Journal of Intelligent & Robotic Systems (2024) 110:74
https://doi.org/10.1007/s10846-024-02101-7

REGULAR PAPER

PADRE – A Repository for Research on Fault Detection and Isolation of Unmanned Aerial Vehicle Propellers

Radosław Puchalski[1] · Quang Ha[2] · Wojciech Giernacki[1] · Huynh Anh Duy Nguyen[2] · Lanh Van Nguyen[2]

Received: 31 October 2023 / Accepted: 18 April 2024 / Published online: 15 May 2024

Abstract
Unmanned aerial vehicles are being used increasingly in a variety of applications. They are more and more often operating in close proximity to people and equipment. This necessitates ensuring maximum stability and flight safety. A fundamental step to achieving this goal is timely and effective diagnosis of possible defects. Popular data-based methods require a large amount of data collected during flights in various conditions. This paper describes an open PADRE database of such measurements for the detection and classification of the most common faults - multirotor propeller failures. It presents the procedure of data acquisition, the structure of the repository and ways to use the various types of data contained therein. The repository enables research on drone fault detection to be undertaken without time-consuming preparation of measurement data. The database is available on GitHub at https://github.com/AeroLabPUT/UAV_measurement_data. The article also introduces new and universal quality indicators for evaluating classifiers with non-uniform parameters, are proposed. They allow comparison of methods tested for a variety of fault classes and with different processing times.

Keywords UAV · Fault detection · FDI · Sensor · IMU · Acoustic detection

1 Introduction

Recent years have shown the increasing number of applications for unmanned aerial vehicles (UAVs), much faster than previously expected. Moreover, the areas in which they are used are constantly being expanded. Drones are used for military [1], agricultural [2], marine monitoring [3], surface inspection [4], forest fire monitoring [5], medical emergency [6], educational [7], logistical [8], forming networks [9] or simply recreational purposes. As drones are often operating in critical tasks or in close proximity to people, the safety of their use has become an important aspect. Any malfunction of the actuator could end up crashing the drone. The tasks performed by UAVs themselves are required to be fail-safe and any failure to carry out a mission can result in serious consequences. The cost of maintaining and operating a fleet of flying robots is also becoming increasingly important. Individual vehicle parts wear out naturally. While manual pre- and post-mission inspection requires only time and possibly paying a crew to handle the task, in-flight inspection usually involves automated procedures. Wear and tear or minor deterioration of components is not always noticeable by humans. Therefore, new effective methods of drone fault detection (FD) are of prime importance. Adequate diagnostics, including determination of type, location and degree of defects, allow for early detection of abnormalities. Such knowledge can be used to alert the pilot, who will decide whether to continue the flight, automatically land the drone or use fault-tolerant control (FTC) [10] to continue the flight despite the malfunction.

Huynh Anh Duy Nguyen and Lanh Van Nguyen contributed equally to this work.

✉ Radosław Puchalski
radoslaw.puchalski@doctorate.put.poznan.pl

Quang Ha
quang.ha@uts.edu.au

Wojciech Giernacki
wojciech.giernacki@put.poznan.pl

Huynh Anh Duy Nguyen
huynhanhduy.nguyen@student.uts.edu.au

Lanh Van Nguyen
lanhvan.nguyen@student.uts.edu.au

[1] Faculty of Control, Robotics and Electrical Engineering, Poznan University of Technology, Piotrowo 3A, Poznań 60-965, Poland

[2] Faculty of Engineering and Information Technology, University of Technology Sydney, 15 Broadway, Sydney 2007, NSW, Australia

Researchers have long been working on effective methods for performing automated diagnostics and classification [11]. The solutions used are customarily divided into two basic groups: model-based methods and data-based methods [12]. The first ones require knowledge of the mathematical model of the drone being used. The more complex the model, the more accurately it can reflect the physical object. A higher level of complexity entails more sophisticated and time-consuming calculations. Since models are approximate virtual equivalents of real vehicles, these methods are always subject to a level of uncertainty. For this reason, among others, more and more work is based on model-free solutions.

In data-based methods, researchers use extensive sensor measurements from which they obtain electrical parameters or various static and dynamic characteristics of flight. Most often, researchers focus on measurements from inertial sensors. Accelerometers [13–15], gyroscopes [16–18] and magnetometers [19–21] are leading the way. Studies that use acoustic measurements [22–24] are also becoming more frequent. Among the most common components of multirotors whose faults can be detected using these methods are the motors, propellers and the sensors themselves [25]. However, it is the propeller, as the component that is in motion and usually protrudes beyond the UAV's fuselage outline, that is considered as the most susceptible to defects. On the one hand, the propeller is a relatively inexpensive and easily replaceable component. However, on the other hand, propellers are often not very resilient, despite being of colossal importance in flight stability and safety. The data-driven methods, as they are called, also have their drawbacks. One of them is the need for large sets of sensory data collected in these vehicles during flights under various conditions. This is not an easy task. In order to record measurements during the occurrence of various possible faults, these faults must either be artificially generated or numerically simulated. In the first case, it is possible to deliberately cause a defect to some elements of the drone, such as propellers, and then conduct experiments. Such a solution entails the destruction of working equipment. In addition, some malfunctions cannot be intentionally caused. Not all sensitive components are equally easy to access. Some defects do not allow flight or make it dangerous. It is also impossible to generate all possible faults. The second way, as with model-based methods, involves imperfection. A computer-simulated malfunction will never 100% match the actual failure and its real impact on other components and flight dynamics.

This paper responds to these needs. The publicly available PADRE – Propeller Anomaly Data REpository [26] is a solution for researchers who want to test their methods without the need for a time-consuming data acquisition process. With off-the-shelf data from various sensors, sorted by type, location and size of the fault, it is much easier to start substantive work on a fault detection method. Having at one's disposal extensive data sets in both time-domain and frequency-domain representation, it is possible, without time-consuming preparation, to train, validate and test various classifiers, compare their effectiveness, note possible problems and limitations.

2 Related Works

Scientists who conduct research on fault detection in unmanned aerial vehicle systems are usually forced to start their work by collecting enough data. They usually use readings from sensors with which the drone is equipped. If it is a commercial design, typically the number and quality of measurements and sampling frequency are severely limited. It could be better for custom designs, but this approach requires even more work. There is a lack of databases available that can be used to test one's own fault detection or classification methods. Typically, these databases are quite monotonous and concerned with specific types of data and a specific flying unit.

The authors in [27] have created an open database of orthorectified images using UAVs. They encourage drone community to expand the repository with their materials. Images of this type can be used to develop maps. However, this is not a database for UAV fault detection.

A large-scale dataset containing measurements from a ground-based platform equipped with an inertial measurement unit, as well as from the UAV inertial sensors, rotor tachometers and virtual cameras is presented in [28], but mainly for visual inertial navigation and 3D reconstruction.

Another example of datasets is for agriculture, which includes proximate and aerial images of various crops, weeds and disorders such as diseases and pests described in [29]. The collected data are expected to be useful in various field applications based on vision and artificial intelligence.

A database created specifically for UAV fault detection is the [30]. The authors of this work have prepared a database for propeller fault detection in a multirotor. They rely on three additional accelerometers mounted on the arms of a custom-fabricated hexarotor. About 5-minute flights took place in a flight cage. Single and double propeller failures on two of the UAV arms were tested.

An existing repository [31] includes data collected from flights on a *Carbon-Z T-28* fixed-wing aircraft. The dataset covers a variety of anomalies and was collected over a total of 66 minutes of flight time. The described database contains many different types of data, including those not processed at all. The investigated failures are also not equally represented by the collected data.

The most similar to our solution is [32]. It describes an extensive repository of data collected during *DJI Flame-Wheel F550* hexarotor flights with chipped propeller blades.

Table 1 Comparison of database contents for multirotor fault detection tasks

Database	UAV type	No. of UAVs	No. of variables	No. of faults	No. of scenarios	No. of flights	Place of flights
[30]	hexarotor	1	9+	1	4	20	outdoor cage
[32]	hexarotor	1	38	2	13	18	outdoor
PADRE	quadrotor	2	32	4	29	29	indoor/outdoor

The database contains MATLAB files that can be directly utilized by the user. The study used an ArduPilot controller with modified software so that measurements could be taken at high frequency. The authors investigated the occurrence of faults with varying degrees of part loss of the propeller. The undoubted advantage of the presented solution is the availability of various measurement data recorded from the drone and the controller, as well as the use of built-in sensors. The disadvantages are the need to modify the controller, the availability of data from only one drone model and only single propeller failures.

Table 1 summarizes a comparison of all databases known to the authors containing measurements for multirotor fault detection and classification tasks.

An undoubted advantage of our database is the posting of measurements from two UAVs, and more will be added over time. Both [30] and [32] include a wide variety of parameters provided from the flight controller. This increases the volume of data and makes it less readable. Not every measurement is suitable for detecting drone failures. In PADRE, all measurements are intended for fault detection and isolation (FDI) tasks. Our database contains flight data from the largest number of different defects and the largest number of various combinations of faults. This makes it possible to detect failures, indicate where they occur, and classify by the size of the damage. PADRE is the only database containing both indoor and outdoor flight data.

3 Data Acquisition and Fault Classification System

Commercial flying vehicles usually have a set of different sensors located near the centre of the fuselage. The data from them are not always freely available to the user. They also often have a low sampling rate. While one central inertial measurement unit (IMU) with a low sampling rate is usually sufficient to determine the orientation of the drone, it is insufficient for effective detection of faults in its drive units. To determine the location and type of fault occurrence in a motor or propeller, sensors need to be located close to them. As such, it was decided to develop a custom system for this purpose. The assumption was that it should be a universal circuit that can be used on different multirotor designs. Due to the use of different UAV power standards, the prepared system should have its own power supply and be energy efficient. In order not to change the behaviour of the drone, it should be lightweight and small in size. At the same time, it must be efficient enough to cope with complex mathematical computations, artificial intelligence algorithms and real-time processing. On top of that, it should facilitate an easy set up for use. The main tasks are to record data from sensors connected to the system and to detect and classify faults and anomalies during flight.

The prepared circuit is based on the high-performance STM32H743IIT6 microcontroller, which, together with the passive electronic parts, is placed on the underside of the PCB.

In the central part, there is a 4-position DIP switch, the setting of which defines the name of the file where the measurements are saved, depending on the specific fault. This shortens the post-processing of the data. The module is also equipped with 4 LEDs, which indicate the location of the fault in the classification mode. There are also connectors on the board: SWD for programming the circuit, UART for communication with the PC, and a power terminal for connecting a Li-Po battery (3.7 V, 250 mAh). The PCB size is approximately 66 mm x 66 mm. The appearance of the top layer of the board is shown in Fig. 1.

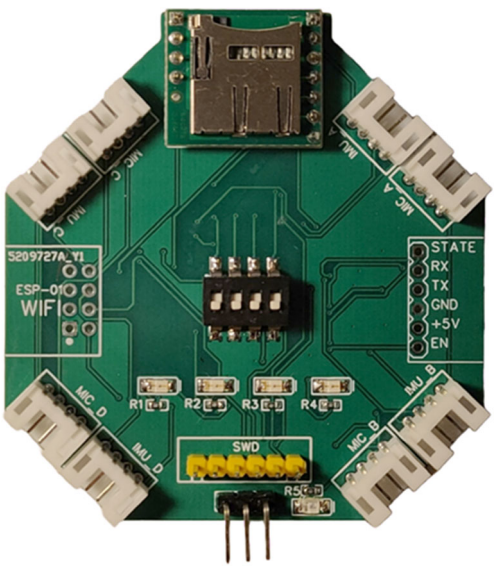

Fig. 1 Top layer of the PCB

Table 2 UAVs parameters [34, 35]

Drone	Parrot Bebop 2	3DR Solo
Dimensions	328 x 328 x 89 mm	250 x 250 x 260 mm
Weight	500 g	1500 g
Propeller size	15.2 cm	25 cm
Battery capacity	2700 mAh	5200 mAh
Flight time	up to 25 minutes	up to 25 minutes
Maximum speed	16 m/s	24.5 m/s
WiFi range	300 m	800 m

Up to now, the system has been used on two different quadcopters: *Parrot Bebop 2* and *3DR Solo*. Both are four-rotor drones in a *quad X* configuration. The basic parameters of those drones are shown in Table 2. These are only examples and do not exhaust the range of applications of the system. It is also possible to use the system for structures with a different number of propellers. Following the objectives of this study, training data for fault detection were collected from the same drone model on which the subsequent detection was performed. It might be possible to use data collected from another UAV with similar parameters, but less accuracy is then expected. However, such experiments have not been conducted.

