation.

$$\begin{aligned}
&\min_{\substack{b_i^{(I,\beta)},\ p_i^{(I,\beta)},\\ b_m^{(U,\beta)},\ p_m^{(U,\beta)}\\ \forall\ \beta \in \{i,o\}}} \sum_{i \in K} \Big(Q_i^{(I,l)}(t) - Q_i^{(I,\alpha)}(t) \Big) b_i^{(I,l)} \\
&+ \sum_{i \in K} \Big(Q_i^{(I,o)}(t) - Q_i^{(I,\alpha)}(t) \Big) b_i^{(I,o)} \\
&+ \sum_{i \in K} \Big(V \frac{\tau_0 s_k^I}{L_k^I B} - \frac{\Upsilon b_k^{(I,l)} B L_k^I}{s_k^I} \Big) \\
&+ \sum_{m \in \mathcal{M}} \Big(Q_m^{(U,l)}(t) - Q_m^{(U,\alpha)}(t) \Big) b_i^{(U,l)} \\
&+ \sum_{m \in \mathcal{M}} \Big(Q_m^{(U,o)}(t) - Q_m^{(U,\alpha)}(t) \Big) b_i^{(U,o)} \\
&+ \sum_{m \in \mathcal{M}} \Big(V \frac{\tau_0 s_m^U}{L_m^I B_m} - \frac{\Upsilon \sum_{k \in K} b_i^{(I,o)} B L_k^I}{s_m^U} \Big) \\
&+ \sum_{i \in K} \sum_{m \in \mathcal{M}} \Big(V \tau_o \mathbb{W} log_2(1 + p_i^{(I,o)} G_i / N_0 \mathbb{W}) / B \\
&- b_i^{(I,o)} \tau_o / (\mathbb{W} log_2(1 + p_i^{(I,o)} G_i / N_0 \mathbb{W})) \Big)
\end{aligned} \tag{4.40}$$

Subject to C1 - C12 and $\overline{Q} < \infty$.

The problem is divided into three subproblems. i) queue backlog scheduling: determines

$b^{I,l}, b^{I,o}, b^{U,l}$ and $b^{U,o}$, ii) local power allocation: optimizes $p^{(I,l)}, p^{(U,l)}$, and iii) offloading power allocation: determines $p^{(I,o)}$.

The first sub-problem of queue backlogs can be written as:

$$\begin{aligned}\min_{\substack{b_i^{(I,\beta)},\ b_i^{(U,\beta)} \\ \forall\ \beta\in\{i,o\}}} & \sum_{i\in K}\Big(Q_i^{(I,l)}(t)-Q_i^{(I,\alpha)}(t)\Big)b_i^{(I,l)} \\ & +\sum_{i\in K}\Big(Q_i^{(I,o)}(t)-Q_i^{(I,\alpha)}(t)\Big)b_i^{(I,o)} \\ & +\sum_{m\in\mathcal{M}}\Big((Q_m^{(U,l)}(t)-Q_m^{(U,\alpha)}(t))\Big)b_i^{(U,l)} \\ & +\sum_{m\in\mathcal{M}}\Big\{(Q_m^{(U,o)}(t)-Q_m^{(U,\alpha)}(t)\Big)b_i^{(U,o)} \\ & \text{Subject to C8 - C10 and } \overline{Q}<\infty.\end{aligned} \tag{4.41}$$

To address the first convex sub-problem, we utilize the queue backlogs present at each IoT device and UAV. When the backlog of the integration queue, $b_i^{\beta,i}$, for all $\beta \in K \cup (\mathcal{M}-1)$, exceeds that of the offloading queue $b_i^{\beta,o}$, the number of batches transferred to the offloading queue is set to its maximum value, $b_{k,\max}^{\beta,o}$. Similarly, if the backlog of the integration queue surpasses that of the local processing queue $b_i^{\beta,l}$, the batches transferred to the local queue are likewise set to their maximum value, $b_{k,\max}^{\beta,l}$. This relationship can be formally expressed as:

$$b_i^{\beta,\gamma}=\begin{cases} b_k^{\beta,\gamma}=b_{k,\max}^{\beta,\gamma} & \text{if } Q_k^{\beta,\gamma}\le Q_k^{\beta,i} \\ 0 & otherwise. \end{cases} \tag{4.42}$$

The second and third subproblems of local processing powers and offloading power for IoT devices, primary UAV and secondary UAVs are convex and non-convex, respectively and can be written as:

$$\min_{\substack{p_i^{(\gamma,l)} \\ \forall\ \gamma\in\{U,I\}}} \sum_{i\in K}\left(V\frac{\tau_0 s_k^I}{L_k^I B}-\frac{\Upsilon b_k^{(I,l)} B L_k^I}{s_k^I}\right)$$

$$+\sum_{m\in\mathcal{M}}\left(V\frac{\tau_0 s_m^U}{L_m^I B_m}-\frac{\Upsilon\sum_{k\in K} b_i^{(I,o)}BL_k^I}{s_m^U}\right) \tag{4.43}$$

Subject to C1 - C6, C10.

and,

$$\min_{p_i^{(I,\gamma)}\forall\ \gamma\in\{I,U\}}\sum_{i\in K}\sum_{m\in\mathcal{M}}\Bigg(V\mathbb{W}\tau_o log_2(1+p_i^{(I,o)}G_i/N_0\mathbb{W})/B$$
$$-b_i^{(I,o)}B/(\mathbb{W}\tau_o log_2(1+p_i^{(I,o)}G_i/N_0\mathbb{W}))\Bigg)$$
Subject to C1 - C7, C10. (4.44)

For the second and third sub-problems, i.e., power allocation for local processing $p_i^{\gamma,l}$ for all $\gamma \in K\cup\mathcal{M}$, and offloading $p_i^{\gamma,o}$ for all $\gamma \in (K\cup(\mathcal{M}-1))$. We utilize a SQP method to solve each problem individually. At each iteration, the nonlinear objective and constraints are locally approximated by quadratic and linear models, respectively. This results in a quadratic programming subproblem whose solution provides a search direction for the original nonlinear problem. The method then updates the solution using a line search or trust-region step to ensure improvement and constraint satisfaction. Intuitively, SQP iteratively improves the solution by narrowing in on the optimal point with increasingly accurate local models, making it well-suited for the smooth but non-convex structure of our power allocation subproblems [101]. The pseudocode for tackling all three sub-problems is presented in Algorithm 1.

4.4 Performance Evaluation

In this section, we present the analysis of the proposed scheme in terms of delay and queue backlog. We compare the performance of the **proposed approach** with several other

Algorithm 1 Proposed solution for FL each iteration.

Initialize
$b_k^{I,\alpha}(t), b_k^{I,o}(t), b_k^{I,l}(t) \leftarrow 0 \ \forall k \in K$
$b_k^{U,\alpha}(t), b_k^{U,o}(t), b_k^{U,l}(t) \leftarrow 0 \ \forall k \in \mathcal{M}$
$p_k^{I,o}(t), p_k^{I,l}(t) \leftarrow 0 \ \forall k \in K$
$p_k^{U,o}(t), p_k^{U,l}(t) \leftarrow 0 \ \forall k \in \mathcal{M}$
Queue decisions
for each device $k \in K \cup \mathcal{M}$ **do**
    if $k \in K$ **then**
        $\gamma \leftarrow I$
    else
        $\gamma \leftarrow U$
    end if
    if $Q_k^{\gamma,\alpha}(t) \geq Q_k^{\gamma,l}(t)$ **then**
        $b_k^{\gamma,l}(t) \leftarrow b_{k,\max}^{\gamma,l}$
    end if
    if $Q_k^{\gamma,\alpha}(t) \geq Q_k^{\gamma,o}(t)$ **then**
        $b_k^{\gamma,o}(t) \leftarrow b_{k,\max}^{\gamma,o}$
    end if
end for
SQP solvers for power allocation
for each IoT device $i \in K$ **do**
    Define the objective function:
    Minimize $p_i^{I,l}(t), \ p_i^{U,l}(t) \ \forall k \in K, \ \forall m \in \mathcal{M}$
    Subject to the constraints:
    $0 \leq p_k^{(I,l)} \leq P_{\max}^I, \forall k \in K$
    $P_{\min}^I \leq p_k^{(I,o)} + p_k^{(I,l)} \leq P_{\max}^I, \forall k \in K$
    $P_{\min}^U \leq p_m^{(U,l)} \leq P_{\max}^U, \forall m \in \mathcal{M}$
    $P_{\min}^U \leq p_m^{(U,l)} + p_m^{(U,o)} \leq P_{\max}^U, \forall m \in \mathcal{M}$
    Solve the local processing problem using SQP:
    $p_i^{I,l}(t), \ p_i^{U,l}(t) \leftarrow$ SQP_Solver(Objective, Constraints)
    Define the objective function:
    Minimize $p_i^{I,o}(t), \ p_i^{U,o}(t) \ \forall k \in K, \ \forall m \in \mathcal{M}$
    Subject to the constraints:
    $0 \leq p_k^{(I,o)} \leq P_{\max}^I, \forall k \in K$
    $P_{\min}^I \leq p_k^{(I,o)} + p_k^{(I,l)} \leq P_{\max}^I, \forall k \in K$
    $P_{\min}^U \leq p_m^{(U,o)} \leq P_{\max}^U, \forall m \in \mathcal{M}$
    $P_{\min}^U \leq p_m^{(U,l)} + p_m^{(U,o)} \leq P_{\max}^U, \forall m \in \mathcal{M}$
    Solve the offloading problem using SQP:
    $p_i^{I,o}(t), \ p_i^{U,o}(t) \leftarrow$ SQP_Solver(Objective, Constraints)
end for
Enforce scheduling decision $b(t), p(t)$

schemes in terms of delay, queue backlog, and system efficiency. These schemes include:

- **Traditional FL** [20], where no offloading is performed and all computation takes place on the IoT devices. Each IoT device performs local FL iterations using its own dataset and transmits the resulting model weights to the aggregation server i.e., the leader UAV-MEC server via follower UAV-MECs. The server aggregates model parameters from all IoT devices and returns a single global model. Each IoT device then updates its local model accordingly before proceeding to the next global iteration.

- **Offload to primary UAV**, which represents a centralized Edge-AI learning strategy where all data is offloaded to the leader UAV for processing and training the FL model.

- **One level offloading:** In this approach, only the follower UAVs are equipped with MEC capabilities, while the leader UAV is not utilized for computation. Straggler IoT devices offload their data optimally to the follower UAVs, which processes the entire dataset locally without forwarding any data to the leader UAV.

- **Random**, where offloading and local processing decisions are made randomly within system constraints.

- **UAFL**, referring to our previous work [99], which does not incorporate queueing mechanisms.

For consistency and fairness in comparison, the baseline methods adopt the following simplifications. The inter-UAV backhaul link is modeled as a fixed-capacity channel without adaptive bandwidth or power allocation, and UAV trajectories are considered fixed throughout the simulation period. These assumptions align with conventional UAV–MEC and FL baselines and help isolate the performance gains introduced by the proposed hierarchical and adaptive optimization framework.

4.4.1 Simulations Setup

We consider a hierarchical FL network comprising $K = 30$ IoT devices, 3 follower UAVs and 1 leader UAV ($N = 4$). UAV positions are assumed to remain static during the entire training process, and no mobility or trajectory optimization is performed. We consider the channel bandwidth to be 20 MHz. Power budget for each follower UAV-MECs and leader UAV is $P^{U,\text{Total}}$ is 10 Watts. At each device, the total power is partitioned into computation power, denoted by $p^{I,l}$ and $p^{U,l}$, and communication power, denoted by $p^{I,o}$ and $p^{U,o}$, for IoT devices and UAVs, respectively. We consider the communication overhead between IoT devices and UAVs for FL iteration, broadcasting weights back to each IoT device to be constant. This overhead can be considered constant as we communicate with a fixed number of IoT devices and UAV-MECs and almost a fixed communication load for each IoT device and UAV-MEC. We consider a minimum and maximum power limit for both computations and communications for each IoT device. The minimum power for an IoT device is P^{I}_{min}=0.01 W, and the maximum power is P^{I}_{max}= 0.5 W for both communication and computations [97]. For UAV, the lower limit for computation power is P^{U}_{min}= 5W, and the upper limit is P^{U}_{max}= 10 W. We consider the MNIST dataset of size 400 Mb in our simulations [94]. When the cumulative arrivals exceed the size of the original MNIST dataset, we generate additional data samples using standard augmentation techniques (geometric transformations [102], CutMix [103], and CutOut [104]). We consider the capacitance coefficient for UAV (ν^{U}_{m}) and IoT (ν^{I}_{k}) to be $1e^{-19}$ [95]. We fixed an iteration of 200ms for parameter aggregation and model broadcast. We consider α to be 1 and β as 40. We set the target training accuracy to be 98%. The results are aggregated over 10^3 Monte-Carlo simulations and include a 95% confidence interval. For the neural network structure utilized by FL, we consider a CNN with two convolutions and a linear output layer with flattening in between. Detailed simulation parameters are shown in Table 5.1.

All simulations were implemented in Python. The traditional optimization components were solved using the `Gurobi` optimizer, whereas the FL framework was implemented in

Table 4.1: Simulation parameters.

Description	**Parameter**	**Value**
IoT devices	K	30
Follower UAV-MEC servers	$M-1$	3
Leader UAV-MEC servers		1
Channel bandwidth	$\mathbb{B}$	20 MHz
Power budget for UAV-MECs	P^U_{total}	10 W
Power budget for IoT devices	P^I_{total}	10 W
Minimum power for IoT device	$P^I_{\min}$	0.01 W
Maximum power for IoT device	P^I_{max}	0.5 W
Minimum computation power for UAV-MEC server	$P^U_{\min}$	5 W
Maximum computation power for UAV-MEC server	P^U_{max}	10 W
Capacitance coefficient for UAV-MEC	ξ	$1e^{-19}$
Capacitance coefficient for IoT	ν	$1e^{-19}$
Dataset utilized for FL	MNIST	400Mb
FL parameter	α	1
FL parameter	β	40
FL training accuracy		98%
FL model	CNN	2 convolution layers and 1 linear layer
Transmission and broadcast delay	D_{ab}	200ms
Minimum data rate	$X_{\min}$	1Mb/s
Iteration time co-efficient	τ_0	4
Batch size	B	627,600 bits
samples per mini-batch (FL)	-	(32)
Noise	N_0	3.98×10^{-21} W/Hz
Carrier frequency	f_c	2.4 GHz
LoS PL exponent	α_{LoS}	2.1 dB
NLoS PL exponent	α_{NLoS}	3.5 dB

Table 4.2: Implementation details.

