New Concepts Of Automated Anomaly Detection In Space Operations Through ML-Based Techniques

IAC-22-B6.2.6

New Concepts Of Automated Anomaly Detection In Space Operations Through ML-Based Techniques

Carlo Ciancarelli[a*], Francesco Corallo[a], Salvatore Cognetta[a], Eleonora Mariotti[a], Livia Manovi [b], Mauro Mangia [b], Alex Marchioni [b], Riccardo Rovatti [b] , Fabio Pareschi [c], Gianluca Setti [c]

[a] *Thales Alenia Space Italia, Italy, carlo.ciancarelli@thalesaleniaspace.com*
[b] *Alma Mater Studiorum - University of Bologna, Italy*
[c] *Politecnico di Torino, Italy*
* Corresponding author

Abstract

Spacecraft health management is a key component to ensure the safety and mission operation life of a satellite complex system. The health monitoring task is pursued exploiting telemetry data, collected using various sensor reading from on-board devices, that can be analyzed to retrieve and early detect anomalies which can lead to critical failures. The traditional monitoring methods, based on simple threshold checks, are now facing with lots of difficulties the increased complexity of the spacecraft, requiring updated and intelligent systems based on data-driven approaches. In this paper we propose different ML-based methods that contribute to the generation of an intelligent anomaly detector, that can face up the numerous telemetry data. Finally we focus on how to optimize and implement the developed models on constrained hardware, representative of spacecraft processors.

Keywords: Machine Learning, Spacecraft, Health Monitoring

1. Introduction

Health Monitoring (HM) and prognostic are key modules for complex systems like Spacecrafts, and play a relevant role to guarantee its reliability, availability and safety objectives.

The complexity of modern space systems, related to their size, autonomy requirements, harsh extraterrestrial environment and the impossibility to perform repairs after the launch, makes the telemetry analysis by ground operators an essential and time-consuming activity. The on-board generated telemetries describe the overall system health status, including measures from sensors like temperature, voltage, absorbed current, etc., which are also constantly monitored by the on-board FDIR (Fault Detection, Isolation and Recovery) function. Moreover fault detection systems like these are required are needed to alert space operations engineers of anomalous behavior and prevent significant failures, which could result in a potential loss of the spacecraft itself.

Expert knowledge is currently required in order to correctly understand and analyze the data coming from the spacecraft, making difficult to provide cost-effective ground operations, especially given the growing number of constellations planned in the near future. The on-board FDIR function classically employs a threshold-based logic, consisting in verifying whether telemetry values are within pre-defined limits or fall outside of them, also referred to as "Out-Of-Limits" (OOL) technique. The definition of the nominal ranges/threshold values and tables for each telemetry sensor represents an expensive and time-consuming task, requiring advanced domain-specific knowledge. Moreover the telemetry monitoring has to be performed for the whole duration of the mission, which can be translated in workload for space operations engineers.

The experience acquired by Thales Alenia Space Italia (TAS-I) in the frame of Space Segment Operations engineering, shows that the increasing operational workload required to analyze spacecrafts telemetries could no longer be met with classical approaches. Innovative and more efficient approaches are required to automatically detect anomalous patterns in the data, reducing the ground operations workload while increasing the mission reliability and availability. The generated telemetry data streams can be analyzed and correlated with a data-driven approach, in order to automate the complex task of failure detection and prognostic. Machine Learning techniques can be exploited to model and predict the system behaviour, allowing to early detect potentially critical anomalies.

2. Related works

Today's onboard FDIR solutions are based on simple Out-Of-Limit checks (OOL) for the identification task of anomalies in telemetry data. In the last decade the increasing maturity of Machine Learning techniques and more specifically for anomaly detection techniques lead to the consideration, through feasibility study, and implementation also in space use cases, constrained by limited Hardware capabilities.

These maturity and reliability was first envisaged by space agencies like NASA and ESA for on ground applications, where ML was applied and deployed to assist agencies operators. [1] In this study we aim to apply and coherently modify these data driven approaches to on-board application, facing up to space constraints.

From a data point of view, this kind of task has to face another difficulty, identified in the composition of the dataset at disposal of Data Engineer and Scientist: the anomalies are (fortunately) rare and spurious events, therefore the available dataset contains high volumes of nominal data, generating an unbalanced set in favor of normal samples. Even if synthetic anomalies can be injected in telemetry to simulate faulty sub-modules, they will not be fully representative of spurious and never before seen anomalous events that can occur in a monitored system.

Therefore, in order to face up this peculiar dataset composition, Machine Learning provides a set of approaches based mainly on unsupervised or semi-supervised learning techniques. The first try to train a model without a knowledge of the data itself; while the second one leverages on the fact that the majority of samples are nominal and the model are trained to learn only the manifold of these samples, if a samples belongs outside this space it is considered as anomalous.

Hereafter a literary review of automated anomaly detection approaches is presented, focusing on unsupervised and semi-supervised models.

Starting from clustering techniques we consider [2], K-means [3] which given an initial but not optimal clustering, relocate each point to its new nearest center, update the clustering centers by calculating the mean of the member points and repeat the relocating-and-updating process until some predefined convergence criteria are satisfied. This method was exploited for health monitoring and anomaly detection applied for industrial robot in [4]. The detection of anomalous events is performed during a post-processing phase, where the test data and the cluster centers, computed by the K-means algorithm after the training procedure, are compared: if the distance from the new data points to the cluster centers exceeded a predefined threshold the data point was flagged as an anomaly.

Random Forest (RF), a Decision Tree approach, was exploited in [5] for condition monitoring and early detection of failures for industrial plant equipment. After the data pre-processing and feature extraction phases, which include the transformation in the frequency domain via the Fast Fourier Transform (FFT), the authors used the extracted features to train the RF model and construct N decision trees trained on N bootstrapped samples of the training data. The resulting model was able to predict the development of the industrial plant health index with better performance of a persistence technique used as benchmark.

Another approach for fault and anomaly detection was developed using Least Squares-SVM in [6], where the authors proposed an LS-SVM regression model for spacecraft telemetry to identify contextual and collective anomalies [7] of in-orbit satellite telemetries. The resulted algorithm consists in an improved version of the classical SVM method, obtaining very good classification and generalization ability for anomaly detection of spacecraft in-orbit telemetries.

The framework proposed in [8] exploited, distinctly, two different approaches VAE [9] and GANomaly [10] trained on nominal data generated by LUNASIM [11] (lunar orbiter AOCS simulator).

The first approach is a probabilistic model that tries to reconstruct data from the input data space via maximum likelihood estimation and variational inference, the second one is an implementation of the Biderectional GAN framework that borrows an AE-style generator and an encoder-style discriminator for parsing the representation of a given training-set.

Hence, anomalies are detected by measuring the reconstruction error for newly input data. It was shown that if nominal and abnormal samples are not clearly different, the VAE obtains noisy output conversely GANomaly could discriminate abnormal samples more effectively because it utilizes AE to reconstruct input data and GAN to match a distribution of generated samples to the target (nominal) distribution.

In addition, different methods can be used concurrently for the Anomaly Detection, using a combination of models that work on the same (or group of) signal or using parallel models that are trained on different TM signals/channels. The first approach is called in literature Ensemble Learning and allows to train multiple models on the same task together, which outputs can be combined using three different techniques.

Even if this approach can potentially have better performance, it has a major drawback given by the high

computational effort and demand that multiple models used concurrently can need. Therefore, it's important to take into account the Hardware constraints imposed by a limited and complex system like a spacecraft.

