IEEE Open Journal of Vehicular Technology

Received 4 October 2023; accepted 11 November 2023. Date of publication 17 November 2023;
date of current version 5 December 2023. The review of this article was coordinated by Editor Vuk Marojevic.

Digital Object Identifier 10.1109/OJVT.2023.3333676

URANUS: Radio Frequency Tracking, Classification and Identification of Unmanned Aircraft Vehicles

DOMENICO LOFÙ [1], PIETRO DI GENNARO [2], PIETRO TEDESCHI [3] (Member, IEEE), TOMMASO DI NOIA [1] (Member, IEEE), AND EUGENIO DI SCIASCIO [1]

[1]Department of Electrical and Information Engineering (DEI), Politecnico di Bari, 70125 Bari, Italy
[2]IMT School for Advanced Studies Lucca, 55100 Lucca, Italy
[3]Autonomous Robotics Research Center, Technology Innovation Institute, Abu Dhabi 9639, UAE

CORRESPONDING AUTHOR: DOMENICO LOFÙ (e-mail: domenico.lofu@poliba.it).

This work was supported in part by the Italian MISE FSC 2014/20 Asse I project "Casa delle Tecnologie Emergenti", and in part by the Italian P.O. Puglia FESR 2014/20 project 6ESURE5 "Secure Safe Apulia".

ABSTRACT Safety and security issues for Critical Infrastructures are growing as attackers adopt drones as an attack vector flying in sensitive airspaces, such as airports, military bases, city centers, and crowded places. Despite the use of UAVs for logistics, shipping recreation activities, and commercial applications, their usage poses severe concerns to operators due to the violations and the invasions of the restricted airspaces. A cost-effective and real-time framework is needed to detect the presence of drones in such cases. In this contribution, we propose an efficient radio frequency-based detection framework called URANUS. We leverage real-time data provided by the Radio Frequency/Direction Finding system, and radars in order to detect, classify and identify drones (multi-copter and fixed-wings) invading no-drone zones. We adopt a Multilayer Perceptron neural network to identify and classify UAVs in real-time, with 90% accuracy. For the tracking task, we use a Random Forest model to predict the position of a drone with an MSE ≈ 0.29, MAE ≈ 0.04, and $R^2 \approx 0.93$. Furthermore, coordinate regression is performed using Universal Transverse Mercator coordinates to ensure high accuracy. Our analysis shows that URANUS is an ideal framework for identifying, classifying, and tracking UAVs that most Critical Infrastructure operators can adopt.

INDEX TERMS UAV, security, safety, drone, cyber physical systems, machine learning.

I. INTRODUCTION AND MOTIVATION

In the last years, Unmanned Aerial Vehicles (UAVs) commonly known as *drones* have become a crucial technology for several types of applications such as environmental monitoring [1], [2], [3], smart grids control [4], crime prevention [5], [6], smart cities management [7], and the military operations [8]. According to authoritative marketing research industries, the UAV market is estimated to be 500,000 units in 2025 and is projected to reach 6.9 million units by 2030. The global drone logistics & transportation market accounted for US$24.58 million in 2018 and is expected to grow at a Compound Annual Growth Rate (CAGR) of 60.6% over the forecast period 2019 − 2027, to account for US$1,626.98 million in 2027 [9]. Factors including increasing developments in the e-commerce sector [10] and rising acceptance owing to various benefits offered are significantly driving the global drone logistics [11] & transportation market [12].

Most commercial drones are autonomous or remotely controlled vehicles, that leverage the standard Wi-Fi frequency bands, i.e. 2.4 GHz [13] and 5.0 GHz [14]. They can be programmed to execute tasks that span from object tracking [15], and delivery, to committing illegal activities such as privacy violations, destroying critical infrastructures, and harming public safety during crowded events [16]. Given the above threats, several countermeasures [17] based on audio, video, thermal, and Radio Frequency signals have been exploited in the last few years for drone identification and tracking. However, the performance of these systems is affected when the

surrounding environment is impaired (e.g. weather conditions, noise, low light visibility). Indeed, most critical infrastructures adopt Radio Frequency / Direction Finding (RFDF) and kinematics radar sensors that track all types of drones by analyzing the reflected signals and comparing them to a database for drone characterization. Due to the high number of unauthorized UAVs operating in the skies, it is crucial to deploy a system framework to track, classify and identify timely, malicious UAVs by leveraging the data provided by radar sensors.

In this paper, we design URANUS, a Machine Learning (ML) framework that analyzes a dataset with data extracted from two RFDF sensors namely *Diana* and *Venus*, and two radar sensors namely *Arcus* and *Alvira* to *(i) identify*, *(ii) classify*, and *(iii) track* drone(s) on a North Atlantic Treaty Organization (NATO) military base (placed in the Counter Unmanned Aerial System (C-UAS) test centre in the Netherlands).

Our framework is trained over a real dataset derived from a data source of UAVs flights provided by NATO [18].

Our prototype adopts popular libraries and tools such as *PyTorch*, *scikit-learn* and *pandas*, available online as open source code [19].

Our main contributions include the following:

1) Identify, classification, and tracking of one or more flying drones;
2) Classification of fixed-wing and multi-copter drones;
3) Analysis of both RFDF, and kinematics sensors to detect drones in Critical Infrastructure (CI);
4) Real-time framework execution.

The remainder of this paper is organized as follows: Section II introduces the technical background. Section III describes the reference scenario and adversarial model, while Section IV details the dataset analysis. Section V shows the proposed architecture, while experiments and results are described in Section VI. Section VII reviews the related works, and Section VIII concludes the paper.

II. PRELIMINARIES

In this section, we introduce some preliminary knowledge adopted throughout the rest of the manuscript. Specifically, Section II-A describes the coordinate systems adopted in this work, while Section II-B and II-C describes the radar parameters and the ML models used in the URANUS, respectively.

A. GCS AND UTM COORDINATE SYSTEMS

The Geographic Coordinate System (GCS) [20] and the Universal Transverse Mercator (UTM) coordinate system [21] are two standard techniques to represent locations on the Earth's surface. Coordinates systems often use a tuple of real numbers (x_1, x_2) to identify an object's unique location, where $x_1 \in \mathbb{R}$, and $x_2 \in \mathbb{R}$.

Geographic Coordinate System (GCS): Latitude and longitude are used to specify the location of a point on the Earth's surface. The *latitude* (1) measures the distance north or south of the Equator, while the *Longitude* (2) measures the distance east or west of the Prime Meridian:

$$\text{Latitude} = \arcsin\left(\frac{Z}{\sqrt{X^2+Y^2+Z^2}}\right), \tag{1}$$

$$\text{Longitude} = \arctan\left(\frac{Y}{X}\right), \tag{2}$$

where X, Y, and Z, represent the Cartesian coordinates of the object to track. Longitude and latitude are defined in the Degrees Minutes Seconds (DMS) form or using Decimal Degrees (DDs) values. An example with coordinates expressed in DMS format is:

$$(-73^\circ\ 58'\ 2'',\ 40^\circ\ 44'\ 58'')$$

while the same place expressed in DD format is:

$$(-73.967385,\ 40.749598).$$

Universal Transverse Mercator (UTM): The Earth is segmented into 60 longitudinal zones, each spanning 6 degrees of longitude [22]. Within each zone, a transverse Mercator projection is used to represent locations:

$$\text{Easting} = \text{Zone Number} \times 10^5 + \text{Easting Value}, \tag{3}$$

$$\text{Northing} = \text{Northern Hemisphere Constant} + \text{Northing Value}, \tag{4}$$

where *Zone Number*, *Easting Value*, and *Northing Value* represent the longitudinal zone, the distance east of the central meridian in *meters*, and the distance north of the Equator in *meters*, respectively. Further, the Northern Hemisphere has a constant of 0, and the Southern Hemisphere has a value of 10,000,000 [22]. The Zone Number and the Hemisphere are adopted to uniquely identify locations within the UTM grid. An example of coordinates in UTM is:

$$18\ N\ 587173\ 4511473$$

where 18 in the UTM zone, N is the Northern Hemisphere, 587173 is the Easting value and 4511473 is the Northing value.

To the best of our knowledge, we are the first to highlight better performances of an ML model that adopts UTM coordinates instead of the canonical GCS. In our regression tests with Random Forest (RF) models, the mean difference between real and predicted values is around 18 meters, a relevant value. Furthermore, the minimum margin between real values and positions provided by Radar Sensor Systems was set to 50 meters by project settings.

B. RANGE AND BEARING RADAR PARAMETERS

In radar systems, *Range* and *Bearing* are two fundamental concepts used to determine the location of a target.

Range: The range refers to the distance between the radar system and the target. It represents the radial distance from the radar transmitter/receiver to the reflecting point on the target. The range (5) is typically measured in units such as *meters* or *nautical miles*, as follows:

$$R = \frac{c \cdot T_R}{2}, \tag{5}$$

where $c = 3 \cdot 10^8$ m/s is the speed of the light, and T_R is the transmitted pulse [23].