Parameter	**Value**	**Specification**
Language	Python	3.11.0
Traditional Optimization	Gurobi	11.0.2
Federated Learning	PyTorch	2.1.0
Compute Engine	Google Cloud Compute	current
Number of vCPU Cores	8	N1
Memory	32 Gb	RAM
Permanent Storage	500Gb	SSD
Graphics Processing Unit	16Gb	Nvidia Tesla T4

Table 4.3: Optimization algorithms computational complexity.

Method	**Per-round decision complexity**
Traditional FL	$O(1)$
Random scheduling	$O(1)$
One-level offloading (Hungarian algorithm)	$O(K^3)$
UAFL	$O(M^3)$
Proposed Approach	$O(2M + M^3) \approx O(M^3)$

`PyTorch`. The experiments were executed on the Google cloud platform (GCP) Compute Engine to ensure consistent computational performance. The second and third deterministic per-slot subproblems were solved using the sequential least squares quadratic programming (SLSQP) algorithm. SLSQP employs a quasi-Newton BFGS-type approximation of the Lagrangian Hessian, a merit-function-based backtracking line search for globalization, and an internal feasibility restoration phase for constraint handling. Finite-difference approximations were used with a step size of 10^{-8} and convergence error margin was 10^{-6}. This configuration provides a balance between computational efficiency and numerical stability, ensuring reliable convergence of the per-slot optimization. The details of the simulation environment are summarized in Table 4.2, and the computational complexities of the baseline algorithms used for comparison are presented in Table 4.3.

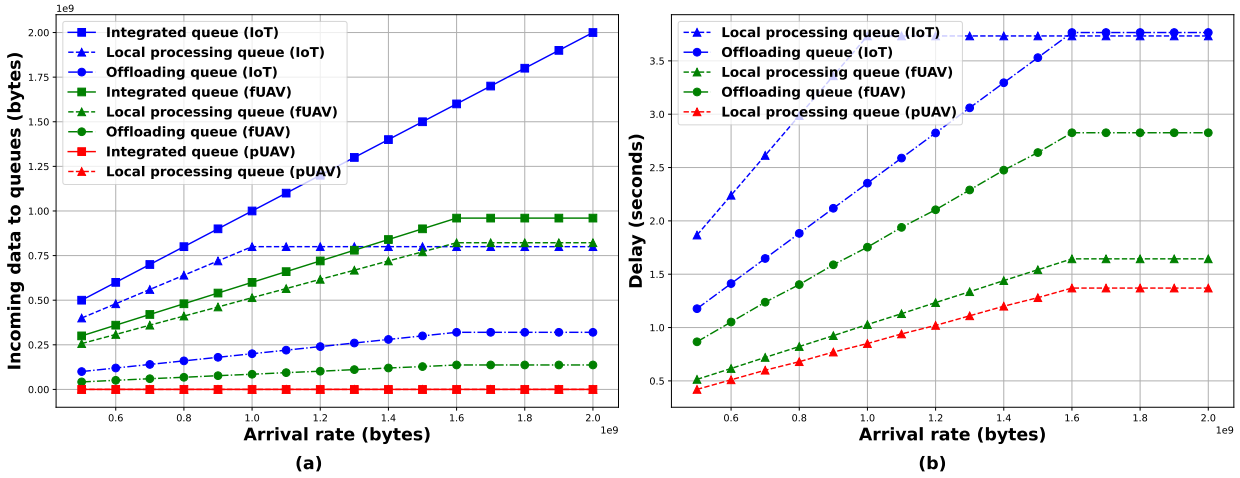

Figure 4.2: Impact of arrival rates on system performance for $K = 30$ IoT devices and $M = 4$ UAVs (3 followers and 1 leader UAVs): (a) incoming data in queue and (b) delay.

4.4.2 Simulation Results

Fig. 4.2 illustrates how varying arrival rates affect both delay and the distribution of data across different queues within the proposed system. Fig. 4.2(a) demonstrates the delays encountered in processing data within various queues. The ”local processing queue (IoT)” delay represents the average time required to process data locally at the IoT devices during each FL iteration. The “offloading queue (IoT)” refers to the average delay incurred when data is offloaded from IoT devices to their connected follower UAVs. Similarly, the “local processing queue (fUAV)” denotes the average delay associated with processing data at the follower UAVs that the IoT devices have offloaded, whereas the “Offloading delay (fUAV)” captures the time taken further to offload data from follower UAVs to the leader UAV. The “local processing queue (pUAV)” signifies the delay associated with processing data at the leader UAV’s local queue. Among all the queues, the most significant delay is observed in the local processing queue at the IoT devices. This is primarily because processing data locally is often the most cost-effective strategy under optimal conditions. However, as the arrival rate increases, more data moves from the integration queue to the local processing queue, resulting in increased delays. At the same time, the rise in the offloading queue contributes to further delays as more data is sent to the UAVs. As the offloaded traffic

increases, follower UAVs experience heavier loads in their integration queues, from which data is then distributed to their respective local processing and offloading queues. This leads to increased processing and offloading delays at the UAVs. When the load surpasses the follower UAVs' capacity, data is further offloaded to the leader UAV's integration queue, which subsequently forwards it to its local processing queue. The results indicate that the local processing queue at the IoT devices becomes saturated when the arrival rate exceeds 1×10^9 bytes, while the offloading queue saturates around 1.6×10^9 bytes.

Fig. 4.2(b) depicts how data is distributed across different queues as the arrival rate increases. As the incoming rate rises, the size of the IoT integration queue also grows, serving as the initial buffer for environmental data batches. In each iteration, the data is forwarded from the integration queue to the local processing and offloading queues using the optimal decision variables generated by our algorithm. This results in a steady increase in the size of the local processing queue as long as it remains advantageous to process data locally. Once local processing becomes suboptimal, the growth of the local queue levels off, while the offloading queue continues to expand, resulting in more data being transmitted to the connected follower UAVs. As the follower UAVs receive additional data, their integration queue sizes also increase accordingly. Based on the optimal decisions made by the proposed algorithm, this data is further distributed between the UAVs' local processing and offloading queues. When the follower UAVs can no longer efficiently handle the additional processing, offloading to the leader UAV increases. However, due to the relatively lower volume of data arriving at the leader UAV compared to its processing capacity, the sizes of its integration and local processing queues remain minimal.

A key insight from Fig. 4.2 is that saturation at the IoT level occurs earlier because IoT devices have significantly lower computational capability and energy budgets than to UAV-MEC nodes. Therefore, even moderate increases in the input arrival rate immediately result in queue buildup at the IoT devices, indicating that the local processing path becomes the bottleneck under high-load. The relatively smoother increase in follower UAV delays

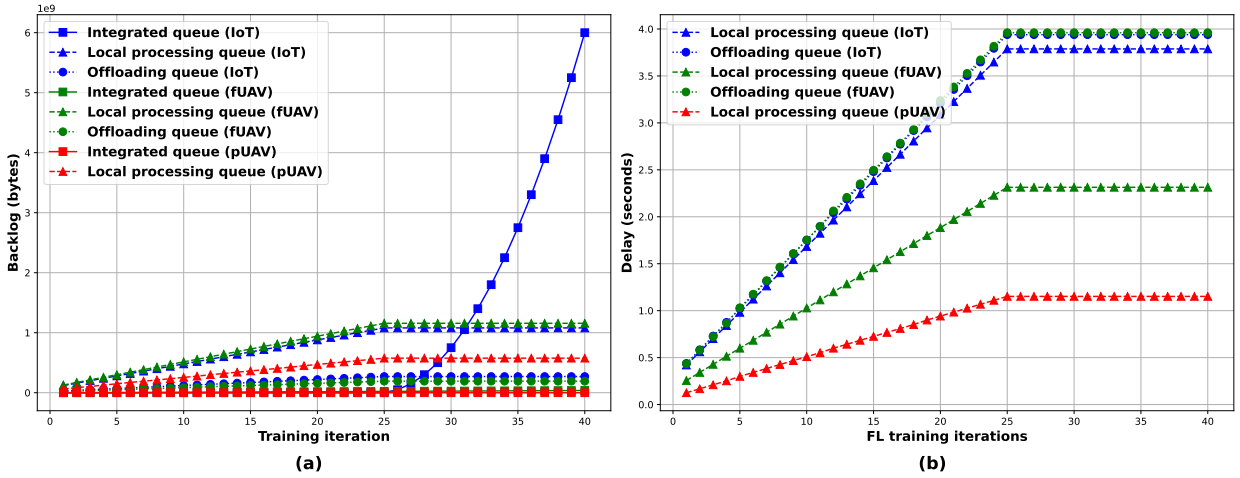

Figure 4.3: Impact of the number of training iterations on system performance under a fixed iteration time constraint for $K = 30$ IoT devices and $M = 4$ UAVs (3 followers and 1 leader UAVs): (a) backlog and (b) delay.

highlights that UAV-MEC nodes can temporarily absorb the excess load due to their higher compute capacity; however, once their local queue begins to fill, a cascading effect occurs where delays rise simultaneously across local and offloading queues. This queue coupling effect physically reflects the hierarchical nature of the HFL architecture, where congestion at lower tiers propagates upward as the system tries to maintain feasible processing rates.

Fig. 4.3 illustrates the system's behavior under constrained conditions, where the data arrival rate exceeds the total processing capacity, and a strict four-second limit is imposed per FL iteration. Fig. 4.3(a) shows the evolution of queue backlogs across different components as the training iterations progress. The results indicate a steady increase in backlog due to the growing volume of data processed within the system. The most significant backlog is observed in the local processing queues of the IoT devices and the follower UAVs, except the IoT integration queue. This is because local processing is the quickest available method, and the system prioritizes sending data to these queues. Consequently, the local processing queues become saturated and stabilize after approximately 25 iterations, once the processing time cap is reached. Following this point, there is also a noticeable increase in the backlog at the leader UAV's local processing queue, as it begins to handle more offloaded data.

The offloading queues at the IoT devices and follower UAVs also exhibit a growing backlog, although it remains significantly lower than that of the local processing queues. This is because offloading takes longer than local processing, leading the system to allocate more data to local processing whenever possible. However, once the local and offloading queues at the IoT devices reach their capacity under the fixed delay constraint, any additional incoming data accumulates in the integration queue, which shows a sharp increase in backlog. In contrast, the integration and offloading queues at the follower and leader UAVs do not exhibit the same rapid growth, as only a limited amount of data can be offloaded from the IoT devices before the delay cap is reached, thereby preventing further buildup in those queues.

Fig. 4.3(b) presents the corresponding delays for each queue type throughout the FL iterations. Initially, the primary contributors to system delay are the local processing queues at the IoT devices and follower UAVs. This reflects the system's preference for swift, local computation in the early stages. As training progresses, the delays in these queues stabilize after approximately the 25th iteration, aligning with the saturation pattern observed in the backlog plots. The delay at the leader UAV's local processing queue also increases during this period. However, it does not reach the same magnitude, as the offloading queues at the follower UAVs reach their limits before the processing capacity of the leader UAV is fully utilized.

An important physical insight from Fig. 4.3 is that the fixed per-iteration delay constraint changes the system's optimal behavior compared to the in-capacity case (Fig. 4.2). Under this constraint, the scheduler prioritizes any computation that can be completed within the four-second deadline, forcing more data into local processing queues and leaving less time for communication-dependent offloading. This shifts the system from an offloading-assisted configuration to a locally dominated setting, which explains why local queues saturate first and remain the primary bottleneck. The sharp growth of the IoT integration queue indicates that once both local processing and offloading paths hit the time constraint, the system

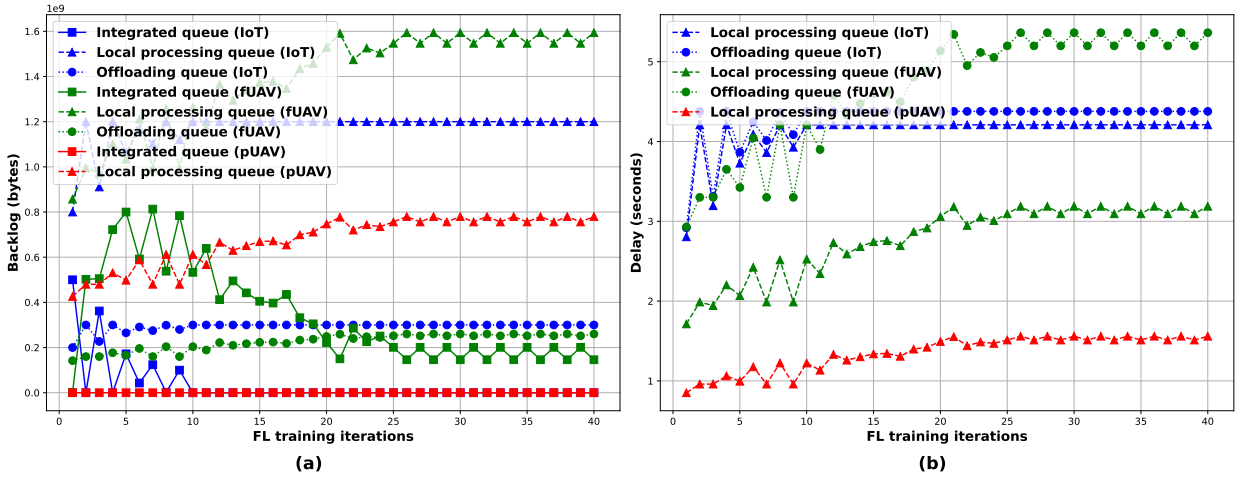

Figure 4.4: System performance under queue size constraints as a function of training iterations for $K = 30$ IoT devices and $M = 4$ UAVs (3 followers and 1 leader UAVs): (a) backlog and (b) delay.

effectively loses the ability to process incoming data. This behavior exposes an inherent limitation of deadline-restricted FL systems, i.e., when the arrival rate exceeds the "deadline-constrained service rate," backlogs will accumulate even if computational resources in higher tiers remain underutilized. The follower and leader UAVs also highlight that high-tier nodes do not naturally become overloaded under a strict delay cap. Because transferring data from lower tiers consumes communication time, the UAVs are unable to receive enough data within the time window to fill their queues, even though they have available processing capacity. Thus, the bottleneck shifts downward in the hierarchy, demonstrating that communication delays, rather than compute shortages, dominate system performance in deadline-constrained scenarios.