The paper is organized as follows. In Section 3 a background on the satellite FDIR mechanisms is provided, as an introduction to the proposed operational use case, described in Section 4. Section 5 is devoted to the mathematical modeling of the adopted anomaly detection ML techniques and the identification of the critical hyperparameters to be tuned, alongside the definition of the score generation mechanisms and the estimation of the memory footprint. Moreover, the experimental settings are described and detection performances in the reference scenario are discussed and compared. Section 6 points out to some hardware implementation strategies for low-power micro-controllers. Finally, some conclusions are drawn.

3. FDIR

The satellite FDIR logic has the aim of detecting, isolating and recovering faults at unit, subsystem or equipment level. It is actually based on a hierarchical architecture trying to confine failures at the lower FDIR levels to minimize outages and provide high system availability; for example, in terms of Avionic subsystem, this mechanism should be able to detect the failure at sensor or actuator level before having problems at attitude level. However, it is a deterministic approach based on several predefined tables containing selected monitoring items and relative recoveries. These tables are designed according to experience and then implemented in the Avionic SW (ASW), they come from different satellite subsystems and each of these stores a set of parameters, but they are fixed and they are not able to detect failures that are not foreseen by the designer. In fact, the goal of the approach is to verify that a proper set of parameters computed by the ASW does not exceed the predefined operational thresholds. After a confirmation time ("filter" configuration data), the detection of the violation of a monitoring criterion foresees a recovery action. This is done in order to eliminate spurious or transient events that don't affect the system. In general, the satellite FDIR is usually handled by software functionalities, but the highest level of the FDIR hierarchy is in charge to the Ground Control Centre (GCC). The GCC is able to send telecommands in order to enable/disable the on-board autonomous FDIR operations, or for setting the FDIR configuration parameters and logics.

As said previously, the FDIR performances could be potentially affected by this limited nature of a table-driven approach, since both operational limits and confirmation time are decided a priori by design. In fact, the approach is not flexible and it is not able to recognize any type of failure, but only those ones that are expected by design and for which the parameters have been selected to be monitored. Furthermore, it does not guarantee any preventive maintenance. When the satellite is affected by unexpected recoveries due to lack of design or prediction capabilities, GCC shall investigate about the anomaly causes and it could take long times to restore the satellite functionalities. For this reason, the introduction of on-board ML approaches for FDIR could overcome these problems, especially in identifying and isolating failures at the lowest level possible (equipment level) thus fostering equipment/software reuse, mission availability and autonomy. In fact, the ML algorithms could be able to analyze a big amount of on-board available data and recognize failures not foreseen by design, or could react faster than the fixed confirmation time.

The table-driven approach has been implemented until now due to space hardware limitations. Space computers are really much less powerful than the terrestrial ones. On-Board Computers are characterized by space qualified components, since hardware shall be reliable to space environment (for example, ionizing radiation). This means that space OBCs are limited by computational capabilities, maximum volatile and non-volatile memories constraints, maximum stack of any algorithm function etc., i.e. all characteristics relevant for ML algorithms usage. In order to use the ML algorithms on-board, it should be also necessary to investigate about the possibility to optimize software coding and reduce the computational needs, paying attention to the use of external AI and ML libraries that should be no hardware dependent or adaptive to space software compilers.

Thanks to the most recent improvements in space processor performance, the use of ML strategies on-board has been recently investigated. Several categories of space processors can be identified, of which different are already used in flight. The low performance and medium-low performance devices are already used for a variety of space missions. The first category consists of a single-core architecture, but AI inference algorithms generally need a very large number of MACs operations to be performed in parallel (especially NN and DNN algorithms) and single-core architectures are not particularly suited to support this kind of processes. The second one consists of multi-core architectures. Scaling the numbers of cores constitutes, with frequency scaling, the easiest way to improve the performances of single-core architectures, and this seems to be in line with the implementation of several ML algorithms with limited computational effort. Today, higher categories of processors (many-core and FPGA/programmable SoC FPGA) are not ready for in-

flight usage since their Technology Readiness Level is too low to be considered as possible choices, but some of these are considered possible candidates for future missions. This augment the possibility to introduce higher demanding ML algorithms on-board, and improve the performance of FDIR logic.

4. Space data

With the aim to verify and test the capabilities of the proposed algorithms in the task of identifying on-board anomalies through the analysis of the telemetry data, thus reducing the on-board reaction time of the FDIR function, a real-life operational use case is proposed hereafter. The data available in this work comes from different sensors on-board a Spacecraft in the LEO (Low Earth Orbit), in particular from the Reaction Wheel (RW) units, which are actuators consisting of a spinning wheel driven by an electric motor, used for three-axis attitude control and stability of a spacecraft: these are typically selected in configurations of three or more units for redundancy, and allow to apply a continuous fine attitude control over the orbit with a high pointing accuracy, which is crucial in the nominal space operations.

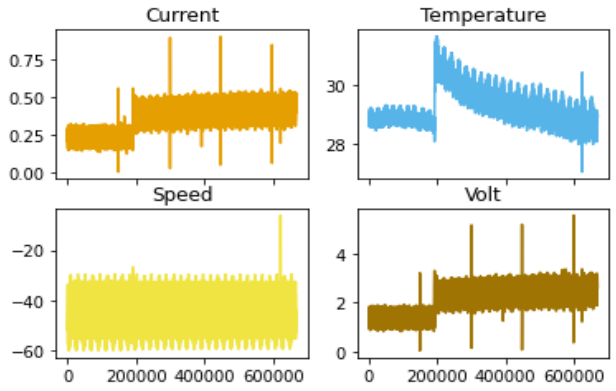

Figure 1. Telemetry data containing anomalous event occurred on a RW used a test case

The monitored sensors mounted on the RW that are exploited for the study are:

- motor current A, current absorbed by the RW;
- thermistor $^{\circ}C$, temperature of the RW;
- speed rad/s, velocity of the RW;
- torque V, torque commanded to the RW.

A sample of the telemetry in nominal behaviour can be seen in Fig. 1, where there are three different lines, each one corresponding to a different sensor: motor current (blue line), temperature (orange line) and angular speed (green line) telemetries of RW.

The satellite does not operate in the nominal operation mode all the time during the mission, and could enter non-nominal operation modes for a number of reasons including orbit maneuvers. These, in turn, create outlier values in the telemetry data that could alter the anomaly detection process. To distinguish the outlier data related to non-nominal operations from the actual anomalous one, each sample at time t_i is also tagged with a flag denoting whether the received values correspond to a nominal or non-nominal operation of the satellite.

An example of anomalous behaviour can be seen in Figure 1, which contains the plots of the telemetry data of the four sensors in anomalous behaviour: we can observe a step in the values of *current, temperature and voltage*, while the *speed* remains at the nominal operation value. This step, which does not happen in the nominal cases, is caused by the anomaly registered in the RW itself. It has to be noted that the values of the sensors are plotted using different y-dimensions.

5. ML algorithms for anomaly detection

In the attempt of making the detector effective, a classical signal processing chain is considered, as depicted in Fig. 2. On-board sensors readings, pre-processed in order to transform heterogeneous sources of information into homogeneous time-series, are prepared to be examined by employing a time window-based partition, so that it is more likely to accomplish the task of uncovering sub-sequences of unusual behavior.

