2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Towards a Reliable and Lightweight Onboard Fault Detection in Autonomous Unmanned Aerial Vehicles

Sai Srinadhu Katta[1] and Eduardo Kugler Viegas[2]

Abstract— **This paper proposes a new model for onboard physical fault detection on autonomous unmanned aerial vehicles (UAV) through machine learning (ML) techniques. The proposal performs the detection task with high accuracies and minimal processing requirements while signaling an unreliable ML model to the operator, implemented in two main phases. First, a wrapper-based feature selection is performed to decrease the feature extraction computational costs, coped with a classification assessment technique to identify ML model unreliability. Second, physical UAV faults are signaled through a multi-view rationale that evaluates a variety of UAV sensors while triggering alerts based on a sliding window scheme. Experiments performed on a real quadcopter UAV with a broken propeller use case shows the proposal's feasibility. Our model can decrease the false-positive rates up to only 0.4%, while also decreasing the computational costs by at least 43% when compared to traditional techniques. Notwithstanding, it can identify ML model unreliability, signaling the UAV operator when model fine-tuning is needed.**

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161183

I. Introduction

Autonomous unmanned aerial vehicles (UAV) are prone to physical failures without the operator's supervision, which may negatively affect their security and the safety of the environment wherein they operate [1]. For example, UAVs may be hit by birds or even collide with undetected nearby objects, which, apart from introducing physical UAV faults (e.g. broken propellers) put the environment at risk [2]. Therefore, ensuring a reliable onboard and real-time detection of UAV physical faults is a must.

Machine learning (ML) has reached a strong relation with recently proposed UAV fault detection approaches due to their promising reported results [3]. In such a case, an ML model is built according to the behavior available in a training dataset, while its performance is evaluated through a test dataset. Although current ML-based schemes can provide highly accurate UAV fault detection accuracies, these techniques often imply in unfeasible associated computational costs [4].

Modern UAVs are typically resource-constrained cyber-physical systems (CPS) that must execute a variety of tasks in real-time in addition to the to-be-deployed fault detection scheme, such as object detection [5], object tracking [6], and state estimation for example [7]. Surprisingly, current proposals in their vast majority overlook their computational costs, resorting to resource-demanding deep learning architectures to increase detection accuracies [2].

Over the last years, UAV physical faults have been detected through the evaluation of the inertial measurement unit (IMU) data [2], while recent works have shown the feasibility of using other UAV sensors, e.g. using an attached microphone [8]. The data generated by the corresponding sensor is continuously collected and used as input to the detection module, which first extracts a predefined set of features before applying the designed ML model, e.g. extraction of a set of statistical values over the last 1 second window of the accelerometer data. Thus, researchers must also provide a lightweight feature extraction procedure apart from using a lightweight ML model.

Fault detection in autonomous UAVs must be reliably executed without the operator's supervision. The reliability of ML-based detection schemes relies upon using a realistic training dataset, a requirement not easily feasible considering the unforeseen situations of real-world UAV environments [9]. In practice, untrained situations cause the ML model error rate to increase, becoming primarily false positives, which motivates operators to disable designed fault detection techniques. Surprisingly, prior works assume that the accuracies measured during the test phase will be evidenced in real-world usage, considering the classification reliability aspects an orthogonal challenge [2], [8].

This paper proposes a new reliable onboard physical fault detection model for autonomous UAVs, implemented in four stages. First, a multi-objective feature selection technique is used to decrease the number of used features for lightweight feature extraction without degrading the system's accuracy. Second, UAV fault detection is performed through classification with a reject option, ensuring that our system rejects new and potential misclassifications. Third, UAV fault detection is performed through a multi-view procedure in a late fusion setting, using several UAV sensors for detection. Finally, the decision engine is implemented through a sliding window scheme, signaling UAV faults according to a predefined number of alerts. As a result, our scheme can significantly decrease the computational costs associated with feature extraction, increase detection accuracy, and adequately signal ML model unreliability as time passes.

In summary, our paper's main contributions are:

- A new publicly available dataset[3] with over 2 hours of flight data collected from a real autonomous Holybro x500 UAV. The dataset depicts a defective quadcopter

[1]Sai is with Secure Systems Research Center (SSRC) at Technology Innovation Institute (TII), United Arab Emirates, Abu Dhabi sai@ssrc.tii.ae

[2]Eduardo is with Secure Systems Research Center (SSRC) at Technology Innovation Institute (TII), United Arab Emirates, Abu Dhabi, and Graduate Program in Informatics (PPGIa) Pontifícia Universidade Catolica do Paraná (PUCPR), Brazil eduardo@ssrc.tii.ae

[3]https://github.com/tiiuae/UAV-Fault-Dataset

UAV flying with a variable number of broken propellers.

