

Convolutional neural network design and evaluation for real-time multivariate time series fault detection in spacecraft attitude sensors

Gallon, Riccardo; Schiemenz, Fabian; Menicucci, Alessandra; Gill, Eberhard

DOI

[10.1016/j.asr.2025.06.068](https://doi.org/10.1016/j.asr.2025.06.068)

Publication date

2025

Document Version

Final published version

Published in

Advances in Space Research

Citation (APA)

Gallon, R., Schiemenz, F., Menicucci, A., & Gill, E. (2025). Convolutional neural network design and evaluation for real-time multivariate time series fault detection in spacecraft attitude sensors. *Advances in Space Research*, 76(5), 2960-2976. <https://doi.org/10.1016/j.asr.2025.06.068>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.



Convolutional neural network design and evaluation for real-time multivariate time series fault detection in spacecraft attitude sensors

Riccardo Gallon^{a,b,*}, Fabian Schiemenz^b, Alessandra Menicucci^a, Eberhard Gill^a

^a Department of Space Systems Engineering, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, HS Delft 2629, Netherlands
^b Airbus Defence and Space GmbH, Claude-Dornier Straße, Immenstaad am Bodensee 88090, Germany

Received 6 June 2024; received in revised form 10 May 2025; accepted 25 June 2025
Available online 1 July 2025

Abstract

Traditional anomaly detection techniques onboard satellites are based on reliable, yet limited, thresholding mechanisms which are designed to monitor univariate signals and trigger recovery actions according to specific European Cooperation for Space Standardization (ECSS) standards. However, Artificial Intelligence-based Fault Detection, Isolation and Recovery (FDIR) solutions have recently raised with the prospect to overcome the limitations of these standard methods, expanding the range of detectable failures and improving response times.

This paper presents a novel approach to detecting stuck values within the Accelerometer and Inertial Measurement Unit of a drone-like spacecraft for the exploration of Small Solar System Bodies (SSSB), leveraging a multi-channel Convolutional Neural Network (CNN) to perform multi-target classification and independently detect faults in the sensors.

Significant attention has been dedicated to ensuring the compatibility of the algorithm within the onboard FDIR system, representing a step forward to the in-orbit validation of a technology that remains experimental until its robustness is thoroughly proven. An integration methodology is proposed to enable the network to effectively detect anomalies and trigger recovery actions at the system level. The detection performances and the capability of the algorithm in reaction triggering are evaluated employing a set of custom-defined detection and system metrics, showing the outstanding performances of the algorithm in performing its FDIR task.

© 2025 The Author(s). Published by Elsevier B.V. on behalf of COSPAR. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Keywords: Aerospace; PUS-based FDIR; Convolutional Neural Network; Time series; Evaluation

1. Introduction

Fault Detection, Isolation, and Recovery (FDIR) is critical in every operational spacecraft, as undetected failures can have severe consequences, potentially jeopardizing mission uptime. Traditional FDIR approaches typically employ straightforward thresholding techniques on onboard parameters (Tipaldi and Bruenjes, 2015). While these methods offer robustness, they may lack efficiency

in detecting certain types of failures. Operational FDIR systems in European satellites are based on the so-called Packet Utilization Standard (PUS) (ECSS, 2016; Savoie, 2023), which defines a number of different Services regulating space-ground communications. Among these, Service 12 (Onboard Monitoring), Service 5 (Event Reporting), Service 19 (Event-Action), and action-type services like Service 18 (Onboard Control Procedure) or 21 (Request Sequencing) are mostly used in FDIR.

Service 12 (S12) is responsible for onboard parameter and functional monitoring (pMon and fMon) for failure detection purposes. The monitoring can apply to the value

* Corresponding author.

E-mail address: riccardo.gallon@airbus.com (R. Gallon).

of a signal, its expected value or its variation, checking instances where they fall Out-of-Limit (OOL) beyond intervals defined during mission design. Typically, a logical combination (AND/OR) of pMons is associated to an fMon, which triggers upon the associated pMons registering OOL for a specified number of consecutive samples over a defined time interval (*persistence*). Service 5 is employed to link the triggered fMon to an Event ID, which usually represents the cause of the detected failure. Service 19 links the Event ID to a recovery action. Finally, action-type services (e.g. S18 or S21) are used to execute the recovery actions.

The FDIR subsystem typically includes multiple chains of the above-mentioned PUS Services, structured across five hierarchical levels defining how to address failures. The levels go from unit to subsystem to system level and can be activated by the adjacent lower level only when the latter cannot resolve or isolate a fault.

Although representing the operational state of the art, PUS-based FDIR presents several well-known limitations, which mainly relate to its specific design for univariate time series and to the parameter monitoring mechanism. On the one hand, failures impacting several telemetry signals coming from different components and/or subsystems, as well as inherently multivariate data (e.g. images) cannot efficiently be analyzed through PUS-based FDIR because they do not fit into the univariate time series definition. On the other hand, failures affecting univariate time series where the signal evolves anomalously within the boundaries set by the thresholds exist, and are traditionally undetectable unless their consequences become visible to the PUS Services on higher levels. Usually, this means the faults propagated and increased in severity.

Stuck values in multivariate time series data are a typical example of faults where the PUS-based detection shows inefficiency. These faults cause the signal to remain stuck on the same value over time, either involving one or all the components, eventually not exceeding the threshold boundaries. The most common detection methodology for stuck values is based on the consequences that they cause in other parameters of the system, eventually falling OOL, and are detected through higher-level FDIR. This approach, while representing the operational state of the art, lets the fault evolve uncontrollably for a certain time, thus constitutes a limitation which is worth trying to mitigate or solve. The application of Artificial Intelligence (AI) algorithms to the problem of stuck values arises from the need to enhance the detection of these faults, especially in terms of early detection and accuracy (Wang et al., 2024). To this purpose, this work employs the use case of the *Astrone KI* spacecraft to provide data and demonstrate the validity of the presented approach. *Astrone KI* is a drone-like spacecraft designed for autonomous relocation maneuvers within the challenging environment of an SSSB, including take-off, landing and cruising over its surface. The system design is thoroughly described in Martin et al. (2021). In this work, it is stated that the *Astrone* vehi-

cle would be a demonstrator of relative navigation over the asteroid surface employing camera- and LiDAR-based navigation, aided by accelerometers and gyroscopes (both enclosed in an Inertial Measurement Unit, IMU). The *Astrone KI* system is meant to provide the multivariate time series for the detection of stuck values by means of the mentioned accelerometer and gyroscope. Most importantly, the infrastructure given by the spacecraft allows to investigate the AI-based FDIR functionalities on a realistic dataset, while enabling the AI integration within the PUS-based FDIR.

The contribution of this work to the state of the art is threefold. First, a multi-target, multi-channel, CNN-based approach to fault detection in time series coming from the accelerometer and gyroscope mounted onboard the vehicle is proposed. Second, a procedure to integrate the algorithm in the framework of a functional simulator of the spacecraft is described, facing the issues connected to online fault detection, defining the pipeline from input pre-processing to the classification of anomalous signal samples, concluding with the recovery action triggering by the PUS-based FDIR chain implemented onboard. Lastly, a set of case-specific metrics is proposed, in order to evaluate the AI algorithm as integrated within the onboard FDIR subsystem, capable of expressing on one side the capability of the algorithm to detect faults, on the other side the performances of the whole AI-based FDIR.

This paper is organized as follows: Section 2 presents a review of the related works on AI-based FDIR for time series. Section 3 details the process of dataset generation and anomaly injection for the purpose of training and testing the neural network. Section 4 presents the FDIR methodology, including data manipulation, CNN algorithm utilization, and a specially designed integration strategy with the onboard system. Section 5 delineates the performance metrics tailored for the solution presented in Section 4, along with the optimization process carried out to refine these metrics and the results obtained. Section 6 presents the conclusions of the paper and an outlook on future work.

2. Background

AI-based onboard FDIR, especially about time series, is a recent trend, but it has been already identified as one of the most promising fields of research due to the foreseen benefits that AI may bring to this specific functionality (Meß et al., 2019; Murphy et al., 2021; Russo and Lax, 2022). More specifically, scientific literature concentrates more on the field of Failure Detection and Isolation than on the Recovery part (Ciancarelli et al., 2023), which typically relies on conventional methods (e.g. configuration change, power-cycle, redundancy switch). A variety of AI methodologies are employed to address the FDI task, where Deep Learning (DL) (Pang et al., 2021) is the

recently-emerged alternative to the more traditional Machine Learning (ML) (Nassif et al., 2021).

ML applications to anomaly detection in time series include mainly Support Vector Machines (SVM) (Xiong et al., 2011; Gao et al., 2012a), Clustering Methods (Liu et al., 2017; Azevedo et al., 2012; Gao et al., 2012b) and Decision Trees (Kuhn and Johnson, 2013). These approaches, together with their very well-established state in academic literature, are also the ones that meet the desired interpretability features in the onboard FDIR. They are indeed based on fully deterministic algorithms, in contrast to Deep Learning algorithms which are essentially black boxes (Gallon et al., 2024).

An example of ML-based FDIR is provided in Xiong et al. (2011) and Gao et al. (2012a), which present the application of an SVM for anomaly detection in real telemetry data including voltages, currents and temperatures. A first pre-processing step is carried out by expressing data in terms of statistical descriptors and applying Principal Component Analysis (PCA). Following, the anomaly detection is performed by the SVM. Clearly, the classifiers SVMs in Xiong et al. (2011) and Gao et al. (2012a) are guided by the choice of the data features given as input, which on the one hand allows to interpret the reasoning of the algorithm, on the other hand excludes possibly better solutions. Related to this, the main reason why DL has replaced ML during the years is that the former, being a black-box model, is able to grasp more complex features of the input data and establish relations among them. Indeed, it has proved itself very efficient in anomaly detection tasks, especially dealing with multi-dimensional data such as images (Wang et al., 2021) or multivariate time series (Li and Jung, 2023; Lakey and Schlippe, 2024), as it will be clarified in the following examples.

Studies on DL-based anomaly detection for time series can be divided in two main approaches: direct classification of the input or its reconstruction.

