Deep Learning for Optimizing Virtual Resources Allocation in MEC-based UAV Environments

Khadidja Zairi, Bouziane Brik, Younes Guellouma, and Hadda Cherroun

***Abstract*—Real-time applications requiring ultra-low latency are driving the deployment of Mobile Edge Computing (MEC) closer to end users. However, when deployed on unmanned aerial vehicles (UAVs), MEC infrastructures face critical constraints in processing, storage, and energy resources. This paper proposes a virtualization-aware MEC framework that integrates GRU-based CPU and memory demand forecasting with dynamic virtual resource allocation in UAV-assisted environments. The architecture combines UAVs, MEC servers, and a central orchestrator to support Collision Detection and Avoidance (CDA) applications through proactive resource provisioning. The proposed approach was evaluated on two heterogeneous UAV datasets, namely OpenSky ADS-B traces and AU-AIR. Experimental results show that the integration of predictive deep learning with virtualization improves resource allocation efficiency compared with non-virtualized baselines. At the same time, GRU achieves more accurate and stable forecasting than RNN and LSTM models. These findings show the effectiveness of proactive virtualization-aware orchestration for the UAV-assisted MEC systems.**

***Index Terms*—Resource allocation, Resource Virtualisation, Collision Detection and Avoidance, UAV, MEC, Deep Learning, GRU forecasting.**

I. Introduction

MOBILE edge computing (MEC) provides real-time applications with ultra-low latency feasible through placing computing resources close to the network edge [1]. This is especially useful for applications that require quick processing of data and in a specific location, like augmented reality, self-driving cars, and industrial automation [2]. It is also useful for UAVs carrying out latency-sensitive tasks such as search-and-rescue, real-time swarm collision avoidance, and on-demand medical/logistics delivery, which critically rely on low-latency edge inference, fast coordination, and timely actuation at the network edge [3], [4].

It has become a major transformational technology in the era of wireless communication that alters how data is processed, stored, and managed at the network edge. As technology continues to improve, the capabilities and uses of UAVs will likely expand substantially. A growing number of platforms and applications that are currently being developed as a result of UAVs have become widespread in a variety of settings [5]. These include aerial photography, videography, surveying, and mapping [6], [7], search and rescue operations, smart agriculture, infrastructure inspection, and delivery services [8], [9]. However, the expected widespread use of UAVs increases the risk of collision-related challenges. MEC servers, particularly those operated on mobile platforms like UAVs [10], [11], frequently encounter constraints regarding processing power, storage capacity, and battery longevity [12].

This issue has been thoroughly examined, resulting in several proposals that utilize MEC [13], [14]. MEC provides a variety of services, including collision detection and avoidance, which involves collecting data from multiple drones through an application to identify potential collisions or send alerts and commands. Thus, deploying the CDA on all MEC hosts is essential to ensure optimal connectivity among all drones in the network. However, this also increases the demand for the computational resources available to the MEC hosts, which require efficient resource management [15].

However, since MEC and UAVs can never leverage full source utilization across a constrained computational facility and in a dynamic operational domain, they are all plagued by resource optimization difficulties by nature [2], [16]. As a viable alternative to MEC's resource optimization limit within UAV surroundings, the notion of virtualization emerges. Virtualization involves acquiring physical resources, such as processing, memory, and network bandwidth, and transforming them into virtual appearances that can be allocated and scaled on the fly [17], [18].

In this specific paradigm, virtualization is introduced as a potential solution to the issue of scarce optimization opportunities that arise when implementing MEC in UAV environments [19]. Virtualization allows abstracting physical resources, namely processing power, storage, and network bandwidth, to create virtual replicas that can be allocated and scaled in real-time to meet the demands of specific applications. Thus, applying virtualization in the MEC implementation in UAVs allows reaching a new level of flexibility, scalability, and resource utilization efficiency.

Therefore, this paper proposes a virtualization-aware orchestration framework for UAV-assisted MEC environments, where lightweight workload forecasting is integrated into the resource allocation loop to provision virtual CPU and memory resources proactively. The objective is to realize the potential of virtualization to circumvent the limitations imposed by the scarce physical resources of UAVs while maximizing MEC

Manuscript received February 26, 2026; revised May 7, 2026. Date of publication September 21, 2026. Date of current version September 21, 2026. The associate editor prof. Giovanni Giambene has been coordinating the review of this manuscript and approved it for publication.

K. Zairi, Y. Guellouma, and H. Cherroun are with the LIM Lab, Amar Telidji University, Laghouat 03000, Algeria (e-mail: kh.zairi@lagh-univ.dz).

B. Brik is with the College of Computing and Informatics, University of Sharjah, UAE.

Digital Object Identifier (DOI): 10.24138/jcomss-2026-0041

capabilities for a wide range of applications. We hope that using these techniques for virtualization will improve the performance, reliability, and flexibility of MEC deployments on UAVs, ensuring, at the same time, that innovative services and applications based on them come to light. A thorough investigation of it brought us to a future where the virtualization-based solution proposed allows the joint improvement of MEC and UAV technologies close to the state of the art, resulting in more advanced wireless communication, data processing, and various other technologies and science fields. This work applies Deep Learning (DL) to predict CPU and memory requirements and allocate virtual CPU and memory resources to MEC host instances effectively, ensuring optimized MEC resource utilization and improved service performance.

We introduce a clustering + Voronoi pipeline for coverage-aware MEC placement and UAV-to-MEC association, which couples spatial distribution with workload forecasting for efficient MEC deployment. We further propose a DL-driven predictive paradigm for forecasting CPU and memory requirements and proactively guiding virtual resource allocation in UAV-assisted MEC environments. A lightweight multivariate GRU predictor is proposed to forecast per-node CPU and memory requirements under UAV mobility, improving provisioning accuracy compared to non-predictive or energy-only schemes [18], [20]. We design a practical orchestration algorithm that maps predicted CPU and memory demand to virtual instances across MEC hosts, enabling proactive provisioning and load balancing in UAV-assisted MEC environments. This specifically addresses the lack of explicit virtualization mechanisms in prior UAV MEC studies [16], [17]. The full framework is validated on two heterogeneous datasets: the large-scale OpenSky ADS-B traces (filtered to Washington State) and the perception-rich AU-AIR UAV traces using identical postprocessing and training protocols to ensure fair cross-dataset comparison. Extensive experimental evaluation, including architectural ablation of Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and deployment comparisons (virtualization+DL, DL-only, baseline), demonstrates improvements in prediction fidelity and resource efficiency. To support reproducibility, the dataset processing scripts, model implementation, and experimental logs will be publicly released upon publication.

The remainder of this paper is structured as follows. Section II reviews the related work on MEC resource allocation, UAV-assisted deployments, and virtualization. Section III presents the proposed methodology, including the GRU-based prediction model and virtualization-driven orchestration framework. The following section IV describes the experimental design, MEC distribution, and datasets used for evaluation. Subsequently, section V reports the results and provides a detailed analysis of prediction accuracy, allocation efficiency, and system performance. Finally, Section VI concludes the paper and highlights possible directions for future research.

II. Related Work

MEC brings cloud services closer to end users in order to reduce latency and improve quality of service (QoS). Extensive research has investigated MEC architectures, deployment models, and application scenarios [2], [21]. The European Telecommunications Standards Institute (ETSI) MEC framework defines standardized functional components and orchestration primitives that enable flexible resource management at the network edge [17]. Despite these advances, MEC systems still face fundamental challenges, including dynamic workload variability, scalable resource orchestration, energy constraints, and security and privacy issues [22], [23]. When MEC is extended to aerial platforms, these challenges are further amplified due to UAV mobility, intermittent connectivity, and rapidly changing service demands. Survey and review works have highlighted the opportunities and open challenges of UAV-assisted MEC, particularly in terms of coverage extension and energy–latency trade-offs [13], [16]. However, these studies remain largely descriptive and do not provide concrete optimization or implementation frameworks for dynamic resource provisioning.

Early research on UAV-assisted MEC primarily relied on optimization-based methods for task offloading and resource allocation. Techniques such as integer programming and heuristic approaches, including Tabu search, were employed for application placement and resource management [24], [25]. While these methods provide tractable solutions under simplified assumptions, they typically consider static or slowly varying workloads and are therefore ill-suited for highly dynamic UAV environments [12]. To better capture mobility effects, trajectory-aware optimization techniques were later introduced. Cui et al. [26], for example, jointly optimized UAV trajectories and MEC resource allocation under power convergence network constraints. Although such approaches demonstrate performance improvements, they generally assume fixed computational capacities at edge nodes and lack proactive adaptation mechanisms to anticipate future workload variations.