Bearing: In radar systems, bearing refers to determining the direction from which a detected signal or echo comes. The True Bearing, referenced to true north, for a radar target is the angle formed between true north and a line directly aimed at the target [24]. Radar systems determine bearing by analyzing the angle at which the received signal is stronger. This latter is usually achieved by using an antenna array or rotating the antenna to scan the surrounding environment.

C. MACHINE LEARNING ALGORITHMS

This section introduces the ML algorithms adopted in our framework.

Multilayer Perceptron (MLP): Multilayer Perceptron (MLP), also known as *Deep Feedforward Networks* or *Feed-forward Neural Networks* [25], it is a type of Artificial Neural Network (ANN) [26] [25] widely used in Machine Learning and pattern recognition tasks. Specifically, we adopted this technique for the classification task. A MLP mimics how neurons interact and work in the human brain. It is characterized by a layered architecture consisting of multiple interconnected nodes organized into three primary layers: the input layer, one or more hidden layer(s), and the output layer. The hidden and output layers have neurons connected to the neurons of their preceding layers and network connections; further, the topology can be fully connected or partially connected.

In MLP neural networks, each unit performs a biased, weighted sum of inputs and passes this activation level through a transfer function to generate output. The Rectified Linear Unit (ReLU) activation function is the preferred default activation function for most feed-forward neural networks. When applied to the output of a linear transformation, it results in a nonlinear transformation.

MLPs have demonstrated outstanding capabilities in modelling complex data patterns for several deep learning architectures. They can be customized with several hidden layers and neurons to accommodate the complexity of different tasks. Each unit resembles a neuron, i.e., it receives input from many other units and computes its activation value.

By tuning the weights of the connections between the nodes in the network, the model learns to predict the target output. Further, an optimization algorithm is adopted to adjust the weights (*i.e. stochastic gradient descent*). In particular, it minimises the difference between the predicted and the actual target output.

The MLP model is represented as a function $f(x)$ that maps the input data x to the output y. The function $f(x)$ is expressed as a composition of other functions, as shown in the following equation:

$$f_a(x) = f_L(f_{L-1}(\ldots(f_2(f_1(x, \theta_1), \theta_2)\ldots), \theta_{L-1}), \theta_L) \quad (6)$$

where f_i is the nonlinear transfer function of the ith hidden layer, θ_i represents the weights connecting the nodes in layer i and layer $i + 1$, and L is the number of layers in the MLP.

Random Forest (RF): It is a typical ML algorithm [27], [28] used for the regression task. A RF model is an ensemble of decision trees that can handle high-dimensionality datasets. Each decision tree is trained on a subset of the data and a subset of the features. The output of the individual decision tree is the average or mode for regression or classification, respectively.

Let Sum of Squared Errors (SSE) be the sum of the squared differences between the predicted and actual values of the target variable. In the regression form, the splitting criteria for node creation follows the largest reduction in the value of the SSE for the predicted output. The splitting process continues until a stopping criteria (such as a maximum tree depth or a minimum number of data points in a leaf node) is defined. The output of the RF for regression is the average of the predicted values of the individual decision tree. The output value of a RF for regression is:

$$\hat{y} = \frac{1}{M} \sum_{i=1}^{M} f_i(x) \quad (7)$$

where $\hat{y}$ is the predicted value of the target variable, M is the number of decision trees in the RF model, and $f_i(x)$ are the predictions of the $i - th$ decision tree for the input features. We adopt the Random Forest in URANUS for the coordinates regression task.

D. PREPROCESSING PRIMITIVES

This section introduces the preprocessing primitives used to transform the original data for the ML models [29].

Dataset Standardization: It is an essential preprocessing step in ML, suitable for algorithms sensitive to the scale of input features [30]. In order to identify patterns and relationships with high accuracy, input features should have the same scale. Input data should be adapted to have 0 mean and a standard deviation equal to 1. To this aim we apply 8 to every single feature of the initial dataset:

$$x_{sf} = \frac{x - \mu}{\sigma} \quad (8)$$

where, x is the considered feature value, μ and σ represents the mean and the standard deviation of the feature, and x_{sf} is the final scaled feature value.

One-hot Encoding: It is a widely used technique [31], [32] for converting categorical data into a binary matrix format suitable for ML models. Different algorithms require numerical input, and the one-hot encoding makes the representation of discrete categories as unique binary vectors.

Specifically, the 9 shows how this technique encodes categories:

$$b_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The vector and category index are represented by i and j, respectively, while the output is a binary vector namely $\bar{b}$. For

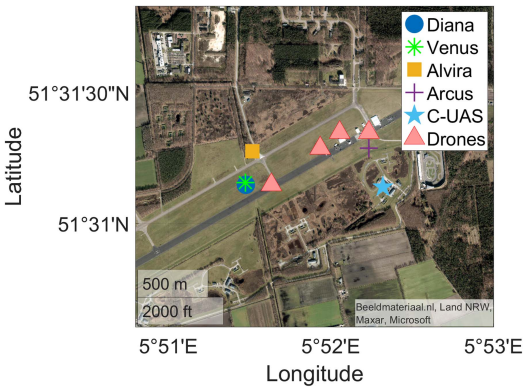

FIGURE 1. Scenario assumed in this work.

TABLE 1. Classification and Characteristics of the Drones

Drone Name	Airframe	Weight $[kg]$	Max Velocity $[m/s]$	RCS $[m^2]$	FCSF $[m^2]$
DJI Mavic Pro	Multi-copter	1	20	0.01	0.02
DJI Mavic 2	Multi-copter	1	20	0.01	0.02
DJI Phantom 4 Pro	Multi-copter	1	20	0.01	0.02
Parrot Disco	Fixed wing	1	20	0.005	0.1

TABLE 2. Description of the Sensor Types With Their Respective Positions

Sensor Name	Sensor Type	Sensor Latitude	Sensor Longitude
Diana	RFDF	51.51913°	5.85795°
Venus	RFDF	51.51927°	5.85791°
Alvira	Radar	51.52126°	5.85860°
Arcus	Radar	51.52147°	5.87056°

any given observation, the binary vector $\bar{b}$ column corresponding to its category is marked with a 1, while all other columns are set to 0. This representation ensures the avoidance of incorrect relational order between categories.

Label encoding: This widely used technique converts categorical data into numerical form [25] and makes it machine-readable, as algorithms work exclusively with numerical data. In this approach, each unique category or label within a feature is assigned a distinct integer value. For instance, for a categorical variable with values like *low*, *medium*, and *high*, label encoding might assign these categories values of 0, 1, and 2, respectively. The primary advantage of label encoding is its simplicity and ability to retain a compact dataset representation.

III. REFERENCE SCENARIO AND ADVERSARIAL MODEL

This section introduces the scenario and the adversarial model considered in our work. Specifically, Section III-A depicts the system model and describes the assumptions, while Section III-B describes the adversary model.

A. SYSTEM MODEL AND ASSUMPTIONS

The scenario assumed in this work is depicted in Fig. 1. We consider the problem of tracking, classifying, and identifying one or multiple drones (multi-copter or fixed wing) in a No-Drone Zone [33]. It is worth noticing that we refer to the scenario described by NATO in [18].

The main components of the considered scenario are the following:

- No-Drone Zone (NDZ) - The Federal Aviation Administration (FAA) introduced the term NDZ to describe an area that does not allow to operate by using a drone or unmanned aircraft system. Examples of NDZ areas are airports, restricted airspaces, government agencies, or also temporary flight restrictions areas such as sports events, presidential movements, and security-sensitive areas.
- Drone - The drone(s) assumed in this scenario are classified with the features reported in Table 1. Each drone is characterized by the commercial name, a code field, and a unique identifier assigned during the drone configuration phase. The airframe represents the type of drone, i.e. a multi-copter or a fixed-wing drone; the weight specifies the maximum weight of the drone, and velocity indicates the maximum speed of the drone. Radar cross-section (RCS) or radar signature measures how much energy the drone reflects towards a radar, i.e. the area seen by radar. Frame Cross Section Frontal (FCSF) defines a frame of the frontal measurement of the cross-section.
- RFDF-Radar Sensor Network - The No-Drone Zone is monitored by two RFDF sensors, namely Diana and Venus, and two radar sensors i.e. Arcus and Alvira (all of them are fictitious names). From one side, Diana and Venus sensors acquire data such as time-of-arrival (timestamp), Receiving Signal Strength (RSS), and beamforming to localize the target. Diana adopts a linear array antenna to estimate the bearing of an intercept; it only reports detections in a 180° sector even if the target is located in the opposite sector. Venus uses a circular array antenna with no bearing ambiguity and provides no range information. Conversely, Arcus and Alvira sensors are 2D radar and 3D radar, respectively. These sensors provide crucial information such as latitude, longitude, altitude, and timestamp, as well as the bearing and range of the drone during the flight. Table 2 summarizes the name, the type, and the sensor location (latitude and longitude coordinates).
- Counter Unmanned Aerial System - It is a central server unit adopted to collect and process (via URANUS) the data generated by the sensor network. The system is used to detect, identify and track the presence of any unauthorized or malicious drone in the No-Drone Zone.