Direct classification applications come mainly from outside the space domain, with this not being an obstacle to their applicability from the perspective of the algorithm only, but instead a resource to provide potentially adaptable ideas. In the case of Cui et al. (2016), the authors provide a multi-channel CNN capable of classifying a time series by feeding each channel with a differently manipulated version of the series itself, in order to orient the feature recognition and boost the successive classification stage. The solution has been tested on 46 datasets coming from different real-world domains and compared to 14 other state-of-the-art classifiers, proving its outstanding performances. Similarly, Canizo et al. (2019) employ mixed Convolutional-Recurrent Neural Network architectures for time series classification, where a comparison is established between 15 different networks including the multi-channel and multi-head cases, as well as different kinds of Convolutional and Recurrent layers. The test dataset is a two-classes dataset drawn from an industrial framework

and dealing, among the others, with linear accelerations, angular rates and voltages.

A hybrid approach between ML and DL, related to the space domain, is presented in Mansell and Spencer (2021), where a One-Class SVM followed by a Long Short-Term Memory (LSTM) neural network performs fault detection and isolation on raw telemetry data coming from an Attitude Determination and Control System (ADCS) simulator inherited from the mission LightSail 2. Here, 21 different kinds of faults at unit level are applied to onboard sensors and actuators, in order to test the FDI capability. The solution relies firstly on the discrimination made by the SVM and some additional rule-based checks to perform the detection task, then complemented with isolation by the LSTM network. Such architecture performs almost perfectly in accuracy on the majority of the injected faults and constitutes a pioneering work on AI-based FDIR in the space domain.

Signal reconstruction methodologies aim typically at predicting the value of time series forward in time. Specifically to FDIR, they are usually complemented by classifiers which make use of the predicted (*reconstructed*) signal to perform the detection and eventually identification tasks (Hundman et al., 2018; Li et al., 2019; Baireddy et al., 2021; Xiang and Lin, 2021). In Hundman et al. (2018), LSTM-based predictors with dynamic thresholding mechanisms are used to perform anomaly detection tasks and are shown to obtain state-of-the-art results on datasets coming from telemetry of well-known NASA satellites (e.g. Mars Reconnaissance Orbiter). The technique is based on applying a sliding window to the dataset made of time series and feeding it to the network, whose output is a reconstructed version of the input window itself. Then, the proper anomaly detection step consists in taking the difference between the output of the network and the real telemetry (*reconstruction error*) and comparing it to a threshold, chosen based on the probability distribution of the data in the specific window (i.e. *dynamic thresholding*).

Li et al. (2019) took over the work of Hundman et al. (2018) introducing a stacked predictor including LSTM layers and an SVM, which was able to perform on the same dataset.

A more recent approach is presented by Baireddy et al. (2021), which employ transfer learning techniques to improve the work of Li et al. (2019) and Hundman et al. (2018), using again the same dataset but varying the predictor shape. In this case, the predictor is indeed based on LSTM and built first as a general predictor for time series, then fine-tuned employing weights transfer to better fit the training data. Finally, a dynamical thresholding mechanism inherited from Hundman et al. (2018) is applied for the classification stage. The solution presented in Baireddy et al. (2021) proves to obtain better performances than both Li et al. (2019) and Hundman et al. (2018), as well as other state-of-the-art methods based on the same predictor-classifier approach. Plus, it introduces transfer

learning in the AI-based FDI, marking an emerging trend in the field-specific academic literature (Di et al., 2021; Baireddy et al., 2021; Zhang et al., 2023). This methodology is indeed capable to mitigate abrupt alterations in model structures when training for closely related tasks, and consequently to reduce the amount of training data needed, particularly in scenarios where those are limited, e.g. real telemetry.

Finally, other notable works in the field of time series anomaly detection employing DL are Alos and Dahrouj (2021), Wang et al. (2024) and Zhong et al. (2021), which faced the use case of stuck value faults in flight data of Unmanned Aerial Vehicles (UAVs). These works propose state-of-the-art algorithms (e.g. LSTMs, Convolutional LSTMs and Transformers) to reconstruct the signals coming from onboard UAV sensors, e.g. accelerometers and gyroscopes, evaluating the reconstruction performance itself (Zhong et al., 2021) or employing a threshold classifier (Alos and Dahrouj, 2021; Wang et al., 2024), similar to the one originally introduced in Hundman et al. (2018).

Techniques of direct classification are typically less employed than signal reconstruction ones in the space domain, because they allow less flexibility in the classification phase. Indeed, while in the former case the algorithm directly outputs the label associated to the input (i.e. *fault/no fault*), the latter outputs in a wide sense the input itself, which has to be classified afterwards by complementary methodologies. In other words, the black-box part of the model goes from input to output in the case of direct classification, while it is limited to the signal reconstruction in the other case, which can instead employ robust and/or interpretable classifiers.

While the signal reconstruction approach may seem more suitable to FDIR applications, it needs specific attention in the integration phase of this specific study, where the match between the reconstruction mechanism and the persistency-reaction triggering mechanism of PUS mandates further investigation. This is not the case for direct classification methodologies which mainly present the challenge of where to place the algorithm in the PUS chain. For this reason, the present research concentrates on the direct classification, although not excluding the signal reconstruction for future work.

Concerning the evaluation of ML-based anomaly detection algorithms with case-specific metrics, the problem consists in defining new ways of evaluating the performances of an AI algorithm in those fields of application where conventional metrics (e.g. Precision, Recall, F1-Score) are not representative. Specifically to failure detection tasks for time series, where faults are more likely to occur in intervals than in single samples, a measure of the overlap of these intervals with the AI prediction can give much better insight in the detection than measuring sample-by-sample metrics (e.g. accuracy).

Mathonsi and Zyl (2022) and Nalepa et al. (2022) deal with this performance evaluation topic applied to predictor-classifier architectures. More specifically, both

works establish a metric about the time interval between prediction and failure occurrence, named *early-detection score*, which corresponds to how early is it possible to predict the fault in their predictor case. In addition, Nalepa et al. (2022) introduce the *Dice coefficient* to measure the overlapping between the predicted signal and the actual one, as well as a new definition of True Positives based on the predicted and actual overlapping intervals.

To the best extent of the author's knowledge, none of the techniques exposed about time series anomaly detection has been integrated to jointly operate within the onboard FDIR subsystem, especially the European PUS-based FDIR as it is exposed in Section 1, leaving a consistent gap in the state of the art. The gap mainly concerns how to interface the AI output with the rest of the PUS chain, which means assessing the capability of the AI to perform Failure Detection, Isolation or ultimately include Recovery. Interfacing the AI and the PUS-based FDIR is a crucial step of the present research, together with the more general system engineering aspects of setting requirements to the AI design and functionalities.

Furthermore, while the work of Nalepa et al. (2022) and Mathonsi and Zyl (2022) presents intervals-based evaluation of their predictor-classifier algorithms, another gap in the state of the art is identified in the application of this evaluation approach to direct-classification algorithms, and how it can be made specific for the case of integration alongside PUS-based FDIR.

3. Fault injection and dataset generation

In the present study, the generation of the dataset to be subject to anomaly detection utilizes components of the AOCS Offline Simulation Environment (AOSE, Wiedermann et al. (2009)) employed in the Astrone KI project, developed within Airbus Defence and Space GmbH. The simulator includes proprietary models of sensors, actuators, dynamics, attitude determination and control algorithms, as well as a PUS-based FDIR module which is used to test the proposed AI-based FDIR functionalities. The vehicle is assumed to fly over a surface derived from the full asteroid model. This simulation includes the accelerometer and gyroscope models, which in turn yields accelerations and angular rates. Fig. 1 provides an overview the simulation environment, highlighting inputs and outputs of the sensors.

A number of random monodirectional trajectories are simulated over equally spaced yaw angles and departing from the same starting point, limiting the flight distance to approximately 100 meters and the maximum height to approximately 10 meters. (Fig. 2). These trajectories are designed to include *take-off*, *cruising* and *landing* phases, as representatives of the three main phases of the relocation maneuvers performed by the vehicle. In addition, the trajectories cover a representative range of flight conditions and are deemed appropriate for the learning of the CNN described at Section 4.2. Clearly, real trajectories can have

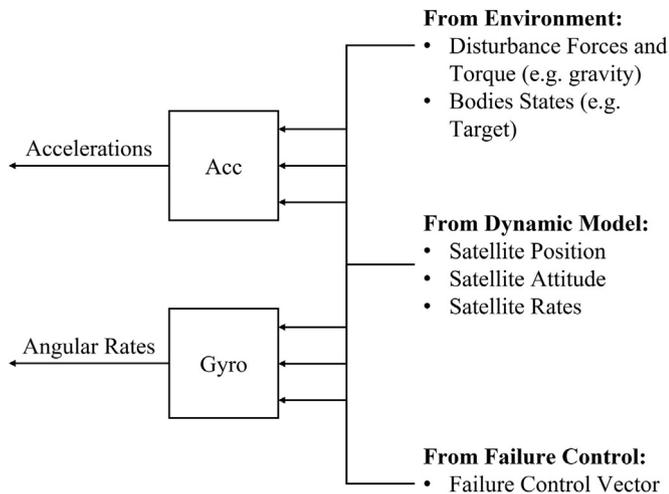


Fig. 1. Sensors simulation environment scheme.

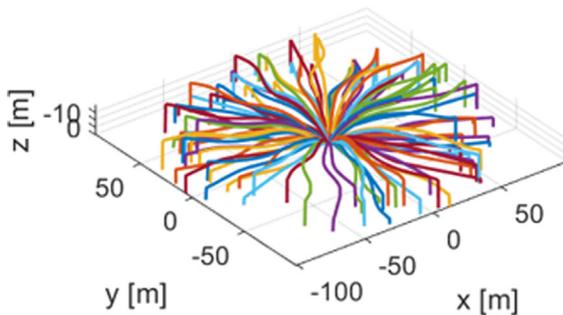


Fig. 2. Randomly generated trajectories.

different flight durations and assume more or less complex shapes, but from the point of view of the network learning there is no difference between these realistic flights and the artificial trajectories used in the dataset.