Table 4 PDM microphone parameters [39]

Module	Adafruit 3492
Chip	MP34DT01-M
Supply voltage	1.64–3.6 V (module 1.8–3.3 V)
Input clock frequency	1–3.25 MHz
SNR	61 dB
Sensitivity	−26 dBFS

Flights of the *Bebop 2* drone were performed in the AeroLAB laboratory using the GY-6500 sensor. The *Solo* flights, on the other hand, took place in real-world conditions. GY-91 modules and PDM microphones were attached to the vehicle arms. The parameters of the inertial sensors are listed in Table 3, while those of the microphones are presented in Table 4.

3.1 Data Acquisition Mode

The primary objective of the developed system was to acquire sensory data from a number of sensors while flying different drones under conditions of various propeller failures. The operation of the algorithm in acquisition modes is shown in Fig. 2.

After initialising all the necessary peripherals and configuring the connected sensors, a CSV file is created on the

Table 3 IMU modules parameters [36–38]

Module	GY-6500	GY-91	
Chip	MPU-6500	MPU-9250	BMP-280
Supply voltage	1.17–3.45 V	2.4–3.6 V	1.71–3.6 V
Communication interfaces	I^2C, SPI		
Accelerometer			
ADC Word Length	16 bit		
Number of axes	3		
Range	±2, ±4, ±8, ±16 g		
Sensitivity	2048–16384 LSB/g		
Output data rate	4000 Hz		
Gyroscope			
ADC Word Length	16 bit		
Number of axes	3		
Range	±250, ±500, ±1000, ±2000 dps		
Sensitivity	16.4–131 LSB/dps		
Output data rate	8000 Hz		
Magnetometer			
ADC Word Length		14 bit	
Number of axes		3	
Range		±4800 μT	
Sensitivity		0.6 μT/LSB	
Barometer			
ADC Word Length			16–20 bit
Number of axes			1
Range			300–1100 hPa
Output data rate			26.7–181.8 Hz

memory card. Its name indicates which drone was used and the ranges of the connected sensors. In addition, by setting the DIP switch levers, damaged propellers are encoded in the file name. This avoids the need to change the programme after each flight with respect to a different location of the damaged propeller. Regular timer interrupts are triggered, the frequency of which defines the sampling rate of the inertial sensors. During each interrupt, data are taken from the measurement registers of all sensor axes. These are written to a buffer that collects subsequent measurements. When it is full, the data are saved to a memory card in a previously created CSV file.

Inertial sensors, such as an accelerometer or gyroscope, can exhibit misalignment between the actual and measured values. Usually using inertial sensors, the first step is to calibrate them. In [40], the authors propose a soft calibration procedure to reduce noise and systematic errors based on the drone's embedded camera. In this study, with trust in

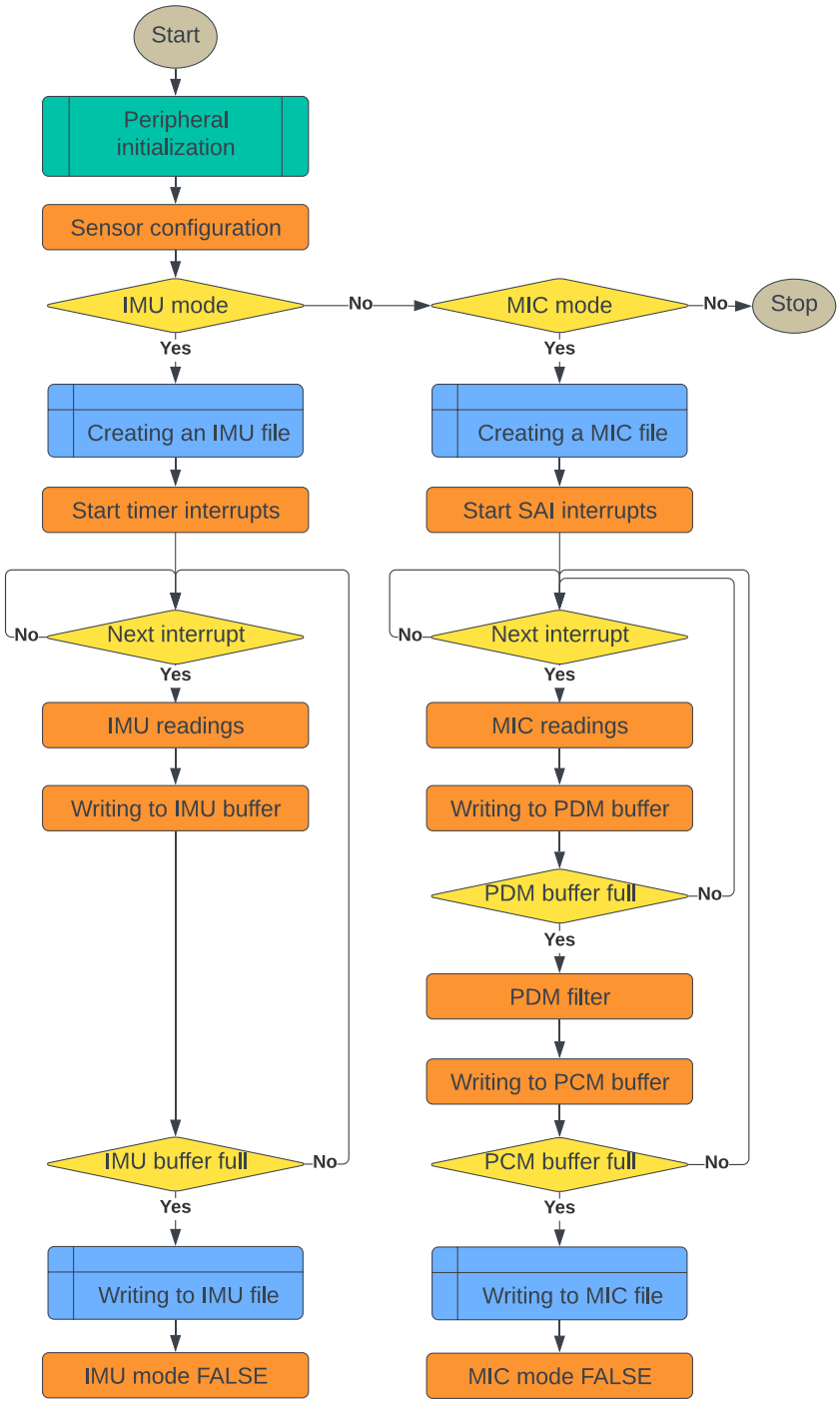

Fig. 2 Data acquisition mode algorithm

the manufacturer's specifications, it was decided to omit the time-consuming calibration procedure in favor of making the measurement process as simple as possible.

After recording the inertial sensor data, the acoustic data acquisition can proceed. The process looks similar. Instead of a clock interrupt, the function responsible for acquiring data from the PDM microphone is called. Once the PDM buffer is full, its data are processed into the pulse-code modulation (PCM) form [41]. The conversion consists of the following steps: low-pass filtering, decimation, high-pass filtering, and gain. In this PDM project, the sampling frequency is 3.072 MHz and the decimation ratio is 64. This gives an effective audio sampling rate of 48 kHz.

In this case, two files are stored on the memory card after each flight. The first contains data from the accelerometers, gyroscopes and barometers, the second file is acoustic data from the microphones.

During the *Bebop 2* drone flights, data were taken from 4 accelerometers (3 axes each) with a range of ±16 g and 4 gyroscopes (3 axes each) with a range of ±1000 dps. The *Solo* drone also took data from accelerometers (3 axes, ±16 g) and gyroscopes (3 axes, ±2000 dps), plus 4 barometers (16-bit) and 4 microphones. As the gyroscope range was insufficient in some of the *Bebop 2* propeller failure scenarios, it was decided to double the range in subsequent experiments. It is likely that some of the GY-91 modules were faulty or contained chips other than those declared by the manufacturer and the register values of the magnetometers could not be read. In that case, the subsequent data were abandoned.

3.2 Fault Detection and Classification Mode

As intended, the collected data can be used to prepare a fault classifier. After the fault classifier implementation in the microcontroller and an extension of the program, the system can operate in fault detection and classification modes. The study used two forms of data and two operating procedures of the program. Based on previously collected data, classifiers were prepared in the form of an artificial neural network (ANN) model. Two types of networks were used in the study. The first was a simple feed-forward network with one `dense` hidden layer. The best models had between 50 and 196 nodes in this layer. The second type was a convolutional neural network with one `conv1D` and one `flatten` layer. Models with the number of filters from 2 to 10 and kernel size from 4 to 360 were studied. Different combinations of sensor types and individual axes were used. Independently, two basic approaches can be distinguished: the use of data in the time domain and in the frequency domain. The main division of possible approaches concerns the frequency of the classification performed. Data can be processed by the classifier after collecting a pre-programmed number of samples, or classification can take place after each successive acquisition of sensory data. The second way requires the use of a circular buffer [42], whereby the oldest data are replaced by new ones and after the buffer is filled for the first time, there is always an adequate number of samples available. An advantage for this is that it involves very fast data processing. All operations from the start of acquisition, through their digital processing and classification must be completed before the next measurement. For the frequency sampling used in the study, which is 500 Hz for inertial sensors, the time of the above-mentioned operations must be less than 2 ms.

Figure 3 presents a simplified algorithm of the system operation in real-time fault classification for inertial sensors. After all the necessary peripherals are initialized and configured, a timer is started, whose interrupts are responsible for a constant sampling rate. These operations are performed once. The subsequent actions described take place in an infinite loop (or with a pre-programmed number of iterations). After each interrupt, data are received from the sensor registers. They are subjected to conversion to a floating-point number with a sign, and then normalized to a value from -1 to +1. In this form they go into the buffer. If the classifier has been trained using frequency data, a fast Fourier transformation is performed. Optionally, before applying it, the data can be multiplied by a selected window function [43]. This prevents the so-called frequency leakage, but increases processing time. Also optional is the selection of only specific frequency ranges [18]. This operation, in turn, makes

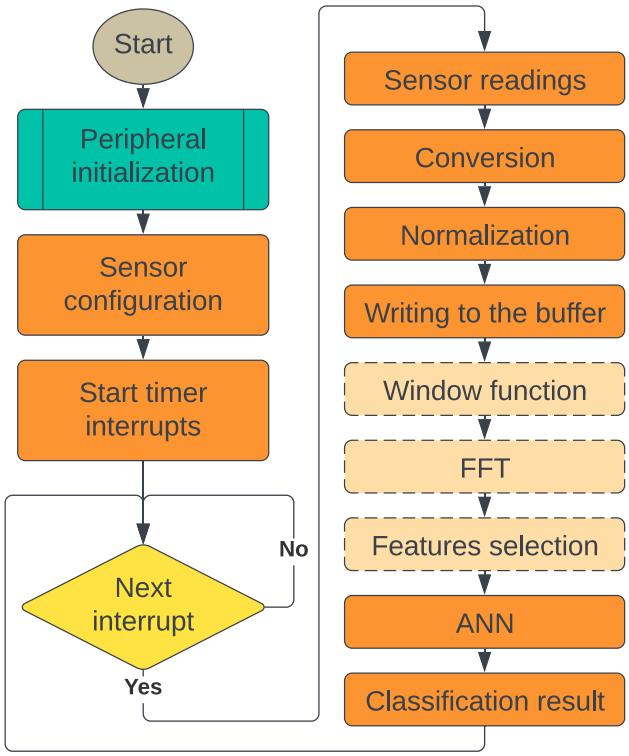

Fig. 3 Fault detection and classification mode algorithm

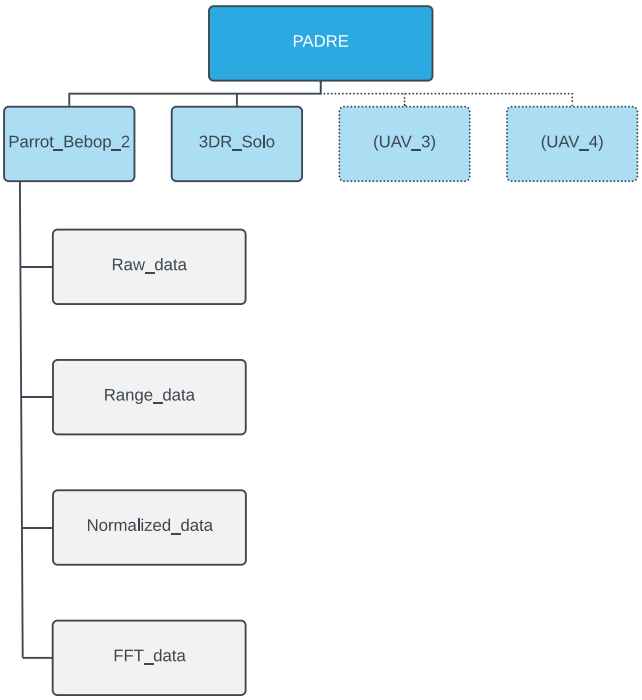

Fig. 4 Structure of the PADRE repository

it possible to use a simpler classifier and reduce the processing time of the neural network. The relevant data – in the time or frequency domain – goes to the input of the ANN, which performs the classification. After each subsequent timer interrupt, all operations running in the loop are repeated. In addition, for the purpose of testing the effectiveness of the classification, their results are stored on a memory card and displayed in real-time using four LEDs.

