SpaceOps-2023, ID # 350

Spacecraft Failure Detection, Isolation and Recovery using Artificial Intelligence: A Study for Implementation on the Edge

Filippo Ales[a]*, Alisa Krstova[a]*, Thomas Chabot[a], Max Ghiglione[d], Mario Castro de Lera[a], Florian Hegwein[a], Andreas Koch[a], Carlos Hervas Garcia[a], Prem Harikrishnan[b], Maen Mallah[c], Rashid Ali[c], Michael Rothe[c], and Laurent Hili[d]

[a] *Airbus Defence and Space GmbH, Claude-Dornier-Str., 88090, Immenstaad, Germany*
[b] *EVOLEO Technologies GmbH, Freibadstraße 30, 81543 München, Germany*
[c] *Fraunhofer Institute for Integrated Circuits (IIS), Am Wolfsmantel 33, 91058 Erlangen, Germany*
[d] *European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), Keplerlaan 1, Postbus 299, 2200 AG Noordwijk, The Netherlands*
* Corresponding Authors

Abstract

The aim of spacecraft Failure Detection, Isolation and Recovery (FDIR) is to guarantee the target of missions' reliability, availability, maintainability and operational autonomy in order to ensure its success despite the potential occurrence of failures. Classical FDIR methodologies require that all potential failure cases be identified during the spacecraft design phase, which can result in significant development and operational costs associated with the resolution of in-orbit anomalies which were not foreseen. It can turn out to be more (cost-)effective to have an on-board system which is learning from the data telemetry so that it can carry out the on-board monitoring activities with little to no prior knowledge of expected failures. Although a plethora of failure/anomaly detection strategies have been developed for time series analysis and used across missions, due to the ever-growing complexity of the spacecraft being built each year, the field of smart anomaly detection both on ground and on board remains challenging. An important limitation relates to hardware and computational resources (a LEON IV processor or space-qualified FPGAs have much less computational power compared to modern GPUs) and therefore adaptation of these techniques is necessary. This work provides a preliminary assessment of the performance of different machine learning techniques in detecting various failure scenarios as well as the specificities and challenges associated with the implementation of such techniques onboard a spacecraft.

Keywords: Artificial Intelligence, Failure Detection, Isolation and Recovery

Acronyms/Abbreviations

AOCS	Attitude and Orbit Control System
ADDICT	Anomaly Detection using DICTionary
AI	Artificial Intelligence
CCN	Convolutional Neural Network
CSW	Central Software
DPU	Data Processing Unit
FDIR	Failure Detection, Isolation and Recovery
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HKTM	Housekeeping Telemetry
HW	Hardware
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
MCL	Model Control Laws
MEO	Medium Earth Orbit
ML	Machine Learning
MPSoC	Multi-Processor System on Chip
OBC	On-Board Computer
TC	Telecommand
TM	Telemetry

1. Introduction

The aim of Failure Detection, Isolation and Recovery (FDIR) is to identify and implement the failure tolerance objectives, as required by the specified target of missions' reliability, availability, maintainability and operational autonomy (failure management), in order to ensure its success despite the potential failure occurrences. FDIR is embedded in the overall functionality of a spacecraft at unit, subsystem and system levels and relies on three basic architectural principles:

- The ability to notice that something non-nominal has happened (fault or failure detection, depending on whether we can observe the event itself or observe - by sampling - the change in the system or component state).
- The ability to isolate the fault or failure. For this, the system must be able to uniquely identify the fault or failure observed and prevent it from propagating and causing other faults or failures.
- The ability to recover in time from the fault or failure that has occurred.

The design of the FDIR includes a trade-off between the maximization of autonomy and mission availability on one side, and satellite design and validation complexity on the other side. Therefore, the design consists in the definition of the best fault-tolerance strategy and techniques, as required by the target reliability, availability and autonomy requirements from which a redundancy scheme is devised to cope with the same objectives.