Fig. 4.4 illustrates the system behavior under queue size limitations. It illustrates how the system responds when the arrival rate exceeds the total processing capacity, resulting in queues filling up. Fig. 4.4(a) depicts the evolution of queue backlogs over time. Initially, IoT devices experience a significant increase in backlog due to high data arrivals. However, due to the imposed queue size limits and the proposed optimization strategy, the backlog levels stabilize by approximately the *11-th* iteration. At the same time, the integration queue at

the follower UAVs shows a gradual increase, reflecting a continuous inflow of data and active allocation between local processing and offloading queues. This indicates a well-balanced distribution of workload within the UAV-MEC system. The leader UAV's integration queue remains largely underutilized, while its local processing queue shows limited activity. This suggests that the amount of data offloaded to the leader UAV remains well within its processing capabilities. Although the trends in queue backlogs may appear irregular, they reflect the adaptive behavior of the proposed optimization algorithm, which dynamically responds to varying arrival rates and system conditions. This result demonstrates the proposed framework's ability to manage queue sizes efficiently, dynamically allocating tasks between local processing and offloading in real-time to ensure system stability and performance.

Fig. 4.4(b) presents the corresponding delay metrics associated with each queue type. In the early iterations, system delay is primarily linked to the local processing and offloading queues of the IoT devices, which initially handle the majority of the workload. As these queues stabilize, the focus of the delay gradually shifts to the follower UAVs, where both the local processing and offloading queues start to accumulate delay. Due to the finite processing capacity of these UAVs, the offloading queue at the follower UAVs ultimately becomes the predominant source of delay. In contrast, the leader UAV's local processing queue has minimal impact on the overall delay because its workload is relatively low compared to its processing capacity, allowing it to manage incoming data efficiently with negligible queuing delay.

Fig. 4.5 presents a comparative analysis of four FL strategies, focusing on the total queue backlog and system delay across training iterations. Fig. 4.5(a) shows the total queue backlog experienced during FL training for each strategy. The results demonstrate that the proposed approach achieves the lowest overall backlog, highlighting its effectiveness in efficiently managing computational and communication resources across the system. Specifically, it reduces the backlog by approximately 15% compared to the next best approach. In contrast, the "offload to primary UAV" strategy results in the highest backlog. This is

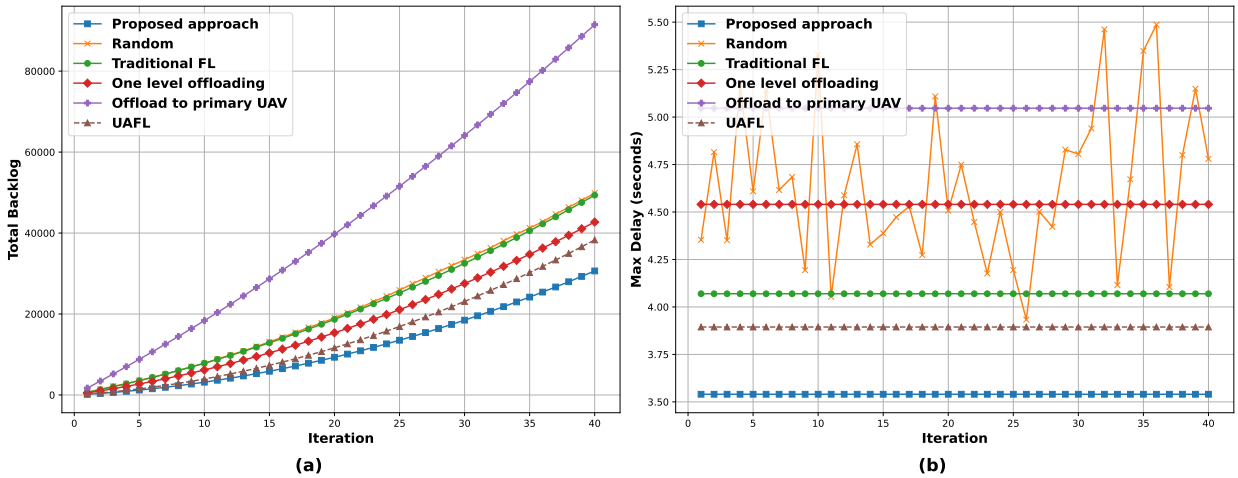

Figure 4.5: Impact of different approaches over training iterations on system performance for $K = 30$ IoT devices and $M = 4$ UAVs (3 followers and 1 leader UAVs): (a) total backlog and (b) maximum system delay.

mainly due to all data being offloaded directly to the primary UAV, which causes excessive communication delays and congestion in offloading queues, especially at the IoT devices and follower UAVs. As a result, local processing resources remain underutilized, worsening the backlog issue. Additionally, the end-to-end training process becomes significantly slower due to the long-range transmission overheads. The "random allocation" strategy, which lacks optimization, performs similarly to the "traditional FL" strategy, where IoT devices rely entirely on local computation. Both approaches exhibit moderate backlog levels but fail to capitalize on the hierarchical processing capabilities of the UAV network. The second-best performance is observed with the "offload to follower UAVs only" strategy, which effectively prevents overwhelming the leader UAV by limiting offloading to tier-II UAVs. However, our proposed solution outperforms all baseline strategies by intelligently balancing the workload across IoT devices, follower UAVs, and the primary UAV, resulting in significant reductions in total queue backlog.

Fig. 4.5(b) illustrates the system delay under the same four FL strategies. Consistent with the backlog trends, the proposed approach achieves the lowest overall delay throughout the training process, reducing the delay by approximately 5% compared to the next

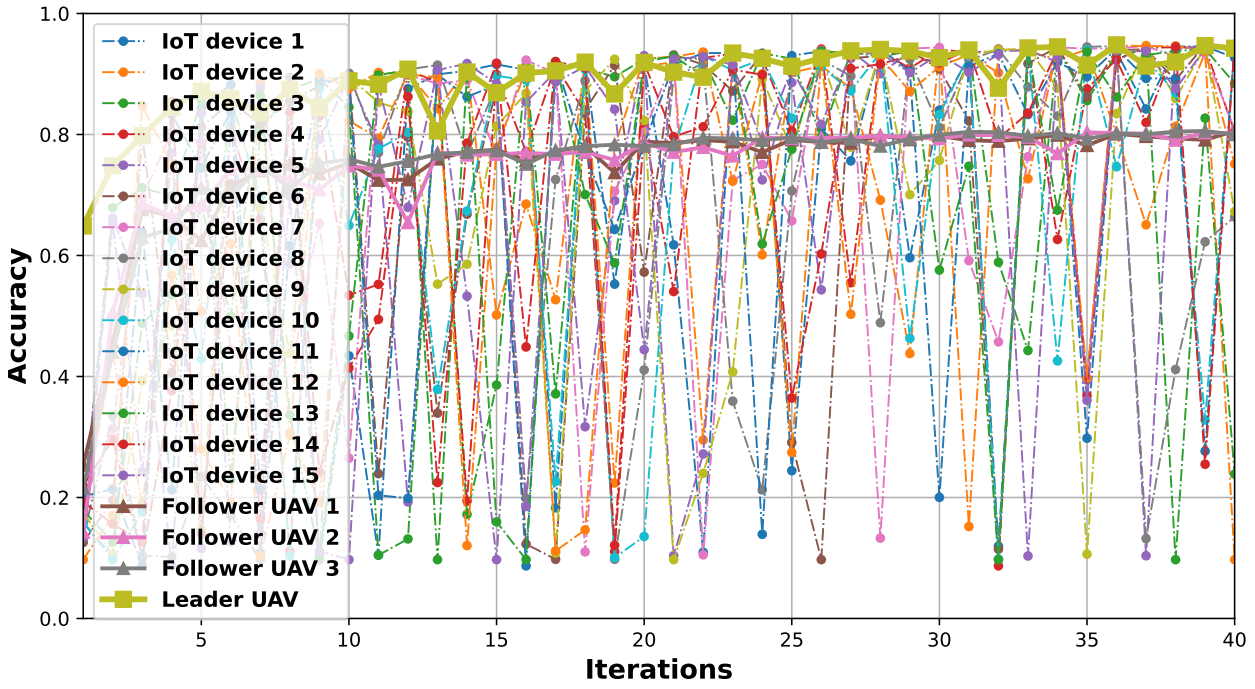

Figure 4.6: FL accuracy for leader UAV, follower UAVs, and IoT devices.

best approach. This can be attributed to its ability to evenly distribute data across the available processing queues evenly, thereby avoiding bottlenecks and maintaining system responsiveness. While queue backlogs remain relatively stable across iterations (as shown in Fig. 4.5(a)), the per-iteration delay fluctuates due to the stochastic nature of data arrivals. Despite these variations, the proposed method consistently maintains a lower delay profile. As expected, the highest delay is associated with the offload to primary UAV" strategy due to excessive transmission and queuing delays at the primary UAV. The "traditional FL" approach demonstrates acceptable performance but lacks the coordination benefits of distributed processing. These results confirm that strategic offloading and efficient resource utilization, achieved through our optimization framework, are crucial for minimizing both delay and backlog in hierarchical UAV-assisted FL systems.

ig. 4.6 demonstrates that the proposed framework achieves the desired model accuracy. The results indicate that local IoT accuracies fluctuate due to random batch arrivals and skewed data distributions. These factors cause frequent fluctuations in local model accu-

racy. However, global model accuracy remains stable due to UAV-MEC aggregation. As observed in the graph, the follower UAVs exhibits relatively stable accuracy compared to the IoT devices. This is because each follower UAV receives offloaded data from multiple IoT devices, resulting in a more inherently i.i.d. dataset than that of any individual IoT device. Although the follower UAVs achieves more stable performance, its accuracy remains lower than that of the global model. This is expected, as the global model aggregates updates from all IoT devices, making its training data even more i.i.d. and diverse than that of the follower UAVs. The purpose of integrating queue structures and hierarchical FL was to ensure smoother training by minimizing fluctuations in iteration time caused by random data arrivals. These results show that the proposed multi-layered architecture preserves convergence while providing more consistent iteration durations.

4.5 Conclusion

This chapter presents a queueing-based hierarchical FL framework designed to address the straggler effect and minimize overall system delay. The proposed framework exploits a hierarchical UAV-MEC architecture, wherein straggler IoT devices offload segments of their local datasets to nearby follower or leader UAV-MEC servers with available computational capacity. By employing multiple queues at each network entity for data integration, local processing, and offloading, we model system dynamics with greater authenticity. We formulated a system-wide delay minimization problem that considers communication power, computational power at both IoT and UAV-MEC devices, offloading decisions, and QoS constraints. This problem is addressed using Lyapunov optimization to ensure delay reduction while adhering to system constraints. Extensive simulations were conducted to assess the performance of the proposed framework in comparison to traditional FL, UAFL [99], total offloading strategies, one-level offloading, and random configurations. The results indicate that the proposed framework significantly outperforms all baseline methods in terms

of minimizing delay and maintaining queue stability. Our study concludes that the effective utilization of the computational and communication capabilities of both UAV-MEC servers and IoT devices can significantly enhance FL performance, particularly in terms of delay and power efficiency. It is worth noting that this chapter does not consider UAV propulsion energy, dynamic selection, or peer-to-peer communication between UAV-MEC servers during the aggregation phase; this aspect will be explored in future work. Additionally, future directions include extending the framework to support larger UAV swarms for broader area coverage and integrating generative AI-based techniques to reduce system delay further and enhance adaptability.

Chapter 5

Delay-Optimized Hierarchical UAV-MEC Federated Learning Using Federated GANs for IoT Networks

5.1 Introduction

As FL scales toward increasingly complex and distributed IoT deployments, new challenges emerge that are not fully addressed by conventional hierarchical or UAV-assisted solutions. In particular, highly irregular data arrivals, severe Non-IID distributions, and limited computational resources at UAV-MEC servers can significantly degrade both convergence speed and model accuracy. These issues are further amplified in mission-critical environments, where timely learning updates and reliable decision-making are essential. To overcome these limitations, this chapter explores a GAN-augmented hierarchical UAV-assisted FL framework that introduces queue-aware data handling and distributed synthetic data generation at UAVs. This integration enables more efficient processing of uneven data arrivals, enhances dataset diversity through federated GAN training, and improves overall learning accuracy while maintaining low system delay. The following section presents the system model that