More recently, reinforcement learning (RL) and deep reinforcement learning (DRL) have emerged as dominant paradigms for dynamic decision-making in UAV-enabled MEC systems. These methods have been applied to joint task offloading, resource allocation, and UAV path planning problems [27], [28]. Zakaryia et al. [29] proposed a multi-agent deep reinforcement learning framework for joint task offloading and resource allocation in multi-UAV MEC environments, achieving improved system throughput. However, their approach relies on reactive decision-making based on instantaneous system states and does not incorporate workload prediction or resource virtualization. Similarly, Wang et al. [30] addressed joint offloading and resource allocation for low-altitude economy scenarios, but their framework does not forecast future resource demands. Other DRL-based studies further explored decentralized control and energy-efficient offloading. Hwang et al. [31] proposed a decentralized Multi-Agent DRL (MADRL) framework for multi-UAV MEC networks, while Liu et al. [32] combined Lyapunov optimization with Deep Deterministic Policy Gradient(DDPG) to reduce energy consumption. Xu et al. [33] optimized energy–latency trade-offs using MADRL, and Zhai et al. [34] introduced collaborative inference-based DRL for UAV-assisted MEC. Despite their effectiveness for dynamic control, these ap-

proaches remain reactive and do not explicitly predict CPU or memory demands, nor do they integrate virtualization-aware resource management.

In parallel, learning-based workload forecasting has gained attention as a means to improve MEC resource provisioning. Machine learning and deep learning techniques have been employed to predict workload dynamics and guide scheduling decisions, including autoencoder-enhanced DRL for large-scale online scheduling [35], graph-based resource management methods [36], and Cybertwin–DL solutions for energy-aware allocation [20]. Brik et al. [18]combined virtualization concepts with LSTM-based mobility prediction to dimension MEC resources for vehicular collision-avoidance services. At the same time, complementary cloud-side studies evaluated CNN-, LSTM-, and Transformer-based predictors for workload forecasting [37]. These works demonstrate the potential of forecasting for proactive resource management; however, they are predominantly ground-based or cloud-centric and do not explicitly address UAV mobility or aerial MEC constraints.

Overall, existing UAV-assisted MEC solutions predominantly focus on reactive task offloading and resource allocation driven by instantaneous system observations. Comprehensive surveys, such as that of Ismail et al. [38], further emphasize the lack of proactive and virtualization-aware resource scheduling mechanisms in current DRL-based MEC frameworks. Advanced scenarios, including semantic-aware optimization in IRS/UAV-enabled MEC networks [39] and intelligent offloading strategies for performance enhancement, have also been explored. Yet, they remain reactive and overlook resource virtualization and demand forecasting. From the surveyed literature, two key gaps emerge: (i) virtualization is rarely treated as a first-class orchestration mechanism in UAV-MEC systems, and (ii) proactive CPU and memory demand forecasting under realistic UAV mobility patterns is largely unexplored. This work addresses these gaps by integrating lightweight GRU-based CPU and memory prediction with a virtualization-driven orchestration framework, enabling proactive and scalable resource allocation in UAV-assisted MEC environments. Table I summarizes representative state-of-the-art methods and highlights the positioning of the proposed approach.

III. Proposed Methodology

The proposed framework, depicted in Fig. 1, presents the MEC Framework for UAV Management with Central Orchestrator. This framework underpins our strategy to improve UAV collision avoidance in an MEC environment. Using deep learning-driven virtualized resource allocation, our system enables real-time decision-making for UAVs navigating complex and dynamic settings.

The framework, illustrated in Fig. 1, encompasses three main components: UAVs, MEC servers, and a Central Orchestrator.

UAVs are central components of our system, operating as autonomous aerial vehicles equipped with various sensors such as GPS, lidar, and cameras. These sensors enable UAVs to gather data about their environment, identify potential collision risks, and communicate relevant information to nearby UAVs

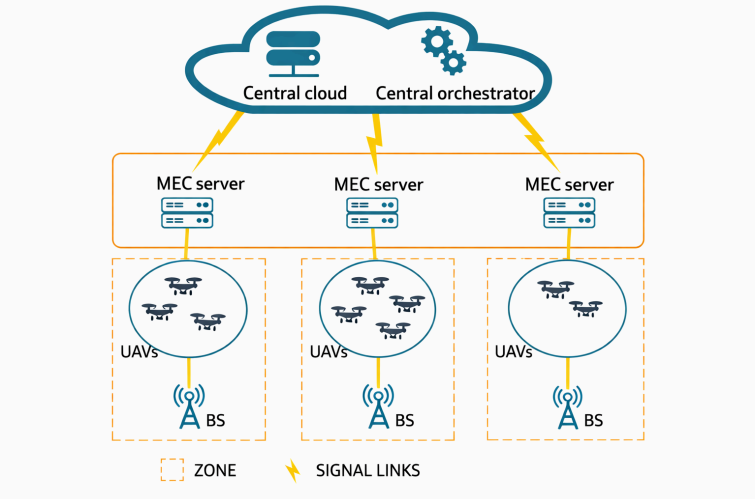


Fig. 1. MEC-enabled architecture illustrating UAV resource management and virtualization mechanisms.

and MEC servers using protocols designed for edge computing communication [41]. MEC Servers are strategically positioned at the edge of the network, just below the Edge Orchestrator; these servers play a crucial role in MEC [21]. They are tasked with hosting and executing applications and services, including those related to UAV management. Our specific focus is CDA systems for UAVs, which facilitate the anticipation and detection of road hazards. Each Edge Server node covers a specific geographical area, collecting data from UAVs within that region. This data contains important information gathered from drone sensors, like speed, location, altitude, and more. MEC servers bring the computing power closer to UAVs, which is especially important for heavy tasks like collision avoidance. Instead of sending raw sensor data all the way to distant cloud servers, UAVs can process it quickly on nearby MEC servers. This not only speeds up data handling but also strengthens the real-time performance of the CDA system, making UAV operations safer.

An integral part of the MEC architecture, the Central Orchestrator is responsible for the effective administration and coordination of all network services and resources. This software-driven entity can run either on dedicated hardware or as a cloud-based service. Its strength comes from the use of DL models, which guide dynamic decisions about resource allocation and virtualization. By analyzing factors such as application demands, available resources, network conditions, and latency requirements, the orchestrator determines the necessity of virtualization and how to best distribute resources to the right places as needed, depending on our algorithm IV-D. These models also constantly look at incoming data in real time, which lets the orchestrator change and adapt resource allocation on the fly. In this process, it is important to set up a feedback loop so that the orchestrator may learn and change its judgments about how to assign resources depending on the results and performance of resources that have already been allocated [2].

In our implementation, the orchestrator acts as an intelligent resource management entity that leverages deep learning

TABLE I
COMPARISON OF REPRESENTATIVE WORKS ON RESOURCE OPTIMIZATION IN UAV-ASSISTED MEC SYSTEMS

Reference	*Scenario*	*Method*	*Pred.*	*Virt.*	*Main Limitation*
Zakaryia et al. [29]	Multi-UAV MEC	MADRL	No	No	Reactive, not predictive
Wang et al. [30]	UAV MEC	Optimization	No	No	Instant decisions
Cui et al. [26]	Joint optimization	MADRL	No	No	No virtualization
Hwang et al. [31]	Multi-UAV MEC	MADRL	No	No	No forecast
Liu et al. [32]	UAV MEC	Lyapunov + DDPG	No	No	Focused energy
Xu et al. [33]	UAV MEC	MADRL	No	No	No proactive vision
Song et al. [16]	UAV-MEC	Survey	No	No	Conceptual analysis without implementation
Brik et al. [18]	IoV MEC	LSTM	Yes	No	Limited to ground vehicles; no virtualization
Lilhore et al. [20]	Edge MEC	DL (cybertwin)	Yes	Partial	Focuses mainly on energy efficiency
Seliem et al. [25]	UAV-MEC	Heuristic	No	No	Ignores resource virtualization
Ouahouah et al. [27]	UAV-MEC	DRL	No	No	High training complexity and energy-oriented design
Jiang et al. [35]	Large-scale MEC	Autoencoder + DRL	No	No	Not tailored for UAV-assisted scenarios
Wang et al. [36]	Dynamic MEC	GNN	No	No	High computational complexity; no UAV evaluation
Kamble et al. [37]	Cloud data centers	DL	Yes	Yes	No MEC or UAV constraints
Hou et al. [40]	UAV-Vehicular MEC	Optimization	No	No	Scalability issues in dense environments
This work	*UAV-MEC*	*GRU-Virtualization*	*Yes*	*Yes*	*No RL-based optimization*

techniques to dynamically optimize resource virtualization. These techniques enable the orchestrator to make data-driven decisions, ensuring efficient allocation of resources to meet the specific requirements of MEC applications and services while maintaining low latency.