The need for having an intelligent FDIR system that performs complex fault (anomaly) detection on-board is imposed by two major limitations of on-ground monitoring. The first comes from the fact that contacts between the ground and space segments are limited to assigned visibility windows whose size depends on the mission (e.g., a 10-minute window every 12-24 hours for Earth Observation missions to a 10-hour window after 2 weeks of no contact for deep space missions). Therefore, it is not possible to rely only on the ground segment to detect potential anomalies on-board, as some failures may require very fast recovery. The second constraint lies in the reduced data availability - due to the limited bandwidth available for data downlink, only a subset of the on-board data pool is downlinked using dedicated housekeeping telemetry (HKTM) packets. Hence, high-frequency anomalous transients would go unnoticed if one would rely only on on-ground processing. Consequently, although an on-ground implementation of (AI-based) anomaly detection algorithms would support operators in preventing certain failures, most of the on-board failure cases need to be detected, isolated and recovered autonomously on-board to ensure the safety and integrity of the mission.

A classical FDIR design strategy requires that all feared events and related failure modes are identified, made observable and that proper monitoring is associated with the observable. This approach implies that all possible failures are identified at a very early stage of the project and that all thresholds are defined and set correctly. While it is possible to implement a failure detection system for small architectures with few failure cases and observables, the task becomes increasingly complex for more advanced systems, leading to higher development and operational costs. Furthermore, as classical FDIR approaches also rely on global monitorings, in the case of a failure that could not be detected locally (i.e. at unit or function levels), a global reconfiguration action (typically Safe Mode) is triggered, which may have a negative effect on mission availability.

For these reasons, we turn to the usage of machine learning (ML) algorithms that can learn from telemetry data to perform on-board monitoring activities with little or no prior explicit knowledge about the underlying system. ML algorithms can be especially advantageous for identifying and isolating failures (anomalies), especially unforeseen ones, at the lowest level (equipment and subsystem level), thus enhancing mission autonomy and availability. The usage of these algorithms is expected to allow for reduced FDIR development costs, while improving failure detectability (an ML-based system should be able to detect a wider range of anomalies including unknown ones) and system reactivity (an ML-based FDIR should react to failures earlier than traditional FDIR algorithms that are based only on simple thresholds).

This paper is organized as follows: Section 2 presents a high-level architecture of an FDIR system based on AI algorithms. Section 3 presents the results of a preliminary analysis of the application of various AI algorithms for enhanced FDIR. Section 4 considers some of the development aspects related to the application of such algorithms on board. Finally, concluding remarks and directions for future research are given in Section 5.

2. Design of an ML-based FDIR system

To deploy real-time health monitoring of spacecraft housekeeping telemetry (at system as well as at unit/equipment level) and to take advantage of machine learning techniques, the ML model(s) will have to monitor hundreds, potentially thousands of TM/TC parameters. Therefore, the system will need to be integrated or connected to the spacecraft OBC in order to have access to the observables (e.g., sensor telemetry or CSW variables). This also enables the use of parameters that are usually hidden or only available on request from ground as, for example, high-frequency status of the AOCS Equipment, Model and Control Laws (MCL) outputs.