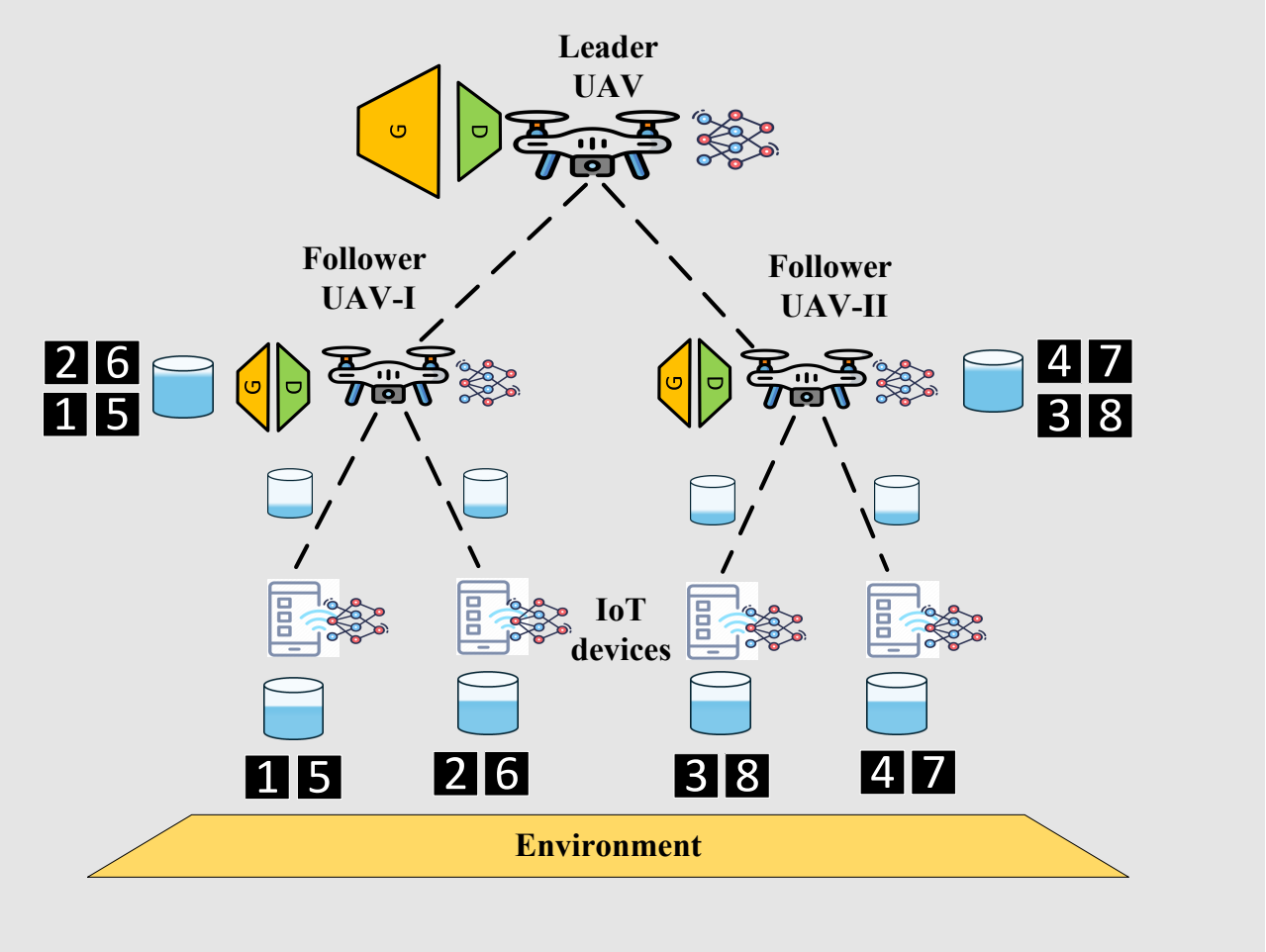

Figure 5.1: System model for UAV-aided FL platform.

supports this framework, including the queue architecture, UAV roles, and collaborative GAN–FL training workflow.

5.2 System Model and Problem Formulation

5.2.1 System Overview

We consider a wireless HFL environment consisting of K IoT devices and $\mathcal{M}$ UAV MEC servers. All IoT devices operate in tier I, while tier II comprises M–1 follower UAV-MEC servers, and tier III contains a single primary UAV-MEC server. Each IoT device in tier

I communicates with a follower UAV-MEC in tier II, and all tier II UAV-MEC servers are connected to the primary UAV-MEC server in tier III, as illustrated in Fig. 5.1. Each IoT device continuously collects environmental data, which is stored in its incoming queue. This queued data is processed locally or forwarded to an offloading queue, depending on current network and computational conditions. UAVs, acting as MEC servers, have spare processing and power resources that can assist IoT devices by handling a portion of their computational workload through data offloading to tier II UAV-MECs.

During each FL iteration, IoT devices generate new data samples that enter their respective incoming queues. The datasets at these IoT devices are assumed to be Non-IID, meaning each device holds data related to a limited subset of global classes. To address this imbalance and reduce local iteration time, IoT devices offload part of their data, optimized for minimizing delay, to UAV-MECs in tier II. Each UAV-MEC server aggregates the offloaded datasets from multiple IoT devices to form a composite local dataset, which is stored in its local database along with previously received data. Once data transfer for the iteration is complete, both IoT devices and UAV-MEC servers perform local model training to generate model parameters. These parameters are then transmitted to the primary UAV-MEC server in tier III, which serves as the global aggregator and computes the global model weights. The updated weights are subsequently distributed back to all IoT devices and UAV-MEC servers to initiate the next iteration.

After a predefined number of FL iterations (for example, 10 rounds), the system temporarily pauses the conventional FL process and begins a hierarchical FL-GAN training phase. In this phase, follower UAV-MEC servers in tier II use the accumulated data in their incoming queues and local databases to train local Generator-Discriminator (G-D) pairs. After several local iterations, each follower UAV-MEC forwards its generator network weights to the leader UAV-MEC server in tier III. The leader UAV-MEC aggregates the received generators using a weighted averaging approach to obtain a global generator. Based on this global generator, the leader also trains its own discriminator to refine the generative

model. Once the global discriminator converges, the leader UAV-MEC broadcasts the updated global generator to all follower UAV-MECs for further training in subsequent global GAN cycles.

Upon convergence of the federated GAN phase, each UAV-MEC server retains a well-trained GAN model capable of generating synthetic samples that closely resemble the true data distribution. These GAN-generated datasets are then used to augment the original FL training data, particularly by generating underrepresented classes within the Non-IID data distribution. This process enhances overall data diversity and model generalization in subsequent hierarchical FL rounds.

5.2.2 Federated GAN Training

The GAN consists of two neural networks: a generator $\mathfrak{G}$ and a discriminator $\mathfrak{D}$. The generator $\mathfrak{G}$ aims to approximate the underlying data distribution p_{data}, while the discriminator $\mathfrak{D}$ estimates the probability that a given sample originates from the real dataset rather than being synthetically generated. The parameters of both models, denoted by W_g and W_d for $\mathfrak{G}$ and $\mathfrak{D}$, are updated iteratively in an adversarial manner. Auxiliary class labels y are provided to both networks to enable conditional generation and discrimination, thereby enhancing the diversity and fidelity of generated samples.

Let $x \sim p_{\text{data}}(x)$ denote a real sample drawn from the true data distribution, and $z \sim p_z(z)$ represent a latent variable sampled from a prior distribution (e.g., Gaussian). The standard GAN objective is formulated as a two-player minimax game with the value function $V(\mathfrak{D}, \mathfrak{G})$:

$$\min_{\mathfrak{G}} \max_{\mathfrak{D}} \mathfrak{V}(\mathfrak{D}, \mathfrak{G}) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log \mathfrak{D}(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathfrak{D}(\mathfrak{G}(z|y)))]. \tag{5.1}$$

To enable FL-based training of GANs, the training process is divided into *local* and *global* phases. Instead of sharing raw data, each user transmits its locally trained generator and

the corresponding learned data distribution. During the global aggregation phase, the FL-GAN enables fine-grained data augmentation, effectively transforming non-independent and identical data (IID) local data into more IID-like samples, thereby facilitating unbiased global learning. Let $p_{\text{data}}(x) = \{p_1(x), \ldots, p_n(x)\}$ denote the set of non-IID local data distributions across U users. Each user U_i trains an independent local GAN consisting of $(\mathfrak{G}_i, \mathfrak{D}_i)$.

Local Objective

Given a local dataset X_i drawn from $p_i(x)$, each user U_i optimizes $(\mathfrak{G}_i, \mathfrak{D}_i)$ by minimizing the following local adversarial loss:

$$\mathcal{L}_{U_i} = \min_{\mathfrak{G}_i} \max_{\mathfrak{D}_i} \mathfrak{V}_{U_i}(\mathfrak{D}_i, \mathfrak{G}_i) = \mathbb{E}_{x \sim p_i(x)}[\log \mathfrak{D}_i(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - \mathfrak{D}_i(\mathfrak{G}_i(z|y)))]. \quad (5.2)$$

The value function $\mathfrak{V}_{U_i}(\mathfrak{D}_i, \mathfrak{G}_i)$ consists of two sub-objectives: (1) the discriminator $\mathfrak{D}_i$ is trained to maximize $\mathfrak{V}_{U_i}$ by distinguishing real and generated samples, while (2) the generator G_i minimizes $\mathfrak{V}_{U_i}$ to deceive $\mathfrak{D}_i$.

Global Objective

Since each pair $(\mathfrak{D}_i, \mathfrak{G}_i)$ is optimized using its local distribution $p_i(x)$, the corresponding generators $\mathfrak{G}_i^*$ may not fully capture the global data distribution. Hence, the global objective $\mathcal{L}_\Omega$ aims to train a unified federated GAN model (D_Ω, G_Ω) that integrates knowledge across all users:

$$\mathcal{L}_\Omega = \min_{\mathfrak{G}_\Omega} \max_{\mathfrak{D}_\Omega} V_\Omega(\mathfrak{V}_{U_1}(\mathfrak{D}_1, \mathfrak{G}_1), \ldots, \mathfrak{V}_{U_n}(\mathfrak{D}_n, \mathfrak{G}_n)). \quad (5.3)$$

However, since the locally trained generators $\mathfrak{G}_i^*$ and discriminators $\mathfrak{D}_i^*$ lack global context, they may fail to generate samples consistent with the true data distribution $p_{\text{data}}(x)$. To overcome this limitation, all local generators are aggregated to form a global generator $\mathfrak{G}_\Omega$, enabling a more representative approximation of $p_g(x) \approx p_{\text{data}}(x)$. Accordingly, the

global loss is reformulated as:

$$\mathcal{L}_{\Omega} = \min_{\mathfrak{G}_{\Omega}} \max_{\mathfrak{D}_{\Omega}} \mathfrak{V}_{\Omega}(\mathfrak{V}_{U_1}(\mathfrak{D}_{\Omega}, \mathfrak{G}_{\Omega}), \ldots, \mathfrak{V}_{U_n}(D_{\Omega}, \mathfrak{G}_{\Omega})). \tag{5.4}$$

This reformulation allows the aggregated model to learn a unified and unbiased representation of the global data distribution, thereby enhancing convergence stability and generation quality across all participating users.

5.2.3 Federated Learning Mechanism

The FL process comprises three main stages: (i) communication among participating entities to initiate and coordinate each FL iteration; (ii) local model training, performed independently on both the IoT devices and the UAV–MEC servers; and (iii) global aggregation, where model updates are consolidated to generate a refined global model. During each iteration, IoT devices use their locally stored datasets, while the UAV–MEC servers process the offloaded data received from multiple IoT devices stored in their local processing queue. After completing local training, the resulting model weights from the IoT devices and UAV–MEC servers are transmitted to the aggregation server (primary UAV–MEC). The aggregation server then combines these weights to produce an updated global model, which is subsequently broadcast back to all IoT devices and UAV–MEC servers to start the next FL round.

Each global FL iteration consists of one or more local training epochs executed at the k-th IoT device. Within each epoch, the k-th device performs multiple iterations over its local dataset batches to minimize the training loss. The local loss function for the k-th IoT device is defined as:

$$F_k(\omega_k^I) = \frac{1}{|b_k^{(I,l)}|} \sum_{i \in b_k^{(I,l)}} f_i(\omega_k^I), \tag{5.5}$$

where ω_k^I denotes the model parameters at the k-th IoT device, f_i represents the loss function for the i-th data sample, and $|b_k^{(I,l)}|$ corresponds to the number of data records in all batches processed locally by the k-th device. This formulation captures the local optimization objective at each IoT device before the aggregation phase, forming the foundation of the hierarchical FL process.

The offloading delay is determined by the ratio between the amount of data to be offloaded and the achievable transmission rate over the wireless channel. For the k-th IoT device, the offloading delay is defined as:

$$D_k^{(I,o)} = \frac{B_k^I \, b_k^{(I,o)}}{R_k^{(I,o)}}, \tag{5.6}$$

where $b_k^{(I,o)}$ corresponds to the number of batches selected for offloading from the k-th IoT device to its serving UAV–MEC and $R_k^{(I,o)}$ represents the data transmission rate between the k-th IoT device and its associated UAV–MEC [99] and can be defined as:

$$R_k^{I,o}(t) = \frac{\tau_0 \mathbb{C} \log_2 \left(1 + p_k^{(I,o)}(t) G_k / N_0 \mathbb{C}\right)}{B}, \tag{5.7}$$

where τ_0 denotes the average duration of each iteration $\mathbb{C}$ is the channel bandwidth, B represents the batch size (in bits) and G_k represents the channel gain for the *k-th* IoT device. The channel gain G_k is computed as $G_k = 10^{-\varphi_k/10}$, where φ_k denotes the path loss experienced by the *k-th* IoT device, following the formulation in [86]. For path-loss characterization, we employ the air-to-ground model used in [99]. B_k^I represents the total data volume per batch, computed as the number of data records multiplied by the size (in bits) per record. The average offloading delay across all IoT devices is expressed as:

$$\overline{D}^{(I,o)} = \frac{1}{K} \sum_{k=1}^{K} D_k^{(I,o)}. \tag{5.8}$$

This formulation quantifies the average transmission latency incurred during the offloading phase and serves as a key metric in evaluating The efficiency of the hierarchical

UAV–MEC FL system. Each IoT device employs stochastic gradient descent (SGD) for local model training. The total computational load $\beth_k$ at the k-th IoT device is expressed as:

$$\beth_k = \Upsilon \, b_k^{(I,l)} \, B_k^I \, L_k^I, \tag{5.9}$$

where Υ denotes the number of local epochs per FL round, $b_k^{(I,l)}$ represents the number of batches per epoch, and L_k^I indicates the number of CPU cycles required to process one bit of data at the k-th IoT device [88]. The local computation delay $D_k^{(I,l)}$ for the k-th IoT device is therefore given by:

$$D_k^{(I,l)} = \frac{\Upsilon \, b_k^{(I,l)} \, B_k^I \, L_k^I}{s_k^I}, \tag{5.10}$$

where $s_k^I = \sqrt{p_k^{(I,l)}/\nu_k^I}$ denotes the computational speed of the k-th IoT device [98]. ν_k^I denotes the capacitance coefficient, and $p_k^{(I,l)}$ represents the processing power allocated to the *k-th* IoT device. This formulation captures the total time required for local processing during each FL iteration and forms a critical component of the overall delay model. Average local processing delay can be calculated as:

$$\overline{D}^{(I,l)} = \frac{1}{K} \sum_{k=1}^{K} D_k^{(I,l)}. \tag{5.11}$$

For the j-th UAV–MEC server, the local loss function for processing offloaded datasets is defined as:

$$F_j(\omega_j^U) = \frac{1}{|b_j^{(U,l)}|} \sum_{i \in b_j^{(U,l)}} f_i(\omega_j^U), \tag{5.12}$$

where ω_j^U denotes the model parameters of the j-th UAV–MEC after processing the offloaded data batches. The term $b_j^{(U,l)}$ represents the set of mini-batches allocated for local training at the UAV–MEC, comprising aggregated data samples offloaded from multiple IoT devices.