In this work, we applied the same workflow to two datasets: OpenSky and AU-AIR. For OpenSky, we filtered the data to aircraft operating over Washington State and used their mobility traces as a proxy for UAV trajectories in our MEC allocation study. This preprocessing step was introduced to construct a large-scale mobility scenario for evaluating MEC placement, workload prediction, and virtual resource allocation under realistic movement patterns. It is therefore part of the dataset preparation and simulation process rather than the virtualization mechanism itself. Following a series of meticulously planned steps, we ensure comprehensive coverage, adaptability to dynamic changes in UAV distribution, and compliance with local regulations. AU-AIR, in contrast, offers UAV mobility traces in a smaller, controlled environment. We derived CPU and memory requirements from motion dynamics. This dual approach allows us to validate our methodology for feature extraction, MEC allocation with K-means and Voronoi, and deep learning prediction to optimize resources in MEC. The methodology consists of four main stages:

A. Feature Extraction

- For the OpenSky dataset, we extended the raw air traffic traces by generating CPU and memory requirements corresponding to UAV resource demand at each time step. The features therefore, included: CPU usage, memory usage, geographical position (longitude, latitude, altitude), and timestamp.
- For the AU-AIR dataset, we used the available attributes, including image resolution, object detection load, velocity, orientation, and altitude to similarly generate CPU and memory requirements for the MEC system using predefined estimation formulas. This ensured consistency in feature space between the two datasets.

B. MEC Allocation (Clustering and Visualization)

- To model the association of UAVs with MEC servers, we applied K-means clustering to the UAV positions in each dataset. The resulting clusters represent the geographical coverage of MEC servers.

C. Deep Learning Prediction

We employ sequence prediction models (RNN, LSTM, GRU) to forecast future CPU and memory requirements at time t+1, based on past temporal sequences to predict UAV resource demand (CPU and memory) at time t+1

To ensure a fair comparison between datasets, we used identical hyperparameters (batch size, number of epochs, optimizer, and learning rate), the same data split, and the same normalization strategy. Training and evaluation were performed separately on each dataset, but under the same experimental settings, which allows for a direct comparative analysis.

D. Virtualized Resource Allocation

After forecasting UAV resource demand (CPU and memory) at time t+1 using DL models, we applied virtualization techniques to dynamically allocate MEC resources. The predicted workloads were mapped to virtualized CPU and memory units on MEC servers to ensure efficient utilization and scalability.

IV. Experimental Design

A. Dataset Preparation and Feature Engineering

To evaluate the proposed methodology, we relied on two complementary datasets: OpenSky and AU-AIR. The OpenSky dataset provides large-scale real-world aircraft trajectories. For our experiments, we restrict the analysis to a specific geographical region (Washington State) to ensure a controlled and consistent spatial scope for MEC deployment modeling. In contrast, the AU-AIR dataset offers UAV trajectories in a controlled environment with perception data, enabling more detailed modeling of computational demand. This dual perspective allowed us to test the framework under both large-scale, unconstrained conditions (OpenSky) and small-scale, perception-intensive scenarios (AU-AIR).

1) OpenSky CPU and Memory Requirement Estimation: The CPU requirement at time t (in MIPS) was defined as:

$$\mathrm{CPU}_t = \gamma_1 V_t + \gamma_2 \frac{H_t}{1000} + \gamma_3 \Delta t. \tag{1}$$

while the memory requirement (in MB) was estimated as:

$$\mathrm{MEM}_t = \delta_1 N_{\mathrm{UAV},t} + \delta_2 V_t + \delta_3. \tag{2}$$

The coefficients were selected based on the statistical characteristics of the OpenSky Washington State dataset and the expected impact of UAV mobility features on MEC workload dynamics.

$$\boldsymbol{\gamma} = [25,\ 18,\ 5], \qquad \boldsymbol{\delta} = [20,\ 10,\ 50]. \tag{3}$$

The coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ capture the intuition that velocity (V_t) and altitude (H_t) increase computational complexity for collision avoidance and communication, whereas memory is primarily influenced by the number of concurrent UAVs and system overhead [42], [43].

- *Velocity (V_t)*: Higher speed increases avoidance prediction complexity $V_t \approx 50$–300 m/s (mean ≈ 175 m/s)
- *Altitude (H_t)*: Greater altitude requires larger communication buffers and extended sensor fusion $H_t \approx 1000$–$35\,204$ m (mean ≈ 18 km, normalized: $H_t/1000 \approx 18$) Greater altitude requires larger communication buffers and extended sensor fusion operations, contributing significantly to the workload
- *Update interval (Δt)*: More frequent state updates increase processing overhead $\Delta t \approx 1$–5 s (typical value: 1 s)reflects periodic state reconstruction costs

This distribution demonstrates that UAV velocity is the dominant workload driver ($\approx 96\%$), which is physically justified as faster aircraft require more frequent trajectory predictions and collision avoidance computations.

For memory requirements, the model reflects the following drivers:

- *Number of UAVs active ($N_{UAV,t}$)* : State caching and trajectory history for each active UAV. The OpenSky dataset shows $N_{\mathrm{UAV},t}$ ranges from 15 to 20 concurrent aircraft at the same time and area.
- *Velocity-dependent buffers (V_t)* : Faster UAVs require larger velocity history and prediction buffers.
- *Fixed overhead*: Framework and system overhead, which represents the minimum amount of memory reserved for the basic operation of MEC services and virtualization, even under light load.

2) AU-AIR CPU and Memory Requirement Estimation: In contrast, the AU-AIR dataset allows us to model perception-induced resource demand. In this case, we derived CPU and memory requirements from motion dynamics, using predefined estimation formulas. This ensured that the dataset captured both mobility and computational aspects of UAV operations in a realistic yet constrained scenario. For AU-AIR, we designed a heuristic estimation model in which CPU requirements grow with object density, velocity, angular maneuvers, and altitude, while memory requirements depend mainly on image resolution, perception complexity, and constant overhead. For each UAV state i at time t, we extracted the following parameters: the number of detected objects $N_{obj,t}$, the image area A_{img}, the UAV velocity magnitude V_t, the mean Euler angle amplitude $A_{angle,t}$, and the altitude H_t

- $N_{\mathrm{obj},t}$: number of detected objects (bounding boxes).
- $A_{\mathrm{img}} = W_{\mathrm{img}} \times H_{\mathrm{img}}$: image area in pixels, where W_{img} and H_{img} denote image width and height, respectively.
- $V_t = \sqrt{v_{x,t}^2 + v_{y,t}^2 + v_{z,t}^2}$: UAV velocity magnitude.
- $A_{\mathrm{angle}}(t) = \frac{1}{3}\big(|\phi_t| + |\theta_t| + |\psi_t|\big)$: mean Euler angle amplitude.
- H_t: UAV altitude.

Then,the *CPU requirement* (in MIPS) is defined as:

$$\begin{aligned}\mathrm{CPU}_t = \alpha_1 N_{\mathrm{obj},t} + \alpha_2 \frac{A_{\mathrm{img}}}{10^6} + \alpha_3 V_t \\ + \alpha_4 A_{\mathrm{angle},t} + \alpha_5 \frac{H_t}{10^4}.\end{aligned} \tag{4}$$

The *memory requirement* (in MB) is given by:

$$\mathrm{MEM}_t = \beta_1 \frac{A_{\mathrm{img}}}{10^6} + \beta_2 N_{\mathrm{obj},t} + \beta_3 V_t + \beta_4. \tag{5}$$

The coefficients were empirically chosen as:

$$\boldsymbol{\alpha} = [15,\ 20,\ 60,\ 40,\ 10], \qquad \boldsymbol{\beta} = [50,\ 10,\ 5,\ 100]. \tag{6}$$

The coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ were selected based on the statistical characteristics of the AU-AIR dataset and the expected impact of perception and motion features on computational workload.