As mentioned above, our scenario assumes the presence of a C-UAS operator in a NDZ, e.g., the one controlling a generic critical infrastructure. Such operators are interested in monitoring the critical infrastructure, looking for malicious and unauthorized UAVs approaching a sensitive area. To this aim, the operators adopt an RFDF Radar sensor network to

TABLE 3. Drone(s) Model Involved in Each Scenario

Scenario	Drone Name
Scenario 1.1	DJI Mavic Pro
Scenario 1.2	DJI Phantom 4 Pro
Scenario 1.3	DJI Mavic Pro
Scenario 1.4	DJI Mavic 2
Scenario 2.1	DJI Phantom 4 Pro and DJI Mavic 2
Scenario 2.2	DJI Phantom 4 Pro and DJI Mavic Pro
Scenario 3	Parrot Disco

capture crucial information in the monitored area to identify, classify and track the owner of the UAV. In this paper, we consider three macro-scenarios as follows:

- Scenario 1.1, 1.2, 1.3, and 1.4. In these scenarios, we assume a single UAV (multi-copter) is flying in the NDZ.
- Scenario 2.1 and 2.2. In these scenarios, we assume two UAVs (multi-copter) are flying in the NDZ.
- Scenario 3. In this scenario, we assume one UAVs (fixed-wing) is flying in the NDZ.

Each flight scenario involves one or more drones with individual flight patterns. According to Table 3, the various scenarios include different drones.

We highlight that the aforementioned scenarios are only a reference for the considered dataset. Our framework can be applied to other potential environments, such as surveillance towers in critical infrastructures, military bases, ports, and airports.

B. ADVERSARY MODEL

In all the scenarios, we assume that an adversary $\mathcal{E}$ has the capabilities to radio-control a single drone or a swarm of drones, and it is interested in reaching a target inside a NDZ. The aims of the adversary can be manifold, e.g., violating the privacy of the area by recording video and/or taking photos of a sensitive area, using it as a bomb in critical infrastructures such as airports, oil&gas industries, nuclear power plants, water treatment facilities, ports, telecommunication networks, or to threat people safety by carrying explosives or radioactive materials or colliding with airplanes during the take-off and landing procedures. Moreover, the adversary $\mathcal{E}$ can control a drone in several ways: (i) through a wireless remote controller, (ii) remotely via the Internet (i.e. the drone supports embedded Subscriber Identity Module (SIM), standard SIM, and cellular Long Term Evolution (LTE) or 5 G technology), (iii) by pre-programming it through way-points to enable the autonomous flight. Conversely, we assume that the drones broadcast data (for several purposes) via the onboard radio transmitter for the whole flight [34].

IV. DATASET ANALYSIS

In this section, we describe the data source used to develop the URANUS framework. Further, we outline the challenges and the proposed techniques to make the dataset suitable for the training and testing phases for our ML models.

Dataset Preprocessing: Preparing raw data for ML analysis and modelling is a critical step. This step is crucial to guarantee that the data are (i) consistent, (ii) flawless, and (iii) complete. These characteristics allow ML models to learn efficiently from the data and make highly accurate predictions. We considered a data source provided by NATO containing real UAV flight measurements recorded in 2020.

The data source is available online as Comma-separated Values (CSV) format [18] which contains 65 files (366 *MBs*) organized in two main sub-folders, namely *training* and *test* folder.

Due to the lack of label information, the test data folder is not considered in our work. We leveraged only the training folder data that contains 37 files (194 *MBs*) organized in seven scenarios. In detail, for each scenario, we have data related to (i) radar and RFDF systems sensor and (ii) UAV flight parameters. Specifically, Table 4 and Table 5 report data examples from *log* and *sensor data* files. It is worth noticing that for every single drone, the log file records UAVs parameters such as the *timestamp*, *latitude*, *longitude*, *speed*, *altitude*, and *drone type*.

Furthermore, the sensor data features such as the RCS, RF, and the UAV parameters are stored along with the related timestamp. Accordingly, the timestamp is adopted as the index of each data sample to merge and correlate the sensor readings and the drone data logs.

Dataset Generation: This procedure includes a summary of the operations performed in the dataset generation. The steps are detailed in Algorithm 1 as following:

1) For each scenario, the algorithm starts by loading the log file of the drone and filtering the data considering the sample window of 1 *sec*, used as the index. Suppose multiple samples are associated with the same timestamp index (i.e. when the difference between each of them is less than 1 ms). In that case, the algorithm considers the first one appearing in the CSV file of the log file being analyzed.
2) After selecting the row from the log, the algorithm scans the CSV file of each radar and RFDF system of the same scenario to bind the rows found based on the value of the timestamp index. If multiple samples are selected, the algorithm filters consider the closest one to the log's timestamp; otherwise, the algorithm inserts a blank row.
3) Data enhancement operations are performed on every sample, i.e. (i) the conversion of the coordinates from the GCS to UTM coordinates system, (ii) adding columns with extra data valuable to help training algorithms or (iii) the number of expected drones in the scenario or its classification. Table 6 shows an example of coordinates conversion from the GCS system to the UTM system.

The final dataset consists of 5,685 samples with 57 columns, used for the dataset analysis described in Section VI. Table 7 shows the full list of the columns of the merged dataset, while Table 10 shows a small dataset sample with a subset of its real columns.

TABLE 4. Data Sample From Drone Log of Scenario 1.1

Latitude	Longitude	Altitude $[m]$	UltrasonicHeight $[m]$	Speed $[m/s]$	Distance $[m]$ from radar	Datetime [UTC]	Timestamp
51.519506	5.857978	0.9	0	2.3	0.04	2020-09-29 12:10:56.647	1601381456647
51.519506	5.857978	1.2	1.2	2.7	0.05	2020-09-29 12:10:56.727	1601381456727
51.519506	5.857978	1.5	1.5	3.1	0.06	2020-09-29 12:10:56.824	1601381456823
51.519506	5.857978	1.8	1.9	3.5	0.06	2020-09-29 12:10:56.928	1601381456927
51.519506	5.857978	2.2	2.2	3.7	0.07	2020-09-29 12:10:57.027	1601381457027
51.519506	5.857978	2.6	2.6	3.8	0.08	2020-09-29 12:10:57.126	1601381457126
51.519506	5.857979	2.9	3	3.7	0.09	2020-09-29 12:10:57.225	1601381457224

TABLE 5. Data Sample From Radar Sensor ARCUS, in Scenario 1.1

Timestamp	Latitude	Longitude	Altitude $[m]$	Speed $[m/s]$	Classification	Reflection $[-]$	Score
2020-09-29T12:19:47.880Z	51.52132905	5.86072255	33.41	16.65	VEHICLE	3.26	0.78
2020-09-29T12:19:47.789Z	51.51840263	5.85496417	41.51	4.73	OTHER	−13.26	0.6
2020-09-29T12:19:47.867Z	51.52171110	5.88431792	74.63	18.34	UNKNOWN	−36.01	0.35
2020-09-29T12:19:46.973Z	51.52369287	5.85944975	34.64	8.01	UNKNOWN	−22.60	0.27
2020-09-29T12:19:46.115Z	51.52206908	5.86990915	84.86	4.80	UNKNOWN	−34.85	0.04
2020-09-29T12:19:47.962Z	51.52163624	5.86985159	51.69	4.36	UNKNOWN	−33.49	0.04
2020-09-29T12:19:48.187Z	51.52763873	5.86642335	44.86	11.91	UNKNOWN	−26.50	0.06
2020-09-29T12:19:48.331Z	51.53229705	5.86715643	129.94	12.3	UNKNOWN	−21.93	0.47

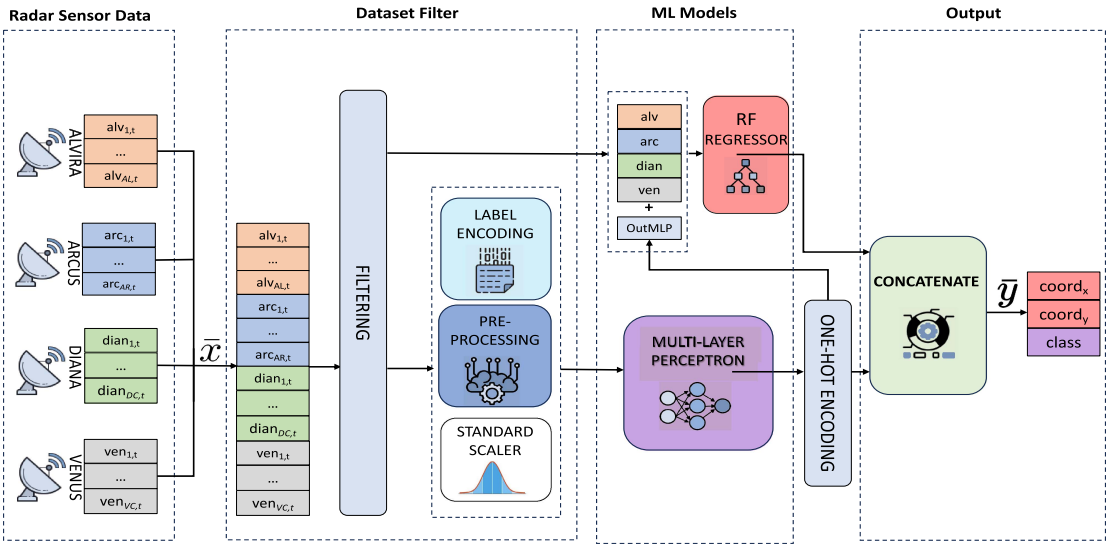

FIGURE 2. Proposed model architecture.