The global aggregated model is then computed as:

$$\omega = \frac{1}{\mathfrak{R}} \left(\sum_{k \in \mathcal{N}} |b_k^{(I,l)}| \, \omega_k^I + \sum_{j \in \mathcal{M}} |b_j^{(U,l)}| \, \omega_j^U \right), \tag{5.13}$$

where $\mathfrak{R} = \sum_{k \in \mathcal{N}} |b_k^{(I,l)}| + \sum_{j \in \mathcal{M}} |b_j^{(U,l)}|$ denotes the total number of data samples participating in the current FL round across all IoT devices and UAV–MEC servers. This weighted aggregation follows the standard *FedAvg* principle, assigning proportional importance based on the amount of data processed at each node. The local computation delay of the j-th UAV–MEC server is expressed as:

$$D_j^{(U,l)} = \frac{\Upsilon \, b_j^{(U,l)} \, B_j^U \, L_j^U}{s_j^U}, \tag{5.14}$$

where Υ denotes the number of epochs per FL round, B_j^U represents the batch size (in bits) processed by the UAV–MEC, L_j^U indicates the number of CPU cycles required to process one bit of data, s_j^U is the computational capacity for local processing [88]. This relationship reflects the inverse-square dependence of computation delay on available processing power, a common feature in dynamic voltage and frequency scaling (DVFS)-based energy models.

The total system processing delay can be defined as:

$$D^{(I,l)} = \frac{K}{K + \mathcal{M}} \{ \sum_{k \in K} D_k^{(I,l)} + \sum_{m \in \mathcal{M}} D_m^{(U,l)} \} + N D_{ab}, \tag{5.15}$$

where D_{ab} denotes aggregation and broadcasting delay, and N is the number of global FL iterations required for model convergence [105].

We formulate an optimization problem with an objective to minimize the maximum delay, which consists of offloading and processing delays:

$$\min_{b_k^I, b_k^o, p_k^{(I,o)}, p_k^{(I,l)}, \mathfrak{G}_i} : W_1 \mathfrak{D} \triangleq \frac{(D^{(I,o)} + D^{(I,l)})}{D_{ref}}$$

$$+ W_2 \frac{\max_{\mathfrak{D}_i} \mathfrak{V}_\Omega(V_{U_1}(\mathfrak{D}_\Omega, \mathfrak{G}_\Omega), \ldots, \mathfrak{V}_{U_n}(\mathfrak{D}_\Omega, \mathfrak{G}_\Omega))}{\mathfrak{V}_{ref}}$$

Subject to:

$$C1 : \sum_{k \in \mathcal{N}} p_k^{(I,o)} + \sum_{k \in \mathcal{N}} p_k^{(I,l)} \leq P_{total}^{IoT}$$

$$C2 : \sum_{j \in \mathcal{M}} p_j^{(U,o)} + \sum_{j \in \mathcal{M}} p_j^{(U,l)} \leq P_{total}^{U}$$

$$C3 : 0 \leq p_k^{(I,o)}, p_k^{(I,l)} \leq P_{max}^{IoT}, \forall k \in \mathcal{N}$$

$$C4 : P_{min}^{IoT} \leq p_k^{(I,o)} + p_k^{(I,l)} \leq P_{max}^{IoT}, \forall k \in \mathcal{N}$$

$$C5 : P_{min}^{U} \leq p_j^{(U,l)}, p_j^{(U,o)} \leq P_{max}^{U} \forall j \in \mathcal{M} \tag{5.16}$$

$$C6 : P_{min}^{U} \leq p_j^{(U,l)} + p_j^{(U,o)} \leq P_{max}^{U} \forall j \in \mathcal{M}$$

$$C7 : \hat{B} log_2(1 + p_k^{(I,o)} G_k / N_0) \geq X_{min}, \forall k \in \mathcal{N}$$

$$C8 : 0 \leq b_j^{(U,\alpha)}(t) \leq b_{j,\max}^{(U,\alpha)}, \quad \forall \alpha \in \{l, o\}$$

$$C9 : \mathfrak{D} < \mathfrak{D}_{max}$$

where D_{ref} and V_{ref} are constants, which are utilized to normalize the scales for delay and GAN respectively. C1 and C2 ensure that total power consumption by IoT devices and UAV-MECs stays within the power budget $(P_{total}^{I}, P_{total}^{U})$. C3- C6 bound the power allocated for local processing and offloading at IoT devices and UAVs, ensuring that valid operational limits $((P_{min}^{I}, P_{max}^{I}), (P_{min}^{U}, P_{max}^{U}))$ are maintained. C7 enforces that the data rate for each IoT device must meet a minimum threshold (X_{min}) to satisfy QoS requirements. C8 ensures that offloading and processing queues $(b_{j,\max}^{(U,l)}, b_{j,\max}^{(U,o)}), (b_{k,\max}^{(I,l)}, b_{k,\max}^{(I,o)})$ do not exceed capacity. C9 ensures that the total system delay $\mathfrak{D}$ remains below a predefined maximum $(\mathfrak{D}_{max})$, enabling efficient task execution. This optimization framework ensures that FL in the UAV-aided IoT network is efficient, stable, and meets QoS constraints while minimizing overall system delay. It is important to note that we minimize the sum of delays across all devices rather than the maximum delay at IoT devices. Our focus on optimizing queue lengths motivates this approach, which inherently balances the delay across devices. By preventing

any single IoT device from experiencing excessively long queues, we implicitly ensure that delay disparities among devices remain minimal.

5.3 Proposed Solution

To solve the dynamic stochastic optimization problem formulated in Section IV, we adopt the Lyapunov drift-plus-penalty (DPP) method, a well-established technique for stabilizing queue-based systems while optimizing long-term performance metrics. The same approach has previously been deployed in our conference version of this work [106], wherein a UAV-assisted FL system without GAN augmentation was analyzed. In the current study, we extend that framework by incorporating data augmentation dynamics and addressing additional power allocation constraints arising from GAN training.

The key idea of the DPP method is to transform the long-term time-average optimization problem into a series of deterministic per-time-slot sub-problems. These sub-problems are constructed by minimizing a weighted combination of the Lyapunov drift (which encourages queue stability) and the penalty function (which captures the objective, such as power consumption). Let Υ denote the drift weight and $V > 0$ denote the control parameter that balances the trade-off between penalty and queue stability. The detailed derivation of the drift-plus-penalty expression has been omitted here for brevity but can be found in [106].

Upon decomposing the problem, we obtain the following three sub-pr

The first sub-problem of queue backlogs can be written as:

$$\begin{aligned}\min_{\substack{b_i^{(I,\alpha)},\, b_i^{(U,\alpha)} \\ \forall\ \alpha\in\{i,o\}}} & \sum_{i\in K}\Big(Q_i^{(I,l)}(t) - Q_i^{(I,\alpha)}(t)\Big)b_i^{(I,l)} \\ & + \sum_{i\in K}\Big(Q_i^{(I,o)}(t) - Q_i^{(I,\alpha)}(t)\Big)b_i^{(I,o)} \\ & + \sum_{m\in\mathcal{M}}\Big((Q_m^{(U,l)}(t) - Q_m^{(U,\alpha)}(t))\Big)b_i^{(U,l)} \end{aligned} \tag{5.17}$$

$$+ \sum_{m \in \mathcal{M}} \Bigg\{ (Q_m^{(U,o)}(t) - Q_m^{(U,\alpha)}(t) \Bigg) b_i^{(U,o)}$$

Subject to C8, C9 and $\overline{Q} < \infty$.

The second and third subproblems of local processing powers and offloading power for IoT devices, primary UAV and secondary UAVs are convex and non-convex, respectively and can be written as:

$$\min_{\substack{p_i^{(\gamma,l)} \\ \forall\ \gamma \in \{U,I\}}} \sum_{i \in K} \left(V \frac{\tau_0 s_k^I}{L_k^I B} - \frac{\Upsilon b_k^{(I,l)} B L_k^I}{s_k^I} \right)$$
$$+ \sum_{m \in \mathcal{M}} \left(V \frac{\tau_0 s_m^U}{L_m^I B_m} - \frac{\Upsilon \sum_{k \in K} b_i^{(I,o)} B L_k^I}{s_m^U} \right) \tag{5.18}$$

Subject to C1 - C6, C9.

and,

$$\min_{p_i^{(I,\gamma)} \forall\ \gamma \in \{I,U\}} \sum_{i \in K} \sum_{m \in \mathcal{M}} \Bigg(V \mathbb{B} \tau_o log_2(1 + p_i^{(I,o)} G_i / N_0)$$
$$- b_i^{(I,o)} / (\mathbb{B} \tau_o log_2(1 + p_i^{(I,o)} G_i / N_0)) \Bigg)$$
Subject to C1 - C7, C9. (5.19)

We employ the SQP method to solve each optimization subproblem independently. In every iteration, the nonlinear objective and constraints are locally approximated by a quadratic model and linear models, respectively. This yields a quadratic programming subproblem whose solution defines a search direction for the original nonlinear problem. The candidate solution is then updated using either a line-search or trust-region strategy to ensure progress and feasibility. Overall, SQP progressively refines the solution by constructing increasingly accurate local approximations around the optimum, making it particularly ef-

fective for the smooth yet non-convex nature of our power allocation subproblems [101]. Our overall algorithmic framework closely follows the approach in [106], with extensions to handle GAN-augmented data queues and dynamic power allocation for local and FL phases. The reader is referred to that work for the complete derivation and convergence analysis of the drift-plus-penalty formulation.

5.4 Performance Evaluation

5.4.1 Simulations Setup

Deep-learning Configurations

We set the parameters $\alpha = 1$ and $\beta = 40$, and define the target training accuracy for the FL model as 98%. The reported results are averaged over 10^3 Monte Carlo simulations and presented with a 95% confidence interval. Prior to training, the dataset is normalized using a mean and variance of 0.5. For the FL model, we employ a Convolutional Neural Network (CNN) consisting of two convolutional layers followed by a flattening operation and a fully connected output layer. Each client performs three local training epochs with a learning rate of 1×10^{-3}, using the ADAM optimizer and cross-entropy loss function. For the GAN, the discriminator comprises three fully connected layers with LeakyReLU activation and a dropout rate of 0.3, followed by a final linear layer with sigmoid activation. The generator adopts a similar architecture, with three fully connected layers using LeakyReLU activation and a final layer with Tanh activation. We employ the ADAM optimizer for both networks, using learning rates of 3×10^{-3} for the generator and 1×10^{-3} for the discriminator. Binary cross-entropy is used as the loss function, and the batch size is set to 64 samples.

Wireless Configurations

We consider a hierarchical FL network comprising $K = 30$ IoT devices, 3 follower UAVs and 1 leader UAV ($N = 4$). We consider the channel bandwidth to be 20 MHz. Power budget

for each follower UAV-MECs and leader UAV is $P^{U,\text{Total}}$ is 10 Watts. At each device, the total power is partitioned into computation power, denoted by $p^{I,l}$ and $p^{U,l}$, and communication power, denoted by $p^{I,o}$ and $p^{U,o}$, for IoT devices and UAVs, respectively. We consider the communication overhead between IoT devices and UAVs for FL iteration, broadcasting weights back to each IoT device to be constant. This overhead can be considered constant as we communicate with a fixed number of IoT devices and UAV-MECs and almost a fixed communication load for each IoT device and UAV-MEC. We consider a minimum and maximum power limit for both computations and communications for each IoT device. The minimum power for an IoT device is P^{I}_{min}=0.01 W, and the maximum power is P^{I}_{max}= 0.5 W for both communication and computations [97]. For UAV, the lower limit for computation power is P^{U}_{min}= 5W, and the upper limit is P^{U}_{max}= 10 W. We consider the MNIST dataset of size 400 Mb in our simulations [94]. When the cumulative arrivals exceed the size of the original MNIST dataset, we generate additional data samples using standard geometric transformations (i.e., rotation, translation, and sheering) [102]. We consider the capacitance coefficient for UAV (ν^{U}_{m}) and IoT (ν^{I}_{k}) to be $1e^{-19}$ [95]. We fixed an iteration time of 200ms for parameter aggregation and model broadcast. ω_1 and ω_1 are set to 0.5 each. V_{ref} is set to 1, and D_{ref} is set to 4 based on average values of multiple runs. Detailed simulation parameters are shown in Table 5.1.

All simulations were implemented in Python. The traditional optimization components were solved using the `Gurobi` optimizer, whereas the FL framework was implemented in `PyTorch`. The experiments were executed on the Google Cloud Platform (GCP) Compute Engine to ensure consistent computational performance. This configuration provides a balance between computational efficiency and numerical stability, ensuring reliable convergence of the per-slot optimization. The details of the simulation environment are summarized in Table 5.1.

Table 5.1: Simulation parameters.