- *Number of objects ($N_{obj,t}$)*: Object detection complexity. AU-AIR dataset shows $N_{obj,t} \approx 5$–50 objects per frame (mean ≈ 25 objects).
- *Image area ($A_{img}/10^6$)*: Higher image resolution increases perception inference time and memory usage. AU-AIR contains high-resolution frames with image areas typically ranging between approximately 2.0 and 4.0 megapixels.
- *Velocity (V_t)*: Higher UAV velocity increases motion prediction complexity and temporal buffering requirements. AU-AIR velocity ranges: $V_t \approx 1$–15 m/s (mean ≈ 5 m/s).

- *Angular maneuvers ($A_{angle,t}$)*: Large pitch, roll, and yaw variations increase stabilization and trajectory correction complexity. AU-AIR: $A_{angle,t} \approx 5$–$45°$ (mean $\approx 20°$).
- *Altitude ($H_t/10^4$)*: Greater altitude affects sensor fusion complexity. AU-AIR: $H_t \approx 5$–120 m (mean ≈ 50 m).

This distribution shows that angular maneuvers and object detection drive the workload, reflecting the intensive perception processing in AU-AIR.

This model captures both perception-related complexity (object density, image resolution) and motion dynamics (velocity, angular maneuvers, altitude), thereby providing realistic CPU and memory requirements for UAV operations in AU-AIR. Finally, to ensure consistency, both datasets were preprocessed with feature normalization (StandardScaler) and an 80/20 train–test split. This common pipeline ensured fair and comparable conditions for model training and evaluation.

B. MEC Distribution and Clustering

To evaluate UAV resource allocation in an edge-enabled environment, we designed a simulation framework for MEC node distribution. This process comprised three main stages: (i) spatial distribution of UAVs, (ii) clustering to approximate MEC server locations, and (iii) visualization through Voronoi diagrams.

1) Spatial Distribution of UAVs: The spatial distribution of UAVs was obtained from the mobility traces described in Section V. Each UAV i is represented by its geographic coordinates at time t:

$$\mathbf{u}_i^t = (\mathrm{lat}_i^t, \mathrm{lon}_i^t), \quad i = 1, \ldots, N. \tag{7}$$

where N denotes the total number of UAVs, the resulting point set $\mathcal{U}^t = \{u_1^t, \ldots, u_N^t\}$ defines the UAV spatial configuration at a given time.

2) Clustering for MEC Placement: To approximate MEC server positions, we applied the k-means clustering algorithm on $\mathcal{U}^t$. The algorithm partitions the UAVs into k disjoint clusters $\{C_1, \ldots, C_k\}$ by minimizing the within-cluster variance:

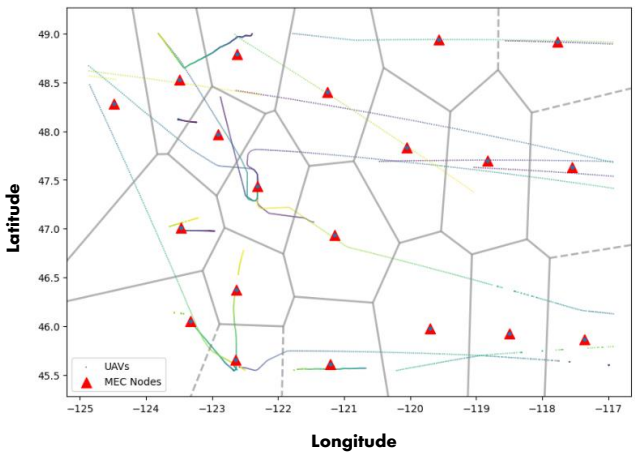
$$\min_{\{C_j\}} \sum_{j=1}^{K} \sum_{\mathbf{u}_i \in C_j} \left\| \mathbf{u}_i^t - \boldsymbol{\mu}_j \right\|^2. \tag{8}$$

where μ_j is the centroid of cluster C_j. These centroids $\{\mu_1, \ldots, \mu_k\}$ were interpreted as candidate MEC server locations.

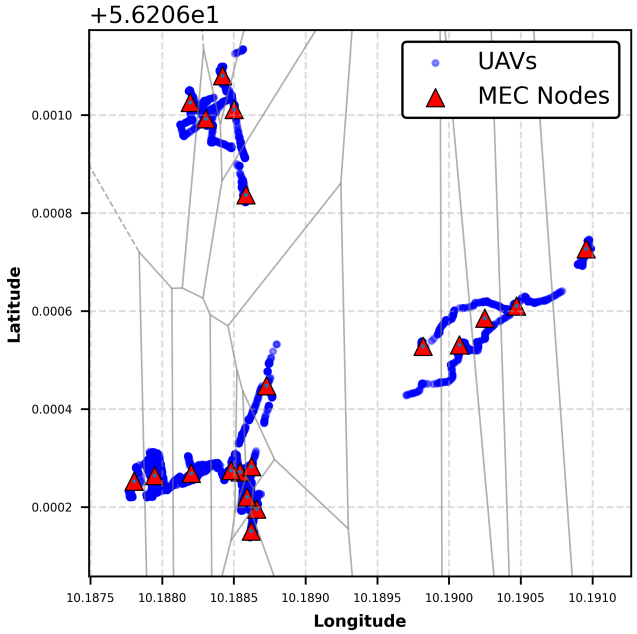
In our experiments, we set $k = 20$, meaning that 20 MEC servers were deployed in both the OpenSky and AU-AIR scenarios. This value was chosen to provide a representative trade-off between UAV coverage granularity and computational feasibility in simulation, while ensuring consistency across datasets.

3) Visualization with Voronoi Diagrams: To analyze spatial coverage, we constructed a Voronoi tessellation based on the cluster centroids $\{\mu_1, \ldots, \mu_k\}$. The Voronoi cell associated with centroid μ_j is defined as:

$$V_j = \left\{ \mathbf{u} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{u} - \boldsymbol{\mu}_j\| \leq \|\mathbf{u} - \boldsymbol{\mu}_m\|, \; \forall m \neq j \right\}. \tag{9}$$



(a) Voronoi for OpenSky dataset



(b) Voronoi for AU-AIR dataset

Fig. 2. Voronoi diagrams illustrating UAV trajectories and corresponding MEC server deployment obtained using K-means clustering for the (a) OpenSky and (b) AU-AIR datasets.

Each cell V_j represents the coverage region of MEC node j, ensuring that every UAV is associated with its geographically nearest MEC server.

We generated Voronoi diagrams for both datasets. In the OpenSky case, the tessellation illustrates large-scale UAV distributions across Washington State, while in the AU-AIR case, it reflects UAV trajectories in a smaller, perception-driven environment. These visualizations (Figs. 2a and 2b) provide a geometric partitioning of UAV-to-MEC associations, highlighting coverage efficiency, potential overlaps, and underserved regions.

By combining k-means clustering with Voronoi analysis, we capture both the statistical distribution of UAVs (through clustering) and the geometric efficiency of MEC coverage

(through tessellation). This dual perspective enables a more rigorous assessment of MEC deployment strategies in both large-scale and controlled UAV scenarios.

C. Deep Learning-based Resource Prediction

The objective of this stage is to forecast UAV computational resource requirements, specifically CPU and memory demand, at the next time step $(t+1)$. Accurate prediction enables proactive MEC resource allocation, thereby minimizing service interruptions and avoiding overload situations.

We employed recurrent neural network architectures, including RNN, LSTM, and GRU, with a particular emphasis on the GRU due to its efficiency in capturing sequential dependencies. The models take as input a multivariate time series composed of: (i) historical CPU and memory usage, and (ii) UAV mobility features such as position, velocity, and altitude. The output layer predicts the required CPU and memory resources at time step $(t+1)$:

$$[\mathrm{CPU}_{t+1},\ \mathrm{MEM}_{t+1}].$$

The proposed models were applied consistently to both datasets (OpenSky and AU-AIR), ensuring that prediction was evaluated under large-scale real-world trajectories as well as controlled UAV perception-driven scenarios. This dual validation demonstrates the generalizability of the approach across heterogeneous UAV environments.

