Monitoring and Smart Decision Architecture for DRONE-FOG Integrated Environment

Wendel Serra[1], **Warley Junior**[1], **Isaac Barros**[1], **Hugo Kuribayashi** [1], **João Carmona** [1]

[1]College of Computer and Electrical Engineering
Federal University of Southern and Southeastern Para, Maraba, PA - Brazil

{wendel.barros, wmvj, isaacx, hugo, jvictor}@unifesspa.edu.br

Abstract. *Due to the limited computing resources of drones, it is difficult to handle computation-intensive tasks locally, hence, fog-based computation offloading has been widely adopted. The effectiveness of an offloading operation, however, is determined by its ability to infer where the execution of code/data represents less computational effort for the drone, so that, by deciding where to offload correctly, the device benefits. Thus, this paper proposes MonDrone-Fog, a novel fog-based architecture that supports image offloading, as well as monitoring and storing the performance metrics related to the drone, wireless network, and cloudlet. It takes advantage of the main machine-learning algorithms to provide offloading decisions with high levels of accuracy, F1, and G-mean. We evaluate the main classification algorithms under our database and the results show that Multi-Layer Perceptron (MLP) and Logistic Regression classifiers achieve 99.64% and 99.20% accuracy, respectively. Under these conditions, MonDrone-Fog works well in dense forests when weather conditions are favorable and can be useful as a support system for SAR missions by providing a shorter runtime for image operations.*

1. Introduction

Unmanned Aerial Vehicles (UAVs), sometimes known as Unmanned Aircraft Systems, are flying machines, typically equipped with small-sized motors, batteries, and hardware computing platforms that provide overall command and control. This is supposed to be highly beneficial and game-changing, not only in military applications, but also in other civilian and strategic applications [Mukherjee et al. 2020]. They can be used in many civil applications due to their ease of deployment, high-mobility, ability to hover, and low maintenance cost. Such vehicles are being utilized to provide wireless coverage, monitor road traffic in real time, perform remote sensing, assist in search-and-rescue (SAR) operations, in addition to uses in surveillance and security, civil infrastructure inspection, and precision agriculture [Shakhatreh et al. 2019].

In most search-and-rescue scenarios using a tactical drone, a single UAV system works according to a series of steps. In the first step, the rescue team defines the area of interest, then the search operation begins with the remote sensing of the target area to find the victim, using a single UAV equipped with vision cameras. After that, real-time aerial images from the targeted area are sent to the Ground Control System (GCS) to collect evidence of the presence of a victim. These images are analyzed by the rescue team to direct the SAR operations optimally. In such a use case, cameras are mounted on the drone and, by applying face recognition and detection methods on the offloaded images, lost persons can be detected in real time in an efficient manner [Waharte and Trigoni 2010]. Due to the computational overhead required by such a use case and given some inherent limitations concerning hardware and energy supply of drones, the time is critical and any delay can result in dramatic consequences (e.g., human losses). Therefore, the image processing time collected by a drone is a challenging issue [Shakhatreh et al. 2019].

The problem of insufficient energy and computing resources can be addressed using computation offloading, i.e., processing-heavy tasks are sent for execution in the cloud, which, in turn, sends the results back to the device. Offloading computation from devices has several benefits. For instance, the battery life of resource-constrained end devices can be extended by avoiding complex local processing [Junior et al. 2019].

Clearly, offloading can involve edge computing platforms instead of (or, in addition to) the centralized cloud [Shahidinejad and Ghobaei-Arani]. For these reasons, in recent years, various novel paradigms have emerged, such as fog computing, mobile edge computing and mobile cloud computing [Premsankar et al. 2018]. Fog computing bridges the gap between the cloud and mobile devices by enabling computing, networking, storage, and data management on network nodes within the close vicinity of resource-constrained IoT devices [Giang et al. 2020]. In addition, fog computing facilitates mobility support, real-time interactions, location awareness, interoperability, scalability, ultra-low latency, high bandwidth, and agility [Yousefpour et al. 2019]. Its infrastructure can provide resources for IoT services at the network edge, which are called fog nodes. They can be resource-poor devices, such as access points, set-top-boxes, routers, switches, end devices, and base stations, or resource-rich machines, such as cloudlet, i.e., light-weight cloud servers that are typically one hop away from mobile devices; and microfog, a subset of fog node (e.g., two Raspberry Pis) enabled by fog cluster to handle data from a sensor linked to a specific IoT service [Torres Neto et al. 2019]. The gain of an offloading operation, however, is determined by its ability to infer where the execution of code/data will represent less computational effort for the drone, so that, by deciding *where to offload* correctly, the drone obtains a benefit [Yi et al. 2015, Shakarami et al. 2020].

