Anomaly Detection Using Deep Learning Respecting the Resources on Board a CubeSat

Ross Horne* and Sjouke Mauw†
University of Luxembourg, L-4364 Esch-sur-Alzette, Grand Duchy of Luxembourg
Andrzej Mizera‡
IDEAS-NCBR, Chmielna 69, 00-801 Warsaw, Poland
André Stemper§
University of Luxembourg, L-4365 Esch-sur-Alzette, Grand Duchy of Luxembourg
and
Jan Thoemel¶
University of Luxembourg, L-1359 Luxembourg, Grand Duchy of Luxembourg

https://doi.org/10.2514/1.I011232

We explore the feasibility of onboard anomaly detection using artificial neural networks for CubeSat systems and related spacecraft where computing resources are limited. We gather data for training and evaluation using a CubeSat in a laboratory for a scenario where a malfunctioning component affects temperature fluctuations across the control system. This data, published in an open repository, guides the selection of suitable features, neural network architecture, and metrics comprising our anomaly detection algorithm. The precision and recall of the algorithm demonstrate improvements as compared to out-of-limit methods, whereas our open-source implementation for a typical microcontroller exhibits small memory overhead, and hence may coexist with existing control software without introducing new hardware. These features make our solution feasible to deploy on board a CubeSat, and thus on other, more advanced types of satellites.

I. Introduction

SPACECRAFT have resource constraints and are subject to component failures best monitored on board due to datalink constraints. For CubeSats [1,2] that support a "low-cost and fast-delivery" paradigm [3], computational resources are particularly limited. CubeSats are accountable for the high percentage of failed missions in low Earth orbit (LEO), partly due to both the use of cheap components and to the low-cost processes of production and verification. The focus of this work is to evaluate the feasibility of improving fault detection on board CubeSats, and related spacecraft, using artificial neural networks (ANNs) by respecting onboard computational limitations.

The aforementioned focus is motivated by the following regulatory reasons. First, due to launch trends, there is increasing concern from launching states about liability issues stemming from conjunctions in LEO. Because, under current international treaties, the launching state itself is liable if a satellite originating from its territory is deemed to be at fault in a conjunction, there is a tendency toward states limiting their liability by requiring that operators are covered by insurance [4]. Second, although the presented developments are relevant to spacecraft in general, we prioritize CubeSats because there is an explicit International Organization for Standardization standard for CubeSats [5], imposing fewer reliability requirements than normal for spacecraft. This leaves a gap between the deregulation advocated by the engineering standard, which is intended to stimulate innovation, and the demands of an insurer. Insurers will increasingly audit steps taken to reduce the possibility of CubeSat failure resulting in creating a dangerous object in LEO, hence auditable verification and system health monitoring of CubeSats may well become essential requirements. In this work, we introduce onboard ANNs to improve anomaly detection and reliability without imposing additional hardware constraints. In this way, we aim to stimulate innovation improving regulatory compliance of CubeSat missions.

There is also a practical reason for considering CubeSats: specifically, that the project team has direct physical access to a CubeSat; i.e., the EduSat of the CubeSat Lab at the University of Luxembourg. The insight gleaned may be translated to other space missions with similar resources and regulatory requirements. To ensure that such missions are targeted, a requirement of this study is that solutions are implementable on a typical CubeSat microcontroller, respecting its limited memory and computational power while leaving space for control software. More specifically, within the aforementioned broader objective, we have focused on a case study that involves anomalies in temperature readings that fall out of the scope of simple static threshold triggers, and we instead demand multivariate analysis of multiple components. Such a multivariate analysis is suited to ANNs.

For this investigation, we have selected a narrow case study in order to obtain datasets for training, testing, and evaluating possible onboard ANN solutions. In nominal conditions, a CubeSat performs a repeated set of health and safety, communications, and data collection tasks. We can assume that these tasks are performed either constantly or periodically due to the periodic orbital motion of the CubeSat around the Earth. Moreover, due to natural orbital motion, the spacecraft is periodically illuminated by the sunlight, which introduces cyclic changes to the temperature values recorded by numerous temperature sensors used to monitor various CubeSat components. Put together, possibly with the exception of some specific dawn/dusk orbits that are minimally affected, there exists a periodic *complex thermal pattern* manifested in the relative timing of temperature value changes measured by individual sensors.

The scenario we investigate aims to capture potential malfunctioning of CubeSat board component(s) manifested in subtle perturbations

Received 19 December 2022; revision received 29 June 2023; accepted for publication 29 June 2023; published online Open Access 23 August 2023. Copyright © 2023 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Postdoctoral Researcher, Faculty of Science, Technology and Medicine, Department of Computer Science, 6, Avenue de la Fonte; ross.horne@uni.lu.

†Full Professor, Faculty of Science, Technology and Medicine, Department of Computer Science, 6, Avenue de la Fonte; sjouke.mauw@uni.lu.

‡Postdoctoral Researcher, Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland; andrzej.mizera@ideas-ncbr.pl (Corresponding Author).

§Research Support Engineer, Faculty of Science, Technology and Medicine, 2, Avenue de l'Université; andre.stemper@uni.lu.

¶Research Scientist, Interdisciplinary Centre for Security, Reliability and Trust, 6, rue Richard Coudenhove-Kalergi; jan.thoemel@uni.lu.

to the norm of the complex temperature pattern inherent to the operation of the satellite in LEO. We consider abnormal situations where the malfunctioning of any element on board results in an increase of its temperature. Although the malfunctioning element itself may not be equipped with a temperature sensor, its abnormal state generates additional heat that is transferred via the mechanisms of thermal conduction (mainly) and thermal radiation (to some limited extent) across the board. Under abnormal conditions, the additional heat introduces disruptions to the characteristic thermal pattern. The proposed solution is to identify subtle changes in the thermal pattern and report them as possible indicators of the malfunctioning of elements on the CubeSat board.

On the CubeSat that we used for the experiments, various board components are equipped with temperature sensors, i.e., four maximum power point trackers (MPPTs), four voltage converters, and one battery monitor, which provide real-time measurements across the board. The placement of the nine sensors is schematically shown in Fig. 1, where the color gradient illustrates simulated temperature distribution during heating by an external source placed in the plane of the board (not shown).

Because our solution is centered on analyzing the complex thermal pattern, it constitutes a step toward the development of a comprehensive health monitoring capability in the sense that it is not limited to a particular CubeSat board component but aimed at detecting anomalies in the operation of any of them. Yet, by focusing on such a specific scenario, we aim to be able to explain the occurrence of anomalies as the appearance of additional heat sources on the CubeSat board.

In Sec. II, we explain our data collection methodology required for training and evaluation. Section III describes the ingredients comprising our anomaly detection algorithm: notably, the features selected, the choice of neural network architecture, and the metric used for anomaly detection. Section IV evaluates the choices of parameters in our architecture to find those that are effective for detecting anomalies in our dataset. Section V evaluates the overhead of our implementation of the algorithm on an STM32H743 32 bit ARM Cortex-M7 microcontroller, which is typically deployed on board a CubeSat.

II. Data Collection Methodology

We make explicit here the engineering model and experimental setup used to generate telemetry from a CubeSat for the purpose of evaluating our neural network solutions. The CubeSat used is a NanoAvionics flightlike engineering model containing an ultra high frequency communications system, an electrical power system (EPS), an attitude control sensor and actuators, and an onboard flight computer with an attitude control algorithm. The model is referred to as the EduSat, and it is depicted in Fig. 2. The EduSat uses the CubeSat protocol [6], which supports a distributed architecture where subsystems are addressable nodes. It is a flightlike unit featuring a high number of typical CubeSat sensors and actuators, albeit reduced in number for cost-efficient education and research.

To simulate the in-orbit conditions of a CubeSat in LEO, the following experimental setup was devised. The EduSat was placed on a *rotation table* configured with a rotation period of 90 min., representing a typical time for a satellite in LEO to complete a full orbit around the Earth. Dedicated software controlled the rotation table engine while logging the time and actual angular position of the EduSat. The EduSat buffer capacity was 1340 telemetry entries, and so partial data was systematically downloaded and stored on a computer, and then later merged into a single telemetry data time series.

The laboratory environment differs from the orbit environment by the presence of ambient air, which provides convective cooling. This heat transport mechanism results in overall lower and less extreme temperatures in the EduSat. The magnitude of the temperatures is not important for our algorithm because we detect patterns and not the exceeding of a limit. The less pronounced features as compared to those expected in orbit are more challenging to a detection algorithm, giving confidence in its functionality.

To mimic the periodical illumination of a CubeSat by sunlight, EduSat was illuminated by two continuous light sources, which were Godox SL100Bi and Neewer SL-60W, that were set to 80% brightness with 6500 K, and 100% brightness with 5600 K, respectively. The light sources were focused at the height of the EduSat's board. This setup provides a representative heating, enabling the development of our algorithm. A full duplication of the thermal in-orbit situation was not attempted because it was not necessary for the temperature pattern anomaly detection. To keep the room temperature stable, the experiments were conducted in a basement room with highly limited sunlight access and no other light sources. The experimental setup is shown in Fig. 3. Room temperature variations could not be fully eliminated, and so the room temperature was constantly monitored and logged during experiments. The experiments mimicking in-orbit conditions consisted of three phases, which are explained as follows:

1) The first phase is Phase I. During this initial phase, the light sources were heating up the EduSat while the rotation was off in order for the EduSat to reach a stable experiment base temperature.

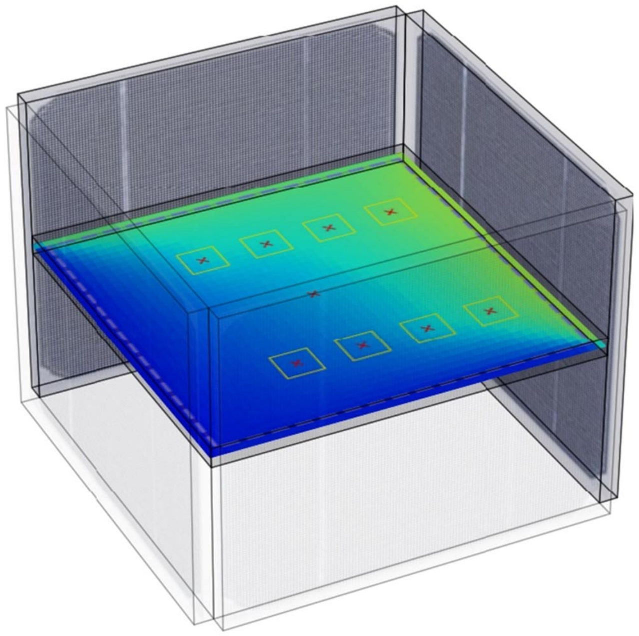


Fig. 1 Schematic illustration of the nine temperature sensors' placements on the CubeSat board indicated with red crosses framed with yellow squares.