All models were trained under identical experimental conditions: feature normalization (StandardScaler), an 80/20 train–test split, and comparable hyperparameter settings (batch size, learning rate, and number of epochs). Table II summarizes the key implementation parameters.

Finally, to operationalize the predictions, we adopted a virtualization-based resource management scheme in which virtual CPUs and memory units are dynamically allocated or released in response to forecasted workloads. This mechanism ensures continuous provisioning of UAV services while maintaining efficient utilization of MEC resources.

TABLE II
IMPLEMENTATION PARAMETERS

Parameter	*Value*
	Data Overview
Datasets	OpenSky, AU-AIR
Utility	UAV mobility and workload modeling
Evaluation mode	Independent cross-dataset evaluation
Time representation	Discrete time series
	Deep Learning
Deep learning Tool	TensorFlow
Activation functions	ReLU (hidden layers)
	Softmax (output layer)
Optimizer	Adam
Loss function	sparse_categorical_crossentropy (SCCE)
Batch size	1000 samples
Number of epochs	20 epochs
Evaluation metrics	MAE, RMSE, MAPE, R^2
	MEC
Number of MEC Hosts	20 hosts

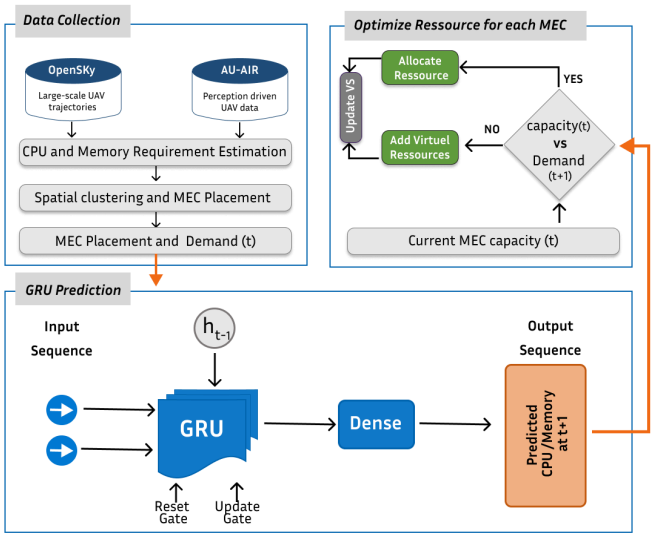


Fig. 3. Overall architecture and general mechanism of the proposed DL-driven virtualised resource allocation framework. The GRU model predicts future CPU and memory demand, which is then used by the MEC orchestrator to proactively adjust virtual resource provisioning in a closed-loop manner.

D. Virtualised Resource Allocation

Once the CPU and memory requirements are predicted, the next stage is to map these demands onto the available MEC servers through virtualisation. The objective is to dynamically provision resources in a way that balances workload across nodes, avoids service interruptions, and maximises overall infrastructure utilisation.

In this framework, MEC resources are abstracted into pools of virtual CPUs (vCPUs) and virtual memory (vMemory). Based on the predicted workload, the orchestrator dynamically adjusts resource allocation by creating, resizing, or releasing virtual instances. Figure 3 illustrates the overall architecture and control mechanism of the proposed framework, providing a high-level view of the closed-loop interaction between GRU-based workload prediction and proactive virtualised resource orchestration at the MEC layer. This proactive strategy enables UAV applications to be provisioned with adequate computational resources while avoiding unnecessary overprovisioning of the underlying physical infrastructure. Algorithm IV-D formalises this process. The orchestrator maintains a virtualisation state (VS), which records the current mapping of UAV demands to MEC resources. At each time step, the orchestrator (i) gathers UAV mobility and resource forecasts, (ii) predicts which UAVs are likely to connect to each MEC node, and (iii) evaluates whether the node's physical capacity is sufficient. If adequate resources are available, they are directly allocated; otherwise, additional virtual resources are instantiated to satisfy demand. Conversely, unused virtual allocations are decommissioned, ensuring that MEC servers adapt elastically to the evolving UAV traffic patterns.

This design tightly couples demand forecasting with virtualised resource management, enabling a closed-loop MEC system that is both adaptive and resilient. By decoupling physical hardware constraints from service delivery, the approach guarantees scalability while maintaining high quality

Algorithm 1 Virtualised Resource Management in MEC

Input: UAV set $\mathcal{U}$ with predicted demands $\{CPU_u, MEM_u\}$ and positions; MEC nodes $\mathcal{N}$ with resource capacities
Output: Updated virtualisation state VS
Initialisation: Load current VS and MEC capacities
for each MEC node $n \in \mathcal{N}$ **do**
  Predict UAVs likely to associate with n
  Update prediction list $\mathcal{U}_n$
end for
for each UAV $u \in \mathcal{U}_n$ **do**
  if capacity of n satisfies $\{CPU_u, MEM_u\}$ **then**
    Allocate resources to u
  else
    Instantiate additional virtual resources for u
  end if
  Update VS accordingly
end for
Remove obsolete allocations for UAVs not in $\mathcal{U}_n$

of service for UAV workloads.

E. Evaluation Metrics and Protocol

The evaluation protocol was designed to capture both the predictive accuracy of the deep learning models and their robustness under varying data conditions. Performance was analyzed along four complementary axes:

- *Model-level comparison:* To evaluate the prediction performance of the proposed models, we compared RNN, LSTM, and GRU architectures using four regression metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2). All metrics are computed on the held-out test set to ensure unbiased evaluation.
- *Resource demand fidelity:* To evaluate the performance of resource demand prediction, the required and predicted CPU usage were compared over time. For clarity, a single MEC host is selected as a representative example, while similar trends were observed across all MEC nodes.
- *System-level efficiency:* is defined as the ratio between the actually utilized resources and the total allocated MEC resources over a given time window. We compute efficiency separately for CPU and memory to capture resource-specific utilization behavior under different orchestration strategies: baseline allocation, deep learning–only prediction, and the proposed joint Virtualization + DL framework.
- *Cross-dataset consistency:* All experiments were performed independently on the OpenSky and AU-AIR datasets, under identical preprocessing, training, and hyperparameter settings. This ensures that differences in results stem from data characteristics rather than training bias, thereby enabling a fair cross-context comparison.

V. Results and Analysis

This section presents the results obtained from our experimental evaluation, focusing on three complementary aspects: (i) Model-Level Comparison, (ii) per-node prediction accuracy of resource demand, and (iii) cross-dataset analysis to assess generalizability. By following the protocol defined earlier, we ensure that results are comparable across models, sample sizes, and datasets.

A. Model-Level Comparison

To evaluate predictive performance, we compared RNN, LSTM, and GRU architectures using standard regression metrics: MAE, RMSE, MAPE, and coefficient of determination R2.

Results on OpenSky Dataset: Table III shows the performance comparison of RNN, GRU, and LSTM models for CPU and memory requirement prediction of the Opensky dataset. We evaluate the result using different metrics to measure both prediction error and model fitting quality, which indicate that GRU achieves the best overall performance across most evaluation metrics.

TABLE III
Performance Comparison of RNN, GRU, and LSTM Models on the Opensky dataset

Model	Target	RMSE	MAE	MAPE (%)	R^2
RNN	CPU R	252.6198	107.6885	4.2979	0.9208
	Memory R	100.3526	44.7626	1.1460	0.9207
GRU	CPU R	228.3029	65.2809	2.8544	0.9543
	Memory R	93.2754	32.7699	0.8637	0.9530
LSTM	CPU R	242.1208	110.3251	6.2820	0.9240
	Memory R	96.1750	41.7977	1.1273	0.9222

Results on AU-AIR Dataset: Similarly, Table IV reports the results on the AU-AIR dataset, confirming that GRU consistently provides more stable and accurate predictions compared to RNN and LSTM architectures.