To address the aforementioned challenges, the main contributions of this paper are as follows:

- A novel fog-based architecture, called MonDrone-Fog, that supports data offloading between drone and cloudlet. We use a face detection and recognition benchmarking application as an example to illustrate the proposed architecture.
- MonDrone-Fog feeds a new database with raw data related to the performance metrics for the drone, wireless network, and cloudlet. These data are pre-processed for the machine learning classifiers evaluation process and, thus, generate the ideal candidate model.
- We evaluate performance of multiple machine-learning classifiers from our own experimental database for the decision-making where to offload images correctly.

The rest of this paper is organized as follows: Related work is discussed in Section 2. Section 3 presents our proposed MonDrone-Fog architecture. Section 4 presents our evaluation of a real-world application. Finally, Section 5 concludes the paper and presents future directions that may be considered.

2. Related work

Previous works proposed solutions to address different aspects of fog computing and UAVs research. We have defined classified it into three features: (i) main goal, (ii) edge paradigm, and (iii) monitored metrics.

Main goal defines the real benefit of using the associated solution. Solutions like [Kalatzis et al. 2018], [Motlagh et al. 2017], [Tang et al. 2019], [Luo et al. 2015] and [Mohamed et al. 2017] are designed to solve problems of civil applications. In [Kalatzis et al. 2018], authors proposed to apply the edge and fog calculation principle to forest fire detection based on the drone. The three-layer ecosystem effectively combines powerful cloud computing resources, rich fog computing resources, and user-perception capabilities through a hierarchical structure. This cooperative method reduces the network transmission overhead and the delay of data processing, based on edge intelligence. The solution [Motlagh et al. 2017] demonstrates how UAVs can be used for crowd surveillance, based on face recognition. They also evaluate the video data processing offloading to a Mobile Edge Computing (MEC) node compared to the local processing of video data on board drone. The results demonstrate the efficiency of the MEC-based offloading approach in processing time, energy drain, and in detecting suspicious persons. In a nutshell, this paper demonstrates the benefits of using a single UAV system with IoT devices for crowd surveillance with the introduced video data processing offloading technique. The authors in [Tang et al. 2019] propose a hybrid fog-computing paradigm that integrates Flying Ad-hoc Network (FANETs) and fog-enabled vehicles in the disaster area, aiming to run the highly demanding applications while meeting strict latency requirements. These studies were not evaluated in scenarios involving Search-and-Rescue (SAR) operations, such as critical infrastructure and forest, unlike MonDrone-Fog, which evaluates the impact of runtime, energy, throughput, and RSSI in different UAV displacement conditions.

Edge paradigm is the deployment of cloud computing-like capabilities at the edge of the network. Similar to MonDrone-Fog, the studies [Kalatzis et al. 2018], [He et al. 2018], [Hou et al. 2019], [Mohamed et al. 2017] and [Tang et al. 2019] use fog computing for the processing and storage of raw data from IoT applications. The paper [Hou et al. 2019] introduces fog computing into a swarm of drones, constructing a task allocation optimization problem that jointly considers latency, reliability, and energy consumption in order to minimize the energy consumption of the swarm of drones when the latency and reliability requirements are met. A fast Proximal Jacobi Alternating Direction Method of Multipliers (ADMM)-based distributed task allocation algorithm is proposed, which decomposes the optimization problem into several subproblems, and each drone can solve the subproblem using their local status information separately. In [He et al. 2018], the authors designed an airborne fog computing system, which is formed by a network of flying objects, such as UAVs, that play an important role as fog nodes. The authors have proposed a Inertial Measurement Unit (IMU) sensor-based Global Positioning System (GPS) spoofing detection scheme, a monocular camera, and an image localization approach for UAV autonomous return to support the security and safety of UAVs. Drone-based fog computing is a new concept that was recently described in [Mohamed et al. 2017], which introduced the advantages of collaborative drones in supporting IoT-based fog computing based on drone mobility, flexibility, and ease of deployment in smart cities. Thus, the collaborative drones can be used to support SAR by loading a fixed fog unit or replacing a faulty or lost unit in the case of a disaster situation.