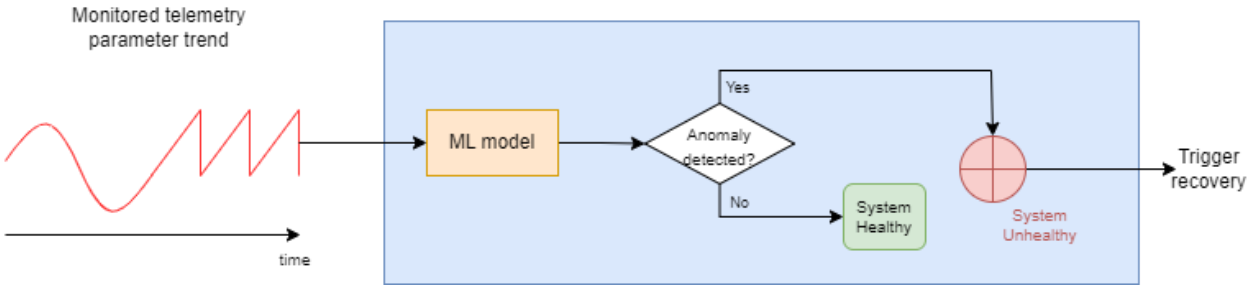


Fig. 1. High-level architecture of an AI-based FDIR system

Figure 1 illustrates the high-level workflow of the envisaged ML-based FDIR system (for simplicity, the workflow is illustrated for a single parameter, but is scalable to multiple parameters): the telemetry parameter is processed by a module which encapsulates the trained ML model for anomaly detection. To make the ML model compatible with the overall on-board architecture, it must be run on a space-grade hardware platform, e.g. an FPGA. If the model detects anomalous behaviour in the parameter data, the appropriate recovery action must be triggered.

Concerning the development of the ML model(s) responsible for the processing of telemetry data, there are multiple design choices to be taken into consideration with respect to the following aspects:

- **Data** - the availability of high-quality data which accurately captures the characteristics of the system being modelled is of paramount importance for the successful practical implementation of any ML model. For the task of ML-based FDIR, depending on the spacecraft equipment/subsystem to be modelled, possible sources of training data include simulation data, test bench data, real in-flight telemetry data (from missions of the same family/type) or a combination thereof. Other aspects to be considered include the possibility to provide accurate labels to known anomaly signatures for model evaluation, as well as how the data used for training the model corresponds to the actual telemetry observed on board (in terms of TM parameter availability, data rate, data types etc.).
- **Model** - the nature of the available data often dictates the use of a specific learning paradigm (supervised, semi-supervised or unsupervised). The model selection, training and testing phase is an iterative process where a series of experiments with different data pre-processing techniques, model types, model architectures and model hyperparameters is performed. The different model variants are evaluated and traded off based on predefined performance evaluation metrics (e.g., precision, sensitivity, specificity etc.) and the result of this phase is a model or collection of models which perform best for the given problem on the available data.
- **Hardware implementation** - as the end goal is to run the ML model on space-grade hardware to enable enhanced FDIR functionalities, the model artefact which has been trained on ground (usually using high-performance workstations or in the cloud) needs to undergo a series of transformations and adaptations so that it is compatible with the target hardware platform. This is usually done using dedicated AI inference development frameworks specific to the different target platforms (such as FPGAs). Some of the most popular development frameworks include Xilinx VitisAI [1], FINN [2] etc. The process of hardware adaptation usually involves optimizing the trained model artefact by means of pruning and/or quantization to reduce the memory and computation footprint of the model. The next steps involve model compilation and mapping to the specific hardware implementation and generating the bitstream HDL code used to synthesize the logic onto the hardware platform. Once the model deployment is finalized, a runtime API is responsible for the inference workflow, i.e., receiving the inputs, feeding them to the hardware platform, performing model inference and outputting the results. The aforementioned steps may slightly differ

between different hardware development frameworks, the choice of which is imposed by the target hardware platform and the model type, as different frameworks only support specific model architectures at the time of writing of this paper.

In the following sections, we present the results of the evaluation of different ML approaches applied to three different FDIR use cases. The use cases cover the implementation of several ML algorithms trained on data from different sources (simulation data, test data and in-flight data) and discuss the model efficiency in detecting the respective anomaly types. Furthermore, we present the outcomes of a prototypical hardware implementation of a ML algorithm applied to solar array telemetry data.

3. Analysis of an AI-based FDIR Algorithm Performance

The initial implementation of an AI-based FDIR algorithm was conceptualized as a de-risking activity with the goal to assess the applicability of machine learning to the problem of enhanced FDIR. For this purpose, two failure cases observed in previous missions were taken as the target problems to be modelled by a set of ML algorithms. To train and evaluate the efficiency and effectiveness of the algorithms in detecting the failure cases, datasets based on simulated and test bench data have been generated, covering realistic instances of anomalies which might occur in orbit.