TABLE IV
Performance Comparison of RNN, GRU, and LSTM Models on the Au Air dataset

Model	Target	RMSE	MAE	MAPE (%)	R^2
RNN	CPU R	93.0337	71.4816	5.5959	0.8745
	Memory R	44.3148	31.1930	6.7264	0.8307
GRU	CPU R	92.8335	71.9800	5.2544	0.9546
	Memory R	44.6225	31.4510	5.8637	0.9172
LSTM	CPU R	94.6425	72.5204	5.6164	0.9087
	Memory R	45.5728	31.8968	6.8080	0.8472

These findings confirm that GRU achieves the most favorable trade-off between model complexity and predictive performance, making it the most suitable candidate for UAV resource demand forecasting in MEC environments.

In fact, the improved performance of GRU is due to its gated architecture that efficiently captures temporal dependencies, while having lower computational complexity than LSTM. This allows for faster convergence and better generalization when UAV mobility patterns are heterogeneous, and the amount of training data is limited. On the other side, RNN suffers from unstable long-term dependency learning. LSTM

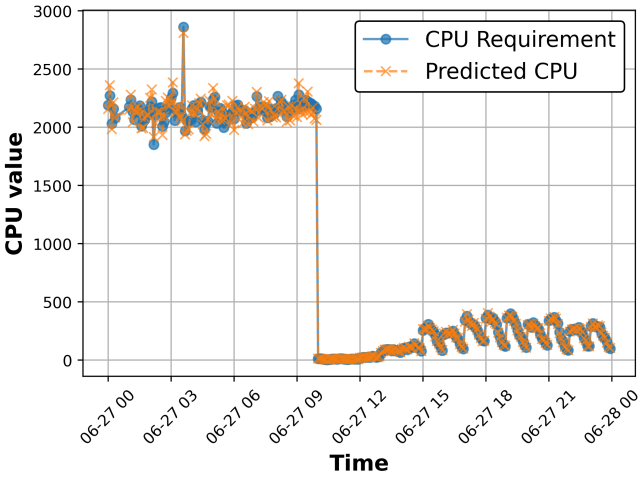
introduces a higher model complexity, which is not always beneficial for the considered workload prediction scenarios.

B. Resource Prediction Fidelity

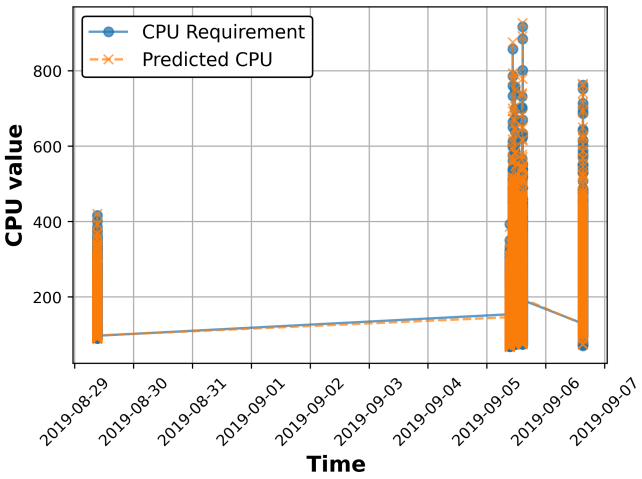
While aggregate scores capture global accuracy, they do not reveal how well predictions follow actual MEC-level workloads over time. To address this, we compared required versus predicted CPU and memory usage at representative MEC servers from both datasets. Importantly, we chose the MEC hosts that were provisioned with identical computational capacity; hence, variations across nodes reflect only the heterogeneity of UAV traffic demand rather than hardware differences. Due to space limitations, only one MEC node is illustrated; however, the observed trends are consistent across all nodes.

Figure 4 presents CPU demand tracking. In both the OpenSky and AU-AIR datasets, GRU-based forecasts closely follow the ground-truth trajectories, effectively capturing overall workload dynamics. Occasional deviations appear during abrupt traffic bursts, especially in high-density nodes, yet the model avoids systematic under-provisioning—an essential property for sustaining UAV service continuity.

Figure 5 illustrates the corresponding memory predictions. Here, GRU again aligns well with actual usage. In the AU-AIR dataset, cyclic patterns emerge due to recurring computational tasks; the GRU adapts effectively to these periodicities, stabilizing memory allocation without excessive over-provisioning. Together, these results demonstrate that GRU-based forecasting achieves high-fidelity temporal prediction across heterogeneous UAV environments. By ensuring high-fidelity predictions of CPU and memory demand, the forecasting models provide actionable input to the virtualization layer, enabling proactive resource orchestration and minimizing the risk of service disruption across MEC hosts.



(a) Required vs. Predicted CPU usage (OpenSky).



(b) Required vs. Predicted CPU usage (AU-AIR).

Fig. 4. Temporal fidelity of CPU usage prediction across representative MEC servers. GRU-based forecasts closely follow actual demand in both datasets, with minor deviations during abrupt workload spikes.

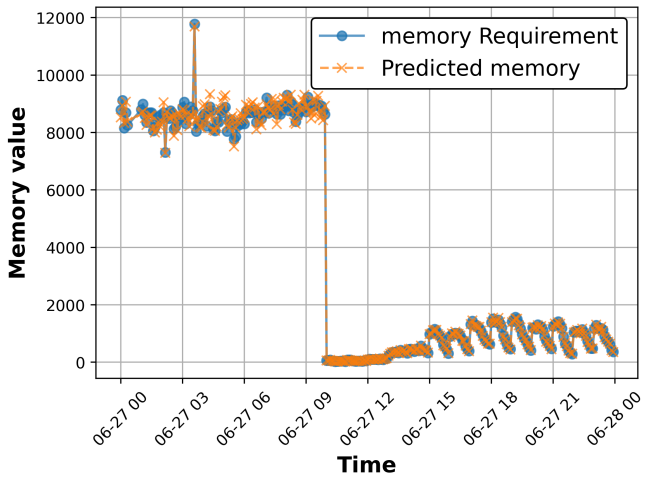
C. Comparative and Cross-Dataset Insights

To complement the model-level and node-level analyses, we performed a system-level comparison across three orchestration setups: (i) virtualization combined with deep learning (Virt+DL), (ii) deep learning without virtualization (DL only), and (iii) a baseline configuration without either. To validate the generality and robustness of our findings, this experiment was conducted independently on both the OpenSky and AU-AIR datasets, which exhibit distinct traffic patterns and workload characteristics. Figure 6a reports CPU efficiency across the three setups, while Figure 6b presents the corresponding memory efficiency. The Virt+DL configuration consistently outperformed alternative approaches across both datasets and metrics. Specifically, CPU efficiency exceeded 91% on both OpenSky (91.5%) and AU-AIR (92.0%), while memory efficiency surpassed 92% in both cases (OpenSky: 92.0%, AU-AIR: 91.5%). The DL-only setup achieved intermediate performance, with CPU efficiency ranging from 81.5% to 83.0% and memory efficiency between 82.0% and 83.3%. The baseline configuration exhibited substantially lower efficiency, with CPU performance consistently below 77.5% and memory efficiency at 74.5% (AU-AIR) to 77.0% (OpenSky). Crucially, the consistency of these efficiency improvements across both independent datasets—despite their different operational conditions—demonstrates that the observed gains are not artifacts of dataset-specific characteristics but rather reflect the fundamental benefits of integrating virtualization with predictive deep learning.

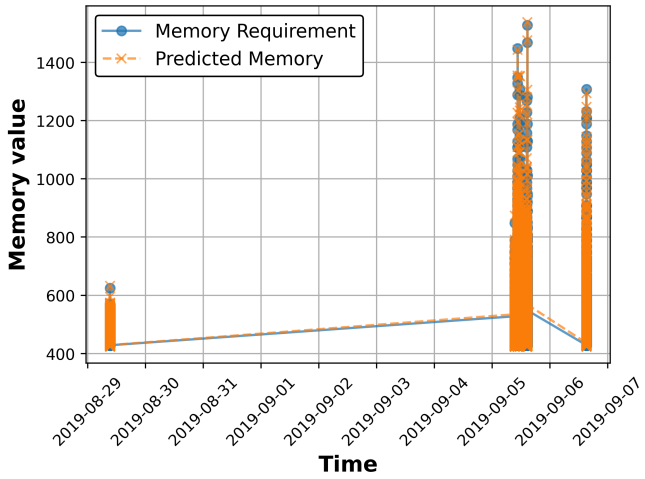
D. Practical Deployment Considerations and Feasibility

A practical deployment of the proposed framework would rely on a hierarchical MEC architecture in which UAVs operate as mobile sensing and edge devices, while computationally demanding tasks are offloaded to MEC servers deployed alongside cellular base stations. Resource management and service orchestration could rely on lightweight containerized microservices coordinated through edge-oriented platforms such as Kubernetes [44] or KubeEdge [45].