- A new reliable and lightweight UAV fault detection scheme with less processing demands (up to 94% less) and better false-positive rates (up to only 0.4%);
- A prototype of our model implemented over ROS2 with minimal processing footprint ($\approx$ 1.1ms/event);

II. Preliminaries

Over the last years, several works have been proposed for detecting UAV physical faults [2], [8], typically making use of simulation environments due to the challenges related to building a realistic UAV dataset. In such a case, physical faults are injected unrealistically, unable to depict a real defective UAV, such as those caused by a broken/deformed propeller or even frame deformation [10].

In general, UAV faults are identified through data-driven-oriented approaches, either using machine learning (ML) or deep learning (DL) techniques [11], [12]. A common yet effective strategy evaluates the UAV IMU's data using the gyroscope, which measures the UAV rotational velocity, and the accelerometer, which measures the UAV gravity-compensated linear acceleration. IMU-based techniques detect faults based on the UAV motion [12]. Due to the high data generation frequency, IMU-related data is summarized in time intervals through statistical-based features [13], e.g. standard deviation of the X-axis acceleration values collected over the last 1 second interval. These hand-designed features are used as input to an ML or DL model, where usually most of the research effort is spent on finding the classification scheme that provides the best accuracy. Besides IMU-oriented techniques, additional UAV sensors have also provided high detection accuracies in recent works, such as audio-based detection of faults [14]. The mentioned approach mounts a microphone on the defective UAV and use it to capture the emitted UAV audio signals. The generated data is then converted to a to-be-classified format (e.g. spectogram [15] or Mel-Frequency Cepstral Coefficients (MFCCs) [16]), before being used as input to the classification scheme.

III. Problem Statement

This section further investigates the aspects that make real-time detection of physical faults in autonomous UAVs challenging. We first introduce the dataset with real UAV faults built in our work, then we evaluate how approaches used in the literature perform over it.

A. A Realistic UAV Physical Fault Dataset

To build a more comprehensive dataset, our work presents a new dataset, namely the *Realistic UAV Physical Fault Dataset*. The built dataset was collected in a real setting considering a defective UAV with a broken propeller use case. To achieve such a goal, we collect the data generated by a Holybro X500 UAV equipped with an Intel UP Xtreme i7 8665UE mission computer, with a mounted Seeed Studio ReSpeaker Mic Array for audio data collection purposes, as shown in Figure 1. The UAV executes autonomous flights as

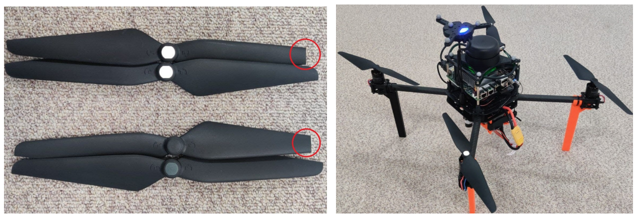

(a) UAV Propellers (b) Holybro X500 UAV

Fig. 1: UAV configuration used in our testbed. The broken region of the defective UAV propeller is highlighted with a red circle.

managed by the PX4 Autopilot. Each flight flies the UAV in randomly defined way-points following an *eight*-shaped setting for $\approx$ 2 minutes. The UAV data is continuously collected throughout the mission execution via the Robot Operating System (ROS2) $v.$ $galactic$. Each flight execution operates the UAV in a normal or fault setting. The first flew the UAV with 4 undamaged propellers. The latter flew the UAV with 1 up to 4 of its propellers broken. The broken propeller cases are generated by cutting $\approx$ 1 centimeter of the propeller edge (see Fig. 1a).

A total of 100 flights were executed compounding over 2 hours of total flight time, out of which 20 flights are in *normal* UAV setting and 80 flights are in *fault* UAV setting.

B. Detection of Physical UAV Faults

Our performed experiments aimed at answering the following research questions (RQ): (***RQ1***) *What is the accuracy performance of traditional techniques for detecting UAV faults?* (***RQ2***) *What are the processing costs of selected techniques?*

We evaluate two sets of classification techniques based on traditional ML and the more recent DL [17], [12]. The classifiers are evaluated using a different view (feature set): Accelerometer, Gyroscope, and Audio. Based on the collected values, the classification goal is to detect UAV physical faults in real-time.