The fault injection is commanded by a vector of bits (*Fault Control Vector*) establishing both the fault to inject and the injection time sample. Injected faults comprise two main cases: *Stuck at last value* and *Stuck at random value*, where *last value* is the one assumed by the signal at the fault occurrence, while *random value* is any other value but the last one, either within or beyond the nominal range. Targeted fault cases are also distinguished by axis occurrence and presence or absence of measurement noise on top of the fault (Fig. 3). Specifically, the fault can interest all the axes at once or a random one. The decision between the two cases and the random axis selection is made at the injection stage.

The process of dataset building has been oriented by a set of requirements to guide the subsequent learning of the network. The established requirements act on the injected faults and include: randomized time sample of occurrence, specific boundaries of duration of the faults and interval of time distance between subsequent faults. The randomized time sample for fault occurrence is representative of the possibility of the fault to occur at any time

during operations. The minimum injected fault duration is set based on expert considerations on realistic duration of stuck values, balancing the need to give time to the AI to recognize the fault and trigger a PUS-based recovery. The maximum duration is also empirically set to be able to insert a high number of faults in the simulations, in order to focus the AI learning on early fault detection. The time interval to separate two successive occurrences of the faults is set within the same sensor and across both ones, in order to be compliant with realistic faulty situations: on the one hand, the time for a previous recovery action to occur in the single sensor shall be kept into account, on the other hand it is very unlikely that two faults occur simultaneously in both sensors. Table 1 shows the numerical values of the fault injection parameters.

The *Maximum Injected Fault Duration and Distance Between Subsequent Faults* in Table 1 are also constrained by the *Widows Length* parameter in Table 4, derived from the optimization process described in Section 5.3. This constraint will be further detailed after a better description of the CNN architecture is provided.

The generated dataset is made by approximately 28 millions data points and 6 features, which are made by the components of the signal along its axes (3) and their derivatives, numerically computed. Figs. 4 and 5 provide examples of the two sensors data and the injected fault cases.

4. Fault detection

4.1. Data analysis

Special attention has been given in this work to proper data scaling as it plays a particularly relevant role in this study, besides being a well-recognized practice in AI literature (Raschka and Mirjalili, 2019).

Due to the presence of stuck at random value faults, the dataset can extend far beyond the nominal operational range of the accelerometer and gyroscope signals, ultimately approaching infinity. Hence, the resulting dataset is characterized by a large portion of data in proximity of the nominal range of the signal, with a reduced percentage of outliers assuming different random values. This spread of the dataset over a wide range of values is generally leading to deterioration of the Neural Network detection performances, unless a proper scaling is applied.

A comparison of different scalars from the *scikit-learn* Python library was carried out with the objective of filtering the outliers from the nominal range of the signal, trying to provide a clear separation between the two and consequently enhance the fault detection. At the same time, an excessive shrink of the unfaulty signal shall be avoided, because it may deteriorate the performances of the Network in recognizing faults within its nominal range, especially stuck at last values.

A simple standardization of the entire dataset distribution would lead to excessive flattening of the nominal-range data near the zero line, as all data are centered and

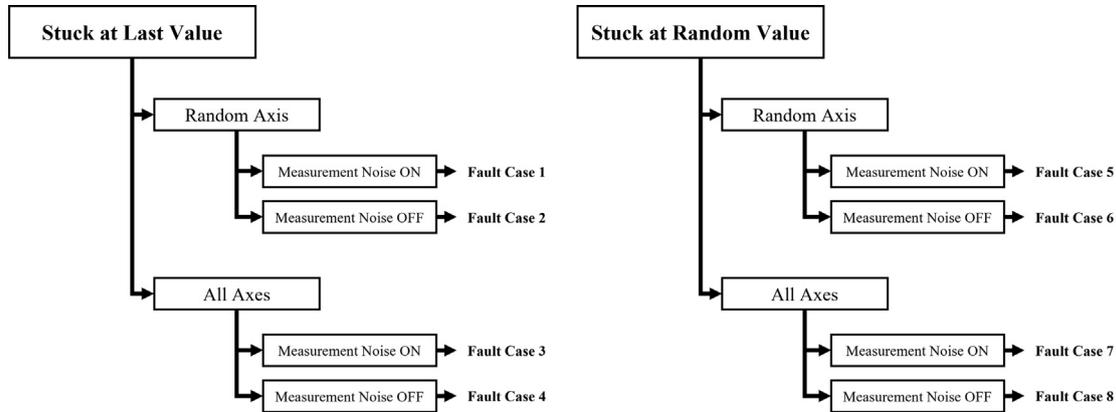


Fig. 3. Injected fault cases.

Table 1
Fault injection parameters.

Parameter	Value (samples)
Minimum injected fault duration	30
Maximum injected fault duration	110
Distance between subsequent faults	305

scaled based on their standard deviation, which is spoiled by the presence of infinite-like values. Hence, although a clear separation between the infinite-like values and the nominal is maintained, the stuck value cases other than these extreme values and the nominal signal are bounded within a range of a few orders of magnitude, leading to a consistent drop of the capability to distinguish one from the other, especially when noise is involved. To address this issue, *scikit-learn*'s *RobustScaler* has been selected to especially exclude infinite-value faults from the scaling computation. This scaler disregards a portion of the dataset when computing scaling factors by only keeping a specific quantile range, which means a percentage of the sorted data.

In the *RobustScaler*, setting an upper bound to the quantile range is useful to exclude infinite-like values from computation, but may also help with those faults where the stuck value slightly deviates from the nominal range. This consideration applies equally to both the signal and its derivative, which is also a feature in the input of the neural network, as detailed in Section 4.2.

Similar reasoning applies to the lower bound of the quantile range. Excluding the lowest values from scaling is particularly beneficial in the derivative of the signal, which sharply drops close to zero when a stuck value occurs, and thus can be adequately distinguished after the scaling stage.

Fig. 6 presents a comparison between three scaler setups. In the first column, the distribution of the dataset is fully standardized, especially clarifying in the gyroscope case the shrink of the nominal range with respect to the random values. The second column represent the default setting of *RobustScaler* from *scikit-learn*, which shows a consistent improvement with respect to the previous case, bounding

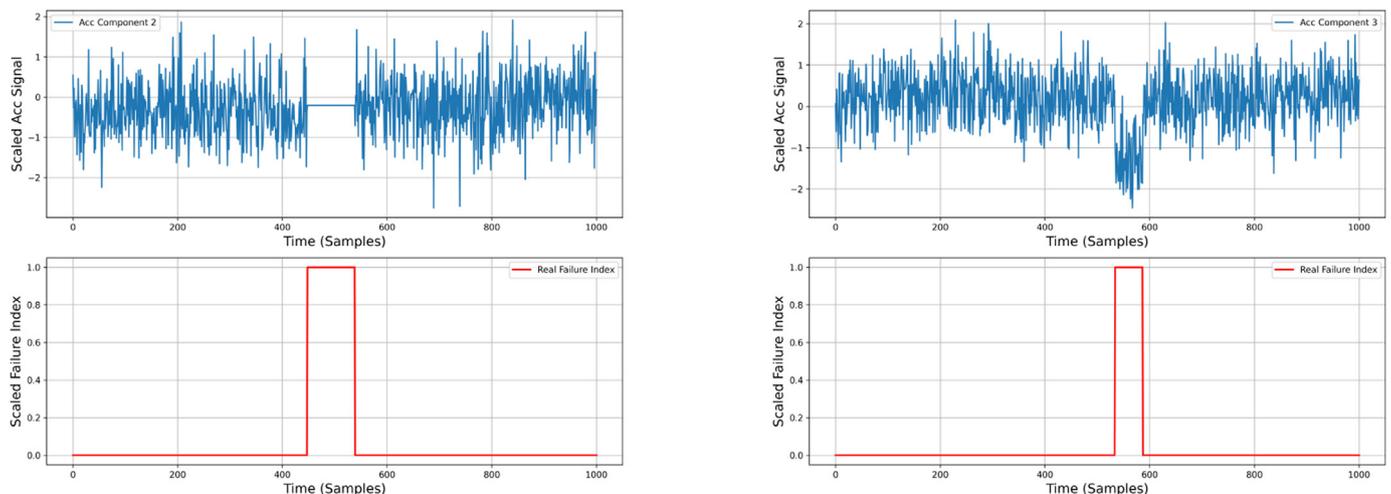


Fig. 4. Examples of faults injected in the accelerometer data (top) and relative index commanding the injection (bottom). Stuck at last value on single axis, without noise (left) and with noise (right). Plots are scaled according to the scaling method described in Section 4.1.

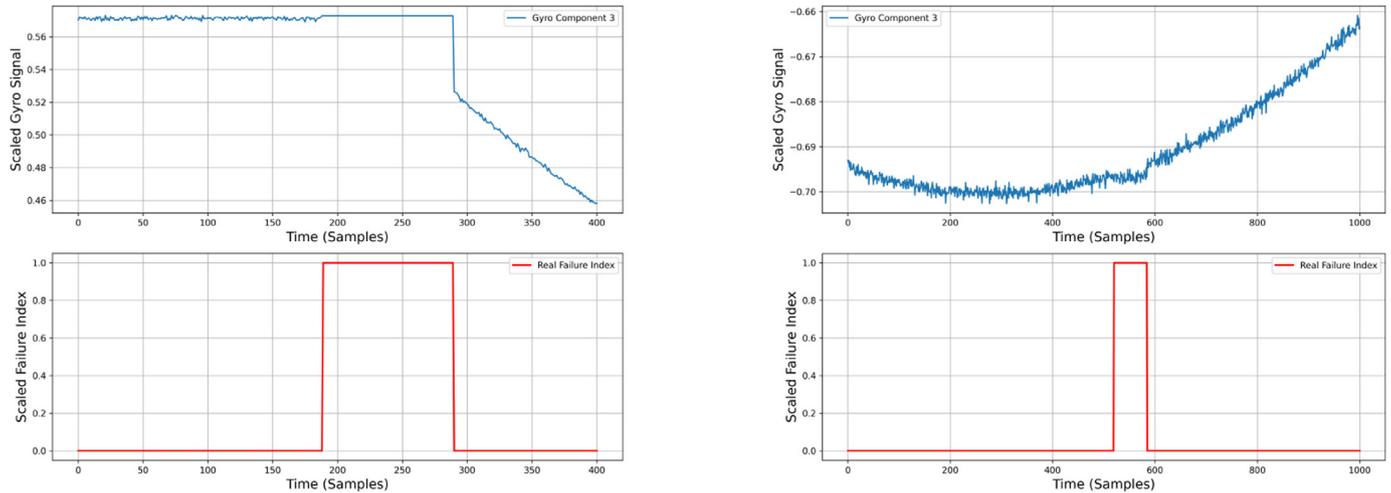


Fig. 5. Examples of faults injected in the gyroscope data (top) and relative index commanding the injection (bottom). Stuck at last value on single axis, without noise (left) and with noise (right). Plots are scaled according to the scaling method described in Section 4.1.

the Accelerometer signal in the $(-1, 1)$ range, widening the nominal range of the gyroscope signal with respect to the outliers, and enhancing the separation of the faults in the two sensors from the rest of the data. The last column utilizes the quantile range obtained after the optimization described in Section 5.3. The selection of the quantile range as a hyperparameter reflects the crucial role of the scaling in this particular use-case to maximize the detection performances, leading to the decision to leave to the optimizer the task to provide the optimal value of this range, more than choosing it by trial and error.