3.1 Equation numbers

The first use case considers a "frozen gyro" failure, i.e. a failure where the angular rate provided by the onboard gyrometer freezes to a non-zero, constant but noisy value. This kind of failure is difficult to detect with classical FDIR methods. Two different machine learning methods have been prototyped and assessed: anomaly detection through Convolutional Neural Network (CNN) with supervised learning, and anomaly detection through a semi-supervised method called ADDICT (Anomaly Detection using DICTionary) that was developed in the frame of a CNES-Airbus doctoral thesis.

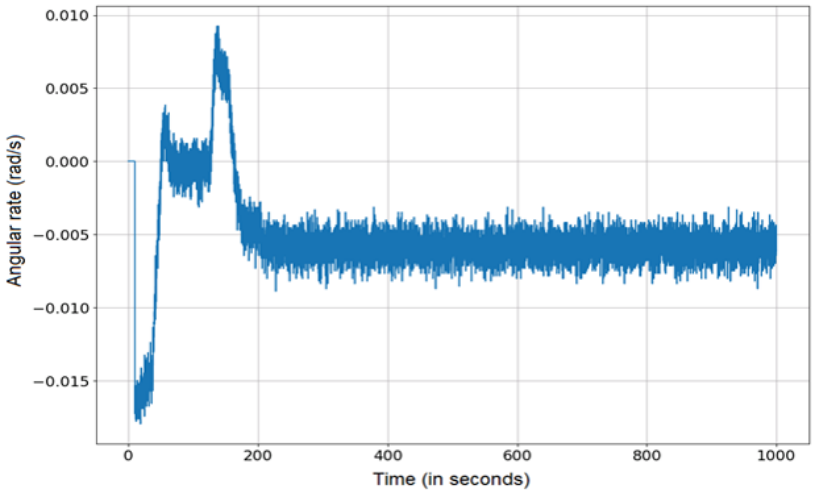


Fig. 2. Example of a frozen gyro failure

Data used within this use case were obtained from a simulator generating satellite dynamics during the Sun acquisition phase, cumulating 9000 simulations and the related data. In order to assess the sensitivity of the algorithms to other failure modes and therefore their ability to correctly isolate failures, an additional set of data simulating thruster failures was generated (3000 additional simulations). All simulation data were then processed in order to be fed to the machine learning algorithms.

The neural networks performances in terms of detection time, rate of false detection, rate of no detection, were then assessed and compared with classical FDIR techniques performances. A synthesis of the results is presented in Figure 3, showing that AI-based FDIR method exhibits better performances than traditional FDIR for this use case. Although initially appealing, the semi-supervised ADDICT algorithm showed lower failure detection performances and longer execution time.

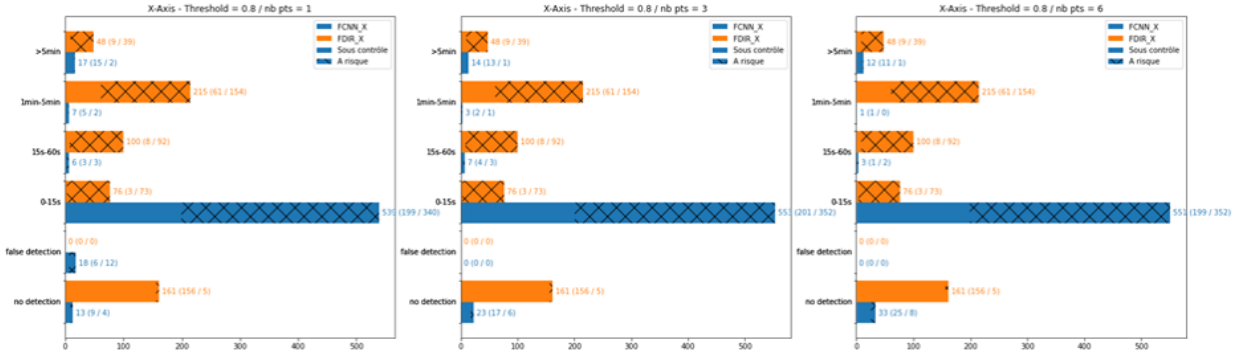


Fig. 3. Neural networks performances (detection time, false detections, no detections) for different tuning values