The ML-based approaches were implemented through a Random Forest (RF), and a Gradient Boosting based on XGBoost (XGB) The ensemble classifiers were implemented with 100 decision trees as their base learners, each using $gini$ as the node split quality metric. The XGB classifier relies upon a 0.3 learning rate value, with $deviance$ as the loss function. The traditional ML makes use of hand-designed features for each view, as follows:

- *Accelerometer*. 96 statistical features in a 1-second periodicity (samples generated in 100hz frequency).
- *Gyroscope*. 96 statistical features in a 1-second periodicity (samples generated in 100hz frequency).
- *Audio*. 640 MFCC features for every 1-second sampled UAV microphone audio.

The DL approaches were evaluated with Transformer Encoder (TrEnc) architecture and a Long-Short Term Memory (LSTM) [18]. In contrast to the traditional ML, the DL-based techniques evaluate the raw data based on 100 timesteps, which account for 1-second of data. At the same

TABLE I: Accuracy and processing time of traditional UAV fault detection techniques. Processing time is based on the average event processing for feature extraction and classification on our implemented prototype (Section V)

View	Cls.	F1	FPR	FNR	Proc. Time (ms) (Feat. Ext., Class.)
Acc.	RF	0.82	4.5	17.9	1.72 (99.2%, 0.80%)
	XGB	0.82	4.5	18.2	1.71 (99.65%, 0.35%)
	LSTM	0.80	4.9	17.4	17.4 (100%, -)
	TrEnc	0.79	5.1	19.2	7.80 (100%, -)
Gyro.	RF	0.76	5.9	22.5	1.72 (99.08%, 0.92%)
	XGB	0.79	5.2	20.1	1.71 (99.62%, 0.38%)
	LSTM	0.72	6.8	26.8	17.65 (100%, -)
	TrEnc	0.72	6.7	24.8	6.60 (100%, -)
Audio	RF	0.73	6.8	27.7	10.53 (99.42%, 0.58%)
	XGB	0.74	6.6	26.4	10.35 (99.67%, 0.33%)
	LSTM	0.74	6.4	24.6	26.87 (38.37%, 61.63%)
	TrEnc	0.79	5.1	19.3	24.61 (58.10%, 41.90%)

time, the audio-based approach also evaluates the MFCC features. The LSTM was implemented with the following architecture: *(I) Input:* 32-sized embedding layer; *(II) LSTM:* two bidirectional 32-sized LSTM layers. *(III) Output:* a 5-unit dense layer. The TrEnc was implemented with the following architecture: *(I) Input:* 128-sized embedding layer; *(II) TransformerEncoder:* two transformer encoder layers with 512 units and 16 heads. *(III) Output:* a 5-unit $dense$ layer. The implemented architectures used the *categorical crossentropy* as $loss$ using $adam$ optimizer, with 100 epochs and a batch size of 64.

The parameters of the selected techniques were empirically set. The dataset was split into *train*, *test*, and *validation*, each respectively composed of 60%, 20%, and 20% of the original flights from our testbed. A random undersampling without replacement is used in the training procedure to balance the occurrence between the classes. The ML classifiers were implemented through *scikit-learn* API $v0.24$, while the DL was implemented on top of PyTorch API $v1.12.1$. The implemented prototype is further described in Section V. The classifiers were evaluated according to their F1, False-Positive (FPR), False-Negative (FNR) rates. The F1 was computed as the harmonic mean of both sensitive and recall metrics. The FPR denotes the ratio of *normal* instances correctly classified as a *fault*, while the FNR denotes the ratio of *fault* instances correctly classified as *normal*.

Our first experiment aims at answering RQ1 and evaluates the accuracy performance of selected techniques for UAV fault detection. Table I shows the accuracy performance of selected techniques according to the used feature set. Widely used approaches in the literature cannot reach low FPR and FNR rates. For instance, the most accurate model (*Acc-based* RF, Table I) presented an FPR of 4.5% and an FNR of 17.9%. As a result, the selected approaches cannot be reliably used in autonomous UAVs, as an FP may unintentionally trigger the UAV landing. In contrast, an FN may allow a defective UAV to put the environment where it operates at risk.