While a real deployment of the approach is beyond the scope of this study, a theoretical complexity can be computed

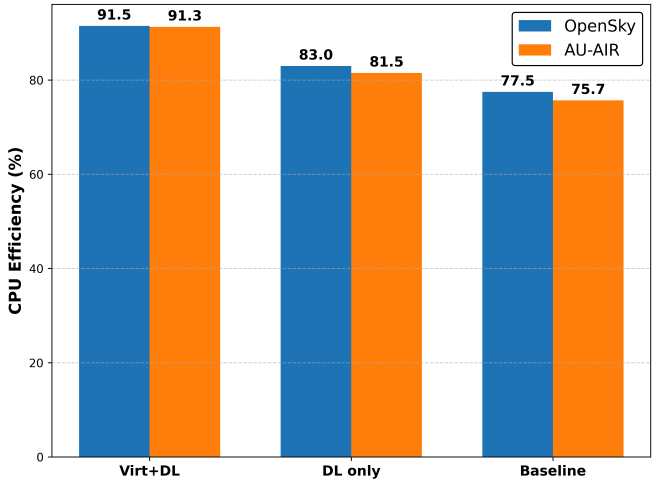


(a) Required vs. Predicted memory usage (OpenSky).

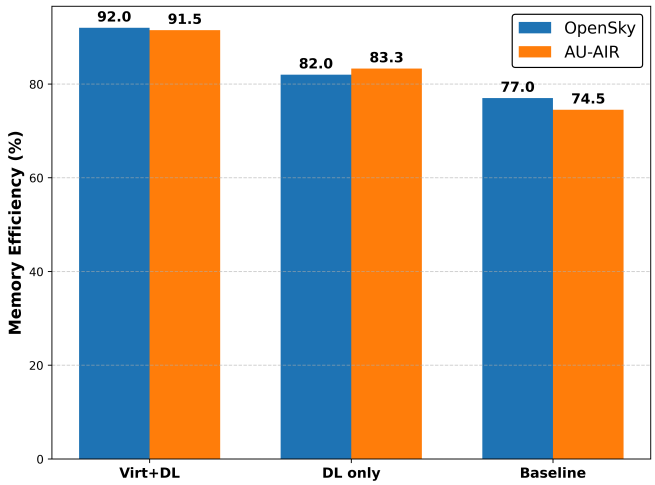


(b) Required vs. Predicted memory usage (AU-AIR).

Fig. 5. Temporal fidelity of memory usage prediction across representative MEC servers. GRU effectively adapts to both stable and cyclic demand patterns, ensuring robust allocation.



(a) CPU efficiency comparison across OpenSky and AU-AIR under the three setups (Virt+DL, DL only, Baseline).



(b) Memory efficiency comparison across OpenSky and AU-AIR under the three setups (Virt+DL, DL only, Baseline).

Fig. 6. Comparative evaluation of orchestration setups: (a) CPU efficiency and (b) Memory efficiency across OpenSky and AU-AIR datasets. The Virt+DL configuration consistently achieves the highest efficiency, demonstrating the joint benefit of virtualization and predictive deep learning.

across three stages: First, the feature extraction scales linearly with the number of UAV observations, leading to $\mathcal{O}(NT)$ (N the number of UAVs and T the number of timesteps). The K-means requires $\mathcal{O}(NKId)$ (I the iteration number and d the spatial dimension); the subsequent Voronoi diagram generation is considered negligible for a small value of K. The dominant computational cost arises from the training of the GRU model (almost quadratic). Nevertheless, this step is done once and before deployment, giving a final linear virtualization orchestration mechanism.

VI. Conclusion

In this research, we introduced a system powered by virtualization and improved by DL to optimize CPU and memory management in MEC contexts that facilitate UAV applications. The virtualization layer ensured flexible allocation and isolation of resources, while the predictive models with a focus on GRU architecture enabled accurate forecasting of resource demand. These two components worked synergistically to create a balanced strategy: DL made the resource demand more predictable, while virtualization made the system more flexible.

We demonstrated the robustness and generalizability of the proposed method by evaluating it on two real-world datasets, OpenSky and AU-AIR. At the model level, GRU consistently outperformed RNN and LSTM baselines, achieving higher R^2 values and faster, more stable convergence. At the system level, resource prediction fidelity analyses confirmed that the GRU-based forecasts closely tracked the actual CPU and memory utilization on MEC nodes, effectively preventing both under- and over-provisioning. Furthermore, comparative evaluations demonstrated that the combined virtualization and deep learning approach delivers superior overall efficiency compared to both standalone deep learning solutions and baseline configurations across all CPU and memory metrics.

These results strongly validate that predictive orchestration is both feasible and effective in MEC-enabled UAV scenarios. The cross-dataset validation further confirms the generalizability of the proposed method across different traffic patterns and workloads.

Nevertheless, some limitations remain. The proposed framework was evaluated in a simulated MEC environment, and large-scale real-world deployment aspects require further investigation, such as orchestration overhead, scalability under dense UAV traffic, and adaptation to highly dynamic mobility patterns. In addition, although the GRU model provides accurate predictions with moderate complexity, the computational cost of continuous forecasting and real-time virtualization-based orchestration may become significant in resource-constrained edge scenarios.

To address these limitations, future work will explore several directions. First, we will incorporate additional contextual features such as UAV mobility patterns, communication constraints, and network topology to improve prediction accuracy. Second, we will develop reinforcement-learning-based orchestration systems capable of adapting online to dynamic workload fluctuations. Third, we will investigate online and federated learning approaches to enable distributed model training across heterogeneous edge nodes without centralizing data. Finally, we will explore lightweight time-series forecasting models, including Temporal Convolutional Networks (TCN) and transformer-based architectures, to further enhance prediction performance in highly dynamic UAV-assisted MEC environments.

ACKNOWLEDGMENTS

This work is supported by the General Direction of Scientific Research and Technological Development (DGRSDT) - Algeria, and performed under the PRFU Project: C00L07N030120220002.

REFERENCES

[1] M. I. Khattak, H. Yuan, A. Khan, A. Ahmad, I. Ullah, and M. Ahmed, "Evolving multi-access edge computing (mec) for diverse ubiquitous resources utilization: A survey: Mi khattak et al." *Telecommunication Systems*, vol. 88, no. 2, p. 71, 2025.

[2] 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, 2017.

[3] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[4] X. Xia, S. M. M. Fattah, and M. A. Babar, "A survey on uav-enabled edge computing: Resource management perspective," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–36, 2023.

[5] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (uavs): Practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, no. 1, pp. 109–137, 2023.

[6] M. Antonakakis and M. Zervakis, "Advances in unmanned aerial vehicle-based sensing and imaging," p. 8094, 2024.

[7] T. Wang, Y. Shi, F. Deuser, S. Huang, G. Hu, S. Liu, Z. Zheng, and R. Zimmermann, "The 3rd workshop on uavs in multimedia: Capturing the world from a new perspective," in *Proceedings of the 3rd International Workshop on UAVs in Multimedia: Capturing the World from a New Perspective*, 2025, pp. 1–3.

[8] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48 572–48 634, 2019.

[9] M. C. Tatum and J. Liu, "Unmanned aerial vehicles in the construction industry," in *Proceedings of the Unmanned Aircraft System Applications in Construction, Creative Construction Conference, Primosten, Croatia*, 2017, pp. 19–22.

[10] J. Tang, S. Lao, and Y. Wan, "Systematic review of collision-avoidance approaches for unmanned aerial vehicles," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4356–4367, 2022.

[11] M. Hatfield, C. Cahill, P. Webley, J. Garron, and R. Beltran, "Integration of unmanned aircraft systems into the national airspace system-efforts by the university of alaska to support the faa/nasa uas traffic management program," *Remote Sensing*, vol. 12, no. 19, p. 3112, 2020.

[12] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Cellular-connected multi-uav mec networks: An online stochastic optimization approach," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6630–6647, 2022.