3.2 Star tracker blinding

The second use case considered data generated from hybrid hardware-software test benches, or "flatsat". This data corresponds to 5 hours of closed-loop tests with actual HW units. During the test, one out of the three onboard star trackers was blinded, leading to the use of an erroneous quaternion within the onboard attitude estimation process, eventually generating a pointing error (shown in Figure 4). This type of anomalous behaviour is typically detected through a higher-level monitoring that triggers a transition to safe mode and mission interruption. A machine learning approach could potentially detect and isolate the star tracker anomaly before triggering the higher-level monitoring, therefore improving the mission availability. The considered method in this use case is the Local Outlier Factor (LOF), which is an unsupervised outlier detection method.

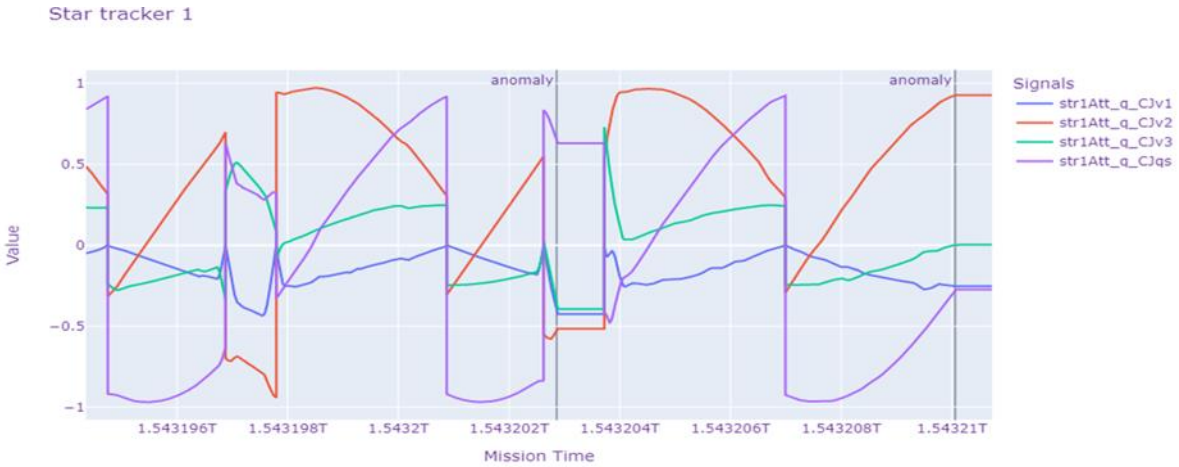


Fig. 4. Example of star tracker blinding

Although the LOF algorithm showed adequate failure detection results in the considered scenario, it should be noted that this algorithm is only able to detect isolated and sudden phenomena. In case of slow drift failure mode, the failure may go undetected.

3.3 Results analysis

The two use cases under consideration and the assessment of the machine learning algorithms allowed demonstrating the feasibility of using such techniques for onboard failure detection. Furthermore, it was demonstrated that neural networks techniques could actually improve failure detection performances with respect to classical FDIR methods. These findings were a justification for further exploration in this direction and expanding the development to cover the implementation on an edge device, i.e. the Xilinx Zynq UltraScale+ MPSoC.