TABLE 6. Example of Coordinates Conversion From GCS to UTM

Format	x	y
GCS	5.857978	51.519506
UTM	698276.88	5711471.08

V. PROPOSED ARCHITECTURE

In this work, we propose a real-time ML framework called URANUS to identify, classify and track UAVs. As shown in Fig. 2 the input is a vector $\bar{x}$ with the processed information coming from radar and RFDF systems. At the same time, the output is the identification, classification, and position of an UAV, represented as $\mathbf{y} = [class, coord_x, coord_y]$. In particular, the *class* can assume the following values {0, 1, 2, 3, 4, 5, 6}, where 0 defines the case in which no drone is inside the NDZ, and the remaining values outline the presence of drones. More information are detailed in Section VI-VI-A.

In the following sections, we describe each part of the model schema shown in Fig. 2. Specifically, in Section V-A, we present dataset preparation steps, while in Section V-B, we describe the several components adopted in the framework.

A. DATASET PREPARATION

Let us consider as input vector $\mathbf{x} = \{alv_t \cup arc_t \cup dian_t \cup ven_t\} \in \mathbb{R}^{1,n}$, where $\mathbf{alv_t}, \mathbf{arc_t}, \mathbf{dian_t}, \mathbf{ven_t}$ are the vector

Algorithm 1: Algorithm for Dataset Generation.

Functions:

- $load\_log\_files(i)$: It loads the log file(s) of i-th scenario.
- $load\_data(i)$: loads the CSV files related to Alvira, Arcus, Venus, and Diana of i-th scenario.
- $load\_sample\_data\_close\_to\_timestp(k)$: It scans the scenario CSV files of Alvira, Arcus, Venus, and Diana and considers, for each of them, the sample with the closest timestamp of the considered one.
- $preprocess\_data(k)$: For each element of the k-th sample, this function applies the Standard Scaler and the Label Encoder.
- $enhancement\_operations(k)$: It enhances the k-th sample with extra information (e.g. conversion of coordinates or information related to the scenario).

Inputs:

- l_i: log file(s) of the i-th scenario.
- t_i: sensor Data, coming from Alvira, Arcus, Venus, and Diana, of the i-th scenario.

Output:

The dataset T is used to train ML models.

```
procedure DatasetGeneration
    i ← 0
    for i ← 0 to len(scenarios) do
        l_i ← load_log_files(i)
        t_i ← load_data(i)
        k ← 0
        for k ← 0 to len(l_i) do
            if (timestp_k − timestp_{k+1}) ≥ 1 s then
                load_sample_data_close_to_timestp(k)
                preprocess_data(k)
                enhancement_operations(k)
            end if
            k ← k + 1
        end for
        i ← i + 1
    end for
end procedure
```

data of Alvira, Arcus, Diana, and Venus at time t, respectively. Specifically, n is the sum of the columns of the vector data provided by the aforementioned sensors. The output vector is $\mathbf{y} = [class, coord_x, coord_y] \in R^{1,3}$, where $class \in \mathbb{N}$, $coord_x \in \mathbb{R}$ and $coord_y \in \mathbb{R}$. The parameter $class$ represents the identification and classification of an object, while $coord_x$ and $coord_y$ depict the position of an UAV.

The features from the sets $x_{1,i}$, $x_{2,i}$, $x_{3,i}$, $x_{4,i}$ where i is the scenario index—sourced from Arcus, Alvira, Diana, and Venus, respectively—are combined with the two log files l namely $l_{1,i}$ (log file of the first drone) and $l_{2,i}$ (log file of the second drone). The goal is to build a consolidated dataset, namely T, for training ML models. Table 8 summarises the notations throughout this article.

B. MODEL SETTINGS

The developed framework is designed to perform multiple operations to identify, track, and classify potential drones. The main tasks of URANUS performed in *real-time* are:

- The framework starts by collecting all the information from the available sensors (both the Radar systems and the RFDF systems).
- Then URANUS identifies if there is a drone in the NDZ zone, using a binary classification, i.e., *DRONE - NO DRONE*.
- When a drone is detected, the framework shows its position on a map and classification information.

Moreover, we verify the the design of the proposed ML model to achieve a high performance. After evaluating different network configurations, the final setting is the following:

- *Radar Sensor Data:* This module communicates with information sources - radars and RFDF systems, specifically - to collect data and make it available for subsequent modules.
- *Dataset Filter:* It is composed of a Label Encoder and a Standard Scaler [35], as described in Section II-D. This module converts the input sensor data to values compatible with the network, for instance, adapting numerical values to be more suitable for ML models or converting raw categorical data into numbers. This module is also responsible for selecting specific parts of the input features provided as input to ML models according to their needs.
- *MLP Network*: This network performs the identification and classification tasks, returning an output indicating whether a single drone or multiple drones are present within the NDZ. As depicted in Fig. 3, the network is defined by 12 input neurons, 2^{18} (512×512) hidden neurons, and 7 output neurons. Further we adopt the One-Hot encoding [32] in order to represent the 7 categorical classes (`CASE_UNKNOWN`, `CASE_FIXED_WING`, `CASE_MAVIC_PRO`, `CASE_PHANTOM4_PRO`, `CASE_MAVIC2`, `CASE_PHANTOM4_PRO_MAVIC2`, `CASE_PHANTOM4_PRO_MAVICPRO`—further details are described in Section VI). For example, the scenario that involves the presence of two drones in the NDZ (`CASE_PHANTOM4_PRO_MAVICPRO`) will be encoded with the array [0,0,0,0,0,1].
- *RF Regressor*: This model has 250 estimators used to predict the position of the identified drones, and it uses a mix of information from the dataset T and the MLP network.
- *Output:* This module interacts with both the MLP Network and the RF Regressor, consolidating their outcomes in a single output array. The results are then displayed on the interface, such as on a map.

According to the proposed defined model (Fig. 2), the MLP layer only accepts standardized data (for continuous values) or categorical data expressed using a Label Encoder (for categorical values), whereas the RF Classification and Regressor layer accepts raw, unprocessed data.

TABLE 7. Full Representation of the Columns in the Merged Dataset

Field Name	Data Source	Unit Measurement	Description
timestamp	drone,radar and RFDF sensors logs	seconds	The reference timestamp of the sample
scenario_name	drone's log	–	Name of the scenario the sample belongs to
latitude		decimal degrees	Information related to the first drone, always identified
longitude		decimal degrees	
altitude		m	
speed		m/s	
latitude_2		decimal degrees	Information, related to the second drone, if identified
longitude_2		decimal degrees	
altitude_2		m	
speed_2		m/s	
AlviraPosition_Latitude	Alvira	decimal degrees	Information of the identified object by the radar
AlviraPosition_Longitude		decimal degrees	
AlviraPosition_Altitude		m	
AlviraVelocity_Azimuth		degrees	
AlviraVelocity_Elevation		m	
AlviraVelocity_Speed		m/s	
Alvira_Classification		–	Classification value of the identified object
Alvira_Reflection		–	Reflection value
Alvira_Score		percentage	Score of the produced classification
Alvira_Alarm		boolean	
ArcusPosition_Latitude	Arcus	decimal degrees	Information of the identified object by the radar
ArcusPosition_Longitude		decimal degrees	
ArcusPosition_Altitude		m	
ArcusVelocity_Azimuth		degrees	
ArcusVelocity_Elevation		m	
ArcusVelocity_Speed		m/s	
Arcus_Classification		–	Classification value of the identified object
Arcus_Reflection		–	Reflection value
Arcus_Score		percentage	Score of the produced classification
Arcus_Alarm		boolean value	Alarm value
DianaSignal_snr_dB	Diana	dB	SNR of the detected signal
DianaSignal_bearing_deg		degrees	Antenna's bearing
DianaSignal_range_m		m	Antenna's range
DianaClassification_type		–	Void or controller
Venus_isThreat	Venus	–	
VenusLinkType_Uplink		–	Void or FHSS
Venus_VenusName		–	Name of the identified object
Venus_RadioId		–	Id of the identified radio
Venus_LifeStatus		–	Identifies the status of the antenna (down/active)
Venus_Frequency		Hz	
Venus_FrequencyBand		–	
Venus_Azimuth		degrees	Antenna's azimuth
Venus_Deviation			
reference_classification	Classification algorithm	–	Label added during the preprocessing phase to train the MLP model

– means that the unit measurement is not specified.