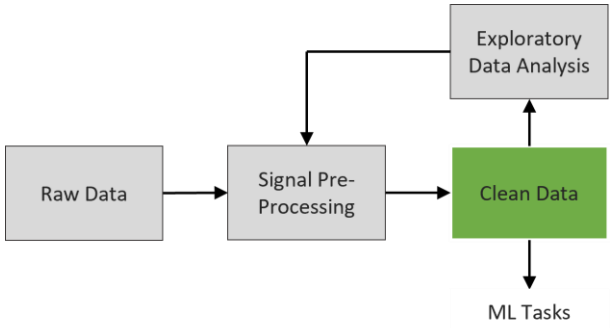

Figure 2. Signal acquisition and processing chain to prepare data in compliance with the ML tasks

Actions performed in the pre-processing stage could include: missing data imputation, filtering/cleaning, re-sampling, re-scaling and low-level feature extraction. Depending on the number of sensors included in the monitoring system, a score value is computed starting from more than one windows of samples. The detectors outputs, i.e., the scores, can then be transformed into final labels in comparison with predefined thresholds. Prepared data are

arranged in a tabular fashion with each row relating to a single timestamp $t_j, j = 0, \dots, N_T - 1$, and columns identifying a specific telemetry signal.

The segments composing the obtained time series y can be compared by transforming them into a multidimensional representation. One may be interested in observing either the correlation across time of the values assumed by a single quantity (upper part of Fig. 3) or across multiple series synchronized by their timestamps (lower part of Fig. 3). Clearly, a semantic understanding of the problem domain is fundamental in order to identify a proper transformation. The simplest transformation consists in building a high dimensional input matrix $X \in \mathbb{R}^{N_{T_i} \times n}$. Nevertheless, it is possible to derive other representation types, e.g., by adopting discretization methods. Other methods such as Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT) are available for compressing the series into numeric coefficients (multi-scale decomposition).

In the optic of not being able to simulate every faulty condition that might occur in a complex system, the monitoring issue, and so the definition of these kinds of detectors, can be addressed by training phases realizing on the recognition of those behaviors which don't exceed the boundary of a nominal one.

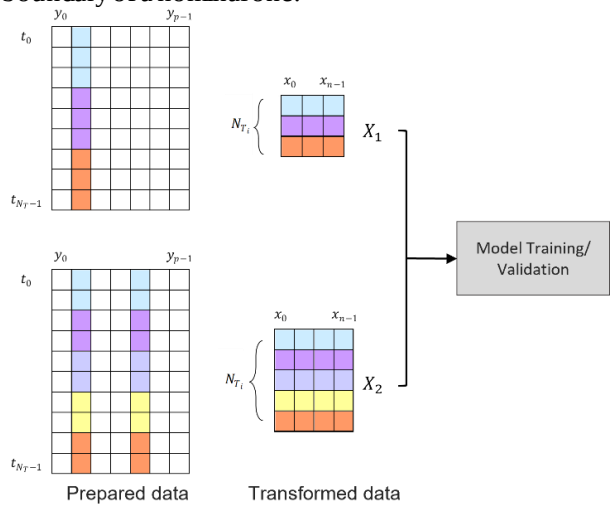

Figure 3. Transformation of prepared data into multidimensional vectors ready for the ML tasks, either considering correlation across time of a specific signal (X1) or across multiple series as well (X2)

Within this framework, let $\eta: \mathbb{R}^n \to \mathbb{R}$ be the function providing a score for input $x_i \in \mathbb{R}^n$. The training phase allows to derive the confidence intervals of the *inlier* scores values, assumed to have an upper bound l^+. Here, the test set is balanced in the amount N_S of good and anomalous examples and undergoes the same signal processing to which the training set is subject (Fig. 4).

A score $\eta(x_i) = l_i \gg l^+$ is symptomatic of a behavior significantly differing from the underlying data generation process. Here, the focus is on four possible detectors which differ in the concept of similarity they rely on, i.e., on how they define the boundary representing the nominal behavior. Moreover, each model has a distinct impact on the memory footprint.

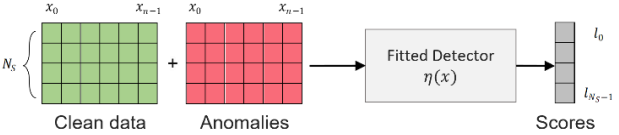

Figure 4. Inference stage

5.1. Similarity-based methods

This section is devoted to the introduction of two well-known ML methods:

- a proximity-based method, the approach called Local Outlier Factor, LOF, which combines the k-nearest neighbors learning method with some data-dependent similarity measure to address the issue of local density variation;
- a non-linear method, One-Class Support Vector Machines, OCSVM. It takes advantage of non-linear mappings to compute similarity measures in what is called a feature space.

In the presented setting, no specific assumption is made on the inputs $x \in \mathbb{R}^n$ other than they live in a metric space. One of the most flexible concepts of similarity measure is indeed found in the scalar or inner product, although, in general, nothings ensures that the input space $\mathcal{X}$ is provided with such an operation nor the linear separability of its objects. Proximity can thus be computed when the collections of sub-sequences of time series are transformed into matrix $X \in \mathbb{R}^{N_{T_i} \times n}$, e.g., by means of the Euclidean distance.

5.1.1. Local Outlier Factor

The model, originally proposed in [12], exploits the concept of k-distance of an observation $x_i \in \mathbb{R}^n$, denoted as $d_k(x_i)$, i.e., the distance between x_i and its nearest neighbor t_i in the training set $\mathcal{T} \subset \mathbb{R}^n$, which is in turn compared to distance $d_k(t_i)$ between t_i and its own neighbors in the training set. It is possible to assert whether a point lies within a k-neighborhood or not by deriving a measure called reachability distance and defined as

$$r_k(x_i, t_j) = \max\{d(x_i, t_j), d_k(x_i)\}, \qquad (1)$$

where $d(x_i, t_j)$ is the actual distance between two points. Then, by taking the inverse of the average r_k of object x_i *from* its neighborhood, one derives a *local reachability density*, namely $l_k(x_i)$, which is the distance at which x_i can be reached by its neighbors: the quantity is small when x_i is isolated from the surrounding neighbors. Finally, the local outlier factor score can be derived by comparing the local reachability densities of the points in neighborhood $N_k(x_i)$ and the object's own value as

$$\eta^{LOF}(x_i) = \frac{1}{|N_k(x_i)|} \sum_{y_i \in N_k(x_i)} \frac{l_k(y_i)}{l_k(x_i)}. \quad (2)$$

The local distance behavior is accounted for by using distance ratios in the definition of the score. In fact, a larger reachability distance indicates that the object in exam shows lower density than the surrounding ones, meaning it lives in a sparser region and can be considered an outlier.

5.1.2. *One-Class Support Vector Machines*

The approach finds its roots in the statistical theory of learning [13] [14] and it allows to find a hyperplane separating the *single-class* normal data examples from the origin with maximal margin ρ. The attempt of minimizing the structural error of such a hyperplane results in the definition of a quadratic programming problem [15]. In order to make the approach more effective, all SVM-based methods adopt the so called kernel trick such that the scalar product between two vectors is generalized as follows

$$\kappa(x, y) = \exp^{-\gamma|x-y|^2}, \quad (3)$$

where this represents the case of a Gaussian kernel function.