Brief ablation studies were conducted showing the impact of each module on the quality of the classifier. Three-axis accelerometer data collected during a *Bebop 2* drone flight for 5 different classes of propeller fault were used. A measurement window of 32 samples, a feed-forward neural network with one hidden layer containing 128 neurons, a batch size of 512, a dropout rate of 0.5, and an L2 regularization of 0.01 were used. Training was conducted for 2000 epochs. For selection features, the frequency range of about 94–188 Hz was analyzed. In other cases, the entire spectrum (0–250 Hz) was examined. The metrics of the models with the highest validation accuracy for each case are shown in Table 5.

Loss and accuracy values do not clearly indicate the validity of using or rejecting particular digital processing modules. Especially, since unified parameters and hyperparameters of the network were used for their preparation. Later experiments, the results of which are discussed in Section 5.2, indicated that this issue should be approached more individually. Depending on the signals used, it is worthwhile to select the width of the measurement window, the domain of the signals or the range of processed frequencies in more detail when using FFT. In particular, attention should be paid to the computing time requirements that will determine the use of particular blocks for processing sensor measurements.

4 Repository Structure and Development

4.1 PADRE Structure

The PADRE database was created to provide easily accessible and easy-to-use drone sensor measurement data for fault detection tasks. This open source repository was created on GitHub in a project called `UAV_measurement_data`. At the time of writing, it contains data from the two mentioned quadcopters, namely the *Parrot Bebop 2* and *3DR Solo*.

The repository includes raw measurements from various sensors, as well as processed data, ready for direct use, for example, at the stage of classifier training. All data comes from the sensors connected to the system as described in Section 3. No control signals or data from other sources are included. The repository is planned to be systematically expanded with measurement data from other drones and using other sensors. The general structure of our repository is shown in Fig. 4.

4.1.1 Parrot Bebop 2

Data from the *Bebop 2* drone were collected during 20 flights, each lasting exactly 172.032 seconds. Two types of damage were considered: a chipped edge (fault number *1*), a bent propeller tip (fault number *2*) and a serviceable propeller

Table 5 Effect of individual processing blocks on classifier metrics

Window function	FFT	Features selection	Train loss	Train accuracy	Valid loss	Valid accuracy	Test loss	Test accuracy
yes	yes	yes	0.1485	0.9586	0.1377	0.9698	0.1429	0.9604
yes	yes	no	0.1292	0.9690	0.1060	0.9828	0.1051	0.9817
no	yes	yes	0.1236	0.9676	0.1109	0.9776	0.1153	0.9698
no	yes	no	0.0996	0.9789	0.0947	0.9823	0.0867	0.9828
no	no	no	0.1400	0.9760	0.1355	0.9812	0.1317	0.9792

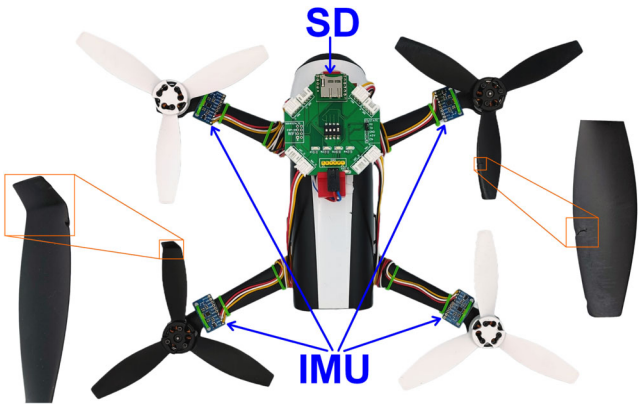

Fig. 5 Deployment of sensors and appearance of faulty propellers in Bebop 2 drone

(fault number *0*). Data were taken during flights with different combinations of defects. A 4-digit coding was used: the first digit corresponds to propeller A, the second digit to propeller B, the third digit to propeller C, and the fourth digit to propeller D. The digit on each item indicates a specific type of fault. The 20 flights carried out consist of individual classes (scenarios) of failures:

- all propellers functional: *0000*,
- single propeller fault: *1000*, *0100*, *0010*, *0001*, *2000*, *0200*, *0020*, *0002*,
- two propeller fault: *1100*, *1020*, *1002*, *0120*, *0102*, *0022*,
- three propeller fault: *1120*, *1102*, *1022*, *0122*,
- all propellers inoperative: *1122*.

The faulty propellers and the placement of the sensors during the *Bebop 2* drone experiment are shown in Fig. 5. During flights, data from 3 axes of each of the 4 accelerometers and 3 axes of each of the 4 gyroscopes were taken 500 times per second. The range of the accelerometers was set to ±16 g, while that of the gyroscopes was set to ±1000 dps. During each flight, 86016 measurements from each of the 24 sensor axes were recorded and stored on a memory card as a CSV file.

4.1.2 3DR Solo

The data from the *3DR Solo* drone were collected during 9 flights, each of which lasted – just like with the *Bebop 2* – 172.032 seconds. Due to the high sampling rate of the microphones, and thus the large volume of the files with these data, the acquisition was reduced 4 times to 43.008 seconds for the audio data. Two degrees of fault were considered: loss of 1 cm of propeller tip (fault number *1*) and loss of 2 cm of propeller tip (fault number *2*). The data collected are for flights with different combinations of faults occurring on one or two propellers. The 9 flights carried out consist of individual classes:

- all propellers operational: *0000*,
- single propeller fault: *2000*, *0200*, *0010*, *0001*,
- fault of two propellers: *2010*, *2001*, *0210*, *0201*.

Subsequent numbers correspond to the following propellers A, B, C and D. In this UAV model, the PCB is located under the fuselage of the drone. The damaged propellers and sensor placement (inertial measurement units IMU and microphones MIC) on the drone during experiments with the *Solo* is shown in Fig. 6.

In these experiments, data were collected from 3 axes of each of 4 accelerometers, 3 axes of each of 4 gyroscopes, single axes of 4 barometers, and 4 digital microphones. Frequency sampling for the accelerometer and gyroscope was 500 Hz, for the microphone – 48 kHz. The barometer data were recorded just like the inertial sensor data 500 times per second. However, the actual rate of change of the pressure sensor readings for 16-bit resolution was about 2 times lower. The range of the accelerometers was set to ±16 g, while the gyroscopes were set to ±2000 dps. During each flight, 86016 measurements were recorded from the inertial and barometer sensors and 2064384 measurements from the microphone. All the data were saved to a memory card. Because of the different sampling rate and the different nature of the microphone data, the acoustic measurements were saved to a separate file than those from the other sensors.

4.2 Time Domain Data

4.2.1 Raw and Range Data

The repository contains several different types of data. Raw sensor readings from *Bebop 2* are stored in hexadecimal form. Each sensor axis consists of two 8-bit registers, so there are

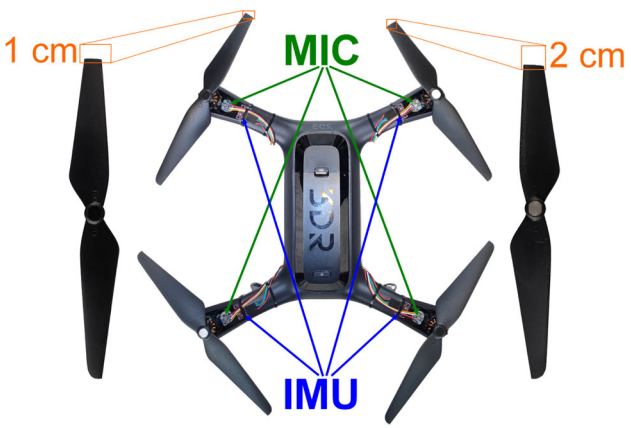

Fig. 6 Deployment of sensors and appearance of faulty propellers in Solo drone

4 ASCII characters per axis. For 24 axes, this gives a total of 96 characters in each of the 86016 lines for a given fault class. These are arranged in following order:

A_ax_H, A_ax_L, A_ay_H, A_ay_L, A_az_H, A_az_L,
A_gx_H, A_gx_L, A_gy_H, A_gy_L, A_gz_H, A_gz_L,
B_ax_H, B_ax_L, B_ay_H, B_ay_L, B_az_H, B_az_L,
B_gx_H, B_gx_L, B_gy_H, B_gy_L, B_gz_H, B_gz_L,
C_ax_H, C_ax_L, C_ay_H, C_ay_L, C_az_H, C_az_L,
C_gx_H, C_gx_L, C_gy_H, C_gy_L, C_gz_H, C_gz_L,
D_ax_H, D_ax_L, D_ay_H, D_ay_L, D_az_H, D_az_L,
D_gx_H, D_gx_L, D_gy_H, D_gy_L, D_gz_H, D_gz_L.

The first letter is the designation of the propeller, the lower case *a* is the accelerometer, *g* is the gyroscope, and *x*, *y* and *z* are the subsequent axes of the sensor. *H* is the most significant byte, *L* is the least significant byte. There are 20 such files in the `Raw_data` directory. The example file name is `Bebop2_16g_1kdps_raw_1022`. It includes information about the drone used, the range of the accelerometer and gyroscope, and the 4-digit class number: fault number *1* for propeller A, operable propeller B, fault number *2* for propellers C and D.

Another type of data is data converted to floating-point numbers with a sign. They have been scaled to correspond to the ground acceleration [g] for the accelerometer and degrees per second [dps] for the gyroscope. The `Ranged_data` folder also includes 20 CSV files for *Bebop 2* sensor readings. Each contains 86016 lines in the layout:

A_ax, A_ay, A_az, A_gx, A_gy, A_gz,
B_ax, B_ay, B_az, B_gx, B_gy, B_gz,
C_ax, C_ay, C_az, C_gx, C_gy, C_gz,
D_ax, D_ay, D_az, D_gx, D_gy, D_gz.

An example file name from this folder is `Bebop2_16g_1kdps_raw_1022`. The markings are identical to those of raw data. They are most suitable for analysis in terms of values of linear accelerations and angular velocities in individual sensor axes.

During *Solo* drone flights, it was decided to save the data in pure binary form. This reduces the file size by 2 times. The HEX form requires 2 ASCII characters to store 8 bits of data. Here, the data are recorded in their original form. On the other hand, this also reduces the writing time, the conversion step to HEX form, hence, is omitted. The only downside of this approach is the need to convert these data later to make them human-understandable. The file structure itself has not changed. Each inertial measurement file still contains 86016 lines. In addition, the files can also store data from barometers. The measurements are saved in the layout:

A_ax_H, A_ax_L, A_ay_H, A_ay_L, A_az_H, A_az_L,
A_gx_H, A_gx_L, A_gy_H, A_gy_L, A_gz_H, A_gz_L,
B_ax_H, B_ax_L, B_ay_H, B_ay_L, B_az_H, B_az_L,
B_gx_H, B_gx_L, B_gy_H, B_gy_L, B_gz_H, B_gz_L,
C_ax_H, C_ax_L, C_ay_H, C_ay_L, C_az_H, C_az_L,
C_gx_H, C_gx_L, C_gy_H, C_gy_L, C_gz_H, C_gz_L,
D_ax_H, D_ax_L, D_ay_H, D_ay_L, D_az_H, D_az_L,
D_gx_H, D_gx_L, D_gy_H, D_gy_L, D_gz_H, D_gz_L,
A_bar_H, A_bar_L, A_bar_X,
B_bar_H, B_bar_L, B_bar_X,
C_bar_H, C_bar_L, C_bar_X,
D_bar_H, D_bar_L, D_bar_X.