TABLE 8. Notation and Brief Description

Notations	Description
x	The input vector to the framework at time t
arc_t	Vector data coming from Arcus at time t
alv_t	Vector data coming from Alvira at time t
$dian_t$	Vector data coming from Diana at time t
ven_t	Vector data coming from Venus at time t
y	The output vector of the framework after processing the input vector x
$class$	Identification and classification value of an object
$coord_x$	Regressed position of a UAV, x coordinate
$coord_y$	Regressed position of a UAV, y coordinate
AR	Number of columns of the CSV files of Arcus
AL	Number of columns of the CSV files of Alvira
DC	Number of columns of the CSV files of Diana
VC	Number of columns of the CSV files of Venus
$x_{1,i}$	Features of Arcus, from $i-th$ scenario
$x_{2,i}$	Features of Alvira, from $i-th$ scenario
$x_{3,i}$	Features of Diana, from $i-th$ scenario
$x_{4,i}$	Features of Venus, from $i-th$ scenario
$l_{1,i}$	Log file of the drone, from $i-th$ scenario
$l_{2,i}$	Second log file of the drone, from $i-th$ scenario, if available
T	Merged dataset used to train ML models

VI. EXPERIMENT AND RESULTS

In this section, we summarize the rationale of the dataset analysis, the metrics and the results, and the implementation details of the URANUS framework. In Section VI-A, we present the preliminary analysis. In this step, we estimate the useful and exploitable information of the dataset T and some details regarding the labelling procedure required to train the considered ML models. Next, in Section VI-B, we evaluate the performance of the MLP and RF trained models. Finally, we provide the implementation details in Section VI-C.

A. RATIONALE OF THE ANALYSIS

As depicted in Fig. 1, RFDF sensors are positioned close to each other in the center of the test field, while radars are positioned at the edges of the NDZ. Radar sensors provide data on the classification, position, bearing (*degrees*), range (*meters*), and reflection of the flying entities. These data are based on the kinematic and reflectivity characteristics of the radar. RFDF sensors, on the other side, contribute to identify UAVs (or the drone controller) based on the RF signature. Before the design phase of ML algorithms used in URANUS, a preliminary analysis phase is performed in order to (i) estimate the number of useful information used, (ii) set project parameters, and (iii) split the dataset into subsets (train, validation, and test).

In order to train the MLP network using a supervised procedure, the *reference_classification* column is added to the dataset during the preprocessing phase, as shown in Table 7. The adopted procedure assigns one of the following values for every row of the dataset T:

- `CASE_UNKNOWN`: No drone is identified.
- `CASE_FIXED_WING`: The Parrot fixed-wing drone is identified.
- `CASE_MAVIC_PRO`: A multi-copter DJI Mavic Pro is identified.
- `CASE_PHANTOM4_PRO`: A DJI Phantom 4 is identified.
- `CASE_MAVIC2`: A DJI MAVIC 2 drone is identified.
- `CASE_PHANTOM4_PRO_MAVIC2`: Both a Phantom 4 Pro and Mavic 2 are identified.
- `CASE_PHANTOM4_PRO_MAVICPRO`: Both a Phantom 4 Pro and a Mavic Pro are identified.

The classification highlighted corresponds to the position detected by the radar sensor systems, specifically Alvira and Arcus. If the position of the identified object by the radars is within a deviation of 50 *meters* from the drone's current position - recorded in the log file(s) - then the procedure assigns to the highlighted row a category ranging from `CASE_FIXED_WING` to `CASE_PHANTOM4_PRO_MAVICPRO`, depending on the nature of the identified object(s). If not, it assigns `CASE_UNKNOWN`. Specifically, fixed-wing and quadcopters (also known as multi-rotor drones) show specific differences. Indeed, the adopted classification procedures take into account their different properties, such as (i) speed, (ii) flight time, and (iii) manoeuvrability. For instance, fixed-wing drones fly faster than multi-rotor drones, making them ideal for exploring large sites quickly. Based on these assumptions, it is possible to estimate the number of useful samples to train ML algorithms, as shown in Table 9. In particular, the table shows a summary of the information available in the generated dataset T, starting from the (i) drone logs, (ii) radar sensor data, and (iii) RFDF sensors data.

As the dataset is divided into scenarios and not all samples contain helpful information for training, the amount of useful information is considerably limited. Indeed, on average, only 31% of the dataset contains a real drone position that is useful for our training.

B. METRICS AND RESULTS

This section describes the training and design procedures to build the modules that process the raw input data. We introduce the metrics and the results for the MLP classifier in Section VI-B1, as well as for the RF regressor in Section VI-B2.

1) MLP CLASSIFIER

The first network trained during our development phase is the MLP classifier. In general, a large amount of data is required to train MLPs for minimizing the selected loss function and train them on how to generalize from the input dataset. The initial 5,685 samples of the dataset T are split into three sets: 70% for training, 1.5% for validation and 28.5% for

TABLE 9. Study of the Useful Information Exploitable for the ML Models for Each Scenario

Scenario	Samples	Drones	No-Drones	Multi-Drone	Samples %	Drones %	No-Drones %
scenario 1.1	799	126	673	0	14.05	15.76	84.23
scenario 1.2	838	233	605	0	14.74	27.8	72.19
scenario 1.3	504	162	342	0	8.86	32.14	67.85
scenario 1.4	568	225	343	0	9.99	39.61	60.38
scenario 2.1	1134	372	762	30	19.94	32.8	67.19
scenario 2.2	1227	671	556	55	21.58	54.68	45.31
scenario 3	615	107	508	0	10.81	17.39	82.60

TABLE 10. Representation of a Portion of the Generated Dataset as an Example With Some Rows and Columns

Timestamp	Latitude	Altitude	Speed	Alvira Latitude	Alvira Longitude	Alvira Altitude	Alvira Speed
1601456510	51.521736	39.7	13.86	51.52033776	5.86232428	32.28467599	8.54083729
1601456511	51.521832	39.8	14.0	51.52037459	5.86249421	33.22264418	8.82986069
1601456512	51.521912	39.8	13.86	51.5203907	5.86265874	34.19893019	8.95946789
1601456513	51.522006	39.8	13.93	0.0	0.0	0.0	0.0
1601456514	51.522085	39.8	13.93	51.52042446	5.86282519	35.14916094	8.95709419
1601456515	51.522181	39.8	13.86	51.52044779	5.86299669	36.16634507	9.2426815
1601456516	51.522259	39.8	13.86	51.5204803	5.86317456	37.17205324	9.52820396
1601456517	51.522346	39.9	13.86	0.0	0.0	0.0	0.0
1601456518	51.522434	39.7	13.93	51.52052729	5.86335146	38.18974604	9.64759445
1601456519	51.522522	39.7	13.79	51.52054668	5.86352701	39.26478952	9.72277355
1601456520	51.522609	39.8	14.07	51.52055977	5.86369514	40.29877538	9.31249523

testing. A standardization procedure is employed to enhance the network's training process, and a Label Encoder is used to encode categorical data in numerical values. For the output layer, One-Hot encoding is applied to represent the categories.

The training procedure aims to minimize a loss function evaluated during the training process on the train and validation sets. At the end of the process, the final network performances are evaluated on the test set. For the MLP network classifier, the selected loss function is the *Categorical Cross-Entropy Loss*, defined by 10. The predicted probability distribution for each input is a vector of C values, where C is the number of classes. Each value represents the probability of the input belonging to a specific class. The true probability distribution for each input is also a vector of C values with the same meaning. The categorical cross-entropy loss is defined as:

$$\mathrm{CE} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{10}$$

where CE is the value of the Categorical Cross-Entropy Loss, y_i and $\hat{y}_i$ are the true and predicted probability of the input belonging to class i, respectively.

The training procedure is performed during the development phase for 50 epochs, specifically with a mini-batch size of 5 and a *Learning Rate* of 0.003. Further, we adopt the Adam optimizer [36] to provide adaptive learning rates for faster and more stable convergence of our Deep Learning (DL) model. Fig. 4 shows the evolution of the loss value with the epochs.