The works [Messous et al. 2017] and [Motlagh et al. 2017] use MEC servers, which have the computation and storage resources within the mobile network, while [Mukherjee et al. 2020] introduces the concept of "EdgeDrone," which is an abstraction that is associated with the concept of Mobile Edge Devices, IoT, and Cloud. The authors in [Mukherjee et al. 2020] proposed a message transfer mechanism for the drone nodes which are actively participating as edge-computing components. This mechanism significantly improved performance in terms of latency, publisher bandwidth, and running time. On the other hand, the game theory was applied in [Messous et al. 2017] to achieve the trade-off between execution time and energy consumption, while considering computation offloading in UAV networks. Unlike previous works, MonDrone-Fog implements cloudlets because they can play a foundational role in service availability in hostile environments that do not have a network infrastructure.

Monitored metrics are raw data related to the overall performance of an edge solution, which are stored in a dataset for analysis. The papers that focus on Disaster Operations, such as [Tang et al. 2019] and [Luo et al. 2015], have explored Battery, Mobility, and Location, whereas the paper that investigates Crowd Surveillance has looked into Route, Energy, and Mission. Our solution is the only one that monitors and evaluates Energy, Time, Throughput, and RSSI in an SAR operation. Energy and Execution Time are the most utilized metrics when evaluating the fog systems. The computing speeds of UAVs will not grow at the same pace as servers' performance. This is due to several constraints, including: form factor (professionals want UAVs that are smaller and lightweight, but with more computational capability) and power consumption (the current battery technology constrains the clock speed of processors, doubling the clock speed and approximately octupling the power consumption. As a result of the above restrictions, it is difficult to offer long battery lifetimes with high clock speeds. Therefore, execution time and energy will continue to be a fog concern in the long term, motivating further research under these topics.

MonDrone-Fog is the first offloading architecture developed and designed to monitor and store performance metrics related to drone, wireless network, and cloudlet in search-and-rescue scenarios. To the best of our knowledge, until now, there has been no work in the literature that uses and evaluates multiple machine-learning classifiers from experimental databases. Its historical database was built automatically from experimental data, i.e., our algorithm orchestrates image-offloading operations and decides the class label (VM01 or VM02) in the database according to the execution time results. In a nutshell, MonDrone-Fog adopts a history-based prediction approach where we utilize the past profiled information as a basis for performance inference in future tasks.

3. MonDrone-Fog design

The proposed system is a three-tier architecture, involving the drone, fog node, and the cloudlet interacting with each other to ensure performance metrics for image offloading and storing. The system architecture is presented in Figure 1.

- **Drone:** Capture and transmit images of different sizes. It has two components: data flow and AT commands. The data flow component allows for the capture and transmission of images and video to stream between the drone and fog node. AT

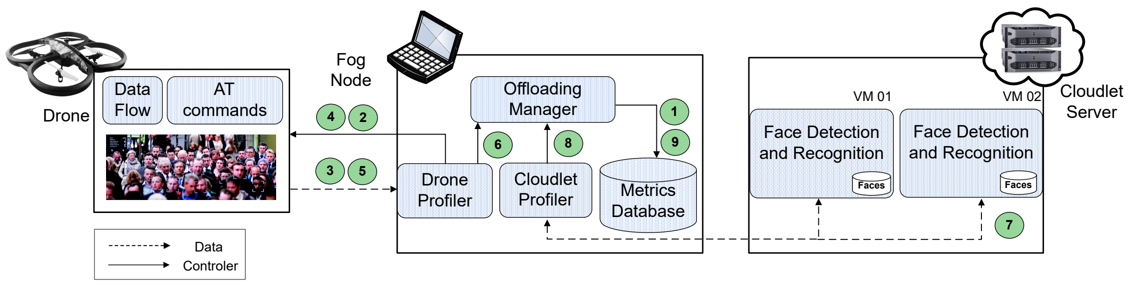

Figure 1. MonDrone-Fog Architecture

commands are intended to control and configure the drone in communication with the fog node.