Our second experiment aims at answering RQ2 and evaluates the processing costs of the selected UAV fault detection techniques. Table I shows the average event processing time according to the used classifier and feature set. The feature extraction task is generally responsible for the most processing-demanding part of the detection scheme. For example, accounting for an average processing part of the RF-based detection of 99.2%, 99.08%, and 99.42% for the accelerometer, gyroscope and audio detection approaches respectively. Therefore, to provide a lightweight detection scheme, proposed fault detection approaches must also address the processing costs related to the feature extraction task.

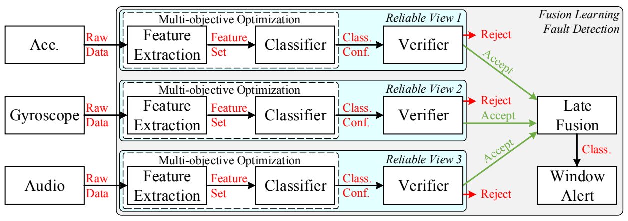


Fig. 2: Overview of our proposed model for reliable and lightweight onboard fault detection in fully autonomous UAVs.

IV. Reliable and Lightweight Onboard Fault Detection in Autonomous UAVs

To address the aforementioned challenges, we propose a new reliable and lightweight onboard fault detection in autonomous UAVs. Our proposal aims to decrease the computational costs of both feature extraction and classification tasks while maintaining classification reliability. The overview of our model is shown in Figure 2 and is implemented in four main steps, namely *Multi-objective Feature Selection*, *Classification Verification*, *Fusion Learning*, and *Window Alert*.

The *Multi-Objective Feature Selection* and *Classification Verification* are performed at a single view (feature set) level. The first aims at selecting the best subset of features that can be used to decrease the feature extraction computational costs, while also improving the system accuracy. The main insight of such an approach is to consider the feature extraction computational costs during the model development process for detecting UAV faults. The latter aims to ensure that our proposed model only uses highly confident classifications. We assess the classification confidence values as a metric of classification correctness, rejecting low-confident decisions.

To improve our system accuracy and reliability, our model uses *Fusion Learning* and *Window Alert*. The *Fusion Learning* evaluates the classification outcome following a multi-view procedure (Fig. 2, *Reliable View* 1 to 3). Thus, our scheme can improve the classification reliability while decreasing the rejection rate caused by our *Classification Verification* scheme. Finally, the *Window Alert* signal faults following a sliding window approach, ensuring that the UAV only acts if a given predefined set of alarms are triggered.

The following subsections further describe our proposed model, including the modules that implement it.

A. Reliable Lightweight Classification

Fault detection in autonomous UAVs must be performed with minimal processing requirements while maintaining classification accuracy and reliability. To address such a challenge, we execute the model building with a multi-objective feature selection task, coped with a verification technique.

The multi-objective feature selection aims at building a classification model that decreases the computational costs of the feature extraction task while maintaining classification accuracy. More specifically, the model-building procedure is implemented through a wrapper-based multi-objective feature selection aiming at minimizing the following objectives:

$$objective_{processing}(f) = \sum_{i=1}^{n} processingTime(f_i) \quad (1)$$

$$objective_{error} = 1 - \frac{\overbrace{TP/(TP+FN)}^{\text{sensitivity}} + \overbrace{TN/(TN+FP)}^{\text{specificity}}}{2} \quad (2)$$

where f denotes the set of used features, *processingTime* is a function that computes the processing cost for the extraction of a given feature f_i, TP the number of true-positive samples, and TN the number of true-negative samples. Thus, our multi-objective feature selection aims to decrease the feature extraction computational costs (Eq. 1, $objective_{processing}$), as well as the classification error (Eq. 2, $objective_{error}$), as computed by the inverse average of sensitivity and specificity.

We use classification with a reject option to ensure classification reliability when our UAV fault detection scheme is used in autonomous settings. Thus, we assess the classification confidence values of classified events through a verifier module (Fig. 2, *Verifier*), rejecting low-confident and potentially misclassified events. The classification confidence is classifier agnostic, e.g., the RF classifier computes the classification confidence according to the ratio of decision trees that classified a given instance as the assigned class.

Our main insight is using classification confidence as a classification correctness measure in production deployment, ensuring that only highly confident classified events are used for triggering alerts. Notwithstanding, it can be used as a ML model quality measure, signaling the operator's unreliable model deployed in the UAV. The classification rejection threshold must be defined according to the operator's needs. A higher rejection threshold can increase the system's accuracy while rejecting more samples as a trade-off. In contrast, a lower rejection threshold can increase the number of classified events, however, it is prone to higher error rates.