Within this framework, for a new observation x_i, the score is computed by

$$\eta^{OCSVM}(x_i) = \sum_{j=0}^{S-1} \alpha_j \kappa(s_j, x_i) - \sum_{i=0}^{S-1} \sum_{j=0, j\neq i}^{S-1} \alpha_j \kappa(s_j, s_i) \quad (4)$$

where $\rho = \sum_{i=0}^{S-1} \sum_{j=1, j\neq i}^{S} \alpha_j \kappa(s_j, s_i)$ is the maximal marginal and $s_0, \dots, s_{S-1}$ is a set of examples in the training set properly selected in the training phase, named support vectors, and for which the cardinality of the set is controlled by a tunable parameter $\nu \in [0,1]$. The basic idea behind a SVM-based detection mechanism is summarized with the schematic in Fig. 5. It resembles the architecture of a simple neural network, in which the weights α_j are derived from a subset of the training examples.

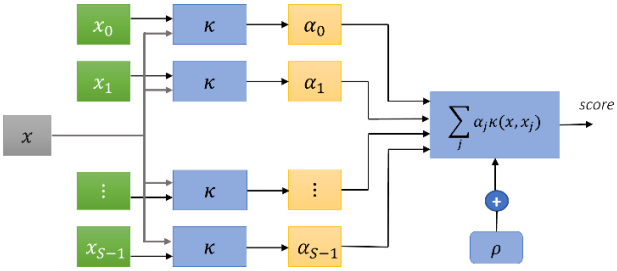

Figure 6. General architecture of a SVM

5.2. *Subspace-based methods*

The kind of techniques described here allows to map each incoming input signal $x_i \in \mathbb{R}^n$ into a lower dimensional subspace where the projection is $z_i \in \mathbb{R}^m$, with $m < n$. The input signal can then be recovered by means of a decoding stage as $\hat{x}_i \in \mathbb{R}^n$ from the low-dimensional representation (see Fig. 6).

- a linear method as Principal Component Analysis, PCA, which allows to extract a low dimensional representation of the data by projecting the windows of samples onto the subspace in which the information content of the signal concentrates;
- a non-linear method representing the generalization of PCA, a deep Neural Network architecture named Autoencoder, AE. Here the goal is to find the manifold which better represents the input signals in a low-dimensional space .

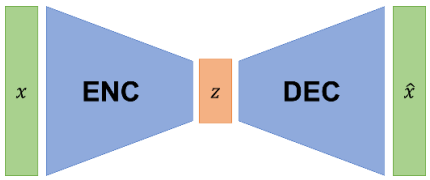

Figure 5. Working principle of PCA and AE

5.2.1. *Principal Component Analysis*

The main goal of this method is to find the so-called *principal components*, i.e, a set of $m < n$ vectors along which the energy of the input signals concentrates in average.

Linear projection is performed by means of a multiplication through a matrix $U \in \mathbb{R}^{n \times m}$ in a way that

$$z_i = U^\top x, \quad \hat{x}_i = U z_i. \quad (5)$$

where U is the matrix minimizing the expectation of the error $\| x_i - \hat{x}_i \|_2$ and it can be computed by arranging as columns of U the eigenvectors of the input signal correlation matrix associated to la largest eigenvalues. Here $\|\cdot\|$ indicates the l_2-norm of a vector.

Starting from the knowledge of U it is possible to define two distinct outlier scores to catch unusual variability both within and outside the learned principal subspace.

1. *Squared Prediction Error*, SPE: it represents both the reconstruction error for x_i and the level of inconsistency with respect to the identified best linear fit, i.e., the amount of energy in the minor subspace.

$$\eta^{SPE}(x_i) = \|x_i - \hat{x}_i\|_2^2 = \| x_i - UU^\top x_i \|_2^2 \quad (6)$$

2. *Hotelling's* T^2: it consists in a statistical test which provides the level of similarity between the energy content of the principal subspace and that of a specific input sample x_i and is defined as:

$$\eta^{T^2}(x_i) = \sum_{l=1}^{m} \frac{(U^\top x_i)_l^2}{\lambda_l}, \quad (7)$$

where $(\cdot)_l$ represents the l-th element of a vector and λ_l is the l-th highest eigenvalue of the input signal correlation matrix.

5.3. Autoencoder

As for PCA, an Autoencoder-based detector encodes some meaningful information extracted, by non-linear projection of the input observation x_i, into a manifold living in a lower dimensional space $z_i \in \mathbb{R}^m$, with $m < n$. This is also the input of a decoder stage which produces as output a vector $\hat{x}_i \in \mathbb{R}^n$ designed to be a replication of x_i (see Fig. 6).
Both encoding and decoding stages involve a neural network represented by the two non-linear functions $f(x_i)$ and $g(z_i)$ such that $\hat{x}_i = g(f(x_i))$. More in detail, in the encoder stage is composed by a convolutional layer (2 filters, strides equal to 4 and a kernel size equal to 40) and a fully connected layer producing z_i and preceded by a layer performing a flattening operation. The decoder is done by transposing the encoder architecture. The network parameters are trained by minimizing a loss function given by the *Mean Squared Error, MSE* computed as the expectation of

$$\eta^{AE}(x) = \| x - \hat{x} \|_2^2. \quad (8)$$

MSE is also used as anomaly score.

5.4. Model tuning and memory requirements

Based on the previous discussion, an estimation of the memory footprint of each method is given in terms of how many digital quantities must be stored. This is a quantity depending on n (the number of samples in each observation), the parameters characterizing each method and also, in some cases, on N_T, i.e., the amount of training examples. The critical parameter which has to be tuned for LOF is k, the cardinality of the neighborhoods along with the number of examples in the training set $\mathcal{T}$. For each of the N_T observations, the values of the k distances of the nearest neighbors need to be stored. Therefore, given the same prediction capabilities, the smallest effective k would be the preferable choice. The number of digital quantities to be stored is then computed as

$$\mathrm{M}_{LOF} = k \times N_T, \quad \text{with} \quad N_T \gg n \quad (9)$$

In the case of OCSVM, the critical parameters are γ, which modulates the kernel width (e.g., the Gaussian kernel in (3)), and ν, which sets a minimum on the number of support vectors and a maximum on the tolerated amount of outliers. Nevertheless, only the former impacts on the required amount of digital words to be locally stored

$$\mathrm{M}_{OCSVM} = n \times \nu N_T + \nu N_T, \quad \text{with} \quad N_T \gg n \quad \text{and} \quad \nu \in (0,1] \quad (10)$$

Regarding the detector based on PCA, the matrix U is the only entity to be stored on board such that

$$\mathrm{M}_{PCA} = n \times m, \quad \text{with} \quad m < n \quad (11)$$

Regarding the number of parameters characterizing the AE detector, this is a value that strongly depends on the adopted architecture.

$$\mathrm{M}_{AE} = n \times m + m + n/2 + 163, \quad \text{with} \quad m < n \quad (12)$$

As an example, with $n = 1000$ and $m = 250$ we have $M_{AE} = 250913$ parameters.
Since outlier scores are computed based on the comparison between groups of samples located in subsequent intervals of time, the size n of the adopted windows affects the achievable performance in terms of abnormal behaviour detectability.

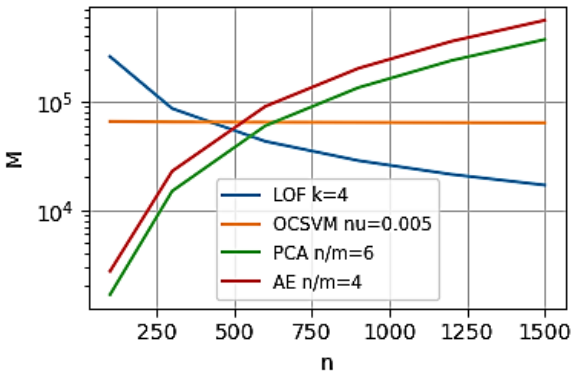

Figure 7 Amount of digital words required by LOF, OCSVM PCA and AE where LOF and OCSVM adopt the N_{T_3} configuration.