Description	**Parameter**	**Value**
IoT devices	K	30
Follower UAV-MEC servers	$M-1$	3
Leader UAV-MEC servers		1
Channel bandwidth	$\mathbb{C}$	20 MHz
Power budget for UAV-MECs	P^U_{total}	10 W
Power budget for IoT devices	P^I_{total}	10 W
Minimum power for IoT device	$P^I_{\min}$	0.01 W
Maximum power for IoT device	P^I_{max}	0.5 W
Minimum computation power for UAV-MEC server	$P^U_{\min}$	5 W
Maximum computation power for UAV-MEC server	P^U_{max}	10 W
Capacitance coefficient for UAV-MEC	ξ	$1e^{-19}$
Capacitance coefficient for IoT	ν	$1e^{-19}$
Dataset utilized for FL	MNIST	400Mb
FL parameter	α	1
FL parameter	β	40
FL training accuracy		98%
Transmission and broadcast delay	D_{ab}	200ms
Minimum data rate	$X_{\min}$	1Mb/s
Network type		CNN
CNN structure		2 conv + flatten + fully connected (FC)
Learning rate		1×10^{-10}
Optimizer		ADAM
Loss function		Cross-Entropy
Discriminator layers		3 FC + dropout (0.3) + sigmoid
Generator layers		3 FC + Tanh
Activation function		LeakyReLU
Optimizer (Gen/Disc)		ADAM
Learning rate (Gen)		3×10^{-3}
Learning rate (Disc)		1×10^{-3}
Loss function		Binary Cross-Entropy

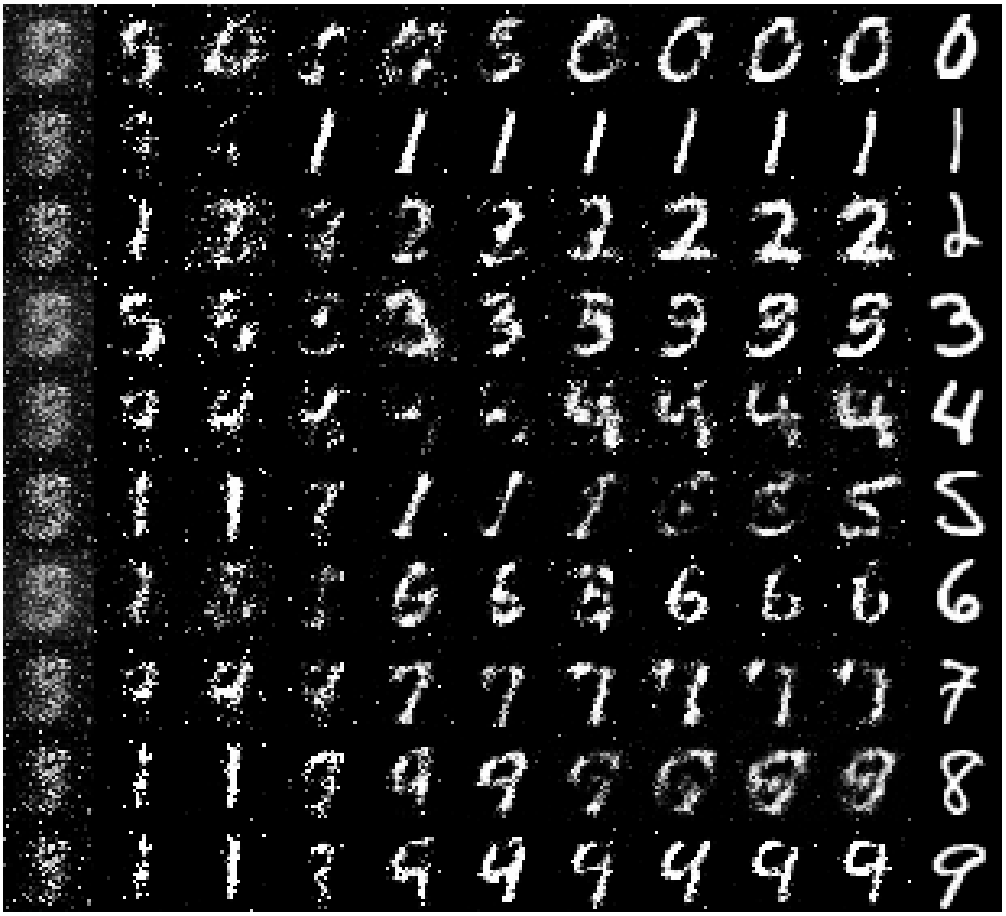

Figure 5.2: GAN data generation.

5.4.2 Simulation Results

Fig. 5.2 illustrates samples generated by the GAN at different stages of training. Each column corresponds to data produced after every five global iterations of GAN training, except for the final column, which depicts the real dataset samples (ground truth). As training progresses, the generator increasingly captures the characteristics of each category. By approximately 45 iterations, the generated samples closely resemble the true data distribution, demonstrating that the generator has effectively learned to represent each class.

Fig. 5.3 shows the offloaded data distributions for follower glsuav-MEC. To analyze the class imbalance in the offloaded dataset, we compute the class distribution at each UAV-MEC server by dividing the number of offloaded samples belonging to each class by the total number of offloaded samples. The resulting class probabilities clearly show that

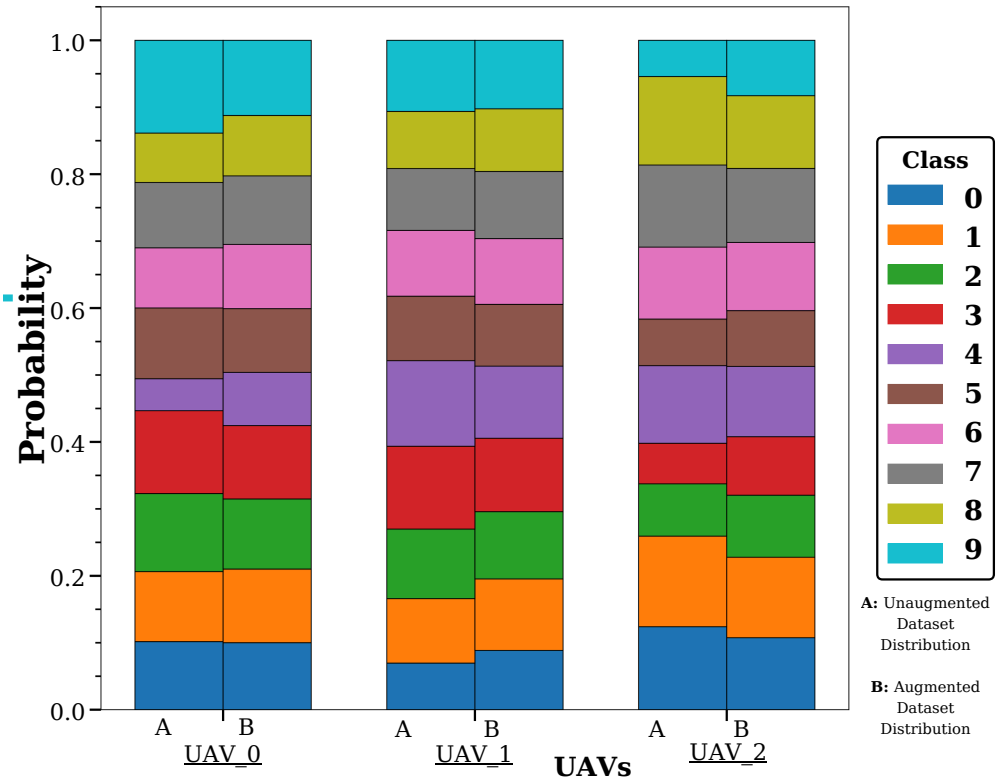

Figure 5.3: Class probabilities at UAVs with and without data-augmentation.

the distribution is non-uniform across classes and across different UAVs. For instance, the probability of class 9 in the dataset offloaded to UAV-1 is significantly higher than that of any other class, indicating an over-representation of class 9 samples. In contrast, at UAV-3, the probability of class 9 is considerably lower, revealing an under-representation of that class. Similar variations are observed for other classes as well, confirming that the dataset is highly imbalanced across different UAV-MEC servers. After applying the trained GAN to selectively generate additional samples for the under-represented classes, the class distribution becomes substantially more uniform. This demonstrates the effectiveness of the proposed GAN-based augmentation strategy in mitigating class imbalance in federated offloading scenarios.

Fig. 5.4 presents the loss functions of the global generator and discriminator at the leader UAV. As observed, the generator loss initially increases during the early iterations, which is typical in GAN training since the generator starts with random weights and gradually learns

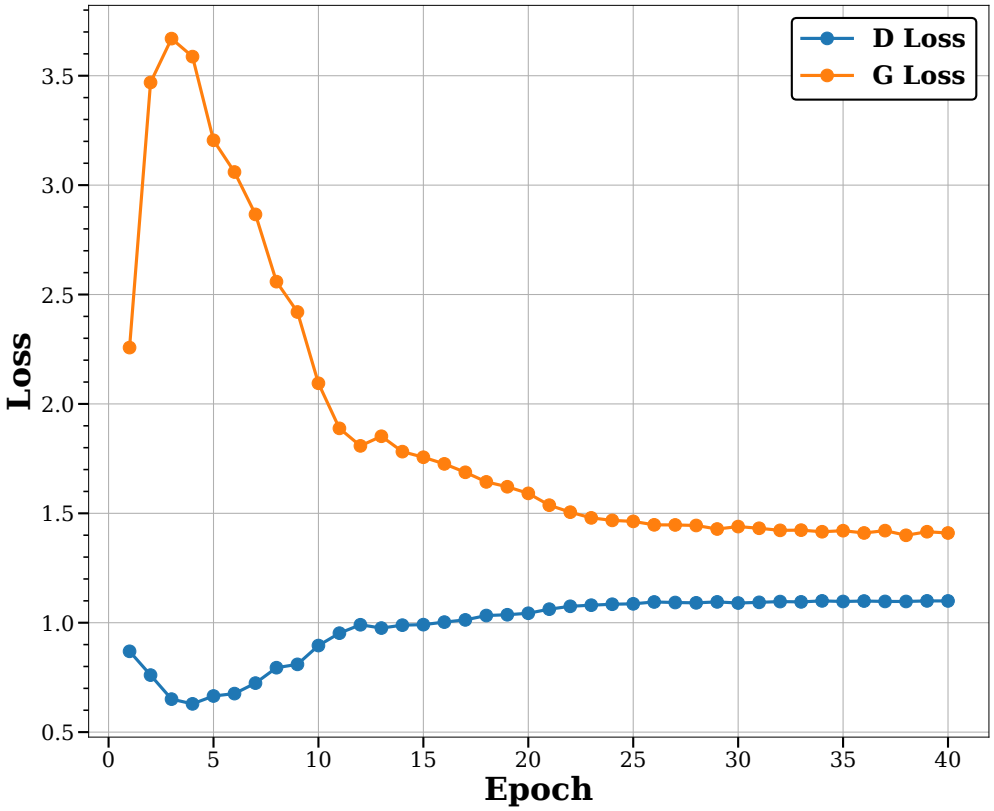

Figure 5.4: Generator and discriminator loss at leader UAV.

to produce realistic outputs. The distributed nature of the training further contributes to this initial fluctuation, as model aggregation from multiple follower UAVs introduces additional variability. As training progresses, the generator loss begins to decrease while the discriminator loss rises, an indication that the generator is improving and producing samples increasingly difficult for the discriminator to classify correctly. Around iteration 35, both losses stabilize, signifying that the adversarial training process has reached equilibrium and the model has achieved convergence.

Fig. 5.6 presents a comparative analysis of several state-of-the-art data augmentation techniques against the proposed GAN-based augmentation method. During the initial phase (Rounds 0–9), the model is trained exclusively on the original dataset to establish a baseline accuracy. Starting from Round 10, different augmentation strategies are incorporated to examine their impact on model convergence and final performance. Representative samples of

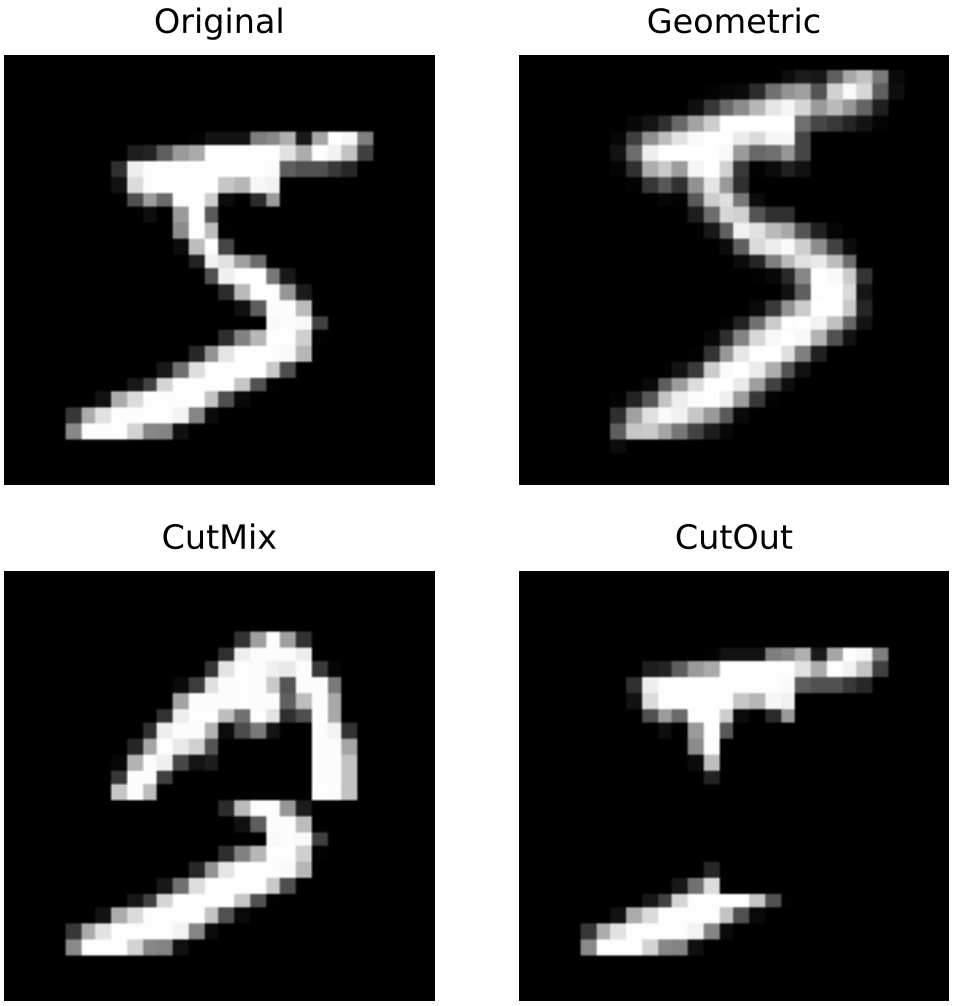

Figure 5.5: Dataset samples of different solution approaches.

all datasets generated via these augmentation strategies are also illustrated in Fig. 5.5, highlighting the visual differences introduced by each method. The first method evaluated is the geometric transformation approach [107], which applies traditional image-space operations such as random rotations, flips, crops, translations, scaling, and zooming. Although widely used in the literature, this method yields the weakest improvement in our experiments. This is primarily because the dataset under study is already well-oriented, and geometric alterations add limited semantic variation while sometimes introducing distortions that hinder learning. The second technique is CutOut [108], where randomly selected rectangular regions within an image are masked out. By forcing the model to rely on multiple discriminative