- **Fog Node:** Controls the drone's actions and movements, monitors its respective activity indexes, and manages and manipulates the information that is collected and transmitted during the whole process. It has four components: offloading manager, drone profiler, cloudlet profiler, and metrics database.
  - **Offloading manager:** this component accommodates an algorithm that triggers offloading operations of the input and output images, captures all metrics between drone and cloudlet, calculates computation time, and saves metric values together with class labels.
  - **Drone profiler:** this component aims to capture network and drone information at runtime. The monitoring of network quality is critical in fog environments since a poor network can cause packet loss and delay in communication between drone and fog node. It monitors round-trip time (RTT), received signal strength indication (RSSI), the drone's battery, throughput, and image size.
  - **Cloudlet profiler:** this component is responsible for monitoring and collecting cloudlet performance data, such as vCPU usage and virtual memory usage for each VM. Moreover, this component receives the processed image with the faces recognized.
  - **Metrics database:** once a cycle is executed by the offloading manager, the previously collected metrics are recorded in the database.
- **Cloudlet Server:** Fog computing resources for processing face recognition and detection methods on images. It has two virtual machines (VMs), one with low processing and memory capacity and another with high processing and memory capacity.

We define components, data (dash line), and control (solid line) operations, represented by circled numbers, as the following: ① Fog node creates the log file. ② Drone profiler requests the face image (1.5MP, 7MP, or 14MP). ③ Drone sends the face image through the wireless channel between the drone and fog node. ④ Drone profiler request the RSSI, battery, and download/upload time. ⑤ Drone sends the image size, RSSI, battery, and download/upload time. ⑥ Drone profiler forwards the face image and other aforementioned metrics. ⑦ Cloudlet profiler sends face image to the VMs. These have a face detection and recognition application. The face detection component utilizes the *Haar Cascades* algorithm implemented in OpenCV library. [Bradski and Kaehler 2008]

It detects faces from the input image and extracts them. For each detected face, the face-recognition component compares it to the image database and returns their corresponding profile. ⑧ Thus, cloudlet profiler receives the quantity of faces detected and recognized, along with the metrics values, such as vCPU, memory and runtime. ⑨ Offloading manager receives and saves all values of the aforementioned metrics into a database.

4. Analysis of Results

In this section, we present the evaluation process of the classification algorithms. The following subsections present the configuration setup and evaluation that are related to our six classification algorithms.

4.1. Evaluation setup

The evaluation process is illustrated in Figure 2. Each step is described below.

In the first step (i), the researcher must run MonDrone-Fog to collect data related to the performance of the whole architecture. At runtime, MonDrone-Fog fills the database with raw data from the metrics. In this phase, experiments are undertaken that change the behavior of the drone, wireless networks, and cloudlet (VMs). Then the MonDrone-Fog analyzes the total execution time of each task between two configurations: VM1 remote, where the whole computing-intensive task is processed remotely in the VM1, and VM2 remote, where the whole computing-intensive task is processed remotely in the VM2. At the end of this phase, the MonDrone-Fog compares the total execution time of each task (VM1 and VM2) for each cycle, then labels those ones whose total execution time is shorter with a "VM1" value and all others with a "VM2" value.