4.2. Convolutional neural network

For the FDI task expressed in the present study, a Convolutional NN architecture has been preferred over other

state-of-the-art algorithms. As described in Gallon et al. (2024), the onboard framework requires a lightweight model which can provide fast inference. CNNs usually require less parameters to accomplish the same task than other commonly used state-of-the-art time series analysis methods (e.g. RNNs). Plus, CNNs provide excellent parallelization capability of the operations involved. Finally, while RNNs are specific for the analysis of temporal trends in time series, in this specific study the use of CNNs for the same task is possible by convolving the signal on its temporal dimension to let the algorithm recognize the temporal trends associated to the stuck value faults.

The CNN performs binary classification of input data into "fault" and "no fault", thus not giving any information on the occurring fault case, axis of occurrence and noise presence. This decision was favored over its multi-

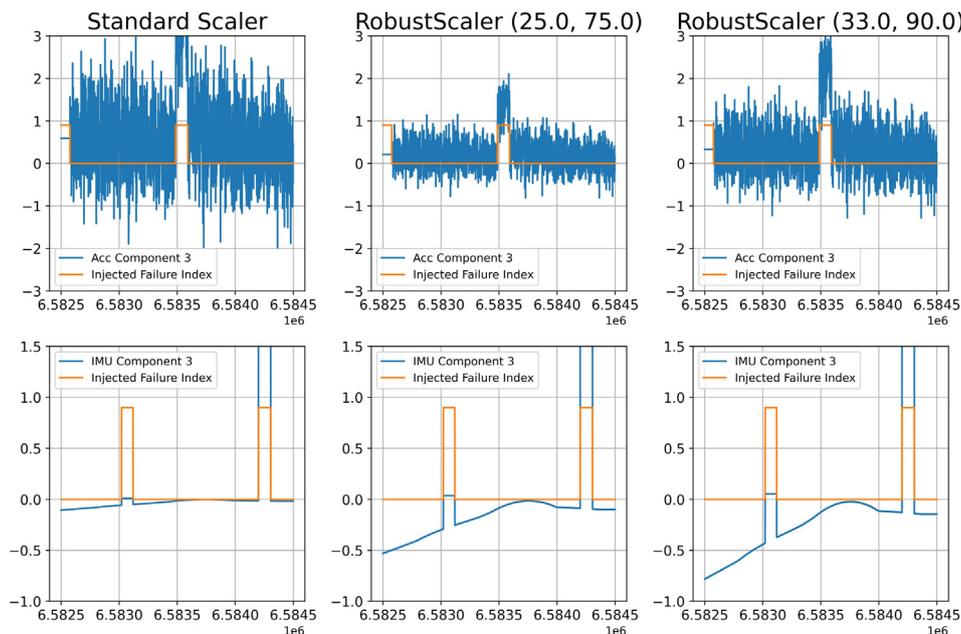


Fig. 6. Comparison of different scalers setups and their effect of sample stuck at random value faults.

class counterpart because, from the perspective of FDIR, there is no distinction among the recovery actions associated with the nature of the injected stuck value cases. However, the classification task must be multi-target, meaning that fault/no fault predictions must be obtained for both sensors under analysis. This is crucial for understanding where the fault is occurring and directing the recovery to the correct sensor.

The network structure must consider the interconnection between the signals of the two sensors, making it necessary to include a joint signal processing step. This ensures that complex scenarios requiring a comparison of the two signals to distinguish a fault are appropriately handled. For instance, a straight flight may result in a flat gyroscope signal, which could be misinterpreted as a stuck value if considered alone without cross-referencing the accelerometer output, which would instead be changing due to local gravity variations.

Inspired by the architecture proposed in Cui et al. (2016), this work proposes a multi-channel convolutional network where the input time series come from the two mentioned sensors, departing from the original idea of Cui et al. (2016), who proposed different manipulations of the very same time series as input. The output is instead made by two different failure indices, again corresponding to the two sensors in analysis (Fig. 7).

The neural network takes input from a sliding window applied to the sensor signal, continually updating with new samples received from the simulator interface. This window comprises the current sample along with a fixed-length history of past samples, aiming to detect the evolution of a stuck value, which typically requires multiple samples to realize over time. In addition, the input window consists of six features: three representing the signal itself and three representing the derivative of the signal, computed at each time sample. The derivative addition aims to improve the recognition of stuck values within the nominal signal as it presents distinctive features in the presence

of these faults. Specifically, the derivative tends to approach approximately zero whenever the fault starts: a precise zero value is obtained in the case of noise absence, while a more oscillating value is observed when noise is added. Also, specific spikes in the derivative can be observed for stuck at random values, whenever the signal jumps to a very different value in the frame of two subsequent samples.

The network structure is meant to emphasize interpretability, allowing for insights into which layer or hyperparameter influences specific output behaviors. On the one hand, each convolutional stage employs 1-dimensional convolutions, analyzing each feature independently, reflecting the independence of each component of the signal and the manifestation of the fault on different axes. On the other hand, the distinction between separate channels and a joint one enables the network to learn features from individual signals independently and capture connections between in a later stage. This approach facilitates the separate feature extraction from each signal in the initial stages and joint classification in subsequent layers following concatenation. Consequently, if the network underperforms on a particular sensor, tuning parameters in the associated branch suffices. Conversely, if the network struggles to detect faults where analysis of a single signal is insufficient, adjustments are made to the joint convolutional layer.

4.3. Integration with onboard FDIR

The requirement of integrating the AI side-by-side with the PUS chain, coming from the project, led this study to consider different integration possibilities, which can eventually be linked to two main categories, either processing the AI failure index as an Event or as a pMon. The main difference between the two is that in the first case a single failure prediction from the AI is straightly translated to a recovery action, while in the second case the same predic-

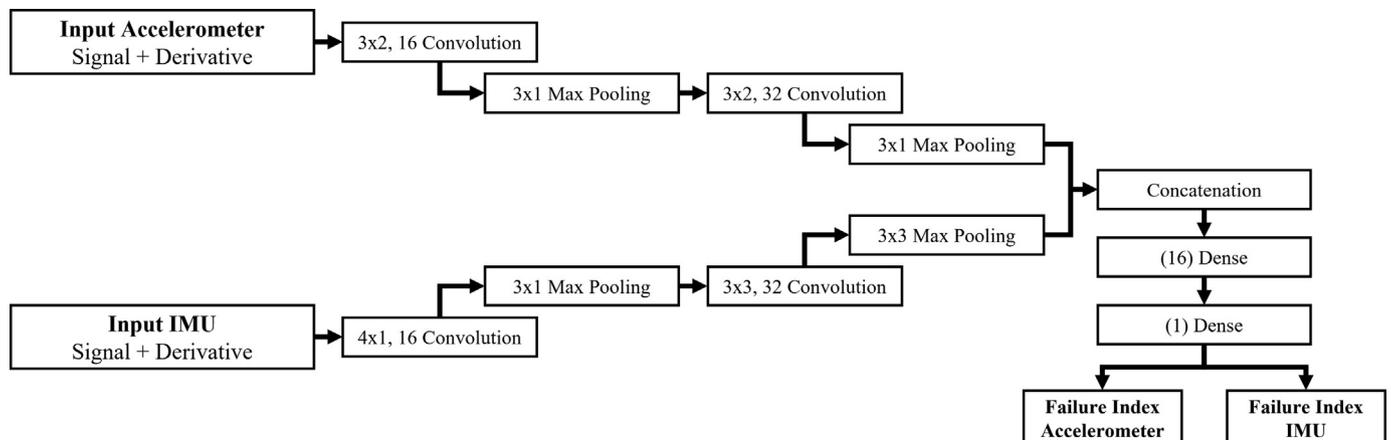


Fig. 7. Structure of the Multi-Channel CNN.

tion is first monitored as a pMon having its associated fMon, which eventually triggers the Event and the recovery afterwards.

In this specific case, the AI output has been decided to be processed as a single pMon with a single associated functional monitoring, with the expected value of the pMon set to zero. Since the output index is a Boolean value, the pMon is triggered whenever the AI output predicts a non-zero value (failure), and the associated fMon is triggered accordingly.

This approach offers several advantages. First, it enhances the robustness of the method to outliers compared to directly linking the prediction to event triggering. This is because it incorporates the persistency check between the failure prediction and event triggering. Additionally, it allows greater flexibility in implementation, particularly regarding the infrastructure surrounding the management of pMons and fMons. Indeed, the disabling of a pMon or fMon is a mechanism already embedded in the PUS-based FDIR (ECSS, 2016), which can be reused for the AI output pMon, therefore enhancing its dependability as it is relying on well-established onboard procedures.

5. Performances evaluation

Building on the research of Nalepa et al. (2022) and Mathonsi and Zyl (2022), this study shares the objective of evaluating the CNN’s ability to detect intervals rather than individual samples. However, unlike those works, this research does not explore intervals within the context of a signal reconstruction architecture. Instead, it addresses the classification of entire faulty intervals. Evaluating whether the CNN correctly classifies an interval is crucial, especially concerning the triggering of reactions in onboard FDIR through the mechanism of *persistency* of positive failure predictions (see Section 1). In this context, the developed set of metrics serves to assess both the capability of the AI algorithm to detect faults (*Detection Metrics*) and the effects of this detection on the onboard FDIR subsystem (*System Metrics*).