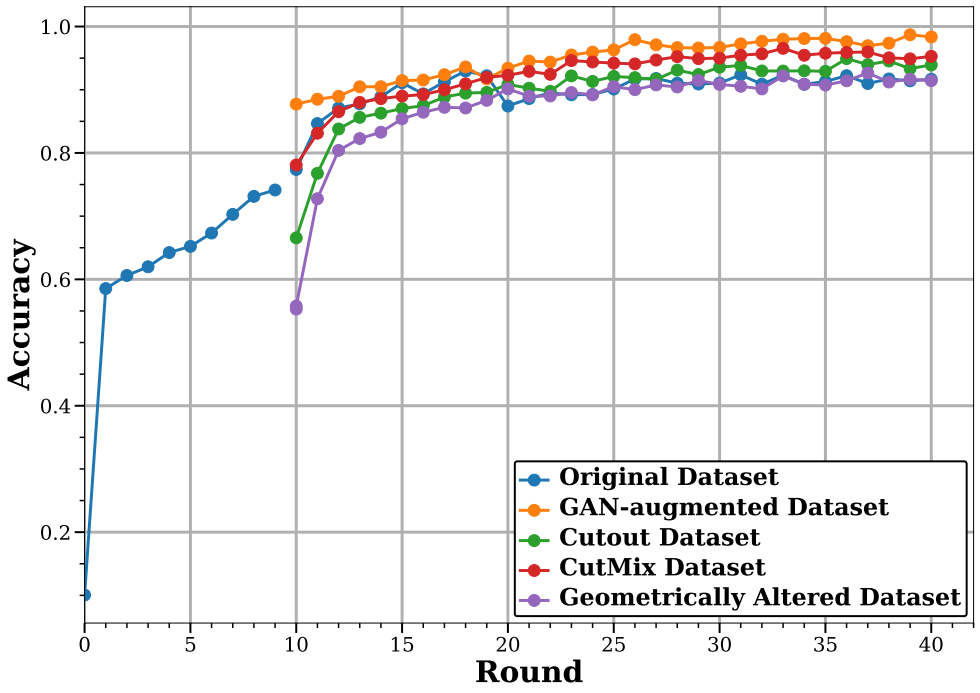

Figure 5.6: Performance of different approaches.

regions rather than a single dominant feature, CutOut demonstrates noticeable improvement over the non-augmented baseline. The third method, CutMix [109], replaces a portion of an image with a corresponding patch from another sample belonging to the same class. Unlike CutOut, which removes information, CutMix substitutes it with meaningful class-consistent content, enabling the model to learn richer mixed-feature representations. This results in slightly better performance than CutOut, as shown in Fig. 5.6. Finally, the proposed GAN-augmented dataset achieves the highest accuracy among all evaluated techniques. By synthesizing realistic samples that preserve the statistical distribution of the original dataset, the GAN introduces greater variability without compromising label fidelity, resulting in both faster convergence and superior final accuracy. In summary, geometric transformations yield limited benefit, CutOut and CutMix offer moderate improvements, whereas the GAN-based augmentation provides the most substantial performance gain, confirming its effectiveness in improving model robustness in FL scenarios.

Fig. 5.7 illustrates the average power consumption across different system components

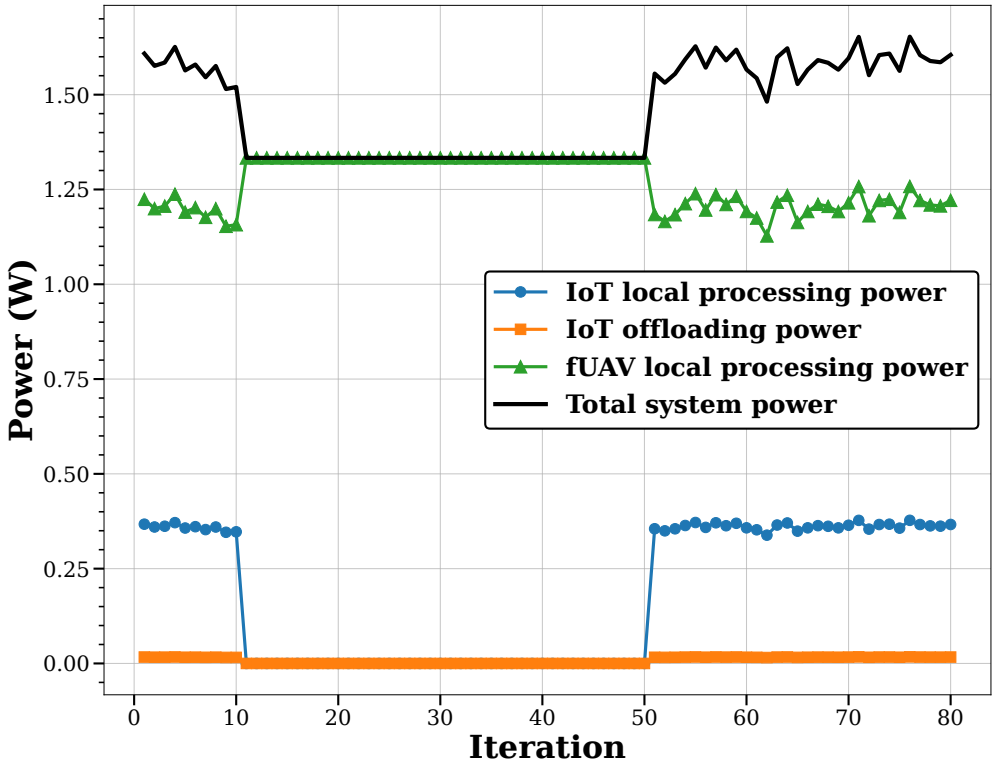

Figure 5.7: Power consumption during FL and GAN training.

during the learning process. During the initial FL iterations (Rounds 1–10), the follower UAV-MEC servers exhibit relatively low power usage, primarily due to limited local processing and lightweight communication tasks. A similar trend is observed after Round 50, when the system transitions back to standard FL. However, in the intermediate phase (Rounds 10–50), corresponding to the GAN training period, there is a noticeable increase in power consumption by the follower UAV-MEC servers. This is attributed to the additional computational load associated with GAN-based dataset generation and local model training. In contrast, the IoT devices exhibit minimal power usage in this interval, as they focus solely on data collection and offloading, without participating in local learning.

Overall, the results clearly show a shift in power distribution during GAN-assisted augmentation, highlighting the increased computation burden on follower UAVs and reduced activity at the IoT devices.

5.5 Conclusion

This chapter presented a novel UAV-aided distributed GAN framework integrated within a hierarchical edge-assisted FL environment. The proposed architecture leverages the computational capabilities of multiple follower UAV-MEC servers and a leader UAV to collaboratively train GAN models using data offloaded from resource-constrained IoT devices. By enabling decentralized and privacy-preserving model training, the system effectively reduces the reliance on centralized data collection and supports scalable learning across geographically distributed IoT devices.

Comprehensive experimental evaluations demonstrated the progressive improvement in generator performance and the stabilizing trend of the adversarial loss functions over successive global iterations. The generator initially exhibited higher loss values due to random initialization and distributed aggregation effects but gradually achieved convergence as the number of training rounds increased. After approximately 35–40 iterations, both the generator and discriminator losses stabilized, indicating that the GAN had reached an equilibrium where generated samples closely approximated the real data distribution.

Overall, the results confirm the feasibility and robustness of the proposed distributed GAN training scheme in UAV-assisted edge environments. Beyond demonstrating stable convergence behavior, the framework provides a practical pathway for on-demand data generation and augmentation in remote or data-scarce regions. Future research will focus on extending the current design to multi-leader and multi-hop topologies, incorporating adaptive communication strategies, and integrating reinforcement learning-based coordination for improved convergence efficiency and reduced energy consumption. Extending the experimental evaluation to more complex datasets (e.g., CIFAR-10 or domain-specific sensing datasets) is an important direction for future work, particularly for large-scale deployments w here model complexity and data heterogeneity are more pronounced.

Chapter 6

Conclusion

This thesis presented a comprehensive investigation into the design, optimization, and enhancement of UAV-aided FL systems for mission-critical IoT applications. Motivated by the increasing demand for reliable, low-latency, and privacy-preserving intelligence in remote and infrastructure-limited environments, this work explored how UAV-mounted MEC platforms, hierarchical FL architectures, queueing mechanisms, and GANs can collectively overcome long-standing challenges such as device heterogeneity, the straggler effect, and Non-IID data distributions.

The research contributions evolved across three progressively advanced frameworks. First, the FL framework demonstrated how partial dataset offloading to a UAV-MEC server, combined with optimized power allocation, can significantly reduce system delay while maintaining energy efficiency in heterogeneous IoT environments. Deterministic optimization and reinforcement-learning-based scheduling were proposed and validated through simulation, showing clear improvements over classical FL and edge learning baselines.

Second, this thesis extended UAFL into a hierarchical architecture through the Hierarchical UAV-aided FL framework enriched with a multi-queue model. This design introduced follower–leader UAV hierarchy, queue-aware scheduling, and Lyapunov-based optimization for power, offloading, and QoS-aware decision making. The hierarchical queue-based struc-

ture substantially improved scalability, system stability, and latency performance, particularly under dynamic data arrivals.

Finally, this thesis introduced a GAN-augmented hierarchical UAV-MEC FL system, addressing the persistent issues of Non-IID data, information islands, and accuracy degradation. By leveraging offloaded datasets to train distributed GANs at follower UAV-MECs and aggregating generator models at the leader UAV, the framework simultaneously achieved delay reduction and accuracy enhancement. To the best of our knowledge, this is the first work to integrate federated GAN training within a queue-aware hierarchical UAV-MEC FL pipeline.

Across all three stages, simulation results consistently demonstrated that the proposed frameworks outperform traditional FL, edge-only learning, and single-tier UAV-assisted learning schemes. The results highlight the critical importance of jointly optimizing computation, communication, and data-processing strategies in UAV-enabled intelligent systems.

6.1 Future Research Directions

Building on the findings of this thesis, several promising avenues remain open for future investigation:

- **UAV trajectory optimization:** Incorporating trajectory control jointly with learning, offloading, and queueing decisions opens the door to fully adaptive UAV behavior. By dynamically repositioning UAVs based on real-time workload imbalance, queue congestion, wireless link quality, or spatial distribution of critical events, the system can reduce communication delay, balance computational load, and improve sensing coverage. Future work may focus on reinforcement-learning-based mobility controllers, multi-UAV cooperative path planning, and predictive mobility models that anticipate data surges before they occur.

- **Security and privacy enhancements:** As hierarchical and queue-aware UAV-

assisted FL systems operate in open and potentially adversarial environments, ensuring data integrity, confidentiality, and robustness becomes crucial. Future research can explore integrating adversarial training to defend against data poisoning, differential privacy mechanisms to limit information leakage from gradient updates, and lightweight secure aggregation designed for bandwidth-limited UAV–IoT networks. These improvements would help guarantee safe deployment in mission-critical scenarios such as disaster response or border surveillance.

- **Lightweight generative models for resource-constrained UAVs:** Although GANs and diffusion models significantly improve learning accuracy under Non-IID data distributions, their computational overhead poses challenges for power-limited UAV platforms. Developing lightweight or compressed generative models through pruning, quantization, neural architecture search, or distilled variants could enable on-board dataset augmentation without breaking energy constraints.

- **Real-world prototyping and experimentation:** To validate scalability and reliability, the proposed hierarchical and queue-aware FL framework should be deployed on real embedded platforms, such as Jetson TX2-powered UAVs, Raspberry Pi-based MEC nodes, or hybrid cloud–edge testbeds. Experimental evaluation under realistic wireless conditions, variable channel gains, intermittent connectivity, and heterogeneous IoT devices, would provide insights into practical limitations, fine-tune energy–delay trade-offs, and bridge the gap between simulation and operational field environments. Such prototypes are essential for future integration into disaster response systems, environmental monitoring, and smart-city infrastructures.

Closing Remarks

Overall, this thesis provides a unified and scalable design for next-generation UAV-assisted FL systems. By integrating optimization theory, queueing models, hierarchical cooperative

learning, and generative modeling, this research advances the state of the art in edge intelligence and lays the foundation for autonomous, distributed, and resilient learning frameworks capable of operating in mission-critical IoT deployments.

Bibliography

[1] L. J. Nelson, R.-G. Lin II, and L. Money, "Relentless rains continue to drench Southern California." `https://www.latimes.com/california/story/2024-02-06/relentless-rains-continue-to-drench-southern-california`, feb. 2024. Los Angeles Times.

[2] D. J. Mihalek and R. M. Frankel, "Texas flash flooding disaster raises questions about rescue and recovery efforts," *ABC News*. Accessed: 2025-07-08.

[3] J. Xu, K. Ota, and M. Dong, "Big data on the fly: UAV-mounted mobile edge computing for disaster management," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2620–2630, Dec. 2020.

[4] Z. Shah, M. Naeem, U. Javed, W. Ejaz, and M. Altaf, "A compendium of radio resource management in UAV-assisted next generation computing paradigms," *Ad Hoc Networks*, vol. 131, p. 102844, Jun. 2022.

[5] J. Kuang, M. Yang, H. Zhu, and H. Qian, "Client selection with bandwidth allocation in federated learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 01–06, IEEE, Feb. 2022.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, Apr. 2017.

[7] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 857–866, IEEE, May 2018.

[8] C. Feng, Z. Zhao, Y. Wang, T. Q. Quek, and M. Peng, "On the design of federated learning in the mobile edge computing systems," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 5902–5916, Sep. 2021.

[9] H. Wu and P. Wang, "Node selection toward faster convergence for federated learning on non-iid data," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3099–3111, Sep. 2022.

[10] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, May 2021.

[11] A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, M. Guizani, Z. Dawy, and W. Nasreddine, "Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data," *Future Generation Computer Systems*, vol. 128, pp. 406–419, Mar. 2022.