5.5. Numerical Evidences

In the training phase, for LOF and OCSVM, the model is fitted in three different configurations based on the size N_T of the training set, using references containing approximately $N_{T_1} n \approx 300K$ samples, $N_{T_2} n \approx 1.3M$ samples, $N_{T_3} n \approx 6.5M$ samples, respectively. Whereas for PCA and AE, since their memory footprint is not directly affected by the amount of training examples, only the third reference set and a set containing $N_{T_4}/n \approx 15.5M$ samples have been considered. The reference test set includes $N_S n$ samples, that is in the order of $\approx 300K$ sensor readings. Both training and test set are passed to a pre-processing stage where filtering and data standardization are performed. Both the processing and the investigation of the ML methods have been carried out in a Python-based environment. Pandas [16] was mainly used as tool for data manipulation, together with libraries such as Numpy [17]. Tensorflow [18] has been used as ML platform for the AE model and Pyod [19], a wrapper of the library Scikit-learn [20], for LOF and OCSVM.

For what concerns the model adopted to represent anomalous events, the presented analysis is specialized to the case of anomalies originated by Gaussian sources. Such kinds of anomalies could be representative of the effect of a large variety of possible system degradation conditions as well as it has been proved that white noise acts as a reference case in modeling the statistics of any possible class of anomalies [21]. Moreover, the clean signal is compared with portions of a time series associated with an actual anomaly recorded on the same channel, i.e., the step anomaly mentioned in Section 4. In the identified reference scenario, the four different cases in Table 1 are under investigation.

Being the test set balanced, one can adopt the area under the Receiver Operating Characteristic curve, ROC (briefly referred to as AUC), as metric to assess detection accuracy [21]. The AUC takes the scores as inputs and provides an indication on the detection capabilities of a model, independently of a possible threshold value.

Assuming value l_i^- is the minimum score assigned by $\eta(x_i)$ to the irregular instances in the test set, the detector is considered able to discriminate between clean signal and outliers if $l_i^- > l^+$, where l^+ is the upper bound of the confidence interval derived after the training phase for the scores associated to *inliers*, as anticipated in the initial part of Section 5.. The total absence of superposition of the scores distributions is assumed equivalent to AUC=1. In the following, histogram representations are used as a visual tool in order to distinguish between the distributions of the scores assigned to the clean signal and to the anomalous ones. The scores have been normalized by removing, at instance level, the mean μ_x and dividing by the standard deviation σ_x of the "good" ones, therefore, the values indicating normality concentrate around zero. A detector shows a good performance when a spread between the 90-th percentile, corresponding to the darker dashed line labeled as $p_x = 90$ in Fig. 8 and 9, of the scores assigned to the nominal behavior and the 10-th percentile, labeled as $p_c = 10$, of the anomaly scores is observed.

The settings adopted here were suggested by recent research. In order to include the required memory footprint in the analysis, Fig. 7 provides an estimation of the amount of digital words to be stored with the produced time-window division.

Table 1. Reference cases for the test phase.

Case	Description
C1	The clean signal is compared to pure white Gaussian noise sharing the signal energy (WGN)
C2	Additive white Gaussian noise is considered, (AWGN), where noise and clean data have the same energy
C3	AWGN as for C2 but with a lower noise intensity modeled by an energy ratio between the signal and noise equal to 5
C4	The signal representing the nominal mode is compared to a profile resembling an actually recorded anomaly

We limit the analysis to the cases $n \geq 600$, value around which an inversion in memory trend is recorded, since the detectors reach AUC=1 for $n \geq 600$ with only few exceptions. Fig.8 shows a comparison between all detectors except AE in their respective least expensive setting (LOF is set up with $k = 4$ as number of nearest neighbors, OCSVM with $\gamma = 1/n, \nu = 0.01$ and $n/m = 6$ for the PCA) when scores of clean signal are compared against the ones of anomalies in Table 1 when $n = 600$. Moreover, the first two methods are trained considering the less memory hungry configuration N_{T_1}. PCA, since its trained offline, is still working with N_{T_3} or even larger sets. In few cases only the AUC measure does not reach the full unitary value: an overlap of the score distributions can be observed for LOF in case C2 and for both LOF and OCSVM dealing with case C3 as well as for the T^2 scoring of PCA with respect to pure WGN (case C1).

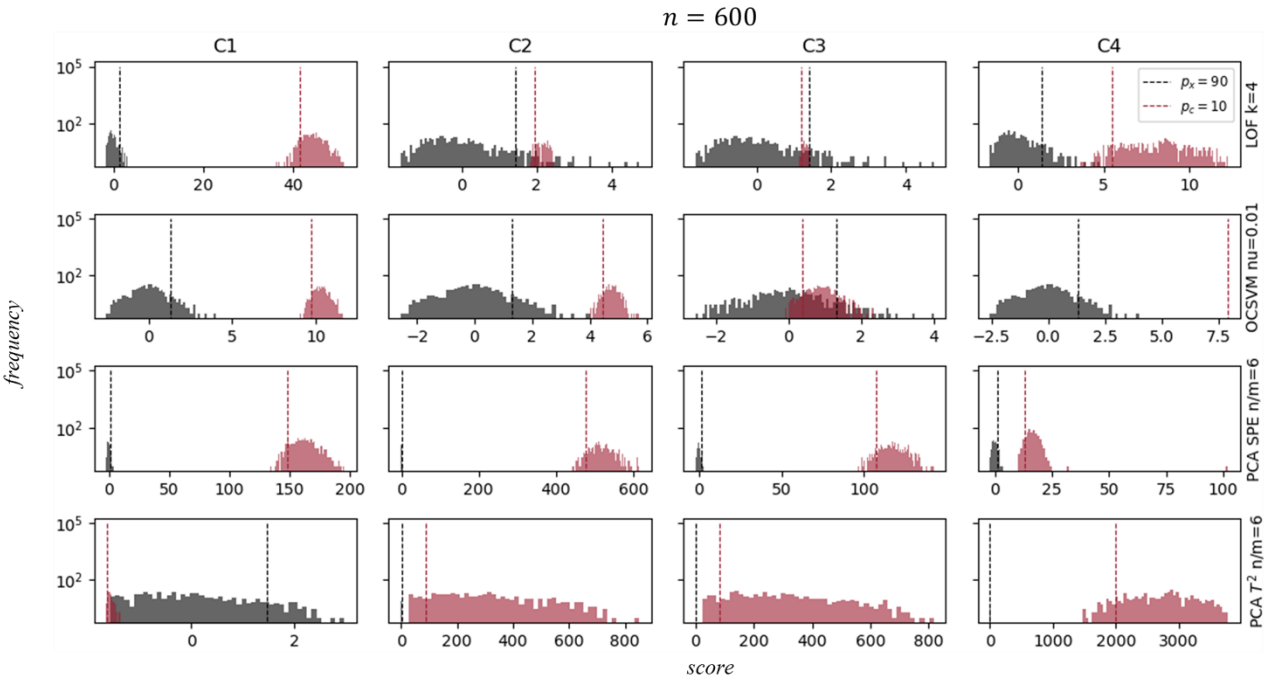

Figure 8. Distribution of the scores obtained with LOF, OCSVM and PCA (both SPE and T^2 scores) in their least expensive setting for windows containing $n = 600$ samples in the cases described in Table 1).