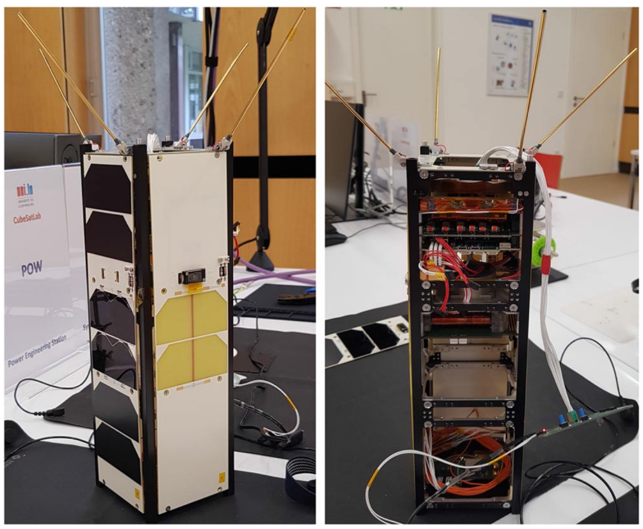


Fig. 2 The EduSat (NanoAvionics Engineering Model, EM) of the CubeSat Lab at the University of Luxembourg. The photograph on the right presents the interior of the EduSat.

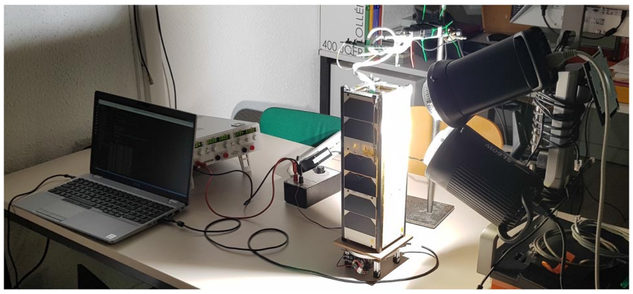


Fig. 3 The experimental setup for data acquisition. Two light sources illuminate the EduSat, which is placed on the rotation table.

2) During Phase II, the EduSat was rotating while being illuminated by the light sources, and the normal conditions' data was collected for a few rotations.

3) The third phase is Phase III. During this final phase, the anomalies were being introduced by turning on battery heating of specified power and duration, which was generating abnormal-conditions data.

During Phase II, the CubeSat was rotating with a constant angular velocity while being illuminated by the light sources. In this sense, Phase II represents oscillatory stable state modulo noise generated by external factors, which could not be fully eliminated due to our limited resources and imperfect laboratory conditions. To make sure that Phase II represented an oscillatory stable state, Phase II was preceded by Phase I. During Phase I, the temperature pattern was monitored although the data was discarded; i.e., the data from Phase I was never used in the training/validation/testing of our approach.

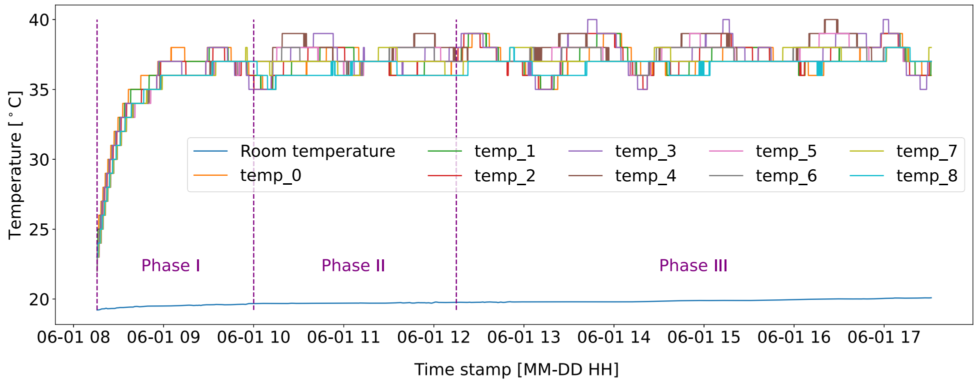
To simulate anomalies in Phase III, battery heating was used to imitate additional heating sources on board that could appear due to malfunctioning of some board component. The power and duration of the battery heating were chosen in such a way that only subtle distortions to the thermal pattern were introduced, which could not be identified by a simple temperature threshold per component. If the heating was too strong in terms of duration and/or power, it would produce visible peaks in the plots. It would then be easy to firmly determine the occurrence of thermal anomalies by choosing a proper threshold value and checking whether the temperature values for a given component were above the threshold.

However, if the temperatures periodically cover the whole nominal range and the anomalies are weak enough to not increase the temperatures to exceed this range, then the thresholding approach is not of use any more. In such circumstances the anomalies can only be detected by the identification of some disturbances to the complex nominal thermal pattern emerging from the variations in the states of the components during the nominal functioning of a satellite as captured by the different sensors.

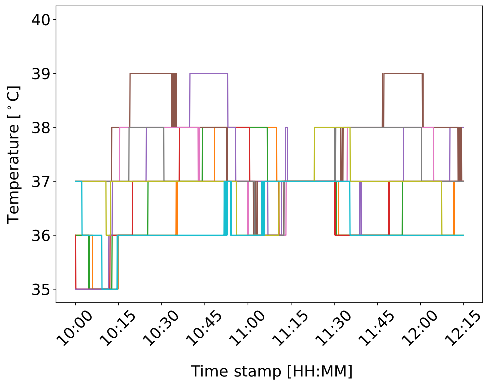
In our study, the proper settings of battery heating power and duration were obtained by the empirical trial-and-error approach. Sufficient time was assured after each introduced anomaly for the EduSat to revert to a normal condition before subsequent anomalies were introduced.

For each experiment, data corresponding to Phase I was excluded; and the remaining time series was split into two datasets: one containing the normal condition (nominal) data of phase II, and the other containing the abnormal (also referred to as anomalous) data of Phase III. One exception is the experiment of 2022/07/20, in which Phase II was not present. This was dictated by the aim to generate abnormal data with longer recovery periods after anomalies; hence, we decided to skip phase II in this case. Examples of normal and abnormal datasets from an experiment are presented in Fig. 4, where each plot shows the readings of the nine EduSat board temperature sensors in time. The resolution of the temperature sensors was 1°C.

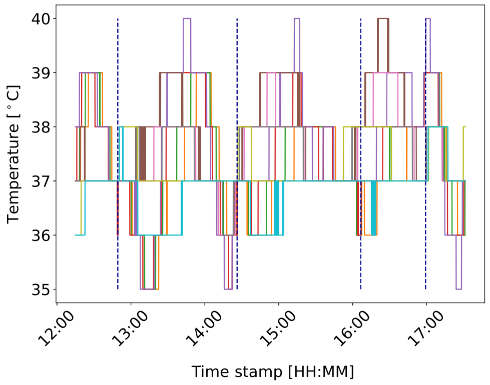
Table 1 summarizes the information on the individual experiments, during which datasets were generated for the training and testing of the models. For each anomaly, the battery heating power and heating duration time are provided in the order of appearance and separated



a) Full dataset containing nine temperature sensor values; blue line in the bottom presents the logged room temperature



b) Normal dataset of the experiment



c) Abnormal dataset of the experiment; starts of anomalies are indicated with dashed dark blue vertical lines

Fig. 4 Exemplary datasets generated in a single experiment. The respective colors for the temperature sensors are the same in all plots.

Table 1 Information on individual data acquisition experiments

Experiment date	No. of anomalies	Battery heating power and time	Frequency, Hz	Nominal data
2022/04/06	3	100%, 60 s ǀ 25% 60 s ǀ 100%, 10 s	0.1	✓
2022/05/18	3	100%, 15 s ǀ 100%, 15 s ǀ 100%, 15 s	0.1	✓
2022/05/20	1	100%, 15 s	0.2	✓
2022/05/30	1	100%, 20 s	0.2	✓
2022/06/01	4	100%, 20 s ǀ 100%, 20 s ǀ 100%, 20 s ǀ 100%, 20 s	0.2	✓
2022/06/03	1	100%, 30 s	0.2	✓
2022/06/08	2	100%, 30 s ǀ 100%, 30 s	0.2	✓
2022/06/15	2	100%, 60 s ǀ 100%, 60 s	0.2	✓
2022/06/22	2	100%, 60 s ǀ 100%, 60 s	0.2	✓
2022/07/20	4	100%, 60 s ǀ 100%, 60 s ǀ 100%, 60 s ǀ 100%, 60 s	0.2	✗

by the "ǀ" delimiter. The frequency column presents the data acquisition frequency used in each experiment. The nominal data column indicates whether phase II was considered in the respective experiment, and hence whether nominal data was generated during the experiment: a checkmark (✓) denotes yes, and a cross (✗) denotes no. An additional experiment, not shown in Table 1, was conducted in order to collect a normal dataset for model validation purposes. We refer to this dataset as the *validation dataset*.

All data generated within this study is made publicly available in the AtMonSat project GitHub repository [7]. In particular, the individual datasets can be accessed using the dataset class of the *dataset.ipynb* Jupyter notebook; see Supplemental Appendix S.B.A.1 for further details.

III. Design of Algorithm Based on Deep Learning

In this section, we elaborate on why we target deep learning rather than traditional out-of-limit (OOL) techniques, and we highlight associated challenges and solutions. Key challenges associated with deep learning in this setting are that the depth and input space of the networks are limited by resource constraints, and that the quantity of data for training is limited. Therefore, to classify normal behaviors, we found it was necessary to select features in order to guide the training rather than relying entirely on deep learning to automatically extract features. We subsequently show our algorithm is capable of extracting selected features from telemetry in real time on board a microcontroller.

A. Background on OOL and Artificial Intelligence Methods

Health monitoring and real-time detection of any symptoms of anomalous behavior in the multivariate telemetry data are important tasks in the operation of satellites [8]. The traditional and commonly used method for this purpose is the OOL technique, in which sensor measurements are checked individually to determine whether they are within predefined ranges [9–11]. However, even the most sophisticated OOL methods fail to detect complex patterns in spacecraft flight data generated by variations in the state of components during the nominal functioning of a satellite, and they are therefore bound to miss many anomalies [11]. Moreover, OOL approaches are not capable of detecting *novel behaviors*, i.e., events that are novel with respect to a set of behaviors known to be nominal, for which onboard data measurements (or their differences) are within the defined OOL thresholds [12,13]. However, novel behaviors are often early indicators of upcoming anomalies and failures. In this sense, OOL techniques do not allow forthcoming problems to be anticipated [13]. Furthermore, it is very costly to develop and maintain the sets of rules of OOL anomaly detection systems because it requires the use of expert knowledge [8].

In recent years, data-driven or learning-based methods that mine relevant information from large datasets have provided breakthroughs in numerous fields. In the context of spacecraft operation, methods based on machine learning or artificial intelligence (AI) techniques offer some effective approaches in the way telemetry data is exploited in the context of anomaly root cause analysis and novelty detection [13]. In particular, AI-based approaches are effective in extracting patterns and correlations in intertwined streams of telemetry data [14]; this data includes many aspects, such as high-dimensionality, multimodality, and heterogeneity [8]. In fact, some experts anticipate that spacecraft operations are an area where AI-based methods can provide the highest benefits in the space engineering domain [13].