[13] F. Shirin Abkenar, P. Ramezani, S. Iranmanesh, S. Murali, D. Chulerttiyawong, X. Wan, A. Jamalipour, and R. Raad, "A survey on mobility of edge computing networks in iot: State-of-the-art, architectures, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2329–2365, 2022.

[14] Z. Liu, Y. Cao, P. Gao, X. Hua, D. Zhang, and T. Jiang, "Multi-uav network assisted intelligent edge computing: Challenges and opportunities," *China Communications*, vol. 19, no. 3, pp. 258–278, 2022.

[15] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 435–15 459, 2022.

[16] Z. Song, X. Qin, Y. Hao, T. Hou, J. Wang, and X. Sun, "A comprehensive survey on aerial mobile edge computing: Challenges, state-of-the-art and future directions," *Computer Communications*, vol. 191, pp. 233–256, 2022.

[17] ETSI (European Telecommunications Standards Institute), "Multi-access Edge Computing (MEC); Framework and Reference Architecture," ETSI, Tech. Rep. ETSI GS MEC 003 V3.1.1, March 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf

[18] B. Brik and A. Ksentini, "Toward optimal mec resource dimensioning for a vehicle collision avoidance system: A deep learning approach," *IEEE Network*, vol. 35, no. 3, pp. 74–80, 2021.

[19] K. Zairi, B. Brik, Y. Guellouma, and H. Cherroun, "Deep learning-driven resource allocation for mec-enabled uav collision avoidance system," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2024, pp. 1412–1417.

[20] U. K. Lilhore, S. Simaiya, S. Dalal, N. Faujdar, R. Alroobaea, M. Alsafyani, A. M. Baqasah, and S. Algarni, "Optimizing energy efficiency in mec networks: a deep learning approach with cybertwin-driven resource allocation," *Journal of Cloud Computing*, vol. 13, no. 1, p. 126, 2024.

[21] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2449–2494, 2022.

[22] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2021.

[23] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[24] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-oriented mec applications placement in a federated edge cloud architecture," in *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE, 2020, pp. 1–6.

[25] H. Seliem, M. H. Ahmed, R. Shahidi, and M. S. Shehata, "Delay analysis for drone-based vehicular ad-hoc networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–7.

[26] J. Cui, Y. Wei, J. Wang, L. Shang, and P. Lin, "Joint trajectory design and resource allocation for uav-assisted mobile edge computing in power convergence network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2025, no. 1, p. 4, 2025.

[27] S. Ouahouah, M. Bagaa, J. Prados-Garzon, and T. Taleb, "Deep-reinforcement-learning-based collision avoidance in uav environment," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4015–4030, 2021.

[28] F. Yu, D. Yang, F. Wu, Y. Wang, and H. He, "Resource optimization for uav-assisted mobile edge computing system based

on deep reinforcement learning," *Physical Communication*, vol. 59, p. 102107, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490723001106

[29] S. A. Zakaryia, M. Meaad, T. Nabil, and M. K. Hussein, "Task offloading and resource allocation for multi-uav asset edge computing with multi-agent deep reinforcement learning," *Computing*, vol. 107, no. 5, pp. 1–31, 2025.

[30] Z. Wang, Y. Ma, B. Hu, J. Du, Y. Yin, M. Isaac, and A. Stefanidis, "Joint offloading and resource allocation optimisation for uav-aided low-altitude economy mec system," in *2025 IEEE 5th International Conference on Computer Communication and Artificial Intelligence (CCAI)*. IEEE, 2025, pp. 545–550.

[31] S. Hwang, H. Lee, M. Kim, and I. Lee, "Multi-agent deep reinforcement learning for decentralized multi-uav mobile edge computing networks," *IEEE Internet of Things Journal*, 2025.

[32] J. Liu, X. Zhang, H. Zhou, X. Lei, H. Li, and X. Wang, "Lyapunov-based deep deterministic policy gradient for energy-efficient task offloading in uav-assisted mec," *Drones*, vol. 9, no. 9, p. 653, 2025.

[33] S. Xu, Q. Liu, C. Gong, and X. Wen, "Energy-efficient multi-agent deep reinforcement learning task offloading and resource allocation for uav edge computing," *Sensors*, vol. 25, no. 11, p. 3403, 2025.

[34] X. B. Zhai, S. Fu, C. Yi, Z. Liu, C. Dong, and C. W. Tan, "Deep reinforcement learning-based task offloading with collaborative inference in uav-assisted mobile edge computing networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 1, pp. 472–482, 2025.

[35] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, "Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale mec networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9278–9290, 2020.

[36] X. Wang, N. Cheng, L. Fu, W. Quan, R. Sun, Y. Hui, T. Luan, and X. S. Shen, "Scalable resource management for dynamic mec: An unsupervised link-output graph neural network approach," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023, pp. 1–6.

[37] T. Kamble, S. Deokar, V. Wadne, D. Gadekar, H. Vanjari, and P. Mange, "Predictive resource allocation strategies for cloud computing environments using machine learning," *Journal of Electrical Systems*, vol. 19, pp. 68–77, 12 2023.

[38] A. A. Ismail, N. E. Khalifa, and R. A. El-Khoribi, "A survey on resource scheduling approaches in multi-access edge computing environment: a deep reinforcement learning study," *Cluster Computing*, vol. 28, no. 3, p. 184, 2025.

[39] W. Zheng, P. Ren, and Q. Li, "Semantic aware intelligent optimization for irs/uav-enabled mec in wideband cognitive radio networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2025, no. 1, p. 50, 2025.

[40] Y. Hou, C. Wang, M. Zhu, X. Xu, X. Tao, and X. Wu, "Joint allocation of wireless resource and computing capability in mec-enabled vehicular network," *China Communications*, vol. 18, no. 6, pp. 64–76, 2021.

[41] E. Chen, J. Chen, A. W. Mohamed, B. Wang, Z. Wang, and Y. Chen, "Swarm intelligence application to uav aided iot data acquisition deployment optimization," *IEEE Access*, vol. 8, pp. 175 660–175 668, 2020.

[42] R. C. Shit and S. Subudhi, "Hierarchical federated graph attention networks for scalable and resilient uav collision avoidance," *arXiv preprint arXiv:2511.11616*, 2025.

[43] G. Wang, X. Wang, Z. Miao, Z. Liu, and X. Hu, "Communication-aided multi-uav collision detection and avoidance based on two-stage curriculum reinforcement learning," *Biomimetic Intelligence and Robotics*, p. 100253, 2025.

[44] S. Koksal, F. O. Catak, and Y. Dalveren, "Flexible and lightweight mitigation framework for distributed denial-of-service attacks in container-based edge networks using kubernetes," *IEEE Access*, vol. 12, pp. 172 980–172 991, 2024.

[45] A. Dubey, S. Sharma, K. Devi, S. P. Singh, B. Balusamy, and P. Samuel, *Addressing the Challenges in Federating Edge Resources*. Cham: Springer Nature Switzerland, 2026, pp. 17–33. [Online]. Available: https://doi.org/10.1007/978-3-031-96265-3_2

Khadidja Zairi is a PhD student in computer science at the Laboratoire d'Informatique et de Mathématique (LIM Lab), Amar Telidji University, Laghouat, Algeria. She received the Bachelor's degree in 2018 and the Master's degree in 2020 in Computer Science. Her research interests include deep learning, resource optimization, and MEC in UAV-enabled networks.

Bouziane Brik received the Engineer, M.Sc., and Ph.D. degrees in Computer Science from the University of Laghouat, Algeria, in 2010, 2013, and 2017, respectively. He is currently an Assistant Professor at the University of Sharjah, UAE, and affiliated with the University of Bourgogne, France. His research interests include 5G and beyond networks, resource management, and machine/deep learning for networking.

Younes Guellouma received the Engineer and Master's degrees from the University of Laghouat, Algeria, in 2006 and 2009, respectively, and the Ph.D. degree in 2017. He is an Associate Professor in the Department of Computer Science at the University of Laghouat. His research interests include artificial intelligence, optimization methods, and theoretical computer science.

Hadda Cherroun is a Full Professor in the Department of Computer Science at Amar Telidji University, Algeria. She received the Engineer, Master's, and Ph.D. degrees in Computer Science from USTHB, Algeria, in 1991, 1997, and 2007, respectively. Her research interests include algorithm design, machine learning, natural language processing, and AI-driven applications.