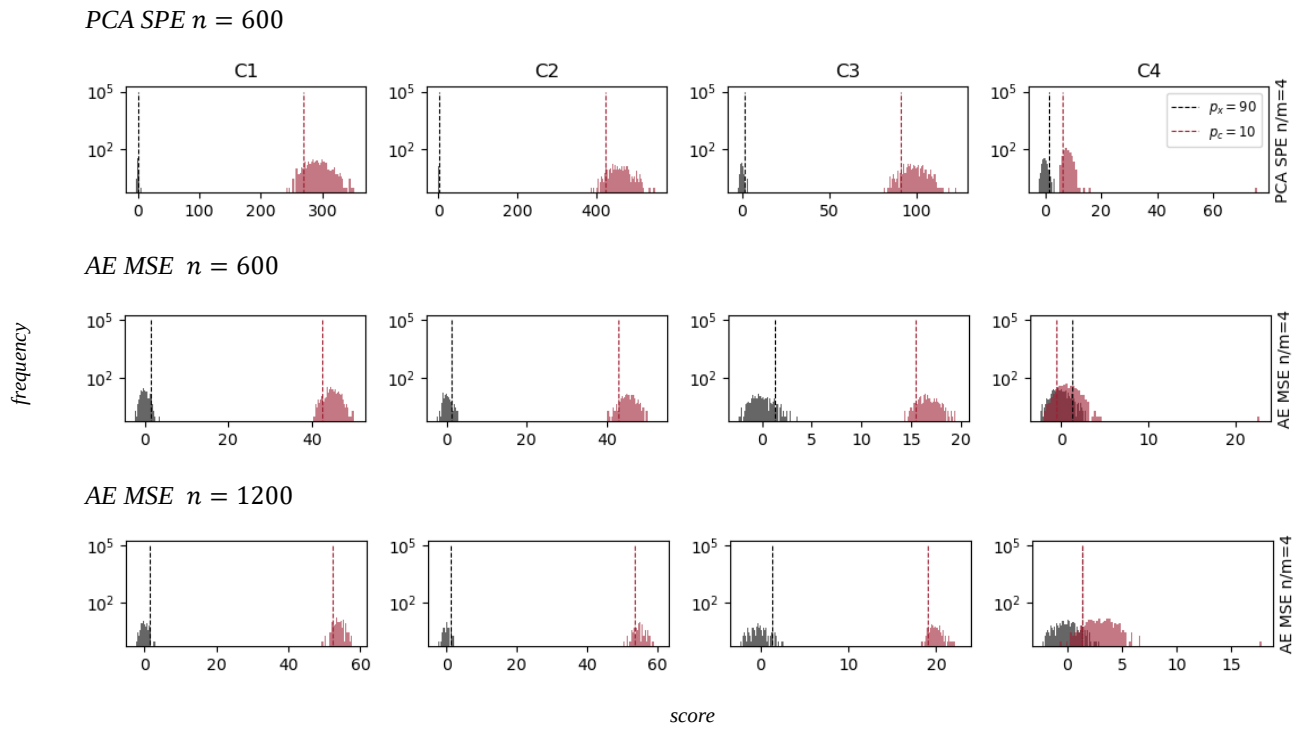

Figure 9. Distribution of the scores obtained with PCA (SPE) and AE for window length $n = 600$ and generated subspace dimension $m = 150$, in the cases described in Table 1).

Both the metrics adopted for PCA succeed in detection when the clean signal is compared to the noisy signals of cases C2 and C3, however, the T^2 scores show more variance. Moreover, all of the detectors appear to correctly flag the step anomaly C4. Fig. 9 depicts the comparison between the performances of PCA and AE when dealing with the aforementioned irregularity cases, proving the models act as proper detectors for cases C1, C2, C3 for $n = 600$ and $n = 1200$ (however, this is true for every n under test) and thus robustness against noise, even if additive and of medium-low intensity with respect to a regular signal. It has been already shown how the PCA scores are able to detect anomaly C4, The most critical case for AE is C4: for $n = 600$, there is a significant overlap between the score distributions, while it performs slightly better when the window length is increased (e.g., $n = 1200$). The AE model with convolutional layers, however, appears to be more versatile as a compression method when window length increases much, therefore is a candidate for solutions in which multiple time series are monitored. Nevertheless, PCA, confirms its already known robustness with a required memory footprint in the same range as the AE model, and perhaps* it will be long before the model goes actually out of fashion.

For the sake of completeness, Fig. 10 demonstrates how the scores deviate in time when encountering the step anomaly shown in the first subplot. All the detectors provide the ability to identify a peak of unusual magnitude, arising suspicions about the behaviour of the monitored system. However, among them, the SPE score for PCA and the MSE for the AE model do not keep track of the deviation of the signal above its usual mean value. In turn, the T^2 score is able to identify such change in behavior: as a matter of fact, the metric is more sensitive to the discrepancies in terms of signal energy. An usual trend in the evolution of the scores can be observed for LOF as well.

5.6. Results discussion

Given the aforementioned performances, the models, properly *tuned*, can be included in a list of candidates for an anomaly detection building block at the foundations of an FDIR mechanism. It appears that the described techniques allow to properly raise a flag in correspondence of corrupted sequences in the time evolution of a telemetry signal. In some cases, the desired performance is achieved even with the configurations of minimum cost, depending on the model, especially in terms of memory footprint. Future developments are expected with pruning approaches (note that it can even be done for LOF, e.g., by ranking the top r k-nearest neighbors, and OCSVM, for which is suffices to imagine the SVM as a non-linear network whose input weights correspond to significant training examples, although domain knowledge is fundamental for the task) and while exploiting data representation techniques such as wavelet transforms, especially in view of an automated system monitoring multiple quantities simultaneously. In the following section, the possible enhancements related to pruning techniques are discussed from a hardware perspective.

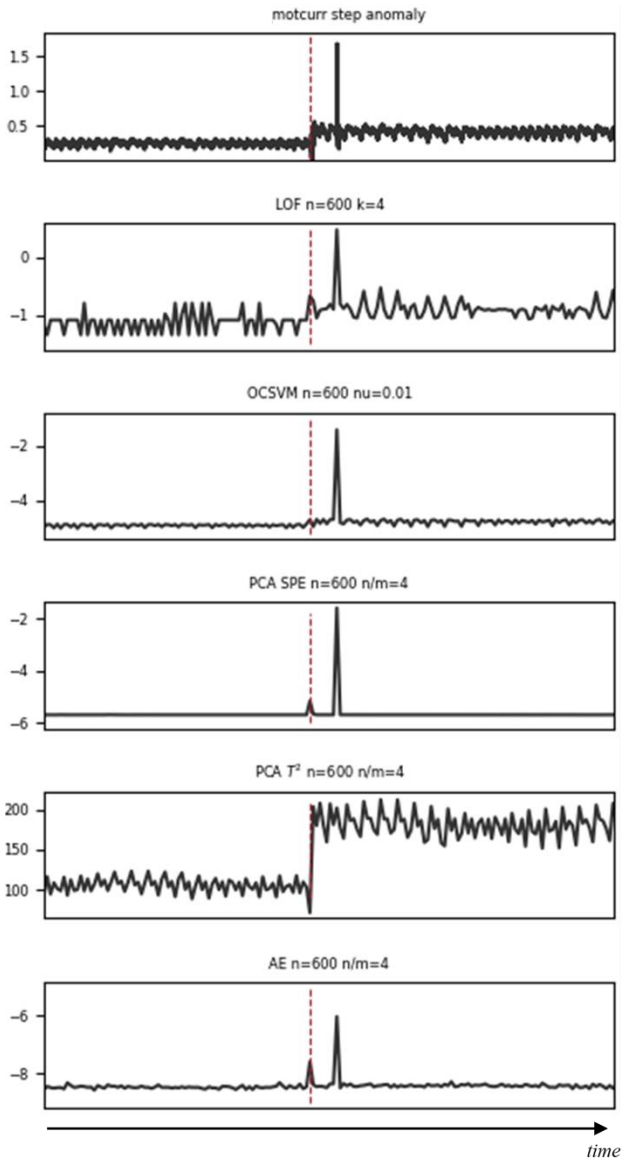

Figure 10. Evolution in time of the scores when the step anomaly of case C4 is encountered. The dashed line marks the beginning of the irregular behavior.