Our problem scenario, described in Sec. I (where a component failure propagates as a pattern of temperature changes across the board), has been selected because it is a concrete instance of a novel behavior suited to the AI-based solution we devise in this section. Several resource constraints have been imposed on the proposed anomaly detection algorithm because deployment to a microcontroller is targeted. In particular, it was necessary for the algorithm to fit into limited memory available on the microcontroller, to be capable of performing real-time inference with the restricted target computational resources, and to operate in the allowed energy consumption range. Memory is particularly pertinent when deep ANNs are deployed, due to the large number of parameters for each layer. Details on the available microcontroller's resources are provided in Supplemental Appendix S.B.E, and the results of the measurements of actual usage of the resources by the microcontroller implementation of our anomaly detection solution are presented in Sec. V.

B. Feature Selection and Engineering

Feature selection is a crucial step in the development of AI-based frameworks. We explain here what raw sensor data we select and how we engineer features from that raw data to indicate the rate of change of sensors in such a way that changes relative to other sensors are also accounted for. The intention is that we obtain a compact feature from which differences in the propagation of temperature changes across the board can be detected by a suitable ANN.

For our problem scenario, nine of the 500 general telemetry attributes of the EduSat are selected, from which we engineer features. Those attributes are the EPS general telemetry attributes corresponding to temperature sensors of the battery monitor, four MPPTs, and four voltage converters on the EduSat board. These telemetry attributes form the first set of features, i.e., the *raw temperature sensors'* features, denoted T_j for $j \in [1, \dots, 9]$.

The raw readings from the nine temperature sensors are used to engineer a second set of nine features (one for each sensor), which is intended to indicate to the ANN the rate of change of each sensor. The engineered features are nine counters for each data point (i.e., readouts from the nine sensors) that each record the number of data points since the last temperature change for the respective sensor. Due to the fact that the arrival of each data point initiates the next iteration of the anomaly detection algorithm in the target implementation on the microcontroller, data points are also referred to as *iterations*, and the set of engineered features is named as the *iterations since the last change* (ISLC) features. The features are denoted by $islc_{T_j}$ for $j \in [1, \dots, 9]$. Formally, we define $islc_{T_j}[0] = 0$ and, for $i \geq 1$, we have

$$islc_{T_j}[i] = \begin{cases} islc_{T_j}[i-1] + 1, & \text{if } T_j[i] = T_j[i-1] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

For example, in Table 2, the raw temperature values recorded by four imaginary sensors $T_{1\text{–}4}$ are presented, where the events of

Table 2 Example of ISLC feature engineering

	T_1	T_2	T_3	T_4	$islc_{T_1}$	$islc_{T_2}$	$islc_{T_3}$	$islc_{T_4}$	$islc^{int}_{T_1}$	$islc^{int}_{T_2}$	$islc^{int}_{T_3}$	$islc^{int}_{T_4}$
0	37	38	38	39	0	0	0	0	0.00	0	0	0
1	37	38	38	39	1	1	1	1	**0.00**	1	1	1
2	$\underline{38}$	38	38	39	0	2	2	2	0.00	2	2	2
3	38	38	38	39	1	3	3	3	1.00	**1**	**1**	3
4	38	$\underline{39}$	$\underline{39}$	39	2	0	0	4	2.00	0	0	4
5	38	39	39	39	3	1	1	5	**1.50**	1	1	5
6	38	39	39	39	4	2	2	6	**1.00**	2	2	6
7	38	39	39	39	5	3	3	7	**0.50**	3	3	7
8	$\underline{39}$	39	39	39	0	4	4	8	0.00	4	4	8
9	39	39	39	39	1	5	5	9	1.00	5	5	**4**
10	39	39	39	$\underline{38}$	2	6	6	0	2.00	6	6	0
11	39	39	39	38	3	7	7	1	**1.33**	7	7	1
12	39	39	39	38	4	8	8	2	**0.67**	8	8	2
13	$\underline{40}$	39	39	38	0	9	9	3	0.00	9	9	3
14	40	39	39	38	1	10	10	4	1.00	10	10	4

temperature change are indicated. The respective ISLC features are presented in columns $islc_{T_{1\text{–}4}}$, where a zero value entry indicates the event of temperature change in comparison to the previous temperature value for the corresponding temperature sensor.

To strongly connect local changes to changes across the board, certain ISLC subsequences are replaced with linearly interpolated values, as we explain here. Let us consider the situation where $islc_{T_j}[i] = 0$ for some $j \in [1, \dots, 9]$ and $i > 0$. Let k be the number of iterations to the previous temperature change event among all the sensors, i.e.,

$$k = \min(\{islc_{T_j}[i-1] + 1\} \cup \{islc_{T_l}[i] | l \in [1, \dots, 9] \wedge l \neq j\}) \tag{2}$$

Then, values $islc_{T_j}[i - k + l]$ for $1 \leq l < k$ are replaced with linearly interpolated values between $islc_{T_j}[i - k]$ and zero, i.e.,

$$islc^{int}_{T_j}[i - k + l] = islc_{T_j}[i - k] - l * (islc_{T_j}[i - k]/k) \tag{3}$$

for $1 \leq l < k$. For an example, see columns $islc^{int}_{T_{1\text{–}4}}$ in Table 2, where interpolated values are highlighted. The interpolation gives rise to *interpolated ISLC* features denoted $islc^{int}_{T_j}$ for $j \in [1, \dots, 9]$. We observed in our experiments that the interpolation is crucial for the considered ANNs to properly train and perform on the anomaly detection task.

The use of interpolation introduces an inherent delay to real-time anomaly detection. The classification of whether the next data point is anomalous or not cannot be provided immediately upon arrival of the data point. Therefore, in order to compute the corresponding interpolated ISLC values, the new data point needs to be stored together with subsequent data points in a dedicated queue until the moment when a data point with a temperature change on one of the sensors is received. At that point, the interpolated ISLC values can be computed, the anomaly detection algorithm can be run for all the data points in the buffer, and finally the buffer can be freed.

For example, let us consider the entries in columns $T_{1\text{–}4}$ in Table 2 with index 5 as the next data point that is available to the anomaly detection algorithm. No temperature change takes place with respect to the data point indexed as four. Therefore, the execution of the anomaly detection algorithm for this data point is postponed. Upon arrival of the data point with index 8, the values in columns $islc^{int}_{T_{1\text{–}4}}$ can be computed and the anomaly detection algorithm run for points with indices 5–8. Nevertheless, our experiments show that the numbers of subsequent data points buffered are usually not large. A histogram showing the distribution of these numbers obtained when computing the interpolated ISLC features for all normal datasets of the experiments in Table 1 is shown in Fig. 5. In our case, the

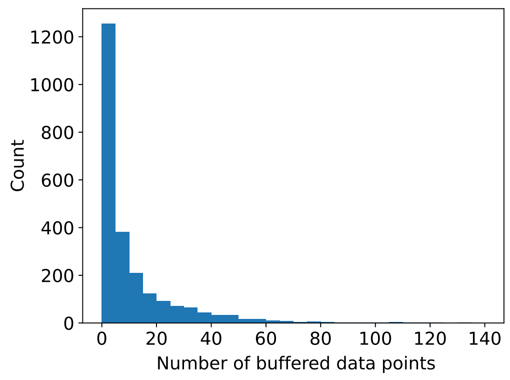


Fig. 5 Histogram of the numbers of data points buffered before interpolated ISLC values could be computed for the normal datasets.

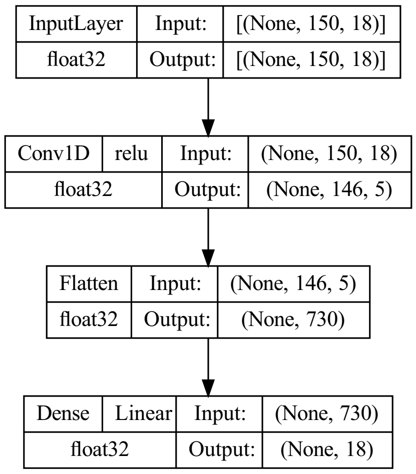
maximum value is 133. Notice that this number may be different for another set of experimental data.

At first sight, one could argue that the delay in anomaly detection caused by the need to buffer 100 data points is too long: with a data acquisition frequency of 0.2 Hz, this would result in a delay of 500 s, i.e., 8 min. and 20 s. However, for our particular scenario, the anomalies are assumed to manifest themselves via temperature changes. Therefore, if an anomaly fitting the scenario occurs, it will cause a temperature change, which in turn will cause the interpolated ISLC values to be computed and the anomaly detection algorithm to be triggered to detect the abnormal behavior of the system. Therefore, we consider this inherent delay to be only a minor disadvantage of the proposed solution because the delay will only be high if the satellite is in a stable state rather than an anomalous state.

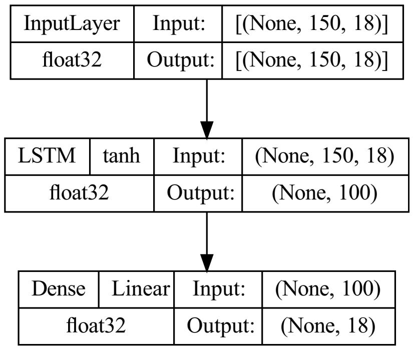
To summarize, both the nine raw temperature sensors features and the nine interpolated ISLC features were used simultaneously for model training and inference. We discuss further the implementation of the algorithm, including how the ISLC interpolation is implemented on the microcontroller for real-time analysis, in Sec. V.

C. Deep-Learning Model Architectures

We considered different deep-learning model architectures: convolution neural network-based (CNN-based), long short term memory-based (LSTM-based), and an autoencoder with different configurations of activation functions and individual layers. As we explain next, only two of the considered architectures provided relevant results, i.e., a CNN-based architecture and an LSTM-based architecture presented in Fig. 6. The remaining two architectures, which demonstrated unsatisfactory anomaly detection results, are shown for the sake of the completeness of presentation in Supplemental Figure S1 in Supplemental Appendix S.A.


InputLayer | Input: | [(None, 150, 18)]
float32 | Output: | [(None, 150, 18)]
Conv1D | relu | Input: | (None, 150, 18)
float32 | Output: | (None, 146, 5)
Flatten | Input: | (None, 146, 5)
float32 | Output: | (None, 730)
Dense | Linear | Input: | (None, 730)
float32 | Output: | (None, 18)

a) CNN-based model (13,613 parameters)


InputLayer | Input: | [(None, 150, 18)]
float32 | Output: | [(None, 150, 18)]
LSTM | tanh | Input: | (None, 150, 18)
float32 | Output: | (None, 100)
Dense | Linear | Input: | (None, 100)
float32 | Output: | (None, 18)

b) LSTM-based version 1 model (49,418 parameters)

Fig. 6 Deep-learning model architectures with configurations that demonstrated relevant anomaly detection results. Graphs were produced with tf.keras.utils.plot_model function of TensorFlow, which uses "None" to represent unknown batch size, which is determined first when the model is executed. (Conv1D, 1-dimensional convolution layer; relu, rectified linear unit activation function).