Data from barometers can be acquired with a resolution of 16 to 20 bits. The first two registers are always used. When taking 16-bit barometer data, the `press_xlsb (bar_X)` register is not used. For 17-bit data, only the most significant bit of the `press_xlsb` register is used, for 18-bit data – two bits, for 19-bit data – three bits, and for 20-bit data – four bits. On the other hand, regardless of the resolution used, all three 8-bit registers of the pressure sensor are always written to the card. During experiments with the *Solo* drone, 16-bit resolution was used in order to obtain the maximum possible acquisition rate. The other designations are identical to those for the *Bebop 2* drone. There are 9 such files in the `Raw_data` directory corresponding to 9 different damage classes of the *Solo* drone.

Data from digital microphones were stored in separate files. In this case, for each class of fault, there were 2064384 lines with the structure:

A_mic, B_mic, C_mic, D_mic.

Each variable is a 16-bit number with no sign [44]. At a later stage, these data are converted to 16-bit numbers with sign, which is a more natural representation of audio signals.

Data from the *Solo* drone have also been converted to common units – [g] and [dps]. Microphone readings were not included in this type of data. Barometric measurements were also omitted. In order to minimize the number of necessary computations, calculations of atmospheric pressure, for which some mathematical transformations are required, and consideration of temperature changes, were not performed. These data are not needed for fault detection tasks. Thus, the structure of these files corresponds entirely to the contents of the folder `Ranged_data` for the drone *Bebop 2*.

4.2.2 Data Normalization

The data in the `Normalized_data` folder are similar to ranged data. They too contain floating-point numbers with a sign, normalized to values from -1 to $+1$. For measurements from *Bebop 2*, -1 corresponds to -16 g, $+1$ is $+16$ g for the accelerometer, for the gyroscope it is -1000 dps and $+1000$ dps, respectively. The contents of the folder and the corresponding files correspond to those in the folder described above. Normalized data are also ready for training artificial neural network models. Normalization by 0 to

+1 can also be used, but due to the directional nature of both sensors, it was decided to normalize to values with sign. To evaluate the repository performance for the detection and classification of propeller faults, we conduct a statistical analysis, which is described in Section 5.1.

In *Solo* data case, the extreme values for the gyroscope – due to the increased measurement range – correspond to 2 times higher angular velocities than the same values for the *Bebop 2*. For this type of data, the measurements from the barometer were incorporated into the readings from the inertial sensors. The structure of these files is as follows:

A_ax, A_ay, A_az, A_gx, A_gy, A_gz, A_bar,
B_ax, B_ay, B_az, B_gx, B_gy, B_gz, B_bar,
C_ax, C_ay, C_az, C_gx, C_gy, C_gz, C_bar,
D_ax, D_ay, D_az, D_gx, D_gy, D_gz, D_bar.

Since atmospheric pressure was not taken into account for these data, the normalization range from 0 to +1 was used. 0 corresponds to zero values from the registers, while +1 is the maximum theoretical reading from 20 bits of registers $(2^{20} - 1)$.

The microphone data were normalized from −1 to +1 taking into account the extreme values of 16-bit integers with sign.

4.3 Frequency Domain Data

4.3.1 Fast Fourier Transform (FFT)

All of the folders described above contained time domain data. Successive lines of files corresponded to consecutive measurements taken with a fixed period. They were differentiated by an interval of 2 ms. The `FFT_data` folder, on the other hand, contains data from the frequency domain. These are normalized data subjected to a discrete Fourier transformation in the form of an FFT [45].

Due to the speed of operation and the high efficiency of propeller fault detection and classification confirmed in earlier studies [18, 46, 47], it was decided to just use FFT. The application of more complex transformations, especially the combination of different methods, would have increased the processing time. Further comparison of the FFT, for example with the Hilbert-Huang or Discrete Wavelet transform, is planned for a subsequent study. The fast Fourier transformation is widely used because of its much lower computational power compared to the Fourier transformation. However, there are some limitations associated with its use. In order to avoid the so-called frequency leakage, i.e., dissipation of energy from the signal's true frequency to adjacent frequencies, the input signal must be periodic within the sample window. The signal should start and end at the same point in its cycle. To get around this limitation, window functions are used, by which all the samples from the analyzed portion of the signal are multiplied. Popular window functions include rectangular window, Hann, Hamming, flat top, Nuttall, Blackman, Blackman-Harris, among others [48]. When examining the effectiveness of the various models trained on the PADRE database, the Hann window is selected as it showed the best performance for a process $x(n)$. The Hann window function, by which successive samples from the processed measurement window are multiplied, is represented by:

$$x_w(n) = x(n) \cdot 0.5 \left(1 - cos \frac{2\pi n}{N-1}\right), \tag{1}$$

where N is the number of processed samples of the measurement window, and n is the consecutive sample number in the time domain signal.

In order to change the signal domain from time to frequency, for the N-point fast Fourier transformation, the following was applied:

$$X(k) = \frac{1}{N} \left| \sum_{k=1}^{N} x(n) \cdot exp\left(-2\pi j(k-1)\frac{(n-1)}{N}\right) \right|, \quad \text{for } 1 \le k \le N, \tag{2}$$

where k is the number of the next bar corresponding to a particular frequency.

Since the frequency spectrum of the signal obtained by the above transformation is, so to speak, a mirror image with respect to the middle sample, the duplicate bars were omitted. For the N-point transform, only $N/2+1$ points were further processed. In addition, the most characteristic frequencies can be selected from the above range, further reducing the computational complexity and classification time.

4.3.2 FFT Data Organization

The FFT data were placed in folders named as follows:

- frame length - the number of samples subjected to Fourier transform,
- applied window function,
- lower frequency range,
- upper frequency range.

The example file name is `Bebop2_16g_FFT_ACCEL_128_Hann_16_36_1022`, which contains information about the frame length equal to 128 samples, the use of the Hann window function, the lower frequency cutoff at the 16-bar level and the upper frequency cutoff at the 36-bar level, `1022` is the standard fault class designation. The frequency corresponding to the lower and the higher frequency cutoffs

are determined from the formulas:

$$f_L = \frac{f_s}{N}(L-1), \tag{3}$$

$$f_H = \frac{f_s}{N}(H-1), \tag{4}$$

where f_s is the sampling frequency, L and H are the lower and upper bar numbers. In the example described here, for $f_s = 500$ Hz, these are the thresholds of about 58.6 Hz and 136.7 Hz, respectively.

This type of frequency data – like normalized time-domain data – is also suitable for direct use in training an artificial intelligence classifier.

Frequency data for *Solo* were prepared analogously to that for *Bebop 2*. Data from various sensors and their combinations, especially accelerometers and gyroscope, were prepared separately. Data from microphones – due to differences in the acquisition frequency range compared to inertial sensors – always appear in separate files.

5 Performance Evaluation

5.1 Statistical Metrics

All data from sensor readings were checked for completeness, integrity and correctness. Table 6 shows the basic statistical parameters of the normalized *Bebop 2* time domain data. Each axis of the sensors used for all 20 fault classes studied is included.

Columns of the table show the number of measurements, mean value Eq. 5, standard deviation Eq. 6, minimum value, first quartile, median, third quartile, and maximum value.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5}$$

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{6}$$

Statistical data from sensor readings mounted on the *Solo* are presented in Table 7.

Table 6 Parrot Bebop 2 data statistics

variable	count	mean	std	min	25%	50%	75%	max
A_aX	1720320	-0.2722	0.1628	-0.8452	-0.3961	-0.2803	-0.1576	0.6501
A_aY	1720320	0.0238	0.1039	-1.0000	-0.0220	0.0342	0.0835	1.0000
A_aZ	1720320	0.0607	0.2697	-1.0000	-0.0121	0.0585	0.1298	1.0000
A_gX	1720320	-0.0006	0.4276	-1.0000	-0.0933	-0.0006	0.0918	1.0000
A_gY	1720320	0.0020	0.0664	-0.4874	-0.0216	0.0020	0.0255	0.4908
A_gZ	1720320	0.0014	0.0365	-0.3734	-0.0228	0.0014	0.0256	0.4871
B_aX	1720320	0.0099	0.0854	-0.4488	-0.0492	0.0066	0.0653	0.4988
B_aY	1720320	-0.0494	0.2469	-1.0000	-0.2033	-0.0512	0.1019	1.0000
B_aZ	1720320	0.0678	0.2169	-0.7701	-0.0717	0.0651	0.2086	1.0000
B_gX	1720320	-0.0010	0.3809	-1.0000	-0.2503	-0.0010	0.2487	1.0000
B_gY	1720320	-0.0031	0.0267	-0.1646	-0.0202	-0.0029	0.0140	0.1767
B_gZ	1720320	0.0001	0.0496	-0.4001	-0.0324	0.0002	0.0327	0.4014
C_aX	1720320	0.0325	0.0793	-0.4896	-0.0228	0.0367	0.0934	0.4602
C_aY	1720320	-0.0349	0.2477	-1.0000	-0.1861	-0.0579	0.1082	1.0000
C_aZ	1720320	0.0541	0.1928	-1.0000	-0.0723	0.0517	0.1794	1.0000
C_gX	1720320	-0.0164	0.2762	-1.0000	-0.1826	-0.0164	0.1493	1.0000
C_gY	1720320	0.0051	0.0294	-0.2045	-0.0138	0.0051	0.0239	0.1959
C_gZ	1720320	-0.0011	0.0494	-0.4028	-0.0354	-0.0011	0.0332	0.4198
D_aX	1720320	-0.0452	0.0723	-0.7101	-0.0986	-0.0462	0.0030	0.6591
D_aY	1720320	0.0993	0.1391	-1.0000	0.0520	0.1090	0.1617	1.0000
D_aZ	1720320	0.0666	0.2542	-1.0000	-0.0943	0.0578	0.2251	1.0000
D_gX	1720320	0.0063	0.3121	-1.0000	-0.1716	0.0066	0.1839	1.0000
D_gY	1720320	-0.0008	0.0321	-0.1262	-0.0260	-0.0006	0.0243	0.1377
D_gZ	1720320	-0.0009	0.0587	-1.0000	-0.0329	-0.0012	0.0306	1.0000