6. Hardware implementation strategies for low-power micro-controllers

When implementing ML based algorithms on board, one of the main issues is the possible lack in resources, as many ML algorithms are, in the general case, quite expensive in terms of resources requirements. In particular, deep neural networks often suffer from over-parametrization and large amounts of redundancy in their models, resulting typically in inefficient computation and memory usage [22]
Mainly, four aspects are to be considered.

- **Energy** - This is indeed an issue affecting any hardware block, due either to a limited energy available, or a limited possibility of dissipating energy. Mitigation of this issue happens both at hardware level (low-power implementation) or at a software level (lower complexity algorithms).
- **Speed** - Some types of data require real-time processing. Indeed, almost any types of data has a validity time range, after which they expire. A natural constraint for any data processing system is that data has to be elaborated before expiration, so elaboration speed has always to be taken into account.
- **Memory footprint** - Memory (either volatile or non-volatile) is typically limited in IoT devices, and this is in constraint with the requirements of many ML approaches. For example, AlexNet [23]is a well-studied network for image classification. Indeed, it has 61M double-precision parameters for a single image, requiring about 250MB of memory. These numbers are clearly out of the possibility for many devices.
- **Availability of dedicated peripherals** - many tasks could be performed in a more efficient way (in terms of energy, speed, or even both) given the availability of a dedicated hardware block. An example could be hardware accelerators for neural networks, that are nowadays coming to interest, and many high-end devices can be found with some accelerators [24]. To consider a much more common example, low-power micro-controllers programmers have often to deal with a single-precision floating-point arithmetic unit, and in some cases with a fixed-point arithmetic unit only, in contrast with the requirements of many ML algorithms.

Here we limit ourselves on two critical aspects. The first one is the efficient arithmetic implementation on a standard micro-controller unit of arithmetic computations. Many algorithms considered in Section 5 require (independently of their implementation) many memory expensive matrix-vector multiplications, matrix-matrix multiplication, or even more complex multiplication. This may be a problem, both for speed and in particular for the huge memory requirement, opposed to the generally small quantity of available memory.
The second considered aspect is the optimization of a neural network architecture, that is a common structure used to implement ML algorithms. In particular, two aspects will be taken into account. The first one is the implementation of the DNN over a finite-precision arithmetic unit, as a double-precision unit is not always available in small, low-energy controllers. The second is the reduction of the number of parameters used by the DNN, a large ratio of them are typically redundant, and can be removed. This process is called *pruning* and, roughly speaking, is nothing more than a trade-off between a (generally) negligible cost in terms of performance drop and an important reduction (generally a decimation) of the number of parameters in the network.

6.1. Fast and memory efficient arithmetic on MCU

Many mathematical libraries are available for the efficient implementation of matrix operations. An example is the CMSIS-DSP library of the CMSIS project[1] developed for ARM micro-controllers. Some techniques employed to improve the efficiency of arithmetic operations are listed here.

- *Loop Unroll:* loops are massively employed in this kind of operations. As a consequence, loop-unrolling significantly increases the performance by removing unnecessary index updates, comparison and branches. Generally, this technique is automatically adopted by compilers when using some optimization flags.
- *Register blocking:* matrix-matrix multiplications are performed as a series of vector-vector dot products (involving the interested row and the interested column) performed through a sequence of multiply-and-accumulate operations. The output result is an entry of the output matrix, and require the reading (i.e., the transfer from the memory) of the whole row and of the whole column. By employing multiple accumulators (i.e., multiple temporary registers) simultaneously, it is possible to compute multiple output values at the same time involving a single row and multiple columns (or either multiple rows and a single column). This leads to a shorter number of memory accesses, with a shorter execution time and lower energy requirements.

[1] online repository https://github.com/ARM-software/CMSIS_5

- *Buffered multiplication:* in a matrix-matrix multiplication, one of the two input matrices may not be used anymore after the operation. In this case, the output matrix could be written in the same memory block used for the matrix that is not necessary anymore. Implementing this is relatively easy when a square matrix is involved, that is a quite common case. As an example, multiplying a $n \times k$ matrix by a $k \times k$ matrix gives rise to a $n \times k$ output matrix. Classical implementation requires a memory space of $(2n + k)k$ values for the storage of the three matrices, which, however, can be almost halved if $n \gg k$ by storing the output matrix in the same memory location of the $n \times k$ input matrix. The computation of each row of the output matrix requires the knowledge of corresponding row only in the rectangular matrix, and the full square matrix. After computing this row in a k-size temporary buffer, the corresponding row in the input matrix is not required anymore and can be overwritten. This overwriting procedure comes at the cost of a slight increase in the computation time.
- *Transposition of square matrices:* The transposition $(\cdot)^T$ of a rectangular matrix requires, in general, to copy the entire matrix in another memory space. In the case of a square matrix, transposition can be obtained overwriting the original matrix, and so removing unnecessary memory blocks, by swapping each value in the two triangular parts. As in the previous case, this comes at the cost of a slight increase in the computation time.
- *Transposed multiplications:* The operations AB^T and $A^T B$ can avoid the transpose operation by modifying the multiplication operation and scanning the transposed matrix row-first instead of column-first (or vice-versa).

6.2. Optimizing a neural network for a low-power MCU

Optimized mathematical libraries are available also for the efficient implementation of neural network. The CMSIS-NN library of the CMSIS project is specialized in the implementation of neural networks. Indeed, the inference of a NN-based algorithm is based on strongly structured matrix-matrix multiplication, with low possibility to introduce any sort of optimization.

Therefore, NN-based algorithms are usually optimized in other ways, limiting the execution energy, memory footprint, execution time, or in general hardware requirements at the cost of a small and controlled reduction in the algorithm accuracy. In other words, the same principle of the well-known lossy compression (accuracy *vs* complexity reduction) is applied here.

A first way to reduce complexity is to limit the precision of all parameters value to a given number of bits. This would allow to use fixed-point arithmetic, making not necessary anymore a single-precision or a double-precision arithmetic unit. However, in the general case, a straightforward rounding of all network parameters to the nearest low-precision representation usually implies a destructive drop in network performance.

Conversely, it is possible to exploit the redundancy of parameters in a neural network by employing quantization-aware training techniques. This would keep a good performance of the network in terms of accuracy when using quantized values in the network inference process. Among the many solutions proposed in the literature, two of them are worth mentioning, namely the fake-quantization [25] and the cosine regularization [26].