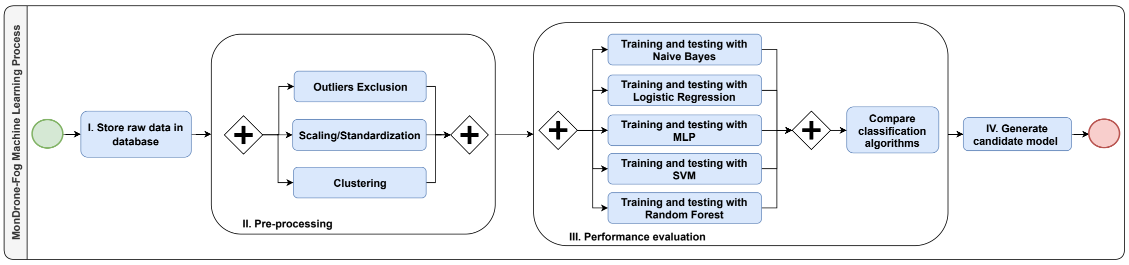
MonDrone-Fog Machine Learning Process
I. Store raw data in database
Outliers Exclusion
Scaling/Standardization
Clustering
II. Pre-processing
Training and testing with Naive Bayes
Training and testing with Logistic Regression
Training and testing with MLP
Training and testing with SVM
Training and testing with Random Forest
Compare classification algorithms
III. Performance evaluation
IV. Generate candidate model

Figure 2. Classifiers performance evaluation process

The next step (ii) covers the preprocessing techniques for transforming raw numerical data into features purpose-built for classification algorithms. The first technique is related to detecting outliers to identify extreme observations, then dropping the observations, since we can't believe those values [Albon 2018]. As a result, 10 records were discarded because outliers were detected in the *RTT* between Drone and Fog Node, and *Throughput* features. The second technique is a common approach to bringing different features onto the same scale: standardization. Using standardization, we center the feature columns at mean 0 with standard deviation 1, so that the feature columns take the form of a normal distribution, which makes it easier to learn the weights [Albon 2018]. The standardization is more practical for our problem, because some linear classifiers, such as the logistic regression and SVM, initialize the weights to 0, or small random values close to 0. The last technique is related to the grouping of observations using clustering. We use k-means clustering to group similar observations and output a new feature containing each observation's group membership [Zheng 2015].

The next step (iii) covers the classifiers evaluation process. We measure the performance of each classifier by means of its accuracy, F1, G-mean, and others metrics (e.g., specificity, sensitivity, precision) in the test data by using 10-fold cross-validation and random seed. For this evaluation, six classification algorithms (MLP, Logistic Regression (LR), SVM, Random Forest (RF), Naive Bayes (NB), and Dummy (DM)) were compared to each other. The Dummy classifier is a simple baseline to compare against our models. According to [Albon 2018], a common measure of a classifier's performance is how much better it is than random guessing. We have used these classifiers because they require fewer resources (e.g., processing, memory) than deep learning techniques, and are fairly accurate [Perera et al. 2013].

Finally, in the last step (iv), the researcher must generate the trained classifiers' models corresponding to those that obtained the highest accuracy. In addition, the files corresponding to the generated models must be saved in the MonDrone-Fog project to allow the fog node to load them at runtime. The objective is to evaluate the classifiers' performance in a real environment and without controlling the behavior of the MonDrone-Fog components.

4.2. Classifiers performance

From the average results obtained, it is possible to see in Table 1 values related to the specificity, sensitivity, precision, FPR, and FNR. From the table, we can observe that the *precision* is highest and similar for MLP and Logistic Regression (i.e., 99.76% and 99.16%, respectively) and lowest for Dummy with 57.22%. The *sensitivity* is maximum for MLP and Logistic Regression, with a rate of 99.49% and 99.27%, respectively, and moderate for Random Forest at 96.51%. In a nutshell, MLP and Logistic Regression were the algorithms that most correctly predicted positive records (offloading to the VM01), as well as outperforming SVM and Naive Bayes. The *specificity* with MLP classifier was 99.63%, slightly higher than Logistic Regression with a difference of 0.18%, and a 1.55% difference compared to SVM.