Table 7 3DR Solo data statistics

variable	count	mean	std	min	25%	50%	75%	max
A_aX	774144	-0.0037	0.2038	-1.0000	-0.1259	-0.0047	0.1137	0.9931
A_aY	774144	0.0269	0.3175	-1.0000	-0.1225	0.0180	0.1623	0.9960
A_aZ	774144	0.0038	0.2033	-1.0000	-0.1173	0.0156	0.1301	0.9906
A_gX	774144	-0.0040	0.0129	-0.1204	-0.0113	-0.0040	0.0033	0.0807
A_gY	774144	-0.0042	0.0181	-0.1431	-0.0157	-0.0043	0.0072	0.1109
A_gZ	774144	-0.0037	0.0311	-0.2542	-0.0186	-0.0035	0.0115	0.3060
A_bar	774144	0.3431	0.0012	0.3410	0.3419	0.3437	0.3440	0.3463
A_mic	18579456	0.0000	0.1824	-1.0000	-0.1160	0.0152	0.1304	1.0000
B_aX	774144	-0.0057	0.2072	-1.0000	-0.1165	-0.0022	0.1055	0.9954
B_aY	774144	-0.0028	0.3683	-1.0000	-0.1626	-0.0086	0.1702	0.9961
B_aZ	774144	0.0671	0.1453	-0.8232	-0.0243	0.0593	0.1502	0.9577
B_gX	774144	-0.0035	0.0142	-0.1110	-0.0114	-0.0033	0.0050	0.1050
B_gY	774144	-0.0060	0.0292	-0.1487	-0.0231	-0.0060	0.0113	0.1970
B_gZ	774144	-0.0061	0.0470	-0.2106	-0.0342	-0.0062	0.0217	0.3297
B_bar	774144	0.3829	0.0012	0.3801	0.3817	0.3834	0.3838	0.3862
B_mic	18579456	0.0000	0.1639	-1.0000	-0.1088	0.0007	0.1073	1.0000
C_aX	774144	-0.0012	0.1911	-1.0000	-0.0977	0.0018	0.0975	0.9922
C_aY	774144	-0.0091	0.2052	-1.0000	-0.1290	-0.0047	0.1261	0.9922
C_aZ	774144	0.0596	0.1209	-0.8062	-0.0113	0.0590	0.1313	0.9192
C_gX	774144	-0.0059	0.0135	-0.1221	-0.0142	-0.0061	0.0021	0.1088
C_gY	774144	-0.0022	0.0245	-0.1180	-0.0177	-0.0020	0.0132	0.1351
C_gZ	774144	-0.0045	0.0262	-0.1659	-0.0220	-0.0042	0.0134	0.1591
C_bar	774144	0.3149	0.0012	0.3125	0.3137	0.3156	0.3159	0.3188
C_mic	18579456	0.0000	0.1797	-1.0000	-0.1176	-0.0037	0.1128	1.0000
D_aX	774144	-0.0030	0.1852	-1.0000	-0.1079	-0.0004	0.1039	0.9958
D_aY	774144	-0.0049	0.2720	-1.0000	-0.1749	-0.0069	0.1568	0.9958
D_aZ	774144	0.0618	0.1473	-0.8800	-0.0291	0.0591	0.1516	0.8403
D_gX	774144	-0.0052	0.0164	-0.1328	-0.0156	-0.0048	0.0051	0.1478
D_gY	774144	-0.0038	0.0241	-0.1494	-0.0190	-0.0039	0.0115	0.1128
D_gZ	774144	-0.0036	0.0256	-0.1485	-0.0202	-0.0049	0.0109	0.1590
D_bar	774144	0.3736	0.0012	0.3712	0.3725	0.3743	0.3746	0.3774
D_mic	18579456	0.0000	0.1455	-1.0000	-0.0893	0.0017	0.0926	1.0000

Histograms of all *Bebop 2* readings in the form of kernel density estimation (KDE) plots [49] are graphically presented in Fig. 7.

The KDE graph reproduces the shape of the kernel density estimator function described by:

$$\hat{f}_h(x) = \frac{1}{N}\sum_{i=1}^{N} K_h(x - x_i). \tag{7}$$

Figure 8 presents KDE histograms for accelerometer, gyroscope and barometer readings from *Solo*. KDE plots for microphone readings are graphically presented in Fig. 9.

In turn, the correlation of the different axes with respect to each other for *Bebop 2* is presented in Fig. 10 using Pearson's correlation coefficient Eq. 8 [50] for two variables x and y:

$$r_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right). \tag{8}$$

Values greater than zero represent positive correlation, values less than zero represent negative correlation. Correlations of sensor axes from *Solo* are shown in Figs. 11 and 12 for inertial sensors (including the barometer) and microphones, respectively.

To assess the quality of classifiers, a few basic indicators are usually applied: accuracy, precision, recall, F1 score [51]. Less frequently, area under the receiver operating characteristics curve [52], Cohen's kappa [53] or Matthews correlation

coefficient [54] can be used. These provide various useful information for the specific application of a given classifier. But precisely because of the very wide range of applications and therefore different requirements, it is very difficult to compare various classifiers with each other. It is obvious that dividing, for example, into 20 different classes is a much more difficult task than a simple binary classification. However, there is a lack of indicators in the scientific literature that take this factor into account. The same is true for the time taken to perform the classification. In many cases it is not an important factor, but in some cases it is crucial. Particularly in the UAV fault detection problem under discussion, the occurrence of propeller damage during flight requires an immediate response.

For this reason, we propose two new metrics to compare classifiers in general with a particular consideration for UAV propeller fault detection. The first metric is **Accurate Classification Quality**: Q_{AC}, to indicate the accuracy considered for all classes of faults:

$$Q_{AC} = A^{\alpha} \sqrt[\alpha]{\ln C}. \tag{9}$$

It takes into account two basic parameters - accuracy and number of classes. A is the averaged accuracy defined by Eq. 10 and has values ranging from 0 to 1.0 [55]:

$$A = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}. \tag{10}$$

In the case of unequal class representations, it is recommended to use the version of balanced accuracy [56]. C is a natural number ≥ 2 specifying the number of classes. TP, TN, FP and FN are respectively True Positive, True Negative, False Positive and False Negative cases used for confusion matrix [57]. In addition, the formula takes into account the α power index, $\alpha > 1$, which is regarded as a nonlinear scaler. Then, the higher the Accurate Classification Quality Eq. 9 value, the better the classifier. The larger the index α, the more significant is A, and the less significant is C. Empirically, a value of $\alpha = 3$ has been determined, for which typical indicator values range from 0 to 1.5. Larger values can be obtained, but they are not realistic in practice. For example, $Q_{AC} = 2.0$ can be achieved with $A = 1.0$ and $C \approx 2981$ (e^8).

The second proposed index is **Accurate Classification-Time Quality**: Q_{ACT}, to indicate the accuracy considered for all classes of faults taking into account the fault detection time. It is represented by:

$$Q_{ACT} = A^{\alpha} \sqrt[\alpha]{\frac{\ln C}{\ln (1 + \tau^2)}}. \tag{11}$$

It may be used for comparing classifiers whose performance depends on the time taken to make a correct classification. For this, there is an additional parameter τ. It is a dimensionless value defined as:

$$\tau = \frac{T}{T_0}. \tag{12}$$

T is the time, expressed in milliseconds, required to perform all the operations necessary to obtain a classification and $T_0 = 1\ ms$ is the period corresponding to acquisition frequency 1 kHz. If the classification is performed cyclically at the smallest possible interval, T parameter is the time between two consecutive classifications.

The impact of the listed parameters on the values of the proposed indicators is presented in Table 8. The equations describing the two indicators were obtained after many rounds of refinement with the aim to emphasize the classification accuracy. The value that can be operationally considered as the boundary for Q_{AC} to provide good classification performance is 1. The quality metrics Eqs. 9 and 11 reach the same value for $\tau \approx 1.31\ ms$.

The two described indicators will find application for classifiers using both time and frequency domain signals.

5.2 Results

Hundreds of different fault classifiers were prepared in the form of artificial neural network models [46] on the data provided in PADRE. Table 9 summarizes the test results of selected models for both *Bebop 2* (B) and *Solo* (S) drones. The models are based on various combinations of accelerometer and gyroscope axis data and one model using acoustic data. The classifiers use both time and frequency domain data. The table presents only the basic parameters of the models tested, while the focus is on their performance and processing time. *ANN T [ms]* shows the inference speed i.e. the processing time in milliseconds by the artificial neural network classifier itself. Columns A, C and T indicate accuracy, number of classes and processing time in milliseconds, respectively. The values of Q_{AC} and Q_{ACT} indices for $\alpha = 3.0$ are presented. It is also marked which models meet the requirements of a real-time system whose total time of all operations is less than 2 ms ($f_s = 500$ Hz). The real-time models utilized a cyclic buffer, while the others used a common buffer.

All the tested classifiers shown in the table differ from each other by at least one parameter. The presented models are the ones which obtained the highest accuracy for a particular set of parameters. An interesting case is the first two models, which achieved similar accuracy for identical neural network and input data parameters. They differ only in the number of classes. The second model, although it achieved slightly lower accuracy, presents a higher Q_{AC} value. The last model