- *Fake-quantization* consists in the emulation of the loss of precision due to quantization, performed by applying quantization to parameters during the feed-forward phase of training. The update of the parameters is still performed with high (i.e., floating-point) precision, as small variations are required for the fine-tuning of the network.
- *Cosine regularization* is used instead to push the parameters – again, stored with high precision during training – near the quantized values allowed. For this purpose, a regularizer function is added during the training to the cost function in order to penalize values far away from the allowed, quantized levels, and promote values that instead are similar. In this way, the overall quantization error is reduced.

Weight quantization techniques where parameters are constrained to binary values represent the most hardware-friendly approach. The ultimate goal is *binary weight quantization*, where parameters can be memorized with a single bit value. Note that multiplication between two 1-bit values is nothing more than an XOR bitwise operation [27]. The exploitation of the inherent redundancy in the number of weights in a neural network for removing (i.e., zeroing) unnecessary parameters is known as *pruning*. A pioneering work of weight pruning is presented in [28]. Authors propose a novel class of layers based on the Multiply and Max-Min computation (MAM2) instead of the classical Multiply and Accumulate. These layer favorites pruning since after each multiplication between the input and the layer weights only the two contributions associated to the maximum and the minimum values contributes to the layer output. Weights never selected for the output computation can be discarded without any impact in the network functionality.

Considering the AE presented here, the adoption of MAM2 layers make possible the pruning of the 97% of the network parameters without any loss in the performance. For the same network, the adoption of standard MAC layer and

considering several pruning approach makes possible the zeroing of at least the 50% of the parameters without any loss of performance.

7. Conclusion

The topic of this work regarded the investigation, implementation and application of machine-learning based techniques for feature extraction and anomaly detection in satellite telemetries. The performance of four well-known ML algorithms, accordingly tuned for telemetry anomaly detection, were assessed in order to study the integration feasibility in on-board spacecraft health monitoring systems. A key point in this research is the decision criteria (anomaly scores, thresholds), used to improve the detection of anomalies and minimizing the false positive together. The observed results in terms of accuracy suggest that the ML models are worth deploying for the application of interest. Finally, we analyzed the possible methods to implement such ML-models in low-power microcontrollers, presenting different memory efficient approaches for arithmetic operations and a novel approach for NN layer optimization. Such optimization steps are required for limited computational power processors, that are at our disposal for satellite on-board processing.
In our research roadmap, further investigations and developments are planned, e.g. in terms of simultaneous monitoring of several telemetry quantities in the same window (multivariate time series analysis), scalability and reliability analysis of developed ML models, focusing on the optimization process needed to implement the ML-based methods in space qualified hardware with limited computational power.

References

[1] L. Spirkovska, D. Iverson, D. Hall, W. Taylor, A. Patterson-Hine, B. Brown, B. Ferrell e R. Waterman, «Anomaly Detection for Next-Generation Space Launch Ground Operations,» in *SpaceOps 2010 Conference*, 2010 .

[2] J. MacQueen, «Some methods for classification and analysis of multivariate observations,» 1967.

[3] S. Lloyd, «Least squares quantization in PCM,» in *IEEE Transactions on Information Theory*, 1982.

[4] F. Farbiz, Y. Miaolong e Z. Yu, «A Cognitive Analytics based Approach for Machine Health Monitoring, Anomaly Detection, and Predictive Maintenance,» in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020.

[5] I. Amihai, R. Gitzel, A. M. Kotriwala, D. Pareschi, S. Subbiah e G. Sosale, «An Industrial Case Study Using Vibration Data and Machine Learning to Predict Asset Health,» in *2018 IEEE 20th Conference on Business Informatics (CBI)*, 2018.

[6] L. Xiong, H.-D. Ma, H.-Z. Fang, K.-X. Zou e D.-W. Yi, «Anomaly detection of spacecraft based on least squares support vector machine,» in *Prognostics and System Health Managment Confernece*, 2011.

[7] V. Chandola, A. Banerjee e V. Kumar, «Anomaly Detection: A Survey,» in *ACM Comput. Surv.*, 2009.

[8] H. Ahn, D. Jung e H.-L. Choi, «Deep Generative Models-Based Anomaly Detection for Spacecraft Control Systems,» in *Sensors*, 2020.

[9] D. P. Kingma e M. Welling, «Auto-encoding variational bayes,» in *arXiv preprint arXiv:1312.6114*, 2013.

[10] S. Akcay, A. Atapour-Abarghouei e T. P. Breckon, «GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training,» in *arXiv*, 2018.

[11] D. Jung, J. Kwon, K. Baek e H. Ahn, «Attitude Control Simulator for the Korea Pathfinder Lunar Orbiter,» 2019.

[12] M. B. Markus, K. Hans-Peter, T. N. Raymond e S. Jorg, «Lof: Identify-,» in *Proceedings of the 2000 ACM SIGMOD International*, New York, 2000.

[13] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1999.

[14] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2007.

[15] S. Bernhard, P. John, S.-T. John, S. Alexander e W. Robert, «Estimating support of a high-dimensional distribution,» *Neural Computation*, p. 13:1443–1471, 2001.

[16] W. McKinney, «Data structures for statistical computing in python,» in *Proceedings of the 9th Python in Science Conference*, 2010.

[17] C. Harris, K. Millman, S. van der Walt e e. al., «Array programming with NumPy,» *Nature*, pp. 585, 357–362, 2020.

[18] A. Martín, A. Ashish, ,. P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean e e. al., «Tensorflow: A system for large-scale machine learning.,» in *In 12th Symposium on Operating Systems Design and Implementation*, 2016.

[19] Y. Zhao, Z. Nasrullah e Z. Li, «PyOD: A Python Toolboxfor Scalable Outlier Detection,» *Journal of machine learning research*, pp. 20(96), pp.1-7, 2019.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel e e. al., «Scikit-learn: Machine learning in Python.,» *Journal of machine learning research*, p. 2825–2830, 2011.

[21] M. Alex, E. Andriy, M. Mauro, R. Riccardo e S. Gianluca, «Anomaly detection based on compressed data: an information theoretic characterization,» 2021.

[22] M. Denil, S. Babak, D. Laurent, R. MarcAurelio e d. F. Nando, in *Advances in Neural Information Processing Systems*, 2013.

[23] A. Krizhevsky, S. Ilya e E. H. Geoffrey, in *Advances in Neural Information Processing Systems*, 2012.

[24] V. Sze, C. Yu-Hsin, Y. Tien-Ju e S. E. Joel, «Efficient Processing of Deep Neural Networks: A Tutorial and Survey.,» *Proceedings of the IEEE*, p. 105 (12): 2295–2329, 2017.

[25] V. Peluso e C. Andrea, «Energy-Driven Precision Scaling for Fixed-Point ConvNets,» in *IFIP/IEEE Int. Conf. Very Large Scale Integr.*, 2018.

[26] C. Song, L. Beiye, W. Wei, L. Hai e C. Yiran, «A Quantization-Aware Regularized Learning Method in Multilevel Memristor-Based Neuromorphic Computing System.,» in *IEEE 6th Non-Volatile Mem. Syst. Appl. Symp.*, 2017.

[27] S. Zhu, H. K. D. Luan e L. Weichen, «XOR-Net: An Efficient Computation Pipeline for Binary Neural Network Inference on Edge Devices,» in *IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 2020.

[28] P. Bich, L. Prono, M. Mangia, F. Pareschi, R. Rovatti e G. Setti, «Aggressively prunable MAM²-based Deep Neural Oracle for ECG acquisition by Compressed Sensing,» in *International Conference on Biomedical Circuits and Systems*, 2022.