B. Fusion Learning

Several techniques have been proposed for UAV fault detection, ranging from increasing the classifier complexity to relying on a different data source. Yet, proposed schemes cannot meet high detection accuracies to be reliably used in autonomous UAV settings. In light of this, our proposed model performs the classification task in a fusion learning setting, following a multi-view rationale.

The proposal classification procedure is shown in Figure 2. It starts with the continuous collection of raw data from multiple UAV sensors, e.g. accelerometer, gyroscope, and audio. The collected data is used as input to a feature extraction module, which extracts, for each view, the set of features selected by our multi-objective feature selection procedure (see Section IV-A). The extracted feature sets are classified by the associated classifier, which outputs a corresponding classification confidence level. The verifier module evaluates the classification confidence output and accepts highly confident classifications based on a predefined threshold. Finally, accepted classifications are used as input by a late fusion module, which goal is to evaluate the accepted classifications for a final decision. The module defines the event label based on the highest sum of the classification confidence scores of accepted instances, according to the following equation.

$$latefusion(c) = max(\sum_{i=1}^{n} c_i^{normal}, \sum_{i=1}^{n} c_i^{fault}) \quad (3)$$

where c is a vector of classification confidence of accepted views, n the number of accepted views, c_i^{normal}, and c_i^{fault} the classification confidence at the i^{th} view, for the normal and fault classes respectively. Thus, our late fusion module establish the event label based on the used set of views in a reliable manner, as we also consider the classification confidence scores. To ensure that misclassifications are not used to signal alerts, we use a sliding window for decision-making (Fig. 2, *Window Alert*). Therefore, the UAV only performs a decision when a given predefined number of fault or normal events are surpassed over the last classified events.

C. Discussion

Our proposal aimed at enabling reliable onboard and lightweight fault detection in autonomous UAVs. A multi-objective feature selection is used to decrease the computational costs of the feature extraction task without trade-offs on accuracy. Classification with a reject option approach is used to improve classification reliability, suppressing potential misclassifications, while also indicating to the operator the UAV model quality. Late fusion with a multi-view procedure is used to provide better classification accuracy and higher reliability, decreasing system rejection by leveraging several UAV sensors. Finally, a sliding window of alerts is used to suppress potential misclassifications as time passes. As a result, our proposed model can provide reliability in UAV fault detection while maintaining system accuracy with significantly reduced processing requirements.

V. Prototype

We implemented a proposal prototype on top of our previously described UAV (see Section III), as shown in Figure 3.

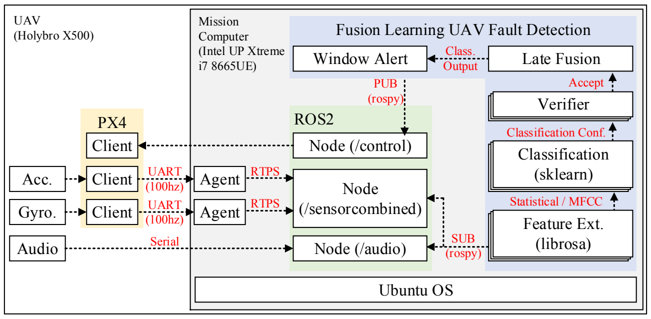


Fig. 3: Proposed model prototype overview.

The prototype is executed on the Intel UP Xtreme i7 8665UE UAV mission computer running an Ubuntu OS $v20.04$. The UAV performs autonomous flights as managed by the PX4 flight controller, connected to the mission computer via an RTPS bridge. The prototype evaluates three different views based on the accelerometer, gyroscope, and audio-collected data. The accelerometer and gyroscope sensors are connected to the PX4 and generate samples at a 100hz frequency. The audio is collected through a Seed Studio ReSpeacker Mic Array connected by a serial connection to the mission computer. The generated data is sent to the associated topic in ROS2 $v.galactic$.

Our prototype continuously ingests the data by subscribing to the related topics using the $rospy$ API. The accelerometer and gyroscope data are ingested through the */sensorcombined* topic, while their statistical features are extracted through python implementation (see Section III-B). The audio is collected through the */audio* topic, having the MFCC features extracted by the $librosa$ API. The features are extracted in a 1-second periodicity and classified using the ML algorithm implemented through *scikit-learn* API $v.0.24$. The *Verifier* module evaluates the classification confidence values through the *predict_proba* function. Finally, the *Window Alert* module decides to land the UAV in a fault-detected setting through a publish on the */control* topic using the $rospy$ API.