We selected the approach of training deep-learning models exclusively with the normal datasets (i.e., regression training) as an alternative version of training the models using both normal and abnormal datasets (i.e., classification type of training). All models were trained with the "mean squared error" (MSE) loss function and the Adam optimizer with TensorFlow default settings. Early stopping callback was used with the following configuration: *monitor*="loss," *min_delta*=$1e^{-2}$, *patience*=10, and *mode*="auto." The number of epochs was set to 500. The input to the models consisted of batches of input tensors of shape (*window_length,number_of_features*), where window_length defines the size of the number of consecutive (historical) data points used to make the inference. For the target model, which is presented in the following, we set *number_of_features*=18. The particular choice of the *window_size* hyperparameter value in the target model used in our anomaly detection algorithm is presented in Sec. IV.A. The output of the models are tensors of shape (*1,number_of_features*); i.e., the models infer a single next data point in terms of interpolated ISLC values and raw temperature values. During training, the inferred outputs were compared to the actual data points using the MSE loss function, which was optimized with the backpropagation algorithm.

D. Anomaly Detection Algorithm

We now explain the anomaly detection algorithm we developed based on an ANN trained on normal data. The algorithm makes use of a metric for calculating errors, which turns out to be an important design decision. The two metrics we considered were 1) the Euclidean distance and 2) the Mahalanobis distance between the two vectors of length number_of_features, which is defined with respect to the estimated mean and inverse of a covariance matrix. The anomaly detection algorithm is devised as follows:

1) In step 1, a deep-learning model is trained on data from all the normal datasets generated with the experiments presented in Table 1 with the settings presented in Sec. III.C. If the Mahalanobis distance is employed, its mean vector and covariance matrix are estimated at this point from the normal condition error vectors obtained by running the trained model on the normal training data and subtracting the prediction from the reference at each data point.

2) In step 2, the trained model is then run on an anomalous dataset. For each inferred point, an error with respect to the actual data point is computed using the chosen metric (the Euclidean or Mahalanobis distance). The errors, which are computed based on the model outputs and the ground-truth feature vectors, are referred to as the *raw anomaly detection signal*.

3) In step 3, the raw anomaly detection signal is postprocessed by considering a threshold value and a so-called *holdoff window*. The threshold value is determined with a separate procedure described in Sec. III.E, whereas the holdoff window is given by a duration specifying the number of time points. The raw anomaly detection signal is scanned in the chronological order of the corresponding data points, one point after the other. If an error value greater than or equal to the threshold value is observed, the corresponding data point is classified as anomalous and all consecutive data points within the holdoff window starting at the anomalous data point are classified as normal. Scanning then continues from the first subsequent error outside the holdoff window.

The postprocessing in step 3 is introduced due to the fact that the information on abnormal behavior is carried in the change of the temperature pattern. However, temperature changes are inherently slow and continuous. Furthermore, there is a delay between the occurrence of a board element malfunction and the time the disturbance in the temperature pattern is registered by the sensors scattered across the board. The delay is determined by the thermal conduction characteristics of the board. Given this, the rationale behind introducing the holdoff window for the postprocessing of the raw anomaly signal is twofold. First, due to continuity of thermal changes, an anomaly is usually indicated by consecutive raw signal values exceeding the threshold. Second, an anomaly can be demonstrated by the occurrence of multiple anomaly signal peaks, one shortly after the other, as observed in some of the generated datasets. The introduction of the holdoff window postprocessing allows all data points exceeding the threshold in a window to be considered as a single anomaly. This cleans the raw anomaly signal for subsequent quantitative evaluation of the performance of the models with respect to appropriate metrics described in Sec. IV. In our case, we set the holdoff window to 60 time points.

E. Determination of the Anomaly Detection Threshold

Both the Euclidean and the Mahalanobis error thresholds are determined with the same approach that mimics the statistical leave-one-out cross-validation (also known as out-of-sample testing) procedure. The experiments presented in Table 1 are used to train a model with the settings provided in Sec. III.C. Then, the fitted model is run on the separate validation dataset, and the errors are computed and stored. For the Mahalanobis distance, the mean and the inverse covariance matrix are estimated from the errors obtained by running inference on all the normal datasets except the validation dataset. The procedure is repeated for all normal datasets kept aside. Finally, all the errors are used to generate a histogram (i.e., an empirical distribution), and the significance threshold value corresponding to a chosen statistical significance level (e.g., 0.05 or 0.01) is used as the respective Euclidean or Mahalanobis error threshold; see Fig. 7.

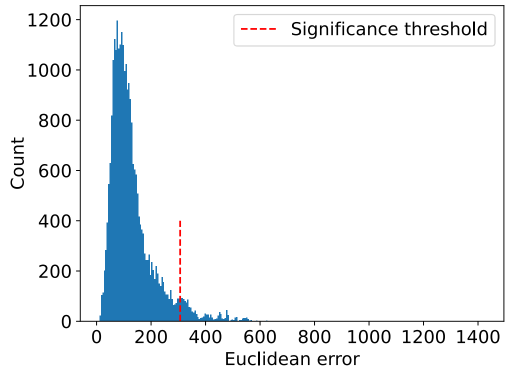


Fig. 7 Empirical distribution of Euclidean errors inferred by the CNN-based model. Significance threshold corresponds to the 0.05 statistical significance level.

IV. Evaluation of the Effectiveness of the Algorithm

This section details the results obtained concerning the effectiveness of the algorithm devised with respect to key parameters. We explain why experiments lead us to clearly prefer the Mahalanobis metric rather than the Euclidean metric. More surprisingly, the CNN architecture significantly outperforms other considered architectures, which we justify via a statistical analysis of data points classified as anomalous. Statistical analysis also justifies that our optimal architecture significantly outperforms an OOL method on most datasets.

A. Evaluation of Hyperparameters Impacting Anomaly Detection

The model described in Sec. III can be configured using several parameters, including the choice of metric and ANN, which we evaluate here to determine which are most effective. To evaluate the choice of metric with respect to a given ANN, the resulting anomaly detection algorithm employing trained models was applied to the anomalous datasets of the experiments presented in Table 1.

The quantitative evaluation of the algorithm performance is conducted using a confusion matrix consisting of the true positives (TPs), false positives (FPs), and false negatives (FNs), as determined by the output of our anomaly detection algorithm. From this, we calculate the values of three standard metrics for classification evaluation, i.e., *precision*, *recall*, and the *F1score*, where

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \qquad \text{and}$$
$$\text{F1 score} = \frac{2\cdot\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}} \tag{4}$$

To count the TPs, we introduce the notion of a TP interval. Let a_i be the index of the first data point recorded by the temperature sensors after (or at) the anomaly start time, i.e., the moment of battery heating being turned on. Because the heating anomaly requires time for the temperature change to reach the sensors, there is an inherent delay, or inertia, before the anomaly is detectable. We therefore allow the anomaly to be identified within a window of data points, which is referred to as the *anomaly inertia window*. In our experiments, we set the length of this window, denoted l_{aiw}, to 60 data points based on our observations made during experimentations. If an anomaly signal is output by the anomaly detection algorithm for any of the data points with indices in the range $[a_i, a_i + l_{\text{aiw}}]$, the anomaly signal is considered as a TP. However, due to the interpolation of the ISLCs, there is a possibility for an anomaly to be detected and identified before the data point with index a_i because the information about the near future is conveyed in the interpolated subsequence of ISLC(s). For example, the subsequence $islc^{int}_{T_1}[4-6]$ in Table 2 contains interpolated values. By observing it, one can deduce that a temperature change will be captured by the temperature sensor T_1 in the eighth data point. To take this possibility into account, the TP interval is expanded to the left as follows. We check whether any of the nine interpolated ISLC features corresponding to the a_ith data point contain an interpolated value. If this is the case, let int_{start} and int_{end} be the start and the end indices of the interpolated subsequence, respectively. In our example, $int_{\text{start}} = 5$ and $int_{\text{end}} = 7$. Then, the TP interval is given by $[int_{\text{start}}, a_i + l_{\text{aiw}}]$. All anomaly detection signals within this interval are considered as TPs. In Supplemental Fig. S8, each anomaly is plotted with its TP interval visualized with light-violet rectangles in the plots with the final output of the anomaly detection algorithm.

Once the TPs are calculated, all the remaining anomaly signal peaks are counted as FPs. Anomalies that are not identified contribute to the number of FNs. Finally, the number of true negatives (TNs) is obtained with the following expression:

$$\text{TN} = \text{total number of data points} - (\text{TP} + \text{FPs} + \text{FN})$$

Next, the precision, recall, and F1 score are calculated in accordance with Eq. (4). The precision and the recall make it possible to assess the performance of a classifier on the minority class of an imbalanced dataset [15] because both metrics are unconcerned with the majority class and only focus on the minority class. In our case, the minority class is "abnormal" (i.e., TPs), whereas the majority class is "normal" (i.e., TNs).

Our experiments revealed that only the use of the Mahalanobis distance metric generated meaningful results. Furthermore, only the CNN-based model outlined in Fig. 6a was capable of providing meaningful results for the noisy real-life data. Surprisingly, the two larger model architectures presented in Supplemental Fig. S1 in Supplemental Appendix S.A were not capable of properly identifying anomalies in our anomalous datasets despite numerous trials with different settings of hyperparameters (data not shown). However, the case of the LSTM-based version 1 model in Fig. 6b was special. The raw anomaly detection signal output by the anomaly detection algorithm using this model for the 2022/03/06 abnormal dataset is shown in Fig. 8. The LSTM-based model failed to identify the anomaly due to the Mahalanobis error threshold being too high, which is indicated by the dashed green line. The high value of the threshold originated from poor performance of the model on the out-of-sample test with the 2022/05/30 normal dataset kept aside. Given this threshold, the

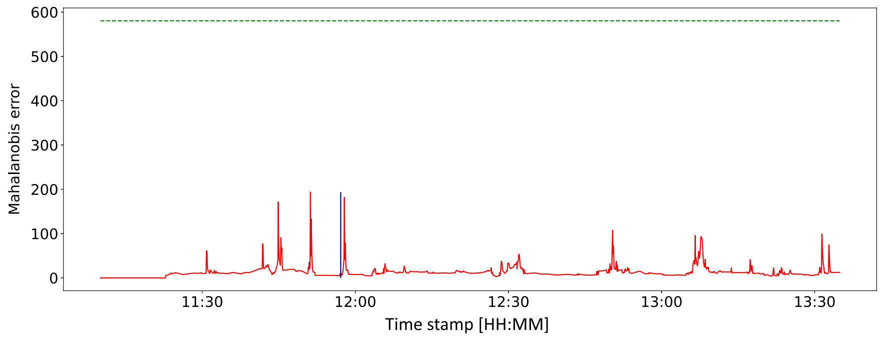


Fig. 8 Raw anomaly detection signal (red) of the LSTM-based version 1 model for the 2022/06/03 abnormal dataset with Mahalanobis threshold (green). (Start of anomaly is indicated with vertical blue line.)

anomaly detection algorithm with the LSTM-based model also performed poorly on other abnormal datasets.