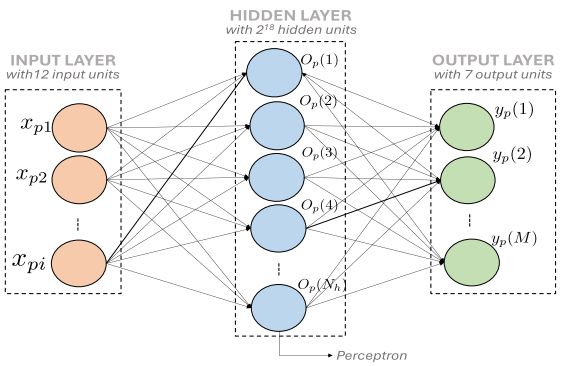

FIGURE 3. Schema representation of a MLPs network used in the URANUS framework for classification task. The considered version is structured with 12 neurons in the input layer, 2^{18} (512 × 512) neurons in the hidden layer(s), and 7 neurons in the output layer.

The final values of the loss for the last epoch are 0.18 and 0.13, for the training and validation phases, respectively. Table 11 shows a summary of the final performances of the MLP classifier, while Fig. 5 shows its Confusion Matrix. Specifically, the Confusion Matrix is used to evaluate the performance of classification models by summarizing the counts of true and false positives and negatives. The four metrics derived from the confusion matrix are *Precision*, *Recall*, *F1-score*, and the *Support of each class*, represented in (11),(12),

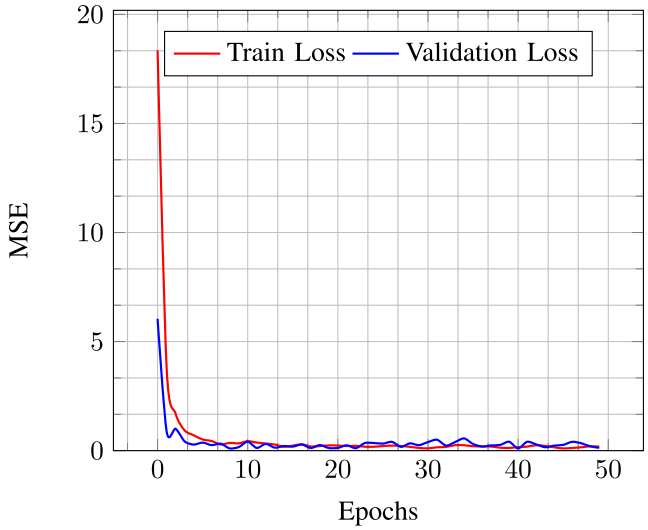

FIGURE 4. Training and Validation losses for the MLP classifier used for the *Identification* and *Classification* tasks.

TABLE 11. Summary of the Performances of the MLP Classifier Network

Classes	Precision	Recall	F1-Score	Support
		One-vs-All		
0	0.98	0.99	0.98	1,096
1	0.73	0.73	0.73	26
2	0.79	0.8	0.8	131
3	0.8	0.73	0.76	188
4	0.74	0.72	0.73	153
5	0.78	0.58	0.67	12
6	0.21	0.43	0.28	14
		Overall		
Accuracy	–	–	0.9	1,620
Macro Avg	0.72	0.71	0.71	1,620
Weighted Avg	0.91	0.9	0.91	1,620

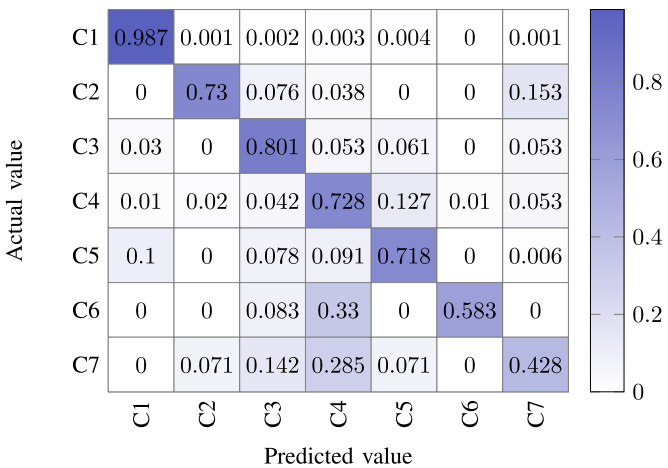

FIGURE 5. Confusion matrix of the MLP network classifier.

(13), and the number of samples in each class, respectively.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{13}$$

In details, the True Positive (TP) represents the model accurately predicts the presence of a drone, False Positive (FP) provides the inaccurate prediction of a drone's presence, and False Negative (FN) depicts an inaccurate prediction of a drone's absence.

2) RF REGRESSOR

Considering the RF network, we filtered the main dataset T by obtaining a sub-dataset with 1,343 samples. The 80% (1,074 samples) are used for the training phase, and the 20% (269 samples) for the test phase. It is worth noticing that we do not adopt any standardization procedure in this case, unlike what is done for the MLP classifier. We use the *Label Encoder* only for the categorical input columns in the derived subset of the initial dataset to manage them as numbers.

The final training results for the RF regressor are evaluated by using the Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 metrics as depicted in (14), (15) and (16), respectively:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} E[(\hat{y}_i - y_i)^2] \tag{14}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{15}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{16}$$

where n is the number of data points, y_i is the i-th observed value of the dependent variable, $\hat{y}_i$ is the i-th predicted value of the dependent variable, and $\bar{y}$ is the mean value of the dependent variable across all observations.

The MSE, MAE, and R^2 amount to 0.29, 0.04, 0.93, respectively. The results confirm the robustness and validity of the chosen model to perform this regression task.

Further, to assess the performance of the regressor, the trained model uses each scenario of the dataset T as input to estimate the medium difference between the real positions of the drone(s) and the predicted ones. Fig. 7 shows the mean differences between predicted and real drone positions in each scenario. In all the considered scenarios, the mean regression error is below 100 meters, and the models perform better when there is only one drone in the scenario. This behaviour is highly predictable, given the low number of sensors available for training the models and detecting drones.

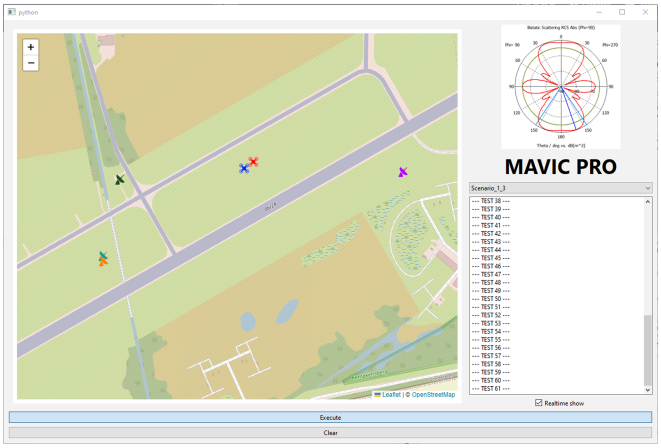

FIGURE 6. Screenshot of the proposed framework in real-time mode. The blue icon depicts the real position of the drone, while the red icon represents its predicted position. Meanwhile, the other radar icons depict the static positions of both radars and RFDF sensors.

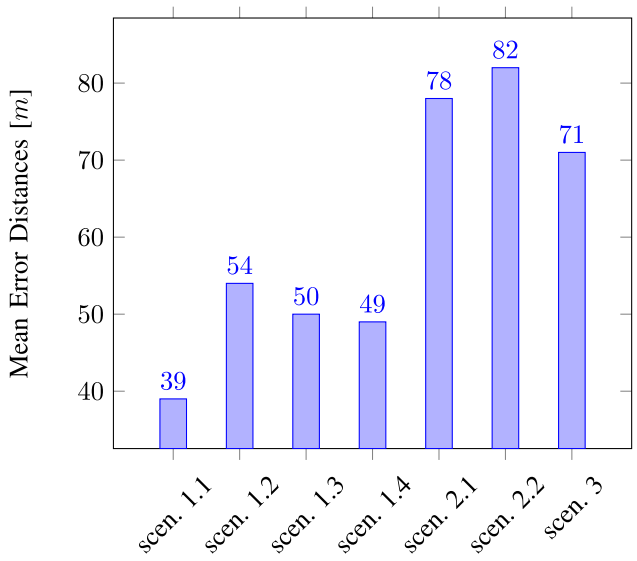

FIGURE 7. Mean differences (in meters) between predicted and real drone positions in the various scenarios.

C. SETUP AND IMPLEMENTATION

The development and test phases are performed on a custom desktop machine running Arch Linux with the Linux Kernel 6.2.11 and CUDA 12.1 [37]. The hardware includes an AMD Ryzen 7 2700*X* Eight-Core Processor with 32 GB of RAM and an NVIDIA 1080 GPU.