5.1. Detection performances

The Detection Metrics only assess the performance of the algorithm. Effectively substituting the classic *precision*, *recall* and *F1-score* metrics used to evaluate classification tasks, the set of metrics proposed in this work is meant

to evaluate the capability of an algorithm to detect faults by measuring case-specific quantities that are more representative in the specific AI-based onboard FDIR framework.

Before properly defining the set of metrics, a few initial definitions need to be given.

The labels vector y and the predicted labels vector y_{pred} are divided into intervals of True (fault) and False (no fault) values. These intervals are referred to as *faults* or *fault intervals*. A *Missed Fault* occurs when there is a complete interval of disagreement between y and y_{pred} , meaning that the algorithm misses the whole fault. On the other hand, if the algorithm detects a fault with a positive prediction immediately after an interval of disagreement between y and y_{pred} , it is referred to as a *Delay* in prediction. Situations where y_{pred} shows oscillations in the prediction while y is consistently faulty are called *Uncertain Prediction*. According to the given definitions, the Detection Metrics are presented in Table 2.

5.2. System performances

Related to the integration of the AI algorithm in a much broader system (i.e. the rest of the spacecraft), the system performances are related to the capability of impacting the components they interface with. More specifically, the AI-based FDIR is meant to exchange telemetry and telecommands with e.g. the OBC, the sensors themselves, etc. From some of these components it only receives data, from the other it establishes a bilateral link and is capable of triggering recovery actions to react to the detected failures. Therefore, the definition of System Metrics is tightly bounded to the integration strategy of the AI algorithm inside the PUS-based onboard FDIR subsystem.

In order to evaluate the performances of the architecture as defined in Section 4.3, a redefinition of the well-established concepts of Precision and Recall was carried out, starting from True Positives (TP), False Positives (FP) and False Negatives (FN), which are now reinterpreted in terms of the recovery actions triggered by the AI output within onboard FDIR. On one side, a predicted fault longer than the persistency time frame and with no real fault subtended is a FP because it triggers an unnecessary recovery action. Conversely, a real fault where no failure prediction is present constitutes a FN because it triggers no reaction. Besides these two extreme cases, it can also happen that a failure prediction with a real fault subtended does not meet the persistency duration require-

Table 2
Detection metrics.

Metric	Type	Objective	Description
Missed Faults Score	Index	→ 0	Percentage of Missed Faults over total number of faults in the dataset
Prediction Delays	Distribution	Shrink	Distribution of the Delays duration over the predicted fault intervals
Uncertain Prediction Score	Index	→ 0	Percentage of faults with uncertain prediction over total number of faults in the dataset
False Positives Duration	Distribution	Shrink	Distribution of the duration of the predicted false positives

ment to trigger a reaction, therefore it has no impact on the system level and does not take part to the System Metrics computation. This case is particularly present in situations where the model is not well optimized, leading to high prediction delays which push the fault detection too late after the actual start, or leading to fragmented failure prediction within the same real faults where all the single fragmented prediction intervals are not able to reach the persistency threshold. Table 3 summarizes all the defined System Metrics.

It is important to note how each of the metrics defined in Table 3 depends on the choice of the persistency value, which is the crucial parameter to determine the triggering of a recovery action. The tuning of the persistency value is therefore the strategy to get the maximum from the System Metrics, which show a certain correlation with the Detection Metrics, as long as they reach an acceptable value. More specifically, an algorithm which is capable of reaching high detection performances likely has also good system performances, because good detection means that most of the faults are detected and the prediction is also smooth, allowing a more accurate reaction triggering based on the specified persistency value. In this best-case scenario, the two set of metrics are strongly correlated. However, taking into account the presence of outliers and false positives in general, delays and so forth, increased robustness to adverse reaction triggering is given by the persistency

tuning. An appropriate tuning can first get rid of single outlier intervals lasting a few simulation samples, but also supply to those cases of uncertain prediction where the fragmentation of the prediction can be faced only by an appropriate persistency value. These considerations allow to pivot around non-perfect system performances and obtain the target results, being also an example of how the System Metrics are partially independent from the Detection Metrics. A summary of the correlation cases of Detection and System Metrics is presented in Fig. 8.

5.3. Optimization

To understand how Detection and System Metrics are utilized for evaluating a specific AI-based FDIR solution, it is crucial to emphasize the underlying requirements. From the perspective of the FDIR subsystem, minimizing false positive-triggered reactions is paramount, as these could affect mission availability, potentially leading to the activation of safe mode. Conversely, while detecting faults is important, it is less critical compared to minimizing false positives, given the nature of the examined scenario. Indeed stuck values can always be indirectly identified by conventional FDIR methods through the propagation of their consequences over time. Hence, even if the AI is not perfect in detecting these faults, they can eventually be identified by PUS as time progresses. This mechanism is

Table 3
System metrics.

Metric	Type	Objective	Description
Reaction Precision Score	Index	→ 1	$\frac{TP}{TP+FP}$
Reaction Recall Score	Index	→ 1	$\frac{TP}{TP+FN}$
False Positives Percentage	Index	→ 1	Percentage of False Positive Reactions over theoretical number of triggered reactions in the dataset

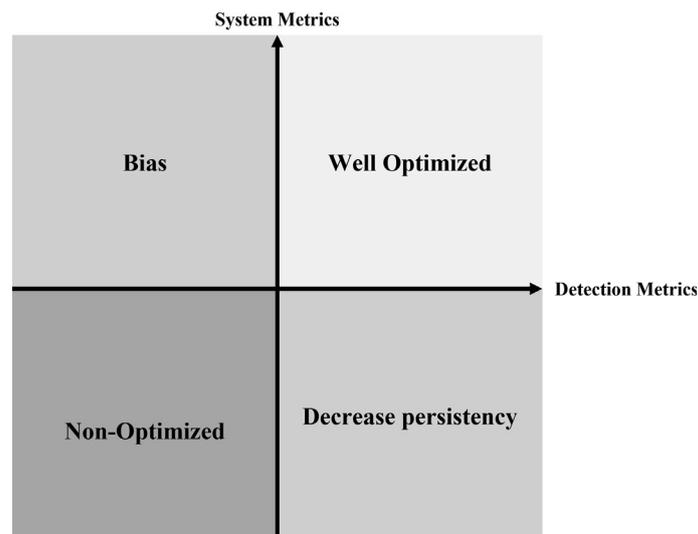


Fig. 8. Correlation scheme of Detection and System Metrics.

remarkable from a system engineering point of view because, since it is clear that an AI-based FDIR solution is not enough dependable (rather experimental) to be applied to an operational scenario, the stuck value faults provide a framework to prove the effectiveness of the AI on a critical functionality, without loading it with critical decision power.

The optimization of the specific AI-based FDIR solution has to focus on the System Metrics, which give a quantitative shape to the requirements mentioned above. However, the Detection Metrics also have to enter the optimization process by being used to double check if the value reached in the system performances is realistic or is subject to bias. Algorithm 1 shows the complete optimization procedure.

Algorithm 1. System metrics optimization

```

1: Start from an Established CNN Architecture
2: for Different Hyperparameters Sets do
3:   for Different Quantile Range of Scaling do
4:     for Different Persistency Values do
5:       Compute the System Metrics
6:       Compute the Objective Function (Eq. 1)
7:     end for
8:   end for
9: end for
10: Retrieve Optimal Parameters Value
11: Tune the Persistency to Optimize the System
    Metrics
12: for Persistency Values in an Interval Around the
    Optimized Value do
13:   Compute the System Metrics
14:   Check Requirements Compliance
15: end for
16: Double-check Detection Metrics
17: Freeze Persistency Value and Other
    Hyperparameters

```

The introduction of persistency in the hyperparameter tuning loop is related to the need to compute and optimize the System Metrics, but is not conceptually against the independence of the System Metrics from the Detection Metrics. Indeed, the independence embedded in the design will be reflected in the optimization loop with the persistency value being optimized on its own, not depending on the rest of the hyperparameters, therefore making identical the process of optimizing the persistency aside the main loop or inside it. In addition, once the algorithm has given the optimized persistency value as output, the task of tuning the parameter for maximizing the system performances is already embedded and may eventually need minor adjustment based on the specific requirements.

Concerning the last step of Algorithm 1, the need to double-check the Detection Metrics at the end of the whole process arises from the idea that biases in metrics could

occur if the algorithm excels in detecting long-lasting faults but struggles with shorter ones or sparse outliers. The optimization process may then lead to a high value of persistency following the long-lasting well-detected faults and even reach very good performances, while a check of the Detection Metrics would potentially show a high rate of Delays, False Positives, and Missed Faults. The check of the Detection Metrics is therefore needed in this case to identify this bias, before making considerations on its criticality. This specific consideration can be seen as an argument against putting the persistency inside the tuning loop in order to avoid the mentioned bias. However, this choice would have prevented the use of the System Metrics as optimization target, which need the persistency to be computed, leading to a conceptual inconsistency.

Based on the importance that the *precision* acquires with respect to *recall* from the system perspective, the objective function is chosen as a generalized F-beta score, shown in Eq. 1.

$$F_{\beta} = \sum_{i=1}^{n_{out}} w_i \frac{(1 + \beta^2) \cdot Pr_i \cdot Re_i}{(\beta^2 \cdot Pr_i) + Re_i} \quad (1)$$

In Eq. (1), the index i ranges from 1 to the number of outputs in the Neural Network, in this case corresponding to the two sensors considered. Pr and Re indicate *precision* and *recall* respectively. w_i and β are two weighting factors, with the former weighting the contribution of the single Network output to the overall score, while the latter balancing the importance of precision and recall in the single terms composing the score.

For the specific objective function employed in this work, w_i and β have been set to 0.5, taking into account the equal weight of the two sensors on the overall result and the already mentioned importance of precision over recall.

5.4. Results

The training of the Neural Network occurs in a supervised way employing labels available from the fault injection stage. Due to the binary classification task, the *binary crossentropy* loss function is employed, together with *Adam* optimizer with learning rate subject to optimization, and an *EarlyStopping* callback is set with patience 5. The batch size is 32768. The rest of the hyper-

Table 4
CNN hyperparameters.