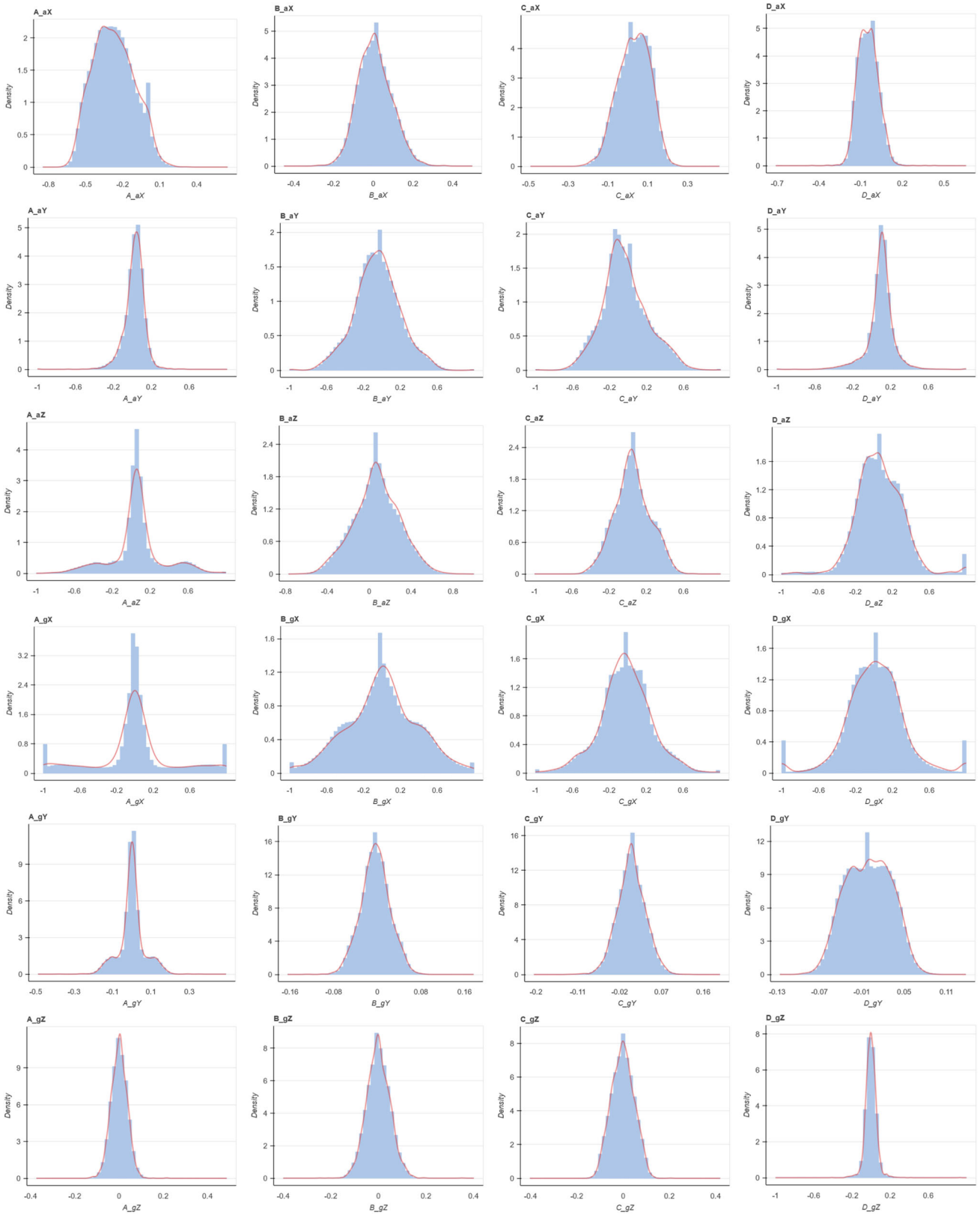

Fig. 7 Bebop 2 KDE plots

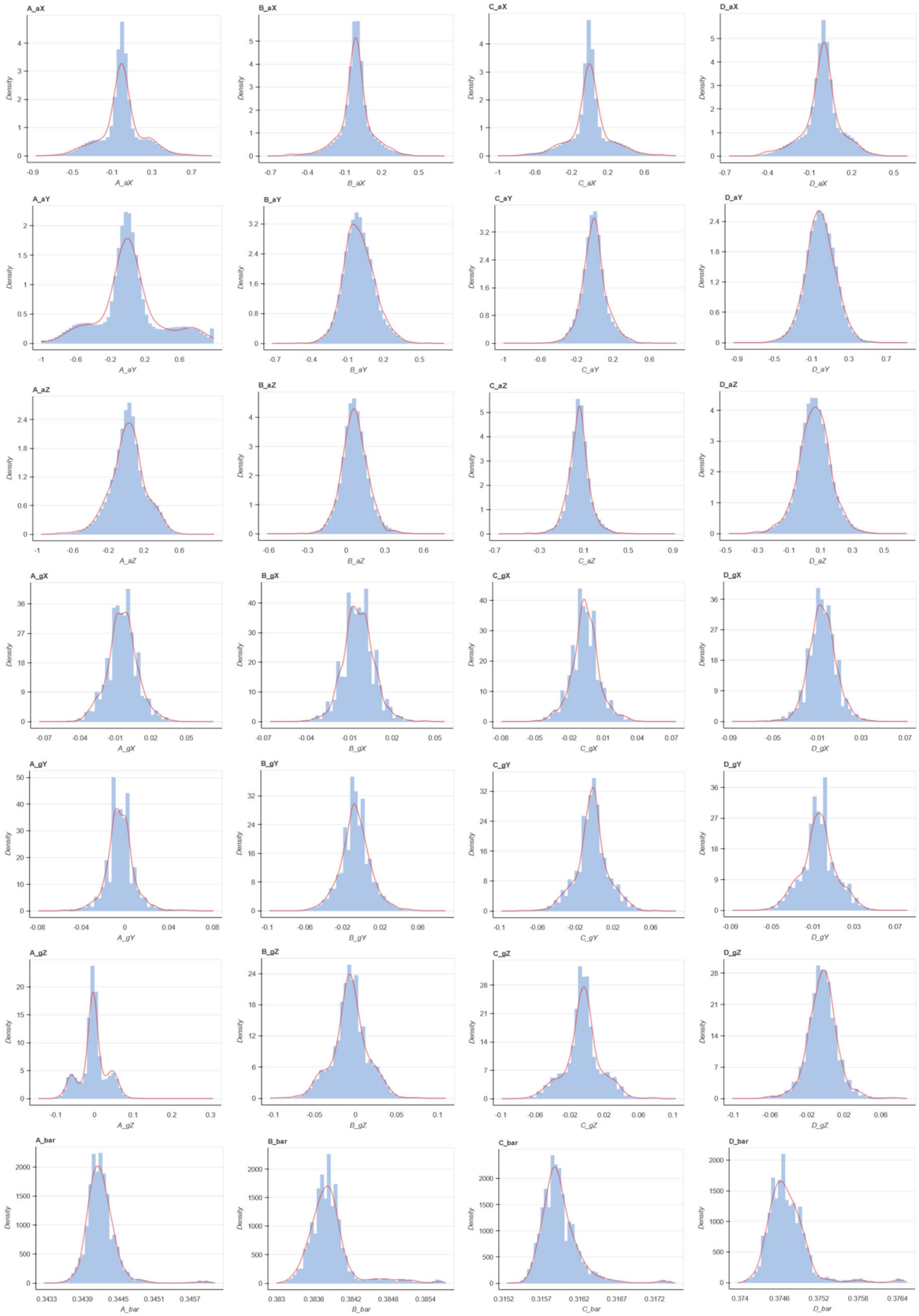

Fig. 8 Solo IMU KDE plots

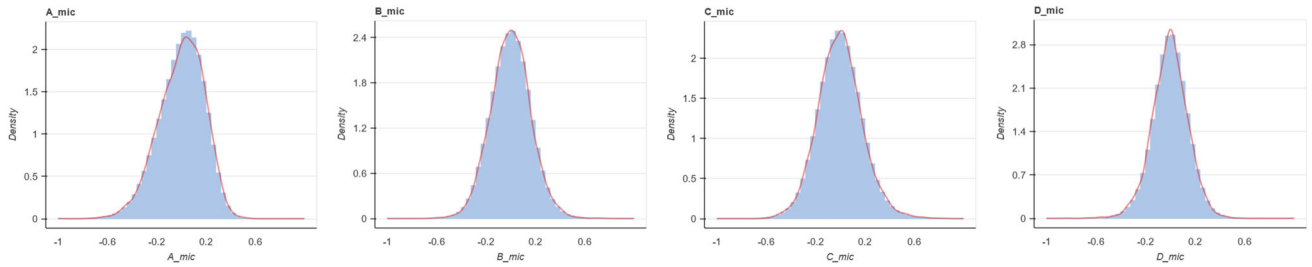

Fig. 9 Solo microphones KDE plots

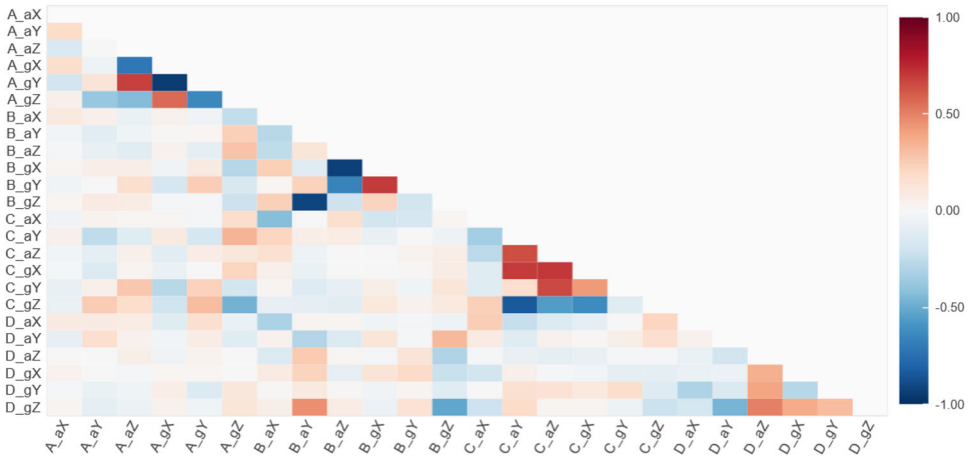

Fig. 10 Pearson correlations of individual sensor axes of the Bebop 2 drone

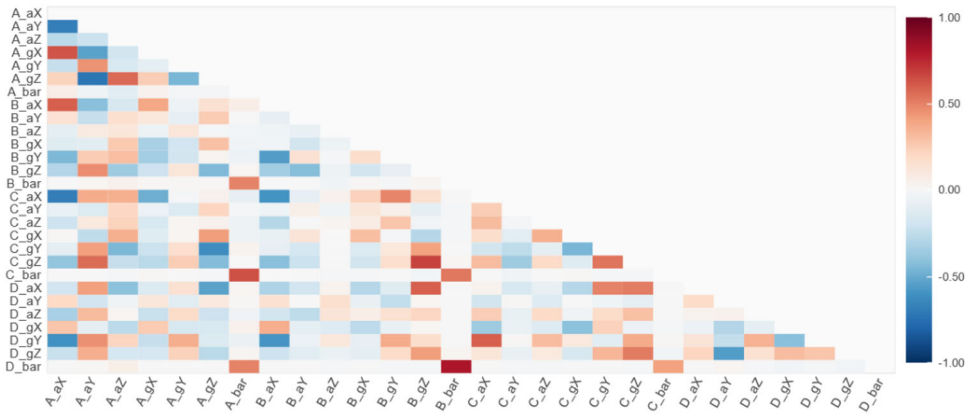

Fig. 11 Pearson correlations of individual IMU sensor axes of the Solo drone

Fig. 12 Pearson correlations of individual microphone readings of the Solo drone

Table 8 Values of proposed indicators for $\alpha = 3.0$ and various parameters A, C and τ

A	C	τ	Q_{AC}	Q_{ACT}	A	C	τ	Q_{AC}	Q_{ACT}	A	C	τ	Q_{AC}	Q_{ACT}
0.99	9	1	1.26	1.43	0.98	2	1	0.83	0.94	0.98	9	0.1	1.22	9.02
0.97	9	1	1.19	1.34	0.98	4	1	1.05	1.19	0.98	9	0.5	1.22	3.11
0.95	9	1	1.11	1.26	0.98	8	1	1.20	1.36	0.98	9	1	1.22	2.02
0.90	9	1	0.95	1.07	0.98	16	1	1.32	1.49	0.98	9	3	1.22	1.16
0.85	9	1	0.80	0.90	0.98	32	1	1.42	1.61	0.98	9	10	1.22	0.83
0.80	9	1	0.67	0.75	0.98	64	1	1.51	1.71	0.98	9	100	1.22	0.62
0.70	9	1	0.45	0.50	0.98	128	1	1.59	1.80	0.98	9	1000	1.22	0.53
0.65	9	1	0.36	0.40	0.98	256	1	1.67	1.88	0.98	9	10000	1.22	0.48

for *Bebop 2* achieved the lowest accuracy value presented, but due to the number of classes equal to 20 and the speed of data processing, it showed relatively high values of Q_{AC} and Q_{ACT} metrics. In the case of *Solo*, all models considered 9 classes, so the descending order of this part of the table applies to both accuracy and Q_{AC}. Acoustic signals showed detection ability for frequency data. The highest performance was achieved by models processing 1024 samples from each channel, which, due to processing time and acquisition rate ($f_s = 48$ kHz), makes real-time detection impossible. On the other hand, they still show the ability to correctly detect propeller faults in normal classifier operating mode.

All the models presented have a Q_{AC} index higher than 1.0, the models that process the data in real-time also achieved a Q_{ACT} index of more than 1.0. This shows the possibility of using the measurements from our repository for the tasks of propeller fault detection (B, S), determination of their location (B, S) and defect severity estimation (S).

Figure 13 shows plots of the loss function and accuracy of an exemplary classifier. Both values for training and validation data are shown. Training was carried out for 5000 epochs for three-axis accelerometer data in the time domain for a measurement window length of 32 samples. A convolutional network with a filter number of 3, a ker-

Table 9 Classification test results for $\alpha = 3.0$ and $T_0 = 2\ ms$

UAV	Accel axes	Gyro axes	Mic axes	Signal domain	Axes no.	Samples per axis	ANN T [ms]	A	C	T [ms]	Q_{AC}	Q_{ACT}	Real time
B	xyz	xyz		freq	24	128	3.21	0.9958	5	281.34	1.16	0.54	
B	xyz	xyz		freq	24	128	3.21	0.9919	9	280.08	1.27	0.59	
B		xyz		time	12	8	0.85	0.9899	9	1.15	1.26	1.91	✓
B		xyz		time	12	2	0.20	0.9888	5	0.47	1.13	3.00	✓
B	xyz	xyz		freq	24	64	2.54	0.9880	20	140.79	1.39	0.68	
B		xyz		time	12	8	0.88	0.9817	20	1.17	1.36	2.05	✓
B		xyz		time	12	4	0.54	0.9732	20	0.82	1.33	2.47	✓
B	z	x		freq	8	128	1.21	0.9703	20	257.43	1.32	0.62	
B	xyz	xyz		freq	24	16	1.30	0.9659	20	35.77	1.30	0.72	
B	z	x		freq	8	32	0.84	0.9625	20	1.95	1.29	1.47	✓
B	xyz	xyz		freq	24	8	0.76	0.9593	20	18.01	1.27	0.78	
B	z	x		freq	8	32	0.38	0.9510	20	1.50	1.24	1.62	✓
S	xyz			time	12	32	4.77	0.9950	9	66.82	1.28	0.67	
S	xyz			time	12	32	1.17	0.9934	9	1.52	1.27	1.66	✓
S	xyz	xyz		freq	24	128	19.12	0.9934	9	291.09	1.27	0.59	
S		xyz		time	12	32	2.75	0.9915	9	64.83	1.27	0.66	
S		xyz		time	12	16	1.47	0.9894	9	1.77	1.26	1.51	✓
S	xyz			time	12	16	0.70	0.9873	9	1.00	1.25	2.06	✓
S	y			freq	4	64	0.51	0.9868	9	127.86	1.25	0.62	
S	y			freq	4	16	0.21	0.9744	9	0.77	1.20	2.33	✓
S			m	freq	4	1024	0.71	0.9743	9	30.74	1.20	0.68	

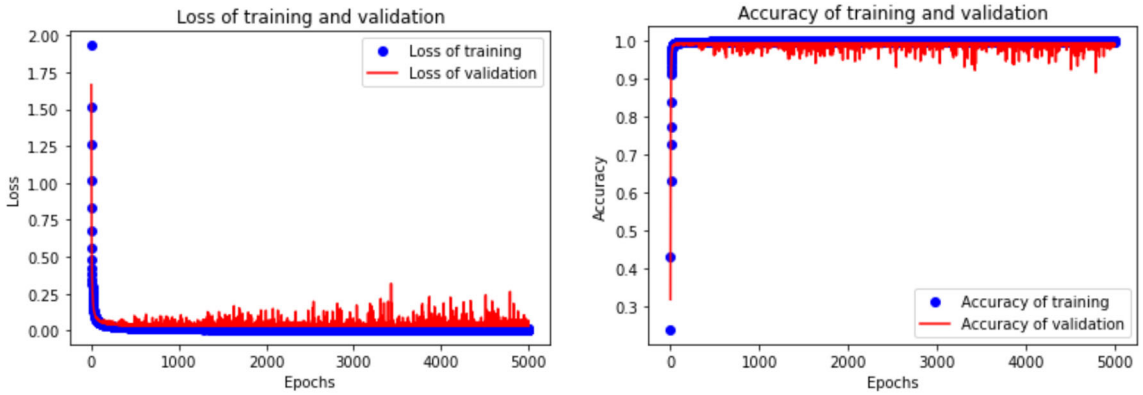

Fig. 13 The convergence curves of the loss function and accuracy during the training of the best *Solo* drone classifier

nel size of 300 and a batch size of 512 was used. Activation function in the form of `softmax`, `rmsprop` optimizer, `categorical_crossentropy` loss function and `acc` metrics were used. The highest test accuracy of 0.9950 and test loss of 0.0273 were achieved by the 1166th epoch model. Training accuracy for this epoch was 0.9996, and training loss was 0.0088. At the validation stage, the values were 0.9928 and 0.0275, respectively.