VI. Evaluation

Our evaluation aims at answering the following research questions (RQ): (***RQ3***) *Does our proposed multi-objective feature selection improve the system's performance?* (***RQ4***) *Does our proposed verifier module improve the system classification accuracy?* (***RQ5***) *What is the accuracy performance of our scheme with late fusion?* (***RQ6***) *What is the proposed model processing performance?*

The experiments use the same dataset used previously (see Section III-A)

A. Reliable Lightweight Classification

Our first experiment aims to answer RQ3 and evaluates how our proposed multi-objective feature selection can decrease the feature extraction computational costs while maintaining the systems' accuracy. We implement our scheme as a wrapper-based feature selection using the *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [19] on top of *pymoo*

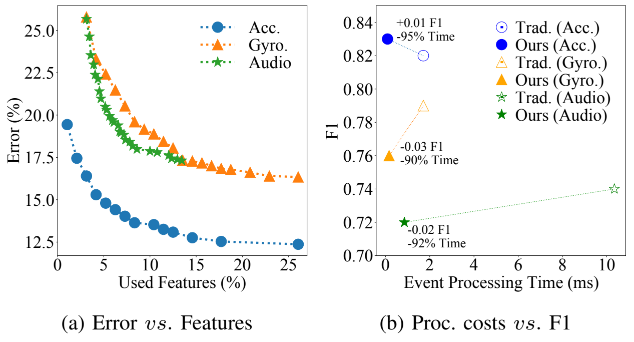


Fig. 4: Multi-objective feature selection for XGB classifier. Event processing time measured as average per event on UP i7 8665UE UAV mission computer.

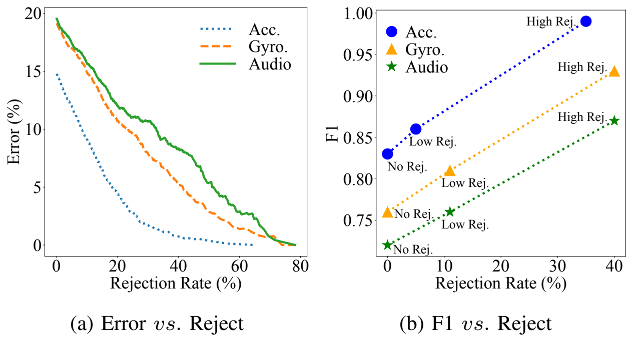


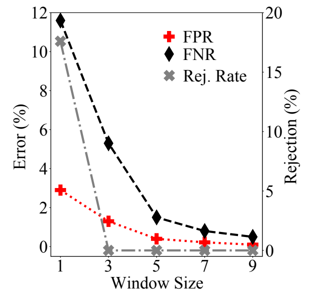
Fig. 5: Verifier module performance with the proposed multi-objective optimization (Fig. 4) with the XGB classifier.

API. The NSGA-II uses a 500 population size, 200 generations, a crossover of 0.3, and a mutation probability of 0.1. The multi-objective feature selection aims at decreasing the $objective_{processing}$ (Eq. 1) and $objective_{error}$ (Eq. 2) as measured on the validation dataset.

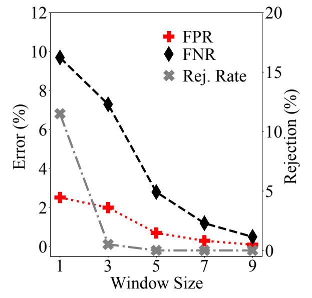
Figure 4a shows the pareto curve of the feature selection technique for the XBT classifier (see Table I, best ML model). Our proposed multi-objective feature selection significantly decreased the feature extraction processing costs with a marginal effect on accuracy. Figure 4b shows the average event processing time for each view according to the used operation point (see Fig. 4a). Our proposal decreased the computational costs by up to 95% (Acc. view), with an accuracy impact of as little as 0.03 in F1 (Gyro. view).

Our second experiment answers RQ4 and evaluates our proposed classification confidence assessment technique (Fig. 2, *Verifier*). We evaluate the error $vs.$ reject tradeoff for the selected operation points (closest to 0% in both objectives) using the Class Related Threshold (CRT) technique. More specifically, we vary the rejection threshold for each class (*normal* and *fault*) in a 0.001 interval.

Figure 5a shows the error $vs.$ reject tradeoff for the XBT classifier using the selected multi-optimization operation points (Fig. 4a). Our proposed scheme that assesses the classification confidence can significantly decrease the classification error. As the rejection operation points should be defined according to the operator's needs, we select three operation points, namely *No-rejection*, *Low-rejection*, and



(a) Low Operation Point



(b) High Operation Point

Fig. 6: Final classification performance of our model according to the used rejection operation point and window size (Fig. 2, *Window Alert*). A higher window size increases accuracy but introduces alert delay as a trade-off caused by the sliding window approach.