Nevertheless, when the out-of-sample test with the 2022/05/30 normal dataset kept aside was excluded from the Mahalanobis error threshold determination procedure of Sec. III.E (i.e., when only the out-of-sample test with the 2022/05/30 dataset was not considered but the dataset itself was used for training in all other out-of-sample tests), the performance significantly improved and was comparable with the results of the CNN-based model. Yet, although the LSTM-based version 1 model has a larger number of trainable parameters, the results were not as good as in the case of the CNN-based model. We present the quantitative evaluation of the anomaly detection algorithm employing the LSTM-based version 1 model with the Mahalanobis error threshold value determined without considering the out-of-sample test with the 2022/05/30 normal dataset in Supplemental Table S1 in Supplemental Appendix S.A.

Given the preceding observations, we henceforth focus on the CNN-based model architecture. Our experiments with different sizes of the window with past data points for inference (i.e., 100, 150, and 200) reveal that the best performance is achieved with the window size of 150. We present the results obtained for the different window sizes in Supplemental Table S2 in Supplemental Appendix S.A. From this point onward, we set the window size to 150 in all experiments. The CNN-based model is trained as described in Sec. III. Additionally, the trained model is validated with the validation dataset, which is a normal condition dataset (see Sec. II). The results of the anomaly detection algorithm run on the validation dataset are presented in Fig. 9.

The quantitative evaluation of the anomaly detection algorithm employing the CNN-based model and the Mahalanobis error threshold value determined with the approach of Sec. III.E is provided in Table 3, covering the anomalous experiments from Table 1. In Supplemental Appendix S.C, we provide the plots for the individual abnormal datasets of the raw anomaly detection signal and the postprocessed anomaly detection signal, which is the final output of the proposed anomaly detection algorithm, in Supplemental Fig. S8.

Next, the precision–recall curves (PRCs) were generated with TPs, FPs, and FNs determined as described earlier in this paper for different values of the Mahalanobis error threshold, and the areas under the PRCs were computed. The curves with the areas and the no-skill lines indicating the proportion of the number of anomalies to the total number of data points for individual abnormal datasets are presented in Fig. 10. Because the anomaly detection domain is highly skewed, precision–recall curves are more informative than receiver operating

Table 3 Quantitative evaluation results of CNN-based model with window size of 150 data points for the individual abnormal datasets[a]

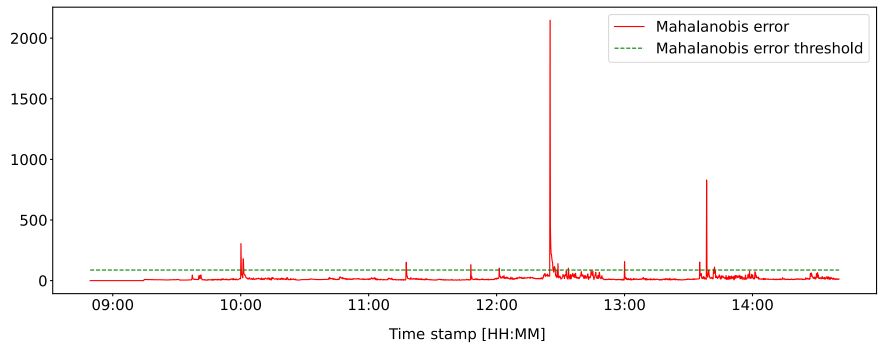
Experiment date	TP	FP	FN	Precision	Recall	F1 score
2022/04/06	2(+1)	0 (0)	1 (−1)	1.00	0.67	0.80
2022/05/18	1 (−1)	2 (+2)	2 (+1)	0.33	0.33	0.33
2022/05/20	1 (+1)	0 (0)	0 (−1)	1.00	1.00	1.00
2022/05/30	1 (+1)	0 (0)	0 (−1)	1.00	1.00	1.00
2022/06/01	3 (+3)	9 (+8)	1 (−3)	0.25	0.75	0.38
2022/06/03	1 (+1)	2 (+2)	0 (−1)	0.33	1.00	0.50
2022/06/08	2 (+2)	2 (+2)	0 (−2)	0.50	1.00	0.67
2022/06/15	2 (0)	1 (+1)	0 (0)	0.67	1.00	0.80
2022/06/22	1 (0)	0 (0)	0 (0)	1.00	1.00	1.00
2022/07/20	4 (+2)	1 2 (+1 2)	0 (−2)	0.25	1.00	0.40

[a]The numbers in the parentheses indicate the increase/decrease in the number of TPs, FPs, and FNs with respect to the benchmark rule-based solution results shown in Table 5.

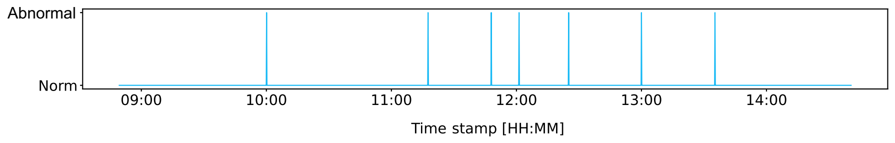
characteristic (ROC) curves because the latter may provide an excessively optimistic evaluation of the performance [16]. Indeed, due to the highly dominating numbers of TNs, the areas under the ROC curves are very close to one for all abnormal datasets (data not shown). Therefore, ROC curves in this case do not reveal the actual performance of our anomaly detection algorithm.

We once again return to the problem of selecting the target deep-learning model for our anomaly detection algorithm. Despite the poor performance of the LSTM-based version 1 model in Fig. 6b, one could argue that lowering the threshold in Fig. 8 could lead to good performance. To verify this, we computed the areas under the precision–recall curves for the anomaly detection algorithm with the LSTM-based model for individual abnormal datasets. The areas obtained with the LSTM-based version 1 model and the CNN-based model are shown in Table 4. For completeness, we also include the results for the LSTM-based version 2 model and the autoencoder model presented in Supplemental Figs. S1a and S1b of Supplemental Appendix S.A, respectively. These results provide further justification for selecting the CNN-based model as the target model for our anomaly detection algorithm. Additional experiments in which other variants of the LSTM-based models and the autoencoder with different numbers, shapes, and activation functions of LSTM layers were considered confirmed our choice (data not shown).

There is a significant variance in the values of the precision, recall, and F1-score metrics for different datasets. This is mainly due to the

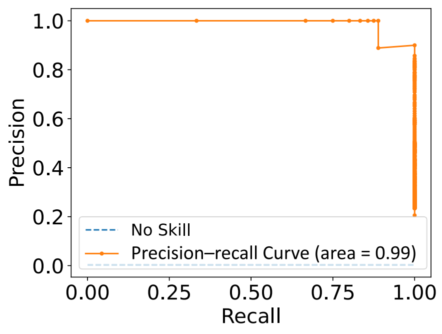


a) The Mahalanobis error (i.e.”, change “distance,” to “distance) the inference error measured with the Mahalanobis distance, on the validation dataset

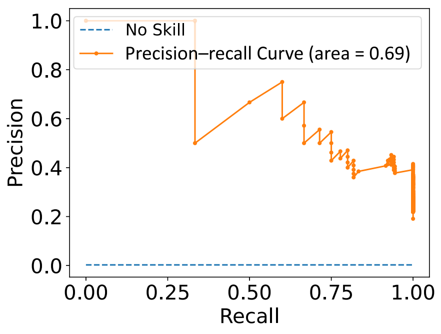


b) The output of the anomaly detection algorithm on the validation dataset. False Positives: 7, False Negatives: 0, True Positives: 0, and True Negatives: 1925

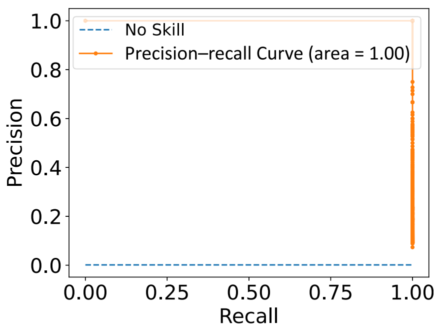
Fig. 9 Validation results of the anomaly detection algorithm employing the CNN-based model in Fig. 6a with window size set to 150.



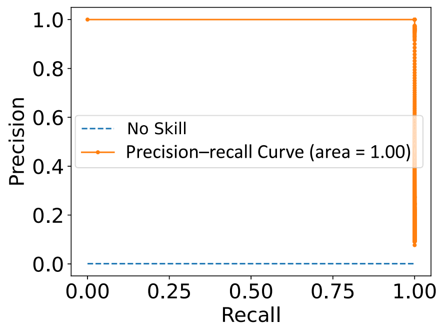
a) Abnormal dataset of 2022/04/06



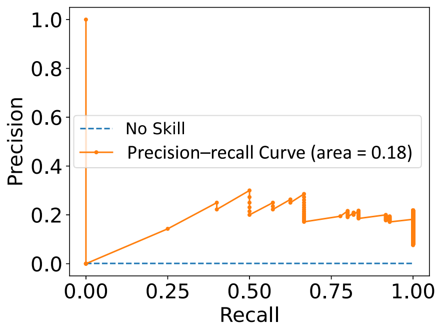
b) Abnormal dataset of 2022/05/18



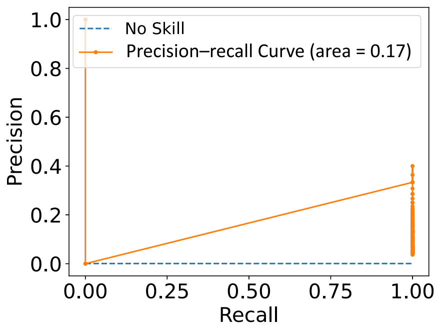
c) Abnormal dataset of 2022/05/20



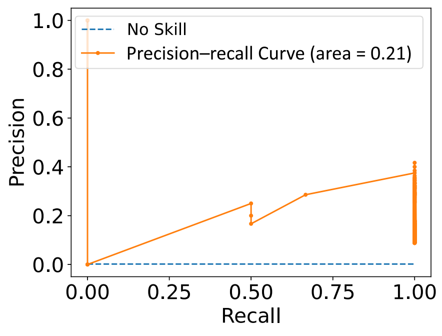
d) Abnormal dataset of 2022/05/30



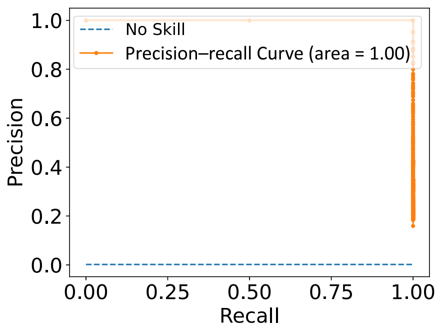
e) Abnormal dataset of 2022/06/01



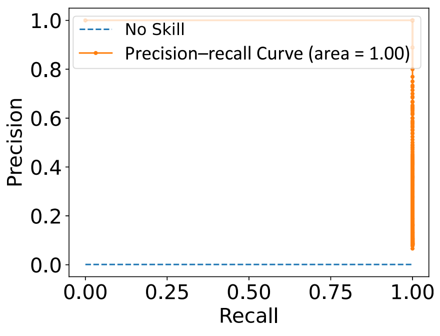
f) Abnormal dataset of 2022/06/03



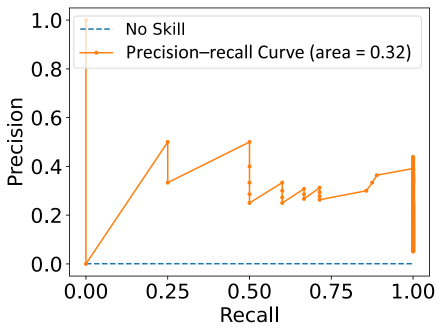
g) Abnormal dataset of 2022/06/08



h) Abnormal dataset of 2022/06/15



i) Abnormal dataset of 2022/06/22



j) Abnormal dataset of 2022/07/20

Fig. 10 Precision–recall curves obtained with the CNN-based model (Fig. 6a) for individual abnormal datasets.