Hyperparameter	Value
Learning Rate	0.0006507411205516692
Training Epochs	32
Quantile Range Lower Bound (%)	33
Quantile Range Upper Bound (%)	90
Windows Length (samples)	180
Persistency (samples)	27

parameters are subject to the optimization process detailed in Algorithm 1 and are shown in Table 4.

It is remarkable to notice that the particular *Windows Length* obtained after the optimization is smaller than the *Distance Between Subsequent Faults* (Table 1). The choice of the latter has been specifically done to ensure that each window would include maximum one fault, while including sufficient samples to allow the CNN to gather significant information on the stuck value eventually present. In other words, the network needs to see sufficient samples history to recognize a fault, if present, but cannot see windows where multiple faults are present, since this situation is not realistic. This consideration is equally valid for both sensors. Hence, the specific value chosen for the *Distance Between Subsequent Faults* in Table 1 allows the optimizer to iterate in a significant range of windows length values, seeking optimal performances while ensuring the presence of one fault at a time.

The training phase showed no significant overfitting of the training data, while desirable performances are obtained on the validation dataset, which are shown hereafter. Based on the dataset building method (see Section 3), it is deemed sufficiently representative of the stuck value faults which may occur in the sensors, and a wider dataset would just refine the present results without significantly adding performance.

The validation dataset is generated by the very same Monte Carlo analysis described in Section 3. Detection and System Metrics evaluated on this dataset are presented in Tables 5 and 6.

Fig. 9 and 10 show the False Positives Duration distribution for the accelerometer and gyroscope case respectively, highlighting the value of 27 samples for the persistency.

Both the two distributions are skewed towards the left side of the persistency, indicating that this particular value effectively filters out most false positives, whether they are outliers or consistently long incorrectly classified intervals. Plus, such a value of the persistency poses itself below the lower bound of the injected fault duration, meaning that no injected fault is inherently neglected. In other words, the CNN is able to detect even the shorter injected faults. This outcome is not trivial, because if the algorithm had been not sufficiently expressive, the persistency could have been pushed back by the optimization algorithm to neglect shorter faults in favor of the longer ones, ultimately falling into the biases outlined in Section 5.2.

Moreover, the two False Positives distributions present different spreading over the range from 30 samples to

Table 6
Resulting system metrics, *persistency* = 27 samples.

Metric	Accelerometer	Gyroscope	Target
Reaction Precision Score	0.99	0.99	→ 1
Reaction Recall Score	0.95	0.85	→ 1
False Positives Percentage	0.0006	0.0057	→ 0

110, which are the boundaries of the injected faults duration. The difference of the spreading in the two sensors is explained by the different performances achieved after the optimization procedure, with the accelerometer performing better than the gyroscope. However, the samples falling beyond the vertical line indicate that there is a number of False Positive intervals which is not filtered out by the persistency, causing the deterioration of the System Metrics, as shown in Table 5.

Lastly, the Prediction Delays Distribution is shown in Fig. 11 and Fig. 12 for both the sensors.

The two distributions namely present a concentration of values below 5 samples, with different shapes, corresponding to those correctly detected faults where the AI inherently needs a few sample to detect the stuck value as it progresses in time. Increasing the Delays Duration, the spreads of the plot over the entire duration range of the injected faults suggests the presence of cases where faults are detected after much of their progress, even too late to achieve a correct triggering of the recovery action. The objective of the optimization is in general to avoid these cases, which means shrinking the distribution of delays towards a small range of Duration, to enhance the detection of most of the faults and consequently improve the System Metrics. However, the presence of delays is inherent in the model and cannot be totally deleted. This fact, considering also the persistency mechanism, leads to the conclusion that the shortest detectable fault for the presented solution shall last a number of samples equal to *minimum observed delay + persistency*.

Comparing the obtained results with other related resources in literature is not simple, as to the knowledge of the authors there is no published work which discusses in addition to the detection performance also the engineering aspects of integrating their work into the wider standardized FDIR ecosystem (i.e. the interplay of AI-based and PUS-based FDIR). Work related to stuck value detection in gyroscopes, but employing classical metrics for the evaluation of the AI performance, was done by Wang et al. (2024). Considering the evaluation process of this work, metrics that can be subject to comparison from this work are Reaction Recall Score and False Positives Percentage, in Wang et al. (2024) named *False Positives Rate* and *True Positives Rate* respectively. Table 7 reports the comparison between the results of the best model in Wang et al. (2024) and the present study.

The True Positives Rate metric on the gyroscope is higher in Wang et al. (2024) than in the present study.

Table 5
Resulting detection metrics.

Metric	Accelerometer	Gyroscope	Target
Missed Faults Score	0.0503	0.1381	→ 0
Uncertain Predictions Score	0.1082	0.0699	→ 0

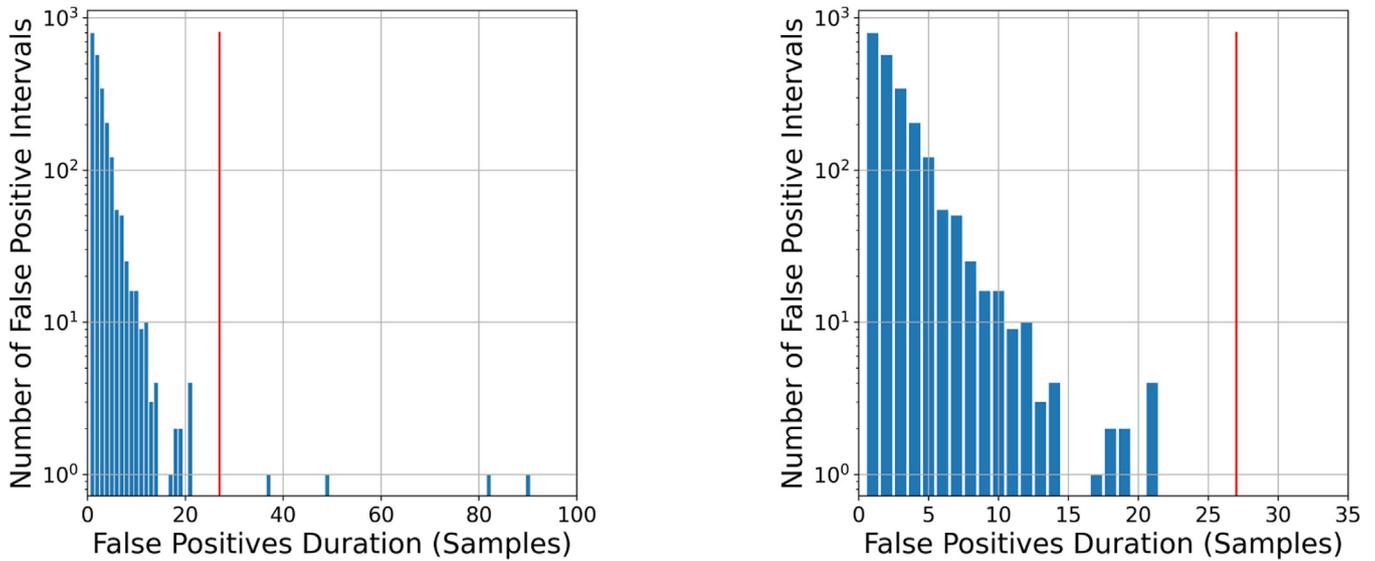


Fig. 9. Distribution of the False Positives Intervals for the accelerometer, persistency highlighted in red. Full duration range (left) and close-up (right).

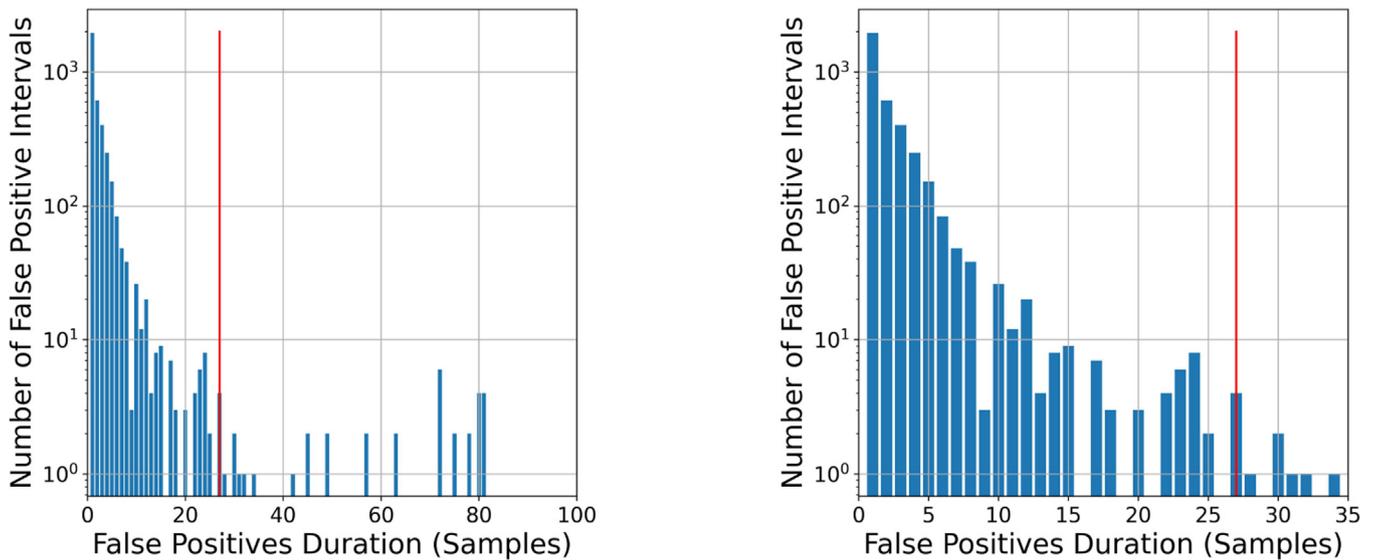


Fig. 10. Distribution of the False Positives Intervals for the gyroscope, persistency highlighted in red. Full duration range (left) and close-up (right).