5.3 Discussion

The PADRE database presented here addresses the growing importance of UAV flight safety research. Although other UAV data repositories are emerging, most of them do not consider elements of the actuator in fault studies. Those databases that are currently available usually contain very large files with all measurements obtainable from the flight controller. Only a small portion of these are suitable for FDI tasks. PADRE was prepared with this gap in mind, so its contents are suitable for direct use in multirotor propeller fault investigation. Of all the data collected, high classification performance was demonstrated by measurements from accelerometers, gyroscopes and microphones. Only data from barometers have so far failed to be used effectively for these purposes. The probable reason for this could be the lack of a reference point in the form of a measurement from an additional barometer located in the center of the fuselage. This problem is planned to be solved by designing the next version of the data acquisition and fault detection system.

Another contribution of this paper is the development of effective metrics to directly compare different classifiers. Although accuracy and similar common metrics usually play the most important role, we should not forget the key factors deciding the fault detection performance. First and foremost, the number of classes that must be included in the classification task directly affects the computational complexity and thus the effectiveness of the classification. Our proposed Q_{AC} solves this problem and allows comparing classifiers that differ arbitrarily in the complexity of the problem being solved. The Q_{ACT} index, on the other hand, allows comparing models operating in systems where detection time is crucial. Especially for unmanned aerial vehicles, the speed of anomaly detection determines mission safety. Hence, a classifier that needs to quickly recognize the occurrence of a fault has limited time resources available to perform calculations. Therefore, real-time systems need to use both computationally efficient hardware and reliable calculations. Consideration of computation time as an important parameter of the system operating on board the UAV should force researchers to look for feasible solutions. If the system is to be applied in real-time under realistic conditions, it must meet processing time requirements. To address that, this paper focuses on parameters that are of colossal importance in the effectiveness of assigning a particular fault to the appropriate class. The first parameter is the number of classes. It is obvious that a larger number of classes presents a more complex problem than binary classification. The new Q_{AC} indicator is proposed to consider that parameter. The second parameter that has not been addressed so far is the classification processing time. When flying a drone, the timing of anomaly detection can determine whether the vehicle can safely land or crash. As confirmed through a series of experiments, our proposed Q_{ACT} indicator takes this parameter into account in an explicit and effective manner.

6 Conclusion

This article has presented the comprehensive development of a versatile repository for drone propeller fault classification, the PADRE measurement database. The repository currently contains measurements from sensors placed on the arms of

2 quadrotor models with various types and combinations of propeller failures. On the *Parrot Bebop 2* drone, flights were conducted with 2 types of propeller faults in 20 different configurations. On the *3DR Solo* drone, experiments were performed with 2 levels of propeller fragment loss in 9 different configurations. It includes CSV files with different data representations that can serve multiple purposes. The main objective of creating this publicly available database is to address the lack of a reliable and useful database for fault detection and classification regarding drone propellers. Given the very time-consuming process of collecting measurements for investigating a data-based fault detector or classifier, the existence of such a database seems justified. Researchers, instead of spending time on lengthy experiments and pre-processing the collected data, can immediately start verifying their method. The quality of the collected data has been confirmed through a number of fault classification experiments carried out in the normal mode and in the real-time regime, using time and frequency domain data and performed both offline (previously collected measurements were tested) and online during flight in real world conditions.

The data repository was also evaluated using various statistical criteria along with our two newly-proposed indicators, the Accurate Classification Quality and the Accurate Classification-Time Quality metrics. To achieve high performance in real-time, fault classifiers have to consider some important parameters. One such parameter is the number of classes. It is obvious that a larger number of classes presents a more complex problem than binary classification. The new Q_{AC} indicator allows comparing classifiers with different numbers of classes considered. Another parameter that has not been addressed so far is the classification processing time. Some classifiers can run offline on stationary computer as long as they need to achieve the highest possible classification accuracy. However, for operational UAVs, fault detection processing time is crucial. The second proposed Q_{ACT} indicator also takes this parameter into account, allowing to compare classifiers with different operating times. Extensive experiments and results obtained have indicated the merits of the proposed repository, PADRE, for fault identification in drone propellers.

In summary, the primary contributions of the present work and research are:

- preparation of an open repository for use by scientists researching drone fault detection and classification,
- testing the effectiveness of fault classification using data from the repository,
- development of two new functional quality metrics for classifiers that take into account the actual requirements of fault classification in flying vehicles.

Supplementary information

A video of a sample flight of the Parrot Bebop 2 drone in the AeroLab is available at http://uav.put.poznan.pl/archives/763. A flight of the 3DR Solo drone in real-world conditions is shown at http://uav.put.poznan.pl/archives/788.

Author Contributions Study concept: Wojciech Giernacki. System design and implementation: Radosław Puchalski. Software preparation: Radosław Puchalski, Huynh Anh Duy Nguyen. Conducting experiments: Radosław Puchalski, Lanh Van Nguyen. Article content: Quang Ha, Wojciech Giernacki. The first draft of the manuscript: Radosław Puchalski, Wojciech Giernacki. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This work has been supported by the Polish National Agency for Academic Exchange (NAWA) under the STER programme, Towards Internationalization of Poznan University of Technology Doctoral School (2022-2024). APC was funded by Poznan University of Technology grant no. 0214/SBAD/0241.

Availability of data and materials The data are available at https://github.com/AeroLabPUT/UAV_measurement_data

Code availability Not applicable

Declarations

Competing interests The authors declare no competing interests.

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

References

1. Utsav, A., Abhishek, A., Suraj, P., Badhai, R.K.: An IoT based UAV network for military applications. In: 2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 122–125 (2021). https://doi.org/10.1109/WiSPNET51692.2021.9419470

2. Amarasingam, N., Ashan Salgadoe, A.S., Powell, K., Gonzalez, L.F., Natarajan, S.: A review of UAV platforms, sensors, and applications for monitoring of sugarcane crops. Remote Sens. Appl.: Soc. Environ. **26**, 100712 (2022). https://doi.org/10.1016/j.rsase.2022.100712
3. Yang, Z., Yu, X., Dedman, S., Rosso, M., Zhu, J., Yang, J., Xia, Y., Tian, Y., Zhang, G., Wang, J.: UAV remote sensing applications in marine monitoring: Knowledge visualization and review. Sci. Total Environ. **838**, 155939 (2022). https://doi.org/10.1016/j.scitotenv.2022.155939
4. Hoang, V.T., Phung, M.D., Dinh, T.H., Ha, Q.P.: System architecture for real-time surface inspection using multiple UAVs. IEEE Syst. J. **14**(2), 2925–2936 (2020). https://doi.org/10.1109/JSYST.2019.2922290
5. Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F.: Fault-tolerant cooperative navigation of networked UAV swarms for forest fire monitoring. Aerosp. Sci. Technol. **123**, 107494 (2022). https://doi.org/10.1016/j.ast.2022.107494
6. Khan, S.I., Qadir, Z., Munawar, H.S., Nayak, S.R., Budati, A.K., Verma, K.D., Prakash, D.: UAVs path planning architecture for effective medical emergency response in future networks. Phys. Commun. **47**, 101337 (2021). https://doi.org/10.1016/j.phycom.2021.101337
7. Shadiev, R., Yi, S.: A systematic review of UAV applications to education. Interact Learn. Environ. 1–30 (2022). https://doi.org/10.1080/10494820.2022.2028858
8. Li, Y., Liu, M., Jiang, D.: Application of unmanned aerial vehicles in logistics: A literature review. Sustainability. **14**(21) (2022). https://doi.org/10.3390/su142114473
9. Pasandideh, F., Costa, J.P.J., Kunst, R., Islam, N., Hardjawana, W., Freitas, E.: A review of flying ad hoc networks: Key characteristics, applications, and wireless technologies. Remote Sens. **14**(18) (2022). https://doi.org/10.3390/rs14184459
10. Qiao, J., Liu, Z., Zhang, Y.: Gain scheduling based pid control approaches for path tracking and fault tolerant control of a quadrotor UAV. (2018). https://doi.org/10.18178/ijmerr.7.4.401-408
11. Zheng, Qinghe, Tian, Xinyu, Zhiguo, Yu., Wang, Hongjun, Elhanashi, Abdussalam, Saponara, Sergio: DL-PR: Generalized automatic modulation classification method based on deep learning with priori regularization. Eng. Appl. Artif. Intell. **122**, 106082 (2023). https://doi.org/10.1016/j.engappai.2023.106082
12. Puchalski, R., Giernacki, W.: UAV fault detection methods, state-of-the-art. Drones. **6**(11) (2022). https://doi.org/10.3390/drones6110330
13. Schijndel, B.A.S., Sun, S., Visser, C.C.: Fast loss of effectiveness detection on a quadrotor using onboard sensors and a Kalman estimation approach. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1–8 (2023). https://doi.org/10.1109/ICUAS57906.2023.10156464
14. Al-Haddad, L.A., Jaber, A.A.: An intelligent fault diagnosis approach for multirotor UAVs based on deep neural network of multi-resolution transform features. Drones. **7**(2) (2023). https://doi.org/10.3390/drones7020082
15. Ashe, A.K., Goli, S., Kandath, H., Gangadharan, D.: Multivariate data analysis for motor failure detection and isolation in a multicopter UAV using real-flight attitude signals. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 9–16 (2023). https://doi.org/10.1109/ICUAS57906.2023.10155856
16. Wang, B., Liu, D., Peng, Y., Peng, X.: Multivariate regression-based fault detection and recovery of UAV flight data. IEEE Trans. Instrum. Meas. **69**(6), 3527–3537 (2019). https://doi.org/10.1109/TIM.2019.2935576
17. Wang, B., Peng, X., Jiang, M., Liu, D.: Real-time fault detection for UAV based on model acceleration engine. IEEE Trans. Instrum. Meas. **69**(12), 9505–9516 (2020). https://doi.org/10.1109/TIM.2020.3001659
18. Puchalski, R., Bondyra, A., Giernacki, W., Zhang, Y.: Actuator fault detection and isolation system for multirotor unmanned aerial vehicles. In: 2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 364–369 (2022). https://doi.org/10.1109/MMAR55195.2022.9874283
19. Sadhu, V., Zonouz, S., Pompili, D.: On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 5255–5261 (2020). https://doi.org/10.48550/arXiv.2005.00336 . IEEE
20. D'Amato, E., Nardi, V.A., Notaro, I., Scordamaglia, V.: A particle filtering approach for fault detection and isolation of UAV IMU sensors: Design, implementation and sensitivity analysis. Sensors. **21**(9) (2021). https://doi.org/10.3390/s21093066
21. Saied, M., Tabikh, A.R., Francis, C., Hamadi, H., Lussier, B.: An informational approach for fault tolerant data fusion applied to a UAV's attitude, altitude, and position estimation. IEEE Sensors J. **21**(24), 27766–27778 (2021). https://doi.org/10.1109/JSEN.2021.3124731
22. Altinors, A., Yol, F., Yaman, O.: A sound based method for fault detection with statistical feature extraction in UAV motors. Appl. Acoust. **183**, 108325 (2021). https://doi.org/10.1016/j.apacoust.2021.108325
23. Gomez, M.S., Koschlik, A.-K., Arts, E., Raddatz, F.: Non-destructive evaluation of the condition of a UAV's propellers by means of acoustics. In: NDE 4.0, Predictive Maintenance, and Communication and Energy Systems in a Globally Networked World, vol. 12049, pp. 22–30 (2022). https://doi.org/10.1117/12.2612770
24. Kołodziejczak, M., Puchalski, R., Bondyra, A., Sladic, S., Giernacki, W.: Toward lightweight acoustic fault detection and identification of UAV rotors. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 990–997 (2023). https://doi.org/10.1109/ICUAS57906.2023.10156624
25. Fourlas, G.K., Karras, G.C.: A survey on fault diagnosis methods for UAVs. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 394–403 (2021). https://doi.org/10.1109/ICUAS51884.2021.9476733
26. Puchalski, R., Kołodziejczak, M., Bondyra, A., Rao, J., Giernacki, W.: PADRE - propeller anomaly data repository for UAVs various rotor fault configurations. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 982–989 (2023). https://doi.org/10.1109/ICUAS57906.2023.10156238
27. Johnson, P., Ricker, B., Harrison, S.: Volunteered drone imagery: Challenges and constraints to the development of an open shared image repository. In: Hawaii International Conference on System Sciences (2017). https://doi.org/10.24251/HICSS.2017.242
28. Antonini, A., Guerra, W., Murali, V., Sayre-McCord, T., Karaman, S.: The blackbird UAV dataset. Int. J. Robotics Res. **39**(10–11), 1346–1364 (2020). https://doi.org/10.48550/arXiv.1810.01987
29. Mylonas, N., Malounas, I., Mouseti, S., Vali, E., Espejo-Garcia, B., Fountas, S.: Eden library: A long-term database for storing agricultural multi-sensor datasets from UAV and proximal platforms. Smart Agricultural Technology. **2**(100028) (2022). https://doi.org/10.1016/j.atech.2021.100028
30. Gururajan, S., Mitchell, K., Ebel, W.: Flights of a multirotor UAS with structural faults: Failures on composite propeller(s). Data. **4**(3) (2019). https://doi.org/10.3390/data4030128
31. Keipour, A., Mousaei, M., Scherer, S.: ALFA: A dataset for UAV fault and anomaly detection. Int. J. Robotics Res. **40**(2–3), 515–520 (2021). https://doi.org/10.1177/0278364920966642
32. Baldini, A., D'Alleva, L., Felicetti, R., Ferracuti, F., Freddi, A., Monteriù, A.: UAV-FD: a dataset for actuator fault detection in multirotor drones. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 998–1004 (2023). https://doi.org/10.1109/ICUAS57906.2023.10156060