[12] D. Manu, A. Alazzwi, J. Yao, Y. Lin, and X. Sun, "AsyncFedGAN: An efficient and staleness-aware asynchronous federated learning framework for generative adversarial networks," *IEEE Transactions on Parallel and Distributed Systems*, Dec. 2024.

[13] A. K. Selvaraj, S. B. Prathiba, A. D. Kumar, R. Dhanalakshmi, T. R. Gadekallu, and G. Srivastava, "Co-Training-Based Personalized Federated Learning With Generative Adversarial Networks for Enhanced Mobile Smart Healthcare Diagnosis," *IEEE Transactions on Consumer Electronics*, Sep. 2024.

[14] Z. Ma, Y. Liu, Y. Miao, G. Xu, X. Liu, J. Ma, and R. H. Deng, "Flgan: Gan-based unbiased federated learning under non-iid settings," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 4, pp. 1566–1581, Aug. 2023.

[15] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran, and Q. E. U. Haq, "Machine learning techniques for 5G and beyond," *IEEE Access*, vol. 9, pp. 23472–23488, Jan. 2021.

[16] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the Internet of Things: A comprehensive survey," *Ad Hoc Networks*, vol. 93, p. 101913, Oct. 2019.

[17] S. Sritharan, H. Weligampola, and H. Gacanin, "A study on deep learning for latency constraint applications in beyond 5G wireless systems," *IEEE Access*, vol. 8, pp. 218037–218061, Nov. 2020.

[18] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[19] A. A. Al-Saedi, E. Casalicchio, and V. Boeva, "An Energy-Aware Multi-Criteria Federated Learning Model for Edge Computing," in *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 134–143, IEEE, Nov. 2021.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, Apr. 2017.

[21] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *Ieee Network*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[22] 3GPP, "Enhanced LTE support for aerial vehicles," *3GPP Technical Report*, vol. TR 36.777, no. V15.0.0, pp. 1–85, Dec. 2017. Includes channel models, path loss models, interference analysis, and connectivity requirements for UAVs in LTE/5G systems.

[23] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1578–1598, Jul. 2021.

[24] M. Lin and Y. Zhao, "Artificial intelligence-empowered resource management for future wireless communications: A survey," *China Communications*, vol. 17, no. 3, pp. 58–77, Mar. 2020.

[25] A. Salh, L. Audah, N. S. M. Shah, A. Alhammadi, Q. Abdullah, Y. H. Kim, S. A. Al-Gailani, S. A. Hamzah, B. A. F. Esmail, and A. A. Almohammedi, "A survey on deep learning for ultra-reliable and low-latency communications challenges on 6G wireless systems," *IEEE Access*, vol. 9, pp. 55098–55131, Mar. 2021.

[26] W. Ejaz, S. K. Sharma, S. Saadat, M. Naeem, A. Anpalagan, and N. A. Chughtai, "A comprehensive survey on resource allocation for CRAN in 5G and beyond networks," *Journal of Network and Computer Applications*, vol. 160, p. 102638, Jun. 2020.

[27] Z. Shah, M. Naeem, U. Javed, W. Ejaz, and M. Altaf, "A compendium of radio resource management in UAV-assisted next generation computing paradigms," *Ad Hoc Networks*, vol. 131, p. 102844, Jun. 2022.

[28] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE communications surveys & tutorials*, vol. 22, no. 2, pp. 1251–1275, Apr. 2020.

[29] A. Mughees, M. Tahir, M. A. Sheikh, and A. Ahad, "Towards energy efficient 5G networks using machine learning: Taxonomy, research challenges, and future research directions," *IEEE Access*, vol. 8, pp. 187498–187522, Oct. 2020.

[30] S. Zhang and D. Zhu, "Towards artificial intelligence enabled 6G: State of the art, challenges, and opportunities," *Computer Networks*, vol. 183, p. 107556, Dec. 2020.

[31] S. S. Sefati, A. U. Haq, R. Craciunescu, S. Halunga, A. Mihovska, O. Fratu, *et al.*, "A comprehensive survey on resource management in 6G network based on Internet of Things," *IEEE Access*, Aug. 2024.

[32] N. Alhussien and T. A. Gulliver, "Toward AI-enabled green 6G networks: A resource management perspective," *IEEE Access*, Sep. 2024.

[33] D. Alsadie, "A comprehensive review of AI techniques for resource management in fog computing: Trends, challenges and future directions," *IEEE Access*, Sep. 2024.

[34] H. Dahrouj, R. Alghamdi, H. Alwazani, S. Bahanshal, A. A. Ahmad, A. Faisal, R. Shalabi, R. Alhadrami, A. Subasi, M. T. Al-Nory, *et al.*, "An overview of machine learning-based techniques for solving optimization problems in communications and signal processing," *IEEE Access*, vol. 9, pp. 74908–74938, May 2021.

[35] S. Taimoor, L. Ferdouse, and W. Ejaz, "Holistic resource management in uav-assisted wireless networks: An optimization perspective," *Journal of Network and Computer Applications*, p. 103439, Sep. 2022.

[36] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, Oct. 2021.

[37] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive internet of things for smart industries," *IEEE Access*, vol. 8, pp. 168854–168864, Sep. 2020.

[38] C. Feng, Z. Zhao, Y. Wang, T. Q. Quek, and M. Peng, "On the design of federated learning in the mobile edge computing systems," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 5902–5916, Sep. 2021.

[39] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, Feb. 2020.

[40] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[41] M. Merluzzi, P. Di Lorenzo, and S. Barbarossa, "Wireless edge machine learning: Resource allocation and trade-offs," *IEEE Access*, vol. 9, pp. 45377–45398, Mar. 2021.

[42] Z. Lin, S. Bi, and Y.-J. A. Zhang, "Optimizing AI service placement and resource allocation in mobile edge intelligence systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7257–7271, Nov. 2021.

[43] J. Feng, W. Zhang, Q. Pei, J. Wu, and X. Lin, "Heterogeneous computation and resource allocation for wireless powered federated edge learning systems," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3220–3233, May 2022.

[44] C. Deng, X. Fang, and X. Wang, "UAV-enabled mobile edge computing for AI applications: Joint model decision, resource allocation and trajectory optimization," *IEEE Internet of Things Journal*, 2022.

[45] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint multiuser DNN partitioning and computational resource allocation for collaborative edge intelligence," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9511–9522, Jan. 2021.

[46] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, Apr. 2016.

[47] S. Wang, Y.-C. Wu, M. Xia, R. Wang, and H. V. Poor, "Machine intelligence at the edge with learning centric power allocation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7293–7308, Nov. 2020.

[48] Y. Shen, Y. Qu, C. Dong, F. Zhou, and Q. Wu, "Joint training and resource allocation optimization for federated learning in UAV swarm," *IEEE Internet of Things Journal*, Feb. 2023.

[49] G. Wang, F. Xu, H. Zhang, and C. Zhao, "Joint resource management for mobility supported federated learning in internet of vehicles," *Future Generation Computer Systems*, vol. 129, pp. 199–211, Apr. 2022.

[50] W. Wen, Z. Chen, H. H. Yang, W. Xia, and T. Q. Quek, "Joint scheduling and resource allocation for hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, January .2022.

[51] J. Ren, G. Yu, and G. Ding, "Accelerating DNN training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 219–232, Jan. 2021.

[52] D. Ismailova and W.-S. Lu, "Penalty convex-concave procedure for source localization problem," in *2016 IEEE Canadian conference on electrical and computer engineering (CCECE)*, pp. 1–4, IEEE, Nov. 2016.

[53] A. A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, M. Guizani, Z. Dawy, and W. Nasreddine, "Communication-efficient hierarchical federated learning for Iot heterogeneous systems with imbalanced data," *Future Generation Computer Systems*, vol. 128, pp. 406–419, Mar. 2022.

[54] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 15–24, IEEE, Dec. 2020.

[55] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, Dec. 2019.

[56] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233–250, Dec. 2020.

[57] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Hierarchical coded gradient aggregation for learning at the edge," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 2616–2621, IEEE, Jun. 2020.

[58] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, *et al.*, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, vol. 29, pp. 854–864, Sep. 2014.

[59] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE transactions on wireless communications*, vol. 17, no. 11, pp. 7590–7605, Nov. 2018.

[60] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[61] Y. Han, D. Li, H. Qi, J. Ren, and X. Wang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1–5, May 2019.

[62] G. Wang, F. Xu, H. Zhang, and C. Zhao, "Joint resource management for mobility supported federated learning in Internet of Vehicles," *Future Generation Computer Systems*, vol. 129, pp. 199–211, Apr. 2022.

[63] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedadapt: Adaptive offloading for iot devices in federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20889–20901, Nov. 2022.

[64] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[65] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2041–2057, Dec. 2021.

[66] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, "Convergence of update aware device scheduling for federated learning at the wireless edge," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3643–3658, Jun. 2021.

[67] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *Transactions on Machine Learning Research*, Sep. 2022.

[68] Y. Qu, C. Dong, J. Zheng, H. Dai, F. Wu, S. Guo, and A. Anpalagan, "Empowering edge intelligence by air-ground integrated federated learning," *IEEE Network*, vol. 35, no. 5, pp. 34–41, Sep. 2021.

[69] Y. Jing, Y. Qu, C. Dong, W. Ren, Y. Shen, Q. Wu, and S. Guo, "Exploiting UAV for air–ground integrated federated learning: A joint UAV location and resource optimization approach," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 3, pp. 1420–1433, Sep. 2023.

[70] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, Oct. 2021.

[71] J. Du, B. Jiang, C. Jiang, Y. Shi, and Z. Han, "Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1035–1050, Apr. 2023.

[72] X. Xu, G. Feng, S. Qin, Y. Liu, and Y. Sun, "Joint UAV Deployment and Resource Allocation: a Personalized Federated Deep Reinforcement Learning Approach," *IEEE Transactions on Vehicular Technology*, Mar. 2024.

[73] Y. Du, Z. Huang, S. Yang, and H. Xiao, "Collaborative task offloading based on deep reinforcement learning in heterogeneous edge networks," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 375–380, IEEE, Jul. 2024.

[74] L. Leconte, M. Jonckheere, S. Samsonov, and E. Moulines, "Queuing dynamics of asynchronous Federated Learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 1711–1719, PMLR, May 2024.

[75] T. Shi, H. Tian, T. Zhang, J. Loo, J. Ou, C. Fan, and D. Yang, "Task scheduling with collaborative computing of MEC system based on federated learning," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pp. 1–5, IEEE, Aug. 2022.

[76] A. Wijesinghe, S. Zhang, and Z. Ding, "PS-FedGAN: An Efficient Federated Learning Framework with Strong Data Privacy," *IEEE Internet of Things Journal*, Jun. 2024.

[77] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A novel federated learning scheme for generative adversarial networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 3633–3649, May. 2023.

[78] L. Wu, H. Lin, and X. Wang, "Federated Training Generative Adversarial Networks for Heterogeneous Vehicle Scheduling in IoV," *IEEE Internet of Things Journal*, Dec. 2024.

[79] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, and X. Cui, "Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10502–10515, Apr. 2022.

[80] X. Cao, G. Sun, H. Yu, and M. Guizani, "PerFED-GAN: Personalized federated learning via generative adversarial networks," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3749–3762, May 2022.

[81] J. Liu, Z. Zhao, X. Luo, P. Li, G. Min, and H. Li, "SlaugFL: Efficient edge federated learning with selective GAN-based data augmentation," *IEEE Transactions on Mobile Computing*, May 2024.

[82] R. Hamdi, A. B. Said, E. Baccour, A. Erbad, A. Mohamed, M. Hamdi, and M. Guizani, "Optimal resource management for hierarchical federated learning over HetNets with wireless energy transfer," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 16945–16958, May 2023.

[83] S. R. Pandey, M. N. Nguyen, T. N. Dang, N. H. Tran, K. Thar, Z. Han, and C. S. Hong, "Edge-assisted democratized learning toward federated analytics," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 572–588, Jun. 2021.

[84] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8866–8870, IEEE, May 2020.

[85] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, Jun. 2020.

[86] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[87] X. Zhong, Y. Guo, N. Li, and Y. Chen, "Joint optimization of relay deployment, channel allocation, and relay assignment for UAVs-aided D2D networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 804–817, Apr. 2020.

[88] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, "Computation offloading for edge-assisted federated learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9330–9344, Sep. 2021.

[89] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, Jun. 2019.

[90] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *arXiv preprint arXiv:2010.13723*, Aug. 2022.

[91] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, p. e2024789118, Apr. 2021.

[92] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2023.

[93] "Enhanced lte support for aerial vehicles," Tech. Rep. TR 36.777 V15.0.0, 3GPP, Dec. 2017.

[94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[95] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *2nd USENIX workshop on hot topics in cloud computing (HotCloud 10)*, Jun. 2010.

[96] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, Apr. 2017.

[97] Y. Song, T. Wang, Y. Wu, L. Qian, and Z. Shi, "Non-orthogonal multiple access assisted federated learning for UAV swarms: An approach of latency minimization," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1123–1128, IEEE, Aug. 2021.

[98] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[99] M. Liaq and W. Ejaz, "Minimizing delay in UAV-aided federated learning for IoT applications with straggling devices," *IEEE Open Journal of the Communications Society*, Nov. 2024.

[100] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, Sep. 2010.

[101] Y. Ma, X. Gao, C. Liu, and J. Li, "Improved sqp and slsqp algorithms for feasible path-based process optimisation," *Computers & Chemical Engineering*, vol. 188, p. 108751, Sep. 2024.

[102] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2020.

[103] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6023–6032, Oct. 2019.

[104] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *arXiv preprint arXiv:1708.04552*, Nov. 2018.

[105] M. Liaq and W. Ejaz, "Minimizing delay in UAV-aided federated learning for IoT applications with straggling devices," *IEEE Open Journal of the Communications Society*, Nov. 2024.

[106] M. Liaq and W. Ejaz, "Computational offloading and delay minimization for UAV-aided edge federated learning," in *ICC 2024-IEEE International Conference on Communications*, pp. 1292–1297, IEEE, Aug. 2024.

[107] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2020.

[108] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *arXiv preprint arXiv:1708.04552*, Nov. 2018.

[109] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features," in *Proceedings of the*

IEEE/CVF International Conference on Computer Vision (ICCV), pp. 6023–6032, Oct. 2019.