Table 1. Average of each measured metric for algorithms (%)

Algorithm	Precision	FPR	FNR	Sensitivity	Specificity
MLP	99.76 ± 0.21	0.24 ± 0.21	0.51 ± 0.24	99.49 ± 0.24	99.63 ± 0.17
LR	99.16 ± 0.14	0.84 ± 0.14	0.73 ± 0.13	99.27 ± 0.13	99.45 ± 0.08
SVM	98.28 ± 0.30	1.72 ± 0.30	2.56 ± 0.25	97.44 ± 0.25	98.08 ± 0.22
RF	97.27 ± 0.21	2.73 ± 0.21	3.49 ± 0.35	96.51 ± 0.35	97.47 ± 0.34
NB	97.89 ± 0.19	2.11 ± 0.19	11.38 ± 0.31	97.31 ± 0.25	90.71 ± 0.36
DM	57.22 ± 0.77	42.78 ± 0.77	57.71 ± 1.20	42.02 ± 2.07	57.52 ± 1.82

The FPR and FNR are referred to as the errors rate. This metric is important for critical situations in rescue operations, because the fog node can decide to offload an image to a VM erroneously, which results in a longer response time from the cloudlet or in crashing it, consequently costing the life of a lost person. According to Table 1, MLP and Logistic Regression have lower FPR (0.24% and 0.84%) compared to SVM, Random

Forest, and Naive Bayes algorithms, while FNR is the lowest for MLP, with a difference of 0.22% compared to Logistic Regression, and 2.05% compared to SVM.

Figure 3 illustrates the results related to Accuracy, F1, and G-mean. The results show that the MLP slightly outperforms both Logistic Regression and SVM classifiers in all metrics. The accuracy of the MLP algorithm is 99.64%, comparable to the Logistic Regression, which achieves 99.20%. The Naive Bayes had the worst accuracy, with 93.53% of records correctly classified. It can be seen that, in all cases, the Dummy classifier has shown to have the worst performance over our database, because is a algorithm that makes predictions using simple rules. This classifier is useful as a simple baseline to compare with our real classifiers.

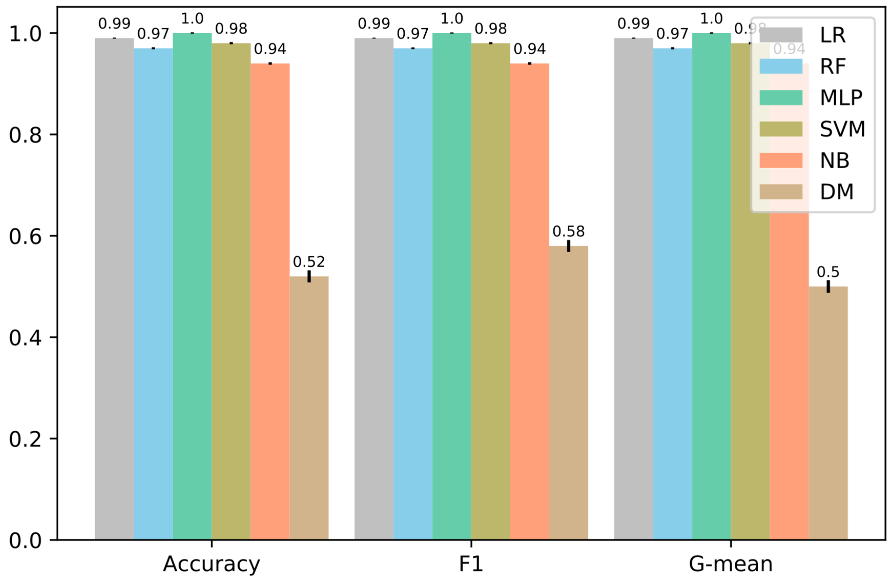

Figure 3. Comparison between the classifiers performance

With this result, we can generate a weighted majority voting model, in which the two classifiers (MLP and Logistic Regression) with better performance are decisive for the choice of the best class.

5. Conclusion and Future Direction

This paper presented MonDrone-Fog, a fog-based architecture for monitoring and storing the performance metrics related to the drone, wireless network, and cloudlet. The proposed solution relies on our historical database for the training and testing of six classification algorithms previously selected. These are MLP, Logistic Regression, SVM, Random Forest, Naive Bayes, and Dummy. Of these, two are chosen for implementation in MonDrone-Fog, MLP and Logistic Regression, since they had the best performance over our database.