33. Tsai, C.-P., Li, W.-C.: A micromechanical frequency controlled pulse density modulator. In: 2022 IEEE 35th International Conference on Micro Electro Mechanical Systems Conference (MEMS), pp. 204–207 (2022). https://doi.org/10.1109/MEMS51670.2022.9699549
34. Parrot: Bebop-pro 3D modeling, All-in-one drone solution for 3D modeling. Accessed 24 Apr 2024 (2024). https://www.parrot.com
35. 3DR: Solo Operation Manual. Accessed 24 Apr 2024 (2015). https://www.wwu.edu/faculty/wallin/esci497_uas/readings/3dr_solo/Solo_FAA_6_22_operations_manual.pdf
36. InvenSense: MPU-6500 Product Specification Revision 1.3. Accessed: 30 Oct 2023 (2020). http://invensense.wpenginepowered.com/wp-content/uploads/2020/06/PS-MPU-6500A-01-v1.3.pdf
37. InvenSense: MPU-9250 Product Specification Revision 1.1. Accessed: 28 Sept 2023 (2016). https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf
38. Bosch: BMP280 – Data sheet. Accessed: 28 Sept 2023 (2021). https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf
39. STMicroelectronics: MP34DT01-M MEMS audio sensor omnidirectional digital microphone. Accessed 28 Sept 2023 (2014). https://www.st.com/en/audio-ics/mp34dt01-m.html
40. Avola, Danilo and Cinque, Luigi and Foresti, Gian Luca and Lanzino, Romeo and Marini, Marco Raoul and Mecca, Alessio and Scarcello, Francesco: A novel transformer-based IMU self-calibration approach through on-board RGB camera for UAV flight stabilization. Sensors. **23**(5) (2023). https://doi.org/10.3390/s23052655
41. Black, H.S., Edson, J.: Pulse code modulation. Trans. Am. Inst. Electr. Eng. **66**(1), 895–899 (1947). https://doi.org/10.1109/T-AIEE.1947.5059525
42. Oshana, R.: Overview of digital signal processing algorithms. In: Oshana, R. (ed.) DSP Software Development Techniques for Embedded and Real-Time Systems. Embedded Technology, pp. 59–121. Newnes, Burlington (2006). https://doi.org/10.1016/B978-075067759-2/50006-5
43. Jothilakshmi, S., Gudivada, V.N.: Large scale data enabled evolution of spoken language research and applications. In: Gudivada, V.N., Raghavan, V.V., Govindaraju, V., Rao, C.R. (eds.) Cognitive Computing: Theory and Applications. Handbook of Statistics, vol. 35, pp. 301–340. Elsevier, Oxford (2016). https://doi.org/10.1016/bs.host.2016.07.005
44. STMicroelectronics: UM2372 STM32Cube PDM2PCM software library for the STM32F4/F7/H7 Series. Accessed: 22 Oct 2023 (2018). https://www.st.com/content/ccc/resource/technical/document/user_manual/group1/6b/43/5e/35/2d/86/49/0d/DM00482421/files/DM00482421.pdf/jcr:content/translations/en.DM00482421.pdf
45. Dabrowski, A., Marciniak, T.: Audio signal processing. In: Oklobdzija, V.G. (ed.) Digital Systems and Applications. CRC Press, Boca Raton (2017). https://doi.org/10.1201/9780849386206
46. Puchalski, R., Giernacki, W., Ha, Q.: Real-time UAV fault detection and classification using measurement data from the PADRE database. In: The 2024 IEEE/SICE International Symposium on System Integration (SII 2024), Ha Long, Vietnam (8-11 JAN 2024). https://doi.org/10.1109/SII58957.2024.10417427
47. Al-Haddad, L.A., Giernacki, W., Shandookh, A.A., Jaber, A.A., Puchalski, R.: Vibration signal processing for multirotor UAVs fault diagnosis: Filtering or multiresolution analysis? Eksploatacja i Niezawodność - Maintenance and Reliability. **26**(1) (2024). https://doi.org/10.17531/ein/176318
48. Taylor, F.J.: Digital signal processing. In: Oklobdzija, V.G. (ed.) Digital Systems and Applications. CRC Press, Boca Raton (2017). https://doi.org/10.1201/9780849386206
49. Wied, D., Weißbach, R.: Consistency of the kernel density estimator: a survey. Stat. Pap. **53**, 1–21 (2012). https://doi.org/10.1007/s00362-010-0338-1
50. Sedgwick, P.: Pearson's correlation coefficient. Bmj. **345** (2012). https://doi.org/10.1136/bmj.e4483
51. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In: Losada, D.E., Fernández-Luna, J.M. (eds.) Advances in Information Retrieval, pp. 345–359. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31865-1_25
52. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Trans. Softw. Eng. **34**(4), 485–496 (2008). https://doi.org/10.1109/TSE.2008.35
53. Vieira, S.M., Kaymak, U., Sousa, J.M.C.: Cohen's kappa coefficient as a performance measure for feature selection. In: International Conference on Fuzzy Systems, pp. 1–8 (2010). IEEE Trans Softw Eng. https://doi.org/10.1109/FUZZY.2010.5584447
54. Chicco, D., Warrens, M.J., Jurman, G.: The Matthews correlation coefficient (MCC) is more informative than Cohen's kappa and Brier score in binary classification assessment. IEEE Access. **9**, 78368–78381 (2021). https://doi.org/10.1109/ACCESS.2021.3084050
55. Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. Int. J Data Min. Knowl. Manag. Process. **5**(2), 01–11 (2015). https://doi.org/10.5121/ijdkp.2015.5201
56. Grandini, M., Bagli, E., Visani, G.: Metrics for multi-class classification: an overview. ArXiv. **abs/2008.05756** (2020). ArXiv:2008.05756
57. Makhtar, M., Neagu, D.C., Ridley, M.J.: Comparing multi-class classifiers: On the similarity of confusion matrices for predictive toxicology applications. In: Yin, H., Wang, W., Rayward-Smith, V. (eds.) Intelligent Data Engineering and Automated Learning - IDEAL 2011, pp. 252–261. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23878-9_31

Radosław Puchalski graduated with a B.E. degree from the Faculty of Computer Science, Poznan University of Technology, Poland, in 2018. Subsequently, he received the M.Sc. degree in automatic control and robotics from the Faculty of Control, Robotics and Electrical Engineering in 2020. He is currently a Ph.D. student at Doctoral School of Poznan University of Technology. He conducts research in the disciplines of automation, electronics, electrical engineering and space technologies. His main research activity involves fault detection of unmanned aerial vehicles. His scientific interests focus on digital signal processing, artificial neural networks and the application of embedded systems in edge processing. He has carried out his research at Imperial College London and University of Technology Sydney.

Quang Ha received the B.E. degree from Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in electrical engineering, and the Ph.D. degrees from Moscow Power Engineering Institute, Moscow, Russia, in complex systems and control, and the University of Tasmania, Australia, in intelligent systems. He is currently an Associate Professor with the Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. His research interests include automation, robotics, and control systems. He has been on the Board of Directors of the International Association for Automation and Robotics in Construction (IAARC) since 2007 and a Senior Member of the Institute for Electrical and Electronic Engineering (IEEE) since 2013. He was an Associate Editor of the IEEE Transactions on Automation Science and Engineering (2009-2013), and currently an Associate Editor for Automation in Construction, Robotica, Frontiers in Robotics and AI, and a Section Editor- Systems and Control Engineering for Electronics. Dr Ha was the recipient of 18 R&D awards and many best paper awards from the IEEE, IAARC and Engineers Australia, including the 2015 Sir George Julius Medal and most recently, the Elsevier 2023 Applied Soft Computing Best Paper Award.

Wojciech Giernacki received the Ph.D. degree in control engineering and robotics from the Poznan University of Technology in 2011 and D.Sc. in 2019. He founded and heads the PUT AeroLab and Unmanned Aerial Vehicles Research Group, as well as the Division of Control and Optimization at the Institute of Robotics and Machine Intelligence. His scientific interests are focused around the issues of UAVs, especially robust and adaptive control, optimization techniques, as well as data fusion from sensors.

Huynh A.D. Nguyen (Member, IEEE, IOTAA) received his B.E. degree in Mechatronics Engineering from Can Tho University, Vietnam, in 2009, and his M.Sc. degree in Mechatronics Engineering from the University of Siegen, Germany, in 2015. He completed his Ph.D. with the Faculty of Engineering and Information Technology at the University of Technology Sydney (UTS), Australia, in 2024. His research interests include IoT technology, data processing and analysis, machine learning, and deep learning for time series.

Lanh V. Nguyen (Member, IEEE) received his B.E. degree in Electrical Engineering from the Thai Nguyen University of Technology (TNUT), Vietnam, in 2011 and his Master of Control System Engineering from the HAN University of Applied Science, the Netherlands, in 2018. He completed his Ph.D. with the Faculty of Engineering and Information Technology at the University of Technology Sydney (UTS), Australia, in 2024. His research interests include automation, robotics, control and navigation for unmanned vehicles, game theory, and evolutionary algorithms.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com