Table 4 Areas under the precision-recall curves (PRCs) of the anomaly detection algorithm run on individual abnormal datasets with different models considered in this study

	Experiment date[a]									
	04/06	05/18	05/20	05/30	06/01	06/03	06/08	06/15	06/22	07/20
CNN PRC area	0.99	0.69	1.00	1.00	0.18	0.17	0.21	1.00	1.00	0.32
LSTM version 1 PRC area	0.90	0.57	1.00	1.00	0.04	0.25	0.16	1.00	1.00	0.30
LSTM version 2 PRC area	0.99	0.42	1.00	1.00	0.08	0.00	0.12	1.00	0.01	0.07
Autoencoder PRC area	0.11	0.80	1.00	0.00	0.03	0.00	0.02	0.99	0.06	0.15

[a]A shorter notation is used to denote the datasets, e.g., 04/06 stands for 2022/04/06.

high level of noise and the fact that the individual experiments were conducted on different days in the laboratory. Although we made all effort to eliminate any external factors that could distort the generated data, given our highly limited experimental laboratory resources and conditions, the external factors still impacted the data. For example, as can be observed by comparing the room temperature logs from different experiments, the base room temperature was changing from one day to another. Because of our efforts, the differences were not very large. Still, they were inevitably impacting the temperature patterns recorded by the EduSat board temperature sensors. Given the 1°C resolution of the sensors, a change in the base room temperature generated differences in the temperature ranges of the profiles recorded by the individual sensors and, even more importantly in the context of our solution, in the timings between temperature changes registered by the individual sensors. Specifically, the latter had a direct and significant impact on the values of the interpolated ISLC features.

The presented results can be reproduced by running the *training/run_saved_models.ipynb* Jupyter notebook in the AtMonSat project GitHub repository [7]. The notebook loads a selected trained model and runs it on the validation and anomalous datasets. The saved and trained deep-learning models considered in this study are made available in the *training/saved_models* folder in the repository.

B. Comparison to a Rule-Based Benchmark Approach

We compared the performance of our proposed AI-based solution with a classical approach to health monitoring in the space industry, i.e., a rule-based approach. For this purpose, we developed the following benchmark procedure. As argued in the previous subsection, there is an inherent inertia before the excessive heat due to malfunctioning of a component reaches the sensors. Furthermore, the delay may vary for individual sensors on the board, which is caused by the differences in their distances to the malfunctioning component. With this in mind, as in the case of TP intervals, we considered the same anomaly inertia window of length l_{aiw}. For a given data point at index i, if all nine sensors recorded a change in temperature within data points of indices in $[i, i + l_{\mathrm{aiw}}]$, then this data point was considered positive; otherwise, it was considered negative. This classification was performed for all data points in the abnormal datasets. The benchmark approach raw classification of data points of the 2022/05/18 abnormal dataset is shown in Fig. 11.

Next, contiguous subsequences of positive data points were considered, which are referred to as *positive intervals*. If a subsequence was longer than $2l_{\mathrm{aiw}}$, then it was split into consecutive positive intervals of length $2l_{\mathrm{aiw}}$, with the last one possibly being shorter. In the benchmark approach, we defined an anomaly to be contained within a positive interval if, and only if, the start time of the anomaly was within the time range of the positive interval given by the time stamps of its data points. An anomaly was considered to be correctly detected, and increased the counter of TPs, if it was contained within some positive interval. If the anomaly was not contained in any of the positive intervals, the anomaly increased the counter of FNs. Finally, all positive intervals within which no anomaly was contained increased the counter of FPs. The results of applying the benchmark approach to the abnormal datasets of the experiments in Table 1 are shown in Table 5. By comparing the results to the ones in Table 3, one can see that our algorithm outperforms the benchmark solution on seven out of 10 datasets in terms of the number of TPs and FNs. On another two datasets, the results are the same; in one case of the 2022/05/18 dataset, the results are worse. We just mention here that the 2022/05/18 dataset seems to be much more noisy in comparison to all the other datasets because the performance of all other models considered in our experiments was consistently significantly worse on the 2022/05/18 dataset (data not shown). However, our approach generates higher numbers of FPs on six out of 10 datasets. Nevertheless, the numbers are small in most of the cases, and only the 2022/06/01 and 2022/07/20 datasets are relatively high, namely, 9 and 12, respectively. All in all, we can conclude that our anomaly detection algorithm outperforms the benchmark solution in almost all cases. These results make a case for constructing more complex AI-based frameworks for health monitoring of CubeSats.

Table 5 Quantitative evaluation results of the rule-based benchmark solution for the individual abnormal datasets

Experiment date	TP	FP	FN	Precision	Recall	F1 score
2022/04/06	1	0	2	1.00	0.33	0.50
2022/05/18	2	0	1	1.00	0.67	0.80
2022/05/20	0	0	1	0.00	0.00	0.00
2022/05/30	0	0	1	0.00	0.00	0.00
2022/06/01	0	1	4	0.00	0.00	0.00
2022/06/03	0	0	1	0.00	0.00	0.00
2022/06/08	0	0	2	0.00	0.00	0.00
2022/06/15	2	0	0	1.00	1.00	1.00
2022/06/22	1	0	0	1.00	1.00	1.00
2022/07/20	2	0	2	1.00	0.50	0.67

V. Performance Evaluation of the Microcontroller Implementation

In this section, the anomaly detection algorithm employing a pretrained CNN-based model in Fig. 6a is evaluated in terms of the following three criteria: execution time, memory usage, and energy consumption. We also draw attention in the following to key aspects of our implementation: notably, how interpolation was realized using buffered features.

A. Implementation for a Microcontroller

The anomaly detection algorithm implemented in C++ is embedded in a testbed, as depicted in the block diagram shown in Fig. 12, to allow the evaluation of the algorithm's performance. The algorithm is evaluated on the arrival of each new data point. For the computation of the interpolated ISLC features, data points are entered into the *interpolating iterations since last change counter* (IISLCC) module, which implements the computation of the interpolated ISLC features. The interpolated ISLC features are forwarded into two serial-in/parallel-out (SIPO) blocks, for which the outputs are two windows of raw temperature sensors and interpolated ISLC features, respectively. The two windows are given as input to the pretrained deep-learning model.

The microcontroller implementation of the deep-learning model is not quantized. Attempts to perform post-training quantization or to run quantization-aware training resulted in poor performance of the models. Therefore, all the results presented here are obtained with the deep-learning model implemented with float32 (single-precision floating-point number representation occupying 32 bits in computer memory). Further technical details on the testbed and the algorithm implementations are provided in Supplemental Appendix S.B.

B. Anomaly Detection Algorithm Execution Time Measurements

We performed detailed measurements of the execution times of the anomaly detection algorithm run on the microcontroller with different settings of the clock frequency. We indicate the exact part of the anomaly detection algorithm for which execution was considered in the measurements with the *measured time* frame in Fig. 12.

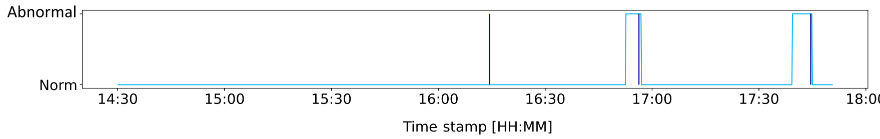


Fig. 11 Benchmark approach raw classification of the 2022/05/18 abnormal dataset data points (cyan) with indicated anomaly start times (dark blue).