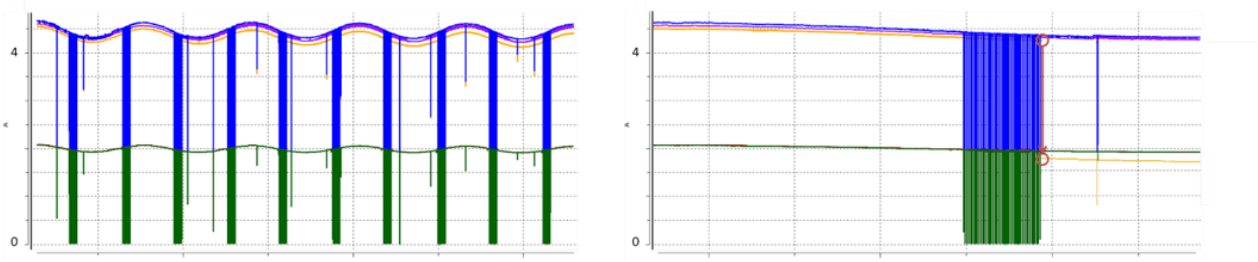
4. Deploying and Testing of an AI-based FDIR Algorithm on the Edge

The use case selected for the implementation on the edge differs from the ones described in the previous paragraph. For the experiment described hereafter, we tested the detection of a current loss due to the impact of a micrometeoroid on a solar array. Such a failure can be experienced in any mission independently of the platform hardware architecture. For training and testing the machine learning models described hereafter, real satellite telemetry has been used. For this use case, we turn to the unsupervised learning paradigm, where the chosen models are trained on data representing nominal spacecraft behaviour and validated on data containing the aforementioned anomaly.

4.1 Use Case Description

During the satellite lifetime, the solar array currents show large variations due to many factors, such as the mission phase (e.g. the satellite orientation to the sun), the presence of earth or moon eclipses, the varying solar activities over the year and last but not least, the degradation induced by the radiation. All these factors complicate the handling of a monitoring with the classical FDIR, especially as a power drop is just another variation which under some circumstances is actually expected.

To this end, the aim is to explore the performance of ML-based techniques in detecting drops in voltage for a MEO mission mounting 2 Solar Array wings composed of 2 separate sections. Figure 5 a) gives an overview on the mid-term trend of solar array currents over several years for the delivered currents of the single sections. It shows all types of variations which are experienced, especially eclipse seasons are very concise events, with power drops to zero. Smaller moon eclipses, orbital maneuvers, the degradational trend and the solar activity variations throughout the year are visible. 5 b) shows the effect of an anomaly where one of the sections exhibits an output reduced by more than 50%. In this scenario, the effect is caused by a micrometeoroid impact which occurs during eclipse season (marked in red), leading to a completely different trend after the eclipse season is over (yellow line). Such a behavior went initially unnoticed by the on-board FDIR and by the ground operators as the system was still operating within its FDIR design limits.



a) b)

Fig. 5. Nominal vs anomalous behaviour of solar array currents

Due to the criticality of the on-board power system, monitoring for such an anomaly on ground would not be beneficial due to the limited amount of the data downlink windows. Hence, depending on the nature of the failure, on the extent of the power drop, and the mission phase, there might be a risk of jeopardizing the mission. However, with an AI-based FDIR, it is possible to early detect such an event and, depending on the severity of the resulting reduction in delivered currents, tailor a specific recovery action which adapts the satellite's electric power consumption (e.g. payload switch-off, modification of battery charging levels etc..) in order to minimize the propagation effect of the failure and potentially avoiding the triggering of a Safe Mode.

4.1 Experimental setup

To detect the impact of the micrometeoroid leading to the anomaly, we aim to construct a model of spacecraft telemetry representing nominal operations. For this, we train deep neural networks on nominal telemetry data in an unsupervised way and then use reconstruction or forecast to evaluate the input telemetry during inference. The deviation between the real and predicted telemetry is calculated and if it lies above a specified threshold, the telemetry is marked as anomalous.