TABLE II: Proposed model accuracies and rejection rates (M.O.=Multi-objective).

View	Method	OP	F1	FPR	FNR	Rej.
Late Fusion		No	0.88	3.1	12.2	-
		Low	0.88	2.9	11.6	1.76
		High	0.90	2.5	9.7	11.5
Acc.	M.O.	-	0.83	4.5	16.3	-
	M.O. + Verifier	Low	0.86	3.4	13.2	5.0
	M.O. + Verifier	High	0.99	0.2	1.4	35.0
Gyro.	M.O.	-	0.76	6.1	24.0	-
	M.O. + Verifier	Low	0.81	4.6	18.3	11.0
	M.O. + Verifier	High	0.93	1.7	8.0	40.0
Audio	M.O.	-	0.72	7.0	27.6	-
	M.O. + Verifier	Low	0.76	5.9	24.4	11.0
	M.O. + Verifier	High	0.87	3.2	17.1	40.0

High-rejection empirically set for each view based on their performance. Figure 5b shows the accuracy improvement obtained by our verification technique according to the used rejection operation points. Our proposed model can improve the F1-Score by up to 0.16 with a rejection of 35% (Acc. view, *High-rejection*).

B. Fusion Learning

To answer RQ5 we investigate our proposed model performance with the implemented multi-objective optimization, classification assessment, and late fusion technique (Fig. 2). More specifically, we apply our proposed late fusion technique (Eq. 3) to the accepted instances by our verifier module. Table II shows the accuracy and rejection rates obtained by our proposed model (Late Fusion). Our proposed scheme can significantly decrease the rejection rate while also improving the accuracy rates obtained by traditional approaches. For instance, with a *Low* rejection operation point, our late fusion technique rejects only 1.76% of instances while presenting an FPR of only 2.9% and an FNR of only 11.6%.

Finally, we evaluate our model implemented with our proposed *Window Alert* module. We trigger alerts if a given predefined number of events are classified as a fault by our late fusion technique, further improving our model accuracy (Fig. 2). Figure 6 shows the FP and FN rates according to the used window alert size. Our scheme was able to improve the obtained accuracies when compared to traditional techniques

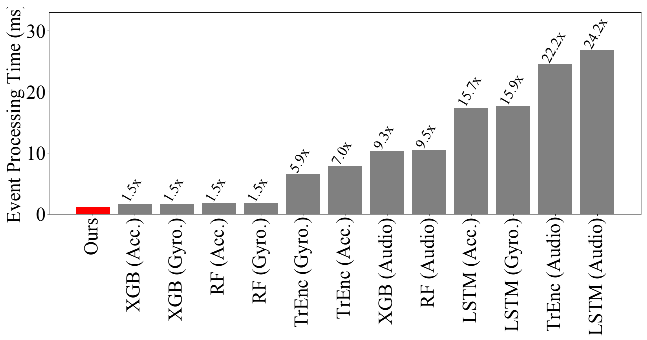


Fig. 7: Average event processing time of evaluated techniques on the UAV mission computer platform (lower is better). Event processing time measured as average per event on UP i7 8665UE UAV mission computer.

significantly. For instance, with a window size of 5 events and a high rejection operation point (Fig. 6b), our scheme can reach an FPR of only 0.4% and an FNR of 0%, reducing the mentioned metrics by 4.1 and 17.9 when compared to the most accurate traditional approach (Acc. RF, Table I).

C. UAV Performance

We evaluate the processing performance of selected techniques when executed on the UAV mission computer (see Section V). Figure 7 shows the average event processing time of selected techniques. Our proposed model decreased the computational costs in both evaluated platforms significantly. For instance, traditional techniques demand at least 1.5× more processing time when compared to our approach. In practice, when compared to the most accurate traditional model (Table I, ACC. RF), our scheme improves the F1 in 0.17, FPR in 4.1, and FNR in 17.9 (Fig. 6b *vs.* Table I), while demanding only 1.11ms of processing time, a 43% decrease.