The discrepancy derives from the Precision-focused optimization described in Section 5.3, preferred over a Recall-focused option. The False Positives Percentage is instead improved, as per the scope of the mentioned optimization. The authors are convinced that the careful considerations on the integration of novel AI-based FDIR solutions with classical PUS-based FDIR enables such advance on this specific metric.

Concerning the evaluation paradigm expressed in this work, an interesting parallel can be established with the work of Nalepa et al. (2022). The dataset employed in Nalepa et al. (2022) provides a number of different anomalies including but not limited to stuck values, therefore the task of fault detection is generally different from the ones exposed in this work. Nevertheless, the authors employ *sequence level evaluation* as expressed

in Section 5. Table 8 provides a comparison between the metrics considered in this study and the ones of Nalepa et al. (2022).

The min A-TP metric reported in 8 is expressed as the average of the A-TP of each fault, while this work presents it as a distribution of values (Fig. 11 and 12). This choice has been deemed more effective to graphically show the global behaviour of the model. As delays are also not employed in numerical optimization, expressing those in terms of mean or median would result in excessive information loss.

5.5. Hardware deployment experiment

In order to support the claim that the present work has taken into account the hardware deployment aspect and

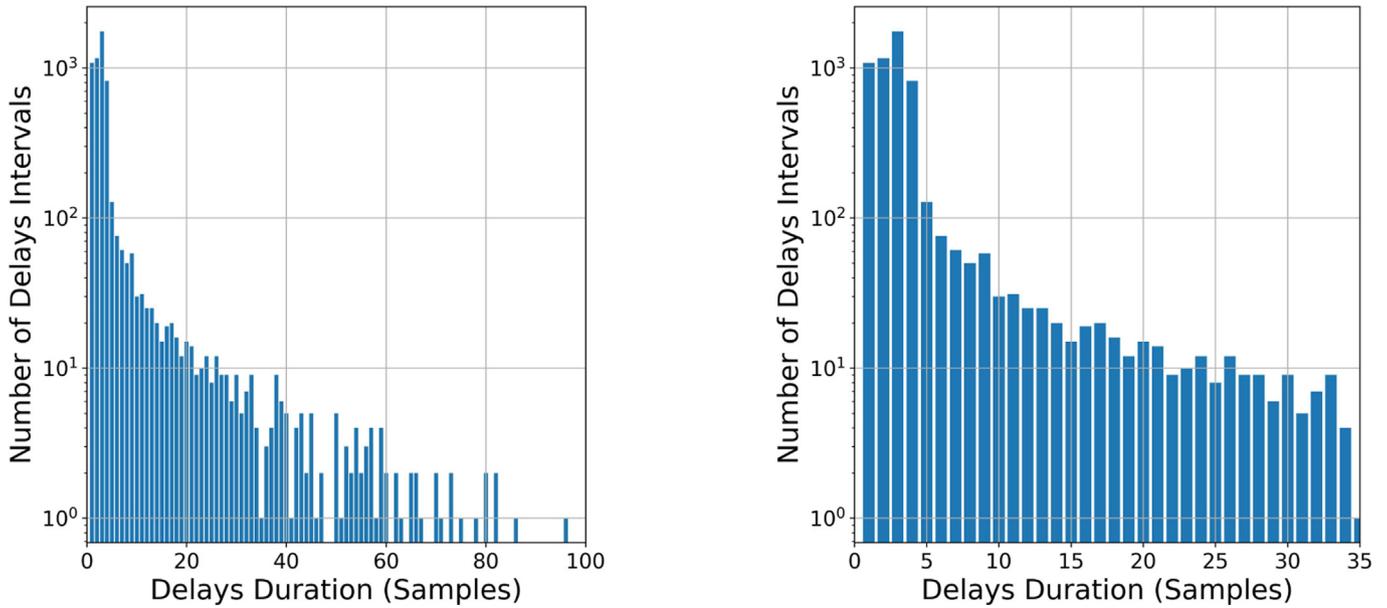


Fig. 11. Prediction Delays Distribution for the accelerometer case. Full duration range (left) and close-up (right).

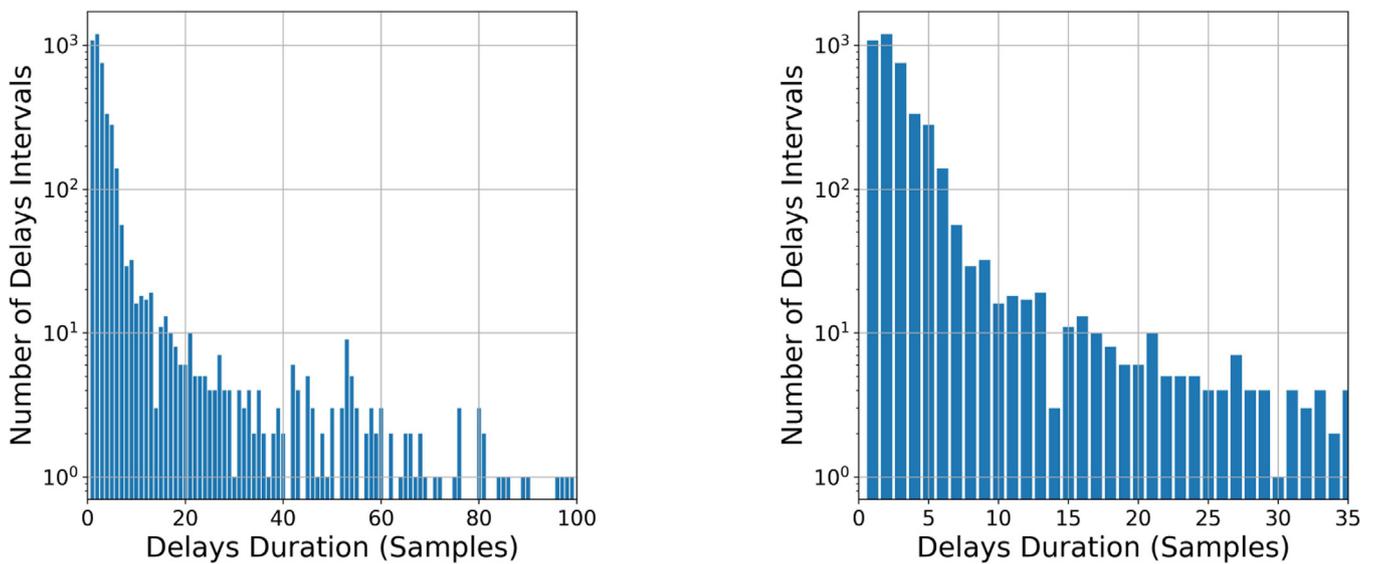


Fig. 12. Prediction Delays Distribution for the gyroscope case. Full duration range (left) and close-up (right).

Table 7
Metrics comparison with the ST-GATrans model from Wang et al. (2024), Table VII.

	Wang et al. (2024)	Present Study (Gyro)
False Positives Rate	0.011	0.0057
True Positives Rate	0.991	0.85

the real-time capability, the model was deployed on space-qualified hardware and the inference time was measured. For this experiment, System-on-Chips (SoCs) constitute a recognized solution for AI deployment in space (Furano et al., 2020; Ghiglione et al., 2021), in particular the *Xilinx Zynq Ultrascale+ (ZUS+)* and the *Xilinx Versal* boards. *Xilinx* provides proprietary tools, such as *Vitis AI*, which

are utilized to setup the workflow for the AI deployment and inference, in particular the model optimization and pruning and the management of processes to be run on the components of the SoC. Table 9 provides the result of the experiment.

The model weight after the pruning and optimization process drops from 414 kB on the local PC to the values shown in the Table 9, comparing to a total RAM value of 16 GB for both the boards. The performances of the model are double-checked after the deployment against the ones obtained on the local PC, both in terms of Detection and System Metrics (Table 6 and 5). The inference time is approximately 1 ms for both the boards. This value can be compared to the update rate of the Astrone KI sim-

Table 8
Metrics comparison with Nalepa et al. (2022), Table 3 and 4.

	(Nalepa et al., 2022)	Present Study
Before True Positive (B-TP)	Measures how many samples before the fault it is possible to predict its occurrence	Missing because it is proper of prediction approaches
After True Positive (A-TP)	Index measuring the delay from the first fault occurrence to when it is predicted, averaged for the fault duration	Distribution showing the delay per each fault
Early Detection (ED)	Averaged version of the B-TP using the duration of each fault	Missing because it is proper of prediction approaches
F1 Score	$2 \frac{Pr-Re}{Pr+Re}$	Same as Eq. 1 with $w_i, \beta = 0$

Table 9
Hardware deployment experiment results

Device	Model Weight	Execution Time
ZUS+	191 kB	0,000949 s
Versal	135 kB	0,001181 s

ulator, which is 0.1 s, or to classical rates of monitoring the accelerometer and gyroscope signal in onboard FDIR (<1 Hz). In both cases, since the execution time on the two devices is lower than the incoming data rate, the real-time capability is ensured.

6. Conclusions and future work

The different stuck value faults considered in the simulation comprehensively represent the various realizations of this class of faults in realistic scenarios on a functional level, making the presented solution applicable on different FDIR levels. The employed simulation method also claims to be representative of realistic occurrences of stuck values, based on RAMS analysis, to enhance the learning of the CNN which can more easily adapt to a potential deployment. Since the dataset construction methodology, detailed in Section 3, is focused on an early detection of the faults, it does not need to include realistic fault duration, thus enhancing the generalization of the detection to stuck values occurring at very different time scales, i.e. in very different components. The only limitation is in the end constituted by the earliest fault detection capability, expressed in the *Prediction Delays* metric (Fig. 11 and Fig. 12), which set the lower bound to the duration of the detectable faults. Clearly, this reasoning has to take into account the persistency value, which has to stay coherent with the targeted faults.

Based on the accelerometer and gyroscope of *Astrone KI*, the proposed dataset has been subject to fault detection via a Convolutional Neural Network (Section 4.2), which has proven its effectiveness in the diagnosis of the faults, achieving outstanding performances. The convolutions-based approach guarantees a partially interpretable algorithm which, in this particular case, allows a clear separation between the analysis of the single sensors and the joint-channel operations. Besides, it is particularly useful for the analysis of different time scales trends in time series, thus to address the stuck values faults presented in this work. Distinguishing a stuck derivative value behind the

white noise to classify it is the most obvious situation, that the present analysis faced, where convolutions served the scope of recognizing trends at different time scales.