We experiment with several model architectures based on autoencoders and feed-forward models. The autoencoder-based models reconstruct the input telemetry (or parts of it, depending on the number of telemetry channels configured), while the feed-forward models work by forecasting one or more time steps into the future. For both types of models, we explore the effect of using different combinations of layers, such as 1-dimensional convolutional layers and recurrent layers like Long Short-Term Memory (LSTM), Bidirectional LSTM and Gated Recurrent Unit (GRU).

For model training and evaluation, we create three datasets (train, validation, test) combining real telemetry and artificially injected anomalies. All three datasets contain seven features (TM channels) representing the currents of the different solar array sections to be monitored by the FDIR and the eclipse flag indicating whether the satellite is currently in a sun or Earth eclipse. The training and validation datasets are composed of real in-flight telemetry of an example mission. The datasets represent nominal mission operations and are used to train the neural networks and tune their hyperparameters, respectively.

The test dataset is obtained by artificially injecting drops of magnitude ranging between 1% and 10% at random timesteps into one of the six currents.

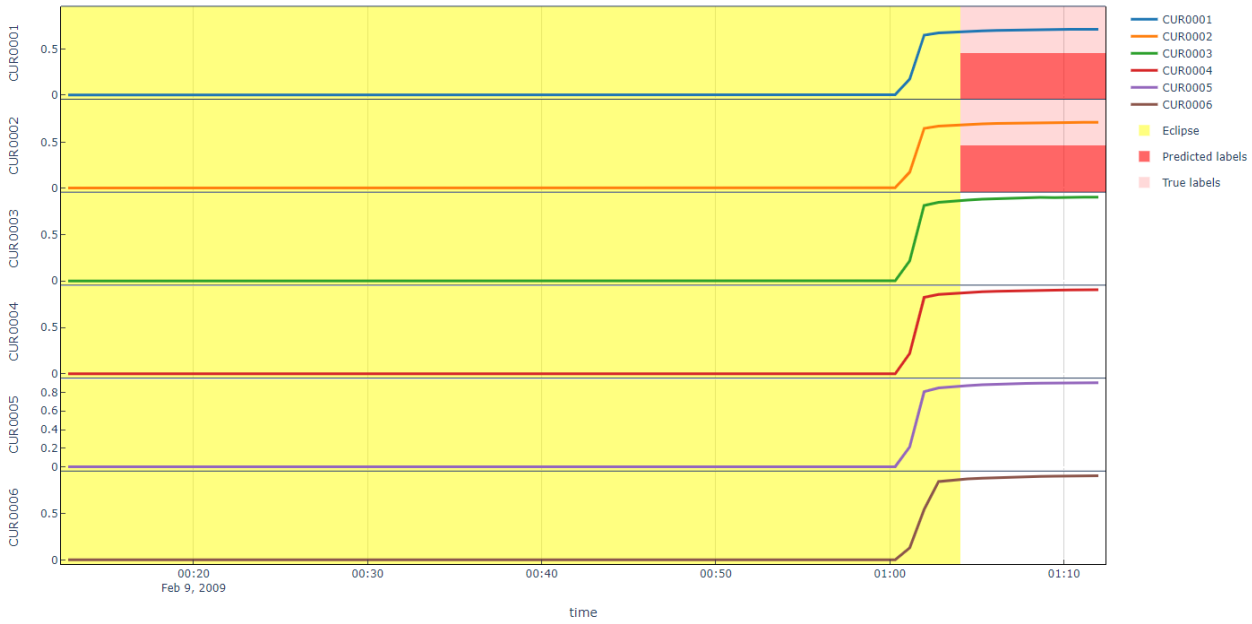


Fig. 6. Results of anomaly detection in six telemetry channels using a neural network model

The best-performing model is an autoencoder network which takes a window of historical telemetry values to reconstruct the current timestep and is based on 1-dimensional convolution layers (Conv1D). It can be seen from Figure 6 that the model achieves very good performance in detecting the onset of the anomaly contained in two out of the six telemetry channels (model features). The drop in current comes right after the eclipse and the model is able to flag this drop as anomalous behaviour. Furthermore, Table 1 below (Original Model Results) shows the values for the following performance metrics averaged across the six features: 1) accuracy (the ratio between the correct detections and the overall number of data points), 2) precision (how many of the anomalies detected by the algorithm were actual anomalies), 3) sensitivity (how many of the anomalies in the data were detected) and 4) specificity (how many of the nominal points were marked as nominal).