Currently, MonDrone-Fog focuses on the binary classification of where to remotely process images with faces in cloudlet (VM01 or VM02). Our architecture does not consider different fog nodes, such as microfogs, that can be a subset of fog devices (e.g., two Raspberry Pis) enabled by a fog cluster to handle data from a sensor onboard

UAVs [Torres Neto et al. 2019]. Thus, further work will consist of increasing the spectrum of offloading opportunities, considering a heterogeneous architecture composed of cloudlet, microfog, access points, and gateways. Additionally, we intend to investigate Multi-Drone systems, which are drones with on-board imaging sensors that are used to locate the position of missing persons in hostile environments [Shakhatreh et al. 2019].

References

[Albon 2018] Albon, C. (2018). *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning.* " O'Reilly Media, Inc.".

[Bradski and Kaehler 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.".

[Giang et al. 2020] Giang, N. K., Lea, R., and Leung, V. C. (2020). Developing applications in large scale, dynamic fog computing: A case study. *Software: Practice and Experience*, 50(5):519–532.

[He et al. 2018] He, D., Qiao, Y., Chan, S., and Guizani, N. (2018). Flight security and safety of drones in airborne fog computing systems. *IEEE Communications Magazine*, 56(5):66–71.

[Hou et al. 2019] Hou, X., Ren, Z., Cheng, W., Chen, C., and Zhang, H. (2019). Fog based computation offloading for swarm of drones. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE.

[Junior et al. 2019] Junior, W., Oliveira, E., Santos, A., and Dias, K. (2019). A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment. *Future Generation Computer Systems*, 90:503–520.

[Kalatzis et al. 2018] Kalatzis, N., Avgeris, M., Dechouniotis, D., Papadakis-Vlachopapadopoulos, K., Roussaki, I., and Papavassiliou, S. (2018). Edge computing in iot ecosystems for uav-enabled early fire detection. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 106–114. IEEE.

[Luo et al. 2015] Luo, C., Nightingale, J., Asemota, E., and Grecos, C. (2015). A uav-cloud system for disaster sensing applications. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

[Messous et al. 2017] Messous, M.-A., Sedjelmaci, H., Houari, N., and Senouci, S.-M. (2017). Computation offloading game for an uav network in mobile edge computing. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[Mohamed et al. 2017] Mohamed, N., Al-Jaroodi, J., Jawhar, I., Noura, H., and Mahmoud, S. (2017). Uavfog: A uav-based fog computing for internet of things. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–8. IEEE.

[Motlagh et al. 2017] Motlagh, N. H., Bagaa, M., and Taleb, T. (2017). Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134.

[Mukherjee et al. 2020] Mukherjee, A., Dey, N., and De, D. (2020). Edgedrone: Qos aware mqtt middleware for mobile edge computing in opportunistic internet of drone things. *Computer Communications*.

[Perera et al. 2013] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454.

[Premsankar et al. 2018] Premsankar, G., Di Francesco, M., and Taleb, T. (2018). Edge computing for the internet of things: A case study. *IEEE Internet of Things Journal*, 5(2):1275–1284.

[Shahidinejad and Ghobaei-Arani] Shahidinejad, A. and Ghobaei-Arani, M. Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach. *Software: Practice and Experience*.

[Shakarami et al. 2020] Shakarami, A., Shahidinejad, A., and Ghobaei-Arani, M. (2020). A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective. *Software: Practice and Experience*.

[Shakhatreh et al. 2019] Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., and Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634.

[Tang et al. 2019] Tang, C., Zhu, C., Wei, X., Peng, H., and Wang, Y. (2019). Integration of uav and fog-enabled vehicle: Application in post-disaster relief. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 548–555. IEEE.

[Torres Neto et al. 2019] Torres Neto, J. R., Rocha Filho, G. P., Mano, L. Y., Villas, L. A., and Ueyama, J. (2019). Exploiting offloading in iot-based microfog: experiments with face recognition and fall detection. *Wireless Communications and Mobile Computing*, 2019.

[Waharte and Trigoni 2010] Waharte, S. and Trigoni, N. (2010). Supporting search and rescue operations with uavs. In *2010 International Conference on Emerging Security Technologies*, pages 142–147. IEEE.

[Yi et al. 2015] Yi, S., Li, C., and Li, Q. (2015). A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42.

[Yousefpour et al. 2019] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, (February).

[Zheng 2015] Zheng, A. (2015). Evaluating machine learning models: a beginner's guide to key concepts and pitfalls.