For the ML models design and implementation, we developed the solution in Python 3. In particular, we adopted the following libraries:

- *PyTorch* 1.13.1 [38]: machine learning framework used for developing and training neural network-based deep learning models in Python. We used this library to model the MLP network.
- *sklearn* 1.2.2 [39]: for the Standard Scaler and Label Encoder, the train/test dataset split and the Random Forest algorithm implementation.
- *numpy* 1.24.2: a common Python library useful to work with arrays and matrices.
- *pandas* 2.0 [40]: a common Python library, particularly for the DataFrame object.
- *matplotlib* 3.7.2: used to show and save graphs.
- *PyQt* 6.5 [38]: the binding of the Qt libraries for Python. This library has been used to create the application's main Graphical User Interface (GUI), exploiting its cross-platform nature.
- *Leaflet* 1.9.4 [41]: the Javascript Geographic Information System (GIS) library used to show real and predicted objects on the interactive map of the base.

Fig. 6 depicts the main functionalities of the URANUS framework, such as drone(s) identification, classification, and real-time tracking on the map.

VII. RELATED WORK

In this section, we review the state of the art on the Radio Frequency (RF) machine learning and deep learning approaches adopted to detect, identify and track drones. Table 12 summarizes the information of related literature and presents the main features of these methods.

For instance, Al-Sa'd et al. [42] collected, analysed, and recorded raw RF signals from several types of drones in different states. Furthermore, they leveraged a deep learning technique to detect and identify malicious drones and their flight mode. The authors designed three Deep Neural Networks (DNNs) to (i) detect the drone, (ii) detect the drone and recognize its type, and (iii) detect the drone, and recognize its type and its state. The authors do not consider fixed-wing drones, and they do not perform any path-tracking operation.

Basak et al. [43] focused on the development of (i) RF drone signal detection, (ii) spectrum localization, and (iii) drone classification by using a two-stage technique. In the first stage, they adopt the Goodness-of-Fit (GoF) sensing for drone detection and the Deep Recurrent Neural Network (DRNN) framework for drone classification. In the second stage, they use the You Only Look Once - lite (YOLO-lite) framework to perform the combined drone RF signal detection, spectrum localization, and drone classification. However, neither multiple detections of drones nor trajectory tracking on a map are considered.

Al-Emadi et al. [44] proposed a real-time RF drone detection and identification framework to inspect the radio spectrum between the drone and its controller. The solution adopts a Convolutional Neural Network (CNN) to train and test an RF dataset released by [42]. The experimental results show the effectiveness and feasibility of using RF signals in combination with a CNN to detect and identify a drone. The proposed solution achieves an F1 score of 99.7% for drone identification. Nevertheless, the authors do not consider fixed-wing drones and drone path-tracking operations.

TABLE 12. Comparison and Overview of Related Contributions on Drones RF Identification, Classification, and Tracking Using Machine Learning and Deep Learning Techniques

Ref.	Analysis on Dataset	Identification and Classification	ML/DL Model	Real Time	Multiple Drone Detection	Fixed-wing Drone Detection	Path-Tracking	Open Source Code
[42]	×	✓	DNN	×	✓	×	×	✓
[43]	×	✓	YOLO-lite	×	×	×	×	×
[44]	✓	✓	CNN	×	×	×	×	×
[45]	✓	✓	1D-CNN	×	×	×	×	×
[46]	✓	✓	XGBoost	×	×	×	×	×
[47]	✓	✓	FC-DNN	×	✓	×	×	×
[48]	✓	✓	DRNN	×	✓	×	×	×
[49]	✓	✓	KNN and XGBoost	×	×	×	×	×
[50]	×	✓	KNN, SVM and RF	×	×	×	×	×
URANUS	✓	✓	MLP + RF	✓	✓	✓	✓	✓

A ✓ symbol indicates the fulfillment of a particular feature, and a × symbol denotes the miss of the feature or that the feature is not applicable.

Allahham et al. [45] investigated deep learning techniques to perform (i) drone detection, (ii) drone detection and type identification, and (iii) drone detection, type and state identification by using a three multi-channel 1-dimensional CNN. The dataset adopted in the experiments is *Drone RF* dataset [51]. The performance for (i) shows an average accuracy of 100%, while (ii) has an accuracy of 94.6%, and, finally, the last one (iii) presents an accuracy of 87.4%.

The authors in [46] developed an RF machine-learning drone detection and identification system by analyzing the low-band RF signals emitted by the flight controller. They proposed three machine learning models based on eXtreme Gradient Boosting (XGBoost) algorithm to detect and identify (i) the presence of a drone, (ii) the presence of a drone and type, and (iii) the presence of a drone, type and the operational mode. The accuracy achieved by the three models is 99.96%, 90.73%, and 70.09%, respectively. The higher the model complexity, the lower the model accuracy. This latter implies the low effectiveness of using the frequency components of a signal as a signature to detect the activities performed by drones. From the results achieved by the models, we deduced that using the frequency components of a signal as a signature to detect drone activities is not very effective. Trajectory tracking on a map is not considered in this case.

Sazdić-Jotić et al. [47] proposed RF detection and identification algorithms to detect and identify single or multiple drones. They built an RF dataset by considering scenarios with (i) a single drone, (ii) two drones, and (iii) three drones. They detect and identify a single drone with an accuracy of 99.8% and 96.1%, respectively, while the results of detecting multiple drones show an average accuracy of 97.3%. The deep learning algorithms used are mainly Fully Connected Deep Neural Networks (FC-DNN). Although the approach performs well, the authors do not consider path tracking.

The authors in [48] presented a DRNN that classifies different drone signals in single-drone and multiple-drone scenarios. The authors built an RF dataset with nine commercial drone types, and further, they evaluated the proposed model in Additive white Gaussian noise (AWGN) and multipath conditions. The model achieved roughly 99% classification accuracy for single and simultaneous multi-drone scenarios. However, The described approach does not take into account drone path tracking and fixed-wing drones.

Ibrahim et al. [49] presented a UAV identification and hierarchical detection approach by leveraging an ensemble learning based on K-Nearest Neighbor (KNN) and XGBoost. The proposed solution can (i) check the availability of a UAV, (ii) specify the type of the UAV, and (iii) determine the flight mode of the detected UAV. This approach reaches a classification accuracy of around 99%. However, the authors do not consider drone path tracking.

Wei et al. [50], proposed a drone detection and identification system based on WiFi signals and high-frequency RF fingerprints. The system (i) performs UAV detection, (ii) extracts the features Fractal Dimension (FD), (iii) Axially Integrated Bispectra (AIB) and Square Integrated Bispectra (SIB) (iii) adopts the Principal Component Analysis (PCA) algorithm for the feature dimensionality reduction, and (iv) applies the Neighborhood Component Analysis (NCA) algorithm for metric learning. Finally, the authors test KNN, Support Vector Machine (SVM), and RF to identify UAVs. They verified their model in two different scenarios, i.e., indoor with a Signal-to-Noise Ratio (SNR) of 10 dB and outdoor with a SNR of 3 dB. In the indoor scenario, the average identification accuracy of FD, AIB, SIB is 100%, 97.23%, and 96.11% respectively. In the outdoor scenario, the identification accuracy of the same features is 100%, 95.00%, and 93.50%, respectively. The authors do not consider drone trajectory tracking.

To sum up, the discussion above confirms that despite there are several contributions to the state-of-the-art, none of them analyze and evaluate the proposed techniques on both

fixed-wing and multi-copter drones for the (i) detection, (ii) classification and (iii) simultaneous tracking, as well as the drone path tracking. Such constraints make previous solutions unsuitable for this problem and call for new domain-specific approaches. Moreover, none of the approaches in the current literature perform tracking, identification, and classification in real-time, but only offline.

VIII. CONCLUSION

In this paper, we proposed URANUS, a framework to prevent and detect unauthorized UAVs for Critical Infrastructures. URANUS can identify, classify, and track multi-copter and fixed-wing drones in real time. Our solution leverages two components: (i) a network of Radio Frequency/Direction Finding radar sensor network distributed in the No-Drone Zone, and (ii) Counter Unmanned Aerial System, a system adopted to collect and process the data generated by the radar sensor network and detect the presence of any unauthorized or malicious drone. URANUS features several properties such as: (i) it relies only on the wireless data collected from the RFDF sensor network; (ii) it can be extended to detect, classify and track different aerial vehicles at the same time; and (iii) it can be integrated with pre-existing drone detection solutions in compliance with the existing regulations. At the same time, our model has been trained on a dataset comprising UAV flights provided by NATO [18]. Our results show that the trained models achieve a good accuracy of 90% for the identification and classification tasks, and we can discriminate between UAVs and fixed-wings. The MLP model achieves an accuracy of 90% with a True Positive Rate (Recall) of ≈ 0.71, and a True Negative Rate of ≈ 0.98. The RF model achieves a MSE ≈ 0.29, MAE ≈ 0.04, $R^2 \approx 0.93$ on the final dataset. Finally, we highlight that we also released the source code of URANUS as open-source to foster the replicability of our results, encourage the deployment and extension of URANUS, and check the viability of further research directions.

ACKNOWLEDGMENT

We would like to acknowledge the use of the "Drone Detection" data source provided by the NATO Communications & Information Agency (NCIA). The findings reported here are solely the responsibility of the authors.