D. Discussion

Our model provided reliable, lightweight, and onboard UAV fault detection in autonomous settings. The reliable lightweight classification was achieved through a multi-objective feature selection that decreases feature extraction computational costs (up to 95%), coped with a verification technique, and was able to improve F1-Score (up to 0.17). Notwithstanding, our proposed late fusion technique significantly decreased the rejection rate (from 35% to only 1.76%) while providing significantly high accuracy rates (only 0.4% of FPR in a 5-second long window size). The prototype has shown the proposal's feasibility, requiring only 1.1ms of average event processing time.

VII. Conclusion

This work proposed a new reliable and lightweight UAV fault detection scheme for onboard execution. The proposed model significantly improved the classification accuracy, decreasing processing costs compared to traditional techniques. Notwithstanding, our scheme provides a model quality metric for fully autonomous settings, enabling operators to identify unreliable ML models deployed in fully autonomous UAVs.

REFERENCES

[1] Z. Xu, D. Deng, and K. Shimada, “Autonomous UAV exploration of dynamic environments via incremental sampling and probabilistic roadmap,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2729–2736, Apr. 2021.

[2] V. Sadhu, S. Zonouz, and D. Pompili, “On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020.

[3] P. Horchulhack, E. K. Viegas, and A. O. Santin, “Toward feasible machine learning model updates in network-based intrusion detection,” *Computer Networks*, vol. 202, p. 108618, 2022.

[4] B. Wang, X. Peng, M. Jiang, and D. Liu, “Real-time fault detection for UAV based on model acceleration engine,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9505–9516, Dec. 2020.

[5] J. Zhang, X. Liang, M. Wang, L. Yang, and L. Zhuo, “Coarse-to-fine object detection in unmanned aerial vehicle imagery using lightweight convolutional neural network and deep motion saliency,” *Neurocomputing*, vol. 398, pp. 555–565, Jul. 2020.

[6] X. Hua, X. Wang, T. Rui, F. Shao, and D. Wang, “Light-weight UAV object tracking network based on strategy gradient and attention mechanism,” *Knowledge-Based Systems*, vol. 224, p. 107071, Jul. 2021.

[7] J. Steinbrener, C. Brommer, T. Jantos, A. Fornasier, and S. Weiss, “Improved state propagation through AI-based pre-processing and down-sampling of high-speed inertial data,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2022.

[8] A. Altinors, F. Yol, and O. Yaman, “A sound based method for fault detection with statistical feature extraction in UAV motors,” *Applied Acoustics*, vol. 183, p. 108325, Dec. 2021.

[9] A. Mairaj, A. I. Baba, and A. Y. Javaid, “Application specific drone simulators: Recent advances and challenges,” *Simulation Modelling Practice and Theory*, vol. 94, pp. 100–117, Jul. 2019.

[10] E. Yel, S. Gao, and N. Bezzo, “Meta-learning-based proactive online planning for UAVs under degraded conditions,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 320–10 327, Oct. 2022.

[11] J. Geremias, E. K. Viegas, A. O. Santin, A. Britto, and P. Horchulhack, “Towards multi-view android malware detection through image-based deep learning,” in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, 2022, pp. 572–577.

[12] B. Wang, X. Peng, M. Jiang, and D. Liu, “Real-time fault detection for UAV based on model acceleration engine,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9505–9516, Dec. 2020.

[13] M. Gjoreski, V. Janko, G. Slapničar, M. Mlakar, N. Reščič, J. Bizjak, V. Drobnič, M. Marinko, N. Mlakar, M. Luštrek, and M. Gams, “Classical and deep learning methods for recognizing human activities and modes of transportation with smartphone sensors,” *Information Fusion*, vol. 62, pp. 47–62, Oct. 2020.

[14] A. Bondyra, M. Kołodziejczak, R. Kulikowski, and W. Giernacki, “An acoustic fault detection and isolation system for multirotor UAV,” *Energies*, vol. 15, no. 11, p. 3955, May 2022.

[15] I. H. B. Pizetta, A. S. Brandão, and M. Sarcinelli-Filho, “UAV thrust model identification using spectrogram analysis,” *Automation*, vol. 2, no. 3, pp. 141–152, Aug. 2021. [Online]. Available: https://doi.org/10.3390/automation2030009

[16] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, “Machine learning inspired sound-based amateur drone detection for public safety applications,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2526–2534, Mar. 2019.

[17] R. R. dos Santos, E. K. Viegas, A. O. Santin, and V. V. Cogo, “Reinforcement learning for intrusion detection: More model longness and fewer updates,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[18] D. Lim, D. Kim, and J. Park, “Momentum observer-based collision detection using LSTM for model uncertainty learning,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.