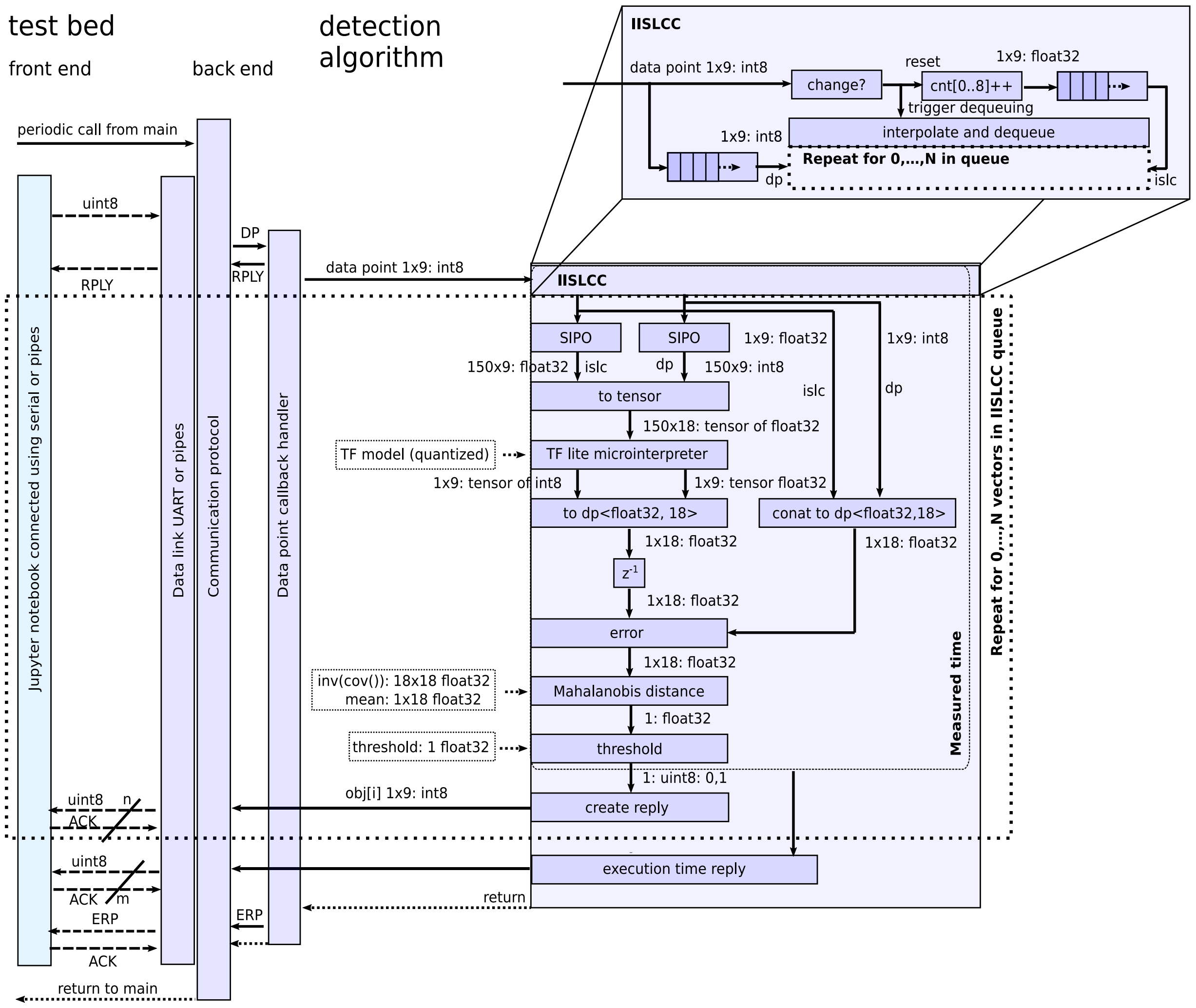


Fig. 12 Flow diagram of the STM32H743 microcontroller implementation. (lite: is the second term of the product name "Tensorflow lite", a mobile library for deploying models on mobile, microcontrollers and other edge devices; int: abbreviation for integer; uint: abbreviation for unsigned integer; float: abbreviation of floating point number; float32: abbreviation of floating point number stored in with a 32 bit (bit is an abbreviation for binary digit); cnt: abbreviation for count; dp: abbreviation for data point; rply: abbreviation for reply; erp: abbreviation for end-of-reply; uart: abbreviation for Universal Asynchronous Receiver/Transmitter; ack: abbreviation for acknowledge; TF: abbreviation for Tensorflow; concat: abbreviation for the english word concatenate.)

To collect larger samples of measurements, we run the anomaly detection algorithm on the microcontroller on data generated both in Phase II and Phase III of the individual experiments, i.e., on merged both normal and abnormal datasets of the individual experiments. We refer to the merged datasets as the *experimental dataset* of the given experiment. The execution times were determined for the processing of each data point of the individual experimental datasets. For example, the execution times for each data point of the 2022/04/06 experimental dataset processed on the microcontroller with the clock frequency set to 298 MHz are shown in Supplemental Fig. S7 in Supplemental Appendix S.C.

Notice that for some data points, the execution times are close to zero. These times are just needed for queuing the data points without computation of the interpolated ISLC features and without the execution of the core part of the anomaly detection algorithm, i.e., the parts shown inside the *repeat for 0, . . . ,N vectors in IISLCC queue* and *repeat for 0, . . . ,N in queue* frames in Fig. 12. The processing of these points is postponed until the moment when interpolated ISLC feature values can be computed, i.e., when a temperature change by any of the temperature sensors is detected at the arrival of some future data point. Once the temperature change is observed, the interpolated ISLC features are computed and the anomaly detection algorithm is run for all the queued data points; see Supplemental Appendix S.B.C.1 for details. Therefore, the subsequences of near-zero execution times are followed by execution time peaks associated with data points that resulted in change detection and dequeuing. The peaks reflect the processing of all the queued data points.

The mean execution times are shown in Figs. 13 and 14. Notice that the cache optimization was enabled both for the data and

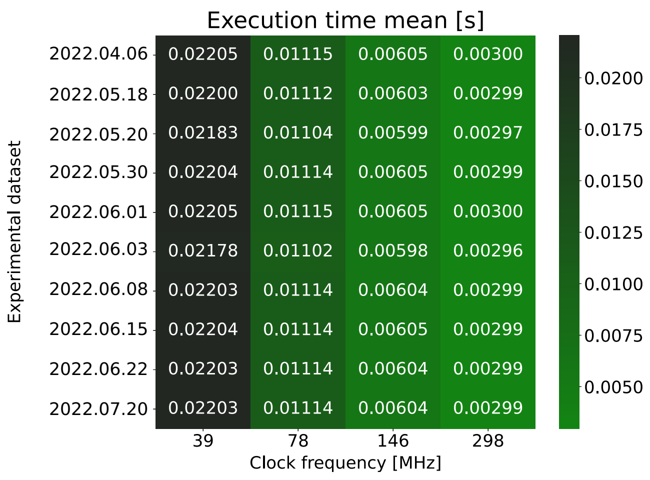


Fig. 13 Mean execution times (in seconds) of the anomaly detection algorithm run with four different settings of the microcontroller's clock frequency.

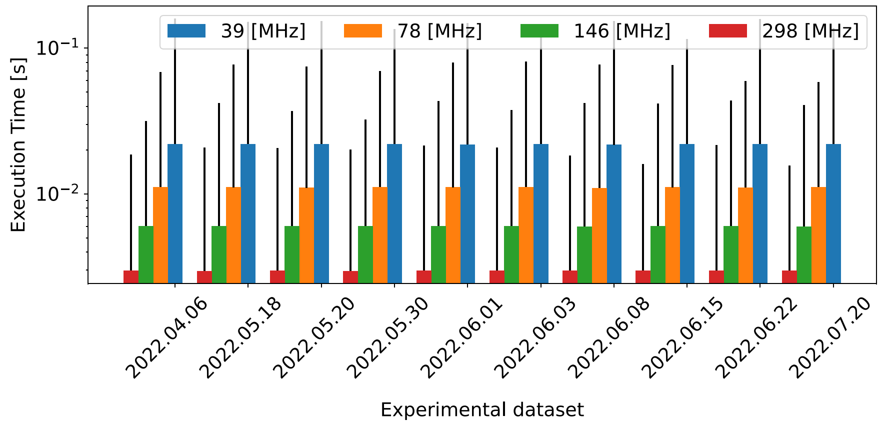


Fig. 14 Anomaly detection algorithm mean execution times with standard deviations (vertical black lines) for four settings of the microcontroller's clock frequency.

instructions. Observe that there is little variation in performance between datasets, indicating that the algorithm is robust. Detailed histograms with anomaly detection algorithm execution times for individual experiments for four different settings of the microcontroller's clock frequency (i.e., 39, 78, 146, and 298 MHz) are provided in Supplemental Table S3 in Supplemental Appendix S.C.

C. Power Consumption Measurements

The algorithm is embedded in a testbed that allows the verification of the results and performance, as detailed in Supplemental Appendix S.B. We measured the power consumed by the microcontroller while running the testbed with the anomaly detection algorithm on individual experimental datasets under different settings of clock frequency. Each measurement consisted of three phases: 1) the microcontroller running in idle state, 2) the microcontroller running the testbed with the anomaly detection algorithm on an experimental dataset, and 3) the microcontroller again in idle state. The average of the instantaneous power over the processing of one full experimental dataset (i.e., over the second phase) is referred to as the *experiment mean power*. The numerical values of the experiment mean powers in watts for individual experimental datasets under different settings of clock frequency are provided in Fig. 15. The detailed plots presenting the instantaneous power consumptions are provided in Supplemental Table S4 of Supplemental Appendix S.C. Notice that a smaller frequency results in less power consumption. Nevertheless, there is a tradeoff between power consumption and the execution time; see Figs. 13 and 14. Therefore, the clock frequency should be set in a way that simultaneously assures an acceptable power consumption level and getting calculations done before the next data point sample arrives.

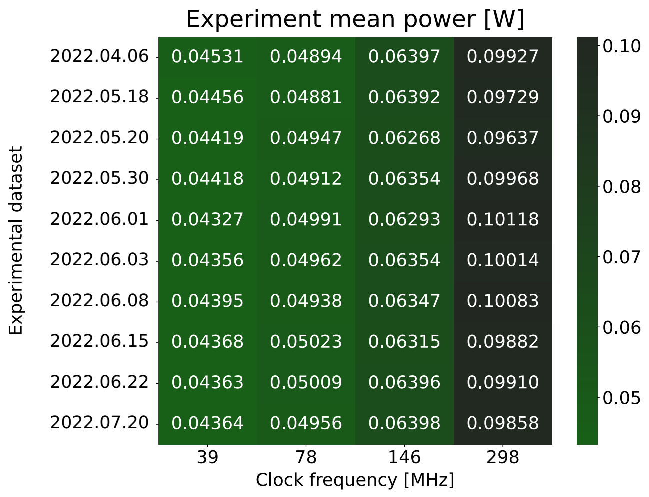
Measurements of *background power* (i.e., the idle state instantaneous power consumption) are included for reference and for the calculation of the *net experiment mean power* consumed by the testbed with the anomaly detection algorithm. The net experiment mean power is calculated as the experiment mean power minus the background mean power, i.e., the average of background power measurements. The net experiment mean power consumptions with the standard deviations calculated in accordance with the expression for the standard deviation of a sum/difference of two uncorrelated random variables (i.e., $\sigma_{X-Y} = \sqrt{\sigma_X^2 + \sigma_Y^2}$) are shown in Fig. 16.

D. Memory Footprint

The implementation of the anomaly detection algorithm used both static memory and stack. To prevent fragmentation, no dynamic memory (heap) was used.

To measure the stack and static memory used by the anomaly detection algorithm, a placebo version of the firmware was implemented. This version contained the complete testbed without the algorithm itself.

The net stack and static memory usage were obtained by subtracting the stack and static memory used by the placebo version from the stack and static memory used by the complete version (i.e., the



	39	78	146	298
2022.04.06	0.04531	0.04894	0.06397	0.09927
2022.05.18	0.04456	0.04881	0.06392	0.09729
2022.05.20	0.04419	0.04947	0.06268	0.09637
2022.05.30	0.04418	0.04912	0.06354	0.09968
2022.06.01	0.04327	0.04991	0.06293	0.10118
2022.06.03	0.04356	0.04962	0.06354	0.10014
2022.06.08	0.04395	0.04938	0.06347	0.10083
2022.06.15	0.04368	0.05023	0.06315	0.09882
2022.06.22	0.04363	0.05009	0.06396	0.09910
2022.07.20	0.04364	0.04956	0.06398	0.09858

Fig. 15 Experiment mean power consumed by the microcontroller while running the test bed.