REFERENCES

[1] S. Liao, J. Wu, J. Li, A. K. Bashir, and W. Yang, "Securing collaborative environment monitoring in smart cities using blockchain enabled software-defined internet of drones," *IEEE Internet Things Mag.*, vol. 4, no. 1, pp. 12–18, Mar. 2021.

[2] M. Bacco, A. Berton, A. Gotta, and L. Caviglione, "IEEE 802.15.4 Air-ground UAV communications in smart farming scenarios," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1910–1913, Sep. 2018.

[3] Y. Inoue and M. Yokoyama, "Drone-based optical, thermal, and 3D sensing for diagnostic information in smart farming–Systems and algorithms–," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2019, pp. 7266–7269.

[4] Z. Zhou, C. Zhang, C. Xu, F. Xiong, Y. Zhang, and T. Umer, "Energy-efficient industrial internet of UAVs for power line inspection in smart grid," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2705–2714, Jun. 2018.

[5] Y.-J. Zheng, Y.-C. Du, H.-F. Ling, W.-G. Sheng, and S.-Y. Chen, "Evolutionary collaborative Human-UAV search for escaped criminals," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 217–231, Apr. 2020.

[6] H. Huang, A. V. Savkin, and W. Ni, "Online UAV trajectory planning for covert video surveillance of mobile targets," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 2, pp. 735–746, Apr. 2022.

[7] S. H. Alsamhi, O. Ma, M. S. Ansari, and S. K. Gupta, "Collaboration of drone and internet of public safety things in smart cities: An overview of qos and network performance optimization," *Drones*, vol. 3, no. 1, 2019, Art. no. 13.

[8] R. Di Pietro, G. Oligeri, and P. Tedeschi, "JAM-ME: Exploiting jamming to accomplish drone mission," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2019, pp. 1–9.

[9] The Insight Partners, "Drone logistics & transportation market size," 2019. [Online]. Available: https://www.theinsightpartners.com/reports/drone-logistics-and-transportation-market

[10] S. Lee, D. Hong, J. Kim, D. Baek, and N. Chang, "Congestion-aware multi-drone delivery routing framework," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9384–9396, Sep. 2022.

[11] B. D. Song, K. Park, and J. Kim, "Persistent UAV delivery logistics: MILP formulation and efficient heuristic," *Comput. Ind. Eng.*, vol. 120, pp. 418–428, 2018.

[12] M. Moshref-Javadi and M. Winkenbach, "Applications and research avenues for drone-based models in logistics: A classification and review," *Expert Syst. Appl.*, vol. 177, 2021, Art. no. 114854.

[13] F. Fabra, C. T. Calafate, J.-C. Cano, and P. Manzoni, "On the impact of inter-UAV communications interference in the 2.4 GHz band," in *Proc. IEEE 13th Int. Wireless Commun. Mobile Comput. Conf.*, 2017, pp. 945–950.

[14] Y. Zeng, J. Lyu, and R. Zhang, "Cellular-connected UAV: Potential, challenges, and promising technologies," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 120–127, Feb. 2019.

[15] C. Fu et al., "Siamese object tracking for unmanned aerial vehicle: A review and comprehensive analysis," *Artif. Intell. Rev.*, vol. 56, pp. 1–61, 2023.

[16] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "PPCA - privacy-preserving collision avoidance for autonomous unmanned aerial vehicles," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1541–1558, Mar./Apr. 2023.

[17] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet Things*, vol. 11, 2020, Art. no. 100218.

[18] North Atlantic Treaty Organization (NATO), "Drone detection - class i unmanned aircraft systems (UAS) tracking, classification and identification challenge," 2022. [Online]. Available: https://www.kaggle.com/c/icmcis-drone-tracking/

[19] P. Di Gennaro, D. Lofù, and P. Tedeschi, "Source code of URANUS," 2023. [Online]. Available: https://github.com/pdigennaro/uranus2/

[20] R. Lu and Y. Li, "A global calibration method for large-scale multi-sensor visual measurement systems," *Sensors Actuators A: Phys.*, vol. 116, no. 3, pp. 384–393, 2004.

[21] R. B. Langley, "The UTM grid system," *GPS World*, vol. 9, no. 2, pp. 46–50, 1998.

[22] M. Inggs, Y. Paichard, and G. Lange, "Passive coherent location system planning tool," in *Proc. IEEE Int. Radar Conf. Surveill. Safer World*, 2009, pp. 1–5.

[23] W. D. Blair, G. A. Watson, T. Kirubarajan, and Y. Bar-Shalom, "Benchmark for radar allocation and tracking in ECM," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 4, pp. 1097–1114, Oct. 1998.

[24] C. M. A. Lara, J. J. Navarro-Corcuera, F. Miehe, and F. Opitz, "Real-time optimized trajectories for 2D emitter localization using a UAVs team," in *Proc. IEEE 24th Int. Radar Symp.*, 2023, pp. 1–7.

[25] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[26] R. Lin, Z. Zhou, S. You, R. Rao, and C.-C. J. Kuo, "Geometrical interpretation and design of multilayer perceptrons," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 21, 2022, doi: 10.1109/TNNLS.2022.3190364.

[27] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[28] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[29] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges.* Berlin, Germany: Springer, 2019.
[30] J. P. Jiawei Han and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011.
[31] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1164–1176, Mar. 2022.
[32] P. Rodríguez, M. A. Bautista, J. Gonzalez, and S. Escalera, "Beyond one-hot encoding: Lower dimensional target embedding," *Image Vis. Comput.*, vol. 75, pp. 21–31, 2018.
[33] *Federal Aviation Admin.*, "No drone zone," 2021. Accessed: Sep. 30, 2023. [Online]. Available: https://www.faa.gov/uas/resources/community_engagement/no_drone_zone/
[34] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "ARID: Anonymous remote IDentification of unmanned aerial vehicles," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2021, pp. 207–218.
[35] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.*Cambridge, MA, USA: MIT Press, 2012.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–13.
[37] NVIDIA, "CUDA Toolkit," 2023. [Online]. Available: https://developer.nvidia.com/cuda-toolkit
[38] R. Computing, "PyQt," 2023. [Online]. Available: https://riverbankcomputing.com/software/pyqt/
[39] scikit-learn, "scikit-learn," 2023. [Online]. Available: https://scikit-learn.org
[40] Pandas, "Pandas," 2023. [Online]. Available: https://pandas.pydata.org,
[41] Leaflet, "Leaflet," 2023. [Online]. Available: https://leafletjs.com
[42] M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "RF-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database," *Future Gener. Comput. Syst.*, vol. 100, pp. 86–97, 2019.
[43] S. Basak, S. Rajendran, S. Pollin, and B. Scheers, "Combined RF-Based drone detection and classification," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 111–120, Mar. 2022.
[44] S. Al-Emadi and F. Al-Senaid, "Drone detection approach based on radio-frequency using convolutional neural network," in *Proc. IEEE Int. Conf. Inf. IoT Enabling Technol.*, 2020, pp. 29–34.
[45] M. S. Allahham, T. Khattab, and A. Mohamed, "Deep learning for RF-Based drone detection and identification: A multi-channel 1-D convolutional neural networks approach," in *Proc. IEEE Int. Conf. Inf. IoT Enabling Technol.*, 2020, pp. 112–117.
[46] O. O. Medaiyese, A. Syed, and A. P. Lauf, "Machine learning framework for RF-Based drone detection and identification system," in *Proc. IEEE 2nd Int. Conf. Smart Cities Autom. Intell. Comput. Syst.*, 2021, pp. 58–64.
[47] B. Sazdić-Jotić, I. Pokrajac, J. Bajčetić, B. Bondžulić, and D. Obradović, "Single and multiple drones detection and identification using RF based deep learning algorithm," *Expert Syst. Appl.*, vol. 187, 2022, Art. no. 115928.
[48] S. Basak, S. Rajendran, S. Pollin, and B. Scheers, "Drone classification from RF fingerprints using deep residual nets," in *Proc. IEEE Int. Conf. Commun. Syst. Netw.*, 2021, pp. 548–555.
[49] I. Nemer, T. Sheltami, I. Ahmad, A.-H. Yasar, and M. A. R. Abdeen, "RF-Based UAV detection and identification using hierarchical learning approach," *Sensors*, vol. 21, no. 6, 2021, Art. no. 1947.
[50] W. Nie, Z.-C. Han, M. Zhou, L.-B. Xie, and Q. Jiang, "UAV detection and identification based on WiFi signal and RF fingerprint," *IEEE Sensors J.*, vol. 21, no. 12, pp. 13540–13550, Jun. 2021.
[51] M. S. Allahham, M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "DroneRF dataset: A dataset of drones for RF-based detection, classification and identification," *Data Brief*, vol. 26, 2019, Art. no. 104313.

Open Access funding provided by 'Politecnico di Bari' within the CRUI CARE Agreement