The model maintains a good trade-off between false positives and false negative detections and is also able to cope with the data gaps resulting from the asynchronous downlinking of the TM packets (the problem of asynchronous telemetry is not expected to happen on board). It is important to consider that the threshold used for flagging anomalies in the data plays an integral part in the overall performance of the model - choosing a higher threshold will result in fewer false positives but may make the model miss subtle anomalies in the data and vice

versa. We choose the threshold that provides the best trade-off between false positive (FP) and false negative (FN) detections - for the best-performing model the threshold value is 0.090.

4.1 Hardware implementation

Regarding the hardware implementation of the model, we use the Xilinx DPU processor which is available for Xilinx FPGAs and allows acceleration of neural networks using the VitisAI framework. As VitisAI does not currently support 1-dimensional convolution layers, we re-implement and re-train the network using 2-dimensional convolutions (Conv2D), which are supported by the framework. By making simple transformations in the data (adding an extra dimension and using padding), we obtain a network which is compatible with the hardware development framework and has an equivalent architecture and performance. The VitisAI compiler compiles and quantizes the model (also with the option to perform quantize-aware re-training of the network), yielding a new model quantized for 8 bits which is computed on DPU.

For initial experiments, we use the SD card image provided by Xilinx. The SD card image consists of a Linux image, drivers and bitstream for the DPU. VitisAI Run Time (VART) provides Python and C++ APIs to send and receive data to and from the DPU. The FPGA inference is done with 18974 frames and results with a total latency of 7.95 seconds with 3188 frames per second.

Table 1. Performance evaluation of the model with the best-performing architecture

	Accuracy	Precision	Sensitivity	Specificity
Original Model Results	0.999840	0.999059	0.999826	0.898581
Quantized Model Results	0.999881	0.999292	0.999944	0.999869

Table 1 presents a comparison of the performance metrics between the inference of the original model performed in Tensorflow and the inference as done on the target hardware platform using the quantized version of the model. It can be seen that the quantized model results with a performance comparable to the original model - the precision and specificity of the quantized model is even higher, possibly due to the fact that the quantized network has a smoothened loss landscape and therefore improved generalization power. The demonstrated performance is considered sufficient with margin and would therefore allow the integration of further telemetry channels.

5. Conclusions

The results of the experiments performed in our research demonstrate the overall feasibility of the enhanced on-board FDIR concept based on machine learning. Considering the current state-of-the-art in the development of machine learning models and their implementation on the edge, the intelligent FDIR solutions should primarily be focused on functional failures that are difficult to detect locally through classical FDIR techniques, e.g. slow drift, erroneous measurement albeit deemed valid by unit internal or local FDIR, thruster failure. The ML-based FDIR could then be used either to replace the existing, classical FDIR on a specific set of units and failures for better performances and faster development, or to complete the existing FDIR for better failure isolation.

Several axes have been identified for future research. The first axis would investigate further machine/deep learning techniques such as generative adversarial networks or networks with attention mechanisms. Further improvement of processing time and reduction in the memory footprint with the respect to the on-board hardware implementation is another topic of high interest. Finally, a third research direction would consist in assessing other use cases and subsystems besides AOCS and power, for instance thermal or propulsion failures.

References

[1] https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (accessed 15.01.2023)
[2] https://finn.readthedocs.io/en/latest/ (accessed 15.01.2023)