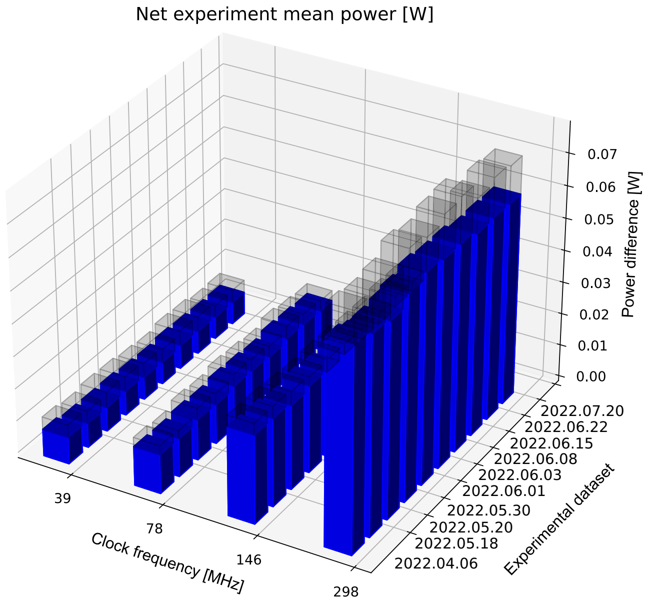


Fig. 16 Net experiment mean power (blue bars) with standard deviation (gray bars) consumed by the microcontroller while running the test bed.

Table 6 Static memory usage of the complete and placebo firmware compiled for the microcontroller

Microcontroller firmware	flash (microcontroller's flash memory)		SRAM (static random-access memory)		
	Code + constants		Data + bss	Stack	Total usage
Complete	297.67 kB	14.53%	51.06 kB	2.77 kB	10.51%
Placebo	127.83 kB	8.44%	16.32 kB	0.58 kB	3.30%
Algorithm (complete-placebo)	168.84 kB	6.09%	34.74 kB	2.19 kB	7.21%

testbed with the algorithm), respectively. The results are provided in Table 6.

The data, bss (block starting symbol – a portion of memory that contains statically allocated variables that have no predefined value), and stack were stored in the memory AXI-SRAM (static random-access memory connected to the core using an AXI interface (Advanced eXtensible Interface, an on-chip communication bus protocol developed by ARM (Advanced RISC Machine))) for domain D1 (512 kB). The code and constants were stored in the microcontroller's flash memory (2 MB). More details on the microcontroller's memory organization can be found in Ref. [17].

VI. Future Work

Regarding future work, a key limitation of the current method was the quantity of data generated and environmental factors that differed between the laboratory and space. This was partly due to what could be achieved with the resources available using EduSat. Nevertheless, the current solution provided effective datasets for model training, evaluation, and testing, even with the noisy data. Rather than obtaining more of the same data, the current promising results make a case for future work acquiring higher-quality laboratory data, or even real in-orbit data (transmitted during payload downtime perhaps), and the fine-tuning of the current solution based on them. For the current datasets, quantization introduced a significant drop in performance. Tantalizingly, quantization was effective on simulated data with no noise, and conditions in space may be between these two extremes. Therefore, the problem of model quantization is another issue that could be further investigated with the aim of producing a solution requiring even less memory and computational power. Other lines of future work include testing the robustness of the solution with respect to different novel anomalies and determining whether additional telemetry could contribute to the detection of certain anomalies. Finally, the observed performance variations of the current approach could be reduced by acquiring real in-orbit data. In-orbit data would eliminate noise introduced by factors that could not be eliminated in the current laboratory experiments, which did not occur in space.

VII. Conclusions

The problem addressed in this work, of onboard fault detection using artificial neural networks that respect the limited resources of a CubeSat system, is reflected on here. The fact that this work shows that computational resources can indeed be respected by the algorithm is a key novelty of the approach proposed, since memory usage of the algorithm was kept within 10% of the memory available, as summarized in Table 6. This supports the argument that additional hardware, such as a powerful dedicated processor, is not required to deploy the solution proposed. Figure 13 suggests that even if clock cycles are limited (e.g., due to scheduling between our algorithm and existing control software deployed on a microcontroller), the performance of the algorithm is well within the bounds of real-time execution because the execution time is far less than the rate at which data arrives, with negligible variation. The power measurements provided in Fig. 16 can be used to estimate the power overhead, depending on the requirements of a particular mission.

To both respect resources and reliably detect anomalies, certain aspects of the current algorithm were designed and evaluated carefully: notably, the deep-learning architecture. Of the deep-learning architectures in Fig. 6 and Supplemental Fig. S1 in Supplemental Appendix S.A., only the CNN showed promising results. The results in Table 4, comparing the area under precision–recall curves, provided a clear indicator that the CNN outperformed the other models. For four datasets where anomaly detection was near perfect both the CNN and the LSTM version 1 models performed well, whereas in the other experiments where noise reduced the effectiveness of anomaly detection, the CNN consistently outperformed the LSTM version 1 model. Additional experiments on simulated data reinforced this observation that LSTMs perform well without noise. Hence, the role of normal background noise in data collected in space as compared to the laboratory data in Table 1 could be evaluated further. A threshold was calculated according to a statistical significance level of 0.05 in Fig. 7, which may of course be varied, depending on the desired anomaly reporting rate of a mission. The precision and recall for CNNs at the statistical significance level of 0.05 are presented in Table 3, which can be compared to Table 5 to see that the current architecture outperforms a benchmark OOL method. It was found that precision–recall curves provided a better evaluation than ROC curves because the precision–recall curves excluded the true negatives dominating the dataset.

We were able to keep the number of parameters comprising our CNNs small, largely due to carefully selecting features. The algorithm for selecting features, described in Sec. III.B, accounts for values of the nine temperature sensors (shown in Fig. 1) as well as the interpolated number of data points since the last discrete change in value for each sensor, as illustrated in Table 2. In the implementation in Fig. 12, a dedicated delay buffers data points until one sensor among all sensors changes to realize interpolation in real time on a microcontroller. We consider the role of these features for reducing the size of a neural network, while retaining anomaly detection effectiveness in real time, to be a significant contribution of this paper. Simpler choices of features (e.g., without interpolation) simply did not yield results, except on synthetic data without noise.

Acknowledgments

More than 50% of the funding was provided in kind by the Department of Computer Science in the Faculty of Science Technology and Medicine at the University of Luxembourg. The research presented in this work was conducted within the AtMonSat project, which was partly funded by the European Space Agency (ESA) under the Open Space Innovation Platform; under which, the ESA organizes and launches campaigns and channels to seek ideas related to space research, which is otherwise known as ESAIdeas. For reference, this project relates to ESA contract no. 400134792/21/NL/GLC/my and ESA activity description no. I-2020-03332. We thank Luis Mansilla and Christophe Honvault (the Technical Representative of the ESA) at the Software Technology Section of the ESA European Space Research and Technology Centre (ESTEC). Their monthly feedback helped guide the AtMonSat project toward relevant outcomes.

References

[1] "CubeSat Design Specification (CDS) Rev. 13," The CubeSat Program, California Polytechnic State Univ., San Luis Obispo, CA, 2015.

[2] "6U CubeSat Design Specification Revision 1.0," The CubeSat Program, California Polytechnic State Univ., TR CP-6UCDS-1.0, San Luis Obispo, CA, 2016.

[3] Stesina, F., and Corpino, S., "Investigation of a CubeSat in Orbit Anomaly Through Verification on Ground," *MDPI Aerospace*, Vol. 7, No. 4, 2020, Paper 38. https://doi.org/10.3390/aerospace7040038

[4] Crepaldi, M., Horne, R., and Mauw, S., "Software Certification as a Limit on Liability: The Case of Cubesat Operations," *Space Law in a Networked World*, edited by P. J. Blount, and M. Hofmann, Brill, Leiden, The Netherlands, 2023, pp. 162–186. Chap. 7. https://doi.org/10.1163/9789004527270_008

[5] "Space Systems—Cube Satellites (CubeSats)," International Organization for Standardization STD 17770:2017, Geneva, Switzerland, 2017.

[6] De Claville Christiansen, J., "CubeSat Space Protocol (CSP): Network-Layer Delivery Protocol for CubeSats and Embedded Systems," GOM-Space ApS TR GS-CSP-1.1, Aalborg, Denmark, 2011.

[7] Stemper, A., "C++ Implementation of the AtMonSat Anomaly Detection Algorithm," *GitHub* [Online Database], 2022, https://github.com/andre-stemper/ATMonSAT.git [retrieved 26 July 2023].

[8] Yairi, T., Takeishi, N., Oda, T., Nakajima, Y., Nishimura, N., and Takata, N., "A Data-Driven Health Monitoring Method for Satellite Housekeeping Data Based on Probabilistic Clustering and Dimensionality Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 53, No. 3, 2017, pp. 1384–1401.
https://doi.org/10.1109/TAES.2017.2671247

[9] Martínez-Heras, J.-A., Donati, A., Kirsch, M. G. F., and Schmidt, F., "New Telemetry Monitoring Paradigm with Novelty Detection," *SpaceOps 2012 Conference*, AIAA, Reston, VA, 2013.
https://doi.org/10.2514/6.2012-1275123

[10] "ECSS-E-ST-70-41C—Telemetry and Telecommand Packet Utilization," European Cooperation for Space Standardization, 2016, https://ecss.nl/standard/ecss-e-st-70-41c-space-engineering-telemetry-and-telecommand-packet-utilization-15-april-2016/.

[11] Meß, J.-G., Dannemann, F., and Greif, F., "Techniques of Artificial Intelligence for Space Applications—A Survey," *European Workshop on On-Board Data Processing (OBDP2019)*, European Space Agency, The Netherlands, 2019, https://indico.esa.int/event/225/contributions/4289/.

[12] Martínez-Heras, J., and Donati, A., "Enhanced Telemetry Monitoring with Novelty Detection," *AI Magazine*, Vol. 35, No. 4, 2014, pp. 37–46.
https://doi.org/10.1609/aimag.v35i4.2553

[13] Tipaldi, M., Feruglio, L., Denis, P., and D'Angelo, G., "On Applying AI-Driven Flight Data Analysis for Operational Spacecraft Model-Based Diagnostics," *Annual Reviews in Control*, Vol. 49, Jan. 2020, pp. 197–211.
https://doi.org/10.1016/j.arcontrol.2020.04.012

[14] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R., "Deep Learning and its Applications to Machine Health Monitoring," *Mechanical Systems and Signal Processing*, Vol. 115, Jan. 2019, pp. 213–237.
https://doi.org/10.1016/j.ymssp.2018.05.050

[15] Weiss, G. M., "Foundations of Imbalanced Learning," *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st ed., edited by Y. Ma, and H. He, Wiley, Hoboken, NJ, 2013, pp. 13–41, Chap. 2.

[16] Branco, P., Torgo, L., and Ribeiro, R. P., "A Survey of Predictive Modeling on Imbalanced Domains," *ACM Computing Surveys*, Vol. 49, No. 2, 2017, pp. 1–50.
https://doi.org/10.1145/2907070

[17] "STM32H742xI/G STM32H743xI/G," STMicroelectronics datasheet, Geneva, Switzerland, 2022, https://www.st.com/resource/en/datasheet/stm32h743vi.pdf [retrieved 25 Oct. 2022].

M. J. Kochenderfer
Associate Editor