A set of case-specific metrics is then proposed to evaluate the CNN in its fault detection task. The design of the metrics focused on the real-time scenario, as well as on the interval-based nature of the faults. The latter is an element of innovation because it detaches from the classic point-wise conception of the fault classification, adapting to a problem which is very common in the field of anomaly detection in time series. Consequently, the developed metrics add a further layer of complexity to the network output processing, but makes it way more interpretable than classic evaluation approaches (i.e. *F1-score*, *Precision*, *Recall*).

Finally, the deployment on two space-qualified FPGAs is presented, proving the compatibility of the model and achieving the same performances as the initial model. Besides, the inference time is proved to enable real-time data processing.

The present research shows promising baseline performances on the *Astrone KI* use-case and proves the feasibility of the approach to accomplish the specific stuck value fault detection task. Concerning possible future research topics, a deployment of the CNN onboard a flying mission shall be the final target to head to. To achieve this, the development shall first pass through the definition of application-dependent requirements (e.g. Precision and False Positives Percentage). These performance criteria shall guide the tuning process, as described in Section 5.3, thus be defined *a priori* based on the specific objectives of the mission, on the type of failure to be addressed and their required recovery actions. As a practical indication, a fine-tuning of w_i and β in Eq. (1), to obtain performances tailored to the specific needs is recommended. Plus, a sensitivity analysis over the mentioned parameters can also be an asset to evaluate the optimality of the obtained solution and enhance its robustness. Concerning the deployment of the presented model on space-representative hardware (e.g. FPGAs), room for future work is left to prove its compatibility with other field-specific edge devices in real-time scenarios and that it is able to achieve comparable performances to the theoretical ones shown in the present work. Most importantly, the hardware shall be tested in a wider simulation environment, e.g. a simulator or an assembly of real components, to confirm the theoretical outcome of respect of the real-time requirements.

Finally, another promising stream of future research could be testing the robustness of the presented algorithm to accelerometers and gyroscopes coming from various manufacturers. This procedure would ensure the algorithm to fit various configurations, enhancing its reusability across different missions and use-cases.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The results presented in this paper have been achieved by the project Astrone - Increasing the Mobility of Small Body Probes, which has received funding from the German Federal Ministry for Economic Affairs and Energy (BMWi) under funding number "50 RA 2130A". The consortium consists of Airbus Defence and Space GmbH, Astos Solutions GmbH, Institute of Automation (Technische Universität Dresden) and Institute of Flight Mechanics and Controls (Universität Stuttgart). Responsibility of the publication contents is with the publishing authors.

References

- Alos, A., Dahrouj, Z., 2021. Using mlstm and multioutput convolutional lstm algorithms for detecting anomalous patterns in streamed data of unmanned aerial vehicles. *IEEE Aerosp. Electron. Syst. Mag.* 37 (6), 6–15.
- Azevedo, D.R., Ambrósio, A.M., Vieira, M., 2012. Applying data mining for detecting anomalies in satellites. In: *2012 Ninth European Dependable Computing Conference*. IEEE, pp. 212–217.
- Baireddy, S., Desai, S.R., Mathieson, J.L., et al., 2021. Spacecraft time-series anomaly detection using transfer learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1951–1960.
- Canizo, M., Triguero, I., Conde, A., et al., 2019. Multi-head cnn-rnn for multi-time series anomaly detection: an industrial case study. *Neurocomputing* 363, 246–260.
- Ciancarelli, C., Mariotti, E., Corallo, F., et al., 2023. Innovative ml-based methods for automated on-board spacecraft anomaly detection. In: *The Use of Artificial Intelligence for Space Applications*. Springer Nature, Switzerland, pp. 213–228.
- Cui, Z., Chen, W., Chen, Y., 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*.
- Di, Z., Shao, H., Xiang, J., 2021. Ensemble deep transfer learning driven by multisensor signals for the fault diagnosis of bevel-gear cross-operation conditions. *Sci. China Technol. Sci.* 64 (3), 481–492.
- ECSS, 2016. *Telemetry and telecommand packet utilization*. Technical Report ECSS-E-ST-70-41C European Space Agency.
- Furano, G., Meoni, G., Dunne, A., et al., 2020. Towards the use of artificial intelligence on the edge in space systems: challenges and opportunities. *IEEE Aerosp. Electron. Syst. Mag.* 35 (12), 44–56. <https://doi.org/10.1109/MAES.2020.3008468>.
- Gallon, R., Schiemenz, F., Krstova, A., et al., 2024. Machine learning-based vs deep learning-based anomaly detection in multivariate time series for spacecraft attitude sensors. In: *Proceedings of SPAICE2024: The First Joint European Space Agency/IAA Conference on AI in and for Space*, pp. 388–392.
- Gao, Y., Yang, T., Xing, N. et al. (2012a). Fault detection and diagnosis for spacecraft using principal component analysis and support vector machines. In *2012 7th IEEE conference on industrial electronics and applications (iciea)* (pp. 1984–1988). IEEE.
- Gao, Y., Yang, T., Xu, M. et al. (2012b). An unsupervised anomaly detection approach for spacecraft based on normal behavior clustering. In *2012 Fifth International Conference on Intelligent Computation Technology and Automation* (pp. 478–481). IEEE.
- Ghiglione, M., Raoofy, A., Dax, G., et al., 2021. Machine learning application benchmark for in-orbit on-board data processing. *European Workshop on On-Board Data Processing*. <https://doi.org/10.5281/zenodo.5520877>.
- Hundman, K., Constantinou, V., Laporte, C., et al., 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395.
- Kuhn, M., Johnson, K., 2013. Classification trees and rule-based models. In: *Applied Predictive Modeling* chapter 14. (pp. 369–413). Springer.
- Lakey, D., Schlippe, T., 2024. A comparison of deep learning architectures for spacecraft anomaly detection. In: *2024 IEEE Aerospace Conference*. IEEE, pp. 1–11.
- Li, G., Jung, J.J., 2023. Deep learning for anomaly detection in multivariate time series: approaches, applications, and challenges. *Inform. Fusion* 91, 93–102.
- Li, T., Comer, M., Delp, E., et al., 2019. A stacked predictor and dynamic thresholding algorithm for anomaly detection in spacecraft. In: *MILCOM 2019–2019 IEEE Military Communications Conference (MILCOM)*. IEEE, pp. 165–170.
- Liu, D., Pang, J., Xu, B., et al., 2017. Satellite telemetry data anomaly detection with hybrid similarity measures. In: *2017 International conference on sensing, diagnostics, prognostics, and control (SDPC)*. IEEE, pp. 591–596.
- Mansell, J.R., Spencer, D.A., 2021. Deep learning fault diagnosis for spacecraft attitude determination and control. *J. Aerospace Inform. Syst.* 18 (3), 102–115.
- Martin, M., Belien, F., Liu, B., et al., 2021. Astrone-gnc for enhanced surface mobility on small solar system bodies. In: *11th International ESA Conference on Guidance, Navigation & Control Systems*.
- Mathonsi, T., Zyl, T.L.v., 2022. Multivariate anomaly detection based on prediction intervals constructed using deep learning. *Neural Comput. Appl.*, 1–15.
- Meß, J.-G., Dannemann, F., Greif, F., 2019. Techniques of artificial intelligence for space applications—a survey. In: *European workshop on on-board data processing (OBPD2019)*. European Space Agency.
- Murphy, J., Ward, J.E., Mac Namee, B., 2021. Machine learning in space: A review of machine learning algorithms and hardware for space applications. *AICS*, 72–83.
- Nalepa, J., Myller, M., Andrzejewski, J., et al., 2022. Evaluating algorithms for anomaly detection in satellite telemetry data. *Acta Astronaut.* 198, 689–701.
- Nassif, A.B., Talib, M.A., Nasir, Q., et al., 2021. Machine learning for anomaly detection: a systematic review. *Ieee Access* 9, 78658–78700.
- Pang, G., Shen, C., Cao, L., et al., 2021. Deep learning for anomaly detection: a review. *ACM Comput. Surv. (CSUR)* 54 (2), 1–38.
- Raschka, S., & Mirjalili, V. (2019). Building good training datasets – data preprocessing. In *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2* chapter 4. Packt publishing ltd.
- Russo, A., Lax, G., 2022. Using artificial intelligence for space challenges: a survey. *Appl. Sci.* 12 (10), 5106.
- Savoir, 2023. *SAVOIR FDIR Handbook*. Technical Report Issue 2. Revision 12 European Space Agency.
- Tipaldi, M., Bruenjes, B., 2015. Survey on fault detection, isolation, and recovery strategies in the space domain. *J. Aerospace Inform. Syst.* 12 (2), 235–256.

- Wang, H., Yu, W., You, J., et al., 2021. A unified framework for anomaly detection of satellite images based on well-designed features and an artificial neural network. *Remote Sens.* 13 (8), 1506.
- Wang, S., Liu, Z., Jia, Z., et al., 2024. Fault detection for uavs with spatial-temporal learning on multivariate flight data. *IEEE Trans. Instrum. Meas.*
- Wiedermann, G., Winkler, S., Gockel, W. et al., 2009. Model based design of the sentinel-2 attitude control system. In: *Deutscher Luft- und Raumfahrtkongress 2009*.
- Xiang, G., Lin, R., 2021. Robust anomaly detection for multivariate data of spacecraft through recurrent neural networks and extreme value theory. *IEEE Access* 9, 167447–167457.
- Xiong, L., Ma, H.-D., Fang, H.-Z., et al., 2011. Anomaly detection of spacecraft based on least squares support vector machine. In: *In 2011 Prognostics and System Health Management Conference*. IEEE, pp. 1–6.
- Zhang, K., Wang, S., Wang, S., et al., 2023. Anomaly detection of control moment gyroscope based on working condition classification and transfer learning. *Appl. Sci.* 13 (7), 4259.
- Zhong, J., Zhang, Y., Wang, J., et al., 2021. Unmanned aerial vehicle flight data anomaly detection and recovery prediction based on spatio-temporal correlation. *IEEE Trans. Reliab.* 71 (1), 457–468.